Πανεπιστήμιο Πελοποννήσου
Τμήμα Επιστήμης και Τεχνολογίας Τηλεπικοινωνιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"*Σχεδίαση και ανάπτυξη εφαρμογής ενημέρωσης χαρτών για κινητές συσκευές λειτουργικού συστήματος Android*"

Επιβλέπων καθηγητής : κ. Τσελίκας Νικόλαος

Ονοματεπώνυμο : Γαλιώτου Κάλλια
Α.Μ. : 2011103

Τρίπολη, Απρίλιος 2013

Πίνακας περιεχομένων

Εισαγωγή

Καθώς η ανάπτυξη των φορητών συσκευών ολοένα και αυξάνεται αντίστοιχα πληθαίνουν και οι ανάγκες που προκύπτουν. Οι χρήστες των φορητών συσκευών αποζητούν ολοένα και περισσότερες υπηρεσίες προκειμένου να ικανοποιήσουν τις ανάγκες αυτές. Αποτέλεσμα των παραπάνω, είναι να υπάρχουν χιλιάδες εφαρμογές κάθε μία από τις οποίες προσφέρει μία υπηρεσία. Οι σημερινές συσκευές κινητής τηλεφωνίας παρέχουν στους χρήστες λειτουργίες και δυνατότητες με σκοπό να καλύψουν ένα μεγάλο μέρος των καθημερινών αναγκών τους όπως: ψυχαγωγία, διασκέδαση, εργασία, άμεση πρόσβαση σε δεδομένα, ικανότητα σύνδεσης στο διαδίκτυο και σχεδόν κάθε είδους επικοινωνία. Η σημασία και η χρησιμότητα των εφαρμογών που σχετίζονται άμεσα με τους χάρτες είναι αδιαμφισβήτητη. Οι χρήστες τέτοιων εφαρμογών έχουν τη δυνατότητα να βρίσκουν την τρέχουσα τοποθεσία τους, να προβάλουν τις κυκλοφοριακές συνθήκες σε πραγματικό χρόνο, να λαμβάνουν λεπτομερείς οδηγίες κατεύθυνσης με τα πόδια ,χρησιμοποιώντας δημόσια συγκοινωνία αλλά και με άλλα μέσα μεταφοράς καθώς και να πλοηγούνται χρησιμοποιώντας προφορικές οδηγίες στροφή προς στροφή μέσω της υπηρεσίας Πλοήγηση Χαρτών Google (Beta). Η εύρεση διευθύνσεων, αξιοθέατων ή επιχειρήσεων απευθείας σε έναν οδικό χάρτη ή σε μια δορυφορική εικόνα είναι μερικές ακόμα πολύ σημαντικές δυνατότητες των εφαρμογών πλοήγησης χαρτών της Google. Με την εφαρμογή Google Latitude οι χρήστες μπορούν να προβάλουν την τοποθεσία που βρίσκονται φιλικά τους πρόσωπα και να μοιράζονται μαζί τους την δική τους τοποθεσία ενώ μέσω της διεπαφής Google Places παρέχεται η δυνατότητα παροχής πληροφοριών για κοντινές τοποθεσίες λαμβάνοντας υπόψη το latitude και longtitude του χρήστη καθώς και την ακτίνα της περιοχής ενδιαφέροντος. Λόγω της μεγάλης εξέλιξης των εφαρμογών πλοήγησης σε Android λειτουργικό σύστημα, το ενδιαφέρον για την ανάπτυξη τέτοιων εφαρμογών συνεχώς μεγαλώνει.

Κεφάλαιο 1

Περιγραφή και σκοπός της εργασίας

Σκοπός της διπλωματικής αυτής εργασίας είναι ο σχεδιασμός και η ανάπτυξη μιας εφαρμογής σε περιβάλλον Android, η λειτουργία της οποίας επικεντρώνεται στη δυνατότητα ενημέρωσης των χρηστών για την ακριβή θέση διάφορων σημείων ενδιαφέροντος πάνω στον χάρτη καθώς και την εμφάνιση επιπλέον πληροφορίας για τα σημεία επιλογής τους. Η χώρα για την οποία έχει υλοποιηθεί η εφαρμογή είναι η Ισπανία. Οι χρήστες έχουν τη δυνατότητα χρησιμοποιώντας την εφαρμογή αυτή να επιλέγουν κάποια ή κάποιες κατηγορίες από αξιοθέατα και να εμφανίζεται στον χάρτη η ακριβής τους θέση. Επιπλέον, δίνεται η δυνατότητα εμφάνισης επιπλέον πληροφορίας ξεχωριστά για το κάθε ένα σημείο επιλογής του χρήστη. Η διαχείριση των σημείων ενδιαφέροντος γίνεται μέσω μιας βάσης δεδομένων χρησιμοποιώντας έναν τοπικό εξυπηρετητή. Η υλοποίηση έγινε χρησιμοποιώντας το Eclipse, ένα πρόγραμμα ανοιχτού κώδικα που σε συνεργασία με τα δωρεάν εργαλεία ανάπτυξης Android αποτελεί το επίσημο εργαλείο ανάπτυξης για εφαρμογές Android.

Κεφάλαιο 2

Ανάλυση αρχιτεκτονικής λειτουργικού συστήματος Android

2.1 Γενικές Πληροφορίες

Οι έξυπνες συσκευές (Smart phones) βασίζονται σε κάποιο λειτουργικό σύστημα δίνοντας τη δυνατότητα στους προγραμματιστές να μπορούν να αναπτύσσουν χρήσιμες και πρωτότυπες εφαρμογές. Τα λειτουργικά συστήματα (OS) που χρησιμοποιούνται από τα σύγχρονα smartphones περιλαμβάνουν το Android της Google, το iOs της Apple, το Symbian της Nokia, το BlackBerry OS της RIM, το Bada της Samsung, το Windows Phone της Microsoft, το webOS της Hewlett-Packard, τα Maemo και MeeGo της Linux, κ.α. Τα λειτουργικά συστήματα μπορούν να εγκατασταθούν σε πολλά διαφορετικά μοντέλα κινητών τηλεφώνων και, συνήθως, κάθε συσκευή μπορεί να λάβει πολλές ενημερωμένες εκδόσεις λογισμικού λειτουργικού συστήματος κατά τη διάρκεια ζωής της. Το Android είναι μια ολοκληρωμένη, ανοιχτή και ελεύθερη πλατφόρμα για κινητές συσκευές όπως smart phones και tablets. Περιλαμβάνει ένα λειτουργικό σύστημα (OS) βασισμένο στον πυρήνα Linux, το απαραίτητο ενδιάμεσο λογισμικό, βιβλιοθήκες και βασικές εφαρμογές. Η πλατφόρμα αυτή επιτρέπει στους προγραμματιστές να γράφουν διαχειρίσιμο κώδικα σε γλώσσα προγραμματισμού Java, ελέγχοντας τη συσκευή μέσω βιβλιοθηκών λογισμικού, οι οποίες έχουν αναπτυχθεί από την Google, ενώ πολλά από τα στοιχεία του λειτουργικού συστήματος είναι γραμμένα σε γλώσσα προγραμματισμού C ή C++.

Ιστορικά στοιχεία

Τον Ιούλιο του 2005, η Google εξαγόρασε την Android Inc, μια μικρή εταιρεία με έδρα το Palo Alto στην California των ΗΠΑ. Εκείνη την εποχή ελάχιστα ήταν γνωστά για τις λειτουργίες της Android Inc, εκτός του ότι ανέπτυσσαν λογισμικό για κινητά τηλέφωνα. Έτσι ξεκίνησε η φημολογία περί σχεδίων της Google προκειμένου να διεισδύσει στην αγορά κινητής τηλεφωνίας. Η παρουσίαση της πλατφόρμας Android στις 5 Νοεμβρίου 2007, έγινε παράλληλα με την ανακοίνωση της ίδρυσης

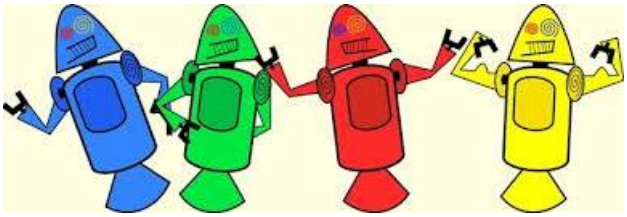
του οργανισμού Open Handset Alliance (OHA), μιας κοινοπραξίας αποτελούμενης από 48 εταιρείες λογισμικού (Intel, NVIDIA, Motorola, Qualcomm, Sprint Nextel κ.α.), κατασκευής υλικού και τηλεπικοινωνιών που στόχο είχαν την ανάπτυξη ανοιχτών προτύπων για κινητές συσκευές. Το πρώτο εμπορικά διαθέσιμο κινητό που έτρεξε με λειτουργικό σύστημα Android ήταν το HTC Dream και κυκλοφόρησε τον Οκτώβριο του 2008. Δυο χρόνια αργότερα, η Google ξεκίνησε τη σειρά Google Nexus. Η Google δημοσίευσε το μεγαλύτερο μέρος του κώδικα του Android υπό τους όρους της Apache License, μιας άδειας ελεύθερου λογισμικού/λογισμικού ανοιχτού κώδικα (ΕΛ/ΛΑΚ).

Η μασκότ του android

Το σήμα κατατεθέν όλων των ενεργειών που σχετίζονται με την πλατφόρμα Android είναι το ευρέως γνωστό πράσινο ανδροειδές ρομποτάκι.

Σχήμα 1: Λογότυπο του Android

Το «bugdroid», όπως ονομάζεται, σχεδιάστρια του οποίου είναι η Irina Blok, δεν ήταν το πρώτο σχέδιο για την μασκότ του λειτουργικού της Google. Πριν από έξι χρόνια κάποιες πρώιμες μασκότ είχαν σχεδιαστεί για να χαρακτηρίζουν την πλατφόρμα του Android, οι οποίες όμως δεν ενθουσίασαν το κοινό στην εσωτερική παρουσίαση που έγινε και απορρίφθηκαν.



Σχήμα 2:Οι πρώτες μασκότ του Android «Dandroids»

Οι εκδόσεις του Android

Το Android έχει υποστεί μια σειρά ενημερώσεων μετά από την αρχική κυκλοφορία του. Η καθεμία από αυτές διορθώνει σφάλματα και προσθέτει νέα χαρακτηριστικά. Η πρώτη έκδοση του λειτουργικού συστήματος η οποία κυκλοφόρησε το 2007 ήταν η Beta έκδοση του Android. Από τον Απρίλιο του 2009, κάθε έκδοση έχει και ένα "κωδικό-όνομα" βασισμένο σε ονόματα γλυκών. Αλφαβητικά, τα ονόματα που έχουν παρουσιαστεί ως τώρα είναι Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich και Jelly Bean. Η έκδοση Jelly Bean εμφανίστηκε στο κοινό στα μέσα του 2012.

Σχήμα 3: Logo και ονομασίες των εκδόσεων του Android

2.2 Χαρακτηριστικά Android

Η πλατφόρμα Android υποστηρίζει ένα σύνολο από χαρακτηριστικά. Η περιήγηση στο διαδίκτυο γίνεται εύκολα με τον ενσωματωμένο φυλλομετρητή (Web Browser), ο οποίος είναι βασισμένος στην ανοιχτή τεχνολογία WebKit. Για την αποθήκευση δεδομένων γίνεται χρήση της βάσης δεδομένων SQLite, ενώ υποστηρίζονται όλες οι σύγχρονες μορφές ήχου, εικόνας και βίντεο. Επιπλέον, η πλατφόρμα περιλαμβάνει βιβλιοθήκες υποστήριξης 2D γραφικών καθώς επίσης υποστηρίζει την απεικόνιση 3D γραφικών βασισμένα σε OpenGL ES 1.0 έκδοσης. Ως προς τη συνδεσιμότητα, οι τεχνολογίες που υποστηρίζονται είναι οι GSM/EDGE, CDMA, EV-DO, UMTS, Bluetooth, και Wi-Fi, ενώ για την ανάπτυξη του λογισμικού περιλαμβάνεται προσομοιωτής συσκευής, εργαλεία διόρθωσης σφαλμάτων, μνήμη και εργαλεία ανάλυσης της απόδοσης του εκτελέσιμου λογισμικού. Τα σημαντικότερα χαρακτηριστικά θα τα αναλύσουμε στη συνέχεια.

2.2.1 Android Software Development Kit (Android SDK)

Η εργαλειοθήκη ανάπτυξης λογισμικού για Android εφαρμογές περιλαμβάνει έναν εξομοιωτή, εργαλεία για την παρουσίαση των χαρακτηριστικών των εφαρμογών και εργαλεία αποσφαλμάτωσης. Το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse είναι η καλύτερη επιλογή για τους προγραμματιστές Android εφαρμογών. Το εργαλείο ανάπτυξης Android εφαρμογών (Android Development Tool - ADT) είναι ένα πρόσθετο εργαλείο, το οποίο χρησιμοποιείται για να αυξήσει και να υποστηρίξει το περιβάλλον παρουσίασης του Eclipse. Το ADT παρέχει έναν γρηγορότερο και ευκολότερο τρόπο ανάπτυξης και αποσφαλμάτωσης μιας Android εφαρμογής.

2.2.2 Εικονική μηχανή Dalvik

Η εικονική μηχανή Dalvik είναι ειδικά σχεδιασμένη για την πλατφόρμα Android και βελτιστοποιήθηκε για τις κινητές συσκευές, οι οποίες αντιμετωπίζουν περιορισμούς ως προς τους πόρους τους (όπως χαμηλή χωρητικότητα μνήμης, μικρό μέγεθος μνήμης και χαμηλή υπολογιστική ισχύ). Η Dalvik είναι μια εικονική μηχανή βασισμένη σε καταχωρητές (registers) και ο μεταγλωττιστής της έχει βελτιστοποιηθεί προκειμένου να έχουμε γρηγορότερη εκτέλεση. Η μηχανή αυτή έχει την ικανότητα να εκτελεί προγράμματα γραμμένα σε γλώσσα προγραμματισμού Java.

2.2.3 Υποστήριξη γραφικών

Το λειτουργικό σύστημα Android υποστηρίζει πολλές διαφορετικές μορφές αρχείων εικόνας συμπεριλαμβανομένων των jpeg, bmp, gif, png και άλλες. Οι τεχνικές κωδικοποίησης video που υποστηρίζονται από το Android είναι οι H.263 και H.264. Για την υποστήριξη τηλεδιάσκεψης χρησιμοποιείται η τεχνική H.263 ενώ για την υποστήριξη υψηλού επιπέδου συμπίεσης (πρότυπο MPEG4) η τεχνική H.264. Όσον αφορά τη συμπίεση αρχείων ήχου υποστηρίζονται οι μορφές mp3, wav, amr και amr-wb. Επίσης, κάποια άλλα χαρακτηριστικά που υποστηρίζονται από το περιβάλλον Android είναι το GPS, μετρητής επιτάχυνσης, πυξίδα. Η εκτέλεση των χαρακτηριστικών αυτών διαφέρει από συσκευή σε συσκευή.

2.2.4 SQLite

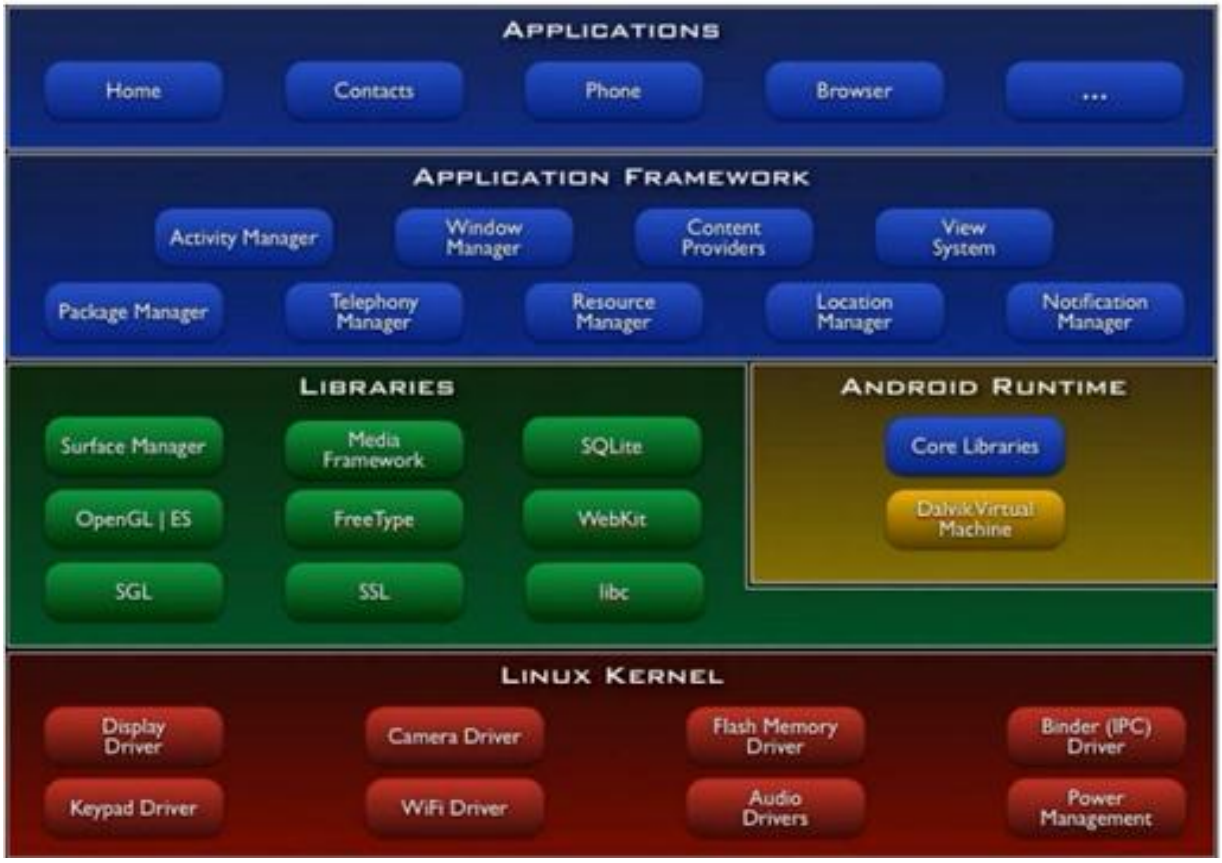
Η SQLite είναι μια ανοιχτού κώδικα βάση δεδομένων, η οποία είναι ενσωματωμένη στο λειτουργικό σύστημα Android. Έτσι, το Android χρησιμοποιεί ένα μικρού μεγέθους σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDMS-Relational Database Management System), περίπου ίσο με 500Kb, για την αποθήκευση δεδομένων. Επίσης, χρησιμοποιεί ένα ενιαίο αρχείο για την καταχώρηση πινάκων, ορισμών και άλλων δεδομένων. Η απλή αρχιτεκτονική αλλά και το μικρό της μέγεθος έκαναν την SQLite κατάλληλη για τις κινητές συσκευές περιορισμένων πόρων.

2.2.5 Συνδεσιμότητα

Το λειτουργικό σύστημα Android υποστηρίζει όλες της καινούριες τεχνολογίες επικοινωνιών. Αυτές είναι το Bluetooth, Wi-Fi, UMTS, CDMA, EDGE, και 3G.

2.3 Επίπεδα Αρχιτεκτονικής

Η αρχιτεκτονική της πλατφόρμας Android αποτελείται από 4 επίπεδα και 5 ομάδες, τα οποία απεικονίζονται στην παρακάτω εικόνα:



Σχήμα 4: Επίπεδα Αρχιτεκτονικής λειτουργικού συστήματος Android

Κάθε επίπεδο στην αρχιτεκτονική αυτή, χρησιμοποιεί τις υπηρεσίες που του προσφέρονται από τα πιο κάτω επίπεδα. Ξεκινώντας από το χαμηλότερο επίπεδο θα αναλύσουμε κάθε ένα από αυτά.

2.3.1 Πυρήνας του Linux

Όπως φαίνεται και από το διάγραμμα, η αρχιτεκτονική του Android βασίζεται στον πυρήνα Linux λειτουργικού συστήματος και συγκεκριμένα στην έκδοση 2.6. Ο πυρήνας αυτός καθιστά το Android ένα εύρωστο λειτουργικό σύστημα. Οι βασικές υπηρεσίες που περιλαμβάνονται στο επίπεδο αυτό είναι η διαχείριση μνήμης, ένα μοντέλο ασφάλειας, η διαχείριση διεργασιών, η στοίβα δικτύου και οι οδηγοί συσκευών (π.χ. οθόνης, κάμερας) και αποτελεί ένα αφαιρετικό επίπεδο μεταξύ της στοίβας υλικού και λογισμικού.

2.3.2 Επίπεδο εκτέλεσης (Android Runtime)

Επάνω από το επίπεδο του πυρήνα Linux, στο δεύτερο επίπεδο από το τέλος, βρίσκονται δύο ομάδες επιπέδων. Οι ενσωματωμένες βιβλιοθήκες στο αριστερό μέρος και το επίπεδο εκτέλεσης του Android στο δεξί μέρος. Το επίπεδο αυτό αποτελείται από την εικονική μηχανή Dalvik, η οποία είναι βασισμένη σε καταχωρητές (registers) και από μερικές βιβλιοθήκες πυρήνα (κληρονομούνται σχεδόν όλα τα χαρακτηριστικά που παρέχονται από τις βιβλιοθήκες πυρήνα της γλώσσας προγραμματισμού Java). Η εικονική μηχανή Dalvik (Dalvik VM) σχεδιάστηκε ειδικά για τις ανάγκες της πλατφόρμας Android και βελτιστοποιήθηκε για τις κινητές συσκευές, για τις οποίες υπάρχει περιορισμός ως προς τους πόρους που μπορούν να χρησιμοποιούν (όπως χαμηλή χωρητικότητα σε μνήμη, μικρό μέγεθος μνήμης, χαμηλή ισχύς επεξεργασίας δεδομένων). Η μηχανή Dalvik έχει τη δυνατότητα να εκτελεί προγράμματα σε γλώσσα προγραμματισμού Java. Πρέπει να αναφέρουμε ότι δεν καταλαβαίνει τον κώδικα Java απευθείας αλλά με τη χρησιμοποίηση ενός εργαλείου dx μετατρέπεται ο κώδικας java σε bytecode (αρχεία .dex), ο οποίος είναι κατανοητός από τη μηχανή Dalvik. Ο λόγος της μετατροπής αυτής του κώδικα java σε bytecode είναι η βελτιστοποίηση του κώδικα, ώστε να μπορεί να μεταγλωττιστεί ευκολότερα στην κινητή συσκευή. Το λειτουργικό σύστημα Android υποστηρίζει την εκτέλεση πολλαπλών στιγμιότυπων της μηχανής Dalvik ταυτόχρονα.

2.3.3 Βιβλιοθήκες

Στο επίπεδο πάνω από τον πυρήνα Linux, εκτός από το επίπεδο εκτέλεσης που αναλύσαμε παραπάνω, βρίσκεται και το επίπεδο των εγγενών βιβλιοθηκών. Το επίπεδο αυτό αποτελείται από ένα σύνολο βιβλιοθηκών γραμμένων σε C ή C++, οι οποίες χρησιμοποιούνται από διάφορα μέρη του λειτουργικού συστήματος. Ο διαχειριστής *Surface Manager* παρέχει τη δυνατότητα δημιουργίας διάφορων επιφανειακών σχεδιασμών στην οθόνη της κινητής συσκευής, καθώς επίσης και την σύνθεση γραφικών επιπέδων δύο και τριών διαστάσεων. Η *SGL* είναι ο πυρήνας για τις βιβλιοθήκες γραφικών δύο διαστάσεων. Οι βιβλιοθήκες SSL βρίσκονται στην κατηγορία των εγγενών βιβλιοθηκών προκειμένου να διασφαλιστεί η προστασία και η ακεραιότητα κατά τη μεταφορά δεδομένων μεταξύ ενός web server (εξυπηρετητή ιστού) και του προγράμματος περιήγησης της κινητής συσκευής. Οι *SQLite* βιβλιοθήκες αντιπροσωπεύουν το σύστημα διαχείρισης βάσεων δεδομένων για την κάθε εφαρμογή στο λειτουργικό σύστημα. Ο *Webkit* είναι ένας ενσωματωμένος περιηγητής ιστού ανοιχτού κώδικα, όπως είναι το Safari για την Apple, με μικρές διαφοροποιήσεις. Επιπλέον, το Android υποστηρίζει υψηλής ποιότητας γραφικά 2D και 3D με την βοήθεια της βιβλιοθήκης *OpenGL* και συγκεκριμένα τη διεπαφή προγραμματισμού εφαρμογών OpenGL ES 1.0.

