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

Mobile Client Για Τη Βέλτιστη Χρήση Των Αστικών Συγκοινωνιών Στην Περιοχή Της Αθήνας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΒΑΣΙΛΕΙΟΥ Β. ΚΟΥΡΤΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2012

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

Mobile Client Για Τη Βέλτιστη Χρήση Των Αστικών Συγκοινωνιών Στην Περιοχή Της Αθήνας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΒΑΣΙΛΕΙΟΥ Β. ΚΟΥΡΤΗ

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22η Οκτωβρίου 2012.

.....	.....	.....
Τιμολέων Σελλής Καθηγητής Ε.Μ.Π.	Ιωάννης Βασιλείου Καθηγητής Ε.Μ.Π.	Γιάννης Σταύρακας Ερευνητής Β' ΙΠΣΥ/Ε.Κ. "Αθηνά"

Αθήνα, Οκτώβριος 2012

.....

ΒΑΣΙΛΕΙΟΣ Β. ΚΟΥΡΤΗΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η εικόνα των συγκοινωνιών στο σύγχρονο αστικό περιβάλλον έχει αλλάξει τα τελευταία χρόνια. Οι κάτοικοι των μεγάλων αστικών κέντρων στρέφονται όλο και περισσότερο προς τις αστικές συγκοινωνίες προσπαθώντας να μειώσουν το φόρτο του οδικού δικτύου και να βελτιώσουν την ποιότητα ζωής τους. Τα μέσα μαζικής μεταφοράς εξελίσσονται ανάλογα, νέες γραμμές δημιουργούνται, παλιές επεκτείνονται και τα δρομολόγια αλλάζουν προκειμένου να εξυπηρετήσουν το συνεχώς αυξανόμενο επιβατικό κοινό. Η αναζήτηση του τρόπου μετάβασης από μία περιοχή σε κάποια άλλη αποτελεί πλέον ένα πολύπλοκο πρόβλημα που η επίλυσή του απαιτεί καλή γνώση του συγκοινωνιακού και οδικού δικτύου. Η ανάγκη αυτή, σε συνδυασμό με την εισαγωγή της τεχνολογίας και του διαδικτύου στην καθημερινότητά μας, έχει δημιουργήσει μια έντονη ερευνητική δραστηριότητα γύρω από τον τομέα των αστικών συγκοινωνιών και της αποδοτικής χρήσης αυτών. Έχουν αναπτυχθεί έξυπνοι αλγόριθμοι δρομολόγησης που βασίζονται σε ανοιχτά πρότυπα (Graphserver, OpenTripPlanner, GTFS κ.α.) και προσφέρουν αξιόπιστα αποτελέσματα βελτιστοποιώντας τη χρήση των αστικών συγκοινωνιών.

Στόχος της παρούσης εργασίας είναι η ανάπτυξη μιας εφαρμογής για κινητές συσκευές Android που, χρησιμοποιώντας τα ήδη υπάρχοντα δεδομένα αστικών συγκοινωνιών της Αθήνας και την ήδη υλοποιημένη server υποδομή του OpenTripPlanner, θα μπορεί να παρουσιάζει στον χρήστη της κινητής συσκευής τη βέλτιστη (λιγότερες μετεπιβιβάσεις, ταχύτερη) διαδρομή μεταξύ δύο σημείων, χρησιμοποιώντας αποκλειστικά αστικές συγκοινωνίες. Παράλληλα, με τη χρήση δεδομένων που βρίσκονται αποθηκευμένα τοπικά στη συσκευή, θα δίνεται η δυνατότητα παρουσίασης και οπτικοποίησης του χρονοδιαγράμματος των αστικών συγκοινωνιών στη διάρκεια μιας ημέρας, καθώς και η παρουσίαση αναλυτικών πληροφοριών για τις υπάρχουσες γραμμές και στάσεις.

Λέξεις Κλειδιά: αστικές συγκοινωνίες, αλγόριθμοι εύρεσης βέλτιστων μονοπατιών, γράφοι αναπαράστασης πολλαπλών μέσων, Graphserver, OpenTripPlanner, GTFS, OpenStreetMap, Android

Abstract

The usage of public transportation system in the modern urban environment has changed in recent years. The residents of urban centers are increasingly turning into public transport as their preference, trying to reduce the burden on the road network and improve their quality of life. Public transportation evolves accordingly, new lines are created, old are getting extended and schedules alter in order to serve the growing traveling public. Finding our way from one region of the city to another has become a complex problem which requires a good knowledge of both the public transit and the road network to get solved. This need, combined with the introduction of technology and the Internet in our daily lives, has created an intense research activity in the field of urban transport and its efficient use. There have been developed intelligent routing algorithms which are based on open standards (Graphserver, OpenTripPlanner, GTFS, etc.) and provide reliable results optimizing the use of public transportation.

The purpose of this thesis is to develop a mobile application for Android OS devices that, using existing data of urban transport in Athens and the already implemented server infrastructure of the OpenTripPlanner, will be able to provide the user of the mobile device with the optimal (less transit, faster) route between two points, using only public transport. Furthermore, using data saved locally on the device, the application will have the ability to present the timetable of urban transport and let the user explore lines and locate stops while minimizing network usage.

Keywords: public transport, shortest path algorithms, multimodal transport graphs, Graphserver, OpenTripPlanner, GTFS, OpenStreetMap, Android

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή μου, κύριο Τιμολέων Σελλή για την ευκαιρία που μου έδωσε να συνεργαστώ με το Ινστιτούτο Πληροφοριακών Συστημάτων και να ασχοληθώ με ένα πολύ ενδιαφέρον θέμα.

Επιπλέον, θα ήθελα να ευχαριστήσω τους Dieter Pfoser και Αλέξανδρο Εφεντάκη για την καθοδήγησή τους και την υποστήριξη που μου προσέφεραν κατά τη διάρκεια εκπόνησης της παρούσας εργασίας.

Τέλος, ευχαριστώ ιδιαίτερα την οικογένειά μου για την υλική και ηθική στήριξη που μου προσέφερε κατά τη διάρκεια των σπουδών μου, αλλά και τους φίλους και συναδέλφους μου για τη συνεργασία και τη φιλία τους που έκαναν αυτά τα χρόνια ακόμη πιο ενδιαφέροντα.

Πίνακας περιεχομένων

1

Εισαγωγή

1.1 Βελτιστοποίηση της χρήσης αστικών συγκοινωνιών με χρήση έξυπνων τηλεφώνων

Τα τελευταία χρόνια έχει γίνει σημαντική δουλειά στον τομέα της βελτιστοποίησης της χρήσης των αστικών συγκοινωνιών με χρήση της τεχνολογίας. Έχουν αναπτυχθεί σημαντικοί αλγόριθμοι και εφαρμογές που βοηθούν του κατοίκους των πυκνοκατοικημένων αστικών κέντρων να χρησιμοποιήσουν τα μέσα μαζικής μεταφοράς, μειώνοντας έτσι τον φόρτο του οδικού δικτύου και βελτιώνοντας την ποιότητα ζωής τους. Χαρακτηριστικά παραδείγματα, που αποτελούν μάλιστα και open source projects είναι το Graphserver και το OpenTripPlanner [1]. Τα projects αυτά αντιμετωπίζουν το πρόβλημα χρησιμοποιώντας ανοιχτά πρότυπα δεδομένων, όπως το GTFS [2] που έχει αναπτυχθεί από τη Google. Με αυτό τον τρόπο αναπτύσσουν τρόπους αναπαράστασης πολλαπλών μέσων και αλγορίθμων δρομολόγησης που μπορούν να εφαρμοστούν αρκετά εύκολα σε όλες τι πόλεις του κόσμου.

Παράλληλα, τα έξυπνα τηλέφωνα έχουν εισβάλει στην καθημερινότητα των περισσότερων ανθρώπων του δυτικού κόσμου. Η δημιουργία του iPhone και η ανάπτυξη του ανοιχτού λειτουργικού Android OS έχουν δώσει ένα νέο ορισμό στη λέξη «smartphone», κάνοντας τα πραγματικά χρήσιμα και εύχρηστα για όλους. Λόγω και της ευκολίας ενσωμάτωσης του Android σε νέες συσκευές (ως open source), η αγορά των κινητών τηλεφώνων έχει μετατραπεί σε αγορά smartphones με του περισσότερους χρήστες κινητής τηλεφωνίας να «τρέχουν» κάποια έκδοση Android στο τηλέφωνό τους.

Καθώς ο κόσμος εξοικειώνεται όλο και περισσότερο με την τεχνολογία των έξυπνων τηλεφώνων και χρησιμοποιεί εφαρμογές που αναπτύσσονται για αυτά, γίνεται εμφανής η ανάγκη της σύνδεσης του κόσμου της βελτιστοποίησης της χρήσης των αστικών συγκοινωνιών με αυτόν των smartphone εφαρμογών.

1.2 Αντικείμενο διπλωματικής

Σκοπός της παρούσης διπλωματικής εργασίας είναι δημιουργία μιας εφαρμογής που εκμεταλλεύεται τις δυνατότητες των κινητών συσκευών με λειτουργικό Android προκειμένου να παρέχει στους χρήστες των κινητών τηλεφώνων χρήσιμες πληροφορίες για τη βελτιστοποίηση των μετακινήσεών τους με τις αστικές συγκοινωνίες στην περιοχή της Αθήνας. Αυτό θα επιτευχθεί αφενός χρησιμοποιώντας την ήδη υπάρχουσα υλοποιημένη server υποδομή του OpenTripPlanner, που έχει αναπτυχθεί στα πλαίσια παλαιότερης εργασίας στο ΙΠΣΥΠ, για τον προσδιορισμό του τρόπου μετάβασης από ένα σημείο της πόλης σε ένα άλλο και αφετέρου με την εγκατάσταση των δεδομένων τοπικά στη συσκευή και τη χρήση offline χαρτών για την παροχή χρήσιμων πληροφοριών.

Κατά την ανάπτυξη και υλοποίηση της συγκεκριμένης εφαρμογής δόθηκε αρκετή έμφαση στην offline χρηστικότητα αυτής, προκειμένου να αποτελέσει χρηστικό εργαλείο για όσο το δυνατόν περισσότερους ανθρώπους. Είναι πολλές οι περιπτώσεις που η χρήση του δικτύου πρέπει να αποφευχθεί. Για παράδειγμα, λόγω του αυξημένου αριθμού χρηστών smartphones, υπάρχουν πολλοί χρήστες που δεν επιθυμούν την εκμετάλλευση του διαδικτύου και για αυτό δεν έχουν συμπεριλάβει τη συσκευή τους σε κάποιο πρόγραμμα δεδομένων. Παράλληλα, υπάρχουν πολλοί τουρίστες που επισκέπτονται την πόλη και δεν επιθυμούν να χρησιμοποιήσουν το πρόγραμμα δεδομένων τους, λόγω των αυξημένων τιμολογίων περιαγωγής. Τέλος δεν είναι λίγα τα σημεία της πόλης όπου η κάλυψη του δικτύου είναι ούτως ή άλλως απαγορευτική για τη χρήση του διαδικτύου μέσα από αυτό. Για αυτές και άλλες περιπτώσεις η εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσης διπλωματικής προσπαθεί να γίνει όσο το δυνατόν λιγότερο εξαρτημένη από το διαδίκτυο κάνοντας, πρακτικά, απαραίτητη την ύπαρξή του μόνο για τον υπολογισμό της βέλτιστης διαδρομής.

