ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΕΠΙΚΟΙΝΩΝΙΩΝ

Διαχειριστής αγροκτήματος από συσκευές με λειτουργικό σύστημα Android

Πτυχιακή εργασία του

Κουκουρή Γεώργιου (2216)

Επιβλέπων: Λάντζος Θεόδωρος

Σέρρες, Απρίλιος 2013

Υπεύθυνη Δήλωση : Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Πληροφορικής & Επικοινωνιών του Τ.Ε.Ι. Σερρών.

Περίληψη

Στην παρούσα πτυχιακή εργασία αναπτύχθηκε η εφαρμογή FarmManager. Ο FarmManager είναι μια εφαρμογή διαχείρισης αγροκτήματος από συσκευές κινητής τηλεφωνίας (SmartPhones) με λειτουργικό σύστημα Android. Αποτελεί το σύγχρονο εργαλείο του αγρότη για την άμεση, ορθή και γρήγορη διαχείριση των χωραφιών, εργασιών και λειτουργιών στο αγρόκτημα απευθείας από το τρακτέρ επάνω. Εκμεταλλεύεται την σύγχρονη τεχνολογία και με ένα απλό, άμεσο και λειτουργικό τρόπο αποτελεί ένα κινητό γραφείο το οποίο ενημερώνει, παρακολουθεί και πληροφορεί των αγρότη για κάθε στάδιο της εργασίας του.

Όλα τα στοιχεία (δεδομένα, εργασίες, καλλιέργειες) μπορούν να διαχειρίζονται εύκολα από το κινητό τηλέφωνο. Υποστηρίζει ηλεκτρονική χαρτογραφική διαχείριση κτημάτων με την βοήθεια των Google Maps.
Ο χρήστης μπορεί:

- Να εισάγει ένα νέο χωράφι και να βλέπει τα ήδη καταχωρημένα χωράφια του αγροκτήματος
- Να εισάγει και να επεξεργάζεται τις εργασίες για κάθε χωράφι
- Να εισάγει και να επεξεργάζεται την καλλιέργεια του κάθε χωραφιού
- Να μεταφέρει τα χωράφια σε νέο καλλιεργητικό έτος
- Να δημιουργεί και να επαναφέρει αντίγραφα ασφαλείας για κάθε καλλιεργητικό έτος
- Να εκτυπώνει λίστα αγροκτήματος έτοιμη για την δήλωση του ΟΠΕΚΕΠΕ

Τέλος παρέχει την δυνατότητα καταχώρησης μέχρι τεσσάρων ιδιοκτητών και ενός ενοικιαστή για κάθε χωράφι του αγροκτήματος.

Παρακάτω θα δούμε λίγα πράγματα για το Android, για την ραγδαία ανάπτυξη της αγοράς των εφαρμογών για mobile πλατφόρμες και τις προοπτικές που ανοίγονται για τους νέους προγραμματιστές την σύγχρονη εποχή.

Στη συνέχεια θα περιγράψουμε την χρήση των εργαλείων που χρειάζονται για την ανάπτυξη των εφαρμογών χρησιμοποιώντας την αντικειμενοστραφής γλώσσα προγραμματισμού Java.

Τέλος θα δούμε πως, χρησιμοποιώντας τα παραπάνω εργαλεία προγραμματισμού και τις εμπειρίες που αποκτήθηκαν από την μελέτη τους, αναπτύχθηκε η εφαρμογή διαχείρισης αγροκτήματος.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 4: Εγκατάσταση και παραμετροποίηση απαιτούμενων εργαλείων

ΚΕΦΑΛΑΙΟ 5: Ανάπτυξη εφαρμογής FarmManager

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στο Λειτουργικό Σύστημα Android

1.1 Τι είναι το Android;

Το Android είναι ένα λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας, το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Τον Ιούλιο του 2005, η Google εξαγόρασε την Android Inc, μια μικρή εταιρεία με έδρα το Palo Alto στην California των ΗΠΑ. Οι συνιδρυτές της Android πήγαν να εργαστούν στην Google συμπεριλαμβανομένων των Andy Rubin (συν-ιδρυτής της Danger), Rich Miner (συν-ιδρυτής της Wildfire Communications, Inc), Nick Sears (πρώην αντιπρόεδρος της T-Mobile), και Chris White (επικεφαλής σχεδιασμού και ανάπτυξης interface στο WebTV). Εκείνη την εποχή ελάχιστα ήταν γνωστά για τις λειτουργίες της Android Inc, εκτός του ότι ανέπτυσσαν λογισμικό για κινητά τηλέφωνα. Αυτή ήταν η αρχή της φημολογίας περί σχεδίων της Google για να διεισδύσει στην αγορά κινητής τηλεφωνίας.

Στην Google, η ομάδα με επικεφαλής τον Rubin ανέπτυξε μια κινητή πλατφόρμα που στηρίζεται στον πυρήνα του Linux, την οποία προώθησαν με την παροχή ενός ευέλικτου, αναβαθμίσιμου συστήματος. Έχει αναφερθεί ότι η Google έχει ήδη συγκεντρώσει μια σειρά από εταίρους hardware και software και επισήμανε στους παρόχους ότι ήταν ανοικτή σε διάφορους βαθμούς συνεργασίας εκ μέρους της. Περισσότερες εικασίες ότι η Google θα εισέλθει στην αγορά κινητής τηλεφωνίας άρχισαν τον Δεκέμβριο του 2006. Δημοσιεύσεις από το BBC και τη The Wall Street Journal πληροφορούσαν ότι η Google ήθελε την έρευνα και τις εφαρμογές σε κινητά τηλέφωνα και εργάζονταν σκληρά για να τις προωθήσουν στην αγορά. Έντυπα και ηλεκτρονικά μέσα ενημέρωσης σύντομα ανέφεραν φήμες ότι η Google ανέπτυσσε μια Google-branded συσκευή. Περισσότερες φήμες ακολούθησαν, αναφέροντας ότι η Google καθόριζε τις τεχνικές προδιαγραφές και έδειχνε πρωτότυπα στους κατασκευαστές κινητών τηλεφώνων και τους φορείς δικτύων.

Τον Σεπτέμβριο του 2007, η InformationWeek κάλυψε μια μελέτη αξιολόγησης αναφέροντας ότι η Google έχει καταθέσει αρκετές πατέντες στον τομέα της κινητής τηλεφωνίας. Τελικά η Google παρουσίασε το smartphone της Nexus One που χρησιμοποιεί το open source λειτουργικό σύστημα Android. Η συσκευή κατασκευάστηκε από την HTC Corporation της Ταϊβάν, και έγινε διαθέσιμη στις 5 Ιανουαρίου 2010.

Εικόνα 1.1: Το Google Nexus One

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance, μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής hardware, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές κινητής τηλεφωνίας. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

Ενδεικτικά, αναφέρονται μερικά μέλη του οργανισμού αυτού [Εικόνα 1.2], για να γίνει ορατή η τεράστια προοπτική που δημιουργείται:

Εικόνα 1.2: Εταιρίες ανάπτυξης λογισμικού και κατασκευής υλικού παγκόσμιας εμβέλειας

Κατά τα λεγόμενά τους, όπως αναφέρεται και στο επίσημο site, η OHA (Open Handset Alliance) αντιπροσωπεύει: «Μια δέσμευση για την ειλικρίνεια, ένα κοινό όραμα για το μέλλον και συγκεκριμένα σχέδια για να κάνει το όραμα μια πραγματικότητα. Για να επιταχύνει την καινοτομία στο κινητό και να προσφέρει στους καταναλωτές μια πλουσιότερη, λιγότερο ακριβή και καλύτερη εμπειρία κινητών τηλεφώνων.»

http://www.openhandsetalliance.com/oha_faq.html

Η OHA ελπίζει να παραδώσει μια καλύτερη εμπειρία λογισμικού κινητών στους καταναλωτές, παρέχοντας μια πλατφόρμα για την ανάπτυξη καινοτόμων κινητών εφαρμογών γρηγορότερα και με υψηλότερη ποιότητα, χωρίς τέλη αδείας για τους προγραμματιστές λογισμικού ή τους κατασκευαστές κινητών τηλεφώνων. Η επιτυχία του Android ως πλατφόρμα κινητών τηλεφώνων θα εξαρτηθεί κατά ένα μεγάλο μέρος από την επιτυχία των συνεργατών της OHA στην κυκλοφορία επιθυμητών κινητών τηλεφώνων και κινητών υπηρεσιών που θα ενθαρρύνουν την υιοθέτηση των Android τηλεφώνων. Οι προγραμματιστές έχουν την ευκαιρία να δημιουργήσουν καινοτόμες, νέες εφαρμογές κινητών για Android ώστε να

ενθαρρυνθούν περισσότερες επιχειρήσεις κινητής τεχνολογίας να γίνουν μέλη της OHA.

Εικόνα 1.3: Λογότυπο πλατφόρμας Android

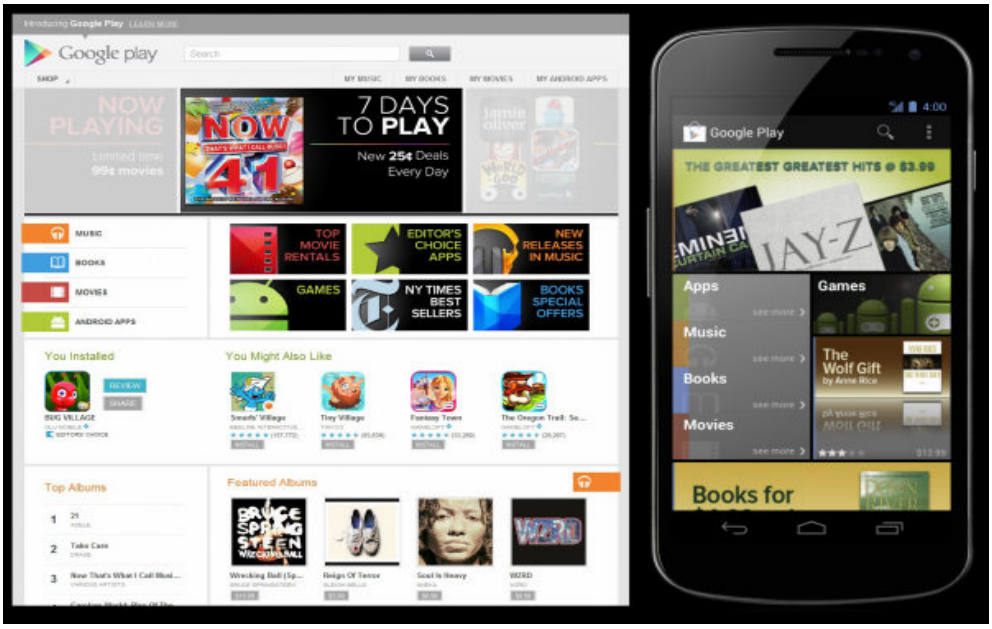
1.2 Χαρακτηρίστηκα Android

Λειτουργίες Οθόνης	Η πλατφόρμα είναι προσαρμόσιμη σε μεγαλύτερη ανάλυση (VGA), δισδιάστατες ψηφιακές γραφικές βιβλιοθήκες, τρισδιάστατα γραφικά βασισμένα στην OpenGL ES 1.0 έκδοση χαρακτηριστικών, καθώς και παραδοσιακές απεικονίσεις οθόνης "έξυπνων" συσκευών κινητής τηλεφωνίας.
Αποθήκευση Δεδομένων	Χρήση βάσης δεδομένων SQLite για τις ανάγκες αποθήκευσης
Συνδεσιμότητα	Το Android υποστηρίζει τεχνολογίες συνδεσιμότητας συμπεριλαμβανομένου GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth, και Wifi
Αποστολή μηνυμάτων	SMS και MMS είναι οι διαθέσιμοι τρόποι ανταλλαγής μηνυμάτων.

Περιήγηση στον Ιστό	Για την περιήγηση στον ιστό το Android διαθέτει ένα φυλλομετρητή βασισμένο στην ανοιχτή τεχνολογία WebKit.
Υποστήριξη Java	Λογισμικό γραμμένο στην Java είναι δυνατόν να μεταγλωττιστεί και να εκτελεστεί στην εικονική μηχανή Dalvik, η οποία είναι μια εξειδικευμένη υλοποίηση εικονική μηχανής, σχεδιασμένη για χρήση σε φορητές συσκευές, παρόλο που δεν είναι μια πρότυπη εικονική μηχανή Java.
Υποστήριξη Πολυμέσων	Το λειτουργικό Android υποστηρίζει τις ακόλουθα μορφές ήχου, στατικής και κινούμενης εικόνας: H.263, H.264 (σε 3GP ή MP4container), MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, MP3,MIDI, OGG Vorbis, WAV, JPEG, PNG, GIF, BMP.
Επιπλέον υποστήριξη hardware	Το λειτουργικό Android μπορεί να συνεργαστεί με κάμερες στατικής ή κινούμενης εικόνας, οθόνες αφής, GPS, αισθητήρες επιτάχυνσης, μαγνητόμετρα, δισδιάστατους καθώς και τρισδιάστατους επιταχυντές γραφικών.
Περιβάλλον Ανάπτυξης Λογισμικού	Περιλαμβάνει ένας προσομοιωτή συσκευής, εργαλεία για διόρθωση σφαλμάτων, μνήμη και εργαλεία ανάλυσης της απόδοσης του εκτελέσιμου λογισμικού καθώς και ένα επιπρόσθετο για το Eclipse IDE.
Αγορά και Εγκατάσταση Εφαρμογών	Παρόμοια με το App Store του iPhone OS, το Android Market είναι ένας κατάλογος εφαρμογών που μπορούν να μεταφορτωθούν και εγκατασταθούν στην συσκευή άμεσα μέσω ασύρματων καναλιών, χωρίς την χρήση υπολογιστή. Αρχικά μόνο δωρεάν εφαρμογές ήταν δυνατόν να εγκατασταθούν. Εφαρμογές επί πληρωμή ήταν μετέπειτα διαθέσιμες στο Android Market στις ΗΠΑ ύστερα από τις 19 Φεβρουαρίου 2009.
Οθόνη Αφής Πολλαπλών Σημείων	Το λειτουργικό Android είχε εξ ορισμού υποστήριξη για οθόνες πολλαπλών σημείων αλλά η δυνατότητα αυτή έχει κλειδωθεί σε επίπεδο πυρήνα (πιθανόν για αποφυγή παραβιάσεων των πατεντών λογισμικού της Apple στις τεχνολογίες οθονών αφής). Κυκλοφορεί μια ανεπίσημη τροποποίηση (mod) που έχει αναπτυχθεί για να υποστηρίζει πολλαπλή επαφή (multi-touch), αλλά απαιτεί δικαιώματα πρόσβασης υπερχρήστη (superuser) στη συσκευή για να γραφεί στη μνήμη flash ένας πυρήνας που να μην είναι υπογεγραμμένος (unsigned kernel).

1.3 Εφαρμογές Android

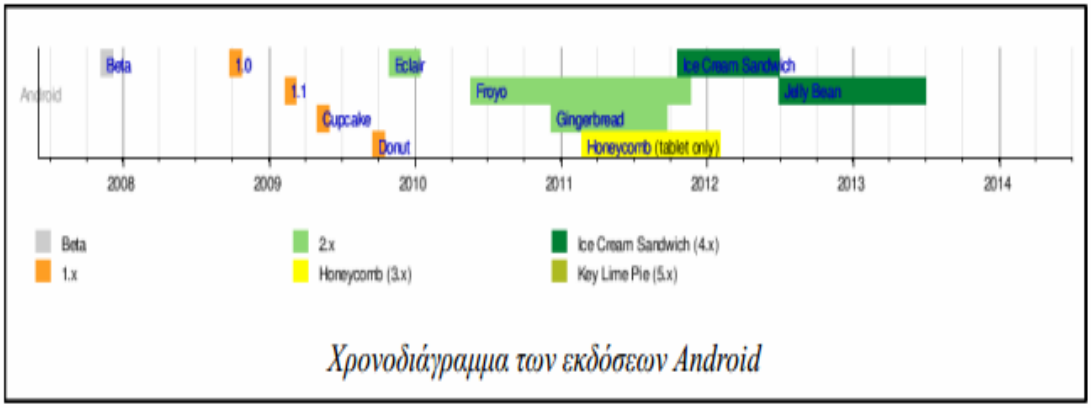
Το Android έχει μια μεγάλη κοινότητα προγραμματιστών που γράφουν εφαρμογές, οι οποίες επεκτείνουν τη λειτουργικότητα των συσκευών. Οι εφαρμογές γράφονται σε μια προσαρμοσμένη έκδοση της JAVA και μπορεί κάνεις να κατεβάσει από το online κατάστημα Google Play (πρώην Android Market) της Google όπως και από άλλα sites. Μέχρι τον Φεβρουάριο του 2013 περισσότερες από 850.000 εφαρμογές ήταν διαθέσιμες για Android ενώ εκτιμάτε ότι ο αριθμός των downloads από το Android Market μέχρι το Σεπτέμβριο του 2012 είχε υπερβεί τα 25 δισεκατομμύρια. Το Android είναι η πρώτη σε πωλήσεις παγκοσμίως πλατφόρμα για smartphones καθώς μέχρι το τελευταίο τρίμηνο του 2012 μετρούσε περισσότερες από 500 εκατομμύρια συσκευές σε χρήση και 1,3 εκατομμύρια ενεργοποιήσεις την ημέρα.



Εικόνα 1.4: Το Google Play σε κανονική και σε mobile έκδοση

1.4 Ιστορικά - Εκδόσεις και χαρακτηριστικά

Όπως αναφέραμε παραπάνω, το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα. Η εξέλιξη του λόγω της open source φύσης του είναι ραγδαία και αυτό αντικατοπτρίζεται στο γεγονός ότι οι 8 κύριες εκδόσεις του έχουν κυκλοφορήσει σε διάστημα 3.5 ετών, από τον Απρίλη του 2009 μέχρι τον Νοέμβριο του 2012.



Εικόνα 1.5:

Στην πληροφορική συνηθίζεται τα προϊόντα hardware και software να κυκλοφορούν εκτός από τον αριθμό έκδοσης τους, και με μία κωδική ονομασία. Η ονομασία αυτή μπορεί να είναι πχ ονόματα πόλεων (Windows Viena, Chicago), ονόματα ζώων (OSX Leopard, Lion), στην περίπτωση όμως του Android τα κώδικα ονόματα έρχονται στη μορφή γλυκού ή επιδόρπιου! Ακόμη, το όνομα της κάθε έκδοσης ξεκινά με το επόμενο γράμμα της Αγγλικής αλφαβήτου σε σχέση με την προηγούμενη έκδοση.

Η πρώτη έκδοση του Android SDK που εμφανίστηκε τον Νοέμβριο του 2007, χαρακτηρίστηκε από τους κατασκευαστές του σαν μια πρώτη ματιά στο SDK του Android, κάτι το οποίο πολλοί παράβλεψαν και βιάστηκαν να κατακρίνουν το Android σαν ένα προβληματικό σύστημα. Στην ουσία όμως το Android δεν παρουσίαζε προβλήματα τα οποία δεν παρουσιάζει οποιοδήποτε σύστημα σε τέτοια πρώιμη φάση. Έτσι το Σεπτέμβριο του 2008, η T-Mobile ανακοινώνει την διαθεσιμότητα του T-Mobile G1, του πρώτου έξυπνου τηλεφώνου (smartphone), βασισμένο στην πλατφόρμα του Android. Λίγες μέρες αργότερα (Οκτώβριο 2008), η Google ανακοινώνει την απελευθέρωση του SDK Release Candidate 1.0. Ακολούθησε τον Φεβρουάριο του 2009 η έκδοση 1.1 σαν μια ανανεωμένη έκδοση του 1.0. Μέχρι τότε το Android δεν υποστήριζε ακόμη την χρήση κουμπιών αφής, παρά μόνο την χρήση των κλασσικών ‘σκληρών’ κουμπιών της συσκευής.

1.4.1 Android 1.5 CUPCAKE

Τον Μάιο του 2009 εμφανίστηκε η έκδοση Android 1.5, ονόματι ‘Cupcake’ το λογότυπο της οποίας φαίνεται και στην Εικόνα 1.6 .

Εικόνα 1.6: Λογότυπο Android 1.5 CUPCAKE

Το ‘Cupcake’ βασίζεται στον Linux Kernel 2.6.27 και εισάγει κάποια καινούργια χαρακτηριστικά και ανανεώσεις στην διεπιφάνεια χρήστη (User Interface):

- Ικανότητα για καταγραφή και παρακολούθηση βίντεο μέσα από την λειτουργία της βιντεοκάμερας, μεταφόρτωση βίντεο στο YouTube και φωτογραφιών στο Picasa απευθείας από το τηλέφωνο, καινούργιο μαλακό πληκτρολόγιο (αφής) με πρόβλεψη κειμένου
- Υποστήριξη προτύπου Bluetooth A2DP και AVRCP
- Ικανότητα αυτόματης σύνδεσης σε μικροσυσκευή Bluetooth από μια συγκεκριμένη απόσταση
- Καινούργια widgets και φάκελοι που μπορούν να δημοσιευτούν στην αρχική οθόνη
- Κινούμενες μεταβάσεις οθόνης

1.4.2 Android 1.6 DONUT

Το ‘Donut’, Android 1.6, ήρθε τον Σεπτέμβριο του 2009.

Εικόνα 1.7: Λογότυπο Android 1.6 DONUT

Η έκδοση αυτή βασίζεται στον Linux Kernel 2.6.29 και εισάγει κάποια καινούργια χαρακτηριστικά όπως:

- Βελτιωμένο Android Market
- Ενσωματωμένη φωτογραφική μηχανή, βιντεοκάμερα και διεπαφή (interface) γκαλερί
- Η γκαλερί επιτρέπει πλέον στους χρήστες την επιλογή πολλαπλών φωτογραφιών προς διαγραφή
- Ανανεωμένη φωνητική αναζήτηση, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενείς (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας κλήσης επαφών
- Ανανεωμένη αναζήτηση με την δυνατότητα αναζήτησης σελιδοδεικτών, ιστορικού, επαφών και στο διαδίκτυο από την αρχική οθόνη
- Ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και με μηχανή μετατροπής κειμένου σε ομιλία (text-to-speech)
- Υποστήριξη για ανάλυση οθονών WVGA
- Βελτιώσεις στην ταχύτητα αναζήτησης και των εφαρμογών της φωτογραφικής μηχανής.

1.4.3 Android 2.0/2.1 ECLAIR

Ακολουθεί το 'Eclair', Android 2.0 τον Νοέμβριο 2009, με τις επανεκδόσεις του σε Android 2.0.1 τον Δεκέμβριο 2009 (Eclair 0.1) και τον Ιανουάριο 2010 με το Android 2.1 (Eclair MR1).

Εικόνα 1.8: Λογότυπο Android 2.1 Éclair

Βασίζεται και αυτή στον Linux Kernel 2.6.29. Ανάμεσα στις άλλες αλλαγές είναι και:

- Βέλτιστη ταχύτητα υλικού
- Υποστήριξη για περισσότερες οθόνες και αναλύσεις
- Βελτιωμένη διεπιφάνεια χρήστη
- Καινούργια διεπιφάνεια χρήσης για την μηχανή αναζήτησης και υποστήριξη του προτύπου HTML5
- Καινούργιες λίστες επαφών
- Καλύτερος λόγος άσπρου – μαύρου για φόντα
- Βελτιωμένοι χάρτες Google (google maps) 3.1.2
- Υποστήριξη Microsoft Exchange

- Ενσωματωμένη υποστήριξη flash για την Camera
- Ψηφιακή μεγέθυνση (zoom)
- Κλάση Motion Event βελτιωμένη ώστε οι κατασκευαστές να μπορούν να παρακολουθούν αποτελεσματικότερα τα γεγονότα πολλαπλής αφής
- Ανανεωμένο εικονικό πληκτρολόγιο
- Bluetooth 2.1

1.4.4 Android 2.2 FROYO

Ακολουθεί το Android 2.2 με το όνομα ‘Froyo’ τον Μάιο του 2010.

Εικόνα 1.9: Λογότυπο Android 2.2 Froyo

Η έκδοση FROYO βασίζεται στον Linux Kernel 2.6.32 και ανάμεσα σε άλλες αλλαγές, περιλαμβάνει:

- Βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση
- Ενσωμάτωση στην μηχανή αναζήτησης, της μηχανής JavaScript του Chrome V8
- Αυξημένη υποστήριξη Microsoft Exchange (σε πολιτικές ασφαλείας, συγχρονισμού ημερολογίου , auto – discovery, GAL look-up, remote wipe)
- Βελτιωμένος προωθητής εφαρμογής (application launcher), με συντομεύσεις προς τις εφαρμογές τηλεφώνου και εφαρμογές της Μηχανής Αναζήτησης
- Σύνδεση USB και λειτουργία δυναμικής ζώνης (hotspot) Wi-Fi • Ανανεωμένη εφαρμογή Αγοράς (Market) με αυτόματη ανανέωση
- Επιλογή για απαγόρευση πρόσβασης δεδομένων μέσω ενός δικτύου κινητής τηλεφωνίας
- Γρήγορη εναλλαγή ανάμεσα σε πολλαπλές γλώσσες του πληκτρολογίου και των λεξικών τους

- Φωνητική κλήση και διαμοιρασμός επαφών με Bluetooth
- Υποστήριξη για αριθμητικούς και αλφαριθμητικούς κωδικούς
- Η μηχανή αναζήτησης μπορεί να αποτυπώσει κινούμενα GIFs
- Υποστήριξη για πεδία μεταφόρτωσης αρχείων στην μηχανή αναζήτησης.
- Υποστήριξη για εγκατάσταση εφαρμογών στην επεκτάσιμη μνήμη
- Υποστήριξη Adobe Flash 10.1

1.4.5 Android 2.3 GINGERBREAD

Η επόμενη έκδοση κυκλοφόρησε πρώτη φορά το Δεκέμβριο του 2010 με πολλές επανακυκλοφορήσεις ως τον Σεπτέμβριο του επόμενου έτους έχοντας τον αριθμό 2.3.7.

Εικόνα 1.10: Λογότυπο Android 2.3 Gingerbread

Αυτή η έκδοση είναι η πιο διαδεδομένη μεταξύ όλων των εκδόσεων του Android με ποσοστό περίπου 55%. Βασίζεται στον ο Linux Kernel 2.6.35.7. Μερικά από τα χαρακτηριστικά της έκδοσης αυτής είναι :

- Επιλογή λέξεων και αντιγραφή επικόλληση με ένα άγγιγμα.
- Ενημερωμένο UI Design και υποστήριξη για πολύ μεγάλα μεγέθη οθονών και αναλύσεων (WXGA και μεγαλύτερες).
- Επανασχεδιασμένο Multi-touch πληκτρολόγιο.

- Αυξημένη υποστήριξη για development
- Download Manager για κατέβασμα μεγάλων αρχείων.
- Βελτιωμένη ενεργειακή διαχείριση.
- Υποστήριξη NFC (Near Field Communication).
- Υποστήριξη πρωτοκόλλου SIP για video κλήσεις.
- Υποστήριξη του πρωτόκολλου WebM για αναπαραγωγή video.
- Υποστήριξη πολλαπλών καμερών (πχ. μπροστά – πίσω).
- Υποστήριξη για βαρόμετρο, γυροσκόπιο, επιταχυνσιόμετρο και άλλους αισθητήρες.
- Μετάβαση από το σύστημα αρχείων YAFFS στο Ext4 στις νέες συσκευές.

1.4.6 Android 3.0 HONEYCOMB

Στη συνέχεια στις 9 Μαΐου του 2011 παρουσιάστηκε η έκδοση Android 3.0 ‘Honeycomb’

Εικόνα 1.11: Λογότυπο Android 3.0 Honeycomb

Η έκδοση αυτή είχε την ιδιαιτερότητα ότι προοριζόταν αποκλειστικά για tablets. Βασίζεται στον Linux Kernel 2.6.36 και οι αλλαγές που έγιναν σε αυτή την έκδοση είναι:

- Υποστήριξη διπύρηνων και τετραπύρηνων επεξεργαστών.
- Νέο, εντελώς διαφορετικό, user interface και widgets.
- Απλοποιημένο multitasking ώστε ο χρήστης να μπορεί με την χρήση ενός πλήκτρου να περνάει από μια εφαρμογή σε άλλη
- Βελτιωμένη υποστήριξη των ταμπλετών.
- Επανασχεδιασμένο Multi-touch πληκτρολόγιο.
- Υποστήριξη usb port για σύνδεση περιφερειακών.
- Ανάπτυξη λογισμικού (scripting) για 3D, σε γλώσσα η οποία καλείται "Renderscript".
- Video chat μέσω Google Talk.

- Ανάγνωση βιβλίων μέσω Google eBooks.
- "Ιδιωτική περιήγηση".

1.4.7 Android 4.0 ICE CREAM SANDWICH

Ακολουθεί η έκδοση ‘Ice Cream Sandwich’ τον Οκτώβριο του 2011 η οποία αποτελεί την προσπάθεια της εταιρίας για ένα ενιαίο λειτουργικό σύστημα για smartphones και tablets. Και σε αυτήν την έκδοση κυκλοφόρησαν κάποιες επανεκδόσεις με την τελική να είναι η 4.0.4

Εικόνα 1.12: Λογότυπο Android 4.0 Ice Cream Sandwich

Για άλλη μια φορά έχουν βελτιωθεί η ταχύτητα και η απόδοση του συστήματος. Βασίζεται στον πυρήνα Linux 3.0.1 και μερικές από τις αλλαγές που φέρνει αυτή η έκδοση είναι:

- Ανανεωμένο UI και ύπαρξη εικονικών πλήκτρων τα οποία παίρνουν την θέση των φυσικών ή αφής πλήκτρων τα οποία υπήρχαν στις συσκευές.
- Βελτιωμένη ασφάλεια του συστήματος.
- Προσθήκη ξεκλειδώματος συσκευής με αναγνώριση προσώπου.
- Ο browser μπορεί να ανοίξει μέχρι και 16 καρτέλες ταυτόχρονα.
- Δυνατότητα τερματισμού εφαρμογών που τρέχουν στο background.
- Καταγραφή συνόλου κίνησης δεδομένων και δυνατότητα εισαγωγής ορίου.

- Ύπαρξη Android Beam που σε συνεργασία με το NFC μπορεί να γίνει αποστολή δεδομένων ανάμεσα σε συσκευές σε κοντινή απόσταση.
- Ύπαρξη wi-fi direct για επικοινωνία μεταξύ συσκευών χωρίς την ύπαρξη access point.
- Βελτιωμένη εφαρμογή κάμερας/βίντεο και υποστήριξη εγγραφής βίντεο σε ανάλυση 1080p.
- Βελτιωμένες ειδοποιήσεις εφαρμογών.

1.4.8 Android 4.1/4.2 JELLY BEAN

Η τελευταία έκδοση, μέχρι τον Μάρτιο του 2013, του λειτουργικού Android είναι η 'Jelly Bean'. Κυκλοφόρησε αρχικά τον Ιούλιο του 2012 στην έκδοση 4.1 και αργότερα επανεκδόθηκε στην έκδοση 4.2.

Εικόνα 1.13: Λογότυπο Android 4.1/4.2 Jelly Bean

Και αυτή η έκδοση προσφέρει ταχύτερη απόκριση του συστήματος από την προηγούμενη έκδοση. Βασίζεται στον πυρήνα Linux 3.0.31 και μερικά από τα χαρακτηριστικά που προσφέρει είναι:

- Επανασχεδιασμένο UI και widgets για χρήση σε tablets και σε smartphones.
- Βελτιωμένος NFC controller.
- Βελτίωση της μηχανής εκτέλεσης των εφαρμογών για περισσότερη ασφάλεια.
- Νέες δυνατότητες μέσω της κάμερας.

- Βελτιωμένες λειτουργίες για τους προγραμματιστές.
- Νέα μηχανή Renderscript για καλύτερα γραφικά.
- Υποστήριξη προβολής σε εξωτερική οθόνη μέσω wi-fi.
- Προσθήκη νέων λειτουργιών κλειδώματος οθόνης.
- Βελτιωμένη φωνητική αναζήτηση.
- Επανασχεδιασμένες ειδοποιήσεις των εφαρμογών.

1.4.9 Android 5.0 Key Lime Pie (?)

Τα επόμενα smartphones και tablets θα τρέχουν την έκδοση του λειτουργικού συστήματος Android 5.0 Key Lime Pie. Αυτό επιβεβαιώνεται τουλάχιστον σαν κωδική ονομασία από υπάλληλο της Google που δημιούργησε ένα ενδιαφέρον γράφημα με την εξέλιξη των εκδόσεων του λειτουργικού συστήματος Android. Τον Μάιο θα έχουμε το ετήσιο event Google I/O 2013 και μέχρι τότε κάθε εβδομάδα θα προσθέτονται και νέες πληροφορίες στο πάζλ.

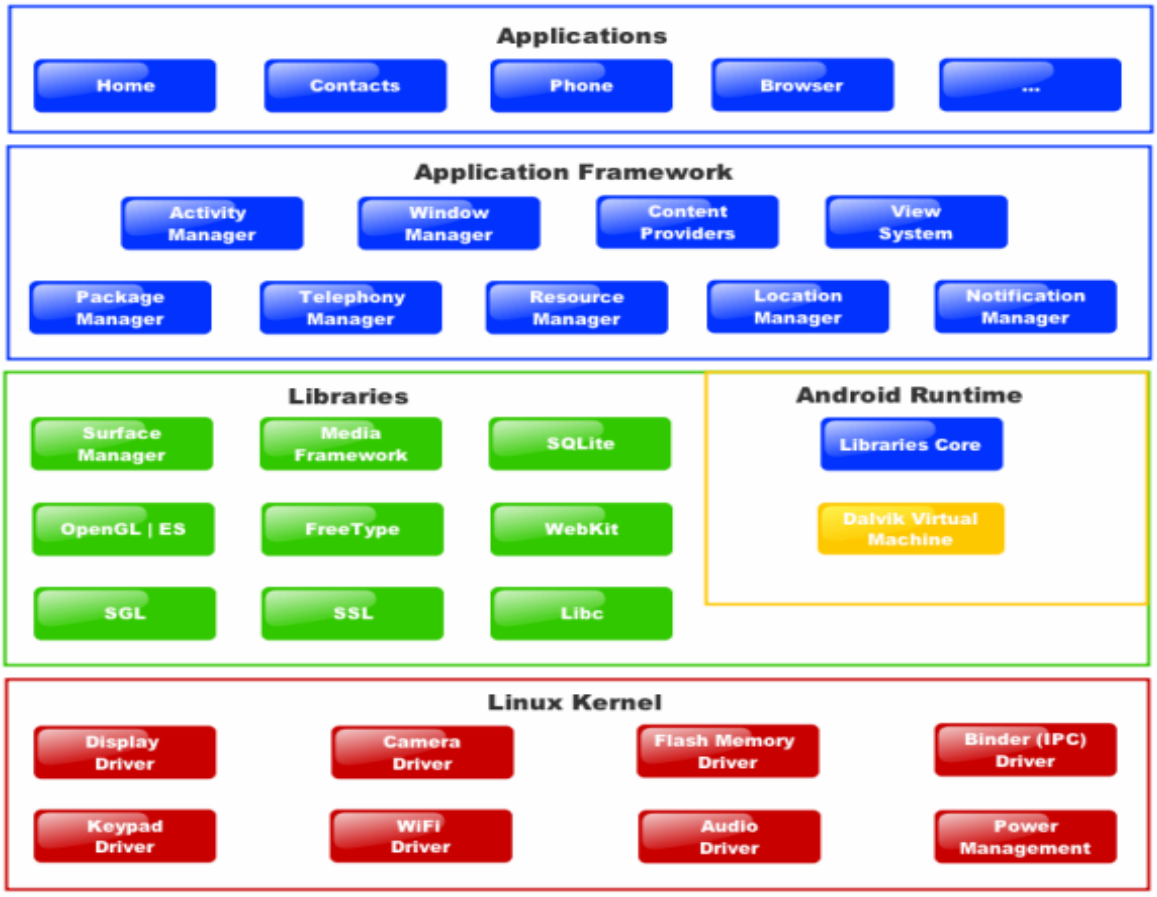
1.5 Αρχιτεκτονική του Android

Το Android δεν είναι μόνο ένα λειτουργικό σύστημα, είναι μια στοίβα λογισμικού. Η λογική πίσω από αυτήν την έκφραση και σε όλη την φιλοσοφία του Android, κρύβεται στο ακόλουθο διάγραμμα με τα βασικά συστατικά του [Εικόνα 1.14].

Η αρχιτεκτονική του Android ενθαρρύνει την έννοια επαναχρησιμοποίησης κώδικα, που επιτρέπει να δημοσιευτούν και να μοιραστούν δραστηριότητες, υπηρεσίες και δεδομένα με άλλες εφαρμογές με πρόσβαση που ελέγχεται από τους περιορισμούς ασφάλειας που έχουν τεθεί. Έτσι γίνεται δυνατή η επέκταση και βελτιστοποίηση υπαρχόντων εφαρμογών ή δημιουργίας καινούργιων, χρησιμοποιώντας κώδικα από αυτές. Οι ακόλουθες υπηρεσίες εφαρμογών είναι οι ακρογωνιαίοι λίθοι της αρχιτεκτονικής όλων των εφαρμογών Android, παρέχοντας το πλαίσιο που θα χρησιμοποιηθεί για τη δημιουργία του λογισμικού :

- Διαχειριστής δραστηριοτήτων (Activity Manager) - Ελέγχει τον κύκλο ζωής των δραστηριοτήτων, συμπεριλαμβανομένης της διαχείρισης του σωρού δραστηριοτήτων.
- Όψεις (Views) - Χρησιμοποιούνται για να κατασκευάσουν διεπαφές χρήστη (user interface) για τις δραστηριότητες (activities).
- Διαχειριστή Ειδοποιήσεων (Notification Manager) - Παρέχει ένα συνεπή και αποτελεσματικό μηχανισμό για να προειδοποιεί τον χρήστη.
- Παροχέας Περιερχομένου (Content Providers) - Επιτρέπουν στις εφαρμογές να μμοιραστούν δεδομένα μεταξύ τους.

- Διαχειριστής Πόρων (Resource Manager) - Υποστηρίζει πόρους εκτός κώδικα όπως συμβολοσειρές και γραφικά.



Εικόνα 1.14: Τα βασικά περιεχόμενα του λειτουργικού συστήματος Android

Από ότι βλέπουμε λοιπόν η αρχιτεκτονική του λειτουργικού συστήματος αποτελείτε από 5 βασικά επίπεδα.

- Τον πυρήνα Linux (Linux Kernel)
- Τις εγγενείς και τις προηγμένες βιβλιοθήκες (Libraries)
- Την εικονική μηχανή Dalvik (Dalvik VM)
- Τον χρόνο εκτέλεσης (Android Runtime)
- Το πλαίσιο εφαρμογής (Application Framework)

Ακολούθως θα περιγράψουμε συνοπτικά τα βασικά αυτά επίπεδα χωρίς να μπούμε σε λεπτομέρειες για όλα τα περιεχόμενα του κάθε επιπέδου. Αν ο αναγνώστης επιθυμεί να μάθει περισσότερα, μπορεί να επισκεφθεί την επίσημη ιστοσελίδα του Android για κατασκευαστές (http://developer.android.com). Κάθε επίπεδο στην αρχιτεκτονική αυτή, χρησιμοποιεί τις υπηρεσίες που του προσφέρονται

από τα πιο πάνω επίπεδα. Ας δούμε τώρα αυτά τα επίπεδα ξεκινώντας από το πιο χαμηλό.

1.5.1 Η εικονική μηχανή Dalvik

Ένα από τα στοιχεία κλειδιά του Android είναι η εικονική μηχανή Dalvik. Το Android χρησιμοποιεί την δικιά του εικονική μηχανή και όχι μια παραδοσιακή, με σκοπό να εξασφαλίσει ότι πολλαπλά στιγμιότυπα τρέχουν αποτελεσματικά σε μια ενιαία συσκευή.

Η Dalvik VM (Virtual Machine) χρησιμοποιεί τον πυρήνα Linux της συσκευής για να χειριστεί τις χαμηλού επιπέδου λειτουργίες που περιλαμβάνουν την ασφάλεια, τον πολυνηματισμό και τη διαχείριση διαδικασιών και μνήμης. Είναι επίσης δυνατό να γραφτούν εφαρμογές C/C++ που τρέχουν άμεσα στο εσωτερικό του λειτουργικού Linux. Αν και μπορεί να γίνει αυτό, στις περισσότερες περιπτώσεις δεν υπάρχει κανένας λόγος.

Μέσω της Dalvik VM επιτυγχάνεται η ρύθμιση της πρόσβασης στο υλικό και στις υπηρεσίες του συστήματος. Με τη χρησιμοποίηση αυτής της εικονικής μηχανής στην εκτέλεση εφαρμογής, η οποία προσφέρει ένα αφαιρετικό στρώμα, οι κατασκευαστές δεν χρειάζεται να ανησυχήσουν για κάποια υλοποίηση υλικού (hardware implementation).

Η Dalvik VM εκτελεί τα Dalvik εκτελέσιμα αρχεία, ένα format βελτιστοποιημένο έτσι ώστε να καταλαμβάνει την ελάχιστη μνήμη. Τα .dex εκτελέσιμα αρχεία δημιουργούνται μετασχηματίζοντας κλάσεις που έχουν μεταγλωττιστεί από Java χρησιμοποιώντας εργαλεία που παρέχονται μέσα στο SDK.

Μια απλή Java VM είναι μια εικονική μηχανή βασισμένη σε στοίβα (stack-based). Η Dalvik VM από την άλλη είναι μια εικονική μηχανή βασισμένη σε μητρώα (registerbased). Με τον τρόπο αυτό αυξάνεται η αποδοτικότητα του επεξεργαστή του κινητού. Επίσης, οι εικονικές μηχανές που είναι βασισμένες σε καταχωρητές (registers) επιτρέπουν ταχύτερους χρόνους εκτέλεσης των μεγάλων προσραμμάτων.

1.5.2 Πυρήνας Linux (Linux kernel)

Η βάση της στοίβας λογισμικού του Android είναι ο πυρήνας Linux, ο οποίος είναι δοκιμασμένος, σταθερός και πετυχημένος και μπορεί να βρεθεί παντού, από ρολόγια χειρός μέχρι υπερυπολογιστές. Ο τροποποιημένος πυρήνας του συστήματος

βασίζεται στην έκδοση 2.6 (και στην έκδοση 3.0.1 για το Android 4.0) του Linux Kernel, η οποία υποστηρίζει όλες τις κύριες λειτουργίες του λειτουργικού συστήματος. Οι λειτουργίες αυτές αφορούν διαχείριση μνήμης, διαχείριση διεργασιών, λειτουργίες δικτύου, ασφάλεια του λειτουργικού, και ένα σύνολο οδηγών υλικού (hardware drivers). Οι οδηγοί αυτοί είναι υπεύθυνοι για την επικοινωνία του software με το hardware της συσκευής. Ενδεικτικά ο πυρήνας του Android περιέχει:

- Οδηγό προβολής οθόνης
- Οδηγό Wifi και Bluetooth
- Οδηγό κάμερας
- κλπ

Ο πυρήνας του Android μπορεί να βασίζεται στον πυρήνα του Linux, αλλά διαφέρει αρκετά από αυτόν. Ο λόγος είναι οι αλλαγές στην αρχιτεκτονική που έχει κάνει η Google για να είναι ελαφρύτερος και βελτιστοποιημένος για χρήση σε κινητές συσκευές. Αυτό σημαίνει ότι παρότι το Android είναι κατά βάση Linux, επί της ουσίας είναι αρκετά δύσκολο να τρέξουν εφαρμογές ή να χρησιμοποιηθούν βιβλιοθήκες από τη μία πλατφόρμα στην άλλη. Ο Linus Torvalds έχει αναφέρει ότι τελικά στο μέλλον το Android και το Linux θα μοιράζονται έναν κοινό πυρήνα, αλλά αυτό θα αργήσει 4-5 χρόνια ακόμα.

1.5.3 Εγγενείς Βιβλιοθήκες (Native Libraries)

Στο αμέσως υψηλότερο επίπεδο βρίσκουμε τις Native Libraries – Εγγενείς Βιβλιοθήκες. Όλες αυτές είναι γραμμένες στην γλώσσα προγραμματισμού C και C++ και μεταγλωττίστηκαν για την συγκεκριμένη αρχιτεκτονική υλικού που χρησιμοποιείται από το τηλέφωνο. Οι βιβλιοθήκες αυτές δεν είναι εφαρμογές που μπορούν να στηθούν από μόνες τους. Υπάρχουν για να μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Από την έκδοση Donut και μετά, οι κατασκευαστές μπορούν να γράφουν τις δικές τους τέτοιες βιβλιοθήκες με την χρήση της Εργαλειοθήκης NDK (Native Development Kit).

1.5.3.1 Βιβλιοθήκες Android

Το Android προσφέρει διάφορα APIs για την ανάπτυξη εφαρμογών. Ο ακόλουθος κατάλογος APIs πυρήνα δίνει μια ιδέα για το τι είναι διαθέσιμο. Όλες οι συσκευές Android υποστηρίζουν το λιγότερο αυτά τα APIs :

- android.util - Το πακέτο βοηθήματος (utility) πυρήνων περιέχει χαμηλού επιπέδου κλάσεις, όπως τα εξειδικευμένα containers, μορφοποιητές (formatters) συμβολοσειρών, και βοηθήματα για parsing XML αρχείων.
- android.os - Το πακέτο λειτουργικού συστήματος παρέχει την πρόσβαση στις βασικές υπηρεσίες του λειτουργικού συστήματος όπως τη διαβίβαση μηνυμάτων, την επικοινωνία μεταξύ των επικοινωνιών, τις λειτουργίες ρολογιών, και την αποσφαλμάτωση (debugging).
- android.graphics - Το API γραφικών παρέχει τις χαμηλού επιπέδου κλάσεις γραφικών που υποστηρίζουν τους καμβάδες (canvases), τα χρώματα, και τα βασικά στοιχεία ζωγραφικής, και παρέχει τη δυνατότητα να ζωγραφίσει κανείς πάνω σε καμβά (canvas).
- android.text - Τα εργαλεία επεξεργασίας κειμένων για αναπαράσταση και ανάλυση κειμένου.
- android.database - Παρέχει τις χαμηλού επιπέδου κλάσεις που απαιτούνται για το χειρισμό των δρομέων (cursors) κατά τη λειτουργία τους με τις βάσεις δεδομένων
- android.content - Το API περιεχομένου (Content) χρησιμοποιείται για να διαχειριστεί την πρόσβαση στα δεδομένα και την έκδοση τους, παρέχοντας υπηρεσίες για την διαχείριση των πόρων, των Παρόχων Περιεχομένου (Content Providers), και των πακέτων.
- android.view – Οι Όψεις (Views) είναι οι κλάσεις πυρήνα διεπαφών χρήστη. Όλες οι διεπαφές χρήστη κατασκευάζονται χρησιμοποιώντας μια σειρά από Όψεις που παρέχουν τα συστατικά της αλληλεπίδρασης χρηστών.
- android.widget - Τοποθετημένες στο πακέτο View, οι κλάσεις widget είναι τα στοιχεία της διεπαφής χρήστη (user-interface) που χρησιμοποιούνται στις εφαρμογές. Περιλαμβάνουν τις λίστες, τα κουμπιά, και τις διατάξεις (layouts).
- com.google.android.maps - Ένα υψηλού επιπέδου API που παρέχει πρόσβαση στις εγγενείς λειτουργίες χαρτών που μπορούν να χρησιμοποιηθούν μέσα σε εφαρμογές. Περιλαμβάνει τη λειτουργία MapView, καθώς επίσης και τις κλάσεις Overlay και MapController που χρησιμοποιούνται για να διαχειριστούν και να ελέγξουν τους ενσωματωμένους χάρτες.

