Κεφάλαιο 1

Στόχος και σκοπός της εργασίας

Η ανάγκη υποστήριξης επαγγελματιών αγροτών από σύγχρονα τεχνολογικά μέσα μας οδήγησε στην υλοποίηση αυτής της εφαρμογής. Υπό την επίβλεψη του κ. Θεόδωρου Λάντζου προσπαθήσαμε να δημιουργήσουμε μία εφαρμογή που θα υποστηρίζει αυτές τις σύγχρονες ανάγκες των επαγγελματιών αγροτών. Μπορεί να αποτελέσει μία πολύ έξυπνη λύση για κάθε επαγγελματία αγρότη που σήμερα έχει την ανάγκη υποστήριξης των εργασιών του από την τεχνολογία. Μέχρι τώρα οι αγρότες συνήθιζαν να γράφουν τις εργασίες τους σε τετράδια ή άλλα χαρτιά. Σήμερα οι εργασίες που λαμβάνουν μέρος σε ένα αγρόκτημα είναι αρκετές και σημαντικές. Έτσι, ο κάθε αγρότης δεν μπορεί να θυμάται κάθε φορά τί εργασία έχει επιτελέσει κατά τη διάρκεια της ημέρας.

Είναι η ιδανική εφαρμογή για όσους επαγγελματίες αγρότες θέλουν να έχουν μία άμεση, γρήγορη και εύκολη διαχείριση των λογαριασμών και των υπηρεσιών τους. Πρόκειται για ένα λογισμικό τελευταίας τεχνολογίας για κινητή τηλεφωνία, το οποίο έρχεται να οργανώσει και να υποστηρίξει το έργο της αγροτικής παραγωγής. Με την εφαρμογή αυτή κάθε αγρότης που παρέχει υπηρεσίες σε κτήματα άλλων θα μπορεί να έχει ολοκληρωμένη διαχείριση των εργασιών που έγιναν, να βλέπει αυτόματα το μέγεθος της εργασίας, το λογαριασμό των πελατών, να επισκοπεί οπτικά τα χωράφια που εργάστηκε, την ημερομηνία και τις λεπτομέρειες της εργασίας που πραγματοποιήθηκε. Έτσι, για παράδειγμα, κάποιος που παρέχει υπηρεσίες θερισμού καταλήγει συχνά να έχει θερίσει 180 στρμ. και να αμείβεται για 150 στρμ. διότι ο παραγωγός κρύβει έκταση. Ο επαγγελματίας, μην γνωρίζοντας τα χωράφια και μην έχοντας τη δυνατότητα να τα μετρήσει με ακρίβεια, αναγκάζεται να αποδέχεται τα λεγόμενα του παραγωγού και να βγαίνει χαμένος.

Με τον επαγγελματία αγρότη όχι μόνο έχουμε αναλυτικά τις εργασίες που έλαβαν μέρος και τη θέση τους στο χάρτη, αλλά ορίζοντας με κλικ στην οθόνη του κινητού τα σύνορα του αγροτεμαχίου βρίσκουμε την έκταση του χωραφιού με απόκλιση 1%-3% ανάλογα με την ακρίβεια του GPS στο κινητό. Ο χρόνος υπολογισμού της έκτασης του χωραφιού είναι 1,5 λεπτό.

Μπορεί να κρατάει δεδομένα εργασιών προς τρίτους. Έτσι, όταν ένας επαγγελματίας σπορέας, θεριστής που εργάζεται σε κτήματα άλλων γεωργών (σε πλήθος 50 και άνω) σε μια εποχή (50 ημερών) μπορεί να καταγράφει απευθείας το είδος της εργασίας και το χωράφι που έλαβε μέρος. Έτσι διαχείριση λογαριασμών και διατήρηση βιβλίων γίνεται παιχνίδι απευθείας από το μηχάνημα επάνω. Η εφαρμογή συντηρεί αρχείο παρελθόντων ετών.

Οι κυριότερες υπηρεσίες του Επαγγελματία αγρότη είναι:

- Παραμετροποίηση σε κάθε είδος παροχής εργασίας.
- Κοστολόγηση της εργασίας.
- Πελατολόγιο με πλήρη στοιχεία ακόμη για έκδοση τιμολογίου.
- Ανάθεση εργασίας σε πελάτη απευθείας πάνω από το χάρτη.
- Προβολή εργασιών σε κτήματα αγρότη και συνολικών εργασιών
- Οπτική προβολή των εργασιών που έλαβαν μέρος σε κτήματα τρίτων επάνω στο χάρτη
- Εμφάνιση και διατήρηση λογαριασμού ανά εργασία και συνολικά
- Προγραμματισμός εργασιών
- Παροχή απόδειξης υπηρεσιών
- Υπολογισμός εμβαδού για καταμέτρηση έκτασης σε οποιοδήποτε σχήμα χωραφιού
- Διατήρηση αρχείου περασμένων ετών

Πλατφόρμα ανάπτυξης εφαρμογής

Η εφαρμογή γράφτηκε για συσκευές Android όπου είναι το ποιο γνωστό και ραγδαία εξελίξιμο οικοσύστημα στον κόσμο.

Καθημερινά ενεργοποιούνται σχεδόν 1 εκατομμύριο συσκευές Android που το καθιστά το πιο περιζήτητο λειτουργικό στον κόσμο για ανάπτυξη εφαρμογών.

Είναι τόσο διαδεδομένο που με τη δημιουργία κάθε νέας εφαρμογής, αυτή γίνεται άμεσα ορατή σε πολλά εκατομμύρια χρήστες και εύκολα διαθέσιμη σε αυτούς.

Σήμερα το λογισμικό Android είναι εγκατεστημένο σε πάρα πολλές συσκευές ανεξαρτήτου κόστους. Αυτό είναι που βοηθάει όλους τους χρήστες να αποκτήσουν μία εφαρμογή χωρίς να σπαταλούν υπέρογκα ποσά στην αγορά μίας πολύ ακριβής συσκευής.

Καθημερινά γράφονται χιλιάδες εφαρμογές πάνω σε αυτό το λογισμικό που προσφέρουν χαρά και διασκέδαση σε πάρα πολλούς χρήστες, αλλά και γνώσεις σε αυτούς που δεν είναι εξοικειωμένοι με το αντικείμενο.

Το τι είναι το Android και με ποιο τρόπο λειτουργεί αναφέρεται λεπτομερώς στην βιβλιογραφία 2,3.

Κεφάλαιο 2

Άλλες εργασίες πάνω σε αυτό το αντικείμενο

Ο Επαγγελματίας αγρότης αποτελεί καινοτόμα εφαρμογή για τους Έλληνες αγρότες που θέλουν να χρησιμοποιήσουν το κινητό Android τους θέλοντας να έχουν μία άμεση, γρήγορη και εύκολη διαχείριση των λογαριασμών και των υπηρεσιών που προσφέρουν στους πελάτες τους. Δεν έχει βρεθεί κάποια αντίστοιχη εργασία για τέτοιου είδους εφαρμογή που να εξυπηρετεί τις παραπάνω ανάγκες για λογισμικό Android.

Τεχνολογικά εργαλεία που χρησιμοποιήθηκαν για τη δημιουργία αυτής της εφαρμογής

Τα εργαλεία που θα χρησιμοποιήσουμε.

- Java JDK (Java Development Kit)
- Android SDK (Standard Development Kit)
- Eclipse IDE

Οι εφαρμογές android γράφονται σε γλώσσα προγραμματισμού Java, καθώς και το Android SDK είναι γραμμένο σε αυτήν. Μπορείτε να χρησιμοποιήσετε οποιονδήποτε Text Editor για να γράψετε την εφαρμογή σας. Όμως εμείς θα αναφερθούμε στην Eclipse όπου προτείνει και η Google.

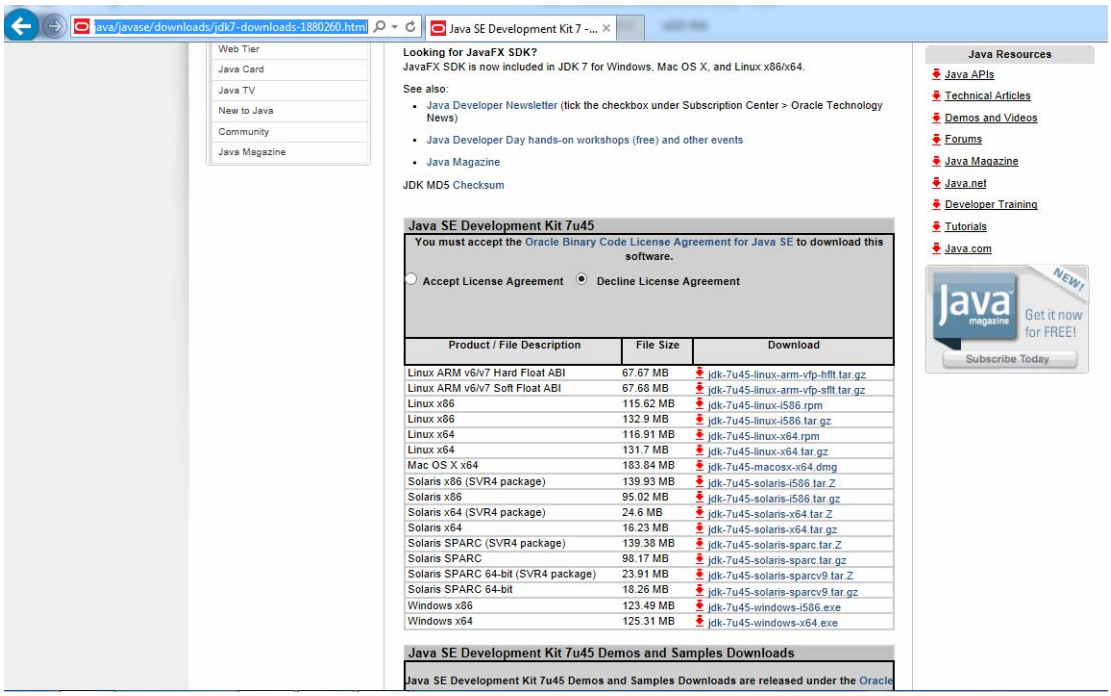
Χρειάζεται να είναι εγκαταστημένο το περιβάλλον Java JDK (Java Development Kit) όπου περιέχει όλες τις βιβλιοθήκες και τα εργαλεία (Compiler , keytool κτλ.) και μπορείτε να το κατεβάσετε από την ιστοσελίδα της Oracle.

Εγκατάσταση του JDK

Από την σελίδα της Oracle

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html

Κατεβάζετε το JDK αφού πρώτα επιλέξετε ότι συμφωνείτε με τους όρους χρήσης της Oracle, επιλέξετε το λειτουργικό σύστημα που έχετε και κατεβάστε την αντίστοιχη έκδοση. Βλέπε την παρακάτω εικόνα.



Σχέδιο 2.1 (Download center της Oracle για το JDK)

Αφού τελειώσετε την εγκατάσταση του JDK πρέπει να εγκαταστήσετε το περιβάλλον του Android (Android SDK).

Το Android SDK (Standard Development Kit) είναι όλες οι βιβλιοθήκες καιτα εργαλεία που χρειάζονται (adb, emulator κτλ.) για να δημιουργήσουμε ,τρέξουμε, δοκιμάσουμε και εκδώσουμε μια εφαρμογή.

Γίνεται λεπτομερής αναφορά για το τι είναι το Android SDK βλέπε βιβλιογραφία 3.

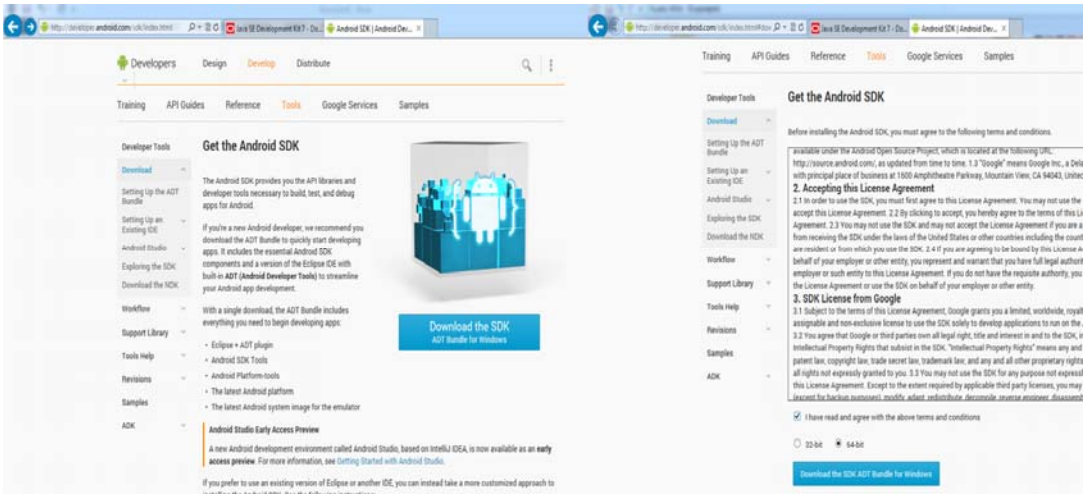
Εγκατάσταση του Android SDK.

Υπάρχουν δύο τρόποι για την εγκατάσταση του SDK.

Ο πρώτος είναι να κατεβάσουμε έτοιμο το bundle του SDK μαζί με τον Java Editor της Eclipse IDE από την ιστοσελίδα του Android.

http://developer.android.com/sdk/index.html

Πατάμε στο *Download the SDK* (βλέπε εικόνα 2.2) και αφού συμφωνήσετε με τους όρους της Google και επιλέξετε την έκδοση Windows που έχετε θα κατεβάσετε ένα zip αρχείο που θα περιέχει το SDK και το Eclipse IDE.

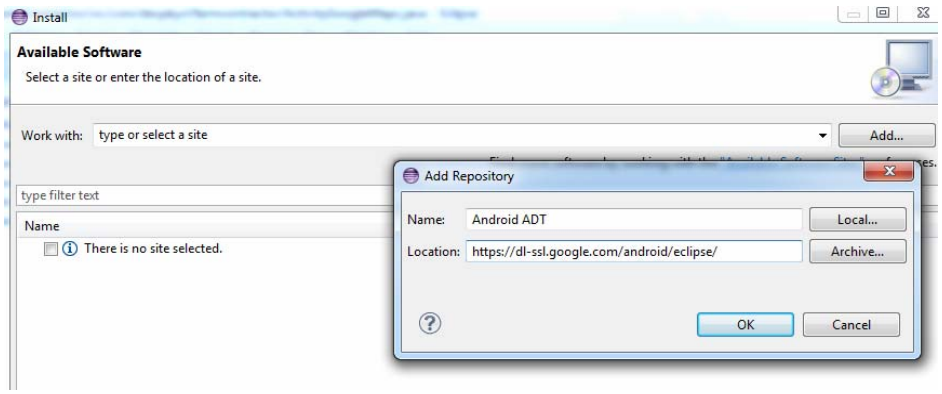


Εικόνα 2.2 (Σελίδα που κατεβάζετε το Android SDK μαζί με το Eclipse IDE)

Ο δεύτερος τρόπος είναι αν έχετε ήδη εγκαταστημένο το Eclipse IDE να περάσετε το ADT (Android Development tool) plugin ώστε να μπορείτε να κατεβάσετε το Android SDK και να το εγκαταστήσετε στην Eclipse .