1.3 Οργάνωση κειμένου

Το κυρίως κείμενο της διπλωματικής εργασίας χωρίζεται σε 9 κεφάλαια καθένα από τα οποία έχει ρόλο να περιγράψει κάποια διαφορετική πτυχή του συστήματος και συνολικά αναλύουν τους λόγους που αναπτύχθηκε αυτή η εφαρμογή, τη βάση στην οποία στηρίχθηκε, τη διαδικασία υλοποίησής της, αλλά και το τελικό αποτέλεσμα.

Στο **δεύτερο κεφάλαιο** γίνεται μια σύντομη ανάλυση του θεωρητικού υποβάθρου πάνω στο οποίο στηρίχθηκε η ανάπτυξη της εργασίας αυτής καθώς και άλλων συστημάτων που χρησιμοποιούνται από την εφαρμογή.

Στο **τρίτο κεφάλαιο** γίνεται μία αναφορά σε παρόμοιες εφαρμογές που έχουν αναπτυχθεί. Κάποιες από αυτές μάλιστα αποτελούν και τη βάση πάνω στην οποία στηρίχθηκε η παρούσα εργασία.

Στο **τέταρτο κεφάλαιο** αναφέρονται και αναλύονται τα βασικά εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής.

Στο **πέμπτο κεφάλαιο** γίνεται η ανάλυση απαιτήσεων της εφαρμογής που χτίστηκε στα πλαίσια της παρούσης εργασίας. Εκεί παρουσιάζονται αναλυτικότερα οι στόχοι της εφαρμογής.

Στο **έκτο κεφάλαιο** παρουσιάζεται η σχεδίαση του συστήματος σε επίπεδο αρχιτεκτονικής. Γίνεται ο διαχωρισμός του συστήματος στα υπο-συστήματά του και παρουσιάζεται το μοντέλο ER της βάσης δεδομένων που χρησιμοποιείται.

Στο **έβδομο κεφάλαιο** παρουσιάζεται η υλοποίηση του συστήματος σε επίπεδο αρχείων και κώδικα. Αναλύονται οι βασικές κλάσεις πάνω στις οποίες στηρίζονται η κλάσεις της εφαρμογής, αναφέρονται οι κλάσεις της εφαρμογής και παρουσιάζεται το τελικό σχήμα της τοπικής βάσης δεδομένων.

Στο **όγδοο κεφάλαιο** πραγματοποιείται μία πιο αναλυτική παρουσίαση τον ιδιαίτερων πτυχών της ανάπτυξης της εφαρμογής που χρήζουν ιδιαίτερης προσοχής, ενώ ακόμη θα παρουσιαστούν τα εργαλεία ανάπτυξης της εφαρμογής.

Το **ένατο κεφάλαιο** παίζει το ρόλο του εγχειριδίου χρήσης της εφαρμογής που παράχθηκε μέσα από την παρούσα εργασία. Παρουσιάζονται τα βασικά σενάρια χρήσης της εφαρμογής με χρήση επεξηγηματικού κειμένου και αρκετών εικόνων από την ίδια την εφαρμογή.

Στο **δέκατο** κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας και δίνονται ιδέες για περαιτέρω επεκτάσεις που μπορούν να γίνουν στην εφαρμογή που δημιουργήθηκε.

2

Θεωρητικό υπόβαθρο

2.1 Γράφοι

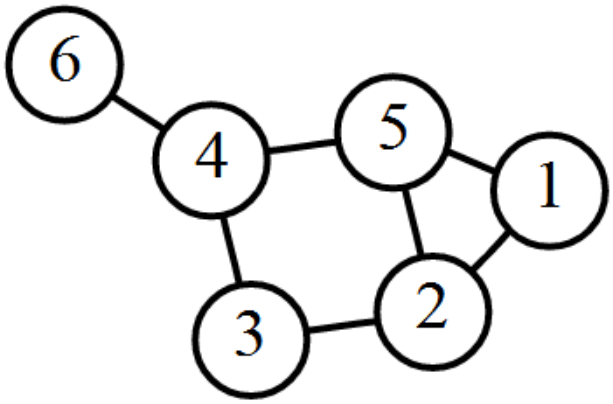
Ένας σχετικά εύκολος και αρκετά χρήσιμος τρόπος να αναπαραστήσει κανείς τα συστήματα μεταφορών είναι με τη χρήση γράφων. Με αυτό τον τρόπο προβλήματα όπως π.χ. η εύρεση βέλτιστων διαδρομών ανάγονται σε προβλήματα επίλυσης γράφων. Σε αυτή τη βάση η αντιμετώπιση των προβλημάτων αφορά ένα πεδίο με πλούσιο θεωρητικό υπόβαθρο μιας και οι γράφοι χρησιμοποιούνται για επίλυση πολλών ειδών προβλημάτων.

Στη γενική περίπτωση ένας γράφος αποτελείται από ένα σύνολο αντικειμένων κάποια από τα οποία συνδέονται μεταξύ τους μέσω συνδέσμων. Τα αντικείμενα αυτά ονομάζονται **κόμβοι** και οι συνδέσεις μεταξύ τους ονομάζονται **ακμές**. Ένας απλός γράφος μπορεί πολύ εύκολα να παρασταθεί σε ένα διάγραμμα όπως αυτό που φαίνεται στην **Εικόνα 2.1.**

Οι ακμές ενός γράφου μπορεί να είναι **κατευθυνόμενες** και να έχουν **βάρη (ή κόστη).** Εφόσον οι ακμές ανάμεσα σε δύο κόμβους ορίζουν μια σχέση, μπορούμε να πούμε ότι μια κατευθυνόμενη ακμή μπορεί να σημαίνει ότι δύο στάσεις λεωφορείου συνδέονται μεταξύ τους προς τη μία κατεύθυνση μόνο. Αντίστοιχα το βάρος της ακμής μπορεί να δείχνει τον απαιτούμενο χρόνο μετακίνησης μεταξύ των στάσεων ή το κόστος της διαδρομής. Ακόμη μπορεί δύο κόμβοι να συνδέονται με παραπάνω από μία ακμές οπότε ο γράφος λέγεται πολλαπλός. Κάτι τέτοιο είναι σύνηθες στην αναπαράσταση των μέσων μεταφοράς σε γράφο.

Σε ένα γράφο μπορεί να υπάρχουν πιο πολύπλοκες σχέσεις από αυτές που περιγράφηκαν. Για παράδειγμα μια διαδρομή μπορεί να αποτελείται από δύο διαφορετικά ΜΜΜ. Η ακμή που

συνδέει τα δύο μέσα θα μπορούσε να έχει ένα μεγάλο βάρος που αντιστοιχεί στην «ποινή» της αλλαγής μέσου. Τα διάφορα βάρη μπορούν να χρησιμοποιηθούν ανάλογα με το πρόβλημα που πρέπει να λυθεί κάθε φορά.



Εικόνα 2.1: Γράφος με 6 κόμβους και 7 ακμές

Παράλληλα ένας γράφος που αφορά τα μέσα μεταφοράς δε χρησιμοποιείται συνήθως μόνος του, αλλά σε συνδυασμό με τον γράφο που αναπαριστά το οδικό δίκτυο της πόλης. Με αυτό τον τρόπο μπορεί να υπολογισθεί καλύτερα η απόσταση μεταξύ δύο κόμβων χρησιμοποιώντας στοιχεία για την κίνηση, το μέγεθος του δρόμου, την ύπαρξη λεωφορειόδρομου κλπ.

2.2 Αλγόριθμοι εύρεσης ελάχιστων μονοπατιών

Ένα σημαντικό πρόβλημα στην θεωρία των γράφων και παράλληλα αυτό που αφορά την παρούσα εργασία, είναι η εύρεση της βέλτιστης διαδρομής από έναν κόμβο του γράφου σε έναν άλλο. Η βέλτιστη αυτή διαδρομή αφορά συνήθως την ελαχιστοποίηση κάποιου κόστους. Αυτό το κόστος μπορεί να είναι ο απαιτούμενος χρόνος, το οικονομικό κόστος κ.α..

Σε θεωρητικό επίπεδο υπάρχει μεγάλο παρελθόν στην ανάλυση γράφων και την επίλυση παρόμοιων προβλημάτων. Στη συνέχεια θα γίνει μα περιγραφή μερικών γνωστών αλγορίθμων που σχετίζονται με αυτό το πρόβλημα.

2.2.1 Ο αλγόριθμος του Dijkstra

Ο αλγόριθμος του Dijkstra, που πήρε την ονομασία του από τον Ολλανδό Edsger Dijkstra που τον δημοσίευσε το 1959, είναι ένας αλγόριθμος που επιλύει το πρόβλημα του ελάχιστου μονοπατιού για ένα γράφο με μη αρνητικά κόστη στις ακμές τους [3]. Ο αλγόριθμος ξεκινά με έναν κόμβο αφετηρία και στο τέλος της εκτέλεσής του έχει εντοπίσει τα μονοπάτια με το ελάχιστο κόστος από αυτόν τον κόμβο προς κάθε άλλο κόμβο του γράφου.

Αν και στη γενική περίπτωση ο αλγόριθμος εντοπίζει τα ελάχιστα μονοπάτια προς όλους του κόμβους, μπορούμε να σταματήσουμε την εκτέλεση του όταν έχει επιλύσει το πρόβλημα για κάποιο συγκεκριμένο κόμβο. Έτσι μπορεί να χρησιμοποιηθεί και για τον εντοπισμό του ελάχιστου μονοπατιού από ένα συγκεκριμένο κόμβο - αφετηρία σε έναν άλλο κόμβο - προορισμό.

2.2.1.1 *Ο αλγόριθμος*

Ονομάζοντας τον κόμβο από τον οποίο ξεκινάμε αφετηρία, θεωρούμε την απόσταση του κόμβου Υ ίση με την απόσταση (κόστος) από τον κόμβο – αφετηρία. Ο αλγόριθμος προσπαθεί να βελτιώσει τις αποστάσεις του κάθε κόμβου εκτελώντας τα παρακάτω βήματα:

1. Απόδοση σε κάθε κόμβο μιας τιμής απόστασης. Αρχικά η τιμή αυτή είναι 0 για τον κόμβο – αφετηρία και άπειρο για όλους τους άλλους κόμβους.
2. Μαρκάρισμα όλων των κόμβων ως κόμβους που δεν έχουν επισκεφθεί. Θέση κόμβου – αφετηρίας ως τρέχοντα κόμβο.
3. Για τον τρέχοντα κόμβο, υπολογισμός της προσωρινής απόστασης από την αφετηρία για όλους τους γειτονικούς του κόμβους. Η προσωρινή απόσταση υπολογίζεται ως το άθροισμα της απόστασης από τον τρέχοντα προς τον κόμβο, με την απόσταση του τρέχοντα από την αφετηρία. Εάν αυτή η προσωρινή απόσταση είναι μικρότερη από την προηγούμενη καταγεγραμμένη, τότε γίνεται αντικατάσταση της καταγεγραμμένης με την τωρινή.
4. Όταν έχουν επισκεφθεί όλοι οι γειτονικοί κόμβοι, αφαιρείται και ο τρέχων κόμβος από τους προς επίσκεψη κόμβους και θεωρείται ότι έχει βρεθεί η τελική και μικρότερη απόστασή του από την αφετηρία.
5. Αν όλοι οι κόμβοι έχουν επισκεφθεί, η εκτέλεση του αλγορίθμου ολοκληρώνεται. Διαφορετικά, τίθεται ως τρέχων ο κόμβος, από τους αυτούς που δεν έχουν επισκεφθεί, με την μικρότερη απόσταση. Η διαδικασία συνεχίζεται από το βήμα 3.

