ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

ΣΧΕΔΙΑΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΩΝ ΣΕ ANDROID ΓΙΑ ΔΙΑΧΕΙΡΙΣΗ ΚΛΗΣΕΩΣ ΤΑΧΙ

Πτυχιακή Εργασία

Χατζημιχαήλ Γεώργιος (2574)

Τσεγγελίδης Φίλανδρος (2441)

Επιβλέπων: Δρ. Α.Τσιμπίρης , Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ , ΜΑΙΟΣ 2013

Υπεύθυνη δήλωση

Υπεύθυνη Δήλωση: Βεβαιώνουμε ότι εμείς είμαστε συγγραφείς της παρούσας πτυχιακής εργασίας και όποια βοήθεια χρειάστηκε , είναι πλήρως τεκμηριωμένη και αναφέρεται στις πηγές της πτυχιακής εργασίας. Είναι ένα έργο που προετοιμάστηκε από εμάς με αρκετό κόπο και δουλειά με απώτερο σκοπό την παρουσίαση μιας ενιαίας λύσης (εφαρμογής) για την διευκόλυνση του χρήστη να χρησιμοποιεί υπηρεσίες διαχείρισης κλήσεως taxi. Όλα αυτά στα πλαίσια του προγράμματος σπουδών του τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.

Περίληψη

Τα ταξί έχουν μπει στη ζωή μας εδώ και πολλά χρόνια και μας έχουν διευκολύνει αρκετά. Λίγο πολύ όλοι μας έχουμε πάρει ταξί είτε να το καλέσουμε από το τηλεφωνικό κέντρο είτε να το παραλάβουμε από μια πιάτσα.

Οι συγκεκριμένες εφαρμογές , μας παρέχουν μεγαλύτερη ευελιξία στον τρόπο παραγγελίας ενός ταξί. Δεν χρειάζεται ούτε τηλεφωνική κλήση ούτε να μετακινούμαστε στην πλησιέστερη πιάτσα, αρκεί να έχουμε ένα smartphone με λειτουργικό Android. Έτσι αναζητούμε ταξί σε όποιο μέρος είμαστε, άμεσα και γρήγορα με ένα click.

1 Εισαγωγή

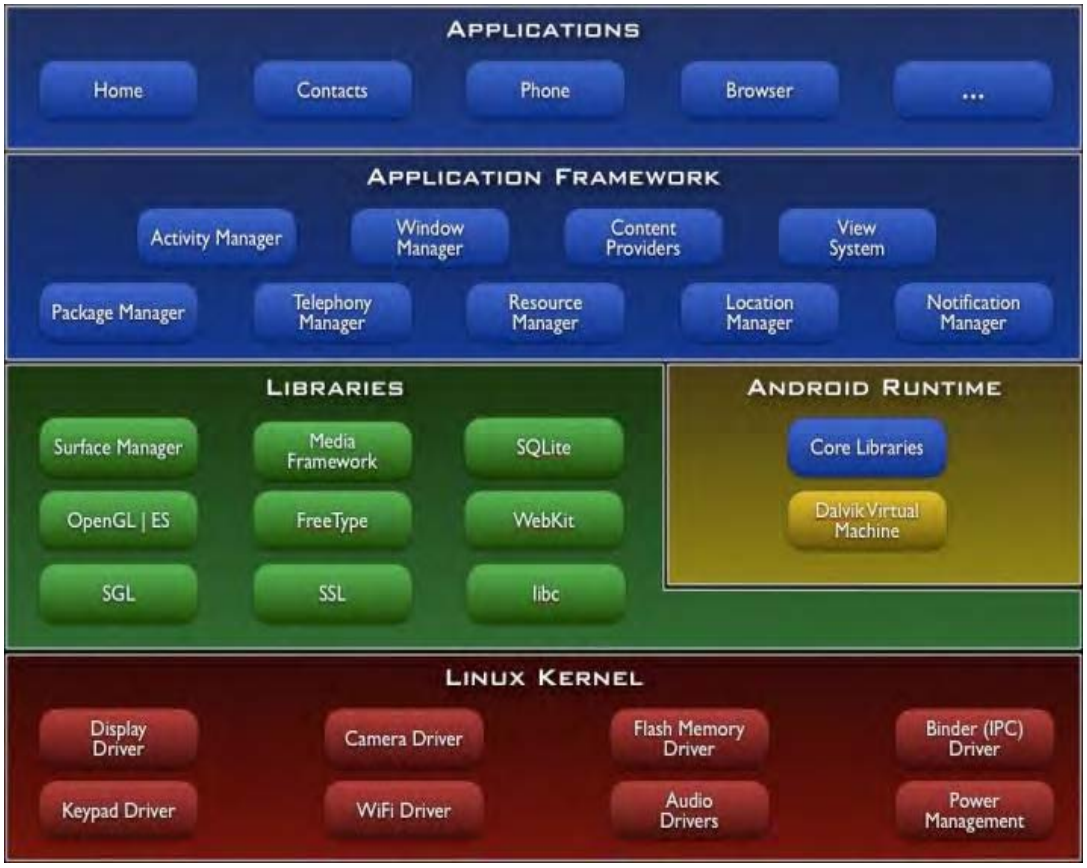
1.1 Βασικές αρχές εφαρμογών Android

1.1.1 Εισαγωγή

Οι εφαρμογές Android γράφονται στη γλώσσα προγραμματισμού Java .Για την ακρίβεια χρησιμοποιείται μια εικονική μηχανή Dalvik virtual machine VM ο οποία βασίζεται σε καταχωρητές και μπορεί να χρησιμοποιεί τις κλάσεις που μεταγλωττίστηκαν από κάποιον java compiler. Το Android SDK (Software Development Kit) αναλαμβάνει την μεταγλώττιση (compilation) του κώδικα και των άλλων resource files και παράγει ένα πακέτο εγκατάστασης με κατάληξη .apk. Με αυτό το αρχείο είναι δυνατή η εγκατάσταση των εφαρμογών στα κινητά με λειτουργικό Android.

Κάθε εφαρμογή που εγκαθίσταται λειτουργεί και "ζει" ανεξάρτητα από την άλλη και έχει το δικό της στιγμιότυπο. Μη ξεχνάμε ότι το λειτουργικό Android είναι ένα multi user Linux σύστημα και κάθε εφαρμογή φαίνεται και σαν ξεχωριστός χρήστης. Εξάλλου η Dalvik VM βελτιστοποιήθηκε για να τρέχει σε πολλά στιγμιότυπα με ελάχιστη χρήση μνήμης. Σε κάθε εφαρμογή ανατίθεται και ένα μοναδικό id βάσει του οποίου αναγνωρίζεται από το λειτουργικό. Φυσικά όπως στα περισσότερα λειτουργικά συστήματα έτσι και εδώ κάθε εφαρμογή έχει και τη δική της ανεξάρτητη διεργασία. Οι διεργασίες αυτές σταματούν να λειτουργούν όταν το λειτουργικό θέλει να εξοικονομήσει πόρους πχ για να ξεκινήσει μια άλλη διεργασία με περισσότερη προτεραιότητα , όπως μια κλήση. Γενικότερα κάθε εφαρμογή έχει πρόσβαση μόνο σε πόρους του συστήματος που χρειάζεται και όχι σε περισσότερους, ως αποτέλεσμα να έχουμε ένα ασφαλές περιβάλλον με σωστή συμπεριφορά και σταθερότητα.

1.1.2 Κύριο μέρος



Σχήμα 1 : Διάγραμμα αρχιτεκτονικής Android

Όπως βλέπουμε γίνεται διαχωρισμός σε επίπεδα (layers). Παρακάτω εξηγούμε για το καθένα

1.1.2.1 Εφαρμογές (Applications)

Κάθε νέα συσκευή που αγοράζουμε στο λειτουργικό android που έχει , υπάρχουν κάποιες ενσωματωμένες εφαρμογές για τη βασική λειτουργία. Παραδείγματα είναι οι επαφές (contacts), ημερολόγιο (calendar), mail , camera κ.α.

1.1.2.2 Εργαλεία εφαρμογών (Application framework)

Ο προγραμματιστής έχει στη διάθεση του τα εργαλεία/υπηρεσίες που του δίνονται για να αναπτύσσει εφαρμογές. Έτσι του παρέχονται :

- ✓ Όψεις (Views)
  Αφορούν καθαρά το user interface. Είναι στην ουσία ο τρόπος που θα εμφανίζονται τα listview,buttons , πλέγματα , mapviews και τα λοιπά components.

- ✓ Παροχείς περιεχομένου (Content Providers)
  Δίνουν τη δυνατότητα να έχουμε πρόσβαση σε κοινά δεδομένα με άλλες εφαρμογές.

- ✓ Διαχειριστής πόρων (Resource Manager)
  Ενσωμάτωση στην εφαρμογή μας κομμάτια που δεν αφορούν κώδικα αλλά εικόνες, αρχεία,strings κτλ.

- ✓ Διαχειριστής ειδοποιήσεων(Notification Manager)
  Μας εμφανίζει προσαρμοσμένες ειδοποιήσεις στην μπάρα ειδοποιήσεων.

- ✓ Διαχειριστή δραστηριοτήτων (Activity Manager)
  Διαχειρίζεται τον κύκλο ζωής των δραστηριοτήτων καθώς και την κατάστασή τους στην στοίβα

1.1.2.3 Βιβλιοθήκες (Libraries)

Το λειτουργικό Android περιέχει ποικίλες βιβλιοθήκες γραμμένες σε C/C++ οι οποίες χρειάζονται για να λειτουργούν τα διάφορα components των εφαρμογών.

Μερικές από αυτές είναι :

- ➢ System C library
  Μια υλοποίηση της C, βελτιστοποιημένη για συσκευές που βασίζονται στο Linux.

- ➢ Media libraries
  Υπάρχει υποστήριξη καθώς και δυνατότητες αναπαραγωγής πολλών μορφών και τύπων τόσο αρχείων ήχου, όσο και βίντεο. (πχ MPEG4,MP3,ACC, JPEG κ.α)
- ➢ Surface Manager
  Διαχειρίζεται την πρόσβαση στα υποσυστήματα απεικόνισης και είναι υπεύθυνη για τη δημιουργία layers 2D, 3D και γενικότερα των γραφικών.

- SQLlite
  Μια σχεσιακή βάση δεδομένων διαθέσιμη για όλες τις εφαρμογές της συσκευής μας. Στην ουσία πρόκειται για τη βάση δεδομένων μας διαφόρων στοιχείων που θέλουμε να κρατάμε και να τα ανακτούμε όταν τα χρειαστούμε.

1.1.2.4 Android Runtime

- Core libraries

  Στην ουσία μας παρέχει όλες τις βασικές βιβλιοθήκες (java) που απαιτούνται για την ολοκληρωμένη λειτουργία.

1.1.2.5 Dalvik Virtual Machine

Όπως αναφέραμε και περιληπτικά παραπάνω , η Dalvik είναι μια εικονική μηχανή που λειτουργεί ως διερμηνέας και εκτελεί αρχεία της μορφής (.dex). Η συγκεκριμένη μορφή είναι βελτιστοποιημένη για καλύτερη δυνατή απόδοση σε συστήματα με χαρτογραφημένη μνήμη. Αυτή η virtual machine βασίζεται σε καταχωρητές και μπορεί να τρέξει κλάσεις που μεταγλωττίστηκαν από κάποιον java compiler .

1.1.2.6 Linux Kernel

Το Android βασίζεται στον πυρήνα του Linux 2.6 για τις βασικές υπηρεσίες του συστήματος , όπως ασφάλεια, διαχείριση κύριας μνήμης, δίκτυο και drivers για τις περιφερειακές συσκευές.

1.2 Συστατικά εφαρμογών Android

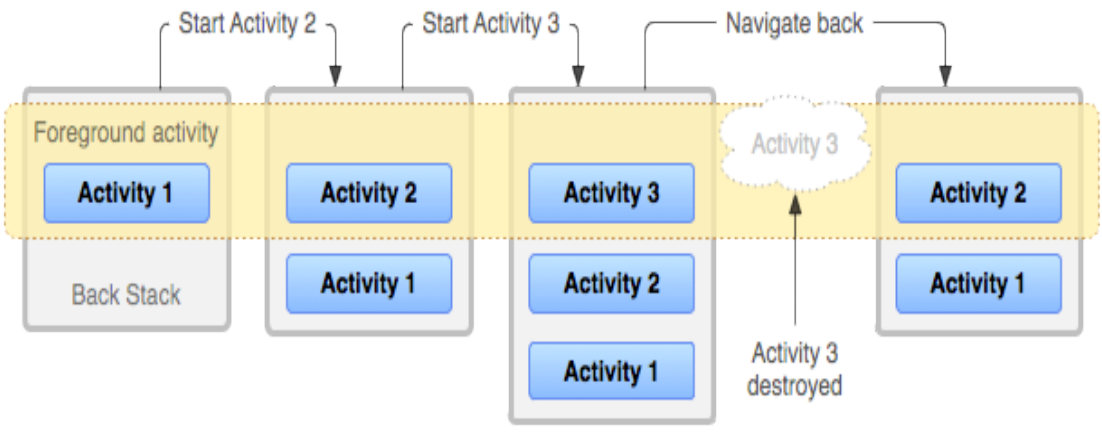
Υπάρχουν 4 ειδών βασικά δομικά συστατικά εφαρμογών Android,

1. **Activities** (Δραστηριότητες)
2. **Services** (Υπηρεσίες)
3. **Content Providers** (Παροχέας περιεχομένου)
4. **Broadcast Receivers**

1.2.1 Κύριο μέρος

1.2.1.1 Activities

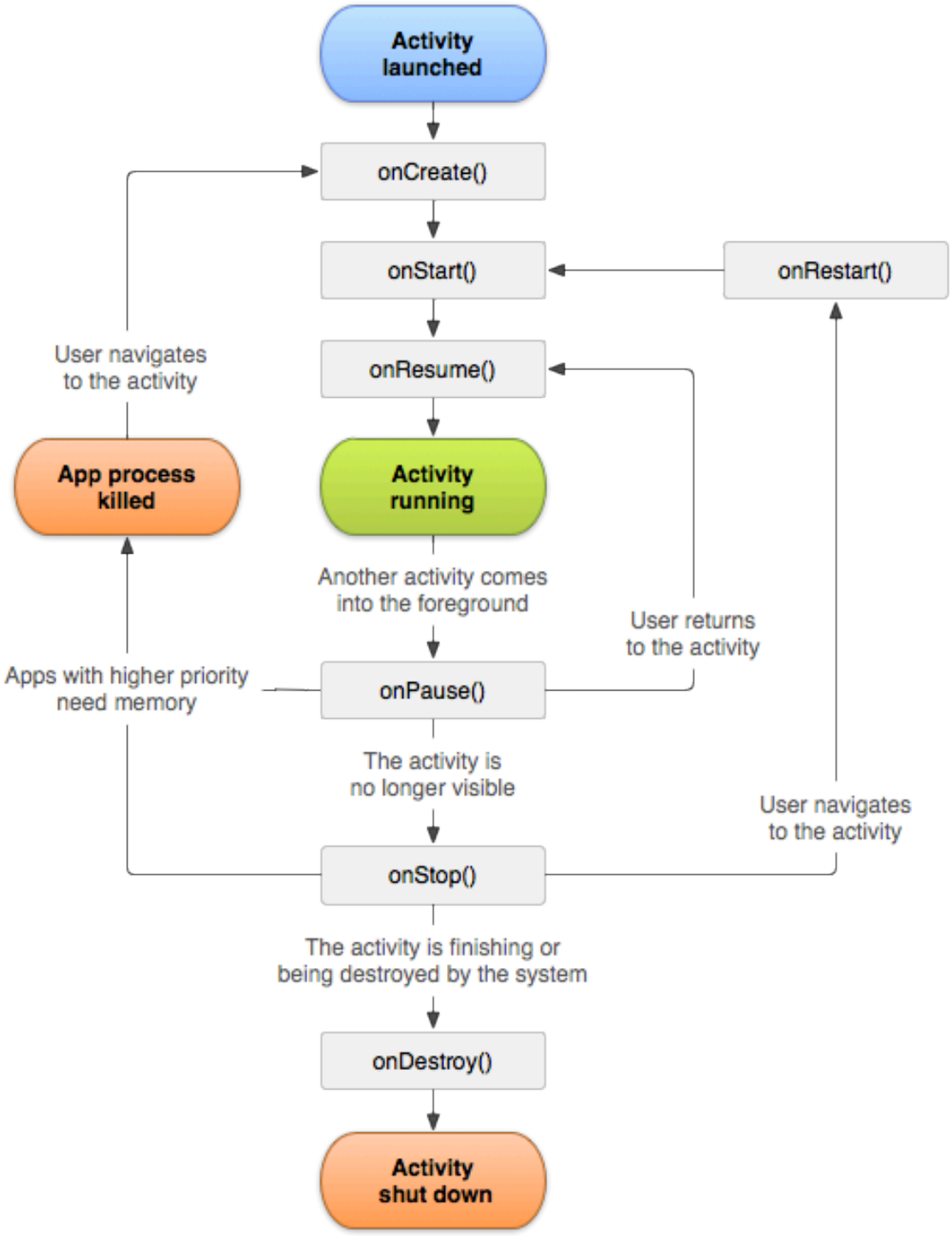
Μια δραστηριότητα αφορά πάντα την οπτική αναπαράσταση της εφαρμογής μας (User interface). Για παράδειγμα στην εφαρμογή μας μια δραστηριότητα είναι η οθόνη με το χάρτη που βλέπει ο πελάτης για να επιλέξει την τοποθεσία που θα γίνει η παραγγελία. Όλες οι εφαρμογές έχουν υποχρεωτικά τουλάχιστον μία δραστηριότητα για να λειτουργήσουν. Μέσα από μια δραστηριότητα μπορούμε να εκκινήσουμε και μια δεύτερη και πάει λέγοντας, αναστέλλοντας την προηγούμενη. Στην ουσία αυτές που αναστέλλονται δεν χάνονται αλλά κρατιούνται σε μια στοίβα (LIFO) δραστηριοτήτων με τις τρέχουσες καταστάσεις τους , ώστε όταν τις ξανακαλέσουμε να συνεχίσουμε από εκεί που ήμασταν.
Στο παρακάτω σχήμα φαίνεται αυτή η διαδικασία στη στοίβα :



Σχήμα 2 : Διάγραμμα στοίβας δραστηριοτήτων

Κάθε δραστηριότητα έχει και ένα κύκλο ζωής , ο οποίος ξεκινά με την δημιουργία της onCreate() και τελειώνει με την καταστροφή της onDestroy() .

Στην επόμενη σελίδα φαίνεται στο διάγραμμα ολόκληρος ο κύκλος ζωής μιας δραστηριότητας τον οποίο και θα εξηγήσουμε :



Σχήμα 3 : Διάγραμμα κύκλου ζωής δραστηριοτήτων

Τρέχουσα μέθοδος	Περιγραφή	Επόμενη μέθοδος
onCreate()	Η πρώτη μέθοδος που φορτώνεται με το που ξεκινάμε μια δραστηριότητα. Εδώ ορίζονται για πρώτη φορά όλα τα αντικείμενα που χρειάζονται με τις κατάλληλες αρχικές τιμές	**onStart()**
onStart()	Καλείται ακριβώς πριν γίνει εμφανίσιμη η δραστηριότητα στον χρήστη. Οπότε προσθέτουμε ότι λειτουργικότητα θέλουμε για αυτή την περίπτωση	**onResume()**
onResume()	Καλείται πριν η δραστηριότητα αρχίζει να αλληλεπιδρά με τον χρήστη. Βρίσκεται στην κορυφή της στοίβας, περιμένοντας για κάποιο user input	**onPause()**
onPause()	Καλείται όταν πάει να γίνει resume μιας άλλης δραστηριότητας. Συνήθης χρήση είναι όταν θέλουμε να αποθηκεύσουμε τις αλλαγές μας για να μην απασχολούμε άσκοπα πόρους cpu	**onResume() ,** αν επιστρέψει στο προσκήνιο αλλίως **onStop()**
onStop()	Καλείται όταν η δραστηριότητα δεν είναι εμφανίσιμη πλέον στο χρήστη. Αυτό συμβαίνει είτε γιατί πρόκειται να τερματιστεί η δραστηριότητα είτε γιατί ξεκινά νέα και την καλύπτει	**OnDestroy()**
OnDestroy()	Καλείται είτε επειδή τελείωσε η εφαρμογή από το χρήστη ηθελημένα , είτε επειδή πρέπει να τερματιστεί βίαια (force closing) για να εξοικονομήσει πόρους συστήματος	**-----**

Πίνακας 1 : Πίνακας κύκλου ζωής δραστηριότητας

1.2.1.2 Services

Είναι άλλο ένα δομικό στοιχείο των εφαρμογών Android. Έχει διαφορές ως προς τον τρόπο λειτουργίας του σε σχέση με τις δραστηριότητες. Πιο συγκεκριμένα τρέχει πάντα στο παρασκήνιο και ποτέ δεν αλληλεπιδρά άμεσα με τον χρήστη , και κατά δεύτερο έχει μεγαλύτερη διάρκεια ζωής από τις δραστηριότητες.
Για παράδειγμα ένα service μπορεί να διαχειρίζεται συνδέσεις δικτύου, βάσεων δεδομένων , ενημέρωση τοποθεσίας ανά διαστήματα , λήψη mail κ.α

Ένα service θα μπορούσαμε να πούμε ότι έχει δύο μορφές λειτουργίας , Started (ξεκίνημα) και Bound (δέσμευση).

Started

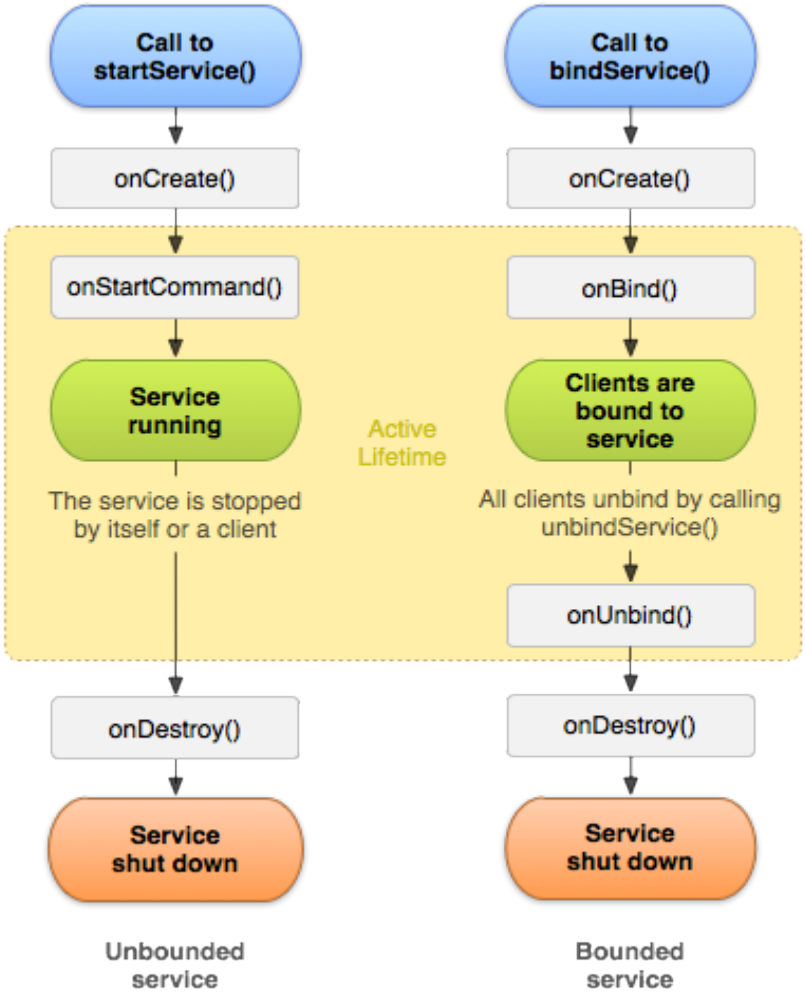
Ξεκινά από κάποια δραστηριότητα συνήθως με τη μέθοδο startService(). Εφόσον ξεκινήσει διατηρείται μακροχρόνια ακόμα και αν το συστατικό που την ξεκίνησε καταστραφεί. Αυτή η μορφή λειτουργίας είναι ιδανική όταν δεν περιμένουμε από το service να μας επιστρέψει κάτι , απλά θέλουμε να γίνει μια διαδικασία και μετά να σταματήσει. Πχ Το ανέβασμα κάποιου αρχείου σε server.

Bound

Μια υπηρεσία δεσμεύεται από το καλούμενο συστατικό (Activity) με την μέθοδο bindService(). Έτσι μας δίνεται η δυνατότητα να στέλνουμε αιτήσεις σε αυτό και να μας επιστέφει αποτελέσματα. Όταν το αποδεσμεύσουμε θα σταματήσεις τη λειτουργία του.

ΠΡΟΣΟΧΗ : Να σημειώσουμε ότι οι υπηρεσίες και οι δραστηριότητες από προεπιλογή τρέχουν στην ίδιο νήμα συνεπώς έχουμε μια διεργασία για όλα μαζί. Αν θέλουμε να δημιουργήσουμε κάποια βαριά υπηρεσία είναι καλό να έχουμε και ξεχωριστή διεργασία διαφορετικά μειώνεται η απόδοση της δραστηριότητας και κατ’ επέκταση της εφαρμογής.

Στο παρακάτω σχήμα φαίνονται οι κύκλοι ζωής ενός service και στις δύο του μορφές:



Σχήμα 4 : Διάγραμμα κύκλου ζωής ενός service

Όπως παρατηρούμε ολόκληρος ο ενεργός κύκλος ζωής ξενικά με τις μεθόδους onStartCommand() ή onBind() και τελειώνει με την onDestroy().

1.2.1.3 Content Provider

Οι content providers ή αλλιώς παροχείς περιεχομένου, είναι υπεύθυνοι για τη διαχείριση του αποθηκευτικού χώρου των δεδομένων. Στην ουσία αποθηκεύονται τα δεδομένα στο file system του κινητού έτσι ώστε και άλλες εφαρμογές να μπορούν τα έχουν πρόσβαση σε αυτά τα κοινά δεδομένα , μεταβάλλοντας τα αν χρειαστεί κιόλας.
Για παράδειγμα το σύστημα Android χρησιμοποιεί έναν content provider που διαχειρίζεται τα δεδομένα μιας επαφής χρήστη, έτσι κάθε άλλη εφαρμογή που έχει τα κατάλληλα δικαιώματα μπορεί να τροποποιήσει αυτά τα δεδομένα.

1.2.1.4 Broadcast Receiver

Οι broadcast receivers είναι ένας μηχανισμός που παρέχει ενημέρωση στην εφαρμογή όταν κάποιο γεγονός πραγματοποιηθεί. Τέτοια γεγονότα είναι για παράδειγμα όταν η στάθμη της μπαταρίας είναι πολύ χαμηλή ή όταν είμαστε εκτός δικτύου κ.α . Δεν παρέχουν user interface συνεπώς αν θέλουμε να ενημερώσουμε τον χρήστη ότι κάτι από αυτά συνέβη , το κάνουμε μέσω notifications.

Εμείς στις εφαρμογές μας χρειάστηκε να χρησιμοποιήσουμε τα 2 από τα 4 βασικά αυτά συστατικά του Android. Συγκεκριμένα χρησιμοποιήσαμε τα activities και το service.

Όσο αφορά την ενεργοποίηση αυτών των συστατικών , τα 3 από αυτά (activity, service,broadcast receiver) ενεργοποιούνται ασύγχρονα μέσω μηνυμάτων intent ή αλλιώς προθέσεις .(intent είναι η περιγραφή για το τι θα συμβεί, μεταφέροντας αυτή την εντολή, δηλαδή κάτι σαν messenger για την εφαρμογή) . Μια πρόθεση (intent) δημιουργείται με το ανάλογο αντικείμενο Intent object στο οποίο ορίζεται η πράξη που θα εκτελεστεί. Για παράδειγμα στις εφαρμογές χρησιμοποιείται κατά κόρων το intent για να ξεκινήσουμε νέες activities.

1.2.1.5 Shared Preferences

Τα shared preferences είναι στην ουσία ένα set από τιμές δεδομένων που αποθηκεύονται στη συσκευή. Ένα preference είναι τύπου key-value με συγκεκριμένο τύπο δεδομένων. Ως πεδίο key έχουμε ένα String που θα αποτελεί και το μοναδικό χαρακτηριστικό για το preference και ως value έχουμε την τιμή που θέλουμε.

Για παράδειγμα αν η εφαρμογή μας θέλει να αποθηκεύει όνομα χρήστη, μας αρκεί ένα preference με τις εξής πληροφορίες :

- Ο τύπος δεδομένων του preference (στην περίπτωσή μας String)
- Το όνομα της τιμής πάλι String (πχ “username”)
- Η ίδια η τιμή δηλαδή αυτό καθεαυτό το username (πχ “AndroidUser123”)

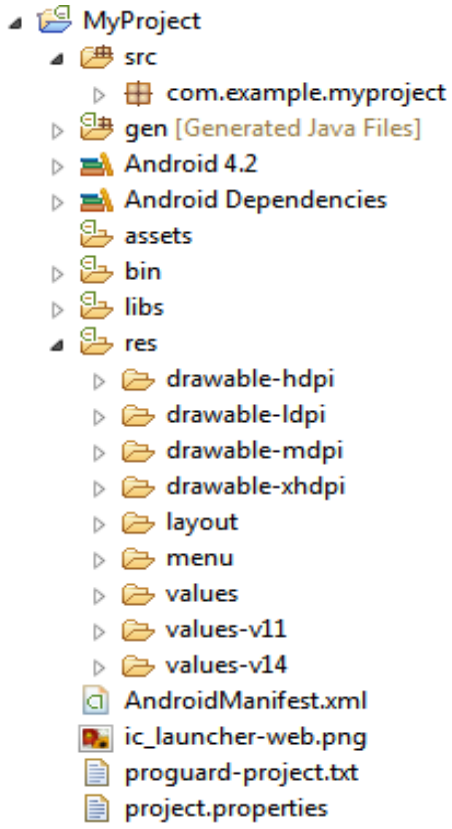
Ένα shared preference μπορεί να υποστηρίζει τους εξής τύπους δεδομένων :

- Boolean τιμές
- Float τιμές
- Integer τιμές
- Long τιμές
- String τιμές
- Τιμές ορισμένες από το χρήστη (αντικείμενα δικών μας κλάσεων πχ Customer)

Να τονίσουμε πως οι shared preferences , από προεπιλογή είναι προσπελάσιμες σε όλη την εφαρμογή μας (στις activities δηλαδή). Αυτό όμως δεν μας εμποδίζει από το να έχουμε περιορισμένες μόνο σε κάποιες activities.

1.3 Στοιχεία και δομή έργων Android

Κάθε έργο android είναι οργανωμένο κατάλληλα ώστε να υπάρχει μια σαφήνεια για τα συστατικά τα οποία αποτελείται. Είναι όλα διαμορφωμένα σε καταλόγους. Πιο αναλυτικά υπάρχουν οι εξής κατάλογοι ή/και αρχεία σε κάθε έργο :



src/ : Περιέχει όλα τα πηγαία αρχεία .java οργανωμένα σε πακέτα. Συνήθως στο workspace κάθε πακέτο είναι οργανωμένο και σε ένα φάκελο. Πχ Στην περίπτωση μας όπως βλέπουμε το σχήμα το πακέτο com.example.myproject το com είναι ο ριζικός κατάλογος, ο example ο υποκατάλογος και το myproject και άλλος υποκατάλογος.

gen/ : Περιέχει τα αρχεία .java που παράγονται αυτόματα από το σύστημα. Εδώ μετά το compilation συνδέονται όλα τα application resources (πηγές) με τον κώδικα

bin/ : Περιέχει το παραγόμενο αρχείο .apk που είναι στην ουσία η εφαρμογή μας

res/ : Περιέχει τους πόρους της εφαρμογής όπως εικόνες/εικονίδια/ήχους κτλ

Σχήμα 5 : Δομή project eclipse

Χωρίζεται στους εξής υποκαταλόγους :

- drawable-xxx
- layout
- values
- menu

Στους υποκαταλόγους **drawable-xxx** τοποθετούμε ποικίλες εικόνες bitmap κ.α διαφόρων μεγεθών για να εμφανίζονται σωστά σε πολλές οθόνες συσκευών. Για παράδειγμα συσκευές με οθόνη ή/και ανάλυση μεγάλη θα φορτώνουν αρχεία από drawable-hdpi και ούτω καθεξής.

Στον υποκατάλογο **layout** περιέχονται τα .xml αρχεία που αφορούν τις οδηγίες για το πώς θα σχεδιαστεί και εμφανιστεί μια οθόνη. Καθαρά για το user interface δηλαδή.

Στους υποκαταλόγους **values** έχουμε συγκεντρωμένα όλα τα αλφαριθμητικά και τυχών color-styles, όλων των components. Δηλαδή τα ονόματα που αντιστοιχούν σε κουμπιά(buttons) , λίστες (list view) , κείμενο (textbox)

*Τα values-v11 και v14 είναι για το API v11και 14 αντίστοιχα. Δηλαδή για τις μεταγενέστερες εκδόσεις android που έχουν extra χαρακτηρίστηκα.

Ο κατάλογος **menu** περιέχει τις δηλώσεις των τυχόν μενού που θα έχουμε σε κάθε οθόνη. πχ Start,about,help κ.α .Ενσωματώθηκε με την τελευταία έκδοση του SDK του Android, και σε νέο project που δημιουργούμε υπάρχει από προεπιλογή.

AndroidManifest.xml

Κάθε εφαρμογή πρέπει να έχει αυτό το αρχείο για να μπορέσει να αναπτυχθεί. Αυτό το αρχείο έχει καταγραφές πραγμάτων που αφορούν την εφαρμογή και την αλληλεπίδραση της με το λειτουργικό Android.

Μερικές από τις πληροφορίες που καταγράφονται είναι οι εξής :

- Τα ονόματα των πακέτων που αποτελείται η εφαρμογή
- Τις δηλώσεις των συστατικών στοιχείων της εφαρμογής , όπως ποιες δραστηριότητες(activities) έχουμε , ποιες υπηρεσίες(services) , υπό ποιες συνθήκες ενεργοποιούνται κτλ.
- Είναι δηλωμένες οι άδειες(permissions) που απαιτούνται για να αλληλεπιδράσουμε με το λειτουργικό android.

> Όπως έχουμε πει κάθε εφαρμογή λειτουργεί και αλληλεπιδρά με τις υπόλοιπες περιορισμένα ,εκτός αν ορίσουμε εμείς ποιους πόρους από τα περιφερειακά της συσκευής θα χρησιμοποιεί .Παραδείγματα ανάθεσης δικαιωμάτων είναι : read/write στην sd-card, πρόσβαση στο gps/wifi/3G κ.α

- Δηλώνεται το ελάχιστο όριο (min api level) , μέγιστο όριο (max api level) στο οποίο θα μπορεί να “τρέξει” η εφαρμογή.

> Υπενθυμίζουμε πως τα api level είναι ακέραιοι αριθμοί και αναφέρονται σε εκδόσεις λειτουργικού android. Συνεπώς εφόσον η εφαρμογή μας αναπτύσσεται ανάμεσα στο min και max api level θα μπορεί να λειτουργεί και στις ανάλογες εκδόσεις.
> Ενδεικτικά αναφέρουμε πως το api level 10 αντιστοιχεί στην έκδοση 2.3.3 και το api level 17 στην 4.2. Και έτσι θέτοντας αυτά σαν όρια η εφαρμογή μας θα μπορεί να λειτουργεί σε όλες τις ενδιάμεσες εκδόσεις . (2.3.3, 3.0.x ,3.1 , 3.2 , 4.0.x ,4.1 ,4.2)

- **Libs**:/ Είναι ο κατάλογος που συνηθίζεται να βάζουμε όλες τις εξωτερικές βιβλιοθήκες (.jar αρχεία) που χρησιμοποιούμε στην εφαρμογή μας . Είναι σημαντικό από τις ιδιότητες του project μας, στην επιλογή build path να επιλέξουμε αυτόν τον κατάλογο για να ενσωματωθούν αυτές οι εξωτερικές βιβλιοθήκες με το υπόλοιπο project.

1.4 Διεργασίες και νήματα στο Android

1.4.1 Εισαγωγή

Όταν μια εφαρμογή ξεκινά για πρώτη φορά , αυτόματα δημιουργείται μια νέα διεργασία στο λειτουργικό με ένα βασικό νήμα εκτέλεσης. Από προεπιλογή όλα τα συστατικά(components) της εφαρμογής "τρέχουν " στην ίδια διεργασία και νήμα (μάλιστα ονομάζεται και main thread).Το κύριο νήμα είναι ζωτικής σημασίας διότι είναι υπεύθυνο για να επικοινωνεί με τα UI widgets (οπτικά συστατικά πχ buttons,edit texts κτλ) και να τα εμφανίζει σωστά. Για παράδειγμα όταν ο χρήστης πατάει σε ένα κουμπί της οθόνης , το UI thread στέλνει ένα touch event στο ανάλογο widget το οποίο με τη σειρά του θέτει την κατάσταση του σε πατημένο(pressed) και εισάγεται (η κατάσταση) σε μια ουρά γεγονότων. Έτσι το UI thread εξάγει αυτή την κατάσταση και ενημερώνει το widget ότι πρέπει να ξανά-ζωγραφιστεί (redraw). Όταν η εφαρμογή απαιτεί να γίνουν "βαριές" δουλείες και γίνονται από το κύριο νήμα , είναι πιθανό να παρατηρηθούν χαμηλές επιδόσεις ειδικά όταν οι λειτουργίες που πρέπει να γίνουν είναι μακροπρόθεσμες (πχ επικοινωνία με δίκτυο, ερωτήματα σε βάσεις δεδομένων κτλ). Από την πλευρά του χρήστη η εφαρμογή φαίνεται να έχει παγώσει και δεν ανταποκρίνεται σε κανένα event, έτσι προκαλείται μια δυσφορία και αγανάκτηση αναγκάζοντας τον να τερματίσει την εφαρμογή ή και πόσο μάλλον να την απεγκαταστήσει .

1.4.2 Χρήση

Συμπερασματικά καλή πρακτική για να φτιάχνουμε αποδοτικές και άμεσα ανταποκρίσιμες εφαρμογής πρέπει να διασφαλίσουμε ότι το κύριο νήμα κάνει την λιγότερη δυνατή δουλειά που χρειάζεται. Οποιαδήποτε ενδεχόμενη μακροσκελής λειτουργία πρέπει να αντιμετωπίζεται σε διαφορετικό νήμα. Ο καλύτερος τρόπος για να το αντιμετωπίσουμε αυτό είναι η χρήση της κλάσης AsyncTask.

Μας επιτρέπει να διενεργούμε ασύγχρονα κάποιες ενέργειες στο παρασκήνιο και να εμφανίζονται τα αποτελέσματα στο προσκήνιο (UI). Η χρήση του είναι αρκετά απλή, το ορίζουμε ως εσωτερική κλάση (inner class) σε μια δραστηριότητα και χρησιμοποιούμε τις υλοποιημένες μεθόδους που έχει. Οι μέθοδοι είναι οι εξής :

- **onPreExecute(parameters)**

  Χρησιμοποιείται προαιρετικά για να ενημερώσουμε τον χρήστη ή να γίνει κάποια ενέργεια/αρχικοποιήσεις/μπάρες προόδου πριν εκτελεστεί η βασική λειτουργία.

- **doInBackground(parameters)**

  Καλείται και λειτουργεί στο παρασκήνιο αμέσως μετά που η onPreExecute() τερματίσει την λειτουργία της. Εδώ γράφουμε τον κώδικα για μια εργασία που θέλουμε να εκτελεστεί .

- **onPostExecute(parameters)**

  Εκτελείται αμέσως μετά την doInBackground() και μεταφέρει τα επιθυμητά αποτελέσματα στο κύριο νήμα, ώστε να βλέπει ο χρήστης ότι η ενέργεια περαιώθηκε.

Παράδειγμα

Ένα παράδειγμα φαίνεται παρακάτω :

```
class Async extends AsyncTask{
        //Run on UI thread
        @Override
        protected void onPreExecute(){}

        //Run in separate thread
        @Override
        protected Z doInBackground(X input){
                //Do task, like getting info from web
                return result;
        }

        //Run on UI thread
        @Override
        protected void onPostExecute(Z result){
                //Do something with data like update UI
        }
}

Async task = new Async();
task.execute(X input);
```

1.5 Μοναδικό αναγνωριστικό συσκευής

1.5.1 Εισαγωγή

Για τις εφαρμογές μας ,τόσο του οδηγού taxi όσο και του πελάτη καθώς και για την μεταξύ τους αλληλεπίδραση θα χρειαστεί για κάθε συσκευή με λειτουργικό σύστημα Android να έχουμε και κάποιο μοναδικό χαρακτηριστικό γνώρισμα (id), ώστε να υπάρχει τη δυνατότητα να ξεχωρίζουμε τις συσκευές στο σύνολό .

Έτσι ο διαμεσολαβητής μηνυμάτων που χρησιμοποιούμε (mosquito server-broker) να είναι σε θέση να μπορεί να στείλει μηνύματα από συσκευή σε συσκευή(ες).

Όσο αφορά το Android SDK , υπάρχουν αρκετοί τρόποι για να αποκτήσουμε μοναδικό id, αν και δεν είναι όλοι τους τόσο αποδοτικοί (ως προς την πληρότητα του id).

1.5.2 IMEI (International Mobile Equipment Identity)

Είναι ένας αριθμός της συσκευής που χρησιμοποιείται από τα GSM(1) δίκτυα κινητής τηλεφωνίας για να αναγνωρίζουν έγκυρες συσκευές. Πρόκειται ουσιαστικά για καθαρά μοναδικό κωδικό.

Χρησιμοποιείται και για λόγους ασφαλείας ,σε περιπτώσεις κλοπών, από τις αρμόδιες αρχές για την εύρεση των κινητών. Συνεπώς αποτελεί ευαίσθητο προσωπικό δεδομένο και δεν προτείνεται σαν λύση προς εμάς, για την παραγωγή μοναδικού id δηλαδή.

1.5.3 Ψευδό-μοναδικό ID (pseudo-unique)

Μπορούμε να παίρνουμε κάποια στοιχεία της συσκευής μας όπως λεπτομέρειες για την έκδοση της ROM, το όνομα του κατασκευαστή , τύπο και όνομα cpu και λοιπές πληροφορίες που αφορούν το υλικό.

Το Android SDK μας παρέχει την κλάση os.Build , έτσι έχουμε πρόσβαση στα ονόματα των μεταβλητών της συσκευής.

Στο παρακάτω πίνακα έχουμε ενδεικτικά κάποιες μεταβλητές που χρησιμοποιούμε για την εξαγωγή του id μαζί με τη σημασία τους :

BOARD	Τον κωδικό όνομα της συσκευής πχ (galaxy ace)
BRAND	Το όνομα της συσκευής (πχ samsung, htc κτλ)
DEVICE	Το εργοστασιακό κωδικό όνομα της συσκευής (πχ S5360)
MANUFACTURER	Ο εργοστασιακός κωδικός της γραμμής παραγωγής

Αν διαλέγαμε μόνο ένα από τα χαρακτηριστικά του παραπάνω πίνακα , θα ήταν λογικό

να υπάρξουν περιπτώσεις σύγχυσης. Δηλαδή , δεν είναι και εντελώς απίθανο να βρεθούν δύο ή και παραπάνω πανομοιότυπες συσκευές οι οποίες να παρουσιάζουν κοινά στοιχεία (να βασίζονται στην ίδια ROM, ίδια CPU και ακόμη ίδια μάρκα).

Για να αποφευχθεί αυτή η σύγχυση κρίνεται σκόπιμο να συνδυάσουμε όλα αυτά τα στοιχεία για να εξάγουμε το id. Έτσι μπορούμε να αποκλείσουμε το ενδεχόμενο να υπάρξουν συσκευές ταυτόχρονα με το ίδιο id.

Ακολουθεί ένα παράδειγμα για το πως εξάγουμε το id .

Παράδειγμα

Όσα στοιχεία χρησιμοποιούμε , από τόσα ψηφία θα αποτελείται το id. Στην περίπτωση του παραδείγματος μας το id θα ήταν αυτής της μορφής id : 6745 . (τυχαία τα νούμερα)

Ο αριθμός 6 αυτού του id θα αντιστοιχούσε στο BOARD , ο αριθμός 7 στο BRAND, ο 4 στο CPU_ABI και ο 5 στο DEVICE.

Για το πλήρες id χρησιμοποιούμε 13 χαρακτηριστικά και φυσικά αποτελείται από 13 ψηφία.

1.5.4 The Android ID

Πρόκειται για έναν 64bit αριθμό ο οποίος παράγεται τυχαία μια φορά κατά την πρώτη εκκίνηση

της συσκευής μας . Σε περίπτωση που κάνουμε κάποια αναβάθμιση του λειτουργικού Android σε κάποια πιο βελτιωμένη έκδοση το 64bit αριθμός θα ξαναδημιουργηθεί.

Τέλος θα έχουμε πάλι διαφορετικό id αν επαναφέρουμε το κινητό μας στις εργοστασιακές ρυθμίσεις (factory reset).

Συμπερασματικά δεν είναι κάποιο μόνιμο id και δεν θα το προτιμήσουμε ως κάποια λύση.

1.5.6 The WLAN MAC Address string & BT(Bluetooth) MAC Address string ως ID

Ως γνωστό όλες οι συσκευές δικτύου έχουν μια μοναδική διεύθυνση αναγνώρισης , την γνωστή και ως MAC ADDRESS (hex μορφή 00:11:22:33:44:55). Είτε πρόκειται για κάρτα δικτύου επιτραπέζιου υπολογιστή ή laptop είτε άλλη φορητή συσκευή όλες έχουν τη δική τους ξεχωριστή διεύθυνση αναγνώρισης.

Και πάλι το Android SDK στην πληθώρα των κλάσεων που μας δίνει , υπάρχει και η αντίστοιχη κλάση για να έχουμε πρόσβαση στο WLAN interface.

Παράδειγμα

Ακριβώς τα ίδια ισχύουν και για τη MAC ADDRESS του bluetooth.

Ένα συμπαντικό μειονέκτημα είναι ότι οι MAC διευθύνσεις αποτελούν και αυτές ευαίσθητες πληροφορίες , και η κατοχή τέτοιων πληροφοριών σε λάθος χέρια εγκυμονούν κινδύνους για κάθε είδους πιθανής επίθεσης.

Άλλο σημαντικό μειονέκτημα είναι ότι για να πάρουμε αυτές τις διευθύνσεις από μερικές συσκευές πρέπει αναγκάστηκα να είμαστε συνδεδεμένοι σε κάποιο δίκτυο, κάτι το περιοριστικό.

Σε ιδανικές περιπτώσεις ο συνδυασμός όλως αυτών των περιπτώσεων, δηλαδή να αναμείξουμε όλα τα επιμέρους id θα είχε ως αποτέλεσμα την δημιουργία του ενός απόλυτου ID.

1.6 Τρόπος αποστολής-λήψης δεδομένων με τον διακομιστή

1.6.1 Εισαγωγή

Για τις ανάγκες των εφαρμογών μας ήταν απαραίτητο να μεταφέρονται δεδομένα από τον server με ενιαίο και δομημένο τρόπο. Καταλήξαμε ότι δύο βολικές λύσεις θα ήταν να επιστρέφουμε τα αποτελέσματα σε μορφή XML ή JSON ,προτιμώντας την δεύτερη λύση (μορφή JSON).

Οι λόγοι που μας οδήγησαν σε αυτή την επιλογή είναι οι εξής :

- Αρκετά πιο απλό, λιγότερη και σαφής σύνταξη εντολών, χαρτογραφώντας πιο άμεσα τις δομές δεδομένων που χρησιμοποιούνται στις μοντέρνες γλώσσες προγραμματισμού.
- Δεν χρειάζεται να ορίζουμε νέα tags ή χαρακτηριστικά για να αναπαριστούμε τα δεδομένα.
- Είναι ακόμα πιο εύκολα αναγνώσιμη από τον άνθρωπο πόσο μάλλον από τη μηχανή. Υπόσχεται ευκολία στο γράψιμο.
- Επεξεργάζεται πιο εύκολα λόγω της απλής δομής

Ένα σημαντικό μειονέκτημα είναι ότι δεν μας παρέχει δυνατότητες εμφάνισης, όπως η XML επειδή δεν είναι γλώσσα περιγραφική (document markup language)

Στα δικά μας δεδομένα τώρα από τη μεριά του server όλα τα PHP scripts τα αποτελέσματα που παράγουν τα τυπώνουμε σε μορφή JSON με τη βοήθεια της εντολής echo json_encode($result).

Παράδειγμα

Ας κάνουμε ένα παράδειγμα για να γίνει κατανοητό . Έστω ο πίνακας drivers στη βάση δεδομένων μας.

id	driverName	taxiPlate
1	charlie	ZMK-4356
2	paul	IMN-2352

Τα δεδομένα όλου αυτού του πίνακα σε μορφή JSON θα ήταν τα εξής :

```
{
    "drivers":[
            {
                "id": "1",
                "driverName": "charlie",
                "taxiPlate": "ZMK-4356",
            },
            {
                "id": "2",
                "driverName": "paul",
                "taxiPlate ": "IMN-2352",
            }
  ] //end of array

} // end of object
```

Από τη μεριά των εφαρμογών Android έχουμε την κατάλληλη κλάση ώστε να μπορεί να χειρίζεται αυτό το response ως object.

```
JSONObject myjObj = makehttpRequest(String url,String method,List
parameters)
```

url : η διαδρομή του php αρχείου πχ http://localhost/files/myphp.php

method : η μέθοδος του request ”POST” / “GET”

parameters : οι τυχόν παράμετροι που θα περάσουμε με post

Χρήση

```
String name =
myjObj.getJSONArray(“drivers”).getJSONObject(1).get(“driverName”)
```

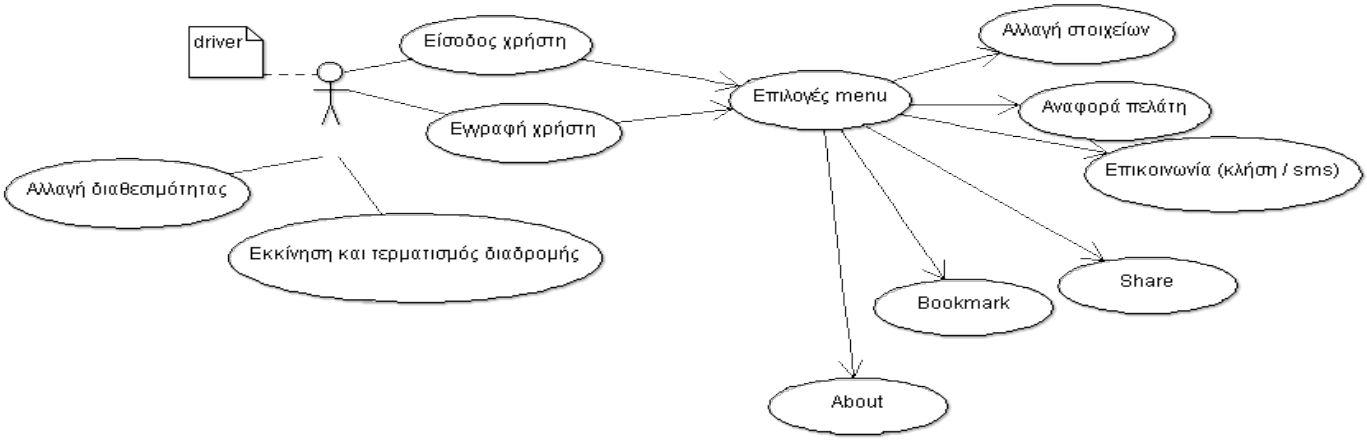
Έτσι παίρνουμε το όνομα “paul”

Έτσι στο αντικείμενο αυτό εκχωρείται όλο το αποτέλεσμα. Συνεπώς τώρα μπορούμε να διαχειριστούμε αυτά τα στοιχεία και μέσα στις εφαρμογές Android όπως θέλουμε.

2 Περιπτώσεις χρήσης εφαρμογών

2.1 Περίπτωση χρήσης οδηγού

1. Είσοδος χρήστη
2. Εγγραφή χρήστη
3. Εκκίνηση και τερματισμός διαδρομής
4. Αλλαγή διαθεσιμότητας
5. Επιλογές μενού
    - Αλλαγή στοιχείων
    - Αναφορά πελάτη
    - Επικοινωνία (κλήση/sms)
    - Κοινοποίηση (Share)
    - Προσθήκη στα αγαπημένα (Bookmark)
    - Σχετικά με την εφαρμογή



Σχήμα 2 : Διάγραμμα περιπτώσεων χρήσης εφαρμογή οδηγού

2.2 Βασική ροή οδηγού

1. **Είσοδος χρήστη**
   Αφού έχει κάνει ήδη εγγραφή θα μπορεί να μπει στο σύστημα δίνοντας τα στοιχεία του.

2. **Εγγραφή χρήστη**
   Εφόσον είναι νέος χρήστης θα πρέπει να κάνει νέα εγγραφή δίνοντας στοιχεία όπως όνομα χρήστη, κωδικός, διεύθυνση κτλ.

3. **Εκκίνηση και τερματισμός διαδρομής**
   Λήγει η παραγγελία και μεταβαίνει στην αρχική οθόνη για νέα παραγγελία

4. **Αλλαγή διαθεσιμότητας**
   Ο οδηγός μπορεί να αλλάξει την διαθεσιμότητα του ενεργός/μη ενεργός έτσι ώστε να μπορεί να δέχεται αιτήματα από πελάτες ή όχι αντίστοιχα.

5. **Επιλογές μενου**
   - **5.1 Αλλαγή στοιχείων**
     Όταν είναι συνδεδεμένος μπορεί να αλλάξει τα προσωπικά του στοιχεία (όνομα, τηλ , διεύθυνση , αρ. πινακίδας κτλ)
   - **5.2 Αναφορά πελάτη**
     Αν για κάποιο λόγο ο οδηγός δυσανασχετήσει με την συμπεριφορά του πελάτη, του δίνεται η δυνατότητα να κάνει αναφορά τον πελάτη για απρεπή συμπεριφορά επιλέγοντας την αιτία από μια λίστα ή και να προσθέσει δικιά του αιτία.
   - **5.3 Επικοινωνία**
     Σε περίπτωση ανάγκης δίνεται η δυνατότητα στον χρήστη να μπορέσει να επικοινωνήσει μέσω αριθμού τηλεφώνου στον πελάτη ή και να στείλει sms.
   - **5.4 Κοινοποίηση**
     Μπορεί να μοιραστεί το site της εφαρμογής με κοινωνικά δίκτυα.
   - **5.5 Προσθήκη στα αγαπημένα**
     Πρόσθήκη στα αγαπημένα το site της εφαρμογής
   - **5.6 Σχετικά με την εφαρμογή**
     Προβολή πληροφοριών σχετικά με την εφαρμογή

2.3 Εναλλακτική ροή οδηγού

1)

i. Να δώσει λάθος στοιχεία. Εμφάνιση ανάλογου μηνύματος.
ii. Να ξεχάσει να συμπληρώσει κάποιο πεδίο. Εμφάνιση ανάλογου μηνύματος.
iii. Να μην μπορεί να επικοινωνήσει με το server για την ταυτοποίηση. Εμφάνιση ανάλογου μηνύματος.

2)

i. Να ξεχάσει να συμπληρώσει κάποιο πεδίο. Εμφάνιση ανάλογου μηνύματος.
ii. Να δώσει στοιχεία που ήδη υπάρχουν. Εμφάνιση ανάλογου μηνύματος.
iii. Να μην μπορεί να επικοινωνήσει με το server για την εγγραφή. Εμφάνιση ανάλογου μηνύματος.

4)

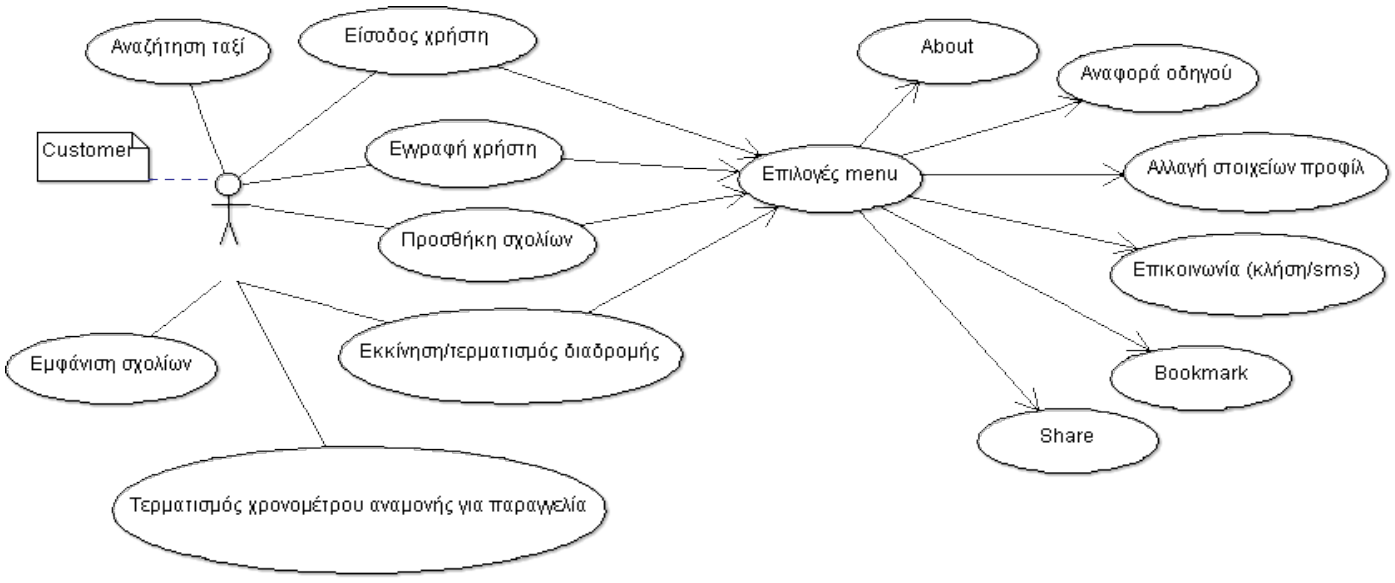
Να μην μπορεί να επικοινωνήσει με το server για να αλλάξει η διαθεσιμότητα . Εμφάνιση ανάλογου μηνύματος.

5)

i. Να μην μπορεί να επικοινωνήσει με το server για να πάρει τον αριθμό τηλεφώνου ώστε να επικοινωνήσει. Εμφάνιση ανάλογου μηνύματος.
ii. Να ξεχάσει να συμπληρώσει κάποιο πεδίο στο edit. Εμφάνιση ανάλογου μηνύματος.
iii. Να μην μπορεί να επικοινωνήσει με το server ώστε να καταχωρηθεί η αναφορά. Εμφάνιση ανάλογου μυνήματος.

2.4 Περίπτωση χρήσης πελάτη

1. Αναζήτηση ταξί
2. Είσοδος χρήστη
3. Εγγραφή χρήστη
4. Προσθήκη σχολίων /rating
5. Επιλογές μενού
    - Αλλαγή στοιχείων
    - Αναφορά πελάτη
    - Επικοινωνία (κλήση/sms)
    - Κοινοποίηση (Share)
    - Προσθήκη στα αγαπημένα (Bookmark)
    - Σχετικά με την εφαρμογή



Σχήμα 1: Διάγραμμα περιπτώσεων χρήσης εφαρμογής πελάτη

2.5 Βασική ροή πελάτη

1. **Αναζήτηση ταξί**
   Ο πελάτης θα μπορεί με το πάτημα ενός κουμπιού να καλέσει ένα ταξί στην τοποθεσία που είναι. Στην συνέχεια θα μπορεί μέσω χάρτη να επιλέξει την ακριβή του τοποθεσία, στην οποία θα έρθει το ταξί.

2. **Είσοδος χρήστη**
   Αφού έχει κάνει ήδη εγγραφή θα μπορεί να μπει στο σύστημα δίνοντας τα στοιχεία του.

3. **Εγγραφή χρήστη**
   Εφόσον είναι νέος χρήστης θα πρέπει να κάνει νέα εγγραφή δίνοντας στοιχεία όπως όνομα χρήστη, κωδικός, διεύθυνση κτλ.

4. **Προσθήκη σχολίων/rating**
   Πριν ολοκληρωθεί η παραγγελία ο χρήστης μπορεί να κάνει κάποια σχόλια για την συνολική του εμπειρία για την διαδρομή καθώς και να βαθμολογήσει τον οδηγό.

5. **Επιλογές μενού**
   - **5.1 Αλλαγή στοιχείων**
     Όταν είναι συνδεδεμένος μπορεί να αλλάξει τα προσωπικά του στοιχεία (όνομα, τηλ , διεύθυνση , αρ. πινακίδας κτλ)
   - **5.2 Αναφορά οδηγού**
     Αν για κάποιο λόγο ο πελάτης δυσανασχετήσει με την συμπεριφορά του οδηγού, του δίνεται η δυνατότητα να κάνει αναφορά τον οδηγό για απρεπή συμπεριφορά επιλέγοντας την αιτία από μια λίστα ή και να προσθέσει δικιά του αιτία.
   - **5.3 Επικοινωνία**
     Σε περίπτωση ανάγκης δίνεται η δυνατότητα στον χρήστη να μπορέσει να επικοινωνήσει μέσω αριθμού τηλεφώνου στον οδηγού ή και να στείλει sms.
   - **5.4 Κοινοποίηση**
     Μπορεί να μοιραστεί το site της εφαρμογής με κοινωνικά δίκτυα.
   - **5.5 Προσθήκη στα αγαπημένα**
     Πρόσθήκη στα αγαπημένα το site της εφαρμογής
   - **5.6 Σχετικά με την εφαρμογή**
     Προβολή πληροφοριών σχετικά με την εφαρμογή

2.6 Εναλλακτική ροή πελάτη

1)

i. Να μην υπάρχει κανένα διαθέσιμο ταξί, εμφάνιση μηνύματος
ii. Αδυναμία επικοινωνίας με τον εξυπηρετητή .Εμφάνιση προειδοποίησης .

2)

i. Να δώσει λάθος στοιχεία. Εμφάνιση ανάλογου μηνύματος.
ii. Να ξεχάσει να συμπληρώσει κάποιο πεδίο. Εμφάνιση ανάλογου μηνύματος.
iii. Να μην μπορεί να επικοινωνήσει με το server για την ταυτοποίηση. Εμφάνιση ανάλογου μηνύματος.

3)

i. Να ξεχάσει να συμπληρώσει κάποιο πεδίο. Εμφάνιση ανάλογου μηνύματος.
ii. Να δώσει στοιχεία που ήδη υπάρχουν. Εμφάνιση ανάλογου μηνύματος.
iii. Να μην μπορεί να επικοινωνήσει με το server για την εγγραφή. Εμφάνιση ανάλογου μηνύματος.

4)

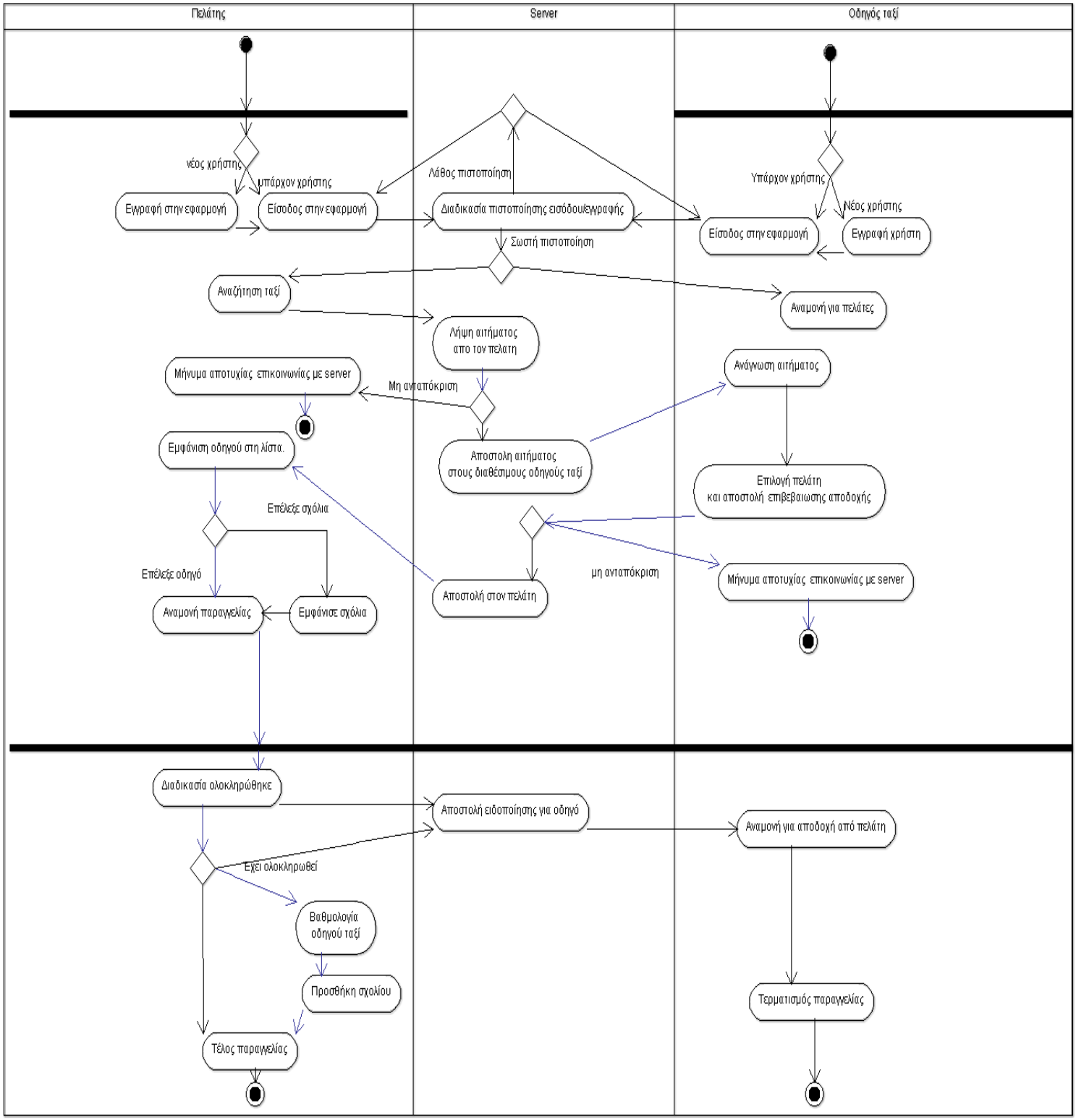
i. Να ξεχάσει να συμπληρώσει κάποιο πεδίο. Εμφάνιση ανάλογου μηνύματος.
ii. Να μην μπορεί να επικοινωνήσει με το server για την εγγραφή. Εμφάνιση ανάλογου μηνύματος.

5)

i. Να μην μπορεί να επικοινωνήσει με το server για να πάρει τον αριθμό τηλεφώνου ώστε να επικοινωνήσει. Εμφάνιση ανάλογου μηνύματος.
ii. Να ξεχάσει να συμπληρώσει κάποιο πεδίο στο edit. Εμφάνιση ανάλογου μηνύματος.
iii. Να μην μπορεί να επικοινωνήσει με το server ώστε να καταχωρηθεί η αναφορά. Εμφάνιση ανάλογου μυνήματος.

2.7 Διάγραμμα δραστηριότητας

Τέλος για να γίνει ίσως ποιο κατανοητή η λειτουργία των εφαρμογών παραθέτουμε στο διάγραμμα δραστηριότητας τη ροή της πληροφορίας και την αλληλεπίδραση μεταξύ server και 2 εφαρμογών.



Σχήμα 3 : Διάγραμμα δραστηριότητας εφαρμογών

2.8 Design Patterns

Κρίθηκε απαραίτητο να χρησιμοποιήσουμε κάποια μοτίβα σχεδίασης για να βελτιωθεί η δομή του κώδικα καθώς και η επαναχρησιμοποίηση και συντήρηση. Τα πρότυπα που χρησιμοποιήθηκαν είναι το Singleton για τις global μεταβλητές γιατί θέλαμε να εξασφαλίσουμε ότι μόνο ένα στιγμιότυπο κάθε μεταβλητής θα είναι ορισμένο, και το Factory pattern το οποίο αναλύουμε παρακάτω.

- **Singleton pattern**

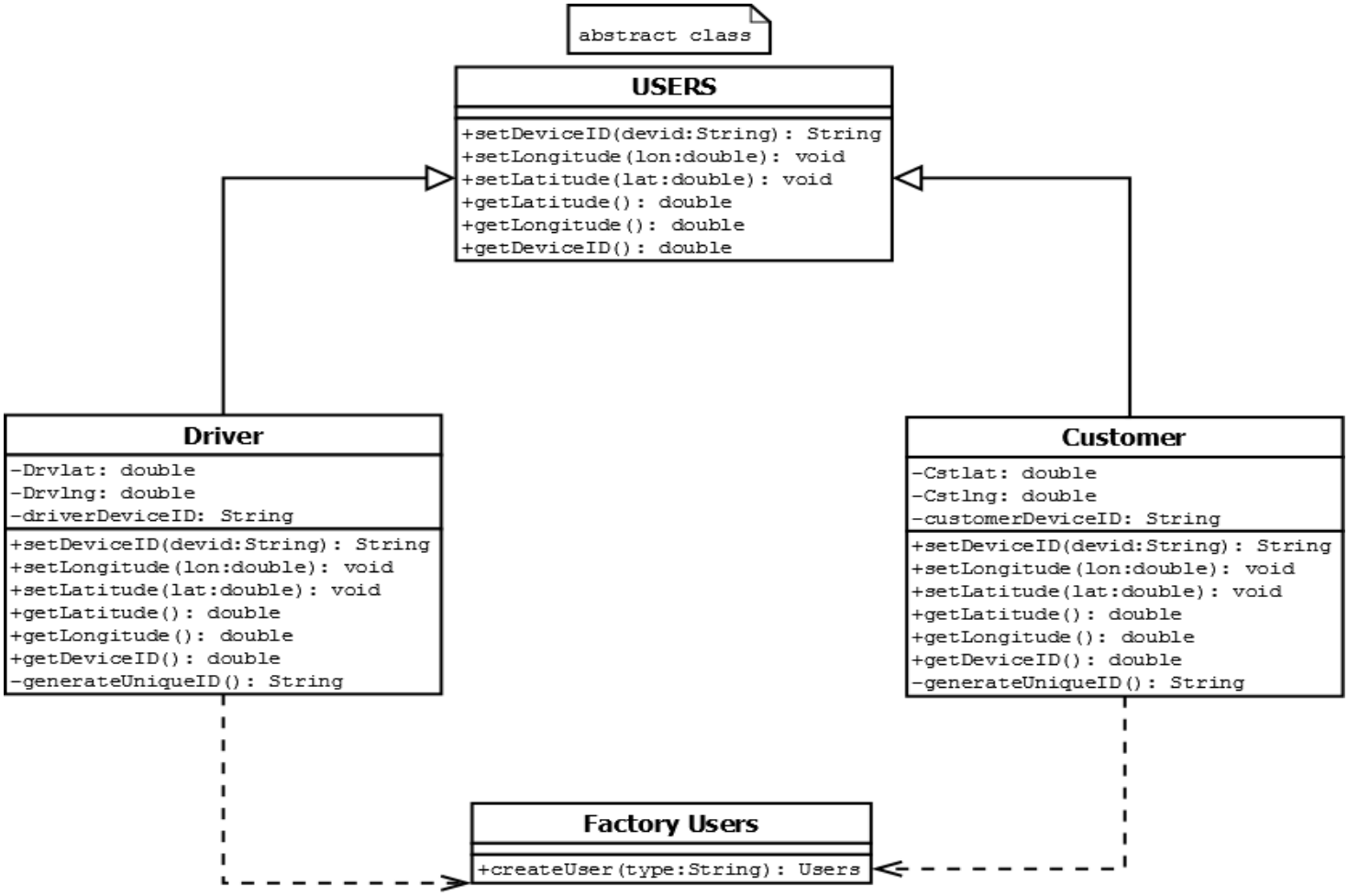
Παράδειγμα για τις global μεταβλητές :

```
[...]

private String MQTT_IP_ADDRESS ;
private int MQTT_PORT;

private    String IP_ADDRESS ;
private    int HTTP_PORT;
private    String BASE_PATH;

private static globalVariables instance = null;

[...]

public static globalVariables getInstance() {
        if (instance == null)
                instance = new globalVariables();
        return instance;
}
```

Κάθε φορά λοιπόν που χρησιμοποιούμε μια από αυτές τις μεταβλητές, δεν την προσπελαύνουμε ευθέως αλλά μέσω της συνάρτησης globalVariables η οποία μας επιστρέφει την τιμή της μεταβλητής εφόσον είναι ορισμένη αλλιώς αρχικοποιείται στον constractor.

- **Factory pattern**

Παρατηρήθηκε πως οι χρήστες παρουσιάζουν κοινές λειτουργίες ως προς τα χαρακτηριστικά τους με μικρές ίσως διαφοροποιήσεις και άλλη υλοποίηση στις μεθόδους του καθενός. Έτσι δημιουργήσαμε ένα πακέτο που υλοποιεί το factory pattern για τη δημιουργία των αντικειμένων.
Έχουμε μια abstract κλάση Users η οποία περιέχει τις κοινές μεθόδους χωρίς υλοποίηση (το interface δηλαδή). Έπειτα έχουμε και τις άλλες δύο κλάσεις Customer και Driver οι οποίες κληρονομούν από την abstract κλάση έχοντας η καθεμία την δική της υλοποίηση.
Ως τελικό αποτέλεσμα είναι να μπορούμε να προσθέτουμε εύκολα και νέους χρήστες πχ(Administrators) καθώς και νέα χαρακτηριστικά χωρίς πολλές αλλαγές και χωρίς δυσνόητο κώδικα. Παρακάτω φαίνεται το διάγραμμα κλάσεων του design pattern.



Σχήμα 4 : Διάγραμμα κλάσεων factory pattern

Για περισσότερες πληροφορίες σχετικά με την υλοποίηση μπορείτε να ανατρέξετε στον πηγαίο κώδικα στο πακέτο app.taxiAnytimeDriver(ή Customer).userTypesFactory και στα αρχεία που περιέχονται μέσα Customer.java, Driver.java, Users.java, UserFactory.java .

3 Τεχνολογίες συστήματος ειδοποιήσεων στο Android

3.1 Εισαγωγή

Σκοπός μας είναι να λαμβάνουμε ειδοποιήσεις τόσο για τους οδηγούς όσο και για τους πελάτες σε πραγματικό χρόνο. Για να πραγματοποιηθεί αυτό υπάρχουν αρκετοί τρόποι.. αν και όχι τόσο αποδοτικοί όλοι τους.

Αυτό που πρέπει να κάνουμε είναι να συγχρονίσουμε τις δυο εφαρμογές, (πελάτη και οδηγό ταξί) ώστε να μπορούν να επικοινωνούν μεταξύ τους. Είναι απαραίτητο να στέλνουν διάφορα μηνύματα (ειδοποιήσεις) μεταξύ τους, όπως για παράδειγμα την στιγμή που ο πελάτης πατάει το κουμπί για να ζητήσει ταξί πρέπει άμεσα να σταλεί μια ειδοποίηση σε όλους τους κοντινούς του, οδηγούς ταξί, η αποδοχή του οδηγού να γυρίσει άμεσα στον πελάτη κ.α

Για να λυθεί το παραπάνω πρόβλημα πρέπει να έχουμε έναν multi-tread server ο οποίος θα λαμβάνει την ειδοποίηση από την μια συσκευή κ θα τη διαμοιράζει σε Ν άλλες συσκευές. Εμείς έχουμε έναν PHP server αλλά όπως είναι γνωστό οι PHP servers δεν είναι σχεδιασμένοι για multi-treading. Άρα θα χρειαστούμε έναν server σε μια γλώσσα όπως η java που υποστηρίζει παράλληλη επικοινωνία του server με πολλούς clients (αυτό που θέλουμε εμείς ουσιαστικά). Και επειδή το ίδιο ακριβώς πρόβλημα υπήρχε πολύ πριν το συναντήσουμε και εμείς, έχουν βγει κάποιες βιβλιοθήκες και κάποια frameworks που υλοποιούν αυτό ακριβώς που θέλουμε. Άρα γιατί να ξανά ανακαλύψουμε τον τροχό? (!)

3.2 Πιθανές λύσεις

Μεγάλες εταιρίες όπως η google και η IBM έχουν σχεδιάσει frameworks ή και ολόκληρα πρωτόκολλα επικοινωνίας για να λύσουν το πρόβλημα με τις ειδοποιήσεις. Εμείς τα αναλύσαμε, και επιλέξαμε αυτό που μας ταίριαζε περισσότερο. Πιο αναλυτικά…

Οι συνηθέστεροι τρόποι για αυτή την λειτουργία είναι :

- Ειδοποίηση μέσω sms
- Μόνιμες συνδέσεις με TCP/IP sockets
- Poll
- Push (IBM MQTT protocol, C2DM,GCM)

Εξηγούμε παρακάτω για το καθένα :

3.2.1 Ειδοποίηση μέσω sms

Ίσως ο λιγότερος αποδοτικός αλλά και με το μεγαλύτερο κόστος, τρόπος υλοποίησης. Κάθε φορά που έχουμε νέα δεδομένα για να ενημερώσουμε τους χρήστες θα στέλνεται και ένα sms ειδικής μορφής για να τον ενημερώσει.

Θετικά:

- Πολύ εύκολος τρόπος υλοποίησης
- Ειδοποιήσεις σε πραγματικό χρόνο

Αρνητικά:

- Το κόστος αποστολής του sms από τον παροχέα υπηρεσιών κινητής τηλεφωνίας
-

  *Μην ξεχνάμε ότι ο σκοπός της εφαρμογής είναι να αποφύγουμε οποιοδήποτε είδους επιπλέον χρέωσης.

3.2.2 Μόνιμες συνδέσεις με TCP/IP sockets (socket programming)

Πρόκειται για τον κλασσικό τρόπο επικοινωνίας μέσα στο διαδίκτυο. Ως γνωστόν για να επιτευχθεί η σύνδεση χρειαζόμαστε τις ip addresses τόσο του server όσο και του κάθε client καθώς και τις αντίστοιχες ports. Η σύνδεση παραμένει σταθερή με τον server στέλνοντας keepalive messages. Έτσι όταν έχουμε νέα δεδομένα στον server , τα στέλνουμε πάνω από την tcp/ip σύνδεση.

Θετικά:

- Λαμβάνει ειδοποιήσεις σε πραγματικό χρόνο όπως θα θέλαμε.

Αρνητικά:

- Είναι πολύ χρονοβόρο και δύσκολο στην υλοποίηση μιας τέτοιας υπηρεσίας, πόσο μάλλον να είναι και αξιόπιστο καθώς και δε συντηρείται εύκολα.
- Το λειτουργικό Android είναι έτσι σχεδιασμένο ώστε να τερματίζει τις υπηρεσίες που εκτελούνται αρκετή ώρα χωρίς να κάνουν κάτι, πόσο μάλλον πχ. όταν υπάρχει χαμηλή στάθμη στην μπαταρία
- Όταν η συσκευή Android μεταβαίνει σε κατάσταση αναμονής πάλι κινδυνεύει να τερματιστεί η υπηρεσία, αλλά εκτός αυτού δεν θα είμαστε σίγουροι ότι θα λάβουμε τα νέα δεδομένα.
- Έχει παρατηρηθεί ότι να διατηρούμε ενεργή συνεχώς αυτή τη σύνδεση , προκαλεί γρήγορη κατανάλωση της μπαταρίας του τηλεφώνου, κάτι που είναι δυσάρεστο για τους χρήστες.

3.2.3 Poll

Η τεχνολογία polling έχει ως φιλοσοφία να στέλνει αιτήματα στον server ανά τακτά χρονικά διαστήματα για να "δει" αν υπάρχουν νέα δεδομένα μέσω ενός service.

Θετικά:

- Είναι αρκετά εύκολο στην υλοποίηση.

Αρνητικά:

- Δεν θα είναι ποτέ σε πραγματικό χρόνο. Διότι θέτοντας polling time πχ 5 λεπτά , θα λάβουμε την ειδοποίηση που έγινε μετά από αυτόν το χρόνο, καθυστερημένα.
  Μάλιστα για να ήταν real-time θα έπρεπε να θέσουμε polling time 1 sec ώστε ανά τόσο διάστημα να ελέγχει για νέα δεδομένα, κάτι πραγματικά καταστροφικό για την μπαταρία της συσκευής, αλλά και την άσκοπη χρήση στην κίνηση δικτύου. Εξάλλου μπορεί και να μην υπήρχαν νέα δεδομένα και θα ελέγχαμε άσκοπα.

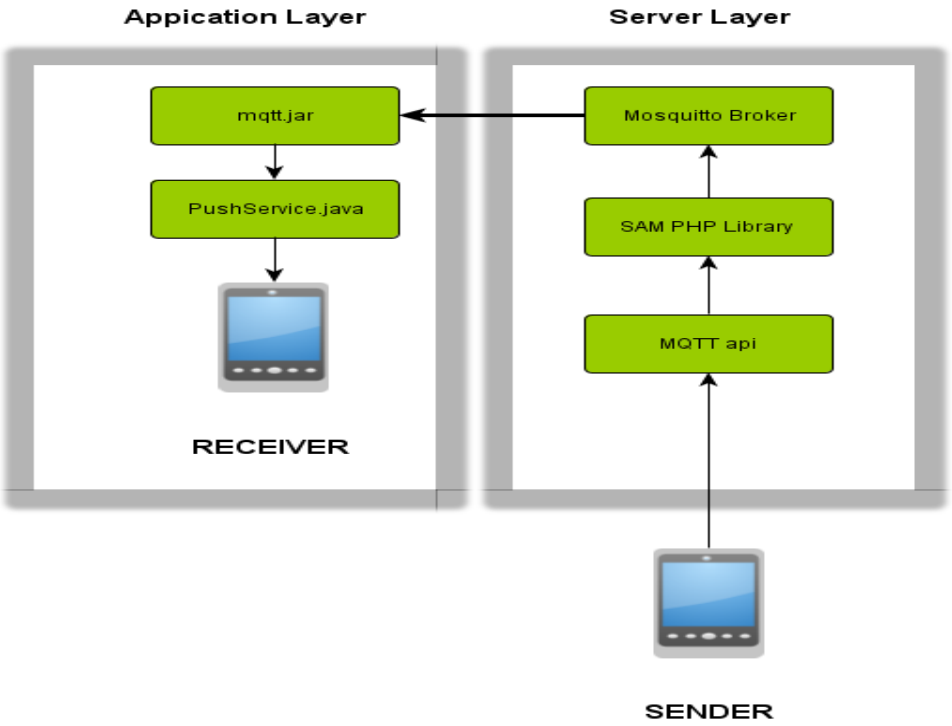
3.2.4 Push MQTT (Message Queue Telemetry Transport)

Το mqtt είναι ένα πρωτόκολλο που δημιούργησε η IBM το 1999 το οποίο έχει να κάνει με ανταλλαγή μηνυμάτων στο διαδίκτυο. Είναι σχεδιασμένο έτσι ώστε να μπορεί να χρησιμοποιηθεί και από συσκευές με μικρές δυνατότητες (μικρές μπαταριές, μικρή μνήμη κλπ), όπως είναι τα κινητά μας .

Για να το χρησιμοποιήσουμε θα χρειαστούμε τον server της εφαρμογής, έναν διαμεσολαβητή (broker), έναν mqtt client και την php βιβλιοθήκη SAM. Τέλος, θα πρέπει να έχουμε το id της συσκευής στην οποία θα στείλουμε το μήνυμα. Εύκολο διότι κάθε συσκευή android έχει ένα μοναδικό device id το οποίο σαν δεδομένο είναι public, άρα έχουμε πρόσβαση σε αυτή.

Στέλνουμε το μήνυμα στον server μας.Ο server το προωθεί μαζί με τα device id αποστολέα και παραλήπτη χρησιμοποιώντας την php βιβλιοθήκη SAM στον διαμεσολαβητή (broker) τον οποίο διανέμει δωρεάν η IBM. Ο διαμεσολαβητής στέλνει το μήνυμα στον java mqtt client τον οποίο θα πρέπει να ενσωματώσουμε στην εφαρμογή μας ώστε να μπορεί να επεξεργαστεί τις ειδοποιήσεις. Ο mqtt client στην συνέχεια στέλνει την ειδοποίηση στην εφαρμογή μας όπου θα πρέπει να γράψουμε κώδικα με τον οποίο θα την διαχειριστούμε.

Ακολουθεί σχεδιάγραμμα σχετικά με την αρχιτεκτονική και ενσωμάτωσης του mqtt στην εφαρμογή μας.



Σχήμα 1 : Αρχιτεκτονική πρωτοκόλλου επικοινωνίας (mqtt)

Πλεονεκτήματα του πρωτοκόλλου

- Σωστή χρήση πόρων ώστε να καταναλώνουμε την λιγότερη δυνατή μπαταρία. Μελέτες έχουν δείξει ότι κατά την διάρκεια που είναι η εφαρμογή είναι στο προσκήνιο, το MQTT καταναλώνει λιγότερη ενεργεία απ ότι το C2DM.
- Το μέγιστο όριο ενός μηνύματος είναι 256MB!
- Είναι σχεδιασμένο ώστε να ανταποκρίνεται πολύ γρηγορότερα σε σχέση με το C2DM όταν έχει να αποστείλει ένα μήνυμα σε πολλούς παραλήπτες.
- Είναι πολύ ελαφρύ και σχεδιασμένο να μπορεί να αντεπεξέρθει σε δίκτυα με χαμηλό εύρος ζώνης (bandwidth)

- Το MQTT έχει τρία επίπεδα QoS (**Q**uality **o**f **S**ervice) τα οποία είναι:
    - 0: Ο διαμεσολαβητής (broker) θα παραδώσει το μήνυμα (ειδοποίηση) χωρίς να λάβει επιβεβαίωση
    - 1: Ο διαμεσολαβητής θα στείλει το μήνυμα τουλάχιστον μια φορά, έως ότου λάβει επιβεβαίωση.
    - 2: Ο διαμεσολαβητής θα παραδώσει το μήνυμα με σιγουριά χρησιμοποιώντας μια χειραψία τεσσάρων βημάτων (four way handshake) [(2)].

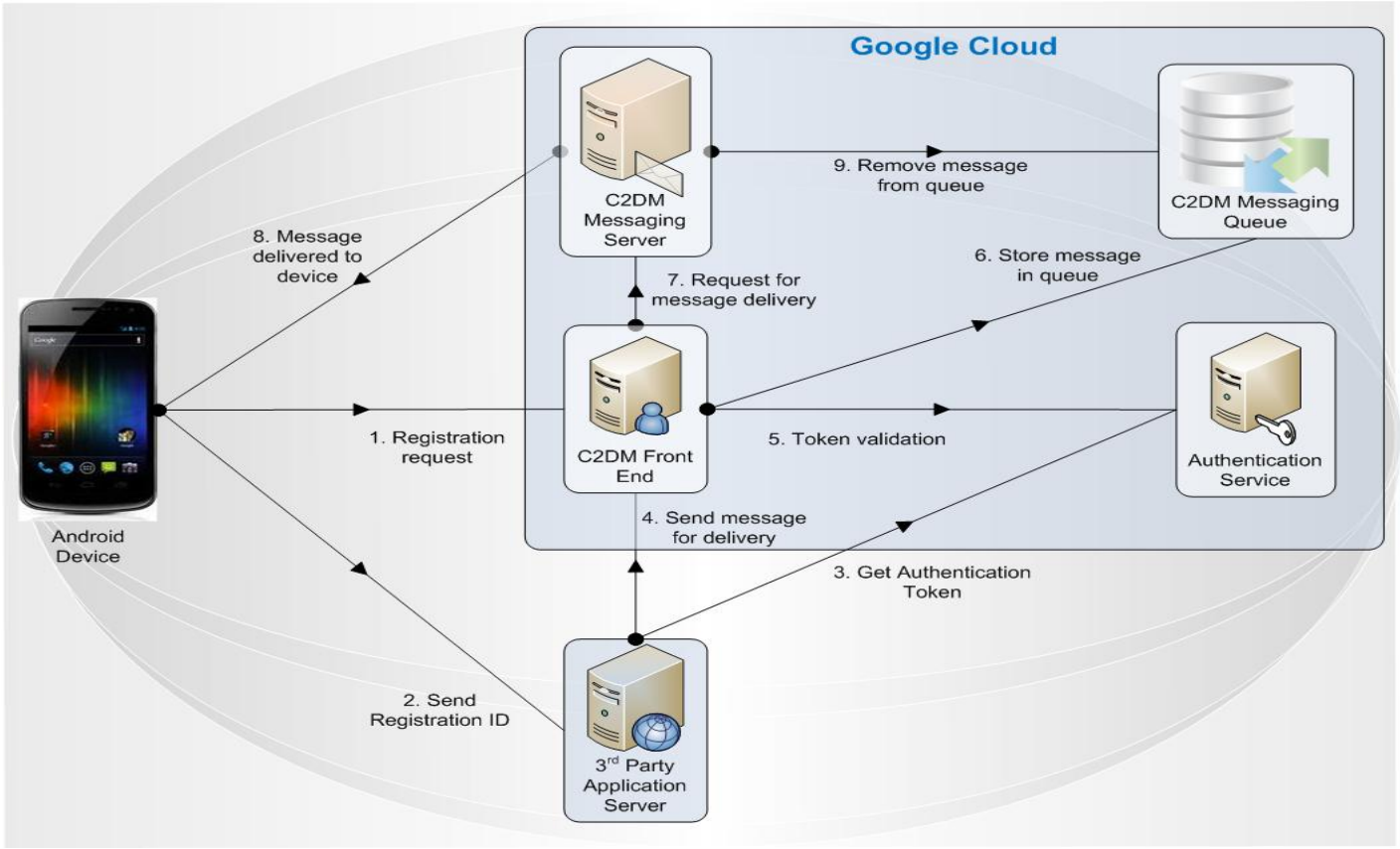
  Άρα μπορούμε να το προσαρμόσουμε στις απαιτήσεις μας.

