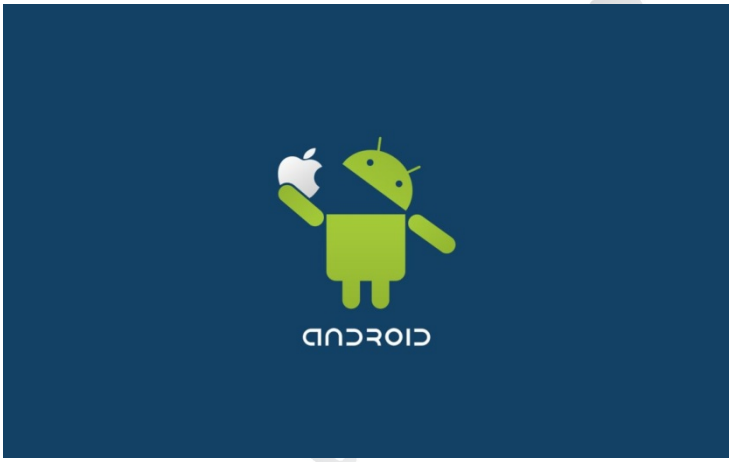
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
Π.Μ.Σ. ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ & ΥΠΗΡΕΣΙΕΣ

ΚΑΤΕΥΘΥΝΣΗ: ΨΗΦΙΑΚΩΝ ΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ



Διπλωματική εργασία με θέμα:

Ανάπτυξη εφαρμογής εντοπισμού και παρακολούθησης πορείας container για φορητή συσκευή με λογισμικό Android.

Φοιτητής Μεταπτυχιακού : Καπετανάκης Εμμανουήλ
ΑΜ:11079

Επιβλέπων Καθηγητής : Μηλιώνης Απόστολος,
Επίκουρος Καθηγητής τμήματος Ψηφιακών Συστημάτων Πανεπιστήμιου Πειραιά

Περιεχόμενα

Contents

Περιεχόμενα εικονογραφήσεων

Περιεχόμενα πινάκων

Περίληψη

Στην παρούσα πτυχιακή εργασία αναλύονται και περιγράφονται τεχνικές ανάπτυξης εφαρμογών για smartphones και αναπτύχθηκε εφαρμογή με χρήση της πλατφόρμας Android SDK.

Στην εισαγωγή αναλύεται και παρουσιάζεται η ραγδαία ανάπτυξη της αγοράς των εφαρμογών για mobile πλατφόρμες και τις προοπτικές που ανοίγονται για τους νέους προγραμματιστές την σύγχρονη εποχή. Γίνεται αναφορά πάνω στις πλέον αναπτυσσόμενες πλατφόρμες και σύγκριση αυτών καθώς και των πλεονεκτημάτων και μειονεκτημάτων που παρουσιάζουν, δίνοντας έμφαση στην πλατφόρμα ανοιχτού κώδικα Android SDK.

Στην συνέχεια περιγράφεται η χρήση και ενσωμάτωση των εργαλείων στο σύστημα ανάπτυξης εφαρμογών του Android, και εμβαθύνουμε πάνω στις πιο χρήσιμες παραμέτρους τους χρησιμοποιώντας την αντικειμενοστραφής γλώσσα προγραμματισμού JAVA. Τέλος, χρησιμοποιώντας τα παραπάνω εργαλεία και τις εμπειρίες που αποκτήθηκαν από την μελέτη αυτών, αναπτύχθηκε μια εφαρμογή διεπαφής χρήστη πάνω στην πλατφόρμα της Google, Android SDK, και περιγράφονται οι δυσκολίες που συναντήθηκαν κατά την ανάπτυξη αυτής.

Εισαγωγή

Από τις αρχές της δεκαετίας του '90 η ανθρωπότητα έχει μπει σε μια νέα εποχή με χαρακτηριστικά που μπορεί να είχαν οραματιστεί επιστήμονες και λογοτέχνες χρόνια πριν, αλλά πήραν για πρώτη φορά υπόσταση λόγω των ιστορικών συνθηκών που έλαβαν χώρα εκείνη την περίοδο και την ολοκλήρωση της μετάλλαξης αυτής βιώνουμε σήμερα. Ο κόσμος του Γκίπσον με την παγκοσμιοποιημένη κοινωνία που όλα ελέγχονται από συστήματα πολύπλοκων υπολογιστικών συστημάτων που επικοινωνούν σε υπέρ-υψηλές ταχύτητες, συσκευές που χρησιμοποιούνται στην καθημερινότητα μας και αποτελούν μέρος των πιο απλών όσο και πιο σύνθετων στιγμών μας. Ένας κόσμος επιστημονικής φαντασίας κάποια χρόνια πριν, ο οποίος είναι μια πραγματικότητα στις μέρες μας, ένας νέος μαγικός μεταβαλλόμενος κόσμος που την εξέλιξή του πια είναι δύσκολο να προβλέψουμε.

Στο πλαίσιο αυτό και έχοντας πια κατοχυρώσει την απρόσκοπτη επικοινωνία και την πρώτη εποχή της πληροφόρησης και του Διαδικτύου, με τους περισσότερους ανθρώπους του αναπτυγμένου και αναπτυσσόμενου κόσμου συνδεδεμένους σε αυτό, βιώνουμε αυτή την στιγμή την ωρίμανση της φάσης του mobility και της πληροφόρησης σε κίνηση. Η επανάσταση που επιτελέστηκε στα τέλη της δεκαετίας του '90 με την εκτεταμένη χρήση του κινητού τηλεφώνου ολοκληρώνεται στις μέρες μας με την σύνδεση αυτής της επανάστασης με την αντίστοιχη του διαδικτύου που έλαβε χώρα την περασμένη δεκαετία.

Διασύνδεση και κίνηση λοιπόν, έρχονται και συνδέονται με το βασικό υποκείμενο της πληροφορικής, τα δεδομένα και κλείνουν με αυτό τον τρόπο το τρίπτυχο που χαρακτηρίζει την μετάδοση πληροφορίας στις μέρες μας. Αποτέλεσμα και έκφανση αυτής της πραγματικότητας είναι ένας ολόκληρος κόσμος κινητών συσκευών με απλές διεπαφές που δίνει την δυνατότητα τόσο παθητικής ενημέρωσης όσο και διαδραστικής επικοινωνίας μέσω των μέσων κοινωνικής δικτύωσης. Ο νέος κόσμος της διασυνδεδεμένης πληροφορίας είναι εδώ και το μέσο που τον παράγει είναι τα έξυπνα τηλέφωνα και οι εφαρμογές των φορητών συσκευών.

Με σαφή επιρροή όλων των παραπάνω, το κεντρικό θέμα της παρούσας εργασίας αποτελεί η ανάπτυξη μιας εφαρμογής, η οποία θα χρησιμοποιηθεί σε φορητές συσκευές android και θα εκμεταλλεύεται πόρους δικτύου, επικοινωνώντας με άλλα σημεία και λαμβάνοντας στοιχεία μέσω γενικότερων υπηρεσιών, και ο τελικός στόχος

βρίσκεται πάνω στην ιδέα της πληροφόρησης, της παροχής δεδομένων κατ' απαίτηση στον τελικό χρήστη. Η εφαρμογή που επιλέχθηκε να υλοποιηθεί, αποτελείται από διάφορα τμήματα που συνθέτουν ένα ενιαίο σύστημα.

Η εφαρμογή αποτελεί ένα εργαλείο πληροφόρησης του χρήστη σχετικά με λεπτομέρειες δρομολογίων που μπορεί να παρέχει ένας φορέας στους ενδιαφερόμενους. Το όλο σύστημα λειτουργεί με μια εφαρμογή επάνω σε μια φορητή συσκευή και μια διαδικτυακή υπηρεσία την οποία η εφαρμογή της φορητής συσκευής «ανακρίνει» για να αντλήσει την πληροφορία που θέλει.

Για την υλοποίηση του συστήματος επιλέχθηκαν οι πιο διαδεδομένες και τεχνολογικά εξελιγμένες λύσεις σε τεχνολογίες και πρότυπα. Επίσης, οι λύσεις που επιλέχθηκαν βασίστηκαν σε συστήματα και εργαλεία ανοιχτού λογισμικού, καθιστώντας μ' αυτό τον τρόπο το σύστημα ευέλικτο, επεκτάσιμο και χωρίς κόστος.

Κεφάλαιο 1ο : Εισαγωγή στο Λειτουργικό Σύστημα Android

1.1 Ιστορική Αναδρομή

Το Android είναι ένα λειτουργικό σύστημα για φορητές συσκευές όπως smartphones και tablets, το οποίο τρέχει πάνω σε πυρήνα λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance, η οποία είναι μια κοινοπραξία εταιριών λογισμικού, κατασκευής hardware και τηλεπικοινωνιών, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις φορητές συσκευές.. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google.

Τον Αύγουστο του 2005, η Google εξαγόρασε την Android Inc, μια μικρή εταιρεία με έδρα το Palo Alto στην California των ΗΠΑ. Οι συνιδρυτές της Android συνέχισαν να εργάζονται στην Google συμπεριλαμβανομένων των Andy Rubin (συν- ιδρυτής της Danger), Rich Miner (συν-ιδρυτής της Wildfire Communications, Inc), Nick Sears (πρώην αντιπρόεδρος της T-Mobile), και Chris White (επικεφαλής σχεδιασμού και ανάπτυξης διεπαφής στο WebTV). Εκείνη την εποχή ελάχιστα ήταν γνωστά για τις λειτουργίες της Android Inc, εκτός του ότι ανέπτυσσαν λογισμικό για κινητά τηλέφωνα. Αυτή ήταν η αρχή της φημολογίας περί σχεδίων της Google για να διεισδύσει στην αγορά κινητής τηλεφωνίας.

Στην Google, η ομάδα με επικεφαλή τον Rubin ανέπτυξε μια πλατφόρμα για φορητές συσκευές που στηρίζεται στον πυρήνα του Linux, την οποία προώθησαν με την παροχή ενός ευέλικτου, αναβαθμίσιμου συστήματος. Έχει αναφερθεί ότι η Google είχε ήδη συγκεντρώσει μια σειρά από εταίρους hardware και software και επισήμανε στους παρόχους ότι ήταν ανοικτή σε διάφορους βαθμούς συνεργασίας εκ μέρους της. Περισσότερες εικασίες ότι η Google θα εισέλθει στην αγορά κινητής τηλεφωνίας άρχισαν τον Σεπτέμβριο του 2006. Δημοσιεύσεις από το BBC και τη The Wall Street Journal πληροφορούσαν ότι η Google ήθελε την έρευνα και τις εφαρμογές σε κινητά τηλέφωνα και εργαζόταν σκληρά για να τις προωθήσουν στην αγορά. Έντυπα και ηλεκτρονικά μέσα ενημέρωσης σύντομα ανέφεραν φήμες ότι η Google ανέπτυσσε μια Google-branded συσκευή. Περισσότερες φήμες ακολούθησαν, αναφέροντας ότι η Google καθόριζε τις τεχνικές προδιαγραφές και έδειχνε πρωτότυπα στους κατασκευαστές κινητών τηλεφώνων και τους φορείς δικτύων.

Τον Σεπτέμβριο του 2007, η InformationWeek κάλυψε μια μελέτη αξιολόγησης αναφέροντας ότι η Google έχει καταθέσει αρκετές πατέντες στον τομέα της κινητής τηλεφωνίας. Τελικά η Google παρουσίασε το smartphone της Nexus One που χρησιμοποιεί το open source λειτουργικό σύστημα Android. Η συσκευή κατασκευάστηκε από την HTC Corporation της Ταϊβάν, και έγινε διαθέσιμη στις 5 Ιανουαρίου 2010.

Εικόνα 1 Google Nexus One

Η πρώτη παρουσίαση της πλατφόρμας Android έγινε στις 5 Νοεμβρίου 2007, παράλληλα με την ανακοίνωση της ίδρυσης του οργανισμού Open Handset Alliance (OHA), μιας κοινοπραξίας 48 τηλεπικοινωνιακών εταιριών, εταιριών λογισμικού καθώς και κατασκευής υλικού, οι οποίες είναι αφιερωμένες στην ανάπτυξη και εξέλιξη ανοιχτών προτύπων στις συσκευές ανοιχτής τηλεφωνίας. Μεταξύ αυτών των εταιριών ήταν η Google, η HTC, η Sony, η Dell, η Intel, η Samsung Electronics, η LG Electronics κ.α.

Η Google το 2008 απελευθέρωσε και τον κώδικα του λειτουργικού συστήματος, καθώς είχε ως στόχο να προσφέρει μια ευέλικτη πλατφόρμα ανάπτυξης για να προσελκύσει προγραμματιστές να δημιουργήσουν εφαρμογές για κινητές συσκευές Android. Στη συνέχεια μέσω αυτών των εφαρμογών θα προσέλκυε καταναλωτές για να αγοράσουν κινητές συσκευές τέτοιου τύπου. Η Google άνοιξε τον κώδικα του λειτουργικού υπό τους όρους της Apache License, μιας ελεύθερης άδειας λογισμικού.

Το Android έχει μια μεγάλη κοινότητα προγραμματιστών, οι οποίοι αναπτύσσουν εφαρμογές που επεκτείνουν την λειτουργικότητα των συσκευών. Τον Σεπτέμβριο του 2013 υπήρχαν περίπου 850.000 εφαρμογές διαθέσιμες για Android και ο εκτιμώμενος αριθμός λήψης εφαρμογών από το Google Play, το βασικό κατάστημα εφαρμογών Android, ήταν 25 δισεκατομμύρια.

Αυτοί οι παράγοντες έχουν επιτρέψει στο Android να γίνει το πιο δημοφιλές χρησιμοποιούμενο λειτουργικό σε έξυπνες φορητές συσκευές και η πρώτη επιλογή σε εταιρείες που απαιτούν μια χαμηλού κόστους, επεκτάσιμη, «ελαφριά» λύση σε λειτουργικό σύστημα για συσκευές υψηλής τεχνολογίας. Το Android έχει παγκοσμίως το 80% της αγοράς σε έξυπνες φορητές συσκευές και κατά το πρώτο τρίμηνο του 2014 κατείχε το 44% σε πωλήσεις με 187 εκατομμύρια συσκευές παγκοσμίως.

Worldwide Smartphone Forecast by Region, Shipments, Market Share and 5-Year CAGR (units in millions)

Operating System	2014 Shipment Volumes*	2014 Market Share	2018 Shipment Volumes*	2018 Market Share	2013-2018 CAGR
Android	997.7	80.2%	1,401.3	77.6%	12.0%
iOS	184.1	14.8%	247.4	13.7%	10.0%
Windows Phone	43.3	3.5%	115.3	6.4%	28.1%
BlackBerry	9.7	0.8%	4.6	0.3%	-25.0%
Others	9.3	0.7%	37.7	2.1%	31.5%
Total	**1,244.1**	**100%**	**1,806.3**	**100%**	**12.3%**

Πίνακας 1 Worldwide Smartphone Forecast

1.2 Εκδόσεις και χαρακτηριστικά

Όπως αναφέραμε παραπάνω, το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα. Η εξέλιξη του λόγω της open source φύσης του είναι ραγδαία και αυτό αντικατοπτρίζεται στο γεγονός ότι οι 9 κύριες εκδόσεις του έχουν κυκλοφορήσει σε διάστημα 4.5 ετών, από τον Απρίλη του 2009 μέχρι τον Νοέμβριο του 2013.

Στην πληροφορική συνηθίζεται τα προϊόντα hardware και software να κυκλοφορούν εκτός από τον αριθμό έκδοσης τους, και με μία κωδική ονομασία. Η κωδική ονομασία μπορεί να είναι πχ ονόματα πόλεων (Windows Viena, Chicago), ονόματα ζώων (OSX Leopard, Lion), στην περίπτωση όμως του Android τα κώδικα ονόματα έρχονται στη μορφή επιδόρπιου!

1.2.1 Android 1.0 – G1 or Astro or Petit Flour – API 1

Η πρώτη εμπορική έκδοση του Android (Android 1.0) κυκλοφόρησε το Σεπτέμβριο του 2008 και λειτούργησε επάνω στην πρώτη συσκευή για Android, την HTC Dream. Τα χαρακτηριστικά της αφορούσαν κυρίως επαγγελματική χρήση, όπως GPS, Bluetooth και μερικές εφαρμογές Google, όπως Gmail, Google Calendar. Ακολούθησε το Φεβρουάριο του 2009 η έκδοση 1.1, η οποία αναφερόταν αρχικά στη συσκευή HTC Dream και αφορούσε κυρίως διόρθωση σφαλμάτων. Παράλληλα δημιουργήθηκε και το Android Market, που για αρχή περιείχε περίπου 35εφαρμογές. Η έκδοση ήταν γνωστή ως Petit Four, ονομασία η οποία όμως δε χρησιμοποιήθηκε ποτέ επίσημα.

1.2.2 Android 1.5 – Cupcake – API 3

Η έκδοση "Cupcake", βασισμένη στο Linux Kernel 2.6.27, παρουσιάστηκε στις 30 Απριλίου του 2009. Υποστηρίζει νέες λειτουργίες για την κάμερα τις συσκευής, όπως η καταγραφή και παρακολούθηση βίντεο από την λειτουργία της κάμερας και η άμεση μεταφόρτωση του βίντεο αλλά και των φωτογραφιών στο Youtube και το Picasa αντίστοιχα απευθείας από το τηλέφωνο. Έχει νέο έξυπνο πληκτρολόγιο με πρόβλεψη κειμένου. Υποστηρίζει πρότυπο Bluetooth A2DP και AVRCP ενώ έχει και την ικανότητα να συνδέεται αυτόματα σε μικροσυσκευές Bluetooth από μια συγκεκριμένη απόσταση. Ακόμα στην έκδοση αυτή έχει νέο γραφικό περιβάλλον με κινούμενες μεταβάσεις οθόνης.

1.2.3 Android 1.6 – Donut – API 4

Η έκδοση 1.6, γνωστή ως Donut, κυκλοφόρησε το Σεπτέμβριο του 2009, βασιζόμενη στον πυρήνα Linux kernel 2.6.29 και περιείχε αρκετές νέες προσθήκες. Είχε βελτιωμένες λειτουργίες σύνδεσης με το Android Market. Επίσης, βελτίωνε τις αναζητήσεις, παρέχοντας πλέον στο χρήστη δυνατότητα αναζήτησης μέσω φωνής. Οι αναζητήσεις μπορούσαν πλέον να γίνονται με κείμενο και φωνή και δεν περιορίζονται μόνο για το web. Η νέα έκδοση έφερε επίσης υποστήριξη για οθόνη αφής υψηλότερης

ανάλυσης από αυτές που χρησιμοποιούνταν έως τότε, βελτίωνε την κάμερα και παρείχε τη δυνατότητα σύνδεσης των κινητών τηλεφώνων μέσω δορυφόρου.

1.2.4 Android 2.0 / 2.1 – Éclair – API 7

Η έκδοση 2.0, γνωστή ως Eclair, κυκλοφόρησε τον Οκτώβριο του 2009, βασιζόμενη επίσης στον πυρήνα Linux Kernel 2.6.29, και αναβαθμίστηκε από την 2.1 τον Ιανουάριο του 2010. Περιείχε αρκετές βελτιώσεις όσον αφορά τη διεπαφή του χρήστη καθώς και την ταχύτερη απόκριση του υλικού. Επίσης εισήγαγε το εικονικό πληκτρολόγιο. Περιείχε βελτιωμένα γραφικά, υποστήριξη του προτύπου html 5 και βελτιωμένους χάρτες. Ενσωματώθηκε η υποστήριξη φλας για την κάμερα, η οποία έχει πλέον και ψηφιακό zoom.

1.2.5 Android 2.2 – Froyo – API 8

Η έκδοση 2.2, γνωστή ως Froyo, κυκλοφόρησε το Μάιο του 2010, βασιζόμενη στον πυρήνα Linux Kernel 2.6.32. Η βασική βελτίωση της συγκεκριμένης έκδοσης ήταν η ταχύτητα του λειτουργικού συστήματος, χάρη στο μηχανισμό Java V8 και τον JIT compiler, ο οποίος εκκινούσε εφαρμογές πολύ πιο γρήγορα. Υποστηρίζονται πλέον το Adobe Flash 10.1 και διορθώνεται η υποστήριξη στον Microsoft Exchange. Ο χρήστης μπορεί να ελέγχει την κίνηση πακέτων δεδομένων και υπάρχει η δυνατότητα εγκατάστασης εφαρμογών στην κάρτα μνήμης. Επίσης, εισήγαγε χαρακτηριστικά όπως USB Tethering και Portable Wi-Fi Hotspot, δίνοντας τη δυνατότητα ένα τηλέφωνο να μετατραπεί σε Wi-Fi Hotspot.

1.2.6 Android 2.3 – Gingerbread – API 10

Η έκδοση 2.3, γνωστή ως Gingerbread, κυκλοφόρησε το Δεκέμβριο του 2010, βασιζόμενη στον πυρήνα Linux Kernel 2.6.35. Στη συγκεκριμένη έκδοση υπάρχουν αλλαγές στο user interface το οποίο έχει γίνει πιο γρήγορο και απλό και υποστηρίζονται οθόνες μεγάλων μεγεθών και αναλύσεων. Υπάρχει πλέον το πρωτόκολλο SIP για κλήσεις μέσω VoIP, και υποστήριζονται οι τύποι video WebM/VP8 και ο κωδικοποιητής AAC. Έχει βελτιωθεί ο ήχος καθώς και οι λειτουργίες απεικόνισης για την ανάπτυξη παιχνιδιών. Υποστηρίζεται το Near Field Communication (NFC) και η ύπαρξη πολλαπλών καμερών. Επίσης έχει βελτιωθεί η ενεργειακή υποστήριξη και έχει γίνει μετάβαση από το σύστημα αρχείων YAFFS στο Ext4.

1.2.7 Android 3.0 – Honeycomb – API 13

Η έκδοση 3.0, γνωστή ως Honeycomb, κυκλοφόρησε το Φεβρουάριο του 2011, βασιζόμενη στον πυρήνα Linux Kernel 2.6.36. Η συγκεκριμένη έκδοση είχε σχεδιαστεί κυρίως για tablets. Περιέχει ένα νέο και διαφορετικό user interface, καθώς όλα πλέον βρίσκονται στην οθόνη και δεν υπάρχει ανάγκη για φυσικά κουμπιά. Υποστηρίζονται διπύρηνοι και τετραπύρηνοι επεξεργαστές και έχει απλοποιηθεί κατά πολύ το multitasking, ώστε ο χρήστης να περνάει από μια εφαρμογή σε άλλη. Υπάρχει η δυνατότητα video chat μέσω του Google talk καθώς και η ανάγνωση βιβλίων μέσω του Google eBooks. Τέλος, μπορούν να κρυπτογραφηθούν όλα τα δεδομένα χρήστη.

1.2.8 Android 4.0 – Ice Cream Sandwich – API 15

Η έκδοση 4.0, γνωστή ως Ice Cream Sandwich, κυκλοφόρησε τον Οκτώβριο του 2011 και αναβαθμίστηκε το Μάρτιο toy 2012, βασιζόμενη στον πυρήνα Linux Kernel 3.0.1. Η απόδοση του συστήματος έχει βελτιωθεί και έχουν ενσωματωθεί σε φορητές συσκευές τα νέα χαρακτηριστικά, τα οποία είχαν εισαχθεί στην προηγούμενη έκδοση για τα tablets. Επίσης, βελτιώθηκε η ασφάλεια του συστήματος με την προσθήκη αναγνώρισης προσώπου για να ξεκλειδώνει η συσκευή. Ο browser μπορεί

να ανοίξει ταυτόχρονα μέχρι και 16 καρτέλες. Υπάρχει η δυνατότητα ο χρήστης να τερματίσει εφαρμογές οι οποίες τρέχουν στο παρασκήνιο, ενώ μπορεί να θέσει και όρια στην κίνηση πακέτων δεδομένων. Η εφαρμογή Android Beam αξιοποιεί πλέον το NFC αφού επιτρέπει την αποστολή δεδομένων από τη συσκευή σε όσους βρίσκονται εντός μιας μικρής ακτίνας εμβέλεια. Τέλος εμφανίζεται και η εφαρμογή Chrome για Android, η πιο γρήγορη έκδοση του Chrome για οποιαδήποτε φορητή συσκευή.

1.2.9 Android 4.1 – JellyBean – API 16 - 18

Η έκδοση 4.1, γνωστή ως Jellybean, κυκλοφόρησε τον Ιούνιο του 2012, βασιζόμενη στον πυρήνα Linux Kernel 3.0.31. Στόχος της έκδοσης ήταν η βελτίωση της λειτουργικότητας και της απόδοσης του user interface. Σημαντικές αλλαγές έχουν γίνει στην περιοχή των ειδοποιήσεων. Επίσης έχουν προστεθεί νέα widget, η υπηρεσία Google Now, ως μέρος του ανανεωμένου Google Search, βελτιώσεις στο ημερολόγιο και την εφαρμογή Chrome, η οποία πλέον υποστηρίζει θεωρητικά απεριόριστο αριθμό καρτελών.

1.2.10 Android 4.4 – KitKat – API 19 – 20

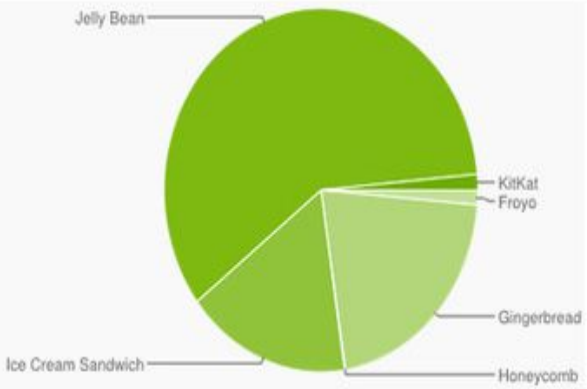
Η έκδοση 4.4, γνωστή ως KitKat, κυκλοφόρησε τον Οκτώβριο του 2013, βασιζόμενη στον πυρήνα Linux Kerner 3.8. Στόχος της έκδοσης ήταν η βελτίωση της λειτουργικότητας και η υποστήριξη ενός ακόμη μεγαλύτερου εύρους συσκευών σε σχέση με τις προηγούμενες εκδόσεις του Android, αφού ως ελάχιστη προτεινόμενη μνήμη συσκευής είναι 340MB. Επίσης, αλλαγές έγιναν στην διαχείριση της κάλυψης, καθώς η μετάβαση από κυψέλη σε κυψέλη θα γίνεται ομαλότερα. Τέλος, βελτιώσεις έγιναν και στην διαχείριση της ενέργειας, με νέο πρόγραμμα διαχείρισης εφαρμογών προσφέροντας μεγαλύτερους χρόνους αναμονής.

Εικόνα 2 Εκδόσεις Android

1.2.11 Εκδόσεις Android σε χρήση

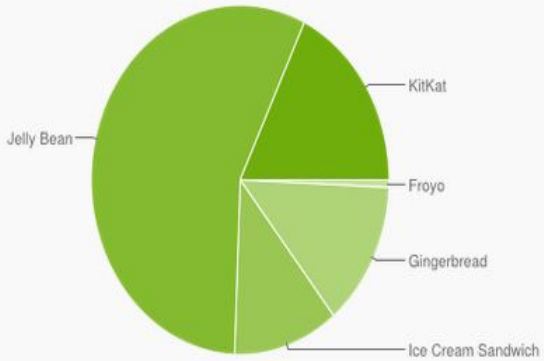
Σύμφωνα με τις τελευταίες μετρήσεις, που έδωσε η Google στην δημοσιότητα, γίνεται πλέον σαφές, ότι όλο και περισσότεροι χρήστες στρέφονται σε νεότερες εκδόσεις του Android, αλλά όχι ακόμα στην τελευταία. Παρόλο που τα ποσοστά της έκδοσης KitKat αυξάνονται ραγδαίως από την αρχή του χρόνου (1.4% τον Ιανουάριο του 2014 σε 17.9% τον Ιούλιο 2014), νικήτρια έκδοση παραμένει η Jelly Bean (59.1% τον Ιανουάριο του 2014 σε 56.5% τον Ιούλιο 2014). Σύμφωνα με εκτιμήσεις της Google, μέχρι το τέλος του 2014 οι συσκευές που θα έχουν την έκδοση KitKat θα φτάσουν το ποσοστό του 40%.