*2.2.2 Ο αλγόριθμος A**

2.2.2.1 *Ευρετικές (Heuristics)*

Πριν προχωρήσουμε στην περιγραφή του αλγορίθμου A* είναι απαραίτητο να περιγραφεί σύντομα η λειτουργία των ευρετικών. Γενικά οι ευρετικές αφορούν τεχνικές που βασίζονται στην εμπειρία για την επίλυση προβλημάτων και κυρίως χρησιμοποιούνται για την επιτάχυνση των διαδικασιών.

Στην θεωρία γράφων μια ευρετική μπορεί να χρησιμοποιηθεί προκειμένου να προβλέψει το επιθυμητό αποτέλεσμα, να επιλέξει τους κλάδους που είναι πιθανότερο να το παράξουν και έτσι να μειώσει τα βήματα που χρειάζονται ώστε να επιτευχθεί αυτό.

2.2.2.2 *Περιγραφή*

Ο αλγόριθμος A* χρησιμοποιεί μια αναζήτηση τύπου «το καλύτερο πρώρα (best-first)» και βρίσκει το μονοπάτι ελάχιστου κόστους από έναν κόμβο – αφετηρία σε έναν κόμβο – προορισμό (που μπορεί να είναι και περισσότεροι από ένας). Η αναζήτηση αυτού του τύπου διασχίσει το γράφο ακλουθώντας την διαδρομή από έναν κόμβο που είναι πιο πολλά υποσχόμενη για να δώσει την επιθυμητή λύση [4].

Η κύρια διαφορά του με τον αλγόριθμο Dijkstra, και άλλους παρόμοιους αλγορίθμους, είναι ότι για την αναζήτηση υιοθετεί μια ευρετική συνάρτηση που είναι το άθροισμα του κόστους μέχρι τον τρέχοντα κόμβο και ευρετικά από τον τρέχοντα μέχρι τον τερματισμό. Έτσι η συνάρτηση αυτή είναι $f(x) = g(x) + h(x)$, όπου:

- $g(x)$ η συνάρτηση που εκφράζει το κόστος από τον κόμβο εκκίνησης μέχρι τον τρέχοντα
- $h(x)$ η συνάρτηση που εκφράζει την ευρετική εκτίμηση της απόστασης από τον τρέχοντα μέχρι τον προορισμό

Αν θέσουμε $h(x) = 0$ ο αλγόριθμος ταυτίζεται με αυτόν του Dijkstra

Η τιμής της $h(x)$ πρέπει να είναι «αποδεκτή», δηλαδή να είναι ίση ή μικρότερη από την πραγματική απόσταση από το στόχο, δε θα πρέπει δηλαδή να γίνεται υπερεκτίμησή της. Έτσι, για μια εφαρμογή όπως η δρομολόγηση, η $h(x)$ μπορεί να αντιπροσωπεύει την ευκλείδεια απόσταση από τον στόχο, μιας και αυτή είναι η μικρότερη δυνατή απόσταση μεταξύ δύο σημείων ή δύο κόμβων.

Αν η ευρετική h ικανοποιεί την επιπλέον προϋπόθεση $h(x) \leq d(x, y) + h(y)$ για κάθε ακμή x, y του γράφου (όπου η d εκφράζει το μήκος της ακμής αυτής), τότε η h ονομάζεται μονότονη (ή συνεπής). Σε αυτή την περίπτωση ο A* μπορεί να υλοποιηθεί πιο αποτελεσματικά. Αυτό χονδρικά σημαίνει ότι κανένας κόμβος δε θα χρειαστεί να επεξεργαστεί παραπάνω από μία φορά. Σε αυτή την περίπτωση ο A* είναι αντίστοιχος με έναν αλγόριθμο Dijkstra με το μειωμένο κόστος $d'(x, y) := d(x, y) - h(x) + h(y)$

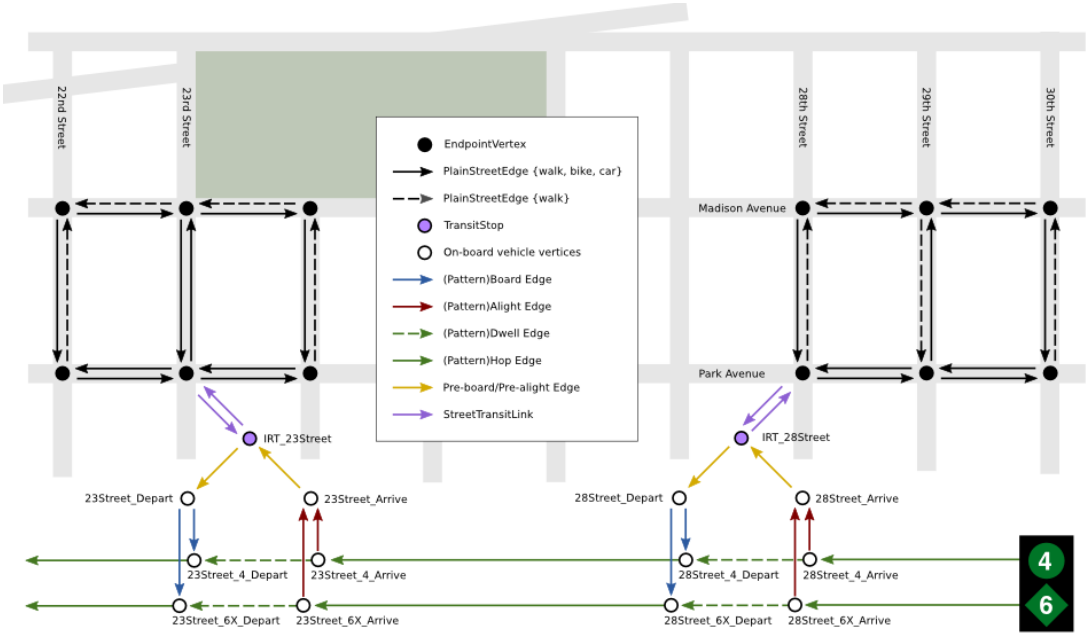
2.3 Γράφοι αναπαράστασης πολλαπλών μέσων

Έχοντας εξετάσει τα βασικά που αφορούν του γράφους και τον εντοπισμό ελάχιστων μονοπατιών σε αυτούς, μπορούμε να δούμε πιο αναλυτικά, χρησιμοποιώντας ένα πραγματικό παράδειγμα, την αναπαράσταση πολλαπλών μέσων σε γράφους. Για το σκοπό αυτό θα χρησιμοποιηθεί το OpenTripPlanner, πάνω στο οποίο θα στηριχθεί και ένα σημαντικό κομμάτι της παρούσας εργασίας.

Το οδικό και συγκοινωνιακό δίκτυο αναπαρίσταται στο OpenTripPlanner χρησιμοποιώντας κατευθυνόμενους γράφους. Ο σκοπός αυτής της παραγράφου είναι να εξηγήσει πως

δομούνται οι κόμβοι και οι κορυφές προκειμένου να βοηθήσουν τον εντοπισμό βέλτιστων μονοπατιών μέσα σε ένα πολύπλοκο συγκοινωνιακό σύστημα.

Η **Εικόνα 2.2** δείχνει ένα μικρό μέρος ενός OTP γράφου για τη Νέα Υόρκη και περιλαμβάνει πληροφορία και για το οδικό και για το συγκοινωνιακό δίκτυο. Σε αυτό το γράφο χρησιμοποιείται μία απλοποιημένη αναπαράσταση του οδικού δικτύου, όπου οι κόμβοι αναπαριστούν τις διασταυρώσεις και οι ακμές τα τμήματα του δρόμου μεταξύ διασταυρώσεων. Αυτό το μοντέλο έχει αντικατασταθεί από νεότερο, αλλά παρουσιάζεται εδώ μιας και το καινούργιο μοντέλο προκύπτει από αυτό. Οι ακμές στο OTP είναι κατευθυνόμενες. Έτσι, ένα τμήμα δρόμου συνήθως αποτελείται από δύο ακμές, μία για κάθε κατεύθυνση. Συγκεκριμένοι τύποι μετακίνησης είναι δυνατό να επιτρέπονται σε μία κατεύθυνση, αλλά όχι και στην άλλη (π.χ. μονόδρομοι, ποδηλατοδρόμοι).



Εικόνα 2.2: Γράφος αναπαράστασης ενός μέρους του οδικού και συγκοινωνιακού δικτύου στο Μανχάταν [5]

Κάθε συγκοινωνιακή στάση έχει τουλάχιστον ένα κόμβο που αντιπροσωπεύει το να είναι κανείς «μέσα» ή «στη» στάση. Θα μπορούσαν να υπάρχουν περισσότεροι κόμβοι για μία στάση αν διαθέτει πολλές αποβάθρες. Οι κόμβοι αυτοί είναι συνδεδεμένοι με ακμές μικρού βάρους στο οδικό δίκτυο. Στην Εικόνα 2.2, χρησιμοποιούνται δύο διαφορετικά μοτίβα ταξιδιού για να αντιπροσωπεύσουν δύο διαφορετικές υπηρεσίες μετρό που τυχαίνει να διέρχονται από τις ίδιες στάσεις, αλλά χωρίζουν σε άλλο σημείο της πόλης.

Από τα παραπάνω γίνεται φανερό ότι η αναπαράσταση πολλαπλών μέσων με τη χρήση γράφων και η διαχείριση προβλημάτων όπως η δρομολόγηση μέσα σε αυτά είναι ένα ανοιχτό πρόβλημα με πλούσιο ερευνητικό ενδιαφέρον. Για το λόγο αυτό έχουν αναπτυχθεί αρκετά open source projects που έχουν σκοπό, ερευνώντας και εντοπίζοντας καινούργιες λύσεις, να

διευκολύνουν τη βελτιστοποίηση της χρήσης και της δρομολόγησης των αστικών συγκοινωνιών σε μια αστική περιοχή. Ένα τέτοιο project είναι και το OpenTripPlanner, που αναφέρθηκε παραπάνω, το οποίο θα χρησιμοποιηθεί εκτενώς στην παρούσα εργασία. Άλλωστε, η εφαρμογή που θα παραχθεί από αυτή τη διπλωματική εργασία θα αποτελέσει, μεταξύ άλλων, και έναν client για την οπτικοποίηση των αποτελεσμάτων δρομολόγησης των αλγορίθμων του OTP.

3

Σχετικές εφαρμογές

Παραπάνω είδαμε το σκοπό της εργασίας αυτής και το πώς χρησιμοποιώντας νέες τεχνολογίες και σύγχρονους αλγορίθμους μπορούμε να βελτιώσουμε τις αστικές συγκοινωνίες. Η συγκεκριμένη εργασία όμως δεν είναι η πρώτη που καταπιάνεται με αυτό το θέμα. Υπάρχουν ήδη πολλές εφαρμογές γύρω από αυτό το χώρο, καθεμία από τις οποίες επικεντρώνει σε κάποιο διαφορετικό σημείο, αλλά όλες έχουν ως κοινό στόχο τη βελτιστοποίηση της χρήσης των αστικών συγκοινωνιών με χρήση των σύγχρονων και προσιτών, στον απλό χρήστη, τεχνολογιών.

