

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

**Ανάπτυξη Εφαρμογής Έξυπνης Αγοράς
για λειτουργικό σύστημα Android**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
ΤΟΥ
ΤΣΑΠΡΑΪΛΗ ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων : Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2015

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Ανάπτυξη Εφαρμογής Έξυπνης Αγοράς για λειτουργικό σύστημα Android

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟΥ

ΤΣΑΠΡΑΪΛΗ ΚΩΝΣΤΑΝΤΙΝΟΥ

Επιβλέπων : Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την

(Υπογραφή)

.....

Καθηγητής Ε.Μ.Π.

(Υπογραφή)

.....

Καθηγητής Ε.Μ.Π.

(Υπογραφή)

.....

Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2015

(Υπογραφή)

.....

ΤΣΑΠΡΑΪΛΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Ε.Μ.Π.

© 2015 – All rights reserved

Περίληψη

Η διπλωματική εργασία πραγματεύεται την ανάπτυξη εφαρμογής (mobile app) για λειτουργικό σύστημα Android. Πρόκειται για εφαρμογή έξυπνων αγορών που σκοπό έχει να δώσει στο χρήστη τη δυνατότητα αγορών από τις φορητές του συσκευές.

Η εισαγωγή του πρώτου έξυπνου κινητού το 2007 και του πρώτου tablet το 2010 άλλαξαν δραματικά την αγορά των προσωπικών υπολογιστών, με αποτέλεσμα σήμερα πολύ μεγάλο ποσοστό του πληθυσμού να χρησιμοποιεί τη φορητή του συσκευή σαν πρωτεύον εργαλείο για την πρόσβαση στο διαδίκτυο. Με βάσει αυτά τα στοιχεία είναι απαραίτητο για οποιαδήποτε ηλεκτρονική υπηρεσία να έχει παρουσία στις φορητές συσκευές, ώστε το μέγιστο δυνατό ποσοστό των χρηστών να μπορεί να έχει πρόσβαση σε αυτή.

Για τη συγκεκριμένη διπλωματική έχει εξομοιωθεί η δημιουργία μίας υπηρεσίας έξυπνων αγορών, δηλαδή μια υπηρεσία η οποία θα προσφέρει στο χρήστη τη δυνατότητα να ελέγχει αυτόματα τη χαμηλότερη τιμή για συγκεκριμένο προϊόν ανάμεσα σε διάφορα ηλεκτρονικά καταστήματα, καθώς και την πιθανή ύπαρξη κουπονιών έκπτωσης.

Η διπλωματική αποτελεί συνέχεια της εφαρμογής συναδέλφου, στην οποία προστέθηκε επιπλέον λειτουργικότητα. Συγκεκριμένα προστέθηκε η δυνατότητα σύνδεσης στην εφαρμογή με χρήση λογαριασμό του κοινωνικού δικτύου facebook, δυνατότητα σύνδεσης με προσωρινό λογαριασμό, δυνατότητα δημιουργίας νέου λογαριασμού στην εφαρμογή, καθώς και η λειτουργία λίστας wishlist, δηλαδή μια λίστας από προϊόντα για τα οποία ο χρήστης ενδιαφέρεται. Η εφαρμογή χρησιμοποιεί μία RESTful υπηρεσία για την εκτέλεση των λειτουργιών σύνδεσης, η λειτουργικότητα της οποίας είναι πέρα από το αντικείμενο της διπλωματικής, και για το λόγο αυτό κάνουμε την παραδοχή, ότι υπάρχουν οι συγκεκριμένες κλήσεις API χωρίς να μας ενδιαφέρει η υλοποίησή τους.

Λέξεις Κλειδιά: <<εφαρμογή android, ηλεκτρονικό εμπόριο, έξυπνες αγορές, κοινωνικές αγορές, κοινωνικό δίκτυο facebook, RESTful API>>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

The diploma thesis concerns the development of an android mobile application. It's a smart shopping application.

Since the inception of the first smartphone in 2007 as well as the first tablet device in 2010, the personal computer space has changed drastically to the point where today, the largest portion of the users are using their mobile device as their primary internet access device.

Based on those facts it's imperative for any online service that wishes to include as big a percentage of internet users as possible, to have a mobile application.

For this diploma thesis, a smart shopping android application was created, i.e. an application that offers the user the ability to find the best price on a specific product based not only on the e-shops selling it, but also on whether there is a discount coupon offering savings on the final price.

We follow-up on the work of another student, extending the functionality offered by the application. In particular the subsystems added were, the ability to use the user facebook account to login to our service, the ability to login with a guest account, the ability to signup for a new account with our service as well as a wishlist operation, i.e. a list of products that the user has taken an interest in. The android application is using a RESTful backend service for the login and signup operation, the function of which is beyond the scope of this thesis. On that account dummy API calls were used, based on the premise that an actual service works on the same principles.

Keywords: <<Android app, e-commerce, smart shopping, social shopping, facebook, RESTful API>>

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

1	Εισαγωγή	1
1.1	Ανάπτυξη Εφαρμογής Android	1
1.2	Αντικείμενο διπλωματικής	2
1.2.1	Συνεισφορά	2
1.3	Οργάνωση κειμένου	3
2	Android	4
2.1	Android Internals	5
2.1.1	Dalvik Virtual Machine	6
2.2	Android SDK	7
2.2.1	Android Debug Bridge	8
2.2.2	Fastboot	9
2.3	Native Development Kit	9
3	Android Development Tutorial	11
3.1	Android SDK	11
3.1.1	Εγκατάσταση σε Windows	12
3.2	Eclipse IDE + ADT plugin	14
3.2.1	Δημιουργία εφαρμογής Hello World	15
3.3	Android SDK	22
3.3.1	Android Manifest	22
3.3.2	Activities	24
3.3.3	Layouts	26
3.3.4	Services – Broadcast Receivers – Content Providers	27
3.3.5	Intents	28
3.3.6	Fragments	30

4 Facebook Log in	32
4.1 Αρχιτεκτονική	32
4.2 Περιγραφή λειτουργιών.....	34
4.2.1 Facebook Session Handling.....	34
4.2.2 Android AsyncTask Rest API call.....	37
5 Δημιουργία προσωρινού λογαριασμού χρήστη - Guest Log in	42
5.1 Αρχιτεκτονική	42
5.2 Περιγραφή λειτουργιών.....	43
5.2.1 Installation Class – Κλάση δημιουργίας μοναδικού κωδικού.....	43
5.2.2 Guest Log in Call.....	46
6 Δημιουργία νέου λογαριασμού - Sign up	53
6.1 Αρχιτεκτονική	53
6.2 Περιγραφή διάταξης (Layout).....	54
6.3 Περιγραφή λειτουργιών.....	58
6.3.1 Χειρισμός διεπαφής χρήστη (UI) sign up.....	58
6.3.2 Κλήση sign up (API Call).....	95
7 Λειτουργία Wishlist	64
7.1 Αρχιτεκτονική	64
7.2 Η κλάση SharedPreferences.....	65
7.3 Περιγραφή λειτουργιών.....	66
7.3.1 Το κουμπί “Add to Wishlist”.....	66
7.3.2 Wishlist fragment.....	72
8 Επίλογος	77
8.1 Σύνοψη	77
8.2 Μελλοντικές επεκτάσεις.....	79
9 Βιβλιογραφία	81

1

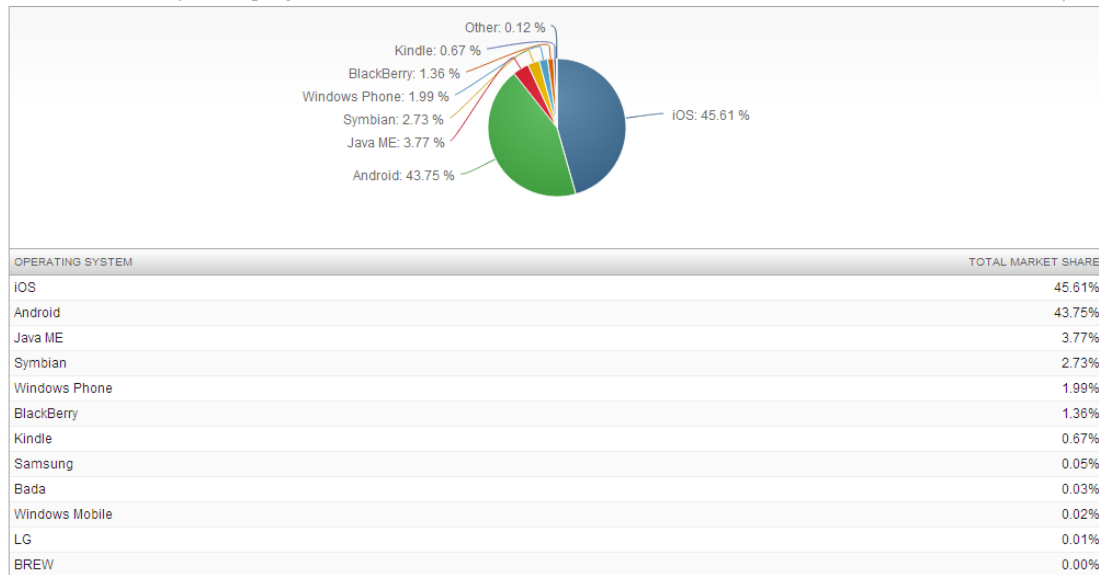
Εισαγωγή

1.1 Ανάπτυξη Εφαρμογής Android

Οι εφαρμογές των έξυπνων κινητών και tablet είναι ένας χώρος ο οποίος έχει γνωρίσει τεράστια ανάπτυξη τα τελευταία χρόνια. Οι χρήστες του διαδικτύου φαίνεται ότι μεταφέρονται μαζικά προς τις κινητές πλατφόρμες. Έτσι η ανάπτυξη εφαρμογών για αυτές τις φορητές πλατφόρμες είναι απαραίτητη για κάθε ενδιαφερόμενο ο οποίος προσπαθεί να προσεγγίσει μεγάλο ποσοστό των χρηστών του διαδικτύου.

Οι δύο βασικές πλατφόρμες φορητών εφαρμογών είναι το λειτουργικό iOS της Apple για συσκευές iPhone και iPad, και το Android της Google που χρησιμοποιείται σε κινητά τηλέφωνα και tablet διαφόρων κατασκευαστών ηλεκτρονικών συσκευών (Samsung, LG κ.α.).

Το λειτουργικό σύστημα Android είναι η δεύτερη δημοφιλέστερη πλατφόρμα την στιγμή της συγγραφής της διπλωματικής εργασίας, με ποσοστό 43.75%, πίσω από το λειτουργικό της Apple με ποσοστό 45.51%^[1]. Βάσει αυτών των στοιχείων η ύπαρξη εφαρμογής για Android, παράλληλα με την εφαρμογή για iOS εξασφαλίζουν την κάλυψη των χρηστών σε ποσοστό κοντά στο 90%.



Εικ. 1. Ποσοστά λειτουργικών Συστημάτων Φορητών Συσκευών

1.2 Αντικείμενο διπλωματικής

Η συγκεκριμένη διπλωματική πραγματεύεται την προσθήκη χαρακτηριστικών σε υπάρχουσα εφαρμογή android.

1.2.1 Συνεισφορά

Η συνεισφορά της συγκεκριμένης διπλωματικής είναι τα εξής χαρακτηριστικά:

- Δημιουργία button για signup – δημιουργία νέου λογαριασμού.
- Δημιουργία button για login με το λογαριασμό Facebook του χρήστη.
- Δημιουργία button για Guest login, για “ανώνυμη” χρήση της εφαρμογής.
- Δημιουργία λειτουργίας wishlist.

1.3 Οργάνωση κειμένου

Η διπλωματική εργασία χωρίζεται στα ακόλουθα κεφάλαια:

Το κεφάλαιο 2 εισάγει το λειτουργικό σύστημα Android, την αρχιτεκτονική του, καθώς και κάποια εργαλεία που βοηθάνε στην ανάπτυξη και την αποσφαλμάτωση εφαρμογών.

Στο κεφάλαιο 3 περιγράφεται ένα tutorial για την ανάπτυξη μια απλής εφαρμογής.

Στο κεφάλαιο 4 ξεκινά η περιγραφή του πρώτου υποσυστήματος που υλοποιήθηκε, το οποίο είναι το facebook log in, ενώ στο κεφάλαιο 5 γίνεται αναφορά στο υποσύστημα Guest log in.

Ακολούθως στο κεφάλαιο 6 παρουσιάζεται το υποσύστημα sign up για τη δημιουργία νέου χρήστη.

Το τελευταίο υποσύστημα που υλοποιήθηκε είναι η λειτουργία wishlist, που παρουσιάζεται στο κεφάλαιο 7.

Εν κατακλείδι δίνεται ο επίλογος στο κεφάλαιο 8 και ολοκληρώνεται η διπλωματική με την βιβλιογραφία που χρησιμοποιήθηκε στο κεφάλαιο 9.

2

Android

Το Android είναι ένα λειτουργικό σύστημα βασισμένο στον πυρήνα του Linux. Έχει σχεδιαστεί κυρίως για φορητές συσκευές με οθόνη αφής όπως smartphones και tablets, ενώ μπορεί να χρησιμοποιηθεί και σε τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto), και ρολόγια (Android Wear).

Το λειτουργικό χρησιμοποιεί την αφή μέσω της οθόνης σαν είσοδο για τις επιλογές του χρήστη. Οι κινήσεις πάνω στην οθόνη αντιγράφουν αυτές της καθημερινής ζωής, όπως κύληση (swiping), χτύπημα (tapping), “τσιμπήματα” (pinching), και “ανάποδα τσιμπήματα” (reverse pinching) για τη διαχείριση εικονικών αντικειμένων στην οθόνη.

Αν και το λειτουργικό είναι σχεδιασμένο πρωτίστως για είσοδο με αφή (touch input), έχει χρησιμοποιηθεί και σε παιχνιδομηχανές, ψηφιακές κάμερες και άλλες ηλεκτρονικές συσκευές.

2.1 Android Internals

Το Android αποτελείται από αρκετά αλληλοεξαρτόμενα κομμάτια:

- Στην καρδιά του Android βρίσκεται ο πυρήνας του Linux που καθιστά δυνατή την επικοινωνία με το υλικό της συσκευής,

αναλαμβάνει τη διαχείριση μνήμης, τον έλεγχο των διεργασιών, τα οποία είναι όλα βελτιστοποιημένα για φορητές και ενσωματωμένες συσκευές.

- τα Compatibility Definition Document (CDD) και Compatibility Test Suit (CTS) που περιγράφουν τις απαραίτητες δυνατότητες που χρειάζονται για να υποστηρίξεται το software stack του Android.
- Βιβλιοθήκες ανοιχτού κώδικα για την ανάπτυξη εφαρμογών, περιλαμβανομένων των SQLite, WebKit, OpenGL και ένα διαχειριστή πολυμέσων.
- Το run-time κομμάτι που χρησιμοποιείται για να εκτελεί τις εφαρμογές Android, αποτελούμενο από την Dalvik Virtual Machine και της βασικές βιβλιοθήκες που προσδίδουν λειτουργικότητα σχεδιασμένη για το Android.
- Το Application Framework που εκθέτει τις υπηρεσίες συστήματος στο στρώμα των εφαρμογών, περιλαμβανομένων των υπηρεσιών διαχείρισης παραθύρων, διαχείρισης τοποθεσίας, βάσεων δεδομένων, τηλεφώνου, και αισθητήρων.
- Το framework διεπαφής χρήστη (User Interface) που χρησιμοποιείται για να εκκινεί και να φιλοξενεί εφαρμογές.
- Ένα σύνολο βασικών προεγκατεστημένων εφαρμογών.
- Το Kit ανάπτυξης λογισμικού (SDK) που χρησιμοποιείται για την ανάπτυξη εφαρμογών, περιλαμβανομένων και των σχετικών εργαλείων, plug-ins, και documentation.



Εικ. 2 To Android Software Stack

2.1.1 Dalvic Virtual Machine

Dalvik Virtual Machine (DVM) ή απλά Dalvik όπως αναφέρθηκε, είναι η εικονική μηχανή διεργασιών (process virtual machine) του Android. Είναι το λογισμικό που τρέχει τις εφαρμογές του Android.

Τα προγράμματα συνήθως είναι γραμμένα σε Java και γίνονται compile σε bytecode για την Java Virtual Machine, ο οποίος κώδικας έπειτα μεταφράζεται σε Dalvik bytecode και αποθηκεύεται σε αρχεία .dex (Dalvik Executable) και .odex (Optimized Dalvik Executable). Επίσης υπάρχει το compact Dalvik Executable φορμά το οποίο είναι σχεδιασμένο για συστήματα με περιορισμένη ταχύτητα επεξεργαστή και μνήμη.

Επίσης να αναφερθεί μία νέα εικονική μηχανή (Virtual Machine) που αναπτύσσεται από την Google, η ART (Android Runtime), η οποία υπάρχει στο Android 4.4 σε beta έκδοση και μπορεί να ενεργοποιηθεί από τον χρήστη. Από την έκδοση Android L η οποία κυκλοφόρησε στα

τέλη του 2014, η ART θα αντικατέστησει πλήρως την Dalvik σαν την προεπιλεγμένη VM του Android.

Επειδή όμως το ποσοστό των συσκευών που τρέχουν το Android L είναι (προς το παρόν) πολύ χαμηλό δεν θα εξετάσουμε περαιτέρω τη νέα αυτή μηχανή.

2.2 Android SDK

Το kit ανάπτυξης λογισμικού Android (SDK) περιλαμβάνει ένα σετ εργαλείων ανάπτυξης. Αυτά είναι ο αποσφαλματωτής (debugger), διάφορες βιβλιοθήκες, έναν εξομοιωτή φορητής συσκευής, τεκμηρίωση (documentation), παραδείγματα κώδικα, και παραδείγματα χρήσης (tutorials).

Κατα τη στιγμή της συγγραφής, οι πλατφόρμες στις οποίες υποστηρίζεται η ανάπτυξη για Android, είναι υπολογιστές που τρέχουν Linux (όλες τις κύριες εκδόσεις του), Mac OS X 10.5.8 ή μεταγενέστερη έκδοση, Windows XP ή μεταγενέστερη έκδοση. Επίσης υπάρχει δυνατότητα ανάπτυξης λογισμικού Android σε συσκευές που τρέχουν android, αν και δεν προτίνεται λόγω των περιορισμένων πόρων που έχουν τα συστήματα αυτά.

Το περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) που υποστηρίζεται επίσημα είναι το Eclipse, με το plugin Android Development Tools (ADT). Το περιβάλλον IntelliJ IDEA IDE, επίσης υποστηρίζει την ανάπτυξη εφαρμογών χωρίς τη χρήση κάποιου Plugin, ενώ το περιβάλλον NetBeans IDE υποστηρίζει ανάπτυξη λογισμικού Android μέσω Plugin. Επιπλέον υπάρχει η επιλογή, να γραφεί κώδικας Java και XML σε κάποιο επεξεργαστή κειμένου (text editor) και με τη χρήση εργαλείων μέσω της γραμμής εντολών (τα Java Development Kit και Apache Ant είναι απαραίτητα) να δημιουργηθούν και να αποσφαλματωθούν εφαρμογές Android, καθώς και να γίνεται έλεγχος συνδεδεμένων συσκευών (πχ η ενεργοποίηση επανακίνησης του κινητού, η απομακρυσμένη εγκατάσταση πακέτων λογισμικού κλπ).

Οι βελτιώσεις στο SDK του Android είναι άρρηκτα συνδεδεμένες με το ίδιο το λειτουργικό. Το SDK υποστηρίζει και παλιότερες εκδόσεις του Android για τις περιπτώσεις που οι προγραμματιστές επιθυμούν οι εφαρμογές τους να τρέχουν σε αυτές. Τα εργαλεία ανάπτυξης είναι

τμήματα που “κατεβαίνουν” από το διαδίκτυο, οπότε αν κάποιος έχει την τελευταία έκδοση των εργαλείων μπορεί να κατεβάσει και παλιότερες εκδόσεις για έλεγχο συμβατότητας.

Οι εφαρμογές Android “πακετάρονται” σε αρχεία τύπου .apk, και αποθηκεύονται στο φάκελο /data/app στο λειτουργικό (η πρόσβαση στο φάκελο είναι δυνατή μόνο σε χρήστη root για λόγους ασφάλειας). Τα πακέτα .apk περιέχουν τα αρχεία .dex, αρχεία πόρων της εφαρμογής, όπως εικόνες, βίντεο, αρχεία ήχου κ.α..

2.2.1 Android Debug Bridge

Το Android Debug Bridge είναι ένα πολυχρηστικό εργαλείο γραμμής εντολών, που χρησιμοποιείται για την επικοινωνία με εξομοιωτή Android ή με μία συνδεδεμένη συσκευή.

Είναι ένα από τα πιο βασικά προγράμματα κατά την ανάπτυξη λογισμικού android καθώς επιτρέπει την αποσφαλμάτωση σε πραγματικό χρόνο των εφαρμογών.

Είναι πρόγραμμα client-server που αποτελείται από τρία βασικά τμήματα:

- Το client που τρέχει στο μηχάνημα ανάπτυξης. Το client μπορεί να κληθεί από γραμμή εντολών δίνοντας μία εντολή του adb. Άλλα εργαλεία όπως το ADT plugin και το Davik Debug Monitor Service (DDMS) μπορούν να δημιουργήσουν επίσης adb clients.
- Το server το οποίο τρέχει σαν διεργασία στο παρασκήνιο στο μηχάνημα ανάπτυξης. Το server διαχειρίζεται την επικοινωνία ανάμεσα στο client και στο adb daemon.
- Το adb daemon το οποίο τρέχει σαν διεργασία στο παρασκήνιο είτε σε εξομοιωτή είτε σε συσκευή.