- android.app - Ένα πακέτο υψηλού επιπέδου που παρέχει πρόσβαση στο μοντέλο εφαρμογών. Το πακέτο εφαρμογών περιλαμβάνει τα Activity και Service APIs που αποτελούν τη βάση για όλες τις Android εφαρμογές.
- android.provider - Για να διευκολύνει την πρόσβαση των προγραμματιστών σε ορισμένους τυποποιημένους Παρόχους Περιεχομένου (Content Providers) (όπως η βάση δεδομένων επαφών), προσφέρονται κλάσεις που παρέχουν πρόσβαση σε τυποποιημένες βάσεις δεδομένων που περιλαμβάνονται σε όλες τις εκδόσεις Android.
- android.telephony - Τα APIs τηλεφωνίας επιτρέπουν την άμεση αλληλεπίδραση με το τηλεφωνικό σωρό (phone stack) της συσκευής, δίνοντας τη δυνατότητα, να γίνει λήψη, και έλεγχος των τηλεφωνημάτων, της κατάστασης του τηλεφώνου, και των μηνυμάτων SMS.
- android.webkit - Το πακέτο WebKit περιλαμβάνει APIs για την εργασία με περιεχόμενο βασισμένο στο διαδίκτυο, συμπεριλαμβανομένης της λειτουργίας wampee για την ενσωμάτωση μηχανών αναζήτησης και ενός διαχειριστή για cookies. Εκτός από το Android APIs, ο σωρός του Android περιλαμβάνει ένα σύνολο βιβλιοθηκών C/C++. Αυτές οι βιβλιοθήκες περιλαμβάνουν:
    - *OpenGL* - Βιβλιοθήκη που χρησιμοποιείται για να υποστηρίξει 3D γραφικά βασισμένη στο Open GL ES 1.0 API.
    - *FreeType* - Υποστήριξη για τη bitmap και τη vector απόδοση γραμματοσειράς.
    - *SGL* - Βιβλιοθήκη πυρήνων που χρησιμοποιείται για να παρέχει μια μηχανή 2D γραφικών.
    - *libc* - Τυποποιημένη βιβλιοθήκη C που βελτιστοποιείται για συσκευές βασισμένες σε Linux.
    - *SQLite* - Ελαφριά μηχανή σχεσιακών βάσεων δεδομένων που χρησιμοποιείται για να αποθηκεύσει δεδομένα εφαρμογών.
    - *SSL* - Υποστήριξη για χρήση των Secure Sockets Layer του κρυπτογραφικού πρωτοκόλλου για ασφαλείς επικοινωνίες μέσω διαδικτύου.

1.5.3.2 Προηγμένες Βιβλιοθήκες Android

Οι βιβλιοθήκες πυρήνα παρέχουν όλες τις λειτουργίες που είναι απαραίτητες για να ξεκινήσει η δημιουργία μιας εφαρμογής για Android, αλλά οι προηγμένες βιβλιοθήκες APIs προσφέρουν πραγματικά συναρπαστικές λειτουργίες.
Επειδή το Android στοχεύει να χρησιμοποιείται από ένα ευρύ φάσμα κινητών, η καταλληλότητα και η εφαρμογή των ακόλουθων APIs ποικίλουν ανάλογα με τη συσκευή επάνω στην οποία εφαρμόζονται.

- android.location - Το τοποκεντρικό API (location-based) δίνει πρόσβαση των εφαρμογών σε πληροφορίες για τη τρέχουσα φυσική θέση της συσκευής. Οι υπηρεσίες βασισμένες στη τοποθεσία παρέχουν πρόσβαση σε πληροφορίες τοποθεσίας χρησιμοποιώντας οποιοδήποτε υλικό ή τεχνολογία είναι διαθέσιμη στη συσκευή και βοηθούν στον προσδιορισμό θέσης.
- android.media - Τα APIs πολυμέσων παρέχουν υποστήριξη για την αναπαραγωγή και εγγραφή αρχείων ήχου και βίντεο, συμπεριλαμβανομένων των πολυμέσων ροής (streamed).
- android.opengl - Το Android προσφέρει μια ισχυρή αποδοτική 3D μηχανή που χρησιμοποιεί το OpenGL ES API με το οποίο δημιουργούνται 3D διεπαφές χρήστη για τις εφαρμογές.
- android.hardware - Όπου είναι διαθέσιμο, το API υλικού εκθέτει υλικό αισθητήρων συμπεριλαμβανομένης της φωτογραφικής μηχανής, του επιταχύμετρου (accelerometer) και των αισθητήρων πυξίδας.
- android.bluetooth, android.net.wifi, και android.telephony - Το Android επίσης παρέχει χαμηλού επιπέδου πρόσβαση στην πλατφόρμα υλικού, συμπεριλαμβανομένου του Bluetooth, του WI-FI, και του υλικού τηλεφωνίας.

1.5.4 Χρόνος Εκτέλεσης (Android Runtime)

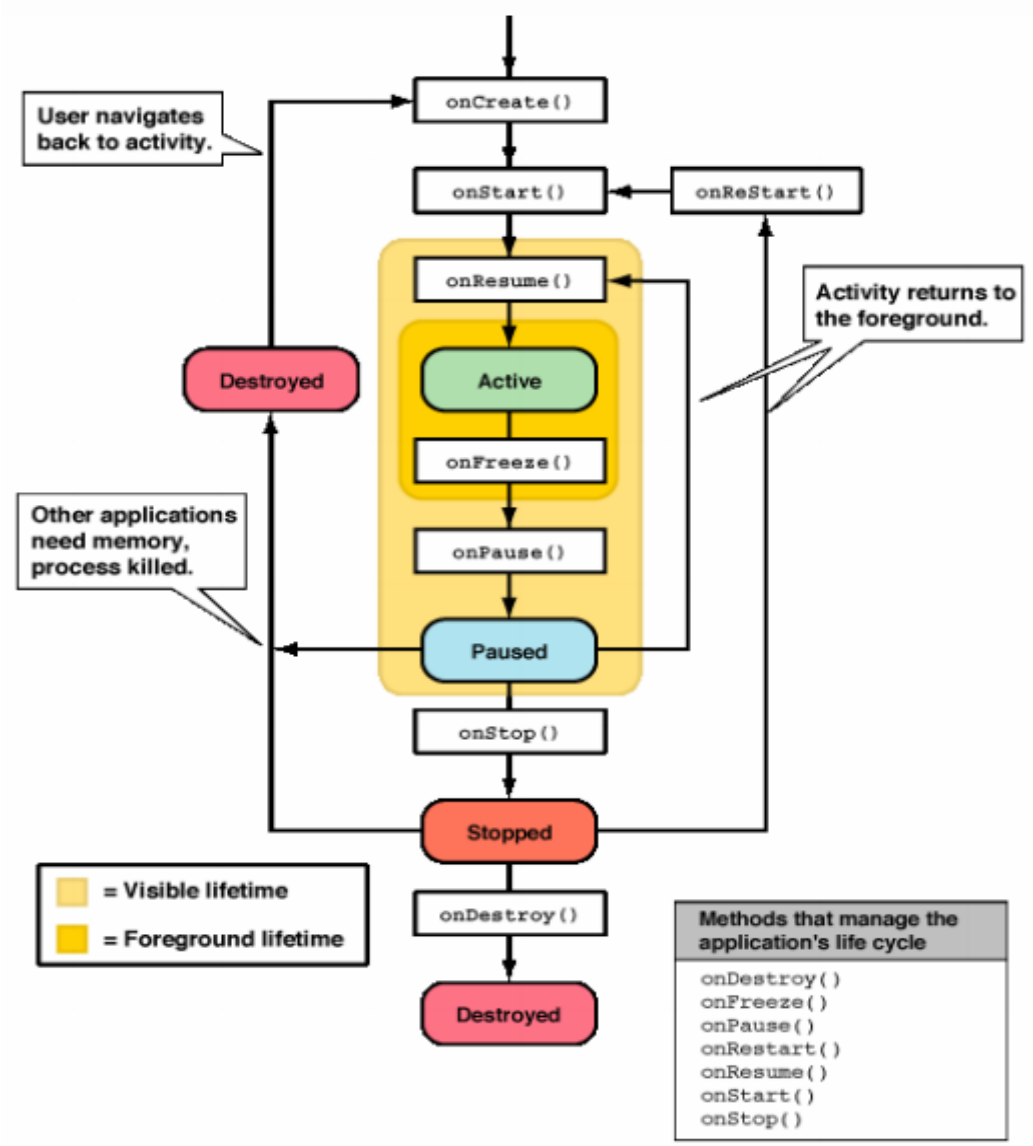
Στο ίδιο επίπεδο με τις εγγενείς βιβλιοθήκες, βρίσκουμε και τον χρόνο εκτέλεσης Android. Εδώ υπάρχουν οι βασικές βιβλιοθήκες της Java και η εικονική μηχανή Dalvik. Η Dalvik είναι μια βελτιστοποιημένη υλοποίηση μιας εικονικής μηχανής Java για φορητές συσκευές από την Google. Η Dalvik τρέχει .dex αρχεία, τα οποία είναι bytecodes που προέρχονται από αρχεία .class και .jar. Εν αντιθέσει όμως με τα .class αρχεία, τα .dex είναι πολύ πιο συμπαγή και αποδοτικά, γεγονός σημαντικό για συσκευές με περιορισμένη μνήμη και μπαταρία. Το Android περιλαμβάνει ένα σύνολο βασικών βιβλιοθηκών που παρέχουν τις περισσότερες από

τις διαθέσιμες λειτουργίες των βασικών βιβλιοθηκών της Java. Κάποια πακέτα και κλάσεις υπάρχουν και στο Android, κάποια άλλα δεν υποστηρίζονται καθόλου, ενώ ταυτόχρονα το Android παρέχει και επιπρόσθετα προσαρμοσμένα στις δικές του ανάγκες.

1.5.5 Πλαίσιο Εφαρμογής (Application Framework)

Πάνω από τις εγγενείς βιβλιοθήκες και το χρόνο εκτέλεσης Android, είναι το πλαίσιο εφαρμογής. Αυτό το επίπεδο παρέχει υψηλού επιπέδου δομικές μονάδες τις οποίες μπορούμε να χρησιμοποιούμε για την κατασκευή των εφαρμογών. Αυτό το πλαίσιο είναι προ-εγκατεστημένο στο Android, αλλά είναι επεκτάσιμο, αφού ο κάθε κατασκευαστής μπορεί να το συμπληρώσει με δικά του κομμάτια. Τα σημαντικότερα δομικά στοιχεία του πλαισίου αυτού είναι:

- Διαχειριστής δραστηριοτήτων - Activity Manager: Υπεύθυνος για τον έλεγχο του χρόνου ζωής των εφαρμογών και για την διατήρηση μιας στοίβας που επιτρέπει την πλοήγηση του χρήστη σε προηγούμενες οθόνες.
- Παροχέας Περιεχομένου - Content Providers: Αυτά τα αντικείμενα περιέχουν δεδομένα που μπορούν να διαμοιραστούν μεταξύ εφαρμογών.
- Διαχειριστής Πόρων - Resource Manager: Οι πόροι είναι οτιδήποτε υπάρχει σε ένα πρόγραμμα και δεν είναι κώδικας. Για παράδειγμα μπορεί να είναι κωδικοί χρωμάτων, αλφαριθμητικοί χαρακτήρες ή ακόμα και έτοιμα σχεδιαγράμματα οθονών φτιαγμένα σε XML, τα οποία μπορεί το πρόγραμμα να καλεί.
- Διαχειριστής τοποθεσίας - Location Manager: Χρησιμοποιείται για να μπορεί να ξέρει το τηλέφωνο που βρίσκεται ανά πάσα στιγμή.
- Διαχειριστής Κοινοποιήσεων - Notification Manager: Ιδανικός τρόπος για την ενημέρωση του χρήστη για γεγονότα που συμβαίνουν, διακριτικά χωρίς να διακόπτεται η εργασία του.



Εικόνα 1.15: Ο κύκλος ζωής μιας δραστηριότητας Android

ΚΕΦΑΛΑΙΟ 2

Εργαλεία και προκλήσεις ανάπτυξης εφαρμογών στο Android

2.1 Κύκλος Ανάπτυξης Εφαρμογής

Η ανάπτυξη εφαρμογών στο Android είναι μια σύνθετη και χρονοβόρα διαδικασία η οποία συνοψίζεται σε 4 βασικά στάδια, αλλά και αρκετά επί μέρους, τα οποία θα σχολιαστούν μεταξύ των βασικών.

2.1.1 Εγκατάσταση Λογισμικού

Στο πρώτο στάδιο της ανάπτυξης ο προγραμματιστής καλείτε να στήσει το περιβάλλον εργασίας στο οποίο θα γίνει ο σχεδιασμός, η ανάπτυξη, ο έλεγχος, και η λειτουργία των εφαρμογών. Μπορεί να επιλέξει όποιο περιβάλλον ανάπτυξης (IDE) τον εξυπηρετεί καλύτερα και να χρησιμοποιήσει όλα τα εργαλεία του Android SDK μηδενός εξαιρουμένου.

Στη συνέχεια θα πρέπει να δημιουργήσει έναν αριθμό από εικονικές συσκευές στην διαχείριση εικονικών συσκευών (AVD) για να δοκιμάσει την λειτουργία της εφαρμογής σε διαφορετικές πραγματικές συνθήκες λειτουργίας. Ιδανικά ο developer θα διαθέτει έναν αριθμό διαφορετικών φυσικών συσκευών ώστε να δοκιμάσει ο ίδιος πως συμπεριφέρεται η εφαρμογή του σε κάθε περίπτωση, όμως αυτή η πρακτική μπορεί να αποδειχθεί πολυδάπανη και χρονοβόρα. Εδώ αναλαμβάνουν δράση η ευελιξία των AVDs, για τις οποίες θα γράψουμε περισσότερα παρακάτω.

Εικόνα 2.1: Πρώτο βήμα – Εγκατάσταση Λογισμικού

2.1.2 Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής

Πρόκειται αν μη τι άλλο για τη πιο χρονοβόρα και πολύπλοκη διαδικασία. Σε αυτό το στάδιο ο προγραμματιστής πρέπει να αποφασίσει για τις δυνατότητες και το περιεχόμενο που θα περιλαμβάνει η εφαρμογή, να εντοπίσει ποιες από αυτές τις δυνατότητες είναι εφικτές και ποιες θέλουν παραπάνω έρευνα για να προστεθούν στο μέλλον, να σχεδιάσει το layout με γνώμονα την λειτουργικότητα και να αποφύγει υπερβολές στο σχεδιασμό, και τέλος να δέσει αρμονικά τον κώδικα με το layout για να φέρει το τελικό αποτέλεσμα.

Η διαδικασία ξεκινάει με ένα νέο Project το οποίο θα περιέχει τον πηγαίο κώδικα, τις εικόνες, τα κείμενα και γενικά ότι χρειάζεται η εφαρμογή για να τρέξει ως οφείλει. Στο project του ο developer θα πρέπει να φροντίσει ώστε το υλικό του να είναι τακτοποιημένο και ο κώδικας του ευανάγνωστος ώστε να ακολουθήσει η διαδικασία του Debugging

Εικόνα 2.2: Δεύτερο βήμα - Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής

2.1.3 Αποσφαλμάτωση (Debugging) και Δοκιμαστική Φάση Εφαρμογής

Η διαδικασία του debugging είναι εξίσου κρίσιμη και μερικές φορές και εξίσου χρονοβόρα με την διαδικασία ανάπτυξης του πηγαίου κώδικα της εφαρμογής. Αποτελείτε από αρκετά επί μέρους στάδια τα οποία αναλύονται παρακάτω.

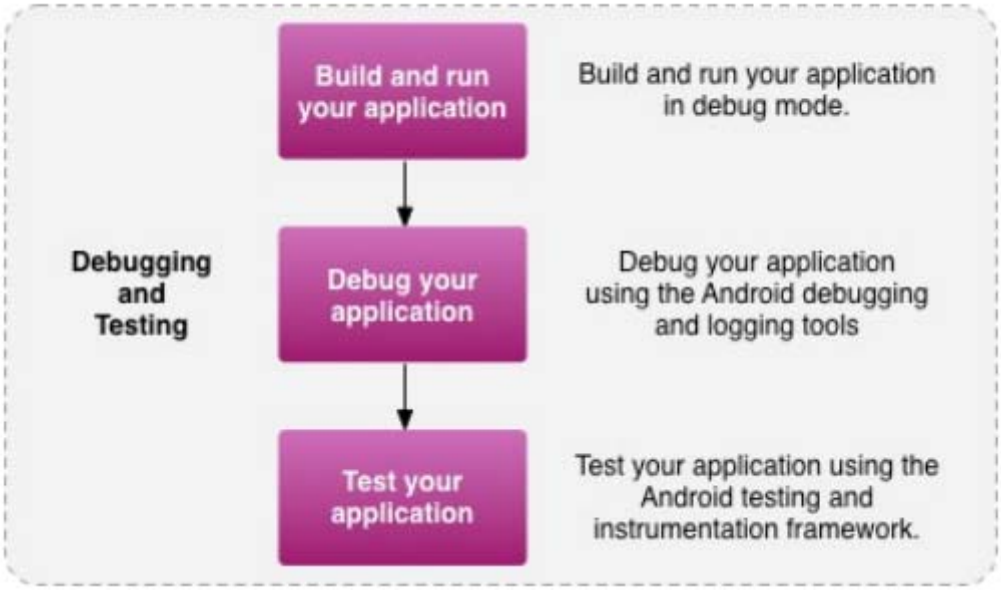
Το πρώτο στάδιο αφορά το αρχικό χτίσιμο της εφαρμογής και η λειτουργία αυτής σε debug mode. Για να γίνει το compile της εφαρμογής φυσικά τα περισσότερα περιβάλλοντα ανάπτυξης (IDE) προϋποθέτουν ότι ο κώδικας δεν έχει κανένα συντακτικό λάθος, αλλιώς ειδοποιούν τον χρήστη να τα διορθώσει. Αφού γίνει το compile η εφαρμογή μπορεί να δοκιμαστεί είτε σε εικονική συσκευή μέσω του AVD Manager, είτε απευθείας σε φυσική συσκευή μέσω ADB push εντολής. Για το ADB θα μιλήσουμε εκτενώς παρακάτω.

Στο δεύτερο στάδιο ο προγραμματιστής καλείτε να αντιμετωπίσει τα λειτουργικά και αισθητικά προβλήματα της εφαρμογής του, πρώτα εντοπίζοντας τα στην λειτουργία της συσκευής και μετά διορθώνοντας τα κομμάτια του κώδικα που δημιουργούν τα σφάλματα. Το κύριο εργαλείο που κάνει αυτή τη διαδικασία εφικτή είναι το "LogCat" το οποίο μας επιστρέφει το stack trace του κώδικα στο σημείο εκείνο που συνέβη το σφάλμα. Υπάρχουν φυσικά και άλλα εργαλεία τα οποία θα αναλυθούν εκτενώς παρακάτω.

Στο τρίτο στάδιο ο προγραμματιστής αφού έχει τελειώσει την αποσφαλμάτωση (debugging) επιστρέφει στο βήμα ένα, δηλαδή στο compile και τη δοκιμή της εφαρμογής σε εικονική η φυσική συσκευή ώστε να διαπιστώσει τα αποτελέσματα του 2 βήματος, της αποσφαλμάτωσης.

Ένα προαιρετικό στάδιο είναι η "Δημόσια δοκιμαστική φάση" της εφαρμογής. Σε αυτή τη φάση εθελοντές προσφέρονται να δοκιμάσουν τις λειτουργίες της εφαρμογής στις συσκευές τους και να αναφέρουν προβλήματα, παρατηρήσεις, προτάσεις και άλλα σχόλια που μπορεί προκύψουν από τη χρήση της εφαρμογής.

Φυσικά η διαδικασία του debugging είναι σαν ένα βρόγχος (loop) που επαναλαμβάνεται συνέχεια μέχρι να εντοπιστούν και να διορθωθούν όλα τα σφάλματα της εφαρμογής, και για αυτό το λόγο μπορεί να αποδειχθεί εξαιρετικά χρονοβόρα.



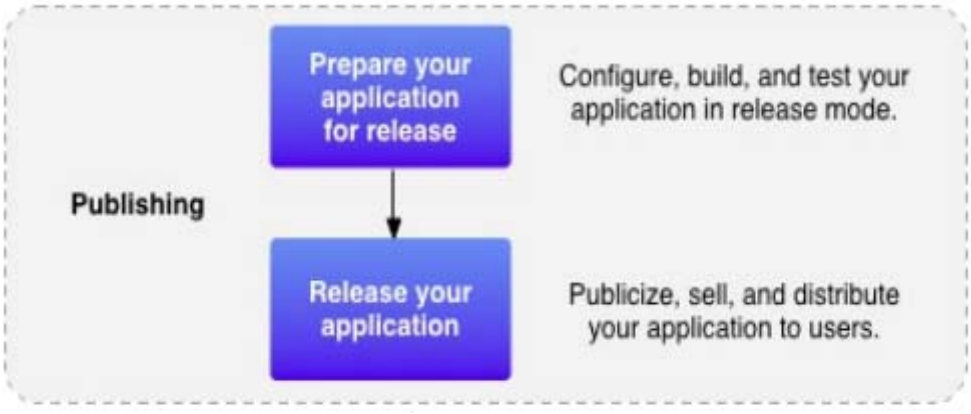
Εικόνα 2.3: Τρίτο βήμα - Δοκιμή και Debugging της εφαρμογής

2.1.4 Τελική έκδοση και δημοσίευση της εφαρμογής στο κοινό

Στο τέταρτο και τελευταίο στάδιο της ανάπτυξης ο προγραμματιστής έχει να κάνει μερικές τελευταίες κινήσεις. Πρώτον πρέπει να έχει διορθώσει όλα τα σφάλματα που προέκυψαν από την διαδικασία αποσφαλμάτωσης, να κάνει τις τελευταίες ρυθμίσεις και tweaks της εφαρμογής, και να κάνει το τελικό compile της εφαρμογής σε κανονική λειτουργία αυτή τη φορά και όχι debug.

Στη συνέχεια ακολουθεί η διάθεση της εφαρμογής με το μέσο της επιλογής του developer. Μπορεί να την διαθέσει στο Google Play, αφού πρώτα κάνει λογαριασμό developer, ή να την διαθέσει σε κάποιο εναλλακτικό market όπως το marketplace της Amazon. Μπορεί κατά τη δημοσίευση σε οποιοδήποτε να ορίσει τιμή πώλησης ή να διαθέσει την εφαρμογή δωρεάν. Τη στιγμή που γράφονται αυτές οι γραμμές δεν είναι εφικτό ακόμη για
τους Έλληνες Developers να διαθέσουν εφαρμογές επί πληρωμή, με την Google να μην έχει ανακοινώσει ακόμη πότε αυτό θα γίνει εφικτό.

Επίσης άλλο ένα μέσο διάθεσης μπορεί να είναι η προσωπική η εταιρική ιστοσελίδα του δημιουργού. Το μειονέκτημα φυσικά σε αυτή τη περίπτωση είναι η έλλειψη ελέγχου για updates της εφαρμογής από έναν αυτόματο μηχανισμό ελέγχου και λήψης όπως είναι τα διάφορα marketplaces



Εικόνα 2.4: Τέταρτο βήμα - Δημοσίευση της Εφαρμογής

2.2 Android SDK

Το Android SDK (Software Developers Kit) αποτελεί μια συλλογή εργαλείων και βιβλιοθηκών που καθιστούν εφικτή την ανάπτυξη εφαρμογών στο Android. Τι στιγμή που γράφονται αυτές οι γραμμές, το SDK έχει φτάσει στην έκδοση r21.1 η οποία υποστηρίζει το Android 4.2. Το λογισμικό ανάπτυξης λοιπόν περιλαμβάνει μια μεγάλη λίστα με εργαλεία ανάπτυξης. Σε αυτά περιλαμβάνονται:

- Εργαλεία Debugging των εφαρμογών
- Βιβλιοθήκες
- Εξομοιωτής συσκευών (Android Virtual Machines)
- Documentation
- Δείγματα Κώδικα
- Tutorials

Το SDK υποστηρίζει πολλά δημοφιλή λειτουργικά συστήματα συμπεριλαμβανομένων όλων των σύγχρονων διανομών Linux, το MAC OS X 10.4.9 και μεταγενέστερα, και τα Windows XP και τις μεταγενέστερες εκδόσεις.

Το λογισμικό ανάπτυξης αποτελείτε από πακέτα τα οποία βρίσκονται αποθηκευμένα σε ένα επίσημο repository της Google, και ο προγραμματιστής μπορεί να κατεβάσει πέραν των βασικών πακέτων, και άλλα τα οποία υποστηρίζουν παλαιότερες εκδόσεις του Android, ή άλλες συσκευές εκτός κινητών συσκευών (πχ Google TV Addon).

Όσον αφορά την υποστήριξη παλαιότερων εκδόσεων του Android, το SDK κάνει εφικτή την υποστήριξη σε αυτές δίνοντας στον προγραμματιστή την δυνατότητα να στοχεύσει αυτός σε πια APIs θα απευθύνεται η εφαρμογή του. Αυτό είναι αναγκαίο λόγω του ότι πολλοί χρήστες έχουν παλαιότερες λειτουργικές συσκευές οι οποίες κυκλοφορήσαν με παλαιότερες εκδόσεις του Android (πχ 1.6 ή 2.1), και ο κατασκευαστής της συσκευής δεν έχει ή δεν πρόκειται να βγάλει αναβάθμιση για την συσκευή τους. Το πρόβλημα αυτό είναι γνωστό σαν διάσπαση του Android (Android Fragmentation) και θα αναλυθεί εκτενώς παρακάτω.

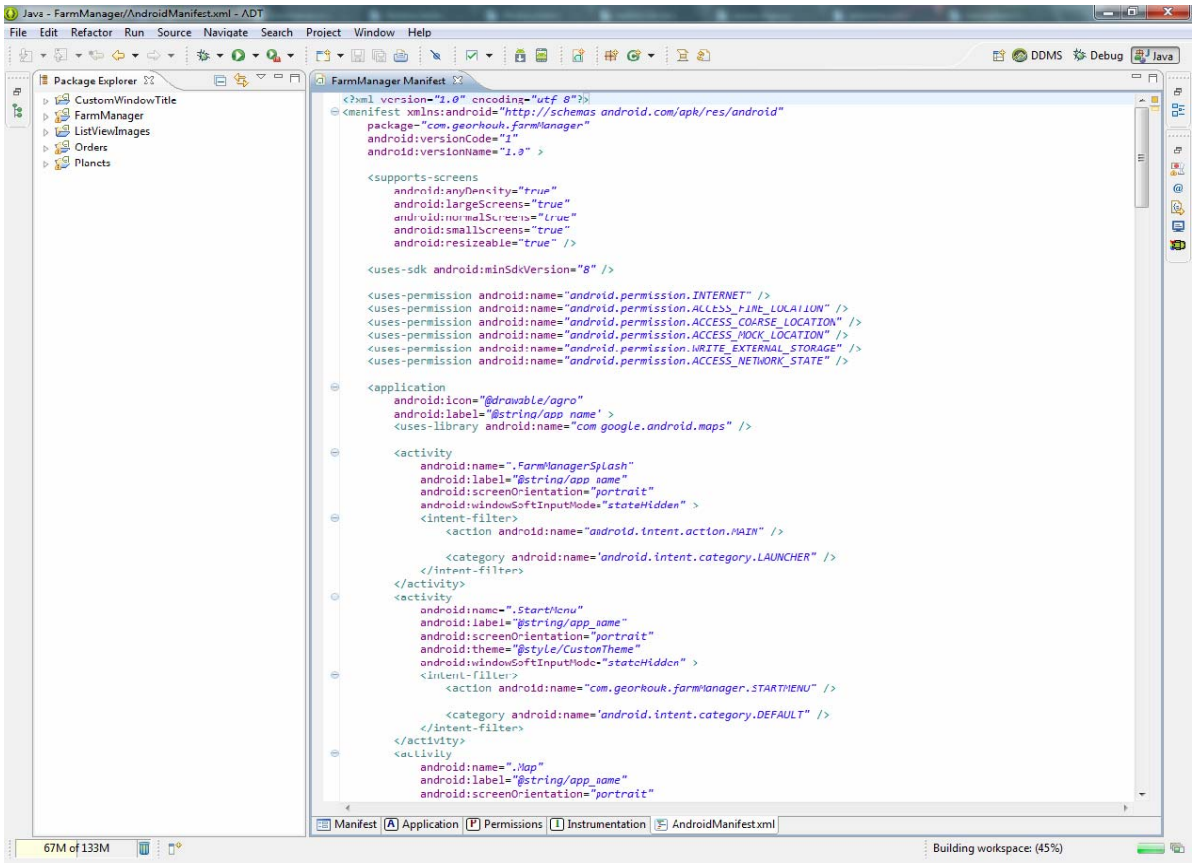
2.3 Χρήση του Eclipse IDE μαζί με το ADT (Android Development Tools)

Ο προγραμματισμός στο Android βασίζεται στην γλώσσα Java και ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει έναν οποιονδήποτε text editor για να γράψει κώδικα για να επεξεργαστεί τα αρχεία *.Java και *.XML και μετέπειτα να τα κάνει compile μέσω γραμμής εντολών χρησιμοποιώντας το JDK (Java Development Kit). Ο συγκεκριμένος τρόπος ανάπτυξης δεν είναι ιδιαίτερα φιλικός στον χρήστη γι' αυτό συνίσταται η χρήση ενός IDE (Integrated Development Environment) που να υποστηρίζει Java, όπως το Eclipse ή το Netbeans.

Η Google υποστηρίζει επίσημα το Eclipse και έχει αναπτύξει ειδικά γι' αυτό το ADT plugin, το οποίο παρέχει σύνδεση με το Android SDK με όλες τις δυνατότητες που περιλαμβάνει αυτό. Επίσης το plugin παρέχει σύνδεση με τον AVD Manager, για διαχείριση και εκκίνηση από το GUI του, εικονικών συσκευών Android για δοκιμές και debugging των εφαρμογών.

Φυσικά όπως είπαμε και παραπάνω, ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει τον Text Editor ή IDE της επιλογής του για τη δημιουργία του κώδικα και μετέπειτα να χρησιμοποιήσει τα εργαλεία JDK και Apache Ant μέσω γραμμής εντολών για να κάνει compile την εφαρμογή του ώστε να την τεστάρει με όλες τις δυνατότητες που το παρέχει το Android SDK.

Η επιλογή ενός IDE που κάνει όλη την πολύπλοκη δουλεία για μας είναι προφανής λοιπόν. Επίσης τα περισσότερα παραδείγματα και άρθρα για το Android στηρίζονται στο γεγονός ότι η πλειονότητα των developers χρησιμοποιεί το Eclipse μαζί με το ADT plugin οπότε ξεκινάνε με αυτό σαν δεδομένο.



Εικόνα 2.5: Προεπιλεγμένη διάταξη του Eclipse IDE

2.4 Προκλήσεις ανάπτυξης εφαρμογών στο Android

Η ανάπτυξη εφαρμογών στο προγραμματιστικό περιβάλλον του Android είναι μια αρκετά απαιτητική διαδικασία, διότι απαιτεί την υποστήριξη εκατοντάδων συσκευών, την υλοποίηση και υποστήριξη ενός λειτουργικού περιβάλλοντος διεπαφής χρήστη, και άλλα πολλά τα οποία θα αναλύσουμε παρακάτω.

2.4.1 Android Design Guidelines

Η υλοποίηση μιας νέας εφαρμογής στο Android αλλά και στα υπόλοιπα λειτουργικά συστήματα, ξεκινάει από τις λειτουργικές απαιτήσεις, δηλαδή τις δυνατότητες και λειτουργίες που θα υποστηρίζει η εφαρμογή, και συνεχίζει με τον σχεδιασμό του UI layout που θα δίνει πρόσβαση στον χρήστη στις παραπάνω λειτουργίες. Ο σχεδιασμός λοιπόν έχει μεγαλύτερη σημασία από τις ίδιες τις λειτουργίες τις εφαρμογής μιας και είναι το σημείο πρόσβασης προς αυτές!

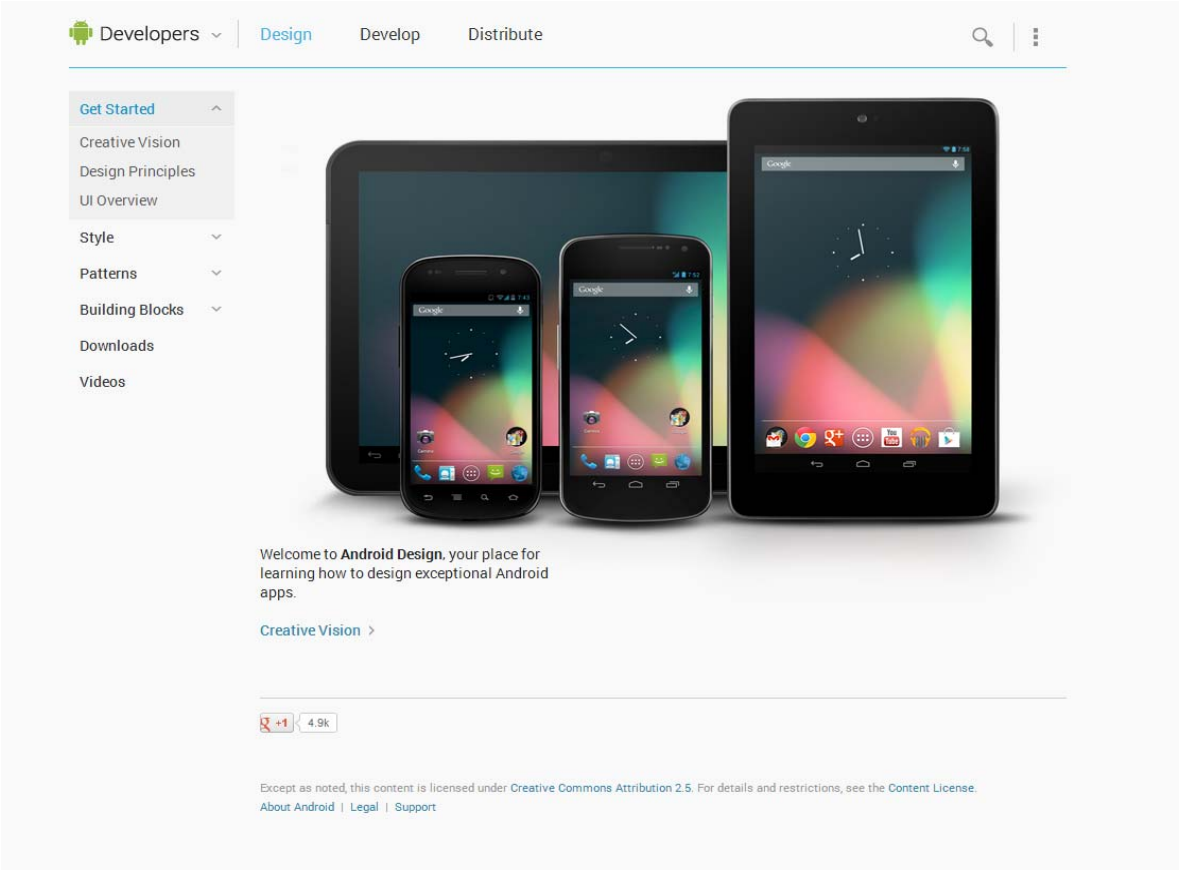
Μια κακοσχεδιασμένη εφαρμογή η οποία κρύβει τις λειτουργίες τις πίσω από πολλά κουμπιά και μενού ενδέχεται να μπερδέψει τον χρήστη και ίσως και να τον αποτρέψει από το να την χρησιμοποιήσει. Αυτό φυσικά δεν είναι επιθυμητό γι' αυτό και υπάρχουν κάποια ενδεικτικά επίσημα Guidelines (οδηγίες) τα οποία υποδεικνύουν στους developers τις ιδεατές και μη συμπεριφορές κατά τον σχεδιασμό της εφαρμογής τους.

Στα πρώτα χρόνια του Android ο σχεδιασμός τον εφαρμογών ήταν είτε καθαρή μεταφορά (port) από άλλο OS (πχ iOS, Symbian) και συνήθως δεν ακολουθούσε καμία κοινή γραμμή σχεδιασμού με τις υπόλοιπες εφαρμογές κάτι που πολλές φορές μπέρδευε τον χρήστη που είχε συνηθίσει το κοινό layout πολλών εφαρμογών των υπολοίπων mobile OSes. Αυτό η Google προσπάθησε να το αλλάξει με την έλευση του Android 4.0 (Ice Cream Sandwich) όποτε και δημοσίευσε στο Ιντερνέτ τη σελίδα Android Design για να καθοδηγήσει τους developers σε μία κοινή γραμμή ανάπτυξης εφαρμογών ώστε να πετύχει αύξηση λειτουργικότητας.

Η αύξηση λειτουργικότητας επιτυγχάνεται λόγω του ότι ο χρήστης θα έχει γνωστά σημεία επαφής σε κάθε εφαρμογή οπότε δεν θα χρειάζεται να ψάχνει εκ νέου πώς να επιστρέψει στην αρχική οθόνη ή που βρίσκεται το μενού των επιλογών, κλπ

Στο site υπάρχει πληθώρα παραδειγμάτων "καλού σχεδιασμού" τα οποία αφορούν την χρήση της Action Bar, την χρήση των swappable tabs, την δυνατότητα δηλαδή να αλλάζουμε οθόνες σέρνοντας αριστερά η δεξιά το δάχτυλό μας στην οθόνη της συσκευής, κλπ. Επίσης υπάρχει και η ενότητα "Downloads" στην οποία υπάρχει ένα πλήρες πακέτο εικονιδίων τα οποία μπορούμε να χρησιμοποιήσουμε στα μενού των εφαρμογών μας η όπου αλλού θέλουμε!

Οι οδηγίες φυσικά είναι ενδεικτικές και όχι αναγκαστικές. Αν κάποιος developer θέλει να "παρεκτραπεί" και να δώσει κάποιο εντελώς διαφορετικό interface το οποίο όμως θα εξυπηρετεί καλύτερα τους δικούς του σκοπούς, μπορεί να το κάνει ελεύθερα. Αυτή άλλωστε είναι και η ομορφιά του Android!



Εικόνα 2.6: Αρχική Σελίδα Android Design

2.4.2 Υποστήριξη πολλαπλών συσκευών

Το λειτουργικό σύστημα Android τρέχει σε μια πληθώρα συσκευών οι οποίες μπορεί να έχουν πολύ διαφορετικές προδιαγραφές η μία από την άλλη. Η διαφοροποίηση των συσκευών εντοπίζεται:

- Στις πολλές εκδόσεις του Android που υπάρχουν. Αυτή τι στιγμή που γράφονται αυτές ο γραμμές υπάρχουν 8 κύριες εκδόσεις (1.5-4.2) με κυρίαρχη έκδοση την 2.3 Gingerbread.
- Στην μεγάλη ποικιλία hardware που κυκλοφορεί στην αγορά. Υπάρχουν συσκευές με επεξεργαστή στα 600Mhz και 256MB διαθέσιμης μνήμης RAM και υπάρχουν και συσκευές με επεξεργαστή 4 πυρήνων (Quad Core) στα 1.5Mhz και 2GB μνήμης RAM. Εκτός από τις διαφορές σε επίπεδο microchip η κύρια διαφορά μεταξύ των συσκευών εντοπίζεται στην μεγάλη ποικιλία διαστάσεων οθόνης και πυκνότητας pixel

Ο developer λοιπόν για να κάνει την εφαρμογή του προσβάσιμη σε όσο τον δυνατόν περισσότερες συσκευές χρηστών, πρέπει να λάβει σοβαρά υπόψη του τις 2

παραπάνω παραμέτρους και να σχεδιάσει την εφαρμογή του έτσι ώστε αυτή να υποστηρίζει την πλειονότητα των συσκευών. Βέβαια αυτό σημαίνει συνεχή προσαρμογή της εφαρμογής στις νέες συνθήκες που μπορεί να προκύψουν, και χρήση των νέων δυνατοτήτων που ενδεχομένως θα παρέχει μια νέα έκδοση του λειτουργικού, χωρίς να επηρεάζεται η υποστήριξη στις παλαιότερες συσκευές. Αυτό είναι μεγάλο ζήτημα το οποίο θα αναλυθεί εκτενώς παρακάτω.

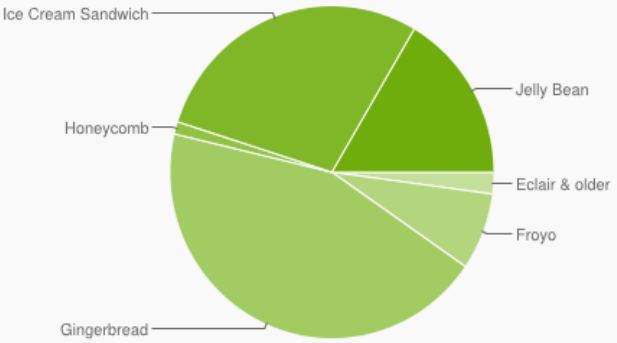
2.4.2.1 Υποστήριξη παλαιότερων εκδόσεων του Android

Όπως γράψαμε παραπάνω αλλά και στο εισαγωγικό κομμάτι της εργασίας, αυτή τη στιγμή υπάρχουν 8 κύριες εκδόσεις του Android με πιο πρόσφατη την έκδοση 4.2 με κωδική ονομασία "Jelly Bean" και κυρίαρχη έκδοση την 2.3.3 "Gingerbread".

Αυτή η συνεχής εξέλιξη της πλατφόρμας αποτελεί πλεονέκτημα αλλά και πρόκληση για τον προγραμματιστή ο οποίος θα πρέπει να ακολουθεί τις εξελίξεις και να χρησιμοποιεί τις νέες δυνατότητες που του προσφέρει η κάθε έκδοση, χωρίς να παραγκωνίζει την υποστήριξη στις παλαιότερες εκδόσεις του Android. Αυτό είναι ένα σημαντικό πρόβλημα καθότι κάποια νέα features δεν υποστηρίζονται στα παλιότερα APIs και άρα καθιστούν αδύνατη τη χρήση τους σε κάποια παλαιότερη έκδοση του Android. Η Google έχει προσπαθήσει να λύσει αυτό το πρόβλημα βγάζοντας μαζί με κάθε νέα έκδοση του λειτουργικού της, και μιας "βιβλιοθήκης συμβατότητας" η οποία αναλαμβάνει να κάνει διαθέσιμα τα νέα εργαλεία στα παλιότερα APIs. Σε γενικές γραμμές η συγκεκριμένη λύση λειτουργεί αρκετά ικανοποιητικά αλλά δυστυχώς δεν μεταφέρονται πάντα όλες οι νέες δυνατότητες στις παλαιότερες εκδόσεις του λειτουργικού συστήματος με αυτό τον τρόπο.

Τρανό παράδειγμα αυτού είναι η υποστήριξη της ActionBar η οποία προστέθηκε στην έκδοση 3.0 του Android και πλέον ανήκει στις βασικές οδηγίες σχεδιασμού εφαρμογών (android design guidelines). Η βιβλιοθήκη υποστήριξης την κάνει διαθέσιμη στις παλαιότερες εκδόσεις, αλλά με σαφής περιορισμούς στην χρήση της! Ευτυχώς το συγκεκριμένο feature το υποστηρίζουν άψογα 2 βιβλιοθήκες που έχουν αναπτυχθεί από προγραμματιστές του Android, και έχει καλυφτεί το κενό που άφησε η Google.

Version	Codename	API	Distribution
1.6	Donut	4	0.2%
2.1	Eclair	7	1.9%
2.2	Froyo	8	7.5%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	43.9%
3.1	Honeycomb	12	0.3%
3.2		13	0.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	28.6%
4.1	Jelly Bean	16	14.9%
4.2		17	1.6%



Data collected during a 14-day period ending on March 4, 2013

Εικόνα 2.7: Ποσοστά των Android εκδόσεων

Όπως βλέπουμε από τα παραπάνω στατιστικά στοιχεία που ενημερώνονται αυτόματα κάθε 2 εβδομάδες από την Google, τη μερίδα του λέοντος κατέχουν οι εκδόσεις 2.3 με 44% και η 4.0 με ποσοστό 28.6%. Η νεότερη έκδοση του Android, 4.2 κατέχει μόλις το 16% περίπου παρότι κυκλοφόρησε τον Οκτώβριο του 2012, δηλαδή 7 μήνες από τη στιγμή που γράφονται αυτές οι γραμμές. Τέλος οι παλαιότερες εκδόσεις 1.6/2.1/2.2 υπάρχουν ακόμα σε συσκευές σε ποσοστό 10% περίπου. Αυτό δυστυχώς πρόκειται για το φαινόμενο διάσπασης (Fragmentation) του Android το οποίο θα αναλυθεί εκτενέστερα παρακάτω.

Γενικά μια καλή προγραμματιστική συμπεριφορά σύμφωνα με την κοινότητα, είναι η εκάστοτε εφαρμογή μας να υποστηρίζει τουλάχιστον το 90% των ενεργών συσκευών όσον αφορά την έκδοση Android που φοράνε. Αυτό από ότι βλέπουμε από το παραπάνω γράφημα είναι σχετικά εύκολο καθώς υποστηρίζοντας μόνο τις καινούργιες εκδόσεις 2.3 και 4.0 αυτή τη στιγμή, έχουμε υποστηρίξει το ~80% των ενεργών συσκευών! Παρόλα αυτά έχουμε όμως αποκλείσει έναν σημαντικό αριθμό χρηστών οι οποίοι χρησιμοποιούν παλαιότερες (1.6, 2.1) αλλά και νεότερες (4.1) συσκευές.

Οι developers πρέπει να βλέπουν και να αξιολογούν το μερίδιο των συσκευών με τέτοιο τρόπο ώστε να υποστηρίζουν όσο το δυνατό περισσότερες συσκευές χρηστών χωρίς όμως να υποβαθμίζουν την ποιότητα και λειτουργικότητα της

συσκευής τους. Πρόκειται για μια λεπτή ισορροπία που επιτυγχάνεται μετά από αρκετή προσπάθεια από μέρος των developers.

Η υποστήριξη των διαφορετικών εκδόσεων ορίζεται στο αρχείο AndroidManifest.xml και εφόσον έχει καθοριστεί ένα κατώτατο στοχευμένο API, η εφαρμογή μας δεν μπορεί να εγκατασταθεί σε συσκευή που φοράει παλαιότερη έκδοση από αυτή που υποστηρίζει το API.

Καλή πρακτική σχεδιασμού λοιπόν είναι να στοχεύσουμε ένα αρκετά χαμηλό API το οποίο όμως δεν θα μας αναγκάσει να κάνουνε συμβιβασμούς στις λειτουργίες της εφαρμογής και στο τέλος να κάνουμε compile την τελική έκδοση με τη νεότερη έκδοση του SDK ώστε να εξασφαλίσουμε υποστήριξη και στις νεότερες συσκευές.

2.4.2.2 Υποστήριξη πολλαπλών διαστάσεων οθόνης και πυκνότητας pixel

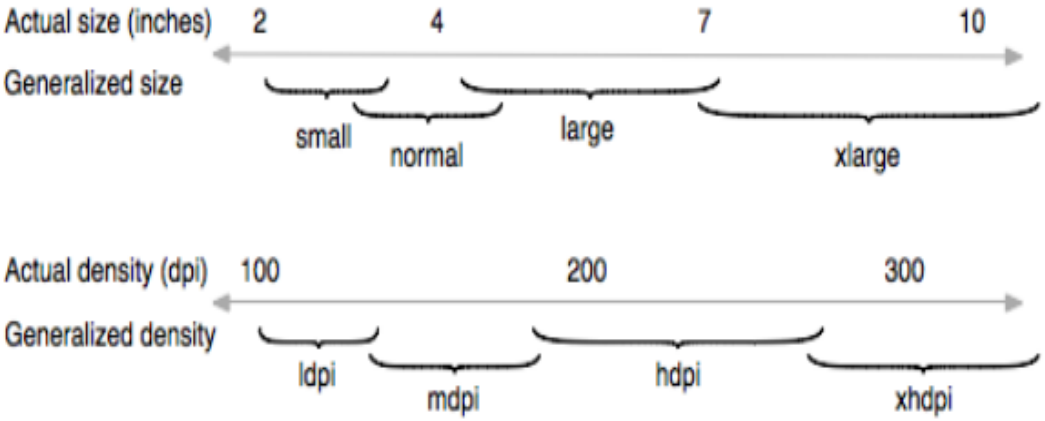
Η ταυτόχρονη υποστήριξη των πολλών εκδόσεων του Android είναι η πρώτη πρόκληση του προγραμματιστή. Η δεύτερη μεγάλη πρόκληση είναι η ταυτόχρονη υποστήριξη των πολλαπλών διαστάσεων οθόνης που διαθέτουν οι εκατοντάδες συσκευές που κυκλοφορούν στην αγορά, και η διαφορετική πυκνότητα pixel που διαθέτει η κάθε μία από αυτές.

Όπως γράψαμε και παραπάνω το γραφικό περιβάλλον μιας εφαρμογής είναι ίσως σημαντικότερο και από τις δυνατότητες που παρέχει, καθότι ένα κακοσχεδιασμένο layout μπορεί να κάνει την εφαρμογή δύσχρηστη ή ακόμη και άχρηστη! Άρα είναι πολύ σημαντικό για τον προγραμματιστή να λάβει υπόψη του την πληθώρα αναλύσεων και διαφορετικών διαστάσεων οθόνης που διαθέτουν οι συσκευές.

