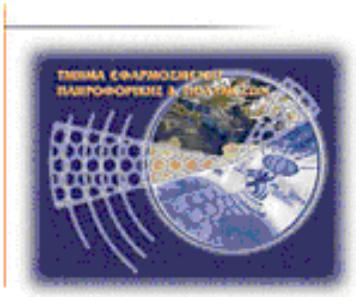
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή Εργασία

Τίτλος:

Σύστημα για ενημέρωση / καθοδήγηση χρηστών Android

Καραντώνη Αγγελα ΑΜ 2621
Χατζηγεωργιάδου Ελισάβετ ΑΜ 2564

Επιβλέπων καθηγητής: Παπαδάκης Νικόλαος

Abstract

The topic of this dissertation is the study, design and development of an application for the Android operating system, which enables a user to identify all domestic airports on a digital map. The user may choose a departure airport and the application will display all possible destinations of all airlines on the map.

For developing the application we will use the open source Android Studio program. Android applications are developed in the Java programming language, while they use XML commands to a large extend, so as to simplify the development of the application's user interface. Indispensable tool for the development of the application is the Android SDKs which includes libraries, the debugger code and the device emulator (emulator), which emulates a virtual Android device on the pc.

Πίνακας περιεχομένων

Ευρετήριο εικόνων

1 Εισαγωγή

1.1 Περίληψη

Το αντικείμενο της πτυχιακής εργασίας είναι η μελέτη, σχεδίαση και ανάπτυξη μιας εφαρμογής για λειτουργικό σύστημα Android, η οποία θα δίνει τη δυνατότητα σε ένα χρήστη να εντοπίζει όλα τα αεροδρόμια της χώρας μέσω ενός ψηφιακού χάρτη. Ο χρήστης μπορεί επίσης να επιλέγει το αεροδρόμιο αναχώρησης και η εφαρμογή θα εμφανίσει όλους τους δυνατούς προορισμούς όλων των εταιριών στο χάρτη.

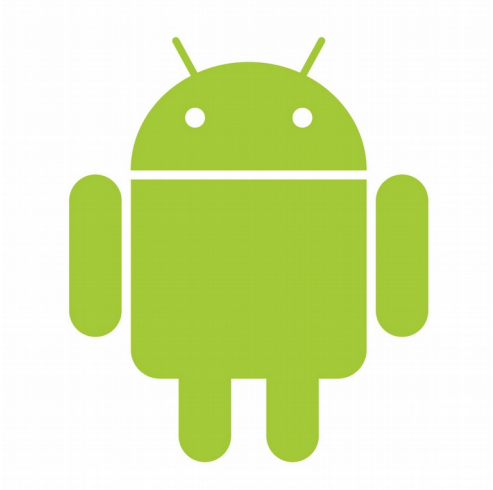
Για την κατασκευή της εφαρμογής θα χρησιμοποιήσουμε το πρόγραμμα ανοιχτού κώδικα Android Studio. Οι εφαρμογές για Android αναπτύσσονται σε γλώσσα προγραμματισμού Java ενώ χρησιμοποιούν XML εντολές, κατά μεγάλο ποσοστό, ώστε να απλοποιηθεί η ανάπτυξη της εφαρμογής στη διεπαφή χρήστη. Απαραίτητο εργαλείο για την ανάπτυξη της εφαρμογής είναι και το Android SDK το οποίο περιλαμβάνει τις βιβλιοθήκες, το πρόγραμμα εντοπισμού σφαλμάτων του κώδικα και τον εξομοιωτή συσκευών (emulator) ο οποίος εξομοιώνει μια εικονική συσκευή Android στον υπολογιστή.

1.2 Εισαγωγή στο Android

1.2.1 Τι είναι το Android

Το Android είναι το πιο διαδεδομένο λογισμικό στον κόσμο, το όνομα του έχει Ελληνική προέλευση και σημαίνει Ανδροείδες δηλαδή ανθρωπόμορφο ρομπότ. Η ιστορία του μετρά πάνω από 6 χρόνια και το ξεκίνημα του έγινε με το HD Dream, από τότε έχουν κατασκευαστεί χιλιάδες android συσκευές. Σε όλο αυτό το χρονικό διάστημα η Google ήταν αυτή που βοήθησε στην εξέλιξή του, δημιουργώντας 10 android εκδόσεις , βασισμένες στον ανοιχτό κώδικα του λογισμικού Linux και με την κάθε μια να προσθέτει και να αναπτύσσει όλο και περισσότερο το λειτουργικό σύστημα . Όλα τα εργαλεία για την κατασκευή μιας Android εφαρμογής διατίθενται δωρεάν και οι εφαρμογές γράφονται στην αντικειμενοστραφή γλώσσα προγραμματισμού Java.

Το android είναι HP σύστημα που συναντάται κυρίως σε ενσωματωμένα συστήματα όπως είναι τα tablet, τα κινητά τηλέφωνα και σε άλλες κινητές συσκευές με οθόνη αφής. Όπως προαναφέρθηκε η Google είναι αυτή που έχει εστιάσει στην εξέλιξη και την ανάπτυξη του πιο γρήγορου συστήματος στον κόσμο, και αυτή η εξέλιξη φαίνεται παρακάτω όπου έχουμε συγκεντρώσει όλες αυτές τις εκδόσεις που εισήγαγαν κάθε φορά και ένα νέο χαρακτηριστικό.



Εικόνα 1: Android Logo

1.2.2 Χαρακτηριστικά Android:

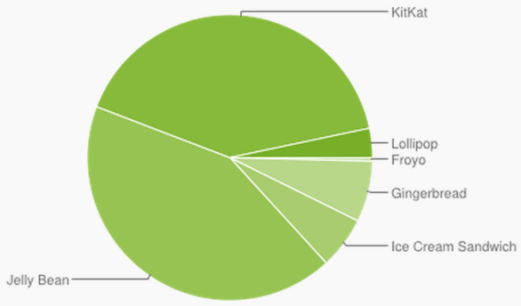
- Υποστηρίζει αρχεία ήχου με επεκτάσεις : WaAV, MPE3, MIDI, AAC. OGG.
- Υποστηρίζει αρχεία στατικής και κινούμενης εικόνας με επεκτάσεις :H.263, H.264, MPEG-4-SP, PNG, JPEG, BMP, GIF.
- Το λογισμικό είναι γραμμένο σε Java και μπορεί να εκτελεστεί στην Dalvik η οποία είναι εικονική μηχανή σχεδιασμένη για χρήση σε φορητές συσκευές.
- Σύστημα διαχείρισης βάσης δεδομένων SQLite.
- Μπορεί να συνεργαστεί με GPS, κάμερα, μαγνητόμετρα, αισθητήρες επιτάχυνσης, δισδιάστατους και τρισδιάστατους επιταχυντές γραφικών.
- Υποστηρίζει τεχνολογία multi-touch.
- Η ανταλλαγή μηνυμάτων γίνεται με SMS και MMS.
- Η αγορά και η εγκατάσταση των εφαρμογών γίνεται απο το Google Play.

1.3 Εκδόσεις και χαρακτηριστικά Android

1.3.1 Εισαγωγή

Η ιστορία του Android ξεκινάει το Νοέμβριο του 2007 και η πρώτη εμπορική έκδοση, Android 1.0 γίνεται το Σεπτέμβριο του 2008. Η εξέλιξη του είναι ραγδαία λόγω του ανοιχτού κώδικα και από τη στιγμή της κυκλοφορίας του έχουν γίνει πολλές αναβαθμίσεις, όλες με σκοπό την διόρθωση διαφόρων προβλημάτων και την προσθήκη νέων πραγμάτων. Οι ονομασίες που δίνονται από τον Απρίλιο του 2009 στις εκδόσεις έχουν ψευδώνυμα που αντιστοιχούν αλφαβητικά σε γλυκά.

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%



Εικόνα 2: Εκδόσεις Android

1.3.2 Android 1.5 Cupcake

Εικόνα 3: Εκδοση Cupcake

Η έκδοση Android 1.5 Cupcake παρουσιάστηκε στις 27 Απριλίου 2009, η οποία ήταν η πρώτη που βασίστηκε στο Linux Kernel 2.6.27. Αρχικά πρέπει να αναφέρουμε πως το χαρακτηριστικό που ανέδειξε το Android 1.5 Cupcake είναι η εισαγωγή ψηφιακού πληκτρολογίου, αφού πριν τα περισσότερα smartphones είχαν φυσικό πληκτρολόγιο QWERTY.

Λειτουργίες Android 1.5 Cupcake :

- Καταγραφή και ανέβασμα βίντεο στο Youtube και το Picasa.
- Καταγραφή και ανέβασμα φωτογραφιών στο Youtube και το Picasa.
- Εγγραφή και αναπαραγωγή βίντεο 3GP και MPEG4.
- Υποστήριξη προτύπου Bluetooth A2DP και AVRCP.
- Αυτόματη σύνδεση σε μικροσυσκευές Bluetooth από συγκεκριμένη απόσταση.

- Ψηφιακό πληκτρολόγιο με αυτόματο έλεγχο ορθογραφικών λαθών.
- Στο πρόγραμμα περιήγησης έχουν τοποθετηθεί οι λειτουργίες αντιγραφή-επικόλληση.
- Δυνατότητα προσθήκης εικόνας σε επαφή.
- Προσθήκη Animation κατά την εκκίνηση.

1.3.3 Android 1.6 Donut

Εικόνα 4: Εκδοση Donut

Η έκδοση Android 1.6 Donut παρουσιάστηκε στις 15 Σεπτεμβρίου 2009 ,η οποία βασίζεται στο Linux Kernel 2.6.29. Οι αλλαγές στο εσωτερικό του είχε ως αποτέλεσμα να υποτιμηθεί από τους καταναλωτές, παρόλο αυτά όμως έχει ταχύτερη απόκριση σε σχέση με την έκδοση “android 1.5 cupcake”.

Λειτουργίες Android 1.6 Donut :

- Υποστήριξη πολλαπλών αρχείων ταυτόχρονα.
- Βελτιωμένο Android Market.
- Βελτιωμένη κάμερα και γκαλερί.
- Μηχανή μετατροπής κειμένου σε ομιλία.
- Φωνητική αναζήτηση με ταχύτερη απόκριση.

1.3.4 Android 2.0 Éclair

Εικόνα 5: Εκδοση Eclair

Η έκδοση Android 2.0 Eclair παρουσιάστηκε στις 26 Οκτωβρίου 2009, η οποία βασίζεται στο Linux Kernel 2.6.29. Αν πρέπει να ξεχωρίσουμε κάποιο χαρακτηριστικό είναι η βελτίωση της κάμερας.

Λειτουργίες Android 2.0 Eclair :

- Λειτουργία σκηνών και εστίασης.
- Χρωματικά εφέ.
- Υποστήριξη προτύπου Bluetooth 2.1.
- Βελτιωμένη εφαρμογή Google Maps 3.1.2.
- Υποστήριξη ψηφιακού zoom από την κάμερα.
- Προσθήκη φλάς στη κάμερα.
- Αναζήτηση αποθηκευμένων μηνυμάτων και εικονομηνυμάτων.
- Βελτιωμένη MotionEvent κλάση ώστε να πραγματοποιούνται multi-touch γεγονότα.

1.3.5 Android 2.2 Froyo

Εικόνα 6: Εκδοση Froyo

Η έκδοση Android 2.2 Froyo παρουσιάστηκε στις 20 Μαΐου 2010, η οποία βασίζεται στο Linux Kernel 2.6.32 και πήρε το όνομα του από το frozen yogurt. Αν απομονώναμε ένα από τα κύρια χαρακτηριστικά του, αυτό θα ήταν η αναβάθμιση της ταχύτητας του Os.

Λειτουργίες Android 2.2 Froyo :

- Υποστήριξη Adobe Flash 10.1.
- Ενσωμάτωση του Chrome V8 Javascript στα browsers applications.
- Υποστήριξη Wi-fi hotspots και συμβατότητα USB.
- Μεγάλη κάρτα μνήμης και απόδοση.
- Εγκατάσταση εφαρμογών στην κάρτα μνήμης και η μεταφορά τους εκεί από τη μνήμη του τηλεφώνου.
- Ανανέωση του Android Market.

1.3.6 Android 2.3 Gingerbread

Εικόνα 7: Εκδοση Gingerbread

Η έκδοση Android 2.3 Gingerbread παρουσιάστηκε στις 6 Δεκεμβρίου 2010, η οποία βασίζεται στο Linux Kernel 2.6.35. Η έκδοση αυτή φτάνει να έχει τη μεγαλύτερη απήχηση αφού είναι πιο γρήγορη από τις προηγούμενες. Βέβαια δε σταμάτησαν εκεί η προγραμματιστές και τον Φεβρουάριο του 2011 επανεκδόθηκε σε Android 2.3.3 με πολλές αλλαγές.

Λειτουργίες Android 2.3 Gingerbread:

- Υποστήριξη οθόνης μεγάλου μεγέθους και ανάλυσης.
- Το user Interface έχει γίνει πιο απλό και γρήγορο.
- Υποστήριξη VoIP για τηλεφωνική ομιλία μέσω internet.
- Αναβάθμιση του πληκτρολογίου αφής.
- Δυνατότητα των λειτουργιών αντιγραφή-επικόλληση σε όλο το Os.

- Αναβάθμιση του ήχου αλλά και των διάφορων εφέ.
- Υποστήριξη του τύπου βίντεο WebM/VP8 και του κωδικοποιητή AAC.
- Προεγκαταστημένη υποστήριξη πολλαπλών καμερών.
- Download manager.
- Υποστήριξη περισσότερων αισθητήρων, όπως το βαρόμετρο και το γυροσκόπιο.

1.3.7 Android 3.0 Honeycomb

Εικόνα 8: Εκδοση Honeycomb

Η έκδοση Android 3.0 Honeycomb παρουσιάστηκε στις 9 Μαΐου 2011 βασισμένη στο Linux Kernel 2.6.36, και η ιδιαιτερότητά του ήταν ότι προοριζόταν αποκλειστικά για tablets. Αργότερα έχουμε την εξέλιξη της σε 3.1 και τέλος σε 3.2.

Λειτουργίες Android 3.0 Honeycomb :

- Ανανέωση γραφικών με αρκετά 3D στοιχεία.
- Αναβάθμιση πληκτρολογίου αφής.
- Εφαρμογή ανάγνωσης Google e-books.
- Υποστήριξη βιντεοκλήσεων μέσω της Google talk.
- Ανανεωμένη έκδοση της Google Maps.
- Υποστήριξη διπύρηνων και τετραπύρηνων επεξεργαστών.
- Στην έκδοση 3.1 έχουμε προσθήκη της επιλογής για τη μεταφορά αρχείου απευθείας από USB.
- Στην έκδοση 3.2 έχουμε την πρόσθεση μεταφοράς αρχείων από SD και τη δυνατότητα zoom.

1.3.8 Android 4.0 Ice Cream Sandwich

Εικόνα 9: Εκδοση Ice Cream Sandwich

Η έκδοση Android 4.0 Ice Cream Sandwich παρουσιάστηκε στις 19 Οκτωβρίου 2011, η οποία βασίζεται στο Linux Kernel 3.0.1. Η έκδοση αυτή έφερε τεράστιες αλλαγές στον σχεδιασμό του Android με το Holo UI.

Λειτουργίες Android 4.0 Ice Cream Sandwich :

- Ακόμη μεγαλύτερη ταχύτητα και απόδοση από τις προηγούμενες εκδόσεις.
- Αναβάθμιση του User Interface.
- Αντικατάσταση των φυσικών πληκτρολογίων αλλά και των αφής με εικονικά πλήκτρα.
- Αναβάθμιση του κλειδώματος με την προσθήκη αναγνώρισης προσώπου.
- Ο browser έχει την ικανότητα να ανοίγει μέχρι 16 καρτέλες ταυτόχρονα.
- Τερματισμός εφαρμογών που βρίσκονται στο backround.
- Υποστήριξη εγκατάστασης βίντεο σε 1080p.
- Δυνατότητα αποστολής δεδομένων εντός μιας συγκεκριμένης απόστασης με την εφαρμογή android beam μέσω του NFC.

1.3.9 Android 4.1 Jelly Bean

Εικόνα 10: Εκδοση Jelly Bean

Η έκδοση Android 4.1 Jelly Bean παρουσιάστηκε στις 27 Ιουνίου 2012, η οποία βασίζεται στο Linux Kernel 3.0.31. Η έκδοση αυτή είναι ιδιαίτερα σημαντική γιατί κατέχει το 50% των android συσκευών. Αυτό που μπορούμε να πούμε ότι ξεχώρισε είναι το Google Now, το οποίο μας παρέχει τα πάντα σύμφωνα με τα ενδιαφέροντα μας.

Λειτουργίες Android 4.1 Jelly Bean:

- Αναβάθμιση του συστήματος ειδοποίησης.
- Δυνατότητα σύνδεσης εξωτερικών ηχείων μέσω USB.
- Βελτίωση αισθητήρων.
- Αναβάθμιση λειτουργιών κάμερας.
- Βελτίωση των γραφικών μέσω του OpenGL ES 3.0.
- Υποστήριξη των tablets με μικρή οθόνη με τη χρήση βελτιωμένης έκδοσης για κινητά τηλέφωνα.

1.3.10 Android 4.4 KitKat

Εικόνα 11: Εκδοση KitKat

Η έκδοση Android 4.4 KitKat παρουσιάστηκε των Σεπτέμβρη 2013,μέσω της Google. Η Google κατάφερε να αναδείξει τη δύναμη του λειτουργικού σε φθηνές συσκευές και με αυτό δίνει τη δυνατότητα σε πολλούς καταναλωτές να αποκτήσουν συσκευές android με λιγότερα χρήματα. Βέβαια για να γίνει αυτό χρειάστηκαν να γίνουν αρκετές βελτιώσεις στο KitKat.