- Δεν έχει κάποιο έτοιμο Interface σχετικά με τις ειδοποιήσεις. Αυτό είναι καλό διότι από την στιγμή που η ειδοποίηση θα φτάσει στην συσκευή μπορούμε εμείς να φτιάξουμε ότι interface θέλουμε και να την διαχειριστούμε.

.

Μειονεκτήματα του πρωτόκολλου

- Σχετικά δύσκολο στην υλοποίηση.
- Ασφάλεια μηνυμάτων. Εάν κάποιος γνωρίζει την ip και το port στα οποία τρέχει ο διαμεσολαβητής, μπορεί να συνδεθεί και να υποκλέψει τα μηνύματα.



Σχήμα 2 : Αρχιτεκτονική πρωτοκόλλου επικοινωνίας (C2DM)

3.2.5 C2DM (Cloud to Device Messaging)

Τι κάνει το framework, η android εφαρμογή (αποστολέας) στέλνει την ειδοποίηση στον server της εφαρμογής, ο server της εφαρμογής με την σειρά του στέλνει την ειδοποίηση στους C2DM servers της Google μέσω http post και τέλος οι C2DM serves στέλνουν το μήνυμα στον παραλήπτη.

Πως λειτούργει το C2DM

1. Πρώτα πρέπει κάνουμε εγγραφή της εφαρμογής μας στην υπηρεσία C2DM. Η υπηρεσία θα μας στείλει πίσω ένα κλειδί ενεργοποίησης.
2. Στην συνέχεια δίνουμε αυτό το κλειδί στον server της εφαρμογής μας.
3. Από τον server μας κάνουμε μια αίτηση γνησιότητας της εφαρμογής στην authentication service της Google. Από εκεί λαμβάνουμε μια ένδειξη γνησιότητας (ένα token)
4. Πλέον μπορούμε να χρησιμοποιήσουμε την υπηρεσία C2DM! Από τον server στέλνουμε το μήνυμα (που θα φτάσει με την μορφή ειδοποίησης) στην υπηρεσία C2DM μαζί με το κλειδί ενεργοποίησης και την ένδειξη γνησιότητας.
5. Από εκεί γίνεται ένας έλεγχος για την αυθεντικότητα των στοιχείων που στείλαμε, πάλι μέσω του authentication service.
6. Αν όλα είναι σωστά το μήνυμα μας πηγαίνει σε έναν server της google που λειτούργει σαν ουρά και αποθηκεύει όλα τα μηνύματα που δέχεται από την C2DM υπηρεσία.

7. Στην συνέχεια ενημερώνει τον Push server ότι υπάρχει ένα μήνυμα για να προωθήσει.
8. Όταν είναι διαθέσιμος αυτός ο server παίρνει το μήνυμα από την ουρά, και το στέλνει στην συσκευή του παραλήπτη. Η οποία αφυπνίζει την εφαρμογή μας εάν δεν είναι στο προσκήνιο.
9. Τέλος ο push server επικοινωνεί με την ουρά ότι μπορεί να διαγράψει το μήνυμα.

Πλεονεκτήματα του framework

- Εύκολο στην υλοποίηση.
- Χρησιμοποιεί έτσι τους πόρους του συστήματος ώστε να μην εξαντλείται γρήγορα η μπαταρία.
- Τα δεδομένα που θα στείλει ή θα λάβει η εφαρμογή θα είναι σίγουρα ασφαλή διότι οι C2DM servers υποστηρίζουν SSL[(3)].
- Η εφαρμογή μπορεί να είναι στο παρασκήνιο (αν για παράδειγμα σβήσει η οθόνη μετά από λίγη ώρα) και να λάβει μια ειδοποίηση.

Μειονεκτήματα του framework

- Το smartphone πρέπει να έχει έκδοση android 2.2 και πάνω. Αυτό πρακτικά σημαίνει ότι όποιος έχει αγοράσει συσκευή πριν το δεύτερο εξάμηνο του 2012 και δεν έχει κάνει αναβάθμιση λογισμικού δεν μπορεί να χρησιμοποιήσει την εφαρμογή (!)
- Στο smartphone πρέπει να υπάρχει η εφαρμογή Play store. Η εφαρμογή δίνετε μαζί με το λογισμικό αλλά αν για οποιοδήποτε λόγο ο χρήστης απεγκαταστήσει το play store δεν θα μπορεί να χρησιμοποιήσει την εφαρμογή μας!
- Ο χρήστης θα πρέπει να έχει Google λογαριασμό. Επίσης θα πρέπει να είναι logged in κάθε φορά που χρησιμοποιεί την εφαρμογή μας.
- Το framework δεν μας δίνει καμία εγγύηση για το αν θα φτάσουν οι ειδοποιήσεις στον προορισμό του ούτε και για την σειρά με την οποία θα φτάσουν!
- Το μέγεθος των μηνυμάτων έχει όριο 1KB. Αυτό σήμερα είναι αρκετό αλλά μετά από κάποιες εκδόσεις είμαστε σίγουροι ότι θα συνεχίσει να μας καλύπτει?
- ***** θα σταλεί ειδοποίηση παρόλο που η εφαρμογή είναι κλειστή

3.3 Ετυμηγορία

Τα άλλα

MQTT

Σχήμα 3 : Η ετυμηγορία

3.3.1 Επιλογή

Μετά από πολύ έρευνα σχετικά με τους παραπάνω τρόπους αποστολής και λήψης ειδοποιήσεων αποφασίσαμε να χρησιμοποιήσουμε το πρωτόκολλο MQTT διότι τα μειονεκτήματα από τις άλλες τεχνολογίες είναι περισσότερα. Εξάλλου αυτό το πρωτόκολλο κατασκευάστηκε με πρωταρχικό σκοπό την εξοικονόμηση πόρων και βελτιστοποίηση.

3.3.2 MQTT στα μέτρα μας

Ως γνωστόν το πρωτόκολλο MQTT είναι πλέον ευρέως διαδεδομένο για την επικοινωνία M2M (machine to machine). Για να λειτουργήσει χρειαζόμαστε και τις ανάλογες βιβλιοθήκες τόσο από πλευράς server όσο και από πλευράς client.

Υπάρχουν πολλές υλοποιήσεις αυτού του πρωτοκόλλου σε ποικίλες γλώσσες προγραμματισμού όπως C,C++,C sharp,JAVA,PERL, PHP κτλ. Από την πλευρά μας χρησιμοποιούμε Android (java στην ουσία) και PHP για να στείλουμε ειδοποιήσεις στον διαμεσολαβητή(broker) .Συνεπώς χρησιμοποιούμε και τις ανάλογες βιβλιοθήκες ,την IA92 για java και την SAM(Simple Asynchronous Messaging) [(2)] για PHP.

Να τονίσουμε ότι ως διαμεσολαβητή διαλέξαμε τον open source mosquito broker ο ποίος τρέχει ως service και περιμένει αιτήματα για να τα στείλει στις ανάλογες συσκευές. Η επικοινωνία του διαμεσολαβητή γίνεται μεταξύ των βιβλιοθηκών (διεπαφών).

SAM[(2)]: Είναι τα αρχικά από Simple Asynchronous Messaging. Παρά πολλές εφαρμογές βασίζονται στην ασύγχρονη ανταλλαγή μηνυμάτων και ουρών στις οποίες αποθηκεύονται προσωρινά τα μηνύματα αυτά μέχρι να φτάσουν στον προορισμό τους. Για όσους γνωρίζουν η php δεν έχει φτιαχτεί για τέτοιες καταστάσεις. Η βιβλιοθήκη SAM έχει να κάνει με php scripting ακριβώς γι αυτό. Παρέχει ένα API (Application Programming Interface) το οποίο δέχεται και στέλνει μηνύματα από και προς τον php server.

4 Τεκμηρίωση βάσης δεδομένων

4.1 Εισαγωγή

Για τις ανάγκες των εφαρμογών απαιτήθηκε να κρατούνται μόνιμα κάποια στοιχεία όπως ο βαθμός ικανοποίησης των χρηστών, στοιχεία λογαριασμού (username , password) κ.α . Συνεπώς χρειάστηκε να υλοποιήσουμε και να σχεδιάσουμε μια σχεσιακή βάση δεδομένων η οποία θα αποθηκεύει τέτοιες πληροφορίες.

Υπήρχαν πολλές λύσεις που μπορούσαμε να διαλέξουμε όπως MICROSOFT SQL SERVER, ORACLE, POSTGRESQL. Αλλά εμείς επιλέξαμε MySQL λόγω της ευχρηστίας και της δυναμικότητας που προσφέρει. Μη ξεχνάμε ότι η MySQL είναι δημοφιλής βάση δεδομένων για διαδικτυακά προγράμματα και ιστοσελίδες. Χρησιμοποιείται σε κάποιες από τις πιο διαδεδομένες διαδικτυακές υπηρεσίες, όπως το YouTube, η Wikipedia, το Google, το Facebook και το Twitter.

4.2 Σχεδιασμός

Η διαδικασία του σωστού σχεδιασμού για να υλοποιηθεί μια βάση δεδομένων απαιτεί την ακολουθία κάποιων συγκεκριμένων βημάτων, που χωρίζονται σε 3 επίπεδα. Αναφέρουμε τα εξής :

- Φυσικό επίπεδο

  Στο σημείο αυτό καθορίζουμε τον τρόπο με τον οποίο τα δεδομένα θα αποθηκεύονται. Δηλαδή τους τύπους δεδομένων των πληροφοριών, τους τυχόν αλγορίθμους διαχείρισης δεδομένων. κ.α

- Λογικό επίπεδο

  Εδώ αγνοούμε τις λεπτομέρειες και πολυπλοκότητα και ασχολούμαστε με τον καθορισμό οντοτήτων και των μεταξύ τους συσχετίσεων.

- Επίπεδο όψης

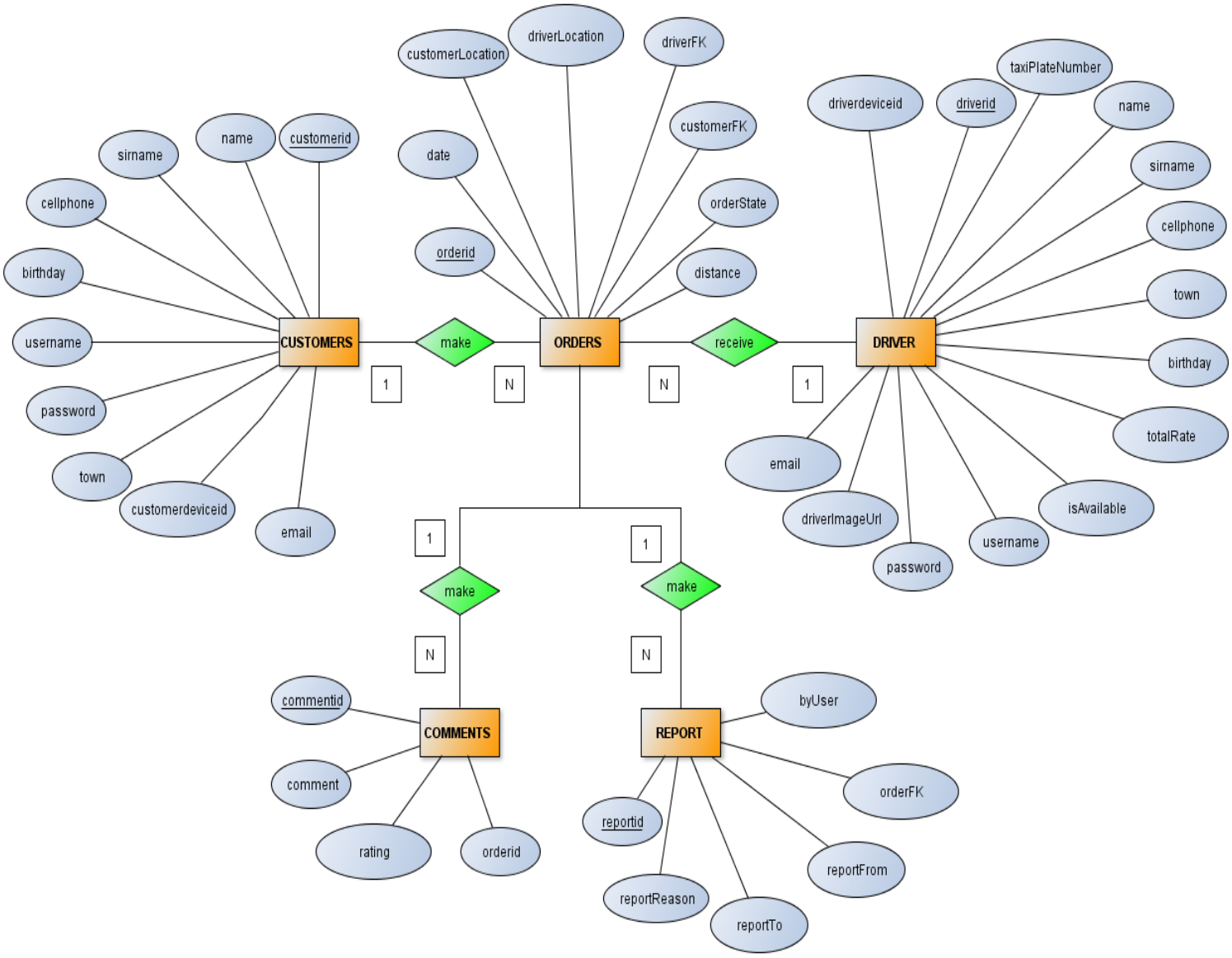
Στο επίπεδο όψης καθορίζουμε συνήθως σε ποια δεδομένα θα έχουν πρόσβαση οι χρήστες. Σε μεγάλες βάσεις δεδομένων αυτό είναι βολικό για λόγους ασφαλείας και ακεραιότητας των δεδομένων. Στην δική μας περίπτωση χρειάζεται όλοι οι χρήστες (δηλ και οδηγοί και πελάτες) να επεμβαίνουν στα ίδια δεδομένα.

Στο επόμενο σχήμα απεικονίζονται τα επίπεδα του σχεδιασμού :

Σχήμα 1 : Επίπεδα βάσης δεδομένων

Διάγραμμα

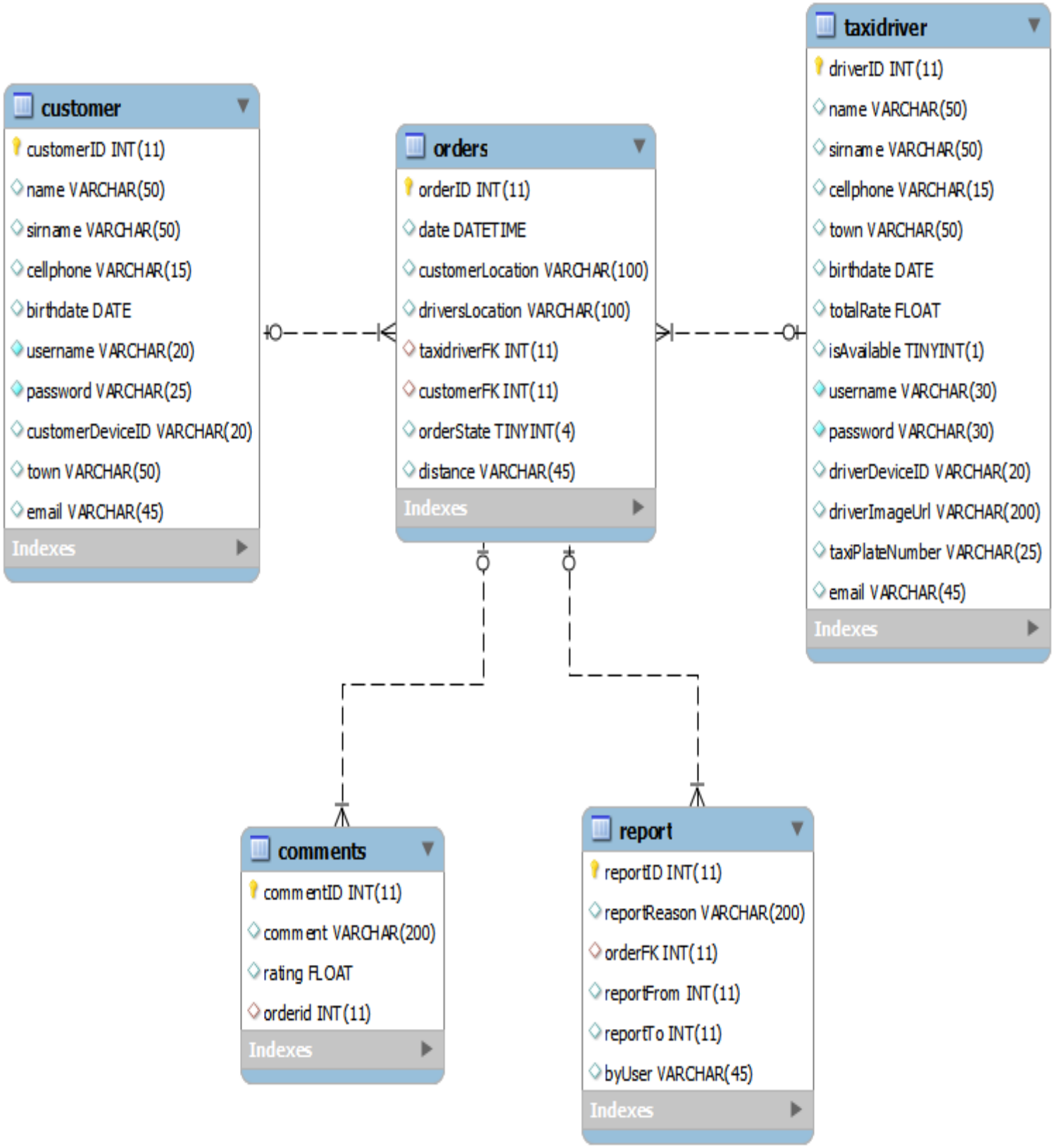
Στο επόμενο σχήμα φαίνεται το διάγραμμα οντοτήτων-συσχετίσεων (E-R) της βάσης δεδομένων.



Σχήμα 2: Διάγραμμα οντοτήτων συσχετίσεων

Οι λόγοι πληθικότητας είναι όπως φαίνονται στο σχήμα. Ένας πελάτης μπορεί να κάνει πολλές παραγγελίες , όπως και ένας οδηγός μπορεί να δεχθεί πολλές κλήσεις. Σε κάθε παραγγελία μπορεί να γίνουν πολλά σχόλια όπως και πολλές αναφορές.

Εφόσον έχουμε σχεδιάσει το μοντέλο οντοτήτων συσχετίσεων , το άλλο στάδιο είναι να το μετατρέψουμε σε σχεσιακό μοντέλο με πίνακες. Φαίνεται παρακάτω :



Σχήμα 3 : Σχεσιακό διάγραμμα βάσης δεδομένων

5 Τεκμηρίωση εφαρμογής server

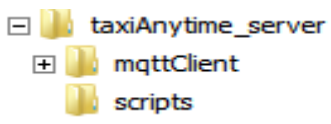
5.1 Εισαγωγή

Όπως είπαμε για τις ανάγκες των εφαρμογών ήταν αναγκαίο να κρατάμε μόνιμα κάποια στοιχεία των χρηστών και των παραγγελιών. Συνεπώς ήταν απαραίτητο να υλοποιήσουμε την κατάλληλη εφαρμογή για να επικοινωνεί μέσω του server με μια βάση δεδομένων για να μπορούν να γίνονται τυπικές ενέργειες όπως εισαγωγή, επιλογή, διαγραφή και ενημέρωση στοιχείων της βάσης. Στην ουσία την εφαρμογή μας την αποτελούν php scripts που εκτελούν αυτές τις ενέργειες.

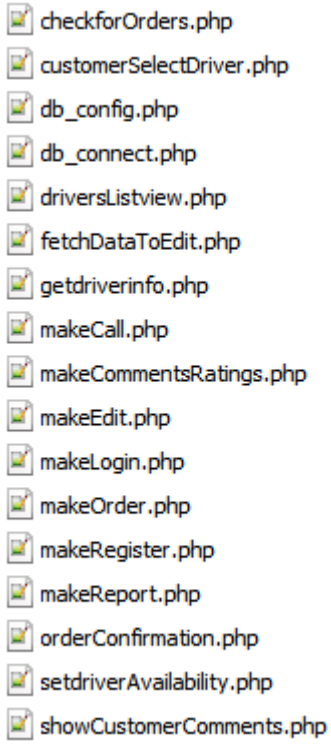
5.2 Κύριο μέρος

Όπως φαίνεται στο σχήμα παρακάτω στον βασικό κατάλογο του server έχουμε 2 υποκαταλόγους τον /scripts και τον /mqttClient . Ο πρώτος (scripts) περιέχει όλα τα αρχεία php για την επικοινωνία με τη βάση δεδομένων, ενώ ο δεύτερος (mqttClient) περιέχει την βιβλιοθήκη SAM (simple asynchronous messages) σε συνδυασμό μαζί με τα δικά μας php scripts για την αποστολή ειδοποιήσεων στον διαμεσολαβητή mosquito broker. Έπειτα ο διαμεσολαβητής θα ξέρει σε ποιες συγκεκριμένες συσκευές θα στείλει την ειδοποίηση.

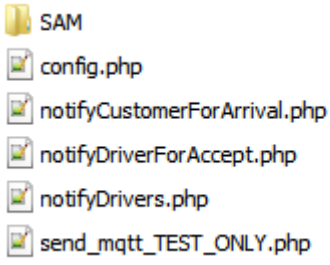
Να σημειώσουμε πως διαχωρίσαμε και την κάθε διαφορετική ενέργεια που θέλουμε να κάνουμε σε διαφορετικά αρχεία php. Έτσι εξασφαλίσουμε καλύτερη δομή και γίνεται πιο κατανοητό σε κάποιον τρίτο που θέλει να διαβάσει τον κώδικα ή να προσθέσει/ αλλάξει κάτι.



Εικόνα 1: Βασικός κατάλογος server



Εικόνα 2 : Υποκατάλογος scripts



Εικόνα 3 : Υποκατάλογος mqttClient

Στη συνέχεια θα αναλύσουμε τη λειτουργία του κάθε αρχείου. Να επισημάνουμε ότι όλα τα αρχεία php καλούνται μέσα από τις εφαρμογές Android μέσω http requests δίνοντας τις κατάλληλες εισόδους και παράγοντας κάποια αποτελέσματα.

5.2.1 Υποκατάλογος /scripts

db_config.php

Αρχείο που ορίζουμε τις μεταβλητές για τη σύνδεση με τη βάση δεδομένων. Είναι μεταβλητές για όνομα χρήστη που θα κάνει σύνδεση στη βάση, κωδικός, το όνομα της βάσης και την ip του server που θα συνδεθούμε ή localhost αν συνδεόμαστε τοπικά.

db_connect.php

Κλάση υπεύθυνη για την διαχείριση της βάσης (σύνδεση/ αποσύνδεση). Απαιτείται να υπάρχει το προηγούμενο αρχείο(db_config) για να πραγματοποιηθεί η σύνδεση. Το αρχείο db_connect.php χρησιμοποιείται σε όλα τα υπόλοιπα αρχεία (γίνεται import) για να μπορούν να πραγματοποιήσουν συναλλαγές με τη βάση δεδομένων μας.

fetchDataToEdit.php

Περιγραφή:

Αρχείο υπεύθυνο για το φόρτωμα των στοιχείων που θα αλλάξουν τόσο του οδηγού όσο και του πελάτη. Δηλαδή όταν ο χρήστης θα πάει στην φόρμα αλλαγής προφίλ θα φαίνονται γεμισμένα τα τρέχοντα στοιχεία του. Έτσι του δίνεται η δυνατότητα να αλλάξει μερικά ή όλα. Τα στοιχεία που μπορεί να αλλάξει ο πελάτης είναι το όνομα, επώνυμο, κινητό τηλ , πόλη, ενώ ο οδηγός μπορεί να αλλάξει τα ίδια στοιχεία με ένα επιπλέον, τον αριθμό πινακίδας.

Είσοδοι:

- Το id της συσκευής
- Τον τύπο πελάτη (“customer” ή “driver”)

Διαδικασία/Εκτέλεση :

Επιλέγουμε τα στοιχεία που θέλουμε να φορτώσουμε (δηλ όνομα, επίθετο κτλ) βάσει του συγκεκριμένου id που δόθηκε. Έπειτα γεμίζουμε αυτά τα στοιχεία σε μορφή JSON.

Αποτέλεσμα :

Τα στοιχεία σε μορφή JSON, σε διαφορετική περίπτωση success 0.

Παράδειγμα :


```
{
    "editDetails": [
        {
            "name": "ΚΥΡΙΑΚΟΣ",
            "sirname": "ΚΥΡΙΑΚΟΥ",
            "cellphone": "6945000000",
            "town": "ΣΕΡΡΕΣ",
            "taxiPlateNumber": "ΕΡΕ-4534"
        }
    ],
    "success": 1
}
```

getdriverinfo.php

Περιγραφή:

Αρχείο για να παίρνουμε τα στοιχεία των οδηγών που είναι διαθέσιμοι για να στέλνουμε σε αυτούς ειδοποιησείς.

Διαδικασία/Εκτέλεση :

Επιλέγουμε στοιχεία όπως όνομα, συνολικό rating, id συσκευής με κριτήριο isAvailable =1 .

Αποτέλεσμα :

Τα στοιχεία σε μορφή JSON, σε διαφορετική περίπτωση success 0.

Παράδειγμα :


```
{
    "drivers": [
        {
            "sirname": "KYPIAKOY",
            "rate": "3",
            "driverImageUrl":
"http://192.168.2.100/taxiAnytime_server/androidBot.jpg",
            "driverDeviceID": "/357777961700933"
        }
    ],
    "success": 1
}
```


makeCall.php

Περιγραφή:

Αρχείο που παίρνει από τη βάση τον αριθμό κινητού ενός συγκεκριμένου χρήστη, για να μπορούμε να κάνουμε κλήση ή και να στείλουμε sms από τις εφαρμογές android.

Είσοδοι:

- Το id της συσκευής
- Τον τύπο πελάτη (“customer” ή “driver”)

Διαδικασία/Εκτέλεση :

Επιλέξουμε με ερώτημα προς τη βάση το πεδίο cellphone που είναι ο αριθμός κινητού που αντιστοιχεί στο συγκεκριμένο id συσκευής που δόθηκε. Έπειτα επιστέφουμε αυτό το αποτέλεσμα σε μορφή JSON.

Αποτέλεσμα :

Τα στοιχεία σε μορφή JSON, σε διαφορετική περίπτωση success 0.

Παράδειγμα :


```
{
    "cellphone": "6945000000",
    "success": 1
}
```


makeCommentsRatings.php

Περιγραφή:

Αρχείο που καταχωρεί στη βάση τα σχόλια και το rating που έκανε ο χρήστης για τη συγκεκριμένη παραγγελία.

Είσοδοι:

- Το rating
- Το σχόλιο
- Το id της συσκευής του οδηγού που διάλεξε
- Το id της τρέχουσας παραγγελίας (πίνακας orders)
- Το id της συσκευής του πελάτη που έκανε σχόλιο

Διαδικασία/Εκτέλεση :

Ελέγχουμε για την ορθότητα των δεδομένων. Έπειτα εισάγουμε στον πίνακα comments τα σχόλια και το rating. Εφόσον έχουμε επιτυχή εγγραφή ενημερώνουμε το συνολικό rating του οδηγού βάσει τον πίνακα comments. Δηλαδή στον πίνακα comments αθροίζουμε το συνολικό rating από όλες τις εγγραφές του πίνακα που αντιστοιχούν στον συγκεκριμένο οδηγό.

Αποτέλεσμα :

Τα στοιχεία σε μορφή JSON success 1 σε επιτυχία, αλλιώς success 0

Παράδειγμα :


```
{
    "success": 1,
    "message": "successfully done"
}
```


Διαφορετικά :


```
{
    "success": 0,
    "message": "not successfull "
}
```

makeEdit.php

Περιγραφή:

Αρχείο για την πραγματοποίηση update των στοιχείων του οδηγού ή του πελάτη.

Είσοδοι:

- Το id της συσκευής
- Τον τύπο πελάτη (“customer” ή “driver”)
- Το κινητό τηλέφωνο
- Τον κωδικό (password)
- Την πόλη
- Τον αριθμό πινακίδας (στην περίπτωση του οδηγού μόνο)

Διαδικασία/Εκτέλεση :

Εφόσον ελέγχεται η ορθότητα των εισερχόμενων δεδομένων, γίνεται η διαδικασία update . Αν αφορά τον πελάτη ενημερώνεται ο πίνακας customer ενώ αλλιώς ο πίνακας taxidriver.

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία, αλλιώς success 0.

Παράδειγμα :

```
{
    "success": 1,
    "message": "successfully done"
}
```

Διαφορετικά :

```
{
    "success": 0,
    "message": "not successfull "
}
```

makeLogin.php

Περιγραφή:

Αρχείο για την πραγματοποίηση εισόδου χρήστη. Γίνεται πιστοποίηση αν υπάρχει ο χρήστης στη βάση.

Είσοδοι:

- Το όνομα χρήστη (username)
- Ο κωδικός (password)
- Το id της συσκευής
- Το είδος του χρήστη (“customer” ή “driver”)

Διαδικασία/Εκτέλεση :

Ελέγχουμε αν υπάρχει το username και το password στη βάση για να διαπιστώσουμε ότι πρόκειται για εγγεγραμμένο χρήστη. Επίσης γίνεται έλεγχος για το αν ο χρήστης είναι τιμωρημένος, δηλαδή αν τον έκαναν αναφορά πάνω από 5 φορές. (Τον αριθμό αυτόν τον ορίσαμε εμείς.) Αν τελικά δώσει λάθος στοιχεία ή είναι τιμωρημένος δεν γίνεται είσοδος στην εφαρμογή. Το πόσες φορές τον έκαναν report το βρίσκουμε αθροίζοντας της εγγραφές του πίνακα report για το συγκεκριμένο id .

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία, αλλιώς success 0.

Παράδειγμα :

```
{
    "success": 1,
    "message": "successfully logged in"
}
```

Διαφορετικά :

```
{
    "success": 0,
    "message": "wrong login or reported "
}
```

makeOrder.php

Περιγραφή:

Αρχείο για την καταχώρηση της νέας παραγγελίας στη βάση. (Δημιουργία νέας εγγραφής στον πίνακα orders)

Είσοδοι:

- Γεωγραφικό μήκος πελάτη
- Γεωγραφικό πλάτος πελάτη
- Γεωγραφικό μήκος οδηγού
- Γεωγραφικό πλάτος οδηγού
- Το id της συσκευής του πελάτη
- Το id της συσκευής του οδηγού
- Η απόσταση μεταξύ οδηγού και πελάτη που έγινε η παραγγελία

Διαδικασία/Εκτέλεση :

Εφόσον ελέγχονται τα δεδομένα για την ορθότητα τους, κάνουμε την εισαγωγή (insert) στον πίνακα orders. Στο σημείο αυτό η παραγγελία μας έχει είναι σε κατάσταση 1. (Δηλαδή έτοιμη , orderState =1).

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία, αλλιώς success 0.

Παράδειγμα :

```
{
    "success": 1,
    "message": "order successfully inserted"
}
```

Διαφορετικά :

```
{
    "success": 0,
    "message": "error"
}
```

makeRegister.php

Περιγραφή:

Αρχείο για την εγγραφή νέου χρήστη, είτε αφορά νέο πελάτη είτε νέο οδηγό.

Είσοδοι:

- Όνομα
- Επώνυμο
- Κινητό
- Ημ. Γενεθλίων
- Όνομα χρήστη (username)
- Κωδικός (password)
- E-mail
- Πόλη
- Το id της συσκευής
- Το είδος του χρήστη (“customer” ή “driver”)

Διαδικασία/Εκτέλεση :

Αρχικά ελέγχουμε αν όλα τα πεδία είναι ορισμένα, δηλαδή αν έχουν κάποια τιμή. Στη συνέχεια ελέγχουμε αν υπάρχει ήδη κάποιος χρήστης με στοιχεία όπως το username ή mail. Αν όλα είναι σωστά γίνεται η εγγραφή (insert στον πίνακα customer ή driver ανάλογα). Σε διαφορετική περίπτωση αν ο χρήστης υπάρχει ήδη ή δεν έχει οριστεί κάποιο πεδίο έχουμε και ανάλογο μήνυμα.

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία εγγραφής, αλλιώς success 0.

Παράδειγμα :

```
{
    "success": 1,
    "message": "registered successfully"
}
```

Διαφορετικά :

```
{
    "success": 0,
    "message": "error"
}
```

makeReport.php

Περιγραφή:

Αρχείο για την δημιουργία νέας αναφορά χρήστη.

Είσοδοι:

- Οι λόγοι που έγινε η αναφορά
- Το id της τρέχουσας παραγγελίας (id πίνακα orders)
- Από ποιον έγινε η αναφορά (πίνακας orders το id taxidriverFK ή customerFK ανάλογα)
- Σε ποιον έγινε αναφορά (πίνακας orders το id taxidriverFK ή customerFK ανάλογα)
- Ο τύπος του χρήστη που έκανε την αναφορά (“driver” ή “customer”)

Διαδικασία/Εκτέλεση :

Ενώ έχουμε ελέγξει για το αν είναι ορισμένα τα δεδομένα που εισήχθησαν, τα κάνουμε εισαγωγή (insert) στον πίνακα report δημιουργώντας μια νέα εγγραφή .

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία αναφοράς, αλλιώς success 0.

Παράδειγμα :

```
{
    "success": 1,
    "message": "successfully reported"
}
```

Διαφορετικά :

```
{
    "success": 0,
    "message": "not reported"
}
```

orderConfirmation.php

Περιγραφή:

Αρχείο για την ενημέρωση της κατάστασης παραγγελίας. Αρχικά όπως είπαμε η παραγγελία έχει κατάσταση 1, που σημαίνει ότι μόλις έχει δημιουργηθεί. Έπειτα αν την αποδεχθεί ο οδηγός αλλάζει σε κατάσταση 2 και τέλος αν την αποδεχθεί και ο πελάτης τότε η τελική κατάσταση είναι 3. Μια παραγγελία θεωρείται ολοκληρωμένη μόνο αν έχει κατάσταση 3.

Είσοδοι:

- Ο τύπος του χρήστη (“driver” ή “customer”)
- Το id της συσκευής του χρήστη που έκανε την αποδοχή
- Το id της επιλεγμένης παραγγελίας (πίνακας orders πεδίο orderid)

Διαδικασία/Εκτέλεση :

Ελέγχοντας αρχικά για τα εισερχόμενα δεδομένα, αν πρόκειται για οδηγό κάνουμε update την κατάσταση της παραγγελίας σε 2. (πίνακας orders πεδίο orderState). Ενώ αν πρόκειται για πελάτη κάνουμε update την κατάσταση της παραγγελίας σε 3.

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία ενημέρωσης , αλλιώς success 0.

Παράδειγμα :

```
{
    "success": 1,
    "message": "successfully updated"
```

setdriverAvailability.php

Περιγραφή:

Αρχείο για να αλλάζει την διαθεσιμότητα του οδηγού σε ενεργός η όχι. Συνεπώς μόνο αν είναι ενεργός θα του έρχονται αιτήματα αλλιώς δεν θα γίνεται τίποτα.

Είσοδοι:

- Η διαθεσιμότητα (true ή false)
- Το id της συσκευής του οδηγού

Διαδικασία/Εκτέλεση :

Γίνεται update το πεδίο isAvailable από τον πίνακα taxidriver με την τιμή που έχουμε ως είσοδο.

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία αλλαγής διαθεσιμότητας, αλλιώς success 0.

Παράδειγμα :

```
{
    "available": "true",
    "success": 1
}
```

Αλλιώς

```
{
    "message": "error",
    "success": 0
}
```

showCustomerComments.php

Περιγραφή:

Αρχείο για την προσκόμιση όλων των σχολίων από άλλους πελάτες για τον συγκεκριμένο οδηγό.

Είσοδοι:

- Το id της συσκευής του οδηγού που επιλέχθηκε.

Διαδικασία/Εκτέλεση :

Επιλέγουμε για να εμφανίσουμε όλους τους πελάτες που σχολίασαν τον συγκεκριμένο οδηγό . Τα στοιχεία που φαίνονται είναι το σχόλια και το ονοματεπώνυμο των άλλων χρηστών.

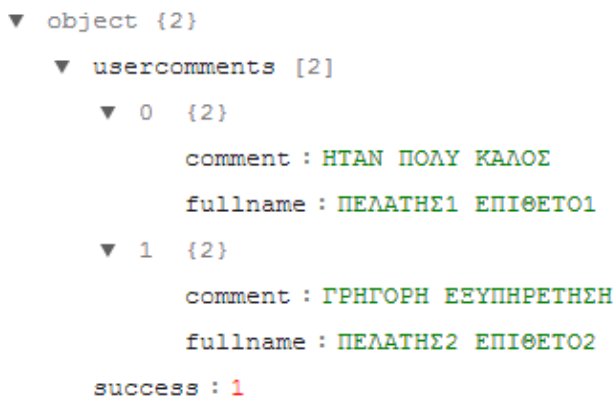
Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία, αλλιώς success 0.

Παράδειγμα :


```
{
    "usercomments": [
        {
            "comment": "ΗΤΑΝ ΠΟΛΥ ΚΑΛΟΣ",
            "fullname": "ΠΕΛΑΤΗΣ1 ΕΠΙΘΕΤΟ1"
        },
        {
            "comment": "ΓΡΗΓΟΡΗ ΕΞΥΠΗΡΕΤΗΣΗ",
            "fullname": "ΠΕΛΑΤΗΣ2 ΕΠΙΘΕΤΟ2"
        }
    ],
    "success": 1
}
```


Η μορφή του παραπάνω ως JSON αντικείμενο φαίνεται παρακάτω :



checkforOrders.php

Περιγραφή:

Αρχείο το οποίο ελέγχει για το αν υπάρχει η παραγγελία που έκανε ο συγκεκριμένος πελάτης. Δηλαδή αν οι παραγγελίες είναι στο αρχικό τους στάδιο. (orderState = 1)

Είσοδοι:

- Το id της συσκευής του πελάτη

Διαδικασία/Εκτέλεση :

Μετράμε το πλήθος των εγγραφών που πληρούν αυτό το κριτήριο (κατάσταση παραγγελίας =1).

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 αν έχει δημιουργηθεί η παραγγελία, αλλιώς success 0.

customerSelectDriver.php

Περιγραφή:

Αρχείο που επιστρέφονται στον οδηγό η τοποθεσία του πελάτη που τον επέλεξε. Τα στοιχεία αυτά είναι γεωγραφικό μήκος και πλάτος.

Είσοδοι:

- Το id της συσκευής του πελάτη.
- Το id της επιλεγμένης παραγγελίας (πεδίο orderid του πίνακα orders)

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία, αλλιώς success 0

Παράδειγμα :

```
{
    "customerLat": "41.23543",
    "customerLon": "24.56778"
    "driverDevID": "/455758689645867"
    "success": 1
}
```

driversListview.php

Περιγραφή:

Αρχείο που εμφανίζει όλους τους οδηγούς που έδειξαν ενδιαφέρον για τον συγκεκριμένο πελάτη.

Είσοδοι:

- Το id της συσκευής του πελάτη

Διαδικασία/Εκτέλεση :

Τα στοιχεία που εμφανίζονται είναι από τον πίνακα orders και είναι : όνομα οδηγού, συνολικό rating, απόσταση και πινακίδα οχήματος ταξί.

Αποτέλεσμα :

Τα στοιχεία απόκρισης σε μορφή JSON ,success 1 σε επιτυχία, αλλιώς success 0

Παράδειγμα :


```
{
    "drivers": [
        {
            "orderid": "14",
            "name": "ΠΑΠΑΔΟΠΟΥΛΟΣ ΘΥΜΙΟΣ",
            "rate": "4",
            "distance": "791.6996",
            "driverImageUrl":
"http://192.168.2.100/taxiAnytime_server/androidBot.jpg",
            "taxiPlateNumber": "ERN-6879",
            "driverDeviceID": "/357717577700933"
        }
    ],
    "success": 1
}
```


Αλλιώς


```
{
    "message": "no drivers!!",
    "success": 0
}
```


5.2.2 Υποκατάλογος /mqttClient

Config.php

Περιγραφή:

Αρχείο ρυθμίσεων για τον διαμεσολαβητή mosquito . Ορίζουμε την IP του μηχανήματος που τρέχει το mosquito καθώς και την κατάλληλη θύρα (PORT).

notifyCustomerForArrival.php

Περιγραφή:

Αρχείο για την ειδοποίηση των πελατών ότι ο οδηγός που επέλεξαν έχει φτάσει.

Είσοδοι:

- Το id της επιλεγμένης παραγγελίας (πεδίο orderid του πίνακα orders)

Διαδικασία/Εκτέλεση :

Αρχικά πραγματοποιείται σύνδεση τόσο με τη βάση όσο και με τον διαμεσολαβητή (mosquitto). Έπειτα συλλέγουμε από τη βάση το id της συσκευής του πελάτη που θα ειδοποιήσουμε και αμέσως μετά στέλνουμε σε αυτό το id την ειδοποίηση μέσω του διαμεσολαβητή. Τέλος κλείνουμε τις ανοικτές συνδέσεις.

notifyDriverForAccept.php

Περιγραφή:

Αρχείο αρμόδιο για την αποστολή ειδοποίησης στον οδηγό ότι ο πελάτης που είχε επιλέξει πριν τον έχει διαλέξει.

Είσοδοι:

- Γεωγραφικό μήκος πελάτη
- Γεωγραφικό πλάτος πελάτη
- Το id της συσκευής του οδηγού που θα σταλεί ειδοποίηση
- Το id της επιλεγμένης παραγγελίας (πεδίο orderid του πίνακα orders)

Διαδικασία/Εκτέλεση :

Πρώτα γίνεται σύνδεση με το διαμεσολαβητή (mosquitto) και στην συνέχεια στέλνεται μήνυμα (ειδοποίηση) στον οδηγό βάσει του id της συσκευής του.

Η ειδοποίηση έχει τη μορφή type.message.lat.lon.deviceid.orderid

Επεξήγηση :

type : έχει τιμή 2 που σημαίνει ότι έγινε αποδοχή (θα δούμε μετά και πότε έχει τιμή 1)

message : Το μήνυμα που θα φαίνεται στον χρήστη.

lat: το γεωγραφικό μήκος

lon : το γεωγραφικό πλάτος

deviceid: το μοναδικό αναγνωριστικό της συσκευής

orderid : το id της επιλεγμένης παραγγελίας

Το μόνο που βλέπει ο χρήστης από αυτό είναι το message, τα άλλα στοιχεία τα επεξεργαζόμαστε εσωτερικά στην εφαρμογή android για να εμφανίσουμε τις συντεταγμένες κτλ.

notifyDrivers.php

Περιγραφή:

Αρχείο για την πραγματοποίηση ειδοποίησης σε όλους τους ενεργούς οδηγούς ότι υπάρχει νέος πελάτης για παραγγελία.

Είσοδοι:

- Γεωγραφικό μήκος πελάτη
- Γεωγραφικό πλάτος πελάτη
- Το id της συσκευής του πελάτη

Διαδικασία/Εκτέλεση :

Πρώτα γίνεται σύνδεση με το διαμεσολαβητή (mosquitto) και στην συνέχεια στέλνεται μήνυμα (ειδοποίηση) στους διαθέσιμους οδηγούς βάσει του id της συσκευής τους.

Η ειδοποίηση έχει τη μορφή type.message.lat.lon.deviceid

Επεξήγηση :

type : έχει τιμή 1 που σημαίνει ότι θα σταλεί νέα ειδοποίηση (συνολικά έχει δύο τιμές)

message : Το μήνυμα που θα φαίνεται στον χρήστη.

lat: το γεωγραφικό μήκος

lon : το γεωγραφικό πλάτος

deviceid: το μοναδικό αναγνωριστικό της συσκευής

Πριν σταλεί η ειδοποίηση καθαρίζουμε και τον πίνακα orders από μη ολοκληρωμένες παραγγελίες του συγκεκριμένου χρήστη ώστε να μην επιλεγεί κάποια κατά λάθος.

Να θυμίσουμε ότι μια παραγγελία έχει τρία στάδια.

1. Αρχικά δημιουργείται (στάδιο 1)
2. Αν την έχει αποδεχθεί ο οδηγός (στάδιο 2)
3. Αν την έχει αποδεχθεί και ο πελάτης (στάδιο 3)

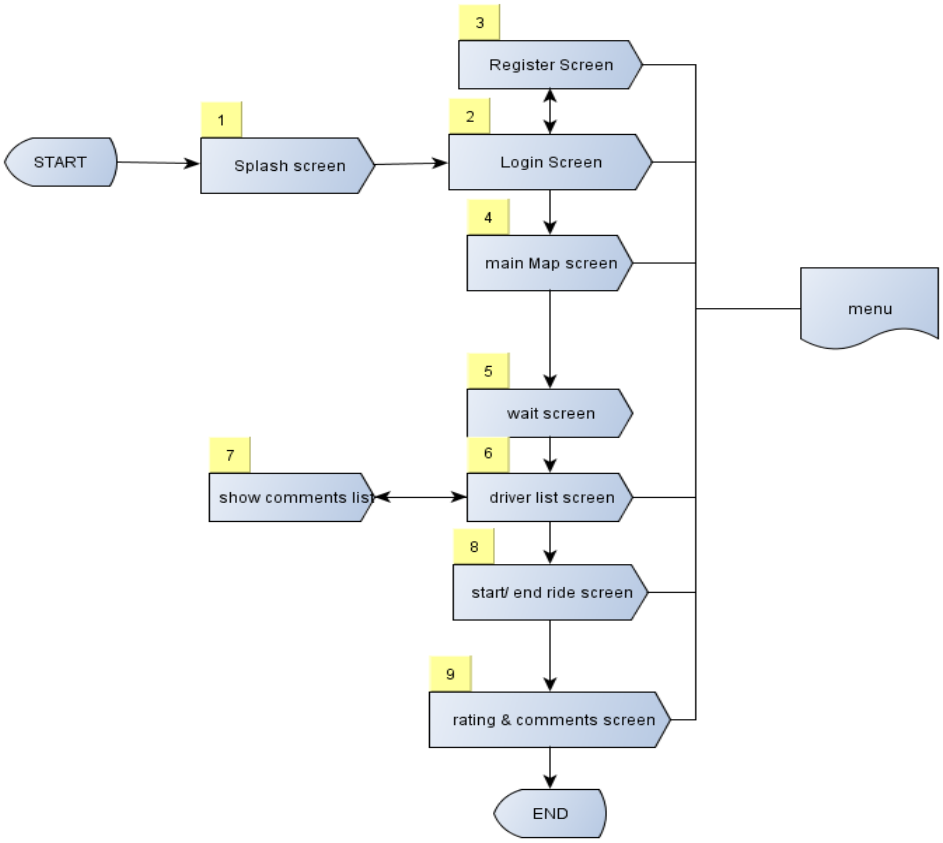
Έτσι αυτές που είναι στο στάδιο 1 ή 2 δεν μας ενδιαφέρουν , συνεπώς αυτές τις καθαρίζουμε από τη βάση και έτσι θα έχουμε μόνο τις ολοκληρωμένες.

6 Τεκμηρίωση εφαρμογής πελάτη

6.1 Εισαγωγή

Στην ενότητα αυτή θα αναλύσουμε για τις οθόνες που συναντά ο πελάτης στην εφαρμογή και το πώς δουλεύουν από πίσω από πλευράς κώδικα. Θα σταθούμε στα βασικότερα και κυριότερα σημεία που κρίνεται απαραίτητο να σχολιαστούν.

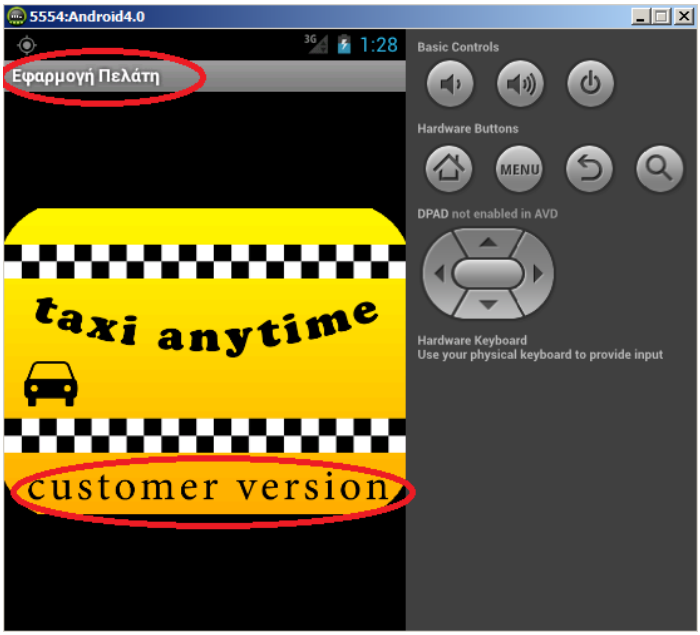
Να σημειώσουμε ότι χρησιμοποιήσαμε την βιβλιοθήκη mqtt.jar και το αντίστοιχο push service (PushService.java) που αναπτύχθηκαν από τρίτους κατασκευαστές (πρωτόκολλο της IBM) , απλά το προσαρμόσαμε στα μέτρα μας χρησιμοποιώντας κατάλληλα την μέθοδο showNotification(…) για το πώς θα εμφανίζονται οι ειδοποιήσεις.



Σχήμα: Απεικόνιση δραστηριοτήτων (activities) εφαρμογής πελάτη

6.2 Αρχική οθόνη (splash screen)

Όταν ο πελάτης ξεκινήσει την εφαρμογή του εμφανίζεται μια εισαγωγική οθόνη . Στο σημείο αυτό ο πελάτης μπορεί να δει ποια έκδοση της εφαρμογής έχει κατεβάσει (driver version ή customer version) , ενώ στο παρασκήνιο γίνεται έλεγχος για το αν είναι εφικτή η σύνδεση με το internet (τους servers της εφαρμογής) και αν είναι ανοικτό το 3G ,wifi και GPS . Σε περίπτωση που κάποιο από τα παραπάνω είναι κλειστό τότε ειδοποιείται ο χρήστης με κατάλληλο μήνυμα ώστε να προβεί στις κατάλληλες ρυθμίσεις.



Εικόνα 1 : Εισαγωγική οθόνη (splash screen)

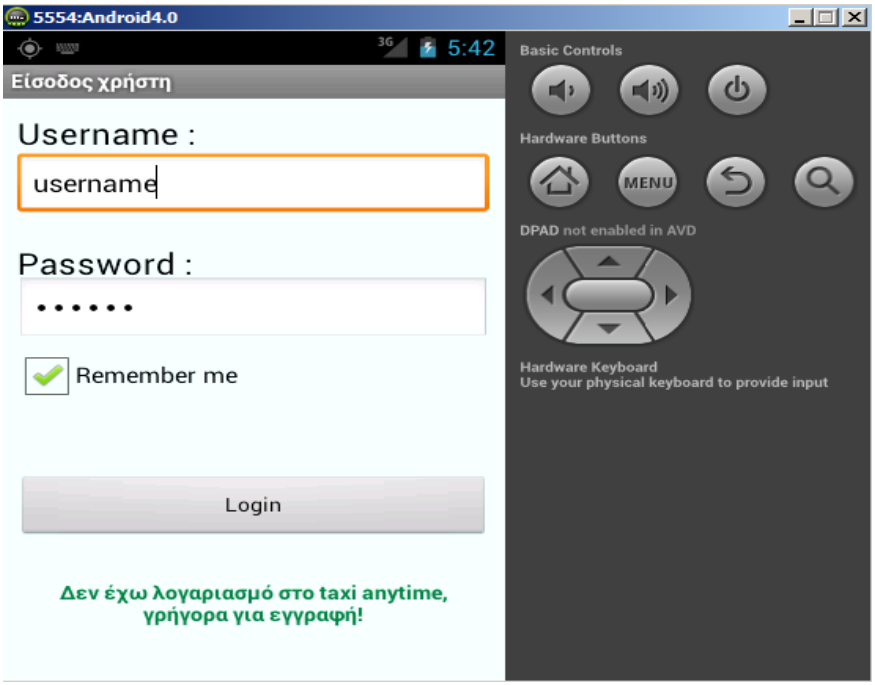
Να σημειώσουμε ότι η εισαγωγική οθόνη έχει μια διάρκεια ζωής που γίνονται όλα αυτά (ορίσαμε 3 sec) και μετά μεταβαίνουμε στην επόμενη οθόνη.

Όλα τα παραπάνω τρέχουν από το πακέτο app.taxiAnytimeCustomer.Common και τα αρχεία CustomerSplashScreen.java όπου εδώ παράγεται η οθόνη και συνδέεται με το αρχείο ConnectionDetectorTask.java στο οποίο γίνεται ο έλεγχος που προαναφέραμε καθώς και με το αρχείο ShowAlertMessage.java για την εμφάνιση μηνυμάτων στο χρήστη.

Για περισσότερες πληροφορίες σχετικά με τον κώδικα βλ το πακέτο app.taxiAnytimeCustomer.Common και τα αρχεία CustomerSplashScreen.java, ConnectionDetectorTask (σε παράρτημα να μπει)

6.3 Οθόνη Εισόδου στην εφαρμογή (login screen)

Σε αυτή την οθόνη ο χρήστης καλείται να δώσει username και password ώστε να μπει στην εφαρμογή. Επίσης έχει την δυνατότητα να αποθηκεύει τα στοιχεία εισαγωγής του ώστε να μη χρειάζεται να τα ξανά- πληκτρολογεί κάθε φορά που ανοίγει την εφαρμογή. Σε περίπτωση που δεν έχει κάνει εγγραφεί στην εφαρμογή στο κάτω μέρος της οθόνης μπορεί να επιλέξει να το κάνει.



Εικόνα 2 :Οθόνη εισόδου χρήστη

Τα δεδομένα που έδωσε ο χρήστης στέλνονται στον server όπου εκεί γίνεται η ταυτοποίηση του. Εάν ο χρήστης υπάρχει και έχει δώσει σωστά στοιχεία πατώντας το login μεταβαίνει στην κύρια οθόνη της εφαρμογής. Σε διαφορετική περίπτωση εμφανίζεται κατάλληλο μήνυμα και παραμένει στη login screen.

Όλα τα παραπάνω τρέχουν από το πακέτο app.taxiAnytimeCustomer.Common και το αρχείο LoginActivity.java. από πλευράς android και από το script makeLogin.php του server.

Το σημαντικότερο σημείο του κώδικα αυτού του αρχείου έχει να κάνει με την αποθήκευση των δεδομένων (username και password) στην συσκευή . Φαίνεται παρακάτω :

```
private SharedPreferences loginPreferences;
private SharedPreferences.Editor loginPrefsEditor;
public static final String PREFERENCE_FILENAME = "LoginInfo";
[…]

loginPreferences = getSharedPreferences(PREFERENCE_FILENAME, MODE_PRIVATE);
        loginPrefsEditor = loginPreferences.edit();

        saveLogin = loginPreferences.getBoolean("saveLogin", false);
        if (saveLogin == true) {
            txtUsername.setText(loginPreferences.getString("username", ""));
            txtPassword.setText(loginPreferences.getString("password", ""));
            saveLoginCheckBox.setChecked(true);
        }
    }
```

Η κλάση sharedPreferences είναι ένας μηχανισμός αποθήκευσης δεδομένων στην συσκευή του κινητού. Αναλυτικά στον τρόπο λειτουργίας του αναφερθήκαμε στο εισαγωγικό κεφάλαιο.

Αρχικά ορίζουμε το αρχείο sharedPreferences , έναν Editor ώστε να μπορούμε να το επεξεργαστούμε , τέλος ορίζουμε και το όνομα του αρχείου στο οποίο θα αποθηκεύσουμε τα στοιχεία.

Πρώτα ανοίγουμε το αρχείο και βάζουμε τον Editor να "κοιτά " στο συγκεκριμένο αρχείο. Έπειτα κάνουμε έλεγχο για το αν υπάρχει ήδη το αρχείο και περιέχει δεδομένα. Αν όντως υπάρχει γεμίζουμε τα edit box του username και του password με αυτά τα δεδομένα και ενεργοποιούμε το checkbox για το αν θέλει ο χρήστης να αποθηκεύσει τα στοιχεία. (checked). Όλο το παραπάνω τρέχει κάθε φορά που μπαίνει στην οθόνη login.

Πάμε να δούμε πως γράφονται τα δεδομένα σε αυτό το αρχείο (sharedpreferences) .

```
public void saveLoginData(boolean loginComplete){

       if(loginComplete == true) {

              if (saveLoginCheckBox.isChecked()) {
                     loginPrefsEditor.putBoolean("saveLogin", true);
                     loginPrefsEditor.putString("username", username);
                     loginPrefsEditor.putString("password", password);
                     loginPrefsEditor.commit();
              }
              else{
                     loginPrefsEditor.clear();
                     loginPrefsEditor.commit();
              }

              finish();
              Intent i = new Intent(getApplicationContext(),CustomerActivity.class);
                     startActivity(i);
             }
        else { //(loginComplete == false)
              new showAlertMessage(this ,"Σφάλμα εισόδου!!!","Δώσατε λάθος στοιχεία ή
είστε μπλοκαρισμένος!");
         }


    }
```

Η παραπάνω συνάρτηση τρέχει κάθε φορά που ο χρήστης πατάει το κουμπί *Login*.

Το όρισμα της συνάρτησης είναι το αποτέλεσμα από την ταυτοποίηση του χρήστη που γίνεται από το server . Αν έχει δώσει σωστά στοιχεία η μεταβλητή loginComplete θα έχει τιμή true , αλλιώς false .

Στην περίπτωση που ο χρήστης έδωσε σωστά στοιχεία και έχει επιλέξει το checkbox *Remember me* , χρησιμοποιώντας τον Editor που εξηγήσαμε παραπάνω για το πώς συνδέεται με το αρχείο sharedpreferences ,γράφουμε σε αυτό τα δεδομένα που έδωσε ο χρήστης.

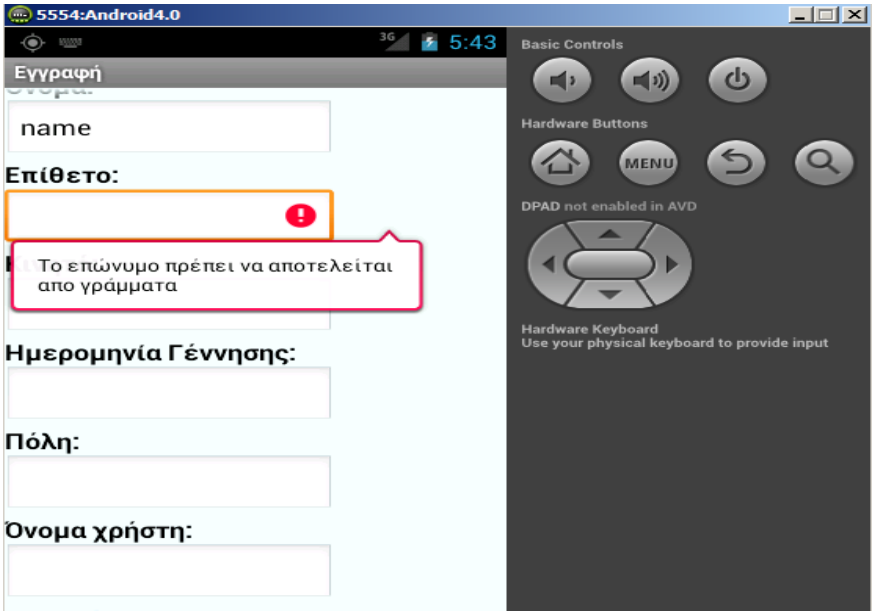
Σε περίπτωση που ο χρήστης δεν έχει επιλεγμένο το *Remember me* , αν το αρχείο είχε δεδομένα το καθαρίζουμε.

Αν πάλι ο χρήστης δεν έχει δώσει σωστά στοιχεία ή είναι μπλοκαρισμένος του εμφανίζεται μήνυμα και δεν τρέχει ότι έχει να κάνει με την εγγραφή στο αρχείο sharedpreferences.

Τέλος τερματίζουμε την τρέχουμε δραστηριότητα και μεταβαίνουμε στην κύρια οθόνη του πελάτη.

6.4 Οθόνη εγγραφής νέου χρήστη (Register screen)

Εδώ ο χρήστης πρέπει να δώσει κάποια προσωπικά στοιχεία ώστε να γίνει η εγγραφή του στην εφαρμογή.



Εικόνα 3: Οθόνη εγγραφή χρήστη

Αφού συμπληρώσει λοιπόν όλα τα απαραίτητα πεδία και πατήσει *Υποβολή* τα δεδομένα στέλνονται στον server ο οποίος ελέγχει αν υπάρχει εγγραφή με ίδιο username ή email. Εάν δεν υπάρχει τέτοια εγγραφή, γίνεται εγγραφή του χρήστη, αλλιώς όχι. Ο server αφού κάνει αυτές τις ενέργειες στέλνει κατάλληλο flag στην εφαρμογή και το διαχειριζόμαστε ανάλογα.

Σημείωση!

Αφού συμπληρώσει ο χρήστης όλα τα πεδία, πατώντας *Υποβολή* γίνεται έλεγχος αυτών σε επίπεδο εφαρμογής (όχι server) εάν έχουν σωστή μορφή, ώστε να αποφεύγεται μεγάλος φόρτος στον server. Σε περίπτωση που κάτι δεν είναι σωστά γραμμένο εμφανίζεται κατάλληλο μήνυμα στο σημείο που υπάρχει το λάθος (βλ εικόνα 3).

Αξίζει να αναφερθεί ο τρόπος με τον οποίο γίνετε ο έλεγχος διότι δεν περιέχει πολλά loops και διαδικασίες για τους κανόνες που πρέπει να υπάρχουν σε κάθε χαρακτήρα του αλφαριθμητικού. Για παράδειγμα ο κωδικός χρήστη να είναι τουλάχιστον 6 χαρακτήρες, το κινητό τηλέφωνο να περιέχει μόνο νούμερα κ.α .

Η λύση είναι η χρήση κανονικών εκφράσεων που μας δίνει η java. Το μόνο που έχουμε να κάνουμε είναι να ορίσουμε την κανονική έκφραση και να δούμε αν ταιριάζει με το αλφαριθμητικό που δόθηκε μέσω της μεθόδου String.matches (κανόνας_κανονικής_έκφρασης).

```
private static final String NAME_PATTERN      = "^[a-zA-Zα-ωΑ-Ω]+$";
private static final String CELLPHONE_PATTERN = "^[0-9]+$";
private static final String USERNAME_PATTERN  = "^[a-zA-Z][[a-zA-Z][0-9]]*$";
private static final String EMAIL_PATTERN     = "^[a-z0-9_\\+-]+(\\.[a-z0-9_\\+-]+)*@[a-z0-9-
]+(\\.[a-z0-9-]+)*\\.([a-z]{2,4})$";
```

Τις κανονικές εκφράσεις (regular expressions) τις ορίζουμε όπως παρακάτω :

Για το NAME_PATTERN , το όνομα πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείτε μόνο από γράμματα είτε Ελληνικά είτε Λατινικά.
Κανονική έκφραση: ^[a-zA-Zα-ωΑ-Ω]+$

Για το CELLPHONE_PATTERN , το κινητό πρέπει να είναι τουλάχιστον από 10 αριθμούς. Μόνο αριθμούς.
Κανονική έκφραση: ^[0-9]+$

Για το USERNAME_PATTERN , το username πρέπει να είναι πάνω από 2 χαρακτήρες (μόνο λατινικοί όλοι) και να αρχίζει με γράμμα ή κάτω παύλα.
Κανονική έκφραση : "^[a-zA-Z][[a-zA-Z][0-9]]*$";

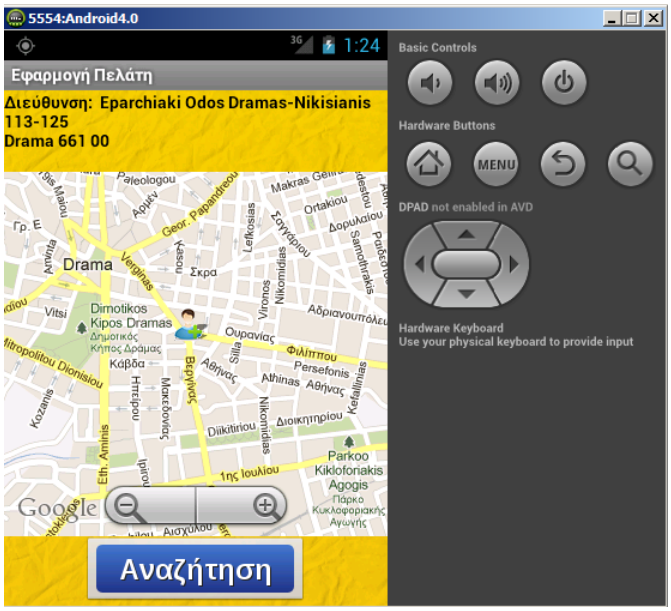
Για το MAIL_PATTERN , το email πρέπει να είναι τουλάχιστον 4 χαρακτήρες, να αρχίζει με γράμμα, αριθμό,_,+ ή -, πρέπει να υπάρχει το @ και η . και μετά την . να έχουμε από 2 έως 4 χαρακτήρες.
Κανονική έκφραση : ^[a-z0-9_\\+-]+(\\.[a-z0-9_\\+-]+)*@[a-z0-9-]+(\\.[a-z0-9-]+)*\\.([a-z]{2,4})$

Παράδειγμα χρήσης :

```
[…]

String NAME_PATTERN      = "^[a-zA-Zα-ωΑ-Ω]+$";
String name = “this_is_a_name” ;

If ( name.matches (NAME_PATTERN ) ) {
       //ταιριάζει με την κανονική έκφραση
}
else {
       //δεν ταιριάζει
}
```

6.5 Κύρια οθόνη εφαρμογής (main screen)

Από εδώ ο χρήστης μπορεί να εκτελέσει την κύρια λειτουργία της εφαρμογής, να καλέσει ταξί! Για να έχει πρόσβαση σε αυτή την οθόνη πρέπει να κάνει επιτυχώς login.



Εικόνα 4 :Βασική οθόνη αναζήτησης ταξί

Αρχικά γίνεται έλεγχος εάν ο χρήστης έχει πρόσβαση στο internet και ανοικτή επικοινωνία με τον server της εφαρμογής. Αν ναι, συνδέεται μέσω της υπηρεσίας mqtt push με τον mosquito broker (διαμεσολαβητή) ώστε να μπορεί να στέλνει και να δέχεται ειδοποιήσεις και να συνδέεται με άλλες συσκευές. Τον αναλυτικό τρόπο λειτουργίας του πρωτοκόλλου επικοινωνίας του mqtt και τον τρόπο με τον οποίο το προσαρμόσαμε στην εφαρμογή για να το χρησιμοποιήσουμε , το έχουμε αναλύσει στο κεφάλαιο 3.

Στη συνέχεια ανοίγουμε το GPS για να λαμβάνονται συντεταγμένες ώστε να μπορέσουμε να "ζωγραφίσουμε" στο χάρτη την τοποθεσία του χρήστη.

Η αποτύπωση του χάρτη και του αντικειμένου που απεικονίζεται ο χρήστης γίνεται με τη βοήθεια του αρχείου CustomerPositionOverlay.java που βρίσκεται στο πακέτο app.taxiAnytimeCustomer.Customer .

Η δουλεία που επιτελεί το αρχείο αυτό είναι αρχικά να σχεδιάσει ένα αρχικό επίπεδο που είναι ο χάρτης και έπειτα κάθε νέο αντικείμενο που προσθέτεται στο χάρτη σχεδιάζεται πάνω στο υπάρχον επίπεδο . Έχει μεθόδους που διαχειρίζεται αυτά τα αντικείμενα όπως αφαίρεση και προσθήκη αντικειμένων στο χάρτη .

Στο παρακάτω παράδειγμα φαίνεται πως λαμβάνουμε τις συντεταγμένες :

```
public GeoPoint gp;
private LocationManager locationManager;
public  Users customer;

customer = UsersFactory.createUser("customer");
locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

customer.setLatitude(myLocation.getLatitude()*1E6) ;
customer.setLongitude(myLocation.getLongitude()*1E6);


Drawable marker=getResources().getDrawable(R.drawable.customer_icon2);
int markerWidth = marker.getIntrinsicWidth();
int markerHeight = marker.getIntrinsicHeight();
marker.setBounds(-markerWidth/2, -markerHeight,markerWidth/2, 0);
customerOverlay = new
CustomerPositionOverlay(marker,getApplicationContext(),myLocation,customer);
```

Πρώτα δηλώνουμε τα αντικείμενα των κλάσεων : γεω-κωδικοποίησης, τοποθεσίας και πελάτη , μετά τα εκχωρούμε τιμές και γεμίζουμε το αντικείμενο πελάτης με τα στοιχεία γεωγραφικό μήκος και πλάτος. Έπειτα ετοιμάζουμε το εικονίδιο (marker) του πελάτη που θα ζωγραφιστεί στο χάρτη, ορίζοντας μήκος-πλάτος και όρια .
Τέλος χρησιμοποιούμε την βοηθητική κλάση CustomerPositionOverlay για να αναλάβει την απεικόνιση των στοιχείων στο χάρτη.

Για παράδειγμα :

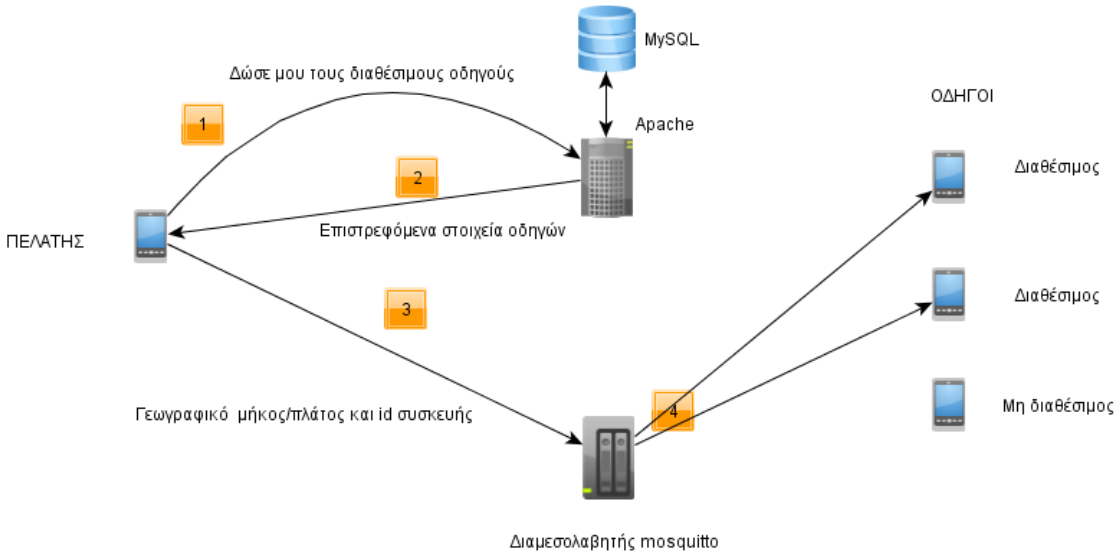
```
public void addItem(GeoPoint p, String title, String snippet)
   {
OverlayItem newItem = new OverlayItem(p, title, snippet);
       items.add(newItem);
       populate();
          }
```

Έτσι εισάγουμε νέο αντικείμενο στο χάρτη και με τη built in μέθοδο populate(), ζωγραφίζεται στο χάρτη.

6.5.1 Κουμπί αναζήτηση

Πατώντας το κουμπί *Αναζήτηση* ο χρήστης , στέλνει ειδοποίηση σε όλους τους διαθέσιμους οδηγούς ότι είναι έτοιμος για να γίνει η παραγγελία. Από τεχνικής πλευράς πρώτα τρέχει ένα script (βρίσκεται στον υποκατάλογο /scripts/getdriverinfo.php του server) το οποίο ελέγχει αν υπάρχουν διαθέσιμοι οδηγοί (μας επιστρέφονται τα στοιχεία των διαθέσιμων οδηγών για να τους στείλουμε ειδοποίηση αλλιώς κατάλληλο μήνυμα ότι δεν βρέθηκαν).
Εφόσον υπάρχουν οδηγοί διαθέσιμοι ,τα στοιχεία που στέλνονται για ειδοποίηση προς τους οδηγούς είναι η τρέχουσα τοποθεσία του πελάτη , δηλαδή το γεωγραφικό του μήκος και πλάτος και το μοναδικό αναγνωριστικό της συσκευής του πελάτη.
Τα δεδομένα στέλνονται μέσω http request στο server και το ανάλογο script που κάνει αυτή τη δουλεία είναι το notifyDrivers.php που βρίσκεται στον υποκατάλογο /mqttClient στα αρχεία που φιλοξενεί ο server.
Το πώς λειτουργούν τα αρχεία .php το αναλύουμε στο κεφάλαιο 5

Για να γίνει πιο κατανοητό στο παρακάτω σχήμα φαίνεται η όλη διαδικασία .



Εικόνα 5 :Διαδικασία ειδοποίησης

Μέρος του κώδικα που στέλνει τα δεδομένα φαίνεται παρακάτω :

```
[…]
myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/"+"
getdriverinfo.php","GET",parametersDrivers);

for(int i=0;i<myjobg.getJSONArray("drivers").length();i++)
{
parameters.add(new
BasicNameValuePair("deviceid"+i,myjobg.getJSONArray("drivers").getJSONObject(i)
.get("driverDeviceID").toString()));

}

       json2.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/mqt
tClient/notifyDrivers.php","POST",parameters);
[…]
```

Πρώτα λαμβάνουμε τις πληροφορίες των διαθέσιμων οδηγών ,εκτελώντας το αρχείο getdriverinfo.php. Τα αποτελέσματα αυτά εκχωρούνται στο αντικείμενο myjobg.

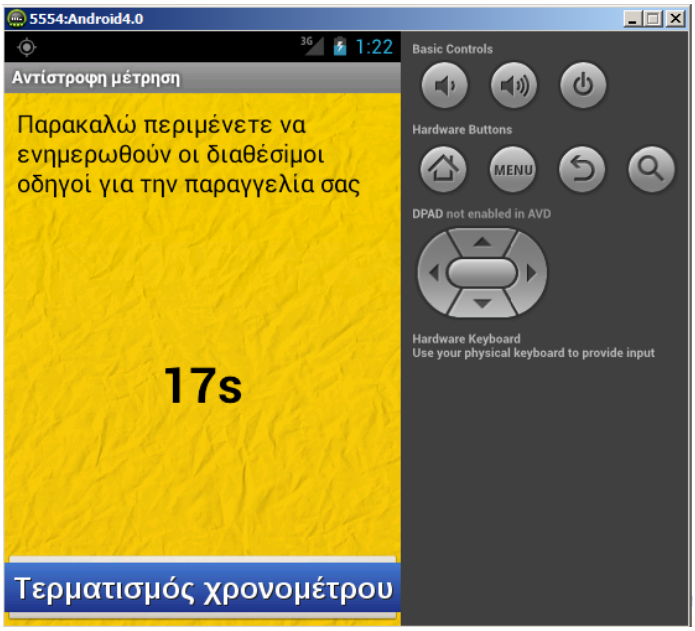
Μετά ετοιμάζουμε τις παραμέτρους που θα περάσουμε στο notifyDrivers.php για να ειδοποιήσουμε τους οδηγούς. Οι παράμετροι αυτοί είναι τα id των συσκευών των οδηγών και τις εξάγουμε από το αντικείμενο myjobj το οποίο έχει ήδη τα στοιχεία που θέλουμε.

Το notifyDrivers.php ειδοποιεί μέσω του διαμεσολαβητή (mosquitto) τις ανάλογες συσκευές των οδηγών.

Για περισσότερες πληροφορίες σχετικά με τη λειτουργία των php scripts βλ. Κεφάλαιο 5.

6.6 Οθόνη αναμονής απαντήσεων από οδηγούς (wait screen)

Αφού λοιπόν ο χρήστης στείλει αίτημα εύρεσης ταξί θα πρέπει να δώσουμε ένα εύλογο χρονικό διάστημα στους οδηγούς ταξί ώστε να δουν το αίτημα του πελάτη και να δηλώσουν ενδιαφέρον εάν το θέλουν. Ο χρόνος που δώσαμε είναι 1 λεπτό. Φυσικά ανά πάσα στιγμή ο χρήστης έχει την δυνατότητα να σταματήσει το χρονόμετρο και να δει όσους οδηγούς δήλωσαν ενδιαφέρον για το αίτημα (αν υπάρχουν).



Εικόνα 5: Αναμονή χρονομέτρου εύρεσης

Όταν ξεκινήσει το χρονόμετρο οι οδηγοί έχουν ήδη ειδοποιηθεί με τον τρόπο που αναφέραμε παραπάνω . Εάν κάποιος οδηγός δηλώσει ενδιαφέρον δημιουργείτε η παραγγελία στο αρχικό της στάδιο (κατάσταση παραγγελίας: προς αποδοχή από πελάτη). Στον ίδιο πίνακα στην βάση δημιουργούνται νέες εγγραφές για κάθε έναν νέο οδηγό που δείχνει ενδιαφέρον για την παραγγελία, με το ίδιο flag (κατάσταση παραγγελίας: προς αποδοχή από πελάτη). Όλα αυτά εκτελούνται από το script του server checkforOrders.php του οποίου ο κώδικας έχει αναλυθεί στο (κεφάλαιο 5 στην παράγραφο με τον υποκατάλογο /scripts)

```
protected int waitTime = 60;
[…]
public void startCounting() {

        countDownTimer = new CountDownTimer(waitTime  * 1000, 1000) {
      @Override
      public void onTick(long millisUntilFinished) {
        lblTimeLeft.setText(String.valueOf(millisUntilFinished / 1000) + "s");
      }

      @Override
      public void onFinish() {
         try {
if ( new
ConnectionDetectorTask(CountdownActivity.this).execute().get().get("connectionState")
){
                              new checkOrdersTask().execute();
                        }
             } catch (Exception e) {
                    new showAlertMessage(CountdownActivity.this ,"Ωχ! Κάτι πήγε
στραβά!!!","Δοκιμάστε αργότερα");


                    e.printStackTrace();
             }
      }

   }.start();

 }
```

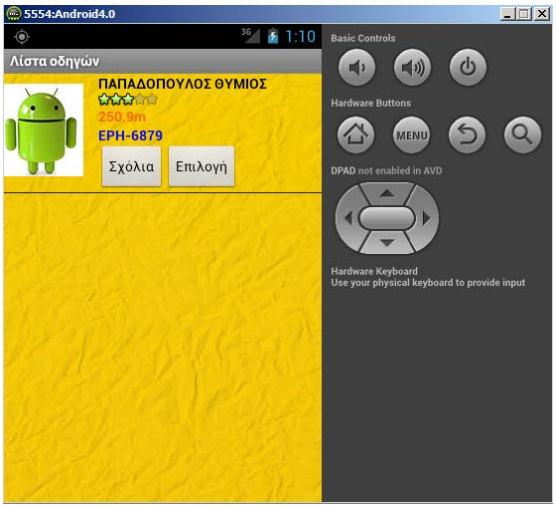
Παραπάνω παραθέτεται ο τρόπος με τον οποίο χρησιμοποιούμε τον CountDownTimer. Αρχικά ορίζουμε έναν CountDownTimer και του δίνουμε τον χρόνο που θα μετρήσει. Είναι 60sec.

Με το που τελειώσει το χρονόμετρο ή ο χρήστης πατήσει *Τερματισμό χρονομέτρου,* αν ο server μας επιστρέψει flag ότι δεν υπάρχουν απαντήσεις στην παραγγελία, εμφανίζεται ένα pop up dialog που ενημερώνει τον χρήστη και τον ρωτάει αν θέλει να ξανά στείλει αίτημα στους οδηγούς ή να επιστρέψει στην κύρια οθόνη της εφαρμογής.

Εάν υπάρχουν απαντήσεις στο αίτημα, μεταβαίνουμε σε νέο activity όπου εμφανίζονται οι πληροφορίες των οδηγών που ενδιαφέρθηκαν για την παραγγελία. Για αυτό το activity θα μιλήσουμε παρακάτω.

6.7 Οθόνη λίστας οδηγών(drivers list screen)

Μετά από την αναζήτηση ο πελάτης περιμένει να δει ποιοι οδηγοί τον έχουν επιλέξει για την παραγγελία. Σε αυτή την οθόνη φαίνεται μια λίστα με τα στοιχεία των οδηγών που έδειξαν ενδιαφέρον. Εμείς αποφασίσαμε να κρατάμε στοιχεία όπως το όνομά του, το συνολικό του rating και τον αριθμό των πινακίδων του, σαφέστατα μπορούσαμε να κρατάμε και άλλα.



Εικόνα 6 :Οθόνη λίστας με οδηγούς

Ο χρήστης έχει δυο επιλογές στο σημείο αυτό είτε να δει σχόλια (οθόνη εμφάνισης σχολίων) που κάνανε άλλοι χρήστες για τον συγκεκριμένο οδηγό είτε να τον επιλέξουμε (οθόνη έναρξης-τερματισμού παραγγελίας) να γίνει η παραγγελία. Και οι δυο επιλογές οδηγούν σε διαφορετικές οθόνες τις οποίες θα εξηγήσουμε στις επόμενες ενότητες.

Τώρα ας αναλύσουμε πως λειτουργεί και πως γεμίζει η λίστα με τους οδηγούς.

Στον κώδικά μας τα ανάλογα αρχεία βρίσκονται στο πακέτο app.taxiAnytimeCustomer.DriversList και είναι το DriverDetails.java και ListViewActivity.java.

Στο DriverDetails.java υλοποιούμε την κλάση που έχει ως δεδομένα τα στοιχεία του οδηγού (όνομα, πινακίδες κτλ). Επίσης εδώ υλοποιούμε μια μέθοδο για να εμφανίζει μόνο ένα δεκαδικό για την απόσταση και αξίζει να τη σχολιάσουμε λίγο.

```
private String distance2dec(String distance)
	{
		String []split = distance.split("\\.");
		String returned;
		try{
			returned =
split[0]+"."+split[1].charAt(0)+"m";
			 }
		catch(Exception e) {
			e.printStackTrace();
			returned = distance+"m";
		}
		return returned;
```

Δέχεται την απόσταση σε μορφή πχ “340.567.” Διαχωρίζουμε το string με βάση την τελεία “.” και θα έχουμε ένα πίνακα δύο θέσεων με τα επιμέρους στοιχεία split[0] = “340” , split[1] = “567” . Τέλος επιστρέφουμε το πρώτο στοιχείο του πίνακα το “340” δηλαδή ,ενωμένο με τον πρώτο χαρακτήρα του δεύτερου στοιχείου του πίνακα δηλαδή από το “456” θα πάρουμε το “4”. Συνολικά θα έχουμε από το 340.567 που μας δόθηκε , το 340.5 .

Το ListViewActivity.java είναι το αρχείο που εμφανίζει τη λίστα με τους οδηγούς. Συνεπώς το πρώτο πράγμα που πρέπει να γίνει είναι να λάβουμε τα στοιχεία των οδηγών που . Αυτό γίνεται με το ανάλογο php script driverslistview.php που βρίσκεται στον υποκατάλογο /scripts του βασικού καταλόγου του server. Μέσα από την εφαρμογή android στέλνεται ένα αίτημα(request) στο driverslistview.php με όρισμα το id της συσκευής του πελάτη. Βάσει του id της συσκευής του πελάτη γίνεται η ταυτοποίηση με τους συγκεκριμένους οδηγούς. Με λίγα λόγια ελέγχεται ο πίνακας orders την βάσης για το ποιες παραγγελίες έχουν orderState =1 (δηλαδή βρίσκονται στο αρχικό στάδιο), και το ποιες εγγραφές του πίνακα αντιστοιχούν στο id της συσκευής του πελάτη. Έτσι θα μας εμφανιστούν όλοι οι οδηγοί που επέλεξε ο πελάτης και όχι όλοι οι οδηγοί γενικά.

```
[...]
ArrayList<DriverDetails> items_details = null;

[…]
parameters.add(new BasicNameValuePair("customerDevID",customerID));
myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/drivers
Listview.php","POST",parameters);
try {
      for(int i=0;i<myjobg.getJSONArray("drivers").length();i++){
      DriverDetails item_details = new DriverDetails();
item_details.setOrderid( Integer.valueOf(
myjobg.getJSONArray("drivers").getJSONObject(i).get("orderid").toString() )  );
item_details.setName(myjobg.getJSONArray("drivers").getJSONObject(i).get("name").to
String()   );
item_details.setDistance(myjobg.getJSONArray("drivers").getJSONObject(i).get("dista
nce").toString());

item_details.setRate(Float.valueOf(myjobg.getJSONArray("drivers").getJSONObject(i).
get("rate").toString())) ;

item_details.setPlateNumber(myjobg.getJSONArray("drivers").getJSONObject(i).get("ta
xiPlateNumber").toString());
item_details.setDriverDeviceId(myjobg.getJSONArray("drivers").getJSONObject(i).get(
"driverDeviceID").toString());

 […]

            return results;
```

Όπως φαίνεται στον κώδικα παραπάνω έχουμε ένα collection (arraylist) τύπου DriversDetails στο οποίο θα καταχωρήσουμε όλα τα στοιχεία των οδηγών που βρέθηκαν.

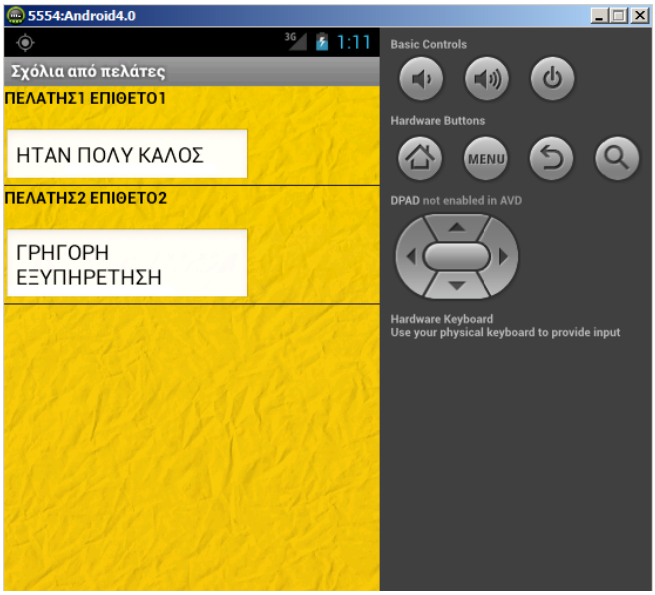
Στο αντικείμενο myjobg κρατάμε το αποτέλεσμα που επέστρεψε το php script που είναι όλα τα στοιχεία των οδηγών σε μορφή json. Έτσι σαρώνουμε όλο αυτό το αντικείμενο και εκχωρούμε κάθε τιμή μου μας ενδιαφέρει στο δικό μας arraylist. Με αυτή τη διαδικασία καταφέραμε να πάρουμε τους οδηγούς μέσα στην εφαρμογή μας και το μόνο που μένει είναι να τους εμφανίσουμε στο ListView.

```
[..]
myListView.setAdapter(new ItemListBaseAdapter(this, items_details));

public class ItemListBaseAdapter extends BaseAdapter {

        private  ArrayList<DriverDetails> driverDetailsArrayList;
        private LayoutInflater l_Inflater;

        public ItemListBaseAdapter(Context context, ArrayList<DriverDetails> results) {
                driverDetailsArrayList = results;
                l_Inflater = LayoutInflater.from(context);
        }
        […]
        public View getView(int position, View convertView, ViewGroup parent) {
                ViewHolder holder;
                if (convertView == null) {
                        convertView = l_Inflater.inflate(R.layout.item_details_drivers_view, null);
                        holder = new ViewHolder();
                        holder.txt_DriverName = (TextView) convertView.findViewById(R.id.name);
                        holder.txt_distance = (TextView) convertView.findViewById(R.id.distance);
                        holder.driversRate = (RatingBar) convertView.findViewById(R.id.ratingBar);
                        holder.plateNumber = (TextView) convertView.findViewById(R.id.plateNumber);
                        holder.itemImage = (ImageView) convertView.findViewById(R.id.photo);
                        convertView.setTag(holder);
                        […]
                } else {
                        holder = (ViewHolder) convertView.getTag();
                }
holder.txt_DriverName.setText(driverDetailsArrayList.get(position).getName();
holder.plateNumber.setText(driverDetailsArrayList.get(position).getPlateNumber());
holder.txt_distance.setText(String.valueOf(driverDetailsArrayList.get(position).getDistance()));
holder.driversRate.setRating(driverDetailsArrayList.get(position).getRate());
holder.itemImage.setImageBitmap(driverDetailsArrayList.get(position).getMyImage() );
        return convertView;
}
```

Η κλάση ItemListBaseAdapter δημιουργεί τις θέσεις των αντικειμένων που θα εμφανιστούν τα δεδομένα μας παράγει στην ουσία το ListView. Γεμίζει τα αντικείμενα με τις τιμές που έχουμε και μας επιστρέφει τις όψεις δηλαδή την οπτική αναπαράσταση του ListView.

