

ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ

ΙΔΡΥΜΑ ΛΑΡΙΣΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Το λειτουργικό σύστημα Android και η υλοποίηση εφαρμογής με το
Google SDK**

ΣΤΥΛΙΑΝΟΣ ΒΟΥΚΑΤΑΣ



ΕΠΙΒΛΕΠΩΝ: ΣΩΜΑΡΑΣ ΧΡΗΣΤΟΣ

ΛΑΡΙΣΑ 2011

ΣΤΥΛΙΑΝΟΣ ΒΟΥΚΑΤΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

**Το λειτουργικό σύστημα Android και η υλοποίηση
εφαρμογής με το Google SDK**

ΕΠΙΒΛΕΠΩΝ: ΣΩΜΑΡΑΣ ΧΡΗΣΤΟΣ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Λαρισα,2011

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Σωμαράς Χρήστος

2.

3.

Η Google με την Open Handset Alliance [1] η οποία είναι μια «συμμαχία» πολλών εταιριών (όπως htc, e-bay, Acer, κτλ) και έχει ως πρωτοπόρο την Google, ανέπτυξε μια ολοκληρωμένη πλατφόρμα για κινητές συσκευές η οποία ονομάζεται Android [2]. Αυτή περιλαμβάνει το λειτουργικό σύστημα, την ενδιάμεση πλατφόρμα σύνδεσης των εφαρμογών με το λειτουργικό σύστημα (middleware) καθώς και τις αντίστοιχες εφαρμογές και το γραφικό τους περιβάλλον. Βασικό χαρακτηριστικό είναι πως μπορεί να εγκατασταθεί και να λειτουργήσει σε οποιαδήποτε κινητή συσκευή ανεξάρτητα από τον κατασκευαστή αυτής.

Στα παρακάτω κεφάλαια θα αναλύσουμε την πλατφόρμα αυτή και θα δούμε τον τρόπο με τον οποίο λειτουργεί ως λειτουργικό σύστημα αλλά και τον τρόπο με τον οποίο μπορούμε να αναπτύξουμε εφαρμογές γι' αυτήν. Τέλος θα δημιουργήσουμε μία εφαρμογή για το λειτουργικό Android, την οποία θα τη δοκιμάσουμε σε διάφορες εικονικές συσκευές με διάφορες εκδόσεις της πλατφόρμας αλλά και σε πραγματικά τηλέφωνα με το παραπάνω λειτουργικό.

ΕΥΧΑΡΙΣΤΙΕΣ

Κατ' αρχάς θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της πτυχιακής μου εργασίας κ. Χρήστο Σωμαρά για τη καθοδήγηση του σε κάθε φάση της δημιουργίας της επίσης θα ήθελα να ευχαριστήσω την μεταδιδακτορική ερευνήτρια του Athens Information Technology Ξένια Ζιούβελου για τις πολύτιμες συμβουλές της. Τέλος θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου, που όλα αυτά τα χρόνια μου συμπαραστέκονται ηθικά και οικονομικά και διαμορφώνουν γύρω μου ένα άνετο περιβάλλον, μέσα στο οποίο μπορώ να εργαστώ και να επεκτείνω τις γνώσεις μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: Εισαγωγή	ix
ΚΕΦΑΛΑΙΟ 2: Περιγραφή του λειτουργικού συστήματος Android	xii
2.1. Αρχιτεκτονική του Android.....	xiii
2.2. Ιστορικό εκδόσεων του Android	xvi
ΚΕΦΑΛΑΙΟ 3: Σύγκριση του Android με άλλα λειτουργικά συστήματα	xviii
3.1. Η αγορά των λειτουργικών συστημάτων	xix
3.2. Κριτήρια κατηγοριοποίησης.....	xx
3.3. Σύγκριση του Android με το Symbian OS και το Windows Mobile	xxiii
3.3.1. Ικανότητα μεταφοράς - Portability.....	xxiii
3.3.2. Αξιοπιστία - Reliability	xxiv
3.3.3. Συνδεσιμότητα - Connectivity.....	xxv
3.3.4. Διαφοροποίηση - Diversity.....	xxvi
3.3.5. Ανοικτό σύστημα - Open.....	xxvi
3.3.6. Πυρήνας - Kernel.....	xxvii
3.3.7. Πρότυπα - Standards	xxviii
3.3.8. Ασφάλεια - Security	xxviii
3.4. Συγκρίσεις- Διαπιστώσεις	xxix
ΚΕΦΑΛΑΙΟ 4: Περιβαλλον ανάπτυξης εφαρμογών	xxxiii
4.1 Λήψη και εγκατάσταση	xxxiv
4.2 Δημιουργία εικονικής συσκευής Android	xxxv
Επεξήγηση των ρυθμίσεων.....	xxxviii
4.3 Δημιουργία νέου project.....	xxxviii

ΚΕΦΑΛΑΙΟ 5: Το Android ως πλατφόρμα ανάπτυξης εφαρμογών.....	xli
Κεφάλαιο 5.1: Ανάλυση των καταλογων σε ένα Android project	xli
5.1.1 Ο καταλόγος src.....	xlii
5.1.2 Ο κατάλογος res.....	xliii
5.1.3 Ο κατάλογος gen	xliv
5.1.4 το xml αρχείο androidmanifest	xlv
5.2 Δομικά στοιχεία του android sdk	l
5.3 Κύκλος ζωής του activity	liii
5.4 Τρόποι δημιουργίας διεπαφής χρήστη – user interface	lv
5.4.1 Θεμελιώδεις κλάσεις για τον σχεδιασμό διεπαφής χρήστη – user interface.....	lv
5.4.2 Μελέτη των μεθόδων σχεδίασης διεπαφής χρήστη – user interface.....	lvi
5.5 Πρότυπες προβολές - Views.....	lxi
5.6 Διαταξεις - Layouts	lxii
5.7 Επισημάνεις	lxvi
ΚΕΦΑΛΑΙΟ 6: Υποστήριξη πολλαπλών οθονών	lxvii
6.1 Όροι και έννοιες	lxvii
6.2 Τρόποι υποστήριξης	lxviii
6.3 Περεταίρω αναλυση για την Προ-κλιμάκωση και την αυτόματη κλιμάκωση των εικόνων	lxxiii
ΚΕΦΑΛΑΙΟ 7: Εντοπισμός σφαλμάτων (Debugging)	lxxiv
7.1 Το εργαλείο Android Debug Bridge (adb)	lxxv
7.2 Το εργαλείο Dalvik Debug Monitor server (ddms).....	lxxvii
7.3 Το εργαλείο Traceview.....	lxxix

7.4 Το εργαλείο logcat.....	lxxx
7.5 Το εργαλείο hierarchy viewer.....	lxxxiii
7.5 Σύνοψη	lxxxv
ΚΕΦΑΛΑΙΟ 8: Δοκιμή – testing των εφαρμογών	lxxxvi
ΚΕΦΑΛΑΙΟ 9: Η android εφαρμογή «cocktail».....	lxxxviii
9.1 Περιγραφή της εφαρμογής «cocktail».....	lxxxviii
ΚΕΦΑΛΑΙΟ 10: Συμπεράσματα	xcv
Βιβλιογραφία	xcvi

ΕΙΣΑΓΩΓΗ

Στην παρούσα πτυχιακή εργασία, περιγράφεται και μελετάται το Software Development Kit (SDK) του σύγχρονου λειτουργικού συστήματος Android της Google. Επίσης μελετάται η αρχιτεκτονική του συστήματος και αναλύονται τα σημαντικότερα τμήματα του.

Κύριος σκοπός της μελέτης είναι να υλοποιηθεί μία εφαρμογή η οποία θα διέπεται από τους κανόνες του αναφερομένου συστήματος με στόχο να λειτουργεί γενικότερα σε συσκευές που πού υποστηρίζονται από το λειτουργικό Android.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν τεχνολογίες Java (JSE), βάσεων δεδομένων (SQLite) και το SDK του Android. Η ανάπτυξη έγινε με το Eclipse IDE και το plug in που παρέχεται για το Android. Τα γραφικά επεξεργάστηκαν / σχεδιάστηκαν με το πρόγραμμα Gimp. Η εφαρμογή αναπτύχθηκε και δοκιμάστηκε στις εξής συσκευές:

- Σε διάφορες εικονικές συσκευές που μας παρέχει το SDK
- Σε κινητό τηλέφωνο HTC Desire με λειτουργικό Android 2.2
- Σε κινητό τηλέφωνο Sony Ericsson xperia x10 mini pro με λειτουργικό 2.1

Επί πλέον μπορούμε να αναφέρουμε ότι ένας από τους βασικούς στόχους της Google, που θέλησε να πετύχει μέσω του Android, είναι η εύκολη λήψη και χρήση νέων εφαρμογών στις κινητές συσκευές των χρηστών. Ωστόσο, βασική προϋπόθεση για την επίτευξη αυτού του στόχου είναι α) οι χρήστες να γνωρίζουν τις δυνατότητες της κινητής τους συσκευής και β) να θέλουν να τις χρησιμοποιήσουν λαμβάνοντας υπόψη πως η χρήση δικτύων δεδομένων για πρόσβαση στο Διαδίκτυο μπορεί να επιφέρει επιπλέον χρεώσεις. Για να μπορέσουμε να εκτιμήσουμε τις ανάγκες των χρηστών θα πρέπει να έχουμε υπόψη τους διαφορετικούς τύπους χρηστών οι οποίοι μπορεί να είναι:

- Ο βασικός χρήστης που χρησιμοποιεί μόνο τις απλές εφαρμογές της συσκευής.

- Ο μέσος χρήστης που χρησιμοποιεί τόσο τις παραδοσιακές λειτουργίες όσο και διάφορες εφαρμογές.
- Ο προχωρημένος χρήστης που έχει πλήρη επίγνωση του περιβάλλοντος της συσκευής και χρησιμοποιεί την πλειονότητα των προχωρημένων εφαρμογών.

Κάθε τύπος χρήστη έχει και διαφορετικές ανάγκες και προσδοκίες από την συσκευή του και φυσικά είναι διατεθειμένος να καταβάλει και το αντίστοιχο κόστος για να τις καλύψει. Έτσι προκύπτει το ερώτημα εάν το Android είναι σε θέση να καλύψει τις ανάγκες κάθε τύπου χρήστη. Λαμβάνοντας υπόψη τα παραπάνω, η δομή της εργασίας είναι η παρακάτω.

Κεφάλαιο 2

Στο κεφάλαιο αυτό θα περιγραφούν τα βασικά στοιχεία της αρχιτεκτονικής του λειτουργικού συστήματος Android και θα παρουσιαστούν τα βασικά χαρακτηριστικά παλαιότερων εκδόσεων του λειτουργικού συστήματος έως και τη σημερινή έκδοση.

Κεφάλαιο 3

Στο κεφάλαιο αυτό το λειτουργικό σύστημα Android συγκρίνεται με άλλα δύο εξίσου σημαντικά λειτουργικά συστήματα τα οποία κατέχουν μεγάλο μερίδιο στην αγορά των κινητών συσκευών. Αρχικά, παρουσιάζονται μερικά σημαντικά στοιχεία που σχετίζονται με την αγορά των λειτουργικών συστημάτων. Κατόπιν καθορίζονται ορισμένα κριτήρια βάσει των οποίων συγκρίνονται τα λειτουργικά συστήματα. Στο τέλος του κεφαλαίου παρουσιάζονται οι σημαντικότερες διαπιστώσεις που προέκυψαν από την σύγκριση αυτή.

Κεφάλαιο 4

Στο κεφάλαιο αυτό παρουσιάζεται το Eclipse που είναι το περιβάλλον που χρησιμοποιήσαμε για την ανάπτυξη της εφαρμογής. Επί πλέον παρουσιάζεται ο τρόπος κατασκευής ενός εξομοιωτή αλλά και η διαδικασία δημιουργίας ενός νέου project.

Κεφάλαιο 5

Στο κεφάλαιο αυτό θα εξετάσουμε το λειτουργικό σύστημα Android ως μία πλατφόρμα ανάπτυξης εφαρμογών. Θα αναλύσουμε τους καταλόγους που περιέχει

ένα Android project, θα περιγράψουμε μερικά από τα δομικά στοιχεία που καθορίζονται από το SDK, θα εμβαθύνουμε στον τρόπο λειτουργίας του Activity (Δραστηριότητα) και στον σημαντικό ρόλο που έχει ο κύκλος ζωής του και τέλος θα μελετήσουμε τους τρόπους δημιουργίας του user interface (διεπαφή χρήστη).

Κεφάλαιο 6

Στο κεφάλαιο αυτό θα αναπτύξουμε διάφορες έννοιες που χρειάζονται για να κατανοήσουμε τους τρόπους με τους οποίους μπορούμε να πετύχουμε τη σωστή εμφάνιση της εφαρμογής μας σε πολλές διαφορετικές οθόνες. Επίσης θα εξηγήσουμε και θα εξετάσουμε αυτούς τους τρόπους.

Κεφάλαιο 7

Στο κεφάλαιο αυτό θα μελετήσουμε τα εργαλεία που μας παρέχει το Android SDK για την διευκόλυνση μας στον εντοπισμό σφαλμάτων αλλά και για την καταγραφή της απόδοσης της εφαρμογής μας.

Κεφάλαιο 8

Στο κεφάλαιο αυτό θα μελετήσουμε τον τρόπο με τον οποίο πρέπει να δοκιμάζουμε τις εφαρμογές μας και θα παρουσιάσουμε ενδεικτικά ένα πίνακα με ορισμένους εξομοιωτές /εικονικές συσκευές που πρέπει να δοκιμάζουμε τις εφαρμογές μας ώστε να δούμε αν λειτουργούν σωστά.

Κεφάλαιο 9

Στο κεφάλαιο αυτό θα γίνει η παρουσίαση της εφαρμογής που υλοποιήσαμε για τον σκοπό αυτής της πτυχιακής.

Κεφάλαιο 10

Στο κεφάλαιο αυτό θα αναφέρουμε τα συνολικά συμπεράσματα που αποκομίσαμε από την μελέτη αυτή και επιπλέον θα αναφέρουμε τις προοπτικές εξέλιξης της εφαρμογής που δημιουργήσαμε.

ΚΕΦΑΛΑΙΟ 2: ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ANDROID

Ο όρος Android έχει ελληνική προέλευση καθώς προέρχεται από την λέξη ανδρ- που έχει την έννοια του άνδρα ή του ανθρώπου και την κατάληξη -ειδές που χαρακτηρίζει κάποιο είδος. Συνεπώς η έννοια που δίδεται στην λέξη Android είναι το «Ανδροειδές» και συμβολίζει το ρομπότ με μορφή ανθρώπου. Στην 2.1 εμφανίζονται και τα αντίστοιχα λογότυπα του λειτουργικού συστήματος για τις εκδόσεις 2.2 και 2.3. 0.



(α)



(β)

Εικόνα 2.1: Τα λογότυπα του λειτουργικού συστήματος Android

(α) στην έκδοση 2.2 και (β) στην έκδοση 2.3

Στις παρακάτω ενότητες θα παρουσιαστεί η αρχιτεκτονική του Android και οι συνιστώσες λογισμικού από τις οποίες αποτελείται. Επιπλέον θα παρουσιαστεί σύντομα η εξέλιξή του ξεκινώντας από τις προγενέστερες εκδόσεις και καταλήγοντας στην τρέχουσα.

2.1. Αρχιτεκτονική του Android

Το Android αποτελείται από ορισμένες συνιστώσες λογισμικού οι οποίες συνθέτουν ένα ενιαίο και ολοκληρωμένο σύστημα. Έτσι, το σύστημα αυτό μπορεί να παρέχει τα μέσα που απαιτούνται για την χρήση νέων εφαρμογών όπως άλλωστε συμβαίνει και με τα λειτουργικά συστήματα των ηλεκτρονικών υπολογιστών. Όπως φαίνεται και στην

Εικόνα 2.2 0, το Android αποτελείται από 4 επίπεδα και από 5 ομάδες συνιστωσών τα οποία περιγράφονται παρακάτω ξεκινώντας από τα χαμηλότερα προς τα υψηλότερα επίπεδα:

Linux Kernel

Το Android βασίζεται στον πυρήνα του Linux για βασικές λειτουργίες όπως είναι η διαχείριση των drivers της συσκευής, διαχείριση μνήμης, διαχείριση διεργασιών καθώς και δικτύωσης που συνεπάγεται την διαχείριση των διεπαφών δικτύου που διαθέτει κάθε συσκευή (πχ. GSM, HSDPA, WiFi, Bluetooth κλπ).

Native Libraries

Οι βιβλιοθήκες του Android είναι γραμμένες στις γλώσσες C και C++ και μπορούν να χρησιμοποιηθούν μέσω κατάλληλου interface της Java. Μερικές από τις κυριότερες είναι α) η βιβλιοθήκη Surface Manager για την δημιουργία παραθύρων καθώς και δισδιάστατων (2D) και τρισδιάστατων (3D) γραφικών, β) η βιβλιοθήκη Media Framework που περιέχει αποκωδικοποιητές (codecs) για αναπαραγωγή αρχείων πολυμέσων όπως MPEG, MP3 κλπ, γ) η βιβλιοθήκη SQLite για την υποστήριξη της βάσης δεδομένων SQL και δ) η βιβλιοθήκη WebKit για την υποστήριξη των φυλλομετρητών (browsers).

Android Runtime

Όπως φαίνεται και στην παρακάτω εικόνα η συνιστώσα του Android Runtime αποτελείται από:

- Βασικές βιβλιοθήκες για την διεπαφή των εφαρμογών Java με το περιβάλλον της συσκευής στην οποία εκτελούνται.
- Τη Dalvik Virtual Machine η οποία είναι υπεύθυνη για την δημιουργία των εκτελέσιμων αρχείων των εφαρμογών προκειμένου να τα «τρέξει» το λειτουργικό σύστημα.

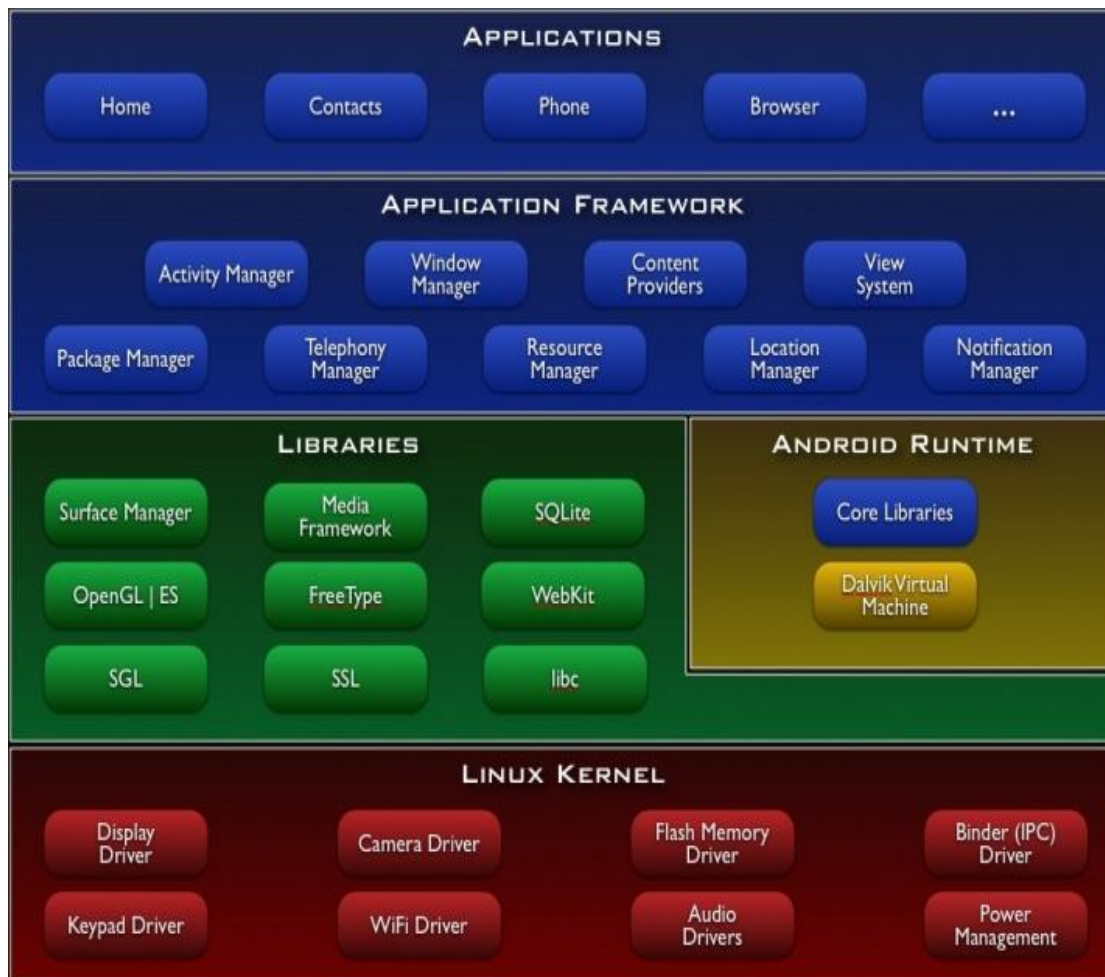
Κάθε εφαρμογή του Android είναι γραμμένη σε γλώσσα Java την οποία το λειτουργικό σύστημα δεν την αντιλαμβάνεται απευθείας. Για τον λόγο αυτό η Dalvik Virtual Machine αναλαμβάνει τη δημιουργία των εκτελέσιμων αρχείων *.dex (Dalvik Executable) τα οποία εκτελούνται από το λειτουργικό σύστημα. Κάθε εκτελέσιμο πρόγραμμα εκτελείται από την δική του Virtual Machine, ακόμα και όταν εκτελούνται παράλληλα, με αποτέλεσμα τα διαφορετικά προγράμματα να μην επηρεάζουν το ένα το άλλο και σε περίπτωση που προκύψει κάποιο σφάλμα σε κάποιο από αυτά να μην προκαλέσει προβλήματα στα υπόλοιπα.

Application Framework

Εφόσον το Android προσφέρει μια ανοικτή πλατφόρμα ανάπτυξης εφαρμογών είναι επόμενο ορισμένες από τις εφαρμογές να είναι αρκετά προχωρημένες και καινοτόμες. Οι εφαρμογές έχουν πρόσβαση στις βασικές βιβλιοθήκες του λειτουργικού συστήματος, μέσω κατάλληλων διεπαφών, και μέσω του Application Framework μπορούν με τη σειρά τους να παρέχουν επιπρόσθετες λειτουργίες-υπηρεσίες προς άλλες εφαρμογές, εφόσον κάτι τέτοιο φυσικά δεν περιορίζεται από τις πολιτικές ασφαλείας του Application Framework. Μερικές από τις πιο βασικές οντότητες που περιλαμβάνονται στο πλαίσιο του Application Framework είναι:

- **View System:** Επιτρέπει την χρήση λιστών, πλαισίων, πεδίων κειμένου, κουμπιών κλπ.

- **Content Providers:** Επιτρέπει στις εφαρμογές την πρόσβαση σε δεδομένα άλλων εφαρμογών ή τον διαμοιρασμό των δικών τους δεδομένων, όπως οι επαφές.
- **Resource Manager:** Παρέχει την πρόσβαση σε πόρους όπως γραφικά και σε αρχεία σχετικά με την διάταξη των στοιχείων του γραφικού περιβάλλοντος. Απλούστερα, ότι δεν είναι κώδικας.
- **Notification Manager:** Διαχειρίζεται τα μηνύματα των εφαρμογών που εμφανίζονται στην status bar, όπως εισερχόμενα μηνύματα, ραντεβού κτλ.
- **Activity Manager:** Διαχειρίζεται τον κύκλο ζωής των εφαρμογών και παρέχει την δυνατότητα μετάβασης στις προγενέστερες καταστάσεις τους.



Εικόνα 2.2: Συνιστώσες λογισμικού λειτουργικού συστήματος Android

Applications

Στην ομάδα των Applications βρίσκονται οι εφαρμογές που θα χρησιμοποιούν τελικά οι χρήστες με διαφάνεια ως προς το τι συμβαίνει πίσω από αυτές ή το τι απαιτείται

για την εκτέλεσή τους από το λειτουργικό σύστημα. Μερικές από τις πιο γνωστές από τις εφαρμογές αυτές είναι ο browser, email client, αποστολή και λήψη SMS, προβολή χαρτών σε συνδυασμό με το στίγμα της συσκευής εάν διαθέτει δέκτη GPS, ημερολόγιο, διαχείριση επαφών, παιχνίδια, RSS readers και πολλές άλλες. Όλες οι εφαρμογές όπως έχει ήδη αναφερθεί είναι γραμμένες σε Java και μπορούν να τρέχουν πολλές παράλληλα χωρίς να επηρεάζει η μια την άλλη.

2.2. Ιστορικό εκδόσεων του Android

Μετά τις αρχικές εκδόσεις του Android ακολούθησαν ενημερώσεις πάνω στην βασική δομή του. Σκοπός των ενημερώσεων αυτών ήταν είτε να διορθώσουν δυσλειτουργίες που είχαν ανακαλυφθεί έως τότε είτε να προσθέσουν νέες λειτουργίες. Χαρακτηριστικό στοιχείο των εκδόσεων του Android είναι πως κάθε μία από αυτές φέρει κωδικό όνομα το οποίο είναι ένα όνομα γλυκού και εμφανίζονται κατά αλφαβητική σειρά ανά έκδοση. Στον παρακάτω πίνακα εμφανίζονται τα βασικά στοιχεία κάθε έκδοσης 0.

Πίνακας 1: Πίνακας εκδόσεων του λειτουργικού συστήματος Android

Στοιχεία έκδοσης	Χαρακτηριστικά
Έκδοση: 1.0 Κωδικός: - Ημερομ.: 23 Σεπτεμβρίου 2008	
Έκδοση: 1.1 Κωδικός: - Ημερομ.: 9 Φεβρουαρίου 2009	
Έκδοση: 1.5 Κωδικός: Cupcake Ημερομ.: 30 Απριλίου 2009 Linux Kernel: 2.6.27	Νέες λειτουργίες για την κάμερα της συσκευής, προσθήκη έξυπνου πληκτρολογίου, νέες λειτουργίες για τις συνδέσεις Bluetooth, νέο γραφικό περιβάλλον και νέα λειτουργία άμεσης δημοσίευσης

Στοιχεία έκδοσης	Χαρακτηριστικά
	αρχείων βίντεο σε γνωστές σελίδες όπως πχ. YouTube.
Έκδοση: 1.6 Κωδικός: Donut Ημερομ.: 15 Σεπτεμβρίου 2009 Linux Kernel: 2.6.29	Ταχύτερη απόκριση, νέες λειτουργίες για διαγραφή πολλών αρχείων ταυτόχρονα, αναζήτηση και κλήση μέσω φωνητικών εντολών, έξυπνες λειτουργίες αναζήτησης, προσθήκη νέων πομποδεκτών, νέα πλατφόρμα ανάπτυξης.
Έκδοση: 2.0 / 2.1 Κωδικός: Eclair Ημερομ.: 26 Οκτωβρίου 2009 Linux Kernel: 2.6.29	Ταχύτερη απόκριση του υλικού, υποστήριξη νέων οθονών και αναλύσεων, νέο γραφικό περιβάλλον, νέος φυλλομετρητής, νέα έκδοση του Google Maps 3.1.2, υποστήριξη φλάς για κάμερες και ψηφιακού zoom, υποστήριξη multi-touch events, νέες λειτουργίες πληκτρολογίου, υποστήριξη του Bluetooth 2.1.
Έκδοση: 2.2 Κωδικός: Froyo Ημερομ.: 20 Μαΐου 2010 Linux Kernel: 2.6.32	Βελτιωμένη ταχύτητα του λειτουργικού και αποδοτικότερη λειτουργία, ενσωμάτωση του JavaScript μηχανισμού του φυλλομετρητή Chrome V8 στην εφαρμογή του Browser, μεγαλύτερη υποστήριξη του Microsoft Exchange, βελτιωμένες λειτουργίες εκκίνησης των εφαρμογών, ενσωμάτωση λειτουργιών για αυτόματη ενημέρωση των εφαρμογών και εγκατάσταση αυτών σε κάρτα μνήμης, υποστήριξη του Adobe Flash 10.1.
Έκδοση: 2.3 Κωδικός: Gingerbread Ημερομ.: 6 Δεκεμβρίου 2010 Linux Kernel: 2.6.35.7	Βελτιωμένη σχεδίαση του γραφικού περιβάλλοντος, υποστήριξη για οθόνες μεγάλων μεγεθών και αναλύσεων, υποστήριξη πρωτοκόλλου SIP για τηλεφωνία μέσω VoIP, υποστήριξη του WebM/VP8 τύπου βίντεο και κωδικοποιητή AAC, βελτίωση ήχου, υποστήριξη Near Field Communication και νέα λειτουργία System-wide copy-paste, νέο περιβάλλον ανάπτυξης και νέες λειτουργίες ήχου και απεικόνισης για ανάπτυξη παιχνιδιών,

Στοιχεία έκδοσης	Χαρακτηριστικά
	υποστήριξη αισθητήρων, βελτιωμένη διαχείριση ενέργειας, υποστήριξη πολλαπλών καμερών και μετάβαση από το σύστημα αρχείων YAFFS στο ext4.