2.3.4 Framework Εφαρμογών

Το επίπεδο πλαισίου εφαρμογών βρίσκεται στο τρίτο επίπεδο ξεκινώντας από το χαμηλότερο και είναι κατά βάση μια ενσωματωμένη εργαλειοθήκη, η οποία παρέχει διαφορετικές υπηρεσίες στις εφαρμογές Android. Όλες αυτές οι υπηρεσίες, οι οποίες χρησιμοποιούνται από βασικές εφαρμογές, είναι διαθέσιμες στους προγραμματιστές εφαρμογών Android, με σκοπό να τους επιτρέπουν την ανάπτυξη καινοτόμων και σύνθετων εφαρμογών. Η αρχιτεκτονική των εφαρμογών είναι σχεδιασμένη ώστε να απλοποιεί την επαναχρησιμοποίηση των διάφορων στοιχείων. Έτσι, κάθε εφαρμογή μπορεί να δημοσιεύει τις δυνατότητες που προσφέρει και στη συνέχεια κάθε άλλη εφαρμογή θα μπορεί να κάνει χρήση αυτών των δυνατοτήτων μέσω των απαραίτητων στοιχείων του επιπέδου πλαισίου εφαρμογών. Τα σημαντικότερα στοιχεία του επιπέδου αυτού θα τα αναλύσουμε στη συνέχεια. Ο Διαχειριστής Δραστηριότητας (*Activity Manager*) ελέγχει τον κύκλο ζωής των εφαρμογών ενώ παρέχει και μια κοινή πλοήγηση για τις εφαρμογές οι οποίες εκτελούνται σε διαφορετικές διεργασίες. Ο Διαχειριστής πακέτου (*Package Manager*) παρακολουθεί όλες τις εφαρμογές οι οποίες βρίσκονται εγκατεστημένες στην συσκευή. Ο Διαχειριστής παραθύρων (*Windows Manager*) ελέγχει την οθόνη της κινητής συσκευής και δημιουργεί επιφάνειες (ένα κατανεμημένο τμήμα μνήμης) για όλες τις εφαρμογές οι οποίες είναι ενεργές. Ο διαχειριστής τηλεφωνίας (*Telephony Manager*) υποστηρίζει εφαρμογές οι οποίες απαιτούν την πρόσβαση σε πληροφορίες ανεξάρτητα από τις υπηρεσίες τηλεφωνίας της συσκευής. Η πρόσβαση σε πληροφορίες τηλεφωνίας είναι προστατευμένη και μερικές εφαρμογές μπορεί να μην έχουν την κατάλληλη άδεια για να προσπελάσουν αυτές τις πληροφορίες (οι περιορισμοί στα δικαιώματα δηλώνονται στο αρχείο Manifest). Οι πάροχοι περιεχομένου (*Content Providers*) υποστηρίζουν τον διαμοιρασμό και την πρόσβαση σε δεδομένα μεταξύ των εφαρμογών. Στο λειτουργικό σύστημα Android ο όρος *πόροι* αναφέρεται σε μη κωδικοποιημένα εξωτερικά στοιχεία. Ο διαχειριστής πόρων (*Resource Manager*) παρέχει πρόσβαση σε τέτοιους εξωτερικούς πόρους σε Android εφαρμογές κατά το διάστημα της ανάπτυξής τους. Η πλατφόρμα Android υποστηρίζει ένα σύνολο από διαφορετικά είδη πόρων συμπεριλαμβανομένων των αρχείων XML (χρησιμοποιούνται για την αποθήκευση δεδομένων, εκτός από αρχεία bitmap και ακατέργαστων δεδομένων), Bitmap (χρησιμοποιούνται για την αποθήκευση εικόνων) και ακατέργαστα αρχεία, όπως αρχεία ήχου, χαρακτήρων και άλλων. Το σύστημα προβολής (*View System*) παρέχει πρόσβαση σε ένα επεκτάσιμο σύνολο όψεων, όπως είναι λίστες, πλαίσια κειμένου, ενσωματωμένοι περιηγητές και κουμπιά. Ο διαχειριστής ειδοποιήσεων (*Notification Manager*) δίνει τη δυνατότητα στις εφαρμογές να ειδοποιούν τους χρήστες της για διάφορα γεγονότα που μπορεί να συμβούν. Η ειδοποίηση μπορεί να γίνει μέσω της πληροφορίας δόνησης της συσκευής, την κατάσταση κατά την οποία αναβοσβήνουν τα LED της κινητής συσκευής, αναπαράγοντας κάποιον ήχο που αντιστοιχεί σε ένα συγκεκριμένο γεγονός ή μέσω ενός εικονιδίου το οποίο θα εμφανίζεται επίμονα στην γραμμή κατάστασης.

Κεφάλαιο 3

Βασικές Αρχές Εφαρμογών

3.1 Βασικά συστατικά εφαρμογών

Η αρχιτεκτονική του λειτουργικού συστήματος Android είναι αρκετά ευέλικτη ώστε να επιτρέπει σε μια εφαρμογή να κάνει χρήση χαρακτηριστικών τα οποία έχουν ήδη αναπτυχθεί από άλλες εφαρμογές. Για παράδειγμα, εάν ένας προγραμματιστής θέλει να αναπτύξει μια εφαρμογή η οποία δημιουργεί και διαβάζει ένα αρχείο pdf και ταυτόχρονα υπάρχει ήδη μια άλλη εφαρμογή η οποία έχει την ικανότητα διαβάσματος pdf αρχείων, τότε ο προγραμματιστής μπορεί να χρησιμοποιήσει το έτοιμο κομμάτι κώδικα που αφορά την ανάγνωση των αρχείων αποφεύγοντας με αυτόν τον τρόπο επιπλέον κόπο και χρόνο. Αυτή η δυνατότητα επαναχρησιμοποίησης εξαλείφει την άσκοπη και περιττή ανάπτυξη νέων εφαρμογών. Ο προγραμματιστής δεν χρειάζεται να ενσωματώσει τον κώδικα άλλων εφαρμογών στον δικό του, απλώς χρησιμοποιεί ένα συγκεκριμένο κομμάτι κώδικα, όποτε αυτό είναι αναγκαίο, στη δική του εφαρμογή. Η αρχιτεκτονική των εφαρμογών Android αποτελείται από πέντε βασικά συστατικά τα οποία χρησιμοποιούνται για να δημιουργηθεί μια εφαρμογή. Στην πραγματικότητα, τα συστατικά αυτά μέρη είναι κάποια αντικείμενα, τα οποία έχουν οριστεί στην εργαλειοθήκη SDK και παρέχουν διαφορετικές μεθόδους μέσω των οποίων μια εφαρμογή μπορεί να αναπτυχθεί. Οι προγραμματιστές χρειάζεται μόνο να καλέσουν και να επεκτείνουν αυτές τις ήδη ορισμένες κλάσεις προκειμένου να τις χρησιμοποιήσουν στη δική τους εφαρμογή.

3.1.1 Δραστηριότητες (Activities)

Μια δραστηριότητα είναι μια μεμονωμένη οθόνη διεπαφής χρήστη σε μια εφαρμογή Android, στην οποία μπορούν να τοποθετηθούν οπτικά στοιχεία, ονομαζόμενα Views και έτσι ο χρήστης μπορεί να επιτελέσει διάφορες ενέργειες αλληλεπιδρώντας με αυτή. Μια εφαρμογή συνήθως αποτελείται από πολλαπλές δραστηριότητες οι οποίες είναι συνδεδεμένες μεταξύ τους. Τυπικά, μια δραστηριότητα σε μια εφαρμογή καθορίζεται από την κεντρική δραστηριότητα, η οποία παρουσιάζεται στο χρήστη όταν γίνεται εκκίνηση της εφαρμογής για πρώτη φορά. Κάθε δραστηριότητα μπορεί να ξεκινήσει μια καινούρια δραστηριότητα προκειμένου για γίνουν διαφορετικές λειτουργίες. Μια δραστηριότητα θα

μπορούσε να θεωρηθεί μια λίστα αντικειμένων, όπως για παράδειγμα ένα σύνολο από ετικέτες σε ένα μενού ή μια σειρά εικονιδίων σε ένα πλαίσιο διαλόγου μιας εφαρμογής. Μια εφαρμογή μπορεί να αποτελείται από μία ή και περισσότερες δραστηριότητες ανάλογα με την αρχιτεκτονική και τον τρόπο σχεδίασής της. Για παράδειγμα, μια εφαρμογή ανταλλαγής γραπτών μηνυμάτων αποτελείται από τη δραστηριότητα σύνταξης νέου μηνύματος, τη δραστηριότητα ανάγνωσης εισερχόμενων μηνυμάτων, τη δραστηριότητα παροχής πληροφοριών για τα σταλμένα μηνύματα και τη δραστηριότητα που δίνει τη δυνατότητα στον χρήστη να επιλέξει την προτεραιότητα των μηνυμάτων. Παρόλο που όλες αυτές οι δραστηριότητες δουλεύουν μαζί και εμπεριέχονται σε μία εφαρμογή, είναι ανεξάρτητες η μία από την άλλη και έχουν δημιουργηθεί σε ξεχωριστές υποκλάσεις της κλάσης για την κύρια δραστηριότητα. Συνήθως, μια δραστηριότητα προβάλλεται σε ένα παράθυρο το οποίο καλύπτει ολόκληρη την οθόνη της κινητής συσκευής, αλλά μερικές από αυτές μπορεί να εκτελούνται σε μικρότερα παράθυρα. Έτσι, μια εφαρμογή που περιέχει πολλές δραστηριότητες μπορεί να εκτελείται σε πολλαπλά παράθυρα. Το οπτικό περιεχόμενο ενός παραθύρου αποτελείται από μια ιεραρχία όψεων-αντικειμένων, τα οποία προέρχονται από την κύρια κλάση όψης. Κάθε όψη (View) ελέγχει ένα τετραγωνικό χώρο μέσα σε ένα παράθυρο. Οι μητρικές όψεις (parent views) περιέχουν και οργανώνουν τη διάταξη των θυγατρικών όψεων (children views). Οι προβολές που βρίσκονται στο κατώτερο επίπεδο ιεραρχίας παρουσιάζονται στο τετραγωνικό πλαίσιο, ελέγχουν και αποκρίνονται στις ενέργειες του χρήστη που λαμβάνουν χώρα στο συγκεκριμένο πλαίσιο. Έτσι, οι προβολές είναι το σημείο στο οποίο γίνεται η αλληλεπίδραση με τον χρήστη της εφαρμογής. Το Android λειτουργικό σύστημα υποστηρίζει ένα σύνολο από έτοιμες όψεις (Views), οι οποίες μπορούν να χρησιμοποιηθούν όπως κουμπιά, πλαίσια κειμένου, γραμμές κύλισης, checkboxes, στοιχεία μενού και άλλα.

3.1.2 Υπηρεσίες (Services)

Η υπηρεσία αναφέρεται στα στοιχεία μιας εφαρμογής τα οποία δεν είναι ορατά. Μια υπηρεσία εκτελείται στο παρασκήνιο μιας δραστηριότητας ή μιας εφαρμογής. Η αλληλεπίδραση ενός χρήστη με μια υπηρεσία είναι δυνατή μέσω μιας διεπαφής, η οποία παρουσιάζεται από τη δραστηριότητα και κατέχει η συγκεκριμένη υπηρεσία. Η δραστηριότητα παρέχει και άλλες ρυθμίσεις που αφορούν μια υπηρεσία. Εάν, για παράδειγμα, ένας χρήστης ενεργοποιήσει την υπηρεσία της αναπαραγωγής ενός μουσικού κομματιού μέσω της δραστηριότητας του media player του δίνεται η δυνατότητα να σταματήσει την αναπαραγωγή, να προχωρήσει το κομμάτι προς τα εμπρός ή προς τα πίσω (reverse-forward). Παρόλα αυτά, η

δραστηριότητα δεν έχει τον συνεχή έλεγχο της υπηρεσίας, ενώ ο χρήστης αναμένει την συνέχιση της υπηρεσίας και αφού ξεκινήσει μια άλλη ενέργεια. Όταν μια δραστηριότητα ξεκινά την αναπαραγωγή ενός μουσικού κομματιού στο παρασκήνιο, τότε υπεύθυνο για την συνέχιση της αναπαραγωγής είναι το σύστημα. Κάθε υπηρεσία επεκτείνει τη βασική κλάση Υπηρεσίας. Όπως οι δραστηριότητες και άλλα στοιχεία, έτσι και οι υπηρεσίες εκτελούνται στο βασικό νήμα της διεργασίας της εφαρμογής. Έτσι, προκειμένου να μην μπλοκάρουν άλλα στοιχεία που εκτελούνται ή τη διεπαφή του χρήστη, οι υπηρεσίες συχνά δημιουργούν νέα νήματα για την εξυπηρέτηση χρονοβόρων εργασιών.

3.1.3 Δέκτες Μετάδοσης (Broadcast Receivers)

Ο δέκτης μετάδοσης είναι ένα ακόμα βασικό στοιχείο των εφαρμογών Android. Χρησιμοποιείται για να λαμβάνει τις μεταδιδόμενες ανακοινώσεις αλλά και για να αντιδρά σύμφωνα με τις προκύπτουσες καταστάσεις. Κανονικά το σύστημα μεταδίδει κάποιες ανακοινώσεις, όπως για παράδειγμα την αλλαγή της ζώνης στην ώρα, τη λήψη μιας φωτογραφίας, την ειδοποίηση για χαμηλή μπαταρία ή την αλλαγή γλώσσας. Μια εφαρμογή μπορεί να ξεκινήσει μια μετάδοση και ένας αποδέκτης μετάδοσης, ο οποίος εμπεριέχεται στις εφαρμογές, μπορεί να αντιδράσει σε αυτή εάν χρειαστεί. Για παράδειγμα, ένας περιηγητής (browser) μπορεί να παράγει μια ανακοίνωση ότι η λήψη ενός αρχείου ολοκληρώθηκε και είναι έτοιμο να χρησιμοποιηθεί από άλλες εφαρμογές. Μια εφαρμογή μπορεί να διαθέτει και παραπάνω από έναν δέκτη μεταδόσεων, έτσι ώστε να λαμβάνει πολλαπλές ανακοινώσεις ταυτόχρονα. Όλοι οι δέκτες μετάδοσης επεκτείνουν την βασική κλάση *BroadcastReceiver*. Σε αντίθεση με τα στοιχεία υπηρεσίας, οι δέκτες μετάδοσης δεν παρέχουν κανένα είδος διεπάφης, αλλά μπορεί να ξεκινήσει μια καινούρια δραστηριότητα ως απάντηση στο μήνυμα το οποίο έλαβε ή μπορεί να ειδοποιήσει τον χρήστη μέσω του διαχειριστή ανακοινώσεων (*Notification Manager*). Ο διαχειριστής αυτός επιτρέπει στην εφαρμογή να ειδοποιεί τους χρήστες της σχετικά με γεγονότα τα οποία συμβαίνουν.

3.1.4 Πάροχοι Περιεχομένου (Content Providers)

Οι πάροχοι περιεχομένου διαχειρίζονται τον διαμοιρασμό και την πρόσβαση σε δεδομένα μεταξύ των εφαρμογών. Στο λειτουργικό σύστημα Android δεν υπάρχει κανένα είδος κεντρικής αποθήκευσης βάσεων δεδομένων τα οποία να είναι

διαθέσιμα για όλα τα Android πακέτα. Οι πάροχοι περιεχομένου είναι ο μόνος τρόπος για την αποθήκευση, την ανάκτηση και τον διαμοιρασμό πληροφοριών μεταξύ των εφαρμογών. Ο κάθε πάροχος περιεχομένου αποθηκεύει ένα συγκεκριμένο σύνολο δεδομένων, ενώ υπάρχουν διαφορετικοί πάροχοι περιεχομένου για κάθε διαφορετικό τύπο δεδομένων όπως αρχεία εικόνων, ήχου, video. Δεν απαιτείται η ανάπτυξη ενός παρόχου εάν δεν υπάρχει η πρόθεση να διαμοιραστούν τα δεδομένα μιας εφαρμογής με άλλες.

3.1.5 Intents

Κατά βάση, τα Intents δεν αποτελούν ένα βασικό συστατικό των εφαρμογών Android, αλλά είναι ένας μηχανισμός ενεργοποίησης συστατικών στο σύστημα Android. Αποτελεί το κεντρικό σύστημα μηνυμάτων και ορίζει ένα μήνυμα για να ενεργοποιήσει ένα συγκεκριμένο συστατικό. Εάν, για παράδειγμα, από την τρέχουσα δραστηριότητά μας θέλουμε να αναφερθούμε σε μια νέα δραστηριότητα απαιτείται να πυροδοτηθεί ένα *Intent (πρόθεση)*, το οποίο να προσδιορίζει τη νέα δραστηριότητα. Επίσης, εάν μέσα από μια δραστηριότητα θέλουμε να ξεκινήσει μια νέα εφαρμογή, τότε θα πρέπει και πάλι να πυροδοτηθεί ένα Intent. Έτσι, πυροδοτώντας ένα Intent γνωστοποιούμε στο σύστημα Android ότι θα πρέπει να κάνει κάποια ενέργεια. Υπάρχουν δύο τύποι Intents στο Android σύστημα: τα σαφή intents (explicit intents) και τα υπονοούμενα (implicit intents). Στην πρώτη κατηγορία εφαρμόζεται μεγάλη ακρίβεια. Προσδιορίζουμε ποια δραστηριότητα θα ενεργοποιηθεί κατά τη λήψη του intent. Η κατηγορία αυτή χρησιμοποιείται συνήθως για την εσωτερική επικοινωνία μεταξύ των εφαρμογών. Όσον αφορά στη δεύτερη κατηγορία Intents, στέλνουμε ένα μήνυμα στο σύστημα Android προκειμένου να βρει την κατάλληλη δραστηριότητα η οποία να μπορεί να ανταποκριθεί σε ένα συγκεκριμένο intent.

3.2 Ενεργοποίηση συστατικών εφαρμογής

Τρία από τα βασικά συστατικά μιας εφαρμογής Android, δραστηριότητες, υπηρεσίες και δέκτες μετάδοσης, ενεργοποιούνται μέσω μηνυμάτων τα οποία ονομάζονται intents. Το αντικείμενο Intent είναι μια παθητική δομή δεδομένων η οποία έχει καταχωρισμένη μια αφηρημένη περιγραφή μιας λειτουργίας που πρέπει να γίνει ή συχνά, στην περίπτωση των δεκτών μετάδοσης, την περιγραφή ενός

γεγονότος, το οποίο έχει γίνει και πρόκειται να ανακοινωθεί. Ο πάροχος περιεχομένου δεν ενεργοποιείται από intents. Όπως προαναφέραμε ένας πάροχος περιεχομένου παρουσιάζει δεδομένα σε εξωτερικές εφαρμογές με τη βοήθεια ενός ή και περισσότερων πινάκων παρόμοιων με αυτούς που συναντάμε σε μια βάση δεδομένων. Μια εφαρμογή αποκτά πρόσβαση στα δεδομένα ενός παρόχου περιεχομένου μέσω του αντικειμένου *Content Resolver* που λειτουργεί σαν πελάτης. Το αντικείμενο αυτό διαθέτει μεθόδους οι οποίες καλούν μεθόδους του αντικειμένου του παρόχου έχοντας ταυτόσημα ονόματα. Οι μέθοδοι του αντικειμένου Content Resolver παρέχουν τις βασικές CRUD (Create, Read, Update, Delete) λειτουργίες αποθήκευσης (δημιουργία, ανάκτηση, ανανέωση, διαγραφή). Για να αποκτήσει πρόσβαση μια εφαρμογή σε έναν Content Provider θα πρέπει να ζητήσει συγκεκριμένες άδειες. Υπάρχουν ξεχωριστές μέθοδοι για την ενεργοποίηση κάθε τύπου συστατικού.

3.2.1 Ενεργοποίηση Δραστηριότητας (Activating Activity)

Όπως αναφέρθηκε μια δραστηριότητα είναι το συστατικό μιας εφαρμογής το οποίο παρέχει μια οθόνη μέσω της οποίας οι χρήστες μπορούν να αλληλεπιδρούν για να κάνουν κάποιες ενέργειες. Για να ξεκινήσει μια δραστηριότητα ή για να της αναθέσουμε να κάνει μια καινούρια λειτουργία, θα πρέπει να περάσει σαν όρισμα ένα αντικείμενο Intent είτε στη μέθοδο *Context.startActivity()* είτε στη μέθοδο *Activity.startActivityForResult()* (όταν θέλουμε να επιστραφεί ένα αποτέλεσμα στην δραστηριότητα που κάλεσε την μέθοδο). Για παράδειγμα, μια δραστηριότητα συγγραφής μηνυμάτων μπορεί να ξεκινήσει μια δραστηριότητα τηλεφωνικού καταλόγου από την οποία ο χρήστης μπορεί να επιλέξει το όνομα της επαφής , και έτσι η αρχική δραστηριότητα συγγραφής μηνυμάτων αναμένει από την δραστηριότητα τηλεφωνικού καταλόγου να επιστρέψει τις επιλεγμένες επαφές. Το αντικείμενο intent περιγράφει τη δραστηριότητα την οποία επιθυμούμε να ξεκινήσει. Συγκεκριμένα είτε προσδιορίζει την ακριβή δραστηριότητα είτε περιγράφει τον τύπο της ενέργειας που θέλουμε να εκτελεστεί. Επιπλέον, ένα αντικείμενο intent μπορεί να μεταφέρει μικρή ποσότητα δεδομένων τα οποία θα χρησιμοποιηθούν από τη δραστηριότητα που πρόκειται να ξεκινήσει.

3.2.2 Ενεργοποίηση υπηρεσίας (Activating Service)

Μια υπηρεσία είναι ένα συστατικό των εφαρμογών Android το οποίο εκτελείται στο παρασκήνιο και δεν υπάρχει άμεση αλληλεπίδραση με τον χρήστη. Η πλατφόρμα

Android παρέχει και εκτελεί προκαθορισμένες υπηρεσίες συστήματος και κάθε εφαρμογή Android μπορεί να τις χρησιμοποιήσει, αφού έχουν δοθεί οι κατάλληλες άδειες. Μια καινούρια υπηρεσία ξεκινάει (ή δίνονται επιπλέον οδηγίες σε μια ήδη εκτελούμενη υπηρεσία) περνώντας σαν όρισμα ένα αντικείμενο intent στη μέθοδο *Context.startService ()*. Όταν ένας χρήστης καλεί μια υπηρεσία, το σύστημα Android καλεί την μέθοδο *onStart ()* περνώντας το αντικείμενο intent. Η μέθοδος *Context.bindService ()* χρησιμοποιείται για να εδραιωθεί μια σύνδεση μεταξύ του καλούμενου συστατικού και της ήδη εκτελούμενης υπηρεσίας.

3.2.3 Ενεργοποίηση δέκτη μετάδοσης (Activating Broadcast Receiver)

Ο δέκτης μετάδοσης είναι ένα συστατικό των Android εφαρμογών το οποίο επιτρέπει την καταγραφή γεγονότων που αφορούν είτε εφαρμογές είτε το σύστημα Android. Όλοι οι καταγεγραμμένοι αποδέκτες ενός γεγονότος θα ειδοποιηθούν από το Android λειτουργικό σύστημα όταν συμβεί το γεγονός αυτό. Για την ενεργοποίηση ενός δέκτη μετάδοσης υπάρχουν τρεις διαφορετικές μέθοδοι, οι *sendBroadcast ()*, *sendOrderedBroadcast ()* και *sendStickyBroadcast ()*. Η επιλογή της κατάλληλης μεθόδου εξαρτάται από τον τύπο της μετάδοσης. Το αντικείμενο intent περνάει σαν όρισμα στην μέθοδο *sendBroadcast ()*, όταν θέλουμε να περιορίσουμε την απάντηση του αποδέκτη, ώστε να μην επιτρέπεται η διάδοση κανενός αποτελέσματος καθώς και η ματαίωση της μετάδοσης. Σε περίπτωση που είναι επιθυμητή η διάδοση οποιουδήποτε αποτελέσματος και επιτρέπεται στον δέκτη η ματαίωση της μετάδοσης, το αντικείμενο intent περνάει σαν όρισμα στην μέθοδο *sendOrderedBroadcast ()*. Το σύστημα Android μεταφέρει το αντικείμενο intent σε όλους τους ενδιαφερομένους δέκτες μετάδοσης με την κλήση των μεθόδων *onReceive()*.

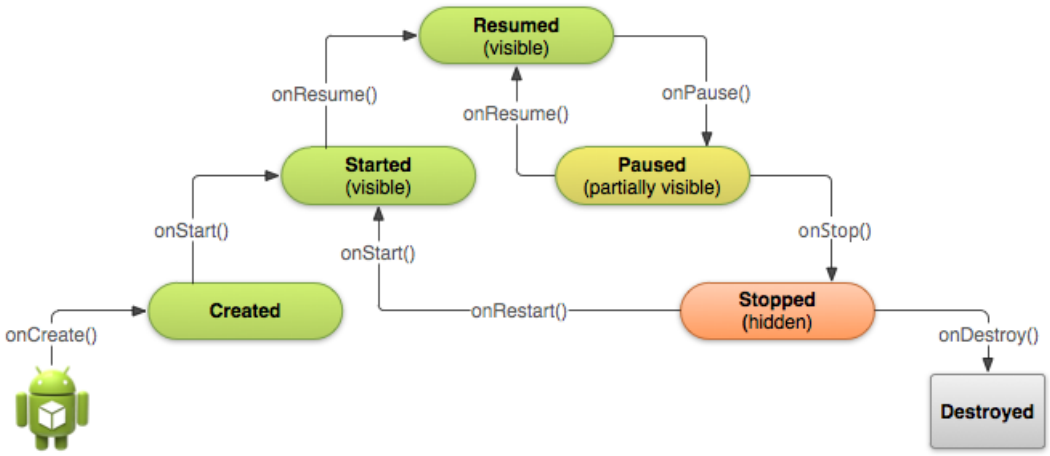
Κεφάλαιο 4

Βασικά μέρη: Κύκλος Ζωής

Ο όρος κύκλος ζωής αναφέρεται στη διάρκεια ενός συστατικού μιας Android εφαρμογής από το ξεκίνημά της, όταν αποκτά υπόσταση, μέχρι την καταστροφή ή το τέλος της. Κατά τη διάρκεια του διαστήματος από την αρχικοποίηση έως το τέλος του, ένα συστατικό μπορεί να είναι ενεργό, ανενεργό, ορατό ή αόρατο. Στη συνέχεια θα αναλύσουμε τον κύκλο ζωής της δραστηριότητας.

4.1 Κύκλος ζωής δραστηριότητας (Activity)

Κατά τη διάρκεια ζωής μιας δραστηριότητας, το σύστημα καλεί ένα σύνολο μεθόδων με μια αλληλουχία παρόμοια με αυτή των βημάτων μιας πυραμίδας. Αυτό συμβαίνει γιατί κάθε επίπεδο του κύκλου ζωής μιας δραστηριότητας αποτελεί ένα ξεχωριστό βήμα στην πυραμίδα. Όταν το σύστημα δημιουργεί μια νέα δραστηριότητα, κάθε μέθοδος που καλείται μετακινεί την κατάσταση της δραστηριότητας προς το επίπεδο κορυφής της πυραμίδας. Η κορυφή της πυραμίδας καταστάσεων αντιπροσωπεύει την κατάσταση κατά την οποία η δραστηριότητα εκτελείται και ο χρήστης μπορεί να αλληλεπιδράσει με αυτή.



