Πτυχιακή Εργασία:

Εφαρμογές για κινητά τηλέφωνα (SmartPhones) και Tablets για την πλοήγηση και διαχείριση ηλεκτρονικών καταστημάτων.

Ομάδα εργασίας : Ζήκος Δήμος – Κοροσίδης Αντώνιος

Ημερομηνία Παράδοσης:

Πρόλογος

Λαμβάνοντας υπόψη ότι το διαδίκτυο και το εμπόριο είναι πλέον άρρηκτα συνδεδεμένα, είναι σχεδόν απαραίτητο για μία οποιαδήποτε επιχείρηση να διατηρεί το δικό της ηλεκτρονικό κατάστημα, προκειμένου να αυξάνει τις πωλήσεις της, διευκολύνει τους πελάτες και προβάλλει ένα πιο εκσυγχρονισμένο πρόσωπο στην αγορά. Η παρούσα πτυχιακή εργασία πραγματοποιήθηκε στο Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλονίκης στο Τμήμα Πληροφορικής. Σκοπός της είναι η εκπόνηση μίας εφαρμογής για φορητές συσκευές που χρησιμοποιούν το λειτουργικό σύστημα Android, η οποία μπορεί να χρησιμοποιείται για την πλοήγηση και την διαχείριση ενός ηλεκτρονικού καταστήματος. Με δεδομένη την ραγδαία εξέλιξη των φορητών συσκευών και τη συνεχώς αυξανόμενη χρήση τους, αντιλαμβάνεται κανείς πως η εφαρμογή αυτή μπορεί να διευκολύνει τόσο τον ιδιοκτήτη του καταστήματος όσο και τους πελάτες, καθώς και να βελτιώσει την επιχείρηση, αυξάνοντας παράλληλα τα έσοδα της.

Για να φέρουμε εις πέρας την εργασία, χρειάστηκε να εφαρμόσουμε όλες τις γνώσεις μας πάνω στον προγραμματισμό με χρήση της γλώσσας προγραμματισμού Java, τα αρχεία XML, την SQL, τις διαδικτυακές υπηρεσίες και τον σχεδιασμό γραφικών. Χρειάστηκε φυσικά να αποκτήσουμε νέες γνώσεις όσον αφορά τη χρήση όλων των παραπάνω για τον προγραμματισμό μιας εφαρμογής Android. Επίσης, δημιουργήσαμε με χρήση του λογισμικού VirtueMart, ένα ηλεκτρονικό κατάστημα με το οποίο επικοινωνεί η εφαρμογή και αντλεί τα απαραίτητα δεδομένα.

Με εκτίμηση,

Ζήκος Δήμος – Κοροσίδης Αντώνιος

Περίληψη

Η παρούσα πτυχιακή εργασία εστιάζει στην μελέτη του λειτουργικού συστήματος Android, για την ανάπτυξη εφαρμογών σε φορητές συσκευές . Στα πλαίσια της εργασίας αναπτύχθηκε μία εφαρμογή για την πλοήγηση πελατών και επισκεπτών ενός ηλεκτρονικού καταστήματος.

Στην εισαγωγή δίνονται μερικές γενικές πληροφορίες σχετικά με το Android και τις εφαρμογές που βασίζονται σε αυτό. Στη συνέχεια στο πρώτο κεφάλαιο γίνεται μία ιστορική αναδρομή του λειτουργικού συστήματος αυτού και επεξηγούνται τα βασικά χαρακτηριστικά του.

Στο δεύτερο κεφάλαιο γίνεται μία αναφορά στην διαδικασία του Rooting, δηλαδή το πώς να αποκτήσει κανείς πλήρη πρόσβαση στο λειτουργικό σύστημα καθώς και οι λόγοι και τα πλεονεκτήματα της διαδικασίας αυτής. Στο τρίτο κεφάλαιο αναφέρεται συνοπτικά η τεχνολογία επικοινωνίας κοντινού πεδίου NFC κάτι που συναντάται στο Android.

Στο τέταρτο κεφάλαιο γίνεται μία εκτεταμένη ανάλυση των χαρακτηριστικών και των συστατικών του Android καθώς και όλων των τεχνολογιών που ενσωματώνει όπως είναι οι βάσεις δεδομένων. Επίσης αναφέρονται και τα στοιχεία που χρειάζεται να γνωρίζει κανείς ώστε να μπορεί να αναπτύσσει εφαρμογές για την πλατφόρμα αυτή.

Στο πέμπτο κεφάλαιο αναλύεται η βάση δεδομένων SQLite την οποία ενσωματώνει το Android και χρησιμοποιείται σε όλες τις εφαρμογές που υπάρχει η ανάγκη χρήσης μιας βάσης δεδομένων ενώ στο έκτο κεφάλαιο περιγράφονται τα web services που βοηθούν στην λήψη και την διαχείριση δεδομένων για την εφαρμογή από το διαδίκτυο.

Στο έβδομο κεφάλαιο περιγράφεται το περιβάλλον ανάπτυξης εφαρμογών καθώς και η λήψη και η εγκατάσταση αυτού. Γίνεται επίσης μια αρχική αναφορά στον τρόπο δημιουργίας εφαρμογών καθώς και της αποσφαλμάτωσης τους, ενώ στο όγδοο κεφάλαιο υπάρχει ένα υπόδειγμα δημιουργίας μιας εφαρμογής με ενσωμάτωση των χαρτών της Google.

Μετά από μια αναφορά στα χαρακτηριστικά και τις δυνατότητες του λογισμικού VirtueMart στο ένατο κεφάλαιο, ακολουθεί στο δέκατο κεφάλαιο ένα πλήρες εγχειρίδιο χρήστη της εφαρμογής που αναπτύχθηκε.

Τέλος στα κεφάλαια έντεκα και δώδεκα αντίστοιχα, γίνεται πλήρης ανάλυση του κώδικα Java που αναπτύχθηκε για την υλοποίηση των λειτουργιών της εφαρμογής και των αρχείων XML που χρησιμοποιούνται για το γραφικό περιβάλλον χρήστη.

Μια επισήμανση των σημαντικότερων σημείων της εργασίας αυτής γίνεται στο τμήμα των συμπερασμάτων ως επίλογος.

Abstract

The current thesis focuses on a research aiming to develop mobile devices applications using the Android operating system. During the research, an application was developed considering the browsing of an electronic shop by customers and visitors.

In the introduction there is some general information about Android and the applications that are developed for it. In the first chapter the historical background of this operating system is mentioned. There is also an explanation of its basic features.

The second chapter refers to the process of Rooting, so that the user can have full access to the operating system. The reasons and the advantages of this process are also mentioned in this chapter. The next chapter gives a quick analysis of the near field communication technology (NFC) which is available in Android.

The fourth chapter contains a wide analysis of the features and the structure of Android and the number of technologies that it supports, such as databases. In this chapter there are also some basics that someone needs to know when developing android applications.

The fifth chapter gives an analysis of the SQLite database which is embedded in Android and is used in all applications that require the use of a database, while in chapter six there is a description for web services which are used in order to retrieve data from the internet to be used in the application.

The next chapter states information about the software development kit and the environment for developing applications. Moreover, there is also an initial report of how to create an application and to debug it. Next, in chapter eight there is an example for creating a Google Maps application.

After referencing some of the features of VirtueMart software in chapter nine, a full user manual of the developed application follows in chapter ten.

At the end of the thesis there are chapters eleven and twelve, where a full analysis of the developed Java code used for the functions of the application is made, as well as of the XML files which are used for the graphic user interface.

The conclusion contains a reference to the most important points of the present thesis.

Εισαγωγή

Το Android είναι μια σχετικά καινούργια τεχνολογία. Η Google αγόρασε την αρχική εταιρία ανάπτυξης λογισμικού Android, την Android Inc. το 2005. Το 2007 ανακοινώθηκε η πρώτη διανομή του Android με την ίδρυση της Open Handset Alliance, μια σύμπραξη 86 εταιριών λογισμικού, κατασκευής hardware και τηλεπικοινωνιών αφοσιωμένες στην ανάπτυξη καινούργιων τεχνολογιών για κινητές συσκευές. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

Το Android έχει μια πολύ μεγάλη κοινότητα προγραμματιστών που γράφουν εφαρμογές επεκτείνοντας τη λειτουργικότητα των συσκευών. Η κοινότητα αυτή είναι ένας από τους κύριους λόγους για την αυξημένη δημοτικότητα της Android πλατφόρμας. Ο χρήστης μπορεί να «κατεβάσει» εφαρμογές τρίτων από διαδικτυακά καταστήματα όπως το Google Play (προηγουμένως γνωστό ως Google Market – το κατάστημα της Google). Ενδεικτικά τον Οκτώβριο του 2011 υπήρχαν περισσότερες από 500.000 εφαρμογές διαθέσιμες στο Android Market ενώ ο αριθμός των «κατεβασμάτων» ξεπερνάει τα 10 δισεκατομμύρια.

Περιεχόμενα

Πρόλογος **2**
Περίληψη **3**
Abstract **4**
Εισαγωγή **5**
1. Ιστορική Αναδρομή **10**
1.1 Ιστορικό εκδόσεων 10
1.2 Google Now 17
1.3 Υποστήριξη Adobe Flash 19
2. Rooting **20**
2.1 Γιατί να γίνει κάποιος Root χρήστης 20
3. Επικοινωνία κοντινού πεδίου (NFC Beam) **22**
3.1 Πώς λειτουργεί 22
3.2 NFC tags 23
4. Τι είναι το Android **25**
4.1 Βασικά χαρακτηριστικά 25
4.2 Από τι αποτελείται το Android 27
4.3 Βασικά συστατικά εφαρμογών του Android 29
4.4 Activities (Δραστηριότητες) 32
4.5 Services (Υπηρεσίες) 35
4.6 Διαχείριση μνήμης 37
4.7 Γραφικό περιβάλλον χρήστη (Graphical User Interface-GUI) 39
4.8 Views (πρότυπες προβολές) 44
4.9 Layouts (διατάξεις) 44
4.10 Υποστήριξη πολλαπλών οθονών 45
5. Βάσεις Δεδομένων **51**
5.1 SQLite 51
6. Web Services **55**
6.1 Ορισμός 55
6.2 Γενικά 55
6.3 Αρχιτεκτονική 55
6.4 Τεχνολογίες των Web Services 56
6.5 XML (eXtensible Markup Language) 56
6.6 WSDL(Web Services Description Language) 58
6.7 UDDI (Universal Description, Discovery and Integration) 58

6.8 SOAP(Simple Object Access Protocol) 59
6.9 Παράδειγμα 60
7. Περιβάλλον ανάπτυξης εφαρμογών 62
7.1 Android SDK 62
7.2 Android NDK (Native Development Kit) 69
7.3 App Inventor for Android 71
7.4 Εντοπισμός σφαλμάτων (Debugging) 74
8. Google Maps 80
8.1 Απόκτηση του API key 80
8.2 Δημιουργία Google Maps Project 81
9. VirtueMart 88
10. Περιγραφή εφαρμογής OnShop 91
11. Ανάλυση Κώδικα Java 99
11.1 About.java 100
11.2 AddItemizedOverlay.java 101
11.3 Cart.java 102
11.4 Categories.java 105
11.5 Contact.java 106
11.6 DashboardLayout.java 108
11.7 Favorites.java 108
11.8 FileCache.java 111
11.9 FirstSync.java 112
11.10 ImageAdapter.java 115
11.11 ImageLoader.java 117
11.12 LazyAdapter.java 117
11.13 Maps.java 119
11.14 MemoryCache.java 120
11.15 NewProducts.java 120
11.16 News.java 122
11.17 OnShop.java 123
11.18 Products.java 128
11.19 Sales.java 131
11.20 Search.java 132
11.21 SearchCategories.java και SearchProducts.java 133
11.22 Settings.java 135
11.23 Splash.java 135

11.24 SQLConnect.java 136
11.25 Sync.java 143
11.26 Utils.java 143
11.27 Vieproduct.java 143
11.28 XMLParser.java 146
12. Ανάλυση των αρχείων XML 149
12.1 Αρχεία XML Layout 149
12.2 Αρχεία XML διαφόρων χρήσεων 161
12.3 Αρχείο AndroidManifest.xml 164
13. Συμπεράσματα 166
14. Βιβλιογραφία – Αναφορές 167

Ευρετήριο Εικόνων

Εικόνα I: Δομή του Android 27
Εικόνα II: Κύκλος ζωής υπηρεσίας 29
Εικόνα III: Παροχέας περιεχομένου 30
Εικόνα IV: Κύκλος ζωής Activity 33
Εικόνα V: Απλό παράδειγμα εφαρμογής στον Emulator 42
Εικόνα VI: Περιγραφή ενός Web Service 55
Εικόνα VII: Τεχνολογίες των Web Services 56
Εικόνα VIII: Δομή μηνύματος SOAP 59
Εικόνα IX: SDK Manger 63
Εικόνα X: Λήψη SDK 65
Εικόνα XI: Λήψη ADT 66
Εικόνα XII: Δημιουργία AVD 67
Εικόνα XIII: Emulator (1) 67
Εικόνα XIV: Emulator (2) 67
Εικόνα XV: Δημιουργία νέου project (1) 68
Εικόνα XVI: Δημιουργία νέου project (2) 68
Εικόνα XVII: App Inventor 73
Εικόνα XVIII: File Explorer 75
Εικόνα XIX: DDMS 76
Εικόνα XX: LogCat 77
Εικόνα XXI: Hierarchy Viewer 79
Εικόνα XXII: Εντολή για το MD5 Fingerprint 81

Εικόνα XXIII: Νέο Maps project 82
Εικόνα XXIV: Νέο Maps project (2) 82
Εικόνα XXV: Νέο Maps project (3) 83
Εικόνα XXVI: Google Maps στον Emulator (1) 85
Εικόνα XXVII: Google Maps στον Emulator (2) 87
Εικόνα XXVIII: Αρχικό μενού 91
Εικόνα XXIX: Αναδυόμενο μενού 91
Εικόνα XXX: Πρώτη εκτέλεση της εφαρμογής 92
Εικόνα XXXI: Μήνυμα επιτυχίας 92
Εικόνα XXXII: Κατηγορίες προϊόντων 93
Εικόνα XXXIII: Προϊόντα 93
Εικόνα XXXIV: Προσθήκη στο καλάθι 94
Εικόνα XXXV: Προβολη προϊόντος 94
Εικόνα XXXVI: Τρόποι επικοινωνίας 95
Εικόνα XXXVII: Newsletter 95
Εικόνα XXXVIII: Καλάθι αγορών 96
Εικόνα XXXIX: Ρυθμίσεις Χρήστη 96
Εικόνα XL: Αναζήτηση 97
Εικόνα XLI: Αυτόματη συμπλήρωση 97
Εικόνα XLII: Καταστήματα 98
Εικόνα XLIII: Συγχρονισμός 98
Εικόνα XLIV: Αρχεία Java 99
Εικόνα XLV: Αρχεία XML 99

Ευρετήριο Πινάκων

Πίνακας I: Εκδόσεις Android 10
Πίνακας II: Τύποι NFC 23
Πίνακας III: Κατηγοριοποίηση αναλύσεων 46

1. Ιστορική Αναδρομή

1.1 Ιστορικό εκδόσεων

Οι εκδόσεις του λειτουργικού συστήματος Android ξεκίνησαν με την κυκλοφορία της πρώτης beta έκδοσης το Νοέμβριο του 2007. Η πρώτη εμπορική έκδοση (Android 1.0) κυκλοφόρησε τον Σεπτέμβριο του 2008. Το λειτουργικό σύστημα του Android που δημιουργήθηκε από την Google και την Handset Alliance έχει δει βασικές ενημερώσεις από τη στιγμή της πρώτης κυκλοφορίας του. Οι ενημερώσεις αυτές διορθώνουν τυχόν προηγούμενα προβλήματα και προσθέτουν νέα χαρακτηριστικά. Από τον Απρίλιο του 2009 κάθε καινούργια έκδοση είχε μια κωδική ονομασία βασισμένη σε γλυκό – επιδόρπιο με αλφαβητική σειρά: Cupcake, Donut, Éclair, Froyo (Frozen yogurt), Gingerbread, Honeycomb, Ice Cream Sandwich και προσφάτως Jelly Bean. Οι δύο πρώτες εκδόσεις ονομάστηκαν μεταγενέστερα Astro και Bender αντίστοιχα αλλά δεν μπορούν να χρησιμοποιηθούν επίσημα για λόγους πνευματικής ιδιοκτησίας. Η τελευταία ενημέρωση του λειτουργικού ήταν η Jelly Bean v4.1 η οποία κυκλοφόρησε τον Ιούνιο του 2012.

ΕΚΔΟΣΗ	ΟΝΟΜΑΣΙΑ
1.5	Cupcake
1.6	Donut
2.0/2.1	Éclair
2.2	Froyo
2.3	Gingerbread
3.1	Honeycomb
4.0	Ice Cream Sandwich
4.1	Jelly Bean

Πίνακας Ι : Εκδόσεις Android

Βασικές αναβαθμίσεις

- Η **Éclair** είναι η έκδοση 2.0/2.1, έφερε αλλαγές στο περιβάλλον και έφερε υποστήριξη HTML5 στον περιηγητή (browser).

- Η **FroYo (Frozen Yogurt)** είναι η έκδοση 2.2, έδωσε μεγαλύτερη ταχύτητα και έφερε υποστήριξη flash μαζί με δυνατότητα Wi-Fi hotspot.

- Η **Gingerbread** είναι η έκδοση 2.3 και βελτίωσε το περιβάλλον όπως και δυνατότητες για πιο "βαριές" εφαρμογές, επιπλέον πρόσθεσε υποστήριξη NFC (Near Field Communication).

- Η **Honeycomb** είναι η έκδοση 3.0/3.1 αποκλειστικά για ταμπλέτες, έφερε αλλαγές κυρίως στο γραφικό περιβάλλον και πρόσθεσε υποστήριξη πολλαπλών πυρήνων μαζί με βελτιωμένα γραφικά.

- Η **Ice Cream Sandwich** δηλαδή η έκδοση 4.0 έχει σκοπό να "ενώσει" τις εκδόσεις για ταμπλέτες και κινητά και να προσθέσει υποστήριξη για την Google TV.

- Η **Jelly Bean** είναι η τελευταία έκδοση του Android (4.1) και εκτός από ταχύτατη πλοήγηση υπόσχεται και μεγάλες αλλαγές στον τομέα της τεχνητής νοημοσύνης.

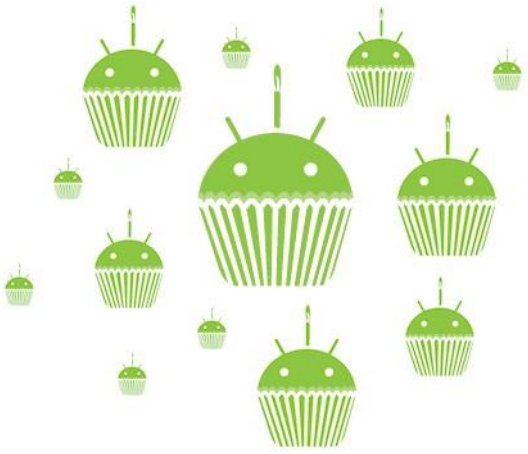
Η πρώτη έκδοση του Android κυκλοφόρησε χωρίς πρακτικά να υπάρχει κινητή συσκευή που να το χρησιμοποιεί. Ο σκοπός ήταν να αποδειχθεί ότι δεν χρειάζεται συσκευή για να ξεκινήσει κάποιος την ανάπτυξη εφαρμογών. Υπάρχουν μερικοί περιορισμοί όσον αφορά το υλικό (πχ αισθητήρας βαρύτητας) αλλά κατά το μεγαλύτερο κομμάτι περιλαμβάνει ό,τι χρειάζεται ο προγραμματιστής για να αναπτύξει εφαρμογές σε αυτό.

Αναλυτικά όλες οι εκδόσεις

Η πρώτη έκδοση του Android SDK που εμφανίστηκε τον Νοέμβριο του 2007, χαρακτηρίστηκε από τους κατασκευαστές του σαν μια πρώτη ματιά στο SDK του Android, κάτι το οποίο πολλοί παράβλεψαν και βιάστηκαν να κατακρίνουν το Android σαν ένα προβληματικό σύστημα. Στην ουσία όμως το Android δεν παρουσίαζε προβλήματα τα οποία δεν παρουσιάζει οποιοδήποτε σύστημα σε τέτοια πρώιμη φάση. Έτσι το Σεπτέμβριο του 2008, η T-Mobile ανακοινώνει την διαθεσιμότητα του T-Mobile G1, του πρώτου έξυπνου τηλεφώνου (smartphone), βασισμένο στην πλατφόρμα του Android. Λίγες μέρες αργότερα (Οκτώβριο 2008), η Google ανακοινώνει την απελευθέρωση του SDK Release Candidate 1.0. Ακολούθησε τον Φεβρουάριο του 2009 η έκδοση 1.1 σαν μια ανανεωμένη έκδοση του 1.0. Μέχρι τότε το Android δεν υποστήριζε ακόμη την χρήση κουμπιών αφής, παρά μόνο την χρήση των κλασσικών «σκληρών» κουμπιών της συσκευής.

Cupcake (1.5)

Το 'Cupcake' (Μάιος 2009) εισάγει κάποια καινούργια χαρακτηριστικά και ανανεώσεις στην διεπιφάνεια χρήστη (User Interface):



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

Donut (1.6)

Η έκδοση 'Donut' (Σεπτέμβριος 2009) εισάγει καινούργια χαρακτηριστικά όπως:

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

Éclair (2.0/2.1)

Το 'Éclair' (Νοέμβριος 2009 το 2.0 και Ιανουάριος 2010 το 2.1) έρχεται με τις ακόλουθες αλλαγές:

- Βέλτιστη ταχύτητα υλικού
- Υποστήριξη για περισσότερες οθόνες και αναλύσεις
- Βελτιωμένη διεπιφάνεια χρήστη με Live Wallpapers (κινούμενα backgrounds νέου σχεδιασμού που αλληλεπιδρούν με τις κινήσεις δαχτύλου του χρήστη)
- Καινούργια διεπιφάνεια χρήσης για την μηχανή αναζήτησης και υποστήριξη του προτύπου HTML5
- Καινούργιες λίστες επαφών
- Καλύτερος λόγος άσπρου – μαύρου για φόντα
- Βελτιωμένοι χάρτες Google (Google maps) 3.1.2
- Υποστήριξη Microsoft Exchange
- Ενσωματωμένη υποστήριξη flash για την Camera
- Ψηφιακή μεγέθυνση (zoom)
- Κλάση MotionEvent βελτιωμένη ώστε οι κατασκευαστές να μπορούν να παρακολουθούν αποτελεσματικότερα τα γεγονότα πολλαπλής αφής
- Ανανεωμένο εικονικό πληκτρολόγιο
- Bluetooth 2.1

Froyo (2.2)

Ακολουθεί το 'Froyo' (Μάιος 2010) με τις παρακάτω αναβαθμίσεις:

- Βελτιστοποιήσεις στην ταχύτητα γενικά του λειτουργικού συστήματος, στην μνήμη και στην απόδοση
- Ενσωμάτωση στην μηχανή αναζήτησης, της μηχανής JavaScript του Chrome V8
- Αυξημένη υποστήριξη Microsoft Exchange (σε πολιτικές ασφαλείας, συγχρονισμού ημερολογίου, auto-discovery, GAL look-up, remote wipe)
- Βελτιωμένος προωθητής εφαρμογής (application launcher), με συντομεύσεις προς τις εφαρμογές τηλεφώνου και εφαρμογές της Μηχανής Αναζήτησης
- Σύνδεση USB και λειτουργία δυναμικής ζώνης (hotspot) Wi-Fi
- Ανανεωμένη εφαρμογή του Market με αυτόματη ανανέωση
- Επιλογή για απαγόρευση πρόσβασης δεδομένων μέσω ενός δικτύου κινητής τηλεφωνίας
- Γρήγορη εναλλαγή ανάμεσα σε πολλαπλές γλώσσες του πληκτρολογίου και των λεξικών τους
- Φωνητική κλήση και διαμοιρασμός επαφών με Bluetooth
- Υποστήριξη για αριθμητικούς και αλφαριθμητικούς κωδικούς
- Η μηχανή αναζήτησης μπορεί να αποτυπώσει κινούμενα GIFs
- Υποστήριξη για πεδία μεταφόρτωσης αρχείων στην μηχανή αναζήτησης
- Υποστήριξη για εγκατάσταση εφαρμογών στην επεκτάσιμη μνήμη
- Υποστήριξη Adobe Flash 10.1

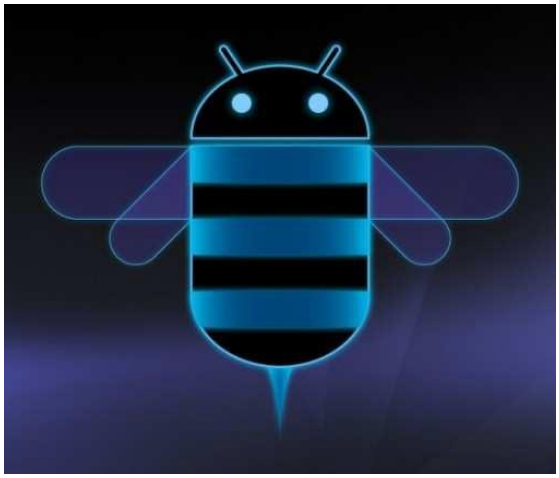
Gingerbread (2.3)

Η έκδοση 'Gingerbread' (Δεκέμβριος 2010) ανάμεσα σε άλλες αλλαγές περιλαμβάνει:

- Βελτιωμένο UI (User Interface) για απλότητα και ταχύτητα
- Πιο γρήγορη, πιο διαισθητική εισαγωγή κειμένου
- Επιλογή λέξεων και αντιγραφή/επικόλληση με ένα άγγιγμα
- Βελτιωμένη ενεργειακή διαχείριση, υποστήριξη NFC (Near Field Communication)
- Υποστήριξη video κλήσης
- Υποστήριξη του πρωτόκολλου WebM για αναπαραγωγή video

Honeycomb (3.1)

Το 'Honeycomb' (Φεβρουάριος 2011), η έκδοση αποκλειστικά για tablets περιλαμβάνει τα παρακάτω καινοτόμα χαρακτηριστικά:



- Υποστήριξη διπύρηνων και τετραπύρηνων επεξεργαστών
- Βελτιωμένη υποστήριξη των ταμπλετών
- Ανάπτυξη λογισμικού (scripting) για 3D, σε γλώσσα η οποία καλείται "Renderscript"
- Video chat μέσω Google Talk
- Google eBooks
- Ιδιωτική περιήγηση

Ice Cream Sandwich (4.0)

Το 'Ice Cream Sandwich' (Οκτώβριος 2011), το ενιαίο πλέον λειτουργικό σύστημα για όλες τις συσκευές, έχει τα ακόλουθα χαρακτηριστικά:

- Χρησιμοποιήσιμο σε tablets και κινητά δηλαδή σε τεράστια γκάμα μεγεθών οθόνης και διαφορετικών αναλύσεων
- Τρισδιάστατο (3D) User Interface
- Δυνατότητα αναγνώρισης προσώπου και κινήσεων ματιών ή/και στόματος
- Πλουσιότερα και ομορφότερα widgets
- Ενισχυμένο multitasking
- Επιτάχυνση υλικού (hardware acceleration) στο UI
- Προσαρμόσιμος launcher
- Δυνατότητα λήψης στιγμιότυπου οθόνης
- Wi-Fi Direct
- Ανανεωμένος φυλλομετρητής ιστού (browser)

Jelly Bean (4.1)

Το 'Jelly Bean' (Ιούλιος 2012) είναι η τελευταία έκδοση και έχει τις παρακάτω αναβαθμίσεις:

- Βελτιωμένη φωνητική αναζήτηση
- Google Now
- Φωτογραφίες επαφών υψηλής ανάλυσης
- Ειδοποιήσεις με δυνατότητες αλληλεπίδρασης
- Βελτιωμένη τεχνητή νοημοσύνη ώστε να «προβλέπει» τις καθημερινές ανάγκες του χρήστη
- Γρηγορότερος browser

Οι σημαντικότερες αλλαγές στο Android 4.1 γνωστό και ως Jelly Bean αφορούν στην περιοχή των Ειδοποιήσεων (Notifications). Η Google επιτρέπει στους προγραμματιστές να προσθέσουν στις εφαρμογές τους λειτουργίες που θα μπορούν να εκτελούνται από τον χρήστη απευθείας από την περιοχή των Ειδοποιήσεων. Έτσι, θα μπορεί κανείς όχι μόνο να δει μια ειδοποίηση αλλά και να πράξει ανάλογα.

Αναπτύσσοντας προς τα κάτω τα Notifications, δηλαδή σύροντας ένα δάκτυλο από το πάνω μέρος της οθόνης προς τα κάτω, ανοίγει το λεγόμενο Notification shade. «Τσιμπώντας» με δύο δάκτυλα μια ειδοποίηση, ο χρήστης βλέπει τις πιθανές σχετικές ενέργειες που μπορεί να εκτελέσει.

Στις αρχικές οθόνες μιας συσκευής με Jelly Bean υπάρχουν νέα widget, που δείχνουν τις εφαρμογές, την μουσική, τα βίντεο, τα παιχνίδια ή τα βιβλία που χρησιμοποιήσαμε τελευταία. Άλλα widget μπορούν επίσης να προτείνουν περιεχόμενο από το Google Play. Το μέγεθός τους μπορεί να αλλάξει (με παρατεταμένο άγγιγμα).

Η αναδιάταξη της αρχικής οθόνης διευκολύνεται, αφού καθώς μετακινεί ο χρήστης ένα widget ή ένα εικονίδιο, «σπρώχνει» άλλα παρακείμενα.

Το πληκτρολόγιο είναι επίσης ανανεωμένο, με σημαντικότερη βελτίωση την πρόβλεψη της επόμενης λέξης που εκτιμάται ότι θα γράψει ο χρήστης, με βάση το ιστορικό της συγγραφής του (βελτιώνεται με το χρόνο και απομένει να δούμε ποιες γλώσσες θα υποστηρίζονται).

Εκτός από την δυνατότητα ξεκλειδώματος της συσκευής με Jelly Bean με την εμφάνιση του προσώπου μας, στο Jelly Bean μπορούμε να ορίσουμε να του κλείνουμε και το μάτι, αυξάνοντας την ασφάλεια (Blink Face Unlock).

1.2 Google Now

Ιδιαίτερο ενδιαφέρον παρουσιάζει η υπηρεσία Google Now ως μέρος του ανανεωμένου Google Search. Υπόσχεται να «φέρνει την σωστή πληροφορία στο σωστό χρόνο», η οποία εξαρτάται από την θέση του χρήστη, και άλλες παραμέτρους που πρέπει να είναι γνωστές στην Google. Έτσι, το πρωί μπορεί να εμφανίζει σε μια κάρτα τον καιρό και την κίνηση στην προγραμματισμένη διαδρομή του χρήστη με βάση το ημερολόγιό του ή να εμφανίζει σε μια άλλη το σκορ στο παιχνίδι που έχει δηλώσει ενδιαφέρον. Οι διάφορες κάρτες του Google

Now εμφανίζονται αυτόματα κατά τη διάρκεια της ημέρας τη χρονική στιγμή που θα φανούν χρήσιμες στο χρήστη.

Ο χρήστης φυσικά μπορεί να επιλέξει ποιες κάρτες θέλει να εμφανίζονται ανάλογα με τις ανάγκες του και να αποκρύψει αυτές που θεωρεί ότι δεν χρειάζεται. Τα αποτελέσματα εμφανίζονται εξατομικευμένα από τα ημερολόγια, τις τοποθεσίες και τις αναζητήσεις του χρήστη.

Όταν η οθόνη του Google Now είναι ανοικτή, ο χρήστης μπορεί να αναφωνήσει "Google" και να δώσει όρους αναζήτησης με φωνητική πληκτρολόγηση. Το Google Search και η φωνητική αναζήτηση δεν είναι κάτι νέο στο Android, ο τρόπος χρήσης του όμως δεν ήταν άμεσος. Μέχρι σήμερα, όταν κάναμε μια αναζήτηση, ως αποτέλεσμα παίρναμε την κλασική λίστα με link που βλέπουμε και στον υπολογιστή μας, τώρα όμως η απάντηση έρχεται άμεσα σε καρτέλες οι οποίες αν δε μας ικανοποιούν, πηγαίνουν στην άκρη για να ψάξουμε περαιτέρω.

Αρχικά το Google Now έχει δέκα διαφορετικές κάρτες και σύντομα αναμένονται και άλλες:

- **Πληροφορίες κίνησης:** Σύμφωνα με το ημερολόγιο το Android γνωρίζει την ώρα που θα φύγει ο χρήστης για τη δουλειά του και εμφανίζει την κάρτα με την κίνηση που θα συναντήσει στη διαδρομή του.
- **Μέσα μαζικής μεταφοράς:** Όταν βρεθεί δίπλα σε στάση λεωφορείου, τρένου κλπ, ο χρήστης ειδοποιείται για τα επερχόμενα μέσα μαζικής μεταφοράς και πότε αναμένονται.
- **Επόμενο ραντεβού:** Για ένα σημειωμένο ραντεβού, το Google Now υπολογίζει τη κίνηση και εμφανίζει την κατάλληλη κάρτα αρκετά νωρίτερα ώστε να μην υπάρξει καθυστέρηση.
- **Πτήσεις:** Στην κάρτα των πτήσεων εμφανίζονται πληροφορίες για τις καθυστερήσεις των προσφάτως αναζητηθέντων πτήσεων καθώς και πληροφορίες κίνησης προς το αεροδρόμιο.
- **Αθλήματα:** Το Google Now τον χρήστη με live αποτελέσματα αγώνων και επερχόμενα παιχνίδια στα sport που τον ενδιαφέρουν. Είναι δυνατή και η αγορά εισιτηρίων μέσω αυτής της κάρτας.
- **Τοποθεσίες:** Η κάρτα αυτή ειδοποιεί τον χρήστη για κοντινά μπαρ, εστιατόρια και άλλα POIs (Places Of Interest), στα οποία είναι δυνατή η ανάγνωση κριτικών (μέσω του Google maps), μέχρι και η κράτηση σε κάποιο εστιατόριο.
- **Καιρός:** Το πρωί το Google Now εμφανίζει τον καιρό στην τρέχουσα τοποθεσία.
- **Μετάφραση:** Σε περίπτωση ταξιδιού στο εξωτερικό, η κάρτα μετάφρασης βοηθάει τον χρήστη στην καθημερινότητα του.
- **Συνάλλαγμα:** Στα ταξίδια εμφανίζεται επίσης η κάρτα συναλλάγματος για εύκολη μετατροπή στο τοπικό νόμισμα.
- **Ώρα στο σπίτι:** Όταν ο χρήστης βρίσκεται σε διαφορετική ζώνη ώρας, το Google Now εμφανίζει μία κάρτα με την τοπική ώρα στην πατρίδα του.

1.3 Υποστήριξη Adobe Flash

Όπως αναμενόταν η Adobe σταμάτησε να υποστηρίζει επίσημα το Android από τη νέα έκδοση 4.1 Jelly Bean ενώ απέσυρε και την εφαρμογή της από το Google Play Store στις 15 Αυγούστου 2012. Το Android 4.1 δεν πήρε πιστοποίηση για να τρέχει το Adobe Flash. Για την Adobe οι συσκευές που δεν έχουν προ-εγκατεστημένο τον Flash Player από τον κατασκευαστή δεν θεωρούνται πιστοποιημένες, παρόλο που οι χρήστες μέχρι σήμερα μπορούσαν να κατεβάσουν από το Google Play το λογισμικό. Οι συσκευές που είχαν προ-εγκατεστημένο τον Flash Player θα συνεχίσουν να τον χρησιμοποιούν αλλά μετά από τις 15 Αυγούστου δεν θα δέχονται ενημερώσεις για αυτόν.

Το Adobe Flash το χρησιμοποιούμε κυρίως για δύο λόγους:

- YouTube ή παρόμοιες βίντεο υπηρεσίες
- Περιήγηση σε ιστοσελίδες

Τα δύο παραπάνω περιέχουν Flash περιεχόμενο. Ακόμα και αν δεν υποστηρίζεται από το Android επίσημα, ο κάθε δημιουργός μπορεί να το υποστηρίξει στην εφαρμογή του που έχει κατασκευάσει χωρίς την προϋπόθεση να υπάρχει ξεχωριστή εφαρμογή για αυτό. Πρόσφατο παράδειγμα η Mozilla η οποία στη νέα έκδοση του Firefox για το Android, έρχεται με υποστήριξη Flash σε αντίθεση με τον προ-εγκατεστημένο Chrome του Jelly Bean. Μπορούμε λοιπόν μέσω αυτής να δούμε όλες τις ιστοσελίδες με Flash περιεχόμενο χωρίς πρόβλημα.

Η Google έχει ήδη επιλέξει το ταχύτερο HTML5 ως αντικαταστάτη και έχει ήδη αρχίσει εδώ και καιρό να «μεταφράζει» τα βίντεό της (YouTube) στο νέο format. Το πόσο γρήγορα θα γίνει η μετάβαση είναι στα χέρια των ιδιοκτητών των ιστοσελίδων ώστε να μπορούν οι χρήστες να επισκέπτονται τις ιστοσελίδες τους από οποιαδήποτε συσκευή.

2. Rooting

"Root" είναι ο λογαριασμός που έχει πρόσβαση σε όλα τα αρχεία ενός συστήματος Linux (αφού το Android βασίζεται στο Linux η λογική είναι ίδια). Αυτός ο χρήστης λέγεται και superuser (υπέρ-χρήστης). Η διαδικασία του rooting στη συσκευή προσφέρει πρόσβαση στο λογαριασμό του superuser (su). Είναι το αντίστοιχο ακριβώς του «διαχειριστή» (administrator) σε ένα σύστημα windows. Είναι επίσης κάτι αντίστοιχο με το να αποκτήσεις πρόσβαση στα Symbian αρχεία ενός τηλεφώνου Nokia, ή με το να κάνεις jailbreak ένα iPhone.

Κάτι που μπορεί καμιά φορά να δημιουργήσει σύγχυση, είναι ότι επίσης root (ρίζα) λέγεται το πρώτο επίπεδο των φακέλων σε ένα σύστημα Linux. Δηλαδή ο φάκελος εγκατάστασης μέσα στον οποίο περιέχονται όλοι οι υπόλοιποι φάκελοι του συστήματος. Είναι ο ' / ' φάκελος (αντίστοιχος του C:\ σε ένα pc), και το home directory του superuser λογαριασμού.

2.1 Γιατί να γίνει κάποιος Root χρήστης

Δεν είναι απαραίτητη διαδικασία για κάθε Android χρήστη. Απλά, αν γίνει rooting στη συσκευή, ο χρήστης αποκτάει δικαιώματα πρόσβασης σε όλα τα αρχεία του συστήματος και μπορεί να κάνει αλλαγές που ένας κανονικός χρήστης δε μπορεί. Με αυτόν τον τρόπο μπορεί να διορθώσει κάποιο πρόβλημα και να αλλάξει δικαιώματα σε αρχεία, έχει με λίγα λόγια τον πλήρη έλεγχο του συστήματος. Φυσικά το root έχει τους γνωστούς κινδύνους ενός administrator λογαριασμού, διότι έχοντας τον πλήρη έλεγχο, ένας άπειρος χρήστης μπορεί για παράδειγμα να σβήσει ή να αλλοιώσει σημαντικά αρχεία συστήματος.

Δυνατότητες που προσφέρονται με τα δικαιώματα του root:

- Εγκατάσταση εφαρμογών που χρειάζονται πρόσβαση στα αρχεία του συστήματος (π.χ. εφαρμογές για πλήρες backup).
- Αφαίρεση εφαρμογών του συστήματος (bloatware apps).
- Μεταφορά εφαρμογών στην κάρτα SD.
- Διόρθωση κάποιων προβλημάτων που έχουνε άμεση σχέση με τα αρχεία του συστήματος (εξαρτάται από τη συσκευή).
- Διαφορετικά εικονίδια και τροποποίηση κατά βούληση ολόκληρου του γραφικού περιβάλλοντος.
- Καλύτερη διαχείριση μνήμης.
- Πρόσβαση σε εφαρμογές που είναι φραγμένες γεωγραφικά (φτιαγμένες να δουλεύουν μόνο σε κράτη που προόριζε ο δημιουργός τους).
- Είναι δυνατή η δημιουργία προσαρμοσμένης (custom) ROM και φυσικά το root είναι βασική προϋπόθεση για τη φόρτωση custom ROMs.
- Προχωρημένες ρυθμίσεις (tweaks) εκ βαθέων όπως overclock, undervolt κλπ.

- Αφαίρεση των διαφημίσεων στις δωρεάν εφαρμογές.
- Δυνατότητα χρήσης εφαρμογών (παιχνιδιών κυρίως) που προορίζονται αποκλειστικά συγκεκριμένες συσκευές.
- Ξεκλείδωμα τυχόν περιορισμών από συγκεκριμένο πάροχο κινητής τηλεφωνίας, και πολλά άλλα.

Μειονεκτήματα

Δυστυχώς υπάρχουν και κάποια μειονεκτήματα σε αυτή τη διαδικασία. Υπάρχει μια μικρή πιθανότητα σε ορισμένες συσκευές να γίνει επαναφορά εργοστασιακών ρυθμίσεων κατά τη διάρκεια της διαδικασίας με αποτέλεσμα να χαθούν δεδομένα. Το πιο βασικό μειονέκτημα είναι βέβαια ότι για όσο χρόνο ο χρήστης έχει root δικαιώματα στη συσκευή του δεν ισχύει η εγγύηση. Αυτό συμβαίνει για ευνόητους λόγους από τις εταιρίες γιατί ο χρήστης έχοντας απεριόριστα δικαιώματα στη συσκευή του μπορεί να προκαλέσει άθελά του ζημιά (η οποία φυσικά δεν θα καλυφθεί από την εργοστασιακή εγγύηση). Η αντιστροφή της διαδικασίας (και η επαναφορά της εγγύησης) είναι εφικτή εγκαθιστώντας το επίσημο λογισμικό (firmware) της αντίστοιχης συσκευής, με την προϋπόθεση φυσικά να είναι σε θέση το κινητό να πραγματοποιήσει αυτή τη διαδικασία. Με απλά λόγια ο μεγαλύτερος κίνδυνος είναι να «κολλήσει» η συσκευή υπό καθεστώς root.

Συμπέρασμα

Συμπερασματικά, αν ένας χρήστης θέλει να ανακαλύψει και να εξερευνήσει σε βάθος το λειτουργικό σύστημα Android θα χρειαστεί τις απεριόριστες δυνατότητες που προσφέρει το root. Σε οποιαδήποτε άλλη περίπτωση δεν υπάρχει λόγος να το κάνει, ειδικά αν δεν θέλει να ρισκάρει τυχόν λάθος (λόγω έλλειψης εμπειρίας) που θα αποβεί μοιραίο για τη συσκευή του.

3. Επικοινωνία κοντινού πεδίου (NFC Beam)

Το Android υποστηρίζει την επονομαζόμενη επικοινωνία κοντινού πεδίου (Near Field Communication). Το NFC αποτελεί μια πρότυπη τεχνολογία συνδεσιμότητας και ανήκει στις ασύρματες τεχνολογίες μικρής εμβέλειας με μέγιστη επιτρεπτή απόσταση για τη δημιουργία επικοινωνίας τα 4 εκατοστά. Το NFC επιτρέπει στον χρήστη τη μεταφορά μικρών πακέτων πληροφορίας μεταξύ μιας «NFC ετικέτας» (NFC tag) και μιας συσκευής ή μεταξύ δύο συσκευών. Λειτουργεί στη συχνότητα των 13,56 MHz και μεταφέρει δεδομένα με ρυθμό έως και 424 kbps.

Η χρησιμότητά του μπορεί να φανεί σε διάφορες περιπτώσεις:

- Πληρωμές
- Εισιτήρια
- Διαφήμιση
- Πελατειακή πίστη
- Ανταλλαγή δεδομένων
- Καταγραφή παρουσίας / Έλεγχος πρόσβασης

Προς το παρόν τα smartphones που υποστηρίζουν το NFC δεν είναι πολλά αλλά η λίστα ανανεώνεται διαρκώς και αρκετά γρήγορα.

3.1 Πώς λειτουργεί

Στο NFC πάντα υπάρχει ένας αποστολέας και ένας δέκτης. Ο αποστολέας ενεργά δημιουργεί ένα πεδίο ραδιοσυχνότητας που μπορεί να τροφοδοτήσει έναν παθητικό στόχο. Επιτρέπει τη γρήγορη ανάγνωση και εγγραφή δεδομένων. Εκτός από τα κινητά υπάρχουν και NFC κάρτες σε μορφή έξυπνων καρτών με διαφορετική χωρητικότητα ανάλογα με την χρήση. Το NFC μπορεί να βρίσκεται σε 3 διαφορετικές καταστάσεις λειτουργίας. Η πρώτη είναι η Read/Write όπου η μια συσκευή είναι ενεργή και η άλλη παθητική και επιτρέπει τις εφαρμογές να μεταδώσουν και να λάβουν δεδομένα. Η δεύτερη είναι η Card emulation όπου επιτρέπει τις NFC συσκευές να συμπεριφέρονται σαν έξυπνη κάρτα. Και η τελευταία είναι η Peer to Peer που ορίζεται για επικοινωνία από συσκευή σε συσκευή σε επίπεδο σύνδεσης.

Το NFC βασίζεται στην επαγωγική ζεύξη, όπου τα αόριστα συνδεδεμένα επαγωγικά κυκλώματα μπορούν να χρησιμοποιηθούν για να μοιράσουν ενέργεια και δεδομένα ανάμεσα σε δύο συσκευές σε πολύ μικρή απόσταση.

Οι Android συσκευές που υποστηρίζουν το NFC συνήθως ψάχνουν για NFC tags όταν δεν είναι κλειδωμένη η οθόνη (εκτός αν το NFC είναι απενεργοποιημένο από το μενού ρυθμίσεων του κινητού). Όταν ανακαλυφθεί ένα tag σε κοντινή απόσταση η επιθυμητή συμπεριφορά είναι να γίνει αυτόματα η κατάλληλη ενέργεια χωρίς να ερωτηθεί ο χρήστης ποια εφαρμογή θέλει να χρησιμοποιηθεί. Λόγω της μικρής απόστασης στην οποία δουλεύει η τεχνολογία με το να αναγκαστεί ο χρήστης να επιλέξει χειροκίνητα μια ενέργεια είναι πολύ πιθανό να μετακινήσει άθελά του τη συσκευή αρκετά ώστε να διακοπεί η σύνδεση. Οπότε καλό θα είναι ο προγραμματιστής να επιλέγει μόνο τα tags που αφορούν την εφαρμογή του ώστε να αποτραπεί ένα αναδυόμενο

παράθυρο επιλογής. Το Android χρησιμοποιεί ένα ειδικό σύστημα κατανομής των tags που ανακαλύπτονται από την ακτίνα (NFC Beam) ψάχνοντας να βρει εγκατεστημένες εφαρμογές που ενδιαφέρονται για τα σαρωμένα δεδομένα.

Η μεγαλύτερη ίσως χρησιμότητα της τεχνολογίας αυτής είναι οι πληρωμές μέσω κινητού. Για παράδειγμα μπορεί να χρησιμοποιηθεί στο μέλλον στα μέσα μαζικής μεταφοράς, όπου θα υπάρχουν συστήματα πληρωμής με την τεχνολογία NFC και οι χρήστες θα περνάνε απλά τις συσκευές τους και θα αγοράζουν άμεσα και εύκολα το εισιτήριο. Βέβαια η περίπτωση αυτή δεν είναι ίδια με την χρήση του NFC για το άνοιγμα ενός URL όπου έχουμε απλή μεταφορά πληροφορίας. Οι πληρωμές μέσω κινητού είναι πολύ πιο πολύπλοκη διαδικασία και περιλαμβάνει μια εφαρμογή «πορτοφολιού» για το κινητό καθώς και πρωτόκολλα που να διασφαλίζουν την ασφάλεια της συναλλαγής.

3.2 NFC tags

Τα NFC tags (ετικέτες) μπορεί να είναι αυτοκόλλητα, περικάρπια, μπρελόκ ή έξυπνες κάρτες που περιέχουν μικροτσίπ με μικρές κεραίες τα οποία μπορούν να αποθηκεύσουν περιορισμένες ποσότητες πληροφοριών για μεταφορά μεταξύ NFC συσκευών, για παράδειγμα κινητών τηλεφώνων. Λόγω της επαγωγικής ζεύξης στην οποία βασίζεται αυτή η τεχνολογία, τα tags δεν απαιτούν μπαταρίες για να λειτουργήσουν γιατί χρησιμοποιούν την ενέργεια της ανοιχτής NFC συσκευής που βρίσκεται κοντά τους.

Τύπος	Προϊόντα	Χωρητικότητα
Type1	Innovision Topaz	96 bytes
Type2	NXP MIFARE Ultralight (C)	48 - 144 bytes
Type3	Sony Felica	1, 4, 9 Kbytes
Type4	NXP DESFire	2, 4, 8 Kbytes
Type5	NXP MIFARE Classic	1, 4 Kbytes

Πίνακας ΙΙ : Τύποι NFC

Υπάρχει ένα μεγάλο εύρος διαφορετικών τύπων πληροφορίας που μπορεί να αποθηκευτεί σε ένα NFC tag. Το μέγεθος των αποθηκευμένων δεδομένων εξαρτάται από τον τύπο του tag που θα χρησιμοποιηθεί αφού το καθένα έχει διαφορετική διαθέσιμη μνήμη. Για παράδειγμα αν θέλουμε να αποθηκεύσουμε ένα URL ή ένα τηλεφωνικό νούμερο μπορούμε να χρησιμοποιήσουμε το βασικό NFC tag (με μέγιστη χωρητικότητα τα 96 bytes) ενώ για κάτι μεγαλύτερο θα χρειαστούμε μεγαλύτερης χωρητικότητας tag (μέχρι και 9 Kbytes).

Συνήθως η πληροφορία αποθηκεύεται σε συγκεκριμένο σύστημα αρχείων, το NDEF (NFC data exchange format) ώστε να διαβάζεται από τις περισσότερες υποστηριζόμενες συσκευές.

Τα tags μπορούν να «κλειδωθούν» ώστε να μην μπορεί να αλλοιωθεί – μετατραπεί η πληροφορία τους μετά την πρώτη εγγραφή. Στα περισσότερα tags αυτή η διαδικασία δεν είναι αντιστρέψιμη οπότε δεν μπορούν να ξεκλειδωθούν μετά το κλείδωμά τους.

Ο ευκολότερος τρόπος να αποθηκεύσουμε σε μια ετικέτα τις πληροφορίες που θέλουμε είναι να έχουμε στη διάθεσή μας μια συσκευή – κινητό τηλέφωνο Android που υποστηρίζει την τεχνολογία αυτή (τα καινούργια BlackBerry και Nokia υποστηρίζουν το NFC επίσης) και χρησιμοποιώντας την κατάλληλη εφαρμογή το "encoding" ενός tag είναι θέμα μερικών λεπτών.

Λόγω της αύξησης των εφαρμογών που επιτρέπουν το encoding ενός NFC tag ο χρήστης μπορεί να κωδικοποιήσει διάφορα tags ώστε να ανοίγουν/κλείνουν το Bluetooth ή το Wi-Fi, να ανοίγουν το αγαπημένο του site μέχρι και να αλλάζουν πολλές από τις ρυθμίσεις του κινητού μόλις πηγαίνει στη δουλειά του για παράδειγμα.

Πλεονεκτήματα

- Οι NFC αλληλεπιδράσεις είναι εύκολες και απλές καθώς δεν χρειάζεται παρά μόνο ένα απλό άγγιγμα.
- Λόγω της μικρής απόστασης που απαιτείται επιβεβαιώνεται η φυσική παρουσία του χρήστη με τον καλύτερο τρόπο.
- Η NFC τεχνολογία διευκολύνει την απλή και γρήγορη εγκατάσταση των ασύρματων τεχνολογιών όπως το Bluetooth και το Wi-Fi.
- Είναι εγγενώς ασφαλής η χρήση καθώς οι μεταδόσεις είναι μικρής εμβέλειας (από ένα άγγιγμα σε μόλις λίγα εκατοστά).
- Μεγάλο εύρος χρησιμότητας.
- Αξιοποιεί τα κινητά τηλέφωνα ως μέσο αλληλεπίδρασης τα οποία είναι ευρέως διαδεδομένα και τα κουβαλάμε πάντα μαζί μας, έχουν επεξεργαστή, έχουν πρόσβαση στο διαδίκτυο, είναι διαδραστικά (πληκτρολόγιο, οθόνη αφής) και διαθέτουν ώριμα λειτουργικά συστήματα.

Μειονεκτήματα

- Τα συστήματα NFC είναι εύκολο να υποκλαπούν γιατί δεν υπάρχει κάποιο αυστηρό μέτρο ασφαλείας.
- Η χρήση του NFC εκπέμπει ακτινοβολία.
- Επειδή η λειτουργία του NFC γίνεται εξ αποστάσεως υπάρχει ο κίνδυνος απώλειας των δεδομένων.

4. Τι είναι το Android

Το Android είναι ένα πακέτο λογισμικού για κινητές συσκευές που περιλαμβάνει ένα λειτουργικό σύστημα, εφαρμογές διαχείρισης hardware και μερικά πολύ σημαντικά προγράμματα για τον χρήστη. Ο όρος Android είναι ελληνικής προέλευσης καθώς προέρχεται από τη λέξη Andro-«ανθρώπινη» και eides-«μορφή, σχήμα». Η έννοια που δίνεται στη λέξη Android είναι το «Ανδροειδές» και συμβολίζεται σαν ένα ρομπότ με ανθρώπινη μορφή, το οποίο είναι και το λογότυπο του λειτουργικού. Το πακέτο ανάπτυξης λογισμικού για το Android (Android SDK) προσφέρεται δωρεάν και παρέχει στον προγραμματιστή πρόσβαση στις διεπαφές προγραμματισμού εφαρμογών (APIs) και τα απαραίτητα εργαλεία ώστε να μπορεί να ξεκινήσει την ανάπτυξη εφαρμογών για την πλατφόρμα του Android χρησιμοποιώντας την γλώσσα προγραμματισμού Java.

4.1 Βασικά χαρακτηριστικά

- **Πλαίσιο εφαρμογών που επιτρέπει την επαναχρησιμοποίηση και αντικατάσταση** στοιχείων/εξαρτημάτων
- **Εικονική μηχανή** βελτιστοποιημένη για κινητές συσκευές
- **Ενσωματωμένος φυλλομετρητής ιστού** βασισμένο στο ελεύθερο λογισμικό WebKit
- **Βελτιστοποιημένα γραφικά βασισμένα σε** δισδιάστατες ψηφιακές γραφικές βιβλιοθήκες, τρισδιάστατα γραφικά βασισμένα στην OpenGL ES 1.0 (προαιρετική υποστήριξη επιτάχυνσης υλικού)
- **SQLite** για δομημένη αποθήκευση δεδομένων
- **Υποστήριξη πολυμέσων για** κοινές μορφές ήχου, βίντεο και εικόνων (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **Υποστήριξη τηλεφωνικού δικτύου GSM** (εξαρτημένο από το hardware)
- **Υποστήριξη τεχνολογιών συνδεσιμότητας Bluetooth, EDGE, 3G, NFC και Wi-Fi** (εξαρτημένο από το hardware)
- **Κάμερα, GPS, πυξίδα και επιταχυνσιόμετρο** (εξαρτημένο από το hardware)
- **Πλούσιο προγραμματιστικό περιβάλλον συμπεριλαμβανομένου ενός προσομοιωτή συσκευής**, εργαλεία αποσφαλμάτωσης, προφίλ μνήμης και απόδοσης και ένα πρόσθετο για το Eclipse IDE

Εφαρμογές

Το Android περιλαμβάνει ένα πλήρες πακέτο βασικών εφαρμογών όπως προγράμματα αποστολής – λήψης email και SMS/MMS, ημερολόγιο, χάρτες, φυλλομετρητή ιστού, επαφές, κλήσεις κλπ. Όλες οι εφαρμογές αυτές είναι γραμμένες στη γλώσσα προγραμματισμού Java.