ΚΕΦΑΛΑΙΟ 3: ΣΥΓΚΡΙΣΗ ΤΟΥ ANDROID ΜΕ ΆΛΛΑ ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Βασικό χαρακτηριστικό του Android είναι πως μπορεί να εγκατασταθεί και να λειτουργήσει σε οποιαδήποτε κινητή συσκευή ανεξάρτητα από τον κατασκευαστή αυτής. Σε αντίθεση με την αγορά των κινητών συσκευών στην οποία ανταγωνίζονται πολλοί κατασκευαστές, στον τομέα των λειτουργικών συστημάτων για κινητές συσκευές υπάρχουν μόνο δυο άλλοι κύριοι ανταγωνιστές που είναι το λειτουργικό σύστημα Symbian OS της Symbian και το λειτουργικό σύστημα Windows Mobile της Microsoft. Συνεπώς, ένας βασικός στόχος για την Google εάν θέλει να επιβιώσει στον τομέα αυτό και να ανταγωνιστεί επιτυχώς αυτά τα λειτουργικά συστήματα είναι να καταλάβει ένα σημαντικό μερίδιο αγοράς αντάξιο του μεγέθους της.

Στο κεφάλαιο αυτό θα συγκρίνουμε αυτά τα λειτουργικά συστήματα με το Android. Ωστόσο, πριν από αυτό κρίνεται σκόπιμο να αποσαφηνίσουμε τις παρακάτω έννοιες:

Λειτουργικό σύστημα

Είναι η οντότητα ενός υπολογιστικού συστήματος η οποία έχει τον ρόλο διεπαφής ανάμεσα στις εφαρμογές και το υλικό του. Η κύρια λειτουργία του είναι η διαχείριση των διαθέσιμων πόρων όπως για παράδειγμα η μνήμη, ο επεξεργαστής, ο σκληρός δίσκος καθώς και μονάδες εισόδου εξόδου όπως η οθόνη και το πληκτρολόγιο.

Κινητό σύστημα

Είναι το υπολογιστικό σύστημα το οποίο έχει την ικανότητα όχι μόνο να μετακινείται αλλά και να παραμένει λειτουργικό κατά την διάρκεια της μετακίνησής του. Χαρακτηριστικό παράδειγμα τέτοιου συστήματος αποτελούν τα κινητά τηλέφωνα τα οποία έχουν την ικανότητα είτε να εγκαταστήσουν την σύνδεση που επιθυμεί ο χρήστης όπου και αν βρίσκονται είτε να διατηρήσουν την σύνδεση του χρήστη (πχ. την ώρα που πραγματοποιεί κάποια κλήση) καθώς μετακινείται και μπορεί να αλλάξει κεραίες κατά την διαδρομή του.

3.1. Η αγορά των λειτουργικών συστημάτων

Οι κινητές συσκευές σήμερα έχουν αλλάξει δραματικά τα τελευταία χρόνια. Οι σύγχρονες συσκευές πέρα από τις παραδοσιακές δυνατότητες κλήσεων και αποστολής μηνυμάτων SMS, ενσωματώνουν πληθώρα λειτουργιών οι οποίες ήταν διαθέσιμες μόνο σε συσκευές Personal Digital Assistants (PDAs). Λαμβάνοντας υπόψη το σύνολο των λειτουργιών που μπορεί να είναι διαθέσιμες σε κάθε συσκευή, κρίνεται αναγκαίο να εξετάσουμε τα χαρακτηριστικά των λειτουργικών συστημάτων τα οποία είναι υπεύθυνα για την διαφοροποίησή τους.

Ένα από τα βασικότερα στοιχεία σχετικά με τα λειτουργικά συστήματα είναι η αγορά στην οποία θα συμμετέχουν και θα πρέπει να είναι ανταγωνιστικά έναντι άλλων λειτουργικών συστημάτων. Η αγορά των εξελιγμένων κινητών συσκευών δεν μπορεί να συγκριθεί εύκολα με άλλες αγορές όπως αυτή των ηλεκτρονικών υπολογιστών, όπου και εκεί υπάρχει ανταγωνισμός ανάμεσα στα λειτουργικά συστήματα, καθώς οι ανάγκες και οι απαιτήσεις των χρηστών ποικίλουν μεταξύ τους. Χαρακτηριστικό είναι το παράδειγμα της Symbian η οποία κατέληξε στην ανάπτυξη του λειτουργικού συστήματος Symbian OS καθώς η αγορά των κινητών συσκευών διαφοροποιείται έναντι άλλων αγορών λόγω των παρακάτω χαρακτηριστικών 0:

- **Mobility:** Οι κινητές συσκευές πέρα από το γεγονός ότι γίνονται ολοένα και μικρότερες παρέχουν και απεριόριστο βαθμό κινητικότητας στον χρήστη.

- **Universal:** Στόχος των κατασκευαστών κινητών συσκευών είναι να παράγουν μαζικά συσκευές που να καλύπτουν κάθε ανάγκη των χρηστών όσο εξεζητημένες και αν είναι αυτές.
- **Connection:** Το επίπεδο συνδεσιμότητας που παρέχεται από τις κινητές συσκευές είναι υψηλό καθώς έχουν την δυνατότητα να συνδέονται τόσο στις κεραίες του παρόχου του δικτύου τους οι οποίες μπορεί να βρίσκονται χιλιόμετρα μακριά όσο και σε δίκτυα μικρότερης εμβέλειας με κοντινότερες κεραίες (πχ WiFi). Επίσης, δεν είναι λίγες και οι εφαρμογές που λειτουργούν με τοπικές συνδέσεις από συσκευή σε συσκευή όπως (πχ Bluetooth).
- **Innovation:** Παρά το γεγονός ότι οι ανάγκες που πρέπει να καλύψουν οι κινητές συσκευές είναι ως επί των πλείστων κοινές για τους περισσότερους χρήστες, οι κατασκευαστές πρέπει να εισάγουν και καινοτομικά στοιχεία στις συσκευές τους δεδομένου ότι ο ανταγωνισμός είναι αυξημένος και στοχεύουν στο μεγαλύτερο δυνατό μερίδιο αγοράς.
- **Open:** Η πλατφόρμα του λειτουργικού συστήματος πρέπει να είναι ανοικτού κώδικα έτσι ώστε να μην περιορίζεται η ανάπτυξη εφαρμογών και η χρήση ανεξάρτητων ή/και διαφορετικών τεχνολογιών.

Τα παραπάνω χαρακτηριστικά είναι αυτά που διαφοροποιούν την αγορά των λειτουργικών συστημάτων για κινητές συσκευές έναντι άλλων αγορών. Η επιτυχία ενός λειτουργικού συστήματος έγκειται στην υποστήριξη των χαρακτηριστικών αυτών χωρίς να περιορίζεται όμως η ανάπτυξη εφαρμογών και λειτουργιών.

3.2. Κριτήρια κατηγοριοποίησης

Προκειμένου να συγκρίνουμε τα λειτουργικά συστήματα μεταξύ τους, είναι ιδιαίτερα χρήσιμο να ορίσουμε ορισμένα κριτήρια σύμφωνα με τα οποία θα μπορέσουμε να τα κατηγοριοποιήσουμε. Παρά το γεγονός ότι οι ανάγκες των τελικών χρηστών είναι το σημαντικότερο ίσως κριτήριο, ωστόσο λόγω της διαφοροποίησής τους από χρήστη σε χρήστη δίνεται έμφαση στη προσέγγιση ενός λειτουργικού συστήματος που θα καλύπτει τις ανάγκες κατηγοριών των χρηστών. Για τον λόγο αυτό τα παρακάτω κριτήρια κατηγοριοποίησης βασίζονται α) στις ιδιότητες και β) στις τεχνικές

λεπτομέρειες που χαρακτηρίζουν τα λειτουργικά συστήματα και θα χρησιμοποιηθούν στην συνέχεια για την σύγκριση αυτών.

Portability

Το χαρακτηριστικό αυτό αναφέρεται στην ικανότητα μεταφοράς από μια τοποθεσία σε μια άλλη. Σε σχέση με τα λειτουργικά συστήματα για κινητές συσκευές, η ικανότητα αυτή αναφέρεται στην δυνατότητα του λειτουργικού συστήματος να χρησιμοποιηθεί από οποιαδήποτε συσκευή.

Reliability

Η αξιοπιστία του λειτουργικού συστήματος σχετίζεται με την ικανότητα να διατηρεί την λειτουργικότητά του υπό κανονικές συνθήκες τουλάχιστον για κάποια συγκεκριμένη χρονική περίοδο.

Connectivity

Η συνδεσιμότητα έχει την έννοια της παροχής σύνδεσης για την μεταφορά σημάτων τα οποία μπορεί να είναι είτε πακέτα δεδομένων είτε σήματα φωνής. Τα λειτουργικά συστήματα των κινητών συσκευών είναι υπεύθυνα για την παροχή της συνδεσιμότητας είτε ενσύρματης είτε ασύρματης. Ειδικότερα δε όσον αφορά την ασύρματη συνδεσιμότητα το λειτουργικό σύστημα θα πρέπει να είναι σε θέση ώστε να διαχειρίζεται κατάλληλα το hardware των πομποδεκτών έτσι ώστε α) να είναι σε θέση να λειτουργούν παράλληλα και β) να μεταφέρονται οι ενεργές συνδέσεις ανάμεσα στους πομποδέκτες όταν αυτό είναι απαραίτητο (πχ. μεταφορά σύνδεσης από το GSM σε 3G).

Diversity

Το χαρακτηριστικό αυτό αναφέρεται στην διαφοροποίηση ενός προϊόντος έναντι άλλων προκειμένου να το κάνει μοναδικό στην αγορά. Στο πλαίσιο των λειτουργικών συστημάτων κινητών συσκευών αυτό μπορεί να επιτυγχάνεται με την ενσωμάτωση

είτε κάποιας επιπρόσθετης λειτουργίας είτε κάποιου υλικού. Χαρακτηριστικά παραδείγματα από το παρελθόν είναι η εισαγωγή παιχνιδιών στα κινητά τηλέφωνα και η ενσωμάτωση δέκτη ραδιοφώνου, αντίστοιχα. Σε κάθε περίπτωση πάντως η στρατηγική του κατασκευαστή της κινητής συσκευής είναι αυτή που θα καθορίσει τα χαρακτηριστικά ή/και υπηρεσίες που θα διαφοροποιήσουν το τελικό προϊόν έναντι άλλων ανταγωνιστικών.

Open

Εξ' ορισμού, ένα ανοικτό σύστημα είναι το σύνολο των συνιστωσών λογισμικού ή/και υλικού που αλληλεπιδρούν μεταξύ τους με προκαθορισμένες διαδικασίες και πρότυπα των οποίων τα χαρακτηριστικά και οι λεπτομέρειες υλοποίησής τους είναι διαθέσιμες τόσο προς τους κατασκευαστές όσο και προς τους προγραμματιστές, χωρίς κανένα κόστος. Έτσι, η ανάπτυξη ανοικτών λειτουργικών συστημάτων συμβάλλει στην δίχως περιορισμούς επέκτασή τους και φυσικά στην ενσωμάτωση οποιασδήποτε νέας τεχνολογίας ή/και εφαρμογής.

Kernel

Μια από τις βασικότερες οντότητες των λειτουργικών συστημάτων είναι ο πυρήνας ο οποίος είναι υπεύθυνος για την διαχείριση μνήμης, ενεργών διαδικασιών και του αποθηκευτικού χώρου. Το μέγεθος του Kernel είναι ένα από τα σημαντικότερα ζητήματα για τα λειτουργικά συστήματα των κινητών συσκευών καθώς φορτώνεται μόνιμα στην μνήμη της συσκευής κατά την εκκίνησή της, γεγονός που περιορίζει κατ' επέκταση την συνολική της χωρητικότητα.

Standards

Τα πρότυπα προδιαγράφουν αρχιτεκτονικές, διαδικασίες και δομές οι οποίες έχουν μελετηθεί και υιοθετηθεί και προτείνονται ανά περίπτωση από διεθνείς οργανισμούς. Τα πρότυπα τα οποία περιβάλλουν το πλαίσιο των λειτουργικών συστημάτων κινητών συσκευών, σχετίζονται με την γλώσσα προγραμματισμού τόσο του ίδιου του λειτουργικού συστήματος όσο και των εφαρμογών, με τα πρωτόκολλα σύνδεσης, δικτύωσης και μεταφοράς δεδομένων.

Security

Η ασφάλεια ενός συστήματος είναι η ικανότητα να προστατεύει τόσο το ίδιο το σύστημα καθώς και τις πληροφορίες που διαχειρίζεται από εξωτερικές επιθέσεις. Το λειτουργικό σύστημα πρέπει να παρέχει το μέγιστο δυνατό βαθμό ασφάλειας σε κάθε επίπεδο είτε για το ίδιο το λειτουργικό σύστημα, είτε για τις εφαρμογές που υποστηρίζει και την διαχείριση και μεταφορά των δεδομένων από και προς το λειτουργικό σύστημα.

Όλα τα παραπάνω κριτήρια μπορούν να καθορίσουν την βάση για την κατηγοριοποίηση των λειτουργικών συστημάτων κινητών συσκευών. Ωστόσο, θα ήταν παράληψη να μην αναφερθούν και άλλα κριτήρια τα οποία πολλοί χρήστες τα θεωρούν σημαντικά για τους ίδιους ή διαφορετικούς λόγους. Ορισμένα από τα βασικότερα από αυτά είναι τα παρακάτω:

Το λειτουργικό σύστημα είναι προσαρμοσμένο για την συσκευή του κατασκευαστή ή είναι ανεξάρτητο της συσκευής;

Καλύπτει περισσότερο τις ανάγκες του χρήστη ή του κατασκευαστή;

Το λειτουργικό σύστημα περιγράφεται επαρκώς προς τον χρήστη ως προς τις λειτουργίες του;

Μπορεί να το προσαρμόσει ο χρήστης βάσει των προτιμήσεών του ή πρέπει να επιλέξει ανάμεσα από εργοστασιακές επιλογές;

Ποια είναι η χωρητικότητα της μπαταρίας και πόσο διαρκεί;

Μπορεί να πραγματοποιήσει παράλληλες λειτουργίες όπως περιήγηση στο Διαδίκτυο ταυτόχρονη κλήση;

Ποιο πρέπει να είναι το επίπεδο του χρήστη για την χρήση του συστήματος;

Μπορούν να εγκατασταθούν εφαρμογές διαφορετικής ή μη τεχνολογίας;

3.3. Σύγκριση του Android με το Symbian OS και το Windows Mobile

Τα λειτουργικά συστήματα για κινητές συσκευές Android, Symbian OS και Windows Mobile θα συγκριθούν κυρίως στο πλαίσιο των κριτηρίων που περιγράφηκαν παραπάνω. Για κάθε ένα από τα κριτήρια αυτά τα λειτουργικά συστήματα θα ταξινομηθούν ανάλογα με τον βαθμό που ικανοποιούν τις απαιτήσεις και προδιαγραφές του κριτηρίου. Στο τέλος της ενότητας θα παρουσιαστούν συγκεντρωτικά τα αποτελέσματα που θα προκύψουν.

3.3.1. Ικανότητα μεταφοράς - Portability

Ξεκινώντας με το πρώτο κριτήριο, αναμφισβήτητα το Symbian OS έχει χρησιμοποιηθεί κατά κόρον τόσο σε απλές όσο και σε πιο εξελιγμένες συσκευές (πχ. Nokia). Ο λόγος για αυτό είναι η προτυποποιημένη αρχιτεκτονική του και η δυνατότητα υποστήριξης εφαρμογών από άλλους κατασκευαστές.

Όμοια και με το Windows Mobile, είναι δυνατό να χρησιμοποιηθεί και σε διαφορετικές συσκευές, ωστόσο διαθέτει πλήθος εφαρμογών που είναι συμβατές με συγκεκριμένες πλατφόρμες υλικού και για το λόγο αυτό το επίπεδο του portability για αυτό το λειτουργικό είναι χαμηλό.

Το Android όμως δεδομένου ότι βασίζεται σε Linux kernel όπως είδαμε και στην

Εικόνα 2.2, αποτελεί πολύ σημαντικό πλεονέκτημα καθώς μπορεί να χρησιμοποιηθεί σε διαφορετικές πλατφόρμες υλικού. Επιπλέον, η δυνατότητα που παρέχει για την επέκτασή του θα αποτελέσει σημαντικό παράγοντα για την εναρμόνισή του και με μελλοντικές πιο σύνθετες πλατφόρμες. Όσον αφορά την Java που είναι η γλώσσα προγραμματισμού που βασίζεται και χρησιμοποιείται για την ανάπτυξη των εφαρμογών, είναι αναμφισβήτητα ανεξάρτητη της πλατφόρμας του υλικού.

Το γεγονός ότι το Symbian OS χρησιμοποιείται κυρίως σε κινητά της κατασκευάστριας εταιρίας Nokia και δεν βασίζεται στην Java κατατάσσεται πίσω από το Android, ενώ το Windows Mobile κατατάσσεται τρίτο πίσω από το Symbian OS λόγω των περιορισμών που αναφέρθηκαν παραπάνω.

3.3.2. Αξιοπιστία - Reliability

Παρά το γεγονός ότι πραγματοποιείται σειρά ελέγχων και δοκιμών τόσο στις εφαρμογές όσο και στα λειτουργικά συστήματα, είναι αρκετά δύσκολο να αποκαλυφθούν όλες οι δυσλειτουργίες τους και ποια θα είναι η συμπεριφορά τους σε κάθε πιθανή περίπτωση. Έτσι, η αξιοπιστία τόσο του λειτουργικού συστήματος όσο γενικά και των εφαρμογών, εξαρτάται κυρίως από την εμπειρία των χρηστών σε αυτά. Ωστόσο επειδή αυτό δεν είναι πρακτικό, καθώς θα πρέπει να περάσει από μια αρκετά μεγάλη δοκιμαστική περίοδο προτού βγει στην αγορά, χρησιμοποιούνται δύο βασικοί παράγοντες για την εκτίμηση της μελλοντικής αξιοπιστίας του λειτουργικού

συστήματος που είναι α) το μέγεθος του λειτουργικού συστήματος και β) η δυνατότητα απομόνωσης των σφαλμάτων [7].

Ο μέσος όρος των γραμμών κώδικα που απαιτούνται για την συγγραφή ενός λειτουργικού συστήματος είναι περίπου ένα εκατομμύριο, ενώ οι δυσλειτουργίες που έχουν εντοπιστεί για ένα λειτουργικό σύστημα βασισμένο στον Linux kernel είναι περίπου 15.000 και για τον Windows kernel περίπου οι διπλάσιες. Επομένως, εκ των αποτελεσμάτων αυτών δεν μπορεί να υποστηριχτεί ότι τα λειτουργικά συστήματα είναι πλήρως αξιόπιστα ενώ αποκαλύπτεται επίσης ότι δεν είναι πλήρως κατανοητός ο τρόπος λειτουργίας τους.

Βάση εμπειρικών στοιχείων θα μπορούσε να βγει το συμπέρασμα ότι το επίπεδο αξιοπιστίας των Symbian OS και Windows Mobile λειτουργικών συστημάτων είναι αρκετά ικανοποιητικό για την πλειονότητα των χρηστών και εφαρμογών. Αυτό βέβαια δε σημαίνει ότι λειτουργούν άψογα αλλά ότι τα προβλήματα που ενδεχομένως να παρουσιάσουν δε θα προκαλέσουν σοβαρές δυσλειτουργίες ή άλλα σοβαρότερα σφάλματα.

Όσον αφορά το Android, δεδομένου ότι δεν έχει αρκετό καιρό στην αγορά είναι δύσκολο να εξαχθούν εμπειρικά συμπεράσματα σχετικά με την αξιοπιστία του. Ωστόσο, το γεγονός ότι το Android βασίζεται σε Linux kernel ο οποίος χρησιμοποιείται παραδοσιακά για Web Servers ή εφαρμογές που απαιτούν υψηλό βαθμό αξιοπιστίας καθώς και σχεδόν δύο δεκαετίες για κινητές συσκευές, είναι ένα θετικό στοιχείο ως προς την αξιοπιστία του ίδιου του λειτουργικού αλλά και των εφαρμογών που θα υποστηρίξει. Για τους λόγους αυτούς δεν υπάρχει κάποιος οριζόντιος όρισμα ότι το Android δεν θα παρέχει τουλάχιστον το ίδιο επίπεδο αξιοπιστίας με τα άλλα λειτουργικά αν όχι υψηλότερο.

3.3.3. Συνδεσιμότητα - Connectivity

Υπάρχουν πάρα πολλοί τρόποι με τους οποίους η κινητή συσκευή μπορεί να συνδεθεί είτε με ηλεκτρονικούς υπολογιστές είτε στο Διαδίκτυο ή ακόμα και με άλλες κινητές συσκευές, εκτυπωτές κλπ. Οι δυο βασικές κατηγορίες συνδεσιμότητας που θα πρέπει όμως να υποστηρίζονται από την συσκευή, ανεξάρτητα από τον τρόπο που αυτές θα χρησιμοποιούνται, είναι η ενσύρματη και ασύρματη σύνδεση.

Ειδικότερα, όσον αφορά την ασύρματη σύνδεση τα πράγματα είναι περισσότερο πολύπλοκα συγκριτικά με την ενσύρματη όπου απαιτείται μόνο ένα καλώδιο (πχ. USB) για την επίτευξη της σύνδεσης. Κάθε τρόπος ασύρματης σύνδεσης απαιτεί α) να είναι εγκατεστημένος στην συσκευή και ο αντίστοιχος πομποδέκτης, β) το λειτουργικό σύστημα να είναι σε θέση να τον διαχειριστεί σε επίπεδο υλικού, γ) να παρέχεται στις εφαρμογές ο τρόπος επικοινωνίας με τον πομποδέκτη και δ) να υποστηρίζονται τα κατάλληλα πρωτόκολλα επικοινωνίας καθώς και να υπάρχει δυνατότητα αναβάθμισης-επέκτασης αυτών.

Οι πιο κοινές τεχνολογίες ασύρματης σύνδεσης που χρησιμοποιούνται πλέον από τις κινητές συσκευές είναι (με χρονολογική σειρά εμφάνισής τους) το GSM, Infrared (υπέρυθρες) 0, Bluetooth, GPRS, UMTS (3G), WiFi, HSDPA 0. Όλες οι τεχνολογίες αυτές υποστηρίζονται από όλα τα υπό μελέτη λειτουργικά συστήματα και παρέχουν την δυνατότητα ανάπτυξης εφαρμογών που θα επικοινωνούν μέσω των τεχνολογιών αυτών.

3.3.4. Διαφοροποίηση - Diversity

Η διαφοροποίηση των κινητών συσκευών δεν έγκειται μόνο στο λειτουργικό σύστημα αλλά και στην ίδια την συσκευή ως τελικό προϊόν. Χαρακτηριστικό παράδειγμα αποτελεί η δυνατότητα αλλαγής πρόσοψης ορισμένων συσκευών στα τέλη της δεκαετίας του 90'. Ωστόσο, η ενσωμάτωση νέων δυνατοτήτων στην συσκευή που θα διαχειρίζονται από το λειτουργικό σύστημα έχει αποτελέσει τον στόχο όλων των εμπλεκόμενων για την δημιουργία μιας συσκευής που θα τους δώσει το ανταγωνιστικό πλεονέκτημα.

Όλοι οι κατασκευαστές των υπό μελέτη λειτουργικών συστημάτων, Symbian, Microsoft και Google, βρίσκονται σε συνεχή επαφή με τους κατασκευαστές κινητών συσκευών (Nokia, Samsung, LG κλπ) για την επίτευξη του ανταγωνιστικού πλεονεκτήματος που θα τους οδηγήσει σε αύξηση των πωλήσεών τους. Με τον τρόπο αυτό πέρα από την ταχεία ανάπτυξη νέων λειτουργιών και εφαρμογών αναπτύσσεται παράλληλα και ο ευρύτερος τομέας των κινητών επικοινωνιών καθώς όσο πιο ισχυρές γίνονται οι συσκευές τόσο μεγαλύτερες είναι και οι απαιτήσεις για γρηγορότερες συνδέσεις και νέες υπηρεσίες από την μεριά του δικτύου.

3.3.5. Ανοικτό σύστημα - Open

Για τη σύγκριση των λειτουργικών συστημάτων για να καταλήξουμε στο κατά πόσο ανοικτά είναι, βάση και των όσων περιγράφηκαν στην ενότητα 3.2, θα πρέπει να εξεταστεί αν πληρούν ορισμένες προϋποθέσεις, που ορίζουν το πλαίσιο των ανοικτών συστημάτων γενικότερα και είναι οι παρακάτω:

- Επιτρέπεται η δωρεάν πρόσβαση στον κώδικα του συστήματος έτσι ώστε να είναι εφικτή η ανάπτυξη επιπρόσθετων λειτουργιών από άλλους προγραμματιστές;
- Επιτρέπεται η επανασχεδίαση ορισμένων λειτουργιών και η αντικατάσταση αυτών ή ολόκληρου του συστήματος;
- Ευθυγραμμίζεται με διεθνή πρότυπα λειτουργίας έτσι ώστε να εγγυηθεί τόσο την ορθή λειτουργία αλλά και το επίπεδο ποιότητας αυτής;
- Επιτρέπεται η δωρεάν χρήση του συστήματος, η ανάπτυξη νέων εφαρμογών και η δημοσίευση αυτών;
- Χρησιμοποιείται γλώσσα προγραμματισμού με ανοικτό πρότυπο, όπως για παράδειγμα η Java;
- Μπορεί να εγκατασταθεί και να χρησιμοποιηθεί σε οποιαδήποτε πλατφόρμα υλικού;

Το μόνο λειτουργικό σύστημα το οποίο πληροί τις παραπάνω προϋποθέσεις είναι το Android και αυτό γιατί σχεδιάστηκε εξ' αρχής από την Google για τον σκοπό αυτό. Πιο συγκεκριμένα, κάθε λειτουργικό σύστημα παρέχει τρόπο ανάπτυξης επιπρόσθετων εφαρμογών για αυτό μέσω του κατάλληλου Software Development Kit (SDK), ωστόσο μόνο το Android βασίζεται στον ανοικτό Linux kernel ο οποίος μπορεί να τροποποιηθεί εάν αυτό κρίνεται αναγκαίο. Επιπλέον, η δωρεάν δημοσίευση νέων εφαρμογών είναι κάτι από το οποίο απέχουν ακόμα κατά πολύ η Symbian και η Microsoft που απαιτούν πληρωμή για κάτι τέτοιο.

3.3.6. Πυρήνας - Kernel

Όπως αναφέρθηκε και στην ενότητα 3.2, το μέγεθος της μνήμης που απαιτεί ο kernel του λειτουργικού παίζει σημαντικό ρόλο στην αξιοπιστία του καθώς όσο μικρότερες

είναι οι απαιτήσεις του σε μνήμη μειώνεται και η πολυπλοκότητά του με αποτέλεσμα να αυξάνονται οι επιδόσεις του. Για να εκτιμήσουμε το μέγεθος μνήμης που απαιτεί καθένα από τα υπό μελέτη λειτουργικά, λαμβάνουμε υπόψη τις ελάχιστες απαιτήσεις τους για την εγκατάστασή τους. Έτσι, το Symbian OS απαιτεί 200 Kbytes ελάχιστη μνήμη η οποία θα δεσμευτεί αποκλειστικά για τον kernel του. Για το Android απαιτούνται 250 Kbytes για τον Linux kernel ενώ το Windows Mobile που βασίζεται στην πλατφόρμα Windows CE 0 απαιτούνται 300 Kbytes.

3.3.7. Πρότυπα - Standards

Η υιοθέτηση προτύπων από τα λειτουργικά συστήματα παρέχουν μεγαλύτερες δυνατότητες προς τους προγραμματιστές για την δημιουργία νέων εφαρμογών καθώς το λειτουργικό σύστημα γίνεται περισσότερο ανοικτό (υποενότητα 3.3.5). Τα βασικά πρότυπα είναι τα παρακάτω:

- Χρήση προτύπου για την αναπαράσταση χαρακτήρων (πχ. Unicode).
- Προτυποποιημένη γλώσσα προγραμματισμού (πχ. Java).
- Προτυποποιημένα πρωτόκολλα δικτύωσης (πχ. TCP, IP).
- Προτυποποιημένα πρωτόκολλα ανταλλαγής e-mail (πχ. POP3, IMAP, SMTP).
- Χρήση προτύπου για την ανταλλαγή μηνυμάτων κειμένου και πολυμέσων, SMS και MMS αντίστοιχα.
- Χρήση προτύπων για επικοινωνία μεταξύ των συσκευών (Bluetooth, Infrared κλπ).
- Χρήση προτυποποιημένων τεχνολογιών για πρόσβαση στο Διαδίκτυο.
- Χρήση προτύπων για επικοινωνία συσκευών διαφορετικής πλατφόρμας (πχ. SyncML).