Σχήμα 5: Πυραμίδα καταστάσεων κύκλου ζωής δραστηριότητας

Επιπλέον, καθώς ο χρήστης «εγκαταλείπει» μια δραστηριότητα, το σύστημα καλεί κάποιες άλλες μεθόδους οι οποίες μεταφέρουν την κατάστασή της προς το χαμηλότερο επίπεδο καταστάσεων προκειμένου να σταματήσει σταδιακά η

λειτουργία της. Όπως φαίνεται και από το *σχήμα 5*, υπάρχουν αρκετές περιπτώσεις στις οποίες μια δραστηριότητα μπορεί να μεταβεί από την μία κατάσταση στην άλλη. Τρεις μόνο από αυτές τις καταστάσεις μπορεί να είναι στατικές, γεγονός που σημαίνει ότι μια δραστηριότητα μπορεί να παραμείνει σε αυτές για ένα μεγάλο χρονικό διάστημα. Οι καταστάσεις αυτές είναι η ενεργή, η paused και η stopped οι οποίες αναλύονται στη συνέχεια.

4.1.1 Ενεργή (Active ή Running)

Οι δραστηριότητες διαχειρίζονται από το σύστημα Android σαν μια στοίβα δραστηριοτήτων. Όταν μια καινούρια δραστηριότητα ξεκινά, τοποθετείται στην κορυφή της στοίβας αυτής και γίνεται η εκτελούμενη δραστηριότητα και ο χρήστης μπορεί να αλληλεπιδράσει απευθείας με αυτή. Η προηγούμενη δραστηριότητα παραμένει χαμηλότερα στην στοίβα από την νέα και δεν θα μπορέσει να εκτελεστεί για όσο διάστημα υπάρχει η από επάνω της δραστηριότητα.

4.1.2 Paused

Μια δραστηριότητα βρίσκεται σε παύση όταν μια άλλη δραστηριότητα έρχεται στην κορυφή της στοίβας. Η δραστηριότητα αυτή είτε είναι διάφανη είτε δεν καλύπτει ολόκληρη την οθόνη και έτσι η δραστηριότητα σε παύση μπορεί να είναι ορατή ολόκληρη ή ένα μέρος της. Μια δραστηριότητα σε παύση διατηρεί όλες τις καταστάσεις της και παραμένει συνδεδεμένη με τον διαχειριστή παραθύρου (*Window Manager*), ενώ μπορεί να τερματιστεί σε καταστάσεις πολύ χαμηλών επιπέδων μνήμης.

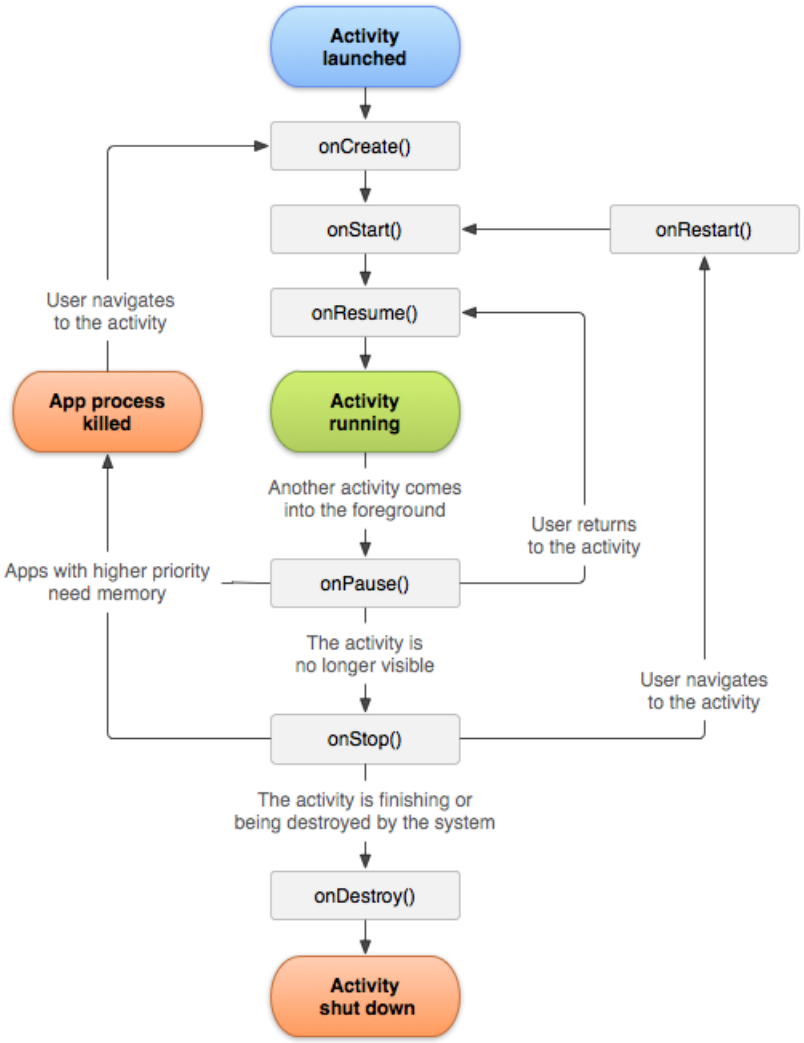
4.1.3 Stopped

Όταν μια δραστηριότητα είναι σταματημένη τότε δεν είναι ορατή στους χρήστες (επισκιάζεται από κάποια άλλη δραστηριότητα) και ενώ δεν μπορούν να αλληλεπιδράσουν απευθείας μαζί της , αυτή διατηρεί όλες τις καταστάσεις της. Εάν όμως η χρήση μνήμης είναι απαραίτητη για κάποια άλλη λειτουργία, τότε η σταματημένη δραστηριότητα καταστρέφεται. Το σύστημα μπορεί να αφαιρέσει μια δραστηριότητα που βρίσκεται στην κατάσταση αυτή είτε καλώντας την μέθοδο *finish()* είτε καταστρέφοντας τη διεργασία της.

4.1.4 Αποθήκευση της κατάστασης δραστηριότητας

Υπάρχει το ενδεχόμενο το σύστημα να τερματίσει μια δραστηριότητα προκειμένου να απελευθερωθεί χώρος μνήμης και ο χρήστης να ξεκινήσει ξανά τη δραστηριότητα αυτή και μπορεί να τη βρει στην κατάσταση στην οποία την άφησε. Πριν τον τερματισμό μιας διαδικασίας δίνεται η δυνατότητα αποθήκευσης της κατάστασής της με τη βοήθεια της μεθόδου *onSaveInstanceState()*. Οι μέθοδοι *onSaveInstanceState()* και *onRestoreInstanceState()* δεν αποτελούν μέρος του κύκλου ζωής μιας δραστηριότητας γιατί δεν καλούνται σε κάθε περίπτωση. Οι μέθοδοι αυτές καλούνται μόνο όταν το σύστημα τερματίζει μια δραστηριότητα. Από την άλλη πλευρά, όταν ο χρήστης επιθυμεί τον τερματισμό μιας δραστηριότητας μέσω κάποιων ενεργειών όπως ακύρωση ή κλείσιμο και δεν αναμένεται ο χρήστης να επιστρέψει σε αυτή, τότε δεν υπάρχει η ανάγκη αποθήκευσης της κατάστασης της δραστηριότητας.

Οι υπόλοιπες καταστάσεις που βλέπουμε στην πυραμίδα του σχήματος 5 (*Created()* και *Started()*) είναι παροδικές και το σύστημα κινείται πολύ γρήγορα από αυτές σε κάποια επόμενη κατάσταση καλώντας την επόμενη μέθοδο του κύκλου ζωής. Αυτό σημαίνει ότι μετά την κλήση της μεθόδου *onCreate()* καλείται αμέσως η μέθοδος *onStart()*, η οποία ακολουθείται από την κλήση της μεθόδου *onResume()*.



Σχήμα 6:Στάδια κύκλου ζωής δραστηριότητας

Παρατηρώντας το σχήμα 6, μπορούμε να ξεχωρίσουμε 3 βασικούς βρόχους που αφορούν στον έλεγχο του πλαισίου δραστηριότητας.

1.Συνολική διάρκεια ζωής

Ο βρόχος αυτός περιλαμβάνει οτιδήποτε συμβαίνει σε μια δραστηριότητα από την κλήση της μεθόδου *onCreate()* έως την κλήση της μεθόδου *onDestroy()*. Κατά τη διάρκεια αυτού του διαστήματος η δραστηριότητα παρουσιάζει την αρχική εγκατάσταση της συνολικής κατάστασης με την μέθοδο *onCreate()* και απελευθερώνει όλους τους πόρους καλώντας τη μέθοδο *onDestroy()*.

2. Η ορατή διάρκεια ζωής

Το χρονικό διάστημα μιας δραστηριότητας από την κλήση της μεθόδου *onStart()* μέχρι την κλήση της αναφερόμενης μεθόδου *onStop()* καλείται ορατός κύκλος ζωής μιας δραστηριότητας. Κατά τη διάρκεια αυτού του διαστήματος μια δραστηριότητα μπορεί να είναι ορατή στην οθόνη παρόλο που μπορεί να μην βρίσκεται στο προσκήνιο και έτσι ο χρήστης να μην μπορεί να αλληλεπιδρά άμεσα με αυτή. Οι μέθοδοι *onStart()* και *onStop()* μπορεί να καλούνται πολλές φορές γεγονός που εξαρτάται από τις ανάγκες των χρηστών αλλά και από τις εναλλαγές μιας δραστηριότητας από μια κατάσταση ορατή ή αόρατη ως προς τον χρήστη.

3. Διάρκεια ζωής στο προσκήνιο

Ο χρόνος μιας δραστηριότητας μεταξύ της κλήσης μεθόδου *onResume()* και της κλήσης μεθόδου *onPause()* αποτελεί τον βρόχο της διάρκειας ζωής στο προσκήνιο για τη δραστηριότητα αυτή. Κατά τη διάρκεια αυτού του διαστήματος, στο οποίο ο χρήστης μπορεί να αλληλεπιδρά άμεσα με τη δραστηριότητα, οι δραστηριότητες μπορούν συχνά να εναλλάσσονται μεταξύ των καταστάσεων resume και paused.

4.2 Διεργασίες και νήματα

Όταν ξεκινάει το συστατικό μιας εφαρμογής και η εφαρμογή δεν διαθέτει κανένα άλλο εκτελούμενο συστατικό, το Android λειτουργικό σύστημα ξεκινά μια καινούρια διεργασία Linux για την εφαρμογή αυτή με ένα νήμα εκτέλεσης. Σαν προεπιλογή, όλα τα συστατικά μιας εφαρμογής εκτελούνται στην ίδια διεργασία και νήμα, το οποίο ονομάζεται και κεντρικό νήμα. Εάν ένα συστατικό μιας εφαρμογής ξεκινήσει ενώ ήδη υπάρχει μια διεργασία για αυτή την εφαρμογή τότε το συστατικό αυτό ξεκινά μέσα στην ίδια διεργασία και χρησιμοποιεί το ίδιο νήμα εκτέλεσης. Ωστόσο μπορεί να γίνει διακανονισμός έτσι ώστε διαφορετικά συστατικά μιας εφαρμογής να τρέχουν σε διαφορετικές διεργασίες, ενώ υπάρχει η δυνατότητα δημιουργίας επιπρόσθετων νημάτων για κάθε διεργασία.

Διεργασίες (Processes)

Το αρχείο *Manifest* ελέγχει τις διεργασίες στις οποίες τα συστατικά μιας εφαρμογής πρόκειται να εκτελεστούν. Τα στοιχεία ενός συστατικού είναι τα *<activity>,*

<service>, <receivers>, <providers>. Όλα αυτά τα στοιχεία περιλαμβάνουν ένα χαρακτηριστικό το οποίο ονομάζεται διεργασία και αποφασίζει για το πού θα εκτελεστεί ένα συστατικό. Τα χαρακτηριστικά αυτά μπορούν να διαμορφωθούν με τέτοιο τρόπο έτσι ώστε κάθε συστατικό να μπορεί να εκτελείται στη δική του διεργασία ή μερικά συστατικά να μοιράζονται μια διεργασία και κάποια άλλα όχι. Επιπλέον, αυτά τα χαρακτηριστικά μπορούν να διαμορφωθούν με τέτοιο τρόπο ώστε τα συστατικά από διαφορετικές εφαρμογές να μπορούν να φιλοξενούνται από μια διεργασία υπό την προϋπόθεση ότι οι εφαρμογές αυτές μοιράζονται το ίδιο Linux ID του χρήστη και έχουν υπογραφεί με τα ίδια πιστοποιητικά. Το λειτουργικό σύστημα Android μπορεί να αποφασίσει το κλείσιμο μιας διεργασίας κάποια στιγμή, όταν τα επίπεδα μνήμης είναι χαμηλά και όταν απαιτείται μνήμη από άλλες διεργασίες οι οποίες εξυπηρετούν πιο άμεσα τον χρήστη. Σαν επακόλουθο, τα δομικά στοιχεία μιας εφαρμογής που σταμάτησε, καταστρέφονται και αυτά. Η διεργασία ξεκινά ξανά για αυτά τα στοιχεία, όταν πρέπει να επιτελέσουν τη λειτουργία τους. Στο σημείο όπου λαμβάνεται η απόφαση για το ποιες διεργασίες θα πρέπει να τερματιστούν, το λειτουργικό σύστημα Android αξιολογεί την σπουδαιότητα της κάθε μιας διεργασίας για τον χρήστη. Ο κύκλος ζωής των διεργασιών παρουσιάζεται στη συνέχεια.

Νήματα (Threads)

Όταν ξεκινά μια εφαρμογή, το σύστημα δημιουργεί ένα νήμα εκτέλεσης το οποίο ονομάζεται κύριο νήμα. Το νήμα αυτό είναι πολύ σημαντικό αφού έχει την ευθύνη για την αποστολή των γεγονότων στην κατάλληλη διεπαφή χρήστη συμπεριλαμβανομένων και των γεγονότων σχεδιασμού. Το σύστημα δεν δημιουργεί ένα ξεχωριστό νήμα για κάθε στιγμιότυπο ενός βασικού στοιχείου μιας εφαρμογής. Όλα τα βασικά στοιχεία που εκτελούνται στην ίδια διεργασία αποκτούν υπόσταση στο νήμα UI και οι κλήσεις συστήματος σε κάθε στοιχείο αποστέλλονται από αυτό το νήμα. Όταν μια εφαρμογή εκτελεί λειτουργίες με έντονους ρυθμούς, από την πλευρά αλληλεπίδρασης του χρήστη, το μοντέλο χρησιμοποίησης ενός μοναδικού νήματος μπορεί να εμφανίσει «φτωχή» παρουσίαση, εκτός και εάν η εφαρμογή εκτελείται με σωστό τρόπο. Πιο συγκεκριμένα, εάν εκτελούνται όλα στο νήμα UI, τότε η εκτέλεση μεγάλων λειτουργιών όπως η πρόσβαση στο διαδίκτυο ή σε μια βάση δεδομένων θα μπλοκάρει το νήμα αυτό. Από την πλευρά των χρηστών η εφαρμογή φαίνεται να βρίσκεται σε μεγάλη αναμονή γεγονός που εάν κρατήσει για μεγάλο χρόνο δεν θα κερδίσει την εμπιστοσύνη τους. Κάθε λειτουργία η οποία διαρκεί για μεγάλο χρονικό διάστημα θα πρέπει να εκτελείται σε ξεχωριστό νήμα. Τα αντικείμενα νημάτων της Java χρησιμοποιούνται για τη δημιουργία νέων νημάτων.

4.3 Κύκλος ζωής και διεργασίες

Το λειτουργικό σύστημα Android προσπαθεί να διατηρήσει τη διεργασία μιας εφαρμογής όσο το δυνατόν περισσότερο, αλλά εν τέλει απαιτείται η απομάκρυνση των παλιών διεργασιών προκειμένου να ανακτηθεί μνήμη για νεότερες και σημαντικότερες διεργασίες. Για να ληφθεί η απόφαση σχετικά με το ποιες διεργασίες θα διατηρηθούν και ποιες θα τερματιστούν το λειτουργικό σύστημα τοποθετεί κάθε διεργασία σε μια ιεραρχία που έχει ως βάση τη σπουδαιότητα των στοιχείων που εκτελούνται σε κάθε διεργασία καθώς επίσης και την κατάσταση στην οποία βρίσκονται τα στοιχεία αυτά. Οι διεργασίες οι οποίες είναι λιγότερο σημαντικές εξαλείφονται πρώτες, στη συνέχεια ακολουθούν αυτές με την αμέσως χαμηλότερη σημασία και με αυτό τον τρόπο το σύστημα ανακτά τους πόρους που χρειάζεται. Η ιεραρχία σπουδαιότητας στην οποία τοποθετούνται οι διεργασίες αποτελείται από πέντε επίπεδα. Τα επίπεδα αυτά παρουσιάζονται αναλυτικά στη συνέχεια ξεκινώντας από το σημαντικότερο σε ιεραρχία επίπεδο.

4.3.1 Διεργασία στο προσκήνιο (Foreground process)

Η διεργασία που βρίσκεται στο προσκήνιο είναι η διεργασία που απαιτείται για τις ενέργειες που κάνει ο χρήστης την τρέχουσα χρονική στιγμή στο σύστημα. Η διεργασία αυτή θεωρείται ότι βρίσκεται στο προσκήνιο όταν είναι αληθής οποιαδήποτε από τις ακόλουθες καταστάσεις.

- Μια διεργασία «τρέχει» μια δραστηριότητα με την οποία ο χρήστης αλληλεπιδρά. Η δραστηριότητα είναι σε ενεργή κατάσταση και έχει κληθεί η μέθοδος *onResume()*.
- Μια διεργασία μπορεί να φιλοξενεί μια υπηρεσία η οποία να είναι συνδεδεμένη με μια δραστηριότητα με την οποία ο χρήστης αλληλεπιδρά. Η διεργασία έχει ένα αντικείμενο υπηρεσίας και στέλνει ή εκτελεί κλήσεις με χρήση των μεθόδων *onCreate(), onStart(), onDestroy()*.
- Η διεργασία περιέχει ένα ενεργό αντικείμενο αποδέκτη λήψης (broadcast receiver) γεγονός που σημαίνει ότι εκτελείται η μέθοδος *onReceive()* του αποδέκτη λήψης.

Σε γενικές γραμμές, πολύ λίγες διεργασίες προσκηνίου μπορεί να υπάρχουν σε μια δεδομένη χρονική στιγμή. Αυτού του τύπου οι διεργασίες τερματίζονται όταν δεν υπάρχει άλλη επιλογή και το σύστημα έχει πολύ χαμηλά επίπεδα μνήμης. Όταν το σύστημα φτάνει σε αυτό το σημείο είναι αναγκαίο από τη

συσκευή να τερματίσει ορισμένες από τις διεργασίες προσκηνίου προκειμένου να διατηρηθεί η διεπαφή χρήστη σε καλό επίπεδο απόκρισης.

4.3.2 Ορατές διεργασίες (visible processes)

Ορατή διεργασία είναι η διεργασία η οποία δεν έχει κάποια βασικά στοιχεία της στο προσκήνιο, αλλά ακόμα μπορεί να επηρεάσει αυτά που βλέπει ο χρήστης στην οθόνη. Μια ορατή διεργασία έχει τις ακόλουθες καταστάσεις.

- Η διεργασία η οποία τρέχει μια δραστηριότητα που δεν είναι ενεργή, αλλά ούτε και βρίσκεται στο προσκήνιο, ωστόσο ο χρήστης μπορεί να τη βλέπει. Βρίσκεται στην κατάσταση παύσης και έχει αρχικοποιηθεί η μέθοδος *onPause().* Κάτι τέτοιο μπορεί να συμβεί εάν μια δραστηριότητα προσκηνίου ξεκινήσει έναν διάλογο, αλλά επιτρέπει στην προηγούμενη δραστηριότητα να είναι ορατή στο πίσω μέρος.
- Η διεργασία μπορεί να φιλοξενεί μια δραστηριότητα η οποία είναι ορατή.

Οι ορατές διεργασίες θεωρούνται πολύ σημαντικές και δεν πρόκειται να τερματιστούν εάν δεν υπάρχει ανάγκη για μνήμη, έτσι ώστε οι διεργασίες προσκηνίου να συνεχίσουν να εκτελούνται.

4.3.3 Διεργασίες υπηρεσίας (Service processes)

Η διεργασία αυτή είναι μια διεργασία η οποία τρέχει μια υπηρεσία που έχει ξεκινήσει με τη μέθοδο *StartService()* και δεν εμπίπτει στις προηγούμενες δύο κατηγορίες. Παρά το γεγονός ότι οι διεργασίες υπηρεσίας δεν είναι άμεσα συνδεδεμένες με οτιδήποτε βλέπει ο χρήστης, σε γενικές γραμμές εκτελούν λειτουργίες για τις οποίες ενδιαφέρεται ο χρήστης (π.χ. αναπαραγωγή ενός μουσικού κομματιού στο παρασκήνιο ή το κατέβασμα ενός αρχείου από το διαδίκτυο). Έτσι, το σύστημα συνεχίζει να εκτελεί τις λειτουργίες αυτές μέχρι το σημείο στο οποίο θα υπάρξει ανάγκη για απελευθέρωση μνήμης, γιατί δεν θα μπορούν να συνεχιστούν ταυτόχρονα με τις ορατές και τις διεργασίες προσκηνίου.

4.3.4 Διεργασίες στο παρασκήνιο (Background processes)

Μια διεργασία παρασκηνίου είναι η διεργασία που διατηρεί μια δραστηριότητα η οποία δεν είναι ούτε ενεργή, ούτε σε κατάσταση παύσης, ούτε ορατή. Για μια τέτοια δραστηριότητα έχει κληθεί η μέθοδος *onStop().* Τέτοιες διεργασίες μπορεί να τερματιστούν οποιαδήποτε στιγμή επειδή δεν έχουν καμία άμεση επιρροή στις αλληλεπιδράσεις και ενέργειες των χρηστών, προκειμένου να ανακτηθεί μνήμη για διεργασίες προσκηνίου, ορατές ή διεργασίες υπηρεσίας. Συνήθως υπάρχουν πολλές διεργασίες παρασκηνίου σε ενεργή κατάσταση και έτσι φυλάσσονται σε μια λίστα με τις πιο πρόσφατα χρησιμοποιημένες, έτσι ώστε να διασφαλίζεται ότι η διεργασία την οποία ο χρήστης χρησιμοποίησε πιο πρόσφατα θα τερματιστεί τελευταία.

4.3.5 Κενές διεργασίες (Empty processes)

Κενή διεργασία είναι η διεργασία η οποία δεν κρατά κανένα ενεργό βασικό στοιχείο της εφαρμογής. Αυτού του τύπου οι διεργασίες χρησιμοποιούνται σαν κρυφή μνήμη, με σκοπό να βελτιωθεί ο χρόνος εκκίνησης για ένα βασικό στοιχείο το οποίο θα χρειαστεί να εκτελεστεί σε αυτές. Το σύστημα τερματίζει αυτές τις διεργασίες για να κρατήσει μια ισορροπία μεταξύ των κρυφών μνημών του πυρήνα και των διεργασιών.

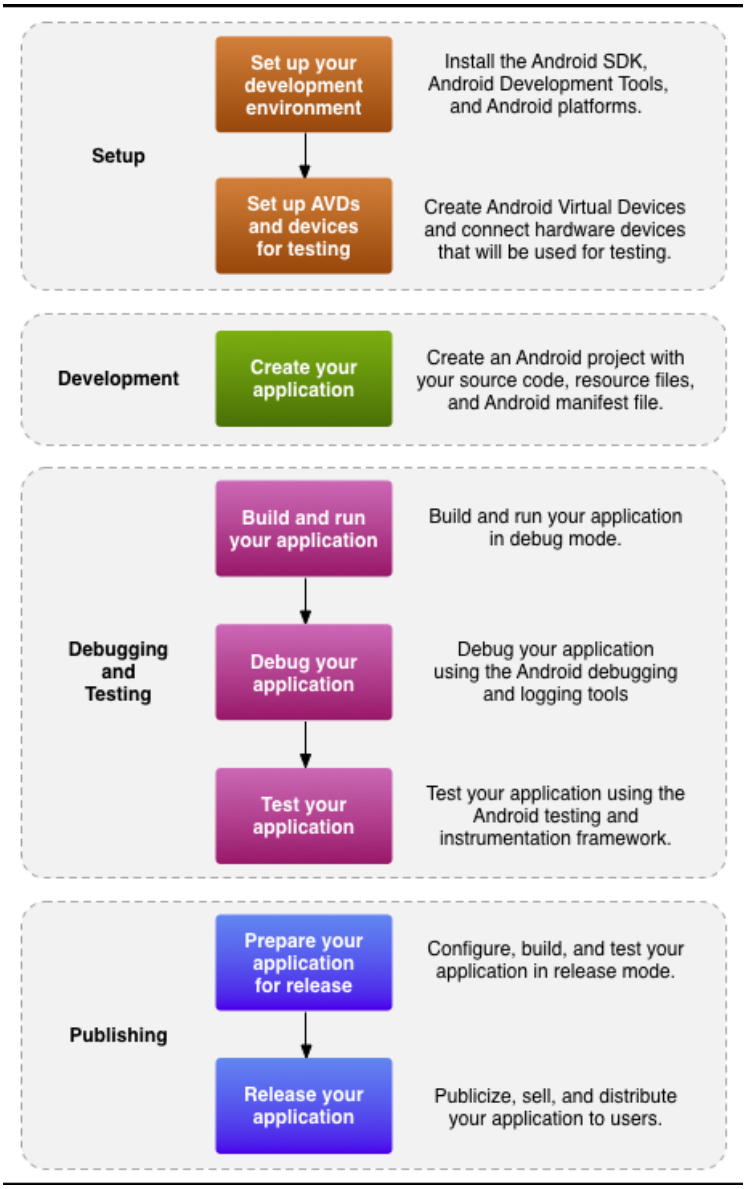
Κεφάλαιο 5

Εργαλεία ανάπτυξης εφαρμογών Android

Το περιβάλλον ανάπτυξης εφαρμογών Android βασίζεται σε 4 βασικά εργαλεία.