Προσπαθώντας να διαχωρίσουμε σε κάποιες βασικές κατηγορίες τις εφαρμογές που ασχολούνται με τη βελτιστοποίηση της χρήσης αστικών συγκοινωνιών, μπορούμε να θεωρήσουμε τις εξής δύο, που συχνά απαντώνται και μαζί:

- **Δρομολόγηση με χρήση αστικών συγκοινωνιών**
- **Παρουσίαση προγράμματος αστικών συγκοινωνιών**

Η πρώτη κατηγορία αφορά τις εφαρμογές που αναλαμβάνουν την αναζήτηση της βέλτιστης διαδρομής για τη μετακίνηση από σημείο σε σημείο χρησιμοποιώντας τις αστικές συγκοινωνίες. Η δεύτερη ασχολείται αποκλειστικά με την παρουσίαση των διαδρομών, των στάσεων, των δρομολογίων και του προγράμματος των αστικών συγκοινωνιών παρέχοντας πιθανόν και αναζήτηση πάνω σε αυτά τα δεδομένα.

Οι σημαντικότερες από αυτές τις εφαρμογές, που έχουν πολλές ομοιότητες και διαφορές με την εφαρμογή που παρουσιάζεται εδώ, παρουσιάζονται στις επόμενες παραγράφους. Κάποιες από αυτές αποτέλεσαν έμπνευση και άλλες τη βάση για αυτήν εδώ την εργασία.

3.1 Δρομολόγηση με χρήση αστικών συγκοινωνιών

Ένα ακόμη σημαντικό κομμάτι της παρούσας εφαρμογής είναι η αναζήτηση βέλτιστης διαδρομής με τη χρήστη των αστικών συγκοινωνιών. Οι εφαρμογές που παρουσιάζονται σε αυτή την ενότητα εντοπίζουν τη λειτουργικότητά τους σε αυτή την κατεύθυνση και έπαιξαν σημαντικό ρόλο στη σχεδίαση της δικής μας εφαρμογής.

3.1.1 Google Transit

Το Google Transit είναι μία υπηρεσία δρομολόγησης με χρήση των αστικών συγκοινωνιών που προσφέρεται από τη Google σε συνδυασμό με τις υπηρεσίες χαρτών Google Maps. Ξεκίνησε ως ανεξάρτητο πρόγραμμα το Δεκέμβριο του 2005 υποστηρίζοντας αρχικά την περιοχή του Portland, Oregon και ενσωματώθηκε στις υπόλοιπες υπηρεσίες των χαρτών της Google τον Οκτώβριο του 2007. Έκτοτε αυξάνεται συνεχώς ο αριθμός των υποστηριζόμενων πόλεων σε όλο τον κόσμο, ενώ πρόσφατα στη λίστα προστέθηκε και η ευρύτερη περιοχή της Αθήνας [6].

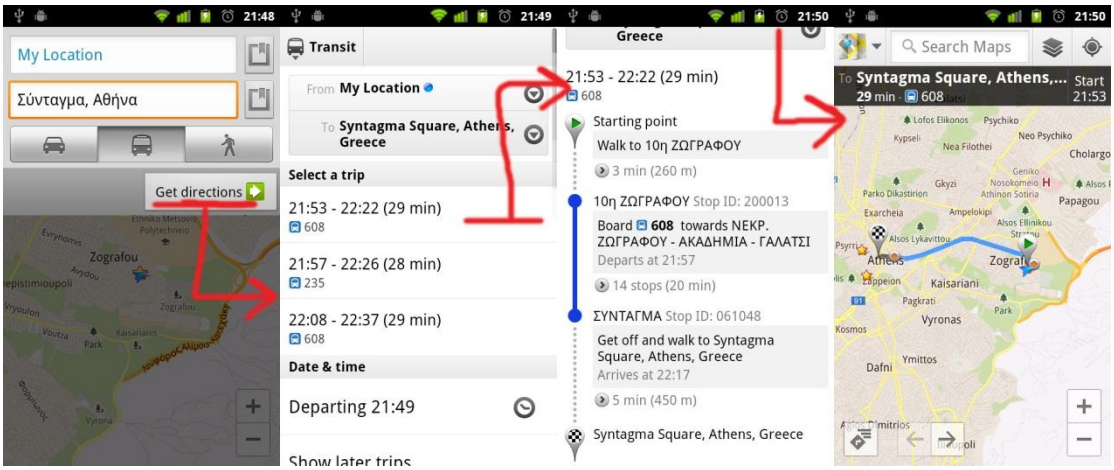
Οι υπηρεσία χαρτών της Google είναι διαθέσιμη όχι μόνο στο web, αλλά και στα περισσότερα smartphones της αγοράς, είτε ως εφαρμογή, είτε μέσω του browser (φυλλομετρητή) του κινητού τηλεφώνου. Έτσι, η υπηρεσία Transit γίνεται διαθέσιμη και στους χρήστες έξυπνων κινητών συσκευών. Οι χρήστες Android συσκευών βρίσκουν την εφαρμογή αυτή προ-εγκατεστημένη στο κινητό τους.

Το Google Transit βασίστηκε στο πρότυπο GTFS για την ανάπτυξή του. Το GTFS, το οποίο θα περιγραφεί εκτενέστερα σε επόμενο κεφάλαιο, προτείνει ένα πρότυπο για την συγκέντρωση και δημοσίευση των δεδομένων των εταιριών αστικών συγκοινωνιών. Με αυτή τη βάση λοιπόν έχει διευκολυνθεί η προσθήκη καινούργιων εταιριών και κατ' επέκταση πόλεων στην υπηρεσία. Κάθε εταιρία που συγκεντρώνει και δημοσιοποιεί τα δεδομένα της σε αυτό το πρότυπο, επιτρέπει στη Google να χρησιμοποιήσει τους, ήδη υλοποιημένους, αλγόριθμούς της και να προσφέρει υπηρεσίες δρομολόγησης στους χρήστες της.

Ο χρήστης του Google Transit έχει τη δυνατότητα να πληκτρολογήσει διεύθυνση ή σημείο ενδιαφέροντος ως σημείο αφετηρίας/προορισμού ή ακόμα και να χρησιμοποιήσει την ακριβή τοποθεσία του που θα βρεθεί μέσω δικτύου ή μέσω GPS από το σύστημα. Στη συνέχεια μπορεί να επιλέξει την επιθυμητή ώρα αναχώρησης ή άφιξης και επιλέγοντας του πλήκτρο «δρομολόγηση» αφήνει το σύστημα να υπολογίσει τη βέλτιστη διαδρομή. Αφού γίνει ο υπολογισμός και, μέσω δικτύου, επιστρέψουν τα αποτελέσματα, ο χρήστης βλέπει μια λίστα με τις διαθέσιμες επιλογές. Επιλέγοντας μία από αυτές μπορεί να δει την αναλυτική περιγραφή της με κείμενο, αλλά και όλη τη λεπτομέρεια της διαδρομής στο χάρτη. Οι οδηγίες φυσικά περιλαμβάνουν, εκτός από τα μέσα μαζικής μεταφοράς, και οδηγίες περπατήματος

από σημείο σε σημείο. Όλοι οι υπολογισμοί γίνονται σε απομακρυσμένο εξυπηρετητή και η συσκευή του χρήστη λαμβάνει τα αποτελέσματα μέσω δικτύου [7].

Τα 4 βασικά σημεία της προηγούμενης περιγραφής φαίνονται στην παρακάτω εικόνα:



Εικόνα 3.1: Google Transit, βασικό παράδειγμα χρήσης σε Android

3.1.2 OpenTripPlanner

Το OpenTripPlanner (OTP) είναι ένα πρόγραμμα ανοιχτού κώδικα που βελτιστοποιεί τη χρήση των αστικών συγκοινωνιών με χρήση χαρτών και αλγορίθμων προγραμματισμού ταξιδιού.

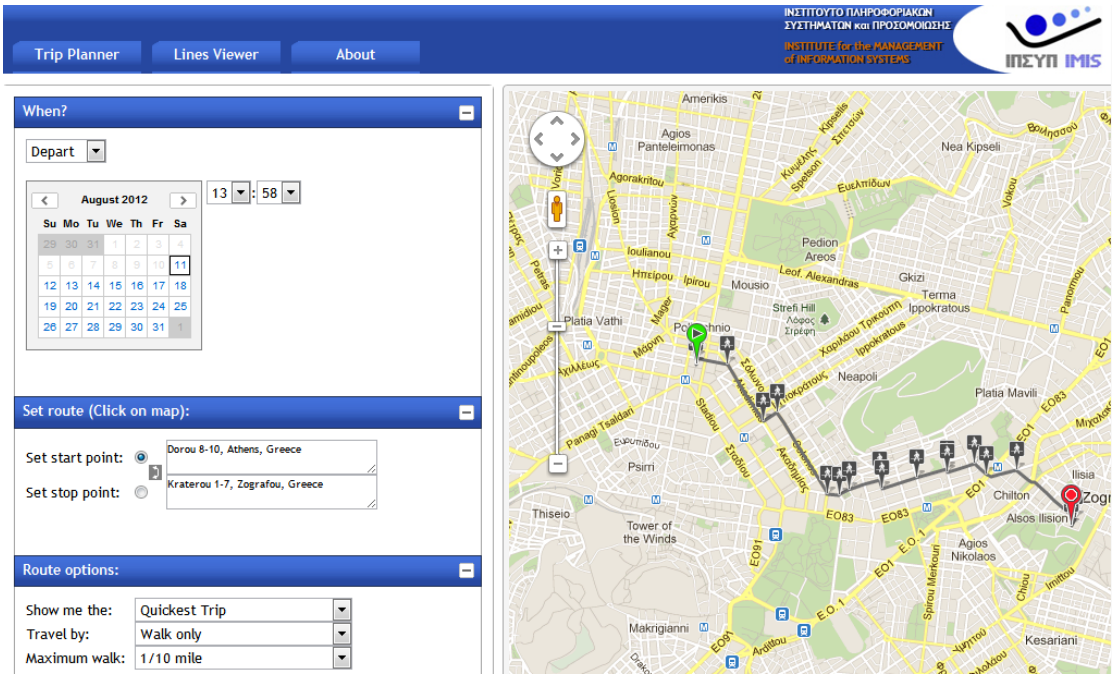
Ο χρήστης του OTP έχει τη δυνατότητα να προγραμματίσει το ταξίδι του δίνοντας στο σύστημα μια σειρά από επιλογές. Αρχικά μπορεί να επιλέξει ανάμεσα από του διαθέσιμους τρόπους ταξιδιού (λεωφορείο, μετρό, περπάτημα κλπ), ενώ έχει τη δυνατότητα να ορίσει το πόσα μέτρα προτίθεται να περπατήσει ή το αν έχει κάποια κινητικά προβλήματα. Εκτός από αυτές τις βασικές πληροφορίες το σύστημα μπορεί να λάβει υπόψη του ακόμα πιο συγκεκριμένες επιλογές. Ο χρήστης μπορεί να επιλέξει για τον τρόπο δρομολόγησης ανάμεσα από τρεις λειτουργίες: γρήγορο ταξίδι, ασφαλές ταξίδι ή λιγότερες μετεπιβιβάσεις.

Η καρδιά του OTP είναι ένας αλγόριθμος δρομολόγησης που χρησιμοποιεί ιεραρχίες συρρίκνωσης για το οδικό δίκτυο και δρομολόγηση σε ακριβή χρόνο για το συγκοινωνιακό δίκτυο. Τα δεδομένα που χρησιμοποιεί μπορεί να προέρχονται από το πρότυπο GTFS, ενώ μπορούν να προστεθούν και άλλα δεδομένα που θα βοηθήσουν την ακρίβεια της δρομολόγησης (π.χ. γράφοι διαδρομών για ποδήλατο).