Όσον αφορά το εύρος των συσκευών μπορεί να φτάσει από τις 2.4' (μικρά chatphones) έως τις 13' (μεγάλα tablets), και οι ανάλυση αυτών των συσκευών ξεκινάει από τα 240x320 pixels (QVGA) και φτάνει μέχρι τα 1920x1080 (Full HD) για τα νεότερα tablets! Δημοφιλείς αναλύσεις είναι η 240x320(QVGA), η 240x400 (WQVGA), η 320x480(HVGA), και φυσικά η πιο δημοφιλής όλων η 480x800 (WVGA). Νεότερες και μεγαλύτερης οθόνης συσκευής υποστηρίζουν και μεγαλύτερες αναλύσεις όπως η 540x960 (qHD) και 720x1280(WXGA).

Όπως βλέπουμε υπάρχει μια πληθώρα αναλύσεων για να υποστηρίξει ο προγραμματιστής η κάθε μία με περισσότερο η λιγότερο διαθέσιμο χώρο στην οθόνη. Το android για πρακτικούς λόγους έχει χωρίσει τις διαφορετικές αναλύσεις οθονών

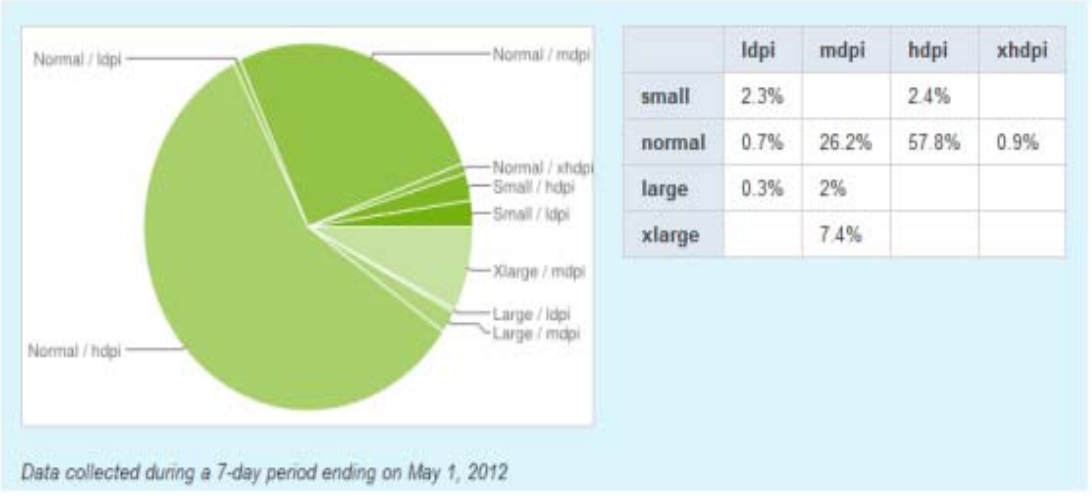
σε τέσσερις κατηγορίες οθονών οι οποίες συσχετίζονται άμεσα με τέσσερις κατηγορίες πυκνότητες pixel ανά ίντσα.



Εικόνα 2.8: Διαχωρισμός Αναλύσεων και Πυκνότητας σε Κατηγορίες

Όπως βλέπουμε από το παραπάνω σχεδιάγραμμα, το Android χωρίζει το μέγεθος τις οθόνης σε 4 επιμέρους κατηγορίες αναλόγως το μέγεθος της σε ίντσες, και οι αναλύσεις χωρίζονται επίσης σε 4 επιμέρους κατηγορίες DPI (Dots Per Inch). Αυτό γίνεται ώστε να διευκολύνει όσο το δυνατόν περισσότερο τους προγραμματιστές, να βελτιώσουν την εμφάνιση των εφαρμογών τους με όσο το δυνατόν λιγότερο κόπο.

Όπως και στην περίπτωση των πολλών εκδόσεων του Android που έχουν πρόσβαση στο μάρκετ, η Google παρέχει στους developers την κατανομή μεγέθους οθόνης προς DPI για να βοηθήσει τους προγραμματιστές να σχεδιάσουν τις εφαρμογές τους αποδοτικότερα.



	ldpi	mdpi	hdpi	xhdpi
small	2.3%		2.4%	
normal	0.7%	26.2%	57.8%	0.9%
large	0.3%	2%		
xlarge		7.4%		

Εικόνα 2.9: Κατανομή των διαφορετικών DPI των οθονών

Από ότι βλέπουμε από το παραπάνω σχεδιάγραμμα, τα πράγματα εν τέλη δεν είναι και τόσο τραγικά για τον developer! Όπως και στις εκδόσεις των API 2 μεγέθη κατέχουν τη μερίδα του λέοντος, και τα 2 μάλιστα στις ίδιες αναλύσεις (normal) αλλά σε διαφορετικό. Σε γενικές γραμμές οι πιο συνηθισμένες διατάξεις που πρέπει να δοκιμάσει ο developer την εφαρμογή του συνοψίζονται στην παρακάτω εικόνα.

	ldpi	**mdpi**	**hdpi**
Small	240x320 pixels 120 dpi		240x400 pixels 240 dpi
Normal		480x800 pixels 160 dpi	480x800 pixels 240 dpi

Εικόνα 2.10: Οι πιο συνηθισμένες διατάξεις των οθονών στο Android

Φυσικά δεν πρέπει να αγνοηθούν οι υπόλοιπες διατάξεις μεγέθους οθόνης προς dpi, αλλά όπως και στην περίπτωση της υποστήριξης των API, αυτό είναι μια λεπτή ισορροπία η οποία είναι στο χέρι του προγραμματιστή να την κρατήσει. Η πλατφόρμα του Android φυσικά εκτός από το να μας δημιουργεί πολύπλοκες καταστάσεις προς διαχείριση, μας δίνει επιλογές και δυνατότητες ώστε να αντεπεξέλθουμε στο απαιτητικό έργο της ανάπτυξης εφαρμογής στο περιβάλλον του. Οι δυνατότητες συνοψίζονται ως εξής:

- Ο developer μπορεί να επιλέξει ποιες οθόνες θα υποστηρίζει η εφαρμογή του με τον ίδιο τρόπο που επιλέγει πια API θα υποστηρίζονται, δηλαδή μέσω του AndroidManifest.xml. Αυτό σημαίνει ότι η υποστήριξη διαφορετικών οθονών είναι καθαρά επιλογή του προγραμματιστή, ο οποίος αν κρίνει αναγκαίο μπορεί να αποκλείσει την λειτουργία της εφαρμογής στις οθόνες που δεν επιθυμεί να υποστηρίξει.
- Στην συνηθέστερη περίπτωση που ο προγραμματιστής θέλει να υποστηρίξει επιπλέον αναλύσεις και διαστάσεις οθόνης πέρα από τις συνηθισμένες, το SDK του δίνει τη δυνατότητα παρέχοντας του δύο εξαιρετικές δυνατότητες:
    - Η πρώτη είναι η δυνατότητα χρήσης διαφορετικών layout ανά διαφορετικό μέγεθος οθόνης. Αυτό σημαίνει ότι ο developer δεν χρειάζεται να συμβιβαστεί σε ένα layout xml ώστε να καλύψει όλες τις οθόνες. Μπορεί να χρησιμοποιήσει όσα θέλει, για να καλυφθούν όσο το δυνατόν περισσότερα μεγέθη και αναλύσεις

οθόνης. Δεν πρέπει να ξεχνάμε άλλωστε ότι οι μεγαλύτερες οθόνες παρέχουν και περισσότερο χώρο στον developer για να προβάλει επιπλέον υλικό, με διαφορετικό ίσως τρόπο από ότι θα το προέβαλε σε μια μικρότερη οθόνη

- Η δεύτερη δυνατότητα που παρέχει είναι αυτή της χρήσης πολλών γραφικών, διαφορετικών διαστάσεων, ώστε να εξυπηρετούνται σωστά όλες οι αναλύσεις οθόνης. Η υλοποίηση αυτής της δυνατότητας είναι εξαιρετικά απλή. Ο developer δημιουργεί φακέλους στο project του, με το όνομα της διάταξης που θέλει να παρέχει γραφικά η layout (πχ drawable – large – hdpi) ή χρησιμοποιεί τους υπάρχοντες φακέλους, και αποθηκεύει στον καθένα το ίδιο γραφικό (*.png, *jpg, *.gif) αλλά στην ανάλυση που επιθυμεί να προβληθεί αυτό στην εκάστοτε διαφορετική διάταξη μεγέθους οθόνης προς DPI.

2.5 Δοκιμή και Αποσφαλμάτωση (Debugging) της Εφαρμογής

Πριν να εκδώσει την εφαρμογή του στο κοινό, ο προγραμματιστής του Android, αλλά και οποιασδήποτε άλλης πλατφόρμας, οφείλει να δοκιμάσει ενδελεχώς, κάθε λειτουργική και αισθητική παράμετρο της εφαρμογής ώστε να εξασφαλίσει ότι ο χρήστης της θα έχει όσο το δυνατόν καλύτερη εμπειρία χρήσης! Καλύτερη εμπειρία χρήσης φυσικά οδηγεί σε καλές εντυπώσεις, και περισσότερους χρήστες με την πάροδο του χρόνου. Μια κακή εμπειρία χρήσης, οδηγεί σε δυσαρεστημένους χρήστες και άρα αποτυχία.

Ο αυτοσκοπός της ανάπτυξης εφαρμογών είναι η ικανοποίηση των αναγκών όσο το δυνατόν μεγαλύτερου αριθμού χρηστών! Αυτό φυσικά δεν επιτυγχάνεται αμέσως. Πρέπει να ακολουθηθεί μια εκτενής περίοδος δοκιμών για να διαπιστωθεί ότι η εφαρμογή θα αποδίδει όπως οφείλει σε όλες τις καθημερινές της χρήσεις, και αφού περάσει επιτυχώς το στάδιο των δοκιμών, ακολουθεί η ώρα της δημοσίευσης.

Πως γίνεται λοιπόν η διαδικασία της αποσφαλμάτωσης; Μιας εφαρμογή μπορεί να αποτελείτε από μερικές απλές κλάσεις των 50 γραμμών κώδικα, ή μπορεί να αποτελείτε από δεκάδες κλάσεις χωρισμένες σε πακέτα, αποτελούμενες από δεκάδες χιλιάδες γραμμές κώδικα η καθεμία! Η έρευνα για σφάλματά στην δεύτερη

περίπτωση, έχει πολύ χειρότερη αναλογία πιθανοτήτων, από την έρευνα για "ψύλλους στα άχυρα".

Η ανάγκη για εργαλεία ελέγχου, καλύπτεται στο έπακρο με μια σειρά εφαρμογών τα οποία δένοντας αρμονικά με το περιβάλλον ανάπτυξης μας (IDE) μας λύνουν τα χέρια, στο δύσκολο έργο του εντοπισμού σφαλμάτων. Τα εργαλεία αυτά αφορούν τον έλεγχο της εφαρμογής σε εικονικές μηχανές (Android Virtual Devices), καταγραφή σφαλμάτων (logcat), εργαλεία ελέγχου μνήμης και άλλων λειτουργιών της συσκευής (DDMS), κλπ.

2.5.1 Android Debug Bridge (ADB)

Για να λειτουργήσουν σωστά τα εργαλεία που αναφέραμε παραπάνω, χρειάζεται κάποιο είδος προγράμματος client-server που να συνδέει τον υπολογιστή με τις συσκευές μας, εικονικές και μη. Τον ρόλο αυτό αναλαμβάνει το Android Debug Bridge (ADB). Πρόκειται για ένα εργαλείο γραμμής εντολών που έρχεται μαζί με το Android SDK και το οποίο αποτελείτε από 3 μέρη:

- Έναν client ο οποίος τρέχει στον υπολογιστή που έχουμε στήσει το SDK. Μπορούμε είτε να τον εκκινήσουμε χειροκίνητα είτε να χρησιμοποιήσουμε κάποιο εργαλείο το οποίο ξεκινάει αυτόματα δικό του client, όπως το DDMS ή το ADT Plugin.
- Έναν server ο οποίος τρέχει σαν υπηρεσία παρασκηνίου στον υπολογιστή που βρίσκεται το SDK, όπως και ο client. Ο server εξασφαλίζει την επικοινωνία μεταξύ του client και του εργαλείου "δαίμονα" (daemon) που τρέχει στη συσκευή.
- Ο "δαίμονας" (daemon) που τρέχει σαν διεργασία παρασκηνίου στην εικονική η πραγματική συσκευή που χρησιμοποιείτε για εξομοίωση.

Όταν ξεκινάει το ADB, ο client ελέγχει αν υπάρχει κάποια υπάρχουσα διεργασία του server που να εκτελείτε ήδη, αλλιώς δημιουργεί μια νέα. Μετά δημιουργεί μια τοπική TCP σύνδεση στην θύρα 5037 και είναι έτοιμος να δεχτεί εντολές. Μετά ελέγχει το εύρος θυρών TCP από 5555 μέχρι 5585, στο οποίο επικοινωνούν οι συσκευές εξομοίωσης, και ελέγχει αν υπάρχουν διαθέσιμες και πόσες είναι αυτές. Αφού εντοπίσει κάποια συσκευή ελέγχει αν σε αυτή τη συσκευή τρέχει ο "δαίμονας", και αν ναι δημιουργείτε σύνδεση adb με την συσκευή.

Αφότου πραγματοποιηθεί σύνδεση μεταξύ συσκευής και client μπορούμε να

χρησιμοποιήσουμε όλες τις δυνατότητες που μας παρέχει το ADB για να ασκήσουμε πλήρη έλεγχο της συσκευής μας. Οι εντολές μέσω γραμμής εντολών δίνονται ως εξής:

adb [-d|-e|-s <serialNumber>] <command>

όπου -d είναι η απευθείας εντολή εάν υπάρχει μόνο μία συνδεδεμένη συσκευή, **-e** η απευθείας εντολή εάν υπάρχει μόνο μία συνδεδεμένη *εικονική* συσκευή, **-s <serialNumber>** η απευθείας εντολή στην συσκευή που έχει το **<serialNumber>,** και **<command>** είναι η εντολή που θέλουμε να εκτελεστεί. Παράδειγμα χρήσης:

adb -s emulator-5556 install farmManager.apk

Στο παραπάνω παράδειγμα γίνεται εγκατάστασης της εφαρμογής με όνομα "farmManager.apk" στην συσκευή εξομοίωσης που ακούει στη θύρα 5556.

2.5.2 Εικονικές Συσκευές Android (Android Virtual Devices – AVD)

Όπως γράψαμε και παραπάνω, ο developer πρέπει πριν να εκδώσει την εφαρμογή του να την δοκιμάσει σε ένα αριθμό συσκευών για να εξασφαλίσει την ομαλή λειτουργία της σε όλες τις συνθήκες. Φυσικά το κόστος των συσκευών είναι αρκετά μεγάλο για να αποθαρρύνει τον προγραμματιστή να έχει στην κατοχή του 10-20 συσκευές για να ελέγξει σε όλες τις λειτουργίες και την εμφάνιση της εφαρμογής του. Το πρόβλημα αυτό έρχεται να λύσει η ύπαρξη των εικονικών συσκευών του Android.

Πρόκειται για μια συσκευή εξομοίωσης η οποία μας επιτρέπει να εξομοιώσουμε την λειτουργία και συμπεριφορά μιας κανονικής συσκευής, ορίζοντας τις επιλογές υλικού και λογισμικού που θέλουμε στον εξομοιωτή του Android. Με αυτό τον τρόπο ο developer μπορεί να ελέγξει την εφαρμογή του σε μια σειρά από πραγματικά σενάρια λειτουργίας και να πάρει γρήγορα και άμεσα feedback για τη λειτουργία της εφαρμογής του. Μια εικονική συσκευή αποτελείτε από:

- *Το προφίλ του υλικού:* Σε αυτό προσδιορίζονται οι ιδιότητες και τα χαρακτηριστικά της εικονικής συσκευής. Μπορούμε παραδείγματος χάρη να ορίσουμε την ανάλυση της οθόνης και την πυκνότητα σε pixel (dpi), το μέγεθος της μνήμης RAM, αν η συσκευή θα έχει κάμερα, υποστήριξη GPS, κλπ.
- *Την έκδοση του Android:* Επιλογή της έκδοσης της πλατφόρμας του Android που θέλουμε να εξομοιώσει η εικονική συσκευή. Μπορούμε επίσης να

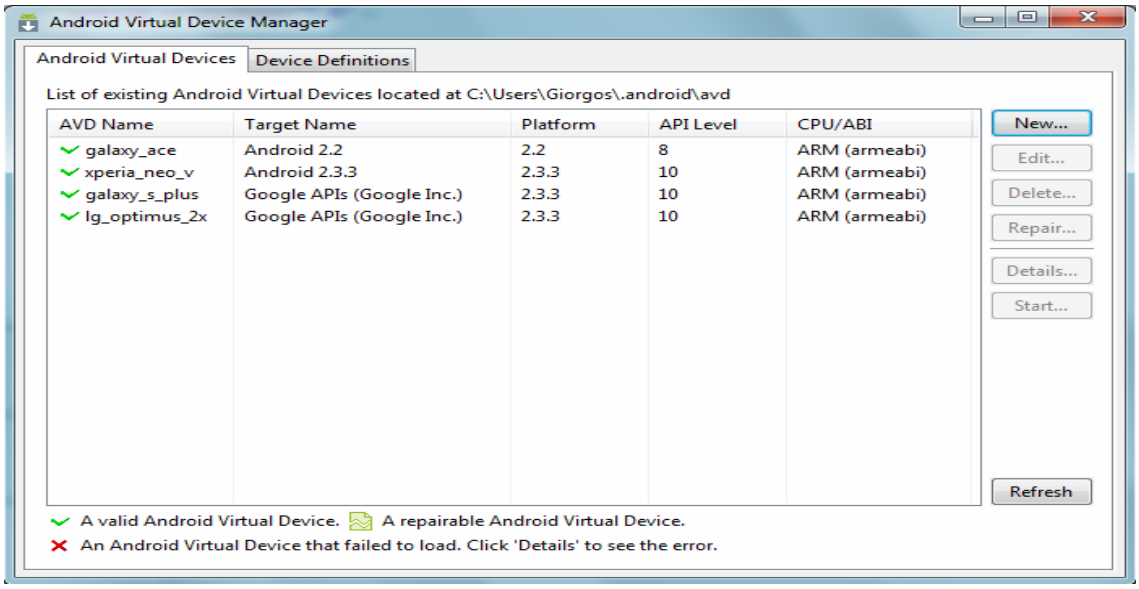
επιλέξουμε και μεταξύ ειδικών εκδόσεων της πλατφόρμας, μεταξύ των οποίων τις Google TV, και άλλων.

- *Έξτρα χώρος αποθήκευσης:* Εδώ αποθηκεύονται όλα τα δεδομένα της εφαρμογής, και επίσης μπορούμε να ορίσουμε μια εικονική κάρτα μνήμης ώστε να επεκτείνουμε τον αποθηκευτικό χώρο, όπως θα κάναμε και σε μια πραγματική συσκευή.

2.5.2.1 Δημιουργία διαφορετικών εικονικών συσκευών

Η δημιουργία εικονικών μηχανών είναι μια ιδιαίτερα εύκολη και γρήγορη διαδικασία. Η διαχείριση αυτών των συσκευών γίνεται από το γραφικό περιβάλλον της εφαρμογής AVD Manager, η οποία έρχεται μαζί με το SDK και στην περίπτωση που χρησιμοποιούμε το Eclipse μαζί με το ADT plugin, αυτή ενσωματώνεται στο γραφικό περιβάλλον του Eclipse.

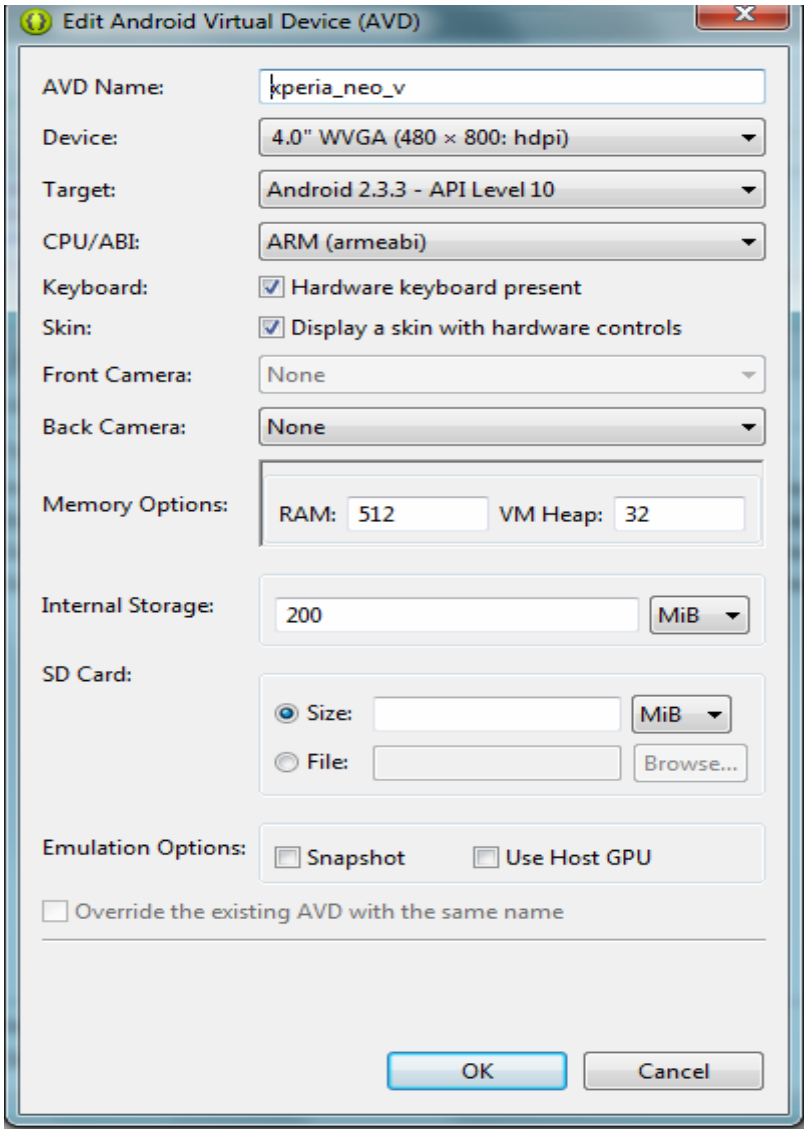
Το γραφικό περιβάλλον του AVD Manager είναι πολύ λιτό. Αποτελείτε από μια λίστα με τις εικονικές συσκευές που έχουμε δημιουργήσει και στη δεξιά πλευρά υπάρχουν 7 κουμπιά διαχείρισης των συσκευών μας.
Παρακάτω (Εικόνα 2.11) εμφανίζεται η λίστα με τις εικονικές συσκευές που δοκιμάστηκε η εφαρμογή FarmManager, κατά τη διάρκεια της δοκιμαστικής της φάσης. Φυσικά το προφίλ υλικού δεν είναι ακριβώς αυτό που διαθέτουν οι συσκευές που κατονομάζονται στην λίστα, αλλά οι διαστάσεις οθόνης και η έκδοση API, είναι ακριβής.



Εικόνα 2.11: Οι εικονικές συσκευές που δοκιμάστηκε η εφαρμογή

Εκτός από της παραπάνω εικονικές συσκευές δοκιμάστηκε και σε Galaxy s2, galaxy nexus και lg optimus 2x με λειτουργικό android 4.0.4 .

Πατώντας το κουμπί New εμφανίζεται ένα νέο παράθυρο δημιουργίας εικονικής συσκευής (Εικόνα 2.12), στο οποίο μπορούμε να ορίσουμε το όνομα, την έκδοση του Android που θέλουμε να τρέξει η συσκευή, την ανάλυση οθόνης, και τα υπόλοιπα χαρακτηριστικά του hardware που επιθυμούμε. Αφού πατήσουμε το κουμπί create AVD, η συσκευή μας είναι έτοιμη προς χρήση και προστίθεται στη λίστα μαζί με τις υπόλοιπες εικονικές συσκευές.

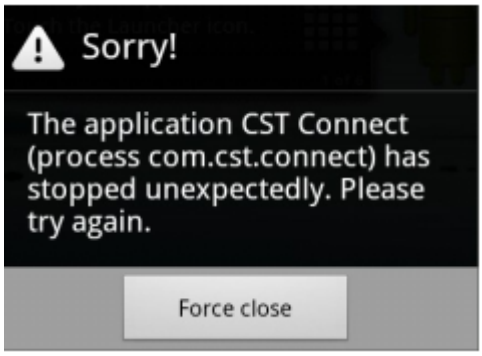


Εικόνα 2.12: Δημιουργία νέας εικονικής συσκευής

2.5.3 Εργαλείο καταγραφής συμβάντων – LogCat

Το Android διαθέτει ένα μηχανισμό καταγραφής συμβάντων, σκοπός του οποίου είναι η συλλογή και προβολή των αρχείων αποσφαλμάτωσης του συστήματος. Τα δεδομένα των διάφορων εφαρμογών αλλά και του λειτουργικού συστήματος συγκεντρώνονται σε μια σειρά από buffers, τους οποίους μετά μπορούμε να προβάλουμε και να φιλτράρουμε με την εντολή "logcat".

Στον προγραμματισμό υπάρχουν οι λεγόμενες "εξαιρέσεις" (exceptions), καταστάσεις δηλαδή που προκύπτουν όταν κάτι δεν πάει καλά, και αυτό έχει σαν αποτέλεσμα την διακοπή λειτουργίας του προγράμματος σε περίπτωση που δεν έχουμε φροντίσει να "χειριστούμε" την εξαίρεση. Συνηθισμένο παράδειγμα εξαίρεσης λειτουργίας του Android είναι η "NullPointerException", η οποία μας εμφανίζεται όταν προσπαθούμε να προσπελάσουμε κάποια μεταβλητή η αντικείμενο που έχει μηδενική (Null) τιμή.



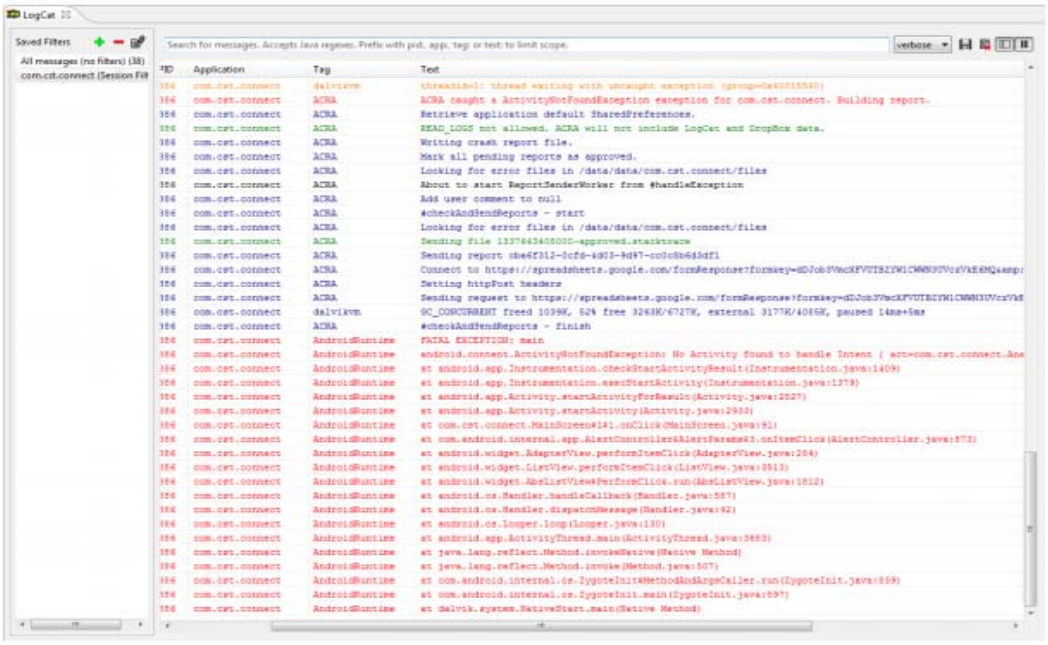
Εικόνα 2.13: Στιγμιότυπο μιας 'εξαίρεσης' σε ένα πρόγραμμα

Προγραμματιστικά βέβαια υπάρχει η δυνατότητα να βάλουμε δικλείδα ασφαλείας σε μερικά επίφοβα σημεία του κώδικα μας, και να σταματήσουμε την απότομη διακοπή λειτουργίας της εφαρμογής, εμφανίζοντας έναντι μόνο το μήνυμα σφάλματος στα logs του συστήματος.

Τα σφάλματα λειτουργίας μιας εφαρμογής στο Android, που προκύπτουν από εξαιρέσεις λειτουργίας, συνήθως προκαλούν τον άμεσο τερματισμό της εφαρμογής προβάλλοντας ένα παράθυρο με το όνομα της εφαρμογής που τερματίστηκε, και ένα απλό μήνυμα σφάλματος (Εικόνα 2.13), δίνοντας μας την "επιλογή" να πατήσουμε "Force Close".

Το LogCat λοιπόν είναι μια εντολή γραμμής εντολών η οποία μπορεί να χρησιμοποιηθεί μέσω του ADB για να δούμε τα debug logs της συσκευής που

δουλεύουμε, και άρα ως συνεπακόλουθο, της εφαρμογής που αναπτύσσουμε ώστε να εντοπίσουμε τις πηγές των σφαλμάτων, οι οποίες συνήθως αν όχι πάντα, είναι exceptions στον κώδικα μας. Το eclipse ενσωματώνει μία GUI έκδοση του LogCat (Εικόνα 2.14) για αποτελεσματικότερη αποσφαλμάτωση του κώδικα μας.



Εικόνα 2.14: Η γραφική απεικόνιση του Logcat όπως εμφανίζεται στο Eclipse IDE

Στην παραπάνω εικόνα βλέπουμε την τυπική διάταξη του logcat τη στιγμή μάλιστα που έχει συμβεί μια εξαίρεση λειτουργίας στην εφαρμογή μας και η οποία εμφανίζεται με κόκκινα χαρακτηριστικά γράμματα ξεκινώντας με τη φράση “Fatal Exception”. Οι κόκκινες γραμμές που προβάλλονται είναι όλες εξαιρετικά σημαντικές, κυρίως όμως μας ενδιαφέρουν αυτές που κάνουν λόγο για τη φύση της εξαίρεσης, και για τη γραμμή του κώδικα αλλά και την κλάση στην οποία συνέβη το σφάλμα! Στην συγκεκριμένη περίπτωση είχαμε μια “ActivityNotFoundException” η οποία συνέβη στην κλάση “MainScreen” στη γραμμή 91.

Με το παραπάνω απλό παράδειγμα μπορούμε να αντιληφθούμε την χρησιμότητα ύπαρξης του logcat. Άμεσα μας έκανε γνωστό το ακριβές σημείο του κώδικα μας που προκάλεσε το σφάλμα και μάλιστα λόγω τις φύσης της εξαίρεσης, μπορούμε να το διορθώσουμε σε χρόνο μηδέν. Η συγκεκριμένη εξαίρεση προκύπτει όταν μέσω ενός Intent κάνουμε μετάβαση από την μία Activity σε μία άλλη, και είτε συνήθως λόγω ορθογραφικού σφάλματος είτε πχ λόγω παράλειψης ενσωμάτωσης της Activity στο αρχείο AndroidManifest, η Activity που ορίζουμε στο Intent, δεν είναι διαθέσιμη.

Βέβαια υπάρχουν πολλές άλλες εξαιρέσεις (πάνω από 200 διαφορετικές) οι οποίες προκύπτουν σε διάφορες ανύποπτες στιγμές και χάρη στο logcat μπορούμε αρχικά να τις εντοπίσουμε και μετά να τις αντιμετωπίσουμε κάνοντας τις απαραίτητες διορθώσεις/αλλαγές στον κώδικα μας.

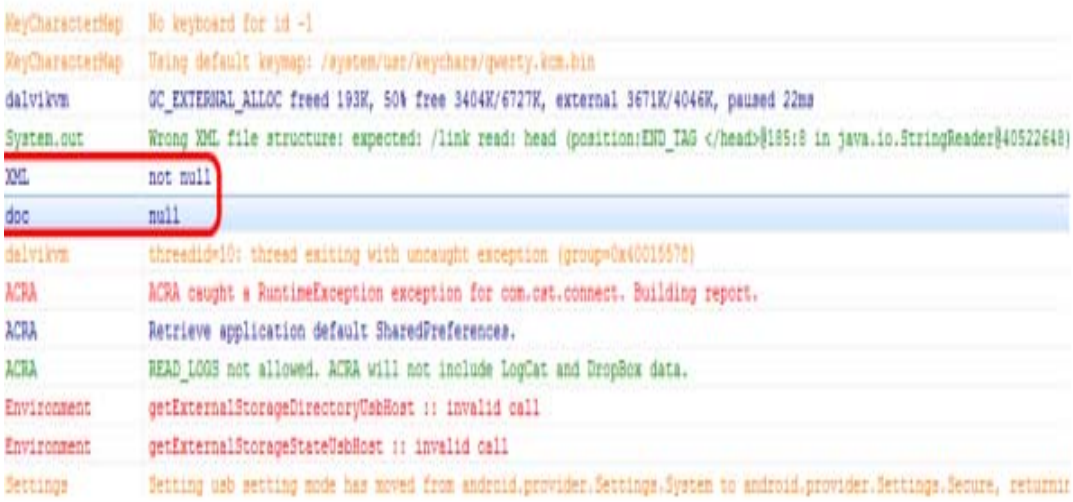
Φυσικά υπάρχουν και τα σφάλματα στον κώδικα τα οποία δεν προκαλούν αναγκαστικό κλείσιμο της εφαρμογής, αλλά παρόλα αυτά συμβάλουν στην μη σωστή λειτουργία της. Αυτού του είδους τα σφάλματα φυσικά δεν τα πιάνει ο compiler, ούτε εμφανίζονται με την μορφή που εμφανίζονται τα σφάλματα που προκύπτουν από εξαιρέσεις. Το αποτέλεσμα συνήθως αυτών των σφαλμάτων είναι μια κενή λίστα, μια λάθος τοποθετημένη εικόνα, κάποιο λάθος κείμενο, κλπ.

Πως αντιμετωπίζουμε λοιπόν ένα σφάλμα το οποίο δεν εμφανίζει κάποιο stack trace με ακριβές σημείο κώδικα προς διόρθωση, όπως συμβαίνει στην περίπτωση των εξαιρέσεων; Για το σκοπό μπορούμε να χρησιμοποιήσουμε το σύστημα καταγραφής του Android, για να πάρουμε τις πληροφορίες που θέλουμε. Η κλάση Log υπάρχει για αυτό ακριβώς το σκοπό. Περιλαμβάνει έναν αριθμό από διαφορετικές μεθόδους αναλόγως με τον τύπο του σφάλματος που ψάχνουμε. Η πιο συνηθισμένη μέθοδος είναι η **Log.d(),** όπου το **d** συμβολίζει τη λέξη Debug. Χρησιμοποιώντας την μέθοδο Log.d(String, String) μπορούμε να εμφανίσουμε στα logs της εφαρμογής πολλές πληροφορίες αποσφαλμάτωσης. Στην περίπτωση πχ μιας κενής λίστας αντικειμένων, μπορούμε να τοποθετήσουμε την μέθοδο με παραμέτρους: **Log.d("Debug", text);** Όπου "**Debug**" είναι η λέξη που θα εμφανιστεί δίπλα από το επιθυμητό Log, και **text** είναι η μεταβλητή τύπου String την τιμή τις οποίας θέλουμε να εμφανίσουμε στο Logcat για να δούμε τι περιέχει. Μια άλλη χρήση της μεθόδου είναι να εντοπίσουμε αν μια μεταβλητή που δεν είναι String, έχει τιμή η είναι κενή και μας επιστρέφει Null. Ακολουθεί παράδειγμα:

```
if (XML==null){
        Log.d ("XML", "null");
}
else{
        Log.d("XML", "not null");
}
if (doc==null){
        Log.d("doc", "null");
}
```

```
else{
        Log.d("doc", "not null");
}
```

Στις παραπάνω γραμμές κώδικα, μέσω μια απλής if else, ελέγχουμε αν οι μεταβλητές "XML" και "doc" είναι κενές και ο έλεγχος μας επιστρέφει "null" ή "not null". Σε κάθε περίπτωση λέμε στην Log.d να μας εκτυπώσει στο Logcat: το αντίστοιχο αποτέλεσμα (Εικόνα 2.15).



Εικόνα 2.15: Χρησιμοποιώντας την μέθοδο Log.d()

Αυτό βέβαια δεν μας επιστρέφει τις πραγματικές τιμές που μπορεί να έχει η μεταβλητές που ελέγξαμε, σε περίπτωση που δεν είναι Null, αλλά έτσι μπορούμε να εντοπίσουμε την υπαίτια για κάποιο NullPointerException που μπορεί να παίρνουμε από την εφαρμογή μας.

2.5.4 Dalvik Debug Monitor Server (DDMS)

Το Android συνοδεύεται μεταξύ των άλλων, από ένα εργαλείο αποσφαλμάτωσης το οποίο ονομάζεται Dalvik Debug Monitor Server (DDMS), και το οποίο παρέχει:

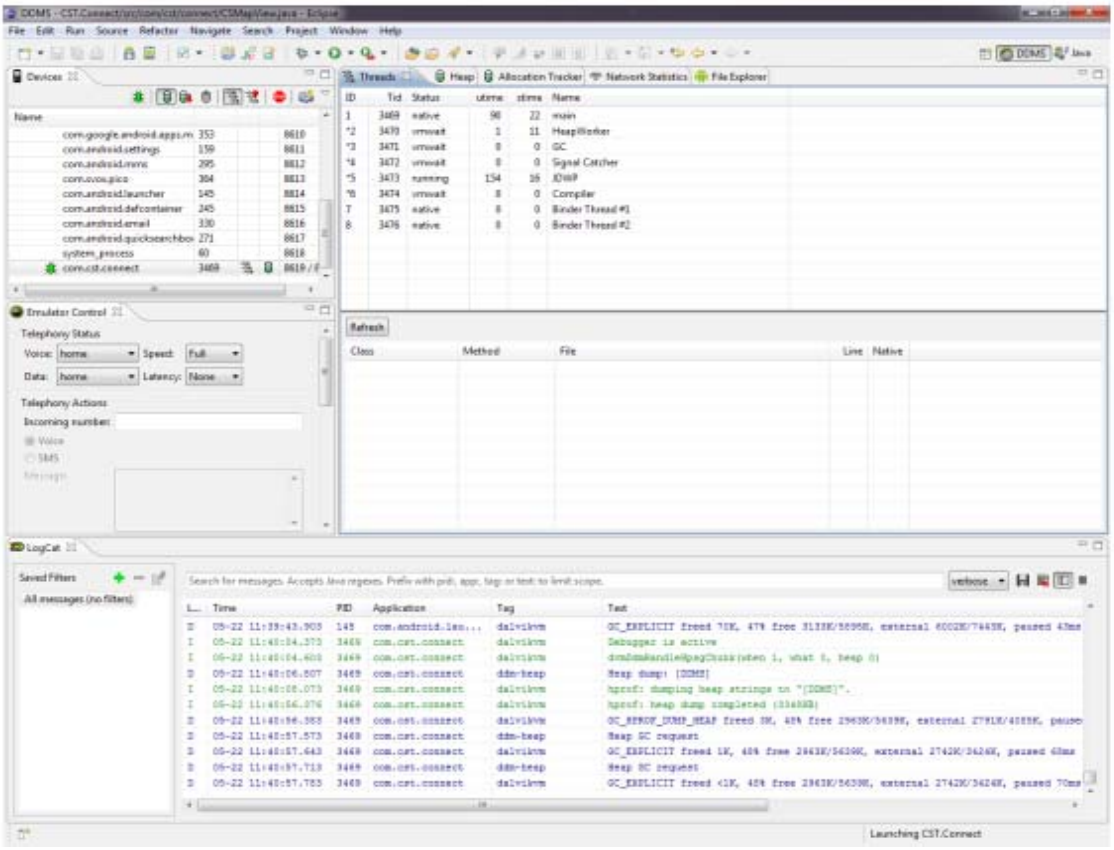
- Υπηρεσίες προώθησης θυρών (port forwarding)
- Λήψη εικόνας τις επιφάνειας εργασίας τις συσκευής
- Πληροφορίες για τις διεργασίες και τα νήματα (threads) τις συσκευής
- Το εργαλείο logcat

- Πληροφορίες δικτύου και εισερχομένων κλήσεων
- Δημιουργία ψευδών SMS και πληροφοριών τοποθεσίας,
- και άλλα πολλά

Το DDMS είναι ενσωματωμένο στο Eclipse και επίσης συμπεριλαμβάνεται στο Android SDK. Δουλεύει κανονικά είτε είναι συνδεδεμένο σε εικονική συσκευή μέσω του emulator είτε είναι συνδεδεμένο σε κανονική φυσική συσκευή. Από προεπιλογή αν είναι συνδεδεμένες δύο συσκευές ταυτόχρονα και η μία είναι εικονική, τότε αυτό θα επιλέξει σαν προεπιλεγμένη την εικονική. Σε προηγούμενο κεφάλαιο μιλώντας για την ασφάλεια στο Android, αναφέραμε ότι κάθε εφαρμογή τρέχει στην δική της εικονική μηχανή (VM). Κάθε μηχανή έχει και μία μοναδική θύρα επικοινωνίας στην οποία μπορεί να συνδεθεί κάποιο εργαλείο αποσφαλμάτωσης.

Κατά την εκκίνηση του το DDMS συνδέεται στο ADB. Αφού συνδεθεί κάποια συσκευή στο υπολογιστή μέσω ADB, αυτόματα δημιουργείτε μια υπηρεσία παρακολούθησης μεταξύ του DDMS και του ADB η οποία ειδοποιεί το DDMS πότε ξεκινάει και πότε σταματάει η λειτουργία μιας εικονικής μηχανής. Όταν η VM βρίσκεται σε λειτουργία, το DDMS παίρνει το process ID της VM και μέσω του ADB δημιουργεί μια σύνδεση με τον debugger της εικονικής μηχανής, μέσω του adb daemon.

Το DDMS αναθέτει μια μοναδική θύρα επικοινωνίας σε κάθε εικονική μηχανή στην συσκευή η στις συσκευές τις οποίες είναι συνδεδεμένο. Η ανάθεση ξεκινάει από την θύρα 8600 για την πρώτη VM και συνεχίζει για όσες εικονικές μηχανές, τρέχουν ταυτόχρονα στην συσκευή μας. Ισχύει ότι η κάθε VM έχει μία θύρα επικοινωνίας για debugging, αλλά το DDMS μπορεί να ακούσει σε πολλές θύρες ταυτόχρονα ώστε να λάβει δεδομένα από παντού.



Εικόνα 2.16: Το παράθυρο το DDMS

Στην παραπάνω εικόνα (Εικόνα 2.16) βλέπουμε ένα τυπικό παράθυρο του DDMS όπως αυτό εμφανίζεται στο Eclipse. Αριστερά φαίνονται οι συνδεδεμένες συσκευές (μία εικονική μέσω emulator) και από κάτω υπάρχει η λίστα με τις διαθέσιμες εικονικές μηχανές που τρέχουν στην συσκευή. Έχοντας επιλέξει την VM της εφαρμογής com.georkouk.farmManager μας έχουν γίνει διαθέσιμα το logcat output στο κάτω μέρος της οθόνης και στο δεξιό κομμάτι της οθόνης οι υπόλοιπες πληροφορίες που συλλέγει ο debugger από την εφαρμογή. Το DDMS είναι ίσως το χρησιμότερο από όλα τα debugging tools μιας και ενσωματώνει από τα πιο απλά (πχ logcat) μέχρι εξειδικευμένα εργαλεία debugging του Android το network statistics, και απευθύνεται κυρίως σε πιο έμπειρους developers οι οποίοι μπορούν να αξιολογήσουν αποτελεσματικότερα τις ενδείξεις που επιστρέφει το εργαλείο.

2.5.5 Application Crash Reporter for Android (ACRA)

Η συλλογή των debugging logs λοιπόν, είναι μια πολύ απλή διαδικασία η οποία πραγματοποιείται πολύ γρήγορα εφόσον συνδέσουμε την συσκευή μας μέσω

του Android Debug Bridge. Τι γίνεται όμως στην περίπτωση που θέλουμε να συλλέξουμε τα logs μιας συσκευής χρήστη η οποία εμφάνισε κάποιο σφάλμα; Όπως αντιλαμβανόμαστε δεν είναι εφικτό να του ζητήσουμε να συνδέσει τη συσκευή του μέσω ADB και αφού τα πάρει να μας τα στείλει με email! Και στην περίπτωση που τυχαίναμε σε περίπτωση συνεργάσιμου και γνώστη χρήστη, πόσο εύκολο θα ήταν να του ζητάγαμε να μας στέλνει συνεχώς το stack trace του logcat ώστε να εντοπίσουμε το σφάλμα και να το διορθώσουμε;

Οι μηχανικοί του Android έχουν μεριμνήσει για αυτό και γι' αυτό έχουν ενσωματώσει στις εφαρμογές που είναι συνδεδεμένες με το marketplace (έκδοση Android 2.2 και πάνω) έναν μηχανισμό απομακρυσμένης συλλογής logs. Τα stack traces των logs γίνονται προσβάσιμα μέσω της developer console του developer στο Google Play, και μέσω αυτών ο προγραμματιστής μπορεί να εντοπίσει το σφάλμα στον κώδικα του και να φροντίσει να το διορθώσει το συντομότερο δυνατό.

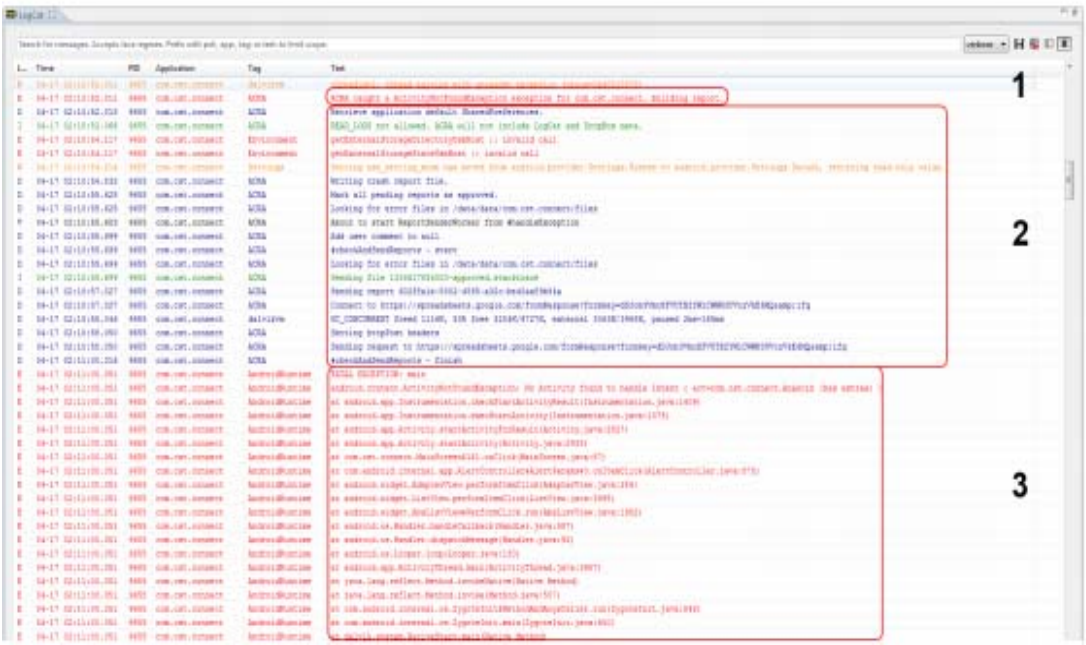
Τι γίνεται όμως στην περίπτωση που κάποιος developer δεν έχει κάνει ακόμη διαθέσιμη την εφαρμογή του στο κοινό, μέσω του Google Play; Η πιο συνηθέστερα, πως γίνεται να συλλέξει ο developers τα logs των χρηστών στην beta testing φάση της εφαρμογής του; Όπως είπαμε παραπάνω είναι εξαιρετικά ενοχλητικό και μη βολικό από την πλευρά μας να ζητάμε συνεχώς από τον χρήστη να μας τα στέλνει με email, και δυστυχώς το Android δεν έχει μεριμνήσει για αυτή την περίπτωση.