Τα περισσότερα από τα παραπάνω πρότυπα υποστηρίζονται και από τα τρία λειτουργικά συστήματα καθώς χρησιμοποιούν τα πιο κοινά σχετικά με την δικτύωση, ανταλλαγή e-mail, μηνυμάτων κλπ. Ωστόσο, το σημείο που διαφοροποιείται το Android έναντι των άλλων είναι για άλλη μια φορά η χρήση της γλώσσας προγραμματισμού Java. Το γεγονός αυτό είναι μεγάλης σημασίας καθώς οι εφαρμογές Java μπορούν να χρησιμοποιηθούν σε οποιαδήποτε πλατφόρμα χωρίς να απαιτείται να ξαναγραφτούν.

3.3.8. Ασφάλεια - Security

Το επίπεδο ασφαλείας των υπό μελέτη λειτουργικών συστημάτων είναι αρκετά ικανοποιητικό ενώ βασίζονται σε διαφορετικό τρόπο λειτουργίας το καθένα. Πιο συγκεκριμένα, το Symbian OS βασίζεται σε μια αρχιτεκτονική ασφαλείας η οποία είναι σε θέση να αντιμετωπίζει κακόβουλες ή κακο-προγραμματισμένες εφαρμογές 0. Η αρχιτεκτονική αυτή βασίζεται σε δύο βασικές οντότητες, την Certificate management και την Cryptography. Αυτές είναι υπεύθυνες για παρέχουν κατάλληλες λειτουργίες σε άλλες εφαρμογές ή οντότητες του λειτουργικού συστήματος σχετικά με την διαχείριση ψηφιακών πιστοποιητικών, εγκατάσταση νέων εφαρμογών, ασφαλείς συνδέσεις, κρυπτογράφηση δεδομένων κλπ.

Το Windows Mobile διαθέτει το δικό του μοντέλο ασφαλείας το οποίο συνδυάζει πολιτικές ασφαλείας, ρόλους και πιστοποιητικά για την έγκριση των αλλαγών στο σύστημα, την απομακρυσμένη σύνδεση και την εκτέλεση των εφαρμογών 0. Με τον τρόπο αυτό κάθε εφαρμογή εκτελείται στο πλαίσιο ασφαλείας που της επιτρέπει το λειτουργικό σύστημα όπως για παράδειγμα σχετικά με την σύνδεση στο δίκτυο, πρόσβαση σε μονάδες μνήμης και υλικό της συσκευής κ.α.

Τέλος, το Android όπως έχει ήδη προαναφερθεί έχει την ικανότητα παράλληλης εκτέλεσης εφαρμογών και λειτουργιών του ίδιου του λειτουργικού συστήματος σε δικές τους αποκλειστικά διαδικασίες 0. Έτσι, ο Linux kernel είναι σε θέση να ελέγχει τις διαδικασίες που εκτελούνται σε αυτόν, ανεξάρτητα από το αν πρόκειται για διαδικασίες εφαρμογών ή του ίδιου του λειτουργικού, σχετικά με τα δικαιώματα πρόσβασης κάθε διαδικασίας. Επιπλέον, το σύστημα διαθέτει εξειδικευμένη λειτουργία προκειμένου να περιορίζει τις επιτρεπόμενες ενέργειες κάθε διαδικασίας όπως για παράδειγμα σύνδεση δικτύου ή προσπέλαση μνήμης κ.α.

3.4. ΣΥΓΚΡΙΣΕΙΣ - ΔΙΑΠΙΣΤΩΣΕΙΣ

Όπως διαπιστώθηκε παραπάνω, και τα τρία λειτουργικά συστήματα έχουν την δική τους σχεδίαση που αντικατοπτρίζει τον δικό τους τρόπο λειτουργίας, όπως προέκυψε για κάθε ένα από τα παραπάνω κριτήρια τα οποία αποτέλεσαν την βάση για την μεταξύ τους σύγκριση. Προκειμένου να καταλήξουμε σε ένα ποιοτικό συμπέρασμα σχετικά με την καταλληλότητα του κάθε λειτουργικού συστήματος για καθένα από τα κριτήρια, μπορούμε να τα βαθμολογήσουμε ξεχωριστά βάση του συμπεράσματος που

προκύπτει για το καθένα ανά κριτήριο. Έτσι, σε περίπτωση που διαπιστώνουμε ότι κάποιο λειτουργικό σύστημα ευθυγραμμίζεται απόλυτα με τις απαιτήσεις του κριτηρίου θα δίδεται ο βαθμός 3. Όμοια, σε περίπτωση που δεν ικανοποιούνται πλήρως οι απαιτήσεις θα δίδεται ο βαθμός 2 και τέλος ο βαθμός 1 θα δίδεται σε περιπτώσεις όπου καλύπτονται μόνο οι βασικές απαιτήσεις.

Βάση των συμπερασμάτων που προκύπτουν ανά κριτήριο στην υπο-ενότητα 3.3, καταλήγουμε στις βαθμολογίες των λειτουργικών συστημάτων οι οποίες φαίνονται στον παρακάτω πίνακα.

Πίνακας 2: Συγκριτικά αποτελέσματα λειτουργικών συστημάτων Android, Symbian OS και Windows Mobile

Κριτήριο	Android	Symbian OS	Windows Mobile
Portability	3	2	1
Reliability	2	2	2
Connectivity	3	3	3
Diversity	3	3	3
Open	3	1	1
Kernel	2	3	1
Standards	3	2	2
Security	3	3	3
Total:	22	19	16

Το άθροισμα των βαθμολογιών ανά λειτουργικό σύστημα μπορεί να μας δώσει εύκολα ένα ποιοτικό συμπέρασμα σχετικά με το λειτουργικό σύστημα το οποίο ευθυγραμμίζεται περισσότερο έναντι των άλλων με τα κριτήρια τα οποία ορίστηκαν. Ωστόσο, θα ήταν λάθος να υποθέσουμε ότι το λειτουργικό σύστημα με την υψηλότερη βαθμολογία είναι κατάλληλο για κάθε τύπο χρήστη παρά το γεγονός ότι κάποια από αυτά υπερέχουν έναντι των άλλων σε ένα ή περισσότερα κριτήρια. Ας θυμηθούμε τους τύπους χρηστών που παρουσιάστηκαν στο

- Ο βασικός χρήστης, που χρησιμοποιεί μόνο τις απλές εφαρμογές της συσκευής.
- Ο μέσος χρήστης, που χρησιμοποιεί τόσο τις παραδοσιακές λειτουργίες όσο και διάφορες εφαρμογές.
- Ο προχωρημένος χρήστης, που έχει πλήρη επίγνωση του περιβάλλοντος της συσκευής και χρησιμοποιεί την πλειονότητα των προχωρημένων εφαρμογών.

Κάθε λειτουργικό σύστημα θα μπορούσε να χρησιμοποιηθεί από κάθε μία από τις παραπάνω κατηγορίες καθώς καμία εταιρία δεν θα δημιουργούσε ένα λειτουργικό σύστημα για κινητές συσκευές το οποίο θα μπορούσε να χρησιμοποιηθεί από μια μειοψηφία χρηστών. Κάθε επιτυχημένο προϊόν, έτσι και το λειτουργικό σύστημα, θα πρέπει να καλύπτει τις ανάγκες των χρηστών κάθε κατηγορίας. Ωστόσο, ορισμένα από τα λειτουργικά συστήματα που μελετάμε ενδείκνυται περισσότερο από τα άλλα για συγκεκριμένες κατηγορίες χρηστών για διάφορους λόγους. Παρακάτω προτείνεται το λειτουργικό σύστημα ανά κατηγορία χρηστών βάση των κριτηρίων.

Ο βασικός χρήστης χρειάζεται ένα λειτουργικό σύστημα για την κινητή του συσκευή με το οποίο να ικανοποιεί τις βασικές του επικοινωνιακές ανάγκες και ταυτόχρονα να του είναι οικείο ώστε να μπορεί να το χρησιμοποιεί εύκολα. Έτσι εύκολα μπορούμε να συμπεράνουμε ότι αυτός ο τύπος χρήστη δεν θα χρησιμοποιήσει λειτουργίες αναδιάρθρωσης του ίδιου του συστήματος ούτε θα εκμεταλλευτεί τις δυνατότητες ανάπτυξης και εγκατάστασης νέων εφαρμογών καθώς και επέκτασης και αναβάθμισης του λειτουργικού συστήματος.

Βασικός παράγοντας για αυτόν τον τύπο χρήστη είναι η ύπαρξη ενός καλά διατυπωμένου εγχειριδίου χρήσης και εύκολη χρήση των βασικών λειτουργιών όπως κλήσεις, αποστολή και λήψη μηνυμάτων SMS και ίσως λειτουργίας της κάμερας της συσκευής για την λήψη φωτογραφιών και βίντεο. Όπως είναι φυσικό, όλα αυτά υποστηρίζονται πλήρως από τα λειτουργικά συστήματα που παρουσιάστηκαν, ωστόσο εάν θα έπρεπε να διαλέξουμε ένα από αυτά ως το πιο ενδεδειγμένο για αυτό τον τύπο χρήστη ίσως η καλύτερη επιλογή θα ήταν το Windows Mobile. Ο κύριος λόγος για αυτό είναι το γεγονός πως το λειτουργικό σύστημα της ίδιας κατασκευάστριας εταιρίας (Microsoft) για προσωπικούς υπολογιστές (Windows XP) είναι το πιο διαδεδομένο σήμερα.

Η εμφάνιση καθώς και η δομή με την οποία παρουσιάζονται τα λειτουργικά συστήματα της Microsoft στον χρήστη είναι παρόμοια. Συνεπώς, οι βασικοί χρήστες θα εμπιστευτούν ευκολότερα ένα τέτοιο λειτουργικό σύστημα θα το νιώθουν οικείο και εύχρηστο χωρίς να απαιτείται χρόνος για να προσαρμοστούν σε αυτό.

Ο μέσος χρήστης πέρα από τις βασικές λειτουργίες έχει ανάγκη και από μερικές περισσότερο εξειζητημένες όπως εφαρμογές ανταλλαγής δεδομένων, συγχρονισμός με άλλες εφαρμογές είτε εσωτερικές στο ίδιο το λειτουργικό είτε εξωτερικές σε άλλες συσκευές, εφαρμογές ανταλλαγής e-mail, κοινωνικής δικτύωσης καθώς και ψυχαγωγικές εφαρμογές. Επίσης, σημαντικές είναι και οι λειτουργίες αλλαγής της διεπαφής του χρήστη τόσο με τις εφαρμογές όσο και με το ίδιο το λειτουργικό σύστημα. Επιπλέον, ο μέσος χρήστης έχει ανάγκη το λειτουργικό σύστημα της συσκευής να υποστηρίζει την παράλληλη εκτέλεση λειτουργιών καθώς για παράδειγμα μπορεί ενώ ακούει μουσική να πλοηγείται και στο Διαδίκτυο. Για αυτόν τον τύπο χρήστη παρά το γεγονός ότι το Windows Mobile θα του δίνει την αίσθηση οικειότητας, όμοια με του βασικού χρήστη, θα μπορούσε πέρα από αυτό να χρησιμοποιήσει και το Symbian OS ή το Android.

Όπως παρουσιάστηκε σε προηγούμενες ενότητες, όλα τα λειτουργικά συστήματα διαθέτουν αυτές τις ικανότητες και επιπλέον αυτός ο τύπος χρήστη είναι διατεθειμένος να προσαρμοστεί στον τρόπο λειτουργίας του συστήματος και να ανακαλύψει την πλειονότητα των δυνατοτήτων του.

Η τελευταία κατηγορία είναι αυτή του προχωρημένου χρήστη ο οποίος είναι διατεθειμένος όχι μόνο να ανακαλύψει κάθε πιθανή λειτουργία του συστήματος αλλά και να μάθει τον τρόπο με τον οποίο μπορεί να το αλλάξει, να ανακαλύψει αδυναμίες και να τις διορθώσει. Επιπλέον, κοινό χαρακτηριστικό των χρηστών της κατηγορίας αυτής είναι πως ο τρόπος χρήσης προς την συσκευή τους μοιάζει αρκετά με τον τρόπο που χειρίζονται και τον προσωπικό τους υπολογιστή, και μάλιστα πολλές φορές αναμένουν και την ίδια συμπεριφορά.

Σε περίπτωση που ο τρόπος λειτουργίας δεν τους ικανοποιεί, ψάχνουν τρόπους ώστε να τον αλλάξουν είτε σε κάτι περισσότερο οικείο πολλές φορές είτε σε κάτι καινοτόμο. Σε κάθε περίπτωση, το εγχειρίδιο χρήσης για αυτόν τον τύπο χρήστη δεν θα φανεί χρήσιμο καθώς μέσα σε μικρό χρονικό διάστημα ενδέχεται τα όσα αναφέρει για την λειτουργία του να μην ισχύουν σε μικρό ή μεγάλο βαθμό. Ωστόσο, τα τεχνικά

χαρακτηριστικά τόσο του λειτουργικού συστήματος όσο και της συσκευής παίζουν σημαντικό ρόλο καθώς μπορούν να επηρεάσουν την ανάπτυξη και χρήση νέων εφαρμογών. Το Android είναι το μόνο λειτουργικό σύστημα που μπορεί να ανταπεξέλθει στις ανάγκες αυτού του τύπου χρήστη καθώς α) είναι ανοικτό και του παρέχεται πρόσβαση σε οποιοδήποτε σημείο εκείνος επιθυμεί και β) μπορεί να προγραμματιστεί εύκολα με την χρήση της προτυποποιημένης γλώσσας Java.

Για άλλη μια φορά είναι σημαντικό να αναφερθεί πως όλα τα λειτουργικά συστήματα μπορούν να χρησιμοποιηθούν από κάθε κατηγορία χρήστη καθώς όλα ευθυγραμμίζονται με τις απαιτήσεις των σύγχρονων λειτουργικών συστημάτων. Το Android διαθέτει πλήθος νέων χαρακτηριστικών που το κάνουν πιο αρεστό σε περισσότερο εξειδικευμένους χρήστες και θα μπορέσει να ανταπεξέλθει όχι μόνο στις τρέχουσες συνθήκες αλλά και σε μελλοντικές εξελίξεις γενικότερα στον ευρύτερο τομέα των κινητών επικοινωνιών.

Συμπερασματικά, βάσει των όσων αναφέρθηκαν παραπάνω μπορούν να εξαχθούν τα παρακάτω:

- Κανένα λειτουργικό σύστημα δεν προκύπτει πως είναι ιδανικό, ωστόσο διαφέρουν ως προς την ευχρηστία τους σε σχέση με τις ανάγκες των χρηστών. Συνεπώς, οι χρήστες θα πρέπει να έχουν καλά καθορισμένες τις απαιτήσεις τους και κατόπιν να επιλέξουν την συσκευή με το λειτουργικό σύστημα που τις καλύπτει στον καλύτερο δυνατό βαθμό.
- Η εξάπλωση του Android έχει ήδη απειλήσει τα πρώην επικρατέστερα λειτουργικά συστήματα καθώς μπορεί πλέον να καλύπτει σε μεγαλύτερο βαθμό τις ανάγκες τόσο των μέσων όσο και των εξειδικευμένων χρηστών. Για τον λόγο αυτό τα λειτουργικά συστήματα Symbian OS και Microsoft Windows έχουν χάσει ένα σημαντικό μερίδιο αγοράς.

Όπως διαπιστώθηκε, οι ανάγκες για κάθε τύπο χρήστη είναι διαφορετικές επομένως και οι αντίστοιχες απαιτήσεις που θα πρέπει να καλύπτει το λειτουργικό σύστημα είναι αρκετά σύνθετες. Οι κατασκευαστές των κινητών συσκευών μπορούν να προσεγγίσουν πληρέστερα κάθε τύπο χρήστη εφόσον έχουν την δυνατότητα να προσαρμόσουν ανάλογα το λειτουργικό σύστημα Android χωρίς κόστος.

ΚΕΦΑΛΑΙΟ 4: ΠΕΡΙΒΑΛΛΟΝ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ

Στο κεφάλαιο αυτό θα περιγραφούν ορισμένες λεπτομέρειες σχετικά με το περιβάλλον ανάπτυξης που παρέχει η Google και στην συνέχεια η διαδικασία για τη δημιουργία μιας εικονικής συσκευής Android (Android Virtual Device). Επιπλέον θα δούμε τον τρόπο που δημιουργούμε ένα νέο project.

4.1 Λήψη και εγκατάσταση

Στην σελίδα <http://developer.android.com/sdk/index.html> υπάρχει διαθέσιμο το SDK, που παρέχει η Google για την ανάπτυξη εφαρμογών για το Android, τόσο για το λειτουργικά συστήματα Windows όσο και για Mac OS X και Linux. Επιπλέον, όπως φαίνεται και στην εικόνα παρακάτω, παρέχονται πληροφορίες για τα επόμενα βήματα που είναι απαραίτητα για την εγκατάσταση και χρήση του SDK.



The screenshot shows the 'Download the Android SDK' page. It includes a table with columns for Platform, Package, Size, and MD5 Checksum. The table lists packages for Windows, Mac OS X (intel), and Linux (386). Below the table, there are instructions on how to set up the SDK, including preparing the development computer, installing the SDK starter package, installing the ADT Plugin for Eclipse, and adding Android platforms.

Platform	Package	Size	MD5 Checksum
Windows	android-sdk_08-windows.zip	32696391 bytes	3e0b09ade5bfa9624bce9ddc164a48cb
	installer_08-windows.exe (Recommended)	32746192 bytes	04ce07b10a9361a1f63c2238bbc1ee3
Mac OS X (intel)	android-sdk_08-mac_08.zip	26797617 bytes	d2e392c4e4680cb2df65b6f2b662c7
Linux (386)	android-sdk_08-linux_08.tar.gz	26817291 bytes	3b626645b223d137d27beefbda0c94bc

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from the table above. (If you're on Windows, download the installer for help with the initial setup.)
3. Install the ADT Plugin for Eclipse (if you'll be developing in Eclipse).
4. Add Android platforms and other components to your SDK.
5. Explore the contents of the Android SDK (optional).

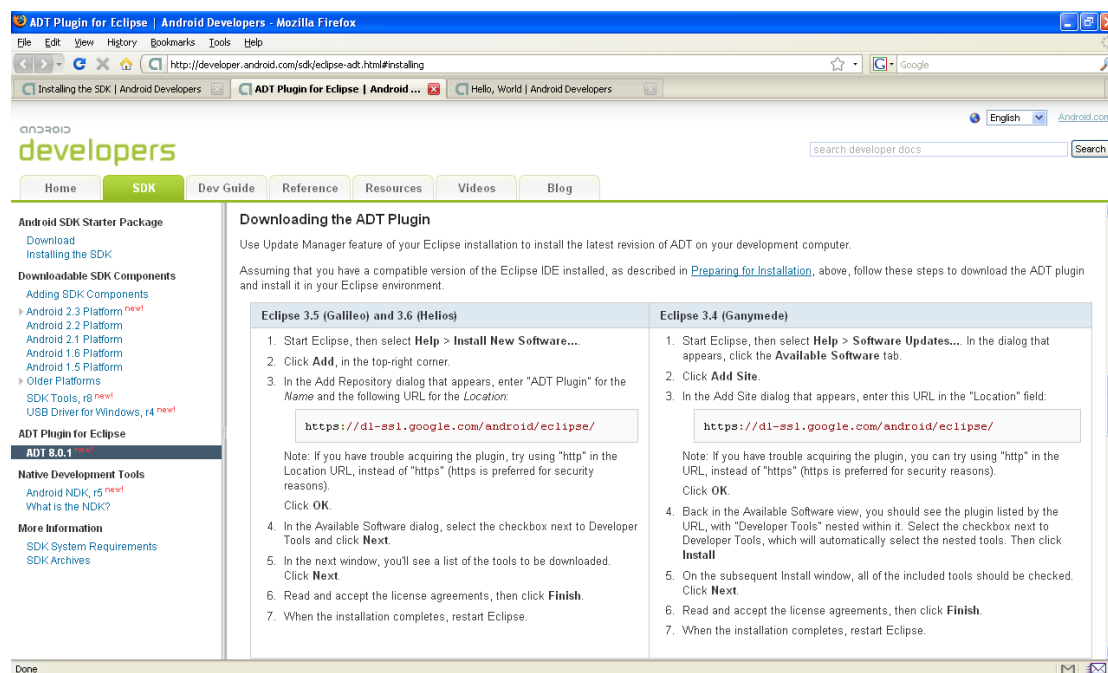
Εικόνα 4.1: Πληροφορίες λήψης, εγκατάστασης και χρήσης του Android SDK

Τα βήματα τα οποία απαιτούνται παρουσιάζονται συνοπτικά παρακάτω [14]:

Προετοιμασία εγκατάστασης: Πριν την εγκατάσταση ο προγραμματιστής πρέπει να ελέγξει εάν έχει ήδη εγκατεστημένο το περιβάλλον ανάπτυξης εφαρμογών Java, JDK (Java Development Kit). Κατόπιν, προτείνεται η χρήση της εφαρμογής Eclipse 0 για την ανάπτυξη του κώδικα της εφαρμογής Android.

Λήψη του Android SDK: Ο προγραμματιστής μπορεί να εντοπίσει και να κατεβάσει το κατάλληλο SDK για το λειτουργικό σύστημα που επιθυμεί.

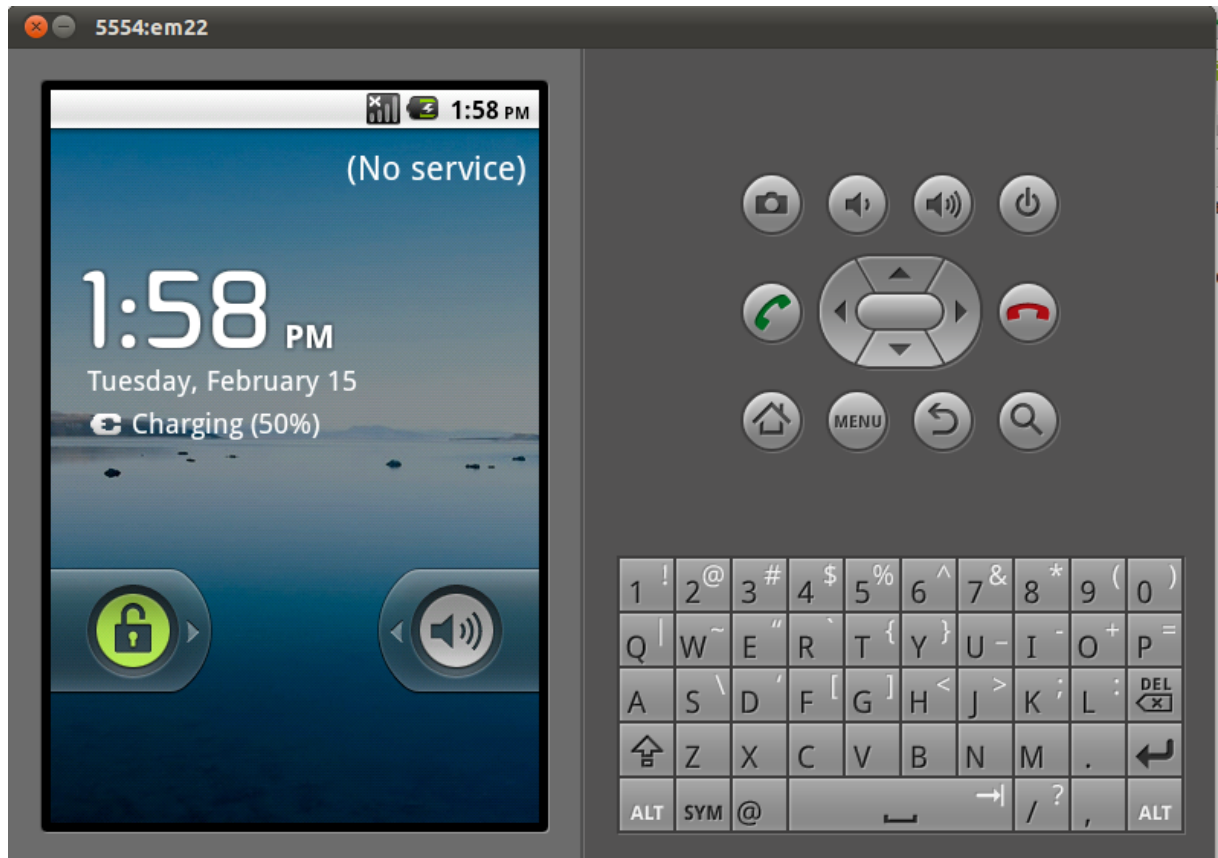
Εγκατάσταση του ADT Plugin για το Eclipse: Όπως φαίνεται και στην Εικόνα , το Android παρέχει ένα plugin για το Eclipse το οποίο ονομάζεται Android Development Tools (ADT) 0. Το plugin αυτό έχει σκοπό να επεκτείνει τις δυνατότητες του Eclipse έτσι ώστε ο προγραμματιστής να μπορεί να δημιουργήσει Android Projects, να κάνει Debug στον κώδικα, να δημιουργήσει γρήγορα το γραφικό περιβάλλον της εφαρμογής, να εξάγει την τελική εφαρμογή ώστε να την μεταφέρει στην κινητή συσκευή κ.α.



Εικόνα 4.2: Οδηγίες λήψης και εγκατάστασης του ADT Eclipse plugin

4.2 δημιουργία εικονικής συσκευής Android

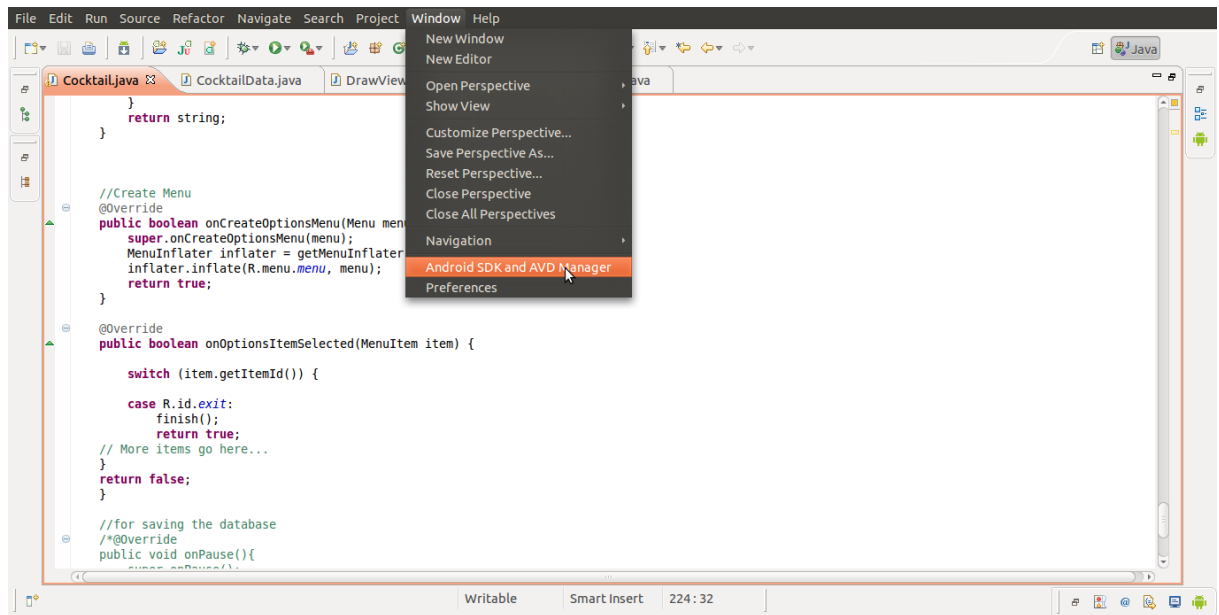
Μετά την λήψη και την εγκατάσταση των παραπάνω είμαστε έτοιμοι να δημιουργήσουμε την εικονική συσκευή που φαίνεται στην Εικόνα 4.3 και να εξηγήσουμε τις ρυθμίσεις που κάναμε.