Το φορμά που ακολουθούν οι εντολές του ADB είναι συνήθως το ακόλουθο:

```
adb [-dl-el-s <serialNumber>] <command>
```

2.2.2 Fastboot

Το fastboot είναι ένα διαγνωστικό πρωτόκολλο που περιλαμβάνεται στο πακέτο SDK και χρησιμοποιείται κυρίως για να

τροποποιεί το flash σύστημα αρχείων μέσω σύνδεσης USB από το μηχάνημα ανάπτυξης. Είναι απαραίτητο η συσκευή να ξεκινήσει σε boot loader mode ή Second Program Loader κατά το οποίο μόνο τα πιο βασικά τμήματα του υλικού της συσκευής χρησιμοποιείται. Αφού ενεργοποιηθεί το πρωτόκολλο στη συσκευή, θα δέχεται μόνο συγκεκριμένο σετ εντολών που θα στέλνονται μέσω USB χρησιμοποιώντας την γραμμή εντολών. Κάποιες από τις πιο βασικές εντολές είναι οι ακόλουθες:

- flash – Ξαναγράφει ένα partition με ένα binary image (δηλαδή αρχεία του λειτουργικού σε διαδίκιο κώδικα έτοιμα να τρέξουν στη συσκευή) που είναι αποθηκευμένο στο μηχάνημα ανάπτυξης.
- erase – Διαγράφει ένα συγκεκριμένο partition.
- reboot – Επανακινεί τη συσκευή είτε στο κύριο λειτουργικό σύστημα, στο partition επαναφοράς συστήματος είτε πάλι στο bootloader.
- devices – Εμφανίζει λίστα με όλες τις συσκευές (με serial number) συνδεδεμένες στο μηχάνημα ανάπτυξης.
- format – Κάνει format σε ένα συγκεκριμένο partition.

2.3 Native Development Kit

Το Android υποστηρίζει τη χρήση βιβλιοθηκών γραμμένες σε c, c++ και άλλες γλώσσες, αφού γίνουν compile σε ARM, MIPS, ή X86 native κώδικα, μέσω του Android Native Development Kit. Οι native κλάσεις μπορούν να κληθούν από κώδικα Java που τρέχει στην Dalvik VM χρησιμοποιώντας την κλήση System.loadLibrary, που είναι κομμάτι των standard Android κλάσεων.

Ολοκληρωμένες εφαρμογές μπορούν να γίνουν compile και να εγκατασταθούν χρησιμοποιώντας τα κλασσικά εργαλεία ανάπτυξης. Παρόλα αυτά το Android Documentation αναφέρει ότι το NDK δεν θα πρέπει να χρησιμοποιείται μόνο επειδή ο προγραμματιστής προτιμά να γράφει κώδικα σε C ή C++, καθώς το NDK αυξάνει την πολυπλοκότητα και οι περισσότερες εφαρμογές δεν θα εμφανίσουν βελτίωση της απόδοσης από τη χρήση του.

Ο ADB debugger προσφέρει γραμμή εντολών με δικαιώματα root, στον εξομοιωτή Android, μέσω του οποίου ο προγραμματιστής μπορεί να τρέξει native ARM, MIPS ή X86 κώδικα. Ο κώδικας μπορεί να γίνει compile με GCC ή Intel C++ Compiler σε κλασσικό υπολογιστικό σύστημα. Η χρήση native κώδικα γίνεται ακόμα πιο περίπλοκη, λόγω του ότι το Android χρησιμοποιεί non-standard βιβλιοθηκης (της libc, γνωστή και ως Bionic).

Τελος αντίθετα με την ανάπτυξη σε Java χρησιμοποιώντας το Eclipse IDE, το NDK είναι βασισμένο σε εργαλεία γραμμής εντολών, και απαιτείται η χειροκίνητη κλήση τους για την δημιουργία και αποσφαλμάτωση των εφαρμογών.

3

Eclipse Tutorial

Σε αυτό το κεφάλαιο θα γίνει μία σύντομη εισαγωγή στην ανάπτυξη εφαρμογών για Android με το περιβάλλον Eclipse.

Θα υλοποιηθεί μια απλή εφαρμογή η οποία είναι η βάση γύρω από την οποία μπορεί να χτιστεί οποιαδήποτε άλλη εφαρμογή.

3.1 Android SDK

Το Android SDK είναι το σύνολο των εργαλείων τα οποία χρησιμοποιούνται στην ανάπτυξη εφαρμογών. Διατίθεται σαν πακέτο από την επίσημη σελίδα του Android (<http://developer.android.com/sdk/index.html>), για κατέβασμα, για οποιονδήποτε προγραμματιστή το επιθυμεί. Οι υποστηριζόμενες

πλατφόρμες είναι οι τρεις μεγάλες, Windows, Linux και OS X για υπολογιστές Macintosh. Το πακέτο περιέχει τα ακόλουθα εργαλεία:

- το Eclipse IDE με το plugin ADT
- τα εργαλεία του Android SDK
- τα εργαλεία του λειτουργικού Android
- Μία έκδοση του λειτουργικού Android
- Μία έκδοση του Android σε μορφή “εικόνας συστήματος” (system image) για τον εξομοιωτή

3.1.1 Εγκατάσταση σε Windows

Για να εγκαταστήσουμε το πακέτο σε λειτουργικό Windows, πηγαίνουμε στην διεύθυνση <http://developer.android.com/sdk/index.html> και πατάμε στο κουμπί “Download Eclipse ADT” όπως φαίνεται στην εικόνα 3.1. Ο browser μας στέλνει σε μία δεύτερη σελίδα όπου παρέχονται οι όροι χρήσης του πακέτου, καθώς και η επιλογή αν θέλουμε να κατεβάσουμε το πακέτο για 32-bit ή 64-bit σύστημα. Το πακέτο κατεβαίνει σε ένα εκτελέσιμο αρχείο (.exe) και αφού κάνουμε την εγκατάσταση είμαστε σχεδόν έτοιμοι να ξεκινήσουμε την ανάπτυξη.

The screenshot shows the Android Developers website. At the top, there is a navigation bar with 'Developers' (with a dropdown arrow), 'Design', 'Develop' (highlighted in orange), and 'Distribute'. Below this is a secondary navigation bar with 'Training', 'API Guides', 'Reference', 'Tools' (highlighted in orange), 'Google Services', and 'Samples'. On the left side, there is a sidebar menu with 'Download' selected, and sub-items like 'Installing the SDK', 'Adding SDK Packages', 'Android Studio', 'Workflow', 'Support Library', 'Tools Help', 'Revisions', 'NDK', and 'ADK'. The main content area is titled 'Get the Android SDK'. It contains a large blue button that says 'Download Eclipse ADT with the Android SDK for Windows'. Below this, there is a list of items included in the bundle: Eclipse + ADT plugin, Android SDK Tools, Android Platform-tools, a version of the Android platform, and a version of the Android system image for the emulator. There is also a section for 'Get Android Studio Beta' which describes it as a new IDE powered by IntelliJ. At the bottom of the page, there is a footer with 'Creative Commons Attribution 2.5' license information and links for 'out Android | Legal | Support'.

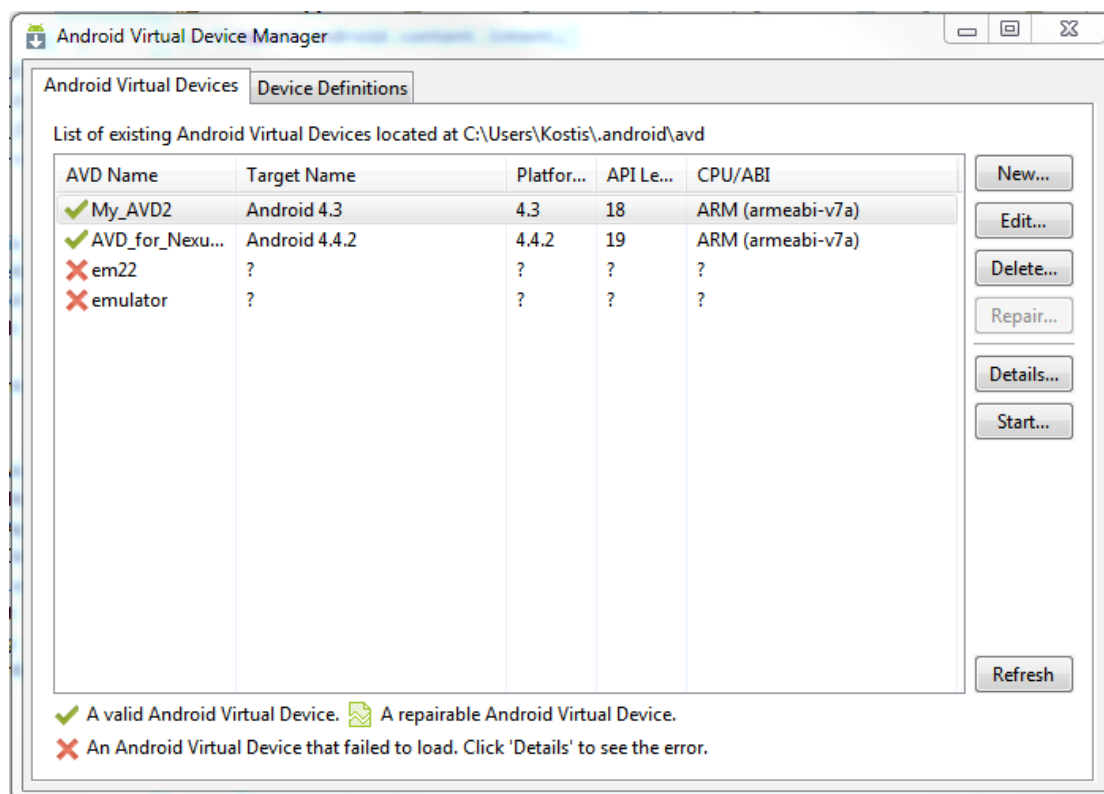
Εικ. 3.1 Η επίσημη σελίδα του Android για να κατεβάσουμε το πακέτο.

Το επόμενο βήμα που πρέπει να κάνουμε πριν ξεκινήσουμε είναι να χρησιμοποιήσουμε τον Android SDK Manager για να ελέγξουμε αν έχουμε τα πιο πρόσφατα εργαλεία. Ο Manager μπορεί να κληθεί μέσω του Eclipse πηγαίνοντας στην επιλογή Window και έπειτα πατώντας “Android SDK Manager”.

Το εργαλείο αυτό χρησιμοποιείται για την διαχείριση των πολλαπλών εκδόσεων του SDK. Κάθε φορά που κυκλοφορεί μία νέα έκδοση του Android είναι καλή πρακτική να ανανεώνονται τα εργαλεία, για να ελέγχουμε την συμβατότητα της εφαρμογής μας με την νέα έκδοση.

Τέλος, ένα ακόμα εργαλείο που θα χρειαστούμε είναι ο διαχειριστής εικονικών συσκευών, Android Virtual Device Manager. Το

πρόγραμμα αυτό χρησιμοποιείται για να δημιουργούμε και να επεξεργαζόμαστε εικονικές συσκευές, τις οποίες χρησιμοποιούμε για να τρέξουν οι εφαρμογές μας. Ο διαχειριστής μπορεί να ενεργοποιηθεί μέσα από το Eclipse πατώντας πάλι στην επιλογή Window και έπειτα Android Virtual Device Manager.



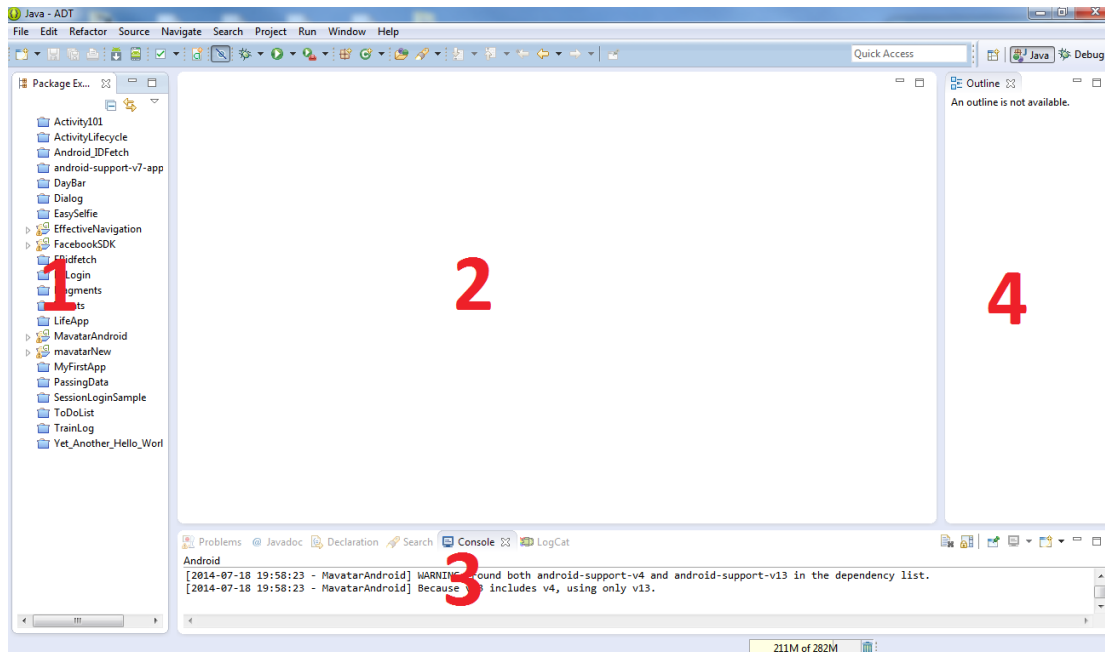
Εικ. 3.2 Ο AVD Manager

Ανοίγοντάς το, το AVD Manager μας δίνει μία λίστα με υπάρχουσες συσκευές, καθώς και επιλογές για δημιουργία, τροποποίηση και διαγραφή συσκευών.

Πατώντας την επιλογή “New...” το πρόγραμμα ανοίγει ένα παράθυρο όπου μπορούμε να δημιουργήσουμε μία νέα συσκευή, είτε από κάποιες προκαθορισμένες, είτε παραμετροποιώντας την συσκευή στις ανάγκες μας.

3.2 Eclipse IDE + ADT plugin

Ανοίγοντας το περιβάλλον Eclipse βλέπουμε 4 μεγάλες περιοχές.



Εικ. 3.3 Το Eclipse IDE

1 Η πρώτη περιοχή (1) στα αριστερά του προγράμματος είναι ο “Εξερευνητής Πακέτων” (Package Explorer) από τον οποίο μπορούμε να έχουμε πρόσβαση στα project στα οποία δουλεύουμε, καθώς και τα αρχεία κώδικα (source files) και τα αρχεία πόρων (resource files).

Στο κέντρο βρίσκεται η βασική περιοχή (2) η οποία χρησιμοποιείται για την συγγραφή και την τροποποίηση των αρχείων. Επιλέγοντας ένα αρχείο από τον εξερευνητή αριστερά αυτό ανοίγει εδώ, για επεξεργασία.

Η περιοχή κάτω (3) μας δίνει πρόσβαση στα αποτελέσματα των εργαλείων που χρησιμοποιούμε, όπως η κονσόλα, ο debugger, τα logs, τα javadocs κλπ.

Η τελευταία περιοχή (4) δεξιά μας δίνει πρόσβαση σε πληροφορίες και εργαλεία, που αντιστοιχούν στο τρέχον αρχείο που είναι ανοιχτό κάθε φορά στην βασική περιοχή (2) του Eclipse.

Τέλος να αναφέρουμε ότι οποιαδήποτε περιοχή μας ενδιαφέρει μπορεί πολύ εύκολα να καταλάβει το σύνολο της οθόνης για διευκόλυνση, κάνοντας διπλό κλικ στο όνομα της αντίστοιχης καρτέλας στην πάνω πλευρά της περιοχής αυτής.

3.2.1 Δημιουργία εφαρμογής Hello World

Είναι γνωστό στους προγραμματιστές ότι κάθε φορά που κάποιος ξεκινά να μάθει μία καινούρια γλώσσα προγραμματισμού, δημιουργεί το Hello World πρόγραμμα, δηλαδή την πιο απλή εφαρμογή που μπορεί να δημιουργηθεί, που συνήθως δεν έχει καμία αλληλεπίδραση με τον χρήστη, απλά εκτυπώνει ένα μήνυμα “Hello World”.

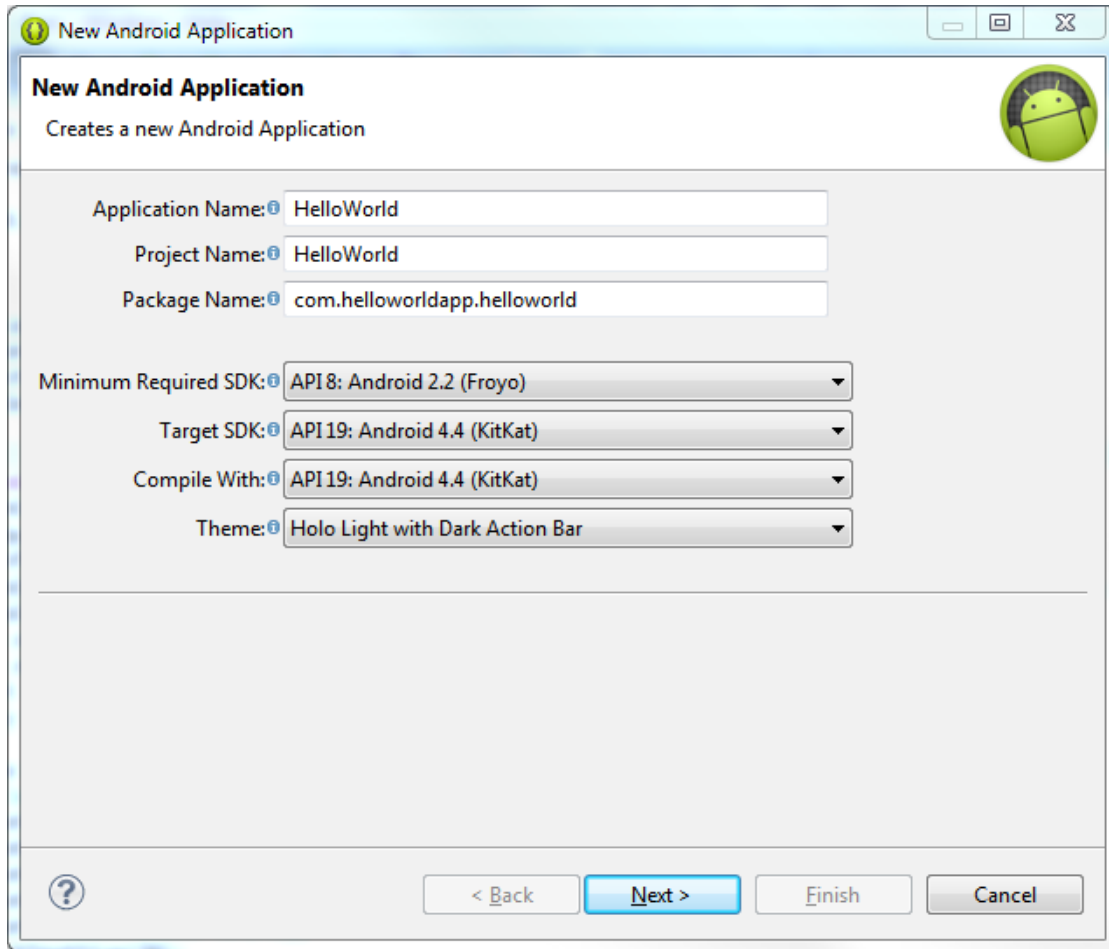
Μπορεί να φαίνεται πολύ απλή η εφαρμογή, και ίσως άχρηστη, αλλά είναι ένα βασικό βήμα κάθε φορά που κάποιος προσπαθεί να μάθει κάτι καινούριο καθώς δείχνει δύο βασικά πράγματα. Καταρχήν ότι έχει εγκαταστήσει σωστά όλα τα απαραίτητα εργαλεία που χρειάζεται, και κατα δεύτερον ότι έκανε σωστά την διαδικασία που απαιτείται για να δημιουργηθεί το εκτελέσιμο αρχείο από τον κώδικα.

Στο Android SDK και συγκεκριμένα στο Eclipse, αυτό γίνεται πολύ εύκολα. Αρχικά πηγαίνουμε στην επιλογή File, έπειτα New και επιλέγουμε από τη λίστα το Android Application Project. Το Eclipse μας δίνει ένα παράθυρο διαλόγου, στο οποίο θα βάλουμε τα βασικά στοιχεία που θέλουμε να έχει το καινούριο μας project.

Τα πρώτα στοιχεία που καλούμαστε να βάλουμε είναι το Application Name, το Project Name και το Package Name. Το Application Name είναι το όνομα της εφαρμογής, όπως θα φαίνεται στο κινητό ή την ταμπλέτα στην οποία θα τρέχει, καθώς και το όνομα το οποίο θα φαίνεται στο google play store αν επιθυμούμε να το ανεβάσουμε.

Το Project Name είναι το όνομα του project στο Eclipse. Αυτό είναι μόνο για το Eclipse και δεν θα φαίνεται κάπου αλλού, θα πρέπει όμως να είναι μοναδικό στο Eclipse.

Τέλος το Package Name είναι το όνομα του πακέτου στο οποίο θα ανήκει ο κώδικας ακολουθώντας τις συμβάσεις της Java. Μια καλή τεχνική που προτίνεται είναι το ονομα του πακέτου να είναι το ανεστραμμένο domain name της εφαρμογής. Για παράδειγμα αν η εφαρμογή μας έχει όνομα HelloWorld και το site είναι helloworldapp.com τότε θα επιλέξουμε το com.helloworldapp.helloworld σαν το όνομα πακέτου.