Η δύναμη του ανοιχτού λογισμικού

Το Android κατασκευάστηκε από τη βάση του ώστε να επιτρέπει τους προγραμματιστές να δημιουργούν εφαρμογές που μπορούν να εκμεταλλευτούν πλήρως αυτά που έχει να προσφέρει η εκάστοτε κινητή συσκευή Android. Το λειτουργικό είναι πραγματικά ανοιχτό. Για παράδειγμα, μια εφαρμογή μπορεί να καλέσει όποτε χρειαστεί οποιαδήποτε από τις λειτουργικότητες τις συσκευής όπως τη δημιουργία κλήσεων, την αποστολή μηνυμάτων, τη χρήση της κάμερας κλπ επιτρέποντας τον προγραμματιστή να δημιουργεί όλο και πιο πλούσιες και συνεκτικές εφαρμογές για τον χρήστη. Το γεγονός ότι βασίζεται στο Linux και το ότι χρησιμοποιεί μια εικονική μηχανή που βελτιστοποιεί τη χρήση μνήμης και της υπολογιστικής ισχύς του υλικού το κάνουν σταθερό λειτουργικό και το γεγονός ότι είναι ανοιχτό το κάνει να μπορεί να επεκταθεί στη χρήση καινούργιων τεχνολογιών μόλις εξελίσσονται. Η πλατφόρμα μπορεί να συνεχίσει να αναπτύσσεται όσο η κοινότητα των προγραμματιστών θα δουλεύει για την ανάπτυξη όλο και πιο καινοτόμων εφαρμογών.

Ισότητα των εφαρμογών

Το Android δεν διαφοροποιεί τις ενσωματωμένες εφαρμογές του πυρήνα του με αυτές που αναπτύσσονται από τρίτους. Όλες έχουν την ίδια πρόσβαση στις δυνατότητες της κινητής συσκευής και έτσι οι χρήστες έχουν ένα ευρύ φάσμα υπηρεσιών. Είναι δυνατή η πλήρης προσαρμογή του κινητού στα ενδιαφέροντα του χρήστη από την αρχική οθόνη του μέχρι και τα στυλ των εφαρμογών. Επίσης ο χρήστης μπορεί να ορίσει ποιες εφαρμογές θα είναι προεπιλεγμένες για την πλοήγηση στον ιστό, για την παρουσίαση φωτογραφιών κλπ.

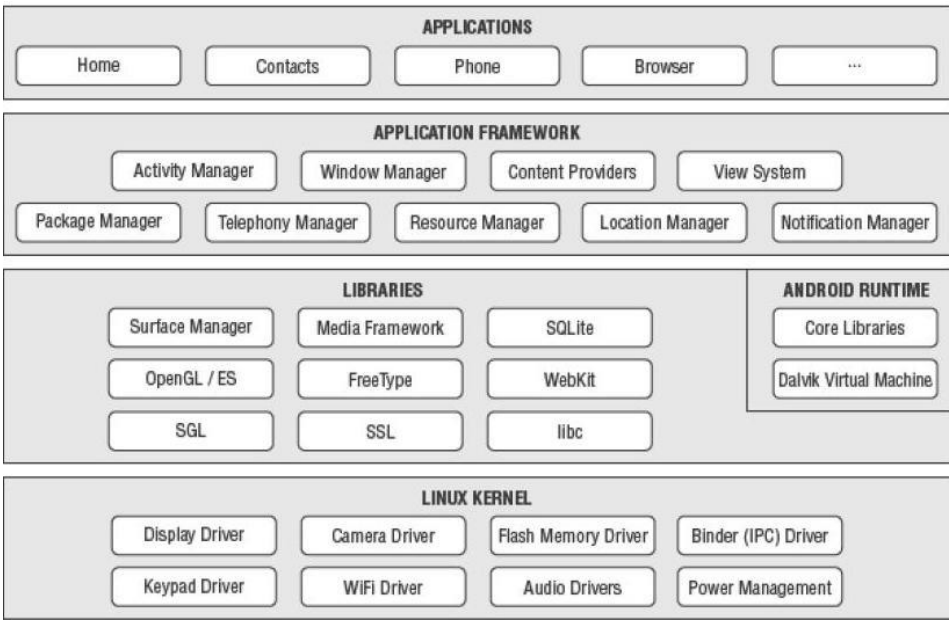
Κανένας περιορισμός μεταξύ των εφαρμογών

Με το Android είναι δυνατή η ανάπτυξη καινοτόμων εφαρμογών χωρίς να υπάρχει περιορισμός χρήσης μεταξύ άλλων υπαρχόντων. Για παράδειγμα ο προγραμματιστής μπορεί να συνδυάσει πληροφορίες που έχει συλλέξει από τον ιστό, τις επαφές του κινητού, το ημερολόγιο, και της τοποθεσίας βάση του GPS σήματος για να προσφέρει στο χρήστη ειδοποιήσεις για το αν οι φίλοι του βρίσκονται αρκετά κοντά ώστε να μπορούν να συναντηθούν.

Εύκολη και γρήγορη ανάπτυξη εφαρμογών

Για τον προγραμματιστή παρέχεται ένα μεγάλο εύρος βιβλιοθηκών και εργαλείων που μπορούν να κάνουν την υλοποίηση πλούσιων εφαρμογών πολύ πιο εύκολη. Για παράδειγμα, η τοποθεσία του κινητού, η επικοινωνία με άλλα κινητά (για δημιουργία peer-to-peer δικτύων) προσφέρονται ελεύθερα στον προγραμματιστή για να υλοποιήσει εφαρμογές που περιορίζονται μόνο από τη φαντασία και τη διορατικότητά του.

4.2 Από τι αποτελείται το Android



Εικόνα Ι : Δομή του Android

- *Linux Kernel*

Αυτός είναι ο Kernel (πυρήνας) στον οποίο βασίζεται το Android και βρίσκεται στο χαμηλότερο επίπεδο. Περιέχει τους Drivers τους οποίους χρειάζεται για να τρέξει το σύστημα, όπως της οθόνης, της κάμερας κ.α. Ειδικότερα, το Android χρησιμοποιεί τον πυρήνα Linux για τη διαχείριση μνήμης, τη διαχείριση διεργασιών, τη δικτύωση και άλλες υπηρεσίες του λειτουργικού συστήματος. Ο πυρήνας είναι ο «σκελετός» του Android.

- *Native Libraries*

Οι Εγγενείς Βιβλιοθήκες βρίσκονται στο αμέσως υψηλότερο επίπεδο και περιλαμβάνουν όλο τον κώδικα που περιέχει το Android OS. Είναι γραμμένες στις γλώσσες C και C++ και δεν είναι εφαρμογές που μπορούν να κληθούν από μόνες τους. Υπάρχουν για να μπορούν να κληθούν από προγράμματα υψηλότερου επιπέδου. Για παράδειγμα, η SQLite βιβλιοθήκη παρέχει υποστήριξη έτσι ώστε μια εφαρμογή να χρησιμοποιήσει την αποθήκευση δεδομένων, η Webkit βιβλιοθήκη παρέχει λειτουργίες για τη διαδικτυακή περιήγηση.

Από την έκδοση Donut (1.6) και μετά, οι προγραμματιστές έχουν τη δυνατότητα να γράφουν τις δικές τους τέτοιες βιβλιοθήκες με τη χρήση της εργαλειοθήκης NDK (Native Development Kit).

- *Android Runtime*

Στο ίδιο επίπεδο με τις βιβλιοθήκες, το Android Runtime (Χρόνος Εκτέλεσης) παρέχει ένα σύνολο βασικών βιβλιοθηκών που επιτρέπουν στους προγραμματιστές να γράψουν εφαρμογές χρησιμοποιώντας τη JAVA. Επίσης περιλαμβάνει την "Dalvik virtual machine", που επιτρέπει κάθε εφαρμογή να τρέξει την δικιά της εργασία. Η Dalvik είναι μια εξειδικευμένη εικονική μηχανή, ειδικά διαμορφωμένη για κινητές συσκευές που έχουν περιορισμένη μνήμη και ισχύ. Το Android περιλαμβάνει ένα σύνολο βασικών βιβλιοθηκών που παρέχουν τις περισσότερες από τις διαθέσιμες λειτουργίες των βασικών βιβλιοθηκών της Java. Κάποια πακέτα και κλάσεις υπάρχουν και στο Android, κάποια άλλα δεν υποστηρίζονται καθόλου, ενώ ταυτόχρονα το Android παρέχει και επιπρόσθετα προσαρμοσμένα στις δικές του ανάγκες.

- *Dalvik VM (Virtual Machine)*

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

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

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

Η Dalvik εκτελεί τα Dalvik εκτελέσιμα αρχεία των οποίων η μορφή είναι βελτιστοποιημένη ώστε να καταλαμβάνουν την ελάχιστη δυνατή μνήμη. Τα .dex εκτελέσιμα δημιουργούνται μετασχηματίζοντας κλάσεις που έχουν μεταγλωττιστεί από την Java χρησιμοποιώντας εργαλεία που παρέχονται μέσα στο SDK.

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

- *Application Framework*

Το διαθέσιμο Πλαίσιο Εφαρμογών εκθέτει υψηλού επιπέδου δυνατότητες του Android στους προγραμματιστές των εφαρμογών ώστε να τις χρησιμοποιήσουν στις εφαρμογές τους. Αυτό το πλαίσιο είναι προ-εγκατεστημένο στο Android, αλλά είναι επεκτάσιμο, αφού ο κάθε κατασκευαστής μπορεί να το συμπληρώσει με δικά του κομμάτια. Τα σημαντικότερα δομικά στοιχεία του πλαισίου αυτού είναι:

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

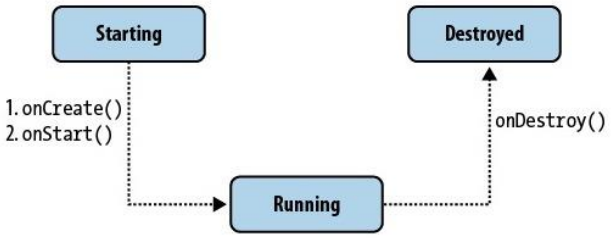
- *Applications*

Το πιο υψηλό επίπεδο, εδώ υπάρχουν οι εφαρμογές "bloatware" που είναι εργοστασιακά εγκατεστημένες στις Android συσκευές (όπως τηλέφωνο, επαφές, μουσική κλπ), όπως επίσης και όλες οι εφαρμογές τρίτων που μπορούν να εγκατασταθούν στη συσκευή.

4.3 Βασικά συστατικά εφαρμογών του Android

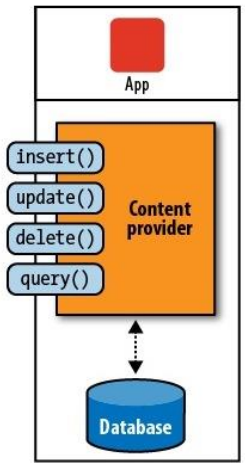
Οι εφαρμογές Android αποτελούνται από κάποια συστατικά, που συνδέονται χρησιμοποιώντας ένα XML αρχείο, το AndroidManifest.xml του κάθε project, το οποίο περιγράφει κάθε συστατικό και πώς αλληλεπιδρά με τα άλλα. Τα παρακάτω συστατικά παρέχουν τις δομικές μονάδες για τις εφαρμογές:

- **Δραστηριότητα (Activity)** - Είναι το επίπεδο παρουσίασης (presentation layer) της εφαρμογής. Μια δραστηριότητα είναι μια απλή οθόνη της εφαρμογής. Κάθε δραστηριότητα υλοποιείται σαν μια κλάση που επεκτείνει (extends) την βασική κλάση Δραστηριότητα (Activity base class). Η συγκεκριμένη κλάση προβάλει μια διεπαφή χρήστη (user interface) αποτελούμενη από Όψεις (Views) και ανταποκρίνεται σε Συμβάντα (Events). Ένα Activity είναι ισοδύναμο με μια Φόρμα (Form) στην ανάπτυξη εφαρμογών για υπολογιστές. Τα Activities είναι ίσως το πιο βασικό συστατικό μιας εφαρμογής αφού είναι υπεύθυνα για κάθε οθόνη που βλέπει ο χρήστης και εξηγούνται αναλυτικότερα παρακάτω.
- **Υπηρεσία (Service)** - Μια Υπηρεσία είναι κώδικας που τρέχει για μεγάλο χρονικό διάστημα και χωρίς διεπαφή χρήστη (UI), ενημερώνοντας τις πηγές δεδομένων και τις ορατές Δραστηριότητες (Activities) και ενεργοποιώντας Ειδοποιήσεις (Notifications). Αυτές τρέχουν και επεξεργάζονται δεδομένα ακόμα και όταν οι δραστηριότητες της εφαρμογής δεν είναι ενεργές ή ορατές. Ένα καλό παράδειγμα



Εικόνα ΙΙ : Κύκλος ζωής υπηρεσίας

είναι μια εφαρμογή που αναπαράγει μουσική από μια λίστα μουσικών κομματιών (media player). Σε μια τέτοια εφαρμογή, θα υπήρχαν κατά πάσα πιθανότητα μία ή και παραπάνω δραστηριότητες που επιτρέπουν στον χρήστη να επιλέξει τραγούδια και να τα αναπαράγει. Ωστόσο, η αναπαραγωγή από μόνη της δεν θα έπρεπε να διαχειρίζεται από την δραστηριότητα γιατί ο χρήστης θα περίμενε την μουσική να συνεχίζει να παίζει ακόμη και μετά την πλοήγησή του σε μια νέα οθόνη. Σε αυτή τη περίπτωση, η δραστηριότητα της αναπαραγωγής μουσικής θα ξεκινούσε μια υπηρεσία για να τρέξει στο παρασκήνιο και να συνεχίσει η μουσική να παίζει. Το σύστημα τότε θα κρατά το service της αναπαραγωγής ενεργό μέχρι να τελειώσει το κομμάτι. Όταν πραγματοποιηθεί σύνδεση σε μια υπηρεσία, μπορεί να υπάρξει επικοινωνία με αυτή μέσω μιας διεπαφής που προσφέρεται από την υπηρεσία. Για το service της μουσικής, αυτό θα επέτρεπε την παύση ή την επιστροφή πίσω στο κομμάτι (rewind) κλπ. Ο κύκλος ζωής των services είναι πολύ απλός, όπως φαίνεται και στο σχήμα.



Εικόνα ΙΙΙ : Παροχέας Περιεχομένου

- **Παροχέας Περιεχομένου (Content Provider)** – Οι εφαρμογές μπορούν να σώζουν τα δεδομένα τους σε αρχεία, σε μια βάση δεδομένων SLQLite ή με οποιοδήποτε άλλο μηχανισμό μπορούν. Ένας παροχέας περιεχομένου, ωστόσο, είναι χρήσιμος ώστε τα δεδομένα μιας εφαρμογής να είναι διαθέσιμα και σε άλλες εφαρμογές. Ένας content provider είναι μια κλάση που υλοποιεί μια συγκεκριμένη ομάδα μεθόδων που επιτρέπουν σε άλλες εφαρμογές να αποθηκεύουν και να επανακτούν δεδομένα του τύπου που διαχειρίζεται ο provider. Οι συσκευές Android περιλαμβάνουν διάφορους εγγενείς παροχείς (Native Content Providers) που εκθέτουν τις χρήσιμες βάσεις δεδομένων, όπως για παράδειγμα των στοιχείων των επαφών του χρήστη.
- **Πρόθεση (Intent)** - Το Android χρησιμοποιεί μια ειδική κλάση που λέγεται Πρόθεση (Intent) για να κινείται από οθόνη σε οθόνη. Η πρόθεση περιγράφει τι θέλει η εφαρμογή να γίνει στη συνέχεια. Τα δύο πιο σημαντικά μέρη της δομής δεδομένων του intent είναι η Δράση (Action) και τα δεδομένα βάσει των οποίων αυτή θα εκτελεστεί. Τυπικές τιμές για μια δράση είναι η MAIN (η κεντρική είσοδος της εφαρμογής), VIEW, PICK, EDIT κλπ. Τα δεδομένα εκφράζονται ως URI (Uniform Resource Indicator). Για παράδειγμα, για να εμφανιστεί μια ιστοσελίδα στον φυλλομετρητή (browser), δημιουργείται ένα intent με action VIEW και τα δεδομένα ως ένα URL.

```
New Intent (android.content.Intent.VIEW_ACTION;
ContentURI.create("http://www.google.com"));
```

Υπάρχει μια σχετική κλάση που λέγεται Φίλτρο Πρόθεσης (Intent Filter). Ενώ ένα intent είναι στην ουσία ένα αίτημα για να γίνει κάτι, το Intent Filter είναι μια περιγραφή του τι είναι δυνατόν να διαχειριστεί ένας Δέκτης Πρόθεσης (Intent Receiver). Μια δραστηριότητα που είναι σε θέση να προβάλει πληροφορίες επικοινωνίας για ένα άτομο, θα ανακοίνωνε με ένα Intent Filter ότι γνωρίζει πώς να διαχειριστεί τη

VIEW_ACTION όταν τα δεδομένα αντιπροσωπεύουν ένα άτομο. Οι δραστηριότητες ανακοινώνουν τα Intent Filters στο AndroidManifest.xml αρχείο.

Η πλοήγηση από οθόνη σε οθόνη πετυχαίνεται με intents. Για να πλοηγηθεί κανείς προς τα μπρος, ένα Activity καλεί την startActivity(myIntent). Το σύστημα τότε κοιτά στα Intent Filters για όλες τις εγκατεστημένες εφαρμογές και διαλέγει τη δραστηριότητα που τα φίλτρα πρόθεσης ταιριάζουν καλύτερα με την παράμετρο 'MyIntent' της κλήσης. Το νέο Activity ενημερώνεται για το intent και ξεκινά. Η διαδικασία της υλοποίησης των προθέσεων συμβαίνει κατά τον Runtime της εφαρμογής, όταν δηλαδή καλείται η startActivity, πράγμα που προσφέρει 2 πλεονεκτήματα - κλειδιά:

1. Οι δραστηριότητες μπορούν να επαναχρησιμοποιούν κάποια λειτουργικότητα από άλλα τμήματα του κώδικα απλά κάνοντας ένα αίτημα υπό την μορφή μιας πρόθεσης.
2. Οι δραστηριότητες μπορούν να αντικατασταθούν οποιαδήποτε στιγμή από μια νέα δραστηριότητα με ένα αντίστοιχο φίλτρο πρόθεσης.

- **Δέκτης Πρόθεσης (Intent Receiver)** - Χρησιμοποιούνται για να εκτελεστεί μια εφαρμογή σε απάντηση ενός εξωτερικού συμβάντος (external event), για παράδειγμα, όταν το τηλέφωνο χτυπά, ή όταν το δίκτυο είναι διαθέσιμο, ή όταν είναι μεσάνυχτα. Οι δέκτες πρόθεσης δεν προβάλουν μια διεπαφή χρήστη (UI), ωστόσο μπορούν να προβάλουν Ειδοποιήσεις (Notifications) για να ειδοποιήσουν τον χρήστη για κάτι σημαντικό που συνέβη. Οι δέκτες πρόθεσης είναι επίσης καταχωρημένοι στο AndroidManifest.xml, αλλά μπορούν επίσης καταχωρηθούν από τον κώδικα χρησιμοποιώντας την Context.registerReceiver(). Η εφαρμογή δεν χρειάζεται να τρέχει για να κληθούν οι δέκτες πρόθεσης που έχει. Το σύστημα θα κινήσει την εφαρμογή, αν χρειαστεί, όταν ένας intent receiver ενεργοποιηθεί. Οι εφαρμογές μπορούν επίσης να στέλνουν τις δικές τους Ανακοινώσεις Πρόθεσης (Intent Broadcasts) σε άλλους με την Context.broadcastIntent().
- **Δέκτης Μετάδοσης (Broadcast Receiver)** - Βασική κλάση για τον κώδικα, που θα λαμβάνει τις προθέσεις που στέλνονται από το sendBroadcast(). Μπορεί είτε δυναμικά να καταχωρηθεί μια περίπτωση αυτής της κατηγορίας με Context.registerReceiver(), είτε να καταχωρηθεί ένα στιγμιότυπο αυτής της κλάσης σε μια εφαρμογή μέσω της ετικέτας <receiver> στο AndroidManifest.xml. Οι δέκτες μετάδοσης αρχίζουν αυτόματα την εφαρμογή αποκρινόμενοι σε ένα εισερχόμενο intent, που στην ουσία κάνει την εφαρμογή, μια εφαρμογή οδηγούμενη από τα γεγονότα. Υπάρχουν δύο σημαντικές κατηγορίες Μεταδόσεων (Broadcasts):
    1. Οι Κανονικές Μεταδόσεις - Normal broadcasts (που στέλνονται με Context.sendBroadcast) είναι απολύτως ασύγχρονες. Όλοι οι δέκτες της μετάδοσης οργανώνονται σε μια απροσδιόριστη διαταγή, συχνά συγχρόνως. Αυτό είναι αποδοτικότερο, αλλά σημαίνει ότι οι δέκτες δεν μπορούν να χρησιμοποιήσουν το αποτέλεσμα ή να αποβάλουν APIs που συμπεριλαμβάνονται εδώ.
    2. Οι Μεταδόσεις Διαταγής - Ordered broadcasts (που στέλνονται με Context.sendOrderedBroadcast) παραδίδονται σε έναν δέκτη τη φορά. Δεδομένου ότι κάθε δέκτης εκτελείται στη συνέχεια, μπορεί να διαδώσει ένα αποτέλεσμα στον επόμενο δέκτη, ή μπορεί να αποβάλει τη μετάδοση έτσι ώστε να μη περάσει σε άλλους δέκτες. Ο χειρισμός της εκτέλεσης των δεκτών διαταγής μπορεί να γίνει με την ιδιότητα του Φίλτρου Προθέσεων Ταιριάσματος (Matching Intent Filter). Οι δέκτες με την ίδια προτεραιότητα τρέχουν με τυχαία σειρά.
- **Ειδοποιήσεις (Notifications)** - Είναι ένα πλαίσιο ειδοποιήσεων χρήστη. Τα notifications δίνουν την δυνατότητα ειδοποίησης του χρήστη χωρίς να υπάρξει παρεμβολή στο τρέχον Activity και του παρέχουν τις εξής δυνατότητες:

1. Να δημιουργήσει ένα νέο εικονίδιο status bar.
2. Να επιδείξει τις πρόσθετες πληροφορίες (και να προωθήσουν μια πρόθεση) στο εκτεταμένο παράθυρο status bar.
3. Να ενεργοποιήσει τα φώτα/LEDs.
4. Να δονηθεί το τηλέφωνο.
5. Να ενεργοποιήσει διάφορους ήχους (για παράδειγμα ringtones).

Οι ειδοποιήσεις είναι ο προτιμότερος τρόπος για τα αόρατα τμήματα της εφαρμογής (δέκτες μετάδοσης, υπηρεσίες, και ανενεργές δραστηριότητες) για να προειδοποιήσουν τους χρήστες ότι κάποια γεγονότα που έχουν εμφανιστεί απαιτούν την προσοχή του χρήστη. Παραδείγματος χάριν, όταν λαμβάνει μια συσκευή ένα μήνυμα κειμένου ή μια εισερχόμενη κλήση, προειδοποιεί με τη λάμψη των φώτων, την παραγωγή ήχων, την επίδειξη εικονιδίων ή με την εμφάνιση μηνυμάτων διαλόγου.

Ο Χειριστής Ειδοποιήσεων (Notification Manager) είναι μια υπηρεσία συστήματος που χρησιμοποιείται για να χειριστεί τις ειδοποιήσεις. Καλείται μέσω της μεθόδου getSystemService, όπως φαίνεται παρακάτω:

String svcName = Context.NOTIFICATION_SERVICE;

NotificationManager notificationManager;

NotificationManager = (NotificationManager) getSystemService (svcName);

Χρησιμοποιώντας τον χειριστή ειδοποιήσεων, μπορούν να ενεργοποιηθούν νέες ειδοποιήσεις, να τροποποιηθούν οι υπάρχουσες ή να αφαιρεθούν εκείνες που δεν είναι απαραίτητες ή επιθυμητές.

4.4 Activities (Δραστηριότητες)

Ένα Activity (δραστηριότητα) στο Android είναι συνήθως μια απλή οθόνη που ο χρήστης βλέπει κάθε φορά. Μια εφαρμογή αποτελείται από πολλά Activities και ο χρήστης μπορεί να πηγαινοέρχεται σε αυτά ανάλογα με τις επιλογές που κάνει. Τα Activities είναι το ορατό κομμάτι μιας εφαρμογής.

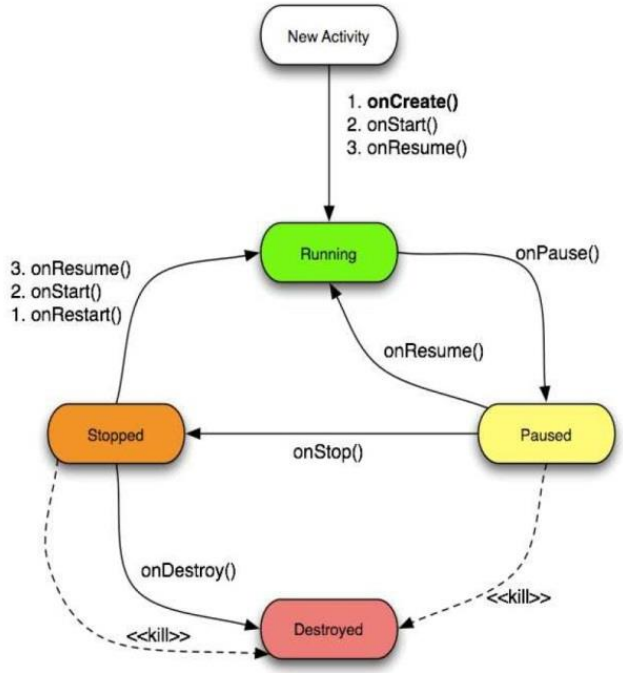
Ένα καλό παράδειγμα είναι ένα τυπικό website. Όπως αυτό αποτελείται από πολλές σελίδες στις οποίες μπορούμε να πλοηγηθούμε, έτσι και μια εφαρμογή Android που αποτελείται από πολλά Activities μας επιτρέπει την πλοήγηση σε αυτά. Αντίστοιχα με την «Αρχική σελίδα» έχουμε "main" Activity η οποία εμφανίζεται συνήθως μόλις ανοίγει η εφαρμογή. Και φυσικά αντίστοιχα με τις υπερσυνδέσεις που μας επιτρέπουν την μετάβαση σε άλλες τοποθεσίες, έτσι και στο Android μπορούμε να μεταβούμε σε άλλη εφαρμογή (πχ να κάνουμε κλήση σε κάποιον απευθείας από την εφαρμογή μας χωρίς να χρειαστεί να ανοίξουμε την εφαρμογή κλήσεων).

Κύκλος ζωής των Activities

Η εκκίνηση ενός Activity κοστίζει υπολογιστικά. Δημιουργείται μια καινούργια Linux διεργασία, δεσμεύεται μνήμη για όλα τα αντικείμενα του γραφικού περιβάλλοντος και γενικά «στήνεται» όλη η οθόνη. Επειδή χρειάζεται να γίνει μια σειρά ενεργειών όταν ανοίγουμε ένα Activity, είναι σπατάλη να κλείνει αμέσως μόλις σταματήσει ο χρήστης να το χρησιμοποιεί. Έτσι ένας «διαχειριστής δραστηριοτήτων» (Activity Manager) αναλαμβάνει να χειριστεί τον κύκλο ζωής του.

Ο διαχειριστής είναι υπεύθυνος για την δημιουργία, καταστροφή και τον χειρισμό των δραστηριοτήτων. Μόλις ο χρήστης ανοίξει για πρώτη φορά μια εφαρμογή, ο διαχειριστής θα δημιουργήσει το Activity και θα το προβάλει στην οθόνη. Όσο ο χρήστης αλλάζει οθόνες τα προηγούμενη Activities μπαίνουν σε κατάσταση αναμονής. Με τον τρόπο αυτό σε κάθε επαναφορά σε προηγούμενο Activity η διαδικασία θα γίνεται πολύ πιο γρήγορα. Οι παλαιότερες διεργασίες θα καταστρέφονται όσο ο χρήστης δεν τις χρησιμοποιεί για αρκετή ώρα για να ελευθερώσουν χώρο για την εκάστοτε ενεργή. Ο μηχανισμός αυτός βελτιώνει την ταχύτητα του γραφικού περιβάλλοντος προσφέροντας έτσι το ζητούμενο αποτέλεσμα στο χρήστη.

Ο προγραμματισμός στο Android είναι αρκετά διαφορετικός σε σχέση με άλλα περιβάλλοντα. Ο προγραμματιστής θα πρέπει να ανταπεξέλθει στις τυχόν αλλαγές της εκάστοτε κατάστασης της εφαρμογής του αντί να κάνει μόνος του τις αλλαγές. Έτσι όσον αφορά τις αλλαγές καταστάσεων σε ένα κύκλο ζωής ενός Activity, δεν μπορεί να επιλέξει σε ποια κατάσταση θα βρίσκεται κάθε φορά αλλά έχει πολλές επιλογές στο να αποφασίσει τι θα γίνεται σε κάθε μετάβαση ανάμεσα στις δυνατές καταστάσεις. Στο σχήμα φαίνονται όλες οι καταστάσεις από τις οποίες μπορεί να περάσει ένα Activity.



Εικόνα IV : Κύκλος ζωής Activity

Κατάσταση εκκίνησης (starting state)

Όταν ένα Activity δεν καταλαμβάνει ακόμα χώρο στη μνήμη λέγεται ότι είναι σε κατάσταση εκκίνησης. Κατά τη διάρκεια της εκκίνησης θα καλεστούν όλες οι μέθοδοι που ο προγραμματιστής θέλει να τρέξουν στην προκειμένη φάση ώστε να μεταβούμε στην κατάσταση λειτουργίας.

Η μετάβαση αυτή είναι η πλέον δαπανηρή υπολογιστικά διαδικασία η οποία επηρεάζει άμεσα και την αυτονομία της μπαταρίας. Αυτός ακριβώς είναι ο λόγος για τον οποίο η φιλοσοφία του Android δεν επιτρέπει την καταστροφή κάθε διεργασίας μόλις εξαφανίζεται από την οθόνη. Είναι

πιθανό να ξαναζητηθεί από το χρήστη λίγο αργότερα και έτσι «συμφέρει» να μείνει ενεργή για κάποιο διάστημα.

Κατάσταση λειτουργίας (running state)

Στην κατάσταση λειτουργίας το Activity είναι εμφανισμένο στην οθόνη και αλληλεπιδρά με τον χρήστη. Σε αυτή τη κατάσταση όλες οι αλληλεπιδράσεις (όπως πληκτρολόγηση) διαχειρίζονται από το τρέχον Activity και γι' αυτό έχει την πρώτη προτεραιότητα όσον αφορά τους πόρους που θα χρειαστεί για να δουλέψει όσο το δυνατόν γρηγορότερα. Αυτό γίνεται για να είναι το τρέχον Activity πάντα υπάκουο στις εντολές του χρήστη σε οποιαδήποτε χρονική στιγμή.

Κατάσταση παύσης (paused state)

Η κατάσταση παύσης δεν είναι συνηθισμένο σενάριο καθώς αποτελεί την περίπτωση που ένα Activity είναι μεν εμφανισμένο στην οθόνη αλλά δεν αλληλεπιδρά με τον χρήστη. Αυτό συμβαίνει σπάνια γιατί, λόγω του μικρού μεγέθους οθόνης, είναι δύσκολο να καταλαμβάνει μια δευτερεύουσα δραστηριότητα ένα μικρό μέρος της οθόνης και όχι όλο το εύρος της. Στην κατάσταση αυτή μπορούμε να οδηγηθούμε αν έχουμε κάποιο παράθυρο διαλόγου μπροστά από το προηγουμένως running Activity. Και σε αυτή την κατάσταση δίνεται αρκετά μεγάλη προτεραιότητα καθώς, λόγω της εμφάνισής του, είναι πολύ περίεργο θέαμα να απομακρυνθεί από την οθόνη.

Κατάσταση διακοπής (stopped state)

Σε κατάσταση διακοπής είναι το Activity όταν δεν εμφανίζεται στην οθόνη αλλά εξακολουθεί να καταλαμβάνει μνήμη. Από αυτή την κατάσταση μπορούμε να το ξαναεμφανίσουμε στην οθόνη και έτσι να γίνει ενεργό (σε κατάσταση λειτουργίας) και πάλι, όπως επίσης και να το «καταστρέψουμε» και να διαγραφεί από τη μνήμη.

Η λογική της διατήρησης δραστηριοτήτων σε αυτή την κατάσταση είναι ότι ο χρήστης είναι αρκετά πιθανό να τις ξαναζητήσει σύντομα και η επανεκκίνηση μιας σταματημένης διεργασίας είναι κατά πολύ προτιμότερο από το να την ξεκινήσουμε από την αρχή καθώς όλα τα αντικείμενα είναι ήδη φορτωμένα στη μνήμη και το μόνο που χρειάζεται είναι να καλεστούν ώστε να έρθουν στο προσκήνιο.

Τα σταματημένα Activities μπορούν να διαγραφούν από τη μνήμη όποτε το λειτουργικό θεωρήσει ότι χρειάζεται να ελευθερώσει χώρο στη μνήμη.

Κατάσταση καταστροφής (destroyed state)

Ένα Activity σε κατάσταση διακοπής δεν είναι πλέον στη μνήμη. Ο Activity Manager αποφάσισε ότι δεν είναι ανάγκη να καταλαμβάνει πλέον χώρο στη μνήμη κι έτσι το απομάκρυνε. Πριν καταστραφεί τελείως ένα Activity μπορεί να εκτελέσει κάποιες λειτουργίες, όπως για παράδειγμα να σώσει πληροφορίες που δεν έχουν αποθηκευτεί μέχρι τότε. Είναι πιθανό να καταστραφεί κι ένα Activity σε κατάσταση παύσης επίσης γι' αυτό και πρέπει να έχει προβλέψει ο

προγραμματιστής να γίνονται οι σημαντικές εργασίες όπως το σώσιμο πληροφοριών μόλις ένα Activity φεύγει από την κατάσταση λειτουργίας και όχι λίγο πριν καταστραφεί.

Σημείωση: Το γεγονός ότι ένα Activity είναι σε κατάσταση λειτουργίας δεν σημαίνει ότι κάνει κάποια ιδιαίτερη διαδικασία. Μπορεί απλά να περιμένει είσοδο από το χρήστη. Όπως επίσης και το γεγονός ότι ένα Activity είναι σταματημένο δεν σημαίνει ότι δεν δουλεύει στο παρασκήνιο. Τα ονόματα των καταστάσεων αναφέρονται στο πώς τις αντιλαμβάνεται ο χρήστης, δηλαδή αν εμφανίζεται μια δραστηριότητα στην οθόνη, αν αλληλεπιδρά με αυτή ο χρήστης ή αν δεν είναι εμφανισμένη καθόλου.

Μέθοδοι που χρησιμοποιούνται στα Activities

- **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 το χρήστη. Εδώ μπορούμε να ξεκινήσουμε animation και μουσική.

- **onPause():** Αυτή η μέθοδος καλείται, όταν το activity περάσει σε αναστολή. Σε αυτό το σημείο 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. Αν οι πόροι του συστήματος είναι χαμηλοί τότε το σύστημα μπορεί να καταστρέψει το activity χωρίς να την εκτελέσει.

- **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.

4.5 Services (Υπηρεσίες)

Όπως αναφέραμε τα services είναι κώδικας που τρέχει στο background χωρίς να υπάρχει κάποιο εμφανές user interface. Τα services τρέχουν ακόμα και αν ανοίξουμε άλλη εφαρμογή την

ώρα που εκτελούνται. Για παράδειγμα, ένα service μπορεί να χειρίζεται μια διαδικτυακή συναλλαγή, να παίζει μουσική να «γράφει» σε κάποιο αρχείο ή να συνεργάζεται με έναν content provider, πάντα στο παρασκήνιο.

Ένα service μπορεί να πάρει δύο μορφές:

- **Started:** Ένα service λέμε ότι είναι started (ξεκινημένο) όταν ένα κομμάτι μιας εφαρμογής (ένα activity για παράδειγμα) το έχει ξεκινήσει καλώντας την μέθοδο startService(). Μόλις ξεκινήσει ένα service μπορεί να τρέχει στο background επ' αόριστον ακόμα κι αν το activity που το κάλεσε καταστραφεί κάποια στιγμή. Συνήθως ένα service εκτελεί μια συγκεκριμένη ενέργεια και δεν επιστρέφει τίποτα εκεί από όπου καλέστηκε. Για παράδειγμα μπορεί να κατεβάζει ή να ανεβάζει ένα αρχείο στο δίκτυο. Όταν τελειώσει η ενέργεια, το service θα σταματήσει μόνο του.
- **Bound:** Ένα service λέμε ότι είναι bound (περιορισμένο) όταν ένα κομμάτι μιας εφαρμογής το καλέσει με τη μέθοδο bindService(). Ένα bound service προσφέρει μια διασύνδεση τύπου client-server που επιτρέπει οποιαδήποτε εφαρμογή να αλληλεπιδρά με αυτό, να στέλνει αιτήσεις, να παίρνει αποτελέσματα και να κάνει όλα τα παραπάνω μεταξύ διαφορετικών διεργασιών με διαδιεργασιακή επικοινωνία (Interprocess Communication-IPC). Ένα bound service μπορεί να συνεχίσει να τρέχει μόνο όσο τρέχει το κομμάτι εφαρμογής που το κάλεσε (με το οποίο είναι συνδεμένο).

Ανεξάρτητα από το ποια από τις δύο μορφές έχει το service μας, οποιαδήποτε εφαρμογή (ή κομμάτι οποιασδήποτε εφαρμογής) μπορεί να το χειριστεί καλώντας το με ένα Intent. Ωστόσο, μπορούμε να το δηλώσουμε ως private στο manifest και να μπλοκάρουμε την πρόσβαση από άλλες εφαρμογές.

Σημείωση: Τα services τρέχουν στο κύριο thread (νήμα) της διεργασίας που τα φιλοξενεί και δεν δημιουργούν καινούργιο thread ούτε τρέχουν σε ξεχωριστή διεργασία (εκτός και αν το καθορίσουμε διαφορετικά). Αυτό σημαίνει ότι αν το service μας πρόκειται να κάνει κάποια «κουραστική» για τον επεξεργαστή ενέργεια, όπως αναπαραγωγή MP3, θα πρέπει να δημιουργήσουμε καινούργιο thread μέσα στο οποίο θα τρέχει το service. Με τη χρήση ενός ξεχωριστού thread, μειώνεται το ρίσκο των σφαλμάτων τύπου Application Not Responding (ANR) και το κύριο thread της εφαρμογής μας μπορεί να μείνει αφιερωμένο στην αλληλεπίδραση με τα activities.

Βασικές μέθοδοι

Για να δημιουργήσουμε ένα service πρέπει αρχικά να δημιουργήσουμε μια υποκλάση της κλάσης Service. Στην υλοποίησή της θα πρέπει να κάνουμε override κάποιες μεθόδους οι οποίες χειρίζονται βασικά χαρακτηριστικά του κύκλου ζωής των services και παρέχουν ένα μηχανισμό που επιτρέπει κομμάτια εφαρμογών να συνδέονται αν χρειαστεί με τα services. Οι πιο σημαντικές μέθοδοι που πρέπει να παρακαμφθούν είναι οι εξής:

- **onStartCommand():** Το σύστημα καλεί αυτή τη μέθοδο όταν ένα κομμάτι μιας εφαρμογής (πχ ένα activity) ζητάει την εκκίνηση του service καλώντας τη μέθοδο startService(). Μόλις εκτελεστεί αυτή η μέθοδος, το service έχει ξεκινήσει και μπορεί να τρέχει στο background επ' αόριστον. Αν υλοποιηθεί έτσι είναι ευθύνη μας να

σταματήσουμε το service όταν τελειώσει η δουλειά του, καλώντας την stopSelf() ή την stopService().

- **onBind():** Το σύστημα καλεί αυτή τη μέθοδο όταν μια εφαρμογή θέλει να συνδεθεί με το service, καλώντας τη bindService(). Σε μια τέτοιου είδους υλοποίηση θα πρέπει να παρέχουμε μια διεπαφή που θα επιτρέπει στο χρήστη να επικοινωνεί με το service, επιστρέφοντας έναν IBinder. Πρέπει πάντα να υλοποιούμε αυτή τη μέθοδο, εκτός αν δεν θέλουμε να επιτρέψουμε σύνδεση, οπότε και θα επιστρέψουμε null.

- **onCreate():** Το σύστημα καλεί αυτή τη μέθοδο μόλις δημιουργηθεί το service ώστε να εκτελέσει διαδικασίες που πρέπει να ρυθμιστούν (πριν ακόμα καλέσει την onStartCommand() ή την onBind()). Αν το service εκτελείται ήδη, αυτή η μέθοδος δεν καλείται.

- **onDestroy():** Το σύστημα καλεί αυτή τη μέθοδο όταν το service δεν χρησιμοποιείται πλέον και πρέπει να καταστραφεί. Το service μας θα πρέπει να την υλοποιήσει ώστε να καθαριστούν τυχόν πόροι, όπως threads, listeners, receivers κλπ. Αυτή είναι και η τελευταία κλήση που λαμβάνει ένα service.

Αν ένα service ξεκινήσει από κάλεσμα της startService() (μέσω της οποίας καλείται η onStartCommand()) τότε το service συνεχίζει να τρέχει μέχρι να σταματήσει με την stopSelf() ή μέχρι να το σταματήσει κάποια εφαρμογή καλώντας την stopService().

Αν ένα service ξεκινήσει από κάλεσμα της bindService() (και η onStartCommand() *δεν* καλεστεί), τότε το service τρέχει μόνο για όσο η εφαρμογή που το κάλεσε είναι συνδεδεμένη με αυτό. Μόλις το service αποσυνδέεται από όλους τους clients που τυχόν είναι συνδεμένοι πάνω του, το σύστημα το καταστρέφει.

Το σύστημα του Android θα κλείσει αναγκαστικά ένα service μόνο όταν η μνήμη που απομένει είναι λίγη και θα πρέπει να ανακτηθούν επαρκείς πόροι συστήματος ώστε να λειτουργεί απρόσκοπτα το activity στο οποίο δουλεύει ο χρήστης. Αν το service είναι συνδεμένο με το ενεργό activity τότε είναι λιγότερο πιθανό να καταστραφεί ενώ στην περίπτωση που το service είναι φτιαγμένο ώστε να τρέχει στο προσκήνιο είναι σχεδόν απίθανο να καταστραφεί. Αντιθέτως, στην περίπτωση που το service έχει ξεκινήσει και ήδη τρέχει για ώρα, το σύστημα θα χαμηλώσει την προτεραιότητά του στη λίστα των διεργασιών παρασκηνίου, οπότε και η πιθανότητα να «σκοτωθεί» κάποια στιγμή από το σύστημα αυξάνεται (το service θα επανεκκινηθεί μόλις οι πόροι γίνουν διαθέσιμοι και πάλι). Για το λόγο αυτό πρέπει να είμαστε προσεκτικοί στη δημιουργία services και να χειριζόμαστε ομαλά την τυχόν επανεκκίνησή τους από το σύστημα.

4.6 Διαχείριση μνήμης

Το Android είναι φτιαγμένο για multitasking. Έτσι, όταν κλείνουμε μια εφαρμογή, το σύστημα δε συμπεριφέρεται όπως τα άλλα λειτουργικά συστήματα (π.χ. Windows) δηλαδή δεν κλείνει τελείως όλες τις διεργασίες (processes) που είχε ανοίξει η εφαρμογή, αλλά τις βάζει στο

παρασκήνιο (background) και τις σκοτώνει όταν νομίζει ότι χρειάζεται. Συγκεκριμένα, τις σκοτώνει όταν η μνήμη του συστήματος πέσει κάτω από ένα όριο. Η ιδέα είναι ότι αυτές οι ανενεργές διεργασίες δε βλάπτουν τη συνολική απόδοση του συστήματος και αν χρειαστεί να ξανανοιχτεί η εφαρμογή όσο τρέχουν στο υπόβαθρο, θα ανοίξει πιο γρήγορα, με αποτέλεσμα ένα γρήγορο και αποδοτικό λειτουργικό. Οι διεργασίες σκοτώνονται με σειρά παλαιότητας, όταν η μνήμη πέσει κάτω από το όριο.

Ο εσωτερικός διαχειριστής μνήμης στο Android διαχωρίζει τις εφαρμογές/διεργασίες που τρέχουν στο τηλέφωνο σε 6 καταστάσεις, από Empty App (νεκρή διεργασία, δεν κάνει απολύτως τίποτα και περιμένει να διαγραφεί ή να ξανανοιχτεί η εφαρμογή που την καλεί), μέχρι Foreground App (που τρέχει εκείνη τη στιγμή). Όταν σταματήσει ο χρήστης να βλέπει μια εφαρμογή, είτε την βάλει στο παρασκήνιο με το home button είτε την κλείσει με το back button, σταματάει αυτόματα να είναι σε κατάσταση Foreground App (προφανώς), και ανάλογα με τη λειτουργία της και τον τρόπο που έχει κωδικοποιηθεί, μπαίνει στο υπόβαθρο σε μια άλλη κατάσταση. Αυτό είναι βασική λειτουργία του Android.

Πρόβλημα που προκύπτει

Το πρόβλημα δυστυχώς είναι ότι η ιδέα δε δουλεύει τέλεια, γιατί προϋποθέτει καλά κωδικοποιημένες εφαρμογές (πράγμα που δυστυχώς δεν ισχύει από τη στιγμή που ο οποιοσδήποτε μπορεί να ανεβάσει την εφαρμογή του στο market) και εξαρτάται επίσης από τις δυνατότητες της συσκευής. Το προεπιλεγμένο όριο μνήμης για να κλείσει μια Empty App διεργασία είναι χαμηλό, π.χ. στο Galaxy S είναι 48MB (σε άλλες συσκευές είναι ακόμα λιγότερο, μέχρι και 24MB). Δηλαδή το τηλέφωνο αρχίζει να σκοτώνει apps που βρίσκονται σε κατάσταση empty, όταν η μνήμη πέσει κάτω από 48MB. Αυτό το όριο είναι πολύ χαμηλό για τα τελευταίας κατηγορίας κινητά, που τρέχουν με 150-200MB ελεύθερα. Μέχρι να πέσει η μνήμη στα 48MB, το υπόβαθρο γεμίζει με νεκρές διεργασίες και καθυστερεί επειδή δεν έχει αρκετή ελεύθερη μνήμη χωρίς λόγο. Επίσης η μπαταρία, η οποία δεν επηρεάζεται από την ελεύθερη μνήμη (κάτι που πιστεύουν λανθασμένα πολλοί), επηρεάζεται από το αν λειτουργεί ο επεξεργαστής ή όχι. Αν οι (άσχημα κωδικοποιημένες) διεργασίες στο υπόβαθρο ζητάνε ακόμα υπολογιστική ισχύ, από κει χάνεται και μπαταρία.

Λύση

Υπάρχουν 2 κοινά διαδεδομένες λύσεις, η πρώτη άκομψη και ενάντια στη φιλοσοφία του Android και η δεύτερη σαφώς καλύτερη:

1. Task manager/Task killer. Ένας καλός task manager μπορεί να δει τις διεργασίες που βρίσκονται και στις 6 καταστάσεις και ειδικά τις Empty. Και όπως στα Windows υπάρχει η δυνατότητα να σκοτωθούν όλες οι εφαρμογές που έχουν διεργασίες που τρέχουν στο background. Από τις εκδόσεις Froyo και μετά, οι εφαρμογές δε σκοτώνονται, αλλά απλά ξαναρχίζουν από την αρχή (re-initialize). Αυτό το σκότωμα/επαναρχικοποίηση δεν ενδείκνυται, τις περισσότερες φορές μάλιστα είναι κακή λύση, γιατί με το να κλείνουν διεργασίες που το Android δε σκόπευε να κλείσει, μπορεί να δημιουργηθεί πρόβλημα και μάλιστα να γίνει το κινητό πιο αργό, γιατί πρέπει να τα ξαναρχίζει όλα απ' την αρχή. Μερικοί task managers έχουν τη δυνατότητα να σκοτώνουν αυτόματα τις διεργασίες στο background μετά από ορισμένη ώρα. Υπάρχουν task managers που προστατεύουν το χρήστη από λάθη έχοντας safe modes, και μπορούν να αποκλειστούν κάποιες εφαρμογές από το να σκοτώνονται αν είναι γνωστό ότι δημιουργείται πρόβλημα. Παρ' όλα αυτά

παραμένει μια βασικά λάθος λύση, και τελείως λάθος λογική στον κόσμο Android. Και ειδικά από Froyo και μετά, οι task killers είναι τελείως άχρηστοι.

2. Το καλό με το Android είναι ότι τα όρια που λέγαμε παραπάνω μπορούν να αλλάξουν. Τα αρχικά όρια για τις 6 καταστάσεις, κάτω από τα οποία σκοτώνονται αυτόματα οι εφαρμογές, βρίσκονται αποθηκευμένα στο αρχείο συστήματος /sys/module/lowmemorykiller/parameters/minfree. Αυτές οι τιμές λοιπόν μπορούν να αλλάξουν και έτσι αντί το σύστημα να περιμένει να φτάσει η μνήμη στα 48MB για να κλείσει τα Empty Apps, να τα σκοτώνει ας πούμε όταν πέφτει κάτω από 100MB. Με αυτόν τον τρόπο ακολουθείται η λογική του Android και υπάρχει πάντα ελεύθερη μνήμη.

Η 2η λύση είναι η πιο σωστή, και είναι εμφανές ότι η συσκευή δουλεύει τέλεια χωρίς να σκοτωθεί καμία εφαρμογή. Φυσικά το minfree, ως αρχείο συστήματος, χρειάζεται δικαιώματα πρόσβασης root για να αλλάξει. Σε συνδυασμό με ένα πρόγραμμα το οποίο μπορεί να ελέγχει αν κάποιες εφαρμογές ανοίγουν χωρίς λόγο, η συσκευή θα λειτουργεί όπως ακριβώς πρέπει.

Θεωρητικά λοιπόν δεν υπάρχει λόγος να χρησιμοποιηθεί κάποιος task manager. Πρακτικά, λόγω των άσχημα κωδικοποιημένων εφαρμογών που υπάρχουν, είναι καλό να υπάρχει κι ένας task manager που να μην τρέχει συνέχεια και φυσικά χωρίς αυτόματο kill, για να σκοτώνει/ξαναρχίζει ο χρήστης οποιαδήποτε εφαρμογή δημιουργεί πρόβλημα. Και δυστυχώς, η λύση του task manager παραμένει η μόνη λύση για παλιότερες συσκευές (με έκδοση Eclair ή παλαιότερη) και γι' αυτές που δεν έχουν δικαιώματα root.

4.7 Γραφικό περιβάλλον χρήστη (Graphical User Interface-GUI)

Δύο είναι τα βασικά στοιχεία του γραφικού περιβάλλοντος στο Android. Το πρώτο είναι Αρχική Οθόνη. Η Αρχική Οθόνη είναι ό,τι ακριβώς είναι η Επιφάνεια Εργασίας για τους υπολογιστές. Η διαφορά είναι ότι η Αρχική Οθόνη των Android συσκευών δεν είναι στατική αλλά εκτείνεται δεξιά και αριστερά. Οι χρήστες των Windows είναι συνηθισμένοι σε στατικές Επιφάνειες Εργασίες. Όσοι όμως έχουν δουλέψει με Linux ξέρουν ότι μπορούν να έχουν δύο και τρεις Επιφάνειες Εργασίες. Η επιφάνεια εργασίας της συσκευής μπορεί να διαμορφωθεί όπως το επιθυμεί ο χρήστης τοποθετώντας συντομεύσεις ή widgets ώστε να του παρέχεται ταχεία πρόσβαση σε ό,τι χρησιμοποιεί πιο συχνά. Μπορεί να προσθέσει από μια απλή συντόμευση για κλήση ενός αγαπημένου προσώπου μέχρι μια εφαρμογή για τον καιρό. Στην Αρχική Οθόνη μπορούμε να επιστρέψουμε από οπουδήποτε κι αν βρισκόμαστε πατώντας το πλήκτρο του Home.

Οι χρήστες του Android έχουν την δυνατότητα με τους Home Launchers να τροποποιήσουν την αρχική τους οθόνη και το μενού και να χρησιμοποιούν επιπλέον δυνατότητες από αυτές που προσφέρει ο προ-εγκατεστημένος launcher. Οι Home Launchers προσθέτουν όλα τα είδη επιπλέον χαρακτηριστικών για τις αρχικές οθόνες του κινητού όπως χειρονομίες, διαφόρων ειδών συντομεύσεις, ακόμα και χαμηλού επιπέδου ρυθμίσεις που μπορούν να βοηθήσουν στην αύξηση της ταχύτητας ενός παλιού κινητού. Οι Home Launchers προσφέρουν πολλές πρόσθετες επιλογές στην συσκευή.

Το δεύτερο βασικό στοιχείο του γραφικού περιβάλλοντος είναι η Οθόνη Εφαρμογών. Είναι το αντίστοιχο του "All Programs" του μενού "Start" των Windows. Εκεί υπάρχουν όλες οι εγκατεστημένες εφαρμογές στο κινητό. Εξυπακούεται ότι αν θέλουμε μπορούμε να εμφανίσουμε οποιαδήποτε εφαρμογή και στην Αρχική Οθόνη. Η οθόνη των εφαρμογών είναι προσβάσιμη αγγίζοντας το κουμπί με τα τετράγωνα.

Η έξοδος από την εφαρμογή που χρησιμοποιούμε γίνεται με το κουμπί Back ή με το Home (με το οποίο θα πάμε στην Αρχική Οθόνη). Υπάρχει όμως μια βασική διαφορά μεταξύ τους. Με το Home η εφαρμογή εξακολουθεί να τρέχει στο παρασκήνιο. Αν θέλουμε να βγούμε από μια εφαρμογή και αυτή ταυτόχρονα να τερματιστεί πρέπει να χρησιμοποιήσουμε το κουμπί Back.

4.7.1 Τρόποι δημιουργίας διεπαφής χρήστη – user interface

Ένα user interface μπορεί να σχεδιαστεί χρησιμοποιώντας μία από τις δύο μεθόδους: α) τη δηλωτική και β) την διαδικαστική. Παρακάτω αναφέρονται μερικές θεμελιώδεις κλάσεις για τον σχεδιασμό user interface καθώς και τα Layouts (διατάξεις).

Θεμελιώδεις κλάσεις για τον σχεδιασμό user interface

- **Views (Προβολές):** Τα 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 αυτή 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 (δραστηριότητες):** Τα **activities**, που περιγράψαμε αναλυτικά στο προηγούμενο κεφάλαιο, αντιπροσωπεύουν το παράθυρο ή την οθόνη που εμφανίζεται. Για να εμφανίσουμε ένα user interface εκχωρούμε ένα view, που μπορεί να είναι ένα προσαρμοσμένο view ή ένα layout, σε ένα activity.

Μέθοδοι σχεδίασης user interface

Από τους δύο τρόπους σχεδίασης, ο διαδικαστικός είναι ουσιαστικά η σχεδίαση μέσω κώδικα. Στην περίπτωση του δηλωτικού τρόπου δεν χρησιμοποιούμε καθόλου κώδικα. Ο δηλωτικός τρόπος σχεδίασης στο Android με XML είναι παρόμοιος με τον σχεδιασμό μιας ιστοσελίδας.

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 ενός view, είτε ένα layout. Αυτό μας δίνει τη δυνατότητα να σχεδιάσουμε το user interface είτε με κώδικα, είτε μέσω XML.