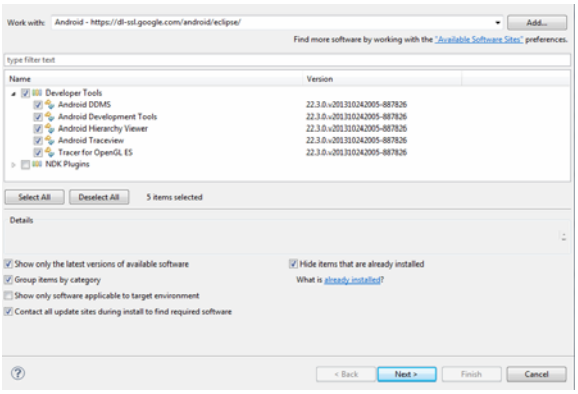
1) Ξεκινήστε την Eclipse και πηγαίνετε Help->Install New Software
2) Πατήστε ADD στο δεξί πάνω μέρος της οθόνης
3) Στο Add repository παράθυρο βάλτε στο Name “Android ADT” και στο Location τον παρακάτω σύνδεσμο.

   https://dl-ssl.google.com/android/eclipse/



Εικόνα 2.3 (Εγκατάσταση του ADT στην Eclipse)

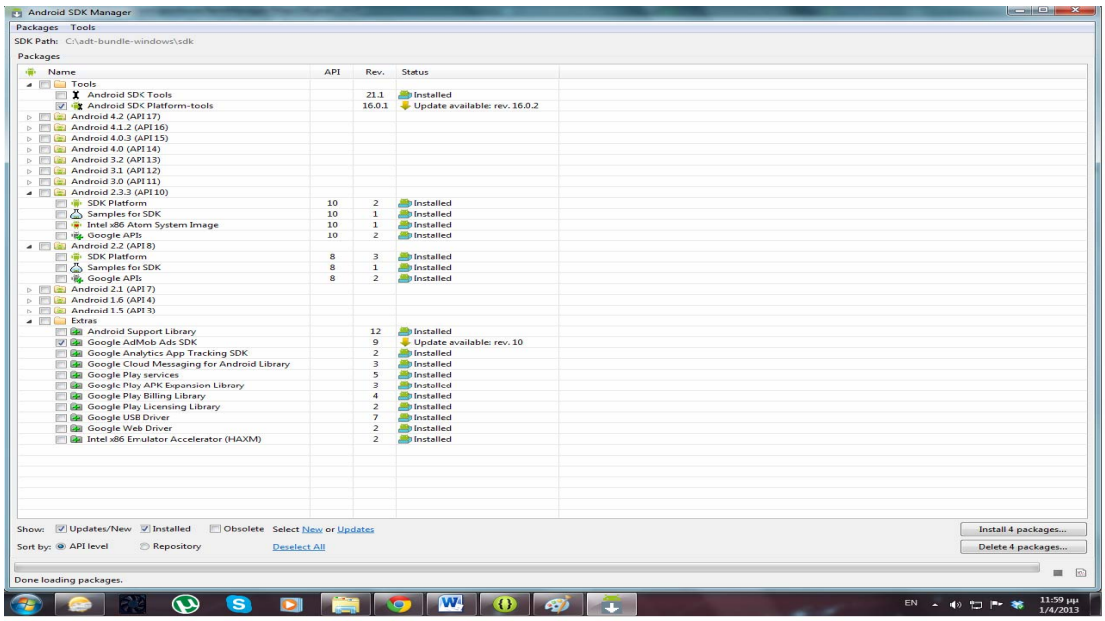
4) Πατάμε οκ και περιμένουμε να εμφανιστή μία λίστα με τα Development Tools



Εικόνα 2.4 (Εγκατάσταση των Development tools)

5) Τα επιλέγουμε και πατάμε επόμενο
6) Περιμένουμε να γίνει η εγκατάσταση και αφού τελειώσει θα μας ζητήσει να κάνει επανεκκίνηση η Eclipse

Μόλις εγκατασταθεί ανοίγουμε το eclipse και πηγαίνουμε στο window→Android Sdk Manager. Εκεί ανοίγει ένα παράθυρο παρόμοιο με την παρακάτω εικόνα.



Εικόνα 2.5 (Android SDK Manager)

Από εδώ επιλέγουμε ποια έκδοση του android θέλουμε να κατεβάσουμε για να αναπτύξουμε την εφαρμογή μας. ΠΡΟΣΟΧΗ: *Κάθε έκδοση είναι παρόμοια με την προηγούμενη, με προσθετά χαρακτηριστικά και βελτιώσεις, οπότε προσοχή με ποια θα ξεκινήσετε.* Εκτός από την έκδοση του Android που θέλουμε επιλέγουμε για

εγκατάσταση τον φάκελο Tools και Extras. Αφού επιλέξουμε όλα αυτά που θέλουμε πατάμε Install x packages, κάνουμε αποδοχή των αδειών χρήσης και περιμένουμε να τελειώσει η εγκατάσταση τους. Η εφαρμογή Farm Contractor έχει αναπτυχθεί για Android 4.1.2 και πάνω οπότε το ελάχιστο που πρέπει να έχουμε είναι η έκδοση 4.1.2. Ο χρόνος εγκατάστασης διαφέρει ανάλογα με το πόσα πακέτα έχουμε επιλέξει να εγκατασταθούν και τον φόρτο του server εκείνη την ώρα. Αφού τελειώσει η εγκατάσταση κλείνουμε τον SDK Manager και κάνουμε επανεκκίνηση το Eclipse.

Δημιουργία Virtual Machine

Η εικονική μηχανή είναι απαραίτητη, εάν δεν διαθέτουμε κάποια συσκευή με λειτουργικό Android, ώστε να μπορούμε να τρέχουμε και να δοκιμάζουμε την εφαρμογή μας. Για να δημιουργήσουμε μια νέα εικονική μηχανή ανοίγουμε το Eclipse και πηγαίνουμε στο window→Android Virtual Device Manager. Στο παράθυρο που ανοίγει εμφανίζονται όλες οι εικονικές συσκευές που έχουμε δημιουργήσει. Για να δημιουργήσουμε μια καινούργια επιλέγουμε New. Στο παράθυρο που ανοίγει συμπληρώνουμε το όνομα της εικονικής μηχανής που θέλουμε, το μέγεθος οθόνης που θέλουμε να έχει, την έκδοση Android που θέλουμε (αν θέλουμε να χρησιμοποιήσουμε τους χάρτες της Google τότε πρέπει στο target να επιλέξουμε την έκδοση Android που περιέχει το Google Api). Επίσης επιλέγουμε τον τύπο του επεξεργαστή και συμπληρώνουμε τα υπόλοιπα πεδία ανάλογα με τις ανάγκες μας. Μόλις τελειώσουμε πατάμε OK και η εικονική μηχανή έχει δημιουργηθεί και είναι έτοιμη για να την ανοίξουμε.

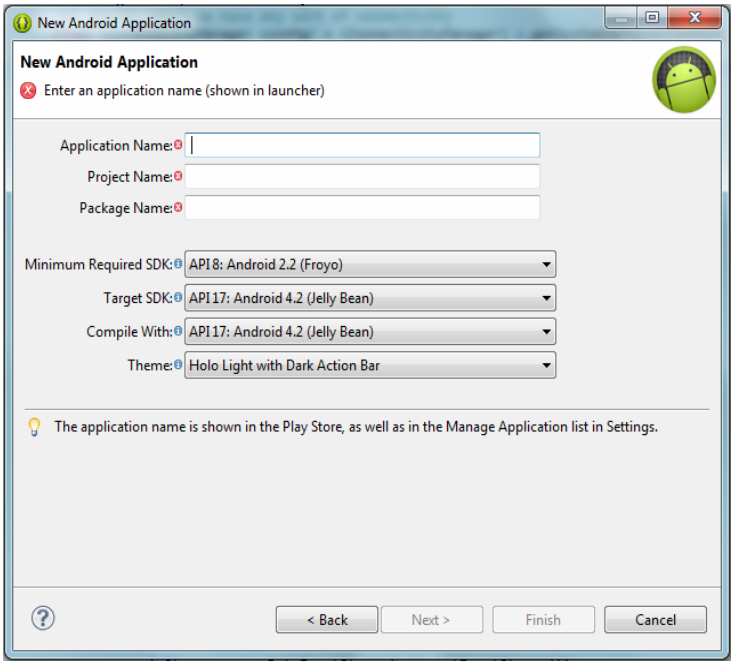
Εκτέλεση εικονικής μηχανής (Virtual Machine)

Για να ανοίξουμε μια εικονική μηχανή ώστε να τρέξουμε μια εφαρμογή που έχουμε κάνει πηγαίνουμε πάλι στο Android Virtual Device Manager, επιλέγουμε την εικονική μηχανή που θέλουμε και πατάμε Start. Περιμένουμε λίγο μέχρι να φορτώσει η εικονική μηχανή και είμαστε έτοιμοι. Πλέον έχουμε ένα εργαλείο που μοιάζει με μια συσκευή Android και μπορούμε να τρέχουμε τις εφαρμογές μας.

Εικόνα 2.6 (Virtual Machine που προσομοιώνει ένα κινητό με έκδοση 2.2)

Δημιουργία νέου Android Project στο Eclipse

Εφόσον έχουμε κάνει όλα τα παραπάνω επιτυχώς είμαστε έτοιμοι για να ξεκινήσουμε την ανάπτυξη εφαρμογών. Αρχικά πηγαίνουμε στο File→New→Other. Στο παραθυράκι που ανοίγει κάνουμε διπλό κλικ στο Android για να ανοίξει η λίστα και επιλέγουμε το Android Application Project. Εκεί ανοίγει ένα νέο παράθυρο στο οποίο θα χρειαστεί να συμπληρώσουμε κάποια στοιχεία για την εφαρμογή μας. Συμπληρώνουμε όνομα εφαρμογής, όνομα project και όνομα πακέτου. Το όνομα πακέτου θα πρέπει να είναι κάτι μοναδικό διότι 2 εφαρμογές δεν μπορούν να χρησιμοποιούν το ίδιο πακέτο. Επίσης συμπληρώνουμε την χαμηλότερη έκδοση Android την οποία θα υποστηρίζει η εφαρμογή μας, την έκδοση Android στην οποία θα την αναπτύξουμε και θα την κάνουμε compile και το θέμα που θα χρησιμοποιήσουμε. Αφού συμπληρώσουμε όλα αυτά πατάμε Next, εκεί επιλέγουμε αν θέλουμε κάτι και πατάμε ξανά Next. Στο επόμενο παράθυρο μπορούμε να ρυθμίσουμε το εικονίδιο της εφαρμογής. Πατάμε Next και στη συνέχεια επιλέγουμε τι τύπου activity θέλουμε να είναι αυτή που θα φτιάξει το eclipse αρχικά. Πατάμε Next, δηλώνουμε το όνομα της κλάσης Activity και τέλος πατάμε Finish και περιμένουμε το eclipse να δημιουργήσει τον φάκελο του project με όλα τα απαραίτητα στοιχεία μέσα.



Εικόνα 2.6 (Νέο android project wizard)

Εισαγωγή ενός έτοιμου project στο Eclipse

Εάν έχουμε ένα έτοιμο project και θέλουμε να το δοκιμάσουμε ή να δούμε τον κώδικά του, ανοίγουμε το eclipse και πηγαίνουμε File→ import. Στο παραθυράκι που ανοίγει επιλέγουμε General→Existing projects into workspace. Εκεί βρίσκουμε το project που θέλουμε και το επιλέγουμε. Επίσης αν θέλουμε επιλέγουμε το "copy project into workspace" αν θέλουμε το project να αντιγραφεί στον φάκελο του eclipse. Πατάμε finish και το project είναι έτοιμο στο eclipse. Για να το τρέξουμε, αν έχουμε κάποια συμβατή συσκευή Android τη συνδέουμε στον υπολογιστή και πατάμε run στον eclipse. Αλλιώς ανοίγουμε έναν emulator από το eclipse και μόλις ανοίξει πατάμε run και περιμένουμε να φορτώσει.

Δημιουργία εκτελέσιμου αρχείου .apk

Αφού έχουμε την εφαρμογή μας έτοιμη και θέλουμε να την δώσουμε και σε άλλους χρήστες να την δοκιμάσουν θα πρέπει να δημιουργήσουμε ένα εκτελέσιμο αρχείο κατάληξης .apk. Τα αρχεία αυτά υποστηρίζονται από το Android και είναι ο τύπος των εφαρμογών που μπορούν να τρέξουν στις συσκευές Android. Για να γίνει αυτό, από το μενού του eclipse πηγαίνουμε στο File→Export. Επιλέγουμε Android→Export Android Application. Στη συνέχεια επιλέγουμε το project που θέλουμε και πατάμε next. Στο επόμενο παραθυράκι μας ζητάει το keystore που θα χρησιμοποιήσουμε. Το keystore είναι στην ουσία η υπογραφή μας για την εφαρμογή και ταυτόχρονα ένα

είδος ασφάλειας. Εάν δεν έχουμε κάποιο έτοιμο επιλέγουμε create new και επιλέγουμε τον φάκελο αποθήκευσης που θέλουμε. Συμπληρώνουμε έναν κωδικό της επιλογής μας και πατάμε next. Στο επόμενο παραθυράκι συμπληρώνουμε τα στοιχεία μας. Στο πεδίο validity(years) πρέπει να βάλουμε μια τιμή μεγαλύτερη από 25. Στη συνέχεια επιλέγουμε την τοποθεσία και το όνομα του αρχείου .apk και πατάμε finish. Το αρχείο .apk της εφαρμογής μας είναι έτοιμο και μπορούμε να δοκιμάσουμε και σε άλλες συσκευές με λειτουργικό Android.

Κεφάλαιο 3

Η σχεδίαση της εφαρμογής

Η εφαρμογή σχεδιάστηκε έτσι ώστε να εξυπηρετεί τις εκδόσεις από 2.2 μέχρι 4.4 (τελευταία έκδοση) του Android χρησιμοποιώντας όλα τα χαρακτηριστικά της κάθε έκδοσης. Με τη βοήθεια της βιβλιοθήκης συμβατότητας της Google καταφέραμε να εντάξουμε πολλά από τα components που περιέχονται σε νεότερες εκδόσεις στην έκδοση 2.2 του Android. Συνολικά ο Επαγγελματίας Αγρότης αποτελείται από 43 κλάσεις εκ των οποίων 27 είναι γραφικές (activities και fragments).

Τα 3 κύρια χαρακτηριστικά της εφαρμογής είναι:

1. Τα frgments που χρησιμοποιήσαμε για να σχεδιάσουμε το UI
2. Η συμβατότητα και η χρήση του ActionBar ανάλογα με την έκδοση του Android.
3. Η σχεδίαση της βάσης καθώς ο FarmContractor είναι ένα πρόγραμμα διεπαφής ανάμεσα στο χρήστη και σε μία βάση δεδομένων.

Δεν θα αναφερθούμε στα κοινά components που χρησιμοποιεί αυτή η εφαρμογή καθώς έχει γίνει λεπτομερής αναφορά στην βιβλιογραφία 3

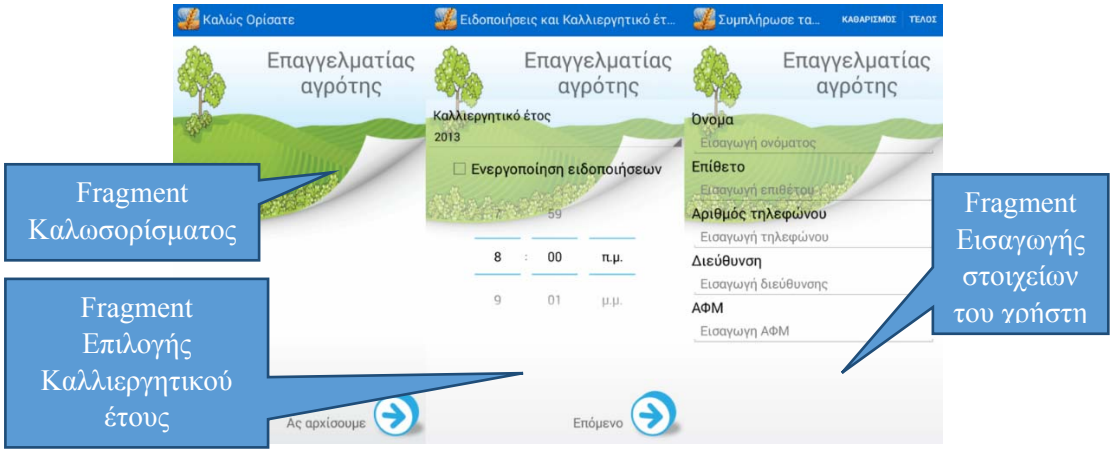
Αυτό που χαρακτηρίζει την σχεδίαση της εφαρμογής είναι η χρήση των fragments που παρουσιάστηκαν για πρώτη φορά στην έκδοση 3.0 του Android.

Τι είναι το fragment;

Ένα fragment αντιπροσωπεύει μια συμπεριφορά ή ένα τμήμα της διεπαφής χρήστη σε μια δραστηριότητα. Μπορούμε να συνδυάσουμε πολλά fragments σε μια ενιαία δραστηριότητα (activity) για την κατασκευή ενός UI πολυ-παραθύρων και να ξαναχρησιμοποιήσουμε ένα fragment σε πολλαπλές δραστηριότητες (activities). Μπορούμε να σκεφτούμε ένα fragment ως ένα αρθρωτό τμήμα μιας δραστηριότητας, η οποία έχει το δικό του κύκλο ζωής, λαμβάνει τις δικά του γεγονότα εισόδου, και το οποίο μπορείτε να προσθέσετε ή να αφαιρέσετε, ενώ η δραστηριότητα εκτελείται (περίπου όπως ένα «υπό activity» που μπορείτε να επαναχρησιμοποιήσετε σε διάφορα activities).

Παράδειγμα χρήσης fragments.

Όταν τρέξει για πρώτη φόρα η εφαρμογή θα εμφανιστεί η διεπαφή χρήστη όπου θα γίνουν οι κατάλληλες παραμετροποιήσεις από τον χρήστη. Αυτή η διεπαφή (Activity) χρησιμοποιεί πολλαπλά fragments.