Version	Codename	API	Distribution
2.2	Froyo	8	1.3%
2.3.3 - 2.3.7	Gingerbread	10	21.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	16.9%
4.1.x	Jelly Bean	16	35.9%
4.2.x		17	15.4%
4.3		18	7.8%
4.4	KitKat	19	1.4%



Πίνακας 2 Android Version Statistics January 2014

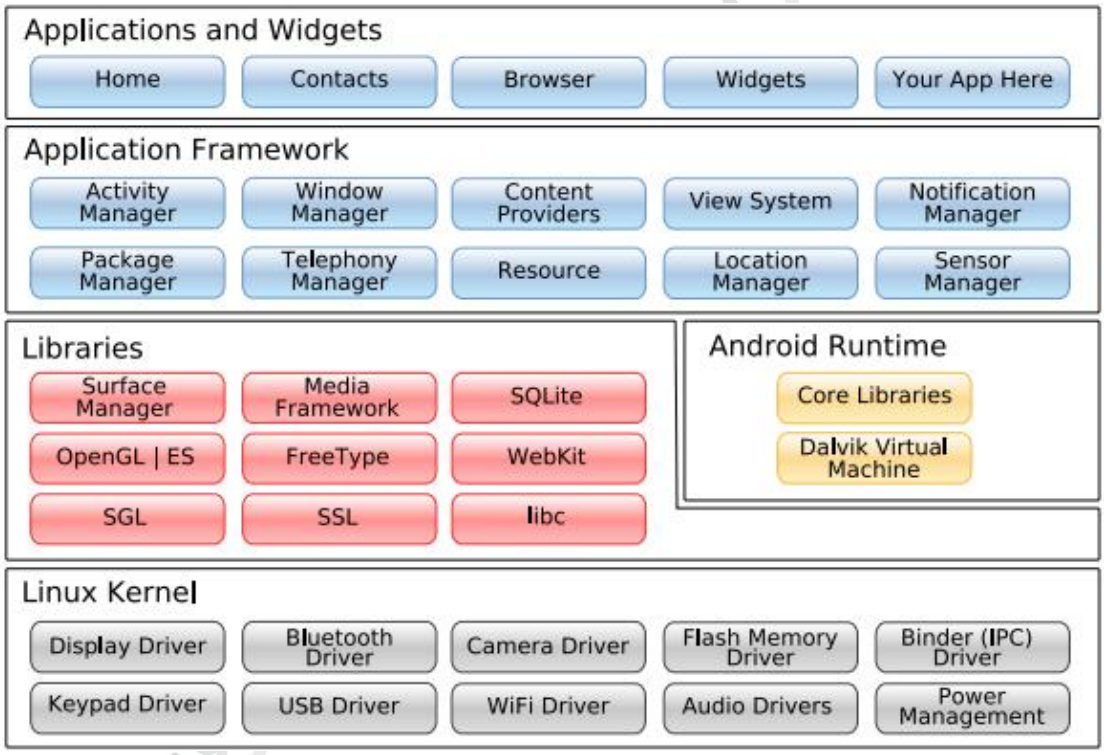
Version	Codename	API	Distribution
2.2	Froyo	8	0.7%
2.3.3 - 2.3.7	Gingerbread	10	13.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	11.4%
4.1.x	Jelly Bean	16	27.8%
4.2.x		17	19.7%
4.3		18	9.0%
4.4	KitKat	19	17.9%



Πίνακας 3 Android Version Statistics July 2014

1.3 Αρχιτεκτονική του Android

Το Android δεν είναι μόνο ένα λειτουργικό σύστημα. Είναι μια στοίβα λογισμικού η οποία αποτελείτε από το λειτουργικό σύστημα, τις υπηρεσίες διασύνδεσης με τις εφαρμογές (middleware) και από τις κύριες (core) εφαρμογές, μεταξύ αυτών, ενός email cient, μιας εφαρμογής διαχείρισης SMS, ενός ημερολογίου, ενός browser, εφαρμογή διαχείρισης επαφών, και άλλες οι οποίες έρχονται δεμένες με την υπόλοιπη στοιβάδα λογισμικού του Android. Το Android αποτελείται από 4επίπεδα και από 5ομάδες συνιστωσών τα οποία περιγράφονται παρακάτω και απεικονίζονται και στην εικόνα 3.



Εικόνα 3 Αρχιτεκτονική Android

1.3.1 Linux Kernel

Η βάση της στοίβας λογισμικού του Android είναι ο πυρήνας Linux. Ο τροποποιημένος πυρήνας του συστήματος βασίζεται στην έκδοση 2.6 (και στην έκδοση 3.8 για το Android 4.4) του Linux Kernel, η οποία υποστηρίζει όλες τις κύριες λειτουργίες του λειτουργικού συστήματος. Οι λειτουργίες αυτές αφορούν διαχείριση

μνήμης, διαχείριση διεργασιών, λειτουργίες δικτύου, ασφάλεια του λειτουργικού, και ένα σύνολο οδηγών υλικού (hardware drivers). Οι οδηγοί αυτοί είναι υπεύθυνοι για την επικοινωνία του software με το hardware της συσκευής. Ενδεικτικά ο πυρήνας του Android περιέχει:

- Οδηγό προβολής οθόνης
- Οδηγό Wifi και Bluetooth
- Οδηγό κάμερας
- κλπ

Ο πυρήνας του Android μπορεί να βασίζεται στον πυρήνα του Linux, αλλά διαφέρει αρκετά από αυτόν. Ο λόγος είναι οι αλλαγές στην αρχιτεκτονική που έχει κάνει η Google για να είναι ελαφρύτερος και βελτιστοποιημένος για χρήση σε κινητές συσκευές. Αυτό σημαίνει ότι παρότι το Android είναι κατά βάση Linux, επί της ουσίας είναι αρκετά δύσκολο να τρέξουν εφαρμογές ή να χρησιμοποιηθούν βιβλιοθήκες από τη μία πλατφόρμα στην άλλη. Ο Linus Torvalds, δημιουργός του πυρήνα Linux, έχει αναφέρει ότι τελικά στο μέλλον το Android και το Linux θα μοιράζονται έναν κοινό πυρήνα, αλλά αυτό υπολογίζεται να πραγματοποιηθεί στις αρχές της δεκαετίας του 2020.

1.3.2 Βιβλιοθήκες

Στο 2ο επίπεδο της στοίβας, βρίσκονται οι βιβλιοθήκες του Android. Αυτές ουσιαστικά αποτελούν τα APIs που είναι διαθέσιμα στους προγραμματιστές για την ανάπτυξη των εφαρμογών. Οι βιβλιοθήκες από μόνες τους δεν αποτελούν εφαρμογές αλλά ενσωματώνονται και χρησιμοποιούνται από τις εφαρμογές για τις διάφορες λειτουργίες που παρέχει η καθεμία από αυτές. Ουσιαστικά αποτελούν ένα από τα δομικά υλικά των εφαρμογών, και άρα είναι αναπόσπαστο τμήμα τους. Οι δυνατότητες των βιβλιοθηκών του Android γίνονται εμφανείς στους προγραμματιστές στην στοίβα του πλαισίου εφαρμογής. Το σύνολο σχεδόν των βιβλιοθηκών είναι γραμμένο σε C και C++, οι οποίες έχουν μεταγλωττιστεί για τη χρήση τους από το λειτουργικό σύστημα.

Μερικές από τις κύριες βιβλιοθήκες του Android είναι:

- Surface Manager: Διαχειρίζεται την πρόσβαση στο υποσύστημα προβολής και συνθέτει δισδιάστατα (2D) και τρισδιάστατα (3D) επίπεδα γραφικών τα οποία προέρχονται από πολλαπλές εφαρμογές.
- Media Framework: Υποστηρίζει την αναπαραγωγή και εγγραφή πολλών δημοφιλών μέσων ήχου και εικόνας.
- SGL: Μηχανή αναπαραγωγής δισδιάστατων (2D) γραφικών.
- OpenGL | ES: Μηχανή αναπαραγωγής δισδιάστατων (3D) γραφικών.
- FreeType: Παρέχει ευκρίνεια γραφικών στις γραμματοσειρές και στα bitmaps των εφαρμογών του συστήματος.
- SQLite: Βάση δεδομένων.
- WebKit: Μηχανή υποστήριξης πλοήγησης στο διαδίκτυο. Χρησιμοποιείται από τον ενσωματωμένο browser της συσκευής αλλά και στα WebViews των εφαρμογών.
- LibC: Παρέχει τις βιβλιοθήκες της γλώσσας C, ειδικά τροποποιημένες για κινητές συσκευές βασισμένες στο Linux.
- SSL: Μηχανή κρυπτογράφησης.

1.3.3 Android RunTime – Χρόνος Εκτέλεσης Εφαρμογής

Ο χρόνος εκτέλεσης των εφαρμογών του Android, βρίσκεται στο ίδιο επίπεδο με τις κύριες βιβλιοθήκες και την μηχανή Dalvik. Στην πράξη, αποτελεί μία ενοποίηση μεταξύ των δυνατοτήτων που παρέχουν οι βιβλιοθήκες και του χρόνου εκτέλεσης της εικονικής μηχανής Dalvik.

1.3.3.1 Virtual Machine Dalvik - Εικονικής Μηχανή Dalvik

Η εικονική μηχανή Dalvik είναι υπεύθυνη για την δημιουργία των εκτελέσιμων αρχείων των εφαρμογών προκειμένου να τα εκτελέσει το λειτουργικό σύστημα. Σχεδόν το σύνολο των APIs του Android βασίζονται στη γλώσσα προγραμματισμού Java. Στην Java υπάρχει η Java Virtual Machine, στην οποία εκτελείται ο κώδικας των εφαρμογών.

Αντιστοίχως, στο Android υπάρχει η Dalvik που εκτελεί τον κώδικα των εφαρμογών. Η κάθε εφαρμογή εκτελείται μέσω της δικής της εικονικής μηχανής στην δική της διεργασία, οπότε καμία εφαρμογή δεν έχει επαφή με την αλλή, παρόλο που εκτελούνται ταυτόχρονα.

1.3.4 Application Framework – Πλαίσιο Εφαρμογής

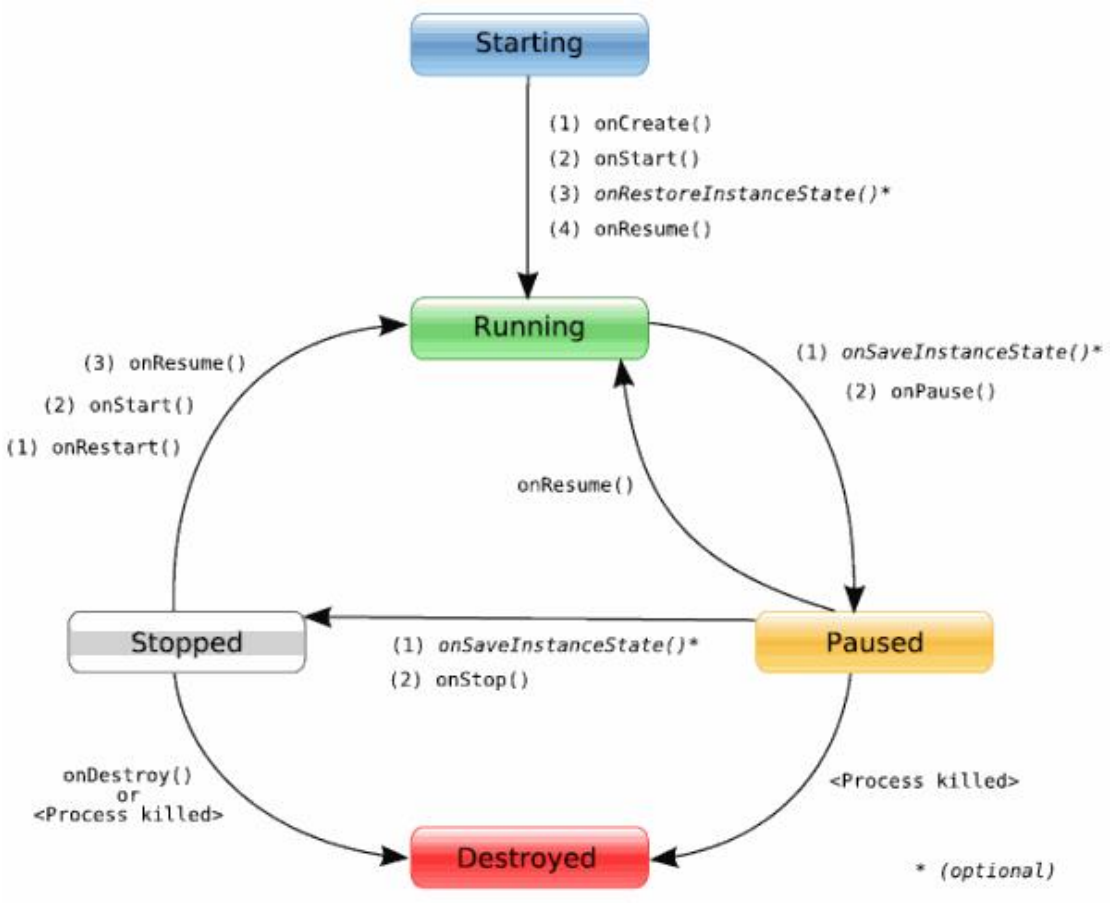
Το Android παρέχει στους developers μια ανοιχτού κώδικα πλατφόρμα ανάπτυξης και τη δυνατότητα να αναπτύξουν με αυτή ιδιαίτερα καινοτόμες και πλούσιες σε υλικό, εφαρμογές. Οι εφαρμογές έχουν πρόσβαση στις βασικές βιβλιοθήκες του λειτουργικού συστήματος, μέσω κατάλληλων διεπαφών, και μέσω του Application Framework μπορούν με τη σειρά τους να παρέχουν επιπρόσθετες λειτουργίες-υπηρεσίες προς άλλες εφαρμογές, εφόσον κάτι τέτοιο φυσικά δεν περιορίζεται από τις πολιτικές ασφαλείας του Application Framework. Οι developers έχουν στην διάθεση τους τη δυνατότητα ελέγχου του υλικού της συσκευής και μέσω αυτής μπορούν να αποκτήσουν πρόσβαση σε υπηρεσίες εντοπισμού, εκτέλεση διεργασιών παρασκηνίου, και πάρα πολλές ακόμη δυνατότητες οι οποίες βασίζονται στα APIs που είναι διαθέσιμα.

Μερικές από τις πιο βασικές οντότητες που περιλαμβάνονται στο πλαίσιο του Application Framework είναι:

- Content Provider – Πάροχος Περιεχομένου: Επιτρέπει στις εφαρμογές να μοιράζονται και να ανταλλάσσουν δεδομένα μιας συγκεκριμένης μορφής, η οποία ορίζεται από τον πάροχο. Χαρακτηριστικά παραδείγματα αποτελούν οι επαφές χρήστη και οι βάσεις δεδομένων των εφαρμογών
- Resource Manager – Διαχειριστής Πόρων: Παρέχει πρόσβαση σε υλικό το οποίο δεν είναι σε μορφή κώδικα, όπως μέσα αναπαραγωγής και αρχεία xml.
- Notification Manager – Διαχειριστής Ειδοποιήσεων: Παρέχει στις εφαρμογές πρόσβαση στις υπηρεσίες ειδοποίησης χρήστη. Παραδείγματα ειδοποιήσεων είναι οι ήχοι, η δόνηση, η ενεργοποίηση της οθόνης, τα toast μηνύματα στο πάνω μέρος της οθόνης.
- View System – Σύστημα Προβολής: αποτελείται από αντικείμενα γραφικού περιβάλλοντος (GUI) τα οποία μπορούν να χρησιμοποιηθούν στον σχεδιασμό

μίας εφαρμογής. Παραδείγματα αντικειμένων είναι τα κουμπιά, τα πεδία εισαγωγής κειμένου, οι λίστες.

- Package Manager – Διαχειριστής Πακέτων: Περιέχει διαφόρων είδους πληροφορίες για τις εφαρμογές που είναι εγκατεστημένες στην συσκευή.
- Location Manager – Διαχειριστής Τοποθεσίας: Περιέχει τις υπηρεσίες τοποθεσίας.
- Activity Manager – Διαχειριστής Δραστηριοτήτων: Διαχειρίζεται τον κύκλο ζωής των δραστηριοτήτων και παρέχει την δυνατότητα πλοήγησης μεταξύ των δραστηριοτήτων, κρατώντας αποθηκευμένη στην μνήμη την σειρά εκτέλεσης αυτών. Στον παρακάτω πίνακα φαίνεται λεπτομερώς ο κύκλος ζωής κάθε δραστηριότητας.



Πίνακας 4 Activity Manager

Περιγραφή των κυριότερων μεθόδων που χρησιμοποιούνται σε ένα Activity:

- **onStart( ):** Αυτή η μέθοδος υποδεικνύει ότι το activity πρόκειται να εμφανιστεί στον χρηστή.

- **onResume( ):** Αυτή η μέθοδος καλείται, όταν το activity είναι έτοιμο για αλληλεπίδραση με τον χρήστη.
- **onPause( ):** Αυτή η μέθοδος καλείται, όταν το activity περάσει σε αναστολή.
- **onStop( ):** Αυτή η μέθοδος καλείται, όταν το activity περάσει σε διακοπή (stopped). Αν οι πόροι του συστήματος είναι χαμηλοί τότε το σύστημα μπορεί να καταστρέψει το activity χωρίς να την καλέσει.
- **onRestart( ):** Αυτή η μέθοδος καλείται, όταν το activity πρόκειται να ξαναεμφανιστεί μετά από μια κατάσταση διακοπής.
- **onDestroy( ):** Αυτή η μέθοδος καλείται, ακριβώς πριν το activity καταστραφεί. Επίσης μπορεί να μην κληθεί ποτέ, λόγω χαμηλών πόρων του συστήματος.

1.3.5 Application and Widgets – Εφαρμογές

Σε αυτή την κατηγορία, βρίσκονται οι εφαρμογές που θα χρησιμοποιήσει ο τελικός χρήστης, με διαφάνεια ως προς το τι συμβαίνει πίσω από αυτές ή τι απαιτεί κάθε εφαρμογή για την εκτέλεση τους από το λειτουργικό σύστημα. Μερικές από τις βασικότερες εφαρμογές είναι ο browser, email client, αποστολή και λήψη SMS, προβολή χαρτών σε συνδυασμό με το στίγμα της συσκευής εάν διαθέτει δέκτη GPS, ημερολόγιο, διαχείριση επαφών, παιχνίδια, RSS readers και πολλές άλλες. Όλες οι εφαρμογές όπως έχει ήδη αναφερθεί είναι γραμμένες σε Java και μπορούν να τρέχουν πολλές παράλληλα χωρίς να επηρεάζει η μία την άλλην.

Κεφάλαιο 2ο : Ανάλυση και ανάπτυξη μιας εφαρμογής Android

Μία εφαρμογή αποτελείται από ένα σύνολο αρχείων και φακέλων, δομημένα με συγκεκριμένη μορφή. Όλες οι εφαρμογές πρέπει να έχουν ένα μοναδικό όνομα πακέτου (package name id), το οποίο χρησιμοποιείται από το λειτουργικό σύστημα για την αναγνώριση της εφαρμογής. Μία εφαρμογή μπορεί να αποτελείται από ένα ή περισσότερα πακέτα (υπο-πακέτα), ανάλογα με την πολυπλοκότητα της εφαρμογής.

Ένα από τα βασικότερα δομικά στοιχεία της εφαρμογής είναι το αρχείο AndroidManifest.xml όπου περιέχονται πολλές χρήσιμες πληροφορίες σχετικά με την εφαρμογή και το σύστημα στο οποίο πρόκειται να τρέξει.
Επίσης, ένα project εμπεριέχει πολλά δομικά στοιχεία που καθορίζονται από το SDK, όπως οι Δραστηριότητες – Activities και η διεπαφή χρήστη – User Interface.

2.1 Περιγραφή του Android Manifest

Κάθε project εφαρμογής περιέχει ένα αρχείο στο οποίο βρίσκονται καταχωρημένες οι σημαντικότερες πληροφορίες της εφαρμογής, και το αρχείο αυτό ονομάζεται AndroidManifest.xml. Πρόκειται όπως λέει και το όνομα του για ένα αρχείο xml μέσα στο οποίο ο προγραμματιστής καταχωρεί τις σημαντικότερες πληροφορίες της εφαρμογής για χρήση από το λειτουργικό σύστημα. Κάποιες από αυτές τις πληροφορίες είναι:

- Το όνομα του πακέτου της εφαρμογής
- Το κανονικό της όνομα που φαίνεται στον χρήστη
- Η έκδοση των APIs που χρησιμοποιούνται
- Ο αριθμός έκδοσης της εφαρμογής
- Οι άδειες χρήσης που ζητάει η εφαρμογή
- Όλες οι δραστηριότητες, πάροχοι περιεχομένου, υπηρεσίες, κλπ. που περιέχει και χρησιμοποιεί η εφαρμογή.

Όπως αντιλαμβανόμαστε πρόκειται για πολύ σημαντικό αρχείο και αποτελεί κύριο δομικό στοιχείο κάθε εφαρμογής.

2.2 Περιγραφή των φακέλων SRC και RES

Δύο βασικοί φάκελοι ενός project είναι οι φάκελοι SRC και RES.

Στον φάκελο SRC (εκ του source) περιέχονται τα αρχεία κλάσης της Java όλων των δραστηριοτήτων – Activities, Content Providers, Services, 3rd party files. Ο φάκελος περιέχει όλα τα απαραίτητα πακέτα της εφαρμογής, που αποτελούνται από αρχεία γραμμένα σε Java και αποτελούν τον μοναδικό πηγαίο φάκελο στο project στον οποίο αποθηκεύονται τα αρχεία του κώδικα.

Στον φάκελο RES (εκ του resource) περιέχονται όλα τα αρχεία ήχου, εικόνας, κειμένου, layouts κλπ. που χρησιμοποιούν οι δραστηριότητες – Activities, που βρίσκονται στον φάκελο SRC. Αξίζει να σημειωθεί, ότι αναλόγως με τον τύπο του αρχείου καταχωρείται σε αντίστοιχο υποφάκελο. Οι περισσότερο γνωστοί και συνηθισμένοι υποφάκελοι του φάκελου RES είναι:

- Ο φάκελος Drawable, που περιέχει τα αρχεία εικόνας που χρησιμοποιεί το project (υποστηριζόμενες καταλήξεις : .png, .jpg, .gif).
- Ο φάκελος Layout, που περιέχει όλα τα αρχεία τύπου xml και ορίζουν τις διάφορες οθόνες που θα βλέπει ο τελικός χρήστης.
- Ο φάκελος Values, που περιέχει όλες τις τιμές και τους πόρους που χρησιμοποιεί και αποδίδει το project σε κάθε μία σταθερά.
- Ο φάκελος APIs, που περιέχει όλα τα διαθέσιμα στοιχεία για την έκδοση Android που χρησιμοποιεί το project.
- Ο φάκελος Libraries, που περιέχει όλες τις διαθέσιμες βιβλιοθήκες που χρησιμοποιεί το project.

2.3 Άλλα δομικά στοιχεία ενός project

Όλα τα δομικά στοιχεία ενός project, πρέπει να αναφέρονται ρητώς στο αρχείο AndroidManifest.xml. Παρακάτω αναλύεται τι αντιπροσωπεύει το κάθε ένα από αυτά.

- Υπηρεσίες – Services : Είναι οι λειτουργίες της εφαρμογής που τρέχουν στο παρασκήνιο και μπορούν να επιστρέψουν αποτελέσματα ακόμη και όταν η εφαρμογή δεν είναι στο προσκήνιο. Π.χ μια εφαρμογή για το ξυπνητήρι μπορεί μέσω μιας υπηρεσίας να συνεχίζει να είναι ενεργή, έστω και αν το κύριο παράθυρο της εφαρμογής δεν είναι στο προσκήνιο.
- Δραστηριότητες – Activities : Αποτελεί ένα από τα βασικότερα στοιχεία ενός project. Το Activity είναι μία οθόνη διεπαφής του χρήστη (GUI) και προβολής πληροφοριών. Ένα project έχει τόσα activities όσες και οι διαφορετικές οθόνες που εμφανίζονται στον χρήστη. Φυσικά, όλα τα activities συνδέονται και συνεργάζονται μεταξύ τους για να προσφέρουν στον χρήστη μια συνολική εμπειρία χρήσης της εφαρμογής.
- Προθέσεις – Intents : Τα intents είναι ο συνδετικός κρίκος μεταξύ των activities. Χρησιμοποιούνται για να εξασφαλίσουν την μετάβαση από το ένα activity σε ένα άλλο και για να ανταλλάξουν δεδομένα μεταξύ τους. Η ανταλλαγή των δεδομένων μπορεί να πραγματοποιηθεί είτε μεταξύ των activities μια εφαρμογής, είτε από την μία εφαρμογή στην άλλην. Π.χ μια εφαρμογή μπορεί να μας παραπέμπει σε μία ιστοσελίδα. Ένα intent θα εκκινήσει τον browser και ένα άλλο θα μας μεταφέρει στην ιστοσελίδα.
- Δέκτες Μετάδοσης – Broadcast Receivers : Τα Broadcast Receivers είναι ένα είδος υπηρεσίας που ενημερώνει το σύστημα Android ή τις υπόλοιπες εφαρμογές για κάποια γεγονότα που συμβαίνουν στο σύστημα. Έχουν διπλό σκοπό καθώς μπορούν να ενημερωθούν για κάποιο συμβάν από άλλες εφαρμογές αλλά και να ειδοποιήσουν τις υπόλοιπες εφαρμογές ή το σύστημα, για κάποιο συμβάν που τις ενεργοποίησε. Συνήθως δεν έχουν γραφικό περιβάλλον, αλλά μπορούν να προβάλουν ειδοποίηση στον χρήστη μέσω της μπάρας ειδοποιήσεων. Π.χ, η λήψη ενός γραπτού μηνύματος (sms) ενώ ακούμε μουσική. Το πρόγραμμα της μουσικής δεν θα διακοπεί, αλλά στην μπάρα ειδοποιήσεων θα φανεί ότι παραλήφθηκε ένα γραπτό μήνυμα.

- Πάροχος Περιεχομένου – Content Providers : Τα Content Providers κάνουν ανταλλαγή δεδομένων από μία εφαρμογή σε μία άλλη. Έχει πιο δύσκολη και πιο σύνθετη λειτουργία από ένα intent που χρησιμοποιείται επίσης για ανταλλαγή δεδομένων. Τα Content Providers διαχειρίζονται συγκεκριμένα δεδομένα ενός project, τα οποία έχει ορίσει ο προγραμματιστής κατά την δημιουργία του. Π.χ, η άντληση των επαφών μίας συσκευής και ενημέρωση μίας βάσης δεδομένων SQL.

2.4 Πολιτική χρήσης ελαχίστων δικαιωμάτων

Από την στιγμή που μία εφαρμογή θα εγκατασταθεί σε μία συσκευή, λειτουργεί αποκλειστικά στην δική της εικονική μηχανή, η οποία αποτελεί και το πλαίσιο ασφαλείας της εφαρμογής (sandbox).

Το Android είναι ένα λειτουργικό σύστημα πολλών χρηστών στο οποίο:

- Η κάθε εφαρμογή αντιμετωπίζεται σαν ένας διαφορετικός χρήστης.
- Κάθε εφαρμογή τρέχει στην δική της εικονική μηχανή (Virtual Machine) απομονωμένη από τις υπόλοιπες εφαρμογές. Η εικονική μηχανή εκκινείτε μόλις ζητηθεί από το σύστημα και κλείνει είτε γιατί δεν χρησιμοποιείτε, είτε γιατί το σύστημα θέλει να απελευθερώσει τους πόρους της μνήμης για χρήση από άλλη εφαρμογή.
- Σε κάθε εφαρμογή αποδίδεται ένας μοναδικός αριθμός ID, ο οποίος είναι άγνωστος στην εφαρμογή. Το λειτουργικό σύστημα αναθέτει συγκεκριμένες άδειες χρήσης στα αρχεία της εφαρμογής και μόνο η εφαρμογή με το σωστό ID μπορεί να έχει πρόσβαση σε αυτά.

Με αυτό τον τρόπο, το Android κατάφερε να χρησιμοποιήσει την πολιτική χρήσης ελαχίστων δικαιωμάτων ως μέθοδος ασφάλειας. Η κάθε εφαρμογή έχει πρόσβαση μέσω του AndroidManifest μόνο σε όσους πόρους συστήματος χρειάζεται. Οι πόροι και τα δικαιώματα που απαιτούνται από μία εφαρμογή γνωστοποιούνται στον χρήστη την στιγμή της εγκατάστασης της και ο χρήστης μπορεί να επιλέξει να μην εγκαταστήσει μία εφαρμογή εφόσον δε συμφωνεί να της παρέχει πρόσβαση στους ζητούμενους πόρους.