6.8 Οθόνη εμφάνισης σχολίων (show comments list)

Επιτελεί τις ίδιες λειτουργίες που αναλύσαμε παραπάνω.Πάλι στον κώδικά μας τα ανάλογα αρχεία βρίσκονται στο πακέτο app.taxiAnytimeCustomer.DriversList και είναι το CommentsDetails.java και ListViewShowComments.java.



Εικόνα 7 :Οθόνη προβολής σχολίων

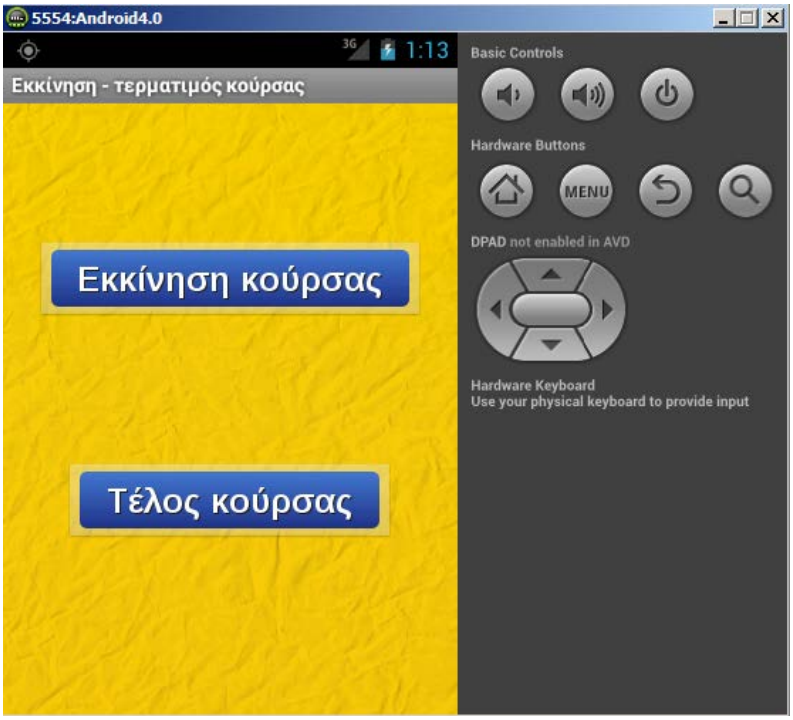
Στο CommentsDetails.java υλοποιούμε την κλάση που έχει ως δεδομένα τα στοιχεία για τα σχόλια όπως το ίδιο το σχόλιο και το όνομα αυτού που το έκανε.

Το ListViewShowComments.java είναι το αρχείο που εμφανίζει τη λίστα με τα σχόλια που κάνανε άλλοι χρήστες για τον οδηγό. Συνεπώς το πρώτο πράγμα που πρέπει να γίνει και πάλι είναι να λάβουμε τα στοιχεία για τα σχόλια . Αυτό γίνεται με το ανάλογο php script showCustomerComments.php που βρίσκεται στον υποκατάλογο /scripts του βασικού καταλόγου του server. Μέσα από την εφαρμογή android στέλνεται ένα αίτημα(request) στο showCustomerComments.php με όρισμα το id της συσκευής του οδηγού ώστε να εμφανιστούν βάση αυτό όλα τα σχόλια από τους πελάτες.

Το πώς γεμίζει το ListView το αναφέραμε παραπάνω, πάλι ακολουθείται η ίδια διαδικασία άρα δεν κρίνεται σκόπιμο να το πούμε ξανά εδώ .

6.9 Οθόνη έναρξης & τερματισμού διαδρομής(start-end ride screen)

Αφού λοιπόν ο πελάτης επιλέξει έναν οδηγό μεταβαίνει σε αυτή την οθόνη όπου περιμένει μέχρι να έρθει ο οδηγός στο σημείο παραλαβής. Να πούμε πως μέχρι στιγμής τα δύο κουμπιά είναι απενεργοποιημένα.



Εικόνα 8 :Οθόνη έναρξης-τερματισμού κούρσας

Όταν φτάσει ο οδηγός στέλνεται ειδοποίηση στον πελάτη και ενεργοποιείται το κουμπί *Εκκίνηση κούρσας*. Όταν το πατήσει ο πελάτης σημαίνει ότι βρίσκεται ήδη μέσα στο ταξί και έχει ξεκινήσει η διαδρομή. Στο σημείο αυτό η παραγγελία μεταβαίνει στην τελική της κατάσταση (διεκπεραιωμένη με flag orderState = 3) .Με το πάτημα του *Εκκίνηση κούρσας* ενεργοποιείται και το άλλο κουμπί *τέλος κούρσας* και με αυτό μεταβαίνουμε στην επόμενη οθόνη που είναι η καταχώρηση σχολίων και βαθμολόγηση του οδηγού ταξί.

Πριν πάμε στην οθόνη της βαθμολογίας ας πούμε τι γίνεται από τεχνικής πλευράς όταν πατάμε το κουμπί *Εκκίνηση κούρσας* .

```
[…]
parameters.add(new BasicNameValuePair("customerid",this.deviceid));
parameters.add(new BasicNameValuePair("orderid",this.orderid));
parameters.add(new BasicNameValuePair("type","customer"));


myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scrip
ts/orderConfirmation.php","POST",parameters);
```

Στέλνουμε από την εφαρμογή Android ως requests το id αναγνωριστικό της συσκευής του πελάτη, το orderid της τρέχουσας παραγγελίας και τον τύπο του χρήστη “customer”. Αυτά τα requests δέχεται ως ορίσματα το αρχείο orderComfirmation.php ,που βρίσκεται στον υποκατάλογο /scripts του βασικού καταλόγου του server, και το μόνο που κάνει είναι να ενημερώσει (διαδικασία update) την κατάσταση της παραγγελίας σε orderState = 3 . Έτσι ολοκληρώνεται η διαδικασία της παραγγελίας. (για το php script βλ. κεφάλαιο 5)

Αξίζει να σημειωθεί πως τα δεδομένα εισέρχονται πρώτα μέσω του service (Push Service) που έχουμε από την πλευρά του Android.

Το πρώτο κομμάτι που τρέχει είναι το εξής :

```
public void publishArrived(String topicName, byte[] payload, int qos,
boolean retained) {
                    // Show a notification
                    String s = new String(payload);
                    showNotification(s);
                    log("Got message: " + s);
            }
```

Από αυτό το κομμάτι κώδικα το μόνο που μας νοιάζει είναι το payload, δηλαδή τα εισερχόμενα δεδομένα που έρχονται σε πρωταρχική μορφή (byte) από το mosquitto και μετατρέπονται σε string που είναι η ειδοποίηση μας. Αυτό το string το περνάμε ως όρισμα στη showNotifications(…) για να εμφανιστεί η ειδοποίηση. Η ειδοποίηση που εμφανίζουμε στην οθόνη αυτή είναι για το αν έχει φτάσει ο οδηγός στο σημείο που είμαστε.

```
private void showNotification(String text) {

            […]

          Intent intent = new Intent(this,CustomerStartEndRide.class);
          intent.setAction(Intent.ACTION_VIEW);
          intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
          intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
          Bundle b = new Bundle();
          b.putInt("driverHasArrived", 1);
          intent.putExtras(b);

      PendingIntent pi =
PendingIntent.getActivity(this.getApplicationContext(), 0,
             intent, 0);
            n.setLatestEventInfo(this.getApplicationContext(), NOTIF_TITLE,
text, pi);
            mNotifMan.notify(NOTIF_CONNECTED, n);
            startActivity(intent);
```

Εμφανίζουμε την ειδοποίηση στην ίδια οθόνη (δηλαδή στην οθόνη *Εκκίνηση/τερματισμός κούρσας*) στέλνοντας ως flag ειδοποίησης driverHasArrived = 1 . Το flag το στέλνουμε στην activity *Εκκίνηση/τερματισμός κούρσας* μέσω αντικειμένου Bundle. Αυτό είναι ένας μηχανισμός για να περνάμε δεδομένα μεταξύ activities και services.

Παρακάτω φαίνεται πως το χρησιμοποιούμε.

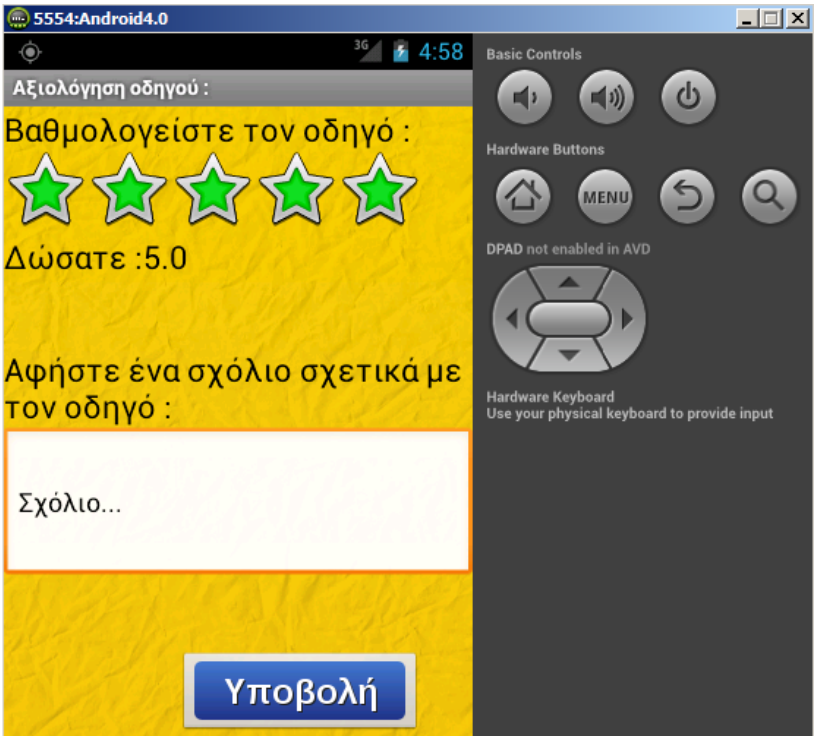
```
        extras = getIntent().getExtras();
                        if(extras != null ){
              if ( extras.getInt("driverHasArrived") == 1 ){
                                    btnStart.setEnabled(true);
                              }
                        }
```

Για να λάβουμε το bundle που στείλαμε από το service χρησιμοποιούμε το getIntent().getExtras(). Αν υπάρχει bundle, συνεπώς αν υπάρχει και κάποιο εισερχόμενο δεδομένο (δηλ . ειδοποίηση) θα είναι ορισμένο αλλιώς θα έχει τιμή null;

Στην περίπτωση που έχουμε εισερχόμενο δεδομένο (που σημαίνει ειδοποίηση ότι ο οδηγός έφτασε) τότε ενεργοποιούμε το κουμπί της εκκίνησης.

6.10 Οθόνη βαθμολόγησης και σχολιασμού οδηγού (rating & comments screen)

Στο σημείο αυτό έχει ολοκληρωθεί η διαδρομή και ο πελάτης δύναται να σχολιάσει καθώς και να βαθμολογήσει τον οδηγό.



Εικόνα 9 :Οθόνη βαθμολογίας

Στο πάνω μέρος της οθόνης επιλέγει πόσα αστεράκια θα βάλει (0-5) ανάλογα με τον βαθμό ικανοποίησης του, και κάτω μπορεί στο πλαίσιο κειμένου να γράψει κάποιες παρατηρήσεις. Πατώντας το πλήκτρο *Υποβολή* αποθηκεύονται αυτές οι επιλογές του στη βάση δεδομένων.

Για να δούμε τι γίνεται από την τεχνική πλευρά :

```
ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();

parameters.add(new BasicNameValuePair("rate",String.valueOf(
ratingBar.getRating())) );
parameters.add(new BasicNameValuePair("comment",String.valueOf(
txtComments.getText())) );
parameters.add(new BasicNameValuePair("driverDevId",String.valueOf(
orderPrefs.getString("selectedDriverDevId", "") ) ) );
parameters.add(new
BasicNameValuePair("orderid",String.valueOf(orderPrefs.getString("orderid",
"")  ) ) );
parameters.add(new
BasicNameValuePair("customerDevID",String.valueOf(fromCustomer.getString("cus
tomerdevid", "")  ) ) );


[…]

myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/m
akeCommentsRatings.php","POST",parameters);
```

Ετοιμάζουμε τα requests rate, comment ,driverDevId, customerDevID, orderid, τα οποία είναι αντίστοιχα η βαθμολογία, το σχόλιο, το μοναδικό αναγνωριστικό συσκευής οδηγού και πελάτη και το id της τρέχουσας παραγγελίας (πεδίο ordered πίνακα orders της βάσης).

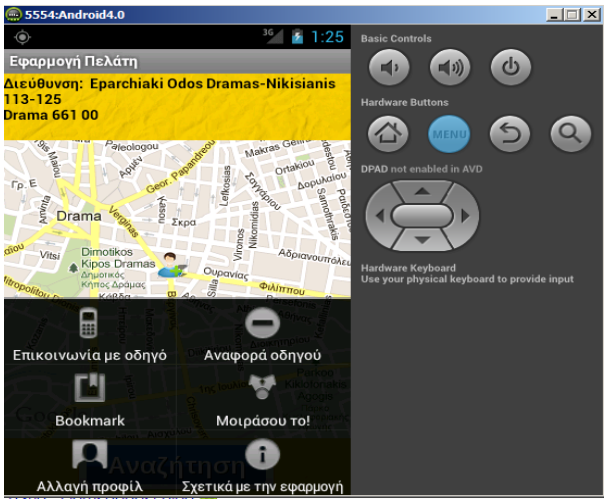
Έπειτα εκτελείται το script makeCommentsRating.php με την εντολή json.makeHttpRequest(…..). Γίνεται στην ουσία εισαγωγή (insert) στη βάση δεδομένων με μια νέα εγγραφή που περιέχει τα στοιχεία που δώσαμε.

6.11 Πλήκτρο μενού (Menu buttons)

Όλα τα κινητά android που πωλούνται στην Ευρώπη έχουν ένα πλήκτρο μενού (). Πατώντας το ο πελάτης έχει τις εξής δυνατότητες.

- Επικοινωνία με οδηγό
- Αναφορά οδηγού
- Bookmark
- Μοιράσου το!
- Αλλαγή προφίλ
- Σχετικά με την εφαρμογή

Να σημειωθεί ότι οι παραπάνω επιλογές είναι ορατές σε όλα τα σημεία της εφαρμογής αλλά δεν λειτουργούν μερικά συνέχεια. Για παράδειγμα εάν ο πελάτης επιλέξει ***Αναφορά οδηγού*** χωρίς να έχει γίνει κάποια κλήση για ταξί θα του εμφανιστεί κατάλληλο μήνυμα και δεν θα έχει σε ποιόν να κάνει αναφορά!
Το μενού εμφανίζεται όπως φαίνετε παρακάτω:



Εικόνα 10 :Βασικό menu

Παρακάτω φαίνεται ο τρόπος με τον οποίο δημιουργούνται τα πλήκτρα μενού.

```
public boolean onCreateOptionsMenu(Menu menu)
  {
      MenuInflater menuInflater = getMenuInflater();
      menuInflater.inflate(R.layout.menu_items, menu);
      return true;
              }
```

```
public boolean onOptionsItemSelected(MenuItem item)
    {
        switch (item.getItemId())
        {
        case R.id.menu_contact:
             // Κώδικας για επικοινωνία με οδηγό
             return true;
        case R.id.menu_report:
             // Κώδικας για αναφορά οδηγού
             return true;
        case R.id.menu_bookmark:
             // Κώδικας για να κάνει ο χρήστης bookmark το site της εφαρμογής
            return true;
        case R.id.menu_share:
             // Κώδικας για να ενημερώσει και άλλους για την εφαρμογή
            return true;
        case R.id.menu_profile:
             // Κώδικας για να αλλάξει το προφίλ του
            return true;
        case R.id.menu_aboutTheApp:
             // Κώδικας για εμφάνιση πληροφοριών σχετικά με την εφαρμογή.
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }
```

Με τον παραπάνω κώδικα καλούμε το xml αρχείο που περιέχει την οπτική αναπαράσταση των πλήκτρων μενού. Ο κώδικας καλείτε στην αρχή κάθε δραστηριότητας ώστε τα πλήκτρα μενού να είναι διαθέσιμα.

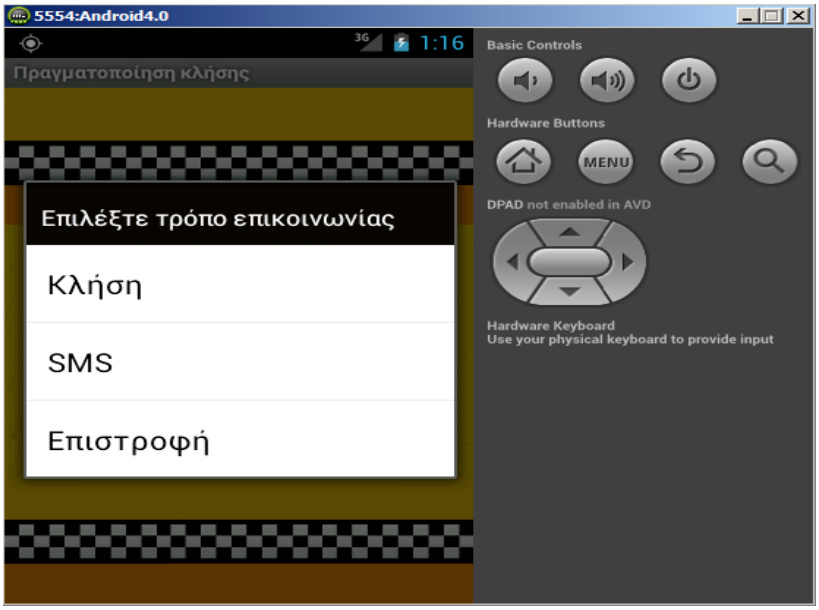
Η παραπάνω συνάρτηση είναι build in συνάρτηση του android. Κάθε επιλογή από το μενού έχει και ένα μοναδικό id. Όταν ο χρήστης πατήσει οπουδήποτε στο μενού, εάν εκεί που θα πατήσει υπάρχει μια επιλογή, τρέχει το case με το συγκεκριμένο id. Αν δεν πατήσει μια επιλογή αλλά γενικά στο μενού θα επιστρέψει όλο το μενού όπως είναι ώστε να μην χαθεί από την οθόνη που κοιτάει ο χρήστης.

Παρακάτω θα αναλύσουμε μία - μία τις επιλογές που έχει ο πελάτης.

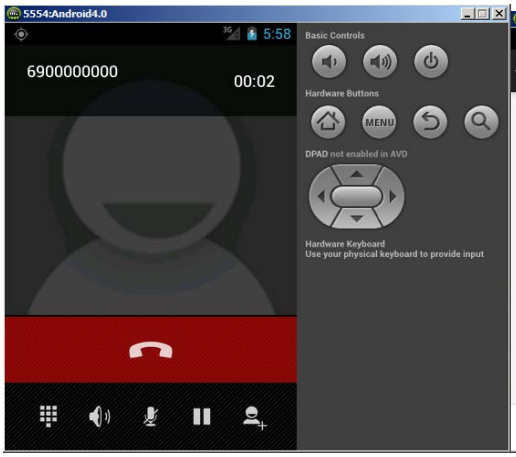
6.11.1 Επικοινωνία με οδηγό

Η λειτουργία αυτή είναι διαθέσιμη από την στιγμή που κάποιος πελάτης έχει επιλέξει τον συγκεκριμένο οδηγό για την παραγγελία.

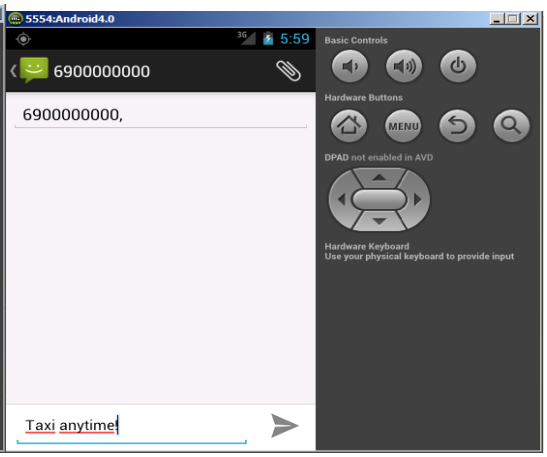
Για τον οποιοδήποτε λόγο ίσως ο πελάτης χρειαστεί να επικοινωνήσει με τον οδηγό για κάτι, έχει την δυνατότητα να το κάνει είτε μέσω μιας κλήσης είτε με το να του στείλει μήνυμα. Αυτό γίνετε όπως φαίνετε παρακάτω.



Εικόνα 11 :Επιλογές επικοινωνίας



Εικόνα 12 :Οθόνη κλήσης



Εικόνα 13 :Οθόνη αποστολής sms

Ευτυχώς για εμάς ο κώδικας με τον οποίο μπορεί να ανοίξει κλήση ή να γίνει αποστολή sms είναι διαθέσιμος.

```
public void call() throws InterruptedException, ExecutionException{
        Contact contact = new Contact(this);
        contact.makeContact();
             }
```

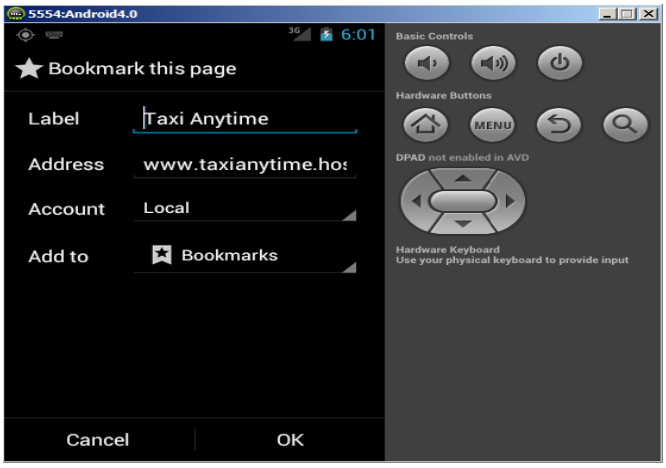
Εδώ καλούμε την κλάση που είναι υπεύθυνη σχετικά με την επικοινωνία. Παρακάτω φαίνεται πως την διαχειριζόμαστε. Παραθέτουμε την κύρια συνάρτηση της κλάσης Contact

```
public void makeContact() throws InterruptedException, ExecutionException {

        if ( new
ConnectionDetectorTask(actContext).execute().get().get("connectionState") ) {

                SharedPreferences pref = actContext.getSharedPreferences("myOrder",
Context.MODE_PRIVATE);
                setmNumber( new GetCellphoneTask ("customer",
                                        pref.getString("selectedDriverDevId",
"").toString() )
                                        .execute().get()
                                        );

                final CharSequence[] items = {"Κλήση", "SMS", "Επιστροφή"};
                AlertDialog.Builder builder = new AlertDialog.Builder(actContext);

                builder.setTitle("Επιλέξτε τρόπο επικοινωνίας");
                builder.setItems(items, new DialogInterface.OnClickListener() {

                        public void onClick(DialogInterface dialog, int item) {
                                if (items[item].equals("Κλήση")) {
                                        actContext.startActivity(new
Intent(Intent.ACTION_CALL, Uri.parse("tel:"+getmNumber())));

                                }
                                else if (items[item].equals("SMS")) {
                                        Intent smsIntent = new
Intent(Intent.ACTION_VIEW);
                                smsIntent.putExtra("sms_body", "Taxi anytime!");
                                smsIntent.putExtra("address", getmNumber());
                                smsIntent.setType("vnd.android-dir/mms-sms");

                            actContext.startActivity(smsIntent);
                                }
                                else
                                        Toast.makeText(actContext, items[item],
Toast.LENGTH_SHORT).show();
                         }

                        });

                        AlertDialog alert = builder.create();
                        alert.show();
```

Αρχικά ελέγχουμε αν υπάρχει σύνδεση με το δίκτυο και με τον server της εφαρμογής. Στην συνέχεια παίρνουμε το id του οδηγού με τον οποίο συνεργαζόμαστε. Αφού έχουμε το id του οδηγού, λαμβάνουμε τον αριθμό του κινητού του από την βάση. Μετά εμφανίζουμε στον χρήστη ένα alert dialog (pop up παράθυρο ουσιαστικά) και καλείτε να επιλέξει με ποιο τρόπο θέλει να επικοινωνήσει (κλήση ή sms). Αν επιλέξει κλήση ανοίγουμε το build in activity του android για να γίνει η κλήση και συμπληρώνεται αυτόματα ο αριθμός (του οδηγού) στον οποίο θα γίνει κλήση. Αν επιλέξει sms ανοίγουμε το build in activity του android για αποστολή sms και γεμίζουμε εμείς με κώδικα το περιεχόμενο του sms και τον αριθμό (του οδηγού) στον οποίο θα σταλεί το sms. Αν πατήσει *Επιστροφή* τον επιστρέφουμε το activity το οποίο είναι και του εμφανίζουμε ένα μήνυμα.

6.11.2 Bookmark

Εδώ μπορεί ο χρήστης να προσθέσει στους σελιδοδείκτες του και το site της εφαρμογής.
Η λειτουργία αυτή είναι διαθέσιμη σε όλα τα σημεία της εφαρμογής.



Εικόνα 14 :Εισαγωγή στα αγαπημένα

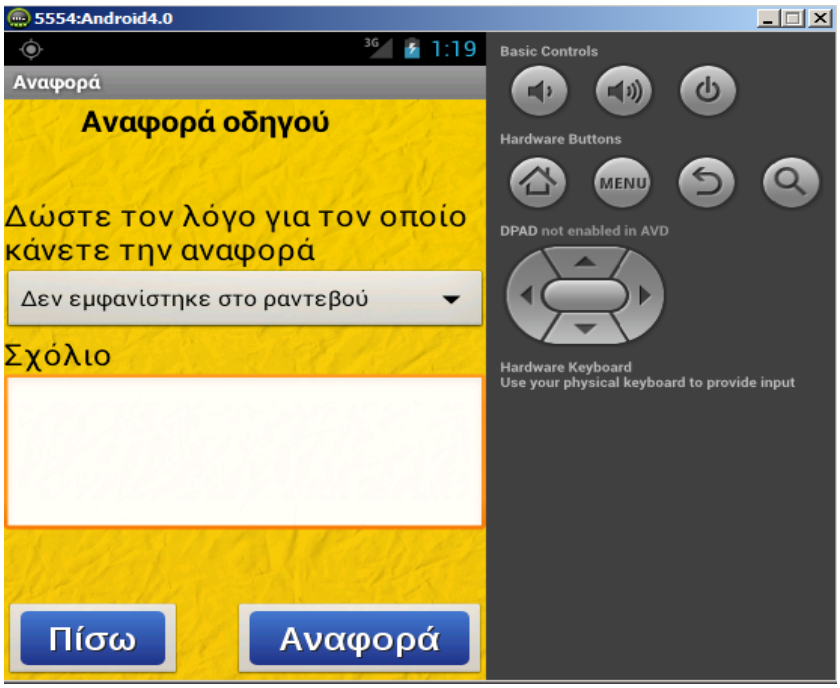
```
[…]
private static final String BOOKMARK_TITLE = "Taxi Anytime";
private static final String BOOKMARK_URL
="www.taxianytime.hostzi.com";

[…]

public static final void saveBookmark(Context c, String title,
String url) {
      Intent i = new Intent(Intent.ACTION_INSERT,
                  android.provider.Browser.BOOKMARKS_URI);
      i.putExtra("title", title);
      i.putExtra("url", url);
      c.startActivity(i);
 }
```

Ανοίγουμε το build in activity του android για προσθήκη σε σελιδοδείκτη, δίνοντας ως ορίσματα σαν τίτλο σελιδοδείκτη το *Taxi Anytime* και το link του site της εφαρμογής.

6.11.3 Αναφορά οδηγού

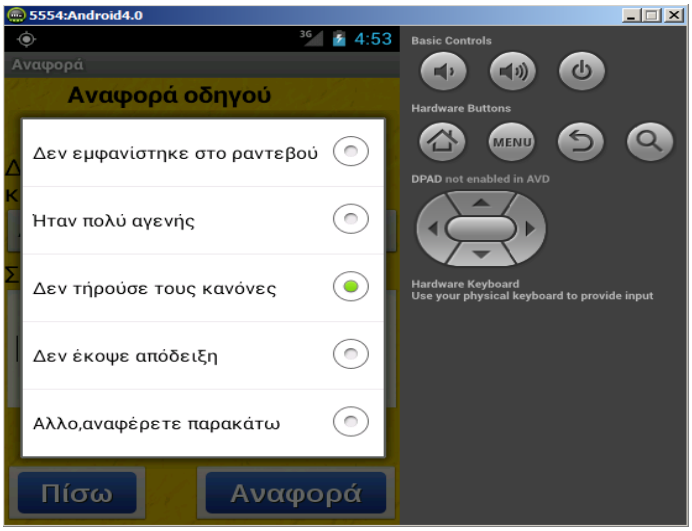
Όπως και πριν, η λειτουργία αυτή είναι διαθέσιμη από την στιγμή που κάποιος πελάτης έχει επιλέξει τον συγκεκριμένο οδηγό για την παραγγελία.

Εδώ εάν για κάποιο λόγο ο πελάτης κρίνει ότι ο οδηγός δεν φέρθηκε σωστά και δεν πρέπει να χρησιμοποιεί την εφαρμογή μπορεί να του κάνει αναφορά. Εάν ένας οδηγός δεχτεί πέντε αναφορές δεν θα μπορεί να ξανά χρησιμοποιήσει την εφαρμογή.



Εικόνα 15: Οθόνη αναφοράς οδηγού

Υπάρχουν κάποιοι σταθεροί λόγοι για τους οποίους ενδεχομένως να θέλει ο πελάτης να κάνει αναφορά στον οδηγό. Εάν ο λόγος δεν είναι στην λίστα μπορεί φυσικά να γράψει τον δικό του.



Εικόνα 16 :Επιλογή λόγων αναφοράς

```
public void report(){

        Intent intent = new
Intent(getApplicationContext(),ReportActivity.class);
        startActivity(intent);
}
```

Αρχικά φωνάζουμε την κλάση που είναι υπεύθυνη για την διαχείριση της αναφοράς προς τον οδηγό, όπως φαίνεται παρακάτω.

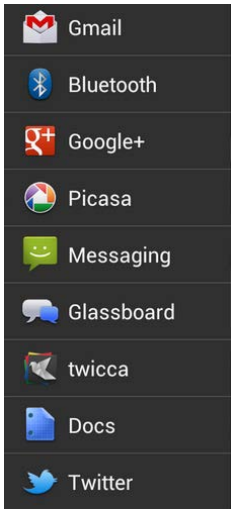
Το σημαντικότερο σημείο του κώδικα είναι το τι γίνεται όταν πατήσει ο χρήστης το κουμπί *Αναφορά* ,παραθέτουμε την αντίστοιχη συνάρτηση .

```
public void onClkReport(View v) throws InterruptedException, ExecutionException {

       if ( new
ConnectionDetectorTask(this).execute().get().get("connectionState") ) {

               if(getRptReasons().equals("Άλλο,αναφέρετε παρακάτω") &&
edtReason.getText().toString().equals(null)){

                     new showAlertMessage(this,"Προσοχή","Πρέπει να δώσετε τον
λόγο της αναφοράς!!!") ;

                  }
               else{

                     totalReasons = getRptReasons()+"
"+edtReason.getText().toString();
                     new AlertDialog.Builder( this )
                     .setTitle( "Επιβεβαίωση αναφοράς" )
                     .setMessage( "Έχετε αναφέρει το εξής : ' "+totalReasons +"'
είστε βέβαιος ότι θέλετε να στείλετε?")
                     .setOnCancelListener( new OnCancelListener() {

                            @Override
                            public void onCancel(DialogInterface dialog) {

                            }
                     })
                     .setPositiveButton( "Ναι", new
DialogInterface.OnClickListener() {
                                   public void onClick(DialogInterface dialog,
int which) {
                                          new CreateReportTask( totalReasons
).execute() ;

                                          }
                     })
                            .setNegativeButton( "Οχι", new
DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog, int
which) {

                                   }
                            } )
                     .show();

               imgBtnReport.setEnabled(false);
         }
      }
```

Αρχικά ελέγχουμε αν υπάρχει σύνδεση με το δίκτυο και με τον server της εφαρμογής. Μετά γίνεται έλεγχος για το αν έχει δώσει τον λόγο για τον οποίο κάνει την αναφορά, αν όχι , δεν μπορεί να προχωρήσει η διαδικασία. Αλλιώς, εμφανίζουμε με ένα alert dialog στο χρήστη τον λόγο που κάνει την αναφορά και του ζητάμε επιβεβαίωση για το αν θα γίνει αναφορά. Αφού ολοκληρωθεί η διαδικασία της αναφοράς και καταχωρηθεί στη βάση απενεργοποιούμε το κουμπί για να μη μπορεί να ξαναπατήσει ο χρήστης αναφορά στον οδηγό.

6.11.4 Μοιράσου το!

Εάν θέλει ο χρήστης να πει και σε άλλους για την εφαρμογή μπορεί να το κάνει από εδώ. Η συγκεκριμένη επιλογή δεν λειτουργεί σε όλα τα κινητά με τον ίδιο τρόπο! Εμφανίζετε μια λίστα με τους τρόπους που έχει ο χρήστης να πει για την εφαρμογή. Με sms, μέσω facebook εάν είναι εγκατεστημένο στο κινητό, μέσω twitter, google+, Bluetooth κλπ. Ανάλογα τι εφαρμογές έχει εγκατεστημένες στο κινητό ο χρήστης.



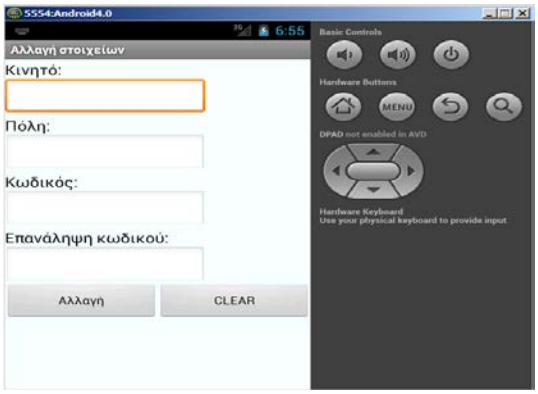
Εικόνα 17 : Share

Ανοίγουμε την build in activity του android , δίνοντας ως ορίσματα το θέμα, τον τίτλο που παραθύρου με τις επιλογές για τον τρόπο μαρασμού (send via)

```
public void share () {
      Intent share = new Intent(Intent.ACTION_SEND);
      String shareBody = "Taxi Anytime";
      share.setType("text/plain");
      share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Taxi Any time");
      share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
      startActivity(Intent.createChooser(share, "Send via"));

}
```

6.11. 5 Αλλαγή προφίλ

Εάν ο χρήστης έχει αλλάξει αριθμό κινητού, e-mail κ.α. Μπορεί να ενημερώσει το προφίλ του από εδώ. Αυτά που δεν μπορεί να αλλάξει είναι το όνομά του, το επίθετό του και το username του.



Εικόνα 18 :Οθόνη αλλαγής στοιχείων προφίλ

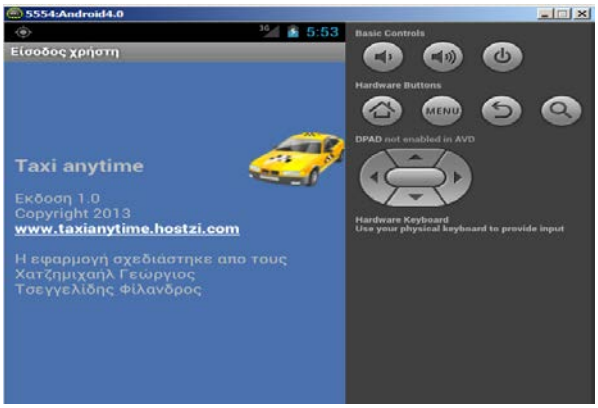
```
public void onClkEdit(View view) {

		if(isValidFormToEdit() == true &&
				AreEditBoxesEmpty(new EditText[]{txtCellphone,
			txtPassword,txtRepatPassword,txtTown}) == false ) {
				try{
if ( new ConnectionDetectorTask(this).execute().get().get("connectionState") ){
				new makeEditTask().execute();
				}
			}
			catch(Exception e){
				e.printStackTrace();
				new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε
αργότερα");
			}
		}
		else{
			new showAlertMessage(this ,"Προσοχή!!!","Ελέγξτε όλα τα πεδία");
		}

}
```

Αφού ανανεώσει το προφίλ του και πατήσει *υποβολή* , θα τρέξει ο παρακάτω κώδικας.

Αρχικά ελέγχουμε για το αν υπάρχει σύνδεση με τους servers και για το αν τα πεδία είναι συμπληρωμένα. Εφόσον το διασφαλίσουμε αυτό κάνουμε το κατάλληλο update στη βάση με τα νέα στοιχεία. Σε διαφορετική περίπτωση εμφανίζουμε κατάλληλα μηνύματα.

6.11.6 Σχετικά με την εφαρμογή

Εδώ μπορεί ο χρήστης να δει πληροφορίες σχετικά με την εφαρμογή και τους δημιουργούς καθώς και να περιηγηθεί στο αντίστοιχο site υποστήριξης .



Εικόνα 19 :Σχετικά με

```
public void aboutTheApp() {

            setContentView(R.layout.menu_about);
                   TextView aboutTextView = (TextView)findViewById(R.id.about);
                   aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                                 "Εκδοση 1.0<br> \n" +
                                 "Copyright 2013<br>  \n" +
                                 "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                                 "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                                 "Χατζημιχαήλ Γεώργιος <br> \n" +
                                 "Τσεγγελίδης Φίλανδρος <br>"));

                   aboutTextView.setLinkTextColor(Color.WHITE);
                   Linkify.addLinks(aboutTextView, Linkify.ALL);

      }
```

Αρχικά έχουμε σχεδιάσει ένα layout(menu_about.xml) με μπλε φόντο και μια εικόνα. Και πάνω σε αυτό το background έχουμε βάλει το κύριο περιεχόμενο (τις πληροφορίες) μέσα σε ένα textview σε μορφή html.

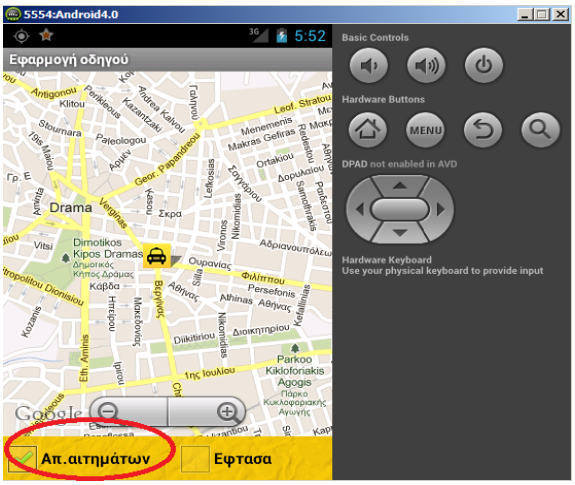
7 Τεκμηρίωση εφαρμογής οδηγού

7.1 Εισαγωγή

Στην ενότητα αυτή θα αναλύσουμε για τις οθόνες που συναντά ο οδηγός στην εφαρμογή και το πώς δουλεύουν από πίσω από πλευράς κώδικα. Πάλι θα μείνουμε στα βασικότερα σημεία που κρίνεται απαραίτητο να σχολιαστούν. Οι οθόνες login, register , splash , καθώς και τα menu λειτουργούν ακριβώς με τον ίδιο τρόπο που λειτουργούν και στον πελάτη συνεπώς δεν θα τα ξανά σχολιάσουμε. Περισσότερες πληροφορίες υπάρχουν στο προηγούμενο κεφάλαιο.

7.2 Κύρια οθόνη χάρτη (main screen map)

Αφού εισέρθει στην εφαρμογή ο χρήστης του εμφανίζεται η παρακάτω οθόνη



Εικόνα 1 : Βασική οθόνη

Όπως κάθε φορά που απαιτείται σύνδεση με το server της εφαρμογής και αποστολή/λήψη δεδομένων από το internet , έτσι και τώρα, γίνεται έλεγχος εάν ο χρήστης έχει πρόσβαση στο internet και ανοικτή επικοινωνία με τον server της εφαρμογής. Αν ναι, συνδέεται μέσω της υπηρεσίας mqtt push με τον mosquito broker (διαμεσολαβητή) ώστε να μπορεί να στέλνει και να δέχεται ειδοποιήσεις και να συνδέεται με άλλες συσκευές. Τον αναλυτικό τρόπο λειτουργίας του πρωτοκόλλου επικοινωνίας του mqtt και τον τρόπο με τον οποίο το προσαρμόσαμε στην εφαρμογή για να το χρησιμοποιήσουμε , το έχουμε αναλύσει στο κεφάλαιο 3.

Ο τρόπος που λαμβάνονται τα δεδομένα (γεωγραφικό μήκος και πλάτος) και δημιουργείται ο χάρτης με βάση αυτά είναι ο ίδιος με αυτόν της εφαρμογής του πελάτη, ο οποίος αναλύθηκε εκεί (Κεφάλαιο 6 , παράγραφος Κύρια οθόνη εφαρμογής).

Ο πλήρης κώδικας βρίσκεται στο πακέτο app.taxiAnytimeCustomer.Customer στο αρχείο CustomerActivity.java. Όπως βλέπουμε ο χρήστης έχει δύο επιλογές (checkboxes), *Απ. Αιτημάτων* και *Έφτασα* . Στην αρχή μόνο το *Απ. Αιτημάτων* είναι ενεργοποιημένο. Ενεργοποιώντας το checkbox *Απ. Αιτημάτων*, αλλάζουμε το flag isAvaliable από 0 σε 1. Αυτό πρακτικά κάνει τον οδηγό διαθέσιμο σε αιτήματα από πελάτες. Η αλλαγή γίνεται μέσα από το php script setdriverAvailability.php

7.2.1 Επιλογή Απάντηση αιτημάτων

Αυτή την στιγμή ο οδηγός είναι σε αναμονή για νέα αιτήματα. Όταν λοιπόν υπάρξει κάποιο ενδιαφέρον από πελάτη θα μπορεί να δεχθεί ειδοποίηση.

Ας δούμε σε ποιο σημείο δέχεται την ειδοποίηση :

```
[...]

        public Bundle extraParameters;
        private final int SEND_NOTIFICATION = 1;
        private final int SENDED_NOTIFICATION_ACK = 2;

        private final int DRAW_CUSTOMER = 1;
        private final int DRAW_DRIVER = -1;
        private final int DRAW_ACCEPTED_CUSTOMER = 2;
        private final int DRAW_AFTER_MAP_CLEAR = 3;

[...]

try {
                extraParameters = getIntent().getExtras();
                inComingNotifications( extraParameters );
        }
        catch (Exception e) {
                e.printStackTrace();
        }
```

Στο extraParameters εκχωρούνται τα δεδομένα που στείλαμε από το Push Service και ανάλογα τι τύπο (flag) έχουμε γίνεται και η κατάλληλη ενέργεια.

```
private void inComingNotifications(Bundle extras) {
     if(extras != null){
               SharedPreferences preferences = getSharedPreferences("fromPush",
MODE_PRIVATE);
               switch( Integer.valueOf( preferences.getString("type","") )) {

                 case SEND_NOTIFICATION: {
                              DriverAcceptOrders("0","/"+this.driver.getDeviceID());
                              Users customer = UsersFactory.createUser("customer");
                              customer.setDeviceID( preferences.getString("customerid", "") );
                              customer.setLatitude(
Double.valueOf(preferences.getString("customerLat", "") ) );
                              customer.setLongitude(
Double.valueOf(preferences.getString("customerLon", "") ) );

                              SharedPreferences SendPrefs =
this.getApplicationContext().getSharedPreferences(
                                        "acceptedCustomer",Context.MODE_PRIVATE);
                              SharedPreferences.Editor prefEditor = SendPrefs.edit();

prefEditor.putString("acceptedCustomerDevid",customer.getDeviceID());
                              prefEditor.commit();
                              drawLocation("customer",customer,DRAW_CUSTOMER);
                              break;
                 }
                 case SENDED_NOTIFICATION_ACK: {

                         DriverAcceptOrders("0","/"+this.driver.getDeviceID());
                         Users customer = UsersFactory.createUser("customer");
                         customer.setDeviceID( preferences.getString("customerAcceptedID",
"") );
                         customer.setLatitude(
Double.valueOf(preferences.getString("customerAcceptLat", "") ) );
                         customer.setLongitude(
Double.valueOf(preferences.getString("customerAcceptLon", "") ) );

                         drawLocation("customer",customer,DRAW_ACCEPTED_CUSTOMER);

                         ckeckReachCustomer.setEnabled(true);
                         ckeckReachCustomer.setChecked(false);
                         checkBoxAccept.setEnabled(false);
                         break;
                 }
            }
          }
          else{
            //push service start
            PushService.actionStart(DriverActivity.this);
            DriverAcceptOrders("0","/"+this.driver.getDeviceID());
            drawLocation("driver",driver,DRAW_DRIVER);
          }
 }
```

Την πρώτη φορά που τρέχει ο κώδικας δεν υπάρχουν εισερχόμενα δεδομένα στο incomingNotification(Bundle = null), άρα τρέχει το else . Εκεί ζωγραφίζεται για πρώτη φορά ο οδηγός και ξεκινάει το service ειδοποιήσεων (Push Service).

Όταν επιλέξει τον χρήστη που έκανε ειδοποίηση, έχουμε ως εισερχόμενο flag 1 (`SEND_NOTIFICATION = 1`) , τότε τρέχει το αντίστοιχο case. Στο σημείο αυτό έρχονται πληροφορίες όπως η τοποθεσία του πελάτη καθώς και το μοναδικό αναγνωριστικό της συσκευής του. Εμείς ανακτούμε αυτές οι πληροφορίες που είχαν αποθηκευτεί με τον μηχανισμό sharedPreferences μέσα από το service (θα αναφερθούμε μετά σε αυτό) .Αμέσως θέτουμε την κατάσταση του οδηγού σε μη ενεργή, ώστε να μην δεχθεί νέα αιτήματα από άλλους πελάτες. Τέλος σχεδιάζουμε αυτόν το πελάτη που επιλέξαμε για να εμφανιστεί στην οθόνη μας.

Στο σημείο αυτό περιμένουμε νέα ειδοποίηση από τον πελάτη για το αν θα μας επιλέξει από τη λίστα του. Όταν γίνει αυτό θα έχουμε ως εισερχόμενο flag 2 (`SENDED_NOTIFICATION_ACK = 2;`) , δηλαδή νέα ειδοποίηση που σημαίνει ότι ο πελάτης μας επέλεξε και ζωγραφίζεται στο χάρτη με νέο εικονίδιο (το παλιό αναιρείται) χρώματος πράσινο .

Για να δούμε πως εισέρχονται τα δεδομένα από το push service . Όπως έχει ξαναδιατυπωθεί από το push service μας ενδιαφέρει μόνο η μέθοδος ShowNotification (String text) , στην οποία εισέρχονται τα δεδομένα στο string text.

Στην μεταβλητή text είναι λοιπόν βρίσκονται όλα τα δεδομένα μας συνενωμένα με διαχωριστικό το χαρακτήρα " : " . Συνεπώς για να πάρουμε το κάθε στοιχείο χωριστά θα πρέπει να διαχωρίσουμε αυτό το αλφαριθμητικό σε επιμέρους πεδία. Ευτυχώς η java μας παρέχει τις αντίστοιχες μεθόδους της κλάσης String , συγκεκριμένα την String.split (...) . Για να δώσουμε ένα παράδειγμα για το τη γενική μορφή έχει το text.

text - > "type": "main notify message": "customer lat" : "customer lon" : "customer device id"

- type : Παίρνει τιμές (1,2) ανάλογα με το είδος της ειδοποίησης που θα κάνουμε . 1 για την αρχική ειδοποίηση και 2 για το ότι μας έχει αποδεχθεί ο πελάτης. Οι τιμές που παίρνει καθορίζονται από ποια php scripts εκτελούνται. (βλ. notifyDrivers.php από το κεφάλαιο 5)
- main notify message : Το κύριο μήνυμα που θα φαίνεται στην οθόνη της ειδοποίησης πχ "Νέος πελάτης!!!".
- customer lat : το γεωγραφικό μήκος του πελάτη
- customer lon : το γεωγραφικό πλάτος του πελάτη
- customer device id : το μοναδικό αναγνωριστικό της συσκευής του πελάτη

Παράδειγμα. 1 : "Νέος πελάτης" : "24.234544" : "41.232423" : "/3514323453465"

```
private void showNotification(String text) throws JSONException,
InterruptedException, ExecutionException {

String[] items = text.split("\\:")    ;
int type = Integer.parseInt(items[0]);
```

[…]

Μετά την διαχώριση κάθε πεδίο θα είναι και το ανάλογο στοιχείο του πίνακα items. Δηλαδή από το παράδειγμα items[0] =”1” , items[1] = “Νέος πελάτης” κτλ

```
private final int NEW_CUSTOMER = 1;
private final int ACCEPTED_CUSTOMER = 2;
```

```
switch(type){
        case NEW_CUSTOMER:{
            //Stelnoume to orderid sthn activity mas
            SharedPreferences SendPrefs =
this.getApplicationContext().getSharedPreferences(
                    "fromPush",Context.MODE_PRIVATE);
            SharedPreferences.Editor prefEditor = SendPrefs.edit();
            prefEditor.putString("customerLat",items[2]);
            prefEditor.putString("customerLon",items[3]);
            prefEditor.putString("customerid",items[4]);
            prefEditor.putString("type","1");
            prefEditor.commit();
            double distanceLimit;
            try{
                Location cloc = new Location("");
                cloc.setLatitude( Double.valueOf(items[2]) );
                cloc.setLongitude(  Double.valueOf(items[3]) );
                distanceLimit = DriverActivity.myLocation.distanceTo(cloc);
            }
            catch(Exception e){
                distanceLimit = DISTANCE_LIMIT*2.0f;
                //σε περίπτωση που υπάρξει σφάλμα ορίζουμε μια απόσταση
                //αυθαίρετα διπλάσια από την μέγιστη ,ώστε να είμαστε σίγουροι ότι
δεν θα σταλεί ειδοποίηση
                e.printStackTrace();
            }
```

Παραπάνω είναι ο κώδικας που τρέχει στο service και έχουμε λάβει ήδη τα δεδομένα που θέλουμε, καταχωρώντας τα σε sharedpreferences ώστε να μπορέσουμε να τα χρησιμοποιήσουμε και στην activity όπως είπαμε παραπάνω. Εδώ κάνουμε και ένα υπολογισμό για το πόσο απέχει ο οδηγός από τον πελάτη. Όπως θα δούμε παρακάτω υπάρχει ένα όριο (ορίσαμε εμείς 20 km) στο οποίο μετά από αυτό δεν θα δέχεται ο οδηγός ειδοποιήσεις από πελάτη. Αν είναι μέσα στο όριο θα σταλεί κανονικά ειδοποίηση.

```
[…]
public  static final double DISTANCE_LIMIT = 20000.0f;
[…]

if(distanceLimit <= DISTANCE_LIMIT && distanceLimit!= 0.0f ) {
            // Αν η απόσταση είναι μικρότερη από κάποια τιμή που ορίζουμε στο
conf.java τότε να στείλει ειδοποίηση
                Notification n = new Notification();
                n.flags |= Notification.FLAG_SHOW_LIGHTS;  // πχ    kati |= 2
είναι ίδιο με  kati = kati | 2
                n.flags |= Notification.FLAG_AUTO_CANCEL;
                n.defaults = Notification.DEFAULT_ALL;
                n.icon = R.drawable.star_big_on;
                n.when = System.currentTimeMillis();

                // Simply open the parent activity
                Intent intent = new Intent(PushService.this,
DriverActivity.class);
                intent.setAction(Intent.ACTION_VIEW);
               intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
               intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                Bundle b = new Bundle();
                b.putString("customer", items[1]);
                intent.putExtras(b);

                PendingIntent pi = PendingIntent.getActivity(this, 0,
                     intent, 0);
                // Change the name of the notification here
                n.setLatestEventInfo(this, NOTIF_TITLE,items[1], pi);
                mNotifMan.notify(NOTIF_CONNECTED, n);
                myContext.startActivity(intent);
            }
            break;
```

7.2.2 Επιλογή Έφτασα

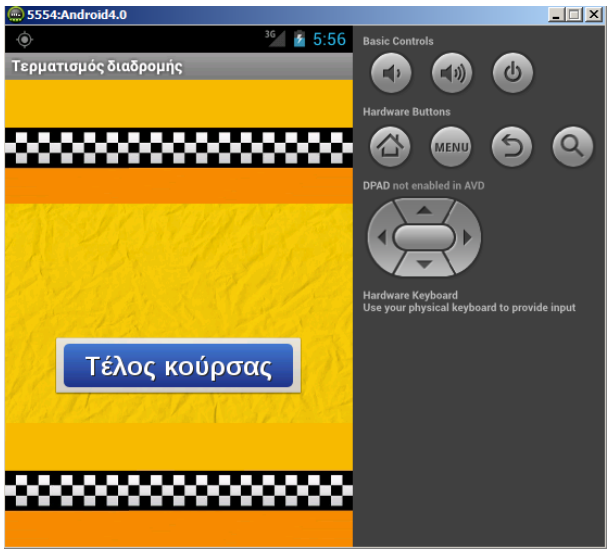
Η άλλη λειτουργία που μπορεί να κάνει ο οδηγός είναι να επιλέξει το checkbox *Έφτασα* . Φυσικά αυτή η επιλογή ενεργοποιείται μόνο όταν υπάρχει κάποιος πελάτης και έχει επιλέξει τον οδηγό. (όταν δηλαδή στο χάρτη φαίνεται ο πελάτης με πράσινο σχήμα). Εμφανίζεται ένα dialog box και ρωτά τον οδηγό αν όντως έφτασε στον πελάτη και είναι έτοιμος για παραλαβή. Στέλνει λοιπόν ειδοποίηση στον πελάτη πατώντας το *NAI* .

```
public void onReachCustomer(View view) throws InterruptedException, ExecutionException
{
        // Is the view now checked?
     if ( new ConnectionDetector(this).execute().get().get("connectionState") ){
         if (((CheckBox) view).isChecked()){
             AlertDialog.Builder builder =  new AlertDialog.Builder(this );
            builder.setTitle( "Ειδοποίηση άφιξης" )
            .setCancelable(false)
            .setMessage( "Βλέπετε τον πελάτη??" )
            .setOnCancelListener( new OnCancelListener() {
                @Override
                public void onCancel(DialogInterface dialog) {

                    finish();   //to finish Activity on which dialog is displayed
                }
            })
            .setPositiveButton( "NAI", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    //Ανακτούμε την παραγγελία(id) και την περνάμε ως όρισμα &
ειδοποιούμε τον πελάτη
                    SharedPreferences incOrder = getSharedPreferences("fromPush",
MODE_PRIVATE);
                    new DriverConfirmOrder("/"+driver.getDeviceID() ,
incOrder.getString("selectedOrderId","") ).execute();
                    clearMap();
                    Intent i = new
Intent(DriverActivity.this,DriverEndRideActivity.class);
                    startActivity(i);
                }
            })
            .setNegativeButton( "OXI", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    ckeckReachCustomer.setChecked(false);
                }
            } );
        AlertDialog alert = builder.create();
        alert.setOwnerActivity(DriverActivity.this);
         alert.show();
       }
   }
}
```

Στην συνέχεια γίνεται καθαρισμός στο χάρτη και μεταβαίνουμε στην επόμενη activity (οθόνη τερματισμού οδηγού) που αναφέρουμε παρακάτω.

7.3 Οθόνη τερματισμού παραγγελίας.

Κατά την διάρκεια της διαδρομής ο οδηγός δεν έχει δικαίωμα να δεχτεί άλλα αιτήματα από πελάτες. Θα πρέπει πρώτα να ολοκληρώσει την τωρινή παραγγελία πρώτα.
Για να το κάνει αυτό θα πρέπει να πατήσει απλά το κουμπί ***Τέλος κούρσας.*** Στην παρακάτω οθόνη βρίσκεται μετά την παραλαβή του πελάτη.



Εικόνα 2 : Τερματισμός παραγγελίας

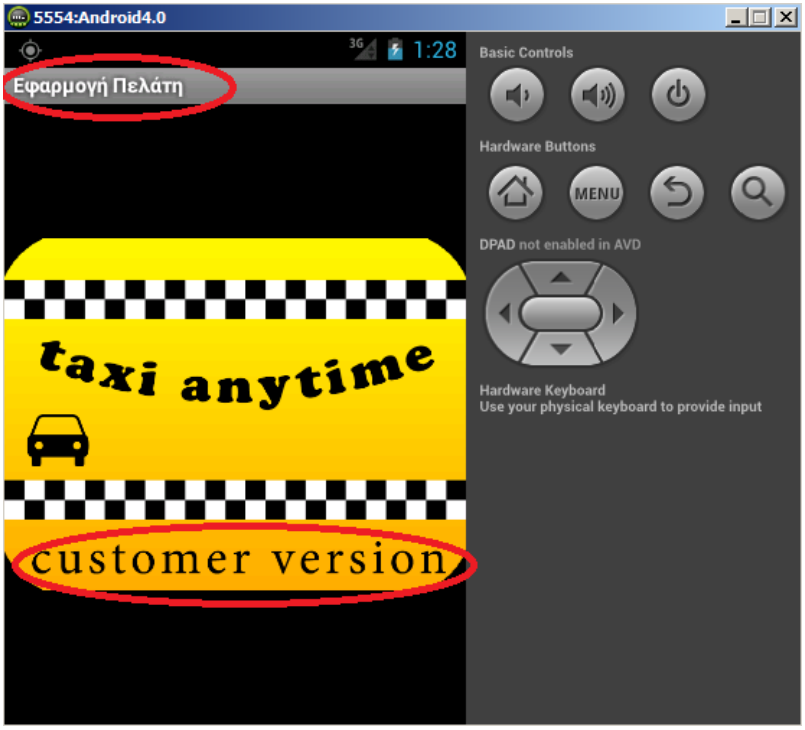
Όταν πατήσει το κουμπί, απλά τερματίζουμε την τρέχουσα activity και μεταφερόμαστε στην κύρια οθόνη της έκδοσης για οδηγούς της εφαρμογής όπως φαίνεται παρακάτω.

```
public void ImgBtnEndRide(View v)
	{
		finish();
		Intent i = new
Intent(getApplicationContext(),DriverActivity.class);

	i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
		startActivity(i);
	}
```

8 Οδηγίες χρήσης εφαρμογής πελάτη

8.1 Εισαγωγή

Με το που ανοίγουμε την εφαρμογή εμφανίζεται μια εισαγωγική οθόνη (splash screen) έως ότου γίνουν οι απαραίτητες διαδικασίες για να ξεκινήσει η εφαρμογή. Από την εισαγωγική οθόνη μπορούμε επίσης να δούμε εάν κατεβάσαμε την σωστή έκδοση της εφαρμογής.

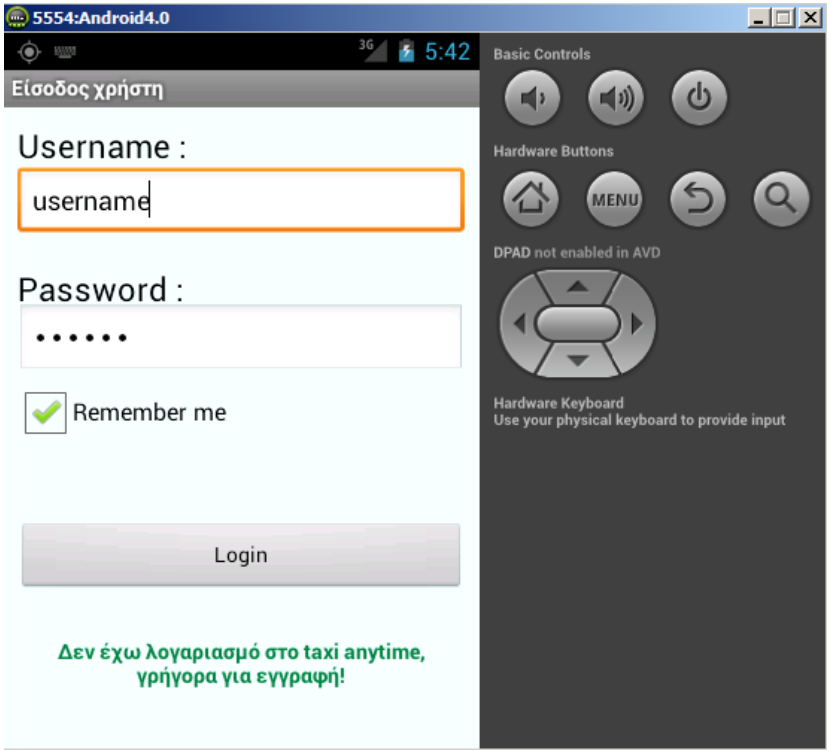


Εικόνα 1: Εισαγωγική οθόνη

Εάν γράφει customer version έχουμε κατεβάσει την εφαρμογή για πελάτες. Επίσης είναι ορατή η έκδοση της εφαρμογής και από την μπάρα πληροφοριών της εφαρμογής.

8.2 Είσοδος στην εφαρμογή

Στην συνέχεια πρέπει ο χρήστης να δώσει username & password για να κάνει είσοδο στην εφαρμογή.



Εικόνα 2 :Οθόνη εισόδου χρήστη

Εάν ο χρήστης έχει ήδη κάνει εγγραφή στην εφαρμογή μπορεί αφού δώσει τα σωστά δεδομένα (username & password) μπορεί να επιλέξει το ***Remember me*** ώστε κάθε επόμενη φορά να με το που πάει να κάνει είσοδο να είναι ήδη συμπληρωμένα αυτά τα πεδία. Εάν ο χρήστης δεν έχει λογαριασμό πατώντας στο κείμενο ***Δεν έχω λογαριασμό […]*** τον παραπέμπουμε σε μια φόρμα για να κάνει εγγραφή.

8.3 Εγγραφή στην εφαρμογή

Ο χρήστης πρέπει να δώσει τις παρακάτω πληροφορίες με τους συγκεκριμένους περιορισμούς:

- Το <u>όνομα</u> πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείται μόνο από γράμματα.
- Το <u>επώνυμο</u> πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείται μόνο από γράμματα.
- <u>Ημερομηνία γέννησης</u> .
- Το <u>κινητό</u> πρέπει να αποτελείται τουλάχιστον από 10 αριθμούς. Μόνο αριθμούς.

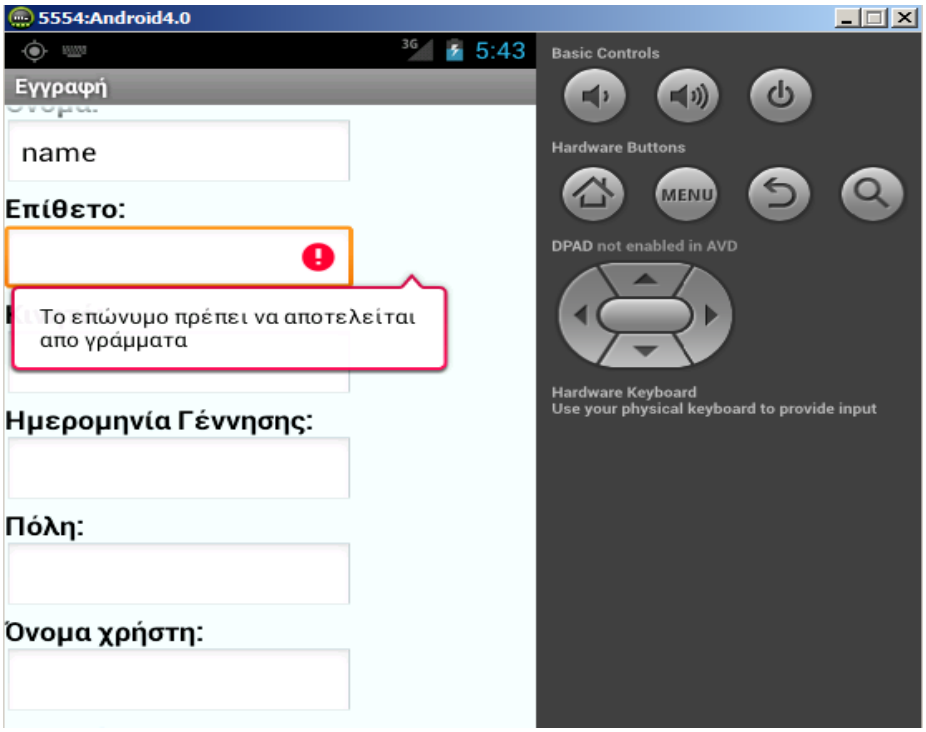
- Η πόλη πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείται μόνο από γράμματα.

- Το username πρέπει να είναι πάνω από 2 χαρακτήρες και να αρχίζει με γράμμα ή κάτω παύλα.

- Ο κωδικός πρέπει να είναι τουλάχιστον 6 ψηφία

- Το email πρέπει να είναι τουλάχιστον 4 χαρακτήρες, να αρχίζει με γράμμα, αριθμό, _, + ή -, πρέπει να υπάρχει το @ και . και μετά την . να έχουμε από 2 έως 4 χαρακτήρες.

Εάν ο χρήστης εισάγει κάτι λάθος δεν θα γίνει η εγγραφή και θα εμφανιστεί που είναι το λάθος όπως φαίνετε παρακάτω



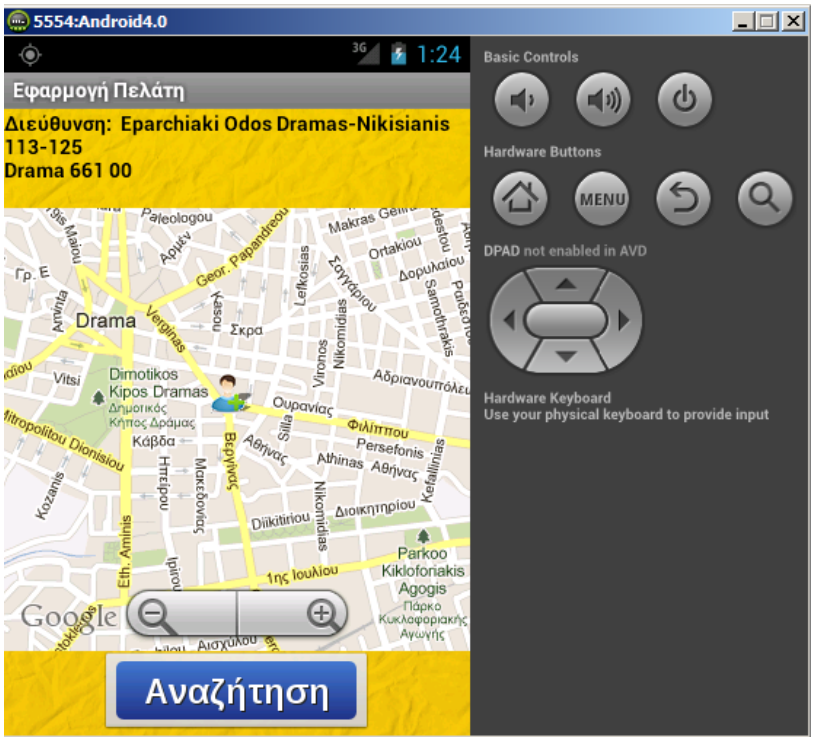
Εικόνα 3: Οθόνη εγγραφή χρήστη

Φυσικά για να ολοκληρωθεί η εγγραφή πρέπει το username και το e-mail του χρήστη να **μην** υπάρχουν ήδη.

Αφού ολοκληρώσει επιτυχώς την εγγραφή, επιστρέφει στον φόρμα εισόδου, και εάν δώσει σωστά τα στοιχεία του μπαίνει επιτυχώς στην εφαρμογή.

8.4 Διαδικασία αναζήτησης ταξί

Αφού κάνουμε εισαγωγή στην εφαρμογή θα μας εμφανιστεί η παρακάτω οθόνη. Αυτή είναι η κύρια οθόνη της εφαρμογής και από εδώ μπορούμε να αναζητήσουμε διαθέσιμους οδηγούς ταξί.



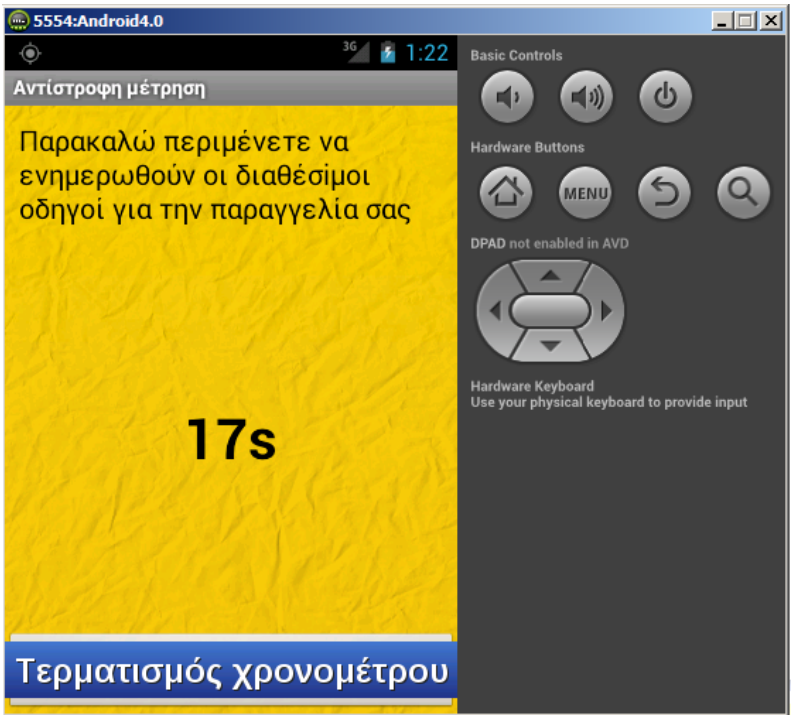
Εικόνα 4 :Βασική οθόνη αναζήτησης ταξί

Όπως μπορούμε να δούμε εμφανίζει πάνω στον χάρτη την τρέχουσα τοποθεσία μας, ακριβώς πάνω από τον χάρτη αναγράφεται αναλυτικά το πού βρισκόμαστε (οδός και αριθμός). Αν και το gps είναι αρκετά ακριβές με διαφορά το πολύ λίγων μέτρων ο πελάτης έχει την δυνατότητα να διαλέξει άλλο σημείο παραλαβής. Κάνοντας ένα tap πάνω στον χάρτη σε διαφορετικό σημείο από αυτό που λέει ότι βρισκόμαστε, θα αλλάξει η τοποθεσία και διεύθυνσή μας.

Όταν επιλέξουμε την διεύθυνση από την οποία θέλουμε να γίνει η παραλαβή πατάμε το κουμπί ***Αναζήτηση*** ώστε να ενημερωθούν οι διαθέσιμοι οδηγοί εκεί κοντά σχετικά με την παραγγελία μας.

8.5 Αναμονή αποτελεσμάτων

Αφού πατήσουμε ***Αναζήτηση*** δίνουμε χρόνο 45 δευτερολέπτων στους οδηγούς ταξί να δηλώσουν ενδιαφέρον ή όχι σχετικά με την παραγγελία μας. Εμείς μπορούμε να παρακολουθούμε πόσος χρόνος απομένει όπως φαίνετε παρακάτω.



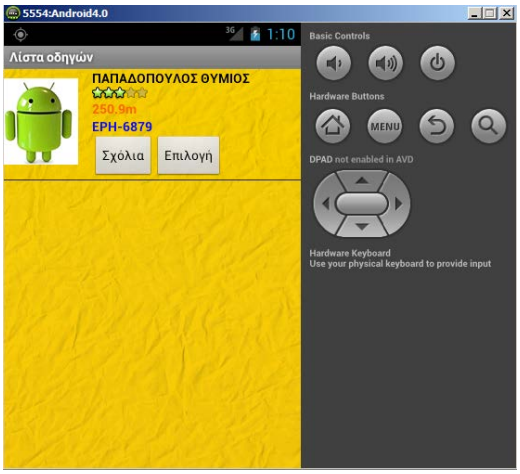
Εικόνα 5: Αναμονή χρονομέτρου εύρεσης

Φυσικά εάν κάποιος δεν θέλει να περιμένει και τα 45 δευτερόλεπτα μπορεί να πατήσει ***Τερματισμός χρονομέτρου*** και να του εμφανιστούν όσοι οδηγοί πρόλαβαν και δήλωσαν ενδιαφέρον για την παραγγελία. Είναι κάτι που **δεν** προτείνεται διότι πρέπει να έχουμε υπ όψιν μας ότι οι οδηγοί ίσως δεν κρατάν εκείνη την στιγμή το κινητό, άρα θα χρειαστούν λίγο χρόνο για να δουν την παραγγελία μας.

Αφού τελειώσει το χρονόμετρο σε περίπτωση που κανένας οδηγός δεν δήλωσε ενδιαφέρον για την παραγγελία, εμφανίζετε κατάλληλο μήνυμα και ο πελάτης έχει την δυνατότητα να ξανά στείλει το ίδιο αίτημα.

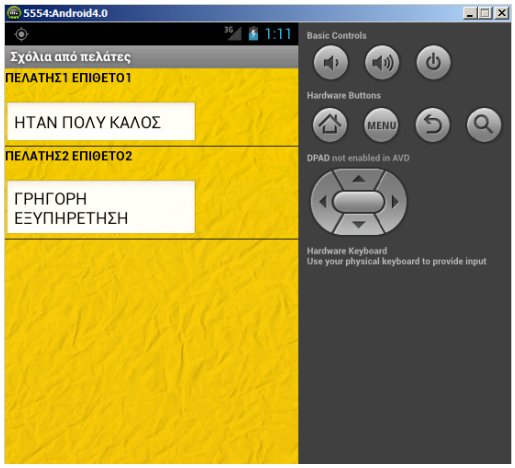
8.6 Διαδικασία επιλογής οδηγού ταξί

Εάν υπάρχουν οδηγοί που δήλωσαν ενδιαφέρον για την παραγγελία, θα εμφανιστούν όλοι σε μια λίστα όπως φαίνετε παρακάτω, ώστε να επιλέξει ο πελάτης με ποιον θέλει να κάνει την διαδρομή.



Εικόνα 6 :Οθόνη λίστας με οδηγούς

Ο πελάτης μπορεί να δει το όνομα του οδηγού, πόσο τον έχουν βαθμολογήσει όσοι πελάτες συνεργάστηκαν μαζί του, τις πινακίδες του αυτοκινήτου καθώς και αν θέλει, τα σχόλια που άφησαν άλλοι πελάτες για τον συγκεκριμένο οδηγό, πατώντας το κουμπί ***Σχόλια.***

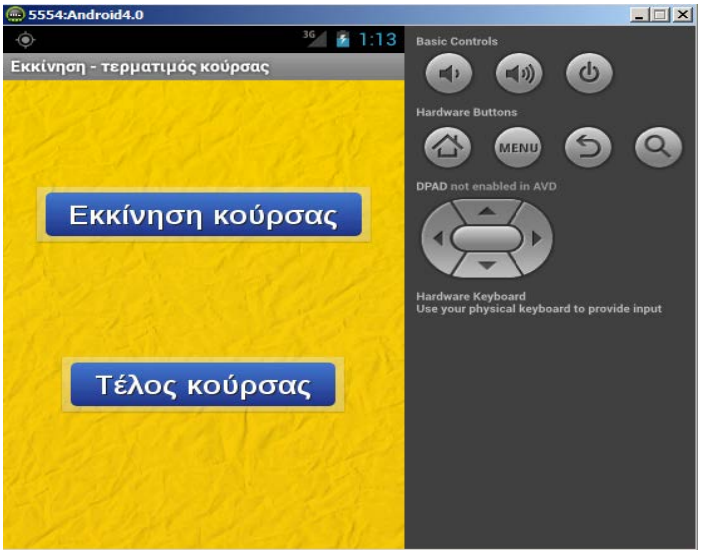


Εικόνα 7 :Οθόνη προβολής σχολίων

Πατώντας το πλήκτρο ***πίσω (***  ***)*** επιστρέφουμε στην λίστα με τους οδηγούς, και πατώντας το κουμπί ***Επιλογή*** στέλνουμε ειδοποίηση στον συγκεκριμένο οδηγό να έρθει να μας πάρει.

8.7 Έναρξη διαδρομής

Τώρα έχουμε μπροστά μας την παρακάτω οθόνη και περιμένουμε τον οδηγό να φτάσει εδώ που είμαστε.



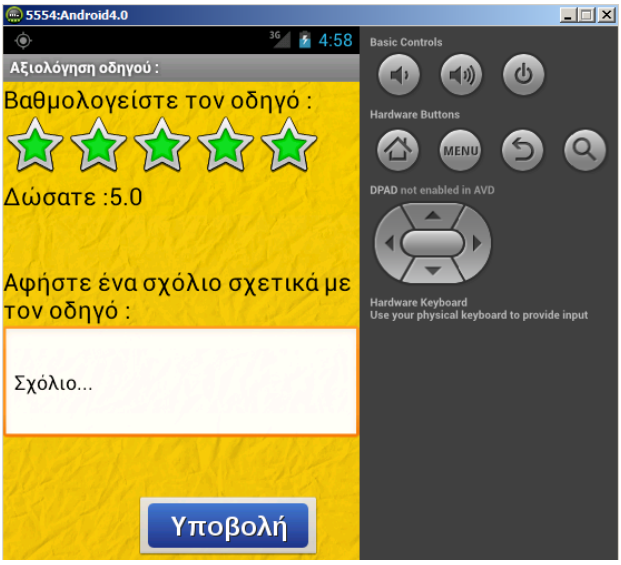
Εικόνα 8 :Οθόνη έναρξης-τερματισμού κούρσας

Αρχικά ο πελάτης δεν μπορεί να πατήσει το ***Τέλος κούρσας*** παρά μόνο το ***Εκκίνηση κούρσας.*** Για να θεωρηθεί ότι ξεκίνησε η διαδρομή θα πρέπει και ο οδηγός να πατήσει αντίστοιχο πλήκτρο από την δική του έκδοση της εφαρμογής.

Όταν ολοκληρωθεί η διαδρομή ο πελάτης μπορεί να πατήσει το ***Τέλος κούρσας*** για να συνεχίσει στην βαθμολόγηση και σχολιασμό του οδηγού.

8.8 Διαδικασία βαθμολόγησης και σχολιασμού οδηγού

Αφού ολοκληρωθεί η διαδρομή, ο πελάτης έχει την δυνατότητα να βαθμολογήσει ή/και να σχολιάσει τον οδηγό. Καλό θα ήταν να ασχοληθεί με αυτό και να μην το αφήσει διότι έτσι θα βοηθήσει και άλλους πελάτες να επιλέξουν ευκολότερα οδηγό την επόμενη φορά που θα χρησιμοποιήσουν την εφαρμογή για αναζήτηση ταξί.



Εικόνα 9 :Οθόνη βαθμολογίας

Τόσο η βαθμολογία όσο και το σχόλιο που θα αφήσει ο χρήστης θα είναι ορατά για όποιο πελάτη τύχει να συνεργαστεί με τον συγκεκριμένο οδηγό. Πατώντας ***Υπολοβή*** η βαθμολογία και τα σχόλια που αφήσαμε θα αποθηκευτούν.

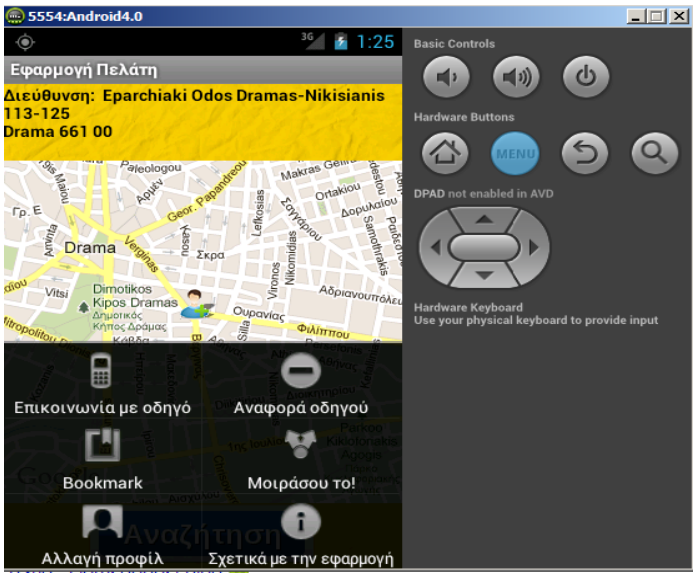
8.9 Πλήκτρο Menu

Όλα τα κινητά android που πωλούνται στην Ευρώπη έχουν ένα πλήκτρο μενού (). Πατώντας το ο πελάτης έχει τις εξής δυνατότητες.

- Επικοινωνία με οδηγό
- Αναφορά οδηγού
- Bookmark
- Μοιράσου το!
- Αλλαγή προφίλ
- Σχετικά με την εφαρμογή

Να σημειωθεί ότι οι παραπάνω επιλογές είναι ορατές σε όλα τα σημεία της εφαρμογής αλλά δεν λειτουργούν μερικά συνέχεια. Για παράδειγμα εάν ο πελάτης επιλέξει ***Αναφορά οδηγού*** χωρίς να έχει γίνει κάποια κλήση για ταξί θα του εμφανιστεί κατάλληλο μήνυμα και δεν θα έχει σε ποιόν να κάνει αναφορά!

Το μενού εμφανίζεται όπως φαίνετε παρακάτω:

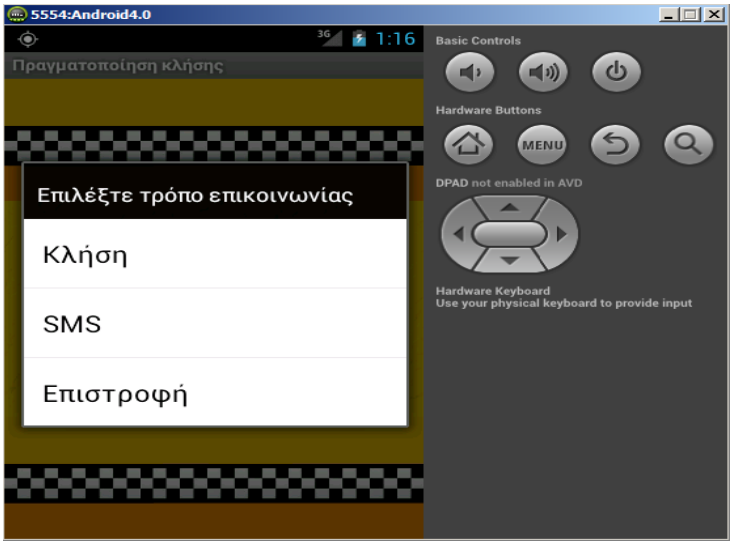


Εικόνα 10 :Βασικό menu

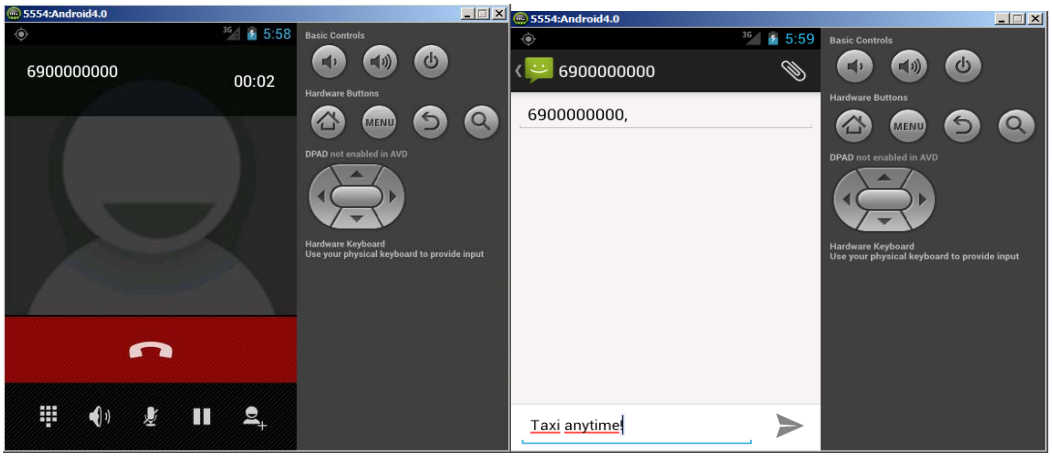
8.9.1 Επικοινωνία με οδηγό

Η λειτουργία αυτή είναι διαθέσιμη από την στιγμή που κάποιος πελάτης έχει επιλέξει τον συγκεκριμένο οδηγό για την παραγγελία.

Για τον οποιοδήποτε λόγο ίσως ο πελάτης χρειαστεί να επικοινωνήσει με τον οδηγό για κάτι, έχει την δυνατότητα να το κάνει είτε μέσω μιας κλήσης είτε με το να του στείλει μήνυμα. Αυτό γίνετε όπως φαίνετε παρακάτω.



Εικόνα 11 :Επιλογές επικοινωνίας



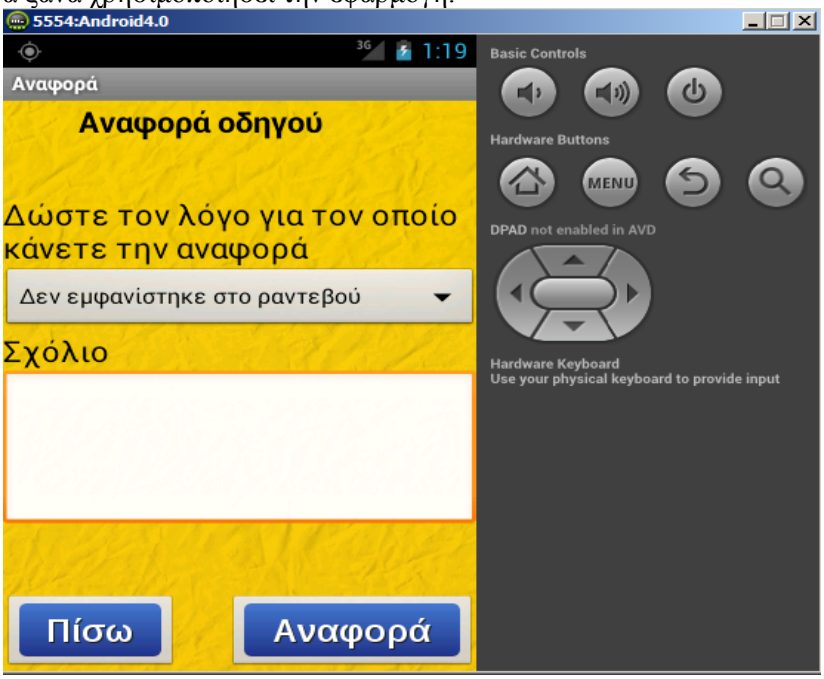
Εικόνα 12 :Οθόνη κλήσης

Εικόνα 13 :Οθόνη αποστολής sms

8.9.2 Αναφορά οδηγού

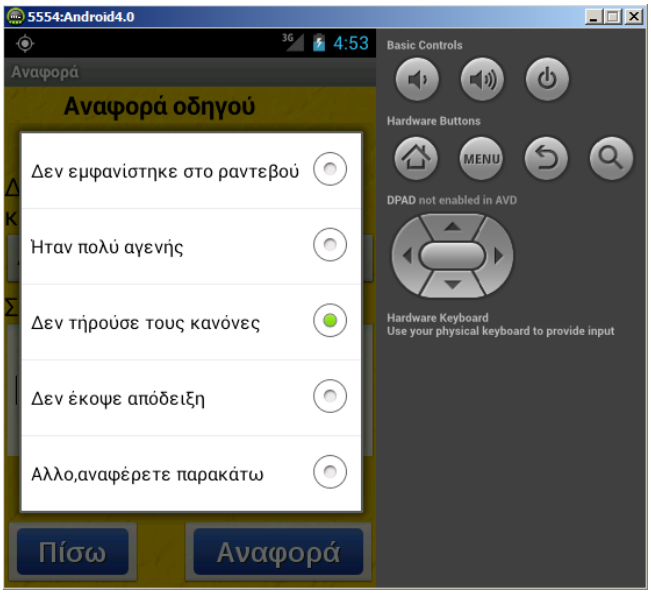
Όπως και πριν, η λειτουργία αυτή είναι διαθέσιμη από την στιγμή που κάποιος πελάτης έχει επιλέξει τον συγκεκριμένο οδηγό για την παραγγελία.

Εδώ εάν για κάποιο λόγο ο πελάτης κρίνει ότι ο οδηγός δεν φέρθηκε σωστά και δεν πρέπει να χρησιμοποιεί την εφαρμογή μπορεί να του κάνει αναφορά. Εάν ένας οδηγός δεχτεί πέντε αναφορές δεν θα μπορεί να ξανά χρησιμοποιήσει την εφαρμογή.