Ένα απλούστατο user interface είναι το εξής:

1. **Με XML (δηλωτικός τρόπος)**

Έστω το Example.java που 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);
      }
}
```

Στον παραπάνω κώδικα, το Layout main.xml που βρίσκεται στον 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 μια xml ετικέτα, την <LinearLayout>, που αποτελεί και το layout που επιλέξαμε ώστε να απεικονίζουμε τα στοιχεία που περιέχει. Επίσης, κάτω από την <LinearLayout>, βρίσκεται η ετικέτα <TextView> που είναι επέκταση της View. Με αυτή 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 ορίσαμε, 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 εμφανή τα περιεχόμενά της.

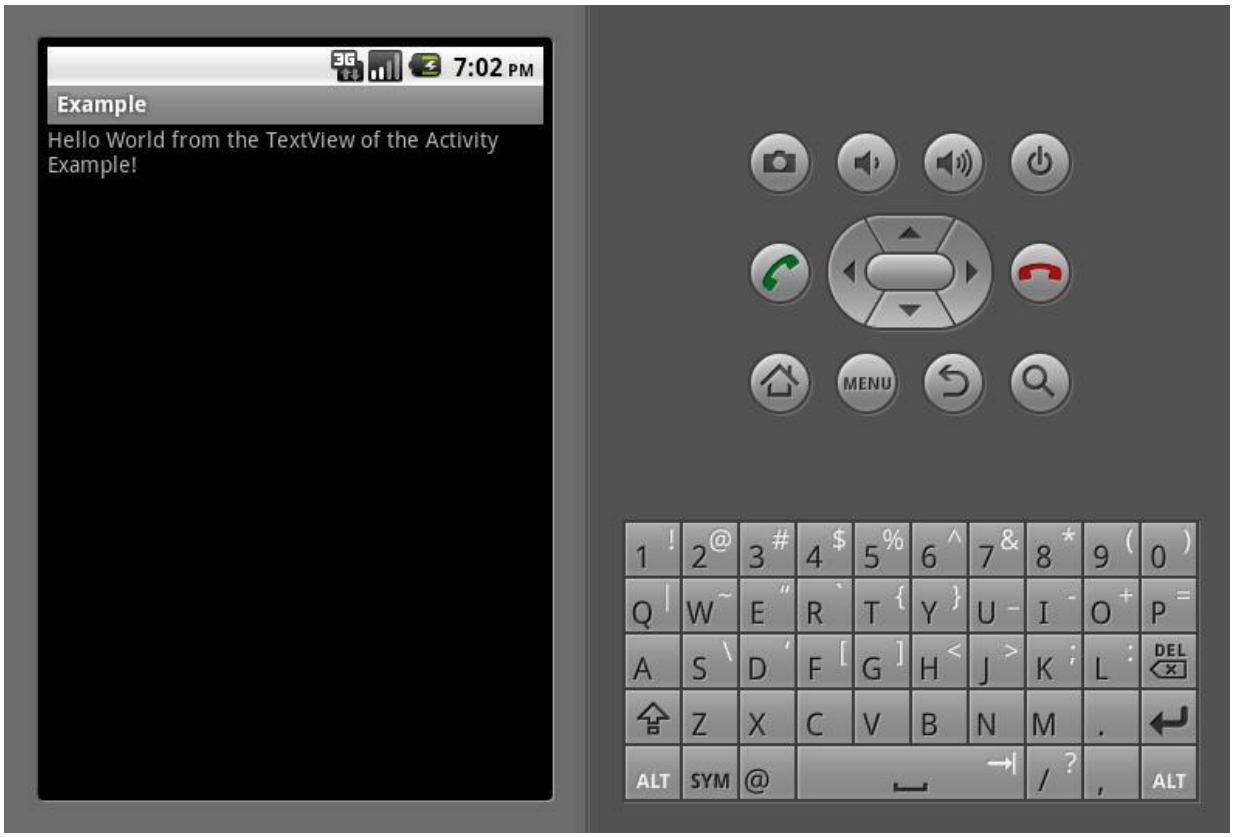
- **text:** Καθορίζει το κείμενο που θα εμφανίζεται. Αυτό γίνεται είτε εισάγοντας απευθείας το κείμενο που θέλουμε, είτε με τη χρήση αναγνωριστικού κειμένου που

θέλουμε από το αρχείο strings.xml που βρίσκεται στον κατάλογο res/values. Αυτό το κάνουμε για 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 Activity Example!".

Εκτελώντας αυτό το project ως Android Application θα εμφανιστεί στον emulator το παρακάτω αποτέλεσμα:



Εικόνα V : Απλό παράδειγμα εφαρμογής στον emulator

Σε αυτό το σημείο πρέπει να αναφέρουμε ότι αν θέλουμε να σχετίσουμε την <TextView> με τον κώδικά μας πρέπει να προσθέσουμε μερικές γραμμές κώδικα. Ο λόγος που απαιτείται ένας τέτοιος συσχετισμός, είναι γιατί σε κάποιο σημείο της εφαρμογής μας μπορεί να χρειαστεί να αλλάξουμε το εμφανιζόμενο κείμενο.

Για να πετύχουμε τη 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). Στη μέθοδο findViewById() βάζουμε το αναγνωριστικό που θέλουμε να συνδέσουμε (που έχουμε ήδη δηλώσει στο view) με ένα αντικείμενο της ίδιας κλάσης. Στο παράδειγμά μας είναι η TextView.
- Να προσθέσουμε το import android.widget.TextView στην κλάση Example.java ώστε να μπορούμε να χρησιμοποιήσουμε την TextView.

2. Με κώδικα (διαδικαστικός τρόπος)

Το ίδιο αποτέλεσμα μπορεί να επιτευχθεί 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 τη μέθοδο setText() που μας παρέχεται. Αυτό το περνάμε σαν παράμετρο στη setContentView() του view που μόλις φτιάξαμε. Εκτελώντας των κώδικα αυτό ως Android Application θα πάρουμε το ίδιο αποτέλεσμα που είδαμε προηγουμένως. Λόγω του ότι δημιουργήσαμε το user interface με κώδικα, δεν χρειάζεται να συνδέσουμε το view με τον κώδικα, αφού η σχέση αυτή υπάρχει ήδη με μέσω του αντικειμένου myTextView. Αν χρειαστεί να σχεδιάσουμε κάτι παραπάνω, όπως για παράδειγμα να προσθέσουμε ένα ή περισσότερα κουμπιά στο user interface που κατασκευάσαμε, τότε πρέπει να χρησιμοποιήσουμε ένα layout το οποίο στον δηλωτικό σχεδιασμό αναγκαστικά το χρησιμοποιούμε εξαρχής.

Σημείωση: Πρέπει να αναφέρουμε ότι και οι δύο τρόποι, ο δηλωτικός με XML και ο διαδικαστικός με κώδικα, είναι σωστοί. Η επιλογή ενός εκ των δύο είναι καθαρά θέμα του

προγραμματιστή. Η Google προτείνει να χρησιμοποιούμε γενικά το δηλωτικό τρόπο εφόσον είναι δυνατόν και έτσι να ξεχωρίζουμε το σχεδιασμό του user interface από τον υπόλοιπο κώδικα της εφαρμογής μας.

4.8 Views (πρότυπες προβολές)

Το Android παρέχει μια εργαλειοθήκη με πρότυπα views για να μας βοηθήσει να δημιουργήσουμε απλά user interfaces. Μερικά από τα πιο γνωστά views είναι:

- **TextView:** Ένα πρότυπο 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 ένα TextView και ένα 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 μια επιλογή μπορεί να είναι ενεργή κάθε φορά.

4.9 Layouts (διατάξεις)

Τα layouts είναι επεκτάσεις της κλάσης ViewGroup και χρησιμοποιούνται για να τοποθετήσουν κατάλληλα στην οθόνη τα στοιχεία που περιέχουν. Κάθε layout χρησιμοποιεί τον δικό του τρόπο τοποθέτησης των στοιχείων και μπορεί να περιέχει άλλα layouts ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα. Το Android SDK περιλαμβάνει ορισμένα απλά layouts για να μας βοηθήσει στην κατασκευή του UI. Μερικά από αυτά είναι:

- **FrameLayout:** Το απλούστερο από τα 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 του καθενός εντός του διαθέσιμου χώρου. Για παράδειγμα σε ένα LinearLayout με οριζόντια διάταξη τοποθετούμε ένα κουμπί με βάρος 0.5 και συνολικό βάρος 1. Αυτό σημαίνει ότι το κουμπί θα καταλάβει το μισό χώρο της γραμμής και θα αφήσει τον υπόλοιπο χώρο για τα υπόλοιπα κουμπιά που θα προσθέσουμε. Για τιμή ίση με 1 απλά θα καταλάβει όλο το χώρο μη δίνοντας σημασία για το χώρο που χρειάζονται τα υπόλοιπα views.

- **TableLayout:** Αυτό το layout μας επιτρέπει να τοποθετήσουμε τα 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 μία οριζόντια κυλιόμενη λίστα.

Τα layouts που μας προσφέρει το SDK μπορεί να μην ικανοποιούν τις ανάγκες μας με αποτέλεσμα να καταφύγουμε σε ένα προσαρμοσμένο layout ή αλλιώς εργαλείο. Για παράδειγμα, αν το layout που θέλουμε να χρησιμοποιήσουμε με επιπλέον μεθόδους είναι ένα LinearLayout, τότε απλά το επεκτείνουμε και προσθέτουμε τις μεθόδους που χρειαζόμαστε.

Αν δεν μας ικανοποιεί κανένα από τα υπάρχοντα layouts μπορούμε να επεκτείνουμε την κλάση 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.

4.10 Υποστήριξη πολλαπλών οθονών

Εξαιτίας των διαφορετικών οθονών των κινητών τηλεφώνων και γενικά των συσκευών που έχουν ως λειτουργικό σύστημα το Android πρέπει να σχεδιάσουμε ανάλογα την εφαρμογή μας. Θα πρέπει είτε να σχεδιαστεί για να λειτουργεί σε συγκεκριμένο μέγεθος οθόνης αποκλείοντας έτσι τα υπόλοιπα μεγέθη, είτε να προνοήσουμε και με κατάλληλους ελέγχους για τον εντοπισμό του μεγέθους και της πυκνότητας των εικονοστοιχείων (pixel) της οθόνης να τρέχει ο κατάλληλος κώδικας. Επειδή κάθε προγραμματιστής θέλει η εφαρμογή του να τρέχει σε όσο το δυνατόν περισσότερες συσκευές παρακάτω εξηγούνται οι βασικότερες έννοιες που πρέπει να γνωρίζουμε πριν μπορέσουμε να πετύχουμε την υποστήριξη πολλαπλών οθονών.

Όροι και έννοιες

- **Μέγεθος οθόνης (screen size):** Για τον καθορισμό του πραγματικού (φυσικού) μεγέθους υπολογίζουμε τη διαγώνιο οθόνης. Για λόγους απλούστευσης το Android χωρίζει όλα τα 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 μία συγκεκριμένη αναλογία αυτών.
- **Πυκνότητα (density):** Με βάση την ανάλυση οθόνης, η ανάλυση είναι η εξάπλωση των pixel σε όλο το πλάτος και το ύψος της οθόνης. Μια οθόνη με χαμηλή πυκνότητα έχει λιγότερα διαθέσιμα pixels εξαπλωμένα σε όλο το εμβαδό της οθόνης. Οθόνες με ίδιο μέγεθος αλλά μεγαλύτερη πυκνότητα έχουν πολλά περισσότερα pixels στην ίδια περιοχή.

  Η πυκνότητα μιας οθόνης είναι πολύ σημαντικό χαρακτηριστικό γιατί ένα στοιχείο του user interface, για παράδειγμα ένα κουμπί, του οποίου το ύψος και το πλάτος ορίζονται από pixels, θα φαίνεται μεγαλύτερο σε μια οθόνη μικρότερης πυκνότητας και μικρότερο σε μια οθόνη μεγαλύτερης πυκνότητας. Για λόγους απλούστευσης το Android χωρίζει όλες τις πραγματικές πυκνότητες οθόνης σε τέσσερις γενικευμένες κατηγορίες: χαμηλή, μεσαία, μεγάλη και πολύ μεγάλη.
- **Pixel ανεξάρτητης πυκνότητας (density-independent pixel (dp)):** Το dp είναι μία εικονική μονάδα μέτρησης pixel. Οι εφαρμογές μπορούν να τη χρησιμοποιούν για να καθορίσουν το user interface τους, δηλαδή να καθορίσουν τις διαστάσεις του layout ή τη θέση με τέτοιο τρόπο που να μην επηρεάζονται από της πυκνότητα της οθόνης.

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

Τρόποι υποστήριξης

Όπως αναφέραμε, το Android χωρίζει το εύρος των πραγματικών υποστηριζόμενων μεγεθών και αναλύσεων οθόνης σε:

- Ένα σύνολο τεσσάρων γενικευμένων μεγεθών: μικρό, κανονικό, μεγάλο και πολύ μεγάλο.

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

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

Πίνακας III : Κατηγοριοποίηση αναλύσεων

- Ένα σύνολο τεσσάρων γενικευμένων πυκνοτήτων: χαμηλή (ldpi), μέτρια (mdpi), υψηλή (hdpi) και πολύ υψηλή (xhdpi).

Στην εικόνα φαίνονται οι τέσσερις κατηγορίες οθόνης καθώς και οι τέσσερις κατηγορίες πυκνότητας. Οι τιμές που περιέχει ο πίνακας είναι ο τύπος και η ανάλυσή 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 κατηγορίας πυκνότητες. Οι εφαρμογές δεν χρειάζεται να συνεργάζονται με το πραγματικό φυσικό μέγεθος ή την πυκνότητα της οθόνης της συσκευής.

Κατά το χρόνο εκτέλεσης (runtime) η πλατφόρμα ανάλογα με την κατηγορία της πυκνότητας ή το μέγεθος της οθόνης, χειρίζεται τη 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 να 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 μέτρια πυκνότητα (mdpi). Απλούστερα, αν έχουμε μια εικόνα και θέλουμε να εμφανίζεται με συγκεκριμένο τρόπο σε συσκευές, ανάλογα με την πυκνότητά τους, τότε αποθηκεύουμε αυτή την εικόνα σε διάφορες αναλύσεις και τοποθετούμε σε κάθε έναν από τους καταλόγους που αντιστοιχεί η ανάλυση. Για παράδειγμα, την εικόνα που θα εμφανίζεται στην υψηλής πυκνότητας οθόνη την τοποθετούμε στο φάκελο drawable-hdpi, την εικόνα που προορίζεται για χαμηλή πυκνότητα στο φάκελο drawable-ldpi κλπ.

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 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, το Android φορτώνει αυτόματα τους κατάλληλους για αυτήν πόρους από την εφαρμογή και τους εμφανίζει χωρίς να τους επεξεργαστεί. Αν δεν ταιριάζουν οι πόροι που είναι διαθέσιμοι για αυτή την οθόνη, τότε αυτόματα φορτώνει τους πόρους που ταιριάζουν καλύτερα σε αυτή με βάση τη γενικευμένη πυκνότητα της τρέχουσας οθόνης.

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

Για παράδειγμα, αν η πυκνότητα της οθόνης είναι υψηλή, φορτώνει τους πόρους που βρίσκονται στο φάκελο με κατάληξη hdpi και τους χρησιμοποιεί χωρίς να τους επεξεργαστεί. Αν δεν υπάρχουν διαθέσιμοι πόροι σε αυτό το 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:** Αν η εφαρμογή δηλώσει ότι δεν υποστηρίζει διαφορετικές πυκνότητες οθόνης, το Android 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.

   Το Android χειρίζεται αυτή την κλιμάκωση χωρίς να το γνωρίζει η εφαρμογή και δηλώνει σε αυτή 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 ανάλυσης 480x800 και περίπου το ίδιο μέγεθος με μια παραδοσιακή HVGA οθόνη, τρέχει μια 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 ορθογωνίου 100x100, το σύστημα θα μετατρέψει αυτόματα με κλιμάκωση τις συντεταγμένες αυτές στην αντίστοιχη ποσότητα, δηλαδή θα το σχεδιάσει με 150x150 pixels ώστε να επιτευχθεί περίπου η ίδια εμφάνιση όπως αν το είχαμε σχεδιάσει για την πυκνότητα αναφοράς. Το ίδιο συμβαίνει και προς την άλλη κατεύθυνση, εφόσον η εφαρμογή εκτελείται σε μα οθόνη χαμηλότερης πυκνότητας, οι συντεταγμένες κλιμακώνονται προς τα κάτω.

Στο αρχείο AndroidManifest.xml έχουμε τη δυνατότητα να δηλώσουμε την ετικέτα <supports-screen> 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 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 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 μια ετικέτα που αξίζει να αναφέρουμε είναι η <uses-sdk> και μπορεί να περιέχει τα γνωρίσματα:

- **minSdkVersion:** Δηλώνουμε το κατώτερο API στο οποίο μπορεί η εφαρμογή μας να τρέξει.
- **targetSdkVersion:** Δηλώνουμε το API για το οποίο σχεδιάστηκε η εφαρμογή μας.
- **maxSdkVersion:** Δηλώνουμε το μεγαλύτερο API στο 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 προαναφερθέντα γνωρίσματα, το σύστημα γνωρίζει αυτόματα ποιες οθόνες υποστηρίζονται διότι ανάλογα με το επιλεγμένο API, το Android γνωρίζει ποιου τύπου οθόνες υποστηρίζονται. Για παράδειγμα, αν στο minSdkVersion επιλέξουμε την έκδοση 9 (ή ανώτερη) του API, τότε αυτόματα η εφαρμογή μας δηλώνει ότι υποστηρίζει και μπορεί να εγκατασταθεί σε οποιαδήποτε οθόνη και πυκνότητα.

Περαιτέρω ανάλυση

Όταν μια εικόνα bitmap έχει φορτωθεί από τους πόρους της εφαρμογής, το Android 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 στον 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.

Το Android 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 περιβάλλον μεσαίας πυκνότητας. Για παράδειγμα, όταν η πυκνότητα συμβατότητας είναι ενεργοποιημένη, αν φορτώσουμε μια εικόνα 76x76 για εμφάνιση σε μια οθόνη υψηλής πυκνότητας, το Android 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 μη 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-nodpi κάτω από τον κατάλογο res και εκεί να τοποθετήσουμε όλους τους πόρους που δεν θέλουμε να κλιμακωθούν.

5. Βάσεις Δεδομένων

Το Android χρησιμοποιεί βάσεις δεδομένων για να αποθηκεύει σημαντικές πληροφορίες οι οποίες δεν πρέπει να χαθούν ακόμα και αν κλείσει η εφαρμογή ή ακόμα και μετά από επανεκκίνηση της συσκευής. Τα δεδομένα αυτά περιλαμβάνουν επαφές, ρυθμίσεις συστήματος, σελιδοδείκτες κλπ.

Στη σημερινή εποχή με την χρήση του internet να είναι όλο και πιο διαδεδομένη, η χρήση τοπικών βάσεων μπορεί να μην φαίνεται σωστή επιλογή λόγω του κινδύνου να χαθούν ευπαθείς πληροφορίες. Παρόλα αυτά, μια τοπική βάση σε μια φορητή συσκευή είναι μια καλή εναλλακτική λύση στον online κόσμο. Σε πολλές περιπτώσεις είναι πιο ασφαλές να κρατάμε τις σημαντικές πληροφορίες σε cloud, αλλά είναι αρκετά σημαντική και η τοπική αποθήκευση για μεγαλύτερη ταχύτητα πρόσβασης στα δεδομένα και για να έχουμε πρόσβαση σε αυτά όταν δεν είναι διαθέσιμη η σύνδεση στο δίκτυο. Σε αυτή την περίπτωση χρησιμοποιούμε την τοπική βάση σαν μνήμη cache.

5.1 SQLite

Η SQLite είναι μια βιβλιοθήκη λογισμικού που υλοποιεί μια αυτοσυντηρούμενη, συναλλακτική SQL μηχανή βάσεων δεδομένων η οποία δεν χρειάζεται server ούτε καμία ρύθμιση παραμέτρων για να λειτουργήσει. Είναι η πιο διαδομένη παγκοσμίως και ο πηγαίος κώδικάς της είναι ελεύθερος να χρησιμοποιηθεί για οποιαδήποτε χρήση.

Αντίθετα με άλλες βάσεις η SQLite δεν απαιτεί τη χρήση ξεχωριστού server αλλά «διαβάζει» και «γράφει» απευθείας σε απλά αρχεία που είναι αποθηκευμένα στο δίσκο. Μια ολοκληρωμένη βάση με τους πίνακές της, τα εναύσματα, τα views κλπ μπορεί να αποθηκευτεί σε ένα και μοναδικό αρχείο. Η μορφή του αρχείου αυτού δουλεύει σε οποιαδήποτε πλατφόρμα (32-bit ή 64-bit).

Η SQLite είναι μια «compact» βιβλιοθήκη. Το υπερβολικά μικρό μέγεθος της βάσης που προκύπτει (350KB με όλες τις λειτουργίες ενεργές) κάνουν την SQLite πολύ δημοφιλή σε μικρές συσκευές (περιορισμένης μνήμης) όπως κινητά τηλέφωνα, PDA και MP3 players. Όσο περισσότερη διαθέσιμη μνήμη υπάρχει στη συσκευή που χρησιμοποιούμε, τόσο πιο γρήγορα τρέχει η SQLite αλλά η απόδοσή της ακόμα και σε περιβάλλον με χαμηλή μνήμη είναι παραπάνω από ικανοποιητική.

Χαρακτηριστικά

- Οι συναλλαγές είναι ατομικές, συνεπείς, απομονωμένες και ανθεκτικές ακόμα και μετά από κατάρρευση του συστήματος ή διακοπές ρεύματος.
- Δεν απαιτείται κανενός είδους διαχείριση ή ρύθμιση παραμέτρων.
- Μια ολοκληρωμένη βάση είναι αποθηκευμένη σε ένα μοναδικό αρχείο.
- Είναι δυνατή η υποστήριξη μέχρι και τεράστιων βάσεων δεδομένων (της τάξης των μερικών terabyte).
- Πολύ μικρό μέγεθος βιβλιοθήκης (λιγότερο από 350KB με όλες τις λειτουργίες ενεργές και λιγότερο από 200KB με παράλειψη των προαιρετικών χαρακτηριστικών).

- Γρηγορότερη στις περισσότερες κοινές λειτουργίες από τις διάσημες μηχανές που χρησιμοποιούν τη λογική client/server.
- Απλή και εύκολη στη χρήση διεπαφή προγραμματισμού (API).
- Προσιτή σαν ένα ANSI-C αρχείο το οποίο μπορεί εύκολα να χρησιμοποιηθεί σε ένα άλλο project.
- Αυτοσυντηρούμενη (χωρίς εξωτερικές εξαρτήσεις).
- Πολλές υποστηριζόμενες πλατφόρμες: Linux, Mac OS-X, Android, iOS και Windows.
- Ελεύθερος πηγαίος κώδικας για κάθε χρήση (εμπορική ή προσωπική).
- Υπάρχει αυτόνομος client για τη διαχείριση των SQLite βάσεων.

Η SQLite στο Android

Η SQLite είναι διαθέσιμη σε κάθε Android συσκευή. Η χρήση της δεν απαιτεί κάποια ρύθμιση εκ των προτέρων. Είναι απαραίτητο μόνο να οριστούν οι δηλώσεις για την δημιουργία και την ενημέρωση της βάσης. Από εκεί και έπειτα η βάση διαχειρίζεται από την πλατφόρμα του Android.

Επειδή η πρόσβαση σε μια SQLite βάση δεδομένων περιλαμβάνει πρόσβαση στο σύστημα αρχείων η διαδικασία γραφής/ανάγνωσης μπορεί να είναι αργή. Για το λόγο αυτό προτείνεται η εκτέλεση των εργασιών στη βάση ασύγχρονα, για παράδειγμα με τη χρήση της AsyncTask κλάσης.

Μια SQLite βάση που δημιουργήθηκε από μια Android εφαρμογή αποθηκεύεται από προεπιλογή στη διαδρομή DATA/data/APP_NAME/databases/FILENAME (με DATA τη διαδρομή όπου επιστρέφει η μέθοδος Environment.getDataDirectory(), APP_NAME το όνομα της εφαρμογής μας και FILENAME το καθορισμένο όνομα της βάσης).

Το framework του Android προσφέρει πολλούς τρόπους για τη χρήση της SQLite εύκολα και αποτελεσματικά. Μεγάλο πλεονέκτημα για τον προγραμματιστή είναι ότι παρόλο που η SQLite χρησιμοποιεί SQL, το Android παρέχει μια υψηλότερου επιπέδου βιβλιοθήκη με ένα περιβάλλον που είναι πολύ ευκολότερο να ενσωματωθεί σε μια εφαρμογή.

Καλό είναι να τονιστεί ότι δεν είναι υποχρεωτική η χρήση της SQLite στο Android. Ο προγραμματιστής μπορεί να χρησιμοποιήσει κάποιο άλλο σύστημα διαχείρισης βάσεων δεδομένων, απλά δεν θα μπορεί να βασιστεί στην υποστήριξη του λειτουργικού. Η SQLite δεν είναι υποκατάστατο ενός ολοκληρωμένου SQL server, απλά είναι εναλλακτική λύση στη χρησιμοποίηση ενός τοπικού αρχείου οποιασδήποτε μορφής.

DbHelper

Το Android παρέχει ένα κομψό περιβάλλον εργασίας ώστε να μπορεί ο προγραμματιστής να κάνει την εφαρμογή του να αλληλεπιδρά με μια SQLite βάση. Για την πρόσβαση στη βάση απαιτείται μια κλάση «βοηθός» που να παρέχει μια σύνδεση με τη βάση, δημιουργώντας τη σύνδεση αν δεν υπάρχει ήδη. Η κλάση αυτή, που παρέχεται από το framework του Android,

ονομάζεται SQLiteOpenHelper. Η database κλάση που επιστρέφεται είναι ένα instance της SQLiteDatabase.

Σημαντικές λειτουργίες

Η κλάση DbHelper προσφέρει ένα υψηλού επιπέδου περιβάλλον το οποίο είναι πιο απλό από αυτό της SQL. Οι προγραμματιστές αντιλήφθηκαν ότι οι περισσότερες εφαρμογές χρησιμοποιούν τις βάσεις δεδομένων μόνο για τις ακόλουθες τέσσερεις ενέργειες, οι οποίες συμβολίζονται με το ακρωνύμιο CRUD: create, read (query), update και delete. Για να ανταπεξέλθει σε αυτές τις απαιτήσεις, η DbHelper προσφέρει τις παρακάτω μεθόδους:

- insert() – Προσθέτει μία ή περισσότερες γραμμές στη βάση
- query() – Ζητάει μία ή περισσότερες σειρές που πληρούν τα δοθέντα κριτήρια
- update() – Αντικαθιστά μία ή περισσότερες σειρές με βάση τα δοθέντα κριτήρια
- delete() – Διαγράφει σειρές με βάση τα δοθέντα κριτήρια

Για τις υπόλοιπες μεθόδους που τυχόν θέλουμε να χρησιμοποιήσουμε πρέπει να ανατρέξουμε απευθείας στην SQLite. Ένα ερώτημα που εγείρεται είναι γιατί να μην χρησιμοποιηθεί SQL αντί για την SQLite. Οι τρεις σημαντικότεροι λόγοι είναι οι εξής:

- Αρχικά από άποψη ασφάλειας, μια SQL δήλωση (statement) είναι κύρια υποψήφια για μια επίθεση ασφάλειας στην εκάστοτε εφαρμογή και κατ' επέκταση στα δεδομένα της, γνωστή και ως επίθεση SQL injection. Αυτό συμβαίνει γιατί μια SQL statement δέχεται είσοδο από το χρήστη και αν δεν είναι σωστά ελεγμένη και απομονωμένη, η εισαγωγή στοιχείων από το χρήστη μπορεί να περιλαμβάνει μια άλλη SQL δήλωση με απροσδόκητα αποτελέσματα.
- Επίσης από πλευράς απόδοσης, η επαναλαμβανόμενη εκτέλεση SQL statements, ρίχνει πάρα πολύ την απόδοση αφού θα πρέπει να γίνει parsing της SQL κάθε φορά που τρέχει μια τέτοια δήλωση.

Τέλος, οι μέθοδοι της DbHelper είναι περισσότερο σταθεροί και είναι λιγότερο πιθανό να περάσουν από τον compiler με λάθη που δεν έχουν εντοπιστεί. Η ενσωμάτωση της SQL σε μια εφαρμογή μπορεί προκαλέσει σφάλματα που εντοπίζονται μόνο κατά τη διάρκεια που τρέχει η εφαρμογή.

Cursors

Ένα query επιστρέφει ένα σετ γραμμών με ένα pointer (δείκτη) ονομαζόμενο κέρσορα (cursor). Μπορούμε να πάρουμε τα αποτελέσματα ένα κάθε φορά από τον κέρσορα, κάνοντάς τον να μετακινείται κάθε φορά στην επόμενη γραμμή. Μπορούμε να μετακινήσουμε τον κέρσορα κατά βούληση στο σετ των αποτελεσμάτων που επιστρέφει το query. Ένας «άδειος» κέρσορας δηλώνει ότι έχουμε ανακτήσει όλες τις γραμμές.

Γενικά, οτιδήποτε κάνουμε με την SQL μπορεί να καταλήξει σε SQL exception (εξαίρεση) γιατί ο κώδικάς της αλληλεπιδρά με κάτι που είναι εκτός του άμεσου ελέγχου μας. Για παράδειγμα,

μπορεί να τελειώσει ο χώρος μια βάσης ή να καταστραφεί με οποιοδήποτε τρόπο. Για το λόγο αυτό είναι καλό να περιβάλουμε τον κώδικα που έχει να κάνει με κλήσεις της βάσης σε μπλοκ try/catch ώστε να χειριζόμαστε τις SQLExceptions.

Ένας εύκολος τρόπος για να γίνει αυτό είναι με συντομεύσεις του Eclipse:

1. Επιλέγουμε τον κώδικα στον οποίο μας ενδιαφέρει να χειριζόμαστε τις εξαιρέσεις.
2. Από το μενού του Eclipse επιλέγουμε Source->Surround With->Try/Catch Block. Το Eclipse θα δημιουργήσει μόνο του τις απαραίτητες try/catch δηλώσεις γύρω από τον κώδικα.
3. Τώρα μπορούμε να χειριστούμε τις τυχόν εξαιρέσεις στο μπλοκ catch. Αυτό μπορεί να γίνει με μία απλή κλήση της Log.e().

6. Web Services

6.1 Ορισμός

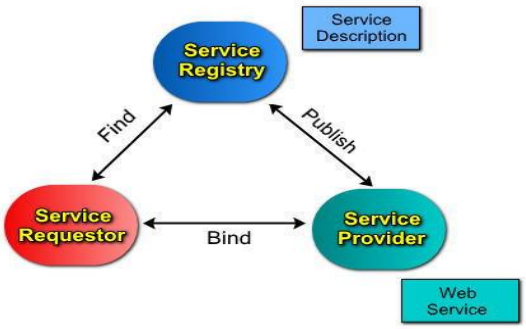
Τα web services είναι μια τεχνολογία που επιτρέπει στις εφαρμογές να επικοινωνούν μεταξύ τους ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού. Ένα web service είναι μια διεπαφή λογισμικού (software interface) που περιγράφει μια συλλογή από λειτουργίες οι οποίες μπορούν να προσεγγιστούν από το δίκτυο μέσω πρότυπων μηνυμάτων XML. Χρησιμοποιεί πρότυπα βασισμένα στη γλώσσα XML για να περιγράψει μία λειτουργία (operation) προς εκτέλεση και τα δεδομένα προς ανταλλαγή με κάποια άλλη εφαρμογή. Μια ομάδα από web services οι οποίες αλληλεπιδρούν μεταξύ τους καθορίζει μια εφαρμογή web services.

6.2 Γενικά

- Οι web services είναι από τα συστατικά των εφαρμογών
- Οι web services επικοινωνούν χρησιμοποιώντας πρωτόκολλα ανοιχτού κώδικα
- Οι web services περιέχουν και περιγράφουν τον εαυτό τους
- Οι web services ανακαλύπτονται με τη χρήση του UDDI
- Οι web services μπορούν να χρησιμοποιηθούν κι από άλλες εφαρμογές
- Η XML είναι η βάση των web services

6.3 Αρχιτεκτονική

Η αρχιτεκτονική των web services βασίζεται στις αλληλεπιδράσεις μεταξύ τριών βασικών ρόλων : του παροχέα υπηρεσίας (service provider), του καταλόγου υπηρεσιών (service registry) και του πελάτη που επιθυμεί μια υπηρεσία (service requestor). Ο πάροχος της υπηρεσίας είναι συνήθως μια εταιρεία που παρέχει πρόσβαση σε μια Web Service και δημοσιεύει την περιγραφή της σε ένα κατάλογο υπηρεσιών. Ο πελάτης βρίσκει την περιγραφή της υπηρεσίας και χρησιμοποιεί τις πληροφορίες της από την περιγραφή για να την δεσμεύσει. Στη παρακάτω εικόνα βλέπουμε ένα σχεδιάγραμμα όπου ο κατάλογος υπηρεσιών παρέχει μια κεντρική τοποθεσία με σκοπό την αποθήκευση των περιγραφών των Web services. Ένας τέτοιας κατάλογος υπηρεσιών είναι κι ο UDDI κατάλογος για παράδειγμα.

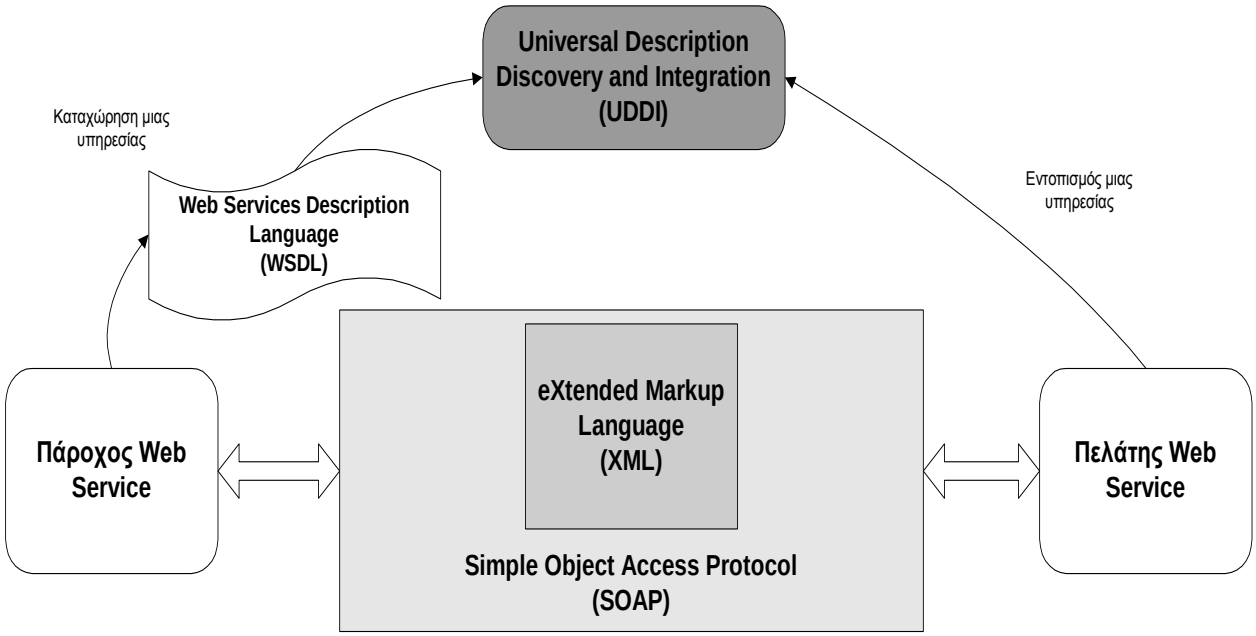


Εικόνα VI : Περιγραφή ενός Web Service

6.4 Τεχνολογίες των Web Services

Οι βασικές τεχνολογίες της πλατφόρμας των Web services είναι :

- **XML** (eXtensible Markup Language) Ομοιόμορφη μορφή και ανταλλαγή δεδομένων
- **SOAP** (Simple Object Access Protocol) Πρότυπο κανάλι επικοινωνίας
- **UDDI** (Universal Description, Discovery and Integration) Καταχώρηση και εντοπισμός των παρεχόμενων υπηρεσιών
- **WSDL** (Web Services Description Language) Πρότυπη περιγραφική γλώσσα για την περιγραφή των παρεχόμενων υπηρεσιών



Εικόνα VII : Τεχνολογίες των Web Services

6.5 XML (eXtensible Markup Language)

Η XML είναι μια γλώσσα ανεξάρτητη από σύστημα και υλικό για την αναπαράσταση δεδομένων και της μορφής τους σε ένα έγγραφο XML. Το έγγραφο αυτό στην πιο απλή του μορφή είναι ένα αρχείο κειμένου το οποίο περιέχει δεδομένα μαζί με σήμανση η οποία καθορίζει τη δομή των δεδομένων. Παρακάτω θα αναφερθούμε στο XML Schema.

Το XML Schema είναι μία προδιαγραφή η οποία μπορεί να ακολουθηθεί όταν δημιουργούμε ένα έγγραφο XML για να εξασφαλίσουμε την ορθότητά του. Τα κύρια χαρακτηριστικά του είναι :

- Χρησιμοποιείται για να καθορίσει έγκυρα στοιχεία και ιδιότητες που μπορούν να χρησιμοποιηθούν σε ένα έγγραφο XML.

- Μπορούμε να καθορίσουμε μια ιεραρχική δομή στοιχείων.
- Μπορεί επίσης να καθοριστεί η διαδοχική οργάνωση μιας συλλογής στοιχείων-παιδιών τα οποία μπορούν να υπάρχουν σε ένα έγγραφο XML.

Κύριοι στόχοι του XML Schema:

- Να μπορέσουν να εκφραστούν μέσα στο πρότυπο αρχές αντικειμενοστραφούς σχεδιασμού οι οποίες μπορούν να βρεθούν σε όλες τις αντικειμενοστραφείς γλώσσες προγραμματισμού.
- Να παρέχεται υποστήριξη για σύνθετους τύπους δεδομένων παρόμοια με την υποστήριξη που υπάρχει στις περισσότερες σχεσιακές βάσεις δεδομένων.

Παράδειγμα ενός XML Schema :

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.books.org" xmlns=http://www.books.org>
        <xsd:element name="BookStore">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Book">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
                                <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
                                <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
                                <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
                                <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name="Title" type="xsd:string"/>
        <xsd:element name="Author" type="xsd:string"/>
        <xsd:element name="Date" type="xsd:string"/>
        <xsd:element name="ISBN" type="xsd:string"/>
        <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Παράδειγμα XML Εγγράφου που υπακούει σε ένα XML Schema :

```
<?xml version="1.0"?>
<BookStore
        xmlns ="http://www.books.org"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.books.org/BookStore.xsd">
        <Book>
                <Title>Web Services Security</Title>
```

```
        <Author>Ravi Trivedi</Author>
        <Date>Dec, 2002</Date>
        <ISBN>1861007655</ISBN>
        <Publisher>Wrox Publishing</Publisher>
    </Book>
</BookStore>
```

6.6 WSDL(Web Services Description Language)

Η WSDL είναι ένα XML Schema για την περιγραφή δικτυακών υπηρεσιών σαν ένα σύνολο από τελικά σημεία που λειτουργούν σε μηνύματα τα οποία περιέχουν πληροφορία είτε προσανατολισμένη στα έγγραφα είτε προσανατολισμένη στις διαδικασίες. Με πιο απλά λόγια η WSDL μας βοηθάει να περιγράψουμε ένα σύνολο από μηνύματα και το πώς αυτά τα μηνύματα ανταλλάσσονται. Η WSDL παρέχει ένα τρόπο στους παροχείς υπηρεσιών να περιγράψουν τη βασική μορφή των αιτήσεων και απαντήσεων των υπηρεσιών πάνω από διαφορετικά πρωτόκολλα και κωδικοποιήσεις.

6.7 UDDI (Universal Description, Discovery and Integration)

- Το UDDI είναι ένας κατάλογος για την αποθήκευση πληροφοριών όσον αφορά τις web services
- Το UDDI είναι επίσης , ένας κατάλογος για τις τεχνικές διεπαφές των web services που περιγράφονται από την WSDL
- Το UDDI επικοινωνεί μέσω του SOAP
- Το UDDI είναι φτιαγμένο πάνω στη πλατφόρμα του Microsoft .NET
- Το UDDI ένα περιβάλλον λογισμικού προσανατολισμένο στις υπηρεσίες τόσο για δημόσια διαθέσιμες υπηρεσίες όσο και για υπηρεσίες που εκτίθενται μόνο εσωτερικά ενός οργανισμού

Οι πληροφορίες που υπάρχουν στο UDDI διακρίνονται σε 3 κατηγορίες :

- *White pages* : Πληροφορίες όπως το όνομα, η διεύθυνση, το τηλέφωνο και άλλες πληροφορίες επικοινωνίας για μία επιχείρηση.
- *Yellow Pages* : Πληροφορίες που κατηγοριοποιούν επιχειρήσεις. Βασίζονται σε υπάρχοντα πρότυπα κατηγοριοποίησης (μη ηλεκτρονικά).
- *Green Pages* : Τεχνικές πληροφορίες για τα web services που παρέχονται από μία επιχείρηση.

Βασικές λειτουργίες του UDDI (βλέπε και εικόνα στο 6.3) :

- *Publish* : Πώς ο προμηθευτής ενός web service καταχωρεί των εαυτό του.
- *Find:* Πώς μία εφαρμογή βρίσκει ένα συγκεκριμένο web service.
- *Bind:* Πώς μία εφαρμογή συνδέεται, και αλληλεπιδρά με ένα web service αφού αυτό βρεθεί.

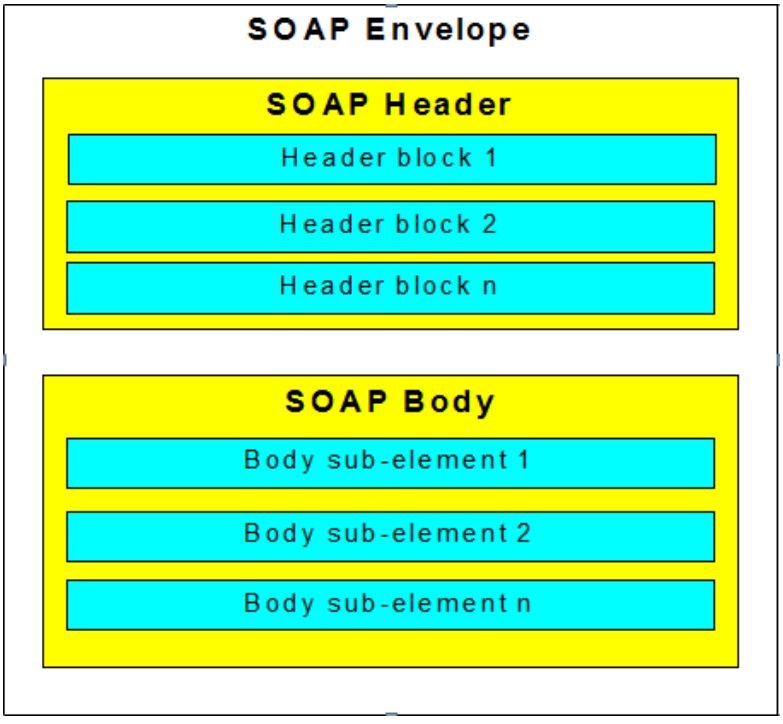
6.8 SOAP(Simple Object Access Protocol)

Το SOAP είναι ένα ελαφρύ πρωτόκολλο προορισμένο για την ανταλλαγή δομημένων πληροφοριών σε ένα αποκεντρωμένο, διανεμημένο περιβάλλον. Χρησιμοποιεί τεχνολογίες XML για να καθορίσει ένα επεκτάσιμο πλαίσιο παρέχοντας μια δομή μηνυμάτων η οποία μπορεί να ανταλλαχθεί πάνω από ποικίλα δικτυακά πρωτόκολλα. Το πλαίσιο έχει σχεδιαστεί να είναι ανεξάρτητο από οποιοδήποτε προγραμματιστικό μοντέλο και σημασιολογία υλοποίησης.

Τα χαρακτηριστικά του είναι :

- Το SOAP είναι ένα πρωτόκολλο επικοινωνίας
- Το SOAP είναι μια δομή για αποστολή μηνυμάτων
- Το SOAP επικοινωνεί μέσω τoy Internet
- Το SOAP είναι ανεξάρτητη πλατφόρμα
- Το SOAP έχει δικιά του ανεξάρτητη γλώσσα
- Το SOAP βασίζεται στην XML
- Το SOAP είναι απλό και επεκτάσιμο
- Το SOAP μπορεί να παρακάμψει firewalls

Δομή ενός μηνύματος SOAP :



Εικόνα VIII : Δομή μηνύματος SOAP

Παράδειγμα SOAP Request :

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope" soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
        <soap:Body xmlns:m="http://www.example.org/stock">
            <m:GetStockPrice>
                <m:StockName>IBM</m:StockName>
            </m:GetStockPrice>
        </soap:Body>
</soap:Envelope>
```

Παράδειγμα SOAP Response :

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-
envelope" soap:encodingStyle=" http://www.w3.org/2001/12/soap-encoding">
    <soap:Body xmlns:m="http://www.example.org/stock">
        <m:GetStockPriceResponse>
        <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>
```

6.9 Παράδειγμα

Στο επόμενο παράδειγμα θα χρησιμοποιήσουμε την ASP.NET για να δημιουργήσουμε μια απλή Web service που μετατρέπει τη θερμοκρασία από Φάρεναϊτ σε Κελσιου κι αντίθετα :

```
<%@ WebService Language="VBScript" Class="TempConvert" %>

Imports System
Imports System.Web.Services

Public Class TempConvert :Inherits WebService

<WebMethod()> Public Function FahrenheitToCelsius
(ByVal Fahrenheit As String) As String
  dim fahr
  fahr=trim(replace(Fahrenheit,",","."))
  if fahr="" or IsNumeric(fahr)=false then return "Error"
  return ((((fahr) - 32) / 9) * 5)
end function

<WebMethod()> Public Function CelsiusToFahrenheit
(ByVal Celsius As String) As String
  dim cel
```

```
  cel=trim(replace(Celsius,",","."))
  if cel="" or IsNumeric(cel)=false then return "Error"
  return ((((cel) * 9) / 5) + 32)
end function