Το κενό λοιπόν αυτό έρχονται να καλύψουν κάποιες βιβλιοθήκες οι οποίες έχουν αναπτυχθεί με τα εργαλεία του Android SDK και σκοπός τους είναι ακριβώς αυτός: η συλλογή και αποστολή των stack traces από το κινητό του χρήστη σε εμάς για περεταίρω ανάλυση! Ένα παράδειγμα τέτοια βιβλιοθήκης είναι το bugsense το οποίο χρησιμοποιείτε από πολλές δημοφιλείς εφαρμογές και χρησιμοποιεί την βιβλιοθήκη ACRA. Η ACRA (αρχικά για "Application Crash Report for Android") είναι μια βιβλιοθήκη του Android η οποία αυτόματα συλλέγει τα δεδομένα των αναφορών σφάλματος και τα μας τα αποστέλλει σε μορφή εγγράφου GoogleDoc. Η βιβλιοθήκη είναι open source (Apache License 2.0) και φιλοξενείτε στην σελίδα
https://github.com/ACRA/acra
https://www.bugsense.com/dashboard
Τα χαρακτηριστικά της βιβλιοθήκης είναι τα εξής:

- Δυνατότητα παραμετροποίησης ειδοποιήσεων στην συσκευή του χρήστη (silent report, ειδοποίηση toast, ειδοποίηση στην Notification Bar + παράθυρο διαλόγου.

- Είναι δυνατόν να χρησιμοποιηθεί σε όλες τις εκδόσεις του Android και όχι μόνο από την 2.2 και πάνω όπως η επίσημη βιβλιοθήκη!
- Λεπτομερείς πληροφορίες για την συσκευή στην οποία συνέβη το σφάλμα λειτουργίας, πολύ περισσότερες από αυτές που δίνει η επίσημη βιβλιοθήκη!
- Μπορούμε να προσθέσουμε και τις δικές μας μεταβλητές περιεχομένου στις αναφορές που στέλνει η βιβλιοθήκη.
- Είναι εφικτή η αποστολή αναφορών χωρίς να έχει συμβεί κάποιο σφάλμα
- λειτουργίας της εφαρμογής!
- Δουλεύει σε όλες τις εφαρμογές είτε αυτές έχουν κυκλοφορήσει μέσω του Google Play, είτε βρίσκονται σε φάση beta testing οπότε η βιβλιοθήκη της Google δεν είναι διαθέσιμη.
- Μπορούμε να επιλέξουμε εκτός από κάποιο έγγραφο GoogleDoc η ACRA να μας στέλνει απευθείας τις αναφορές σε κάποιον http server, είτε να μας τις στέλνει απευθείας μέσω email.
- Τα reports που έχουμε ρυθμίσει να αποθηκεύονται σε κάποιο έγγραφο GoogleDoc, μπορεί να γίνει προσβάσιμο άμεσα από μια ομάδα developers ώστε να αντιμετωπισθεί γρηγορότερα το σφάλμα.

Όπως βλέπουμε η ACRA έχει πολλές εξαιρετικές δυνατότητες και πραγματικά λύνει τα χέρια του developer όσον αφορά το κομμάτι του remote debugging. Η πιο συνηθισμένη χρήση της βιβλιοθήκης είναι τα silent reports ώστε ο χρήστης πέρα από το παράθυρο του Force Close (Εικόνα 2.13) δεν παρατηρεί κάτι άλλο. Μετά το silent report ενημερώνεται άμεσα το έγγραφο GoogleDoc που έχουμε δημιουργήσει και συνδέσει με τη βιβλιοθήκη και μετέπειτα μπορούμε να επιλέξουμε να ενημερωνόμαστε άμεσα με email κάθε φορά που γίνεται αλλαγή στο έγγραφό μας. Με λίγα λόγια ένα σύστημα άμεσου bug reporting! Φυσικά μπορεί να χρησιμοποιηθεί ταυτόχρονα με την επίσημη βιβλιοθήκη της Google, αλλά μάλλον η δεύτερη είναι περιττή.



Εικόνα 2.17: Η λειτουργία της ACRA όπως φαίνεται στο LogCat

Στην παραπάνω εικόνα βλέπουμε τι συμβαίνει στην συσκευή του χρήστη όταν πραγματοποιείται κάποιο κρίσιμο σφάλμα λειτουργίας. Καταρχήν (βήμα 1) η ACRA "πιάνει" την εξαίρεση (exception) που προέκυψε και ξεκινάει άμεσα να χτίζει την αναφορά της. Στο δεύτερο βήμα ακολουθεί η διαδικασία συλλογής δεδομένων από όλες τις πηγές που έχουμε ορίσει και μετά η ACRA αποστέλλει την αναφορά στο έγγραφό Google Doc που έχουμε ορίσει. Στο τρίτο και τελευταίο βήμα η ACRA αφήνει το σύστημα να συνεχίσει τη λειτουργία του και να εμφανίσει στα logs το stack trace του σφάλματος που προέκυψε, μαζί με το παράθυρο Force Close που εμφανίζεται στον χρήστη. Αυτή η διαδικασία είναι εντελώς αόρατη στον χρήστη και από ότι φαίνεται στην εικόνα παραπάνω (Εικόνα 2.17) καθυστερεί ελάχιστα το σύστημα μιας και διαρκεί μόλις 8-10ms!

2.6 Κατακερματισμός του Android

Στις προηγούμενες σελίδες έγινε λεπτομερή αναφορά σε όλες τις προκλήσεις που έχουν να αντιμετωπίσουν οι developers του Android, και επίσης έγινε λεπτομερής ανάλυση της διαδικασίας ανάπτυξης εφαρμογών αλλά και των διαθέσιμων εργαλείων ανάπτυξης, δοκιμής και αποσφαλμάτωσης που παρέχει το

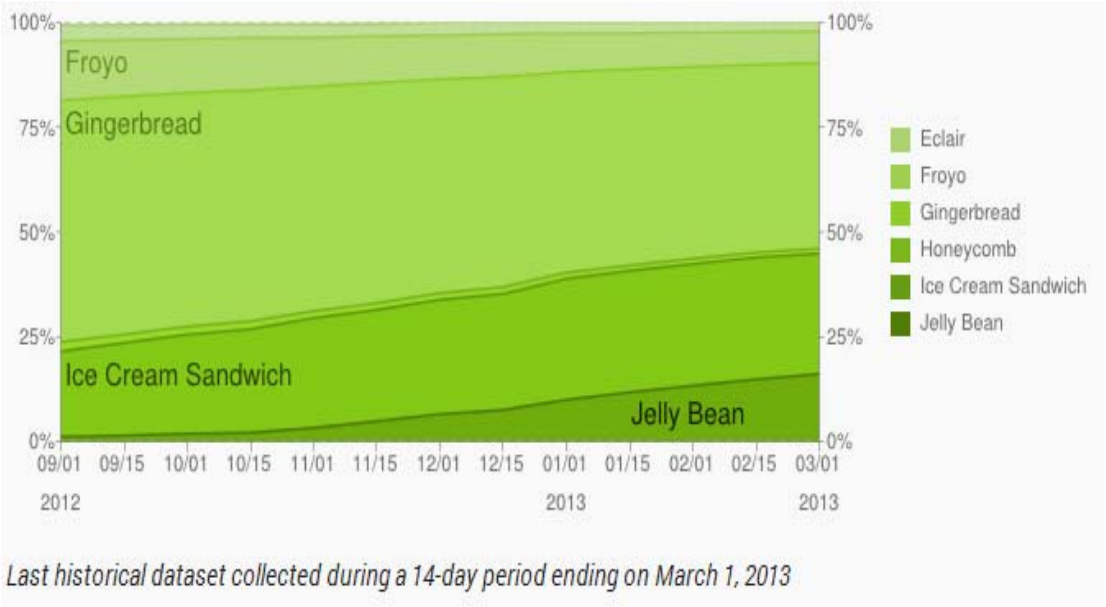
Android και η κοινότητα του. Σε πολλά σημεία έγινε αναφορά στο πρόβλημα κατακερματισμού του Android. Τι είναι αυτό όμως;

Όπως γνωρίζουμε το Android ανήκει και αναπτύσσεται από την Google η οποία ανά τακτά διαστήματα χρονικά διαστήματα κάνει διαθέσιμη την νέα έκδοση με τη μορφή πηγαίου κώδικα στην κοινότητα. Ο κώδικας είναι open source φυσικά και οποιοσδήποτε μπορεί να τον κατεβάσει τοπικά και είτε να τον χρησιμοποιήσει για να φτιάξει μια έκδοση του λειτουργικού για λειτουργία σε κάποια συσκευή, είτε να συνεισφέρει στην περεταίρω ανάπτυξη του. Αυτή είναι και η δύναμη του λειτουργικού συστήματος! Η open source φύση του έχει προσελκύσει δεκάδες χιλιάδες ταλαντούχους developers ανά τον πλανήτη όπως και μεγάλες εταιρίες τηλεπικοινωνιών, οι οποίες επενδύουν εκατομμύρια δολάρια για ανάπτυξη συσκευών με λειτουργικό σύστημα Android!

Το Android λοιπόν χάρη στη δυναμική του έχει κατακτήσει ένα μεγάλο μερίδιο της αγοράς και είναι διαθέσιμο σε δεκάδες συσκευές κινητής τηλεφωνίας. Συνήθως οι κατασκευαστές κυκλοφορούν τις συσκευές τους είτε με την νεότερη έκδοση του λειτουργικού συστήματος, είτε με την αμέσως προηγούμενη μαζί με την υπόσχεση της αναβάθμισης στην νεότερη όταν αυτή έχει δοκιμαστεί επαρκώς από την εταιρία. Δυστυχώς όμως το θέμα των αναβαθμίσεων είναι αρκετά “ευαίσθητο”, διότι οι εταιρίες με στόχο την ευκολία και το κέρδος δεν κυκλοφορούν επαρκείς αναβαθμίσεις. Αντί αυτού προτιμούν να διαθέτουν τις νέες εκδόσεις λειτουργικού στις νεότερες συσκευές τους, με σκοπό να τις προτιμήσουν οι χρήστες, αφήνοντας τις παλιές συσκευές με την υπόσχεση τις αναβάθμισης.

Το κενό των εταιριών έρχεται να καλύψει η τεράστια κοινότητα developers του Android οι οποίοι όντας κάτοχοι “παραμελημένων“ συσκευών φροντίζουν να παρέχουν οι ίδιοι τις πολυπόθητες αναβαθμίσεις στα κινητά τους και άρα και συνήθως και των υπόλοιπων χρηστών! Αυτές οι εκδόσεις του Android είναι γνωστές ως “Custom Roms” και φυσικά υποστηρίζονται αποκλειστικά από τους developers τους και όχι από τις κατασκευάστριες εταιρίες των κινητών.

Παρά την τεράστια υποστήριξη της κοινότητας ο κατακερματισμός στο Android παραμένει υψηλός και εκτός και αν αλλάξει κάτι στον τρόπο των αναβαθμίσεων, τότε ίσως στο μέλλον να διογκωθεί ακόμη περισσότερο. Στο παρακάτω γράφημα φαίνεται καλύτερα το πρόβλημα.



Εικόνα 2.18: Χρονικό πλαίσιο υιοθέτησης των εκδόσεων του Android

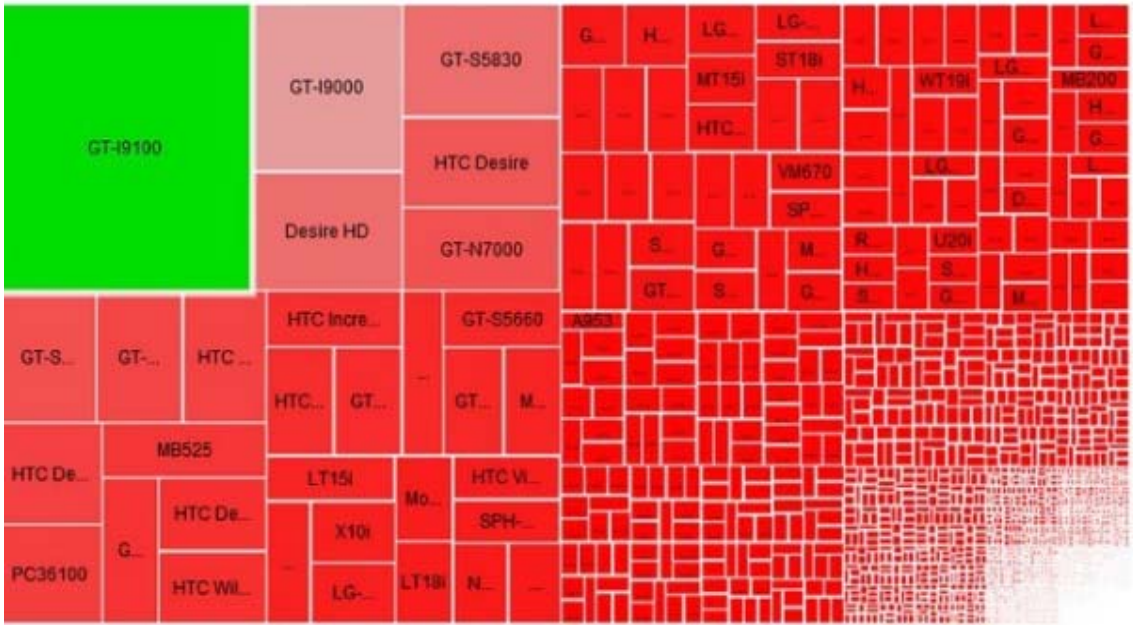
Όπως βλέπουμε το μοτίβο υιοθέτησης των νέων εκδόσεων του Android είναι αρκετά αργό, και αυτό οφείλετε κυρίως στην αδιαφορία των κατασκευαστών να παρέχουν στις παλαιότερες συσκευές υποστήριξη. Οι περισσότερες συσκευές υποστηρίζουν τις νεότερες εκδόσεις του Android χάρη στη συνεισφορά της κοινότητας, αλλά οι περισσότεροι χρήστες δεν έχουν τις απαραίτητες γνώσεις η εμπειρία για να χρησιμοποιήσουν κάποια από αυτές τις custom εκδόσεις.

Φυσικά όλες αυτές οι συσκευές δεν γίνεται να αγνοηθούν από τους προγραμματιστές εφαρμογών του Android γιατί αποτελούν μέρος τις πίτας και άρα πιθανοί χρήστες των εφαρμογών τους. Παραπάνω αναφερθήκαμε λεπτομερώς στις προκλήσεις προγραμματισμού σε διαφορετικά API και τα προβλήματα συμβατότητας που προκαλούν στον προγραμματιστή αυτά. Είναι συχνό φαινόμενο πολλές εταιρίες να επιλέγουν πρώτα το iOS για να αναπτύξουν εφαρμογές, με το Android να ακολουθεί δεύτερο και ο κύριο λόγος φυσικά είναι η υποστήριξη των εκατοντάδων συσκευών που τρέχουν τις διάφορες εκδόσεις του λειτουργικού συστήματος.

Μεγάλες εταιρίες ανάπτυξης mobile εφαρμογών μάλιστα αποσύρουν την υποστήριξη τους στο Android γιατί το θεωρούν ασύμφορο! Παράδειγμα αυτής της περίπτωσης είναι το παιχνίδι “Battleheart” της εταιρίας Mika Mobile που είναι διαθέσιμο για Android και iOS. Κάποιος μπορεί εύκολα αν αντιληφθεί ότι εφόσον μια μεγάλη εταιρία με πολλούς developers βρίσκει ασύμφορη για ανάπτυξη, την πλατφόρμα του Android, ο απλός developer που προσπαθεί να βγάλει ένα μικρό εισόδημα από εκεί, βρίσκεται σε ακόμη δυσκολότερη θέση.

2.6.1 Στατιστικά κατακερματισμού του Android από την εφαρμογή OpenSignalMaps

Για να γίνει πιο αντιληπτό το μέγεθος του προβλήματος, θα αναφερθούμε σε μια πρόσφατη δημοσίευση blog του κατασκευαστή της εφαρμογής OpenSignalMaps. Οι developers της συγκεκριμένης εφαρμογής συνέλεξαν στατιστικά στοιχεία όλων των συσκευών που χρησιμοποίησαν τις υπηρεσίες τους για διάστημα 6 μηνών και αποφάσισαν να μοιραστούν τα αποτελέσματα με τη μορφή γραφικών παραστάσεων.

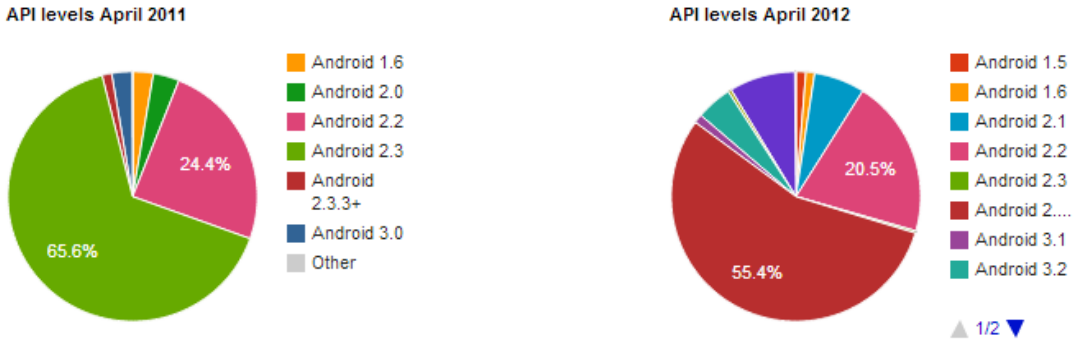


Εικόνα 2.19: Γραφική απεικόνιση συσκευών που χρησιμοποίησαν το OpenSignalMaps

Το παραπάνω εντυπωσιακό σχεδιάγραμμα είναι η γραφική αναπαράσταση της κατανομής του συνόλου των διαφορετικών συσκευών οι οποίες χρησιμοποίησαν την εφαρμογή σε διάστημα 6 μηνών. Στο σχεδιάγραμμα (εικόνα 2.19) απεικονίζονται περίπου 4000 διαφορετικές συσκευές, με κυρίαρχη το Samsung Galaxy S II! Φυσικά ο πραγματικός αριθμός συσκευών είναι αρκετά μικρότερος, αλλά εμφανίζονται πολύ περισσότερες λόγω της χρήσης custom roms οι οποίες μερικές φορές αλλάζουν την έκδοση build της συσκευής. Αυτό εξηγεί γιατί περίπου 1400 από τις παραπάνω συσκευές εμφανίζονται μόνο μία φορά στα στατιστικά χρήσης της εφαρμογής.

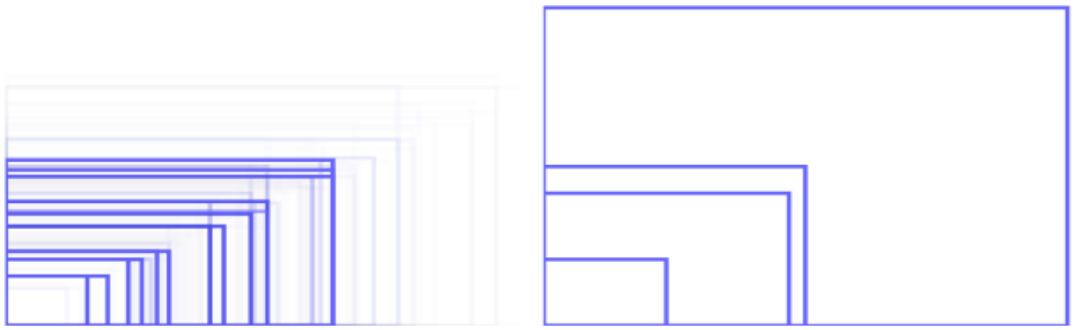
Οι μετρήσεις της εταιρίας φυσικά δεν σταμάτησαν στον αριθμό των συσκευών οι οποίες χρησιμοποίησαν την εφαρμογή, αλλά και στο λειτουργικό σύστημα (API Level) που υποστήριζαν αυτές. Τα 2 γραφήματα, αν και παλιά μπορούμε να δούμε αυτό που θέλουμε, που ανάρτησε η εταιρία έχουν απόσταση ενός

έτους το ένα από το άλλο και μας δείχνουν περίπου την ίδια κατανομή που είδαμε και στο σχεδιάγραμμα του Android Market. Κυρίαρχη έκδοση πέρσι ήταν η 2.3 ενώ φέτος κυρίαρχη έκδοση είναι η 2.3.3, στην οποία έχουν διορθωθεί κάποια bugs σε σχέση με την 2.3 και ουσιαστικά έχει ελάχιστες διαφορές στα APIs της. Επίσης πέρσι οι 2 πιο δημοφιλείς εκδόσεις του Android, καταλάμβαναν το 90% της αγοράς ενώ φέτος με την έλευση του Android 4.0 και 4.1 αυτό το ποσοστό έχει πέσει γύρω στο 75%.



Εικόνα 2.20: Κατανομή των APIs (Απρίλιος 2011 - Απρίλιος 2012)

Τέλος η εταιρία δημοσίευσε και τα στατιστικά στοιχεία των διαφορετικών αναλύσεων των συσκευών που χρησιμοποίησαν την εφαρμογή στο διάστημα των 6 μηνών και κατόπιν την σύγκριναν με το εύρος οθονών των συσκευών που χρησιμοποιούν iOS. Στην περίπτωση του Android μάλιστα ελαχιστοποίησε την κατανομή των διαστάσεων, τονίζοντας τις 13 πιο δημοφιλείς. Σε σχέση με τις μόλις 4 διαφορετικές αναλύσεις των συσκευών του iOS εύκολα αντιλαμβανόμαστε την πρόκληση υποστήριξης όλων αυτών των διαφορετικών αναλύσεων οθόνης ταυτόχρονα.



Εικόνα 2.21: Σύγκριση αναλύσεων Android vs iOS

ΚΕΦΑΛΑΙΟ 3

Δομή εφαρμογών Android

3.1 Συστατικά στοιχεία εφαρμογών

Με τον όρο αυτό εννοούμε όλα τα απαραίτητα δομικά στοιχεία μιας εφαρμογής Android. Το κάθε στοιχείο από αυτά είναι στην ουσία ένας τρόπος πρόσβασης του λειτουργικού συστήματος στην εφαρμογή μας. Μπορούμε να τα χωρίσουμε σε 4 βασικά στοιχεία:

- Δραστηριότητες (Activities)
- Υπηρεσίες (Services)
- Πάροχοι περιεχομένου (content provider)
- Καθολικοί παραλήπτες μηνυμάτων (broadcast receivers)

3.1.1 Activities

Αποτελούν το βασικότερο στοιχείο κάθε εφαρμογής. Η κάθε activity αποτελεί μια διαφορετική οθόνη στην εφαρμογή μας, με την οποία ο χρήστης αλληλεπιδρά. Αποτελεί στην ουσία το γραφικό περιβάλλον που βλέπει ο χρήστης.

Κάθε εφαρμογή αποτελείται συνήθως από αρκετές activities, οι οποίες συνδέονται μεταξύ τους, δηλαδή κάθε activity μπορεί να καλέσει κάποια άλλη. Η πρώτη οθόνη που θα δει ο χρήστης μόλις ανοίξει η εφαρμογή αποτελεί την κεντρική activity (main), από την οποία προέρχονται όλες οι υπόλοιπες. Τη στιγμή που η εφαρμογή καλεί μια νέα activity, ο χρήστης βλέπει την νέα οθόνη και με το πλήκτρο back μπορεί να επιστρέψει στην προηγούμενη, δηλαδή έχουμε μια μορφή στοίβας LIFO. Αν δηλαδή η activity A καλέσει την activity B και αυτή την activity C (A→B→C) τότε από την activity C με το πλήκτρο back επιστρέφουμε στην B και από εκεί στην A. Φυσικά μπορούμε κάποιες activities να επιλέξουμε να μην αποθηκεύονται στην στοίβα ή ακόμα και να αδειάσουμε τελείως την στοίβα. Δηλαδή αν στην περίπτωσή μας αποφασίσουμε να μην αποθηκεύσουμε την B τότε από την C με το πλήκτρο back θα μεταφερθούμε στην A. Από την A αν πατήσουμε το πλήκτρο back θα βγούμε τελείως από την εφαρμογή μας.

Κάθε activity πρέπει να υλοποιεί την μέθοδο onCreate() η οποία καλείται τη στιγμή που ξεκινάει η activity. Η μέθοδος αυτή καλείται μόνο όταν ανοίγουμε μια νέα activity είτε από κάποιο κουμπί είτε από κάποια επιλογή, δηλαδή δεν εκτελείται αν επιστρέψουμε σε μια activity με το πλήκτρο back. Όταν επιστρέφουμε σε κάποια προηγούμενη activity βλέπουμε το στιγμιότυπο που αφήσαμε και η activity δεν ξαναδημιουργείται. Επίσης μέσα στην μέθοδο αυτή πρέπει να δημιουργηθεί και η διεπαφή χρήστη (User Interface ή UI), η οποία δηλώνεται με την εντολή setContentView(). Θα δούμε αναλυτικά παρακάτω για το UI.

Για να ξεκινήσουμε μια activity μέσα από μια άλλη activity χρησιμοποιούμε τις παρακάτω εντολές:

```
Intent in = new Intent (this, MyActivity.class);
startActivity (in);
```

Για να τερματίσουμε μια activity, αρκεί να χρησιμοποιήσουμε την εντολή finish(). Σε γενικές γραμμές πάντως το android διαχειρίζεται από μόνο του τον κύκλο ζωής της κάθε activity, οπότε δε χρειάζεται να τις τερματίζουμε εμείς, παρά μόνο όταν θέλουμε οπωσδήποτε να μη μπορεί ο χρήστης να επιστρέψει σε αυτό το στιγμιότυπο της activity. Ο κύκλος ζωής και οι αντίστοιχες μέθοδοι των activities φαίνονται στην εικόνα 1.15

3.1.2 Services

Μια υπηρεσία (service) αποτελείται από ένα στοιχείο της εφαρμογής, το οποίο τρέχει στο παρασκήνιο (background) και μπορεί να εκτελέσει αρκετές και χρονοβόρες λειτουργίες. Ειδοποιός διαφορά με την activity είναι ότι η service δεν παρέχει UI. Επίσης η service μπορεί να εκτελείται ακόμα και όταν τρέξουμε κάποια διαφορετική εφαρμογή από αυτήν που την ξεκίνησε.

Παράδειγμα υπηρεσίας αποτελεί η εφαρμογή αναπαραγωγής μουσικής. Όταν ακούμε μουσική, είναι πιθανό να θέλουμε να συνεχίσουμε να την ακούμε ακόμα και όταν τρέξουμε κάποια άλλη εφαρμογή και δίχως να απαιτούμε γραφικό περιβάλλον για την αναπαραγωγή, Επιπλέον όλη η επικοινωνία με το δίκτυο θέλουμε να συνεχίσει όταν ανοίξουμε κάποια άλλη εφαρμογή και να μην σταματήσει, άρα θα πρέπει να λειτουργεί σαν service.

Δύο βασικές λειτουργίες που σχετίζονται με τις υπηρεσίες είναι η startActivity() και η bindService(). Η πρώτη καλείται από κάποια activity για να

ξεκινήσει η service και θα συνεχίσει να εκτελείται ακόμα και αν η activity που την κάλεσε τερματιστεί. Για το λόγο αυτό η service δεν επιστρέφει κάποιο αποτέλεσμα στην activity που την κάλεσε, αλλά εκτελεί κάποια λειτουργία όπως το κατέβασμα κάποιου αρχείου από το internet. Η δεύτερη λειτουργία 'δένει' τη service με κάποιο άλλο στοιχείο της εφαρμογής πχ με μια activity έτσι ώστε να υπάρχει αλληλεπίδραση μεταξύ των στοιχείων αυτών όπως πχ αποστολή και λήψη αποτελεσμάτων. Στην περίπτωση αυτή αν τερματιστεί το στοιχείο που έχει 'δεθεί' με την service, τότε τερματίζεται και η service.

Προηγουμένως αναφέραμε ότι η service δεν παρέχει διεπαφή χρήστη, επομένως γεννιέται το ερώτημα, πως θα ενημερώνεται ο χρήστης για κάποιο αποτέλεσμα της υπηρεσίας. Υπάρχουν 2 τρόποι: η ειδοποίηση Toast και η ειδοποίηση μέσω της status bar. Οι ειδοποιήσεις αυτές αναλύονται στην παράγραφο 3.2.4 καθώς αφορούν το UI.

Σε αντίθεση με τις activities, το λειτουργικό σύστημα δεν διαχειρίζεται τον κύκλο ζωής των services (εκτός και αν υπάρχει έλλειψη υπολογιστικών πόρων), επομένως η κάθε service θα πρέπει να δηλώνει το πότε θα σταματήσει με την εντολή stropSelf().

3.1.3 Content Providers

Οι content providers διαχειρίζονται αποθηκευτικούς χώρους για δεδομένα, οι οποίοι είναι προσπελάσιμοι από οποιαδήποτε εφαρμογή. Αποτελούν το μοναδικό τρόπο για αποθήκευση δεδομένων τα οποία θέλουμε να είναι προσβάσιμα από άλλες εφαρμογές. Για παράδειγμα μια εφαρμογή τηλεφωνικού καταλόγου θα θέλαμε να αποθηκεύει σε συγκεκριμένο χώρο τις επαφές μας, ώστε αργότερα αν θελήσουμε να χρησιμοποιήσουμε διαφορετική εφαρμογή να μπορεί αυτή να εντοπίσει και να επεξεργαστεί τα ήδη υπάρχοντα δεδομένα. Παρόμοια παραδείγματα αποτελούν τα αρχεία ήχου, βίντεο, οι εικόνες κλπ τα οποία θέλουμε να αποθηκεύονται σε κάποιους προεπιλεγμένους χώρους, ώστε να είναι προσβάσιμα από περισσότερες εφαρμογές. Στο πλαίσιο αυτό υπάρχει ενσωματωμένη βάση δεδομένων στο λειτουργικό σύστημα Android (SQLite database) , στην οποία μπορούν να αποθηκεύουν ή να διαβάζουν δεδομένα οι content providers.

3.1.4 Broadcast Receivers

Οι broadcast receivers ενεργοποιούνται μόλις ειδοποιηθούν από το λειτουργικό σύστημα για κάποιο γεγονός. Τέτοια γεγονότα μπορεί να είναι ότι η οθόνη έχει σβήσει, ότι η στάθμη της μπαταρίας είναι χαμηλή κλπ. Για παράδειγμα αρκετές εφαρμογές λήψης φωτογραφιών απενεργοποιούν το φλάς σε περίπτωση χαμηλής μπαταρίας για λόγους εξοικονόμησης ενέργειας. Ενημερώνονται δηλαδή από τον broadcast receiver για το γεγονός αυτό και πράττουν ανάλογα. Οι broadcast receivers δεν παρέχουν UI, αλλά μπορούν να ενημερώσουν το χρήστη για κάποιο γεγονός μέσω των status bar notifications.

3.2 User Interface (UI)

Η διεπαφή χρήστη έχει τεράστια σημασία για κάθε εφαρμογή. Αποτελεί την τελική εικόνα που βλέπει ο χρήστης, το γραφικό περιβάλλον στο οποίο θα περιηγείται και ενεργοποιεί όλες τις λειτουργίες της εφαρμογής. Το UI και η λειτουργικότητα της εφαρμογής αποτελούν αλληλένδετα στοιχεία και χωρίς το ένα δε μπορεί να υπάρξει το άλλο. Πολλές φορές μάλιστα είναι δυσκολότερος ο σχεδιασμός ενός όμορφου και εύχρηστου περιβάλλοντος εργασίας, παρά η ίδια η λειτουργικότητα της εφαρμογής. Καθίσταται σαφές λοιπόν, ότι απαιτεί μεγάλη προσπάθεια και προσοχή η δημιουργία ενός γραφικού περιβάλλοντος που θα προσελκύει τους χρήστες και θα τους ωθεί να χρησιμοποιούν μια συγκεκριμένη εφαρμογή έναντι μιας άλλης με τις ίδιες λειτουργίες.

3.2.1 Layout

Σε κάθε οθόνη της εφαρμογής, πρωταρχικό στοιχείο του γραφικού περιβάλλοντος αποτελεί η διάταξη των γραφικών στοιχείων ή layout. Το layout περιλαμβάνει όλα τα γραφικά στοιχεία της οθόνης, τα οποία μπορεί να είναι διατεταγμένα σε επιμέρους layouts. Υπάρχουν 4 είδη layout:

- LinearLayout (γραμμική διάταξη)
- RelativeLayout (σχετική διάταξη)
- FrameLayout (διάταξη πλαισίου)
- TableLayout (διάταξη πίνακα)

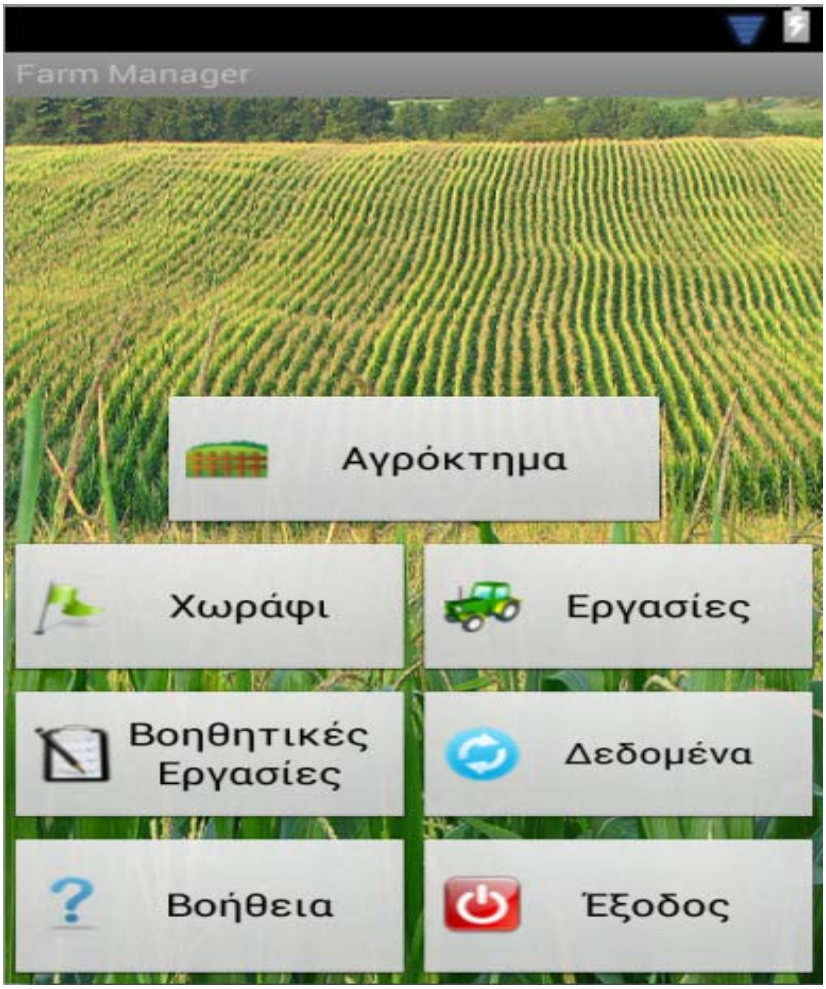
Το *LinearLayout* αποτελεί διάταξη στοιχείων σε οριζόντια ή κατακόρυφη σειρά. Αν δηλαδή δηλώσουμε 3 στοιχεία A,B,C μέσα σε ένα οριζόντιο LinearLayout τότε τα στοιχεία αυτά θα εμφανίζονται στην οθόνη σε μια οριζόντια διάταξη με τη σειρά που τα δηλώσαμε το ένα δίπλα στο άλλο. Το *RelativeLayout* μας δίνει περισσότερη ελευθερία στη δήλωση των γραφικών στοιχείων, με την έννοια ότι κάθε στοιχείο μπορούμε να επιλέξουμε να το εμφανίσουμε σε συγκεκριμένο σημείο της οθόνης, πχ στην αρχή, στο κέντρο, στο τέλος ή να επιλέξουμε τη θέση του σε σχέση με κάποιο άλλο στοιχείο. Το *FrameLayout* αποτελεί την απλούστερη διάταξη στοιχείων. Είναι απλά ένας κενός χώρος τον οποίο μπορούμε να γεμίσουμε με κάποιο αντικείμενο πχ μια εικόνα. Για το λόγο αυτό συνήθως χρησιμοποιείται σαν ρίζα (root) στο δέντρο των γραφικών στοιχείων της οθόνης. Το *TableLayout* όπως φανερώνει και το όνομά του αποτελεί διάταξη πίνακα, δηλαδή μπορεί να διατάσσει τα παιδιά του (children) σε σειρές και στήλες. Σε όλα τα παραπάνω στοιχεία μπορούμε να ρυθμίσουμε αρκετές παραμέτρους όπως μέγεθος (πλάτος και ύψος), οριζόντια ή κατακόρυφη διάταξη, βαρύτητα (layout gravity) κ.τ.λ.

Υπάρχουν 2 τρόποι για να δηλώσουμε τα layouts (όπως και κάθε άλλο γραφικό στοιχείο) της εφαρμογής μας: μέσω αρχείων xml ή μέσα στις activities της εφαρμογής μας. Τα αρχεία xml αποτελούν στατικό τρόπο δημιουργίας των γραφικών στοιχείων. Τα αποθηκεύουμε σε συγκεκριμένο φάκελο του project μας και τα καλούμε για δημιουργία του γραφικού περιβάλλοντος μέσα από τις activities. Παρουσιάζουν δενδρική δομή για εύκολη συγγραφή και κατανόηση του περιεχομένου τους. Πολλές φορές ωστόσο δε γνωρίζουμε εξαρχής τη διάταξη που θα χρησιμοποιήσουμε, διότι ίσως να εξαρτάται από ορισμένες επιλογές του χρήστη. Στις περιπτώσεις αυτές δημιουργούμε δυναμικά τα layouts μέσα στις activities και ορίζουμε εκεί τις παραμέτρους αυτών. Ωστόσο ο πιο εύκολος και συνηθέστερος τρόπος είναι (αν έχουμε τη δυνατότητα) να δημιουργήσουμε για κάθε οθόνη ένα διαφορετικό xml αρχείο που θα περιλαμβάνει τη διάταξη όλων των γραφικών της στοιχείων και να το καλέσουμε μέσα από την αντίστοιχη activity. Ακολουθούν 2 παραδείγματα διάταξης των γραφικών στοιχείων της οθόνης.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/start_menu3"
    android:orientation="vertical" >
```

```
<LinearLayout
    android:id="@+id/l1"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:layout_alignParentBottom="true"
    android:layout_marginTop="5dp"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/btStartMenuAboutUs"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0.5"
        android:textSize="17sp"
        android:drawableLeft="@drawable/help"
        android:text="@string/stHelp" >
    </Button>

    <Button
        android:id="@+id/btStartMenuExit"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0.5"
        android:textSize="17sp"
        android:drawableLeft="@drawable/exit"
        android:text="@string/stStartMenuExit" >
    </Button>
</LinearLayout>

<LinearLayout
    android:id="@+id/l2"
    android:layout_width="match_parent"
    android:layout_height="75dp"
    android:layout_above="@id/l1"
    android:layout_marginTop="5dp"
    android:gravity="center"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/btStartMenuSettings"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0.5"
        android:textSize="17sp"
        android:drawableLeft="@drawable/task"
        android:text="@string/stStartMenuSettings" >
    </Button>

    <Button
        android:id="@+id/btStartMenuSync"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="0.5"
        android:textSize="16sp"
        android:drawableLeft="@drawable/sync"
        android:text="@string/stStartMenuSync" >
    </Button>
</LinearLayout>
```

```
     <LinearLayout
        android:id="@+id/l3"
        android:layout_width="match_parent"
        android:layout_height="75dp"
        android:layout_above="@id/l2"
        android:layout_marginTop="5dp"
        android:gravity="center"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btStartMenuNewFarm"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="0.5"
            android:textSize="17sp"
            android:drawableLeft="@drawable/new_farm"
            android:text="@string/stStartMenuNewFarm" >
        </Button>

        <Button
            android:id="@+id/btStartMenuNewJob"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:textSize="17sp"
            android:layout_weight="0.5"
            android:drawableLeft="@drawable/new_job"
            android:text="@string/stStartMenuNewJob" >
        </Button>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="75dp"
        android:layout_above="@id/l3"
        android:layout_marginTop="5dp"
        android:gravity="center"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btStartMenuMap"
            android:layout_width="200dp"
            android:textSize="17sp"
            android:layout_height="match_parent"
            android:drawableLeft="@drawable/fence"
            android:text="@string/stStartMenuMap" >
        </Button>
    </LinearLayout>

</RelativeLayout>
```



Εικόνα 3.1: Παράδειγμα κατασκευής ενός RelativeLayout και πολλών επιμέρους LinearLayout τα οποία περιλαμβάνουν διάφορα κουμπιά

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:background="#000044">
<TableRow>
        <TextView
                android:text="User Name:"
                android:width ="120px"
        />
        <EditText
                android:id="@+id/txtUserName"
                android:width="200px" />
</TableRow>
<TableRow>
        <TextView
                android:text="Password:"
        />
```

```
        <EditText
                android:id="@+id/txtPassword"
                android:password="true" 27
                />
</TableRow>
<TableRow>
        <CheckBox
                 android:id="@+id/chkRememberPassword"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="Remember Password"
                />
</TableRow>
<TableRow>
        <Button
                android:id="@+id/buttonSignIn"
                android:text="Log In"
         />
</TableRow>
</TableLayout>
```

Εικόνα 3.2: Παράδειγμα κατασκευής ενός TableLayout το οποίο ζητάει όνομα χρήστη και κωδικό πρόσβασης

3.2.2 Μενού

Τα μενού αποτελούν ένα σημαντικό κομμάτι της διεπαφής χρήστη για κάθε οθόνη της εφαρμογής, διότι παρέχουν στον χρήστη ένα γνωστό και φιλικό τρόπο για να εισάγει τις επιλογές του. Στο λειτουργικό σύστημα Android, υπάρχουν 3 διαφορετικά είδη μενού: το μενού επιλογών (options menu), το μενού πλαισίου

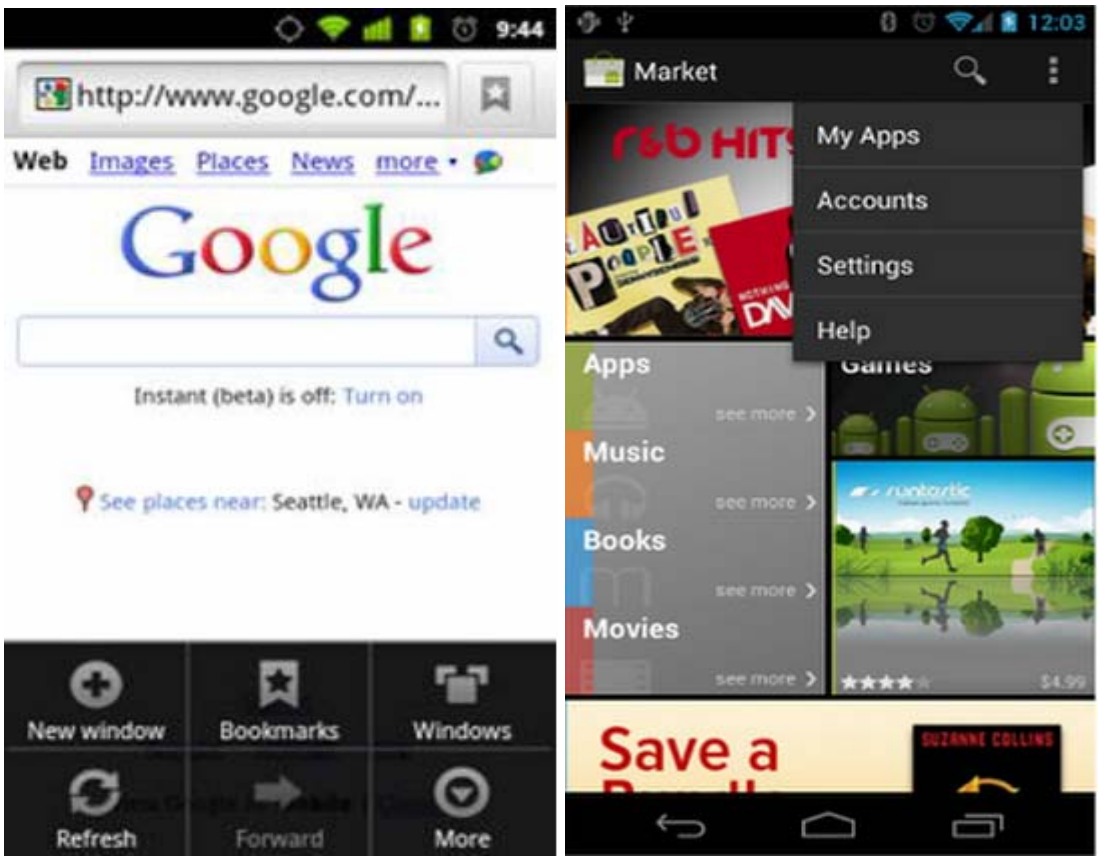
(context menu) και το υπομενού (submenu), τα οποία δηλώνονται και αυτά σε αρχεία xml.

3.2.2.1 Options menu

Το options menu αποτελεί το βασικότερο μενού μιας εφαρμογής. Εμφανίζεται τη στιγμή που πατάμε το κουμπί menu της συσκευής μας και περιέχει όλες τις βασικές επιλογές τις εφαρμογής μας. Αποτελεί κυρίως τον τρόπο με τον οποίο περιηγούμαστε μεταξύ των διαφορετικών οθονών και activities της εφαρμογής μας. Στον κώδικα της εφαρμογής μας ορίζουμε κάθε επιλογή του μενού σε ποια activity θα οδηγήσει τον χρήστη.

Όμως από την έκδοση android 3.0+ η google προτείνει να μην χρησιμοποιούμε το options menu αλλά να το αντικαταστήσουμε με action bar. Το action bar είναι στην ουσία το ίδιο με το options menu απλά οι επιλογές του options menu είναι ορατές και προσβάσιμες πλέον στο πάνω μέρος δίπλα στον τίτλο της εφαρμογής μας. Αυτό έγινε διότι στις καινούργιες εκδόσεις android δεν υπάρχει κουμπί menu ούτε και κανονικών κουμπιών στις περισσότερες συσκευές, αλλά όλα αυτά υποστηρίζονται με software keys.

Βέβαια για λόγους συμβατότητας, εφαρμογές που είχαν γραφτεί για να υποστηρίζουν options menu θα συνεχίσουν να δουλεύουν κανονικά. Οι επιλογές του options menu σε αυτές τις εφαρμογές θα είναι προσβάσιμες από τον χρήστη πατώντας στις 3 τελείες που θα εμφανίζονται στην εφαρμογή στα activities που είχαν Options menu [Εικόνα 3.4]



Αριστερά εικόνα 3.3: Χρήση του options menu σε εφαρμογές μέχρι Android 2.3
Δεξιά εικόνα 3.4: Χρήση action bar στις εφαρμογές. Επίσης διακρίνουμε τις επιλογές του options menu σε νέες εκδόσεις Android.

3.2.2.2 Context Menu

Γνωρίζουμε όλοι τις επιλογές που μας δίνει το δεξί κλικ σε διάφορα προγράμματα ή εικονίδια στα λειτουργικά συστήματα που χρησιμοποιούνται στους υπολογιστές. Το ζήτημα είναι πως θα εμφανίσουμε παρόμοιες επιλογές στις εφαρμογές Android, αφού διαθέτουμε οθόνη αφής. Τη λύση έρχεται να μας δώσει το context menu το οποίο περιλαμβάνει αυτές ακριβώς τις επιλογές σε οποιοδήποτε γραφικό στοιχείο το ορίσουμε (εικόνα, κείμενο κλπ) και ενεργοποιείται από τον χρήστη με παρατεταμένο πάτημα του στοιχείου αυτού. Για παράδειγμα στο context menu ενός κειμένου θα ορίζαμε επιλογές αντιγραφή, αποκοπή κλπ, στο context menu μιας διεύθυνσης Url σε εφαρμογή web browser θα ορίζαμε επιλογές άνοιγμα, άνοιγμα σε νέα καρτέλα κλπ.