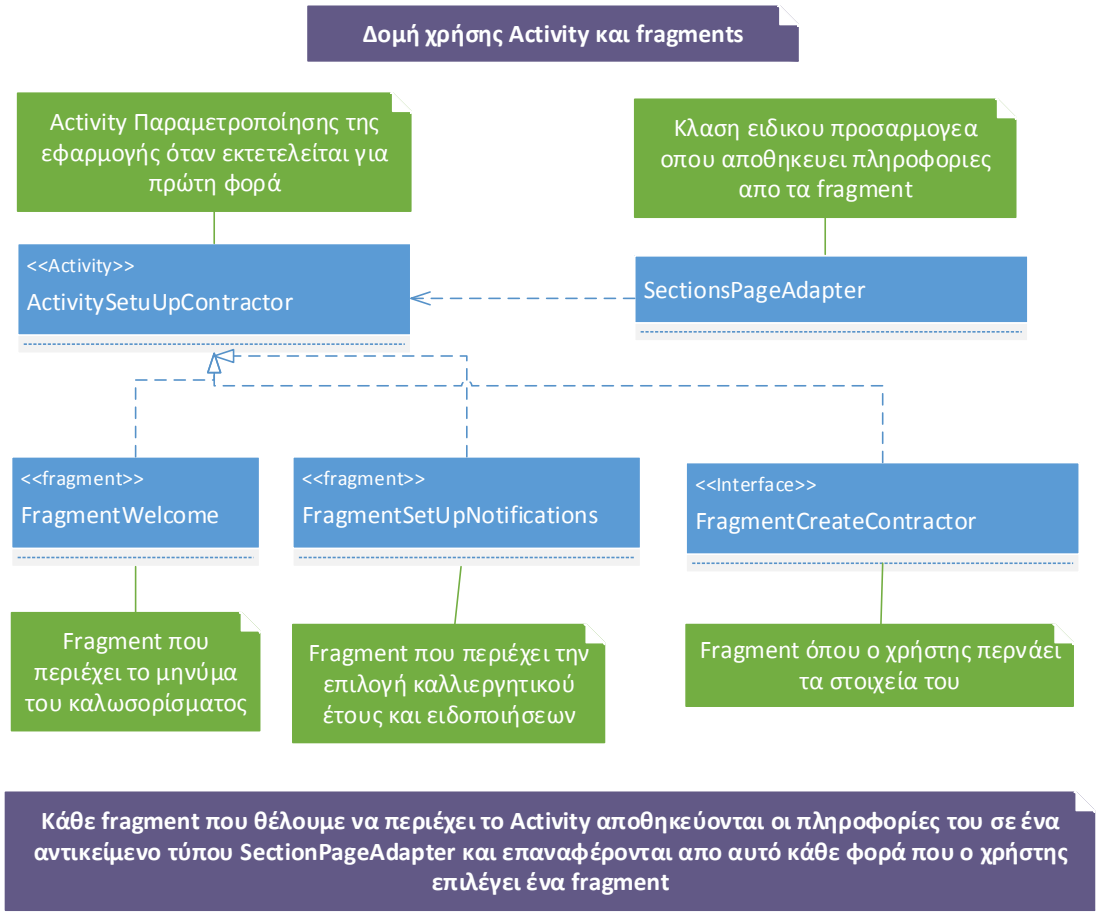


Εικόνα 3.1 (Διεπαφή πρώτης εκτέλεσης της εφαρμογής)

Η οθόνη αυτή [Εικόνα 3.1] εμφανίζεται μόνο την πρώτη φορά που τρέχει η εφαρμογή μετά την εγκατάσταση. Εδώ χρειάζεται να ορίσει το τρέχων καλλιεργητικό έτος και να συμπληρώσει τα στοιχεία του ο χρήστης.

Όπως αναφέραμε και παραπάνω η εφαρμογή μας χρησιμοποιεί fragments για την δημιουργία ενός UI πολύ-παραθύρου.

Στο παραπάνω σχήμα φαίνονται 3 οθόνες όπου η κάθε μια είναι ένα ξεχωριστό fragment και διαχειρίζονται από ένα activity.



Εικόνα 3.2 (Διάγραμμα προβολής χρήσης fragment από Activity)

Παράδειγμα του Activity SetUpContractor

Ο κώδικας σε Java

```
// Η κλάση του Activity μας κληρονομεί το FragmentActivity που βρίσκεται στην βοηθητική κλάση της Google όπου μας επιτρέπει να χρησιμοποιούμε τα fragments μέχρι την έκδοση 2.1

public class ActivitySetUpContractor extends FragmentActivity {

        private SectionsPagerAdapter mSectionsPagerAdapter;
        private ViewPager mViewPager;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_setup_contractor);


        final ImageView btnNextFragment=(ImageView)findViewById(R.id.btnNextWelcomeFragment;
        final CustomTextView
tvWelcomeNextView=(CustomTextView)findViewById(R.id.tvWelcomeMessage);
                 // Set up the ViewPager with the sections adapter.
        mViewPager = (ViewPager) findViewById(R.id.pager);
        // Create the adapter that will return a fragment for each sections
        // of the app.
        mSectionsPagerAdapter = new
SectionsPagerAdapter(getSupportFragmentManager(),this,mViewPager);

        mViewPager.setAdapter(mSectionsPagerAdapter);
```

```
        // Προσθήκη των fragments στο activity μας.

        mSectionsPagerAdapter.addFragment(FragmentWelcome.class, null,
getText(R.string.welcome).toString());
        mSectionsPagerAdapter.addFragment(FragmentSetupNotification.class, null,
getText(R.string.notifications_and_farm_year).toString());
        mSectionsPagerAdapter.addFragment(FragmentCreateContractor.class, null,
getText(R.string.enter_your_information).toString());


}
```

Ο κώδικας του layout σε XML activity_setup_contractor.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/background" >

    <android.support.v4.view.ViewPager
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/pager"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_alignParentBottom="true"
        android:layout_below="@+id/tvWorkProfit"
        tools:context=".ActivitySetUpContractor" >

    <!--
    This title strip will display the currently visible page title, as well as the page
    titles for adjacent pages.
    -->

    <android.support.v4.view.PagerTitleStrip
        android:id="@+id/pager_title_strip"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="top"
        android:paddingBottom="4dp"
        android:paddingTop="4dp"
        android:textColor="#fff"
        android:visibility="gone" />

</android.support.v4.view.ViewPager>

    <ImageView
        android:id="@+id/btnNextWelcomeFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:src="@drawable/next_button" />

    <com.despkyri.farmcontractor.CustomTextView
        android:id="@+id/tvWelcomeMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="18dp"
        android:layout_toLeftOf="@+id/btnNextWelcomeFragment"
        android:text="@string/lets_start"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#90000000" />

    <com.despkyri.farmcontractor.CustomTextView
        android:id="@+id/tvWorkProfit"
        android:layout_width="210dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginRight="15dp"
        android:layout_marginTop="12dp"
```

```
        android:gravity="center"
        android:text="@string/app_name"
        android:textColor="#90000000"
        android:textSize="30sp" />

</RelativeLayout>
```

Παράδειγμα του fragment SetUpNotifications

Ο κώδικας σε Java

```
// Η κλάση του fragment μας κληρονομεί το Fragment που βρίσκεται στην
βοηθητική κλάση της Google όπου μας επιτρέπει να χρησιμοποιούμε τα
fragments μέχρι την έκδοση 2.1



public class FragmentSetupNotification extends Fragment{
        private View mView; // Είναι το view του fragment
        private ContentSettings settings;
        private int spinnerMotionID;
        @Override
        public void onCreate(Bundle savedInstanceState) {
                // TODO Auto-generated method stub
                super.onCreate(savedInstanceState);
                spinnerMotionID=-1;

        }

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
                final DataHelperSettings dataSettings=new DataHelperSettings(getActivity());
                settings = dataSettings.readSettings();
                //To View του fragment ποιο layout προβάλει
                mView=inflater.inflate(R.layout.fragment_set_notification, container,false);

                Spinner spinners=(Spinner)mView.findViewById(R.id.spinnerYears);

                int pos=settings.getFarmingYear()-2013;
                if(pos>=0)
                        spinners.setSelection(pos);
                spinners.setOnTouchListener(new OnTouchListener(){

                        @Override
                        public boolean onTouch(View arg0, MotionEvent arg1) {
                                spinnerMotionID = arg1.getAction();

                                return false;
                        }

                });
                spinners.setOnItemSelectedListener(new OnItemSelectedListener(){

                        @Override
                        public void onItemSelected(AdapterView<?> adapter, View arg1,
                                        int position, long arg3) {

                                try {
                                        String
name=(String)adapter.getItemAtPosition(position);
                                        DataHelperContractor contractor=new
DataHelperContractor(getActivity());
                                        if(contractor.hasContractor())
                                        {

                                                if(spinnerMotionID == MotionEvent.ACTION_UP)

        changeFarmYear(Integer.parseInt(name));
                                                spinnerMotionID=-1;
```

```
                                        }
                                        else
                                        {

        settings.setFarmingYear(Integer.parseInt(name));
                                                dataSettings.saveSettings(settings);
                                        }
                                        spinnerMotionID=-1;
                                }
                                catch(NumberFormatException e) {
                                        spinnerMotionID=-1;
                                }

                        }

                        @Override
                        public void onNothingSelected(AdapterView<?> arg0) {
                                spinnerMotionID=-1;

                        }

                });

                final TimePicker timePicker =
(TimePicker)mView.findViewById(R.id.timePickerNotify);

                timePicker.setCurrentHour(settings.getHours());
                timePicker.setCurrentMinute(settings.getMinutes());
                final CheckBox cb=(CheckBox)mView.findViewById(R.id.cbEnableNotifications);
                cb.setChecked(settings.isNotify());

                cb.setOnCheckedChangeListener(new OnCheckedChangeListener(){

                        @Override
                        public void onCheckedChanged(CompoundButton arg0, boolean checked) {
                                if(checked)
                                {
                                        settings.setHours(timePicker.getCurrentHour());
                                        settings.setMinutes(timePicker.getCurrentMinute());
                                }
                                settings.setNotify(checked);
                                dataSettings.saveSettings(settings);
                                enableNotifications();

                        }

                });


                timePicker.setOnTimeChangedListener(new OnTimeChangedListener(){

                        @Override
                        public void onTimeChanged(TimePicker arg0, int arg1, int arg2) {
                                settings.setNotify(false);
                                cb.setChecked(false);
                                enableNotifications();

                        }

                });


                return mView;
        }
        public void changeFarmYear(int newYear)
        {
                IODatabase ioDatabase=new IODatabase(getActivity());
                if(ioDatabase.backupFileExist(newYear))
                {
                        confirmRestoreYear(newYear);

                }
                else
                {
```

```
                        confirmChangeYear(newYear);
                }
        }
        public void confirmRestoreYear(final int newYear)
        {
                AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());
                builder.setIcon(R.drawable.ic_action_device_access_storage_1);
                builder.setTitle(R.string.restore_farm_year);
                builder.setMessage(R.string.restore_farm_year_message);
                builder.setPositiveButton(R.string.restore_alternative,new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                                IODatabase ioDatabase=new IODatabase(getActivity());
                                ioDatabase.backupDatabase(settings.getFarmingYear());
                                if(ioDatabase.backupDatabase(settings.getFarmingYear()))
                                {
                                        ioDatabase.restoreDatabase(newYear);
                                        Toast.makeText(getActivity(),
R.string.restored_successfully_database, Toast.LENGTH_SHORT).show();
                                        dialog.dismiss();
                                        getActivity().finish();
                                }

                        }
                });
                builder.setNegativeButton(R.string.cancel,new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                                Spinner
spinners=(Spinner)mView.findViewById(R.id.spinnerYears);

                                int pos=settings.getFarmingYear()-2013;
                                if(pos>=0)
                                        spinners.setSelection(pos);
                                dialog.dismiss();

                        }
                });
                builder.setOnCancelListener(new OnCancelListener(){

                        @Override
                        public void onCancel(DialogInterface arg0) {
                                Spinner
spinners=(Spinner)mView.findViewById(R.id.spinnerYears);

                                int pos=settings.getFarmingYear()-2013;
                                if(pos>=0)
                                        spinners.setSelection(pos);

                        }

                });
                AlertDialog dialog=builder.create();
                dialog.show();


        }
        public void confirmChangeYear(final int newYear)
        {
                AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());
                builder.setIcon(R.drawable.ic_action_device_access_storage_1);
                builder.setTitle(R.string.farm_year);
                builder.setMessage(R.string.create_new_year);
                builder.setPositiveButton(R.string.create,new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                                IODatabase io=new IODatabase(getActivity());
                                if(io.backupDatabase(settings.getFarmingYear()))
```

```
                                {
                                        settings.setFarmingYear(newYear);
                                        DataHelperSettings dataSettings=new
DataHelperSettings(getActivity());
                                        dataSettings.saveSettings(settings);
                                        OpenHelperDatabase db=new
OpenHelperDatabase(getActivity());
                                        db.wipeDatabase();
                                        dialog.dismiss();
                                        getActivity().finish();

                                }

                        }
                });
                builder.setOnCancelListener(new OnCancelListener(){

                        @Override
                        public void onCancel(DialogInterface arg0) {
                                Spinner
spinners=(Spinner)mView.findViewById(R.id.spinnerYears);

                                int pos=settings.getFarmingYear()-2013;
                                if(pos>=0)
                                        spinners.setSelection(pos);

                        }

                });
                builder.setNegativeButton(R.string.cancel,new
DialogInterface.OnClickListener() {

                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                                Spinner
spinners=(Spinner)mView.findViewById(R.id.spinnerYears);

                                int pos=settings.getFarmingYear()-2013;
                                if(pos>=0)
                                        spinners.setSelection(pos);
                                dialog.dismiss();

                        }
                });
                AlertDialog dialog=builder.create();
                dialog.show();

        }
        public void enableNotifications()
        {

        Intent intent = new Intent(getActivity(), NotificationService.class);
            PendingIntent pintent = PendingIntent.getService(getActivity(), 0, intent, 0);
            AlarmManager alarm =
(AlarmManager)getActivity().getSystemService(Context.ALARM_SERVICE);


            if(settings.isNotify())
            {
                Calendar cal = Calendar.getInstance();

                cal.set(Calendar.HOUR_OF_DAY, settings.getHours());
                cal.set(Calendar.MINUTE, settings.getMinutes());
                cal.set(Calendar.SECOND, 00);
                alarm.setRepeating(AlarmManager.RTC_WAKEUP,
cal.getTimeInMillis(),24*60*60*1000, pintent);
            }
            else
                alarm.cancel(pintent);
        }
```

}

Ο κώδικας του layout σε XML fragment_set_notifications.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/scrollView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/white_transp" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <com.despkyri.farmcontractor.CustomTextView
            android:id="@+id/customTextView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="10dp"
            android:text="@string/choose_year"
            android:textColor="@color/black"
            android:textSize="18sp" />

        <Spinner
            android:id="@+id/spinnerYears"
            android:layout_width="match_parent"
            android:layout_height="0dip"
            android:layout_weight="1"
            android:entries="@array/years"
            android:prompt="@string/choose_year" />

        <CheckBox
            android:id="@+id/cbEnableNotifications"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:layout_marginTop="10dp"
            android:text="@string/enable_notifications"
            android:textColor="@color/black"
            android:textSize="20sp" />

        <TimePicker
            android:id="@+id/timePickerNotify"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:scrollbarStyle="insideInset" />
    </LinearLayout>

</ScrollView>
```

ActionBar

Ένα χαρακτηριστικό που χρησιμοποιεί η εφαρμογή για εκδόσεις λογισμικού πάνω από 3.0 είναι το ActionBar.

Τι είναι το ActionBar;

Το ActionBar είναι ένα χαρακτηριστικό παράθυρο που προσδιορίζει τη θέση του χρήστη, και του παρέχει ενέργειες και τρόπους πλοήγησης. Το ActionBar προσφέρει στους χρήστες ένα οικείο περιβάλλον σε όλες τις εφαρμογές που το σύστημα προσαρμόζεται ανάλογα με τις διαφορετικές διαμορφώσεις της οθόνης.

Εικόνα 3.3 (Η μπάρα ActionBar)

Το ActionBar παρέχει αρκετές βασικές λειτουργίες:

- Παρέχει ένα ειδικό χώρο για να δώσει στην εφαρμογή μια ταυτότητα και υποδείξει τη θέση του χρήστη στην εφαρμογή.
- Κάνει σημαντικές ενέργειες εμφανέστατες και προσβάσιμες με ένα προβλέψιμο τρόπο (όπως Αναζήτηση).
- Υποστηρίζει την πλοήγηση και προβάλλει τις μεταγωγές μέσα στις εφαρμογές (με καρτέλες ή drop-down λίστες).