end class
```

Αυτό το αρχείο έχει σωθεί σαν .asmx .Αυτή είναι η επέκταση για τις XML Web Services.

7. Περιβάλλον ανάπτυξης εφαρμογών

7.1 Android SDK

Μέσω του Android Software Development Kit (SDK) προκύπτουν ολοένα και νέες εφαρμογές που δημιουργούνται για το λειτουργικό σύστημα Android. Οι εφαρμογές αναπτύσσονται συνήθως στη γλώσσα προγραμματισμού Java που χρησιμοποιεί το Android. Από τον Απρίλιο του 2011, έχουν αναπτυχθεί για το Android πάνω από 200.000 εφαρμογές , με πάνω από 3 δισεκατομμύρια downloads. Η χρήση της πλατφόρμας Android έχει επίσης αυξηθεί, γιατί προτιμάται από τους προγραμματιστές των κινητών. Μια έρευνα τον Ιούνιο του 2011 έδειξε ότι πάνω από το 67% των κινητών χρησιμοποιούν εφαρμογές που αναπτύχθηκαν με χρήση της εν λόγω πλατφόρμας.

Το Android SDK παρέχει τα απαραίτητα εργαλεία για την ανάπτυξη προγραμμάτων χρησιμοποιώντας την γλώσσα προγραμματισμού Java. Τα χαρακτηριστικά του παρέχουν τεράστια ευελιξία και δυνατότητα ανάπτυξης έξυπνων εφαρμογών για κινητά τηλέφωνα, παρέχοντας δυνατότητες ανάπτυξης σε επιχειρηματίες όλων των κλάδων, είτε πρόκειται για αυτοματοποίηση πωλήσεων, ψυχαγωγία, παιχνίδια αλλά και κάθε άλλου είδους επιχειρήσεις. Ακόμα, παρέχει APIs για την χρήση web browser, εμφάνιση δισδιάστατων και τρισδιάστατων γραφικών, δομημένη αποθήκευση δεδομένων σε βάση δεδομένων, εμφάνιση πολυμεσικού υλικού (ήχος, βίντεο, εικόνες), χρήση των τεχνολογιών GSM, Bluetooth, EDGE, 3G και Wi-Fi, χρήση συσκευών όπως φωτογραφική μηχανή, GPS, πυξίδα, επιταχυνσιόμετρο.

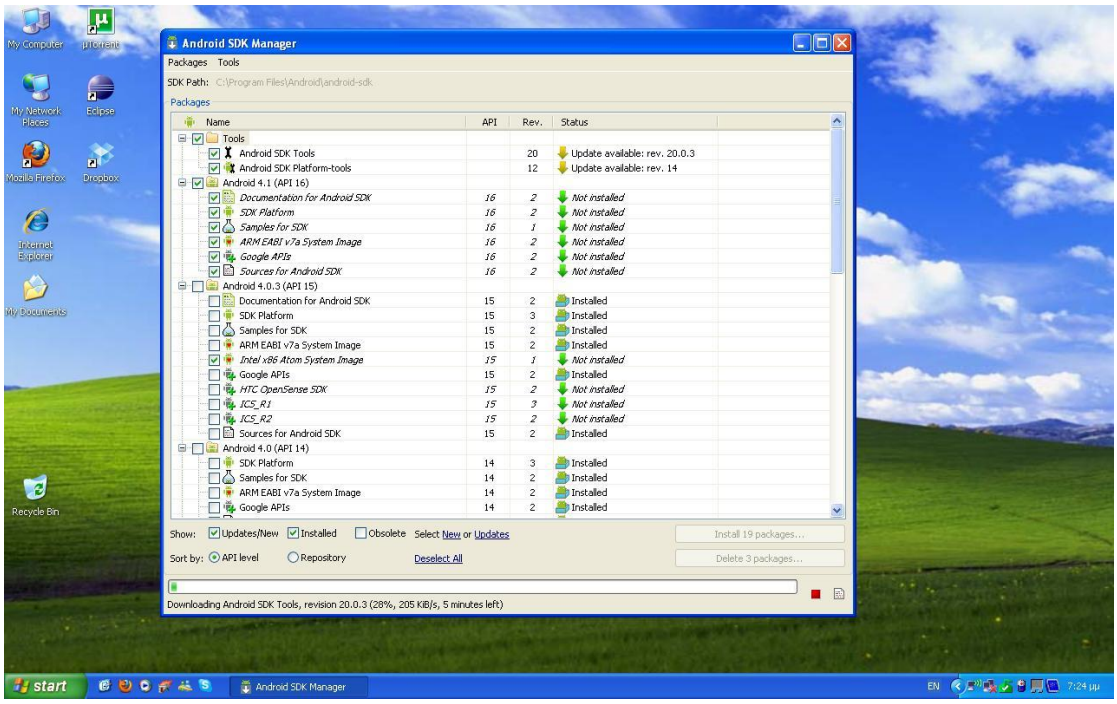
Ένα σημαντικό γεγονός που ευνοεί την ανάπτυξη εφαρμογών είναι πως το πακέτο Android SDK συνεργάζεται με το Eclipse και συνεπώς ο προγραμματιστής μπορεί εύκολα και γρήγορα να βλέπει τις αλλαγές του κώδικα στην εικονική συσκευή (emulator) που του παρέχει το Android SDK, χωρίς να χρειάζεται να εξάγει κάθε φορά την εφαρμογή και να την εγκαθιστά σε κινητό. Επίσης, ο emulator είναι πολύ αξιόπιστος, καθώς έχει ακριβώς την ίδια συμπεριφορά με αυτή που θα είχε η εφαρμογή εάν είχε εγκατασταθεί σε ένα κινητό τηλέφωνο Android. Ένα άλλο σημαντικό πλεονέκτημα είναι το γεγονός πως σε αναβαθμίσεις του λογισμικού, η εφαρμογή εξακολουθεί να δουλεύει χωρίς την ανάγκη επανασχεδιασμού κάποιων σημαντικών κομματιών του κώδικα, που αφορούν την αλληλεπίδραση της εφαρμογής με τα δομικά μέρη (hardware) του κινητού τηλεφώνου.

Χαρακτηριστικά του SDK

Το Android SDK είναι ένα πολύ χρήσιμο εργαλείο για την εξερεύνηση των «ενδοτέρων» του Android. Παρακάτω, φαίνονται τα βασικότερα χαρακτηριστικά του.

- Εφαρμογή πλαισίου (SDK Manager) που επιτρέπει την επαναχρησιμοποίηση και την αντικατάσταση στοιχείων
- Βελτιστοποιημένη Dalvik εικονική μηχανή για κινητές συσκευές
- Ολοκληρωμένο πρόγραμμα περιήγησης, το οποίο βασίζεται στο open source WebKit
- Βελτιστοποιημένα γραφικά, τα οποία τροφοδοτούνται από μια προσαρμοσμένη βιβλιοθήκη 2D και 3D γραφικών με βάση τις προδιαγραφές του OpenGL ES 1.0 (επιτάχυνση hardware προαιρετικά)
- SQLite για δομημένη αποθήκευση δεδομένων

- Υποστήριξη πολυμέσων για αρχεία ήχου, βίντεο, ακόμα και εικόνων (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Τηλεφωνία (εξαρτώμενη από το hardware)
- Bluetooth, EDGE, 3G, Wi-Fi (εξαρτώμενα από το hardware)
- Φωτογραφική μηχανή, GPS, πυξίδα, και επιταχυνσιόμετρο (εξαρτώμενα από το hardware)
- Πλούσιο περιβάλλον ανάπτυξης, συμπεριλαμβανομένου ενός εξομοιωτή συσκευής, εργαλεία για τον εντοπισμό σφαλμάτων, μνήμη και προφίλ απόδοσης όπως επίσης και ένα plugin για το Eclipse IDE



Εικόνα IX : SDK Manager

Ο SDK Manager μας επιτρέπει να κάνουμε update στα εγκατεστημένα στοιχεία.

Εικονική συσκευή Android (emulator)

Προκειμένου να γίνει ευκολότερη η διαδικασία της ανάπτυξης και αποσφαλμάτωσης μιας εφαρμογής, το Android SDK περιλαμβάνει έναν εξομοιωτή μιας εικονικής κινητής συσκευής, η οποία τρέχει το λειτουργικό του Android. Έτσι δεν είναι η αναγκαία η ύπαρξη πραγματικής κινητής συσκευής για την εκτέλεση και δοκιμή των εφαρμογών. Ο εξομοιωτής προσομοιώνει ένα μεγάλο πλήθος λειτουργιών μιας τυπικής συσκευής, η οποία τρέχει το Android:

- Παρέχει μια ποικιλία πλήκτρων πλοήγησης και ελέγχου
- Παρέχει μια οθόνη για την προβολή των εφαρμογών που τρέχουν στον εξομοιωτή
- Επιτρέπει στις εφαρμογές την χρήση των υπηρεσιών που προσφέρει η πλατφόρμα του Android, δηλαδή την κλήση άλλων εφαρμογών, την πρόσβαση στο δίκτυο, την

αναπαραγωγή ήχου και βίντεο, την αποθήκευση και επαναφορά δεδομένων, την ειδοποίηση χρήστη, το γραφικό περιβάλλον του Android

Επίσης παρέχει ένα πλήθος λειτουργιών για την ευκολότερη αποσφαλμάτωση:

- Κονσόλα για την καταγραφή της εξόδου του πυρήνα
- Προσομοίωση διακοπών (όπως η άφιξη SMS μηνύματος ή τηλεφωνικής κλήσης)
- Προσομοίωση καθυστέρησης και απώλειας στο κανάλι δεδομένων
- Προσομοίωση λήψης δεδομένων θέσης από την συσκευή GPS.

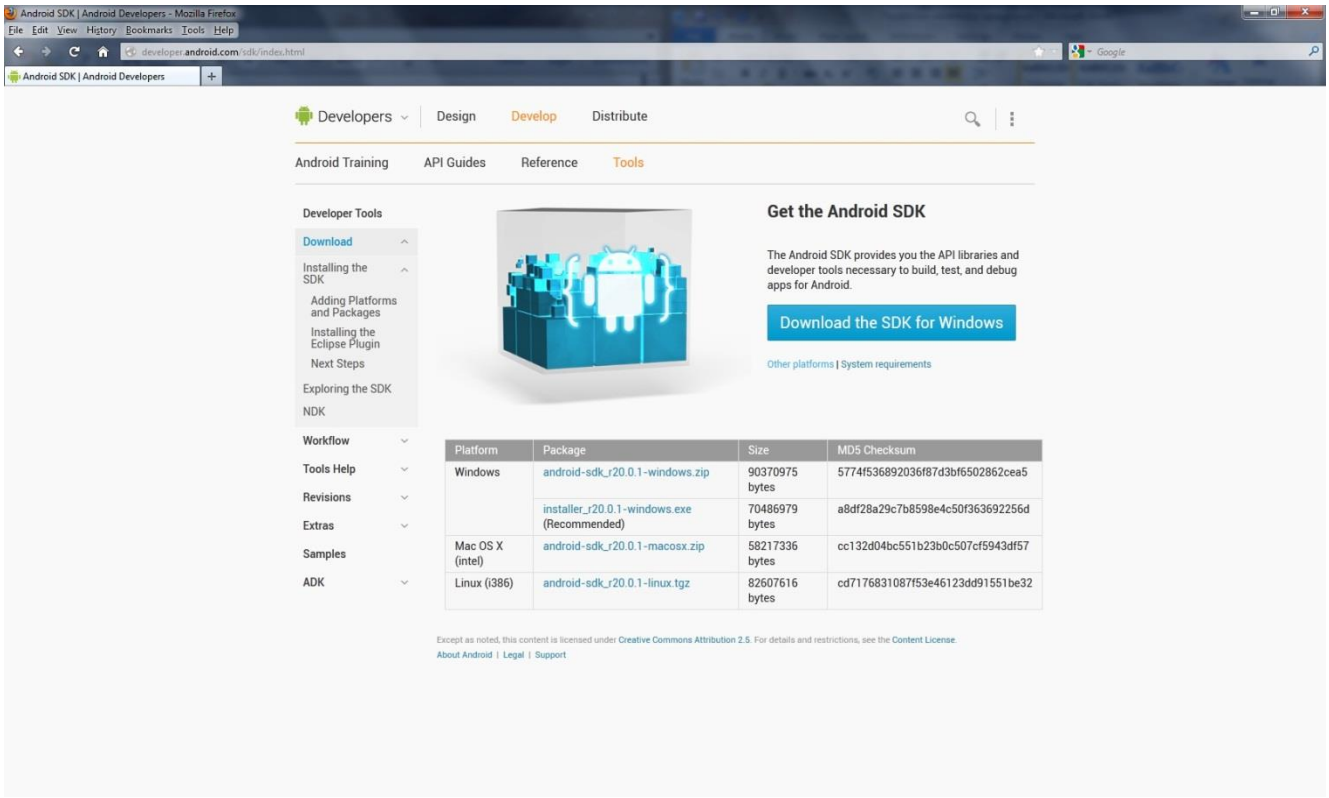
Άλλα εργαλεία του Android SDK

Το Android SDK περιλαμβάνει μερικά ακόμη εργαλεία για την ανάπτυξη εφαρμογών:

- Το Dalvik Debug Monitor Service (DDMS) το οποίο επιτρέπει την διαχείριση των διεργασιών στον εξομοιωτή ή στην συσκευή. Συγκεκριμένα δίνεται η δυνατότητα port-forwarding υπηρεσιών, λήψη screenshots, εμφάνιση πληροφοριών για τον σωρό και τα νήματα, logcat, εμφάνιση πληροφοριών ράδιο και πληροφοριών διεργασιών, προσομοίωση εισερχόμενων κλήσεων και μηνυμάτων, προσομοίωση δεδομένων θέσης κ.α.
- Την Android Debug Bridge (ADB) η οποία επιτρέπει την διαχείριση της κατάστασης του εξομοιωτή ή της συσκευής. Μέσω του ADB είναι δυνατή η εκτέλεση εντολών φλοιού, η διαχείριση της προώθησης θυρών και η αντιγραφή από και προς την συσκευή ή τον εξομοιωτή.
- Το Android Asset Packaging Tool (AAPT) το οποίο δίνει την δυνατότητα δημιουργίας .apk αρχείων τα οποία περιέχουν τα εκτελέσιμα αρχεία και τους πόρους μιας εφαρμογής.
- Την Android Interface Description Language (AIDL) η οποία επιτρέπει την δημιουργία κώδικα που επιτρέπει σε δύο διεργασίες σε μια συσκευή βασισμένη στο Android να συνομιλούν χρησιμοποιώντας διαδιεργασιακή επικοινωνία.
- Το sqlite3 το οποίο επιτρέπει την πρόσβαση στα δεδομένα της SQLite που δημιουργούνται από τις διάφορες εφαρμογές.
- Το Traceview που επιτρέπει την γραφική προβολή της ανάλυσης των trace log data που δημιουργούν οι διάφορες εφαρμογές.
- Το mksdcard το οποίο βοηθά στην δημιουργία εικονικού δίσκου ο οποίος μπορεί να χρησιμοποιηθεί από τον εξομοιωτή για την προσομοίωση της παρουσίας εξωτερικής αποθηκευτικής κάρτας (όπως η SD card).
- Το dx tool το οποίο μετατρέπει τα αρχεία .class από java bytecode σε Android bytecode.
- Το UI/Application Exerciser Monkey το οποίο είναι ένα πρόγραμμα που τρέχει στον εξομοιωτή και παράγει ψευδό-τυχαίες σειρές από συμβάντα χρήστη όπως clicks, touches, gestures καθώς επίσης και έναν αριθμό από συμβάντα συστήματος.
- Το activitycreator το οποίο είναι ένα script που δημιουργεί Ant build αρχεία τα οποία μπορούν να χρησιμοποιηθούν για την μεταγλώττιση των εφαρμογών.

7.1.1 Λήψη και εγκατάσταση

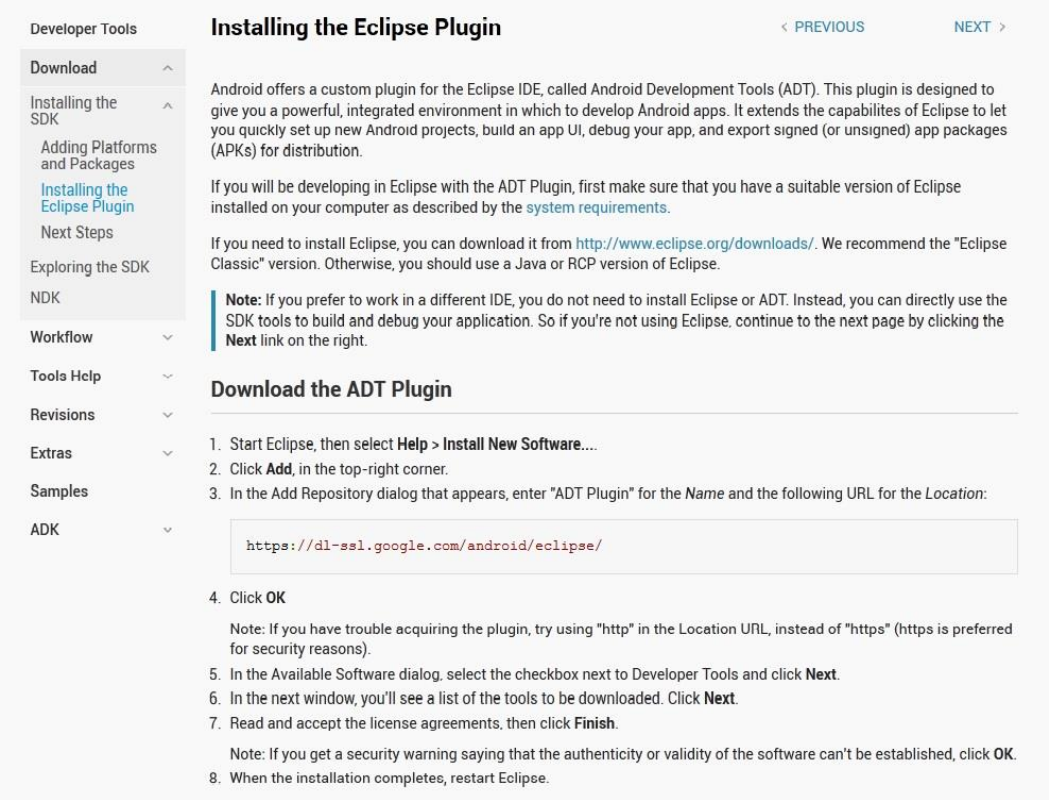
Στην ιστοσελίδα http://developer.android.com/sdk/index.html υπάρχει διαθέσιμο το SDK που παρέχει η Google για την ανάπτυξη εφαρμογών για το Android, τόσο για τα λειτουργικά συστήματα Windows όσο και για MAC OS X και Linux. Επιπλέον, όπως φαίνεται στην εικόνα, παρέχονται πληροφορίες για τα επόμενα βήματα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



Εικόνα X : Λήψη του SDK

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

Λήψη του SDK: Ο προγραμματιστής πρέπει να κατεβάσει το κατάλληλο SDK ανάλογα με το λειτουργικό σύστημα που χρησιμοποιεί (στην περίπτωσή μας Windows).



Εικόνα XI : Λήψη του ADT

Εγκατάσταση του ADT Plugin για το Eclipse: Όπως φαίνεται στην εικόνα, το Android παρέχει ένα Plugin για το Eclipse που ονομάζεται Android Development Tools (ADT). Το Plugin αυτό έχει σκοπό να επεκτείνει τις δυνατότητες του Eclipse ώστε ο προγραμματιστής να μπορεί να δημιουργήσει Android Projects, να κάνει debug τον κώδικά του, να δημιουργήσει γρήγορα το γραφικό περιβάλλον της εφαρμογής, να εξάγει την τελική εφαρμογή ώστε να την μεταφέρει στην κινητή συσκευή κλπ.

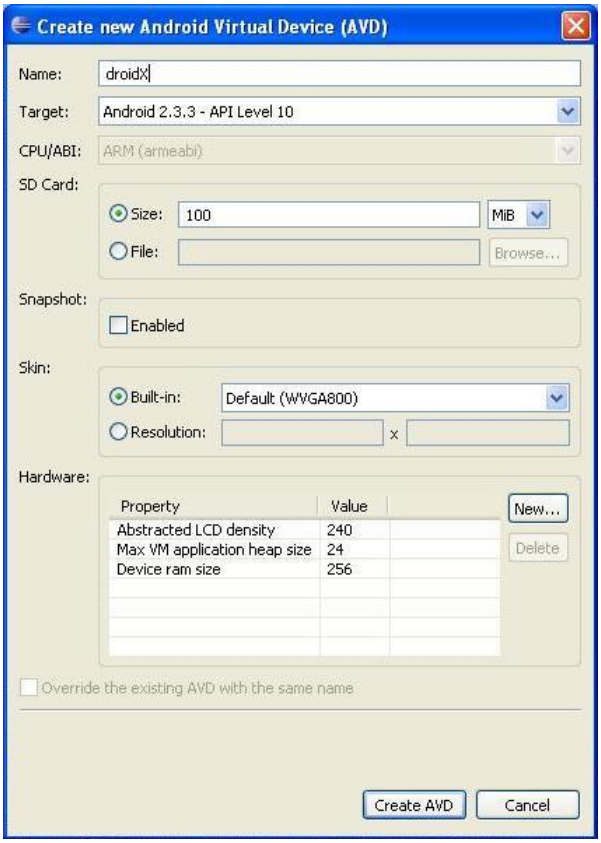
7.1.2 Δημιουργία εικονικής συσκευής Android (Android Virtual Device – AVD)

Μετά τη λήψη και εγκατάσταση των παραπάνω είμαστε έτοιμοι να δημιουργήσουμε την εικονική συσκευή (emulator) στην οποία θα τρέχουμε την εφαρμογή μας. Η δημιουργία της εικονικής συσκευής γίνεται με τα παρακάτω βήματα:

- Από το Eclipse πηγαίνουμε στο μενού Window και επιλέγουμε Android SDK and AVD Manager.
- Στο παράθυρο που έχει ανοίξει πατάμε New…, εισάγουμε τα ζητούμενα στοιχεία και πατάμε Create AVD.

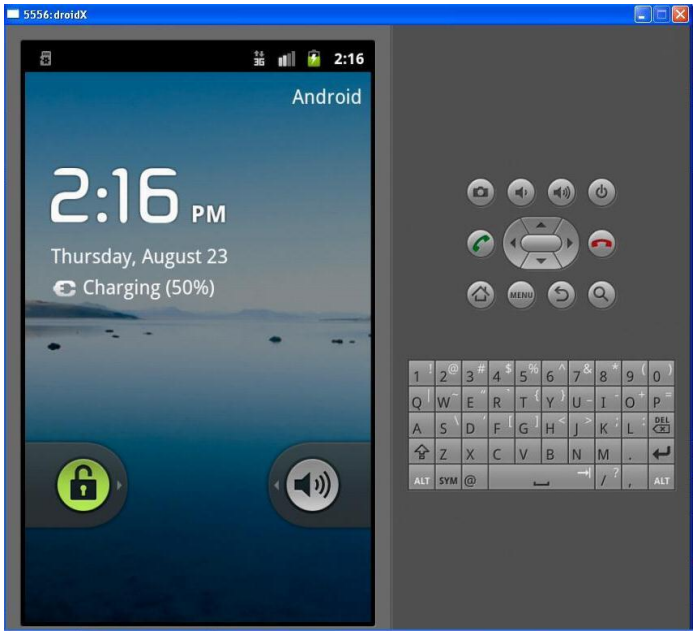
Επεξήγηση των ρυθμίσεων

- Στο πεδίο Name δίνουμε το όνομα που θέλουμε να έχει η εικονική συσκευή, στην προκειμένη περίπτωση droidX.
- Στο πεδίο Target επιλέγουμε την έκδοση του λειτουργικού συστήματος Android που θέλουμε να έχει η συσκευή. Επιλέξαμε την 2.3.3 (Gingerbread).
- Στο πεδίο SD Card βάζουμε τη χωρητικότητα που θέλουμε να έχει η εξωτερική κάρτα μνήμης και είναι προαιρετικό πεδίο.
- Στο πεδίο Skin επιλέγουμε το μέγεθος της οθόνης, είτε από τις υπάρχουσες επιλογές, είτε καθορίζουμε μόνοι μας τις διαστάσεις της στο πεδίο resolution.
- Τέλος, στο πεδίο Hardware μπορούμε να καθορίσουμε διάφορα χαρακτηριστικά που θέλουμε να έχει η συσκευή μας. Αφήσαμε την προκαθορισμένη ιδιότητα για την πυκνότητα (pixel density).

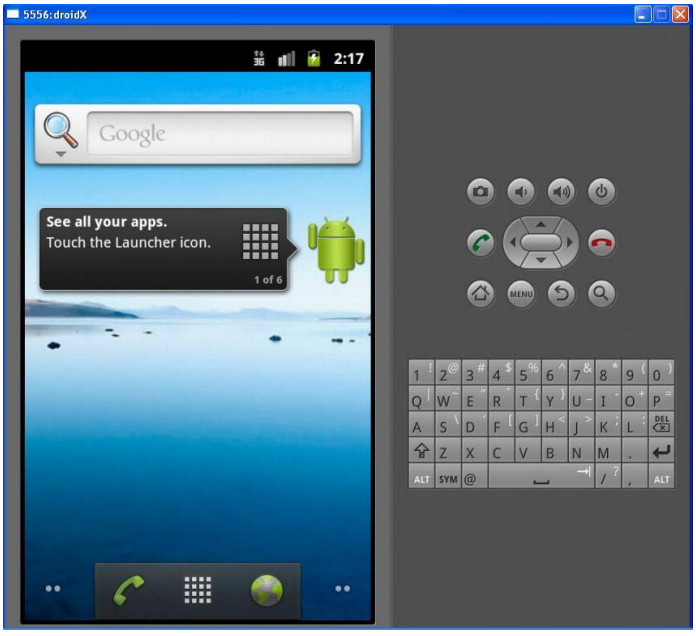


Εικόνα XII : Δημιουργία AVD

Το αποτέλεσμα φαίνεται στις εικόνες:



Εικόνα XIII : Emulator (1)

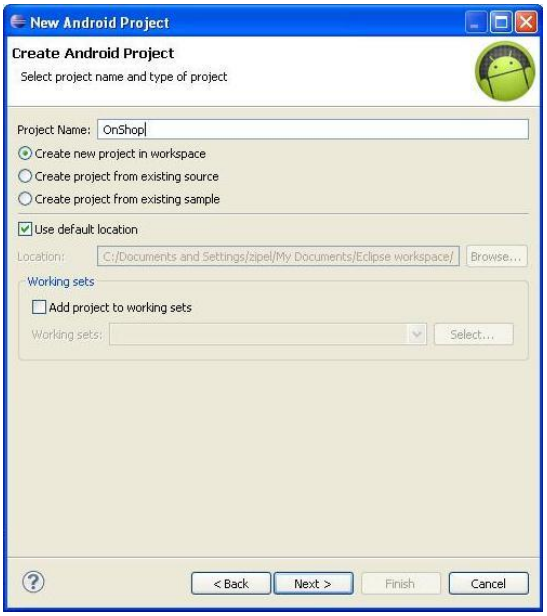


Εικόνα XIV : Emulator (2)

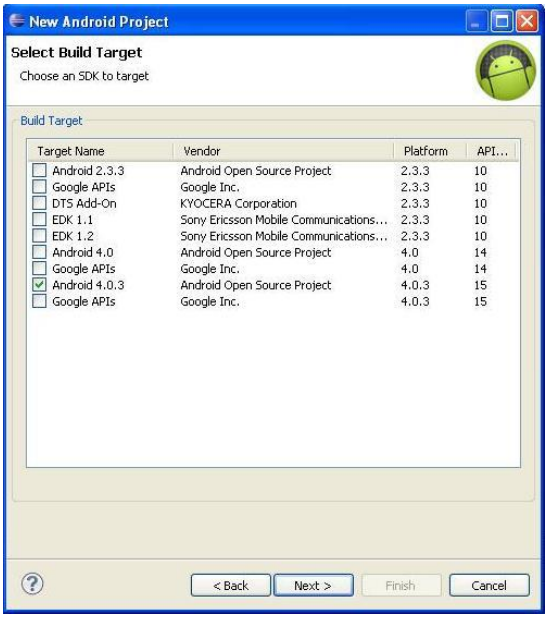
7.1.3 Δημιουργία νέου project

Για να δημιουργήσουμε ένα νέο Android project ακολουθούμε την παρακάτω διαδικασία:

- Επιλέγουμε από το μενού File->New->Project…
- Στο παράθυρο που ανοίγει επιλέγουμε Android->Android Project και πατάμε Next
- Συμπληρώνουμε το όνομα που επιθυμούμε για το project
- Επιλέγουμε την πλατφόρμα (έκδοση λειτουργικού) για την οποία θα σχεδιάσουμε την εφαρμογή μας
- Εισάγουμε το όνομα της εφαρμογής
- Επιλέγουμε το πακέτο (package name) στο οποίο ανήκει η εφαρμογή μας
- Επιλέγουμε να δημιουργηθεί νέα δραστηριότητα (Activity) με όνομα ίδιο με το όνομα της εφαρμογής μας
- Τέλος εισάγουμε τον αριθμό του API που αντιστοιχεί στην πλατφόρμα που επιλέξαμε και πατάμε το Finish



Εικόνα XV : Δημιουργία νέου Project (1)



Εικόνα XVI : Δημιουργία νέου Project (2)

Για την εκτέλεση οποιουδήποτε project στην εικονική συσκευή κάνουμε δεξί κλικ στο project και επιλέγουμε Run As->Android Application. Τότε ξεκινάει η εικονική συσκευή και όταν ολοκληρωθεί η φόρτωσή της εγκαθίσταται η εφαρμογή μας και εκτελείται. Μετά από τα παραπάνω βήματα μπορούμε να ξεκινήσουμε την ανάπτυξη του κώδικα της εφαρμογής.

7.2 Android NDK (Native Development Kit)

Το Native Development Kit (NDK) είναι ένα σύνολο εργαλείων που επιτρέπει την ενσωμάτωση μερών (components) που κάνουν χρήση του ατόφιου κώδικα στις Android εφαρμογές. Οι Android εφαρμογές τρέχουν στην Dalvik virtual machine (εικονική μηχανή Dalvik). Είναι ένα συμπλήρωμα του SDK (Software Development Kit) που παρέχει εργαλεία που επιτρέπουν την ενσωμάτωση ατόφιου (native) κώδικα στις εφαρμογές του Android. Το NDK επιτρέπει την εκτέλεση μερών των εφαρμογών, χρησιμοποιώντας γλώσσες ατόφιου κώδικα, όπως η C και η C++. Αυτό μπορεί να παρέχει πλεονεκτήματα σε συγκεκριμένες τάξεις εφαρμογών με τη μορφή της επαναχρησιμοποίησης του υπάρχοντος κώδικα και σε μερικές περιπτώσεις την αυξημένη ταχύτητα. Η ικανότητα άμεσης κλήσης της λειτουργίας στο OpenGL βελτιώνει την απόδοση γραφικών σε μεγάλες οθόνες κινητών.

Χαρακτηριστικά του NDK

- Είναι ένα εργαλείο που επιτρέπει την ενσωμάτωση components που χρησιμοποιούν C και C++
- Προσφέρει παροχές σε κάποιες κλάσεις εφαρμογών
- Επιτρέπει την επαναχρησιμοποίηση ήδη υπάρχοντος κώδικα
- Αυξάνει την ταχύτητα εκτέλεσης
- Περιέχει τα API, τεκμηρίωση και ενδεικτικές εφαρμογές που βοηθούν στη συγγραφή του ατόφιου (ατομικού) κώδικα.

Συστατικά του NDK

- Ένα σύνολο εργαλείων και αρχείων «χτισίματος» (build files) που χρησιμοποιούνται για να παραχθεί ατόφιος κώδικα σε βιβλιοθήκες από πηγές C και C++
- Έναν τρόπο ενσωμάτωσης των αντίστοιχων ατόφιων βιβλιοθηκών σε ένα αρχείο πακέτου εφαρμογών (.apk), το οποίο μπορεί να αναπτυχθεί σε Android συσκευές
- Ένα σύνολο ατόφιων επικεφαλίδων και βιβλιοθηκών συστήματος (system headers and libraries), τα οποία υποστηρίζονται σε όλες τις μελλοντικές εκδόσεις της Android πλατφόρμας, αρχίζοντας από την έκδοση 1.5. Εφαρμογές που χρησιμοποιούν λειτουργίες ατόφιου κώδικα πρέπει να τρέχουν σε Android 2.3 ή μεταγενέστερα.
- Τεκμηρίωση, δείγματα και tutorials (φροντιστήρια)

Η τελευταία έκδοση του NDK υποστηρίζει τα εξής ARM σύνολα οδηγιών:

- ARMv5TE (που συμπεριλαμβάνει τις Thumb-1 εντολές)
- ARMv7-A (που συμπεριλαμβάνει τις Thumb-2 και VFPv3-D16 εντολές, με προαιρετική υποστήριξη για NEON/VFPv3-D32 εντολές)
- x86 εντολές

Ο κώδικας μηχανής ARMv5TE όπως είναι αναμενόμενο τρέχει σε συσκευές Android που είναι βασισμένες στο ARM. Ο ARMv7-A τρέχει μόνο σε συσκευές όπως: Verizon Droid ή Google Nexus One που έχουν συμβατή CPU. Η κύρια διαφορά ανάμεσα στα δύο σύνολα εντολών είναι ότι ο ARMv7-A εμπεριέχει hardware FPU, Thumbs-2 και εντολές NEON. Μπορεί να γίνει χρήση του ενός ή και των δύο συνόλων εντολών (το ARMv5TE είναι το προεπιλεγμένο), αλλά η αλλαγή στο ARMv7-A είναι τόσο εύκολη, όσο η προσθήκη μιας και μόνο γραμμής στο αρχείο εφαρμογών

Application.mk, χωρίς να χρειαστεί να αλλάξει οτιδήποτε άλλο στο αρχείο. Επίσης μπορεί να γίνει build και τις δύο αρχιτεκτονικές ταυτόχρονα και αποθήκευση όλων στο τελικό .apk αρχείο. Ολοκληρωμένες πληροφορίες παρέχονται στο CPU-ARCH-ABIS.HTML στο πακέτο NDK. Το NDK παρέχει σταθερές επικεφαλίδες (headers) για τη libc (βιβλιοθήκη της C), libm (βιβλιοθήκη της Math), OpenGL ES (βιβλιοθήκη 3D γραφικών), τη διεπαφή JNI και άλλες βιβλιοθήκες.

Πότε γίνεται ανάπτυξη σε ατόφιο κώδικα

Το NDK δεν θα ωφελήσει στις περισσότερες εφαρμογές. Ένας developer πρέπει να ζυγίσει τα πλεονεκτήματα έναντι των μειονεκτημάτων. Ειδικά, η χρήση ατόφιου κώδικα δεν έχει αποτέλεσμα στην αύξηση της αυτόματης απόδοσης, αλλά πάντα αυξάνει την πολυπλοκότητα της εφαρμογής. Γενικά, η χρήση ατόφιου κώδικα γίνεται μόνο αν είναι απαραίτητο για την εφαρμογή και όχι για λόγους προτιμήσεων, όπως για παράδειγμα προγραμματισμός σε C/ C++.

Τυπικά καλές υποψήφιες λειτουργίες για το NDK είναι αυτές που είναι αυτοπεριοριζόμενες και που κάνουν εντατική χρήση της CPU και που παρόλα αυτά δεν δεσμεύουν πολλή μνήμη, όπως είναι η επεξεργασία σήματος, η εξομοίωση φυσικής κτλ. Η απλή επανακωδικοποίηση μιας μεθόδου σε C, συνήθως δεν φέρνει αποτέλεσμα μεγάλης αύξησης στην απόδοση. Κατά την εξέταση του αν πρέπει ή όχι ο προγραμματισμός πρέπει να γίνει με ατόφιο κώδικα, είναι απαραίτητη η επεξεργασία των απαιτήσεων και το αν το API του Android framework παρέχει τη λειτουργικότητα που απαιτείται. Παρόλα αυτά το NDK μπορεί να είναι ένας αποτελεσματικός τρόπος επαναχρησιμοποίησης ενός μεγάλου μέρους του υπάρχοντος κώδικα σε C/C++.

Το Android framework παρέχει δυο τρόπους χρήσης του ατόφιου κώδικα:

- Συγγραφή της εφαρμογής χρησιμοποιώντας το Android framework και χρήση του JNI για την προσπέλαση των API που παρέχονται από το Android NDK. Αυτή η τεχνική επιτρέπει την εκμετάλλευση της ευκολίας του Android framework, αλλά επιτρέπει ακόμα και την συγγραφή ατόφιου κώδικα όταν χρειάζεται. Είναι δυνατή η εγκατάσταση εφαρμογών που χρησιμοποιούν ατόφιο κώδικα μέσω του JNI σε συσκευές που τρέχουν Android 1.5 ή μεταγενέστερο.
- Συγγραφή μιας ατόφιας μεθόδου, η οποία επιτρέπει την υλοποίηση των lifecycle callbacks με ατομικό ατόφιο κώδικα. Το Android SDK παρέχει την NativeActivity κλάση, που είναι κλάση που διευκολύνει τον προγραμματιστή, αφού ειδοποιεί τον κώδικά για κάθε δραστηριότητα lifecycle callbacks (onCreate (), onPause (), onResume (), κτλ). Η υλοποίηση των callbacks στον κώδικα μπορεί να γίνει με τέτοιο τρόπο που να χειρίζονται αυτά τα γεγονότα, όταν προκύπτουν. Εφαρμογές που χρησιμοποιούν ατόφιες μεθόδους πρέπει να τρέχουν σε Android 2.5 ή μεταγενέστερο.
- Δεν είναι εφικτές οι προσπελάσεις χαρακτηριστικών όπως Services και Content Providers με ατομικό κώδικα, οπότε αν η χρήση αυτών ή οποιοδήποτε άλλων API frameworks κρίνεται απαραίτητη, θα πρέπει να γίνει συγγραφή κώδικα JNI ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα.

7.3 App Inventor for Android

Το App Inventor for Android αποτελεί ένα νέο, δωρεάν οπτικό περιβάλλον προγραμματισμού με πλακίδια (blocks), για τη δημιουργία εφαρμογών για κινητά τηλέφωνα με λειτουργικό σύστημα Android. Δημιουργήθηκε στο MIT αλλά υποστηρίζεται από την Google και από τον Ιανουάριο του 2012 είναι open-source.

Οι συγκεκριμένες εφαρμογές τρέχουν και σε emulator. Αναπτύχθηκε στα εργαστήρια της Google από μια ομάδα με επικεφαλής τον καθηγητή του MIT Hal Abelson (Abelson, 2009). Το App Inventor χρησιμοποιείται και δοκιμάζεται ήδη ως πλατφόρμα διδασκαλίας και εισαγωγής στον προγραμματισμό, τόσο στην τριτοβάθμια, όσο και στη σχολική εκπαίδευση. Το περιβάλλον του App Inventor έχει πολλές ομοιότητες με το περιβάλλον του Scratch και του Alice, με τη διαφορά ότι οι εφαρμογές που δημιουργούνται τρέχουν σε έξυπνα τηλέφωνα (smart phones). Σε πρόσφατη έρευνα η οποία πραγματοποιήθηκε από την οργάνωση Ανθρωπιστική δράση με Δωρεάν Λογισμικό Ανοικτού Κώδικα (Humanitarian Free and Open Source Software - HFOSS) και χρηματοδοτήθηκε από το Εθνικό Ίδρυμα Επιστημών των Ηνωμένων Πολιτειών (National Science Foundation) έγινε προσπάθεια να απαντηθεί το ερώτημα αν το App Inventor είναι κατάλληλο για τη διδασκαλία του προγραμματισμού και ειδικότερα της υπολογιστικής σκέψης στη σχολική εκπαίδευση.

Πλεονεκτήματα

Τα πρώτα αποτελέσματα είναι ενθαρρυντικά αφού οι ερευνητές επισημαίνουν τα εξής πλεονεκτήματα:

- Εύκολο στη χρήση περιβάλλον με πολλές δυνατότητες
- Αντικειμενοστραφές μοντέλο οπτικού προγραμματισμού με δομές ελέγχου καθοδηγούμενες από γεγονότα (event-driven)
- Μάθηση μέσω της λύσης προβλημάτων
- Επιπλέον κίνητρα στους μμαθητές σε σχέση με το Scratch και Alice, εξαιτίας της φορητότητας και της πρακτικής χρήσης των εφαρμογών που δημιουργούνται
- Ύπαρξη emulator που σημαίνει ότι δεν χρειάζονται πολλές συσκευές για την εισαγωγή στη σχολική τάξη
- Υποστήριξη από τη Google

Η διαφορά σε σχέση με το Android SDK είναι ότι δεν χρειάζονται προγραμματιστικές γνώσεις ή εμπειρία για την δημιουργία του. Σύμφωνα με την Google, ο καθένας μπορεί να αναπτύξει εφαρμογές με το App Inventor, από τους επαγγελματίες προγραμματιστές έως και τους μαθητές, στο επίπεδο μίας σχολικής αίθουσας.

Τι είναι το App Inventor

Το App Inventor (Εφευρέτης Εφαρμογών) είναι μια What You See Is What You Get ("Αυτό που βλέπεις, είναι αυτό παίρνεις") λύση της Google, που επιτρέπει τη δημιουργία εφαρμογών για το Android, μέσω απλού drag'n'drop. Θεωρητικά, ακόμη και ένας αρχάριος με μηδενικές γνώσεις προγραμματισμού, μπορεί μέσω του App Inventor μέσα σε λιγότερο από μια ώρα να έχει δημιουργήσει τη δική του εφαρμογή για το Android. Φυσικά δεν χρειάζεται καν να έχει κάποιος ένα smartphone ή tablet με αυτό το λειτουργικό σύστημα, αφού μπορεί να τρέξει την εφαρμογή του σε emulator. Όπως χαρακτηριστικά αναφέρει η Google, «...για πολλούς ανθρώπους, η κινητή τους συσκευή και η πρόσβαση που επιτυγχάνεται στο διαδίκτυο μέσω αυτής, έχει διάφορους περιορισμούς. Το App Inventor για Android δίνει σε όλους, ανεξαρτήτως προγραμματιστικών ικανοτήτων, την ευκαιρία να ελέγχουν και να μετασχηματίζουν την εμπειρία της επικοινωνίας τους. Παρατηρούμε μάλιστα πως οι άνθρωποι αισθάνονται υπερήφανοι με το να γίνονται δημιουργοί της κινητής τεχνολογίας και όχι απλοί χρήστες της».

Με το App Inventor, οι χρήστες μπορούν να δημιουργήσουν πρακτικά όποια εφαρμογή θέλουν: από ένα απλό παιχνίδι έως εφαρμογές που θα χρησιμοποιούν τους αισθητήρες της συσκευής (εγγύτητας, προσανατολισμού κλπ). Επιπλέον, είναι δυνατή και η δημιουργία εφαρμογών εκμάθησης, όπως η δημιουργία ενός κουΐζ ερωτήσεων ή η εκμετάλλευση της τεχνολογίας text-to-speech για εκφώνηση διαφόρων εντολών από τη συσκευή.

Όπως ήδη αναφέρθηκε, για την χρήση της εφαρμογής δεν είναι απαραίτητες οι γνώσεις προγραμματισμού, μιας και αντί για τη συγγραφή κώδικα ο δημιουργός αποφασίζει για την εμφάνιση και τη συμπεριφορά της εφαρμογής μέσα από ένα καλαίσθητο παραθυρικό περιβάλλον. Μάλιστα, οι δημιουργοί της πρόσθεσαν διάφορα blocks εντολών, τα οποία «ενεργοποιούνται» πατώντας το αντίστοιχο πλήκτρο και εκτελούν προκαθορισμένες λειτουργίες. Για παράδειγμα, υπάρχουν blocks επανάληψης εντολών τύπου for, do-while κλπ, blocks αποθήκευσης πληροφοριών, καθώς και blocks κώδικα για την εκτέλεση λειτουργιών κάτω από συγκεκριμένες συνθήκες. Επίσης, είναι διαθέσιμα ακόμη και blocks για επικοινωνία με τοποθεσίες κοινωνικής δικτύωσης, όπως το Twitter.

Άλλες ενδιαφέρουσες πτυχές του App Inventor είναι η συνεργασία με τους δέκτες GPS, ώστε να «χτιστούν» εφαρμογές που θα χρησιμοποιούν πληροφορίες εντοπισμού θέσης. Ακόμη υπάρχουν αρκετές άλλες εφαρμογές, ίσως πιο πρακτικής φύσεως, οι οποίες εκμεταλλεύονται ή υποβοηθούν τις καθημερινές λειτουργίες ενός κινητού τηλεφώνου.

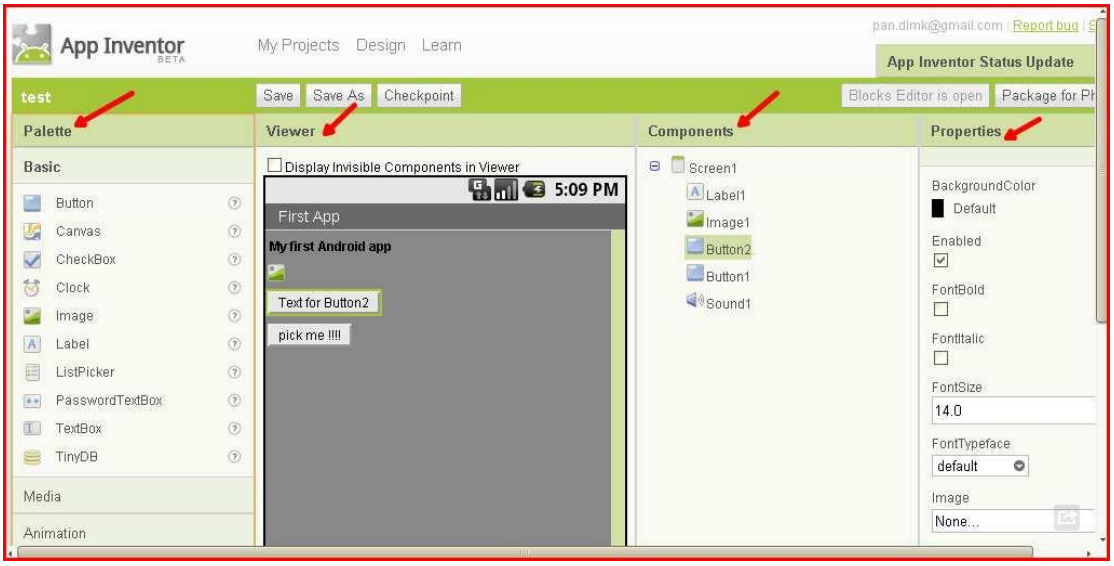
Εάν κάποιος θέλει να φτιάξει τη δική του Android εφαρμογή χωρίς να αναλωθεί στην συγγραφή εκατοντάδων γραμμών κώδικα, η Google προτείνει το εύχρηστο περιβάλλον ανάπτυξης εφαρμογών του App Inventor, για δημιουργία εφαρμογών από τον καθένα. Πρόκειται για μια πληρέστατη συλλογή εργαλείων που επιτρέπουν σε οποιονδήποτε, δίχως προγραμματιστικό υπόβαθρο, να δημιουργήσει προγράμματα για τα κινητά που τρέχουν Android. Οι λειτουργίες του τηλεφώνου απεικονίζονται ως χρωματιστά τουβλάκια, τα οποία οι χρήστες μπορούν να σέρνουν και να συνδέουν πάνω στην οθόνη του υπολογιστή τους. Οι μαθητευόμενοι προγραμματιστές μπορούν έτσι να δημιουργούν παζλ και άλλα παιχνίδια ή προγράμματα που αξιοποιούν ταυτόχρονα πολλές επιμέρους δυνατότητες του Android.

Για παράδειγμα, ένας από τους εθελοντές που δοκίμασαν το App Inventor ανέπτυξε μια εφαρμογή που ενημερώνει τους φίλους του για το πού ακριβώς βρίσκεται μέσω της λειτουργίας GPS. Ακόμα και μαθητές δημοτικού μπόρεσαν να αξιοποιήσουν τα νέα εργαλεία στις δοκιμές που

διεξήγαγε η Google εδώ και ενάμιση χρόνο. Το App Inventor βασίστηκε στο Open Blocks, ένα σύστημα προγραμματισμού που αναπτύχθηκε για εκπαιδευτικούς λόγους στο MIT. Το σύστημα περιλαμβάνει μια βιβλιοθήκη κώδικα Java και τη γλώσσα προγραμματισμού Scratch. Η συγκεκριμένη πρόταση αναμένεται να οδηγήσει σε αύξηση των διαθέσιμων εφαρμογών για το Android.

Να σημειώσουμε πως το App Inventor δεν μπορεί βέβαια να δημιουργήσει με λίγα κλικ λογισμικό επαγγελματικού επιπέδου που θα χρειαζόταν χιλιάδες γραμμές κώδικα, αλλά είναι σίγουρα μια καλή αρχή για όποιον θέλει να εξοικειωθεί με τον προγραμματισμό για κινητά, ενώ μπορεί κάλλιστα να χρησιμοποιηθεί και για "prototyping", δηλαδή για "γρήγορο στήσιμο προσχεδίων εφαρμογών" τις οποίες, στη συνέχεια, μπορεί κανείς να συνεχίσει να αναπτύσσει με πρόσθετο κώδικα. Πλέον είναι αρκετά εύκολη η δημιουργία εφαρμογών για το κινητό.

Η Google με το App Inventor, δεν μπορούσε να κάνει τα πράγματα πιο εύκολα για όλους αυτούς που δε θέλουν να ακολουθήσουν τον δρόμο της Java, για να φτιάξουν μια απλή εφαρμογή. Ο χρήστης απλά ενώνει blocks με κώδικα, έτσι ώστε να σχηματιστεί η λογική του προγράμματος που θέλει να φτιάξει. Είναι ο προγραμματισμός στη πιο απλή και κατανοητή μορφή του. Όλες οι δυνατότητες είναι μπροστά του και ο χρήστης απλά καλείται να επιλέξει και να ενώσει τα κομμάτια που τον ενδιαφέρουν. Κάτι τέτοιο συμβαίνει στο μυαλό του κάθε προγραμματιστή όταν φτιάχνει ένα πρόγραμμα, αλλά χάρις σε αυτό το εργαλείο, αυτή η διαδικασία γίνεται στην οθόνη του καθενός.



Εικόνα XVII : App Inventor

Αυτός ο τρόπος προγραμματισμού δεν είναι καινούργιος. Παραπάνω, είχε αναφερθεί το Scratch, μια γλώσσα προγραμματισμού για αρχάριους, που επίσης χρησιμοποιεί blocks κώδικα. Εκνευριστικός τρόπος για να προγραμματίζει ένας προγραμματιστής, αλλά εύκολος για κάποιον αρχάριο.

Όπως στα περισσότερα εργαλεία ανάπτυξης εφαρμογών με γραφικό περιβάλλον, πρώτα σχεδιάζουμε το γραφικό περιβάλλον και μετά τον κώδικα που ανταποκρίνεται στο κάθε συμβάν. Η οθόνη χωρίζεται στην palette, όπου βρίσκονται όλα τα διαθέσιμα controls που μπορεί να έχει η εφαρμογή μας, στον viewer, που αντιπροσωπεύει την εμφάνιση της εφαρμογής μας, τα components που είναι μια λίστα με τα controls που χρησιμοποιούμε και τέλος τα properties, που είναι οι ιδιότητες του κάθε control.

Μόλις φτιάξουμε το γραφικό περιβάλλον που επιθυμούμε, τότε καλείται ο blocks editor, μια java εφαρμογή που μας επιτρέπει να "κολλήσουμε" μαζί blocks με κώδικα. Το κάθε control έχει τα δικά του blocks κώδικα, ανάλογα με την λειτουργία του. Έτσι με ευκολία έχουμε πρόσβαση στις κυριότερες λειτουργίες της συσκευής μας.

Η δοκιμή της εφαρμογής μπορεί να γίνει είτε στον εξομοιωτή που υπάρχει, είτε απευθείας στη συσκευή. Συγκριτικά, η καλύτερη λύση είναι οι δοκιμές να γίνονται σε μια πραγματική συσκευή, καθώς ο emulator είναι κάπως αργός.

Συνοψίζοντας, το App Inventor είναι μια πολύ έξυπνη ιδέα μιας και δίνει την ευκαιρία σε όποιον ενδιαφέρεται αλλά δεν έχει τις απαραίτητες γνώσεις, να φτιάξει μια εφαρμογή για Android. Είναι ουσιαστικά το μοναδικό δωρεάν εργαλείο για εύκολη ανάπτυξη εφαρμογών. Το μόνο αρνητικό χαρακτηριστικό του είναι ότι, κατά τον προγραμματισμό, η χρήση των blocks είναι μια αργή διαδικασία.

7.4 Εντοπισμός σφαλμάτων (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 αναλυθούν αυτά τα εργαλεία παράλληλα με τη διευκόλυνση που μας παρέχει η χρήση του Eclipse. Για να λάβουμε πληροφορίες σχετικά με τον εντοπισμό σφαλμάτων από την εφαρμογή μας, πρέπει να προσθέσουμε στο AndroidManifest.xml στην ετικέτα <application> την ιδιότητα android:debuggable=”true”.

7.4.1 Android Debug Bridge (adb)

Το εργαλείο Android Debug Bridge (adb), που μας παρέχεται μαζί με το SDK και βρίσκεται στον κατάλογο <sdk >/platform-tools/, μας δίνει ένα τρόπο για να διαχειριστούμε την κατάσταση του εξομοιωτή ή μιας συνδεδεμένης με usb συσκευής Android. Το adb αποτελείται από τρεις συνιστώσες:

1. **Ένα πρόγραμμα πελάτη (client)** που τρέχει στον υπολογιστή στον οποίο 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 μπορούν επίσης να δημιουργήσουν πελάτες adb.
2. **Ένα διακομιστή (server)** ο οποίος τρέχει ως διαδικασία υπόβαθρου (background process) στον υπολογιστή που αναπτύσσουμε την εφαρμογή. Ο διακομιστής διαχειρίζεται την επικοινωνία μεταξύ του πελάτη και του adb daemon που εκτελείται σε έναν εξομοιωτή ή συσκευή.
3. **Ένα δαίμονα (daemon)** ο οποίος τρέχει ως background process για κάθε εξομοιωτή ή συσκευή.

Όταν ξεκινάμε έναν adb client, ελέγχεται πρώτα κατά πόσον υπάρχει μια διαδικασία adb server 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 clients. Όλοι οι clients χρησιμοποιούν τη θύρα 5037 για να επικοινωνήσουν με τον server 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, που είναι το φάσμα που χρησιμοποιείται από τις εικονικές μηχανές/συσκευές.

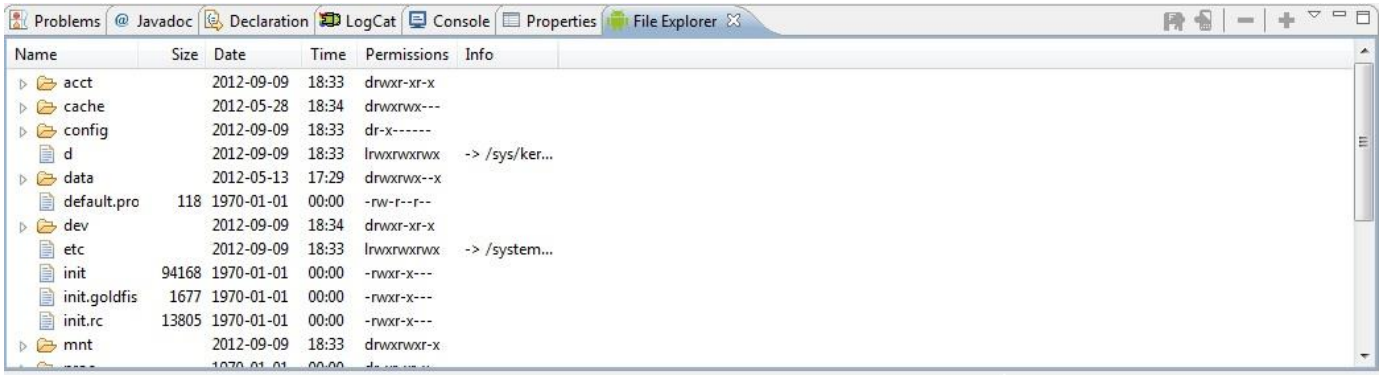
Όταν ο server βρει έναν adb daemon, δημιουργεί μια σύνδεση με αυτή την πόρτα (port). Μετά από αυτή τη διαδικασία είμαστε έτοιμοι και μπορούμε πλέον να χρησιμοποιήσουμε εντολές adb για τον έλεγχο και τη διαχείριση αυτών των εικονικών μηχανών/συσκευών.

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 server.
- **Η εντολή adb install <path_to_apk>** εγκαθιστά το apk δηλαδή την εφαρμογή στη συσκευή.
- **Η εντολή adb pull <remote><local>** αντιγράφει ένα αρχείο από τη συσκευή στον υπολογιστή μας.
- **Η εντολή adb push <local><remote>** αντιγράφει ένα αρχείο από τον υπολογιστή μας στη συσκευή.

Το adb έχει πληθώρα εντολών και παραμέτρων που μπορούμε να κάνουμε χρήση. Χρησιμοποιώντας το Eclipse και το ADT plugin μας διευκολύνει με την παραθυρική του εμφάνιση και αυτοματοποιεί πολλές διαδικασίες που χρειάζεται να γίνουν ώστε να διαχειριζόμαστε τη συσκευή.

Ένα απλό παράδειγμα είναι οι εντολές αντιγραφής αρχείου, οι οποίες μπορούν να γίνουν πολύ πιο εύκολα από το παράθυρο File Explorer. Το παράθυρο αυτό μπορούμε να το εμφανίσουμε από το μενού Window->Show View->Other και να επιλέξουμε File Explorer. Θα εμφανιστεί το παρακάτω 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 XVIII : File Explorer

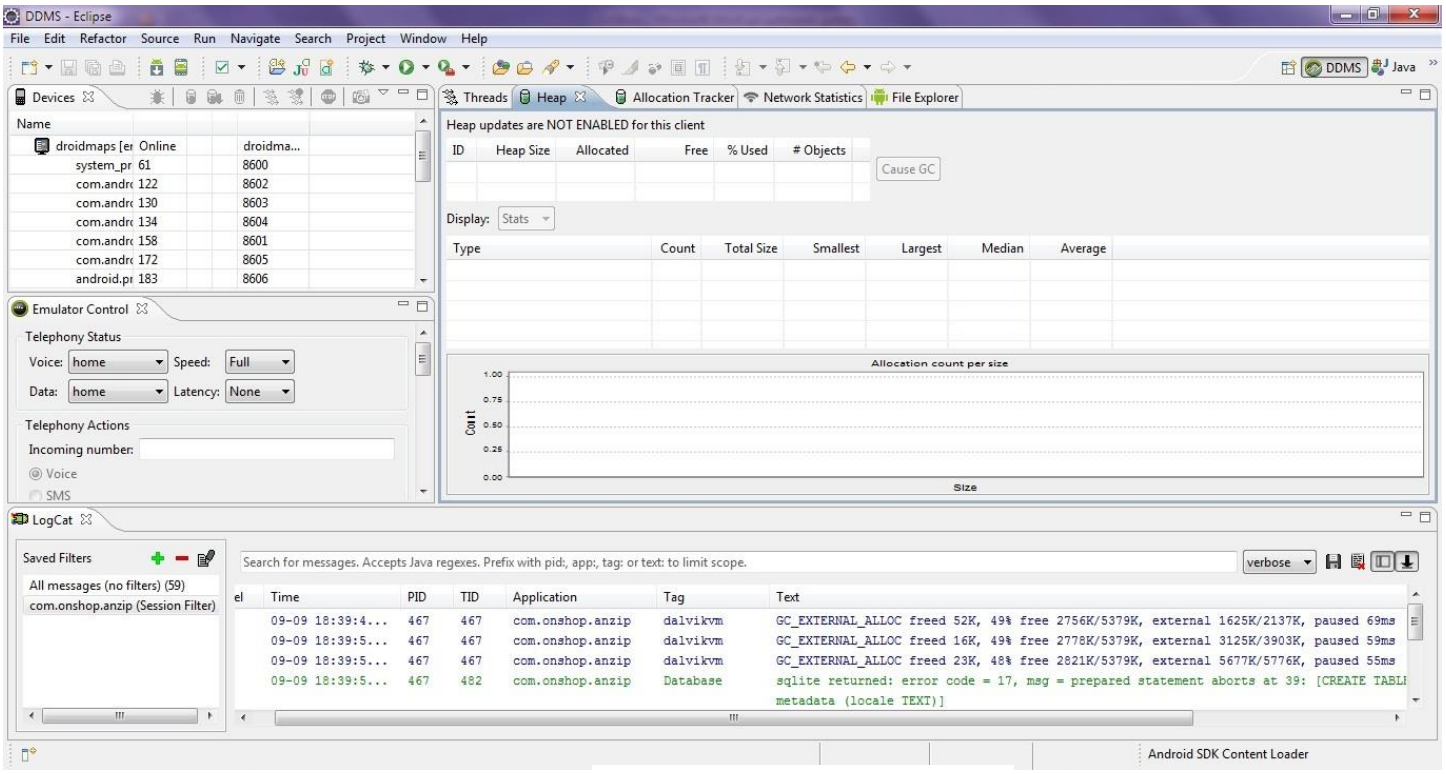
7.4.2 Dalvik Debug Monitor Server (DDMS)

Μαζί με το SDK του Android μας παρέχεται και το εργαλείο Dalvik Debug Monitor Server, το οποίο βρίσκεται στον κατάλογο <sdk>/tools. Αυτό μας παρέχει πληροφορίες για τα νήματα και το heap της συσκευής, βγάζει φωτογραφία (screenshot) την οθόνη της συσκευής, προσομοιώνει συντεταγμένες για το GPS, δημιουργεί εικονικές κλήσεις και μηνύματα (sms) 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 στην εικονική μηχανή μπορούμε να δούμε το DDMS επιλέγοντας στο Eclipse: Window->Open Perspective->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 συσκευών.

- **Emulator Control (Έλεγχος 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 τη λανθάνουσα κατάσταση – latency.
    - Τηλεφωνικές Ενέργειες (Telephony Actions): Παρέχει έναν τρόπο να κάνουμε μια ψεύτικη κλήση ομιλίας ή μηνύματος (sms) προς τον εξομοιωτή.
    - Εντοπισμός Ελέγχου (Location Control): Παρέχει έναν τρόπο να στείλουμε ένα ψεύτικο μήνυμα GPS στον εξομοιωτή.
- **Logcat:** Βρίσκεται στο κάτω παράθυρο και δείχνει όλα τα δεδομένα που καταγράφουμε από την συσκευή σε πραγματικό χρόνο.
- **Thread, Heap, Allocation Tracker, File Explorer:** Είναι οι τέσσερις καρτέλες του δεξιού πλαισίου και είναι ως επί το πλείστον οι καρτέλες που χρησιμοποιούνται για την ανάλυση της διαδικασίας.



Εικόνα XIX : DDMS

7.4.3 Traceview

Το εργαλείο Traceview προβάλει 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 θα 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 τερματιστεί τότε μπορούμε να χρησιμοποιήσουμε το εργαλείο Traceview ώστε να εξετάσουμε τα

αρχεία που καταγράφτηκαν για να πάρουμε τις χρήσιμες πληροφορίες κατά τον χρόνο εκτέλεσης, όπως για την κλήση των μεθόδων κλπ.

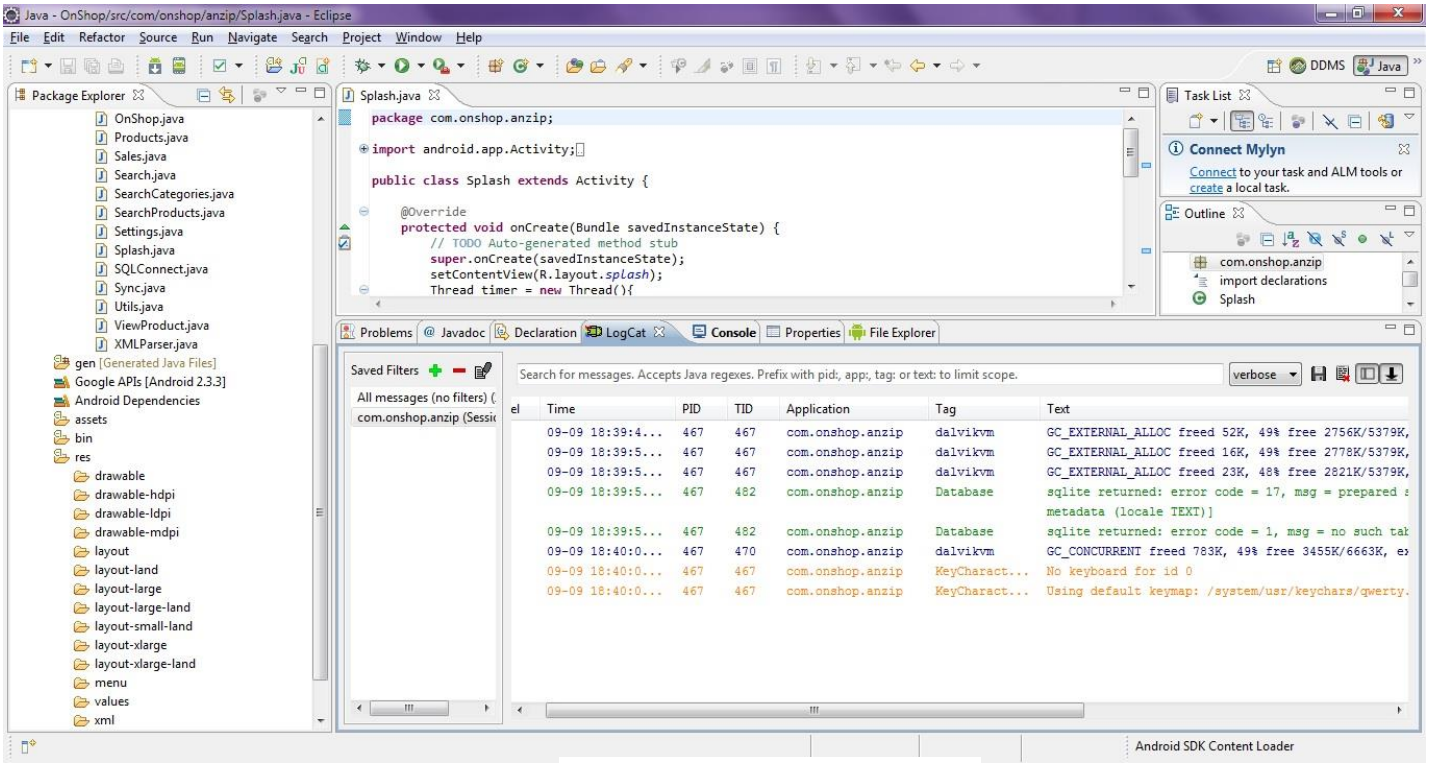
Για παράδειγμα, αν θέλουμε να καταγράψουμε τις πληροφορίες και τους χρόνους από τη στιγμή που δημιουργείται μέχρι τη στιγμή που τερματίζει η εφαρμογή μας, τότε θα πρέπει να χρησιμοποιήσουμε τις μεθόδους της Debug. Συγκεκριμένα, στην αρχή της onCreate() θα βάλουμε την Debug.startMethodTracing("something") 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 της πραγματικής συσκευής μας. Τέλος εκτελούμε από το κέλυφος (shell) την εντολή traceview something.trace ώστε να πάρουμε τη γραφική απεικόνιση των πληροφοριών μας.

7.4.4 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 Logcat στο περιβάλλον ανάπτυξης, αν δεν είναι ήδη εμφανισμένο, πηγαίνουμε στο μενού Window->Show View->Other και επιλέγουμε Logcat.



Εικόνα XX : LogCat

Χρησιμοποιώντας την κλάση Log στην εφαρμογή μας μπορούμε να καταγράφουμε μηνύματα τα οποία εμφανίζονται στο Logcat. Η καταγραφή αυτή είναι 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 εκτελείται μια μέθοδος της εφαρμογής μας, μπορούμε να χρησιμοποιήσουμε την Log.d() στην αρχή της για να το πετύχουμε. Στην αρχή της κλάσης μας δηλώνουμε την TAG κατά αυτόν τον τρόπο:

```
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.