Λειτουργίες Android 4.4 KitKat :

- Δυνατότητα εκτύπωσης μέσω Wi-Fi.
- Υποστήριξη Bluetooth MAP.
- Βελτίωση στην ασφάλεια και επίλυση προβλημάτων προηγούμενων εκδόσεων.
- Βελτίωση σχεδιασμού με άσπρα αντί μπλέ στοιχεία.
- Απεγκατάσταση εφαρμογών που δεν υποστηρίζονται από το Android Market.
- Εγκατάσταση εικονικών κουμπιών στο μενού ανεξάρτητα από τα φυσικά.

1.3.11 Android 5.0 Lollipop

Εικόνα 12: Εκδοση Lollipop

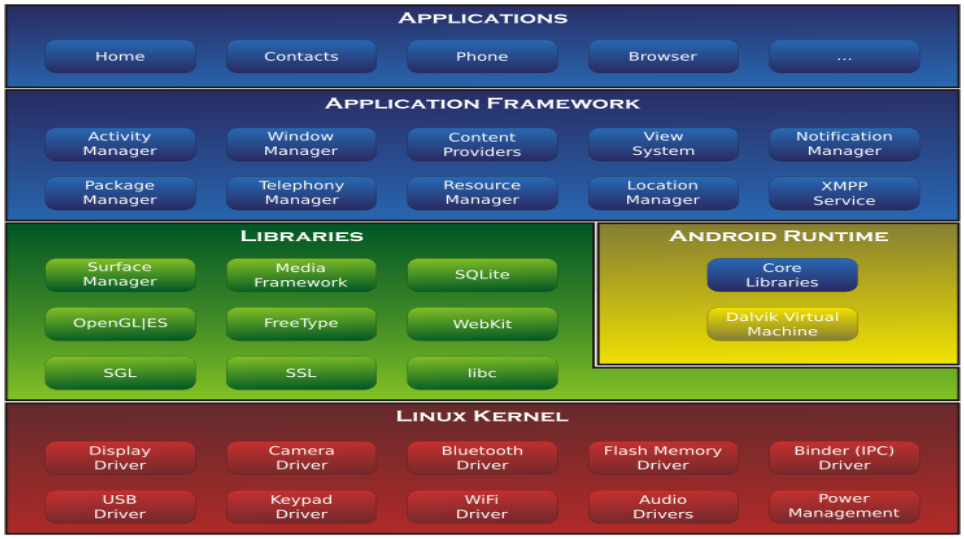
Η έκδοση Android 5.0 Lollipop παρουσιάστηκε τον Μάρτιο 2015.Όπως φαίνεται είναι η καλύτερη έκδοση του android, με το material desing της Google ,με την αναβάθμιση του Os αλλά και με τα πολλά εφέ.

Λειτουργίες Android 5.0 Lollipop:

- Λειτουργία επισκέπτη για άνετη κοινή χρήση της συσκευής.
- Εξοικονόμηση μπαταρίας.
- Smart Lock για αυξημένη ασφάλεια.
- Πλήρης υποστήριξη NFC.
- Εγκατάσταση εφαρμογών από το World Wide Web.
- Προηγμένες λειτουργίες φωτογραφίας

1.4 Αρχιτεκτονική

Η αρχιτεκτονική του λειτουργικού συστήματος Android αποτελείται από μια στοίβα πέντε επιπέδων, που σκοπό έχουν να παρέχουν στο χρήστη την δυνατότητα να αξιοποιήσει τους πόρους του συστήματος με τον καλύτερο δυνατό τρόπο.



Εικόνα 13: Αρχιτεκτονική Android

1.4.1 Πυρήνας Linux

Ο Linux Kernel είναι αυτός στον οποίο βασίζεται το λειτουργικό σύστημα Android και βρίσκεται στο τελευταίο επίπεδο. Αυτό είναι το επίπεδο αφαίρεσης που εξυπηρετεί το χρήστη στο να αναλάβει την διασύνδεση του hardware κάθε συσκευής με τις εφαρμογές. Είναι υπεύθυνο για την διαχείριση μνήμης, την ασφάλεια και την διαχείριση του συστήματος. Ακόμα σε αυτό το επίπεδο έχουμε τους οδηγούς (devices drivers) που μπορεί να υπάρχουν σε κάθε συσκευή όπως είναι το υποσύστημα ασύρματης σύνδεσης στο διαδίκτυο - WiFi, η οθόνη, ο ήχος, κ.α.

1.4.2 Επίπεδο βιβλιοθηκών

Οι βιβλιοθήκες του συστήματος είναι υλοποιημένες στη γλώσσα προγραμματισμού C++ και τρέχουν στον πυρήνα του Linux . Είναι προσβάσιμες στους προγραμματιστές μέσω του επιπέδου πλαισίου εφαρμογής. Οι βιβλιοθήκες έχουν όλο τον κώδικα του Android, για παράδειγμα η SqLite βιβλιοθήκη δίνει την δυνατότητα στην εφαρμογή να χρησιμοποιεί την αποθήκευση δεδομένων. Ουσιαστικά οι εφαρμογές εκμεταλλεύονται τις λειτουργίες που παρέχουν οι βιβλιοθήκες.

1.4.3 Επίπεδο εκτέλεσης

Στο ίδιο επίπεδο με τις βιβλιοθήκες συναντάμε το τμήμα Android Runtime το οποίο παρέχει ένα σύνολο βιβλιοθηκών οι οποίες επιτρέπουν στους προγραμματιστές να φτιάξουν εφαρμογές χρησιμοποιώντας γλώσσα προγραμματισμού Java και την εικονική μηχανή Dalvik η οποία πήρε το όνομα της από τον προγραμματιστή που την ανέπτυξε. Κάθε εφαρμογή του Android εκτελείται σε ένα δικό της Dalvik απομονώνοντας την από τις άλλες εφαρμογές, με αυτό τον τρόπο εξασφαλίζεται η ασφάλεια και η ευστάθεια του λειτουργικού συστήματος.

1.4.4 Επίπεδο πλαισίου εφαρμογών

Το Application Framework αφορά περισσότερο τους προγραμματιστές λογισμικού. Σε αυτό το επίπεδο οι προγραμματιστές μπορούν να κατασκευάσουν καινοτόμες εφαρμογές και να χρησιμοποιήσουν μια πληθώρα από API's για να ενσωματώσουν πρόσβαση στα αποθηκευτικά μέσα των συσκευών, διεπαφή χρήστη, σύνδεση στο διαδίκτυο κ.α. Οι βιβλιοθήκες είναι γραμμένες σε γλώσσα προγραμματισμού Java. Η αρχιτεκτονική αυτού του επιπέδου είναι διαμορφωμένη με τέτοιο τρόπο ώστε να δίνει στο χρήστη τη δυνατότητα να αλλάξει τα συστατικά της κάθε εφαρμογής. Η κλάση Activity Manager διαχειρίζεται τον κύκλο ζωής των εφαρμογών και ανήκει στο πακέτο android.app. Η κλάση Content Provider επιτρέπει τον διαμοιρασμό από μια εφαρμογή σε μια άλλη. Όμως η πιο σημαντική κλάση είναι η Views η οποία περιλαμβάνει τα ορατά στοιχεία μιας διεπαφής (Buttons, Textview κτλ.)

1.4.5 Επίπεδο εφαρμογών

Στο επίπεδο των εφαρμογών βρίσκονται όλες οι προ εγκατεστημένες εφαρμογές του Android όπως είναι το ημερολόγιο, οι χάρτες, τα SMS μηνύματα και η λίστα επαφών. Όλες οι εφαρμογές βρίσκονται σε αυτό το επίπεδο.

1.5 Το Android στην παγκόσμια αγορά

Το Android είναι σχεδόν ο απόλυτος κυρίαρχος της αγοράς έχοντας έναν πολύ σημαντικό αντίπαλο το λειτουργικό IOS της Apple αλλά και δύο ακόμη, το Windows phone και το Tizen της Samsung.
Σύμφωνα με την εταιρία IDC το 2014 το Android κατείχε το ποσοστό 81,5% παγκοσμίως καθώς θεωρήθηκε ο νικητής των "έξυπνων" κινητών συσκευών. Στη δεύτερη θέση δεν θα μπορούσε να είναι κανείς άλλος από το ios της Apple με ποσοστό 19,7%. Όπως φαίνεται από το ποσοστό της έρευνα τρίτος ανταγωνιστικός αντίπαλος για τις δύο αυτές εταιρίες δεν έχει υπάρξει ακόμα, αφού οι δυο τους συγκέντρωσαν το ποσοστό 96% παγκοσμίως. Επίσης η εταιρία IDC για να κάνει πιο κατανοητό το τι μπορεί να σημαίνει το ποσοστό 96% σε πωλήσεις συσκευών, παρουσίασε ότι το 2014 πουλήθηκαν σε καταναλωτές 1,3 δισεκατομμύρια κινητά τηλέφωνα από τα οποία μόνο τα 192 εκατομμύρια ήταν της Apple ενώ πάνω 1 δισεκατομμύριο ήταν της Google.

1.5.1 Διαμάχες

Οι συγκρούσεις της Google με μεγάλες εταιρίες δε φαίνεται να τελειώνουν ποτέ, αφού εδώ και χρόνια δέχεται συνεχώς μηνύσεις από τους αντιπάλους της. Το "παιδί" της, το λειτουργικό σύστημα Android την έχει κάνει τον απόλυτο κυρίαρχο της αγοράς και γι' αυτό το λόγο οι αντίπαλες εταιρίες την πολεμούν. Σύμφωνα με όσα έχουν αναφέρει στελέχη της Google ένας ισχυρός αντίπαλός της ήταν ο Steve Jobs ο ιδρυτής της apple, ο οποίος φαινόταν ότι μισούσε το android αφού το θεωρούσε κλεμμένο προϊόν. Ύστερα από πολλές δικαστικές διαμάχες και λίγο πριν το θάνατο του Jobs οι δύο εταιρίες κατέληξαν σε συμβιβασμό, με τα στελέχη της Google να υποστηρίζουν ότι όλο αυτό το μίσος για το android ήταν απλά μια "ασπίδα" για

να κεντρίσουν το ενδιαφέρον των καταναλωτών ώστε να αυξηθούν οι πωλήσεις της Apple.Οι διαμάχες της Google όμως δε σταματάνε εδώ , αφού εδώ και πέντε χρόνια βρίσκετε στις δικαστικές αίθουσες με την Oracle corp.Η υπόθεση αφορά τη γλώσσα προγραμματισμού Java της Oracle, την οποία η Google χρησιμοποιεί για τον σχεδιασμό του λειτουργικού συστήματος android. Η δίκη αυτή παραμένει ανοιχτή με την Oracle να ζητά 1 δισεκατομμύριο δολάρια αποζημίωση και την κατοχή των δικαιωμάτων για τη χρήση της γλώσσας Java από την Google, ενώ η Google το αρνείται και υποστηρίζει ότι η χρήση της Java είναι ελεύθερη.

1.5.2 Συσκευές Android vs IOS

Εικόνα 14: Android vs IOS

Οι λόγοι που οι συσκευές Android έχουν υπερισχύσει των συσκευών IOS και θεωρούνται οι καλύτερες αυτή τη στιγμή στην αγορά είναι οι εξής:

- Υπάρχει μεγάλη ποικιλία Android συσκευών με διαφορετικές δυνατότητες η κάθε μία, δίνοντας έτσι την ευκαιρία στον καταναλωτή να επιλέξει αυτή που τον ικανοποιεί.
- Οι συσκευές Android μπορούν να εξυπηρετήσουν όλη την αγορά, αφού και κάποιος που δε διαθέτει μεγάλο χρηματικό ποσό για την αγορά συσκευής μπορεί να αποκτήσει μια με μικρό ποσό.
- Το customization μιας συσκευής IOS δεν μπορεί να φτάσει το επίπεδο του Android, τόσο στο σύστημα αρχείων, όσο και στην προσθήκη widget ή την αφαίρεση εφαρμογών.
- Στις Android συσκευές έχουμε την ταυτόχρονη χρήση και λειτουργία πολλαπλών εφαρμογών σε διαφορετικά παράθυρα.
- Στα Android ο χρήστης έχει τη δυνατότητα να λαμβάνει τις πληροφορίες του στο taskbar.
- Οι συσκευές IOS φορτίζουν μόνο με το καλώδιο της Apple, ενώ οι Android με το micro-usb το οποίο μπορεί ο χρήστης να το βρει παντού.
- Οι IOS συσκευές ξεκλειδώνουν μόνο με το αποτύπωμα δακτύλου ή με κωδικό ,ενώ στις συσκευές android υπάρχουν περισσότερες επιλογές.
- Στις Android συσκευές μπορείς να κατεβάσεις μουσική απ όπου θέλεις, ενώ στο ios μόνο μέσω iTunes.

- Πολλές συσκευές Android έχουν πομπό IR και μπορούν να χρησιμοποιηθούν από πολλές ηλεκτρονικές συσκευές ως τηλεχειριστήριο.
- στα Android ο χρήστης μπορεί να μεγαλώσει το χώρο αποθήκευσης, αλλά και να αφαιρέσει επιτόπου τα δεδομένα του με χρήση microSD. Στο ios υπάρχει ο περιορισμός στην ενσωματωμένη μνήμη.

1.6 Δομή Εργασίας

Στο κεφάλαιο αυτό αναφερθήκαμε θεωρητικά στο λειτουργικό σύστημα Android και αποτελεί το πρώτο κεφάλαιο της εργασίας με τίτλο "Εισαγωγή". Στο δεύτερο κεφάλαιο "Εγκατάσταση εργαλείων ανάπτυξης Android" παρουσιάζουμε όλα τα εργαλεία και τα προγράμματα που χρησιμοποιήθηκαν για την κατασκευή της εφαρμογής. Στο τρίτο κεφάλαιο αναλύουμε τα κύρια σημεία του κώδικα, και στο τέταρτο κεφάλαιο κάνουμε παρουσίαση της εφαρμογής. Στο τελευταίο κεφάλαιο με τίτλο "Βιβλιογραφία" αναφέρουμε όλες τις πηγές που μας βοήθησαν για να υλοποιήσουμε την εφαρμογή.

2 Εγκατάσταση εργαλείων ανάπτυξης Android

Το λειτουργικό σύστημα Android είναι μια open source πλατφόρμα, δηλαδή μια πλατφόρμα ανοιχτού κώδικα η οποία δίνει τη δυνατότητα στους προγραμματιστές να υλοποιήσουν σε Java όποια εφαρμογή θέλουν εντελώς δωρεάν. Για την δημιουργία μιας Android εφαρμογής απαραίτητη είναι η εγκατάσταση κάποιων προγραμμάτων τα οποία διατίθενται δωρεάν στο διαδίκτυο.

2.1 Java & JDK

Η Google για το σχεδιασμό του Android χρησιμοποίησε τη γλώσσα προγραμματισμού Java.

Εικόνα 15: Java & Android

Η γλώσσα προγραμματισμού java είναι μια αντικειμενοστραφής γλώσσα η οποία παρουσιάστηκε το 1995 από την εταιρία Sun Microsystem. Στις αρχές του 1991 η εταιρία Sun ξεκίνησε την αναζήτηση της για μια νέα γλώσσα η οποία θα μπορούσε να αποτελέσει την πλατφόρμα ανάπτυξης λογισμικού για καταναλωτικά ηλεκτρονικά (μικρές φορητές συσκευές) . Η γλώσσα που υπήρχε τότε ήταν η σε όλους γνωστή C++, όμως δε μπορούσε να εφαρμοστεί και να καλύψει αυτές τις απαιτήσεις. Έπειτα από πολλά πειράματα ο James Gosling κατέληξε στην Oak, μια γλώσσα που είχε πολλά κοινά με την C++ αλλά είχε πιο έντονο αντικειμενοστραφή χαρακτήρα. Σύντομα οι υπεύθυνοι της εταιρίας ανακάλυψαν ότι αυτό το όνομα υπήρχε ήδη καταχωρημένο και αποφάσισαν να το μετονομάσουν σε Java , αφού μέχρι τότε αποτελούσε την αγαπημένη ποικιλία καφέ των δημιουργών της. Από την παρουσίαση της και μετά η Java έχει μια ανοδική πορεία και είναι πλέον από τις πιο δημοφιλής γλώσσες στον τομέα της πληροφορικής. Το 2006 γίνετε γλώσσα ανοιχτού κώδικα και το 2010 η εταιρία λογισμικού Oracle έπειτα από πολλές συζητήσεις έρχεται σε συμφωνία για την εξαγορά της Sun Microsystem.

