ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ
ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΛΟΠΟΙΗΣΗ ΕΦΑΡΜΟΓΗΣ ΚΙΝΗΤΟΥ
ΤΗΛΕΦΩΝΟΥ ΜΕ ΧΡΗΣΗ ΤΗΣ
ΠΛΑΤΦΟΡΜΑΣ ANDROID

Αποστολίδης Χρίστος (ΑΜ: 03/2407)

Επιβλέπον καθηγητής: Κεραμόπουλος Ευκλείδης

Περίληψη

Το παρόν κείμενο παρουσιάζει την εφαρμογή ReProTool σε πλατφόρμα Android.

Η όλη ιδέα βασίζεται στο ήδη υπάρχον web application ReProTool. Η εφαρμογή λαμβάνει την ώρα από το κινητό και υπολογίζει από τη τοπική βάση δεδομένων ποια είναι τα μαθήματα τα οποία ο φοιτητής είναι εγγεγραμμένος στο τρέχον τυπικό εξάμηνο. Στη συνέχεια τα εμφανίζει στην οθόνη του κινητού σε μορφή λίστας. Ο φοιτητής επιλέγει το μάθημα που επιθυμεί και έχει τη δυνατότητα να συμπληρώσει τις ώρες τις οποίες αφιερώνει για τη μελέτη του εντός και εκτός τάξης εβδομαδιαίως, όπως επίσης μπορεί να γράψει σημειώσεις για το κάθε μάθημα. Τα δεδομένα αυτά σώζονται σε βάση δεδομένων για την μετέπειτα επεξεργασία τους με σκοπό τη εξαγωγή στατιστικών στοιχείων τα οποία παρουσιάζονται με τη μορφή γραφήματος και τα οποία θα βοηθήσουν τους φοιτητές στη μελέτη τους.

Για την δημιουργία της εφαρμογής χρησιμοποιήθηκε το λογισμικό Eclipse το οποίο είναι ένα πρόγραμμα ανοιχτού κώδικα που σε συνεργασία με τα δωρεάν εργαλεία ανάπτυξης Android λογισμικού που προσφέρει η Google αποτελεί ένα πολύ ισχυρό εργαλείο ανάπτυξης εφαρμογών Android.

Επίσης στο πρώτο κεφάλαιο γίνεται ιστορική αναδρομή, περιγραφή και ανάλυση των χαρακτηριστικών και δυνατοτήτων του Λειτουργικού Συστήματος Android και κάποιων τεχνολογιών που σχετίζονται άμεσα με αυτό.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στο Λειτουργικό Σύστημα Android

Εικόνα 1.1 - Το λογότυπο του Android

1.1. Τι είναι το Android [1]

Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα, βασισμένο στο Linux, για φορητές συσκευές όπως τηλέφωνα και ταμπλέτες. Το γεγονός αυτό σημαίνει ότι μπορεί εύκολα να επεκταθεί και να τροποποιηθεί για να συμβαδίζει και να υιοθετεί τις τελευταίες τεχνολογίες και εξελίξεις.

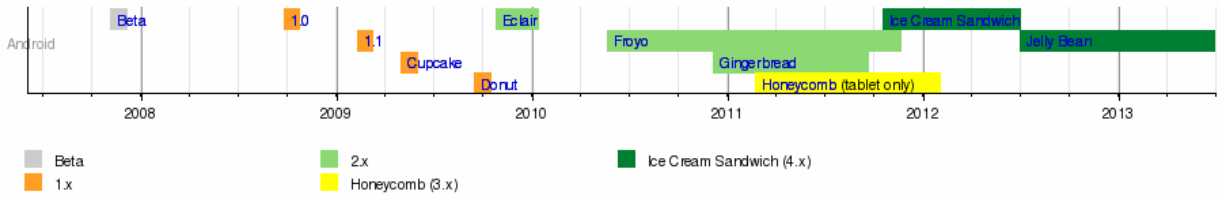
Η Εταιρία Android A.E. ιδρύθηκε στην Καλιφόρνια των Ηνωμένων Πολιτειών τον Οκτώβριο του 2003 και στις 17 Αυγούστου 2005 η Google την εξαγόρασε κάνοντας την εξολοκλήρου θυγατρική της. Στις 5 Νοεμβρίου 2007 ανακοινώθηκε η ίδρυση του οργανισμού Open Handset Alliance η οποία είναι μια κοινοπραξία εταιριών λογισμικού, κατασκευής υλικού και τηλεπικοινωνιών, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις φορητές συσκευές. Παράλληλα την ίδια μέρα έγινε κι η πρώτη παρουσίαση Android κινητού. Η Google παρουσίασε το πρώτο διαθέσιμο smartphone της Nexus One που χρησιμοποιεί το open source

λειτουργικό σύστημα Android στις 5 Ιανουαρίου 2010 το οποίο ήταν κατασκευασμένο από την HTC.

Ο πηγαίος κώδικας του Android είναι διαθέσιμος σύμφωνα με μια Ελεύθερη και Ανοικτού Κώδικα Άδεια Χρήσης. Η Google δημοσιεύει τον πηγαίο κώδικα κάτω από την άδεια χρήσης Apache έκδοση 2.0, ενώ οι αλλαγές που γίνονται στον πυρήνα Linux είναι κάτω από την GNU General Public License έκδοση 2. με τη συμμετοχή του Open Handset Alliance και είναι άμεσα δημόσια διαθέσιμες. Τα υπόλοιπα μέρη του Λειτουργικού συστήματος Android αναπτύσσονται ιδιωτικά, με τον πηγαίο κώδικα να δίνεται δημόσια μετά την ολοκλήρωση και δημοσιοποίηση κάθε νέας έκδοσής του. Η Google σε κάθε νέα έκδοση, συνεργάζεται με κατασκευαστές κινητών τηλεφώνων για την παραγωγή ενός κινητού το οποίο δίνει έμφαση στα νέα χαρακτηριστικά της έκδοσης αυτής.

1.2. Η εξέλιξη του Android [2]

Η Google αναβαθμίζει τις εκδόσεις του Λειτουργικού Συστήματος Android κάθε έξι με εννέα μήνες. Κάθε έκδοση έχει έναν αύξων αριθμό και ένα όνομα το οποίο προέρχεται από κάποιο γλυκό το οποίο ξεκινά με το επόμενο γράμμα της Αγγλικής αλφαβήτου σε σχέση με την προηγούμενη έκδοση.



Εικόνα 1.1 - Χρονοδιάγραμμα Εκδόσεων του Android

Android 1.5 Cupcake

Η πρώτη σταθερή έκδοση του Android ήταν η Cupcake 1.5. Παρουσιάστηκε τον Απρίλιο του 2009 και ήταν βασισμένη στο πυρήνα Linux 2.6.27. Η έκδοση αυτή ενσωμάτωσε πολλά σημαντικά χαρακτηριστικά, τα περισσότερα από τα οποία πλέον είναι σήμα κατατεθέν του Android. Τα σημαντικότερα από αυτά:

- Κινούμενες μεταβάσεις οθόνης
- Υποστήριξη Widget στην αρχική οθόνη
- Υποστήριξη εξωτερικών έξυπνων πληκτρολογίων με δυνατότητα πρόβλεψης κειμένου
- Υποστήριξη αυτόματης εναλλαγής μεταξύ κάθετης και οριζόντιας διάταξης
- Δυνατότητα καταγραφής και αναπαραγωγής βίντεο σε μορφές MPEG-4 και 3GP, όπως επίσης και δυνατότητα μεταφόρτωσης βίντεο στο YouTube και φωτογραφιών στο Picasa απευθείας από το τηλέφωνο
- Υποστήριξη προτύπου Bluetooth A2DP και AVRCP και ικανότητα αυτόματης σύνδεσης με συσκευή η οποία το υποστηρίζει από μια συγκεκριμένη απόσταση

Εικόνα 1.2 - Το λογότυπο του Android 1.5 Cupcake

Android 1.6 Donut

Η επόμενη έκδοση παρουσιάστηκε το Σεπτέμβριο του 2009 με την ονομασία Donut 1.6 και βασιζόταν στον πυρήνα Linux 2.6.29. Τα νέα χαρακτηριστικά της ήταν:

- Δυνατότητα αναζήτησης στο διαδίκτυο, σελιδοδεικτών, ιστορικού και επαφών από την αρχική οθόνη
- Ταχύτερη απόκριση σε σχέση με την προηγούμενη έκδοση
- Βελτιωμένο Android Market
- Υποστήριξη οθονών με ανάλυση WVGA
- Υποστήριξη για Virtual Private Networks
- Ανανεωμένη υποστήριξη τεχνολογιών για CDMA/EVDO, 802.1x, VPNs και με μηχανή μετατροπής κειμένου σε ομιλία (text-to-speech)
- Ενσωματωμένη φωτογραφική μηχανή, βιντεοκάμερα και διεπαφή γκαλερί. Η γκαλερί επιτρέπει πλέον στους χρήστες την επιλογή πολλαπλών φωτογραφιών για διαγραφή
- Ανανεωμένη φωνητική αναζήτηση, με ταχύτερη απόκριση και βαθύτερη ολοκλήρωση με εγγενείς (native) εφαρμογές, συμπεριλαμβανομένης της δυνατότητας κλήσης επαφών
- Υποστήριξη της υπηρεσίας πλοήγισης Google turn-by-turn

Εικόνα 1.3 - Το λογότυπο του Android 1.6 Donut

Android 2.0-2.1 Eclair

Η νέα έκδοση βασισμένη και αυτή στον Linux Kernel 2.6.29, παρουσιάστηκε στις 26 Οκτωβρίου του 2009, ενώ τον Ιανουάριο του επόμενου έτους επανεκδόθηκε σε Android 2.1 Eclair (MR1). Μερικά από τα νέα χαρακτηριστικά:

- Ταχύτερη απόκριση του υλικού σε σχέση με τις δυο προηγούμενες
- Υποστήριξη για περισσότερες οθόνες και αναλύσεις
- Βελτίωση των υπηρεσιών ημερολογίου και χαρτών της Google
- Ενσωματωμένη υποστήριξη flash για την Camera
- Ψηφιακή μεγέθυνση (zoom)
- Ανανεωμένο εικονικό πληκτρολόγιο
- Καινούργιες λίστες επαφών
- Βελτιωμένοι χάρτες Google (google maps) 3.1.2

- Βελτίωση της κλάσης MotionEvent ώστε να υπάρχει η δυνατότητα για γεγονότα πολλαπλής αφής (multitouch events)
- Υποστήριξη της έκδοσης 2.1 του πρωτοκόλλου Bluetooth και όλων νέων χαρακτηριστικών που έφερε

Εικόνα 1.4 - Το λογότυπο του Android 2.0 Eclair

Android 2.2 Froyo

Τον Μάιο του 2010 κυκλοφόρησε η έκδοση Froyo 2.2. Βασίστηκε στον πυρήνα Linux 2.6.32 και μερικές από τις βελτιώσεις και τα νέα χαρακτηριστικά που έφερε ήταν:

- Βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος λόγω καλύτερης διαχείρισης μνήμης
- Ενσωμάτωση στην μηχανή αναζήτησης, της μηχανής Javascript του Chrome V8
- Βελτιώσεις στο χειρισμό της κάμερας
- Νέο πολυγλωσσικό πληκτρολόγιο με γρήγορη εναλλαγή ανάμεσα σε πολλαπλές γλώσσες του

- Υποστήριξη για εγκατάσταση εφαρμογών στην επεκτάσιμη μνήμη
- Βελτιωμένος application launcher
- Σύνδεση USB και λειτουργία WiFi hotspot
- Ανανεωμένη εφαρμογή Android Market με αυτόματη ενημέρωση
- Υποστήριξη Adobe Flash 10.1

Εικόνα 1.5 - Το λογότυπο του Android 2.2 Froyo

Android 2.3.3–2.3.7 Gingerbread

Η επόμενη έκδοση κυκλοφόρησε για πρώτη φορά το Δεκέμβριο του 2010 με πολλές επανακυκλοφορήσεις μέχρι τον Σεπτέμβριο του ίδιου έτους. Μερικά από τα χαρακτηριστικά της έκδοσης αυτής είναι:

- Βελτιωμένο User Interface για απλότητα και ταχύτητα
- Επιλογή λέξεων και αντιγραφή επικόλληση με ένα άγγιγμα
- Υποστήριξη πολλαπλών καμερών (μπροστά και πίσω)
- Βελτιωμένη ενεργειακή διαχείριση

- Υποστήριξη για αισθητήρες όπως βαρόμετρο, γυροσκόπιο και επιταχυνσιόμετρο και υποστήριξη NFC
- Υποστήριξη video κλήσης
- Υποστήριξη οθόνων μεγάλων μεγεθών και αναλύσεων
- Επιλογή αντιγραφής/επικόλλησης με ένα άγγιγμα
- Μετάβαση από το σύστημα αρχείων YAFFS στο ext4 στις νέες συσκευές

Εικόνα 1.6 - Το λογότυπο του Android 2.3 Gingerbread

Android 3.0 Honeycomb

Ακολούθησε η έκδοση με τον αριθμό 3.0 και ονομασία Honeycomb. Βασίστηκε στον πυρήνα Linux 2.6.36 και παρουσιάστηκε πρώτη φορά τον Φεβρουάριο του 2011 με επανεκδόσεις ως τον Φεβρουάριο του 2012. Είχε την ιδιαιτερότητα ότι προοριζόταν αποκλειστικά για tablets και για αυτό το λόγο δεν κυκλοφόρησε ποτέ ελεύθερα ο πηγαίος της κώδικας. Τα χαρακτηριστικά που εισήγαγε είναι:

- Επανασχεδιασμός του γραφικού περιβάλλοντος ώστε να είναι ευκολότερη η χρήση σε συσκευές με μεγάλες οθόνες όπως οι ταμπλέτες
- Υποστήριξη για διπύρηνους και τετραπύρηνους επεξεργαστές
- Βελτιωμένη υποστήριξη για 2D και 3D γραφικών
- Υποστήριξη για φυσικά πληκτρολόγια μέσω Bluetooth ή USB
- Επανασχεδιασμός του εικονικού πληκτρολογίου αφής για ταχύτερη και πιο ακριβή απόκριση
- Κρυπτογράφηση των δεδομένων χρήστη
- Video chat μέσω Google Talk

Εικόνα 1.7 - Το λογότυπο του Android 3.0 Honeycomb

Android 4.0.3–4.0.4 Ice Cream Sandwich

Η επόμενη έκδοση βασισμένη στον πυρήνα Linux 3.0.1, παρουσιάστηκε στις 19 Οκτωβρίου του 2011. Αυτή η έκδοση ουσιαστικά ενσωμάτωνε την εμφάνιση της έκδοσης 3 που ήταν μόνο για ταμπλέτες με την έκδοση 2.3.7. Τα νέα χαρακτηριστικά που είχε:

- Νέο User Interface με εικονικά πλήκτρα τα οποία παίρνουν τη θέση των φυσικών ή αφής που υπήρχαν στις συσκευές
- Δυνατότητα στον χρήστη να τερματίσει εφαρμογές οι οποίες τρέχουν στο background, ενώ μπορεί να θέσει και όρια στην κίνηση πακέτων δεδομένων
- Ο browser μπορεί να ανοίξει ταυτόχρονα μέχρι και 16 καρτέλες
- Η εφαρμογή Android Beam αξιοποιεί πλέον το NFC αφού επιτρέπει την αποστολή δεδομένων από τη συσκευή σε όσες βρίσκονται εντός εμβέλειας. Επίσης με το Wi-Fi Direct οι συσκευές μπορούν να συνδεθούν μεταξύ τους ασύρματα χωρίς την μεσολάβηση κάποιου access point
- Βελτίωση της ασφάλειας του συστήματος με την προσθήκη αναγνώρισης προσώπου για να ξεκλειδώσει η συσκευή
- Υποστήριξη εγγραφής βίντεο σε 1080p

Εικόνα 1.8 - Το λογότυπο του Android 4.0 Ice Cream Sandwich

Android 4.1.x–4.3.x Jelly Bean

Τον Ιούλιο του 2012 κυκλοφόρησε η έκδοση 4.1 με την ονομασία Jelly Bean η οποία βασίστηκε στον πυρήνα Linux 3.0.31. Κάποια από τα χαρακτηριστικά της έκδοσης αυτής είναι:

- Βελτίωση στην απόκριση του συστήματος
- Ανανεωμένο σύστημα ειδοποιήσεων
- Δυνατότητα χρήσης εξωτερικής συσκευής USB
- Δυνατότητα χρήσης της υπηρεσίας Google Wallet
- Βελτιωμένη φωνητική αναζήτηση
- Από την έκδοση 4.2 χρήση του Google Chrome ως προεπιλεγμένου περιηγητή

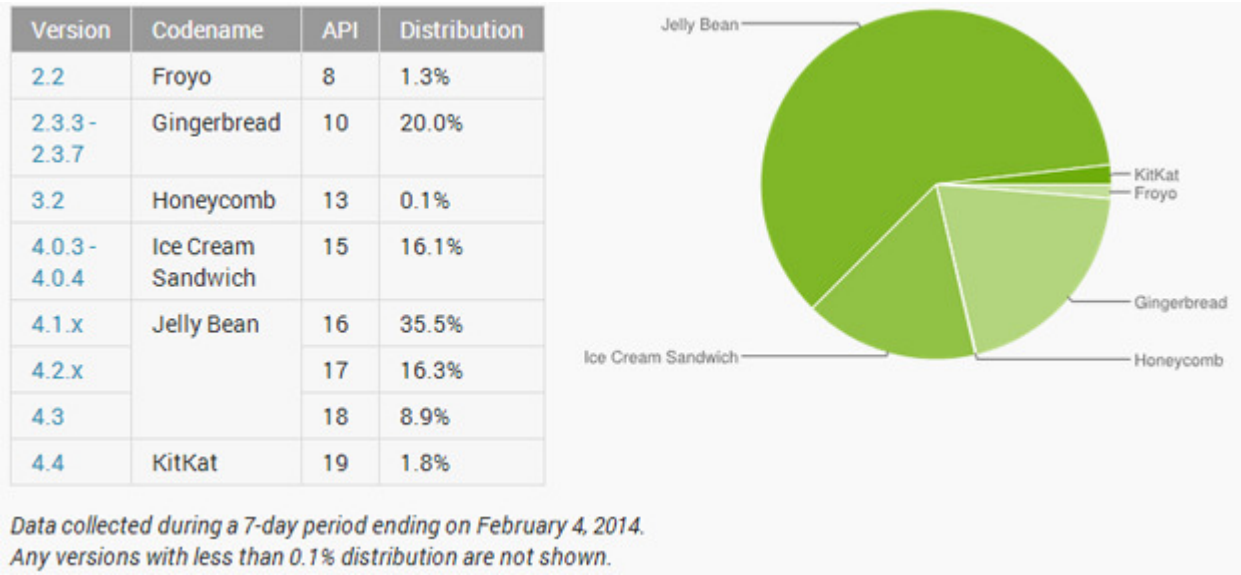
Εικόνα 1.8 - Το λογότυπο του Android 4,1 Jelly Bean

Android 4.4 KitKat

Η τελευταία έκδοση 4.4 τη στιγμή που γράφεται το άρθρο κυκλοφόρησε τον Οκτώβριο του 2013 με την ονομασία KitKat. Κάποια από τα χαρακτηριστικά της έκδοσης αυτής είναι:

- Καλύτερη διαχείριση μνήμης με ελαχίστη απαιτουμένη τα 512MB RAM
- Ενσωμάτωση αποθηκευτικού χώρου με το Google Drive
- Ασύρματη εκτύπωση μέσω WiFi ή Bluetooth
- Καλύτερη παραμετροποίηση της κύριας οθόνης
- Αναβάθμιση του Google Hangouts

Εικόνα 1.9 - Το λογότυπο του Android 4,4 KitKat



Version	Codename	API	Distribution
2.2	Froyo	8	1.3%
2.3.3 - 2.3.7	Gingerbread	10	20.0%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	16.1%
4.1.x	Jelly Bean	16	35.5%
4.2.x		17	16.3%
4.3		18	8.9%
4.4	KitKat	19	1.8%

Data collected during a 7-day period ending on February 4, 2014.
Any versions with less than 0.1% distribution are not shown.

Εικόνα 1.9 - Πίνακας ποσοστών χρήσης των διάφορων εκδόσεων του Android

Όπως φαίνεται στην πιο πάνω εικόνα, το Jelly Bean είναι εγκατεστημένο στο μεγαλύτερο μέρος των συσκευών συγκεντρώνοντας το 51.3% αυτών. Η τελευταία έκδοση KitKat συγκεντρώνει μόλις το 1.4% των συσκευών λόγω του ότι δεν έχει περάσει πολύς καιρός από την επίσημη κυκλοφορία της. Το Honeycomb, ως μια έκδοση αποκλειστικά για ταμπλέτες και ουσιαστικά με μηδαμινό χρόνο κυκλοφορίας, βρίσκεται στο 0.1%. Τέλος οι παλαιότερες εκδόσεις του Android (Cupcake, Donut, Eclair, Ice Cream Sandwich) με τον καιρό θα συρρικνώνονται όπως είναι λογικό. Το Android έχει καταπληκτικά χαρακτηριστικά, πολλαπλές δυνατότητες οι οποίες αναβαθμίζονται και εξελίσσονται με την πάροδο του χρόνου και παρέχει καταπληκτικά εργαλεία για την ανάπτυξη εφαρμογών που κάνουν την ζωή του κατασκευαστή πραγματικά πολύ πιο εύκολη.

1.3. Γιατί επιλογή Android;

Υπάρχουν αρκετοί λόγοι για τους οποίους ένα άτομο μπορεί να επιλέξει μια συσκευή Android έναντι του ανταγωνισμού και κατ' επέκταση ο λόγος που έγινε η επιλογή του Android ως πλατφόρμα ανάπτυξης σε αυτή την πτυχιακή εργασία. Καταρχήν είναι μια πραγματικά ανοιχτή, ελεύθερη πλατφόρμα ανάπτυξης, βασισμένη στο Linux η οποία διαθέτει πάμπολλες ενσωματωμένες υπηρεσίες που μπορούν να κάνουν την εμπειρία του χρήστη μοναδική, όπως υψηλής ποιότητας γραφικά και ήχο, μηχανή αναζήτησης, υπηρεσίες βασισμένες στην τοποθεσία και πανίσχυρη SQL βάση δεδομένων. Ενσωματώνει αυτόματη διαχείριση του κύκλου ζωής μιας εφαρμογής και πολλαπλές δικλίδες ασφαλείας ανάμεσα στα προγράμματα. Η διαχείριση μνήμης με αποτέλεσμα τη χαμηλή κατανάλωση γίνεται με εξαιρετικό τρόπο. Επίσης διαθέτει αρχιτεκτονική βασισμένη σε δομικά στοιχεία τα οποία μπορούν να τροποποιηθούν, να ολοκληρωθούν και να προσαρμοστούν στις ανάγκες κάθε προγραμματιστή και κατά συνέπεια του κάθε τελικού χρήστη. Επιπροσθέτως οι οθόνες μπορούν να τροποποιηθούν κατάλληλα για να υποστηρίζουν οποιαδήποτε ανάλυση, μέγεθος και προσανατολισμό οθόνης. Τέλος υπάρχει φορητότητα ανάμεσα σε ένα ευρύ φάσμα από συσκευές. Αυτό έρχεται σαν απόρροια του γεγονότος ότι όλα τα προγράμματα γράφονται σε Java και εκτελούνται από την εικονική μηχανή Dalvik.