2.5 Εργαλεία ανάπτυξης εφαρμογής

Τα εργαλεία για την ανάπτυξη εφαρμογής Android και ποικίλουν ανάλογα με τον τρόπο σκέψης κάθε προγραμματιστή. Μερικά από τα εργαλεία που χρησιμοποιήθηκαν σε αυτή την εφαρμογή αναλύονται παρακάτω.

2.5.1 Android SDK

Το Android Software Developer Kit (SDK) είναι μια συλλογή βιβλιοθηκών και εργαλείων που καθιστούν εφικτή την ανάπτυξη εφαρμογών σε περιβάλλον Android. Η έκδοση που είναι διαθέσιμη αυτή την στιγμή από την Google είναι η 23.0.2 (Ιούλιος 2014) και μερικά από τα εργαλεία που περιλαμβάνει είναι ένας debugger και ένας εξομοιωτής QEMY (Quick EMUlator) για εικονική λειτουργία συσκευής σε Η/Υ.

Εικόνα 4 Android SDK

2.5.2 Eclipse

Ο προγραμματισμός στο Android βασίζεται στην γλώσσα Java και ο κάθε προγραμματιστής μπορεί να χρησιμοποιήσει έναν οποιονδήποτε text editor για να

γράψει κώδικα για να επεξεργαστεί τα αρχεία *.Java και *.XML και μετέπειτα να τα κάνει compile μέσω γραμμής εντολών χρησιμοποιώντας το JDK (Java Development Kit). Ο συγκεκριμένος τρόπος ανάπτυξης δεν είναι ιδιαίτερα φιλικός στον χρήστη γιαυτό συνίσταται η χρήση ενός IDE (Integrated Development Environment) που να υποστηρίζει Java, όπως το Eclipse ή το Netbeans.

Η Google υποστηρίζει επισήμως το Eclipse και έχει αναπτύξει ειδικά για αυτό το ADT (Android Development Tool) plugin, το οποίο παρέχει σύνδεση με το Android SDK με όλες τις δυνατότητες που περιλαμβάνει αυτό. Επίσης το plugin παρέχει σύνδεση με τον AVD (Android Virtual Device) Manager, για διαχείριση και εκκίνηση μέσω γραφικού περιβάλλοντος, εικονικών συσκευών Android για δοκιμές και debugging των εφαρμογών.

Εικόνα 5 Eclipse

2.6 Ανάλυση ανάπτυξης εφαρμογής

Η ανάπτυξη εφαρμογών στο Android είναι μία σύνθετη και ιδιαίτερα χρονοβόρα διαδικασία, η οποία βασίζεται σε 3 βασικά στάδια. Πρώτον στην συγκέντρωση και εγκατάσταση των απαραίτητων λογισμικών, δεύτερον στην μελέτη και ανάπτυξη του κώδικα, και τρίτον στην αποσφαλμάτωση της εφαρμογής, στην δοκιμή της και δημοσίευση της στο ευρύ κοινό.

2.6.1 Συγκέντρωση και εγκατάσταση λογισμικού

Σε αυτό το στάδιο της ανάπτυξης εφαρμογής, ο προγραμματιστής πρέπει να στήσει το περιβάλλον εργασίας στο οποίο θα γίνει ο σχεδιασμός, η ανάπτυξη, ο έλεγχος και η λειτουργία της εφαρμογής. Υπάρχει πληθώρα εργαλείων που μπορεί να επιλέξει, οπότε μπορεί να επιλέξει όποιο περιβάλλον ανάπτυξης (IDE) τον εξυπηρετεί καλύτερα και να χρησιμοποιήσει όλα τα εργαλεία του Android SDK που του προσφέρει η Google.

Έπειτα, πρέπει να δημιουργήσει έναν αριθμό από εικονικές συσκευές στην διαχείριση εικονικών συσκευών (Android Virtual Devices – AVD) για να δοκιμάσει την λειτουργία της εφαρμογής σε διαφορετικές πραγματικές συνθήκες λειτουργίας και σε διαφορετικές εκδόσεις λογισμικού Android. Ιδανικά ο developer θα διαθέτει έναν αριθμό διαφορετικών φυσικών συσκευών ώστε να δοκιμάσει ο ίδιος πως συμπεριφέρεται η εφαρμογή του σε κάθε περίπτωση, όμως αυτή η πρακτική μπορεί να αποδειχθεί πολυδάπανη και χρονοβόρα.

2.6.2 Ανάπτυξη Κώδικα Εφαρμογής

Είναι η περισσότερο πολύπλοκη και χρονοβόρα διαδικασία για κάθε προγραμματιστή. Σε αυτό το στάδιο, ο προγραμματιστής πρέπει να αποφασίσει για τις δυνατότητες και το περιεχόμενο που θα περιλαμβάνει η εφαρμογή, να εντοπίσει ποιες από αυτές τις δυνατότητες είναι εφικτές και ποιες θέλουν παραπάνω έρευνα για να προστεθούν στο μέλλον, να σχεδιάσει το layout με γνώμονα την λειτουργικότητα και να αποφύγει υπερβολές στο σχεδιασμό, και τέλος να δέσει αρμονικά τον κώδικα με το layout για να φέρει το επιθυμητό αποτέλεσμα. Η διαδικασία ξεκινάει με ένα νέο Project το οποίο θα περιέχει τον πηγαίο κώδικα, τις εικόνες, τα κείμενα και γενικά ότι χρειάζεται η εφαρμογή για να τρέξει ως οφείλει. Στο project του ο προγραμματιστής θα πρέπει να φροντίσει ώστε η εφαρμογή του να είναι τακτοποιημένη και ο κώδικας του ευανάγνωστος ώστε να ακολουθήσει η διαδικασία της αποσφαλμάτωσης – Debugging.

2.6.3 Αποσφαλμάτωση, Δοκιμή και Δημοσίευση της Εφαρμογής

Η διαδικασία της αποσφαλμάτωσης – Debugging είναι εξίσου κρίσιμη και μερικές φορές και εξίσου χρονοβόρα με την διαδικασία ανάπτυξης του πηγαίου κώδικα της εφαρμογής. Αποτελείται από τέσσερα επί μέρους στάδια.

Το πρώτο στάδιο αφορά το αρχικό στήσιμο της εφαρμογής και η λειτουργία αυτής σε debug mode. Για να γίνει το compile της εφαρμογής φυσικά τα περισσότερα περιβάλλοντα ανάπτυξης (IDE) προϋποθέτουν ότι ο κώδικας δεν έχει κανένα συντακτικό λάθος, αλλιώς ειδοποιούν τον χρήστη να τα διορθώσει. Αφού γίνει το compile η εφαρμογή μπορεί να δοκιμαστεί είτε σε εικονική συσκευή μέσω του AVD Manager, είτε απευθείας σε φυσική συσκευή μέσω ADB push εντολής.

Στο δεύτερο στάδιο ο προγραμματιστής καλείτε να αντιμετωπίσει τα λειτουργικά και αισθητικά προβλήματα της εφαρμογής του, πρώτα εντοπίζοντας τα στην λειτουργία της συσκευής και μετά διορθώνοντας τα κομμάτια του κώδικα που δημιουργούν τα σφάλματα. Το κύριο εργαλείο που κάνει αυτή τη διαδικασία εφικτή είναι το "LogCat" το οποίο μας επιστρέφει το stack trace του κώδικα στο σημείο εκείνο που συνέβη το σφάλμα. Υπάρχουν φυσικά και πολλά άλλα εργαλεία που κάνουν την ίδια ανάλυση.

Στο τρίτο στάδιο ο προγραμματιστής αφού έχει τελειώσει την αποσφαλμάτωση (debugging) επιστρέφει στο βήμα ένα, δηλαδή στο compile και τη δοκιμή της εφαρμογής σε εικονική ή φυσική συσκευή ώστε να διαπιστώσει τα αποτελέσματα του δεύτερου βήματος, της αποσφαλμάτωσης. Η διαδικασία της debug μπορεί να παρομοιαστεί σαν ένας βρόγχος (loop), που επαναλαμβάνεται συνεχώς μέχρι εντοπιστούν και να διορθωθούν όλα τα σφάλματα της εφαρμογής, γεγονός που κάνει ιδιαίτερα χρονοβόρα την διαδικασία.

Στο τέταρτο και τελευταίο στάδιο, ο προγραμματιστής έχει να κάνει μερικές τελευταίες κινήσεις. Πρώτον πρέπει να έχει διορθώσει όλα τα σφάλματα που προέκυψαν από την διαδικασία debug, να κάνει τις τελευταίες ρυθμίσεις και tweaks της εφαρμογής, και να κάνει το τελικό compile της εφαρμογής σε κανονική λειτουργία αυτή τη φορά και όχι debug. Στη συνέχεια ακολουθεί η διάθεση της εφαρμογής με το μέσω της επιλογής του developer.

Κεφάλαιο 3ο : Container Tracker

3.1 Ανάλυση ιδέας υλοποίησης εφαρμογής

Ο αρχικός στόχος της συγκεκριμένης διπλωματικής εργασίας ήταν να δημιουργηθεί μία εφαρμογή η οποία θα ήταν χρήσιμη και άκρως εφαρμόσιμη σε εταιρίες διακίνησης εμπορευματοκιβωτίων. Η σύλληψη της ιδέας για την συγκεκριμένη εφαρμογή ξεκίνησε από το γεγονός ότι πολλές εταιρίες στοχεύουν στην παρακολούθηση των φορτίων τους είτε για λόγους ασφαλείας, είτε για υπολογισμό χρόνου, κόστους, είτε ακόμα και για ενημέρωση των πελατών τους.

Συνήθως για να συλλέξει κάποιος αυτές τις πληροφορίες απαιτείται η αναζήτηση στοιχείων σε διάφορους πίνακες εάν βρίσκονται σε ηλεκτρονική μορφή, αλλιώς πρέπει να βρουν άλλους τρόπους ενημέρωσης που είναι ιδιαίτερα δύσκολοι και χρονοβόροι, ειδικά εάν υποθέσουμε ότι ένα container κάνει διηπειρωτικό ταξίδι ή ταξίδι μέσω θάλασσας.

Η ιδέα για την απεικόνιση των δρομολογίων προφανώς δεν είναι κάτι νέο ή πρωτοποριακό. Υπάρχουν και άλλες εφαρμογές που έχουν αναπτυχθεί σε παρόμοια φιλοσοφία. Πρόκειται για εφαρμογές που έχουν αναπτυχθεί εξολοκλήρου και ειδικά για συγκεκριμένη εταιρία.

Η διαφορετικότητα της συγκεκριμένης εφαρμογής έγκειται στο γεγονός ότι αποτελεί έναν γενικότερο κανόνα, όπου θα μπορούσε να εξατομικευθεί σύμφωνα με τα πρότυπα και τις ανάγκες της εκάστοτε εταιρίας. Επίσης, έχει την δυνατότητα απεικόνισης της διαδρομής σε 3D χάρτη. Ωστόσο, η εφαρμογή δε βασίζεται στην προβολή κάποιας ιστοσελίδας σε φορητή συσκευή Android αλλά στην δυνατότητα προβολής των δεδομένων μέσα από ειδική εφαρμογή αποκλειστικά υλοποιημένη για λειτουργικό Android.

Σε κάποιο σενάριο χρήσης, τα containers θα μπορούσαν να φέρουν και εξοπλισμό GPS, ώστε να μπορούν να μεταδίδουν την τοποθεσία τους. Η εφαρμογή θα λάμβανε τις συντεταγμένες και θα μπορούσε να απεικονίσει την τοποθεσία του στον χάρτη, τις καιρικές συνθήκες της περιοχής, να υπολογίσει την αναμενόμενη ώρα άφιξης στον προορισμό και άλλες πολλές χρήσιμες πληροφορίες.

3.2 Περιγραφή εφαρμογής

Στα πλαίσια της συγκεκριμένης διπλωματικής αναπτύχθηκε μία εφαρμογή, η οποία παρέχει τη δυνατότητα αναζήτησης και εμφάνισης δρομολογίων containers. Η γενική φιλοσοφία είναι ότι ο χρήστης της φορητής συσκευής Android μπορεί μέσω αυτής να αναζητήσει διάφορες πληροφορίες για το container. Οι πληροφορίες είναι διαθέσιμες μέσω διαδικτυακών υπηρεσιών. Η εφαρμογή δίνει την δυνατότητα αναζήτησης μέσω διάφορων φίλτρων που θα ορίσει ο χρήστης. Ο SQL Server απαντάει στην συσκευή με όλες τις διαθέσιμες πληροφορίες. Ο χρήστης μετά έχει την δυνατότητα απεικόνισης στον χάρτη και on time ενημέρωσης για το container.

Οι πληροφορίες που είναι διαθέσιμες για ένα container είναι:

- η γεωγραφική του τοποθεσία
- οι καιρικές συνθήκες σε εκείνη την γεωγραφική τοποθεσία
- οι καιρικές συνθήκες σε τοποθεσία άφιξης
- η λεπτομερής ημερομηνία αναχώρησης καθώς και της άφιξης
- τυχόν ενδιάμεσες στάσεις
- τα μέσα μεταφοράς με τα οποία μετακινείται το container
- το περιεχόμενο του container
- γραφική απεικόνιση σε χάρτη του δρομολογίου του container
- ειδοποίηση για τυχόν καθυστερήσεις άφιξης στον προορισμό

Η υλοποίηση της εφαρμογής χωρίζεται σε τρία τμήματα. Το πρώτο τμήμα είναι η δημιουργία μίας βάσης δεδομένων, όπου υπάρχουν όλες οι πληροφορίες σχετικά με τα δρομολόγια. Το δεύτερο τμήμα είναι η δημιουργία σύνδεσης εφαρμογής και βάσης δεδομένων. Το τρίτο τμήμα είναι η υλοποίηση μια εφαρμογής για φορητές συσκευές Android, η οποία συνδέεται στην βάση και αναζητάει πληροφορίες.

3.3 Περιγραφή και ανάλυση βάσης δεδομένων εφαρμογής

Μία βάση δεδομένων είναι μία οργανωμένη συλλογή των δεδομένων, δηλαδή ένα ηλεκτρονικό σύστημα που επιτρέπει την πρόσβαση, διαχείριση και ενημέρωση των δεδομένων. Η βάση δεδομένων είναι ένα από τα θεμέλια του επιχειρηματικού κόσμου, καθώς η ικανότητα της οργάνωσης, της επεξεργασίας και διαχείρισης των

πληροφοριών με ένα καλά δομημένο τρόπο, είναι το κλειδί της αποδοτικότητας για πολλούς τομείς της σύγχρονης κοινωνίας.

Εικόνα 6 phpMyAdmin

Οι βάσεις δεδομένων διαιρείται σε λογικές μονάδες, που ονομάζονται πίνακες και σχετίζονται μεταξύ τους μέσα από τη βάση δεδομένων. Είναι μία μέθοδος που χρησιμοποιείται για τη διάρθρωση των δεδομένων που χειρίζονται οι σχέσεις, οι οποίες είναι πλέγμα που μοιάζει με μαθηματικές δομές που αποτελούνται από στήλες και γραμμές. Ένας πίνακας είναι μία συλλογή από εγγραφές και κάθε εγγραφή σε έναν πίνακα περιέχει τα ίδια πεδία. Τα περιεχόμενα ενός πίνακα μπορεί να υποστούν μόνιμη αποθήκευση για μελλοντική χρήση.

Στην εφαρμογή έχει δημιουργηθεί μία βάση δεδομένων «Package_Tracker» με 3πίνακες.

Ο πρώτος πίνακας «Package» έχει τα στοιχεία δρομολογίου του Container. Πρωτεύων κλειδί είναι το “ID” που είναι ο κωδικός του δρομολογίου. Περιέχει όλες τις πληροφορίες του δρομολογίου, όπως τις συντεταγμένες, τα μέσα με τα οποία μεταφέρεται το Container, εάν εκπίπτει σε κάποια ειδική κατηγορία και τις ημερομηνίες αναχώρησης και άφιξης.

Στήλη	Τύπος	Περιγραφή Πεδίου
ID	Char(10)	Κωδικός Δρομολογίου Container
Origin_Lat	Double	Γεωγραφικό Πλάτος Σημείου Αναχώρησης
Origin_Long	Double	Γεωγραφικό Μήκος Σημείου Αναχώρησης
Origin_Address	VarChar(150)	Διεύθυνση Σημείου Αναχώρησης
Destination_Lat	Double	Γεωγραφικό Πλάτος Σημείου Προορισμού
Destination_long	Double	Γεωγραφικό Μήκος Σημείου Προορισμού
Destination_Address	VarChar(150)	Διεύθυνση Σημείου Προορισμού
Current_Lat	Double	Γεωγραφικό Πλάτος Τωρινής Τοποθεσίας
Current_Long	Double	Γεωγραφικό Μήκος Τωρινής Τοποθεσίας
Current_Address	VarChar(150)	Διεύθυνση Τωρινής Τοποθεσίας
Depart_Time	Datetime	Ημερομηνία και Ώρα Αναχώρησης
Arrive_Time	Datetime	Αναμενόμενη Ημερομηνία και Ώρα Άφιξης
Medium	VarChar(100)	Οχήματα Μεταφοράς Container
Info	VarChar(250)	Άλλες πληροφορίες για το Container

Πίνακας 5 Package Schema

Ο δεύτερος πίνακας «Stopover» αποθηκεύει τις στάσεις που κάνει κάθε Container. Πρωτεύων κλειδί είναι το “Package_ID” που αντιστοιχείται με το αναγνωριστικό “ID” του δρομολογίου. Περιέχει πληροφορίες για τις συντεταγμένες της κάθε στάσης που έκανε το Container καθώς και την ώρα που έμεινε εκεί.

Στήλη	Τύπος	Περιγραφή Πεδίου
Package_ID	Char(10)	Κωδικός Container
Origin_Lat	Double	Γεωγραφικό Πλάτος Σημείου Στάσης
Origin_Long	Double	Γεωγραφικό Μήκος Σημείου Στάσης
Origin_Address	VarChar(150)	Διεύθυνση Σημείου Στάσης
Depart_Time	Datetime	Ημερομηνία και Ώρα Αναχώρησης
Arrive_Time	Datetime	Ημερομηνία και Ώρα Άφιξης

Πίνακας 6 Stopover Schema

Ο τρίτος πίνακας «History» αποθηκεύει όλο το ιστορικό για τα δρομολόγια όλων των Containers. Πρωτεύων κλειδί είναι το "Package_ID" που αντιστοιχείται με το αναγνωριστικό "ID" του δρομολογίου. Σε αυτό το σημείο, αξίζει να αναφερθεί ότι στον πίνακα «History» έχει προστεθεί μία στήλη "Mobile_ID" ώστε να δίνεται η δυνατότητα μόνο σε συγκεκριμένες συσκευές να έχουν πληροφορίες για ένα Container και αυτό κυρίως για λόγους ασφαλείας.

Στήλη	Τύπος	Περιγραφή Πεδίου
Mobile_ID	VarChar(10)	Αναγνωριστικό Συσκευής Android
Package_ID	Char(10)	Κωδικός Container

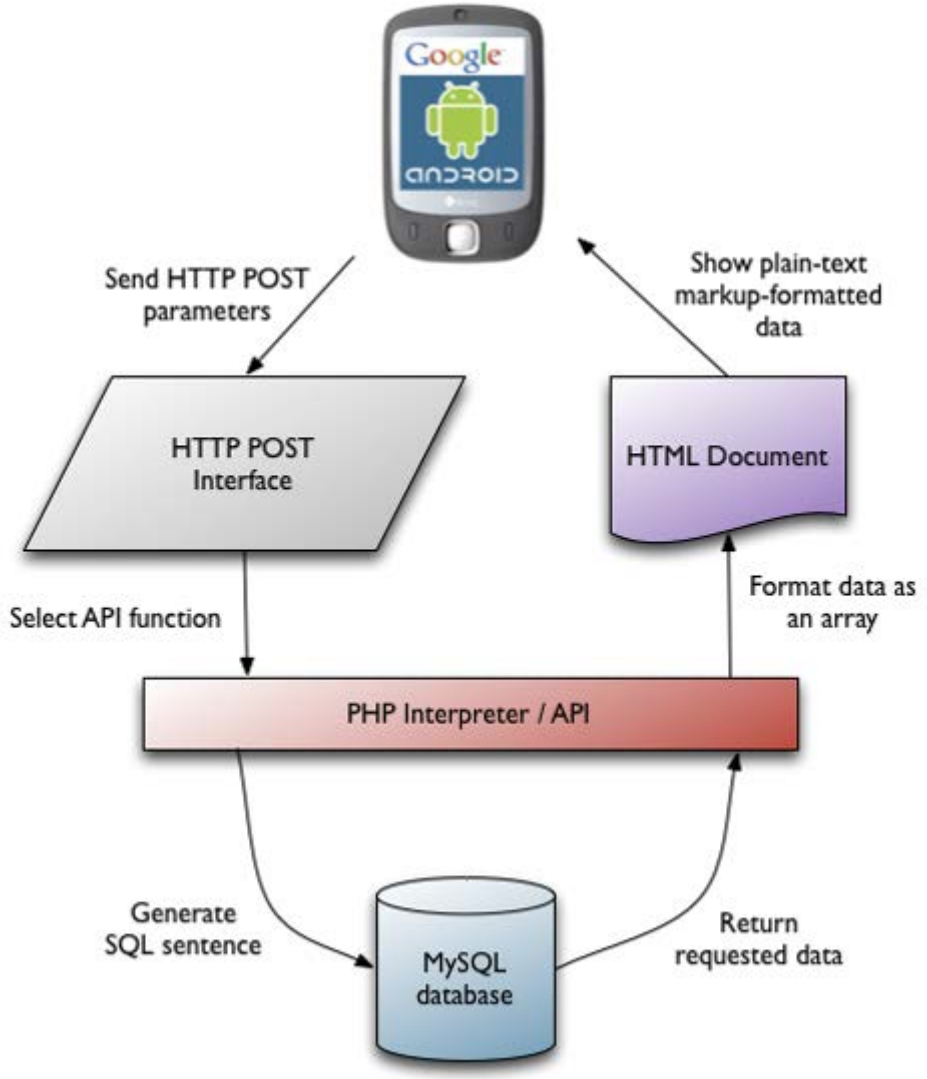
Πίνακας 7 History Schema

3.4 Περιγραφή και ανάλυση συνδεσιμότητας μεταξύ βάσης δεδομένων και εφαρμογής

Για να συνδεθεί μία βάση δεδομένων με μία εφαρμογή Android υπάρχουν 2τρόποι. Ο πρώτος και πιο λανθασμένος είναι η απευθείας σύνδεση. Αυτό πραγματοποιείται μέσω JDBC (Java DataBase Connectivity), προϋποθέτοντας την ύπαρξη του JDBC Driver στον Server. Ο συγκεκριμένος τρόπος όχι μόνο δεν είναι δυνατός για όλες τις βάσεις αλλά είναι και λάθος. Οι βασικοί λόγοι που η συγκεκριμένη τεχνική θεωρείται λάθος είναι η ασφάλεια και η απόδοση. Επίσης, είναι λάθος

προγραμματιστική πρακτική καθώς όπως θα δούμε στην συνέχεια, η εφαρμογή διαχωρίζεται καλύτερα σε client - server. Ο δεύτερος τρόπος, όπου χρησιμοποιήθηκε και σε αυτή την εφαρμογή, είναι μέσω Web Services, ο οποίος ενδείκνυται γενικά καθώς είναι προσβάσιμος ανεξαρτήτως της εφαρμογής ή της συσκευής. Για χρόνια το SOAP (Simple Object Access Protocol) ήταν και εξακολουθεί να είναι το πιο δημοφιλές web service. Δυστυχώς όμως στο android υπάρχει υποτυπώδης υποστήριξη σε αυτό μιας και η Google ,όπως και άλλοι κολοσσοί της πληροφορικής, έχουν στραφεί προς το ταχύτατα αναδυόμενο REST (Represantational State Transfer) . Η δύναμη του REST είναι η απλότητα, καθώς βασίζεται στα βασικά συστατικά του internet (HTTP, URL). Επίσης είναι ευέλικτο καθώς υποστηρίζει διάφορους τύπους δεδομένων (MIME) όπως XML, JSON αλλά και plain text.

Στην παρακάτω εικόνα, απεικονίζεται η λογική που ακολουθήθηκε για την σύνδεση μεταξύ βάσης δεδομένων και εφαρμογής.



Εικόνα 7 Html Post

Για να "διαβάσει" η εφαρμογή την απάντηση της βάσης δεδομένων είναι αναγκαία η ύπαρξη μίας MarkUp Language. Στην περίπτωση μας χρησιμοποιήθηκε η JSON – JavaScript Object Notation.

Εικόνα 8 JSON

Στην εφαρμογή υπάρχει ένα αρχείο Java, όπου δηλώνεται η διεύθυνση IP του Server, για να γίνει η αρχική σύνδεση στην βάση δεδομένων.

ServerConnection.java

```
package my.application.containertracker;

public class ServerConnection {
    private static String server_url = "http://192.168.43.246/packagetracker/";

    public static String getServerUrl(){
        return server_url;
    }
}
```

Εικόνα 9 Server Connection

Επίσης υπάρχει και ένας «μεταφραστής» JSONParser, για την ανταλλαγή των πληροφοριών.

JSONParser.java

```
package my.application.containertracker;

import java.io.BufferedReader;

public class JSONParser {

    static InputStream is = null;
    static JSONObject jObj = null;
    static String json = "";

    // constructor
    public JSONParser() {

    }

    // function get json from url
    // by making HTTP POST or GET mehtod
    public JSONObject makeHttpRequest(String url, String method,
            List<NameValuePair> params) {

        // Making HTTP request
        try {

            // check for request method
            if(method == "POST"){
                // request method is POST
                // defaultHttpClient
                DefaultHttpClient httpClient = new DefaultHttpClient();
                HttpPost httpPost = new HttpPost(url);
                httpPost.setEntity(new UrlEncodedFormEntity(params));

                HttpResponse httpResponse = httpClient.execute(httpPost);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();

            }else if(method == "GET"){
                // request method is GET
                DefaultHttpClient httpClient = new DefaultHttpClient();
                String paramString = URLEncodedUtils.format(params, "utf-8");
                url += "?" + paramString;
                HttpGet httpGet = new HttpGet(url);

                HttpResponse httpResponse = httpClient.execute(httpGet);
                HttpEntity httpEntity = httpResponse.getEntity();
                is = httpEntity.getContent();
            }
```

Εικόνα 10 JSONParser

3.5 Περιγραφή και ανάλυση εφαρμογής

Η υλοποίησή της έγινε με τις βιβλιοθήκες του Android κατά βάση και τη χρήση κάποιων πρόσθετων για να υποστηριχθούν ιδιαίτερες υλοποιήσεις, όπως είναι η

εμφάνιση του καιρού στον τελικό προορισμό και την στιγμή που κάνουμε την αναζήτηση, ειδικά διαμορφωμένος χάρτης για να φαίνονται οι ακτοπλοϊκές γραμμές.