Η Java χαρακτηρίζεται από τα εξής:

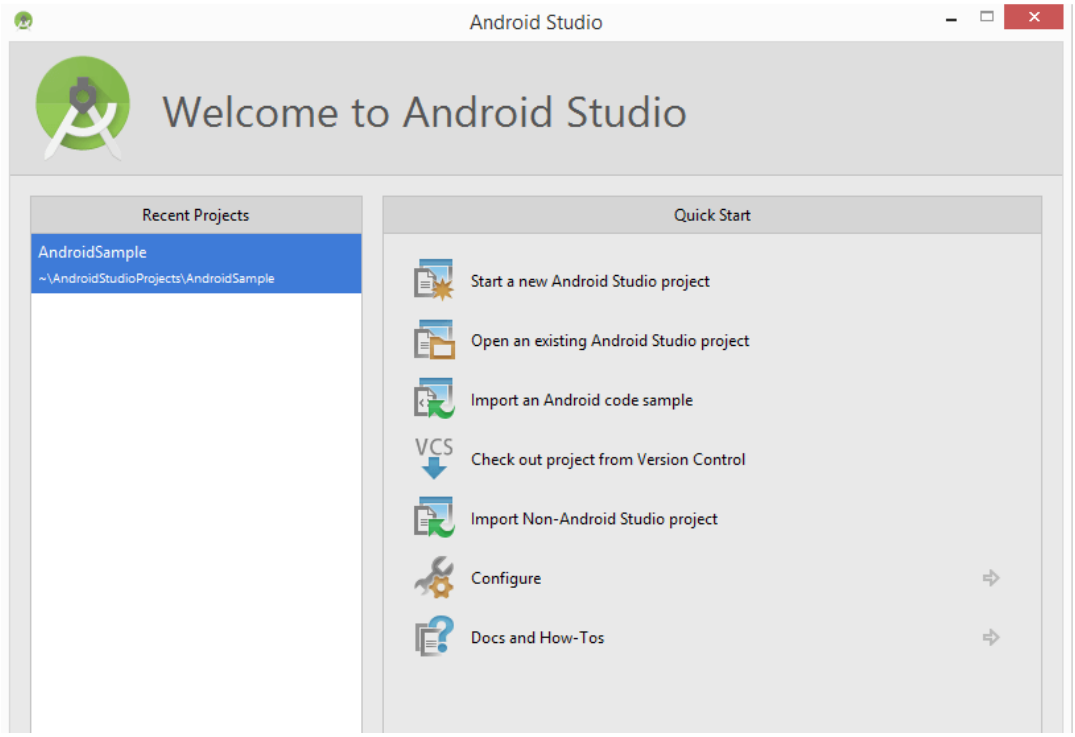
- Απλή: Ένας από τους στόχους των κατασκευαστών της ήταν να δημιουργήσουν μια γλώσσα εύκολη στη χρήση και να γίνεται γρήγορα κατανοητή από τους χρήστες. Μπορεί να βασίστηκε πάνω στη C++, παρόλο αυτά είναι πιο απλή ,αφού έχει εξαλειφθεί η χρήση pointer και η διαχείριση της μνήμης γίνετε από τη java.
- Αντικειμενοστραφής: Όταν μια γλώσσα είναι αντικειμενοστραφής εννοούμε ότι αποτελείτε από μια σειρά αντικειμένων τα οποία συμβάλλουν στο σχεδιασμό ενός προγράμματος. Ένα αντικείμενο είναι ο συνδυασμός δεδομένων και χαρακτηρίζετε σαν ένα "black box".Μέσα σε ένα πρόγραμμα υπάρχουν πολλά αντικείμενα τα οποία συνδέονται μεταξύ τους με βάση την έννοια της κληρονομικότητας. Οι αντικειμενοστραφής γλώσσες χρησιμοποιούνται για τη δημιουργία εφαρμογών.
- Πολυνηματική: Η java σε σχέση με τη C++ παρέχει έμφυτη τη δυνατότητα αντιμετώπισης πολλών καταστάσεων, δηλαδή μπορεί να κάνει πολλά και διαφορετικά πράγματα ταυτόχρονα.
- Ασφαλής:Είναι μια γλώσσα ανοιχτού κώδικα ,για το λόγο αυτό οι κατασκευαστές της και έδωσαν ιδιαίτερη προσοχή στην ασφάλεια. Η μέθοδος που χρησιμοποίησαν είναι η ασύμμετρη κρυπτογραφία, έτσι οι περισσότεροι ιοί καταπολεμούνται.
- Γλώσσα υψηλού επιπέδου: Η γλώσσα java είναι πιο απλή από τη C++ γιατί κάνει χρήση λέξεων που βρίσκονται πιο κοντά στη φυσική μας γλώσσα παρά στη γλώσσα μηχανής.
- Διαχείριση μνήμης: Η java κάνει μόνη της τη διαχείριση της μνήμης της αυτόματα ,μέσω του αποκομιστή απορριμμάτων.
- Δυναμική: Όταν ο προγραμματιστής επιλέξει την εκτέλεση του προγράμματος, τότε γίνετε η διασύνδεση των δεδομένων και των μεθόδων. Κλάσεις μπορούν να μεταφερθούν από το δίκτυο και να εκτελεστούν τοπικά χωρίς να είναι απαραίτητη η ενσωμάτωση τους στον κώδικα του προγράμματος.
- Ανεξάρτητη από το σύστημα: Είναι ένα από τα κυριότερα χαρακτηριστικά της java ,αφού ένας κώδικας γραμμένος σε java μπορεί να τρέξει το ίδιο σε windows,linux,unix χωρίς να χρειαστεί ξανά μεταγλώττιση ή να αλλάξει ο κώδικας λόγο αλλαγής λειτουργικού συστήματος.
- Ουδέτερη της υποκείμενης αρχιτεκτονικής: Η java έχει κατασκευαστεί έτσι ώστε να μπορεί να υποστηρίζει διαδικτυακές εφαρμογές. Λόγω του ότι κάθε υπολογιστής μπορεί να έχει διαφορετικό λειτουργικό σύστημα το πρόγραμμα θα πρέπει να περάσει από δύο διαδικασίες. Η πρώτη είναι η μεταγλώττιση και έπειτα η ερμηνεία. Το πρόγραμμα java περνάει μόνο μια φορά από τη διαδικασία μεταγλώττισης, ενώ από τη διαδικασία ερμηνείας περνάει κάθε φορά που ο χρήστης επιλέγει να τρέξει το πρόγραμμα.
- Υποστήριξη πολυμέσων: Πριν τη δημιουργία της java οι εφαρμογές πολυμέσων περιείχαν ήχο, εικόνα, κτλπ. Αλλά οι χρήστες δεν είχαν τη δυνατότητα εκτέλεσης τους. Μετά τη java αυτό γίνετε παρελθόν και όλοι οι χρήστες πλέον μπορούν και εκτελούν τα προγράμματα στο περιβάλλον τους.

Για την ανάπτυξη εφαρμογών στο λειτουργικό σύστημα Android απαραίτητο εργαλείο είναι το **JDK** (Java Development Kit). Διατίθεται δωρεάν από την Oracle από τον παρακάτω σύνδεσμο http://www.oracle.com/technetwork/articles/javase/index-jsp-138363.html η τρέχουσα έκδοση είναι η 8.

2.2 Android Studio

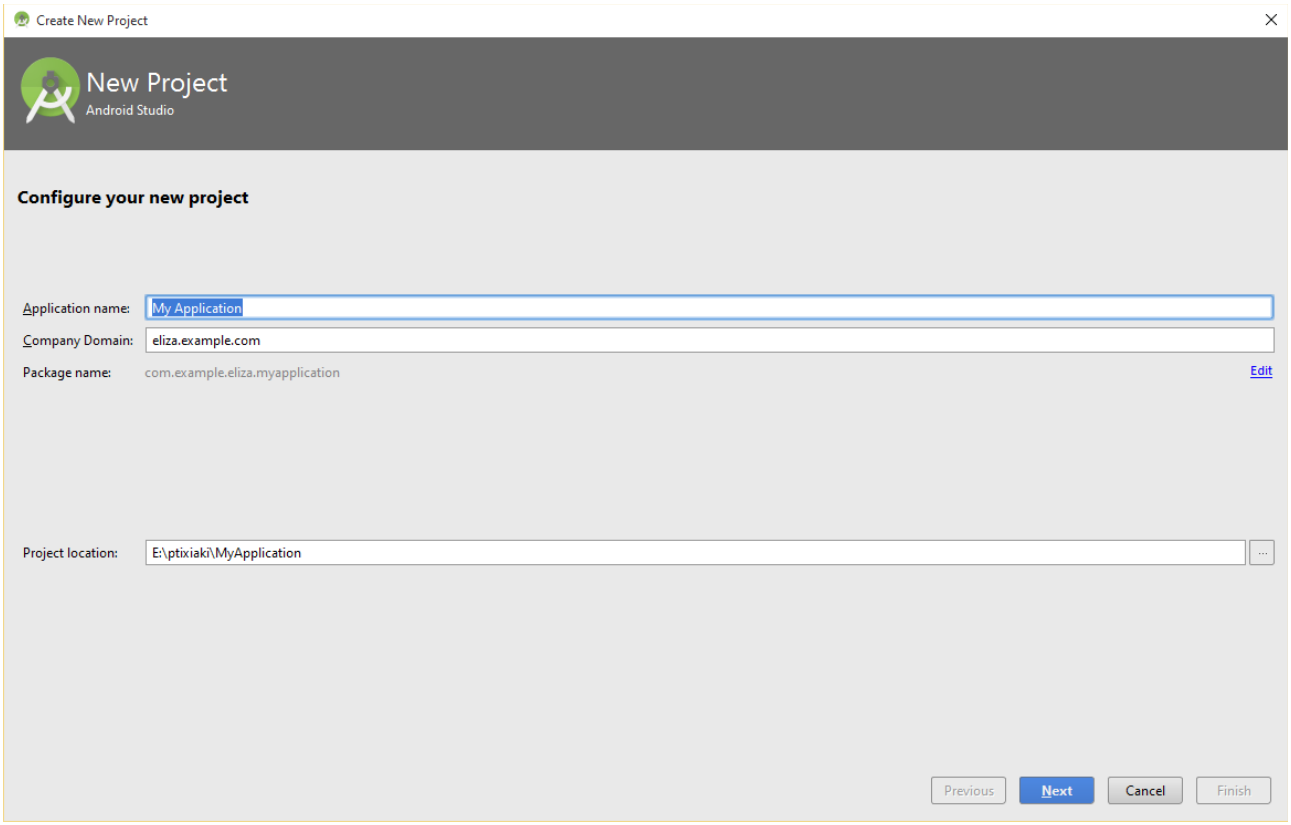
Το επίσημο Android IDE διατίθεται δωρεάν από το παρακάτω link: http://developer.android.com/sdk/index.html.

Εκτελώντας το πρόγραμμα για πρώτα φορά θα δούμε το παράθυρο που φαίνεται στην εικόνα και θα επιλέξουμε το “Start a new Android Studio project” για να φτιάξουμε ένα καινούργιο project.



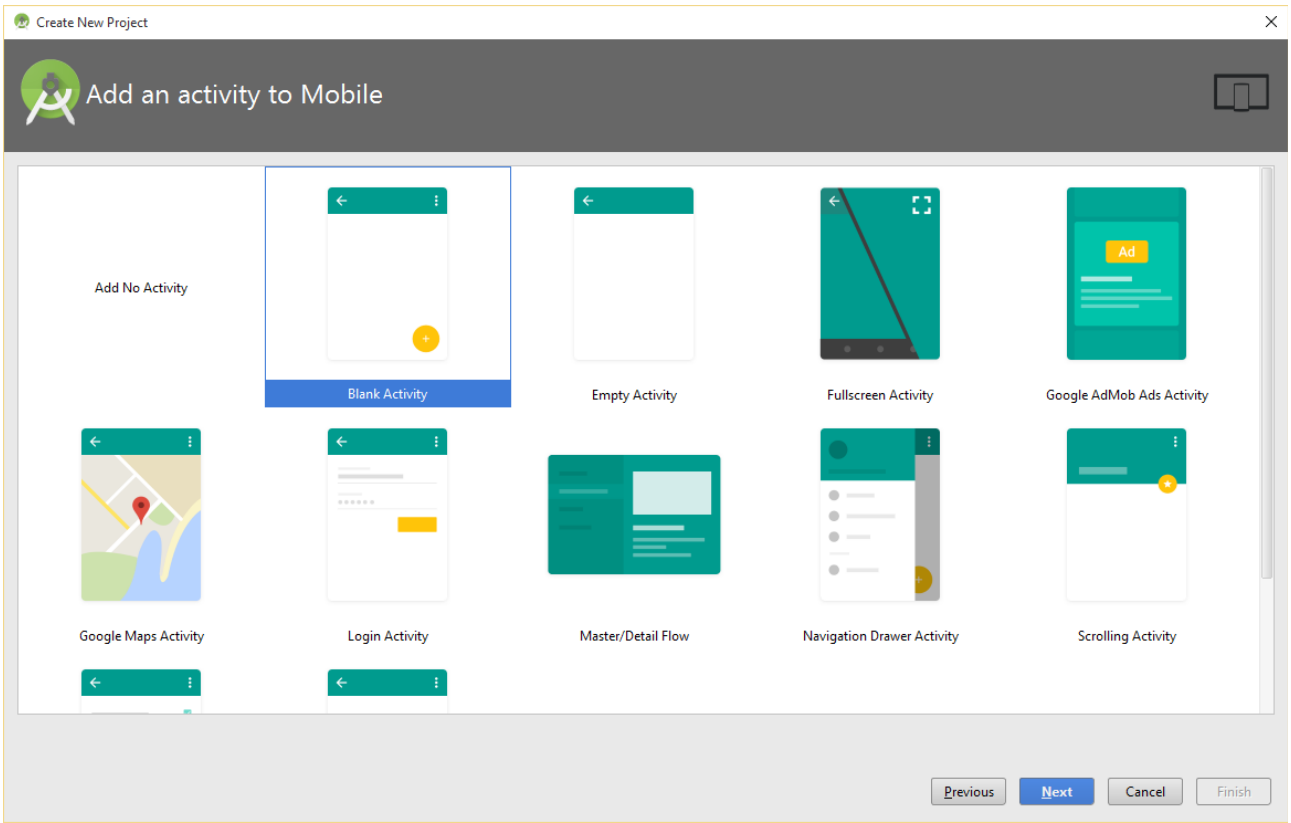
Εικόνα 16: Αρχική οθόνη Android Studio

Στη συνέχεια επιλέγουμε το όνομα που θα δώσουμε στο project και την τοποθεσία στο σκληρό δίσκο που θα αποθηκευτεί η εφαρμογή.



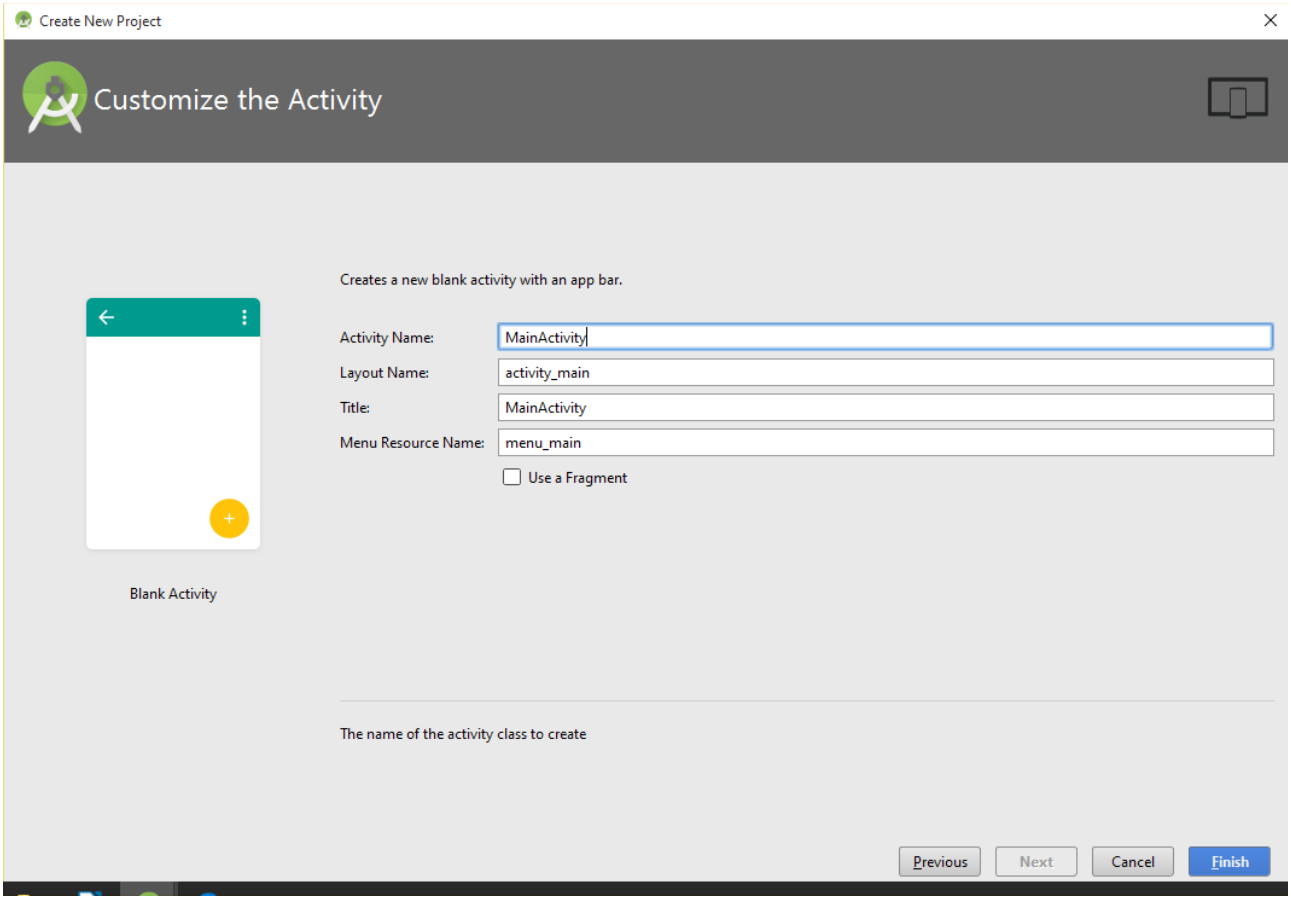
Εικόνα 17: Δήλωση ονόματος εφαρμογής

Στο επόμενο παράθυρο μας δίνετε η επιλογή να διαλέξουμε τον τύπο του πρωταρχικού activity.