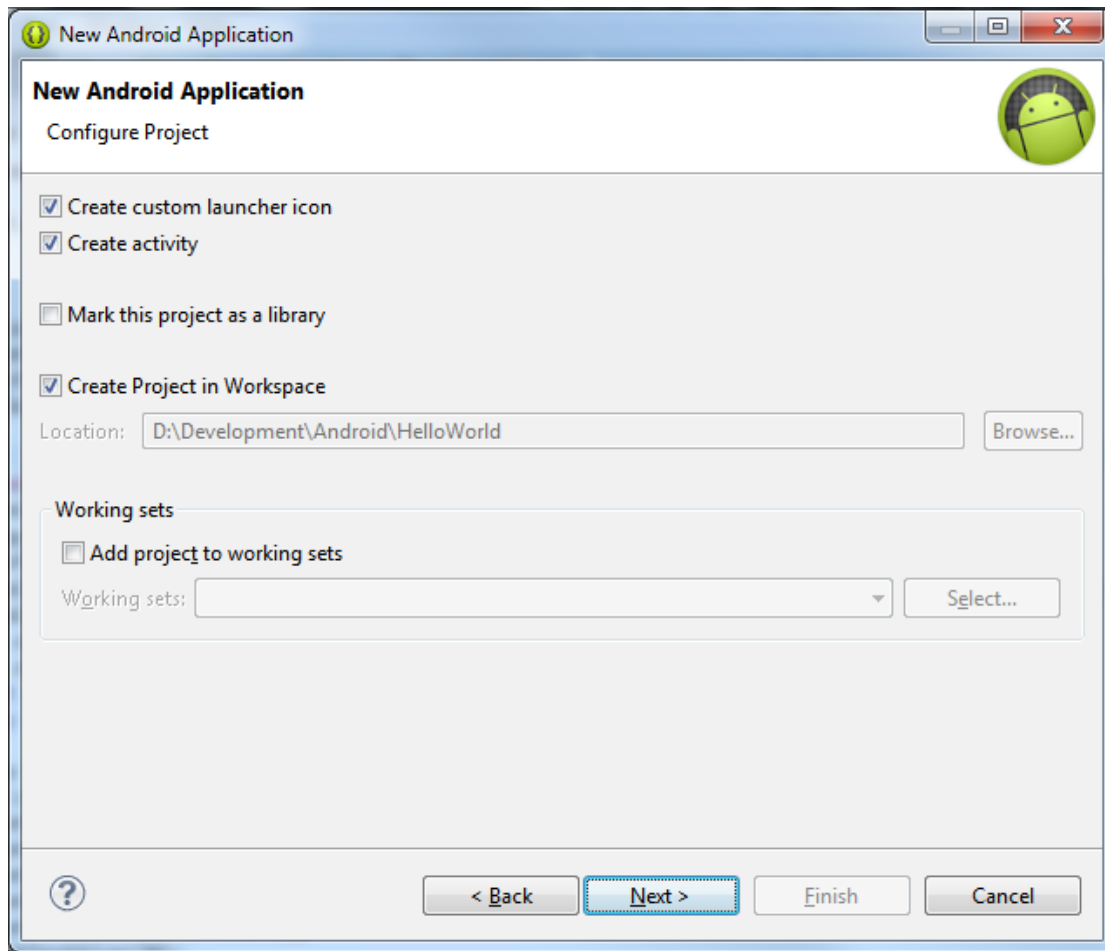


Εικ. 3.4 Δημιουργία νέου Project, αρχικές επιλογές

Τέλος στις επόμενες επιλογές που μας δίνει, βάζουμε Minimum Required SDK που είναι η μικρότερη έκδοση του Android που υποστηρίζει η εφαρμογή μας. Target SDK και Compile With βάζουμε την έκδοση την οποία χρησιμοποιήσαμε για να την ανάπτυξη, και όπως αναφέραμε, είναι καλή πρακτική να βάζουμε την τελευταία έκδοση του Android, έτσι ώστε να ξέρουμε ότι η εφαρμογή μας θα τρέχει σωστά σε κάθε έκδοση, αφού το Android είναι συμβατό ως προς τις προηγούμενες εκδόσεις, αλλά όχι απαραίτητα ως προς τις επόμενες.

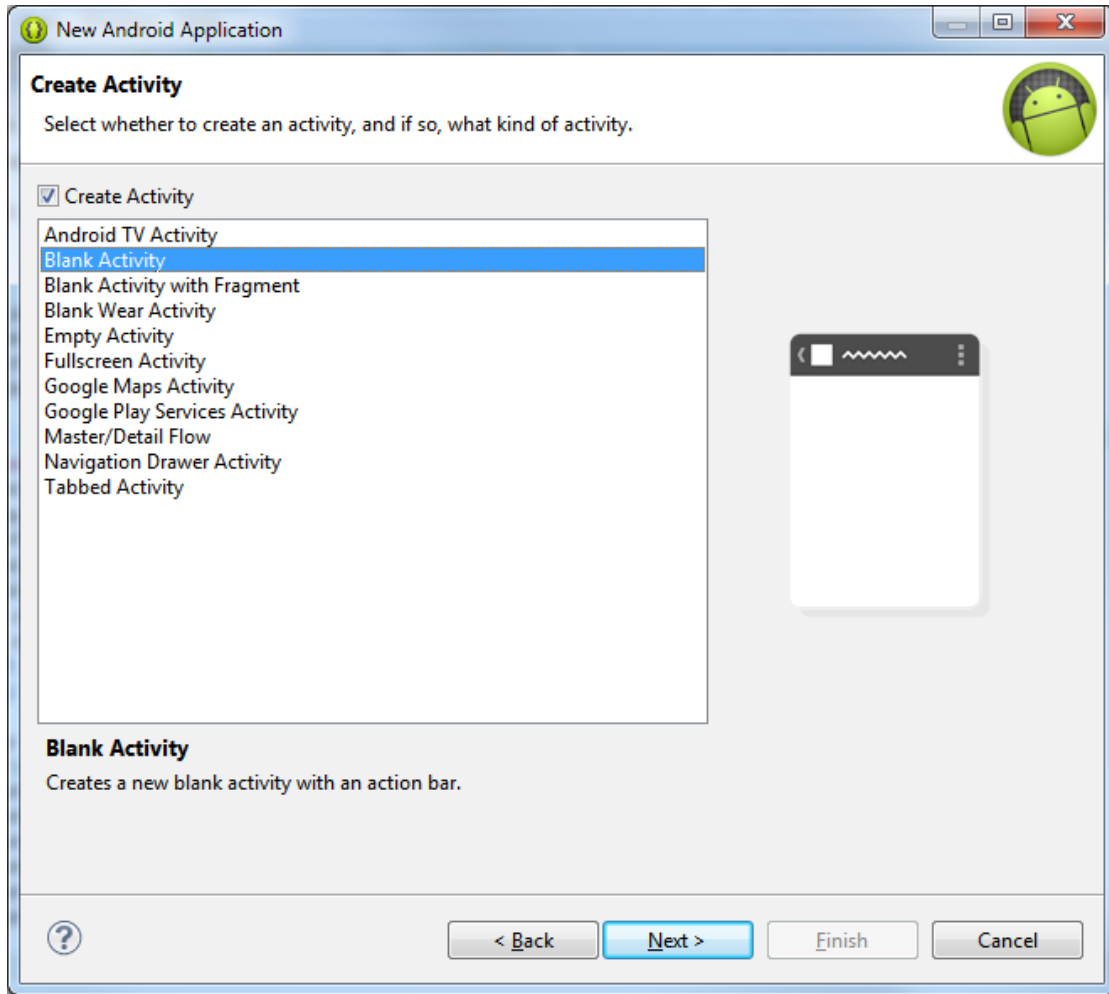
Η τελευταία επιλογή είναι για το θέμα το οποίο θα έχει η εφαρμογή μας, δηλαδή ένα σύνολο επιλογών που αφορούν την εμφάνιση.

Αφού κάνουμε τις επιλογές που θέλουμε πατάμε next.



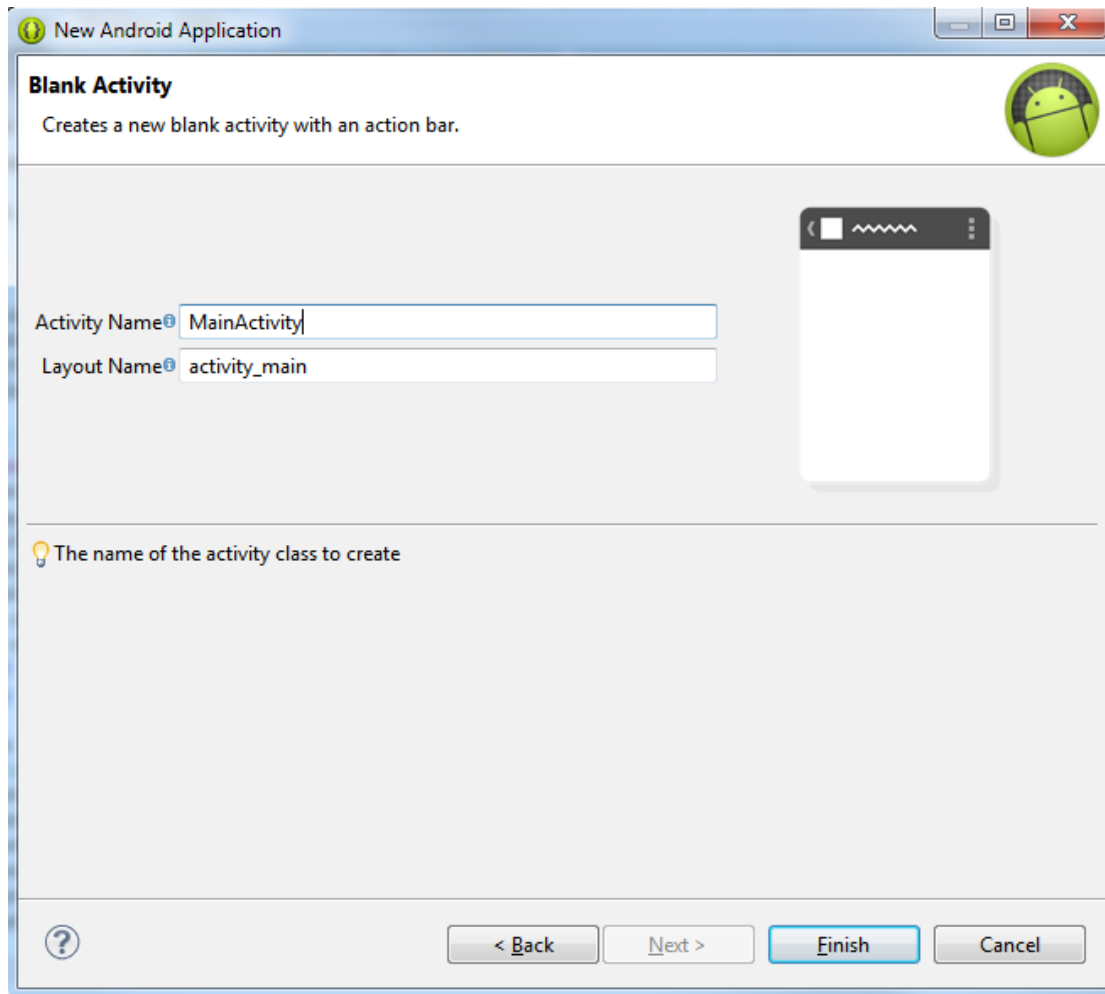
Εικ. 3.5 Δημιουργία νέου project, βήμα 2ο

Στην επόμενη οθόνη που εμφανίζεται η βασική επιλογή είναι το **Create Activity**, καθώς τα activities είναι από τα πιο βασικά κομμάτια του Android, όπως θα δούμε παρακάτω, οπότε θα πρέπει να είναι επιλεγμένο αυτό. Επίσης αποεπιλέγουμε το **Create custom launcher icon**, καθώς για την βασική μας εφαρμογή δεν είναι απαραίτητο κάτι τέτοιο και πατάμε **next**.



Εικ. 3.6 Δημιουργία νέου project, βήμα 3ο

Στην τρίτη οθόνη ο οδηγός μας ζητά να επιλέξουμε τι είδος activity θα δημιουργήσει. Όπως βλέπουμε από την έκδοση 4.4 και έπειτα υποστηρίζεται η ανάπτυξη εφαρμογών για έξυπνα ρολόγια, για έξυπνες τηλεοράσεις, και το Eclipse μας βοηθάει σε αυτό. Προς το παρόν επιλέγουμε Blank Activity για την εφαρμογής μας και πατάμε next.



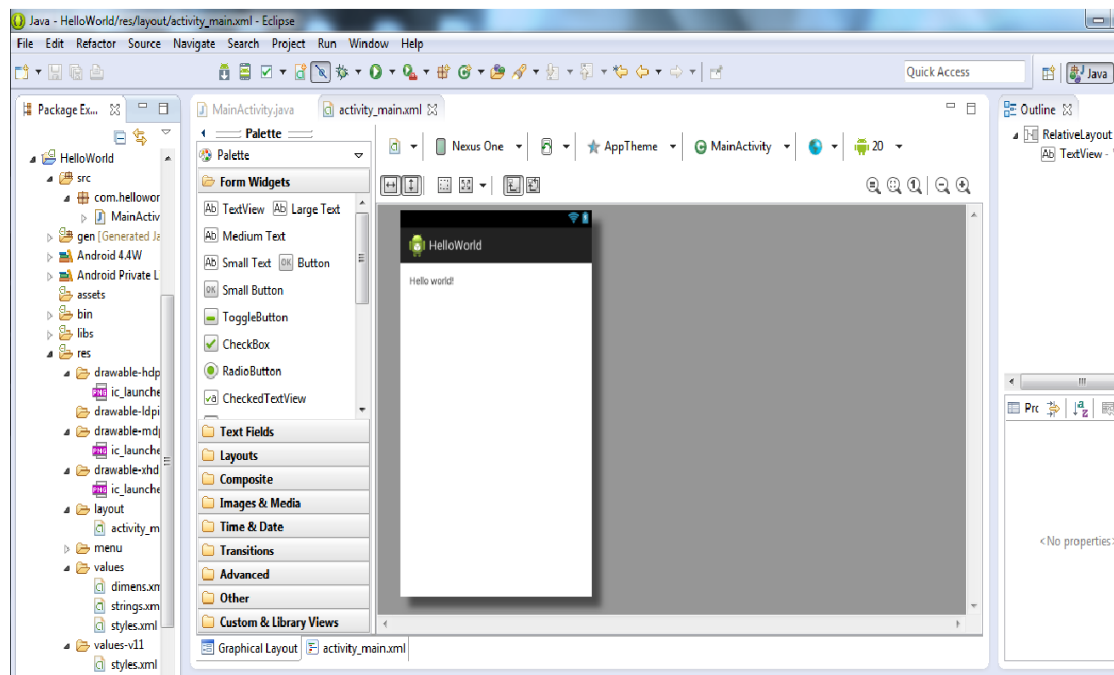
Εικ. 3.7 Δημιουργία νέου project, βήμα 4ο

Στο τελευταίο βήμα μας ζητείται να δώσουμε όνομα στην activity και στο αρχείο layout. Προς το παρόν τα αφήνουμε με τα ονόματα που μας προτείνει το Eclipse και πατάμε finish. Μετά από λίγα δευτερόλεπτα, το Eclipse θα δημιουργήσει όλα τα απαραίτητα αρχεία και φακέλους που απαιτούνται και θα ανοίξει μια προεπισκόπηση της οθόνης της εφαρμογής μας κατά προσέγγιση.

Παρατηρώντας το Eclipse βλέπουμε ότι στην περιοχή αριστερά έχει δημιουργηθεί, όπως αναφέραμε, ο σκελετός των φακέλλων της εφαρμογής. Όλοι οι φάκελλοι και οι υποφάκελοι δεν είναι απαραίτητο να περιέχουν αρχεία, αλλά δημιουργούνται για να μας βοηθήσει το Eclipse να έχουμε τα κατά σύμβαση ονόματα.

Οι πιο βασικοί φάκελοι είναι οι ακόλουθοι με τις λειτουργίες τους:

- /src : Ο φάκελος που περιέχει όλα τα αρχεία κώδικα. Αυτά οργανώνονται περαιτέρω σε υποφακέλους αναλόγως με το πακέτο στο οποίο ανήκουν.
- /res : Ο φάκελος που περιέχει τα αρχεία που είναι σχετικά με την εμφάνιση της εφαρμογής. Θα πρέπει να αναφέρουμε κάποιους βασικούς υποφακέλους.
 - /layout : Εδώ βρίσκονται τα αρχεία που περιγράφουν ακριβώς την εμφάνιση της κάθε οθόνης. Τα αρχεία είναι .xml και περιγράφουν γραφικά στοιχεία καθώς και τις ιδιότητες αυτών τα οποία συνθέτουν την τελική εικόνα της εφαρμογής.
 - /values : Τα αρχεία αυτά περιέχουν σταθερές τιμές. Το android προωθεί τον καθορισμό οποιασδήποτε τιμής (κείμενο, χρώμα, διαστάσεις) πριν χρησιμοποιηθούν στον κώδικα.

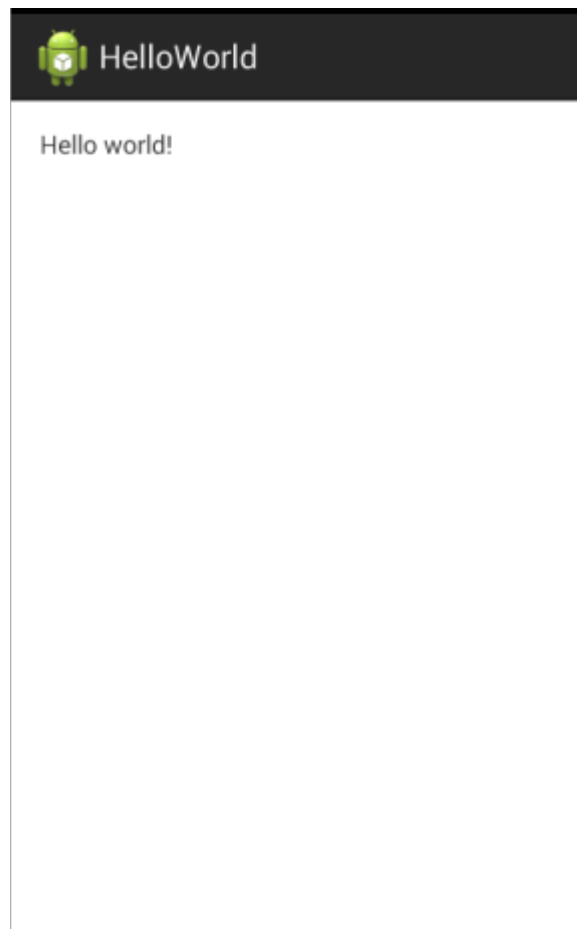


Εικ. 3.8 Η προεπισκόπηση της εφαρμογής μας

Θα αναφερθούμε στην επόμενη ενότητα με λεπτομέρειες για όλα τα προηγούμενα και πως επηρεάζουν την εφαρμογή.

Για να τρέξουμε το app, κάνουμε δεξί κλικ στον βασικό φάκελο του project πατάμε Run As και μετά Android Application. Το Eclipse θα ανοίξει τον emulator με το Android Virtual Device που έχουμε

προεπιλεγμένο και αφού ξεκινήσει θα τρέξει απευθείας η εφαρμογή, δίνοντας μας την ακόλουθη εικόνα:



Εικ. 3.9 To User Interface του Hello World App.

3.3 Android SDK

Αφού δείξαμε στην προηγούμενη ενότητα πως δημιουργούμε ένα νέο project για μία εφαρμογή android, θα αναλύσουμε τα βασικά τμήματα από τα οποία αποτελείται κάθε εφαρμογή και πως αυτά αλληλεπιδρούν για να πάρουμε το επιθυμητό αποτέλεσμα.

3.3.1 Activities

Το activity είναι το δομικό στοιχείο κάθε εφαρμογής, το οποίο παρέχει μία “οθόνη” με την οποία ο χρήστης αλληλεπιδρά. Σε κάθε activity δίνεται ένα παράθυρο στο οποίο σχεδιάζει το User Interface. Το παράθυρο αυτό συνήθως πιάνει όλη την οθόνη αλλά μπορεί να είναι και μικρότερο ή να βρίσκεται πάνω από άλλα παράθυρα.

Μία εφαρμογή συνήθως έχει πολλαπλά activities τα οποία αλληλοσυνδέονται. Ο γενικός κανόνας είναι ότι υπάρχει ένα activity το οποίο θα είναι το βασικό (main), το οποίο τρέχει όταν ο χρήστης ξεκινάει την εφαρμογή. Κάθε φορά που ξεκινάει ένα νέο activity, το προηγούμενο σταματάει, αλλά το σύστημα το διατηρεί σε μία στοίβα (back stack). Το νέο activity μπαίνει στην κορυφή της στοίβας και γίνεται το ενεργό activity. Το back stack ακολουθεί την αρχή last in, first out, οπότε όταν το τρέχον activity τελειώσει ή ο χρήστης πατήσει το κουμπί back, αυτό βγαίνει από τη στοίβα και γίνεται ενεργό το προηγούμενο activity, το οποίο βρίσκεται πλέον και στην κορυφή της λίστας.

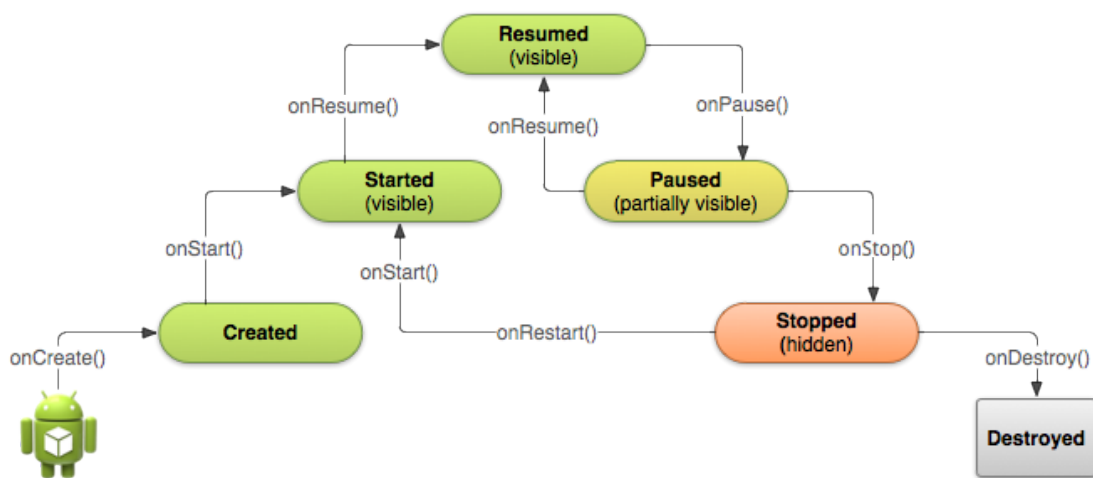
Όταν ένα activity σταματήσει, επειδή ξεκινάει ένα νέο, ενημερώνεται για αυτή την αλλαγή κατάστασης από τις μεθόδους κλήσης του κύκλου ζωής του (lifecycle callback methods). Υπάρχουν αρκετές τέτοιες μέθοδοι που μπορεί να λάβει ένα activity, λόγω αλλαγής της κατάστασης του – είτε επειδή το σύστημα το δημιουργεί, το σταματάει, το επαναφέρει, ή το καταστρέφει – και κάθε callback μέθοδος δίνει στον προγραμματιστή της δυνατότητα να κάνει κάτι σχετικό με την αλλαγή κατάστασης. Για παράδειγμα, όταν σταματάει, το activity πρέπει να αποδεσμεύει μεγάλα αντικείμενα όπως συνδέσεις δικτύου ή συνδέσεις σε βάσεις δεδομένων. Όταν το activity συνεχίσει, μπορεί ο προγραμματιστής να ξαναδεσμεύσει τους απαραίτητους πόρους και να συνεχίσει τις ενέργειες που διακόπηκαν. Οι αλλαγές αυτές κατάστασης είναι όλες μέρος του κύκλου ζωής του activity.