1.4. Αρχιτεκτονική του Android [1]

Το Android δεν είναι μόνο ένα λειτουργικό σύστημα. Είναι μια στοίβα λογισμικού η οποία αποτελείται από το λειτουργικό σύστημα, τις υπηρεσίες διασύνδεσης με τις εφαρμογές (middleware) και τέλος από τις κύριες (core) εφαρμογές. Η αρχιτεκτονική του λειτουργικού συστήματος αποτελείται από 4 βασικά επίπεδα:

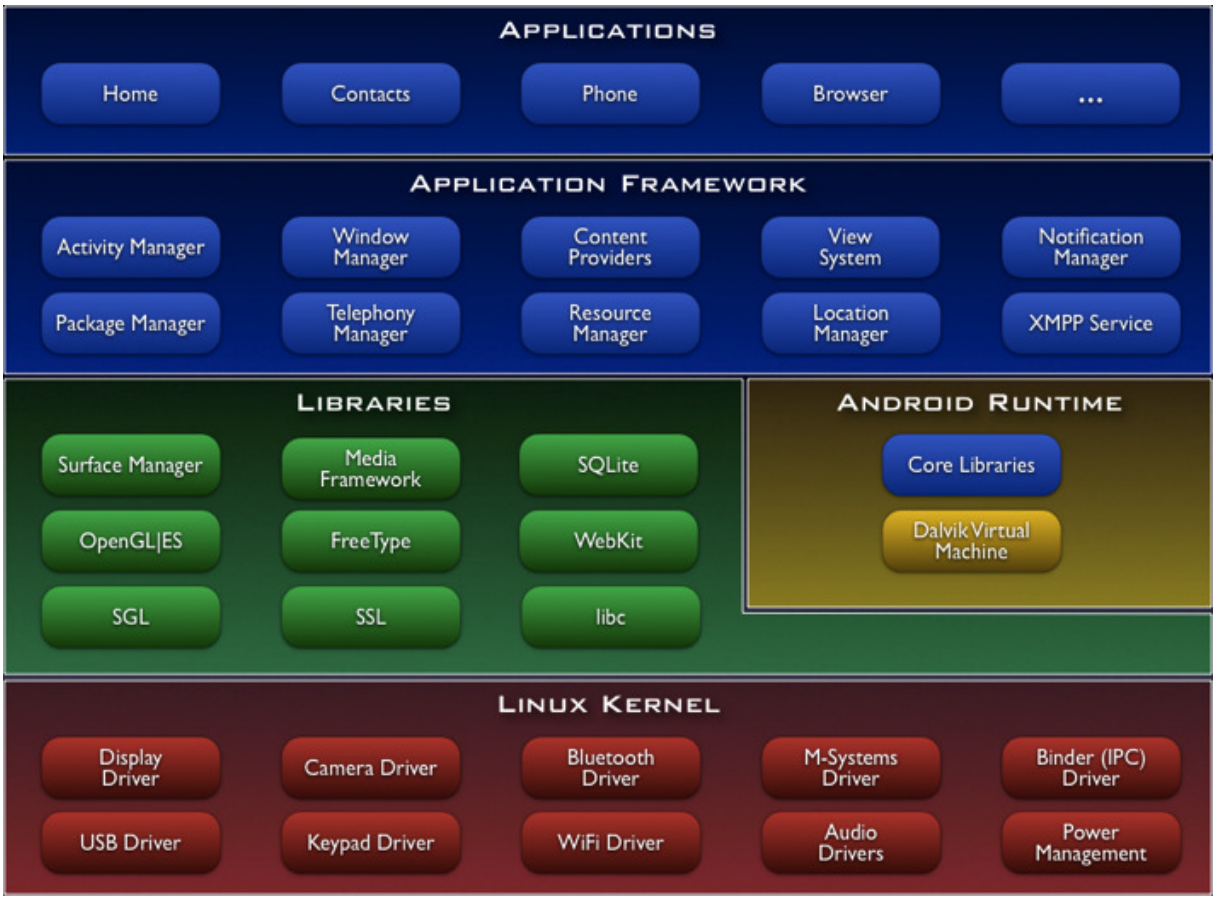
Τον πυρήνα του λειτουργικού (Linux Kernel) στον οποίο βασίζεται το Android και βρίσκεται στο χαμηλότερο επίπεδο. Υποστηρίζει όλες τις κύριες λειτουργίες του λειτουργικού συστήματος όπως τη διαχείριση μνήμης και διεργασιών, τις λειτουργίες δικτύου, την ασφάλεια και ένα σύνολο οδηγών υλικού τους οποίους χρειάζεται το σύστημα για να τρέξει.

Στο πιο πάνω επίπεδο είναι οι βιβλιοθήκες (Libraries) οι οποίες είναι γραμμένες στην γλώσσα προγραμματισμού C και C++ και μεταγλωττίστηκαν για τη συγκεκριμένη αρχιτεκτονική υλικού που χρησιμοποιείται από τη συσκευή. Οι βιβλιοθήκες αυτές δεν είναι εφαρμογές που μπορούν να σταθούν από μόνες τους. Υπάρχουν για να μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Στο ίδιο επίπεδο βρίσκεται ο χρόνος εκτέλεσης (Runtime), ένα σύνολο βασικών βιβλιοθηκών που επιτρέπουν στους προγραμματιστές να γράψουν εφαρμογές χρησιμοποιώντας JAVA οι οποίες εκτελούνται μέσω της εικονικής μηχανής Dalvik.

Στο επόμενο επίπεδο βρίσκεται το πλαίσιο εφαρμογής (Application Framework) το οποίο είναι προ-εγκατεστημένο στο Android, αλλά μπορεί να επεκταθεί από τους προγραμματιστές. Παρέχει υψηλού επιπέδου δομικές μονάδες τις οποίες μπορούμε να χρησιμοποιούμε για την κατασκευή των εφαρμογών μας. Τα σημαντικότερα δομικά στοιχεία του πλαισίου αυτού είναι ο Διαχειριστής δραστηριοτήτων (Activity Manager), ο Διαχειριστής Πόρων (Resource Manager), ο Διαχειριστής Τοποθεσίας (Location Manager), ο Παροχέας Περιεχομένου (Content Providers), και ο Διαχειριστής Κοινοποιήσεων (Notification Manager).

Οι εφαρμογές (Applications) βρίσκονται στο ψηλότερο επίπεδο και είναι αυτό που βλέπουν οι χρήστες χωρίς να γνωρίζουν την όλη διαδικασία από κάτω. Είναι προγράμματα που καταλαμβάνουν ολόκληρη την οθόνη και αλληλεπιδρούν με το

χρήστη αντίθετα με τα widgets τα οποία λειτουργούν σε μικρά τετράγωνα μέσα στην αρχική οθόνη.



Εικόνα 1.10 - Η οπτική απεικόνιση της αρχιτεκτονικής του Android

1.5. Συμπεράσματα

Σε αυτό το κεφάλαιο κάναμε μια εισαγωγή στο λειτουργικό σύστημα Android αναφέροντας ιστορικά την εξέλιξη του, τις εκδόσεις και τα βασικά χαρακτηριστικά του. Αναλύσαμε τον τρόπο λειτουργίας της πλατφόρμας και των εφαρμογών της. Τέλος έγινε αναφορά στο γιατί επιλέξαμε στην συγκεκριμένη πλατφόρμα για την ανάπτυξη της εφαρμογής μας.

ΚΕΦΑΛΑΙΟ 2

Προκλήσεις ανάπτυξης εφαρμογών στο Android και Design

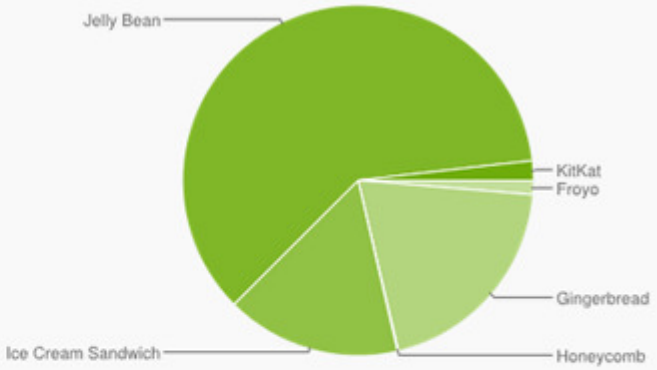
2.1. Guidelines

Η υλοποίηση μιας Android εφαρμογής, όπως και κάθε εφαρμογής στα υπόλοιπα λειτουργικά συστήματα, ξεκινάει από τις λειτουργικές απαιτήσεις. Αυτές είναι οι δυνατότητες και λειτουργίες που θα υποστηρίζει η εφαρμογή, και συνεχίζει με τον σχεδιασμό της διεπαφής χρήστη (User Interface). Ο σχεδιασμός έχει μεγαλύτερη σημασία από τις ίδιες τις λειτουργίες της εφαρμογής αφού είναι το σημείο πρόσβασης προς αυτές. Μια κακοσχεδιασμένη εφαρμογή η οποία κρύβει τις λειτουργίες της πίσω από υπομενού και κρυμμένα κουμπιά ενδέχεται να μπερδέψει και να δυσκολέψει τον χρήστη σε σημείο που να τον αποτρέψει από τη χρήση της. Αυτό φυσικά δεν το επιθυμεί κανένας προγραμματιστής γι αυτό και υπάρχουν κάποιες ενδεικτικές οδηγίες (Guidelines) οι οποίες υποδεικνύουν στους προγραμματιστές το σωστό τρόπο για τον σχεδιασμό της εφαρμογής. Αυτό συγκεκριμένα καθιερώθηκε με την έλευση του Android 4.0 (Ice Cream Sandwich) και την δημοσίευση στο internet σελίδας για Android Design, έτσι ώστε να καθοδηγήσει τους developers σε μία κοινή γραμμή ανάπτυξης εφαρμογών, με απώτερο στόχο την αύξηση της λειτουργικότητας. Αυτό επιτυγχάνεται με την εκμάθηση στον χρήστη των σημείων επαφής σε κάθε εφαρμογή, οπότε δεν θα χρειάζεται να ψάχνει σε κάθε μια πώς να επιστρέψει στην αρχική οθόνη ή που βρίσκεται το μενού των επιλογών, κοκ. Στη σελίδα καθοδήγησης υπάρχουν παραδείγματα “καλού σχεδιασμού” τα οποία αφορούν την χρήση της Action Bar, την χρήση των swappable tabs (η δυνατότητα να αλλάζουμε οθόνες σέρνοντας το δάχτυλό μας στην οθόνη της συσκευής), και άλλα πολλά. Επίσης υπάρχει και η δυνατότητα για κατέβασμα πακέτου εικονιδίων τα οποία μπορούμε να χρησιμοποιήσουμε στο μενού των εφαρμογών μας η όπου αλλού θέλουμε. Φυσικά αν ο developer το επιθυμεί μπορεί να σχεδιάσει το δικό του interface το οποίο θα τον εξυπηρετεί καλύτερα, αφού οι οδηγίες αυτές είναι ενδεικτικές και όχι αναγκαστικές.

2.2. Υποστήριξη παλαιότερων εκδόσεων του Android

Αυτή τη στιγμή υπάρχουν 19 διαθέσιμες εκδόσεις του Android με πιο πρόσφατη την έκδοση 4.4 KitKat (API 19) με κυρίαρχη την έκδοση την 4.1 Jelly Bean (API 16). Αυτή η συνεχής εξέλιξη της πλατφόρμας αποτελεί πρόκληση για τον προγραμματιστή ο οποίος θα πρέπει να ακολουθεί να χρησιμοποιεί τις νέες δυνατότητες που του παρέχει η κάθε νέα έκδοση, χωρίς να παραγκωνίζει την υποστήριξη στις προηγούμενες. Αρκετά από τα νέα χαρακτηριστικά δεν υποστηρίζονται στα παλιότερα APIs, με αποτέλεσμα η χρήση τους να είναι αδύνατη σε κάποια έκδοση με παλιότερο API. Η λύση στο πρόβλημα είναι οι βιβλιοθήκες συμβατότητας που προσφέρει η Google σε κάθε νέα έκδοση του λειτουργικού της. Σκοπός τους είναι να κάνουν διαθέσιμα τα νέα εργαλεία στα παλιότερα APIs. Σ' αυτό πολύτιμη βοήθεια δίνουν οι προγραμματιστές του Android που αναπτύσσουν βιβλιοθήκες για να υποστηρίξουν τους περιορισμούς χρήσης και το κενό που άφησε η Google μεταξύ των εκδόσεων.

Version	Codename	API	Distribution
2.2	Froyo	8	1.3%
2.3.3 - 2.3.7	Gingerbread	10	20.0%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	16.1%
4.1.x	Jelly Bean	16	35.5%
4.2.x		17	16.3%
4.3		18	8.9%
4.4	KitKat	19	1.8%



Data collected during a 7-day period ending on February 4, 2014.
Any versions with less than 0.1% distribution are not shown.

Εικόνα 2.1 - Στατιστικά στοιχεία ενεργών συσκευών Android – Φεβρουάριος 2014 [8]

Όπως βλέπουμε από τα παραπάνω στατιστικά στοιχεία που ενημερώνονται αυτόματα κάθε δύο εβδομάδες από την Google, τη μερίδα του λέοντος κατέχει η έκδοση 4.1.X με ποσοστό 35,5% και ακολουθεί η σχετικά απαρχαιωμένη έκδοση 2.3.X με ποσοστό 20,0%. Οι νεότερη έκδοση του Android 4.4 KitKat κατέχει μόλις το 1,8% κάτι το οποίο οφείλεται στο -κακό- φαινόμενο διάσπασης (fragmentation) του

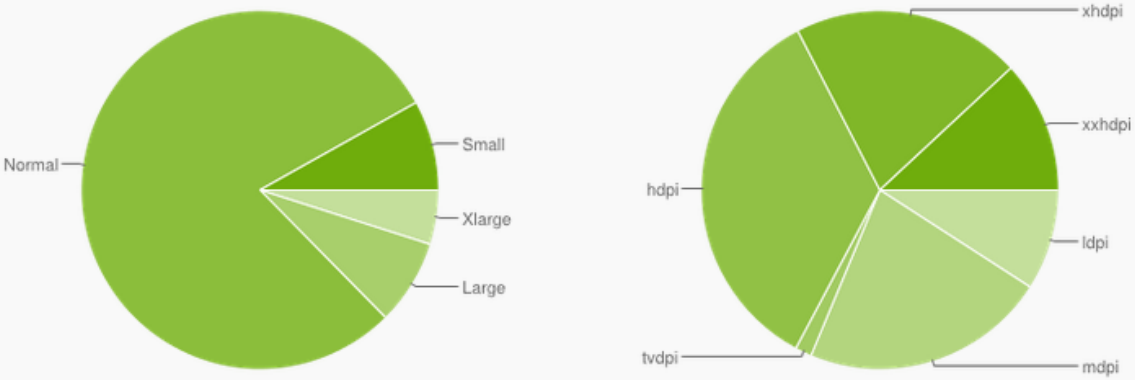
Android. Γενικά μια καλή προγραμματιστική συμπεριφορά αποδεκτή από την κοινότητα, είναι η εκάστοτε εφαρμογή να υποστηρίζει περίπου το 90% των ενεργών συσκευών. Αυτό αν κρίνουμε από το παραπάνω γράφημα μπορεί να επιτευχθεί εύκολα, καθώς υποστηρίζοντας τις εκδόσεις 4.0 και άνω, υποστηρίζουμε το 80% περίπου των ενεργών συσκευών. Οι προγραμματιστές πρέπει να βλέπουν και να αξιολογούν το μερίδιο των συσκευών με τέτοιο τρόπο ώστε να υποστηρίζουν όσο το δυνατό περισσότερες συσκευές χωρίς όμως να υποβαθμίζουν την ποιότητα και τη λειτουργικότητα της εφαρμογής τους. Πρόκειται για μια λεπτή ισορροπία που επιτυγχάνεται μετά από αρκετή προσπάθεια από μέρος τους. Η υποστήριξη των διαφορετικών εκδόσεων ορίζεται στο αρχείο AndroidManifest.xml και εφόσον έχει καθοριστεί το κατώτατο επιθυμητό API, η εφαρμογή δεν μπορεί να εγκατασταθεί σε συσκευή που φοράει παλαιότερη έκδοση από αυτή που υποστηρίζει το API. Μια συνετή πρακτική σχεδιασμού είναι η στόχευση σε ένα σχετικά χαμηλό API το οποίο όμως δεν θα μας αναγκάσει να κάνουμε υποχωρήσεις και συμβιβασμούς στις λειτουργίες της εφαρμογής μας.

2.3. Υποστήριξη πολλαπλών συσκευών και πολλαπλών διαστάσεων οθόνης και πυκνότητας pixel [9]

Είναι κατανοητό ότι υπάρχει μια πληθώρα συσκευών οι οποίες έχουν διαφορετικά χαρακτηριστικά, αλλά πρέπει να τρέχουν την ίδια εφαρμογή εξίσου σωστά. Ένα από τα χαρακτηριστικά αυτά είναι οι πολλές Android εκδόσεις που κυκλοφορούν σήμερα στην αγορά. Επίσης, υπάρχει τεράστια ποικιλία τεχνικών χαρακτηριστικών σε συσκευές οι οποίες συχνά διαφέρουν από χώρα σε χώρα ακόμα και για την ίδιο μοντέλο. Υπάρχουν συσκευές με διπύρηνο η τετραπύρηνο επεξεργαστή χρονισμένο σε διαφορετικές συχνότητες, οι όποιοι μπορεί να είναι και διαφορετικής αρχιτεκτονικής 32 bit ή 64 bit. Ακόμη, ένα άλλο σημαντικό χαρακτηριστικό που μπορεί να διαφέρει στις συσκευές είναι η μνήμη RAM η οποία σήμερα κυμαίνεται από 1 GB μέχρι 3 GB στις συσκευές που κυκλοφορούν.

Η σημαντικότερη όμως πρόκληση για τον developer είναι η ταυτόχρονη υποστήριξη των πολλαπλών διαστάσεων οθόνης όπως επίσης και η διαφορετική πυκνότητα pixel που διαθέτουν οι συσκευές που κυκλοφορούν στην αγορά. Όπως είναι κατανοητό, το γραφικό περιβάλλον μιας εφαρμογής είναι ίσως σημαντικότερο και από τις δυνατότητες που παρέχει, καθότι ένα κακοσχεδιασμένο ή μη προσαρμόσιμο με όλους τους τύπους οθονών layout μπορεί να κάνει την εφαρμογή δύσχρηστη έως άχρηστη. Όσον αφορά το εύρος των Android συσκευών μπορεί να φτάσει από τις 2.6' (μικρά smartphones) έως τις 12.2' (μεγάλα tablets), η ανάλυση αυτών ξεκινάει από τα 240x320 pixels (QVGA) και φτάνει μέχρι τα 2560×1440 (Quad HD) με την πιο δημοφιλή να είναι η 480x800 (WVGA).

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	8.1%						8.1%
Normal	0.2%	13.6%		33.7%	19.9%	11.9%	79.3%
Large	0.7%	4.3%	1.5%	0.6%	0.6%		7.7%
Xlarge	0.1%	4.3%		0.3%	0.2%		4.9%
Total	9.1%	22.2%	1.5%	34.6%	20.7%	11.9%	



Data collected during a 7-day period ending on March 3, 2014.
Any screen configurations with less than 0.1% distribution are not shown.

Εικόνα 2.2 - DPI Dashboard [8]

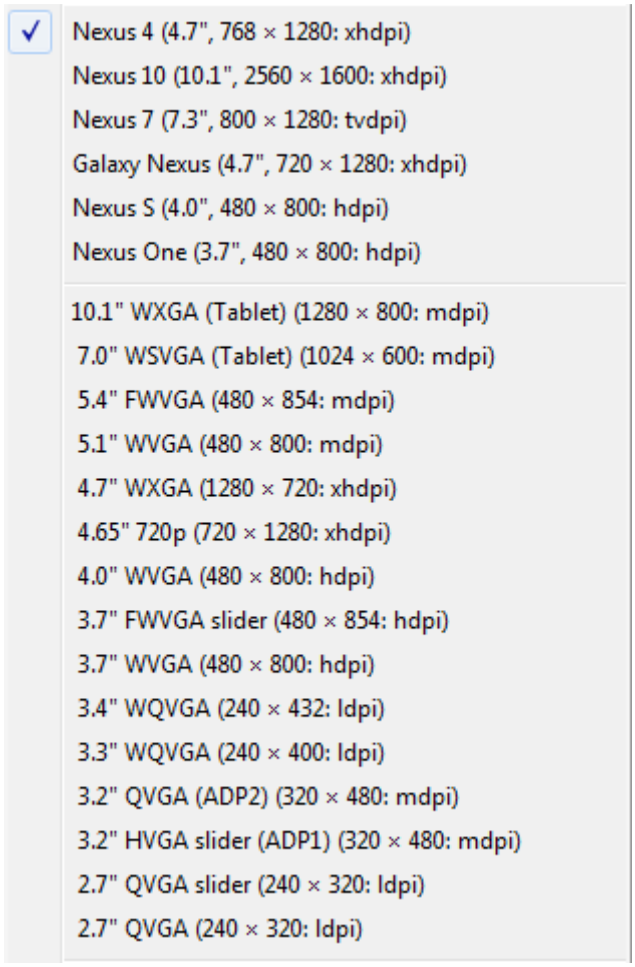
Το Android για πρακτικούς λόγους έχει χωρίσει τις διαφορετικές αναλύσεις οθονών σε τέσσερις κατηγορίες οι οποίες συσχετίζονται με την πυκνότητα εικονοκυττάρου ανά ίντσα (dpi). Όπως βλέπουμε από το παρακάτω σχεδιάγραμμα, το Android χωρίζει το μέγεθος τις οθόνης σε τέσσερις επιμέρους κατηγορίες οθονών αναλόγως το μέγεθος σε ίντσες, και οι αναλύσεις χωρίζονται επίσης σε τέσσερις επιμέρους κατηγορίες dpi. Αυτό γίνεται ώστε να διευκολυνθούν όσο το δυνατόν περισσότερο οι προγραμματιστές να βελτιώσουν την εμφάνιση των εφαρμογών τους, αφού υπάρχει μια πληθώρα αναλύσεων για να υποστηρίξει, η κάθε μία με περισσότερο η λιγότερο διαθέσιμο χώρο στην οθόνη.

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *hdpi*	Extra high density (320), *xhdpi*
Small screen	**QVGA (240x320)**		480x640	
Normal screen	**WQVGA400 (240x400)** **WQVGA432 (240x432)**	**HVGA (320x480)**	**WVGA800 (480x800)** **WVGA854 (480x854)** 600x1024	640x960
Large screen	**WVGA800** (480x800)** **WVGA854** (480x854)**	**WVGA800* (480x800)** **WVGA854* (480x854)** 600x1024		
Extra Large screen	1024x600	**WXGA (1280x800)**† 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600

* To emulate this configuration, specify a custom density of 160 when creating an AVD that uses a WVGA800 or WVGA854 skin.
** To emulate this configuration, specify a custom density of 120 when creating an AVD that uses a WVGA800 or WVGA854 skin.
† This skin is available with the Android 3.0 platform

Εικόνα 2.3 - Ανάλυση οθονών και dpi

Η Google παρέχει στους developers την κατανομή μεγέθους οθόνης προς dpi για να τους βοηθήσει να σχεδιάσουν τις εφαρμογές τους αποδοτικότερα [10]. Από ότι βλέπουμε από το παραπάνω σχεδιάγραμμα, τα πράγματα για τον προγραμματιστή δεν είναι τόσο δύσκολα όσο φαίνονται αφού το επικρατέστερο μέγεθος είναι το WVGA 480x800 240 dpi. Φυσικά πρέπει να λάβει υπόψη και τις υπόλοιπες διατάξεις μεγέθους οθόνης προς dpi, αλλά όπως και στην περίπτωση της υποστήριξης των API, αυτό έγκειται στην απόφαση του και αυτό δηλώνεται στο αρχείο AndroidManifest.xml. Εδώ καταλήγουμε στο γεγονός 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. Βεβαία, υπάρχει και η δυνατότητα χρήσης διαφορετικών από τα συνηθισμένα layout ανά διαφορετικό μέγεθος οθόνης μέσω του SDK. Αυτό σημαίνει ότι ο developer δεν χρειάζεται να συμβιβαστεί με ένα layout xml ώστε να καλύψει όλες τις οθόνες. Μπορεί να χρησιμοποιήσει όσα είναι αναγκαία για να καλυφθούν όσο το δυνατόν περισσότερα μεγέθη και αναλύσεις οθόνης. Η υλοποίηση αυτής της δυνατότητας είναι 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.