- Εγκατάσταση του περιβάλλοντος Java JDK (Java Development Kit) και JRE (Java Runtime Environment)
- Εγκατάσταση του Eclipse IDE (Integrated Development Environment) (Ολοκληρωμένο περιβάλλον προγραμματισμού σε Java)
- Παραμετροποίηση και εγκατάσταση του Android SDK (Software Development Kit)
- Εγκατάσταση του ADT (Android Development Tools) στο Eclipse

Η ανάπτυξη εφαρμογών για συσκευές Android διευκολύνεται από μια ομάδα εργαλείων που παρέχονται στο SDK. Η πρόσβαση σε αυτά τα εργαλεία επιτυγχάνεται μέσω ενός πρόσθετου στοιχείου στο περιβάλλον Eclipse που ονομάζεται Eclipse ADT ή από τη γραμμή εντολών. Η ανάπτυξη εφαρμογών μέσω του περιβάλλοντος Eclipse είναι η πιο συνηθισμένη μέθοδος, αφού παρέχει τη δυνατότητα να επικαλείται άμεσα τα εργαλεία που χρειάζονται κατά την ανάπτυξη της εφαρμογής. Τα βασικά βήματα για την ανάπτυξη εφαρμογών φαίνονται στην παρακάτω εικόνα:



Σχήμα 7:Στάδια ανάπτυξης εφαρμογής Android

Ρύθμιση(Setup)

Κατά τη διάρκεια αυτού του σταδίου θα πρέπει να εγκατασταθεί και να ρυθμιστεί το περιβάλλον ανάπτυξης. Επίσης, δίνεται η δυνατότητα δημιουργίας εικονικών συσκευών Android (AVDs-Android Virtual Devices) ή η σύνδεση συσκευών υλικού στις οποίες μπορούν να εγκατασταθούν οι εφαρμογές.

Ανάπτυξη (Development)

Κατά τη διάρκεια αυτού του επιπέδου γίνεται η ανάπτυξη και η δημιουργία του project της Android εφαρμογής. Το project περιλαμβάνει όλο τον πηγαίο κώδικα και τα αρχεία πόρων για την εφαρμογή.

Εντοπισμός σφαλμάτων και δοκιμή (Debugging and Testing)

Κατά τη φάση αυτή, δίνεται η δυνατότητα δημιουργίας του προγράμματος σε ένα πακέτο .apk με δυνατότητες αποσφαλμάτωσης, το οποίο μπορεί να εγκατασταθεί και να εκτελεστεί στον εξομοιωτή ή στη συσκευή Android. Εάν χρησιμοποιείται το περιβάλλον υλοποίησης Eclipse, τότε τα πακέτα αυτά δημιουργούνται κάθε φορά που γίνεται αποθήκευση του έργου. Στη συνέχεια ο προγραμματιστής μπορεί να διορθώσει τον κώδικα της εφαρμογής χρησιμοποιώντας ένα συμβατό πρωτόκολλο εντοπισμού σφαλμάτων μαζί με τα εργαλεία εντοπισμού και καταγραφής σφαλμάτων που παρέχονται με το Android SDK.

Έκδοση της Εφαρμογής (Publishing)

Η φάση αυτή σχετίζεται με τη ρύθμιση της εφαρμογής για την απελευθέρωσή της δημοσίως και τη διανομή της εφαρμογής στους χρήστες.

5.1 Android Software Development Kit (SDK)

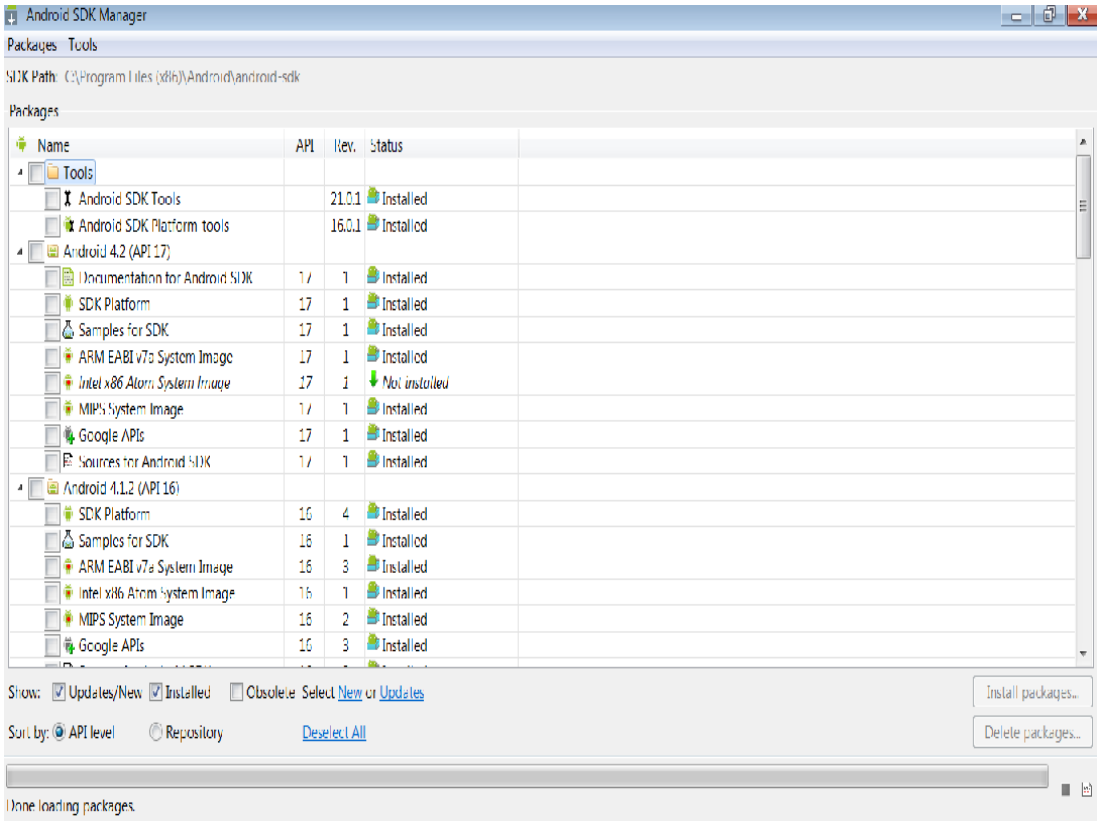
Το Android Software Development Kit (SDK) περιλαμβάνει μια πλήρη σειρά από εργαλεία ανάπτυξης. Τα εργαλεία αυτά αποτελούνται από ένα πρόγραμμα εντοπισμού σφαλμάτων, βιβλιοθήκες των διεπαφών προγραμματισμού εφαρμογών, έναν εξομοιωτή συσκευής, δείγματα κώδικα εφαρμογών καθώς και σχετικά εγχειρίδια. Το λογισμικό SDK υποστηρίζει τη χρήση παλιότερων εκδόσεων της πλατφόρμας Android σε περίπτωση που οι προγραμματιστές επιθυμούν την εκτέλεση των εφαρμογών τους στις εκδόσεις αυτές και είναι απαραίτητο για την δημιουργία, δοκιμή και αποσφαλμάτωση των Android εφαρμογών. Η εργαλειοθήκη SDK ξεχωρίζει τα εργαλεία, τις πλατφόρμες και τα άλλα στοιχεία που χρησιμοποιεί σε πακέτα τα οποία μπορούν οι προγραμματιστές των Android εφαρμογών να μεταφορτώσουν μέσω του Διαχειριστή SDK. Επιπλέον, εάν κυκλοφορήσει μια καινούρια έκδοση της πλατφόρμας Android ή κάποιο από τα ήδη υπάρχοντα εργαλεία αναβαθμιστούν, μέσω του διαχειριστή SDK δίνεται η δυνατότητα μεταφόρτωσης των νέων στοιχείων στο περιβάλλον εργασίας του προγραμματιστή.

Ένα σημαντικό γεγονός που διευκολύνει την ανάπτυξη εφαρμογών είναι πως το πακέτο Android SDK συνεργάζεται με το Eclipse και έτσι ο προγραμματιστής μπορεί εύκολα και γρήγορα να βλέπει τις αλλαγές του κώδικα στον εξομοιωτή που του παρέχει το Android SDK, χωρίς να χρειάζεται να εγκαθιστά την εφαρμογή σε μια πραγματική κινητή συσκευή.

Χαρακτηριστικά του SDK

Μερικά χαρακτηριστικά του εργαλείου SDK είναι :

- SQLite για δομημένη αποθήκευση δεδομένων
- Φωτογραφική μηχανή, πυξίδα, GPS και επιταχυνσιόμετρο
- Bluetooth, EDGE, 3G, WiFi
- Ολοκληρωμένο πρόγραμμα περιήγησης
- Βελτιστοποιημένη Dalvik εικονική μηχανή για κινητές συσκευές
- Υποστήριξη πολυμέσων για αρχεία ήχου, βίντεο, εικόνων
- GSM τηλεφωνία
- Βελτιστοποιημένα γραφικά



Σχήμα 8:SDK διαχειριστής

Ο εξομοιωτής (emulator)- Dalvik Virtual Machine (DVM)

Ο εξομοιωτής είναι μια εικονική κινητή συσκευή η οποία τρέχει το λειτουργικό σύστημα του Android και προσομοιώνει ένα μεγάλο πλήθος λειτουργιών μιας τυπικής συσκευής. Κάθε μια από τις εικονικές συσκευές επάνω στις οποίες θα δοκιμάσουμε την εφαρμογή μας περιγράφεται από συγκεκριμένες ρυθμίσεις, οι οποίες αποθηκεύονται σε ένα Android Virtual Device (AVD). Μέσα από το Android SDK και τον AVD Manager μπορούμε να δημιουργήσουμε ένα AVD. Ο διαχειριστής AVD παρέχει μια γραφική διεπαφή χρήστη μέσα από την οποία ο προγραμματιστής μπορεί να δημιουργήσει και να ελέγχει τις Android εικονικές συσκευές. Χρησιμοποιώντας το περιβάλλον Eclipse ο διαχειριστής αυτός μπορεί να ξεκινήσει είτε πατώντας το κατάλληλο εικονίδιο από την γραμμή εργαλείων είτε επιλέγοντας από το μενού Window → AVD Manager. Όταν μια εφαρμογή εκτελείται σε έναν εξομοιωτή τότε μπορεί να χρησιμοποιεί υπηρεσίες της πλατφόρμας Android έχοντας τη δυνατότητα να επικαλεστεί άλλες εφαρμογές, να έχει πρόσβαση στο διαδίκτυο, να αναπαράγει αρχεία ήχου και εικόνας, να αποθηκεύει και να ανακτά δεδομένα, να ειδοποιεί τον χρήστη και να προσφέρει αλλαγές στα γραφικά και στα θέματα που χρησιμοποιεί η συσκευή. Επιπλέον, οι εικονικές συσκευές Android επιτρέπουν στον προγραμματιστή να καθορίσει συγκεκριμένα χαρακτηριστικά υλικού για την συσκευή και να δημιουργήσει πολλές διαμορφώσεις προκειμένου να ελέγξει διάφορες πλατφόρμες του Android.

Δημιουργία Εξομοιωτή

Η δημιουργία της εικονικής συσκευής Android είναι απαραίτητη μιας και αντιπροσωπεύει το σύστημα στο οποίο θα τρέξει η εφαρμογή. Η δημιουργία μιας AVD μπορεί να γίνει ακολουθώντας τα παρακάτω βήματα:

1. Κάνουμε εκκίνηση του διαχειριστή AVD

2. Επιλογή New στο παράθυρο του διαχειριστή

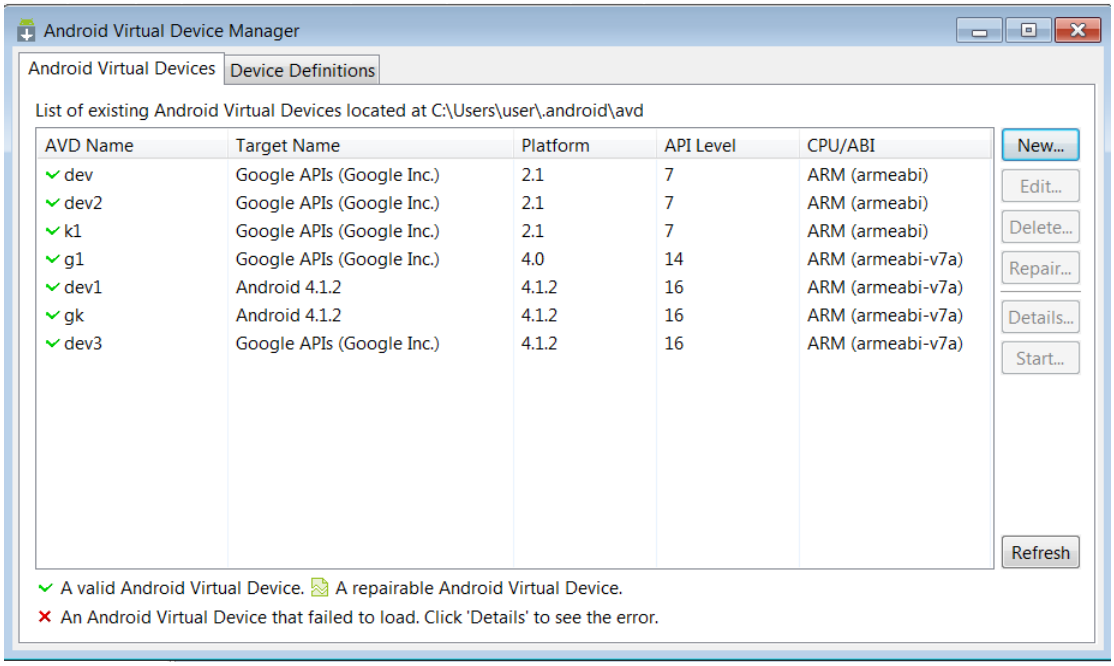
3. Επιλογή ονόματος του AVD

4. Επιλογή έκδοσης του Android που θα τρέχει στον εξομοιωτή

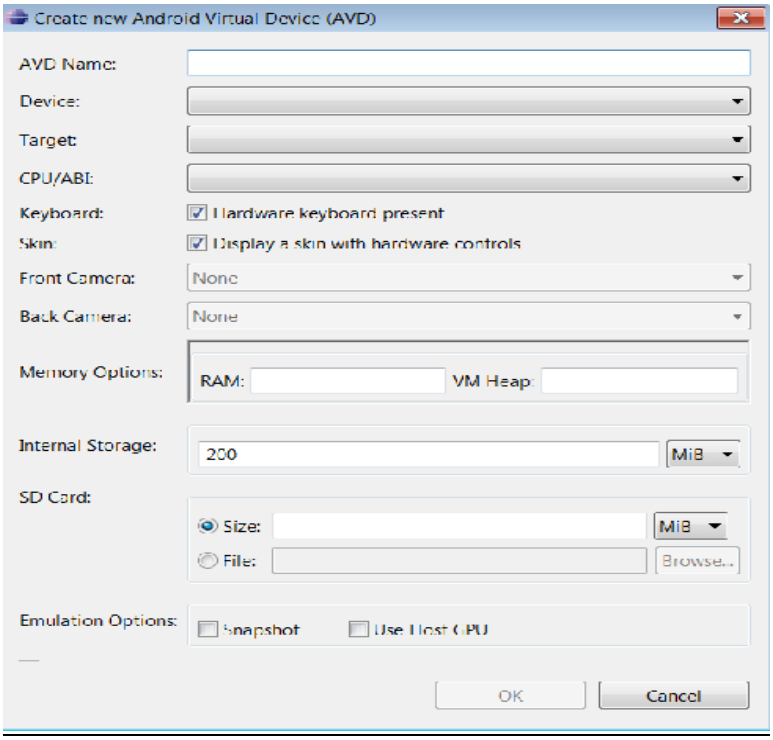
5. Επιλογή του μεγέθους της SD κάρτας και της ανάλυσης οθόνης

6. Επιλογή του υλικού που θα επιθυμούμε να υποστηρίζει ο εξομοιωτής

7. Επιλογή Create AVD



Σχήμα 9:Διαχειριστής Android Virtual Device



Σχήμα 10:Δημιουργία νέου AVD

Σχήμα 11: Android Virtual Machine

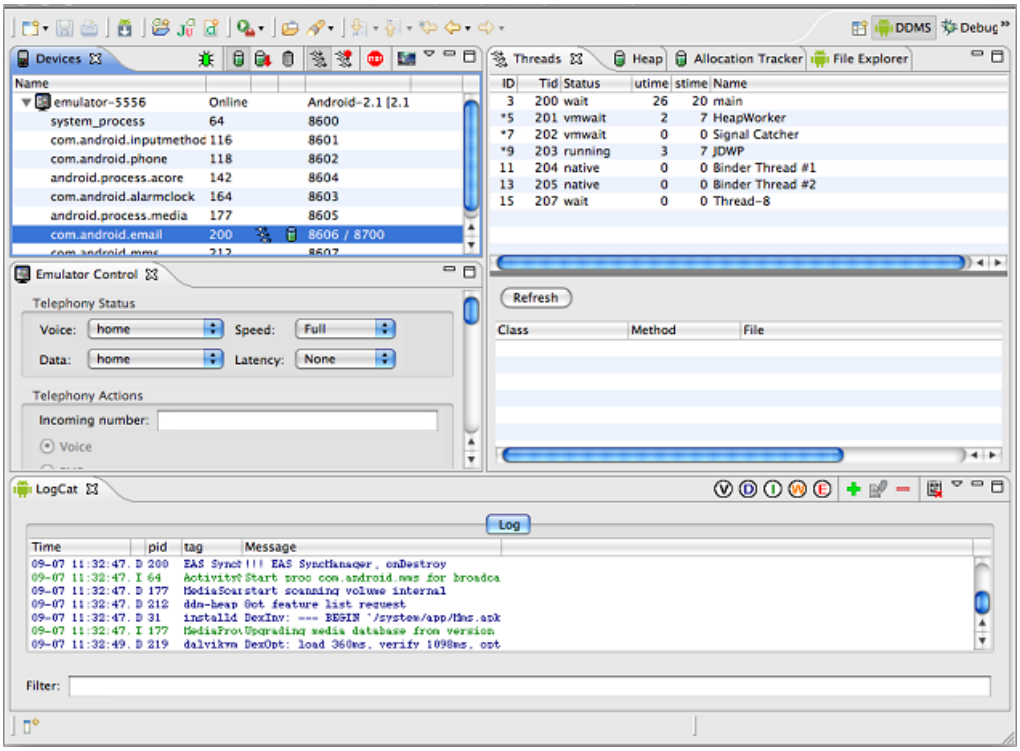
Μερικές λειτουργίες που μπορεί να προσομοιώσει ο εξομοιωτής είναι:

- Λήψη δεδομένων θέσης από τη συσκευή GPS
- Ποικιλία πλήκτρων πλοήγησης και ελέγχου
- Οθόνη για την προβολή των εκτελούμενων εφαρμογών
- Επιτρέπει στις εφαρμογές τη χρήση των υπηρεσιών που προσφέρει η πλατφόρμα του Android, δηλαδή την κλήση άλλων εφαρμογών, την πρόσβαση στο δίκτυο, την αναπαραγωγή ήχου και βίντεο, την αποθήκευση και επαναφορά δεδομένων, την ειδοποίηση χρήστη, το γραφικό περιβάλλον του Android.

Dalvik Debug Monitor Service (DDMS)

Το Android SDK, για την ανάπτυξη εφαρμογών περιλαμβάνει ένα εργαλείο αποσφαλμάτωσης το οποίο λέγεται DDMS. Το εργαλείο αυτό επιτρέπει τη διαχείριση των διεργασιών στον εξομοιωτή ή στη συσκευή. Πιο συγκεκριμένα, παρέχει υπηρεσίες port-forwarding, λήψη screenshots, πληροφορίες που αφορούν τα νήματα και τον σωρό, logcat, πληροφορίες σχετικές με διεργασίες και πληροφορίες ράδιο, προσομοίωση δεδομένων θέσης και άλλα. Αφού εγκαταστήσουμε την εφαρμογή μας στην εικονική συσκευή μπορούμε να μεταβούμε στο DDMS επιλέγοντας στο Eclipse από το μενού Window → Open Perspective → DDMS. Στο Android λειτουργικό σύστημα κάθε εφαρμογή εκτελείται στη δική της διεργασία και κάθε μια από αυτές τρέχει στη δική της εικονική

μηχανή(VM). Κάθε εικονική μηχανή εκθέτει μια μοναδική θύρα στην οποία μπορεί να συνδεθεί ο debugger. Όταν ξεκινά το DDMS συνδέεται στην android debug bridge (adb), ένα εργαλείο γραμμής εντολών που επιτρέπει την επικοινωνία με ένα στιγμιότυπο εξομοιωτή. Όταν μια συσκευή συνδέεται, δημιουργείται μια υπηρεσία ελέγχου εικονικής μηχανής μεταξύ της adb και του DDMs, η οποία ειδοποιεί το DDMS όταν ξεκινάει ή τερματίζεται μια εικονική μηχανή στη συσκευή. Όταν εκτελείται μια εικονική μηχανή, η υπηρεσία DDMS ανακτά το ID της διεργασίας (pid) μέσω της γέφυρας adb και ανοίγει μια σύνδεση με τον debugger της εικονικής μηχανής μέσω του δαίμονα adb της συσκευής. Κατ΄ αυτόν τον τρόπο το DDMS μπορεί να έχει επικοινωνία με την εικονική μηχανή μέσω ενός πρωτοκόλλου.



Σχήμα 12:DDMS Perspective

5.2 Android Development Tools (ADT)

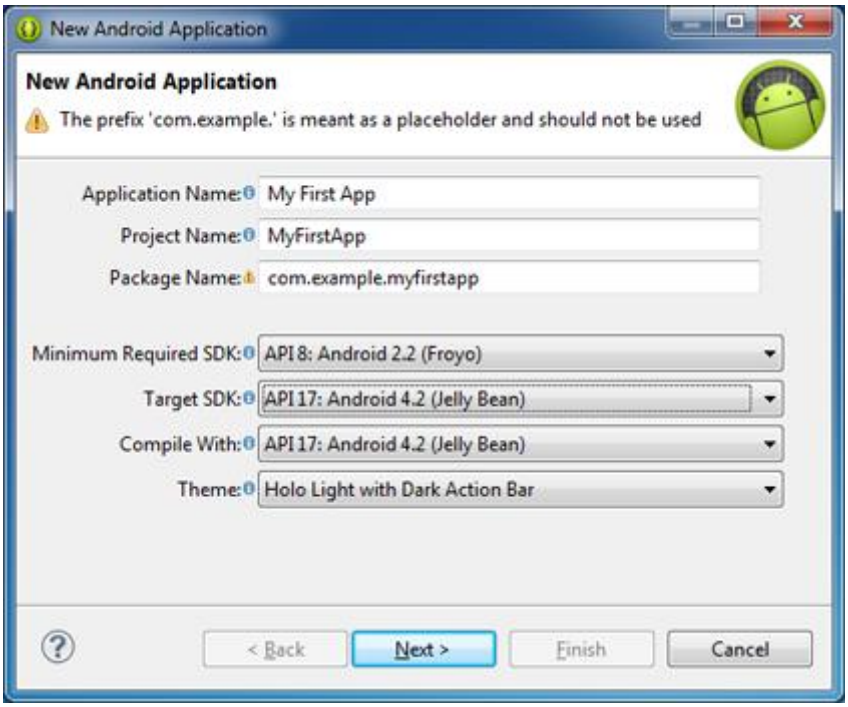
Το ADT είναι ένα πρόσθετο στοιχείο στο Eclipse το οποίο έχει σχεδιαστεί για να δώσει ένα δυναμικό και ενσωματωμένο περιβάλλον ανάπτυξης των Android εφαρμογών. Το πρόσθετο αυτό επεκτείνει τις ικανότητες του Eclipse και επιτρέπει στον προγραμματιστή να δημιουργεί γρήγορα Android έργα και διεπαφές εφαρμογών, να προσθέτει πακέτα βασισμένα στο πλαίσιο προγραμματιστικών διεπαφών του Android, να γίνεται έλεγχος λαθών στις εφαρμογές που

αναπτύσσονται μέσω των εργαλείων SDK, ακόμα και να κάνει εξαγωγή αρχείων .apk προκειμένου να διαμοιράσει την εφαρμογή.

5.3 Βασικά μέρη Εφαρμογών

Οι Android εφαρμογές γράφονται σε γλώσσα προγραμματισμού Java. Ο κώδικας μεταγλωττίζεται με τη βοήθεια των εργαλείων SDK σε ένα πακέτο Android, αρχείο με κατάληξη .apk. Όλος ο κώδικας σαν ένα μόνο αρχείο .apk θεωρείται σαν μια εφαρμογή και είναι το αρχείο το οποίο χρησιμοποιούν οι συσκευές για να εγκαταστήσουν την εφαρμογή. Μέσα από τη δημιουργία μιας απλής Android εφαρμογής θα αναλύσουμε τα βασικά μέρη από τα οποία αποτελείται. Αφού έχει γίνει η εγκατάσταση του Android SDK, του πρόσθετου ADT για το Eclipse και έχουν ληφθεί οι απαραίτητες ενημερώσεις για τα εργαλεία και τις πλατφόρμες μέσω του διαχειριστή SDK μπορεί να ξεκινήσει η δημιουργία μιας εφαρμογής Android.

Το πρώτο βήμα είναι μέσω του Eclipse να δημιουργήσουμε ένα νέο project εφαρμογής Android, επιλέγοντας *New → Android Application Project → Next*. Η φόρμα που εμφανίζεται φαίνεται στην παρακάτω εικόνα.



Σχήμα 13:Φόρμα δημιουργίας νέου project για android εφαρμογή

Στο στάδιο αυτό επιλέγουμε το όνομα της εφαρμογής το οποίο θα εμφανίζεται στους χρήστες, το όνομα του project που είναι το όνομα του φακέλου στον οποίο βρίσκεται το project, τον χώρο ονομάτων του πακέτου της εφαρμογής. Το πεδίο *Minimum Required SDK* είναι η παλαιότερη έκδοση του Android που μπορεί να υποστηρίξει η εφαρμογή και υποδηλώνεται χρησιμοποιώντας το επίπεδο API. Το πεδίο *Target SDK* υποδηλώνει την πιο πρόσφατη έκδοση του Android με την οποία θα δοκιμαστεί η εφαρμογή. Επιπλέον, θα πρέπει στα επόμενα βήματα να δοθεί ένα όνομα στη δραστηριότητα, το οποίο θα είναι το όνομα της κλάσης που θα παραχθεί και θα είναι μια υποκλάση της κλάσης Activity του Android. Μετά την ολοκλήρωση αυτών των βημάτων γίνεται η δημιουργία του φακέλου του project τον οποίο μπορεί να δει ο προγραμματιστής με την βοήθεια του *Package Explorer* στο περιβάλλον ανάπτυξης Eclipse.