3.2.2.3 Submenu

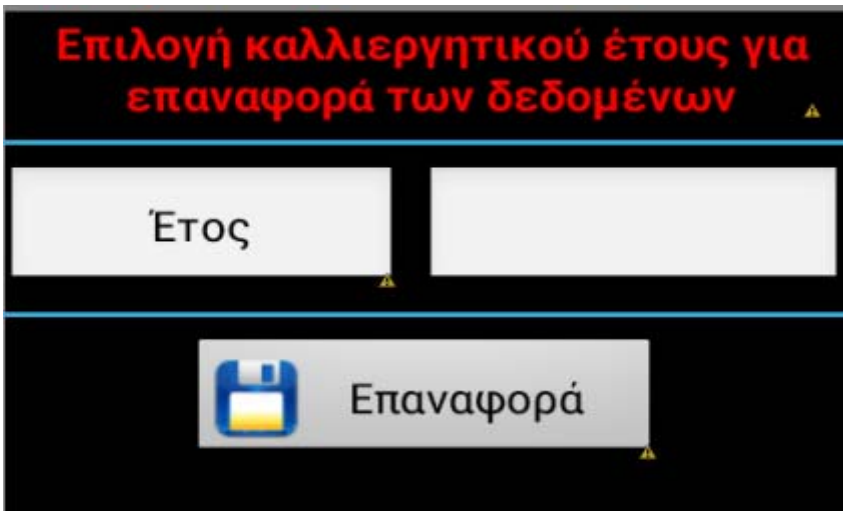
Το submenu μπορεί να προστεθεί σαν επιλογή στα 2 παραπάνω menu και όπως δείχνει και το όνομά του ανοίγει ένα επιπλέον μενού για περισσότερες επιλογές. Χρησιμοποιείται σε περιπτώσεις που η εφαρμογή μας εκτελεί αρκετές λειτουργίες και θέλουμε να τις οργανώσουμε σε μενού. Κάτι αντίστοιχο δηλαδή με τις επιλογές Αρχείο, Επεξεργασία, Προβολή κτλ που διαθέτουν τα περισσότερα προγράμματα.

3.2.3 Dialogs

Ο διάλογος (dialog) είναι συνήθως ένα μικρό παράθυρο που εμφανίζεται στην οθόνη μπροστά από την activity που τον κάλεσε. Η activity αυτή χάνει την εστίαση που είχε (focus) και το παράθυρο του διαλόγου είναι το μοναδικό με το οποίο μπορεί να αλληλεπιδράσει ο χρήστης. Χρησιμοποιείται είτε για ενημέρωση του χρήστη για κάποιο γεγονός είτε για να ορίσει ο χρήστης κάποια επιλογή του. Τα κυριότερα είδη διαλόγων είναι ο AlertDialog (διάλογος ειδοποίησης) και ο ProgressDialog (διάλογος προόδου).

Ο AlertDialog είναι ο πιο συνηθισμένος διάλογος. Αποτελείται από ένα τίτλο, ένα μήνυμα, ορισμένα κουμπιά ή μια λίστα από επιλογές. Για κάθε κουμπί του διαλόγου, ορίζουμε μέσα στην activity τις ενέργειες που θα ακολουθήσουν όταν το πατήσει ο χρήστης.

```
AlertDialog restoreAlert;
restoreAlert = new AlertDialog.Builder(StartMenu.this).create();
                restoreAlert.requestWindowFeature(Window.FEATURE_NO_TITLE);
        Rect displayRectangle = new Rect();
        Window window = this.getWindow();
        window.getDecorView().getWindowVisibleDisplayFrame(displayRectangle);
        // inflate and adjust layout
        LayoutInflater inflater =
(LayoutInflater)this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View layout = inflater.inflate(R.layout.restore_dialog, null);
        layout.setMinimumWidth((int)(displayRectangle.width() * 0.8f));
        //layout.setMinimumHeight((int)(displayRectangle.height() * 0.33f));
        restoreAlert.setCancelable(true);
        restoreAlert.setView(layout);
        restoreAlert.show();
```



Εικόνα 3.5: Παράδειγμα AlertDialog

Ο ProgressDialog αποτελεί ουσιαστικά επέκταση του AlertDialog και χρησιμοποιείται όταν θέλουμε να εμφανίσουμε στο χρήστη την πρόοδο για κάποια ενέργεια. Για παράδειγμα όταν θέλουμε να κατεβάσουμε κάποιες εικόνες από το διαδίκτυο και να τις εμφανίσουμε στο χρήστη, θα χρειαστούμε κάποιο χρονικό διάστημα για να ολοκληρωθεί αυτή η ενέργεια. Οπότε για να μη βλέπει ο χρήστης μια κενή μαύρη οθόνη, εμφανίζουμε έναν ProgressDialog και στο background εκτελούμε τις χρονοβόρες διαδικασίες.

```
ProgressDialog dialog = ProgressDialog.show (MyActivity.this, "", "Loading. Please wait...", true);
```

Εικόνα 3.6: Παράδειγμα ProgressDialog

Υπάρχουν 2 τρόποι για να ακυρώσουμε έναν διάλογο. Τις περισσότερες φορές θέλουμε να δώσουμε στο χρήστη την ελευθερία να ακυρώσει έναν διάλογο με τον πιο φυσικό τρόπο του Android, δηλαδή το back button. Το γεγονός αυτό το επιτυγχάνουμε ορίζοντας τον διάλογο μας cancelable (ακυρώσιμο). Ακόμη, στις ενέργειες που ακολουθούν όταν ο χρήστης πατήσει κάποιο κουμπί, τις περισσότερες φορές πρέπει να συμπεριλάβουμε και την εντολή dismiss() ώστε ο διάλογος μας να εξαφανιστεί και έπειτα να συνεχίσει η ροή της εφαρμογής ανάλογα με την επιλογή του χρήστη.

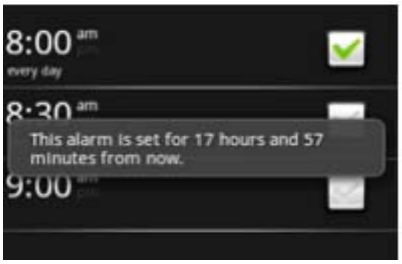
3.2.4 Ειδοποιώντας τον χρήστη

Σε πολλές περιπτώσεις θέλουμε να ενημερώσουμε το χρήστη για κάποιο γεγονός ή αποτέλεσμα σχετικό με την εφαρμογή μας. Ορισμένα από αυτά τα γεγονότα απαιτούν κάποια απάντηση από το χρήστη και άλλα όχι. Για παράδειγμα όταν ο χρήστης αποθηκεύει ένα αρχείο, θα θέλαμε να δει κάποιο μήνυμα ότι το αρχείο αποθηκεύτηκε επιτυχώς. Ή όταν η εφαρμογή μας τρέχει στο background και θέλει να ενημερώσει το χρήστη για κάποιο γεγονός, θα πρέπει να στείλει κάποια ειδοποίηση την οποία ο χρήστης να μπορεί να «ανοίξει» όταν αυτός επιθυμεί. Στην πρώτη περίπτωση χρησιμοποιούμε toast notification ενώ στη δεύτερη status bar notification.

3.2.4.1 Toast Notification

Η toast notification είναι ένα μήνυμα που εμφανίζεται για λίγα δευτερόλεπτα στο παράθυρο που βρίσκεται ο χρήστης, οποιασδήποτε εφαρμογής και αν είναι αυτό. Ο χώρος που καταλαμβάνει είναι ο ελάχιστος απαιτούμενος ώστε το μήνυμα να είναι εμφανές, ενώ ο χρήστης μπορεί όσο εμφανίζεται το μήνυμα να αλληλεπιδρά με την activity στην οποία βρίσκεται. Δεν υπάρχει κάποια επιλογή σε αυτήν την ειδοποίηση, παρά μόνο ενημέρωση, δηλαδή ο χρήστης δεν μπορεί να αλληλεπιδράσει με την ειδοποίηση. Χρησιμοποιείται συνήθως για μικρά μηνύματα που δεν απαιτούν κάποια ενέργεια από το χρήστη όπως για παράδειγμα «Το αρχείο αποθηκεύτηκε επιτυχώς», «Το ξυπνητήρι ορίστηκε στις …» κλπ. Παρακάτω βλέπουμε ένα τέτοιο παράδειγμα.

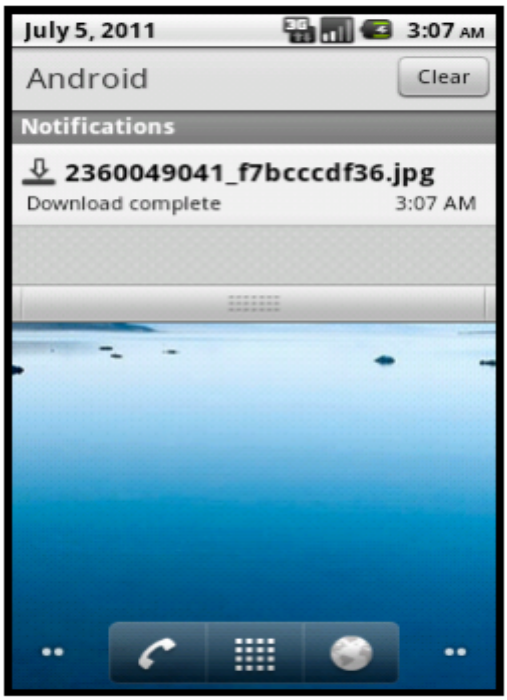
```
Context c = getApplicationContext ();
CharSequence text = "This alarm is set for " + hours + " hours and " + minutes + "
minutes from  now.";
Toast t = Toast.makeText(c, text, Toast.LENGTH_SHORT);
toast.show();
```



Εικόνα 3.7: Παράδειγμα Toast notification

3.2.4.2 Status Bar Notification

Η status bar notification, όπως φανερώνει και το όνομά της, είναι μια ειδοποίηση η οποία εμφανίζεται στην status bar του κινητού τηλεφώνου μας, και την οποία μπορούμε να ανοίξουμε είτε βρισκόμαστε στο κεντρικό μενού του τηλεφώνου μας, είτε σε κάποια εφαρμογή. Αντίθετα με την toast, η status bar notification μπορεί να επιλεχθεί και να ξεκινήσει κάποια λειτουργία ανάλογα με τις ενέργειες που έχουμε ορίσει στον κώδικα της εφαρμογής. Για παράδειγμα, όταν κατεβάζουμε ένα αρχείο από το διαδίκτυο, όταν η λήψη ολοκληρωθεί. Θα θέλαμε να επιλέξουμε την ειδοποίηση αυτή και με τον τρόπο αυτό είτε να ανοίξουμε τον φάκελο που βρίσκεται το αρχείο είτε να το τρέξουμε [Εικόνα 3.8]. Τις περισσότερες φορές οι toast notifications ενεργοποιούνται από activities, ενώ οι status bar notifications από services.



Εικόνα 3.8: Παράδειγμα status bar notification

3.3 Android Manifest

Σε κάθε εφαρμογή πρέπει να υπάρχει το αρχείο AndroidManifest.xml, το οποίο δημιουργείται αυτόματα όταν ξεκινάμε καινούργιο project μιας android εφαρμογής. Το αρχείο αυτό περιέχει βασικές πληροφορίες σχετικά με την εφαρμογή, τις οποίες το λειτουργικό σύστημα πρέπει να γνωρίζει προτού τρέξει οποιοδήποτε

άλλο κώδικα. Οι σημαντικότερες από αυτές τις πληροφορίες περιγράφονται παρακάτω:

- Η ονομασία του πακέτου της Java εφαρμογής (Java Package).
- Η έκδοση της εφαρμογής (πχ 1.0, 1.2.3 κλπ).
- Το όνομα της εφαρμογής καθώς και το εικονίδιό της.
- Η ελάχιστη έκδοση του λειτουργικού συστήματος Android που απαιτεί η εφαρμογή (min sdk version).
- Οι άδειες που απαιτούνται για να εκτελεστούν ορισμένες λειτουργίες της εφαρμογής. Για παράδειγμα αν η εφαρμογή μας χρησιμοποιεί τη sdcard της συσκευής μας θα πρέπει να δηλώσουμε και την αντίστοιχη άδεια. Αν θέλουμε να έχουμε πρόσβαση στο διαδίκτυο από την εφαρμογή μας, θα πρέπει να δηλώνεται η αντίστοιχη άδεια κτλ. Αν δεν δηλώσουμε τις άδειες που απαιτούνται για τις διάφορες λειτουργίες της εφαρμογής, τότε θα εμφανίζεται σφάλμα και δεν θα λειτουργεί. Οι άδειες αυτές περιγράφονται στον χρήστη τη στιγμή που εγκαθιστά την εφαρμογή και θα πρέπει να συμφωνήσει με αυτές για να ολοκληρωθεί η εγκατάσταση. Με τον τρόπο αυτό, ο χρήστης αισθάνεται ασφαλής απέναντι σε κακόβουλες εφαρμογές. Για παράδειγμα αν ο χρήστης επιχειρήσει να εγκαταστήσει μια εφαρμογή άσχετη με τηλεφωνικές κλήσεις ή μηνύματα και δει πριν την εγκατάσταση ότι αυτή ζητάει άδεια για τηλεφωνικές κλήσεις ή αποστολή μηνυμάτων, τότε θα καταλάβει ότι η εφαρμογή αυτή είναι πιθανότατα κακόβουλη και δεν πρέπει να εγκατασταθεί. Αν βέβαια οι συγκεκριμένες αυτές άδειες δεν αναγράφονται, τότε ο χρήστης είναι σίγουρος ότι η εφαρμογή δε θα μπορέσει με κανένα τρόπο να στείλει κάποιο γραπτό μήνυμα ή να πραγματοποιήσει κάποια τηλεφωνική κλήση. Ακόμα και αν επιχειρούσε να το κάνει, η εφαρμογή θα εμφάνιζε σφάλμα τη στιγμή που επιχειρούσαμε να την τρέξουμε και δεν θα λειτουργούσε.
- Όλα τα συστατικά στοιχεία (activities, services, content providers, broadcast receivers) της εφαρμογής. Για παράδειγμα αν δημιουργήσουμε κάποια κλάση για activity της εφαρμογής, χωρίς να την δηλώσουμε στο AndroidManifest.xml, δε θα μπορέσει να λειτουργήσει.
- Οι εξωτερικές βιβλιοθήκες που χρησιμοποιεί η εφαρμογή μας. Για παράδειγμα αν η εφαρμογή μας, σε κάποιο σημείο του κώδικα τρέχει την εφαρμογή google maps, τότε θα πρέπει να δηλωθεί η βιβλιοθήκη com.google.android.maps

Ακολουθεί για παράδειγμα ένα κομμάτι από το AndroidManifest.xml της εφαρμογής farm manager.

```
<? xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   Package ="com.georkouk.farmManager"
   android:versionCode="1"
   android:versionName="1.0" >
   <supports-screens
      android:anyDensity="true"
      android:largeScreens="true"
      android:normalScreens="true"
      android:smallScreens="true"
      android:resizeable="true" />
   <uses-sdk android:minSdkVersion="8" />
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
   <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
   <uses-permission
android:name="android.permission.ACCESS_MOCK_LOCATION" />
   <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
   <application
      android:icon="@drawable/agro"
      android:label="@string/app_name" >
      <uses-library android:name="com.google.android.maps" />
      <activity
         android:name=".FarmManagerSplash"
         android:label="@string/app_name"
         android:screenOrientation="portrait"
         android:windowSoftInputMode="stateHidden" >
```

```
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".StartMenu"
        android:label="@string/app_name"
        android:screenOrientation="portrait"
        android:theme="@style/CustomTheme"
        android:windowSoftInputMode="stateHidden" >
        <intent-filter>
            <action android:name="com.georkouk.farmManager.STARTMENU" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
</application>
</manifest>
```

ΚΕΦΑΛΑΙΟ 4

Εγκατάσταση και παραμετροποίηση απαιτούμενων εργαλείων

4.1 Εργαλεία ανάπτυξης εφαρμογών Android

Όπως είδαμε και πιο πριν για να αναπτύξουμε εφαρμογές σε Android χρησιμοποιούμε το Eclipse, το οποίο είναι ένα δωρεάν εργαλείο προγραμματισμού με πολλές δυνατότητες. Βέβαια μπορούμε να χρησιμοποιήσουμε όποιο εργαλείο προγραμματισμού για Java θέλουμε, αλλά εμείς θα χρησιμοποιήσουμε το Eclipse, αφού αυτό είναι που προτείνει και η Google για την ανάπτυξη εφαρμογών και παρέχει και μια σειρά από έτοιμες βιβλιοθήκες για χρήση μαζί με το Eclipse που κάνουν εύκολη τη ζωή των προγραμματιστών, το λεγόμενο ADT (Android Developer

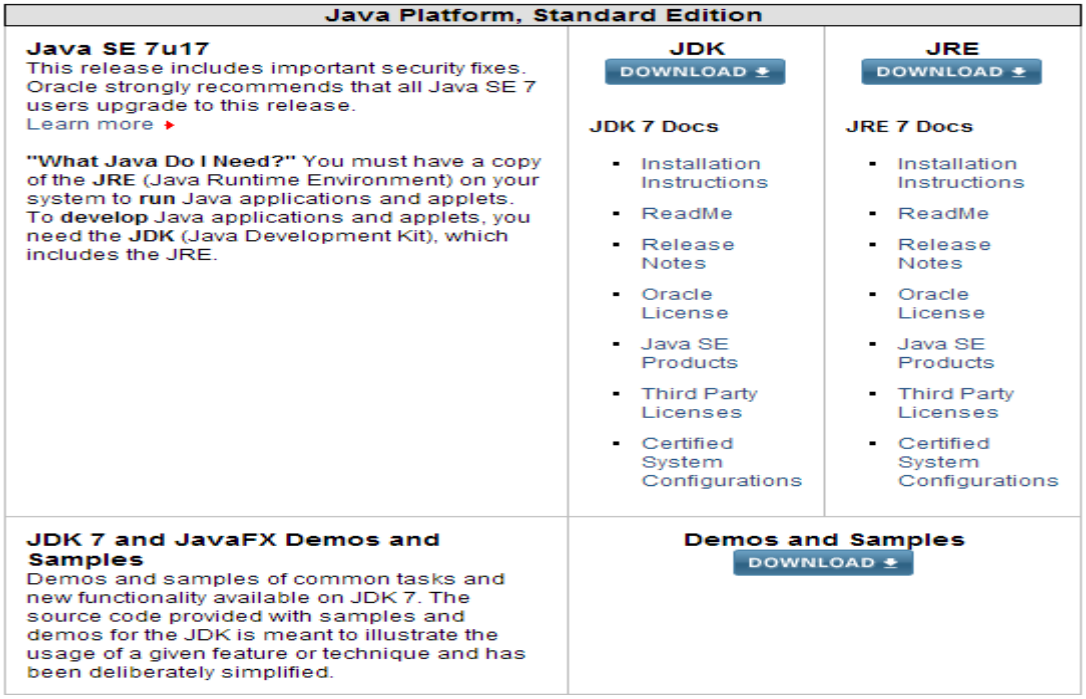
Tools). Παρακάτω θα δούμε όλα τα βήματα από την εγκατάσταση όλων των απαραίτητων εργαλείων μέχρι την δημιουργία και εκτέλεση ενός νέου project σε Android. Η εγκατάσταση έγινε σε υπολογιστή με λειτουργικό σύστημα Windows 7, αν και δεν θα δείτε διαφορές αν χρησιμοποιείτε κάποια άλλη έκδοση των windows ή Linux.

4.1.1 Εγκατάσταση Java

Για να μπορέσει να λειτουργήσει το Eclipse χρειαζόμαστε να έχουμε εγκατεστημένο στον υπολογιστή μας το JDK(Java Developer Kit) της Java, το οποίο περιλαμβάνει όλες της απαραίτητες βιβλιοθήκες και εργαλεία της Java που θα χρειαστούμε. Για να γίνει αυτό πηγαίνουμε στο επίσημο site της Oracle από τον παρακάτω σύνδεσμο

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Και κατεβάζουμε την τελευταία έκδοση του JDK. Την ώρα που γράφονταν το κείμενο η τελευταία έκδοση ήταν η 7. Αφού το κατεβάσουμε το εκτελούμε και περιμένουμε να τελειώσει η εγκατάσταση.



Εικόνα 4.1: Λήψη του Java JDK από το επίσημο site

4.1.2 Εγκατάσταση Eclipse

Στη συνέχεια, αφού τελειώσουμε με την εγκατάσταση του JDK, θα χρειαστεί να κατεβάσουμε το eclipse από τον παρακάτω σύνδεσμο

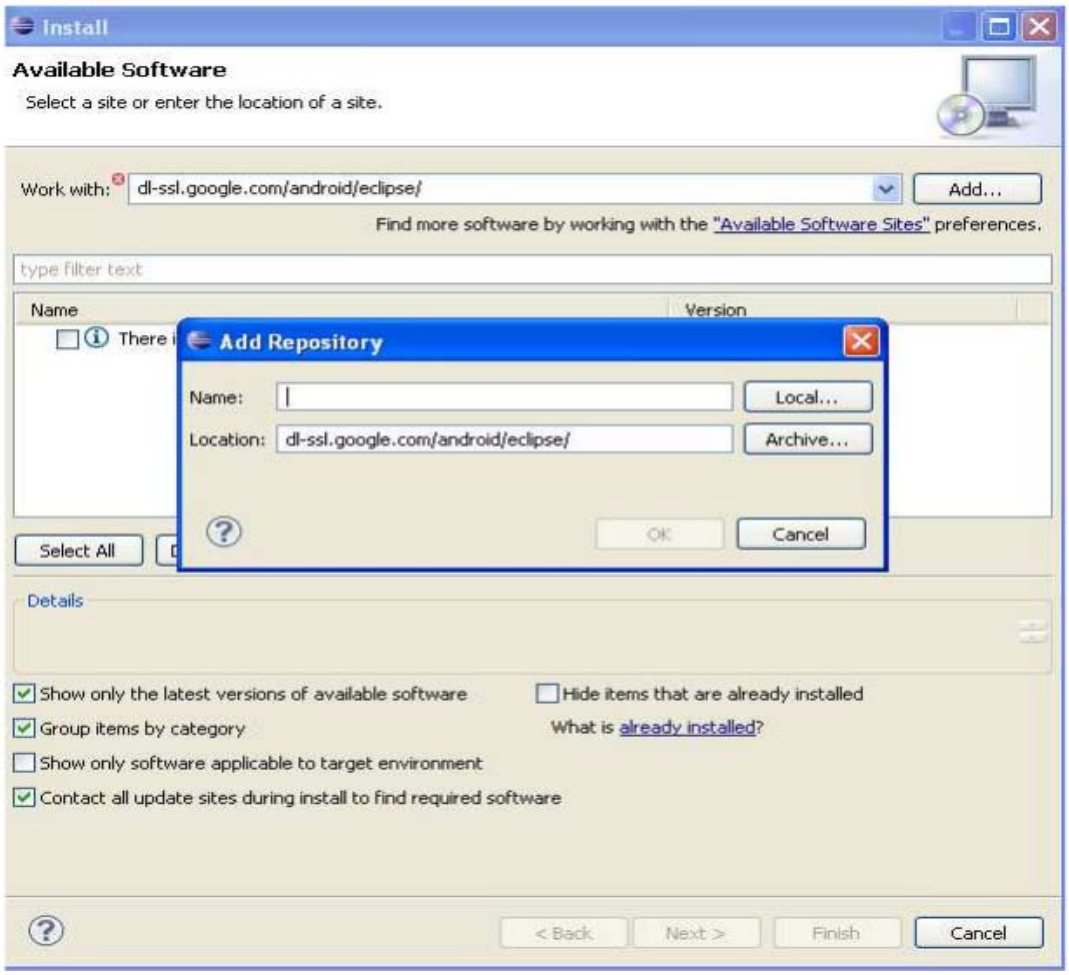
http://www.eclipse.org/downloads/

Μπορούμε να κατεβάσουμε όποια έκδοση είναι συμβατή με Java. Η Google προτείνει την χρήση του Eclipse Classic οπότε και θα κατεβάσουμε αυτή την έκδοση. Αφού τελειώσει η λήψη κάνουμε αποσυμπίεση τα δεδομένα σε έναν φάκελο της επιλογής μας, έστω C:/eclipse .

Εικόνα 4.2: Διάφορες εκδόσεις του Eclipse

4.1.3 Εγκατάσταση του Android ADT

Εφόσον έχουμε τελειώσει και με την εγκατάσταση του Eclipse, θα χρειαστεί να εγκαταστήσουμε το plugin που δίνει η Google το οποίο περιέχει έτοιμες βιβλιοθήκες του Android. Για να γίνει αυτό ανοίγουμε το Eclipse. Αρχικά θα μας ζητήσει να ορίσουμε τον φάκελο όπου θα αποθηκεύονται τα project μας. Αν θέλουμε κάποιο συγκεκριμένο φάκελο τον επιλέγουμε αλλιώς αφήνουμε τον προεπιλεγμένο. Στη συνέχεια πηγαίνουμε στο Help→Install New Software. Στο παράθυρο που θα ανοίξει στο πεδίο “Work With” γράφουμε https://dl-ssl.google.com/android/eclipse/ και πατάμε add. Στο νέο παράθυρο που θα ανοίξει γράφουμε ένα όνομα ώστε αργότερα να μπορούμε να τα διαχωρίζουμε και πατάμε OK.



Εικόνα 4.3: Εγκατάσταση του ADT (1)

Περιμένουμε λίγο μέχρι να μας εμφανίσει τα Developer Tools. Μόλις τα εμφανίσει τα επιλέγουμε όλα και πατάμε Next. Εκεί αποδεχόμαστε τις άδειες χρήσης και περιμένουμε να τελειώσει η εγκατάσταση. Οτιδήποτε μήνυμα μας εμφανίσει κατά τη διάρκεια της εγκατάστασης πατάμε OK για να συνεχίσει. Μόλις ολοκληρωθεί η εγκατάσταση κλείνουμε το eclipse.

4.1.4 Εγκατάσταση του Android SDK

Τελειώνοντας με την εγκατάσταση των απαιτούμενων εργαλείων θα χρειαστεί να εγκαταστήσουμε το Android SDK. Πρόκειται για το εργαλείο με το οποίο μπορούμε να κατεβάσουμε όλες τις απαραίτητες βιβλιοθήκες και εργαλεία για κάθε έκδοση του Android ξεχωριστά. Σε αυτό περιέχονται και όλα τα απαραίτητα για τη δημιουργία εικονικών μηχανών (virtual machines). Για να γίνει αυτό πηγαίνουμε στο επίσημο site της Google για το Android στον παρακάτω σύνδεσμο http://developer.android.com/sdk/index.html

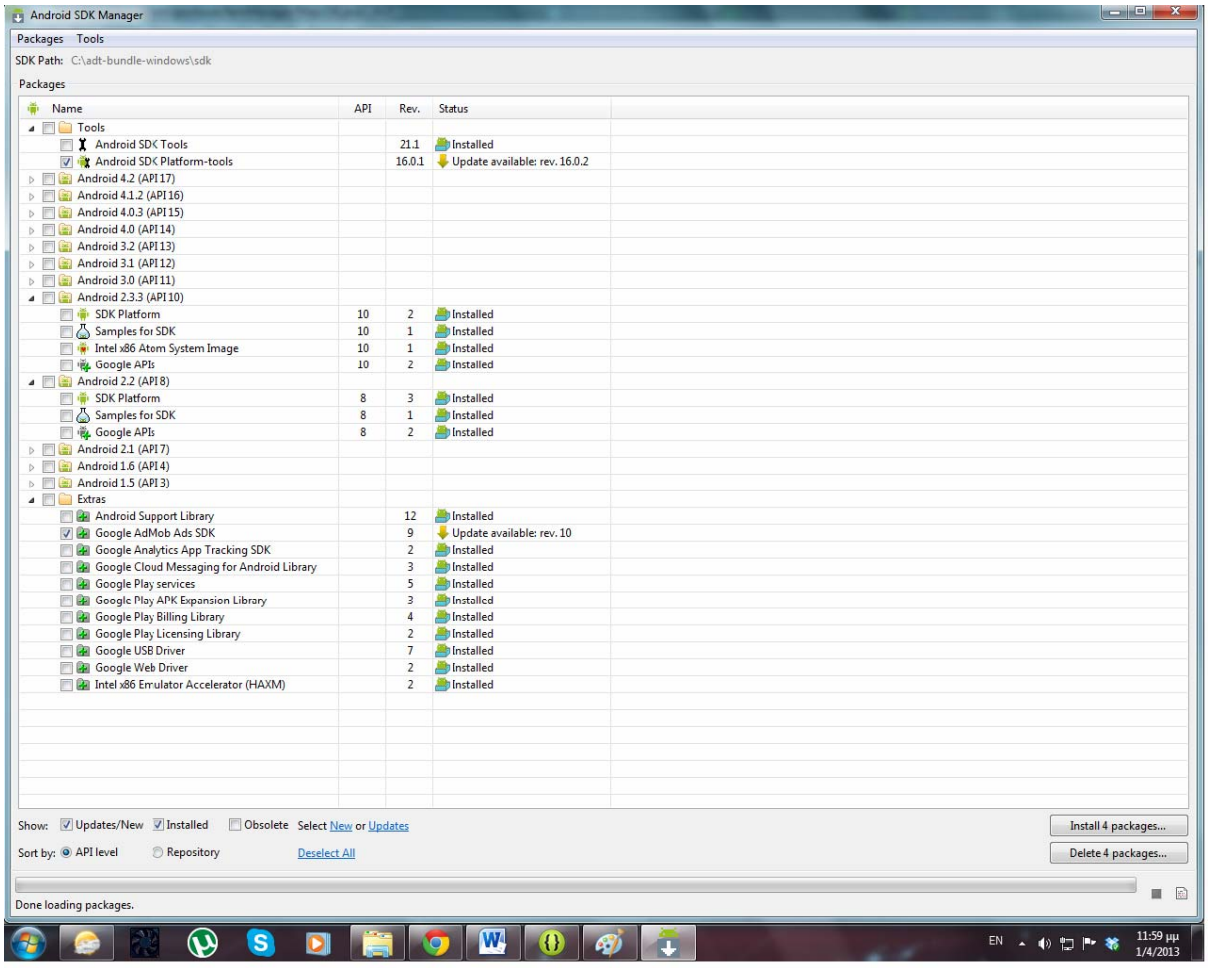
και κατεβάζουμε το αρχείο installer_r21.1-windows.exe το οποίο είναι για windows. Αν χρησιμοποιούμε κάποιο άλλο λειτουργικό θα πρέπει να κατεβάσουμε το αντίστοιχο που υπάρχει στη λίστα.

SDK Tools Only

Platform	Package	Size	MD5 Checksum
Windows 32 & 64-bit	android-sdk_r21.1-windows.zip	99360755 bytes	dbece8859da9b66a1e8e7cd47b1e647e
	installer_r21.1-windows.exe (Recommended)	77767013 bytes	594d8ff8e349db9e783a5f2229561353
Mac OS X 32 & 64-bit	android-sdk_r21.1-macosx.zip	66077080 bytes	49903cf79e1f8e3fde54a95bd3666385
Linux 32 & 64-bit	android-sdk_r21.1-linux.tgz	91617112 bytes	3369a439240cf3dbe165d6b4173900a8

Εικόνα 4.4: Λήψη του Android SDK από το επίσημο site

Αφού κατέβει το εκτελούμε και περιμένουμε να τελειώσει η εγκατάσταση του. Μόλις εγκατασταθεί ανοίγουμε το eclipse και πηγαίνουμε στο window→Android Sdk Manager. Εκεί ανοίγει ένα παράθυρο παρόμοιο με την παρακάτω εικόνα.



Εικόνα 4.5: 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 farmManager έχει αναπτυχθεί για Android 2.2 και πάνω οπότε το ελάχιστο που πρέπει να έχουμε είναι η έκδοση 2.2. Ο χρόνος εγκατάστασης διαφέρει ανάλογα με το πόσα πακέτα έχουμε επιλέξει να εγκατασταθούν και τον φόρτο του server εκείνη την ώρα, οπότε να είστε λίγο υπομονετικοί κατά την εγκατάσταση. Αφού τελειώσει η εγκατάσταση κλείνουμε τον SDK Manager και κάνουμε επανεκκίνηση το Eclipse.

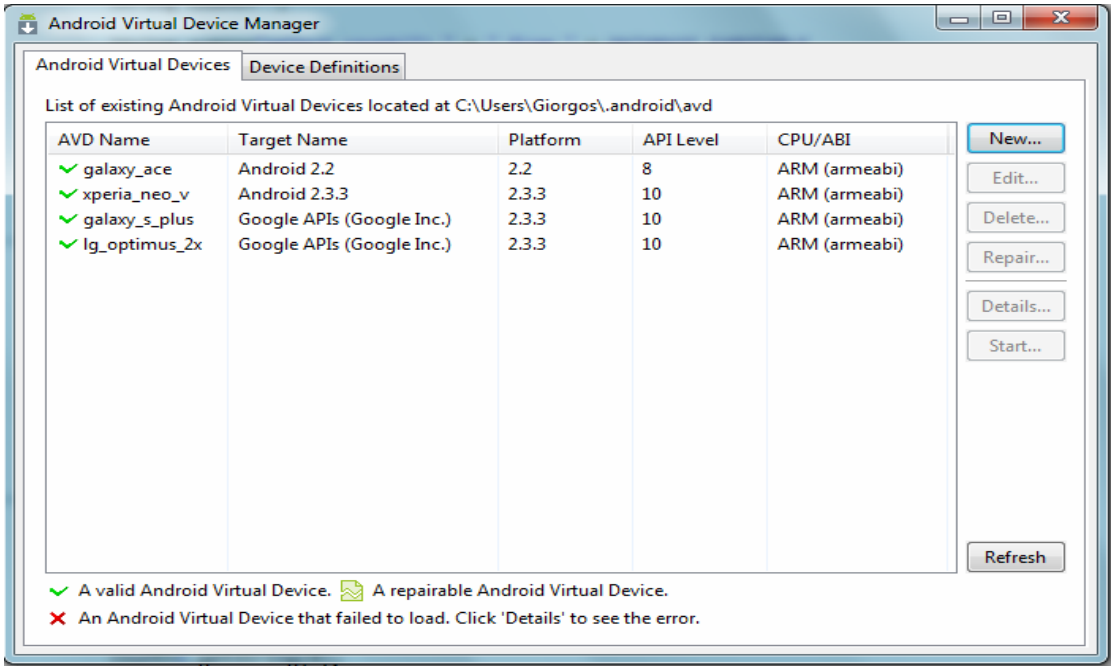
4.2 Ρύθμιση παραμέτρων και δημιουργία εικονικής μηχανής

Αφού τελειώσουμε με την εγκατάσταση όλων των παραπάνω εργαλείων πλέον είμαστε σχεδόν έτοιμοι για να ξεκινήσουμε την ανάπτυξη εφαρμογών Android. Παρακάτω θα δούμε πως φτιάχνουμε μια εικονική μηχανή για να μπορούμε να τρέχουμε την εφαρμογή μας και πως τη ρυθμίζουμε.

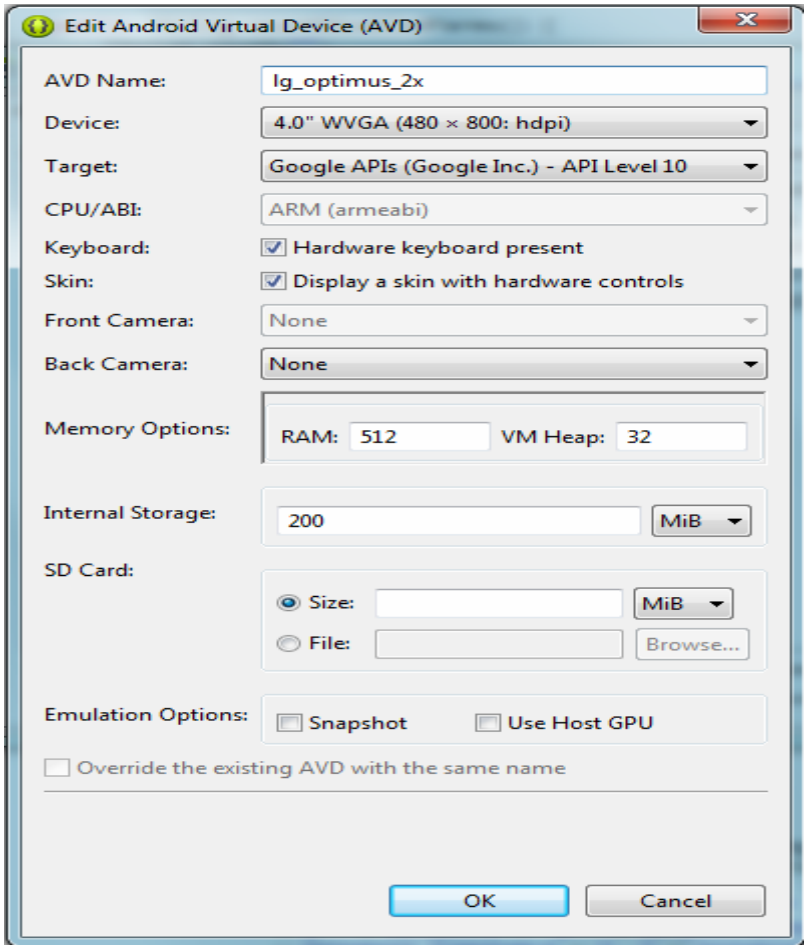
4.2.1 Δημιουργία Virtual Machine

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 [Εικόνα 4.6] εμφανίζονται όλες οι εικονικές συσκευές που έχουμε δημιουργήσει. Για να δημιουργήσουμε μια καινούργια επιλέγουμε New. Στο παράθυρο που ανοίγει [Εικόνα 4.7] 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.



Εικόνα 4.6: Android Virtual Device Manager



Εικόνα 4.7: Δημιουργία νέας εικονικής μηχανής

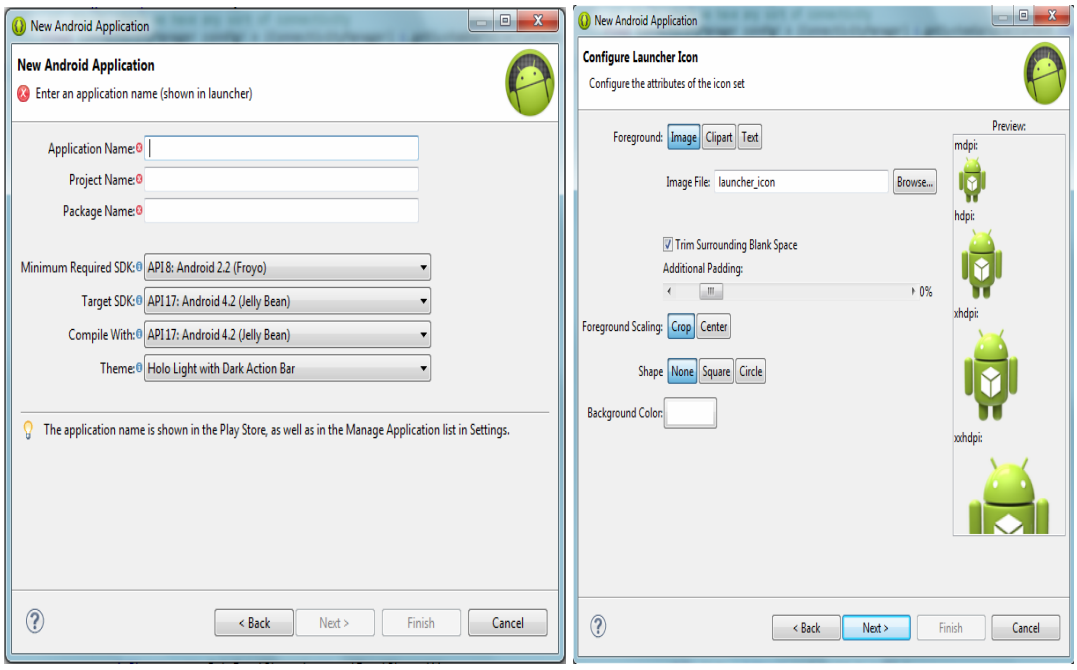
4.2.2 Εκτέλεση εικονικής μηχανής (Virtual Machine)

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 4.8: Παράδειγμα μιας εικονικής μηχανής Android 2.2 σε λειτουργία

4.3 Δημιουργία νέου Android Project στο Eclipse

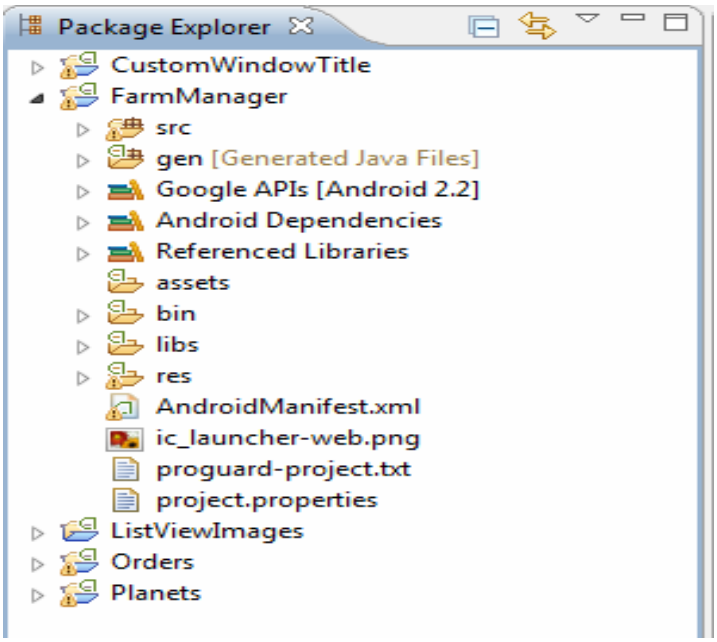
Εφόσον έχουμε κάνει όλα τα παραπάνω επιτυχώς είμαστε έτοιμοι για να ξεκινήσουμε την ανάπτυξη εφαρμογών. Ας δούμε λοιπόν πως φτιάχνουμε ένα νέο project για Android στο eclipse. Αρχικά πηγαίνουμε στο File→New→Other. Στο παραθυράκι που ανοίγει κάνουμε διπλό κλικ στο Android για να ανοίξει η λίστα και επιλέγουμε 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 4.9: Κάποια από τα βήματα δημιουργίας νέου project στο Eclipse

4.4 Δομή ενός Android project στο Eclipse

Εφόσον έχουμε δημιουργήσει ένα νέο project, όπως περιγράψαμε προηγουμένως, ας δούμε λίγο τη δομή των φακέλων του project και τι μπορούμε να βάλουμε στον κάθε φάκελο.

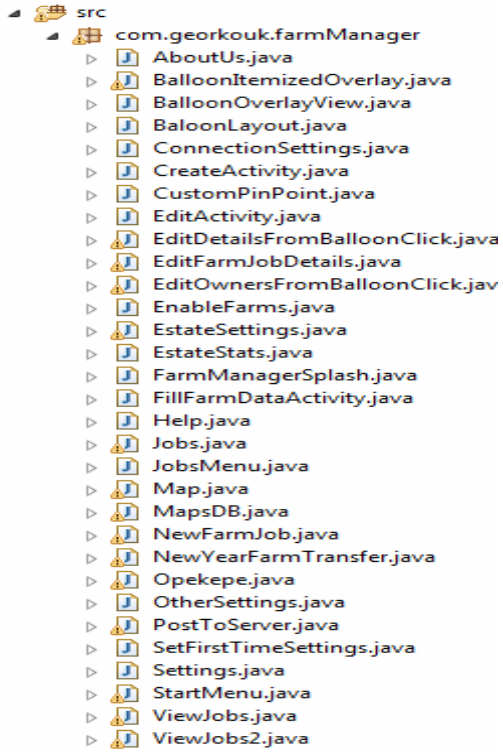


Εικόνα 4.10: Η δομή φακέλων του project FarmManager

Αυτή είναι η δομή του project της πτυχιακής αυτής εργασίας που ανέπτυξα. Όλα τα android project παρουσιάζουν την ίδια δομή φακέλων και αρχείων. Παρακάτω βλέπουμε την ανάλυση των φακέλων και των αρχείων τους.

- Src/

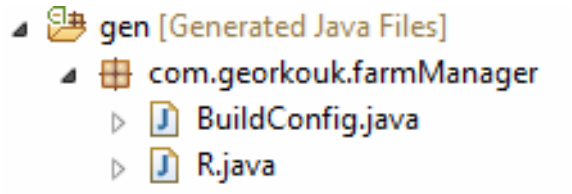
  Ο φάκελος αυτός αρχικά περιέχει τα πακέτα τα οποία έχουμε δηλώσει στην εφαρμογή μας. Στην περίπτωσή μας έχουμε δηλώσει ένα μόνο πακέτο το "com.georkouk.farmManager" οπότε και εμφανίζεται μόνο αυτό. Στη συνέχεια κάτω από το πακέτο εμφανίζονται όλες οι κλάσεις που έχουμε δηλώσει στο πακέτο μας



Εικόνα 4.11: Περιεχόμενα φακέλου src/

- Gen/

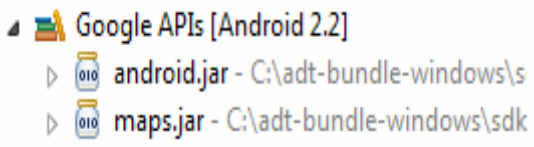
Ο φάκελος αυτός περιέχει για το πακέτο που χρησιμοποιούμε στην εφαρμογή μας, το αρχείο R.java το οποίο ουσιαστικά είναι μια κλάση η οποία κατασκευάζεται αυτόματα από το σύστημα και δεν πρέπει να την τροποποιούμε γιατί μπορεί να δημιουργηθεί πρόβλημα. Η κλάση αυτή συνδέει όλα τα application resources με τον κώδικα της εφαρμογής μας. Έχουμε εξηγήσει ότι όλα αυτά τα resources δηλώνονται σε ξεχωριστά αρχεία και όχι μέσα στις κλάσεις της εφαρμογής μας. Η κλάση λοιπόν αυτή δηλώνει κάθε στοιχείο από τις εικόνες, τα γραφικά στοιχεία, τα strings, τα styles ή themes που χρησιμοποιούμε στην εφαρμογή μας, σαν μια ακέραια σταθερά (int) και με τιμή έναν μοναδικό δεκαεξαδικό αριθμό, ώστε να αναφερόμαστε σε αυτές μέσα από τις δικές μας κλάσεις σαν δημόσιες σταθερές.



Εικόνα 4.12: Περιεχόμενα φακέλου gen/

- Google Apis

Πρόκειται για τις βιβλιοθήκες που επιλέγονται αυτόματα από το σύστημα όταν εμείς δηλώσουμε την ελάχιστη έκδοση Android με την οποία θα είναι συμβατή η εφαρμογή μας. Το σύστημα τοποθετεί τις κατάλληλες βιβλιοθήκες για να λειτουργεί η εφαρμογή μας στις εκδόσεις αυτές (αρκεί να έχουμε εγκαταστήσει τις εκδόσεις αυτές μέσω του ADT). Προφανώς το ιδανικό θα ήταν η εφαρμογή μας να ήταν συμβατή με όλες τις εκδόσεις όμως όπως έχουμε εξηγήσει κάτι τέτοιο είναι δύσκολο καθώς κάθε έκδοση έχει διαφορετικές δυνατότητες. Συγκεκριμένα η εφαρμογή μας FarmManager είναι συμβατή με εκδόσεις Android 2.2 +. Το αρχείο λοιπόν android.jar αποτελεί τη βασική βιβλιοθήκη του λειτουργικού συστήματος, ενώ το αρχείο maps.jar αποτελεί τη βιβλιοθήκη που χρειάζεται ώστε να λειτουργούν οι χάρτες της Google στην εφαρμογή μας. Τα παραπάνω αρχεία προστίθενται αυτόματα όταν κατά την κατασκευή νέου project επιλέγουμε Build Target για την εφαρμογή μας



Εικόνα 4.13: Βιβλιοθήκες του λειτουργικού συστήματος Android

- Referenced Libraries

Στον φάκελο αυτό ο προγραμματιστής μπορεί να προσθέτει βιβλιοθήκες που θέλει να χρησιμοποιήσει μέσα στην εφαρμογή.

- Assets/

Στον φάκελο αυτό μπορούμε να αποθηκεύσουμε οποιοδήποτε αρχείο θέλουμε ή και να δημιουργήσουμε υποφακέλους. Ωστόσο όλα τα application resources αποθηκεύονται σε διαφορετικό φάκελο, πράγμα που καθιστά τον φάκελο αυτό όχι και πολύ χρήσιμο.