Η βασική διεπαφή για το χρήστη του OTP είναι μέσα από έναν web browser (φυλλομετρητή) όπου μια ιστοσελίδα παρουσιάζει στο χρήστη το χάρτη της περιοχής δίνοντας του τη δυνατότητα να επιλέξει σημείο έναρξης και προορισμού καθώς και τις συγκεκριμένες επιλογές δρομολόγησης που αναφέρθηκαν πιο πάνω. Πέρα από τη βασική διεπαφή το OTP προσφέρει και ένα Application Programming Interface (API, Διεπαφή προγραμματισμού

εφαρμογών) το οποίο δίνει τη δυνατότητα επικοινωνίας με το OTP και τη χρήση του αλγόριθμου δρομολόγησης σε άλλες, εξωτερικές, εφαρμογές. Αυτό θα αποτελέσει και τον τρόπο λειτουργίας της υπηρεσίας δρομολόγησης της παρούσας εφαρμογής [1].

Η εφαρμογή μας θα χρησιμοποιήσει το OpenTripPlanner σε μία εφαρμογή του για την περιοχή της Αθήνας η οποία έχει κατασκευαστεί από το ΙΠΣΥΠ (Ινστιτούτο Πληροφοριακών Συστημάτων) και λειτουργεί και ως web εφαρμογή χρησιμοποιώντας Google Maps για την παρουσίαση όπως φαίνεται στην Εικόνα 3.2 [8].



Εικόνα 3.2: Web εφαρμογή που χρησιμοποιεί το OTP (ΙΠΣΥΠ)

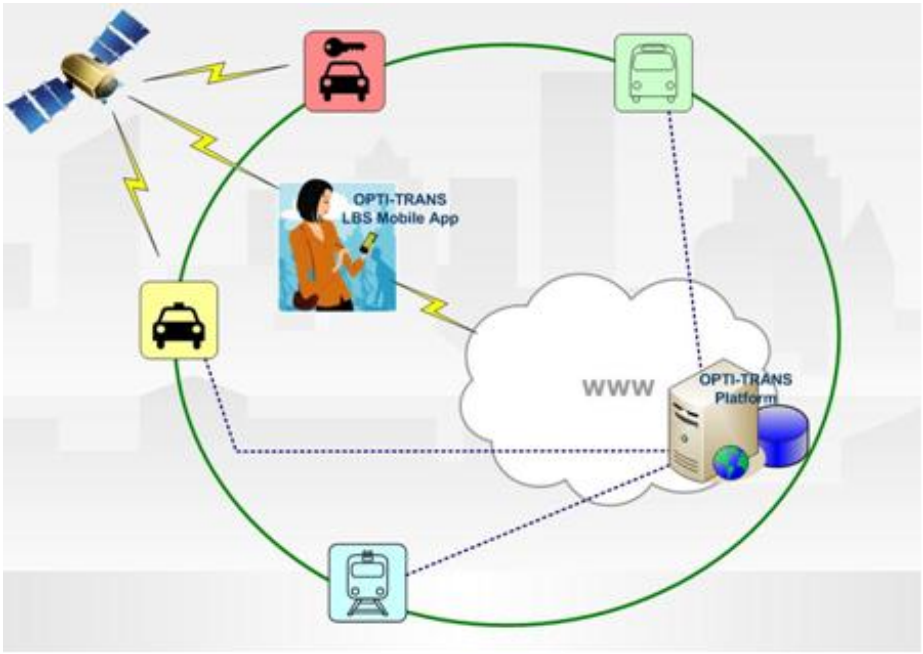
3.1.3 OPTI-TRANS

«Το OPTI-TRANS είναι ένα έργο που έχει σκοπό να δημιουργήσει μία πλατφόρμα GNSS (Global Navigation Satellite System) η οποία θα παρέχει στους κατοίκους της πόλης και τους ταξιδιώτες τη δυνατότητα να προγραμματίσουν το ταξίδι τους με αποτελεσματικό τρόπο αξιοποιώντας τα δημόσια και ιδιωτικά μέσα μεταφοράς. Θα εμφανίσει το βέλτιστο συνδυασμό για τις διαδρομές τους, με βάση την τοποθεσία τους.»

Αυτή είναι μια σύντομη περιγραφή του σκοπού του OPTI-TRANS όπως δίνεται από την επίσημη ιστοσελίδα του [9]. Το OPTI-TRANS είναι ένα έργο συγχρηματοδοτούμενο από την Ευρωπαϊκή Επιτροπή και υλοποιείται από μια κοινοπραξία που αποτελείται από τους Ε.Κ.Ε.Φ.Ε. Δημόκριτος, Telefonika I+D, AVEGO και EMT με συντονιστή το ερευνητικό κέντρο «Δημόκριτος».

Το σημαντικότερο στοιχείο του έργου είναι ότι στοχεύει στη δημιουργία τόσο της πλατφόρμας υπολογισμού βέλτιστων διαδρομών όσο και της εφαρμογής για κινητές

συσκευές, καλύπτοντας το μεγαλύτερο δυνατό μέρος χρηστών. Αυτό το πετυχαίνει δημιουργώντας ξεχωριστή εφαρμογή για συσκευές με λειτουργικό iOS και Android, αλλά και εφαρμογή Java που μπορεί να «τρέξει» στα περισσότερα τηλέφωνα της αγοράς. Μία σύντομη περιγραφή του έργου φαίνεται στην Εικόνα 3.3.



Εικόνα 3.3: Γενική εικόνα της πλατφόρμας του OptiTrans

Το βασικό σενάριο χρήσης της εφαρμογής μοιάζει πολύ με αυτό του Google Transit. Ο χρήστης επιλέγει διεύθυνση αφετηρίας/προορισμού και χρόνο και εκτελεί την αναζήτηση. Λαμβάνει τα αποτελέσματα, επιλέγει ένα από αυτά και βλέπει τη διαδρομή στο χάρτη, αλλά και τις λεπτομέρειες της σε μορφή κειμένου. Η βασική διαφορά είναι ότι στο πρώτο βήμα ο χρήστης έχει τη δυνατότητα να επιλέξει τα μέσα μεταφοράς με τα οποία προτιμάει να ταξιδέψει ή αυτά που θέλει να αποφύγει, ενώ έχει επιλογή να χρησιμοποιήσει και TAXI.

3.2 Παρουσίαση προγράμματος αστικών συγκοινωνιών

Αυτή η κατηγορία αφορά τις εφαρμογές που αφορούν κυρίως την παροχή πληροφοριών σχετικά με τις αστικές συγκοινωνίες και δεν δίνουν τη δυνατότητα στο χρήστη να αναζητήσει κάποια βέλτιστη διαδρομή. Αυτές οι εφαρμογές περιλαμβάνουν λειτουργίες όπως παρουσίαση προγράμματος δρομολογίων και οπτικοποίηση μιας διαδρομής πάνω σε χάρτη. Οι λειτουργίες αυτές αποτελούν ένα σημαντικό κομμάτι της παρούσας διπλωματικής και ο τρόπος υλοποίησης τους στις παρακάτω εφαρμογές αποτέλεσε σημείο έναρξης και ενέπνευσε την υλοποίησή μας.

3.2.1 Transportation in Athens

Το Transportation in Athens είναι μία από τις λίγες android εφαρμογές που αφορούν τα μέσα μαζικής μεταφοράς της περιοχής της Αθήνας. Αναλαμβάνει να παρουσιάσει τι διαδρομές των αστικών συγκοινωνιών και όχι να οδηγήσει το χρήστη χρησιμοποιώντας κάποιο αλγόριθμο δρομολόγησης.

Το σημαντικό στοιχείο της εφαρμογής είναι ότι, χρησιμοποιώντας τοπική βάση δεδομένων, μπορεί να δώσει πληροφορίες χωρίς τη χρήση διαδικτύου. Με αυτό τον τρόπο δίνει στο χρήστη τη δυνατότητα ενημέρωσης για βασικά στοιχεία μια διαδρομής σε σημεία όπου η κάλυψη του δικτύου είναι ελλιπής, ενώ μπορεί να χρησιμοποιηθεί και από χρήστες που δεν διαθέτουν υπηρεσίες δεδομένων στο κινητό τους τηλέφωνο.

Η κυρίως πληροφορία-λειτουργικότητα που προσφέρει η εφαρμογή είναι:

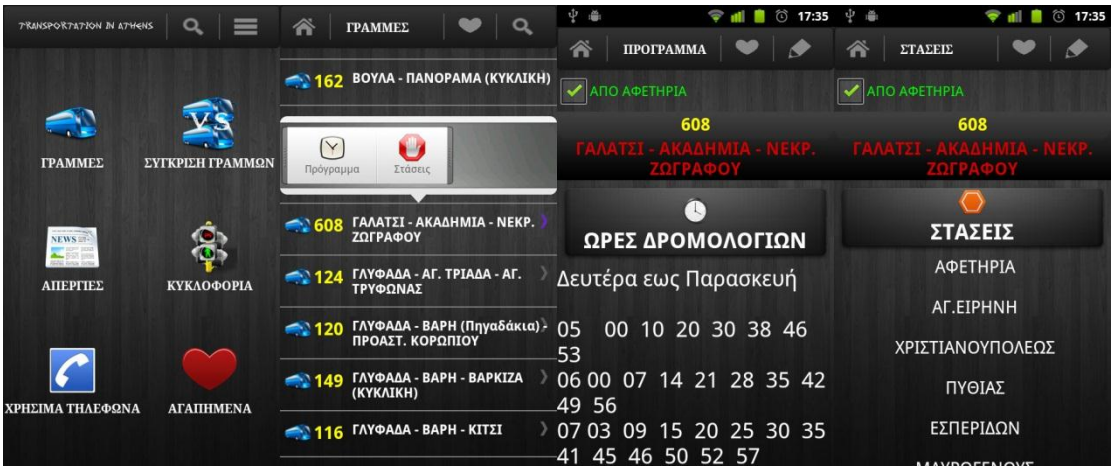
- Ώρες εκκίνησης δρομολογίου από αφετηρία/στάση
- Στάσεις διαδρομής
- Σύγκριση 2 διαδρομών
- Αναζήτηση διαδρομής
- Αποθήκευση αγαπημένων διαδρομών

Εκτός από αυτά περιλαμβάνει χρήσιμα τηλέφωνα και πληροφορίες για απεργίες των μέσων μεταφοράς και την κίνηση στους δρόμους. Οι δύο αυτές τελευταίες υπηρεσίες είναι και οι μόνες που απαιτούν σύνδεση στο διαδίκτυο.

Στην πρώτη οθόνη της εφαρμογής ο χρήστης έχει τη δυνατότητα να επιλέξει ανάμεσα από τις κατηγορίες «Γραμμές», «Σύγκριση Γραμμών», «Απεργίες», «Κυκλοφορία», «Χρήσιμα Τηλέφωνα» και «Αγαπημένα». Αν επιλέξει κανείς το εικονίδιο «Γραμμές» βρίσκεται σε μια οθόνη που περιλαμβάνει σε λίστα όλες τις διαθέσιμες γραμμές για όλα τα μέσα του ΟΑΣΑ στην Αθήνα. Πατώντας σε μία από τις διαδρομές αυτές ο χρήστης έχει δύο επιλογές: «Πρόγραμμα» και «Στάσεις». Επιλέγοντας το πρώτο παρουσιάζονται τα δρομολόγια τις διαδρομής, που αντιστοιχούν στις ώρες εκκίνησης από την αφετηρία ή το τέρμα της γραμμής. Αν από την άλλη επιλέξουμε «Στάσεις» τότε βρισκόμαστε σε μία λίστα από στάσεις που απεικονίζουν την πορεία στης διαδρομής. Κι εδώ υπάρχει η επιλογή της παρουσίασης από την αφετηρία ή από το τέρμα.