Επίσης, λόγω του μεγάλου όγκου πληροφορίας που μας παρέχει το Logcat αλλά και λόγω της δικιάς μας χρήσης των διαφόρων τύπων μηνύματος, μπορεί να χρειαστεί να βλέπουμε μόνο τα μηνύματα που θέλουμε πχ debug, error κλπ. Αυτό το πετυχαίνουμε χρησιμοποιώντας τα φίλτρα που μας παρέχει το Logcat.

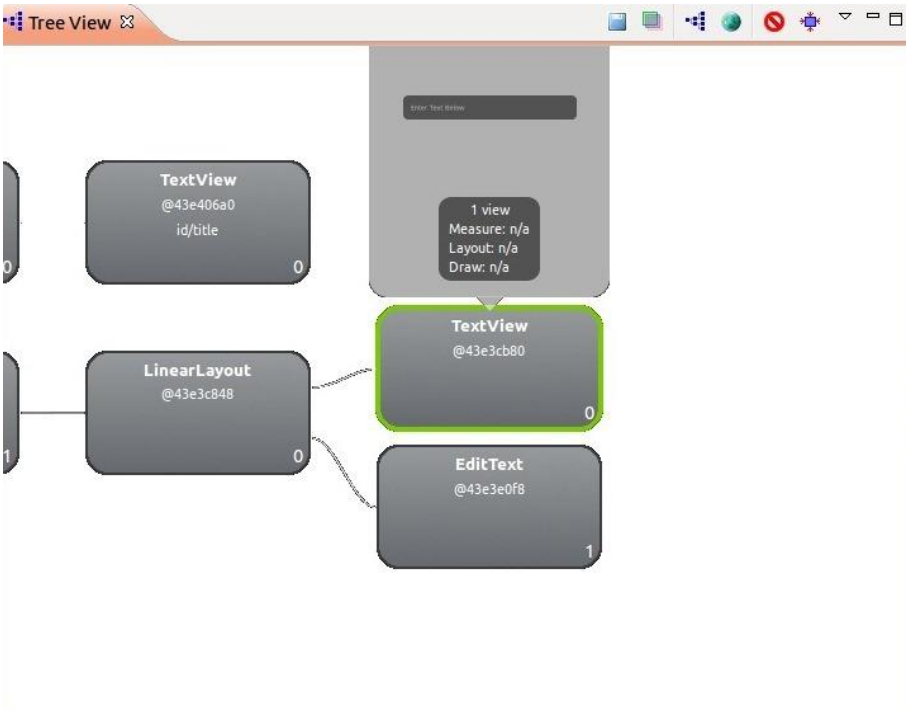
7.4.5 Hierarchy viewer

Ένας χρήσιμος τρόπος για να εντοπίσουμε και να κατανοήσουμε το περιβάλλον εργασίας χρήστη (user interface) είναι με τη χρήση του εργαλείου Hierarchy Viewer. Αυτό μας παρέχει μια οπτική αναπαράσταση της ιεραρχίας των Views της διάταξης. Για να εργαστούμε με αυτό το εργαλείο στο Eclipse πηγαίνουμε στο Window->Open Perspective->Other και επιλέγουμε Hierarchy View.

Στα αριστερά μας θα υπάρχει η λίστα με τις συνδεδεμένες συσκευές και μπορούμε να επιλέξουμε ποιας συσκευής τις εφαρμογές θέλουμε να προβάλλουμε. Στη συνέχεια επιλέγουμε την εφαρμογή που θέλουμε για εντοπισμό σφαλμάτων ή για βελτιστοποίηση του περιβάλλοντος χρήστη. Τα τέσσερα παράθυρα που εμφανίζονται είναι:

- **View Properties:** Εμφανίζει τις ιδιότητες του επιλεγμένου View
- **Tree View:** Ένα δενδρικό διάγραμμα των Views
- **Tree Overview:** Σε μικρογραφία όλο το δενδρικό διάγραμμα των Views
- **Layout View:** Προβάλλει το επιλεγμένο στοιχείο πάνω στο Layout

Όλα τα παραπάνω σχετίζονται μεταξύ τους και όταν επιλέγουμε ένα View 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 δένδρου δεν μπορεί να είναι μεγαλύτερο από 10 και το πλάτος του δεν μπορεί να είναι ευρύτερο από 50.



Εικόνα XXI : Hierarchy Viewer

Επιλέγοντας κάποιο από τα Views, μας εμφανίζονται πληροφορίες για αυτή μέσα στο Tree View, σε ένα πλαίσιο πάνω από το View που αναφέρει τους χρόνους που χρειάστηκε για κάθε ένα από τα Views που περιέχει, συμπεριλαμβανομένου και του ίδιου. Ανάλογα με το πόσο καθυστερεί ένα View στην εφαρμογή μας, το εργαλείο εμφανίζει και ένα χρώμα γύρω από αυτό. Το πράσινο χρώμα υποδηλώνει καλό χρόνο ενώ το κίτρινο και το κόκκινο χρώμα όχι. Σε περίπτωση που υπάρχει κίτρινο ή κόκκινο χρώμα σε κάποιο από τα Views, δεν σημαίνει απαραίτητα ότι δεν έχει καλή απόδοση αφού αυτό μπορεί να συμβαίνει σε ορισμένες μόνο περιπτώσεις, όπως όταν καλείται μία μέθοδος του View για ένα συγκεκριμένο συμβάν.

Σύνοψη

Τα προηγούμενα είναι τα εργαλεία που μας προσφέρει το Android SDK ώστε να μας βοηθήσει στον εντοπισμό σφαλμάτων του κώδικα αλλά και στην καταγραφή της απόδοσής του. Ο σκοπός της καταγραφής αυτής είναι να εντοπίσουμε ποιες μέθοδοι «καθυστερούν» την εφαρμογή μας ώστε, αν μπορούμε, να τις βελτιώσουμε. Είναι εμφανής η διευκόλυνση που μας προσφέρει το IDE Eclipse μαζί με το plugin του Android, αφού πολλές από τις διαδικασίες που γίνονται μέσω του shell το SDK τις αυτοματοποιεί. Φυσικά παράλληλα με τα εργαλεία αποσφαλμάτωσης του SDK, μπορούμε να χρησιμοποιήσουμε και τα debugging εργαλεία του Eclipse, προχωρώντας έτσι σε άλλες μεθόδους εντοπισμού σφαλμάτων, όπως για παράδειγμα με σημεία διακοπής (breakpoints).

8. Google Maps

Οι χάρτες Google είναι άρρηκτα συνδεδεμένοι με την πλατφόρμα του Android. Εκτός από την απλή χρήση τους μέσω της προ-εγκατεστημένης εφαρμογής, οι χάρτες επιτρέπουν στον προγραμματιστή να τους ενσωματώσει και να τους προσαρμόσει στην εφαρμογή του. Ο προγραμματιστής μπορεί ανάλογα με τις ανάγκες της εφαρμογής του να αλλάξει τον τρόπο που εμφανίζονται οι χάρτες, να κρατήσει το γεωγραφικό μήκος και πλάτος μιας τοποθεσίας, ακόμα και να προσθέσει δείκτες σε επιλεγμένες τοποθεσίες.

Για την ενσωμάτωση των χαρτών σε μια εφαρμογή πρέπει ο προγραμματιστής να αιτηθεί ένα «κλειδί», το λεγόμενο API key από την Google. Το κλειδί αυτό θα πρέπει να δηλωθεί στο xml όπου θα εμφανίζονται οι χάρτες (στο MapView). Με τις κατάλληλες αλλαγές στον κώδικα είναι δυνατή και η λειτουργία ζουμ. Προγραμματιστικά μπορεί να αλλάξει και ο τρόπος εμφάνισης του χάρτη δηλαδή να εμφανίζεται σαν δορυφορική εικόνα, σαν χάρτης ή σαν εικόνα από το επίπεδο του δρόμου (όπου είναι εφικτή αυτή η λειτουργία). Φυσικά είναι δυνατή η εμφάνιση μιας συγκεκριμένης γεωγραφικά τοποθεσίας σύμφωνα με ορισμένο μήκος και πλάτος αλλά και το αντίθετο, η εμφάνιση δηλαδή των συντεταγμένων γνωρίζοντας μια διεύθυνση ή πατώντας απλά ένα σημείο στο χάρτη. Τέλος, μια πολύ χρήσιμη λειτουργία είναι η τοποθέτηση δεικτών, οι οποίοι συνήθως έχουν το σχήμα πινέζας, στα σημεία ενδιαφέροντος. Η λειτουργία αυτή βοηθάει το χρήστη να βλέπει τα επονομαζόμενα POIs (Points Of Interest) ακόμα και χωρίς να έχει κάνει αρκετό ζουμ στο χάρτη.

Google APIs Add-On

Το add-on του Google API είναι μια επέκταση στο προγραμματιστικό περιβάλλον του Android που μας επιτρέπει να αναπτύξουμε εφαρμογές που περιέχουν ένα σύνολο προσαρμοσμένων βιβλιοθηκών και υπηρεσιών. Σημαντικό χαρακτηριστικό αυτού του add-on είναι η εξωτερική βιβλιοθήκη χαρτών που περιέχει και η οποία μας επιτρέπει να προσθέσουμε δυνατότητα εμφάνισης χαρτών στην εφαρμογή μας.

Το πρόσθετο αυτό παρέχει επίσης ένα συμβατό σύστημα που τρέχει στον emulator του Android ώστε να μπορούμε να δοκιμάζουμε και να κάνουμε debug την εφαρμογή μας. Το σύστημα περιλαμβάνει τη βιβλιοθήκη των χαρτών και άλλα επιπρόσθετα στοιχεία τα οποία τυχόν θα χρειαστούμε για να έχουμε πρόσβαση σε υπηρεσίες της Google.

8.1 Απόκτηση του API key

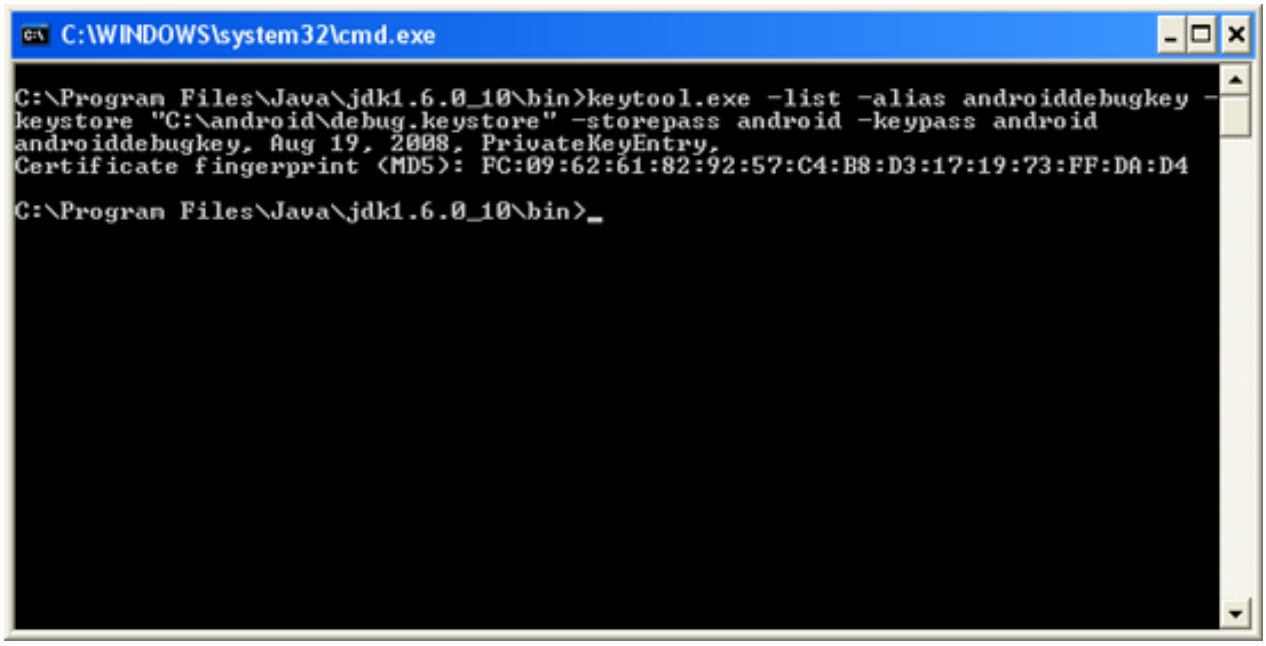
Για να αιτηθούμε ένα τέτοιο κλειδί για τους χάρτες Google θα πρέπει να ακολουθήσουμε τα βήματα που αναφέρονται παρακάτω (αναλυτική περιγραφή υπάρχει και στο http://code.google.com/android/toolbox/apis/mapkey.html).

Αρχικά αν δοκιμάζουμε την εφαρμογή στον προσομοιωτή του Android πρέπει να βρούμε το πιστοποιητικό αποσφαλμάτωσης του SDK που βρίσκεται στο φάκελο "C:\Documents and

Settings\<username>\Local Settings\Application Data\Android". Το όνομα αυτού του αρχείου θα είναι debug.keystore. Αν δοκιμάζουμε την εφαρμογή μας σε πραγματική συσκευή, θα πρέπει να αντικατασταθεί αυτό το αρχείο με ένα που θα δημιουργήσουμε εμείς. Για λόγους απλούστευσης θα μεταφέρουμε αυτό το αρχείο στη ρίζα του δίσκου (για παράδειγμα σε ένα φάκελο "C:\Android").

Στη συνέχεια θα χρησιμοποιήσουμε την εφαρμογή keytool.exe που υπάρχει στο JDK για να εξάγουμε το MD5 αποτύπωμα (MD5 fingerprint). Το αποτύπωμα είναι απαραίτητο στην αίτησή μας για την απόκτηση του API key που θέλουμε. Το keytool.exe μπορούμε να το βρούμε συνήθως στη διαδρομή "C:\Program Files\Java\<JDK_version_number>\bin". Τρέχοντας την παρακάτω εντολή γίνεται η εξαγωγή του αποτυπώματος:

```
keytool.exe -list -alias androiddebugkey -keystore "C:\android\debug.keystore" -storepass android -keypass android
```



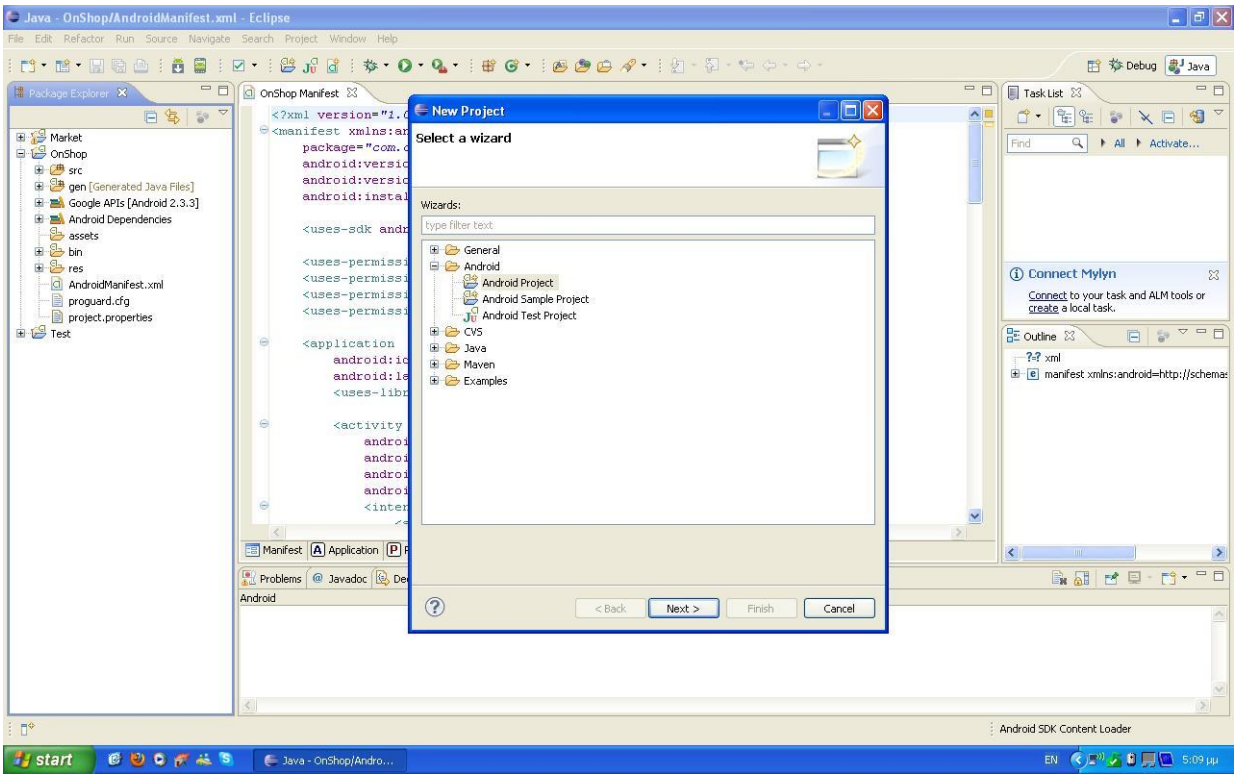
Εικόνα XXII : Εντολή για το MD5 Fingerprint

Τώρα θα πρέπει να αντιγράψουμε το MD5 fingerprint και να ακολουθήσουμε τις οδηγίες της ιστοσελίδας http://code.google.com/android/maps-api-signup.html για να πάρουμε το API key.

8.2 Δημιουργία Google Maps Project

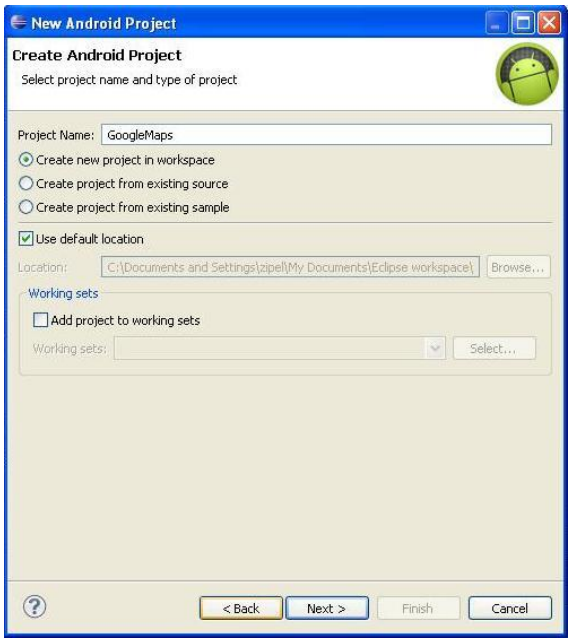
Τώρα είμαστε έτοιμοι να δημιουργήσουμε ένα καινούργιο Android Project στο Eclipse ώστε να φτιάξουμε μία εφαρμογή που θα περιέχει Google Maps.

Πατάμε File->New->Project… και στο παράθυρο που ανοίγει επιλέγουμε Android Project όπως φαίνεται και στην εικόνα:



Εικόνα XXIII : Νέο Maps Project

Στο επόμενο παράθυρο επιλέγουμε ένα όνομα για το Project μας:



Εικόνα XXIV : Νέο Maps Project (2)

Τέλος, δίνουμε όνομα στο package που θα χρησιμοποιήσουμε καθώς και στο αρχικό μας activity και ορίζουμε για ποια έκδοση του λειτουργικού προορίζεται η εφαρμογή μας:

Εικόνα XXV : Νέο Maps Project (3)

Φυσικά είναι απαραίτητη η αλλαγή του AndroidManifest.xml αρχείου προσθέτοντας τη βιβλιοθήκη των χαρτών και την άδεια για τη χρήση του internet για να μπορέσουμε να χρησιμοποιήσουμε τους χάρτες στην εφαρμογή μας:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.onshop.anzip"
    android:versionCode="1"
    android:versionName="1.0.0">
   <application android:icon="@drawable/icon" android:label="@string/app_name">
    <uses-library android:name="com.google.android.maps" />
      <activity android:name=".MapsActivity"
            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-permission android:name="android.permission.INTERNET" />
</manifest>
```

```
</xml>
```

Εμφάνιση του χάρτη

Για να εμφανίσουμε το χάρτη στην εφαρμογή μας πρέπει να δημιουργήσουμε ένα καινούριο xml αρχείο (ή να τροποποιήσουμε ένα ήδη υπάρχον). Προσθέτοντας το στοιχείο <com.google.android.maps.MapView> θα εμφανιστεί ο χάρτης στο activity μας. Αυτό το MapView θα τοποθετηθεί μέσα σε ένα <RelativeLayout>:

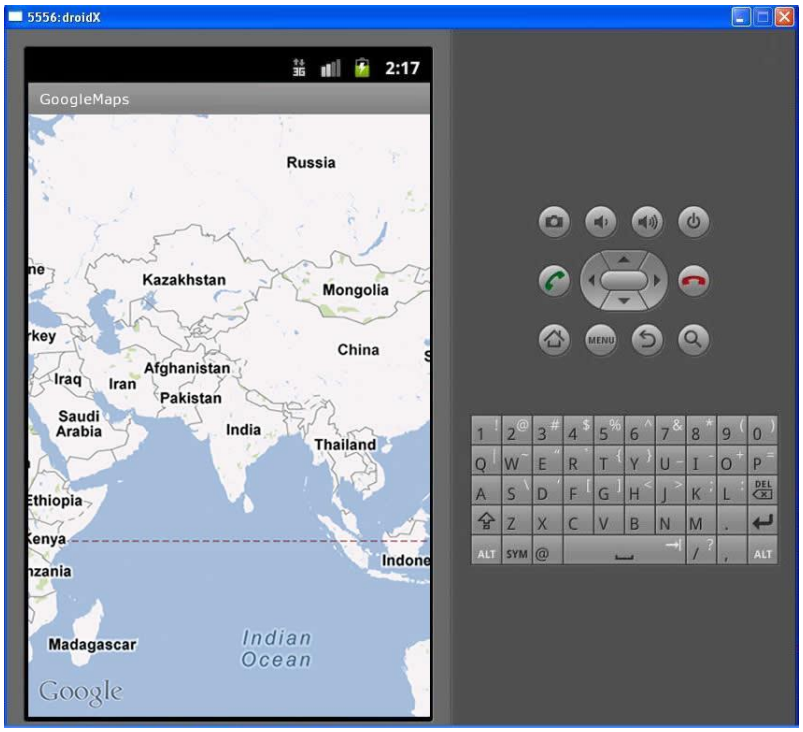
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
    <com.google.android.maps.MapView
      android:id="@+id/mapView"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:enabled="true"
      android:clickable="true"
      android:apiKey="0l4sCTTyRmXTNo7k8DREHvEaLar2UmHGwnhZVHQ"
      />
 </RelativeLayout>
```

Το χαρακτηριστικό android:apiKey είναι αυτό που μας δόθηκε από την Google ακολουθώντας τα προηγούμενα βήματα.

Τέλος, θα πρέπει να τροποποιήσουμε το MapsActivity.java αρχείο:

```
package com.onshop.anzip;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import android.os.Bundle;

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

    @Override
   protected boolean isRouteDisplayed() {
      return false;
   }
}
```

Το μόνο που μένει είναι να δοκιμάσουμε την εφαρμογή μας στον emulator και ο χάρτης θα εμφανιστεί στην οθόνη της συσκευής όπως φαίνεται και στην εικόνα.



Εικόνα XXVI: Google Maps στον Emulator

Λειτουργία Ζουμ

Με την παραπάνω διαδικασία δώσαμε στον χρήστη τη δυνατότητα να πλοηγηθεί στους χάρτες της Google μέσω της εφαρμογής μας και η επιθυμητή τοποθεσία θα ενημερώνεται άμεσα μέσω internet. Δεν υπάρχει όμως η δυνατότητα ζουμ σε μια συγκεκριμένη τοποθεσία. Για να προσθέσουμε αυτή τη λειτουργία πρέπει να κάνουμε τα παρακάτω βήματα.

Πρώτα προσθέτουμε ένα Layout ακόμα στο xml:

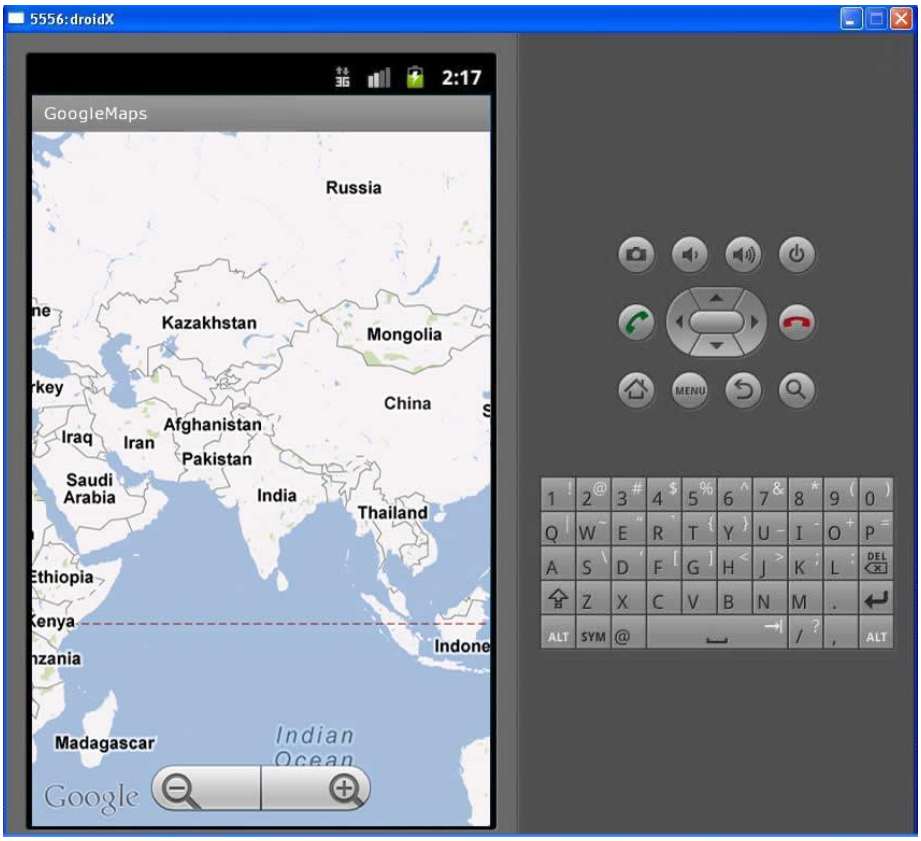
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">

   <com.google.android.maps.MapView
      android:id="@+id/mapView"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:enabled="true"
      android:clickable="true"
      android:apiKey="0l4sCTTyRmXTNo7k8DREHvEaLar2UmHGwnhZVHQ"
      />

   <LinearLayout android:id="@+id/zoom"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
```

```
    android:layout_centerHorizontal="true"
    />

</RelativeLayout>
```

Το LinearLayout θα χρησιμοποιηθεί για να εμφανίζονται τα κουμπιά ελέγχου του ζουμ (+,-) στην οθόνη.

Στο MapsActivity.java θα προσθέσουμε κάποια imports και ένα κομμάτι κώδικα μετά από τη μέθοδο setContentView:

```
package com.onshop.anzip;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.MapView.LayoutParams;

import android.os.Bundle;
import android.view.View;
import android.widget.LinearLayout;

public class MapsActivity extends MapActivity
{
   MapView mapView;

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

      mapView = (MapView) findViewById(R.id.mapView);
      LinearLayout zoomLayout = (LinearLayout)findViewById(R.id.zoom);
      View zoomView = mapView.getZoomControls();

      zoomLayout.addView(zoomView,
         new LinearLayout.LayoutParams(
            LayoutParams.WRAP_CONTENT,
            LayoutParams.WRAP_CONTENT));
      mapView.displayZoomControls(true);

   }

   @Override
   protected boolean isRouteDisplayed() {
      // TODO Auto-generated method stub
      return false;
   }
}
```



Εικόνα XXVII : Google Maps στον Emulator (2)

Με τις παραπάνω αλλαγές εμφανίζονται πλέον δύο κουμπιά + και - πάνω στο χάρτη με τα οποία μπορούμε πλέον να κάνουμε ζουμ σε μία τοποθεσία.

9. VirtueMart

Το Virtuemart είναι μία δωρεάν Open Source εφαρμογή ανάπτυξης ηλεκτρονικού καταστήματος. Δεν μπορεί να λειτουργήσει αυτόνομα παρά μόνο ως πρόσθετη εφαρμογή του δημοφιλούς συστήματος διαχείρισης περιεχομένου (CMS) Joomla!.

Είναι γραμμένο με γλώσσα προγραμματισμού PHP και αποθηκεύει τα δεδομένα του σε βάση δεδομένων MySQL. Εκτός από την βασική εφαρμογή συνοδεύεται από μία σειρά modules που εμφανίζουν χρήσιμες πληροφορίες (νέα προϊόντα, δημοφιλή προϊόντα κλπ.)

Γενικά χαρακτηριστικά

- Δυνατότητα χρήσης Secure Sockets Layer (https) Κρυπτογράφηση (128-bit)
- Διαφορετικά είδη φόρου
    - Υπολογισμός φόρου ανάλογα με τη διεύθυνση αποστολής
    - Υπολογισμός φόρου ανάλογα με τη διεύθυνση του καταστήματος
    - EU Mode (Υπολογισμός φόρου ανάλογα με τη διεύθυνση του καταστήματος όταν ο πελάτης προέρχεται από χώρα της Ευρωπαϊκής Ένωσης EU Country)
- Οι αγοραστές μπορούν να διαχειριστούν τους λογαριασμούς τους
- Διαχείριση διεύθυνσης αποστολής: Οι αγοραστές μπορούν να ορίσουν διαφορετικές διευθύνσεις αποστολής από την διεύθυνση χρέωσης
- Ιστορικό παραγγελιών: Ο αγοραστής μπορεί να δει όλες τις προηγούμενες παραγγελίες και τις λεπτομέρειές τους
- E-mail επιβεβαίωσης παραγγελίας (μπορείτε να του δώσετε τη μορφή που θέλετε) αποστέλλεται στον αγοραστή και στον ιδιοκτήτη
- Υποστήριξη πολλαπλών νομισμάτων (μπορείτε να επιτρέψετε στους πελάτες σας να αλλάξουν το νόμισμα πληρωμής)
- Υποστήριξη πολλαπλών γλωσσών (με τη χρήση του Joom!Fish ή του Nooku).

Χαρακτηριστικά καταλόγου προϊόντων

- Ισχυρό περιβάλλον διαχείρισης βασισμένο σε Javascript
- Διαχείριση άπειρων προϊόντων και κατηγοριών
- Μπορεί να χρησιμοποιηθεί ως ηλεκτρονικό κατάστημα ή ως κατάλογος προϊόντων (μπορείτε να απενεργοποιήσετε την εμφάνιση των τιμών)
- Γρήγορη αναζήτηση για προϊόντα, κατηγορίες και κατασκευαστές. Φίλτρα με βάση τα χαρακτηριστικά ή τις εκπτώσεις των προϊόντων
- Αξιολόγηση & κριτική προϊόντων (με αυτόματη ή ελεγχόμενη δημοσίευση)
- Δυνατότητα χαρακτηρισμού προϊόντων ως "special"
- Διαθεσιμότητα προϊόντων: εμφάνιση πιθανής ημερομηνίας αποστολής προϊόντων
- Διαχείριση προϊόντων με δυνατότητα μεταφόρτωσης (προγράμματα, e-books, φωτογραφίες)
- Υπενθύμιση εγγεγραμμένων πελατών – ("Το προϊόν είναι πάλι διαθέσιμο")

Χαρακτηριστικά διαχειριστή

- Πολλαπλή χρήση εικόνων και αρχείων (όπως έντυπα προδιαγραφών, διαφημιστικά) ανά προϊόν
- Ιδιότητες (όπως χρώμα ή μέγεθος) μπορούν να οριστούν για κάθε προϊόν
- Τύποι προϊόντων για ταξινόμηση (όπως "Car", "Motorbike" ή "Music Album")
- Ομάδες αγοραστών για τους πελάτες (επιτρέπει διαφορετικά επίπεδα κοστολόγησης και τρόπου πληρωμής)
- Πολλαπλές τιμές ανά προϊόν (βασισμένες στην ποιότητα και/ή στην ομάδα αγοραστών)
- Ευέλικτη εμφάνιση τιμών (με ή χωρίς φόρους)
- Απ' ευθείας μετατροπή διαφορετικών νομισμάτων με βάση τις ισοτιμίες από την ΕΚΤ και άλλες τράπεζες
- Στατιστικά καταστήματος / Πίνακας ελέγχου με σύνολα νέων πελατών, νέων παραγγελιών κ.λπ.
- Έλεγχος επιπέδων αποθήκης για τα προϊόντα
- Διαχείριση παραγγελιών με ιστορικό παραγγελιών, ειδοποιήσεις πελατών και δυνατότητα επεξεργασίας παραγγελιών
- Διάφορες αναφορές: πουλημένα είδη, μηνιαίο/ετήσιο εισόδημα
- Διαχείριση κατάστασης παραγγελίας
- Διαχείριση νομισμάτων, χωρών & περιοχών

Τρόποι πληρωμής

- Δυνατότητα πληρωμής με πιστωτική κάρτα
- Προκαθορισμένοι τρόποι πληρωμής όπως authorize.net®, PayPal, 2Checkout, eWay, Worldpay, PayMate και NoChex
- Δυνατότητα επέκτασης του καταστήματός μας με άλλους τρόπους πληρωμής χρησιμοποιώντας την δυνατότητα API

Τρόποι αποστολής

- Ευέλικτη διαμόρφωση μεταφορέων και κόστους μεταφορικών
- Απ' ευθείας υπολογισμός κόστους μεταφορών χρησιμοποιώντας γνωστούς μεταφορείς (π.χ. InterShipper, UPS, USPS, FedEx or Canada Post).
- Δυνατότητα επέκτασης του καταστήματός μας με άλλους τρόπους μεταφορικών χρησιμοποιώντας την δυνατότητα API

Απαιτήσεις συστήματος

- Apache 1.3.x ή νεότερο Συνιστάται: Apache 2.2.x
- MySQL 3.23.x ή νεότερη Συνιστάται: MySQL 5.0.x
- PHP 4.2.x ή νεότερη Συνιστάται: PHP 5.2.x
- Joomla! 1.0 ή 1.5.x: Συνιστάται: Joomla! 1.5.15

- Εάν εγκαταστήσετε το Joomla τοπικά στον υπολογιστή σας πρέπει να έχετε εγκατεστημένη μία εφαρμογή που υποστηρίζει Apache, MySQL, PHP. Συνιστάται WAMP: http://www.wampserver.com/en/
- Web Browser Internet Explorer 6+, Firefox 1.5+, Safari 2+ and Opera 9+. Συνιστάται Firefox 3
- Για την Περιοχή Διαχείρισης απαιτείται η ενεργοποίηση της Javascript.
- Ο Web Browser πρέπει να έχει ενεργοποιημένα τα Cookies.

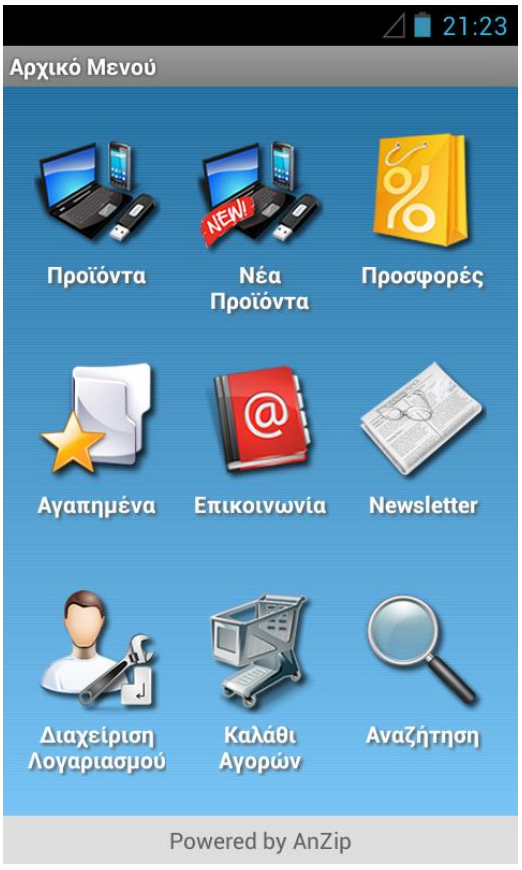
10. Περιγραφή εφαρμογής OnShop

Στο κεφάλαιο αυτό θα παρουσιάσουμε ένα εγχειρίδιο χρήσης της εφαρμογής OnShop. Θα περιγραφούν αναλυτικά όλες οι λειτουργίες της ενώ παραθέτονται και εικόνες της εφαρμογής κατά την λειτουργία της σε μια συσκευή κινητού τηλεφώνου.

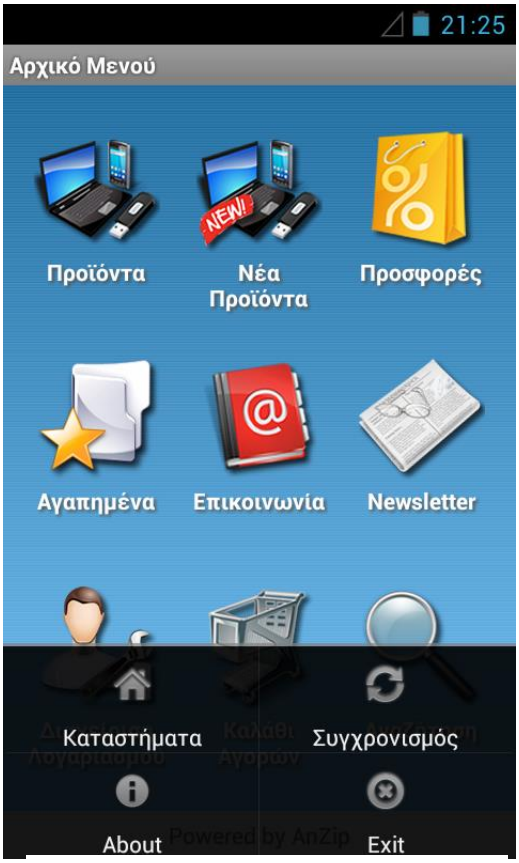
Μόλις ανοίξουμε την εφαρμογή εμφανίζεται η εικόνα εισαγωγής με το λογότυπο του καταστήματος και αμέσως μετά εμφανίζεται το κεντρικό μενού με τις εξής επιλογές:

- **Προϊόντα**
- **Νέα Προϊόντα**
- **Προσφορές**
- **Αγαπημένα**
- **Επικοινωνία**
- **Newsletter**
- **Διαχείριση Λογαριασμού**
- **Καλάθι Αγορών**
- **Αναζήτηση**

Επίσης στο κεντρικό μενού, πατώντας το πλήκτρο του μενού στη συσκευή εμφανίζεται και ένα αναδυόμενο μενού επιλογών με τις επιλογές: Καταστήματα, Συγχρονισμός, About και Exit. Παρακάτω βλέπουμε στιγμιότυπα του Activity αυτού.

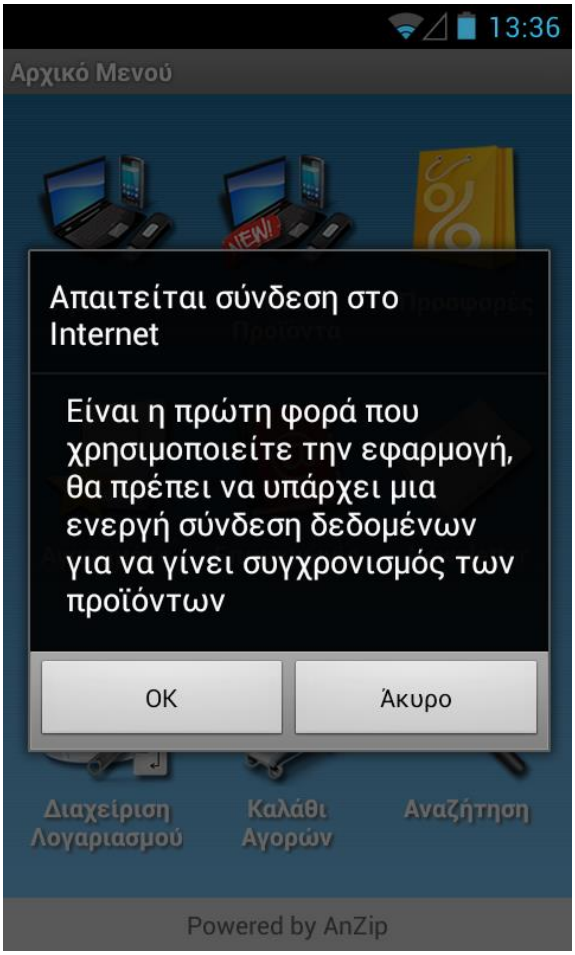


Εικόνα XXVIII : Αρχικό μενού

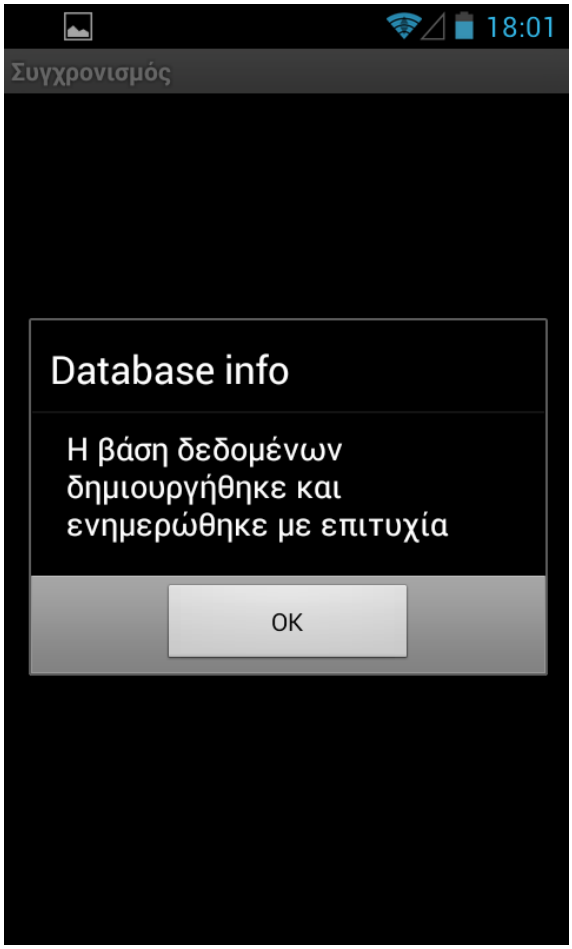


Εικόνα XXIX : Αναδυόμενο μενού

Στην περίπτωση που χρησιμοποιούμε την εφαρμογή για πρώτη φορά θα μας ζητηθεί να γίνει συγχρονισμός των προϊόντων του καταστήματος για να αποθηκευτούν στην τοπική βάση δεδομένων, κάτι που θα μας δώσει τη δυνατότητα να λειτουργούμε την εφαρμογή και χωρίς σύνδεση στο internet. Για να γίνει όμως αυτό αρχικά θα πρέπει να έχουμε μια σύνδεση στο δίκτυο. Για το σκοπό αυτό γίνεται έλεγχος και αν η εφαρμογή εκτελείται πρώτη φορά βλέπουμε την παρακάτω εικόνα στα αριστερά:



Εικόνα XXX : Πρώτη εκτέλεση της εφαρμογής



Εικόνα XXXI : Μήνυμα επιτυχίας

Αν πατήσουμε το άκυρο η εφαρμογή θα κλείσει ενώ πατώντας το οκ και έχουμε μια σύνδεση δεδομένων θα γίνει ο συγχρονισμός των προϊόντων και θα εμφανιστεί ο διάλογος της εικόνας δεξιά παραπάνω.

Συνεχίζοντας στο κεντρικό μενού, πατώντας το κουμπί Προϊόντα εμφανίζονται όλες οι κατηγορίες των προϊόντων που υπάρχουν και στην ιστοσελίδα του καταστήματος. Από εκεί μπορούμε να επιλέξουμε την επιθυμητή κατηγορία και θα εμφανιστούν όλα τα προϊόντα καθώς και οι υποκατηγορίες που ανήκουν στην αρχική κατηγορία που επιλέξαμε.

Παρακάτω μπορούμε να δούμε στιγμιότυπα από το πώς εμφανίζεται η λίστα των κατηγοριών καθώς και ένα στιγμιότυπο από τα προϊόντα και τις υποκατηγορίες μιας κατηγορίας.

Εικόνα XXXII : Κατηγορίες Προϊόντων

Εικόνα XXXIII : Προϊόντα

Τα επόμενα κουμπιά στο κεντρικό μενού είναι τα Νέα Προϊόντα και οι Προσφορές, καθένα από τα οποία εμφανίζουν τα προϊόντα που βρίσκονται στις κατηγορίες Νέες Παραλαβές και Προσφορές αντίστοιχα στο ηλεκτρονικό κατάστημα. Το κουμπί Αγαπημένα που είναι το επόμενο, εμφανίζει τα προϊόντα τα οποία ο χρήστης έχει ορίσει ως αγαπημένα.

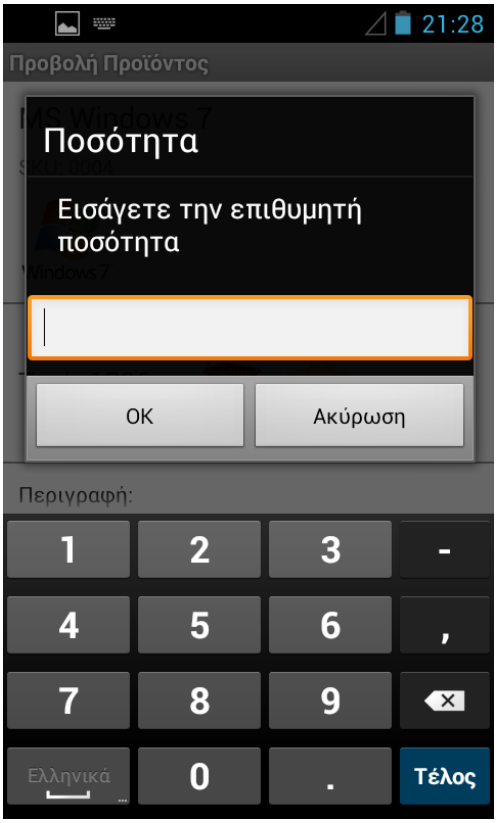
Επίσης, μια ακόμα λειτουργία είναι πως σε όλες τις λίστες εμφάνισης δίνεται στο χρήστη η δυνατότητα για αύξουσα ή φθίνουσα ταξινόμηση με βάση το όνομα ή το κόστος των προϊόντων απλά πατώντας το κουμπί του μενού.

Συνεχίζοντας, κάνοντας κλικ πάνω σε ένα οποιοδήποτε προϊόν, ανοίγει ένα νέο παράθυρο το οποίο μας δίνει περισσότερες πληροφορίες για το προϊόν αυτό και μας δίνει και κάποιες επιλογές. Πιο συγκεκριμένα, μας εμφανίζει τα εξής χαρακτηριστικά του προϊόντος:

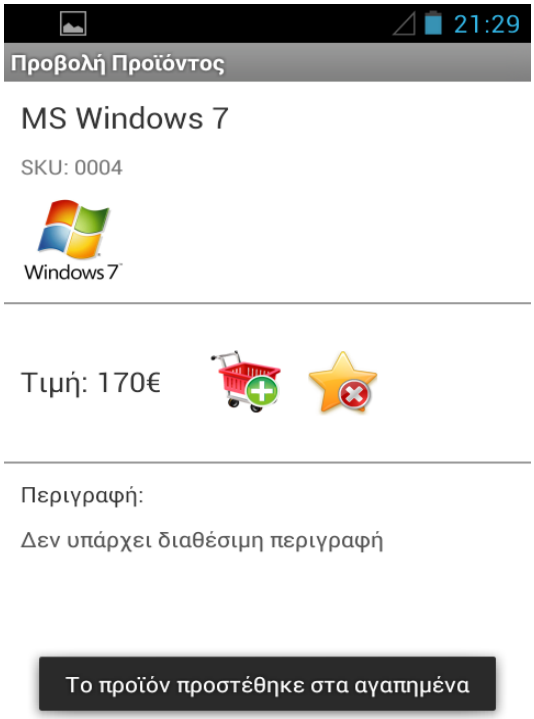
- Τον πλήρη τίτλο
- Τον κωδικό SKU
- Την πλήρη εικόνα του

- Την περιγραφή
- Την τιμή
- Κουμπί προσθήκης – αφαίρεσης στο καλάθι
- Κουμπί προσθήκης – αφαίρεσης στα αγαπημένα

Αν ο χρήστης επιθυμεί να προσθέσει το προϊόν στα αγαπημένα ή στο καλάθι απλά κάνει κλικ επάνω στο αντίστοιχο κουμπί. Στην περίπτωση που το προϊόν πρόκειται να προστεθεί στο καλάθι εμφανίζεται πρώτα ένα παράθυρο διαλόγου που ζητά από τον χρήστη να εισάγει την επιθυμητή ποσότητα και αφότου την εισάγει εμφανίζεται κατάλληλο μήνυμα επιτυχίας της προσθήκης. Στην περίπτωση της προσθήκης στα αγαπημένα απλά εμφανίζεται το μήνυμα της επιτυχούς προσθήκης. Παρακάτω βλέπουμε στιγμιότυπα από τις παραπάνω διαδικασίες



Εικόνα XXXIV : Προσθήκη στο καλάθι



Εικόνα XXXV : Προβολή προϊόντος

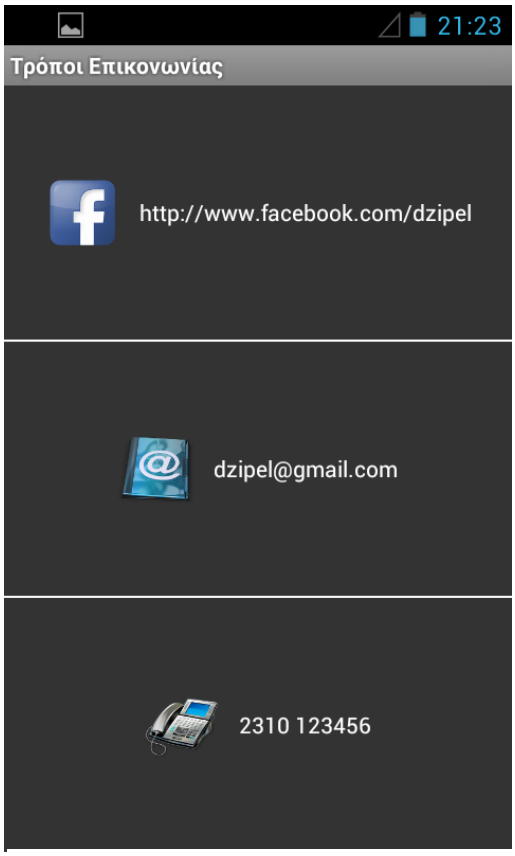
Στην δεξιά εικόνα παρατηρούμε πως μετά την προσθήκη το κουμπί της προσθήκης στα αγαπημένα έχει πλέον γίνει κουμπί αφαίρεσης. Το ίδιο ισχύει και για το κουμπί της προσθήκης στο καλάθι.

Το επόμενο κουμπί μας εμφανίζει τις πληροφορίες επικοινωνίας και μας δίνει τη δυνατότητα να χρησιμοποιήσουμε κάθε μία από αυτές απλά κάνοντας κλικ πάνω τους. Αν κάνουμε κλικ στο Facebook θα μεταφερθούμε στην αντίστοιχη σελίδα στο internet. Κάνοντας κλικ στη διεύθυνση e-mail μπορούμε να στείλουμε ένα e-mail προς τον διαχειριστή του καταστήματος ενώ κάνοντας κλικ στο τηλέφωνο γίνεται αυτόματα κλήση προς το κατάστημα. Ο χρήστης

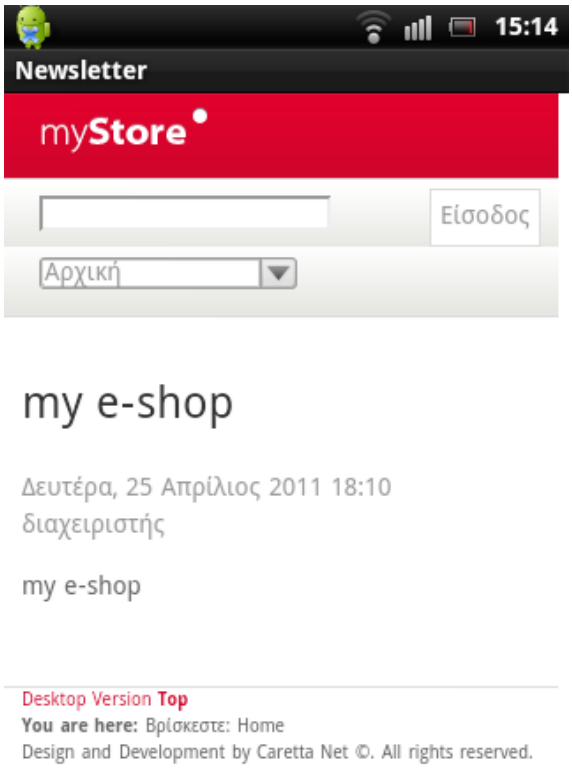
μπορεί να διαλέξει με ποια εφαρμογή θα στείλε το e-mail αν έχει εγκατεστημένες περισσότερες από 1 για τον σκοπό αυτό ενώ το ίδιο ισχύει και για το άνοιγμα της σελίδας στο internet.

Το κουμπί Newsletter μας ανοίγει ένα παράθυρο το οποίο δείχνει τα νέα του καταστήματος σε μια ειδικά διαμορφωμένη ιστοσελίδα για το σκοπό αυτό. Η χρήση του απαιτεί σύνδεση στο internet.

Στιγμιότυπα του παραθύρου που δείχνει τους διαφόρους τρόπους επικοινωνίας καθώς και του newsletter φαίνονται παρακάτω



Εικόνα XXXVI : Τρόποι επικοινωνίας



Εικόνα XXXVII : Newsletter

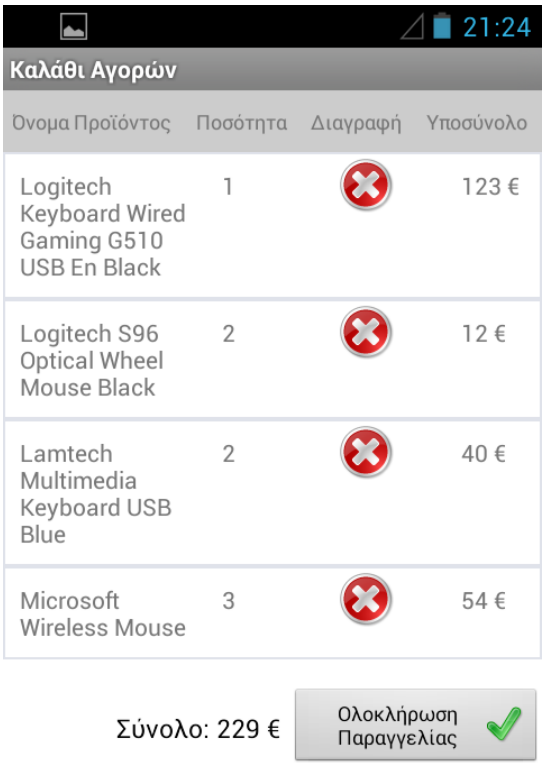
Το κουμπί ρυθμίσεις χρήστη δίνει στον χρήστη της εφαρμογής την δυνατότητα να κάνει είσοδο στο κατάστημα, προκειμένου να δει στοιχεία για το λογαριασμό του ή να τροποποιήσει κάποια από αυτά. Θα πρέπει να υπάρχει σύνδεση στο Internet για την χρήση της λειτουργίας αυτής.

Το προτελευταίο κουμπί του αρχικού μενού, το κουμπί καλάθι αγορών, εμφανίζει στον χρήστη τα προϊόντα που έχει προσθέσει προηγουμένως στο καλάθι του προς αγορά. Τα προϊόντα εμφανίζονται σε μορφή πίνακα με εγγραφές το κάθε προϊόν που περιέχεται στο

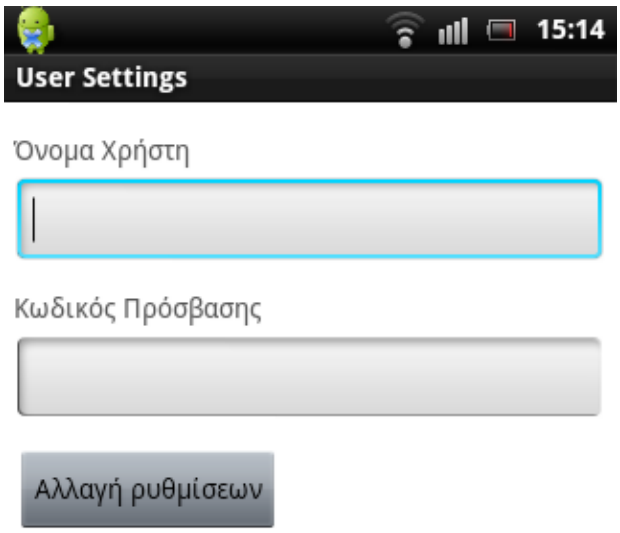
καλάθι ενώ για κάθε ένα από αυτά φαίνεται το όνομα του, η ποσότητα προς αγορά, ένα κουμπί διαγραφής από το καλάθι και το υποσύνολο κόστους του κάθε προϊόντος ανάλογα με την ποσότητα.

Επίσης στο κάτω μέρος του καλαθιού εμφανίζεται το σύνολο κόστους των προϊόντων στο καλάθι και το κουμπί ολοκλήρωσης της παραγγελίας.

Παρακάτω βλέπουμε στιγμιότυπα των ρυθμίσεων και του καλαθιού:



Εικόνα XXXVIII : Καλάθι αγορών

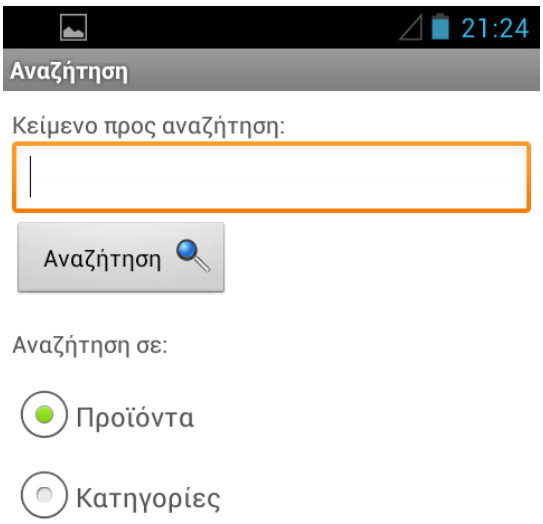


Εικόνα XXXIX : Ρυθμίσεις χρήστη

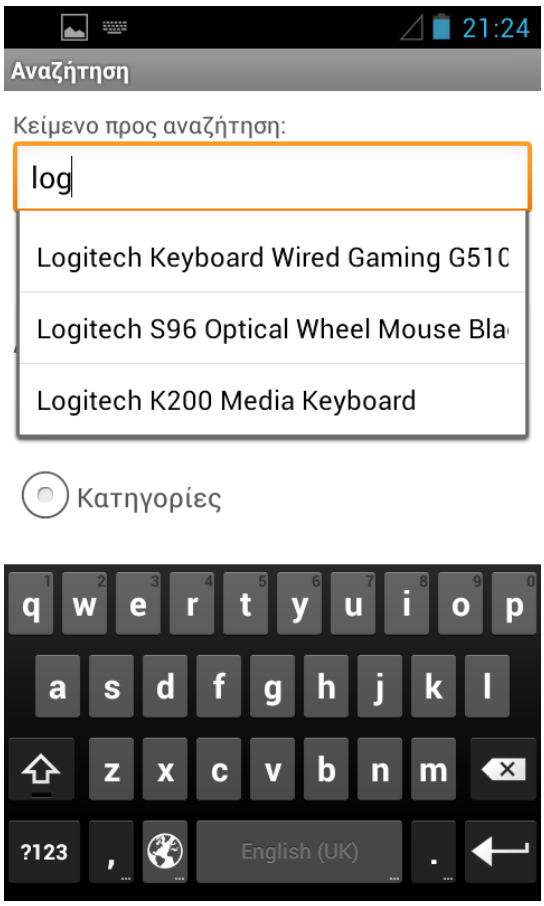
Το τελευταίο κουμπί του μενού, δίνει στον χρήστη τη δυνατότητα να αναζητήσει προϊόντα ή κατηγορίες που υπάρχουν στο κατάστημα με βάση το όνομα τους. Τα αποτελέσματα επιστρέφονται σε μορφή λίστας ή σε μορφή πλέγματος για προϊόντα ή κατηγορίες αντίστοιχα.

Κατά την αναζήτηση, δίνεται ευκολία και βοήθεια στον χρήση με την αυτόματη συμπλήρωση του κειμένου προς αναζήτηση. Κάθε φορά δηλαδή που πρόκειται να αναζητηθεί κάτι, πληκτρολογώντας ο χρήστης λαμβάνει προτάσεις οι οποίες σχετίζονται με αυτό που έχει ήδη γράψει και αντιστοιχούν στα ήδη υπάρχοντα προϊόντα ή κατηγορίες του καταστήματος.

Μπορούμε να δούμε πως φαίνονται κατά την εκτέλεση της εφαρμογής τα παραπάνω στα δύο αυτά στιγμιότυπα:



Εικόνα XL : Αναζήτηση



Εικόνα XLI : Αυτόματη συμπλήρωση

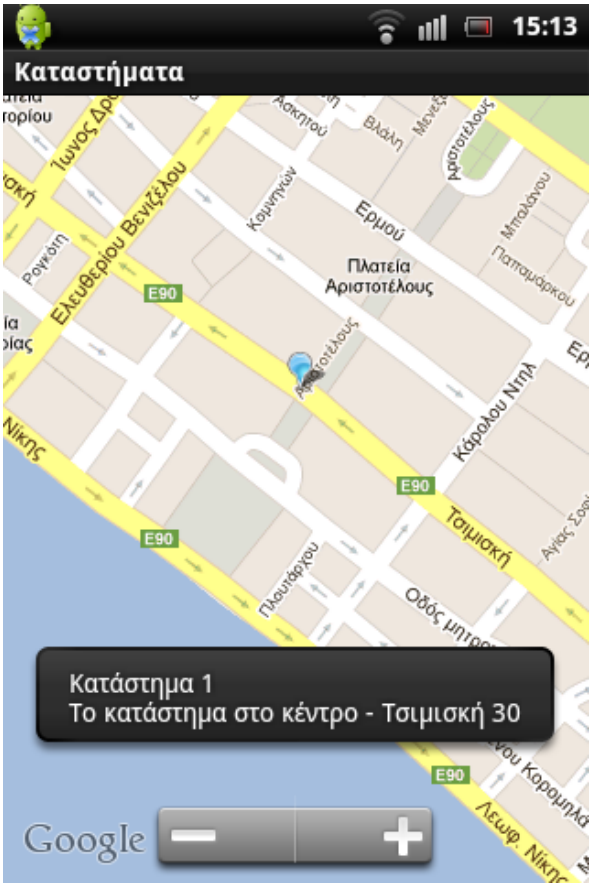
Τέλος, στο αναδυόμενο μενού που αναφέρθηκε αρχικά υπάρχουν τα κουμπιά Καταστήματα, Συγχρονισμός, About και Exit.

Το κουμπί Καταστήματα μας εμφανίζει ένα παράθυρο το οποίο με χρήση των Google Maps μας δείχνει τα καταστήματα που υπάρχουν και κάνοντας κλικ στο καθένα από αυτά λαμβάνουμε πληροφορίες για την ακριβή τοποθεσία τους.

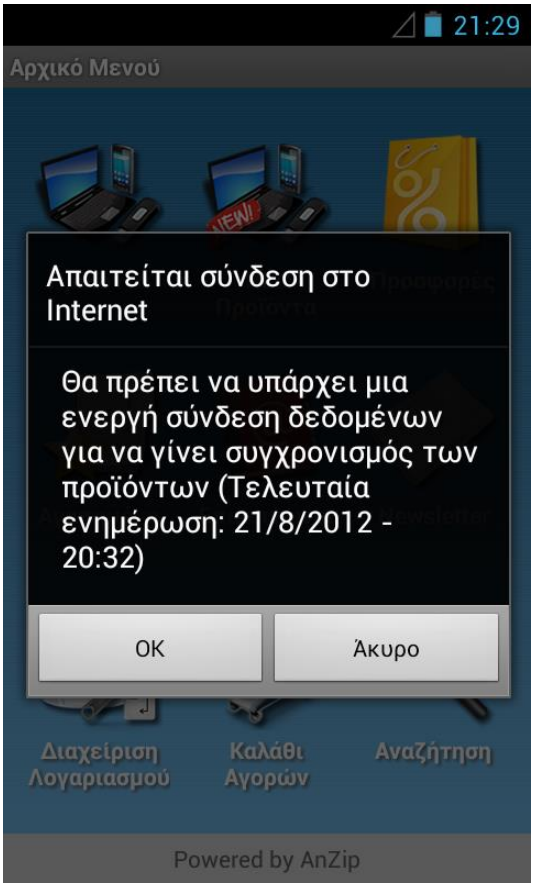
Το κουμπί συγχρονισμός κάνει ουσιαστικά μία ανανέωση των προϊόντων που βρίσκονται αποθηκευμένα στη συσκευή μας. Συγχρονίζει δηλαδή την συσκευή μας με το ηλεκτρονικό κατάστημα έτσι ώστε να μπορούμε να βλέπουμε νέα προϊόντα που έχουν προστεθεί και αλλαγές που έχουν γίνει από τον διαχειριστή σε υπάρχοντα προϊόντα. Το ίδιο φυσικά ισχύει και για τις κατηγορίες. Επίσης μας ενημερώνει για το πότε έγινε ο τελευταίος επιτυχής συγχρονισμός. Για να γίνει ο συγχρονισμός απαιτείται σύνδεση στο Internet.

Το κουμπί About μας εμφανίζει πληροφορίες για την εφαρμογή ενώ το κουμπί Exit τερματίζει την εφαρμογή.

Στιγμιότυπα των παραθύρων εμφάνισης καταστημάτων και του συγχρονισμού μπορούμε να δούμε παρακάτω:



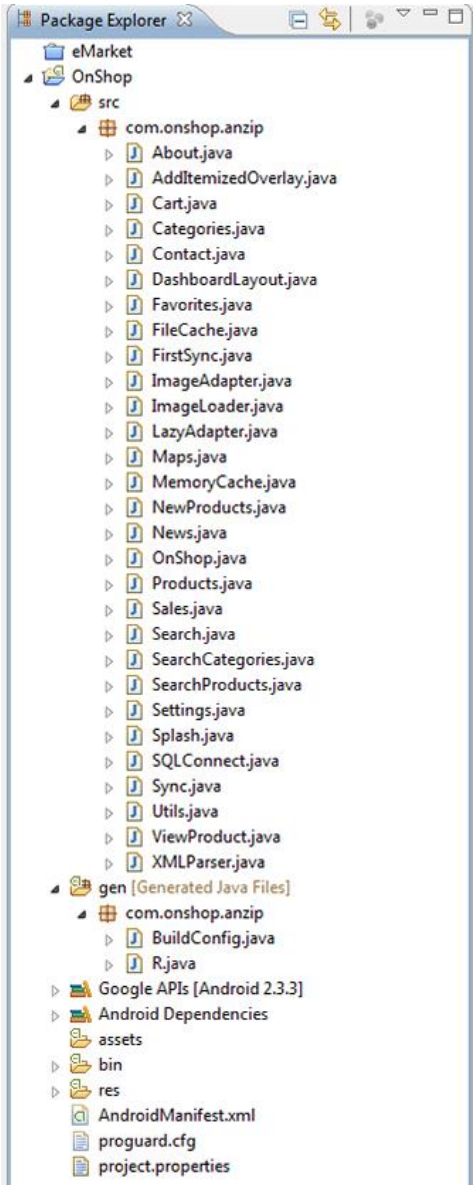
Εικόνα XLII : Καταστήματα



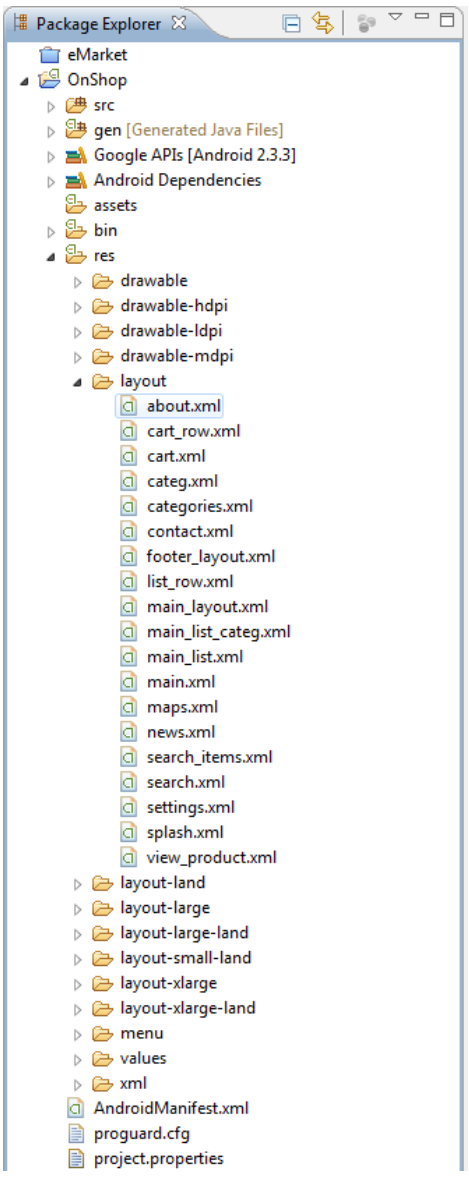
Εικόνα XXXVIII : Συγχρονισμός

11. Ανάλυση Κώδικα Java

Στο κεφάλαιο αυτό θα γίνει πλήρης ανάλυση του κώδικα που αναπτύχθηκε. Θα επεξηγηθούν τα σημαντικά κομμάτια κώδικα καθώς και οι μέθοδοι που καλούνται για την επίτευξη των διαφόρων λειτουργιών της εφαρμογής. Η ανάλυση θα γίνει σε δύο μέρη. Στο πρώτο μέρος θα αναλυθούν τα αρχεία που περιέχουν τον κώδικά σε Java τα οποία είναι αυτά που ουσιαστικά εκτελούν όλες τις λειτουργίες ενώ στο δεύτερο κομμάτι θα παραθέσουμε και θα αναλύσουμε τα αρχεία XML που χρησιμοποιήθηκαν για την δημιουργία του γραφικού περιβάλλοντος της εφαρμογής. Σε αυτό το σημείο κρίνεται σκόπιμο να δείξουμε όλα τα αρχεία τύπου Java και όλα τα αρχεία XML της εφαρμογής χρησιμοποιώντας εικόνες από την διαχείριση αρχείων του προγράμματος Eclipse.



Εικόνα XLIV : Αρχεία Java



Εικόνα XLV : Αρχεία XML

Πρωτού ξεκινήσουμε την ανάλυση θα γίνει μια αναφορά στους φακέλους που χρησιμοποιήθηκαν από την ιεραρχία την οποία φαίνεται στις εικόνες αυτές, έτσι ώστε να γνωρίζουμε τι τύπος αρχείων βρίσκεται στον κάθε φάκελο και για ποιο λόγο χρησιμοποιείται.

Στον κατάλογο src υπάρχουν όλα τα αρχεία τύπου java που έχουμε δημιουργήσει για την εφαρμογή μας και υπάγονται στο πακέτο το οποίου το όνομα το είχαμε δώσει αρχικά κατά τη δημιουργία του project. Το όνομα πακέτου πρέπει να το δώσουμε προσεκτικά έτσι ώστε να είναι μοναδικό, διότι κατά τη φάση της δημοσίευσης της εφαρμογής στο Google Play δεν θα είναι δυνατό να την δημοσιεύσουμε εάν το όνομα πακέτου που έχουμε δώσει υπάρχει και σε άλλη ήδη υπάρχουσα εφαρμογή. Είναι συνετό λοιπόν το όνομα αυτό να έχει σχέση με το όνομα της εφαρμογής και με τον δημιουργό της εφαρμογής ώστε να εξασφαλίσουμε την μοναδικότητα του.

Παρακάτω βλέπουμε τον κατάλογο gen μέσα στον οποίο υπάρχουν αρχεία java τα οποία δημιουργούνται αυτόματα. Πιο συγκεκριμένα το αρχείο BuildConfig.java περιέχει τις πληροφορίες για τη μεταγλώττιση του προγράμματος όπως ενώ το αρχείο R.java περιέχει μια αντιστοίχηση όλων των στοιχείων που υπάρχουν ως resources στην εφαρμογή μας όπως οι εικόνες ή τα αρχεία XML Layouts σε έναν ακέραιο. Δημιουργείται δηλαδή αυτόματα μια μεταβλητή τύπου int για κάθε αρχείο που βρίσκεται στον φάκελο res για τον οποίο θα μιλήσουμε λίγο παρακάτω, έτσι ώστε να μπορούμε να αναφερθούμε στο καθένα από αυτά μέσα από τον κώδικα μας.

Ο φάκελος Google APIs είναι ο φάκελος στον οποίο περιέχονται όλες οι βιβλιοθήκες του Android, αυτές που ουσιαστικά μας επιτρέπουν να προγραμματίζουμε σε αυτό, ενώ ο εικονικός φάκελος Android Dependencies είναι μια αναφορά σχετικά τις βιβλιοθήκες πάνω στις οποίες βασίζεται το project μας. Τέλος στον φάκελο res, αποθηκεύονται όλοι οι πόροι που υπάρχουν στην εφαρμογή μας όπως αρχεία xml, xml layouts, εικόνες, ήχοι και διάφορα αρχεία xml που εξυπηρετούν σκοπούς που θα αναλύσουμε και παρακάτω.

Έχοντας αναλύσει όλα τα παραπάνω, μπορούμε πλέον να περάσουμε στην ανάλυση του κώδικα των αρχείων java και έπειτα των αρχείων xml. Σε αυτό το σημείο αξίζει να σημειωθεί πως μερικά αρχεία είναι σχεδόν ίδια μεταξύ τους αφού εκτελούν παρόμοιες εργασίες. Σε τέτοιες περιπτώσεις απλά θα αναλύεται ένα εκ των αρχείων και θα αναφέρεται αυτό με το οποίο μοιάζουν παραθέτοντας και τις μεταξύ τους διαφορές. Επίσης να αναφέρουμε πως συνήθως για κάθε αρχείο πρώτα θα εμφανίζεται ο κώδικας και μετά η ανάλυση του.

11.1 About.java