- Libs/

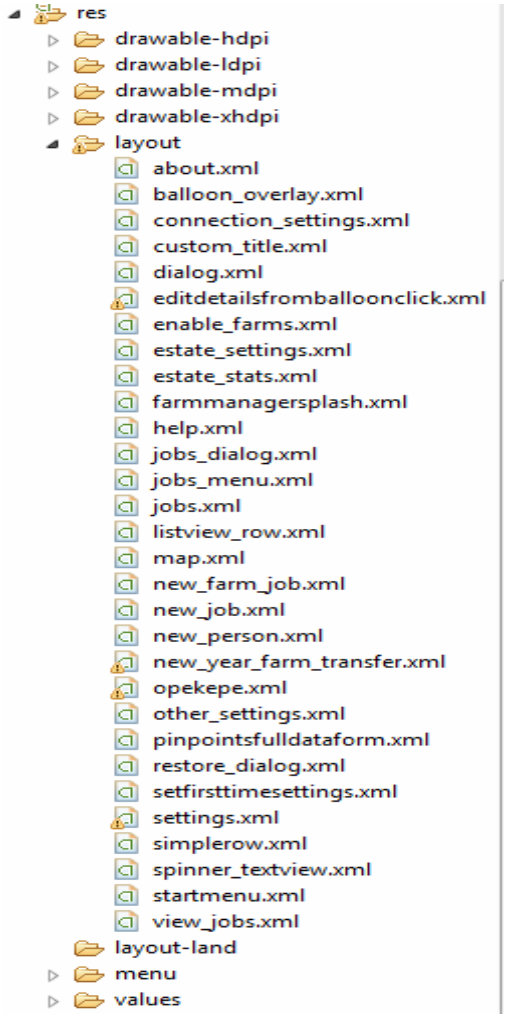
Ο φάκελος αυτός αποτελεί τη φυσική τοποθεσία των βιβλιοθηκών "Referenced Libraries" που εξηγήθηκαν παραπάνω. Ο προγραμματιστής τοποθετεί στον φάκελο αυτό τα αρχεία .jar που θέλει να χρησιμοποιήσει και

στη συνέχεια τα συνδέει με το λειτουργικό. Τα συνδεμένα αυτά αρχεία αποτελούν τις Referenced Libraries.

- Res/

Στον φάκελο αυτό αποθηκεύονται όλα τα application resources που χρησιμοποιούμε. Συγκεκριμένα στον φάκελο drawable αποθηκεύουμε όλα τα αρχεία εικόνας που χρησιμοποιούμε στην εφαρμογή. Στο φάκελο layout αποθηκεύονται τα αρχεία xml στα οποία δηλώνουμε το user interface της κάθε οθόνης. Ο φάκελος values περιλαμβάνει το αρχείο strings.xml με όλα τα strings της εφαρμογής και το αρχείο styles.xml με όλα τα styles τα οποία χρησιμοποιούμε.

Τέλος παρατηρούμε τρία ακόμα αρχεία τα οποία δημιουργούνται αυτόματα από το σύστημα. Το ένα από αυτά είναι το AndroidManifest.xml το οποίο έχουμε αναλύσει. Τα άλλα 2 αρχεία αφορούν το σύστημα και δεν πρέπει να τα τροποποιούμε.



Εικόνα 4.14: Όλα τα resources της εφαρμογής,

4.5 Προετοιμασία χαρτών Google για χρήση στην εφαρμογή μας

Οι Χάρτες Google για κινητά είναι μια εφαρμογή σε Java, η οποία εκδόθηκε από την Google το 2006, με σκοπό να τρέχει σε οποιοδήποτε βασισμένο σε Java κινητό ή φορητή συσκευή. Πολλά από τα χαρακτηριστικά των Χαρτών Google υπάρχουν και σε αυτή την εφαρμογή. Στις 28 Νοεμβρίου του 2008 κυκλοφόρησε και η δεύτερη έκδοση της εφαρμογής αυτής, οι Χάρτες Google για Κινητά 2.0. Σε αυτήν την καινούργια έκδοση, η Google υλοποίησε μια υπηρεσία παρόμοια με το παγκόσμιο σύστημα εντοπισμού θέσης (GPS-like), η οποία όμως δεν χρειαζόταν δέκτη GPS. Η λειτουργία 'η τοποθεσία μου' (my location), λειτουργεί με χρήση της τοποθεσίας GPS του κινητού, αν αυτή είναι διαθέσιμη.

Το Android προσφέρει στους προγραμματιστές την δυνατότητα να ενσωματώσουν κάποιο χάρτη στην εφαρμογή τους. Αυτό μπορεί να γίνει με δύο τρόπους. Σύμφωνα με τον πρώτο τρόπο, που είναι και ο πιο απλός, ο προγραμματιστής μπορεί να ενσωματώσει στην οθόνη ένα στοιχείο το οποίο καλείται 'WebView'. Στο 'WebView', ο προγραμματιστής μπορεί να απεικονίσει οποιαδήποτε διαδικτυακή εφαρμογή με αντίστοιχο τρόπο που αυτή απεικονίζεται σε έναν φυλλομετρητή και κατ' επέκταση τους Χάρτες Google. Η Google όμως θέλοντας να δώσει περισσότερη ελευθερία στους προγραμματιστές, δημιούργησε ένα άλλο στοιχείο διεπιφάνειας χρήστη, το 'MapView' γεγονός που μας οδηγεί στην δεύτερη εναλλακτική. Το MapView σε συνδυασμό με την κλάση MapActivity και τις διεπαφές mapping APIs, είναι ένα ισχυρό εργαλείο στα χέρια οποιουδήποτε προγραμματιστή που θέλει να δημιουργήσει καινοτόμες και πρωτοποριακές εφαρμογές με την χρήση των χαρτών. Με αυτό τον τρόπο ο κατασκευαστής μπορεί να χρησιμοποιήσει υπάρχων χάρτες και να τους τροποποιήσει όπως αυτός θέλει, με την δυνατότητα να χειριστεί τις αλληλεπιδράσεις του χρήστη με τον χάρτη, να τοποθετήσει δικά του δεδομένα πάνω σε αυτόν κ.τ.λ. Το πακέτο για την χρησιμοποίηση των mapping APIs δεν ανήκει στο πλαίσιο του Android, αλλά σε αυτό της Google (com.google.android.maps), γεγονός που πρέπει να δηλωθεί ξεχωριστά. Επιπρόσθετα για την χρησιμοποίηση των χαρτών η Google απαιτεί την απόκτηση από τον εκάστοτε προγραμματιστή ενός map-api key, κλειδί το οποίο προσφέρει η ίδια (http://code.google.com/android/maps-api-signup.html). Για την απόκτηση του κλειδιού ο ενδιαφερόμενος πρέπει να δώσει το αποτύπωμα MD5 του ψηφιακού πιστοποιητικού το οποίο θα χρησιμοποιηθεί για την υπογραφή της

εφαρμογής. Οδηγίες για το πώς θα πάρουμε το αποτύπωμα MD5 υπάρχουν στο παραπάνω link.

4.6 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 χρησιμοποίηση 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.

4.7 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.

4.8 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.

ΚΕΦΑΛΑΙΟ 5

Ανάπτυξη εφαρμογής FarmManager

5.1 Εισαγωγή

Εφόσον είδαμε τι είναι το Android, μεθοδολογία ανάπτυξης εφαρμογών και την προετοιμασία των εργαλείων προγραμματισμού που θα χρειαστούμε ήρθε η ώρα να αναπτύξουμε την εφαρμογή FarmManager. Ας δούμε όμως λίγα πράγματα για την εφαρμογή πριν προχωρήσουμε στο σχεδιασμό και την ανάπτυξή της.

Οι απαιτήσεις πληροφορίας έχουνε αυξηθεί σημαντικά για τους αγρότες. Οι καλλιεργητές δεν είχαν ποτέ άλλοτε την ανάγκη να αντιμετωπίσουν τον παγκόσμιο ανταγωνισμό, την μεγάλη εξειδίκευση λιπασμάτων, φαρμάκων, σπορικών και ασθενειών στις καλλιέργειες τους.

- Πρώτα από όλα το μέγεθος του αγροκτήματος άλλαξε οριστικά τα τελευταία χρόνια. Από 40-50 στρέμματα έχουμε αγροκτήματα 300 με 600 στρέμματα αποτελούμενα από πολλά τεμάχια 30 το ελάχιστο διαφόρων μεγεθών συνήθως μικρών τα οποία αυτά, άλλα είναι ιδιόκτητα, άλλα είναι ενοικιαζόμενα με επικαρπία αγρότη και άλλα δουλεύονται στο όνομα αυτού που έχει την επιδότηση!. Και όλα αυτά να δηλώνονται στον ΟΠΕΚΕΠΕ με κωδικούς για τους οποίους ο αγρότης έχει άγνοια κι αυτός να γνωρίζει μόνο ονομασίες χωραφιών (π.χ. καράτσικα, καράουρμάν, βάλτα, κ.τ.λ.) και ιδιοκτήτες. Έρχεται η εποχή των δηλώσεων και ο αγρότης με τον συνάδελφο της ένωσης ξοδεύουνε μέρες και μέρες για να βγάλουνε άκρη σε μια δήλωση όπου στο τέλος της ημέρας όλες οι δηλώσεις έχουνε λαθάκια τα οποία μπορούνε αποβούν και μοιραία.

Μπορεί να σκεφτεί κανείς λοιπόν, τι γίνεται όταν έχουμε αγρόκτημα με πολύ τεμαχισμό και όταν το μέγεθος του αγροκτήματος αυξάνεται σημαντικά σε 600, 1000 και πάνω στρέμματα όπου κάθε σημείωση σε χαρτί είναι ανεπαρκείς;. Πρακτικά το χάος και μόνον όσοι το ζούνε μπορούνε να το καταλάβουνε.

- Σήμερα οι εργασίες που λαμβάνουνε μέρος σε ένα αγρόκτημα είναι και αυτές αρκετές και σημαντικές. Η επιτυχία στην καλλιέργεια στηρίζεται στην σωστή προετοιμασία του εδάφους, σπορά στην κατάλληλη εποχή με σπορικά κατάλληλα για το κάθε χωράφι, ψεκασμούς στην κατάλληλη χρονική στιγμή και στο κατάλληλο στάδιο φυτών, αρδεύσεις και συγκομιδή στο κατάλληλο στάδιο. Πολλές είναι οι εργασίες που πρέπει να γίνουν. Όπως καταλαβαίνει ο αναγνώστης δεν επαρκεί μόνο να είσαι καλός γεωργός που ξέρεις να χειρίζεσαι άριστα τα μηχανήματα. Πρέπει να μπορείς να συνδυάσεις ένα σύνολο γνώσεων και να μπορείς να επεξεργαστείς εμπειρία γνώσης παλαιότερων εφαρμογών και αποτελέσματα αυτών τα οποία θα σε φέρουν σε βελτίωση των αποτελεσμάτων σου στην καλλιέργεια. Για παράδειγμα πώς θα απαντήσεις εάν σπορικό πήγε ή δεν πήγε καλά όταν εσύ δεν θυμάσαι πότε το έσπειρες και τι σπορικό έβαλες;. Οι περισσότεροι θυμούνται χονδρικά ‘έσπειρα 25 Μαρτίου με 5 Απριλίου’ και έβαλα μόνο δύο ποικιλίες σπόρων. Και κάθε χρόνο πάμε για νέες δοκιμές ανάλογα με το τι θα μας πουλήσουνε!.

Ο FarmManager λοιπόν είναι μια εφαρμογή διαχείρισης αγροκτήματος από συσκευές κινητής τηλεφωνίας (SmartPhones) με λειτουργικό σύστημα Android. Όλα τα στοιχεία (δεδομένα, εργασίες, καλλιέργειες) μπορούν να διαχειρίζονται εύκολα από το κινητό τηλέφωνο. Υποστηρίζει ηλεκτρονική χαρτογραφική διαχείριση κτημάτων με την βοήθεια των Google Maps. Ο χρήστης μπορεί:

- Να εισάγει ένα νέο χωράφι και να βλέπει τα ήδη καταχωρημένα χωράφια του αγροκτήματος
- Να εισάγει και να επεξεργάζεται τις εργασίες για κάθε χωράφι
- Να εισάγει και να επεξεργάζεται την καλλιέργεια του κάθε χωραφιού
- Να μεταφέρει τα χωράφια σε νέο καλλιεργητικό έτος
- Να δημιουργεί και να επαναφέρει αντίγραφα ασφαλείας για κάθε καλλιεργητικό έτος
- Να εκτυπώνει λίστα αγροκτήματος έτοιμη για την δήλωση του ΟΠΕΚΕΠΕ

Τέλος παρέχει την δυνατότητα καταχώρησης μέχρι τεσσάρων ιδιοκτητών και ενός ενοικιαστή για κάθε χωράφι του αγροκτήματος.

Το λογισμικό είναι πλήρες παραμετροποιημένο για τις ανάγκες του κάθε αγρότη και της κάθε καλλιέργειας. Δηλαδή, δεν έχετε ένα πλήθος στάνταρ εργασιών που μπορείτε να κάνετε στο χωράφι. Μπορείτε να ορίσετε τις δικές σας νέες εργασίες και να τις χειριστείτε όπως εσείς ξέρετε. Ξεκινάμε ορίζοντας ένα έτος χρήσης και το όνομα του αγροκτήματος. Την πρώτη φορά θα πρέπει να αφιερώσετε λίγο χρόνο να καταχωρήσετε τα χωράφια σας που ανήκουν στο αγρόκτημα. Εάν οριστεί το αγρόκτημα την πρώτη φορά τότε την επόμενη χρονιά απλά μεταφέρεται τα χωράφια σε νέα χρήση έχοντας την δυνατότητα να προσθέσετε ή να αναιρέσετε χωράφια.

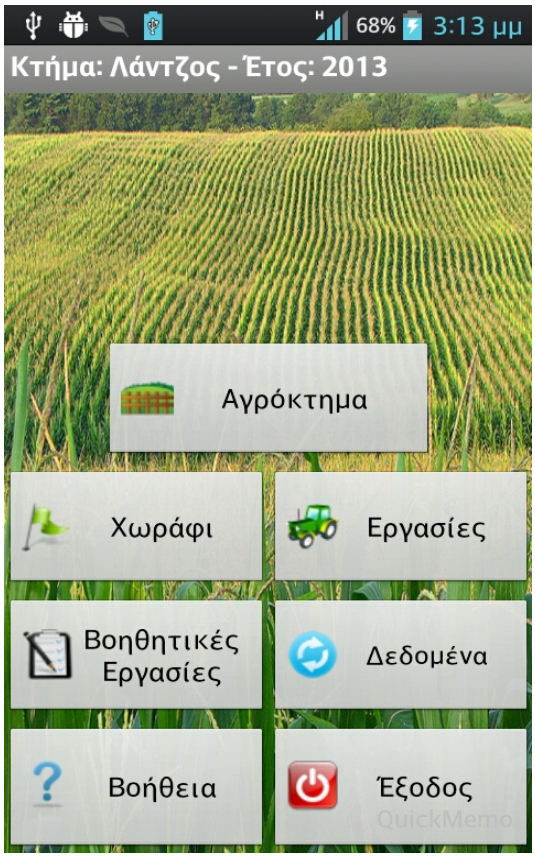
Η διαχείριση των χωραφιών γίνεται πολύ εύκολα με την χρήση του γεωγραφικού χάρτη από το google maps online και με την οθόνη αφής. Απλά, μεταφέρεστε στο χωράφι που θέλετε και κάνοντας παρατεταμένο κλικ ορίζεται το χωράφι που ανήκει στο αγρόκτημα. Στην πορεία δηλώνετε όλα τα σταθερά στοιχεία του χωραφιού (όνομα, τοποθεσία, τύπος εδάφους, έκταση, ιδιοκτήτης, αριθμός ΟΠΕΚΕΠΕ). Εάν ένας ιδιοκτήτης δεν υπάρχει τότε απλά καταχωρείται τα στοιχεία του ιδιοκτήτη (όνομα, επώνυμο, ΑΦΜ, τηλέφωνο). Στο χωράφι ορίζετε και την σχέση που έχετε. Δηλαδή εάν είναι με ενοίκιο την διάρκεια του συμβολαίου ή εάν θα το δηλώνετε εσείς ή ο ιδιοκτήτης. Όταν ορίσετε ένα χωράφι και το αποθηκεύετε τότε επάνω στον google maps θα βγει μια σημαία η οποία θα σας δείχνει το νέο χωράφι που ορίσατε. Όταν τελειώσετε τον ορισμό των χωραφιών τότε θα δείτε μια περιοχή που θα έχει σημαίες. Αυτά θα είναι τα χωράφια σας μέσα στο αγρόκτημα.

Στις εργασίες τα πράγματα είναι πολύ πρακτικά και εύκολα. Μεταφέρεστε από το google maps απευθείας στο χωράφι που θέλετε. Όταν πατάτε επάνω στην σημαία, σας εμφανίζει το όνομα του χωραφιού, την ιδιότητα (ιδιόκτητο ή ενοικιαζόμενο) και τι είναι σπαρμένο. Πατώντας διπλό κλικ στην οθόνη αφής μπορείτε να λάβετε τα πλήρη στοιχεία του χωραφιού και να ορίσετε εργασία. Οι εργασίες μπορούν να ορισθούν απευθείας και από το κεντρικό μενού, όπου επιλέγετε το χωράφι από μια λίστα, το είδος της εργασίας, την ημερομηνία και σημειώσεις. Εάν μια εργασία δεν υπάρχει μπορείτε να την ορίσετε εσείς ή να διαγράψετε σε περίπτωση που δεν σας αφορούνε. Για παράδειγμα εσείς μπορείτε να έχετε δένδρα και οι εργασίες σας να είναι διαφορετικές από τις καλοκαιρινές καλλιέργειες. Τότε δεν έχετε απλά να αλλάξετε το ονόματα της καλλιέργειας και τα ονόματα των εργασιών σας για να κάνετε την δουλειά σας.

Τα δεδομένα μιας χρονιάς μπορούνε να αποθηκευτούνε σε αρχείο και να το πάρετε σαν αντίγραφο ασφαλείας Για να μπορείτε να επισκοπήσετε εργασίες περασμένων ετών δεν έχετε απλά να αλλάξετε το έτος χρήσης και αυτόματα όλα τα δεδομένα θα έρθουνε μπροστά σας. Η εφαρμογή είναι πάρα πολύ γρήγορη και απαιτεί πολύ μικρό μέγεθος μνήμης και αποθηκευτικό χώρο για την διαχείριση των δεδομένων. Ενδεικτικά, για αγρόκτημα 600 στρεμμάτων καλαμποκιού με 60 τεμάχια και πλήρες εργασίες χρειάστηκαν μόνον 60kb δεδομένα όταν ένα smartphone διαθέτει αποθηκευτικό χώρο 4GB μιας σειράς οικονομικής μέχρι 200 ευρώ.

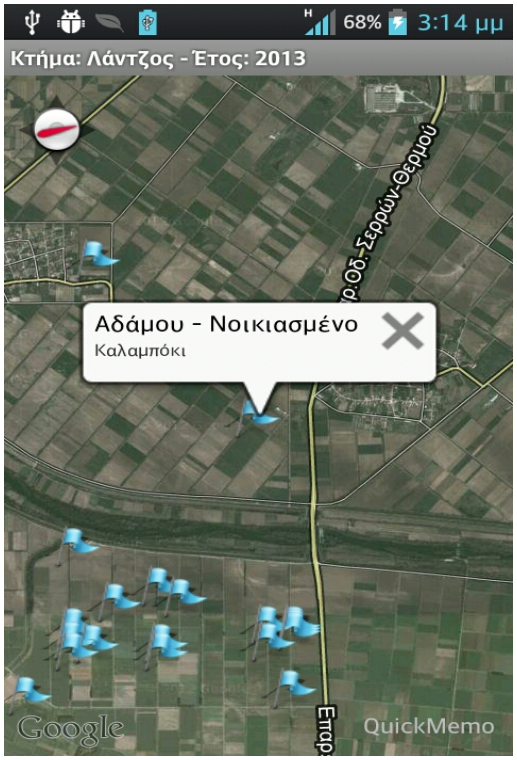
5.2 Σχεδιασμός και υλοποίηση εφαρμογής

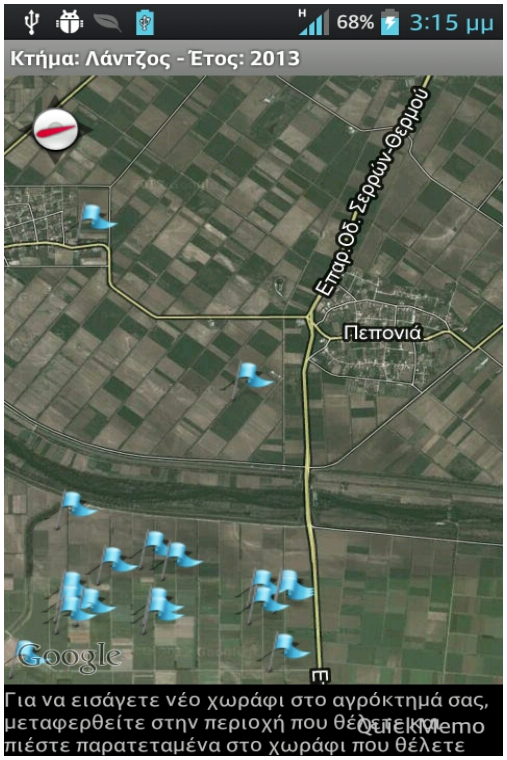
Αρχικά θα ξεκινήσουμε σχεδιάζοντας της οθόνες που θα μας χρειαστούν για την εφαρμογή μας. Παρακάτω φαίνονται οι κεντρικές οθόνες της εφαρμογής μας.



Αριστερά Εικόνα 5.1: Ρυθμίσεις εγκατάστασης εφαρμογής

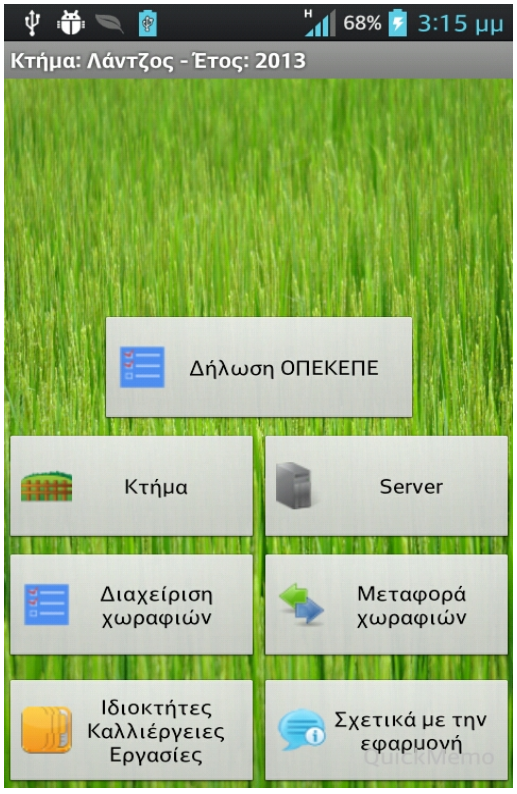
Δεξιά Εικόνα 5.2: Κεντρικό μενού της εφαρμογής





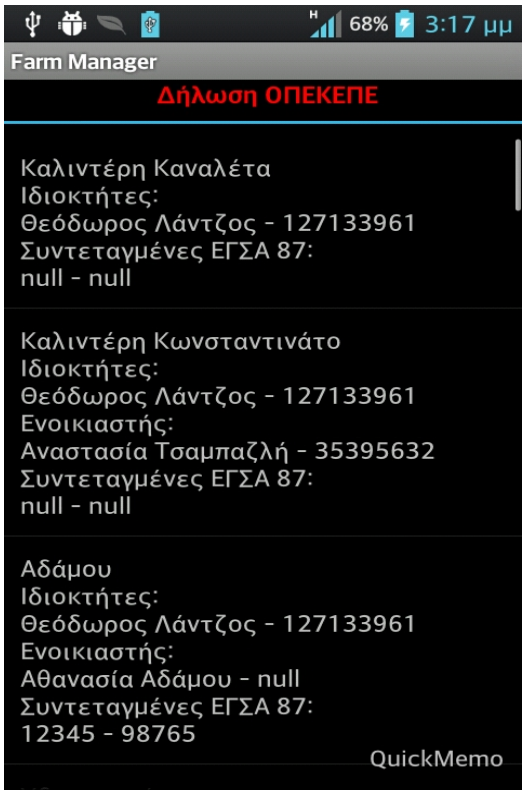
Αριστερά Εικόνα 5.3: Εμφάνιση χωραφιών του αγροκτήματός μας

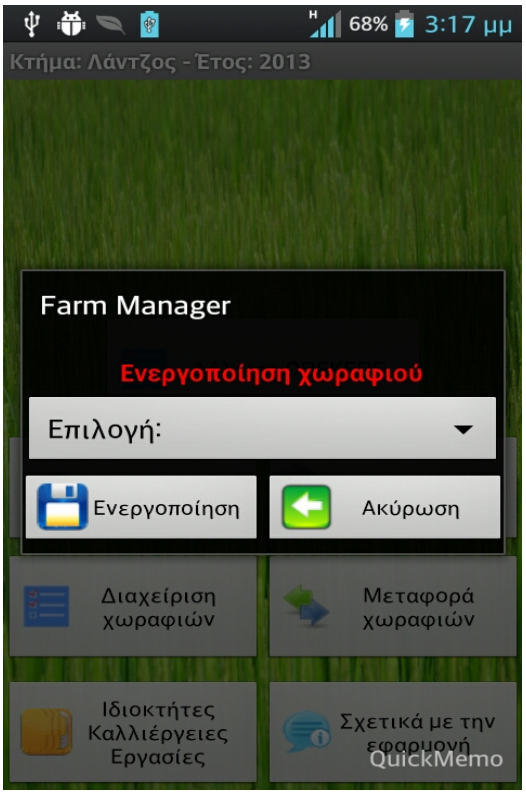
Δεξιά Εικόνα 5.4: Εισαγωγή νέου χωραφιού στο αγρόκτημα



Αριστερά Εικόνα 5.5: Βοηθητικές εργασίες της εφαρμογής

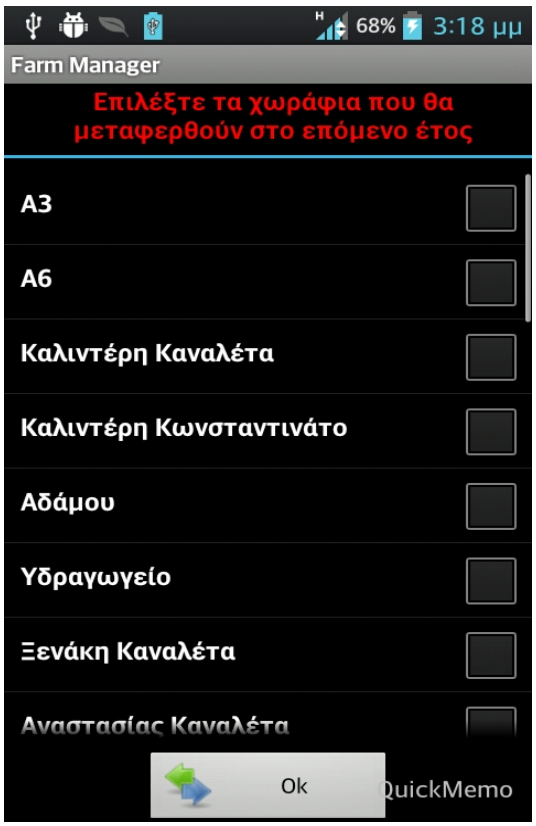
Δεξιά Εικόνα 5.6: Μενού ιδιοκτητών/εργασιών/καλλιεργειών

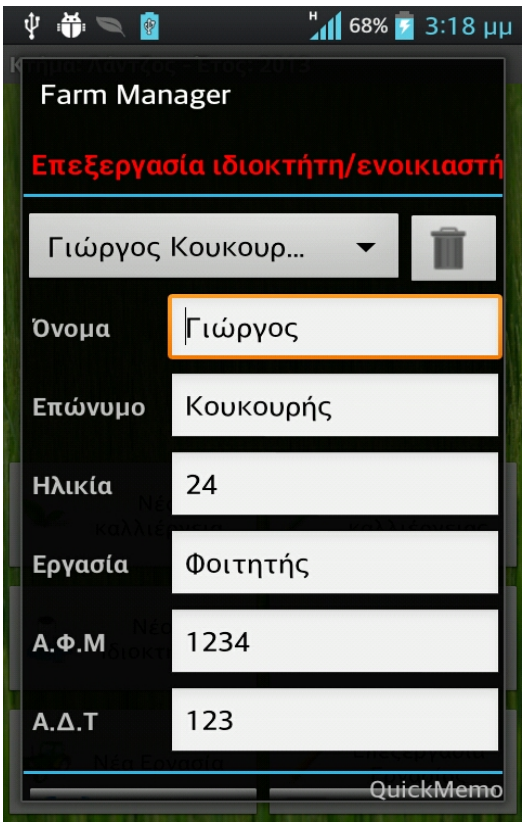




Αριστερά Εικόνα 5.7: Προβολή δήλωσης ΟΠΕΚΕΠΕ

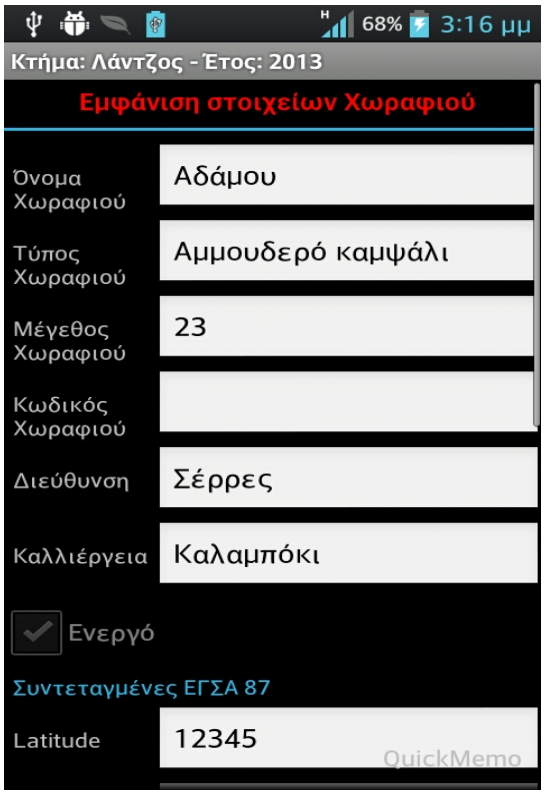
Δεξιά Εικόνα 5.8: Ενεργοποίηση χωραφιών

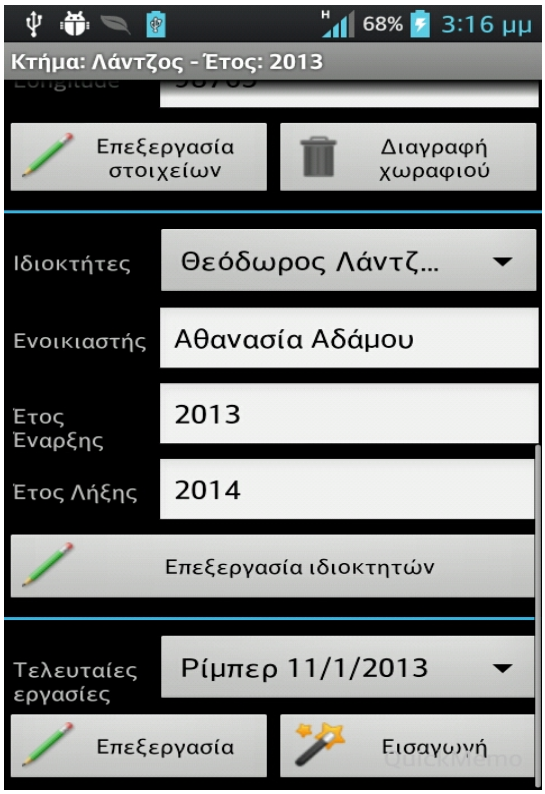




Αριστερά Εικόνα 5.9: Μεταφορά χωραφιών σε νέο καλλιεργητικό έτος

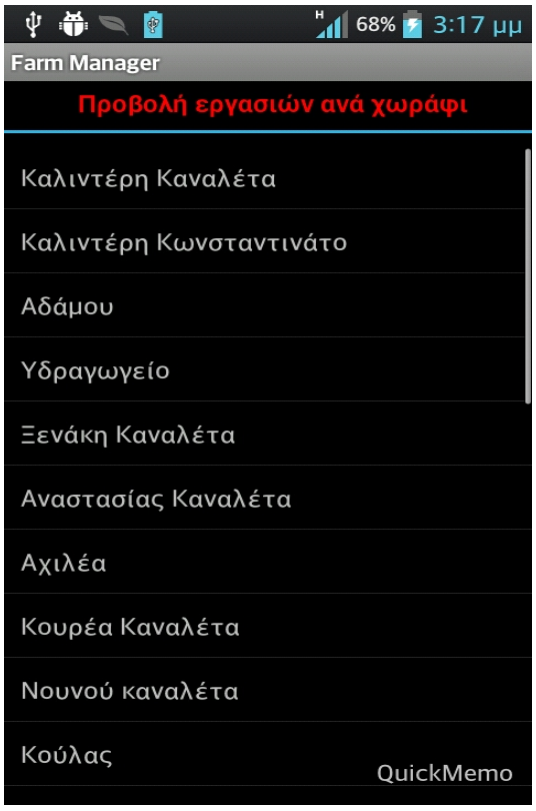
Δεξιά Εικόνα 5.10: Επεξεργασία στοιχείων ιδιοκτήτη/ενοικιαστή

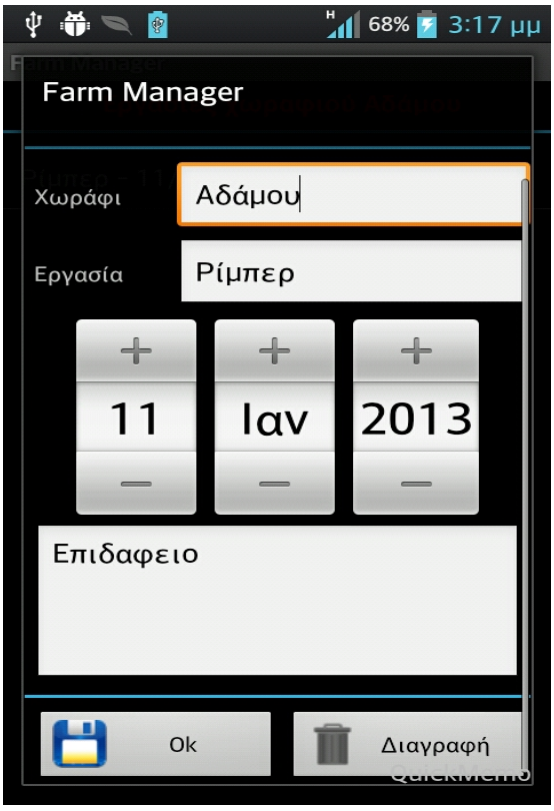




Αριστερά Εικόνα 5.11: Προβολή στοιχείων χωραφιού

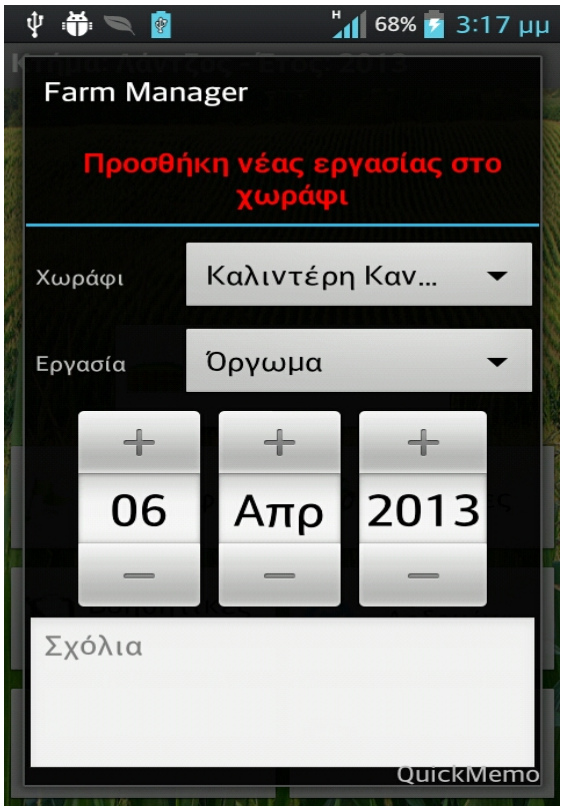
Δεξιά Εικόνα 5.12: Προβολή στοιχείων χωραφιού





Αριστερά Εικόνα 5.13: Προβολή εργασιών ανά χωράφι

Δεξιά Εικόνα 5.14: Προβολή στοιχείων εργασίας ενός χωραφιού



Εικόνα 5.15: Εισαγωγή νέας εργασίας σε χωράφι

5.2.1 Οθόνη ρυθμίσεων εγκατάστασης

Η οθόνη αυτή [Εικόνα 5.1] εμφανίζεται μόνο την πρώτη φορά που ανοίγουμε την εφαρμογή μετά την εγκατάσταση. Εδώ χρειάζεται να ορίσουμε τα στοιχεία μας τα οποία και αποθηκεύονται. Παρακάτω φαίνεται ο κώδικας του xml που χρειάστηκε για τη δημιουργία της οθόνης.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/scrollView1"    android:layout_width="match_parent"
android:layout_height="wrap_content" >
   <LinearLayout
      android:layout_width="match_parent"   android:layout_height="wrap_content"
      android:orientation="vertical" >
      <LinearLayout
         android:layout_width="match_parent"    android:layout_height="wrap_content"
         android:orientation="horizontal" >
         <TextView
            android:id="@+id/textView1"   android:layout_width="match_parent"
            android:layout_height="match_parent"   android:gravity="center"
            android:text="@string/stSetSettingsTitle"     android:textSize="22sp" />
      </LinearLayout>
      <LinearLayout
         android:layout_width="match_parent"    android:layout_height="wrap_content"
         android:layout_marginTop="7dp"    android:orientation="horizontal" >
         <TextView
```

```
      android:id="@+id/textView2"    android:layout_width="match_parent"
      android:layout_height="match_parent"   android:layout_weight="70"
      android:gravity="center"          android:text="@string/stPersonName"
      android:textSize="18sp" />
   <EditText
      android:id="@+id/etSetSettingsName"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="30" />
</LinearLayout>
<LinearLayout
   android:layout_width="match_parent"       android:layout_height="wrap_content"
   android:layout_marginTop="3dp"       android:orientation="horizontal" >
   <TextView
      android:id="@+id/textView3"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="70"
      android:gravity="center"          android:text="@string/stPersonSurname"
      android:textSize="18sp" />
   <EditText
      android:id="@+id/etSetSettingsSurname"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="30" />
</LinearLayout>
<LinearLayout
   android:layout_width="match_parent"       android:layout_height="wrap_content"
   android:layout_marginTop="3dp"       android:orientation="horizontal" >
   <TextView
      android:id="@+id/textView4"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="70"
      android:gravity="center"          android:text="@string/stPersonAge"
      android:textSize="18sp" />
   <EditText
      android:id="@+id/etSetSettingsAge"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:inputType="number"
      android:layout_weight="30" />
</LinearLayout>
<LinearLayout
   android:layout_width="match_parent"       android:layout_height="wrap_content"
   android:layout_marginTop="3dp"       android:orientation="horizontal" >
   <TextView
      android:id="@+id/textView5"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="70"
      android:gravity="center"          android:text="@string/stPersonJob"
      android:textSize="18sp" />
   <EditText
      android:id="@+id/etSetSettingsJob"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="30" />
</LinearLayout>
<LinearLayout
   android:layout_width="match_parent"       android:layout_height="wrap_content"
   android:layout_marginTop="3dp"       android:orientation="horizontal" >
   <TextView
      android:id="@+id/textView6"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="70"
      android:gravity="center"          android:text="@string/stPersonADT"
      android:textSize="18sp" />

   <EditText
      android:id="@+id/etSetSettingsADT"          android:layout_width="match_parent"
      android:layout_height="match_parent"          android:layout_weight="30" />
</LinearLayout>
<LinearLayout
   android:layout_width="match_parent"       android:layout_height="wrap_content"
```

```
    android:layout_marginTop="3dp"          android:orientation="horizontal" >
    <TextView
      android:id="@+id/textView7"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:layout_weight="70"
      android:gravity="center"            android:text="@string/stPersonAFM"
      android:textSize="18sp" />
    <EditText
      android:id="@+id/etSetSettingsAFM"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:inputType="number"
      android:layout_weight="30" />
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"          android:layout_height="wrap_content"
    android:layout_marginTop="3dp"          android:orientation="horizontal"
    android:visibility="gone" >
    <TextView
      android:id="@+id/textView9"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:layout_weight="70"
      android:gravity="center"            android:text="@string/stPersonPassword"
      android:textSize="18sp" />
    <EditText
      android:id="@+id/etSetSettingsPassword"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:layout_weight="30" />
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"          android:layout_height="wrap_content"
    android:layout_marginTop="3dp"          android:orientation="horizontal" >
    <TextView
      android:id="@+id/textView10"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:layout_weight="70"
      android:gravity="center"            android:text="@string/stCurrentYear"
      android:textSize="18sp" />
    <EditText
      android:id="@+id/etSetSettingsCurrentYear"
      android:layout_width="match_parent"            android:layout_height="match_parent"
      android:inputType="number"            android:layout_weight="30" />
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"          android:layout_height="wrap_content"
    android:layout_marginTop="3dp"          android:orientation="horizontal" >
    <TextView
      android:id="@+id/textView13"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:layout_weight="70"
      android:gravity="center"            android:text="@string/stLandName"
     android:textSize="18sp" />
    <EditText
      android:id="@+id/etSetSettingsLandName"            android:layout_width="match_parent"
      android:layout_height="match_parent"            android:layout_weight="30" />
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"          android:layout_height="wrap_content"
    android:layout_marginBottom="3dp"          android:layout_marginTop="3dp"
    android:gravity="center"          android:orientation="horizontal" >
    <Button
      android:id="@+id/btSetSettingsOk"            android:layout_width="180dp"
      android:layout_height="wrap_content"          android:drawableLeft="@drawable/save_item"
      android:text="Αποθήκευση ρυθμίσεων" >
    </Button>
  </LinearLayout>
</LinearLayout></ScrollView>
```

Βλέπουμε ότι για τη δημιουργία της οθόνης χρησιμοποιήσαμε ένα **ScrollView** ώστε να φαίνεται καλά σε διαφορετικά μεγέθη οθονών και να μπορούμε να έχουμε την δυνατότητα κύλισης της οθόνης. Μέσα στο **ScrollView** έχουμε ένα μεγάλο **LinearLayout** στο οποίο δηλώνουμε ότι θα έχει κάθετη κατεύθυνση περιεχομένων ώστε τα περιεχόμενά του να μπαίνουν το ένα κάτω από το άλλο. Μέσα στο μεγάλο **LinearLayout** έχουμε πολλά μικρά **LinearLayout** με κατεύθυνση οριζόντια και μέσα σε αυτά έχουμε τα **EditText** και **TextView** που χρειαζόμαστε. Το αποτέλεσμα του παραπάνω κώδικα φαίνεται στην εικόνα 5.1

Για να λειτουργήσει αυτή η οθόνη, να εμφανίζεται μόνο την πρώτη φορά εκτέλεσης της εφαρμογής και να αποθηκεύει τα στοιχεία που συμπληρώνουμε χρειάστηκε να γράψουμε κάποιο κώδικα οποίος φαίνεται παρακάτω.