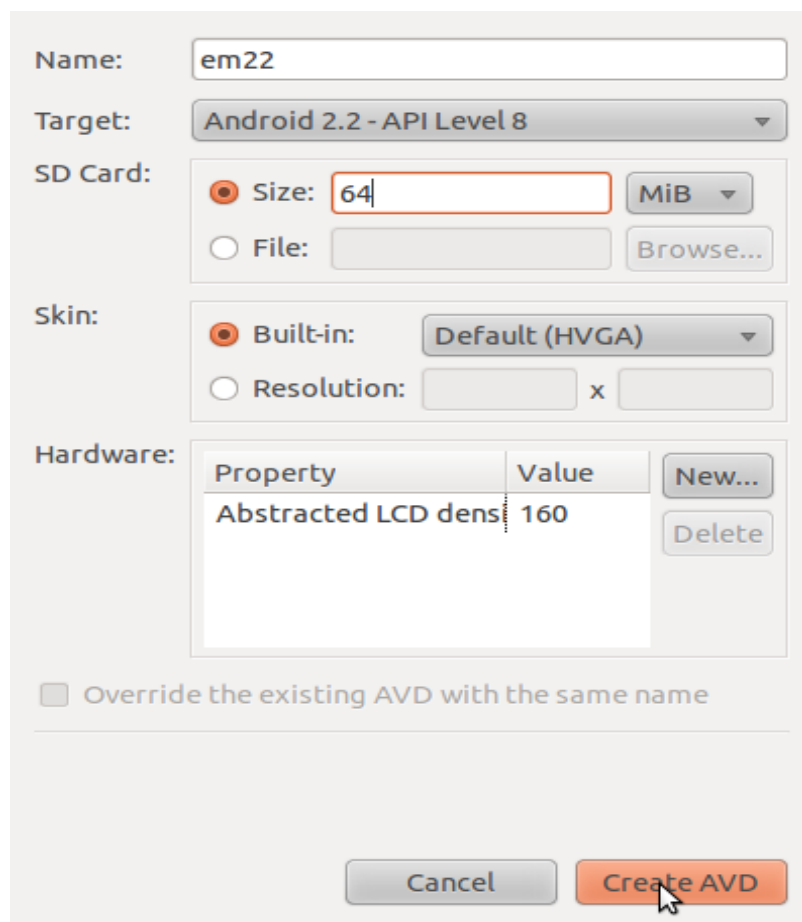
Εικόνα 4.3: Η εικονική συσκευή που χρησιμοποιήσαμε

Τα βήματα για να τη δημιουργήσουμε περιγράφονται παρακάτω:

- Ανοίγουμε το Eclipse, πηγαίνουμε στο μενού *Window* και επιλέγουμε *Android SDK and AVD Manager*. Εικόνα 4.4
- Στο παράθυρο που μας ανοίγεται πατάμε στο κουμπί *New...* εισάγουμε τα στοιχεία που βλέπουμε στην Εικόνα 4.5 και πατάμε το κουμπί *Create AVD*



Εικόνα 4.4: Επιλογή του Android SDK and AVD Manager



Εικόνα 4.5: Δημιουργία εικονικής συσκευής

Επεξήγηση των ρυθμίσεων

- Στο πεδίο *Name* δίνουμε το όνομα που θέλουμε να έχει η εικονική συσκευή, στην προκειμένη περίπτωση em22
- Στο πεδίο *Target* επιλέγουμε την έκδοση του λειτουργικού συστήματος Android που θέλουμε η συσκευή να έχει. Επιλέχτηκε η έκδοση 2.2
- Στο πεδίο *SD Card* βάζουμε τη χωρητικότητα που θέλουμε να έχει η εξωτερική κάρτα μας και είναι προαιρετικό πεδίο. Εισάγαμε 64 MB
- Στο πεδίο *Skin* επιλέγουμε το μέγεθος της οθόνης μας, είτε από τις ήδη υπάρχουσες επιλογές είτε καθορίζουμε μόνοι μας τις διαστάσεις της στο πεδίο *resolution*. Επιλέχθηκε το πεδίο *Built-in* με τιμή HVGA
- Τέλος, στο πεδίο *Hardware* μπορούμε να καθορίσουμε διάφορα χαρακτηριστικά που θέλουμε να έχει η συσκευή μας. Αφήσαμε την προκαθορισμένη ιδιότητα για την πυκνότητα – *density* με την τιμή 160. Περισσότερα για το θέμα της πυκνότητας και της ανάλυσης θα δούμε σε παρακάτω κεφάλαια

4.3 δημιουργία νέου *project*

Για να δοκιμάσουμε την εικονική συσκευή μας, δημιουργούμε ένα νέο Android project ακολουθώντας την παρακάτω διαδικασία:

- Επιλέγουμε από το μενού *File*→*New*→*Project*..
- Στο παράθυρο που μας ανοίγεται επιλέγουμε *Android*→*Android Project* και πατάμε *Next*
- Συμπληρώνουμε το όνομα του καινούργιου μας Project, Εικόνα 4.6
 - Επιλέγουμε την πλατφόρμα για την οποία θα σχεδιάσουμε την εφαρμογή μας
 - Εισάγουμε το όνομα της εφαρμογής μας

- Εισάγουμε το πακέτο στο οποίο θα ανήκει η εφαρμογή μας
- Επιλέγουμε να δημιουργηθεί νέα δραστηριότητα (Activity) με όνομα ίδιο με το όνομα της εφαρμογής μας. Περισσότερα για τις δραστηριότητες θα εξετάσουμε σε επόμενο κεφάλαιο.
- Τέλος εισάγουμε τον αριθμό του API που αντιστοιχεί στην πλατφόρμα που επιλέξαμε και πατάμε το Finish.

New Android Project

Creates a new Android Project resource.



Location:

Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	API Lev
<input type="checkbox"/> Android 1.5	Android Open Source Project	1.5	3
<input type="checkbox"/> Android 1.6	Android Open Source Project	1.6	4
<input type="checkbox"/> Android 2.1-update1	Android Open Source Project	2.1-update1	7
<input checked="" type="checkbox"/> Android 2.2	Android Open Source Project	2.2	8
<input type="checkbox"/> Android 2.3	Android Open Source Project	2.3	9

Standard Android platform 2.1-update1

Properties

Application name:

Package name:

Create Activity:

Min SDK Version:

Εικόνα 4.6: Δημιουργία νέου Project

Για να εκτελέσουμε το project και να δοκιμάσουμε ότι η εικονική συσκευή λειτουργεί, κάνουμε δεξί κλικ πάνω στο project και επιλέγουμε Run As→Android Application. Η εικονική συσκευή ξεκινάει και όταν ολοκληρωθεί η φόρτωση της εγκαθιστά την εφαρμογή μας και την εκτελεί. Η Εικόνα 4.7 δείχνει το τελικό αποτέλεσμα.



Εικόνα 4.7: Τελικό αποτέλεσμα

Μετά από την εκτέλεση των παραπάνω βημάτων μπορεί να ξεκινήσει η ανάπτυξη του κώδικα της εφαρμογής [17].

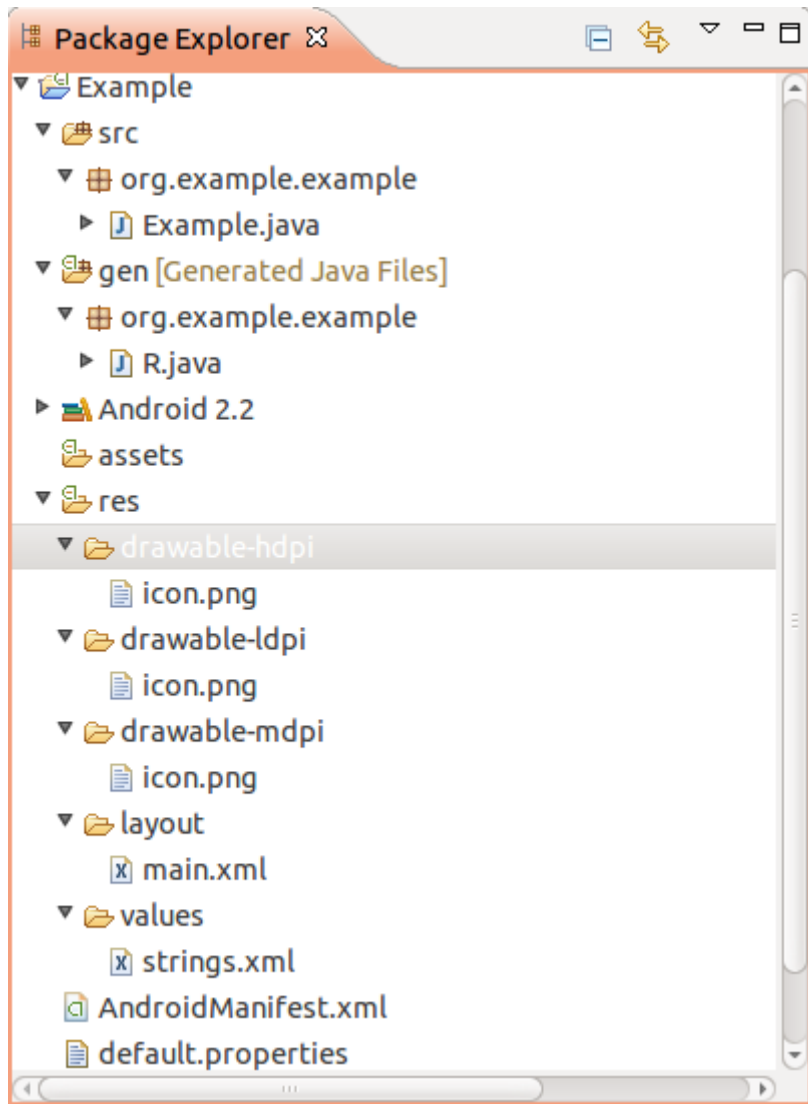
ΚΕΦΑΛΑΙΟ 5: ΤΟ ANDROID ΩΣ ΠΛΑΤΦΟΡΜΑ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ

Στο κεφάλαιο 2 περιγράψαμε τη δομή του λειτουργικού συστήματος Android και αναλύσαμε τις συνιστώσες που το αποτελούν. Στο κεφάλαιο αυτό θα εξετάσουμε το λειτουργικό σύστημα Android ως μία πλατφόρμα ανάπτυξης εφαρμογών.

Για την πληρέστερη κατανόηση του θα προβούμε στην ανάλυση της πρότυπης δομής που μας παρέχει το Android SDK της Google για την δημιουργία μιας εφαρμογής και συγκεκριμένα θα αναλύσουμε τους καταλόγους, κυρίως αυτούς που χρησιμοποιήσαμε στην εφαρμογή που δημιουργήσαμε, και βρίσκονται κάτω από το project μας. Θα περιγράψουμε μερικά από τα δομικά στοιχεία που καθορίζονται από το SDK, θα εμβαθύνουμε στον τρόπο λειτουργίας του Activity (Δραστηριότητα) και στον σημαντικό ρόλο που έχει ο κύκλος ζωής του. Τέλος θα μελετήσουμε τους τρόπους δημιουργίας του user interface (διεπαφή χρήστη).

Κεφάλαιο 5.1: Ανάλυση των καταλόγων σε ένα Android project

Σε αυτήν την ενότητα θα δούμε τους καταλόγους που δημιουργούνται όταν ξεκινάμε ένα νέο Android project. Θα εξετάσουμε τα περιεχόμενα τους αλλά και τι μπορούμε να προσθέσουμε σε αυτούς και θα αναλύσουμε το xml αρχείο AndroidManifest. Στην Εικόνα 5.1 βλέπουμε τους καταλόγους που δημιουργήθηκαν για το project με το όνομα Example μέσα από το περιβάλλον ανάπτυξης.



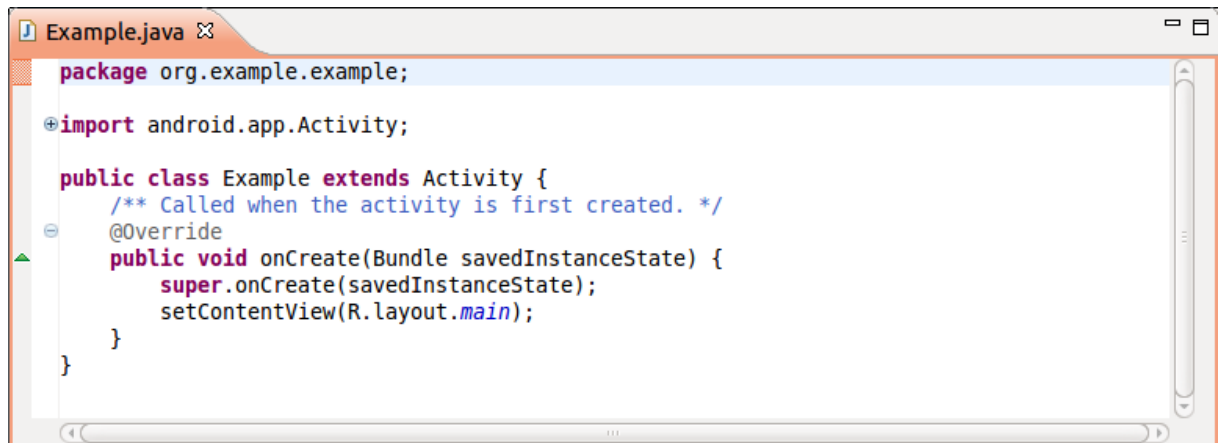
Εικόνα 5.1: Οι κατάλογοι που δημιουργήθηκαν για το project Example

5.1.1 Ο καταλόγος src

Το όνομα αυτού του καταλόγου έχει προέρθει από την συντομογραφία της Αγγλικής λέξης source που σημαίνει «πηγή». Σε αυτόν βρίσκεται όλος ο πηγαίος κώδικας μας, είτε είναι ένα αρχείο δηλαδή μια κλάση είτε είναι πολλές, πάντα όμως κάτω από ένα πακέτο.

Όπως βλέπουμε από την παραπάνω εικόνα, κάτω από τον κατάλογο /src υπάρχει το πακέτο που έχουμε δηλώσει ότι ανήκουν οι κλάσεις μας και το όνομα αυτού είναι συνήθως η προσωπική μας ιστοσελίδα. Επίσης στην ίδια εικόνα βλέπουμε και το αρχείο Example.java που δημιουργήθηκε αυτόματα από το SDK του Android και

κληρονομεί από την κλάση Activity και θα αναλύσουμε περισσότερα για αυτό στο Κεφάλαιο 5.2. Στην Εικόνα 5.2 βλέπουμε τι περιέχει η κλάση Example.java και θα αναλυθεί στο Κεφάλαιο 5.2.



```
package org.example.example;

import android.app.Activity;

public class Example extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Εικόνα 5.2: Τα περιεχόμενα του αρχείου Example.java

5.1.2 ο κατάλογος res

Ο κατάλογος αυτός περιέχει τους πόρους (resources) και το όνομα του είναι μια συντομογραφία της Αγγλικής λέξης resources. Όπως αναφέραμε και στο κεφάλαιο 2.1 στην εξήγηση του Resource Manager οι πόροι μπορούν να είναι κείμενο, εικόνες, ήχος ή ποιο απλά ότι δεν είναι κώδικας.

Σε αυτόν τον κατάλογο δημιουργούμε και αποθηκεύουμε τους πόρους ανάλογα με τον τύπο τους. Για παράδειγμα, αν έχουμε μια εικόνα τύπου jpeg ή png τότε πρέπει να πάει στον αντίστοιχο κατάλογο που ξεκινάει έτσι /res/drawable και μπορεί να έχει κατάληξη -hdpi, -mdpi, -ldpi. Αν έχουμε ένα xml αρχείο το οποίο καθορίζει την εμφάνιση που θα έχει η εφαρμογή μας τότε πρέπει να πάει στον κατάλογο /res/layout. Τέλος πέρα από τα υπάρχοντα επιθήματα μπορούμε να προσθέσουμε και δικά μας ανάλογα με τον τύπο των πόρων και τις ανάγκες που έχουμε. Περισσότερα για αυτό το θέμα θα δούμε στο επόμενο Κεφάλαιο 6 (υποστήριξη πολλαπλών οθονών)

Όλοι οι πόροι μας μεταγλωττίζονται από τον resource compiler (μεταγλωττιστή πόρων), ο οποίος τους συμπιέζει και τους πακετάρει δημιουργώντας μια κλάση με το όνομα R η οποία περιέχεται στον παρακάτω κατάλογο με το όνομα GEN. Η κλάση R περιέχει τα αναγνωριστικά που χρησιμοποιούμε έτσι ώστε να μας επιτρέπει να αναφερόμαστε σε αυτούς τους πόρους μέσα από το πρόγραμμα μας.

5.1.3 ο κατάλογος gen

Στο κατάλογο αυτό θα βρούμε την κλάση R που αναφέραμε προηγουμένως στο 5.1.2 και όπως συμβαίνει στο κατάλογο /src έτσι και εδώ η κλάση αυτή βρίσκεται κάτω από το πακέτο που ορίσαμε.

Κάτι πολύ σημαντικό που πρέπει να σημειώσουμε εδώ είναι ότι η κλάση R διαχειρίζεται αυτόματα από το Android plug-in του Eclipse. Κάθε φορά που βάζουμε ένα αρχείο οπουδήποτε στον κατάλογο /res το plug-in παρατηρεί την αλλαγή και προσθέτει τα κατάλληλα αναγνωριστικά IDs των πόρων στο αρχείο R.java για εμάς. Το R.java μένει πάντα συγχρονισμένο. Αν το ανοίξουμε θα δούμε ότι περιέχει κάτι σαν το παρακάτω:

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found. It
 * should not be modified by hand.
 */

package org.example.example;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Οι παραπάνω δεκαεξαδικοί αριθμοί είναι ακέραιοι αριθμοί οπού ο Resource Manager του Android τους χρησιμοποιεί για να φορτώσει τα πραγματικά δεδομένα όπως αλφαριθμητικά και αλλά στοιχεία τα οποία συγκεντρώνονται μέσα στο πακέτο μας. Οι τιμές που βλέπουμε ότι ισούνται δεν αντιστοιχούν στις πραγματικές τιμές των δεδομένων μας αλλά σε τιμές που αφορούν τους ίδιους τους πόρους μας. Πρέπει να σημειώσουμε πως σχεδόν κάθε πρόγραμμα Android, συμπεριλαμβανομένου και του ίδιου του Framework του Android, έχει μία κλάση R [18].

5.1.4 to xml αρχείο androidmanifest

Κάθε εφαρμογή πρέπει να έχει ένα αρχείο AndroidManifest.xml [19] με ακριβώς αυτό το όνομα στον αρχικό (root) κατάλογο του. Στο αρχείο παρουσιάζονται οι απαραίτητες πληροφορίες σχετικά με την εφαρμογή μας στο σύστημα του Android, πληροφορίες τις οποίες χρειάζεται το σύστημα πριν μπορέσει να εκτελέσει οποιοδήποτε κώδικα της εφαρμογής. Μεταξύ άλλων το AndroidManifest κάνει και τα ακόλουθα:

- Ονομάζει το Java πακέτο της εφαρμογής. Το όνομα του πακέτου λειτουργεί ως μοναδικό αναγνωριστικό ID για την εφαρμογή
- Περιγράφει τις συνιστώσες της εφαρμογής. Για παράδειγμα, τις activities , services (υπηρεσίες), content providers (παρόχους περιεχομένου) κτλ
- Ονομάζει τις κλάσεις που εφαρμόζουν καθεμία από τις συνιστώσες και δημοσιεύει τις ικανότητές τους. Για παράδειγμα, ποιες Intents (Προθέσεις) μπορούν να χειριστούν
- Οι δηλώσεις αυτές αφήνουν το σύστημα Android να γνωρίζει ποιες είναι οι συνιστώσες και υπό ποιες συνθήκες μπορούν να ενεργοποιούνται
- Δηλώνει ποια δικαιώματα η εφαρμογή πρέπει να έχει προκειμένου να γίνει δυνατή η πρόσβαση σε προστατευμένες περιοχές του API και να μπορεί να αλληλεπιδρά με άλλες εφαρμογές.
- Δηλώνει τα δικαιώματα που οι άλλοι οφείλουν να έχουν προκειμένου να αλληλεπιδράσουν με στοιχεία της εφαρμογής
- Δηλώνει το ελάχιστο επίπεδο του Android API που απαιτεί η εφαρμογή

Πολλά από τα παραπάνω μπορούμε να τα δούμε σε ένα τυπικό αρχείο AndroidManifest.xml που αυτόματα το SDK δημιούργησε μόλις φτιάξαμε ένα νέο Android project, και έχει ως εξής:

```
<?xml version="1.0" encoding="utf-8"?>
```

```

<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.example.example"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".Example"
            android:label="@string/app_name">
            <intent-filter>
                <action
android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="8" />

</manifest>

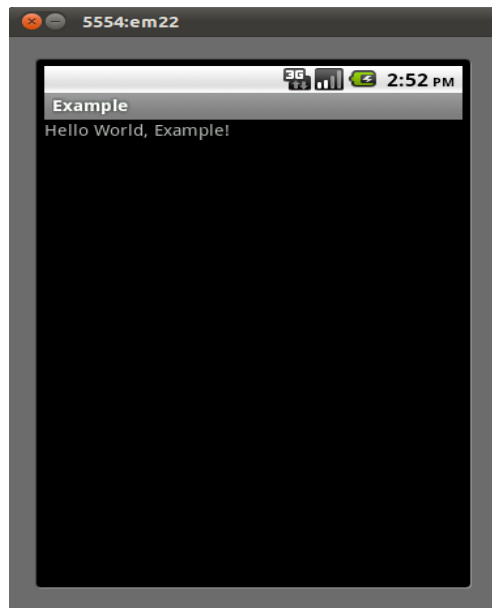
```

Από τον παραπάνω xml κώδικα θα αναλύσουμε τις ετικέτες που χρησιμοποιούνται και είναι αυτές που απαιτούνται να υπάρχουν ώστε να λειτουργήσει η εφαρμογή μας σε μια Android συσκευή. Επίσης πρέπει να αναφέρουμε ότι σχεδόν όλα τα γνωρίσματα πρέπει να φέρουν μπροστά τους την λέξη android. Τέλος, ενδεικτικά θα αναφέρουμε ποιές άλλες ετικέτες και γνωρίσματα μπορούμε να χρησιμοποιήσουμε.

- Η ετικέτα <xml> : Με αυτήν την ετικέτα καθορίζουμε ότι στο αρχείο αυτό περιέχεται κώδικάς σε xml μορφή. Το **version** μας ενημερώνει για την έκδοση xml που χρησιμοποιούμε και είναι η έκδοση 1 και το **encoding** για την κωδικοποίηση που χρησιμοποιούμε και είναι το utf-8.
- Η ετικέτα <manifest> αποτελεί την αρχική ετικέτα του αρχείου και πρέπει να περιέχει την ετικέτα <application> και προαιρετικά μπορεί να περιέχει άλλες ετικέτες, ενδεικτικά τις <permission>, <permission-group>, <uses-permission>, κτλ. Τα γνωρίσματα που περιέχει η ετικέτα<manifest> του παραπάνω κώδικα είναι τα εξής:
 - **package:** σε αυτό το γνώρισμα δηλώνεται το πακέτο που χρησιμοποιείται. Στο συγκεκριμένο παράδειγμα είναι το πακέτο org.example.example
 - **android:versionCode:** Εδώ δηλώνεται η έκδοση του κώδικα μας και μπορεί να είναι οποιαδήποτε ακέραια τιμή θέλουμε. Συνήθως οι μεγαλύτεροι αριθμοί δηλώνουν και πιο πρόσφατη έκδοση. Αυτό το

γνώρισμα χρησιμοποιείται εσωτερικά από το σύστημα και δεν είναι ορατό από τον χρήστη. Στο παράδειγμα μας είναι η έκδοση 1

- **android:versionName:** Εδώ δηλώνουμε τον αριθμό έκδοσης που θέλουμε να εμφανίζεται στον χρήστη. Όπως και πριν η έκδοσή μας είναι η 1.0
- Η ετικέτα `<application>` ορίζει την εφαρμογή μας. Συμπεριλαμβάνεται στην ετικέτα `<manifest>` και μπορεί να περιέχει άλλες ετικέτες όπως `<activity>`, `<service>`, `<provider>` κα. Τα γνωρίσματα που περιέχει η εν λόγω ετικέτα στον παραπάνω κώδικα είναι τα εξής:
 - **android:icon:** Με αυτό το γνώρισμα δηλώνουμε το εικονίδιο που θα εμφανίζει η εφαρμογή μας στον χρήστη. Αυτή η εικόνα θα πρέπει να βρίσκεται σε έναν από τους φακέλους `/res/drawable`
 - **android:label:** Εδώ καθορίζουμε το όνομα της εφαρμογής που θα εμφανίζεται στον χρήστη.
- Η ετικέτα `<activity>` δηλώνει ένα activity το οποίο αποτελεί εν μέρει και το user interface της εφαρμογής. Όλα τα activities της εφαρμογής μας οφείλουν να αντιπροσωπεύονται από μια ετικέτα `<activity>` μέσα στο αρχείο `AndroidManifest` της εφαρμογής μας. Το σύστημα δεν μπορεί να δει ένα activity το οποίο δεν έχει δηλωθεί στο αρχείο, οπότε δεν πρόκειται και να το εκτελέσει. Αυτή η ετικέτα συμπεριλαμβάνεται στην ετικέτα `<application>` και μπορεί να περιέχει τις ετικέτες `<intent-filter>` και `<meta-data>`. Στον παραπάνω κώδικα βλέπουμε τα εξής γνωρίσματα:
 - **android:name:** Εδώ δηλώνουμε το όνομα του activity μας. Κανονικά τα ονόματα δηλώνονται με το πλήρες όνομα τους, δηλαδή για το παράδειγμα μας έτσι `org.example.example.Example`, αλλά για λόγους συντομίας βάζουμε μια τελεία μπροστά από το όνομα του activity μας και αυτόματα το σύστημα παίρνει το πακέτο που έχουμε δηλώσει.
 - **android:label:** Σε αυτό το γνώρισμα δηλώνουμε το όνομα που θέλουμε να έχει το activity μας και εμφανίζεται στον τίτλο της όταν εκτελείτε. Συνήθως βάζουμε το όνομα της εφαρμογής μας. Εικόνα 5.3



Εικόνα 5.3: Εμφανίζεται το όνομα που εισάγαμε στο πεδίο label

- Η ετικέτα `<intent-filter>` καθορίζει τους τύπους των intents που ανταποκρίνεται το activity μας και δηλώνει της δυνατότητες του τελευταίου καθώς και το τι μπορεί ή όχι να χειριστεί. Περισσότερα για τα intents θα δούμε παρακάτω. Αυτή η ετικέτα συμπεριλαμβάνεται σε μία ετικέτα `<activity>` ή `<service>` ή `<receiver>` και πρέπει να περιέχει την ετικέτα `<action>`. Επιπρόσθετα μπορεί να περιέχει τις `<category>` και `<data>`.
 - Η ετικέτα `<action>` πρέπει να συμπεριλαμβάνεται στην `<intent-filter>` αν θέλουμε να ορίσουμε ποια intents θα περνάνε από το φίλτρο. Αν δεν έχουμε ορίσει κανένα `<action>` τότε το activity μας δεν λαμβάνει και δεν χειρίζεται κανένα intent. Στο παράδειγμα μας περιέχει το γνώρισμα name με την τιμή `android.intent.action.MAIN` που δηλώνει πως το activity μας δεν χρειάζεται δεδομένα ώστε να ξεκινήσει. Σαν να είναι η main μιας java εφαρμογής.
 - Η ετικέτα `<category>` περιέχει επιπρόσθετες πληροφορίες για το είδος της συνιστώσας που πρέπει να χειριστεί το intent. Στο παράδειγμα μας έχουμε το γνώρισμα name με την τιμή `android.intent.category.LAUNCHER` που δηλώνει ότι το activity μας είναι το πρωταρχικό activity ενός project ή αλλιώς της εφαρμογή μας

και βρίσκεται στην λίστα των ανώτερων επιπέδων (top level) της εκκίνησης εφαρμογών.

Τα activities που μπορούν να αρχικοποιούν, δηλαδή να ξεκινούν εφαρμογές και ταυτόχρονα να τις αντιπροσωπεύουν στον application launcher (προωθητή εφαρμογών), περιέχουν τα στοιχεία, <action> και <category> που περιγράψαμε με αυτές τις τιμές.

- Η ετικέτα <uses-sdk> μας δίνει τη δυνατότητα να εκφράσουμε τη συμβατότητα της εφαρμογής μας με μία ή με περισσότερες εκδόσεις της Android πλατφόρμας, δηλώνοντας τον ακέραιο αριθμό που αντιπροσωπεύει το επίπεδο του API Εικόνα 5.4. Συμπεριλαμβάνεται στην ετικέτα <manifest> και μπορεί να περιέχει τα γνωρίσματα minSdkVersion, targetSdkVersion και maxSdkVersion.

Από το παράδειγμα μας βλέπουμε ότι έχει οριστεί το γνώρισμα minSdkVersion με τιμή 8. Αυτό σημαίνει ότι για να τρέξει η εφαρμογή μας σε μια Android συσκευή πρέπει να έχει λειτουργικό σύστημα με έκδοση 2.2 και πάνω.

Platform Version	API Level
Android 2.3.3	10
Android 2.3	9
Android 2.2	8
Android 2.1	7
Android 2.0.1	6
Android 2.0	5
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

Εικόνα 5.4: Αντιστοιχία έκδοσης πλατφόρμας και έκδοσης API

5.2 Δομικά στοιχεία του android sdk

Σε αυτή την ενότητα θα περιγράψουμε συνοπτικά τα πιο σημαντικά δομικά στοιχεία ή αλλιώς δομικά αντικείμενα, που καθορίζονται στο Android SDK. Κάθε προγραμματιστής που αναπτύσσει εφαρμογές για την πλατφόρμα αυτήν πρέπει να γνωρίζει τουλάχιστον τι κάνουν και πότε αυτά χρησιμοποιούνται. Τα δομικά στοιχεία που πρέπει να γνωρίζει είναι τα εξής: α)Activities β)Intents γ)Services και δ)Content Providers.

Ένα βασικό στοιχείο που πρέπει να μελετήσουμε είναι η ασφάλεια των εφαρμογών και κατ' επέκταση του λειτουργικού συστήματος[20]. Στο κεφάλαιο 3.3.8 μιλήσαμε για την ασφάλεια χωρίς όμως να μελετήσουμε την εξειδικευμένη λειτουργία που διαθέτει προκειμένου να περιορίζει τις επιτρεπόμενες ενέργειες κάθε διαδικασίας. Αυτό θα μελετήσουμε σε αυτό το κεφάλαιο.

- **Activities**

Ένα activity είναι στην ουσία ένα user interface . Οι εφαρμογές μπορούν να ορίζουν ένα ή περισσότερα activity ώστε να χειριστούν διάφορες φάσεις του προγράμματος. Κάθε activity είναι υπεύθυνο να αποθηκεύσει την κατάσταση του ώστε να μπορεί να αποκατασταθεί αργότερα, ως μέρος του κύκλου ζωής της εφαρμογής. Τα activities κληρονομούν από την κλάση Context και έτσι μπορούμε να τα χρησιμοποιούμε ώστε να έχουμε πρόσβαση στους πόρους της εφαρμογής μας.

- **Intents**

Ένα intent είναι ένας μηχανισμός για την περιγραφή μιας ειδικής ενέργειας, όπως η «επιλογή μιας φωτογραφίας» ή «κλήση μιας επαφής».

Στο Android οτιδήποτε σχεδόν περνάει μέσα από τα intents και έτσι έχουμε αφθονία ευκαιριών για την αντικατάσταση ή την επαναχρησιμοποίηση κατασκευαστικών στοιχείων.

Για παράδειγμα υπάρχει intent «στείλε ένα email». Εάν η εφαρμογή μας πρέπει να στείλει ένα e-mail, τότε μπορούμε να καλέσουμε αυτό το intent. Ή αν γράφουμε μια νέα εφαρμογή e-mail μπορούμε να δηλώσουμε ένα activity ώστε να χειριστεί αυτό το intent και να αντικαταστήσει το πρότυπο πρόγραμμα αλληλογραφίας. Την επόμενη φορά που κάποιος θα προσπαθήσει να στείλει ένα email και έχει εγκαταστήσει την εφαρμογή μας, θα έχει την επιλογή να χρησιμοποιήσει το πρόγραμμά μας, αντί για το συνηθισμένο.

- **Services**

Ένα service είναι μία αποστολή (task) που εκτελείται στο παρασκήνιο χωρίς την απευθείας αλληλεπίδραση του χρήστη, παρόμοια με ένα daemon του Linux. Για παράδειγμα ένα πρόγραμμα που παίζει μουσική. Η μουσική μπορεί να αρχίσει από ένα activity, αλλά αν θέλουμε να κρατήσουμε την μουσική να παίζει ακόμα και όταν ο χρήστης έχει εξέλθει από user interface της μουσικής εφαρμογής και έχει μεταβεί σε ένα διαφορετικό πρόγραμμα. Έτσι, ο κώδικας που κάνει την μουσική να συνεχίζει να παίζει πρέπει να είναι σε ένα service. Αργότερα, ένα άλλο activity μπορεί να συνδεθεί με την εν λόγω υπηρεσία και να της «πει» να αλλάξει τραγούδια ή να σταματήσει την μουσική.

Το Android έρχεται ενσωματωμένο με πολλά services στα οποία μπορούμε να αποκτήσουμε πρόσβαση μέσω ενός *κατανοητού* API.

- **Content Provider**

Ένας Content Provider είναι ένα σύνολο δεδομένων τυλιγμένο σε μια προσαρμοσμένη API για να το διαβάζουμε και να το γράφουμε. Αυτός είναι ο καλύτερος τρόπος για να μοιράζονται τα συνολικά στοιχεία μεταξύ των εφαρμογών. Για παράδειγμα, η Google παρέχει ένα Content Provider για τις επαφές. Όλες οι πληροφορίες που υπάρχουν στον τηλεφωνικό κατάλογο όπως ονόματα, διευθύνσεις, τηλεφωνικοί αριθμοί, κτλ μπορούν να χρησιμοποιηθούν από οποιαδήποτε εφαρμογή μέσω του content provider.

- **Security**

Όπως αναφέρθηκε προηγουμένως στο κεφάλαιο 3.3.8, κάθε εφαρμογή τρέχει στη δική της διαδικασία του Linux. Το υλικό απαγορεύει σε μία διαδικασία να

αποκτήσει πρόσβαση στην μνήμη μιας άλλης διαδικασίας. Επιπλέον, για κάθε εφαρμογή υπάρχει συγκεκριμένο όνομα χρήστη (user id). Οποιαδήποτε αρχεία δημιουργηθούν δεν μπορούν να διαβαστούν ή να γραφτούν από άλλες εφαρμογές. Επιπλέον, η πρόσβαση σε ορισμένες κρίσιμες λειτουργίες περιορίζεται, και πρέπει συγκεκριμένα να ζητηθεί άδεια από την εφαρμογή που θέλει να τις χρησιμοποιήσει στο αρχείο Android-Manifest.xml που εξετάσαμε στο κεφάλαιο 5.1.

Όταν η εφαρμογή έχει εγκατασταθεί, ο Package Manager (Διαχειριστής Πακέτων) δίνει την άδεια ώστε να χρησιμοποιηθούν οι λειτουργίες εφόσον υπάρχουν πιστοποιητικά. Αν δεν υπάρχουν πιστοποιητικά προτρέπει τον χρήστη να επιλέξει αν θέλει η όχι να του χορηγηθεί άδεια. Παρακάτω είναι μερικές από τις πιο κοινές άδειες που μπορεί να χρειαστούν:

- INTERNET: Απόκτηση πρόσβασης στο Διαδίκτυο
- READ_CONTACTS: Διαβάζει, αλλά δεν γράφει, στα δεδομένα των επαφών του χρήστη
- WRITE_CONTACTS: Γράφει, αλλά δεν διαβάζει, στα δεδομένα των επαφών του χρήστη
- RECEIVE_SMS: Παρακολουθεί τα εισερχόμενα μηνύματα κειμένου – SMS
- ACCESS_COARSE_LOCATION: Χρησιμοποιεί προσεγγιστικά την θέση του παρόχου, όπως πύργους κινητής τηλεφωνίας ή ασύρματα δίκτυα – wifi
- ACCESS_FINE_LOCATION: Χρησιμοποιεί έναν ακριβέστερο πάροχο, όπως το GPS

Για παράδειγμα, αν θέλουμε η εφαρμογή μας να παρακολουθεί τα εισερχόμενα μηνύματα κειμένου – SMS που λαμβάνουμε πρέπει να το δηλώσουμε στο AndroidManifest.xml της εφαρμογής μας όπως παρακάτω:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.android.app.myapp" >
```

```
<uses-permission android:name="android.permission.RECEIVE_SMS"
/>
...
</manifest>
```