```
package com.onshop.anzip;

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

public class About extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about);
```

```
        }
    }
```

Το πρώτο αρχείο που θα αναλύσουμε είναι το αρχείο που αναλαμβάνει την λειτουργία της επιλογής About στο αναδυόμενο μενού της εφαρμογής. Το αρχείο αυτό αποτελεί μια βάση θα λέγαμε για όλα τα αρχεία που θα δούμε παρακάτω για αυτό και ξεκινάμε με αυτό.

Στην πρώτη γραμμή του δηλώνεται ότι το αρχείο αυτό ανήκει στο πακέτο της εφαρμογής με όνομα com.onshop.anzip. Η γραμμή αυτή υπάρχει σε κάθε αρχείο java της εφαρμογής. Παρακάτω βλέπουμε τα διάφορα import άλλων κλάσεων του πακέτου του Android που πρέπει να κάνουμε έτσι ώστε να μπορέσουμε να χρησιμοποιήσουμε ενσωματωμένες μεθόδους και ιδιότητες των αντικειμένων των κλάσεων αυτών. Στο συγκεκριμένο παράδειγμα γίνεται import των κλάσεων Bundle και Activity για τη χρήση τους παρακάτω. Το κομμάτι αυτό δεν θα αναλύεται για κάθε αρχείο απλά θα παρατίθεται έτσι ώστε να ξέρουμε ποιες κλάσεις χρησιμοποιήθηκαν και για ποιον λόγο.

Οι περισσότερες κλάσεις που δημιουργούμε για την εφαρμογή επεκτείνουν την κλάση Activity, δίνοντας μας τη δυνατότητα να χρησιμοποιούμε μεθόδους της κλάσης αυτής, μία εκ των οποίων είναι η onCreate που λαμβάνει ως όρισμα ένα Bundle και καλείται όταν το Activity μας πάει να εκτελεστεί. Περισσότερες πληροφορίες σχετικά με αυτό υπάρχουν στην ενότητα 4.4 (κύκλος ζωής των Activities). Μέσα στην μέθοδο onCreate βλέπουμε δύο γραμμές που θα τις συναντούμε σε κάθε κλάση μας, είναι η super.onCreate(savedInstanceState); η οποία καλεί τον δομητή της κλάσης Activity με όρισμα το Bundle που δώσαμε στην onCreate της δικής μας κλάσης ενώ η επόμενη γραμμή χρησιμοποιεί μια πολύ χρήσιμη μέθοδο, την setContentView που παίρνει ως όρισμα ένα αρχείο layout από τα resources της εφαρμογής. Στη συγκεκριμένη περίπτωση το R.layout.about που αντιστοιχεί στο αρχείο about.xml στον φάκελο layout το οποίο θα αναλυθεί στο επόμενο κεφάλαιο. Έτσι λοιπόν με τη χρήση του απλού αυτού κομματιού κώδικα, δημιουργείται ένα νέο Activity, με γραφικό της περιβάλλον τα περιεχόμενα ενός αρχείου xml της επιλογής του προγραμματιστή.

11.2 AddItemizedOverlay.java

Η κλάση αυτή αναλαμβάνει την δημιουργία μιας εικονικής «επικάλυψης» επάνω από το Activity που μας εμφανίζει τους χάρτες της Google, για το οποίο θα μιλήσουμε παρακάτω. Επάνω στην επικάλυψη αυτή μπορούμε να προσθέσουμε διάφορα custom στοιχεία όπως σημεία ενδιαφέροντος σε σημεία τα οποία θέλουμε. Στην περίπτωση μας, τις διευθύνσεις των καταστημάτων που επιθυμούμε στα αντίστοιχα σημεία. Επειδή είναι αρκετά μεγάλη κλάση και τα περισσότερα στοιχεία της είναι δομητές καθώς και μέθοδοι αλλαγής/ανάγνωσης, θα αναφέρουμε μόνο τις σημαντικότερες μεθόδους της που αφορούν την εφαρμογή μας. Το ίδιο θα κάνουμε και για τις υπόλοιπες κλάσεις που είναι μεγάλες.

```
@Override
protected boolean onTap(int index) {
        Toast.makeText(this.context, mapOverlays.get(index).getTitle() + "\n" +
        mapOverlays.get(index).getSnippet(), Toast.LENGTH_LONG).show();
    return(true);
}
```

Η μέθοδος αυτή αναλαμβάνει να μας εμφανίζει τα αντίστοιχα μηνύματα κάθε φορά που κάνουμε κλικ που πάνω σε ένα σημείο ενδιαφέροντος. Εμφανίζεται λοιπόν ένα Toast, το οποίο είναι ένα πλαίσιο που μας εμφανίζει ένα σύντομο μήνυμα και ανάλογα με το ποιο από τα σημεία επιλέχθηκε. Τα ορίσματα της μεθόδου makeText της κλάσης Toast είναι το Context, δηλαδή σε ποιο Activity ανήκει το Toast, το κείμενο που θέλουμε να εμφανίζεται που στην συγκεκριμένη περίπτωση είναι το title και το snippet του αντικειμένου που επιλέξαμε στον χάρτη και η διάρκεια εμφάνισης του μηνύματος που εδώ έχουμε εμφάνιση για αρκετή ώρα (LENGTH_LONG).

Η επόμενη μέθοδος προς ανάλυση είναι η onTouchEvent η οποία αναλαμβάνει την διαχείριση του γεγονότος του κλικ πάνω στην επικάλυψη που δημιουργήσαμε και είναι η εξής:

```
@Override
public boolean onTouchEvent(MotionEvent event, MapView mapView) {
   switch (event.getAction()) {
   case MotionEvent.ACTION_DOWN:
if ((System.currentTimeMillis() - systemTime) < ViewConfiguration.getDoubleTapTimeout()) {
mapView.getController().zoomInFixing((int) event.getX(), (int)event.getY()); }
      systemTime = System.currentTimeMillis();
      break;
   }

   return false;

}
```

Αυτό που κάνει ουσιαστικά η μέθοδος αυτή, είναι να κάνει zoom in στο συγκεκριμένο σημείο που έγινε το διπλό κλικ. Αυτό επιτυγχάνεται με την γραμμή

```
mapView.getController().zoomInFixing((int) event.getX(), (int)event.getY());
```

Παίρνουμε δηλαδή το οριζόντιο και κάθετο σημείο που πιέστηκε στην οθόνη και λέμε στον controller του Mapview μας να κάνει ζουμ στα σημεία αυτά. Επίσης επειδή πρέπει η διαδικασία αυτή να γίνεται στο διπλό κλικ, ελέγχουμε μέσω της γραμμής

```
if ((System.currentTimeMillis() - systemTime)
```

Αν το τελευταίο κλικ έγινε σύντομα σε σχέση με το προηγούμενο τότε εκτελούμε κανονικά τις ενέργειες που θέλουμε και στο τέλος, αφού γίνει το zoom in αποθηκεύουμε την τιμή που έγινε το πρώτο κλικ σε μια προσωρινή μεταβλητή για να γίνει επιτυχώς ο έλεγχος

```
systemTime = System.currentTimeMillis();
```

11.3 Cart.java

Η κλάση αυτή αναλαμβάνει να μας δείξει τα περιεχόμενα του καλαθιού καθώς και να διαχειριστεί τα γεγονότα της διαγραφής προϊόντων από αυτό καθώς και τον υπολογισμό του υποσυνόλου του κάθε προϊόντος με βάση την ποσότητα του και του γενικού συνόλου του καλαθιού. Αρχικά βλέπουμε την δήλωση της κλάσης και μερικών μεταβλητών. Ας δούμε πρώτα τι είδους μεταβλητές έχουμε δηλώσει και να πούμε δύο λόγια για την χρήση τους αργότερα μέσα στην κλάση.

```
public class Cart extends Activity {

	private View vi;
	private static LayoutInflater inflater=null;
	private SQLConnect addSQLtoList = new SQLConnect(this);
	private TableLayout cartTable;
	private TextView cartTotal;
	private int sum = 0;
```

Έχουμε λοιπόν δηλώσει μια μεταβλητή τύπου View που θα χρησιμοποιείται για να δημιουργούμε Views δυναμικά μέσα στην κλάση μας. Πρακτικά αυτό σημαίνει πως για κάθε προϊόν μέσα στο καλάθι μας υπάρχει ένα δηλωμένο από εμάς είδος view μέσα στα αρχεία xml layouts και συγκεκριμένα στο cart_row.xml, το οποίο περιέχει όλα τα στοιχεία που χρειαζόμαστε σε κάθε γραμμή του καλαθιού μας. Κάθε φορά που δημιουργείται μία νέα γραμμή η μεταβλητή vi παίρνει τις τιμές για κάθε στοιχείο (τίτλος, ποσότητα, τιμή) του αντίστοιχου προϊόντος και μετά προστίθεται στη λίστα με τα περιεχόμενα του καλαθιού Ακολουθεί ένας LayoutInflater που αναλαμβάνει να προσθέσει το view που δημιουργήθηκε στη λίστα με τα περιεχόμενα που ήδη υπάρχουν, δηλαδή στην μεταβλητή τύπου TableLayout με όνομα cartTable. Έπειτα βλέπουμε ένα αντικείμενο της κλάσης SQLConnect που δημιουργούμε για να επικοινωνούμε με την βάση δεδομένων μας. Τέλος ένα TextView με όνομα cartTotal που εμφανίζει το σύνολο του καλαθιού και μια μεταβλητή integer με όνομα sum που χρησιμεύει στις αριθμητικές πράξεις για τον υπολογισμό του συνόλου.

Συνεχίζουμε με την προσπέλαση της βάσης και την κλήση της μεθόδου getCartContents που επιστρέφει τα περιεχόμενα του καλαθιού και την αποθήκευση τους σε μια τοπική μεταβλητή ArrayList. Επίσης γίνεται έλεγχος για το αν το καλάθι δεν περιέχει τίποτα και εμφανίζεται ανάλογο μήνυμα.

Σε αυτό το σημείο να αναλύσουμε μια γραμμή κώδικα που θα συναντήσουμε πολύ συχνά. Πρόκειται για την γραμμή cartTotal = (TextView)findViewById(R.id.*tvCartTotal*); Η γραμμή αυτή αντιστοιχίζει μια μεταβλητή που δηλώσαμε στην κλάση μας με ένα από τα resources της εφαρμογής με τη χρήση της μεθόδου findViewById που παίρνει ως όρισμα το id ενός από τα resources. Εδώ για παράδειγμα έχουμε μια μεταβλητή τύπου TextView με όνομα cartTotal και αντιστοιχίζεται με το TextView με id tvCartTotal που υπάρχει στα αρχεία xml layout. Αυτό θα το συναντάμε πολύ συχνά για διάφορους τύπους Views όπως Buttons, TextViews, CheckBox και TableLayout.

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

		addSQLtoList.open();
		ArrayList<HashMap<String, String>> cartData = addSQLtoList.getCartContents();
		addSQLtoList.close();

		cartTable = (TableLayout)findViewById(R.id.cartTable);
		inflater = (LayoutInflater)this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
		cartTotal = (TextView)findViewById(R.id.tvCartTotal);
```

```
if (cartData.size()==0) {
        TextView tv = new TextView(Cart.this);
        tv.setText("Δεν υπάρχουν προϊόντα στο καλάθι αγορών");
        tv.setPadding(15, 15, 15, 15);
        tv.setTextColor(Color.DKGRAY);
        cartTable.addView(tv);
}
```

Από το σημείο αυτό και έπειτα γίνεται η ανάγνωση των δεδομένων από την βάση μας και η προσθήκη τους στο TableLayout που περιέχει μια μία τις γραμμές κάθε προϊόντος. Για κάθε στοιχείο του cartData που περιέχει όλες τις εγγραφές από τη βάση δημιουργείται ένα view με τις αντίστοιχες τιμές για τα πεδία τίτλος, τιμή, ποσότητα και υποσύνολο που υπολογίζεται βάση των δύο τελευταίων. Επίσης υπολογίζεται και το σύνολο με τη βοήθεια της μεταβλητής sum.

```
for (int i=0; i<cartData.size(); i++) {

HashMap<String, String> cart = new HashMap<String, String>();
cart = cartData.get(i);

vi = inflater.inflate(R.layout.cart_row,null);

   TextView cartTitle = (TextView)vi.findViewById(R.id.tvCartTitle);
   TextView cartQuan = (TextView)vi.findViewById(R.id.tvCartQuan);
   TextView cartSubTotal = (TextView)vi.findViewById(R.id.tvCartSubTotal);
   Button cartRemove = (Button)vi.findViewById(R.id.btnCartRemove);

   cartTitle.setText(cart.get("prodName"));
cartQuan.setText(cart.get("prodQuantity"));

int price=0;
int quantity=0;
int subTotal=0;

price = Integer.parseInt(cart.get("prodPrice"));
quantity = Integer.parseInt(cart.get("prodQuantity"));
subTotal = quantity*price;

cartSubTotal.setText(Integer.toString(subTotal)+" €");
cartRemove.setTag(cart.get("prodWebID"));
cartRemove.setTag(R.id.btnProdSubTotal, subTotal);

sum = sum + subTotal;
```

Τέλος για την κλάση αυτή έχουμε την μέθοδο διαχείρισης του γεγονότος του κλικ στο κουμπί αφαίρεσης από το καλάθι κάθε προϊόντος. Αφαιρείται λοιπόν από την βάση μας το προϊόν που επιλέχθηκε προς αφαίρεση και έπειτα γίνεται μια επανεκκίνηση του Activity για να ανανεωθεί η τιμή του συνόλου και να αφαιρεθεί το προϊόν και από τη λίστα με ta views των προϊόντων.

```
cartRemove.setOnClickListener(new View.OnClickListener() {

   @Override
   public void onClick(View v) {
```

```
                addSQLtoList.open();
                addSQLtoList.deleteFromCart((String)v.getTag());
                addSQLtoList.close();

                Intent i = getIntent();
                finish();
                startActivity(i);

                }
        });

        cartTable.addView(vi);

        }

        cartTotal.setText("Σύνολο: "+ Integer.toString(sum)+ " €");

        }
}
```

11.4 Categories.java

Η κλάση αυτή αναλαμβάνει να εμφανίσει στον χρήστη όλες τις κατηγορίες προϊόντων που υπάρχουν αποθηκευμένες στην βάση δεδομένων μόλις αυτός κάνει κλικ στο κουμπί Προϊόντα του κεντρικού μενού. Και εδώ όπως και στην κλάση Cart, χρησιμοποιούμε ένα τύπο View που έχει δημιουργηθεί από εμάς για να εμφανίζονται τα τετράγωνα με τις κατηγορίες όπως επιθυμούμε. Στην συγκεκριμένη περίπτωση έχουμε δημιουργήσει έναν δικό μας adapter που αναλαμβάνει να προσθέσει τα δεδομένα από ένα ArrayList με εγγραφές της βάσης δεδομένων σε ένα GridView. Πρώτα γίνεται η δήλωση των μεταβλητών που θα χρειαστούν και είναι το GridView και ο ImageAdapter.

```
public class Categories extends Activity {

        private GridView categories;
        private ImageAdapter adapter;

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

Πιο κάτω βλέπουμε την εμφάνιση ενός διαλόγου προόδου με το μήνυμα Φόρτωση κατηγοριών... που εμφανίζεται μέχρι να φορτωθούν όλα τα προϊόντα. Πιο κάτω βλέπουμε με την ίδια διαδικασία όπως και στην κλάση Cart να παίρνουμε τα δεδομένα από την βάση. Εδώ βέβαια καλούμε την μέθοδο categoriesToGrid με όρισμα 0. Η μέθοδος αυτή παίρνει σαν όρισμα το id της μητρικής κατηγορίας που ανήκει η κατηγορία και εφόσον θέλουμε τις αρχικές κατηγορίες που δεν έχουν κάποια μητρική το όρισμα αυτό είναι 0. Στη συνέχεια περνάμε τις εγγραφές ως όρισμα στον adapter που έχουμε φτιάξει, τον ορίζουμε ως adapter του GridView μας και αυτός αναλαμβάνει την προσθήκη των views σε αυτό. Μόλις ολοκληρωθεί η διαδικασία ο διάλογος προόδου εξαφανίζεται.

```
ProgressDialog dialog = ProgressDialog.show(Categories.this,"", "Φόρτωση κατηγοριών...", true);
SQLConnect addSQLtoGrid = new SQLConnect(this);
addSQLtoGrid.open();
ArrayList<HashMap<String, String>> categoriesData = addSQLtoGrid.categoriesToGrid("0");
addSQLtoGrid.close();

categories = (GridView) findViewById(R.id.gvCategories);
adapter = new ImageAdapter(Categories.this, categoriesData);

if (categoriesData.size()==0) {
TextView tv = new TextView(Categories.this);
tv.setText("Δεν υπάρχουν κατηγορίες");
tv.setPadding(15, 15, 15, 15);
tv.setTextColor(Color.DKGRAY);
categories.setEmptyView(tv);
}

categories.setAdapter(adapter);

dialog.dismiss();
```

Αφού γίνουν όλα τα παραπάνω θα πρέπει να ορίσουμε τι θα γίνεται όταν ο χρήστης κάνει κλικ πάνω στο κουμπί μιας κατηγορίας. Έτσι λοιπόν υπερβαίνουμε την μέθοδο OnItemClick μέσα στην οποία παίρνουμε το id της κατηγορίας στην οποία έγινε κλικ και έπειτα ξεκινάμε την κλάση Products δημιουργώντας ένα intent από την τρέχουσα κλάση προς την κλάση Products και προσθέτοντας σε αυτό ως extra το id, που θα χρειαστεί αργότερα η κλάση Products για να λειτουργήσει, πράγμα που θα δούμε παρακάτω.

```
categories.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String cID = ((TextView) view.findViewById(R.id.tvCategoryID)).getText().toString();
Intent in = new Intent(getApplicationContext(), Products.class);
in.putExtra("catID", cID);
startActivity(in);
                }
        });
}
}
```

11.5 Contact.java

Επόμενη κλάση είναι η αυτή που έχει να κάνει με το μενού επικοινωνίας και τις ενέργειες που γίνονται από αυτή. Σε πρώτη φάση, δηλώνονται όλα τα κουμπιά που δίνουν τις διαθέσιμες επιλογές και έπειτα ορίζονται οι ενέργειες που θα γίνονται στην περίπτωση του κλικ. Γίνεται λοιπόν αρχικά η δήλωση των μεταβλητών και η σύνδεση τους με τα αντίστοιχα resources όπως περιγράψαμε πιο πάνω.

```
public class Contact extends Activity implements OnClickListener{

	private Button link, send_mail, dial;

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

	link = (Button) findViewById(R.id.facebook);
	link.setOnClickListener(this);

	send_mail = (Button) findViewById(R.id.mail);
	send_mail.setOnClickListener(this);

	dial = (Button) findViewById(R.id.phone);
	dial.setOnClickListener(this);

	}
```

Στο σημείο αυτό γίνεται εξακρίβωση σε ποιο View έγινε το κλικ, με βάση το id του καθενός. Έπειτα ανάλογα με το τι θέλουμε να κάνουμε καλούμε κάποια από τις ήδη υπάρχουσες εφαρμογές στη συσκευή μας να αναλάβει την κάθε ενέργεια. Πρώτα έχουμε το άνοιγμα μιας σελίδας στο Facebook και έτσι καλούμε τον browser του android προς εκκίνηση, δίνοντας του ένα όρισμα για το πιο url θα ανοίξει ως δεδομένο μέσα στο intent που δημιουργήσαμε. Το ίδιο ισχύει και για την αποστολή e-mail και για την κλήση ενός τηλεφωνικού αριθμού.

```
	public void onClick(View v) {
		switch (v.getId()) {

		case R.id.facebook:
			String url = "http://www.facebook.com";
			Intent fbIntent = new Intent(android.content.Intent.ACTION_VIEW);
			fbIntent.setData(Uri.parse(url));
			startActivity(fbIntent);
			break;

		case R.id.mail:
			String emailaddress = "dzipel@gmail.com";
			Intent mailIntent = new Intent(android.content.Intent.ACTION_SEND);
			//mailIntent.setData(Uri.parse(emailaddress));
			mailIntent.putExtra(Intent.EXTRA_EMAIL, emailaddress);
			mailIntent.setType("plain/text");
			startActivity(mailIntent);
			break;

		case R.id.phone:
	String number = "tel:6938523638";
	Intent callIntent = new Intent(Intent.ACTION_CALL, Uri.parse(number));
	startActivity(callIntent);
			break;

		}
```

```
    }
}
```

11.6 DashboardLayout.java

Η κλάση αυτή έχει δημιουργηθεί από την Google και μπορεί να χρησιμοποιηθεί από προγραμματιστές εφαρμογών android. Δεν θα αναλυθεί απλά θα αναφέρουμε τι πετυχαίνουμε με τη χρήση της. Όπως δείχνει και το όνομα της η κλάση αυτή είναι υπεύθυνη για τη δημιουργία του Layout για το Dashboard της εφαρμογής μας, που στην περίπτωση μας είναι το αρχικό μενού επιλογών. Μέσα λοιπόν σε αυτή την κλάση υπάρχουν υπολογισμοί οι οποίοι τοποθετούν σωστά όλα τα αντικείμενα που υπάρχουν στο dashboard που στην περίπτωση μας είναι μόνο κουμπιά, με βάση το μέγεθος της οθόνης που έχουμε και τον αριθμό τον αντικειμένων που υπάρχουν. Έτσι η στοίχιση παραμένει σωστή ακόμα και όταν προσθέσουμε ή αφαιρέσουμε κάποιο στοιχείο, σε όλα τα διάφορα μεγέθη οθονών.

11.7 Favorites.java

Στην κλάση Favorites γίνεται η εμφάνιση των προϊόντων που ο χρήστης έχει ορίσει ως αγαπημένα. Τα δεδομένα έρχονται από την τοπική βάση και τοποθετούνται σε μορφή λίστας. Επίσης υπάρχει η δυνατότητα ταξινόμησης τους και θα δούμε πως επιτυγχάνεται αυτό. Αρχικά δηλώνουμε μία μεταβλητή τύπου ListView και έναν adapter για την λίστα αυτή. Επίσης έχουμε ένα String που περιέχει τον τρόπο με τον οποίο θα ταξινομήσουμε τα στοιχεία της λίστας. Επίσης γίνεται ένας έλεγχος για το αν η λίστα είναι κενή η όχι και εμφανίζεται κατάλληλο μήνυμα στην πρώτη περίπτωση. Υπάρχουν επίσης ένας διάλογος προόδου κατά τη διαδικασία της φόρτωσης και η αντίστοιχη κλήση της μεθόδου της βάσης favoritesToList που έχουν αναφερθεί και σε προηγούμενο υποκεφάλαιο.

```
public class Favorites extends Activity {

    private ListView list;
    private LazyAdapter adapter;
    String order;

    @Override
    protected void onCreate(Bundle savedInstanceState) {

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

        Intent in = getIntent();
        String order = "";

        if (in.getExtras()!=null)
```

```
order = in.getStringExtra("order");

ProgressDialog dialog = ProgressDialog.show(Favorites.this, "", "Φόρτωση προϊόντων...",
true);

dialog.show();

SQLConnect addSQLtoList = new SQLConnect(this);

addSQLtoList.open();
ArrayList<HashMap<String, String>> listData = addSQLtoList.favoritesToList(order);
addSQLtoList.close();
```

Στο σημείο αυτό θέτουμε ως όρισμα για τον adapter της λίστας το σύνολο δεδομένων που έχουμε πάρει από την τοπική βάση ενώ πιο κάτω γίνεται η διαχείριση του γεγονότος κλικ πάνω σε κάποιο στοιχείο της λίστας από το οποίο παίρνουμε όλα τα δεδομένα που αντιστοιχούν σε αυτό (id, sku, τίτλος, περιγραφή, τιμή, εικόνα) έτσι ώστε να τα τοποθετήσουμε ως επιπλέον δεδομένα στο intent του Activity το οποίο ξεκινάμε για την αναλυτική προβολή ενός προϊόντος. Το Activity αυτό αναλύεται πιο κάτω.

```
list = (ListView) findViewById(R.id.list);
adapter = new LazyAdapter(this, listData);
list.setEmptyView(findViewById(R.id.emptylist));
list.setAdapter(adapter);

list.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String pWebID = ((TextView)view.findViewById(R.id.tvListProdWebID)).getText().toString();
String pSKU = ((TextView) view.findViewById(R.id.tvListProdSKU)).getText().toString();
String pName = ((TextView) view.findViewById(R.id.tvListProdName)).getText().toString();
String pDesc = ((TextView) view.findViewById(R.id.tvListProdDesc)).getText().toString();
String pPrice = ((TextView) view.findViewById(R.id.tvListProdPrice)).getText().toString();
String pImage = ((TextView) view.findViewById(R.id.tvListProdImage)).getText().toString();

Intent in = new Intent(getApplicationContext(), ViewProduct.class);

in.putExtra("prodWebID", pWebID);
in.putExtra("prodSKU", pSKU);
in.putExtra("prodName", pName);
in.putExtra("prodDesc", pDesc);
in.putExtra("prodPrice", pPrice);
in.putExtra("prodImage", pImage);

startActivity(in);
        }
});

dialog.dismiss();
}
```

Πιο κάτω υπάρχει η υπέρβαση της μεθόδου onResume της κλάσης Activity, που συμβαίνει όταν ένα Activity κάνει επανεκκίνηση. Σε αυτή τη μέθοδο κάνουμε έλεγχο της μεταβλητής order

στην οποία είχαμε αναφερθεί πιο πριν έτσι ώστε να την δώσουμε ως όρισμα στην κλήση της μεθόδου πρόσβασης στην βάση δεδομένων και να μας επιστραφούν οι εγγραφές με την επιθυμητή ταξινόμηση. Έπειτα τοποθετούνται και πάλι στη λίστα με την σειρά που επιλέξαμε. Η επιλογή αυτή γίνεται πιο κάτω με τη χρήση ενός αναδυόμενου μενού.

```
@Override
        protected void onResume() {
        // TODO Auto-generated method stub
        super.onResume();

        Intent in = getIntent();
        order = "";

        if (in.getExtras()!=null)
        order = in.getStringExtra("order");

        SQLConnect addSQLtoList = new SQLConnect(this);
        addSQLtoList.open();
        ArrayList<HashMap<String, String>> listData = addSQLtoList.favoritesToList(order);
        addSQLtoList.close();

        list = (ListView) findViewById(R.id.list);
        adapter = new LazyAdapter(this, listData);
        list.setEmptyView(findViewById(R.id.emptylist));

        list.setAdapter(adapter);
        }
```

Εδώ βλέπουμε την μέθοδο που καλείται κατά την στιγμή που ένα αναδυόμενο μενού επιλογών δημιουργείται, συνήθως πιέζοντας το πλήκτρο του μενού στη συσκευή. Υπάρχουν τέσσερα αντικείμενα στο μενού αυτό που ταξινομούν τη λίστα με βάση το όνομα ή το κόστος, με αύξουσα ή φθίνουσα ταξινόμηση για κάθε ένα. Στην μέθοδο onCreateOptionsMenu, ορίζουμε ουσιαστικά ποιο αρχείο XML Menu θέλουμε να χρησιμοποιηθεί για το μενού μας, ενώ πιο κάτω η μέθοδος OnMenuItemSelected είναι αυτή που καλείται όταν ο χρήστης επιλέξει ένα αντικείμενο του μενού. Έπειτα με βάση το id του γίνεται η κατάλληλη ενέργεια, αλλάζει δηλαδή η τιμή της μεταβλητής order και στην συνέχεια επανεκκινείται το Activity για να ανανεωθεί η λίστα. Για παράδειγμα αν γίνει κλικ στο στοιχείο με id sortbynamea σημαίνει πως θέλουμε να γίνει αύξουσα ονομαστική ταξινόμηση και η τιμή της μεταβλητής order γίνεται ORDER BY p.name ASC. Αντίστοιχες ενέργειες γίνονται και για τα υπόλοιπα κουμπιά του μενού.

```
@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {

        super.onCreateOptionsMenu(menu);
        MenuInflater minf = getMenuInflater();
        minf.inflate(R.menu.sortmenu, menu);
        return true;
}

@Override
public boolean onMenuItemSelected(int id, MenuItem item) {
        switch (item.getItemId()) {
```

```
                case R.id.sortbynamea:
                        Intent in = new Intent(getApplicationContext(), Favorites.class);
                        in.putExtra("order", "ORDER BY p.name ASC");
                        finish();
                        startActivity(in);
                        break;

                case R.id.sortbypricea:
                        Intent in2 = new Intent(getApplicationContext(), Favorites.class);
                        in2.putExtra("order", "ORDER BY p.price_with_VAT ASC");
                        finish();
                        startActivity(in2);
                        break;

                case R.id.sortbynamed:
                        Intent in3 = new Intent(getApplicationContext(), Favorites.class);
                        in3.putExtra("order", "ORDER BY p.name DESC");
                        finish();
                        startActivity(in3);
                        break;

                case R.id.sortbypriced:
                        Intent in4 = new Intent(getApplicationContext(), Favorites.class);
                        in4.putExtra("order", "ORDER BY p.price_with_VAT DESC");
                        finish();
                        startActivity(in4);
                        break;
                }
                return false;
        }

}
```

11.8 FileCache.java

Η κλάση αυτή αναλαμβάνει την τοπική αποθήκευση των αρχείων εικόνων έτσι ώστε να μπορούν να εμφανίζονται και χωρίς να υπάρχει μία σύνδεση δεδομένων. Έχει 2 βασικές μεθόδους την getFile και την clear οι οποίες αναλαμβάνουν την εμφάνιση ενός αρχείου με βάση το url που έχει στο web και την εκκαθάριση όλων των αρχείων της μνήμης cache αντίστοιχα. Στον δομητή της κλάσης υπάρχει έλεγχος για το αν υπάρχει κάρτα μνήμης στο κινητό μέσω της γραμμής:

```
if (android.os.Environment.getExternalStorageState().equals(android.os.Environment.MEDIA_MOUNTED))
```

Και έπειτα δημιουργεί έναν φάκελο με όνομα OnShop για την αποθήκευση των αρχείων αν αυτός δεν υπάρχει.

```
public class FileCache {

   private File cacheDir;

   public FileCache(Context context){

   if (android.os.Environment.getExternalStorageState().equals(android.os.Environment.MEDIA_MOUNTED))
         cacheDir=new File(android.os.Environment.getExternalStorageDirectory(),"OnShop");
```

```
        else
            cacheDir=context.getCacheDir();
        if(!cacheDir.exists())
            cacheDir.mkdirs();
    }

    public File getFile(String url){
        // Onomazoume ta arxeia me basi to hashcode
        String filename=String.valueOf(url.hashCode());
        File f = new File(cacheDir, filename);
        return f;

    }

    public void clear(){
        File[] files=cacheDir.listFiles();
        if(files==null)
            return;
        for(File f:files)
            f.delete();
    }
}
```

11.9 FirstSync.java

Η FirstSync πρόκειται για την κλάση που αναλαμβάνει τον πρώτο συγχρονισμό των προϊόντων του ηλεκτρονικού καταστήματος με αυτά στην συσκευή. Είναι μια αρκετά μεγάλη κλάση αλλά αξίζει να αναλυθεί ολόκληρη καθώς εκτελεί μία από τις σημαντικότερες λειτουργίες για την εφαρμογή. Στο πρώτο στάδιο δηλώνουμε τα url τα οποία μας επιστρέφουν μετά από την κλήση ενός web service, τα αρχεία XML που μπορούμε να περάσουμε από την διαδικασία του Parsing ώστε να κρατήσουμε τα δεδομένα που χρειαζόμαστε από αυτό. Επίσης μαζί με την δήλωση των url αυτό δηλώνουμε σαν μεταβλητές String όλα τα ονόματα των elements και των attributes που μας ενδιαφέρουν και εμφανίζονται στο αρχείο XML. Επίσης δημιουργείται και ένα αντικείμενο τύπου SQL Connect για την απευθείας αποθήκευση των δεδομένων των XML στην τοπική βάση.