Εικόνα 2.4 - Διάφορα μεγέθη οθονών ανά dpi

ΚΕΦΑΛΑΙΟ 3

3.1. Ανατομία μιας εφαρμογής Android

Υπάρχουν 4 τμήματα κατασκευής (building blocks) σε μια εφαρμογή Android τα οποία πρέπει να δηλωθούν στο αρχείο AndroidManifest.xml:

- Activity (Δραστηριότητα)
- Intent Receiver (Δέκτης Πρόθεσης)
- Content Provider (Παροχής Περιεχομένου)
- Service (Υπηρεσία)

3.1.1. Activity (Δραστηριότητα)

Τα Activities είναι τα πιο κοινά από τα 4 τμήματα που προαναφέρθηκαν. Μια δραστηριότητα συνήθως είναι μια απλή οθόνη της εφαρμογής. Κάθε δραστηριότητα υλοποιείται σαν μια κλάση που επεκτείνει (extends) την βασική κλάση Δραστηριότητα (Activity base class). Η κλάση προβάλει μια διεπαφή χρήστη (user interface) αποτελούμενη από Εικόνες (Views) και απαντά σε Συμβάντα (Events). Οι πλείστες εφαρμογές αποτελούνται από πολλαπλές οθόνες. Για παράδειγμα μπορούμε να έχουμε μια εφαρμογή ανταλλαγής γραπτών μηνυμάτων η οποία θα μπορούσε να έχει μια οθόνη που εμφανίζει τις επαφές για να διαλέξουμε σε ποιον θα σταλεί το μήνυμα. Μια δεύτερη οθόνη για να γράψουμε και να αποστείλουμε το μήνυμα στην επιλεγμένη επαφή, άλλη οθόνη για να βλέπουμε παλιά μνήματα και άλλη για να αλλάξουμε τις ρυθμίσεις. Κάθε μια από αυτές τις οθόνες υλοποιείται σαν μια δραστηριότητα και η μετάβαση σε άλλη οθόνη επιτυγχάνεται με την έναρξη μιας νέας δραστηριότητας. Όταν μια νέα οθόνη ανοίγει, η προηγούμενη οθόνη μπαίνει σε μια στοίβα ιστορικού (history stack). Ο χρήστης μπορεί να πλοηγηθεί σε προηγούμενες οθόνες μέσω του ιστορικού. Οι οθόνες μπορούν επίσης να αφαιρεθούν από το ιστορικό σε περιπτώσεως που δεν χρειάζεται να παραμείνουν προσβάσιμες. Το Android διατηρεί ιστορικό για κάθε εφαρμογή που εκκίνησε από την κεντρική οθόνη (home screen).

3.1.2. Intent and Intent Filters (Πρόθεση και φίλτρα Πρόθεσης)

Το Android χρησιμοποιεί μια ειδική κλάση που λέγεται Πρόθεση (Intent) για να κινείται από οθόνη σε οθόνη. Η πρόθεση περιγράφει τι θέλει η εφαρμογή να γίνει στη συνεχεία. Τα δυο πιο σημαντικά μέρη της δομής δεδομένων της πρόθεσης είναι η δράση (action) και τα δεδομένα βάσει των οποίων αυτή θα εκτελεστεί. Τυπικές τιμές για μια δράση είναι η main, η view, edit, η pick κλπ.

Η κλάση Φίλτρο Πρόθεσης (IntentFilter) είναι μια περιγραφή του τι είναι δυνατό να διαχειριστεί ένας Δέκτης Πρόθεσης (intent reciever).

Η πλοήγηση από οθόνη σε οθόνη πετυχαίνεται με τις Προθεσεις. Για την πλοήγηση προς τα εμπρός μια δραστηριότητα καλεί την start Activity(myIntent). Το σύστημα κοιτά τα φίλτρα Προθέσεων (Intent filters) για όλες τις εγκατεστημένες εφαρμογές και διαλέγει αυτήν που τα φίλτρα Πρόθεσης ταιριάζουν καλύτερα με την myIntent παράμετρο της κλάσης. Τότε η νέα δραστηριότητα ενημερώνεται για την Πρόθεση και ξεκινά.

3.1.3. Intent Receiver (Δέκτης Πρόθεσης)

Μπορούμε να χρησιμοποιήσουμε ένα IntentReceiver (Δέκτης Πρόθεσης) όταν θέλουμε η εφαρμογή μας να εκτελεστεί σε απάντηση ενός εξωτερικού συμβάντος (external event). Οι Δέκτες Πρόθεσης μπορούν να προβάλουν Ειδοποιήσεις (Notifications) στον χρήστη. Είναι καταχωρημένοι στο AndroidManifest.xml, αλλά μπορούν να καταχωρηθούν και μέσω της Context.registerReceiver().

3.1.4. Content Provider (Πάροχος Περιεχομένου)

Οι εφαρμογές μπορούν να σώζουν τα δεδομένα τους σε αρχεία, σε μια SQLite βάση δεδομένων, σε προτιμήσεις (preferences) ή σε οποιοδήποτε άλλο μηχανισμό μπορούν. Ένας Content Provider είναι χρήσιμος αν θέλουμε τα δεδομένα της εφαρμογής να είναι διαθέσιμα και σε άλλες εφαρμογές. Επίσης είναι μια κλάση που υλοποιεί μια συγκεκριμένη ομάδα μεθόδων που επιτρέπουν σε άλλες εφαρμογές να αποθηκεύσουν και να επανακτήσουν δεδομένα του τύπου που διαχειρίζεται ο Content Provider.

3.1.5. Service

Service είναι κώδικας που τρέχει για μεγάλο χρονικό διάσημα και χωρίς διαπαφή χρηστή (UI). Για παράδειγμα, μια εφαρμογή αναπαραγωγής μουσικής (media player) η οποία αναπαράγει μουσική από μια λίστα τραγουδιών. Σε τέτοια περίπτωση πιθανόν να υπάρχουν μια ή και παραπάνω Δραστηριότητες που επιτρέπουν στο χρηστή να επιλέξει τραγούδια για αναπαραγωγή. Η Δραστηριότητα δεν πρέπει να διαχειρίζεται την αναπαραγωγή γιατί ο χρηστής επιθυμεί η μουσική να συνεχίζει να παίζει ακόμα και μετά από πλοήγηση σε άλλη οθόνη. Σε αυτή την περίπτωση η Δραστηριότητα της αναπαραγωγής μουσικής θα ξεκινούσε μια υπηρεσία χρησιμοποιώντας την Context.startService() για να τρέξει στο φόντο και να συνεχίσει η μουσική να παίζει. Σ' αυτό το σημείο αξίζει να αναφέρουμε ότι μπορούμε να συνδεθούμε σε μια υπηρεσία (και να την εκκινήσουμε) με τη μέθοδο Context.bindService().

3.2. Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής στο Android [5]

Η ανάπτυξη εφαρμογών στο Android είναι μια σύνθετη και χρονοβόρα διαδικασία η οποία αποτελείται από 4 βασικά στάδια.

3.2.1. 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. Υπάρχουν περιβάλλοντα ανάπτυξης (IDE) από τα οποία μπορεί να επιλέξει όποιο τον εξυπηρετεί καλύτερα και να χρησιμοποιήσει όποιο εργαλείο του Android SDK θεωρεί χρήσιμο. Ακολούθως θα πρέπει μέσω της διαχείρισης εικονικών συσκευών (AVD) να δημιουργήσει τις αντίστοιχες εικονικές συσκευές στις οποίες θέλει να τρέχει το πρόγραμμα του. Η ευελιξία των AVDs θα βοηθήσει να κερδηθεί πολύτιμος χρόνος.

3.2.2. Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής

Σε αυτό το στάδιο της πιο χρονοβόρας και πολύπλοκης διαδικασίας ο προγραμματιστής πρέπει να αποφασίσει για τις δυνατότητες και το περιεχόμενο που θα περιλαμβάνει η εφαρμογή. Καλείται να εντοπίσει ποιες από αυτές τις δυνατότητες είναι εφικτές και ποιες θέλουν παραπάνω έρευνα, να σχεδιάσει την εμφάνιση με γνώμονα την λειτουργικότητα και να αποφύγει υπερβολές στο σχεδιασμό έτσι ώστε να έχει το καλύτερο δυνατό τελικό αποτέλεσμα. Η διαδικασία ξεκινάει με ένα νέο Project το οποίο θα περιέχει τον πηγαίο κώδικα, τις εικόνες, τα κείμενα και ότι χρειάζεται η εφαρμογή για να τρέξει. Ο προγραμματιστής θα πρέπει να φροντίσει ώστε το υλικό του να είναι σωστά δομημένο κι ο κώδικας του ευανάγνωστος για μελλοντική επεξεργασία.

3.2.3. Αποσφαλμάτωση (Debugging) και Δοκιμαστική Φάση Εφαρμογής

Η διαδικασία του debugging αποτελείται από αρκετά επί μέρους στάδια τα οποία αναλύονται παρακάτω.

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

Το επόμενο στάδιο είναι η Ανάπτυξη Πηγαίου Κώδικα Εφαρμογής. Εδώ ο προγραμματιστής καλείτε να αντιμετωπίσει τα λειτουργικά και αισθητικά προβλήματα της εφαρμογής του, πρώτα εντοπίζοντας τα στην λειτουργία 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 εντοπίζει το σημείο εκείνο που συνέβη το σφάλμα.

Στο τρίτο στάδιο ο προγραμματιστής αφού έχει τελειώσει την αποσφαλμάτωση (debugging) επιστρέφει στο αρχικό βήμα, δηλαδή στο compile και τη δοκιμή της εφαρμογής σε εικονική η φυσική συσκευή ώστε να διαπιστώσει τα αποτελέσματα του δευτέρου βήματος, της αποσφαλμάτωσης. Η διαδικασία του debugging επαναλαμβάνεται συνέχεια μέχρι να εντοπιστούν και να διορθωθούν όλα τα σφάλματα της εφαρμογής, και για αυτό το λόγο μπορεί να αποδειχθεί εξαιρετικά χρονοβόρα.

3.2.4. Τελική έκδοση και δημοσίευση της εφαρμογής στο κοινό

Στο τέταρτο και τελευταίο στάδιο της ανάπτυξης ο προγραμματιστής κάνει τους τελευταίους ελέγχους και τυχόν ρυθμίσεις της εφαρμογής, αφού έχει διορθώσει όλα τα σφάλματα που προέκυψαν από την διαδικασία αποσφαλμάτωσης. Ακολούθως γίνεται το τελικό compile της εφαρμογής σε κανονική λειτουργία αυτή τη φορά και όχι debug. Τέλος ακολουθεί η διάθεση της εφαρμογής, δωρεάν ή επί πληρωμή με το μέσο της επιλογής του προγραμματιστή, όπως για παράδειγμα στο Google Play, αφού πρώτα κάνει λογαριασμό developer, ή να την διαθέσει σε κάποιο εναλλακτικό market ή στην προσωπική του ιστοσελίδα.

ΚΕΦΑΛΑΙΟ 4

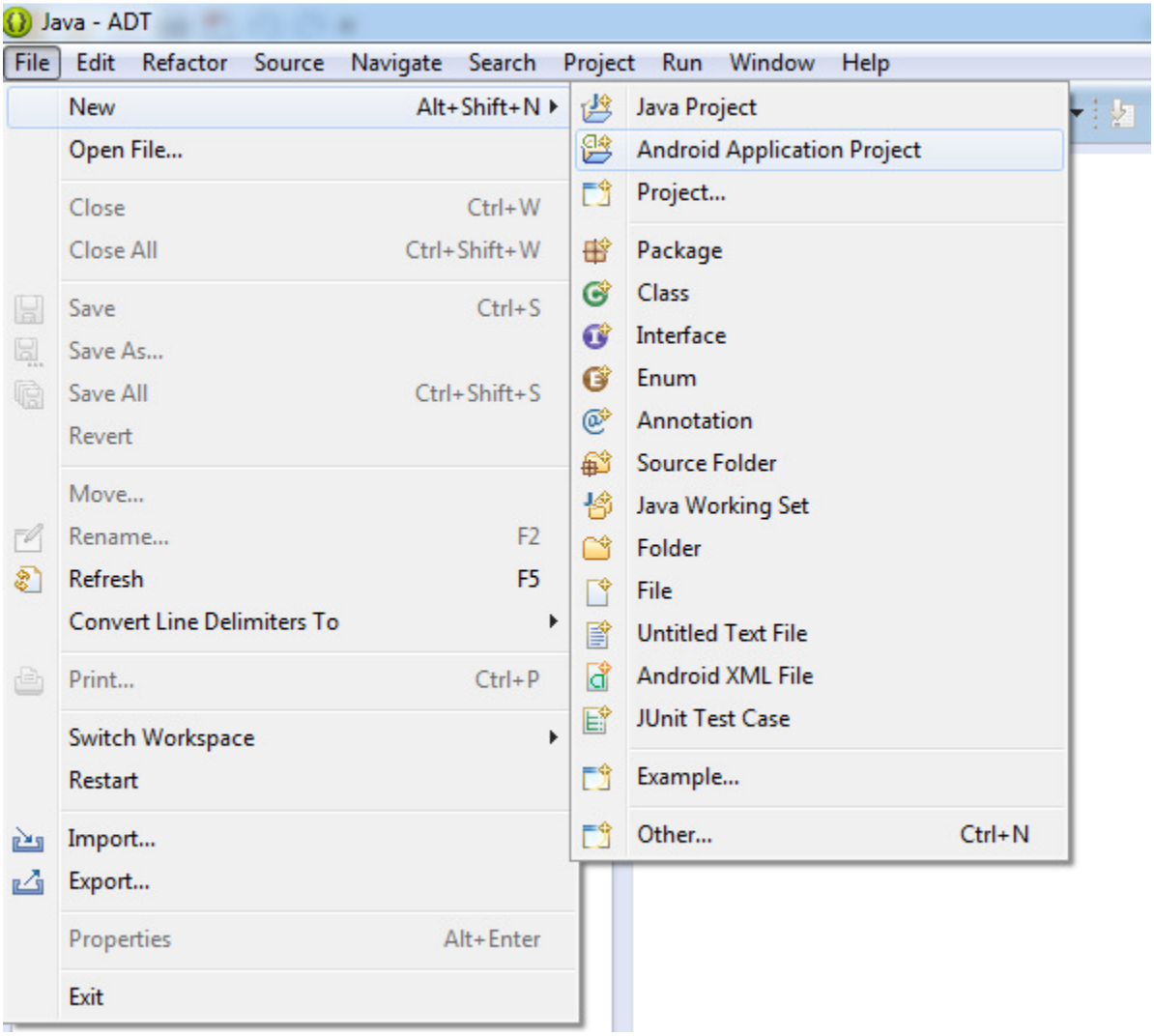
Εισαγωγή στο Περιβάλλον Ανάπτυξης [6]

Αρχικά το πρώτο πράγμα που πρέπει να κάνουμε είναι να εγκαταστήσουμε στον υπολογιστή μας όλα τα απαραίτητα προγράμματα που θα χρειαστούμε για να αναπτύξουμε την Android εφαρμογή. Το περιβάλλον ανάπτυξης αποτελείται από τα ακόλουθα εργαλεία και είναι δωρεάν: Java, IDE και SDK. Αξίζει να σημειωθεί ότι πρέπει να εγκατασταθούν με την προαναφερθείσα σειρά γιατί το ένα έχει ως προϋπόθεση την εγκατάσταση του προηγούμενου. Απαραίτητη για τη ανάπτυξη κώδικα Android εφαρμογής είναι η γνώση προγραμματισμού σε Java.

Πριν ξεκινήσουμε να δούμε πώς μπορούμε να δημιουργήσουμε ένα νέο project πρέπει πρώτα να έχουμε εγκαταστήσει το Android SDK. Απαραίτητη προϋπόθεση είναι η ύπαρξη του πιο πρόσφατου JDK, το οποίο μπορεί κανείς να κατεβάσει από την εξής ιστοσελίδα http://www.oracle.com/technetwork/java/javase/downloads/index.html καθώς επίσης και το IDE το οποίο μπορούμε να βρούμε στο σύνδεσμο http://www.eclipse.org/downloads. Επόμενο βήμα, είναι η εγκατάσταση του Android SDK το οποίο μπορούμε να κατεβάσουμε από http://developer.android.com/sdk/index.html.Τέλος, ένα χρήσιμο εργαλείο για την εύκολη δημιουργία εφαρμογών είναι το ADT Plugin [7] η εγκατάσταση του όποιου γίνεται μέσα από το περιβάλλον του Eclipse με την εξής διαδικασία: ο χρήστης επιλέγει από το μενού Help και Install New Software. Από το παράθυρο που εμφανίζεται επιλέγει το κουμπί Add και προσθέτει το ADT plugin τοποθετώντας στο πεδίο Location την διεύθυνση https://dl-ssl.google.com/android/eclipse/ και ολοκληρώνει την εγκατάσταση αποδέχοντας τους όρους.

Μετά από αυτή τη διαδικασία είμαστε έτοιμοι να δημιουργήσουμε και να τρέξουμε το πρώτο μας Android Project μέσα από όλα τα βήματα που απαιτούνται.

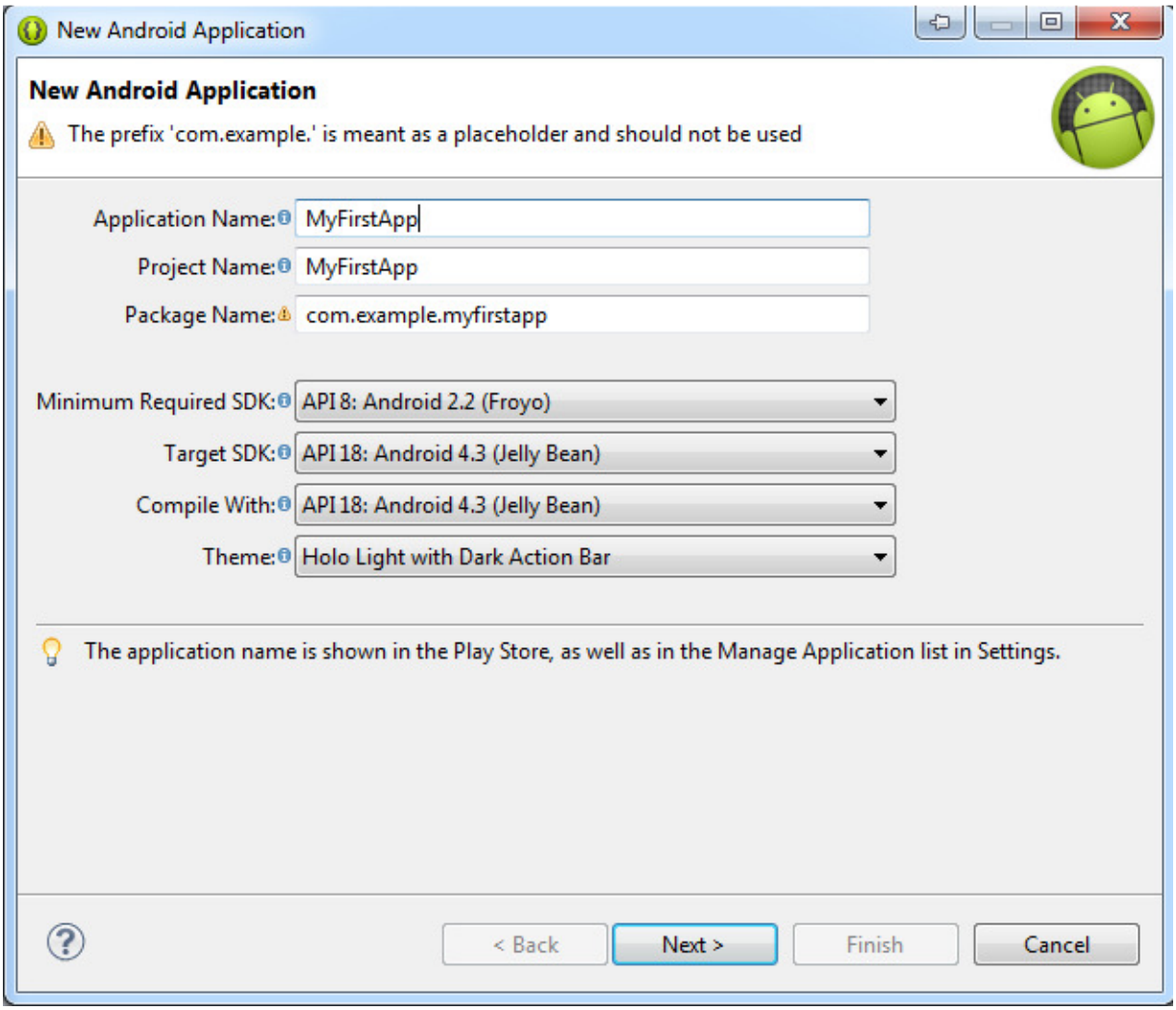
Αρχικά ξεκινάμε το Eclipse. Πρώτο βήμα είναι η δημιουργία ενός νέου Android Application Project, το όποιο γίνεται από το File στο κυρίως μενού και επιλέγουμε New και ακολούθως την επιλογή Android Application Project.



Εικόνα 4.1 - Δημιουργία ενός Android Application Project

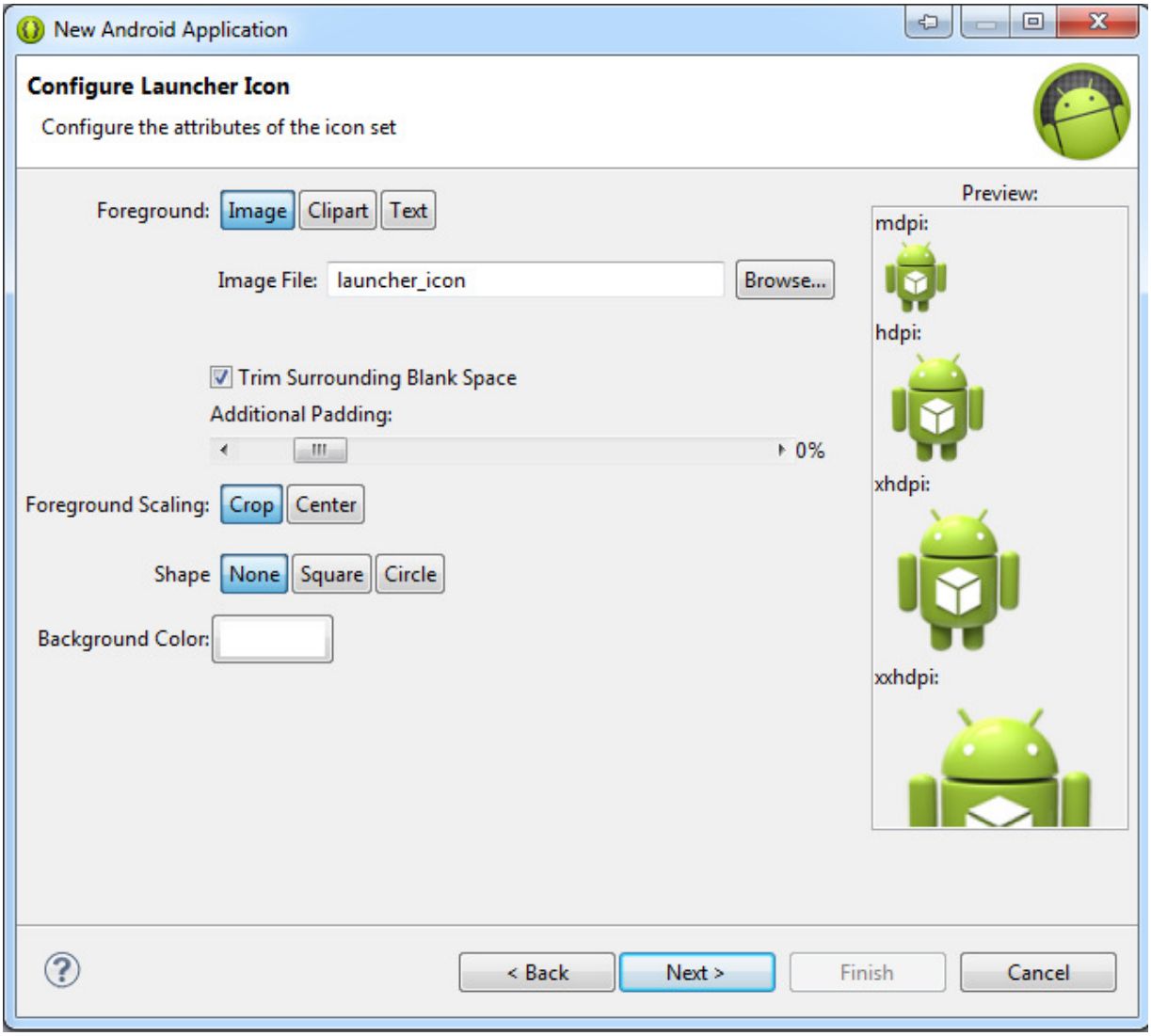
Τώρα έχουμε μπροστά μας το New Android Application παράθυρο, με τα πεδία που πρέπει να συμπληρωθούν να είναι τα εξής: Στο πεδίο Application Name γράφουμε το όνομα που θέλουμε να δώσουμε στην εφαρμογή μας. Είναι το όνομα που θα εμφανίζεται και στην Android συσκευή όταν εγκαταστήσουμε την εφαρμογή μας. Στο πεδίο Project Name δηλώνουμε το όνομα της εφαρμογής. Είναι το όνομα το οποίο θα έχει η εφαρμογή και θα βλέπουν οι χρήστες της. Τα packages, όπως είναι η ονομασία

τους, ταξινομούν και ομαδοποιούν τα java αρχεία για ευκολότερη αντιμετώπιση σφαλμάτων σε επίπεδο κώδικα αλλά και για να καθορίζουν τα δικαιώματα πρόσβασης. Σε εφαρμογές Android είναι υποχρεωτικό να δημιουργήσουμε ένα πακέτο μέσα στο οποίο θα περιέχεται η java κλάση της Android εφαρμογής. Στη συνεχεία επιλέγουμε την έκδοση API στην οποία θέλουμε να αναπτύξουμε την εφαρμογή ή αφήνουμε την προεπιλεγμένη ως έχει. Σ' αυτό το σημείο αξίζει να σημειώσουμε ότι ο αριθμός στο πεδίο Minimum SDK εμφανίζεται η επιλογή API που διαλέξαμε στο προηγούμενο βήμα και είναι η χαμηλότερη έκδοση που θα υποστηρίζει η εφαρμογή. Αφού συμπληρώσουμε όλα τα απαραίτητα στοιχεία πατάμε το Next.