Εικόνα 14: Οθόνη αναφοράς οδηγού

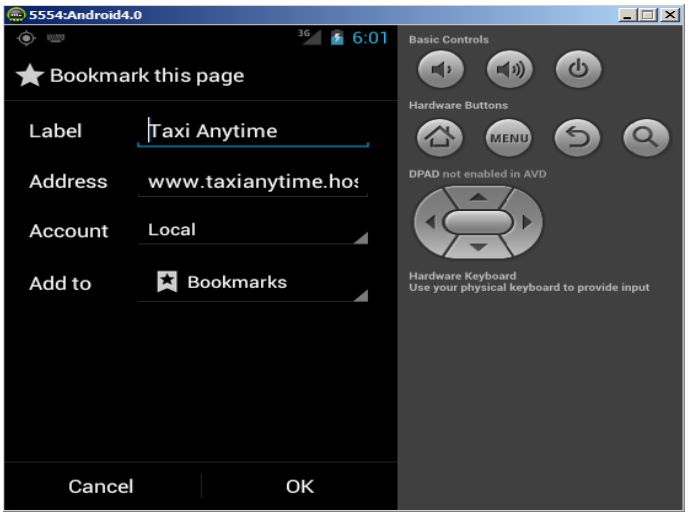
Υπάρχουν κάποιοι σταθεροί λόγοι για τους οποίους ενδεχομένως να θέλει ο πελάτης να κάνει αναφορά στον οδηγό. Εάν ο λόγος δεν είναι στην λίστα μπορεί φυσικά να γράψει τον δικό του.



Εικόνα 15 :Επιλογή λόγων αναφοράς

8.9.3 Bookmark

Εδώ μπορεί ο χρήστης να προσθέσει στους σελιδοδείκτες του και το site της εφαρμογής. Η λειτουργία αυτή είναι διαθέσιμη σε όλα τα σημεία της εφαρμογής.



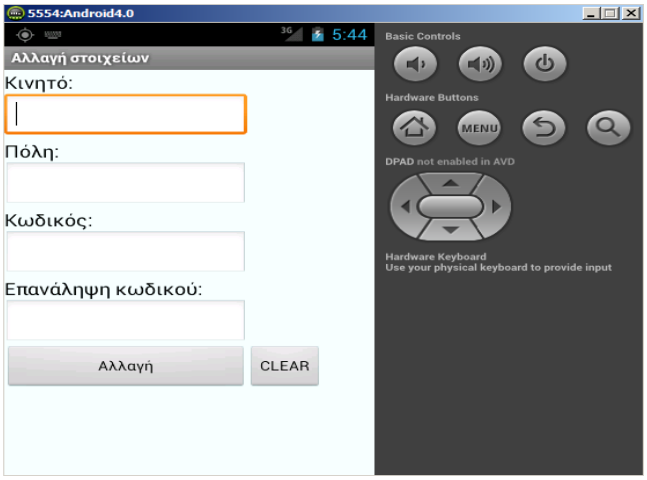
Εικόνα 16 :Εισαγωγή στα αγαπημένα

8.9.4 Μοιράσου το!

Εάν θέλει ο χρήστης να πει και σε άλλους για την εφαρμογή μπορεί να το κάνει από εδώ. Η συγκεκριμένη επιλογή δεν λειτουργεί σε όλα τα κινητά με τον ίδιο τρόπο! Εμφανίζετε μια λίστα με τους τρόπους που έχει ο χρήστης να πει για την εφαρμογή. Με sms, μέσω facebook εάν είναι εγκατεστημένο στο κινητό, μέσω twitter, google+, Bluetooth κλπ. Ανάλογα τι εφαρμογές έχει εγκατεστημένες στο κινητό ο χρήστης.

8.9.5 Αλλαγή προφίλ

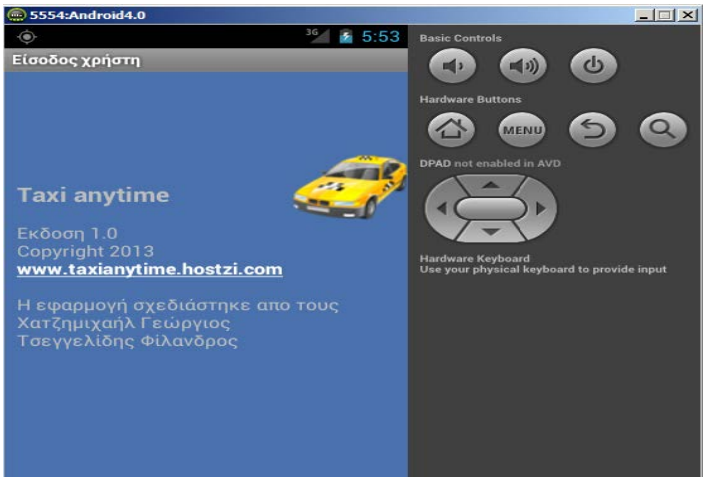
Εάν ο χρήστης έχει αλλάξει αριθμό κινητού, e-mail κ.α. Μπορεί να ενημερώσει το προφίλ του από εδώ. Αυτά που δεν μπορεί να αλλάξει είναι το όνομά του, το επίθετό του και το username του.



Εικόνα 17 :Οθόνη αλλαγής στοιχείων προφίλ

8.9.6 Σχετικά με την εφαρμογή

Εδώ μπορεί ο χρήστης να δει πληροφορίες σχετικά με την εφαρμογή.



Εικόνα 18 :Σχετικά με

9 Οδηγίες χρήσης εφαρμογής οδηγού

9.1 Εισαγωγή

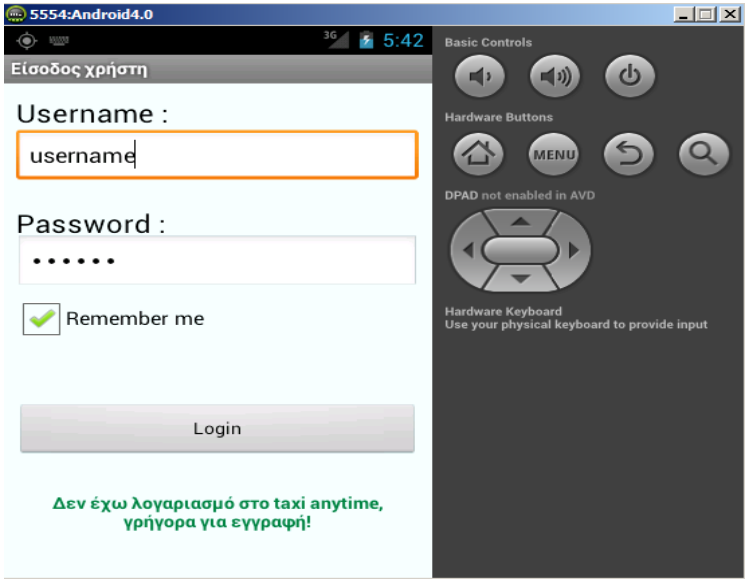
Με το που ανοίγουμε την εφαρμογή εμφανίζεται μια εισαγωγική οθόνη (splash screen) έως ότου γίνουν οι απαραίτητες διαδικασίες για να ξεκινήσει η εφαρμογή. Από την εισαγωγική οθόνη μπορούμε επίσης να δούμε εάν κατεβάσαμε την σωστή έκδοση της εφαρμογής.

Εικόνα 1: Εισαγωγική οθόνη

Εάν γράφει driver version έχουμε κατεβάσει την εφαρμογή για οδηγούς ταξί.

9.2 Είσοδος στην εφαρμογή

Στην συνέχεια πρέπει ο χρήστης να δώσει username & password για να κάνει είσοδο στην εφαρμογή.



Εικόνα 2 :Οθόνη εισόδου

Εάν ο χρήστης έχει ήδη κάνει εγγραφή στην εφαρμογή μπορεί αφού δώσει τα σωστά δεδομένα (username & password) μπορεί να επιλέξει το ***Remember me*** ώστε κάθε επόμενη φορά να με το που πάει να κάνει είσοδο να είναι ήδη συμπληρωμένα αυτά τα πεδία. Εάν ο χρήστης δεν έχει λογαριασμό πατώντας στο κείμενο ***Δεν έχω λογαριασμό […]*** τον παραπέμπουμε σε μια φόρμα για να κάνει εγγραφή.

9.3 Εγγραφή στην εφαρμογή

Ο χρήστης πρέπει να δώσει τις παρακάτω πληροφορίες με τους συγκεκριμένους περιορισμούς:

- Το <u>όνομα</u> πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείται μόνο από γράμματα.
- Το <u>επώνυμο</u> πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείται μόνο από γράμματα.
- <u>Ημερομηνία γέννησης</u> .
- Το κινητό πρέπει να αποτελείται τουλάχιστον από 10 αριθμούς. Μόνο αριθμούς.
- Η <u>πόλη</u> πρέπει να είναι πάνω από 2 χαρακτήρες και να αποτελείται μόνο από γράμματα.

- Αριθμό πινακίδας του ταξί.

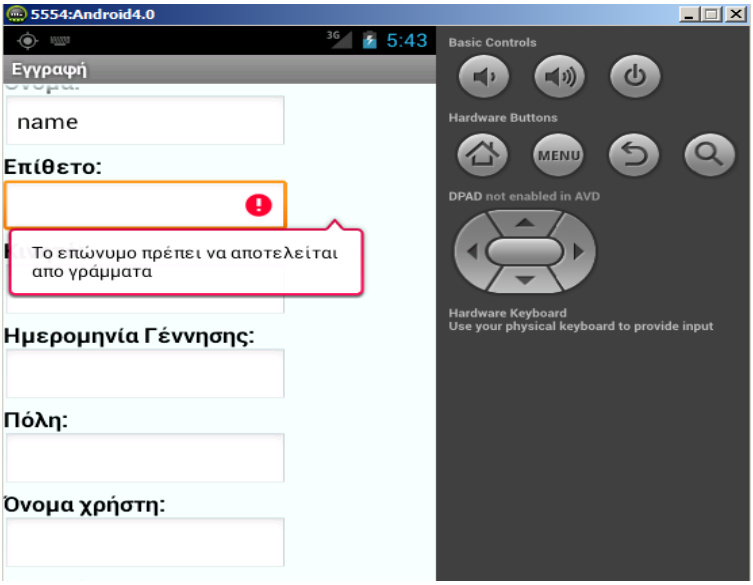
- Το username πρέπει να είναι πάνω από 2 χαρακτήρες και να αρχίζει με γράμμα ή κάτω παύλα.

- Ο κωδικός πρέπει να είναι τουλάχιστον 6 ψηφία

- Το email πρέπει να είναι τουλάχιστον 4 χαρακτήρες, να αρχίζει με γράμμα, αριθμό, _, + ή -, πρέπει να υπάρχει το @ και . και μετά την . να έχουμε από 2 έως 4 χαρακτήρες.

-

Εάν ο χρήστης εισάγει κάτι λάθος δεν θα γίνει η εγγραφή και θα εμφανιστεί που είναι το λάθος όπως φαίνετε παρακάτω:



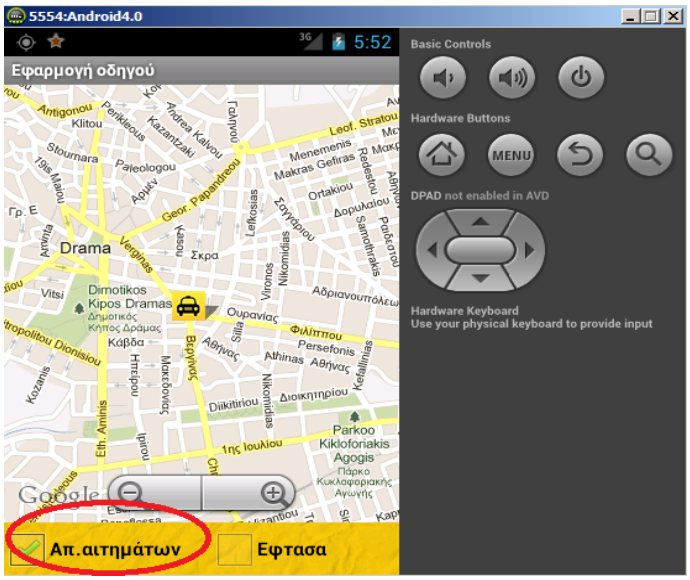
Εικόνα 3: Οθόνη εγγραφής

Φυσικά για να ολοκληρωθεί η εγγραφή πρέπει το username και το e-mail του χρήστη να μην υπάρχουν ήδη.

Αφού ολοκληρώσει επιτυχώς την εγγραφή, επιστρέφει στον φόρμα εισόδου, και εάν δώσει σωστά τα στοιχεία του μπαίνει επιτυχώς στην εφαρμογή.

9.4 Διαδικασία επικοινωνίας με πελάτες

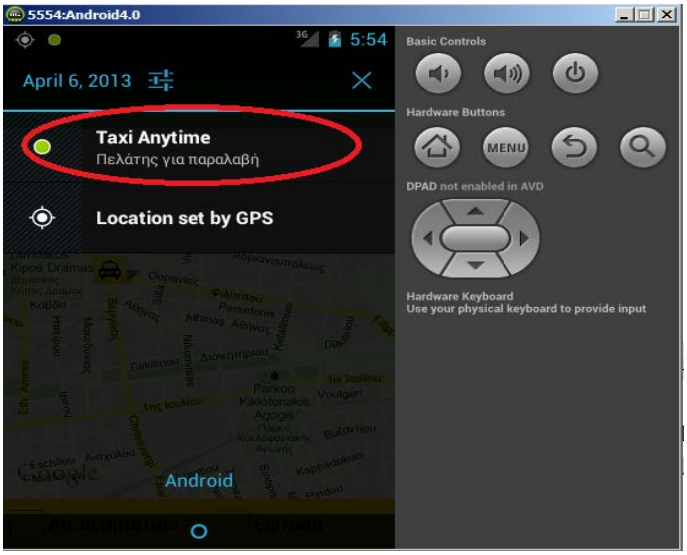
Αφού εισέρθει στην εφαρμογή ο χρήστης του εμφανίζεται η παρακάτω οθόνη



Εικόνα 4 : Βασική οθόνη

Εδώ μπορεί να δει την τοποθεσία του στον χάρτη. Αν δεν είναι επιλεγμένο το ***Απ. Αιτημάτων*** (Αποδοχή αιτημάτων) ο οδηγός θεωρείται απασχολημένος και δεν μπορεί να λάβει ειδοποιήσεις από πελάτες. Όταν επιλέξει το ***Απ. Αιτημάτων*** και κάποιος πελάτης εκεί κοντά ψάχνει ταξί μέσω της έκδοσης για πελάτες, θα λάβει μια ειδοποίηση ο οδηγός

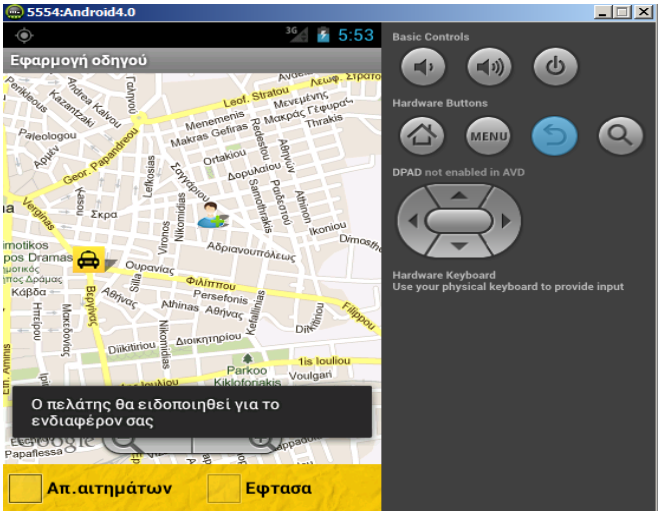
με την τοποθεσία του πελάτη. Όταν φτάσει η ειδοποίηση θα κάνει έναν ήχο στο κινητό άρα ο οδηγός δεν είναι υποχρεωμένος να έχει την προσοχή του στο κινητό.



Εικόνα 5 : Ειδοποίηση από πελάτη

Όταν ο οδηγός ανοίξει την ειδοποίηση θα δει την τοποθεσία που βρίσκεται ο ίδιος, και αυτή του πελάτη. Εάν επιθυμεί να δηλώσει ενδιαφέρον για την παραγγελία το μόνο που έχει να κάνει είναι ένα tap πάνω στο εικονίδιο του πελάτη. Αφού το κάνει, θα του εμφανιστεί ένα μήνυμα επιβεβαίωσης.

Πλέον ο οδηγός περιμένει να δει εάν ο συγκεκριμένος πελάτης τον έχει επιλέξει για την παραγγελία. Ο πελάτης θα μπορεί να δει χρήσιμες πληροφορίες όπως τα στοιχεία του οδηγού, τις πινακίδες του αυτοκινήτου του καθώς και πώς τον βαθμολόγησαν άλλοι χρήστες που συνεργάστηκαν μαζί του ή ότι άλλα σχόλια έχουν κάνει γι αυτόν.

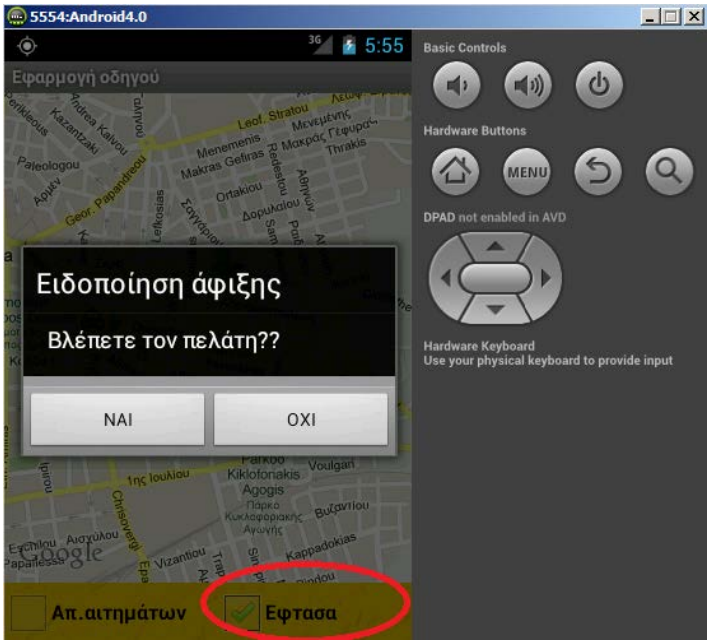


Εικόνα 6 : Οθόνη επιλογής πελάτη

Εάν τελικά ο πελάτης τον επιλέξει για την παραγγελία, ο οδηγός θα λάβει άλλη μια ειδοποίηση που θα τον ενημερώνει γι αυτό.

9.5 Διαδικασία παραλαβής και μεταφοράς πελάτη

Όταν λοιπόν γίνει η επιλογή του οδηγού, ο οδηγός βλέπει στην οθόνη του την αρχική του θέση, και το σημείο παραλαβής του πελάτη. Όταν βλέπει τον πελάτη και τον πλησιάσει, θα πρέπει να πατήσει το ***Έφτασα,*** και τότε θα του εμφανιστεί ένα μήνυμα επιβεβαίωσης ότι όντως βρήκε τον πελάτη.



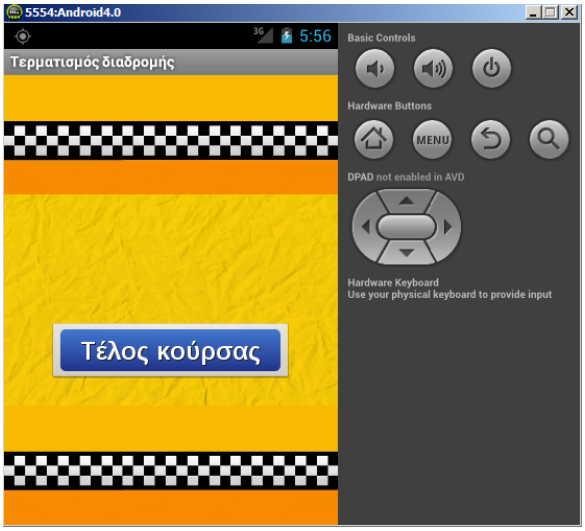
Εικόνα 7 : Ειδοποίηση πελάτη για άφιξη

Εάν και ο πελάτης δηλώσει μέσω της δικής του εφαρμογής ότι έχει βρει τον οδηγό, η διαδικασία της μεταφοράς μπορεί να ξεκινήσει.

9.6 Τερματισμός παραγγελίας & αναμονή για νέο πελάτη

Κατά την διάρκεια της διαδρομής ο οδηγός δεν έχει δικαίωμα να δεχτεί άλλα αιτήματα από πελάτες. Θα πρέπει πρώτα να ολοκληρώσει την τωρινή παραγγελία πρώτα.

Για να το κάνει αυτό θα πρέπει να πατήσει απλά το κουμπί ***Τέλος κούρσας.*** Στην παρακάτω οθόνη βρίσκεται μετά την παραλαβή του πελάτη.



Εικόνα 8 : Τερματισμός παραγγελίας

Αφού λοιπόν ολοκληρωθεί η διαδρομή και πατήσει το κουμπί ο οδηγός, θα μεταφερθεί στην αρχική οθόνη μετά την είσοδο στην εφαρμογή και θα είναι σε θέση να δεχτεί και άλλες παραγγελίες.

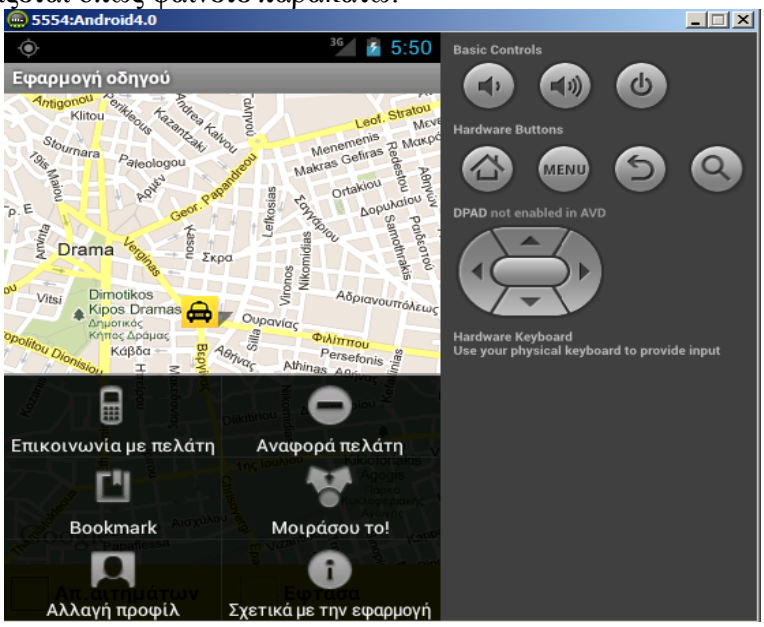
9.7 Πλήκτρο Menu

Όλα τα κινητά android που πωλούνται στην Ευρώπη έχουν ένα πλήκτρο μενού (). Πατώντας το ο οδηγός ταξί έχει τις εξής δυνατότητες.

- Επικοινωνία με πελάτη
- Αναφορά πελάτη
- Bookmark
- Μοιράσου το!
- Αλλαγή προφίλ
- Σχετικά με την εφαρμογή

Να σημειωθεί ότι οι παραπάνω επιλογές είναι ορατές σε όλα τα σημεία της εφαρμογής αλλά δεν λειτουργούν μερικά συνέχεια. Για παράδειγμα εάν ο οδηγός επιλέξει ***Αναφορά πελάτη*** χωρίς να έχει γίνει κάποια κλήση για ταξί θα του εμφανιστεί κατάλληλο μήνυμα και δεν θα έχει σε ποιόν να κάνει αναφορά!

Το μενού εμφανίζεται όπως φαίνετε παρακάτω:

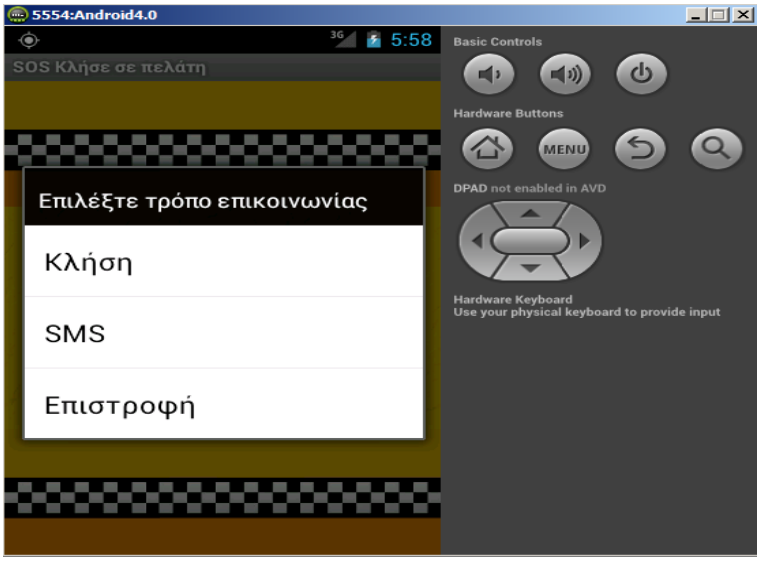


Εικόνα 9 : Βασικά menu

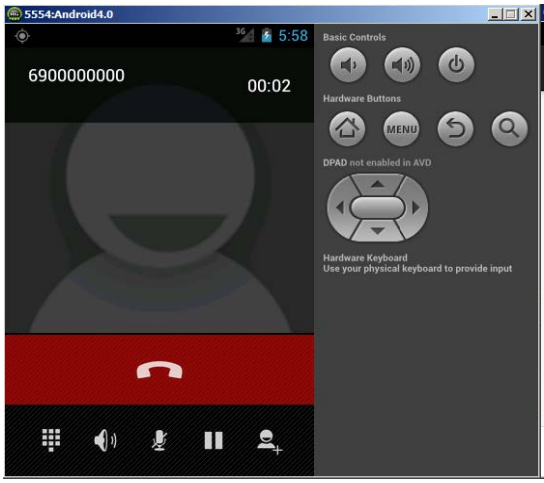
9.7.1 Επικοινωνία με πελάτη

Η λειτουργία αυτή είναι διαθέσιμη από την στιγμή που κάποιος πελάτης έχει επιλέξει τον συγκεκριμένο οδηγό για την παραγγελία.

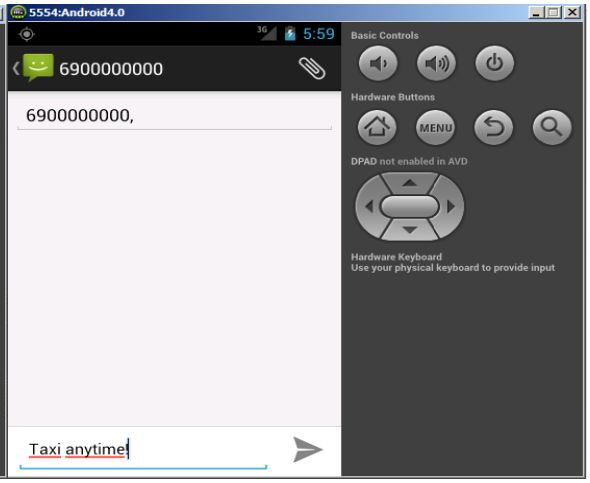
Για τον οποιοδήποτε λόγο ίσως ο οδηγός χρειαστεί να ενημερώσει τον πελάτη για κάτι όπως, «έπεσα σε κίνηση, θα αργήσω 10 λεπτά» ή οτιδήποτε άλλο, έχει την δυνατότητα να επικοινωνήσει με τον πελάτη είτε με το να του κάνει μια κλήση είτε με το να του στείλει μήνυμα. Αυτό γίνετε όπως φαίνετε παρακάτω.



Εικόνα 10 : Οθόνη επικοινωνίας



Εικόνα 11: Οθόνη κλήσης

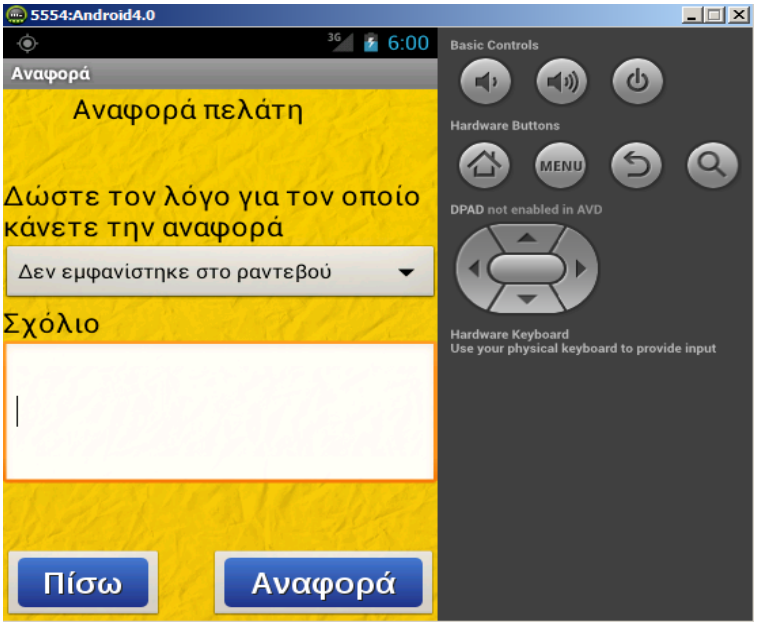


Εικόνα 12: Οθόνη αποστολής sms

9.7.2 Αναφορά πελάτη

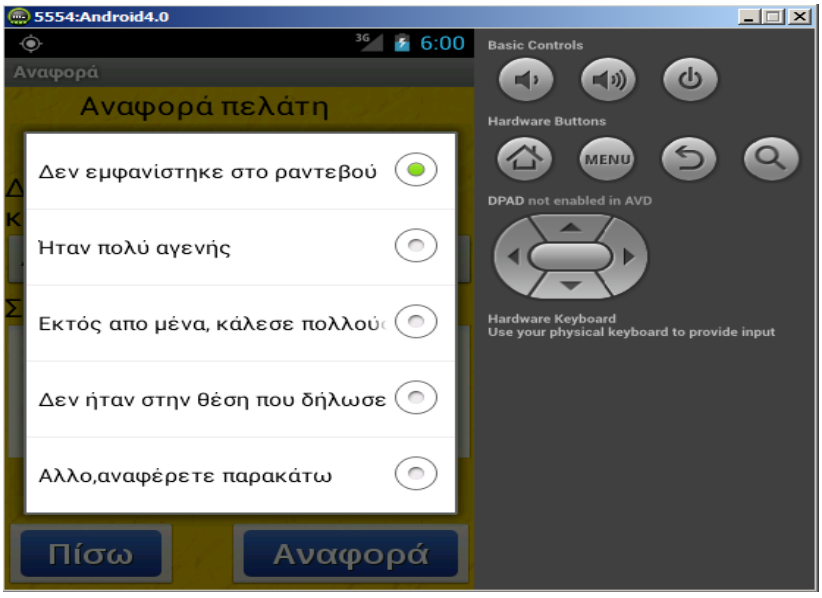
Όπως και πριν, η λειτουργία αυτή είναι διαθέσιμη από την στιγμή που κάποιος πελάτης έχει επιλέξει τον συγκεκριμένο οδηγό για την παραγγελία.

Εδώ εάν για κάποιο λόγο ο οδηγός κρίνει ότι ο πελάτης δεν φέρθηκε σωστά και δεν πρέπει να χρησιμοποιεί την εφαρμογή μπορεί να του κάνει αναφορά. Εάν ένας πελάτης δεχτεί πέντε αναφορές δεν θα μπορεί να ξανά χρησιμοποιήσει την εφαρμογή.



Εικόνα 13: Οθόνη αναφοράς πελάτη

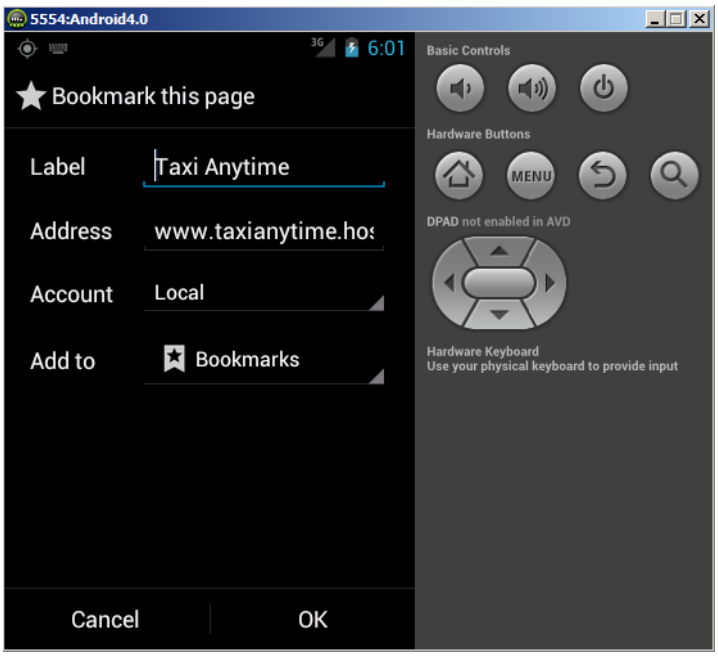
Υπάρχουν κάποιοι σταθεροί λόγοι για τους οποίους ενδεχομένως να θέλει ο οδηγός να κάνει αναφορά στον πελάτη. Εάν ο λόγος δεν είναι στην λίστα μπορεί φυσικά να γράψει τον δικό του.



Εικόνα 14: Επιλογή λόγων αναφοράς

9.7.3 Bookmark

Εδώ μπορεί ο χρήστης να προσθέσει στους σελιδοδείκτες του και το site της εφαρμογής. Η λειτουργία αυτή είναι διαθέσιμη σε όλα τα σημεία της εφαρμογής.



Εικόνα 15 :Εισαγωγή στα αγαπημένα

9.7.4 Μοιράσου το!

Εάν θέλει ο χρήστης να πει και σε άλλους για την εφαρμογή μπορεί να το κάνει από εδώ. Η συγκεκριμένη επιλογή δεν λειτουργεί σε όλα τα κινητά με τον ίδιο τρόπο! Εμφανίζεται μια λίστα με τους τρόπους που έχει ο χρήστης να πει για την εφαρμογή. Με sms, μέσω facebook εάν είναι εγκατεστημένο στο κινητό, μέσω twitter, google+, Bluetooth κλπ. Ανάλογα τι εφαρμογές έχει εγκατεστημένες στο κινητό ο χρήστης.

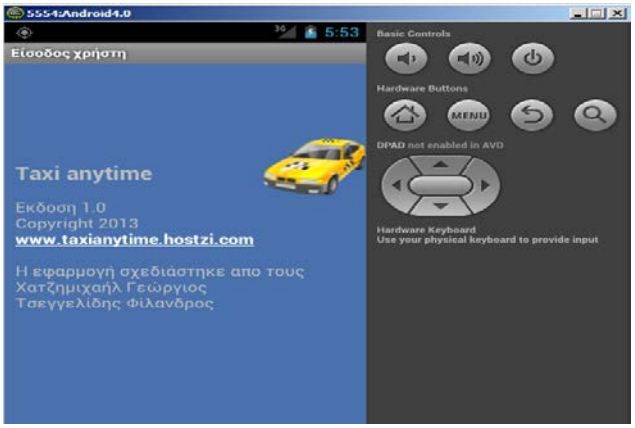
9.7.5 Αλλαγή προφίλ

Εάν ο χρήστης έχει αλλάξει αριθμό κινητού, e-mail κ.α. Μπορεί να ενημερώσει το προφίλ του από εδώ. Αυτά που δεν μπορεί να αλλάξει είναι το όνομά του, το επίθετό του και το username του.

Εικόνα 16: Οθόνη αλλαγής στοιχείων προφίλ

9.7.6 Σχετικά με την εφαρμογή

Εδώ μπορεί ο χρήστης να δει πληροφορίες σχετικά με την εφαρμογή.



Εικόνα 17 :Σχετικά με

10 Συστήματα Διαχείρισης Περιεχομένου (CMS)

10.1 Εισαγωγή

Όπως κάθε εφαρμογή, κατά την γνώμη μας, χρειάζεται και μια ιστοσελίδα για υποστήριξη, ενημέρωση & βοήθεια των χρηστών με ότι έχει να κάνει γύρο από την εφαρμογή και τις εκδόσείς της, έτσι και η δική μας έχει την ιστοσελίδα www.taxianytime.hostzi.com .

Στο κεφάλαιο αυτό θα αναλύσουμε μέχρι τα CMS και θα παρουσιάσουμε τους λόγους για τους οποίους επιλέξαμε ένα, για την κατασκευή της ιστοσελίδας μας.

Να σημειωθεί ότι η ιστοσελίδα της εφαρμογής **δεν** είναι στις απαιτήσεις της παρούσας πτυχιακής και για τον λόγο αυτό δεν θα εμβαθύνουμε στην ανάλυσή της. Αποφασίσαμε μόνοι μας να την κάνουμε.

10.2 Τί είναι το Σύστημα Διαχείρισης Περιεχομένου (CMS) ;

Το Content Management System (CMS) είναι μία μορφή λογισμικού που αυτοματοποιεί τις διαδικασίες δημιουργίας, οργάνωσης, ελέγχου και δημοσίευσης περιεχομένου σε μία πληθώρα μορφών. Τα περισσότερα CMS έχουν την δυνατότητα να διαχειριστούν περιεχόμενο στις εξής μορφές: κείμενα, εικόνες, βίντεο, java animation, πρότυπα σχεδίασης, βάσεις δεδομένων κ.α. Το Σύστημα Διαχείρισης Περιεχομένου είναι λογισμικό το οποίο επιτρέπει στον οποιονδήποτε, ακόμα και αν δεν έχει ιδιαίτερες γνώσεις προγραμματισμού και γλώσσας HTML, να δημιουργήσει και να διαχειριστεί με τρόπο εύκολο και γρήγορο την ιστοσελίδα του.

10.3 Τί προσφέρει το Σύστημα Διαχείρισης Περιεχομένου;

Το Σύστημα Διαχείρισης Περιεχομένου προσφέρει γραφικό περιβάλλον το οποίο δίνει άμεση πρόσβαση στο περιεχόμενο της ιστοσελίδας. Επιπρόσθετα, η τροποποίηση ή προσθήκη του περιεχομένου (κειμένου και φωτογραφιών) μπορεί να γίνει με έναν γραφικό editor όμοιο με αυτόν που χρησιμοποιείται στους επεξεργαστές κειμένου. Η πληροφορία οργανώνεται αποδοτικά σε κατηγορίες και υποκατηγορίες και παρουσιάζεται με τρόπο φιλικό στο χρήστη αλλά και στο διαχειριστή.

10.4 Ποια είναι τα πλεονεκτήματα από την χρήση ανοικτού κώδικα CMS;

- Μείωση των εξόδων για την διατήρηση μίας ιστοσελίδας
- Με την βοήθεια των CMS μπορεί να αυξηθεί κατακόρυφα η ποιότητα μίας ιστοσελίδας με την χρήση υψηλής ποιότητας προτύπων σχεδίασης, που θα δίνουν μία εντυπωσιακή εικόνα για τον ιδιοκτήτη τους.
- Λιγότερες ανάγκες εκπαίδευσης. Με τις έτοιμες φόρμες εισαγωγής, μορφοποίησης και προεπισκόπησης, που προσφέρουν, δεν απαιτούνται πλέον ειδικές γνώσεις προγραμματισμού και σχεδίασης ιστοσελίδων.
- Παρέχει τη δυνατότητα αυτοματοποίησης των διεργασιών ρουτίνας. Π.χ εφαρμόζει την ίδια μορφοποίηση σε όλες τις ιστοσελίδες. Οι επιλογές και γενικότερα η πλοήγηση αναπαράγεται επίσης αυτόματα.
- Παρέχεται μεγαλύτερη ομοιομορφία και συνοχή, βελτιωμένο σύστημα πλοήγησης, αυξημένη ευελιξία και επιτάχυνση της διαδικασίας αλλαγών και δημιουργίας νέων σελίδων.

10.5 Ποια είναι τα μειονεκτήματα από την χρήση ανοικτού κώδικα CMS;

- Το ότι είναι δωρεάν δεν σημαίνει ότι δεν κοστίζουν τίποτα. Πρώτα από όλα πρέπει να δοθεί έμφαση στο γεγονός ότι ενώ τα open-source CMS είναι δωρεάν αυτό δεν σημαίνει ότι δεν κοστίζουν και τίποτα. Μεγάλο μέρος της προσπάθειας αλλά και του κόστους αναφέρεται στην υλοποίηση καθαυτή καθώς και στην διαδικασία τροποποίησης.
- Είναι πιθανόν να μη μπορούν να υποστηρίξουν δημιουργία μεγάλων κόμβων. Τα περισσότερα open-source CMS εστιάζουν σε μικρές ή μεσαίες υλοποιήσεις και δεν έχουν τα χαρακτηριστικά των εμπορικών CMS που απευθύνονται σε μεγάλες επιχειρήσεις.
- Έλλειψη εμπορικής υποστήριξης. Τα open-source CMS πάσχουν από έλλειψη εμπορικής υποστήριξης γεγονός το οποίο δεν συμβαίνει στα εμπορικά CMS.
- Μικρότερη ωριμότητα. Η πλειοψηφία των CMS που βασίζονται σε κοινότητες (community based) αποτελούν λιγότερο ώριμα συστήματα από τα ανάλογα εμπορικά. Αυτό συμβαίνει λόγο του έντονου ανταγωνισμού που υπάρχει μεταξύ των εμπορικών CMS.

10.6 Λειτουργικά χαρακτηριστικά CMS

Με τον όρο λειτουργικά χαρακτηριστικά εννοούμε το σύνολο των δυνατοτήτων που προσφέρει μια πλατφόρμα CMS για την διαχείριση των οντοτήτων της.

Ακολουθεί μια παράθεση όλων των λειτουργικών χαρακτηριστικών ενός CMS. Δεξιά από κάθε λειτουργικό χαρακτηριστικό βρίσκεται εντός παρένθεσης η σήμανση Υ/ Π, που αντιστοιχεί στα υποχρεωτικά και τα προχωρημένα χαρακτηριστικά ενός συστήματος αντίστοιχα.

10.6.1 Χρήστες και ομάδες χρηστών

10.6.1.1 Συλλογή

- Δυνατότητα ασφαλούς εισόδου και εξόδου (login - logout) στο / από το σύστημα και αναγνώρισης από αυτό όλων των χρηστών με βάση ένα username και password (Υ)
- Ορισμός νέου χρήστη (είτε με διαδικασία εγγραφής που εκτέλει ο ίδιος ο χρήστης είτε με ενέργεια του διαχειριστή χρηστών) με αρχικοποίηση ιδιοτήτων όπως ένα μοναδικό user ID, name, type (τύπος ομάδας στην οποία ανήκει) (Υ)
- Εύρεση / ανανέωση στοιχείων χρηστών από ένα LDAP[(4)] directory (Π)

10.6.1.2 Διαχείριση

- Παρουσίαση των μετα-δεδομένων (π.χ. στοιχεία επικοινωνίας) ενός χρήστη (Υ)
- Δυνατότητα απομακρυσμένης εισόδου στο σύστημα ενός χρήστη με δικαιώματα διαχείρισης περιεχομένου ή χρηστών, ώστε να προβεί σε αναγκαίες αλλαγές. Η απομακρυσμένη πρόσβαση να μπορεί να γίνεται βεβαίως με ασφαλή τρόπο, χρησιμοποιώντας κατάλληλα πρωτόκολλα κρυπτογράφησης και αναγνώρισης της ταυτότητας του χρήστη (Π)
- Ενεργοποίηση και απενεργοποίηση υπάρχοντος χρήστη (Υ)
- Τοποθέτηση χρήστη σε μία ή περισσότερες ομάδες χρηστών (Υ)
- Εύρεση και παρουσίαση των χρηστών που έχουν δικαιώματα πρόσβασης σε έναν κόμβο της ιεραρχίας ή σε ένα στοιχείο περιεχομένου της εφαρμογής (Υ)
- Δυνατότητα σύνδεσης χρηστών σαν guests με περιορισμένα ένα permissions (Υ)
- Υποστήριξη για το σύστημα NIS (Network Information System) του Unix. Το NIS παλιότερα ονομαζόταν YP (Yellow Pages) και η βασική του λειτουργία είναι η κοινή χρήση, από πολλά μηχανήματα, πληροφοριών ρυθμίσεων (configuration info). Στα περισσότερα συστήματα το NIS αντικαταστάθηκε από το πρωτόκολλο LDAP (Π)
- Δυνατότητα ενσωμάτωσης με το ευρετήριο δικτύου των Windows (Active Directory) για έλεγχο και ταυτοποίηση χρηστών (Π)
- Υποστήριξη της προδιαγραφής PAM (Pluggable Authentication Modules) του Linux. Το PAM είναι μία τεχνολογία που επιτρέπει την συγγραφή προγραμμάτων τα οποία μπορούν να χρησιμοποιούν προχωρημένες μεθόδους ταυτοποίησης χρήστη (user authentication) αλλά είναι ανεξάρτητα από το σχήμα ταυτοποίησης που χρησιμοποιείται κάθε φορά, όπως για παράδειγμα ένα απλό σχήμα “Username –

password” ή κάτι πιο εξελιγμένο όπως μια smart card. Αυτό επιτυγχάνεται με το «φόρτωμα» κάποιων συστατικών κατά το χρόνο εκτέλεσης με την μέριμνα του διαχειριστή συστήματος (Π)

10.6.2 Ρόλοι οριζόμενοι για χρήστες / ομάδες χρηστών

10.6.2.1 Διαχείριση

- Απόδοση ή αφαίρεση ρόλων σε μία ομάδα χρηστών, από τους ρόλους που έχουν οριστεί στη συγκεκριμένη εφαρμογή (Υ)
- Διαχείριση της πρόσβασης στο περιεχόμενο με βάση το σχήμα Owner / Group/ Others του Unix (Π)

10.6.3 Στοιχεία περιεχομένου (πόροι)

10.6.3.1 Συλλογή

- Δυνατότητα σύνδεσης με ένα OPML[(5)] directory για αυτόματη ανάγνωση και εισαγωγή XML Syndications (Π)
- Παροχή ενός περιβάλλοντος εξειδικευμένου editor (είτε word editor είτε spreadsheet) με την δυνατότητα εισαγωγής style sheets ή εφαρμογής κάποιου template (όπως σε ένα .dot αρχείο στο Word) με ενεργά πεδία, μακροεντολές και συνδέσμους σε εξωτερικές λίστες και αρχεία με metadata (Π)
- Δυνατότητα ψηφιοποίησης εγγραφών και άλλων hardcopy εκδόσεων πόρων περιεχομένου, με διασύνδεση interfaces σχετικών συσκευών (scanners, videocards, microphones) (Π)
- Δυνατότητα να υπάρχει διαθέσιμο ένα καλά τεκμηριωμένο API με μεθόδους για εισαγωγή και αποθήκευση εξωτερικών δεδομένων και ενσωμάτωση (integration) του CMS με εξωτερικές εφαρμογές (Π)

10.6.3.2 Διαχείριση

- Δημιουργία συνδέσμου προς δικτυακούς τόπους. Ο συγγραφέας ενός συνδέσμου πρέπει να μπορεί να καθορίσει τουλάχιστον ένα τίτλο, μια περιγραφή και το σχετικό σύνδεσμο (URL) (Υ)
- Σύστημα scheduling για περιοδική αυτόματη εκτέλεση κάποιων εσωτερικών triggers (π.χ. ανανέωση ή διαγραφή περιεχομένου με βάση συγκεκριμένα κριτήρια) (Π)
- Ύπαρξη συστήματος caching για ταχύτερη φόρτωση του περιεχομένου των σελίδων σε ένα web site, επιταχύνοντας την διαδικασία αναζήτησης στο repository (Π)
- Χρήση έτοιμων templates για δημιουργία νέων πόρων, μέσα από web browser based περιβάλλον (Υ)
- Εύκολη διαχείριση πολυμεσικού περιεχομένου (εικόνες, video, αρχεία ήχου) με χρήση media gallery για γρήγορη ομαδοποίηση / αντίστοιχα για έγγραφα ένα document gallery (Π)
- Υποστήριξη μακροεντολών για αυτόματη εκτέλεση μιας σειράς ενεργειών (εκτέλεση script σε κάποια γνωστή ή custom γλώσσα scripting) (Π)
- Υποστήριξη clustering (για κατανεμημένα περιβάλλοντα) και imaging του περιεχομένου και των εφαρμογών (Π)

10.6.3.3 Δημοσίευση

- Δυνατότητα αποστολής επιλεγμένου πληροφοριακού υλικού της εφαρμογής, με λειτουργία της ίδιας της εφαρμογής (π.χ. αποστολή ενός άρθρου) (Υ)
- Επικοινωνία με εξωτερικές Web Services και άλλες εφαρμογές μέσω XMLRPC, SOAP, HTTP (Π)
- Εξαγωγή σε data warehouse για data mining περιεχομένου (Π)
- Δυνατότητα σύνδεσης με εξωτερική φορητή συσκευή και αποστολή περιεχομένου σε XML format. Χρήση ασύρματων πρωτοκόλλων μεταφοράς όπως το SMS (Short Messaging System)και το WAP (Wireless Application Protocol) με την WML (Wireless Markup Language) (Π)

10.6.4 Δομές οργάνωσης περιεχομένου (κόμβοι)

10.6.4.1 Διαχείριση

Αναζήτηση κόμβων περιεχομένου με βάση το όνομα ή metadata (Υ)

10.6.4.2 Δημοσίευση

Προβολή της πλήρους διαδρομής στην λογική ιεραρχία των κόμβων που βρισκόμαστε ανά πάσα στιγμή (Υ)

10.6.5 Δομές παρουσίασης περιεχομένου (π.χ. σελίδες, φόρμες)

10.6.5.1 Διαχείριση

Δημιουργία και ενεργοποίηση / απενεργοποίηση νέας φόρμας αποστολής στοιχείων από τους χρήστες (Υ)

10.6.5.2 Δημοσίευση

Σύστημα online help με αναδυόμενες φόρμες και μενού σχετικά με την λειτουργία κάθε σελίδας (Υ)

10.6.6 Δομές εισαγωγής περιεχομένου (π.χ. φόρμες)

10.6.6.1 Συλλογή

- Ενσωματωμένος news aggregator (Π)
- Φόρμα αναζήτησης στο web και εισαγωγής των αποτελεσμάτων από τις δημοφιλέστερες μηχανές (Π)
- Φόρμα εισαγωγής ιδιοτήτων (properties) για τους πόρους περιεχομένου, με στόχο την παραμετροποίηση της εμφάνισης του portal (Υ)
- Δυνατότητα εισαγωγής ερώτησης σχετικά με την λειτουργία του συστήματος και το πώς διεκπεραιώνεται μια εργασία σε φυσική γλώσσα ή με λέξεις κλειδιά (Π)
- Δυνατότητα αλλαγής της εμφάνισης του περιβάλλοντος της εφαρμογής (αλλαγή theme) με κατάλληλη επιλογή του χρήστη μέσω φόρμας (Π)
- Πρόληψη για άτομα με ειδικές ανάγκες (Accessibility features) / Χρήση εναλλακτικών μορφών και μεθόδων εισαγωγής περιεχομένου (Π)

10.6.7 Ροές διαχείρισης (workflows)

10.6.7.1 Συλλογή

Σύστημα έγκρισης από τους διαχειριστές (approval) για εισαγωγή νέου περιεχομένου στο repository

10.6.7.2 Διαχείριση

- Καθορισμός εάν ένα στοιχείο περιεχομένου (σύνδεσμος, κείμενο, έγγραφο κοκ) είναι δημοσιεύσιμο ή βρίσκεται ακόμη σε κατάσταση συγγραφής (Υ)
- Αυτόματη καταγραφή του συντάκτη, της ημερομηνίας δημοσίευσης ενός στοιχείου περιεχομένου (Υ)
- Δυνατότητα όλων των χρηστών authors (δημιουργοί περιεχομένου) για λειτουργία remove edit permissions. Δηλαδή κάθε συγγραφέας μπορεί να απενεργοποιεί προσωρινά την δυνατότητα που έχουν άλλοι χρήστες να επεξεργάζονται ένα πόρο, όταν αυτός δουλεύει σε αυτόν τον πόρο, δηλαδή κλείδωμα του πόρου για consistency (Υ)
- Δυνατότητα αποστολής εσωτερικά μηνυμάτων μεταξύ των εγγεγραμμένων χρηστών (Π)

10.6.8 Αντικείμενα προβολής (π.χ. banners)

10.6.8.1 Συλλογή

Εισαγωγή νέων εξωτερικών banners και δυνατότητα σχεδιασμού νέων (Υ)

10.6.8.2 Δημοσίευση

Δυνατότητα αυτόματης εναλλαγής των banners με βάση κάποιο συμβόλαιο χρονοχρέωσης. Εφαρμογή μεθόδων marketing και διαφήμισης μιας εταιρείας (Π)

10.6.9 Οντότητες για κοινότητες (communities entities)

10.6.9.1Συλλογή

- Συστήματα δημοσκοπήσεων / σφυγμομετρήσεων (polls)
- Δυνατότητα download manager και παροχής online δίσκου για αποθήκευση τοπικών και downloaded εγγράφων (Π)
- Δυνατότητα για voice browsing (φωνητικές εντολές). Γενικότερα μέριμνα για accessibility features (Π)

10.6.9.2 Διαχείριση

- Δημιουργία νέου audience που διαθέτει ένα μοναδικό audience ID, name, έναν τουλάχιστο στόχο (goal) που πρέπει να ακολουθεί καθώς και πιο σύνθετες δομές για την αποθήκευση στοιχείων που σχετίζονται με το προφίλ του συγκεκριμένου audience, δηλαδή στατιστικά στοιχεία σχετικά με τις τάσεις, δημογραφία, επιθυμίες, νοοτροπία των μελών του audience αυτού. Συνεπώς όταν ορίζεται μέσα στο CMS ένας χρήστης που ανήκει στο συγκεκριμένο audience, αυτόματα κληρονομεί όλες τις ιδιότητες του audience αυτού. (Υ)

- Ενεργοποίηση / απενεργοποίηση audience ή διαγραφή του από το σύστημα (Υ)
- Δυνατότητα προσθήκης plug-in και ρύθμισης εξωτερικών προγραμμάτων και εφαρμογών ώστε να συνεργάζονται με τις σελίδες και το περιεχόμενο της online κοινότητας (Π)

10.6.9.3 Δημοσίευση

- Πίνακας ανακοινώσεων με ελεγχόμενη πρόσβαση (για συγγραφή ή ανάγνωση) (Υ)
- Προβολή σελίδας με στατιστικά στοιχεία σχετικά με την κίνηση και το προφίλ του χρήστη (Υ)
- Φόρμα για πρόσβαση μέσω web στους web-based POP3 email λογαριασμούς (Π)
- Παρουσίαση, σε κάθε σελίδα, μιας λίστας με μερικούς προτεινόμενους σχετικούς με το περιεχόμενο της σελίδας εξωτερικούς δεσμούς Web (URL). Η λειτουργία αυτή μπορεί να βασίζεται και στην συλλογή προσωπικών στοιχείων προφίλ του χρήστη (Π)
- Δυνατότητα για real time chatting (με κάποια Java-based εφαρμογή) με άλλους χρήστες της κοινότητας που βρίσκονται online εκείνη τη στιγμή (Π)

10.7 Ποια είναι τα δημοφιλέστερα CMS ανοιχτού κώδικα;

- **Joomla**

- **Drupal**

- **Wordpress**

10.8 Joomla vs Wordpress vs Drupal

Joomla!® Το Joomla έχει πολύ πιο πλούσια χαρακτηριστικά και από τη κατασκευή του. Προτείνεται για πιο σύνθετα site και μπορεί να καλύψεις ακόμα και τις πιο δύσκολες απαιτήσεις. Προσφέρεται για την δημιουργία portal και απευθύνεται συνήθως σε πιο έμπειρους χρήστες που απαιτούν μία πιο ιδιαίτερη σχεδίαση στην ιστοσελίδα τους και αναζητούν κάτι περισσότερο από ένα απλό ιστολόγιο. Κυκλοφορούν χιλιάδες modules, plugins αλλά και templates τα οποία μπορούν να εγκατασταθούν αρκετά εύκολα και γρήγορα, να το μετατρέψουν σε ένα πολύ δυνατό σύστημα διαχείρισης και παρουσίασης του περιεχομένου και να το μεταμορφώσουν από blog μέχρι eshop και από forum και newsportal μέχρι videoblog ή photogallery . Προσφέρει δυναμική διαχείριση του πρωτοσέλιδου, του περιεχομένου καθώς και της διαμόρφωσης των πλευρικών στηλών με τα μενού και τα modules, υστερεί όμως έναντι του Wordpress στο ότι δεν έχει κάποιο ενσωματωμένο σύστημα για σχολιασμό των άρθρων (comenting system) και απαιτείται για αυτό κάποιο πρόσθετο plugin . Κρίνεται απαραίτητα μια στοιχειώδης εκπαίδευση στους αρθογράφους για τον σωστό τρόπο δημοσίευσης των άρθρων τους καθώς και για τα χρησιμοποίηση τυχόν πρόσθετων ενθεμάτων. Το Joomla έχει και αυτό την δυνατότητα να είναι φιλικό με τις μηχανές αναζήτησης , Search Engine Friendly, με ενεργοποίηση του από τον πίνακα διαχείρισης ή με την προσθήκη κάποιου plugin όπως το OpenSEF.

WordPress Το Wordpress από την κατασκευή του είναι μια καθαρή blogging πλατφόρμα , με πολύ καλό σύστημα διαχείρισης των σχολίων (comments) με την οποία μπορείτε να ξεκινήσετε τη δημοσίευση και ανάρτηση των άρθρων σας στο ιστολόγιό σας (blog). Τα άρθρα έχουν χρονολογική ταξινόμηση. Απευθύνεται συνήθως σε αρχάριους, οι οποίοι δεν έχουν καμία γνώση σχεδιασμού και κατασκευής ιστοσελίδων καθώς και HTML, CSS, PHP και MYSQL, αλλά τους ενδιαφέρει μόνο η διαχείριση του περιεχομένου, δηλαδή η ανάρτηση άρθρων. Από την αρχική εγκατάσταση του δεν διαθέτει κάποιο menu πλοήγησης, αν και αυτό λύνετε με κάποιο άλλο template ή plugin. Το Wordpress μπορεί να επεκταθεί με τη χρήση plugin και την διαμόρφωσή του με templates τα οποία εύκολα μπορείτε να κατεβάσετε από το Wordpress.org. Είναι από την κατασκευή του φιλικό προς τις μηχανές αναζήτησης και υπάρχουν επίσης διάφορα plugins που το κάνουν ακόμα πιο SEO friendly.

Είναι προφανές ότι το Joomla πλεονεκτεί σε πολλά σημεία το ανταγωνιστικό του CMS Wordpress και ιδιαίτερα όταν πρόκειται για σύνθετους σχεδιασμούς ιστοσελίδων και σχετικά πεπειραμένους χρήστες, η εγκατάσταση ενός Joomla κρίνετε επιβεβλημένη. Από την άλλη μεριά το Wordpress κρίνεται απαραίτητο να εγκατασταθεί, όταν οι απαιτήσεις στον σχεδιασμό της ιστοσελίδας είναι περιορισμένες και ο χρήστης άπειρος, κλασική εφαρμογή του η δημιουργία ιστολογίου για ανεξάντλητο σχολιασμό.

Το Drupal κέρδισε το βραβείο του καλύτερου CMS για τρίτη χρονιά το 2009 και υποστηρίζεται από μια από τις ισχυρότερες κοινότητες προγραμματιστών ανοιχτού κώδικα στον κόσμο. Το Drupal είναι κατάλληλο για σύνθετες εγκαταστάσεις διαχείρισης περιεχομένου και κάθετες εφαρμογές. Η εξατομικευμένη φύση των θεματικών παραλλαγών του και η σύνθετη και ευέλικτη αρχιτεκτονική του απαιτεί εξειδικευμένους σχεδιαστές και προγραμματιστές. Το Drupal, μετά από επαγγελματική προσαρμογή, παραμένει το πιο εργονομικό και εύχρηστο σύστημα για τον τελικό χρήστη. Σημαντικό είναι να αναφερθεί ότι πλεονεκτεί του Joomla στο οτι μπορεί να εκτελεστεί σε διάφορες πλατφόρμες, συμπεριλαμβανομένων των λειτουργικών συστημάτων Windows, Mac OS X, Linux, FreeBSD, ή οποιασδήποτε πλατφόρμας που υποστηρίζει είτε το διακομιστή ιστοσελίδων Apache HTTP Server (έκδοση 1.3+), είτε το Internet Information Services (έκδοση IIS5+), καθώς επίσης και τη γλώσσα προγραμματισμού PHP (έκδοση 4.3.3+). Το Drupal απαιτεί μια βάση δεδομένων όπως η MySQL και η PostgreSQL για την αποθήκευση του περιεχομένου και των ρυθμίσεών του.

Με βάση όλα τα παραπάνω επιλέξαμε να χρησιμοποιήσουμε το joomla για την δημιουργία της ιστοσελίδας της εφαρμογής.

10.9 Γνωριμία με το CMS Joomla

10.9.1 Ιστορία του Joomla

Το Joomla είναι μια δωρεάν εφαρμογή ελευθέρου λογισμικού για την δημιουργία δυναμικών ιστοσελίδων. Το ελεύθερο λογισμικό όπως ορίζεται από το Ίδρυμα Ελευθέρου Λογισμικού (Free Software Foundation) είναι λογισμικό που μπορεί να χρησιμοποιηθεί, αντιγραφεί, μελετηθεί, τροποποιηθεί και αναδιανεμηθεί χωρίς περιορισμό.

Το Joomla είναι σχεδιασμένο στη γλώσσα προγραμματισμού PHP και τα δεδομένα αποθηκεύονται σε βάση δεδομένων MySQL. Μπορεί να χρησιμοποιηθεί για ερασιτεχνικές και προσωπικές ιστοσελίδες καθώς και για επαγγελματικές.

Το όνομα "Joomla" είναι μια φωνητική γραφή της γλώσσας Σουαχίλι (Swahili) στην οποία η λέξη "jumla" σημαίνει "όλοι μαζί" ή "ως σύνολο". Επέλεξαν αυτό το όνομα για να αντικατοπτρίζει τη δέσμευση της Κοινότητας και την ομάδα ανάπτυξης του έργου. Η πρώτη έκδοση του Joomla (Joomla 1.0.0) ανακοινώθηκε στις 16 Σεπτεμβρίου 2005. Αυτή ήταν μια νέα ονομασία της έκδοσης Mambo 4.5.2.3 σε συνδυασμό με διορθώσεις κάποιων σφαλμάτων (bug) στο περιβάλλον διαχείρισης και στον κώδικα ασφαλείας της.

10.9.2 Χαρακτηριστικά του Joomla

Τα βασικότερα χαρακτηριστικά του Joomla είναι τα εξής:

1. Είναι ΣΔΠ ανοιχτού κώδικα.
2. Διαθέτει μεγάλη κοινότητα χρηστών παγκοσμίως στο www.joomla.org, αλλά και στην Ελλάδα στο www.joomla.gr.
3. Παρέχει μεγάλη ευελιξία στη δημοσίευση περιεχομένου.
4. Διαθέτει διαχειριστή αρχείων (Media Manager) για μεταφόρτωση και διαχείριση αρχείων και πολυμέσων.
5. Είναι αρκετά εύκολο στη χρήση του ακόμη και από αρχάριους χρήστες Η/Υ.
6. Παρέχει την υπηρεσία ενημέρωσης γεγονότων (RSS).
7. Διαθέτει κάδο ανακύκλωσης για τα αντικείμενα περιεχομένου προς διαγραφή.
8. Παρέχει ειδικό μηχανισμό για τις μηχανές αναζήτησης (SEO – Search Engine Optimization).
9. Παρέχει τη δυνατότητα διαχείρισης διαφημίσεων.
10. Είναι πολυγλωσσικό cms.
11. Μπορούν να ενσωματωθούν δεκάδες πρόσθετες εφαρμογές, των οποίων η εγκατάσταση είναι αρκετά εύκολη.
12. Υπάρχουν πολλά επίπεδα χρηστών.
13. Διαθέτει στατιστικά επισκεψιμότητας του κόμβου.
14. Είναι ένας πλήρης μηχανισμός διαχείρισης της βάσης δεδομένων του site.
15. Έχει δυναμικό Forum / Poll / Voting για άμεσα επί τόπου αποτελέσματα.
16. Τρέχει σε Linux, FreeBSD, MacOSX server, Solaris και AIX.

10.9.3 Εκτεταμένη Διαχείριση και Δυνατότητες:

1. Χρησιμοποιεί Module για απομακρυσμένη υποβολή από τον συγγραφέα για Νέα, άρθρα, FAQs και Links.
2. Δημοσιεύει απεριόριστες σελίδες και άρθρα χωρίς κανέναν απολύτως περιορισμό.
3. Υπάρχει η δυνατότητα προσθήκης photo galleries, βιβλιοθήκες αρχείων, βιβλία επισκεπτών και φόρμες επικοινωνίας
4. Εύκολη διαχείριση online των PNGs, PDFs, DOCs, XLSs, GIFs και JPEGs με τηβοήθεια του Image library.
5. Παρέχει Αυτόματο Path-Finder.
6. News feed manager. Επιλέγει πάνω από 360 news feeds από όλο τον κόσμο.
7. Archive manager. Υπάρχει η δυνατότητα να μπουν τα παλαιά άρθρα στην "κατάψυξη" αντί να τα διαγραφούν εντελώς.
8. Email-a-friend και Print-format για κάθε άρθρο.
9. Ενσωματωμένος επεξεργαστής κειμένου αντίστοιχος του Word Pad.
10. Εμφάνιση και αισθητική την οποία διαμορφώνει ο χρήστης.
11. Διαχείριση των Template (πρότυπα)
12. Δυνατότητα προεπισκόπησης καθώς είναι εφικτή η προβολή αυτών των τμημάτων της ιστοσελίδας που έχουν δημιουργηθεί πριν παρουσιαστούν online.
13. Banner manager για διαφημιστική προβολή.

14. Προσαρμογή του σχεδιασμού των templates στις επιθυμίες του πελάτη, προσθήκη γραφικών, των λογοτύπων και των σλόγκαν.
15. Εύκολη διαχείριση και διαμόρφωση του πρωτοσέλιδου με αναδιάταξη των άρθρων.
16. Ενεργοποίηση των feeds RSS 2.0 και Atom (τροφοδοσίες).
17. Δυνατότητα κατασκευής πολυγλωσσικήςιστοσελίδας (Multilanguage)
18. Επέκτασή του σε ηλεκτρονικό κατάστημα (eshop)
19. Δυνατότητα λήψης αντιγράφου ασφαλείας του site (back up)
20. Δυνατότητα βελτιστοποίησης της ιστοσελίδας Joomla στις μηχανές αναζήτησης (SEO) διότι το Joomla είναι ένα Search Engine Friendly CMS (SEF) .

Η κατασκευή ενός Joomla site μπορεί να καλύψει τις ανάγκες του απλού ιδιώτη ήμικρού επαγγελματία, μέχρι και τις ανάγκες ενός συλλόγου, μιας μεγάλης επιχείρησης, ενός Φορέα ή Οργανισμού. Παρέχει ένα ασφαλές και ευέλικτο περιβάλλον εργασίας για την δυναμική διαχείριση του περιεχομένου του site (συχνές και συνεχείς προσθήκες και αλλαγές στο περιεχόμενο) , εύκολα και γρήγορα με μία μικρή στοιχειώδη βασική εκπαίδευση.

10.9.4 Η δομή του Joomla

10.9.4.1 Δημόσιο Τμήμα (Front End)

Το Δημόσιο Τμήμα είναι στην ουσία αυτό που βλέπει ο τελικός χρήστης, δηλαδή η ιστοσελίδα μας. Μέσα στο δημόσιο τμήμα, λοιπόν, βρίσκονται τα άρθρα, τα μενού και γενικά όλα τα στοιχεία που θέλουμε να εμφανίζονται στην ιστοσελίδα.

10.9.4.2 Περιοχή Διαχείρισης (Back End)

Η περιοχή διαχείρισης είναι το «εργαστήριο» του Joomla. Μέσα από την περιοχή διαχείρισης ο Διαχειριστής (Administrator) μπορεί να προσθέτει περιεχόμενο, να εμφανίζει ή να αποκρύπτει στοιχεία, να δημιουργεί χρήστες και γενικά να εκμεταλλεύεται όλες τις δυνατότητες του συστήματος.

10.9.4.3 Μενού (Menu)

Τα μενού στο Joomla είναι τα αντικείμενα εκείνα που βοηθάνε τον χρήστη στην πλοήγηση μέσα στον ιστότοπό μας. Μέσα από τα μενού θα συνδέσουμε τα αντικείμενα περιεχομένου μας ώστε να καταστεί δυνατή η πρόσβαση σε αυτά. Τα μενού μπορούν να είναι οριζόντια ή κατακόρυφα και μπορούμε να έχουμε όσα μενού θέλουμε καθώς και να τα τοποθετούμε μέσα στην ισοσελίδα μας όπου θέλουμε. Δημιουργούνται δυναμικά και συνδέονται με αντικείμενα του Joomla όπως, άρθρα, κατηγορίες και ενότητες.

10.9.4.4 Επεκτάσεις (Extensions)

Οι επεκτάσεις στο Joomla είναι από τα βασικότερα στοιχεία. Με τη βοήθεια των επεκτάσεων βοηθάμε τον ιστότοπο μας να επεκτείνεται με νέες εφαρμογές και λειτουργίες. Οι επεκτάσεις του Joomla χωρίζονται σε πέντε κατηγορίες οι οποιές είναι οι παρακάτω:

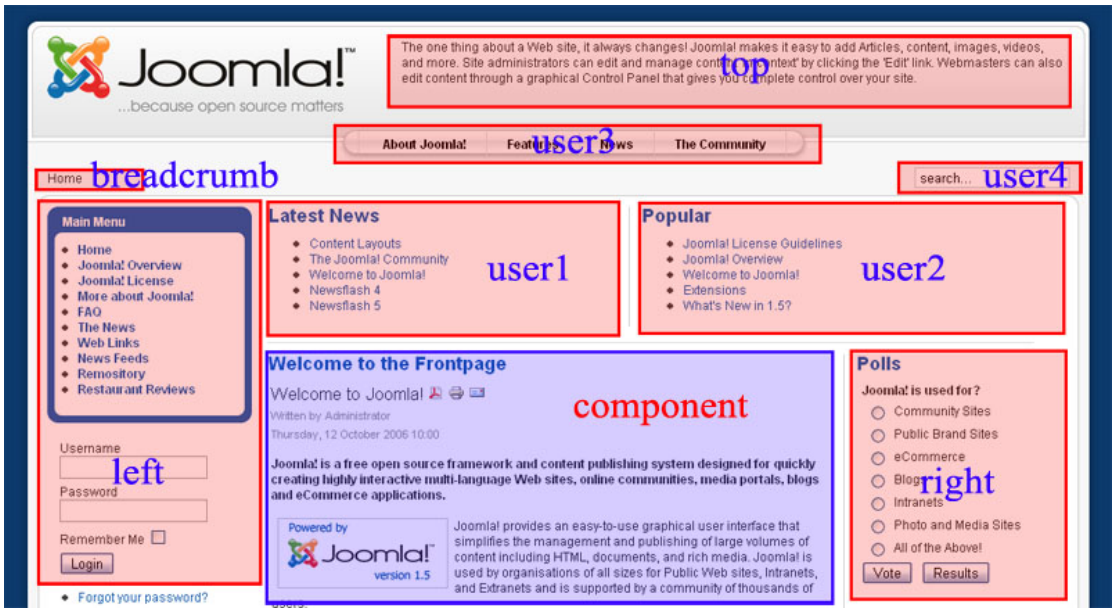
- **Εφαρμογές (Components):** Οι εφαρμογές χρησιμοποιούνται για να μπορεί το Joomla να επεκτείνεται και τρέχουν μέσα στο Joomla. Άλλες από αυτές τις εφαρμογές είναι εμπορικές και άλλες ελεύθερης διανομής. Μερικές από αυτές είναι εφαρμογές για e-shop (π.χ. σύστημα καλαθιού), για gallery φωτογραφιών, για e-learning.
- **Ενθέματα (Modules):** Τα ενθέματα είναι τα "κουτιά" μέσα στα οποία εμφανίζεται το περιεχόμενο, οι εφαρμογές, τα πρόσθετα και γενικά όλα τα αντικείμενα που εμφανίζονται στο δημόσιο τμήμα. Η θέση τους στον ιστότοπο καθορίζεται από το αρχείο index.php του επιλεγμένου template ενώ το στυλ της εμφάνισής τους από το αρχείο CSS του template. Είναι λοιπόν μια μικροεφαρμογή η οποία τρέχει σε κάποια θέση της ταμπλέτας. Για παράδειγμα το main menu στην αρχική σελίδα ενός ιστότοπου, είναι ένα module. Κάθε ένθεμα πρέπει να έχει μοναδικό όνομα ώστε να μην μπερδεύεται με τα άλλα. Τα ενθέματα μπορούν να περιέχουν μενού, διαφημίσεις, ψηφοφορίες, άλλες εφαρμογές ή περιεχόμενο της επιλογής μας. Μπορούμε να δημιουργήσουμε αντίγραφα ενθεμάτων και να τα τοποθετήσουμε σε διαφορετικά σημεία στον ιστότοπό μας. Επιπλέον μέσα από τη διαχείριση και τις παραμέτρους μπορούμε να ορίσουμε πότε και σε ποιους θα εμφανίζονται.
- **Πρόσθετα (Plug-Ins):** Τα πρόσθετα είναι κομμάτια κώδικα τα οποία εκτελούν κάποιες ειδικές λειτουργίες. Για παράδειγμα μία μηχανή αναζήτησης, που είναι στο Joomla και εμφανίζεται στην ιστοσελίδα μας και που μπορεί ο χρήστης να αναζητεί περιεχόμενο μέσα από αυτήν, είναι ένα πρόσθετο. Τα πρόσθετα, λοιπόν, είναι κάποιες εφαρμογές που είναι ενσωματωμένες στο Joomla μπορούμε όμως να ενεργοποιήσουμε ή να απενεργοποιήσουμε κάποια από αυτά οποιαδήποτε χρονική στιγμή με πολύ βέβαια προσοχή για να μην υπάρξει πρόβλημα στην ομαλή λειτουργία του ιστότοπου. Είναι λοιπόν όπως λέει και το όνομα του ένα πρόσθετο το οποίο τρέχει παράλληλα με το Joomla και του προσδίδει κάποιες επιπλέον δυνατότητες. (Μερικά components για να λειτουργήσουν απαιτούν την ύπαρξη και του αντίστοιχου plugin). Στο Joomla υπάρχουν έξι διαφορετικοί τύποι προσθέτων οι οποίοι είναι οι εξής:
  - **Authentication Plug-ins:** Τα πρόσθετα αυτά είναι για την επικύρωση, την εγγραφή και τη σύνδεση των χρηστών.
  - **Content Plug-ins:** Τα πόσθετα αυτά χρησιμοποιούνται για τις διάφορες λειτουργίες του περιεχομένου όπως η εμφάνιση εικόνων.
  - **Editors Plug-ins:** Τα πρόσθετα αυτά χρησιμοποιούνται για τις λειτουργίες των κειμενογράφων του Joomla.
  - **Search Plug-ins:** Τα πρόσθετα αυτά ελέγχουν τη συνάρτηση της αναζήτησης στο ένθεμα Search. Με αυτόν τον τρόπο μπορούμε να αναζητούμε περιεχόμενο στις παραπάνω περιπτώσεις.
  - **System Plug-ins :** Τα πρόσθετα αυτά χρησιμεύουν για την λειτουργία του Joomla, όπως είναι η πρόσβαση στα log files που κρατάνε πληροφορίες για τον

server, η ενεργοποίηση της συνάρτησης αποσφαλμάτωσης (debugging) του Joomla, η ενεργοποίηση της λειτουργίας της προσωρινής αποθήκευσης δεδομένων και η αποθήκευση των στοιχείων ενός χρήστη ώστε να απομνημονεύεται από το σύστημα.

- **User-Joomla:** Το πρόσθετο αυτό δημιουργεί τους χρήστες στην βάση δεδομένων.

10.9.4.5 Πρότυπα (Templates):

Τα πρότυπα χρησιμεύουν για να διαχωριστεί το περιεχόμενο από την εμφάνιση. Στα πρότυπα ορίζονται τα χρώματα, η θέση των ενθεμάτων και γενικά όλη η σχεδίαση του ιστότοπου μας. Τα πρότυπα είναι ουσιαστικά ο τρόπος με τον οποίο θα εμφανίζεται το δικό μας Joomla. Κανείς δε θέλει μία ιστοσελίδα η οποία θα μοιάζει με κάποια άλλη και πολύ περισσότερο με κάποιες άλλες. Για αυτό το λόγο είτε θα αλλάξουμε τον τρόπο εμφάνισης μόνοι μας, είτε θα επιλέξουμε κάποιο πρότυπο που έχει κατασκευάσει κάποιος άλλος.



Εικόνα : Joomla template με τις εξ ορισμού θέσεις του

10.9.5 Εγκατάσταση του Joomla

Για να εγκαταστήσουμε το joomla χρειαζόμαστε έναν Apache server ο οποίος να υποστηρίζει php και MySQL. Άρα για να κάνουμε την εγκατάσταση θα πρέπει να έχουν στον υπολογιστή μας κάποιο πρόγραμμα όπως το Xampp ή το Wamp, και τα δυο σηκώνουν server με ότι χρειαζόμαστε για το jooml, ή ακόμα καλύτερα να βρούμε στο δίκτυο έναν web hosting server που θα παρέχει ότι χρειαζόμαστε δωρεάν. Έτσι το site θα έχει και δικό του domain και θα είναι online συνέχεια.

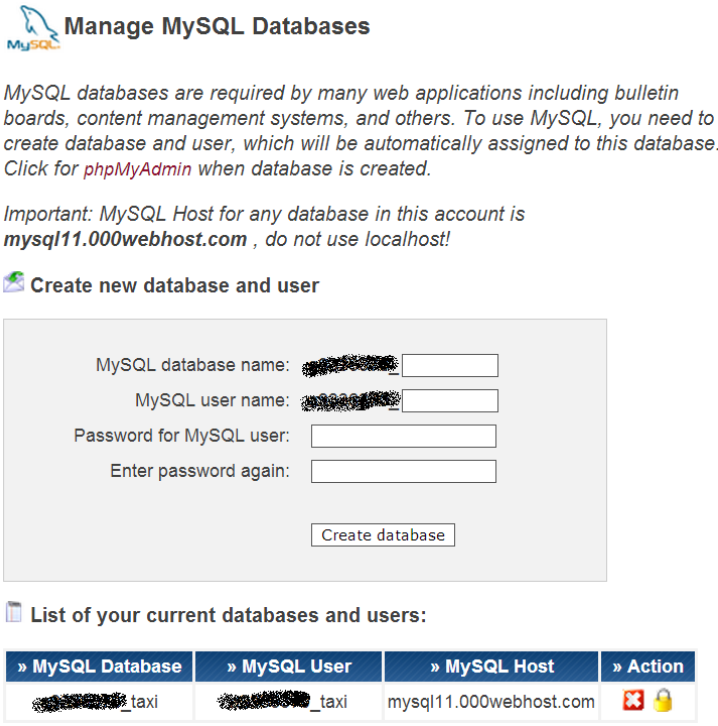
Εμείς επιλέξαμε την δεύτερη λύση. Συγκεκριμένα το site http://www.000webhost.com

Αφού λοιπόν κάνουμε έναν λογαριασμό στο site και δώσουμε το domain της ιστοσελίδας μας, θα πρέπει να φτιάξουμε μια βάση δεδομένων τν οποία θα χρησιμοποιεί το joomla όπως φαίνεται παρακάτω.

Αρχικά πηγαίνουμε στο CPanel του site μας και επιλέγουμε το MySQL

Εικόνα: Κομμάτι από το CPanel.

Στην συνέχεια συμπληρώνουμε τα στοιχεία που χρειάζεται για την δημιουργία της βάσης.

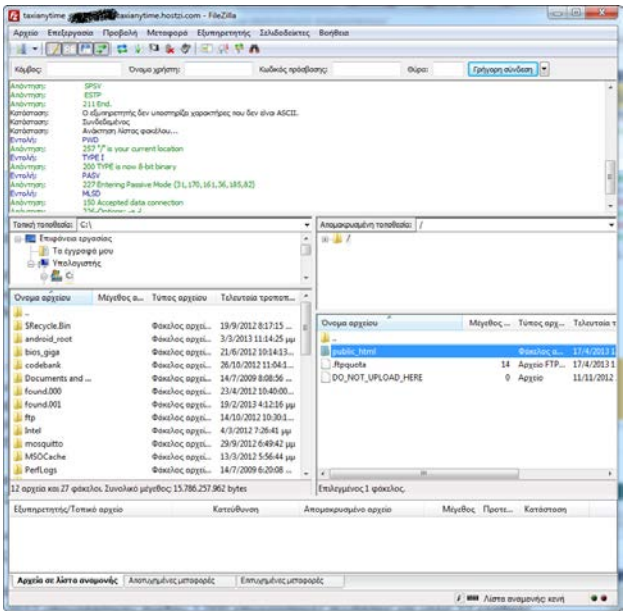


Εικόνα: Δημιουργία βάσης δεδομένων

Το τελευταίο που χρειαζόμαστε είναι το… joomla! Μπορούμε να το κατεβάσουμε από την επίσημη ιστοσελίδα τους http://www.joomla.org/download.html

Όταν κάναμε εγγραφή στο http://www.000webhost.com μας έδωσε έναν ftp server. Εκεί έχουμε πρόσβαση με τα στοιχεία του λογαριασμού μας. Τώρα χρειαζόμαστε έναν ftp client για να τον διαχειριζόμαστε. Επιλέξαμε το πολύ γνωστό filezilla οπου είναι διαθέσιμο δωρεάν στην επίσημη ιστοσελίδα τους https://filezilla-project.org/download.php

Για να ξεκινήσουμε την εγκατάσταση θα πρέπει πρώτα να μεταφέρουμε το joomla στον ftp μας. Ανοίγουμε το filezilla και συνδεόμαστε με τα παραπάνω στοιχεία στον ftp.

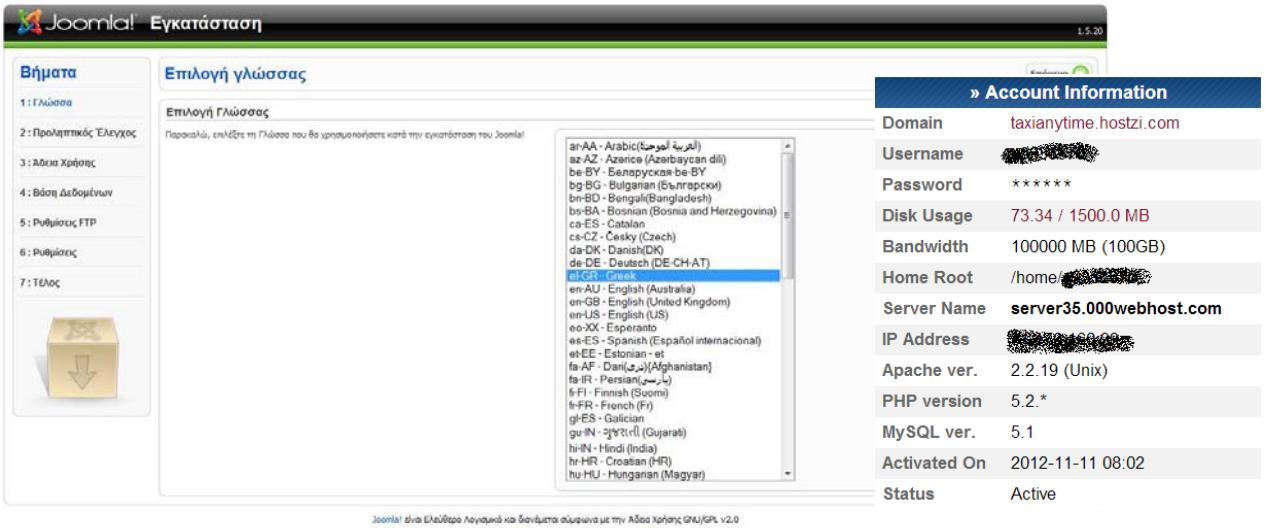


Εικόνα: ftp client (filezilla)

Τώρα μπορούμε να μεταφέρουμε τον φάκελο του joomla που κατεβάσαμε, στον φάκελο public_html. Η διαδικασία αυτή θα πάρει αρκετά λεπτά.

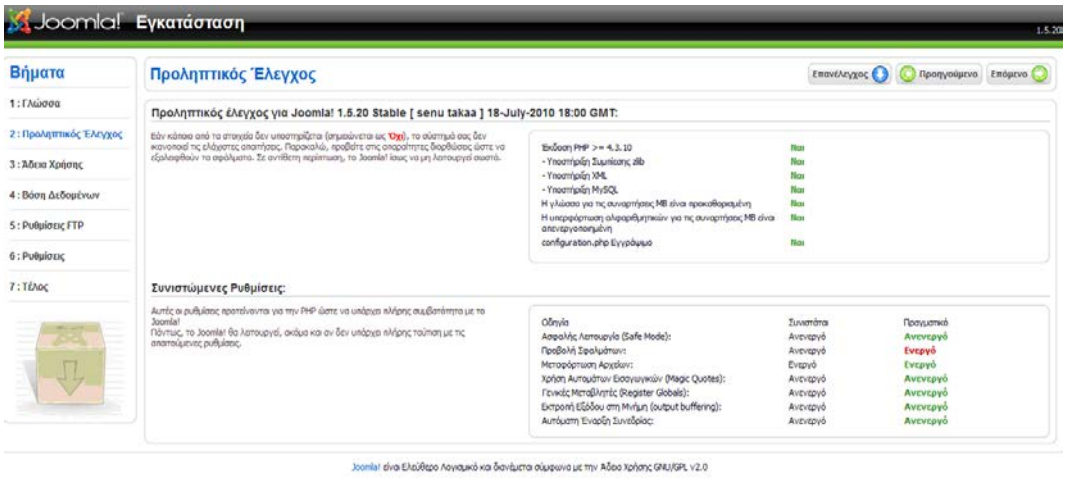
Όταν ολοκληρωθεί η μεταφορά ανοίγουμε έναν browser και πληκτρολογούμε το link του site μας. Στην συγκεκριμένη περίπτωση είναι το www.taxianytime.hostzi.com . Έτσι μεταβαίνουμε στην εγκατάσταση του joomla.

Αρχικά επιλέγουμε γλώσσα.



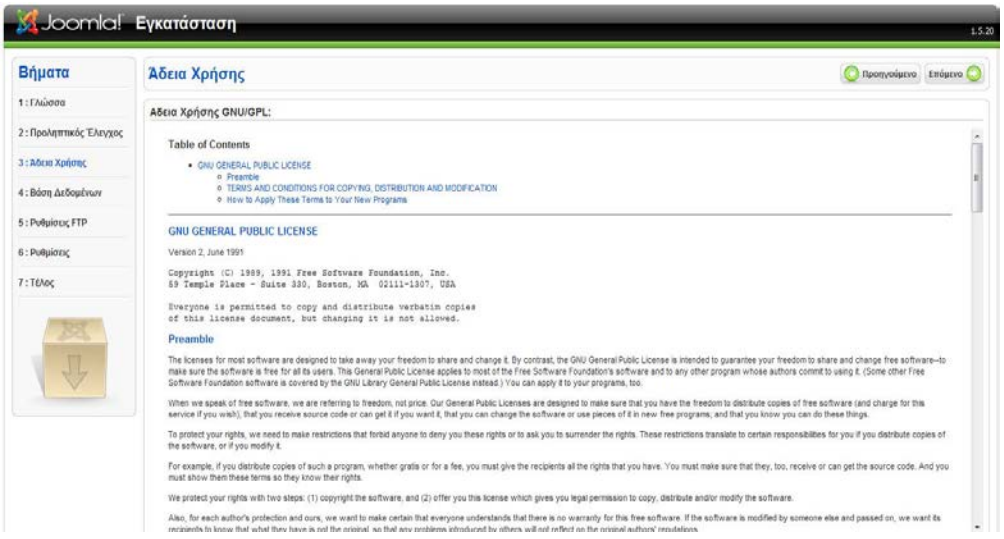
Εικόνα : 1ο βήμα εγκατάστασης Joomla, επιλογή γλώσσας.

Στην συνέχεια το Joomla ελέγχει κάποια στοιχεία στον υπολογιστή μας και στο server πριν γίνει η εγκατάσταση. Για να μπορέσουμε να προχωρήσουμε σωστά θα πρέπει τα στοιχεία στο πάνω μέρος της σελίδας να έχουν τη σήμανση «Ναι» και να είναι πράσινα, ενώ στο κάτω μέρος της σελίδας είναι κάποια στοιχεία που προτείνονται.



Εικόνα : 2ο βήμα εγκαταστασης joomla προληπτικός έλεγχος.