Κάθε activity δημιουργείται σαν υποκλάση της κλάσης “Activity” του Android ή κάποιας υποκλάσης του. Η καινούρια κλάση πρέπει να υλοποιεί τις callback μεθόδους τις οποίες όπως αναφέραμε καλεί το σύστημα όταν περνά ανάμεσα στις διαφορετικές καταστάσεις του κύκλου ζωής, οι βασικότερες των οποίων είναι:

- onCreate() Η πιο βασική μέθοδος η οποία πρέπει σχεδόν πάντα να υλοποιείται. Καλείται όταν δημιουργείται το activity από το

σύστημα, οπότε στην υλοποίηση πρέπει να περιλαμβάνεται η αρχικοποίηση των απαραίτητων τμημάτων της εφαρμογής. Επίσης εδώ καλείται η μέθοδος `setContentView()` η οποία ορίζει το layout για το user interface του activity.

- `onPause()` Αυτή η μέθοδος καλείται από το σύστημα σαν πρώτη ένδειξη ότι ο χρήστης φεύγει από το activity (αν και αυτό δεν σημαίνει πάντα ότι το activity θα καταστραφεί). Εδώ συνήθως πρέπει να γίνονται όποιες αλλαγές πρέπει να διατηρηθούν πέρα από το τρέχον session του χρήστη.



Εικ. 3.10 Ο κύκλος ζωής ενός Android Activity.

Οι υπολοιπες μέθοδοι, καθώς και η σειρά με την οποία το σύστημα τις καλεί φαίνονται στην εικόνα 3.10. Κάθε νέο activity ξεκινάει όπως αναφέραμε με την `onCreate()`. Έπειτα καλείται η `onStart()` η οποία καλείται κάθε φορά που το activity έχει δημιουργηθεί αλλά δεν έχει καταστραφεί, όπως για παράδειγμα αν κάποια ενέργεια της εφαρμογής μας ή του συστήματος ξεκινά κάποια άλλη εφαρμογή. Όταν η εφαρμογή αυτή σταματήσει θα επανέλθει η δικιά μας εφαρμογή στο προσκήνιο και το σύστημα θα καλέσει την μέθοδο `onStart()` και όχι την `onCreate()`. Η μέθοδος `onResume()` καλείται όταν το activity μας ήταν μερικώς ορατό αλλά κάποιο άλλο activity ήταν στο προσκήνιο όπως για παράδειγμα ένα popup παράθυρο. Έτσι όταν το popup παράθυρο τελειώσει και επανέλθει το activity στο προσκήνιο τότε καλείται η `onResume()`. Όπως αναφέραμε όταν ένα activity δημιουργείται τότε περνάει και από τις τρεις μεθόδους με τη σειρά που φαίνεται στην εικ. 3.10.

Αντίθετα στις περιπτώσεις που αναφέραμε, όταν το activity δεν βρίσκεται πλέον στο προσκήνιο, καλείται αρχικά η `onPause()` αν πρόκειται για κάτι το οποίο δεν θα πιάνει όλη την οθόνη, η `onStop()` αν πρόκειται για άλλο activity που θα έρθει στο προσκήνιο, και τέλος η `onDestroy()` όταν το activity θα καταστραφεί, για παράδειγμα όταν κλείνει η εφαρμογή μας, ή όταν το σύστημα την κλείσει για να ελευθερώσει πόρους.

Τέλος πρέπει να αναφέρουμε ότι κάθε activity πρέπει να δηλώνεται στο android manifest κάτι που θα αναλύσουμε σε επόμενη υπενότητα. Προς το παρόν ας αναφέρουμε ότι κάθε activity πρέπει να έχει έναν κόμβο `<activity>` κάτω από τον κόμβο `<application>`.

3.3.2 Layouts

Τα layouts στο Android ορίζουν οπτικές δομές για το User Interface ενός activity. Δηλώνοντας ένα Layout ο προγραμματιστής στην ουσία σχεδιάζει την εμφάνιση που θα έχει το activity κατά το χρόνο εκτέλεσης. Τα layouts μπορούν να δηλωθούν με δύο τρόπους:

- Δηλώση User Interface στοιχείων σε XML. Το Android προσφέρει λεξιλόγιο σε XML το οποίο αντιστοιχεί σε κλάσεις `View` και υποκλάσεις.
- Υλοποίηση στοιχείων του layout κατά την εκτέλεση. Η εφαρμογή μπορεί να δημιουργεί αντικείμενα των κλάσεων `View` και `ViewGroup` προγραμματιστικά.

Το Android framework δίνει την ευελιξία στον προγραμματιστή να χρησιμοποιήσει οποιοδήποτε από τους δύο τρόπους επιθυμεί (είτε και τους δύο) για να δηλώσει και να διαχειριστεί το User Interface της εφαρμογής. Για παράδειγμα μπορεί να δηλώσει τα βασικά layouts σε XML και έπειτα να προσθέσει κώδικα που θα τα τροποποιεί, αυτά και τις ιδιότητές τους κατά τη διάρκεια της εκτέλεσης.

Ο προτεινόμενος τρόπος δήλωσης είναι μέσω των αρχείων XML, και αυτό διότι μπορούν να δημιουργηθούν διαφορετικά αρχεία, με διαφορετική συμπεριφορά για διαφορετικά μεγέθη οθονών, προσανατολισμού ή και γλωσσών. Επίσης διαχωρίζεται η παρουσίαση της εφαρμογής, από τον κώδικα που χειρίζεται την συμπεριφορά της, το οποίο επιτρέπει την αλλαγή για παράδειγμα της εμφάνισης χωρίς να

χρειάζεται να ξαναγίνει compile ο κώδικας. Τέλος η δήλωση με XML επιτρέπει την ευκολότερη οπτικοποίηση του layout, και γίνεται ευκολότερη η αποσφαλμάτωση.

Για να δηλώσουμε ένα Layout, ξεκινάμε με τον τύπο του Layout που θα χρησιμοποιήσουμε, και θέτουμε αυτό σαν πατέρα ή αρχικό κόμβο στο XML. Οι τέσσερις βασικοί τύποι είναι:

- Linear layout: Αυτός ο τύπος μας επιτρέπει να τοποθετήσουμε τα στοιχεία του UI γραμμικά είτε οριζόντια είτε κάθετα.
- Relative layout: Ο σχετιστικός τύπος στον οποίο κάθε στοιχείο έχει έναν κωδικό (ID) και κάθε στοιχείο μπαίνει σε μία θέση σχετικά με κάποιο άλλο στοιχείο (αριστερά, δεξιά, πάνω, κάτω).
- List View: Σε αυτόν τον τύπο δημιουργείται μία λίστα με στοιχεία τα οποία συνήθως τοποθετούνται κάθετα. Το προτέρημα αυτού του τύπου είναι ότι μπορούμε να βάλουμε στοιχεία τα οποία συνήθως υπερβαίνουν το μέγεθος της οθόνης.
- Grid View: Ο τελευταίος τύπος μας δίνει τη δυνατότητα να δημιουργήσουμε πλέγματα εικονικών στοιχείων.

Όλοι οι παραπάνω τύποι layout μπορούν να χρησιμοποιηθούν επανηλειμμένα εμφολευμένοι ο ένας στον άλλον για την δημιουργία πολυπλοκων Layout.

Το Android προσφέρει ένα τεράστιο εύρος UI στοιχείων, τα κυριότερα από τα οποία είναι:

- TextView: Γραφικό στοιχείο για την παρουσίαση κειμένου.
- EditText: Γραφικό στοιχείο μέσω του οποίου ο χρήστης μπορεί να εισάγει κείμενο.
- Button: Γραφικό στοιχείο που αναπαριστά ένα κουμπί για την υλοποίηση κάποιας ενέργειας, για παράδειγμα αποστολή στοιχείων που εισήχθησαν σε κάποιο EditText.

Με τα τρία αυτά βασικά στοιχεία, και την χρήση των ιδιοτήτων τους, μπορούν να υλοποιηθούν πολύ πολύπλοκες εφαρμογές. Το σύνολο βέβαια των στοιχείων, τα οποία μπορούν να βρεθούν στην διεύθυνση <http://developer.android.com/guide/topics/ui/index.html>, υλοποιούν κάθε μορφής εφαρμογή, και καθώς το Android εξελίσσεται η λίστα αυτών θα αυξάνεται.

3.3.3 *Android Manifest*

Κάθε εφαρμογή πρέπει να έχει το `AndroidManifest.xml` αρχείο (με αυτό το ακριβές όνομα) στο `root directory`. Το αρχείο `manifest` παρουσιάζει πληροφορίες της εφαρμογής απαραίτητες στο σύστημα, πληροφορίες τις οποίες το σύστημα πρέπει να έχει πριν ξεκινήσει την εκτέλεση της εφαρμογής. Κάποιες από τις λειτουργίες που εκτελεί το `manifest` είναι οι ακόλουθες:

- Ορίζει το πακέτο Java της εφαρμογής. Το όνομα του πακέτου έχει το ρόλο του μοναδικού αναγνωριστικού της εφαρμογής.
- Περιγράφει τα τμήματα της εφαρμογής. Ορίζει τις κλάσεις της Java που υλοποιούν κάθε κομμάτι, και τις δυνατότητες τους (π.χ. ποιά μηνύματα τύπου `Intent` μπορεί να χειριστεί). Αυτές οι δηλώσεις ενημερώνουν το σύστημα ποιά είναι τα τμήματα και υπό ποιές συνθήκες μπορεί να εκτελεστεί το καθένα.
- Καθορίζει ποιές διεργασίες θα δεχθούν ποιά τμήματα της εφαρμογής.
- Δηλώνει ποιά δικαιώματα πρέπει να έχει η εφαρμογή ώστε να μπορέσει να έχει πρόσβαση σε προστατευμένα τμήματα του `Android API`.
- Επίσης δηλώνει ποιά δικαιώματα πρέπει να έχουν οι υπόλοιπες εφαρμογές για να αλληλεπιδράσουν με τα διάφορα τμήματα της εφαρμογής.
- Δηλώνει την χαμηλότερη έκδοση του `Android` που απαιτείται.
- Καταγράφει τις βιβλιοθήκες με τις οποίες πρέπει να συνδεθεί η εφαρμογή.

Η δομή του αρχείου `manifest` ακολουθεί τις αρχές των αρχείων `XML`. Αρχικά δηλώνουμε στο αρχείο ότι πρόκειται για `XML` με την εντολή:

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

Έπειτα ξεκινάμε με τον αρχικό κόμβο που είναι πάντα ο `<manifest>`. Μέσα σε αυτόν θέτουμε αρχικά κόμβους σχετικούς με τα δικαιώματα που απαιτεί ή απαιτούνται για να τρέξει η εφαρμογή. Έπειτα ξεκινάει ο κόμβος `<application>` μέσα στον βαζουμε ένα ξεχωριστό κόμβο για κάθε `<activity>` που έχουμε, με όσες ιδιότητες θέλουμε να δηλώσουμε. Μια αναλυτική εισαγωγή με όλα τα στοιχεία που μπορούμε

να χρησιμοποιήσουμε, υπάρχει στη σελίδα του Android ([http:// developer.android.com /guide /topics /manifest /manifest-intro.html#filestruct](http://developer.android.com/guide/topics/manifest/manifest-intro.html#filestruct)).

3.3.4 Services – Broadcast Receivers – Content Providers

Τα services είναι τμήματα της εφαρμογής που αναπαριστούν είτε τη “θέληση” μια εφαρμογής να υλοποιήσει κάποια πράξη που μπορεί να τρέχει για μεγαλύτερο χρονικό διάστημα από το σύνηθες χωρίς να αλληλεπιδρά με το χρήστη ή να προσφέρει κάποια λειτουργικότητα για χρήση από άλλες εφαρμογές. Τα services προσφέρουν δύο απλά χαρακτηριστικά.

- Ένα μηχανισμό για να ενημερώσουν το σύστημα για κάτι που θέλει να κάνει η εφαρμογή στο παρασκήνιο.
- Ένα μηχανισμό για να εκθέσει η εφαρμογή κάποια από τη λειτουργικότητα της σε άλλες εφαρμογές.

Το BroadcastReceiver είναι ένα κομμάτι της εφαρμογής που χρησιμοποιείται για να δέχεται η εφαρμογή Intents. Αυτός ο μηχανισμός βέβαια δεν μπορεί να χειριστεί Intents που δημιουργούνται για το startActivity().

Τα ContentProviders είναι το τελευταίο από τα βασικά κομμάτια από τα οποία μπορεί να αποτελείται μια εφαρμογή. Η χρήση του είναι για την αποστολή δεδομένων σε πολλαπλές άλλες εφαρμογές. Ένα παράδειγμα είναι τα δεδομένα των επαφών του τηλεφώνου τα οποία μπορεί να χρησιμοποιηθούν από πολλαπλές εφαρμογές, οπότε αυτά αποθηκεύονται σε content provider. Αν τα δεδομένα θα χρησιμοποιηθούν από μία μόνο εφαρμογή χρησιμοποιείται αντίθετα η κλάση SQLiteDatabase που χειρίζεται τη βάση δεδομένων SQLite που χρησιμοποιεί το Android.

3.3.5 Intents

Τα intents είναι αντικείμενα επικοινωνίας που χρησιμοποιούνται για να ζητήσουν μία ενέργεια από κάποιο άλλο τμήμα της εφαρμογής. Υπάρχουν πολλοί τρόποι με τους οποίους τα intents διεκπεραιώνουν την

επικοινωνία ανάμεσα στα τμήματα της εφαρμογής, παρόλα αυτά οι βασικές χρήσεις είναι τρεις:

- Για την εκκίνηση ενός activity.

Το activity αντιπροσωπεύει μία οθόνη στην εφαρμογή. Μπορούμε να ξεκινήσουμε ένα νέο activity περνώντας ένα intent στη μέθοδο `startActivity()`. Το Intent περιγράφει το activity που θα ξεκινήσει και περιέχει τα απαραίτητα δεδομένα.

Αν θέλουμε να πάρουμε κάποιο αποτέλεσμα από το activity που ξεκινήσαμε καλούμε αντίστοιχα τη μέθοδο `startActivityForResult()`. Το αρχικό activity λαμβάνει το αποτέλεσμα σαν ένα δεύτερο αντικείμενο Intent στην μέθοδο `onActivityResult()`.

- Για την εκκίνηση ενός service.

Όπως αναφέραμε τα services είναι τμήματα της εφαρμογής που εκτελούν εργασίες στο παρασκήνιο χωρίς την αλληλεπίδραση του χρήστη. Ξεκινάμε ένα service για να εκτελέσει μία λειτουργία (όπως για παράδειγμα το “κατέβασμα” ενός αρχείου) περνώντας ένα αντικείμενο Intent στη μέθοδο `startService()`. Το αντικείμενο service περιγράφει το service που θα ξεκινήσει και περιέχει τα απαραίτητα δεδομένα.

- Για την παράδοση ενός broadcast

Τα broadcasts είναι μηνύματα που μπορούν να λάβουν όλες οι εφαρμογές. Για να στείλουμε ένα broadcast περνάμε ένα αντικείμενο Intent στις μεθόδους `sendBroadcast()`, `sendOrderedBroadcast()`, και `sendStickyBroadcast()`.

Υπάρχουν δύο τύποι Intent:

- Τα explicit Intents καθορίζουν το τμήμα της εφαρμογής που θα εκτελεστεί ονομαστικά (το πλήρες όνομα της κλάσης). Συνήθως χρησιμοποιείται για τμήματα της εφαρμογής μας για τα οποία γνωρίζουμε το ακριβές όνομα της κλάσης του activity ή του service που θέλουμε να εκτελέσουμε.
- Τα Implicit Intents δεν καθορίζουμε κάποιο τμήμα της εφαρμογής συγκεκριμένα αλλά ορίζουν γενικές ενέργειες που θέλουμε να εκτελεστούν, το οποίο επιτρέπει τμήματα άλλων εφαρμογών να τις χειριστούν. Για παράδειγμα να θέλουμε να δείξουμε μία τοποθεσία στο χρήστη, χρησιμοποιούμε ένα Implicit Intent για να

μπορέσει μια κατάλληλη εφαρμογή (πχ η εφαρμογή google maps) να την προβάλει.

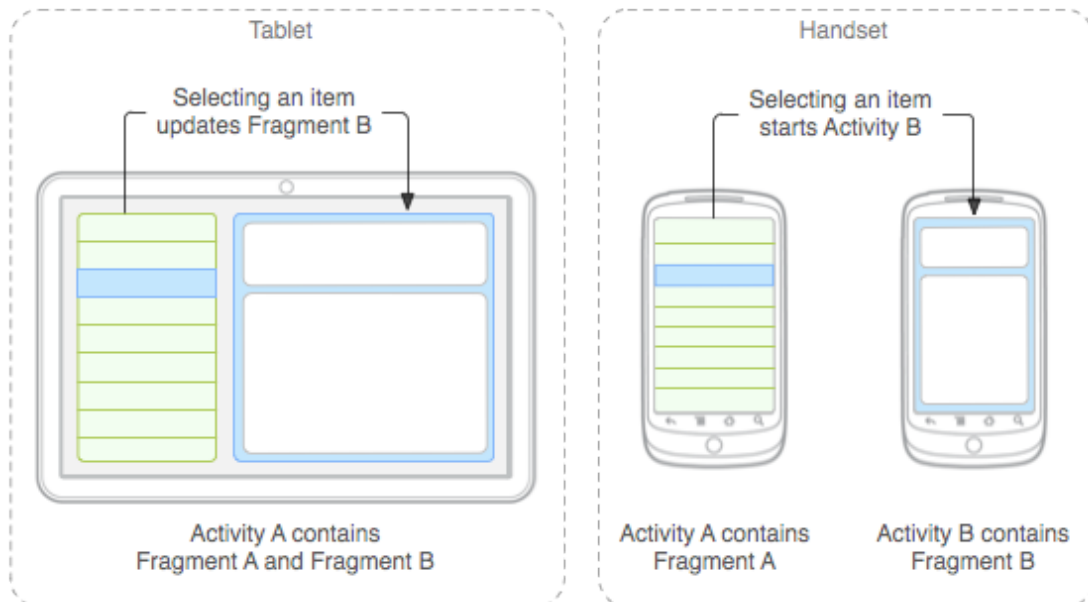
3.3.6 Fragments

Το τελευταίο κομμάτι του API που πρέπει να αναφέρουμε πριν κλείσουμε το εισαγωγικό tutorial είναι τα fragments. Τα fragments είναι ένας νέο τρόπος δημιουργίας User Interface που εισήχθη στην έκδοση 3.0 του Android. Οι εκδόσεις 3.x του Android ήταν μόνο για tablets, και καθώς αυτά είχαν μεγαλύτερες οθόνες από τα κινητά τηλέφωνα, οι προγραμματιστές μπορούσαν να δημιουργήσουν δυναμικά και ευέλικτα Uis.

Η λογική των fragments, εξυπηρετεί την δημιουργία μικρότερων τμημάτων UI τα οποία μπορούν να ανανεώνονται, να δημιουργούνται και να καταστρέφονται, χωρίς να επιρρεάζουν ολόκληρο το layout της εφαρμογής. Για παράδειγμα, μια εφαρμογή ειδήσεων, μπορεί, αντί να έχει ένα μοναδικό στοιχείο, που θα περιέχει όλες τις ειδήσεις, και πατώντας σε κάποια από αυτές να δημιουργείται νέο layout που θα πιάνει όλη την οθόνη, με την χρήση fragments, μπορούμε να έχουμε 2 τμήματα στην οθόνη, μία λίστα στα αριστερά που θα έχει τους τίτλους των ειδήσεων και ένα τμήμα στα δεξιά που θα εμφανίζεται η είδηση. Έτσι αντί να πηγαίνουμε πίσω μπρος στα activities κάθε φορά που θέλουμε να διαβάσουμε μία νέα είδηση, μπορούμε απλά πατώντας σε κάποια είδηση στο fragment στα αριστερά, να ανανεώνει το fragment της είδησης στα δεξιά.

Το δεύτερο πλεονέκτημα της χρήσης fragments είναι η επαναχρησιμοποίηση τους από πολλαπλά activities. Κάθε fragment, όπως και τα activities αποτελείται από μία κλάση η οποία υλοποιεί τη λογική του fragment, και ένα .xml layout το οποίο περιγράφει την γραφική του δομή. Όπως αναφέραμε τα fragments πλέον από την έκδοση 4.0 του Android και έπειτα είναι μέρος του κοινού API για tablets, κινητά αλλά και τις νέες συσκευές όπως smartTVs, android wearables κλπ, οπότε μπορούμε να χρησιμοποιήσουμε τα fragments αναλόγως με το μέγεθος της οθόνης που έχουμε. Για παράδειγμα αν η εφαρμογής μας τρέχει σε ένα tablet θα ακολουθείται η λογική που αναφέραμε προηγουμένως για την εφαρμογή ειδήσεων. Αντιθετα η ίδια εφαρμογή σε κινητά με μικρότερη οθόνη, μπορεί να χρησιμοποιεί τα δύο fragments με την λογική της αντικατάστασης του fragment των τίτλων ειδήσεων, με την

είδηση. Το κέρδος για τον προγραμματιστή σε αυτή την περίπτωση είναι ότι δεν χρειάζεται να σχεδιάσει διαφορετικά activities και layouts για τις δύο περιπτώσεις, αλλά χρησιμοποιώντας τα δύο fragments να δημιουργεί συνδυασμούς που θα βελτιστοποιούν την εμπειρία χρήσης αναλόγως με την συσκευή.



Εικ. 3.11 Παραδειγμα σχεδιασμού με fragments για διαφορετικές οθόνες.

4

Facebook Log in

Το πρώτο σύστημα που υλοποιήθηκε στην εφαρμογή είναι το login με τον λογαριασμό facebook του χρήστη. Η χρήση του facebook account έχει γίνει πολύ δημοφιλής σαν μέσο δημιουργίας λογαριασμού σε διαφορετικές υπηρεσίες τον τελευταίο καιρό καθώς αφενός επιτρέπει στο χρήστη να ξεκινήσει να χρησιμοποιεί την υπηρεσία πολύ γρήγορα, χωρίς να μπαίνει στη χρονοβόρα διαδικασία δημιουργίας λογαριασμού, αφετέρου δε δίνει στους δημιουργούς της υπηρεσίας την ευκαιρία να πάρουν τα στοιχεία που ζητάνε – με την συγκατάθεση πάντα του χρήστη – για να έχουν γνώση του προφίλ του χρήστη.