Οι βασικές αυτές λειτουργίες φαίνονται στην Εικόνα 3.4. Αν στη πρώτη οθόνη επιλέξουμε «Σύγκριση Γραμμών» έχουμε τη δυνατότητα να συγκρίνουμε δύο γραμμές ως προς όλες τι πληροφορίες του (δύο λίστες η μία δίπλα στην άλλη). Αν επιλέξουμε «Απεργίες» θα δούμε σε μία λίστα τις πιο πρόσφατες ανακοινώσεις για απεργίες, ενώ αντίστοιχα μπορούμε να ενημερωθούμε για την κυκλοφορία ή να δούμε και να καλέσουμε χρήσιμα τηλέφωνα για την περιοχή της Αθήνας. Τέλος επιλέγοντας «Αγαπημένα» ο στην οθόνη εμφανίζεται μία λίστα

ίδια με αυτή της επιλογής «Γραμμές» με τη διαφορά ότι περιλαμβάνει μόνο τις γραμμές που ο χρήστης έχει αποθηκεύσει στα «Αγαπημένα» [10].



Εικόνα 3.4: Transportation in Athens, βασικές οθόνες

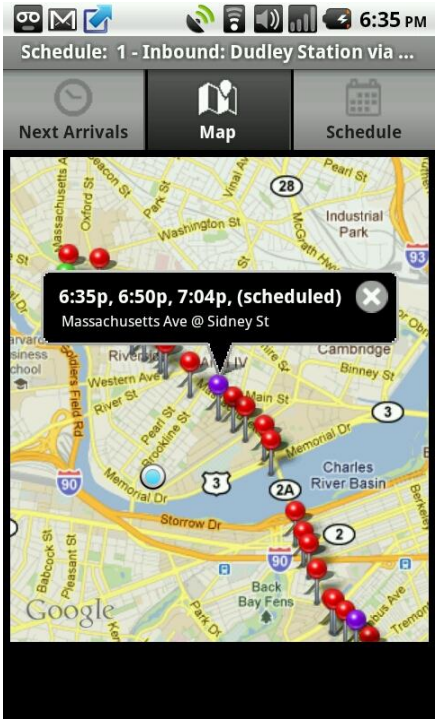
3.2.2 OpenMBTA

Το OpenMBTA είναι μια δωρεάν, ανοιχτού κώδικα πλατφόρμα που έχει σκοπό την παροχή χρονοδιαγραμμάτων και δεδομένων σε πραγματικό χρόνο (real-time data) στους επιβάτες των αστικών συγκοινωνιών της Βοστόνης. Παρέχει τις πληροφορίες αυτές μέσω ενός απλού browser κινητού τηλεφώνου, ενώ διαθέτει και πιο «πλούσιες» εφαρμογές για συσκευές iPhone, iPad και Android.

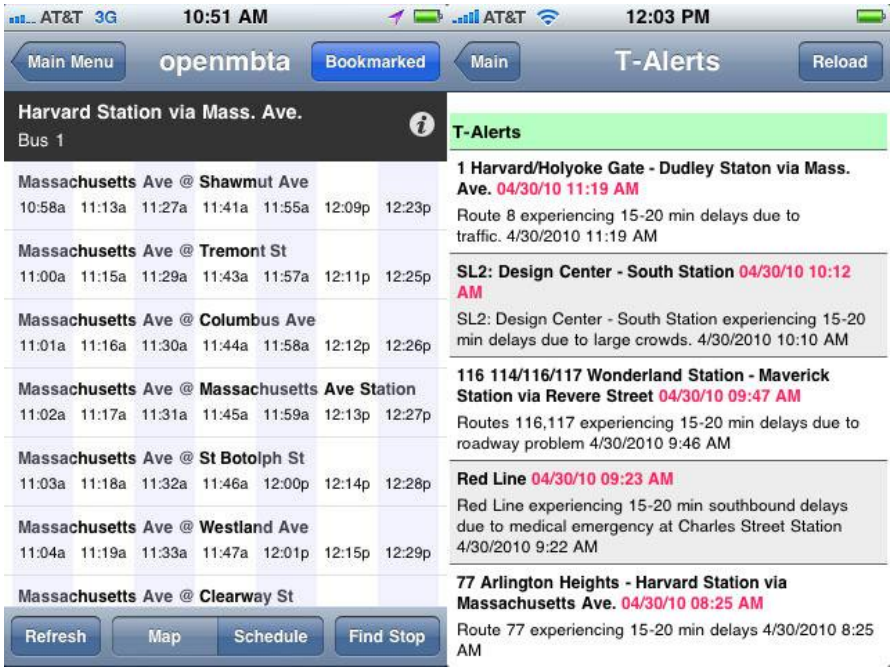
Η εφαρμογές αυτές βασίζονται πλήρως στο διαδίκτυο για τη λειτουργία της. Αυτό τις κάνει πολύ ευέλικτες σε αλλαγές των χρονοδιαγραμμάτων, ενώ τους δίνει ταυτόχρονα και τη δυνατότητα για παροχή real-time δεδομένων. Αυτό βέβαια καθιστά απαραίτητη τη σύνδεση στο διαδίκτυο μέσω του κινητού στο χρήστη, κάτι που φυσικά είναι πιο σύνηθες στις ΗΠΑ από ότι στην Ελλάδα αν και το καθεστώς αυτό αλλάζει και στη χώρα μας με τη διάδοση των έξυπνων τηλεφώνων.

Εκτός από τις εφαρμογές για κινητές συσκευές, το OpenMBTA περιλαμβάνει και μία εφαρμογή εξυπηρετητή (server). Αυτό καθιστά πολύ πιο εύκολη τη διαχείριση μεγάλου όγκου δεδομένων και αποτελεί το μέσο με το οποίο επικοινωνούν οι εφαρμογές στα τερματικά για την αναζήτηση και την παρουσίαση της πληροφορίας.

Μερικές από τις λειτουργίες τις εφαρμογής φαίνονται στις παρακάτω εικόνες:



Εικόνα 3.5: OpenMBTA σε Android



Εικόνα 3.6: OpenMBTA σε iPhone

Στις παραπάνω εικόνες παρουσιάζονται κάποιες κύριες οθόνες της εφαρμογής σε Android και iPhone. Παρόμοια είναι και η εικόνα στο iPad όπου, λόγω περισσότερου χώρου, παρουσιάζονται οι διαδρομές και οι πληροφορίες για αυτές στην ίδια οθόνη.

Κάθε διαδρομή μπορεί να παρουσιαστεί στο χάρτη, ενώ επιλέγοντας κάποια από τις στάσεις ο χρήστης ενημερώνεται για τις ώρες στις οποίες διέρχεται το μέσο από αυτή. Το σύστημα παρέχει, επίσης, προειδοποιήσεις για έκτακτες αλλαγές δρομολογίων, ενώ παρουσιάζει

ακόμη και Tweets με το hash tag #mbta. Αυτή η τελευταία λειτουργία παρουσιάζει ενδιαφέρον, αφού δίνει τη δυνατότητα στους ίδιους του επιβάτες να ενημερώσουν ο ένας τον άλλο για τυχών έκτακτες αλλαγές λόγω κακοκαιρίας, καταστροφών ή ατυχημάτων [11].

3.3 Η συμβολή του aTransit

Είδαμε στην παραπάνω ανάλυση ότι υπάρχουν πολύ σημαντικές εφαρμογές τόσο στον τομέα παρουσίασης δεδομένων αστικών συγκοινωνιών, όσο και στον πιο εξειδικευμένο και ερευνητικά ενδιαφέρον, τομέα της δρομολόγησης μέσα στην πόλη. Η καινούργια εφαρμογή aTransit, που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, καταπιάνεται και με του δύο αυτούς τομείς δίνοντας μεγάλη έμφαση στις offline υπηρεσίες.

Αυτή ακριβώς η έμφαση που θα δοθεί στο offline κομμάτι της εφαρμογής είναι που τη διαφοροποιεί από τις εφαρμογές που παρουσιάστηκαν σε αυτό εδώ το κεφάλαιο και άλλες παρόμοιες. Ο χρήστης θα έχει τη δυνατότητα να ενημερωθεί για το πρόγραμμα των αστικών συγκοινωνιών, να αναζητήσει διαδρομές και στάσεις χωρίς να συνδεθεί στο διαδίκτυο. Θα ενσωματωθούν offline χάρτες που θα δίνουν ακόμη περισσότερη πληροφορία στο χρήστη. Τέλος, αν και η δρομολόγηση θα απαιτεί σύνδεση στο internet θα δίνεται η δυνατότητα αποθήκευσης των αποτελεσμάτων για μετέπειτα offline χρήση.

Σκοπός λοιπόν είναι να δημιουργηθεί μια εφαρμογή που θα δίνει όλες τις δυνατότητες που περιγράφηκαν σε αυτό το κεφάλαιο, μειώνοντας στο ελάχιστο την απαιτούμενη συνδεσιμότητα στο διαδίκτυο. Με αυτό τον τρόπο θα προκύψει μία καλή εμπειρία χρήσης αφού πλέον η απόκριση της εφαρμογής επηρεάζεται μόνο από τη συσκευή του χρήστη και όχι από την ποιότητα του δικτύου και το αν βρίσκεται σε εξωτερικό χώρο ή όχι. Παράλληλα, θα αποφευχθούν οι ανεπιθύμητες χρεώσεις και θα γίνει ευκολότερη η χρήση των συγκοινωνιών και για τους επισκέπτες της χώρας μας.

4

Δεδομένα, εργαλεία και τεχνολογίες

Στο παρόν κεφάλαιο παρουσιάζονται και αναλύονται οι διάφορες τεχνολογίες που θα χρησιμοποιηθούν στην παρούσα εργασία. Αρχικά παρουσιάζεται η μορφή των δεδομένων στα οποία θα βασιστούμε, στη συνέχεια οι υποδομές δρομολόγησης και χαρτών που θα χρησιμοποιηθούν και τέλος θα γίνει μία εισαγωγή στο λειτουργικό σύστημα Android και τις βασικές αρχές λειτουργίας του. Ότι παρουσιάζεται σε αυτό το κεφάλαιο είναι δουλειά τρίτων η οποία θα χρησιμοποιηθεί στην σχεδίαση και την υλοποίηση της εφαρμογής, καθώς επίσης και έννοιες που θεωρήθηκαν απαραίτητες για την κατανόηση των ακόλουθων κεφαλαίων.

4.1 General Transit Feed Specification

Το General Transit Feed Specification (GTFS) ορίζει μια κοινή μορφή οργάνωσης δεδομένων για δρομολόγια αστικών συγκοινωνιών και τη γεωγραφική τους πληροφορία. Επιτρέπει στις εταιρίες αστικών συγκοινωνιών να δημοσιοποιήσουν τα δεδομένα των δρομολόγιών της, ενώ ταυτόχρονα βοηθάει τους προγραμματιστές να αναπτύξουν λογισμικό που να καταναλώνει τέτοιου είδους δεδομένα. Η ιδέα για το GTFS ξεκίνησε από την Portland TriMet [12], που είναι εταιρία αστικών συγκοινωνιών στο Portland, Oregon των ΗΠΑ. Αναπτύχθηκε σε συνεργασία με τη Google η οποία και το χρησιμοποιεί για την υπηρεσία δρομολόγησης στους χάρτες της (Google Maps).