Στην επόμενη οθόνη εμφανίζεται η **Άδεια Χρήσης GNU/GPL** που χρησιμοποιεί το **Joomla**. Κάνουμε κλικ στο κουμπί **Επόμενο.**

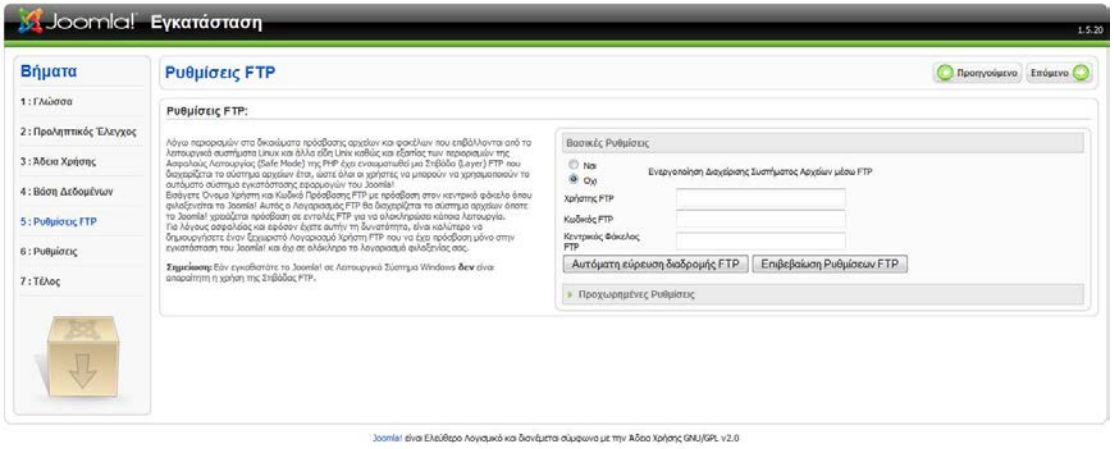


Εικόνα: 3ο βήμα εγκατάστασης ,άδεια χρήσης.

Είμαστε στο πιο σημαντικό βήμα. Εδώ πρέπει να εισαγάγουμε τα στοιχεία της βάσης δεδομένων με την οποία θα συνεργάζεται το **Joomla**. Εμφανίζεται η παρακάτω οθόνη. Στην περιοχή **Βασικές Ρυθμίσεις** εισάγουμε στα πλαίσια κειμένου τα αντίστοιχα δεδομένα.

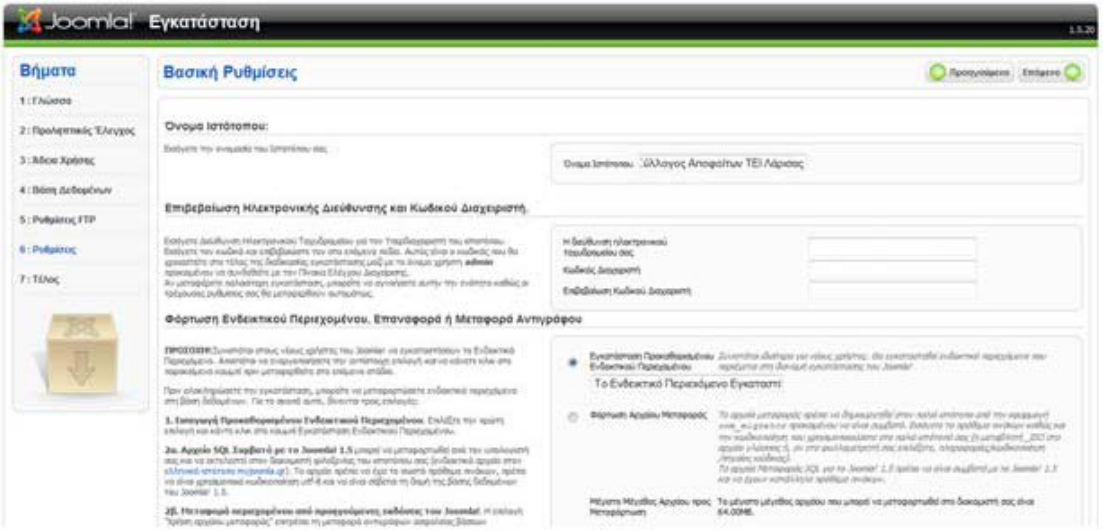
Εικόνα : 4ο βήμα εγκατάστασης joomla, ρύθμιση της Β.Δ

Για λόγους ασφαλείας και προστασίας των αρχείων που χρησιμοποιεί το **Joomla** εδώ μας δίνεται η δυνατότητα δημιουργίας ενός **FTP** (File Transfer Protocol) λογαριασμού. Θα το προσπεράσουμε κάνοντας κλικ στο κουμπί **Επόμενο**.



Εικόνα : 5ο βήμα εγκατάστασης Joomla, ρυθμίσεις ftp.

Στο 6ο βήμα της εγκατάστασης εμφανίζεται η παρακάτω οθόνη στην οποία κάνουμε κάποιες βασικές ρυθμίσεις.



Εικόνα : 6ο βήμα εγκατάστασης joomla, βασικές ρυθμίσεις.

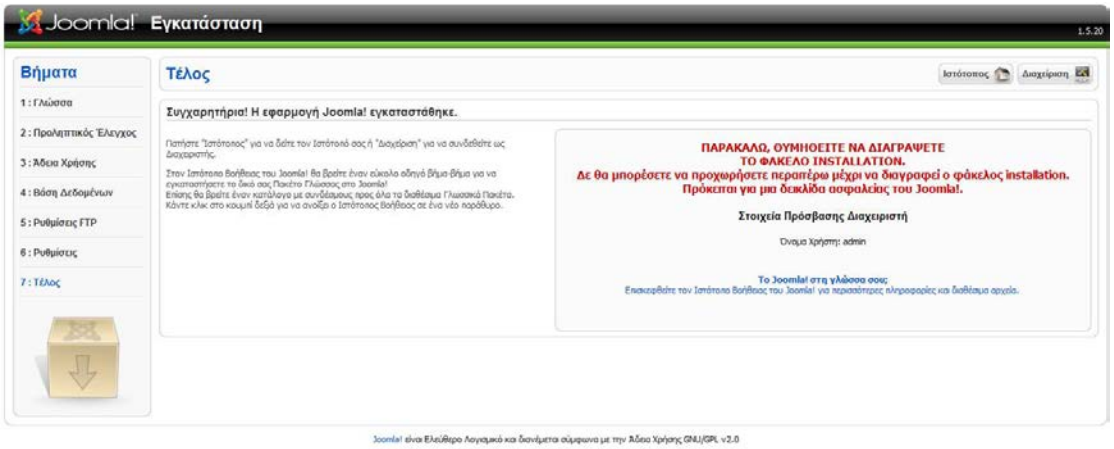
- το πλαίσιο κειμένου **Όνομα Ιστότοπου** εισάγουμε το όνομα της ιστοσελίδας που θα δημιουργήσουμε.

- Στο πλαίσιο κειμένου **Η διεύθυνση ηλεκτρονικού ταχυδρομείου σας** εισάγουμε το **e-mail** του διαχειριστή της ιστοσελίδας το οποίο θα χρησιμεύσει στη μετέπειτα επικοινωνία με τους χρήστες .

- Στο πλαίσιο κειμένου **Κωδικός Διαχειριστή** εισάγουμε τον κωδικό για να μπορούμε να συνδεόμαστε στην Περιοχή Διαχείρισης.

- Στο πλαίσιο κειμένου **Επιβεβαίωση Κωδικού Διαχειριστή** επαναλαμβάνουμε τον ίδιο κωδικό. Στη συνέχεια, κάνουμε κλικ στο κουμπί **Εισαγωγή Ενδεικτικού Περιεχομένου**.

Εάν όλα τα βήματα διεξαχθούν αισίως ,τότε εμφανίζεται η παρακάτω οθόνη.



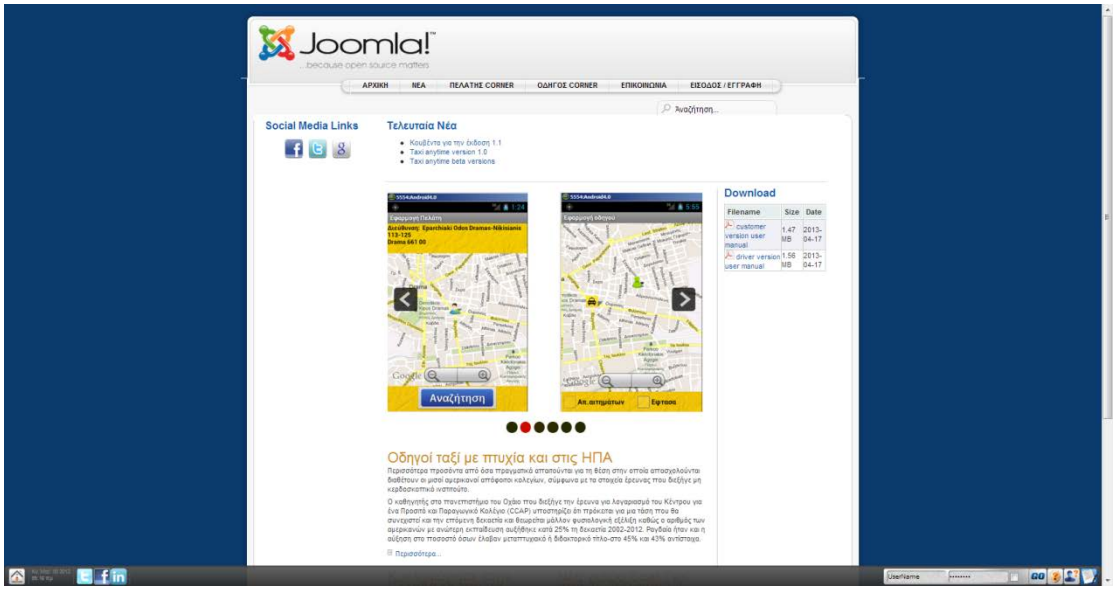
Εικόνα : 7ο και τελευταίο βήμα εγκατάστασης joomla.

Στο τελευταίο βήμα το **Joomla** μας ενημερώνει ότι η εγκατάσταση ολοκληρώθηκε με απόλυτη επιτυχία.

Υπάρχει μία προειδοποίηση με κόκκινα γράμματα η οποία υπενθυμίζει ότι πρέπει να οπωσδήποτε να διαγράψουμε το φάκελο με όνομα **installation** από το φάκελό με το site μας καθαρά για λόγους ασφαλείας.

Κατασκευή της ιστοσελίδας

Για να δούμε την ιστοσελίδα πληκτρολογούμε το link στον browser μας. Το default θέμα που προσφέρει το joomla είναι όπως φαίνεται παρακάτω.



Εικόνα: default template του joomla

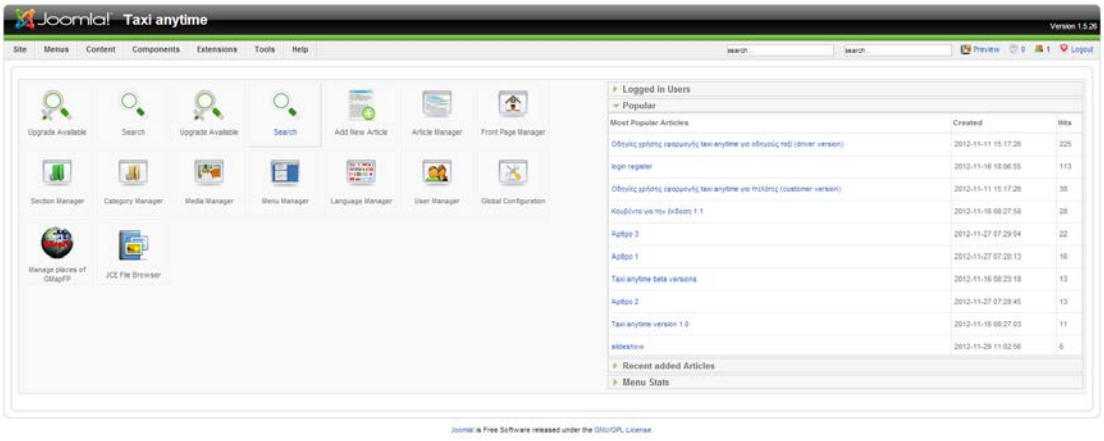
Σημείωση: Το site είναι ήδη ολοκληρωμένο. Για λόγους αναπαράστασης έγινε αλλαγή template σε default. Τα μενού, τα άρθρα, το slideshow, το user bar κλπ όπως είναι λογικό δεν θα υπάρχουν στο πρώτο άνοιγμα της ιστοσελίδας και οι θέσεις τους είτε θα είναι κενές είτε θα υπάρχουν ενδεικτικά αντικείμενα.

Αυτό που δείξαμε είναι το front-end το οποίο βλέπουν οι χρήστες. Για να αποκτήσουμε πρόσβαση στο back-end θα πρέπει στο τέλος του url να προσθέσουμε το /administrator. Δηλαδή http://taxianytime.hostzi.com/administrator/ . Εδώ προσθέσαμε ένα παραπάνω επίπεδο ασφαλείας και έστω και για να ανοίξει η σελίδα για να κάνει Login στο back-end κάποιος θα πρέπει να έχει το κατάλληλο username & password. Εάν τα έχει θα μεταβεί στην σελίδα εισόδου στο back-end όπως φαίνεται παρακάτω.

Εικόνα : Σελίδα διαχείρισης ιστότοπου κατά τη διαδικασία του log in.

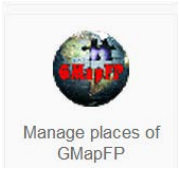
Το password επιλέχθηκε έτσι ώστε με brute force attack να χρειαστούν μερικά εκατομμύρια χρόνια να σπάσει!

Αφού δώσουμε σωστά τα στοιχεία εισόδου θα μεταφερθούμε στο control panel του site.



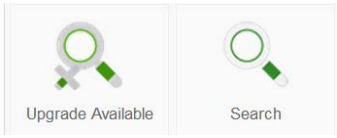
Εικόνα : Σελίδα διαχείρισης ιστότοπου

Από εδώ πλέον μπορούμε να κάνουμε τα πάντα! Το τι κάνει κάθε κουμπί το λέει και ο τίτλος του είναι πολύ αναλυτικά, δεν χρειάζεται κάποια παραπάνω εξήγηση. Θα αναφερθούμε μόνο σε αυτά που δεν θα δει κάποιος την πρώτη φορά που θα μπει στο control panel. Αυτά είναι:



Έχει να κάνει με ένα component που προσθέσαμε το οποίο έχει να κάνει με την δημιουργία χάρτη μέσω Google maps στην ιστοσελίδα. Πέρα από την εστίαση και ένα marker πάνω στον χάρτη στο σημείο που θέλουμε ο χρήστης θα μπορεί να ζητάει και οδηγίες για το πώς θα φτάσει στην συγκεκριμένη τοποθεσία.

Το JCE είναι κυρίως ένας text editor με πολύ περισσότερες δυνατότητες από αυτές που προσφέρει ο default editor του joomla. Από το συγκεκριμένο κουμπί μπορούμε να διαχειριστούμε ότι εικόνες της ιστοσελίδας μας.



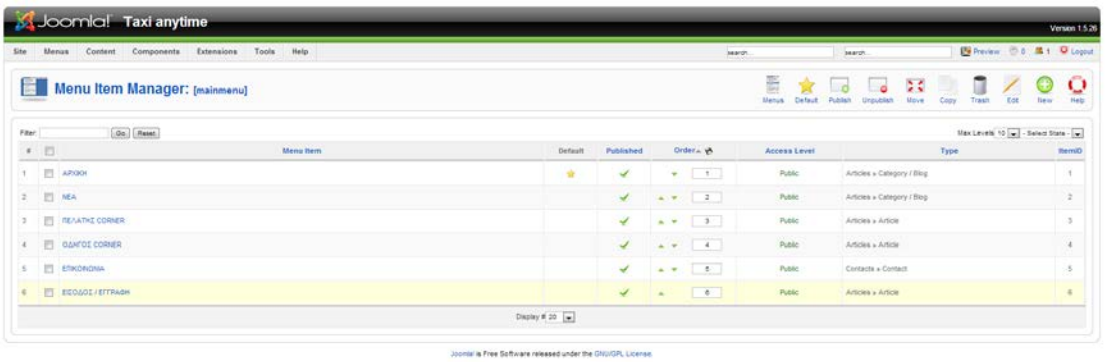
Αυτά τα κουμπιά έχουν να κάνουν με ένα module που προσθέσαμε για αναζήτηση στο site. Το βρήκαμε μακράν καλύτερο από την default αναζήτηση του Joomla.

Όλα τα άλλα κουμπιά κάνουν αυτό ακριβώς που λέει ο τίτλος τους ☺.

10.9.6 Υλοποίηση ιστοσελίδας

10.9.6.1 Διαχείριση Μενού

Είναι η υπηρεσία με την οποία καθορίζουμε τη θέση που θα έχουν τα μενού στον κόμβο μας, τις κατηγορίες των μενού στο σύστημά μας καθώς και το περιεχόμενό τους. Μέσω της υπηρεσίας διαχείρισης μενού μπορεί ο διαχειριστής να προσθέσει, να επεξεργαστεί να διαγράψει ή και να ενημερώσει τα μενού σου συστήματος.

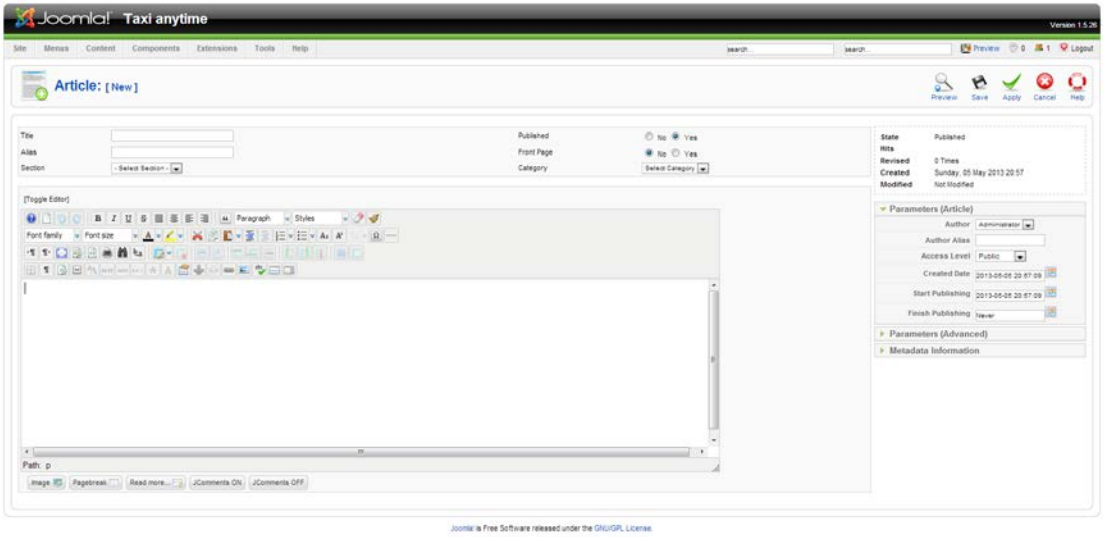


Εικόνα: Διαχείριση Μενού

Δημιουργία νέας σελίδας-άρθρου

Ιδιαίτερα σημαντική είναι η λειτουργία της διαχείρισης του περιεχομένου της ιστοσελίδας, το οποίο μπορούμε να επεξεργαστούμε χρησιμοποιώντας τον JCE Text Editor του Joomla για να δημιουργήσουμε νέες σελίδες και να τροποποιήσουμε τις ήδη υπάρχουσες. Επιλέγοντας την ενότητα αρχικά και στη συνέχεια την κατηγορία στην οποία θέλουμε να εμφανίζεται η νέα ιστοσελίδα που θα δημιουργήσουμε, μπορούμε με απλά βήματα να την εμπλουτίσουμε με το περιεχόμενο της αρεσκείας μας και να τη δημοσιεύσουμε.

Στην εικόνα που ακολουθεί μπορούμε να διακρίνουμε την κονσόλα δημιουργίας νέων σελίδων-άρθρων, τον κενό χώρο για την εγγραφή του κειμένου ,επάνω τον τίτλο τον οποίο θα έχει και τα κουμπιά δεξιά για τις περαιτέρω πιο εξειδικευμένες ρυθμίσεις.

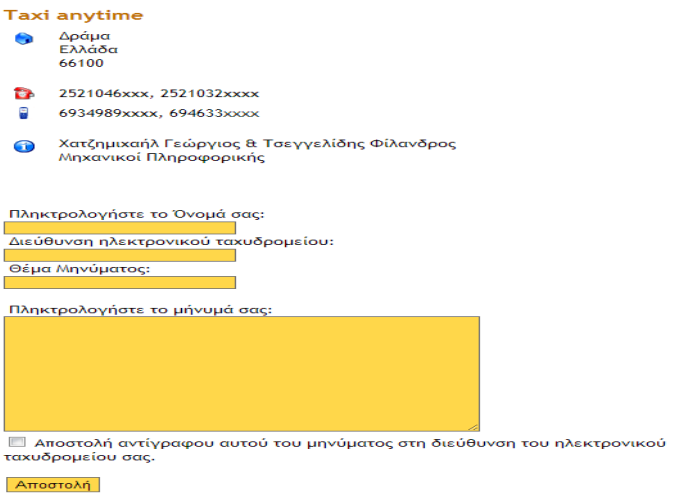


Εικόνα: Δημιουργία νέου άρθρου.

10.9.6.2 Επικοινωνία

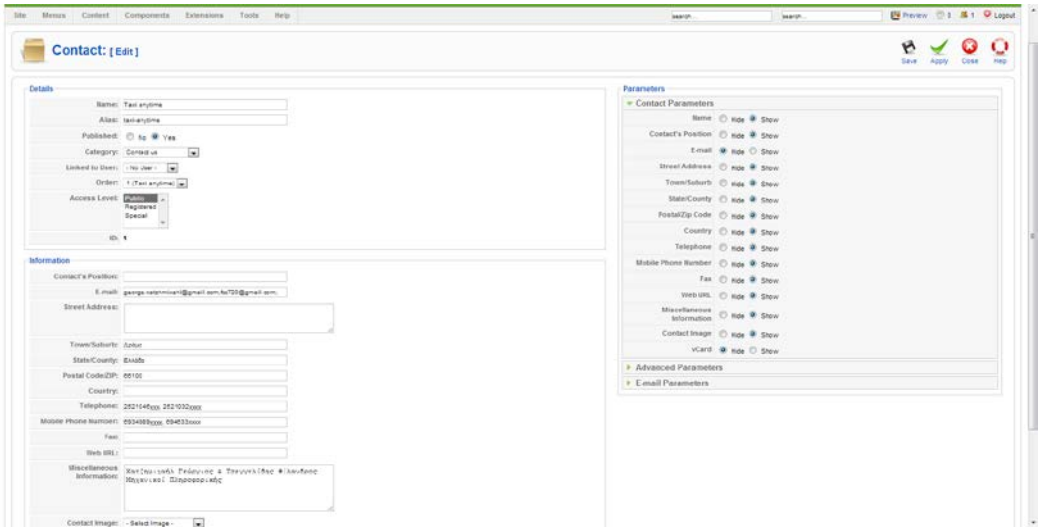
Υπάρχει η δυνατότητα για τους χρήστες της ιστοσελίδας να επικοινωνήσουν με τους

διαχειριστές συμπληρώνοντας τα στοιχεία τους σε μία φόρμας επικοινωνίας. Η φόρμα επικοινωνίας που χρησιμοποιήθηκε στο site φαίνεται παρακάτω.



Εικόνα: Φόρμα επικοινωνίας

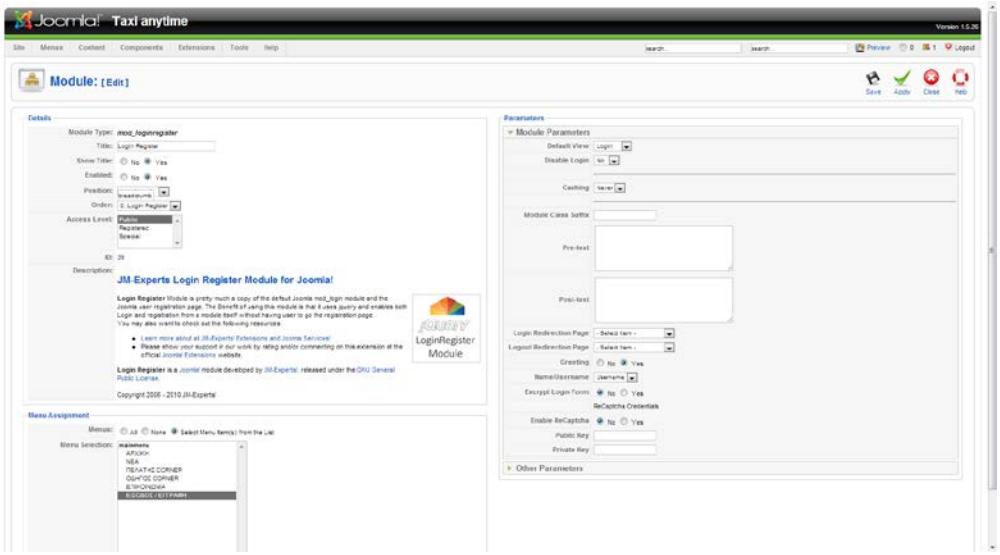
Για να δημιουργήσουμε την φόρμα πηγαίνουμε από τα *Components-> Contacts-> Contacts*, & επιλέγουμε *New*



Εικόνα : Περιοχή Διαχείρισης ΦόρμαςΕπικοινωνίας.

Σύνδεση Χρηστών

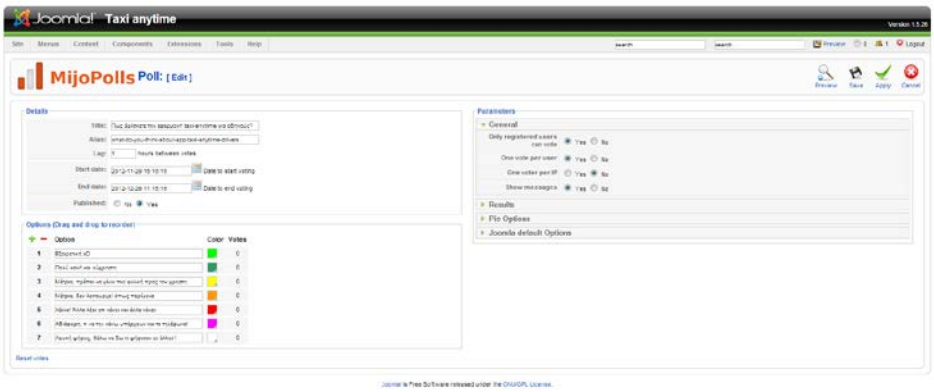
Για τη σύνδεση των χρηστών στην ιστοσελίδα χρησιμοποιείται το JM-Experts Login Register Module του οποίου η περιοχή διαχείρισης απεικονίζεται στην εικόνα που ακολουθεί. Το συγκεκριμένο module δεν κάνει τίποτα παραπάνω από το να συνδέει με ένα JQuery τα default login & register του joomla σε ένα module.



Εικόνα : Περιοχή διαχείρισης ενθέματος για τη σύνδεση χρηστών στο site.

10.9.6.3 Δημοσκοπήσεις

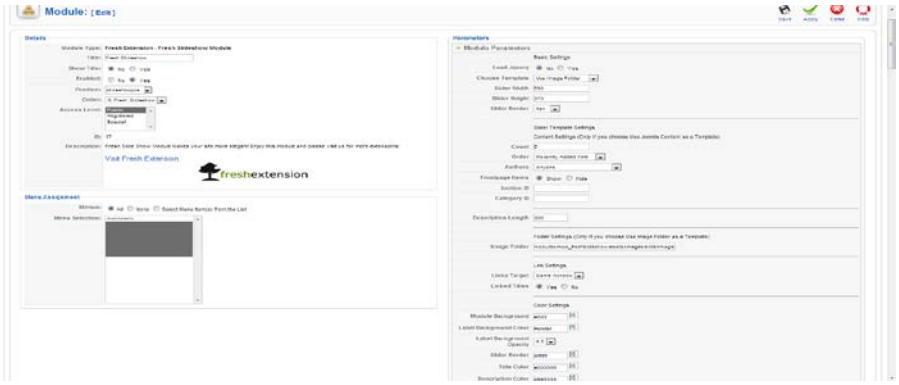
Οι εγγεγραμμένοι χρήστες της ιστοσελίδας, έχουν το δικαίωμα να εκφράσουν την άποψή τους για διάφορα θέματα που τίθενται από τους διαχειριστές του site. Κάθε εγγεγραμμένος χρήστης έχει δικαίωμα να απαντήσει μόνο μια φορά σε κάθε ερωτηματολόγιο ή δημοσκόπηση. Για τις δημοσκοπήσεις χρησιμοποιήσαμε το MijoPolls component. Στο σχήμα που ακολουθεί φαίνεται η περιοχή διαχείρισης των δημοσκοπήσεων.



Εικόνα :Περιοχή διαχείρισης δημοσκοπήσεων

10.9.6.4 Slideshow

Πρόκειται για το ένθεμα που βρίσκεται στο επάνω μέρος της αρχικής σελίδας και είναι υπεύθυνο για τη συνεχή εναλλαγή των φωτογραφιών. Η εφαρμογή αυτή μας επιτρέπει να ορίσουμε έναν φάκελο με φωτογραφίες της αρεσκείας μας , οι οποίες θα εναλλάσσονται στο template σε χρονικό διάστημα της επιλογής μας, προσδίδοντας ιδιαίτερο στυλ στην ιστοσελίδα. Εμείς χρησιμοποιήσαμε το Module **Fresh Slideshow.** Οι ρυθμίσεις της περιοχής διαχείρισης του ενθέματος φαίνονται στην ακόλουθη εικόνα.



Εικόνα :Περιοχή διαχείρισης slideshow

11 Εγκατάσταση περιβάλλοντος ανάπτυξης και οδηγίες

11.1 ΡΥΘΜΙΣΗ ΠΑΡΑΜΕΤΡΩΝ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΑΝΑΠΤΥΞΗΣ ΓΙΑ ANDROID

11.1.1 Εισαγωγή

Για να γράψουμε εφαρμογές Android θα πρέπει να πρώτα να εγκαταστήσουμε και να ρυθμίσουμε το περιβάλλον ανάπτυξης.

Συνεπώς απαιτείται το παρακάτω λογισμικό :

- Java Development Kit (JDK) κατά προτίμηση την τελευταία έκδοση.
- Το IDE Eclipse (Eclipse IDE for Java EE Developers Juno 4.2 package)
- Το πρόσθετο εργαλείο (plugin) του Eclipse Android Development Tools (ADT)
- Το Android SDK

Να σημειώσουμε ότι η Google για να διευκολύνει ακόμη περισσότερο τους προγραμματιστές από το κόπο στησίματος όλων αυτών των εργαλείων, τώρα τελευταία τα παρέχει όλα μαζί σε ένα πακέτο το λεγόμενο ADT Bundle . Το μόνο που χρειάζεται να κάνει ο χρήστης είναι αποσυμπίεση και εκτέλεση του Eclipse.

Κρίνεται όμως απαραίτητο να αναφερθούμε και στις οδηγίες για το πώς θα ρυθμιστεί και χειροκίνητα το περιβάλλον ανάπτυξης εξηγώντας τον τρόπο εγκατάσταση του καθενός στοιχείου ξεχωριστά.

11.1.2 Οδηγίες

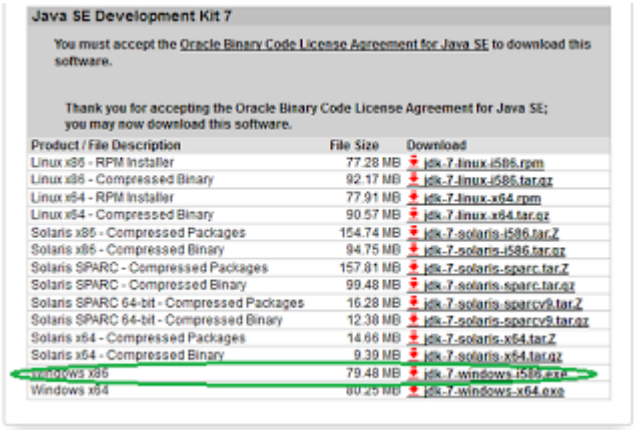
11.1.2.1 Java Development Kit (JDK)

Επισκεπτόμαστε το δικτυακό τόπο
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Και διαλέγουμε το Java Platform(JDK)

Εικόνα 1: Java Platform

Επιλέγουμε Accept license στο check box και διαλέγουμε την έκδοση ανάλογα με το λειτουργικό έχουμε (συνήθως Windows x86).



Εικόνα 2 : Java Platform

Αφού γίνει η λήψη, η διαδικασία της εγκατάστασης είναι τυπική χωρίς επιπλέον ρυθμίσεις. Θα δημιουργηθεί ένα φάκελος σε C:\Program Files\Java\jdk1.7.xx

11.1.2.2 IDE Eclipse

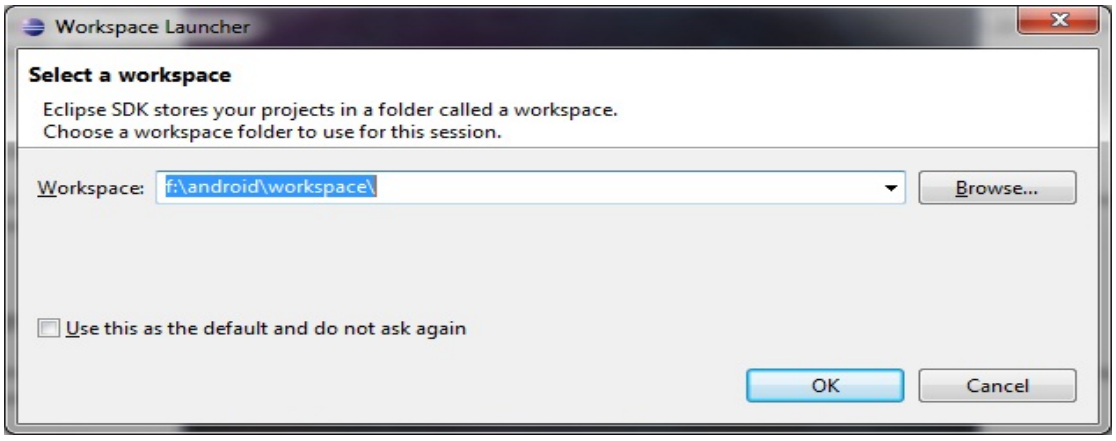
Επισκεπτόμαστε το δικτυακό τόπο

http://www.Eclipse.org/downloads/ και επιλέγουμε Eclipse IDE for Java EE Developers 32/64 bit ανάλογα με το λειτουργικό μας.

Εικόνα 3: Eclipse

Το αποσυμπιέζουμε σε κάποιο κατάλογο και από εκεί τρέχουμε το Eclipse.exe. Δεν χρειάζεται εγκατάσταση μιας και είναι portable.

Με το που ανοίγουμε πρώτη φορά το Eclipse μας ζητά το μέρος που θα αποθηκεύονται τα έργα μας.

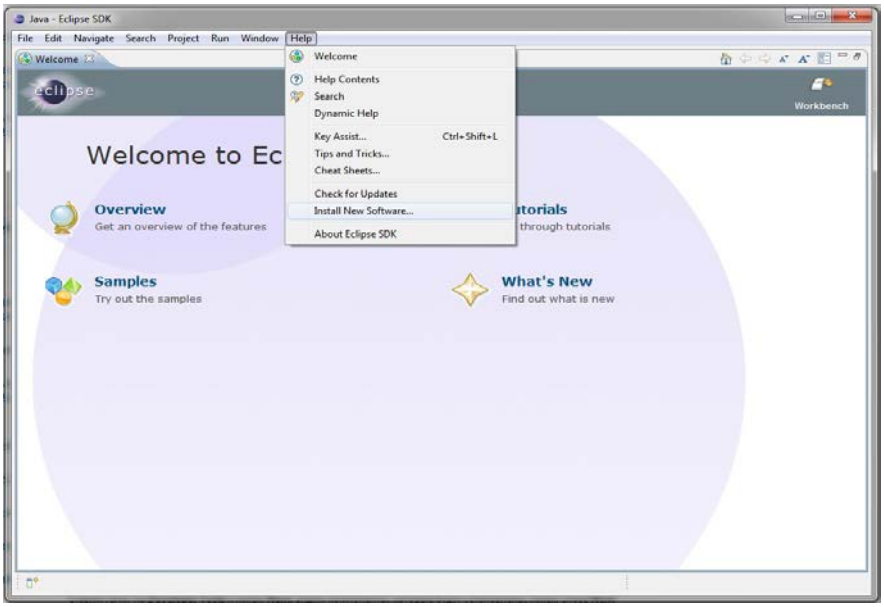


Εικόνα 4 : Eclipse workspace

Καλό είναι λοιπόν να θυμόμαστε αυτή την διαδρομή.

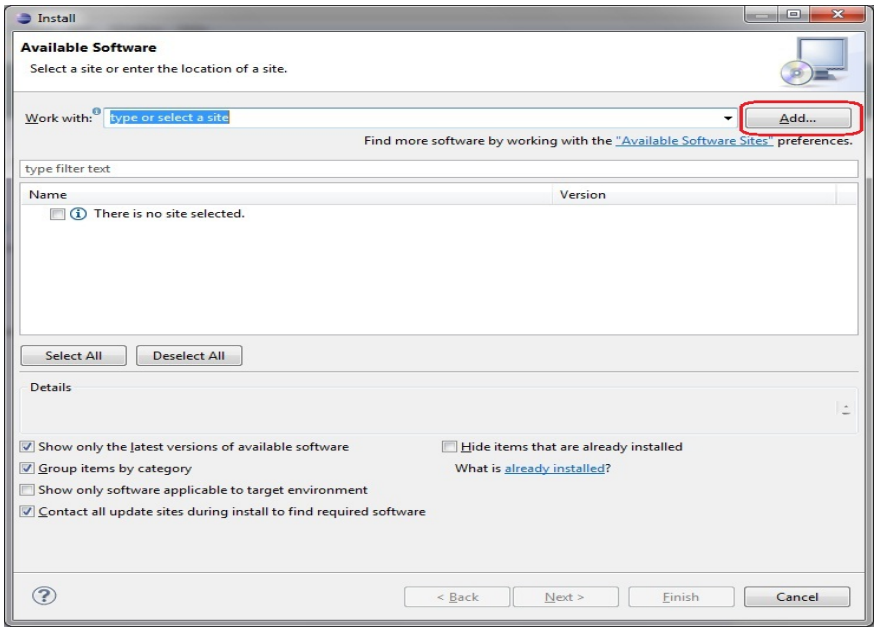
11.1.2.3 Πρόσθετο ADT

Έτσι μετά από αυτό μεταβαίνουμε στην κύρια οθόνη του Eclipse και είναι ώρα να εγκαταστήσουμε το πρόσθετο ADT.



Εικόνα 5: Eclipse main

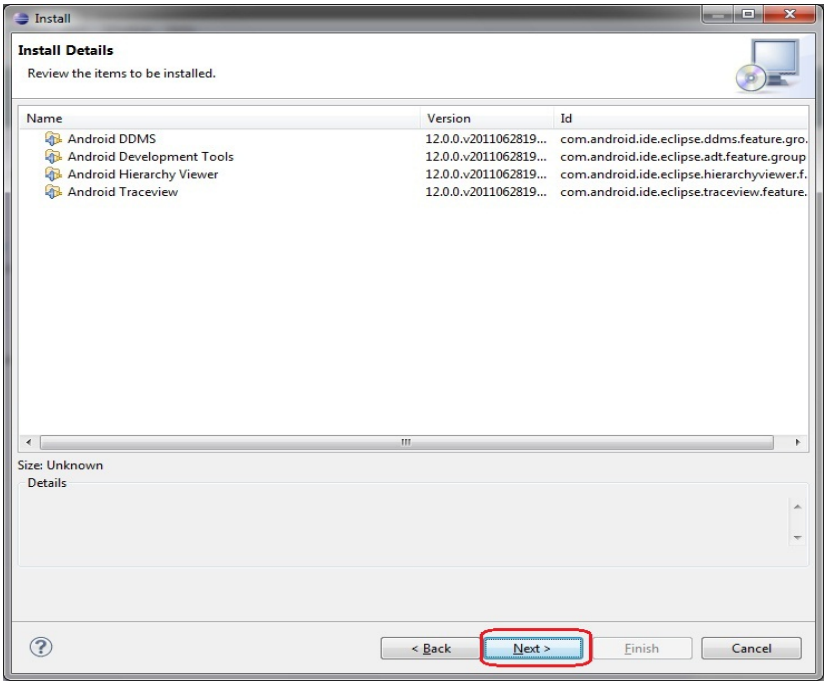
Από το menu Help -> Install new software και έχουμε την εξής εικόνα



Εικόνα 6 : Eclipse Install new software

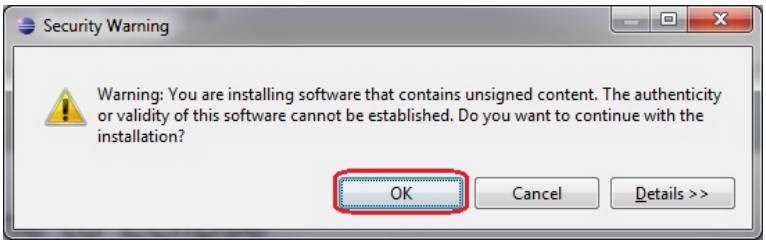
Επιλέγοντας Add.. γράφουμε σαν name ADT plugin και στο location το link https://dl-ssl.google.com/android/Eclipse/

Με αυτόν τον τρόπο μας εμφανίζονται τα απαραίτητα developer tools, πατάμε next



Εικόνα 7 : Eclipse ADT

Αποδεχόμαστε τους όρους και συνεχίζει η εγκατάσταση. Εάν σε κάποιο σημείο κατά την εγκατάσταση μας βγάλει μήνυμα προειδοποίησης για μη υπογεγραμμένο περιεχόμενο το αγνοούμε πονώντας οκ.



Εικόνα 8: ADT Warning

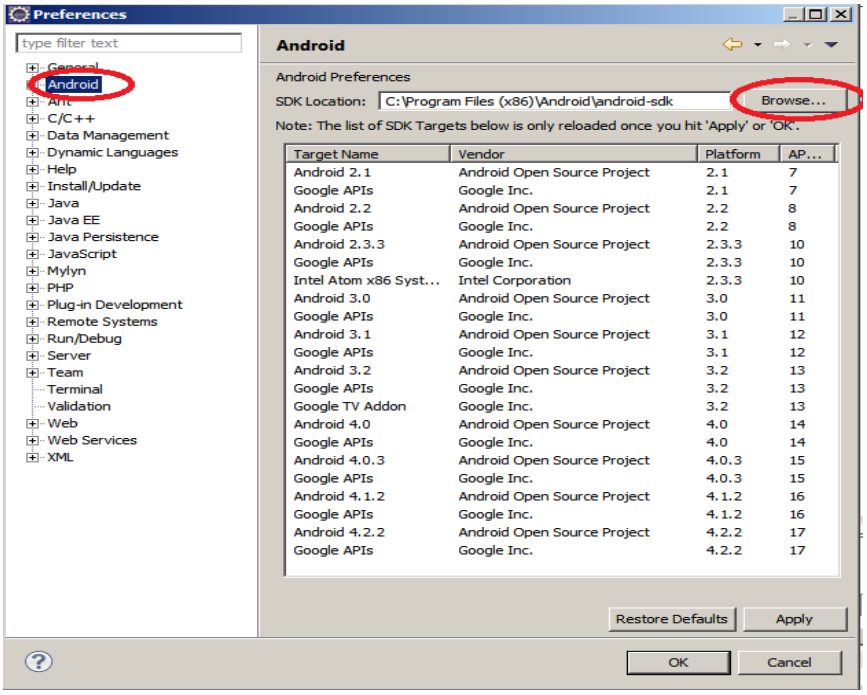
Όταν ολοκληρωθεί κάνουμε επανεκκίνηση του Eclipse, θα μας το ζητήσει εξάλλου.

11.1.2.4 Android SDK

Το τελευταίο κομμάτι που έμεινε είναι το android SDK. Μεταβαίνουμε στη σελίδα http://developer.android.com/SDK/index.html , στην επιλογή use existing ide κατεβάζουμε το SDK.

Και εδώ η διαδικασία της εγκατάστασης είναι τυπική χωρίς επιπλέον ρυθμίσεις. Τώρα αυτό που έχουμε να κάνουμε είναι να συνδέσουμε το SDK με το περιβάλλον ανάπτυξης μας.

Μέσα από το Eclipse ,Windows ->Preferences



Εικόνα 9 : Android SDK location

Από αριστερά πατάμε στην καρτέλα Android και ως SDK location βάζουμε την τοποθεσία που εγκαταστάθηκε το Android SDK.

Τέλος προτείνεται να πάμε πάλι στο menu του Eclipse , Windows-> SDK Manager και να ενημερώσουμε τα πακέτα που μας προτείνει.

11.2 ΡΥΘΜΙΣΗ ΠΑΡΑΜΕΤΡΩΝ ΠΕΡΙΒΑΛΛΟΝΤΟΣ ΑΝΑΠΤΥΞΗΣ ΓΙΑ WAMP SERVER

11.2.1 Εισαγωγή

Ο Wamp server είναι στην ουσία ένα περιβάλλον ανάπτυξης web εφαρμογών σε περιβάλλον Windows. (αντίστοιχο με XAMPP). Μας επιτρέπει να αναπτύσσουμε εφαρμογές σε PHP και MySQL εύκολα και γρήγορα.

11.2.2 Οδηγίες

Μεταβαίνουμε στην ιστοσελίδα http://www.wampserver.com/en/ και επιλέγουμε download από το menu. Ανάλογα με το τι λειτουργικό έχουμε 32bit ή 64 bit.

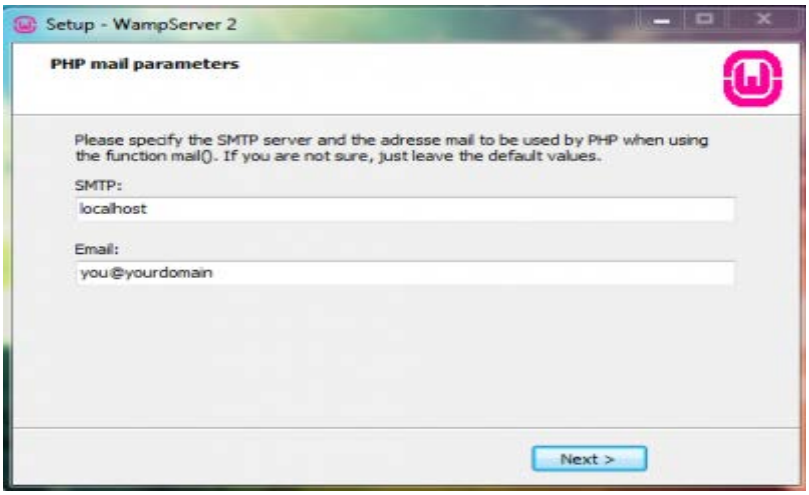
Κατά προτίμηση θα επιλέγουμε εκείνη την έκδοση με τα τις εξής λεπτομέρειες :

- Apache 2.2.22
- MySql 5.5.24
- PHP 5.3.13

Ξεκινώντας την εγκατάσταση διαλέγουμε το path που θα είναι τα αρχεία μας στον server

Εικόνα 10 : Wamp directory

Πατάμε Next, ως στοιχεία SMTP και local host αφήνουμε τα προεπιλεγμένα .



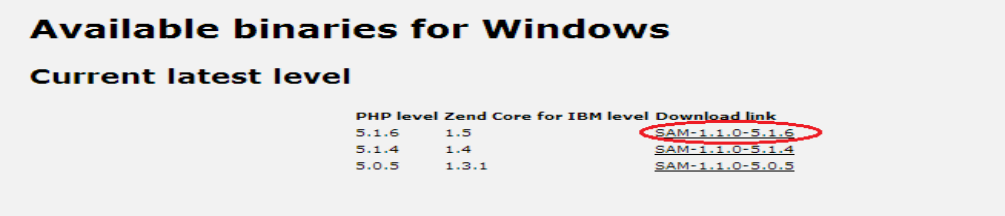
Εικόνα 11 : Wamp settings

Αφού ολοκληρωθεί η εγκατάσταση , για να διαπιστώσουμε ότι λειτουργεί σωστά πρώτα πρέπει να το εικονίδιο του wamp στο system tray(γραμμή εργαλείων windows) να είναι πράσινο. Εφόσον είναι ανοίγουμε έναν περιηγητή και σαν διεύθυνση δίνουμε http://localhost . Αν μας βγάλει την παρακάτω εικόνα τότε έχει εγκατασταθεί σωστά αλλιώς θα πρέπει να επαναλάβουμε την εγκατάσταση ή και να ανατρέξουμε τα troubleshoots από το site του Wamp

Εικόνα 12 : Wamp localhost

Μια τελευταία ρύθμιση που έχουμε να κάνουμε είναι να εγκαταστήσουμε την εξωτερική βιβλιοθήκη SAM (simple asynchronous messages) η οποία χρειάζεται για την επικοινωνία με το mosquito (τον διαμεσολαβητή).

Πηγαίνουμε στην τοποθεσία http://project-sam.awardspace.com/downloads.htm και κατεβάζουμε την τελευταία έκδοση .



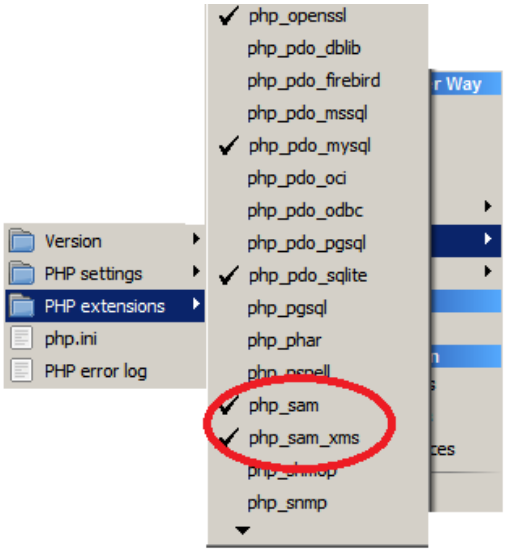
Εικόνα 13: Sam library

Το συμπιεσμένο αρχείο περιέχει τα :

- php_sam_xms.dll
- php_sam.dll

Τα κάνουμε αποκοπή στην τοποθεσία C:\wamp\bin\php\php5.3.13.

Τώρα θα φαίνονται σαν επιλογές στα extensions της php στο Wamp, οπότε θα τα ενεργοποιήσουμε . Κάνουμε από click στο εικονίδιο του Wamp-> php -> php extensions και στη λίστα επιλέγουμε τα php_sam_xms και php_sam .



Εικόνα 14 : php extensions

Στο σημείο αυτό ολοκληρώσαμε και την ρύθμιση του server και είμαστε έτοιμοι να αναπτύξουμε.

11.3 ΕΡΓΑΛΕΙΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

- ✓ Eclipse IDE (Juno 4.2)
- ✓ Java Development Kit (JDK)
- ✓ Wamp Server
- ✓ MySQL Workbrench 5.2
- ✓ Notepad ++
- ✓ ArgoUml
- ✓ Dia
- ✓ yEd Graph Editor
- ✓ Joomla

Παρακάτω αναλύουμε λίγο για το καθένα :

- **Eclipse IDE (Juno 4.2)**

  Το βασικό περιβάλλον εργασίας για την ανάπτυξη εφαρμογών. Πρόκειται για ένα ολοκληρωμένο σύστημα ανάπτυξης που υποστηρίζει πολλές γλώσσες προγραμματισμού (c/c++, java, php), με αρκετά καλή δομή και σχετικά εύκολο στην εκμάθησή του.

  - **ADT Plugin**
    Είναι μια προέκταση του βασικού IDE και μας δίνει τη δυνατότητα και τα εργαλεία (πχ DDMS,LogCat,debug κ.α) για την ανάπτυξη Android εφαρμογών .

  - **Android SDK**
    Είναι όλες οι βιβλιοθήκες και διεπαφές που χρειάζονται για την εύκολη ανάπτυξη Android εφαρμογών.

- **Java Development Kit (JDK)**

  Στην ουσία είναι ένα πρόγραμμα για να γράφουμε Java εφαρμογές. Αποτελείται από ένα εκτελέσιμο περιβάλλον που ενσωματώνεται στο επίπεδο πάνω από το λειτουργικό σύστημα , καθώς και από τα εργαλεία για να προγραμματίσουμε (compiler,debugger κτλ)

- **Wamp Server**
  Είναι ένα χρήσιμο περιβάλλον ανάπτυξης δικτυακών εφαρμογών για τα Windows (υπάρχει και τα xampp που είναι cross-platform)
  Έχει συγκεντρωμένους τον Apache web server και τον MySQL Server που χρειαζόμαστε για να αποθηκεύουμε ότι είναι απαραίτητο σε μια βάση και το phpMyAdmin για την εύκολη διαχείριση .

- **MySQL Workbrench 5.2**
  Είναι ένα βοηθητικό εργαλείο οπτικής σχεδίασης (και αναπαράστασης) της βάσης με ενσωματωμένη την ανάπτυξη κώδικα SQL , γενική διαχείριση, σχεδιασμό και συντήρηση του MySQL server.

- **Notepad ++**
  Ίσως ο δημοφιλέστερος text editor. Είναι ελαφρύς και χρησιμοποιείται αρκετά για συγγραφή κώδικα σε ποικίλες γλώσσες. Εμείς τον χρησιμοποιήσαμε για τα php scripts .

- **ArgoUml**
  Χρήσιμο εργαλείο για την παραγωγή διαγραμμάτων (UML diagrams) που αναπαριστούν την κατάσταση του λογισμικού, τον αρχιτεκτονικό σχεδιασμό, με απώτερο σκοπό την ευκολότερη και γρηγορότερη κατανόηση του συστήματος λογισμικού.
  Εμείς το χρειαστήκαμε για τα διαγράμματα περιπτώσεων χρήσης.

- **Dia**
  Χρησιμοποιήσαμε αυτό το πρόγραμμα για το σχεδιασμό του μοντέλου οντοτήτων – συσχετίσεων (E-R) της βάσης δεδομένων. Έχει αρκετές επιπλέον λειτουργίες όπως διαγράμματα ροής , UML διαγράμματα , διαγράμματα κυκλωμάτων (ηλεκτρικά, ψηφιακά), διαγράμματα δικτύων κ.α

- **yEd Graph Editor**
  Είναι μια ισχυρή desktop εφαρμογή η οποία αναπαράγει γρήγορα και αποτελεσματικά υψηλής ποιότητας διαγράμματα. Μπορούμε να δημιουργήσουμε διαγράμματα χειροκίνητα ή να εισάγουμε εξωτερικά δεδομένα για ανάλυση.
  Εμείς το χρησιμοποιούμε για την οπτική αναπαράσταση του συστήματος μας για να γίνει πιο κατανοητή η ροή της πληροφορίας.

- **Joomla**
  Ίσως ο δημοφιλέστερος διαχειριστής περιεχομένου. Το επιλέξαμε για να κατασκευάσουμε το υποστηρικτικό site για τις εφαρμογές μας.

- **FileZilla**
  Είναι ένας ftp client τον οποίο χρησιμοποιούμε για να διαχειριζόμαστε απομακρυσμένα τα αρχεία της ιστοσελίδάς μας.

12 Μελλοντικές επεκτάσεις και συμπεράσματα

12.1 Μελλοντικές επεκτάσεις

Όπως σε κάθε εφαρμογή πάντα θα υπάρχει δυνατότητα περαιτέρω βελτίωσης και εξέλιξης έτσι και στη δικιά μας μπορούμε να επισημάνουμε κάποιες βελτιώσεις και προσθήκες. Μπορεί δηλαδή ότι η πτυχιακή να έχει ολοκληρωθεί, οι εκδόσεις της εφαρμογής taxi anytime όμως όχι! Αν και πλέον η εφαρμογή είναι πλήρως λειτουργική, αυτό δεν σημαίνει ότι δεν υπάρχει περιθώριο βελτίωσης. Έχουμε ήδη ετοιμάσει μια λίστα με το τι θα περιλαμβάνουν οι επόμενες εκδόσεις της εφαρμογής

12.1.1 Μερικές προσθήκες:

- Εντολή αναζήτησης βάσει φωνητικής εντολής
- Δυνατότητα live – chat
- Δυνατότητα παραγγελίας και εκτός mobile , αλλά μέσα από site (web service).
- Μεταφορά του μενού από το πλήκτρο menu, σε μια μπάρα μενού που θα είναι στο πάνω μέρος της οθόνης σας.
- Προσθήκη προαιρετικού πεδίου "αυτοκίνητο" στο προφίλ του χρήστη (driver version) ώστε οι πελάτες να ξέρουν και με τί αυτοκίνητο θα γίνει η μεταφορά! Η πληροφορία θα είναι ορατή στους πελάτες (customer version).
- Δήλωση των οδηγών εάν επιτρέπουν κατοικίδια στο αυτοκίνητο (driver version). Η πληροφορία θα είναι ορατή στους πελάτες (customer version).
- Θα δώσουμε την δυνατότητα στους πελάτες να επιλέγουν οι ίδιοι τους την μέγιστη απόσταση ανάμεσα σε αυτούς και τους οδηγούς οι οποίοι θα ενημερωθούν για την παραγγελία. **(Προς το παρόν το έχουμε ορίσει εμείς)**

12.1.2 Μερικές βελτιώσεις :

- Βελτίωση του user interface (με χρήση κάποιων έτοιμων βιβλιοθηκών)
- Βελτίωση του τρόπου επιλογής εμφανιζόμενης εικόνας (browsing μέσα από το κινητό και όχι χειροκίνητα δίνοντας URL)

- Βελτίωση και ασφάλεια στον κώδικα java με το εργαλείο Pro Guard για να είναι πιο δύσκολη όποια προσπάθεια για ανάστροφη μηχανική.
- Ενσωμάτωση του Google Maps Android Api v2 που διορθώνει πολλά bugs από το api v1 και προσθέτει νέα χαρακτηριστικά .
- Ενσωμάτωση της πρόσφατης τεχνολογίας GCM (Google Cloud Messaging) για το σύστημα ειδοποιήσεων, γιατί δεν υπάρχουν οι περιορισμοί που υπήρχαν στο C2DM (το προηγούμενο service της Google).

12.2 Συμπεράσματα

Στην πτυχιακή αυτή εργασία, μάθαμε από το μηδέν μια νέα γλώσσα προγραμματισμού, java για android. Όπου τελικά διαπιστώσαμε ότι δεν ήταν κάτι δύσκολο διότι η java είναι μια καθαρά αντικειμενοστραφής γλώσσα, και εμείς κατά την διάρκεια των σπουδών μας διδαχθήκαμε πολύ καλά αντικειμενοστραφή προγραμματισμό. Αυτό που είχαμε να κάνουμε ουσιαστικά ήταν να μάθουμε τις συντακτικές διαφορές ανάμεσα σε C++ που ήδη γνωρίζαμε, και σε java για android.

Η ενασχόληση με την παρούσα πτυχιακή μας επέφερε αρκετό κέρδος και διεύρυνε αρκετά τις γνώσεις μας πάνω στην ανάπτυξη λογισμικού κινητών συσκευών. Κατανοήσαμε τον τρόπο που δομούνται τέτοιες εφαρμογές, το πώς συμπεριφέρονται και αλληλεπιδρούν με άλλες καθώς και με το δίκτυο.

Η εμπειρία με το λειτουργικό Android ήταν αρκετά ευχάριστη. Υπάρχουν πάρα πολλές πηγές που μπορεί να συλλέξει κάποιος πληροφορίες και υλικό για να αναπτύσσει εφαρμογές. Η Google έχει πολύ εύχρηστο documentation που προσελκύει συνέχεια νέους προγραμματιστές να ενασχοληθούν με αυτές τις υπηρεσίες. Επίσης όλα τα προγραμματιστικά εργαλεία παρέχονται δωρεάν.

Εμείς στην συγκεκριμένη εργασία ερευνήσαμε τα διάφορα συστήματα ειδοποιήσεων που μπορούν να χρησιμοποιηθούν στο Android , καταλήγοντας σε ποιο μας ταιριάζει και κατανοώντας τον τρόπο λειτουργίας τέτοιων πρωτοκόλλων. Ως αποτέλεσμα προσφέραμε μια ολοκληρωμένη και ενιαία λύση στο σύστημα διαχείρισης κλήσεως ταξί, βελτιώνοντας την ποιότητα παροχής υπηρεσιών αναζήτησης οδηγού ταξί από πελάτες.

Μερικά επιπλέον οφέλη :

- Μάθαμε ένα νέο IDE, το eclipse, το οποίο είναι μακράν καλύτερο από την Borland που διδαχθήκαμε στην σχολή.
- Μάθαμε νέο design pattern και μελετήσαμε προγραμματίστηκες τεχνικές ώστε ο κώδικας μας να είναι αρκετά καλής ποιότητας.
- Εξασκήσαμε τις γνώσεις μας στην **PHP** φτιάχνοντας έναν **PHP** server για τις εφαρμογές.

- Είδαμε πως είναι να παίρνεις κομμάτι της δουλειάς κάποιου άλλου και να πρέπει να την κατανοήσεις και να την τροποποιήσεις ώστε να την προσαρμόσεις στις δικές σου απαιτήσεις (αναφερόμαστε στο πρωτόκολλο MQTT της IBM)

- Μελετήσαμε τα web CMS και μάθαμε να χρησιμοποιούμε ένα από αυτά (**J**oomla) ώστε να κατασκευάσουμε την ιστοσελίδα των εφαρμογών.
- Μάθαμε πώς είναι να δουλεύουμε συνεργατικά σε βαθμό που ο ένας να εξαρτάτε από την δουλειά του άλλου για να προχωρήσει το project, κάτι το οποίο προσπαθήσαμε να αποφύγουμε όσο αυτό ήταν δυνατό.

Νιώθουμε περήφανοι για το αποτέλεσμα που καταφέραμε και διαπιστώσαμε ότι στον προγραμματισμό τίποτα δεν είναι αδύνατο.

Βιβλιογραφία και πηγές

- Donn Felker, Joshua Dobbs "Android Application Development for Dummies". 2010
- Shane, Conder, Lauren, Darcey "Android Wireless Application Development " . 2η Έκδοση 2012
- Mark L. Murphy "Beginning Android". 2009
- Reto Meier "Professional Android 2Application Development". 2010
- Lauren Darcey , Shane Conder "Sams Teach yourself Android Application Developement". 2010
- James Steele , Nelson To "The Android Developer's Cookbook". 2010
- http://developer.android.com/guide/components/index.html
- https://developers.google.com/maps/documentation/android/v1/hello-mapview
- http://www.androidsnippets.com/distance-between-two-gps-coordinates-in-meter
- http://www.vogella.com/articles/Android/article.html
- https://developer.android.com/reference/android/os/Build.html
- http://developer.android.com/guide/topics/ui/layout/listview.html
- http://www.grokkingandroid.com/android-checking-connectivity/
- http://tokudu.com/2010/how-to-implement-push-notifications-for-android/
- http://mosquitto.org/documentation/
- http://mqtt.org/documentation
- http://en.wikipedia.org/wiki/Content_management_system
- http://www.cmsreview.com/
- http://pacific.jour.auth.gr/content_management_systems/
- http://www.web-builders.gr/cms-advantage.htm
- www.joomla24.com
- www.siteground.com/joomla-hosting/joomla-templates.htm

- http://www.rockettheme.com/joomla?gclid=CKnR9Oj72KoCFQcOfAodrHgG-Q
- www.joomla.gr
- http://en.wikipedia.org/wiki/Joomla
- http://onestoryeveryday.com/wordpress-vs-drupal-vs-joomla-essential-reading-2011.html

Επεξηγήσεις όρων

- **GSM**[1]**:** είναι ένα κοινό Ευρωπαϊκό ψηφιακό σύστημα κινητής τηλεφωνίας
- **Four way handshake**[2] : Είναι μια μέθοδος στις TCP συνδέσεις ώστε να ξέρουμε ότι τα δεδομένα που στέλνουμε και λαμβάνουμε φτάνουν με βεβαιότητα στον προορισμό τους
- **SSL**[3] : Το SSL (Secure Sockets Layer) έχει να κάνει με κρυπτογράφηση. Δημιουργείται ένα ιδιωτικό κρυπτογραφημένο κανάλι ανάμεσα στην εφαρμογή (client) και τον server.
- **LDAP**[4] :Το LDAP είναι ένα πρωτόκολλο ανοικτού προτύπου για την πρόσβαση σε υπηρεσίες καταλόγου X.500. Το πρωτόκολλο τρέχει πάνω από το επίπεδο μεταφοράς ενός δικτύου, στην περίπτωση του Διαδικτύου αυτό είναι το TCP**.**
- **OPML**[5] : Είναι ένα είδος αρχείων που χρησιμοποιούνται για να φτιάξουν καταλόγους, τους οποίους θα μπορούν να χρησιμοποιήσουν και άλλα προγράμματα.

Γλωσσάρι

VM Virtual Machine

SDK Software Development Kit

MQTT Message Queuing Telemetry Transport

IMEI International Mobile Station Equipment Identity

JSON Javascript Object Notation

C2DM Cloud to Device Messaging

GCM Google Cloud Messaging

QOS Quality Of Service

SAM Simple Asynchronous Messaging

E-R Entity–Relationship

SSL Secure Sockets Layer

CMS Content Management System

GSM Global System for Mobile communications

ΠΑΡΑΡΤΗΜΑ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ ΕΦΑΡΜΟΓΩΝ

ΕΦΑΡΜΟΓΗ ΠΕΛΑΤΗ

ПАКЕТО app.taxiAnytimeCustomer.Common

ΑΡΧΕΙΟ ConnectionDetectorTask.java

```
package app.taxiAnytimeCustomer.Common;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;


/**
 *
 * @info
 *Κλάση υπεύθυνη για να ελέγχει τη διαθεσιμότητα των
 * δυο server που απαιτούνται για να λειτουργήσει η εφαρμογή
 * καθώς και για το αν είναι συνδεδεμένος ο χρήστης σε wifi/3g/gps
 */
public class ConnectionDetectorTask  extends AsyncTask<String, String , HashMap<String,
Boolean> > {


    public Activity activityContext;
    private ProgressDialog pDialog;
    private int timeOut = 2000 ; //ms
    private int HTTP_GOOD_REQUEST = 200;
    private HashMap<String, Boolean> conState = new HashMap<String, Boolean>();

    public ConnectionDetectorTask(Activity ActCon) {
        activityContext = ActCon;

    }



    /**
         * @info
         * Πριν ξεκινήσει η διαδικασία ελέγχου  εμφανίζεται μήνυμα στην οθόνη χρήστη
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία ελέγχου   εμφανίζεται μήνυμα στην οθόνη χρήστη
         * */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(activityContext);
        pDialog.setMessage("Περιμένετε..");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);
        pDialog.show();
    }
```

```
/**
     * @return HashMap με τις καταστάσεις σύνδεσης
     * @info
     * Εδώ γίνεται ο έλεγχος για τη σύνδεση
     * @details
     *
     *
     * */
    @Override
    protected HashMap<String, Boolean> doInBackground(String... args) {

        try {
            conState.put("serverState", ServersAreReachable(activityContext) );
            conState.put("serviceState", isConnectingToInternet(activityContext) );

            if( ServersAreReachable(activityContext) == true &&
                isConnectingToInternet(activityContext) == true){
                conState.put("connectionState", true) ;
            }
            else{
                conState.put("connectionState",false) ;
            }
        }

        catch(Exception e){
            conState.put("serverState", false );
            conState.put("serviceState", false );
            conState.put("connectionState",false) ;

            e.printStackTrace();
        }

        return conState;
    }

    /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αν τελικώς έχουμε σύνδεση ξεκινά η επόμενη δραστηριότητα,
         * διαφορετικά εμφανίζεται ανάλογο μύνημα και τερματίζει η εφαρμογή.
         *
         * **/
    @Override
    protected void onPostExecute(HashMap<String, Boolean> result) {
        // dismiss the dialog once done
        pDialog.dismiss();
        pDialog = null;

        //Log.d("servicestate - serverstate-con",String.valueOf(serviceState)+"-
"+String.valueOf(serverState)+"-"+String.valueOf(connectionState));

        if(result.get("serviceState") == false) {
            new showAlertMessage(activityContext ,"Ωχ!Αδυναμία σύνδεσης!!!","Άνοίξτε το GPS
ή/και to wifi/3g");
        }
        else if(result.get("serverState") == false){
            new showAlertMessage(activityContext ,"Ωχ!Αδυναμία σύνδεσης!!!","Αδύναμία
επικοινωνίας με τους servers! Δοκιμάστε αργότερα.");
        }

    }
//=====================================================

/**
 * @info
 * Μέθοδος για τον έλεγχο διαθεσιμότητας του server
 *
 * @details
 * Αρχικά προσπαθεί να κάνει ένα request στην ip του server,
 * θέτοντας ένα timeout 3 sec.
 * Αν είναι σε λειτουργία ο server θα πάρουμε ένα θετικό response (HTTP 200) ,
 * διαφορετικά δεν υπάρχει σύνδεση
 *
 * @return
 * true  : Όταν υπάρχει σύνδεση
 * false : Όταν δεν είναι εφικτή η σύνδεση στον server
 *
 */
```

```
private  boolean ServersAreReachable(Activity activContext) {


        final ConnectivityManager connMgr = (ConnectivityManager)
activContext.getSystemService(Context.CONNECTIVITY_SERVICE);
        final NetworkInfo netInfo = connMgr.getActiveNetworkInfo();

        if (netInfo != null && netInfo.isConnected()) {
            //  Some sort of connection is open, check if server is reachable
            try {
                URL url = new URL("http://"+globalVariables.getInstance().getIP_ADDRESS());
                HttpURLConnection urlc = (HttpURLConnection) url.openConnection();

                urlc.setConnectTimeout(timeOut);
                urlc.connect();
                if (urlc.getResponseCode() == HTTP_GOOD_REQUEST) {
                    return true;
                } else { // Anything else is unwanted
                    return false;
                }
            } catch (IOException e) {

            }
        }

            return true;
    }

//================================================================================
    /**
     * @info
     * Γίνεται έλεγχος για το αν υπάρχει σύνδεση της συσκευής μας με το internet(wifi ή 3G),
     * καθώς και αν το gps είναι ανοικτό.
     *
     * @details
     * Χρησιμοποιούμε τις ενσωματωμένες κλάσεις του android
(LocationManager,ConnectivityManager),
     * για να ελέγξουμε αν υπάρχει ενεργή συνδεσιμότητα
     *
     * @return
     * true  : Αν είναι ανοικτό το GPS και έχουμε πρόσβαση στο internet
     * false : Αν δεν ισχύει έστω και ένα από τα παραπάνω
     */
    private boolean isConnectingToInternet(Activity activContext)
    {

        try {
            LocationManager GpsSservice =
(LocationManager)activContext.getSystemService(Context.LOCATION_SERVICE);
            ConnectivityManager connManager
=(ConnectivityManager)activContext.getSystemService(Context.CONNECTIVITY_SERVICE);
            boolean GpsEnabled =
GpsSservice.isProviderEnabled(LocationManager.GPS_PROVIDER);
            NetworkInfo info = connManager.getActiveNetworkInfo();


            if( !GpsEnabled) {

                return false;
            }
            else if(info.getState() != NetworkInfo.State.CONNECTED ) {

                return false;
            }

        }
        catch(Exception e)
        {
            e.printStackTrace();
            return false;
        }

        return true;
    }



}
```

ΑΡΧΕΙΟ Contact.java

```
package app.taxiAnytimeCustomer.Common;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.AsyncTask;
import android.widget.Toast;


/**
 *  @info
 *  Κλάση υπεύθυνη για τη διαχείριση κλήσεως/sms σε περίπτωση που χρειαστεί
 *
 */

public class Contact  {

    protected Activity actContext;
    private String mNumber;

    public Contact(Activity theContext) {
        actContext = theContext;
        this.mNumber = null;
    }



    /**
     *
     *
     * @throws ExecutionException
     * @throws InterruptedException
     *
     * @info
     * Πατώντας το κουμπί επικοινωνία μπορεί να επιλεξει sms/κλήση
     *
     * @details
     * Αρχικά λαμβάνουμε τον αριθμό του οδηγού της παραγγελίας , έπειτα εμφανίζεται ένα
dialog box
     * για να επιλέξει ο χρήστης τον τρόπο επικοινωνίας (κλήση/ sms)
     *
     */

    public void makeContact() throws InterruptedException, ExecutionException {

         if ( new ConnectionDetectorTask(actContext).execute().get().get("connectionState")
) {

             SharedPreferences pref = actContext.getSharedPreferences("myOrder",
Context.MODE_PRIVATE);
             setmNumber( new GetCellphoneTask ("customer",
                        pref.getString("selectedDriverDevId", "").toString() )
                        .execute().get()
                        );

             final CharSequence[] items = {"Κλήση", "SMS", "Επιστροφή"};
             AlertDialog.Builder builder = new AlertDialog.Builder(actContext);

             builder.setTitle("Επιλέξτε τρόπο επικοινωνίας");
             builder.setItems(items, new DialogInterface.OnClickListener() {

                 public void onClick(DialogInterface dialog, int item) {
```

```
                    if (items[item].equals("Κλήση")) {
                       actContext.startActivity(new Intent(Intent.ACTION_CALL,
Uri.parse("tel:"+getmNumber())));

                    }
                    else if (items[item].equals("SMS")) {
                       Intent smsIntent = new Intent(Intent.ACTION_VIEW);
                       smsIntent.putExtra("sms_body", "Taxi anytime!");
                       smsIntent.putExtra("address", getmNumber());
                       smsIntent.setType("vnd.android-dir/mms-sms");

                    actContext.startActivity(smsIntent);
                        //startActivity(new Intent(Intent.ACTION_VIEW,
Uri.fromParts("sms:"+getmNumber(), null)));
                    }
                    else
                        Toast.makeText(actContext, items[item], Toast.LENGTH_SHORT).show();
                 }

                });

                AlertDialog alert = builder.create();

                alert.show();

         }//end if

  }


    public void setmNumber(String mNumber) {
        this.mNumber = mNumber;
    }

    public String getmNumber() {
        return mNumber;

    }


//=============================================================================================
=

    /**
     * @info
     * Μέθοδος για την προσκόμιση του κινητού τηλεφώνου τον συγκεριμένο οδηγού/πελάτη
     *
     * @details
     * Η κλάση "τρέχει " στο παρασκήνιο και επικοινωνεί με τη βάση δεδομένων για την
προσκόμιση
     * του αριθμού κινητου τηλεφώνου
     */


 protected class GetCellphoneTask extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters = new  ArrayList<NameValuePair>() ;
        private JSONObject myjobg = null;;
        private httpJSONParser json = new httpJSONParser();
        protected ProgressDialog pDialog;
        private String usrType;
        private String driverDevId;

        public GetCellphoneTask () {

            this.usrType = null;
            this.driverDevId = null;

        }
        public GetCellphoneTask (String userType,String drvDevId) {
            //  Auto-generated constructor stub
            this.usrType = userType;
            this.driverDevId = drvDevId;
        }
```

```
        /**
         * @info
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη.
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη.
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(actContext);
            pDialog.setMessage("Loading..");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα (ο αριθμός τηλεφώνου)
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα (ο αριθμός τηλεφώνου)
         *
         * */
        protected String doInBackground(String... args) {

            parameters.add(new BasicNameValuePair("client",this.usrType));
            parameters.add(new BasicNameValuePair("Deviceid",this.driverDevId ));

            try {
                myjobg = json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()
+"/scripts/makeCall.php","GET",parameters);

                if (Integer.parseInt(myjobg.get("success").toString()) == 1){
                    return (myjobg.get("cellphone").toString() );

                }
                else{

                    return ("");
                }

            } catch (NumberFormatException e) {
                // Auto-generated catch block
                e.printStackTrace();
            } catch (JSONException e) {
                //  Auto-generated catch block
                e.printStackTrace();
            }

            return ("");

        }



        /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * **/
        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done
                pDialog.dismiss();
                pDialog = null;

        }

    }

}//end class file
```

ΑΡΧΕΙΟ CustomerSplashScreen.java

```
package app.taxiAnytimeCustomer.Common;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ImageView;
import app.taxiAnytimeCustomer.R;


/**
 * @info
 * Εισαγωγική κλάση, χρησιμοποιείται πρώτη για να καλωσορίσει τον χρήστη στην εφαρμογή.
 */
public class CustomerSplashScreen extends Activity {

    private ImageView imageView;
    private int activityTimeOut = 2000; // Χρόνος σε ms



     /**
     * @info
     * Αρχικοποίηση activity
     *
     * @details
     * Αντιστοιχούμε την όψη του xml αρχείου με το αντικείμενο imageview , ώστε να μπορούμε
να το διαχειριστούμε.
     * Έπειτα ελέγχουμε για το αν υπάρχει ενεργή σύνδεση στο internet(είτε wifi είτε 3g),
καθώς και αν είναι
     * ανοικτό το gps
     */

      @Override
      public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash_screen);

        setImageView((ImageView)findViewById(R.id.splashImage));



            try {
             if ( new ConnectionDetectorTask(this).execute().get().get("connectionState")
){
                 startMainActivity();
             }

            } catch (Exception e) {
                e.printStackTrace();
                new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε αργότερα");
            }

    }

//============================================================

/**
 * @info
 * Μετά από κάποιο timeout (πχ 3 sec) ξεκινά η βασική δραστηριότητα login,
 *  και τερματίζεται η τρέχουσα(splash screen)
 */
private void startMainActivity(){

    Handler handler = new Handler();
    // run a thread after N seconds to start the home screen
    handler.postDelayed(new Runnable() {
```

```
        @Override
        public void run() {


            finish();
            // start the home screen
              Intent intent = new Intent(CustomerSplashScreen.this, LoginActivity.class);
              startActivity(intent);

        }

    }, activityTimeOut);
}

public ImageView getImageView() {
    return imageView;
}

public void setImageView(ImageView imageView) {
    this.imageView = imageView;
}


}
```

APXEIO EditProfileActivity.java

```
package app.taxiAnytimeCustomer.Common;

import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;


public class EditProfileActivity extends Activity {

    private EditText txtCellphone,
    txtPassword,txtRepatPassword,txtTown;

    protected SharedPreferences prefs ;
    private static final String NAME_PATTERN      = "^[a-zA-Zα-ωΑ-Ω]+$";
    private static final String CELLPHONE_PATTERN = "^[0-9]+$";

    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";
```

```
      /**
       * @info
       * Αρχικοποίηση menu buttons
       *
       * @details
       * Αρχικοποίηση menu buttons
       */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.layout.menu_items, menu);
        return true;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);
        setContentView(R.layout.edit_profile);

        txtCellphone = (EditText)findViewById(R.id.txtCellphone);
        txtTown = (EditText)findViewById(R.id.txtTown);
        txtPassword = (EditText)findViewById(R.id.txtPassword);
        txtRepatPassword = (EditText)findViewById(R.id.txtRepatPassword);


        prefs = getSharedPreferences("fromCustomer", MODE_PRIVATE);


        new fetchDataToEditTask( prefs.getString("customerdevid","") ).execute();

    }

    @Override
    public void onBackPressed() {

        finish();

        super.onBackPressed();
    }

//=========================================================================
protected void jsonToTextEdits(JSONObject jo) throws JSONException{

    if(jo != null){


txtCellphone.setText(jo.getJSONArray("editDetails").getJSONObject(0).get("cellphone").toStri
ng() );

txtTown.setText(jo.getJSONArray("editDetails").getJSONObject(0).get("town").toString() );

    }

}
//=============================================================================

    /**
     * @info
     * Έλεγχος ορθότητας δεδομένων για αλλαγή στοιχείων στην εφαρμογή
     *
     * @details
     * Με χρήση κανονικών εκφράσεων (regex) γίνετε ελεγχος ορθότητας δεδομένων.
     *
     * -Το όνομα πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα.
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     * -Το επώνυμο πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο
γράμματα.
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     *  -Το κινητό πρέπει να ειναι τουλάχιστον απο 10 αριθμους. Μόνο αριθμούς.
     *  κανονική έκφραση: ^[0-9]+$
     *
     * -Η πόλη πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα .
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     *
     *
     *  -Ο κωδικός πρέπει να ειναι τουλάχιστον 6 ψηφία
     *
```

```
     *
     *
     * @return
     * true : έχει δώσει σωστά στοιχεία.
     * false: δεν έχει δώσει σωστά στοιχεία.
     */
    private boolean isValidFormToEdit() {


        if (! txtCellphone.getText().toString().matches(CELLPHONE_PATTERN) ||
txtCellphone.getText().length() < 10)  {
            txtCellphone.setError("Το κινητό πρέπει να αποτελείτε απο αριθμούς");
            return false;
        }
        if (! txtTown.getText().toString().matches(NAME_PATTERN) ||
txtTown.getText().length() < 3) {
            txtTown.setError("Δώστε μια έγκυρη πόλη");
            return false;
        }

        if (txtPassword.getText().length() < 6) {
            txtPassword.setError("Ο κωδικός πρέπει να είναι τουλάχιστον 6 χαρακτήρες");
            return false;
        }

        if (! txtPassword.getText().toString().equals(txtRepatPassword.getText().toString())
) {
            txtRepatPassword.setError("Δεν ειναι ίδιοι οι κωδικοί πρόσβασης)");
            return false;
        }

        return true;

    }
//=============================================================================================

    /**
     * @param fields  Array με τα πεδία edit
     * @info
     * Ελέγχει αν όλα τα editboxes είναι κενά
     * @return
     * true : Αν έστω και ένα πεδίο είναι άδειο
     * false : Αν δεν είναι κανένα άδειο
     */
    private boolean AreEditBoxesEmpty(EditText[] fields){
        for(int i=0; i<fields.length; i++){
            EditText currentField=fields[i];
            if(currentField.getText().toString().length()<=0){
                return true;
            }
        }
        return false;
}


//=============================================================================
    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Αρχίζει την διαδικασία του edit
     *
     * @details
     * Κάνει έλεγχο αν έχουμε gps και σύνδεση στο internet ανοικτά και αν ναι, ξεκινάει την
     * διαδικασία της αλλαγής στοιχείων. Αλλιώς ενημερώνει την χρήστη
     *
     */
    public void onClkEdit(View view) {



        if(isValidFormToEdit() == true &&
                AreEditBoxesEmpty(new EditText[]{txtCellphone,
                txtPassword,txtRepatPassword,txtTown}) == false ) {
```

```
            try{

                if ( new ConnectionDetectorTask(this).execute().get().get("connectionState")
){
                    new makeEditTask().execute();
                }

            }
            catch(Exception e){
                e.printStackTrace();
                new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε αργότερα");
            }

        }
        else{
            new showAlertMessage(this ,"Προσοχή!!!","Ελέγξτε όλα τα πεδία");
        }

}

//=========================================================================
     /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Καθαρίζει όλα τα πεδία της φόρμας.
     *
     * @details
     * Καθαρίζει όλα τα πεδία της φόρμας.
     */
    public void onClkClearFields(View view){



            txtCellphone.setText("");
            txtTown.setText("");
            txtPassword.setText("");
            txtRepatPassword.setText("");
     }

//=========================================================================

class fetchDataToEditTask extends AsyncTask<String, String, String> {

        /**
         * @info
         * Κλάση υπεύθυνη για την διαδικασία της εγγραφής χρηστών
         */
        private ArrayList<NameValuePair> parameters ;
        private String deviceID;
        private ProgressDialog pDialog;
        protected JSONObject myjobg ;


        public fetchDataToEditTask(String cstID){

            deviceID = cstID;
            }

                /**
                 * @info
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 *
                 * @details
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 * */

         @Override
            protected void onPreExecute() {
                   pDialog = new ProgressDialog(EditProfileActivity.this);
                   pDialog.setIndeterminate(false);
                   pDialog.setMessage("Αλλαγή στοιχείων...");
                   pDialog.setCancelable(true);
                   pDialog.show();

            }

                /**
                 * @info
```

```
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * @details
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * */

            @Override
            protected String doInBackground(String... urls) {

                    parameters = new ArrayList<NameValuePair>();

                    try {

                        parameters.add(new BasicNameValuePair("deviceid",deviceID) );
                        parameters.add(new BasicNameValuePair("usertype","customer") );


                        myjobg = null ;
                        httpJSONParser json = new httpJSONParser();
                        myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/fetchDataToEdit.
php","POST",parameters);

                        //Log.d("editdetails",myjobg.toString() );
                        if(Integer.parseInt(myjobg.get("success").toString()) == 1) {

                            //do something


                        }

                    } catch (Exception e) {

                        e.printStackTrace();
                    }

                    return null;
                }

                /**
                 * @info
                 *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα
                 *
                 * @details
                 * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα και μεταβαίνουμε στην login δραστηριότητα.
                 *
                 * **/
                protected void onPostExecute(String file_url) {
                    // dismiss the dialog once done

                    pDialog.dismiss();
                    pDialog = null;

                    try {
                        jsonToTextEdits ( myjobg );
                    } catch (JSONException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }


                }
        }

//=========================================================================
```

```
class makeEditTask extends AsyncTask<String, String, String> {

        /**
         * @info
         * Κλάση υπεύθυνη για την διαδικασία της εγγραφής χρηστών
         */
        private ArrayList<NameValuePair> parameters ;
        private ProgressDialog pDialog;
        private JSONObject myjobg ;
        private httpJSONParser json ;
        private boolean check;



                /**
                 * @info
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 *
                 * @details
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 * */

                @Override
                protected void onPreExecute() {
                    super.onPreExecute();
                    pDialog = new ProgressDialog(EditProfileActivity.this);
                    pDialog.setMessage("Άλλαγή στοιχείων...");
                    pDialog.setIndeterminate(false);
                    pDialog.setCancelable(true);
                    pDialog.show();
                }


                /**
                 * @info
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * @details
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * */

                protected String doInBackground(String... args) {

                try {
                    parameters = new ArrayList<NameValuePair>();

                    parameters.add(new
BasicNameValuePair("cellphone",String.valueOf(txtCellphone.getText())));
                    parameters.add(new
BasicNameValuePair("password",String.valueOf(txtPassword.getText())));
                    parameters.add(new
BasicNameValuePair("town",String.valueOf(txtTown.getText())));

                    parameters.add(new BasicNameValuePair("deviceid",
prefs.getString("customerdevid","") ) );
                    parameters.add(new BasicNameValuePair("usertype","customer") );

                    myjobg = null;
                    json = new httpJSONParser();
                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeEdit.php","P
OST",parameters);

                        if(Integer.parseInt(myjobg.get("success").toString()) == 1) {

                            check = true;

                        }
                        else{
                            check = false;
                        }

                    } catch (NumberFormatException e) {
```

```
                    e.printStackTrace();
                } catch (JSONException e) {

                    e.printStackTrace();
                }
                return null;
            }
            /**
             * @info
             *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα
             *
             * @details
             * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα και μεταβαίνουμε στην login δραστηριότητα.
             *
             * **/
            protected void onPostExecute(String file_url) {
                pDialog.dismiss();
                pDialog = null;

                if(this.check == true){
                    Toast toast = Toast.makeText(getApplicationContext(),"Ολοκληρώθηκαν
οι αλλαγές!", Toast.LENGTH_LONG);
                    toast.show();
                }
                else {
                    Toast toast = Toast.makeText(getApplicationContext(),"Μη επιτυχής
αλλαγή στοιχείων!!!", Toast.LENGTH_LONG);
                    toast.show();
                }
            }
        }
//=====================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
 * @info
 * Κώδικας σχετικά με το hardware button menu
 *
 *
 * @details
 * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
 * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
 * Έχει τις δυνατότητες bookmark, share & profile.
 *
 * */
@Override
public boolean onOptionsItemSelected(MenuItem item){
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
    switch (item.getItemId()) {

    case R.id.menu_contact:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_report:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;

    case R.id.menu_bookmark:
        saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
        return true;
    case R.id.menu_share:
        share();
        return true;
    case R.id.menu_profile:
        Toast.makeText(this, "Είστε ήδη εδώ!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_aboutTheApp:
        aboutTheApp();
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }

}
```

```
/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

      Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
      startActivity(intent);
}

/**
 * @param c το περιεχόμενο της δραστηριότητας.
 * @param title ο τίτλος του σελιδοδείκτη.
 * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
 *
 * @info
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 *
 * @details
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 */
public static final void saveBookmark(Context c, String title, String url) {
    Intent i = new Intent(Intent.ACTION_INSERT,
                  android.provider.Browser.BOOKMARKS_URI);
    i.putExtra("title", title);
    i.putExtra("url", url);
    c.startActivity(i);
}

/**
 * @info
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
 * με την εφαρμογή
 *
 * @details
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
 * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
 * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
 */
public void share () {
    Intent share = new Intent(Intent.ACTION_SEND);
    String shareBody = "Taxi Anytime";
    share.setType("text/plain");
    share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
    share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    startActivity(Intent.createChooser(share, "Send via"));

}

/**
 * @info
 * Πληροφορίες σχετικά με την εφαρμογή.
 *
 * @details
 * Πληροφορίες σχετικά με την εφαρμογή.
 */
public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

}
```

```
/**
 * @info
 * Αλλαγή προφίλ
 *
 * @details
 * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
 */
public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);
}

}
```

APXEIO globalVariables.java

```
package app.taxiAnytimeCustomer.Common;


/** @info
 * Σε αυτο το αρχειο γινετε αρχικοποιηση των global μεταβλητων.
 * Γινετε χρηση του Singleton pattern για να εχουμε μοναδικη αναφορα στην
 * συγκεκριμενη κλαση
 *
 */
public class globalVariables {

        //MQTT BROKER
        private String MQTT_IP_ADDRESS ;
        private int MQTT_PORT;

        //WAMP SERVER
        private    String IP_ADDRESS ;
        private    int HTTP_PORT;
        private    String BASE_PATH;


        //instance
        private static globalVariables instance = null;
        public globalVariables() {

            MQTT_IP_ADDRESS = "192.168.2.100";
            MQTT_PORT = 1883;

            IP_ADDRESS = "192.168.2.100";
            HTTP_PORT = 80;
            BASE_PATH =
"http://"+IP_ADDRESS+":"+String.valueOf(HTTP_PORT)+"/taxiAnytime_server";

        }

        public static globalVariables getInstance() {
            if (instance == null)
                instance = new globalVariables();
            return instance;
        }

        public String getBASE_PATH() {
            return BASE_PATH;
        }

        public int getHTTP_PORT() {
            return HTTP_PORT;
        }
        public String getIP_ADDRESS() {
            return IP_ADDRESS;
        }
        public String getMQTT_IP_ADDRESS() {
            return MQTT_IP_ADDRESS;
        }
        public int getMQTT_PORT() {
            return MQTT_PORT;
        }


}
```