Το Android μπορεί να περιορίσει ακόμα και την πρόσβαση σε ολόκληρα μέρη του συστήματος με την χρήση xml ετικετών στο AndroidManifest.xml. Ακόμα μπορούμε να περιορίσουμε ποιος μπορεί να ξεκινήσει ένα activity, ποιός να ξεκινήσει ή να δεσμεύσει μια υπηρεσία, κ.α.

Από τα παραπάνω δομικά στοιχεία που περιγράψαμε θα ασχοληθούμε διεξοδικά , στο επόμενο κεφάλαιο, με το Activity γιατί αποτελεί μέρος οποιασδήποτε οπτικής εφαρμογής που αποφασίσουμε να δημιουργήσουμε.

5.3 κύκλος ζωής του activity

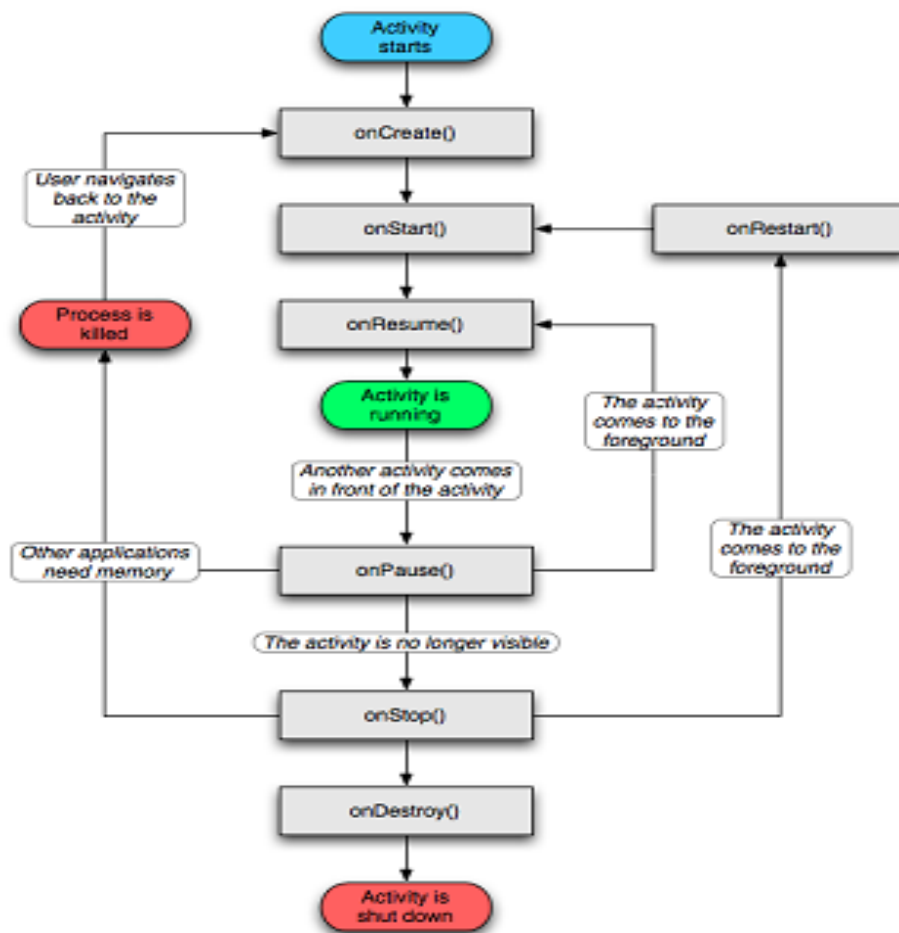
Κάθε user interface αντιπροσωπεύεται από μία κλάση Activity και κάθε Activity έχει τον δικό του κύκλο ζωής [21]. Αξίζει να το προσέξουμε ιδιαίτερος και να αναλύσουμε τις τέσσερις καταστάσεις του, καθώς και τις μεθόδους του.

Το σύστημα επίσης διαχειρίζεται τις Activities σαν μια στοίβα από activities. Όταν ένα νέο activity ξεκινάει, αυτό τοποθετείται στην κορυφή της στοίβας και περνάει στην εκτελέσιμη κατάσταση. Το προηγούμενο activity παραμένει πάντα πιο κάτω στην στοίβα και δεν έρχεται στο προσκήνιο αν δεν τελειώσει το καινούργιο. Στην Εικόνα 5.5 βλέπουμε τον κύκλο ζωής του Activity.

Οι τέσσερις καταστάσεις είναι:

- Αν ένα activity βρίσκεται στο προσκήνιο, δηλαδή στην κορυφή της στοίβας, τότε είναι ενεργό (active) ή αλλιώς εκτελείται (running).
- Αν ένα activity έχει χάσει την εστίαση του, δηλαδή κάτι άλλο είναι ενεργό πάνω από αυτό, τότε είναι σε αναστολή (paused). Ένα activity σε αναστολή είναι απολύτως «ζωντανό», διατηρεί όλες του τις πληροφορίες, αλλά μπορεί οποιαδήποτε στιγμή, το σύστημα να το «σκοτώσει» λόγω χαμηλών πόρων.
- Αν ένα activity επισκιαστεί από ένα άλλο , τότε είναι σε διακοπή (stopped). Επίσης διατηρεί όλες του τις πληροφορίες, αλλά δεν είναι στο προσκήνιο. Το σύστημα όμως μπορεί να το «σκοτώσει» λόγω χαμηλών πόρων

- Αν ένα activity έχει ανασταλεί ή διακοπεί, το σύστημα μπορεί να το διαγράψει από τη μνήμη είτε καλώντας το να τελειώσει (finish), ή απλά με τη «διακοπή» της διαδικασίας. Όταν εμφανίζεται και πάλι στο χρήστη, θα πρέπει να επανέλθει στην προηγούμενη κατάσταση.



Εικόνα 5.5:Ο Κύκλος της Ζωής του Activity

Παρακάτω περιγράφονται οι μέθοδοι που μπορούν να χρησιμοποιηθούν σε ένα activity:

- **onCreate(Bundle):** Αυτή η μέθοδος καλείται, όταν ξεκινάει το activity για πρώτη φορά και μπορούμε να τη χρησιμοποιήσουμε για να προβάλλουμε το user interface. Παίρνει μία παράμετρο, είτε κενό (null) είτε την προηγούμενη κατάσταση στην οποία βρισκόταν, αφού την είχαμε «σώσει» με την `onSaveInstanceState()` μέθοδο.

- **onStart()**: Αυτή η μέθοδος υποδεικνύει ότι το activity πρόκειται να εμφανιστεί στον χρήστη.
- **onResume()**: Αυτή η μέθοδος καλείται, όταν το activity είναι έτοιμο για αλληλεπίδραση με τον χρήστη. Εδώ μπορούμε να ξεκινήσουμε τα animation και την μουσική.
- **onPause()**: Αυτή η μέθοδος καλείται, όταν το activity περάσει σε αναστολή . Εδώ καλό θα ήταν να αποθηκεύουμε τις πληροφορίες που θέλουμε, γιατί μπορεί να είναι και η τελευταία κατάσταση στην οποία θα βρεθεί το activity.
- **onStop()**: Αυτή η μέθοδος καλείται, όταν το activity περάσει σε διακοπή (stopped). Αν οι πόροι του συστήματος είναι χαμηλοί τότε το σύστημα μπορεί να καταστρέψει το activity χωρίς να την καλέσει.
- **onRestart()**: Αυτή η μέθοδος καλείται, όταν το activity πρόκειται να ξαναεμφανιστεί μετά από μια κατάσταση διακοπής.
- **onDestroy()**: Αυτή η μέθοδος καλείται, ακριβώς πριν το activity καταστραφεί. Επίσης, μπορεί να μην κληθεί ποτέ λόγω χαμηλών πόρων.

5.4 Τρόποι δημιουργίας διεπαφής χρήστη – user interface

Ένα user interface μπορεί να σχεδιαστεί χρησιμοποιώντας μία από τις δύο μεθόδους: α)την δηλωτική και β)την διαδικαστική. Σε αυτό το κεφάλαιο θα μελετήσουμε τις μεθόδους αυτές μέσα από ένα παράδειγμα για την κάθε μια. Πρώτα όμως, θα δούμε μερικές θεμελιώδης κλάσεις για τον σχεδιασμό user interface και τέλος θα μελετήσουμε τι είναι τα Layouts (διατάξεις). Ενδεικτικά θα αναφέρουμε πως λειτουργούν μερικά από αυτά και πως χρησιμοποιούνται με τις μεθόδους σχεδιασμού που αναφέραμε.

5.4.1 Θεμελιώδης κλάσεις για τον σχεδιασμό διεπαφής χρήστη – user interface

- **Views (Προβολές)**: Οι views είναι η βασική κλάση για όλα τα οπτικά στοιχεία διασύνδεσης γνωστά και ως widgets (χειριστήρια). Όλοι οι ρυθμιστές περιβάλλοντος εργασίας χρήστη (user interface controls),

συμπεριλαμβανομένων και των Layouts, προέρχονται από αυτήν την προβολή

- **ViewGroup (Ομάδες προβολής):** Οι ομάδες προβολής είναι προέκταση της κλάσης View και μπορεί να περιέχει πολλαπλά child Views (παιδιά προβολών). Η κλάση ViewGroup επεκτείνεται (extends) ώστε να παρέχει στους Layout Managers (διαχειριστές διάταξης) τη βοήθεια σχεδίασης των στοιχείων του activity μας.
- **Activities:** Τα activities τα περιγράψαμε αναλυτικά στο προηγούμενο κεφάλαιο, αντιπροσωπεύουν το παράθυρο ή την οθόνη που εμφανίζονται. Για να εμφανίσουμε ένα user interface εκχωρούμε μία View, που μπορεί να είναι μία προσαρμοσμένη View ή ένα Layout, σε ένα activity.

5.4.2 Μελέτη των μεθόδων σχεδίασης διεπαφής χρήστη – user interface

Όπως αναφέραμε και προηγουμένως οι τρόποι σχεδίασης είναι ο διαδικαστικός και ο δηλωτικός. Όταν λέμε διαδικαστικός εννοούμε τον σχεδιασμό μέσω κώδικα. Για παράδειγμα, όταν προγραμματίζουμε μια εφαρμογή με Swing στην Java διαχειριζόμαστε με κώδικα ποια αντικείμενα θα χρησιμοποιήσουμε πχ JButton και που θα τα βάλουμε πχ σε ένα JFrame. Από την άλλη, στον δηλωτικό σχεδιασμό δεν έχουμε καθόλου κώδικα. Για παράδειγμα όταν σχεδιάζουμε μια απλή ιστοσελίδα με HTML περιγράφουμε το τι θέλουμε να βλέπουμε και όχι πώς θέλουμε να το κάνει. Παρόμοια με αυτόν τον τρόπο, της ιστοσελίδας, είναι και ο δηλωτικός σχεδιασμός στο Android με XML.

Ένα νέο Activity αρχίζει με μια κενή οθόνη, πάνω στην οποία θα σχεδιαστεί το user interface. Για να το προβάλλουμε στην οθόνη, κάνουμε override την μέθοδο onCreate() του Activity και καλούμε την μέθοδο setContentView() περνώντας σαν παράμετρο, είτε ένα στιγμιότυπο μιας View είτε ένα Layout. Αυτό μας δίνει την δυνατότητα να σχεδιάσουμε το user interface είτε με κώδικα είτε με XML.

Για να δούμε τους τρόπους με τους οποίους μπορούμε να σχεδιάσουμε ένα user interface θα χρησιμοποιήσουμε σαν παράδειγμα το project Example που δημιουργήσαμε στο κεφάλαιο 4.

1. Με XML ή αλλιώς με τον δηλωτικό τρόπο

Ανοίγοντας το Example.java κάτω από τον κατάλογο /src βλέπουμε ότι περιέχει τα παρακάτω:

```
package org.example.example;

import android.app.Activity;
import android.os.Bundle;

public class Example extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Όπως διαπιστώνεται από τον παραπάνω κώδικα, το Layout, main .xml που έχουμε δηλώσει και που βρίσκεται κάτω από τον κατάλογο res/layout το περνάμε σαν παράμετρο στη μέθοδο setContentView(). Το αρχείο main.xml περιέχει:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

Το παραπάνω αρχείο περιέχει μία xml ετικέτα, την <LinearLayout>, που αποτελεί και το Layout που επιλέξαμε ώστε να απεικονίζουμε τα στοιχεία που περιέχει. Περισσότερα για τα Layouts θα δούμε αργότερα. Επίσης, κάτω από την <LinearLayout>, βρίσκεται η ετικέτα <TextView> και είναι επέκταση της View. Με αυτήν την ετικέτα μπορούμε να εμφανίζουμε κείμενο στην οθόνη. Τα γνωρίσματα που περιέχει είναι τα εξής:

- **layout_width**: Καθορίζει το πλάτος που θα καταλάβει η View. Με την τιμή fill_parent που δώσαμε καταλαμβάνεται όλο το πλάτος της οθόνης.

- **layout_height:** Καθορίζει το ύψος που θα καταλάβει η View. Με την τιμή wrap_content καταλαμβάνει όσο ύψος χρειάζεται ώστε να είναι εμφανές τα περιεχόμενα της.
- **text:** Καθορίζει το κείμενο που θα εμφανίζεται. Αυτό γίνεται με 2 τρόπους:
 - Εισάγουμε απευθείας το κείμενο που θέλουμε ή
 - Με την χρήση αναγνωριστικού του κειμένου που θέλουμε από το αρχείο strings.xml, που βρίσκεται κάτω από τον κατάλογο /res/values. Αυτό το κάνουμε ώστε να διαχωρίσουμε τον σχεδιασμό του user interface ακόμα και από τα κείμενα. Το αρχείο strings.xml περιέχει τα παρακάτω:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello World from the TextView
    of the Activity Example!</string>
  <string name="app_name">Example</string>
</resources>
```

Από το xml αρχείο βλέπουμε ότι το αναγνωριστικό που εισάγαμε στο γνώρισμα text έχει την τιμή “Hello World from the TextView of the Example!”

Εκτελώντας το project αυτό σαν Android Application θα εμφανιστεί στον εξομοιωτή το αποτέλεσμα που φαίνεται στην Εικόνα 5.6.



Εικόνα 5.6:Εμφάνιση του κειμένου που ορίσαμε

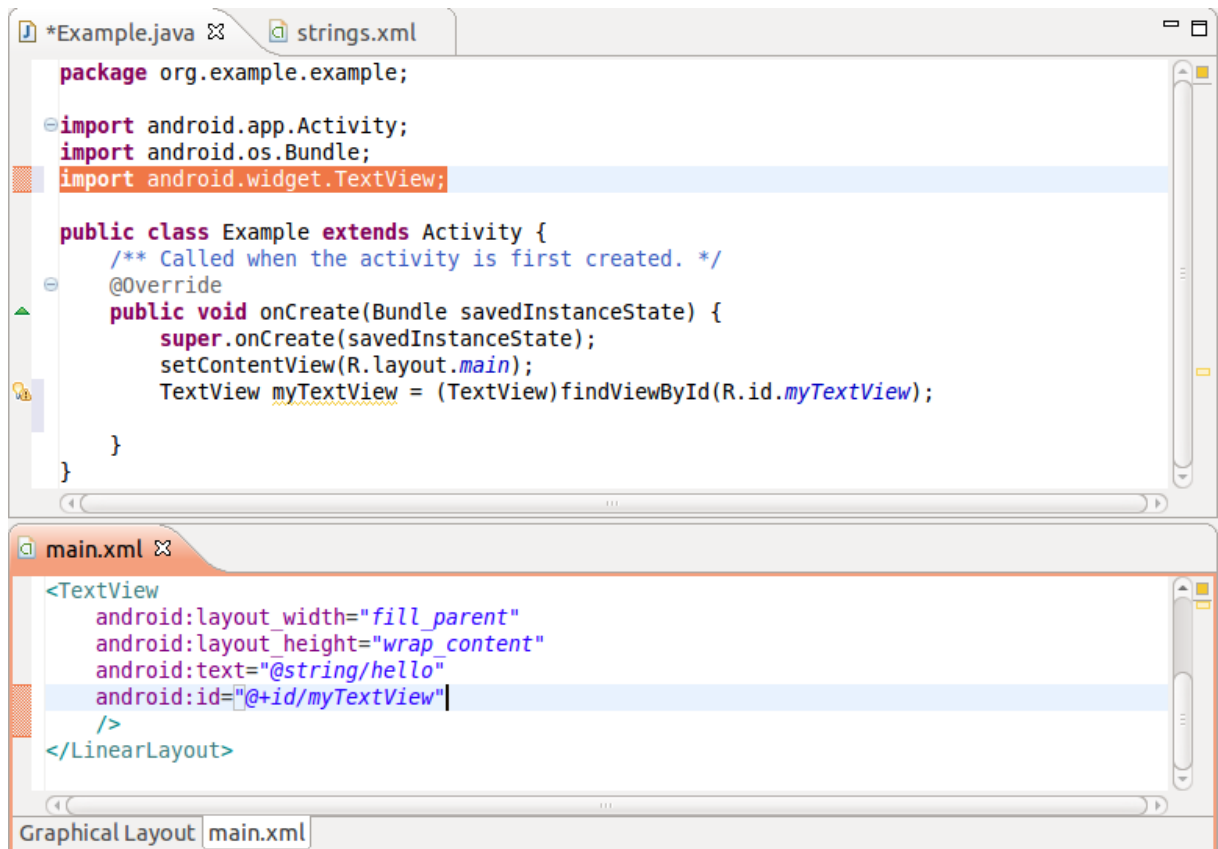
Το αποτέλεσμα που εμφανίστηκε ήταν και το αναμενόμενο. Επίσης πρέπει να αναφέρουμε ότι αν θέλουμε να σχετίσουμε την `<TextView>` με τον κώδικα μας πρέπει να προσθέσουμε μερικές γραμμές κώδικα. Ο λόγος που χρειαζόμαστε ένα τέτοιο συσχετισμό, είναι γιατί σε κάποιο σημείο της εφαρμογής μας μπορεί να χρειαστεί να αλλάξουμε το κείμενο που εμφανίζεται.

Για να πετύχουμε την σύνδεση αυτή πρέπει:

- Να προσθέσουμε το γνώρισμα `android:id="@+id/myTextView"` στην `<TextView>` στο αρχείο `main.xml`
- Να προσθέσουμε στην κλάση `Example.java` κάτω από την `setContentView()` την εντολή `TextView myTextView = (TextView) findViewById(R.id.myTextView)`. Στην μέθοδο `findViewById()` βάζουμε το αναγνωριστικό που θέλουμε να συνδέουμε και έχουμε δηλώσει στην ίδια την `View` με ένα αντικείμενο της ίδιας κλάσης. Στο παράδειγμά μας είναι η `TextView`.

- Να προσθέσουμε τέλος το `import android.widget.TextView` στην `Example.java` ώστε να μπορούμε να χρησιμοποιήσουμε την `TextView`.

Μετά από αυτές τις αλλαγές ο κώδικας μας πρέπει να μοιάζει όπως στην Εικόνα 5.7.