5.3.1 Κλάσεις Java

Ο κατάλογος /src προέρχεται από τη συντομογραφία της λέξης source που σημαίνει πηγή. Σε αυτόν βρίσκεται όλος ο πηγαίος κώδικας σε Java, δηλαδή οι κλάσεις πάντα όμως κάτω από ένα πακέτο.

5.3.2 Αρχείο AndroidManifest.xml

Το manifest είναι ένα δομημένο XML αρχείο που πάντα έχει την ονομασία AndroidManifest.xml για όλες τις εφαρμογές. Το αρχείο AndroidManifest.xml είναι ένα αρχείο που απαιτείται για κάθε εφαρμογή Android. Το αρχείο αυτό περιγράφει κάθε συστατικό της εφαρμογής και πώς αλληλεπιδρά με τα υπόλοιπα. Βρίσκεται στον root φάκελο της εφαρμογής και έχει τα εξής χαρακτηριστικά:

- Ονομάζει το java πακέτο της εφαρμογής
- Περιγράφει τα στοιχεία της εφαρμογής, δηλαδή τις δραστηριότητες, τους παρόχους περιεχομένου, τις υπηρεσίες και τους δέκτες προθέσεων
- Ονομάζει τις κλάσεις που υλοποιούν κάθε ένα στοιχείο και δημοσιεύει τις δυνατότητές του
- Ορίζει ποιες άδειες πρέπει να έχει η εφαρμογή ώστε να έχει πρόσβαση σε προστατευόμενα μέρη του API και να αλληλεπιδρά με άλλες εφαρμογές
- Ορίζει το ελάχιστο επίπεδο του android API που απαιτεί η εφαρμογή
- Παραθέτει τις βιβλιοθήκες με τις οποίες πρέπει να συνδέεται η εφαρμογή

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.app"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.app.MainActivity"
            android:label="@string/app_name" >
      <intent-filter>
       <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Σχήμα 14: Αρχείο AndroidManifest.xml

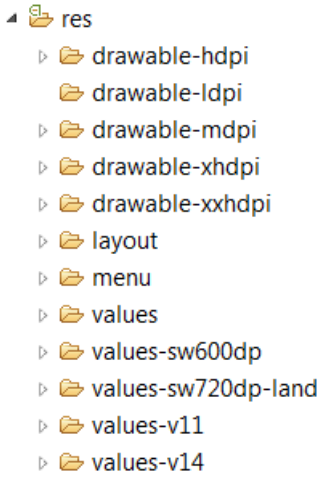
Από τον παραπάνω κώδικα θα αναλύσουμε τις βασικές ετικέτες που χρησιμοποιούνται.

Η ετικέτα <xml> καθορίζει ότι το αρχείο αυτό περιέχει κώδικα σε μορφή xml. Το όρισμα version ενημερώνει για την έκδοση xml (έκδοση 1.0) και το όρισμα encoding ενημερώνει για την κωδικοποίηση που χρησιμοποιείται (utf-8). Η ετικέτα <manifest> αποτελεί την αρχική ετικέτα του αρχείου και θα πρέπει να περιέχει την ετικέτα <application> και προαιρετικά μπορεί να περιλαμβάνει και άλλες. Με το όρισμα package δηλώνεται το πακέτο που χρησιμοποιείται (com.example.app), με το όρισμα *android:versionCode* δηλώνεται η έκδοση του κώδικα η οποία δεν είναι ορατή στον χρήστη και το *android:versionName* δηλώνουμε τον αριθμό έκδοσης που θέλουμε να εμφανίζεται στον χρήστη. Η ετικέτα <uses-sdk> εκφράζει τη συμβατότητα της εφαρμογής με ένα ή περισσότερα επίπεδα του API. Μπορεί να έχει τα ορίσματα *android:minSdkVersion* (εδώ η έκδοση 8), *android:targetSdkVersion* και *android:maxSdkVersion*. Η ετικέτα <application> ορίζει την εφαρμογή και μπορεί να περιέχει άλλες ετικέτες όπως <activity>, <service>, <provider>. Με το γνώρισμα *android:icon* δηλώνουμε το εικονίδιο που θα εμφανίζει η εφαρμογή μας στον χρήστη, με το *android:label* καθορίζεται το όνομα της εφαρμογής που θα εμφανίζεται στον χρήστη και με το *android:debuggable* ορίζουμε αν μπορεί ή όχι να γίνεται αποσφαλμάτωση. Η ετικέτα <activity> δηλώνει μια

δραστηριότητα και το όρισμα *android:name* δηλώνει το όνομα της υποκλάσης η οποία υλοποιεί τη δραστηριότητα. Τα γνωρίσματα icon και label αναφέρονται σε φακέλους που περιέχουν μια εικόνα ή μια ετικέτα που χαρακτηρίζουν τη δραστηριότητα. Τα υπόλοιπα συστατικά δηλώνονται με παρόμοιο τρόπο, <service> για τα services, <receiver> για τα broadcast receivers και <provider> για τα content providers. Συστατικά τα οποία δεν έχουν δηλωθεί στο αρχείο manifest δεν είναι ορατά από το σύστημα και συνεπώς δε μπορούν να εκτελεστούν. Η ετικέτα <intent-filter> καθορίζει τους τύπους των intents που ανταποκρίνεται το συγκεκριμένο activity. Η ετικέτα αυτή συμπεριλαμβάνεται σε μια ετικέτα <activity>, <service>, <receiver> και πρέπει να περιέχει την ετικέτα <action>. Στο παράδειγμα του αρχείου manifest η ετικέτα περιέχει το όρισμα *android:name* με την τιμή *android.intent.action.MAIN* που δηλώνει ότι η δραστηριότητα δεν χρειάζεται δεδομένα ώστε να ξεκινήσει. Η <category> είναι μια άλλη ετικέτα η οποία περιέχει πληροφορίες για το είδος της συνιστώσας που πρέπει να χειριστεί το intent. Το όρισμα *android:name* με τιμή *android.intent.category.LAUNCHER* δηλώνει πως η δραστηριότητα είναι πρώτη στη στοίβα.

5.3.3 Resources

Ο κατάλογος /res το όνομα του οποίου προέρχεται από τη συντομογραφία της λέξης resources περιέχει τους πόρους στους οποίους θα γίνεται αναφορά μέσω των αρχείων του project. Ο όρος πόροι μπορεί να αναφέρεται σε αρχεία εικόνας, ήχου ή οποιοδήποτε αρχείο δεν έχει τη μορφή κώδικα.



Σχήμα 15: Υποκατάλογοι του καταλόγου res

Οι πόροι ανάλογα με τον τύπο τους αποθηκεύονται σε ξεχωριστό κατάλογο. Οι εικόνες τύπου jpeg ή png αποθηκεύονται στον κατάλογο /res/drawable. Ο κατάλογος /res/layout περιλαμβάνει τα αρχεία .xml τα οποία καθορίζουν την

εμφάνιση που θα έχει η εφαρμογή. Ο φάκελος /res/values περιέχει και αυτός αρχεία xml τα οποία περιλαμβάνουν τιμές στις οποίες θα αναφερθεί η εφαρμογή. Αυτά τα αρχεία xml δηλώνουν πίνακες, χρώματα, διαστάσεις, αλφαριθμητικές τιμές κλπ. Ο μεταγλωττιστής πόρων χρησιμοποιείται για τη μεταγλώττιση όλων των πόρων συμπιέζοντάς τους και δημιουργεί μια κλάση με το όνομα R η οποία βρίσκεται στο κατάλογο /gen.

5.3.4 Αρχεία

Κατάλογος gen

Ο κατάλογος /gen περιέχει την κλάση java με το όνομα R. Μέσω της R κλάσης μπορούμε να αναφερόμαστε στους πόρους μέσα από το πρόγραμμά μας. Η κλάση αυτή διαχειρίζεται αυτόματα από το android plugin του Eclipse και έτσι κάθε φορά που προσθέτουμε ένα αρχείο στον κατάλογο /res το plugin προσθέτει τα κατάλληλα αναγνωριστικά των πόρων στο αρχείο R.java. Ένα παράδειγμα της μορφής ενός τέτοιου αρχείου φαίνεται στην παρακάτω εικόνα:

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

package com.javacodegeeks.android.googlemaps;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int bubble=0x7f020000;
        public static final int cross=0x7f020001;
        public static final int ic_action_search=0x7f020002;
        public static final int ic_launcher=0x7f020003;
        public static final int ic_launcher1=0x7f020004;
        public static final int mark=0x7f020005;
        public static final int marker_default=0x7f020006;
        public static final int marker_hotel=0x7f020007;
        public static final int marker_monu=0x7f020008;
        public static final int marker_museum=0x7f020009;
        public static final int mylocation=0x7f02000a;
        public static final int normalview=0x7f02000b;
        public static final int poi=0x7f02000c;
```

```
    public static final int popup_frame=0x7f02000d;
    public static final int pwdremover=0x7f02000e;
    public static final int ruins_marker=0x7f02000f;
   public static final int satelliteview=0x7f020010;
        public static final int zoo_marker=0x7f020011;
    }
    public static final class id {
        public static final int button1=0x7f070003;
        public static final int container=0x7f070000;
        public static final int infoBubble=0x7f070009;
        public static final int locationName=0x7f07000a;
        public static final int map_view=0x7f070001;
        public static final int mapview_norm=0x7f070008;
        public static final int mapview_satellite=0x7f070007;
        public static final int menu_settings=0x7f07000b;
        public static final int my_location=0x7f070006;
        public static final int textView1=0x7f070002;
        public static final int tv_lat=0x7f070004;
        public static final int tv_lng=0x7f070005;
    }
    public static final class layout {
        public static final int activity_gmaps=0x7f030000;
        public static final int menu=0x7f030001;
        public static final int popup=0x7f030002;
    }
    public static final class menu {
        public static final int activity_gmaps=0x7f060000;
    }
    public static final class string {
        public static final int Lat_=0x7f04000a;
        public static final int Lon_=0x7f04000b;
        public static final int TextView=0x7f040009;
        public static final int X_=0x7f04000c;
        public static final int Y_=0x7f04000d;
        public static final int app_name=0x7f040000;
        public static final int hello_world=0x7f040001;
        public static final int lat=0x7f040005;
        public static final int longti=0x7f040007;
        public static final int menu_settings=0x7f040002;
        public static final int msg=0x7f04000e;
        public static final int points=0x7f040004;
        public static final int select_poi=0x7f040008;
        public static final int title_activity_gmaps=0x7f040003;
        public static final int unknown=0x7f040006;
    }
    public static final class style {
        public static final int AppTheme=0x7f050000;
    }
}
```

Σχήμα 16: Παράδειγμα αρχείου R.java

Φάκελος Assets

Στον φάκελο *assets* αποθηκεύονται οι πόροι της εφαρμογής τους οποίους δεν θέλουμε να διαχειριστούμε σαν να ήταν πόροι (δεδομένα της εφαρμογής όπως αρχεία ή κατάλογοι).

Βιβλιοθήκη

Η βιβλιοθήκη περιέχει αρχεία με την κατάληξη .jar (Java archives). Τα αρχεία αυτά ορίζουν ένα άθροισμα πολλών αρχείων σε ένα και χρησιμοποιούνται για την διανομή κλάσεων java και μεταδεδομένων. Το όνομα της βιβλιοθήκης είναι είτε “Android” ακολουθούμενο από την έκδοση την οποία έχουμε επιλέξει, είτε Google APIs ακολουθούμενο από την έκδοση Android σε αγκύλες, για παράδειγμα Google APIs [Android 4.1.2].

Κεφάλαιο 6

Σχεδιασμός και ανάπτυξη της εφαρμογής

6.1 Αρχιτεκτονική της εφαρμογής

Google Maps για κινητά

Το βασικό μέρος της εφαρμογής, που αναπτύχθηκε για την εργασία αυτή, σχετίζεται άμεσα με τη χρήση των Google maps, μιας υπηρεσίας διαδικτυακής χαρτογράφησης, που παρέχεται από την Google. Τα Google maps android APIs αποτελούν μέρος της πλατφόρμας υπηρεσιών Google Play και επιτρέπουν στους προγραμματιστές Android εφαρμογών την ενσωμάτωση διαδραστικών και «πλούσιων» σε δυνατότητες χαρτών. Η ενσωμάτωση των χαρτών Google σε Android εφαρμογές είναι υλοποιήσιμη με δύο τρόπους .

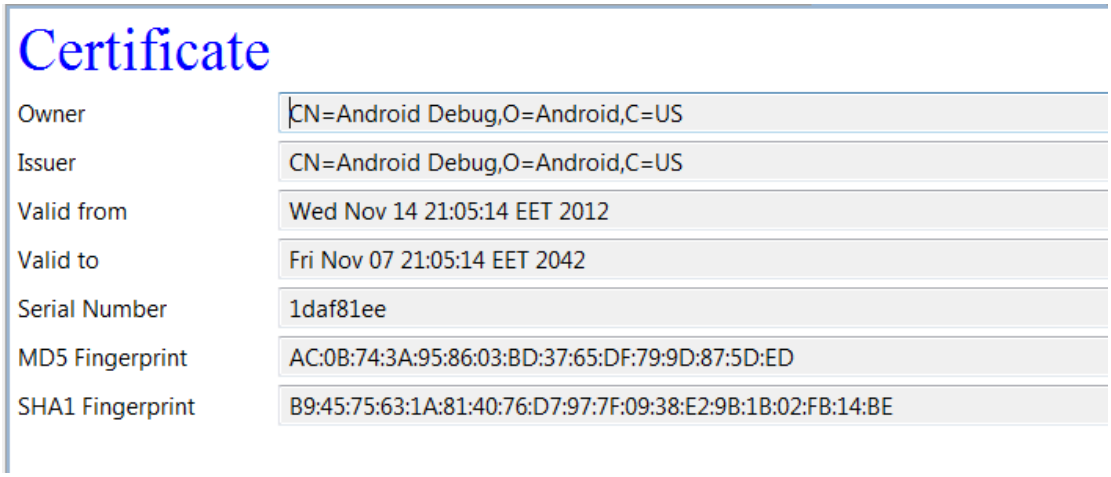
Ο πρώτος τρόπος γίνεται με τη χρησιμοποίηση της όψης 'webview' μέσα στην εφαρμογή. Έτσι επιτυγχάνεται η απεικόνιση οποιασδήποτε ιστοσελίδας όπως αυτή θα εμφανιζόταν σε έναν φυλλομετρητή στον υπολογιστή και κατ΄επέκταση και των χαρτών της Google. Ο δεύτερος τρόπος ενσωμάτωσης, ο οποίος έχει χρησιμοποιηθεί και για τη συγκεκριμένη εφαρμογή, υλοποιείται με τη χρήση του στοιχείου 'MapView' το οποίο σε συνδυασμό με το MapActivity και το Google maps API δίνει τη δυνατότητα στους προγραμματιστές να εκμεταλλευτούν τους υπάρχοντες χάρτες. Επιπλέον, με τη βοήθεια της εξωτερικής βιβλιοθήκης που παρέχει το Google Maps και το πακέτο *com.google.android.maps* το οποίο περιέχει η βιβλιοθήκη, ο προγραμματιστής της εφαρμογής έχει τη δυνατότητα να προσθέσει ισχυρές δυνατότητες χαρτογράφησης. Η βιβλιοθήκη αυτή θα πρέπει να προστεθεί στο αρχείο AndroidManifest.xml. Η βασική κλάση του πακέτου *com.google.android.maps* είναι η MapView η οποία είναι υποκλάση της View. Η MapView είναι μια όψη η οποία εμφανίζει έναν χάρτη με δεδομένα που παίρνει από τους εξυπηρετητές της Google και μπορεί να υποστηρίξει επάνω της άλλα views όπως LinearLayouts, buttons, checkboxes, textviews. Ο χάρτης μπορεί να εμφανιστεί με διάφορες όψεις, απλός χάρτης, δορυφορική όψη, κίνηση στους δρόμους, ενώ προσφέρεται η δυνατότητα εστίασης (zoom) σε οποιοδήποτε σημείο επιθυμεί ο χρήστης μέσω των κουμπιών που εμφανίζονται στο κάτω μέρος του χάρτη. Η βασική κλάση της εφαρμογής θα πρέπει να επεκτείνει την κλάση MapActivity, μια υποκλάση της android.app.Activity. Η υποκλάση αυτή παρέχει σημαντικές

δυνατότητες για τους χάρτες, ενώ στο εσωτερικό κάθε κλάσης MapActivity θα πρέπει να περιλαμβάνεται η μέθοδος *isRouteDisplayed()*. Η μέθοδος αυτή είναι απαραίτητη προκειμένου να γνωρίζει ο server εάν εμφανίζονται ή όχι πληροφορίες για τις διαδρομές.

Σε πολλές περιπτώσεις, οι προγραμματιστές επιθυμούν την προσθήκη και τη δημιουργία δικών τους στοιχειών στο χάρτη. Προκειμένου να γίνει αυτό εφικτό θα πρέπει να χρησιμοποιηθεί η κλάση ***ItemizedOverlay***. Η κλάση αυτή διαχειρίζεται ένα σύνολο αντικειμένων Overlay (μεμονωμένα αντικείμενα τα οποία τοποθετούνται στο χάρτη) σαν μια λίστα από ***OverlayItems***. Για να προστεθεί ένα αντικείμενο πάνω στον χάρτη θα πρέπει να δημιουργηθεί ένα στιγμιότυπο της κλάσης Overlay και να προστεθεί στη λίστα που αντλείται από τη μέθοδο *MapView.getOverlays().* Μερικές σημαντικές μέθοδοι της κλάσης Overlay είναι η *draw()* με την οποία σχεδιάζεται το Overlay πάνω στον χάρτη και η *onTap()* με την οποία μπορούν να διαχειριστούν τα «πατήματα» στα αντικείμενα του χάρτη. Μια ακόμα σημαντική κλάση που σχετίζεται με τη χρησιμοποίηση χαρτών είναι η Geopoint. Η κλάση αυτή αναπαριστά ένα ζευγάρι από γεωγραφικά μήκη και πλάτη σε micro degrees.

Αρχικά θα πρέπει με τη βοήθεια του διαχειριστή Android SDK να εγκατασταθεί το στοιχείο Google play services και στη συνέχεια να γίνει εισαγωγή της βιβλιοθήκης στο Eclipse. Προκειμένου να μπορέσει να χρησιμοποιηθεί ένας χάρτης google σε μια εφαρμογή θα πρέπει ο προγραμματιστής να αποκτήσει ένα έγκυρο κλειδί για τον χάρτη. Το κλειδί αυτό είναι δωρεάν και μπορεί να χρησιμοποιηθεί για οποιαδήποτε εφαρμογή καλέσει το Maps API, ενώ υποστηρίζει άπειρο αριθμό χρηστών. Για τη διαδικασία της δημιουργίας κλειδιού μέσω της κονσόλας Google APIs θα πρέπει να παρέχεται το κλειδί υπογραφής της εφαρμογής καθώς και το όνομα του πακέτου. Κάθε κλειδί είναι μοναδικό και σχετίζεται με ένα συγκεκριμένο πιστοποιητικό, βασισμένο στο αποτύπωμα MD5 του πιστοποιητικού. Με τη βοήθεια ενός προσθέτου για το Eclipse δημιουργείται το πιστοποιητικό της εφαρμογής και στη συνέχεια με την παροχή του MD5 αποτυπώματος στην σελίδα https://developers.google.com/maps/documentation/android/v1/maps-api-signup?hl=el

παράγεται το map API κλειδί.



Σχήμα 17: πιστοποιητικό εφαρμογής με τη βοήθεια του plugin keytool

Σχήμα 18: Χάρτης Google σε Android συσκευή

Android και GPS

Μια από τις πιο ενδιαφέρουσες κατηγορίες εφαρμογών για κινητές συσκευές είναι αυτή που σχετίζεται άμεσα με τον εντοπισμό θέσης του χρήστη, εφόσον το Android υποστηρίζει GPS δέκτες. Για να δοθούν αυτές οι δυνατότητες στις εφαρμογές χρησιμοποιούνται οι κλάσεις του πακέτου *android.location* σε συνδυασμό με την προγραμματιστική διεπαφή εφαρμογής Google Maps Android. Το κεντρικό συστατικό του πλαισίου τοποθεσίας είναι η υπηρεσία συστήματος LocationManager η οποία προσφέρει πρόσβαση στις υπηρεσίες εντοπισμού της κινητής συσκευής. Όπως και με άλλες υπηρεσίες συστήματος δεν χρειάζεται να δημιουργηθεί άμεσα μια κλάση LocationManager, αλλά αιτείται ένα στιγμιότυπο από το σύστημα καλώντας τη μέθοδο *getSystemService(Context.LOCATION_SERVICE)*. Η μέθοδος αυτή επιστρέφει ένα handle σε ένα καινούριο στιγμιότυπο της κλάσης LocationManager. Όταν μια εφαρμογή αποκτήσει έναν LocationManager θα έχει την

ικανότητα να ζητήσει μια λίστα των παρόχων τοποθεσίας για τις πιο πρόσφατες θέσεις του χρήστη. Επιπλέον, η εφαρμογή θα έχει τη δυνατότητα να κάνει εγγραφή ή διαγραφή περιοδικών ενημερώσεων της τρέχουσας θέσης από έναν πάροχο τοποθεσίας. Παρόλο που η χρήση GPS είναι πιο ακριβής, λειτουργεί μόνο σε εξωτερικό περιβάλλον, ενώ καταναλώνει πολύ ισχύ και δεν γνωστοποιείται η θέση του χρήστη πολύ γρήγορα. Το δίκτυο παρόχων τοποθεσίας του Android αποφασίζει για την θέση του χρήστη χρησιμοποιώντας σταθμούς βάσης καθώς και σήματα Wi-Fi, έχοντας έτσι γρηγορότερη ανταπόκριση, λιγότερη κατανάλωση μπαταρίας ενώ η λειτουργία τους περιλαμβάνει τόσο εσωτερικούς όσο και εξωτερικούς χώρους. Με την κλήση της μεθόδου *requestLocationUpdates()* ο προγραμματιστής δηλώνει ότι επιθυμεί τη λήψη ενημερώσεων τοποθεσίας από τον LocationManager. Μια από τις παραμέτρους της μεθόδου αυτής είναι ο LocationListener. Ο LocationListener της εφαρμογής θα πρέπει να υλοποιεί διάφορες μεθόδους τις οποίες θα μπορεί να καλέσει ο LocationManager σε περίπτωση που ο χρήστης αλλάξει θέση ή όταν η κατάσταση της υπηρεσίας αλλάξει. Η πρώτη παράμετρος της μεθόδου requestLocationUpdates() είναι ο τύπος του παρόχου τοποθεσίας που θα χρησιμοποιηθεί, ενώ δίνεται η δυνατότητα μέσω της δεύτερης και τρίτης παραμέτρου της μεθόδου να ελέγχεται η συχνότητα με την οποία ο listener θα λαμβάνει ενημερώσεις. Θα πρέπει να αναφερθεί ότι στην περίπτωση που ο προγραμματιστής επιλέξει η ενημέρωση θέσης να γίνεται μέσω του GPS θα χρησιμοποιηθεί το όρισμα GPS_PROVIDER σαν όρισμα της μεθόδου requestLocationUpdates(), ενώ εάν επιλέξει η ενημέρωση θέσης να γίνεται μέσω του δικτύου θα χρησιμοποιηθεί το όρισμα NETWORK_PROVIDER.

Για να είναι εφικτή η λήψη ενημερώσεων θέσης είτε από το δίκτυο είτε από το GPS σύστημα θα πρέπει να ληφθούν κάποιες άδειες οι οποίες δηλώνονται στο αρχείο manifest. Οι άδειες αυτές θα πρέπει να είναι είτε η ACCESS_COARSE_LOCATION είτε η ACCESS_FINE_LOCATION.

Παροχή εικονικών δεδομένων θέσης

Με τη βοήθεια του εξομοιωτή Android ο προγραμματιστής μπορεί να ελέγξει τη λειτουργία μιας εφαρμογής η οποία περιλαμβάνει τον εντοπισμό θέσης της κινητής συσκευής. Οι τρόποι με τους οποίους μπορούν να αποσταλούν εικονικά δεδομένα τοποθεσίας στην εφαρμογή είναι τρεις:

- *Χρησιμοποιώντας το Eclipse*
  Από το περιβάλλον του Eclipse επιλέγεται από το μενού η καρτέλα Window → Show View → Other → Emulator Control. Στην καρτέλα αυτή μπορεί ο προγραμματιστής να εισάγει στο πεδίο Location Controls, τις γεωγραφικές

συντεταγμένες, να εισάγει αρχείο GPX για αναπαραγωγή διαδρομής ή μέσω αρχείου KML για πολλαπλά σημεία τοποθεσίας.

- *Μέσω του DDMS (Dalvik Debug Monitor Server*)
  Χρησιμοποιώντας το εργαλείο DDMS δίνεται η δυνατότητα στον προγραμματιστή να προσομοιώσει τα δεδομένα θέσης είτε αποστέλλοντας ατομικές συντεταγμένες στην συσκευή, είτε μέσω αρχείου GPX περιγράφοντας μια διαδρομή προκειμένου να την αναπαραγάγει η συσκευή είτε με τη βοήθεια αρχείου KML περιγράφοντας μεμονωμένα σημεία με σκοπό την αναπαραγωγή της αλληλουχίας τους στην συσκευή.

- *Με την εντολή geo στην κονσόλα του εξομοιωτή*

Βασική Λειτουργία

Ο βασικός άξονας της εφαρμογής είναι η παροχή συγκεκριμένων υπηρεσιών μέσω της χρησιμοποίησης χαρτών Google σε κινητή συσκευή. Οι λειτουργίες που παρέχονται είναι οι εξής:

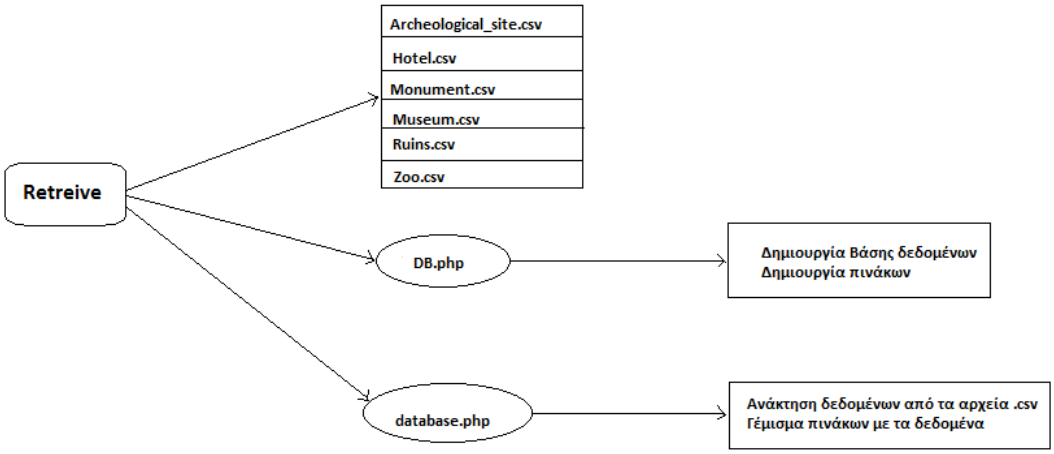
- Δημιουργία βάσης δεδομένων αποτελούμενη από 6 πίνακες, κάθε ένας αναφερόμενος σε μια κατηγορία από σημεία ενδιαφέροντος που θα έχει τη δυνατότητα να επιλέξει ο χρήστης μέσω ενός μενού
- Εντοπισμός τρέχουσας θέση του χρήστη με την αποστολή των γεωγραφικών συντεταγμένων (Latitude, Longtitude) μέσω του DDMS perspective
- Δυνατότητα μεγέθυνσης και σμίκρυνσης του χάρτη
- Δυνατότητα αλλαγής της προβολής του χάρτη σε δορυφορική όψη και επαναφορά στην κανονική προβολή
- Μενού επιλογής 6 κατηγοριών από σημεία ενδιαφέροντος στην περιοχή της Ισπανίας
- Επιλογή μιας ή περισσότερων κατηγοριών με αποτέλεσμα την εμφάνιση πινεζών διαφορετικού χρώματος για την κάθε μία πάνω στον χάρτη
- Δυνατότητα αφαίρεσης μιας ή περισσοτέρων κατηγοριών πινεζών από τον χάρτη
- Εμφάνιση επιπλέον σχετικής πληροφορίας με το «πάτημα» μίας πινέζας

Η εφαρμογή αποτελείται από τρεις βασικούς φακέλους.

1. Φάκελος Retrieve
2. Φάκελος php
3. Φάκελος AndroidGoogleMapsProject

Φάκελος Retrieve

Ένα βασικό μέρος της εφαρμογής είναι αυτό της δημιουργίας μιας βάσης δεδομένων προκειμένου να καταχωρούνται οι γεωγραφικές συντεταγμένες και η ονομασία των σημείων ενδιαφέροντος ξεχωριστά για κάθε κατηγορία. Η γραφική απεικόνιση σε υψηλό επίπεδο φαίνεται στην παρακάτω εικόνα:



Σχήμα 19: Αρχιτεκτονική δημιουργίας βάσης δεδομένων

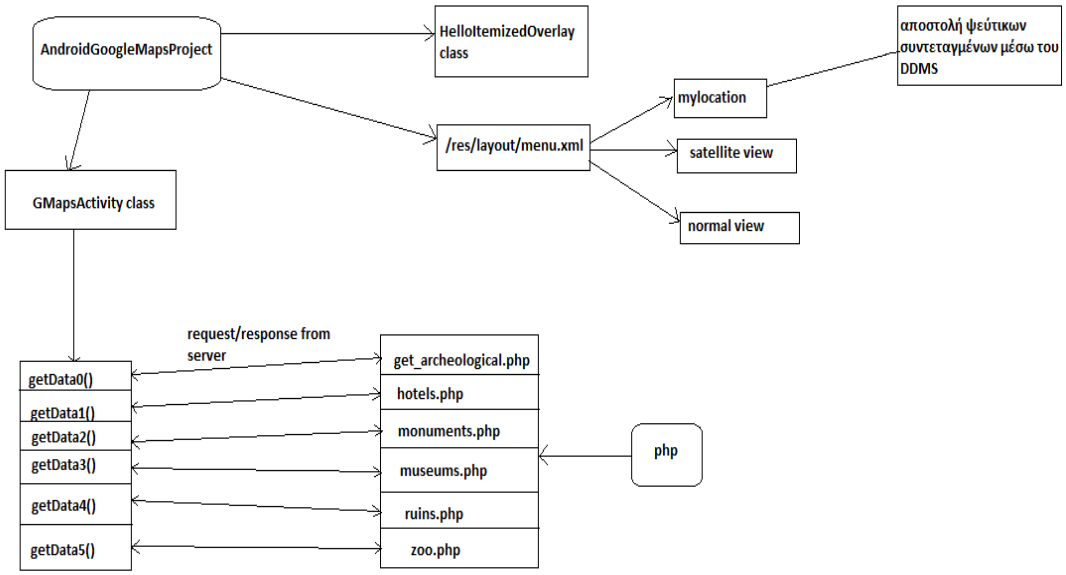
Η βάση δεδομένων καθώς και οι 6 πίνακες δημιουργούνται με την εκτέλεση του αρχείου DB.php. Έχοντας εκτελέσει το αρχείο DB.php έχουν δημιουργηθεί τόσο η βάση δεδομένων με όνομα *data* όσο και οι έξι πίνακες με ονόματα *archaeological_site, hotels, monuments, museums, ruins, zoos*. Οι πληροφορίες των σημείων αρχικά βρίσκονται σε έξι αρχεία τύπου csv. Με την εκτέλεση του αρχείου database.php τα αρχεία csv ανοίγονται και διαβάζονται προκειμένου να εισαχθούν τα δεδομένα τους στα κατάλληλα πεδία των πινάκων.

Η δομή της βάσης δεδομένων φαίνεται στην παρακάτω εικόνα:

Σχήμα 20: Βάση δεδομένων και πίνακες

Φάκελοι AndroidGoogleMapsProject και php

Ο φάκελος του project AndroidGoogleMapsProject περιλαμβάνει την κύρια εφαρμογή με τη βασική κλάση GmapsActivity. Η λειτουργία της εφαρμογής συνδέεται με τα αρχεία του φακέλου php από την πλευρά του τοπικού server ο οποίος χρησιμοποιείται για τη συγκεκριμένη εφαρμογή. Η αρχιτεκτονική με την οποία συνδέονται οι οντότητες αυτές μεταξύ τους φαίνεται στην παρακάτω εικόνα:



Σχήμα 21: Αρχιτεκτονική εφαρμογής

6.2 Απαιτήσεις της εφαρμογής

Για την ανάπτυξη της εφαρμογής GMapsActivity χρησιμοποιήθηκαν ορισμένα περιβάλλοντα υλοποίησης:

- Windows 7 Professional Sp1
- Ενσωματωμένο περιβάλλον ανάπτυξης Java Eclipse , Helios 2
- Sun Java SE Development Kit (JDK) 5.2
- Android ADT Plug-in για Eclipse 21.1.0
- Android SDK tools revision 21.1
- Google API έκδοση 16
- Keytool plug-in για το Eclipse έκδοση 1.4.2
- Android emulator WQVGA432 Platform 4.1.2
- Wamp Server Version 2.0

6.3 Σχεδιασμός και ανάπτυξη της εφαρμογής

Στην ενότητα που ακολουθεί θα παρουσιαστούν και θα αναλυθούν τα βασικότερα τμήματα κώδικα τα οποία αναπτύχθηκαν για να εξυπηρετηθούν οι λειτουργίες της εφαρμογής όπως αναφέρθηκαν παραπάνω.

6.3.1 Βάση δεδομένων και σημεία ενδιαφέροντος

Πριν το ξεκίνημα της ανάπτυξης της εφαρμογής Android GMapsActivity είναι απαραίτητη η δημιουργία μιας βάσης δεδομένων, η οποία θα αποτελείται από έξι πίνακες. Αυτό, όπως αναφέρθηκε, επιτυγχάνεται εκτελώντας το αρχείο DB.php το οποίο βρίσκεται στον φάκελο Retrieve. Απαραίτητη προϋπόθεση είναι η ύπαρξη ενός τοπικού εξυπηρετητή που στη συγκεκριμένη εργασία είναι ο Wamp Server και μέσω του phpmyAdmin θα μπορεί να γίνεται η διαχείριση της βάσης δεδομένων. Αρχικά δημιουργείται μια σύνδεση με τον mysql server μέσω της εντολής *mysql_connect()*.