ΑΡΧΕΙΟ httpJSONParser.java

```
package app.taxiAnytimeCustomer.Common;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;



/**
 *@info
 *  Κλάση για τη διαχείριση http post/get στη βάση mysql (Σε μορφή json)
 *
 * Στην ουσία μετατρέπει την απόκριση json από ένα http request που γίνεται ,
 * σε αντικείμενο JSON σε java ώστε να μπορούμε να το επεξεργαστούμε.
 *
 *@external_source
 *Αυτή η κλάση χρησιμοποιήτε ευρέως από όλους όσους γράφουν εφαρμογές android
 *και είναι για την επικοινωνία με την βάση δεδομένων
 *
 *πηγή: www.androidhive.info
 */
public class httpJSONParser {

    InputStream is ;
    JSONObject jObj ;
    String json ;
    final int BUFFER_SIZE = 8192;


    public httpJSONParser() {
        is = null;
        jObj = null;
        json = null;

    }

    // function get json from url
    // by making HTTP POST or GET mehtod
    /**
     * @param url η διεύθυνση των php scripts μέσα στον server μας, τα οποία εμείς γράψαμε.
     * @param method μέθοδος επικοινωνίας με τον server (post ή get)
     * @param params οι είσοδοι στα php scripts μας, οι παραμέτροί τους.
     *
     * @info
     * Γράφει και διαβάζει απο την βάση δεδομένων
     *
     * @details
     * Αφού γίνει η σύνδεση με την βάση γίνετε έλεγχος τι θέλουμε να κάνουμε.
     * Αν η μέθοδος ειναι post, στέλνουμε και αποθηκεύουμε δεδομένα στην βάση.
     * Αν η μέθοδος ειναι get, λαμβάνουμε αποτελέσματα από την βάση και τα επιστρέφουμε.
     *
     * @return
     * JSONObject : πρακτικά είναι το αποτέλεσμα των php scripts σε μορφή JSON
     */
```

```
    public JSONObject makeHttpRequest(String url, String method,List<NameValuePair> params)
{

        // Making HTTP request
        try {

            // check for request method
            if(method == "POST"){
                // request method is POST
                // defaultHttpClient
                DefaultHttpClient httpClient = new DefaultHttpClient();

                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new UrlEncodedFormEntity(params,"UTF-8"));

                HttpResponse httpResponse = httpClient.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }
            else if(method == "GET"){

                // request method is GET
                DefaultHttpClient httpClient = new DefaultHttpClient();
                String paramStringGet = URLEncodedUtils.format(params, "UTF-8");
                url += "?" + paramStringGet;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse = httpClient.execute(httpGet);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();
            }


        }
        catch (UnsupportedEncodingException e) {
            e.printStackTrace();

        }
        catch (ClientProtocolException e){
            e.printStackTrace();
        }
        catch (IOException e){
            e.printStackTrace();
        }

        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(is,"UTF-
8"),BUFFER_SIZE);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();
            json = sb.toString();
        }
        catch (Exception e){
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }

        // try parse the string to a JSON object
        try{
            jObj = new JSONObject(json);
        }
        catch (JSONException e){
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }

        // return JSON String
        return jObj;

    }
}
```

ΑΡΧΕΙΟ LoginActivity.java

```
package app.taxiAnytimeCustomer.Common;


import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.Customer.CustomerActivity;
import app.taxiAnytimeCustomer.userTypesFactory.Users;
import app.taxiAnytimeCustomer.userTypesFactory.UsersFactory;




/**
 * @info
 * Είσοδος Χρήστη
 * Ελέγχεται η είσοδος του χρήστη με σύστημα login,
 * και αν είναι επιτυχής συνεχίζουμε στην εφαρμογή.
 *
 */
public class LoginActivity extends Activity {

    private EditText txtUsername,txtPassword;
    private CheckBox saveLoginCheckBox;
    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";


    private SharedPreferences loginPreferences;
    //Ο editor ειναι για να μπορουμε να τροποποιησουμε τα δεδομενα που εχουμε αποθηκευμενα.
    private SharedPreferences.Editor loginPrefsEditor;
    public static final String PREFERENCE_FILENAME = "LoginInfo";

    private Boolean saveLogin;
    private String username,password;
```

```
     /**
      * @info
      * Αρχικοποίηση menu buttons
      *
      * @details
      * Αρχικοποίηση menu buttons
      */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.layout.menu_items, menu);
        return true;
    }


    /**
     * @info
     * Αρχικοποίηση activity
     *
     * @details
     * Αντιστοιχούμε τις όψεις των xml αρχείων με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
     * Έπειτα χρησιμοποιούμε τον εσωτερικό μηχανισμό του android (SharedPreferences) για την
,
     * αποθήκευση στοιχείων(login) στο αρχείο ώστε να είναι προσβάσιμο στην εφαρμογή.
     *
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login_layout);

        txtUsername = (EditText)findViewById(R.id.editTextUsername);
        txtPassword = (EditText)findViewById(R.id.editTextPassword);
        saveLoginCheckBox = (CheckBox)findViewById(R.id.saveLoginCheckBox);



        loginPreferences = getSharedPreferences(PREFERENCE_FILENAME, MODE_PRIVATE);
        //Βαζουμε τον editor να μπορει να βλεπει για να κανει επεξεργασια -το συγκεκριμενο-
preference
        loginPrefsEditor = loginPreferences.edit();

        saveLogin = loginPreferences.getBoolean("saveLogin", false);
        if (saveLogin == true) {
            txtUsername.setText(loginPreferences.getString("username", ""));
            txtPassword.setText(loginPreferences.getString("password", ""));
            saveLoginCheckBox.setChecked(true);
        }
    }



    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @throws InterruptedException
     * @throws ExecutionException
     *
     * @info
     * Γίνεται η ορθή διαδικασία εισαγωγής χρήστη στην εφαρμογή
     *
     * @details
     * Οταν πατησει το κουμπι login ο χρηστης γινετε ελεγχος των στοιχειων που εδωσε
     * και αν αυτα ειναι σωστα τον βαζει στην εφαρμογη. Σε διαφορετικη περιπτωση
     * εμφανιζει μηνυμα λαθους.Επίσης γίνεται έλεγχος για τον αν υπάρχει σύνδεση της
συσκευής
     * τόσο με το internet όσο και με τον server πριν γίνει οτιδήποτε
     *
     */
    public void onClkLogin(View view) throws InterruptedException, ExecutionException  {

            username = txtUsername.getText().toString();
            password = txtPassword.getText().toString();

           if( !fieldsAreEmpty(username,password) ){
```

```
            try{
                 if ( new
ConnectionDetectorTask(this).execute().get().get("connectionState") ){
                     new checkLoginTask(username,password).execute();
                 }

            }
            catch(Exception e){
                new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε αργότερα");
                e.printStackTrace();
            }

         }

    }

    private boolean fieldsAreEmpty(String username,String password){

        if(username == null || password == null){
            return true;
        }
        else{
            return false;
        }



    }


  //================================================================================

    /**
     * @param loginComplete μας δείχνει αν το Login έχει γίνει επιτυχώς
     *
     * @info Ελέγχουμε αν θέλει να αποθηκευτεί το login και αν είναι σωστό γίνεται μετάβαση
στην επόμενη activity
     *
     * @details
     * Αν έχει ολοκληρωθεί το Login επιυχώς και ο χρήστης έχει επιλέξει την "remember me"
επιλογή
     * γράφουμε τα στοιχεία του στο SharedPreferences αρχείο ώστε να τα τραβήξουμε την
επόμενη
     * φορά που θα τρέξει η εφαρμογή
     */
    public void saveLoginData(boolean loginComplete){

        if(loginComplete == true) {

            if (saveLoginCheckBox.isChecked()) {
                loginPrefsEditor.putBoolean("saveLogin", true);
                loginPrefsEditor.putString("username", username);
                loginPrefsEditor.putString("password", password);
                loginPrefsEditor.commit();
            }
            else{
                loginPrefsEditor.clear();
                loginPrefsEditor.commit();
            }

            finish();
            Intent i = new Intent(getApplicationContext(),CustomerActivity.class);
            startActivity(i);
            }
        else { //(loginComplete == false)
            new showAlertMessage(this ,"Σφάλμα εισόδου!!!","Δώσατε λάθος στοιχεία ή είστε
μπλοκαρισμένος!");
         }


    }


  //================================================================================
```

```
    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Γίνετε μετάβαση την δραστηριότητα του register
     *
     * @details
     * Γίνετε μετάβαση την δραστηριότητα του register
     */
    public void onClkGoToRegister(View view){
        finish();
        Intent i = new Intent(getApplicationContext(),RegisterCustomerActivity.class);
        startActivity(i);

    }
//=====================================================================
    /**
     * @info
     * Η κλάση αυτή είναι υπεύθυνη για τον έλεγχο ορθότητας των δεδομένων που
     * έδωσε ο χρήστης.
     *
     */
    class checkLoginTask extends AsyncTask<String, Void, String> {

        private ProgressDialog Dialog;

        private String Username;
        private String Password;
        private boolean loginState;


        public checkLoginTask() {
            Username = null;
            Password = null;
            loginState = false;

        }

        public checkLoginTask(String usrname,String pass) {
            // Auto-generated constructor stub
            Username = usrname;
            Password = pass;

        }

        /**
         * @info
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
         */
        @Override
        protected void onPreExecute() {
               Dialog = new ProgressDialog(LoginActivity.this);
               Dialog.setIndeterminate(false);
               Dialog.setMessage("Είσοδος...");
               Dialog.setCancelable(true);
               Dialog.show();

        }

        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφεται true ή false
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφεται true ή false.
         *
         * @return
         * true : ο χρήστης έδωσε σωστά δεδομένα
         * false: ο χρήστης δεν έδωσε σωστά δεδομένα
         */
```

```
        protected String doInBackground(String... urls) {

            ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>(4);

            Users customer = UsersFactory.createUser("customer");

            parameters.add(new BasicNameValuePair("username",Username ));
            parameters.add(new BasicNameValuePair("password",Password ));
            parameters.add(new BasicNameValuePair("deviceid","/"+customer.getDeviceID() ));
            parameters.add(new BasicNameValuePair("type","customer"));

            JSONObject myjobg = null;
            httpJSONParser json = new httpJSONParser();

             try {
                myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeLogin.php","
POST",parameters );


                     if(Integer.parseInt(myjobg.get("success").toString() ) == 1) {
                        loginState = true;
                    }
                    else {
                        loginState = false;
                    }
                }
                catch (Exception e) {
                    e.printStackTrace();
                    loginState = false;
                }

                return null;
        }

        /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * **/
        @Override
        protected void onPostExecute(String result) {
          Dialog.dismiss();
          Dialog = null;
          saveLoginData(loginState);
        }

 }


  //======================================================================
  //menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
   * @info
   * Κώδικας σχετικά με το hardware button menu
   *
   *
   * @details
   * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
   * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
   * Έχει τις δυνατότητες bookmark, share & profile.
   *
   * */
  @Override
  public boolean onOptionsItemSelected(MenuItem item){
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
      switch (item.getItemId()){

      case R.id.menu_contact:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;
```

```
    case R.id.menu_report:

      Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
      return true;

    case R.id.menu_bookmark:
        saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
        return true;
    case R.id.menu_share:
      share();
        return true;
    case R.id.menu_profile:
      Toast.makeText(this, "Πρέπει να κάνετε login πρώτα!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_aboutTheApp:
      aboutTheApp();
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }

}

/**
 * @info
 * Μεταβαίνουμε στην δραστηριότητα της κλήσης/sms
 */
public void call(){
    Intent intent = new Intent(getApplicationContext(),Contact.class);
    startActivity(intent);
}

/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

    Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
    startActivity(intent);
}

/**
 * @param c το περιεχόμενο της δραστηριότητας.
 * @param title ο τίτλος του σελιδοδείκτη.
 * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
 *
 * @info
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 *
 * @details
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 */
public static final void saveBookmark(Context c, String title, String url) {
    Intent i = new Intent(Intent.ACTION_INSERT,
                  android.provider.Browser.BOOKMARKS_URI);
    i.putExtra("title", title);
    i.putExtra("url", url);
    c.startActivity(i);
}

/**
 * @info
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
 * με την εφαρμογή
 *
 * @details
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
 * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί είναι καθαρά θέμα τι εφαρμογές
 * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
 */
public void share () {
  Intent share = new Intent(Intent.ACTION_SEND);
  String shareBody = "Taxi Anytime";
  share.setType("text/plain");
  share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
  share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
  startActivity(Intent.createChooser(share, "Send via"));

}
```

```
  /**
   * @info
   * Πληροφορίες σχετικά με την εφαρμογή.
   *
   * @details
   * Πληροφορίες σχετικά με την εφαρμογή.
   */
  public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

  }


  /**
   * @info
   * Αλλαγή προφίλ
   *
   * @details
   * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
   */
  public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);
  }


}
```

APXEIO RegisterCustomerActivity.java

```
package app.taxiAnytimeCustomer.Common;

import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.userTypesFactory.Users;
import app.taxiAnytimeCustomer.userTypesFactory.UsersFactory;
```

```
/**
 * @info
 * Κλάση υπεύθυνη για ότι έχει να κάνει με το register νέων χρηστών.
 *
 */
public class RegisterCustomerActivity extends Activity {

    private EditText txtName,txtSirName,txtCellphone,txtBirthday,
                     txtUsername,txtPassword,txtRepatPassword,txtEmail,
                     txtTown;

    private static final String NAME_PATTERN      = "^[a-zA-Zα-ωΑ-Ω]+$";
    private static final String CELLPHONE_PATTERN = "^[0-9]+$";
    private static final String USERNAME_PATTERN  = "^[a-zA-Z][[a-zA-Z][0-9]]*$";
    private static final String EMAIL_PATTERN     = "^[a-z0-9_\\+-]+(\\.[a-z0-9_\\+-]+)*@[a-
z0-9-]+(\\.[a-z0-9-]+)*\\.([a-z]{2,4})$";

    private final String BOOKMARK_TITLE = "Taxi Anytime";
    private final String BOOKMARK_URL ="www.taxianytime.hostzi.com";



     /**
  * @info
  * Αρχικοποίηση menu buttons
  *
  * @details
  * Αρχικοποίηση menu buttons
  */
 @Override
 public boolean onCreateOptionsMenu(Menu menu){
     MenuInflater menuInflater = getMenuInflater();
     menuInflater.inflate(R.layout.menu_items, menu);
     return true;
 }


    /**
     * @info
     * Αρχικοποίηση activity
     *
     * @details
     * Αντιστοιχούμε τις όψεις των xml αρχείων με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
     *
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.register);


        txtName = (EditText)findViewById(R.id.txtName);
        txtSirName = (EditText)findViewById(R.id.txtSirName);
        txtCellphone = (EditText)findViewById(R.id.txtCellphone);
        txtBirthday = (EditText)findViewById(R.id.txtBirthday);
        txtTown = (EditText)findViewById(R.id.txtTown);
        txtUsername = (EditText)findViewById(R.id.txtUsername);
        txtPassword = (EditText)findViewById(R.id.txtPassword);
        txtRepatPassword = (EditText)findViewById(R.id.txtRepatPassword);
        txtEmail = (EditText)findViewById(R.id.txtEmail);

    }

    /**
     * @info
     * Οταν πατήσει το hardware back (κουμπί συσκευής) τον επιστρέφει στο login
     *
     * @details
     * Οταν πατήσει το hardware back (κουμπί συσκευής) τον επιστρέφει στο login
     *
     */
    @Override
    public void onBackPressed() {

        finish();
        Intent i = new Intent(getApplicationContext(),LoginActivity.class);
        startActivity(i);
```

```
        super.onBackPressed();
    }

    /**
     * @info
     * Έλεγχος ορθότητας δεδομένων για εγγραφή στην εφαρμογή
     *
     * @details
     * Με χρήση κανονικών εκφράσεων (regex) γίνετε ελεγχος ορθότητας δεδομένων.
     *
     * -Το όνομα πρέπει να ειναι πάνω απο 2 χαρακτήρες και να αποτελείτε μονο απο γράμματα.
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     * -Το επώνυμο πρέπει να ειναι πάνω απο 2 χαρακτήρες και να αποτελείτε μονο απο
γράμματα.
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     *  -Το κινητό πρέπει να ειναι τουλάχιστον απο 10 αριθμους. Μόνο αριθμούς.
     *  κανονική έκφραση: ^[0-9]+$
     *
     * -Η πόλη πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα .
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     * -Το username πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αρχίζει με γράμμα ή κάτω
παύλα.
     *  κανονική έκφραση: ^[a-zA-Z][[a-zA-Z][0-9]]*$
     *
     *  -Ο κωδικός πρέπει να ειναι τουλάχιστον 6 ψηφία
     *
     *  -Το email πρεπει να ειναι τουλάχιστον 4 χαρακτήρες, να αρχίζει με γράμμα,αριθμό, ,+
ή -, πρέπει να υπάρχει
     *  το @ και η . και μετά την . να έχουμε απο 2 έως 4 χαρακτήρες
     *  κανονική έκφραση: ^[a-z0-9_\\+-]+(\\.[a-z0-9_\\+-]+)*@[a-z0-9-]+(\\.[a-z0-9-
]+)*\\.([a-z]{2,4})$
     *
     * @return
     * true : έχει δώσει σωστά στοιχεία.
     * false: δεν έχει δώσει σωστά στοιχεία.
     */
    private boolean isValidFormToRegister() {


        if (! txtName.getText().toString().matches(NAME_PATTERN) ||
txtName.getText().length() < 3)    {
            txtName.setError("Το όνομα πρέπει να αποτελείται απο γράμματα");
            return false;
        }

        if (! txtSirName.getText().toString().matches(NAME_PATTERN) ||
txtSirName.getText().length() < 3 ) {
            txtSirName.setError("Το επώνυμο πρέπει να αποτελείται απο γράμματα");
            return false;


        }

        if (! txtCellphone.getText().toString().matches(CELLPHONE_PATTERN) ||
txtCellphone.getText().length() < 10)  {
            txtCellphone.setError("Το κινητό πρέπει να αποτελείται απο αριθμούς");
            return false;

        }

        if (! txtTown.getText().toString().matches(NAME_PATTERN) ||
txtTown.getText().length() < 3) {
            txtTown.setError("Δώστε μια έγκυρη πόλη");
            return false;
        }

        if (! txtUsername.getText().toString().matches(USERNAME_PATTERN) ||
txtUsername.getText().length() < 3) {
            txtUsername.setError("Το username δεν είναι έγκυρο");
            return false;
        }

        if (txtPassword.getText().length() < 6) {
            txtPassword.setError("Ο κωδικός πρέπει να είναι τουλάχιστον 6 χαρακτήρες");
            return false;
        }
```

```
        if (! txtPassword.getText().toString().equals(txtRepatPassword.getText().toString())
) {
            txtRepatPassword.setError("Δεν ειναι ίδιοι οι κωδικοί πρόσβασης)");
            return false;
        }

        if(! txtEmail.getText().toString().matches(EMAIL_PATTERN) ||
txtEmail.getText().length() < 4)  {
            txtEmail.setError("Δώστε ενα έγκυρο email");
            return false;
        }


        return true;


    }


    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Αρχίζει την διαδικασία του register
     *
     * @details
     * Κάνει έλεγχο αν έχουμε gps και σύνδεση στο internet ανοικτά και αν ναι, ξεκινάει την
     * διαδικασία για το register. Αλλιώς ενημερώνει την χρήστη
     *
     */
    public void onClkRegister(View view) {

        if(isValidFormToRegister() == true) {

            try{
                 if ( new
ConnectionDetectorTask(this).execute().get().get("connectionState") ){
                     new makeRegisterTask().execute();
                 }
            }
            catch(Exception e){
                e.printStackTrace();
                new showAlertMessage(RegisterCustomerActivity.this ,"Ωχ! Κάτι πήγε
στραβά!!!","Δοκιμάστε αργότερα");
            }
        }

    }


     /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Καθαρίζει όλα τα πεδία της φόρμας.
     *
     * @details
     * Καθαρίζει όλα τα πεδία της φόρμας.
     */
    public void onClkClearFields(View view){

            txtName.setText("");
            txtSirName.setText("");
            txtCellphone.setText("");
            txtBirthday.setText("");
            txtTown.setText("");
            txtUsername.setText("");
            txtPassword.setText("");
            txtRepatPassword.setText("");
            txtEmail.setText("");
     }


//=============================================================================================
    /**
     * @info
     * Κλάση υπεύθυνη για την διαδικασία της εγγραφής χρηστών
     */

class makeRegisterTask extends AsyncTask<String, String, String> {
```

```
    private ArrayList<NameValuePair> parameters ;
    protected ProgressDialog pDialog;
    private JSONObject myjobg ;
    private httpJSONParser json ;
    private boolean check;

            public makeRegisterTask() {
                //Auto-generated constructor stub

                parameters = new ArrayList<NameValuePair>();
                myjobg = null;
                json = new httpJSONParser();
            }



            /**
             * @info
             * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
             *
             * @details
             * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
             * */

            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(RegisterCustomerActivity.this);
                pDialog.setMessage("Εγγραφή..");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(true);
                pDialog.show();
            }


            /**
             * @info
             * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε τα
δεδομένα
             * του χρήστη για εγγραφή στον server.
             *
             * @details
             * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε τα
δεδομένα
             * του χρήστη για εγγραφή στον server.
             *
             * */
            @Override
            protected String doInBackground(String... args) {
                Users customer = UsersFactory.createUser("customer");

                parameters.add(new
BasicNameValuePair("name",String.valueOf(txtName.getText())));
                parameters.add(new
BasicNameValuePair("sirname",String.valueOf(txtSirName.getText())));
                parameters.add(new
BasicNameValuePair("cellphone",String.valueOf(txtCellphone.getText())));
                parameters.add(new
BasicNameValuePair("birthday",String.valueOf(txtBirthday.getText())));
                parameters.add(new
BasicNameValuePair("username",String.valueOf(txtUsername.getText())));
                parameters.add(new
BasicNameValuePair("password",String.valueOf(txtPassword.getText())));
                parameters.add(new
BasicNameValuePair("town",String.valueOf(txtTown.getText())));
                parameters.add(new
BasicNameValuePair("mail",String.valueOf(txtEmail.getText())));

                parameters.add(new BasicNameValuePair("deviceid","/"+customer.getDeviceID())
);
                parameters.add(new BasicNameValuePair("usertype","customer") );



                try {

                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeRegister.php
","POST",parameters);
```

```
                    if(Integer.parseInt(myjobg.get("success").toString()) == 1) {

                        //do something
                        check = true;
                    }
                    else {

                        check = false;
                    }
                } catch (NumberFormatException e) {
                    //  Auto-generated catch block
                    e.printStackTrace();
                } catch (JSONException e) {
                    //  Auto-generated catch block
                    e.printStackTrace();
                }

                return null;
            }

            /**
             * @info
             *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα
             *
             * @details
             * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα και μεταβαίνουμε στην login δραστηριότητα.
             *
             * **/
            @Override
            protected void onPostExecute(String file_url) {
                // dismiss the dialog once done

                pDialog.dismiss();
                pDialog = null;

                if(check == true){

                    Toast toast = Toast.makeText(RegisterCustomerActivity.this,"Επιτυχής
Εγγραφή!!!", Toast.LENGTH_SHORT);
                    toast.show();

                    finish();
                    Intent i = new Intent(getApplicationContext(),LoginActivity.class);
                    startActivity(i);
                }
                else{
                    new showAlertMessage(RegisterCustomerActivity.this ,"Ωχ! Δώσατε λάθος
στοιχεία ή υπάρχει ήδη!!","Δοκιμάστε αργότερα");
                }
            }

        }


//=====================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
 * @info
 * Κώδικας σχετικά με το hardware button menu
 *
 *
 * @details
 * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
 * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
 * Έχει τις δυνατότητες bookmark, share & profile.
 *
 * */
@Override
public boolean onOptionsItemSelected(MenuItem item){
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
    switch (item.getItemId()){

    case R.id.menu_contact:
```

```
            Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_report:

            Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
            return true;

        case R.id.menu_bookmark:
            saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
            return true;
        case R.id.menu_share:
            share();
            return true;
        case R.id.menu_profile:
            Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_aboutTheApp:
            aboutTheApp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }

    }

    /**
     * @info
     * Μεταβαίνουμε στην δραστηριότητα της κλήσης/sms
     */
    public void call(){
          Intent intent = new Intent(getApplicationContext(),Contact.class);
          startActivity(intent);
    }

    /**
    * @info
    * Μεταβαίνουμε στην δραστηριότητα της αναφοράς
    */
    public void report(){

          Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
          startActivity(intent);
    }

    /**
     * @param c το περιεχόμενο της δραστηριότητας.
     * @param title ο τίτλος του σελιδοδείκτη.
     * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
     *
     * @info
     * Κάνει σελιδοδείκτη το site της εφαρμογής.
     *
     * @details
     * Κάνει σελιδοδείκτη το site της εφαρμογής.
     */
    public static final void saveBookmark(Context c, String title, String url) {
        Intent i = new Intent(Intent.ACTION_INSERT,
                      android.provider.Browser.BOOKMARKS_URI);
        i.putExtra("title", title);
        i.putExtra("url", url);
        c.startActivity(i);
    }

    /**
     * @info
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή
     *
     * @details
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
     * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
     */
    public void share () {
        Intent share = new Intent(Intent.ACTION_SEND);
        String shareBody = "Taxi Anytime";
        share.setType("text/plain");
        share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
        share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
        startActivity(Intent.createChooser(share, "Send via"));
```

```
}

/**
 * @info
 * Πληροφορίες σχετικά με την εφαρμογή.
 *
 * @details
 * Πληροφορίες σχετικά με την εφαρμογή.
 */
public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

}

/**
 * @info
 * Αλλαγή προφίλ
 *
 * @details
 * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
 */
public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);
}



}
```

APXEIO ReportActivity.java

```
package app.taxiAnytimeCustomer.Common;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnCancelListener;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
```

```
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;









/**
 * @info
 * Κλάση υπεύθυνη για την αναφορά οδηγού απο τον πελάτη.
 *
 */
public class ReportActivity extends Activity implements OnItemSelectedListener {

     private Spinner spinner ;
     private EditText edtReason;
     private ImageButton imgBtnReport;

     private String RptReasons ;
     private String totalReasons;
     private ProgressDialog pDialog;
     private final String BOOKMARK_TITLE = "Taxi Anytime";
     private final String BOOKMARK_URL ="www.taxianytime.hostzi.com";



     /**
   * @info
   * Αρχικοποίηση menu buttons
   *
   * @details
   * Αρχικοποίηση menu buttons
   */
  @Override
  public boolean onCreateOptionsMenu(Menu menu){
      MenuInflater menuInflater = getMenuInflater();
      menuInflater.inflate(R.layout.menu_items, menu);
      return true;
  }

     /**
    * @info
    * Αρχικοποίηση activity
    *
    * @details
    * Αρχικοποίηση activity
    */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.report);

         spinner = (Spinner) findViewById(R.id.ReportReasons);
         edtReason=(EditText)findViewById(R.id.comment);
         imgBtnReport = (ImageButton)findViewById(R.id.imageButtonSubmitReport);
         imgBtnReport.setEnabled(true);
         edtReason.setText(" ");


     // Create an ArrayAdapter using the string array and a default spinner layout

     ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
     R.array.reportReasons, android.R.layout.simple_spinner_item);
     // Specify the layout to use when the list of choices appears
     adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
     // Apply the adapter to the spinner
     spinner.setAdapter(adapter);
     spinner.setOnItemSelectedListener(this);

    }


    /**
```

```
     * @param v το View για το button
     * @throws ExecutionException
     * @throws InterruptedException
     *
     * @info
     * Κάνει την αναφορά του οδηγού και την καταγράφει στην βάση.
     *
     * @details
     * Αρχικά κάνει έλεγχο αν υπάρχει σύνδεση με το δίκτυο και αν μπορεί να συνδεθεί
     * στον server της εφαρμογής.
     * Αν υπάρχει βλέπουμε τι έδωσε ο χρήστης και αν συμπλήρωσε όλα τα απαραίτητα πεδία,
     * προχωράμε στην αναφορά
     *
     */
    public void onClkReport(View v) throws InterruptedException, ExecutionException {

        if ( new ConnectionDetectorTask(this).execute().get().get("connectionState") ) {

            if(getRptReasons().equals("Αλλο,αναφέρετε παρακάτω") &&
edtReason.getText().toString().equals(null)){

                new showAlertMessage(this,"Προσοχή","Πρέπει να δώσετε τον λόγο της
αναφοράς!!!") ;

              }
            else{

                //Ως όρισμα είναι το σύνθετο string χωρισμένο σε δύο μέρη με "+" ανάμεσα
                totalReasons = getRptReasons()+" "+edtReason.getText().toString();
                new AlertDialog.Builder( this )
                .setTitle( "Επιβεβαίωση αναφοράς" )
                .setMessage( "Εχετε αναφέρει το εξής : ' "+totalReasons +"'  είστε βέβαιος
ότι θέλετε να στείλετε?")
                .setOnCancelListener( new OnCancelListener() {

                    @Override
                    public void onCancel(DialogInterface dialog) {

                    }
                })
                .setPositiveButton( "Ναι", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        new CreateReportTask( totalReasons ).execute() ;

                    }
                })
                .setNegativeButton( "Οχι", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {

                    }
                } )
            .show();

             imgBtnReport.setEnabled(false);
        }
      }

}

//=============================================================================================
================

    /**
     * @param v View για το button
     *
     * @info
     * (build-in)Συνάρτηση για τον τερματισμό της τρέχουσας activity με το πάτημα του back
(hardware button)
     */
    public void onClkBack(View v) {

        finish();
      }


//=============================================================================================
================
    /**
     * @info
     * Επιστρέφει όποιο στοιχείο έχει επιλέξει από το spinner ο χρήστης
     */
```

```
    public void onItemSelected(AdapterView<?> parent, View view,int pos, long id) {

        setRptReasons(parent.getItemAtPosition(pos).toString());

    }

    public void onNothingSelected(AdapterView<?> arg0) {

    }

    public String getRptReasons() {
        return RptReasons;
    }

    public void setRptReasons(String rptReasons) {
        RptReasons = rptReasons;
    }

//=====================================================================


/**
 *
 * @info
 * Με την κλάση αυτή γίνετε η εισαγωγή report στη βάση δεδομένων
 */

protected class CreateReportTask extends AsyncTask<String, String, String> {

    private ArrayList<NameValuePair> parameters =  new ArrayList<NameValuePair>() ;
    private String Repreasons;


public CreateReportTask(String repReason) {
    //  Auto-generated constructor stub

    this.Repreasons = repReason;

}


/**
     * @info
     * Πριν ξεκινήσει η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη
     *
     * @details
     * Πριν ξεκινήσει η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη
     * */
            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(ReportActivity.this);
                pDialog.setMessage("Δημιουργία αναφοράς..");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(true);
                pDialog.show();
            }

            /**
             * @info
             * Γίνετε η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη
             *
             * @details
             * Γίνετε η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη.
             * Στέλνουμε στοιχεία όπως : λόγοι report , από ποιον έγινε και σε ποιν
αναφέρεται και όλα
             * αυτά για την συγκεκριμένη παραγγελία
             * */

            protected String doInBackground(String... args) {

            try {

                SharedPreferences orderPrefs = getSharedPreferences("myOrder",
MODE_PRIVATE);
                SharedPreferences prefs = getSharedPreferences("fromCustomer",
MODE_PRIVATE);

                parameters.add(new BasicNameValuePair("getReasons", this.Repreasons ));
                parameters.add(new
BasicNameValuePair("orderid",String.valueOf(orderPrefs.getString("orderid", "")  ) ) );
```

```
                parameters.add(new
BasicNameValuePair("reportfrom",prefs.getString("customerdevid","") ) );
                parameters.add(new
BasicNameValuePair("reportTo",orderPrefs.getString("selectedDriverDevId", "") ) );
                parameters.add(new BasicNameValuePair("userType","customer" ) );
                // Building Parameters


                    // getting JSON Object
                    JSONObject myjobg = null;
                    httpJSONParser json = new httpJSONParser();
                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeReport.php",
"POST",parameters);

                // check for success tag
                    if (Integer.parseInt(myjobg.get("success").toString()) == 1) {
                        // successfully reported

                    } else {
                        // failed to make report

                    }
                }
                catch (JSONException e) {
                    e.printStackTrace();

                }
                catch(Exception e)
                {
                    e.printStackTrace();
                }

                return null;
            }

            /**
             * @info
             *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα
             *
             * @details
             * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα.
             *
             * **/
            protected void onPostExecute(String file_url) {
                // dismiss the dialog once done
                pDialog.dismiss();
            }

        }
//================================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
* @info
* Κώδικας σχετικά με το hardware button menu
*
*
* @details
* Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
* με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
* Έχει τις δυνατότητες bookmark, share & profile.
*
* */
@Override
public boolean onOptionsItemSelected(MenuItem item){
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
  switch (item.getItemId()){

  case R.id.menu_contact:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;
  case R.id.menu_report:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
```

```
    return true;

  case R.id.menu_bookmark:
      saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
      return true;
  case R.id.menu_share:
    share();
      return true;
  case R.id.menu_profile:
    Toast.makeText(this, "Είστε ήδη εδώ!!", Toast.LENGTH_SHORT).show();
      return true;
  case R.id.menu_aboutTheApp:
    aboutTheApp();
      return true;
  default:
      return super.onOptionsItemSelected(item);
  }

}



/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

      Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
      startActivity(intent);
}

/**
* @param c το περιεχόμενο της δραστηριότητας.
* @param title ο τίτλος του σελιδοδείκτη.
* @param url το site στο οποίο θα κάνει σελιδοδείκτη.
*
* @info
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*
* @details
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*/
public static final void saveBookmark(Context c, String title, String url) {
  Intent i = new Intent(Intent.ACTION_INSERT,
               android.provider.Browser.BOOKMARKS_URI);
  i.putExtra("title", title);
  i.putExtra("url", url);
  c.startActivity(i);
}

/**
* @info
* Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
* με την εφαρμογή
*
* @details
* Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
* με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
* για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
*/
public void share () {
    Intent share = new Intent(Intent.ACTION_SEND);
    String shareBody = "Taxi Anytime";
    share.setType("text/plain");
    share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
    share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    startActivity(Intent.createChooser(share, "Send via"));

}

/**
* @info
* Πληροφορίες σχετικά με την εφαρμογή.
*
* @details
* Πληροφορίες σχετικά με την εφαρμογή.
*/
public void aboutTheApp() {

    setContentView(R.layout.menu_about);
```

```
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

}


/**
* @info
* Αλλαγή προφίλ
*
* @details
* Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
*/
public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);
}


}//end class
```

APXEIO showAlertMessage.java

```
package app.taxiAnytimeCustomer.Common;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import app.taxiAnytimeCustomer.R;

/**
 * @info
 * Κλάση για την διαχείριση των προηδοποιητικών μυνημάτων
 * Μπορεί να χρησιμοποιηθεί από όλες τις κλάσεις
 *
 */
public class showAlertMessage {

private Activity actContext ;
private String title ;
private String message ;

    public showAlertMessage() {
        actContext = null;
        title = null;
        message = null;

    }
    public showAlertMessage(Activity act,String tlt,String msg) {

        actContext = act;
        title = tlt;
        message = msg;

        showConnectionAlertMessage(actContext,title,message);
    }

        /**
         *
         *
         * @param message - Το μήνυμα
         *
         * @info
         * Συνάρτηση εμφάνισης μυνήματος σφάλματος
         *
```

```
         * @details
         *Συνάρτηση εμφάνισης μυνήματος σφάλματος
         *
         *
         *
         * */

         public void showConnectionAlertMessage (Activity actAcontext , String title,String
message) {

             AlertDialog alertDialog = new AlertDialog.Builder(
                     actAcontext ).create();

                alertDialog.setTitle(title);
                alertDialog.setMessage(message);
                alertDialog.setIcon(R.drawable.tick);

                // Setting OK Button
                alertDialog.setButton("OK", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {


             }
             });

                // Showing Alert Message
                alertDialog.show();
        }

}
```

ΠΑΚΕΤΟ
app.taxiAnytimeCustomer.Customer

ΑΡΧΕΙΟ CommentsAndRating.java

```
package app.taxiAnytimeCustomer.Customer;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.RatingBar;
import android.widget.RatingBar.OnRatingBarChangeListener;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;

import app.taxiAnytimeCustomer.Common.Contact;

import app.taxiAnytimeCustomer.Common.ConnectionDetectorTask;
import app.taxiAnytimeCustomer.Common.EditProfileActivity;
import app.taxiAnytimeCustomer.Common.ReportActivity;
import app.taxiAnytimeCustomer.Common.globalVariables;
import app.taxiAnytimeCustomer.Common.httpJSONParser;


/**
 * @info
 *Κλάση σχετική με την βαθμολόγιση και σχολιασμό του οδηγού της παραγγελίας.
 */
public class CommentsAndRating extends Activity {

    protected RatingBar ratingBar;
    protected TextView txtRatingValue;
    protected EditText txtComments;
    protected Button btnMakeReport;
    protected CheckBox checkConfirmOrder;
    protected Button btnSubmit;



    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";


  /**
  * @info
  * Αρχικοποίηση menu buttons
  *
  * @details
  * Αρχικοποίηση menu buttons
  */

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
          MenuInflater menuInflater = getMenuInflater();
          menuInflater.inflate(R.layout.menu_items, menu);
          return true;
      }


    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.rating);

        ratingBar = (RatingBar) findViewById(R.id.ratingBar);

        txtRatingValue = (TextView) findViewById(R.id.txtRatingValue);
        txtComments = (EditText) findViewById(R.id.comments);


        addListenerOnRatingBar();



    }


    /**
     * @info
     * Αν αλλάξει ο χρήστης το rating θα του εμφανίσει τι έχει επιλέξει.
     *
     */
    public void addListenerOnRatingBar() {


        ratingBar.setOnRatingBarChangeListener(new OnRatingBarChangeListener() {
            public void onRatingChanged(RatingBar ratingBar, float rating,
                    boolean fromUser) {
                txtRatingValue.setText(String.valueOf(rating));

            }
        });
    }


     /**
     * @param v  το View για το button
     * @throws ExecutionException
     * @throws InterruptedException
     *
     * @info
     *  Πατώντας το κουμπί γίνεται υποβολή του σχόλιου
     * @details
     * Αρχικά γίνεται έλεγχος για το αν υπάρχει σύνδεση με το internet καθώς και με τους
server,
     * και στη συνέχεια μετατρεπει τα δεδομενα σε καταλληλη μορφη και τα στελνει στον server
     * να τα αποθηκευσει στην βαση.
     *
     *
     */
    public void onClkSubmit(View v) throws InterruptedException, ExecutionException {


             if(fieldsAreOk() == true ) {
                 if ( new
ConnectionDetectorTask(this).execute().get().get("connectionState") ){
                     new CreateCommentTask().execute();

                     finish();
                     Intent toListview = new
Intent(CommentsAndRating.this,CustomerActivity.class);
                     startActivity(toListview );
                 }

             }

}

private boolean fieldsAreOk(){

        if(this.txtComments.getText().toString().length() == 0 ||
(this.ratingBar.getRating() == 0.0)){
            Toast toast =
Toast.makeText(CommentsAndRating.this.getApplicationContext(),"Παρακαλούμε ψηφίστε ή/και
αφήστε σχόλιο.", Toast.LENGTH_LONG);
            toast.show();

            return false;
        }
        else
            return true;
```

```
        }
//=====================================================


    /**
     *
     * @info
     * Με την κλάση αυτή γίνετε η προσθήκη σχόλιου
     */
protected class CreateCommentTask extends AsyncTask<String, String, String> {


    private ProgressDialog pDialog;
    /**
         * @info
         * Πριν ξεκινήσει η διαδικασία  εμφανίζεται μήνυμα στην οθόνη χρήστη
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία εμφανίζεται μήνυμα στην οθόνη χρήστη για να τον
         * ενημερώσει ότι κάτι γίνεται
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(CommentsAndRating.this);
            pDialog.setMessage("Καταχώρηση σχόλιου..");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }


        protected String doInBackground(String... args) {


            SharedPreferences orderPrefs = getSharedPreferences("myOrder", MODE_PRIVATE);
            SharedPreferences fromCustomer = getSharedPreferences("fromCustomer",
MODE_PRIVATE);

            // Building Parameters
            ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
            parameters.add(new BasicNameValuePair("rate",String.valueOf(
ratingBar.getRating())) );
            parameters.add(new BasicNameValuePair("comment",String.valueOf(
txtComments.getText())) );
            parameters.add(new BasicNameValuePair("driverDevId",String.valueOf(
orderPrefs.getString("selectedDriverDevId", "") ) ) );
            parameters.add(new
BasicNameValuePair("orderid",String.valueOf(orderPrefs.getString("orderid", "")  ) ) );
            parameters.add(new
BasicNameValuePair("customerDevID",String.valueOf(fromCustomer.getString("customerdevid",
"")  ) ) );


            try {
                // getting JSON Object
                JSONObject myjobg = null;
                httpJSONParser json = new httpJSONParser();
                myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeCommentsRati
ngs.php","POST",parameters);


            // check for success tag


                if (Integer.parseInt(myjobg.get("success").toString()) == 1) {
                    // successfully commented

                } else {
                    // failed to make a comment

                }

            }
            catch (JSONException e) {
                e.printStackTrace();

            }
```

```
            return null;
        }


        protected void onPostExecute(String file_url) {

            pDialog.dismiss();
            pDialog = null;
        }

    }

//======================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
   * @info
   * Κώδικας σχετικά με το hardware button menu
   *
   *
   * @details
   * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
   * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
   * Έχει τις δυνατότητες bookmark, share & profile.
   *
   * */
  @Override
  public boolean onOptionsItemSelected(MenuItem item)
  {
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
      switch (item.getItemId())
      {

      case R.id.menu_contact:

        try {
            call();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
      case R.id.menu_report:

        report();
        return true;

      case R.id.menu_bookmark:
          saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
          return true;
      case R.id.menu_share:
        share();
          return true;
      case R.id.menu_profile:
        changeProfile();
          return true;
      case R.id.menu_aboutTheApp:
        aboutTheApp();
          return true;
      default:
          return super.onOptionsItemSelected(item);
      }

  }

  /**
   * @throws ExecutionException
 * @throws InterruptedException
 * @info
   * Μεταβαίνουμε στην δραστηριότητα της κλήσης/sms
   */
  public void call() throws InterruptedException, ExecutionException{
      Contact contact = new Contact(this);
      contact.makeContact();
  }

  /**
 * @info
 * Μεταβαίνουμε στην δραστηριότητα της αναφοράς
```

```
 */
public void report(){

      Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
      startActivity(intent);
  }

  /**
   * @param c το περιεχόμενο της δραστηριότητας.
   * @param title ο τίτλος του σελιδοδείκτη.
   * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
   *
   * @info
   * Κάνει σελιδοδείκτη το site της εφαρμογής.
   *
   * @details
   * Κάνει σελιδοδείκτη το site της εφαρμογής.
   */
  public static final void saveBookmark(Context c, String title, String url) {
      Intent i = new Intent(Intent.ACTION_INSERT,
                    android.provider.Browser.BOOKMARKS_URI);
      i.putExtra("title", title);
      i.putExtra("url", url);
      c.startActivity(i);
  }

  /**
   * @info
   * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
   * με την εφαρμογή
   *
   * @details
   * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
   * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
   * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
   */
  public void share () {
    Intent share = new Intent(Intent.ACTION_SEND);
    String shareBody = "Taxi Anytime";
    share.setType("text/plain");
    share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
    share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    startActivity(Intent.createChooser(share, "Send via"));

  }

  /**
   * @info
   * Πληροφορίες σχετικά με την εφαρμογή.
   *
   * @details
   * Πληροφορίες σχετικά με την εφαρμογή.
   */
  public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

  }


  /**
   * @info
   * Αλλαγή προφίλ
   *
   * @details
   * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
   */
  public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
```

```
    startActivity(intent);
  }



}//end of class
```

ΑΡΧΕΙΟ CountdownActivity.java

```
package app.taxiAnytimeCustomer.Customer;



import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.Common.ConnectionDetectorTask;
import app.taxiAnytimeCustomer.Common.globalVariables;
import app.taxiAnytimeCustomer.Common.httpJSONParser;
import app.taxiAnytimeCustomer.Common.showAlertMessage;
import app.taxiAnytimeCustomer.DriversList.ListViewActivity;




/**
 * @info
 * Η κλάση αυτή ειναι υπεύθυνη για έναν timer ο οποίος είναι υπεύθυνος για τον χρόνο
 * που θα περιμένει ο χρήστης ανταπόκριση στην κλήση του για ταξί απο τους οδηγούς.
 *
 */

public class CountdownActivity extends Activity {
  /** Properties **/


  protected Button btnStartTimer;
  protected TextView lblTimeLeft;

  protected int waitTime = 20;
  public boolean noResults ;
  protected CountDownTimer countDownTimer;
  protected Context context ;

  protected SharedPreferences myPrefs ;
  protected String customerid;


/**
* @info
* Αρχικοποίηση activity
*
* @details
* Αντιστοιχούμε τις όψεις του xml αρχείου με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
* Ανακτούμε το device id απο ένα αρχείο SharedPreferences, καλούμε συνάρτηση
παραμετροποίησης του
```

```
* time, και τον ξεκινάμε
*
*/

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.countdown);

    lblTimeLeft = (TextView) findViewById(R.id.txtWaitTime);

    context = this;

    SharedPreferences prefs = getSharedPreferences("fromCustomer", MODE_PRIVATE);
    customerid = prefs.getString("customerdevid",null);

    setWaitTime(waitTime);
    startCounting();



  }

  /** Methods **/

  /**
   * @param seconds Ποσα δευτερολεπτα θα ειναι σε αναμονη
   *
   * @info
   * Δινουμε τον χρονο για τον οποιο θα περιμενει ο πελατης
   *
   * @details
   * Δινουμε τον χρονο για τον οποιο θα περιμενει ο πελατης
   *
   */
  public void setWaitTime(int seconds) {
    waitTime = seconds;
    lblTimeLeft.setText(String.valueOf(waitTime) + "s");
  }




  /**
   * @info
   * Αρχιζει την αντιστροφη μετρηση
   */
  public void startCounting() {

      countDownTimer = new CountDownTimer(waitTime  * 1000, 1000) {

      @Override
      public void onTick(long millisUntilFinished) {
          lblTimeLeft.setText(String.valueOf(millisUntilFinished / 1000) + "s");
      }

      @Override
      public void onFinish() {
          try {
            if ( new
ConnectionDetectorTask(CountdownActivity.this).execute().get().get("connectionState") ){
                new checkOrdersTask().execute();

              }
        } catch (Exception e) {
            new showAlertMessage(CountdownActivity.this ,"Ωχ! Κάτι πήγε
στραβά!!!","Δοκιμάστε αργότερα");


            e.printStackTrace();

        }

      }

    }.start();

  }
```

```
/**
 * @param v το View για το button
 *
 * @info
 * Σταματάμε τον timer όταν πατήσει το κουμπί ο χρήστης.
 *
 * @details
 * Σταματάμε τον timer όταν πατήσει το κουμπί ο χρήστης.
 */


public void onImgBtnTerminateTimer(View v){

    countDownTimer.cancel();
    countDownTimer.onFinish();

}



/**
 *
 * @info
 * Σταματάμε τον timer όταν πατήσει το hardware κουμπί back ο χρήστης.
 *
 * @details
 * Σταματάμε τον timer όταν πατήσει το hardware κουμπί back ο χρήστης.
 */

@Override
public void onBackPressed() {

    countDownTimer.cancel();
    finish();

    super.onBackPressed();
    }

//=============================================================================
/**
 * @info
 * Βοηθητική μέθοδος για να τερματίσει το timer πριν λήξει ο χρόνος.
 *
 * @details
 * Βοηθητική μέθοδος για να τερματίσει το timer πριν λήξει ο χρόνος.
 * Σε αυτή την περίπτωση βλέπουμε πόσοι οδηγοί απάντησαν μέχρι τώρα.
 *
 * @param result το αποτέλεσμα που λαμβάνει από το asyntask
 */
protected void onForceStopTimer(boolean result){

    if(result == true) {
          new AlertDialog.Builder( context )
            .setTitle( "Ουπς!" )
            .setMessage( "Δεν υπάρχουν διαθέσιμοι οδηγοί," +
                    " Θελετε να ξανα προσπαθήσετε?" )
            .setPositiveButton( "Ναι", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {

                    countDownTimer .cancel();
                    countDownTimer = null;
                    startCounting();


                }
            })
            .setNeutralButton( "Επιστροφή", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {

                    //κωδικας για επιστροφη εκει που διαλεγει σημειο.
                    countDownTimer .cancel();
                    countDownTimer = null;

                    finish();
                    Intent intent = new
Intent(CountdownActivity.this,CustomerActivity.class);
                    startActivity(intent);


                }
```

```
            }).show();
        }
        else { //if(noResults == false)
        {
            //κωδικας για να μεταβουμε στο listview στο οποιο θα
            //εμφανιστουν οι ταξιτζιδες που εδειξαν ενδιαφερον

            finish();
            Intent toListview = new Intent(CountdownActivity.this,ListViewActivity.class);
            startActivity(toListview );

        }
       }
}

//================================================================================

    /**
     * Background Async Task to check for orders
     * */



    /**
     * @info
     * Ελέγχει στην βάση αν υπάρχει διαθέσιμη παραγγελία απο τον συγκεκριμένο πελάτη (εδώ
είναι που
     * χρειάζεται και το device id)
     */
    protected class checkOrdersTask extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
        private JSONObject myjobg = null;
        private httpJSONParser json = new httpJSONParser();
        private ProgressDialog pDialog;

        /**
         * @info
         * Πριν ξεκινήσει η διαδικασία του ελέγχου  εμφανίζεται μήνυμα στην οθόνη χρήστη.
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία του ελέγχου  εμφανίζεται μήνυμα στην οθόνη χρήστη.
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(CountdownActivity.this);
            pDialog.setMessage("Περιμένετε...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }


        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και βλέπουμε αν έχουν
δηλώσει
         * οδηγοί ενδιαφέρον για την συγκεκριμένη παραγγελία
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και βλέπουμε αν έχουν
δηλώσει
         * οδηγοί ενδιαφέρον για την συγκεκριμένη παραγγελία
         *
         * */

        protected String doInBackground(String... args) {

            try {

                    parameters.add(new BasicNameValuePair("customerdeviceid",customerid));
                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/checkforOrders.p
hp","POST",parameters);

                    if(Integer.parseInt(myjobg.get("success").toString()) == 1){
                        noResults = false;
                    }
                    else{
                        noResults = true;
```

```
                }

            } catch (NumberFormatException e) {
                // Auto-generated catch block
                e.printStackTrace();
            } catch (JSONException e) {
                //  Auto-generated catch block
                e.printStackTrace();
            }

            return null;
        }

        /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα εάν το επιθυμεί ο χρήστης.
         *
         * @details
         * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα εάν το επιθυμεί ο χρήστης. Επίσης σε περίπτωση που δέν
         * έχει αποτελέσματα η παραγγελία του μπορεί να ξανά στείλει ειδοποίηση στους
οδηγούς
         *
         * **/
        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done

            //this.parameters.remove(parameters);
            pDialog.dismiss();
            pDialog = null;

            onForceStopTimer(noResults);
        }

    }

//=========================================================

}
```

APXEIO CustomerActivity.java

```
package app.taxiAnytimeCustomer.Customer;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.Common.ConnectionDetectorTask;
import app.taxiAnytimeCustomer.Common.EditProfileActivity;
```

```
import app.taxiAnytimeCustomer.Common.globalVariables;
import app.taxiAnytimeCustomer.Common.httpJSONParser;
import app.taxiAnytimeCustomer.Common.showAlertMessage;
import app.taxiAnytimeCustomer.PushService.PushService;
import app.taxiAnytimeCustomer.userTypesFactory.Users;
import app.taxiAnytimeCustomer.userTypesFactory.UsersFactory;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;



/**
 * @info
 * Η βασικότερη κλάση της εφαρμογής.
 * Παράγει τον χάρτη , χρησιμοποιεί το gps και ξεκινάει η υπηρεσία(Push Service) και
εγγράφεται σε αυτήν η
 * συσκευή μας,ώστε να μπορούμε να στέλνουμε αλλά και να λαμβάνουμε ειδοποιήσεις .
 *
 *
 */

public class CustomerActivity extends MapActivity
{


    private MapView mv;
    private MapController mc;
    public GeoPoint gp;
    private LocationManager locationManager;
    private  LocationListener loclistener;
    public Location myLocation;
    public CustomerPositionOverlay customerOverlay;
    public static TextView txtAddress;

    private final String BOOKMARK_TITLE = "Taxi Anytime";
    private final String BOOKMARK_URL ="www.taxianytime.hostzi.com";


    public  Users customer;



  @Override
  protected boolean isRouteDisplayed()
  {
    return false;
  }


    /**
   * @info
   * Αρχικοποίηση menu buttons
   *
   * @details
   * Αρχικοποίηση menu buttons
   */
  @Override
  public boolean onCreateOptionsMenu(Menu menu)
  {
      MenuInflater menuInflater = getMenuInflater();
      menuInflater.inflate(R.layout.menu_items, menu);
      return true;
  }


    /**
   * @info
   * Αρχικοποίηση activity
   *
   * @details
   * Αντιστοιχούμε τις όψεις του xml αρχείου με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
   * Εισάγουμε το id της συσκευής σε αρχείο SharedPreferences για να μπορέσουμε να το
χρησιμοποιήσουμε αργότερα
   * στις επόμενες δραστηριότητες/service.
   * Ξεκινάει το service μετά από όλα αυτά λαμβάνουμε την τοποθεσία gps με την βοήθεια της
κατάλληλης
```

```
     * κλασης (ConnectionManager) και ζωγραφίζουμε το εικονίδιο του χρήστη.
     *   *
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);



      txtAddress = (TextView)findViewById(R.id.txtAddress);
      customer = UsersFactory.createUser("customer");


     Editor editor = getSharedPreferences(PushService.TAG, MODE_PRIVATE).edit();
     editor.putString(PushService.PREF_DEVICE_ID, customer.getDeviceID());
     editor.commit();


   //Για να έχουμε πρόσβαση στο id της συσκευής σε όλες τις activities
      SharedPreferences pref = getSharedPreferences("fromCustomer", MODE_PRIVATE);
      SharedPreferences.Editor edit = pref.edit();
      edit.putString("customerdevid","/"+customer.getDeviceID());
      edit.commit();

   try{

     //start the service
      PushService.actionStart(getApplicationContext());

      mv=(MapView) findViewById(R.id.myMapView);
      mv.setBuiltInZoomControls(true);
      mv.setSatellite(false);
      mv.setStreetView(true);

      mc= mv.getController();
      mc.setZoom(16);



      locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
      Criteria criteria = new Criteria();
      criteria.setAccuracy(Criteria.NO_REQUIREMENT);  //NO_REQUIREMENT
      criteria.setPowerRequirement(Criteria.NO_REQUIREMENT);  //NO_REQUIREMENT
      String provider = locationManager.getBestProvider(criteria, true);



      myLocation = locationManager.getLastKnownLocation(provider);
      loclistener = new mylocationlistener(myLocation);
      locationManager.requestLocationUpdates(provider,0,0,loclistener);



      customer.setLatitude(myLocation.getLatitude()*1E6) ;
      customer.setLongitude(myLocation.getLongitude()*1E6);

      gp = new GeoPoint((int)customer.getLatitude(),(int)customer.getLongitude());
      mc.animateTo(gp);

      Drawable marker=getResources().getDrawable(R.drawable.customer_icon2);
      int markerWidth = marker.getIntrinsicWidth();
      int markerHeight = marker.getIntrinsicHeight();
      marker.setBounds(-markerWidth/2, -markerHeight,markerWidth/2, 0);

      customerOverlay = new
CustomerPositionOverlay(marker,getApplicationContext(),myLocation,customer);
      customerOverlay.addItem(gp,customer.getDeviceID(),null);
      mv.getOverlays().add(customerOverlay);


CustomerPositionOverlay.updateAddress(myLocation.getLatitude(),myLocation.getLongitude());

    }
    catch(Exception e){
        new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Επανεκκινήστε την εφαρμογή και
δοκιμάστε αργότερα");
        e.printStackTrace();

    }
```

```
}//oncreate


/**
* @info
* Τερματισμός του service για αποδέσμευση πόρων
*/
@Override
protected void onDestroy() {
    // Auto-generated method stub
     PushService.actionStop(getApplicationContext());
    super.onDestroy();
}

 //=============================================================================================
  /**
 * @param text Η διεύθυνση που βρίσκεται ο χρήστης
 *
 * @info
 * Θέτουμε στο texview τη διεύθυνση του χρήστη.
 *
 * @details
 * Θέτουμε στο texview τη διεύθυνση του χρήστη και κάνουμε refresh ,ώστε να εμφανιστεί.
 *
 */
public static void setTextAddress(String text) {
      txtAddress.setText(text);
      txtAddress.refreshDrawableState();
    }

//=============================================================================================
/**
*
*
* @info
* Για μελλοντική χρήση, όταν θέλουμε να έχουμε την νέα τοποθεσία αν αλλάξουμε
* θέση
*
*Είναι αυτοπαραγώμενη κλάση του SDK
*
*/

  private class mylocationlistener implements LocationListener {

      public mylocationlistener(Location myloc) {
        //  Auto-generated constructor stub


    }
        @Override
        public void onLocationChanged(Location location) {
            if (location != null) {



            }
        }

        @Override
        public void onProviderDisabled(String provider) {
            //  Auto-generated method stub

        }

        @Override
        public void onProviderEnabled(String provider) {
            //  Auto-generated method stub

        }

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
            //  Auto-generated method stub

        }
  }
 //=============================================================================================
```

```
 /**
 * @param v  το View για το button
 * @throws ExecutionException
 * @throws InterruptedException
 *
 * @info
 *  Πατώντας το κουμπί γίνεται αναζήτηση για τους διαθέσιμους οδηγούς
 * @details
 * Αρχικά γίνεται έλεγχος για το αν υπάρχει σύνδεση με το internet καθώς και με τους server,
 * και στη συνέχεια πραγματοποιείται η ειδοποίηση στο παρασκήνιο
 *
 *
 */
public void OnimgBtnClickSearch(View v) throws InterruptedException, ExecutionException
 {

    if ( new ConnectionDetectorTask(this).execute().get().get("connectionState") ){

        new NotifyDriversTask().execute();
      }

 }


//=============================================================================================
=

    /**
     * Background Async Task to notify drivers
     * */


/**
 *
 * @info
 * Με την κλάση αυτή γίνετε ενημέρωση των διαθέσιμων οδηγών για νέο πελάτη
 */

    class NotifyDriversTask extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
        private ArrayList<NameValuePair> parametersDrivers = new ArrayList<NameValuePair>();
        private ProgressDialog pDialog;

        /**
         * @info
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(CustomerActivity.this);
            pDialog.setMessage("Ειδοποίηση οδηγών..");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        /**
         * @info
         * Γίνετε ενημέρωση των διαθέσιμων οδηγών για νέο πελάτη.
         *
         * @details
         * Παίρνουμε την τρέχουσα τοποθεσία του πελάτη (γεωργαφικό μήκος & πλάτος), και το
id της
         * συσκευης και τα κραταμε σε ενα Arraylist, μέσω ενος php script βλέπουμε ποιοί
οδηγοί
         * είναι διαθέσιμοι, και στέλνουμε σε αυτούς τα δεδομένα.
         * */
        @Override
        protected String doInBackground(String... args) {


            parameters.add(new
BasicNameValuePair("lat",String.valueOf(CustomerPositionOverlay.currentLat)));
            parameters.add(new
BasicNameValuePair("lng",String.valueOf(CustomerPositionOverlay.currentLon)));
            parameters.add(new BasicNameValuePair("customerid","/"+customer.getDeviceID()));
```

```
            JSONObject myjobg = null  ;
            httpJSONParser json = new httpJSONParser();
            httpJSONParser json2 = new httpJSONParser();

            try {

                    //String orderid = myjobg.get("orderid").toString();
                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/"+"getdriverinfo
.php","GET",parametersDrivers);

                    if(Integer.valueOf( myjobg.get("success").toString()) == 1)
                    {
                        for(int i=0;i<myjobg.getJSONArray("drivers").length();i++)
                        {
                            parameters.add(new
BasicNameValuePair("deviceid"+i,myjobg.getJSONArray("drivers").getJSONObject(i).get("driverD
eviceID").toString()));

                        }


json2.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/mqttClient/notifyDriver
s.php","POST",parameters);

                    }

            } catch (NumberFormatException e) {
                // Auto-generated catch block
                e.printStackTrace();
            } catch (JSONException e) {
                //  Auto-generated catch block
                e.printStackTrace();
            }
            return null;
        }

        /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα, και πηγαίνουμε σε οθόνη αναμονής έως ότου
         * ανταποκριθούν οι οδηγοί.
         *
         * **/
        @Override
        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done

            this.parameters.remove(parameters);
            this.parametersDrivers.remove(parametersDrivers);


            pDialog.dismiss();


            Intent intent = new Intent(CustomerActivity.this,CountdownActivity.class);
            startActivity(intent);


        }

    }

//======================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
     * @info
     * Κώδικας σχετικά με το hardware button menu
     *
     *
     * @details
     * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
     * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
```

```
     * Έχει τις δυνατότητες bookmark, share & profile.
     *
     * */
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        // Single menu item is selected do something
        // Ex: launching new activity/screen or show alert message
        switch (item.getItemId())
        {

        case R.id.menu_contact:

            Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_report:

            Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
            return true;

        case R.id.menu_bookmark:
            saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
            return true;
        case R.id.menu_share:
            share();
            return true;
        case R.id.menu_profile:
            changeProfile();
            return true;
        case R.id.menu_aboutTheApp:
            aboutTheApp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }

    }


    /**
     * @param c το περιεχόμενο της δραστηριότητας.
     * @param title ο τίτλος του σελιδοδείκτη.
     * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
     *
     * @info
     * Κάνει σελιδοδείκτη το site της εφαρμογής.
     *
     * @details
     * Κάνει σελιδοδείκτη το site της εφαρμογής.
     */
    public static final void saveBookmark(Context c, String title, String url) {
        Intent i = new Intent(Intent.ACTION_INSERT,
                      android.provider.Browser.BOOKMARKS_URI);
        i.putExtra("title", title);
        i.putExtra("url", url);
        c.startActivity(i);
    }

    /**
     * @info
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή
     *
     * @details
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
     * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms
κλπ)
     */
    public void share () {
        Intent share = new Intent(Intent.ACTION_SEND);
        String shareBody = "Taxi Anytime";
        share.setType("text/plain");
        share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
        share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
        startActivity(Intent.createChooser(share, "Send via"));

    }

    /**
     * @info
```

```
     * Πληροφορίες σχετικά με την εφαρμογή.
     *
     * @details
     * Πληροφορίες σχετικά με την εφαρμογή.
     */
    public void aboutTheApp() {

        setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

    }


    /**
     * @info
     * Αλλαγή προφίλ
     *
     * @details
     * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
     */
    public void changeProfile(){

        Intent intent = new Intent(this.getApplicationContext(), EditProfileActivity.class);
        startActivity(intent);
    }
//=============================================================



}//end class
```

ΑΡΧΕΙΟ CustomerPositionOverlay.java

```
package app.taxiAnytimeCustomer.Customer;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.drawable.Drawable;
import android.location.Address;
```

```
import android.location.Geocoder;
import android.location.Location;

import app.taxiAnytimeCustomer.userTypesFactory.Users;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.MapView;
import com.google.android.maps.OverlayItem;


/**
 * @info
 * Βοηθητική κλάση για την εισαγωγή στοιχείων στο χάρτη.
 * Χρησιμοποιείται σε συνεργασία με την customerActivity και μας επιτρέπει να ζωγραφίζουμε
αλλά και
 * να διαχειρίζομαστε πολλαπλά items πάνω στο χάρτη.
 * Τα items αυτά αποθηκεύονται σε arraylist
 *
 *
 */
public class CustomerPositionOverlay extends ItemizedOverlay<OverlayItem>
{


   private ArrayList<OverlayItem> items ;

   protected static Location mlocation;
   protected static Context mContext;
   protected Users mCustomer;
   public static double currentLat,currentLon;



   public static void setLocation(Location location) {
        mlocation = location;
      }

   /**
 * @param defaultMarker Το εικονίδιο που θα ζωγραφιστεί
 * @param context το περιεχόμενο της δραστηριότητας που καλούμε
 * @param location η τρέχουσα τοποθεσία
 * @param customer το αντικείμενο του πελάτη
 *
 * @info
 * Αρχικοποιεί τις παραμέτρους
 *
 */
public CustomerPositionOverlay(Drawable defaultMarker,Context context,Location
location,Users customer)
   {

       super(defaultMarker);

       boundCenterBottom(defaultMarker);
       items = new ArrayList<OverlayItem>();
       mContext = context;
       mlocation = location;
       mCustomer = customer;

       currentLat = location.getLatitude();
       currentLon = location.getLongitude();

       populate();

   }



   /**
 * @param p τα γεωγραφικά στοιχεία
 * @param title ο τίτλος του στοιχείου
 * @param snippet μια περιγραφή του στοιχειου
 *
 * @info
 * Προσθέτει στο arraylist τα αντκείμενα
 *
 * @details
```

```
 * Προσθέτει στο arraylist τα αντικείμενα και να δημοσιοποιεί στο χάρτη (τα εμφανίζει)
 *
 *
 */
public void addItem(GeoPoint p, String title, String snippet)
   {

       OverlayItem newItem = new OverlayItem(p, title, snippet);
       items.add(newItem);
        populate();

    }


/**
* @info
* Αυτοπαραγόμενη μέθοδος για τη ζωγραφιά των αντικειμώνων
*/
public void draw(Canvas canvas, MapView mapView, boolean shadow)
   {

       super.draw(canvas, mapView, shadow);

   }


/**
 * @param p το γεωγραφικό μήκος που κάναμε tap στο χάρτη
 * @param mapView το αντιστοιχο αντικείμενο για να το διαχειριστούμε
 *
 *
 * @info
 * Χρησιμοποιείται/ενεργοποιείται όταν κάνουμε tap πάνω στο χάρτη
 *
 * @details
 * Επειδή θέλουμε να υπάρχει μόνο ένα αντικείμενο , αυτό του πελάτη κάνουμε remove όλο το
list
 * και προσθέτουμε νέο , με την ενημερωμένη τοποθεσία.
 * Έπειτα ανανεώνουμε τον χάρτη ώστε να φαίνεται το νέο σημείο που είμαστε
 *
 *
 */
   @Override
   public boolean onTap(GeoPoint p, MapView mapView)
   {

    this.items.removeAll(items);

    this.addItem(p,null,null);

    mlocation.setLatitude(p.getLatitudeE6()/1E6);
    mlocation.setLongitude(p.getLongitudeE6()/1E6);
    updateAddress(p.getLatitudeE6()/1E6,p.getLongitudeE6()/1E6);
    currentLat = p.getLatitudeE6()/1E6;
    currentLon = p.getLongitudeE6()/1E6;

    mapView.invalidate();  //refresh map
    populate();

    return super.onTap(p,mapView);
   }


   /**
   * @info
   * Αυτοπαραγόμενη μέθοδος που δημιουργεί τα αντικείμενα του χάρτη
   */
    @Override
    protected OverlayItem createItem(int i) {

        return (items.get(i));
    }

    /**
    * @info
    * Αυτοπαραγόμενη μέθοδος που επιστρέφει το μέγεθος της λίστας  των αντικειμένων
    * που υπάρχουν στο χάρτη
    *
    * @return
    * size
    */
```

```
    @Override
    public int size(){
        return(items.size());
    }


     /**
     * @param lat Το γεωγραφικό μήκος του χρήστη
     * @param lon Το γεωγραφικό πλάτος του χρήστη
     *
     * @info
     * Μέθοδος για να λαμβάνουμε την διεύθυνση έχοντας τα γεωγραφικά μήκη και πλάτη
     *
     * @details
     * Χρησιμοποιείται η built-in κλάση  Geocoder και Address,
     * για να πάρουμε την διεύθυνση βάσει τα γεωγραφικά μήκη και πλάτη.
     * 'Επειτα περνάμε την διεύθυνση στο textedit
     *
     */
    public static void updateAddress(double lat,double lon) {

         Geocoder gc = new Geocoder(mContext, Locale.getDefault());
          try {
            List<Address> addresses = gc.getFromLocation(lat,lon, 1);

            if(addresses != null) {
                Address address = addresses.get(0);
                StringBuilder sb = new StringBuilder("Διεύθυνση:\t");

                for (int i = 0; i < address.getMaxAddressLineIndex(); i++)
                  sb.append(address.getAddressLine(i)).append("\n");

                //setting text
                CustomerActivity.setTextAddress(sb.toString());
            }

          } catch (IOException e) {}

}


}//end main class
```

APXEIO CustomerStartEndRide.java

```
package app.taxiAnytimeCustomer.Customer;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
```

```
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.Common.Contact;
import app.taxiAnytimeCustomer.Common.ConnectionDetectorTask;
import app.taxiAnytimeCustomer.Common.EditProfileActivity;
import app.taxiAnytimeCustomer.Common.ReportActivity;
import app.taxiAnytimeCustomer.Common.globalVariables;
import app.taxiAnytimeCustomer.Common.httpJSONParser;



/**
 * @info
 * Δραστηριότητα για την εκκίνηση και τερματισμό της διαδρομής
 *
 */
public class CustomerStartEndRide extends Activity{


    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";
    protected ImageButton btnStart;
    protected ImageButton btnEnd;
    private Bundle extras;

     /**
     * @info
     * Αρχικοποίηση activity
     *
     * @details
     * Αντιστοιχούμε τις όψεις του xml αρχείου σε αντικείμενα , ώστε να μπορούμε να το
διαχειριστούμε.
     * Εδώ ελέγχουμε αν υπάρχει εισερχόμενη ειδοποίηση από το service (για το αν έχει φτάσει
ο οδηγός).
     * Εφόσον υπάρχει ενεργοποιείται το κουμπί για να μπορέσουμε να αρχίσουμε τη διαδρομή,
διαφορετικά
     * αναμένουμε
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        //  Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.start_end_ride);

        btnStart = (ImageButton) findViewById(R.id.imageButtonStartRide);
        btnEnd = (ImageButton) findViewById(R.id.imageButtonEndRide);



        btnEnd.setEnabled(false);
        btnStart.setEnabled(false);


            //Αν δηλαδή λάβαμε ειδοποίηση ενεργοποιούμε τα widgets
                extras = getIntent().getExtras();
                if(extras != null ){
                    if ( extras.getInt("driverHasArrived") == 1 ){
                        btnStart.setEnabled(true);
                    }

                }


    }


     /**
       * @info
       * Αρχικοποίηση menu buttons
       *
       * @details
       * Αρχικοποίηση menu buttons
       */
      @Override
      public boolean onCreateOptionsMenu(Menu menu)
      {
          MenuInflater menuInflater = getMenuInflater();
          menuInflater.inflate(R.layout.menu_items, menu);
```

```
            return true;
        }


        /**
         * @param v το View για το button
         *
         * @throws ExecutionException
         * @throws InterruptedException
         *
         * @info
         * Πατώντας το κουμπί ξεκινά η διαδρομή (orderState = 3)
         *
         * @details
         * Αρχικά ελέγχουμε για το αν υπάρχει ενεργή σύνδεση στο internet και το gps αν
είναι ανοικτό,
         * έπειτα ανακτούμε από τις SharedPreferences τα στοιχεία της παραγγελίας που
επιλέχθηκε καθώς και
         * το id της συσκευής του πελάτη. Αυτά χρειάζονται παρακάτω στο AsynTask.
         *
         */
    public void onClickStartRide(View v) throws InterruptedException, ExecutionException
    {

        if ( new ConnectionDetectorTask(this).execute().get().get("connectionState") ){

            SharedPreferences preferencesOrders = getSharedPreferences("myOrder",
MODE_PRIVATE);
            SharedPreferences preferencesCustomer = getSharedPreferences("fromCustomer",
MODE_PRIVATE);

            new CustomerConfirmOrderTask( preferencesCustomer.getString("customerdevid",
""), preferencesOrders.getString("orderid", "")   ).execute();

            btnStart.setEnabled(false);
            btnEnd.setEnabled(true);

         }


    }



    /**
     * @param v το View για το button
     *
     * @info
     * Μεταβαίνουμε στην επόμενη δραστηριότητα (comments)
     */
    public void onClickEndRide(View v)
    {

        finish();
        Intent intent = new Intent(getApplicationContext(), CommentsAndRating.class);
        startActivity(intent);

    }

//=============================================================================================


    /**
     * @info
     * Μέθοδος για την αποστολή επιβεβαίωσης του πελάτη
     *
     * @details
     * Η κλάση "τρέχει " στο παρασκήνιο και επικοινωνεί με τη βάση δεδομένων για να θέσει
     * ότι η παραγγελία είναι αποδεκτή από τον πελάτη (orderState =3)
     *
     * */
    protected class CustomerConfirmOrderTask extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
        private ProgressDialog pDialog;
        private String deviceid;
        private String orderid;
```

```
        public CustomerConfirmOrderTask(final String customerDeviceid,final String orderid)
{
            //  Auto-generated constructor stub
            this.deviceid = customerDeviceid;
            this.orderid = orderid;
        }


        /**
         * @info
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη.
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη.
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(CustomerStartEndRide.this);
            pDialog.setMessage("Περιμένετε..");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * και θέτουμε την νέα κατάσταση παραγγελίας
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * και θέτουμε την νέα κατάσταση παραγγελίας
         *
         * */
        protected String doInBackground(String... args) {

            parameters.add(new BasicNameValuePair("customerid",this.deviceid));
            parameters.add(new BasicNameValuePair("orderid",this.orderid));
            parameters.add(new BasicNameValuePair("type","customer"));

            try {
                // getting JSON Object
                JSONObject myjobg = null;
                httpJSONParser json = new httpJSONParser();
                myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/orderConfirmatio
n.php","POST",parameters);

                if (Integer.parseInt(myjobg.get("success").toString()) == 1) {

                }

            }
            catch (Exception e) {
                e.printStackTrace();

            }
            return null;
        }

        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done

            pDialog.dismiss();
            pDialog = null;

        }



    }

    //=====================================================================
    //menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
     * @info
     * Κώδικας σχετικά με το hardware button menu
```

```
*
*
* @details
* Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
* με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
* Έχει τις δυνατότητες bookmark, share & profile.
*
* */
 @Override
 public boolean onOptionsItemSelected(MenuItem item){
   // Single menu item is selected do something
   // Ex: launching new activity/screen or show alert message
     switch (item.getItemId()){

     case R.id.menu_contact:{
           try {
               call();

           } catch (Exception e) {
               e.printStackTrace();
           }
           return true;
     }
     case R.id.menu_report:{
         report();

       return true;
     }
     case R.id.menu_bookmark:{
         saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
       return true;
     }
     case R.id.menu_share:{
           share();
         return true;
     }
     case R.id.menu_profile:{
         changeProfile();
         return true;
     }
     case R.id.menu_aboutTheApp:{
         aboutTheApp();
         return true;
     }
     default:
         return super.onOptionsItemSelected(item);
     }

 }

 public void call() throws InterruptedException, ExecutionException{
     Contact contact = new Contact(this);
     contact.makeContact();
 }

 public void report(){
     Intent intent = new Intent(CustomerStartEndRide.this,ReportActivity.class);
     startActivity(intent);
 }


   /**
    * @param c το περιεχόμενο της δραστηριότητας.
    * @param title ο τίτλος του σελιδοδείκτη.
    * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
    *
    * @info
    * Κάνει σελιδοδείκτη το site της εφαρμογής.
    *
    * @details
    * Κάνει σελιδοδείκτη το site της εφαρμογής.
    */
 public static final void saveBookmark(Context c, String title, String url) {
     Intent i = new Intent(Intent.ACTION_INSERT,
                   android.provider.Browser.BOOKMARKS_URI);
     i.putExtra("title", title);
     i.putExtra("url", url);
     c.startActivity(i);
 }
  /**
    * @info
```

```
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή
     *
     * @details
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
     * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms
κλπ)
     */
    public void share () {
      Intent share = new Intent(Intent.ACTION_SEND);
      String shareBody = "Taxi Anytime";
      share.setType("text/plain");
      share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
      share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
      startActivity(Intent.createChooser(share, "Send via"));

    }

    /**
     * @info
     * Πληροφορίες σχετικά με την εφαρμογή.
     *
     * @details
     * Πληροφορίες σχετικά με την εφαρμογή.
     */
    public void aboutTheApp() {

      setContentView(R.layout.menu_about);
          TextView aboutTextView = (TextView)findViewById(R.id.about);
          aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                  "Έκδοση 1.0<br> \n" +
                  "Copyright 2013<br>  \n" +
                  "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                  "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                  "Χατζημιχαήλ Γεώργιος <br> \n" +
                  "Τσεγγελίδης Φίλανδρος <br>"));

          aboutTextView.setLinkTextColor(Color.WHITE);
          Linkify.addLinks(aboutTextView, Linkify.ALL);

    }

    /**
     * @info
     * Αλλαγή προφίλ
     *
     * @details
     * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
     */
    public void changeProfile(){

      Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
      startActivity(intent);
    }




}
```

ПАКЕТО app.taxiAnytimeCustomer.Driver List

APXEIO CommentsDetails.java

```
package app.taxiAnytimeCustomer.DriversList;

/**
 * @info
 * Κλάση που περιέχει όλες τις πληροφορίες σχετικά με τα σχόλια και τους πελάτες
 *
 */
public class CommentsDetails {

    private String name ;
    private String comment;



    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }


public void setComment(String com) {
    this.comment = com;
}
public String getComment() {
    return comment;
}

}
```

APXEIO DriverDetails.java

```
package app.taxiAnytimeCustomer.DriversList;

import android.graphics.Bitmap;
import android.util.Log;
/**
 * @info
 * Κλάση που περιέχει όλες τις πληροφορίες σχετικά με τον οδηγό ταξί
 *
 */
public class DriverDetails {

    private String name ;
    private String distance;
    private float rate;
    private String plateNumber;
    private String driverDeviceId;
    private int orderid;
    private Bitmap myImage;


    public void setMyImage(Bitmap myImage) {
        this.myImage = myImage;
    }
    public Bitmap getMyImage() {
        return myImage;
    }
    public String getDriverDeviceId() {
        return driverDeviceId;
    }
    public void setDriverDeviceId(String driverDeviceId) {
        this.driverDeviceId = driverDeviceId;
    }

    public String getPlateNumber() {
        return plateNumber;
    }
    public int getOrderid() {
        return orderid;
    }
    public void setOrderid(int orderid) {
        this.orderid = orderid;
```

```
    }
    public void setPlateNumber(String plateNumber) {
        this.plateNumber = plateNumber;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDistance() {
        return distance;
    }
    public void setDistance(String dist) {

        this.distance = distance2dec(dist);
    }
    public float getRate() {
        return rate;
    }
    public void setRate(float rating) {
        this.rate = rating;
    }


    /**
     * @param distance Η απόσταση μεταξύ πελάτη και οδηγού.
     *
     * @info
     * Μετατρέπει την απόσταση απο μορφή xxx.xxxxx σε xxx.x
     *
     * @details
     * Δέχεται την απόσταση ως string, πχ 456.34
     * Το χωρίζει σε δυο υπο-string με βάση την ' . '
     * και επιστρέφει το πρώτο υπομέρος μαζί με τον πρώτο χαρακτήρα του δεύτερου,
     * ως αποτέλεσμα να έχουμε 1 δεκαδικό ακρίβεια
     *
     *
     *
     * @return calculatedDistance η απόσταση μεταξύ οδηγού και πελάτη σε μορφή 400.5m
     *
     *
     */
    private String distance2dec(String distance)
    {

        String []split = distance.split("\\.");

        String returned;
        try
        {
            returned = split[0]+"."+split[1].charAt(0)+"m";

        }
        catch(Exception e)
        {
            e.printStackTrace();
            returned = distance+"m";
        }


        return returned;

    }


}
```

ΑΡΧΕΙΟ ListViewActivity.java

```
package app.taxiAnytimeCustomer.DriversList;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.RatingBar;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.Common.ConnectionDetectorTask;
import app.taxiAnytimeCustomer.Common.globalVariables;
import app.taxiAnytimeCustomer.Common.httpJSONParser;
import app.taxiAnytimeCustomer.Customer.CustomerStartEndRide;


/**
 * @info
 * Κλάση υπεύθυνη για την διαχείρηση του listview που περιέχει
 * την λίστα όλων των ενδιαφερόμενων οδηγών ταξί σχετικά με μια κλήση.
 */
public class ListViewActivity extends ListActivity {
    /** Called when the activity is first created. */


public Bundle extras;
protected String customerid;
protected ListView myListView ;
protected Activity thisActivity;


public Activity getThisActivity() {
    return thisActivity;
}

/**
 * @info
 * Αρχικοποίηση activity
 *
 * @details
 * Γίνετε έλεγχος για το αν είμαστε συνδεδεμένοι στο internet και αν έχουμε gps ανοικτό
 * Στην συνέχεια εμφανίζει τα δεδομένα όλων των οδηγών που αποδέχτηκαν την συγκεκριμένη
 *  παραγγελία
 *
 */
    @Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.drivers_listview);
        myListView = (ListView)findViewById(android.R.id.list);
        thisActivity = this;
        SharedPreferences prefs = getSharedPreferences("fromCustomer", MODE_PRIVATE);
        customerid = prefs.getString("customerdevid","");
```

```
        ArrayList<DriverDetails> items_details = null;

                try {
                    if ( new
ConnectionDetectorTask(this).execute().get().get("connectionState") ){

                        items_details = new setListViewDataTask(customerid).execute().get();

                        myListView.setAdapter(new ItemListBaseAdapter(this, items_details));

                    }
                } catch (InterruptedException e) {

                    e.printStackTrace();
                } catch (ExecutionException e) {

                    e.printStackTrace();
                }

}//onCreate


    @Override
    public ListView getListView() {

        return super.getListView();
    }

//================================================================================
    /**
     * @info
     * Μέθοδοι για να "ετοιμάσουν" το listview
     *
     * @details
     * Εδώ παίρνουμε τα δεδομένα των οδηγών τα τοποθετούμε σωστά στο listview.
     */


class setListViewDataTask extends AsyncTask<String,String,  ArrayList<DriverDetails>> {

        private ArrayList<DriverDetails> results = new ArrayList<DriverDetails>();

        private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
        private JSONObject myjobg = null;
        private httpJSONParser json = new httpJSONParser();
        private String customerID;

        public setListViewDataTask (String cst) {
            //  Auto-generated constructor stub
            customerID = cst;
        }

        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα (οι πληροφορίες των οδηγών που ενδιαφέρθηκαν
         * για την συγκεκριμένη κλήση)
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα (οι πληροφορίες των οδηγών που ενδιαφέρθηκαν
         * για την συγκεκριμένη κλήση)
         *
         * */
        protected  ArrayList<DriverDetails> doInBackground(String... params) {

            parameters.add(new BasicNameValuePair("customerDevID",customerID));
            myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/driversListview.
php","POST",parameters);

            try
            {
                for(int i=0;i<myjobg.getJSONArray("drivers").length();i++)
                {
                    DriverDetails item_details = new DriverDetails();

                    item_details.setOrderid( Integer.valueOf(
myjobg.getJSONArray("drivers").getJSONObject(i).get("orderid").toString() )  );
```

```
item_details.setName(myjobg.getJSONArray("drivers").getJSONObject(i).get("name").toString()
);
                    item_details.setDistance(
myjobg.getJSONArray("drivers").getJSONObject(i).get("distance").toString());

item_details.setRate(Float.valueOf(myjobg.getJSONArray("drivers").getJSONObject(i).get("rate
").toString())) ;

item_details.setPlateNumber(myjobg.getJSONArray("drivers").getJSONObject(i).get("taxiPlateNu
mber").toString());

item_details.setDriverDeviceId(myjobg.getJSONArray("drivers").getJSONObject(i).get("driverDe
viceID").toString());
                    item_details.setMyImage(
downloadImage(myjobg.getJSONArray("drivers").getJSONObject(i).get("driverImageUrl").toString
() ) ) ;

                    results.add(item_details);

                }

            }
            catch (JSONException e)
            {
                //  Auto-generated catch block
                e.printStackTrace();
            }

            return results;

    }

        /**
         * @param url το path της εικόνας
         *
         * @info
         * Κατεβάζει την εικόνα απο την βάση για να την βάλει στο listview
         *
         * @details
         * Κατεβάζει την εικόνα απο την βάση για να την βάλει στο listview
         *
         * @return myImage η τρέχον εικόνα
         */
        private Bitmap downloadImage (String url)
        {

            Bitmap myImage = null;
            try {
                InputStream in = new java.net.URL(url).openStream();
                myImage = BitmapFactory.decodeStream(in);
            }
            catch (Exception e)
            {
                e.printStackTrace();

            }
            return myImage;
        }


}
//=======================================================
private static class ViewHolder {
    public TextView txt_DriverName;
    public TextView plateNumber;
    public TextView txt_distance;
    public ImageView itemImage;
    public RatingBar driversRate;
    public Button showComments,selectDriver;
}


//=======================================================================================
====

public class ItemListBaseAdapter extends BaseAdapter {


    private  ArrayList<DriverDetails> driverDetailsArrayList;
```

```
    private LayoutInflater l_Inflater;

    /**
     * @param context Το περιεχόμενο της δραστηριότητας που καλεί το listview
     * @param results Όλα τα δεδομένα του listview
     *
     * @info
     * Παράγει το listview με όλα τα δεδομένα μέσα για να μπορούμε να τα χρησιμοποιήσουμε
     *
     * @details
     * Παράγει το listview με όλα τα δεδομένα μέσα για να μπορούμε να τα χρησιμοποιήσουμε
     */
    public ItemListBaseAdapter(Context context, ArrayList<DriverDetails> results) {
        driverDetailsArrayList = results;
        l_Inflater = LayoutInflater.from(context);
    }

    public int getCount() {
        return driverDetailsArrayList.size();
    }

    public Object getItem(int position) {
        return driverDetailsArrayList.get(position);
    }

    public long getItemId(int position) {
        return position;
    }

    /**
     * @param context position Η θέση του αντικειμένου μέσα στην λίστα
     * @param convertView το κομμάτι της λίστας απο το οποίο θα πάρουμε τα δεδομένα
     * @param parent όλο το Listview απο το οποίο θα πάρουμε δεδομένα
     *
     * @info
     * Βρήσκει και κρατάει τα δεδομένα μίας συγκεκριμένης όψης μέσα στο listview
     *
     * @details
     * Βρήσκει και κρατάει τα δεδομένα μίας συγκεκριμένης όψης μέσα στο listview
     */
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;
        if (convertView == null) {
            convertView = l_Inflater.inflate(R.layout.item_details_drivers_view, null);
            holder = new ViewHolder();
            holder.txt_DriverName = (TextView) convertView.findViewById(R.id.name);
            holder.txt_distance = (TextView) convertView.findViewById(R.id.distance);
            holder.driversRate = (RatingBar) convertView.findViewById(R.id.ratingBar);
            holder.plateNumber = (TextView) convertView.findViewById(R.id.plateNumber);
            holder.itemImage = (ImageView) convertView.findViewById(R.id.photo);
            convertView.setTag(holder);


            ((Button)
convertView.findViewById(R.id.btnShowComments)).setOnClickListener(onShowCommentsListener);
            ((Button)
convertView.findViewById(R.id.btnSelectDriver)).setOnClickListener(onSelectDriverListener);



        } else {
            holder = (ViewHolder) convertView.getTag();
        }


        holder.txt_DriverName.setText(driverDetailsArrayList.get(position).getName());
        holder.plateNumber.setText(driverDetailsArrayList.get(position).getPlateNumber());

holder.txt_distance.setText(String.valueOf(driverDetailsArrayList.get(position).getDistance(
)));
        holder.driversRate.setRating(driverDetailsArrayList.get(position).getRate());
        holder.itemImage.setImageBitmap(driverDetailsArrayList.get(position).getMyImage() );

        return convertView;
    }


    private OnClickListener onShowCommentsListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            final int position = getListView().getPositionForView(v);
```

```
            if (position != ListView.INVALID_POSITION) {

                DriverDetails obj_DriverDetails = driverDetailsArrayList.get( position );
                //Κρατάμε το deviceid  που επιλέχθηκε
                SharedPreferences OrderPrefs = getThisActivity().getSharedPreferences(
                      "myOrder", Context.MODE_PRIVATE);
                SharedPreferences.Editor prefEditor = OrderPrefs.edit();

prefEditor.putString("selectedDriverDevId",obj_DriverDetails.getDriverDeviceId()  );
                prefEditor.commit();


                Intent intent = new Intent(getThisActivity(), ListViewShowComments.class);
                getThisActivity().startActivity(intent);
                //Log.d("name????",String.valueOf(driverDetailsArrayList.get( position
).getName() ));
            }


        }
    };

    private OnClickListener onSelectDriverListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            final int position = getListView().getPositionForView(v);
            if (position != ListView.INVALID_POSITION) {

                DriverDetails obj_DriverDetails = driverDetailsArrayList.get( position );
                new AcceptDriversTask( Integer.valueOf(  obj_DriverDetails.getOrderid()
),getThisActivity() ).execute();
                Toast toast = Toast.makeText(getThisActivity(),"Θα ειδοποιηθείτε όταν φτάσει
ο οδηγός!",Toast.LENGTH_LONG);
                toast.show();

                //Κρατάμε το orderid  που επιλέχθηκε
                SharedPreferences OrderPrefs = getThisActivity().getSharedPreferences(
                      "myOrder", Context.MODE_PRIVATE);

                SharedPreferences.Editor prefEditor = OrderPrefs.edit();

prefEditor.putString("selectedDriverDevId",obj_DriverDetails.getDriverDeviceId()  );
                prefEditor.putString("orderid",String.valueOf(
obj_DriverDetails.getOrderid() ) );
                prefEditor.commit();

                //finish();
                Intent intent = new Intent(getThisActivity(), CustomerStartEndRide.class);
                getThisActivity().startActivity(intent);


            }


        }
    };

}

//================================================================================


/**
 * Background Async Task για να ειδοποιήσει τους οδηγούς
 * */

/**
 * @info
 * Κλάση για να ειδοποιήσει τον οδηγό που επέλεξε ο πελάτης για την παραγγελία
 *
 * @details
 * ΕΚλάση για να ειδοποιήσει τον οδηγό που επέλεξε ο πελάτης για την παραγγελία
 */
protected class AcceptDriversTask extends AsyncTask<String, String, String> {

            private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
            private ArrayList<NameValuePair> parameters2 = new ArrayList<NameValuePair>();
            private ProgressDialog pDialog;
            private int orderID;
            private Activity myAct;
```

```
            public AcceptDriversTask(int orderid, Activity act) {
                //  Auto-generated constructor stub
                orderID = orderid;
                myAct = act;
            }
            public Activity getMyActivity() {
                return myAct;
            }
            /**
             * @info
             * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη.
             *
             * @details
             * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη.
             * */
            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(getMyActivity());
                pDialog.setMessage("Περιμένετε..");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(true);
                pDialog.show();
            }




            /**
             * @info
             * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
             * επιστρέφονται τα αποτελέσματα (γεωργαφικό μήκος & πλάτος του πελάτη και το id
του
             * οδηγού που επιλέξαμε)
             *
             * @details
             * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
             * επιστρέφονται τα αποτελέσματα (γεωργαφικό μήκος & πλάτος του πελάτη και το id
του
             * οδηγού που επιλέξαμε).
             * Βάση το id του οδηγού του στέλνουμε ειδοποίηση ότι έχει επιλεχτεί για την
πραγγελία.
             * Επίσης του στέλνουμε και την τοποθεσία του πελάτη.
             *
             * */
            protected String doInBackground(String... args) {

                parameters.add(new BasicNameValuePair("orderid",String.valueOf(orderID) ));

                JSONObject myjobg = null  ;
                httpJSONParser json = new httpJSONParser();

                try {

                        SharedPreferences prefs =
getMyActivity().getSharedPreferences("fromCustomer", 0);

                        parameters.add(new BasicNameValuePair("customerid",
prefs.getString("customerdevid",null) ) );
                        parameters.add(new BasicNameValuePair("orderid",
String.valueOf(orderID) ) );
                        myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/"+"customerSelec
tDriver.php","POST",parameters);


                    parameters2.add(new
BasicNameValuePair("driverDevIdtoSend",myjobg.get("driverDevID").toString() ) );
                    parameters2.add(new
BasicNameValuePair("customerlat",myjobg.get("customerLat").toString() ) );
                    parameters2.add(new
BasicNameValuePair("customerlon",myjobg.get("customerLon").toString() ) );
                    parameters2.add(new BasicNameValuePair("selectedOrderid",
String.valueOf( orderID ) ) );
```

```
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/mqttClient/notifyDriverF
orAccept.php","POST",parameters2);

                } catch (NumberFormatException e) {
                    // Auto-generated catch block
                    e.printStackTrace();
                } catch (JSONException e) {
                    //  Auto-generated catch block
                    e.printStackTrace();
                }

                return null;









            }

            /**
             * @info
             *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα
             *
             * @details
             * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα
             *
             * **/
            protected void onPostExecute(String file_url) {
                // dismiss the dialog once done

                pDialog.dismiss();
                pDialog = null;

            }
    }



} //end of class file
```