Εικόνα 5.7:Τελικό αποτέλεσμα του κώδικα

2. Με κώδικα

Το αποτέλεσμα που πετύχαμε με τον δηλωτικό τρόπο μπορούμε να το πετύχουμε και με τον παρακάτω κώδικα:

```
package org.example.example;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Example extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

```

        TextView myTextView = new TextView(this);
        setContentView(myTextView);
        myTextView.setText("Hello World from the TextView of the
Activity Example!");
    }
}

```

Εδώ δημιουργήσαμε ένα αντικείμενο τύπου `TextView` και του ορίσαμε το κείμενο που θέλουμε με την μέθοδο `setText()` που μας παρέχεται. Αυτό το περνάμε σαν παράμετρο στην `setContentView()` της `View` που μόλις φτιάξαμε. Εκτελώντας τον παραπάνω κώδικα ως `Android Application` θα πάρουμε το αποτέλεσμα της Εικόνας 5.6 που είδαμε παραπάνω. Λόγω του ότι δημιουργήσαμε το `user interface` με κώδικα, μας απαλλάσσει από το να συνδέσουμε την `View` με τον κώδικα αφού η σχέση αυτή υπάρχει ήδη μέσω του αντικείμενου `myTextView`. Αν χρειαστεί να σχεδιάσουμε κάτι παραπάνω, πχ να προσθέσουμε ένα ή περισσότερα κουμπιά στο `user interface` που κατασκευάσαμε, τότε πρέπει να χρησιμοποιήσουμε ένα `Layout` που στον δηλωτικό σχεδιασμό αναγκαστικά χρησιμοποιούμε από την αρχή.

5.5 προτυπες προβολες - Views

Το `Android` παρέχει μια εργαλειοθήκη με πρότυπες `Views` [22] για να μας βοηθήσει να δημιουργήσουμε απλά `user interface`. Παρακάτω περιγράφουμε μερικές από τις πιο γνωστές από αυτές:

- **TextView:** Μία πρότυπη `View` που είναι μόνο για ανάγνωση κειμένου ετικέτας, υποστηρίζει πολλαπλές γραμμές κειμένου, μορφοποίηση κειμένου και αυτόματη αναδίπλωση λέξεων.
- **EditText:** Ένα επεξεργάσιμο πλαίσιο εισαγωγής κειμένου που δέχεται πολλαπλές εγγραφές, αναδίπλωση λέξεων και υπόδειξη κειμένου.
- **Spinner:** Είναι ένα σύνθετο στοιχείο ελέγχου που εμφανίζει μια `TextView` και μια `ListView` (σχετική προβολή λίστας) η οποία μας επιτρέπει να επιλέξουμε ένα στοιχείο από μια λίστα για να εμφανιστεί στο πλαίσιο κειμένου.
- **Button:** Είναι ένα πρότυπο κουμπί.

- **CheckBox:** Είναι ένα κουμπί δύο καταστάσεων με δυνατότητα επιλογής.
- **RadioButton:** Είναι ένα κουμπί δύο καταστάσεων που αντιπροσωπεύει μια ομάδα επιλογών εκ των οποίων μόνο μία επιλογή μπορεί να είναι ενεργή κάθε φορά.

5.6 διαταξεις - Layouts

Τα Layouts [23] είναι επεκτάσεις της κλάσης ViewGroup και χρησιμοποιούνται για να τοποθετήσουν κατάλληλα στην οθόνη τα στοιχεία που περιέχουν. Κάθε Layout χρησιμοποιεί τον δικό της τρόπο τοποθέτησης των στοιχείων και μπορεί να περιέχει άλλα Layouts ώστε να πετύχουμε το επιθυμητό αποτέλεσμα.

Το Android SDK περιλαμβάνει ορισμένα απλά Layouts για να μας βοηθήσει να κατασκευάσουμε το user interface. Μερικές από αυτές είναι οι παρακάτω:

- **FrameLayout:** Το απλούστερο των Layouts, απλά «καρφιτσώνει» κάθε View που περιέχει στην πάνω αριστερή γωνία.
- **LinearLayout:** Αυτό το Layout στοιχίζει κάθε View, είτε σε κάθετη είτε σε οριζόντια γραμμή. Ένα Layout με κάθετη διάταξη έχει μια στήλη από Views, ενώ παράλληλα ένα Layout με οριζόντια διάταξη έχει μια σειρά από Views. Ο LinearLayout Manager (διαχειριστής διάταξης) μας επιτρέπει να ορίσουμε ένα «βάρος» για κάθε View, δηλαδή το σχετικό μέγεθος της κάθε μίας εντός του διαθέσιμου χώρου.

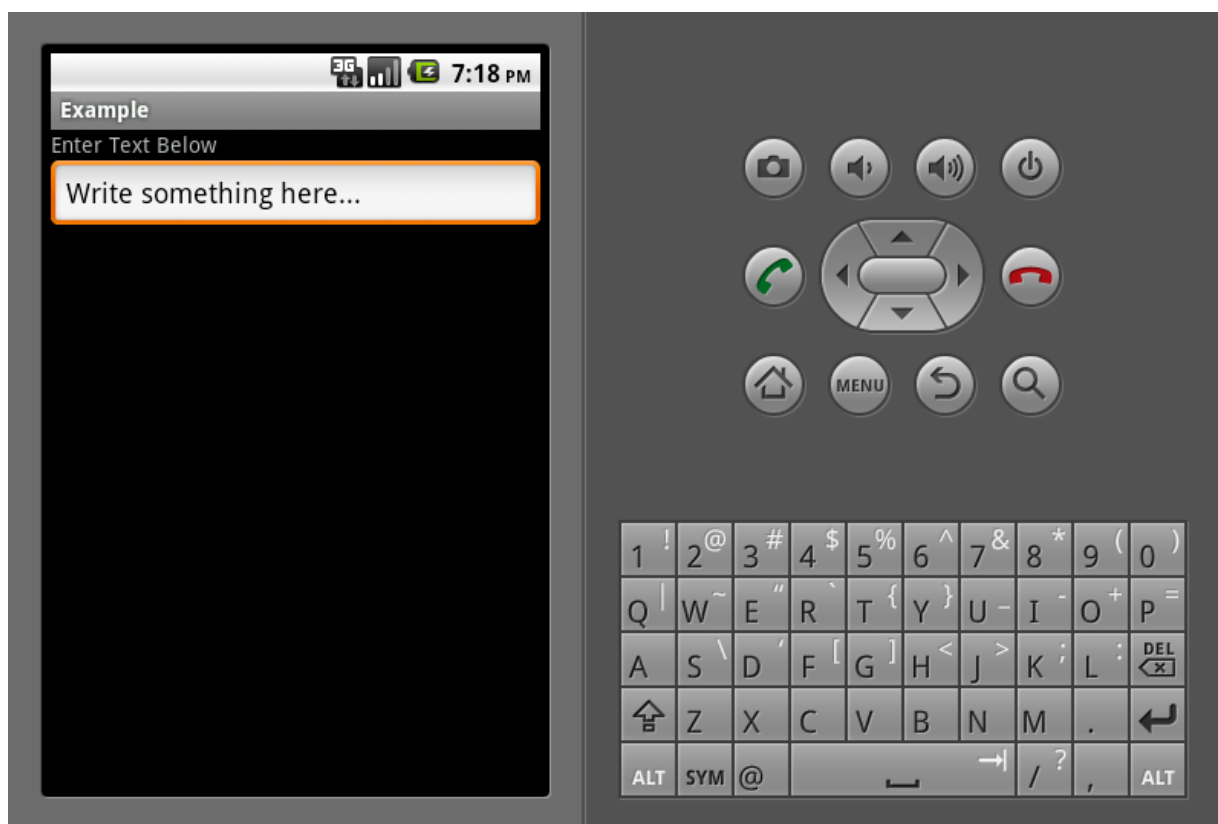
Για παράδειγμα σε ένα Linear Layout με οριζόντια διάταξη τοποθετούμε ένα κουμπί με βάρος 0,5 και συνολικό βάρος 1. Αυτό σημαίνει ότι το κουμπί θα καταλάβει τον μισό χώρο της γραμμής για αυτό και θα αφήσει τον υπόλοιπο για τα υπόλοιπα κουμπιά που θα προσθέσουμε. Για τιμή ίση με 1 απλά θα καταλάβει όλο το χώρο μη δίνοντας σημασία για τον χώρο που χρειάζονται οι άλλες Προβολές.

- **TableLayout:** Αυτό το Layout μας επιτρέπει να τοποθετήσουμε τις Views χρησιμοποιώντας ένα πλέγμα γραμμών και στηλών. Οι πίνακες μπορούν να

εκτείνονται σε πολλαπλές γραμμές και στήλες, και οι στήλες μπορούν να ρυθμιστούν ώστε να συρρικνώνονται ή να αναπτύσσονται.

- **Gallery:** Ένα Layout αυτού του τύπου εμφανίζει μια απλή σειρά στοιχείων σε μια οριζόντια κυλιόμενη λίστα.

Παρακάτω θα δούμε ένα κοινό παράδειγμα για τους δύο τρόπους δημιουργίας user interface, χρησιμοποιώντας την LinearLayout με κάθετη διάταξη των Views στην οποία θα προσθέσουμε μία TextView και μία EditText και θα μας εμφανίζει την οθόνη που φαίνεται στην Εικόνα 5.8



Εικόνα 5.8: Η οθόνη του παραδείγματος

1. Πρώτον με τον δηλωτικό τρόπο.

Με αυτόν τον τρόπο αλλάζοντας το αρχείο main.xml και βάζοντας το παρακάτω

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```

        android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:id="@+id/myTextView"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/editTxt"
        android:id="@+id/myEditText"
    />
</LinearLayout>

```

Και αλλάζοντας την Example.java στο παρακάτω επιτυγχάνεται το αποτέλεσμα της παραπάνω Εικόνας:

```

package org.example.example;

import android.app.Activity;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;

public class Example extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView myTextView =
        (TextView)findViewById(R.id.myTextView);
        EditText myEditText =
        (EditText)findViewById(R.id.myEditText);

    }
}

```

Από τις παραπάνω αλλαγές αξίζει να παρατηρήσουμε τις εξής:

- Το γνώρισμα orientation της LinearLayout, στο οποίο καθορίζουμε το Layout που θα τοποθετούνται οι Views. Οι τιμές είναι vertical και horizontal. Εδώ επιλέχθηκε η vertical (κάθετη).
- Οι τιμές που έχουν τα γνωρίσματα width και height της LinearLayout είναι και οι «γενικές οδηγίες» που ακολουθούν οι Views που περιέχει. Στην

ουσία η τιμή του *Layout* υπερಿಸχύει της τιμής της *View*, όσον αφορά την επεκτασιμότητα της δεύτερης.

- Στην `Example.java` προσθέτουμε το `import android.widget.EditText` ώστε να συνδέσουμε αυτήν την *View* με τον κώδικα μας μέσω της γραμμής `EditText myEditText = (EditText)findViewById(R.id.myEditText).`

2. **Με κώδικα** που σημαίνει ότι πρέπει να αλλάξουμε την `Example.java` σύμφωνα με τον παρακάτω κώδικα:

```
package org.example.example;

import android.app.Activity;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

public class Example extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout ll = new LinearLayout(this);
        ll.setOrientation(LinearLayout.VERTICAL);

        TextView myTextView = new TextView(this);
        EditText myEditText = new EditText(this);

        myTextView.setText("Enter Text Below");
        myEditText.setText("Write something here...");

        int lHeight =
        LinearLayout.LayoutParams.FILL_PARENT;
        int lWidth =
        LinearLayout.LayoutParams.WRAP_CONTENT;

        ll.addView(myTextView, new
        LinearLayout.LayoutParams(lHeight, lWidth));
        ll.addView(myEditText, new
        LinearLayout.LayoutParams(lHeight, lWidth));
        setContentView(ll);
    }
}
```

```
}  
}
```

Από τον παραπάνω κώδικα διαπιστώνουμε τα εξής:

- Ότι έχει δηλωθεί ένα αντικείμενο `LinearLayout` και θα πρέπει να προσθέσουμε το `import android.widget.LinearLayout` στη κλάση ώστε να μπορούμε να χρησιμοποιήσουμε το συγκεκριμένο `Layout`.
- Με τη μέθοδο `setOrientation()` καθορίζουμε την κάθετη εμφάνιση των `Views`.
- Με την μέθοδο `setText()` καθορίζουμε το κείμενο που θέλουμε να εμφανίζεται.
- Στις ακέραιες μεταβλητές `IHeight` και `IWidth` αποθηκεύουμε τις τιμές που θέλουμε να έχει το `Layout`.
- Με τη μέθοδο `addView()` προσθέτουμε στο `Layout` τις `Views` που θέλουμε να εμφανιστούν περνώντας επίσης και τις παραμέτρους του `Layout`.

5.7 ΕΠΙΣΗΜΑΝΣΕΙΣ

Πρέπει να αναφέρουμε ότι και οι δύο τρόποι, ο δηλωτικός με `xml` και ο διαδικαστικός με κώδικα, είναι σωστοί. Το ποιον από τους δύο θα επιλέξουμε είναι καθαρά θέμα προτίμησης του προγραμματιστή. Η Google μας προτείνει να χρησιμοποιούμε το δηλωτικό τρόπο εφόσον είναι δυνατόν και έτσι να ξεχωρίζουμε το σχεδιασμό του `user interface` από τον υπόλοιπο κώδικα της εφαρμογής μας.

Επίσης, τα `Layouts` που μας προσφέρει το `SDK` μπορεί να μην ικανοποιούν τις ανάγκες μας με αποτέλεσμα να καταφύγουμε σε ένα προσαρμοσμένο `Layout` η αλλιώς εργαλείο [24]. Για παράδειγμα, αν το `Layout` που θέλουμε να χρησιμοποιήσουμε με επιπλέον μεθόδους είναι η `LinearLayout`, τότε απλά την επεκτείνουμε και προσθέτουμε τις μεθόδους που χρειαζόμαστε.

Αν δεν μας ικανοποιεί κανένα από τα `Layouts` τότε μπορούμε να επεκτείνουμε την `ViewGroup` ή την `View` ώστε να φτιάξουμε κάτι προσαρμοσμένο για την εφαρμογή μας. Αυτό συμβαίνει συχνά στις εφαρμογές παιχνιδιών και ειδικά σε

αυτές που δεν απαιτούν μεγάλο ρυθμό επεξεργασίας πλαισίων[25]. Ένα τέτοιο παράδειγμα είναι το παιχνίδι Snake από τον επίσημο οδηγό του Android [26].

ΚΕΦΑΛΑΙΟ 6: ΥΠΟΣΤΗΡΙΞΗ ΠΟΛΛΑΠΛΩΝ ΟΘΟΝΩΝ

Εξ αιτίας της διαφορετικότητας των κινητών τηλεφώνων και της οθόνης που διαθέτουν οι συσκευές που έχουν ως λειτουργικό σύστημα το Android πρέπει να σχεδιάσουμε ανάλογα και την εφαρμογή μας. Δηλαδή είτε να σχεδιαστεί για να λειτουργεί σε συγκεκριμένο μέγεθος οθόνης και έτσι να αποκλείσουμε τα υπόλοιπα μεγέθη, είτε να προνοήσουμε και με κατάλληλους ελέγχους για τον εντοπισμό του μεγέθους και της πυκνότητας της οθόνης να τρέχει και ο κατάλληλος κώδικας. Επειδή κάθε προγραμματιστής θέλει η εφαρμογή του να “τρέχει” σε όσο το δυνατόν περισσότερες συσκευές [27] σε αυτό το Κεφάλαιο θα αναπτύξουμε διάφορες έννοιες τις οποίες θα εξηγήσουμε πριν προχωρήσουμε και θα εξετάσουμε τους τρόπους με τους οποίους μπορούμε να πετύχουμε αυτήν την υποστήριξη.

6.1 όροι και εννοιες

- **Μέγεθος οθόνης (Screen size):** Για τον καθορισμό του πραγματικού φυσικού μεγέθους, υπολογίζουμε τη διαγώνιο οθόνης. Για λόγους απλούστευσης, το Android χωρίζει όλα τα πραγματικά μεγέθη των οθονών σε τέσσερα γενικά μεγέθη: μικρό, κανονικό, μεγάλο, και πολύ μεγάλο
- **Ανάλυση (resolution):** Είναι ο συνολικός αριθμός των φυσικών pixels σε μια οθόνη. Αν και εκφράζεται συχνά ως γινόμενο πλάτους επί ύψος, δεν συνεπάγεται μια συγκεκριμένη αναλογία αυτών

- **Πυκνότητα (density):** Με βάση την ανάλυση της οθόνης είναι η εξάπλωση των pixel σε όλη το φυσικό πλάτος και ύψος της οθόνης. Μια οθόνη με χαμηλή πυκνότητα έχει λιγότερα διαθέσιμα pixels εξαπλωμένα σε όλο το πλάτος και το ύψος της οθόνης. Μια οθόνη με υψηλότερη πυκνότητα έχει περισσότερα, μερικές φορές πολύ περισσότερα, εξαπλωμένα pixels σε ολόκληρη την ίδια περιοχή.

Η πυκνότητα μιας οθόνης είναι σημαντικό χαρακτηριστικό γιατί ένα στοιχείο του user interface, όπως ένα κουμπί, του οποίου το ύψος και το πλάτος ορίζονται από pixels, θα φαίνεται μεγαλύτερο σε μία οθόνη με μικρότερη πυκνότητα και μικρότερο σε οθόνη μεγαλύτερης πυκνότητας. Για λόγους απλούστευσης το Android χωρίζει όλες τις πραγματικές πυκνότητες οθόνης σε τέσσερις γενικευμένες πυκνότητες: χαμηλή, μεσαία, μεγάλη, και πολύ μεγάλη

- **Pixel ανεξάρτητης πυκνότητας (density-independent pixel (dp)):** Είναι μια εικονική μονάδα pixel όπου οι εφαρμογές μπορούν να χρησιμοποιήσουν την εικονική μονάδα για να καθορίσουν το user interface τους, δηλαδή να καθορίσουν τις διαστάσεις του Layout ή την θέση με τέτοιο τρόπο που να μη επηρεάζονται από την πυκνότητα της οθόνης.

Το Pixel ανεξάρτητης πυκνότητας αντιστοιχεί σε 1 φυσικό pixel σε μια οθόνη 160 dpi, που είναι και η πυκνότητα αναφοράς την οποία έχει ορίσει η πλατφόρμα. Κατά το χρόνο εκτέλεσης, η πλατφόρμα αυτόματα χειρίζεται οποιαδήποτε κλιμάκωση των μονάδων dp προκειμένου να εμφανιστεί το σωστό αποτέλεσμα, με βάση την πραγματική πυκνότητα της οθόνης. Η μετατροπή των μονάδων dp σε pixels οθόνης είναι απλή: $\text{pixels} = \text{dps} * (\text{πυκνότητα} / 160)$. Για παράδειγμα, σε 240 dpi της οθόνης, 1 dp θα ισοδυναμούσε με 1,5 πραγματικά pixels.

6.2 τροποι υποστηριξης

Όπως αναφέρθηκε και προηγουμένως, το Android χωρίζει το εύρος των πραγματικών υποστηριζόμενων μεγεθών και αναλύσεων σε :

- Ένα σύνολο τεσσάρων γενικευμένων μεγεθών: μικρό, κανονικό, μεγάλο, και πολύ μεγάλο
- Ένα σύνολο τεσσάρων γενικευμένων πυκνοτήτων: χαμηλή (ldpi), μέτρια (mdpi), υψηλή,(hdpi), και πολύ υψηλή (xhdpi)

	Low density (120), <i>ldpi</i>	Medium density (160), <i>mdpi</i>	High density (240), <i>hdpi</i>	Extra high density (320), <i>xhdpi</i>
<i>Small screen</i>	QVGA (240x320)			
<i>Normal screen</i>	WQVGA400 (240x400) WQVGA432 (240x432)	HVGA (320x480)	WVGA800 (480x800) WVGA854 (480x854)	
<i>Large screen</i>		WVGA800* (480x800) WVGA854* (480x854)		
<i>Extra Large screen</i>				

* To emulate this configuration, specify a custom density of 160 when creating an AVD that uses a WVGA800 or WVGA854 skin.

Εικόνα 6.1: Μεγέθη και πυκνότητες οθόνης

Στην Εικόνα 6.1 βλέπουμε τις 4 κατηγορίες οθόνης καθώς και τις 4 κατηγορίες για την πυκνότητα. Οι τιμές που περιέχει ο πίνακας είναι ο τύπος οθόνης και η ανάλυση της και περιέχονται στο Android SDK ώστε να δημιουργούμε τις αντίστοιχες εικονικές μηχανές.

Οι εφαρμογές μπορούν να παρέχουν προσαρμοσμένους πόρους, κυρίως Layouts, για οποιαδήποτε από τα τέσσερα γενικευμένης κατηγορίας μεγέθη και μπορούν να παρέχουν πόρους για οποιαδήποτε από τις τέσσερις γενικευμένης κατηγορίες πυκνότητας. Οι εφαρμογές δεν χρειάζεται να συνεργάζονται με το πραγματικό φυσικό μέγεθος ή την πυκνότητα της οθόνης της συσκευής.

Κατά το χρόνο εκτέλεσης (run time) η πλατφόρμα ανάλογα με την κατηγορία της πυκνότητας ή το μέγεθος της οθόνης, χειρίζεται την φόρτωση του κατάλληλου μεγέθους των πόρων. Δηλαδή είτε φορτώνει τους πόρους από τον αντίστοιχο κατάλογο είτε αν δε βρει τότε προσαρμόζει πόρους από άλλο πιο κοντινό στα δεδομένα κατάλογο με τρόπο ώστε αντιστοιχούν στις πραγματικές τιμές pixel της οθόνης.

Οι γενικευμένες κατηγορίες μεγέθους και πυκνότητας της οθόνης βασίζονται σε μια ρύθμιση αναφοράς (baseline) στην οποία έχει εκχωρηθεί το μέγεθος μιας κανονικής (normal) οθόνης με πυκνότητα μέτρια (mdpi). Απλούστερα, αν έχουμε μία εικόνα και θέλουμε να εμφανίζεται με συγκεκριμένο τρόπο σε συσκευές, ανάλογα με την πυκνότητα τους, τότε φτιάχνουμε αυτήν την εικόνα σε διάφορες αναλύσεις και την

τοποθετούμε σε κάθε ένα από τους καταλόγους που αντιστοιχεί η ανάλυση. Πχ την εικόνα που θα εμφανίζεται στην υψηλής πυκνότητας οθόνη, στον φάκελο drawable-hdpi, την εικόνα για χαμηλής πυκνότητας, στον drawable-ldpi κτλ

Αν και η πλατφόρμα μας επιτρέπει να παρέχουμε προσαρμοσμένους πόρους για τις διάφορες διαστάσεις και διαμορφώσεις πυκνότητας, δεν χρειάζεται να γράψουμε προσαρμοσμένο κώδικα ή να παρέχουμε προσαρμοσμένους πόρους για κάθε συνδυασμό μεγέθους και πυκνότητας της οθόνης. Η πλατφόρμα παρέχει ισχυρές δυνατότητες συμβατότητας, που περιγράφονται παρακάτω, και μπορεί να τις χρησιμοποιήσει ώστε να χειριστεί το μεγαλύτερο μέρος του έργου της εφαρμογής μας σε οποιαδήποτε οθόνη συσκευής. Κατά το χρόνο εκτέλεσης, η πλατφόρμα προσφέρει δύο τύπους υποστήριξης στην εφαρμογή μας, για να εξασφαλίσει την καλύτερη δυνατή εμφάνιση στην τρέχουσα οθόνη της συσκευής:

1. **Προ-κλιμάκωση των πόρων:** Με βάση την πυκνότητα της τρέχουσας οθόνης, η πλατφόρμα φορτώνει αυτόματα τους κατάλληλους για αυτήν πόρους από την εφαρμογή μας και τους εμφανίζει χωρίς να τους επεξεργαστεί. Αν δεν ταιριάζουν οι πόροι που είναι διαθέσιμοι για αυτήν την οθόνη, τότε αυτόματα φορτώνει τους πόρους που ταιριάζουν καλύτερα σε αυτήν με βάση την γενικευμένη πυκνότητα της τρέχουσας οθόνης.

Η πλατφόρμα υποθέτει ότι η εφαρμογή μας ακολουθεί τις ρυθμίσεις αναφοράς (baseline) δηλαδή μια οθόνη με πυκνότητα μεσαίας κατηγορίας (160dpi). Έτσι εξασφαλίζει την καλή εμφάνιση των πόρων μας για αυτήν την οθόνη. Εφόσον δεν έχουν φορτωθεί πόροι για την τρέχουσα πυκνότητα, από τον αντίστοιχο κατάλογο των πόρων, τότε φορτώνει τους πόρους αναφοράς.

Για παράδειγμα, εάν η πυκνότητα της τρέχουσας οθόνης είναι υψηλή, φορτώνει τους πόρους που βρίσκονται στο φάκελο με κατάληξη hdpi και τους χρησιμοποιεί χωρίς να τους επεξεργαστεί. Εάν δεν υπάρχουν διαθέσιμοι πόροι σε αυτόν τον φάκελο, τότε η πλατφόρμα χρησιμοποιεί τους πόρους προεπιλογής και τους κλιμακώνει από την πυκνότητα αναφοράς που είναι η μέτρια, σε πόρους για υψηλή πυκνότητα.

2. **Αυτόματη κλιμάκωση των pixel σε διαστάσεις και συντεταγμένες:** Εάν η εφαρμογή δηλώσει ότι δεν υποστηρίζει διαφορετικές πυκνότητες οθόνης, η

πλατφόρμα αυτόματα κλιμακώνει τις τιμές που είναι σε pixel και αφορούν συντεταγμένες και τις τιμές που είναι σε pixel και αφορούν διαστάσεις. Το κάνει αυτό ώστε να εξασφαλίσει ότι τα στοιχεία που έχουν οριστεί με τιμές σε pixel θα εμφανίζονται περίπου στο ίδιο φυσικό μέγεθος όπως θα ήταν στην πυκνότητα αναφοράς (160dpi).

Η πλατφόρμα χειρίζεται αυτή την κλιμάκωση χωρίς να το γνωρίζει η εφαρμογή και δηλώνει σε αυτήν τις συνολικά διαβαθμισμένες διαστάσεις και όχι τις φυσικές διαστάσεις της οθόνης.

Για παράδειγμα, μια συσκευή με λειτουργικό σύστημα Android, οθόνη τύπου WVGA και πυκνότητα υψηλή (240dpi), η οποία είναι 480x800 και περίπου το ίδιο μέγεθος με μία παραδοσιακή HVGA οθόνη, τρέχει μία εφαρμογή που δηλώνει ότι δεν υποστηρίζει πολλαπλές πυκνότητες. Στην περίπτωση αυτή, το σύστημα θα «ξεγελάσει» την εφαρμογή όταν αυτή ρωτήσει για τις διαστάσεις της οθόνης και θα της δηλώσει διαστάσεις 320x533. Στη συνέχεια, όταν η εφαρμογή ξεκινήσει τις λειτουργίες σχεδίασης στην οθόνη, όπως η σχεδίαση ενός ορθογώνιου 100x100, το σύστημα θα μετατρέψει αυτόματα με κλιμάκωση τις συντεταγμένες αυτές στην αντίστοιχη ποσότητα, δηλαδή θα το σχεδιάσει με 150x150 ώστε να πετύχουμε περίπου την ίδια εμφάνιση όπως αν το είχαμε σχεδιάσει για την πυκνότητα αναφοράς. Το ίδιο συμβαίνει και προς την άλλη κατεύθυνση, εφόσον η εφαρμογή εκτελείται σε μια οθόνη χαμηλότερης πυκνότητας, οι συντεταγμένες κλιμακώνονται προς τα κάτω.

Στο αρχείο `AndroidManifest.xml` έχουμε την δυνατότητα να δηλώσουμε την ετικέτα `<supports-screen>` [28] η οποία παρέχει πληροφορίες για το ποιες συσκευές υποστηρίζει η εφαρμογή μας καθώς και αν υποστηρίζει όλες τις πυκνότητες ή όχι. Αυτές, παίρνουν δύο τιμές, `true` ή `false` και δηλώνονται με τα γνωρίσματα `smallScreens` για μικρές οθόνες, `normalScreens` για κανονικές, `largeScreens` για μεγάλες, `xlargeScreens` για πολύ μεγάλες οθόνες και τέλος το `anyDensity` για την πυκνότητα.

Αν ένα γνώρισμα οθόνης έχει τιμή `true` σημαίνει ότι υποστηρίζει αυτόν τον τύπο οθόνης και επομένως μπορούμε να την εγκαταστήσουμε. Σε αντίθετη περίπτωση δεν υποστηρίζεται και δεν μπορούμε να την εγκαταστήσουμε.

Αν το γνώρισμα της πυκνότητας έχει την τιμή true τότε το σύστημα ξέρει ότι υποστηρίζουμε όλες τις πυκνότητες και δεν εφαρμόζει καμία κλιμάκωση. Για τιμή false «ξεγελά» το σύστημα και χρησιμοποιεί την πυκνότητα αναφοράς (160dpi) και την κλιμάκωση των πόρων. Ένα παράδειγμα της ετικέτας <supports-screen> που υποστηρίζει όλες τις οθόνες εκτός των μικρών οθονών και όλες τις πυκνότητες είναι το παρακάτω:

```
<supports-screens    android:smallScreens="false"    android:normalScreens="true"
android:largeScreens="true"                android:xlargeScreens="true"
android:anyDensity="true"/>
```

Άλλη μία ετικέτα που αξίζει να αναφέρουμε είναι η <uses-sdk> [29] και μπορεί να περιέχει τα γνωρίσματα:

- **minSdkVersion:** Δηλώνουμε το κατώτερο API για το οποίο μπορεί η εφαρμογή μας να τρέξει
- **targetSdkVersion:** Δηλώνουμε το API για το οποίο σχεδιάστηκε η εφαρμογή μας
- **maxSdkVersion:** Δηλώνουμε το μεγαλύτερο API για το οποίο μπορεί να τρέξει η εφαρμογή μας

Αν δεν χρησιμοποιήσουμε κάποιο από τα παραπάνω γνωρίσματα, τα οποία δεν είναι υποχρεωτικά, τότε το σύστημα θεωρεί ότι κάνει για όλα τα API ξεκινώντας από το 1. Προτείνεται να χρησιμοποιούμε τουλάχιστον το γνώρισμα minSdkVersion για να αποφύγουμε την εγκατάσταση και κατ' επέκταση τη δυσλειτουργία ή και την μη λειτουργία της εφαρμογής στις συσκευές που δεν πληρούν τις προϋποθέσεις.

Καλύτερα ακόμα είναι να χρησιμοποιούμε και το targetSdkVersion ώστε να καθορίζουμε για ποιο API είναι γραμμένη η εφαρμογή μας. Χρησιμοποιώντας το γνώρισμα minSdkVersion ή και τα δύο γνωρίσματα που αναφέραμε, το σύστημα γνωρίζει αυτόματα ποιες οθόνες υποστηρίζονται δηλαδή ανάλογα με το επιλεγμένο API αυτόματα το σύστημα γνωρίζει ποιού τύπου οθόνες υποστηρίζονται, Πίνακας 6.1 [30]. Για παράδειγμα αν στο minSdkVersion επιλέξουμε την έκδοση 9 του API τότε αυτόματα η εφαρμογή μας δηλώνει ότι υποστηρίζει και μπορεί να εγκατασταθεί σε οποιαδήποτε οθόνη και πυκνότητα.

Ιδιότητες	Προεπιλεγμένη τιμή όταν το minSdkVersion ή το targetSdkVersion είναι μικρότερο ή ίσο του 4	Προεπιλεγμένη τιμή όταν το minSdkVersion ή το targetSdkVersion είναι μεγαλύτερο ή ίσο του 5
android:smallScreens	false	true
android:normalScreens	true	true
android:largeScreens	false	true
android:anyDensity	false	true
Ιδιότητες	Προεπιλεγμένη τιμή όταν το minSdkVersion ή το targetSdkVersion είναι μικρότερο ή ίσο του 8	Προεπιλεγμένη τιμή όταν το minSdkVersion ή το targetSdkVersion είναι μεγαλύτερο ή ίσο του 9
android:xlargeScreens	false	true

Πίνακας 6.1: Υποστήριξη οθονών από το ανάλογο API

6.3 Περαιτέρω αναλυση για την Προ-κλιμάκωση και την αυτόματη κλιμάκωση των εικωνων

Όταν μια εικόνα bitmap έχει φορτωθεί από τους πόρους της εφαρμογής, η πλατφόρμα προσπαθεί να την προ-επεξεργαστεί ώστε να ταιριάζει με την πυκνότητα της οθόνης. Για παράδειγμα, αν έχουμε ένα 100x100 εικονίδιο στον κατάλογο res /drawable/ και

φορτωθεί το εικονίδιο ως bitmap σε μια οθόνη υψηλής πυκνότητας, το Android αυτόματα θα κλιμακώσει την εικόνα και θα παράγει μια εικόνα bitmap 150x150.