3.5.1 Βασική δομή εφαρμογής

Η δομή μιας εφαρμογής Android αποτελείται από τους παρακάτω βασικούς φακέλους. Η βασική υλοποίηση του κώδικα της εφαρμογής βρίσκεται μέσα στον φάκελο src. Ο φάκελος gen περιέχει βασικά java αρχεία που δημιουργούνται αυτόματα. Το βασικότερο είναι το αρχείο R.java το οποίο δημιουργείται αυτόματα από τις μεταβλητές των resources της εφαρμογής και κρατάει τις τιμές για αυτές. Ο φάκελος libs περιέχει όλες τις πρόσθετες βιβλιοθήκες που χρησιμοποιούνται στην εφαρμογή. Τέλος ο φάκελος res περιέχει όλα τα πρόσθετα resources της εφαρμογής. Περιέχει φακέλους οι οποίοι περιέχουν τα γραφικά της εφαρμογής, εικόνες, βίντεο, αρχεία xml που καθορίζουν τη διάταξη και τα αντικείμενα που περιέχει μέσα της η κάθε οθόνη, τα στοιχεία των μενού των οθονών, διάφορες τιμές για στυλ, χρώματα, διαστάσεις, κείμενα της εφαρμογής. Πιο βασικός φάκελος είναι ο layout, ο οποίος περιέχει αρχεία τύπου xml, τα οποία περιγράφουν τη γραφική απεικόνιση με τα αντικείμενά της για κάθε οθόνη. Σε κάθε κλάση Activit, που είναι η κλάση που αντιπροσωπεύει την οθόνη μέσω της κλήσης της συνάρτησης setContentView, δένουμε την κλάση της οθόνης με το xml αρχείο που την περιγράφει. Τέλος, εκτός των φακέλων το πιο σημαντικό αρχείο, το οποίο αναφέρθηκε και προτύτερα, είναι το AndroidManifest.xml το οποίο περιέχει διάφορες παραμέτρους για την εφαρμογή. Παράμετροι που περιέχει το συγκεκριμένο αρχείο είναι η έκδοση του λειτουργικού που υποστηρίζει η εφαρμογή, το θέμα του στυλ που χρησιμοποιεί, ποια είναι η βασική οθόνη εκκίνησης της εφαρμογής, δηλώσεις για όλες τις οθόνες, καθώς και τα δικαιώματα που θέλει η εφαρμογή στη συσκευή, όπως για παράδειγμα τη διάθεση του δικτύου και του Internet για τη συγκεκριμένη εφαρμογή.


```
<uses-permission android:name="my.application.containertracker.permission.MAPS_RECEIVE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Εικόνα 11 Παράδειγμα δικαιωμάτων εφαρμογής

3.5.2 Βασικές οντότητες εφαρμογής

Το κύριο κομμάτι ανάπτυξης της εφαρμογής είναι ο κώδικας που αναπτύχθηκε και βρίσκεται στον φάκελο src. Δημιουργήθηκε ένα πακέτο με το όνομα my.application.containertracker και μέσα σε αυτό συμπεριλήφθηκαν όλες οι κλάσεις της εφαρμογής.

Όσον αφορά την υλοποίηση του κώδικα, έχει δημιουργηθεί ένα γενικό αντικείμενο που χρησιμοποιείται για την κλήση και σύνδεση της βάσης δεδομένων, ένα αντικείμενο για κάθε οντότητα της εφαρμογής, ένα αντικείμενο για κάθε οντότητα του πεδίου εύρεσης container, ένα αντικείμενο για την διαχείριση του χάρτη και ένα αντικείμενο για κάθε οθόνη της εφαρμογής.

Το κύριο Activity της εφαρμογής είναι “MyContainersActivity” που ελέγχει εάν υπάρχει δίκτυο, εάν έχει γίνει σωστά η σύνδεση με την βάση δεδομένων, εάν έχουν φορτώσει οι χάρτες και περιμένει το αίτημα του χρήστη για να καλέσει το κατάλληλο activity.

Το “SearchActivity” δίνει την δυνατότητα στον χρήστη να πραγματοποιήσει μία αναζήτηση με γνώμονα το “PackageID”, “Origin”, “Destination”, “Stopover”. Όταν ο χρήστης εισάγει μία από αυτές τις τιμές, καλείται το “SearchResultsActivity”, που κάνει ένα αίτημα «HTTP_Request» στην βάση δεδομένων αναζητώντας μία από αυτές τις τιμές. Εάν βρει περισσότερα από ένα δρομολόγια, τα εμφανίζει στον χρήστη, καλώντας τον να επιλέξει ένα. Εδώ πρέπει να σημειωθεί ότι ανεξαρτήτως του φίλτρου που επέλεξε ο χρήστης, η αναζήτηση στην βάση δεδομένων γίνεται πάντα με το “PackageID”. Μόλις επιλεχθεί ένα, καλείται το επόμενο Activity, το “MapActivity”, που κάνει 2^{ο} έλεγχο για την ύπαρξη των χαρτών και τοποθετεί το στίγμα στον χάρτη σύμφωνα με τις συντεταγμένες που έλαβε. Τοποθετεί επίσης Flags, υποδεικνύοντας την αφετηρία και τον προορισμό και απεικονίζει με μία γραμμή την διαδρομή που έχει ακολουθηθεί μέχρι στιγμής. Δίνεται εκ νέου η δυνατότητα στον χρήστη να αντλήσει νέες πληροφορίες με την ύπαρξη button.

Το “InfoActivity” στέλνει ξανά «HTTP Request» στην βάση δεδομένων ζητώντας να σταλθούν όλες οι πληροφορίες για το επιλεγμένο “PackageID”. Επίσης εισάγει και τις τιμές από ένα άλλο πακέτο, το

my.application.containertracker.weatherapp, που είναι οι πληροφορίες των καιρικών συνθηκών.

Το πακέτο my.application.containertracker.weatherapp συνδέεται στην ιστοσελίδα http://openweather.org και αντλεί πληροφορίες για την θερμοκρασία, την ατμοσφαιρική πίεση, την υγρασία, την κατεύθυνση και την ένταση των ανέμων για την τωρινή τοποθεσία του container αλλά και για το σημείο προορισμού.

Για την οντότητα των στάσεων, έχει κατασκευαστεί το "StopoverActivity", που μέσω νέου request, καλεί το stopovers.php για να αντλήσει πληροφορίες για τα σημεία στα οποία σταμάτησε το container

Για την οντότητα του ιστορικού, έχει κατασκευαστεί το "HistoryActivity", που μέσω νέου request, καλεί το gethistory.php για να αντλήσει πληροφορίες. Ο χρήστης καλείται να βάλει τον αναγνωριστικό κωδικό του container που επιθυμεί. Με αυτό το "PackageID" γίνεται αναζήτηση στην βάση και στέλνει πίσω τα αποτελέσματα.

Σε αυτό το σημείο πρέπει να προστεθεί, ότι το "HistoryActivity", ζητάει από την συσκευή Android να της δώσει το "MobileID", δηλαδή το IMEI – International Mobile Equipment Identity, που αποτελεί έναν μοναδικό αριθμό για κάθε συσκευή. Αυτός ο αριθμός συγκρίνεται με το "MobileID" και αν ταιριάζουν, τότε μόνο εμφανίζεται το δρομολόγιο στον χρήστη. Φυσικά, στην εφαρμογή δεν είναι απαραίτητο ο κάθε χρήστης να δημιουργήσει ατομικό λογαριασμό, αλλά για λόγους προστασίας δεδομένων, ο διαχειριστής της Βάσης Δεδομένων σε συμφωνία με τον πελάτη – χρήστη μπορεί να ¨αποκρύψει¨ ένα δρομολογίο και να εμφανίζεται μόνο σε συγκεκριμένες συσκευές. Εάν η τιμή "MobileID" είναι κενή, το δρομολόγιο είναι εμφανές σε όλους τους χρήστες.

3.5.3 Βασικές οθόνες εφαρμογής

Όσον αφορά τις οθόνες της εφαρμογής της φορητής συσκευής, για κάθε οθόνη έχει δημιουργηθεί μια χωριστή κλάση τύπου Activity, η οποία δένεται με ένα xml αρχείο στο φάκελο res/layout που περιγράφει τη δομή της.

Η αρχική οθόνη της εφαρμογής καλωσορίζει τον χρήστη στην εφαρμογή και τον καλεί να πατήσει το κουμπί «Continue» για να μπει στο κυρίως menu. Το κυρίως menu αποτελείται από 3 buttons.