Εικόνα 18: Επιλογή Activity

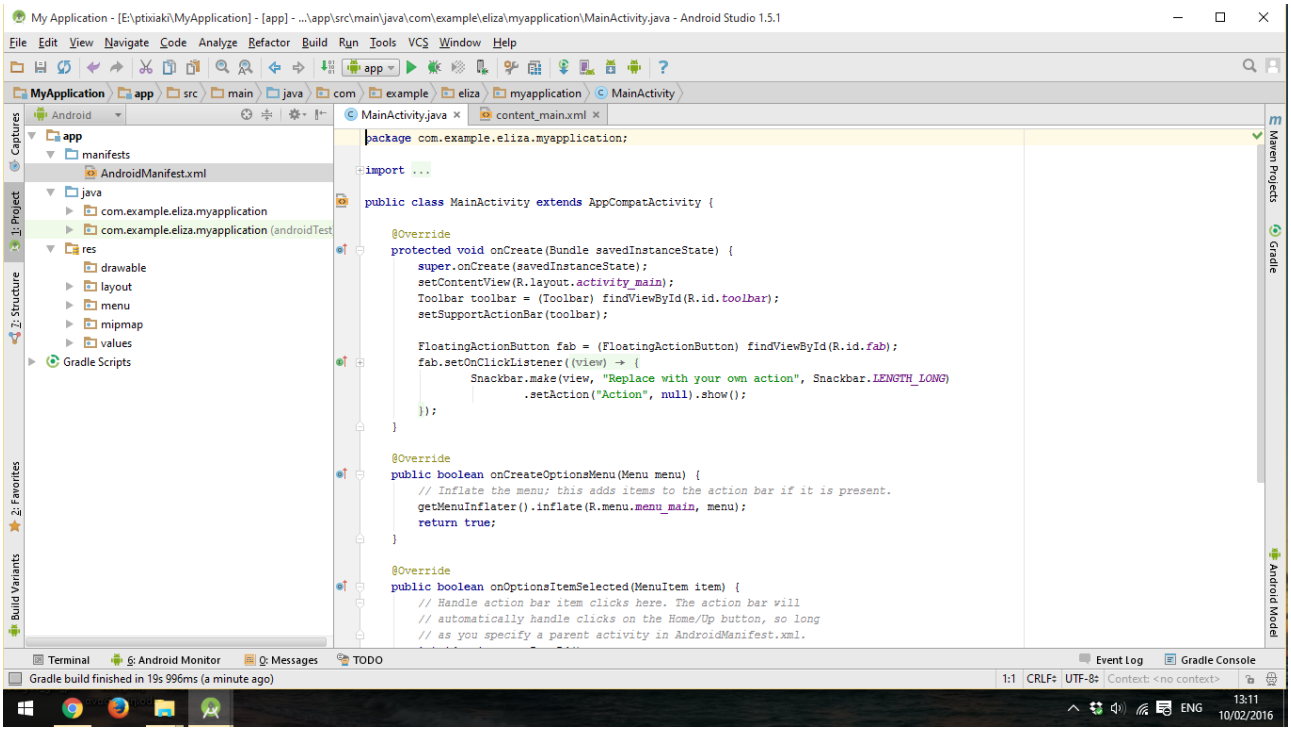
Τέλος δίνουμε στο activity που επιλέξαμε το όνομα που θέλουμε και είναι αυτό που θα εμφανίζεται στο Action Bar της εφαρμογής.



Εικόνα 19: Δήλωση ονόματος Activity

2.3 Περιβάλλον εργασίας Android Studio

Αντικρίζοντας το περιβάλλον εργασίας του Android Studio βλέπουμε τους φακέλους του project ιεραρχικά δομημένους και στα δεξιά εμφανίζεται το αρχείο MainActivity.java του οποίο το όνομα το επιλέξαμε εμείς όταν φτιάξαμε το project, όλα τα activity που θα δημιουργήσουμε θα αποθηκευτούν σε αυτό το φάκελο.



Εικόνα 20: MainActivity

2.4 Manifest.xml

Σε κάθε project στο Android Studio περιλαμβάνεται το αρχείο AndroidManifest.xml που αποθηκεύεται πάντα στην κορυφή της ιεραρχίας των αρχείων. Ορίζει τη δομή, τα συστατικά και τις απαιτήσεις της εφαρμογής. Πρόκειται για ένα αρχείο xml που περιλαμβάνει τις σημαντικότερες πληροφορίες της εφαρμογής.

Κάποιες από τις πληροφορίες είναι:

- Ο τίτλος της εφαρμογής.
- Το εικονίδιο που βλέπουν οι χρήστες.
- Ο αριθμός έκδοσης εφαρμογής.
- Πληροφορίες του SDK.
- Τα δικαιώματα της εφαρμογής.

Παρουσιάζουμε ένα μέρος του κώδικα από το αρχείο της εφαρμογής μας

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.traveler.travelerassistant" >

     <!--permissions-internet access-->
    <uses-permission android:name=
    "android.permission.INTERNET" />
    <uses-permission android:name=
    "android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name=
    "android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name=
    "com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <!--
 The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but are recommended.
    -->
    <uses-permission android:name=
    "android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name=
    "android.permission.ACCESS_FINE_LOCATION" />
```

```
      <!--application Icon-->
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version"/>
        <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="@string/google_maps_key" />
```

```
         <!--Statement of Activity -->
        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps"
            android:theme="@style/splashScreenTheme" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name=
                "android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

2.5 Φάκελοι res και src

Ο κατάλογος res περιέχει όλους τους πόρους που δομούν την εφαρμογή. Οι πόροι είναι τα δεδομένα της εφαρμογής, υπάρχουν πολλοί και διαφορετικοί τύποι, όπως για παράδειγμα τα αρχεία κειμένου, εικόνας, xlm layout τα οποία αποθηκεύονται ανάλογα με το είδος τους σε διαφορετικούς υποφακέλους.

Ο φάκελος res έχει τους ακόλουθους φακέλους χωρισμένους ανάλογα με το περιεχόμενο τους :

- Το λειτουργικό σύστημα Android υποστηρίζει αρχεία εικόνων τύπου png, gif, jpg. Ο φάκελος **drawable** περιέχει τα αρχεία εικόνας που χρησιμοποιεί η εφαρμογή και XML αρχεία τα οποία ορίζουν διαγράμματα, σχήματα και χρώματα που αλλάζουν μέγεθος ανάλογα με τις διαστάσεις της οθόνης.
- Ο φάκελος **layout** περιέχει όλα τα αρχεία xml που ορίζουν την εφαρμογή. Με την χρήση αρχείων XML γίνεται πιο εύκολη η αλλαγή εμφάνισης της εφαρμογής.
- Στο φάκελο **menu** βρίσκονται αρχεία xml που είναι υπεύθυνα για τη δημιουργία menu σε συγκεκριμένα σημεία της εφαρμογής.
- Στο φάκελο **values** περιέχονται όλοι οι πόροι κειμένου που χρησιμοποιούνται στην εφαρμογή .

Στο φάκελο src περιέχονται τα αρχεία που συνθέτουν την εφαρμογή μας π.χ java αρχεία, Service Content Providers, Activities κ.ά. Πρέπει να σημειωθεί ότι τα αρχεία του κώδικα της εφαρμογής αποθηκεύονται μόνο στον φάκελο src.

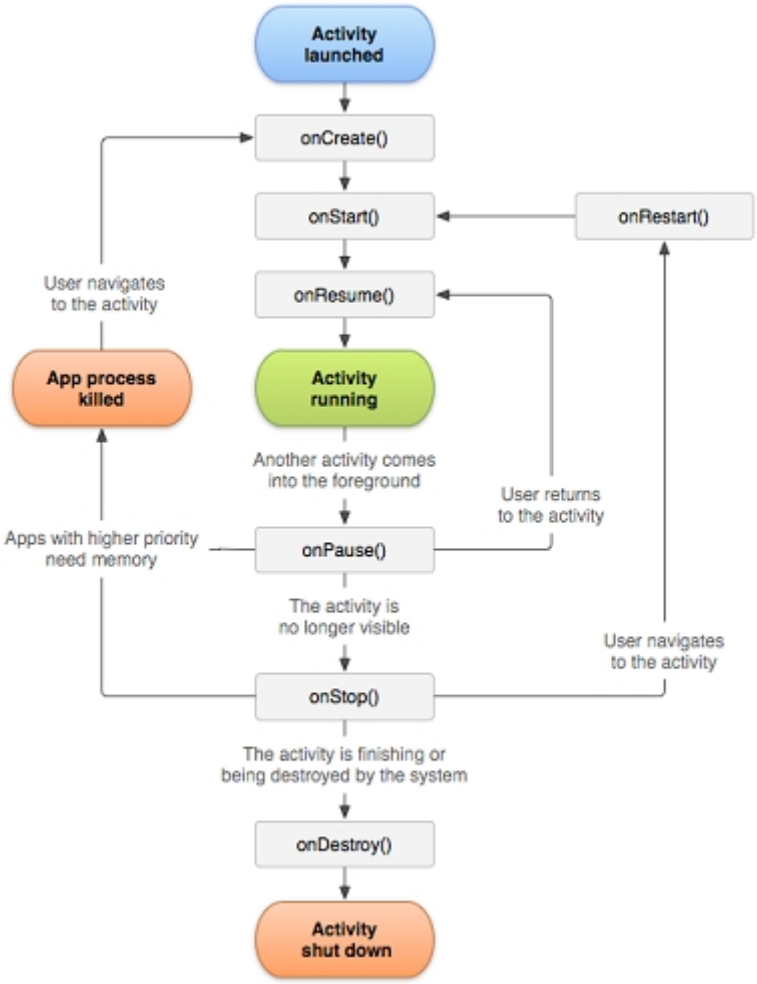
2.6 Ασφάλεια στο Android

Όταν μια εφαρμογή εγκατασταθεί σε μια συσκευή λειτουργεί στη δική της εικονική μηχανή. Ο μηχανισμός ασφαλείας που παρέχεται από το Android είναι τα δικαιώματα που θέτουν περιορισμούς στις ενέργειες που κάνουν οι διεργασίες. Η εφαρμογή έχει πρόσβαση μόνο στους πόρους του συστήματος που χρειάζεται μέσω του αρχείου AndroidManifest.xml. Τα δικαιώματα και οι πόροι που απαιτεί η εφαρμογή εμφανίζονται κατά την εγκατάσταση και ο χρήστης έχει τη δυνατότητα να αποφασίσει αν συμφωνεί η διαφωνεί με την πολιτική των δικαιωμάτων. Σημαντικός παράγοντας είναι οι αξιολογήσεις και οι κριτικές άλλων χρηστών. Το λειτουργικό σύστημα Android είναι σύστημα πολλών χρηστών. Η κάθε εφαρμογή αντιμετωπίζεται σαν διαφορετικός χρήστης όπου το σύστημα της δίνει έναν μοναδικό αριθμο ID ο οποίος όμως είναι άγνωστος από την εφαρμογή. Το σύστημα δίνει άδειες χρήσης στα αρχεία της εφαρμογής και μόνο η εφαρμογή με το σωστό ID μπορεί να έχει πρόσβαση.

3 Προγραμματισμός Android

Στο παρόν κεφάλαιο θα εξηγήσουμε τον τρόπο με τον οποίο δημιουργήσαμε την εφαρμογή. Ένα Activity αποτελείται από αρχεία xml και κλάσεις Java. Πριν όμως αναλύσουμε τα παραπάνω θα πρέπει να κατανοήσουμε τη λειτουργία και τη δομή του Activity.

3.1 Κύκλος ζωής του Activity



Εικόνα 21: Κύκλος ζωής ενός Activity

Κάθε εφαρμογή Android είναι ένα σύνολο εργασιών. Activity ονομάζεται η κάθε εργασία. Το λειτουργικό σύστημα Android μπορεί να εκτελέσει πολλές εφαρμογές ταυτόχρονα αρκεί να έχει την απαιτούμενη ισχύ και μνήμη, όμως κάθε χρονική στιγμή εκτελεί μόνο μια εργασία (Activity). Γι αυτό το λόγο το Android τοποθετεί τα Activity σε μια στοίβα. Όταν ξεκινήσει μια εργασία τότε τοποθετείται στην κορυφή της στοίβας και η προηγούμενη παραμένει από κάτω μέχρι να τελειώσει η τρέχουσα και να έρθει αυτή στο προσκήνιο.

Στην παραπάνω εικόνα απεικονίζεται ο κύκλος ζωής του Activity. Οι παραλληλόγραμμες εικόνες είναι οι μέθοδοι callback που εφαρμόζονται σε διάφορες διεργασίες όταν μια Activity κινείται μεταξύ διαφόρων καταστάσεων. Τα έγχρωμα παραλληλόγραμμα δείχνουν τις καταστάσεις που μπορεί να βρεθεί μια Activity.

Μέθοδοι που καλούνται κατά τη διάρκεια ζωής ενός Activity :

- Η **onCreate()** είναι η πιο σημαντική μέθοδος και καλείται όταν η δραστηριότητα δημιουργείται για πρώτη φορά. Στο σημείο αυτό θα πρέπει να ορίσουμε τον πόρο που περιγράφει την διεπαφή χρήση ώστε να μπορέσουμε να εμφανίσουμε στην οθόνη το επιθυμητό αποτέλεσμα.

- Η μέθοδος o**nStart()** καλείται ακριβώς πριν εμφανιστεί η διεπαφή στον χρήση. Την κλήση αυτή ακολουθεί η onResume() και η κλήση onStop().

- Η μέθοδος **onRestart()** εκτελείται όταν ένα Activity μεταβαίνει από την κλήση onStop() στην κατάσταση Running. Καλείται πριν από την onStart().

- **OnResume()** καλείται πριν γίνει διαθέσιμη η διεπαφή στον χρήση. Η δραστηριότητα είναι στην κορυφή της στοίβας και είναι το κατάλληλο σημείο έναρξης βίντεο και ήχου.

- Με την μέθοδο **onPause()** ενημερώνεται το τρέχον Activity ότι ένα άλλο Activity θα βρεθεί στην κορυφή της στοίβας. Στο σημείο αυτό πρέπει να αποθηκευτούν τα δεδομένα σε μόνιμα μέσα αποθήκευσης και να σταματήσουν οι διεργασίες που καταναλώνουν χρόνο στον επεξεργαστή.

- Η **onStop()** καλείται όταν δεν είναι πλέον ορατή στο χρήστη. Αυτό συμβαίνει γιατί το λειτουργικό σύστημα τερματίζει το Activity η γιατί μπορεί να το καλύπτει κάποιο άλλο.

- Η μέθοδος **onDestroy()** καλείται όταν ένα Activity καταστρέφεται. Η Activity έχει ολοκληρώσει τον κύκλο ζωής και τερματίζει ή το λειτουργικό σύστημα την καταστρέφει για λόγους οικονομίας μνήμης.

3.2 Ανάλυση κώδικα

3.2.1 airport.java

Το αρχείο java που θα αναλύσουμε είναι το Airport.java, σε αυτό το αρχείο δημιουργούμε την κλάση airport η οποία κάνει extends στην Activity. Αυτό σημαίνει ότι η airport μπορεί να κληρονομήσει τις μεθόδους και τις ιδιότητες της Activity. Αρχικά πάνω από τη δημιουργία της κλάσης δηλώνουμε το πακέτο μας και έπειτα κάνουμε import τις παρακάτω βιβλιοθήκες για να δημιουργήσουμε την κλάση.

```
package com.example.traveler.travelerassistant;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.Spinner;
import android.widget.TextView;

import java.util.ArrayList;
```

Δημιουργήσαμε μια @Override μέθοδο την onCreate() η οποία καλείτε όταν αρχίζει η δραστηριότητα, ο setContentView αναλαμβάνει να συμπεριλάβει το xml αρχείο activity_airport.

```
public class airport extends Activity {

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

Με τον παρακάτω κώδικα παίρνουμε την επιλογή αεροδρομίου από τον χάρτη.

```
    final int airind=getIntent().getExtras().getInt("dbindex");
```

Στη συνέχεια χρησιμοποιούμαι τη μέθοδο findViewById με παράμετρο το id, που στην περίπτωσή μας είναι το airportcity για να μπορέσουμε να κάνουμε την αντιστοιχία του TextView με αυτό του xml αρχείου. Αφού καλέσουμε με αυτόν τον τρόπο το αρχείο xml που θέλουμε, για να το επιστρέψουμε χρησιμοποιούμε την μέθοδο getStringArray η οποία επιστρέφει έναν πίνακα χαρακτήρων. Αυτός ο πίνακας χαρακτήρων στην σελίδα μας εμφανίζετε ως το όνομα της πόλης που επιλέξαμε.

```
    final TextView aircity=
    (TextView)findViewById(R.id.airportcity);
    final String[] airportc =
    getResources().getStringArray(R.array.airportcity);
    String city = airportc[airind];
    aircity.setText(city);
```

Από το αρχείο activity_airport.xml ο κώδικας του <TextView../>. Με το TextView δηλώνουμε ότι η περιγραφή που θα ακολουθήσει θα αφορά ένα πεδίο κειμένου. Αρχικά βάζουμε το tag “<” και μετά το είδος TextView, στη συνέχεια αρχίζουμε να προσθέτουμε διάφορα χαρακτηριστικά όπως το μέγεθος,το id το οποίο είναι μοναδικό για κάθε είδος,την τοποθέτηση,το χρώμα,τη γραμματοσειρά κ.α που θα αποτελέσουν την περιγραφή του είδους TextView.

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/airportcity"
    android:gravity="center|center_horizontal"
    android:inputType="textMultiLine"
    android:textAlignment="center"
    android:textColor="#0827BF"
    android:textSize="26sp"
    android:textStyle="bold|italic"
    android:ellipsize="none"
    android:singleLine="false"
    android:visibility="visible"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:linksClickable="false"
    android:layout_gravity="center_horizontal"
    android:background="@drawable/rounded_corner" />
```

Για να εμφανίσουμε στην σελίδα το όνομα του αεροδρομίου που επιλέξαμε κάνουμε χρήση πάλι τις ίδιες μεθόδους,τη μέθοδο findViewById με παράμετρο το id, που στην περίπτωσή αυτή είναι το airportname και τη μέθοδο getStringArray η οποία επιστρέφει έναν πίνακα χαρακτήρων δηλαδή το όνομα του αεροδρομίου.

```
TextView airname=
(TextView) findViewById(R.id.airportname);
final String[] airportn =
getResources().getStringArray(R.array.airportname);
String name = airportn[airind];
```

Από το αρχείο activity_airport.xml ο κώδικας του <TextView../>.