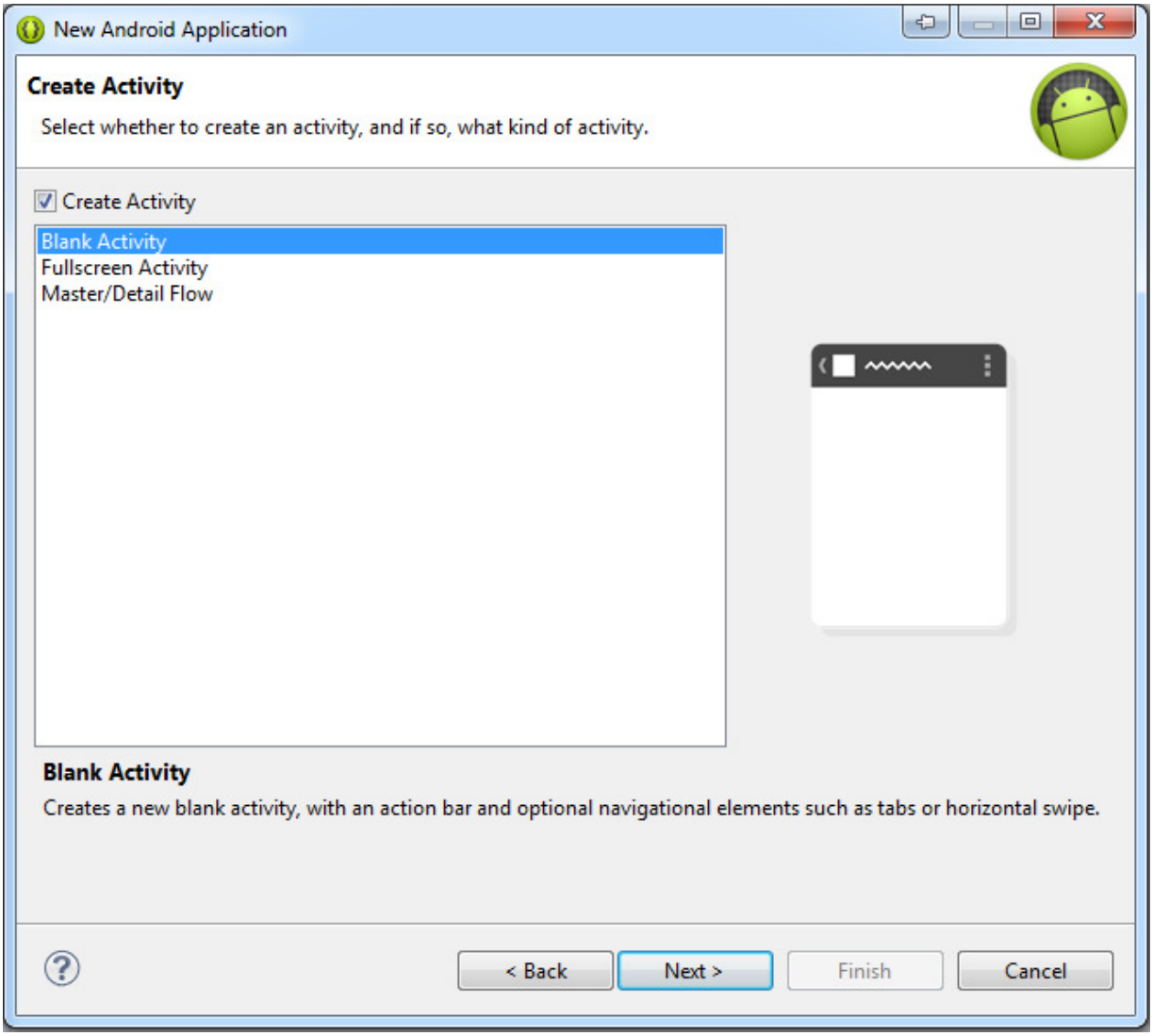
Εικόνα 4.2 - Δημιουργία ενός Android Application

Ακολούθως εμφανίζεται το παράθυρο Configure Launcher Icon στο οποίο μπορούμε να επιλέξουμε το εικονίδιο με το οποίο θα εμφανίζεται η εφαρμογή στην οθόνη της συσκευής μας, καθώς και άλλες ιδιότητες του όπως το χρώμα ή το σχήμα. Κατόπιν πατάμε το κουμπί Next.



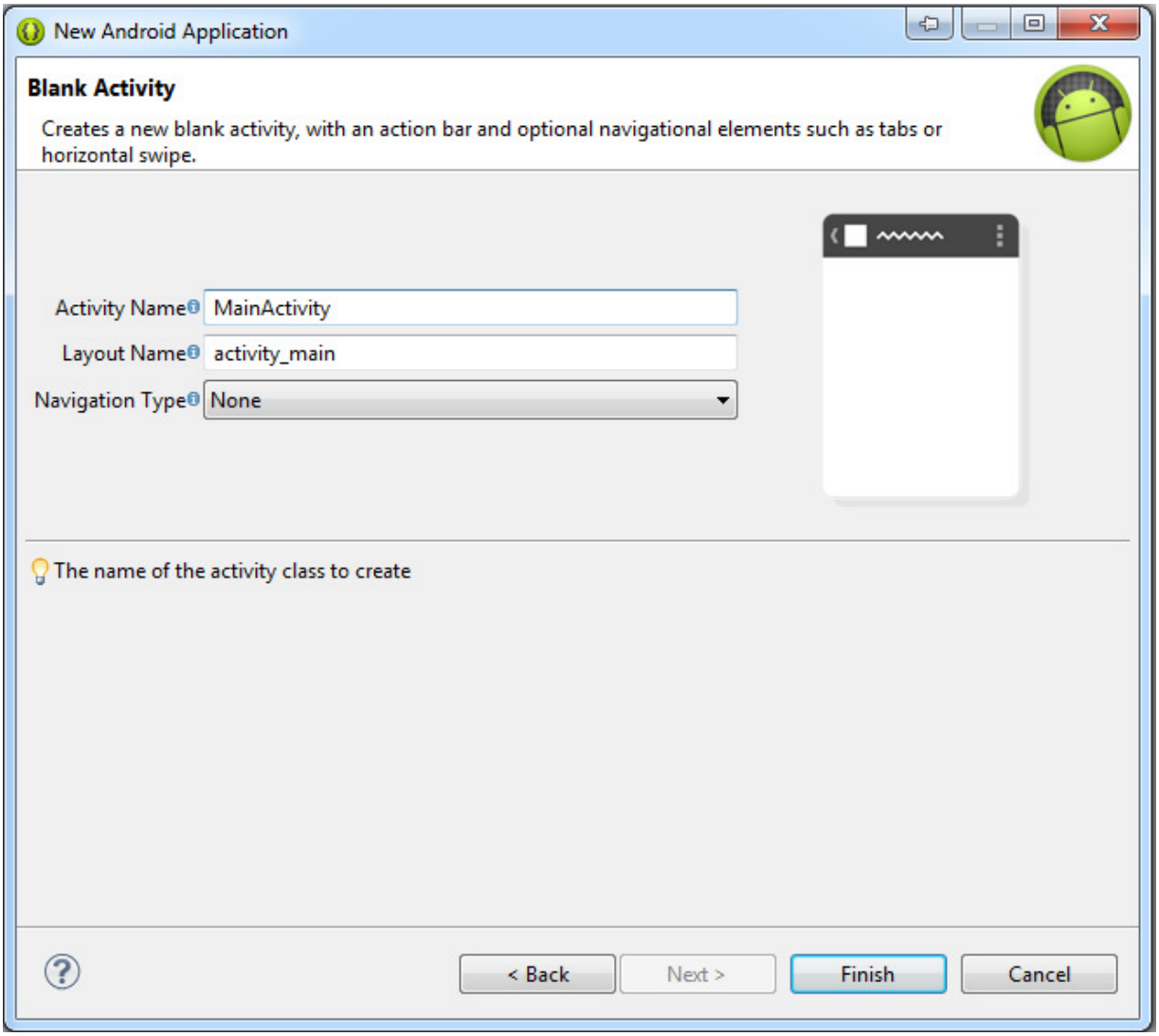
Εικόνα 4.3 - Επιλογή εικονιδίου εφαρμογής

Στο παράθυρο που εμφανίζεται μας δίνεται τη δυνατότητα να επιλέξουμε μέσω του check box αν θέλουμε να δημιουργηθεί αυτόματα μια νέα Activity. Στη συνεχεία πατάμε Next.



Εικόνα 4.4 - Δημιουργία Κλάσης

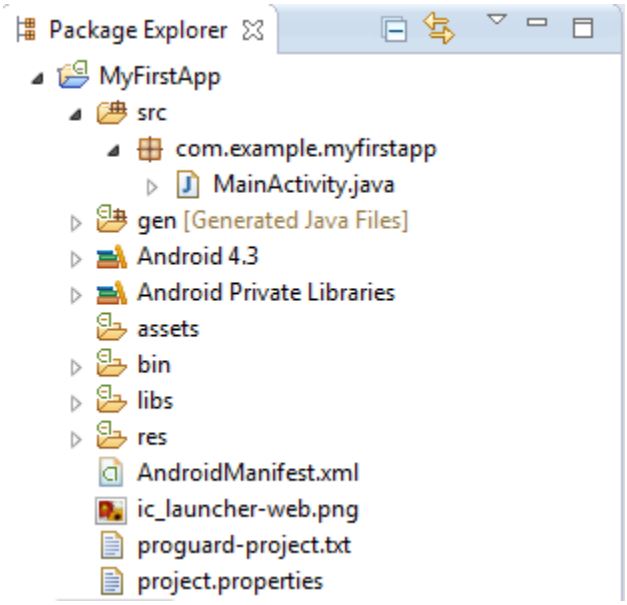
Στη συνέχεια εμφανίζεται το παράθυρο New Blank Activity στο οποίο ορίζουμε το όνομα που θέλουμε να έχει η Activity που δημιουργήσαμε καθώς και το όνομα του αντίστοιχου layout στο οποίο αναφέρεται.



Εικόνα 4.5 - Επιλογή ονομάτων κλάσεων

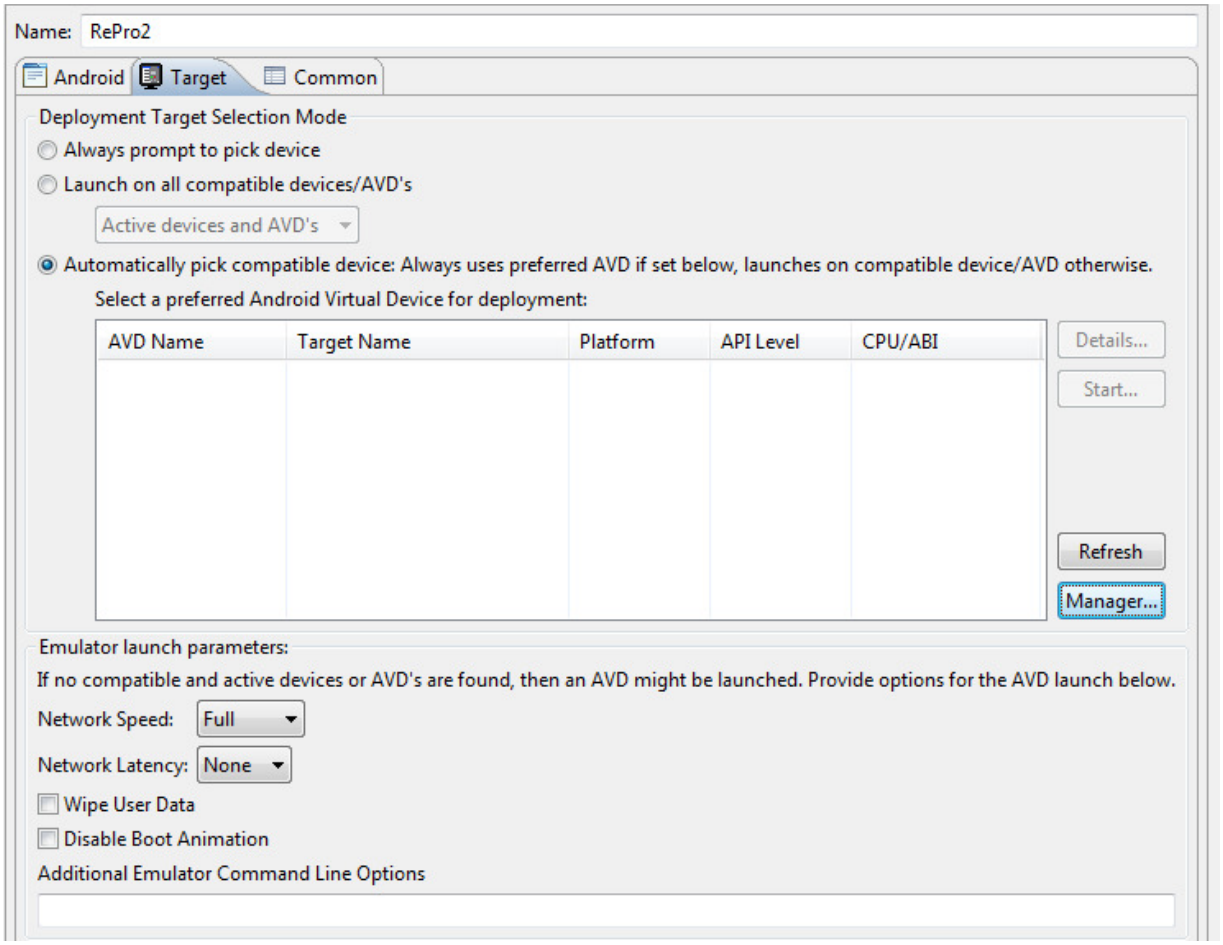
Τέλος, όταν συμπληρωθούν τα αναγκαία πεδία ενεργοποιείται το κουμπί Finish με το οποίο ολοκληρώνεται η διαδικασία δημιουργίας νέου project.

Μετά το τέλος της διαδικασίας, μπορούμε να δούμε αριστερά στον Package Explorer του Eclipse τα αρχεία από τα οποία αποτελείται το project μας.



Εικόνα 4.6 - Package explorer

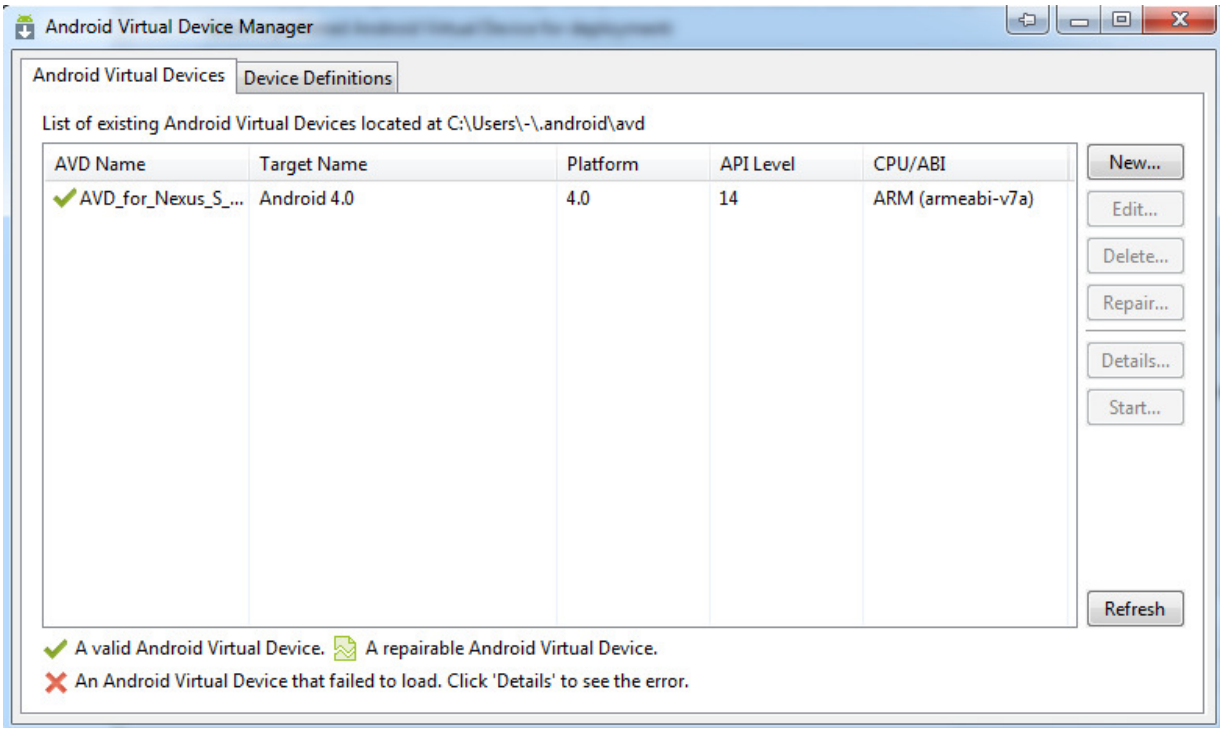
Η εκτέλεση και δοκιμή της εφαρμογές μπορεί να γίνει σε εξομοιωτή (emulator) στο περιβάλλον ανάπτυξης, ο όποιος προσομοιώνει μια Android συσκευή ή κατευθείαν σε Android συσκευή. Αυτό είναι πολύ χρήσιμο γιατί ο προγραμματιστής μπορεί να αναπτύξει, αλλά και ταυτόχρονα να δοκιμάζει τις Android εφαρμογές αν τρέχουν σωστά, χωρίς να διαθέτει Android συσκευή.



Εικόνα 4.7 - AVD

Ο εξομοιωτής Android που μπορεί να προσομοιώσει αρκετές προεγκατεστημένες συσκευές διαφόρων εκδόσεων, αλλά και δηλωμένων από τον προγραμματιστή ονομάζεται Dalvik Virtual Machine (DVM). Ο DVM είναι μια εικονική μηχανή που επιτρέπει την μέγιστη δυνατή απόδοση της εφαρμογής στην εικονική συσκευή, αφού οι πόροι που χρησιμοποιούνται είναι οι ίδιοι μ' αυτούς που χρησιμοποιούνται εσωτερικά στην αληθινή συσκευή. Κάθε φορά που μια καινούργια εφαρμογή ξεκινάει, ο DVM δημιουργεί ένα νέο αντίγραφο του εαυτού του το οποίο αντιστοιχείται στην συγκεκριμένη εφαρμογή, με όλους τους διαθέσιμους πόρους του συστήματος. Γι' αυτό το λόγο καμία εφαρμογή δεν έχει επαφή με την άλλη, παρόλο που μπορεί να εκτελούνται ταυτόχρονα. Από εκεί και πέρα ο DVM παίρνει όλες τις Java εντολές της εφαρμογής, τον XML κώδικα, μαζί με οποιαδήποτε άλλα αρχεία εικόνας και ήχου που

μπορεί να υπάρχουν, και τα μεταφράζει σε δυαδικό κώδικα μηχανής (bytecode) για τον “καταλάβει” η Android πλατφόρμα και να τον εκτελέσει [11].



Εικόνα 4.8 - AVD manager

ΚΕΦΑΛΑΙΟ 5

Ανάλυση μεθοδολογίας προγραμματισμού με την χρήση του Android SDK

Σκοπός

Σε αυτό το Κεφάλαιο θα ασχοληθούμε με:

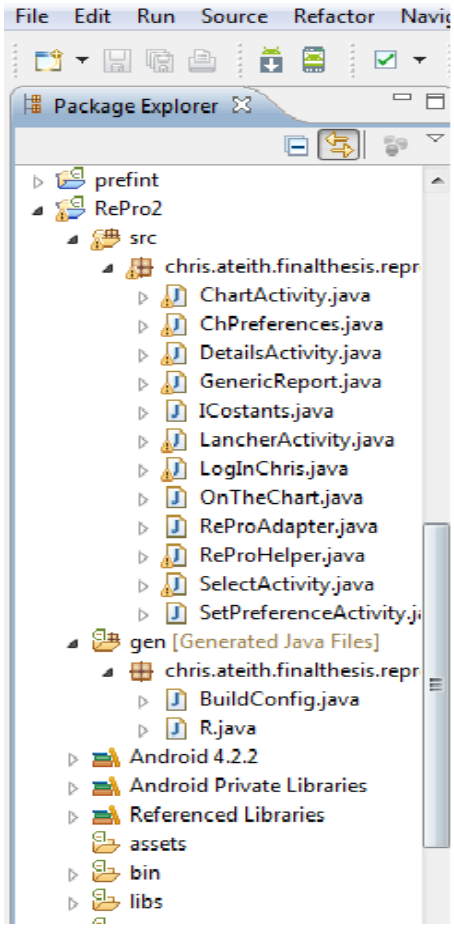
- την περιγραφή του Android SDK.
- την παρουσίαση της δομής ενός προγράμματος για την δημιουργίας μιας εφαρμογής για android κινητά.
- Τον διαχωρισμό της δομής σε Frontend και Backend.
- την περιγραφή με παραδείγματα του Frontend .
- και τη περιγραφή με παραδείγματα του Backend .
- Περιγραφή του τρόπου αποθήκευσης δεδομένων.
- επίσης θα κάνουμε μια αναφορά στην εισαγωγή βιβλιοθηκών σε προγράμματα
- και θα περιγράψουμε πως το πρόγραμμα που δημιουργήσαμε μπορεί να δημοσιευτεί στο Google Play.

Προσδοκώμενα αποτελέσματα. Ο αναγνώστης:

- να κατανοήσει πλήρως τον τρόπο με τον οποίο δομείται ένα πρόγραμμα για android κινητά .
- να μπορεί να χρησιμοποιεί τα διαγράμματα της Ενοποιημένης Γλώσσας
- να μπορεί να αναπτύσσει τη γραφική διασύνδεση χρήστη
- και να μπορεί να αξιοποιεί τις δυνατότητες εργαλείων υποστήριξης της διαδικασίας ανάπτυξης.
- Να είναι σε θέση να δημοσιεύει τις εφαρμογές του στην πλατφόρμα της Google.

Παρουσίαση

Σκοπός του κεφαλαίου αυτού είναι η παρουσίαση των σταδίων ανάπτυξης, μιας εφαρμογής και ειδικότερα το πώς δομείται ένας κώδικας ο οποίος προορίζεται για Android εφαρμογές. Το Κεφάλαιο αυτό έχει ως στόχο αφενός να περιγράψει θεωρητικά τη μεθοδολογία, εστιάζοντας πάντα στην εφαρμογή που δημιουργήθηκε ως μέρος της διπλωματικής εργασίας, αφετέρου να παρουσιάσει και να επεξηγήσει ως μέρος παραδειγμάτων τα ποιο σημαντικά κομμάτια του κώδικα που υλοποιήσαμε. Για να γίνει η υλοποίηση όλων των πάνω χρησιμοποιήθηκε ως λογισμικό το eclipse καθώς επίσης έγινε εγκατάσταση του Android SDK TOOL.