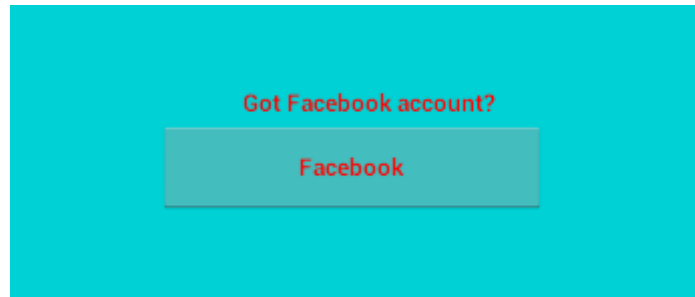
4.1 Αρχιτεκτονική

Το σύστημα για login με λογαριασμό facebook, αποτελείται από τρία τμήματα. Τον server του facebook, τον server της υπηρεσίας μας, και το android application. Αυτό που θέλουμε να πετύχουμε, είναι να έχει ο server της υπηρεσίας μας, κάποια στοιχεία για τον χρήστη έτσι ώστε να μπορεί να δημιουργήσει ένα προφίλ για αυτόν. Ο server της υπηρεσίας μας θα πάρει τα στοιχεία αυτά απευθείας από τον server του facebook με τη χρήση ενός access token. Το token δημιουργείται από το facebook και

περιέχει πληροφορίες για τον χρήστη, ποιά στοιχεία του έχει εξουσιοδοτήσει να δοθούν στην υπηρεσία μας, καθώς και ποιά είναι η υπηρεσία μας. Το token αυτό είναι προσωρινό, έχει δηλαδή διάρκεια ζωής περίπου μία ώρα.

Ο χρήστης, ανοίγοντας την εφαρμογή μας, έχει τη επιλογή να κάνει login με το λογαριασμό facebook. Πατώντας το κουμπί για την επιλογή αυτή, δημιουργείται μια σύνδεση είτε με το facebook android app (αν υπάρχει εγκατεστημένο), είτε μέσω browser στη σελίδα του facebook, για σύνδεση. Ένημερώνεται για τα στοιχεία που ζητάμε, και αν θέλει να δώσει στην υπηρεσία μας εξουσιοδότηση για να τα πάρει. Αφού ο χρήστης το δεχθεί, το facebook δημιουργεί το μοναδικό access token, το οποίο και μας στέλνει στο app. Η εφαρμογή με τη σειρά της, στέλνει το token πίσω στο server μας, ο οποίος κάνει τον έλεγχο για να δει αν ο χρήστης έχει ξαναχρησιμοποιήσει την υπηρεσία μας. Αν είναι η πρώτη φορά που το κάνει με το facebook account, τότε δημιουργείται το προφίλ, και λαμβάνουμε πίσω στο app το μοναδικό ID της υπηρεσίας μας, με το οποίο αναγνωρίζεται ο χρήστης. Αυτό ελέγχεται κάνοντας κλήση για signin στο service μας με το authentication-token. Αν ο server της υπηρεσίας μας απαντήσει ότι δεν υπάρχει ο χρήστης για login με αυτό το token, τότε καλείται το αντίστοιχο service για sign-up με το facebook token.

Τέλος πρέπει να αναφέρουμε ότι για λόγους ασφάλειας το facebook μας ζητά να δημιουργήσουμε ένα app στο server του, το οποίο θα συνδέει το android app μας, με το server μας για να μην είναι δυνατόν να χρησιμοποιηθεί το token από μη εξουσιοδοτημένους server. Αυτό υπολοποιείται αφενός για το server μας δηλώνοντας τη διεύθυνση του (URL) και αφετέρου δίνοντας μας ένα μοναδικό ID το οποίο δηλώνουμε στο android app μας, και το οποίο χρησιμοποιείται από το facebook SDK για να ταυτοποιείται το android app μας με το facebook app.



Εικ. 4.1 Το κουμπί για σύνδεση με λογαριασμό facebook

4.2 Περιγραφή Λειτουργιών

Θα περιγράψουμε τώρα πως υλοποιήσαμε το σύστημα, μόνο από τη σκοπιά της εφαρμογής μας. Η υλοποίηση του server της υπηρεσίας μας είναι πέρα από το πεδίο αυτής της διπλωματικής. Για να μπορέσουμε όμως να το χρησιμοποιήσουμε στον κώδικα μας, θα δεχθούμε ότι υπάρχει σε κάποιο URL ένα RESTful service. Συγκεκριμένα υπάρχουν 2 URLs, ένα για `signin`, και ένα `signur`, και έστω ότι το πρώτο call γίνεται με HTTP GET, στέλνοντας το token σαν παράμετρο, και το δεύτερο γίνεται με HTTP POST στέλνοντας το token με JSON object σαν παράμετρο στο POST call.

4.2.1 Facebook session handling

Θα δείξουμε πως χειριζόμαστε lifecycle του facebook session με το SDK. Όπως αναφέρθηκε σε κάθε `app` που θέλουμε να χρησιμοποιήσουμε facebook login πρέπει αρχικά να δημιουργήσουμε ένα `app` στο server του facebook. Αυτό ξεκινάει από την διεύθυνση `developers.facebook.com`. Εκεί από το tab "Apps" διαλέγουμε "Add new app". Τα παράθυρα διαλόγου που θα μας βγάλει η σελίδα, μας καθοδηγεί επιλέγοντας αρχικά το όνομα της εφαρμογής μας, και έπειτα μας δίνει ένα μοναδικό ID, το οποίο βάζουμε στο manifest αρχείο μας. Αυτό το ID χρησιμοποιείται όπως είπαμε για να γνωρίζει ο server του facebook ποιο είναι το `app` που ζητά πρόσβαση κάθε φορά.

Έπειτα αφού συμπληρώνουμε την διεύθυνση (URL) του server της εφαρμογής μας, είμαστε έτοιμοι να χρησιμοποιήσουμε το facebook login στο `app` μας.

Το επόμενο βήμα είναι να κατεβάσουμε το SDK του facebook για Android. Πηγαίνουμε στη διεύθυνση <https://developers.facebook.com/resources/facebook-android-sdk-current.zip> και παίρνουμε το συμπιεσμένο αρχείο το οποίο μεταξύ άλλων περιέχει τη βιβλιοθήκη, καθώς και παραδείγματα χρήσης.

Για να το χρησιμοποιήσουμε πρέπει να το εισάγουμε στο Eclipse. Ξεκινάμε πατώντας File -> Import... και επιλέγουμε το φάκελο, στον οποίο το έχουμε αποσυμπιέσει. Αφού το Eclipse το εισάγει πηγαίνουμε στο project μας και επιλέγουμε "properties" κάνοντας δεξιά κλικ πάνω του, μετά την επιλογή "Android" και τέλος στην περιοχή "Library" δεξιά πατάμε add και προσθέτουμε το FacebookSDK σαν βιβλιοθήκη που θα χρησιμοποιεί το project μας.

Το τελευταίο βήμα πριν αρχίσουμε να γράφουμε κώδικα για τον χειρισμό του Facebook session είναι να προσθέσουμε κάποια στοιχεία που απαιτούνται στο AndroidManifest. Αρχικά δημιουργούμε ένα string value, με όνομα app_id και τιμή τον κωδικό που μας έχει δώσει το application στον server του facebook.

Έπειτα ανοίγουμε το manifest και προσθέτουμε τα ακόλουθα μέσα στο application tag:

- Ένα activity tag με name:"com.facebook.LoginActivity"

```
<activity android:name="com.facebook.LoginActivity"/>
```

- Το ακόλουθο meta-data tag

```
<meta-data android:value="@string/app_id" android:name="com.facebook.Sdk.ApplicationId"/>
```

Τώρα λοιπόν είμαστε έτοιμοι για να κάνουμε τις κλήσεις μας στο GraphUser του facebook. Το βασικό δομικό στοιχείο, είναι το αντικείμενο Session. Αυτό κρατάει όλα τα στοιχεία του χρήστη και αντιπροσωπεύει μία "συνεδρία" σύνδεσης της εφαρμογής με το server του facebook. Το SDK μας δίνει δύο τρόπους για να δημιουργήσουμε το session. Ο πρώτος τρόπος και ευκολότερος είναι μέσω της κλάσης LoginButton. Η κλάση αυτή είναι ένα widget το οποίο το εισάγουμε σε ένα View layout, και υλοποιεί ένα κουμπί το οποίο πατώντας το ο χρήστης συνδέεται στο facebook (δίνοντας username και password) και έπειτα του ζητείται να επιβεβαιώσει την πρόσβαση της εφαρμογής μας στα στοιχεία του. Αφού γίνει αυτό το session έχει δημιουργηθεί και μπορούμε να το χειριστούμε όπως και στην επόμενη περίπτωση.

Ο δεύτερος τρόπος λέγεται native facebook login και παρακάμπτει το κουμπί που αναφέραμε. Η υλοποίηση γίνεται χειριζόμενοι απευθείας το session ξεκινώντας το με τη κλήση της μεθόδου openActiveSession(). Σε αυτή δημιουργούμε ένα νέο callback το οποίο ενεργοποιείται κάθε φορά που αλλάζει η κατάσταση του session (από κλειστό σε ανοιχτό) και κάνοντας override την μέθοδο call ζητάμε τα στοιχεία του χρήστη που θέλουμε. Στην περίπτωση της εφαρμογής μας ζητάμε μόνο το e-mail. Τα στοιχεία που θέλουμε να πάρουμε τα δηλώνουμε σαν 3η παράμετρο στην μέθοδο openActiveSession() σαν List. Η υλοποίηση αυτή συνιστάται μόνο σε περιπτώσεις που δεν μας ενδιαφέρει να συνεχίσουμε τον κύκλο ζωής του session πέρα από τη λήψη του access token.

Αφού δημιουργηθεί το νέο session με τα στοιχεία που ζητάμε κάνουμε ένα Request για να τραβήξουμε τα στοιχεία αυτά. Μόλις αυτό ολοκληρωθεί έχουμε το accessToken που χρειαζόμαστε για τον server της υπηρεσίας μας, το οποίο θα στείλουμε με AsyncTask, το οποίο θα αναλύσουμε στην επόμενη ενότητα.

```

public void FacebookLoginCall (View view)
{
    // start Facebook Login
    Session.openActiveSession(this, true, Arrays.asList("public_profile","email"), new
Session.StatusCallback() {

        // callback when session changes state
        @Override
        public void call(Session session, SessionState state, Exception exception)
        {
            if (session.isOpened())
            { // make request to the /me APIs
                Request.newMeRequest(session, new Request.GraphUserCallback() {

                    // callback after Graph API response with user object
                    @Override
                    public void onCompleted(GraphUser user, Response response)
                    {
                        if (user != null)
                        {
                            FacebookLoginCall flc = new FacebookLoginCall(mContext,activity);
                            flc.execute(Session.getActiveSession().getAccessToken());
                        }
                    }
                }
            }
        }
    }).executeAsync();
}

```

```
    }  
    }  
  });  
}
```

4.2.2 *Android AsyncTask API call*

Το δεύτερο βήμα στην υλοποίηση του facebook login είναι να στείλουμε το access token που έχουμε πάρει, πίσω στο server. Ένα βασικό σημείο που πρέπει να επισημάνουμε είναι η χρήση του AsyncTask στο Android. Κάθε εφαρμογή ξεκινάει με ένα βασικό thread το οποίο αναλαμβάνει να εκτελέσει όλες τις λειτουργίες. Αν για κάποιο λόγο το thread τρέχει μία λειτουργία για ένα χρονικό διάστημα μεγαλύτερο από μερικά δευτερόλεπτα τότε το λειτουργικό θεωρεί ότι το app έχει “κολλήσει” και πετάει ένα Popup παράθυρο το οποίο λέει ότι η εφαρμογή μας δεν αποκρίνεται.

Ο σωστός χειρισμός τέτοιων λειτουργιών οι οποίες αναμένουμε να διαρκέσουν κάποιο χρονικό διάστημα είναι μέσω του μηχανισμού AsyncTask του Android. Το AsyncTask είναι μία κλάση η οποία δημιουργεί ένα νέο thread στο οποίο δίνουμε να εκτελέσει τον κώδικα που θέλουμε. Χρησιμοποιείται κατά κόρον για κλήσεις σε server καθώς λόγω των καθυστερήσεων του δικτύου είναι σχεδόν σίγουρο ότι θα έχουν διάρκεια πάνω από την αναμενόμενη.

Για να υλοποιήσουμε ένα AsyncTask δημιουργούμε μία νέα κλάση η οποία κάνει extend την κλάση AsyncTask<String, Integer, String>. Οι τρεις βασικές μέθοδοι είναι οι onPreExecute(), doInBackground() και onPostExecute(), με την σειρά που καλούνται από το σύστημα.

Η onPreExecute() καλείται πριν αρχίσει να τρέχει ο κώδικας που θέλουμε και χρησιμοποιείται για να εκτελέσουμε προκαταρκτικές λειτουργίες όπως αρχικοποίηση μεταβλητών που ίσως χρειαστούν και για τη δημιουργία κάποιου popup μηνύματος που θα ενημερώνει το χρήστη ότι εκτελείται κάποια λειτουργία πχ μία loading screen.

Έπειτα καλείται η κυρίως μέθοδος που είναι η doInBackground η οποία τρέχει τον κώδικα που θέλουμε ασύγχρονα από τη βασική εφαρμογή. Στην περίπτωση μας εδώ δημιουργείται το αντικείμενο HttpGet ή HttpPost αναλόγως με την κλήση που έχουμε να κάνουμε και δημιουργείται το αντικείμενο JSON αν απαιτείται. Η μέθοδος αυτή είναι η μόνη η οποία καλείται εξωτερικά, και καλείται μέσω της μεθόδου

execute(String... params) του αντικειμένου της κλάσης που δημιουργήσαμε. Η μέθοδος αυτή δέχεται σαν όρισμα ένα πίνακα από Strings τα οποία συνήθως είναι στοιχεία που θα χρειαστούμε για τις κλήσεις στο server.

Η δεύτερη κλάση που χρειάστηκε για την κλήση μας είναι η HttpGet. Είναι κλάση της Java η οποία επιτρέπει την δημιουργία και τον χειρισμό αντικειμένων για χρήση με το πρωτόκολλο HTTP. Ο constructor παίρνει σαν παράμετρο το URL στο οποίο θα γίνει η κλήση μας. Πρέπει να έχουμε φροντίσει να περιλάβουμε οποιεσδήποτε παραμέτρους απαιτεί η κλήση κατά δημιουργία του αντικειμένου. Έπειτα μέσω της μεθόδου setHeader μπορούμε να θέσουμε όποια headers χρειάζεται η κλήση.

Τέλος δημιουργούμε ένα αντικείμενο της κλάσης DefaultHttpClient. Ο constructor δέχεται όρισμα τύπου HttpParams, το οποίο θέτουμε μέσω του constructor BasicHttpParams του Android. Επίσης θέτουμε στην κλάση HttpConnectionParams, παραμέτρους σχετικά με το timeout της σύνδεσης αν δεν μπορεί να δημιουργηθεί, για να μη τρέχει το συγκεκριμένο thread επ' αόριστα.

Αφού τελειώσουμε με τις παραμέτρους καλούμε την μέθοδο execute() του αντικειμένου DefaultHttpClient και η επιστροφή της μεθόδου είναι ένα αντικείμενο κλάσης HttpResponse. Από αυτό παίρνουμε ένα αντικείμενο τύπου HttpEntity μέσω της μεθόδου getEntity() και αντίστοιχα από αυτό παίρνουμε ένα αντικείμενο τύπου InputStream. Τέλος αυτό το αντικείμενο το επεξεργαζόμαστε με ένα αντικείμενο τύπου InputStreamReader το οποίο μετατρέπει την ακολουθία byte του InputStream σε ακολουθία χαρακτήρων βάσει συγκεκριμένου encoding που δίνουμε εμείς. Το αποτέλεσμα είναι ένα σύνολο χαρακτήρων το οποίο το επεξεργαζόμαστε σαν string, αλλά στην ουσία είναι ένα JSON αντικείμενο. Από αυτό μπορούμε να πάρουμε, βάσει των κλειδιών που έχουμε, τις τιμές της απάντησης το server που μας ενδιαφέρουν, όπως status για να δούμε αν η κλήση έγινε με επιτυχία, καθώς και τα δεδομένα που μας ενδιαφέρουν

```
public class FacebookLoginCall extends AsyncTask <String, Integer, String > {  
    private ProgressDialog dialog;  
    private Context mContext;
```

```

private Activity activity;
private String status;

public FacebookLoginCall(Context c,Activity activity){
    this.mContext = c;
    this.activity = activity;
}

@Override
protected String doInBackground(String... params) {
    try{
        Looper.prepare();
        HttpGet httpget = new HttpGet(Properties.server + "/login/api_by_fb?token=" +
                                                    params[0]);

        // Depends on your web service
        httpget.setHeader("Content-type", "application/json");
        InputStream inputStream = null;
        HttpParams httpParameters = new BasicHttpParams();
        // Set the timeout in milliseconds until a connection is established.
        // The default value is zero, that means the timeout is not used.
        int timeoutConnection = 10000;
        HttpConnectionParams.setConnectionTimeout(httpParameters,
                                                    timeoutConnection);

        // Set the default socket timeout (SO_TIMEOUT)
        // in milliseconds which is the timeout for waiting for data.
        int timeoutSocket = 10000;
        HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);
        // create object of DefaultHttpClient
        DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);

        HttpResponse response = httpClient.execute(httpget);
        HttpEntity entity = response.getEntity();
        inputStream = entity.getContent();
        BufferedReader reader = new BufferedReader ( new InputStreamReader
                                                    ( inputStream, "UTF-8"), 8);

        StringBuilder sb = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null)
        {
            sb.append(line + "\n");
        }
        String tmp = sb.toString();
        JSONObject loginResponse = new JSONObject(sb.toString());
        status = loginResponse.get("status").toString();
        System.out.println("Login:"+tmp);
    }
}

```

```

        System.out.println("Status:"+status);
        if(status.equals("Success"))
        {
            System.out.println("login response:" + loginResponse.getString (
"mav_user_api_key" ) );
            Properties.mav_user_api_key = loginResponse.getString("mav_user_api_key");
        } else {
            System.out.println("Login Failed!");
        }
        return loginResponse.toString();
    } catch(Exception e){
        e.printStackTrace();
        return null;
    }
}

```

@Override

```

protected void onPreExecute() {
    dialog = ProgressDialog.show(mContext, null, "Loading...");
}

```

@Override

```

protected void onPostExecute(String result) {
    dialog.dismiss();
    if (result == null){
        CharSequence text = "Error Connecting to the Server";
        int duration = Toast.LENGTH_LONG;
        Toast toast = Toast.makeText(mContext, text, duration);
        toast.show();
    } else {
        if (status.equals("Success")) {
            activity.finish();
        } else {
            if (result.equals("Invalid facebook key"))
            {
                CharSequence text = "First time Facebook user, please Sign Up with your
Facebook account first.";
                int duration = Toast.LENGTH_LONG;
                Toast toast = Toast.makeText(mContext, text, duration);
                toast.show();
            }
        }
    }
}
}
}
}
}

```

5

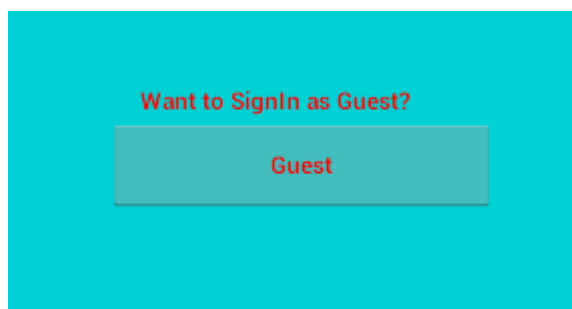
Guest Log in

Το επόμενο υποσύστημα που υλοποιήθηκε είναι η λειτουργία login ενός χρήστη με προσωρινό λογαριασμό. Η λειτουργία αυτή είναι χρήσιμη για χρήστες οι οποίοι θέλουν να πάρουν μια γρήγορα άποψη για την υπηρεσία μας, χωρίς να μπαίνουν στην διαδικασία της δημιουργίας λογαριασμού, κάτι το οποίο μπορεί να αποθαρύνει αρκετούς από τη χρήση της.

5.1 Αρχιτεκτονική

Το υποσύστημα αυτό αποτελείται από μία κλάση η οποία είναι υπεύθυνη για την δημιουργία ενός μοναδικού κωδικού, ο οποίος χρησιμοποιείται σαν αναγνωριστικό της εφαρμογής στη συσκευή, και από μία κλάση για την κλήση της υπηρεσίας στον server. Όπως και στην

περίπτωση του facebook login χρησιμοποιήσαμε το AsyncTask του Android για τις κλήσεις.



Εικ. 5.1 Σύνδεση στο λογαριασμό με προσωρινό λογαριασμό

5.2 Περιγραφή Λειτουργιών

5.2.1 Installation – Κλάση δημιουργίας μοναδικού κωδικού

Το βασικότερο πρόβλημα που αντιμετωπίστηκε στην υλοποίηση του guest login ήταν η δημιουργία του μοναδικού κωδικού με τον οποίο θα αναγνωρίζεται η εφαρμογή μας από τον server.

Υπάρχουν πολλοί τρόποι με τους οποίους γινόταν αυτό, κυρίως παλαιότερα, και συνήθως είχαν σχέση με κάποιο από τα δεδομένα του τηλεφώνου όπως το IMEI, το MEID ή το ESN. Ο κωδικός αυτός λαμβάνεται με την κλήση της μεθόδου TelephonyManager.getDeviceID(). Παρ' όλα αυτά πλέον το Android χρησιμοποιείται και σε συσκευές όπως tablet και τηλεοράσεις τα οποία δεν έχουν τέτοιους κωδικούς.

Ένας δεύτερος κωδικός που χρησιμοποιείται αλλά παρόλα αυτά δεν συνιστάται, είναι η διεύθυνση MAC της κάρτας δικτύου της συσκευής. Και αυτός ο κωδικός όμως, κατά περιπτώσεις μπορεί να μην είναι διαθέσιμος, ή μπορεί να αλλάξει (υπάρχουν εργαλεία που μπορούν να το κάνουν αυτό σε όλα τα λειτουργικά συστήματα βασισμένα στον πυρήνα Linux).