Το ActionBar το χρησιμοποιούμε σαν μενού για να προβάλλουμε τις σημαντικές ενέργειες που κάνει ο χρήστης όπως για παράδειγμα προσθήκη, διαγραφή, αναζήτηση κτλ.

Μέχρι τη στιγμή που ολοκληρώθηκε η εφαρμογή το ActionBar δεν υποστηρίζονταν από τη βιβλιοθήκη συμβατότητας της Google. Γι' αυτό έπρεπε να βρούμε έναν άλλο τρόπο για να προβάλλουμε τις ενέργειες του χρήστη (Μενού), όπου αυτά αποτελούνταν από buttons και dialogs. Πρόσφατα η Google κυκλοφόρησε μία νέα αναβάθμιση της βιβλιοθήκης συμβατότητας όπου μας επιτρέπει να χρησιμοποιούμε το ActionBar μέχρι την έκδοση 2.1. Παρόλα αυτά εμείς χρησιμοποιούμε την παραπάνω λύση που αναφέρθηκε.

Ο κώδικας που μας επιτρέπει να βλέπουμε ποια έκδοση τρέχει η εφαρμογή μας και να προσαρμόζεται το UI ανάλογα με την έκδοση:

android.os.Build.VERSION.SDK_INT

όπου SDK_INT είναι η έκδοση του Android.

Όταν δημιουργούμε ένα μενού αυτόματα μπορούμε να εμφανίσουμε τις επιλογές του στο ActionBar αν επιλέξουμε SHOW_AS_ACTION

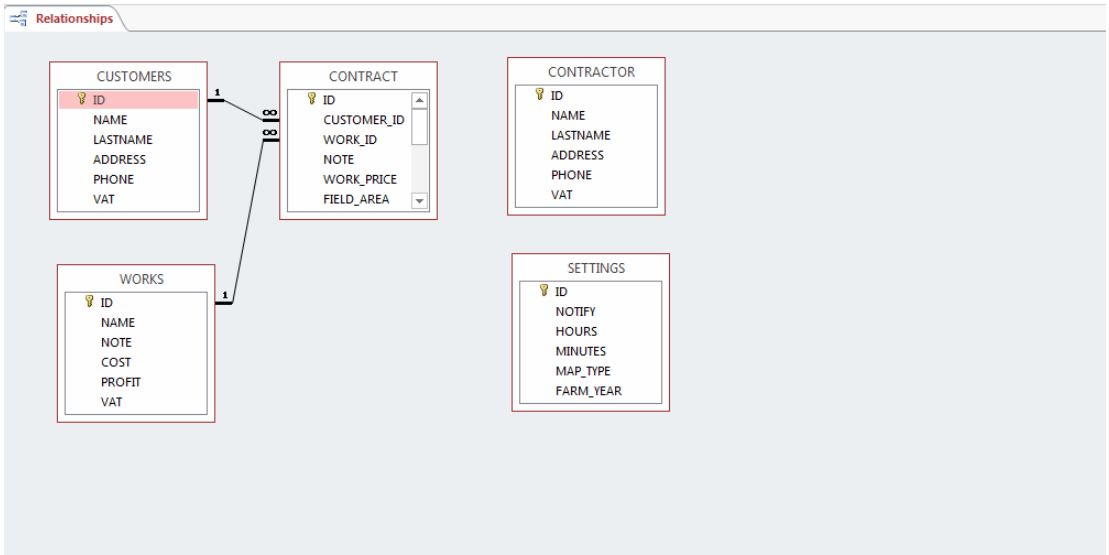
```
android:showAsAction="ifRoom|withText
```

Παράδειγμα ενός μενού:

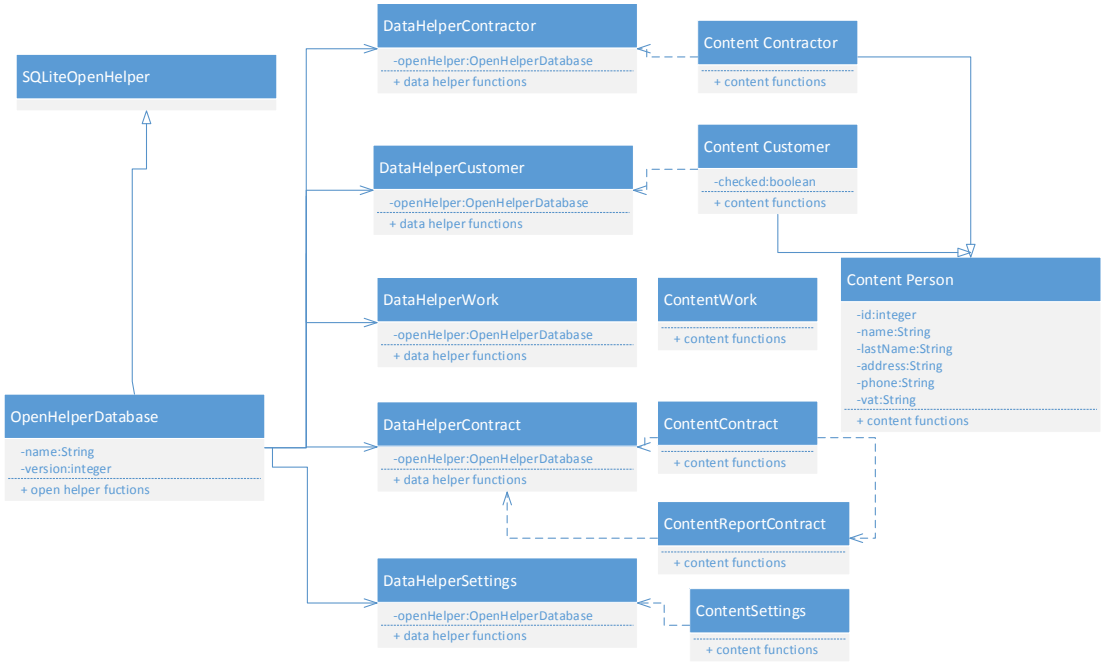
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/menu_map_options"
android:showAsAction="ifRoom|withText"
android:title="@string/action_settings">
        <menu>
            <item android:id="@+id/menu_map_satellite"
android:title="@string/satellite" android:showAsAction="ifRoom|withText"
android:checkable="true" android:orderInCategory="0"/>
        </menu>
    </item>


</menu>
```

SQLite βάση δεδομένων

Η SQLite βάση δεδομένων είναι αυτή που υποστηρίζει το Android για αποθήκευση δεδομένων. Είναι μια lite μορφή της SQL πράγμα που την κάνει ιδανική για χρήση σε φορητές συσκευές. Για τη χρησιμοποίησή της σε κάποια εφαρμογή δεν απαιτείται να κατεβάσουμε κάτι. Απλά την δημιουργούμε και την διαχειριζόμαστε κατευθείαν με κώδικα, καθώς υπάρχει ενσωματωμένη στις βιβλιοθήκες του Android.



Εικόνα 3.4 (Διάγραμμα E-R της βάσης του Farm Contractor)



Εικόνα 3.4 (Διάγραμμα κλάσεων που δείχνει την δομή της βάσης του Farm Contractor)

Παράδειγμα των κλάσεων που χρησιμοποιούνται για την προσπέλαση των δεδομένων ενός πίνακα της βάσης.

OpenHelperDatabase Είναι η κλάση όπου δημιουργεί, αναβαθμίζει και επιστρέφει σύνδεση με βάση.

Ο κώδικας.

```
/**
 * Δημιουργει και επιστρεφει συνδεση με την βαση
 * @author <b>Despotis/Kyriazopoulos</b>
 *
 */
public class OpenHelperDatabase extends SQLiteOpenHelper {
	private static String name="farmcontractor.db"; //The name of database
	private static int version=1; //Version of database

	//Constructor
	public OpenHelperDatabase(Context context) {
		super(context, name, null, version);
		// TODO Auto-generated constructor stub
	}


	@Override
	public void onCreate(SQLiteDatabase db) {
		//Run create database queries
		//db.g
		//this.getWritableDatabase().
		createDatabase(db);

	}
```

```
	@Override
	public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
		//onUpgrage triggers when the version of  database change

	}

	/**
	 * Περιεχει ολα τα ερωτηματα που δημιουργουν τους πινακες
	 * @param db
	 */
	public void createDatabase(SQLiteDatabase db)
	{
		try
		{
			String query="create table CUSTOMERS (ID integer primary key
autoincrement,NAME COLLATE NOCASE,LASTNAME COLLATE NOCASE,ADDRESS COLLATE NOCASE, PHONE
COLLATE NOCASE,VAT COLLATE NOCASE)";
			db.execSQL(query);

			query="create table CONTRACTOR (ID integer primary key
autoincrement,NAME COLLATE NOCASE,LASTNAME COLLATE NOCASE,ADDRESS COLLATE NOCASE, PHONE
COLLATE NOCASE,VAT COLLATE NOCASE)";
			db.execSQL(query);

			query="create table WORKS (ID integer primary key autoincrement,NAME
COLLATE NOCASE,NOTE COLLATE NOCASE,COST real,PROFIT real,VAT real)";
			db.execSQL(query);

			query="create table CONTRACT (ID integer primary key
autoincrement,CUSTOMER_ID integer,WORK_ID integer,NOTE COLLATE NOCASE,WORK_PRICE
real,FIELD_AREA real,DISCOUNT real,COMPLETED integer,NOTIFY_ME integer,YEAR integer,MONTH
integer,DAY integer,LATITUDE real,LONGITUDE real)";
			db.execSQL(query);

			query="create table SETTINGS (ID integer primary key ,NOTIFY
integer,HOURS integer,MINUTES integer,MAP_TYPE integer,FARM_YEAR integer)";
			db.execSQL(query);

			query="Insert into SETTINGS
(ID,NOTIFY,HOURS,MINUTES,MAP_TYPE,FARM_YEAR) VALUES(0,0,8,0,4,2013)";
			db.execSQL(query);
		}
		catch(SQLiteException e)
		{
			Log.e("DatabaseOpenHelper::createDatabase", e.getMessage());
		}


	}

	public void wipeDatabase()
	{
		SQLiteDatabase db=this.getWritableDatabase();
		String query="delete from customers";
		db.execSQL(query);
		query="delete from works";
		db.execSQL(query);
		query="delete from contract";
		db.execSQL(query);
		db.close();


	}


}
```

Για κάθε πίνακα που περιέχει η βάση δημιουργούμε και μια αντίστοιχη κλάση για την προσθήκη, εξαγωγή , αναβάθμιση και διαγραφή των δεδομένων του πίνακα. Τα δεδομένα επιστρέφονται σε ένα Content Provider για κάθε γραμμή του πίνακα. Όταν

θέλουμε να επιστρέψουμε αρκετές γραμμές του πίνακα επιστρέφουμε μια λίστα από Contents για κάθε πίνακα.

Κλάση ContentContractor

```
/**
 * Περιεχει ολες τις πληροφοριες ενος "συμβολαιου"
 * @author Panos
 *
 */
public class ContentContract {
	private int id; // Ο μοναδικος κωδικος του
	private int customerId; //Ο κωδικος πελατη
	private int workId; //Ο κωδικος εργασιας
	private String note; //Σημειωση
	private double workPrice; //Τιμη ανα μονοδα εκτασης χωραφιου
	private double fieldArea;//Εκταση χωραφιου
	private double discountPercent; //Εκπτωση συμβολαιου
	private boolean completed; //Ολοκληρωμενο
	private boolean notify; //Ειδοποιηση
	private LatLng point;//Σημειο χωραφιου στον χαρτη
	private int year; //Χρονος
	private int month; //Μηνας
	private int day;//Μερα συμβολαιου
	public ContentContract()
	{
		id=-1;
		customerId=-1;
		workId=-1;
		workPrice=0;
		fieldArea=0;
		discountPercent=0;
		year=0;
		day=0;
		month=0;
		note="";
		completed=true;
		notify=false;
		point=new LatLng(0,0);
	}
	/**
	 * Καθαρισμος συμβολαιου
	 */
	public void clear()
	{
		id=-1;
		customerId=-1;
		workId=-1;
		workPrice=0;
		fieldArea=0;
		discountPercent=0;
		year=0;
		day=0;
		month=0;
		note="";
		completed=true;
		notify=false;
		point=new LatLng(0,0);
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public int getCustomerId() {
		return customerId;
	}
	public void setCustomerId(int customerId) {
		this.customerId = customerId;
	}
	public int getWorkId() {
```

```
        return workId;
}
public void setWorkId(int workId) {
        this.workId = workId;
}
public double getFieldArea() {
        return fieldArea;
}
public void setFieldArea(double fieldArea) {
        this.fieldArea = fieldArea;
}
public void setFieldArea(String fieldArea) {
        try {
                this.fieldArea = Double.parseDouble(fieldArea);
        }
        catch(NumberFormatException e)
        {
                this.fieldArea = 0;

        }
}

public double getDiscountPercent() {
        return discountPercent;
}
public void setDiscountPercent(double discountPercent) {
        this.discountPercent = discountPercent;
}
public void setDiscountPercent(String discountPercent) {
        try {
                this.discountPercent = Double.parseDouble(discountPercent);
        }catch(NumberFormatException e)
        {
                this.discountPercent=0;
        }
}
public boolean isCompleted() {
        return completed;
}
public int getCompleted()
{
        return completed?1:0;
}
public void setCompleted(boolean completed) {
        this.completed = completed;
}
public boolean isNotify() {
        return notify;
}
public int getNotify()
{
        return notify?1:0;
}
public void setNotify(boolean notify) {
        this.notify = notify;
}
public LatLng getPoint() {
        return point;
}
public void setPoint(LatLng point) {
        this.point = point;
}
public void setPoint(double latitude,double longidute)
{
        this.point=new LatLng(latitude,longidute);
}
public double getPriceWithoutDiscount()
{
        return Math.round(((fieldArea*workPrice*100.00)/100)*100.0)/100.00;

}
public String getPriceWithoutDiscountToString()
{
        try
```

```
		{
			return Double.toString(getPriceWithoutDiscount())+" €";
		}
		catch (NumberFormatException e)
		{
			return "0 €";
		}
	}
	public double getPriceWithDiscount()
	{
		double temp=getPriceWithoutDiscount();
		temp-=((getPriceWithoutDiscount()*discountPercent)/100);
		temp=Math.round(temp*100.00)/100.00;
		if(temp<0)
			temp=0;
		return temp;
	}
	public String getPriceWithDiscountToString()
	{
		try
		{
			return Double.toString(getPriceWithDiscount())+" €";
		}
		catch (NumberFormatException e)
		{
			return "0 €";
		}
	}
	public double getWorkPrice() {
		return workPrice;
	}
	public void setWorkPrice(double workPrice) {
		this.workPrice = workPrice;
	}
	public void setWorkPrice(String workPrice)
	{
		try
		{
			this.workPrice=Double.parseDouble(workPrice);
		}
		catch(NumberFormatException e)
		{
			this.workPrice=0;
		}
	}

	public int getYear() {
		return year;
	}
	public void setYear(int year) {
		this.year = year;
	}
	public int getMonth() {
		return month;
	}
	public void setMonth(int month) {
		this.month = month;
	}
	public int getDay() {
		return day;
	}
	public void setDay(int day) {
		this.day = day;
	}
	public String getNote() {
		return note;
	}
	public void setNote(String note) {
		this.note = note;
	}

}
```

Κλάση DataHelperContractor περιέχει όλες τις συναρτήσεις για το διάβασμα, ενημέρωση, διαγραφή και δημιουργία τιμών για τον πίνακα Contractor

```
public class DataHelperContractor {
private OpenHelperDatabase openHelper;

	public DataHelperContractor(Context context)
	{
		openHelper=new OpenHelperDatabase (context);
	}
	/**
	 * Διαγραφη τον πελατη με το id πελατη που περνει σαν εισοδο
	 * @param customID
	 */
	public void deleteContractor(int contractorID)
	{
		try
		{
			String query="delete from CONTRACTOR where id="+contractorID;

			SQLiteDatabase db=openHelper.getWritableDatabase();

			db.execSQL(query);

			db.close();


		} catch(SQLiteException e)
		{
			Log.e("ContractorDataHelper::deleteContractor", e.getMessage());
		}

	}
	/**
	 * Dimiourgi ena pelati
	 * @param customer dedomena pelati
	 * @return true an dimiourgithike false an den dimiourgio
	 */
	public boolean createContractor(ContentContractor contractor)
	{
		boolean created=false;
		try
		{
			String query="insert into CONTRACTOR(NAME,LASTNAME,ADDRESS,PHONE,VAT)
values('"+contractor.getName()+"','"+

	contractor.getLastName()+"','"+contractor.getAddress()+"','"+contractor.getPhone()+"',
'"+contractor.getVat()+"')";



			if(!existVAT(contractor.getVat()))
			{
				SQLiteDatabase db=openHelper.getWritableDatabase();
				db.execSQL(query);
				created=true;
				db.close();
			}
			else
				Log.e("ContractorDataHelper::createContractor","To id
yparxh");