```
package com.georkouk.farmManager;
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class SetFirstTimeSettings extends Activity implements OnClickListener{
	EditText etName,etSurname,etAge,etJob,etAdt,etAfm,etYear,etEstateName;
	Button btSettingsOk;
	MapsDB db=new MapsDB(this);

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		setContentView(R.layout.setfirsttimesettings);
		etName=(EditText) findViewById(R.id.etSetSettingsName);
		etSurname=(EditText) findViewById(R.id.etSetSettingsSurname);
		etAge=(EditText) findViewById(R.id.etSetSettingsAge);
		etJob=(EditText) findViewById(R.id.etSetSettingsJob);
		etAdt=(EditText) findViewById(R.id.etSetSettingsADT);
		etAfm=(EditText) findViewById(R.id.etSetSettingsAFM);
		etYear=(EditText) findViewById(R.id.etSetSettingsCurrentYear);
```

```
                etEstateName=(EditText) findViewById(R.id.etSetSettingsLandName);
                btSettingsOk=(Button) findViewById(R.id.btSetSettingsOk);
                btSettingsOk.setOnClickListener(this);
        }

        public void onClick(View v) {
                // TODO Auto-generated method stub
                switch (v.getId()){
                        case R.id.btSetSettingsOk:
                                db.open();
                                db.createPerson(etName.getText().toString().trim(),
                                etSurname.getText().toString().trim(),
                                etAge.getText().toString().trim(), etAdt.getText().toString().trim(),
                                etAfm.getText().toString().trim(),etJob.getText().toString().trim());
                                db.createEstate(etEstateName.getText().toString().trim());
                                db.close();

                                db.open();
                                db.createCrop("Δεν έχει καταχωρηθεί καλλιέργεια");
                                db.close();

                                SharedPreferences.Editor editor =
SetFirstTimeSettings.this.getSharedPreferences("Settings", 0).edit();
                                editor.putBoolean("settingsOk", true);
                                editor.putString("currentYear", etYear.getText().toString());
                                editor.commit();
Intent openStartingPoint = new Intent("com.georkouk.farmManager.STARTMENU");
startActivity(openStartingPoint);
finish();
                }
        }
}
```

Όπως βλέπουμε στην OnCreate() συνάρτηση η οποία καλείται την πρώτη φορά που ανοίγει η οθόνη δηλώνουμε πιο xml θα χρησιμοποιήσουμε καθώς επίσης και τα components που θα χρησιμοποιήσουμε ανάλογα με το id που τα έχουμε δηλώσει στο xml. Στη συνέχεια αφού ο χρήστης συμπληρώσει τα στοιχεία του και πατήσει αποθήκευση τότε χρησιμοποιούμε συναρτήσεις της βάσης για να αποθηκεύσουμε τα στοιχεία αυτά. Θα δούμε αργότερα το πώς λειτούργει η βάση.

Αφού αποθηκεύσουμε τα στοιχεία, κρατάμε σε αρχείο SharedPreferences, που είναι στην ουσία αρχείο xml της εφαρμογής για να κρατάει διάφορες τιμές που θέλουμε, ότι έχουμε ήδη συμπληρώσει τα στοιχεία αυτά ώστε την επόμενη φορά που θα ξανανοίξει η εφαρμογή να μην μας ξαναεμφανίσει αυτή την οθόνη αλλά να πάει κατευθείαν στον αρχικό μενού της εφαρμογής. Τέλος καλούμε το Activity του αρχικού μενού της εφαρμογής και κλείνουμε την τωρινή οθόνη ώστε να μην μένει στην μνήμη του κινητού αφού δεν μας χρειάζεται πλέον.

5.2.2 Οθόνη κεντρικού μενού εφαρμογής

Η οθόνη αυτή αποτελεί την κεντρική οθόνη της εφαρμογής και την πρώτη οθόνη που βλέπει ο χρήστης μόλις ανοίξει την εφαρμογή [Εικόνα 5.2]. Από δω μπορεί ο χρήστης να δει το αγρόκτημά του, να εισάγει νέο χωράφι στο αγρόκτημα, να εισάγει μια νέα εργασία στο αγρόκτημα ή να δει τις ήδη καταχωρημένες εργασίες στα χωράφια του έχοντας την επιλογή να διαλέξει προβολή εργασιών ανά χωράφι ή ανά εργασία, να μεταφερθεί στις βοηθητικές εργασίες της εφαρμογής, να κρατήσει δεδομένα ασφαλείας για το τρέχον έτος ή και να επαναφέρει δεδομένα στην εφαρμογή, να δει τη βοήθεια για τη λειτουργία της εφαρμογής και όταν θέλει να κλείσει την εφαρμογή. Παρακάτω βλέπουμε τον κώδικα που χρειάστηκε για να δημιουργηθεί το xml αρχείο της οθόνης.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent" android:layout_height="match_parent"
  android:background="@drawable/start_menu3"
  android:orientation="vertical" >
   <LinearLayout
     android:id="@+id/l1"       android:layout_width="match_parent"
     android:layout_height="80dp" android:layout_alignParentBottom="true"
      android:layout_marginTop="5dp"   android:orientation="horizontal" >
      <Button
         android:id="@+id/btStartMenuAboutUs"          android:layout_width="match_parent"
 android:layout_height="match_parent"          android:layout_weight="0.5"
android:textSize="17sp"          android:drawableLeft="@drawable/help"
         android:text="@string/stHelp" >
      </Button>
      <Button
         android:id="@+id/btStartMenuExit"          android:layout_width="match_parent"
         android:layout_height="match_parent"          android:layout_weight="0.5"
         android:textSize="17sp"          android:drawableLeft="@drawable/exit"
         android:text="@string/stStartMenuExit" >
      </Button>
   </LinearLayout>
   <LinearLayout
     android:id="@+id/l2"       android:layout_width="match_parent"
```

```
    android:layout_height="75dp"       android:layout_above="@id/l1"
    android:layout_marginTop="5dp"       android:gravity="center"
    android:orientation="horizontal" >
    <Button
      android:id="@+id/btStartMenuSettings"          android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="0.5"          android:textSize="17sp"
      android:drawableLeft="@drawable/task"          android:text="@string/stStartMenuSettings" >
    </Button>
    <Button
      android:id="@+id/btStartMenuSync"          android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="0.5"          android:textSize="16sp"
      android:drawableLeft="@drawable/sync"          android:text="@string/stStartMenuSync" >
    </Button>
  </LinearLayout>
   <LinearLayout
    android:id="@+id/l3"       android:layout_width="match_parent"
    android:layout_height="75dp"       android:layout_above="@id/l2"
    android:layout_marginTop="5dp"       android:gravity="center"
    android:orientation="horizontal" >
    <Button
      android:id="@+id/btStartMenuNewFarm"          android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_weight="0.5"          android:textSize="17sp"
     android:drawableLeft="@drawable/new_farm" android:text="@string/stStartMenuNewFarm" >
    </Button>
    <Button
      android:id="@+id/btStartMenuNewJob"          android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:textSize="17sp"          android:layout_weight="0.5"
      android:drawableLeft="@drawable/new_job"    android:text="@string/stStartMenuNewJob" >
    </Button>
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="75dp"       android:layout_above="@id/l3"
    android:layout_marginTop="5dp"       android:gravity="center"
    android:orientation="horizontal" >
    <Button
      android:id="@+id/btStartMenuMap"   android:layout_width="200dp"
      android:textSize="17sp"    android:layout_height="match_parent"
      android:drawableLeft="@drawable/fence"
      android:text="@string/stStartMenuMap" >
    </Button>
        </LinearLayout>
</RelativeLayout>
```

Βλέπουμε ότι σε αυτή την οθόνη χρησιμοποιήσαμε το `RelativeLayout.` Αυτό είναι ένα είδος Layout με το οποίο μπορείς να κάνεις συσχετίσεις μεταξύ των components για καλύτερη οργάνωση στην οθόνη. Για παράδειγμα μπορείς να πεις σε ένα component να μπει στο κάτω μέρος τις οθόνης και σε ένα άλλο component να μπει από πάνω του ή δεξιά του κτλ. Για να είναι όμως λειτουργική αυτή η οθόνη χρειάστηκε και κάποιος κώδικας ώστε να δουλεύουν τα κουμπιά που έχει.

```
public class StartMenu extends Activity implements OnClickListener{
	Button btStartMaps, btExit, btAboutUs, btSettings, btNewFarm, btNewJob, btSync;
	TextView custom;
	EditText restoreYear;
	AlertDialog alert, restoreAlert;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
		super.onCreate(savedInstanceState);
		requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
		setContentView(R.layout.startmenu);
		 getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE,
R.layout.custom_title);
		initialize();		}

	@Override
	protected void onResume() {
		// TODO Auto-generated method stub
		super.onResume();
		SharedPreferences prefs = StartMenu.this.getSharedPreferences("Settings",0);
		String year=prefs.getString("currentYear", "");
		MapsDB db=new MapsDB(this);
		db.open();
		String name=db.estateName();
		db.close();
		String title="Κτήμα: " + name + " - Έτος: " + year;
		custom.setText(title);
		//this.setTitle(title);
	}

	public void initialize(){
		btStartMaps=(Button)findViewById(R.id.btStartMenuMap);
		btExit=(Button)findViewById(R.id.btStartMenuExit);
		btAboutUs=(Button) findViewById(R.id.btStartMenuAboutUs);
		btSettings=(Button) findViewById(R.id.btStartMenuSettings);
		btNewFarm=(Button) findViewById(R.id.btStartMenuNewFarm);
		btNewJob=(Button) findViewById(R.id.btStartMenuNewJob);
		btSync=(Button) findViewById(R.id.btStartMenuSync);
		custom=(TextView) findViewById(R.id.tvCustomTitle);
```

```
        btStartMaps.setOnClickListener(this);
        btExit.setOnClickListener(this);
        btAboutUs.setOnClickListener(this);
        btSettings.setOnClickListener(this);
        btNewFarm.setOnClickListener(this);
        btNewJob.setOnClickListener(this);
        btSync.setOnClickListener(this);
}

public void stats(View v){
        Intent stats=new Intent("com.georkouk.farmManager.ESTATESTATS");
        startActivity(stats);
}


public void onClick(View v) {
        // TODO Auto-generated method stub
        switch(v.getId()){
        case(R.id.btStartMenuMap):
                Intent map=new Intent("com.georkouk.farmManager.MAP");
                startActivity(map);
                break;
        case(R.id.btStartMenuAboutUs):
                Intent help=new Intent("com.georkouk.farmManager.HELP");
                startActivity(help);
                break;
        case(R.id.btStartMenuSettings):
                Intent settings=new Intent("com.georkouk.farmManager.SETTINGS");
                startActivity(settings);
                break;
        case(R.id.btStartMenuExit):
                this.finish();
                break;
        case(R.id.btStartMenuNewFarm):
                SharedPreferences.Editor editor = StartMenu.this.getSharedPreferences(
                                "Misc", 0).edit();
                editor.putString("mapFrom", "startMenu");
                editor.commit();
                Intent newFarm=new Intent("com.georkouk.farmManager.MAP");
                startActivity(newFarm);
```

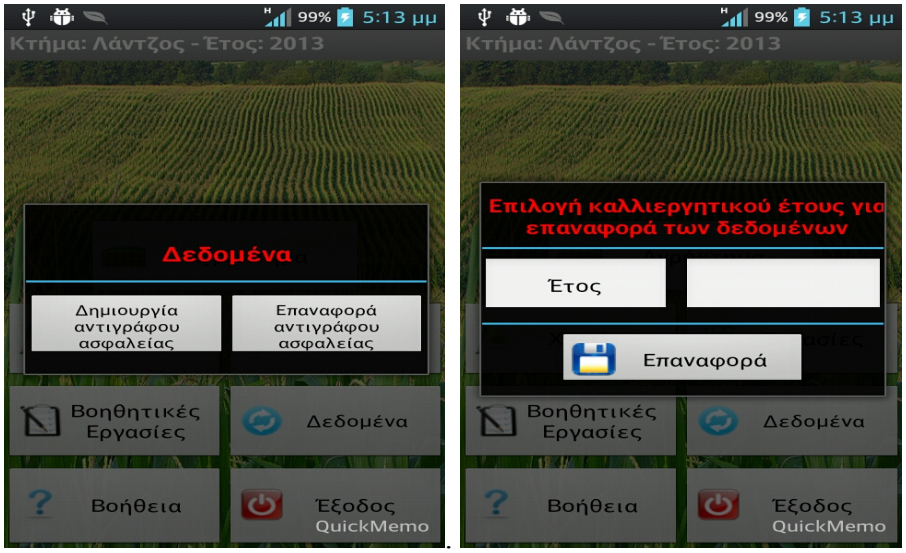
```
                    break;
                case(R.id.btStartMenuNewJob):
                    Intent newJob=new Intent("com.georkouk.farmManager.JOBS");
                    startActivity(newJob);
                    break;
                case(R.id.btStartMenuSync):
                    alert = new AlertDialog.Builder(StartMenu.this).create();
                alert.requestWindowFeature(Window.FEATURE_NO_TITLE);
                Rect displayRectangle = new Rect();
                Window window = this.getWindow();
                window.getDecorView().getWindowVisibleDisplayFrame(displayRectangle);
                // inflate and adjust layout
                LayoutInflater inflater =
(LayoutInflater)this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                View layout = inflater.inflate(R.layout.dialog, null);
                layout.setMinimumWidth((int)(displayRectangle.width() * 0.8f));
                //layout.setMinimumHeight((int)(displayRectangle.height() * 0.33f));
                alert.setCancelable(true);
                alert.setView(layout);
                alert.show();
                    break;
                }
        }
}
```

Όπως κάθε κλάση που θέλουμε να λειτουργεί σαν οθόνη για τον χρήστη, έτσι και αυτή κάνει extend Activity. Το implements OnClickListener χρειάζεται ώστε τα κουμπιά της οθόνης να μπορούν να «ακούνε» τα κλικ των χρηστών. Αρχικά πάλι στην OnCreate() δηλώνουμε πιο xml θα μας εμφανίζει όταν ανοίγει και δηλώνουμε όλα τα components που έχουμε βάλει στην οθόνη. Στη συνέχεια χρησιμοποιούμε τη switch-case ώστε να πάρουμε όλες τις δυνατές περιπτώσεις κλικ του χρήστη. Ανάλογα την περίπτωση του κλικ κάνουμε και τις αντίστοιχες ενέργειες που πρέπει να γίνουν.

Με παρόμοιο τρόπο και ανάλογα με την κάθε περίπτωση και τις λειτουργίες που θέλουμε φτιάχνουμε και τις υπόλοιπες οθόνες της εφαρμογής μας.

5.3 Δημιουργία AlertDialog

Όπως είχαμε δει και σε προηγούμενο κεφάλαιο το AlertDialog είναι ένα είδος διαλόγου προς τον χρήστη. Εμφανίζεται σε μικρό παραθυράκι στο κέντρο της οθόνης και μπορούμε να προσθέσουμε και κουμπιά προτρέποντας τον χρήστη να επιλέξει κάτι. Παρακάτω βλέπουμε τη λειτουργία των αντιγράφων ασφαλείας η οποία έχει γίνει με AlertDialogs



Εικόνες 5.16 , 5.17: AlertDialog για επιλογή δημιουργίας ή επαναφοράς δεδομένων και AlertDialog για την επαναφορά δεδομένων

Αρχικά, για την εικόνα 5.16, δημιουργήσαμε ένα δικό μας xml αρχείο το οποίο φαίνεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"   android:layout_height="match_parent"
   android:orientation="vertical" >
   <LinearLayout
      android:layout_width="match_parent"      android:layout_height="wrap_content"
      android:layout_marginBottom="3dp"      android:layout_marginTop="3dp"
      android:gravity="center"      android:orientation="horizontal" >
      <TextView
         android:layout_width="fill_parent"         android:layout_height="wrap_content"
         android:layout_marginTop="10dp"         android:gravity="center"
         android:padding="10dp"         android:text="Δεδομένα"
         android:textColor="#FF0000"         android:textSize="20sp"
         android:textStyle="bold" />
   </LinearLayout>
   <View
      android:layout_width="match_parent"      android:layout_height="2dp"
      android:layout_marginBottom="8dp"      android:layout_marginTop="4dp"
      android:background="#33B5E5" />
   <LinearLayout
      android:layout_width="match_parent"      android:layout_height="wrap_content"
      android:layout_marginBottom="3dp"      android:layout_marginTop="3dp"
```

```
    android:gravity="center"       android:orientation="horizontal" >
    <Button
      android:id="@+id/btBackup"          android:layout_width="match_parent"
      android:layout_height="wrap_content"          android:layout_weight="50"
      android:onClick="backupDB"          android:padding="5dp"
      android:text="Δημιουργία αντιγράφου ασφαλείας" />
    <Button
      android:id="@+id/btRestore"          android:layout_width="match_parent"
      android:layout_height="wrap_content"          android:layout_weight="50"
      android:onClick="restoreDB"          android:padding="5dp"
      android:text="Επαναφορά αντιγράφου ασφαλείας" />
  </LinearLayout>
  <LinearLayout
    android:layout_width="match_parent"       android:layout_height="wrap_content"
    android:layout_marginBottom="3dp"       android:layout_marginTop="3dp"
    android:gravity="center"       android:orientation="horizontal" >
    <Button
      android:id="@+id/btSendToServer"          android:layout_width="match_parent"
      android:layout_height="wrap_content"          android:onClick="sendToServer"
      android:padding="5dp"          android:visibility="gone"
      android:text="Αποστολή δεδομένων"/>
  </LinearLayout>
</LinearLayout>
```

Θα μπορούσαμε βέβαια να χρησιμοποιήσουμε το default layout ενός AlertDialog χωρίς να κάνουμε κάποιο δικό μας, αλλά επιλέξαμε να κάνουμε ένα δικό μας ώστε να μπορούμε να σχεδιάσουμε όπως θέλουμε. Βλέπουμε ότι είναι ένα απλό layout.

Για να εμφανίζεται το παραπάνω dialog μόλις ο χρήστης κάνει κλικ στο κουμπί δεδομένα, χρησιμοποιήθηκε ο παρακάτω κώδικας.

```
alert = new AlertDialog.Builder(StartMenu.this).create();
alert.requestWindowFeature(Window.FEATURE_NO_TITLE);
Rect displayRectangle = new Rect();
Window window = this.getWindow();
window.getDecorView().getWindowVisibleDisplayFrame(displayRectangle);
// inflate and adjust layout
LayoutInflater inflater =
(LayoutInflater)this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
View layout = inflater.inflate(R.layout.dialog, null);
layout.setMinimumWidth((int)(displayRectangle.width() * 0.8f));
//layout.setMinimumHeight((int)(displayRectangle.height() * 0.33f));
alert.setCancelable(true);
alert.setView(layout);
alert.show();
```

Βλέπουμε ότι και εδώ δηλώνουμε πιο xml θα χρησιμοποιήσουμε αν θέλουμε κάποιο δικό μας. Επίσης ορίζουμε το μέγεθος που θα έχει το παραθυράκι. Ο κώδικας που εκτελείται όταν κάποιο από τα 2 κουμπιά πατηθεί από τον χρήστη φαίνεται παρακάτω.

```
public boolean backupDB(View v){

alert.dismiss();
SharedPreferences prefs = StartMenu.this.getSharedPreferences("Settings",0);
String year=prefs.getString("currentYear", "");
//creating a new folder for the database to be backuped to
```

```
        File direct = new File(Environment.getExternalStorageDirectory() + "/FarmManager/"+year);
          if(!direct.exists())
           {
              direct.mkdir();
           }
          MapsDB db=new MapsDB(this);
          db.exportDB(year, true);
                    return true;
}

public boolean restoreDB(View v){
                    alert.dismiss();
                    restoreAlert = new AlertDialog.Builder(StartMenu.this).create();
                    restoreAlert.requestWindowFeature(Window.FEATURE_NO_TITLE);
            Rect displayRectangle = new Rect();
            Window window = this.getWindow();
            window.getDecorView().getWindowVisibleDisplayFrame(displayRectangle);
            // inflate and adjust layout
            LayoutInflater inflater =
(LayoutInflater)this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            View layout = inflater.inflate(R.layout.restore_dialog, null);
            layout.setMinimumWidth((int)(displayRectangle.width() * 0.8f));
            //layout.setMinimumHeight((int)(displayRectangle.height() * 0.33f));
            restoreAlert.setCancelable(true);
            restoreAlert.setView(layout);
            restoreAlert.show();
                    return true;
}
```

Στην πρώτη συνάρτηση απλά κλείνουμε το dialog και δημιουργούμε ένα αντίγραφο ασφαλείας των δεδομένων της εφαρμογής μας. Στην δεύτερη συνάρτηση η οποία είναι για την επαναφορά δεδομένων εμφανίζουμε και ένα ακόμα dialog που προτρέπουμε τον χρήστη να συμπληρώσει για πιο έτος θέλει να γίνει η επαναφορά των δεδομένων. Το xml αρχείο για αυτό το dialog φαίνεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/scrollView1"    android:layout_width="match_parent"
   android:layout_height="240dp" >
   <LinearLayout
      android:layout_width="match_parent"       android:layout_height="wrap_content"
      android:orientation="vertical" >
     <LinearLayout
        android:layout_width="match_parent"          android:layout_height="wrap_content"
        android:layout_marginBottom="3dp"          android:layout_marginTop="3dp"
        android:gravity="center"          android:orientation="horizontal" >
        <TextView
           android:id="@+id/tvPersonTitle"             android:layout_width="300dp"
           android:layout_height="wrap_content"             android:layout_marginLeft="5dp"
           android:gravity="center"
           android:text="Επιλογή καλλιεργητικού έτους για επαναφορά των δεδομένων"
           android:textColor="#FF0000"             android:textSize="18sp"
           android:textStyle="bold" >
        </TextView>
     </LinearLayout>
     <View
        android:layout_width="match_parent"          android:layout_height="2dp"
        android:layout_marginBottom="8dp"          android:layout_marginTop="4dp"
```

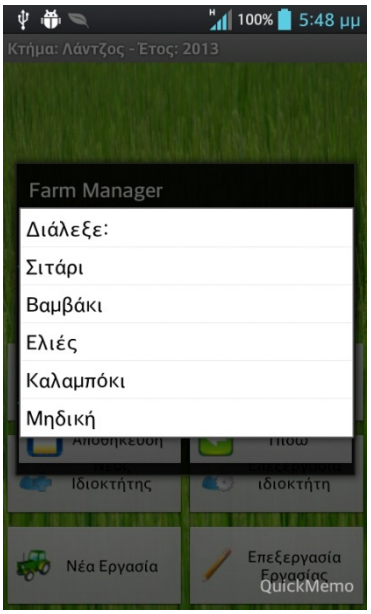
```
        android:background="#33B5E5" />
    <LinearLayout
        android:layout_width="fill_parent"          android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"          android:focusableInTouchMode="true"
        android:gravity="center"          android:orientation="horizontal" >
        <EditText
            android:id="@+id/editText5"            android:layout_width="match_parent"
            android:layout_height="wrap_content"            android:layout_weight="50"
            android:editable="false"            android:focusable="false"
            android:gravity="center"            android:layout_marginRight="10dp"
            android:padding="15dp"            android:text="Ετος"
            android:textSize="17sp" />

        <EditText
            android:id="@+id/etSettingsRestoreYear"            android:layout_width="match_parent"
            android:layout_height="wrap_content"            android:layout_weight="50"
            android:imeOptions="actionDone"            android:padding="15dp"
            android:inputType="number"            android:singleLine="true"
            android:textSize="17sp" />
    </LinearLayout>
    <View
        android:layout_width="match_parent"          android:layout_height="2dp"
        android:layout_marginBottom="8dp"          android:layout_marginTop="4dp"
        android:background="#33B5E5" />
    <LinearLayout
        android:layout_width="match_parent"          android:layout_height="wrap_content"
        android:layout_marginBottom="3dp"          android:gravity="center"
        android:orientation="horizontal" >
        <Button
            android:id="@+id/btRestoreYearSelect"            android:layout_width="180dp"
            android:layout_height="wrap_content"  android:drawableLeft="@drawable/save_item"
            android:onClick="restoreDB2"            android:padding="10dp"
            android:text="Επαναφορά"            android:textSize="17sp" >
        </Button>
    </LinearLayout>
  </LinearLayout>
</ScrollView>
```

Ο κώδικας που υπάρχει για να λειτουργεί το κουμπί φαίνεται παρακάτω.

```
public boolean restoreDB2(View v){
restoreYear=(EditText) restoreAlert.findViewById(R.id.etSettingsRestoreYear);
InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
imm.hideSoftInputFromWindow(restoreYear.getWindowToken(), 0);
restoreAlert.dismiss();
String year=restoreYear.getText().toString();

//creating a new folder for the database to be backuped from
File direct = new File(Environment.getExternalStorageDirectory() + "/FarmManager/"+year);

if(!direct.exists())
{
        direct.mkdir();
}
    MapsDB db=new MapsDB(this);
    db.importDB(year);
   SharedPreferences prefs = StartMenu.this.getSharedPreferences("Settings",0);
  String title=prefs.getString("title", "");
  custom.setText(title);
return true;       }
```

5.4 Χρήση spinner στην εφαρμογή

Το spinner είναι ένα drop-down menu το οποίο μπορούμε να χρησιμοποιήσουμε πολλούς τρόπους. Μπορούμε απλά να εμφανίζουμε πολλές πληροφορίες στο χρήστη και για να μην πιάνουν χώρο χρησιμοποιούμε ένα spinner ή μπορούμε να εμφανίζουμε στο χρήστη ένα μενού με διάφορες επιλογές. Η δημιουργία του είναι απλή και παρακάτω θα δούμε τη δημιουργία ενός spinner από την εφαρμογή.



Εικόνα 5.18: Εμφάνιση καλλιεργειών με χρήση spinner

Για να γεμίσουμε ένα spinner χρειαζόμαστε έναν ArrayAdapter στον οποίο δηλώνουμε τις τιμές που θέλουμε να εμφανίζουμε και ένα xml με ένα TextView μέσα ώστε να μπορεί να τα εμφανίσει όπως φαίνεται παρακάτω.

```
allCrops=null;
db.open();
allCrops=db.fetchAllCrops();
db.close();
allCropsList.clear();
allCropsList.add("Διάλεξε:");
for (int i = 1; i < allCrops.size(); i++) {
        allCropsList.add(allCrops.get(i).get(1));
}
cropsSpinnerAdapter = new ArrayAdapter<String>(this, R.layout.spinner_textview, allCropsList);
spJobs.setAdapter(cropsSpinnerAdapter);
spJobs.dispatchSetSelected(true);
```

Το xml για τον Adapter είναι το παρακάτω.

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/etListSpinner"    android:layout_width="fill_parent"
 android:layout_height="wrap_content"  android:padding="4dp"
 android:singleLine="true" android:textColor="#000000"
```

```
 android:textSize="18sp" >
</TextView>
```

Εφόσον θέλουμε ο χρήστης να μπορεί να επιλέγει από τη λίστα κάτι θα πρέπει να κάνουμε το spinner να ακούει στα κλικ του χρήστη. Αυτό γίνεται με τον παρακάτω τρόπο.

```
spJobs=(Spinner) findViewById(R.id.spJobs);
spJobs.setOnItemSelectedListener(
new OnItemSelectedListener() {
public void onItemSelected(
AdapterView<?> parent, View view, int pos, long id) {
        if(pos!=0){
                        selectedItem=parent.getItemAtPosition(pos).toString();
                        fillJobsCropsForm(allCrops.get(pos).get(1));
                        selectedId=allCrops.get(pos).get(0);
                        beforeChangeName=allCrops.get(pos).get(1);
        }
        else if(pos==0){
                        selectedId="";
                        beforeChangeName="";
        }
}

public void onNothingSelected (AdapterView<?> arg0) {}

} );
```

5.5 Χρήση listview στην εφαρμογή

Το listview είναι ένας άλλος τρόπος για προβολή δεδομένων στον χρήστη και δυνατότητα επιλογής από αυτά. Δουλεύει με παρόμοιο τρόπο με το spinner μόνο που το listview από την αρχή καταλαμβάνει χώρο, ενώ στο spinner πρέπει να κάνεις κλικ πάνω του για να δεις τα δεδομένα του. Και εδώ χρησιμοποιούμε adapter για να το γεμίσουμε και κάποιο αρχείο xml με το οποίο ορίζουμε το πώς θα φαίνεται. Υπάρχουν πολλές επιλογές όσον αφορά το layout ενός listview. Μπορούμε πχ να έχουμε κείμενο και εικόνα σε κάθε γραμμή, μόνο κείμενο, μόνο εικόνες, checkboxes ανά γραμμή κλπ. Όπως βλέπουμε είναι πιο εύχρηστο από το spinner. Παρακάτω θα δούμε τον τρόπο δημιουργίας ενός listview στην οθόνη μεταφοράς χωραφιών σε νέο έτος της εφαρμογής μας [Εικόνα 5.9].

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"    android:layout_height="fill_parent"
   android:orientation="vertical" >
   <LinearLayout
      android:layout_width="match_parent"       android:layout_height="wrap_content"
      android:orientation="horizontal" >
      <TextView
         android:id="@+id/textView11"           android:layout_width="wrap_content"
         android:layout_height="wrap_content"          android:layout_marginBottom="2dp"
```

```
        android:layout_marginLeft="5dp"          android:gravity="center"
        android:text="Επιλέξτε τα χωράφια που θα μεταφερθούν στο επόμενο έτος"
        android:textColor="#FF0000"          android:textSize="16sp"
        android:textStyle="bold" />
    </LinearLayout>
    <View
      android:layout_width="match_parent"       android:layout_height="2dp"
      android:layout_marginBottom="8dp"       android:layout_marginTop="4dp"
      android:background="#33B5E5" />
    <LinearLayout
      android:layout_width="match_parent"       android:layout_height="wrap_content"
      android:layout_weight="2.5"       android:orientation="horizontal" >
      <ListView
        android:id="@+id/mainListView"          android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
      </ListView>
    </LinearLayout>
    <LinearLayout
      android:layout_width="match_parent"       android:layout_height="wrap_content"
      android:gravity="center"       android:orientation="horizontal" >
      <Button
        android:id="@+id/btTransferOk"          android:layout_width="150dp"
        android:layout_height="wrap_content"          android:layout_gravity="center"
        android:layout_marginTop="5dp"       android:drawableLeft="@drawable/transfer_right_left"
        android:text="Ok" />
    </LinearLayout>
</LinearLayout>
```

Αυτό είναι το xml της οθόνης στην εικόνα 5.9. Όπως βλέπουμε είναι ένα απλό xml και περιέχει το listview και το κουμπί OK. Για να γεμίσουμε το listview χρησιμοποιούμε τον παρακάτω κώδικα.

```
db.open();
allFarms=db.fetchAllFarmNames();
db.close();
ArrayList<mItems> planetList = new ArrayList<mItems>();
for(int i=0;i<allFarms.size();i++){
          planetList.add(new mItems(allFarms.get(i).get(0)));
}
// Set our custom array adapter as the ListView's adapter.
listAdapter = new SelectArralAdapter(this, planetList);
mainListView.setAdapter(listAdapter);
```

Ο adapter που χρησιμοποιούμε εδώ είναι μια δικιά μας τροποποιημένη έκδοση του ArrayAdapter διότι μέσα στο listview έχουμε ckeckboxes σε κάθε γραμμή και θέλουμε όταν ο χρήστης κάνει κλικ σε κάποια γραμμή ή checkbox να κρατάει την επιλογή του σε μια λίστα ώστε να ξέρει το πρόγραμμα ποια χωράφια θέλει ο χρήστης να μεταφέρει στο νέο έτος. Για το λόγο αυτό και επειδή το listview ορισμένες φορές κατά το scrolling χάνει τις τιμές του κάναμε τον δικό μας adapter ώστε να λυθεί το πρόβλημα και να δουλεύει σωστά η εφαρμογή μας. Παρακάτω φαίνεται ο adapter.

```
private static class SelectArralAdapter extends ArrayAdapter<mItems> {
private LayoutInflater inflater;
public SelectArralAdapter(Context context, List<mItems> planetList) {
        super(context, R.layout.simplerow, R.id.rowTextView, planetList);
        // Cache the LayoutInflate to avoid asking for a new one each time.
        inflater = LayoutInflater.from(context);
        }
       @Override
        public View getView(int position, View convertView, ViewGroup parent) {
                        // Planet to display
        mItems planet = (mItems) this.getItem(position);

                        // The child views in each row.
        CheckBox checkBox;
        TextView textView;
                        // Create a new row view
        if (convertView == null) {
                convertView = inflater.inflate(R.layout.simplerow, null);
              // Find the child views.
                textView = (TextView) convertView.findViewById(R.id.rowTextView);
                checkBox = (CheckBox) convertView.findViewById(R.id.CheckBox01);
                // Optimization: Tag the row with it's child views, so we don't  have to
                // call findViewById() later when we reuse the row.
        convertView.setTag(new SelectViewHolder(textView, checkBox));
        // If CheckBox is toggled, update the planet it is tagged with.
        checkBox.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        CheckBox cb = (CheckBox) v;
                        mItems planet = (mItems) cb.getTag();
                        planet.setChecked(cb.isChecked());
                        if(planet.isChecked()){
                                deleted.remove(planet.getName());
                        }
                        else{
                                deleted.add(planet.getName());
                        }
                }
        });
        }
        // Reuse existing row view
        else {
        // Because we use a ViewHolder, we avoid having to call findViewById().
        SelectViewHolder viewHolder = (SelectViewHolder) convertView.getTag();
        checkBox = viewHolder.getCheckBox();
        textView = viewHolder.getTextView();
        }
        // Tag the CheckBox with the Planet it is displaying, so that we can
        // access the planet in onClick() when the CheckBox is toggled.
        checkBox.setTag(planet);
        // Display planet data
        checkBox.setChecked(planet.isChecked());
        textView.setText(planet.getName());
        return convertView;
        }
}
```

Κάθε φορά που ο χρήστης επιλέγει ένα χωράφι εκτελείται ο παρακάτω κώδικας ο οποίος αφαιρεί από την λίστα το χωράφι αυτό. Η λίστα αυτή στο τέλος περιέχει τα

χωράφια που δεν θα μεταφερθούν στο νέο έτος και όταν ο χρήστης πατήσει OK σβήνονται και μένουν μόνο τα υπόλοιπα χωράφια που είχε επιλέξει.

```
mainListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
public void onItemClick(AdapterView<?> parent, View item, int position, long id) {
mItems planet = listAdapter.getItem(position);
planet.toggleChecked();
SelectViewHolder viewHolder = (SelectViewHolder) item.getTag();
viewHolder.getCheckBox().setChecked(planet.isChecked());
if(planet.isChecked()){
        deleted.remove(planet.getName());
}
else{
        deleted.add(planet.getName());
}
}
});
```

5.6 Χρήση της SQLite στην εφαρμογή

Όπως είχαμε δει και πιο πριν το Android παρέχει την SQLite βάση δεδομένων για αποθήκευση δεδομένων. Παρέχει έτοιμες βιβλιοθήκες για τη χρήση της. Το μόνο που έχουμε να κάνουμε είναι να γράψουμε κώδικα και ερωτήματα sql για την δημιουργία της βάσης και την τροποποίησή της με δεδομένα. Ας δούμε λοιπόν τι χρειαζόμαστε για να μπορέσουμε να χρησιμοποιήσουμε την SQLite . Για το σκοπό αυτό έχουμε φτιάξει στην εφαρμογή μας την παρακάτω συνάρτηση.

```
public class MapsDB {
//person table----------------------------------------
public static final String KEY_PERSON = "_person";
public static final String KEY_NAME = "name";
public static final String KEY_SURNAME = "surname";
public static final String KEY_AGE = "age";
public static final String KEY_JOB = "job";
public static final String KEY_ADT = "adt";
public static final String KEY_AFM = "afm";
//farm table-----------------------------------------------
public static final String KEY_FARM = "_farm";
public static final String KEY_FARMNAME = "farmName";
public static final String KEY_FARMTYPE = "farmType";
public static final String KEY_FARMSIZE = "farmSize";
public static final String KEY_FARMCODE = "farmCode";
public static final String KEY_WORKING = "working";
public static final String KEY_RENTED = "rented";
//pinpoints table-------------------------------------------
public static final String KEY_PINPOINTS = "_pinpoints";
public static final String KEY_LATITUDE = "latitude";
public static final String KEY_LONGITUDE = "longitude";
public static final String KEY_EGSALAT = "egsaLatitude";
public static final String KEY_EGSALON = "egsaLongitude";
public static final String KEY_ADDRESS = "address";
//jobs table-----------------------------------------------
public static final String KEY_JOBS = "_jobs";
public static final String KEY_JOBTYPE = "jobType";
```

```
//farmJobs table--------------------------------------------
public static final String KEY_DATE = "date";
public static final String KEY_COMMENTS = "comments";
//crops table-----------------------------------------------
public static final String KEY_CROPS = "_crops";
public static final String KEY_CROPSTYPE = "cropsType";
//renders table---------------------------------------------
public static final String KEY_START = "start";
public static final String KEY_END = "end";
//estate table----------------------------------------------
public static final String KEY_ESTATE = "_estate";
public static final String KEY_ESTATENAME = "estateName";
//database tables names-------------------------------------
private static final String DATABASE_NAME = "farmManager";
private static final String DATABASE_PERSONTABLE = "person";
private static final String DATABASE_FARMTABLE = "farm";
private static final String DATABASE_PERSONFARMTABLE="personFarm";
private static final String DATABASE_PINPOINTSTABLE = "pinpoints";
private static final String DATABASE_JOBSTABLE = "jobs";
private static final String DATABASE_FARMJOBSTABLE = "farmJobs";
private static final String DATABASE_CROPSTABLE = "crops";
private static final String DATABASE_ESTATETABLE = "estate";
private static final String DATABASE_RENDERSTABLE = "renders";
private static final int DATABASE_VERSION = 1;

private DBHelper ourHelper;
private final Context ourContext;
private SQLiteDatabase ourDatabase;
Cursor c;


public static class DBHelper extends SQLiteOpenHelper {

public DBHelper(Context context) {
super(context, DATABASE_NAME, null, DATABASE_VERSION);
// TODO Auto-generated constructor stub
}

@Override
public void onCreate(SQLiteDatabase db) {
// TODO Auto-generated method stub
db.execSQL("Create table " + DATABASE_ESTATETABLE + " (" + KEY_ESTATE
+ " integer primary key autoincrement, " + KEY_ESTATENAME
+ " text NOT NULL );");
db.execSQL("Create table " + DATABASE_PERSONTABLE + " (" + KEY_PERSON
+ " integer primary key autoincrement, " + KEY_NAME
+ " text NOT NULL, " + KEY_SURNAME + " text NOT NULL, " + KEY_AGE
+ " text, " + KEY_JOB + " text, " + KEY_ADT
+ " text, " + KEY_AFM + " integer );");
db.execSQL("Create table " + DATABASE_FARMTABLE + " (" + KEY_FARM
+ " integer primary key autoincrement, " + KEY_FARMNAME + " text NOT NULL, "
+ KEY_FARMTYPE + " text, " + KEY_FARMSIZE + " real, "
+ KEY_FARMCODE + " text, " + KEY_WORKING + " integer, "
+ KEY_RENTED + " integer, " + KEY_ESTATE + " integer references "
+ DATABASE_ESTATETABLE + " ( " + KEY_ESTATE + " ), " + KEY_CROPS
+ " integer references " + DATABASE_CROPSTABLE + " ( " + KEY_CROPS + "));");
db.execSQL("Create table " + DATABASE_PINPOINTSTABLE + " (" + KEY_PINPOINTS
+ " integer primary key autoincrement, " + KEY_LATITUDE + " real NOT NULL, "
+ KEY_LONGITUDE + " real NOT NULL, " + KEY_ADDRESS + " text NOT NULL, "
+ KEY_EGSALAT + "text, " + KEY_EGSALON + "text, "
```

```
+ KEY_FARM + " integer NOT NULL references " + DATABASE_FARMTABLE
+ " (" + KEY_FARM + " ));");
db.execSQL("Create table " + DATABASE_JOBSTABLE + " (" + KEY_JOBS
+ " integer primary key autoincrement, " + KEY_JOBTYPE + " text NOT NULL );");
db.execSQL("Create table " + DATABASE_FARMJOBSTABLE + " ( " + KEY_FARM
+ " integer NOT NULL references " + DATABASE_FARMTABLE
+ " (" + KEY_FARM + " ), " + KEY_JOBS + " integer NOT NULL references "
+ DATABASE_JOBSTABLE + " (" + KEY_JOBS + " ), " + KEY_DATE
+ " integer, " + KEY_COMMENTS + " text  );");
db.execSQL("Create table " + DATABASE_CROPSTABLE + " (" + KEY_CROPS
+ " integer primary key autoincrement, " + KEY_CROPSTYPE + " text NOT NULL );");
db.execSQL("Create table " + DATABASE_RENDERSTABLE + " ( " + KEY_FARM
+ " integer NOT NULL references " + DATABASE_FARMTABLE
+ " (" + KEY_FARM + " ), " + KEY_PERSON + " integer NOT NULL references "
+ DATABASE_PERSONTABLE + " (" + KEY_PERSON + " ), " + KEY_START
+ " text, " + KEY_END + " text );");
db.execSQL("Create table " + DATABASE_PERSONFARMTABLE + " ( " + KEY_FARM
+ " integer NOT NULL references " + DATABASE_FARMTABLE
+ " (" + KEY_FARM + " ), " + KEY_PERSON + " integer NOT NULL references "
+ DATABASE_PERSONTABLE + " (" + KEY_PERSON + " ));");
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
// TODO Auto-generated method stub
db.execSQL("drop table if exists " + DATABASE_PERSONTABLE);
db.execSQL("drop table if exists " + DATABASE_FARMTABLE);
db.execSQL("drop table if exists " + DATABASE_PINPOINTSTABLE);
db.execSQL("drop table if exists " + DATABASE_JOBSTABLE);
db.execSQL("drop table if exists " + DATABASE_FARMJOBSTABLE);
db.execSQL("drop table if exists " + DATABASE_CROPSTABLE);
db.execSQL("drop table if exists " + DATABASE_RENDERSTABLE);
db.execSQL("drop table if exists " + DATABASE_ESTATETABLE);
db.execSQL("drop table if exists " + DATABASE_PERSONFARMTABLE);
onCreate(db);

}
}//end of DBHelper class

public MapsDB(Context c) {
ourContext = c;
}

public MapsDB open() throws SQLException {
ourHelper = new DBHelper(ourContext);
ourDatabase = ourHelper.getWritableDatabase();
return this;
}

public void close() {
ourHelper.close();

}

……..
```

Εδώ μπαίνουν όλες οι συναρτήσεις με τα ερωτήματα sql που χρησιμοποιούμε στην εφαρμογή μας. Για παράδειγμα

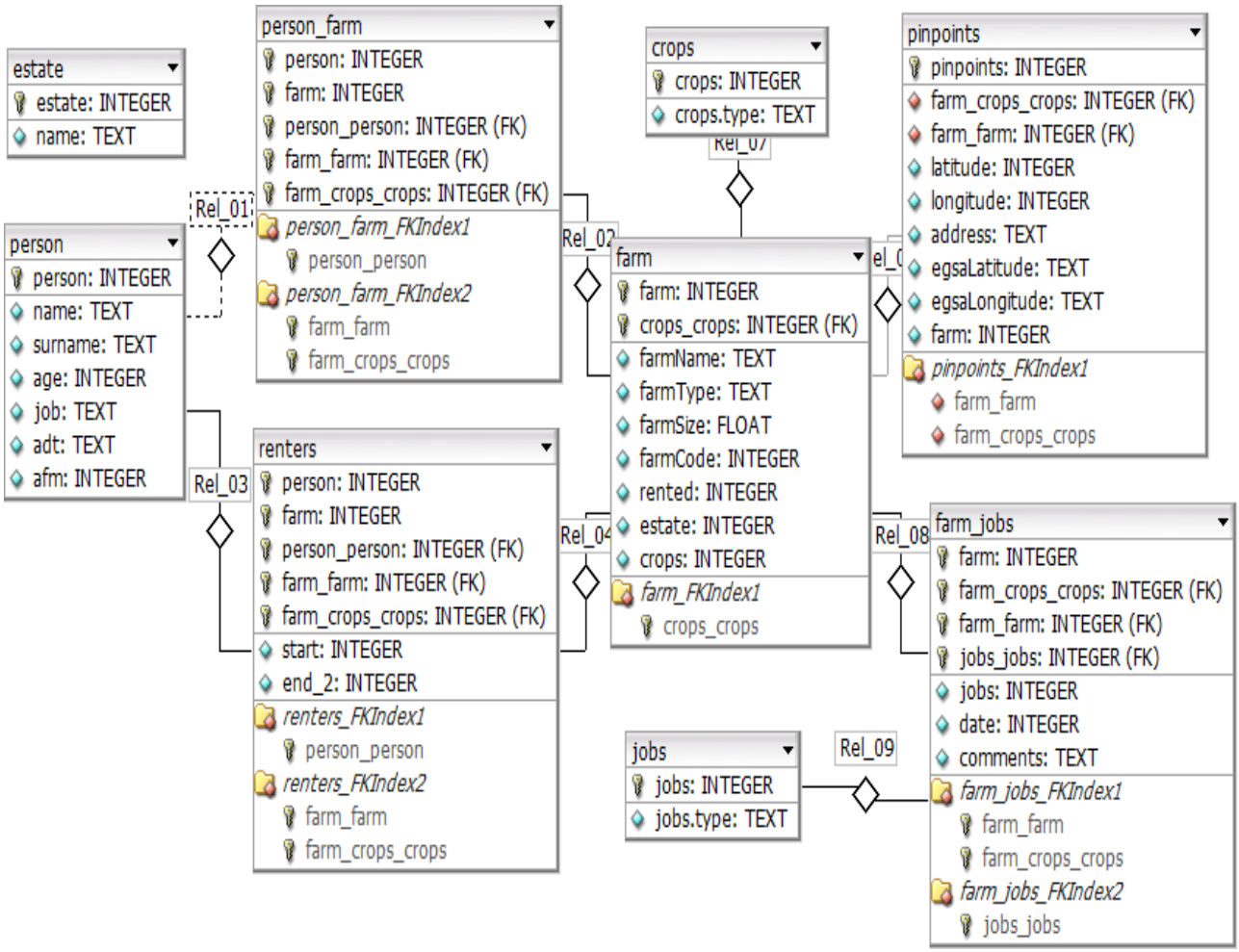
```
public void createPerson(String name, String surname, String age, String adt, String afm, String job){
ContentValues cv = new ContentValues();
```

```
cv.put(KEY_NAME, name);
cv.put(KEY_SURNAME, surname);
cv.put(KEY_AGE, age);
cv.put(KEY_JOB, job);
cv.put(KEY_ADT, adt);
cv.put(KEY_AFM, afm);
ourDatabase.insertWithOnConflict(DATABASE_PERSONTABLE, null, cv, DATABASE_VERSION);
}
……..
}//end of MapsDB class
```

Η κλάση MapsDB είναι η γενική κλάση η οποία περιέχει μέσα όλα όσα αφορούν την διαχείριση της βάσης. Αρχικά δηλώνουμε σε σταθερές μεταβλητές όλα τα πεδία και τους πίνακες της βάσης μας. Αυτό το κάνουμε για έναν απλό λόγω. Για να μπορούμε αν αποφασίσουμε να αλλάξουμε την ονομασία ενός πεδίου να μπορούμε να το αλλάξουμε μόνο μέσα στην μεταβλητή αυτή και να μην ψάχνουμε να βρούμε που χρησιμοποιούμε το πεδίο αυτό μέσα στον κώδικα και να το αλλάξουμε. Θα χρειαστούμε και μια βοηθητική κλάση που κληρονομεί το `SQLiteOpenHelper`. Η κλάση αυτή είναι υπεύθυνη για την δημιουργία της βάσης και την διαχείρισή της. Με την βοήθειά της μπορούμε να ανοίγουμε την σύνδεση με την βάση, εκτελούμε τα ερωτήματα που θέλουμε και να κλείνουμε την σύνδεση με την βάση. Είδαμε παραπάνω την συνάρτηση createPerson(….). Για να καλέσουμε την συνάρτηση αυτή από οποιοδήποτε σημείο της εφαρμογής μας και να δημιουργήσουμε μια εγγραφή στον πίνακα person της βάσης μας απλά εκτελούμε τον παρακάτω κώδικα.

```
MapsDB db= new MapsDB(this);
db.open();
db.createPerson(…);
db.close();
```

Προσοχή πρέπει να δοθεί στην διαχείριση των ανοιχτών συνδέσεων στην βάση. Δεν πρέπει να ανοίγουμε άλλη σύνδεση αν υπάρχει ήδη κάποια ανοιχτή διότι θα βγάζει πρόβλημα στο πρόγραμμα και δεν θα δουλεύει. Θα πρέπει λοιπόν να προσέχουμε πότε ανοίγουμε την σύνδεση και πότε την κλείνουμε.



Διάγραμμα E-R της βάσης δεδομένων που χρησιμοποιείται στην εφαρμογή

5.7 Χρήση των Google Maps στην εφαρμογή

Ο FarmManager βασίζεται στους online χάρτες της google για την χαρτογράφηση των χωραφιών του αγροκτήματος μας. Χρησιμοποιεί τους χάρτες για να βρει τις συντεταγμένες των χωραφιών και να τα εμφανίζει πάνω στον χάρτη. Παρακάτω θα δούμε όλη τη διαδικασία από την αρχή ώστε να μπορέσουμε να χρησιμοποιήσουμε τους χάρτες για εισαγωγή και προβολή των χωραφιών.
Αρχικά δημιουργούμε το xml της οθόνης με τον χάρτη που φαίνεται παρακάτω.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"   android:layout_height="fill_parent"
   android:orientation="vertical" >
   <TextView
      android:id="@+id/tvMap"       android:visibility="gone"
      android:layout_width="wrap_content"       android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
```

```
      android:text="Για να εισάγετε νέο χωράφι στο αγρόκτημά σας, μεταφερθείτε στην περιοχή που
θέλετε και πιέστε παρατεταμένα στο χωράφι που θέλετε" />
   <com.google.android.maps.MapView
      android:id="@+id/mvMap"        android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:apiKey="0BiGDCXauNVBC_JMuMmpnEBor2FYOXO_kdssHpA"
      android:clickable="true"        android:layout_above="@+id/tvMap"
      android:state_enabled="true" />
</RelativeLayout>
```

Εδώ δηλώνουμε και το κλειδί που έχουμε πάρει από το site της google για να μπορέσει να λειτουργήσει ο χάρτης. Στη συνέχεια δημιουργούμε την κλάση όπου θα γίνεται η επεξεργασία του χάρτη και των χωραφιών.