Το 1ο button: “Track your Container”, καλεί το Search Activity, που περιγράφηκε παραπάνω και έχει 4 edit texts, τα PackageID text, Origin text, Destination text και Stopover text και ένα button, το Close button. Όποιο από αυτό συμπληρωθεί, μετατρέπεται σε string array και συγκρίνεται με τις τιμές που υπάρχουν στο πίνακα «Package» της βάσης δεδομένων «Package_Tracker». Μέσω του πρωτεύοντος κλειδιού “ID” που είναι ο κωδικός του δρομολογίου επιστρέφει στην οθόνη του χρήστη την λέξη “Success” και τις πληροφορίες του δρομολογίου. Εάν δεν βρεθεί κάποια αντιστοιχία μεταξύ του string array και των τιμών της ΒΔ, εμφανίζεται αντίστοιχο μήνυμα και ο χρήστης καλείται να συμπληρώσει άλλο πεδίο αναζήτησης. Μόλις η αναζήτηση είναι επιτυχής, μεταφερόμαστε στην οθόνη Search_Result όπου δείχνει όλα τα IDs δρομολογίων. Όταν επιλεχθεί ένα Package_ID εμφανίζεται ένα μήνυμα “Calculating Directions” και μεταφερόμαστε στο Map_Layout όπου απεικονίζεται το δρομολόγιο στον χάρτη, με δυνατότητες zoom in & zoom out και περιήγησης στον χάρτη. Στο κάτω μέρος της εικόνας υπάρχει το Button Info, όπου μας μεταφέρει στο Info_Layout και περιέχει τα text fields αφετηρίας, προορισμού, τωρινής τοποθεσίας, μέσο μεταφοράς container και ημερομηνίες αναχώρησης και άφιξης. Αυτές οι πληροφορίες έρχονται από την ΒΔ, μέσω του PackageID. Η οθόνη από κάτω συνοδεύεται με 4buttons, stopovers, weather in current position, weather in destination position και close. Το 1ο button μας μεταφέρει σε άλλη οθόνη όπου περιέχει όλες τις πληροφορίες που έχει η ΒΔ για το συγκεκριμένο δρομολόγιο. Το 2ο και 3ο button εμφανίζει οθόνες όπου αντλεί πληροφορίες από την ιστοσελίδα http://openweather.org για τις πόλεις του προορισμού και της τωρινής τοποθεσίας. Το 4ο button, close, μας επιστρέφει στον οθόνη του χάρτη.

Το 2ο button: “Tracking History”, καλεί το History Activity, όπως περιγράφηκε παραπάνω. Στο παρασκήνιο, έχει ήδη τρέξει η ταυτοποίηση της συσκευής και μόλις αντιστοιχηθεί και έχει άδεια αποδοχής πληροφοριών, εμφανίζει έναν χάρτη με στίγματα για όλα τα δρομολόγια. Ο χρήστης είναι ελεύθερος να δει τις πληροφορίες κάθε δρομολογίου ακουμπώντας το κάθε στίγμα. Μόλις βρει το ενδιαφερόμενο πατάει πάνω στον κωδικό του, PackageID και μεταφέρεται σε αναλυτικό χάρτη περιγραφής του δρομολογίου με δυνατότητα ανάγνωσης περισσότερων πληροφοριών μέσω του Info button.

Το 3ο button: “About Application”, καλεί το About Activity, όπως περιγράφηκε παραπάνω. Έχει πολλά text fields με τα στοιχεία της εφαρμογής και στοιχεία επικοινωνίας.

3.6 Συνοπτική επισκόπηση και σύγκριση με παρόμοια διαθέσιμα συστήματα της αγοράς

Με την ραγδαία ανάπτυξη του λογισμικού Android, το επίσημο market της Google, πλημμύρισε με εκατομμύρια εφαρμογές. Ένας σωστός προγραμματιστής πριν αρχίσει να σχεδιάζει πρέπει να κάνει μία κρίσιμη ερώτηση: «Σε τι θα διαφέρει η δική μου εφαρμογή από τις άλλες που υπάρχουν>»?

Με κίνητρο αυτή την ερώτηση, αφιέρωσα πολύ χρόνο σε αντίστοιχες εφαρμογές στο Google Play Store και προς έκπληξη μου δε βρήκα κάποια εφαρμογή που να διαθέτει στον χρήστη 3D απεικόνιση σε Google Maps.

- Arkas Line: Τα κριτήρια αναζήτησης ενός εμπορευματοκιβωτίου είναι παρόμοια με αυτά της Container Tracker. Δεν χρησιμοποιούνται όμως χάρτες και δεν δίνεται η δυνατότητα ενημέρωσης του χρήστη σε πραγματικό χρόνο.
- KCTC: Πλούσια εφαρμογή σε επιλογές καθώς δίνει την δυνατότητα στον χρήστη να υπολογίσει το κόστος μεταφοράς του εμπορευματοκιβωτίου. Δίνει στον χρήστη την δυνατότητα να ενημερωθεί σε πραγματικό χρόνο που βρίσκεται το φορτίο του, εμφανίζοντας τις γεωγραφικές συντεταγμένες. Δεν περιέχει απεικόνιση των συντεταγμένων σε χάρτη.
- EZ Track GPS Container: Παρόμοια εφαρμογή με την KCTC, χωρίς το λογιστικό μέρος, αλλά με αναφορά στο ενδεικτικό κόστος μετακίνησης. Δεν περιέχει απεικόνιση του εμπορευματοκιβωτίου στον χάρτη και δεν περιλαμβάνει πολλές λεπτομέρειες για το είδος του φορτίου, ούτε το χρονοδιάγραμμα που θα τηρηθεί.
- Florens Container: Έχει την δυνατότητα μεταφοράς εμπορευματοκιβωτίου μόνο μέσω θαλάσσιων δρομολογίων. Δεν περιέχει απεικόνιση των συντεταγμένων σε χάρτη, αλλά δίνει την δυνατότητα στον χρήστη να διαχειριστεί την κράτηση του με αλλαγή προορισμού κλπ. Επίσης έχει καρτέλα με αναλυτικές χρεώσεις της μεταφοράς.
- H&S Container Line: Εφαρμογή παρόμοια με την Container Tracker. Δίνει στον χρήστη δυνατότητα απεικόνισης του εμπορευματοκιβωτίου σε πραγματικό χρόνο σε 3D χάρτη, αλλά περιορίζεται μόνο σε δρομολογία στον Ρήνο ποταμό.

3.7 Συμπεράσματα και προτάσεις επέκτασης

Στόχος της παρούσας εργασίας ήταν η ανάπτυξη μιας εφαρμογής, η οποία θα ήταν χρήσιμη σε εταιρίες διανομής Containers και θα έδινε ταυτόχρονα τη δυνατότητα μελέτης και χρήσης νέων τεχνολογιών και προτύπων ανάπτυξης.

Η εκπόνηση της εργασίας αυτής έδωσε τη δυνατότητα της μελέτης του λειτουργικού συστήματος Android. Πρόκειται για ένα λειτουργικό σύστημα που στοχεύει στην καλύτερη χρήση των χαρακτηριστικών μιας φορητής συσκευής, έχοντας ως στόχο να δώσει το δυνατόν καλύτερο αποτέλεσμα. Προσφέρει πάρα πολλά εργαλεία και οι δυνατότητες του, καλύπτουν σχεδόν κάθε είδους ανάγκη του χρήστη. Πρόκειται για μια ιδιαίτερα εξελισσόμενη πλατφόρμα που αυτή τη στιγμή κατέχει το μεγαλύτερο μερίδιο στην αγορά των φορητών συσκευών. Στα πλαίσια της εργασίας χρησιμοποιήθηκαν τα πιο κοινά στοιχεία του και λίγες ιδιαίτερες δυνατότητές του, αλλά μέσω της έρευνας έγιναν γνωστές πολλές από τις δυνατότητές του και ο τρόπος με τον οποίο μπορούν να υλοποιηθούν.

Επίσης, μελετήθηκε η γλώσσα PhP και η δυνατότητα δημιουργίας βάσης δεδομένων. Μέσω της PhP πετύχαμε τον στόχο μας που ήταν να δημιουργήσουμε έναν web – server, απ' όπου θα αντλεί πληροφορίες η Android εφαρμογή.

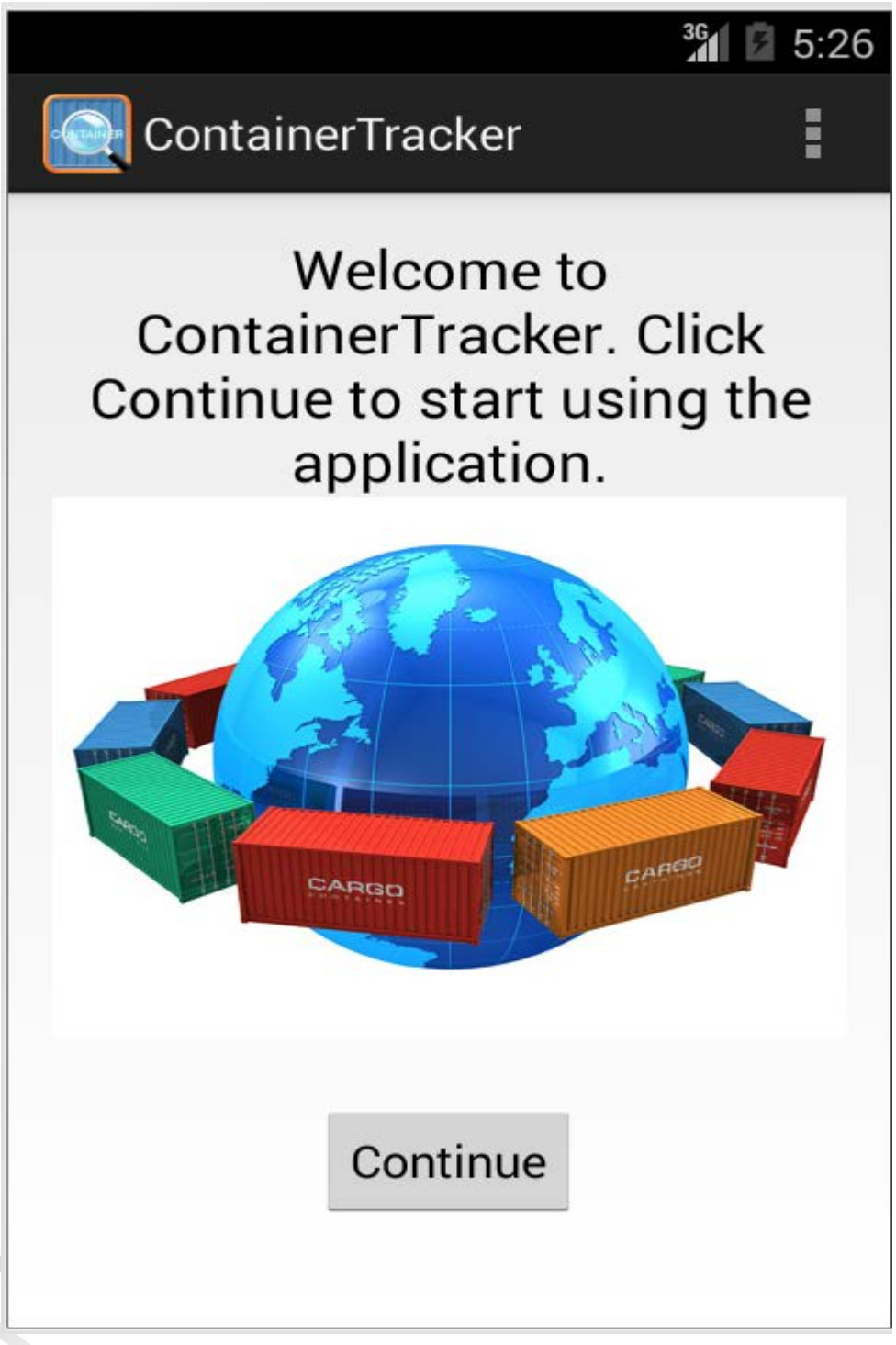
Το σύστημα θα μπορούσε να αναπτυχθεί και περαιτέρω προσθέτοντας σε αυτήν διάφορα ενδιαφέροντα χαρακτηριστικά και νέες δυνατότητες, τόσο στην εφαρμογή της φορητής συσκευής όσο και στο κεντρική βάση δεδομένων.

Κάποιες προτάσεις όσον αφορά τη φορητή συσκευή είναι η δημιουργία μιας περισσότερο εξελιγμένης υπηρεσίας η οποία θα παρέχει ειδοποιήσεις – alarms στον χρήστη όταν ένα δρομολόγιο έχει υπερβεί τον αναμενόμενο χρόνο άφιξης. Επίσης, θα μπορούσε να υπάρχει πεδίο, ώστε ο χρήστης να προσθέτει ειδικές επισημάνσεις και προσωπικά σχόλια επάνω στις πληροφορίες του δρομολογίου. Ακόμη, πολλές εφαρμογές που υπάρχουν ήδη διαθέσιμες στο Google Play, μπορούν να υπολογίσουν και το κόστος μεταφοράς του εμπορευματοκιβωτίου, οπότε θα μπορούσε να προστεθεί και αυτό στην εφαρμογή.

Η εκπόνηση της συγκεκριμένης εργασίας ήταν μια δημιουργική και εποικοδομητική διαδικασία επάνω σε νέες τεχνολογίες και πρότυπα και ταυτόχρονα δημιουργήθηκε ένα μικρό σύστημα το οποίο μπορεί να εξελιχθεί σε κάτι πολύ πιο

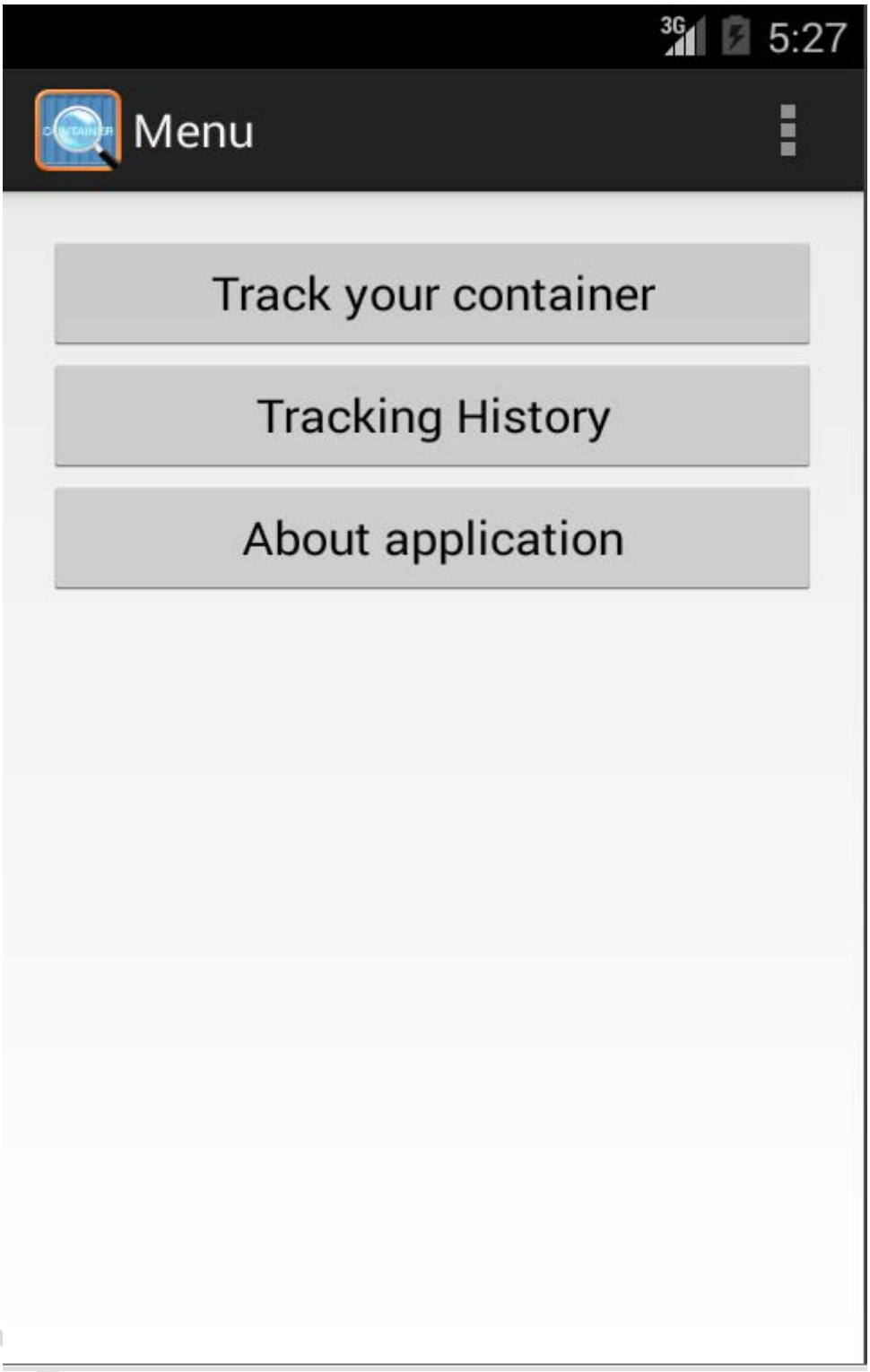
σύνθετο και ιδιαίτερα χρήσιμο για κάποια μικρή ή ακόμα και μεγάλη εταιρία που διακινεί Containers.

Κεφάλαιο 4ο : Παρουσίαση Container Tracker



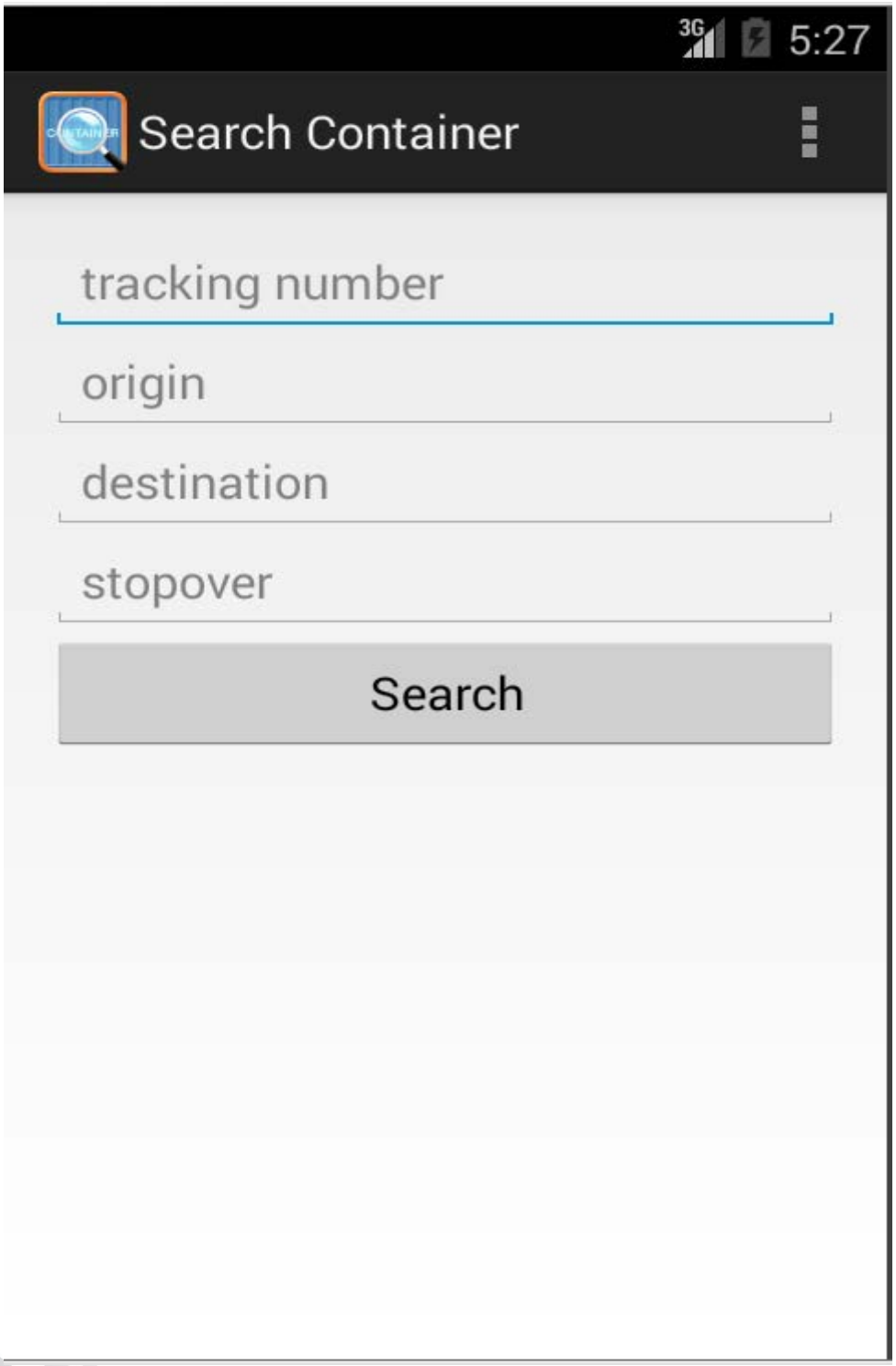
Εικόνα 12 Αρχική Οθόνη Εφαρμογής

Αυτή είναι η αρχική οθόνη της εφαρμογής με το λογότυπο της εφαρμογής και ένα μήνυμα καλωσορίσματος.



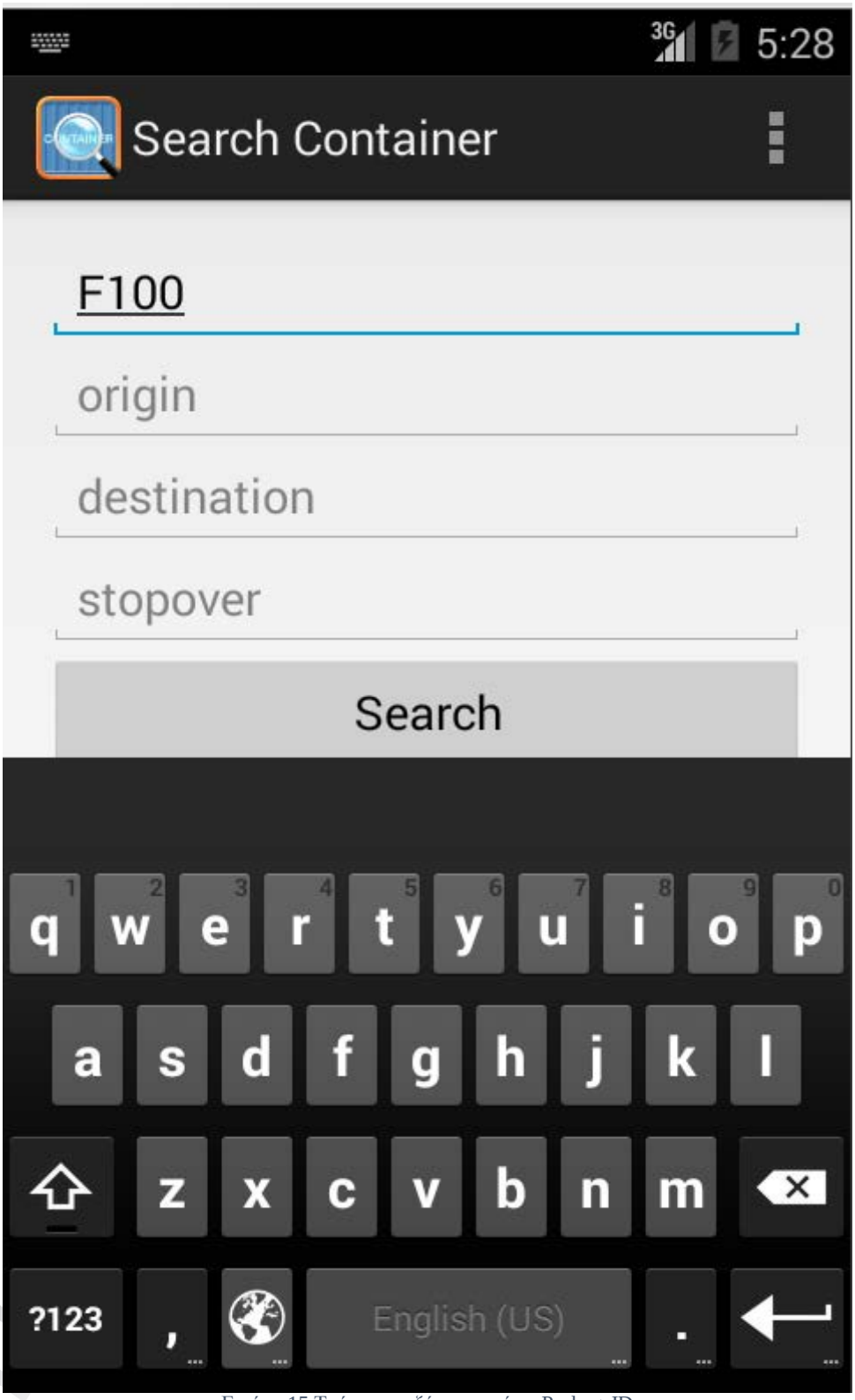
Εικόνα 13 Μενού Εφαρμογής

Αυτή είναι η 2η εικόνα της εφαρμογής, με το κυρίως μενού. Από εδώ ο χρήστης μπορεί είτε να αναζητήσει το container, είτε να δει το ιστορικό ενός δρομολογίου. Στο About Tab έχει διάφορες πληροφορίες για την εφαρμογή.



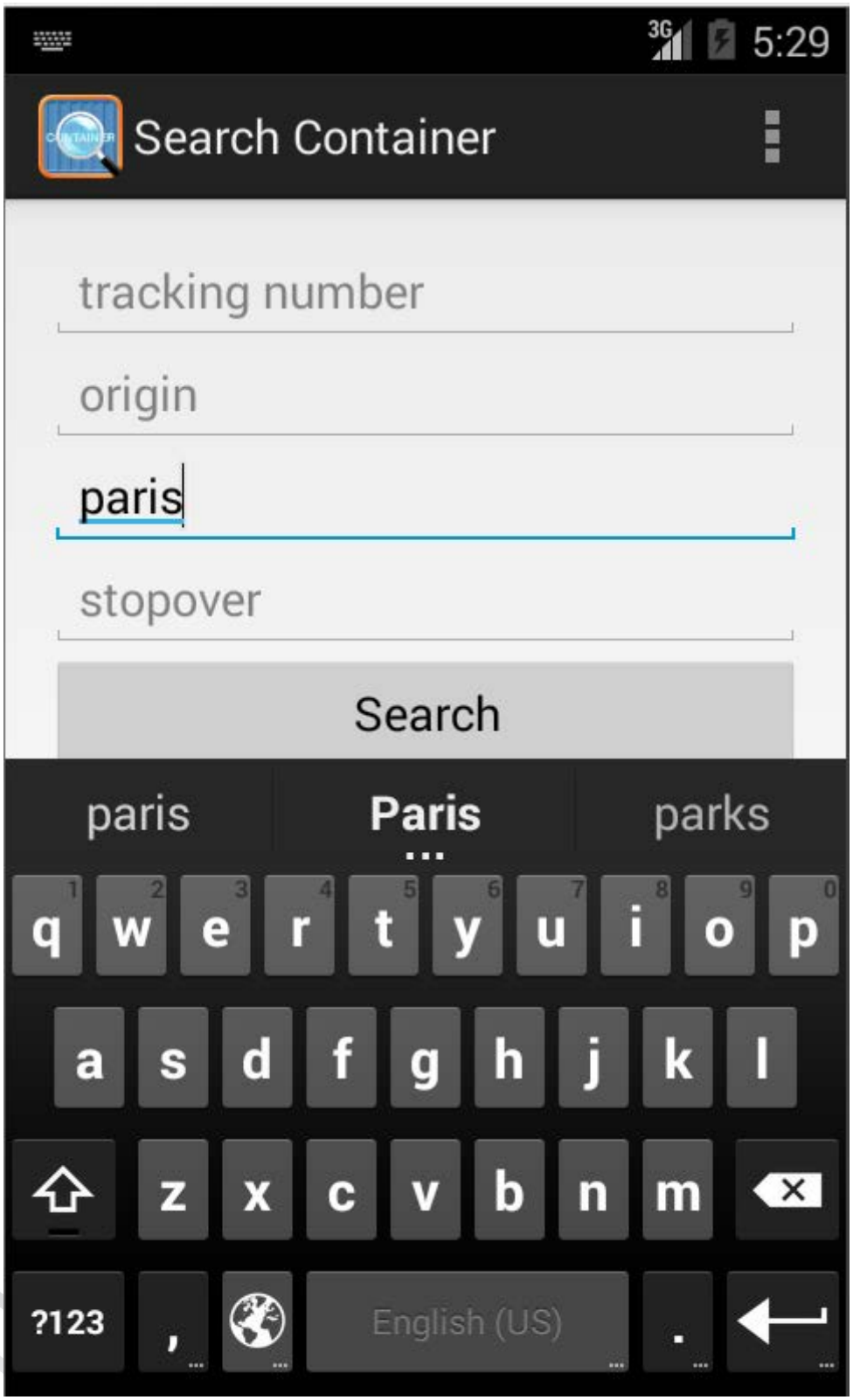
Εικόνα 14 Αναζήτηση Container με διάφορα κριτήρια

Σε αυτή την εικόνα απεικονίζονται τα φίλτρα αναζήτησης ενός δρομολογίου. Ο χρήστης μπορεί να εισάγει το Tracking number αν το γνωρίζει, αλλιώς μπορεί να πραγματοποιήσει αναζήτηση είτε εισάγοντας την αφετηρία είτε τον προορισμό. Επίσης μπορεί να εισάγει την τοποθεσία μίας ενδιάμεσης στάσης στην οποία βρίσκεται το container.



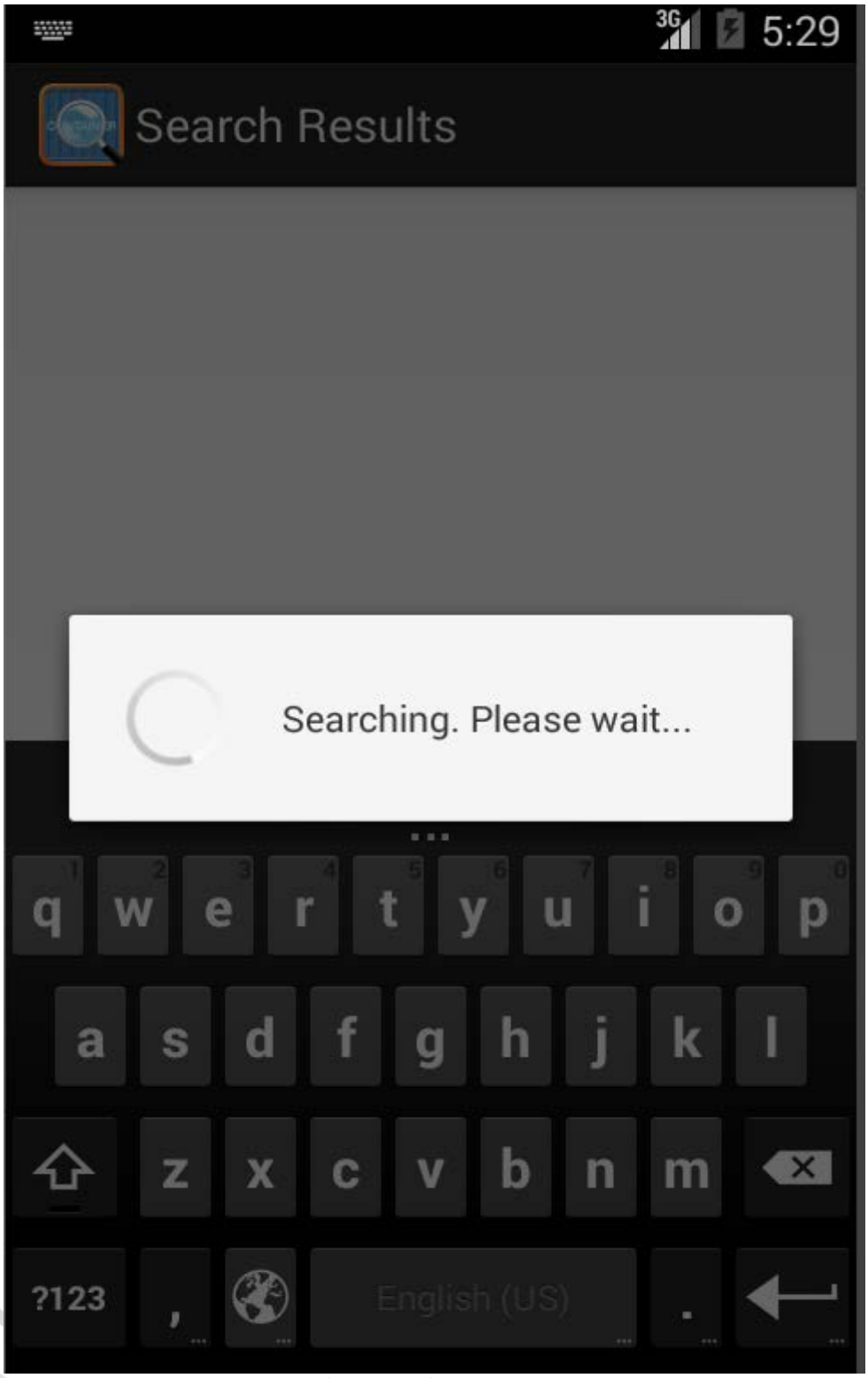