		} catch(SQLiteException e)
		{
			Log.e("ContractorDataHelper::createContractor", e.getMessage());
		}

		return created;
	}
```

```
public boolean hasContractor()
{
	boolean value=false;

	try
	{
		String query="select id from CONTRACTOR";
		SQLiteDatabase db=openHelper.getWritableDatabase();
		Cursor cursor=db.rawQuery(query, null);

		if(cursor.moveToFirst())
			value=true;

		cursor.close();
		db.close();
	}
	catch(SQLiteException e)
	{
		Log.e("ContractorDataHelper::hasContractor", e.getMessage());
	}

	return value;
}

/**
 *
 * @param id
 * @return
 */
public ContentContractor getContractor()
{
	ContentContractor customer=new ContentContractor();
	try
	{
		String query="select * from CONTRACTOR ";
		SQLiteDatabase db=openHelper.getWritableDatabase();
		Cursor cursor=db.rawQuery(query, null);

		if(cursor.moveToFirst())
		{
			customer.setId(cursor.getInt(0));
			customer.setName(cursor.getString(1));
			customer.setLastName(cursor.getString(2));
			customer.setAddress(cursor.getString(3));
			customer.setPhone(cursor.getString(4));
			customer.setVat(cursor.getString(5));
		}
		cursor.close();
		db.close();

	} catch(SQLiteException e)
	{
		Log.e("ContractorDataHelper::getContractor", e.getMessage());
	}
	return customer;
}



/**
 *
 * @param vat
 * @return
 */
public boolean existVAT(String vat)
{
	boolean exists=false;
	try
	{
		String query="select vat from CONTRACTOR where vat='"+vat+"'";

		SQLiteDatabase db=openHelper.getWritableDatabase();

		Cursor cursor=db.rawQuery(query, null);
```

```
                        exists=cursor.moveToFirst();
                        cursor.close();
                        db.close();

                } catch(SQLiteException e)
                {
                        Log.e("ContractorDataHelper::existVAT", e.getMessage());
                }

                return exists;
        }

        public boolean updateContractor(ContentContractor contractor)
        {
                boolean success=false;


                try {
                        SQLiteDatabase db=openHelper.getWritableDatabase();

                        String query="update CONTRACTOR set
NAME='"+contractor.getName()+"',LASTNAME='"+contractor.getLastName()+"',ADDRESS='"

        +contractor.getAddress()+"',PHONE='"+contractor.getPhone()+"',VAT='"+contractor.getVat
()+"' where id="+contractor.getId();

                        db.execSQL(query);

                        db.close();
                        success=true;
                }catch (SQLiteException e) {

        Log.e("ContractorDataHelper::updateContractor",e.getLocalizedMessage());
                        success=false;
                }
                return success;
        }