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/airportname"
    android:gravity="center|center_horizontal"
    android:inputType="textMultiLine"
    android:textAlignment="center"
    android:textColor="#080521"
    android:textSize="22sp"
    android:textStyle="bold|italic"
    android:ellipsize="none"
    android:singleLine="false"
    android:visibility="visible"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:linksClickable="false"
    android:layout_gravity="center_horizontal"
    android:background="@drawable/rounded corner" />
```

Το όνομα που μας επέστρεψε η μέθοδος getStringArray το καταχωρήσαμε στη μεταβλητή name η οποία είναι τύπου χαρακτήρα. Στη συνέχεια κάνοντας χρήση της δομής επανάληψης if ελέγχουμε αν μας έχει επιστραφεί κάποιο όνομα. Αν η συνθήκη μας είναι αληθής τότε το όνομα εμφανίζετε, αν είναι ψευδής τότε το όνομα χάνετε.

```
if(name!=null && name.length()> 0)
{airname.setVisibility(View.VISIBLE);    }
else { airname.setVisibility(View.GONE);}
airname.setText(name);
```

Με τον παρακάτω κώδικα καλούμε τη μέθοδο findViewById με παράμετρο το id, το οποίο είναι το citysp για να μπορέσουμε να κάνουμε την αντιστοιχία του Spinner με αυτό του xml αρχείου.

```
final Spinner citysp = (Spinner) findViewById(R.id.citysp);
```

Ακολουθεί το αρχείο activity_airport.xml με τον κωδικό <Spinner../>.Χρησιμοποιούμε το Spinner γιατί θέλουμε να κατασκευάσουμε ένα παράθυρο επιλογών. Ο Spinner παρέχει ένα γρήγορο τρόπο επιλογής τιμής από ένα σύνολο. Επιλέγοντας το κουμπί αυξομείωσης εμφανίζεται το μενού με όλες τις τιμές ,από τις οποίες μπορούμε να επιλέξουμε. Αρχικά βάζουμε το tag “<” και μετά το είδος Spinner ,στη συνέχεια αρχίζουμε να προσθέτουμε διάφορα χαρακτηριστικά όπως το μέγεθος,το id,τη λειτουργία προβολής για τις επιλογές,το χρώμα,το μέγεθος γραμμάτων κ.ά που θα αποτελέσουν την περιγραφή του είδους Spinner.

```
<Spinner
    android:layout_width="wrap_content"
    android:layout_height="30sp"
    android:id="@+id/citysp"
    android:layout_gravity="center"
    android:textAlignment="gravity"
    android:textColor="#302A54"
    android:textSize="18sp"
    android:gravity=
    "center|center_vertical|center_horizontal|fill_vertical"
    android:visibility="visible"
    android:dropDownWidth="wrap_content"
    android:spinnerMode="dialog"
    android:transitionName="Επιλέξτε προορισμό"
    android:layoutMode="opticalBounds"
    android:background="#7E71BD"
    android:dropDownSelector="@drawable/arrow" />
```

Για να μπορέσουμε να εκμεταλλευτούμε τον παραπάνω Spinner στον κώδικά μας, πρέπει να δημιουργήσουμε έναν ArrayAdapter. Πρώτα κάνουμε τη δήλωση του τύπου του αντικειμένου που πρόκειται να μετατραπεί και στη συνέχεια προσθέτουμε τις παραμέτρους. Οι παράμετροι που θα χρησιμοποιήσουμε είναι το στοιχείο xml simple_spinner_item και η ArrayList. Με την παράμετρο ArrayList πετυχαίνουμε τη δημιουργία μιας λίστας δεδομένων.

Η καρτέλα στην οποία θα εμφανίζεται η λίστα με τις πόλεις προορισμού θέλουμε να περιέχει όλες τις πόλεις εκτός από την πόλη αναχώρησης ,αυτό το πετύχαμε κάνοντας τους παρακάτω ελέγχους.

```
ArrayAdapter<String> adapter=
new ArrayAdapter<String>
(this,android.R.layout.simple_spinner_item,
new ArrayList<String>());
String[] citiest=getResources().getStringArray(R.array.city);
int maxc=citiest.length;
final String[] airc=new String[maxc-1];
final String[] airn=new String[maxc-1];
int cind=0;
for (int i=0;i<maxc;i++){
    if (i!=airind+1){
        adapter.add(citiest[i]);
    }
}
for (int i=0;i<maxc-1;i++){
    if (i!=airind){
        airc[cind]=airportc[i];
        airn[cind]=airportn[i];
        cind++;
    }
}
```

Με τον παρακάτω κώδικα καθορίζουμε το layout που θα εμφανιστεί η λίστα.

```
    adapter.setDropDownViewResource
    android.R.layout.simple_spinner_dropdown_item);
```

Στη συνέχεια αφού κάνουμε την εφαρμογή του Adapter στον Spinner,με τη μέθοδο FindViewById γίνετε η αντιστοιχία του component με αυτό του xml αρχείου. Το Spinner που δημιουργήσαμε αναφέρεται στην πόλη προορισμού. Όταν ο χρήστης επιλέξει την πόλη προορισμού τότε ενεργοποιείτε η μέθοδος setOnItemSelectedListener. Για να πάρουμε αυτό το στοιχείο από τον Spinner χρησιμοποιήσαμε τη μέθοδο OnItemSelect την οποία ορίσαμε ως Override.

```
citysp.setAdapter(adapter);
final TextView airportto=(TextView) findViewById(R.id.cityto);
final int[] cityto = new int[1];
citysp.setOnItemSelectedListener
(new AdapterView.OnItemSelectedListener() {

    @Override
    public void onItemSelected(AdapterView<?> arg0,
    View arg1,int arg2, long arg3) {
        if (arg2 == 0) {
            airportto.setText("Αεροδρόμιο προορισμού");
        } else {
            String aircityto =
            airc[arg2-1] + "\n" + airn[arg2-1];
            airportto.setVisibility(View.VISIBLE);
            airportto.setText(aircityto);
            cityto[0] =arg2;
        }
    }
```

Στον παρακάτω κώδικα δημιουργήσαμε το κουμπί <Εύρεση πτήσης> και του δηλώσαμε έναν Listener. Όταν ο χρήστης πατήσει το κουμπί τότε γίνετε μετάβαση στο chooseflight.java μέσω της Intent ,όπου στέλνονται και τα id με τη μέθοδο putExtra.

```
final Button saveBtn = (Button) findViewById(R.id.findflight);
saveBtn .setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        String aircityto =
        airc[cityto[0]-1] + "\n" + airn[cityto[0]-1];;
        Intent intc =
        new Intent(getApplicationContext(), chooseflight.class);
        intc.putExtra("cityfrom",airind);
        intc.putExtra("cityto",cityto[0]);
        intc.putExtra("airportto",aircityto);
        startActivity(intc);
    }
});
```

Στο αρχείο chooseflight.java οι ενέργειες που κάνουμε είναι η δημιουργία της καρτέλας εμφάνισης πτήσης καθώς επίσης και η ενημέρωση στον χρήστη όταν δεν υπάρχει πτήση. Αρχικά δημιουργούμε δύο πίνακες οι οποίοι παίρνουν τα στοιχεία τους από το αρχείο DBAdapter.java.

```
    String[] flightdetails =
    new String[]{DBAdapter.airflights[0],
    DBAdapter.airflights[1], DBAdapter.airflights[2]};
```

Στη συνέχεια κάνουμε εφαρμογή των στοιχείων αυτών στην καρτέλα μας αφού πρώτα έχουμε δημιουργήσει έναν SimpleCursorAdapter. Για να γίνει η συμπλήρωση της καρτέλας με τα στοιχεία της πτήσης,αρχικά παίρνουμε τις πληροφορίες από τη βάση και στη συνέχεια τις εμφανίζουμε. Στον παρακάτω κώδικα δηλώνουμε σαν global μεταβλητές τα στοιχεία που παίρνουμε από το αρχείο DBAdapter, το οποίο είναι ο συνδετικός κρίκος με την βάση δεδομένων. Η σύνδεση αυτή επιτεύχθηκε με την κατασκευή των ανάλογων μεθόδων,μια τέτοια μέθοδος είναι η getSingleFlightRow. Αυτό που πετυχαίνουμε καλώντας την είναι να δούμε αν υπάρχει η πτήση που επιλέξαμε. Αν υπάρχει,τότε εμφανίζετε ένα μήνυμα ειδοποίησης toast το οποίο ενημερώνει σχετικά με την πτήση,όμως το μήνυμα αυτό δε γίνετε αντιληπτό από τον χρήστη. Έπειτα μέσω της Intent μεταβήκαμε από την υπάρχουσα δραστηριότητα στην flight.java, ενώ παράλληλα στείλαμε και τα id με την μέθοδο putExtra.

```
private void flightchosen(final int indto) {
    final int indfrom =
    getIntent().getExtras().getInt("cityfrom");
    ListView flightslist =
    (ListView) findViewById(R.id.lVflights);
    flightslist.setOnItemClickListener
    (new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent,
        View viewClicked, int position, long rowid) {
            Cursor flight = Tadb.getSingleFlightRow
            (Tadb.dbTablesf[indfrom], rowid);
            int from = indfrom;
            if (flight.moveToFirst()) {
                long rid =
                flight.getLong(DBAdapter.COL_ROWID);
                String city =
                flight.getString(DBAdapter.COL_ARRIVAL);
                String time=
                flight.getString(DBAdapter.COL_TIME);
                String airline=
                flight.getString(DBAdapter.COL_AIRLINE);
                String flightn=
                flight.getString(DBAdapter.COL_FLIGHT);
                String message =
                "id= " + rid + " and city is " + city;
```

```
                Toast.makeText(chooseflight.this,
                message, Toast.LENGTH_LONG).show();
                String airportto=
                getIntent().getExtras().getString("airportto");
                Intent intc =
                new Intent(getApplicationContext(), flight.class);
                intc.putExtra("cityfrom", from);
                intc.putExtra("cityto", city);
                intc.putExtra("rowid", rid);
                intc.putExtra("time",time);
                intc.putExtra("airline",airline);
                intc.putExtra("flight",flightn);
                intc.putExtra("airportto",airportto);
                intc.putExtra("indto",indto);
                startActivity(intc);}
            flight.close();
        }
    });
}
```

Στην activity flight.java δημιουργήσαμε μια @Override μέθοδο την onCreate() η οποία καλείτε όταν αρχίζει η δραστηριότητα, ο setContentView αναλαμβάνει να συμπεριλάβει το xml αρχείο activity_flight. Στη συνέχεια κάνοντας χρήση της μεθόδου findViewById πετυχαίνουμε την αντιστοιχία των TextView με αυτά του xml αρχείου καθώς επίσης και την αντιστοιχία των ImageView.Τέλος ανάλογα με τον προορισμό που επιλέγει ο χρήστης εμφανίζονται στην καρτέλα τα στοιχεία της πτήσης.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_flight);
    TextView airfrom=
    (TextView) findViewById(R.id.cityfrom);
    TextView airto=
    (TextView) findViewById(R.id.cityto);
    TextView timetv=
    (TextView) findViewById(R.id.time);
    TextView airlinetv=
    (TextView) findViewById(R.id.airline);
    TextView flighttv=
    (TextView) findViewById(R.id.flight);
    TextView airporttotv=
    (TextView) findViewById(R.id.airportto);
    String cityto=
    getIntent().getExtras().getString("cityto");
    int indfrom=
    getIntent().getExtras().getInt("cityfrom");
    String time=
    getIntent().getExtras().getString("time");
    String airline=
    getIntent().getExtras().getString("airline");
    String flightn=
    getIntent().getExtras().getString("flight");
    String airportto=
    getIntent().getExtras().getString("airportto");
    final int indto=
    getIntent().getExtras().getInt("indto");
```

```
    final int indtof=indto-1;
    String[] city = getResources().getStringArray(R.array.city);
    String airportf = city[indfrom+1];
    airfrom.setText(airportf);
    airto.setText(cityto);
    timetv.setText(time);
    airlinetv.setText(airline);
    flighttv.setText(flightn);
    airporttotv.setText(airportto);
    final String url;
```

```
if (airline.equalsIgnoreCase("Sky Express")){
ImageView img= (ImageView) findViewById(R.id.airlineic);
img.setImageResource(R.drawable.sky_express);
url="http://www.skyexpress.gr";}
else if (airline.equalsIgnoreCase("Aegean Airlines")){
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.aegean);
    url="http://el.aegeanair.com/";}
else if (airline.equalsIgnoreCase("Olympic Air")){
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.olympic);
    url="https://www.olympicair.com/";}
else if (airline.equalsIgnoreCase("Swiss")){
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.swiss);
    url="http://www.swiss.com/ch/en";}
else if (airline.equalsIgnoreCase("RyanAir")){
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.ryanair);
    url="http://www.ryanair.com/gr/";}
```

```
else if (airline.equalsIgnoreCase("Minoan Air")){
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.minoanair);
    url="http://www.minoanair.com/";}
else if (airline.equalsIgnoreCase("Astra Airlines")){
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.astra);
    url="http://www.astra-airlines.gr/";}
else {
    ImageView img= (ImageView) findViewById(R.id.airlineic);
    img.setImageResource(R.drawable.argo);
    url="http://argoairways.blogspot.gr/";}
```

Αν για τον προορισμό που επιλέχθηκε δεν υπάρχει κάποια πτήση τότε μας εμφανίζετε στην οθόνη του Layout ένα μήνυμα που λέει ότι “δεν υπάρχουν πτήσεις από .. προς .. ! ..

Τέλος για την εξυπηρέτηση των χρηστών της εφαρμογής μας επιλέξαμε να κατασκευάσουμε πέντε κουμπιά τα οποία θα περιέχουν :τις καφετέριες,τα εστιατόρια,τα καταστήματα,τα ξενοδοχεία και την ενοικίαση οχημάτων αντίστοιχα.Ο τρόπος κατασκευής των κουμπιών είναι ο ίδιος οπότε για την παρουσίαση τους επιλέξαμε να δείξουμε το κουμπί < café >. Αρχικά κατασκευάσαμε ένα ImageButton και του δηλώσαμε έναν Listener. Με τη χρήση της μεθόδου findViewById πετυχαίνουμε την αντιστοιχία του ImageButton με αυτό του αρχείου xml. Όταν ο χρήστης επιλέξει το κουμπί τότε ενεργοποιείτε η μέθοδος onClick() και μέσω της Intent μεταβαίνουμε στην κλάση Benefits ενώ παράλληλα γίνετε και η μεταφορά δύο id με την putExtra.

```
final ImageButton cafe=
(ImageButton) findViewById(R.id.btcaffef);
cafe.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intb=
        new Intent(getApplicationContext(),Benefits.class);
        String type="cafe";
        intb.putExtra("city",airind);
        intb.putExtra("type",type);
        startActivity(intb);
    }
});
```

Από το αρχείο activity_airport.xml ο κώδικας του <ImageButton/>. Αρχικά βάζουμε το tag "<" και μετά το είδος του αντικειμένου που θα περιγράψουμε.Στη συνέχεια αρχίζουμε να προσθέτουμε διάφορα χαρακτηριστικά όπως το μέγεθος,το id,την τοποθέτηση,την εικόνα του κουμπιού την οποία παίρνουμε από τον φάκελο drawable κ.α.

```
<ImageButton
    android:layout_width="fill_parent"
    android:layout_height="80sp"
    android:id="@+id/btcaffef"
    android:layout_weight="1"
    android:src="@drawable/coffee"
    android:scaleType="center"
    android:clickable="true"
    android:layout_gravity="start"
    android:contentDescription="icon"
    android:background="@null"/>
```

Στο αρχείο Benefits.java όπου και μεταβήκαμε γίνετε η κατασκευή της καρτέλας με τις ανάλογες παροχές. Αρχικά δημιουργούμε έναν πίνακα ο οποίος παίρνει τα πεδία του από το αρχείο της βάσης το DBAdapter. Μέσω της μεθόδου findViewById κάνουμε την αντιστοιχία των ListView και TextView με αυτά του activity_benefits.xml.

```
String[] benefitname = new String[]{DBAdapter.benefits[1]};
ListView benefitslist = (ListView) findViewById(R.id.benefit);
TextView benefittv=(TextView) findViewById(R.id.benefittv);
```

Στη συνέχεια με τη χρήση της equalsIgnoreCase συγκρίνουμε τα δύο String ,το type και το όνομα του κουμπιού. Στο type καταχωρείτε κάθε φορά το όνομα του κουμπιού που έχουμε επιλέξει.Όταν αυτό περνάει στην κλάση Benefits γίνετε ο έλεγχος μέχρι η συνθήκη να βγεί αληθής. Άν βγεί αληθής τότε δημιουργούμε έναν SimpleCursorAdapter και ανάλογα με τις παραμέτρους που του δίνουμε τοποθετούνται τα στοιχεία που παίρνουμε από τη βάση δεδομένων στην καρτέλα μας.

```
else if (type.equalsIgnoreCase("food")){
    int[] viewbenefits = new int[]{R.id.restaurantn};
    SimpleCursorAdapter benefits;
    benefits = new SimpleCursorAdapter
    (this, R.layout.item_restaurants, benefit,
    benefitname, viewbenefits);
    benefitslist.setAdapter(benefits);
    benefittv.setText(getString(R.string.restaurant));
}
```