Ο τρίτος κωδικός που μπορεί να χρησιμοποιηθεί είναι το ANDROID_ID. Αυτή η τιμή λαμβάνεται από το Settings.Secure.ANDROID_ID. Είναι 64-bit κωδικός και δημιουργείται όταν η συσκευή ανοίγει για πρώτη φορά. Το πρώτο μειονέκτημα είναι ότι η τιμή αυτή σβήνει όταν η συσκευή επανέρχεται στις εργοστασιακές

της ρυθμίσεις. Το δεύτερο μεινέκτημα είναι ότι σε συσκευές που τρέχουν εκδόσεις του Android πριν την 2.2 δεν είναι 100% αξιόπιστο σύμφωνα με τους προγραμματιστές του Android.

Ο τελευταίος τρόπος, ο οποίος και χρησιμοποιήθηκε, είναι η δημιουργία ενός 128-bit κωδικού μέσω της κλάσης UUID της Java. Η δημιουργία γίνεται με κλήση της μεθόδου `randomUUID()` η οποία παράγει έναν ψευδοτυχαίο κωδικό. Λόγω του μεγάλου εύρους του κωδικού – 128-bit ή 38 δεκαδικά ψηφία – είναι σχεδόν απίθανο να δημιουργηθούν δύο ίδιοι αριθμοί σε δυο διαφορετικές συσκευές.

Ο κωδικός αυτός ανατίθεται σε μία μεταβλητή τύπου `String`, και έπειτα αποθηκεύεται σε ένα αρχείο τοπικά. Κάθε φορά που ζητείται ο κωδικός αυτός, ελέγχουμε το συγκεκριμένο αρχείο. Αν το αρχείο έχει κωδικό τον διαβάζουμε, αλλιώς τον δημιουργούμε.

Η κλάση όπως αναφέραμε είναι η `Installation`. Περιέχει τρεις μεθόδους, την `id()`, την `readInstallationFile()` και την `writeInstallationFile()`. Η μέθοδος που καλείται για την δημιουργία ή ανάγνωση του `UniqueID` είναι η `id`. Η μέθοδος αυτή δηλώνεται `synchronized` καθώς δεν θέλουμε δύο αντικείμενα της μεθόδου να έχουν πρόσβαση στο αρχείο αποθήκευσης την ίδια στιγμή καθώς δεν μπορούμε να υπολογίσουμε ποιο από τα δύο αντικείμενα θα τη χρησιμοποιήσει πρώτο.

Η κλάση περιέχει δύο μεταβλητές την `sID` που θα κρατάει το `UniqueID` και το `final string INSTALLATION` το οποίο αρχικοποιούμε στο όνομα του αρχείου που θα αποθηκεύσουμε τον κωδικό.

Η κλήση της μεθόδου `id` ξεκινά με τον έλεγχο του `sID` αν είναι `null`, δηλαδή αν είναι η πρώτη φορά που καλείται η μέθοδος από την δημιουργία του αντικειμένου. Αν δεν είναι, επιστρέφεται το `UniqueID`. Αλλιώς ελέγχουμε αν υπάρχει το αρχείο στο `file directory` της συσκευής. Αν δεν υπάρχει καλείται η μέθοδος `writeInstallationFile()` η οποία δημιουργεί τον κωδικό με την κλάση `UUID` και τον αποθηκεύει στο αρχείο. Αν υπάρχει το αρχείο παραλείπεται αυτό το βήμα και καλείται η `readInstallationFile()`, διαβάζουμε τον αποθηκευμένο κωδικό και τον επιστρέφουμε.

```
import java.io.File;
```

```
import java.io.FileOutputStream;
```

```
import java.io.IOException;
```

```

import java.io.RandomAccessFile;
import java.util.UUID;
import android.content.Context;

public class Installation {
    private static String sID = null;
    private static final String INSTALLATION = "INSTALLATION";

    public synchronized static String id(Context context) {
        if (sID == null) {
            File installation = new File(context.getFilesDir(), INSTALLATION);
            try {
                if (!installation.exists())
                    writeInstallationFile(installation);
                sID = readInstallationFile(installation);
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        }
        return sID;
    }

    private static String readInstallationFile(File installation) throws IOException {
        RandomAccessFile f = new RandomAccessFile(installation, "r");
        byte[] bytes = new byte[(int) f.length()];
        f.readFully(bytes);
        f.close();
        return new String(bytes);
    }

    private static void writeInstallationFile(File installation) throws IOException {
        FileOutputStream out = new FileOutputStream(installation);
        String id = UUID.randomUUID().toString();
        out.write(id.getBytes());
        out.close();
    }
}

```

5.2.2 Guest Log in Call

Το δεύτερο τμήμα του υποσυστήματος είναι η κλήση στο API της υπηρεσίας μας με το UniqueID που δημιουργήσαμε. Όπως και στην περίπτωση του facebook sign-in η κλήση γίνεται με τη δημιουργία ενός AsyncTask ώστε να επικοινωνήσουμε ασύγχρονα με το server χωρίς να μπλοκάρουμε το βασικό thread της εφαρμογής.

Δημιουργήσαμε μία νέα κλάση η οποία κάνει extend την AsyncTask. Το API call σε αυτή την περίπτωση ήταν Post call οπότε δημιουργούμε ένα HttpPost αντικείμενο. Επίσης η κλήση απαιτεί την αποστολή δεδομένων (το UniqueID) με JSON αντικείμενο.

Τα αντικείμενα JSON είναι ο defacto τρόπος αποστολής δεδομένων με webservices σημερα. Ο λόγος της ευρείας υιοθέτησης τους είναι η ευκολία δημιουργίας και τροποποίησης. Είναι συλλογές από ζευγάρια key-value. Το key είναι το κλειδί με το οποίο αναγνωρίζεται κάθε τιμή (value). Τα κλειδιά μπορεί να είναι μόνο τύπου string και δύο κλειδιά μέσα στο ίδιο αντικείμενο δεν μπορούν να έχουν το ίδιο όνομα. Πέρα από αυτές τις προϋποθέσεις δεν υπάρχει κάτι άλλο που να μας δεσμεύει σχετικά με το όνομα του κλειδιού, συνήθως όμως επιλέγονται ονόματα που να προσδιορίζουν ακριβώς σε τι αναφέρεται η τιμή.

Οι τιμές (values) μπορούν να είναι να είναι οποιουδήποτε τύπου επιθυμούμε, με τα συνήθη να είναι "String", κάποιο αριθμητικό δεδομένο, κάποιο αντικείμενο, ένα άλλο JSON αντικείμενο, ή null για κενή τιμή.

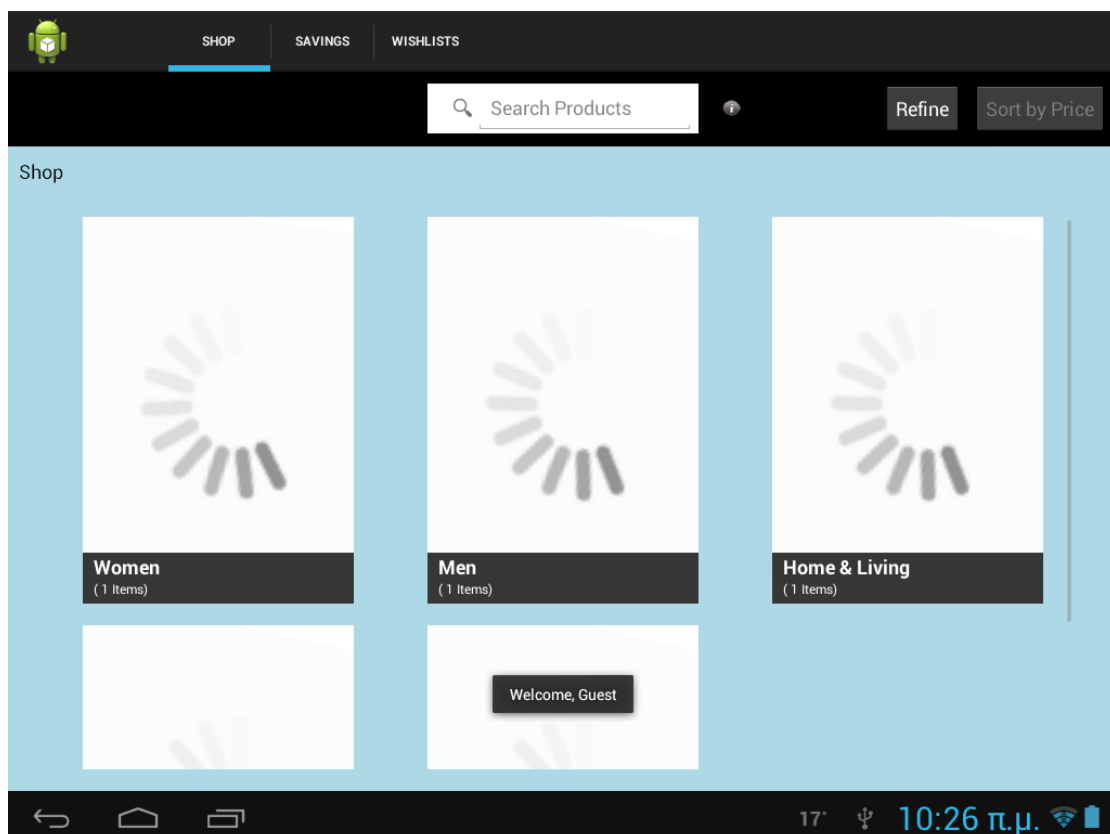
Η Java προσφέρει την κλάση JSONObject για τον χειρισμό JSON αντικειμένων. Ο χειρισμός των αντικειμένων είναι εύκολος. Δημιουργούμε ένα νέο αντικείμενο με τον constructor JSONObject().

Για να προσθέσουμε κάποιο ζευγάρι Key-value καλούμε την μέθοδο put(String key, Object value), όπου το Object value μπορεί να είναι οποιουδήποτε τύπου αναφέραμε προηγουμένως.

Στη περίπτωση της κλήσης μας το μόνο που απαιτεί η υπηρεσία είναι ένα ζευγάρι με key το string "unique_key" και τιμή ένα string με το UniqueID. Αυτό το παίρνουμε από την παράμετρο params[0] της μεθόδου doInBackground(String... params) που όπως αναφέραμε στο προηγούμενο κεφάλαιο στέλνεται από την μέθοδο .execute(String... params) του

αντικειμένου της κλάσης GuestLoginCall που δημιουργήσαμε στο βασικό thread.

Η υπόλοιπη κλάση δεν έχει μεγάλες διαφορές από αυτή του facebook login call. Στην απάντηση του server παίρνουμε τα δεδμεμένα user_apri_key και anonymous_id τα οποία θέτουμε σε δύο global μεταβλητές στις οποίες έχουν πρόσβαση όλες οι κλάσεις για να τις χρησιμοποιήσουν.



Εικ. 5.2 Η εφαρμογή μας ενημερώνει για την επιτυχία του log in

```
import java.io.BufferedReader;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
  
import org.apache.http.HttpEntity;  
import org.apache.http.HttpResponse;  
import org.apache.http.client.methods.HttpPost;  
import org.apache.http.entity.StringEntity;  
import org.apache.http.impl.client.DefaultHttpClient;
```

```

import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.widget.Toast;

public class GuestLoginCall extends AsyncTask<String, Integer, String > {
    private ProgressDialog dialog;
    private Context mContext;
    private Activity activity;
    private String status;

    public GuestLoginCall(Context c,Activity activity){
        this.mContext = c;
        this.activity = activity;
    }

    @Override
    protected String doInBackground(String... params) {
        try{
            HttpPost httppost = new HttpPost(Properties.server + "/mdid");
            // Depends on your web service
            httppost.setHeader("Content-type", "application/json");
            JSONObject json = new JSONObject();
            json.put("unique_key", params[0]);
            json.put("bundle", "12345");
            StringEntity se = new StringEntity(json.toString());
            se.setContentType(new BasicHeader(HTTP.CONTENT_TYPE, "application/json"));
            httppost.setEntity(se);
            InputStream inputStream = null;

```

```

HttpParams httpParameters = new BasicHttpParams();
// Set the timeout in milliseconds until a connection is established.
// The default value is zero, that means the timeout is not used.
int timeoutConnection = 10000;
HttpConnectionParams.setConnectionTimeout(httpParameters,
timeoutConnection);
// Set the default socket timeout (SO_TIMEOUT)
// in milliseconds which is the timeout for waiting for data.
int timeoutSocket = 10000;
HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);
// create object of DefaultHttpClient
DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);

HttpResponse response = httpClient.execute(httppost);
HttpEntity entity = response.getEntity();

InputStream inputStream = entity.getContent();
BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream, "UTF-8"), 8);
StringBuilder sb = new StringBuilder();

String line = null;
while ((line = reader.readLine()) != null)
{
    sb.append(line + "\n");
}
String tmp = sb.toString();
JSONObject loginResponse = new JSONObject(sb.toString());
status = loginResponse.get("status").toString();
System.out.println("Login:"+tmp);
System.out.println("Status:"+status);
if(status.equals("Success")){
    System.out.println("login response:"+ loginResponse. GetString
("mav_user_api_key" ));
    Properties.user_api_key = loginResponse.getString("user_api_key");
    Properties.anonymous_id = loginResponse.getString("anonymous_id");
}

```

```

        else{
            System.out.println("Login Failed!");
        }
        return "Success!";
    }catch(Exception e){
        e.printStackTrace();
        return null;
    }
}

@Override
protected void onPreExecute() {
    dialog = ProgressDialog.show(mContext, null, "Loading...");
}

@Override
protected void onPostExecute(String result) {
    dialog.dismiss();
    if (result==null){
        CharSequence text = "Error Connecting to the Server";
        int duration = Toast.LENGTH_LONG;
        Toast toast = Toast.makeText(mContext, text, duration);
        toast.show();
    }
    else {
        if (status.equals("Success")){
            Toast.makeText(mContext, "Guest", Toast.LENGTH_SHORT).show();
            activity.finish();
        }
        else{
            CharSequence text = "Wrong username or password!";
            int duration = Toast.LENGTH_LONG;
            Toast toast = Toast.makeText(mContext, text, duration);
            toast.show();
        }
    }
}
}
}

```

}

6

Λειτουργία Sign Up

Το τρίτο υποσύστημα που υλοποιήθηκε είναι η δημιουργία συστήματος για sign up, δηλαδή εγγραφή νέου χρήστη στην υπηρεσία μας.

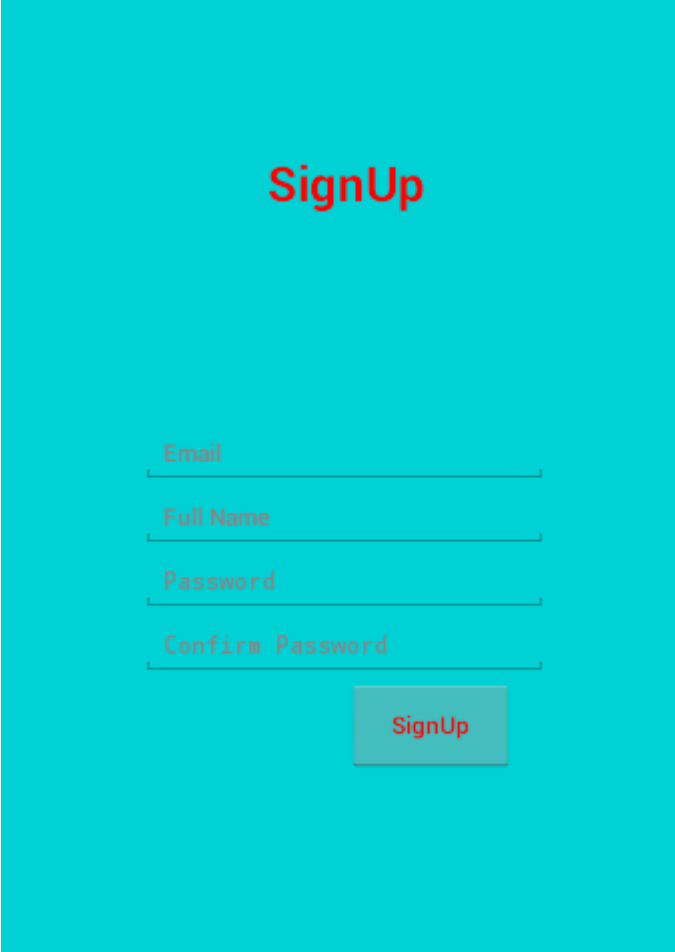
6.1 Αρχιτεκτονική

Το σύστημα sign up απαιτεί την δημιουργία φόρμας στην εφαρμογή για την εισαγωγή των στοιχείων του χρήστη

(ονοματεπώνυμο, email, και κωδικό), έλεγχο της εφαρμογής των στοιχείων ώστε να είναι κατά τα πρότυπα που θέλουμε και τέλος την χρήση του API για την πραγματοποίηση της λειτουργίας στο server.

Υλοποιήθηκε ένα fragment για την εισαγωγή των στοιχείων του χρήστη ώστε να είναι δυνατή η χρήση της φόρμας είτε παράλληλα με τις επιλογές για σύνδεση (login με το account, login με facebook ή login σαν επισκέπτης) αν το επιτρέπει το μέγεθος της οθόνης, είτε σε μικρές οθόνες πατώντας σε κάποιο κουμπί ο χρήστης να αντικαθίσταται το fragment της σύνδεσης με αυτό της πρώτης εγγραφής του χρήστη.

Στο layout έχουμε 4 πεδία για την εισαγωγή των στοιχείων του χρήστη, καθώς και το κουμπί που συνδέεται με την μέθοδο για την επιβεβαίωση των στοιχείων. Αυτή η μέθοδος, αν όλα τα στοιχεία είναι σωστά, καλεί την κλάση η οποία κατά τα γνωστά κάνει ασύγχρονη κλήση στο server.



The image shows a 'SignUp' form on a red background. At the top, the text 'SignUp' is written in a large, bold, red font. Below this, there are four input fields, each with a label above it: 'Email', 'Full Name', 'Password', and 'Confirm Password'. The labels are in a light blue color. Each input field is a simple white line with rounded ends. Below the input fields, there is a red button with the text 'SignUp' in white. The entire form is centered on the red background.

Εικ. 6.1 Δημιουργία νέου λογαριασμού

6.2 Περιγραφή διάταξης (layout)

Το layout του fragment αποτελείται από 4 πεδία EditText και ένα Button. Τα EditText είναι τα πεδία στα οποία ο χρήστης θα εισάγει τα στοιχεία του, ένα για κάθε στοιχείο που απαιτείται (ονοματεπώνυμο, email, κωδικός και επαλήθευση κωδικού). Το Android μας δίνει τη δυνατότητα, μέσω του `android:hint` element να εισάγουμε ένα κείμενο το οποίο θα εμφανίζεται πάνω στο πεδίο και θα αναφέρει στο χρήστη τι πρέπει να συμπληρώσει (πχ email, password κλπ). Το κείμενο αυτό εξαφανίζεται όταν ο χρήστης πατήσει στο πεδίο για να εισάγει κείμενο. Επίσης χρησιμοποιήθηκε το element `android:inputType`. Εισάγοντας την τιμή `textPassword`, το android θα εμφανίζει αστερίσκους καθώς ο χρήστης θα πληκτρολογεί τον κωδικό του ώστε να μην είναι εμφανής για λόγους ασφάλειας.

Επιπλέον αυτό το element μπορεί να επιρρεάσει σε κάποιες συσκευές τον τύπο του εικονικού πληκτρολογίου που εμφανίζεται στην οθόνη, ώστε να βοηθήσουμε τον χρήστη στην εισαγωγή στοιχείων. Για παράδειγμα αν θέσουμε την τιμή `textEmailAddress` στο element πεδίο που θα εισάγει το email του ο χρήστης, το πληκτρολόγιο μπορεί να αλλάξει ώστε να περιλαμβάνει το χαρακτήρα `@` χωρίς να χρειάζεται να πατήσει κάποιο επιπλέον κουμπί ο χρήστης.

Αυτές οι λεπτομέρειες μπορούν να μειώσουν το χρόνο που ο χρήστης ξοδεύει κάνοντας πράγματα τα οποία δεν είναι σχετικά με την υπηρεσία μας, και έτσι να επιρρεαστεί συνολικά η εμπειρία από τη χρήση της εφαρμογής.

Τέλος το πεδίο Button υλοποιεί την κλήση της μεθόδου η οποία κάνει τον έλεγχο των δεδομένων που εισήγαγε ο χρήστης, και πετάει κάποιο σχετικό μήνυμα αν κάποιο από αυτά δεν είναι σύμφωνα με τις προδιαγραφές που έχουμε θέσει. Αυτό γίνεται με δύο τρόπους. Ο πρώτος είναι να εισάγουμε το element `android:onClick` στο αρχείο `.xml` του `u layout`, δίνοντας του σαν τιμή το όνομα της μεθόδου που επιθυμούμε να κληθεί. Αυτός είναι και ο τρόπος που χρησιμοποιήθηκε εδώ.

Ο δεύτερος τρόπος μέσω του activity το οποίο χρησιμοποιεί το fragment. Χρησιμοποιώντας τη μέθοδο `findViewById(R.id.viewID)` δίνουμε

σαν παράμετρο το id που έχουμε δηλώσει στο element android:id στο layout, και μας επιστρέφει η μέθοδος ένα αντικείμενο τύπου View (το οποίο μπορούμε να μετατρέχουμε σε όποιο αντικείμενο View επιθυμούμε μέσω της αντίστοιχης κάστας), το οποίο αποθηκεύουμε σε μία μεταβλητή. Με αυτή τη μεταβλητή μπορούμε να αλλάξουμε όλες τις ιδιότητες του μέσω των αντίστοιχων μεθόδων. Για αυτό που ζητάμε όμως καλούμε την μέθοδο `.setOnClickListener()` μέσα στην οποία κάνουμε `@Override` την μέθοδο `onClick()` στην οποία εισάγουμε τον κώδικα που επιθυμούμε να εκτελεστεί.

```
<?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:background="@color/DarkTurquoise"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/signup_title"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:gravity="center"
        android:text="@string/SignUp"
        android:textColor="@color/Red"
        android:textSize="@dimen/big_text_size"
        android:textStyle="bold" />
    <EditText
        android:id="@+id/signup_email_field"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="3"
        android:hint="@string/Email"
        android:inputType="textEmailAddress"
        android:textSize="@dimen/text_size"
        android:textStyle="bold" />
    <EditText
        android:id="@+id/signup_name_field"
        android:layout_width="0dp"
```

```
android:layout_height="wrap_content"  
android:layout_weight="3"  
android:hint="@string/Fullname"  
android:textSize="@dimen/text_size"  
android:textStyle="bold" />
```

<EditText

```
android:id="@+id/signup_password_field"  
android:layout_width="0dp"  
android:layout_height="wrap_content"  
android:layout_weight="3"  
android:hint="@string/Password"  
android:inputType="textPassword"  
android:textSize="@dimen/text_size"  
android:textStyle="bold" />
```

<EditText

```
android:id="@+id/signup_confirm_password_field"  
android:layout_width="0dp"  
android:layout_height="wrap_content"  
android:layout_weight="3"  
android:hint="@string/ConfirmPassword"  
android:inputType="textPassword"  
android:textSize="@dimen/text_size"  
android:textStyle="bold" />
```

<Button

```
android:id="@+id/signin_button"  
android:layout_width="0dp"  
android:layout_height="wrap_content"  
android:layout_weight="1"  
android:onClick="SignupCall"  
android:text="@string/SignUp"  
android:textColor="@color/Red"  
android:textSize="@dimen/text_size"  
android:textStyle="bold" />
```

</LinearLayout>

6.3 Περιγραφή Λειτουργιών

6.3.1 Χειρισμός διεπαφής χρήστη (UI) λειτουργίας Sign Up

Ο χειρισμός των δεδομένων που εισήγαγε ο χρήστης γίνεται μέσω της μεθόδου SignupCall. Όπως αναφέρθηκε στην προηγούμενη ενότητα έχουμε συνδέσει το πάτημα του κουμπιού για την αποστολή των δεδομένων με την προαναφερόμενη μέθοδο μέσω του element android:onClick στο layout.