Εικόνα 15 Τρόπος αναζήτησης μέσω PackageID

Όπως προαναφέραμε η αναζήτηση μπορεί να γίνει μέσω του Tracking Number ή αλλιώς PackageID.



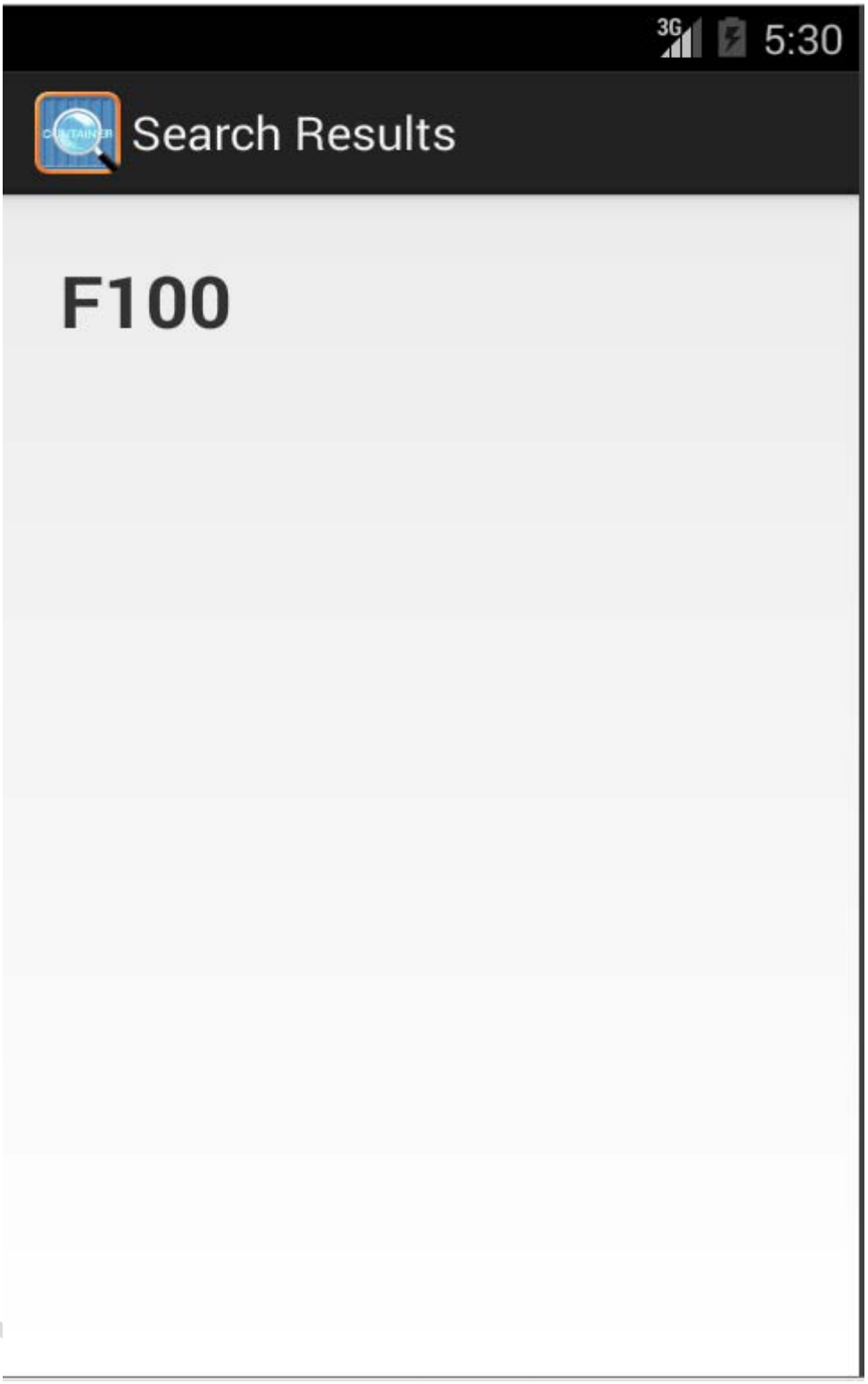
Εικόνα 16 Τρόπος αναζήτησης μέσω προορισμού

Όπως προαναφέραμε, η εφαρμογή δίνει την δυνατότητα αναζήτησης και μέσω της γεωγραφικής τοποθεσίας του container.



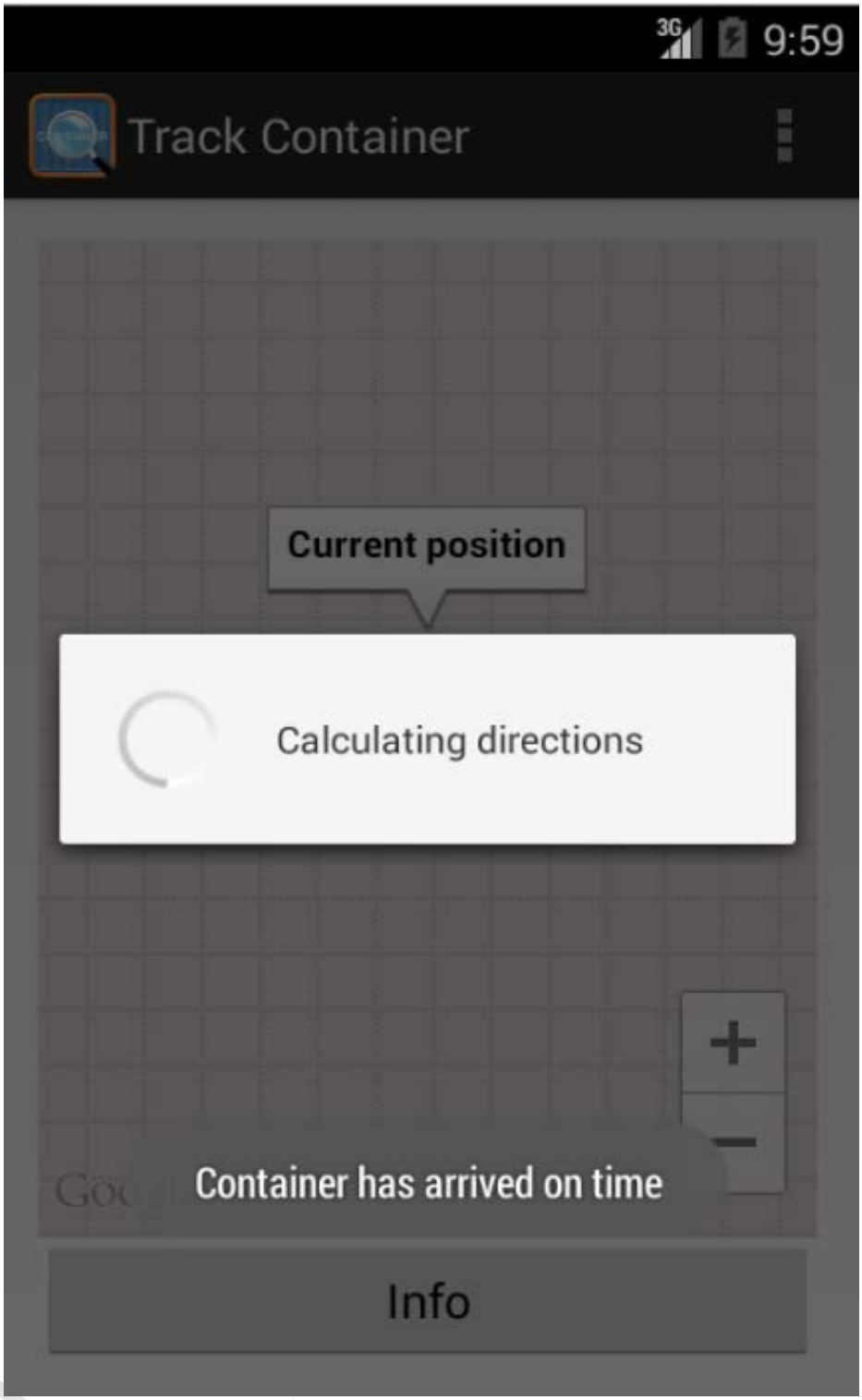
Εικόνα 17 Διαδικασία σύνδεσης με ΒΔ και αναζήτηση

Σε αυτή την εικόνα απεικονίζεται ένα φιλικό μήνυμα προς τον χρήστη, ενώ στο παρασκήνιο πραγματοποιείται σύνδεση της εφαρμογής στην Βάση Δεδομένων.



Εικόνα 18 Αποτελέσματα αναζήτησης

Με όποιον τρόπο και αν γίνει η αναζήτηση από τον χρήστη, η εφαρμογή θα δείχνει στα αποτελέσματα τον αριθμό δρομολογίου του container. Αυτό γίνεται γιατί στην Βάση Δεδομένων, το PackageID είναι το πρωτεύων κλειδί.

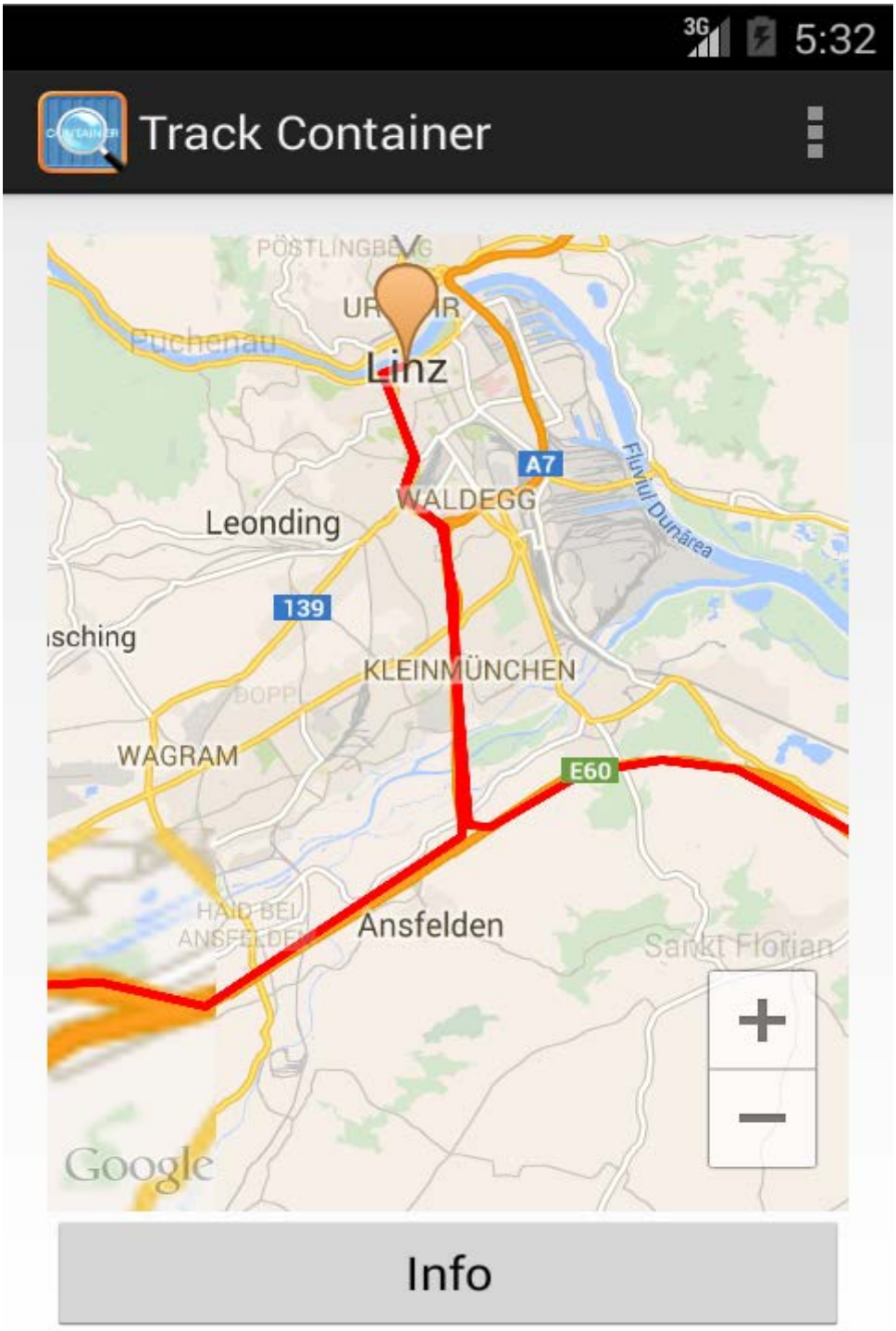


Εικόνα 19 Σύνδεση με Google Maps και προετοιμασία χάρτη

Σε αυτή την εικόνα ο χρήστης βλέπει το Google Maps Layout. Στο παρασκήνιο, η εφαρμογή έχει αρχίσει να κατεβάζει τις πληροφορίες από την Βάση Δεδομένων και μέσω σύνδεσης στο ίντερνετ πραγματοποιεί απεικόνιση τους στους χάρτες της Google. Επίσης ο χρήστης πληροφορείται εάν το δρομολογίο έχει φτάσει στον προορισμό και αν είναι εντός χρονοδιαγράμματος.

Εικόνα 20 Απεικόνιση τρέχουσας τοποθεσίας Container

Σε αυτή την εικόνα απεικονίζεται η τρέχουσα θέση του container. Ο χρόνος ανανέωσης των χαρτών έχει οριστεί στο 1λεπτό.

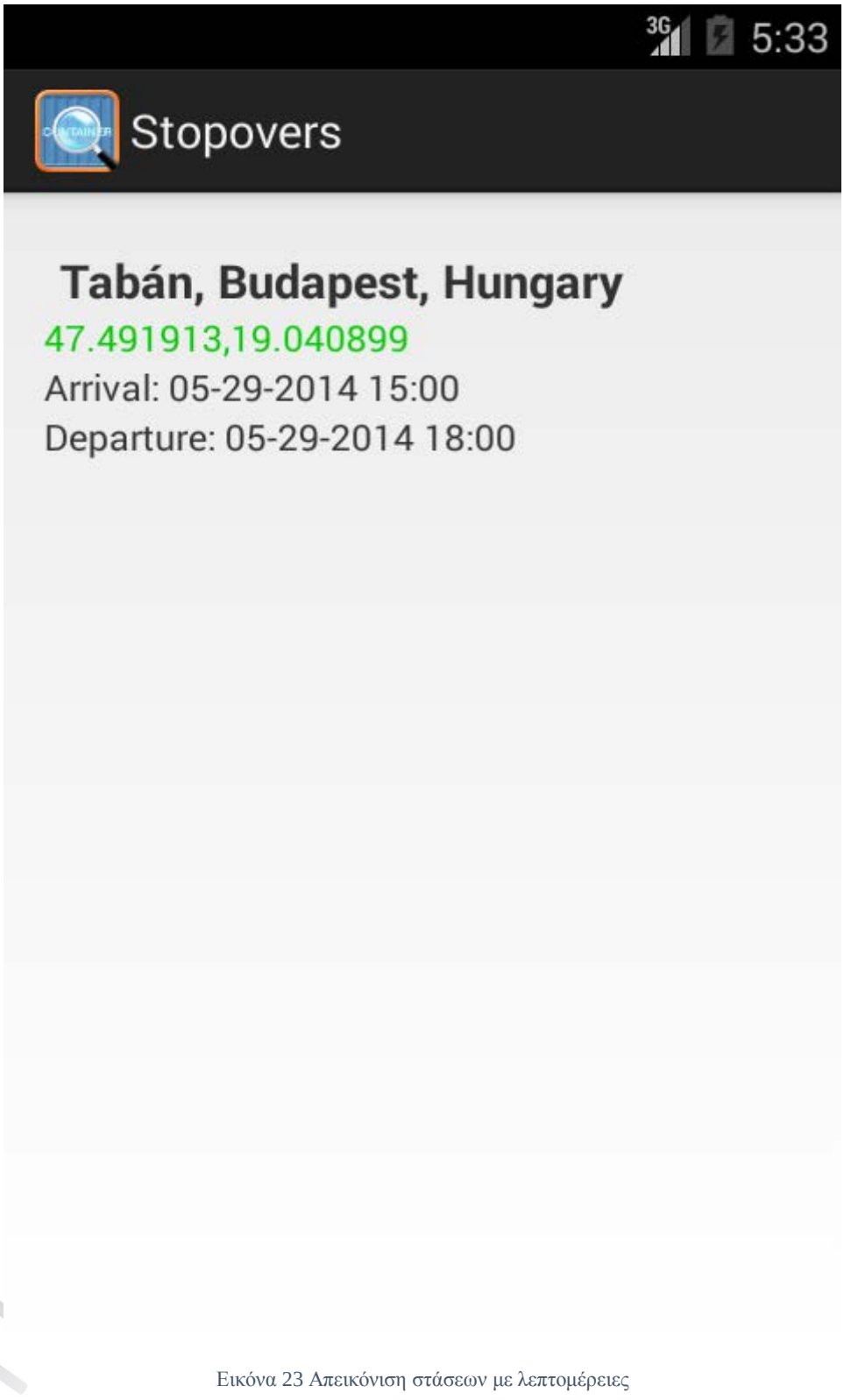


Εικόνα 21 Δυνατότητα zoom-in και scroll στον χάρτη για περισσότερες πληροφορίες

Ο χρήστης έχει την δυνατότητα να κάνει περιήγηση στον χάρτη καθώς και να κάνει zoom – in για προβολή μεγαλύτερης ανάλυσης χάρτη και να δει περισσότερες λεπτομέρειες. Με την κόκκινη γραμμή απεικονίζεται η μέχρι στιγμής πορεία του Container.

Εικόνα 22 Αναλυτικές πληροφορίες Container

Το κουμπί Info δίνει στον χρήστη την δυνατότητα να δει αναλυτικότερα το γεωγραφικό στίγμα του container, το χρονοδιάγραμμα αναχώρησης και άφιξης, το μέσο μεταφοράς και πληροφορίες σχετικά με το περιεχόμενο του Container.

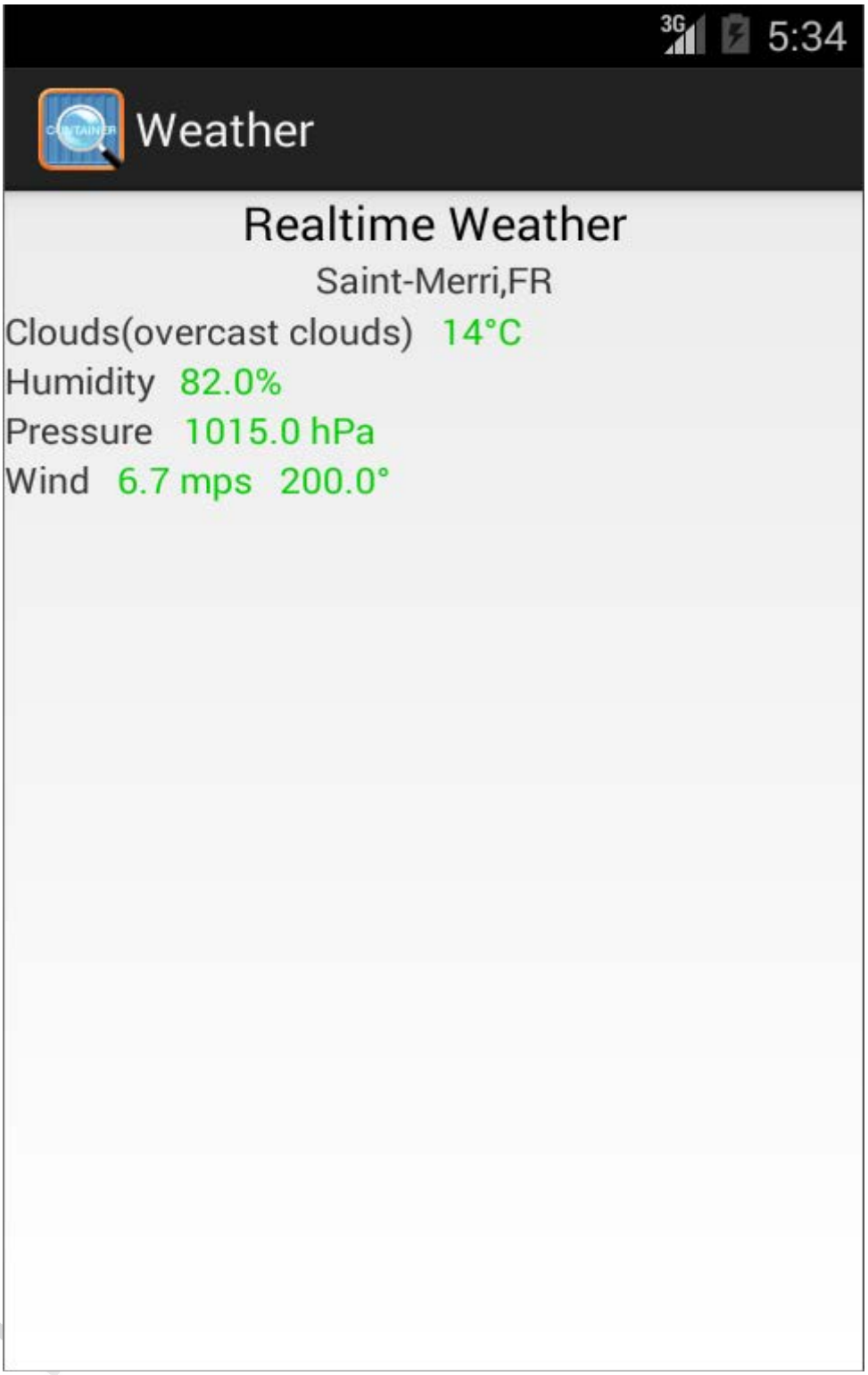


Εικόνα 23 Απεικόνιση στάσεων με λεπτομέρειες

Με την επιλογή του κουμπιού Stopovers, ο χρήστης βλέπει την τοποθεσία όπου έμεινε το container ακίνητο για ένα χρονικό διάστημα.

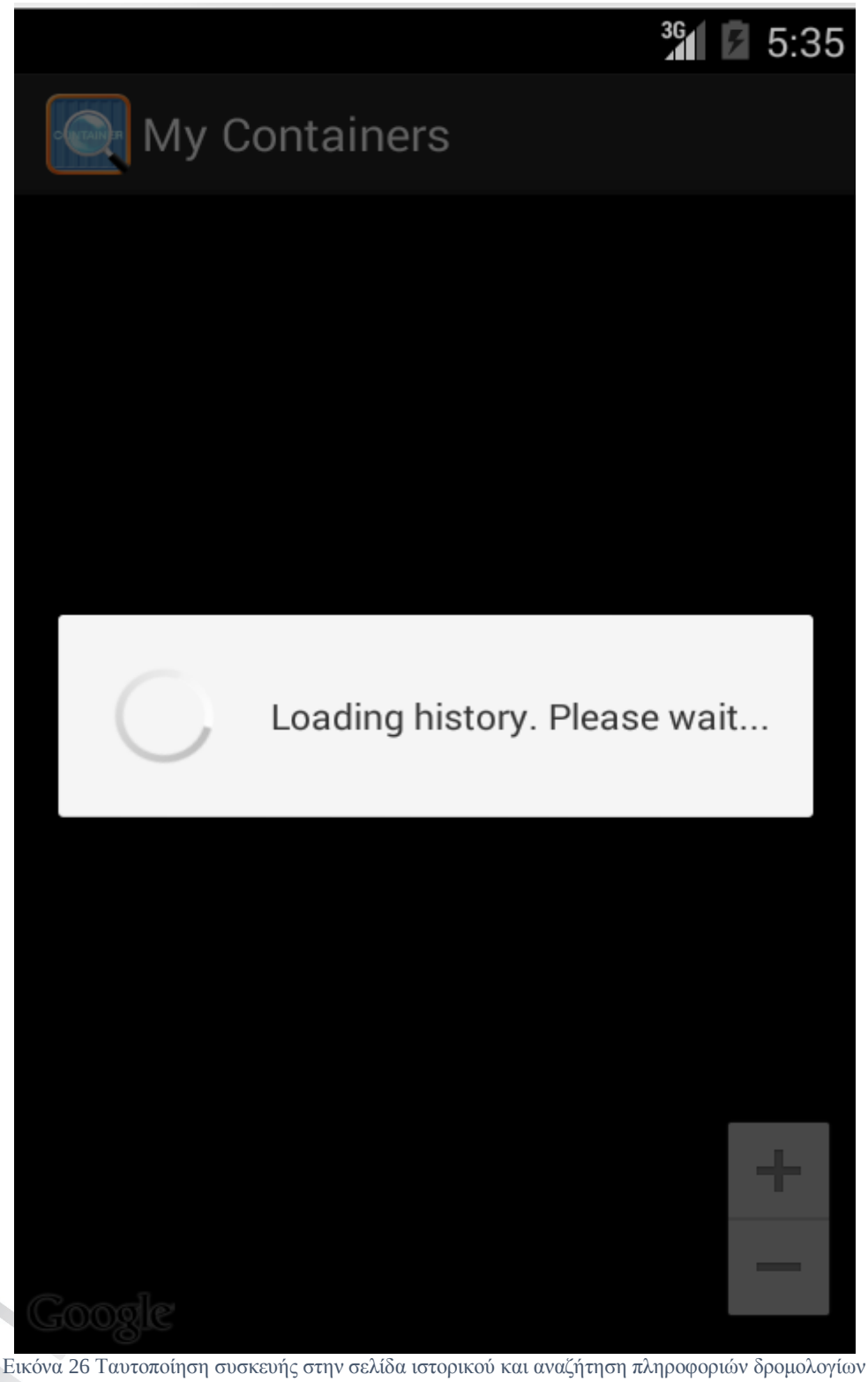
Εικόνα 24 Καιρικές συνθήκες στην τρέχουσα τοποθεσία

Ο χρήστης έχει την δυνατότητα να δει τις καιρικές συνθήκες στην παρούσα τοποθεσία του Container. Η πληροφορία αυτή προέρχεται από την ιστοσελίδα http://openweather.org



Εικόνα 25 Καιρικές συνθήκες στον προορισμό

Ο χρήστης έχει την δυνατότητα να δει και τις καιρικές συνθήκες στον προορισμό. Η πληροφορία αυτή προέρχεται από την ιστοσελίδα http://openweather.org



Εικόνα 26 Ταυτοποίηση συσκευής στην σελίδα ιστορικού και αναζήτηση πληροφοριών δρομολογίων

Σε αυτή την εικόνα, απεικονίζεται ένα φιλικό μήνυμα προς τον χρήστη ενώ στο παρασκήνιο γίνεται ταυτοποίηση της συσκευής του με τις επιτρεπόμενες συσκευές που έχουν πρόσβαση στο ιστορικό ενός δρομολογίου Container.

Εικόνα 27 Ιστορικό δρομολογίων

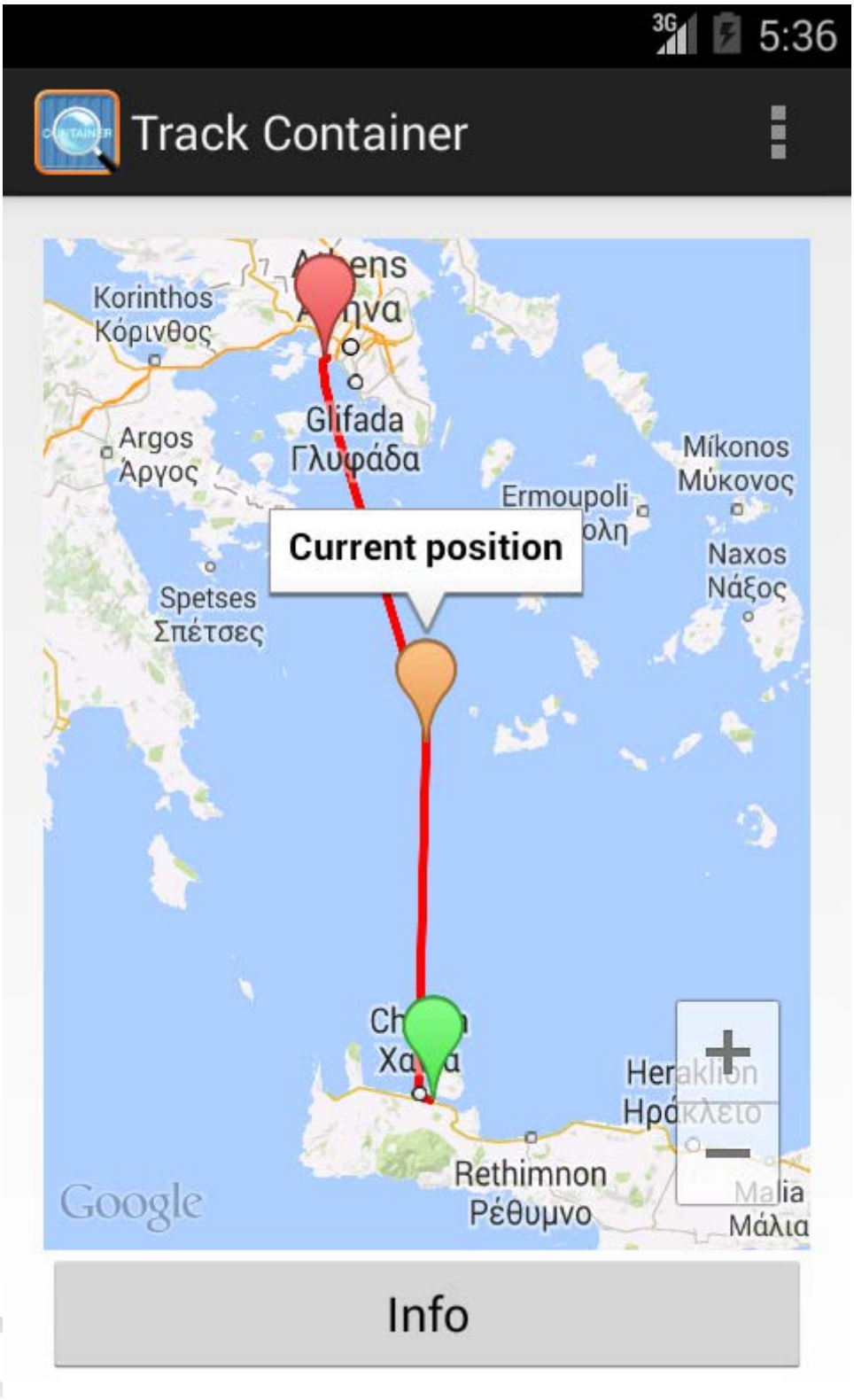
Σε αυτή την εικόνα, ο χρήστης έχει περάσει την αυθεντικοποίηση της συσκευής του και του απεικονίζεται ο χάρτης με όλα τα δρομολόγια για τα οποία μπορεί να δει λεπτομέρειες.

Εικόνα 28 Clickable δείκτες

Ο χρήστης μπορεί να επιλέξει όποιο δρομολόγιο επιθυμεί και να δει μία την τρέχουσα τοποθεσία του, χωρίς να χρειάζεται να δει άλλες πληροφορίες.

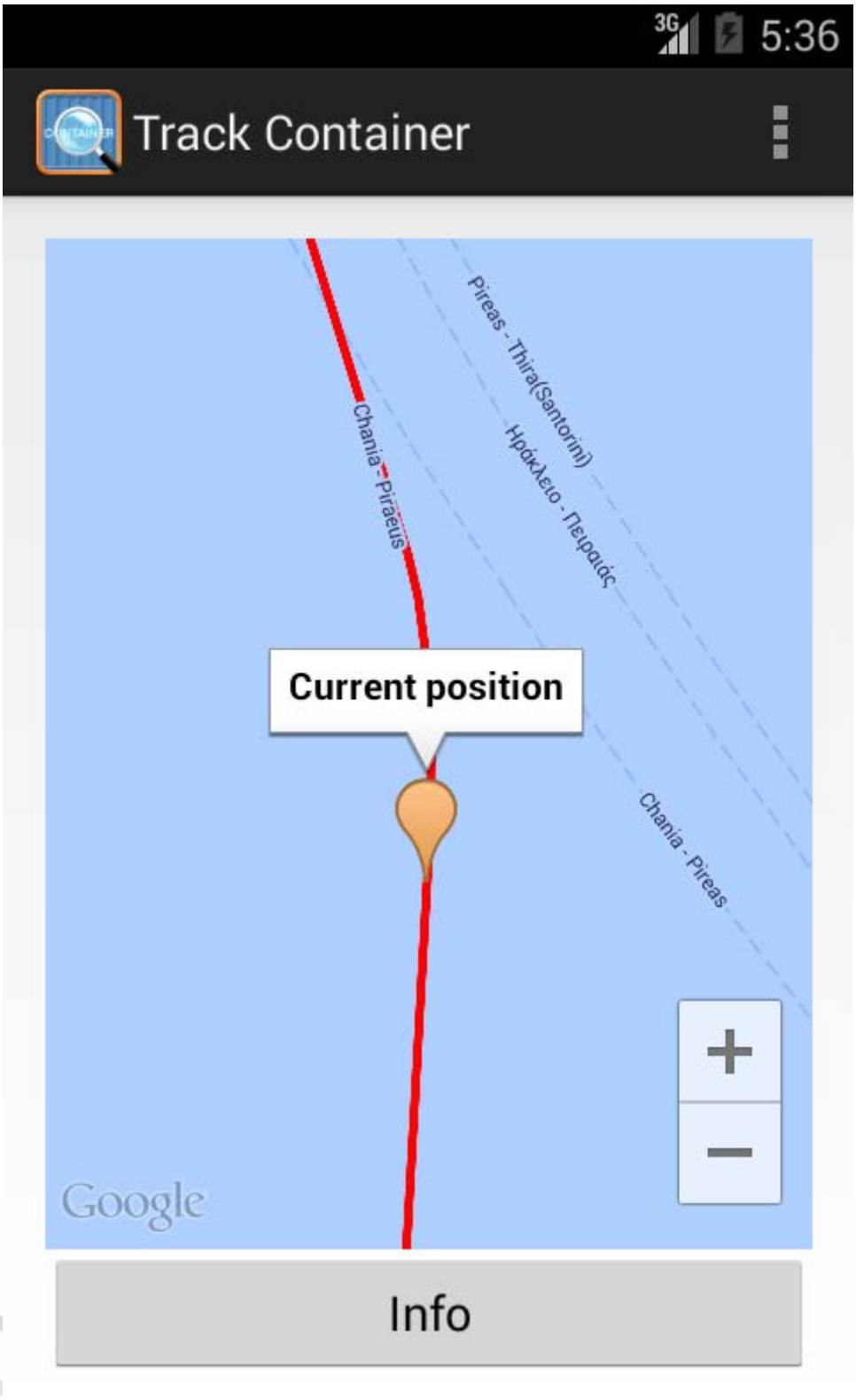
Εικόνα 29 Φόρτωση ιστορικού επιλεγμένου δρομολογίου

Στην εικόνα φαίνεται η προετοιμασία των Google Maps.



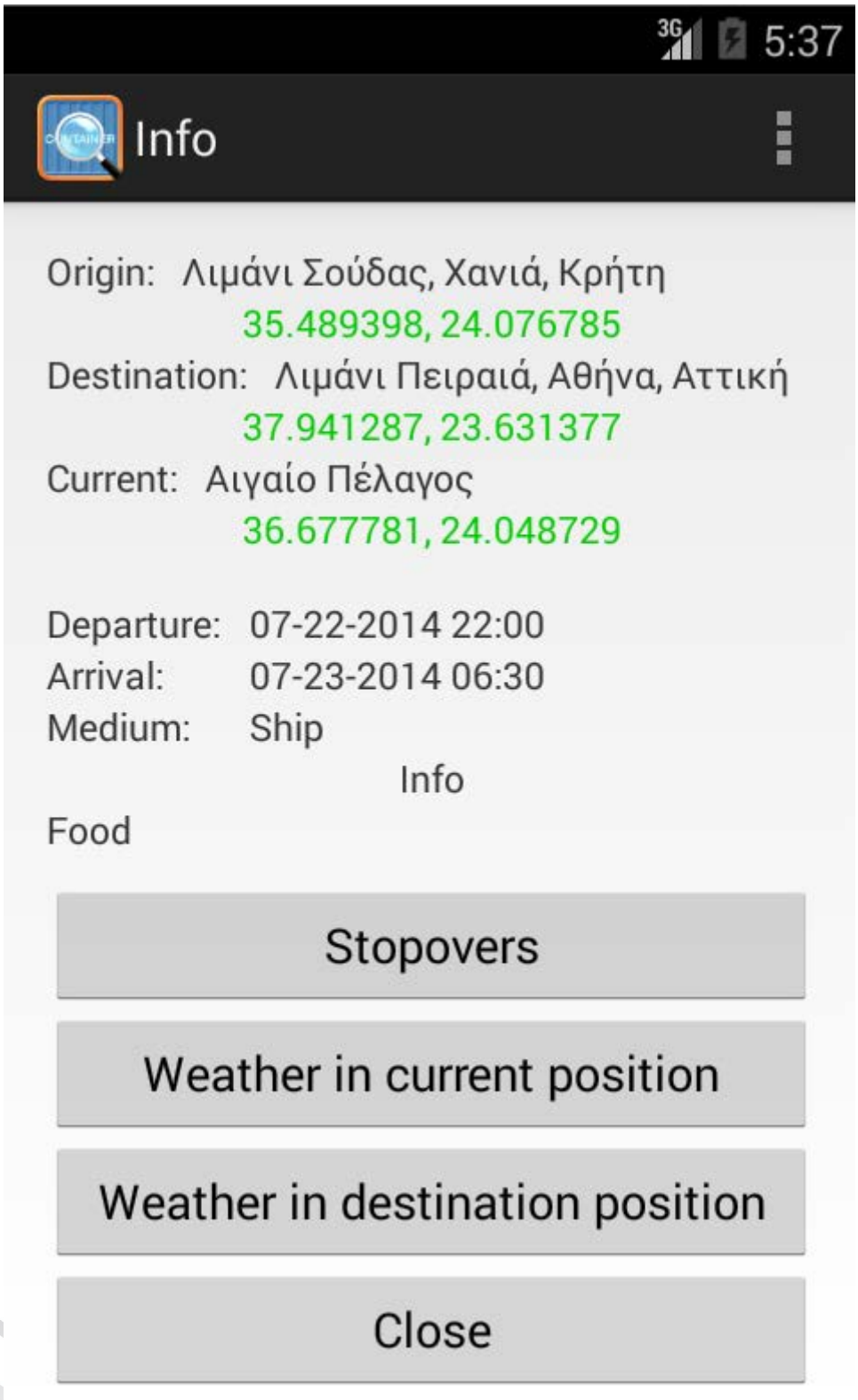
Εικόνα 30 Απεικόνιση δρομολογίου και τρέχουσας θέσης Container

Στην εικόνα απεικονίζεται το επιλεγμένο δρομολόγιο.



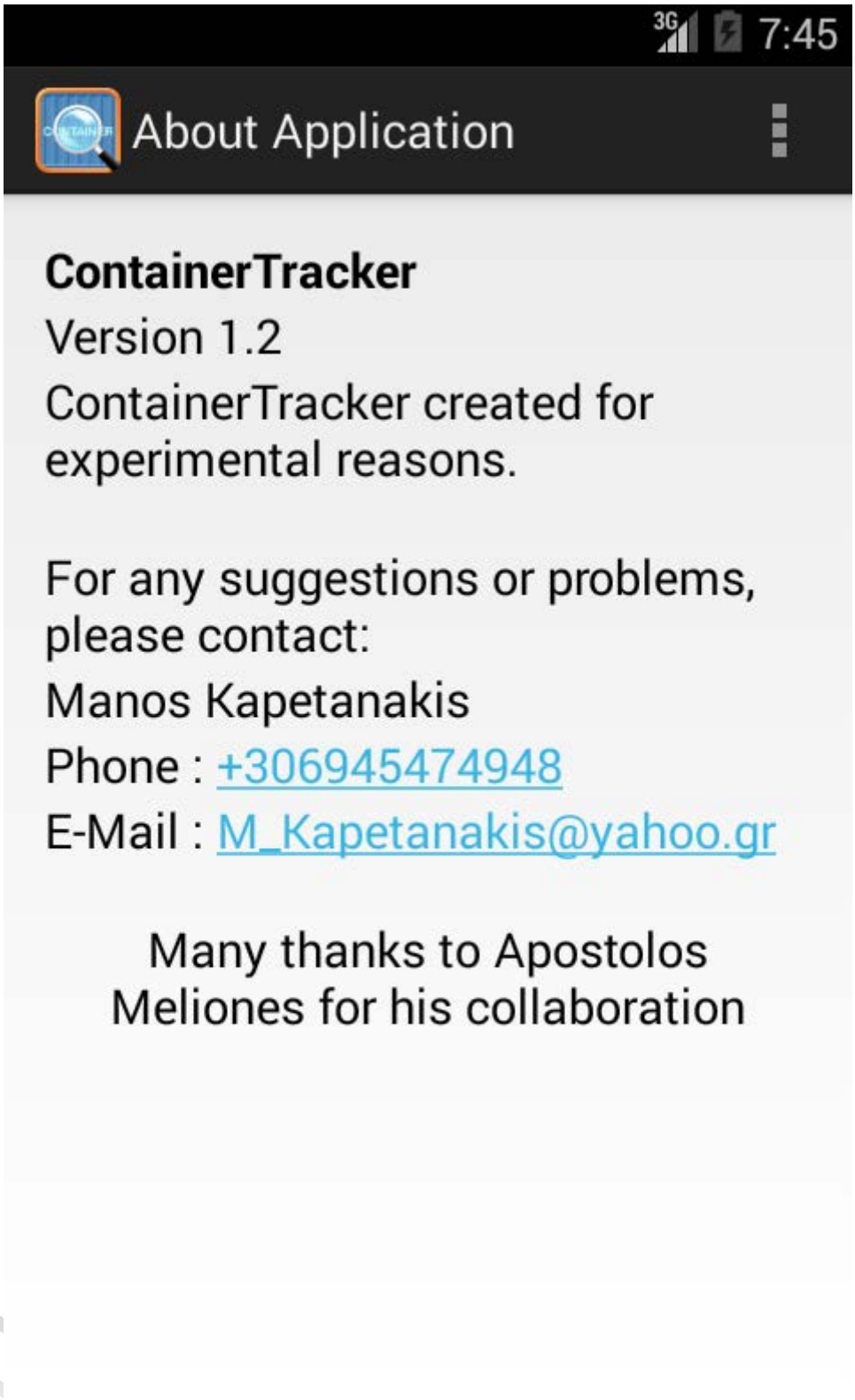
Εικόνα 31 Απεικόνιση ναυτιλιακών «δρόμων»

Στην εικόνα απεικονίζονται οι ναυτιλιακές ρότες που οφείλουν να ακολουθούν τα πλοία.



Εικόνα 32 Αναλυτικές πληροφορίες

Ο χρήστης μπορεί να δει περισσότερες πληροφορίες.



Εικόνα 33 Πληροφορίες για την εφαρμογή

Η 3η επιλογή που έχει ο χρήστης όταν βρίσκεται στο κυρίως μενού. Στο About Tab φαίνεται η έκδοση της εφαρμογής καθώς και τα απαραίτητα στοιχεία επικοινωνίας σε περίπτωση που ο χρήστης θέλει κάποια διευκρίνιση.

Παράρτημα : Κώδικας εφαρμογής Container Tracker

ContainerTrackerManifest.java

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

    <uses-sdk
        android:minSdkVersion="11"
        android:targetSdkVersion="19" />

    <permission
        android:name="my.application.containertracker.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission
android:name="my.application.containertracker.permission.MAPS_RECEIVE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />

    <uses-feature
```

```
    android:glEsVersion="0x00020000"
    android:required="true" />

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

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name="my.application.containertracker.MapActivity"
        android:label="@string/title_activity_map" >
    </activity>

    <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyBrHJpwNuvz4yZEXFKXFnhuII_er-DRG4c" />
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version" />

    <activity
        android:name="my.application.containertracker.SearchActivity"
        android:label="@string/title_activity_search" >
    </activity>
    <activity
```

```
            android:name="my.application.containertracker.MenuActivity"
            android:label="@string/title_activity_menu" >
        </activity>
        <activity
            android:name="my.application.containertracker.InfoActivity"
            android:label="@string/title_activity_info" >
        </activity>
        <activity
            android:name="my.application.containertracker.HistoryActivity"
            android:label="@string/title_activity_history" >
        </activity>
        <activity
            android:name="my.application.containertracker.AboutActivity"
            android:label="@string/title_activity_about" >
        </activity>
        <activity
            android:name="my.application.containertracker.MyContainersActivity"
            android:label="@string/title_activity_my_containers" >
        </activity>
        <activity
            android:name="my.application.containertracker.WeatherActivity"
            android:label="@string/title_activity_weather" >
        </activity>
        <activity
            android:name="my.application.containertracker.SearchResultsActivity"
            android:label="@string/title_activity_search_results" >
        </activity>
        <activity
            android:name="my.application.containertracker.StopoversActivity"
            android:label="@string/title_activity_stopovers" >
        </activity>
    </application>
</manifest>
```