APXEIO ListViewShowComments.java

```
package app.taxiAnytimeCustomer.DriversList;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ListActivity;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
```

```
import android.widget.EditText;
import android.widget.ListView;
import android.widget.TextView;
import app.taxiAnytimeCustomer.R;
import app.taxiAnytimeCustomer.Common.ConnectionDetectorTask;
import app.taxiAnytimeCustomer.Common.globalVariables;
import app.taxiAnytimeCustomer.Common.httpJSONParser;


/**
 * @info
 * Κλάση υπεύθυνη για την διαχείρηση του listview που περιέχει
 * την λίστα όλων των ενδιαφερόμενων οδηγών ταξί σχετικά με μια κλήση.
 */
public class ListViewShowComments extends ListActivity {
    /** Called when the activity is first created. */



protected String driverDevId;
protected ListView myListView ;
protected Activity thisActivity;


public Activity getThisActivity() {
    return thisActivity;
}

/**
 * @info
 * Αρχικοποίηση activity
 *
 * @details
 * Γίνετε έλεγχος για το αν είμαστε συνδεδεμένοι στο internet και αν έχουμε gps ανοικτό
 * Στην συνέχεια εμφανίζει τα σχόλια όλων των πελατών για τον συγκεκριμένο οδηγό
 *
 *
 */
    @Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.driver_listview_show_comments);
        myListView = (ListView)findViewById(android.R.id.list);
        thisActivity = this;

        ArrayList<CommentsDetails> items_details = null;

        SharedPreferences prefs = getSharedPreferences("myOrder", MODE_PRIVATE);
        driverDevId = prefs.getString("selectedDriverDevId",null);

                try {
                    if ( new
ConnectionDetectorTask(this).execute().get().get("connectionState") ){

                        items_details = new setListViewDataTask (driverDevId
).execute().get();

                        myListView.setAdapter(new ItemListBaseAdapter(this, items_details));

                    }
                } catch (InterruptedException e) {

                    e.printStackTrace();
                } catch (ExecutionException e) {

                    e.printStackTrace();
                }

}//onCreate


    @Override
    public ListView getListView() {

        return super.getListView();
    }

//================================================================================
    /**
     * @info
     * Μέθοδοι για να "ετοιμάσουν" το listview
```

```
     *
     * @details
     * Εδώ παίρνουμε τα δεδομένα (σχόλια και ποιοι τα έκαναν)  και τα τοποθετούμε σωστά στο
listview.
     */


class setListViewDataTask extends AsyncTask<String,String,  ArrayList<CommentsDetails>> {

        private ArrayList<CommentsDetails> results = new ArrayList<CommentsDetails>();

        private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
        private JSONObject myjobg = null;
        private httpJSONParser json = new httpJSONParser();
        private String driverID;

        public setListViewDataTask (String cst) {
            //  Auto-generated constructor stub
            driverID = cst;
        }

        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα
         *
         *
         *
         * */
        protected  ArrayList<CommentsDetails> doInBackground(String... params) {

            parameters.add(new BasicNameValuePair("driverDevID",driverID));
            myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/showCustomerComm
ents.php","POST",parameters);

            try{
                for(int i=0;i<myjobg.getJSONArray("usercomments").length();i++){
                    CommentsDetails item_details = new CommentsDetails();


item_details.setName(myjobg.getJSONArray("usercomments").getJSONObject(i).get("fullname").to
String() );
                    item_details.setComment(
myjobg.getJSONArray("usercomments").getJSONObject(i).get("comment").toString() ) ;

                    results.add(item_details);
                }

            }
            catch (JSONException e)
            {
                //  Auto-generated catch block
                e.printStackTrace();
            }

            return results;

    }

}
//=====================================================
private static class ViewHolder {
    public TextView txt_DriverName;
    public EditText comments;


}


//=============================================================================================

public class ItemListBaseAdapter extends BaseAdapter {


    private  ArrayList<CommentsDetails> commentsArrayList;

    private LayoutInflater l_Inflater;

    /**
     * @param context Το περιεχόμενο της δραστηριότητας που καλεί το listview
```

```
     * @param results Όλα τα δεδομένα του listview
     *
     * @info
     * Παράγει το listview με όλα τα δεδομένα μέσα για να μπορούμε να τα χρησιμοποιήσουμε
     *
     * @details
     * Παράγει το listview με όλα τα δεδομένα μέσα για να μπορούμε να τα χρησιμοποιήσουμε
     */
    public ItemListBaseAdapter(Context context, ArrayList<CommentsDetails> results) {
        commentsArrayList = results;
        l_Inflater = LayoutInflater.from(context);
    }

    public int getCount() {
        return commentsArrayList.size();
    }

    public Object getItem(int position) {
        return commentsArrayList.get(position);
    }

    public long getItemId(int position) {
        return position;
    }

    /**
     * @param context position Η θέση του αντικειμένου μέσα στην λίστα
     * @param convertView το κομμάτι της λίστας απο το οποίο θα πάρουμε τα δεδομένα
     * @param parent όλο το Listview απο το οποίο θα πάρουμε δεδομένα
     *
     * @info
     * Βρήσκει και κρατάει τα δεδομένα μίας συγκεκριμένης όψης μέσα στο listview
     *
     * @details
     * Βρίσκει και κρατάει τα δεδομένα μίας συγκεκριμένης όψης μέσα στο listview
     */
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;
        if (convertView == null) {
            convertView = l_Inflater.inflate(R.layout.item_details_show_comments, null);
            holder = new ViewHolder();
            holder.txt_DriverName = (TextView) convertView.findViewById(R.id.name);
            holder.comments = (EditText)convertView.findViewById(R.id.editTextComments);
            convertView.setTag(holder);

        } else {
            holder = (ViewHolder) convertView.getTag();
        }


        holder.txt_DriverName.setText(commentsArrayList.get(position).getName());
        holder.comments.setText( commentsArrayList.get(position).getComment());


        return convertView;
    }


}

//================================================================================

@Override
public void onBackPressed() {
    finish();
    super.onBackPressed();
}


} //end of class file
```

ΠΑΚΕΤΟ app.taxiAnytimeCustomer.PushService

ΑΡΧΕΙΟ PushService.java

```
[…]
// Display the topbar notification
    /**
     * @param text Το εισερχόμενο μύνημα από το mosquitto
     *
     * @info
     * Εμφανίζει το εισερχόμενο μύνημα ως ειδοποίηση στο χρήστη
     *
     * @details
     * Εμφανίζει το εισερχόμενο μύνημα ως ειδοποίηση στο χρήστη.
     * Θέτει τις παραμέτρους ειδοποίησης και χρησιμοποιεί το μηχανισμό intent
     * για να ξεκινήσει η ανάλογη activity
     */
    private void showNotification(String text) {

        Notification n = new Notification();

        n.flags = Notification.FLAG_SHOW_LIGHTS;
        n.flags = Notification.FLAG_AUTO_CANCEL;
        n.defaults = Notification.DEFAULT_ALL;
        n.icon = app.taxiAnytimeCustomer.R.drawable.tick;
        n.when = System.currentTimeMillis();


        // Simply open the parent activity
        Intent intent = new Intent(this,CustomerStartEndRide.class);
        intent.setAction(Intent.ACTION_VIEW);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
        Bundle b = new Bundle();
        b.putInt("driverHasArrived", 1);
        intent.putExtras(b);



        PendingIntent pi = PendingIntent.getActivity(this.getApplicationContext(), 0,
         intent, 0);

        // Change the name of the notification here
        n.setLatestEventInfo(this.getApplicationContext(), NOTIF_TITLE, text, pi);

        mNotifMan.notify(NOTIF_CONNECTED, n);


        startActivity(intent);

    }
[…]
```

ΠΑΚΕΤΟ app.taxiAnytimeCustomer.usersTypeFactory

ΑΡΧΕΙΟ Customer.java

```
package app.taxiAnytimeCustomer.userTypesFactory;

import android.os.Build;


/**
 *@info
 *Πληροφορίες σχετικά με τον πελάτη
 *
 */

public class Customer extends Users {


    private double Cstlat;
    private double Cstlng;
    private String customerDeviceID;



    public Customer() {
        // Auto-generated constructor stub

        Cstlat = 0.0f;
        Cstlng = 0.0f;
        customerDeviceID = generateUniqueID();

    }


public void setLatitude(double cstlat) {
    Cstlat = cstlat;
}
public void setLongitude(double cstlng) {
    Cstlng = cstlng;
}


    public void setDeviceID(String devid) {
        customerDeviceID = devid;

}
    public double getLatitude() {
        return Cstlat;
    }
    public double getLongitude() {
        return Cstlng;
    }

    public String getDeviceID() {

        return customerDeviceID;

    }

    /**
     * @info
     * Συνάρτηση για την εξαγωγή μοναδικού id συσκευής
     *
     * @details
     * Παίρνουμε πληροφορίες από το υλικό της συσκευής(cpu brand,board κτλ) ,
     * τα συνθέτουμε και παράγουμε το τελικό id
     *
     *
     * @return Το μοναδικό id της συσκευής
     */
    private String generateUniqueID()
    {
        String id;
        try
        {
            id = "35" +
                    Build.BOARD.length()%10+ Build.BRAND.length()%10 +
                    Build.CPU_ABI.length()%10 + Build.DEVICE.length()%10 +
                    Build.DISPLAY.length()%10 + Build.HOST.length()%10 +
                    Build.ID.length()%10 + Build.MANUFACTURER.length()%10 +
                    Build.MODEL.length()%10 + Build.PRODUCT.length()%10 +
                    Build.TAGS.length()%10 + Build.TYPE.length()%10 +
```

```
                    Build.USER.length()%10 ; //13 digits
        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }

        return id;
    }


}
```

APXEIO Driver.java

```
package app.taxiAnytimeCustomer.userTypesFactory;

import android.os.Build;


/**
 *@info
 *Πληροφορίες σχετικά με τον οδηγό
 *
 */
public class Driver extends Users  {

    private  double Drvlat;
    private double Drvlng;
    private String driverDeviceID;


    public Driver() {
        // TODO Auto-generated constructor stub
        Drvlat = 0.0;
        Drvlng = 0.0;
        driverDeviceID = generateUniqueID();

    }


    //Sets - gets

    public void setDeviceID(String devid) {
        driverDeviceID = devid;

    }

    public void setLatitude(double drvlat) {
        Drvlat = drvlat;
    }

    public void setLongitude(double dvrlng) {
        Drvlng = dvrlng;
    }

    public double getLatitude() {
        return Drvlat;
    }

    public double getLongitude() {
        return Drvlng;
    }


    public String getDeviceID() {
        return driverDeviceID;
    }


    /**
     * @info
     * Συνάρτηση για την εξαγωγή μοναδικού id συσκευής
     *
     * @details
     * Παίρνουμε πληροφορίες από το υλικό της συσκευής(cpu brand,board κτλ) ,
```

```
     * τα συνθέτουμε και παράγουμε το τελικό id
     *
     *
     * @return Το μοναδικό id της συσκευής
     */
    private String generateUniqueID()
    {
        String id;
        try
        {
            id = "35" +
                    Build.BOARD.length()%10+ Build.BRAND.length()%10 +
                    Build.CPU_ABI.length()%10 + Build.DEVICE.length()%10 +
                    Build.DISPLAY.length()%10 + Build.HOST.length()%10 +
                    Build.ID.length()%10 + Build.MANUFACTURER.length()%10 +
                    Build.MODEL.length()%10 + Build.PRODUCT.length()%10 +
                    Build.TAGS.length()%10 + Build.TYPE.length()%10 +
                    Build.USER.length()%10 ; //13 digits

        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }

        return id;
    }

  }
```

APXEIO Users.java

```
package app.taxiAnytimeCustomer.userTypesFactory;

/**
 * @info
 * Abstract κλάση για τις κοινές μεθόδους οδηγού και πελάτη
 * Χρησιμοποιείται για το factory pattern
 *
 */

public abstract class Users {

     public abstract void setLatitude(double lat);
     public abstract void setLongitude(double lng);
     public abstract void setDeviceID(String devid);
     public abstract double getLatitude();
     public abstract double getLongitude();
     public abstract String getDeviceID();



}
```

APXEIO UsersFactory.java

```
package app.taxiAnytimeCustomer.userTypesFactory;
/**
 * @info
 * Κλάση για την δημιουργία χρηστών με τη μέθοδο factory pattern
 *
 *
 */
public class UsersFactory {

    public static Users createUser(String type)
    {
         if (type.equals("customer")){
               return new Customer();
            }
         else if (type.equals("driver")){
               return new Driver();
            }
            return null;
    }

}
```

```
//Παράδειγμα κλήσης : Users customer = UsersFactory.createUser("customer");
```

ΕΦΑΡΜΟΓΗ ΟΔΗΓΟΥ

ПАКЕТО
app.taxiAnytimeDriver.Common

ΑΡΧΕΙΟ ConnectionDetectorTask.java

```
package app.taxiAnytimeDriver.Common;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.HashMap;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.AsyncTask;



/**
 *
 * @info
 *Κλάση υπεύθυνη για να ελέγχει τη διαθεσιμότητα των
 * δυο server που απαιτούνται για να λειτουργήσει η εφαρμογή
 * καθώς και για το αν είναι συνδεδεμένος ο χρήστης σε wifi/3g/gps
 */
public class ConnectionDetector  extends AsyncTask<String, String , HashMap<String, Boolean>
> {


    public Activity activityContext;
    private ProgressDialog pDialog;
    private int timeOut = 2000 ; //ms
    private int HTTP_GOOD_REQUEST = 200;
    private HashMap<String, Boolean> conState = new HashMap<String, Boolean>();

    public ConnectionDetector(Activity ActCon) {
        activityContext = ActCon;

    }



    /**
         * @info
         * Πριν ξεκινήσει η διαδικασία ελέγχου  εμφανίζεται μήνυμα στην οθόνη χρήστη
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία ελέγχου   εμφανίζεται μήνυμα στην οθόνη χρήστη
         * */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(activityContext);
        pDialog.setMessage("Περιμένετε..");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);
        pDialog.show();
    }



/**
     * @return HashMap με τις καταστάσεις σύνδεσης
     * @info
     * Εδώ γίνεται ο έλεγχος για τη σύνδεση
     * @details
     *
     *
     * */
    @Override
    protected HashMap<String, Boolean> doInBackground(String... args) {

        try {
            conState.put("serverState", ServersAreReachable(activityContext) );
            conState.put("serviceState", isConnectingToInternet(activityContext) );

            if( ServersAreReachable(activityContext) == true &&
                isConnectingToInternet(activityContext) == true){
                conState.put("connectionState", true) ;
```

```
            }
            else{
                conState.put("connectionState",false) ;
            }
        }

        catch(Exception e){
            conState.put("serverState", false );
            conState.put("serviceState", false );
            conState.put("connectionState",false) ;

            e.printStackTrace();
        }

        return conState;
    }

    /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αν τελικώς έχουμε σύνδεση ξεκινά η επόμενη δραστηριότητα,
         * διαφορετικά εμφανίζεται ανάλογο μύνημα και τερματίζει η εφαρμογή.
         *
         * **/
    protected void onPostExecute(HashMap<String, Boolean> result) {
        // dismiss the dialog once done
        pDialog.dismiss();
        pDialog = null;

        //Log.d("servicestate - serverstate-con",String.valueOf(serviceState)+"-
"+String.valueOf(serverState)+"-"+String.valueOf(connectionState));

        if(result.get("serviceState") == false) {
            new showAlertMessage(activityContext ,"Ωχ!Αδυναμία σύνδεσης!!!","Ανοίξτε το GPS
ή/και το wifi/3g");
        }
        else if(result.get("serverState") == false){
            new showAlertMessage(activityContext ,"Ωχ!Αδυναμία σύνδεσης!!!","Αδύναμία
επικοινωνίας με τους servers! Δοκιμάστε αργότερα.");
        }

    }
//=====================================================

/**
 * @info
 * Μέθοδος για τον έλεγχο διαθεσιμότητας του server
 *
 * @details
 * Αρχικά προσπαθεί να κάνει ένα request στην ip του server,
 * θέτοντας ένα timeout 3 sec.
 * Αν είναι σε λειτουργία ο server θα πάρουμε ένα θετικό response (HTTP 200) ,
 * διαφορετικά δεν υπάρχει σύνδεση
 *
 * @return
 * true  : Όταν υπάρχει σύνδεση
 * false : Όταν δεν είναι εφικτή η σύνδεση στον server
 *
 */
private  boolean ServersAreReachable(Activity activContext) {


        final ConnectivityManager connMgr = (ConnectivityManager)
activContext.getSystemService(Context.CONNECTIVITY_SERVICE);
        final NetworkInfo netInfo = connMgr.getActiveNetworkInfo();

        if (netInfo != null && netInfo.isConnected()) {
            //  Some sort of connection is open, check if server is reachable
            try {
                URL url = new URL("http://"+globalVariables.getInstance().getIP_ADDRESS());
                HttpURLConnection urlc = (HttpURLConnection) url.openConnection();

                urlc.setConnectTimeout(timeOut);
                urlc.connect();
                if (urlc.getResponseCode() == HTTP_GOOD_REQUEST) {
                    return true;
                } else { // Anything else is unwanted
                    return false;
```

```
                }
            } catch (IOException e) {

                }
        }

            return true;
    }

//================================================================================
    /**
     * @info
     * Γίνεται έλεγχος για το αν υπάρχει σύνδεση της συσκευής μας με το internet(wifi ή 3G),
     * καθώς και αν το gps είναι ανοικτό.
     *
     * @details
     * Χρησιμοποιούμε τις ενσωματωμένες κλάσεις του android
(LocationManager,ConnectivityManager),
     * για να ελέγξουμε αν υπάρχει ενεργή συνδεσιμότητα
     *
     * @return
     * true  : Αν είναι ανοικτό το GPS και έχουμε πρόσβαση στο internet
     * false : Αν δεν ισχύει έστω και ένα από τα παραπάνω
     */
    private boolean isConnectingToInternet(Activity activContext)
    {

        try {
            LocationManager GpsSservice =
(LocationManager)activContext.getSystemService(Context.LOCATION_SERVICE);
            ConnectivityManager connManager
=(ConnectivityManager)activContext.getSystemService(Context.CONNECTIVITY_SERVICE);
            boolean GpsEnabled =
GpsSservice.isProviderEnabled(LocationManager.GPS_PROVIDER);
            NetworkInfo info = connManager.getActiveNetworkInfo();


            if( !GpsEnabled) {

                return false;
            }
            else if(info.getState() != NetworkInfo.State.CONNECTED ) {

                return false;
            }

        }
        catch(Exception e)
        {
            e.printStackTrace();
            return false;
        }

        return true;
    }



}
```

APXEIO Contact.java

```
package app.taxiAnytimeDriver.Common;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;


import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
```

```
import android.content.SharedPreferences;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;


import app.taxiAnytimeDriver.R;




/**
 * @author fil/george
 * @details  Κλάση υπεύθυνη για τη διαχείριση κλήσεως/sms σε περίπτωση που χρειαστεί
 *
 *
 *
 */

public class Contact  {


    private String mNumber;
    protected Activity actContext;
    protected ProgressDialog pDialog;


    public Contact(Activity theContext) {
        actContext = theContext;
        this.mNumber = null;

    }



    /**
     *
     *
     * @throws ExecutionException
     * @throws InterruptedException
     *
     * @info
     * Πατώντας το κουμπί επικοινωνία μπορεί να επιλεξει sms/κλήση
     *
     * @details
     * Αρχικά λαμβάνουμε τον αριθμό του οδηγού της παραγγελίας , έπειτα εμφανίζεται ένα
dialog box
     * για να επιλέξει ο χρήστης τον τρόπο επικοινωνίας (κλήση/ sms)
     *
     */
    public void makeContact() throws InterruptedException, ExecutionException{

        if ( new ConnectionDetector(actContext).execute().get().get("connectionState") ) {

        SharedPreferences pref = actContext.getSharedPreferences("acceptedCustomer",
Context.MODE_PRIVATE);

        setmNumber ( new GetCellphone("driver",
                    pref.getString("acceptedCustomerDevid","").toString() )
                    .execute().get()
                    );


        final CharSequence[] items = {"Κλήση", "SMS", "Επιστροφή"};
        AlertDialog.Builder builder = new AlertDialog.Builder(actContext);

        builder.setTitle("Επιλέξτε τρόπο επικοινωνίας");
        builder.setItems(items, new DialogInterface.OnClickListener() {

           public void onClick(DialogInterface dialog, int item) {
               if (items[item].equals("Κλήση")) {
                   actContext.startActivity(new Intent(Intent.ACTION_CALL,
Uri.parse("tel:"+getmNumber())));

               }
```

```
                else if (items[item].equals("SMS")) {

                     Intent smsIntent = new Intent(Intent.ACTION_VIEW);
                     smsIntent.putExtra("sms_body", "Taxi anytime!");
                     smsIntent.putExtra("address", getmNumber());
                     smsIntent.setType("vnd.android-dir/mms-sms");

                  actContext.startActivity(smsIntent);

                }
                else
                    Toast.makeText(actContext, items[item], Toast.LENGTH_SHORT).show();
            }

        });

        AlertDialog alert = builder.create();

        alert.show();

    }//end if
}


    public void setmNumber(String mNumber) {
        this.mNumber = mNumber;
    }

    public String getmNumber() {
        return mNumber;

    }




//=============================================================================================
=

    /**
     * @info
     * Μέθοδος για την προσκόμιση του κινητού τηλεφώνου τον συγκεριμένο οδηγού/πελάτη
     *
     * @details
     * Η κλάση "τρέχει " στο παρασκήνιο και επικοινωνεί με τη βάση δεδομένων για την
προσκόμιση
     * του αριθμού κινητου τηλεφώνου
     */

    protected class GetCellphone extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters = new  ArrayList<NameValuePair>() ;
        private JSONObject myjobg = null;;
        private httpJSONParser json = new httpJSONParser();

        private String usrType;
        private String customerDevId;

        public GetCellphone() {


            this.usrType = null;
            this.customerDevId = null;

        }
        public GetCellphone(String userType,String drvDevId) {
            //  Auto-generated constructor stub
            this.usrType = userType;
            this.customerDevId = drvDevId;
        }

        /**
         * Before starting background thread Show Progress Dialog
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(actContext);
            pDialog.setMessage("Loading..");
```

```
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα (ο αριθμός τηλεφώνου)
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφονται τα αποτελέσματα (ο αριθμός τηλεφώνου)
         *
         * */
        protected String doInBackground(String... args) {

            parameters.add(new BasicNameValuePair("client",this.usrType));
            parameters.add(new BasicNameValuePair("Deviceid",this.customerDevId ));


            myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeCall.php","G
ET",parameters);

            try
            {
                if (Integer.parseInt(myjobg.get("success").toString()) == 1)
                {
                    return ( myjobg.get("cellphone").toString() );

                }
                else
                {

                    return "";
                }

            } catch (NumberFormatException e) {
                // Auto-generated catch block
                e.printStackTrace();
            } catch (JSONException e) {
                //  Auto-generated catch block
                e.printStackTrace();
            }


            return null;
        }

        /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * **/
        @Override
        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done
                pDialog.dismiss();

        }

    }



}//end class file
```

APXEIO DriverSplashScreen.java

```
package app.taxiAnytimeDriver.Common;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ImageView;
import app.taxiAnytimeDriver.R;



public class DriverSplashScreen extends Activity {

    private ImageView imageView;
    private int activityTimeOut = 3000; // Χρόνος σε ms




      @Override
      public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash_screen);

        setImageView((ImageView)findViewById(R.id.splashImage));


         try {
              if ( new ConnectionDetector(this).execute().get().get("connectionState") ){
                  startMainActivity();
              }
              else {
                  new showAlertMessage(this ,"Ωχ!Αδυναμία σύνδεσης!!!","Δοκιμάστε
αργότερα");
              }
            } catch (Exception e) {
                e.printStackTrace();
                new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε αργότερα");
            }



    }

//============================================================

private void startMainActivity()
{
    Handler handler = new Handler();

    // run a thread after N seconds to start the home screen
    handler.postDelayed(new Runnable() {

        @Override
        public void run() {

            finish();
            // start the home screen

              Intent intent = new Intent(DriverSplashScreen.this, LoginActivity.class);
              startActivity(intent);


        }

    }, activityTimeOut);
}

public ImageView getImageView() {
    return imageView;
}

public void setImageView(ImageView imageView) {
    this.imageView = imageView;
}
```

```
}
```

APXEIO EditProfileActivity.java

```
package app.taxiAnytimeDriver.Common;

import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeDriver.R;



public class EditProfileActivity extends Activity {

    private EditText txtCellphone,
    txtPassword,txtRepatPassword,txtCarPlate,txtTown,txtImageUrl;

    protected SharedPreferences prefs ;
    private static final String NAME_PATTERN      = "^[a-zA-Zα-ωΑ-Ω]+$";
    private static final String CELLPHONE_PATTERN = "^[0-9]+$";

    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";



      /**
      * @info
      * Αρχικοποίηση menu buttons
      *
      * @details
      * Αρχικοποίηση menu buttons
      */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.layout.menu_items, menu);
        return true;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState){

        super.onCreate(savedInstanceState);
        setContentView(R.layout.edit_profile);


        txtCellphone = (EditText)findViewById(R.id.txtCellphone);
        txtTown = (EditText)findViewById(R.id.txtTown);
        txtCarPlate = (EditText)findViewById(R.id.txtCarPlate);
        txtPassword = (EditText)findViewById(R.id.txtPassword);
        txtRepatPassword = (EditText)findViewById(R.id.txtRepatPassword);
        txtImageUrl = (EditText)findViewById(R.id.txtImageUrl);
```

```
        prefs = getSharedPreferences("fromDriver", MODE_PRIVATE);


        new fetchDataToEditTask( prefs.getString("driverDeviceId","") ).execute();

    }

    @Override
    public void onBackPressed() {

        finish();

        super.onBackPressed();
    }

//============================================================================
protected void jsonToTextEdits(JSONObject jo) throws JSONException{

    if(jo != null){


txtCellphone.setText(jo.getJSONArray("editDetails").getJSONObject(0).get("cellphone").toStri
ng() );

txtTown.setText(jo.getJSONArray("editDetails").getJSONObject(0).get("town").toString() );

txtCarPlate.setText(jo.getJSONArray("editDetails").getJSONObject(0).get("taxiPlateNumber").t
oString() );

txtImageUrl.setText(jo.getJSONArray("editDetails").getJSONObject(0).get("driverImageUrl").to
String() );
    }

}
//============================================================================

    /**
     * @info
     * Έλεγχος ορθότητας δεδομένων για αλλαγή στοιχείων στην εφαρμογή
     *
     * @details
     * Με χρήση κανονικών εκφράσεων (regex) γίνετε ελεγχος ορθότητας δεδομένων.
     *
     * -Το όνομα πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα.
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     * -Το επώνυμο πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο
γράμματα.
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     *  -Το κινητό πρέπει να ειναι τουλάχιστον απο 10 αριθμους. Μόνο αριθμούς.
     *  κανονική έκφραση: ^[0-9]+$
     *
     * -Η πόλη πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα .
     *  κανονική έκφραση: ^[a-zA-Z]+$
     *
     *
     *
     *  -Ο κωδικός πρέπει να ειναι τουλάχιστον 6 ψηφία
     *
     *
     *
     * @return
     * true : έχει δώσει σωστά στοιχεία.
     * false: δεν έχει δώσει σωστά στοιχεία.
     */
    private boolean isValidFormToEdit() {


        if (! txtCellphone.getText().toString().matches(CELLPHONE_PATTERN) ||
txtCellphone.getText().length() < 10)  {
            txtCellphone.setError("Το κινητό πρέπει να αποτελείτε απο αριθμούς");
            return false;
        }
        if (! txtTown.getText().toString().matches(NAME_PATTERN) ||
txtTown.getText().length() < 3) {
            txtTown.setError("Δώστε μια έγκυρη πόλη");
            return false;
        }
```

```
        if (txtPassword.getText().length() < 6) {
            txtPassword.setError("Ο κωδικός πρέπει να είναι τουλάχιστον 6 χαρακτήρες");
            return false;
        }

        if (! txtPassword.getText().toString().equals(txtRepatPassword.getText().toString())
) {
            txtRepatPassword.setError("Δεν ειναι ίδιοι οι κωδικοί πρόσβασης)");
            return false;
        }

        return true;

    }
//=============================================================================================
======
    /**
     * @param fields  Array με τα πεδία edit
     * @info
     * Ελέγχει αν όλα τα editboxes είναι κενά
     * @return
     * true : Αν έστω και ένα πεδίο είναι άδειο
     * false : Αν δεν είναι κανένα άδειο
     */
    private boolean AreEditBoxesEmpty(EditText[] fields){
        for(int i=0; i<fields.length; i++){
            EditText currentField=fields[i];
            if(currentField.getText().toString().length()<=0){
                return true;
            }
        }
        return false;
}


//=========================================================================
    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Αρχίζει την διαδικασία του edit
     *
     * @details
     * Κάνει έλεγχο αν έχουμε gps και σύνδεση στο internet ανοικτά και αν ναι, ξεκινάει την
     * διαδικασία της αλλαγής στοιχείων. Αλλιώς ενημερώνει την χρήστη
     *
     */
    public void onClkEdit(View view) {



        if(isValidFormToEdit() == true &&
                AreEditBoxesEmpty(new EditText[]{txtCellphone,
                txtPassword,txtRepatPassword,txtTown,txtCarPlate}) == false ) {


            try{

                if ( new ConnectionDetector(this).execute().get().get("connectionState") ){
                    new makeEditTask().execute();
                }

            }
            catch(Exception e){
                e.printStackTrace();
                new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε αργότερα");
            }

        }
        else{
            new showAlertMessage(this ,"Προσοχή!!!","Ελέγξτε όλα τα πεδία");
        }

}

//=========================================================================
     /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Καθαρίζει όλα τα πεδία της φόρμας.
```

```
     *
     * @details
     * Καθαρίζει όλα τα πεδία της φόρμας.
     */
    public void onClkClearFields(View view){


            txtCellphone.setText("");
            txtTown.setText("");
            txtCarPlate.setText("");
            txtPassword.setText("");
            txtRepatPassword.setText("");
            txtImageUrl.setText("");
     }

//=========================================================================

    /**
     * @info
     * Κλάση υπεύθυνη για την διαδικασία της εγγραφής χρηστών
     */
class fetchDataToEditTask extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters ;
        private String deviceID;
        private ProgressDialog pDialog;
        protected JSONObject myjobg ;


        public fetchDataToEditTask(String drvID){

            deviceID = drvID;
            }

                /**
                 * @info
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 *
                 * @details
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 * */

         @Override
            protected void onPreExecute() {
                   pDialog = new ProgressDialog(EditProfileActivity.this);
                   pDialog.setIndeterminate(false);
                   pDialog.setMessage("Αλλαγή στοιχείων...");
                   pDialog.setCancelable(true);
                   pDialog.show();

            }

                /**
                 * @info
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * @details
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * */

            @Override
            protected String doInBackground(String... urls) {

                    parameters = new ArrayList<NameValuePair>();

                    try {

                        parameters.add(new BasicNameValuePair("deviceid",deviceID) );
                        parameters.add(new BasicNameValuePair("usertype","driver") );


                        myjobg = null ;
                        httpJSONParser json = new httpJSONParser();
```

```
                        myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/fetchDataToEdit.
php","POST",parameters);

                        //Log.d("editdetails",myjobg.toString() );
                        if(Integer.parseInt(myjobg.get("success").toString()) == 1) {

                            //do something


                        }

                    } catch (Exception e) {

                        e.printStackTrace();
                    }

                    return null;
                }

                /**
                 * @info
                 *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα
                 *
                 * @details
                 * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα και μεταβαίνουμε στην login δραστηριότητα.
                 *
                 * **/
                protected void onPostExecute(String file_url) {
                    // dismiss the dialog once done

                    pDialog.dismiss();
                    pDialog = null;

                    try {
                        jsonToTextEdits ( myjobg );
                    } catch (JSONException e) {

                        e.printStackTrace();
                    }


                }
        }

//================================================================================
/**
 * @info
 * Κλάση υπεύθυνη για την διαδικασία της εγγραφής χρηστών
 */
class makeEditTask extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters ;
        private ProgressDialog pDialog;
        private JSONObject myjobg ;
        private httpJSONParser json ;
        private boolean check;

    /**
    * @info
    * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
    *
    * @details
    * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
    * */

                @Override
                protected void onPreExecute() {
                    super.onPreExecute();
                    pDialog = new ProgressDialog(EditProfileActivity.this);
                    pDialog.setMessage("Αλλαγή στοιχείων...");
                    pDialog.setIndeterminate(false);
                    pDialog.setCancelable(true);
                    pDialog.show();
                }


                /**
                 * @info
```

```
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * @details
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * */

                protected String doInBackground(String... args) {

                try {
                    parameters = new ArrayList<NameValuePair>();

                    parameters.add(new
BasicNameValuePair("cellphone",String.valueOf(txtCellphone.getText())));
                    parameters.add(new
BasicNameValuePair("password",String.valueOf(txtPassword.getText())));
                    parameters.add(new
BasicNameValuePair("town",String.valueOf(txtTown.getText())));
                    parameters.add(new
BasicNameValuePair("taxiplate",String.valueOf(txtCarPlate.getText())));
                    parameters.add(new BasicNameValuePair("deviceid",
prefs.getString("driverDeviceId","") ) );
                    parameters.add(new BasicNameValuePair("imageUrl",
String.valueOf(txtImageUrl.getText())));
                    parameters.add(new BasicNameValuePair("usertype","driver") );

                    myjobg = null;
                    json = new httpJSONParser();
                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeEdit.php","P
OST",parameters);

                        if(Integer.parseInt(myjobg.get("success").toString()) == 1) {

                            check = true;

                        }
                        else{
                            check = false;
                        }

                    } catch (NumberFormatException e) {

                        e.printStackTrace();
                    } catch (JSONException e) {

                        e.printStackTrace();
                    }

                    return null;
                }

                /**
                 * @info
                 *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα
                 *
                 * @details
                 * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα και μεταβαίνουμε στην login δραστηριότητα.
                 *
                 * **/
                protected void onPostExecute(String file_url) {
                    pDialog.dismiss();
                    pDialog = null;

                    if(this.check == true){
                        Toast toast = Toast.makeText(getApplicationContext(),"Ολοκληρώθηκαν
οι αλλαγές!", Toast.LENGTH_LONG);
                        toast.show();
                    }
                    else {
                        Toast toast = Toast.makeText(getApplicationContext(),"Μη επιτυχής
αλλαγή στοιχείων!!!", Toast.LENGTH_LONG);
                        toast.show();
                    }
                }
```

```
        }

//=========================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
 * @info
 * Κώδικας σχετικά με το hardware button menu
 *
 *
 * @details
 * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
 * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
 * Έχει τις δυνατότητες bookmark, share & profile.
 *
 * */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
    switch (item.getItemId())
    {

    case R.id.menu_contact:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_report:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;

    case R.id.menu_bookmark:
        saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
        return true;
    case R.id.menu_share:
        share();
        return true;
    case R.id.menu_profile:
        Toast.makeText(this, "Είστε ήδη εδώ!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_aboutTheApp:
        aboutTheApp();
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }

}



/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

      Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
      startActivity(intent);
}

/**
 * @param c το περιεχόμενο της δραστηριότητας.
 * @param title ο τίτλος του σελιδοδείκτη.
 * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
 *
 * @info
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 *
 * @details
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 */
public static final void saveBookmark(Context c, String title, String url) {
    Intent i = new Intent(Intent.ACTION_INSERT,
                  android.provider.Browser.BOOKMARKS_URI);
    i.putExtra("title", title);
    i.putExtra("url", url);
```

```
    c.startActivity(i);
}

/**
 * @info
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
 * με την εφαρμογή
 *
 * @details
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
 * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
 * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
 */
public void share () {
    Intent share = new Intent(Intent.ACTION_SEND);
    String shareBody = "Taxi Anytime";
    share.setType("text/plain");
    share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
    share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    startActivity(Intent.createChooser(share, "Send via"));

}

/**
 * @info
 * Πληροφορίες σχετικά με την εφαρμογή.
 *
 * @details
 * Πληροφορίες σχετικά με την εφαρμογή.
 */
public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

}


/**
 * @info
 * Αλλαγή προφίλ
 *
 * @details
 * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
 */
public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);
}

}
```

ΑΡΧΕΙΟ globalVariables.java

```
package app.taxiAnytimeDriver.Common;


/** @info
 * Σε αυτο το αρχειο γινετε αρχικοποιηση των global μεταβλητων.
 * Γινετε χρηση του Singleton pattern για να εχουμε μοναδικη αναφορα στην
 * συγκεκριμενη κλαση
 *
 */
public class globalVariables {

        //MQTT BROKER
        private String MQTT_IP_ADDRESS ;
```

```
        private int MQTT_PORT;

        //WAMP SERVER
        private    String IP_ADDRESS ;
        private    int HTTP_PORT;
        private    String BASE_PATH;


        //instance
        private static globalVariables instance = null;

        public globalVariables() {

            MQTT_IP_ADDRESS = "192.168.2.100";
            MQTT_PORT = 1883;

            IP_ADDRESS = "192.168.2.100";
            HTTP_PORT = 80;
            BASE_PATH =
"http://"+IP_ADDRESS+":"+String.valueOf(HTTP_PORT)+"/taxiAnytime_server";

        }


        public static globalVariables getInstance() {
            if (instance == null)
                instance = new globalVariables();
            return instance;
        }


        public String getBASE_PATH() {
            return BASE_PATH;
        }

        public int getHTTP_PORT() {
            return HTTP_PORT;
        }

        public String getIP_ADDRESS() {
            return IP_ADDRESS;
        }

        public String getMQTT_IP_ADDRESS() {
            return MQTT_IP_ADDRESS;
        }

        public int getMQTT_PORT() {
            return MQTT_PORT;
        }


}
```

APXEIO httpJsonParser.java

```
package app.taxiAnytimeDriver.Common;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;
```

```
import android.util.Log;


/**
 *@info
 *  Κλάση για τη διαχείριση http post/get στη βάση mysql (Σε μορφή json)
 *
 * Στην ουσία μετατρέπει την απόκριση json από ένα http request που γίνεται ,
 * σε αντικείμενο JSON σε java ώστε να μπορούμε να το επεξεργαστούμε.
 *
 *@external_source
 *Αυτή η κλάση χρησιμοποιήτε ευρέως από όλους όσους γράφουν εφαρμογές android
 *και είναι για την επικοινωνία με την βάση δεδομένων
 *
 *πηγή: www.androidhive.info
 */
public class httpJSONParser
{

    InputStream is ;
    JSONObject jObj ;
    String json ;
    final int BUFFER_SIZE = 8192;


    public httpJSONParser()
    {
        is = null;
        jObj = null;
        json = null;

    }

    // function get json from url
    // by making HTTP POST or GET mehtod
    /**
     * @param url η διεύθυνση των php scripts μέσα στον server μας, τα οποία εμείς γράψαμε.
     * @param method μέθοδος επικοινωνίας με τον server (post ή get)
     * @param params οι είσοδοι στα php scripts μας, οι παραμέτροί τους.
     *
     * @info
     * Γράφει και διαβάζει απο την βάση δεδομένων
     *
     * @details
     * Αφού γίνει η σύνδεση με την βάση γίνετε έλεγχος τι θέλουμε να κάνουμε.
     * Αν η μέθοδος ειναι post, στέλνουμε και αποθηκεύουμε δεδομένα στην βάση.
     * Αν η μέθοδος ειναι get, λαμβάνουμε αποτελέσματα από την βάση και τα επιστρέφουμε.
     *
     * @return
     * JSONObject : πρακτικά είναι το αποτέλεσμα των php scripts σε μορφή JSON
     */
    public JSONObject makeHttpRequest(String url, String method,List<NameValuePair> params)
    {

        // Making HTTP request
        try {

            // check for request method
            if(method == "POST"){
                // request method is POST
                // defaultHttpClient
                DefaultHttpClient httpClient = new DefaultHttpClient();

                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new UrlEncodedFormEntity(params,"UTF-8"));

                HttpResponse httpResponse = httpClient.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }
            else if(method == "GET")
            {
                // request method is GET
                DefaultHttpClient httpClient = new DefaultHttpClient();
                String paramStringGet = URLEncodedUtils.format(params, "UTF-8");
                url += "?" + paramStringGet;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse = httpClient.execute(httpGet);
```

```
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();
            }


        }
        catch (UnsupportedEncodingException e)
        {
            e.printStackTrace();

        }
        catch (ClientProtocolException e)
        {
            e.printStackTrace();
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }

        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(is,"UTF-
8"),BUFFER_SIZE);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();
            json = sb.toString();
        }
        catch (Exception e)
        {
            Log.e("Buffer Error", "Error converting result " + e.toString());
        }

        // try parse the string to a JSON object
        try
        {
            jObj = new JSONObject(json);
        }
        catch (JSONException e)
        {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }

        // return JSON String
        return jObj;

    }
}
```

APXEIO LoginActivity.java

```
package app.taxiAnytimeDriver.Common;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
```

```
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;


import app.taxiAnytimeDriver.R;
import app.taxiAnytimeDriver.Driver.DriverActivity;
import app.taxiAnytimeDriver.userTypesFactory.Users;
import app.taxiAnytimeDriver.userTypesFactory.UsersFactory;

/**
 * @info
 * Είσοδος Χρήστη
 * Ελέγχεται η είσοδος του χρήστη με σύστημα login,
 * και αν είναι επιτυχής συνεχίζουμε στην εφαρμογή.
 *
 */
public class LoginActivity extends Activity {

    private EditText txtUsername,txtPassword;
    private CheckBox saveLoginCheckBox;
    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";
    private SharedPreferences loginPreferences;
    //Ο editor ειναι για να μπορουμε να τροποποιησουμε τα δεδομενα που εχουμε αποθηκευμενα.
    private SharedPreferences.Editor loginPrefsEditor;
    public static final String PREFERENCE_FILENAME = "LoginInfo";

    boolean isLoginDataCorrect = false;
    private Boolean saveLogin;
    private String username,password;



      /**
       * @info
       * Αρχικοποίηση menu buttons
       *
       * @details
       * Αρχικοποίηση menu buttons
       */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.layout.menu_items, menu);
        return true;
    }

    /**
     * @info
     * Αρχικοποίηση activity
     *
     * @details
     * Αντιστοιχούμε τις όψεις των xml αρχείων με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
     * Έπειτα χρησιμοποιούμε τον εσωτερικό μηχανισμό του android (SharedPreferences) για την
,
     * αποθήκευση στοιχείων(login) στο αρχείο ώστε να είναι προσβάσιμο στην εφαρμογή.
     *
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login_layout);

        txtUsername = (EditText)findViewById(R.id.editTextUsername);
        txtPassword = (EditText)findViewById(R.id.editTextPassword);
        saveLoginCheckBox = (CheckBox)findViewById(R.id.saveLoginCheckBox);

        //Παιρνουμε τα δεδομενα (αν υπαρχουν) απο το αρχειο που ορισαμε
        loginPreferences = getSharedPreferences(PREFERENCE_FILENAME, MODE_PRIVATE);
        //Βαζουμε τον editor να μπορει να βλεπει για να κανει επεξεργασια -το συγκεκριμενο-
preference
        loginPrefsEditor = loginPreferences.edit();

        saveLogin = loginPreferences.getBoolean("saveLogin", false);
        if (saveLogin == true) {
            txtUsername.setText(loginPreferences.getString("username", ""));
            txtPassword.setText(loginPreferences.getString("password", ""));
            saveLoginCheckBox.setChecked(true);
```

```
        }
    }

    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @throws InterruptedException
     * @throws ExecutionException
     *
     * @info
     * Γίνεται η ορθή διαδικασία εισαγωγής χρήστη στην εφαρμογή
     *
     * @details
     * Οταν πατησει το κουμπι login ο χρηστης γινετε ελεγχος των στοιχειων που εδωσε
     * και αν αυτα ειναι σωστα τον βαζει στην εφαρμογη. Σε διαφορετικη περιπτωση
     * εμφανιζει μηνυμα λαθους.Επίσης γίνεται έλεγχος για τον αν υπάρχει σύνδεση της
συσκευής
     * τόσο με το internet όσο και με τον server πριν γίνει οτιδήποτε
     *
     */
    public void onClkLogin(View view) throws InterruptedException, ExecutionException
    {


            username = txtUsername.getText().toString();
            password = txtPassword.getText().toString();

            if( !fieldsAreEmpty(username,password) ){

                try{
                    if ( new
ConnectionDetector(this).execute().get().get("connectionState") ){
                        new checkLoginTask(username,password).execute();
                    }

                }
                catch(Exception e){
                    new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε
αργότερα");
                    e.printStackTrace();
                }

             }

    }

 private boolean fieldsAreEmpty(String username,String password){

        if(username == null || password == null){
            return true;
        }
        else{
            return false;
        }
 }

//=============================================================================
 /**
  * @param loginComplete μας δείχνει αν το Login έχει γίνει επιτυχώς
  *
  * @info Ελέγχουμε αν θέλει να αποθηκευτεί το login και αν είναι σωστό γίνεται μετάβαση
στην επόμενη activity
  *
  * @details
  * Αν έχει ολοκληρωθεί το Login επιυχώς και ο χρήστης έχει επιλέξει την "remember me"
επιλογή
  * γράφουμε τα στοιχεία του στο SharedPreferences αρχείο ώστε να τα τραβήξουμε την επόμενη
  * φορά που θα τρέξει η εφαρμογή
  */
    protected void saveLoginData(boolean loginComplete)
    {

        if(loginComplete == true) {

            if (saveLoginCheckBox.isChecked()) {
                loginPrefsEditor.putBoolean("saveLogin", true);
                loginPrefsEditor.putString("username", username);
                loginPrefsEditor.putString("password", password);
                loginPrefsEditor.commit();
            }
            else{
```

```
                loginPrefsEditor.clear();
                loginPrefsEditor.commit();
            }

            finish();
            Intent i = new Intent(getApplicationContext(),DriverActivity.class);
            startActivity(i);
        }
       else {// if(loginComplete == false)
           new showAlertMessage(this ,"Σφάλμα εισόδου!!!","Δώσατε λάθος στοιχεία ή είστε
μπλοκαρισμένος!");
            }


    }


  //=============================================================================

    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Γίνετε μετάβαση την δραστηριότητα του register
     *
     * @details
     * Γίνετε μετάβαση την δραστηριότητα του register
     */
    public void onClkGoToRegister(View view){


        finish();
        Intent i = new Intent(getApplicationContext(),RegisterDriverActivity.class);
        startActivity(i);

    }
//=====================================================================


    /**
     * @info
     * Η κλάση αυτή είναι υπεύθυνη για τον έλεγχο ορθότητας των δεδομένων που
     * έδωσε ο χρήστης.
     *
     */
    class checkLoginTask extends AsyncTask<String, Boolean, String> {

        private ProgressDialog Dialog;

        private String Username;
        private String Password;
        private boolean loginState;


        public checkLoginTask() {

            Username = null;
            Password = null;
            loginState = false;
        }

        public checkLoginTask(String usrname,String pass) {
            // Auto-generated constructor stub
            Username = usrname;
            Password = pass;

        }


        /**
         * @info
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
         *
         * @details
         * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη χρήστη
         */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            Dialog = new ProgressDialog(LoginActivity.this);
```

```
            Dialog.setMessage("Είσοδος..");
            Dialog.setIndeterminate(false);
            Dialog.setCancelable(true);
            Dialog.show();
        }

           /**
         * @info
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφεται true ή false
         *
         * @details
         * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και μας
         * επιστρέφεται true ή false.
         *
         * @return
         * true : ο χρήστης έδωσε σωστά δεδομένα
         * false: ο χρήστης δεν έδωσε σωστά δεδομένα
         */
        protected String doInBackground(String... args) {

            ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>(4);

            Users driver = UsersFactory.createUser("driver");

            parameters.add(new BasicNameValuePair("username",Username ));
            parameters.add(new BasicNameValuePair("password",Password ));
            parameters.add(new BasicNameValuePair("deviceid","/"+driver.getDeviceID() ));
            parameters.add(new BasicNameValuePair("type","driver"));

            JSONObject myjobg = null;
            httpJSONParser json = new httpJSONParser();

            try {

                 myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeLogin.php","
POST",parameters);

                if(Integer.parseInt(myjobg.get("success").toString()) == 1) {


                     loginState = true;

                }
                else {

                    loginState = false;

                }
            }
            catch (Exception e) {
                e.printStackTrace();
                loginState = false;
            }


            return null;
        }

        /**
         * @info
         *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * @details
         * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
         * συνεχίζεται στο κύριο νήμα
         *
         * **/
        protected void onPostExecute(String file_url) {
            // dismiss the dialog once done
            Dialog.dismiss();

            saveLoginData ( loginState );


        }

    }
//================================================================================
```

```
/**
 * @param item η επιλογή που έκανε ο χρήστης απο το μενού
 *
* @info
* Κώδικας σχετικά με το hardware button menu
*
*
* @details
* Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
* με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
* Έχει τις δυνατότητες bookmark, share & profile.
*
* */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
// Single menu item is selected do something
// Ex: launching new activity/screen or show alert message
switch (item.getItemId())
{

case R.id.menu_contact:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;
case R.id.menu_report:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;

case R.id.menu_bookmark:
    saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
    return true;
case R.id.menu_share:
    share();
    return true;
case R.id.menu_profile:
    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;
case R.id.menu_aboutTheApp:
    aboutTheApp();
    return true;
default:
    return super.onOptionsItemSelected(item);
}

}



/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

  Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
  startActivity(intent);
}

/**
* @param c το περιεχόμενο της δραστηριότητας.
* @param title ο τίτλος του σελιδοδείκτη.
* @param url το site στο οποίο θα κάνει σελιδοδείκτη.
*
* @info
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*
* @details
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*/
public static final void saveBookmark(Context c, String title, String url) {
Intent i = new Intent(Intent.ACTION_INSERT,
             android.provider.Browser.BOOKMARKS_URI);
i.putExtra("title", title);
i.putExtra("url", url);
c.startActivity(i);
}

/**
* @info
```

```
    * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
    * με την εφαρμογή
    *
    * @details
    * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
    * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί είναι καθαρά θέμα τι εφαρμογές
    * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
    */
    public void share () {
    Intent share = new Intent(Intent.ACTION_SEND);
    String shareBody = "Taxi Anytime";
    share.setType("text/plain");
    share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
    share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    startActivity(Intent.createChooser(share, "Send via"));
    }

    /**
    * @info
    * Πληροφορίες σχετικά με την εφαρμογή.
    *
    * @details
    * Πληροφορίες σχετικά με την εφαρμογή.
    */
    public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

    }

    /**
    * @info
    * Αλλαγή προφίλ
    *
    * @details
    * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
    */
    public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);

    }
}
```

APXEIO RegisterActivity.java

```
package app.taxiAnytimeDriver.Common;

import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
```

```
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;



import app.taxiAnytimeDriver.R;
import app.taxiAnytimeDriver.userTypesFactory.Users;
import app.taxiAnytimeDriver.userTypesFactory.UsersFactory;

/**
 * @info
 * Κλάση υπεύθυνη για ότι έχει να κάνει με το register νέων χρηστών.
 *
 */
public class RegisterDriverActivity extends Activity {

    private EditText txtName,txtSirName,txtCellphone,txtBirthday,
                     txtTown,txtCarPlate,txtUsername,txtPassword,
                     txtRepatPassword,txtEmail;


    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";
    private static final String NAME_PATTERN      = "^[a-zA-Zα-ωΑ-Ω]+$";
    private static final String CELLPHONE_PATTERN = "^[0-9]+$";
    private static final String USERNAME_PATTERN  = "^[a-zA-Z][[a-zA-Z][0-9]]*$";
    private static final String EMAIL_PATTERN     = "^[a-z0-9_\\+-]+(\\.[a-z0-9_\\+-]+)*@[a-
z0-9-]+(\\.[a-z0-9-]+)*\\.([a-z]{2,4})$";



      /**
       * @info
       * Αρχικοποίηση menu buttons
       *
       * @details
       * Αρχικοποίηση menu buttons
       */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.layout.menu_items, menu);
        return true;
    }




    /**
     * @info
     * Αρχικοποίηση activity
     *
     * @details
     * Αντιστοιχούμε τις όψεις των xml αρχείων με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
     *
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.register);


        txtName = (EditText)findViewById(R.id.txtName);
        txtSirName = (EditText)findViewById(R.id.txtSirName);
        txtCellphone = (EditText)findViewById(R.id.txtCellphone);
        txtBirthday = (EditText)findViewById(R.id.txtBirthday);
        txtTown = (EditText)findViewById(R.id.txtTown);
        txtCarPlate = (EditText)findViewById(R.id.txtCarPlate);
        txtUsername = (EditText)findViewById(R.id.txtUsername);
        txtPassword = (EditText)findViewById(R.id.txtPassword);
        txtRepatPassword = (EditText)findViewById(R.id.txtRepatPassword);
        txtEmail = (EditText)findViewById(R.id.txtEmail);

    }
```

```
     /**
      * @info
      * Οταν πατήσει το hardware back (κουμπί συσκευής) τον επιστρέφει στο login
      *
      * @details
      * Οταν πατήσει το hardware back (κουμπί συσκευής) τον επιστρέφει στο login
      *
      */
     @Override
     public void onBackPressed() {

         finish();
         Intent i = new Intent(getApplicationContext(),LoginActivity.class);
         startActivity(i);


         super.onBackPressed();
     }



     /**
      * @info
      * Έλεγχος ορθότητας δεδομένων για εγγραφή στην εφαρμογή
      *
      * @details
      * Με χρήση κανονικών εκφράσεων (regex) γίνετε ελεγχος ορθότητας δεδομένων.
      *
      * -Το όνομα πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα.
      *  κανονική έκφραση: ^[a-zA-Z]+$
      *
      * -Το επώνυμο πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο
γράμματα.
      *  κανονική έκφραση: ^[a-zA-Z]+$
      *
      *  -Το κινητό πρέπει να ειναι τουλάχιστον απο 10 αριθμους. Μόνο αριθμούς.
      *  κανονική έκφραση: ^[0-9]+$
      *
      * -Η πόλη πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αποτελείτε μονο απο γράμματα .
      *  κανονική έκφραση: ^[a-zA-Z]+$
      *
      * -Το username πρέπει να ειναι πάνω απο 2 χαρακτήσες και να αρχίζει με γράμμα ή κάτω
παύλα.
      *  κανονική έκφραση: ^[a-zA-Z][[a-zA-Z][0-9]]*$
      *
      *  -Ο κωδικός πρέπει να ειναι τουλάχιστον 6 ψηφία
      *
      *  -Το email πρεπει να ειναι τουλάχιστον 4 χαρακτήρες, να αρχίζει με γράμμα,αριθμό,_,+
ή -, πρέπει να υπάρχει
      *  το @ και η . και μετά την . να έχουμε απο 2 έως 4 χαρακτήρες
      *  κανονική έκφραση: ^[a-z0-9_\\+-]+(\\.[a-z0-9_\\+-]+)*@[a-z0-9-]+(\\.[a-z0-9-
]+)*\\.([a-z]{2,4})$
      *
      * @return
      * true : έχει δώσει σωστά στοιχεία.
      * false: δεν έχει δώσει σωστά στοιχεία.
      */
     private boolean isValidFormToRegister() {


         if (! txtName.getText().toString().matches(NAME_PATTERN) ||
txtName.getText().length() < 3)    {
             txtName.setError("Το όνομα πρέπει να αποτελείται απο γράμματα");
             return false;
         }

         else if (! txtSirName.getText().toString().matches(NAME_PATTERN) ||
txtSirName.getText().length() < 3 ) {
             txtSirName.setError("Το επώνυμο πρέπει να αποτελείται απο γράμματα");
             return false;
         }

         else if (! txtCellphone.getText().toString().matches(CELLPHONE_PATTERN) ||
txtCellphone.getText().length() != 10)  {
             txtCellphone.setError("Το κινητό πρέπει να αποτελείται απο 10 αριθμούς");
             return false;
         }

         else if (! txtTown.getText().toString().matches(NAME_PATTERN) ||
txtTown.getText().length() < 3) {
```

```
            txtTown.setError("Δώστε μια έγκυρη πόλη");
            return false;
        }

        else if (! txtUsername.getText().toString().matches(USERNAME_PATTERN) ||
txtUsername.getText().length() < 3) {
            txtUsername.setError("Το username δεν είναι έγκυρο");

            Log.d("username?","here!!!");

            return false;
        }

        else if (txtPassword.getText().length() < 6) {
            txtPassword.setError("Ο κωδικός πρέπει να είναι τουλάχιστον 6 χαρακτήρες");
            return false;
        }


        else if (!txtPassword.getText().toString().equals(
txtRepatPassword.getText().toString() ) ) {
            txtRepatPassword.setError("Δεν ειναι ίδιοι οι κωδικοί πρόσβασης)");
            return false;
        }

        else if(! txtEmail.getText().toString().matches(EMAIL_PATTERN) ||
txtEmail.getText().length() < 4)  {
            txtEmail.setError("Δώστε ενα έγκυρο email");
            return false;
        }
        else {

            return true;
        }

    }

    /**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Αρχίζει την διαδικασία του register
     *
     * @details
     * Κάνει έλεγχο αν έχουμε gps και σύνδεση στο internet ανοικτά και αν ναι, ξεκινάει την
     * διαδικασία για το register. Αλλιώς ενημερώνει την χρήστη
     *
     */
public void register(View view) {

        if(isValidFormToRegister() == true) {


            try{
                 if ( new ConnectionDetector(this).execute().get().get("connectionState") ){
                     new makeRegister().execute();
                 }
            }
            catch(Exception e){
                e.printStackTrace();
                new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Δοκιμάστε αργότερα");
            }
        }

}

//============================================================================
/**
     * @param view Η όψη στην οποία βρίσκεται το κουμπί
     *
     * @info
     * Καθαρίζει όλα τα πεδία της φόρμας.
     *
     * @details
     * Καθαρίζει όλα τα πεδία της φόρμας.
     */
     public void clearFields(View view){

            txtName.setText("");
            txtSirName.setText("");
            txtCellphone.setText("");
```

```
            txtBirthday.setText("");
            txtTown.setText("");
            txtCarPlate.setText("");
            txtUsername.setText("");
            txtPassword.setText("");
            txtRepatPassword.setText("");
            txtEmail.setText("");
     }

//=============================================================================================

     /**
         * @info
         * Κλάση υπεύθυνη για την διαδικασία της εγγραφής χρηστών
         */
    class makeRegister extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters ;
        private JSONObject myjobg ;
        private httpJSONParser json ;
        private boolean check;
        private ProgressDialog pDialog;

                public makeRegister() {
                    //Auto-generated constructor stub

                    parameters = new ArrayList<NameValuePair>();
                    myjobg = null;
                    json = new httpJSONParser();


                }
                /**
                 * @info
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 *
                 * @details
                 * Πριν ξεκινήσει η διαδικασία προσκόμισης  εμφανίζεται μήνυμα στην οθόνη
χρήστη
                 * */


                @Override
                protected void onPreExecute() {
                    super.onPreExecute();
                    pDialog = new ProgressDialog(RegisterDriverActivity.this);
                    pDialog.setMessage("Εγγραφή..");
                    pDialog.setIndeterminate(false);
                    pDialog.setCancelable(true);
                    pDialog.show();
                }

                /**
                 * @info
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * @details
                 * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request και στέλνουμε
τα δεδομένα
                 * του χρήστη για εγγραφή στον server.
                 *
                 * */
                @Override
                protected String doInBackground(String... args) {


                    Users driver = UsersFactory.createUser("driver");


                    parameters.add(new
BasicNameValuePair("name",String.valueOf(txtName.getText())));
                    parameters.add(new
BasicNameValuePair("sirname",String.valueOf(txtSirName.getText())));
                    parameters.add(new
BasicNameValuePair("cellphone",String.valueOf(txtCellphone.getText())));
                    parameters.add(new
BasicNameValuePair("birthday",String.valueOf(txtBirthday.getText())));
```

```
                    parameters.add(new
BasicNameValuePair("username",String.valueOf(txtUsername.getText())));
                    parameters.add(new
BasicNameValuePair("password",String.valueOf(txtPassword.getText())));
                    parameters.add(new
BasicNameValuePair("town",String.valueOf(txtTown.getText())));
                    parameters.add(new
BasicNameValuePair("taxiplate",String.valueOf(txtCarPlate.getText())));
                    parameters.add(new
BasicNameValuePair("mail",String.valueOf(txtEmail.getText())));

                    parameters.add(new
BasicNameValuePair("deviceid","/"+driver.getDeviceID()) );
                    parameters.add(new BasicNameValuePair("usertype","driver") );


                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeRegister.php
","POST",parameters);

                    try {
                        if(Integer.parseInt(myjobg.get("success").toString()) == 1) {

                            //do something
                            check = true;
                        }
                        else {
                            check = false;
                        }
                    } catch (NumberFormatException e) {
                        //  Auto-generated catch block
                        e.printStackTrace();
                    } catch (JSONException e) {
                        //  Auto-generated catch block
                        e.printStackTrace();
                    }


                    return null;
                }

                /**
                 * @info
                 *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα
                 *
                 * @details
                 * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
                 * συνεχίζεται στο κύριο νήμα και μεταβαίνουμε στην login δραστηριότητα.
                 *
                 * **/
                @Override
                protected void onPostExecute(String file_url) {
                    // dismiss the dialog once done
                    pDialog.dismiss();
                    pDialog = null;

                    if(check == true){

                        Toast toast = Toast.makeText(RegisterDriverActivity.this,"Επιτυχής
Εγγραφή!!!", Toast.LENGTH_SHORT);
                        toast.show();

                        finish();
                        Intent i = new Intent(getApplicationContext(),LoginActivity.class);
                        startActivity(i);

                    }
                    else{
                        new showAlertMessage(RegisterDriverActivity.this ,"Ωχ! Δώσατε λάθος
στοιχεία ή υπάρχει ήδη!!","Δοκιμάστε αργότερα");
                    }

                }

            }

//=============================================================================
    //menus

        /**
```

```
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
 * @info
 * Κώδικας σχετικά με το hardware button menu
 *
 *
 * @details
 * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
 * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
 * Έχει τις δυνατότητες bookmark, share & profile.
 *
 * */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
    switch (item.getItemId())
    {

    case R.id.menu_contact:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_report:

        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;

    case R.id.menu_bookmark:
        saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
        return true;
    case R.id.menu_share:
        share();
        return true;
    case R.id.menu_profile:
        Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
        return true;
    case R.id.menu_aboutTheApp:
        aboutTheApp();
        return true;
    default:
        return super.onOptionsItemSelected(item);
    }

}



/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

      Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
      startActivity(intent);
}

/**
 * @param c το περιεχόμενο της δραστηριότητας.
 * @param title ο τίτλος του σελιδοδείκτη.
 * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
 *
 * @info
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 *
 * @details
 * Κάνει σελιδοδείκτη το site της εφαρμογής.
 */
public static final void saveBookmark(Context c, String title, String url) {
    Intent i = new Intent(Intent.ACTION_INSERT,
                android.provider.Browser.BOOKMARKS_URI);
    i.putExtra("title", title);
    i.putExtra("url", url);
    c.startActivity(i);
}

/**
 * @info
 * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
```

```
     * με την εφαρμογή
     *
     * @details
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
     * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms
κλπ)
     */
    public void share () {
        Intent share = new Intent(Intent.ACTION_SEND);
        String shareBody = "Taxi Anytime";
        share.setType("text/plain");
        share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
        share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
        startActivity(Intent.createChooser(share, "Send via"));

    }

    /**
     * @info
     * Πληροφορίες σχετικά με την εφαρμογή.
     *
     * @details
     * Πληροφορίες σχετικά με την εφαρμογή.
     */
    public void aboutTheApp() {

        setContentView(R.layout.menu_about);
            TextView aboutTextView = (TextView)findViewById(R.id.about);
            aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                    "Εκδοση 1.0<br> \n" +
                    "Copyright 2013<br>  \n" +
                    "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                    "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                    "Χατζημιχαήλ Γεώργιος <br> \n" +
                    "Τσεγγελίδης Φίλανδρος <br>"));

            aboutTextView.setLinkTextColor(Color.WHITE);
            Linkify.addLinks(aboutTextView, Linkify.ALL);

    }


    /**
     * @info
     * Αλλαγή προφίλ
     *
     * @details
     * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
     */
    public void changeProfile(){

        Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
        startActivity(intent);
    }



}
```

APXEIO ReportActivity.java

```
package app.taxiAnytimeDriver.Common;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONException;
import org.json.JSONObject;
```

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnCancelListener;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeDriver.R;



/**
 * @info
 * Κλάση υπεύθυνη για την αναφορά οδηγού απο τον πελάτη.
 *
 */


public class ReportActivity extends Activity implements OnItemSelectedListener {

     private Spinner spinner ;
     private EditText edtReason;
     private ImageButton imgBtnReport;

    private String RptReasons ;
    private String totalReasons;
    private ProgressDialog pDialog;

    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";



      /**
      * @info
      * Αρχικοποίηση menu buttons
      *
      * @details
      * Αρχικοποίηση menu buttons
      */
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater menuInflater = getMenuInflater();
        menuInflater.inflate(R.layout.menu_items, menu);
        return true;
    }

     /**
  * @info
  * Αρχικοποίηση activity
  *
  * @details
  * Αρχικοποίηση activity
  */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.report);

         spinner = (Spinner) findViewById(R.id.ReportReasons);
         edtReason=(EditText)findViewById(R.id.comment);
         imgBtnReport = (ImageButton)findViewById(R.id.imageButtonSubmitReport);
```

```
        imgBtnReport.setEnabled(true);
        edtReason.setText(" ");

     // Create an ArrayAdapter using the string array and a default spinner layout

     ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
     R.array.reportReasons, android.R.layout.simple_spinner_item);
     // Specify the layout to use when the list of choices appears
     adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
     // Apply the adapter to the spinner
     spinner.setAdapter(adapter);
     spinner.setOnItemSelectedListener(this);

    }

    /**
     * @param v το View για το button
     * @throws ExecutionException
     * @throws InterruptedException
     *
     * @info
     * Κάνει την αναφορά του οδηγού και την καταγράφει στην βάση.
     *
     * @details
     * Αρχικά κάνει έλεγχο αν υπάρχει σύνδεση με το δίκτυο και αν μπορεί να συνδεθεί
     * στον server της εφαρμογής.
     * Αν υπάρχει βλέπουμε τι έδωσε ο χρήστης και αν συμπλήρωσε όλα τα απαραίτητα πεδία,
     * προχωράμε στην αναφορά
     *
     */
    public void onClkReport(View v) throws JSONException, InterruptedException,
ExecutionException
    {

        if ( new ConnectionDetector(this).execute().get().get("connectionState") ) {

            if(getRptReasons().equals("Άλλο,αναφέρετε παρακάτω") &&
edtReason.getText().toString().equals(null)){

                new showAlertMessage(this,"Προσοχή","Πρέπει να δώσετε τον λόγο της
αναφοράς!!!") ;

              }
            else{

                //Ως όρισμα είναι το σύνθετο string χωρισμένο σε δύο μέρη με "+" ανάμεσα
                totalReasons = getRptReasons()+" "+edtReason.getText().toString();
               new AlertDialog.Builder( this )
               .setTitle( "Επιβεβαίωση αναφοράς" )
               .setMessage( "Έχετε αναφέρει το εξής : ' "+totalReasons +"'  είστε βέβαιος
ότι θέλετε να στείλετε?")
               .setOnCancelListener( new OnCancelListener() {

                   @Override
                   public void onCancel(DialogInterface dialog) {

                   }
               })
               .setPositiveButton( "Ναι", new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int which) {
                       new CreateReport( totalReasons ).execute() ;

                       }
               })
               .setNegativeButton( "Οχι", new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int which) {

                   }
               } )
           .show();

            imgBtnReport.setEnabled(false);
        }
    }


     }
```

```
//=============================================================================================
================
    /**
     * @param v View για το button
     *
     * @info
     * (build-in)Συνάρτηση για τον τερματισμό της τρέχουσας activity με το πάτημα του back
(hardware button)
     */
    public void onClkBack(View v)
      {
        finish();
      }


//=============================================================================================
================

    /**
     * @info
     * Επιστρέφει όποιο στοιχείο έχει επιλέξει από το spinner ο χρήστης
     */

    public void onItemSelected(AdapterView<?> parent, View view,int pos, long id) {

        setRptReasons(parent.getItemAtPosition(pos).toString());

    }

    public void onNothingSelected(AdapterView<?> arg0) {
        //  Auto-generated method stub
    }

    public String getRptReasons() {
        return RptReasons;
    }

    public void setRptReasons(String rptReasons) {
        RptReasons = rptReasons;
    }

//=================================================================

    /**
     *
     * @info
     * Με την κλάση αυτή γίνετε η εισαγωγή report στη βάση δεδομένων
     */
protected class CreateReport extends AsyncTask<String, String, String> {

        private ArrayList<NameValuePair> parameters =  new ArrayList<NameValuePair>() ;
        private String Repreasons;


    public CreateReport(String repReason) {
        //  Auto-generated constructor stub

        this.Repreasons = repReason;

    }




/**
     * @info
     * Πριν ξεκινήσει η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη
     *
     * @details
     * Πριν ξεκινήσει η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη
     * */
            @Override
            protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(ReportActivity.this);
                pDialog.setMessage("Δημιουργία αναφοράς..");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(true);
                pDialog.show();
            }
```

```
            /**
             * @info
             * Γίνετε η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη
             *
             * @details
             * Γίνετε η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη.
             * Στέλνουμε στοιχεία όπως : λόγοι report , από ποιον έγινε και σε ποιν
αναφέρεται και όλα
             * αυτά για την συγκεκριμένη παραγγελία
             * */
            protected String doInBackground(String... args) {

            try {

                SharedPreferences orderPrefs = getSharedPreferences("fromPush",
MODE_PRIVATE);
                SharedPreferences acceptedCustomerPref =
getSharedPreferences("acceptedCustomer", MODE_PRIVATE);
                SharedPreferences driverPref = getSharedPreferences("fromDriver",
MODE_PRIVATE);

                Log.d("order-from-to",String.valueOf(orderPrefs.getString("selectedOrderId",
""))+"-"+String.valueOf(driverPref.getString("driverDeviceId", "" )) +"-"+
String.valueOf(acceptedCustomerPref.getString("acceptedCustomerDevid","")) );

                parameters.add(new BasicNameValuePair("getReasons", this.Repreasons ));
                parameters.add(new
BasicNameValuePair("orderid",orderPrefs.getString("selectedOrderId", "")  )  );
                parameters.add(new
BasicNameValuePair("reportfrom",driverPref.getString("driverDeviceId", "" )) );
                parameters.add(new
BasicNameValuePair("reportTo",acceptedCustomerPref.getString("acceptedCustomerDevid","") )
);
                parameters.add(new BasicNameValuePair("userType","driver" ) );
                // Building Parameters


                    // getting JSON Object
                    JSONObject myjobg = null;
                    httpJSONParser json = new httpJSONParser();
                    myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeReport.php",
"POST",parameters);

                // check for success tag
                    if (Integer.parseInt(myjobg.get("success").toString()) == 1) {
                        // successfully reported

                    } else {
                        // failed to make report

                    }
                }
                catch (JSONException e) {
                    e.printStackTrace();

                }
                catch(Exception e)
                {
                    e.printStackTrace();
                }

                return null;
            }

            /**
             * @info
             *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα
             *
             * @details
             * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
             * συνεχίζεται στο κύριο νήμα.
             *
             * **/
            protected void onPostExecute(String file_url) {
                // dismiss the dialog once done
                pDialog.dismiss();
            }
```

```
        }
//=============================================================================================
//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
* @info
* Κώδικας σχετικά με το hardware button menu
*
*
* @details
* Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
* με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
* Έχει τις δυνατότητες bookmark, share & profile.
*
* */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
  switch (item.getItemId())
  {

  case R.id.menu_contact:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;
  case R.id.menu_report:

      Toast.makeText(this, "Είστε ήδη εδώ!!", Toast.LENGTH_SHORT).show();
    return true;

  case R.id.menu_bookmark:
      saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
      return true;
  case R.id.menu_share:
    share();
      return true;
  case R.id.menu_profile:
      Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
      return true;
  case R.id.menu_aboutTheApp:
    aboutTheApp();
      return true;
  default:
      return super.onOptionsItemSelected(item);
  }

}



/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

      Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
      startActivity(intent);
}

/**
* @param c το περιεχόμενο της δραστηριότητας.
* @param title ο τίτλος του σελιδοδείκτη.
* @param url το site στο οποίο θα κάνει σελιδοδείκτη.
*
* @info
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*
* @details
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*/
public static final void saveBookmark(Context c, String title, String url) {
  Intent i = new Intent(Intent.ACTION_INSERT,
                android.provider.Browser.BOOKMARKS_URI);
  i.putExtra("title", title);
  i.putExtra("url", url);
  c.startActivity(i);
```

```
}

/**
* @info
* Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
* με την εφαρμογή
*
* @details
* Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
* με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
* για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
*/
public void share () {
    Intent share = new Intent(Intent.ACTION_SEND);
    String shareBody = "Taxi Anytime";
    share.setType("text/plain");
    share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
    share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
    startActivity(Intent.createChooser(share, "Send via"));

}

/**
* @info
* Πληροφορίες σχετικά με την εφαρμογή.
*
* @details
* Πληροφορίες σχετικά με την εφαρμογή.
*/
public void aboutTheApp() {

    setContentView(R.layout.menu_about);
        TextView aboutTextView = (TextView)findViewById(R.id.about);
        aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                "Έκδοση 1.0<br> \n" +
                "Copyright 2013<br>  \n" +
                "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                "Χατζημιχαήλ Γεώργιος <br> \n" +
                "Τσεγγελίδης Φίλανδρος <br>"));

        aboutTextView.setLinkTextColor(Color.WHITE);
        Linkify.addLinks(aboutTextView, Linkify.ALL);

}


/**
* @info
* Αλλαγή προφίλ
*
* @details
* Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
*/
public void changeProfile(){

    Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
    startActivity(intent);
}


}//end class
```

ΑΡΧΕΙΟ showAlertMessage.java

```
package app.taxiAnytimeDriver.Common;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import app.taxiAnytimeDriver.R;


/**
 * @info
 * Κλάση για την διαχείριση των προηδοποιητικών μυνημάτων
 * Μπορεί να χρησιμοποιηθεί από όλες τις κλάσεις
 *
 */
public class showAlertMessage {

private Activity actContext ;
private String title ;
private String message ;

    public showAlertMessage() {
        actContext = null;
        title = null;
        message = null;

    }
    public showAlertMessage(Activity act,String tlt,String msg) {

        actContext = act;
        title = tlt;
        message = msg;

        showConnectionAlertMessage(actContext,title,message);
    }

        /**
         *
         *
         * @param message - Το μήνυμα
         *
         * @info
         * Συνάρτηση εμφάνισης μυνήματος σφάλματος
         *
         * @details
         *Συνάρτηση εμφάνισης μυνήματος σφάλματος
         *
         *
         *
         * */

         public void showConnectionAlertMessage (Activity actAcontext , String title,String
message) {

            AlertDialog alertDialog = new AlertDialog.Builder(
                    actAcontext ).create();

               alertDialog.setTitle(title);
               alertDialog.setMessage(message);
               alertDialog.setIcon(R.drawable.tick);

               // Setting OK Button
               alertDialog.setButton("OK", new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int which) {


            }
            });

               // Showing Alert Message
               alertDialog.show();
        }

}
```

ПАКЕТО
app.taxiAnytimeDriver.Driver

APXEIO DriverActivity.java

```
package app.taxiAnytimeDriver.Driver;


import java.util.ArrayList;
import java.util.concurrent.ExecutionException;


import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONObject;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnCancelListener;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.CheckBox;
import android.widget.TextView;
import android.widget.Toast;

import app.taxiAnytimeDriver.R;

import app.taxiAnytimeDriver.Common.ConnectionDetector;
import app.taxiAnytimeDriver.Common.EditProfileActivity;
import app.taxiAnytimeDriver.Common.ReportActivity;
import app.taxiAnytimeDriver.Common.globalVariables;
import app.taxiAnytimeDriver.Common.httpJSONParser;
import app.taxiAnytimeDriver.Common.showAlertMessage;
import app.taxiAnytimeDriver.pushService.PushService;
import app.taxiAnytimeDriver.userTypesFactory.Users;
import app.taxiAnytimeDriver.userTypesFactory.UsersFactory;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;



/**
 * @info
 *Η κύρια δραστηριότητα του οδηγού.
 *Απεικονίζει πάνω στο χάρτη την τοποθεσία του.
 *Ζωγραφίζει τους διαθέσιμους πελάτης στον χάρτη κλπ.
 *Χρησιμοποιεί το gps και ξεκινάει η υπηρεσία(Push Service) και εγγράφεται σε αυτήν η
 * συσκευή μας,ώστε να μπορούμε να στέλνουμε αλλά και να λαμβάνουμε ειδοποιήσεις .
 */

public class DriverActivity extends MapActivity
{


    private Users driver;
```

```
    public static MapView mv;
    private MapController mc;

    private LocationManager locationManager;
    private  LocationListener loclistener;
    private ProgressDialog pDialog;
    public static Location myLocation;
    public static CheckBox checkBoxAccept;
    public static CheckBox ckeckReachCustomer;
    public static  Context myContext;
    public static Activity myActivity;
    public static DriverPositionOverlay driverOverlay;
    public static Bundle extraParameters;

    private final int SEND_NOTIFICATION = 1;
    private final int SENDED_NOTIFICATION_ACK = 2;
    private final int NEW_CUSTOMER = 1;
    private final int ACCEPTED_CUSTOMER = 2;
    private final int DRAW_CUSTOMER = 1;
    private final int DRAW_DRIVER = -1;
    private final int DRAW_ACCEPTED_CUSTOMER = 2;
    private final int DRAW_AFTER_MAP_CLEAR = 3;

    private final String BOOKMARK_TITLE = "Taxi Anytime";
    private final String BOOKMARK_URL ="www.taxianytime.hostzi.com";


  @Override
  protected boolean isRouteDisplayed()
  {
    return true;
  }

  /**
  * @info
  * Αρχικοποίηση activity
  *
  * @details
  * Αντιστοιχούμε τις όψεις του xml αρχείου με αντικείμενα , ώστε να μπορούμε να τα
διαχειριστούμε.
  * Εισάγουμε το id της συσκευής σε αρχείο SharedPreferences για να μπορέσουμε να το
χρησιμοποιήσουμε αργότερα
  * στις επόμενες δραστηριότητες/service.
  * Ξεκινάει το service μετά από όλα αυτά λαμβάνουμε την τοποθεσία gps με την βοήθεια της
κατάλληλης
  * κλασης (ConnectionManager) και ζωγραφίζουμε το εικονίδιο του χρήστη.
  * Επίσης ζωγραφίζουμε και τον χάρτη με σημειωμένη την τοποθεσία του οδηγού
  *
  */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);



    mv=(MapView) findViewById(R.id.myMapView);
    checkBoxAccept = (CheckBox) findViewById(R.id.checkBoxAccept);
    checkBoxAccept.setChecked(false);
    ckeckReachCustomer = (CheckBox)findViewById(R.id.checkBoxReachCustomer);
    ckeckReachCustomer.setEnabled(false);

    myContext = getApplicationContext(); // για το shared preferences κάτω
    myActivity = DriverActivity.this;   // για να μπορεί να χρησιμοποιηθεί στα static
    //-----------------------
    //initialize driver
    driver = UsersFactory.createUser("driver");

    Editor editor = getSharedPreferences(PushService.TAG, MODE_PRIVATE).edit();
    editor.putString(PushService.PREF_DEVICE_ID, driver.getDeviceID());
    editor.commit();

     //Για να έχουμε πρόσβαση στο id της συσκευής σε όλες τις activities
    SharedPreferences pref = getSharedPreferences("fromDriver", 0);
    SharedPreferences.Editor edit = pref.edit();
    edit.putString("driverDeviceId","/"+driver.getDeviceID());
    edit.commit();
```

```
    mv.setBuiltInZoomControls(true);
    mv.setStreetView(true);
    mv.setSatellite(false);
    mc= mv.getController();
    mc.setZoom(16);


try {
    locationManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);

    Criteria criteria = new Criteria();
    criteria.setAccuracy(Criteria.NO_REQUIREMENT);  //NO_REQUIREMENT
    criteria.setPowerRequirement(Criteria.NO_REQUIREMENT);         //NO_REQUIREMENT
    String provider = locationManager.getBestProvider(criteria, true);
    myLocation = locationManager.getLastKnownLocation(provider);
    loclistener = new mylocationlistener(myLocation);

    locationManager.requestLocationUpdates(provider,0,0,loclistener);

    driver.setLatitude(myLocation.getLatitude());
    driver.setLongitude(myLocation.getLongitude());

    // o odhgos dhmiourgeitai katw ..sto  else


  }
  catch(Exception e){
      new showAlertMessage(this ,"Ωχ! Κάτι πήγε στραβά!!!","Επανεκκινήστε την εφαρμογή και
δοκιμάστε αργότερα");
      e.printStackTrace();

  }

    try {
        extraParameters = getIntent().getExtras();
        inComingNotifications( extraParameters );
    }
    catch (Exception e) {
        e.printStackTrace();
    }


  }//oncreate




  /**
      * @info
      * Αρχικοποίηση menu buttons
      *
      * @details
      * Αρχικοποίηση menu buttons
      */
  @Override
  public boolean onCreateOptionsMenu(Menu menu)
  {
      MenuInflater menuInflater = getMenuInflater();
      menuInflater.inflate(R.layout.menu_items, menu);
      return true;
  }



//=============================================================================================

  /**
 * @info
 * Καθαρισμός των αντικειμένων πάνω στον χάρτη.
 */
public static void clearMap()
  {
    if( driverOverlay != null && extraParameters != null && !mv.getOverlays().isEmpty() ){
        extraParameters = null;
        OverlayInstance.OverlayInstance = null;
        driverOverlay.items.clear();
        driverOverlay.driver = null;
        driverOverlay = null;
```

```
        mv.getOverlays().clear();
        mv.postInvalidate();

        mv = null;

    }

  }

//============================================================================================
/**
* @info
* build-in συνάρτηση για την διαχείρηση τερματισμού δραστηριότητας
*
* @details
* Όταν τερματιστεί η δραστηριότητα του οδηγού κλείνουμε το push service
* και καθαρίζουμε τον χάρτη
*/
@Override
protected void onDestroy() {
    //Auto-generated method stub
    //Καθαρισμός χάρτη και κλείσιμο service

    PushService.actionStop(DriverActivity.this);
    clearMap();
    super.onDestroy();
}

//============================================================================================
  /**
   * @param view Η όψη στην οποία βρίσκεται το κουμπί
 * @throws ExecutionException
 * @throws InterruptedException
   *
   * @info
   * Διαχείρηση checkbox σχετικά με το άν είναι διαθέσιμος ο οδηγός
   *
   * @details
   * Αν τικάρει ο οδηγός το checkbox τότε μπορεί να δέχεται κλήσεις.
   * Αλλιώς όχι.
   */
 public void onCheckAcceptClicked(View view) throws InterruptedException, ExecutionException
{
        // Is the view now checked?

     if ( new ConnectionDetector(this).execute().get().get("connectionState") ){

         if (((CheckBox) view).isChecked()){
             DriverAcceptOrders("1","/"+this.driver.getDeviceID());

         }
         else{
             DriverAcceptOrders("0","/"+this.driver.getDeviceID());

      }

    }

}

//============================================================================================
 /**
 * @param view Η όψη στην οποία βρίσκεται το κουμπί
 * @throws ExecutionException
 * @throws InterruptedException
 *
 * @info
 * Διαχείρηση checkbox σχετικά με το άν έχει φτάσει ο οδηγός στον πελάτη
 *
 * @details
 * 'Οταν ο οδηγός φτάσει στον πελάτη ενεργοποιεί αυτό το checkbox
 * Αν επιλέξει ναι τότε ενεργοποιείται η εργασία ειδοποίησης (DriverConfirmOrder),
 * αν όχι τότε θέτει το checkbox σε ανενεργό
 *
 */
 public void onReachCustomer(View view) throws InterruptedException, ExecutionException {
        // Is the view now checked?

     if ( new ConnectionDetector(this).execute().get().get("connectionState") ){
```

```
        if (((CheckBox) view).isChecked()){

            AlertDialog.Builder builder =  new AlertDialog.Builder(this );

           builder.setTitle( "Ειδοποίηση άφιξης" )
           .setCancelable(false)
           .setMessage( "Βλέπετε τον πελάτη??" )
           .setOnCancelListener( new OnCancelListener() {

               @Override
               public void onCancel(DialogInterface dialog) {


                   finish();   //to finish Activity on which dialog is displayed
               }
           })
           .setPositiveButton( "NAI", new DialogInterface.OnClickListener() {

               public void onClick(DialogInterface dialog, int which) {

                   //do something

                   //Ανακτούμε την παραγγελία(id) και την περνάμε ως όρισμα & ειδοποιούμε
τον πελάτη
                   SharedPreferences incOrder = getSharedPreferences("fromPush",
MODE_PRIVATE);
                   new DriverConfirmOrder("/"+driver.getDeviceID() ,
incOrder.getString("selectedOrderId","") ).execute();

                   clearMap();
                   Intent i = new Intent(DriverActivity.this,DriverEndRideActivity.class);
                   startActivity(i);

               }
           })
           .setNegativeButton( "OXI", new DialogInterface.OnClickListener() {
               public void onClick(DialogInterface dialog, int which) {

                   ckeckReachCustomer.setChecked(false);