Η μέθοδος έχει δύο τοπικές μεταβλητές τύπου boolean, την filled και την passwordMatch, οι οποίες αρχικοποιούνται σε false. Η filled αντιστοιχεί στην λογική κατάσταση όπου όλα τα πεδία EditText έχουν συμπληρωθεί. Αυτό δεν ισχύει αρχικά. Κάνουμε σειριακά έλεγχο για κάθε ένα από τα τέσσερα πεδία, και αν κάποιο από αυτά είναι κενό ενημερώνουμε τον χρήστη ώστε να το συμπληρώσει. Αν όλα είναι συμπληρωμένα τότε η μεταβλητή filled τίθεται true.

Εν συνεχεία κάνουμε έλεγχο στις τιμές που έχει δώσει ο χρήστης στα πεδία password και confirmPassword. Οι δύο τιμές συγκρίνονται και αν είναι ίδιες η μεταβλητή passwordMatch τίθεται και αυτή σε τιμή true.

Αν και οι δύο τιμές είναι true τότε τα πεδία έχουν συμπληρωθεί σωστά οπότε μπορούμε να πάρουμε τα στοιχεία και να τα χρησιμοποιήσουμε στο API call. Υλοποιούμε ένα αντικείμενο της κλάσης SignupCall και ξεκινάμε την ασύγχρονη εκτέλεση παίρνοντας στην μέθοδο .execute() τις τιμές των EditText.

```
public void SignupCall (View view){
```

```
    Boolean filled = false;
```

```
    Boolean passwordMatch = false;
```

```
    EditText et1 = (EditText) findViewById(R.id.singup_email_field);
```

```
    EditText et2 = (EditText) findViewById(R.id.singup_name_field);
```

```

EditText et3 = (EditText) findViewById(R.id.singup_password_field);
EditText et4 = (EditText) findViewById(R.id.singup_confirm_password_field);
if (et1.getText().toString().length() == 0)
    Toast.makeText(getApplicationContext(), "Please provide your email address.",
        Toast.LENGTH_SHORT).show();

else if (et2.getText().toString().length() == 0)
    Toast.makeText(getApplicationContext(), "Please provide your Full Name.",
        Toast.LENGTH_SHORT).show();

else if (et3.getText().toString().length() == 0)
    Toast.makeText(getApplicationContext(), "Please provide a password.",
        Toast.LENGTH_SHORT).show();

else if (et4.getText().toString().length() == 0)
    Toast.makeText(getApplicationContext(), "Please provide the confirmation of the
        password.", Toast.LENGTH_SHORT).show();

else
    filled = true;

if (et3.getText().toString().compareTo(et4.getText().toString()) != 0){
    Toast.makeText(getApplicationContext(), "The password and the confirmation password
        do not match.", Toast.LENGTH_SHORT).show();

    passwordMatch = false;
}
else
    passwordMatch = true;
if (filled && passwordMatch){
    SignupCall suc = new SignupCall(mContext,activity);
    suc.execute(et1.getText().toString(), et2.getText().toString(), et3.getText().toString(),
        et4.getText().toString());
}
}
}

```

6.3.2 Κλήση Sign up (API Call)

Αφού έχουμε πάρει τα δεδομένα από τον χρήστη, συνεχίζουμε με την κλήση στο API της υπηρεσίας μας. Η κλήση χρησιμοποιεί μέθοδο HTTP POST με αποστολή αντικειμένου json, με key "user" και value ένα δεύτερο αντικείμενο JSON.

Το δεύτερο αντικείμενο έχει τέσσερα ζευγάρια key-value, τα τέσσερα δεδομένα που ζητήσαμε από το χρήστη. Αρχικά δημιουργούμε το πρώτο αντικείμενο και χρησιμοποιούμε την μέθοδο put() για να προσθέσουμε τις τιμές. Έπειτα περνάμε το αντικείμενο αυτό στο δεύτερο JSON σαν value, με key “user” όπως αναφέρθηκε.

Κάνουμε την κλήση στο server και αναμένουμε την απάντηση. Ελέγχουμε τα πεδία στο αντικείμενο JSON που μας επιστράφηκε, συγκεκριμένα την τιμή στο key “status” για το αν η κλήση ήταν επιτυχής. Αν δεν ήταν ενημερώνεται ο χρήστης για το πρόβλημα (που συνήθως θα έχει να κάνει με την σύνδεση στο διαδίκτυο, αν έχουμε υλοποιήσει σωστά τον κώδικα).

Σε περίπτωση επιτυχίας ο χρήστης ενημερώνεται για να συνδεθεί με τον νέο λογαριασμό στην υπηρεσία.

```
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicHeader;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.protocol.HTTP;
import org.json.JSONObject;

import android.app.ProgressDialog;
import android.content.Context;
import android.os.AsyncTask;
import android.util.Log;
import android.widget.Toast;

public class SignupCall extends AsyncTask<String, Integer, String >{
```

```

private ProgressDialog dialog;
private Context mContext;
private String status;

public SignupCall(Context c,Activity activity){
    this.mContext = c;
}

@Override
protected String doInBackground(String... params) {
    try{
        HttpPost httpPost = new HttpPost(Properties.server + "/signup.json");
        // Depends on your web service
        JSONObject json = new JSONObject();
        JSONObject json2 = new JSONObject();
        json2.put("password_confirmation",params[3]);
        json2.put("password",params[2]);
        json2.put("name",params[1]);
        json2.put("email",params[0]);
        json.put("user",json2);
        StringEntity se = new StringEntity(json.toString());
        se.setContentType("application/json");
        se.setContentEncoding(new BasicHeader(HTTP.CONTENT_TYPE,
                                                    "application/json"));

        httpPost.setEntity(se);
        httpPost.setHeader("Content-type", "application/json");
        httpPost.setHeader("User-agent", "Android machine");
        httpPost.setHeader("Accept", "application/json");
        InputStream inputStream = null;
        HttpParams httpParameters = new BasicHttpParams();
        // Set the timeout in milliseconds until a connection is established.
        // The default value is zero, that means the timeout is not used.
        int timeoutConnection = 10000;
        HttpConnectionParams.setConnectionTimeout(httpParameters, timeoutConnection);
        // Set the default socket timeout (SO_TIMEOUT)
        // in milliseconds which is the timeout for waiting for data.
        int timeoutSocket = 10000;
        HttpConnectionParams.setSoTimeout(httpParameters, timeoutSocket);
    }
}

```

```

// create object of DefaultHttpClient
DefaultHttpClient httpClient = new DefaultHttpClient(httpParameters);
HttpResponse response = httpClient.execute(httpPost);
HttpEntity entity = response.getEntity();
InputStream inputStream = entity.getContent();
BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream,
                                                                    "UTF-8"), 8);

StringBuilder sb = new StringBuilder();
String line = null;
while ((line = reader.readLine()) != null)
{
    sb.append(line + "\n");
}
JSONObject loginResponse = new JSONObject(sb.toString());
status = loginResponse.toString();
Log.d("Signup", status);
return status;
}catch(Exception e){
    e.printStackTrace();
    return e.toString();
}
}

@Override
protected void onPreExecute() {
    dialog = ProgressDialog.show(mContext, null, "Loading...");
}

@Override
protected void onPostExecute(String result) {
    dialog.dismiss();
    if (!result.equals("Success")){
        int duration = Toast.LENGTH_LONG;
        Toast toast = Toast.makeText(mContext, "There was the following error trying to
                                                                    Signup: " + result, duration);

        toast.show();
    }
    else{

```

```
    if (status.equals("Success"))
        Toast.makeText(mContext, "Account successfully created! You may now Sign In.",
                        Toast.LENGTH_LONG).show();
    }
}
```


7

Λειτουργία Wishlist

Το τελευταίο υποσύστημα που υλοποιήσαμε είναι η λειτουργία wishlist, δηλαδή μία λίστα προϊόντων για τα οποία ο χρήστης ενδιαφέρεται, αλλά δεν προτίθεται να τα αγοράσει άμεσα.

7.1 Αρχιτεκτονική

Η λειτουργικότητα που αναμένεται είναι η ακόλουθη. Η δυνατότητα να υπάρχει σε κάθε προϊόν ένας τρόπος να το προσθέσουμε σε μία λίστα, και η λίστα αυτή να είναι διαθέσιμη στο χρήστη να την δει όποτε το επιθυμεί.

Αρχικά χρειάστηκε η προσθήκη ενός Button σε κάθε σελίδα προϊόντος το οποίο θα προσθέτει το αντικείμενο στη λίστα του χρήστη. Η λίστα θα αποθηκεύεται τοπικά στη συσκευή. Το android προσφέρει τη δυνατότητα αποθήκευσης ζευγαριών key-value με την κλάση SharedPreferences. Θα αποθηκευθεί ο μοναδικός κωδικός του προϊόντος σαν key και το όνομα του σαν value.

Η μέθοδος που χειρίζεται το πάτημα του Button κάνει έλεγχο αν το προϊόν υπάρχει ήδη στη λίστα του χρήστη. Αν υπάρχει τότε το πάτημα το αφαιρεί.

Τέλος δημιουργήθηκε ένα tab το οποίο εμφανίζει τη λίστα. Βασίζεται σε ένα GridView, το οποίο παίρνει τα δεδομένα από το αρχείο που αποθηκεύσαμε τα key-values.

7.2 SharedPreferences

Η κλάση SharedPreferences δημιουργήθηκε από το android για το χειρισμό των δεδομένων που επιστρέφονται από την μέθοδο `getSharedPreferences(String, int)`. Τα δεδομένα αυτά συνήθως αποτελούν προτιμήσεις της εφαρμογής, οι οποίες πρέπει να διατηρηθούν αφού κλείσει η εφαρμογή, οπότε δεν μπορεί να είναι αποθηκευμένες στην μνήμη. Η μέθοδος παίρνει δύο ορίσματα, το πρώτο είναι ένα String που δηλώνει το όνομα του αρχείου στο οποίο θα αποθηκευτούν οι τιμές και το δεύτερο είναι μία σταθερά που έχει να κάνει με τα δικαιώματα που θα έχει η εφαρμογή πάνω στο αρχείο.

Η συνήθης δήλωση της δευτερης παραμέτρου είναι η σταθερά `MODE_PRIVATE` που σημαίνει ότι η εφαρμογή μπορεί να διαβάσει και να γράψει το αρχείο, και η πρόσβαση στο αρχείο γίνεται μόνο από αυτή την εφαρμογή. Οι άλλες τρεις τιμές που μπορεί να πάρει αυτή η παράμετρος είναι `MODE_WORLD_READABLE`, `MODE_WORLD_WRITEABLE` και `MODE_MULTI_PROCESS`.

Οι τιμές `MODE_WORLD_READABLE` και `MODE_WORLD_WRITEABLE` δηλώνουν ότι το αρχείο προτιμήσεων μπορεί να χρησιμοποιηθεί και από άλλες εφαρμογές είτε μόνο για ανάγνωση, είτε και για εγγραφή. Συνιστάται να αποφεύγονται αυτές οι τιμές για λόγους ασφάλειας.

Τέλος η τρίτη τιμή `MODE_MULTI_PROCESS` επιτρέπει την πρόσβαση στο αρχείο από πολλαπλές διεργασίες της εφαρμογής παράλληλα. Υπάρχουν περιπτώσεις που αυτό είναι επιθυμητό, αλλά πρέπει να γνωρίζουμε ότι αυτό μπορεί να αλλάξει την κατάσταση των τιμών του αρχείου με μη ντετερμινιστικό τρόπο οπότε όταν χρησιμοποιείται πρέπει να είμαστε σίγουροι ότι μπορούμε να καθορίσουμε την ακριβή συμπεριφορά που αναμένουμε από την εφαρμογή.

Αφού δημιουργηθεί ένα αντικείμενο της κλάσης SharedPreferences, καλούμε την μέθοδο `getSharedPreferences()` και αναθέτουμε την επιστρεφόμενη τιμή στο αντικείμενο.

Όπως αναφέρθηκε δεν πρέπει να χειριζόμαστε το αρχείο προτιμήσεων απευθείας για να μπορούμε να είμαστε σίγουροι ότι τα

δεδομένα μιας διεργασίας θα αποθηκευτούν πριν κάποια άλλη διεργασία μπορεί να τα χρησιμοποιήσει. Για αυτό το σκοπό το android υποστηρίζει την ανάγνωση και εγγραφή των δεδομένων στο αρχείο μέσω της κλάσης editor της SharedPreferences.

Η κλάση editor είναι στην ουσία ένα buffer στο οποίο αποθηκεύονται όλες οι αλλαγές που θέλουμε να κάνουμε στο αρχείο, και υλοποιούνται μόνο όταν καλέσουμε τη μέθοδο commit().

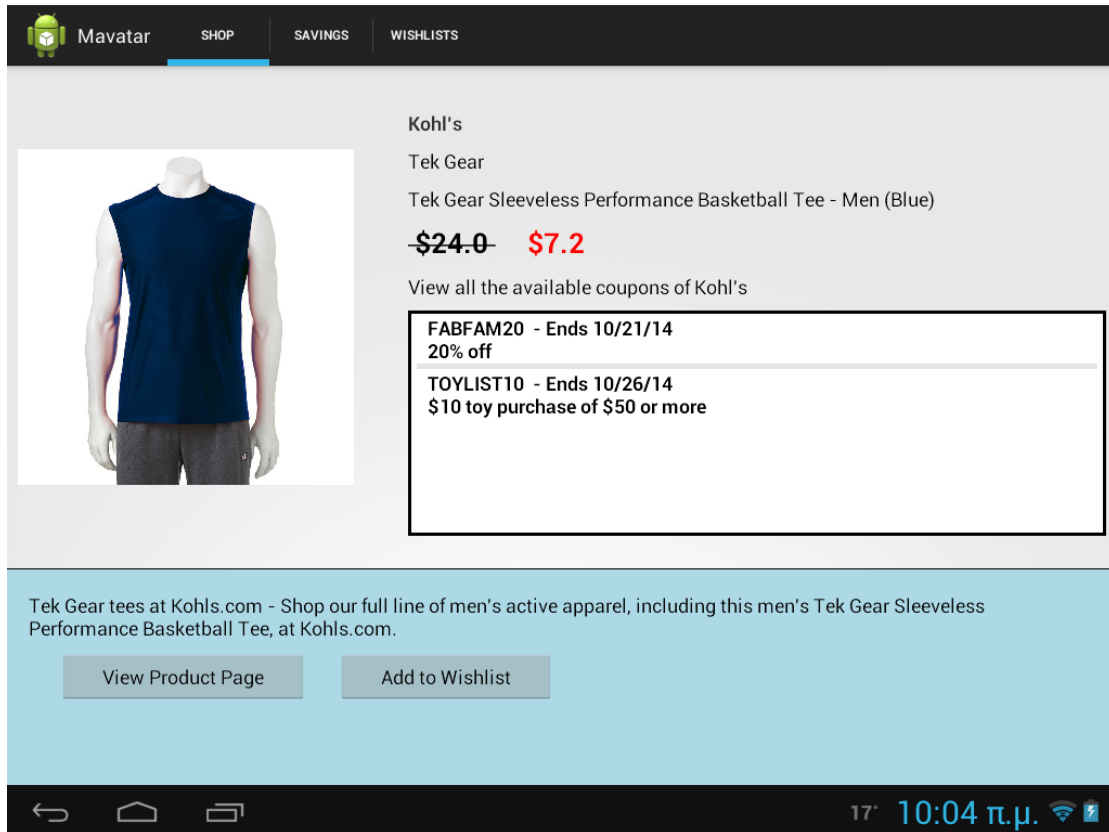
7.3 Περιγραφή Λειτουργιών

7.3.1 Κουμπί “Add to Wishlist”

Όπως αναφέρθηκε στην προηγούμενη ενότητα απαιτούνται δύο αντικείμενα, ένα της κλάσης SharedPreferences και ένα της υποκλάσης SharedPreferences.Editor. Δημιουργούμε αρχικά το πρώτο αντικείμενο (sharedPref) καλώντας την μέθοδο getSharedPreferences, με το string με id wishlist_file. Η μέθοδος ελέγχει αν υπάρχει το αρχείο και αν δεν υπάρχει, το δημιουργεί.

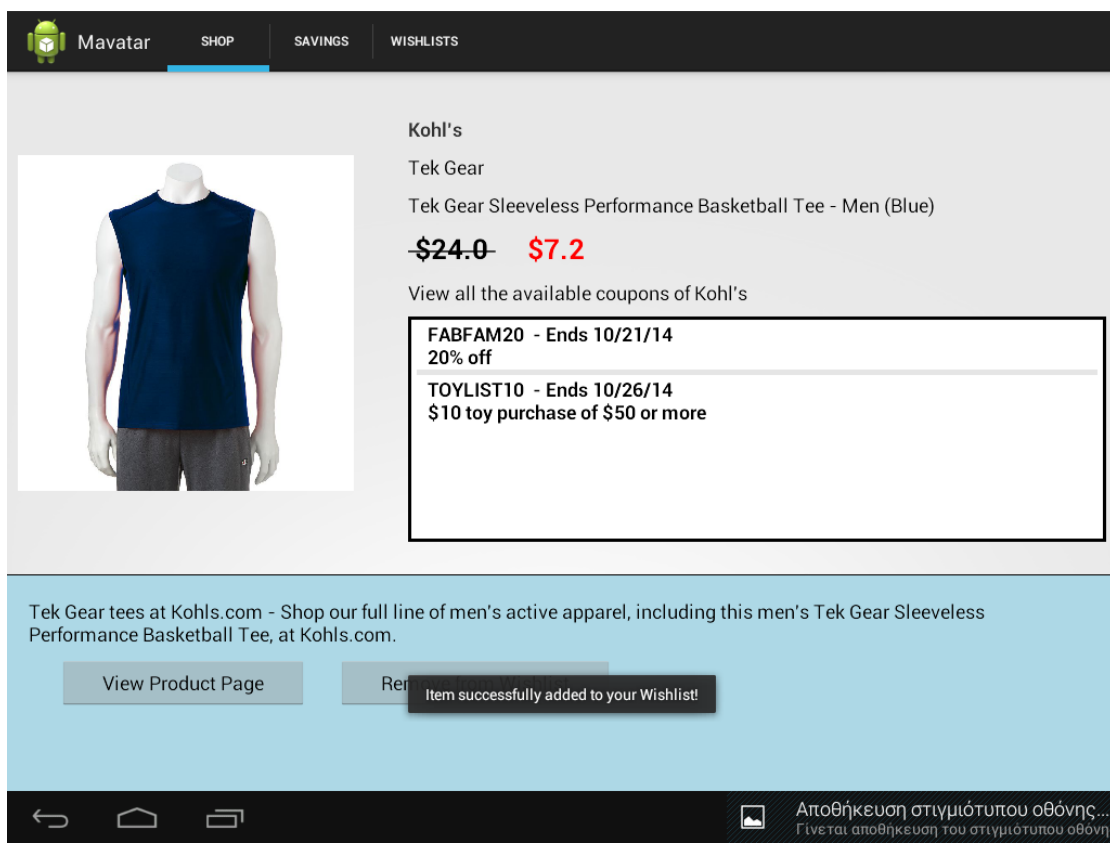
Έπειτα δημιουργείται το δεύτερο αντικείμενο (editor) καλώντας την μέθοδο edit() στο αντικείμενο sharedPref.

Εν συνεχεία χρησιμοποιείται η μέθοδος getViewById για να πάρουμε ένα αντικείμενο του View Button το οποίο θα χρησιμοποιήσουμε για να θέσουμε τον clickListener ο οποίος θα χειρίζεται τη λειτουργικότητα που θέλουμε κατά το πάτημα του Button.



Εικ. 7.1 To Button Add to Wishlist

Γίνεται ένας πρώτος έλεγχος στο προϊόν για να ελεγχθεί να βρίσκεται ήδη στη λίστα του χρήστη. Αν υπάρχει, θέτουμε το κείμενο στο κουμπί σε “Remove from Wishlist” δηλαδή αφαίρεση από τη λίστα του χρήστη. Σε διαφορετική περίπτωση το κείμενο διατηρείται σε “Add to Wishlist”.