}
```

Κεφάλαιο 4

Οδηγίες εγκατάστασης και χρήσης της εφαρμογής

Οδηγίες εγκατάστασης

Για android έως 2.3 (Gingerbread)

Από τις Ρυθμίσεις -> Εφαρμογές-> Άγνωστες πήγες (Ενεργοποιούμε)

Για android από 4 (ICS)
Από τις Ρυθμίσεις -> Ασφάλεια-> Άγνωστες πήγες (Ενεργοποιούμε)

Προϋποθέτει:

Android 2.2 και μεγαλύτερο
OpenGL 2.0
Google Play Account

Αφού έχουμε ενεργοποιήσει τις άγνωστες πήγες κατεβάζουμε τα apk από το site και τα εγκαθιστούμε.

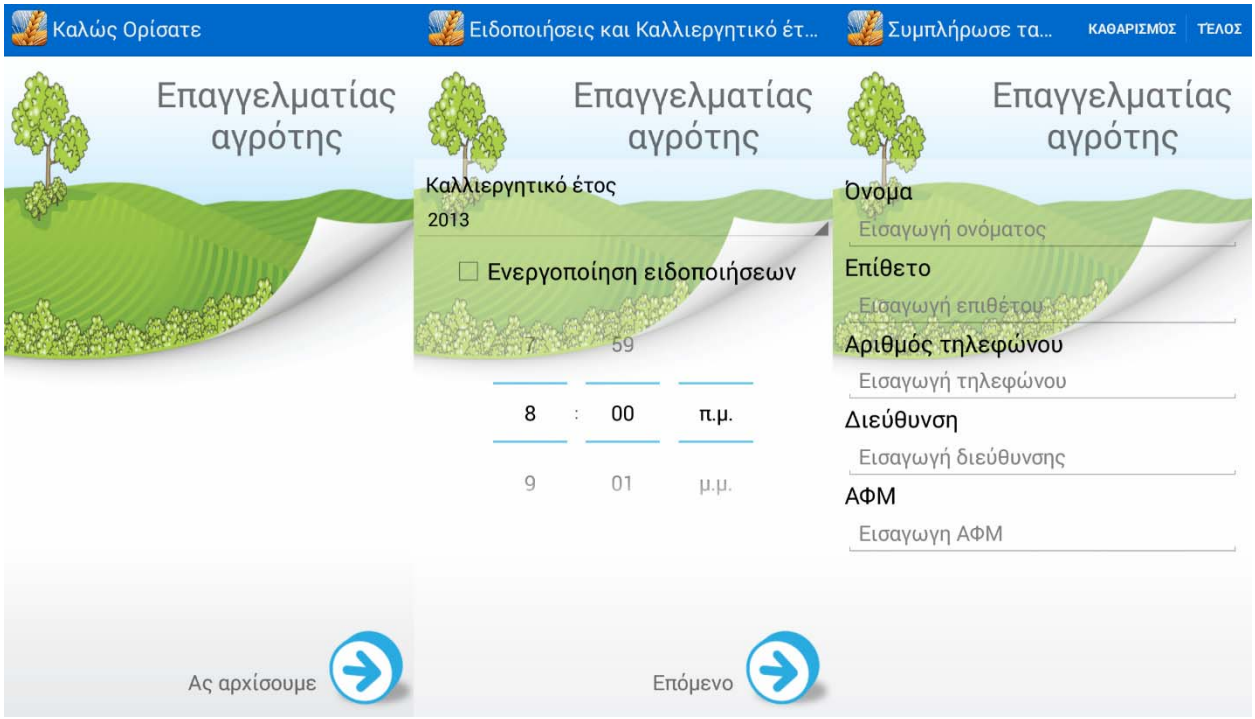
Αν τα κατεβάσουμε από υπολογιστή τότε τα περνάμε μέσα στην συσκευή και με έναν file manager τα εγκαθιστούμε.

Παρουσίαση της εφαρμογής

Πρώτο ξεκίνημα

Στο πρώτο ξεκίνημα βρίσκονται όλες οι παραμετροποιήσεις που πρέπει να γίνουν όταν ***τρέξει*** για πρώτη φορά η εφαρμογή.

Πατώντας το κουμπί ***Ας αρχίσουμε*** ή ***μετακινώντας (scroll) προς τα αριστερά*** εμφανίζονται οι επιλογές.



Σχήμα 4.1

Ειδοποιήσεις και Καλλιεργητικό έτος

Στην πρώτη καρτέλα επιλέγετε το τρέχων καλλιεργητικό έτος, αν ενεργοποιηθούν οι ειδοποιήσεις για εργασίες που είναι σε εξέλιξη καθώς και την ώρα που θα δημιουργούνται αυτές αν υπάρχουν.

Στην συνέχεια πατώντας το κουμπί ***Επόμενο*** ή ***μετακινώντας (scroll) προς τα αριστερά*** εμφανίζεται η επόμενη καρτέλα.

Συμπληρώστε τα στοιχεία σας.

Στην καρτέλα αυτήν συμπληρώνονται τα στοιχεία του χρήστη.

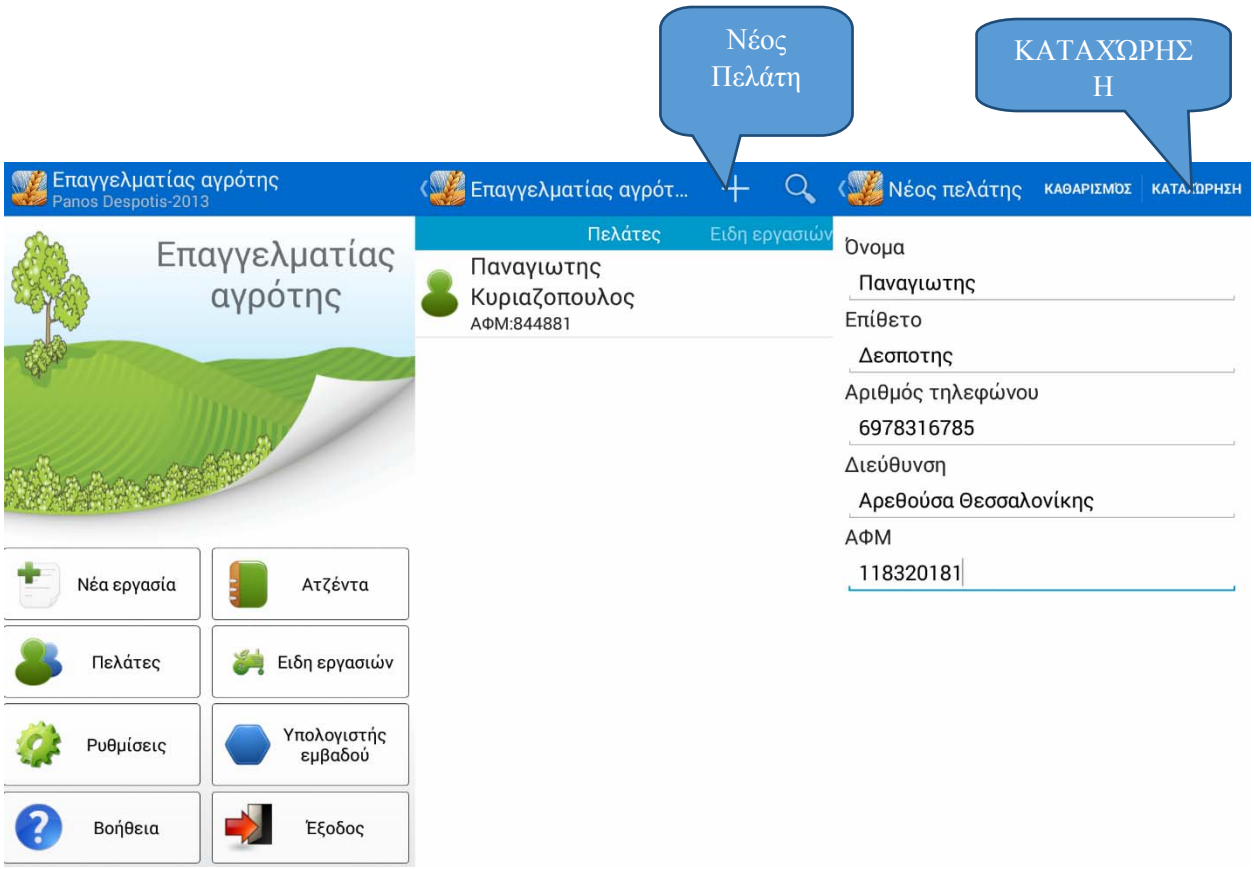
Αυτό το βήμα δεν είναι απαραίτητο.

Όταν τελειώσει η παραμετροποίηση πατώντας ***ΤΕΛΟΣ*** ξεκινάει η εφαρμογή.

Πελάτες

Νέος πελάτης

Για να δημιουργήσουμε νέο πελάτη από το κεντρικό μενού επιλέγουμε Πελάτες-> Νέος Πελάτης

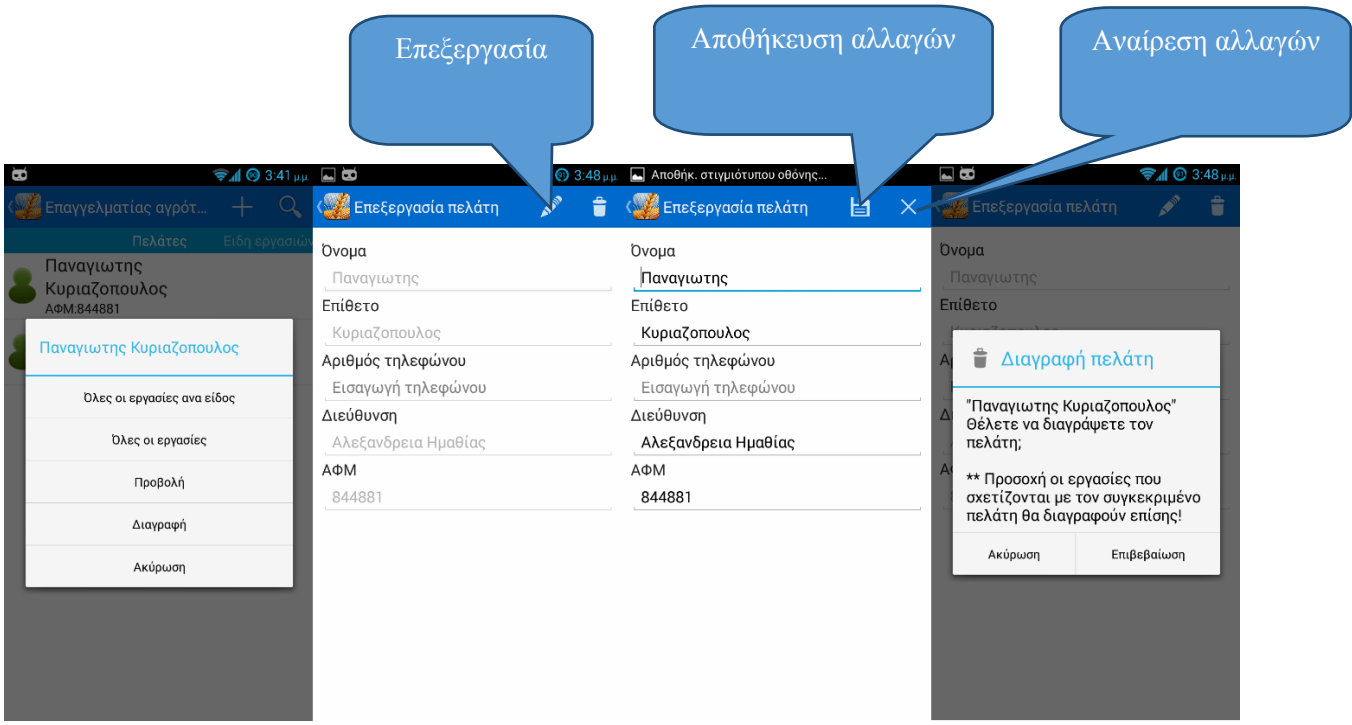


Σχέδιο 4.2

Συμπληρώνουμε τα στοιχεία. Το όνομα και το επίθετο είναι υποχρεωτικά!
Το ΑΦΜ δεν είναι υποχρεωτικό αλλά είναι μοναδικό . Δεν μπορεί δύο πελάτες να έχουν το ίδιο ΑΦΜ.
Όταν συμπληρώσετε να στοιχεία του πελάτη πατήστε ***ΚΑΤΑΧΩΡΗΣΗ.***

Προβολή πελάτη

Για να προβάλετε να στοιχεία ενός πελάτη από την καρτέλα πελάτη όπου βρίσκεστε πατήστε πάνω του για να εμφανιστεί το μενού πελάτη και επιλέξτε προβολή.



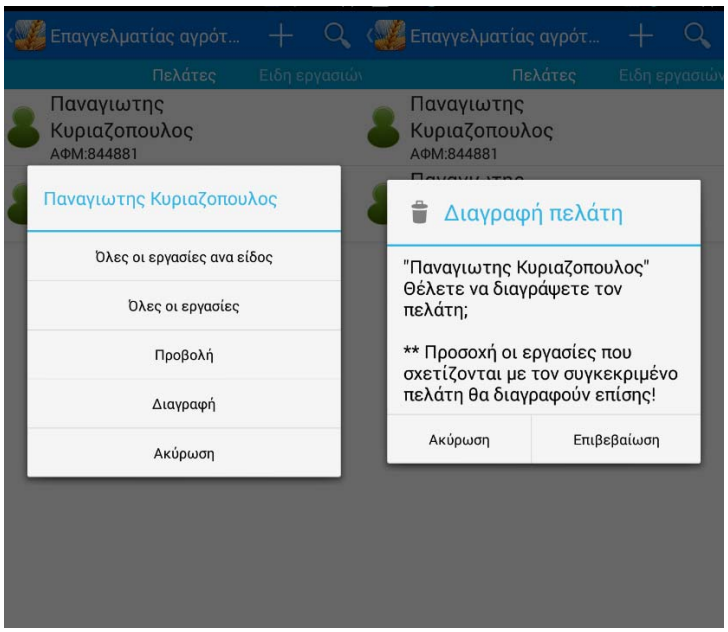
Σχέδιο 4.3

Από την προβολή όπου βρίσκεστε μπορείτε είτε να επεξεργαστείτε τα στοιχεία του ή να τον διαγράψετε.

**** Προσοχή όταν διαγράφετε έναν πελάτη διαγράφονται και οι εργασίες που σχετίζονται με τον συγκεκριμένο πελάτη.*

Διαγραφή πελάτη

Μπορείτε να διαγράψετε τον πελάτη κατευθείαν από το μενού πελάτη χωρίς να πάτε στην προβολή.

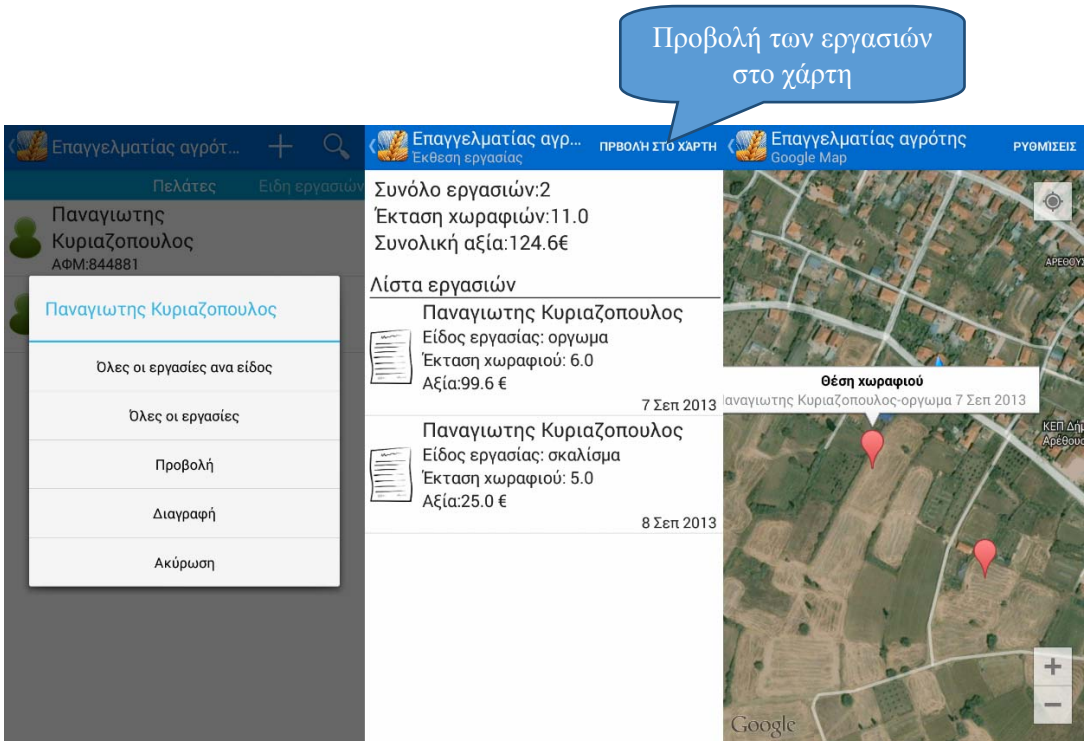


Σχέδιο 4.4

Προβολή εργασιών πελάτη

Α) Προβολή όλων των εργασιών πελάτη

Προβάλετε όλες τις εργασίες του πελάτη επιλέγοντας τον από την καρτέλα πελάτων και πατώντας ‘’ Όλες οι εργασίες’’.
Εμφανίζετε η λίστα με τις εργασίες που έχουν καταχωρηθεί στον πελάτη καθώς και το πλήθος και άθροισμα της αξίας όλων των εργασιών.
Μπορείτε επίσης και να προβάλετε τις εργασίες στον χάρτη εφόσον έχετε καταχώρηση σημεία χωραφιού για κάθε εργασία.

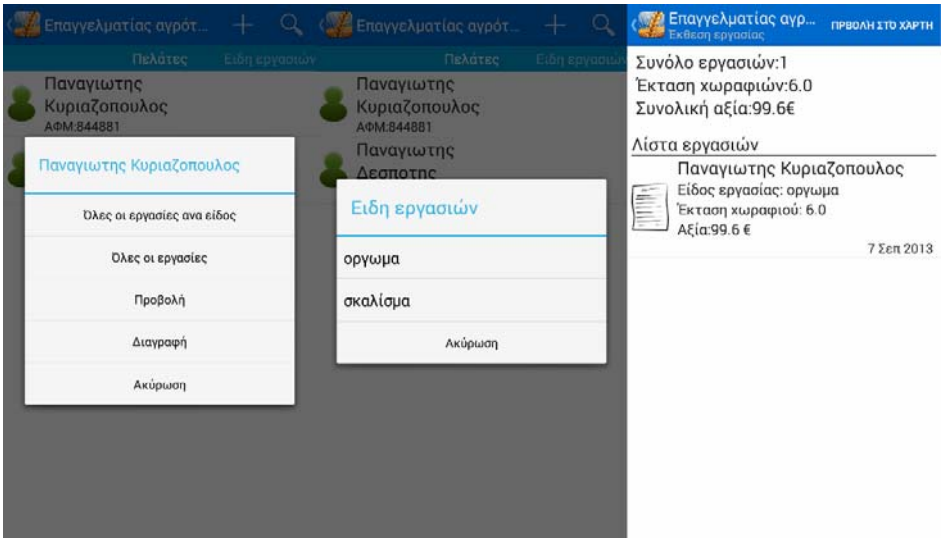


Σχέδιο 4.5

Όταν πατήσετε σε ένα σημείο χωραφιού στο χάρτη εμφανίζετε ένα παραθυράκι με περισσότερες πληροφορίες για την συγκεκριμένη εργασία. Πατώντας πάνω του μας πηγαίνει στην προβολή εργασίας.

Β) Προβολή εργασιών ανά είδος εργασίας.

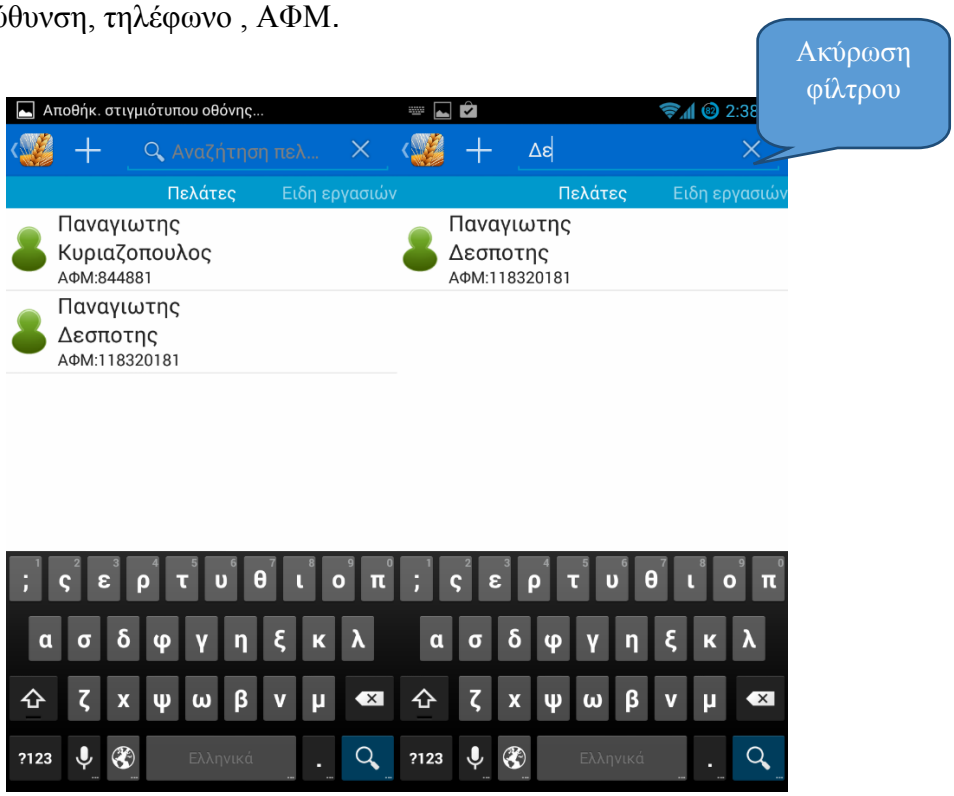
Για να προβάλετε εργασίες ενός πελάτη με συγκεκριμένο είδος εργασίας (π.χ. όλα τα οργώματα για τον συγκεκριμένο πελάτη) επιλέξτε ‘’Όλες οι εργασίες ανά είδος’’ και έπειτα επιλέξτε το είδος.



Σχέδιο 4.6

Αναζήτηση πελάτη