AboutActivity.java

```
package my.application.containertracker;

import android.os.Bundle;
import android.app.Activity;
import android.text.method.LinkMovementMethod;
import android.text.util.Linkify;
import android.view.Menu;
import android.widget.EditText;
import android.widget.TextView;

public class AboutActivity extends Activity {

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

		TextView phoneTxt = (TextView) findViewById(R.id.textViewPhone);
		Linkify.addLinks(phoneTxt, Linkify.ALL);
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.about, menu);
		return true;
	}

}
```

GetDirectionASyncTask.java

```
package my.application.containertracker;

import java.util.ArrayList;
import java.util.Map;
import org.w3c.dom.Document;
import com.google.android.gms.maps.model.LatLng;
import android.app.ProgressDialog;
import android.os.AsyncTask;
import android.widget.Toast;

public class GetDirectionsAsyncTask extends AsyncTask<Map<String, String>,
Object, ArrayList<LatLng>>
{
   public static final String USER_CURRENT_LAT = "user_current_lat";
   public static final String USER_CURRENT_LONG = "user_current_long";
   public static final String DESTINATION_LAT = "destination_lat";
   public static final String DESTINATION_LONG = "destination_long";
   public static final String STOPOVERS = "stopovers";
   public static final String DIRECTIONS_MODE = "directions_mode";
   private MapActivity activity;
   private Exception exception;
   private ProgressDialog progressDialog;

   public GetDirectionsAsyncTask(MapActivity activity)
   {
      super();
      this.activity = activity;
   }

   public void onPreExecute()
   {
      progressDialog = new ProgressDialog(activity);
```

```
        progressDialog.setMessage("Calculating directions");
        progressDialog.show();
    }

    @Override
    public void onPostExecute(ArrayList<LatLng> result)
    {
        progressDialog.dismiss();
        if (exception == null)
        {
            activity.handleGetDirectionsResult(result);
        }
        else
        {
            processException();
        }
    }

    @Override
    protected ArrayList<LatLng> doInBackground(Map<String, String>... params)
    {
        Map<String, String> paramMap = params[0];
        try
        {
            LatLng fromPosition = new
LatLng(Double.valueOf(paramMap.get(USER_CURRENT_LAT)) ,
Double.valueOf(paramMap.get(USER_CURRENT_LONG)));
            LatLng toPosition = new
LatLng(Double.valueOf(paramMap.get(DESTINATION_LAT)) ,
Double.valueOf(paramMap.get(DESTINATION_LONG)));
            GMapV2Direction md = new GMapV2Direction();
            Document doc = md.getDocument(fromPosition, toPosition,
paramMap.get(DIRECTIONS_MODE), paramMap.get(STOPOVERS));
            ArrayList<LatLng> directionPoints = md.getDirection(doc);
```

```
            return directionPoints;
        }
        catch (Exception e)
        {
            exception = e;
            return null;
        }
    }

    private void processException()
    {
        Toast.makeText(activity, "Error retriving data", 3000).show();
    }
}
```

GMAPv2Direction.java

```
package my.application.containertracker;

import java.io.InputStream;
import java.net.URLEncoder;
import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import android.util.Log;

import com.google.android.gms.maps.model.LatLng;

public class GMapV2Direction {
   public final static String MODE_DRIVING = "driving";
   public final static String MODE_WALKING = "walking";

   public GMapV2Direction() { }

   public Document getDocument(LatLng start, LatLng end, String mode, String
stopovers) {
      try {
```

```
                String url = "http://maps.googleapis.com/maps/api/directions/xml?"
                    + "origin=" + start.latitude + "," + start.longitude
                    + "&destination=" + end.latitude + "," + end.longitude
                    +   "&sensor=false&units=metric&mode=driving&waypoints="   +
URLEncoder.encode(stopovers, "UTF-8");


            HttpClient httpClient = new DefaultHttpClient();
            HttpContext localContext = new BasicHttpContext();
            HttpPost httpPost = new HttpPost(url);
            HttpResponse response = httpClient.execute(httpPost, localContext);
            InputStream in = response.getEntity().getContent();
            DocumentBuilder                    builder                    =
DocumentBuilderFactory.newInstance().newDocumentBuilder();
            Document doc = builder.parse(in);
            return doc;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    public String getDurationText (Document doc) {
        NodeList nl1 = doc.getElementsByTagName("duration");
        Node node1 = nl1.item(0);
        NodeList nl2 = node1.getChildNodes();
        Node node2 = nl2.item(getNodeIndex(nl2, "text"));
        Log.i("DurationText", node2.getTextContent());
        return node2.getTextContent();
    }

    public int getDurationValue (Document doc) {
        NodeList nl1 = doc.getElementsByTagName("duration");
        Node node1 = nl1.item(0);
```

```
    NodeList nl2 = node1.getChildNodes();
    Node node2 = nl2.item(getNodeIndex(nl2, "value"));
    Log.i("DurationValue", node2.getTextContent());
    return Integer.parseInt(node2.getTextContent());
}

public String getDistanceText (Document doc) {
    NodeList nl1 = doc.getElementsByTagName("distance");
    Node node1 = nl1.item(0);
    NodeList nl2 = node1.getChildNodes();
    Node node2 = nl2.item(getNodeIndex(nl2, "text"));
    Log.i("DistanceText", node2.getTextContent());
    return node2.getTextContent();
}

public int getDistanceValue (Document doc) {
    NodeList nl1 = doc.getElementsByTagName("distance");
    Node node1 = nl1.item(0);
    NodeList nl2 = node1.getChildNodes();
    Node node2 = nl2.item(getNodeIndex(nl2, "value"));
    Log.i("DistanceValue", node2.getTextContent());
    return Integer.parseInt(node2.getTextContent());
}

public String getStartAddress (Document doc) {
    NodeList nl1 = doc.getElementsByTagName("start_address");
    Node node1 = nl1.item(0);
    Log.i("StartAddress", node1.getTextContent());
    return node1.getTextContent();
}

public String getEndAddress (Document doc) {
    NodeList nl1 = doc.getElementsByTagName("end_address");
    Node node1 = nl1.item(0);
```

```
    Log.i("StartAddress", node1.getTextContent());
    return node1.getTextContent();
}

public String getCopyRights (Document doc) {
    NodeList nl1 = doc.getElementsByTagName("copyrights");
    Node node1 = nl1.item(0);
    Log.i("CopyRights", node1.getTextContent());
    return node1.getTextContent();
}

public ArrayList<LatLng> getDirection (Document doc) {
    NodeList nl1, nl2, nl3;
    ArrayList<LatLng> listGeopoints = new ArrayList<LatLng>();
    nl1 = doc.getElementsByTagName("step");
    if (nl1.getLength() > 0) {
        for (int i = 0; i < nl1.getLength(); i++) {
            Node node1 = nl1.item(i);
            nl2 = node1.getChildNodes();

            Node locationNode = nl2.item(getNodeIndex(nl2, "start_location"));
            nl3 = locationNode.getChildNodes();
            Node latNode = nl3.item(getNodeIndex(nl3, "lat"));
            double lat = Double.parseDouble(latNode.getTextContent());
            Node lngNode = nl3.item(getNodeIndex(nl3, "lng"));
            double lng = Double.parseDouble(lngNode.getTextContent());
            listGeopoints.add(new LatLng(lat, lng));

            locationNode = nl2.item(getNodeIndex(nl2, "polyline"));
            nl3 = locationNode.getChildNodes();
            latNode = nl3.item(getNodeIndex(nl3, "points"));
            ArrayList<LatLng> arr = decodePoly(latNode.getTextContent());
            for(int j = 0 ; j < arr.size() ; j++) {
                listGeopoints.add(new LatLng(arr.get(j).latitude, arr.get(j).longitude));
```

```
            }

            locationNode = nl2.item(getNodeIndex(nl2, "end_location"));
            nl3 = locationNode.getChildNodes();
            latNode = nl3.item(getNodeIndex(nl3, "lat"));
            lat = Double.parseDouble(latNode.getTextContent());
            lngNode = nl3.item(getNodeIndex(nl3, "lng"));
            lng = Double.parseDouble(lngNode.getTextContent());
            listGeopoints.add(new LatLng(lat, lng));
        }
    }

    return listGeopoints;
}

private int getNodeIndex(NodeList nl, String nodename) {
    for(int i = 0 ; i < nl.getLength() ; i++) {
        if(nl.item(i).getNodeName().equals(nodename))
            return i;
    }
    return -1;
}

private ArrayList<LatLng> decodePoly(String encoded) {
    ArrayList<LatLng> poly = new ArrayList<LatLng>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;
    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
```

```
            int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lat += dlat;
            shift = 0;
            result = 0;
            do {
               b = encoded.charAt(index++) - 63;
               result |= (b & 0x1f) << shift;
               shift += 5;
            } while (b >= 0x20);
            int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
            lng += dlng;

            LatLng position = new LatLng((double) lat / 1E5, (double) lng / 1E5);
            poly.add(position);
         }
         return poly;
      }
   }
```

HistoryActivity.java

```
package my.application.containertracker;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;

public class HistoryActivity extends ListActivity {
        // Progress Dialog
        private ProgressDialog pDialog;
        // Creating JSON Parser object
        JSONParser jParser = new JSONParser();
```

```
ArrayList<HashMap<String, String>> historyList;

// JSON Node names
private static final String TAG_SUCCESS = "success";
private static final String TAG_HISTORY = "history";
private static final String TAG_ID = "package_id";
private static final String TAG_MESSAGE = "message";

private String device_id;

// products JSONArray
JSONArray history = null;

// url to get all products list
private static String url_history = ServerConnection.getServerUrl() +
"gethistory.php";

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

    // get device id
    TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
    device_id = telephonyManager.getDeviceId();

    // Hashmap for ListView
    historyList = new ArrayList<HashMap<String, String>>();

    // Loading products in Background Thread
    new LoadHistory().execute();
```

```
                // Get listview
                ListView lv = getListView();
                // on seleting single history
                // launching MapActivity
                lv.setOnItemClickListener(new OnItemClickListener() {

                        @Override
                        public void onItemClick(AdapterView<?> parent, View view,
                                        int position, long id) {
                                // getting values from selected ListItem
                                String package_id = ((TextView)
view.findViewById(R.id.id))
                                                .getText().toString();

                                // Starting new intent
                                Intent intent = new Intent(getApplicationContext(),
                                                MapActivity.class);
                                // sending pid to next activity
                                intent.putExtra("id", package_id);

                                // starting new activity and expecting some response back
                                startActivityForResult(intent, 100);
                        }
                });
        }

        /**
         * Background Async Task to Load history by making HTTP Request
         * */
        class LoadHistory extends AsyncTask<String, String, String> {
                int success;
                String message;

                /**
```

```
         * Before starting background thread Show Progress Dialog
         * */
        @Override
        protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(HistoryActivity.this);
                pDialog.setMessage("Loading history. Please wait...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

        /**
         * getting history from url
         * */
        protected String doInBackground(String... args) {
                // Building Parameters
                List<NameValuePair> params = new
ArrayList<NameValuePair>();
                params.add(new BasicNameValuePair("device_id",
device_id));
                // getting JSON string from URL
                JSONObject json = jParser.makeHttpRequest(url_history,
"GET",
                                params);

                if (json == null) {
                        success = 0;
                        message = "Could not connect to server";
                } else {
                        try {
                                // Checking for SUCCESS TAG
                                int success = json.getInt(TAG_SUCCESS);
```

```
                    if (success == 1) {
                        // history found
                        // Getting Array of history
                        history = json.getJSONArray(TAG_HISTORY);

                        // looping through history
                        for (int i = 0; i < history.length(); i++) {
                            JSONObject c = history.getJSONObject(i);

                            // Storing each json item in variable
                            String id = c.getString(TAG_ID);

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

                            // adding each child node to HashMap key => value
                            map.put(TAG_ID, id);
                            map.put(TAG_ID, id);

                            // adding HashList to ArrayList
                            historyList.add(map);
                        }
                    }else {
                        message = json.getString(TAG_MESSAGE);
                    }
                } catch (JSONException e) {
                    e.printStackTrace();
                }
```

```
                }
                return null;
            }

            /**
             * After completing background task Dismiss the progress dialog
             * **/
            protected void onPostExecute(String file_url) {
                // dismiss the dialog after getting all products
                pDialog.dismiss();

                // updating UI from Background Thread
                runOnUiThread(new Runnable() {
                    public void run() {
                        /**
                         * Updating parsed JSON data into ListView
                         * */
                        ListAdapter adapter = new SimpleAdapter(
                                HistoryActivity.this, historyList,
                                R.layout.container_item,      new
String[] { TAG_ID,
                                                              TAG_ID   },   new
int[] { R.id.id, R.id.id });
                        // updating listview
                        setListAdapter(adapter);
                    }
                });
            }
        }

}
```

InfoActivity.java

```
package my.application.containertracker;

import my.application.containertracker.weatherapp.JSONWeatherParser;
import my.application.containertracker.weatherapp.WeatherHttpClient;
import my.application.containertracker.weatherapp.model.Weather;

import org.json.JSONException;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.view.Menu;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

public class InfoActivity extends Activity {
        private TextView cityText;
        private TextView condDescr;
        private TextView temp;
        private TextView press;
        private TextView windSpeed;
        private TextView windDeg;
        private TextView hum;
        private ImageView imgView;

        private String curWeatherCoordinates;
        private String destWeatherCoordinates;
```

```
    private String id;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_info);
        TextView infoTv = ((TextView) findViewById(R.id.textViewInfo));
        infoTv.setMovementMethod(new ScrollingMovementMethod());

        Intent intent = getIntent();
        ((TextView) findViewById(R.id.textViewOrigin)).setText(intent
                .getStringExtra("origin"));
        ((TextView) findViewById(R.id.textViewOriginAddr)).setText(intent
                .getStringExtra("originAddress"));
        ((TextView) findViewById(R.id.TextViewDestination)).setText(intent
                .getStringExtra("destination"));
        ((TextView)
findViewById(R.id.textViewDestinationAddr)).setText(intent
                .getStringExtra("destinationAddress"));
        ((TextView) findViewById(R.id.TextViewCurrent)).setText(intent
                .getStringExtra("current"));
        ((TextView) findViewById(R.id.textViewCurrentAddr)).setText(intent
                .getStringExtra("currentAddress"));
        /*((TextView) findViewById(R.id.TextViewStopOver)).setText(intent
                .getStringExtra("stopover"));*/
        ((TextView) findViewById(R.id.textViewDepartTime)).setText(intent
                .getStringExtra("departure"));
        ((TextView) findViewById(R.id.TextViewArrival)).setText(intent
                .getStringExtra("arrival"));
        ((TextView) findViewById(R.id.TextViewMedium)).setText(intent
                .getStringExtra("medium"));
        infoTv.setText(intent.getStringExtra("info"));

        id = intent.getStringExtra("id");
```

```
            curWeatherCoordinates =
intent.getStringExtra("curWeatherCoordinates");
            destWeatherCoordinates =
intent.getStringExtra("destWeatherCoordinates");

    }
    public void openCurWeather(View view){
        Intent intent = new Intent(this, WeatherActivity.class);
        intent.putExtra("weatherCoordinates", curWeatherCoordinates);
        startActivity(intent);
    }
    public void openDestWeather(View view){
        Intent intent = new Intent(this, WeatherActivity.class);
        intent.putExtra("weatherCoordinates", destWeatherCoordinates);
        startActivity(intent);
    }
    public void openStopover(View view){
        Intent intent = new Intent(this, StopoversActivity.class);
        intent.putExtra("id", id);
        startActivity(intent);
    }
    public void closeInfo(View view) {
        this.finish();
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.info, menu);
        return true;
    }

}
```

JSONParser.java

```
package my.application.containertracker;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.utils.URLEncodedUtils;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;

public class JSONParser {

    static InputStream is = null;
    static JSONObject jObj = null;
    static String json = "";

    // constructor
    public JSONParser() {
```

```
}

// function get json from url
// by making HTTP POST or GET mehtod
public JSONObject makeHttpRequest(String url, String method,
        List<NameValuePair> params) {

    // Making HTTP request
    try {

        // check for request method
        if(method == "POST"){
            // request method is POST
            // defaultHttpClient
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpPost httpPost = new HttpPost(url);
            httpPost.setEntity(new UrlEncodedFormEntity(params));

            HttpResponse httpResponse = httpClient.execute(httpPost);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();

        }else if(method == "GET"){
            // request method is GET
            DefaultHttpClient httpClient = new DefaultHttpClient();
            String paramString = URLEncodedUtils.format(params, "utf-8");
            url += "?" + paramString;
            HttpGet httpGet = new HttpGet(url);

            HttpResponse httpResponse = httpClient.execute(httpGet);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();
        }
```

```
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

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

        // try parse the string to a JSON object
        try {
            jObj = new JSONObject(json);
        } catch (JSONException e) {
            Log.e("JSON Parser", "Error parsing data " + e.toString());
        }
        // return JSON String
        return jObj;
    }
}
```

MapActivity.java

```
package my.application.containertracker;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import my.application.containertracker.R;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.provider.Settings.Secure;
import android.telephony.TelephonyManager;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.PopupWindow;
```

```
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

public class MapActivity extends Activity {
        // Google Map
        private GoogleMap googleMap;
        private String device_id;
        public LatLng originLatLng;
        public String originAddress;
        public LatLng destinationLatLng;
        public String destinationAddress;
        public LatLng currentLatLng;
        public String currentAddress;
        public String stopovers;
        public String departTime;
        public String arriveTime;
        public String medium;
        public String info;

        private Polyline newPolyline;

        String id;
        // Progress Dialog
        private ProgressDialog pDialog;
        // JSON parser class
```

```
        JSONParser jsonParser = new JSONParser();
        // service url
        private static final String url_container_details =
ServerConnection.getServerUrl() + "getpackageinfo.php";

        // JSON Node names
        private static final String TAG_SUCCESS = "success";
        private static final String TAG_PACKAGE = "package";
        private static final String TAG_ID = "id";
        private static final String TAG_MESSAGE = "message";
        private static final String TAG_ORIGIN_LAT = "origin_lat";
        private static final String TAG_ORIGIN_LONG = "origin_long";
        private static final String TAG_ORIGIN_ADDR = "origin_address";
        private static final String TAG_DESTINATION_LAT = "destination_lat";
        private static final String TAG_DESTINATION_LONG = "destination_long";
        private static final String TAG_DESTINATION_ADDR =
"destination_address";
        private static final String TAG_CURRENT_LAT = "current_lat";
        private static final String TAG_CURRENT_LONG = "current_long";
        private static final String TAG_CURRENT_ADDR = "current_address";
        private static final String TAG_STOPOVER = "stopover";
        private static final String TAG_DEPART_TIME = "depart_time";
        private static final String TAG_ARRIVE_TIME = "arrive_time";
        private static final String TAG_MEDIUM = "medium";
        private static final String TAG_INFO = "info";

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

                //get device id
                TelephonyManager telephonyManager =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
```

```
            device_id = telephonyManager.getDeviceId();

            Intent intent = getIntent();

            id = intent.getStringExtra("id");

            new GetContainerDetails().execute();

        }

        public void initializeMap() {
            ((Button)findViewById(R.id.btnInfo)).setEnabled(true);
            if (googleMap == null) {
                googleMap = ((MapFragment)
getFragmentManager().findFragmentById(
                        R.id.map)).getMap();

                // check if map is created successfully or not
                if (googleMap == null) {
                    Toast.makeText(getApplicationContext(),
                            "Sorry! unable to create maps",
Toast.LENGTH_SHORT)
                            .show();
                } else {
                    // create marker
                    MarkerOptions originMarker = new
MarkerOptions().position(
                            originLatLng).title("Origin");
                    originMarker.icon(BitmapDescriptorFactory

        .defaultMarker(BitmapDescriptorFactory.HUE_GREEN));
                    MarkerOptions destinationMarker = new
MarkerOptions().position(
                            destinationLatLng).title("Destination");
```

```
                MarkerOptions currentMarker = new
MarkerOptions().position(
                        currentLatLng).title("Current position");
                currentMarker.icon(BitmapDescriptorFactory

        .defaultMarker(BitmapDescriptorFactory.HUE_ORANGE));

                googleMap.addMarker(originMarker);
                googleMap.addMarker(destinationMarker);

        googleMap.addMarker(currentMarker).showInfoWindow();

                findDirections(originLatLng.latitude,
originLatLng.longitude,
                        destinationLatLng.latitude,
                        destinationLatLng.longitude,
                        GMapV2Direction.MODE_DRIVING);

                CameraPosition cameraPosition = new
CameraPosition.Builder()
                        .target(currentLatLng).zoom(7).build();

                googleMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(cameraPosition));
                }
            }
        }

        public void handleGetDirectionsResult(ArrayList<LatLng> directionPoints) {
            PolylineOptions rectLine = new PolylineOptions().width(5).color(
                    Color.RED);

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

```
                rectLine.add(directionPoints.get(i));
            }
            if (newPolyline != null) {
                newPolyline.remove();
            }
            newPolyline = googleMap.addPolyline(rectLine);
        }

        public void findDirections(double fromPositionDoubleLat,
                double fromPositionDoubleLong, double toPositionDoubleLat,
                double toPositionDoubleLong, String mode) {
            Map<String, String> map = new HashMap<String, String>();
            map.put(GetDirectionsAsyncTask.USER_CURRENT_LAT,
                    String.valueOf(fromPositionDoubleLat));
            map.put(GetDirectionsAsyncTask.USER_CURRENT_LONG,
                    String.valueOf(fromPositionDoubleLong));
            map.put(GetDirectionsAsyncTask.DESTINATION_LAT,
                    String.valueOf(toPositionDoubleLat));
            map.put(GetDirectionsAsyncTask.DESTINATION_LONG,
                    String.valueOf(toPositionDoubleLong));
            map.put(GetDirectionsAsyncTask.DIRECTIONS_MODE, mode);
            map.put(GetDirectionsAsyncTask.STOPOVERS, stopovers);

            GetDirectionsAsyncTask asyncTask = new
GetDirectionsAsyncTask(this);
            asyncTask.execute(map);
        }

        @Override
        protected void onResume() {
            super.onResume();
            // initializeMap();
        }
```

```
    public void openInfo(View view) {
        Intent intent = new Intent(this, InfoActivity.class);
        intent.putExtra("id", id);
        intent.putExtra("origin", originLatLng.latitude + ", " +
originLatLng.longitude);
        intent.putExtra("originAddress", originAddress);
        intent.putExtra("destination", destinationLatLng.latitude + ", " +
destinationLatLng.longitude);
        intent.putExtra("destinationAddress", destinationAddress);
        intent.putExtra("current", currentLatLng.latitude + ", " +
currentLatLng.longitude);
        intent.putExtra("currentAddress", currentAddress);
        /*if(stopoverLatLng.latitude == 0 && stopoverLatLng.longitude == 0)
            intent.putExtra("stopover","-");
        else
            intent.putExtra("stopover", stopoverLatLng.latitude + ", " +
stopoverLatLng.longitude);*/
        intent.putExtra("departure", departTime);
        intent.putExtra("arrival", arriveTime);
        intent.putExtra("medium", medium);
        intent.putExtra("info", info);
        intent.putExtra("curWeatherCoordinates",
"lat="+currentLatLng.latitude+"&lon="+currentLatLng.longitude);
        intent.putExtra("destWeatherCoordinates",
"lat="+destinationLatLng.latitude+"&lon="+destinationLatLng.longitude);
        startActivity(intent);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.map, menu);
        return true;
```

```
    }

    /**
     * Retrieves container info from server
     *
     */
    class GetContainerDetails extends AsyncTask<String, String, String> {
        int success;
        String message;

        /**
         * Before starting background thread Show Progress Dialog
         * */
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(MapActivity.this);
            pDialog.setMessage("Loading container details. Please wait...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        /**
         * Getting container details in background thread
         * */
        protected String doInBackground(String... params1) {
            try {
                // Building Parameters
                ArrayList<NameValuePair> params = new
ArrayList<NameValuePair>();
                params.add(new BasicNameValuePair("id", id));
                params.add(new BasicNameValuePair("device_id",
device_id));
```

```
                    // getting container details by making HTTP request
                    // Note that container details url will use GET request
                    JSONObject json = jsonParser.makeHttpRequest(
                            url_container_details, "GET", params);

                    if (json == null) {
                        success = 0;
                        message = "Could not connect to server";
                    } else {
                        // json success tag
                        success = json.getInt(TAG_SUCCESS);
                        if (success == 1) {
                            // successfully received container details
                            JSONArray packageObj =
json.getJSONArray(TAG_PACKAGE); // JSON
                                                    // Array

                            // get first package object from JSON
Array
                            JSONObject packageVar =
packageObj.getJSONObject(0);

                            // package with this id found
                            originLatLng = new LatLng(

      packageVar.getDouble(TAG_ORIGIN_LAT),

      packageVar.getDouble(TAG_ORIGIN_LONG));
                            originAddress =
packageVar.getString(TAG_ORIGIN_ADDR);
                            destinationLatLng = new LatLng(
```

```
                packageVar.getDouble(TAG_DESTINATION_LAT),

                packageVar.getDouble(TAG_DESTINATION_LONG));
                                        destinationAddress =
packageVar.getString(TAG_DESTINATION_ADDR);
                                        currentLatLng = new LatLng(

                packageVar.getDouble(TAG_CURRENT_LAT),

                packageVar.getDouble(TAG_CURRENT_LONG));
                                        currentAddress =
packageVar.getString(TAG_CURRENT_ADDR);
                                        stopovers =
packageVar.getString(TAG_STOPOVER);
                                        stopovers += "|" + currentLatLng.latitude
+ "," + currentLatLng.longitude;

                                        departTime =
packageVar.getString(TAG_DEPART_TIME);
                                        arriveTime =
packageVar.getString(TAG_ARRIVE_TIME);
                                        medium =
packageVar.getString(TAG_MEDIUM);
                                        info =
packageVar.getString(TAG_INFO);

                                    } else {
                                        message =
json.getString(TAG_MESSAGE);
                                    }
                                }
                            } catch (JSONException e) {
                                e.printStackTrace();
```

```
                }

                return null;
            }

            /**
             * After completing background task Dismiss the progress dialog and
             * initialize map or print error message
             *
             **/
            protected void onPostExecute(String file_url) {
                // dismiss the dialog once got all details
                pDialog.dismiss();
                if (success != 1) {
                    Toast.makeText(getApplicationContext(), message,
                            Toast.LENGTH_LONG).show();
                } else {
                    initializeMap();
                }
            }
        }
}
```

MenuActivity.java

```
package my.application.containertracker;

import my.application.containertracker.R;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;

public class MenuActivity extends Activity {

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

	public void openSearch(View view) {
		Intent intent = new Intent(this, SearchActivity.class);
		startActivity(intent);
	}

	public void openHistory(View view) {
		Intent intent = new Intent(this, MyContainersActivity.class);
		startActivity(intent);
	}

	public void openAbout(View view) {
		Intent intent = new Intent(this, AboutActivity.class);
		startActivity(intent);
	}
```

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }

}
```