4.1.1 Προδιαγραφές

Μια συλλογή δεδομένων σε πρότυπο GTFS αποτελείται από μια σειρά από αρχεία κειμένου (txt) συγκεντρωμένα σε ένα συμπιεσμένο αρχείο (zip). Κάθε αρχείο κειμένου μοντελοποιεί

μια διαφορετική πτυχή του συνόλου της πληροφορίας: στάσεις, διαδρομές, δρομολόγια τιμές εισιτηρίων κλπ. Ακολουθώντας τους οδηγούς δημιουργίας ενός GTFS πακέτου, μία εταιρία συγκοινωνιών μπορεί να δημοσιεύσει τα δεδομένα της και να δώσει τη δυνατότητα στους προγραμματιστές να συμπεριλάβουν τα δρομολόγια της στις εφαρμογές τους [2].

Στον παρακάτω πίνακα φαίνονται τα αρχεία που μπορεί να περιλαμβάνονται σε μια συλλογή δεδομένων στο πρότυπο GTFS:

agency.txt	**Απαιτούμενο**	Οι εταιρίες (1 ή περισσότερες) που παρέχουν τις υπηρεσίες αστικών συγκοινωνιών.
stops.txt	**Απαιτούμενο**	Οι στάσεις από τις οποίες διέρχονται τα οχήματα.
routes.txt	**Απαιτούμενο**	Διαδρομές. Μια διαδρομή είναι ένα σύνολο δρομολογίων που αποτελούν μία συγκεκριμένη, αναγνωρίσιμη, υπηρεσία.
trips.txt	**Απαιτούμενο**	Δρομολόγια. Ένα δρομολόγιο είναι μια σειρά από δύο ή περισσότερες στάσεις σε συγκεκριμένο χρόνο.
stop_times.txt	**Απαιτούμενο**	Οι χρόνοι στους οποίου διέρχεται ένα συγκεκριμένο δρομολόγιο από μια συγκεκριμένη στάση.
calendar.txt	**Απαιτούμενο**	Ημερομηνίες στις οποίες λειτουργεί μια συγκεκριμένη υπηρεσία, καθώς και οι μέρες της εβδομάδας τις οποίες λειτουργεί.
calendar_dates.txt	Προαιρετικό	Εξαιρέσεις λειτουργίας συγκεκριμένων υπηρεσιών σε κάποιες ημερομηνίες/ημέρες.
fare_attributes.txt	Προαιρετικό	Πληροφορίες για τα ναύλα κάθε διαδρομής.
fare_rules.txt	Προαιρετικό	Κανόνες που διέπουν την πολιτική ναύλων για κάθε διαδρομή.
shapes.txt	Προαιρετικό	Κανόνες για την απεικόνιση των διαδρομών σε χάρτη.
frequencies.txt	Προαιρετικό	Συχνότητα δρομολογίων για διαδρομές με μεταβλητή συχνότητα.
transfers.txt	Προαιρετικό	Κανόνες για τη δημιουργία συνδέσεων μεταξύ διαδρομών
feed_info.txt	Προαιρετικό	Μετά-δεδομένα για την ίδια τη συλλογή δεδομένων, όπως το όνομα του εκδότη και η ημερομηνία έκδοσής της.

Πίνακας 4.1: Αρχεία στο πρότυπο GTFS [2]

Μπορεί κανείς να αντιληφθεί από τη σύντομη περιγραφή ότι το GTFS ενοποιεί τον τρόπο συγκέντρωσης και παρουσίασης των προγραμμάτων των αστικών συγκοινωνιών διευκολύνοντας έτσι την παραγωγή εφαρμογών που τα χρησιμοποιούν.

4.1.2 Δεδομένα ΟΑΣΑ

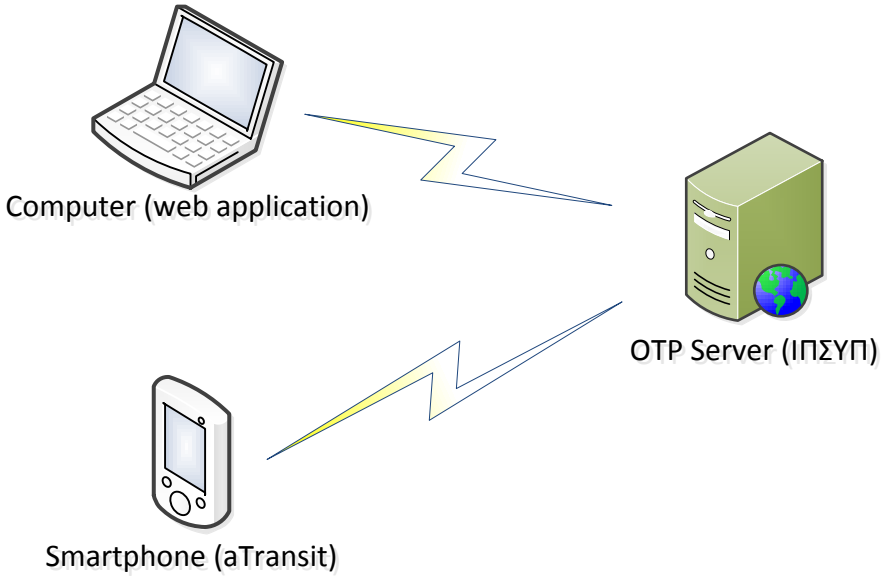
Τα δεδομένα που θα χρησιμοποιηθούν στη συγκεκριμένη εφαρμογή συγκεντρώθηκαν στο πρότυπο GTFS. Αν και δεν καλύπτουν το σύνολο των δεδομένων που προτείνονται από το πρότυπο, περιλαμβάνουν όλους του βασικούς πίνακες που είναι απαραίτητοι για να υλοποιηθεί η εφαρμογή που προδιαγράφηκε στα πλαίσια αυτής της εργασίας.

Πιο συγκεκριμένα συγκεντρώθηκαν τα αρχεία **agency.txt**, **trips.txt**, **routes.txt**, **stops.txt**, **stop_times.txt** και **frequencies.txt**. Αυτά καλύπτουν τις ανάγκες των offline δεδομένων της εφαρμογής. Για τη λειτουργία δρομολόγησης το σύστημα χρησιμοποιεί εξωτερικό server ο οποίος όμως βασίζεται στο ίδιο σετ δεδομένων. Από αυτά τα αρχεία θα προκύψει και η τοπική βάση δεδομένων της εφαρμογής με τρόπο που θα παρουσιαστεί στα επόμενα κεφάλαια που αφορούν τον σχεδιασμό και την υλοποίησή της.

4.2 OpenTripPlanner

Στο κεφάλαιο 3 περιγράφηκε το OTP ως υπηρεσία εύρεσης βέλτιστης διαδρομής προς το χρήστη. Όπως αναφέρθηκε κι εκεί, η καρδιά του συστήματος βρίσκεται στην αναπαράσταση των πολλαπλών μέσων σε γράφους και στους αλγόριθμους αναζήτησης βέλτιστων μονοπατιών πάνω σε αυτούς. Η πολυπλοκότητα αυτού του συστήματος, που βρίσκεται πίσω από την υπηρεσία του OTP, γίνεται φανερή αν θυμηθούμε την σύντομη αναφορά στους γνωστούς αλγορίθμους βέλτιστων μονοπατιών και τον τρόπο αναπαράστασης πολλαπλών μέσων που εφαρμόζει το OTP, από το κεφάλαιο 2.

Το σύστημα που είδαμε ως παράδειγμα στο κεφάλαιο 3 αφορά την εφαρμογή του OTP που έχει φτιαχτεί για το ΙΠΣΥΠ και θα χρησιμοποιηθεί και στην παρούσα εργασία. Εκεί η web εφαρμογή (front-end) αποτελεί τον client που επικοινωνεί με τον server όπου πραγματοποιείται η αναπαράσταση των μέσων σε γράφω και η εκτέλεση των αλγορίθμων. Στη συνέχεια αναλαμβάνει την παρουσίαση αυτών σε κείμενο και χάρτη προκειμένου να επικοινωνήσει το αποτέλεσμα αυτό με το χρήστη.



Εικόνα 4.1: Ο ρόλος του OTP ως server

Όπως και η web εφαρμογή εκείνου του συστήματος, έτσι και η παρούσα εργασία σκοπό έχει την παρουσίαση των αποτελεσμάτων των αλγορίθμων με τρόπο κατανοητό προς το χρήστη. Για το σκοπό αυτό θα χρησιμοποιηθεί ο ίδιος server του συστήματος του ΙΠΣΥΠ και στη συνέχεια θα εκμεταλλευτούν οι δυνατότητες του smartphone για την παρουσίαση των δεδομένων. Έτσι μπορούμε να πούμε ότι η παρούσα εφαρμογή θα αποτελέσει ένα ακόμη front-end του συστήματος, αυτή τη φορά για χρήστες smartphones.

Ο τρόπος επικοινωνίας του client με τον server θα εξηγηθεί στα επόμενα κεφάλαια αφού αφορά τεχνικές λεπτομέρειες του συστήματος. Είναι πάντως έτσι φτιαγμένο το OTP ώστε να επιτρέπει πολλούς client και διατηρεί τον server ανεξάρτητο από τον εξωτερικό κόσμο των clients.

4.3 OpenStreetMap

Το OpenStreetMap (OSM) [13] είναι ένα συνεργατικό έργο με σκοπό τη δημιουργία ενός δωρεάν και με δυνατότητα επεξεργασίας χάρτη του κόσμου. Οι χάρτες δημιουργήθηκαν χρησιμοποιώντας συσκευές GPS (Global Positioning System, Παγκόσμιο Σύστημα Θεσιθεσίας), αεροφωτογραφίες, άλλες δωρεάν πηγές πληροφοριών ή απλά με βάση τη γνώση των κατοίκων στις διάφορες περιοχές.

Η προσέγγιση του OSM στο χώρο των χαρτών είναι εμπνευσμένη από ιστοσελίδες τύπου wiki (π.χ.Wikipedia). Ο χάρτης μπορεί να επεξεργαστεί από οποιονδήποτε εγγεγραμμένο χρήστη, ενώ κάθε αλλαγή αποθηκεύεται ξεχωριστά με δυνατότητα διόρθωσης και αναθεώρησης σε προηγούμενες καταστάσεις.

Εκτός από το βασικό χάρτη του κόσμου, με πληροφορίες για σημεία ενδιαφέροντος, διαδρομές GPS κλπ., υπάρχουν και πιο εξειδικευμένοι χάρτες όπως χάρτες για ποδηλασία (OpenCycleMap), ναυτικοί χάρτες (OpenSeaMap) και άλλες υλοποιήσεις που συνεχώς εξελίσσονται και ενημερώνονται από μια μεγάλη κοινότητα χρηστών και προγραμματιστών [14].