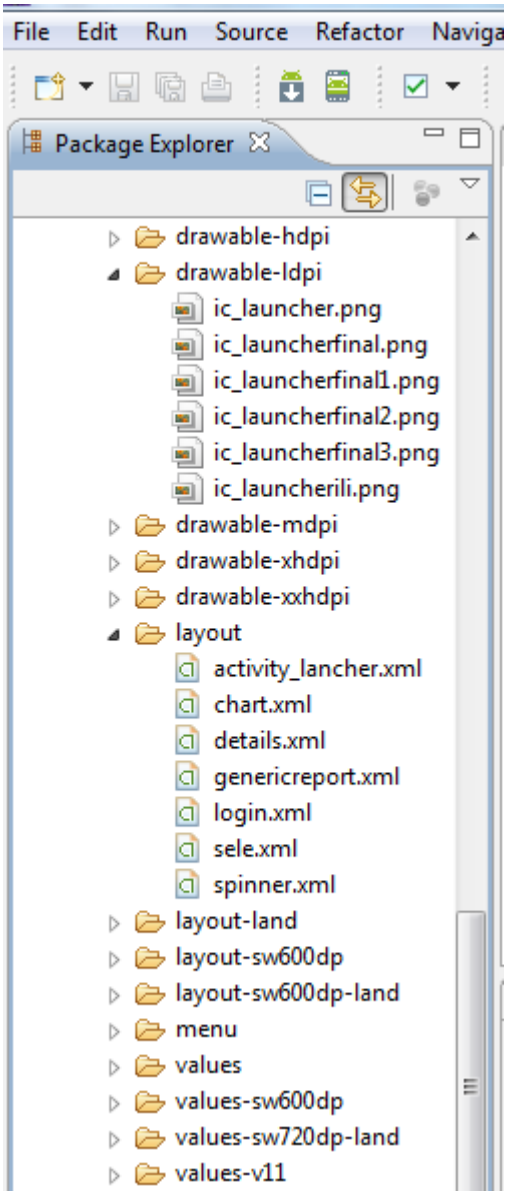


Εικόνα 5.1

5.1. Ανατομία μιας Android εφαρμογής

Βασική προϋπόθεση για να μπορέσουμε να προχωρήσουμε στην περιγραφή της δομής του προγράμματος μας είναι η κατανόηση της ανατομίας που έχει ένα πρόγραμμα, που προορίζεται για android κινητά. Μπορούμε να παρατηρήσουμε στην εικόνα 5.1, ότι έχουμε μια πληθώρα αρχείων τα οποία αποτελούν ένα Android πρόγραμμα. Οι διάφοροι φάκελοι με τα αρχεία τους είναι οι ακόλουθοι:

- Το src- Περιέχει τα .java αρχεία του κώδικα.
- Το gen- Περιέχει τα R.java αρχεία τα οποία δημιουργούνται κατά την εκτέλεση και απευθύνονται στα αρχεία που χρησιμοποιεί το πρόγραμμα.
- Android.4.4 βιβλιοθήκη – το αντικείμενο αυτό περιέχει ένα αρχείο android.jar, το οποίο περιλαμβάνει όλες τις κλάσεις βιβλιοθηκών που χρειάζονται για την ανάπτυξη μιας android εφαρμογής.
- Assets – περιλαμβάνει όλα τα αρχεία όπως HTML, αρχεία κειμένου και βάσεις δεδομένων τα οποία χρησιμοποιούνται από την εφαρμογή μας.
- Bin – περιλαμβάνει τα αρχεία τα οποία δημιουργήθηκαν από το ADT. Πιο συγκεκριμένα το αρχείο .apk, το οποίο είναι ένα δυαδικό αρχείο που περιλαμβάνει οτιδήποτε χρειάζεται η εφαρμογή για να εκτελεστεί.



Εικόνα 5.2

- Res – περιλαμβάνει όλα τα resources τα οποία χρησιμοποιεί η εφαρμογή. Επίσης περιλαμβάνει και μερικούς υποφακέλους για παράδειγμα drawable-<*resolution*>, layout, και values, όπως φαίνεται στην εικόνα 5.2.
- AndroidManifest.xml – είναι ίσως το πιο σημαντικό κομμάτι του κώδικα, στο οποίο περιγράφονται οι άδειες της εφαρμογής, δηλώνονται οι κλάσεις και καθορίζονται οι συσκευές οι οποίες μπορούν να χρησιμοποιήσουν την εφαρμογή.

5.2. Το αρχείο AndroidManifest.xml [3]

Στην ενότητα αυτήν θα μιλήσουμε για το τελευταίο θέμα με το οποίο κλείσαμε την προηγούμενη ενότητα, και ίσως αν όχι το πιο σημαντικό ένα από τα σημαντικότερα αρχεία του κώδικα ενός προγράμματος για android κινητά, το αρχείο androidManifest.xml. Στην δικιά μας περίπτωση ο κώδικας του αρχείου φαίνεται παρακάτω.

```
<?xmlversion="1.0"encoding="utf-8"?>

<manifestxmlns:android="http://schemas.android.com/apk/res/android"
package="chris.ateith.finalthesis.repro2"
android:versionCode="1"
android:versionName="1.0"
android:installLocation="auto">

<uses-sdk
android:minSdkVersion="11"
android:targetSdkVersion="17"/>

<application
android:allowBackup="true"
android:icon="@drawable/ic_launcherfinal3"
android:label="@string/app_name"
android:theme="@style/AppTheme">
<activity
android:name="chris.ateith.finalthesis.repro2.LancherActivity"
android:label="@string/app_name"
android:theme="@android:style/Theme.Black.NoTitleBar">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>

<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activityandroid:name="chris.ateith.finalthesis.repro2.LogInChris"/>
<activityandroid:name="chris.ateith.finalthesis.repro2.ChPreferences"/>
<activityandroid:name="chris.ateith.finalthesis.repro2.SelectActivity"/>
<activityandroid:name="chris.ateith.finalthesis.repro2.SetPreferenceActivity"/>
<activityandroid:name="chris.ateith.finalthesis.repro2.DetailsActivity"/>
<activityandroid:name="chris.ateith.finalthesis.repro2.GenericReport"></activity>
</application>

</manifest>
```

Το androidManifest.xml περιλαμβάνει λεπτομερείς πληροφορίες σχετικά με την εφαρμογή όπως:

- Καθορίζει το όνομα του πακέτου της εφαρμογής, στην συγκεκριμένη περίπτωση είναι το *chris.ateith.finalthesis.repro2.*
- Την έκδοση του κώδικα της εφαρμογής, ο οποίος στην περίπτωση μας είναι android:versionCode=*"1".* Ο αριθμός αυτός δείχνει στην συσκευής την έκδοση της εφαρμογής και αν αυτή πρέπει αναβαθμιστεί.
- Το όνομα της έκδοσης της εφαρμογής που είναι 1.0 και ορίζεται με android:versionName=*"1.0".*Αυτό το νούμερο ή διαφορετικά το όνομα είναι αυτό που βλέπει ο χρήστης.
- Το android:minSdkVersion=*"11"*, που στην περίπτωση μας είναι 11. Ο αριθμός αυτός καθορίζει την ελάχιστη έκδοση του λειτουργικού συστήματος στην οποία μπορεί να εγκατασταθεί η εφαρμογή.
- Το android:targetSdkVersion=*"17"*, είναι η έκδοση του λειτουργικού για την οποία προορίζεται η εφαρμογή, χωρίς βέβαια αυτό να σημαίνει ότι δεν μπορεί να εγκατασταθεί και σε νεότερες εκδόσεις. Γεγονός είναι πως πολλοί προγραμματιστές επιλέγουν να μην το συμπεριλάβουν στο manifest της εφαρμογής τους.
- Το android:icon=*"@drawable/ic_launcherfinal3"*, είναι αυτό που καθορίζει το εικονίδιο που υπάρχει στην επιφάνεια με τις εφαρμογές, και το οποίο στην περίπτωση μας είναι τοποθετημένο στον φάκελο drawable.
- Το όνομα μια εφαρμογής, δηλαδή έτσι όπως θα φαίνεται στον χρήστη που θα την κάνει εγκατάσταση, καθορίζεται με android:label=*"@string/app_name"*, όπου app_name είναι μια σταθερά τύπου string η οποία κρατά το όνομα της εφαρμογής και βρίσκεται μέσα στο αρχείο string.xml.

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>

<stringname="app_name">RePro</string>
<stringname="hello_world">Hello world!</string>
<stringname="menu_settings">Settings</string>
</resources>
<stringname="menu_settings">Settings</string>

</resources>
```

Παρατηρώ ότι η μεταβλητή app_name έχει ως τιμή το RePro, που είναι και το όνομα της εφαρμογής που δημιουργήθηκε στα πλαίσια της διπλωματικής.

Για να κάνουμε χρήση ενός από τα θέματα που παρέχει το android ή για να εφαρμόσουμε ένα από αυτά που οι ίδιοι καθορίσαμε, χρησιμοποιούμε το android:theme="@android:style/Theme.Black.NoTitleBar".

Αυτό που ακολουθεί στην συνέχεια είναι ο ορισμός των Activities, που αποτελούν τις διάφορες οθόνες ή παράθυρα της εφαρμογής και συνήθως είναι κλάσεις της εφαρμογής. Στην προκειμένη περίπτωση έχουμε 7 τέτοιες κλάσεις. Ο τρόπος που περιγράφω κάθε μια από αυτές είναι ο ακόλουθος:

```
<activity
android:name="chris.ateith.finalthesis.repro2.LancherActivity"
android:label="@string/app_name"
android:theme="@android:style/Theme.Black.NoTitleBar">
<intent-filter>
<actionandroid:name="android.intent.action.MAIN"/>

<categoryandroid:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
```

Στην παραπάνω περίπτωση διαλέξαμε να παρουσιάσουμε τον ορισμό της αρχικής Activity, η οποία είναι η οθόνη εκκίνησης της εφαρμογής. Με το android:name="chris.ateith.finalthesis.repro2.LancherActivity" , ορίζω το όνομα της το οποίο είναι LancherActivity, και παρατηρώ ότι χρησιμοποίησα το πλήρες όνομα του πακέτου της εφαρμογής για τον ορισμό του ονόματος, πράγμα που δεν είναι

απαραίτητο. Στην συνέχεια βλέπω χαρακτηριστικά τα οποία τα περιγράψαμε και προηγουμένως τα οποία μπορούν να παραγοντοποιήσουν κάθε Activity ξεχωριστά.

- Κατά τον προσδιορισμό της παραπάνω Activity βλέπω ότι υπάρχει ένα στοιχείο το οποίο ονομάζεται <intent-filter>:
- Η δράση του φίλτρου αυτό προσδιορίζεται ως <actionandroid:name=*"android.intent.action.MAIN"*/>, που σημαίνει ότι η activity αυτή χρησιμοποιείται ως το σημείο εισόδου της εφαρμογής.
- Η κατηγορία του φίλτρου ορίζεται android:name=*android.intent.category.LAUNCHER,* με τον τρόπο αυτό μας λέει ότι μπορεί να ξεκινάει από το εικονίδιο που βρίσκεται στην επιφάνεια εργασίας.

Με τα παραπάνω περιγράψαμε με τρόπο λεπτομερή το βασικότερο αρχείο εφαρμογής που είναι το AndroidManifest.xml. Παρακάτω θα προχωρήσουμε στην περιγραφή ενός αρχείου που θα μας βοηθήσει να καταλάβουμε πως δομείται το user interface της εφαρμογής, που για λόγους συντομίας θα αναφερόμαστε σε αυτό με τα αρχικά UI.

5.3. Τρόπος ορισμού του User Interface (UI)

Παρακάτω θα περιγράψουμε μέρος του κώδικα ο οποίος εκτελεί τον ρόλο αυτό, δηλαδή καθορίζει την εμφάνιση μιας Activity, για λεπτομερέστερη περιγραφή όλων των αντικειμένων που κάνει χρήση το Android και τα οποία ονομάζονται views, μπορούμε να ενημερωθούμε από πληθώρα βιβλίων που υπάρχουν στην αγορά, μιας και ο σκοπός του κειμένου αυτού δεν είναι η δημιουργία ενός εγχειριδίου, αλλά η περιγραφή μέρους του κώδικα που αναπτύχθηκε για την διπλωματική εργασία.

Όλα τα αρχεία που έχουν να κάνουν με το UI της εφαρμογής βρίσκονται μέσα στον φάκελο res, και το καθένα ειδικότερα μέσα σε έναν υποφάκελο που ξεκινάει το όνομα με layout. Στο σημείο αυτό θα ανοίξουμε μια παρένθεση , για να τονίσουμε ότι μπορούν να υπάρχουν πολλοί φάκελοι που να αρχίζουν με το layout, κάθε ένας από αυτούς αποθηκεύει αρχεία κατάλληλα για διαφορετικές συσκευές. Για παράδειγμα ο layout-land, περιέχει αρχεία τα οποία χρησιμοποιούνται όταν η συσκευή περιστρέφεται.

Εικόνα 5.3

Το layout-sw600dp και layout-sw600dp-land, είναι αποκλειστικά για συσκευές με μικρότερη διάμετρο 600dp, που όπως είναι αντιληπτό είναι τα Tablets.

Στην συνέχεια θα παραθέσουμε αρχείο κώδικα, το οποίο είναι υπεύθυνο για την διαμόρφωση του UI, της Activity με το όνομα genericReport, η περιγραφή της οποίας θα ακολουθήσει σε επόμενη παράγραφο.

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
>
<ScrollView
android:id="@+id/scroll"
android:layout_height="fill_parent"
android:layout_width="fill_parent"
android:fillViewport="true">
<RelativeLayout
android:id="@+id/scrcont"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
>
<TextView
android:id="@+id/rephead"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/headrep"
android:textSize="15sp"
android:textStyle="bold"
android:textColor="@color/background"
android:padding="5dp"
/>
<TextView
android:id="@+id/rephead2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/headrep1"
android:layout_below="@+id/rephead"
android:textSize="12sp"
android:padding="5dp"
android:layout_marginTop="5dp"/>
<LinearLayout
android:id="@+id/div"
android:layout_width="fill_parent"
android:layout_height="2dp"
android:background="@color/background"
android:layout_below="@+id/rephead2"
android:padding="5dp"
android:layout_marginTop="3dp"
android:layout_marginLeft="5dp"
android:layout_marginRight="5dp"
android:orientation="horizontal"></LinearLayout>
<!-- to scroll gianaxwresei to periexomenotou fragment -->

<RelativeLayout
android:id="@+id/textcoll"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:layout_below="@+id/div"
    android:padding="5dp"
    >
    <TextView
    android:id="@+id/selese"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="13sp"
    android:text="@string/selese"
    android:textStyle="bold"
    android:layout_marginTop="10dip"
    android:layout_alignParentLeft="true"/>
    <TextView
    android:id="@+id/seleles"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/selele"
    android:textStyle="bold"
    android:textSize="13sp"
    android:layout_below="@+id/selese"
    android:layout_marginTop="20dip"
    android:layout_alignParentLeft="true"/>
    </RelativeLayout>
    <RelativeLayout
    android:id="@+id/spinners"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_toRightOf="@+id/textcoll"
    android:layout_alignBaseline="@+id/textcoll"
    android:layout_below="@+id/div"
    android:padding="5dp"
    >
    <Spinner
    android:id="@+id/semestersele"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_marginTop="5dp"
    />
    <Spinner
    android:id="@+id/semesterle"
    android:layout_height="wrap_content"
    android:layout_width="match_parent"
    android:layout_below="@+id/semestersele"
    android:layout_alignParentRight="true"
    android:layout_marginTop="7dip"
    />
    </RelativeLayout>
    <Button
    android:id="@+id/subComm"
    android:layout_height="wrap_content"
```

```
    android:layout_width="wrap_content"
    android:layout_below="@+id/spinners"
    android:layout_alignParentRight="true"
    android:padding="8dip"
    android:text="@string/report"
    android:layout_marginTop="10dip"
    android:textSize="12sp"
    android:textStyle="bold"
    />
    <FrameLayout
    android:id="@+id/statist"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_marginTop="20dp"
    android:background="@color/white"
    android:layout_below="@+id/subComm"></FrameLayout>

    </RelativeLayout><!--telosscrcont -->
    </ScrollView>
    </RelativeLayout>
```

Ο παραπάνω κώδικας αποτελεί το αρχείο με το όνομα genericeport.xml, ο οποίος βρίσκεται μέσα στον φάκελο res/layout. Όπως τονίσαμε και προηγουμένως σκοπός μας δεν είναι να δημιουργήσουμε άλλο ένα εγχειρίδιο, αλλά να περιγράψουμε ορισμένα σημεία του κώδικα που έχουν ενδιαφέρον για τον αναγνώστη.

Κάνοντας μια ανάγνωση του κώδικα από πάνω προς τα κάτω παρατηρούμε τα εξής:

Με μια πρώτη ματιά βλέπουμε ότι ένα αρχείο υπεύθυνο για την διαμόρφωση του UI μιας Activity, είναι εξολοκλήρου φτιαγμένο με XML.

- <RelativeLayout/> -είναι αυτό που στην ορολογία του android προγραμματισμού ονομάζεται ViewGroup. Αποτελεί μια συλλογή από αντικείμενα (κουμπιά, πεδία κειμένου κ.τ.λ.π). Κοινώς είναι το πλαίσιο μέσα στο οποίο οργανώνουμε άλλα απαραίτητα στοιχεία για το UI. Ως ViewGroup επίσης έχουμε :
- AbsoluteLayout
- LinearLayout
- TableLayout
- FrameLayout
- ScrollView

Παρατηρώντας βλέπουμε ότι το RelativeLayout έχει και ορισμένα χαρακτηριστικά:

- android:layout_width=”match_parent” – καθορίζω το μήκος του στοιχείου να είναι σε πλήρη αντιστοιχία με το πατρικό.
- android:layout_height=’’match_parent’’- το ίδιο με προηγουμένως και για το ύψος.

Στην συνέχεια συναντάμε κάποια στοιχεία τα οποία ονομάζονται views, και αναλυτικά είναι:

- <TextView> - χρησιμεύει για να παρουσιάζει κείμενο στον χρήστη.
- <Button>- μας παρουσιάζει ένα κουμπί.
- <Spinner> - είναι μια λίστα , η οποία εμφανίζεται και μας δίνει την δυνατότητα πολλαπλών επιλογών, όπως φαίνεται παρακάτω.

Εικόνα 5.4

Στο σημείο αυτό θα θέλαμε να κάνουμε μια παρένθεση για να περιγράψουμε πολύ βασικά στοιχεία τα οποία είναι απαραίτητα να κατανοήσουμε τόσο στην δημιουργία του UI, όσο και κατά την διαδικασία υλοποίησης του κώδικα της εφαρμογής μας.

5.4. Κατανόηση μονάδων μέτρησης σχετικές με τις διαστάσεις της συσκευής

Μεγάλη σημασία έχει να καταλάβουμε ότι ο καθορισμός, ενός ομοιομόρφου UI αποτελεί μια δύσκολη υπόθεση για τον προγραμματισμό για Android συσκευές. Αυτό γιατί όπως όλοι μας έχουμε παρατηρήσει με μια βόλτα στα καταστήματα ηλεκτρικών ειδών, υπάρχει μια πληθώρα συσκευών τα οποία χρησιμοποιούν το Android λειτουργικό. Αποτέλεσμα του γεγονότος αυτού είναι η δυσκολία προσαρμογής του UI έτσι ώστε να έχουμε μια ομοιομορφία στην εμφάνιση.

Επιστρέφοντας στον καθορισμό του μήκους ή του ύψους, ενός αντικειμένου, μπορούμε να χρησιμοποιήσουμε διάφορες μονάδες μέτρησης, όπως:

Εικόνα 5.5

- dp – ή densityIndependentpixel – αυτή αποτελεί την ενδεικνυόμενη μονάδα μέτρησης ενός αντικειμένου .
- sp – Scale-independentpixel – είναι παρόμοιο με το dp, και χρησιμοποιείται στον καθορισμό του μεγέθους γραμματοσειράς.
- pt – Point – ένα point είναι το 1/72 της ίντσας.
- px – Pixel – αποτελεί το pixel της οθόνης. Η χρήση της μονάδας αυτής δεν ενδείκνυται αφού οι συσκευές που θα κάνουν χρήση της εφαρμογής διαφέρουν, ως προς την ανάλυση της οθόνης.

Στην εικόνα 5.5 έχουμε μια συσκευή Nexus S. Έχει μια οθόνη 4’, μήκος οθόνης 2.04’ .Η ανάλυση της συγκεκριμένης συσκευής είναι 480(μήκος) X 800(ύψος). Το pixeldensity υπολογίζεται εάν διαιρέσουμε το μήκος 480 px με το μήκος σε inch που

είναι 2.04, το οποίο μας δίνει ανάλυση ίση με 235dpi. Όπως μπορούμε να καταλάβουμε από το παράδειγμα, η pixeldensity μιας συσκευής διαφέρει ανάλογα με το μέγεθος της οθόνης και την ανάλυση .

Το Android, όπως αναφέραμε και προηγουμένως, καθορίζει τέσσερις αναλύσεις οθόνης:

- Lowdensity (*ldpi*) — 120 dpi
- Mediumdensity (*mdpi*) — 160 dpi
- Highdensity (hdpi) — 240 dpi
- Extra High density (*xhdpi*) — 320 dpi

Για να μπορέσουν βέβαια όλα αυτά να έχουν νόημα και να είναι ορατά στον χρήστη θα πρέπει να πάρουν ζωή μέσα από τον κώδικά.

5.5. Καθορισμός των χρωμάτων

Όπως και σε όλα τα πράγματα της ζωής το μάτι παίζει το μεγαλύτερο ρόλο όταν πρέπει να επιλέξουμε μέσα από μια πληθώρα, έτσι και σε μια εφαρμογή ο καθορισμός των χρωμάτων έχει πρωτεύοντα ρόλο στην επιτυχία της. Τι σημαίνει όμως καθορισμός χρωμάτων;

Κάθε αντικείμενο έχει ένα στοιχείο μέσω του οποίου μπορούμε να καθορίσουμε το χρώμα του. Για να καθορίσουμε το χρώμα πρέπει να κάνουμε χρήση του χαρακτηριστικού android:background = “@color/white”. Πώς μπορεί όμως να ερμηνευτεί αυτή η γραμμή του κώδικα; Από τα συμφραζόμενα μπορεί να καταλάβει κάποιος ότι δίνουμε στο περίγραμμα του αντικειμένου χρώμα άσπρο. Αυτό βέβαια είναι εν μέρη ακριβές .

Όλα τα χρώματα σε μια android εφαρμογή καθορίζονται μέσα σε ένα xml αρχείο το οποίο ονομάζεται colors και βρίσκεται μέσα στο values, values/colors.xml, και έχει την μορφή που φαίνεται παρακάτω.

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
<colorname="background">#2151ed</color>
<colorname="background2">#428cf1</color>
<colorname="white">#ffffff</color>
<colorname="yellow">#e8bb20</color>
<colorname="blk">#000000</color>
<colorname="gray">#d4d2d2</color>
<colorname="graylight">#f6f6f6</color>
</resources>
```

Παρατηρώντας τον παραπάνω κώδικα βλέπω ότι για να καθορίσω ένα χρώμα θα πρέπει να χρησιμοποιήσω το αντικείμενο <color/> , μέσα στο οποίο θα πρέπει να ορίσω ένα όνομα όποιο εγώ θέλω name=”white”, μπορώ να βάλω όποιο όνομα εγώ θέλω για να περιγράψω ένα χρώμα . Πολύ βασικό βέβαια είναι το ίδιο το χρώμα το οποίο το καθορίζω με μια 16 μορφή τύπου #ffffff , για το χρώμα άσπρο για παράδειγμα.

5.6. Κώδικας υλοποίησης της λειτουργικότητας του προγράμματος

Στην προηγούμενη ενότητα, πήραμε μια γεύση για το πώς δομείται το UI μιας εφαρμογής. Όλα αυτά βέβαια δεν θα είχαν νόημα αν δεν μπορούσαμε να τους δώσουμε λειτουργικότητα μέσα από τον κώδικα της εφαρμογής, ο οποίος είναι φτιαγμένος με Java.

Παρακάτω θα περιγράψουμε αναλυτικά κομμάτι του κώδικα της RePro εφαρμογής που κατασκευάσαμε στα πλαίσια της διπλωματικής εργασίας.

5.7. Τι σημαίνει και πώς ορίζεται μια Activity

Η υποενότητα αυτή αρχίζει με το πώς μπορούμε να ορίσουμε μια activity, η οποία αποτελεί και το βασικό δομικό κομμάτι ενός προγράμματος για android κινητά. Για να δημιουργήσουμε μια Activity, θα πρέπει να κατασκευάσουμε μια κλάση σε Java η οποία θα επεκτείνεται στην μητρική κλάση Activity.