```
public class FirstSync extends Activity {

        private static final String CatURL="http://www.caretta-
net.gr/Default.aspx?m=getAllProductCategories";
        private static final String KEY_CATEGORY = "Category";
        private static final String KEY_CATEGORY_ID = "ID";
        private static final String KEY_CATEGORY_NAME = "Name";
        private static final String KEY_CATEGORY_FULLIMAGE = "FullImage";
        private static final String KEY_CATEGORY_THUMBNAIL = "Thumbnail";
        private static final String KEY_CATEGORY_PARENTID = "ParentID";

        private static final String ProdURL="http://www.caretta-net.gr/default.aspx?m=getProducts";
        private static final String KEY_PRODUCT = "Product";
        private static final String KEY_PRODUCT_WEBID = "WebID";
        private static final String KEY_PRODUCT_SKU = "SKU";
```

```
private static final String KEY_PRODUCT_NAME = "Name";
private static final String KEY_PRODUCT_SHORTDESC = "shortDescription";
private static final String KEY_PRODUCT_DESC = "Description";
private static final String KEY_PRODUCT_PRICE = "Price_with_VAT";
private static final String KEY_PRODUCT_FULLIMAGE = "productImage";
private static final String KEY_PRODUCT_THUMB = "thumbnail";
private static final String KEY_PRODUCT_CATEGORYID = "ID";

private SQLConnect addCatData = new SQLConnect(FirstSync.this);
private SQLConnect addProductData = new SQLConnect(FirstSync.this);

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

Μετά από τις δηλώσεις ξεκινά η διαδικασία του Parsing με την εμφάνιση ενός διαλόγου προόδου που εμφανίζεται για όσο διαρκεί η διαδικασία. Σε αυτό το σημείο πρέπει να σημειωθεί πως οι χρονοβόρες διαδικασίες όπως το parsing, πρέπει να εκτελούνται σε ένα νήμα διαφορετικό από το νήμα το οποίο αναλαμβάνει το γραφικό περιβάλλον της εφαρμογής. Για να γίνει αυτό χρησιμοποιείται η κλάση AsyncTask. Αρχικά δημιουργούμε μια κλάση που επεκτείνει την AsyncTask και δηλώνουμε 3 παραμέτρους. Η πρώτη δείχνει τι τύπου αποτέλεσμα θα δώσει εκτέλεση, η δεύτερη τι τύπου μεταβλητή θα δώσουμε στην μέθοδο η οποία αναλαμβάνει την προσωρινή παύση της εκτέλεσης του νήματος και η τελευταία παράμετρος δείχνει τι τύπου μεταβλητή θα επιστρέψει η μέθοδος που εκτελείται μόλις ολοκληρωθεί η εκτέλεση του νήματος. Στην περίπτωση μας είναι και οι τρεις void. Μέσα στην κλάση που δημιουργούμε υπερβαίνουμε την μέθοδο doInBackground. Σε αυτή προσθέτουμε όλες τις χρονοβόρες διαδικασίες που πρόκειται να εκτελεστούν στο ξεχωριστό νήμα. Τέλος εκτελούμε την διαδικασία αυτή χρησιμοποιώντας την μέθοδο execute.

```
final ProgressDialog dialog = ProgressDialog.show(FirstSync.this, "", "Συγχρονισμός προϊόντων...",
true);
dialog.show();

//Categories Sync
class GetCategories extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... params) {

        addCatData.open();
```

Αφότου ανοίξουμε την βάση προς εγγραφή με την παραπάνω γραμμή, δημιουργούμε ένα αντικείμενο τύπου XMLParser και το συσχετίζουμε με το URL που έχουμε δηλώσει στην αρχή. Έπειτα επιλέγουμε το αρχικό element από το οποίο θα πάρουμε τα child nodes και attributes. Για κάθε στοιχείο της NodeList που έχουμε, παίρνουμε τα στοιχεία που θέλουμε με την μέθοδο getValue αν πρόκειται για μια τιμή μεταξύ δύο elements ή την μέθοδο getAtribute αν πρόκειται για κάποια ιδιότητα ενός από τα elements.

```
XMLParser parser = new XMLParser();
String xml = parser.getXmlFromUrl(CatURL);
Document doc = parser.getDomElement(xml);
NodeList nl = doc.getElementsByTagName(KEY_CATEGORY);
```

```
if (nl.getLength() == 0)
return null;

        // looping through all product nodes
        for (int i = 0; i < nl.getLength(); i++) {
        // creating new HashMap
        Element e = (Element) nl.item(i);
        String correctURL="";

        if (!e.getAttribute(KEY_CATEGORY_THUMBNAIL).startsWith("http")){
        correctURL = "http://www.caretta-net.gr/demos/e-
        shop/components/com_virtuemart/shop_image/category/" +
        e.getAttribute(KEY_CATEGORY_THUMBNAIL);

        }
        else {
        correctURL=e.getAttribute(KEY_CATEGORY_THUMBNAIL);
        }
addCatData.createCategoriesEntry(Integer.parseInt(e.getAttribute(KEY_CATEGORY_ID)),
e.getAttribute(KEY_CATEGORY_NAME), e.getAttribute(KEY_CATEGORY_FULLIMAGE), correctURL,
Integer.parseInt(e.getAttribute(KEY_CATEGORY_PARENTID)));
}
addCatData.close();
        return null;
        }
}

        GetCategories catData = new GetCategories();
        catData.execute();
```

Στην περίπτωση της δικής μας εφαρμογής έχουμε χωρίσει τις διαδικασίες του parsing για τα προϊόντα και τις κατηγορίες σε δύο ξεχωριστά νήματα. Πιο κάτω φαίνεται το parsing των προϊόντων, κατά το οποίο γίνεται χρήση και της μεθόδου getValue.

```
        //Products Sync
        class GetProducts extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... params) {
        // TODO Auto-generated method stub

        addProductData.open();

        XMLParser parser = new XMLParser();
        String xml = parser.getXmlFromUrl(ProdURL); // getting XML from URL
        Document doc = parser.getDomElement(xml); // getting DOM element

        NodeList nl = doc.getElementsByTagName(KEY_PRODUCT);
        // looping through all product nodes
        for (int i = 0; i < nl.getLength(); i++) {
        Element e = (Element) nl.item(i);
        Element e2 = null;
        NodeList nl2 = e.getElementsByTagName(KEY_CATEGORY);
        for (int j = 0; j < nl2.getLength(); j++) {
                e2 = (Element) nl2.item(j);
```

```
		addProductData.createProdCatEntry(e.getAttribute(KEY_PRODUCT_WEBID),Integer.parseInt(e2.get
Attribute(KEY_PRODUCT_CATEGORYID)));
		}

		addProductData.createProductsEntry(e.getAttribute(KEY_PRODUCT_WEBID),e.getAttribute(KEY_PR
ODUCT_SKU), parser.getValue(e, KEY_PRODUCT_NAME), parser.getValue(e, KEY_PRODUCT_SHORTDESC),
parser.getValue(e, KEY_PRODUCT_DESC), Integer.parseInt(parser.getValue(e, KEY_PRODUCT_PRICE)),
parser.getValue(e, KEY_PRODUCT_THUMB), parser.getValue(e, KEY_PRODUCT_FULLIMAGE));

		}
		addProductData.close();
		return null;
			}
		}

		GetProducts prodData = new GetProducts();
		prodData.execute();

		dialog.dismiss();
```

Μόλις και δύο οι διαδικασίες ολοκληρωθούν κλείνουμε την βάση δεδομένων και τον διάλογο προόδου και αποθηκεύεται στη μεταβλητή της εφαρμογής lastUpdate μέσω της χρήσης των SharedPreferences η τιμή της ώρας και της ημερομηνίας που έγινε ο συγχρονισμός. Οι ρυθμίσεις της εφαρμογής είναι ουσιαστικά τιμές μεταβλητών που αποθηκεύονται ακόμα και μετά τον τερματισμό της εφαρμογής για μελλοντικής χρήσης. Μπορούμε να τις προσπελάσουμε και να τις τροποποιήσουμε με την χρήση του PreferenceManager. Στη συγκεκριμένη περίπτωση έχουμε την αποθήκευση ενός long που δείχνει το timestamp της στιγμής του update.

```
		long now = System.currentTimeMillis()/1000;
		SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
			SharedPreferences.Editor prefEditor = settings.edit();
			prefEditor.putLong("lastUpdate", now);
			prefEditor.commit();
```

Τέλος εμφανίζεται κατάλληλο μήνυμα προς τον χρήστη πως η διαδικασία ολοκληρώθηκε με επιτυχία και τα δεδομένα υπάρχουν πλέον αποθηκευμένα στη συσκευή του.

```
			AlertDialog alertDialog = new AlertDialog.Builder(FirstSync.this).create();
			alertDialog.setTitle("Database info");
			alertDialog.setMessage("Η βάση δεδομένων δημιουργήθηκε και ενημερώθηκε με επιτυχία");
			alertDialog.setButton("OK", new DialogInterface.OnClickListener() {
				public void onClick(DialogInterface dialog, int which) {
					finish();
				}
			});
				alertDialog.show();
		}
}
```

11.10 ImageAdapter.java

Στην κλάση ImageAdapter ορίζουμε τον τρόπο με τον οποίο ένα σύνολο δεδομένων που έχουμε από κάποια πηγή, συνήθως από την βάση δεδομένων μας, προσαρμόζονται πάνω σε έναν τύπο view. Πιο συγκεκριμένα εδώ αναπτύσσεται η διαδικασία ενσωμάτωσης ενός ArrayList σε ένα GridView. Αρχικά λοιπόν δηλώνουμε μία μεταβλητή τύπου Activity για να παίρνουμε πληροφορίες για το Activity που χρησιμοποιείται ο adapter, μια ArrayList που είναι η πηγή δεδομένων, έναν LayoutInflater του οποίου η χρήση έχει εξηγηθεί και σε προηγούμενο κεφάλαιο και μία μεταβλητή τύπου ImageLoader. Προς το παρόν να πούμε απλώς πως χρησιμοποιείται για την προβολή εικόνων από αυτές που είναι αποθηκευμένες στο τηλέφωνο σε οποιοδήποτε view μπορεί να προβάλει εικόνες.

```
public class ImageAdapter extends BaseAdapter {

    private Activity activity;
    private ArrayList<HashMap<String, String>> data;
    private ImageLoader imload;
    private static LayoutInflater inflater=null;

        public ImageAdapter (Activity a, ArrayList<HashMap<String, String>> d) {
                activity = a;
                data =d;
                imload = new ImageLoader(activity.getApplicationContext());
      inflater = (LayoutInflater)activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        }

        @Override
        public int getCount() {
                return data.size();
        }

        @Override
        public Object getItem(int position) {
                return position;
        }

        @Override
        public long getItemId(int position) {
                return position;
        }
```

Στο σημείο αυτό, για κάθε εγγραφή της βάσης δεδομένων δημιουργείται ένα καινούριο view και αντιστοιχίζονται οι ιδιότητες του με αυτές της βάσης. Έπειτα προστίθεται στο μητρικό view στο οποίο υπάγονται όλα τα views αυτά. Εδώ προστίθενται σε ένα GridView.

```
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
      View vi=convertView;
      if(convertView==null)
      vi = inflater.inflate(R.layout.categories, null);

     TextView catID = (TextView)vi.findViewById(R.id.tvCategoryID); // Category ID
     TextView catName = (TextView)vi.findViewById(R.id.tvCategoryName); // Category Name
     ImageView catThumb=(ImageView)vi.findViewById(R.id.ivCategoryThumb); // Category Thumb
```

```
HashMap<String, String> category = new HashMap<String, String>();
category = data.get(position);

catID.setText(category.get("catID"));
catName.setText(category.get("catName"));
imload.DisplayImage(category.get("catThumb"),catThumb);

return vi;    } }
```

11.11 ImageLoader.java

Όπως προαναφέρθηκε στο υποκεφάλαιο 11.10 η κλάση αυτή έχει να κάνει με την εμφάνιση εικόνων στα διάφορα views της εφαρμογής μας, όταν αυτές βρίσκονται αποθηκευμένες στην μνήμη της συσκευής. Λόγω του ότι είναι αρκετά εκτεταμένη και περιέχει μεθόδους που δεν είναι απαραίτητο να αναλυθούν στα πλαίσια της εργασίας αυτής, θα γίνει αναφορά στη σημαντικότερη μέθοδο της που χρησιμοποιούμε συχνά στην εφαρμογή μας. Η μέθοδος αυτή είναι η displayImage και φαίνεται παρακάτω

```
final int stub_id = R.drawable.no_image;
 public void DisplayImage(String url, ImageView imageView)
 {
    imageViews.put(imageView, url);
    Bitmap bitmap=memoryCache.get(url);
    if(bitmap!=null)
       imageView.setImageBitmap(bitmap);
    else
    {
       queuePhoto(url, imageView);
       imageView.setImageResource(stub_id);
    }

 }
```

Αυτό που κάνει είναι να ελέγχει αν μια εικόνα βρίσκεται στην τοπική μνήμη, να την αναζητεί και να την προβάλει στο Imageview που είναι το δεύτερο όρισμα της, ενώ σε διαφορετική περίπτωση εμφανίζει την εικόνα no_image που είναι η default όταν δεν βρεθεί αυτή που ψάχνουμε.

11.12 LazyAdapter.java

Η LazyAdapter είναι μια κλάση αντίστοιχη με την ImageAdapter, η διαφορά τους είναι πως αυτή προσαρμόζει τα δεδομένα της βάσης σε μια ListView αντί για ένα GridView. Παρατίθεται για να γίνουν εμφανείς οι διαφορές μεταξύ των δύο αυτών κλάσεων και γιατί χρησιμοποιείται σε πολλές κλάσεις της εφαρμογής.

```
public class LazyAdapter extends BaseAdapter {

    private Activity activity;
    private ArrayList<HashMap<String, String>> data;
     private static LayoutInflater inflater=null;
    public ImageLoader imageLoader;


    public LazyAdapter(Activity a, ArrayList<HashMap<String, String>> d) {
    activity = a;
    data=d;
    inflater = (LayoutInflater)activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    imageLoader=new ImageLoader(activity.getApplicationContext());
     }

      public int getCount() {
       return data.size();
      }

     public Object getItem(int position) {
     return position;
    }

    public long getItemId(int position) {
    return position;
     }

    public View getView(int position, View convertView, ViewGroup parent) {
    View vi=convertView;
     if(convertView==null)
     vi = inflater.inflate(R.layout.list_row, null);

    TextView prodWebID= (TextView)vi.findViewById(R.id.tvListProdWebID); //Product WebID
    TextView prodSKU = (TextView)vi.findViewById(R.id.tvListProdSKU); //Product SKU
    TextView prodName = (TextView)vi.findViewById(R.id.tvListProdName); // Product Name
    TextView prodSDesc = (TextView)vi.findViewById(R.id.tvListProdSDesc); // Prod ShortDescription
    TextView prodDesc = (TextView)vi.findViewById(R.id.tvListProdDesc); //Product Description
    TextView prodPrice = (TextView)vi.findViewById(R.id.tvListProdPrice); // Product Price
    TextView prodImage=(TextView)vi.findViewById(R.id.tvListProdImage); // Product Image URL
    mageView prodThumbIV=(ImageView)vi.findViewById(R.id.ivListProdThumb); // Product Thumb

  HashMap<String, String> product = new HashMap<String, String>();
  product = data.get(position);

  // Setting all values in listview
  prodWebID.setText(product.get("prodWebID"));
  prodSKU.setText(product.get("prodSKU"));
  prodName.setText(product.get("prodName"));
  prodSDesc.setText(product.get("prodSDesc"));
  prodDesc.setText(product.get("prodDesc"));
  prodPrice.setText(product.get("prodPrice") + "€");
  prodImage.setText(product.get("prodImage"));
  imageLoader.DisplayImage(product.get("prodThumb"), prodThumbIV);
  return vi;
 }
}
```

11.13 Maps.java

Συνεχίζουμε με την κλάση Maps. Αυτή είναι υπεύθυνη για την προβολή του χάρτη όταν ο χρήστης επιλέξει «Καταστήματα» από το αναδυόμενο μενού στην κεντρική οθόνη. Παρατηρούμε ότι η κλάση αυτή δεν επεκτείνει την Activity αλλά την MapActivity. Δηλώνουμε λοιπόν μία μεταβλητή τύπου MapView και έναν MapController για να διαχειριζόμαστε το MapView. Να σημειώσουμε βέβαια πως για να έχουμε πρόσβαση στο Google Maps μέσω της εφαρμογής μας, πρέπει πρώτα να ακολουθήσουμε την διαδικασία που περιγράφεται στο κεφάλαιο 8.

```
public class Maps extends MapActivity {

    private MapView mapView;
    private MapController mc;

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

Στο σημείο αυτό ενεργοποιούμε τα κουμπιά για το zoom in και zoom out στον χάρτη, ενώ λίγο πιο κάτω, θέτουμε τις συντεταγμένες τις οποίες θέλουμε να εμφανίζονται καθώς και το επίπεδο του zoom κατά την εκκίνηση του MapActivity. Έτσι στην δική μας περίπτωση μόλις εμφανίζονται οι χάρτες είναι τοποθετημένοι στο κέντρο της Θεσσαλονίκης

```
    //Zooming
    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);

    mc = mapView.getController();

    double lat = Double.parseDouble("40.6334646");
    double lon = Double.parseDouble("22.941757");
    GeoPoint geoPoint = new GeoPoint((int)(lat * 1E6), (int)(lon * 1E6));
    mc.animateTo(geoPoint);
    mc.setZoom(15);
    mapView.invalidate();
```

Παρακάτω φαίνεται πως δημιουργούμε ένα OverLay για τον χάρτη μας και μετά πως προσθέτουμε ορισμένα σημεία στον χάρτη μας. Έχουμε διαλέξει μια εικόνα ενός marker και το θέτουμε ως drawable ως εικόνα δηλαδή στο σημείο ενδιαφέροντος που δημιουργήσαμε. Επίσης βάζουμε και έναν τίτλο και ένα snippet (μικρό κείμενο) στο σημείο αυτό. Αφότου προσδώσουμε όλες αυτές τις ιδιότητες, προσθέτουμε το σημείο στο Overlay.

```
    //Add marker
    List<Overlay> mapOverlays = mapView.getOverlays();
    Drawable drawable = this.getResources().getDrawable(R.drawable.marker);
    AddItemizedOverlay itemizedOverlay = new AddItemizedOverlay(drawable, this);
      OverlayItem overlayitem = new OverlayItem(geoPoint, "Κατάστημα 1", "Το κατάστημα στο κέντρο
      - Τσιμισκή 30");
```

```
 itemizedOverlay.addOverlay(overlayitem);
 mapOverlays.add(itemizedOverlay);
 mapView.invalidate();
}

@Override
  protected boolean isRouteDisplayed() {
return false;
}
}
```

11.14 MemoryCache.java

Επόμενη κλάση είναι η MemoryCache, σκοπός της οποίας είναι η προσωρινή αποθήκευση των αρχείων της εφαρμογής που υπάρχουν στην μνήμη της συσκευής, έτσι ώστε να είναι γρηγορότερη η προσπέλαση τους και ως επακόλουθο να έχουμε μια ταχύτερη εφαρμογή , ιδιαίτερα όταν υπάρχουν Activities με πολλαπλές εικόνες. Περιέχει μεθόδους προσπέλασης όπως η get και προσθήκης δεδομένων στην cache όπως η put καθώς και καθαρισμού αυτής με την μέθοδο clear.

```
public class MemoryCache {
 private Map<String, SoftReference<Bitmap>> cache=Collections.synchronizedMap(new
HashMap<String, SoftReference<Bitmap>>());

  public Bitmap get(String id){
     if(!cache.containsKey(id))
         return null;
       SoftReference<Bitmap> ref=cache.get(id);
       return ref.get();
   }

   public void put(String id, Bitmap bitmap){
      cache.put(id, new SoftReference<Bitmap>(bitmap));
   }

 public void clear() {
    cache.clear();
 }
}
```

11.15 NewProducts.java

Η κλάση αυτή περιέχει όλες εκείνες τις ενέργειες που χρειάζονται για να προβληθούν τα προϊόντα που ανήκουν στην κατηγορία Νέες Παραλαβές. Αυτό επιτυγχάνεται με ένα ερώτημα στην βάση δεδομένων θέτοντας του ως όρισμα το id 158 που είναι το id της εν λόγω κατηγορίας. Έπειτα παίρνουμε τα δεδομένα και μέσω ενός LazyAdapter τα προσθέτουμε σε μια ListView, με τους ίδιους τρόπους που αναλύθηκαν και νωρίτερα.

```
public class NewProducts extends Activity {
private ListView list;
private LazyAdapter adapter;

@Override
protected void onCreate(Bundle savedInstanceState) {

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

Intent in = getIntent();
String order = "";

if (in.getExtras()!=null)
order = in.getStringExtra("order");

ProgressDialog dialog = ProgressDialog.show(NewProducts.this, "", "Φόρτωση προϊόντων...", true);

dialog.show();

SQLConnect addSQLtoList = new SQLConnect(this);

addSQLtoList.open();
ArrayList<HashMap<String, String>> listData = addSQLtoList.catProductsToList("158", order);
addSQLtoList.close();

list = (ListView) findViewById(R.id.list);
adapter = new LazyAdapter(this, listData);

list.setEmptyView(findViewById(R.id.emptylist));
list.setAdapter(adapter);
list.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String pWebID = ((TextView) view.findViewById(R.id.tvListProdWebID)).getText().toString();
String pSKU = ((TextView) view.findViewById(R.id.tvListProdSKU)).getText().toString();
String pName = ((TextView) view.findViewById(R.id.tvListProdName)).getText().toString();
String pDesc = ((TextView) view.findViewById(R.id.tvListProdDesc)).getText().toString();
String pPrice = ((TextView) view.findViewById(R.id.tvListProdPrice)).getText().toString();
String pImage = ((TextView) view.findViewById(R.id.tvListProdImage)).getText().toString();

Intent in = new Intent(getApplicationContext(), ViewProduct.class);

in.putExtra("prodWebID", pWebID);
in.putExtra("prodSKU", pSKU);
in.putExtra("prodName", pName);
in.putExtra("prodDesc", pDesc);
in.putExtra("prodPrice", pPrice);
in.putExtra("prodImage", pImage);

startActivity(in);
        }
});

dialog.dismiss();
```

```
    }
```

Και εδώ έχουμε υλοποίηση της δυνατότητας ταξινόμησης όλων των στοιχείων της λίστας που προσφέρεται στον χρήστη. Ο τρόπος είναι ίδιος με αυτόν της κλάσης Favorites.

```
    @Override
    public boolean onCreateOptionsMenu(android.view.Menu menu) {

        super.onCreateOptionsMenu(menu);
        MenuInflater minf = getMenuInflater();
        minf.inflate(R.menu.sortmenu, menu);
        return true;
    }

    @Override
    public boolean onMenuItemSelected(int id, MenuItem item) {
        switch (item.getItemId()) {

        case R.id.sortbynamea:
            Intent in = new Intent(getApplicationContext(), NewProducts.class);
            in.putExtra("order", "ORDER BY p.name ASC");
            finish();
            startActivity(in);
            break;

        case R.id.sortbypricea:
            Intent in2 = new Intent(getApplicationContext(), NewProducts.class);
            in2.putExtra("order", "ORDER BY p.price_with_VAT ASC");
            finish();
            startActivity(in2);
            break;

        case R.id.sortbynamed:
            Intent in3 = new Intent(getApplicationContext(), NewProducts.class);
            in3.putExtra("order", "ORDER BY p.name DESC");
            finish();
            startActivity(in3);
            break;

        case R.id.sortbypriced:
            Intent in4 = new Intent(getApplicationContext(), NewProducts.class);
            in4.putExtra("order", "ORDER BY p.price_with_VAT DESC");
            finish();
            startActivity(in4);
            break;
        }
        return false;
    }
}
```

11.16 News.java

Η κλάση αυτή αναλαμβάνει κατά την εκτέλεση της να συνδέσει μια μεταβλητή τύπου Webview με ένα webview που υπάρχει στα resources της εφαρμογής και μετά μέσω της μεθόδου loadUrl που υπάρχει στα webviews να φορτώσει ένα επιθυμητό url, στο οποίο θα υπάρχουν είτε σε μορφή ιστοσελίδας ή κάποιου εγγράφου (π.χ. pdf) τα νέα και οι προσφορές του καταστήματος. Στα πλαίσια της εφαρμογής δεν έχει γίνει κάτι τέτοιο διότι ανοίγουμε ένα κανονικό url, απλά υποδεικνύεται ο τρόπος που μπορεί να γίνει η διαδικασία αυτή

```
public class News extends Activity {

private WebView wv;

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

        wv = (WebView) findViewById(R.id.wvNews);
        wv.loadUrl("http://www.caretta-net.gr/demos/e-shop/");
}

}
```

11.17 OnShop.java

Στο αρχείο OnShop.java υπάρχουν αν όχι τα πιο σημαντικά, πολύ σημαντικά κομμάτια κώδικα της εφαρμογής καθώς εδώ γίνονται όλες οι διαδικασίες για την εμφάνιση του κεντρικού μενού και για τις ενέργειες που γίνονται κάθε φορά που κάνει μια επιλογή ο χρήστης. Ας τις δούμε αναλυτικά.

Αρχικά δηλώνουμε τις μεταβλητές που αντιστοιχούν στα κουμπιά του μενού, μία μεταβλητή ελέγχου της κατάστασης της σύνδεσης δικτύου και δημιουργούμε και ένα αντικείμενο PreferenceManager για να έχουμε πρόσβαση στις ρυθμίσεις της εφαρμογής. Γίνεται έλεγχος στην boolean μεταβλητή firstrun και αν είναι αληθής εμφανίζεται κατάλληλο μήνυμα στον χρήστη πως πρέπει να γίνει συγχρονισμός των προϊόντων.

```
public class OnShop extends Activity implements OnClickListener{ //Kentriko menu me upomenu
epilogon

private Button products, new_products, sales, favorites, contact, news, settings, cart, search;
private boolean netOk;

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

```
SharedPreferences getPreferences =
PreferenceManager.getDefaultSharedPreferences(getBaseContext()); //Dimiourgoume mia metabliti
gia na diaxeiristoume ta preferences
boolean firstLaunch = getPreferences.getBoolean("firstRun", true);

if (firstLaunch) {

AlertDialog.Builder alertDialog = new AlertDialog.Builder(OnShop.this);
 alertDialog.setTitle("Απαιτείται σύνδεση στο Internet");
alertDialog.setMessage("Είναι η πρώτη φορά που χρησιμοποιείτε την εφαρμογή, θα πρέπει να
υπάρχει μια ενεργή σύνδεση δεδομένων για να γίνει συγχρονισμός των προϊόντων");
 alertDialog.setPositiveButton("OK", new DialogInterface.OnClickListener() {
```

Αν ο χρήστης επιλέξει το OK στον διάλογο γίνεται έλεγχος για την κατάσταση της σύνδεσης δικτύου και αν υπάρχει τη στιγμή αυτή ξεκινάμε το Activity FirstSync για να γίνει ο συγχρονισμός και θέτουμε την τιμή της μεταβλητής firstRun σε false, αλλιώς εμφανίζουμε μήνυμα στον χρήστη και η εφαρμογή τερματίζεται. Το ίδιο συμβαίνει και στην επιλογή άκυρο.

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

 netOk = isOnline();

 if (netOk) {
 Intent openFirstSynActivity = new Intent(OnShop.this, FirstSync.class);
 OnShop.this.startActivity(openFirstSynActivity);

 SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
 SharedPreferences.Editor prefEditor = settings.edit();
 prefEditor.putBoolean("firstRun", false);
 prefEditor.commit();

  } else {
  AlertDialog alertDialog = new AlertDialog.Builder(OnShop.this).create();
 alertDialog.setTitle("Network info");
alertDialog.setMessage("Δεν υπάρχει σύνδεση δεδομένων, δεν μπορεί να γίνει ανανέωση των
προϊόντων");
 alertDialog.setButton("OK", new DialogInterface.OnClickListener() {
 public void onClick(DialogInterface dialog, int which) {
          finish();
          }
 });
 alertDialog.show();
          }
    }
 });

 alertDialog.setNegativeButton("Άκυρο", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
          finish();
    }
 });

alertDialog.show();
}
```

Παρακάτω έχουμε την σύνδεση όλων των κουμπιών που δηλώσαμε στην αρχή με τα resources της εφαρμογής καθώς και την τοποθέτηση listeners για το γεγονός OnClick σε κάθε ένα από αυτά.

```
products = (Button) findViewById(R.id.btn_products);
products.setOnClickListener(this);

new_products = (Button) findViewById(R.id.btn_nproducts);
new_products.setOnClickListener(this);

sales = (Button) findViewById(R.id.btn_sales);
sales.setOnClickListener(this);

favorites = (Button) findViewById(R.id.btn_favorites);
favorites.setOnClickListener(this);

contact = (Button) findViewById(R.id.btn_contact);
contact.setOnClickListener(this);

news = (Button) findViewById(R.id.btn_news);
news.setOnClickListener(this);

settings = (Button) findViewById(R.id.btn_settings);
settings.setOnClickListener(this);


cart = (Button) findViewById(R.id.btn_cart);
cart.setOnClickListener(this);

search = (Button) findViewById(R.id.btn_search);
search.setOnClickListener(this);

}
```

Εδώ περιγράφονται οι διαδικασίες που θα γίνονται στην περίπτωση του κλικ σε κάποιο κουμπί, παρατηρούμε πως σε όλα δημιουργείται ένα νέο intent το οποίο εκκινεί ένα άλλο Activity.

```
public void onClick(View v) {
        switch (v.getId()) {

        case R.id.btn_products:
                Intent openProductsActivity = new Intent(OnShop.this,Categories.class);
                OnShop.this.startActivity(openProductsActivity);
                break;

        case R.id.btn_nproducts:
                Intent openNewProductsActivity = new Intent(OnShop.this, NewProducts.class);
                OnShop.this.startActivity(openNewProductsActivity);
                break;

        case R.id.btn_sales:
                Intent openSalesActivity = new Intent(OnShop.this,Sales.class);
                OnShop.this.startActivity(openSalesActivity);
                break;

        case R.id.btn_favorites:
```

```
                Intent openFavoritesActivity = new Intent(OnShop.this, Favorites.class);
                OnShop.this.startActivity(openFavoritesActivity);
                break;

        case R.id.btn_contact:
                Intent openContactActivity = new Intent(OnShop.this, Contact.class);
                OnShop.this.startActivity(openContactActivity);
                break;

        case R.id.btn_news:
                Intent openNewsActivity = new Intent(OnShop.this, News.class);
                OnShop.this.startActivity(openNewsActivity);
                break;

        case R.id.btn_settings:
                Intent openSettingsActivity = new Intent(OnShop.this, Settings.class);
                OnShop.this.startActivity(openSettingsActivity);
                break;

        case R.id.btn_cart:
                Intent openCartActivity = new Intent(OnShop.this, Cart.class);
                OnShop.this.startActivity(openCartActivity);
                break;

        case R.id.btn_search:
                Intent openSearchActivity = new Intent(OnShop.this,   Search.class);
                OnShop.this.startActivity(openSearchActivity);
                break;

        }
}
```

Παρακάτω υπάρχει η μέθοδος δήλωσης του αναδυόμενου μενού καθώς και οι μέθοδοι που περιγράφουν τι θα γίνεται μόλις ο χρήστης επιλέξει κάποιο στοιχείο του. Ειδικά για την περίπτωση του συγχρονισμού, για να ενημερώσουμε τον χρήστη για το πότε έγινε ο τελευταίος συγχρονισμός, παίρνουμε από τις ρυθμίσεις της εφαρμογής την τιμή της μεταβλητής lastUpdate που περιέχει ένα timestamp, το μετατρέπουμε σε κανονική ημερομηνία και το εμφανίζουμε μαζί με το μήνυμα που εμφανίζουμε στον χρήστη πριν το συγχρονισμό.

```
@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {

        super.onCreateOptionsMenu(menu);
        MenuInflater minf = getMenuInflater();
        minf.inflate(R.menu.simple_menu, menu);
        return true;
}

@Override
public boolean onMenuItemSelected(int id, MenuItem item) {
        switch (item.getItemId()) {
        case R.id.about:
                Intent openAboutActivity = new Intent(OnShop.this, About.class);
                startActivity(openAboutActivity);
                break;
```

```
            case R.id.update:

            SharedPreferences getPreferences =
            PreferenceManager.getDefaultSharedPreferences(getBaseContext());
            final long lastUpdate = getPreferences.getLong("lastUpdate", 0);

            Calendar mydate = Calendar.getInstance();
            mydate.setTimeInMillis(lastUpdate*1000);
            int month = mydate.get(Calendar.MONTH);
            month++;

StringBuffer timestamp = new StringBuffer();
if (mydate.get(Calendar.HOUR_OF_DAY) >= 10)
   timestamp.append(mydate.get(Calendar.HOUR_OF_DAY));
else
   timestamp.append("0" + mydate.get(Calendar.HOUR_OF_DAY));

timestamp.append(":");

if (mydate.get(Calendar.MINUTE) >= 10)
   timestamp.append(mydate.get(Calendar.MINUTE));
else
   timestamp.append("0" + mydate.get(Calendar.MINUTE));

            String lastUpdateOn = mydate.get(Calendar.DAY_OF_MONTH)+"/"+ month
            +"/"+mydate.get(Calendar.YEAR)+" - "+timestamp+"";

AlertDialog.Builder alertDialog = new AlertDialog.Builder(OnShop.this);
       alertDialog.setTitle("Απαιτείται σύνδεση στο Internet");
          alertDialog.setMessage("Θα πρέπει να υπάρχει μια ενεργή σύνδεση δεδομένων για να γίνει
          συγχρονισμός των προϊόντων (Τελευταία ενημέρωση: "+lastUpdateOn+")");
       alertDialog.setPositiveButton("OK", new DialogInterface.OnClickListener() {

          public void onClick(DialogInterface dialog, int which) {
                              Intent openSyncActivity = new Intent(OnShop.this, Sync.class);
                              startActivity(openSyncActivity);
          }
      });
      alertDialog.setNegativeButton("Άκυρο", new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                    }
          });

       alertDialog.show();

                    break;

          case R.id.exit:
                    finish();
                    break;

          case R.id.maps:
                    Intent openMapsActivity = new Intent(OnShop.this, Maps.class);
                    startActivity(openMapsActivity);
                    break;
          }
```

```
            return false;
        }
```

Τέλος, μια απλή μέθοδος που αναλαμβάνει να κάνει έλεγχο για το αν υπάρχει διαθέσιμη σύνδεση δεδομένων στη συσκευή τη στιγμή αυτή. Αυτό επιτυγχάνεται με την χρήση της μεθόδου getActiveNetworkInfo();

```
        public boolean isOnline() {
            ConnectivityManager cm = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
            NetworkInfo netInfo = cm.getActiveNetworkInfo();
            if (netInfo != null && netInfo.isConnected()) {
                return true;
            }
            return false;

        }

}
```

11.18 Products.java

Η κλάση αυτή έχει πολλές ομοιότητες με προηγούμενες κλάσεις αλλά η σημαντική της διαφορά είναι πως συνδυάζει τις διαδικασίες που εκτελούνται σε δύο άλλες κλάσεις. Πιο συγκεκριμένα σε αυτή την κλάση έχουμε συνδυασμό της προβολής των κατηγοριών και της προβολής των προϊόντων και αυτό γιατί με τη χρήση της θέλουμε να προβάλουμε τα προϊόντα αλλά και τις υποκατηγορίες που υπάρχουν σε μια κατηγορία. Έτσι λοιπόν έχουμε σε μία κλάση κώδικα παρόμοιο με αυτόν της κλάσης Categories και της NewProducts.

Στο πρώτο μέρος της γίνεται η διαδικασία που έχει περιγραφεί και πιο πριν, για την εμφάνιση των κατηγοριών με ένα συγκεκριμένο id μητρικής κατηγορίας, μόνο που αυτή τη φορά τα διάφορα views αντί να προστίθενται σε ένα GridView, προστίθενται σε ένα οριζόντιο ScrollView με όνομα catHolder.

```
public class Products extends Activity {

    private ListView list;
    private LazyAdapter adapter;
    private static LayoutInflater inflater=null;
    private ImageLoader imload;
    private String cID;;
    private View vi;
    @Override
    protected void onCreate(Bundle savedInstanceState) {

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

    Intent in = getIntent();
```

```
String order = "";

if (in.getStringExtra("order")!=null)
order = in.getStringExtra("order");

cID = in.getStringExtra("catID");

ProgressDialog dialog = ProgressDialog.show(Products.this, "", "Φόρτωση προϊόντων...", true);

dialog.show();

SQLConnect addSQLtoList = new SQLConnect(this);

addSQLtoList.open();
ArrayList<HashMap<String, String>> subCategoriesData = addSQLtoList.categoriesToGrid(cID);
addSQLtoList.close();

LinearLayout catHolder = (LinearLayout)findViewById(R.id.categoriesHolder);
inflater = (LayoutInflater)this.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
imload = new ImageLoader(this.getApplicationContext());

if (subCategoriesData.size()==0) {

        TextView t = new TextView(this);
        t.setText("Δεν υπάρχουν υποκατηγορίες στην κατηγορία αυτή");
        t.setTextColor(Color.WHITE);
        t.setTextSize(13);
        t.setPadding(5,10,10,10);
        catHolder.addView(t);
}

else {

        for (int i=0; i<subCategoriesData.size(); i++) {

        HashMap<String, String> category = new HashMap<String, String>();
        category = subCategoriesData.get(i);

        vi = inflater.inflate(R.layout.categories,null);

    TextView subCatID = (TextView)vi.findViewById(R.id.tvCategoryID); //SubCategory ID
    TextView subCatName = (TextView)vi.findViewById(R.id.tvCategoryName);
    ImageView subCatThumb = (ImageView)vi.findViewById(R.id.ivCategoryThumb);
    subCatID.setText(category.get("catID"));
    subCatName.setText(category.get("catName"));
    imload.DisplayImage(category.get("catThumb"),subCatThumb);

    vi.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View catView) {
Intent in = new Intent(getApplicationContext(), Products.class);
TextView subCatID2 = (TextView)catView.findViewById(R.id.tvCategoryID);
in.putExtra("catID", subCatID2.getText());
        startActivity(in);
        }
});
```

```
catHolder.addView(vi);
        }
}
```

Από το σημείο αυτό, ξεκινά η δεύτερη διαδικασία, δηλαδή το γέμισμα της λίστας των προϊόντων. Η διαδικασία έχει περιγραφεί και σε προηγούμενο κεφάλαιο. Και σε αυτή την κλάση υπάρχουν τα κομμάτια του κώδικα που αναλαμβάνουν την ταξινόμηση των στοιχείων της λίστας.

```
addSQLtoList.open();
ArrayList<HashMap<String, String>> listData = addSQLtoList.catProductsToList(cID,order);
addSQLtoList.close();

list = (ListView) findViewById(R.id.list2);
adapter = new LazyAdapter(this, listData);
list.setEmptyView(findViewById(R.id.emptylist2));
list.setAdapter(adapter);

list.setOnItemClickListener(new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

String pWebID = ((TextView) view.findViewById(R.id.tvListProdWebID)).getText().toString();
String pSKU = ((TextView) view.findViewById(R.id.tvListProdSKU)).getText().toString();
String pName = ((TextView) view.findViewById(R.id.tvListProdName)).getText().toString();
String pSDesc = ((TextView) view.findViewById(R.id.tvListProdSDesc)).getText().toString();
String pDesc = ((TextView) view.findViewById(R.id.tvListProdDesc)).getText().toString();
String pPrice = ((TextView) view.findViewById(R.id.tvListProdPrice)).getText().toString();
String pImage = ((TextView) view.findViewById(R.id.tvListProdImage)).getText().toString();

Intent in = new Intent(getApplicationContext(), ViewProduct.class);
in.putExtra("prodWebID", pWebID);
in.putExtra("prodSKU", pSKU);
in.putExtra("prodName", pName);
in.putExtra("prodSDesc", pSDesc);
in.putExtra("prodDesc", pDesc);
in.putExtra("prodPrice", pPrice);
in.putExtra("prodImage", pImage);

        startActivity(in);
        }
});

dialog.dismiss();

}

@Override
public boolean onCreateOptionsMenu(android.view.Menu menu) {

        super.onCreateOptionsMenu(menu);
        MenuInflater minf = getMenuInflater();
        minf.inflate(R.menu.sortmenu, menu);
        return true;
}
```

```
	@Override
	public boolean onMenuItemSelected(int id, MenuItem item) {
		switch (item.getItemId()) {

		case R.id.sortbynamea:
			Intent in = new Intent(getApplicationContext(), Products.class);
			in.putExtra("order", "ORDER BY p.name ASC");
			in.putExtra("catID", cID);
			finish();
			startActivity(in);
			break;

		case R.id.sortbypricea:
			Intent in2 = new Intent(getApplicationContext(), Products.class);
			in2.putExtra("order", "ORDER BY p.price_with_VAT ASC");
			in2.putExtra("catID", cID);
			finish();
			startActivity(in2);
			break;

		case R.id.sortbynamed:
			Intent in3 = new Intent(getApplicationContext(), Products.class);
			in3.putExtra("order", "ORDER BY p.name DESC");
			in3.putExtra("catID", cID);
			finish();
			startActivity(in3);
			break;

		case R.id.sortbypriced:
			Intent in4 = new Intent(getApplicationContext(), Products.class);
			in4.putExtra("order", "ORDER BY p.price_with_VAT DESC");
			in4.putExtra("catID", cID);
			finish();
			startActivity(in4);
			break;
		}
		return false;
	}


}
```

11.19 Sales.java

Ο κώδικας της κλάσης αυτής είναι πανομοιότυπος με τον κώδικα της κλάσης NewProducts για αυτό και δεν θα αναφερθούμε σε αυτή. Η μοναδική διαφορά είναι πως αλλάζει το όρισμα στην μέθοδο προσπέλασης της βάσης δεδομένων, ώστε να μας επιστρέφει τα προϊόντα που ανήκουν στην κατηγορία Προσφορές.

11.20 Search.java

Η κλάση αυτή περιέχει όλες τις διαδικασίες για να εκτελείται η αναζήτηση προϊόντων και κατηγοριών. Όπως σε όλες τις κλάσεις, αρχικά κάνουμε την συσχέτιση των μεταβλητών με τα resources της εφαρμογής ενώ μετά από αυτό εκτελούμε την μέθοδο getAllProducts() που είναι ένα ερώτημα στην βάση δεδομένων που επιστρέφει όλα τα προϊόντα, έτσι ώστε αυτό το σύνολο δεδομένων να το ορίσουμε ως προσαρμογέα του AutoCompleteTextView και να μας δίνει τις προτάσεις που πρέπει όσο πληκτρολογούμε σε αυτό.

```
public class Search extends Activity {

    private String[] items;

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

final AutoCompleteTextView searchTerms = (AutoCompleteTextView) findViewById(R.id.acSearch);
final Button searchButton = (Button)findViewById(R.id.btnDoSearch);
final RadioButton rbProducts = (RadioButton)findViewById(R.id.rbProducts);
final RadioButton rbCategories = (RadioButton)findViewById(R.id.rbCategories);

final SQLConnect sqlData = new SQLConnect(this);

sqlData.open();
items = sqlData.getAllProducts();
sqlData.close();

ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, R.layout.search_items, items);
searchTerms.setAdapter(adapter);
```

Η επιλογή για το αν θα αναζητήσουμε προϊόντα ή κατηγορίες γίνεται με δύο RadioButtons. Παρακάτω φαίνονται οι διαδικασίες που γίνονται εάν ο χρήστης κάνει κλικ σε ένα από αυτά. Ουσιαστικά αλλάζουμε τα δεδομένα που περιέχει ο προσαρμογέας του AutoCompleteTextView ανά περίπτωση, εκτελώντας ένα νέο ερώτημα στην βάση.

```
rbProducts.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
    sqlData.open();
    items = sqlData.getAllProducts();
    sqlData.close();

    ArrayAdapter<String> adapter = new ArrayAdapter<String>(Search.this, R.layout.search_items,
    items);
    searchTerms.setAdapter(adapter);
            }
    });
```

```
rbCategories.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v) {
sqlData.open();
items = sqlData.getAllCategories();
sqlData.close();

ArrayAdapter<String> adapter = new ArrayAdapter<String>(Search.this, R.layout.search_items,
items);
searchTerms.setAdapter(adapter);
} });
searchButton.setOnClickListener(new View.OnClickListener() {
```

Τέλος εδώ περιγράφονται οι ενέργειες που γίνονται στην περίπτωση που πατήσουμε το κουμπί αναζήτηση, γίνεται δηλαδή έλεγχος αν αναζητούμε κατηγορίες ή προϊόντα και ανάλογα ξεκινάμε το κατάλληλο Activity, δίνοντας στο intent ως επιπλέον δεδομένα για μεταφορά στο Activity που πρόκειται να ξεκινήσει, αυτά που έχει πληκτρολογήσει ο χρήστης στο TextView.

```
@Override
public void onClick(View v) {

if (rbProducts.isChecked() && searchTerms.getText().toString()!="") {
sqlData.open();
ArrayList<HashMap<String, String>> productsData =
sqlData.searchProducts(searchTerms.getText().toString());
sqlData.close();
Intent in = new Intent(getApplicationContext(), SearchProducts.class);
in.putExtra("pData", productsData);
startActivity(in);
}

else if (rbCategories.isChecked() && searchTerms.getText().toString()!="") {
sqlData.open();
ArrayList<HashMap<String, String>> categoriesData =
sqlData.searchCategories(searchTerms.getText().toString());
sqlData.close();
Intent in = new Intent(getApplicationContext(), SearchCategories.class);
in.putExtra("cData", categoriesData);
startActivity(in);
        }}
});
}

}
```

11.21 SearchCategories.java και SearchProducts.java

Οι δύο αυτές κλάσεις είναι αυτές που περιέχουν τα δεδομένα που εκτελούνται όταν ο χρήστης κάνει πλέον την αναζήτηση που θέλει χρησιμοποιώντας όλα όσα αναλύσαμε στο προηγούμενο κεφάλαιο. Εδώ βλέπουμε μόνο την κλάση SearchCategories καθώς η μοναδική της

διαφορά με την SearchProducts είναι πως η τελευταία τοποθετεί τα αποτελέσματα της αναζήτησης σε λίστα αντί για πλέγμα όπως κάνει η αναζήτηση κατηγοριών. Επειδή όμως η διαφοροποίηση της τοποθέτησης των δεδομένων με τους δύο αυτούς τρόπους έχει ήδη αναλυθεί στο κεφάλαιο 11.18 εδώ θα αναλυθεί μόνο η πρώτη κλάση.

Δηλώνεται λοιπόν το GridView που θα δεχτεί τα αποτελέσματα καθώς και ο προσαρμογέας του και έπειτα εκτελείται ένα ερώτημα στην βάση με όρισμα τα δεδομένα που έχουμε πάρει από το Activity Sales. Αυτό γίνεται στη γραμμή

ArrayList<HashMap<String, String>> categoriesData =(ArrayList<HashMap<String, String>>)getIntent().getSerializableExtra("cData");

Μετά από αυτό, εμφανίζουμε τα δεδομένα η σε περίπτωση που δεν υπάρχουν καταχωρήσεις στην βάση με τα κριτήρια αυτά, εμφανίζεται κατάλληλο μήνυμα.

```
public class SearchCategories extends Activity {

    private GridView categories;
    private ImageAdapter adapter;

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

    ProgressDialog dialog = ProgressDialog.show(SearchCategories.this,"", "Φόρτωση κατηγοριών...",
    true);

    @SuppressWarnings("unchecked")
    ArrayList<HashMap<String, String>> categoriesData =(ArrayList<HashMap<String,
    String>>)getIntent().getSerializableExtra("cData");

    categories = (GridView) findViewById(R.id.gvCategories);
    adapter = new ImageAdapter(SearchCategories.this, categoriesData);

    TextView tv = (TextView)findViewById(R.id.emptygrid);
    tv.setText("Δεν υπάρχουν κατηγορίες με τα κριτήρια αυτά");
    categories.setEmptyView(tv);

    categories.setAdapter(adapter);

    dialog.dismiss();

    categories.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

    String cID = ((TextView) view.findViewById(R.id.tvCategoryID)).getText().toString();

    Intent in = new Intent(getApplicationContext(), Products.class);

    in.putExtra("catID", cID);

    startActivity(in);
    } }); }
```

11.22 Settings.java

Η κλάση αυτή, αναλαμβάνει να δώσει στον χρήστη την δυνατότητα να κάνει είσοδο στο κατάστημα με τα στοιχεία του έτσι ώστε να δει και αν επιθυμεί να επεξεργαστεί κάποια από αυτά. Όλη η διαδικασία γίνεται μέσω ενός web service, δίνοντας του ως παραμέτρους στο url τα στοιχεία εισόδου που πληκτρολόγησε ο χρήστης. Έτσι, δεν υπάρχει κάτι σημαντικό που θα πρέπει να αναλυθεί από την πλευρά του κώδικα.

11.23 Splash.java

Πρόκειται για το πρώτο Activity που καλείται μόλις εκτελεστεί η εφαρμογή και εμφανίζει το splash screen. Το πιο σημαντικό της στοιχείο είναι η δήλωση ενός Thread το οποίο θέτουμε σε κατάσταση «ύπνου» για 2 δευτερόλεπτα έτσι ώστε το συγκεκριμένο Activity να είναι ενεργό μόνο για αυτό το διάστημα και μετά να ξεκινήσει το επόμενο που είναι το κεντρικό μενού. Δηλώνεται λοιπόν ένα νέο Thread και το θέτουμε σε αναμονή με στην γραμμή sleep(2000). Στη συνέχεια εκτελούμε αυτό το Thread με την μέθοδο start()

Παρατηρούμε εδώ πως υπάρχει υπέρβαση της μεθόδου onPause() η οποία εκτελείται μόλις ένα Activity περάσει στο παρασκήνιο και μέσα σε αυτή καλούμε την μέθοδο finish() που τερματίζει το Activity, κάτι το οποίο είναι επιθυμητό στην περίπτωση αυτή, καθώς δεν θα χρειαστεί να εμφανίσουμε ξανά το splash screen.

```
public class Splash extends Activity {

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// TODO Auto-generated method stub
	super.onCreate(savedInstanceState);
	setContentView(R.layout.splash);
	Thread timer = new Thread(){
	public void run(){
	try {
		sleep(2000);
	}catch (InterruptedException e){
		e.printStackTrace();
	}finally{
		Intent openOnShopActivity = new Intent("com.onshop.anzip.ONSHOPACTIVITY");
		startActivity(openOnShopActivity);
			}
		}
	}; timer.start();
	}

	@Override
	protected void onPause() {
	super.onPause();
```

```
            finish();
        } }
```

11.24 SQLConnect.java

Η κλάση αυτή, περιέχει πολύ σημαντικές λειτουργίες της εφαρμογής, καθώς ένα πολύ μεγάλο κομμάτι της εφαρμογής έχει να κάνει με την εγγραφή, ανάγνωση ή τροποποίηση των δεδομένων που υπάρχουν στην βάση δεδομένων. Θα αναλυθούν οι σημαντικότερες μέθοδοι της. Στο πρώτο κομμάτι της έχουμε την δήλωση του ονόματος της βάσης δεδομένων, των ονομάτων των πινάκων και των πεδίων αυτών. Έχουμε από έναν πίνακα για τις κατηγορίες, τα προϊόντα, το καλάθι αγορών, τα αγαπημένα και έναν πίνακα-ένωση των πινάκων κατηγορίες και προϊόντα με όνομα κατηγορίες προϊόντων προκειμένου να ορίσουμε μια σχέση πολλά προς πολλά μεταξύ τους. Επίσης υπάρχουν μεταβλητές που έχουν να κάνουν με την διαχείριση και την δημιουργία της βάσης. Έναν DBHelper και μία μεταβλητή τύπου SQLiteDatabase.

```
public class SQLConnect {

    private static final String DATABASE_NAME = "OnShopDB";
    private static final String DATABASE_CATEGORIES = "categories";
    private static final String DATABASE_PRODUCTS = "products";
    private static final String DATABASE_FAVORITES = "favorites";
    private static final String DATABASE_CART = "cart";
    private static final String DATABASE_PRODUCT_CAT = "productCategories";
    private static final int DATABASE_VERSION = 1;

    //Categories
    private static final String KEY_CATEGORY_ID = "id";
    private static final String KEY_CATEGORY_NAME = "name";
    private static final String KEY_CATEGORY_FULLIMAGE = "fullImage";
    private static final String KEY_CATEGORY_THUMBNAIL = "thumbnail";
    private static final String KEY_CATEGORY_PARENTID = "parentID";
    //Products
    private static final String KEY_PRODUCT_WEBID = "webid";
    private static final String KEY_PRODUCT_SKU = "sku";
    private static final String KEY_PRODUCT_NAME = "name";
    private static final String KEY_PRODUCT_SHORTDESC = "shortDescription";
    private static final String KEY_PRODUCT_DESC = "description";
    private static final String KEY_PRODUCT_PRICE = "price_with_VAT";
    private static final String KEY_PRODUCT_FULLIMAGE = "productImage";
    private static final String KEY_PRODUCT_THUMB = "thumbnail"; // Product Category parent
    //Cart
    private static final String KEY_QUANTITY= "quantity";

    private DBHelper myHelper;
    private final Context myContext;
    private SQLiteDatabase myDB;
```

Μετά την δήλωση των μεταβλητών, δημιουργούμε μια κλάση μέσα στην κλάση μας που επεκτείνει την κλάση SQLiteOpenHelper μέσα στην οποία υπάρχουν οι μέθοδοι για τις ενέργειες που εκτελούνται μόλις δημιουργηθεί η βάση, κατά την ανανέωση της καθώς και μεθόδους που

ανοίγουν και κλείνουν την βάση προς ανάγνωση ή εγγραφή. Έτσι λοιπόν έχουμε τις μεθόδους open και close αλλά και την μέθοδο onCreate μέσα στην οποία υπάρχουν οι SQL εντολές που δημιουργούν τους πίνακες της βάσης. Τέλος υπάρχει και η μέθοδος onUpgrade που καλείται όταν έχουμε αλλάξει την έκδοση της βάσης.

```
private static class DBHelper extends SQLiteOpenHelper {

        public DBHelper(Context context) {
                super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
                //Categories Table
                db.execSQL("CREATE TABLE " + DATABASE_CATEGORIES + " (" +
                KEY_CATEGORY_ID + " INTEGER PRIMARY KEY, " + KEY_CATEGORY_NAME
                + " TEXT, " + KEY_CATEGORY_FULLIMAGE
                + " TEXT, " + KEY_CATEGORY_THUMBNAIL
                + " TEXT, " + KEY_CATEGORY_PARENTID + " INTEGER );");

                //Products Table
                db.execSQL("CREATE TABLE " + DATABASE_PRODUCTS + " (" +
                KEY_PRODUCT_WEBID + " TEXT PRIMARY KEY , " + KEY_PRODUCT_SKU
                + " TEXT, " + KEY_PRODUCT_NAME
                + " TEXT, " + KEY_PRODUCT_SHORTDESC
                + " TEXT, " + KEY_PRODUCT_DESC
                + " TEXT, " + KEY_PRODUCT_PRICE
                + " INTEGER, " + KEY_PRODUCT_THUMB
                + " TEXT, " + KEY_PRODUCT_FULLIMAGE + " TEXT );");

                //Product Categories Table
                db.execSQL("CREATE TABLE " + DATABASE_PRODUCT_CAT + " ( "+
                KEY_PRODUCT_WEBID + " TEXT, " + KEY_CATEGORY_ID
                + " INTEGER , PRIMARY KEY ("+ KEY_PRODUCT_WEBID +" , "+
                KEY_CATEGORY_ID+" )" + " FOREIGN KEY("+ KEY_PRODUCT_WEBID +")
                REFERENCES " + DATABASE_PRODUCTS + "("+ KEY_PRODUCT_WEBID +"), "
                + " FOREIGN KEY("+ KEY_PRODUCT_WEBID +") REFERENCES " +
                DATABASE_CATEGORIES + "("+ KEY_CATEGORY_ID +")); ");

                //Favorites Table
                db.execSQL("CREATE TABLE " + DATABASE_FAVORITES + " ( "+
                KEY_PRODUCT_WEBID + " TEXT, FOREIGN KEY("+ KEY_PRODUCT_WEBID +")
                REFERENCES " + DATABASE_PRODUCTS + "("+ KEY_PRODUCT_WEBID +")); ");

                //Cart Table
                db.execSQL("CREATE TABLE " + DATABASE_CART + " ( " + KEY_PRODUCT_WEBID
                + " TEXT, " + KEY_QUANTITY
                + " INTEGER, FOREIGN KEY("+ KEY_PRODUCT_WEBID +") REFERENCES " +
                DATABASE_PRODUCTS + "("+ KEY_PRODUCT_WEBID +")); ");
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
                db.execSQL("DROP TABLE IF EXISTS " + DATABASE_CATEGORIES);
                db.execSQL("DROP TABLE IF EXISTS " + DATABASE_PRODUCTS);
                db.execSQL("DROP TABLE IF EXISTS " + DATABASE_CART);
                db.execSQL("DROP TABLE IF EXISTS " + DATABASE_FAVORITES);
```

```
                db.execSQL("DROP TABLE IF EXISTS " + DATABASE_PRODUCT_CAT);
                onCreate(db);
        }
}

public SQLConnect(Context c) {
        myContext = c;
}

public SQLConnect open() throws SQLException {
        myHelper = new DBHelper(myContext);
        myDB = myHelper.getWritableDatabase();
        return this;
}

public void close() {
        myHelper.close();
}
```

Στο δεύτερο στάδιο, έχουμε τις μεθόδους προσθήκης δεδομένων. Πιο συγκεκριμένα για κάθε πίνακα έχουμε την αντίστοιχη μέθοδο που εισάγει δεδομένα σε αυτόν. Τα δεδομένα προς εισαγωγή δίνονται ως ακέραιοι ή συμβολοσειρές ως ορίσματα της μεθόδου κατά την κλήση της. Έπειτα όλες αυτές οι μεταβλητές εισάγονται σε μία μεταβλητή τύπου δεδομένων ContentValues με όνομα cv και αυτή με τη σειρά της χρησιμοποιείται από την μέθοδο insertOrThrow του αντικειμένου myDB που είναι η ουσιαστικά η βάση μας. Στα ορίσματα της μεθόδου υπάρχει ο πίνακας πάνω στον οποίο δουλεύουμε και η μεταβλητή cv.

```
public long createCategoriesEntry(int catID, String catName, String fullImage, String thumb, int
parentID) {
        ContentValues cv = new ContentValues();
        cv.put(KEY_CATEGORY_ID, catID);
        cv.put(KEY_CATEGORY_NAME, catName);
        cv.put(KEY_CATEGORY_FULLIMAGE, fullImage);
        cv.put(KEY_CATEGORY_THUMBNAIL, thumb);
        cv.put(KEY_CATEGORY_PARENTID, parentID);
        try {
                return myDB.insertOrThrow(DATABASE_CATEGORIES, null, cv);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
                return -1;
        }
}

public long createProductsEntry(String prodWebID, String prodSKU, String prodName, String
shortDesc , String desc , int price, String thumb, String fullImage) {
        ContentValues cv = new ContentValues();
        cv.put(KEY_PRODUCT_WEBID, prodWebID);
        cv.put(KEY_PRODUCT_SKU, prodSKU);
        cv.put(KEY_PRODUCT_NAME, prodName);
        cv.put(KEY_PRODUCT_SHORTDESC, shortDesc);
        cv.put(KEY_PRODUCT_DESC, desc);
        cv.put(KEY_PRODUCT_PRICE, price);
        cv.put(KEY_PRODUCT_THUMB, thumb);
        cv.put(KEY_PRODUCT_FULLIMAGE, fullImage);
```

```
		try {
			return myDB.insertOrThrow(DATABASE_PRODUCTS, null, cv);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
			return -1;
		}
	}
```

Στο επόμενο κομμάτι έχουμε τις μεθόδους ανάκτησης δεδομένων και εμφάνισης στους σε διάφορα views ανά περίπτωση. Πρώτη μέθοδος είναι η categoriesToGrid που ανακτά τα στοιχεία μιας κατηγορίας με βάση το parent id της. Αφού πάρουμε τα δεδομένα με την εκτέλεση της μεθόδου query που παίρνει ως ορίσματα τον πίνακα που δουλεύουμε, έναν πίνακα από strings με τα πεδία που θέλουμε, και την συνθήκη επιλογής (Where). Το αποτέλεσμα της εκτέλεσης της αποθηκεύεται σε έναν cursor που βοηθάει στην διαχείριση των ανακτημένων δεδομένων

```
public ArrayList<HashMap<String, String>> categoriesToGrid(String pID) {

ArrayList<HashMap<String, String>> dataList = new ArrayList<HashMap<String, String>>();

String[] columns = new String[] {KEY_CATEGORY_ID, KEY_CATEGORY_NAME,
KEY_CATEGORY_THUMBNAIL};
Cursor c=null;
try {
	c = myDB.query(DATABASE_CATEGORIES, columns, KEY_CATEGORY_PARENTID+ "=" +
pID, null, null, null, null);
} catch (Exception e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
}
```

Μετά με βάση το index κάθε στήλης που βρίσκεται ο cursor, παίρνουμε τα δεδομένα που θέλουμε και τα τοποθετούμε σε ένα hashmap που με τη σειρά του προστίθεται σε έναν ArrayList. Το τελευταίο επιστρέφεται και χρησιμοποιείται για την εμφάνιση των δεδομένων σε άλλες κλάσεις.