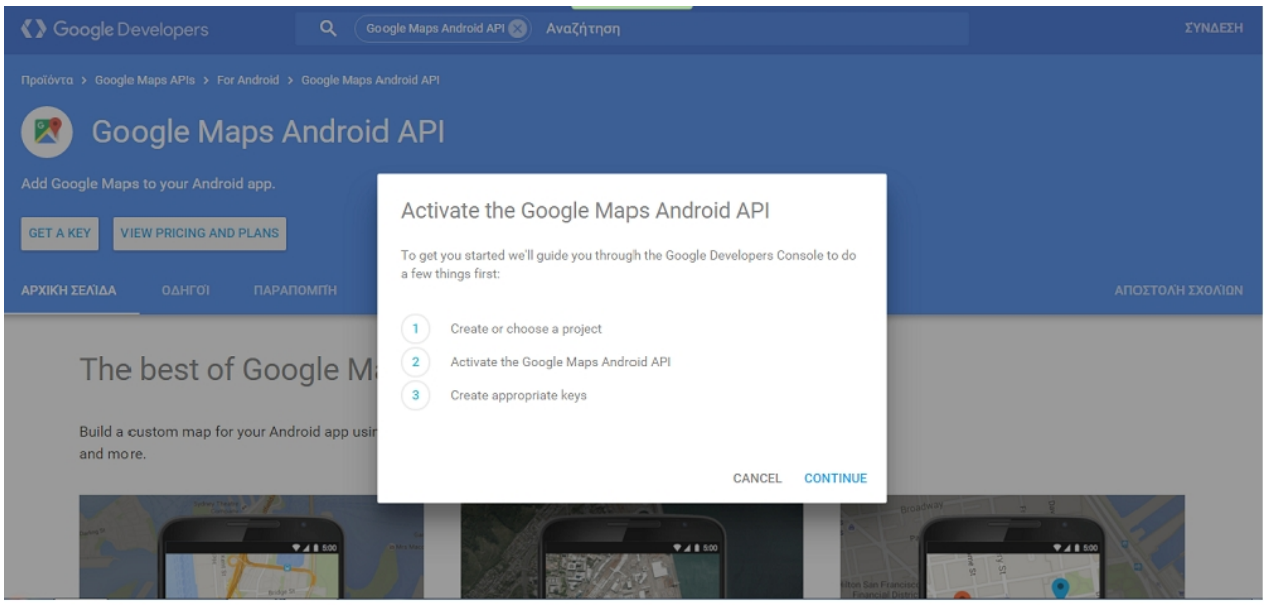
3.3 Google Maps API

3.3.1 Χρήση του χάρτη

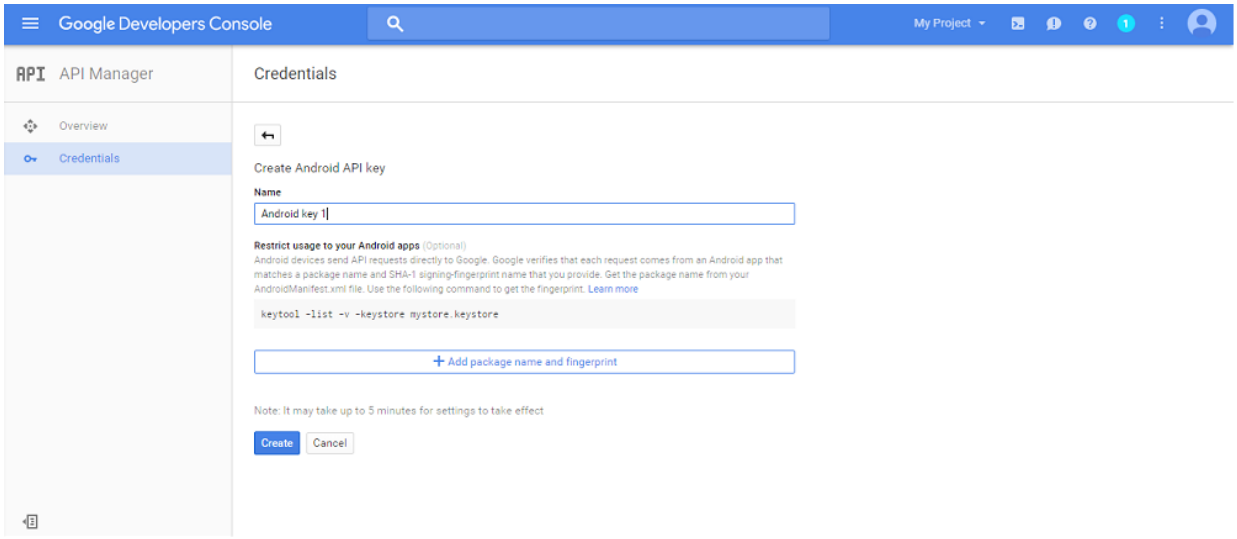
Η Google έδωσε τη δυνατότητα στους προγραμματιστές να ενσωματώσουν χάρτες της στις ιστοσελίδες-εφαρμογές που κατασκευάζουν. Αυτό έγινε μέσω του Google Maps API, όπου με το API κλειδί ο κάθε κατασκευαστής μπορεί να χρησιμοποιήσει χάρτες της Google Maps. Στην εφαρμογή μας κάναμε χρήση του χάρτη της Ελλάδος και πάνω σε αυτόν με ένα σήμα εντοπισμού θέσης δείχνουμε στον χρήστη τα σημαντικότερα αεροδρόμια της. Για να το πετύχουμε αυτό δημιουργήσαμε το αρχείο MapsActivity.java ,μέσα στο οποίο κάνουμε την προβολή του χάρτη χρησιμοποιώντας το API της Google Maps καθώς επίσης και τον εντοπισμό των αεροδρομίων με τη δημιουργία μεθόδων. Για να μπορέσουμε να ολοκληρώσουμε την εμφάνιση του χάρτη στο layout μας, θα πρέπει να πάρουμε από την Google το API κλειδί. Αν το κλειδί που θα χρησιμοποιήσουμε είναι λάθος τότε ο χάρτης μας δεν θα εμφανιστεί ποτέ.

3.3.2 Λήψη κλειδιού από τη Google

Για να μπορέσουμε να χρησιμοποιήσουμε στο layout μας τον χάρτη, θα πρέπει να πάρουμε από τη Google ένα κλειδί μέσα από το οποίο θα επιβεβαιώνει ότι κάθε αίτηση προέρχεται από μια εφαρμογή android που ταιριάζει με το όνομα πακέτου και το κλειδί.Για να πάρουμε το κλειδί μπαίνουμε στην σελίδα https://developers.google.com/maps/ documentation/android-api/ επιλέγουμε το κουμπί “Get a Key” όπου μας εμφανίζει



Αφού έχουμε δεχθεί όλους τους όρους για το API μεταβαίνουμε στη σελίδα "Google Developers Console" όπου από εκεί παίρνουμε το κλειδί μας. Στη συνέχεια συμπληρώνουμε τα παρακάτω πεδία:



3.3.3 Layout για το χάρτη

Η δημιουργία layout του χάρτη θα γίνει με διαφορετικό τρόπο σε σχέση με τα άλλα της εφαρμογής μας. Αρχικά δημιουργούμε ένα layout με όνομα "activity_maps.xml" και στη συνέχεια αντικαθιστούμε την LinearLayout με ένα Fragment. Το Fragment αντιπροσωπεύει μια συμπεριφορά ή ένα τμήμα της διεπαφής χρήστη σε μια Activity.Μέσα στο Fragment εισάγουμε τις ήδη γνωστές παραμέτρους αλλά και δύο νέες,την ".MapsActivity" οι λειτουργίες της οποίας υπάρχουν στο αρχείο "AndroidManifest.xml" καθώς επίσης και την "com.google.android.gms.maps.SupportMapFragment" έτσι ώστε να μπορέσουμε να πραγματοποιήσουμε ένα χάρτη στην εφαρμογή μας.Στον παρακάτω κώδικα φαίνεται το xml αρχείο.

```
<fragment
    xmlns:android=
    "http://schemas.android.com/apk/res/android"
    xmlns:tools=
    "http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/map"
    tools:context=".MapsActivity"
    android:name=
    "com.google.android.gms.maps.SupportMapFragment" />
```

3.3.4 Τοποθέτηση του χάρτη και εντοπισμός αεροδρομίων

Το αρχείο στο οποίο θα ασχοληθούμε με τη δημιουργία του χάρτη είναι το MapsActivity.java. Αρχικά πάνω από τη δημιουργία της κλάσης MapsActivity δηλώνουμε το πακέτο μας και στη συνέχεια κάνουμε import τις βιβλιοθήκες για να δημιουργήσουμε την κλάση.

```
package com.example.traveler.travelerassistant;

import android.content.Intent;
import android.content.res.Resources;
import android.database.Cursor;
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
```

Δημιουργήσαμε την μέθοδο onCreate μέσα από την οποία καλούμε το αρχείο activity_maps.xml,την openDB() όπου ανοίγουμε τη σύνδεση με τη βάση δεδομένων, καθώς επίσης και τις μεθόδους setUpMapIfNeeded(), και loadData().

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);
    setUpMapIfNeeded();
    openDB();
    loadData();
}
```

Παρακάτω δημιουργούμε την μεθόδο loadData. Με την Tadb.getFligthRows()

```
Cursor cursor = Tadb.getFlightRows(1);
```

Καλούμε την αντίστοιχη μέθοδο που υπάρχει στο αρχείο DBAdapter.java για να ελέγξουμε αν υπάρχουν

δεδομένα στη βάση. Αν η μέθοδος μας επιστρέψει την τιμή 0 σημαίνει ότι η βάση είναι άδεια και θα πρέπει να εισάγουμε τα δεδομένα μας. Για να πάρουμε τα δεδομένα των πτήσεων από τα xml αρχεία τα οποία είναι αποθηκευμένα στο φάκελο res δημιουργούμε τον παρακάτω κώδικα.

```
public void loadData() {
    Cursor cursor = Tadb.getFlightRows(1);
    if (cursor.getCount() == 0) {
        Resources res = getResources();
        String[] airlines =
        res.getStringArray(R.array.airlines);
        String[] times =
        res.getStringArray(R.array.times);
```

```
String[] flights =
res.getStringArray(R.array.flights);
String[] arrivals =
res.getStringArray(R.array.arrivals);
int maxindf = airlines.length;
for (int ind = 0; ind < maxindf; ind++) {
  String airliner = airlines[ind];
  String timer = times[ind];
  String flightr = flights[ind];
  String arrivalr = arrivals[ind];
  int airid = MapsActivity.this.getResources().getIdentifier
  (airliner, "array", MapsActivity.this.getPackageName());
  int timeid = MapsActivity.this.getResources().getIdentifier
  (timer, "array", MapsActivity.this.getPackageName());
  int flightid = MapsActivity.this.getResources().getIdentifier
  (flightr, "array", MapsActivity.this.getPackageName());
  int arrivalid = MapsActivity.this.getResources().getIdentifier
  (arrivalr, "array", MapsActivity.this.getPackageName());
  String[] airline = res.getStringArray(airid);
  String[] flight = res.getStringArray(flightid);
  String[] arrival = res.getStringArray(arrivalid);
  String[] time = res.getStringArray(timeid);
```

Αφού πάρουμε τα δεδομένα στη συνέχεια τα περνάμε στη βάση μας.

```
    int num = airline.length;
    for (int i = 0; i < num; i++) {
        Tadb.insertFlightRow(Tadb.dbTablesf[ind],
        airline[i], time[i], flight[i], arrival[i]);
    }
}
```

Ακριβώς με τον ίδιο τρόπο παίρνουμε και τα δεδομένα από τα xml αρχεία που αφορούν τις παροχές των αεροδρομίων και τα εισάγουμε στη βάση.

```
String[] types = res.getStringArray(R.array.sctypes);
String[] names = res.getStringArray(R.array.scnames);
int maxindsc = types.length;
for (int ind = 0; ind < maxindsc; ind++) {
    String typer = types[ind];
    String namer = names[ind];
    int typeid = MapsActivity.this.getResources().getIdentifier
    (typer, "array", MapsActivity.this.getPackageName());
    int nameid = MapsActivity.this.getResources().getIdentifier
    (namer, "array", MapsActivity.this.getPackageName());
    String[] type = res.getStringArray(typeid);
    String[] name = res.getStringArray(nameid);

    int num = type.length;
    for (int i = 0; i < num; i++) {
        Tadb.insertBenefitsRow(Tadb.dbTablessc[ind],
        type[i], name[i]);
    }
}
```

Στη συνέχεια στη μέθοδο getUpMapIfNeeded ελέγχουμε αν ο χάρτης έχει καλεστεί . Αν δεν έχουμε καλέσει τον χάρτη στην εφαρμογή μας τότε μέσω της SupportMapFragment τον παίρνουμε και καλούμε τη μέθοδο setUpMap όπου εκεί θα τοποθετήσουμε τις λειτουργίες του.

```
private void setUpMapIfNeeded() {
    if (mMap == null) {
        mMap = ((SupportMapFragment)
        getSupportFragmentManager().findFragmentById(R.id.map))
                .getMap();
        if (mMap != null) {
            setUpMap();
        }
    }
}
```

Τέλος στη μέθοδο setUpMap θα δημιουργήσουμε και θα αρχικοποιήσουμε τον χάρτη. Αρχικά καθορίζουμε τη θέση του χάρτη και στη συνέχεια δηλώνουμε τους δείκτες για τα αεροδρόμια.

```
private void setUpMap() {
    LatLng centerl = new LatLng(38.39935, 23.69633);
    mMap.moveCamera(CameraUpdateFactory
    .newLatLngZoom(centerl, (float) 5.6));
```

```
final Marker athensm;
final Marker alexandroupolim;
final Marker chaniam;
final Marker heraklionm;
final Marker ioaninam;
final Marker mykonosm;
final Marker sitiam;
final Marker skiathosm;
final Marker santorinim;
final Marker kalamatam;
final Marker kastoriam;
final Marker kavalam;
final Marker kerkyram;
final Marker kosm;
final Marker mytilinim;
final Marker rodosm;
final Marker thessalonikim;
final Marker samosm;
final Marker volosm;
```

Αφού έχουμε ολοκληρώσει με την τοποθέτηση του χάρτη και με τις δηλώσεις των δεικτών συνεχίζουμε βάζοντας τους δείκτες στο χάρτη μας. Για να γίνει αυτό αρχικά φορτώνουμε τα δεδομένα (όνομα, γεωγραφικό πλάτος, γεωγραφικό μήκος) των αεροδρομίων από το String.xml.

```
String[] aircity=getResources()
.getStringArray(R.array.airportcity);
String[] airname=getResources()
.getStringArray(R.array.airportname);
String[] lat=getResources().getStringArray(R.array.Latitude);
String[] lng=getResources().getStringArray(R.array.Longitude);
```

Στη συνέχεια σε κάθε δείκτη καταχωρούμε και τα αντίστοιχα δεδομένα.Στο παρακάτω κώδικα βλέπουμε τις καταχωρήσεις για τους πρώτους 9 δείκτες, το ίδιο κάνουμε και για τους υπόλοιπους 10.

```
alexandroupolim = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[0]),
Float.parseFloat(lng[0]))).title(aircity[0]).snippet(airname[0]));
athensm = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[1]),
Float.parseFloat(lng[1]))).title(aircity[1]).snippet(airname[1]));
chaniam = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[2]),
Float.parseFloat(lng[2]))).title(aircity[2]).snippet(airname[2]));
heraklionm = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[3]),
Float.parseFloat(lng[3]))).title(aircity[3]).snippet(airname[3]));
ioaninam = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[4]),
Float.parseFloat(lng[4]))).title(aircity[4]).snippet(airname[4]));
kalamatam = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[5]),
Float.parseFloat(lng[5]))).title(aircity[5]).snippet(airname[5]));
kastoriam = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[6]),
Float.parseFloat(lng[6]))).title(aircity[6]).snippet(airname[6]));
kavalam = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[7]),
Float.parseFloat(lng[7]))).title(aircity[7]).snippet(airname[7]));
kerkyram = mMap.addMarker(new MarkerOptions()
.position(new LatLng(Float.parseFloat(lat[8]),
Float.parseFloat(lng[8]))).title(aircity[8]).snippet(airname[8]));
```

Έπειτα αφού έχουμε τοποθετήσει τους δείκτες περνάμε το παράθυρο πληροφοριών. Όταν ο χρήστης το επιλέξει, ενεργοποιείτε η airport.java η οποία παίρνει το δείκτη της πόλης που επιλέχθηκε.(Βλέπουμε τον κώδικα για τους 9 πρώτους δείκτες)

```
mMap.setOnInfoWindowClickListener
(new GoogleMap.OnInfoWindowClickListener() {
    @Override
    public void onInfoWindowClick(Marker marker) {
        if (marker.equals(alexandroupolim)) {
            Intent int0 =
            new Intent(getApplicationContext(), airport.class);
            int0.putExtra("dbindex",0);
            startActivity(int0);}
```

```
        else if (marker.equals(athensm)) {
            Intent int1 =
            new Intent(getApplicationContext(), airport.class);
            int1.putExtra("dbindex",1);
            startActivity(int1);}
        else if (marker.equals(chaniam)) {
            Intent int2 =
            new Intent(getApplicationContext(), airport.class);
            int2.putExtra("dbindex",2);
            startActivity(int2);}
        else if (marker.equals(heraklionm)) {
            Intent int4 =
            new Intent(getApplicationContext(), airport.class);
            int4.putExtra("dbindex",3);
            startActivity(int4);}
        else if (marker.equals(ioaninam)) {
            Intent int5 =
            new Intent(getApplicationContext(), airport.class);
            int5.putExtra("dbindex",4);
            startActivity(int5);}
```

```
        else if (marker.equals(kalamatam)) {
            Intent int6 =
            new Intent(getApplicationContext(), airport.class);
            int6.putExtra("dbindex",5);
            startActivity(int6);}
        else if (marker.equals(kastoriam)) {
            Intent int7 =
            new Intent(getApplicationContext(), airport.class);
            int7.putExtra("dbindex",6);
            startActivity(int7);}
        else if (marker.equals(kavalam)) {
            Intent int8 =
            new Intent(getApplicationContext(), airport.class);
            int8.putExtra("dbindex",7);
            startActivity(int8);}
        else if (marker.equals(kerkyram)) {
            Intent int3 =
            new Intent(getApplicationContext(), airport.class);
            int3.putExtra("dbindex",8);
            startActivity(int3);}
        else if (marker.equals(kosm)) {
            Intent int9 =
            new Intent(getApplicationContext(), airport.class);
            int9.putExtra("dbindex",9);
            startActivity(int9);}
```

3.4 Βάση Δεδομένων

3.4.1 Εισαγωγή στην SQLite

Η SQLite είναι μια σχεσιακή βάση δεδομένων (RDBMS) ανοιχτού κώδικα.

Τα κύρια χαρακτηριστικά είναι :

- Δεν χρειάζεται εγκατάσταση.
- Έχει μικρές απαιτήσεις σε μνήμη, σε περίπτωση που υπάρχει διαθέσιμη περισσότερη μνήμη λειτουργεί πιο γρήγορα.
- Έχει μικρό μέγεθος, περίπου 300KB.
- Αποθηκεύεται σε ένα μοναδικό αρχείο.
- Τα δεδομένα αποθηκεύονται σε ένα φάκελο στην συσκευή τα οποία είναι σφαλή και προσβάσιμα μόνο από την συγκεκριμένη εφαρμογή.
- Αποθηκεύει διαφορετικούς τύπους δεδομένων όπως float, string, long χωρίς κάποιο πρόβλημα.
- Μπορείς να μοιραστείς τα δεδομένα με άλλες εφαρμογές αν χρησιμοποιήσεις Content Provider.

3.4.2 DBAdapter.java

Το αρχείο το οποίο είναι ο συνδετικός κρίκος των Activities με τη βάση μας είναι το DBAdapter.java. Αρχικά δημιουργούμε μια κλάση με όνομα DBAdapter μέσα στην οποία θα κάνουμε τις δηλώσεις της βάσης δεδομένων και των πινάκων. Στον παρακάτω κώδικα βλέπουμε τις δηλώσεις για τη βάση.