```
package chris.ateith.finalthesis.repro2;

importjava.util.Calendar;

importandroid.app.Activity;
importandroid.content.Intent;
importandroid.content.res.Resources;
importandroid.database.Cursor;
importandroid.os.Bundle;
importandroid.util.Log;
importandroid.view.Menu;
importandroid.view.MenuItem;
importandroid.view.View;
importandroid.view.View.OnClickListener;
importandroid.widget.AdapterView;
importandroid.widget.ArrayAdapter;
importandroid.widget.Button;
importandroid.widget.Spinner;
importandroid.widget.Toast;
importandroid.widget.AdapterView.OnItemSelectedListener;

public class SelectActivity extends Activity  implements
OnItemSelectedListener{
int month, weekofyear, semeofschool;
Spinner semeS, semeW, semeL;
String[]semester;
```

```
String week[];
String week1[];
String weeknull[];
String lesson[];
String lesson1[];
Button subB;

Resources res;
ArrayAdapter<String> adapter2;
ArrayAdapter<String> adapter3;
intpos;
privateReProAdapterdbHelper;

//-- orizw tis metablhtes pou 8a apo8hkeuw tis --
//-- epiloges tou xrhsth sta spinner           --

String semeSele, weekSele, lessonSele;
```

Παρατηρώντας τον πιο πάνω κώδικα, βλέπουμε ότι μια Activity ξεκινάει με το όνομα του πακέτου της εφαρμογής, και στην συνέχεια με το import εισάγουμε όλες τις κλάσεις του συστήματος τις οποίες θα χρησιμοποιήσουμε στην Activity μας. Στην συνέχεια ορίζουμε όπως αναφέραμε προηγουμένως μια Activity, με το όνομα SelectActivity, η οποία δεν είναι τίποτα άλλο από μια κλάση σε Java η οποία επεκτείνεται στην υπερκλάση Activity (publicclassSelectActivityextendsActivity). Η συγκεκριμένη κλάση βέβαια κάνει implementOnItemSelectedListener, για να μπορούμε να ακούμε και να υλοποιούμε διάφορα κλίκ πάνω στην οθόνη του κινητού μας. Στην συνέχεια ορίζουμε όλες τις μεταβλητές και τα διάφορα αντικείμενα τα οποία θα χρησιμοποιήσουμε στην συνέχεια. Όπως αναφέραμε προηγουμένως κατά την περιγραφή του UI, το UI της συγκεκριμένης Activity (sele.xml) αποτελείται από τρία spinner και ένα button. Τα αντικείμενα αυτά για να πάρουν ζωή και λειτουργικότητα θα πρέπει να οριστούν μέσα στην Activity μας ως εξής :

- Spinner semeS, semeW, semeL;
- Button subB;

Διαγράφοντας το πρόγραμμα προς τα κάτω συναντάμε κάτι πολύ ενδιαφέρον το οποίο ορίζεται ως (ArrayAdapter<String>adapter2;). Μέσα στο αντικείμενο είναι ένα String το οποίο θα επικολληθεί σε κάθε ένα από τα Spinner που ορίσαμε και θα το αναλύσουμε παρακάτω.

5.8. Spinner, πώς του δίνουμε ζωή

Όπως έχουμε πει ο Spinner είναι μια dropdown λίστα από την οποία ο χρήστης μπορεί να επιλέξει ένα αντικείμενο. Πώς όμως προσθέτω δεδομένα που θα εμφανίζονται, στην dropdown επιλογή ενός Spinner;

Οι τιμές οι οποίες θα επικολληθούν, βρίσκονται σε ένα xml αρχείο το οποίο ονομάζεται arrays.xml και βρίσκεται μέσα στο φάκελο values/arrays.xml. Τι είναι όμως αυτό το νέο αρχείο; Όπως καταλαβαίνουμε από το όνομα του αποθηκεύει πίνακες με την μορφή xml, όπως μπορείτε να δείτε και μόνοι σας στον παρακάτω κώδικα.

```
<?xmlversion="1.0"encoding="utf-8"?>
<resources>
<arrayname="Semester">
<item>Εαρινό</item>
<item>Χειμερινό</item>
</array>
<arrayname="Semester2">
<item>ΕπιλέξτεΕξάμηνο</item>
<item>Εαρινό</item>
<item>Χειμερινό</item>
</array>
<arrayname="Week">
<item>1: 1/1/2013</item>
<item>2: 6/1/2013</item>
<item>3: 13/1/2013</item>
<item>4: 20/1/2013</item>
<item>5: 27/1/2013</item>
<item>6: 3/2/2013</item>
<item>7: 10/2/2013</item>
<item>8: 17/2/2013</item>
<item>9: 24/2/2013</item>
<item>10: 3/3/2013</item>
<item>11: 10/3/2013</item>
<item>12: 17/3/2013</item>
<item>13: 24/3/2013</item>
<item>14: 31/3/2013</item>
<item>15: 7/4/2013</item>
<item>16: 14/4/2013</item>
<item>17: 21/4/2013</item>
<item>18: 28/4/2013</item>
<item>19: 5/5/2013</item>
<item>20: 12/5/2013</item>
<item>21: 19/5/2013</item>
<item>22: 26/5/2013</item>
</array>
<arrayname="Week1">
<item>1: 1/9/2013</item>
<item>2: 8/9/2013</item>
```

```
<item>3: 15/9/2013</item>
<item>4: 22/9/2013</item>
<item>5: 29/9/2013</item>
<item>6: 6/10/2013</item>
<item>7: 13/10/2013</item>
<item>8: 20/10/2013</item>
<item>9: 27/10/2013</item>
<item>10: 3/11/2013</item>
<item>11: 10/11/2013</item>
<item>12: 17/11/2013</item>
<item>13: 24/11/2013</item>
<item>14: 1/12/2013</item>
<item>15: 8/12/2013</item>
<item>16: 15/12/2013</item>
<item>17: 22/12/2013</item>
<item>18: 29/12/2013</item>
</array>
<arrayname="weeknull">
<item></item>
</array>
<arrayname="Lessons_1">
<item>ΤεχνητήΝοημοσύνη</item>
<item>ΓραμμικήΆλγεβρα</item>
<item>ΨηφιακήΣχεδίαση</item>
</array>

<arrayname="Lessons_2">
<item>ΑριθμητικήΑνάλυση</item>
<item>ΔομέςΔεδομένων</item>
<item>ΣχεδίασηΑλγορίθμων</item>
</array>

<arrayname="sele">
<item>ΕπιλέξτεΜάθημα</item>
</array>
</resources>
```

Στον κώδικα έχουμε ορίσει ορισμένους πίνακες οι οποίοι θα χρησιμοποιηθούν για τους σκοπούς της διπλωματικής.

Για να πετύχουμε το εφέ της δυνατότητας που μας δίνει ένας Spinner, θα πρέπει να πάρουμε έναν από τους παραπάνω πίνακες και με την βοήθεια του Adapter να το προσαρτήσουμε στον Spinner. Με τον κώδικα που ακολουθεί πετυχαίνουμε τον σκοπό αυτό.

```
semeS = (Spinner)findViewById(R.id.semestersele);
semeW = (Spinner)findViewById(R.id.semesterweeksele);
semeL = (Spinner)findViewById(R.id.semesterlessonsele);
```

Με τον παραπάνω τρόπο ορίζω (initialize) τα αντικείμενα τύπου Spinner. Προχωρώντας στην συνέχεια θα πρέπει τις τιμές από το array.xml, να τις πάρουμε και να τις αποθηκεύσουμε σε πίνακες τύπου String έτσι ώστε να μπορούμε να τις προσαρτήσουμε στους Spinner.

```
week = getResources().getStringArray(R.array.Week);
week1 = getResources().getStringArray(R.array.Week1);
weeknull = getResources().getStringArray(R.array.weeknull);
semester = getResources().getStringArray(R.array.Semester);
lesson = getResources().getStringArray(R.array.Lessons_1);
lesson1 = getResources().getStringArray(R.array.Lessons_2);
```

Βέβαια την σημαντικότερη δουλειά την κάνει ο Adapter, στον οποίο θα επικολλήσω έναν από τους παραπάνω πίνακες.

```
//--o adptoras apla syndeeii to periexomeno ton pinakwn me ton spinner--

finalArrayAdapter<String> adapter1=newArrayAdapter<String>
(this,R.layout.spinner,semester);
```

Ο Spinner μπορεί να έχει και μια μορφή ως προς την εμφάνιση της λίστας, η οποία καθορίζεται ως εξής.

```
adapter1.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

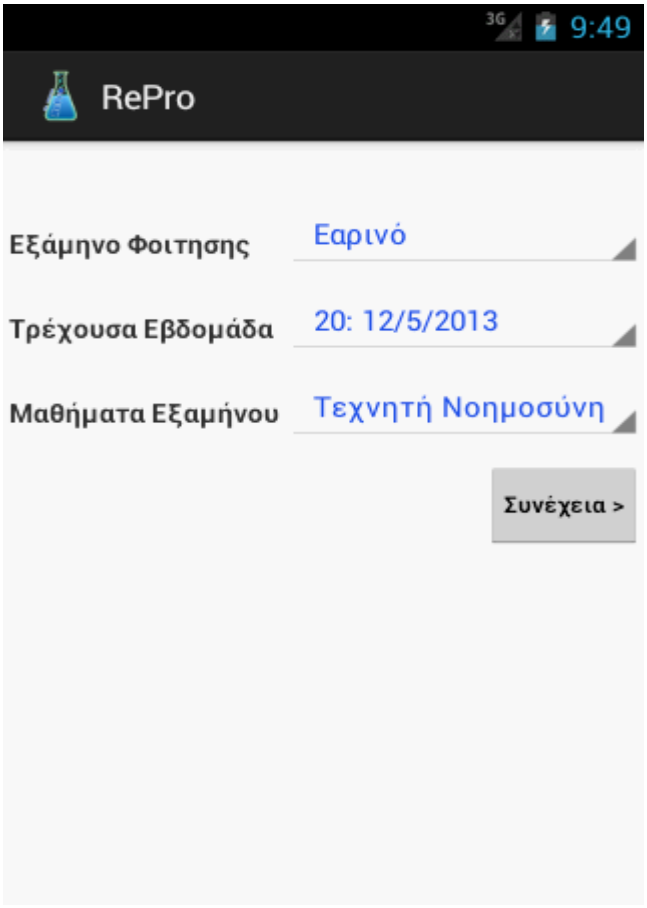
Όλα τα παραπάνω για να έχουν σημασία, θα πρέπει να προσαρτήσουν, πάνω στον καθόλα υπεύθυνο ο οποίος δεν είναι άλλος από τον Spinner . Αυτό πετυχαίνεται με τον παρακάτω κώδικα, ο οποίος δίνει και μια επιπλέον λειτουργικότητα αυτήν της επιλογής ενός αντικειμένου όταν κάνω κλικ πάνω σε μια τιμή από την λίστα.

```
//--oi adaptors pou dhmiourgh8hkan me thn seira tous mpainoun sta spinner--

semeS.setAdapter(adapter1);
semeS.setOnItemSelectedListener(this);
```

5.9. Εκκίνηση μιας Activity και επικοινωνία

Κλείνοντας την παρένθεση που ανοίξαμε σχετικά με τα Spinner, θα συνεχίσουμε την περιγραφή του κώδικα μας και θα αναφερθούμε σε ένα πολύ σημαντικό ζήτημα το οποίο αφορά στην μετάβαση μας από ένα συγκεκριμένο Activity σε ένα άλλο, καθώς και την επικοινωνία τους, μέσω της μεταφοράς πληροφοριών.

Συνεχίζοντας την ανάλυση του κώδικα που αφορά στην SelectActivity εικόνα 5.6, θα θέλαμε να τονίσουμε ότι ως σκοπό έχει να πάρει τις επιλογές του χρήστη, που αφορούν στο εξάμηνο φοίτησης την εβδομάδα του εξαμήνου, και το μάθημα για το οποίο θέλει να προσθέσει ορισμένες πληροφορίες σχετικά με την παρακολούθηση του.



Εικόνα 5.6

Οι πληροφορίες αυτές με την σειρά τους θα αποσταλούν μετά το πάτημα του κουμπιού με το όνομα Συνέχεια, στην ActivityDetailActivity.class. Καταρχάς για να μπορέσει ένα κουμπί να κάνει όλα αυτά θα πρέπει να υπάρχει, δηλαδή να οριστεί το αντικείμενο τύπου κουμπί όπως φαίνεται στον κώδικα που ακολουθεί.

```
subB = (Button)findViewById(R.id.subsele);
```

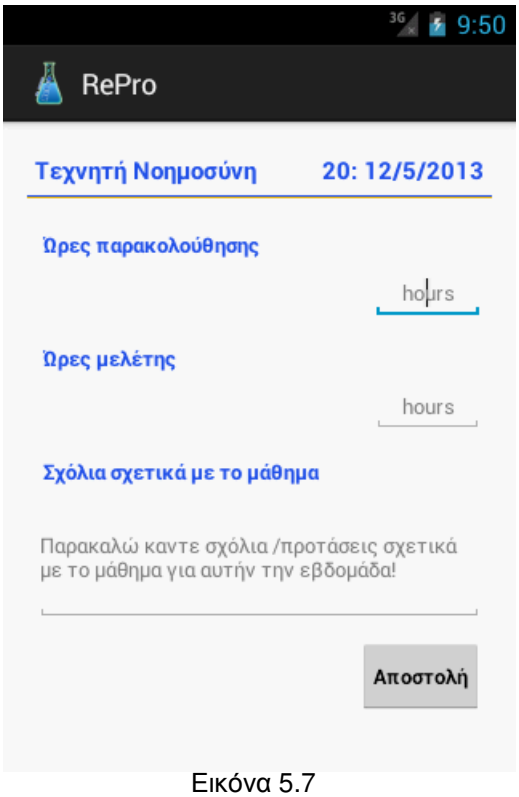
Το αμέσως επόμενο βήμα είναι να ορίσουμε την συμπεριφορά του όταν το πιέσουμε.

```
//--Ti 8a gineiotanpathsw to submit, fysika 8a metabwsthnDetailsActivity--
subB.setOnClickListener(newOnClickListener(){}
```

Προς το παρόν το κομμάτι του κώδικα δεν εκτελεί τίποτα όσο και να πατάμε το κουμπί. Θα πρέπει να δημιουργήσω ένα γεγονός (Intent) που να αφορά στην μετάβαση σε μια Activity με το όνομα DetailActivity.class.

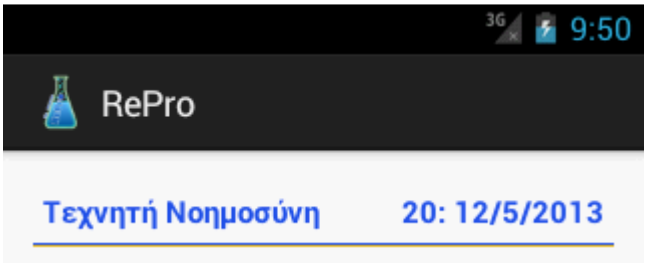
```
Intent intent = newIntent(SelectActivity.this,DetailsActivity.class);
startActivity(intent);
```

Με το startActivity πυροδοτώ το γεγονός και μεταβαίνω στην DetailActivity, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 5.7

Παρατηρώντας όμως την DetailActivity, βλέπουμε ότι έχει ως τίτλο το όνομα του μαθήματος, Τεχνητή Νοημοσύνη, καθώς επίσης και την εβδομάδα για την οποία ο χρήστης έχει επιλέξει να εισάγει πληροφορίες.



Εικόνα 5.8

Για να μπορέσει η DetailActivity, να γνωρίζει τις επιλογές που έκανε ο χρήστης στην προηγούμενη Activity εικόνα 5.6, θα πρέπει να υπάρχει ένας τρόπος οι πληροφορίες αυτές να έχουν σταλεί. Την δυνατότητα αυτήν μας την δίνει το Android με τον παρακάτω κώδικα.

```
//--tropos me ton opoio apostelnw pollaples plirofories metaksy activities--

intent.putExtra(ICostants.INFO_LESSON, lessonSele);
intent.putExtra(ICostants.INFO_WEEK, weekSele);
```

Ο κώδικας εισάγει κάποια επιπλέον πληροφορία στο γεγονός με το όνομα intent, μέσω του putExtra. Πώς όμως λειτουργεί η putExtra; Βλέπουμε ότι έχει δύο ορίσματα τύπου String,το ICostants.*INFO_LESSON* και το lessonSele. Το lessonSele είναι μια μεταβλητή τύπου String η οποία έχει ως τιμή την τιμή που ο χρήστης επέλεξε από τον Spinner:

```
semeL = (Spinner)findViewById(R.id.semesterlessonsele);
```

Πώς όμως παίρνουμε την τιμή επιλογής από έναν Spinner;

```
publicvoidgetSpinnerSelection(){
      semeSele = (String) semeS.getSelectedItem();
      weekSele = (String) semeW.getSelectedItem();
      lessonSele = (String) semeL.getSelectedItem();      }
```

Όπως βλέπουμε παραπάνω έχουμε δημιουργήσει μια συνάρτηση με την οποία παίρνουμε τις τιμές που ο χρήστης έχει επιλέξει από τους διάφορους Spinner, ένας από τους οποίους είναι και αυτός που δίνει το μάθημα το οποίο ο χρήστης έχει επιλέξει.

```
lessonSele = (String) semeL.getSelectedItem();
```

Αυτό που έχει εξαιρετικό ενδιαφέρον είναι η μεταβλητή IConstants.*INFO_LESSON* την οποία χρησιμοποιεί η putExtra και στην οποία αποθηκεύει την τιμή της lessonSele, με την μορφή keyvalue. Για να δώσουμε μια περιγραφή αυτής της σταθεράς θα ανοίξουμε μια παρένθεση στην υποενότητα που ακολουθεί.

5.10. Σταθερές

Όπως αναφέραμε η IConstants.*INFO_LESSON* δεν είναι τίποτα άλλο, από μια σταθερά η οποία αποθηκεύει μια τιμή τύπου String. Οι σταθερές αυτού του τύπου μπορούν να οριστούν μέσα σε ένα αρχείο με το όνομα IConstants.java, και το οποίο δεν είναι τίποτε άλλο από ένα Interface όπως φαίνεται παρακάτω.

```
package chris.ateith.finalthesis.repro2;
//--apo8hkeuw tis metblhtes ws sta8eres pou 8a xreiastoun--
//--gia na dhmiourghsw ton pinka ths bashs mas           --
publicinterfaceICostants {
      publicstaticfinalintDATABASE_VERSION=1;
      publicstaticfinal String DATABASE_NAME="repro_db";
      publicstaticfinal String TABLE_NAME="repro";
      publicstaticfinal String LESSON_NAME="flesson";
      publicstaticfinal String IN_HOURS="hourin";
      publicstaticfinal String OUT_HOURS="hourout";
      publicstaticfinal String COMMENTS="comments";
      publicstaticfinal String SEMESTER="semester";
      publicstaticfinal String SEME_WEEK="semeweek";
      publicstaticfinal String KEY_ID="_id";

      //--giata extra poustelnoumemesw intent--
      publicstaticfinal String INFO_ID="info";
      publicstaticfinal String INFO_LESSON="infoless";
      publicstaticfinal String INFO_WEEK="infoweek";
      publicstaticfinal String INFO_SEME="infoseme";
}
```

Ο κώδικας είναι αυτός που χρησιμοποιούμε για την υλοποίηση της διπλωματικής εργασίας μας, και φαίνονται όλες οι σταθερές που έχουν οριστεί για τον σκοπό αυτό.

5.11. Δημιουργία μενού αλληλεπίδρασης

Τα μενού είναι βασικά εργαλεία λειτουργικότητας ενός προγράμματος, τα οποία μας δίνουν επιπλέον δυνατότητες αλληλεπίδρασης με μία εφαρμογή. Μέσα από την εμπειρία μας όλοι έχουμε αλληλεπιδράσει με μενού, και γνωρίζουμε ότι δεν είναι ορατά όλα τα πεδία με την πρώτη ματιά. Αυτό σημαίνει ότι τις περισσότερες φορές θα πρέπει να υπάρχει κάποιο είδος ενέργειας από μέρους του χρήστη για να εμφανιστούν οι επιλογές ενός μενού.

Θα μπορούσαμε να αναφέρουμε ότι υπάρχουν δύο είδη μενού:

- **Optionsmenu –** το οποίο είναι σχετικό με μια Activity, και στο Android αυτό πυροδοτείται με την πίεση του κουμπιού MENU της συσκευής.
- **Contextmenu –** το οποίο εμφανίζει επιλογές για ένα συγκεκριμένο αντικείμενο (view) μιας Activity. Για να ενεργοποιηθεί θα πρέπει να επιλέξουμε το αντικείμενο και να πιέσουμε παρατεταμένα επάνω του.

Στην περίπτωση μας, δηλαδή στην περίπτωση της SeleActivity.java, κάνουμε χρήση του πρώτου μενού, με την εισαγωγή του παρακάτω κώδικα.

```
    @Override

    public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.mainmenu, menu);
    //--βαζω το id το πρωτο item στην μεταβλητη τυπου μενου--
    //-- με τον τροπο αυτο θα μπορω να την απενεργοποιησω --
    //-- se periptwsh pou exw times gia user kai password --

    returntrue;
          }
```

Όπως σε όλα τα αντικείμενα τα οποία μας παρέχουν μια δυνατότητα επιλογής μέσα από μια λίστα, έτσι και στο μενού θα πρέπει να καθορίσουμε την λίστα των επιλογών μας μέσα από την γραμμή του κώδικα.

```
getMenuInflater().inflate(R.menu.mainmenu, menu);
```

Τι αντιπροσωπεύουν όμως οι μεταβλητές R.menu.mainmenu και menu; Το menu είναι αντικείμενο τύπου μενού στο οποίο δίνουμε μια μορφή, όσον αφορά το UI όπως επίσης και τις επιλογές, μέσω του αρχείου mainmenu.xml, το οποίο βρίσκεται στον φάκελο menu/maimenu.xml. Η μορφή του οποίου φαίνεται στο παρακάτω κώδικα.

```
<?xmlversion="1.0"encoding="utf-8"?>
<menuxmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/exit"
android:orderInCategory="101"
android:showAsAction="never"
android:title="@string/menu_exit"/>
<item
android:id="@+id/stat"
android:orderInCategory="102"
android:showAsAction="never"
android:title="@string/menu_stat"/>
</menu>
```

Ο ορισμός του μενού ξεκινάει με

```
<menuxmlns:android="http://schemas.android.com/apk/res/android">
```

και συνεχίζει με την εισαγωγή των επιλογών του μενού. Στην δικιά μας περίπτωση έχουμε δύο επιλογές, η μία είναι η έξοδος και ο τερματισμός του προγράμματος και η άλλη στην εξαγωγή στατιστικών αποτελεσμάτων με την μορφή γραφήματος.

Για να εισάγουμε μια επιλογή μέσα σε ένα μενού αρκεί να προσθέσουμε τον παρακάτω κώδικα.

```
<item
android:id="@+id/exit"
android:orderInCategory="101"
android:showAsAction="never"
android:title="@string/menu_exit"/>
```

Με τον κώδικα αυτό ορίζουμε μία από τις επιλογές του μενού, η οποία αφορά την έξοδο μας από την εφαρμογή. Κάνοντας μια ανάγνωση από την αρχή βλέπουμε ότι με το android:id="@+id/exit", δίνουμε ένα id το οποίο πρέπει να είναι και μοναδικό και το οποίο χαρακτηρίζει το συγκεκριμένο αντικείμενο, και αποτελεί το αναγνωριστικό του συγκεκριμένου αντικειμένου, κάθε φορά που αναφερόμαστε σε αυτό. Στην συνέχεια βλέπουμε το χαρακτηριστικό android:orderInCategory="101", με το οποίο καθορίζουμε την σειρά με την οποία θα εμφανίζεται η συγκεκριμένη επιλογή μέσα στην λίστα των επιλογών του μενού. Με το χαρακτηριστικό android:title="@string/menu_exit", καθορίζουμε το όνομα που θα έχει η συγκεκριμένη επιλογή του μενού.

Μετά την εμφάνιση του μενού θα πρέπει να του δώσουμε και μια λειτουργικότητα, δηλαδή ποιες θα είναι οι ενέργειες που θα εκτελούνται με το πάτημα των διαφόρων επιλογών. Έτσι για την υλοποίηση της λειτουργικότητας χρησιμοποιείται ο παρακάτω κώδικας.