Μπορείτε να αναζητήσετε έναν πελάτη από την καρτέλα των πελάτων. Με το όνομα, επίθετο ,διεύθυνση, τηλέφωνο , ΑΦΜ.

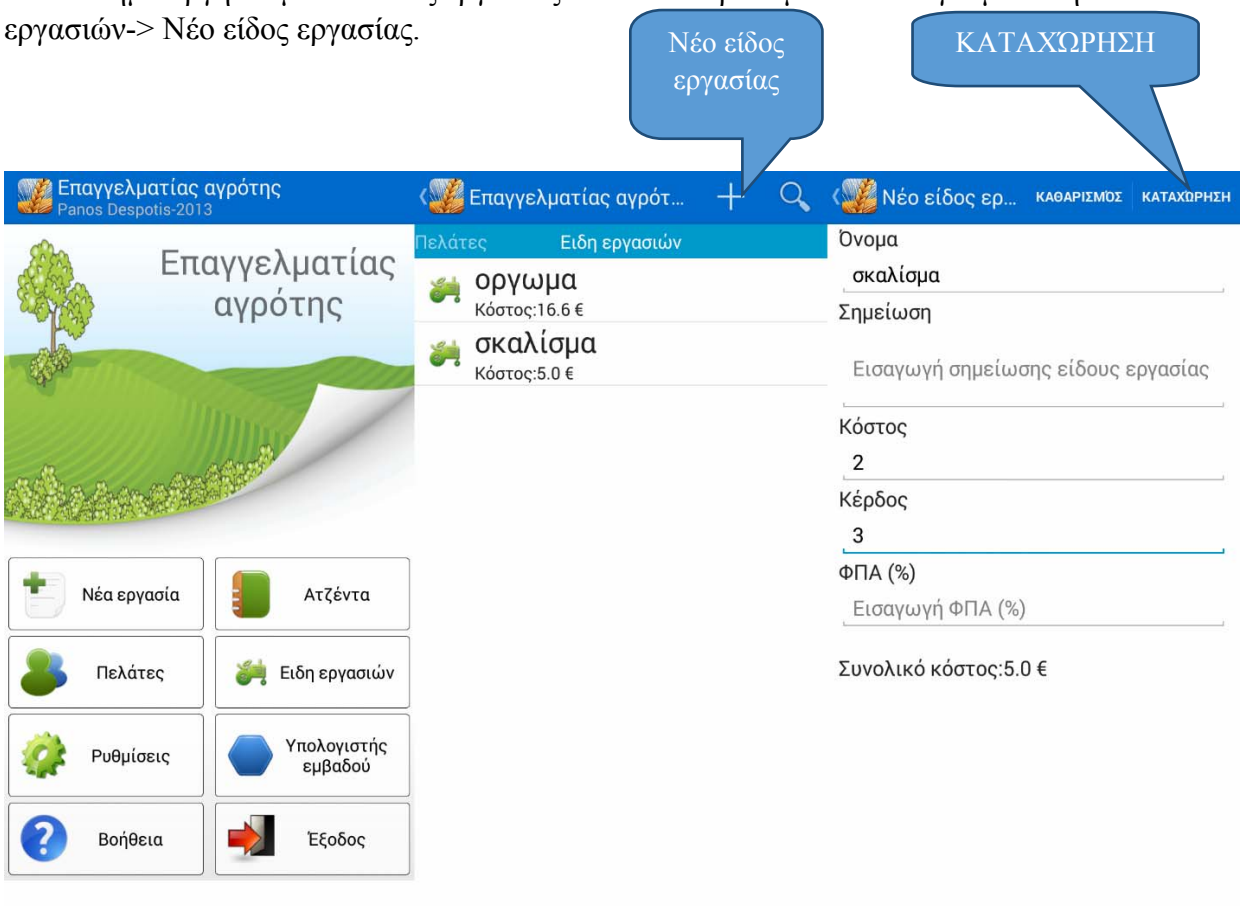


Σχέδιο 4.7

Είδη εργασιών

Νέο είδος εργασίας

Για να δημιουργήσουμε νέο είδος εργασίας από το κεντρικό μενού επιλέγουμε Είδη εργασιών-> Νέο είδος εργασίας.

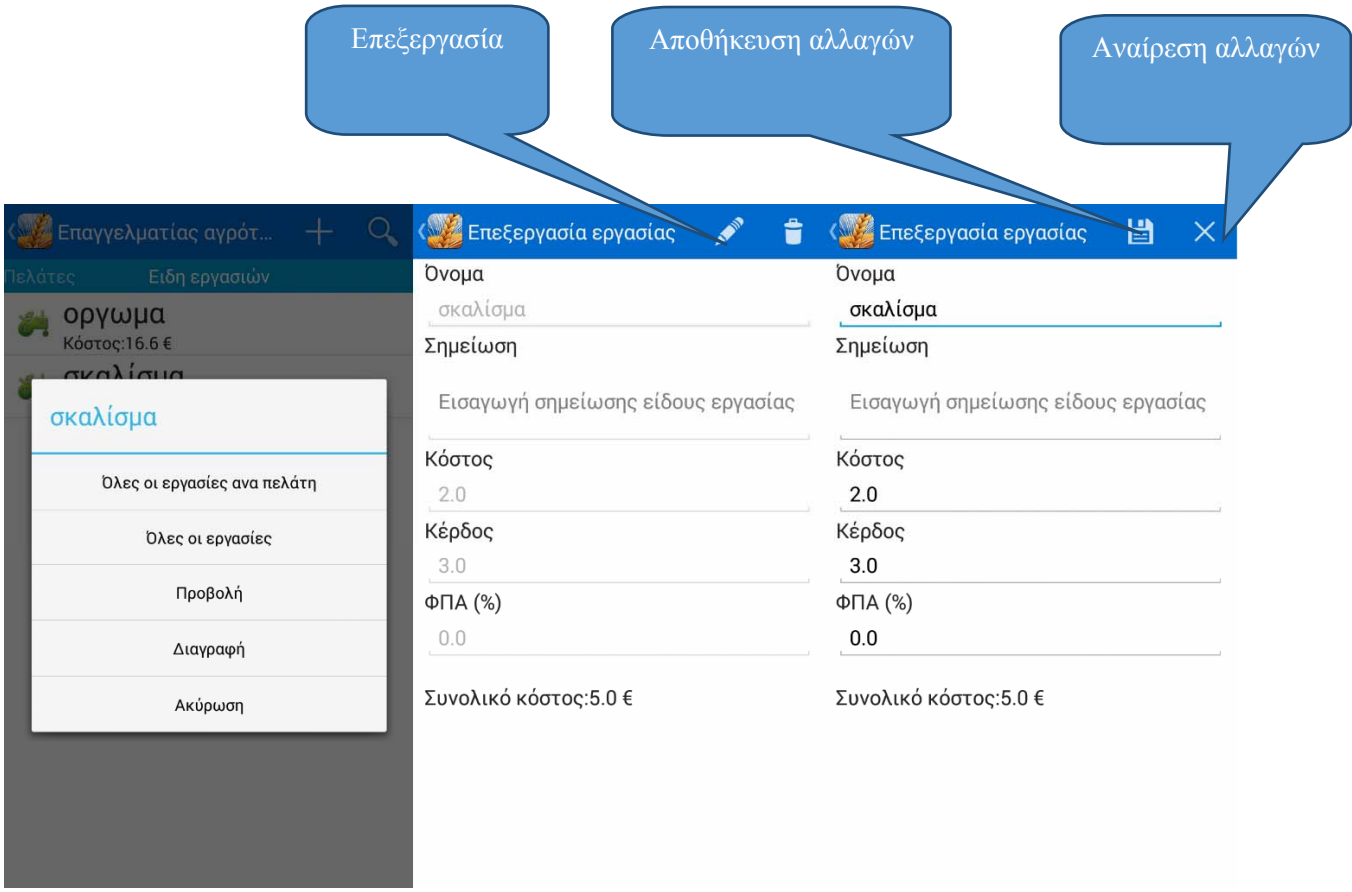


Σχέδιο 4.8

Συμπληρώνουμε τα στοιχεία. Το όνομα είναι υποχρεωτικό!
Το κόστος και το κέρδος δεν είναι υποχρεωτικά αλλά αν δεν συμπληρωθεί τουλάχιστον ένα από τα δύο το είδος θα έχει μηδενική τιμή.
Όταν συμπληρώσετε να στοιχεία του είδους πατήστε **ΚΑΤΑΧΏΡΗΣΗ**.

Προβολή είδους εργασίας

Για να προβάλετε τα στοιχεία ενός είδους εργασίας από την καρτέλα Είδη εργασίας όπου βρίσκεστε πατήστε πάνω σε μία για να εμφανιστεί το μενού είδους και επιλέξτε προβολή.



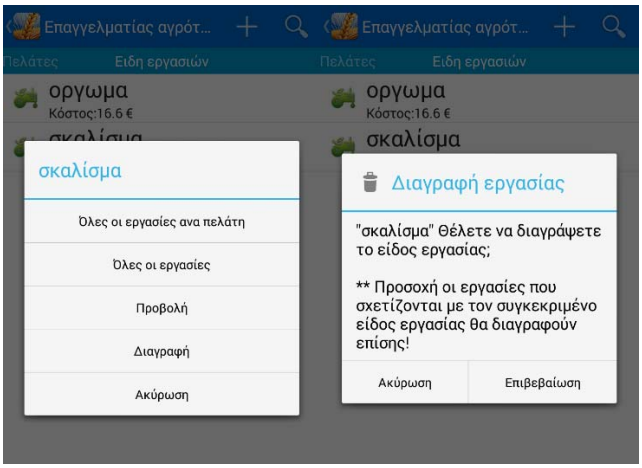
Σχέδιο 4.9

Από την προβολή όπου βρίσκεστε μπορείτε είτε να επεξεργαστείτε τα στοιχεία της ή να την διαγράψετε.

**** Προσοχή όταν διαγράφετε ένα είδος εργασίας διαγράφονται και οι εργασίες που σχετίζονται με τον συγκεκριμένο είδος.*

Διαγραφή είδους εργασίας

Μπορείτε να διαγράψετε το είδος εργασίας κατευθείαν από το μενού του είδους χωρίς να πάτε στην προβολή.



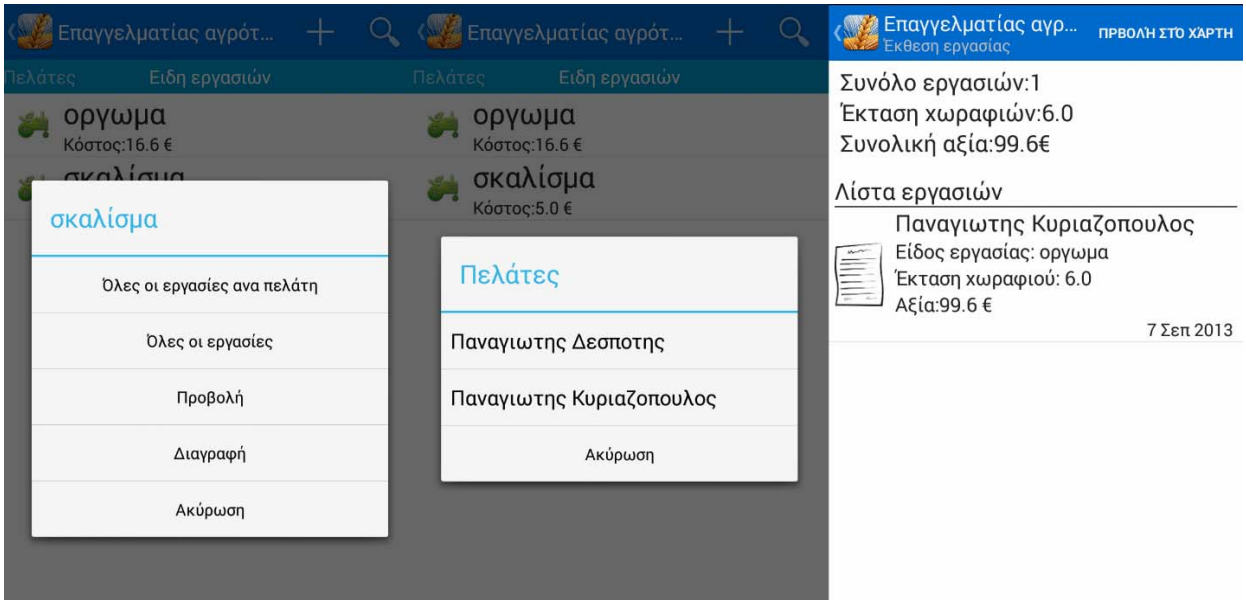
Σχέδιο 4.10

Προβολή εργασιών του είδους εργασίας

Α) Προβολή όλων των εργασιών του είδους

Προβάλετε όλες τις εργασίες του είδους επιλέγοντας το από την καρτέλα είδη εργασίας και πατώντας ‘’ Όλες οι εργασίες’’.
Εμφανίζετε η λίστα με τις εργασίες που έχουν γίνει με το είδος καθώς και το πλήθος και άθροισμα της αξίας όλων των εργασιών.
Μπορείτε επίσης και να προβάλετε τις εργασίες στον χάρτη εφόσον έχετε καταχώρηση σημεία χωραφιού για κάθε εργασία.

Β) Προβολή εργασιών ανά πελάτη.

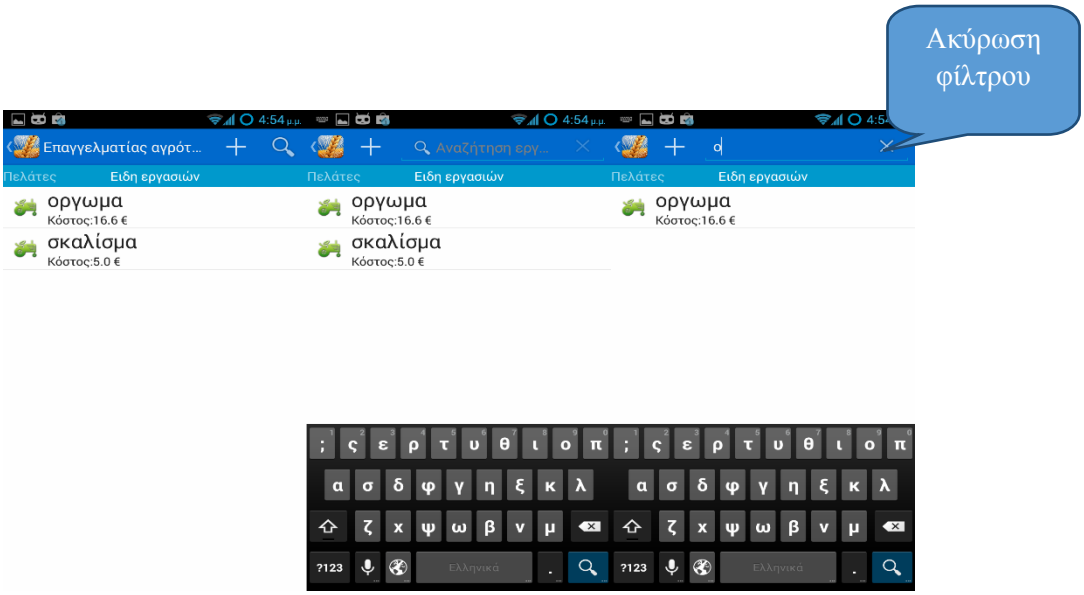


Σχέδιο 4.11

Για να προβάλετε εργασίες ενός πελάτη με συγκεκριμένο είδος εργασίας (π.χ. όλα τα οργώματα για τον συγκεκριμένο πελάτη) επιλέξτε ‘’Όλες οι εργασίες ανά είδος’’ και έπειτα επιλέξτε το είδος.

Αναζήτηση ειδών εργασίας

Μπορείτε να αναζητήσετε ένα είδος εργασίας από την καρτέλα των ειδών εργασιών. Με το όνομα είδους ή την σημείωση.

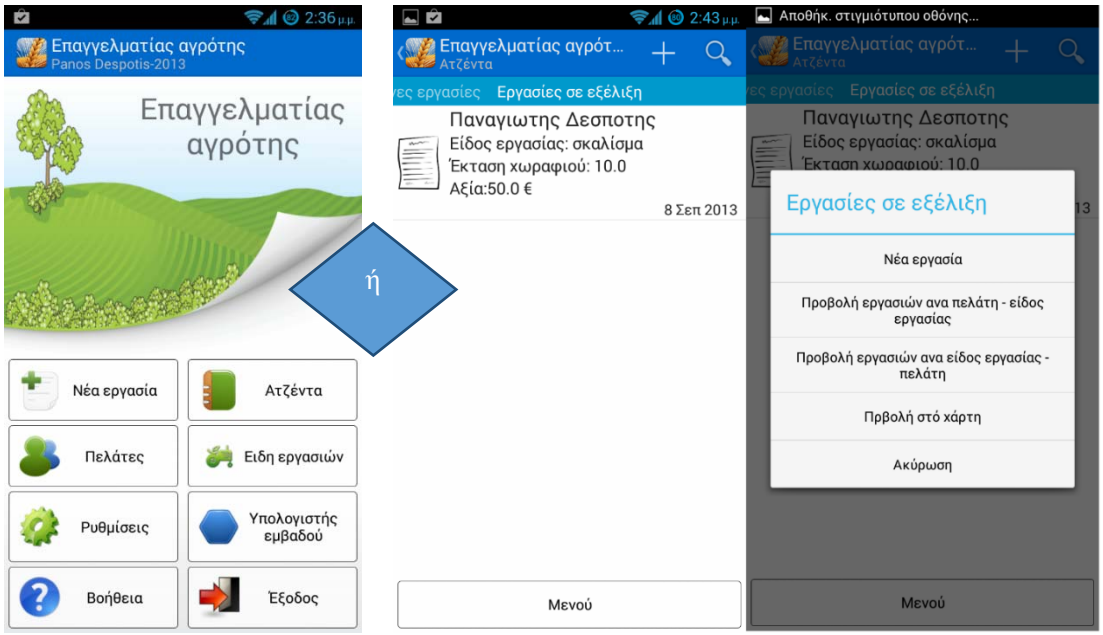


Σχέδιο 4.12

Εργασίες-Ατζέντα

Νέα εργασία

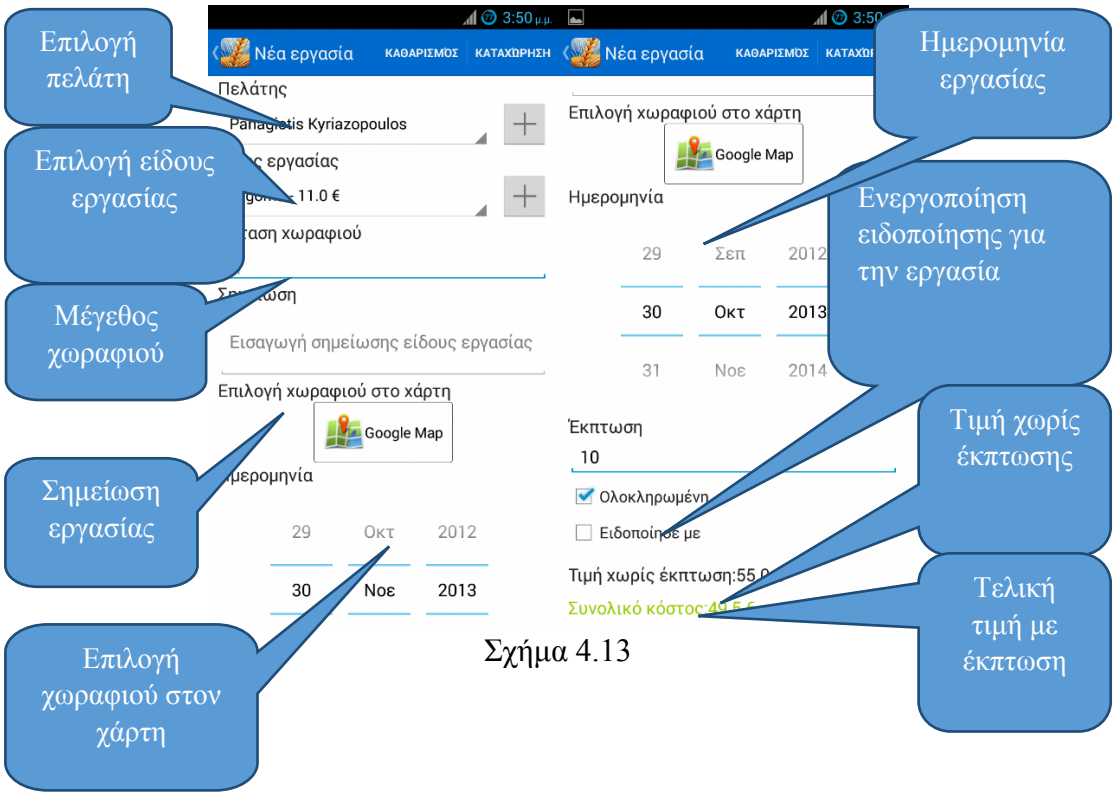
Για να δημιουργήσετε μια νέα εργασία από το κεντρικό μενού πατήστε **Νέα εργασία** ή από το μενού της ατζέντας επιλέξτε Νέα εργασία.



Σχήμα 4.13

Συμπληρώστε τα στοιχεία της φόρμας. Απαραίτητα ***είναι η επιλογή πελάτη, είδους εργασίας και έκταση χωραφιού***.
Μπορείτε επίσης να επιλέξετε και το χωράφι που θα γίνει η εργασία στον χάρτη, την ημερομηνία που θα γίνει η εργασία καθώς και ένα ποσοστό έκπτωσης στην τελική

τιμή της εργασίας. Αν η εργασία δεν έχει ολοκληρωθεί μπορείτε να πάρετε ειδοποίηση για την συγκεκριμένη εργασία αν επιλέξετε Ειδοποίησε με.



Σχήμα 4.13

Ολοκληρωμένες εργασίες – Εργασίες σε εξέλιξη

Οι εργασίες χωρίζονται σε δύο κατηγορίες στις Ολοκληρωμένες και σε Εξέλιξη Ολοκληρωμένες θεωρούνται οι εργασίες που έχουν γινεί και σε εξέλιξη αυτές που θα γίνουν στο μέλλον.

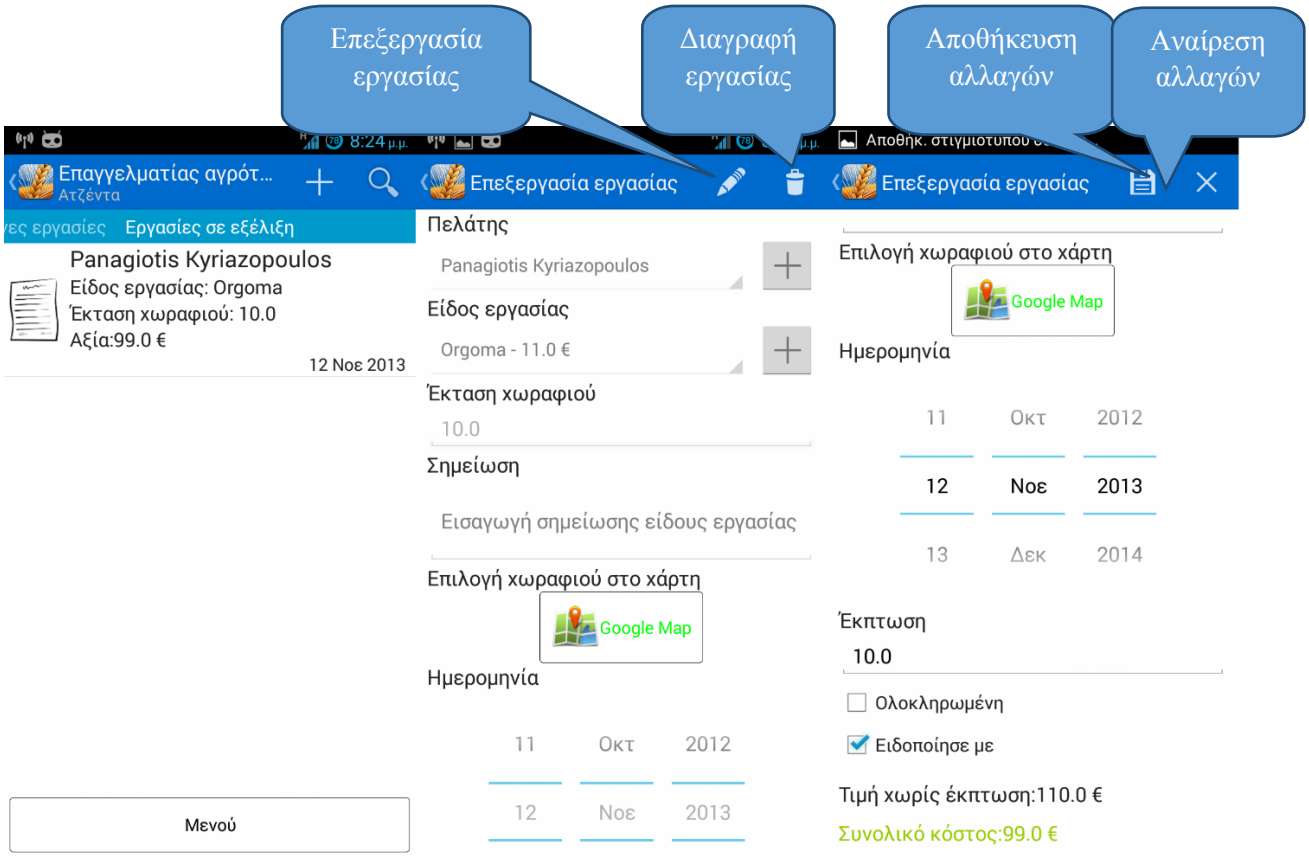
Αν επιλέξετε την εργασία ως Ολοκληρωμένη τότε η εργασία τοποθετείτε στις ολοκληρωμένες εργασίες στην ατζέντα και καταργείτε η ειδοποίηση αν έχει.

Αν επιλέξετε την εργασία ως Ειδοποίησε με τότε καταργείτε η κατάσταση ολοκληρωμένη αν έχει επιλεχθεί και η εργασία τοποθετείτε στις σε εξέλιξη εργασίες στην ατζέντα και ενεργοποιήτε η ειδοποίηση για αυτήν.

Προβολή εργασίας

Για να προβάλουμε μια εργασία από το κεντρικό μενού επιλέγουμε Ατζέντα.