```
$con = mysql_connect("localhost","root");
if (!$con)
{
      die('Could not connect: ' . mysql_error());
}
```

ενώ με διαδοχικά queries δημιουργείται η βάση δεδομένων.

```
mysql_query("CREATE DATABASE data",$con);
```

Για τον κάθε πίνακα εκτελείται ένα ξεχωριστό query με τη βοήθεια του οποίου δημιουργείται ο πίνακας και τα πεδία που τον αποτελούν. Για τον πρώτο πίνακα ο κώδικας είναι ο εξής:

```
mysql_query("CREATE TABLE Archaeological_site(
            longtitude varchar(25),
            latitude varchar(25),
            archeol_name varchar(40))")
            or die(mysql_error());
```

Με παρόμοιο τρόπο δημιουργούνται και οι υπόλοιποι πέντε πίνακες.

Στη συνέχεια θα πρέπει να εκτελεστεί το αρχείο *database.php* , το οποίο βρίσκεται επίσης στον φάκελο Retrieve. Με το αρχείο αυτό κάνουμε εισαγωγή των δεδομένων που βρίσκονται στα αρχεία .csv στα κατάλληλα πεδία των πινάκων. Τα αρχεία που

περιέχουν την πληροφορία των σημείων ενδιαφέροντος αποτελούνται από τρεις στήλες. Στην πρώτη βρίσκονται τα Longtitudes των σημείων, στην δεύτερη στήλη βρίσκονται τα Latitudes και στην τρίτη στήλη τα ονόματά τους. Αρχικά, αφού γίνει η σύνδεση με τον τοπικό εξυπηρετητή και επιλεγεί η βάση δεδομένων *data*, ανοίγονται ένα προς ένα τα αρχεία .csv προκειμένου να διαβαστεί η πληροφορία που διαθέτουν:

```
$con = mysql_connect("localhost","root");
if (!$con)
{
      die('Could not connect: ' . mysql_error());
}

mysql_select_db("data", $con) or die(mysql_error());
$fname1 = "Monument.csv";
$fname2 = "Archaeological_site.csv";
$fname3 = "Hotel.csv";
$file1= fopen("Monument.csv", "r");
$file2= fopen("Archaeological_site.csv", "r");
$file3= fopen("Hotel.csv", "r");
$file4= fopen("Zoo.csv", "r");
$file5= fopen("Museum.csv", "r");
$file6= fopen("Ruins.csv", "r");
```

Στη συνέχεια, με τη βοήθεια της συνάρτησης fgetcsv προσθέτουμε το πρώτο στοιχείο της κάθε εγγραφής του αρχείου στο πρώτο πεδίο των πινάκων(longtitude), το δεύτερο στοιχείο στο πεδίο latitude και το τρίτο στο πεδίο ονόματος της κάθε κατηγορίας. Τα στοιχείο στις εγγραφές των csv αρχείων χωρίζονται με τον χαρακτήρα “;”.

```
while (($info1 = fgetcsv($file1, 1000, ";")) !==false)
{
     for($i=0;$i<=2;$i++){$info1[$i] = str_replace(",", ".", $info1[$i]);}

            $att01 = mysql_real_escape_string($info1[0]);
            $att11 = mysql_real_escape_string($info1[1]);
            $att21 = mysql_real_escape_string($info1[2]);
            $sql = "INSERT INTO `monuments`
(`longtitude`,`latitude`,`mon_name`)VALUES ('$att01','$att11','$att21')";
            mysql_query($sql) or die(mysql_error());

}
fclose($file1);
```

Έτσι, οι πίνακες της κάθε κατηγορίας περιέχουν τις απαραίτητες πληροφορίες για τα σημεία ενδιαφέροντος.

6.3.2 Εμφάνιση Χάρτη

Χρησιμοποιώντας το περιβάλλον προγραμματισμού Eclipse Helios και έχοντας κάνει όλα τα απαραίτητα βήματα για να ξεκινήσει ο προγραμματισμός μιας εφαρμογής Android, δημιουργούμε έναν νέο φάκελο Android Application, που θα είναι ο φάκελος του Android project. Το επόμενο βήμα που θα πρέπει να γίνει είναι η εμφάνιση του Google χάρτη μέσω της εφαρμογής. Αυτό θα επιτευχθεί μετά από μια σειρά βημάτων που παρουσιάζονται στη συνέχεια.

Απαραίτητη προϋπόθεση είναι η εγκατάσταση της βιβλιοθήκης Google Maps στο περιβάλλον SDK. Η βιβλιοθήκη Maps εμπεριέχεται στο πρόσθετο Google APIs, το οποίο μπορεί να εγκατασταθεί χρησιμοποιώντας τους διαχειριστές SDK και AVD. Μετά την εγκατάσταση του πρόσθετου Google APIs μέσω του SDK, κατά τη δημιουργία του Android project θα πρέπει στις ιδιότητές του να τεθεί σαν build target το «Google APIs by Google Inc». Επιπλέον, ένα σημαντικό στοιχείο είναι ότι θα πρέπει να δημιουργηθεί μια AVD (Android Virtual Device) συσκευή και να τεθεί ως build target το Google APIs που έχουμε θέσει κατά τη δημιουργία της εφαρμογής. Ύστερα από την δημιουργία της νέας Android εφαρμογής, επειδή η βιβλιοθήκη Maps δεν αποτελεί μέρος της βιβλιοθήκης Android, θα πρέπει να δηλωθεί στο αρχείο manifest. Στο αρχείο AndroidManifest.xml θα πρέπει να προστεθεί σαν παιδί του στοιχείου <application> η δήλωση της βιβλιοθήκης:

```
<application
        android:icon="@drawable/ic_launcher1"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:allowBackup="true">
        <uses-library android:name="com.google.android.maps" />
```

Επιπλέον, η εφαρμογή θα πρέπει να αιτηθεί πρόσβαση στο internet προκειμένου να ανακτά τα επιμέρους τμήματα του χάρτη. Η αίτηση για αυτή την άδεια γίνεται επίσης στο αρχείο AndroidManifest.xml όπου προστίθεται σαν παιδί του στοιχείου <manifest>:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javacodegeeks.android.googlemaps"
    android:versionCode="1"
    android:versionName="1.0" >
    ……..
    <uses-permission android:name="android.permission.INTERNET"/>
```

Το αρχείο που βρίσκεται στη διαδρομή /res/layout/activity_gmaps.xml είναι ουσιαστικά το αρχείο που μας δείχνει τα στοιχεία τα οποία θα εμφανίζονται στην

κεντρική οθόνη της εφαρμογής. Για να γίνει δυνατή η εμφάνιση του Google χάρτη θα πρέπει ανοίγοντας το αρχείο αυτό να προστεθεί ένα στοιχείο όψης MapView.

```
<com.google.android.maps.MapView
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/map_view"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_alignParentLeft="true"
      android:layout_alignParentTop="true"
      android:apiKey="0YOVS_yaeKlDIDvv4pHuHW_s_2ygmA8VbMWj1Zg"
      android:clickable="true"
      android:enabled="true" >
  </com.google.android.maps.MapView>
```

Με το χαρακτηριστικό *android:clickable* προσδιορίζεται εάν επιθυμείται αλληλεπίδραση των χρηστών με τον χάρτη. Στην περίπτωση που το χαρακτηριστικό αυτό τεθεί στην τιμή false, τότε οποιαδήποτε επαφή με τον χάρτη δεν επιφέρει κανένα αποτέλεσμα. Ένα βασικό χαρακτηριστικό που θα πρέπει να υπάρχει είναι το android:api στο οποίο θα πρέπει ο προγραμματιστής να εισάγει το κλειδί του Google maps android api για την εφαρμογή. Με αυτό, αποδεικνύεται ότι η εφαρμογή και το πιστοποιητικό υπογραφής βρίσκονται εγγεγραμμένα στην υπηρεσία χαρτών.

Για να ολοκληρωθούν τα βήματα εμφάνισης του χάρτη θα πρέπει να γίνουν κάποιες προσθήκες και στο αρχείο java της κεντρικής δραστηριότητας. Το αρχείο αυτό στην παρούσα εφαρμογή ονομάζεται GmapsActivity.java. Η δραστηριότητα αυτή θα πρέπει να επεκτείνει μια υποκλάση της κλάσης Activity που ονομάζεται MapActivity και παρέχεται από τη βιβλιοθήκη Maps. Με την αλλαγή αυτή, η κεντρική δραστηριότητα της εφαρμογής κληρονομεί πολλές βασικές δυνατότητες χαρτών.

```
public class GmapsActivity extends MapActivity
```

Μέσα σε κάθε MapActivity απαιτείται η δήλωση της μεθόδου isRouteDisplayed() , για τη διενέργεια ορισμένων υπολογισμών από την υπηρεσία χαρτών. Το @Override είναι μια δήλωση στη Java η οποία υποδηλώνει ότι η μέθοδος που βρίσκεται από κάτω της υπερισχύει των μεθόδων της υπερκλάσης στην οποία ανήκει.

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

Η μέθοδος onCreate() η οποία υλοποιείται στο αρχείο GmapsActivity.java είναι η μέθοδος που καλείται όταν ξεκινάει η εφαρμογή και γίνονται όλες οι αρχικοποιήσεις που απαιτούνται. Μέσα σε αυτή τη μέθοδο θα πρέπει να δηλωθεί η φόρτωση του αρχείου activity_gmaps.xml:

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

Τέλος, για την προσθήκη δυνατοτήτων εστίασης στον χάρτη παρέχεται από την κλάση MapView ένα εργαλείο το οποίο μπορεί να κληθεί από τη μέθοδο onCreate() ως εξής:

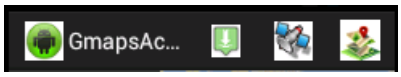
```
mapView = (MapView) findViewById(R.id.map_view);
mapView.setBuiltInZoomControls(true);
```

6.3.3 Τρέχουσα θέση, *δορυφορική και κανονική προβολή χάρτη*

Στην συνέχεια, θα αναλύσουμε τις τρεις επιλογές που έχει ο χρήστης της εφαρμογής μέσω κατάλληλου μενού που αναπτύχθηκε καθώς και τον τρόπο υλοποίησής τους.

Αρχικά, με τη δημιουργία ενός νέου αρχείου xml θα δημιουργήσουμε τα 3 κουμπιά επιλογής έχοντας προσθέσει στον φάκελο /res/drawable τρία αρχεία εικόνων που η κάθε μία υποδηλώνει την εκάστοτε λειτουργία. Το αρχείο menu.xml υλοποιεί το μενού επιλογών και βρίσκεται στον φάκελο /res/layout. Με τις ετικέτες <item> δημιουργούνται τρία στοιχεία MenuItem και καθένα αναπαριστά τα ξεχωριστά αντικείμενα του μενού. Τα χαρακτηριστικά που διαθέτουν είναι ένα αναγνωριστικό id, το εικονίδιο που θα εμφανίζεται, και ο τίτλος τους.

```
<?xml version="1.0" encoding="UTF-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item android:id= "@+id/my_location"
        android:icon="@drawable/mylocation"
        android:title="My Location"
        android:showAsAction="always" />

    <item android:id= "@+id/mapview_satellite"
        android:icon="@drawable/satelliteview"
        android:title="Satellite View"
        android:showAsAction="always"/>

    <item android:id= "@+id/mapview_norm"
        android:icon="@drawable/normalview"
        android:title="Normal View"
         android:showAsAction="always"/>

 </menu>
```



Σχήμα 22: Μενού επιλογών

Προκειμένου να προσδιορίσουμε το μενού επιλογών στη δραστηριότητα της εφαρμογής θα πρέπει να κληθεί η μέθοδος onCreateOptionsMenu().

```
@Override
 public boolean onCreateOptionsMenu(Menu menu)
 {
 MenuInflater menuInflater = getMenuInflater();
 menuInflater.inflate(R.layout.menu, menu);
 return true;
 }
```

Για να μπορέσουμε να διαχειριστούμε ξεχωριστά κάθε στοιχείο του μενού θα πρέπει να κληθεί η μέθοδος onOptionsItemSelected(). Ο διαχωρισμός των κουμπιών γίνεται με βάση το id τους, τα οποία έχουν προσδιοριστεί στο αρχείο menu.xml. Έτσι, γίνονται διαφορετικές ενέργειες στα εκάστοτε click events.

```
@Override
 public boolean onOptionsItemSelected(MenuItem item)
 {
    switch (item.getItemId())
     {
      case R.id.my_location:
      // Single menu item is selected do something
      locLstnr.gpsCurrentLocation();
      return true;

     case R.id.mapview_norm:    //normal view
    Toast.makeText(GmapsActivity.this, "Map Normal Street View",
Toast.LENGTH_SHORT).show();
     if(mapView.isSatellite()==true){
     mapView.setSatellite(false);
 }
 return true;

 case R.id.mapview_satellite:      //Satellite view
 Toast.makeText(GmapsActivity.this, "Map Satellite View",
Toast.LENGTH_SHORT).show();
 if(mapView.isSatellite()==false){
 mapView.setSatellite(true);
 }
 return true;

 default:
 return super.onOptionsItemSelected(item);
 }//switch
 }//onoptionsitemselected
```

Αναφορικά με τη λειτουργία του πρώτου κουμπιού εντοπισμού θέσης, θα πρέπει να αναφερθεί ότι η αποστολή των συντεταγμένων γίνεται μέσω του εργαλείου DDMS. Στο συγκεκριμένο σημείο της τρέχουσας θέσης της κινητής συσκευής θα τοποθετείται μια πινέζα (mark.png), το εικονίδιο της οποίας βρίσκεται στον φάκελο /res/drawable. Αρχικά, στο αρχείο AndroidManifest.xml θα πρέπει να δηλωθεί η αίτηση δύο αδειών για να είναι δυνατή η λήψη ενημερώσεων θέσης είτε από το GPS είτε από το ασύρματο δίκτυο ή τους σταθμούς βάσης.

```
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
     <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

Κατά τη δημιουργία της εικονικής συσκευής AVD θα πρέπει στον εξοπλισμό hardware της συσκευής να υπάρχει υποστήριξη GPS. Το Android λειτουργικό σύστημα περιέχει το πακέτο android.location το οποίο παρέχει την προγραμματιστική διεπαφή εφαρμογής προκειμένου να καθοριστεί η τρέχουσα γεωγραφική θέση. Η κλάση LocationManager παρέχει πρόσβαση στην υπηρεσία location του Android.

Στο αρχείο GmapsActivity.java εισάγουμε τις εξής κλάσεις:

```
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
```

Στη συνέχεια, στην κεντρική κλάση GmapsActivity.java θα πρέπει να δημιουργήσουμε δύο βασικά αντικείμενα: ένα LocationManager αντικείμενο και ένα LocationListener αντικείμενο.