Αυτός ο μηχανισμός προ-κλιμάκωσης λειτουργεί ανεξάρτητα από την πηγή. Για παράδειγμα, μια εφαρμογή η οποία προορίζεται για οθόνη υψηλής πυκνότητας μπορεί να έχει εικόνες -bitmaps μόνο στο res/drawable-hdpi / κατάλογο. Αν μία από τις εικόνες είναι 240x240 και φορτωθεί σε μια οθόνη μέσης πυκνότητας, η αντίστοιχη εικόνα θα μετατραπεί σε 160x160.

Η πλατφόρμα προ-κλιμακώνει τους πόρους ανάλογα με τις ανάγκες της, είτε η εφαρμογή εκτελείται με ενεργοποιημένη την πυκνότητα συμβατότητας, είτε όχι (όπως καθορίζεται από την τιμή του android: anyDensity). Ωστόσο όταν τρέχει με πυκνότητα συμβατότητας, η πλατφόρμα συνεχίζει να αναφέρει το μέγεθος της προ-κλιμακωμένης εικόνας και των άλλων πόρων σαν να είχαν φορτωθεί σε ένα περιβάλλον μεσαίας πυκνότητας. Για παράδειγμα, όταν η πυκνότητα-συμβατότητας είναι ενεργοποιημένη, εάν φορτώσουμε μια εικόνα 76x76 για εμφάνιση σε μια οθόνη υψηλής πυκνότητας, η πλατφόρμα θα προ-κλιμακώσει την εικόνα σε 114x114 εσωτερικά. Παρόλα αυτά, το API θα δηλώνει ακόμη το μέγεθος της εικόνας ως 76x76 στην εφαρμογή μας.

Τέλος υπάρχει περίπτωση να μην θέλουμε το Android αυτόματα να κλιμακώσει τους πόρους μας αλλά να εμφανίζονται όπως ακριβώς σχεδιάστηκαν. Ο τρόπος για να το πετύχουμε αυτό είναι να δημιουργήσουμε, αν δεν υπάρχει, τον κατάλογο drawable-hdpi κάτω από τον κατάλογο res και εκεί να τοποθετήσουμε όλους τους πόρους που δεν θέλουμε να κλιμακωθούν.

ΚΕΦΑΛΑΙΟ 7: ΕΝΤΟΠΙΣΜΟΣ ΣΦΑΛΜΑΤΩΝ (DEBUGGING)

Το Android SDK περιλαμβάνει ένα σύνολο εργαλείων για να μας βοηθήσει στον εντοπισμό σφαλμάτων και στη βελτίωση της απόδοσης των εφαρμογών μας. Στα παρακάτω υποκεφάλαια θα μελετήσουμε αυτά τα εργαλεία [31] και θα δούμε τη διευκόλυνση που μας παρέχει η χρήση του IDE, Eclipse.

Για να λαμβάνουμε όμως πληροφορίες σχετικά με τον εντοπισμό σφαλμάτων από την εφαρμογή μας στο μηχάνημα που κάνουμε την ανάπτυξη των εφαρμογών, πρέπει να

προσθέσουμε στο AndroidManifest.xml στην ετικέτα <application> την ιδιότητα android:debuggable="true".

7.1 το εργαλείο Android Debug Bridge (adb)

Το εργαλείο Android Debug Bridge (adb), που έρχεται μαζί με το SDK και βρίσκεται στον κατάλογο <sdk>/platform-tools/, παρέχει έναν τρόπο για να διαχειριστεί την κατάσταση εξομοιωτή / εικονική συσκευή ή μιας συνδεδεμένης με usb συσκευής Android. Το adb είναι χτισμένο από τρεις συνιστώσες:

- **Ένα πρόγραμμα πελάτη (client)**, που τρέχει στον υπολογιστή που αναπτύσσουμε την εφαρμογή μας. Μπορούμε να δημιουργήσουμε έναν πελάτη από ένα κέλυφος – shell χρησιμοποιώντας την κατάλληλη εντολή του adb. Άλλα εργαλεία του Android, όπως το ADT plugin και το DDMS επίσης μπορούν να δημιουργήσουν πελάτες adb.
- **Ένα διακομιστή (server)**, ο οποίος τρέχει ως διαδικασία υποβάθρου (background process) στον υπολογιστή που αναπτύσσουμε την εφαρμογή. Ο διακομιστής διαχειρίζεται την επικοινωνία μεταξύ του πελάτη και του adb δαίμονα που εκτελείτε σε έναν εξομοιωτή ή συσκευή.
- **Ένα δαίμονα (daemon)**, ο οποίος τρέχει ως διαδικασία υποβάθρου για κάθε εξομοιωτή ή συσκευή.

Όταν ξεκινάμε έναν πελάτη adb, ο πελάτης ελέγχει πρώτα κατά πόσον υπάρχει μια διαδικασία διακομιστή adb που να εκτελείται ήδη. Εάν δεν υπάρχει, ξεκινάει η διαδικασία διακομιστή. Όταν ο διακομιστής ξεκινήσει, δεσμεύει την τοπική θύρα TCP 5037 και ακούει τις εντολές που αποστέλλονται από τους adb πελάτες. Όλοι οι adb πελάτες χρησιμοποιούν τη θύρα 5037 για να επικοινωνήσουν με τον διακομιστή adb και στη συνέχεια αυτός, συνδέεται με όλες τις τρέχουσες εικονικές μηχανές ή συσκευές. Αυτές τις εντοπίζει σαρώνοντας όλους τους αριθμούς με μονό αριθμό στην περιοχή 5555 – 5585 δηλαδή το φάσμα που χρησιμοποιείται από τις εικονικές μηχανές / συσκευές.

Όταν ο διακομιστής βρει ένα δαίμονα adb, δημιουργεί μια σύνδεση με αυτήν την πόρτα (port). Μετά από αυτήν την διαδικασία είμαστε έτοιμοι και μπορούμε να

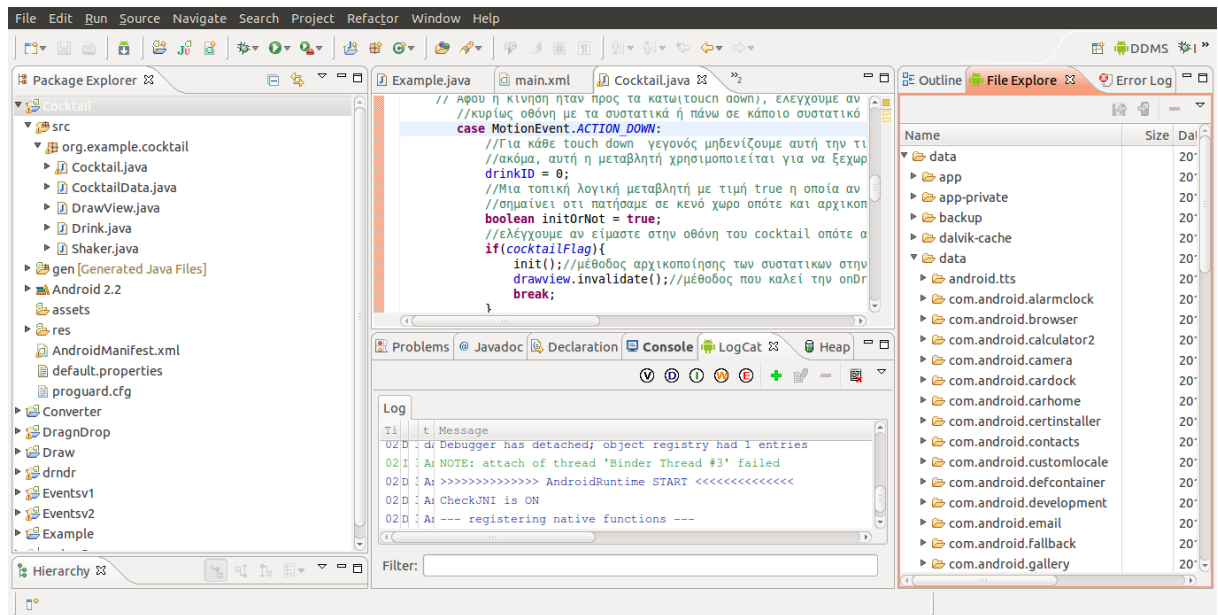
χρησιμοποιήσουμε εντολές adb για τον έλεγχο και την διαχείριση αυτών των εικονικών μηχανών/συσκευών.

Μερικές ενδεικτικές εντολές του adb είναι και οι παρακάτω:

- Η εντολή **adb devices** μας επιστρέφει μια λίστα με τις εικονικές μηχανές / συσκευές που έχει συνδεθεί ο διακομιστής
- Η εντολή **adb install** *<path_to_apk>* εγκαθιστά το apk ή αλλιώς την εφαρμογή στην εικονική μηχανή / συσκευή
- Η εντολή **adb pull** *<remote>* *<local>* αντιγράφει ένα αρχείο από την συσκευή στον υπολογιστή μας
- Η εντολή **adb push** *<local>* *<remote>* αντιγράφει ένα αρχείο από τον υπολογιστή μας στην συσκευή

Το adb έχει πληθώρα εντολών και παραμέτρων [32] που μπορούμε να κάνουμε χρήση. Χρησιμοποιώντας όμως το Eclipse και το ADT plugin μας διευκολύνει με την παραθυρική του εμφάνιση και αυτοματοποιεί / κρύβει πολλές διαδικασίες που χρειάζεται να γίνουν ώστε να διαχειριζόμαστε την εικονική μηχανή / συσκευή.

Για παράδειγμα οι εντολές αντιγραφής αρχείου μπορούν να γίνουν ευκολότερα από το παράθυρο File Explorer, Εικόνα 7.1. Αυτό μπορούμε να το βρούμε και να το προσθέσουμε στο περιβάλλον ανάπτυξης από το μενού Window → Show View → Other και επιλέγουμε File Explorer. Στα δεξιά μας εμφανίζεται αυτό το παράθυρο. Ακόμα η δημιουργία apk πακέτων, η ψηφιακή υπογραφή αυτών και άλλων χαρακτηριστικών και εντολών γίνεται ευκολότερα μέσα από το ADT plugin του Eclipse.

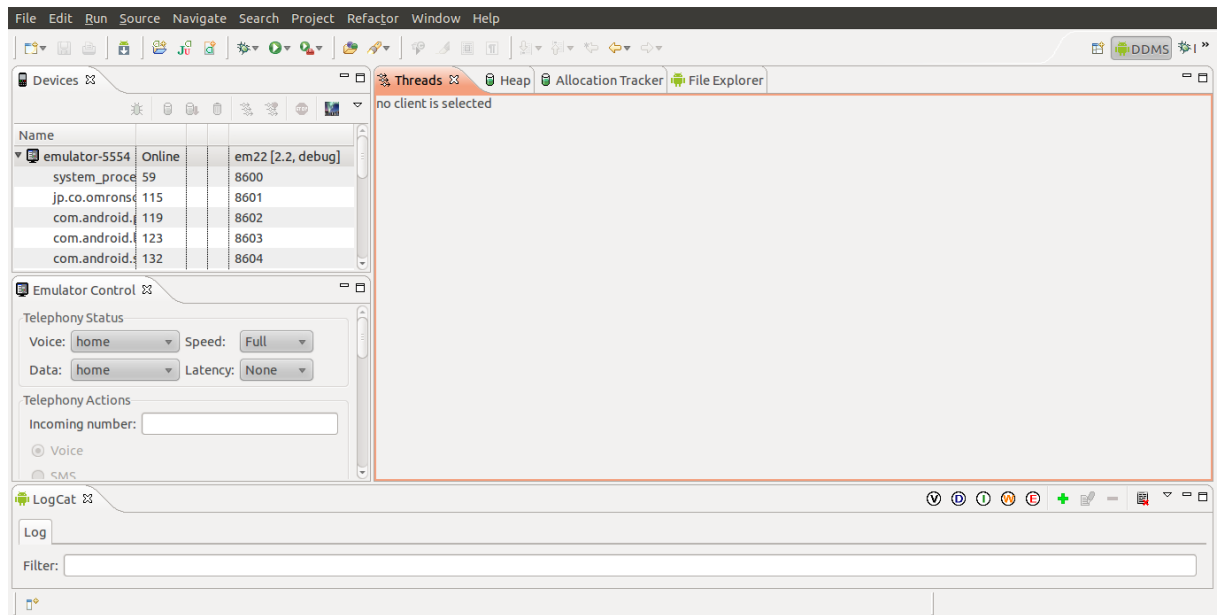


Εικόνα 7.1: Το παράθυρο File Explorer φαίνεται στα δεξιά μας

7.2 το εργαλείο *Dalvik Debug Monitor server (ddms)*

Μαζί με το SDK του Android έρχεται και το εργαλείο Dalvik Debug Monitor Server (DDMS) και βρίσκεται στον κατάλογο <sdk>/tools. Αυτό μας παρέχει πληροφορίες, για τα νήματα και το heap της συσκευής, τη διαδικασία , βγάζει φωτογραφία την οθόνη της συσκευής, προσομοιώνει συντεταγμένες για το GPS, δημιουργεί εικονικά μηνύματα (sms) και κλήσεις και πολλά άλλα. Το DDMS λειτουργεί ως μεσάζων για να συνδέσουμε το IDE με τις εφαρμογές που τρέχουν στο σύστημα. Αφού εγκαταστήσουμε την εφαρμογή μας στη εικονική μηχανή / συσκευή μπορούμε να δούμε το DDMS επιλέγοντας στο Eclipse, Window → Open Perspective → DDMS.

Εικόνα 7.2



Εικόνα 7.2: Τα παράθυρα του DDMS

Μέσα στο DDMS υπάρχουν τέσσερις ομάδες που παρέχουν διάφορα είδη εντοπισμού σφαλμάτων δεδομένων:

- **Devices (Συσκευές):** Εμφανίζει τις συνδεδεμένες συσκευές Android, συμπεριλαμβανομένων των εικονικών και των πραγματικών Android συσκευών
- **Emulator Control (Ελεγχος Εξομοιωτή):** Παρέχει πολλαπλούς ελέγχους για την προσθήκη γεγονότων και δεδομένων στον εξομοιωτή όπως:
 - **Κατάσταση Τηλεφώνου (Telephony Status):** προσδιορίζει τη μορφή φωνής και δεδομένων, την ταχύτητα του δικτύου και την λανθάνουσα κατάσταση - latency.
 - **Τηλεφωνικές Ενέργειες (Telephony Actions):** Παρέχει έναν τρόπο να ώστε να κάνουμε μία ψεύτικη κλήση ομιλίας ή μηνύματος (SMS) προς τον εξομοιωτή.
 - **Εντοπισμός Ελέγχου (Location Control):** Παρέχει έναν τρόπο ώστε να στείλουμε ένα ψεύτικο μήνυμα GPS στον εξομοιωτή.
- Το κάτω παράθυρο έχει την καρτέλα LogCat και δείχνει όλα τα δεδομένα που καταγράφουμε από την συσκευή σε πραγματικό χρόνο.

- Το δεξιό πλαίσιο περιέχει τέσσερις καρτέλες: Thread, Heap, Allocation Tracker, και την File Explorer. Αυτές είναι ως επί το πλείστον οι καρτέλες που χρησιμοποιούνται για να αναλύσουμε τη διαδικασία.

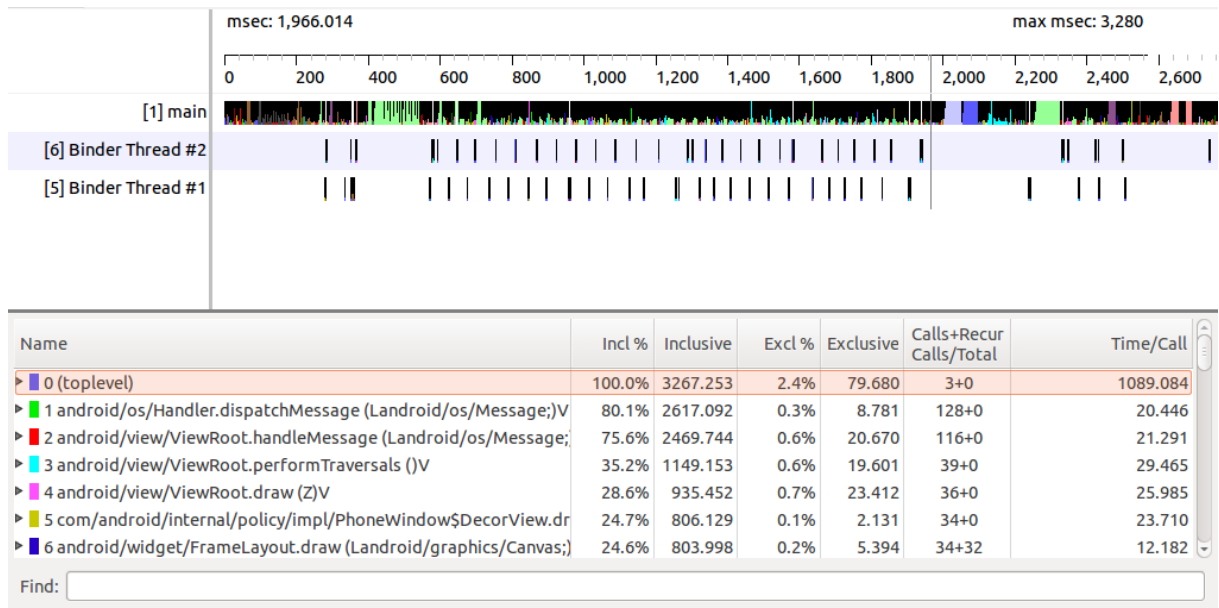
7.3 το εργαλείο Traceview

Το εργαλείο Traceview προβάλλει γραφικά τα αρχεία καταγραφής που αποθηκεύτηκαν από την εφαρμογή μας και μπορεί να μας βοηθήσει να καταγράψουμε και να διορθώσουμε την απόδοσή της.

Για να χρησιμοποιήσουμε το εργαλείο Traceview, θα πρέπει να δημιουργήσουμε αρχεία καταγραφής που περιέχουν τις πληροφορίες που θέλουμε να αναλύσουμε. Για να το καταφέρουμε αυτό, πρέπει να συμπεριλάβουμε την κλάση Debug στον κώδικά μας και να καλέσουμε τις μεθόδους της ώστε να καταγράψουμε τις πληροφορίες στο δίσκο. Όταν η εφαρμογή μας τερματίσει τότε μπορούμε να χρησιμοποιήσουμε το εργαλείο Traceview ώστε να εξετάσουμε τα αρχεία που καταγράφηκαν και να πάρουμε τις χρήσιμες πληροφορίες κατά τον χρόνο εκτέλεσης, όπως ο χρόνος για τη κλήση των μεθόδων κλπ.

Για παράδειγμα, αν θέλουμε να καταγράψουμε τις πληροφορίες και τους χρόνους από την στιγμή που δημιουργείται μέχρι την στιγμή που τερματίζει η εφαρμογή μας, τότε θα πρέπει να χρησιμοποιήσουμε τις μεθόδους της Debug. Συγκεκριμένα στην αρχή της onCreate() θα βάλουμε την Debug.startMethodTracing("something") και στο τέλος της onDestroy() την Debug.stopMethodTracing().

Στην πρώτη μέθοδο της Debug που χρησιμοποιήσαμε βάζουμε ένα όνομα που θέλουμε για το αρχείο που θα περιέχει τις πληροφορίες μας για το TraceView και θα βρίσκεται με το όνομα something.trace στην SD κάρτα της εικονικής ή πραγματικής συσκευής μας. Τέλος εκτελούμε από το κέλυφος (shell) την εντολή traceview something.trace ώστε να πάρουμε την γραφική απεικόνιση των πληροφοριών μας. Ένα παράδειγμα μιας γραφικής απεικόνισης φαίνεται στην Εικόνα 7.3.

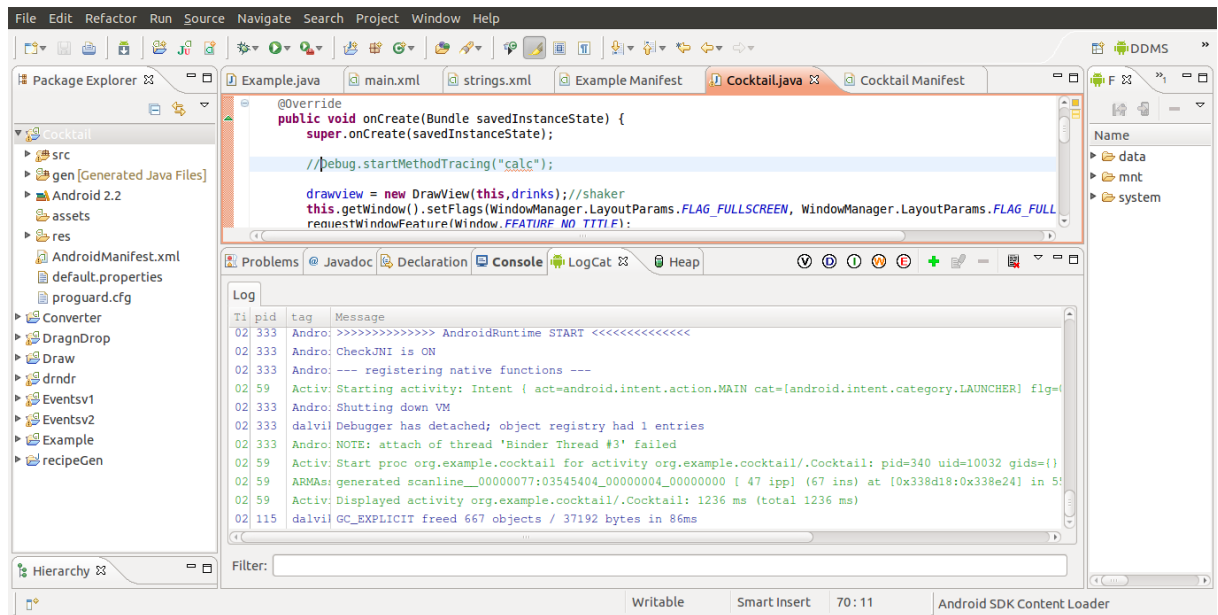


Εικόνα 7.3: Αποτελέσματα του εργαλείου traceview

7.4 το εργαλείο logcat

Το LogCat είναι ένα εργαλείο που μας παρέχει το Android SDK για την καταγραφή των δεδομένων του συστήματος και της εφαρμογής σε πραγματικό χρόνο. Μπορούμε να το χειριστούμε ως αυτόνομο εργαλείο ή ως μέρος του εργαλείου DDMS.

Για να εμφανίσουμε το εργαλείο αυτό στο περιβάλλον ανάπτυξης πηγαίνουμε από το μενού Window → Show View → Other και επιλέγουμε LogCat. Εικόνα 7.4



Εικόνα 7.4: Το παράθυρο του εργαλείου LogCat

Χρησιμοποιώντας την κλάση Log [33] στην εφαρμογή μας μπορούμε να καταγράφουμε μηνύματα τα οποία εμφανίζονται στο logcat. Κάτι αντίστοιχο με την εμφάνιση μηνυμάτων της System.out.println στο παράθυρο Console στη Java. Μερικές από τις μεθόδους της Log είναι οι παρακάτω:

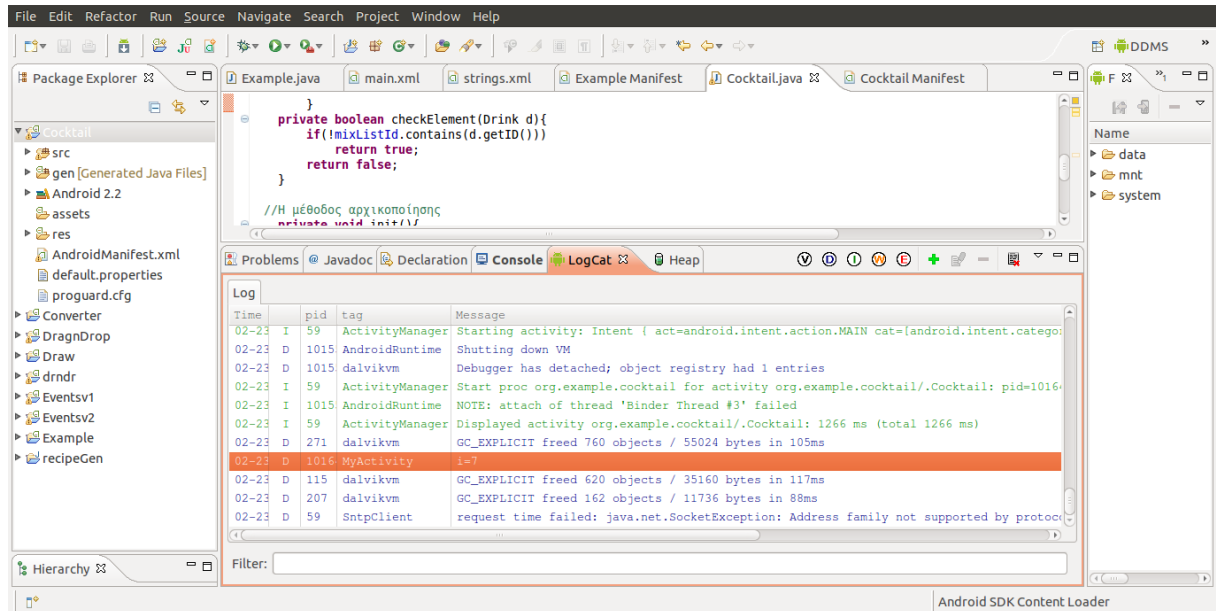
- Log.v(): verbose
- Log.d(): debug
- Log.i(): info
- Log.w(): warn
- Log.e(): error

Όλες οι μέθοδοι έχουν δύο ορίσματα, την σταθερή αλφαριθμητική τιμή TAG που ορίζουμε εμείς, και από σύμβαση έχει αυτό το όνομα, και το κείμενο που θέλουμε να εμφανίζει. Για παράδειγμα, αν θέλουμε να εμφανίζουμε ένα μήνυμα στο logcat κάθε φορά που εκτελείτε μία μέθοδος της εφαρμογής μας μπορούμε να χρησιμοποιήσουμε την Log.d() στην αρχή αυτής για να το πετύχουμε. Στην αρχή της κλάσης μας δηλώνουμε την TAG έτσι

- private static final String TAG = "MyActivity";

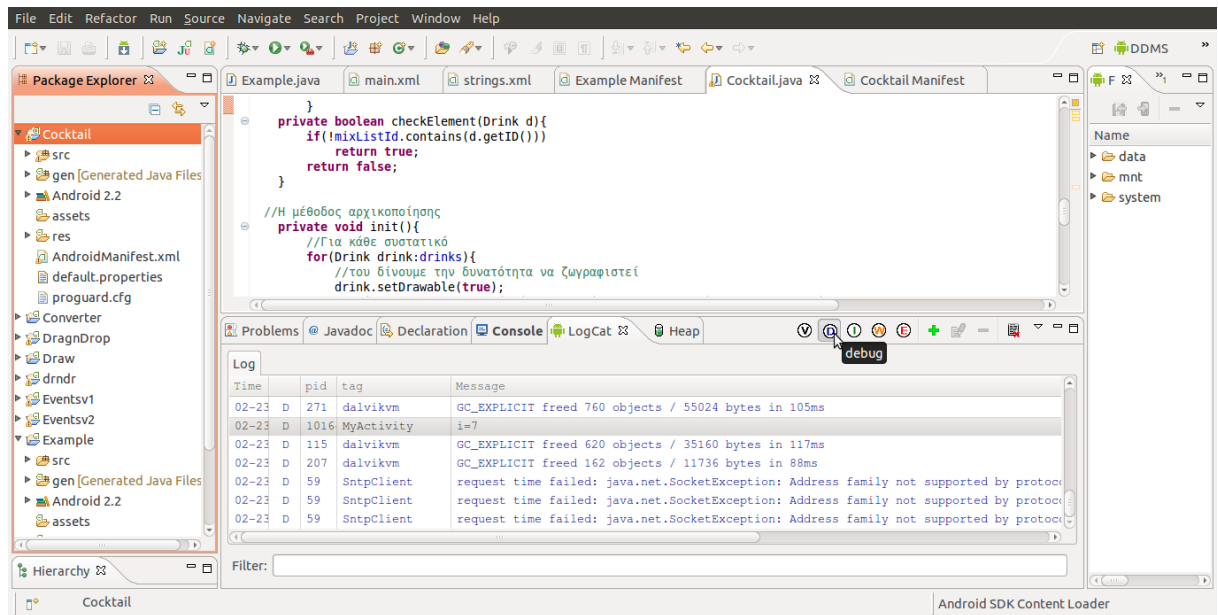
- και χρησιμοποιούμε στην αρχή της μεθόδου μας την `Log.d(TAG, "i=" + i);`

Κάνοντας τα παραπάνω θα εμφανιστεί στο logcat το κείμενο με την τιμή που περιέχει η δεύτερη παράμετρος της Log και με TAG αυτό που ορίσαμε. Εικόνα 7.5



Εικόνα 7.5: Αποτέλεσμα χρήσης της Log στο LogCat

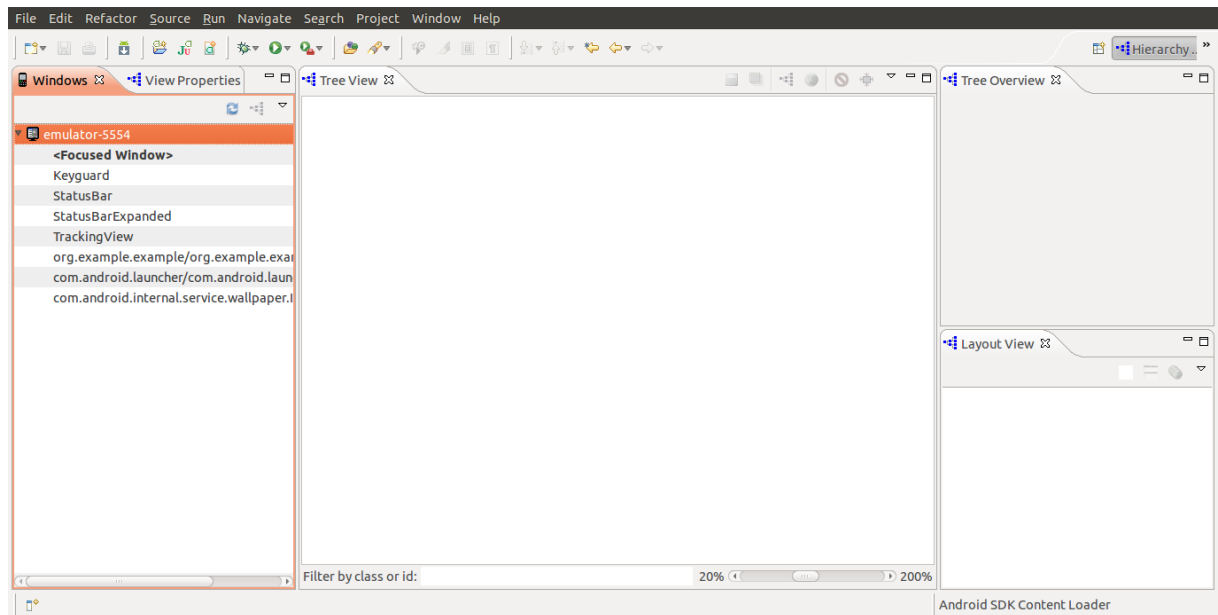
Επίσης λόγω του μεγάλου όγκου πληροφορίας που μας παρέχει το logcat αλλά και λόγω της δικιάς μας χρήσης των διαφόρων τύπων μηνύματος μπορεί να χρειαζόμαστε να βλέπουμε μόνο τα μηνύματα που θέλουμε, πχ debug, error κτλ. Αυτό το πετυχαίνουμε χρησιμοποιώντας τα φίλτρα που μας παρέχει το logcat. Στην Εικόνα 7.6 βλέπουμε μόνο τα μηνύματα που είναι τύπου debug.