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
          switch(item.getItemId()){
          caseR.id.exit:{
          Intent intent = newIntent(Intent.ACTION_MAIN);
          finish();
          returntrue;
                        }
          caseR.id.stat:{
    Intent intent= newIntent(SelectActivity.this,GenericReport.class);
                startActivity(intent);
                returntrue;
                        }
           }
          returnfalse;
    }
}
```

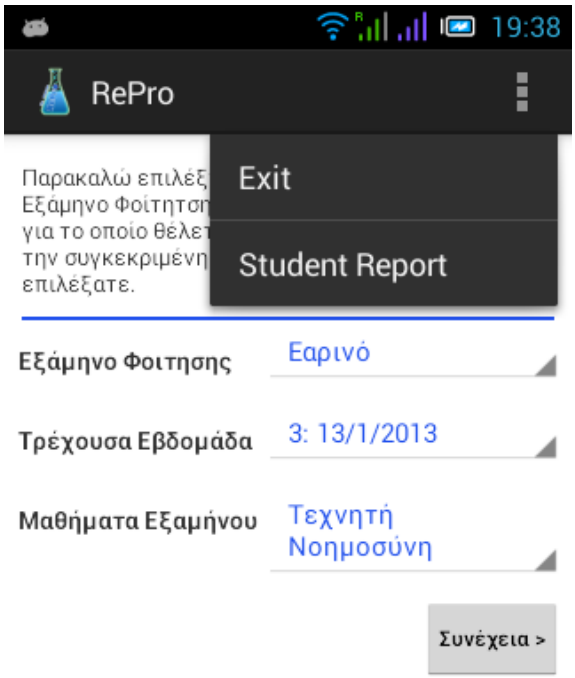
Παρατηρούμε ότι η συνάρτηση παίρνει ως όρισμα ένα αντικείμενο τύπου MenuItem. Για να μπορέσω να καθορίσω τι θα γίνεται όταν ο χρήστης επιλέξει συγκεκριμένη επιλογή, θα πρέπει να υλοποιήσω μια switch, η οποία ανάλογα με το id της επιλογής θα μπαίνει σε συγκεκριμένο Case. Έτσι αν ο χρήστης έχει πιέσει το exit, τότε μπαίνουμε στο πρώτο Case, με το οποίο ορίζω ένα Intent για τον τερματισμό από το πρόγραμμα.

```
Intent intent = new Intent(Intent.ACTION_MAIN);
```

Αν ο χρήστης επιλέξει την δεύτερη επιλογή StudentReport, τότε μπαίνουμε στην δεύτερη Case και καλούμε την Activity με το όνομα GenericReport.class.

```
Intent intent= new Intent(SelectActivity.this,GenericReport.class);
```

Αυτό πού βλέπει ο χρήστης ως αποτέλεσμα όλου του κώδικα για τον ορισμό του μενού, φαίνεται στην εικόνα που ακολουθεί.



Εικόνα 5.9

5.12. Μέθοδοι αποθήκευσης δεδομένων

Σίγουρα κατά την ανάπτυξη μια εφαρμογής για Android κινητά θα μας δημιουργηθεί η ανάγκη, για ένα τρόπο αποθήκευσης των δεδομένων που χρησιμοποιούνται κατ' επανάληψη. Οι μέθοδοι οι οποίοι χρησιμοποιήθηκαν για το σκοπό αυτό κατά την υλοποίηση της διπλωματικής εργασίας είναι οι ακόλουθοι :

- Χρήση μιας εσωτερικής Βάσης Δεδομένων
- SharedPreferences

5.13. Βάση δεδομένων SQLite [12]

Για την αποθήκευση πιο πολύπλοκων δεδομένων μια σχεσιακή βάση δεδομένων προσφέρει ταχύτερη και πιο ευέλικτη πρόσβαση από ότι τα απλά αρχεία ή ο μηχανισμός SharedPreferences. Η πλατφόρμα Android παρέχει έμφυτη υποστήριξη της SQLite , μιας ελαφριάς έκδοσης αλλά με ολοκληρωμένη λειτουργικότητα σχεσιακή βάση δεδομένων την οποία μπορούμε να χρησιμοποιήσουμε για την αποθήκευση των δεδομένων των εφαρμογών μας. Κάθε εφαρμογή μπορεί να αποκτήσει ένα δικό της στιγμιότυπο της βάσης, το οποίο αποθηκεύεται στη διαδρομή /data/data/<package_name>/databases της συσκευής.

Οι βασικοί τύποι δεδομένων που υποστηρίζει η SQLite είναι ο TEXT που χρησιμοποιείται για την αποθήκευση αλφαριθμητικών, ο INTEGER που χρησιμοποιείται για την αποθήκευση ακεραίων τιμών και ο REAL για την αποθήκευση δεκαδικών αριθμητικών τιμών. Θα πρέπει ωστόσο να τονίσουμε ότι η SQLite δεν ελέγχει από μόνη της αν οι τύποι των προς εγγραφή τιμών ταυτίζονται με αυτούς των στηλών του πίνακα. Στην εφαρμογή που αναπτύξαμε δημιουργήσαμε μια τέτοια βάση δεδομένων , ακολουθώντας τα εξής βήματα:

- Δημιουργία και άνοιγμα βάσης
- Δημιουργία πίνακα
- Δημιουργία διαδικασίας εισαγωγής εγγραφών
- Κλείσιμο βάσης

Για να δούμε στην πράξη όλα αυτά θα παρουσιάσουμε κομμάτια του κώδικα που αναπτύξαμε για τους σκοπούς αυτούς.

```
package chris.ateith.finalthesis.repro2;
//--apo8hkeuw tis metblhtes ws sta8eres pou 8a xreiastoun--
//--gia na dhmiourghsw ton pinka ths bash smas           --

publicinterfaceICostants {
 publicstaticfinalintDATABASE_VERSION=1;
 publicstaticfinal String DATABASE_NAME="repro_db";
 publicstaticfinal String TABLE_NAME="repro";
 publicstaticfinal String LESSON_NAME="flesson";

publicstaticfinal String IN_HOURS="hourin";
 publicstaticfinal String OUT_HOURS="hourout";
 publicstaticfinal String COMMENTS="comments";
 publicstaticfinal String SEMESTER="semester";
 publicstaticfinal String SEME_WEEK="semeweek";
 publicstaticfinal String KEY_ID="_id";

}
```

Στο interfaceICostants ορίζουμε τις σταθερές που θα χρησιμοποιήσουμε σε διάφορα σημεία του κώδικα όπως το όνομα της βάσης, το όνομα του πίνακα, τα ονόματα των στηλών που θα περιέχονται σε αυτόν τον πίνακα κ.α.

Η σταθερά DATABASE_VERSION, αφορά στον αριθμό έκδοσης του σχήματος της βάσης. Πρόκειται για έναν αύξοντα αριθμό ο οποίος λογικά θα πρέπει να ξεκινάει πάντα από 1, ενώ μελλοντικά αν γίνουν αλλαγές στο σχήμα της βάσης θα αυξάνει. Τον αριθμό αυτόν ελέγχει η πλατφόρμα κάθε φορά που ενημερώνει την εφαρμογή ώστε να κρίνει αν χρειάζεται να καλέσει την onUpgrade(), μέσω της οποίας ενημερώνεται το σχήμα στην τελευταία έκδοση.

Η τιμή KEY_ID αποτελεί το όνομα της στήλης που θα χρησιμοποιηθεί ως πρωτεύον κλειδί στον πίνακα. Θα πρέπει πάντοτε να χρησιμοποιούμε την συγκεκριμένη τιμή στις εφαρμογές μας, καθώς αρκετές από τις συναφείς λειτουργίες της πλατφόρμας βασίζονται στη συγκεκριμένη σύμβαση.

Για τη δημιουργία και την αναβάθμιση μιας βάσης δεδομένων σε μια εφαρμογή Android η συνήθης πρακτική είναι να επεκτείνουμε την κλάση SQLiteOpenHelper και να υπερκαλύπτουμε τις κατάλληλες μεθόδους της.

```
	package chris.ateith.finalthesis.repro2;


	importandroid.content.Context;
	importandroid.database.sqlite.SQLiteDatabase;
	importandroid.database.sqlite.SQLiteException;
	importandroid.database.sqlite.SQLiteDatabase.CursorFactory;
	importandroid.database.sqlite.SQLiteOpenHelper;
	importandroid.util.Log;


	public class ReProHelper extends SQLiteOpenHelper{

		private static final String DATABASE_CREATE="create table "+
		ICostants.TABLE_NAME + " ("+
		ICostants.KEY_ID +" integer primary key autoincrement, "+
		ICostants.LESSON_NAME + " text not null, "+
		ICostants.IN_HOURS + " text, " +
		ICostants.OUT_HOURS + " text, " +
		ICostants.COMMENTS + " text, " +
		ICostants.SEMESTER + " text not null, " +
		ICostants.SEME_WEEK + " text not null);";

	publicReProHelper(Context context) {

	super(context, ICostants.DATABASE_NAME, null, ICostants.DATABASE_VERSION);
		// TODO Auto-generated constructor stub
	}

	@Override
	public void onCreate(SQLiteDatabasedb) {
		// TODO Auto-generated method stub
		try {
			db.execSQL(DATABASE_CREATE);
			}
			catch (SQLiteException e){
				Log.v("Create table exception", e.getMessage());
			}
	}

	@Override
	public void onUpgrade(SQLiteDatabasedb, intoldVersion, intnewVersion) {
		// TODO Auto-generated method stub
		db.execSQL("DROP TABLE IF EXISTS repro");
		onCreate(db);
	}

}
```

Μια από αυτές είναι η onCreate() στην οποία τοποθετείται ο κώδικας δημιουργίας της βάσης ενώ στην onUpgrade() τοποθετείται ο κώδικας για την ενημέρωση του σχήματος της βάσης σε περίπτωση που υπάρξουν αλλαγές σε κάποια μελλοντική έκδοση. Το βασικό συστατικό του παραπάνω κώδικα είναι το SQLstatement που δημιουργεί το σχήμα της βάσης και που έχει οριστεί ως privatestatic σταθερά. Το statement αυτό όπως μπορούμε να δούμε χρησιμοποιείται από την onCreate(). Ο constructor της συγκεκριμένης κλάσης δημιουργεί ένα αντικείμενο τύπου SQLiteOpenHelper αρχικοποιώντας το με το τρέχον context, το όνομα της βάσης και την έκδοση του σχήματος. Σύμφωνα με την τεκμηρίωση του Android η βάση δημιουργείται με την πρώτη κλήση είτε του getReadableDatabase() είτε του getWriteableDatabase().

Αφού δημιουργήσουμε την βάση δεδομένων μπορούμε να αλληλεπιδράσουμε με αυτήν δημιουργώντας , διαγράφοντας ή ενημερώνοντας εγγραφές μέσω των μεθόδων insert(), delete() και update() αντίστοιχα της κλάσης SQliteDatabase.

Για την εκτέλεση ερωτημάτων (queries) μπορούμε να χρησιμοποιήσουμε είτε την μέθοδο rawQuery() είτε την query(). Η μεν πρώτη δέχεται και εκτελεί ένα ερώτημα που είναι γραμμένο σε SQL, η δε δεύτερη δέχεται με την μορφή παραμέτρων μόνο τις συνθήκες από τις οποίες εκτελούνται τα διάφορα τμήματα του SQL. Για να γίνει καλύτερα αυτό αντιληπτό, παραθέτουμε τον εξής κώδικα:

```
Cursor cursor= database.query(ICostants.TABLE_NAME, new String[]{
      ICostants.KEY_ID,ICostants.LESSON_NAME,ICostants.IN_HOURS,ICostants.OUT_HOUR
S, ICostants.COMMENTS, ICostants.SEMESTER,ICostants.SEME_WEEK},
ICostants.LESSON_NAME + " ='"+lesson+"' AND "+ ICostants.SEME_WEEK +
"='"+semeWeek+"'", null, null, null,null);
```

Στην έκδοση αύτη της query(), η πρώτη παράμετρος αναφέρεται στο όνομα του πίνακα του σχήματος, ενώ η δεύτερη στις στήλες που θέλουμε να επιστραφούν. Η τρίτη είναι πολύ σημαντική γιατί αφορά το WHERE τμήμα του statement. Στην παράμετρο αυτή γράφουμε την έκφραση με την μορφή αλφαριθμητικού χωρίς να συμπεριλάβουμε το ίδιο το WHERE. Η 5^{η} παράμετρος αναφέρεται στην στήλη στην οποία θέλουμε να κάνουμε ομαδοποίηση, η 6^{η} στον ορισμό του HAVING και τέλος, η

7η καθορίζει τη στήλη που θα χρησιμοποιηθεί για ταξινόμηση. Αν το ερώτημα μας δεν περιέχει WHERE, GROUPBY, HAVING η κάποιο από τα υπόλοιπα τμήματα, περνάμε ως παράμετρο την τιμή null.

Παρατηρούμε ότι η query(), επιστρέφει ένα αντικείμενο τύπου Cursor, το οποίο είναι αντίστοιχο του αντικειμένου ResultSet της Java, μπορεί να περιέχει έναν πίνακα αποτελεσμάτων ή να είναι άδειο, σε περίπτωση που το ερώτημα δεν επιστρέψει αποτελέσματα. Η μέθοδος getCount() επιστρέφει τον αριθμό των εγγραφών που περιέχονται σε αυτό, ενώ για την προσπέλαση τους χρησιμοποιούμε αρχικά το moveToFirst() που μας μεταφέρει στην πρώτη εγγραφή και στην συνέχεια το moveToNext(). Η isAfterLast() υποδεικνύει αν υπάρχουν και άλλες εγγραφές επιστρέφοντας true ή αν έχουμε φτάσει στο τέλος επιστρέφει false.

Στην κλάση ReProAdapter που ακολουθεί υλοποιούνται όλες οι λειτουργίες επικοινωνίας με την βάση και γίνεται χρήση των αντικειμένων και των μεθόδων που προαναφέρθηκαν.

```
package chris.ateith.finalthesis.repro2;


importandroid.content.ContentValues;
importandroid.content.Context;
importandroid.database.Cursor;
importandroid.database.SQLException;
importandroid.database.sqlite.SQLiteDatabase;
importandroid.util.Log;


public class ReProAdapter {
	//--h sta8era xreiazetai gia na epistrepsw ta apotelesmata se
	auksousa seira--
		private static final String GET_ALL_ORDER_BY =
		ICostants.SEME_WEEK + " ASC";
		private Context context;
	//me thn sqlite mporw na allhlepidrasw me thn bash
		privateSQLiteDatabase database;
	//dhmiourgw ena antikeimeno me to opoio orizw thn bash mou
```

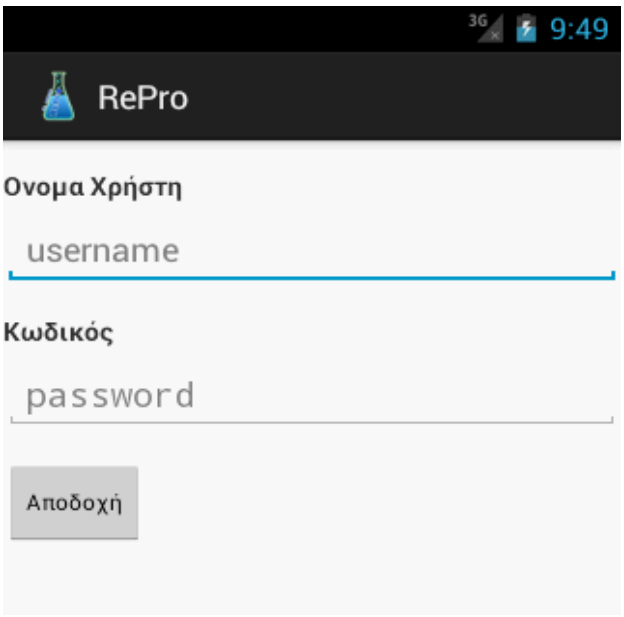
```
	privateReProHelperdbHelper;

	publicReProAdapter(Context context){
		this.context=context;
	}
//---opens the database---

	publicReProAdapter open() throwsSQLException{
		dbHelper= newReProHelper(context);
		database= dbHelper.getWritableDatabase();
		Log.v("anoigw", "bash");
		returnthis;

		}

		//---closes the database---

	publicvoid close(){
		Log.v("kleinw", "bash");
		dbHelper.close();
		}

		//---insert a class into the reprotable---

	publiclongcreateLesson(String flesson,Stringhourin, String
	hourout,Stringcomments,String semester, String semeweek){
		Log.v("apouhkeuw", "tis times");
	ContentValuesinitialValues =createContentValues (flesson,hourin, hourout,
	comments, semester,semeweek);
	returndatabase.insert(ICostants.TABLE_NAME, null, initialValues);

		}

	//---fetch lesson of certain semester and certain semesterweek---

	public Cursor fetchLessonOfSemeWeek(String lesson, String semeWeek){
		Log.v("diabazw", "timh");
		Log.v("epistrefw", "timh ma8hma: "+lesson+" week: "+semeWeek);
		Cursor cursor= database.query(ICostants.TABLE_NAME, new String[]{

	ICostants.KEY_ID,ICostants.LESSON_NAME,ICostants.IN_HOURS,ICostants.OUT_HOUR
	S, ICostants.COMMENTS, ICostants.SEMESTER,ICostants.SEME_WEEK},
	ICostants.LESSON_NAME + " ='"+lesson+"' AND "+ ICostants.SEME_WEEK +
	"='"+semeWeek+"'", null, null, null,null);
			if(cursor!=null){
				cursor.moveToFirst();
			}
			return cursor;

		}

	//---fetch lesson of certain semester---
	public Cursor fetchLessonOfSeme(String lesson, String seme){
		Log.v("epistrefw", "timh ma8hma: "+lesson+" eksamhno: "+seme);
		Cursor cursor= database.query(ICostants.TABLE_NAME, new String[]{
```

```
                  ICostants.IN_HOURS,ICostants.OUT_HOURS,ICostants.SEME_WEEK},
                  ICostants.LESSON_NAME + " ='"+lesson+"' AND "+
      ICostants.SEMESTER + "='"+seme+"'", null, null, null, GET_ALL_ORDER_BY);
                  if(cursor!=null){
                        cursor.moveToFirst();
                  }
                  return cursor;

            }
      //---fetch lesson of certain semester and week mikroterhthstrexousas---
      public Cursor fetchLessonOfSememin(String lesson, String seme, String week){
            Log.v("epistrefw", "timh ma8hma: "+lesson+" eksamhno: "+seme+" giathn
            week "+week);
            Cursor cursor= database.query(ICostants.TABLE_NAME, new String[]{
                  ICostants.IN_HOURS,ICostants.OUT_HOURS,
                  ICostants.SEME_WEEK}, ICostants.LESSON_NAME + " ='"+lesson+"'
                  AND "+ ICostants.SEMESTER + "='"+seme+"' AND "+
                  ICostants.SEME_WEEK+ "='"+week+"'", null, null, null,
                  GET_ALL_ORDER_BY);

                  if(cursor!=null){
                        cursor.moveToFirst();
                        }
                  return cursor;
                        }
      //--update a specific lesson from RePro table--

      publicbooleanupdateLesson(Long cid,Stringflesson,Stringhourin,Stringhourout,
      String comments,String semester, String semeweek){
            Log.v("update", "timh");
            ContentValues
      updateValues=createContentValues(flesson,hourin,hourout,comments,semester,se
      meweek);

      returndatabase.update(ICostants.TABLE_NAME, updateValues, ICostants.KEY_ID +
      " = " + cid, null)>0;

            }

//---dhmiourgwzeughsthlhstimhstaopoiataapouhkeuwsemetablhthtypouContentValuesgia
to repro---

      privateContentValuescreateContentValues(String
      flesson,Stringhourin,Stringhourout, String comments,String semester, String
      semeweek ){
            ContentValues values=newContentValues();
            values.put(ICostants.LESSON_NAME, flesson);
            values.put(ICostants.IN_HOURS, hourin);
            values.put(ICostants.OUT_HOURS, hourout);
            values.put(ICostants.COMMENTS, comments);
            values.put(ICostants.SEMESTER, semester);
            values.put(ICostants.SEME_WEEK, semeweek);
            returnvalues;
            }
}
```

5.14. SharedPreferences

Η δεύτερη μέθοδος αποθήκευσης δεδομένων που χρησιμοποιήσαμε στην εφαρμογή μας είναι η SharedPreferences η οποία είναι ένα interface που μια εφαρμογή μπορεί να χρησιμοποιήσει για να κάνει αποθήκευση δεδομένων σε ζεύγη τύπου κλειδί-τιμή. Η πληροφορία αποθηκεύεται σε ένα xml αρχείο σε συγκεκριμένη διαδρομή του filesystem της συσκευής, η οποία εξαρτάται από το όνομα του πακέτου που περιέχει το συγκεκριμένο Activity.

Ο μηχανισμός αυτός χρησιμοποιείται συνήθως για την αποθήκευση ρυθμίσεων σχετικές με την εφαρμογή. Μπορεί επίσης να χρησιμοποιηθεί όπως και στην δικιά μας περίπτωση, για να αποθηκεύσει πληροφορίες σύνδεσης (login) εικόνα 5.10 όπως ένα username ή ένα password.



Εικόνα 5.10

Τα δεδομένα τύπου SharedPreferences είναι προσβάσιμα από όλα τα συστατικά της εφαρμογής που τα δημιούργησε. Ένα Activity μπορεί να διαβάσει το αρχείο των SharedPreferences χρησιμοποιώντας τη μέθοδο getPreferences(), ενώ αν χρησιμοποιούνται περισσότερα αρχεία από την εφαρμογή, μέσω της getsharedPreferences(). Αν το αρχείο των SharedPreferences βρεθεί στο φάκελο θα ανοίξει για ανάγνωση, ενώ αν δεν υπάρχει θα δημιουργηθεί. Υπάρχουν τρεις διαφορετικοί τύποι πρόσβασης, οι οποίοι καθορίζονται κατά την κλήση των προαναφερθέντων μεθόδων.

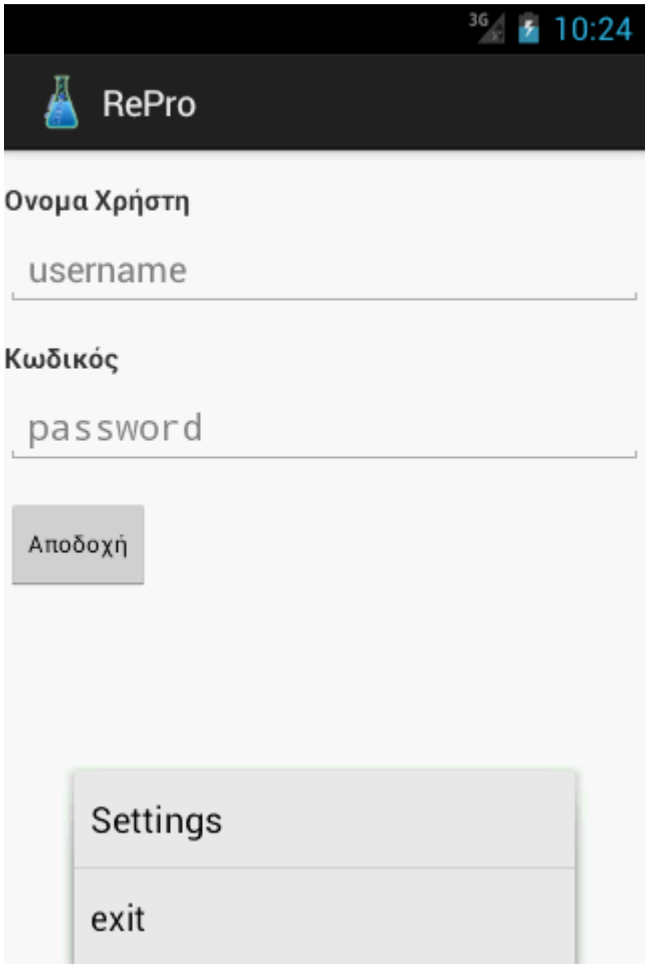
- MODE_PRIVATE: Μόνο η εφαρμογή από όπου καλείται η μέθοδος έχει πρόσβαση στο XML αρχείο.

- MODE_WORLD_READABLE: Όλες οι εφαρμογές μπορούν να διαβάσουν το αρχείο.
- MODE_WORLD_WRITEABLE: Όλες οι εφαρμογές μπορούν να διαβάσουν και να γράψουν στο XML.

Έχοντας αρχικοποιήσει ένα αντικείμενο τύπου SharedPreferences, απαιτείται ένα αντικείμενο τύπου Editor για να αναλάβει την εγγραφή των ζευγαριών κλειδιού-τιμή στο αρχείο, μέσω της αντίστοιχης μεθόδου. Για κάθε βασικό τύπο υπάρχει η αντίστοιχη put μέθοδος που λαμβάνει ως παραμέτρους ένα αλφαριθμητικό που ορίζει το όνομα του κλειδιού και την τιμή του τύπου που θέλουμε να αποθηκεύσουμε. Για μια ακέραια τιμή η κατάλληλη μέθοδος είναι putInt(), για μια Boolean η putBoolean() κ.ο.κ. Ομοίως για την ανάκτηση μιας τιμής υπάρχει η αντίστοιχη get μέθοδος που λαμβάνει ως πρώτη παράμετρο επίσης το όνομα του κλειδιού και ως δεύτερη μια default τιμή που θα επιστραφεί σε περίπτωση που η τιμή για το συγκεκριμένο κλειδί στο αρχείο είναι κενή.

Μέσω του μηχανισμού των SharedPreferences το Android παρέχει έναν ομοιόμορφο τρόπο για την επιλογή των ρυθμίσεων για όλες τις εφαρμογές, μέσω μιας στάνταρτ οθόνης. Χρησιμοποιώντας μια PreferenceCategory μπορούμε να δηλώσουμε ένα σετ ρυθμίσεων στην ίδια κατηγορία ενώ με ένα PreferenceScreen ορίζουμε μια νέα οθόνη ρυθμίσεων. Το συγκεκριμένο συστατικό τοποθετείται ως root στο xml αρχείο της οθόνης ρυθμίσεων και μετά σε αυτό τοποθετούνται τα διάφορα άλλα συστατικά που μπορεί να είναι ένα CheckBoxPreference, EdittextPreference κλπ.

Όπως έχουμε αναφέρει στην εφαρμογή μας μέσω SharedPrefernces αποθηκεύονται τα username και password του χρήστη. Τα στοιχεία αυτά εισάγονται μέσα από την αντίστοιχη οθόνη, η οποία προβάλλεται την πρώτη φορά που ο χρήστης θα τρέξει την εφαρμογή και οι τιμές για το username και password είναι κενές επιλέγοντας από το μενού settings εικόνα 5.11. Σε περίπτωση που επιλέξει settings εκτελείται το παρακάτω αρχείο.



Εικόνα 5.11

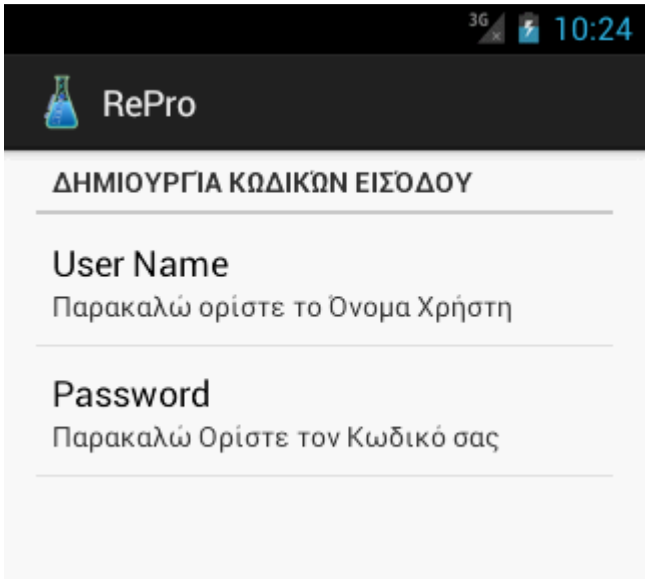
```
<?xmlversion="1.0"encoding="utf-8"?>
<PreferenceScreenxmlns:android="http://schemas.android.com/apk/res/android">
<PreferenceCategory
android:title="Δημιουργίακωδικώνεισόδου">
<EditTextPreference
android:title="User Name"
android:key="username"
android:summary="Παρακαλώ ορίστε το Όνομα Χρήστη"/>
<EditTextPreference
android:title="Password"
android:key="password"
android:summary="Παρακαλώ Ορίστε τον Κωδικό σας"/>
</PreferenceCategory>
</PreferenceScreen>
```

Στην συγκεκριμένη οθόνη ρύθμισης χρησιμοποιούνται δύο EditPreferences πεδία, ένα για το username και ένα για το password του χρήστη. Το χαρακτηριστικό key όνομα του Κλειδιού μέσω του οποίου αποκτούμε πρόσβαση στην αποθηκευμένη τιμή, ενώ τα χαρακτηριστικά title και summary προβάλουν έναν τίτλο και μια περιγραφή αντίστοιχα. Η οθόνη ρυθμίσεων ανασυγκροτείται και προβάλλεται από την Activity ChPreferences της οποίας ο κώδικας είναι ο ακόλουθος.