```
int icatID = c.getColumnIndex(KEY_CATEGORY_ID);
int icatName = c.getColumnIndex(KEY_CATEGORY_NAME);
int icatThumb = c.getColumnIndex(KEY_CATEGORY_THUMBNAIL);

for (c.moveToFirst(); !c.isAfterLast(); c.moveToNext()) {

HashMap<String, String> map = new HashMap<String, String>();

map.put("catID", c.getString(icatID));
map.put("catName", c.getString(icatName));
map.put("catThumb", c.getString(icatThumb));
dataList.add(map);

}
return dataList;
}
```

Παρακάτω υπάρχει μια παρόμοια μέθοδος, με την βασική διαφορά πως χρησιμοποιούμε την μέθοδο rawQuery, που εκτελεί ένα ερώτημα που γράφει ο προγραμματιστής. Εδώ για παράδειγμα έχουμε ένα ερώτημα ένωσης των πινάκων Κατηγορίες και Προϊόντα για να έχουμε ως αποτέλεσμα όλες τις κατηγορίες στις οποίες ανήκει ένα προϊόν.

```
public ArrayList<HashMap<String, String>> catProductsToList(String catid, String order) {

ArrayList<HashMap<String, String>> dataList = new ArrayList<HashMap<String, String>>();

        Cursor c=null;
        try {
        c = myDB.rawQuery("Select p.webid, p.sku, p.name, p.shortDescription,
        p.description, p.price_with_VAT, p.productImage, p.thumbnail" +
        " From products p" +
        " Inner Join productCategories pc On pc.webid = p.webid" +
        " Inner Join categories c On c.id = pc.id" +
        " Where c.id = "+catid+" "+order+"" ,null);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
}

int iprodWebID = c.getColumnIndex(KEY_PRODUCT_WEBID);
int iprodSKU = c.getColumnIndex(KEY_PRODUCT_SKU);
int iprodName = c.getColumnIndex(KEY_PRODUCT_NAME);
int iprodSDesc = c.getColumnIndex(KEY_PRODUCT_SHORTDESC);
int iprodDesc = c.getColumnIndex(KEY_PRODUCT_DESC);
int iprodPrice = c.getColumnIndex(KEY_PRODUCT_PRICE);
int iprodImage = c.getColumnIndex(KEY_PRODUCT_FULLIMAGE);
int iprodThumb = c.getColumnIndex(KEY_PRODUCT_THUMB);

for (c.moveToFirst(); !c.isAfterLast(); c.moveToNext()) {

HashMap<String, String> map = new HashMap<String, String>();

map.put("prodWebID", c.getString(iprodWebID));
map.put("prodSKU", c.getString(iprodSKU));
map.put("prodName", c.getString(iprodName));
map.put("prodSDesc", c.getString(iprodSDesc));
map.put("prodDesc", c.getString(iprodDesc));
map.put("prodPrice", c.getString(iprodPrice));
map.put("prodImage", c.getString(iprodImage));
map.put("prodThumb", c.getString(iprodThumb));

dataList.add(map);

}
return dataList;
}
```

Μια βοηθητική μέθοδος που κάνει έλεγχο για το αν ένα προϊον υπάρχει στα αγαπημένα έτσι ώστε να ξέρουμε αν όταν θα εμφανιστούν οι λεπτομέρειες του θα πρέπει να το προσθέσουμε ή να το αφαιρέσουμε από αυτά είναι η παρακάτω που απλά εκτελεί ένα ερώτημα για τον έλεγχο ενός συγκεκριμένου προϊόντος και αν το αποτέλεσμα είναι κενό τότε επιστρέφει true και σε αντίθετη

περίπτωση false. Ίδια μέθοδος υπάρχει και για τον έλεγχο ύπαρξης ενός προϊόντος στο καλάθι αγορών.

```
public boolean existsInFavorites(String webID) {
        String[] columns = new String[] {KEY_PRODUCT_WEBID};
        Cursor c=null;
        try {
                c = myDB.query(DATABASE_FAVORITES, columns , "webid=?", new String[]
                {webID}, null, null, null);
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        if (c.getCount()==0) {
                return false;
        }
        else {
                return true;
        }
}
```

Μια ακόμα απλή μέθοδος, είναι η μέθοδος διαγραφής ενός προϊόντος από το καλάθι αγορών και υπάρχει και για την διαγραφή από τα αγαπημένα. Καλείται λοιπόν η μέθοδος delete, με όρισμα το id του προϊόντος που θέλουμε να διαγράψουμε.

```
public void deleteFromCart(String webID) {
        try {
                myDB.delete(DATABASE_CART, "webid=?", new String[] {webID});
        } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
```

Στο κεφάλαιο 11.20 είχε αναφερθεί πως πρέπει να εκτελεστεί ένα ερώτημα που επιστρέφει όλα τα υπάρχοντα προϊόντα και ένα ακόμα που επιστρέφει όλες τις κατηγορίες για να μπορούμε να έχουμε τη δυνατότητα του auto complete. Αυτό γίνεται μέσω δύο μεθόδων που είναι όμοιες μεταξύ τους και έτσι παραθέτουμε μόνο την μία που αφορά τα προϊόντα.

```
public String[] getAllProducts() {
Cursor cursor=null;
try {
        cursor = myDB.query(DATABASE_PRODUCTS, new String[] {KEY_PRODUCT_NAME}, null,
        null, null, null, null);
} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

 if(cursor.getCount() >0) {
String[] str = new String[cursor.getCount()];
int i = 0;
while (cursor.moveToNext())
```

```
        {
            str[i] = cursor.getString(cursor.getColumnIndex(KEY_PRODUCT_NAME));
            i++;
         }
        return str;
        } else {
        return new String[] {};
    }
}
```

Τέλος υπάρχει η μέθοδος searchProduts που εκτελεί την αναζήτηση στον πίνακα των προϊόντων με βάση τα κριτήρια που έδωσε ο χρήστης. Χρησιμοποιείται και εδώ η μέθοδος rawQuery με ένα ερώτημα στο οποίο υπάρχει η γραμμή " WHERE p.name LIKE '%"+terms+"%'". Αυτό είναι και το σημείο το οποίο γίνεται η επιλογή των προϊόντων. Υπάρχει και η μέθοδος searchCategories που κάνει την αντίστοιχη δουλειά για τις κατηγορίες.

```
public ArrayList<HashMap<String, String>> searchProducts(String terms) {

ArrayList<HashMap<String, String>> dataList = new ArrayList<HashMap<String, String>>();

        String sql = " SELECT p.webid, p.sku, p.name, p.shortDescription, p.description, p.price_with_VAT,
        p.productImage, p.thumbnail" +
        " FROM products p" +
        " WHERE p.name LIKE '%"+terms+"%'";
        Cursor c=null;
                try {
                        c = myDB.rawQuery(sql, null);
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        int iprodWebID = c.getColumnIndex(KEY_PRODUCT_WEBID);
        int iprodSKU = c.getColumnIndex(KEY_PRODUCT_SKU);
        int iprodName = c.getColumnIndex(KEY_PRODUCT_NAME);
        int iprodSDesc = c.getColumnIndex(KEY_PRODUCT_SHORTDESC);
        int iprodDesc = c.getColumnIndex(KEY_PRODUCT_DESC);
        int iprodPrice = c.getColumnIndex(KEY_PRODUCT_PRICE);
        int iprodImage = c.getColumnIndex(KEY_PRODUCT_FULLIMAGE);
        int iprodThumb = c.getColumnIndex(KEY_PRODUCT_THUMB);

        for (c.moveToFirst(); !c.isAfterLast(); c.moveToNext()) {

        HashMap<String, String> map = new HashMap<String, String>();

        map.put("prodWebID", c.getString(iprodWebID));
        map.put("prodSKU", c.getString(iprodSKU));
        map.put("prodName", c.getString(iprodName));
        map.put("prodSDesc", c.getString(iprodSDesc));
        map.put("prodDesc", c.getString(iprodDesc));
        map.put("prodPrice", c.getString(iprodPrice));
        map.put("prodImage", c.getString(iprodImage));
        map.put("prodThumb", c.getString(iprodThumb));

        dataList.add(map);
```

```
        }
        return dataList;

    }
}
```

11.25 Sync.java

Ο συγχρονισμός των προϊόντων είναι μια διαδικασία όμοια με αυτή του αρχικού συγχρονισμού για αυτό και δεν θα αναλυθεί ο κώδικας της κλάσης αυτής. Η μόνη διαφορά είναι πως στο url που χρησιμοποιούμε προστίθεται μία ακόμα παράμετρος, η LastUpdatedOn η οποία επιστρέφει τα προϊόντα και τις κατηγορίες που έχουν τροποποιηθεί έπειτα από ένα συγκεκριμένο timestamp.

11.26 Utils.java

Η κλάση αυτή περιέχει μερικές βοηθητικές μεθόδους που χρησιμοποιούνται από άλλες κλάσεις για να αυτοματοποιηθούν μερικές διαδικασίες και υπολογισμοί. Η ανάλυση της δεν είναι αναγκαία για τις ανάγκες της εργασίας.

11.27 Vieproduct.java

Η κλάση αυτή αναλαμβάνει την δημιουργία του Activity, που προβάλλει τις λεπτομέρειες ενός προϊόντος. Αρχικά παραλαμβάνουμε όλα τα δεδομένα που μας έχουν σταλεί από το προηγούμενο Activity για να τα χρησιμοποιήσουμε για την εμφάνιση τίτλου, sku, εικόνων, περιγραφής και τιμής.

```
public class ViewProduct extends Activity {

        private SQLConnect addSQLtoList = new SQLConnect(this);
        private CheckBox addToCart;
        private String pWebID;

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

                // Pairnoume dedomena apo to proigoumeno intent

                Intent in = getIntent();

                pWebID = in.getStringExtra("prodWebID");
                String pName = in.getStringExtra("prodName");
                String pDesc = in.getStringExtra("prodDesc");
```

```
String pPrice = in.getStringExtra("prodPrice");
String pImage = in.getStringExtra("prodImage");
String pSKU = in.getStringExtra("prodSKU");
// Emfanisi ton dedomenon

TextView prodWebID = (TextView) findViewById(R.id.tvViewProdWebID);
TextView prodSKU = (TextView) findViewById(R.id.tvViewProdSKU);
TextView prodName = (TextView) findViewById(R.id.tvViewProdName);
TextView prodDesc = (TextView) findViewById(R.id.tvViewProdDesc);
TextView prodPrice = (TextView) findViewById(R.id.tvViewProdPrice);
ImageView prodImage = (ImageView) findViewById(R.id.ivViewProdImage);
addToCart = (CheckBox) findViewById(R.id.cbAddToCart);
CheckBox addToFav = (CheckBox) findViewById(R.id.cbAddToFav);
```

Έπειτα, γίνεται έλεγχος για το αν το προϊόν υπάρχει στο καλάθι αγορών ή στα αγαπημένα έτσι ώστε να εμφανιστεί η αντίστοιχη εικόνα αφαίρεσης η προσθήκης καθώς και να οριστούν οι κατάλληλες ενέργειες για το κλικ επάνω στο checkbox που δείχνει την ύπαρξη η μη του προϊόντος σε κάθε έναν από τους δύο αυτούς πίνακες της βάσης.

```
addSQLtoList.open();
boolean isInCart = addSQLtoList.existsInCart(pWebID);
boolean isInFav = addSQLtoList.existsInFavorites(pWebID);
addSQLtoList.close();

if (isInCart==false) {
addToCart.setChecked(false);
}

else {
addToCart.setChecked(true);
}

if (isInFav==false) {
addToFav.setChecked(false);
}

else {
addToFav.setChecked(true);
}

prodWebID.setText(pWebID);
prodSKU.setText("SKU: " + pSKU);
prodName.setText(pName);
prodDesc.setText(pDesc);
prodPrice.setText("Τιμή: " + pPrice);
ImageLoader imageLoader = new ImageLoader(getApplicationContext());
imageLoader.DisplayImage(pImage,prodImage);

addToCart.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
```

Παρακάτω αναλύονται οι διαδικασίες που γίνονται για την εισαγωγή και την διαγραφή ενός προϊόντος από το καλάθι αγορών και από τα αγαπημένα. Όλες οι μέθοδοι καλούν την αντίστοιχη μέθοδο της κλάσης SQLConnect με όρισμα το id του προϊόντος και έπειτα εμφανίζουν ένα κατάλληλο μήνυμα Toast, με μόνη διαφορά πως κατά την προσθήκη στο καλάθι που πριν να γίνει η

κλήση της μεθόδου για την εισαγωγή στην βάση, εμφανίζεται ένας διάλογος εισαγωγής της επιθυμητής ποσότητας και γίνεται έλεγχος πως η ποσότητα που εισήγαγε ο χρήστης είναι αριθμός, μέσω της μεθόδου isNumeric.

```
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

if (!isChecked) {
        addSQLtoList.open();
        addSQLtoList.deleteFromCart(pWebID);
        addSQLtoList.close();

} else {

AlertDialog.Builder alert = new AlertDialog.Builder(ViewProduct.this);
alert.setTitle("Ποσότητα");
alert.setMessage("Εισάγετε την επιθυμητή ποσότητα");
final EditText input = new EditText(getBaseContext());
input.setInputType(InputType.TYPE_CLASS_NUMBER);
alert.setView(input);

alert.setPositiveButton("OK", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int whichButton) {
String value = input.getText().toString();
onBackPressed();
if(isNumeric(value)) {
addSQLtoList.open();
addSQLtoList.createCartEntry(pWebID ,Integer.parseInt(value));
addSQLtoList.close();
Toast t = Toast.makeText(ViewProduct.this, "Το προϊόν προστέθηκε στο καλάθι",
Toast.LENGTH_LONG);
t.show();
} else {
Toast t = Toast.makeText(ViewProduct.this, "Θα πρέπει να εισάγετε ακέραιο αριθμό",
Toast.LENGTH_LONG);
t.show();
addToCart.setChecked(false);

        }

 }
});

alert.setNegativeButton("Ακύρωση", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int whichButton) {
addToCart.setChecked(false);
dialog.dismiss();
         }
});

alert.setOnCancelListener(new DialogInterface.OnCancelListener() {

@Override
public void onCancel(DialogInterface dialog) {
addToCart.setChecked(false);
        }
});
```

```
alert.show();
} }
});

addToFav.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
if (!isChecked) {
        addSQLtoList.open();
        addSQLtoList.deleteFromFav(pWebID);
        Toast t = Toast.makeText(ViewProduct.this, "Το προϊόν αφαιρέθηκε από τα
        αγαπημένα", Toast.LENGTH_LONG);
        t.show();
        addSQLtoList.close();

} else {
        addSQLtoList.open();
        addSQLtoList.createFavoritesEntry(pWebID);
        addSQLtoList.close();
        Toast t = Toast.makeText(ViewProduct.this, "Το προϊόν προστέθηκε στα
        αγαπημένα", Toast.LENGTH_LONG);
        t.show();
        }
}
});
}
```

Αυτή είναι η μέθοδος isNumeric που ελέγχει το κατά πόσο ένα string περιέχει μόνο αριθμούς.

```
public boolean isNumeric(String string) throws IllegalArgumentException
{
  boolean isnumeric = false;
  if (string != null && !string.equals(""))  {
        isnumeric = true;
        char chars[] = string.toCharArray();

        for(int d = 0; d < chars.length; d++)
                {
      isnumeric &= Character.isDigit(chars[d]);
      if(!isnumeric)
      break;
    }
  }
  return isnumeric;
}

}
```

11.28 XMLParser.java

Μια ακόμα εξαιρετικά χρήσιμη κλάση είναι η XMLParser, σε αυτή περιγράφονται οι ενέργειες που γίνονται για το parsing των δεδομένων που μας δίνονται από τα αρχεία XML των web services. Πρώτη μέθοδος που βλέπουμε είναι getXmlFromUrl που με τη χρήση ενός HTTP request παίρνουμε ένα έγγραφο xml.

```
public class XMLParser {
	public XMLParser() {

	}
	public String getXmlFromUrl(String url) {
		String xml = null;

		try {
			// defaultHttpClient
			DefaultHttpClient httpClient = new DefaultHttpClient();
			HttpPost httpPost = new HttpPost(url);
			HttpResponse httpResponse = httpClient.execute(httpPost);
			HttpEntity httpEntity = httpResponse.getEntity();
			xml = EntityUtils.toString(httpEntity);

		} catch (UnsupportedEncodingException e) {
			e.printStackTrace();
		} catch (ClientProtocolException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		}
		// return XML
		return xml;
	}
```

Στη συνέχεια, από το έγγραφο XML που λάβαμε νωρίτερα, παίρνουμε τα dom elements που περιέχει ενώ με τις μεθόδους getElementValue και getValue κρατάμε τα περιεχόμενα ενός συγκεκριμένου κόμβου ή τα περιεχόμενα που βρίσκονται μεταξύ ενός συγκεκριμένου element αντίστοιχα.

```
	public Document getDomElement(String xml) {
		Document doc = null;
		DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
		try {

		DocumentBuilder db = dbf.newDocumentBuilder();

		InputSource is = new InputSource();
		is.setCharacterStream(new StringReader(xml));
		doc = db.parse(is);

		} catch (ParserConfigurationException e) {
			Log.e("Error: ", e.getMessage());
			return null;
		} catch (SAXException e) {
			Log.e("Error: ", e.getMessage());
			return null;
		} catch (IOException e) {
```

```
                Log.e("Error: ", e.getMessage());
                return null;
            }

            return doc;
        }

        public final String getElementValue(Node elem) {
            Node child;
            if (elem != null) {
                if (elem.hasChildNodes()) {
                    for (child = elem.getFirstChild(); child != null; child = child
                            .getNextSibling()) {
                        if (child.getNodeType() == Node.TEXT_NODE) {
                            return child.getNodeValue();
                        }
                    }
                }
            }
            return "";
        }

        public String getValue(Element item, String str) {
            NodeList n = item.getElementsByTagName(str);
            return this.getElementValue(n.item(0));
        }
}
```

12. Ανάλυση των αρχείων XML

Κατά την ανάπτυξη μιας android εφαρμογής, γίνεται πολύ συχνή χρήση για διάφορους σκοπούς. Ο πιο συνηθισμένος και πιο σημαντικός λόγος χρήσης τους είναι για την δημιουργία αρχείων layout με τα οποία δημιουργείται το γραφικό περιβάλλον της εφαρμογής. Εκτός από αυτό μπορούμε να τα χρησιμοποιήσουμε για αποθήκευση μεταβλητών, για την δημιουργία διαφόρων στυλ και θεμάτων για την εφαρμογή μας, για τη δημιουργία μενού καθώς και για τις γενικές ρυθμίσεις χρήστη. Παρακάτω θα αναλύσουμε όλα τα προαναφερθέντα καθώς έχουν χρησιμοποιηθεί για την ανάπτυξη της εφαρμογής. Αξίζει βέβαια να πούμε πως τα αρχεία XML και ειδικότερα τα αρχεία XML Layout, παρουσιάζουν πολλές ομοιότητες μεταξύ τους και έτσι δεν χρειάζεται να τα αναλύσουμε όλα. Θα γίνει αναφορά στα σημαντικότερα εξ αυτών.

12.1 Αρχεία XML Layout

12.1.1 About.xml

Ξεκινώντας την ανάλυση με ένα από τα απλά αρχεία XML που υπάρχουν. Αυτό αναλαμβάνει το γραφικό περιβάλλον για το Activity Που εμφανίζει πληροφορίες σχετικά με την εφαρμογή. Παρακάτω φαίνεται ο κώδικας του.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:gravity="center_vertical|center_horizontal"
   android:orientation="vertical"
   android:background="#FFFFFF" >

  <TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:padding="15dp"
     android:text="OnShop"
     android:textColor="#333333"
     android:textSize="24dp" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:padding="15dp"
      android:text="Ζήκος Δήμος - Κοροσίδης Αντώνιος"
      android:textColor="#333333"
      android:textSize="16dp" />
</LinearLayout>
```

Αρχικά βλέπουμε την δήλωση πως το αρχείο αυτό αποτελεί ένα αρχείο XML, την έκδοση που χρησιμοποιούμε και την κωδικοποίηση, κάτι που γίνεται σε όλα τα αρχεία XML. Έπειτα γίνεται ο ορισμός ενός control/view του LinearLayout. Αυτό τοποθετεί όλα τα controls που περιέχει σε μία οριζόντια ή κάθετη γραμμική σειρά ανάλογα με το τον προσανατολισμό που του έχουμε δώσει στην ιδιότητα android:orientation. Στη συγκεκριμένη περίπτωση έχουμε κάθετο προσανατολισμό.

Άλλες σημαντικές ιδιότητες που υπάρχουν στα περισσότερα controls είναι το πλάτος και το ύψος με όνομα layout_width και layout_height αντίστοιχα και τιμές που μπορεί να είναι wrap_content εάν θέλουμε να είναι σε μέγεθος όσο ακριβώς και τα περιεχόμενα του ή fill_parent/match_parent αν θέλουμε να καταλαμβάνει όλο τον διαθέσιμο χώρο του parent view στο οποίο βρίσκεται. Επίσης το χρώμα φόντου (background) και το gravity που ορίζει την στοίχιση των περιεχομένων.

Μέσα στο στοιχείο αυτό υπάρχουν δύο TextView, με τη χρήση των οποίων μπορούμε να εμφανίσουμε ένα επιθυμητό κείμενο. Η ιδιότητα text ορίζει τι κείμενο θα περιέχουν ενώ το textSize και το textColor το μέγεθος και το χρώμα του κειμένου.

12.1.2 Cart_row

Επόμενο αρχείο προς ανάλυση είναι το cart_row.xml, που ορίζει την εμφάνιση που θα έχει στο γραφικό περιβάλλον κάθε γραμμή με ένα προϊόν στο Activity του καλαθιού αγορών. Μέσα λοιπόν σε ένα LinearLayout που είδαμε παραπάνω, υπάρχει ένα TableRow που ορίζει την γραμμή ενός πίνακα μέσα σε ένα TableLayout που θα δούμε στη συνέχεια. Κάτι που αξίζει να αναφερθεί είναι πως το TableRow έχει για φόντο ένα άλλο αρχείο xml το cell.xml που του προσδίδει κάποιες ιδιότητες. Περισσότερα για το αρχείο αυτό υπάρχουν στο επόμενο υποκεφάλαιο.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TableRow
        android:id="@+id/cartRow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/cell" >

        <TextView
            android:id="@+id/tvCartTitle"
            android:layout_width="120dp"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:paddingLeft="10dp"
            android:paddingRight="5dp"
            android:paddingTop="10dp"
            android:text="TextView"
            android:textColor="#777777" />

        <TextView
            android:id="@+id/tvCartQuan"
            android:layout_width="50dp"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:paddingLeft="10dp"
            android:paddingRight="5dp"
            android:paddingTop="10dp"
            android:text="1000"
            android:textColor="#777777" />

        <Button
            android:id="@+id/btnCartRemove"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dp"
            android:layout_marginRight="30dp"
            android:background="@drawable/delete" />

        <Button
            android:id="@+id/btnProdSubTotal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:visibility="gone" />

        <TextView
            android:id="@+id/tvCartSubTotal"
            android:layout_width="70dp"
            android:layout_height="wrap_content"
            android:paddingBottom="10dp"
            android:paddingLeft="10dp"
            android:paddingRight="5dp"
            android:paddingTop="10dp"
            android:text="1000"
            android:textColor="#777777" />
    </TableRow>
</LinearLayout>
```

Μέσα στο TableRow υπάρχουν πλέον και Button τα οποία ορίζουν όπως προδίδει και το όνομα τους κουμπιά. Οι ιδιότητες τους είναι αντίστοιχες των textViews ενώ στο background τους μπορούμε να ορίσουμε μία εικόνα έτσι ώστε το κουμπί να φαίνεται ως μία εικόνα ορισμένη από εμάς και να μην έχει την default εμφάνιση του. Κάτι επίσης πολύ σημαντικό που βλέπουμε στο αρχείο αυτό είναι η ιδιότητα id, με βάση την οποία το κάθε view αντιστοιχίζεται σε έναν ακέραιο στο R.java που αναλύσαμε στο προηγούμενο κεφάλαιο και έπειτα μπορούμε μέσω αυτού να αναφερθούμε στο view αυτό στον κώδικα μας.

12.1.3 Categ.xml

Το αρχείο αυτό περιέχει το GridView που αναλαμβάνει να εμφανίσει όλες τις κατηγορίες σε μορφή πλέγματος. Μερικές από τις σημαντικές ιδιότητες του είναι το numColumns που ορίζει πόσες στήλες θα έχει, το strechMode που ορίζει πως θα διαχειρίζεται το πλέγμα σε κάθε διάφορα μεγέθη οθονών, εδώ έχει οριστεί ως columnWidth, κάτι που έχει ως αποτέλεσμα το «τέντωμα» των στηλών ώστε να χωρέσουν σωστά στην οθόνη. Τέλος επίσης σημαντικές ιδιότητες είναι το verticalSpacing και horizontalSpacing που ορίζουν το κάθετο και οριζόντιο κενό που υπάρχει ανάμεσα στα στοιχεία του πλέγματος.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF"
    android:orientation="vertical" >

    <GridView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/gvCategories"
```

```
        android:layout_width="match_parent"
        android:layout_height="fill_parent"
        android:background="#FFFFFF"
        android:gravity="center_vertical|center_horizontal"
        android:horizontalSpacing="2dp"
        android:numColumns="3"
        android:stretchMode="columnWidth"
        android:verticalSpacing="2dp" >
    </GridView>

    <TextView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/emptygrid"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF"
        android:paddingLeft="10dp"
        android:paddingTop="15dp"
        android:text="Δεν υπάρχουν κατηγορίες"
        android:textColor="#555555"
        android:visibility="gone" />
</LinearLayout>
```

Εδώ να εξηγήσουμε την χρήση του TextView που εμφανίζεται μόνο στην περίπτωση που το GridView είναι κενό ώστε να μας δώσει ένα κατάλληλο μήνυμα. Παρατηρούμε πως η ιδιότητα visibility είναι gone που σημαίνει πως δεν είναι ορατό και αυτό αλλάζει μόνο στην περίπτωση που το ζητήσει ο προγραμματιστής όταν δηλαδή το GridView δεν περιέχει τίποτα. Η χρήση ενός τέτοιου TextView γίνεται και σε πολλά άλλα αρχεία XML όπως σε αυτά με ListViews.

12.1.4 Categories.xml

Το αρχείο categories.xml είναι αυτό που αναλαμβάνει την εμφάνιση των αντικειμένων του που θα τοποθετούνται στο GridView που αναλύθηκε πριν. Επίσης σε αυτό γίνεται η χρήση του RelativeLayout, ένα είδος στοίχισης που κάθε στοιχείο τοποθετείται με βάση την θέση ενός άλλου στοιχείου, υπάρχουν δηλαδή σχέσεις μεταξύ τους όσον αφορά την στοίχιση. Βλέπουμε λοιπόν την ιδιότητα android:layout_below="@+id/ivCategoryThumb" στο TextView με id CategoryName που ορίζει πως το στοιχείο αυτό θα βρίσκεται κάτω από το ImageView με id ivCategoryThumb. Νέο επίσης στοιχείο είναι τα ImageView που μόλις αναφέρθηκαν. Αυτά απλά εμφανίζουν μία εικόνα από τα resources και μπορούν να τις δώσουν μερικές ιδιότητες όπως τα κενά τριγύρω της, για παράδειγμα android:paddingTop="5dp" και android:paddingBottom="5dp"

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="110dp"
    android:layout_height="100dp"
    android:background="@drawable/myborder"
    android:gravity="center_vertical|center_horizontal">

    <ImageView
        android:id="@+id/ivCategoryThumb"
        android:paddingTop="5dp"
        android:paddingBottom="5dp"
        android:layout_width="80dp"
```

```
        android:layout_height="70dp"
        android:src="@drawable/btn_contact"
        android:background="@null"
        android:scaleType="centerInside" />

    <TextView
        android:id="@+id/tvCategoryName"
        android:layout_width="80dp"
        android:layout_height="40dp"
        android:layout_below="@+id/ivCategoryThumb"
        android:gravity="center_horizontal"
        android:textColor="#000000"
            android:text="hardware - hardwareeee"
            android:background="@null"
            android:textSize="12dp"/>

    <TextView
        android:id="@+id/tvCategoryID"
            android:visibility="gone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" /> </RelativeLayout>
```

Κάτι ακόμα που πρέπει να σημειωθεί είναι η χρήση TextViews όπως το παραπάνω, που δεν είναι ορατά αλλά χρησιμεύουν στην αποθήκευση κάποιων τιμών που δεν είναι αναγκαίο να φαίνονται αλλά είναι απαραίτητα για την εκτέλεση ορισμένων λειτουργιών. Έτσι λοιπόν εδώ υπάρχει ένα TextView που για κάθε κατηγορία, κρατάει ως τιμή το id της χωρίς αυτό να φαίνεται πουθενά, κάτι που όμως είναι πολύ σημαντικό για την εκτέλεση ερωτημάτων προς την βάση.

12.1.5 Contact.xml

Το αρχείο αυτό αναλαμβάνει την εμφάνιση της οθόνης επικοινωνίας στην χρήστη. Όλα τα στοιχεία είναι γνωστά εκτός του στοιχείο View. Με τα στοιχεία view μπορούμε να πετύχουμε κάποια πρακτικά πράγματα, όπως την σχεδίαση γραμμών, τον χρωματισμό περιοχών και τη δημιουργία ενός κενού ανάμεσα σε άλλα views. Εδώ χρησιμοποιείται για να ορίσει διαχωριστικές γραμμές ανάμεσα στους τρόπους επικοινωνίας, έχει δηλαδή ύψος μόνο 1dp που δίνει την ψευδαίσθηση της γραμμής και ένα χρώμα της επιλογής μας.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#333333"
    android:orientation="vertical"
    android:weightSum="3" >

    <Button
        android:id="@+id/facebook"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:background="@android:color/transparent"
        android:drawableLeft="@drawable/facebook"
        android:drawablePadding="12dp"
        android:text="http://www.facebook.com/dzipel"
        android:textColor="#FFFFFF" />
```

```
        <View android:layout_width="fill_parent"
      android:layout_height="1dp"
      android:background="#FFFFFF"></View>

   <Button
      android:id="@+id/mail"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:layout_weight="1"
      android:background="@android:color/transparent"
      android:drawableLeft="@drawable/email"
      android:drawablePadding="12dp"
      android:text="dzipel@gmail.com"
      android:textColor="#FFFFFF" />

        <View android:layout_width="fill_parent"
      android:layout_height="1dp"
      android:background="#FFFFFF"></View>

   <Button
      android:id="@+id/phone"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:layout_weight="1"
      android:background="@android:color/transparent"
      android:drawableLeft="@drawable/phone"
      android:drawablePadding="12dp"
      android:text="2310 123456"
      android:textColor="#FFFFFF" />
</LinearLayout>
```

12.1.6 List_row.xml

Το αρχείο List_row.xml αναλαμβάνει να δώσει την επιθυμητή εμφάνιση σε κάθε γραμμή ενός ListView. Και εδώ χρησιμοποιείται το RelativeLayout, που περιέχει ένα ImageView που εμφανίζει το thumbnail του προϊόντος, ένα textview για τον τίτλο, ένα για την σύντομη περιγραφή, ένα για την τιμή και μια εικόνα που εμφανίζει ένα διακοσμητικό βελάκι σε κάθε γραμμή. Τα υπόλοιπα TextViews χρησιμεύουν στην αποθήκευση πληροφοριών για κάθε προϊόν όπως αναφέρθηκε και νωρίτερα.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:background="@drawable/list_selector"
   android:orientation="horizontal"
   android:padding="5dip" >

   <!-- ListRow Left sided image -->

   <LinearLayout
      android:id="@+id/thumbnail"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentLeft="true"
```

```
    android:layout_marginRight="5dip"
    android:background="@drawable/image_bg"
    android:padding="3dip" >

    <ImageView
        android:id="@+id/ivListProdThumb"
        android:layout_width="50dip"
        android:layout_height="50dip" />
</LinearLayout>

<!-- Product Name -->

<TextView
    android:id="@+id/tvListProdName"
    android:layout_width="190dp"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/thumbnail"
    android:layout_toRightOf="@+id/thumbnail"
    android:text="Default Product"
    android:textColor="#040404"
    android:textSize="14dp"
    android:textStyle="bold"
    android:typeface="sans" />

<TextView
    android:id="@+id/tvListProdSDesc"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/tvListProdName"
    android:layout_marginTop="2dip"
    android:layout_toRightOf="@+id/thumbnail"
    android:text="Default Description"
    android:textColor="#343434"
    android:textSize="12dp" />

<TextView
    android:id="@+id/tvListProdPrice"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignTop="@id/tvListProdName"
    android:layout_marginRight="5dip"
    android:gravity="right"
    android:text="10"
    android:textColor="#10bcc9"
    android:textSize="12dip"
    android:textStyle="bold" />

<!-- Right sided arrow image -->

<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_centerVertical="true"
    android:src="@drawable/arrow" />

<TextView
    android:id="@+id/tvListProdSKU"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:visibility="gone" />

<TextView
    android:id="@+id/tvListProdDesc"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone" />

    <TextView
        android:id="@+id/tvListProdImage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone" />

    <TextView
        android:id="@+id/tvListProdWebID"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone" />
</RelativeLayout>
```

12.1.7 Main_layout.xml

Στο Layout αυτό υπάρχουν μόνο κουμπιά καθώς αποτελεί το γραφικό περιβάλλον του κεντρικού μενού. Μια όμως πολύ βασική διαφορά είναι πως δεν χρησιμοποιεί ένα από τα προκαθορισμένα Layout αλλά το DashboardLayout που υπάρχει στο πακέτο της εφαρμογής. Το αρχείο DashboardLayout.java περιγράφει πως πρέπει να γίνεται η στοίχιση των αντικειμένων στο συγκεκριμένο είδος Layout και έχουμε μιλήσει για αυτό στο προηγούμενο κεφάλαιο. Επίσης κάθε κουμπί που ορίζεται έχει σαν ιδιότητα style το DashboardButton κάτι που τους δίνει ορισμένες κοινές ιδιότητες που ορίζονται σε ένα άλλο αρχείο XML που θα δούμε παρακάτω.

```
<com.onshop.anzip.DashboardLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/mainback" >

    <Button
        android:id="@+id/btn_products"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_products"
        android:text="Προϊόντα" />

    <Button
        android:id="@+id/btn_nproducts"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_nproducts"
        android:text="Νέα\nΠροϊόντα" />

    <Button
        android:id="@+id/btn_sales"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_sales"
        android:text="Προσφορές" />

    <Button
        android:id="@+id/btn_favorites"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_favorites"
        android:text="Αγαπημένα" />
```

```
    <Button
        android:id="@+id/btn_contact"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_contact"
        android:text="Επικοινωνία" />

    <Button
        android:id="@+id/btn_news"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_news"
        android:text="Newsletter" />

    <Button
        android:id="@+id/btn_settings"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_settings"
        android:text="Διαχείριση\nΛογαριασμού" />

    <Button
        android:id="@+id/btn_cart"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_cart"
        android:text="Καλάθι\nΑγορών" />

    <Button
        android:id="@+id/btn_search"
        style="@style/DashboardButton"
        android:drawableTop="@drawable/btn_search"
        android:text="Αναζήτηση" />
</com.onshop.anzip.DashboardLayout>
```

12.1.8 Main_list_categ.xml

Στο αρχείο αυτό γίνεται ένας συνδυασμός από views. Συγκεκριμένα, είναι το αρχείο που χρησιμοποιείται για την διαμόρφωση του γραφικού περιβάλλοντος του Products Activity. Συνδυάζει δηλαδή ένα HorizontalScrollView που είναι μια σειρά από οριζόντια αντικείμενα το ένα δίπλα στο άλλο και ένα ListView που υλοποιεί μία λίστα. Το κάθε ένα από αυτά, «γεμίζει» με στοιχεία αφότου γίνουν οι διαδικασίες που περιγράψαμε στο κεφάλαιο 11.18.

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

    <TextView android:layout_width="fill_parent"
        android:layout_height="22dp"
        android:text=" Υποκατηγορίες"
        android:textColor="#333333"
        android:textSize="14dp"
        android:background="@drawable/textlines" />

    <HorizontalScrollView android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#999999" >
```

```
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/categoriesHolder"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#999999"
    android:orientation="horizontal">

    </LinearLayout>
    </HorizontalScrollView>

     <TextView android:layout_width="fill_parent"
        android:layout_height="22dp"
        android:text=" Προϊόντα"
        android:textColor="#333333"
        android:textSize="14dp"
        android:background="@drawable/textlines" />

    <ListView
        android:id="@+id/list2"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:divider="#b5b5b5"
        android:dividerHeight="1dp"
        android:listSelector="@drawable/list_selector"
        android:background="#FFFFFF" />

        <TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/emptylist2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF"
        android:text="Δεν υπάρχουν προϊόντα"
        android:textColor="#555555"
        android:paddingTop="15dp"
                    android:paddingLeft="10dp"
        android:visibility="gone"/>
</LinearLayout>
```

12.1.9 Main_list.xml

Ένα από τα πολύ σημαντικά και συχνά χρησιμοποιούμενο αρχείο XML είναι αυτό που ορίζει μία λίστα. Υπάρχουν οι ιδιότητες divider και dividerHeight που ορίζουν το χρώμα της διαχωριστικής γραμμής των στοιχείων της λίστας καθώς και το ύψος της. Επίσης, η ιδιότητα listSelector ορίζει την εμφάνιση του κάθε στοιχείου. Στην προκειμένη περίπτωση γίνεται η χρήση του αρχείου list_selector που περιέχει τις απαιτούμενες πληροφορίες.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
        android:background="#FFFFFF"
        android:divider="#b5b5b5"
        android:dividerHeight="1dp"
        android:listSelector="@drawable/list_selector" />

        <TextView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/emptylist"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF"
        android:text="Δεν υπάρχουν προϊόντα"
        android:textColor="#555555"
        android:paddingTop="15dp"
                    android:paddingLeft="10dp"
        android:visibility="gone"/>
</LinearLayout>
```

12.1.10 Maps.xml

Προκειμένου να εμφανίσουμε τους χάρτες της Google, κάνουμε την χρήση ενός MapView και ορίζουμε το apiKey που χρειάζεται ως ιδιότητα για να μπορούν να λειτουργούν κανονικά. Περισσότερες πληροφορίες σχετικά με την χρήση των Google Maps υπάρχουν στο κεφάλαιο 8.

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mapView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:apiKey="0M82UEJoiigTNtt3PuAR-ye30_NZt50hr8dPGRg"/>
```

12.1.11 News.xml

Για την προβολή των νέων του καταστήματος γίνεται χρήση ενός νέου control, του WebView, που λειτουργεί όπως ένα παράθυρο ενός browser μέσα στην εφαρμογή δείχνοντας μας μία επιθυμητή ιστοσελίδα.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <WebView
            android:id="@+id/wvNews"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent" />
</LinearLayout>
```

12.1.12 Search.xml

Για την υλοποίηση του γραφικού περιβάλλοντος της οθόνης αναζήτησης ορίζεται το παρακάτω αρχείο XML. Σε αυτό υπάρχουν μερικά controls στα οποία δεν έχει αναφορά ως τώρα. Πρόκειται για τα AutoCompleteTextView, RadioGroup και RadioButton.

Το AutoCompleteTextView είναι ένα πλαίσιο κειμένου στο οποίο μπορούμε να ορίσουμε μία πηγή δεδομένων έτσι ώστε να μας εμφανίζει προτάσεις συμπλήρωσης κατά την πληκτρολόγηση, κάτι που είναι πλέον πολύ διαδεδομένο. Το RadioGroup ορίζει μία ομάδα από RadioButton που περιέχει, ενώ τα RadioButtons είναι τα γνωστά κυκλικά κουμπιά με τα οποία κάνουμε επιλογές.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:background="#FFFFFF"
   android:paddingTop="10dp"
          android:paddingLeft="5dp"
          android:paddingRight="5dp">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Κείμενο προς αναζήτηση:"
      android:textColor="#555555"/>

   <AutoCompleteTextView
      android:id="@+id/acSearch"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:ems="10" >
   </AutoCompleteTextView>


   <Button
      android:id="@+id/btnDoSearch"
      android:layout_width="130dp"
      android:layout_height="wrap_content"
      android:layout_marginBottom="15dp"
      android:drawableRight="@drawable/searchbtn"
      android:text="Αναζήτηση" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Αναζήτηση σε:"
      android:textColor="#555555"
      android:layout_marginBottom="10dp" />

   <RadioGroup
      android:id="@+id/radioGroup1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >

      <RadioButton
         android:id="@+id/rbProducts"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:textColor="#555555"
```

```
        android:text="Προϊόντα"
        android:checked="true"/>

    <RadioButton
        android:id="@+id/rbCategories"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#555555"
        android:text="Κατηγορίες" />
  </RadioGroup>
</LinearLayout>
```

12.2 Αρχεία XML διαφόρων χρήσεων

Όπως αναφέρθηκε και στην αρχή του κεφαλαίου, τα XML αρχεία δεν χρησιμοποιούνται μόνο για την δημιουργία XML Layouts. Παρακάτω παρατίθενται αρχεία XML που εκτελούν άλλες λειτουργίες και υπάρχουν στην εφαρμογή.

12.2.1 Simple_menu.xml

Το simple_menu.xml αποτελεί το αρχείο που αναλαμβάνει να δημιουργήσει ένα αναδυόμενο μενού. Κάθε στοιχείο του μενού ορίζεται ως item και έχει τις ιδιότητες id, icon και title που αντιστοιχούν στο id για την χρήση του στον κώδικα, το εικονίδιο του και στο κείμενο που εμφανίζει.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"><item
      android:id="@+id/maps"
      android:icon="@drawable/ic_menu_home"
      android:title="Καταστήματα"/>
   <item
      android:id="@+id/update"
      android:icon="@drawable/ic_menu_refresh"
      android:title="Συγχρονισμός"/>
   <item
      android:id="@+id/about"
      android:icon="@drawable/ic_menu_about"
      android:title="About"/>
   <item
      android:id="@+id/exit"
      android:icon="@drawable/ic_menu_exit"
      android:title="Exit"/>
</menu>
```

12.2.2 Styles.xml

Με τα αρχεία style, μπορούμε να ορίσουμε ένα στυλ το οποίο υιοθετείται από πολλά controls παρόμοια μεταξύ τους. Μπορούμε να ορίσουμε πλήθος ιδιοτήτων για αυτά και να ισχύουν σε όλα τα controls που θα έχουν οριστεί να ακολουθούν ένα συγκεκριμένο στυλ. Σημαντικό πλεονέκτημα μιας τέτοιας υλοποίησης είναι πως μια αλλαγή στο αρχείο αυτό θα έχει ως αποτέλεσμα την αλλαγή σε όλα τα controls που το χρησιμοποιούν χωρίς να τροποποιήσουμε το καθένα ξεχωριστά.

```
<resources>
   <style name="DashboardButton">
      <item name="android:layout_gravity">center_vertical</item>
      <item name="android:layout_width">wrap_content</item>
      <item name="android:layout_height">wrap_content</item>
      <item name="android:gravity">center_horizontal</item>
      <item name="android:drawablePadding">2dp</item>
      <item name="android:textSize">14dp</item>
      <item name="android:textStyle">bold</item>
      <item name="android:textColor">#FFFFFF</item>
      <item name="android:shadowColor">#535252</item>
      <item name="android:shadowDx">2</item>
      <item name="android:shadowDy">2</item>
      <item name="android:shadowRadius">2</item>
      <item name="android:background">@null</item>
   </style>

  <style name="FooterBar">
      <item name="android:layout_width">fill_parent</item>
      <item name="android:layout_height">30dp</item>
      <item name="android:orientation">horizontal</item>
      <item name="android:background">#dedede</item>
   </style>
</resources>
```

12.2.3 Strings.xml

Στο xml αυτό αποθηκεύονται συμβολοσειρές στις οποίες μπορούμε να αναφερθούμε σε οποιοδήποτε άλλο xml αρχείο ή στον κώδικα μας. Κάθε μία από αυτές έχει ένα αναγνωριστικό, το name. Η τροποποίηση μια συμβολοσειράς έχει ως αποτέλεσμα την αλλαγή του κειμένου που περιέχει σε όλα τα σημεία της εφαρμογής που χρησιμοποιείται χωρίς να είναι ανάγκη να τροποποιηθεί όπου αυτή εμφανίζεται.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app_name">OnShop</string>
   <string name="no_data">Δεν υπάρχουν δεδομένα!</string>

</resources>
```

12.2.4 Preferences.xml

Πρόκειται για ένα αρχείο το οποίο αναλαμβάνει το γραφικό περιβάλλον για τις διάφορες ρυθμίσεις της εφαρμογής που αποθηκεύονται και μετά το πέρας της εκτέλεσής της. Υπάρχουν

πολλά είδη ρυθμίσεων. Εδώ γίνεται χρήση ενός CheckBoxPreference με όνομα FirstTime, αρχική τιμή true και αναγνωριστικό firstRun καθώς και ενός EditTextPreference με αναγνωριστικό lastUpdate. Το πρώτο χρησιμοποιείται για να διαπιστώσουμε αν η εφαρμογή εκτελείται για πρώτη φορά ενώ το δεύτερο για την αποθήκευση του timestamp του τελευταίου update που έγινε.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >

   <CheckBoxPreference android:title="FirstTime"
      android:defaultValue="true"
      android:key="firstRun"
      android:summary="FirstTimeAppRun" />
   <EditTextPreference android:key="lastUpdate" android:title="LastSync"/>

</PreferenceScreen>
```

12.2.5 Cartcheckbox.xml

Με το αρχείο αυτό δηλώνουμε ένα custom checkbox δίνοντας του τα χαρακτηριστικά που εμείς θέλουμε και όχι τα προκαθορισμένα. Με την χρήση ενός selector που περιέχει μέσα items, ορίζουμε για κάθε πιθανό state, την εικόνα την οποία θέλουμε να έχει το checkbox. Έτσι για παράδειγμα άλλη εικόνα εμφανίζεται όταν είναι επιλεγμένο, άλλη την ώρα που γίνεται κλικ σε αυτό και άλλη όταν είναι μη επιλεγμένο. Το ίδιο μπορεί να γίνει και για άλλα controls όπως είναι τα Buttons.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:drawable="@drawable/addtocart"
      android:state_pressed="true"
      android:state_focused="true"
      android:state_selected="true"
      android:state_checkable="true"
      android:state_checked="false"
      android:state_enabled="true"
      android:state_window_focused="true" ></item>

   <item android:drawable="@drawable/cartremove"
      android:state_checked="true" ></item>

   <item android:drawable="@drawable/addtocart"
      android:state_pressed="false"
      android:state_focused="false"
      android:state_selected="false"
      android:state_checkable="false"
      android:state_checked="false"
      android:state_enabled="false"
      android:state_window_focused="false" ></item>

</selector>
```

12.2.6 Textlines.xml

Με το αρχείο αυτό ορίζουμε ένα σχήμα και τις ιδιότητες αυτού όπως χρώμα και πάχος γραμμής γύρω από αυτό, το οποίο περιβάλλει κάποιο άλλο control. Ουσιαστικά δημιουργεί ένα περίγραμμα που μπορούμε να το χρησιμοποιήσουμε όπου επιθυμούμε.

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >
    <item>
      <shape
       android:shape="rectangle">
          <stroke android:width="1dp" android:color="#FF000000" />
          <solid android:color="#FFDDDDDD" />

       </shape>
   </item>

   <item android:top="1dp" android:bottom="1dp">
      <shape
       android:shape="rectangle">
          <stroke android:width="1dp" android:color="#FFDDDDDD" />
          <solid android:color="#00000000" />
       </shape>
   </item>

</layer-list>
```

12.2.7 Gradient_bg.xml

Το εφέ διαβάθμισης χρώματος είναι πολύ διαδεδομένο και ευχάριστο στα σύγχρονα γραφικά περιβάλλοντα των εφαρμογών. Με τον παρακάτω κώδικα ορίζουμε ένα τέτοιο εφέ διαβάθμισης δίνοντας το χρώμα με το οποίο ξεκινά, το χρώμα που έχει στη μέση και το χρώμα με το οποίο τελειώνει η διαβάθμιση, καθώς και την γωνία την οποία ακολουθεί.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
  <gradient
      android:startColor="#f1f1f2"
      android:centerColor="#e7e7e8"
      android:endColor="#cfcfcf"
      android:angle="270" />

</shape>
```

12.3 Αρχείο AndroidManifest.xml

Το αρχείο αυτό υπάρχει σε κάθε εφαρμογή και περιέχει σημαντικές πληροφορίες σχετικά με αυτή. Αρχικά ορίζεται σε αυτό το όνομα πακέτου της εφαρμογής, η έκδοση και το όνομα της

έκδοσης όπως επίσης και μερικά ακόμα χαρακτηριστικά. Για παράδειγμα το που θα εγκαθίσταται η εφαρμογή ως προεπιλογή. Εδώ με την ιδιότητα preferExternal, προτιμάται η εξωτερική μνήμη εάν αυτή υπάρχει. Επίσης στο αρχείο αυτό ορίζονται οι άδειες που θα πρέπει να πάρουμε από τον χρήστη πριν εγκαταστήσει την εφαρμογή, για την εκτέλεση ορισμένων βασικών λειτουργιών του τηλεφώνου που δύναται να γίνουν μέσω της εφαρμογής, αλλά μπορεί ο χρήστης να μην τις επιθυμεί όπως η κλήση ενός αριθμού ή η πρόσβαση στο Internet που πιθανώς θα του επιφέρει κάποιο κόστος ή η εγγραφή δεδομένων στην εξωτερική μνήμη. Τέλος μέσα στο αρχείο αυτό δηλώνονται όλα τα Activities της εφαρμογής και οι ιδιότητες τους όπως ο τίτλος που θα εμφανίζεται για κάθε ένα στο title bar ή ο προσανατολισμός που θα έχει ένα Activity.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.onshop.anzip"
    android:installLocation="preferExternal"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />

    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name" >
        <uses-library android:name="com.google.android.maps" />

        <activity
            android:name=".Splash"
            android:theme="@android:style/Theme.NoTitleBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".OnShop"
            android:configChanges="orientation"
            android:label="Αρχικό Μενού" >
            <intent-filter>
                <action android:name="com.onshop.anzip.ONSHOPACTIVITY" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Products"
            android:label="Προϊόντα" >
        </activity>
        <activity
            android:name=".NewProducts"
            android:label="Νέα Προϊόντα" >
</activity>
```

13. Συμπεράσματα

Στη συγκεκριμένη εργασία επιχειρήθηκε η ανάπτυξη ενός λογισμικού για φορητές συσκευές για την πλοήγηση σε ένα ηλεκτρονικό κατάστημα. Για την υλοποίηση της, απαιτήθηκαν γνώσεις για τον προγραμματισμό στο λειτουργικό Android. Συγκεκριμένα κρίθηκε απαραίτητη η γνώση της αντικειμενοστραφούς γλώσσας προγραμματισμού Java με την οποία έγινε ο προγραμματισμός ολόκληρης της εφαρμογής. Εξίσου ίδιας σημασίας ήταν η γνώση των αρχείων XML που δημιουργήθηκαν για την δημιουργία του γραφικού περιβάλλοντος της εφαρμογής καθώς και της γλώσσας ερωτημάτων βάσεων δεδομένων SQL για την συγγραφή ερωτημάτων επικοινωνίας με την βάση δεδομένων. Απαραίτητη ήταν και η δημιουργία web services που υλοποιούσαν την λήψη δεδομένων από το internet προς την συσκευή. Τέλος προκειμένου να δημιουργηθούν τμήματα το γραφικού περιβάλλοντος, βασική προϋπόθεση ήταν η γνώση δημιουργίας και επεξεργασίας γραφικών.

Όλη η εργασία έγινε με την χρήση του λογισμικού Eclipse στο οποίο είχε εγκατασταθεί το Android Developer Tool και φυσικά με την ύπαρξη του Android SDK. Τα web services δημιουργήθηκαν σε περιβάλλον Microsoft Visual Studio ενώ για την δημιουργία των γραφικών χρησιμοποιήθηκε το λογισμικό Adobe Fireworks.

Ο μόνος περιορισμός που υπήρξε, τέθηκε από την ανάγκη ύπαρξης πολλαπλών συσκευών οι οποίες θα είχαν διαφορετικό μέγεθος οθόνης, αναλύσεις και έκδοση του λογισμικού Android έτσι ώστε να μπορεί να γίνει αποσφαλμάτωση της εφαρμογής όπως επίσης και διορθώσεις στο γραφικό περιβάλλον σε πραγματικές συνθήκες.

Κάτι που εύκολα διαπιστώνεται σχετικά με την εφαρμογή είναι πως υπάρχουν περιθώρια επέκτασης της. Έτσι μπορεί κάλλιστα να προστεθούν επιλογές όπως ο αναγνώστης QR-Codes που μπορεί να υπάρχουν για τα προϊόντα, η επιλογή προβολής των πιο πρόσφατα εμφανισμένων προϊόντων η κατηγοριών, η χρήση πολλαπλών τρόπων πληρωμής, η ενημέρωση του χρήστη για την διαθεσιμότητα κάθε προϊόντος και η προβολή των κοντινότερων καταστημάτων με βάση την τρέχουσα θέση του χρήστη με χρήση GPS.

Τέλος, με την ανάλυση της εφαρμογής, δημιουργήθηκε ένα εγχειρίδιο χρήσης της εφαρμογής για τον απλό χρήστη που εξασφαλίζει σε αυτόν μια πλήρη εικόνα των δυνατοτήτων και των λειτουργιών της καθώς και ένα σημαντικό βοήθημα για τον προγραμματιστή εφαρμογών Android ώστε να κατανοήσει και να αξιοποιήσει τον αναπτυχθέντα κώδικα.

14. Βιβλιογραφία – Αναφορές

Βιβλιογραφία

Marko Gargenta, (2011), Learning Android, Εκδόσεις O'Reilly Media

Matt Egan - Rosie Hattersley, (2011), The Complete Guide to Google Android, Εκδόσεις IDG Communications

Ιστότοποι

http://www.webcitation.org/5wk7sIvVb

http://www.bgr.com/2011/10/21/android-market-surpasses-500000-published-apps/

http://www.wired.com/gadgetlab/2011/12/10-billion-apps-detailed/

http://tech.in.gr/presentations/article/?aid=1231202650

http://techblog.gr/software/adobe-flash-player-not-supported-in-android-jellybean-58441/

http://www.doctorandroid.gr/2012/06/to-adobe-flash-41-jelly-bean.html#.T_FFv5FKxbw

http://developer.android.com/resources/dashboard/platform-versions.html

http://www.myphone.gr/forum/showthread.php?t=306146

http://developer.android.com/guide/topics/connectivity/nfc/index.html

http://kimtag.com/s/nfc_tags

http://tech.in.gr/short-news/?aid=1231116834

http://developer.android.com/guide/basics/what-is-android.html

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

http://www.myphone.gr/forum/showthread.php?p=3645022

http://www.androidpolice.com

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

http://www.sqlite.org/

http://www.vogella.com/articles/AndroidSQLite/article.html

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

http://mobiforge.com/developing/story/using-google-maps-android

http://developer.android.com/guide/topics/location/index.html