```
public class Map extends MapActivity implements
LocationListener,GestureDetector.OnGestureListener, GestureDetector.OnDoubleTapListener,
OnLongClickListener {

public MapView map;
List<Overlay> overLayList;
Drawable d, md;
long start;
long stop;
MyLocationOverlay myLocationOverlay;
MapController controller;
int x, y;
GeoPoint touchedPoint;
boolean finish=false;
LocationManager lm;
String towers;
int lat = 0;
double pinLat = 0;
int longi = 0;
double pinLongi = 0;
CustomPinPoint myPin;
double x1,y1,x2,y2;
float di;
float[] results=new float[10];
GestureDetector gestureDetector = null;
int pressedFlag=0,pinsFlag=0;
TextView tvMap;
GeoPoint touchedPointForInsert;
int gotLat,gotLongi;
String farmName="", cropName="";
List<List<String>> enabledFarms;
List<String> enabledFarmsId = new ArrayList<String>();
List<List<Double>> coordinates;
// ----------------------------------------------------------------------
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.map);
SharedPreferences prefs = Map.this.getSharedPreferences("Settings",0);
String year=prefs.getString("currentYear", "");
MapsDB db1=new MapsDB(this);
db1.open();
String name=db1.estateName();
db1.close();
String title="Κτήμα: " + name + " - Έτος: " + year;
this.setTitle(title);
```

```
SharedPreferences from = Map.this.getSharedPreferences("Misc",0);
String temp2=from.getString("mapFrom", "");
if(temp2.equalsIgnoreCase("startMenu")){
pressedFlag=1;
tvMap=(TextView) findViewById(R.id.tvMap);
tvMap.setVisibility(View.VISIBLE);
}

SharedPreferences.Editor editor = Map.this.getSharedPreferences(
"Misc", 0).edit();
editor.putBoolean("finish", false);
editor.putString("mapFrom", "");
editor.commit();
map = (MapView) findViewById(R.id.mvMap);
map.setBuiltInZoomControls(true);
map.setHapticFeedbackEnabled(true);
map.setOnLongClickListener(this);
map.setClickable(true);
map.setEnabled(true);
overLayList = map.getOverlays();
gestureDetector = new GestureDetector(this);
gestureDetector.setOnDoubleTapListener(this);
myLocationOverlay = new MyLocationOverlay(Map.this, map);
overLayList.add(myLocationOverlay);
myLocationOverlay.enableCompass();
controller = map.getController();
map.setSatellite(true);
d = getResources().getDrawable(R.drawable.ic_launcher);
md = getResources().getDrawable(R.drawable.pin1);
myPin = new CustomPinPoint(md, map);
// Displaying all the pinPoints from the database
try {
MapsDB db = new MapsDB(Map.this);
db.open();
enabledFarms=db.fetchEnabledFarms();
db.close();

for (int i = 0; i < enabledFarms.size(); i++){
enabledFarmsId.add(enabledFarms.get(i).get(0));
}
db.open();
coordinates=db.fetchEnabledFarmsCoordinates(enabledFarmsId);
db.close();
if(coordinates.size()!=0){
overLayList.add(myPin);
pinsFlag=1;
for (int i = 0; i < coordinates.size(); i++) {
pinLat =  coordinates.get(i).get(0);
pinLongi =  coordinates.get(i).get(1);
GeoPoint ourLocation = new GeoPoint((int)pinLat, (int)pinLongi);
String temp="";
if(enabledFarms.get(i).get(2).equalsIgnoreCase("1")){
temp="Νοικιασμένο";
}
else{
temp="Ιδιόκτητο";
}
OverlayItem overlayItem = new OverlayItem(ourLocation, enabledFarms.get(i).get(1)
+ " - " + temp,
enabledFarms.get(i).get(3));
```

```
myPin.insertPinPoint(overlayItem);
}
}
} catch (Exception e) {
e.printStackTrace();
}
}

public boolean onLongClick(View arg0) {
return false;
// TODO Auto-generated method stub
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
MenuInflater menuInflater=getMenuInflater();
menuInflater.inflate(R.menu.my_menu, menu);
return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
switch(item.getItemId()){
case R.id.toggleView:
if (map.isSatellite()) {
map.setSatellite(false);
map.setStreetView(true);
} else {
map.setStreetView(false);
map.setSatellite(true);
}
break;
case R.id.exit:
finish();
break;
}
return false;
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
super.onActivityResult(requestCode, resultCode, data);
if(resultCode==RESULT_OK){
if(requestCode==1){ //requestCode=1 drives to PinpointsOnlyFarmName.java
SharedPreferences prefs = Map.this.getSharedPreferences("Misc",0);
boolean enabled =prefs.getBoolean("farmEnabled", false);
if(enabled){
Bundle gotData=data.getExtras();
String gotCrop=gotData.getString("gCrop");
String gotRented=gotData.getString("gRented");
String gotFarmName=gotData.getString("gFarmName");
String title="";
title+= gotFarmName+" - "+gotRented;
OverlayItem overlayItem = new OverlayItem(touchedPointForInsert, title, gotCrop);
myPin.insertPinPoint(overlayItem);
if(pinsFlag==0){
overLayList.add(myPin);
pinsFlag=1;
}
```

```
map.postInvalidate();
}
}
else if(requestCode==2){ //requestCode=2 drives to PinpointsDataFull.java
SharedPreferences prefs = Map.this.getSharedPreferences("Misc",0);
boolean enabled =prefs.getBoolean("farmEnabled", false);
if(enabled){
Bundle gotData=data.getExtras();
String gotCrop=gotData.getString("gCrop");
String gotRented=gotData.getString("gRented");
String gotFarmName=gotData.getString("gFarmName");
String title="";
title+= gotFarmName+" - "+gotRented;
OverlayItem overlayItem = new OverlayItem(touchedPointForInsert, title, gotCrop);
myPin.insertPinPoint(overlayItem);
if(pinsFlag==0){
overLayList.add(myPin);
pinsFlag=1;
}
map.postInvalidate();
}
}
}
}

public void onLongPress(MotionEvent m) {
if(pressedFlag==1){
x=(int) m.getX();
y=(int) m.getY();
AlertDialog alert = new AlertDialog.Builder(Map.this).create();
alert.setTitle("Μενού συμπλήρωσης στοιχείων ");
alert.setMessage("Επιλογή");
alert.setButton("Πλήρης συμπλήρωση", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
SharedPreferences.Editor editor = Map.this.getSharedPreferences(
"Settings", 0).edit();
editor.putString("from", "full");
editor.commit();
touchedPointForInsert = map.getProjection().fromPixels(x, y);
String bAddress=findAddressFromCoordinates(touchedPointForInsert);
Bundle b=new Bundle();
b.putInt("bLat", touchedPointForInsert.getLatitudeE6());
b.putInt("bLongi", touchedPointForInsert.getLongitudeE6());
b.putString("bAddress", bAddress);
Intent in = new Intent(Map.this,FillFarmDataActivity.class);
in.putExtras(b);
SharedPreferences.Editor editor4 = Map.this.getSharedPreferences(
"Misc", 0).edit();
editor4.putBoolean("finish", false);
editor4.commit();
startActivityForResult(in, 2);
pressedFlag=0;
}
});
alert.setButton2("Μόνο ονομασία χωραφιού", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub
SharedPreferences.Editor editor = Map.this.getSharedPreferences(
"Settings", 0).edit();
editor.putString("from", "onlyName");
```

```
editor.commit();
touchedPointForInsert = map.getProjection().fromPixels(x, y);
String bAddress=findAddressFromCoordinates(touchedPointForInsert);
Bundle b=new Bundle();
b.putInt("bLat", touchedPointForInsert.getLatitudeE6());
b.putInt("bLongi", touchedPointForInsert.getLongitudeE6());
b.putString("bAddress", bAddress);
Intent in = new Intent(Map.this,FillFarmDataActivity.class);
in.putExtras(b);
SharedPreferences.Editor editor5 = Map.this.getSharedPreferences(
"Misc", 0).edit();
editor5.putBoolean("finish", false);
editor5.commit();
startActivityForResult(in, 1);
pressedFlag=0;
}
});
alert.show();
}
}

public String findAddressFromCoordinates(GeoPoint g){
String addr="";
Geocoder geoCoder= new Geocoder(this, Locale.getDefault());
try{
List<Address>address=geoCoder.getFromLocation(g.getLatitudeE6()/1E6 , g.getLongitudeE6()/1E6,
1);
if(address.size()>0){
for(int i=0; i<address.get(0).getMaxAddressLineIndex(); i++){
addr +=address.get(0).getAddressLine(i)+" , ";
}
}
} catch (IOException e) {
e.printStackTrace();
}
return addr;
}

@Override
protected void onPause() {
myLocationOverlay.disableCompass();
myLocationOverlay.disableMyLocation();
SharedPreferences prefs = Map.this.getSharedPreferences("Misc",0);
finish=prefs.getBoolean("finish", false);
if(finish){
finish();
}
super.onPause();
}

@Override
protected void onResume() {
myLocationOverlay.enableCompass();// !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!enable
// when exporting the app
super.onResume();
}

public boolean onDoubleTap(MotionEvent me) {
int x=(int) me.getX();
int y=(int) me.getY();
```

```
GeoPoint p = map.getProjection().fromPixels(x, y);
controller.animateTo(p);
controller.zoomIn();
return true;
}

@Override
public boolean dispatchTouchEvent(MotionEvent ev) {
gestureDetector.onTouchEvent(ev);
return super.dispatchTouchEvent(ev);
}
}
```

Με το ίδιο τρόπο που δείξαμε στις άλλες οθόνες της εφαρμογής φορτώνουμε το xml του χάρτη. Ορίζουμε ένα αντικείμενο MapView το οποίο είναι υπεύθυνο για την διαχείριση του χάρτη. Ο χάρτης για να μπορεί να εμφανίζει πάνω του τα σημαιάκια των χωραφιών χρειάζεται ένα List<Overlay> overLayList το οποίο συνδέουμε με το αντικείμενο MapView. Για το κάθε χωράφι έχουμε κάνει μια δικιά μας κλάση ώστε να κρατάμε τα στοιχεία που θέλουμε και φαίνεται παρακάτω.

```
public class CustomPinPoint extends BalloonItemizedOverlay<OverlayItem> {
private ArrayList<OverlayItem> pinPoints = new ArrayList<OverlayItem>();
private Context c;
public CustomPinPoint(Drawable defaultMarker, MapView mapView) {
super(boundCenter(defaultMarker), mapView);
c = mapView.getContext();
}

@Override
protected OverlayItem createItem(int i) {
// TODO Auto-generated method stub
return pinPoints.get(i);
}

@Override
public int size() {
// TODO Auto-generated method stub
return pinPoints.size();
}

public void insertPinPoint(OverlayItem overlayItem) {
// TODO Auto-generated method stub
pinPoints.add(overlayItem);
populate();

}

@Override
protected boolean onBalloonTap(int index, OverlayItem item) {
Bundle b=new Bundle();
String[] farmName=pinPoints.get(index).getTitle().split(" - ");
b.putString("bFarmName", farmName[0]);
Intent i=new Intent(c,EditDetailsFromBalloonClick.class);
i.putExtras(b);
i.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
SharedPreferences.Editor editor5 = c.getSharedPreferences(
```

```
"Misc", 0).edit();
editor5.putBoolean("finish", true);
editor5.commit();
c.startActivity(i);
return true;
}

}
```

Η κλάση αυτή κρατάει τη λίστα με τα χωράφια που θέλουμε να εισάγουμε στον χάρτη. Για κάθε νέο χωράφι που θα εισάγουμε καλούμε από την κυρίως κλάση του χάρτη την συνάρτηση insertPinPoint(…). Ως όρισμα στη συνάρτηση δίνουμε ένα αντικείμενο τύπου OverlayItem το οποίο περιέχει τη θέση του χωραφιού και μια ονομασία του. Το αντικείμενο τύπου CustomPinPoint προστίθεται στο overlayList του χάρτη και έτσι μπορούμε να εμφανίζουμε τα σημαιάκια πάνω στον χάρτη. Οι υπόλοιπες συναρτήσεις που φαίνονται στην κλάση του χάρτη είναι για την διαχείριση των χωραφιών στην βάση. Για τη λειτουργία του χάρτη χρειάζονται μόνο αυτά που είπαμε παραπάνω.

Κάνοντας κλικ πάνω σε ένα σημαιάκι του χάρτη ανοίγει ένα «μπαλόνι» και δείχνει κάποια στοιχεία που εμείς του έχουμε βάλει. Για να γίνει αυτό χρησιμοποιήθηκε μια έτοιμη βιβλιοθήκη την κλάση της οποίας κληρονομεί η κλάση CustomPinPoint. Η βιβλιοθήκη αυτή εμφανίζει το «μπαλόνι» όταν κάνουμε κλικ σε μια σημαία και κάνοντας κλικ στο «μπαλόνι» μπορούμε να μεταφερθούμε σε μια νέα οθόνη.

5.8 Μετατροπή ενός activity σε dialog με αλλαγή του style

Μπορούμε πολύ εύκολα μια οθόνη να την μετατρέψουμε σε παράθυρο διαλόγου πολύ απλά αλλάζοντας το style της στο manifest αρχείο της εφαρμογής μας όπως φαίνεται παρακάτω.

```
<activity
        android:name=".EnableFarms"          android:label="@string/app_name"
        android:screenOrientation="portrait"          android:theme="@android:style/Theme.Dialog"
        android:windowSoftInputMode="stateHidden" >
        <intent-filter>
           <action android:name="com.georkouk.farmManager.ENABLEFARMS" />
           <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
      </activity>
```

Το αποτέλεσμα αυτού φαίνεται στην εικόνα 5.8.

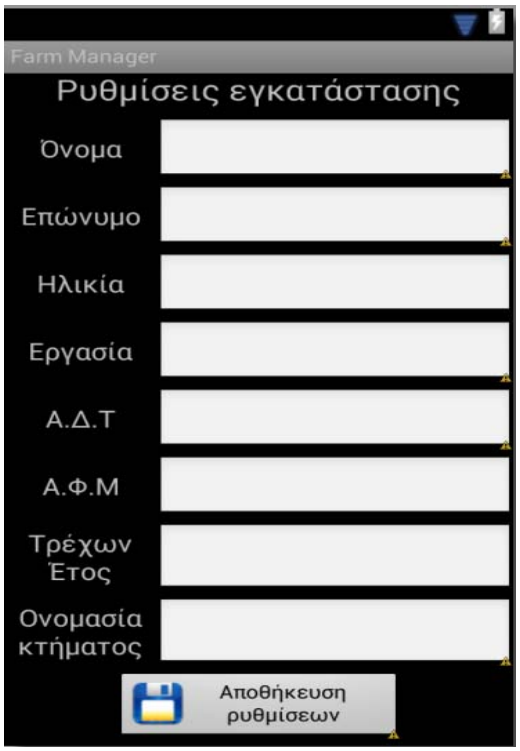
ΚΕΦΑΛΑΙΟ 6
Οδηγίες εγκατάστασης και χρήσης της εφαρμογής

6.1 Οδηγίες εγκατάστασης

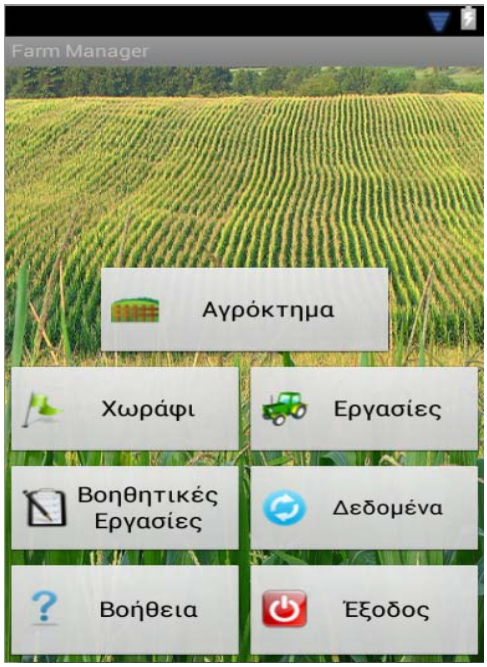
Η συγκεκριμένη εφαρμογή υποστηρίζει συσκευές με λειτουργικό σύστημα Android 2.2 και μεγαλύτερο. Το πρώτο πράγμα που θα χρειαστεί να κάνουμε είναι να επιτρέψουμε στη συσκευή μας να εγκαταστήσει μια εφαρμογή που δεν έχουμε κατεβάσει από το Android Market. Υπάρχει ειδική επιλογή από το λογισμικό για αυτή τη δουλεία και εμείς πρέπει να την ενεργοποιήσουμε. Για να το κάνουμε αυτό πηγαίνουμε στις ρυθμίσεις και από εκεί επιλέγουμε "Εφαρμογές". Στη συνέχεια θα εμφανιστεί μια οθόνη με διάφορες επιλογές, στην κορυφή των οποίων θα υπάρχει η επιλογή "Άγνωστες Πηγές" .Ελέγχουμε ότι είναι επιλεγμένη για να μας επιτρέπεται να εγκαταστήσουμε εφαρμογές εκτός του Android Market. Στη συνέχεια μεταφέρουμε το apk αρχείο της εφαρμογής στην SD card του κινητού μας. Η διαδικασία είναι η ίδια που θα κάναμε για να μεταφέρουμε ένα οποιοδήποτε αρχείο μουσικής ή video.
Αφού κάναμε όλα τα παραπάνω το μόνο που απομένει είναι να εγκαταστήσουμε έναν File Manager μέσω του οποίου θα εγκαταστήσουμε την εφαρμογή μας. Στο Android Market θα βρούμε αρκετούς, όπως τον ASTRO File Manager ή τον ES File Explorer. Αφού εγκαταστήσουμε όποιον μας αρέσει, τον τρέχουμε και στη συνέχεια θα μας εμφανιστεί μία οθόνη αντίστοιχη του explorer των windows που βλέπουμε τα αρχεία μας. Το μόνο που απομένει να κάνουμε είναι να βρούμε την εφαρμογή μας, να την επιλέξουμε και στη συνέχεια να επιλέξουμε εγκατάσταση από την επιλογή που θα μας εμφανιστεί. Η εφαρμογή βρίσκεται πλέον εγκατεστημένη στη συσκευή μας. Για να δείτε πως θα περάσετε μια έτοιμη βάση μέσα στην εφαρμογή μεταφερθείτε στο τέλος του οδηγού χρήσης.

6.2 Λειτουργία εφαρμογής

Την πρώτη φορά που θα ανοίξουμε την εφαρμογή θα χρειαστεί να συμπληρώσουμε τα στοιχεία που φαίνονται στην παρακάτω εικόνα. Τα στοιχεία αυτά αποθηκεύονται και δεν θα ξαναζητηθούν.

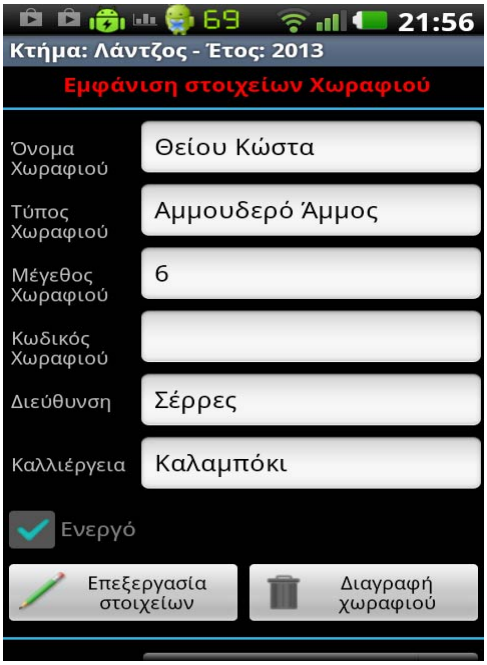


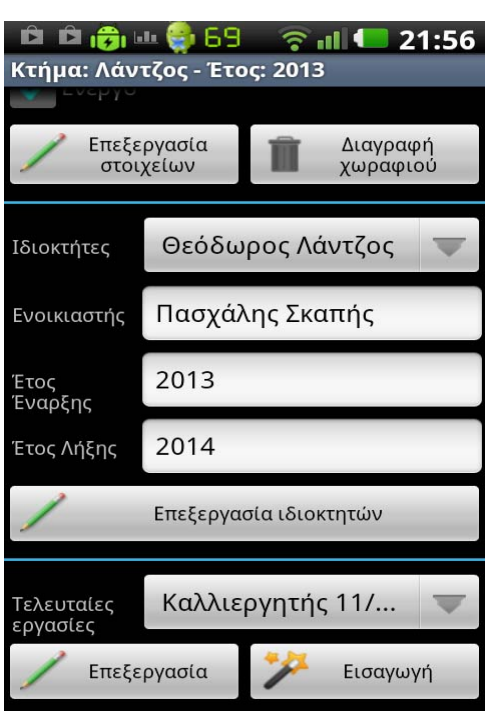
Αφού συμπληρώσουμε τα παραπάνω στοιχεία μεταφερόμαστε στο **κεντρικό μενού** της εφαρμογής

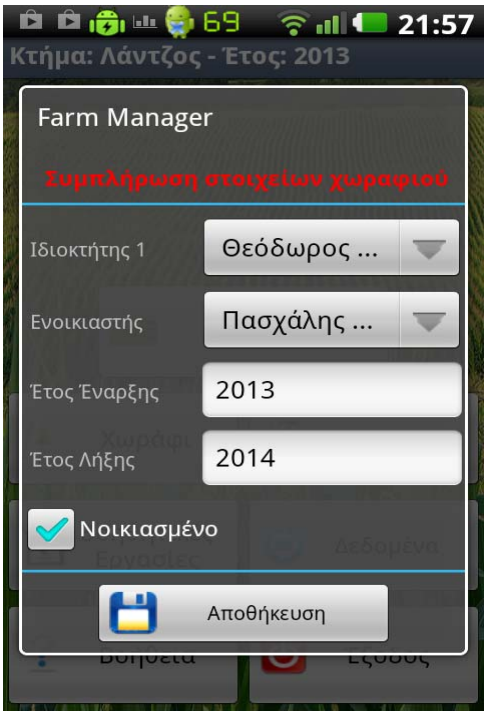


Πατώντας στον τίτλο του παραθύρου μπορούμε να δούμε κάποια **στατιστικά** για το αγρόκτημά μας όπως φαίνεται στην παρακάτω εικόνα.

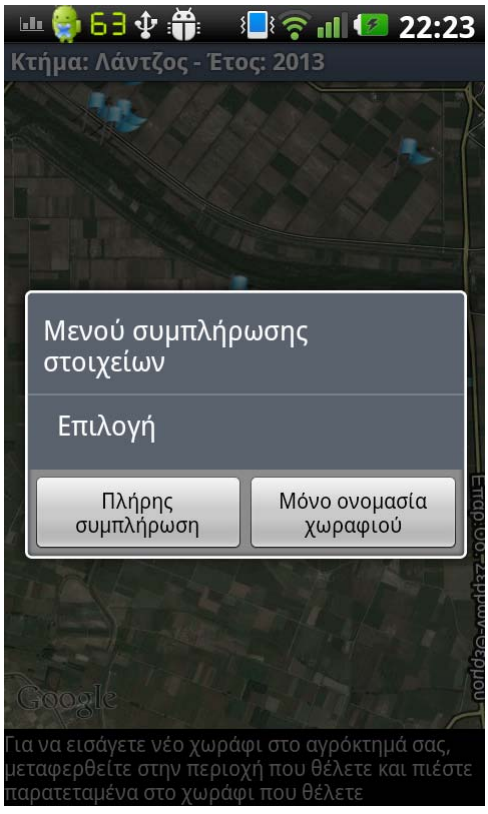
Από το κεντρικό μενού επιλέγοντας **αγρόκτημα** ανοίγει ο χάρτης στον οποίο μπορούμε να δούμε τα χωράφια που έχουμε δηλώσει στην εφαρμογή. Πατώντας πάνω σε ένα χωράφι ανοίγει ένα παραθυράκι με κάποια στοιχεία του χωραφιού. Εκεί κάνοντας κλικ στο παραθυράκι μπορούμε να δούμε και να επεξεργαστούμε όλα τα στοιχεία του χωραφιού που έχουμε επιλέξει.

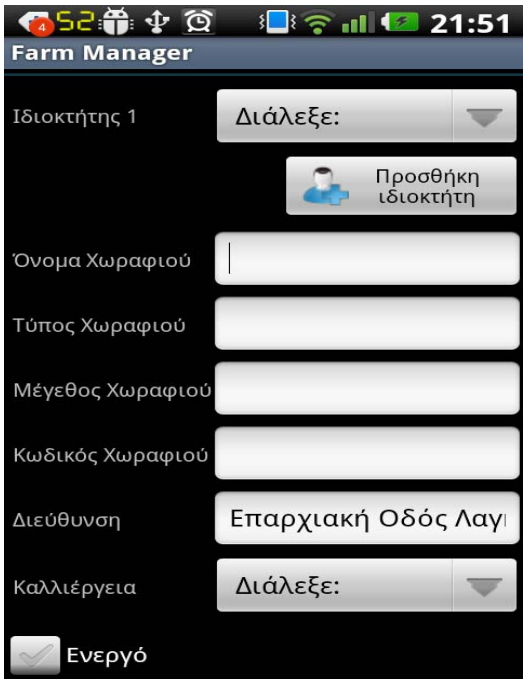




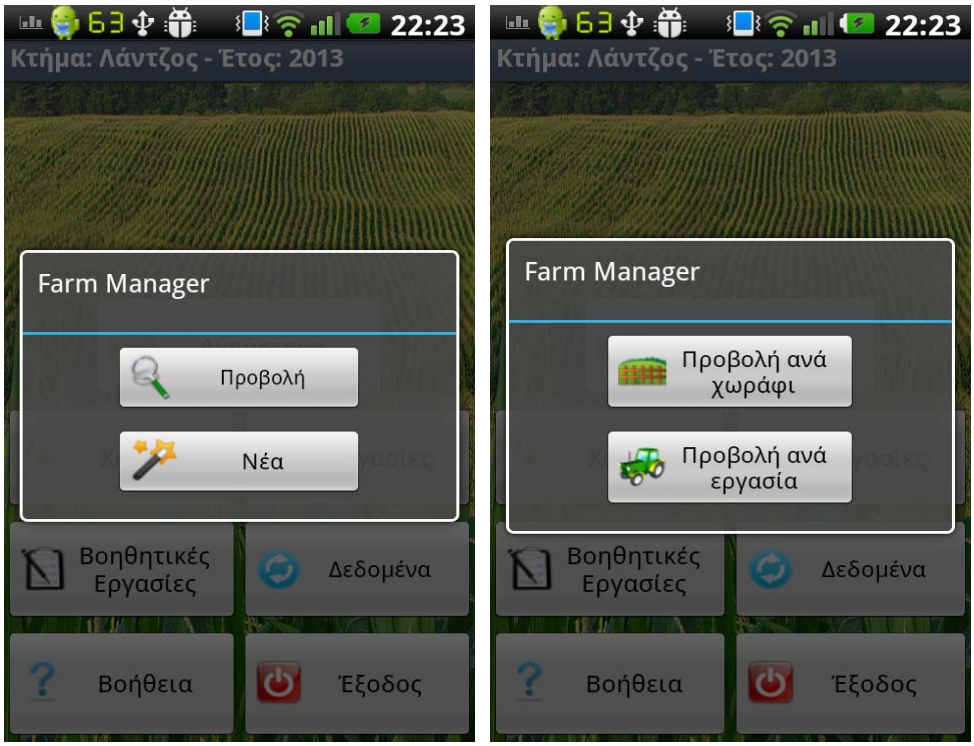


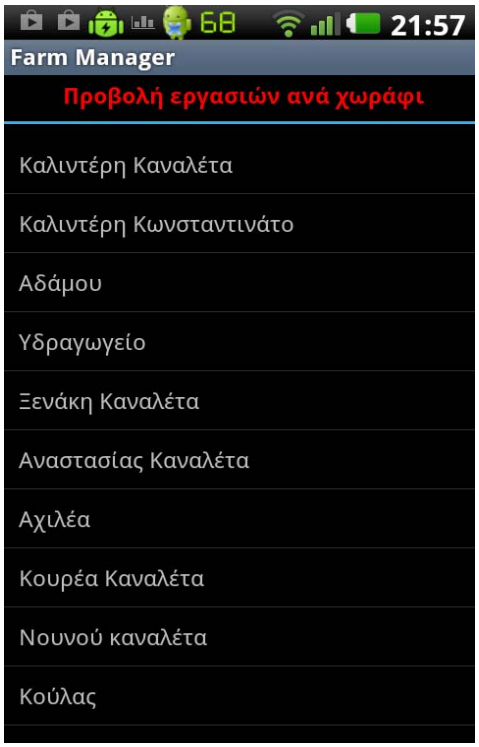
Στο κεντρικό μενού πατώντας στο κουμπί **χωράφι** μπορούμε να εισάγουμε ένα νέο χωράφι στο αγρόκτημά μας. Απλά μεταφερόμαστε στην περιοχή που θέλουμε, βρίσκουμε το χωράφι που θέλουμε και πιέζοντας παρατεταμένα εμφανίζεται το μενού για τη συμπλήρωση των στοιχείων του χωραφιού.

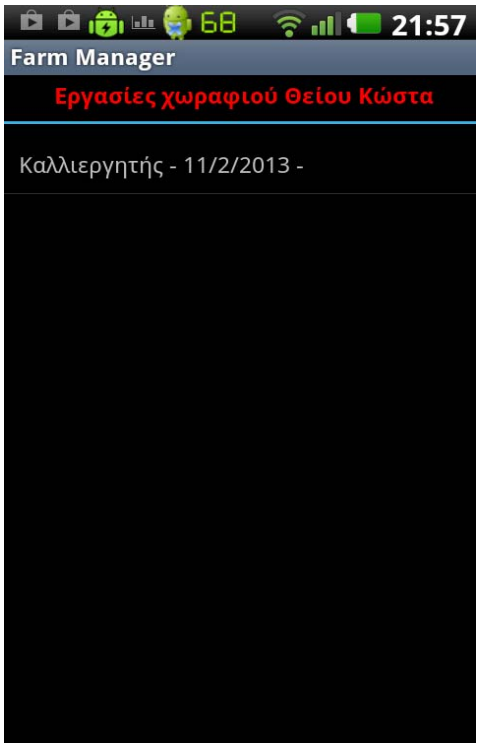


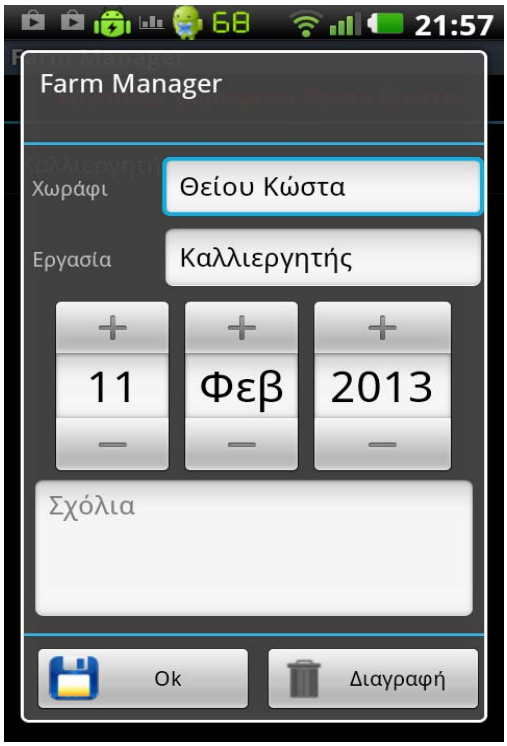


Εκεί αφού συμπληρώσουμε τα απαραίτητα στοιχεία μεταφερόμαστε στον χάρτη του αγροκτήματος οπού μπορούμε να δούμε και το νέο χωράφι που μόλις έχουμε εισάγει. Στο μενού **εργασίες** μπορούμε είτε να εισάγουμε νέα εργασία σε κάποιο χωράφι είτε να δούμε τις ήδη υπάρχουσες εργασίες στα χωράφια μας.

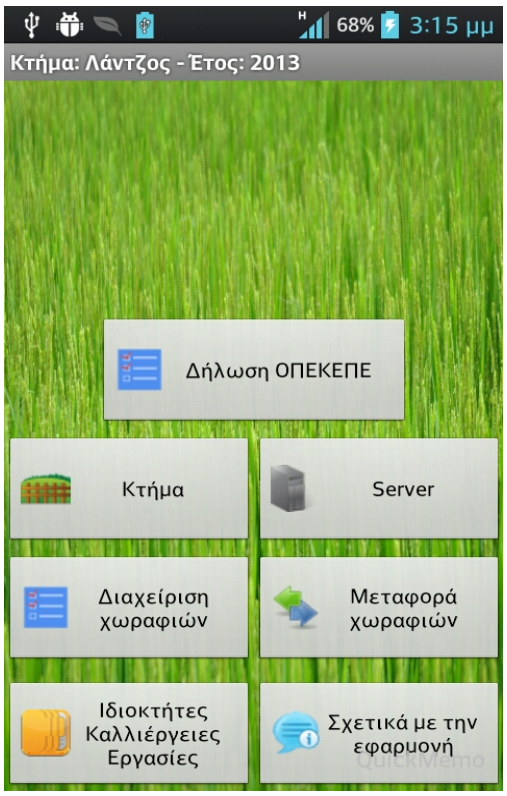


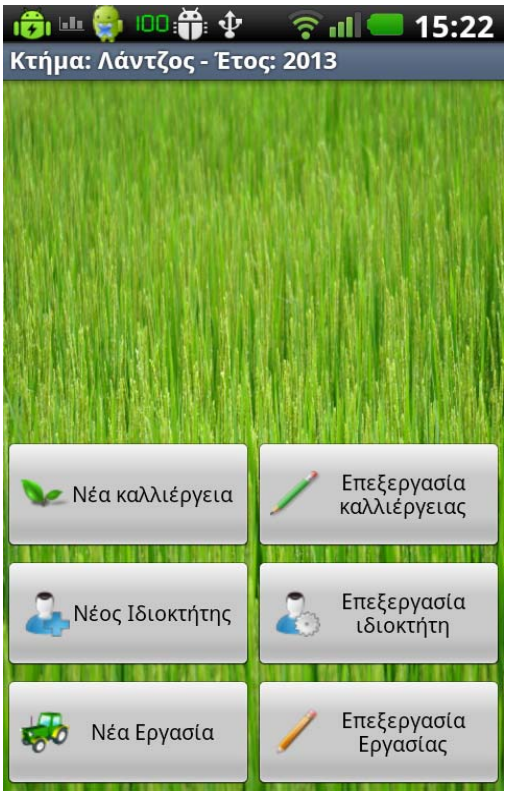


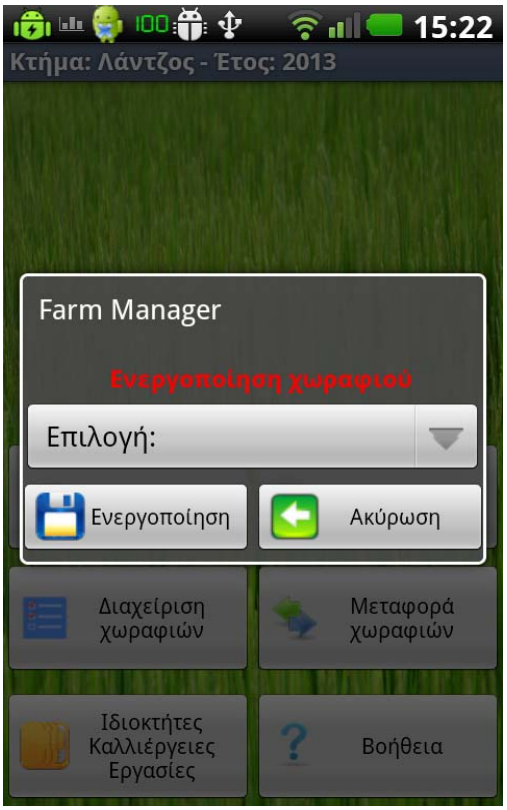




Στις **<u>βοηθητικές εργασίες</u>** μπορούμε να ρυθμίσουμε διάφορα πράγματα για την εφαρμογή μας όπως στοιχεία του κτήματος, να εισάγουμε και να επεξεργαστούμε ιδιοκτήτες, καλλιέργειες και εργασίες στην βάση, να διαχειριστούμε να ανενεργά χωράφια που έχουμε δηλώσει, να κάνουμε μεταφορά των χωραφιών στο νέο έτος ή να δούμε τη βοήθεια για την εφαρμογή.

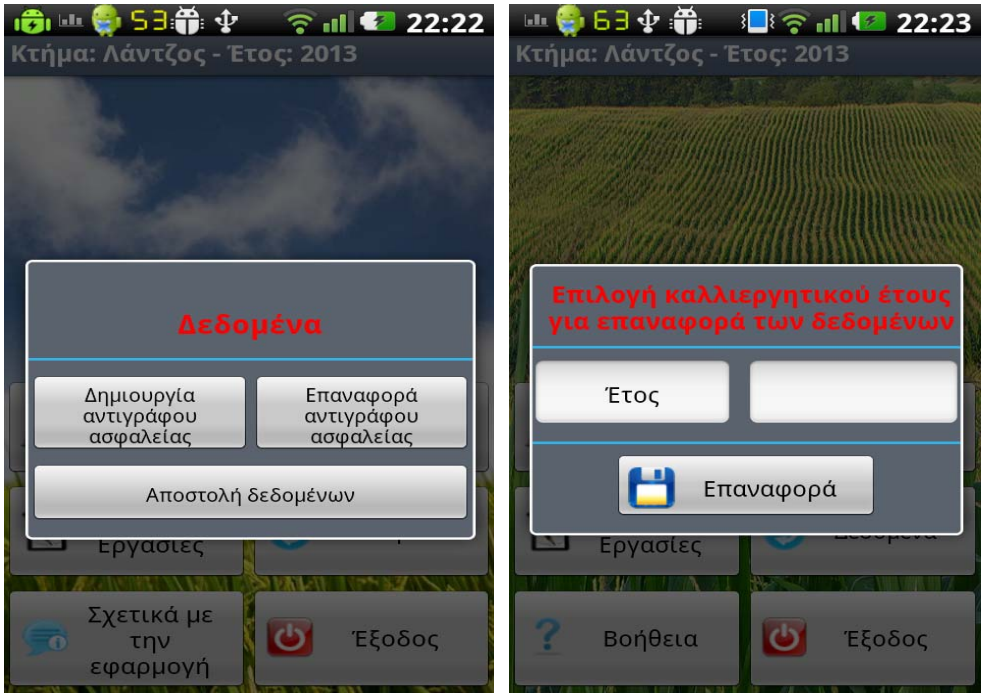






Στα **δεδομένα** μπορούμε ανά τακτά χρονικά διαστήματα να παίρνουμε backup των δεδομένων της εφαρμογής μας, να τα επαναφέρουμε ξανά πίσω στην εφαρμογή όποτε

θέλουμε ή μπορούμε να στείλουμε τα δεδομένα σε κάποιον server . Τα αντίγραφα ασφαλείας κρατούνται στην sd card του κινητού σε φακέλους ανάλογα με το έτος στο οποίο έχουμε δηλώσει. Τα αντίγραφα αυτά είναι πολύ μικρά σε μέγεθος (~1mb) και λιγότερο οπότε μπορούμε άνετα να κρατάμε αντίγραφα πολλών ετών.



6.3 Μεταφορά έτοιμης βάσης στην εφαρμογή

Εάν έχουμε έτοιμη κάποια βάση και θέλουμε να την περάσουμε εύκολα μέσα στο πρόγραμμα, χρησιμοποιούμε τον file manager που επιλέξαμε στην αρχή για να εγκαταστήσουμε την εφαρμογή. Βρίσκουμε τον φάκελο FarmManager που υπάρχει στην sd card(αν δεν υπάρχει τον δημιουργούμε) και εκεί μέσα φτιάχνουμε έναν υποφάκελο με όνομα το έτος χρήσης που θέλουμε. Στη συνέχεια μέσα στον φάκελο βάζουμε την βάση που θέλουμε. Αφού κάνουμε αυτά ανοίγουμε την εφαρμογή και πηγαίνουμε στα δεδομένα και μετά επιλέγουμε επαναφορά. Θα μας ζητήσει να εισάγουμε έτος. Βάζουμε το έτος το οποίο είχαμε ονομάσει τον φάκελο πριν και επιλέγουμε επαναφορά. Αν όλα έχουν γίνει σωστά θα μπορούμε να διαχειριστούμε τα στοιχεία της βάσης που μόλις έχουμε εισάγει.

6.4 Μεταφορά χωραφιών σε νέο καλλιεργητικό έτος

Εάν θέλουμε να μεταφέρουμε τα χωράφια του αγροκτήματός μας σε νέο έτος τότε απλά πηγαίνουμε από το κεντρικό μενού στις βοηθητικές εργασίες και επιλέγουμε **μεταφορά χωραφιών**. Εκεί μας εμφανίζεται μια λίστα με όλα τα δηλωμένα χωράφια για το τρέχον έτος. Επιλέγουμε όλα τα χωράφια που θέλουμε να μεταφέρουμε και πατάμε OK.

ΠΡΟΣΟΧΗ: Το πρόγραμμα πρώτα δημιουργεί ένα αντίγραφο ασφαλείας των δεδομένων του τρέχοντος έτους καλλιέργειας και στη συνέχεια κάνει τη μεταφορά. Γι' αυτό το λόγο είναι σημαντικό πρώτα να κάνουμε την μεταφορά των χωραφιών και στην συνέχεια να αλλάξουμε το έτος

6.5 Ορολογία

Αγρόκτημα: Το σύνολο των χωραφιών που έχουμε.
Εργασία: Μια ενέργεια που κάνουμε στο χωράφι. (πχ. Σπορά).
Καλλιέργεια: Αυτό που σπέρνουμε σε ένα χωράφι (πχ. Βαμβάκι).
Ιδιοκτήτης: Αυτός στον οποίο ανήκει ένα χωράφι.

Ενοικιαστής: Αυτός ο οποίος χρησιμοποιεί κάποιο από τα χωράφια μας για ένα διάστημα.

Καλλιεργητικό έτος: Το έτος για το οποίο χρησιμοποιούμε το αγρόκτημα.

Αντίγραφα ασφαλείας: Η αντιγραφή των δεδομένων της εφαρμογής σε άλλο μέρος ώστε σε περίπτωση κάποιου προβλήματος να μπορούμε να τα επαναφέρουμε.

ΚΕΦΑΛΑΙΟ 7
Συμπεράσματα και μελλοντική εξέλιξη

Η εφαρμογή FarmManager δημιουργήθηκε στα πλαίσια πτυχιακής εργασίας του τμήματος Πληροφορικής και Επικοινωνιών του ΤΕΙ Σερρών με σκοπό την εκμάθηση προγραμματισμού σε Android συσκευές, καθώς επίσης και τη δημιουργία μιας πρωτοποριακής εφαρμογής για τον αγρότη που όμοιά της δεν υπάρχει. Είδαμε ότι η ανάπτυξη εφαρμογής στο Android είναι μια ιδιαίτερα απαιτητική εργασία η οποία απαιτεί σημαντικές γνώσεις Java και ταλέντο στον προγραμματισμό, πάνω από όλα όμως χρειάζεται το μεράκι του προγραμματιστή να δημιουργήσει κάτι πραγματικά χρήσιμο. Το τελικό όμως αποτέλεσμα μας ανταμείβει.

Για την δημιουργία και τη δόμηση της εφαρμογής λάβαμε υπόψη πολλές παραμέτρους. Για παράδειγμα ότι η εφαρμογή θα μπορεί να χρησιμοποιηθεί και από μεγάλης ηλικίας ανθρώπους που δεν έχουν πολλές γνώσεις πάνω στην νέα τεχνολογία οπότε και σχεδιάσαμε την εφαρμογή με γνώμονα την ευκολία χρήσης. Όλες οι λειτουργίες της εφαρμογής είναι προσβάσιμες από κουμπιά για πιο εύκολη χρήση. Επίσης το πρόγραμμα είναι πολύ ελαφρύ σε απαιτήσεις και μπορεί να λειτουργήσει και σε κινητά της οικονομικής σειράς οπότε μειώνουμε σημαντικά το κόστος.

Το κυριότερο όμως είναι ότι η εφαρμογή είναι πλήρως εξελίξιμη. Μελλοντικά θα μπορούσε η εφαρμογή να αναπτυχθεί και να προστεθούν τα παρακάτω πράγματα:

- Διαχείριση ολοκληρωμένων δεδομένων

Τα δεδομένα που διαχειριζόμαστε από το farm manager την εφαρμογή του κάθε αγρότη να μπορούμε να τα ομαδοποιήσουμε σε ένα κεντρικό υπολογιστή για να εξαγωγή γνώσης και στατιστικών. Για παράδειγμα, από τα 30 αγροκτήματα του Νομού Σερρών, που έσπειραν καλαμπόκι και υβρίδιο Χ, τι αποτελέσματα είχε το 2012. Έτσι θα μπορούμε να έχουμε όλοι οι αγρότες πληροφορία που θα βασίζετε σε μεγάλο εύρος δεδομένων και όχι σε στατιστικά εταιρειών τα οποία είναι λίγο ή πολύ

τροποποιημένα. Το Χ υβρίδιο καλαμποκιού σε τι χωράφι καλλιεργήθηκε, με τι λίπανση και με τι άρδευση; Όπως καταλαβαίνετε τα δεδομένα του καθενός μας εάν συγκεντρωθούνε ηλεκτρονικά, κρατώντας το απόρρητο των προσωπικών δεδομένων μπορούνε να αποτελέσουνε μεγάλο μοχλό εξέλιξης για όλους μας και να δώσουνε μια καλύτερη σοδειά στις δύσκολες αυτές ημέρες. Δεν είναι οι εποχές που επιτρέπουνε πειραματισμούς και αποτυχίες όταν τα δεδομένα και η γνώση υπάρχει.

- Οδηγός φυτοπροστασίας

Στο πρόγραμμα αυτό έχουμε έναν οδηγό φυτοπροστασίας. Έχοντας ανεβάσει στο smartphone μια βάση δεδομένων από ασθένειες και εχθρούς μιας καλλιέργειας, μπορούμε να πάμε στο χωράφι μας και να κάνουμε επισκόπηση. Σε ότι συναντάμε που είναι εχθρός της καλλιέργειας μπορούμε να τραβήξουμε φωτογραφία και με τα δεδομένα της βάσης να βρούμε τι είναι αυτό που είδαμε. Έτσι μπορούμε να εντοπίσουμε ζιζάνια, έντομα που δεν γνωρίζουμε και απευθείας να μάθουμε από την βάση του smartphone τι είναι αυτά και με τι δραστική ουσία μπορούμε να τα αντιμετωπίσουμε χωρίς να πανικοβαλλόμαστε και να τρέχουμε στο γεωπόνο σε πρώτο στάδιο. Εάν είναι κάτι που δεν το έχουμε καταχωρημένο στην βάση μπορούμε να το προσθέσουμε και στην συνέχεια να υπάρχει έτσι ώστε να μας βοηθήσει σε μελλοντικές ανάγκες.

- Οδηγός αγοράς εφοδίων

Αυτή η εφαρμογή στηρίζεται στο να μπορούνε οι αγρότες και οι έμποροι να καταχωρούνε τιμές εφοδίων. Έτσι οι αγρότες από μια ιστοσελίδα ή από το smartphone να μπορούνε να κάνουνε πρακτικές ερωτήσεις και να εντοπίζουνε τις καλύτερες τιμές της αγοράς για εφόδια στην περιοχή τους ή ευρύτερα. Έτσι ο κάθε αγρότης θα μπορεί να έχει μια καλύτερη εκτίμηση του κόστους των εφοδίων του και θα μπορεί να συμβουλεύετε για τις παγίδες των εφοδίων όταν αυτές είναι πολύ χαμηλές αλλά δεν καλύπτουν τις προδιαγραφές ή απαιτήσεις.

- Διαχείριση εξοπλισμού

Εδώ είναι μια εφαρμογή σε όπου μπορούμε να κρατάμε απευθείας δεδομένα για τον εξοπλισμό που διαθέτουμε. Έτσι συντήρηση, επισκευές, επιδιορθώσεις, κόστος, νέες αποκτήσεις και ότι έχει να κάνει με τον πάγιο εξοπλισμό που διαθέτει ένας αγρότης να μπορεί να διαχειρίζεται ηλεκτρονικά και να υπάρχει αρχείο.

- Επαγγελματίας Αγρότης

Εδώ είναι μια εφαρμογή σε όπου μπορούμε να κρατάμε 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) να μπορεί θα καταγράφει απευθείας το είδος της εργασίας και το χωράφι που έλαβε μέρος. Έτσι διαχείριση λογαριασμών, έκδοση αποδείξεων και διατήρηση βιβλίων γίνεται παιχνίδι απευθείας από το μηχάνημα επάνω. Η εφαρμογή συντηρεί αρχείο παρελθόντων ετών.

- Εδαφολογικός χάρτης

Η εφαρμογή αυτή στηρίζεται σε smartphones και επιτρέπει εταιρείες και αγρότες να καταχωρούνε εδαφολογικές αναλύσεις. Απλά επάνω στον χάρτη σημαδεύεις το χωράφι και εισάγεις τα δεδομένα της εδαφολογικής ανάλυσης. Σε ατομικό επίπεδο η εφαρμογή αυτή εξυπηρετεί τον αγρότη στην επιλογή σωστών λιπάνσεων και σπόρων. Σε ευρύτερο όμως φάσμα εάν τα δεδομένα συγκεντρωθούνε μπορούμε να έχουμε εδαφολογικούς χάρτες περιοχών και να μπορούμε να επιλέγουμε σπορικά που συγκεντρωτικά ευδοκιμούν στην περιοχή. Επίσης, αναλύσεις εδάφους σε βάθος χρόνου μπορούνε να μας οδηγήσουνε σε παρακολούθηση και σύγκριση μεταβολών στο έδαφος από την χρήση χημικών λιπασμάτων, αρδεύσεων και γεωτρήσεων.

- Εφαρμογές τηλεπισκόπησης

Σε αυτές τις εφαρμογές έχουμε εξοπλισμό που καταγράφει υγρασία και αέρα όπου με smartphone μπορούμε άμεσα να πάρουμε δεδομένα από το χωράφι όπως για υγρασία και αέρα προτού πάμε εκεί και δούμε ότι ένας ψεκασμός είναι αδύνατος.

Βιβλιογραφία

- http://developer.android.com/guide/basics/what-is-android.html
- http://developer.android.com/guide/basics/what-is-android.html
- http://developer.android.com/guide/topics/manifest/manifest-intro.html
- http://developer.android.com/guide/developing/index.html
- http://developer.android.com/sdk/installing.html
- http://www.eclipse.org/
- http://developer.android.com/sdk/eclipse-adt.html#installing
- http://developer.android.com/design/index.html
- http://nemertes.lis.upatras.gr/jspui/bitstream/10889/4661/1/%CE%94%CE%99%CE%A0%CE%9B%CE%A9%CE%9C%CE%91%CE%A4%CE%99%CE%9A%CE%97%20%CE%95%CE%A1%CE%93%CE%91%CE%A3%CE%99%CE%91.pdf

- http://developer.android.com/resources/dashboard/platform-versions.html
- http://developer.android.com/resources/dashboard/screens.html
- http://developer.android.com/guide/practices/screens_support.html
- http://developer.android.com/guide/developing/tools/adb.html
- http://developer.android.com/guide/developing/devices/index.html
- http://developer.android.com/guide/developing/tools/logcat.html
- www.stackoverflow.com
- www.myphone.gr
- www.androidgreece.gr
- http://el.wikipedia.org/wiki/Android
- http://www.angroid.gr/%CF%84%CE%B9-%CE%B5%CE%AF%CE%BD%CE%B1%CE%B9-2/
- http://www.greekradar.gr/key-lime-pie-%CE%B7-%CE%B5%CF%80%CF%8C%CE%BC%CE%B5%CE%BD%CE%B7-android-%CE%AD%CE%BA%CE%B4%CE%BF%CF%83%CE%B7/
  http://www.kassapoglou.com/lesson-1-setting-up-an-android-development-environment/
- http://www.greeceandroid.gr/markets
- http://www.myphone.gr/forum/showthread.php?t=306146

Από το παρακάτω σύνδεσμο μπορείτε να κατεβάσετε την εφαρμογή FarmManager δωρεάν.

http://web2.teiser.gr/web-programming/farmManager/welcome.html