               }
           } );


       AlertDialog alert = builder.create();
       alert.setOwnerActivity(DriverActivity.this);
        alert.show();

      }
  }

}


 //================================================================
 /**
 * @param extras αντικείμενο που περιέχει τις πληροφορίες του πελάτη
 *
 * @info
 * Διαχειρίζεται τις ειδοποιήσεις που δέχετε απο τον πελάτη
 *
 * @details
 * Αν η ειδοποίηση είναι ότι υπάρχει διαθέσιμος πελάτης για παραλλαβή δημιουργεί το
 * αντικείμενο του πελάτη με όλες τις πληροφορίες του και τον ζωγραφίζει στον χάρτη
 * για να δει ο οδηγός άν θέλει να δηλώσει ενδιαφέρος γι αυτή την παραγγελία.
 *
 * Αν η ειδοποίηση είναι ότι ο πελάτης έχει δεχτεί τον συγκεκριμένο οδηγό για την παραγγελία
 * ο οδηγός δέν μπορεί να δεχτεί άλλες ειδοποιήσεις για παραγγελίες (τα checkboxes γίνονται
false)
 *
 *
 *
 */
 private void inComingNotifications(Bundle extras)
 {

     if(extras != null){
```

```
              SharedPreferences preferences = getSharedPreferences("fromPush",
MODE_PRIVATE);
              switch( Integer.valueOf( preferences.getString("type","") ))
              {
                case SEND_NOTIFICATION:
                {


                                DriverAcceptOrders("0","/"+this.driver.getDeviceID());
                                Users customer = UsersFactory.createUser("customer");


                                customer.setDeviceID( preferences.getString("customerid", "") );
                                customer.setLatitude(
Double.valueOf(preferences.getString("customerLat", "") ) );
                                customer.setLongitude(
Double.valueOf(preferences.getString("customerLon", "") ) );


                                //Log.d("accepted_customer",
preferences.getString("customerAcceptedID", "") );
                                SharedPreferences SendPrefs =
this.getApplicationContext().getSharedPreferences(
                                          "acceptedCustomer",Context.MODE_PRIVATE);
                                SharedPreferences.Editor prefEditor = SendPrefs.edit();

prefEditor.putString("acceptedCustomerDevid",customer.getDeviceID());
                                prefEditor.commit();


                                drawLocation("customer",customer,DRAW_CUSTOMER);


                                break;
                  }

                  case SENDED_NOTIFICATION_ACK:
                  {

                          DriverAcceptOrders("0","/"+this.driver.getDeviceID());
                          Users customer = UsersFactory.createUser("customer");


                          customer.setDeviceID( preferences.getString("customerAcceptedID",
"") );
                          customer.setLatitude(
Double.valueOf(preferences.getString("customerAcceptLat", "") ) );
                          customer.setLongitude(
Double.valueOf(preferences.getString("customerAcceptLon", "") ) );



                          drawLocation("customer",customer,DRAW_ACCEPTED_CUSTOMER);

                          ckeckReachCustomer.setEnabled(true);
                          ckeckReachCustomer.setChecked(false);
                          checkBoxAccept.setEnabled(false);

                          break;
                  }


              }
            }
            else
            {

              //push service start
              PushService.actionStart(DriverActivity.this);

              DriverAcceptOrders("0","/"+this.driver.getDeviceID());
              drawLocation("driver",driver,DRAW_DRIVER);

              //Toast toast = Toast.makeText(getBaseContext(),"Καλώς
Ορίσατε!",Toast.LENGTH_LONG);
              //toast.show();

            }
```

```
 }


 //=========================================================
 /**
* @param clientType(το αντικείμενο του πελάτη ή του οδηγού)
* @param userType(τύπος αντικειμένου "customer" ή "driver" ανάλογα)
* @param notificationType (1:Για νέο πελάτη , 2:Για λήψη επιβεβαίωσης αποδοχής , -1 : κενό)
*
* @info
* Ζωγραφίζει πάνω στο mapView την τοποθεσία του πελάτη βάσει του γ.μήκους/πλάτους
*
* @details
* Ζωγραφίζει πάνω στο mapView την τοποθεσία του πελάτη βάσει του γ.μήκους/πλάτους
*
*/
  protected void drawLocation(String userType,Users clientType,int notificationType)
    {

        Drawable marker = null;
        String messageSnippet = null ;

        GeoPoint geopoint = new
GeoPoint((int)(clientType.getLatitude()*1E6),(int)(clientType.getLongitude()*1E6));

        mc.animateTo(geopoint);

        if(userType == "customer") {

            switch(notificationType)
            {
                case NEW_CUSTOMER:
                {

                    marker = getResources().getDrawable(R.drawable.customer_icon2);
                    messageSnippet = "new_customer";
                    break;
                }
                case ACCEPTED_CUSTOMER:
                {
                    marker = getResources().getDrawable(R.drawable.accepted_customer);
                    messageSnippet = "accepted_customer";
                    break;
                }

            }

        }
        else
        {
                messageSnippet = "new_driver";
                marker = getResources().getDrawable(R.drawable.taxi_icon);

        }


        int markerWidth = marker.getIntrinsicWidth();
        int markerHeight = marker.getIntrinsicHeight();
        marker.setBounds(-markerWidth/2, -markerHeight,markerWidth/2, 0);



       driverOverlay = OverlayInstance.getOverlayInstance(marker, DriverActivity.this);
       driverOverlay.addOverlayItem(geopoint,userType,messageSnippet, marker,clientType);


        mv.getOverlays().add( driverOverlay);
        mv.invalidate();



    }
//=============================================================================

static void DriverAcceptOrders(String canAccept, String driverDevID)  {

        new DriverAcception(canAccept,driverDevID).execute();
```

```
}

static void makeOrder(Users customers,Users driver)  {

        new readyOrder(customers,driver).execute();


}
//================================================================================

/**
* @info
* Κλάση για να αποδεχθεί ο οδηγός την παραγγελία (ενεμερώνει τον πίνακα order orderState =2)
*
* @details
* Η κλάση "τρέχει " στο παρασκήνιο και επικοινωνεί με τη βάση δεδομένων για την καταχώρηση
* της κατάστασης παραγγελίας
*/
protected class DriverConfirmOrder extends AsyncTask<String, String, String> {

    private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
    private ArrayList<NameValuePair> parameters2 = new ArrayList<NameValuePair>(1);
    private String deviceid;
    private String orderid;

    public DriverConfirmOrder(final String driverDeviceid,final String orderid) {
        //  Auto-generated constructor stub
        this.deviceid = driverDeviceid;
        this.orderid = orderid;
    }

    /**
     * @info
     * Πριν ξεκινήσει η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη.
     *
     * @details
     * Πριν ξεκινήσει η διαδικασία εισαγωγής εμφανίζεται μήνυμα στην οθόνη χρήστη.
     * */
    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(DriverActivity.this);
        pDialog.setMessage("Loading..");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(true);
        pDialog.show();
    }

    /**
     * @info
     * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request
     * για να καταχωρηθεί η κατάσταση παραγγείας
     *
     * @details
     * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request
     * για να καταχωρηθεί η κατάσταση παραγγείας
     *
     * */
    @Override
    protected String doInBackground(String... args) {


        parameters.add(new BasicNameValuePair("driverid",this.deviceid));
        parameters.add(new BasicNameValuePair("orderid",this.orderid));
        parameters.add(new BasicNameValuePair("type","driver"));

        parameters2.add(new BasicNameValuePair("orderid",this.orderid));
        try {
            // getting JSON Object
            JSONObject myjobg = null;
            httpJSONParser json = new httpJSONParser();
            myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/orderConfirmatio
n.php","POST",parameters);


            //Ενημέρωση πελάτη για την άφιξη του οδηγού
            httpJSONParser json2 = new httpJSONParser();

json2.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/mqttClient/notifyCustom
erForArrival.php","POST",parameters2);
```

```
        // check for success tag

            if (Integer.parseInt(myjobg.get("success").toString()) == 1) {
                // maybe something here

            } else {}

        }
        catch (Exception e) {
            e.printStackTrace();

        }
        return null;
    }
    /**
     * @info
     *  Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
     * συνεχίζεται στο κύριο νήμα
     *
     * @details
     * Αφού ολοκληρωθεί η διαδικασία, αποδεσμεύουμε το dialog και η εκτέλεση
     * συνεχίζεται στο κύριο νήμα
     *
     * **/
    @Override
    protected void onPostExecute(String file_url) {
        // dismiss the dialog once done
        pDialog.dismiss();
        pDialog = null;

    }


}



//============================================================================================

/**
* @info
* Κλάση για να αλλάζει τη διαθεσιμότητα του οδηγού
*
* @details
* Η κλάση "τρέχει " στο παρασκήνιο και επικοινωνεί με τη βάση δεδομένων για την καταχώρηση
* της κατάστασης διαθεσιμότητας του οδηγού
*/
static class DriverAcception extends AsyncTask<String, String, String> {

    private ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();
    private String accept;
    private String driverid;

    public DriverAcception(String accept0,String driverid0) {
        //  Auto-generated constructor stub
        this.accept = accept0;
        this.driverid = driverid0;
    }


    /**
     * @info
     * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request
     * για να καταχωρηθεί η κατάσταση διαθεσιμότητας του οδηγού
     *
     * @details
     * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request
     * για να καταχωρηθεί η κατάσταση διαθεσιμότητας του οδηγού
     *
     * */
    protected String doInBackground(String... args) {


        parameters.add(new BasicNameValuePair("available",String.valueOf(this.accept)));
        parameters.add(new BasicNameValuePair("driverid",this.driverid));

        try {
            // getting JSON Object
```

```
            JSONObject myjobg = null;
            httpJSONParser json = new httpJSONParser();
            myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/setdriverAvailab
ility.php","POST",parameters);


        // check for success tag

            if (Integer.parseInt(myjobg.get("success").toString()) == 1) {
                // maybe something here

            } else {}

        }
        catch (Exception e) {
            e.printStackTrace();

        }
        return null;
    }

}

//============================================================================================
/**
* @info
* Κλάση για να δημιουργείται η παραγγελία στη βάση
*
* @details
* Η κλάση "τρέχει " στο παρασκήνιο και επικοινωνεί με τη βάση δεδομένων για την καταχώρηση
* της νέας παραγγελίας
*/
static class readyOrder extends AsyncTask<String, String, String> {

    ArrayList<NameValuePair> parameters = new ArrayList<NameValuePair>();

    httpJSONParser json = new httpJSONParser();
    JSONObject myjobg =null ;
    Users customers,driver;


    public readyOrder(Users Customers,Users Driver) {
        //  Auto-generated constructor stub
        this.customers = Customers;
        this.driver = Driver;

    }

    /**
     * @info
     * Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request
     * για να καταχωρηθεί η νέα παραγγελία
     *
     * @details
     *  Εδώ γίνεται η επικοινωνία με τη βάση κάνοντας http request
     * για να καταχωρηθεί η νέα παραγγελία
     *
     * */
    protected String doInBackground(String... args) {

         float distance =
getCalculatedDistance(customers.getLatitude(),customers.getLongitude(),driver.getLatitude(),
driver.getLongitude());


            try {

                parameters.add( new BasicNameValuePair("customerlat",String.valueOf(
customers.getLatitude() ) ) );
                parameters.add( new BasicNameValuePair("customerlon",String.valueOf(
customers.getLongitude() ) ) );
                parameters.add( new
BasicNameValuePair("driverlat",String.valueOf(driver.getLatitude() ) ) );
                parameters.add( new
BasicNameValuePair("driverlon",String.valueOf(driver.getLongitude() ) ) );
                parameters.add( new
BasicNameValuePair("customerDevice",customers.getDeviceID()));
                parameters.add( new
BasicNameValuePair("driverDevice","/"+driver.getDeviceID()));
```

```
                parameters.add( new BasicNameValuePair("distance",String.valueOf(distance)
));


                myjobg =
json.makeHttpRequest(globalVariables.getInstance().getBASE_PATH()+"/scripts/makeOrder.php","
POST",parameters);


                if(Integer.parseInt(myjobg.get("success").toString()) == 1)
                {

                    //kratame to orderid pou 8a steiloume otan etoimazetai mia paraggelia se
shared preferences

                    SharedPreferences OrderPrefs = myContext.getSharedPreferences(
                              "myOrder", Context.MODE_PRIVATE);
                    SharedPreferences.Editor prefEditor = OrderPrefs.edit();
                    prefEditor.putString("orderid",myjobg.get("orderid").toString());
                    prefEditor.commit();

                }

            }
            catch(Exception e)
            {
                e.printStackTrace();

            }

        return null;
    }

    /**
     * @param lat_a
     * @param lng_a
     * @param lat_b
     * @param lng_b
     *
     * @info
     * Μέθοδος για τον υπολογισμό της απόστασης μεταξύ οδηγού και  επιλεγμένου πελάτη
     *
     * @details
     * Υπολογίζεται βάσει  τα γεωγραφικά μήκη και πλάτη
     * @return
     */
    private float getCalculatedDistance(double lat_a, double lng_a, double lat_b, double
lng_b)
     {
                Location locationA = new Location("point A");
                Location locationB = new Location("point B");


                try
                {
                    locationA.setLatitude(lat_a);
                    locationA.setLongitude(lng_a);

                    locationB.setLatitude(lat_b);
                    locationB.setLongitude(lng_b);


                    return locationA.distanceTo(locationB);
                }
                catch(Exception e)
                {
                    e.printStackTrace();
                    return 0.0f;
                }
    }

}


//================================================================================
/**
*
*
* @info
* Για μελλοντική χρήση, όταν θέλουμε να έχουμε την νέα τοποθεσία αν αλλάξουμε
* θέση
```

```
*
*Είναι αυτοπαραγώμενη κλάση του SDK
*
*/
private class mylocationlistener implements LocationListener {

    public mylocationlistener(Location myloc) {
        //  Auto-generated constructor stub

    }
    @Override
    public void onLocationChanged(Location location) {
        if (location != null) {

        }
    }

    @Override
    public void onProviderDisabled(String provider) {
        // Auto-generated method stub

    }

    @Override
    public void onProviderEnabled(String provider) {
        //  Auto-generated method stub

    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
        // Auto-generated method stub

    }
}



//======================================================================
//menus

/**
 * @param item η επιλογή που έκανε ο χρήστης απο το μενού
 *
* @info
* Κώδικας σχετικά με το hardware button menu
*
*
* @details
* Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
* με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
* Έχει τις δυνατότητες bookmark, share & profile.
*
* */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
// Single menu item is selected do something
// Ex: launching new activity/screen or show alert message
switch (item.getItemId())
{

case R.id.menu_contact:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;
case R.id.menu_report:

    Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
    return true;

case R.id.menu_bookmark:
    saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
    return true;
case R.id.menu_share:
    share();
    return true;
case R.id.menu_profile:
    changeProfile();
    return true;
case R.id.menu_aboutTheApp:
```

```
    aboutTheApp();
    return true;
default:
    return super.onOptionsItemSelected(item);
}

}



/**
* @info
* Μεταβαίνουμε στην δραστηριότητα της αναφοράς
*/
public void report(){

  Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
  startActivity(intent);
}

/**
* @param c το περιεχόμενο της δραστηριότητας.
* @param title ο τίτλος του σελιδοδείκτη.
* @param url το site στο οποίο θα κάνει σελιδοδείκτη.
*
* @info
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*
* @details
* Κάνει σελιδοδείκτη το site της εφαρμογής.
*/
public static final void saveBookmark(Context c, String title, String url) {
Intent i = new Intent(Intent.ACTION_INSERT,
              android.provider.Browser.BOOKMARKS_URI);
i.putExtra("title", title);
i.putExtra("url", url);
c.startActivity(i);
}

/**
* @info
* Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
* με την εφαρμογή
*
* @details
* Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
* με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
* για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms κλπ)
*/
public void share () {
Intent share = new Intent(Intent.ACTION_SEND);
String shareBody = "Taxi Anytime";
share.setType("text/plain");
share.putExtra(android.content.Intent.EXTRA_SUBJECT, "");
share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
startActivity(Intent.createChooser(share, "Send via"));


}

/**
* @info
* Πληροφορίες σχετικά με την εφαρμογή.
*
* @details
* Πληροφορίες σχετικά με την εφαρμογή.
*/
public void aboutTheApp() {

setContentView(R.layout.menu_about);
    TextView aboutTextView = (TextView)findViewById(R.id.about);
    aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
            "Έκδοση 1.0<br> \n" +
            "Copyright 2013<br>  \n" +
            "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
            "Η εφαρμογή σχεδιάστηκε απο τους \n" +
            "Χατζημιχαήλ Γεώργιος <br> \n" +
            "Τσεγγελίδης Φίλανδρος <br>"));

    aboutTextView.setLinkTextColor(Color.WHITE);
    Linkify.addLinks(aboutTextView, Linkify.ALL);
```

```
}


/**
* @info
* Αλλαγή προφίλ
*
* @details
* Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
*/
public void changeProfile(){

Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
startActivity(intent);
}


}//end class
```

ΑΡΧΕΙΟ EndRideActivity.java

```
package app.taxiAnytimeDriver.Driver;

import java.util.concurrent.ExecutionException;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Html;
import android.text.util.Linkify;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import app.taxiAnytimeDriver.R;
import app.taxiAnytimeDriver.Common.Contact;
import app.taxiAnytimeDriver.Common.EditProfileActivity;
import app.taxiAnytimeDriver.Common.ReportActivity;


/**
 * @info
 * Δραστηριότητα για τερματισμό της διαδρομής
 *
 */

public class DriverEndRideActivity extends Activity {

    private static final String BOOKMARK_TITLE = "Taxi Anytime";
    private static final String BOOKMARK_URL ="www.taxianytime.hostzi.com";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.endride_activity);

    }
```

```
  /**
      * @info
      * Αρχικοποίηση menu buttons
      *
      * @details
      * Αρχικοποίηση menu buttons
      */
  @Override
  public boolean onCreateOptionsMenu(Menu menu)
  {
      MenuInflater menuInflater = getMenuInflater();
      menuInflater.inflate(R.layout.menu_items, menu);
      return true;
  }


public void ImgBtnEndRide(View v)
{

    finish();
    Intent i = new Intent(getApplicationContext(),DriverActivity.class);
    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(i);


}


@Override
protected void onDestroy() {
    //  Auto-generated method stub



    super.onDestroy();
}


//=========================================================================
//menus

//menus

    /**
     * @param item η επιλογή που έκανε ο χρήστης απο το μενού
     *
 * @info
 * Κώδικας σχετικά με το hardware button menu
 *
 *
 * @details
 * Στην παρούσα φάση ο χρήστης δεν εχει δικαίωμα να κανει αναφορά ούτε επικοινωνία
 * με κάποιον οδηγό διότι πολυ απλά δεν έχει ξεκινήσει η παραγγελία.
 * Έχει τις δυνατότητες bookmark, share & profile.
 *
 * */
@Override
public boolean onOptionsItemSelected(MenuItem item)
{
    // Single menu item is selected do something
    // Ex: launching new activity/screen or show alert message
    switch (item.getItemId())
    {

    case R.id.menu_contact:

        try {
            call();

        } catch (Exception e) {
            e.printStackTrace();
        }
        return true;
    case R.id.menu_report:

        report();
        return true;

    case R.id.menu_bookmark:
```

```
            saveBookmark(this,BOOKMARK_TITLE,BOOKMARK_URL);
            return true;
        case R.id.menu_share:
            share();
            return true;
        case R.id.menu_profile:
            Toast.makeText(this, "Δεν μπορείτε τώρα!!", Toast.LENGTH_SHORT).show();
            return true;
        case R.id.menu_aboutTheApp:
            aboutTheApp();
            return true;
        default:
            return super.onOptionsItemSelected(item);
        }

    }

    /**
     * @throws ExecutionException
     * @throws InterruptedException
     * @info
     * Μεταβαίνουμε στην δραστηριότητα της κλήσης/sms
     */
    public void call() throws InterruptedException, ExecutionException{
          Contact contact = new Contact(this);
          contact.makeContact();
    }

    /**
    * @info
    * Μεταβαίνουμε στην δραστηριότητα της αναφοράς
    */
    public void report(){

          Intent intent = new Intent(getApplicationContext(),ReportActivity.class);
          startActivity(intent);
    }

    /**
     * @param c το περιεχόμενο της δραστηριότητας.
     * @param title ο τίτλος του σελιδοδείκτη.
     * @param url το site στο οποίο θα κάνει σελιδοδείκτη.
     *
     * @info
     * Κάνει σελιδοδείκτη το site της εφαρμογής.
     *
     * @details
     * Κάνει σελιδοδείκτη το site της εφαρμογής.
     */
    public static final void saveBookmark(Context c, String title, String url) {
        Intent i = new Intent(Intent.ACTION_INSERT,
                      android.provider.Browser.BOOKMARKS_URI);
        i.putExtra("title", title);
        i.putExtra("url", url);
        c.startActivity(i);
    }

    /**
     * @info
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή
     *
     * @details
     * Δίνει την επιλογή στον χρήστη να μοιραστεί με άλλους πληροφορίες σχετικά
     * με την εφαρμογή. Το με ποιό τρόπο θα το μοιραστεί ειναι καθαρά θέμα τι εφαρμογές
     * για διαμοιρασμό εχει στο smartphone εγκατεστημένες (gmail, facebook, tweeter, sms
κλπ)
     */
    public void share () {
        Intent share = new Intent(Intent.ACTION_SEND);
        String shareBody = "Taxi Anytime";
        share.setType("text/plain");
        share.putExtra(android.content.Intent.EXTRA_SUBJECT, "Subject Here");
        share.putExtra(android.content.Intent.EXTRA_TEXT, shareBody);
        startActivity(Intent.createChooser(share, "Send via"));

    }

    /**
     * @info
     * Πληροφορίες σχετικά με την εφαρμογή.
```

```
     *
     * @details
     * Πληροφορίες σχετικά με την εφαρμογή.
     */
    public void aboutTheApp() {

        setContentView(R.layout.menu_about);
            TextView aboutTextView = (TextView)findViewById(R.id.about);
            aboutTextView.setText(Html.fromHtml("<h3>Taxi anytime</h3> \n" +
                    "Έκδοση 1.0<br> \n" +
                    "Copyright 2013<br>  \n" +
                    "<b>www.taxianytime.hostzi.com</b><br><br> \n \n" +
                    "Η εφαρμογή σχεδιάστηκε απο τους \n" +
                    "Χατζημιχαήλ Γεώργιος <br> \n" +
                    "Τσεγγελίδης Φίλανδρος <br>"));

            aboutTextView.setLinkTextColor(Color.WHITE);
            Linkify.addLinks(aboutTextView, Linkify.ALL);

    }


    /**
     * @info
     * Αλλαγή προφίλ
     *
     * @details
     * Μεταβαίνουμε στην αντίστοιχη δραστηριότητα για την αλλαγή προφίλ
     */
    public void changeProfile(){

        Intent intent = new Intent(getApplicationContext(), EditProfileActivity.class);
        startActivity(intent);
    }
}
```

ΑΡΧΕΙΟ DriverPositionOverlay.java

```
package app.taxiAnytimeDriver.Driver;


import java.util.ArrayList;
import java.util.List;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Rect;
import android.graphics.drawable.Drawable;
import android.location.Location;
import android.util.Log;
import android.widget.Toast;
import app.taxiAnytimeDriver.userTypesFactory.Users;
import app.taxiAnytimeDriver.userTypesFactory.UsersFactory;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.ItemizedOverlay;
import com.google.android.maps.OverlayItem;


/**
 * @info
 * Βοηθητική κλάση για την εισαγωγή στοιχείων στο χάρτη.
 * Χρησιμοποιείται σε συνεργασία με την DriverActivity και μας επιτρέπει να ζωγραφίζουμε
αλλά και
 * να διαχειρίζομαστε πολλαπλά items πάνω στο χάρτη.
 * Τα items αυτά αποθηκεύονται σε arraylist
 *
 *
 */
class  DriverPositionOverlay extends ItemizedOverlay<OverlayItem>
{

    public ArrayList<OverlayItem> items ;
    protected Users driver;
    protected Users customer;
```

```
  private  Activity mContext;


  /**
   * @param defaultMarker Το εικονίδιο που θα ζωγραφιστεί
   * @param context το περιεχόμενο της δραστηριότητας που καλούμε
   *
   * @info
   * Αρχικοποιεί τις παραμέτρους
   *
   */

    public DriverPositionOverlay(Drawable defaultMarker,Activity mycontext)
      {
        super(boundCenterBottom(defaultMarker));

          items = new ArrayList<OverlayItem>();
          driver = UsersFactory.createUser("driver");
          customer = UsersFactory.createUser("customer");
          mContext = mycontext;
          populate();

      }

    public void setDriverPositionOverlay( Drawable defaultMarker,Activity mycontext )
    {
        boundCenterBottom(defaultMarker);
        mContext = mycontext;
        populate();
    }



  public Users getCustomer(){

          return ( this.customer );

  }


  public void removeCustomer(){

          this.customer = null;

  }



  public void updateWithNewLocation(Location location)
  {

  }

  public void removeItem(OverlayItem item) {


      items.remove(item);


      DriverActivity.mv.postInvalidate();

}

    @Override
    protected OverlayItem createItem(int i)
    {

        return items.get(i);
    }


  public void addOverlayItem(OverlayItem overlayItem) {
   items.add(overlayItem);

   populate();
```

```
    }



    public void addOverlayItem(GeoPoint point, String title,String info, Drawable
altMarker,Users usr) {


      try {

          if(title == "customer"){
            customer = usr;

          }
          else //if(title == "driver")
          {
            driver = usr;

          }

          OverlayItem overlayItem = new OverlayItem(point, title, info);
          overlayItem.setMarker( boundCenterBottom(altMarker) );
          addOverlayItem(overlayItem);

      } catch (Exception e) {
          // : handle exception
          e.printStackTrace();
      }

    }



    @Override
    public int size()
    {
        //  Auto-generated method stub
        return items.size();
    }



    @Override
    protected boolean onTap(int index)
    {


        /*
         * Toast toast =
Toast.makeText(mContext,items.get(index).getSnippet(),Toast.LENGTH_SHORT);
        toast.show();
        Log.d("index-size",String.valueOf(index)+"-"+String.valueOf(items.size()));
        Log.d("customerSize",String.valueOf(customers.size()));
         Log.d("index-size-max_index",String.valueOf(index)+"-"+String.valueOf(items.size())
+ String.valueOf(maxIndex) );
         */

        try {
            OverlayItem item = items.get(index);


                if(getCustomer()!= null && size()>0 && index>-1 && item.getSnippet().equals(
"new_customer" ) ){

                    DriverActivity.makeOrder(getCustomer() ,driver);
                    removeCustomer();
                    removeItem(item);

                    Toast toast = Toast.makeText(mContext,"Ο πελάτης θα ειδοποιηθεί για το
ενδιαφέρον σας",Toast.LENGTH_LONG);
                    toast.show();
                }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
```

```
        //Log.d("index-size-mvsize",String.valueOf(index)+"-
"+String.valueOf(items.size())+"-"+String.valueOf(DriverActivity.mv.getOverlays().size()));


        return (true);
 }



}
```

ΑΡΧΕΙΟ OverlayInstance.java

```
package app.taxiAnytimeDriver.Driver;

import android.app.Activity;
import android.content.Context;
import android.graphics.drawable.Drawable;



/**
 *
 * @info
 * Τεχνική singleton για να εξασφαλίσουμε ότι μόνο ένα αντικείμενο overlay
 * θα υπάρχει και πάνω σε αυτό θα ζωγραφίζονται τα markers
 *
 */
public class OverlayInstance {

    static DriverPositionOverlay OverlayInstance = null ;



    /**
     * @param mMarker
     * @param mContext
     * @return Το στιγμιότυπο DriverPositionOverlay, new αν δεν υπάρχει, δηλ εκτελείται
πρώτη φορά
     */
    //sync -> This guarantees that changes to the state of the object are visible to all
threads.

    public static synchronized DriverPositionOverlay getOverlayInstance(Drawable
mMarker,Activity mContext){
         if(OverlayInstance == null){
             OverlayInstance = new DriverPositionOverlay(mMarker, mContext);
         }
         else
         {
             OverlayInstance.setDriverPositionOverlay(mMarker,mContext );
         }
         return OverlayInstance;
       }


}
```

ПАКЕТО
app.taxiAnytimeDriver.PushService

ΑΡΧΕΙΟ PushService.java

```
[…]
/**
     * @param text Το εισερχόμενο μύνημα από το mosquitto
     *
     * @info
     * Εμφανίζει το εισερχόμενο μύνημα ως ειδοποίηση στο χρήστη
     *
     * @details
     * Εμφανίζει το εισερχόμενο μύνημα ως ειδοποίηση στο χρήστη.
     * Θέτει τις παραμέτρους ειδοποίησης και χρησιμοποιεί το μηχανισμό intent
     * για να ξεκινήσει η ανάλογη activity
     */
    // Display the topbar notification
private void showNotification(String text) throws JSONException, InterruptedException,
ExecutionException {

        //type : info : customerLat : customerLon : customerid
                /*items[0] -> type
                 *items[1] -> message
                 *items[2] -> customerLat
                 *items[3] -> customerLon
                 *items[4] -> customerid
                 *
                 */

    String[] items = text.split("\\:")   ;
    int type = Integer.parseInt(items[0]);

    switch(type)
    {

        case NEW_CUSTOMER:
        {

            //Stelnoume to orderid sthn activity mas
            SharedPreferences SendPrefs = this.getApplicationContext().getSharedPreferences(
                    "fromPush",Context.MODE_PRIVATE);
            SharedPreferences.Editor prefEditor = SendPrefs.edit();
            prefEditor.putString("customerLat",items[2]);
            prefEditor.putString("customerLon",items[3]);
            prefEditor.putString("customerid",items[4]);
            prefEditor.putString("type","1");
            prefEditor.commit();

            double distanceLimit;
            try
            {

                Location cloc = new Location("");
                cloc.setLatitude( Double.valueOf(items[2]) );
                cloc.setLongitude(  Double.valueOf(items[3]) );

                distanceLimit = DriverActivity.myLocation.distanceTo(cloc);

            }
            catch(Exception e)
            {
                distanceLimit = DISTANCE_LIMIT*2.0f; //σε περίπτωση που υπάρξει σφάλμα
ορίζουμε μια απόσταση αυθαίρετα διπλάσια από την μέγιστη ,ώστε να είμαστε σίγουροι ότι δεν
θα σταλεί ειδοποίηση
                e.printStackTrace();
            }

        //  Log.d("distance???",String.valueOf( distanceLimit) );

            if(distanceLimit <= DISTANCE_LIMIT && distanceLimit!= 0.0f ) // Αν η απόσταση
είναι μικρότερη από κάποια τιμή που ορίζουμε στο conf.java τότε να στείλει ειδοποίηση
            {


                Notification n = new Notification();
                n.flags |= Notification.FLAG_SHOW_LIGHTS;  // ΠΧ    kati |= 2 είναι ίδιο με
kati = kati | 2
                n.flags |= Notification.FLAG_AUTO_CANCEL;
                n.defaults = Notification.DEFAULT_ALL;
                n.icon = R.drawable.star_big_on;
                n.when = System.currentTimeMillis();
```

```
                // Simply open the parent activity
                Intent intent = new Intent(PushService.this, DriverActivity.class);
                intent.setAction(Intent.ACTION_VIEW);
               intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
               intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
                Bundle b = new Bundle();
                b.putString("customer", items[1]);
                intent.putExtras(b);

                PendingIntent pi = PendingIntent.getActivity(this, 0,
                     intent, 0);

                // Change the name of the notification here
                n.setLatestEventInfo(this, NOTIF_TITLE,items[1], pi);

                mNotifMan.notify(NOTIF_CONNECTED, n);


                myContext.startActivity(intent);

            }

            break;
        }

        case ACCEPTED_CUSTOMER:
        {

            //Stelnoume to orderid sthn activity mas
            SharedPreferences SendPrefs = this.getApplicationContext().getSharedPreferences(
                      "fromPush",Context.MODE_PRIVATE);
            SharedPreferences.Editor prefEditor = SendPrefs.edit();
            prefEditor.putString("customerAcceptLat",items[2]);
            prefEditor.putString("customerAcceptLon",items[3]);
            prefEditor.putString("customerAcceptedID",items[4]);
            prefEditor.putString("selectedOrderId",items[5]);
            prefEditor.putString("type","2");
            prefEditor.commit();

            Notification n = new Notification();
            n.flags |= Notification.FLAG_SHOW_LIGHTS;
            n.flags |= Notification.FLAG_AUTO_CANCEL;
            n.defaults = Notification.DEFAULT_ALL;
            n.icon = R.drawable.presence_online;
            n.when = System.currentTimeMillis();

            // Simply open the parent activity
            Intent intent = new Intent(PushService.this, DriverActivity.class);
            intent.setAction(Intent.ACTION_VIEW);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT);
            Bundle b = new Bundle();
            b.putString("customerAccept", items[1]);
            intent.putExtras(b);

            PendingIntent pi = PendingIntent.getActivity(this, 0,
                     intent, 0);
            // Change the name of the notification here
            n.setLatestEventInfo(this, NOTIF_TITLE,items[1], pi);

            mNotifMan.notify(NOTIF_CONNECTED, n);

            myContext.startActivity(intent);

            break;
        }
    }
}
```

ПАКЕТО app.taxiAnytimeDriver.usersTypeFactory

ΑΡΧΕΙΟ Customer.java

```
package app.taxiAnytimeCustomer.userTypesFactory;

import android.os.Build;


/**
 *@info
 *Πληροφορίες σχετικά με τον πελάτη
 *
 */

public class Customer extends Users {


    private double Cstlat;
    private double Cstlng;
    private String customerDeviceID;



    public Customer() {
        // Auto-generated constructor stub

        Cstlat = 0.0f;
        Cstlng = 0.0f;
        customerDeviceID = generateUniqueID();

    }


public void setLatitude(double cstlat) {
    Cstlat = cstlat;
}
public void setLongitude(double cstlng) {
    Cstlng = cstlng;
}


    public void setDeviceID(String devid) {
        customerDeviceID = devid;

}
    public double getLatitude() {
        return Cstlat;
    }
    public double getLongitude() {
        return Cstlng;
    }

    public String getDeviceID() {

        return customerDeviceID;

    }

    /**
     * @info
     * Συνάρτηση για την εξαγωγή μοναδικού id συσκευής
     *
     * @details
     * Παίρνουμε πληροφορίες από το υλικό της συσκευής(cpu brand,board κτλ) ,
     * τα συνθέτουμε και παράγουμε το τελικό id
     *
     *
     * @return Το μοναδικό id της συσκευής
     */
    private String generateUniqueID()
    {
        String id;
        try
        {
            id = "35" +
                    Build.BOARD.length()%10+ Build.BRAND.length()%10 +
                    Build.CPU_ABI.length()%10 + Build.DEVICE.length()%10 +
                    Build.DISPLAY.length()%10 + Build.HOST.length()%10 +
                    Build.ID.length()%10 + Build.MANUFACTURER.length()%10 +
                    Build.MODEL.length()%10 + Build.PRODUCT.length()%10 +
                    Build.TAGS.length()%10 + Build.TYPE.length()%10 +
```

```
                    Build.USER.length()%10 ; //13 digits
        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }

        return id;
    }


}
```

APXEIO Driver.java

```
package app.taxiAnytimeCustomer.userTypesFactory;

import android.os.Build;


/**
 *@info
 *Πληροφορίες σχετικά με τον οδηγό
 *
 */
public class Driver extends Users  {

    private  double Drvlat;
    private double Drvlng;
    private String driverDeviceID;


    public Driver() {
        // TODO Auto-generated constructor stub
        Drvlat = 0.0;
        Drvlng = 0.0;
        driverDeviceID = generateUniqueID();

    }


    //Sets - gets

    public void setDeviceID(String devid) {
        driverDeviceID = devid;

    }

    public void setLatitude(double drvlat) {
        Drvlat = drvlat;
    }

    public void setLongitude(double dvrlng) {
        Drvlng = dvrlng;
    }

    public double getLatitude() {
        return Drvlat;
    }

    public double getLongitude() {
        return Drvlng;
    }


    public String getDeviceID() {
        return driverDeviceID;
    }


    /**
     * @info
     * Συνάρτηση για την εξαγωγή μοναδικού id συσκευής
     *
     * @details
     * Παίρνουμε πληροφορίες από το υλικό της συσκευής(cpu brand,board κτλ) ,
```

```
     * τα συνθέτουμε και παράγουμε το τελικό id
     *
     *
     * @return Το μοναδικό id της συσκευής
     */
    private String generateUniqueID()
    {
        String id;
        try
        {
            id = "35" +
                    Build.BOARD.length()%10+ Build.BRAND.length()%10 +
                    Build.CPU_ABI.length()%10 + Build.DEVICE.length()%10 +
                    Build.DISPLAY.length()%10 + Build.HOST.length()%10 +
                    Build.ID.length()%10 + Build.MANUFACTURER.length()%10 +
                    Build.MODEL.length()%10 + Build.PRODUCT.length()%10 +
                    Build.TAGS.length()%10 + Build.TYPE.length()%10 +
                    Build.USER.length()%10 ; //13 digits

        }
        catch(Exception e)
        {
            e.printStackTrace();
            return null;
        }

        return id;
    }

  }
```

APXEIO Users.java

```
package app.taxiAnytimeCustomer.userTypesFactory;

/**
 * @info
 * Abstract κλάση για τις κοινές μεθόδους οδηγού και πελάτη
 * Χρησιμοποιείται για το factory pattern
 *
 */

public abstract class Users {

     public abstract void setLatitude(double lat);
     public abstract void setLongitude(double lng);
     public abstract void setDeviceID(String devid);
     public abstract double getLatitude();
     public abstract double getLongitude();
     public abstract String getDeviceID();



}
```

APXEIO UsersFactory.java

```
package app.taxiAnytimeCustomer.userTypesFactory;
/**
 * @info
 * Κλάση για την δημιουργία χρηστών με τη μέθοδο factory pattern
 *
 *
 */
public class UsersFactory {

    public static Users createUser(String type)
    {
         if (type.equals("customer")){
                return new Customer();
```

```
            }
        else if (type.equals("driver")){
               return new Driver();
            }
            return null;
    }

}

//Παράδειγμα κλήσης : Users customer = UsersFactory.createUser("customer");
```

ΕΦΑΡΜΟΓΗ SERVER

Υποκατάλογος taxiAnytime_server\scripts

ΑΡΧΕΙΟ checkforOrders.php

```php
<?php
/*
@info
Αρχείο php για έλεγχο διαθέσιμων παραγγελιών  βάσει του πίνακα orders
@details
Με το sql  ερώτημα βλέπουμε το πλήθος αυτών που ενδιαφέρονται, έτσι αν υπάρχουν (count>0)
η απόκριση json θα έχει success tag "1" αλλιώς "0"

*/

header('Content-type=application/json; charset=utf-8');

$jsonResponse = array();
$customerID = null;


require_once __DIR__ .'/db_connect.php';
$database = new DB_CONNECT();



if(isset($_REQUEST["customerdeviceid"]) )
{
    $customerID = $_REQUEST["customerdeviceid"];
}
else
    $customerID ="-";



mysql_query("SET NAMES UTF8");
$sql = "SELECT COUNT(*)
        FROM orders o INNER JOIN customer c ON o.customerFK = c.customerID
        WHERE customerFK = (SELECT customerID FROM customer WHERE customerDeviceID =
'".$customerID."') AND orderState = 1";


$query  = mysql_query($sql) or die(mysql_error());



if (mysql_num_rows($query) > 0) //Αν βρέθηκαν εγγραφές
{

    $rows = mysql_fetch_array($query);

    if($rows[0] > 0) //το row[0] το result από το query
    {
        $jsonResponse["success"] = 1;

    }
    else
        $jsonResponse["success"] = 0;

    // echoing JSON response
   print json_encode($jsonResponse);

}
else
{
    $jsonResponse["success"] = 0;
    $jsonResponse["message"] = "some error";

    // echo no users JSON
    print json_encode($jsonResponse);
}




/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα
```

```
*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

APXEIO customerSelectDriver.php

```
<?php
/*
@info
Αρχείο που επιστρέφει τα στοιχεία του πελάτη που επέλεξε έναν συγκεκριμένο οδηγό από το
listview.

@details
Συγκεκριμένα επιστρέφεται η τοποθεσία του πελάτη,στον συγκεκριμένο οδηγό που επέλεξε από το
listview, ώστε
στην οθόνη του οδηγού να ζωγραφιστεί στην ανάλογη τοποθεσία ο πελάτης

*/

header('Content-type=application/json; charset=utf-8');
$jsonResponse = array();


require_once __DIR__ .'/db_connect.php';
$database = new DB_CONNECT();

$customerID = null;
$orderID = null;
if(isset($_REQUEST["customerid"]) && isset($_REQUEST["orderid"]) ){
    $customerID = $_REQUEST["customerid"];
    $orderID = $_REQUEST["orderid"];
}
else{
    $customerID = "-";
    $orderID = "-";

}

mysql_query("SET NAMES UTF8");
$sql = "SELECT td.driverDeviceID,o.customerLocation
                        FROM taxidriver td INNER JOIN orders o ON (td.driverID =
o.taxidriverFK)
                        WHERE o.customerFK = (SELECT customerID FROM customer WHERE
customerDeviceID ='".$customerID."') AND o.orderState = 1
                        AND o.orderID = '".$orderID."' " ;


$query  = mysql_query($sql) or die(mysql_error());

if (mysql_num_rows($query) > 0) {

    $rows = mysql_fetch_array($query);

    $location = splitCustomerLocation( $rows["customerLocation"] );
    $jsonResponse["customerLat"] = $location[0];
    $jsonResponse["customerLon"] = $location[1];
    $jsonResponse["driverDevID"] = $rows["driverDeviceID"];
    $jsonResponse["success"] = 1;

    // echoing JSON response
   print json_encode($jsonResponse);

}
else {
    $jsonResponse["success"] = 0;
```

```
    $jsonResponse["message"] = "some error";

    // echo no users JSON
    print json_encode($jsonResponse);
}

/*
@info
Συνάρτηση για τον διαχωρισμό της τοποθεσίας

@details
Η τοποθεσία βρίσκεται στη βάση σε μορφή lat+lon, εμείς τη διαχωρίζουμε
και επιστρέφουμε το array με πρώτο στοιχείο το lat και δεύτερο lon

@return array Με τα διαχωρισμένα στοιχεία

*/
function splitCustomerLocation($location){
    if($location != null){
        $loc = explode("+",$location);
        return $loc;
    }
    else
        return null;
}



/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}

?>
```

ΑΡΧΕΙΟ db_config.php

```
<?php

/*
@info
Αρχείο που ορίζει τις μεταβλητές για τη σύνδεση της βάσης

*/

define('DB_USER', "root"); // db user

define('DB_PASSWORD',""); // db password (mention your db password here)

define('DB_DATABASE', "taxianytime"); // database name

define('DB_SERVER', "localhost"); // db server


?>
```

ΑΡΧΕΙΟ db_connect.php

```php
<?php

/*
@info
Κλάση για τη διαχείριση της σύνδεσης με τη βάση

@details
Συγκεκριμένα επιστρέφεται η τοποθεσία του πελάτη,στον συγκεκριμένο οδηγό που επέλεξε από το
listview, ώστε
στην οθόνη του οδηγού να ζωγραφιστεί στην ανάλογη τοποθεσία ο πελάτης

*/
class DB_CONNECT {

    function __construct() {

        $this->connect();
    }

    function __destruct(){
        // closing db connection
        $this->close();
    }


    function connect(){

        require_once __DIR__ .'/db_config.php';
        $con = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or die(mysql_error());
        $db = mysql_select_db(DB_DATABASE) or die(mysql_error()) or die(mysql_error());
        return $con;
    }

    function close(){
        mysql_close();
    }

}

?>
```

ΑΡΧΕΙΟ driversListview.php

```php
<?php
/*
@info
Αρχείο για την προβολή του listview με τους οδηγούς

@details
Εμφανίζονται στον πελάτη όλοι οι οδηγοί οι οποίοι έδειξαν ενδιαφέρον.
Τα στοιχεία που φαίνονται είναι όνομα,rate,distance,taxiplatenumber

*/
header('Content-type=application/json; charset=utf-8');
$jsonResponse = array();


require_once __DIR__ .'/db_connect.php';
$database = new DB_CONNECT();

$customerDevice = null;
if(isset($_REQUEST["customerDevID"]) ){
    $customerDevice = $_REQUEST["customerDevID"];
}
else
    $customerDevice ="-";


mysql_query("SET NAMES UTF8");

$query  = mysql_query("SELECT
orderID,name,sirname,town,totalRate,driverImageUrl,taxiPlateNumber,driverDeviceID,distance
                        FROM orders  INNER JOIN taxidriver  ON (driverID = taxidriverFK)
                        WHERE customerFK = (SELECT customerID FROM customer WHERE
customerDeviceID = '".$customerDevice."') AND orderState = 1 ")
```

```
                    or die(mysql_error());

if (mysql_num_rows($query) > 0){

    $jsonResponse["drivers"] = array();
    $driver = array();
    while ($row = mysql_fetch_array($query)){
        $driver["orderid"] = $row["orderID"];
        $driver["name"] = $row["sirname"]." ".$row["name"];
        $driver["rate"] = $row["totalRate"];
        $driver["distance"] = $row ["distance"];
        $driver["driverImageUrl"] = $row["driverImageUrl"];
        $driver["taxiPlateNumber"] = $row["taxiPlateNumber"];
        $driver["driverDeviceID"] = $row["driverDeviceID"];

        array_push($jsonResponse["drivers"],$driver );
    }

    $jsonResponse["success"] = 1;


    // echoing JSON response
    print json_encode($jsonResponse);
}
else {
    $jsonResponse["success"] = 0;
    $jsonResponse["message"] = "No drivers found";

    // echo no users JSON
    print json_encode($jsonResponse);
}



/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}



?>
```

APXEIO fetchDataToEdit.php

```
<?php
/*
@info
Κλάση για την προσκόμιση των στοιχείων εγγραφής χρήστη

@details
Επιστρέφονται τα στοιχεία του χρήστη που χρειάζονται για να γίνει το edit.
*/

header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$fetchData = new FetchData();
$fetchData->fetchDataToEdit($_REQUEST["deviceid"],$_REQUEST["usertype"]) ;

//$fetchData->fetchDataToEdit("/357777961700933","driver") ;

//=============================================================================================
```

```php
class FetchData
{
    private $jsonResponse;

    function __construct(){

    }
//=============================================================================================
====================
    public function fetchDataToEdit($deviceID,$userType){
        if( isset($deviceID) && isset($userType) ){
            switch($userType){
                case "customer":{

                    $sql ="SELECT name,sirname,cellphone,town FROM customer
                           WHERE customerDeviceID ='".$deviceID."' LIMIT 1";

                    break;
                }
                case "driver":{
                    $sql ="SELECT name,sirname,cellphone,town,taxiPlateNumber,driverImageUrl
FROM taxidriver
                           WHERE driverDeviceID ='".$deviceID."' LIMIT 1";
                    break;
                }

            }

            mysql_query("SET NAMES UTF8");
            $result = mysql_query($sql);


            if (mysql_num_rows($result) >0 ){
                while( $row = mysql_fetch_assoc($result) ){
                    $rows[] = $row ; //γέμισμα με τα αποτελέσματα
                }
                $this->jsonResponse["editDetails"] = $rows;
                $this->jsonResponse["success"] = 1;

                // echoing JSON response
                print json_encode($this->jsonResponse);
            }
            else {
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "not successful";
                print json_encode($this->jsonResponse);
            }

        }
        else{
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "not successful";

            // echo no users JSON
            print json_encode($this->jsonResponse);
        }

    }//end function

}//end class


function debug($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

APXEIO getdriverinfo.php

```php
<?php

/*
Βοηθητικό αρχείο για να λάβουμε τα στοιχεία των οδηγών οι οποίοι είναι
διαθέσιμοι.(όνομα,rate,id συσκευής )

*/

header('Content-Type: application/json; charset=utf-8');
$jsonResponse = array();
require_once __DIR__ .'/db_connect.php';
$database = new DB_CONNECT();


mysql_query("SET NAMES UTF8");

$query  = mysql_query("SELECT sirname,totalRate,driverImageUrl,driverDeviceID
                    FROM taxidriver where isAvailable = 1") or die(mysql_error());

if (mysql_num_rows($query) > 0)
{

    $jsonResponse["drivers"] = array();
    $driver = array();
    while ($row = mysql_fetch_array($query))
    {

        $driver["sirname"] = $row["sirname"];
        $driver["rate"] = $row["totalRate"];
        $driver["driverImageUrl"] = $row["driverImageUrl"];
        $driver["driverDeviceID"] = $row["driverDeviceID"];


        array_push($jsonResponse["drivers"],$driver );
    }

    $jsonResponse["success"] = 1;


    // echoing JSON response
    print json_encode($jsonResponse);

}
else
{
    $jsonResponse["success"] = 0;
    $jsonResponse["message"] = "No driver found";

    // echo no users JSON
    print json_encode($jsonResponse);
}




?>
```

APXEIO makeCall.php

```php
<?php

/*
@info
Κλάση για την πραγματοποίηση κλήσης/sms σε περίπτωση ανάγκης

@details
Λαμβάνει από τη βάση το κινητό τηλέφωνο του συγκεκριμένου οδηγού ή πελάτη ανάλογα

*/

header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
```

```php
$database = new DB_CONNECT();

$caller = new call();
$caller->makecall( $_REQUEST["client"] ,$_REQUEST["Deviceid"]) ;


//==========================================================================
class call
{
    private $jsonResponse;
    private $client;
    private $deviceid;
    private $sql;
    private $query;
    private $row;

    function __construct()
    {
        $this->jsonResponse = array();
        $this->client = null;
        $this->deviceid = null;
        $this->sql = "";
        $this->query = null;
        $this->row = null;
    }

    //set ========================================================================
    public function setClient($subject)
    {
        $this->client = $subject;
    }

    public function setDeviceid($subject)
    {
        $this->deviceid = $subject;
    }

    public function setSql($subject)
    {
        $this->sql = $subject;
    }

    //get
    public function getClient()
    {
        return $this->client;
    }

    public function getSql()
    {
        return $this->sql;
    }

    public function getDeviceid()
    {
        return $this->deviceid ;
    }

    public function getQuery()
    {
        return $this->query;
    }
//============================================================================================

    public function makeCall($client,$devid)
    {
        if(isset( $client) && isset( $devid ) )
        {

            $this->setDeviceid($devid);
            mysql_query("SET NAMES UTF8");

            switch($client)
            {
                case "customer" :
                {
                    $this->setSql ( "SELECT cellphone FROM taxidriver where driverDeviceID =
'".$this->getDeviceid()."' LIMIT 1" ) ;
                    break;
                }
                case "driver" :
```

```
                {
                    $this->setSql ("SELECT cellphone FROM customer where customerDeviceID =
'".$this->getDeviceid()."' LIMIT 1" ) ;
                    break;
                }

            }
            $this->query = mysql_query( $this->getSql() );


            if (mysql_num_rows( $this->getQuery() ) > 0)
            {

                // $jsonResponse["coordinates"] = array(); //Σύνθετο array
                $this->row = mysql_fetch_array( $this->getQuery() );
                $this->jsonResponse["cellphone"] = $this->row["cellphone"];
                $this->jsonResponse["success"] = 1;

                // echoing JSON response
                print json_encode($this->jsonResponse);
            }
            else
            {
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "not successful";
                print json_encode($this->jsonResponse);
            }

        }
        else
        {
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "not successful";

            // echo no users JSON
            print json_encode($this->jsonResponse);

        }

    }

}


function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

APXEIO makeCommentsRatings.php

```php
<?php
/*
@info
Κλάση για την αποθήκευση σχολίων και rating

@details
Εισάγεται στην βάση το συνολικό rating και τα σχόλια

*/

header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$commentRating = new CommentsRating();

$commentRating->makeCommentsAndRate($_REQUEST["rate"],$_REQUEST["comment"],$_REQUEST["driverDevId"],$_REQUEST["orderid"],$_REQUEST["customerDevID"]);


//=============================================================================================
class CommentsRating
{

    private $jsonResponse;
    private $comments;
    private $rate;
    private $orderid;
    private $driverDeviceId;
    private $totalRating;
    private $customerDevID;


    function __construct()
    {
        $this->jsonResponse = array();
        $this->comments = null;
        $this->rate = null;
        $this->orderid = null;
        $this->driverDeviceId = null;
        $this->totalRating = null;
        $this->customerDevID = null;

    }


    public function makeCommentsAndRate($rate,$comment,$driverdevid,$orderid,$customerDevid)
    {
        if(isset($rate) && isset($comment) && isset ($driverdevid) && isset($orderid) && isset($customerDevid)  )  //Αν έχουν τιμή
        {
            $this->comments = $comment;
            $this->rate = $rate;
            $this->driverDeviceId = $driverdevid;
            $this->orderid = $orderid;
            $this->customerDevID = $customerDevid;

             mysql_query("SET NAMES UTF8");
             $query_make_comment = mysql_query(" INSERT INTO comments
                                                 SET comment ='".$this->comments."',
                                                 rating = '".$this->rate."',
                                                 orderid = '".$this->orderid."' ")   ;
            //Υπολογίζει το συνολικό rating άθροισμα/πλήθος από τον πίνακα comments
             $results = mysql_query("
             SELECT SUM(c.rating)/COUNT(*)
             FROM taxidriver td INNER JOIN orders o ON o.taxidriverFK = td.driverID INNER JOIN comments c ON o.orderID = c.orderid
             WHERE td.driverDeviceID = '".$this->driverDeviceId."'
             AND o.orderState = 3 LIMIT 1 " );


            if( $query_make_comment == true && mysql_num_rows($results) > 0) //Αν έγινε κανονικά εισαγωγή
            {
```

```
                $this->totalRating = $this->calculateRating( mysql_fetch_array($results) ) ;


                 mysql_query("UPDATE taxidriver
                            SET totalRate = '".$this->totalRating."'
                            WHERE driverDeviceID = '".$this->driverDeviceId."' LIMIT 1 " );

                // successfully updated
                $this->jsonResponse["success"] = 1;
                $this->jsonResponse["message"] = "successfully inserted.";

                // echoing JSON response
                echo json_encode($this->jsonResponse);
            }
            else
            {
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "Required field(s) is missing";
                echo json_encode($this->$jsonResponse);
            }

        }

        else
        {
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "Required field(s) is missing";
            echo json_encode($this->$jsonResponse);
        }

        //Καθαρίζουμε πρώτα τις μη ολοκληρωμένες παραγγελίες , ώστε να μας εμφανίζονται μόνο
οι καινούργιες
        mysql_query("DELETE FROM orders WHERE orderState<>3 AND customerFK =(SELECT
customerID FROM customer WHERE customerDeviceID = '".$this->customerDevID."' LIMIT 1) ");

    }


/*
@info
Συνάρτηση για τον υπολογισμό του τελικου rating

@details
Στρογγυλοποιεί το τελικό rating στην πλησιέστερη μισάδα (βλ παράδειγμα κάτω)

@source
To round to the nearest half number:

1. Multiply by 2
2. Add 0.5.
3. Erase the decimal part of the number (truncate).
4. Divide by 2

Examples:
7.2 -> 14.4 -> 14.9 -> 14 -> 7
7.4 -> 14.8 -> 15.3 -> 15 -> 7.5
7.6 -> 15.2 -> 15.7 -> 15 -> 7.5
7.8 -> 15.6 -> 16.1 -> 16 -> 8
link: http://answers.yahoo.com/question/index?qid=20061212113854AABP8Gj

*/


//το αποτέλεσμα του fetching είναι το τελικό rating
private function calculateRating($parameter){
        $piece = explode(".", ( $parameter[0]*2 + 0.5 ) );
        return ( $piece[0]/2 ) ;
    }

}//end class




function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);
```

```
}


?>
```

APXEIO makeEdit.php

```
<?php

/*
@info
Κλάση για την αλλαγή στοιχείων των χρηστών

@details
Δίνουμε την δυνατότητα στον χρήστη να μπορεί να αλλάξει στοιχεία όπως όνομα,επίθετο,τηλ κτλ

*/

header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();

$editor = new Edit();


if($_REQUEST["usertype"] == "customer"){

$editor->makeEdit(mysql_real_escape_string( $_REQUEST["cellphone"]
),mysql_real_escape_string( $_REQUEST["password"]),
                        mysql_real_escape_string(
$_REQUEST["town"]),mysql_real_escape_string( $_REQUEST["deviceid"]),
                        null,null, $_REQUEST["usertype"] );


}
else{
$editor->makeEdit(mysql_real_escape_string( $_REQUEST["cellphone"]
),mysql_real_escape_string( $_REQUEST["password"]),
                        mysql_real_escape_string(
$_REQUEST["town"]),mysql_real_escape_string( $_REQUEST["deviceid"]),
                        mysql_real_escape_string(
$_REQUEST["taxiplate"]),$_REQUEST["imageUrl"], $_REQUEST["usertype"] );

}

//for debugging
/*
$editor->makeEdit("new","new",
                        "serres_new","/567586867956",
                        "ere-34523", "driver" );

*/


//======================================================================
class Edit
{

    private $jsonResponse;
    private $cellphone;
    private $password ;
    private $town;
    private $deviceid;
    private $imageurl;

    private $taxiplate;


    private $sql ;

    function __construct()
    {
        $this->jsonResponse = array();
        $this->cellphone = NULL;
        $this->password = NULL;
        $this->town = NULL;
```

```
        $this->deviceid = NULL;

        $this->taxiplate = NULL;
        $this->imageurl = "";

        $this->sql = "";

    }

    public function
makeEdit($cellphone,$password,$town,$deviceid,$taxiplate,$imageurl,$userType) {

        switch($userType)
        {
            case "customer":{

                if( isset($cellphone) && isset($password) && isset($deviceid) &&
isset($town)  ) {


                    $this->cellphone = $cellphone;
                    $this->password = $password;
                    $this->deviceid = $deviceid;
                    $this->town = $town;

                        $this->sql = "UPDATE customer
                                      SET
                                      cellphone = '".$this->cellphone."',
                                      password = '".$this->password."',
                                      town = '".$this->town."'
                                      WHERE customerDeviceID = '".$this->deviceid."' ";


                }
                break;
            }
            case "driver":{

                $this->imageurl = $imageurl ;
                if( isset($cellphone)  && isset($password) && isset($deviceid) &&
isset($town) && isset($taxiplate) )  //Αν έχουν τιμή
                {
                    $this->cellphone = $cellphone;
                    $this->password = $password;
                    $this->town = $town;
                    $this->deviceid = $deviceid;
                    $this->taxiplate = $taxiplate;

                    if($this->imageurl == ""){
                        $this->imageurl ="-" ;
                    }

                    $this->sql = "UPDATE taxidriver
                                      SET
                                      cellphone = '".$this->cellphone."',
                                      password = '".$this->password."',
                                      town = '".$this->town."',
                                      driverImageUrl = '".$this->imageurl."',
                                      taxiPlateNumber = '".$this->taxiplate."'
                                      WHERE driverDeviceID = '".$this->deviceid."' ";

                }
                break;
            }

        } //end switch


            mysql_query("SET NAMES UTF8");
             mysql_query($this->sql);

            if( mysql_affected_rows() > 0 ) // αν έγινε ενημέρωση
            {
                $this->jsonResponse["success"] = 1;
                $this->jsonResponse["message"] = "successfully updated";

                // echoing JSON response
                echo json_encode($this->jsonResponse);
            }
            else
            {
```

```
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "Error";

                echo json_encode($this->jsonResponse);
            }

    }

}


//============================================
//for debug
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

APXEIO makeLogin.php

```
<?php
/*
@info
Κλάση υπεύθυνη για την διαδικασία ορθής εισαγωγής χρήστη.

*/

//header('Content-Type: text/html; charset=utf-8');
header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$login = new login();

$login->makeLogin( mysql_real_escape_string( $_REQUEST["username"]
),mysql_real_escape_string( $_REQUEST["password"]
),$_REQUEST["deviceid"],$_REQUEST["type"]);

//======================================================================
class login{
    private $jsonResponse;
    private $username ;
    private $password ;
    private $deviceid;
    private $usertype ;
    private $queryLogin;
    private $sql ;
    private $report_value ;

    function __construct(){
        $this->jsonResponse = array();
        $this->username = null;
        $this->password = null;
        $this->usertype = null;
        $this->deviceid = null;
        $this->queryLogin = "";
        $this->sql = "";
        $this->report_value = 5;
    }

//set
    public function setUsername($subject){
        $this->username = $subject;
    }


    public function setPassword($subject){
```

```
        $this->password = $subject;
    }

    public function setUserType($subject){
        $this->usertype  = $subject;
    }


    //get
    public function getUserName(){
        return $this->username;
    }

    public function getSql(){
        return  $this->sql ;
    }

    public function getPassWord(){
        return $this->password ;
    }
    public function getQueryLogin(){
        return $this->queryLogin;
    }

    public function GetUserType(){
        return $this->usertype;
    }


/*
@info
Μέθοδος για την διαδικασία ορθής εισαγωγής χρήστη.
Σε περίπτωση σωστής εισαγωγής έχουμε success "1" διαφορετικά "0"
*/

    public function makeLogin($user,$pass,$devid,$type)
    {
        if( isset($user) && isset($pass) && isset($devid) && isset($type) )  //Αν έχουν τιμή
        {
            $this->setUsername($user);
            $this->setPassword($pass);
            $this->setUserType($type);
            $this->deviceid = $devid;

            switch($this->GetUserType())
            {
                case "customer" :
                {


                    $sqlCheckForBan = mysql_query("SELECT COUNT(*)
                                                   FROM report
                                                   WHERE reportTo = (SELECT customerID FROM
customer WHERE customerDeviceID = '".$this->deviceid."' LIMIT 1)
                                                   AND byUser = 'driver' LIMIT 1"); //driver
γιατι έγινε από οδηγό το report

                    $row = mysql_fetch_array($sqlCheckForBan);
                    if ($row[0] >= $this->report_value ) //Αν έχει πάνω από 5 reports, στο
row[0] είναι το αποτέλεσμα
                    {

                            $this->sql = ""; //δίνουμε άδειο sql άρα δεν θα εκτελεστεί τπτ
                    }
                    else
                    {

                            $this->sql = "SELECT username,password
                                    FROM  customer
                                    WHERE username = '".$this->getUserName()."' AND password
= '".$this->getPassWord()."' LIMIT 1 "  ;
                    }

                    break;
                }
                case "driver" :
                {

                    $sqlCheckForBan = mysql_query("SELECT COUNT(*)
                                                   FROM report
```

```
                                                WHERE reportTo = (SELECT driverID FROM
taxidriver WHERE driverDeviceID = '".$this->deviceid."' LIMIT 1)
                                                AND byUser = 'customer' LIMIT 1");// customer
γιατί έχει από πελάτη το report

                    $row = mysql_fetch_array($sqlCheckForBan);
                    if ($row[0] >= $this->report_value ) //Αν έχει πάνω από 5 reports, στο
row[0] είναι το αποτέλεσμα από το query
                    {

                            $this->sql = ""; //δίνουμε άδειο sql άρα δεν θα εκτελεστεί τπτ
                    }
                    else
                    {

                        $this->sql ="SELECT username,password
                        FROM  taxidriver
                        WHERE username = '".$this->getUserName()."' AND password = '".$this-
>getPassWord()."' LIMIT 1 ";

                    }
                    break;
                }
                default:
                    $this->sql ="";
                    break;
            }

            $this->queryLogin = mysql_query($this->getSql());

            if(mysql_num_rows( $this->getQueryLogin() ) == 1){ // αν βρέθηκε


                    $this->jsonResponse["success"] = 1;
                    $this->jsonResponse["message"] = "successfully logged in.";

                echo json_encode($this->jsonResponse);
            }
            else{
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "Wrong login";

                echo json_encode($this->jsonResponse);
            }
        }
        else
        {
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "wrong login or reported ";

            echo json_encode($this->jsonResponse);
        }
    }


}



/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

ΑΡΧΕΙΟ makeOrder.php

```php
<?php
/*
@info
Κλάση υπεύθυνη για την δημιουργία παραγγελίας.
Θέτουμε την κατάσταση της παραγγελίας σε 1.

*/

header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$neworder = new Orders();
$neworder->makeOrder( fixLatLon($_REQUEST["customerlat"])
,fixLatLon($_REQUEST["customerlon"]),

fixLatLon($_REQUEST["driverlat"]),fixLatLon($_REQUEST["driverlon"]),$_REQUEST["customerDevic
e"],
                    $_REQUEST["driverDevice"],$_REQUEST["distance"]);


//san orismata vazoume ta requests



//================================================================================

class Orders{
    private $jsonResponse;

    private $CustomerLatitude;
    private $CustomerLongitude;
    private $DriverLatitude;
    private $DriverLongitude;
    private $customerid;
    private $driverid;
    private $orderid;
    private $distance;


    function __construct(){
        $this->jsonResponse = array();

        $this->CustomerLatitude = null;
        $this->CustomerLongitude= null;
        $this->DriverLatitude = null;
        $this->DriverLongitude = null;
        $this->driverid = null;
        $this->customerid = null;
        $this->orderid = null;
        $this->distance = null;

    }

    //set-get magic methods
    public function __get($property) {
            if (property_exists($this, $property)) {
                return $this->$property;
            }
    }

    public function __set($property, $value) {
        if (property_exists($this, $property)) {
            $this->$property = $value;
        }
    }

/*
@info
Μέθοδος για την δημιουργία παραγγελίας.

@details
Δέχεται ώς ορίσματα( requests) τα γ.μ/γ.π οδηγού και πελάτη , τα id των συσκευών και την
απόσταση τους ,
```

```
τα οποία καταχωρούνται στη βάση στον πίνακα order .
Έχουμε δηλαδή μια νέα παραγγαλία με orderState = 1;

*/
public function
makeOrder($customerlat,$customerlon,$driverlat,$driverlon,$customerdevice,$driverdevice,$dis
tance){
    if(isset($customerlat) && isset($customerlon) && isset($driverlat) && isset($driverlon)
&& isset($customerdevice) && isset($driverdevice) && isset($distance) ){
        $this->CustomerLatitude = $customerlat;
        $this->CustomerLongitude = $customerlon;
        $this->DriverLatitude = $driverlat;
        $this->DriverLongitude = $driverlon;
        $this->driverid = $driverdevice;
        $this->customerid =$customerdevice;
        $this->distance = $distance;



        $sql = "INSERT INTO orders
                SET
                customerLocation = '".$this->CustomerLatitude."+".$this-
>CustomerLongitude."',
                driversLocation =  '".$this->DriverLatitude."+".$this->DriverLongitude."',
                date = NOW(),
                taxidriverFK = (SELECT driverID FROM taxidriver WHERE driverDeviceID =
'".$this->driverid."' ),
                customerFK = (SELECT customerID FROM customer WHERE customerDeviceID =
'".$this->customerid."' ),
                distance = '".$this->distance."',
                orderState = 1" ;

        $insert_query = mysql_query($sql);

        $this->orderid = mysql_insert_id();


        //Αν έγινε κανονικά εισαγωγή,ενημερώνουμε τον πίνακα με τις ανάλογες τιμές
        if($insert_query) {
            // successfully updated
            $this->jsonResponse["orderid"] = $this->orderid ;
            $this->jsonResponse["success"] = 1;
            $this->jsonResponse["message"] = "order successfully inserted.";

            // echoing JSON response
            echo json_encode($this->jsonResponse);
        }
        else{
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "error";
            echo json_encode($this->jsonResponse);
        }

    }
    else{

        $this->jsonResponse["success"] = 0;
        $this->jsonResponse["message"] = "error";
        echo json_encode($this->$jsonResponse);
    }

 }//end function


}//end class

/*
@info
Συνάρτηση για να διορθώνει τα δεκαδικά από τα γεωγραφικά μήκη/πλάτη

@example
πχ αν το εισερχόμενο string είναι : 23.345678653434  (12 δεκαδικά)
θα αλλάξει σε 23.34567865 (8 δεκαδικά)
αν ήταν 23.345 (3 δεκαδικά)
θα αλλάξει σε 23.34500000 (8 δεκαδικά)

*/
function fixLatLon($string)
{
    $latpieces = explode(".",$string); //σπάμε το string
```

```
    $fixedSize = 8; // πλήθος δεκαδικών που θέλουμε

    if( (strlen($latpieces[1]) > $fixedSize) ) //αν το δεύτερο μέρος είναι μεγαλύτερο
    {
        $latpieces[1] = substr_replace($latpieces[1] ,"",-(strlen($latpieces[1])-
$fixedSize)); //αφαιρούμε τα πλεονάζοντα δεδαδικά ώστε να είναι όσο και το fixed size
        //echo $latpieces[1];
        return $latpieces[0].".".$latpieces[1]; // επιστρέφουμε το επιθυμητό
    }
    else if(strlen($latpieces[1]) < $fixedSize) //αν το δεύτερο μέρος είναι μικρότερο
    {
        $zeros = null;
        for($i=0;$i<($fixedSize - strlen($latpieces[1]));$i++)  //γεμίζουμε όσα λείπουν με
"0"
        {
            $zeros[$i] = "0";

        }
        return $latpieces[0].".".$latpieces[1].implode($zeros);
    }
    else
        return $latpieces[0].".".$latpieces[1]; // αν είναι ίδιο το μέθεθος των δεκαδικών
}


/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}



?>
```

APXEIO makeRegister.php

```
<?php
/*
@info
Κλάση υπεύθυνη για την δημιουργία παραγγελίας.
Θέτουμε την κατάσταση της παραγγελίας σε 1.

*/

header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$neworder = new Orders();
$neworder->makeOrder( fixLatLon($_REQUEST["customerlat"])
,fixLatLon($_REQUEST["customerlon"]),

fixLatLon($_REQUEST["driverlat"]),fixLatLon($_REQUEST["driverlon"]),$_REQUEST["customerDevic
e"],
                      $_REQUEST["driverDevice"],$_REQUEST["distance"]);


//san orismata vazoume ta requests
```

```
//============================================================================

class Orders{
    private $jsonResponse;

    private $CustomerLatitude;
    private $CustomerLongitude;
    private $DriverLatitude;
    private $DriverLongitude;
    private $customerid;
    private $driverid;
    private $orderid;
    private $distance;


    function __construct(){
        $this->jsonResponse = array();

        $this->CustomerLatitude = null;
        $this->CustomerLongitude= null;
        $this->DriverLatitude = null;
        $this->DriverLongitude = null;
        $this->driverid = null;
        $this->customerid = null;
        $this->orderid = null;
        $this->distance = null;

    }

    //set-get magic methods
    public function __get($property) {
            if (property_exists($this, $property)) {
                return $this->$property;
            }
    }

    public function __set($property, $value) {
        if (property_exists($this, $property)) {
            $this->$property = $value;
        }
    }

/*
@info
Μέθοδος για την δημιουργία παραγγελίας.

@details
Δέχεται ώς ορίσματα( requests) τα γ.μ/γ.π οδηγού και πελάτη , τα id των συσκευών και την
απόσταση τους ,
τα οποία καταχωρούνται στη βάση στον πίνακα order .
Έχουμε δηλαδή μια νέα παραγγαλία με orderState = 1;

*/
public function
makeOrder($customerlat,$customerlon,$driverlat,$driverlon,$customerdevice,$driverdevice,$dis
tance){
    if(isset($customerlat) && isset($customerlon) && isset($driverlat) && isset($driverlon)
&& isset($customerdevice) && isset($driverdevice) && isset($distance) ){
        $this->CustomerLatitude = $customerlat;
        $this->CustomerLongitude = $customerlon;
        $this->DriverLatitude = $driverlat;
        $this->DriverLongitude = $driverlon;
        $this->driverid = $driverdevice;
        $this->customerid =$customerdevice;
        $this->distance = $distance;



        $sql = "INSERT INTO orders
                SET
                customerLocation = '".$this->CustomerLatitude."+".$this-
>CustomerLongitude."',
                driversLocation =  '".$this->DriverLatitude."+".$this->DriverLongitude."',
                date = NOW(),
                taxidriverFK = (SELECT driverID FROM taxidriver WHERE driverDeviceID =
'".$this->driverid."' ),
                customerFK = (SELECT customerID FROM customer WHERE customerDeviceID =
'".$this->customerid."' ),
                distance = '".$this->distance."',
```

```
                orderState = 1" ;

        $insert_query = mysql_query($sql);

        $this->orderid = mysql_insert_id();


        //Αν έγινε κανονικά εισαγωγή,ενημερώνουμε τον πίνακα με τις ανάλογες τιμές
        if($insert_query) {
            // successfully updated
            $this->jsonResponse["orderid"] = $this->orderid ;
            $this->jsonResponse["success"] = 1;
            $this->jsonResponse["message"] = "order successfully inserted.";

            // echoing JSON response
            echo json_encode($this->jsonResponse);
        }
        else{
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "error";
            echo json_encode($this->jsonResponse);
        }

    }
    else{

        $this->jsonResponse["success"] = 0;
        $this->jsonResponse["message"] = "error";
        echo json_encode($this->$jsonResponse);
    }

 }//end function


}//end class

/*
@info
Συνάρτηση για να διορθώνει τα δεκαδικά από τα γεωγραφικά μήκη/πλάτη

@example
πχ αν το εισερχόμενο string είναι : 23.345678653434  (12 δεκαδικά)
θα αλλάξει σε 23.34567865 (8 δεκαδικά)
αν ήταν 23.345 (3 δεκαδικά)
θα αλλάξει σε 23.34500000 (8 δεκαδικά)

*/
function fixLatLon($string)
{
    $latpieces = explode(".",$string); //σπάμε το string
    $fixedSize = 8; // πλήθος δεκαδικών που θέλουμε

    if( (strlen($latpieces[1]) > $fixedSize) ) //αν το δεύτερο μέρος είναι μεγαλύτερο
    {
        $latpieces[1] = substr_replace($latpieces[1] ,"",-(strlen($latpieces[1])-
$fixedSize)); //αφαιρούμε τα πλεονάζοντα δεδαδικά ώστε να είναι όσο και το fixed size
        //echo $latpieces[1];
        return $latpieces[0].".".$latpieces[1]; // επιστρέφουμε το επιθυμητό
    }
    else if(strlen($latpieces[1]) < $fixedSize) //αν το δεύτερο μέρος είναι μικρότερο
    {
        $zeros = null;
        for($i=0;$i<($fixedSize - strlen($latpieces[1]));$i++)  //γεμίζουμε όσα λείπουν με
"0"
        {
            $zeros[$i] = "0";

        }
        return $latpieces[0].".".$latpieces[1].implode($zeros);
    }
    else
        return $latpieces[0].".".$latpieces[1]; // αν είναι ίδιο το μέθεθος των δεκαδικών
}


/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
```

```
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}




?>
```

APXEIO makeReport.php

```
<?php
/*
@info
Κλάση υπεύθυνη για την δημιουργία αναφοράς

*/
//header('Content-Type: text/html; charset=utf-8');
header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();

$rep = new report();
$rep->makeReport(
$_REQUEST["getReasons"],$_REQUEST["orderid"],$_REQUEST["reportfrom"],$_REQUEST["reportTo"],$
_REQUEST["userType"] );



//============================================================================================
class report
{
    private $jsonResponse ;
    private $reportReason ;
    private $orderid;
    private $reportFrom;
    private $reportTo;
    private $sql;
    private $query;

    function __construct()
    {
        $this->jsonResponse = array() ;
        $this->reportReason = null ;
        $this->sql = "";
        $this->query = null;
        $this->orderid = null;
        $this->reportFrom = null;
        $this->reportTo = null;
        $this->userType = null;
    }

//============================================================


    public function setQuery($subject)
    {
        $this->query = $subject;
    }

    public function setSql($subject)
    {
        $this->sql = $subject;
    }

    public function getSql()
    {
        return $this->sql;
    }
```

```
    public function getQuery()
    {
        return $this->query;
    }
//=========================================================
/*
@info
Μέθοδος για την αναφορά
@details
Δημιουργία αναφοράς με τα εξής στοιχεία :
-λόγοι
-orderid
-από
-σε ποιόν

Επιστρέφει σε μορφή json την απόκριση "success"= 1 σε επιτυχή αλλαγή και "success"= 0 σε μή
επιτυχή

*/
    public function makeReport($reasons,$orderid,$repFrom,$repTo,$usrType)
    {
            if(isset ($reasons) && isset($orderid) && isset($repFrom) && isset($repTo) &&
isset($usrType) )
            {
                $this->reportReason = $reasons;
                $this->orderid = $orderid;
                $this->reportFrom = $repFrom;
                $this->reportTo = $repTo;
                $this->userType = $usrType;

                mysql_query("SET NAMES UTF8");

                switch( $this->userType )
                {
                    case "customer":
                    {
                        $this->setSql( "INSERT INTO report
                                SET reportReason = '".$this->reportReason."' ,
                                orderFK = '".$this->orderid."',
                                byUser = 'customer',
                                reportFrom = ( SELECT customerID from customer WHERE
customerDeviceID = '".$this->reportFrom."' LIMIT 1),
                                reportTo = ( SELECT driverID from taxidriver WHERE
driverDeviceID = '".$this->reportTo."' LIMIT 1) " );
                        break;
                    }
                    case "driver":
                    {
                        $this->setSql( "INSERT INTO report
                                SET reportReason = '".$this->reportReason."' ,
                                orderFK = '".$this->orderid."',
                                byUser = 'driver',
                                reportFrom = ( SELECT driverID from taxidriver WHERE
driverDeviceID = '".$this->reportFrom."' LIMIT 1),
                                reportTo = ( SELECT customerID from customer WHERE
customerDeviceID = '".$this->reportTo."' LIMIT 1) ") ;
                        break;
                    }

                }



                $this->setQuery ( mysql_query($this->getSql()) ) ;
                if( $this->getQuery() ) //Αν έγινε κανονικά εισαγωγή
                {
                    // successfully updated
                    $this->jsonResponse["success"] = 1;
                    $this->jsonResponse["message"] = "successfully reported.";

                    // echoing JSON response
                    echo json_encode($this->jsonResponse);
                }
                else
                {
                    $this->jsonResponse["success"] = 0;
                    $this->jsonResponse["message"] = "not reported";
                    echo json_encode($this->jsonResponse);
                }
            }
            else
```

```
            {
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "not reported";
                echo json_encode($this->jsonResponse);
            }

    } //end function



} //end class


/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

ΑΡΧΕΙΟ orderConfirmation.php

```
<?php
/*
@info
Αρχείο για την ενημέρωση της κατάστασης της παραγγελίας.
Θυμίζουμε :
orderState = 1 : Υπάρχει έτοιμη παραγγελία
orderState = 2 : Την έχει αποδεχθεί ο οδηγός
orderState = 3 : Την έχει αποδεχθεί ο πελάτης, και αυτό είναι το τελικό στάδιο της
παραγγελίας.

*/

//header('Content-Type: text/html; charset=utf-8');
header('Content-type=application/json; charset=utf-8');
$jsonResponse = array();
require_once __DIR__ .'/db_connect.php';
$database = new DB_CONNECT();


$deviceID = null;
$orderID = null;

/*
$_REQUEST["type"] = "customer";
$_REQUEST["orderid"] ="1";
$_REQUEST["customerid"] = "/356778781788245";
//$_REQUEST["driverid"] = "/356778781788245";
*/

if( isset($_REQUEST["type"]) ){

    switch( $_REQUEST["type"] ){
        case "driver";{
```

```
            if(isset($_REQUEST["driverid"]) && isset($_REQUEST["orderid"])){
                $deviceID = $_REQUEST["driverid"];
                $orderID = $_REQUEST["orderid"];
            }
            else{
                $deviceID = "-";
                $orderID = "-";
            }

            mysql_query("SET NAMES UTF8");
            $sql = "UPDATE orders
            SET orderState = 2
            WHERE taxidriverFK = (SELECT driverID FROM taxidriver WHERE driverDeviceID =
'".$deviceID."' ) AND orderID ='".$orderID."' " ;
            break;
        }
        case "customer":{
            if(isset($_REQUEST["customerid"]) && isset($_REQUEST["orderid"])){
                $deviceID = $_REQUEST["customerid"];
                $orderID = $_REQUEST["orderid"];
            }
            else{
                $deviceID = "-";
                $orderID = "-";
            }
            mysql_query("SET NAMES UTF8");
            $sql = "UPDATE orders
            SET orderState = 3
            WHERE customerFK = (SELECT customerID FROM customer WHERE customerDeviceID
='".$deviceID."') AND orderID='".$orderID."'  " ;
            break;
        }
    }

    $result = mysql_query($sql);     //mysql_query επιστρέφει TRUE on σε επιτυχία ή FALSE σε
λάθος

    if ( mysql_affected_rows() > 0  ) {
        $jsonResponse["message"] = "updated";
        $jsonResponse["success"] = 1;

        // echoing JSON response
        print json_encode($jsonResponse);
    }
    else {
        $jsonResponse["success"] = 0;
        $jsonResponse["message"] = "some error";

        // echo no users JSON
        print json_encode($jsonResponse);
    }
}
else{
    $jsonResponse["success"] = 0;
    $jsonResponse["message"] = "some error";

    // echo no users JSON
    print json_encode($jsonResponse);

}

/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

APXEIO setdriverAvailability.php

```php
<?php

/*
@info
Κλάση για την  αλλαγή της διαθεσιμότητας του οδηγού.

*/


//header('Content-Type: text/html; charset=utf-8');
header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$available = new availability();
$available->makeAvailability($_REQUEST["available"],$_REQUEST["driverid"]) ;


//=============================================================================================
====================
class availability {

    private $jsonResponse;
    private $driverid;
    private $available;
    private $sql;
    private $query;


    function __construct(){
        $this->jsonResponse = array();
        $this->driverid = null;
        $this->available = null;
        $this->sql = "";
        $this->query = null;

    }

    //set ================================================================
    public function setDriverid($subject)
    {
        $this->driverid = $subject;
    }

    public function setAvailable($subject)
    {
        $this->available = $subject;
    }

    public function setSql($subject)
    {
        $this->sql = $subject;
    }

    //get
    public function getAvailable()
    {
        return $this->available;
    }

    public function getSql()
    {
        return $this->sql;
    }

    public function getDriverid()
    {
        return $this->driverid ;
    }
```

```
    public function getQuery()
    {
        return $this->query ;
    }

//============================================================================
/*
@info
Μέθοδος για αλλαγή της διαθεσιμότητας του οδηγού.

@details
Δέχεται την επιθυμητή διαθεσιμότητα ("1" αν θέλουμε να είναι διαθέσιμος και "0" αν όχι),
καθώς και το
συγκεκριμένο id της συσκευής

Επιστρέφει σε μορφή json την απόκριση "success"= 1 σε επιτυχή αλλαγή και "success"= 0 σε μή
επιτυχή
*/
    public function makeAvailability($available,$driverid)
    {
        if( isset($available) && isset($driverid) ){
            $this->setAvailable($available);
            $this->setDriverid($driverid) ;

            mysql_query("SET NAMES UTF8");
            $this->setSql("UPDATE taxidriver SET isAvailable = '".$this->getAvailable()."'
WHERE driverDeviceID = '".$this->getDriverid()."' " ) ;
            $this->query = mysql_query( $this->getSql() );

            if ( $this->getQuery() ) {
                $this->jsonResponse["available"] = $this->getAvailable() ;
                $this->jsonResponse["success"] = 1;

                // echoing JSON response
                print json_encode($this->jsonResponse);
            }
            else {
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "not successful";
                print json_encode($this->jsonResponse);
            }

        }
        else{
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "not successful";

            // echo no users JSON
            print json_encode($this->jsonResponse);

        }

    }//end function

}//end class



/*
@info
Ανεξάρτηση μέθοδος για αποσφαλμάτωση

@details
Γράφει σε αρχείο τα αποτελέσματα

*/
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

APXEIO showCustomerComments.php

```php
<?php
/*
@info
Κλάση για την εμφάνιση σχολίων από άλλους πελάτες για τον συγκεκριμένο οδηγό


*/

//header('Content-Type: text/html; charset=utf-8');
header('Content-type=application/json; charset=utf-8');
require_once __DIR__ . '/db_connect.php';
$database = new DB_CONNECT();


$showCom = new ShowCustomerComments();
$showCom->showComments(  $_REQUEST["driverDevID"]  ) ;


//=============================================================================================
class ShowCustomerComments
{
    private $jsonResponse;

    function __construct(){

        $this->jsonResponse = array();
    }
//=============================================================================================
    public function showComments( $deviceID ){

        if( isset($deviceID) ){


            $sql ="SELECT com.comment,c.name,c.sirname
                   FROM comments AS com INNER JOIN orders AS o
                   ON(o.orderID =com.orderid) INNER JOIN taxidriver AS td
                   ON (o.taxidriverFK = td.driverID) INNER JOIN customer AS c
                   ON (o.customerFK = c.customerID)
                WHERE td.driverDeviceID = '".$deviceID."'";

            mysql_query("SET NAMES UTF8");
            $result = mysql_query($sql);


            if (mysql_num_rows($result) > 0 ){

                $this->jsonResponse["usercomments"] = array();
                $users = array();
                    while ($row = mysql_fetch_array($result)){
                        $users["comment"] = $row["comment"];
                        $users["fullname"] = $row["name"]." ".$row["sirname"];

                        array_push($this->jsonResponse["usercomments"],$users );
                    }

                $this->jsonResponse["success"] = 1;

                print json_encode($this->jsonResponse);
            }
            else {
                $this->jsonResponse["success"] = 0;
                $this->jsonResponse["message"] = "not successful";
                print json_encode($this->jsonResponse);
            }

        }
        else{
            $this->jsonResponse["success"] = 0;
            $this->jsonResponse["message"] = "not successful";
```

```
            // echo no users JSON
            print json_encode($this->jsonResponse);
        }

    }//end function

}//end class


function debug($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

Υποκατάλογος taxiAnytime_server\mqttClient

APXEIO config.php

```php
<?php
/*
@info
Αρχείο για δήλωση μεταβλητών σύνδεσης του broker

*/
$SAM_MQTTCONF = array('SAM_HOST' => '192.168.2.100',
                      'SAM_PORT' => 1883);



?>
```

APXEIO notifyCustomerForArrival.php

```php
<?php
/*
@info
Αρχείο για την ειδοποίηση των πελατών ότι ο οδηγός που επέλεξαν έχει φτάσει

*/
header('Content-Type: text/html; charset=utf-8');
require('SAM/php_sam.php');
require('config.php');

//create a new connection object
$conn = new SAMConnection();

$conn->connect(SAM_MQTT,$SAM_MQTTCONF);

require_once '../scripts/db_connect.php';
$database = new DB_CONNECT();


if( isset ( $_REQUEST["orderid"] )  ){
    $orderID = $_REQUEST["orderid"];

     mysql_query("SET NAMES UTF8");
            $sql = "SELECT c.customerDeviceID
            FROM orders o INNER JOIN customer c ON c.customerID = o.customerFK
            WHERE orderID ='".$orderID."' " ;
            $query = mysql_query($sql);

    if (mysql_num_rows($query) > 0)
    {
        $row = mysql_fetch_array($query);
        $target = $row["customerDeviceID"];
        $message ="Ο οδηγός έχει φτάσει!!";
        $conn->send('topic://'.$target,$message);
    }
    $conn->disconnect();

}

//for debug

function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}
```

```
?>
```

APXEIO notifyDriverForAccept.php

```
<?php

/*
@info
Αρχείο για την ειδοποίηση των οδηγών ότι τον επέλεξε ο πελάτης και είναι έτοιμος για
παραλαβή

*/

header('Content-Type: text/html; charset=utf-8');
require('SAM/php_sam.php');
require('config.php');

$conn = new SAMConnection();

//type:1 = notifyDrivers
//type:2 = message for acception

if( isset($_REQUEST["customerlat"]) && isset($_REQUEST["customerlon"]) &&
isset($_REQUEST["driverDevIdtoSend"]) && isset ($_REQUEST["selectedOrderid"]) ){

    $message = "2".":"."Πελάτης για
παραλαβή".":".$_REQUEST["customerlat"].":".$_REQUEST["customerlon"].":".$_REQUEST["driverDev
IdtoSend"].":".$_REQUEST["selectedOrderid"] ;
    $target = $_REQUEST["driverDevIdtoSend"];
}
else{
    $message = "";
    $target = "";
}


$conn->connect(SAM_MQTT,$SAM_MQTTCONF);

$conn->send('topic://'.$target,$message);
$conn->disconnect();



//for debug

function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}


?>
```

ΑΡΧΕΙΟ notifyDrivers.php

```php
<?php
/*
@info
Αρχείο για την ειδοποίηση των οδηγών

*/

header('Content-Type: text/html; charset=utf-8');
require('SAM/php_sam.php');
require('config.php');

//create a new connection object
$conn = new SAMConnection();

require_once '../scripts/db_connect.php';
$database = new DB_CONNECT();

//type:1 = notifyDrivers

$message = "1".":"."Νέος
Πελάτης".":".$_REQUEST["lat"].":".$_REQUEST["lng"].":".$_REQUEST["customerid"];

//καθαρισμός για παλαιότερες μη ολοκληρωμένες παραγγελίες
                    $devid = $_REQUEST["customerid"];
                    mysql_query ("DELETE FROM orders WHERE
                                        customerFK =(SELECT customerID FROM customer WHERE
customerDeviceID = '".$devid."' LIMIT 1)
                                        AND orderState <>3") ;



//$target = "/357777961700933";
$conn->connect(SAM_MQTT,$SAM_MQTTCONF);


foreach ($_REQUEST as $key => $target){
        if($key != "lat" && $key !="lng" && $key !="customerid"){
            $conn->send('topic://'.$target,$message);
        }

}

$conn->disconnect();



//for debug
function write2file($string)
{
    $file = fopen("test.txt","a");
    fwrite($file,$string);
    fclose($file);

}



?>
```