```
package chris.ateith.finalthesis.repro2;

      importandroid.os.Bundle;
      importandroid.preference.PreferenceActivity;
      importandroid.preference.PreferenceFragment;


      public class ChPreferences extends  PreferenceFragment{

             @Override
            public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

                    // Load the preferences from an XML resource
            addPreferencesFromResource(R.xml.preferences);

            }

}
```

Στο συγκεκριμένο activity η addpreferencesFromResource() είναι υπεύθυνη για την οπτικοποίηση των περιεχομένων του xml αρχείου και την προβολή τους στην οθόνη, όπως φαίνεται στην εικόνα 5.12.

Το κεντρικό Activity της εφαρμογής ελέγχει αν τα username και password έχουν οριστεί από τον χρήστη και σε περίπτωση που δεν έχει γίνει, θα προβληθεί η οθόνη εισαγωγής τους. Αν έχουν ήδη οριστεί θα προβληθεί η οθόνη του σχήματος 5.8 με απενεργοποιημένη την επιλογή settings.



Εικόνα 5.12

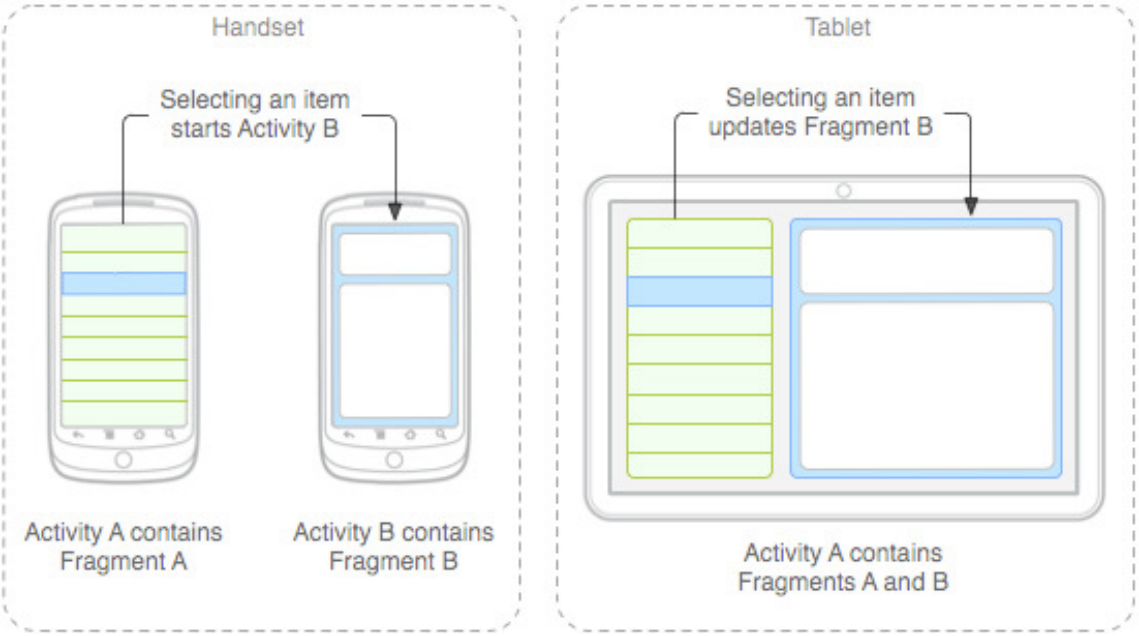
Ο κώδικας της LogInChris ο οποίος και εκτελεί όλα αυτά που προαναφέραμε φαίνεται πιο κάτω.

```
        prefs = PreferenceManager.getDefaultSharedPreferences(this);

        //--pairnw   ta preferences an den yparxoun   tote null --
        userN = prefs.getString("username", null);
        passW = prefs.getString("password", null);
```

Για να λειτουργήσει η εφαρμογή σωστά δεν θα πρέπει να παραλείψουμε να δηλώσουμε το preferenceActivity στο Manifest.

5.15. Fragments

Δεν θα μπορούσαμε να κλείσουμε το κεφάλαιο περιγραφής του κώδικα της εφαρμογής χωρίς να κάνουμε αναφορά σε ένα από τα ποιο σημαντικά εργαλεία που μας παρέχει το Android και δεν είναι άλλο από τα Fragments. Στις προηγούμενες ενότητες είδαμε τι είναι και πως χρησιμοποιούμε μια Activity, είναι γεγονός ότι σε συσκευές με μικρή οθόνη, μια Activity καλύπτει εξολοκλήρου την περιοχή της οθόνης. Τι συμβαίνει όταν μια Activity προβάλλεται από μεγαλύτερες συσκευές όπως είναι οι ταμπλέτες ή όταν στην ίδια οθόνη θέλουμε να προβάλλουμε 2 ή 3 Activity; Αυτήν την περίπτωση έρχονται να καλύψουν τα Fragments, τα οποία είναι μίνι Activities εικόνα 5.13.



Εικόνα 5.13

Όταν σκεφτόμαστε τα Fragments θα πρέπει να τα θεωρούμε ως κάποιο είδος Activity. Κάθε ένα από αυτά έχει μια λειτουργικότητα και ένα UI. Στην εικόνα 5.13, βλέπουμε ένα Fragment που αποτελείται από μία λίστα και ένα άλλο από ένα διαφορετικό layout. Σκεφτείτε την εφαρμογή να τρέχει σε μία ταμπλέτα όπου ο χώρος είναι μεγαλύτερος και τα δύο αυτά Fragments μπορούν να προβληθούν το ένα δίπλα στο άλλο, μέσα στην ίδια Activity. Είναι φανερό ότι με την μέθοδο αυτήν μπορούμε να

έχουμε μια πιο συγκροτημένη μορφή του UI της εφαρμογής. Πέραν βέβαια των οπτικών προτερημάτων, η χρήση των Fragments προσδίδει και δυναμικό χαρακτήρα σε μια εφαρμογή μιας και μπορούν να αφαιρούνται ή να προστίθενται δυναμικά.

Για την υλοποίηση της δυνατότητας του χρήστης μας να αναπαράγει ένα γράφημα σχετικά με τις ώρες που κατανάλωσε για ένα μάθημα, κάναμε χρήση της μεθοδολογίας των Fragment. Για τη δημιουργία ενός Fragment σε μια εφαρμογή Android, η συνήθης πρακτική είναι να επεκτείνουμε την κλάση Fragment και να υπερκαλύπτουμε τις κατάλληλες μεθόδους της.

```
public class ChartActivityextends Fragment {
@Override
publicvoidonActivityCreated(Bundle savedInstanceState) {
super.onActivityCreated(savedInstanceState);
//--einai poly basiko mesa se fragment panta getActivity()--

            }
@Override
publicView onCreateView(LayoutInflaterinflater,ViewGroup container, Bundle
savedInstanceState) {
      //---Inflate the layout for this fragment---
      returninflater.inflate(R.layout.chart, container, false);
      }

@Override
publicvoidonStart() {
                  super.onStart();
      }
}
```

Ένα Fragment συμπεριφέρεται όμοια με μία Activity, αποτελείται από μια κλάση γραμμένη σε Java ,καθώς επίσης και από ένα αρχείο γραμμένο σε XML το οποίο είναι υπεύθυνο για το UI του. Όπως έχουμε τονίσει η κλάση του Fragment επεκτείνεται σε Fragment, όπως φαίνεται στον πιο πάνω κώδικα, και υπερκαλύπτει τις διάφορες μεθόδους της. Μία από τις μεθόδους αυτές είναι και η onCreateView(), η οπόια είναι και αυτή που επιστρέφει ένα view. Η μέθοδος αυτή παίρνει ως όρισμα ένα αντικείμενο τύπου LayoutInflaterinflater, το οποίο είναι απαραίτητο για να δημιουργήσει το UI του Fragment από το αρχείο που το περιλαμβάνει, στην προκειμένη το chart.xml. Το όρισμα container αναφέρεται στο μητρικό ViewGroup,

μέσα στο οποίο θέλουμε να προσαρμόσουμε το συγκεκριμένο Fragment. Τέλος το αντικείμενο savedInstanceState τύπου Bundle, μας δίνει την δυνατότητα να αποθηκεύσουμε το Fragment στην προηγούμενη κατάσταση του.

Η επόμενη μέθοδος που είναι onActivityCreated(), εκτελείται αμέσως μετά την onCreateView(). Αφού πλέον το Fragment είναι σε ορατή κατάσταση, δηλαδή έχει πάρει μορφή, στην συνέχεια σειρά παίρνει η μέθοδος onStart().

Ωστόσο ο ορισμός του Fragment που περιγράψαμε προηγουμένως , είναι η μισή από την δουλεία που πρέπει να κάνουμε προκειμένου να προσαρτήσουμε την μεθοδολογία αυτή στην εφαρμογή μας. Την άλλη μισή λειτουργία αναλαμβάνει να την εκτελέσει η Activity με το όνομα GenericReport, στο layout της οποίας μέσα θα προβληθεί το Fragment. Για το σκοπό αυτό θα πρέπει μέσα στο layout της να κρατήσει χώρο για τον σκοπό αυτό. Παρακάτω δίνουμε μέρος του κώδικα του XML της GenericReport.

```
<?xmlversion="1.0"encoding="utf-8"?>
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
>
<ScrollView
android:id="@+id/scroll"
android:layout_height="fill_parent"
android:layout_width="fill_parent"
android:fillViewport="true">
<RelativeLayout
android:id="@+id/scrcont"
android:layout_width="fill_parent"

android:layout_height="wrap_content"

 .
<FrameLayout
android:id="@+id/statist"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_marginTop="20dp"
android:background="@color/white"
android:layout_below="@+id/subComm"></FrameLayout>

</RelativeLayout><!--telosscrcont -->
</ScrollView>
</RelativeLayout>
```

Τα περισσότερα από τα στοιχεία του κώδικα είναι γνωστά από την περιγραφή τους σε παραπάνω ενότητες για τον λόγο αυτό και δεν παραθέτουμε ολόκληρο το αρχείο, εκεί που αρχίζει βέβαια να έχει περισσότερο ενδιαφέρον είναι σε αυτό το πεδίο με το όνομα FrameLayout. Το FrameLayout όπως και αναφέραμε όταν περιγράφαμε το UI, αποτελεί ένα container μέσα στο οποίο μπορούν να υπάρχουν διάφορες views η μία πάνω στην άλλη. Στην περίπτωση μας όμως είναι ένα container μέσα στο οποίο θα προβληθεί το Fragmentμε το όνομα ChartActivity.

Η οθόνη μέσα στην οποία προβάλλεται το διάγραμμα, εκτελείται από το αρχείο με όνομα GenericReport.java, ο κώδικας του οποίου που σχετίζεται με την προβολή του fragment φαίνεται παρακάτω.

```
FragmentManagerfragmentManager;
FragmentTransactionfragmentTransaction;
```

```
publicvoiduploadfrag(){
        Log.v("mpainei", "stofrag");
        fragmentManager = getFragmentManager();
        fragmentTransaction =fragmentManager.beginTransaction();
        getSpinnerSelection();

//--dhmiourgw to bundle gia na steilw extra plhrofories sto frag poukalw
//--sxetika me tis epiloges tou xrhsth (einai basikos tropos apostolhs
plhroforiwn se frag apo activity)
```

```
Bundle sendchoice = newBundle();
sendchoice.putString("Seme", semeSele);
sendchoice.putString("Less", lessonSele);
frag1 = newChartActivity();
frag1.setArguments(sendchoice);
fragmentTransaction.replace(R.id.statist, frag1);
fragmentTransaction.commit();
// Log.v("mpainei", "oloklhrwnei");
}
```

Στον παραπάνω κώδικα παρατηρούμε ορισμένα συστατικά τα οποία δεν τα έχουμε αναφέρει σε προηγούμενες ενότητες. Για να προσθέσουμε ένα Fragment μέσα σε μια

Activity, χρειάζεται να χρησιμοποιήσουμε την κλάση FragmentManager, της οποίας και δημιουργούμε ένα στιγμιότυπο:

```
fragmentManager = getFragmentManager();
```

Θα πρέπει επίσης να χρησιμοποιήσουμε και την κλάση FragmentTransaction, για να μπορέσουμε να προσθέτουμε ή να αφαιρούμε Fragments από την Activity μας. Παρατηρούμε μέσα στην μέθοδο uploadfrag(), την διαδικασία η οποία ακολουθείται για την προσθήκη ενός Fragment. Αρχικά ορίζουμε ένα αντικείμενο τύπου Chartactivity(), το οποίο είναι και το Fragment μας στην συνέχεια θέλουμε το αντικείμενο αυτό να το επικολλήσουμε στην FrameLayout με idstatist.

```
fragmentTransaction.replace(R.id.statist, frag1);
fragmentTransaction.commit();
```

Ο παραπάνω κώδικας αρχικά προσθέτει το fragment μας με το όνομα frag1 στο FrameLayout με id statist, και στην συνέχεια εκτελεί την .commit(), για να λάβει το γεγονός χώρα. Είναι σημαντικό να πούμε ότι δεν θα υπάρξει καμία επικόλληση του Fragment αν δεν εκτελεστεί η commit().

5.16. Πακετάρισμα μιας εφαρμογής

Έχοντας ολοκληρώσει την ανάπτυξη του κώδικα της εφαρμογής, μπορούμε να την διαθέσουμε στο ευρύ κοινό ανεβάζοντας την στο GooglePlay. Τα βήματα τα οποία θα πρέπει να ακολουθήσουμε είναι τα εξής:

- Από το eclipse επιλέγουμε **File->Export…** και στον διάλογο που θα εμφανιστεί επιλέγουμε **ExportAndroidApplication** το οποίο βρίσκεται μέσα στον φάκελο **Android**.

- Στο επόμενο διάλογο μας ζητείται η διαδρομή προς το δικό μας ηλεκτρονικό κλειδί που θα χρησιμοποιήσουμε για την ψηφιακή υπογραφή της εφαρμογής μας. Για να είμαστε σε θέση να κάνουμε publish εφαρμογή στο GooglePlay, θα οπωσδήποτε να την έχουμε πρώτα υπογράψει με το προσωπικό μας κλειδί.

- Έχοντας δημιουργήσει το προσωπικό μας κλειδί, το eclipse θα υπογράψει ψηφιακά την εφαρμογή μας και θα την πακετάρει στην διαδρομή που θα του υποδείξουμε ως ένα αρχείο με κατάληξη .apk.

Από την στιγμή αυτή το αρχείο είναι έτοιμο να διανεμηθεί σε όλους όσους διαθέτουν Android συσκευή. Σε περίπτωση βέβαια που θέλουμε να δημοσιοποιήσουμε την εφαρμογή μας, στο ευρύ κοινό θα πρέπει να ακολουθήσουμε τα εξής βήματα:

- Θα πρέπει να δημιουργήσουμε έναν λογαριασμό Developer με την Google. Η δημιουργία ενός τέτοιου λογαριασμού είναι μια εύκολη διαδικασία η οποία απαιτεί εφάπαξ πληρωμή του ποσού των 25 δολαρίων.

- Αν επιθυμούμαι η εφαρμογή μας να διατίθεται με κάποιο αντίτιμο, θα πρέπει επίσης να διαθέτουμε έναν λογαριασμό με κάποιον συμβεβλημένο μηχανισμό ηλεκτρονικών συναλλαγών.

Όσοι βέβαια ανησυχούν σχετικά με τα πνευματικά δικαιώματα των εφαρμογών τους θα πρέπει να γνωρίζουν πως σύμφωνα με την συνθήκη της Βέρνης η οποία ισχύει στο μεγαλύτερο κομμάτι του πλανήτη, το οποιοδήποτε ηλεκτρονικό υλικό που μοιράζεται από κάποιο μέσο, αυτομάτως γίνεται copyrighted. Παρόλα αυτά, αποτελεί

κοινή πρακτική σε κάθε εφαρμογή ο προγραμματιστής να χρησιμοποιεί το σύμβολο © μαζί με μια ημερομηνία. Σε εφαρμογές που περιλαμβάνουν και κάποιου είδους πληρωμής, αποτελεί συνηθισμένη πρακτική να περιλαμβάνουν και κάποιο είδος συμφωνίας, η οποία καθορίζει τους όρους οι οποίοι θα διέπουν τη σύμβαση μεταξύ του χρήστη και του προγραμματιστή.

Βιβλιογραφία

Έχουν εκδοθεί πολλά βιβλία σχετικά με το Android. Για τις ανάγκες της διπλωματικής εργασίας χρησιμοποιήσαμε αρκετά από αυτά τα οποία και προτείνουμε για περαιτέρω ανάγνωση.

I. Beginning Android™ 4 Application Development Published by John Wiley & Sons, Inc**.ISBN**: 978-1-118-19954-1.

II. Programming Android **by ZigurdMednieks, Laird Dornin, G. Blake Meike**, and **Masumi Nakamura**.PublishedbyO'ReillyMedia.**ISBN**: 978-1-449-38969-7.

III. The Android developer's cookbook : building applications with the Android SDK by **James Steele, Nelson To**. Published by Addison Wesley**.ISBN**-13: 978-0-321-74123-3 (pbk. :alk. paper).**ISBN**-10: 0-321-74123-4 (pbk. :alk. paper).

IV. Sams Teach Yourself Android™ Application Development in 24 Hours by **Lauren Darcey, Shane Conder**.**ISBN** 978-0-321-67335-0 (pbk.).

V. Android User Interface Development by**Jason Morris.**Published by Packt Publishing Ltd.**ISBN** 978-1-849514-48-4.

VI. The Business of Android Apps Development: Making and Marketing Apps That Succeedby **Mark Rollins.** Published by Apress.**ISBN**-13 (pbk): 978-1-4302-3942-0.**ISBN**-13 (electronic): 978-1-4302-3943-7

Εκτός από τα βιβλία που αναφέρθηκαν, ένα μεγάλο μέρος της προσπάθειας μας στηρίχθηκε σε πηγές του διαδικτύου ορισμένες από τις οποίες σας τις επισυνάπτουμε με σκοπό την περαιτέρω μελέτη τους.

1. Ανάλυση της αρχιτεκτονικής του Android και ανάλυση των βασικών συστατικών μιας εφαρμογής του Android, καθώς και ο συσχετισμός τους με την αρχιτεκτονική του συστήματος
   http://developer.android.com/guide/basics/what-is-android.html

2. Εξέλιξη του Android
   http://developer.android.com/about/dashboards/index.html

3. Πληροφορίες σχετικά με τη δομή και τη χρήση του του αρχείου AndroidManifest.xml
   http://developer.android.com/guide/topics/manifest/manifest-intro.html

4. Οι φάσεις του σχεδιασμού τις εφαρμογής
   http://developer.android.com/guide/developing/index.html

5. Οδηγίες εγκατάστασης του Android SDK
   http://developer.android.com/sdk/installing.html

6. Προτεινόμενο εργαλείο για υλοποίηση του project
   http://www.eclipse.org

7. Οδηγίες εγκατάστασης του ADT plugin
   http://developer.android.com/sdk/eclipse-adt.html#installing

8. Ο καταμερισμός των εκδόσεων του Android όπως αυτά καταγράφονται από την πρόσβαση των συσκευών στο Google Play Store.
   http://developer.android.com/resources/dashboard/platform-versions.html

9. Ο καταμερισμός των διαστάσεων οθόνης προς την πυκνότητα pixel των συσκευών
   http://developer.android.com/resources/dashboard/screens.html

10. Οδηγίες υποστήριξης πολλαπλών αναλύσεων στις εφαρμογές
    http://developer.android.com/guide/practices/screens_support.html

11. Διαχείριση εικονικών συσκευών
    http://developer.android.com/guide/developing/devices/index.html

12. Αποθήκευση δεδομένων
    http://developer.android.com/guide/topics/data/data-storage.html