```
private static final String TAG = "DBAdapter";

public static final int DATABASE_VERSION = 1;

public static final String KEY_ROWID = "_id";
public static final int COL_ROWID = 0;
```

Στη συνέχεια δηλώνουμε τα πεδία των πινάκων και τους αριθμούς των πεδίων αντίστοιχα. Παρακάτω δηλώνουμε τα πεδία που θα χρησιμοποιήσουμε για τη δημιουργία του πίνακα των πτήσεων.

```
public static final String KEY_AIRLINE = "airline";
public static final String KEY_TIME = "time";
public static final String KEY_FLIGHT = "flight";
public static final String KEY_ARRIVAL = "arrival";
public static final String[] airflights =
{KEY_AIRLINE,KEY_TIME,KEY_FLIGHT,KEY_ARRIVAL};
public static final String[] airflightst = new String[]
{KEY_ROWID, KEY_AIRLINE,KEY_TIME,KEY_FLIGHT,KEY_ARRIVAL};

public static final int COL_AIRLINE = 1;
public static final int COL_TIME = 2;
public static final int COL_FLIGHT = 3;
public static final int COL_ARRIVAL=4;
```

Τέλος δηλώνουμε τα πεδία τα οποία περιέχουν όλους τους προορισμούς της εφαρμογής μας και τα τοποθετούμε στους πίνακες dbTablesf και dbTablessc αντιστοιχα

```
public static final String DATABASE_NAME = "TAdb";
public static final String DATABASE_TABLE = "Alexandroupolif";
public static final String DATABASE_TABLE2 = "Athensf";
public static final String DATABASE_TABLE3 = "Chaniaf";
public static final String DATABASE_TABLE4 = "Heraklionf";
public static final String DATABASE_TABLE5 = "Ioanninaf";
public static final String DATABASE_TABLE6 = "Kalamataf";
public static final String DATABASE_TABLE7 = "Kastoriaf";
public static final String DATABASE_TABLE8 = "Kavalaf";
public static final String DATABASE_TABLE9 = "Kerkyraf";
public static final String DATABASE_TABLE10 = "Kosf";
public static final String DATABASE_TABLE11 = "Mykonosf";
public static final String DATABASE_TABLE12 = "Mytilinif";
public static final String DATABASE_TABLE13 = "Rhodesf";
public static final String DATABASE_TABLE14 = "Samosf";
public static final String DATABASE_TABLE15 = "Santorinif";
public static final String DATABASE_TABLE16 = "Sitiaf";
public static final String DATABASE_TABLE17 = "Skiathosf";
public static final String DATABASE_TABLE18 = "Thassalonikif";
public static final String DATABASE_TABLE19 = "Volosf";
```

```
public static final String DATABASE_TABLE20 = "Alexandroupolisc";
public static final String DATABASE_TABLE21 = "Athensc";
public static final String DATABASE_TABLE22 = "Chaniasc";
public static final String DATABASE_TABLE23 = "Heraklionsc";
public static final String DATABASE_TABLE24 = "Ioanninasc";
public static final String DATABASE_TABLE25 = "Kalamatasc";
public static final String DATABASE_TABLE26 = "Kastoriasc";
public static final String DATABASE_TABLE27 = "Kavalasc";
public static final String DATABASE_TABLE28 = "Kerkyrasc";
public static final String DATABASE_TABLE29 = "Kossc";
public static final String DATABASE_TABLE30 = "Mykonossc";
public static final String DATABASE_TABLE31 = "Mytilinisc";
public static final String DATABASE_TABLE32 = "Rhodessc";
public static final String DATABASE_TABLE33 = "Samossc";
public static final String DATABASE_TABLE34 = "Santorinisc";
public static final String DATABASE_TABLE35 = "Sitiasc";
public static final String DATABASE_TABLE36 = "Skiathossc";
public static final String DATABASE_TABLE37 = "Thassalonikisc";
public static final String DATABASE_TABLE38 = "Volossc";
```

```
public static final String[] dbTablesf={DATABASE_TABLE,
DATABASE_TABLE2,DATABASE_TABLE3,DATABASE_TABLE4,
DATABASE_TABLE5,DATABASE_TABLE6,DATABASE_TABLE7,
DATABASE_TABLE8,DATABASE_TABLE9,DATABASE_TABLE10,
DATABASE_TABLE11,DATABASE_TABLE12,DATABASE_TABLE13,
DATABASE_TABLE14,DATABASE_TABLE15,DATABASE_TABLE16,
DATABASE_TABLE17,DATABASE_TABLE18,DATABASE_TABLE19};
```

```
public static final String[] dbTablessc={DATABASE_TABLE20,
DATABASE_TABLE21,DATABASE_TABLE22,DATABASE_TABLE23,
DATABASE_TABLE24,DATABASE_TABLE25,DATABASE_TABLE26,
DATABASE_TABLE27,DATABASE_TABLE28,DATABASE_TABLE29,
DATABASE_TABLE30,DATABASE_TABLE31,DATABASE_TABLE32,
DATABASE_TABLE33,DATABASE_TABLE34,DATABASE_TABLE35,
DATABASE_TABLE36,DATABASE_TABLE37,DATABASE_TABLE38};
```

Αφού έχουμε κάνει τις δηλώσεις των πεδίων αρχίζουμε να κατασκευάζουμε τους πίνακες που θα χρησιμοποιήσουμε στη βάση μας. Αρχικά δημιουργούμε έναν πίνακα τον dbcrstr με μέγεθος τόσο όσο είναι το άθροισμα των dbTablesf και dbTablessc. Σε κάθε θέση του dbcrstr θα αποθηκεύσουμε ένα πίνακα, στις πρώτες 19 θέσεις θα αποθηκεύσουμε τους πίνακες των πτήσεων και στις επόμενες 19 τους πίνακες των παροχών που προσφέρει κάθε αεροδρόμιο.

```
public static String[] dbCreate(){
    int fnum=dbTablesf.length;
    int scnum=dbTablessc.length;
    String[] dbcrstr = new String[fnum+scnum];
```

Στον παρακάτω κώδικα πετυχαίνουμε μέσω της for τη δημιουργία των πρώτων 19 πινάκων, όπου κάθε πίνακας παίρνει ως όνομα ένα προορισμό και ως πεδία τα (airline,time,flight,arrival).

```
for (int indf=0;indf<fnum;indf++){
dbcrstr[indf] = "create table " + dbTablesf[indf]
    + " (" + KEY_ROWID + " integer primary key autoincrement, "
    + airflights[0] + " text not null, "
    + airflights[1] + " time not null, "
    + airflights[2] + " string not null, "
    + airflights[3] + " text not null"
    + ");";}
```

Ακριβώς την ίδια υλοποίηση κάνουμε και για την κατασκευή των επόμενων 19 πινάκων που θα περιέχουν τις παροχές των αεροδρομίων. Οι πίνακες παίρνουν ως όνομα ένα προορισμό και ως πεδία τα(type,name).

```
for (int indsc=0;indsc<scnum;indsc++){
dbcrstr[fnum+indsc] = "create table " + dbTablessc[indsc]
    + " (" + KEY_ROWID + " integer primary key autoincrement, "
    + benefits[0] + " text not null, "
    + benefits[1] + " text not null"
    + ");";}
```

Στη συνέχεια κάνουμε μια νέα κλάση την DBhandler η οποία έχει ως υπερκλάση την SQLiteOpenHelper και είναι εμφωλευμένη της DBAdapter.

```
private static class DBhandler extends SQLiteOpenHelper {
    DBhandler(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

Μέσα στην οποία ολοκληρώνουμε τη δημιουργία των πινάκων που θα υπάρχουν στη βάση μας μέσω της μεθόδου onCreate.

```
public void onCreate(SQLiteDatabase _db) {
    String[] dbcrstr=dbCreate();
    int exnum=dbcrstr.length;
    for (int ex=0;ex<exnum;ex++){
    _db.execSQL(dbcrstr[ex]);
    }
}
```

Στη συνέχεια για την ολοκλήρωση της βοηθητικής μας κλάσης myDBHandler χρησιμοποιήσαμε τη μέθοδο onUpgrade.

```
public void onUpgrade(SQLiteDatabase _db,
int oldVersion, int newVersion){
    Log.w(TAG, "Upgrading application's database from version"
    + oldVersion + " to " + newVersion +
    ", which will destroy all old data!");

    int dbf=dbTablesf.length;
    for (int ind=0;ind<dbf;ind++){
    _db.execSQL("DROP TABLE IF EXISTS " + dbTablesf[ind]);}

    int dbsc=dbTablessc.length;
    for (int ind=0;ind<dbsc;ind++){
    _db.execSQL("DROP TABLE IF EXISTS " + dbTablessc[ind]);}

    onCreate(_db);
}
```

Συνεχίζουμε με την δημιουργία της βάσης,και αφού έχουμε τελειώσει με τις δηλώσεις και τις κατασκευές των πινάκων, θα δημιουργήσουμε τις μεθόδους που θα αποτελέσουν τη λειτουργία της. Αρχικά η μέθοδος open θα ανοίγει τη βάση για την εισαγωγή δεδομένων και η μέθοδος close θα την κλείνει.

```
 private final Context context;
 private DBhandler myDBHandler;
 private SQLiteDatabase db;

 public DBAdapter(Context ctx) {
     this.context = ctx;
     myDBHandler = new DBhandler(context);
 }

 public DBAdapter open() {
     db = myDBHandler.getWritableDatabase();
     return this;
 }

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

Στον παρακάτω κώδικα με τη δημιουργία της μεθόδου insertFlightRow παίρναμε τα πεδία των πτήσεων στη βάση δεδομένων. Αρχικά με την put τα τοποθετούμε με τη σειρά που θέλουμε και στη συνέχεια τα βάζουμε στη βάση.

```
public long insertFlightRow(String dbasetable, String airline,
    String time, String flight, String arrival) {

    ContentValues initialValues = new ContentValues();
    initialValues.put( airflights[0], airline);
    initialValues.put( airflights[1], time);
    initialValues.put( airflights[2], flight);
    initialValues.put( airflights[3], arrival);

    return db.insert(dbasetable, null, initialValues);
}
```

Το ίδιο κάνουμε και για τα πεδία των παροχών των αεροδρομίων. Δημιουργήσαμε τη μέθοδο insertBenefitsRow και τα περάσαμε στη βάση δεδομένων.

```
public long insertBenefitsRow(String dbasetable,
    String type, String name) {

    ContentValues initialValues = new ContentValues();
    initialValues.put( benefits[0], type);
    initialValues.put( benefits[1], name);

    return db.insert(dbasetable, null, initialValues);
}
```

Στη συνέχεια δημιουργήσαμε 4 μεθόδους μέσω των οποίων μπορούμε να διαχειριστούμε τα δεδομένα της βάσης μας. Τη μέθοδο getAllFlightRows για να μπορούμε σύμφωνα με τις παραμέτρους να πάρουμε όλες τις πτήσεις που υπάρχουν για τον προορισμό που επέλεξε ο χρήστης. Η μέθοδος αυτή καλείτε από το αρχείο chooseflight.java:

```
Cursor flight=Tadb.getAllFlightRows(Tadb.dbTablesf[indfrom],cityto);
```

και παίρνει ως παραμέτρους τις δυο πόλεις προορισμού. Τη δεύτερη μέθοδο getBenefitsRows τη δημιουργήσαμε για να μπορέσουμε μέσω των παραμέτρων της, να πάρουμε όλες τις παροχές του αεροδρομίου ανάλογα με το κουμπί παροχών που θα επιλέξει ο χρήστης. Η μέθοδος αυτή καλείτε από το αρχείο Benefits.java:

```
Cursor benefit=Tadb.getBenefitsRows(Tadb.dbTablessc[airind],type);
```

και παίρνει ως παραμέτρους την πόλη που έχει επιλεγεί και τον τύπο του κουμπιού. Στη συνέχεια με τη χρήση της μεθόδου getFlightRows μπορούμε να δούμε αν υπάρχουν στοιχεία στη βάση μας. Η μέθοδος αυτή καλείτε από το αρχείο MapsActivity.java:

```
  Cursor cursor = Tadb.getFlightRows(1);
```

και παίρνει ως παράμετρο ένα id. Διευκρινίζουμε ότι αν η βάση μας είναι άδεια το id είναι μηδέν. Τέλος η τέταρτη μέθοδός μας είναι η getSingleFlightRow μέσα από τη οποία μπορούμε να πάρουμε όποιο στοιχείο θέλουμε από τη βάση. Η μέθοδος αυτή καλείτε από το αρχείο chooseflight.java:

```
Cursor flight=Tadb.getSingleFlightRow(Tadb.dbTablesf[indfrom],rowid);
```

και ως παραμέτρους παίρνει την πόλη αναχώρησης και το id

```
public Cursor getAllFlightRows(String dbasetable,
 String cityto) {
    String where=KEY_ARRIVAL+" IN ('"+cityto+"')";
    Cursor c =  db.query(true, dbasetable, airflightst,
            where, null, null, null, null, null);
    if (c != null) {
        c.moveToFirst();
    }
    return c;
}
```

```
public Cursor getBenefitsRows(String dbasetable,
 String type) {
    String where=KEY_TYPE+" IN ('"+type+"')";
    Cursor c =  db.query(true, dbasetable, benefitst,
            where, null, null, null, null, null);
    if (c != null) {
        c.moveToFirst();
    }
    return c;
}
```

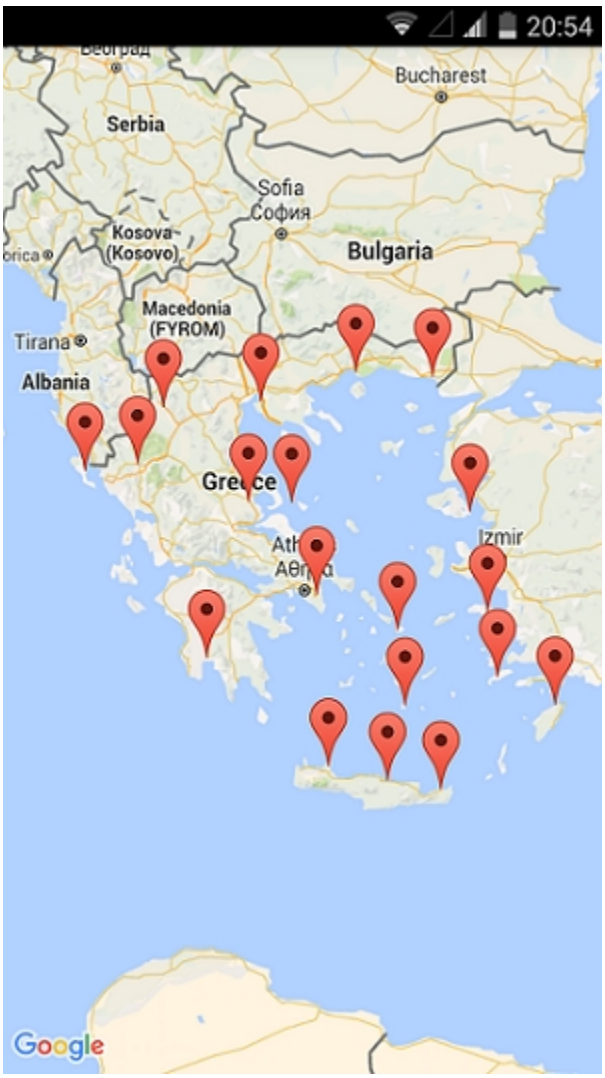
```
public Cursor getFlightRows(long rowId) {
    String where = KEY_ROWID + "=" + rowId;
    Cursor c =  db.query(true, DATABASE_TABLE, airflightst,
            where, null, null, null, null, null);
    if (c != null) {
        c.moveToFirst();
    }
    return c;
}