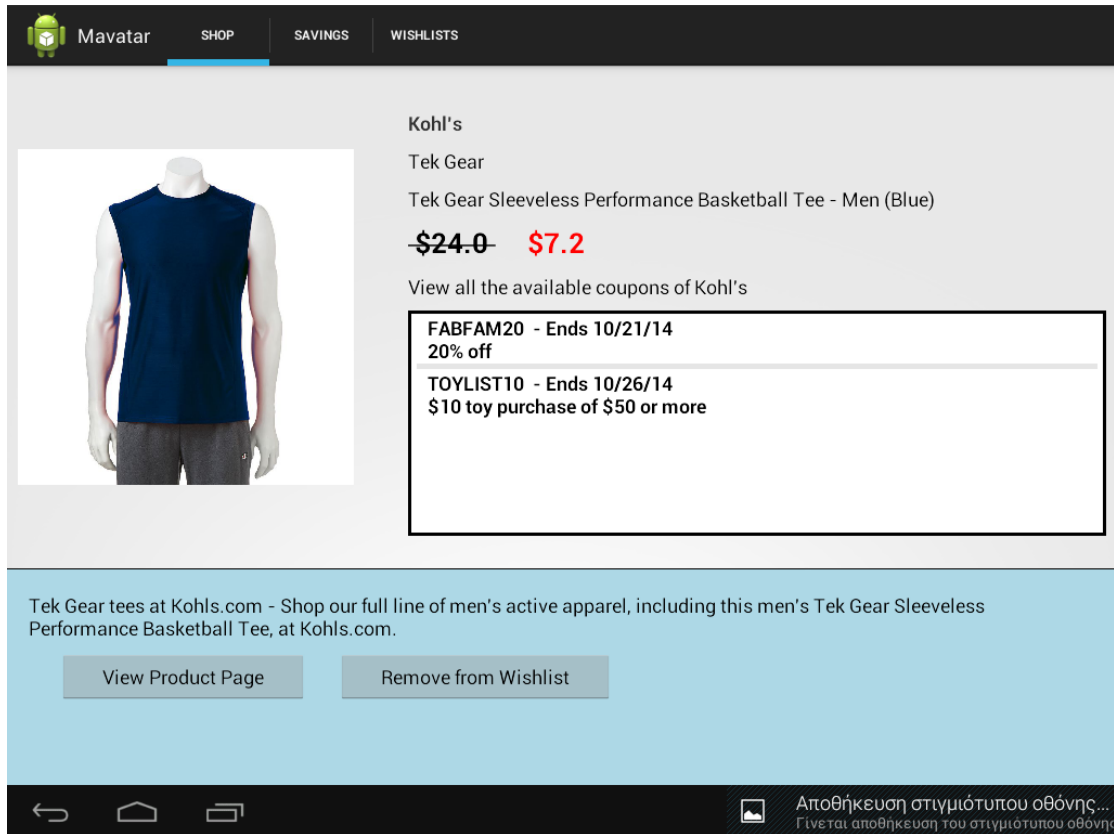


Εικ. 7.2 Η εφαρμογή μας ενημερώνει για τη επιτυχή προσθήκη προϊόντος

Τέλος τίθεται ο clickListener πάνω στο Wishlist Button και κάνοντας override την onClick μέθοδο, χρησιμοποιούνται οι μέθοδοι put(id, name) και remove(id) για να προτεθούν ή να αφαιρεθούν οι πληροφορίες για το συγκεκριμένο προϊόν.

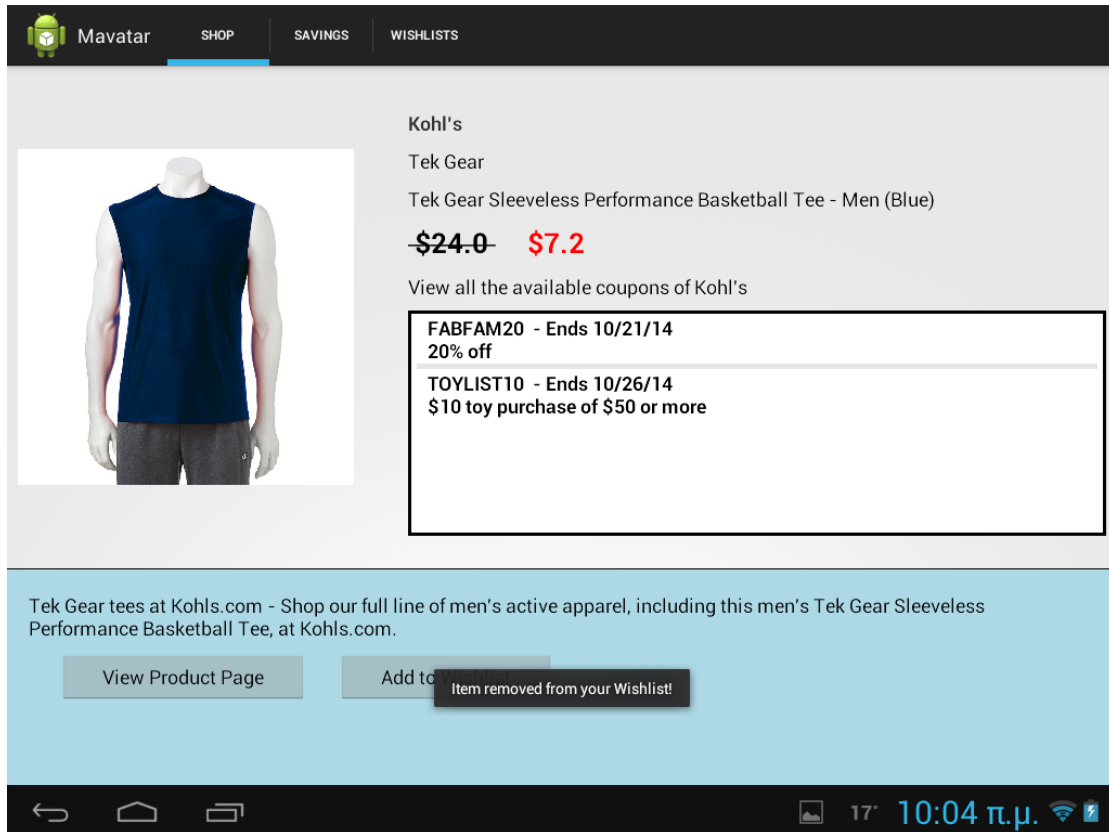


Εικ. 7.3 Η Wishlist μετά από επιτυχή προσθήκη προϊόντος



Εικ. 7.4 Το Button Remove from Wishlist για προϊόν που υπάρχει στη λίστα.

Αφού τελειώσει αυτή η μέθοδος καλείται η `commit()` για να εφαρμόσει τις αλλαγές από τον editor στο αρχείο, και ενημερώνεται ο χρήστης με ένα μήνυμα για την επιτυχία της ενέργειας, καθώς επίσης αλλάζει και το κείμενο του Button αναλόγα με την περίπτωση.



Εικ. 7.5 Η εφαρμογή μας ενημερώνει για την αφαίρεση προϊόντος από τη λίστα.

```
Context mContext = getActivity();
//We create the file to store the wishlist data
SharedPreferences sharedPref = mContext.getSharedPreferences
    (getString(R.string.wishlist_file),
    Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();

existsInWishlist = (!(sharedPref.getString(id.toString(),"noval").equals("noval")));
```

```

addwishlistbutton = (Button) getActivity().findViewById(R.id.AddToWishlistButton);

if (existsInWishlist)
    addwishlistbutton.setText("Remove from Wishlist");
else
    addwishlistbutton.setText("Add to Wishlist");

/*Setting up the Wishlist button functionality*/
addwishlistbutton.setOnClickListener(new View.OnClickListener(){

    @Override
    public void onClick(View v) {

        if (!existsInWishlist)
        {
            editor.putString(id.toString(), name.toString());
            editor.commit();
            Toast.makeText(mContext, "Item successfully added to your Wishlist!",
                Toast.LENGTH_SHORT).show();

            addwishlistbutton.setText("Remove from Wishlist");
            existsInWishlist = true;
        } else {
            editor.remove(id.toString());
            editor.commit();
            Toast.makeText(mContext, "Item removed from your Wishlist!",
                Toast.LENGTH_SHORT).show();

            addwishlistbutton.setText("Add to Wishlist");
            existsInWishlist = false;
        }
    }
});

```

7.3.2 Wishlist Fragment

Τέλος για να ολοκληρωθεί το υποσύστημα δημιουργήθηκε ένα fragment το οποίο αναλαμβάνει να διαβάσει τα δεδομένα από το αρχείο προτιμήσεων και να τα παρουσιάσει στο χρήστη.

Το fragment αποτελείται από την κλάση (η οποία κάνει extend την κλάση fragment) και από το layout. Το layout .xml περιγράφεται στην επόμενη ενότητα. Η κλάση που δημιουργήθηκε έχει δύο βασικές μεθόδους, την onCreateView() η οποία αναλαμβάνει στην οποία ορίζεται το layout που θα χρησιμοποιηθεί, και την onActivityCreated() η οποία καλείται την πρώτη φορά που δημιουργείται το fragment.

Για την προβολή της λίστας των προϊόντων χρησιμοποιήθηκε ένα GridView. Το GridView είναι ένας τύπος προβολής δεδομένων βασισμένος σε ένα πλέγμα. Τα δεδομένα προβάλλονται στα κελιά του πλέγματος, τα οποία είναι ίδιου μεγέθους και επιλέγεται αυτόματα από το android ώστε να χωράνε στην οθόνη κατά πλάτος όσα κελιά έχουμε δηλώσει.

Συμπληρωματικά στο πλέγμα δημιουργείται ένα αντικείμενο της κλάσης ArrayAdapter. Οι Adapters είναι αντικείμενα τα οποία αναλαμβάνουν την οργάνωση των δεδομένων, που θα παρουσιαστούν.

Για την περίπτωση της λίστας χρησιμοποιήθηκε η κλάση που αναφέρθηκε ArrayAdapter, η οποία δίνεται από το android για προβολή απλού τύπου δεδομένων, συνήθως μία λίστα από strings. Ο constructor της κλάσης έχει τρία ορίσματα. Το πρώτο είναι το context της εφαρμογής, το δεύτερο είναι ένα id που προσδιορίζει το View που θα χρησιμοποιηθεί για την προβολή κάθε string, και το τρίτο όρισμα είναι μία λίστα με τα string προς παρουσίαση.

Από τα τρία ορίσματα ιδιαίτερο ενδιαφέρον έχει το δεύτερο. Στην εφαρμογή χρησιμοποιήθηκε ένα View το οποίο παρέχεται από το android για την προβολή απλών δεδομένων που δεν χρειάζονται ιδιαίτερη παρουσίαση. Το View προσδιορίζεται από το id android.R.layout.simple_list_item_1, και προσδιορίζει ένα απλό TextView στο οποίο περνάνε τα δεδομένα της λίστας.

Η δύναμη των adapters στο android βρίσκεται στην παραμετροποίηση αυτών των Views. Τα Views δημιουργούνται σε ένα layout αρχείο και προσδιορίζονται από το element android:id κατά τη σύμβαση του android.

Επιπλέον για κάθε πιο περίπλοκο View που θέλουμε να χρησιμοποιήσουμε πρέπει να δημιουργήσουμε και μία κλάση η οποία θα είναι υποκλάση της BaseAdapter.

Η κλάση αυτή θα περιγράφει πως τα δεδομένα που έχουμε, τα οποία για κάθε κελί μπορεί να είναι μία εικόνα, ένα κείμενο, και οποιοδήποτε άλλο δεδομένο μπορεί να προβληθεί με τα βασικά Views του android, θα αντιστοιχηθούν στα πεδία του View που έχουμε δημιουργήσει.

Στην εφαρμογή που δημιουργήθηκε, επειδή το μόνο από τα δεδομένα που θέλουμε να παρουσιάζεται στο χρήστη είναι το όνομα του προϊόντος δεν θεωρήθηκε αναγκαία η δημιουργία ξεχωριστού adapter.

Τέλος το GridView που χρησιμοποιήθηκε δηλώνεται σε ένα αρχείο layout, “κάτω” από ένα LinearLayout, δηλώνοντας τις επιθυμητές παραμέτρους, όπως το χρώμα του background, τον αριθμό των κελιών σε κάθε σειρά, και παραμέτρους σχετικές με τα κενά γύρω από κάθε κελί.

Όσον αφορά τον κώδικα του fragment αναλαμβάνει να ενοποιήσει τα τμήματα που αναφέραμε, δηλώνεται καταρχήν το layout (στην περίπτωση της εφαρμογής, το wishlist_fragment) μέσω του inflater, στη μέθοδο onCreateView.

Έπειτα δηλώνουμε την λειτουργικότητα στην μέθοδο OnActivityCreated. Δημιουργείται ένα αντικείμενο ArrayList με όνομα values, το οποίο θα περιέχει τα ονόματα των προϊόντων. Δημιουργείται επίσης ένα αντικείμενο τύπου Map<String, ?> το οποίο είναι ένα ζευγάρι key-value το οποίο επιτρέπει όμως το τύπος της τιμής (value) να μην είναι γνωστός κατά εκ των προτέρων αλλά κατά την εκτέλεση.

Τα δεδομένα σε αυτή την δομή είναι τελικά τα προϊόντα που έχει επιλέξει ο χρήστης και τα οποία θα αποτυπωθούν τελικά στην εφαρμογή.

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
```

```

import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.GridView;

```

```

public class WishlistFragment extends Fragment {

```

```

    private Context mContext;
    private SharedPreferences sharedPref;
    private GridView grid;

```

```

    @Override

```

```

    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.wishlist_fragment, container, false);
    }

```

```

    @Override

```

```

    public void onActivityCreated (Bundle savedInstanceState)

```

```

    {
        super.onActivityCreated(savedInstanceState);
        grid = (GridView) getActivity().findViewById(R.id.wish);
        mContext = getActivity();

        ArrayList<String> values = new ArrayList<String>();
        //We create the file to store the wishlist data
        sharedPref = mContext.getSharedPreferences( getString(R.string.wishlist_file),
                                                    Context.MODE_PRIVATE);

        Map<String, ?> wishlistItems = sharedPref.getAll();
        for(Map.Entry<String, ?> entry : wishlistItems.entrySet())
        {
            values.add(entry.getValue().toString());
        }
    }

```

```

    ArrayAdapter<String> adapter = new ArrayAdapter<String>(getActivity(),
        android.R.layout.simple_list_item_1, values);

    grid.setAdapter(adapter);
}
}

```

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="8dp"
    android:paddingRight="8dp"
    android:background="@color/LightBlue">
    <GridView android:id="@+id/wish"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/margin_size_small"
        android:columnWidth="@dimen/margin_size_xlarge"
        android:gravity="center"
        android:horizontalSpacing="@dimen/margin_size_small"
        android:numColumns="3"
        android:paddingBottom="@dimen/margin_size_medium"
        android:paddingLeft="@dimen/margin_size_large"
        android:paddingRight="@dimen/margin_size_large"
        android:paddingTop="@dimen/margin_size_medium"
        android:stretchMode="columnWidth"
        android:verticalSpacing="@dimen/margin_size_medium">
    </LinearLayout>

```

8

Επίλογος

Έχοντας ολοκληρώσει την παρουσίαση της εφαρμογής θα παρουσιάσουμε τα συμπεράσματα που αντλήσαμε.

8.1 Σύνοψη

Συνοψίζοντας την εφαρμογή, δημιουργήθηκαν τέσσερα βασικά συστήματα που είναι απαραίτητα σε κάθε εφαρμογή ηλεκτρονικού εμπορίου. Τα τρία από τα συστήματα έχουν να κάνουν με τον λογαριασμό του χρήστη.

Αρχικά να αναφερθεί ότι η δυνατότητα να χρησιμοποιήσει κάποιος τον λογαριασμό του χρήστη στην εφαρμογή, ανοίγει μεγάλους ορίζοντες για τον δημιουργό. Τη στιγμή που γράφεται η διπλωματική το facebook είναι το μεγαλύτερο κοινωνικό δίκτυο που υπάρχει στον πλανήτη αριθμώντας πάνω από ένα δισεκατομμύριο χρήστες.

Οι προβλέψεις για το μέλλον αναφέρουν ότι ο αριθμός αυτός αναμένεται να αυξηθεί καθώς όλο και περισσότεροι άνθρωποι στις αναπτυσσόμενες χώρες αποκτούν πρόσβαση στο διαδίκτυο.

Η χρήση του κοινωνικού στοιχείου σε μία εφαρμογή δημιουργεί τις προϋποθέσεις για την γρήγορη εξάπλωση της χρήσης της καθώς κάθε χρήστης είναι εν δυναμει διαφημιστής. Έτσι το πρώτο βήμα είναι να χρησιμοποιηθεί το facebook login για να συνδεθεί ο χρήστης στην

υπηρεσία μας, καθώς να μπορούμε να έχουμε τα στοιχεία που επιθυμούμε για αυτόν. Επίσης μας δίνει τη δυνατότητα να εισάγουμε περισσότερα κοινωνικά στοιχεία στην εφαρμογή σαν επέκταση στο μέλλον όπως για παράδειγμα την δυνατότητα να σχολιάζουν και να δίνουν την άποψη τους οι φίλοι του χρήστη σε κάποιο προϊόν.

Κατά δεύτερον η δυνατότητα χρήσης ενός προσωρινού λογαριασμού για κάποιον χρήστη του δίνει τη δυνατότητα να δοκιμάσει γρήγορα την εφαρμογή, χωρίς να χρειάζεται να περάσει κάποια λεπτά δίνοντας τα στοιχεία του σε μία υπηρεσία για την οποία δεν είναι σίγουρος αν θα καταλήξει να τη χρησιμοποιεί, καθώς στη σημερινή εποχή όπου τα ερεθίσματα στο διαδίκτυο έρχονται με τεράστια ταχύτητα, κάθε καθυστέρηση στη χρήση της υπηρεσίας μπορεί να είναι κρίσιμη για την προσέλκυση ή μη του χρήστη.

Κατά τρίτον υλοποιήθηκε η δυνατότητα δημιουργία ξεχωριστού λογαριασμού για τους χρήστες που το επιθυμούν και δεν έχουν ή δεν θέλουν να χρησιμοποιήσουν το λογαριασμό τους στο facebook, καθώς υπάρχει μια μεγάλη μερίδα χρηστών οι οποίοι ανησυχούν για τα προσωπικά τους δεδομένα και αυτό τους προκαταβάλει αρνητικά στη χρήση του λογαριασμού τους στο facebook σε οποιαδήποτε άλλη υπηρεσία.

Τέλος το τελευταίο υποσύστημα που υλοποιήθηκε είναι μία λειτουργικότητα για την δημιουργία wishlist. Οι λίστες αυτές είναι λίστες προϊόντων για τα οποία ο χρήστης ενδιαφέρεται αλλά για διάφορους λόγους δεν επιθυμεί την άμεση αγορά τους. Έτσι είμαστε υποχρεωμένοι να του δίνουμε τη δυνατότητα να τα αποθηκεύει για μελλοντική χρήση.

8.2 Μελλοντικές επεκτάσεις

Βάσει αυτών που αναφέρθηκαν υπάρχουν πολλές δυνατότητες για μελλοντικές επεκτάσεις στη χρήση του κοινωνικού στοιχείου της εφαρμογής αλλά και στη λειτουργία wishlist.

Αρχικά το κοινωνικό στοιχείο μέσω της χρήσης του λογαριασμού facebook του χρήστη, επιτρέπει την δημιουργία εμπειρίας η οποία θα επεκτείνεται από τον χρήστη στους φίλους του. Εισάγωντας στοιχεία όπως η δυνατότητα να σχολιάζουν ή να ψηφίζουν οι φίλοι του χρήστη σε κάποιο προϊόν, αυξάνουμε την πιθανότητα κάποιος από αυτούς να

δοκιμάσει την υπηρεσία, ενώ ταυτόχρονα ο χρήστης περνά περισσότερο χρόνο στην υπηρεσία, σε σχέση με την μη κοινωνική χρήση της εφαρμογής.

Στην λειτουργία προσωρινού λογαριασμού του χρήστη, μπορεί να προστεθεί η δυνατότητα να συνδεθεί η χρήση που έχει κάνει με ένα νέο λογαριασμό, είτε απευθείας στην εφαρμογή μας, είτε στον λογαριασμό facebook. Η μέχρι τώρα υλοποίηση δεν υποστηρίζει αυτή τη λειτουργία, με αποτέλεσμα κάποιος χρήστης ο οποίος δοκίμασε την εφαρμογή και αποφάσισε ότι του αρέσει η υπηρεσία θέλει να συνεχίσει να τη χρησιμοποιεί, καλείται να δημιουργήσει νέο λογαριασμό χάνοντας ότι έχει κάνει ως τώρα.

Αυτό θα μπορούσε να υλοποιηθεί προσθέτοντας ένα νέο tab στην εφαρμογή το οποίο θα περιέχει τις πληροφορίες του χρήστη, και δυνατότητα σύνδεσης του τρέχοντος προσωρινού λογαριασμού με το facebook του χρήστη ή και με ξεχωριστό λογαριασμό ο οποίος θα μπορεί να δημιουργηθεί στην ίδια καρτέλα.

Τέλος η λειτουργία wishlist έχει επίσης μεγάλες δυνατότητες επέκτασης. Αρχικά μπορεί να συνδεθεί η λίστα προϊόντων απευθείας στο server της υπηρεσίας. Κάθε φορά που θα πατάμε το κουμπί για προσθήκη στη λίστα σε κάποιο προϊόν θα γίνεται κάποια κλήση στο server η οποία θα το αποθηκεύει εκεί. Επιπλέον έτσι θα μπορούμε να λαμβάνουμε τη λίστα με απευθείας κλήση και να προβάλλουμε τα προϊόντα με περισσότερες πληροφορίες όπως η εικόνα του προϊόντος, η τιμή του κλπ.

Επίσης μπορούμε να συνδυάσουμε τη λειτουργία αυτή με το λογαριασμό facebook του χρήστη, και να τη χρησιμοποιήσουμε για παράδειγμα προσφέροντας στους φίλους του χρήστη μία λίστα δώρων για τα γενέθλια του.

Όπως φαίνεται η εφαρμογή έχει μεγάλες δυνατότητες επέκτασης. Παρ' όλα αυτά στο πλαίσιο μιας διπλωματικής εργασίας ήταν μία πολύ καλή ευκαιρία για την εκμάθηση και χρήση αρκετών από τις δυνατότητες που προσφέρει το android σαν πλατφόρμα και η εμπειρία που αποκομίσθηκε θα είναι χρήσιμη για επαγγελματική εργασία πάνω στο αντικείμενο.

9

Βιβλιογραφία

- [1] **Mobile/Tablet Operating System Market Share**
<http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qptimeframe=M>
- [2] **Facebook SDK**
<https://developers.facebook.com/>
- [3] **Android Developer Training**
<http://developer.android.com/training/index.html>
- [4] **Android API Guides**
<http://developer.android.com/guide/index.html>
- [5] **Android Package Reference**
<http://developer.android.com/reference/packages.html>
- [6] **Professional Android 4 Application Development**
ISBN-10: 1118102274
- [7] **Using lists in Android (ListView) - Tutorial**
<http://www.vogella.com/tutorials/AndroidListView/article.html>
- [8] **«Έξυπνη» εφαρμογή ηλεκτρονικού εμπορίου για Ταμπλέτες με λειτουργικό σύστημα Android**, Χουρδάκης, Στέφανος - Μάριος Γ.; Chourdakis, Stefanos - Marios G.
URI: <http://hdl.handle.net/123456789/38431>