```
LocationManager locMgr;        //location manager
MyLocationListener locLstnr;    //location listener
```

Επίσης, δημιουργούμε μια καινούρια κλάση, την *MyLocationListener,* μέσα στην κεντρική κλάση GmapsActivity, στην οποία δηλώνονται οι μέθοδοι οι οποίες σχετίζονται με τον LocationListener της εφαρμογής για το συγκεκριμένο κουμπί. Η κλάση MyLocationListener είναι:

```
public class MyLocationListener implements LocationListener
 {
    public void onLocationChanged(Location loc)
    {
       loc.getLatitude();
       loc.getLongitude();
       tvlat.setText(""+loc.getLatitude());
       tvlong.setText(""+loc.getLongitude());

       this.gpsCurrentLocation();
     }

    public void gpsCurrentLocation()
     {
       if(marker!=null){
       listOfOverLays.remove(marker);
       }

       listOfOverLays = mapView.getOverlays();

       String coordinates[] = {""+tvlat.getText(),
""+tvlong.getText()};
       double lat = Double.parseDouble(coordinates[0]);
       double lng = Double.parseDouble(coordinates[1]);
       GeoPoint p = new GeoPoint(
                   (int) (lat * 1E6),
                   (int) (lng * 1E6));
       marker = new MyMapOverlays(p);
       listOfOverLays.add(marker);
       mapView.invalidate();
       mc.animateTo(p);
       mc.setZoom(15);

     }
```

```
     public void onProviderDisabled(String provider)
     {Toast.makeText( getApplicationContext(),
       "Gps Disabled",
     Toast.LENGTH_SHORT ).show();
     }
     public void onProviderEnabled(String provider)
    {Toast.makeText( getApplicationContext(),
     "Gps Enabled",
     Toast.LENGTH_SHORT).show();
     }
      public void onStatusChanged(String provider, int status, Bundle
extras)
     {
     }
 }
```

Στη μέθοδο *onLocationChanged(),* με τη βοήθεια των μεθόδων getLatitude(), getLongtitude() λαμβάνονται σε βαθμούς τα latitude και longtitude του σημείου της κινητής συσκευής, τα οποία έχουν αποσταλεί από το DDMS και στη συνέχεια ανατίθενται σε δύο κρυφά textviews με ονόματα tvlat, tvlong. Στη συνέχεια, δηλώνεται η μέθοδος *gpsCurrentLocation()*. Στη μέθοδο αυτή καταχωρούνται τα latitude, longtitude σε έναν πίνακα coordinates[] και με τη χρησιμοποίηση της κλάσης *GeoPoint* κατασκευάζεται ένα γεωγραφικό σημείο σε micro degrees. Στη συνέχεια, δημιουργείται ένα καινούριο στιγμιότυπο της κλάσης *MyMapOverlays* με παράμετρο το γεωγραφικό σημείο το οποίο έχει ήδη δημιουργηθεί και προστίθεται στην λίστα από τα Overlays. Το αντικείμενο *mc* της κλάσης *mapcontroller* καλώντας την μέθοδο *animateTo(point)* μετατοπίζει τον χάρτη στο σημείο της παραμέτρου της. Επίσης, στην κλάση *MyLocationListener* βρίσκονται δύο μέθοδοι, η *onProviderEnabled()* , η οποία καλείται όταν ο πάροχος ενεργοποιηθεί από τον χρήστη και η *onProviderDisabled()* , η οποία καλείται όταν ο πάροχος απενεργοποιηθεί από τον χρήστη. Τέλος, η μέθοδος *onStatusChanged()* καλείται όταν αλλάξει η κατάσταση του παρόχου.

Εκτός από την κλάση *MyLocationListener* που περιγράφτηκε, θα χρειαστεί να δημιουργηθεί επιπλέον και η κλάση MyMapOverlays, η οποία επεκτείνει την κλάση *com.google.android.maps.Overlay* .Η κλάση αυτή είναι απαραίτητη από την στιγμή που επιθυμούμε να τοποθετηθεί στην τρέχουσα θέση του χρήστη ένας marker. Σε αυτή δηλώνονται όλες οι μέθοδοι που σχετίζονται με την τοποθέτηση του marker σε ένα συγκεκριμένο σημείο στον χάρτη ή με τη διαγραφή του από αυτόν.

Ο κώδικας της κλάσης αυτής φαίνεται παρακάτω:

```
public class MyMapOverlays extends com.google.android.maps.Overlay
 {
       private List<OverlayItem> mOverlays = new
ArrayList<OverlayItem>();
     GeoPoint location = null;
     public MyMapOverlays(GeoPoint location)
      {
       super();
       this.location = location;
      }

     public void clear() {
       // TODO Auto-generated method stub
        }
       public void remove(OverlayItem overlay) {
                mOverlays.remove(overlay);
          }
      @Override
      public void draw(Canvas canvas, MapView mapView, boolean
shadow)
      {
         super.draw(canvas, mapView, shadow);
          Point screenPoint = new Point();
          mapView.getProjection().toPixels(this.location,
screenPoint);

canvas.drawBitmap(BitmapFactory.decodeResource(getResources(),
R.drawable.mark),
          screenPoint.x, screenPoint.y , null);
     }
 }
```

Στην περίπτωση, λοιπόν, που επιλεγεί το κουμπί με id my_location, όπως είδαμε από τη μέθοδο ***onOptionsItemSelected(MenuItem item),*** μέσω του αντικειμένου ***locLstnr*** που δημιουργήσαμε καλείται η μέθοδος *gpsCurrentLocation()* της κλάσης *MyLocationListener.*

Όσον αφορά την χρήση των δύο άλλων κουμπιών ο χρήστης έχει τη δυνατότητα να αλλάζει τον τρόπο προβολής του χάρτη σε μορφή από δορυφόρο και επαναφορά σε κανονική. Αυτό επιτυγχάνεται με την κλήση της μεθόδου *isSatellite()* η οποία μεταβάλλεται αναλόγως με το κουμπί που επιλέγεται από τον χρήστη. Έτσι, εάν επιλεγεί το κουμπί με id mapview_satellite, τότε ελέγχεται εάν η επιστρεφόμενη τιμή της είναι false και αλλάζει σε true, ενώ στην επιλογή του κουμπιού κανονικής προβολής εάν η επιστρεφόμενη τιμή της μεθόδου isSatellite() είναι true αλλάζει σε false.

6.3.4 Μενού επιλογών για κατηγορίες POIs

Στην κεντρική οθόνη της εφαρμογής εκτός από τον χάρτη και το μενού επιλογών για την τρέχουσα θέση και την αλλαγή εμφάνισης του χάρτη υπάρχει και ένα ακόμα

κουμπί. Το κουμπί «Select POIs» δίνει τη δυνατότητα στον χρήστη να επιλέξει μία από τις πέντε κατηγορίες σημείων ενδιαφέροντος με τη βοήθεια μιας λίστας από checkboxes.

Τα χαρακτηριστικά του κουμπιού αυτού καθορίζονται στο αρχείο activity_gmaps.xml:

```
<Button
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:background="#525252"
      android:drawableLeft="@drawable/poi"
      android:text="@string/points"
      android:textColor="#98FB98"
      />
```

Για να μπορέσει να εμφανιστεί η λίστα επιλογών με το πάτημα του κουμπιού θα πρέπει να υπάρχει ένας *onClickListener* στο συγκεκριμένο view. Στην κεντρική κλάση της εφαρμογής *GmapsActivity.java*, και συγκεκριμένα στην μέθοδο *onCreate()* στο όνομα *selectpoiButton* , το οποίο έχει δηλωθεί στο τμήμα δηλώσεων της κλάσης, αντιστοιχίζεται στο κουμπί με id *button1* και στη συνέχεια ορίζεται ένας συγκεκριμένος onClickListener για το συγκεκριμένο κουμπί:

```
selectpoiButton = (Button) findViewById(R.id.button1);
selectpoiButton.setOnClickListener(new View.OnClickListener()
 {
      public void onClick(View view)
      {
         switch(view.getId())
          {
            case R.id.button1:
            showSelectpoisDialog();
            // TODO: Show the colours dialog
            break;
        default:
           break;
          }
     }
 });
```

Η μέθοδος *onClick* καλείται όταν ένα view έχει πατηθεί. Στο αρχείο GmapsActivity.java έχει ορισθεί ένας πίνακας τύπου CharSequence[] οποίος περιέχει τα ονόματα των κατηγοριών ενδιαφέροντος.

```
protected CharSequence[] pois = new CharSequence[] { "Archaeological
sites", "Hotels", "Monuments", "Museums", "Ruins", "Zoos" };
```

Ο πίνακας αυτός θα χρησιμοποιηθεί στην εμφάνιση του μενού.

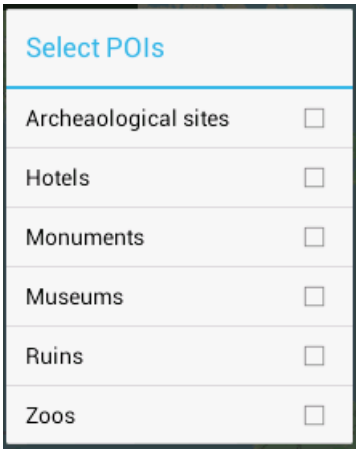
Στην περίπτωση που επιλεγεί το κουμπί *Select POIs* καλείται η μέθοδος *showSelectpoisDialog()*. Επειδή, το μενού επιλογών των checkboxes γίνεται με τη μορφή alert box χρησιμοποιείται ο παρακάτω κώδικας στο εσωτερικό της μεθόδου *showSelectpoisDialog():*

```
protected void showSelectpoisDialog() {
        final boolean[] checkedpois = new boolean[pois.length];
        final int count = pois.length;
        for(int i = 0; i < count; i++)
        checkedpois[i] = selectedpois.contains(pois[i]);

        DialogInterface.OnMultiChoiceClickListener poisDialogListener
= new DialogInterface.OnMultiChoiceClickListener()
          {
             public void onClick(DialogInterface dialog, int which,
boolean isChecked)
              {
            if(isChecked)
             {
                        selectedpois.add(pois[which]);
                   if(which==0){ //archeological sites selected
                         getData0();
                        progressDialog =
ProgressDialog.show(GmapsActivity.this, "", "Loading...");
                        new Thread()
                            {public void run()
                              {try
                                {
                                  sleep(1000);}
                               catch (Exception e) {
                                  Log.e("tag", e.getMessage());}
                            //dismiss the progress dialog
                               progressDialog.dismiss();}
                            }.start();

                          }
                       if(which==1){  // Hotels selected

                            getData1();


                            progressDialog =
ProgressDialog.show(GmapsActivity.this, "", "Loading...");
                        new Thread()
                            {public void run()
                              {try
                                {
                                  sleep(1000);}
                               catch (Exception e) {
                                  Log.e("tag", e.getMessage());}
                            //dismiss the progress dialog
                               progressDialog.dismiss();}
                            }.start();}
```

```
                            if(which==2){    //monuments selected

                                getData2();
                                progressDialog =
ProgressDialog.show(GmapsActivity.this, "", "Loading...");
                               new Thread()
                                   {public void run()
                                     {try
                                       {
                                         sleep(1000);}
                                      catch (Exception e) {
                                         Log.e("tag", e.getMessage());}
                                    //dismiss the progress dialog
                                       progressDialog.dismiss();}
                                   }.start();}
                               if(which==3){     //museums selected

                                         getData3();
                                         progressDialog =
ProgressDialog.show(GmapsActivity.this, "", "Loading...");
                                    new Thread()
                                         {public void run()
                                           {try
                                             {
                                               sleep(1000);}
                                            catch (Exception e) {
                                               Log.e("tag",
e.getMessage());}
                                         //dismiss the progress dialog
                                             progressDialog.dismiss();}
                                         }.start();}
                               if(which==4){     //Ruins selected
                                         getData4();
                                         progressDialog =
ProgressDialog.show(GmapsActivity.this, "", "Loading...");
                                    new Thread()
                                         {public void run()
                                           {try
                                             {
                                               sleep(1000);}
                                            catch (Exception e) {
                                               Log.e("tag",
e.getMessage());}
                                         //dismiss the progress dialog
                                             progressDialog.dismiss();}
                                         }.start();}
                               if(which==5){     //Zoos selected

                                         getData5();
                                         progressDialog =
ProgressDialog.show(GmapsActivity.this, "", "Loading...");
                                    new Thread()
                                         {public void run()
                                           {try
                                             {
                                               sleep(10000);}
                                            catch (Exception e) {
                                               Log.e("tag",
e.getMessage());}
                                         //dismiss the progress dialog
```

```
                                        progressDialog.dismiss();}
                                    }.start();}
              }//if
            else
                {
                selectedpois.remove(pois[which]);
                Toast.makeText(getBaseContext(), "UnChecked
"+pois[which],Toast.LENGTH_SHORT).show();
                if(which==0)
                rm0();
                if(which==1)
                      rm1();
                if(which==2)
                      rm2();
                if(which==3)
                      rm3();
                if(which==4)
                      rm4();
                if(which==5)
                      rm5();
                }//else
             onChangeSelectedpois();
            }//onClick
          };//poisDialogListener
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Select POIs");
        builder.setMultiChoiceItems(pois, checkedpois,
poisDialogListener);
        AlertDialog dialog = builder.create();
        dialog.show();
      }//showSelectpoisDialog
```

Εκτός από τον πίνακα *pois* χρησιμοποιούνται δύο βοηθητικοί πίνακες. Ο ένας είναι ο *selectedpois,* ένας δυναμικός πίνακας ο οποίος καταχωρεί τις επιλογές του χρήστη και ο λογικού τύπου πίνακας *checkedpois* , ο οποίος καταδεικνύει την ύπαρξη της επιλογής ή όχι του χρήστη για όλες τις κατηγορίες σημείων. Επιπλέον, δημιουργείται ένα καινούριο αντικείμενο της κλάσης *AlertDialog.Builder*, το οποίο θα αντιστοιχεί στο κουμπί *Select POIs*. Στη συνέχεια, με την κλήση της μεθόδου *setMultiChoiceItems* της κλάσης *AlertDialog.Builder*, τίθεται η λίστα των στοιχείων που επιθυμούμε να εμφανιστούν στο πλαίσιο διαλόγου σαν περιεχόμενο (τα περιεχόμενα του πίνακα pois), ενώ ταυτόχρονα συσχετίζεται η λίστα αυτή με τον *poisDialogListener* που έχει υλοποιηθεί παραπάνω. Με την κλήση της μεθόδου *builder.create*() γίνεται η δημιουργία ενός *AlertDialog* και εμφανίζεται το πλαίσιο διαλόγου καλώντας την μέθοδο *dialog.show*().



Σχήμα 23: Μενού επιλογών

Για να είναι δυνατή η εκτέλεση κάποιου τμήματος κώδικα όταν επιλέγονται στοιχεία από ένα πλαίσιο διαλόγου πολλαπλών επιλογών, χρησιμοποιείται η διεπαφή ***DialogInterface.OnMultiChoiceClickListener***. Στο εσωτερικό της μεθόδου *showSelectpoisDialog*(), δημιουργείται ένα αντικείμενο της κλάσης *DialogInterface.OnMultiChoiceClickListener* και συγκεκριμένα το ***poisDialogListener.*** Στο εσωτερικό της διεπαφής αυτής καλείται η μέθοδος onClick(), έτσι ώστε όταν επιλέγεται μια κατηγορία να εμφανίζονται οι markers στα κατάλληλα σημεία του χάρτη για τη συγκεκριμένη κατηγόρια, ενώ εάν αφαιρείται μια επιλογή να αφαιρούνται οι markers από τον χάρτη για τη συγκεκριμένη κατηγορία. Έτσι, μέσα στην μέθοδο onClick() με μια δομή επιλογής ελέγχεται η επιλογή ενός checkbox και στη συνέχεια, ανάλογα με την κατηγορία στην οποία ανήκει, καλούνται οι αντίστοιχες μέθοδοι. Όταν ο χρήστης επιλέξει μια ή περισσότερες κατηγόριες από το μενού καλούνται για κάθε επιλογή του πέντε διαφορετικές μέθοδοι: getData0(), getData1(), getData2(), getData3(), getData4(). Οι μέθοδοι αυτές είναι υπεύθυνες για την εμφάνιση των markers της κάθε κατηγόριας και είναι παρόμοιες μεταξύ τους. Η μέθοδος getData0() καλείται όταν ο χρήστης επιλέξει την κατηγορία Archaeological_sites, και είναι η παρακάτω:

```
public void getData0()
 {
      String result = "";
      InputStream isr = null;
      try{
         HttpClient httpclient = new DefaultHttpClient();
         HttpPost httppost = new
HttpPost("http://10.0.2.2/php/get_archeaological.php");
         HttpResponse response = httpclient.execute(httppost);
         HttpEntity entity = response.getEntity();
         isr = entity.getContent();
```

```
 }
 catch(Exception e){
         Log.e("log_tag", "Error in http connection "+e.toString());
 }
 try{    BufferedReader reader = new BufferedReader(new
InputStreamReader(isr,"iso-8859-1"),8);
         StringBuilder sb = new StringBuilder();
         String line = null;
         while ((line = reader.readLine()) != null) {
                 sb.append(line + "\n");
         }
         isr.close();
         result=sb.toString();
 }
 catch(Exception e){
         Log.e("log_tag", "Error  converting result "+e.toString());
 }

try {
      Drawable drawable0 =
this.getResources().getDrawable(R.drawable.marker_default);
      itemizedoverlay0 = new HelloItemizedOverlay(drawable0, this);
      JSONArray jArray = new JSONArray(result);
      mapOverlays = mapView.getOverlays();
      for(int i=0; i<jArray.length();i++){
            JSONObject json = jArray.getJSONObject(i);
            double d1 =json.getDouble("longtitude");
            double d2 =json.getDouble("latitude");
            String d3=json.getString("archeol_name");
            point = new GeoPoint((int) (d2 * 1E6),
                                 (int) (d1 * 1E6));
            overlayItem0= new OverlayItem((GeoPoint) point, d3, d3);

            itemizedoverlay0.addOverlay(overlayItem0);

         }
         mapOverlays.add(itemizedoverlay0);
         mapOverlays = mapView.getOverlays();
         mapView.invalidate();
       mapView.getController().animateTo(point);
       mc.setZoom(7);
}
catch (Exception e) {
      // TODO: handle exception
         Log.e("log_tag", "Error Parsing Data "+e.toString());
    }
}
```

Από το αρχείο αυτό αρχικά θα πρέπει να γίνει προσπέλαση του πίνακα archaeological_site από τη βάση δεδομένων που έχουμε δημιουργήσει, προκειμένου να μπορέσει να χρησιμοποιηθεί το περιεχόμενό του. Για το λόγο αυτό, δημιουργείται ένας καινούριος *httpClient* και εκτελείται μια αίτηση HTTP POST στη διεύθυνση του αρχείου ***get_archaeological.php.*** Το αρχείο php που υλοποιεί την πλευρά του εξυπηρετητή, αφού κάνει σύνδεση με τη βάση δεδομένων, επιλέγει από τον πίνακα archaeological_site όλα του τα δεδομένα τα οποία και καταχωρεί σε

κωδικοποίηση JSON. Η κωδικοποίηση JSON είναι μια αναπαράσταση κειμένου πολύπλοκων περιπτώσεων δεδομένων. Ένα παράδειγμα της μορφής κειμένου σε αυτή φαίνεται παρακάτω:

```
[{"longtitude":"-5.7546943","latitude":"43.4840812","archeol_name":"Archaeological site Villa romana de Vera"},{"longtitude":"-4.6894038","latitude":"43.393293","archeol_name":"Archaeological site Idolo de Pe\u00f1a Tu"},{"longtitude":"-4.8672684","latitude":"37.8864401","archeol_name":"Archaeological site Medinat al-Zahra"}]
```

Τα κωδικοποιημένα δεδομένα μετατρέπονται σε αλφαριθμητικά και δημιουργούνται τα γεωγραφικά σημεία που σχηματίζονται από τις συντεταγμένες Latitude και Longtitude του κάθε σημείου. Οι markers των γεωγραφικών σημείων κάθε κατηγορίας σχηματίζουν ένα διαφορετικό itemizedoverlay και με τη βοήθεια της μεθόδου add() προστίθενται στο χάρτη.

Μια βασική κλάση απαραίτητη για όλες τις λειτουργίες που μπορούν να γίνουν στα διάφορα Overlays του χάρτη είναι η HelloItemizedOverlay.java και βρίσκεται στον φάκελο /src. Το περιεχόμενο της κλάσης αυτής φαίνεται παρακάτω:

```
public class HelloItemizedOverlay extends
ItemizedOverlay<OverlayItem>
{

      GeoPoint geo;
      MapView mapView;
      Context mContext;
      private ArrayList<OverlayItem> mOverlays = new
ArrayList<OverlayItem>();


      public HelloItemizedOverlay(Drawable defaultMarker, Context
context) {
              super(boundCenterBottom(defaultMarker));

              mContext = context;
              populate();
            }

       public void removeOverlay(OverlayItem overlayItem0){
             mOverlays.remove(overlayItem0);
             setLastFocusedIndex(-1);
             populate();
      }

      public void clear(){
        mOverlays.clear();
        populate();
             }
```

```
      public void addOverlay(OverlayItem overlay){
          mOverlays.add(overlay);
          setLastFocusedIndex(-1);
          populate();
      }
      @Override
      protected OverlayItem createItem(int i) {
        return mOverlays.get(i);
      }

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


      @Override
    public void draw(Canvas canvas, MapView mapView,
                     boolean shadow) {
      super.draw(canvas, mapView, shadow);
    }

      @Override
      protected boolean onTap(int index) {
            OverlayItem item = mOverlays.get(index);
            mOverlays.get(index).getTitle();
            geo=item.getPoint();
            AlertDialog.Builder dialog = new
AlertDialog.Builder(mContext);
            dialog.setIcon(android.R.drawable.ic_dialog_info);
            dialog.setTitle(item.getTitle());

            dialog.setPositiveButton("OK", new
DialogInterface.OnClickListener() {

              public void onClick(DialogInterface dialog, int which)
{
                  // TODO Auto-generated method stub