public Cursor getSingleFlightRow(String dbasetable,
 long rowId) {
    String where = KEY_ROWID + "=" + rowId;
    Cursor c =  db.query(true, dbasetable, airflightst,
            where, null, null, null, null, null);
    if (c != null) {
        c.moveToFirst();
    }
    return c;
}
```

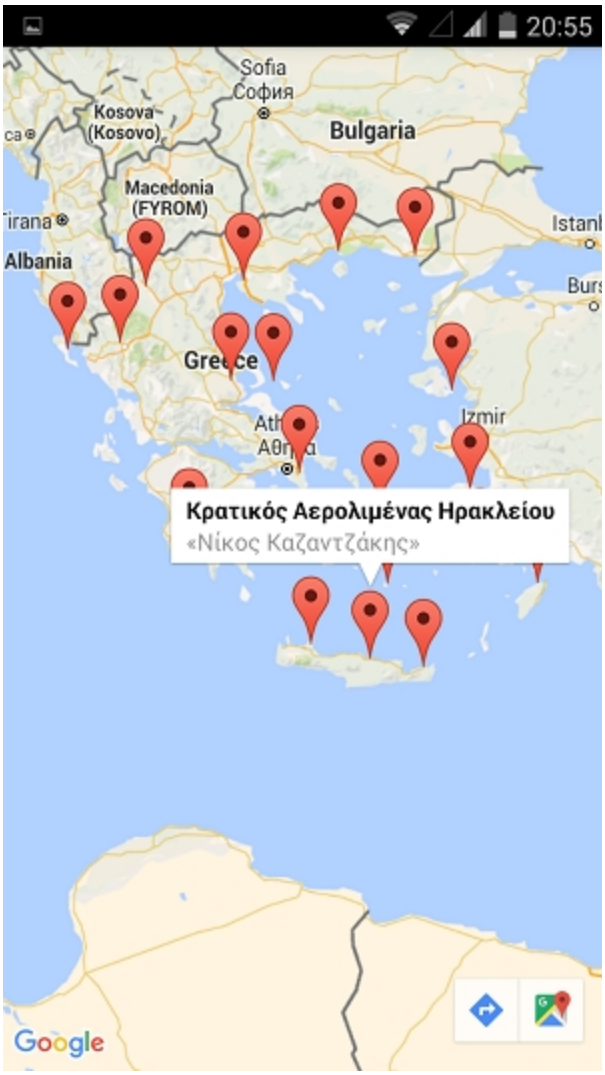
4 Χρήση της εφαρμογής

Στο πρώτο Layout της εφαρμογής ο χρήστης βλέπει τον χάρτη της Ελλάδος και 19 από τα πιο σημαντικά αεροδρόμια της χώρας.

Στο επόμενο Layout γίνετε η επιλογή του αεροδρόμιου αναχώρησης, στην περίπτωση μας διαλέξαμε το αεροδρόμιο Ηρακλείου.



Εικόνα 22: Πρώτη οθόνη - Χάρτης Ελλάδος

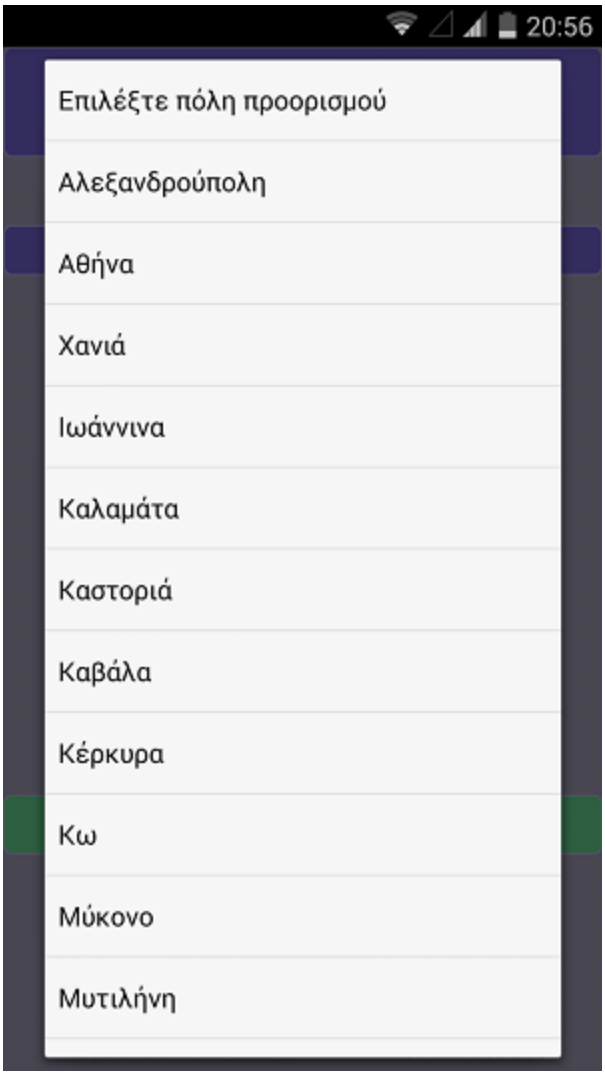


Εικόνα 23: Επιλογή αεροδρομίου

Έχουμε μεταβεί στο Αεροδρόμιο Ηρακλείου και εδώ μας δίνετε η δυνατότητα να επιλέξουμε είτε την πόλη προορισμού είτε τις παροχές του αεροδρομίου.

Αν επιλέξαμε το button “επέλεξε πόλη προορισμού” τότε μας εμφανίζεται η λίστα με όλα τα αεροδρόμια με τους πιθανούς προορισμούς.

Εικόνα 24: Παροχές αεροδρομίου

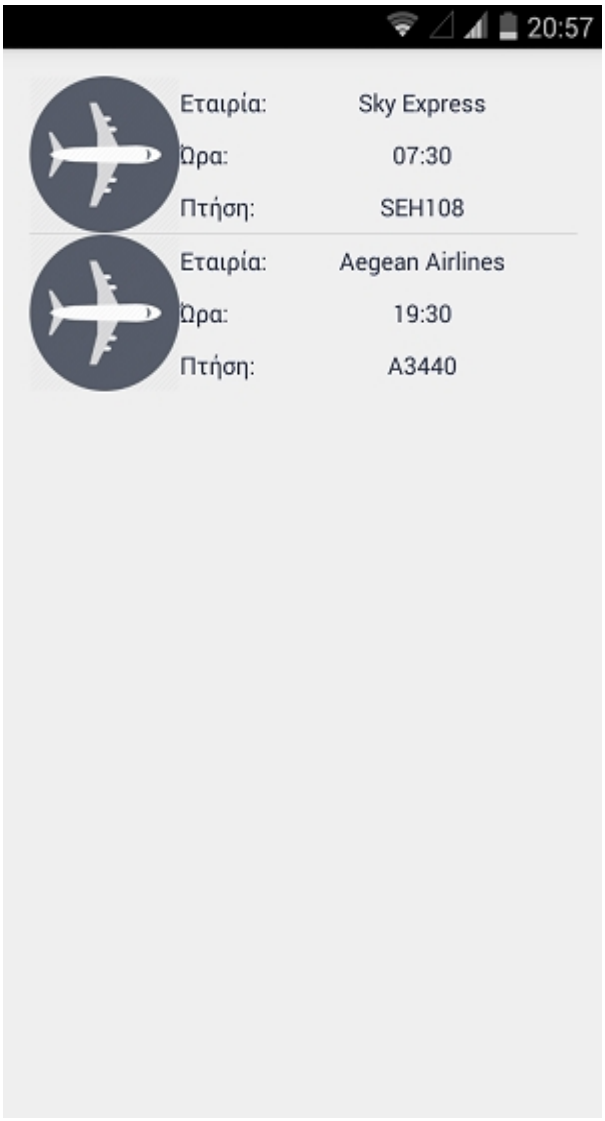


Εικόνα 25: Λίστα αεροδρομίων

Από τη λίστα των προορισμών επιλέγουμε τυχαία την Αθήνα. Αφού γίνει η επιλογή εμφανίζετε στην οθόνη μας η πόλη και το αεροδρόμιο άφιξης.

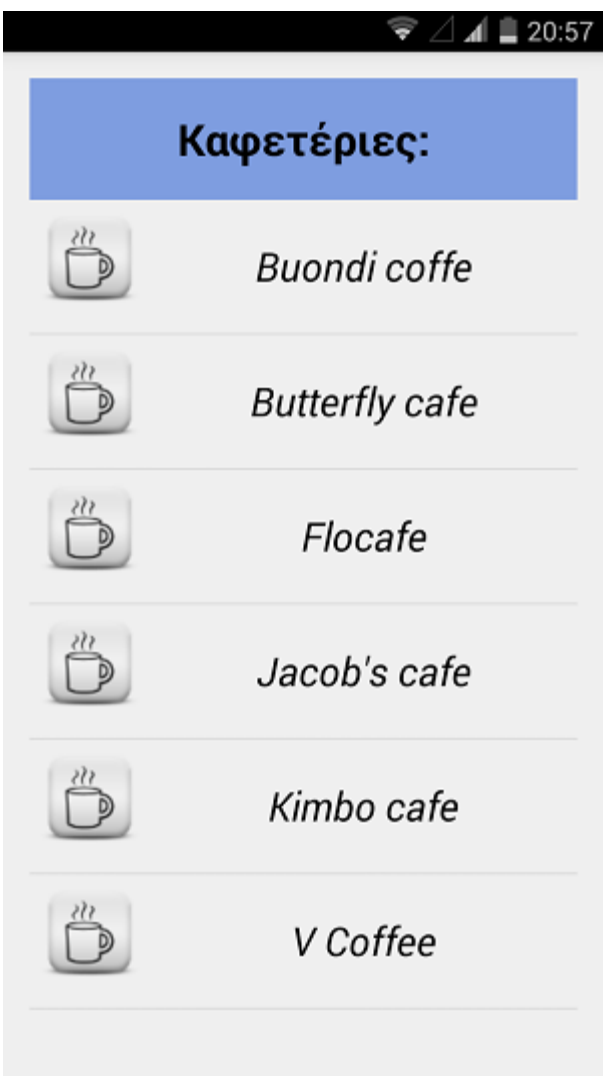
Στη συνέχεια επιλέγοντας το κουμπί “Εύρεση πτήσης” μας εμφανίζετε η καρτέλα με όλες τις εταιρίες που πραγματοποιούν δρομολόγια από Ηράκλειο με προορισμό την Αθήνα. Αν επιλέξουμε κάποιον προορισμό για τον οποίο δεν υπάρχει απευθείας πτήση τότε το σύστημα μας ενημερώνει με ένα μήνυμα.

Εικόνα 26: Διεπαφή χρήστη για εύρεση πτήσης

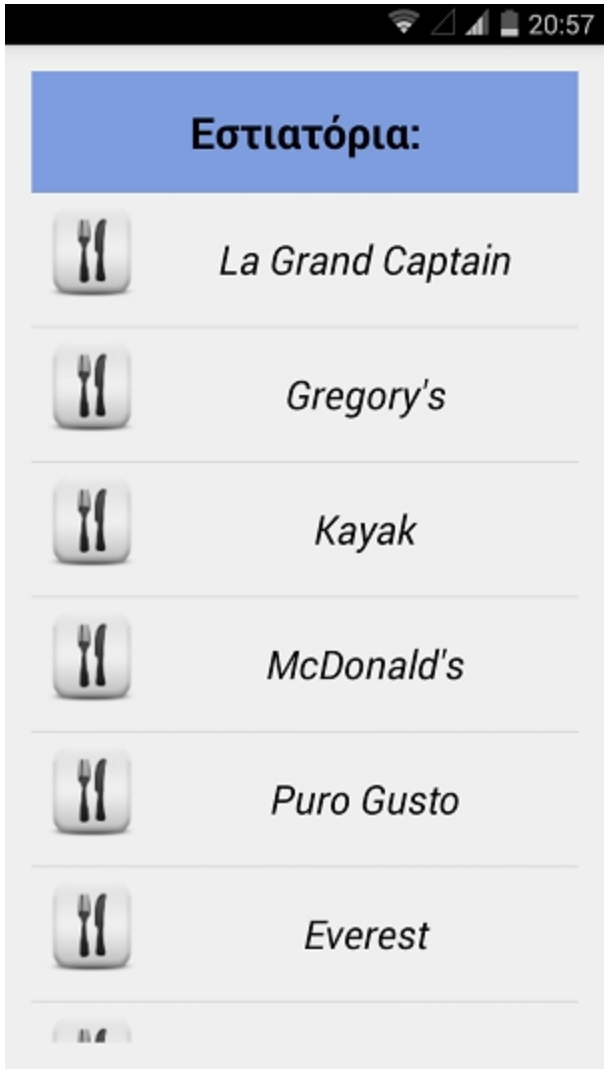


Εικόνα 27: Διαθέσιμες πτήσεις

Αν επιλέξουμε κάποιο από τα κουμπιά παροχών της εικόνας, τότε ανάλογα με την επιλογή που έγινε μας εμφανίζετε η αντίστοιχη καρτέλα. Παρακάτω βλέπουμε τα εστιατόρια και τις καφετέριες που βρίσκονται στο χώρο του αεροδρομίου.



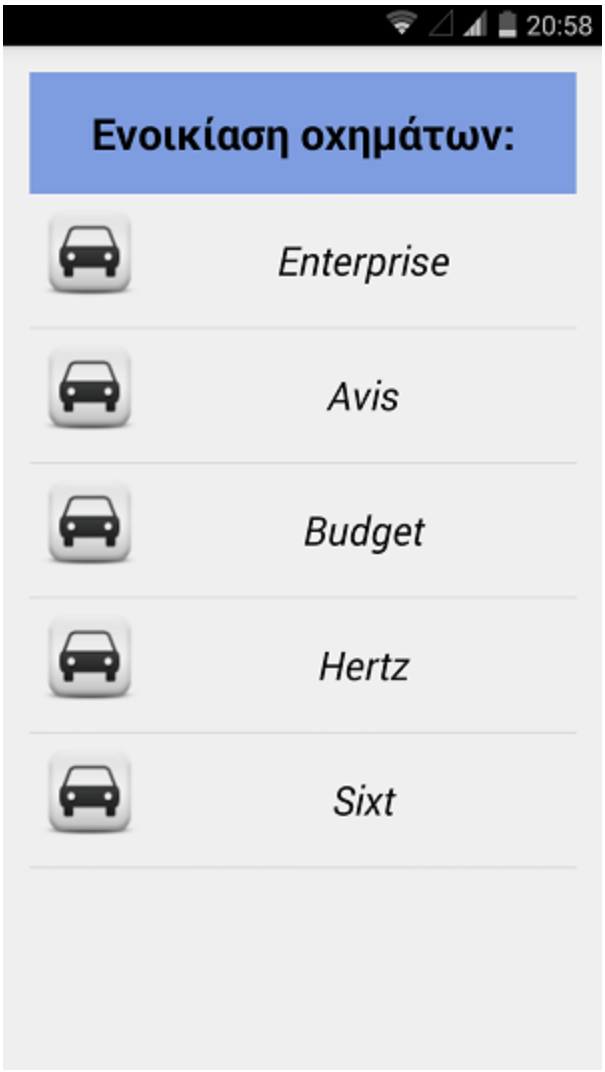
Εικόνα 29: Διεπαφή χρήστη για καφετέριες



Εικόνα 28: Διεπαφή χρήση για εστιατόρια

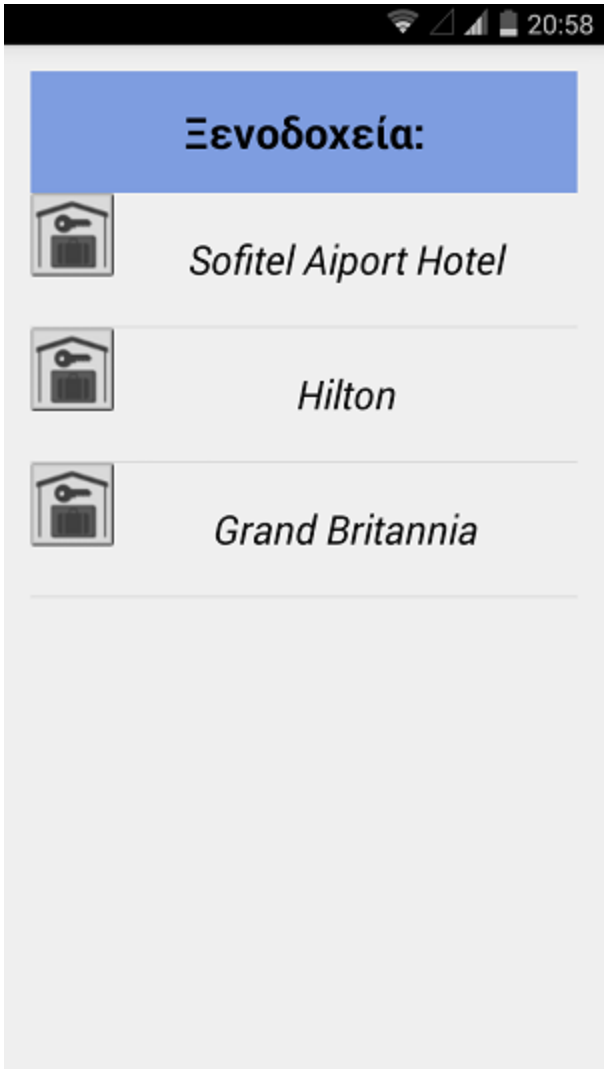
Στη συνέχεια βλέπουμε τις καρτέλες των καταστημάτων και των γραφείων ενοικίασης αυτοκινήτων.

Εικόνα 30: Διεπαφή χρήστη για καταστήματα



Εικόνα 31: Διεπαφή χρήστη για ενοικίαση αυτοκινήτου

Τέλος βλέπουμε την καρτέλα με τα κοντινά ξενοδοχεία.



Εικόνα 32: Διεπαφή χρήστη για ξενοδοχεία

5 Βιβλιογραφία

Επίσημη σελίδα Android
http://developer.android.com/index.html

Stack Overflow
http://developer.android.com/index.html

Wikipedia
https://en.wikipedia.org/

Google
https://www.google.com

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

Δωρεάν βοηθητικά μαθήματα
https://www.edx.org/
https://www.coursera.org/
https://www.edx.org/

Android Tutorial
https://www.youtube.com/
http://www.instructables.com/
http://www.vogella.com/tutorials/android.html
http://www.tutorialspoint.com/android/
https://www.udemy.com/

Βιβλία

Wallace Jackson, " Android Apps for Absolute Beginners " - Third Edition - Apress

Paul Deitel, Harvey Deitel, Abbey Deitel "Android Προγραμματισμος" - Δεύτερη Έκδοση - Εκδόσεις Μ.Γκιούρδας

Mike van Drongelen, " Android Studio Cookbook " - Packt Publishing