Εικόνα 7.6: Εμφάνιση μόνο των μηνυμάτων debug

7.5 το εργαλείο hierarchy viewer

Ένας χρήσιμος τρόπος για να εντοπίσουμε και να κατανοήσουμε το περιβάλλον εργασίας χρήστη - ui είναι με τη χρήση του εργαλείου Hierarchy Viewer. Αυτό μας παρέχει μια οπτική αναπαράσταση της ιεραρχίας των Προβολών της διάταξης. Για να εργαστούμε με αυτό το εργαλείο στο Eclipse πηγαίνουμε στο Window → Open Perspective→Other και επιλέγουμε Hierarchy View. Θα εμφανιστεί ένα περιβάλλον όπως στην Εικόνα 7.7



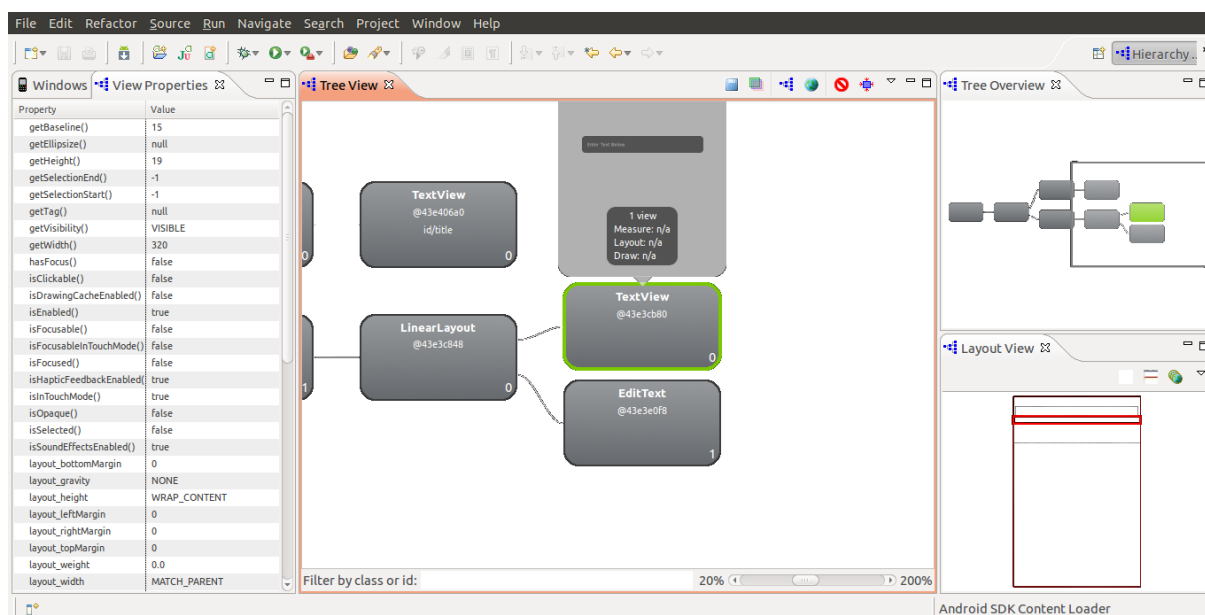
Εικόνα 7.6: Το περιβάλλον του εργαλείου Hierarchy Viewer

Στα αριστερά, βλέπουμε την λίστα με τις συνδεδεμένες εικονικές συσκευές ή και πραγματικές και επιλέγουμε ποιανής συσκευής τις εφαρμογές θέλουμε να προβάλουμε. Στη συνέχεια επιλέγουμε την εφαρμογή που θέλουμε για εντοπισμό σφαλμάτων ή για βελτιστοποίηση του περιβάλλοντος χρήστη. Επιλέχθηκε η εφαρμογή Example από προηγούμενα παραδείγματα. Στην παρακάτω εικόνα, Εικόνα 7.7, υπάρχουν τέσσερα παράθυρα τα οποία είναι:

- **View Properties:** Εμφανίζει τις ιδιότητες της επιλεγμένης Προβολής
- **Tree View:** Ένα δενδρικό διάγραμμα των Προβολών
- **Tree Overview:** Σε μικρογραφία όλο το δενδρικό διάγραμμα Προβολών
- **Layout View:** Προβάλλει το επιλεγμένο στοιχείο πάνω στην Διάταξη

Όλα τα παραπάνω σχετίζονται μεταξύ τους και όταν επιλέγουμε μία Προβολή από το παράθυρο του Tree View όλα τα υπόλοιπα παράθυρα ενημερώνονται και εμφανίζουν τις κατάλληλες πληροφορίες. Σε ένα σύστημα Android, υπάρχει περιορισμός σχετικά με το δενδρικό διάγραμμα που μία εφαρμογή μπορεί να δημιουργήσει. Συγκεκριμένα το βάθος του δέντρου δεν μπορεί να είναι μεγαλύτερο από 10 και το πλάτος του ευρύτερο από 50.

Επιλέγοντας κάποια από τις Προβολές, μας εμφανίζονται πληροφορίες για αυτή μέσα στο Tree View, σε ένα πλαίσιο πάνω από την Προβολή, που αναφέρει τους χρόνους που χρειάστηκε για κάθε μία από τις προβολές που περιέχει, συμπεριλαμβανομένου και της ίδιας. Ανάλογα με το πόσο καθυστερεί μία προβολή την εφαρμογή μας η αλλιώς τον χρόνο που χρειάζεται μία Προβολή να εμφανιστεί, το εργαλείο μας εμφανίζει και ένα χρώμα γύρο από αυτήν. Το πράσινο θεωρείτε καλός χρόνος ενώ το κόκκινο και κίτρινο όχι. Εικόνα 7.7. Αν δούμε κόκκινο ή κίτρινο χρώμα σε κάποια από αυτές δεν σημαίνει απαραίτητα ότι δεν έχει καλή απόδοση αφού αυτό μπορεί να συμβαίνει σε ορισμένες περιπτώσεις, όπως όταν καλείτε μια μέθοδο της προβολής αυτής για ένα συγκεκριμένο συμβάν.



Εικόνα 7.7: Η εφαρμογή Example στο εργαλείο Hierarchy Viewer

7.5 Συνοψη

Παραπάνω είδαμε τα εργαλεία που μας προσφέρει το Android SDK ώστε να μας βοηθήσει στον εντοπισμό σφαλμάτων του κώδικα αλλά και στην καταγραφή της απόδοσης του. Ο σκοπός της καταγραφής αυτής, είναι να εντοπίσουμε ποιες μέθοδοι «καθυστερούν» την εφαρμογή μας ώστε αν μπορούμε να τις βελτιώσουμε. Ακόμα, είδαμε την διευκόλυνση που μας προσφέρει το IDE Eclipse μαζί με το plugin του

Android αφού πολλές από τις διαδικασίες που γίνονται μέσω του κέλυφος του SDK τις αυτοματοποιεί. Αναφέρουμε επίσης ότι μπορούμε να χρησιμοποιήσουμε και τα εργαλεία του Eclipse για τον εντοπισμό σφαλμάτων και έτσι να προχωρήσουμε και σε άλλες μεθόδους εντοπισμού σφαλμάτων, όπως με σημεία διακοπής – breakpoints κ.α

ΚΕΦΑΛΑΙΟ 8: ΔΟΚΙΜΗ – TESTING ΤΩΝ ΕΦΑΡΜΟΓΩΝ

Στο κεφάλαιο αυτό θα μελετήσουμε τον τρόπο με τον οποίο πρέπει να δοκιμάζουμε τις εφαρμογές μας [34], ώστε να είμαστε σίγουροι ότι η εφαρμογή μας υποστηρίζει όσο το δυνατόν περισσότερους τύπους οθόνης και εκδόσεις πλατφόρμας.

Όσο καλά και αν σχεδιάσαμε την εφαρμογή μας και όσο πιστά και αν ακολουθήσαμε τις οδηγίες για την βέλτιστη εμφάνιση του γραφικού περιβάλλοντος και της συμβατότητας με πολλές συσκευές μπορεί να διαπιστώσουμε μέσα από δοκιμές ότι δεν εμφανίζεται όπως θα θέλαμε ή δεν εκτελείτε σωστά έως και καθόλου σε μερικές συσκευές. Αν βρούμε πρόβλημα, σε θέμα υποστήριξης, για κάποια από αυτές τις συσκευές, τότε προσπαθούμε να το διορθώσουμε αλλιώς αποκλείουμε τις συσκευές με το συγκεκριμένο χαρακτηριστικό. Τα πιθανά προβλήματα υποστήριξης μπορεί να είναι:

- Λόγω οθόνης, επειδή κάποιες συσκευές διαθέτουν, είτε πολύ μικρές οθόνες με χαμηλή πυκνότητα, συνήθως σε αυτές εμφανίζεται πρόβλημα, είτε γιατί έχουν πολύ μεγάλες οθόνες με υψηλή πυκνότητα και η εφαρμογή μας δεν εμφανίζεται όπως θα θέλαμε.
- Λόγω API, επειδή μπορεί να κάνουμε χρήση κάποιας λειτουργίας ενός API που είναι μεγαλύτερο από αυτό της συσκευής και σαν αποτέλεσμα να υπολειτουργεί ή και να μην λειτουργεί.

Αν διαπιστώσουμε πως το πρόβλημα οφείλεται στην πρώτη περίπτωση, λόγω οθόνης, τότε αποκλείουμε όλες τις συσκευές με αυτόν τον τύπο οθόνης ώστε η εφαρμογή μας να μην εγκαθίσταται σε αυτές. Αυτό το επιτυγχάνουμε μέσω του αρχείου `AndroidManifest.xml`, που αναλύσαμε στο Κεφάλαιο 5.1.4, και χρησιμοποιώντας την `xml` ετικέτα `<support-screens>` βάζουμε την τιμή `false` στον αντίστοιχο τύπο οθόνης.

Αν το πρόβλημα είναι λόγω API, δηλαδή κάνουμε χρήση κάποιας λειτουργίας η οποία δεν υποστηρίζεται σε παλαιότερα API και αυτό δεν μπορεί να αντικατασταθεί με άλλο κώδικα, τότε πάλι μέσω του αρχείου `AndroidManifest.xml` το αποκλείουμε. Αυτό γίνεται με το να ανεβάσουμε το ελάχιστο επίπεδο του API στο επίπεδο που ξεκίνησε να υποστηρίζεται η λειτουργία μας. Για παράδειγμα, αν το επίπεδο υποστήριξης της λειτουργίας είναι το 8 τότε στο `AndroidManifest.xml` αλλάζουμε στην ετικέτα `<uses-sdk>` την ιδιότητα `android:minSdkVersion="8"`.

Για να δοκιμάσουμε την εφαρμογή μας, χρησιμοποιούμε εξομοιωτές με διάφορους τύπους οθόνης και με διάφορες εκδόσεις Android. Μερικοί από αυτούς τους τύπους εξομοίωσης [35] μπορούν να είναι αυτοί του πίνακα 8.1.

Όνομα εξομοιωτή	Αριθμός έκδοσης του Android	Τύπος οθόνης
em15	1.5	HVGA(320x480)
em16	1.6	HVGA(320x480)
em16-qvga	1.6	QVGA(200x320)
em21-854	2.1	WVGA854(480x854)
em22-800	2.2	WVGA800(480x800)
em22-1024	2.2	Προσαρμοσμένο (1024x600)

em22	2.2	HVGA(320x480)
------	-----	---------------

Πίνακας 8.1: Χαρακτηριστικά εξομοιωτών

Τέλος, για να είμαστε σίγουροι ότι η εφαρμογή εκτελείτε σωστά δεν είναι αρκετό να την δοκιμάσουμε μόνο σε εξομοιωτές αλλά πρέπει να την δοκιμάσουμε και σε τουλάχιστον μία ή περισσότερες πραγματικές συσκευές.

ΚΕΦΑΛΑΙΟ 9: Η ANDROID ΕΦΑΡΜΟΓΗ «COCKTAIL»

Στο κεφάλαιο αυτό θα παρουσιάσουμε την εφαρμογή που υλοποιήσαμε για τον σκοπό αυτής της πτυχιακής.

9.1 περιγραφή της εφαρμογής «cocktail»

Σκοπός της εφαρμογής αυτής είναι ο χρήστης της κινητής συσκευής να μπορεί να εισάγει με drag n' drop ορισμένα συστατικά-ποτά στο σέικερ και αφού πατήσει πάνω στο σέικερ ή κουνήσει το τηλέφωνο του, η εφαρμογή να εμφανίζει το κοκτέιλ που θα προκύψει, ως αποτέλεσμα τις μίξης των συστατικών.

Πίνακας 9.1: Λίστα των cocktails της εφαρμογής

Cocktail	Συστατικά
BLACK RUSSIAN	Vodka, Kahlua

BRAVE BULL	Tequila, Kahlua
DAIQUIRI	Rum, Lime, Sugar
ENGLISH HIGHBALL	Gin, Brandy, Vermouth, Soda
FIFTY FIFTY COCKTAIL	Gin, Vermouth
FLORIDA SPECIAL	Rum, Vermouth, Grapefruit
HARVARD COOLER	Sugar, Soda, Brandy
HOT DECK	Whiskey, Martini, Rum
IRISH	Brandy, Martini, Lemon juice
GIN & TONIC	Gin, Tonic
KENTUCKY COCKTAIL	Whiskey, Anana juice
GIN FIZZ	Gin , Lemon juice, Soda
RICKEY	Whiskey, Soda, Lemon juice
VERA RUSH	Rum, Anana juice

Όλα τα συστατικά που περιέχει ο παραπάνω πίνακας εμφανίζονται στην οθόνη της κινητής συσκευής από την εφαρμογή από όπου ο χρήστης μπορεί να τα επιλέξει και να τα «ρίξει» στο σέικερ. Η Εικόνα 9.1 εμφανίζει την οθόνη αυτή.



Εικόνα 9.1: Το γραφικό περιβάλλον της εφαρμογής μέσα από τον εξομοιωτή

Όταν ο χρήστης ρίξει κάποια συστατικά στο σέικερ, τότε αυτά εξαφανίζονται από την οθόνη και προστίθενται στο σέικερ. Στην παρακάτω εικόνα φαίνεται πως ο χρήστης έχει εισάγει τα συστατικά Whiskey, Martini, Rum.



Εικόνα 9.2: Τα συστατικά του χρήστη περιέχονται στο shaker

Βάσει του παραπάνω πίνακα, από τα συστατικά αυτά αποτελείται το κοκτέιλ Hot Deck. Επομένως όταν ο χρήστης πατήσει πάνω στο σέικερ, το αποτέλεσμα αναμένεται να είναι το κοκτέιλ αυτό, όπως φαίνεται και στην παρακάτω εικόνα. Τέλος, ο χρήστης μπορεί να πραγματοποιήσει την ίδια διαδικασία εκ νέου πατώντας οπουδήποτε στον κενό χώρο.



Εικόνα 9.3: Αποτέλεσμα μίξης των περιεχομένων του shaker

Στην περίπτωση που εισάγουμε συστατικά τα οποία δεν δημιουργούν κάποιο συνδυασμό κοκτέιλ, λαμβάνουμε ένα μήνυμα μη ύπαρξης του κοκτέιλ στη λίστα. Στην παρακάτω Εικόνα εισάγουμε τα συστατικά Vodka, Whiskey, Gin τα οποία δεν αποτελούν κάποιο κοκτέιλ.



Εικόνα 9.4: Εμφάνιση μηνύματος μη ύπαρξης του cocktail στη λίστα

Για να τερματίσουμε την εφαρμογή και να επιστρέψουμε στην αρχική οθόνη του συστήματος, πατάμε το κουμπί μενού της συσκευής το οποίο μας εμφανίζει το μενού με την επιλογή exit, την οποία και επιλέγουμε. Εικόνα 9.5



Εικόνα 9.5: Εμφάνιση του μενού και των επιλογών του

ΚΕΦΑΛΑΙΟ 10: ΣΥΜΠΕΡΑΣΜΑΤΑ

Από τα παραπάνω κεφάλαια καταλήγουμε ότι η πλατφόρμα της Google, Android είναι και παρέχει τα εξής:

1. Είναι ένα ολοκληρωμένο λειτουργικό σύστημα για κινητές συσκευές.
2. Αποτελεί μια ολοκληρωμένη πλατφόρμα ανάπτυξης εφαρμογών και προσφέρει πληθώρα εργαλείων και μεθόδων.
3. Παρέχει πολλά και τεκμηριωμένα κείμενα ανάπτυξης εφαρμογών και του τρόπου λειτουργίας του συστήματος
4. Μπορεί να καλύψει τις ανάγκες τόσο των μέσων όσο και των εξειδικευμένων χρηστών.
5. Οι κατασκευαστές των κινητών συσκευών έχουν την δυνατότητα να προσαρμόσουν ανάλογα με τις ανάγκες τους το λειτουργικό σύστημα Android χωρίς κόστος.

Μέσα από την ενασχόληση των παραπάνω τεχνολογιών, που μελετήσαμε στα προηγούμενα κεφάλαια, καταλήγουμε στην επίτευξη του στόχου μας.

Στόχος μας ήταν, η εφαρμογή που αναπτύχθηκε να λειτουργεί γενικότερα σε συσκευές που υποστηρίζονται από το λειτουργικό Android. Συνεπώς, η εφαρμογή δοκιμάστηκε σε εικονικές συσκευές αλλά και σε πραγματικά τηλέφωνα όπως περιγράψαμε και στην εισαγωγή και λειτούργησε επιτυχώς.

Επιπλέον, η εφαρμογή που αναπτύχθηκε θα μπορούσε με αρκετή επιτυχία να υπάρξει στο μάρκετ του Android ως μία εφαρμογή διασκέδασης.

Με την προσθήκη δε κάποιων επιπλέον χαρακτηριστικών στην εφαρμογή θα μπορούσε εύκολα να εξελιχθεί σε ένα ολοκληρωμένο παιχνίδι πάζλ και συναγωνισμού. Για παράδειγμα με την προσθήκη του χαρακτηριστικού της συγκέντρωσης βαθμών μπορούμε να επιτύχουμε κάτι αντίστοιχο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- [1] Πληροφορίες για την Google και την Open Handset Alliance
<http://www.google.com/intl/en/corporate/>
<http://www.openhandsetalliance.com/>
- [2] Πληροφορίες για το λειτουργικό σύστημα Android
<http://www.android.com/>
- [3] Στην παρακάτω πηγή βρίσκεται το logo του Android
http://en.wikipedia.org/wiki/File:Android_logo.svg
- [4] Πληροφορίες για τα χαρακτηριστικά και την αρχιτεκτονική του Android μπορούμε να βρούμε και στις παρακάτω πηγές
<http://developer.android.com/guide/basics/what-is-android.html>
Από το βιβλίο Hello, Android Introducing Google's Mobile Development Platform, 3rd Edition του Ed Burnette
- [5] Στις παρακάτω πηγές παρέχονται πληροφορίες για τις διάφορες εκδόσεις και των API της πλατφόρμας του Android
<http://developer.android.com/resources/dashboard/platform-versions.html>
- [6] Περισσότερες πληροφορίες για την ανάπτυξη λογισμικού για την πλατφόρμα Symbian βρίσκονται στο παρακάτω σύνδεσμο
<http://www.medialab.sonera.fi/workspace/SymbianAppDevelopmentWhitePa.pdf>
- [7] Ένα καλογραμμένο και συνοπτικό κείμενο πάνω στην ασφάλεια των λειτουργικών συστημάτων από τον Andrew S. Tanenbaum, Jorrit N. Herde και τον Herbert Bos με τίτλο
“Can We Make Operating Systems Reliable and Secure?” βρίσκεται στον παρακάτω σύνδεσμο
<http://www.cs.vu.nl/~ast/publications/computer-2006a.pdf>
- [8] Περισσότερες πληροφορίες για την υπέρυθη ακτινοβολία μπορούμε να πάρουμε από την Wikipedia παρακάτω
<http://en.wikipedia.org/wiki/Infrared>
- [9] Επίσημες ιστοσελίδες και πληροφορίες για την ασύρματη σύνδεση Bluetooth, Wi-Fi και άλλων πρωτοκόλλων βρίσκονται στις πηγές
<http://www.bluetooth.com> & <http://www.ieee.org/>

- [10] Πληροφορίες για το λειτουργικό σύστημα της Microsoft για κινητές συσκευές υπάρχουν στον σύνδεσμο
<http://msdn.microsoft.com/en-us/library/ms905511.aspx>
- [11] Στην παρακάτω πηγή περιγράφεται το μοντέλο ασφαλείας του λειτουργικού συστήματος Symbian
http://library.forum.nokia.com/topic/S60_5th_Edition_Cpp_Developers_Library/GUID-35228542-8C95-4849-A73F-2B4F082F0C44/sdk/doc_source/guide/platsecsdk/index.html
- [12] Στην παρακάτω πηγή περιγράφεται το μοντέλο ασφαλείας του λειτουργικού συστήματος Windows
<http://msdn.microsoft.com/en-us/library/cc182298.aspx>
- [13] Στις παρακάτω πηγές περιγράφεται το μοντέλο ασφαλείας του λειτουργικού συστήματος Android
<http://developer.android.com/guide/topics/security/security.html>
Στο βιβλίο Professional Android 2 Application Development του Reto Meier
- [14] Ο Τρόπος εγκατάστασης του Android Sdk βρίσκεται στην παρακάτω πηγή <http://developer.android.com/sdk/installing.html>
- [15] Ο σύνδεσμος για να κατεβάσουμε το IDE Eclipse
<http://www.eclipse.org/downloads/>
- [16] Πληροφορίες για τον τρόπο εγκατάστασης του Android plug in στο Eclipse μπορούμε να πάρουμε από τις παρακάτω πηγές
<http://developer.android.com/sdk/eclipse-adt.html#installing>
- [17] Η ανάπτυξη του πρώτου παραδείγματος από την Google για το Android βρίσκεται στην διεύθυνση
<http://developer.android.com/resources/tutorials/hello-world.html>
- [18] Περισσότερες πληροφορίες για την κλάση R υπάρχουν στην παρακάτω πηγή
<http://d.android.com/reference/android/R.html>
- [19] Περισσότερες πληροφορίες για το αρχείο AndroidManifest.xml και ανάλυση όλων των xml ετικετών και των γνωρισμάτων τους
<http://developer.android.com/guide/topics/manifest/manifest-intro.html>
- [20] Περιγραφή της ασφάλειας και αναλυτική επεξήγηση όλων των δικαιωμάτων που μπορούμε να χρησιμοποιήσουμε στην εφαρμογή μας

<http://developer.android.com/guide/topics/security/security.html>

[21] Μια αναλυτική περιγραφή των μεθόδων και του κύκλου ζωής του Activity βρίσκεται στα παρακάτω

<http://developer.android.com/reference/android/app/Activity.html>

Hello Android και The Android Developer's Cookbook Building Applications with the Android SDK του James Steele και Nelson To

[22] Πληροφορίες και ανάλυση των περισσότερων View που συνήθως χρειαζόμαστε

<http://developer.android.com/guide/tutorials/views/index.html>

[23] Στα παρακάτω υπάρχει αναλυτική περιγραφή των τύπων και των χαρακτηριστικών των Layouts

<http://developer.android.com/guide/topics/ui/layout-objects.html>

Unlocking Android A DEVELOPER'S GUIDE του W. FRANK ABLESON, CHARLIE COLLINS και ROBI SEN

[24] Πληροφορίες για την κατασκευή προσαρμοσμένης View μπορούμε να βρούμε στην διεύθυνση

<http://developer.android.com/guide/topics/ui/custom-components.html>

[25] Μια παρουσίαση του τρόπου κατασκευής δισδιάστατων και τρισδιάστατων γραφικών, κινούμενων εικόνων και του OpenGL ES στο Android βρίσκουμε στα παρακάτω

<http://developer.android.com/guide/topics/graphics/index.html>

Hello Android

[26] Το παιχνίδι Snake που δημιούργησε η Google και ο πηγαίος κώδικας του <http://developer.android.com/resources/samples/Snake/index.html>

[27] Στις παρακάτω πηγές έχουμε την επεξήγηση της πυκνότητας, των dpi, της ανάλυσης της οθόνης κα. Ακόμα αναλύονται οι καλύτερες πρακτικές που μπορούμε να ακολουθήσουμε για την υποστήριξη πολλαπλών οθονών

http://developer.android.com/guide/practices/screens_support.html

The Android Developer's Cookbook Building Applications with the Android SDK και στο Hello Android

[28] Αναλύεται η xml ετικέτα <supports-screens> και τα γνωρίσματα που μπορεί να περιέχει

<http://developer.android.com/guide/topics/manifest/supports-screens-element.html>

[29] Σε αυτήν την πηγή αναλύεται η ετικέτα <uses-sdk>, τα γνωρίσματα της και οι τιμές που παίρνουν

<http://developer.android.com/guide/topics/manifest/uses-sdk-element.html>

[30] Ο πίνακας που δημιουργήσαμε βασίζεται σε πληροφορίες που πήραμε από τον επίσημο οδηγό του Android και μπορούμε να τις βρούμε στην παρακάτω πηγή

http://developer.android.com/guide/practices/screens_support.html

[31] Στις παρακάτω πηγές γίνεται ανάλυση των εργαλείων για τον εντοπισμό σφαλμάτων που μας παρέχει το SDK του Android αλλά και το ίδιο το Eclipse

<http://developer.android.com/guide/developing/debug-tasks.html>

[32] Μια αναλυτική περιγραφή του εργαλείου Android Debug Bridge (adb) μπορούμε να την βρούμε στην παρακάτω πηγή

<http://developer.android.com/guide/developing/tools/adb.html>

[33] Σε αυτήν την πηγή περιγράφεται λεπτομερώς η βιβλιοθήκη Log

<http://developer.android.com/reference/android/util/Log.html>

[34] Σε αυτές τις πηγές παρέχονται οδηγίες για το τι πρέπει η δεν πρέπει να κάνουμε ώστε να πετύχουμε την καλύτερη δυνατή υποστήριξη σε πολλαπλές οθόνες

http://developer.android.com/guide/practices/screens_support.html

&Hello android

[35] Σε αυτές τις πηγές περιέχονται προτινόμενες εικονικές συσκευές / εξομοιωτές που μπορούμε να δημιουργίσουμε ώστε να δοκιμάζουμε τις εφαρμογές μας

http://developer.android.com/guide/practices/screens_support.html

Hello Android και Professional Android 2 Application Development