Στην ατζέντα υπάρχουν δυο καρτέλες. Οι ολοκληρωμένες εργασίες και οι εργασίες σε εξέλιξη.
Για να προβάλουμε μια εργασία πατάμε πάνω της.



Σχέδιο 4.14

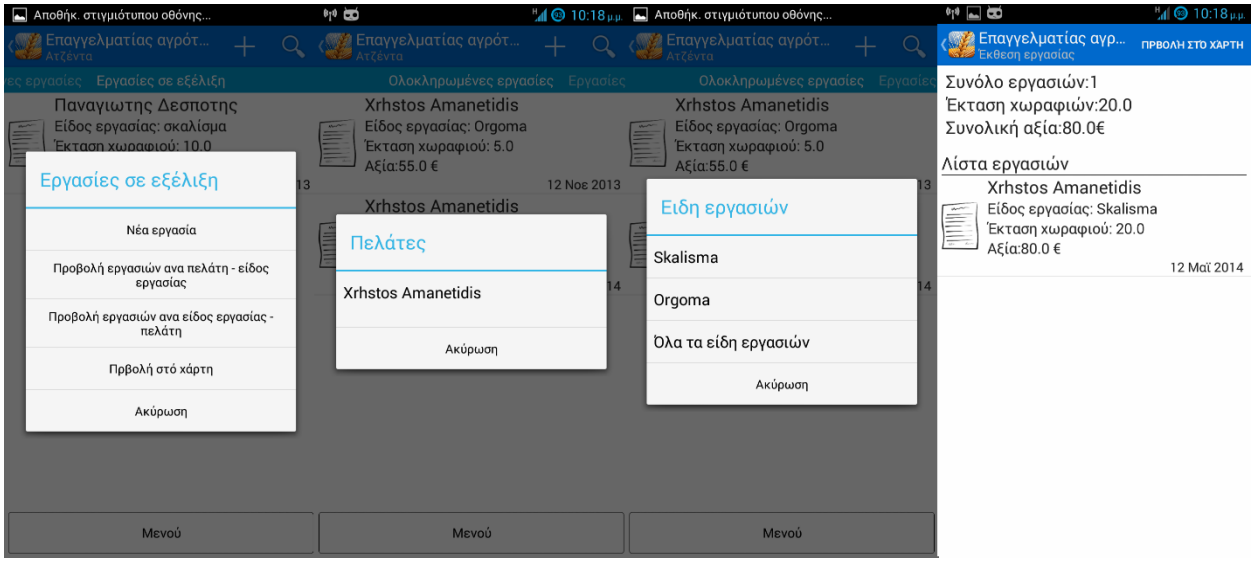
Για να επεξεργαστείτε τα στοιχεία μίας εργασίας από την προβολή πατήστε το μολύβι για να ενεργοποιηθεί η φόρμα και να μπορείτε να κάνετε αλλαγές.
Πατώντας στο εικονίδιο της δισκέτας αποθηκεύονται οι αλλαγές. Αν θέλετε να αναιρέσετε τις αλλαγές που έχετε κάνει πατήστε στο εικονίδιο της αναίρεσης.

Προβολή εργασιών ανά πελάτη – είδος εργασίας

Μπορείτε να προβάλλετε τις εργασίες που έχετε κάνει σε έναν πελάτη ανά είδος εργασίας επιλέγοντας από το μενού της ατζέντας.

*** Τα αποτελέσματα φιλτράρονται ανάλογα σε ποια καρτέλα βρίσκεστε. Δηλαδή στις εργασίες σε εξέλιξη ή στις ολοκληρωμένες εργασίες.

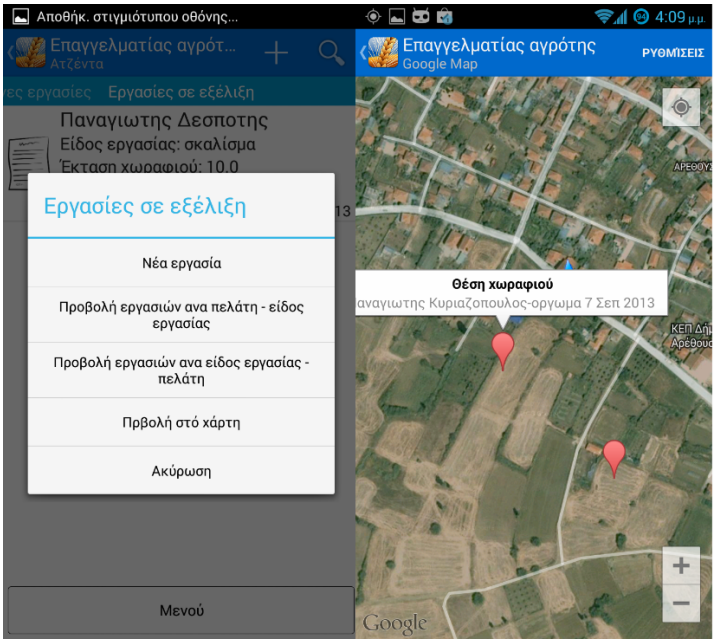
Επίσης μπορείτε να προβάλετε και τις εργασίες ανά είδος εργασίας – πελάτη.



Σχέδιο 4.15

Προβολή εργασιών στο χάρτη.

Μπορείτε να προβάλετε όλες τις εργασίες ανάλογα σε ποια καρτέλα βρίσκεστε (Ολοκληρωμένες – Σε εξέλιξη εργασίες) στο χάρτη.

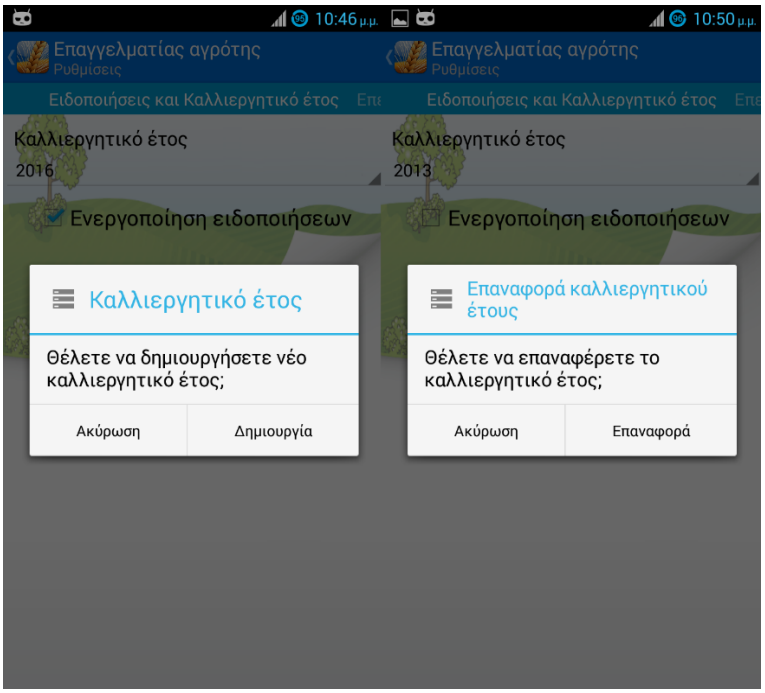


Σχέδιο 4.16

Ρυθμίσεις

Ειδοποιήσεις και καλλιεργητικό έτος

Με τη δημιουργία ενός νέου καλλιεργητικό έτους παίρνεται ένα αντίγραφο ασφαλείας με όλα τα δεδομένα και δημιουργείται το νέο με όλα τα πεδία κενά. Αν επιλεγεί ένα έτος που ήδη υπάρχει τότε γίνεται αυτόματα επαναφορά των δεδομένων.



Σχέδιο 4.17

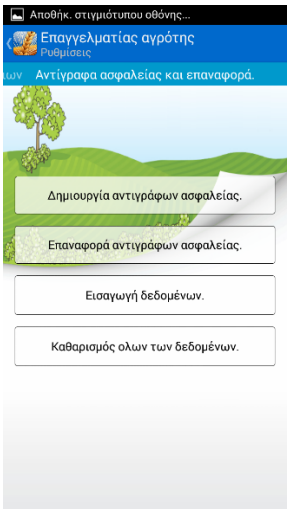
Ειδοποιήσεις

Μπορείτε να παίρνετε καθημερινά ειδοποιήσεις για τις εργασίες που είναι σε εξέλιξη και έχουν προγραμματιστεί να γίνουν τη συγκεκριμένη μέρα ρυθμίζοντας και την ώρα που το κινητό σας θα σας ειδοποιεί.

Επεξεργασία στοιχείων

Μπορείτε να αλλάξετε τα προσωπικά σας στοιχεία.

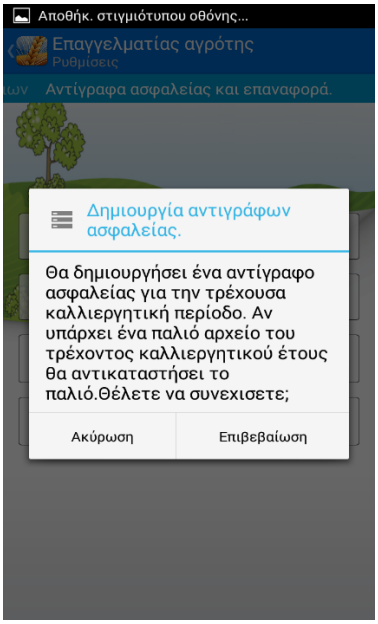
Αντίγραφα Ασφαλείας και Επαναφορά



Σχέδιο 4.18

Δημιουργία Αντιγράφων Ασφαλείας.

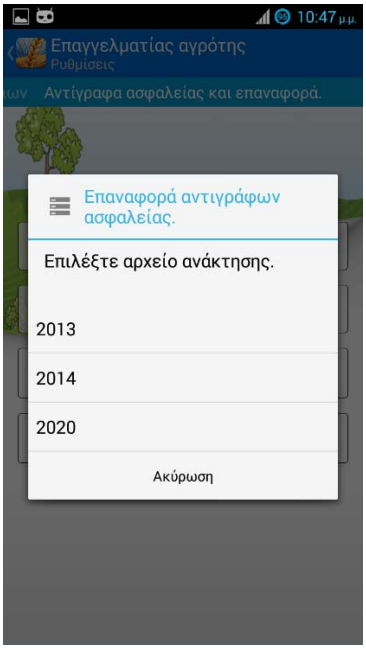
Με την επιλογή της συγκεκριμένης ρύθμισης μπορείτε να αποθηκεύετε όλα τα δεδομένα της τρέχουσας βάσης στο αντίγραφο ασφαλείας του τρέχοντος καλλιεργητικού έτους. Αν υπάρχει ήδη αντίγραφο ασφαλείας του έτους τότε θα αντικατασταθεί.



Σχέδιο 4.19

Επαναφορά Αντιγράφων Ασφαλείας

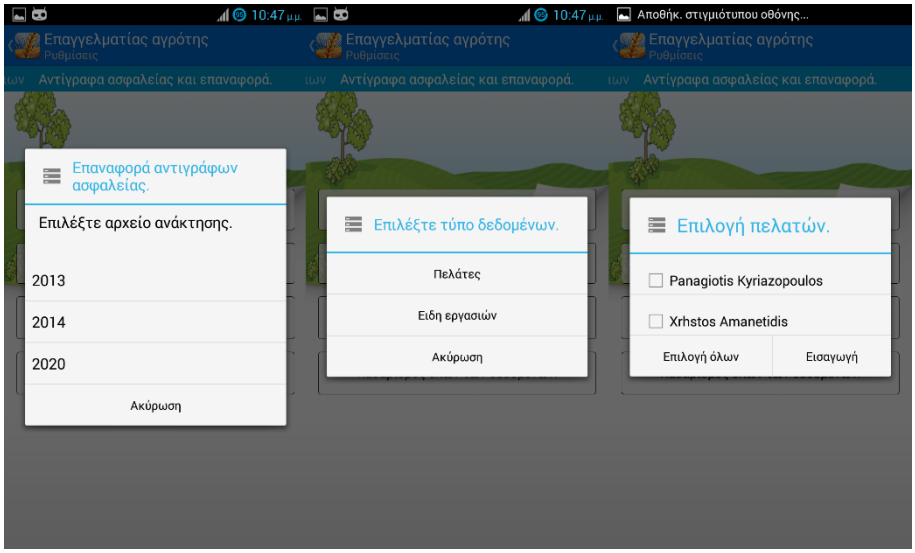
Μπορείτε άμεσα να επανακτήσετε όλα τα δεδομένα ενός καλλιεργητικού έτους επιλέγοντάς το.



Σχέδιο 4.20

Εισαγωγή Δεδομένων

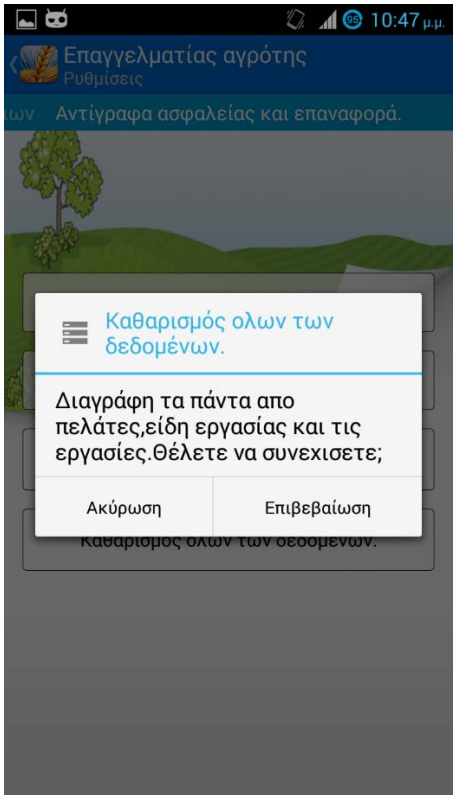
Μπορείτε να εισάγετε πελάτες και είδη εργασιών από άλλα καλλιεργητικά έτη.



Σχέδιο 4.21

Καθαρισμός όλων των δεδομένων.

Διαγράφει όλα τα δεδομένα του τρέχοντος καλλιεργητικού έτους χωρίς να μεταβάλλει το αντίγραφο ασφαλείας του τρέχοντος καλλιεργητικού έτους (αν υπάρχει).



Σχέδιο 4.22

Κεφάλαιο 5

Σε ποιο βαθμό εκπληρώθηκαν οι στόχοι της εφαρμογής;

Η εφαρμογή εκπληρώνει όλους τους στόχους. Στο μέλλον υπάρχει η δυνατότητα να εντάξουμε την παροχή έκδοσης αποδείξεων, εισαγωγή ημερολογίου για την καλύτερη διαχείριση των εργασιών και του χρόνου, cloud services ώστε να έχεις ταυτόχρονη πρόσβαση στα δεδομένα τόσο από το κινητό όσο και από τον υπολογιστή, δυνατότητα ευκολότερης αποστολής της βάσης μέσω Bluetooth ή NFC.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1) http://developer.android.com/develop/index.html
2) www.stackoverflow.com
3) Πτυχιακή εργασία του σπουδαστή Κουκουρή Γεώργιου.
4) http://www.eclipse.org
5) http://www.oracle.com/index.html
6) https://developers.google.com/maps/documentation/android/
7) http://www.vogella.com