              }
          });
            dialog.show();
            return true;
          }
}
```

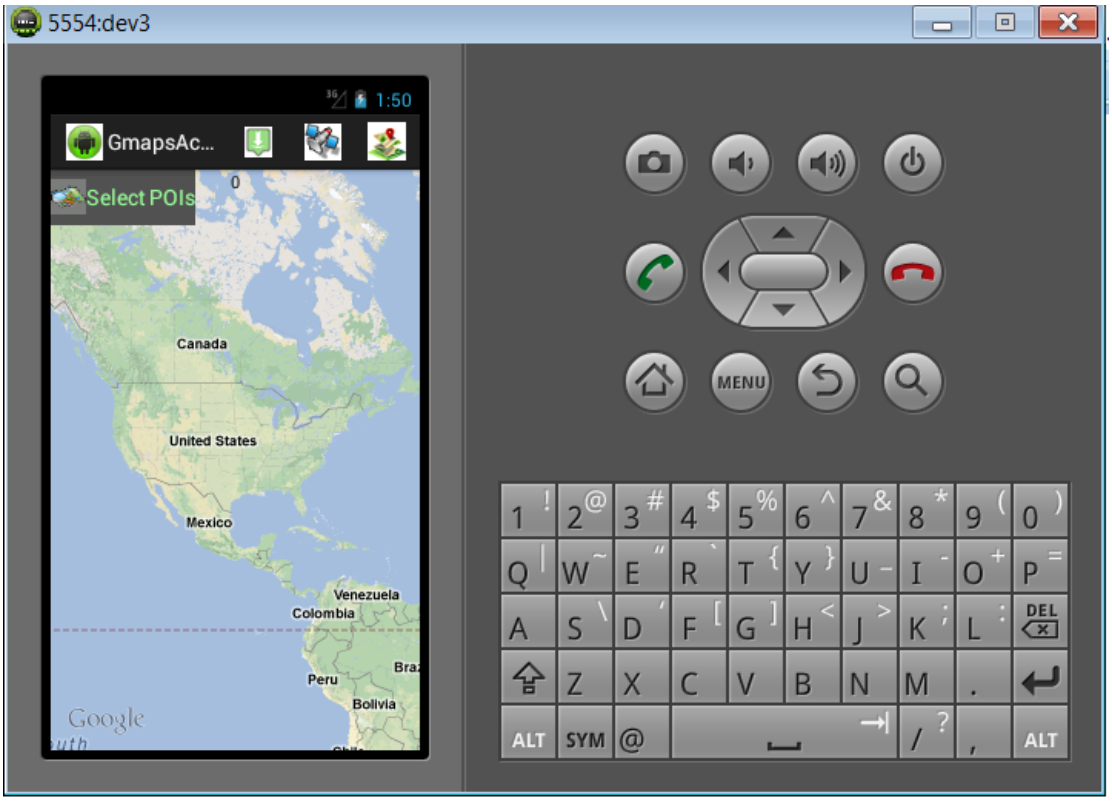
Στο εσωτερικό της δηλώνονται μέθοδοι που σχετίζονται με τη δημιουργία, τη διαγραφή και την σχεδίαση ενός OverlayItem καθώς και με την εμφάνιση ενός πλαισίου διαλόγου στην περίπτωση που ο χρήστης «πατήσει» επάνω σε έναν marker.

Κεφάλαιο 7

Σενάρια χρήσης και εκτέλεση της εφαρμογής

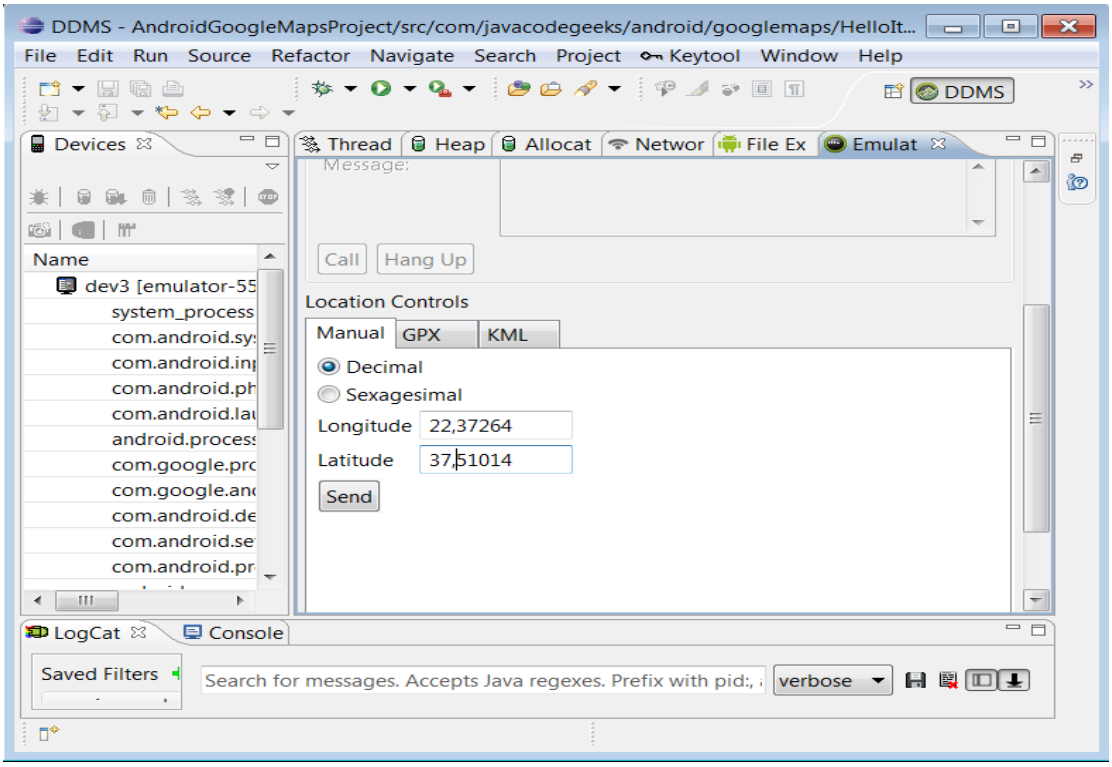
Σε αυτό το κεφάλαιο θα παρουσιαστεί ο τρόπος με τον οποίο μπορεί ένας χρήστης να αλληλεπιδράσει με την εφαρμογή καθώς και οι δυνατότητες που του δίνονται.

Η οθόνη που εμφανίζεται στον χρήστη κατά την έναρξη της εφαρμογής GmapsActivity είναι η παρακάτω:

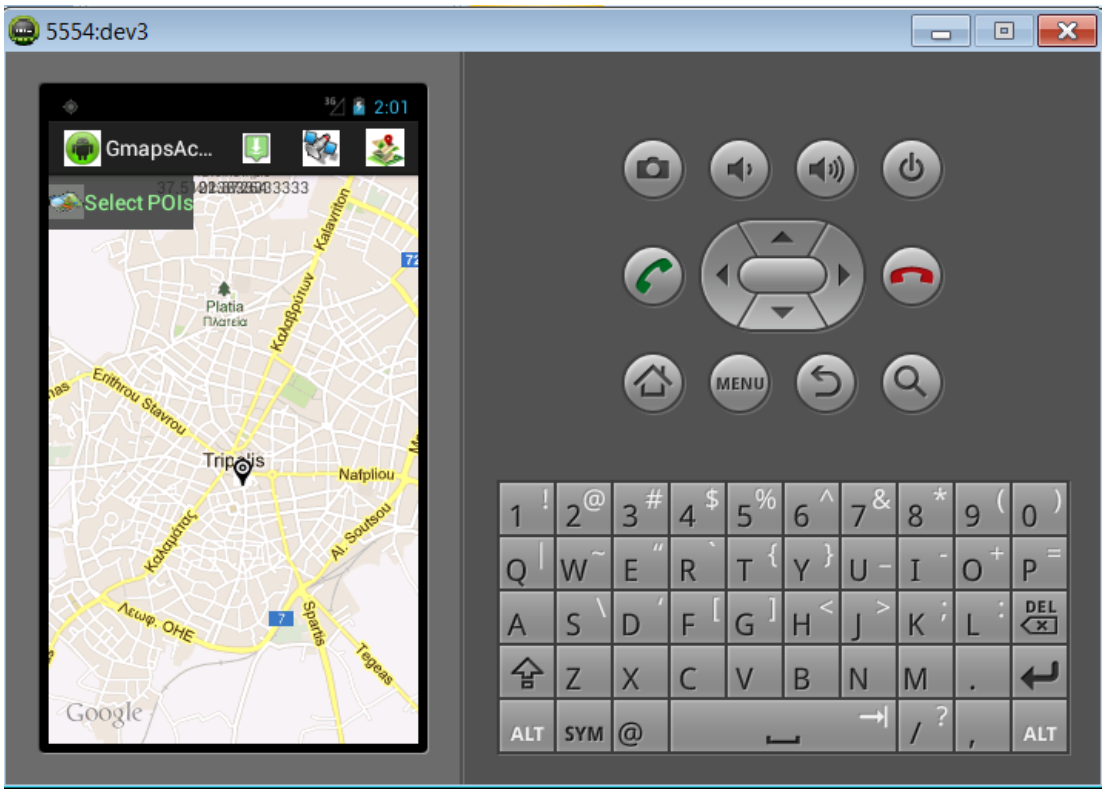


Αρχικά, υπάρχει η δυνατότητα αλλαγής της εμφάνισης του χάρτη σε προβολή από δορυφόρο, πατώντας το μεσαίο κουμπί από το μενού επιλογών στο επάνω μέρος της οθόνης:

Η επαναφορά σε κανονική εμφάνιση γίνεται πατώντας το δεξί κουμπί. Η λειτουργία του αριστερού κουμπιού είναι η προβολή του σημείου που βρίσκεται η κινητή συσκευή, αφού πρώτα έχουν σταλεί οι συντεταγμένες του μέσω του DDMS perspective. Υποθετικά αποστέλλονται οι συντεταγμένες για την Τρίπολη:

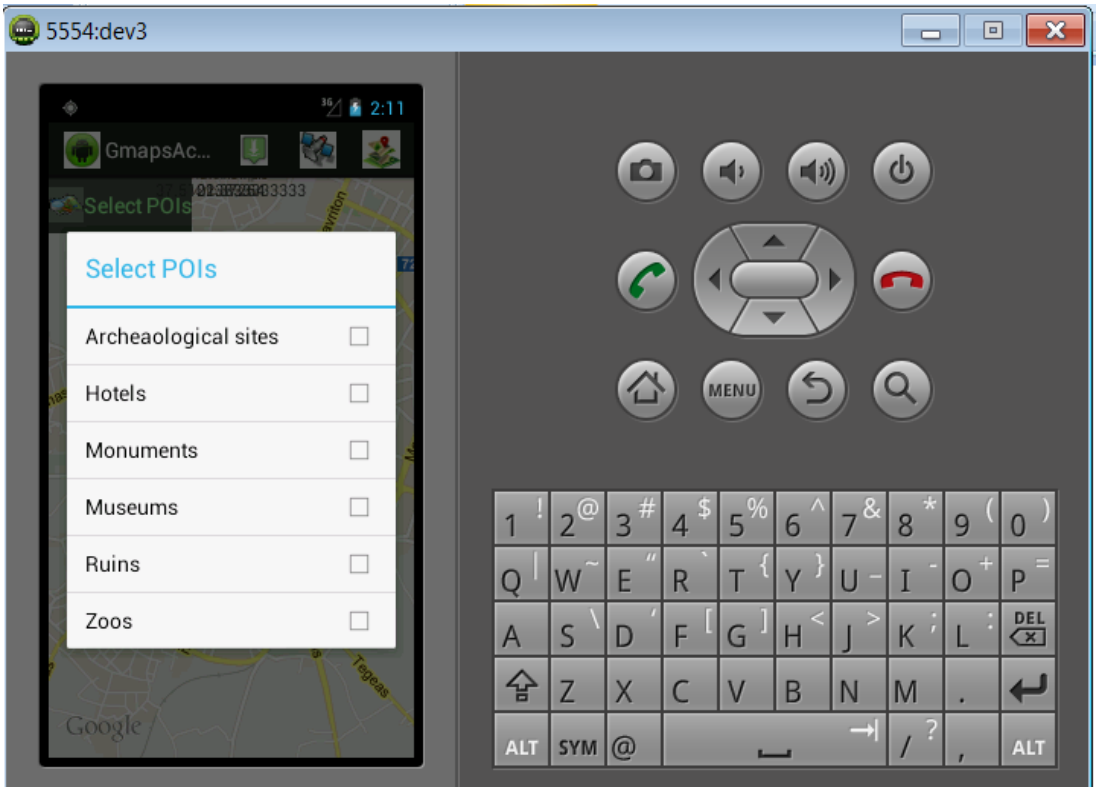


Το αποτέλεσμα με το πάτημα του κουμπιού Send είναι η μεταφορά του χάρτη στην τοποθεσία της Τρίπολης και η τοποθέτηση ενός marker σε αυτή:



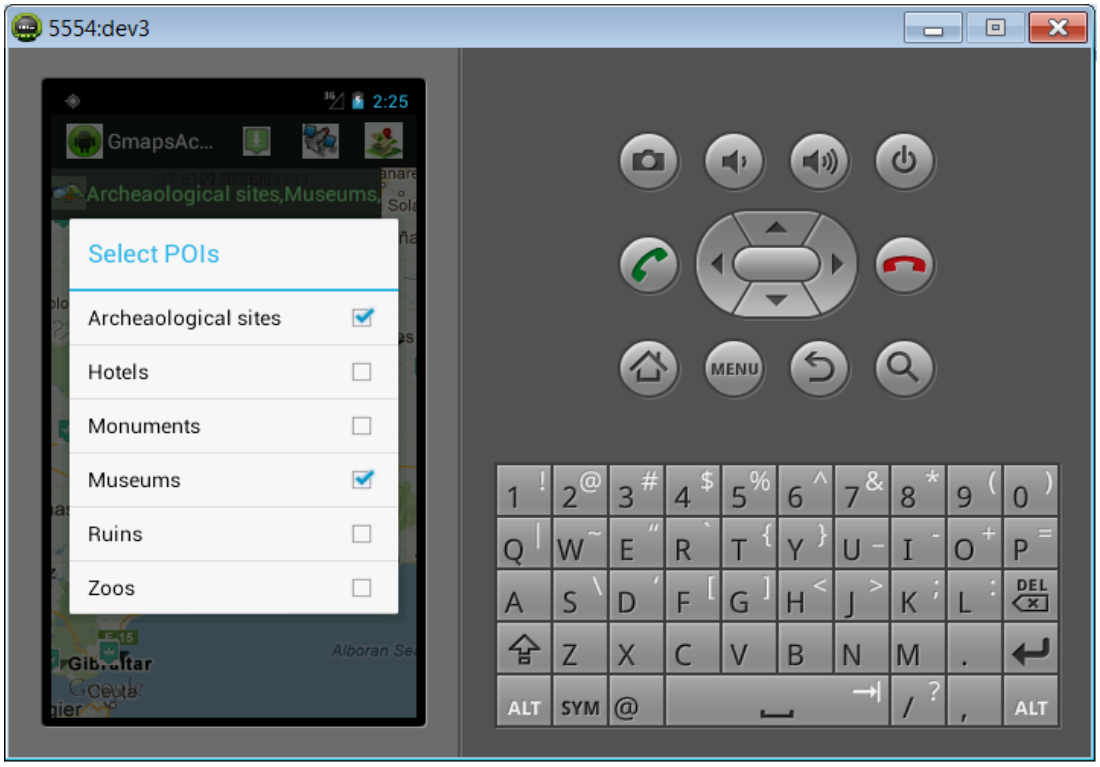
Με αυτή τη λειτουργία, εάν αλλάξει η θέση του χάρτη και δείχνει σε κάποια άλλη περιοχή με το πάτημα του κουμπιού αυτού ο χάρτης μεταφέρεται στη θέση της Τρίπολης.

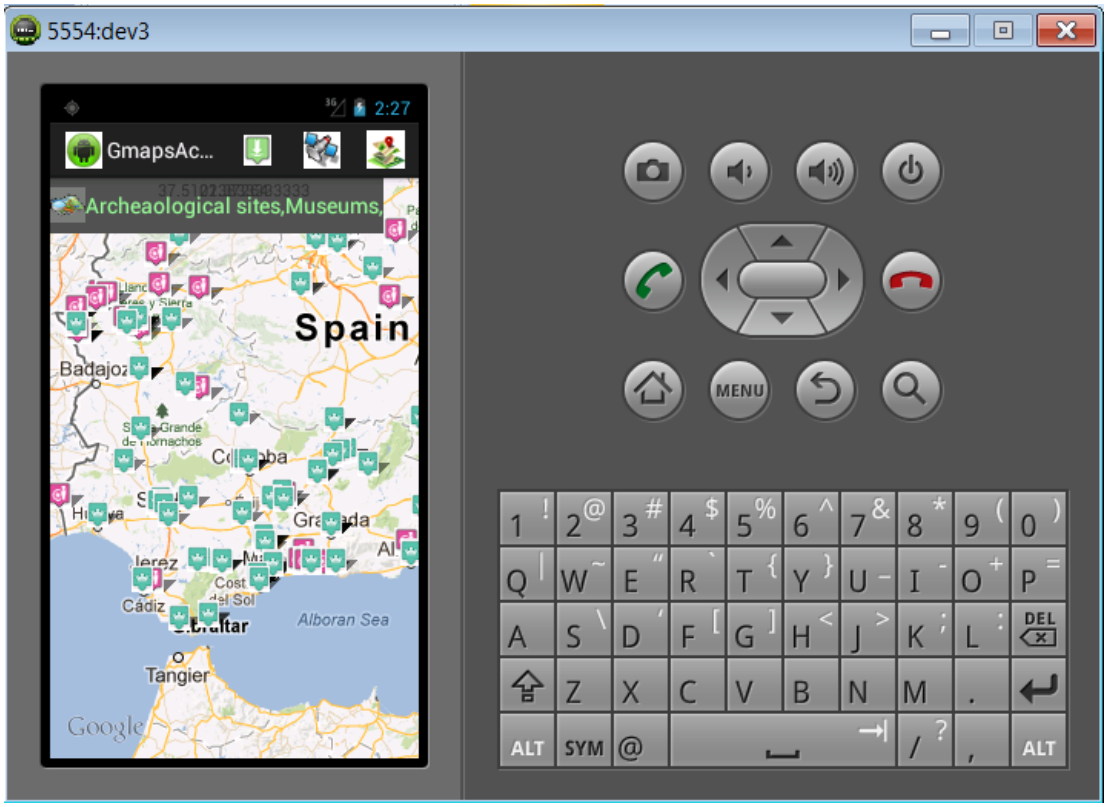
Το πάτημα του κουμπιού ***Select POIs*** είναι μια ακόμη δυνατότητα που δίνει η εφαρμογή στον χρήστη. Μέσω αυτού, εμφανίζεται μια λίστα από τις κατηγορίες σημείων ενδιαφέροντος τις οποίες μπορεί να επιλέξει:



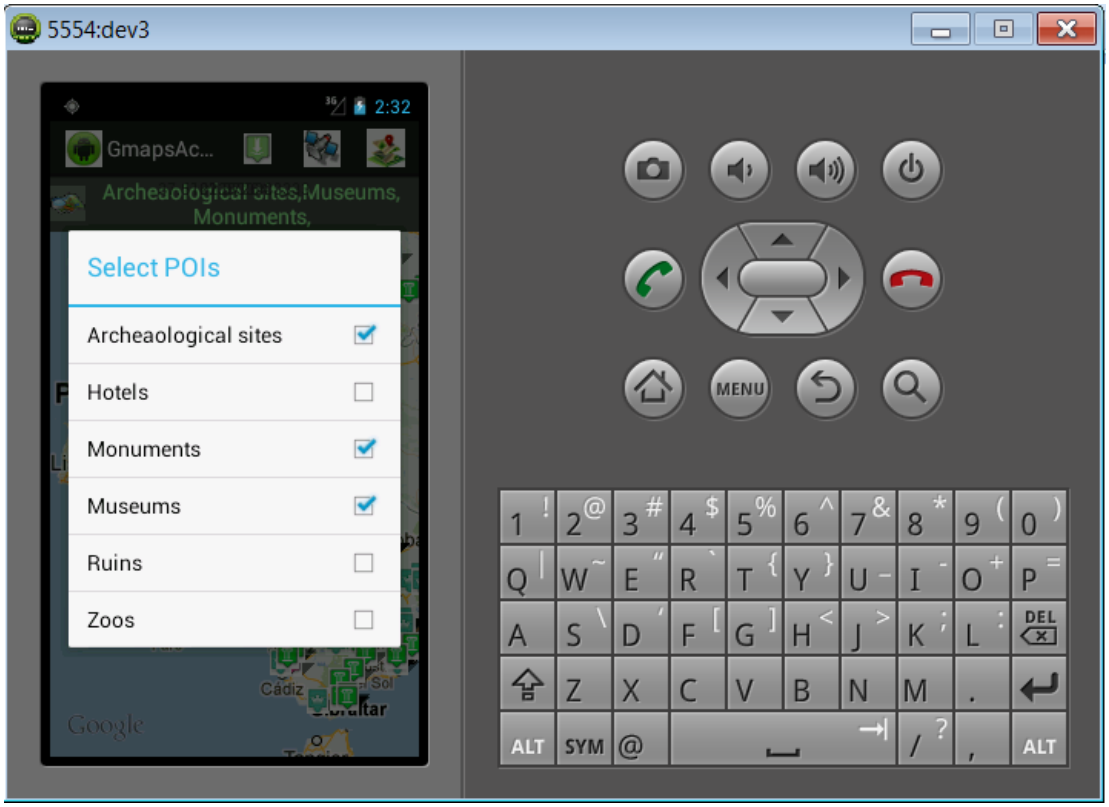
Όταν ο χρήστης επιλέξει μια ή και περισσότερες κατηγορίες εμφανίζονται με διαφορετικό εικονίδιο οι markers στα κατάλληλα σημεία στον χάρτη.

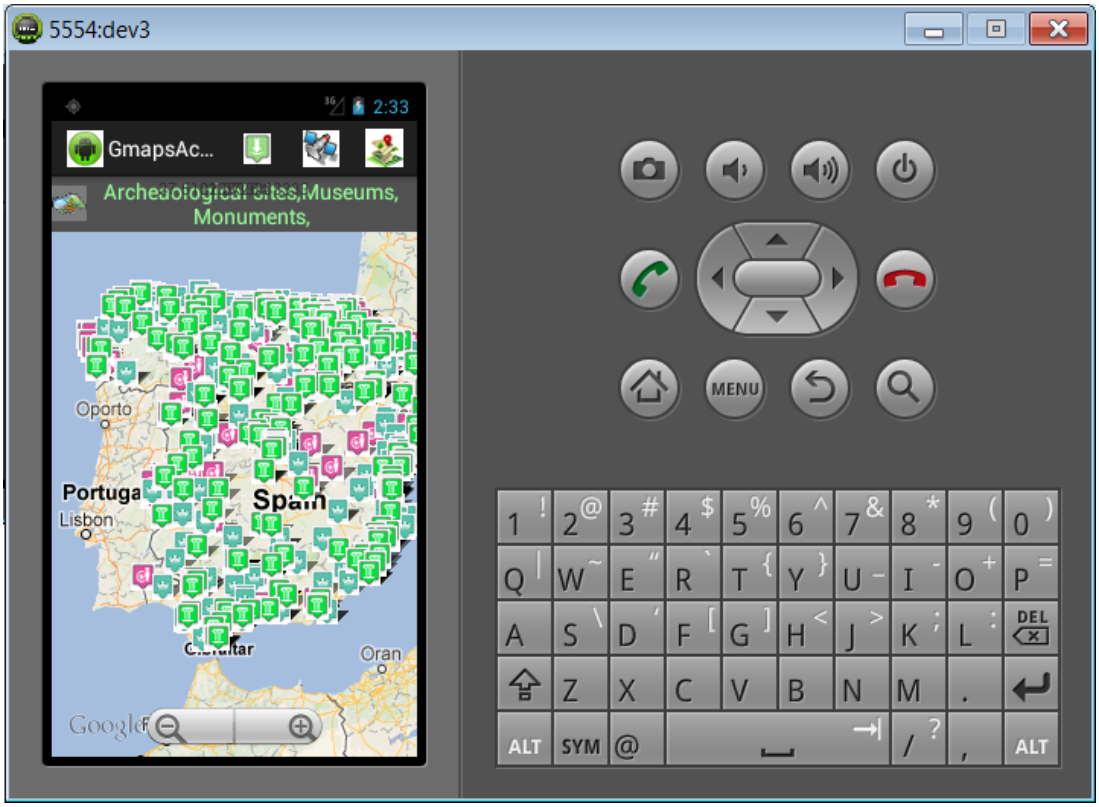
Στην περίπτωση που ο χρήστης επιλέξει τις κατηγορίες «Αρχαιολογικοί χώροι» (τοποθετούνται στον χάρτη τα ροζ εικονίδια) και «Μουσεία» (τοποθετούνται στο χάρτη τα πράσινα εικονίδια) παρουσιάζεται το εξής αποτέλεσμα:





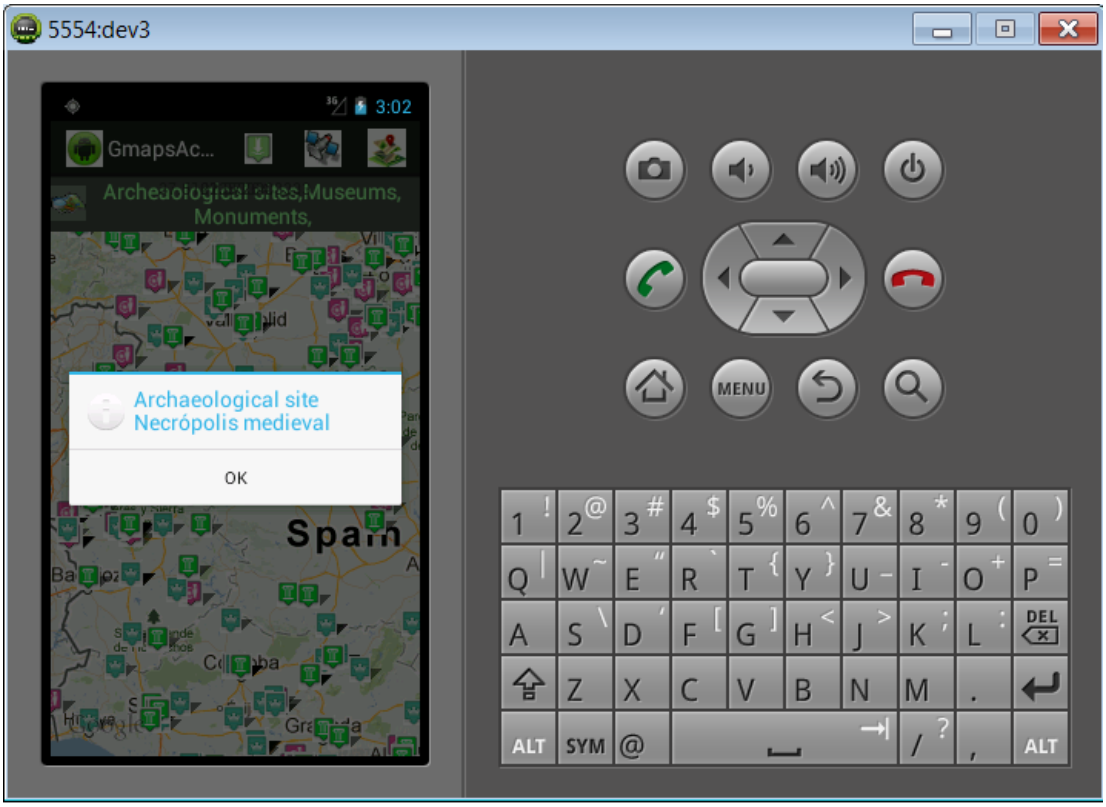
Σε περίπτωση που ο χρήστης επιθυμεί την επιλογή και μιας ακόμη κατηγορίας, για παράδειγμα των «Μνημείων» προστίθενται και οι markers αυτής :





Εάν μέσω του μενού ο χρήστης αφαιρέσει μια κατηγορία σημείων, τότε οι markers της συγκεκριμένης κατηγορίας αφαιρούνται ενώ συνεχίζουν να εμφανίζονται οι markers των υπόλοιπων επιλεγμένων κατηγοριών.

Επιπλέον, ο χρήστης έχει τη δυνατότητα πατώντας επάνω σε οποιονδήποτε marker που βρίσκεται τοποθετημένος στον χάρτη να λαμβάνει το όνομα της τοποθεσίας αυτής μέσω ενός alert box το οποίο περιέχει την πληροφορία. Όπως έχει αναφερθεί, τα ονόματα των σημείων ενδιαφέροντος βρίσκονται αποθηκευμένα στην τρίτη στήλη των πινάκων στην βάση δεδομένων.



Κεφάλαιο 8

Συμπεράσματα και επίλογος

Αναμφισβήτητα, η έννοια της φορητότητας έχει εξαπλωθεί σε όλους σχεδόν τους τομείς της καθημερινής μας ζωής. Οι δυνατότητες που προσφέρουν οι φορητές συσκευές στους κατόχους τους μέσα από μια πληθώρα εφαρμογών που ολοένα και αυξάνονται, είναι ο λόγος που έχουν κερδίσει την εμπιστοσύνη και την προτίμηση των καταναλωτών. Μια από τις σημαντικότερες κατηγορίες εφαρμογών είναι και αυτές της πλοήγησης, μέσα από τις οποίες οι χρήστες τους λαμβάνουν σχεδόν κάθε είδους πληροφορία σχετικά με τις διαδρομές που θέλουν να ακολουθήσουν, τη χρονική διάρκεια για την ολοκλήρωσή τους, την τοποθεσία σημείων ενδιαφέροντος σε συγκεκριμένες περιοχές και πολλά άλλα στοιχεία. Σκοπός της εργασίας αυτής ήταν η ανάπτυξη μιας εφαρμογής για smart phones με κύριο άξονα την πλοήγηση. Το λειτουργικό σύστημα Android, με βάση το οποίο αναπτύχθηκε η εφαρμογή GmapsActivity, παρείχε τη δυνατότητα ανάδειξης των λειτουργιών που μπορεί να έχει μια εφαρμογή μέσω των χαρτών της Google. Οι χάρτες της Google μπορούν να ενσωματωθούν στο εργαλείο ανάπτυξης της εφαρμογής Android, διότι η Google έχει ελευθερώσει το API για το συγκεκριμένο λειτουργικό.

Συμπερασματικά, θα μπορούσαμε να πούμε ότι η εφαρμογή είναι ολοκληρωμένη, αν και περιορίζεται στα πλαίσια μιας συγκεκριμένης χώρας. Η διαχείρισή της από τον χρήστη είναι αρκετά κατανοητή ενώ ταυτόχρονα μπορεί να αποτελέσει ένα χρήσιμο εργαλείο. Οι εφαρμογές που σχετίζονται με την πλοήγηση μπορούν να επεκταθούν σε μεγάλο βαθμό. Πιο συγκεκριμένα, η παρούσα εργασία θα μπορούσε να επεκταθεί δίνοντας τη δυνατότητα στον χρήστη να επιλέξει τα σημεία ενδιαφέροντος μόνο γύρω από μια περιοχή συγκεκριμένης ακτίνας ή ακόμα προσθέτοντας επιπλέον κατηγορίες σημείων αντλώντας τις γεωγραφικές τους συντεταγμένες απευθείας από το διαδίκτυο.

Κεφάλαιο 9

Βιβλιογραφία και ηλεκτρονικές πηγές

Technical Report Analysis report on Android Application Framework and existing Security Architecture , http://serg.imsciences.edu.pk ,February, 2010

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

http://mirnauman.wordpress.com/2012/02/07/

https://developers.google.com/maps/documentation/android/v1/

http://www.javacodegeeks.com/2011/02/android-google-maps-tutorial.html

http://poi-osm.tucristal.es/#

http://mobiforge.com/developing/story/using-google-maps-android

http://www.vogella.com/articles/Android/article.html

Android Tutorial, Larry Walters,2011

Mobile Devices An Introduction to the Android Operating Enviroment, Dominique Heger, http://www.dhtusa.com/media/AndroidInternals.pdf

http://www.wikipedia.org/

http://www.eclipse.org/downloads/

http://universimmedia.pagesperso-orange.fr/geo/loc.htm

http://www.w3schools.com/xml/default.asp

http://stackoverflow.com

http://www.raywenderlich.com/5527/getting-started-with-android-development

http://keytool.sourceforge.net/

City Guide over Android, Hanjie Shu , Spring, 2010, Supervisor: John Krogstie, Jacqueline Floch

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

http://images.google.com/

http://www.openhandsetalliance.com/android_overview.html

http://www.joomplus.gr/component/k2/item/83-install-wamp-server.html

http://mobile.dzone.com/news/android-tutorial-how-parse