MyContainersActivity.java

```
package my.application.containertracker;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.GoogleMap.OnInfoWindowClickListener;
import com.google.android.gms.maps.GoogleMap.OnMapClickListener;
import com.google.android.gms.maps.GoogleMap.OnMarkerClickListener;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.view.View;
```

```
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.Toast;

public class MyContainersActivity extends Activity implements
OnInfoWindowClickListener{
        private GoogleMap googleMap;
        // Progress Dialog
        private ProgressDialog pDialog;
        // Creating JSON Parser object
        JSONParser jParser = new JSONParser();

        ArrayList<HashMap<String, Object>> historyList;

        // JSON Node names
        private static final String TAG_SUCCESS = "success";
        private static final String TAG_HISTORY = "history";
        private static final String TAG_ID = "package_id";
        private static final String TAG_CURRENT_LAT = "current_lat";
        private static final String TAG_CURRENT_LONG = "current_long";
        private static final String TAG_CURRENT_LATLONG = "current_latlong";
        private static final String TAG_CURRENT_ADDR = "current_address";
        private static final String TAG_MESSAGE = "message";

        private String device_id;

        // products JSONArray
        JSONArray history = null;
```

```
        // url to get all products list
        private static String url_history = ServerConnection.getServerUrl() +
"gethistory.php";

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

                // get device id
                TelephonyManager telephonyManager = (TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE);
                device_id = telephonyManager.getDeviceId();

                // Hashmap for ListView
                historyList = new ArrayList<HashMap<String, Object>>();

                // Loading products in Background Thread
                new LoadMyActivities().execute();

                /*
                 * // Get listview ListView lv = getListView(); // on seleting single
                 * history // launching MapActivity lv.setOnItemClickListener(new
                 * OnItemClickListener() {
                 *
                 * @Override public void onItemClick(AdapterView<?> parent, View
view,
                 * int position, long id) { // getting values from selected ListItem
                 * String package_id = ((TextView) view.findViewById(R.id.id))
                 * .getText().toString();
                 *
                 * // Starting new intent Intent intent = new
                 * Intent(getApplicationContext(), MapActivity.class); // sending pid to
                 * next activity intent.putExtra("id", package_id);
```

```
                *
                * // starting new activity and expecting some response back
                * startActivityForResult(intent, 100); } });
                */
        }

        public void initializeMap() {
                if (googleMap == null) {
                        googleMap = ((MapFragment)
getFragmentManager().findFragmentById(
                                        R.id.my_containers_map)).getMap();

                        // check if map is created successfully or not
                        if (googleMap == null) {
                                Toast.makeText(getApplicationContext(),
                                                "Sorry! unable to create maps",
Toast.LENGTH_SHORT)
                                                .show();
                        } else {
                                googleMap.setOnInfoWindowClickListener(this);

                                for (int i = 0; i < historyList.size(); i++) {
                                        HashMap<String, Object> map =
historyList.get(i);

                                        MarkerOptions newMarkerOpt = new
MarkerOptions().position(
                                                        (LatLng)
map.get(TAG_CURRENT_LATLONG)).title((String)map.get(TAG_ID)).snippet((Strin
g)map.get(TAG_CURRENT_ADDR));
                                        newMarkerOpt.icon(BitmapDescriptorFactory

        .defaultMarker(BitmapDescriptorFactory.HUE_ORANGE));
```

```
                    Marker newMarker =
googleMap.addMarker(newMarkerOpt);

                }



                CameraPosition cameraPosition = new
CameraPosition.Builder()
                        .target(new LatLng(38.281384,
23.797483)).zoom(4).build();

                googleMap.animateCamera(CameraUpdateFactory
                        .newCameraPosition(cameraPosition));
            }
        }
    }

    @Override
    public void onInfoWindowClick(Marker marker) {
        // Starting new intent
        Intent intent = new Intent(getApplicationContext(),
                MapActivity.class);
        // sending pid to next activity
        intent.putExtra("id", marker.getTitle());
        // starting new activity and expecting some response back
        startActivityForResult(intent, 100);
    }

    /**
     * Background Async Task to Load history by making HTTP Request
     * */
    class LoadMyActivities extends AsyncTask<String, String, String> {
        int success;
```

```
String message;

/**
 * Before starting background thread Show Progress Dialog
 * */
@Override
protected void onPreExecute() {
        super.onPreExecute();
        pDialog = new ProgressDialog(MyContainersActivity.this);
        pDialog.setMessage("Loading history. Please wait...");
        pDialog.setIndeterminate(false);
        pDialog.setCancelable(false);
        pDialog.show();
}

/**
 * getting history from url
 * */
protected String doInBackground(String... args) {
        // Building Parameters
        List<NameValuePair> params = new
ArrayList<NameValuePair>();
        params.add(new BasicNameValuePair("device_id",
device_id));
        // getting JSON string from URL
        JSONObject json = jParser.makeHttpRequest(url_history,
"GET",
                        params);

        if (json == null) {
                success = 0;
                message = "Could not connect to server";
        } else {
                try {
```

```
                    // Checking for SUCCESS TAG
                    success = json.getInt(TAG_SUCCESS);

                    if (success == 1) {
                        // history found
                        // Getting Array of history
                        history = json.getJSONArray(TAG_HISTORY);

                        // looping through history
                        for (int i = 0; i < history.length(); i++) {
                            JSONObject c = history.getJSONObject(i);

                            // Storing each json item in variable
                            String id = c.getString(TAG_ID);
                            Double current_lat = c.getDouble(TAG_CURRENT_LAT);
                            Double current_long = c.getDouble(TAG_CURRENT_LONG);
                            String current_address = c.getString(TAG_CURRENT_ADDR);

                            // creating new HashMap
                            HashMap<String, Object> map = new HashMap<String, Object>();

                            // adding each child node to HashMap key => value
                            map.put(TAG_ID, id);

                            map.put(TAG_CURRENT_LATLONG, new LatLng(
```

```
                                                    current_lat,
current_long));

        map.put(TAG_CURRENT_ADDR, current_address);

                                                    // adding HashList to ArrayList
                                                    historyList.add(map);
                                                }
                                            } else {
                                                    message                         =
json.getString(TAG_MESSAGE);
                                            }
                                        } catch (JSONException e) {
                                                e.printStackTrace();
                                        }
                                }
                                return null;
                        }
                        /**
                         * After completing background task Dismiss the progress dialog
                         * **/
                        protected void onPostExecute(String file_url) {
                                // dismiss the dialog once got all details
                                pDialog.dismiss();
                                if (success != 1) {
                                        Toast.makeText(getApplicationContext(), message,
                                                        Toast.LENGTH_LONG).show();
                                } else {
                                        initializeMap();
                                }
                        }
                }
}
```

SearchActivity.java

```
package my.application.containertracker;

import my.application.containertracker.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class SearchActivity extends Activity {

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

    public void openMap(View view) {
        String package_id = ((EditText)
findViewById(R.id.editTextPackageId))
                .getText().toString();
        String destination = ((EditText)
findViewById(R.id.EditTextDestination))
                .getText().toString();
        String origin = ((EditText) findViewById(R.id.EditTextOrigin))
                .getText().toString();
        String stopover = ((EditText) findViewById(R.id.EditTextStopover))
                .getText().toString();

        Intent intent = new Intent(this, SearchResultsActivity.class);
```

```
            intent.putExtra("package_id", package_id);
            intent.putExtra("origin", origin);
            intent.putExtra("destination", destination);
            intent.putExtra("stopover", stopover);
            startActivity(intent);
        }

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.search, menu);
            return true;
        }

}
```

SearchResultsActivity.java

```
package my.application.containertracker;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;

public class SearchResultsActivity extends ListActivity {
    // Progress Dialog
    private ProgressDialog pDialog;
    // Creating JSON Parser object
    JSONParser jParser = new JSONParser();

    ArrayList<HashMap<String, String>> resultsList;
```

```
        public String package_id;
        public String origin;
        public String destination;
        public String stopover;

        // JSON Node names
        private static final String TAG_SUCCESS = "success";
        private static final String TAG_HISTORY = "results";
        private static final String TAG_ID = "id";
        private static final String TAG_MESSAGE = "message";

        JSONArray results = null;

        // url to get all products list
        private static String url_search = ServerConnection.getServerUrl() +
"searchcontainers.php";

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

                Intent intent = getIntent();
                package_id = intent.getStringExtra("package_id");
                origin = intent.getStringExtra("origin");
                destination = intent.getStringExtra("destination");
                stopover = intent.getStringExtra("stopover");

                // Hashmap for ListView
                resultsList = new ArrayList<HashMap<String, String>>();

                // Loading products in Background Thread
                new LoadResults().execute();
```

```
        // Get listview
        ListView lv = getListView();
        // on seleting single history
        // launching MapActivity
        lv.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
                // getting values from selected ListItem
                String package_id = ((TextView)
view.findViewById(R.id.id))
                        .getText().toString();

                // Starting new intent
                Intent intent = new Intent(getApplicationContext(),
                        MapActivity.class);
                // sending pid to next activity
                intent.putExtra("id", package_id);

                // starting new activity and expecting some response back
                startActivityForResult(intent, 100);
            }
        });
    }

    /**
     * Background Async Task to Load history by making HTTP Request
     * */
    class LoadResults extends AsyncTask<String, String, String> {
        int success;
        String message;

        /**
```

```
         * Before starting background thread Show Progress Dialog
         * */
        @Override
        protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(SearchResultsActivity.this);
                pDialog.setMessage("Searching. Please wait...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

        /**
         * getting history from url
         * */
        protected String doInBackground(String... args) {
                // Building Parameters
                List<NameValuePair> params = new
ArrayList<NameValuePair>();
                params.add(new BasicNameValuePair("id", package_id));
                params.add(new BasicNameValuePair("origin", origin));
                params.add(new BasicNameValuePair("destination",
destination));
                params.add(new BasicNameValuePair("stopover", stopover));
                // getting JSON string from URL
                JSONObject json = jParser.makeHttpRequest(url_search,
"GET",
                                params);

                if (json == null) {
                        success = 0;
                        message = "Could not connect to server";
                } else {
                        try {
```

```
                    // Checking for SUCCESS TAG
                    int success = json.getInt(TAG_SUCCESS);

                    if (success == 1) {
                        // history found
                        // Getting Array of history
                        results = json.getJSONArray(TAG_HISTORY);

                        // looping through history
                        for (int i = 0; i < results.length(); i++) {
                            JSONObject c = results.getJSONObject(i);
                            // Storing each json item in variable
                            String id = c.getString(TAG_ID);

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

                            // adding each child node to HashMap key => value
                            map.put(TAG_ID, id);
                            map.put(TAG_ID, id);

                            // adding HashList to ArrayList
                            resultsList.add(map);
                        }
                    }else {
                        message = json.getString(TAG_MESSAGE);
                    }
                } catch (JSONException e) {
```

```
                    e.printStackTrace();
                }
            }
            return null;
        }
        /**
         * After completing background task Dismiss the progress dialog
         * **/
        protected void onPostExecute(String file_url) {
            // dismiss the dialog after getting all products
            pDialog.dismiss();

            // updating UI from Background Thread
            runOnUiThread(new Runnable() {
                public void run() {
                    /**
                     * Updating parsed JSON data into ListView
                     * */
                    ListAdapter adapter = new SimpleAdapter(
                            SearchResultsActivity.this,
resultsList,
                            R.layout.container_item,      new
String[] { TAG_ID,
                                    TAG_ID   },   new
int[] { R.id.id, R.id.id });
                    // updating listview
                    setListAdapter(adapter);
                }
            });
        }
    }

}
```

ServerConnection.java

```
package my.application.containertracker;

public class ServerConnection {
	private static String server_url = "http://192.168.2.4/packagetracker/";

	public static String getServerUrl(){
		return server_url;
	}
}
```

StopoversActivity.java

```
package my.application.containertracker;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import my.application.containertracker.HistoryActivity.LoadHistory;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.ListActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.AsyncTask;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

public class StopoversActivity extends ListActivity {
        // Progress Dialog
```

```
    private ProgressDialog pDialog;
    // Creating JSON Parser object
    JSONParser jParser = new JSONParser();

    ArrayList<HashMap<String, String>> stopoverList;

    // JSON Node names
    private static final String TAG_SUCCESS = "success";
    private static final String TAG_STOPOVER = "stopover";
    private static final String TAG_ID = "package_id";
    private static final String TAG_LAT = "lat";
    private static final String TAG_LONG = "long";
    private static final String TAG_LATLONG = "latlong";
    private static final String TAG_ADDRESS = "address";
    private static final String TAG_ARRIVE_TIME = "arrive_time";
    private static final String TAG_DEPART_TIME = "depart_time";
    private static final String TAG_MESSAGE = "message";

    private String package_id;

    // products JSONArray
    JSONArray history = null;

    // url to get all products list
    private static String url_stopover = ServerConnection.getServerUrl() +
"getstopover.php";

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

        Intent intent = getIntent();
        package_id = intent.getStringExtra("id");
```

```
        // Hashmap for ListView
        stopoverList = new ArrayList<HashMap<String, String>>();

        // Loading products in Background Thread
        new LoadStopover().execute();
}

/**
 * Background Async Task to Load history by making HTTP Request
 * */
class LoadStopover extends AsyncTask<String, String, String> {
        int success;
        String message;

        /**
         * Before starting background thread Show Progress Dialog
         * */
        @Override
        protected void onPreExecute() {
                super.onPreExecute();
                pDialog = new ProgressDialog(StopoversActivity.this);
                pDialog.setMessage("Loading stopovers. Please wait...");
                pDialog.setIndeterminate(false);
                pDialog.setCancelable(false);
                pDialog.show();
        }

        /**
         * getting history from url
         * */
        protected String doInBackground(String... args) {
                // Building Parameters
```

```
                List<NameValuePair> params = new
ArrayList<NameValuePair>();
                params.add(new BasicNameValuePair("id", package_id));
                // getting JSON string from URL
                JSONObject json = jParser.makeHttpRequest(url_stopover,
"GET",
                        params);

                if (json == null) {
                    success = 0;
                    message = "Could not connect to server";
                } else {
                    try {
                        // Checking for SUCCESS TAG
                        int success = json.getInt(TAG_SUCCESS);

                        if (success == 1) {
                            // history found
                            // Getting Array of history
                            history =
json.getJSONArray(TAG_STOPOVER);

                            // looping through history
                            for (int i = 0; i < history.length(); i++) {
                                JSONObject c =
history.getJSONObject(i);

                                // Storing each json item in
variable
                                String lat =
c.getString(TAG_LAT);
                                String lon =
c.getString(TAG_LONG);
```

```
                                String address =
c.getString(TAG_ADDRESS);
                                String arrive_time =
c.getString(TAG_ARRIVE_TIME);
                                String depart_time =
c.getString(TAG_DEPART_TIME);

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

                                // adding each child node to
HashMap key => value
                                map.put(TAG_LATLONG, lat +
"," + lon);
                                map.put(TAG_ADDRESS,
address);
                                map.put(TAG_ARRIVE_TIME,
"Arrival: " + arrive_time);
                                map.put(TAG_DEPART_TIME,
"Departure: " + depart_time);

                                // adding HashList to ArrayList
                                stopoverList.add(map);
                            }
                        }else {
                            message =
json.getString(TAG_MESSAGE);
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
                return null;
```

```
        }

        /**
         * After completing background task Dismiss the progress dialog
         * **/
        protected void onPostExecute(String file_url) {
            // dismiss the dialog after getting all products
            pDialog.dismiss();

            // updating UI from Background Thread
            runOnUiThread(new Runnable() {
                public void run() {
                    /**
                     * Updating parsed JSON data into ListView
                     * */
                    ListAdapter adapter = new SimpleAdapter(
                            StopoversActivity.this,
stopoverList,
                            R.layout.stopover_item,      new
String[] { TAG_ADDRESS,
                                                    TAG_LATLONG,
TAG_ARRIVE_TIME, TAG_DEPART_TIME }, new int[] { R.id.address, R.id.latlon,
R.id.arrive_time, R.id.depart_time });
                    // updating listview
                    setListAdapter(adapter);
                }
            });

        }

    }
}
```

WeatherActivity.java

```
package my.application.containertracker;

import my.application.containertracker.weatherapp.JSONWeatherParser;
import my.application.containertracker.weatherapp.WeatherHttpClient;
import my.application.containertracker.weatherapp.model.Weather;

import org.json.JSONException;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

public class WeatherActivity extends Activity {
	private TextView cityText;
	private TextView condDescr;
	private TextView temp;
	private TextView press;
	private TextView windSpeed;
	private TextView windDeg;
	private TextView hum;
	private ImageView imgView;

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

```
        Intent intent = getIntent();

        cityText = (TextView) findViewById(R.id.cityText);
        condDescr = (TextView) findViewById(R.id.condDescr);
        temp = (TextView) findViewById(R.id.temp);
        hum = (TextView) findViewById(R.id.hum);
        press = (TextView) findViewById(R.id.press);
        windSpeed = (TextView) findViewById(R.id.windSpeed);
        windDeg = (TextView) findViewById(R.id.windDeg);
        imgView = (ImageView) findViewById(R.id.condIcon);

        JSONWeatherTask task = new JSONWeatherTask();
        task.execute(new
String[]{intent.getStringExtra("weatherCoordinates")});
    }

    private class JSONWeatherTask extends AsyncTask<String, Void, Weather>
{

        @Override
        protected Weather doInBackground(String... params) {
            Weather weather = new Weather();
            String data = ( (new
WeatherHttpClient()).getWeatherData(params[0]));

            try {
                weather = JSONWeatherParser.getWeather(data);

                // Let's retrieve the icon
                weather.iconData = ( (new
WeatherHttpClient()).getImage(weather.currentCondition.getIcon()));

            } catch (JSONException e) {
                e.printStackTrace();
```

```
                }
                return weather;

        }
                @Override
                protected void onPostExecute(Weather weather) {
                        super.onPostExecute(weather);

                        if (weather.iconData != null && weather.iconData.length > 0) {
                                Bitmap img =
BitmapFactory.decodeByteArray(weather.iconData, 0, weather.iconData.length);
                                imgView.setImageBitmap(img);
                        }

                        cityText.setText(weather.location.getCity() + "," +
weather.location.getCountry());
                        condDescr.setText(weather.currentCondition.getCondition() +
"(" + weather.currentCondition.getDescr() + ")");
                        temp.setText("" + Math.round((weather.temperature.getTemp()
- 273.15)) + "\u00B0C");
                        hum.setText("" + weather.currentCondition.getHumidity() +
"%");
                        press.setText("" + weather.currentCondition.getPressure() + "
hPa");
                        windSpeed.setText("" + weather.wind.getSpeed() + " mps");
                        windDeg.setText("" + weather.wind.getDeg() + "\u00B0");

                }
        }

}
```

WellcomeActivity.java

```
package my.application.containertracker;

import my.application.containertracker.R;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;

public class WellcomeActivity extends Activity {

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

	public void openMenu(View view) {
		Intent intent = new Intent(this, MenuActivity.class);
		startActivity(intent);
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.wellcome, menu);
		return true;
	}

}
```

JSonWeatherParserActivity.java

```
/**
 * This is a tutorial source code
 * provided "as is" and without warranties.
 *
 * For any question please visit the web site
 * http://www.survivingwithandroid.com
 *
 * or write an email to
 * survivingwithandroid@gmail.com
 *
 */
package my.application.containertracker.weatherapp;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import my.application.containertracker.weatherapp.model.Location;
import my.application.containertracker.weatherapp.model.Weather;

/*
 * Copyright (C) 2013 Surviving with Android (http://www.survivingwithandroid.com)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
```

```
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
public class JSONWeatherParser {

	public static Weather getWeather(String data) throws JSONException  {
		Weather weather = new Weather();

		// We create out JSONObject from the data
		JSONObject jObj = new JSONObject(data);

		// We start extracting the info
		Location loc = new Location();

		JSONObject coordObj = getObject("coord", jObj);
		loc.setLatitude(getFloat("lat", coordObj));
		loc.setLongitude(getFloat("lon", coordObj));

		JSONObject sysObj = getObject("sys", jObj);
		loc.setCountry(getString("country", sysObj));
		loc.setSunrise(getInt("sunrise", sysObj));
		loc.setSunset(getInt("sunset", sysObj));
		loc.setCity(getString("name", jObj));
		weather.location = loc;

		// We get weather info (This is an array)
		JSONArray jArr = jObj.getJSONArray("weather");

		// We use only the first value
		JSONObject JSONWeather = jArr.getJSONObject(0);
		weather.currentCondition.setWeatherId(getInt("id", JSONWeather));
```

```
            weather.currentCondition.setDescr(getString("description",
JSONWeather));
            weather.currentCondition.setCondition(getString("main",
JSONWeather));
            weather.currentCondition.setIcon(getString("icon", JSONWeather));

            JSONObject mainObj = getObject("main", jObj);
            weather.currentCondition.setHumidity(getInt("humidity", mainObj));
            weather.currentCondition.setPressure(getInt("pressure", mainObj));
            weather.temperature.setMaxTemp(getFloat("temp_max", mainObj));
            weather.temperature.setMinTemp(getFloat("temp_min", mainObj));
            weather.temperature.setTemp(getFloat("temp", mainObj));

            // Wind
            JSONObject wObj = getObject("wind", jObj);
            weather.wind.setSpeed(getFloat("speed", wObj));
            weather.wind.setDeg(getFloat("deg", wObj));

            // Clouds
            JSONObject cObj = getObject("clouds", jObj);
            weather.clouds.setPerc(getInt("all", cObj));

            // We download the icon to show


            return weather;
        }


        private static JSONObject getObject(String tagName, JSONObject jObj)
throws JSONException {
            JSONObject subObj = jObj.getJSONObject(tagName);
            return subObj;
        }
```

```
	private static String getString(String tagName, JSONObject jObj) throws JSONException {
		return jObj.getString(tagName);
	}

	private static float  getFloat(String tagName, JSONObject jObj) throws JSONException {
		return (float) jObj.getDouble(tagName);
	}

	private static int  getInt(String tagName, JSONObject jObj) throws JSONException {
		return jObj.getInt(tagName);
	}

}
```

WeatherHTTPClientActicity.java

```
package my.application.containertracker.weatherapp;

import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class WeatherHttpClient {

   private static String BASE_URL = "http://api.openweathermap.org/data/2.5/weather?";
   private static String IMG_URL = "http://openweathermap.org/img/w/";


   public String getWeatherData(String location) {
      HttpURLConnection con = null ;
      InputStream is = null;

      try {
         con = (HttpURLConnection) ( new URL(BASE_URL + location)).openConnection();
         con.setRequestMethod("GET");
         con.setDoInput(true);
         con.setDoOutput(true);
         con.connect();

         // Let's read the response
         StringBuffer buffer = new StringBuffer();
         is = con.getInputStream();
         BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

```
            String line = null;
            while (  (line = br.readLine()) != null )
                buffer.append(line + "\r\n");

            is.close();
            con.disconnect();
            return buffer.toString();
        }
        catch(Throwable t) {
            t.printStackTrace();
        }
        finally {
            try { is.close(); } catch(Throwable t) {}
            try { con.disconnect(); } catch(Throwable t) {}
        }

        return null;

    }

    public byte[] getImage(String code) {
        HttpURLConnection con = null ;
        InputStream is = null;
        try {
            con = (HttpURLConnection) ( new URL(IMG_URL + code)).openConnection();
            con.setRequestMethod("GET");
            con.setDoInput(true);
            con.setDoOutput(false);
            con.connect();

            // Let's read the response
            is = con.getInputStream();
            byte[] buffer = new byte[1024];
```

```
            ByteArrayOutputStream baos = new ByteArrayOutputStream();

            while ( is.read(buffer) != -1)
                baos.write(buffer);

            return baos.toByteArray();
        }
        catch(Throwable t) {
            t.printStackTrace();
        }
        finally {
            try { is.close(); } catch(Throwable t) {}
            try { con.disconnect(); } catch(Throwable t) {}
        }

        return null;

    }
}
```

Location.java

```
package my.application.containertracker.weatherapp.model;
/**
 * This is a tutorial source code
 * provided "as is" and without warranties.
 *
 * For any question please visit the web site
 * http://www.survivingwithandroid.com
 *
 * or write an email to
 * survivingwithandroid@gmail.com
 *
 */

import java.io.Serializable;
/*
 * Copyright (C) 2013 Surviving with Android (http://www.survivingwithandroid.com)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
public class Location implements Serializable {
```

```
private float longitude;
private float latitude;
private long sunset;
private long sunrise;
private String country;
private String city;

public float getLongitude() {
        return longitude;
}
public void setLongitude(float longitude) {
        this.longitude = longitude;
}
public float getLatitude() {
        return latitude;
}
public void setLatitude(float latitude) {
        this.latitude = latitude;
}
public long getSunset() {
        return sunset;
}
public void setSunset(long sunset) {
        this.sunset = sunset;
}
public long getSunrise() {
        return sunrise;
}
public void setSunrise(long sunrise) {
        this.sunrise = sunrise;
}
public String getCountry() {
        return country;
}
```

```
	public void setCountry(String country) {
		this.country = country;
	}
	public String getCity() {
		return city;
	}
	public void setCity(String city) {
		this.city = city;
	}

}
```

Weather.java

```
package my.application.containertracker.weatherapp.model;
/**
 * This is a tutorial source code
 * provided "as is" and without warranties.
 *
 * For any question please visit the web site
 * http://www.survivingwithandroid.com
 *
 * or write an email to
 * survivingwithandroid@gmail.com
 *
 */

/*
 * Copyright (C) 2013 Surviving with Android (http://www.survivingwithandroid.com)
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
public class Weather {

        public Location location;
```

```
public CurrentCondition currentCondition = new CurrentCondition();
public Temperature temperature = new Temperature();
public Wind wind = new Wind();
public Rain rain = new Rain();
public Snow snow = new Snow()      ;
public Clouds clouds = new Clouds();

public byte[] iconData;

public  class CurrentCondition {
        private int weatherId;
        private String condition;
        private String descr;
        private String icon;


        private float pressure;
        private float humidity;

        public int getWeatherId() {
                return weatherId;
        }
        public void setWeatherId(int weatherId) {
                this.weatherId = weatherId;
        }
        public String getCondition() {
                return condition;
        }
        public void setCondition(String condition) {
                this.condition = condition;
        }
        public String getDescr() {
                return descr;
        }
```

```
	public void setDescr(String descr) {
		this.descr = descr;
	}
	public String getIcon() {
		return icon;
	}
	public void setIcon(String icon) {
		this.icon = icon;
	}
	public float getPressure() {
		return pressure;
	}
	public void setPressure(float pressure) {
		this.pressure = pressure;
	}
	public float getHumidity() {
		return humidity;
	}
	public void setHumidity(float humidity) {
		this.humidity = humidity;
	}


}

public  class Temperature {
	private float temp;
	private float minTemp;
	private float maxTemp;

	public float getTemp() {
		return temp;
	}
	public void setTemp(float temp) {
```

```
		this.temp = temp;
	}
	public float getMinTemp() {
		return minTemp;
	}
	public void setMinTemp(float minTemp) {
		this.minTemp = minTemp;
	}
	public float getMaxTemp() {
		return maxTemp;
	}
	public void setMaxTemp(float maxTemp) {
		this.maxTemp = maxTemp;
	}

}

public  class Wind {
	private float speed;
	private float deg;
	public float getSpeed() {
		return speed;
	}
	public void setSpeed(float speed) {
		this.speed = speed;
	}
	public float getDeg() {
		return deg;
	}
	public void setDeg(float deg) {
		this.deg = deg;
	}
```

```
}

public  class Rain {
        private String time;
        private float ammount;
        public String getTime() {
                return time;
        }
        public void setTime(String time) {
                this.time = time;
        }
        public float getAmmount() {
                return ammount;
        }
        public void setAmmount(float ammount) {
                this.ammount = ammount;
        }



}

public  class Snow {
        private String time;
        private float ammount;

        public String getTime() {
                return time;
        }
        public void setTime(String time) {
                this.time = time;
        }
        public float getAmmount() {
                return ammount;
```

```
		}
		public void setAmmount(float ammount) {
			this.ammount = ammount;
		}


	}

	public  class Clouds {
		private int perc;

		public int getPerc() {
			return perc;
		}

		public void setPerc(int perc) {
			this.perc = perc;
		}


	}

}
```

Βιβλιογραφία

1. A.Silberschatz, H.Korth, S.Sudarsham, 2002, "Η πλήρης θεωρία των βάσεων δεδομένων, Συστήματα βάσεων δεδομένων", 4η έκδοση, Μ. Γκιούρδας
2. L.Welling, L.Thomson, 2002, "Ανάπτυξη Web Εφαρμογών με PHP και MySQL", Μ. Γκιούρδας
3. L.Darcey, S.Conder, 2011, "Ανάπτυξη Εφαρμογών με το Android", 2η έκδοση, Μ. Γκιούρδας
4. J. Simon, 2012, "Head First Android Development, A Learner's Guide to Creating Applications for Android Devices", O'Reilly Media
5. O. Cinar, 2012, "Android Apps with Eclipse", Apress
6. L. Richardson, S. Ruby, 2007, "RESTful Web Services, Web services for the real world", 2η έκδοση, O'Reilly Media
7. R. Meier, 2012, "Professional Android 4 Application Development", John Wiley & Sons

Internet Sites

1. http://en.wikipedia.org/wiki/Android_%28operating_system%29
2. http://www.allaboutandroid.gr/?p=6362
3. http://techcrunch.com/2014/05/06/android-still-growing-market-share-by-winning-first-time-smartphone-users/
4. http://www.businessinsider.com/iphone-v-android-market-share-2014-5
5. http://www.androidcentral.com/android-versions
6. http://www.hongkiat.com/blog/android-evolution/
7. https://developer.android.com/about/dashboards/index.html
8. http://ows.edb.utexas.edu/site/collaborative-bluetooth-edumanet/android-sdk-2