Το OSM διαθέτει περισσότερους από 800000 εγγεγραμμένους χρήστες [15] (Σεπτέμβριος 2012) οι οποίοι συμμετέχουν ενεργά στην διαμόρφωση του παγκόσμιου χάρτη προσθέτοντας σημεία ενδιαφέροντος, διορθώνοντας ή δημιουργώντας λεπτομέρειες του χάρτη κλπ.. Το OSM καλύπτει το σύνολο του πλανήτη και παρέχει ανοιχτά τα δεδομένα του για προσωπική χρήση, σε διάφορα ανοιχτά πρότυπα αρχείων βασισμένα στην xml. Εκτός από τον παγκόσμιο χάρτη διατίθενται για download και αποσπάσματα του χάρτη για συγκεκριμένες χώρες ή πόλεις [16].

Υπάρχουν διάφορα εργαλεία με τα οποία μπορεί κανείς να επεξεργαστεί και να αποσπάσει κομμάτια του χάρτη ανάλογα με τη χρήστη που επιθυμεί. Ένα από αυτά είναι το osmosis [17] η χρήση του οποίου θα περιγραφή σε επόμενο κεφάλαιο που θα καταπιαστεί με τις τεχνικές λεπτομέρειες του συστήματος. Με χρήση εργαλείων όπως το mapsforge που περιγράφεται στην επόμενη παράγραφο μπορούν αυτοί οι χάρτες μπορούν να χρησιμοποιηθούν για offline παρουσίαση των δεδομένων τους σε smartphones.

4.4 Mapsforge – Βιβλιοθήκη για offline rendering χαρτών

Το mapsforge είναι μια βιβλιοθήκη ανοιχτού κώδικα για το Android OS που δημιουργήθηκε με σκοπό να προσφέρει ένα ελεύθερο και ανοιχτό εργαλείο που θα δώσει τη δυνατότητα στους προγραμματιστές να δημιουργούν εύκολα εφαρμογές που βασίζονται στο OpenStreetMap [18]. Η βιβλιοθήκη είναι δημιουργία του Computer Science Institute of Freie Universität Berlin Χρησιμοποιώντας αυτή τη βιβλιοθήκη, μας δίνεται δυνατότητα να προσθέσουμε στην εφαρμογή μας offline χάρτες (OpenStreetMap) προκειμένου να παρουσιάσουμε στο χρήστη δεδομένα χωρίς τη χρήση διαδικτύου.

Οι βασικές δυνατότητες που προσφέρει η βιβλιοθήκη είναι:

- Γρήγορη σχεδίαση χαρτών τοπικά στη συσκευή (χωρίς χρήση δικτύου)
- Ευέλικτο σύστημα επιστρώσεων (overlays) πάνω στο χάρτη (overlay API)
- Συμπυκνωμένος τύπος αρχείων για αποθήκευση των χαρτών
- Υποστήριξη multi-touch gestures (π.χ. pinch-to-zoom με χρήση δύο δαχτύλων)

4.5 Android

Το Android OS είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα για φορητές συσκευές όπως έξυπνα τηλέφωνα (smartphones) και υπολογιστές τύπου tablet. Βασίζεται στον πυρήνα, ανοιχτού κώδικα, Linux και αναπτύσσεται από τη Google με την υποστήριξη της Open Handset Alliance, η οποία αποτελεί μια κοινοπραξία από 86 εταιρίες υλικού, λογισμικού και τηλεπικοινωνιών [19]. Λόγω του ανοιχτού χαρακτήρα του και της υποστήριξης από πολλές και μεγάλες εταιρίες παραγωγής φορητών συσκευών, το Android έχει διαδοθεί ευρέως και χρησιμοποιείται από ένα μεγάλο μέρος χρηστών έξυπνων φορητών συσκευών. Τελευταία γίνεται χρήση του και σε άλλου είδους συσκευές όπως «έξυπνες» οικιακές συσκευές, συστήματα πολυμέσων και πλοήγησης αυτοκινήτων, ακόμα και ρολόγια χειρός.

Εκτός από το λειτουργικό σύστημα, η Google παρέχει στους και το Android SDK(Software Development Kit). Το Android SDK είναι μια σειρά εργαλείων που βοηθούν τους προγραμματιστές να αναπτύξουν εφαρμογές (applications) για συσκευές με Android. Ακόμη, οι προγραμματιστές έχουν τη δυνατότητα να μοιράσουν ή να πουλήσουν τις εφαρμογές τους μέσα από το Android Market, που συνοδεύει σχεδόν κάθε συσκευή με Android, ή μέσω άλλων προμηθευτών. Λόγω της μεγάλης απήχησης και του συνεχώς αυξανόμενου πλήθους χρηστών, το Android έχει γίνει δημοφιλές και στους προγραμματιστές με αποτέλεσμα να έχουν αναπτυχθεί χιλιάδες εφαρμογές, μέσα στα λίγα χρόνια ζωής του.

4.5.1 Αρχιτεκτονική

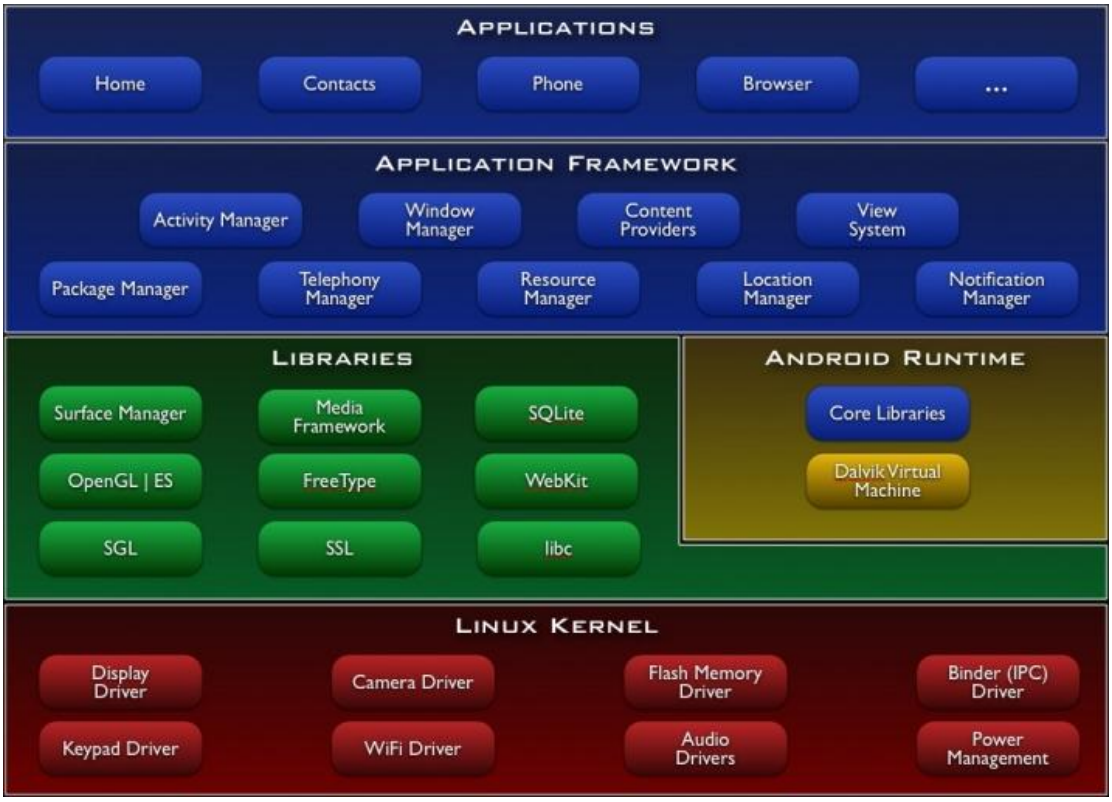
Στην **Εικόνα 4.2** απεικονίζεται η αρχιτεκτονική του Android OS. Στις επόμενες παραγράφους θα περιγραφούν με συντομία τα διάφορα μέρη του.

4.5.1.1 *Applications (εφαρμογές)*

Το Android συνοδεύεται από ένα σύνολο βασικών εφαρμογών που υπάρχουν προ-εγκατεστημένες σε όλες τις συσκευές που το χρησιμοποιούν. Μερικές από αυτές είναι email client, εφαρμογή για SMS, χάρτες, browser(φυλλομετρητής), επαφές και άλλες. Όλες οι εφαρμογές είναι γραμμένες σε Java.

4.5.1.2 *Application Framework*

Το Application Framework είναι αυτό που δίνει τη δυνατότητα στους προγραμματιστές να αποκτήσουν πρόσβαση στις διάφορες λειτουργίες και συστήματα της συσκευής, όπως να αποκτήσουν πληροφορίες για την τοποθεσία της ή να χρησιμοποιήσουν την φωτογραφική μηχανή, να καθορίσουν ένα ξυπνητήρι ή να παίξουν μουσική. Αυτά είναι λίγα αλλά χαρακτηριστικά παραδείγματα.



Εικόνα 4.2: Αρχιτεκτονική Android

Οι προγραμματιστές έχουν πλήρη πρόσβαση στα ίδια framework APIs (Application Programming Interfaces) που χρησιμοποιούνται και από της βασικές εφαρμογές. Η αρχιτεκτονική των εφαρμογών είναι φτιαγμένη έτσι ώστε να απλοποιεί την επαναχρησιμοποίηση των υποσυστημάτων. Κάθε εφαρμογή μπορεί να «δημοσιοποιήσει» στο σύστημα τις δυνατότητές της και οποιαδήποτε άλλη εφαρμογή μπορεί να χρησιμοποιήσει αυτές τις δυνατότητες, εφόσον βέβαια δεν εκπίπτουν προβλήματα αξιοπιστίας. Ο ίδιο μηχανισμός επιτρέπει στο χρήστη να επιλέγει το συστατικό ποιας εφαρμογής θα χρησιμοποιεί για να ολοκληρώσει κάποια συγκεκριμένη εργασία.

Κάθε εφαρμογή είναι ένα σύνολο από services (υπηρεσίες) και συστήματα. Μερικά από τα πιο βασικά είναι:

- Ένα πλούσιο και επεκτάσιμο σύνολο από όψεις (**Views**) που μπορούν να χρησιμοποιηθούν για την υλοποίηση μια εφαρμογής. Μερικές από αυτές είναι λίστες, πλέγματα, πλαίσια εισαγωγής κειμένου, κουμπιά, ακόμη και ένας ενσωματωμένος web browser
- Πάροχοι περιεχομένου (**Content Providers**) που επιτρέπουν στις εφαρμογές να αποκτούν πρόσβαση σε δεδομένα που ανήκουν σε άλλες εφαρμογές (όπως επαφές) ή να μοιραστούν τα δικά τους δεδομένα
- Ένα διαχειριστή πόρων (**Resource Manager**) ο οποίος δίνει πρόσβαση σε πόρους εκτός κώδικα, όπως μεταφρασμένο κείμενο, εικόνες ή αρχεία καθορισμού διάταξης σελίδας (layout files)

- Ένα διαχειριστή κοινοποιήσεων (**Notification Manager**) που δίνει τη δυνατότητα στις εφαρμογές να αναρτήσουν κοινοποιήσεις στην μπάρα κατάστασης (status bar) που υπάρχει συνήθως στο πάνω ή κάτω μέρος της οθόνης σε μια συσκευή Android
- Ένα διαχειριστή δραστηριοτήτων (**Activity Manager**) που αναλαμβάνει τη διαχείριση του κύκλου ζωής των εφαρμογών και παρέχει μία κοινή στοίβα πλοήγησης ανάμεσα στης εφαρμογές

4.5.1.3 *Libraries (βιβλιοθήκες)*

Το Android περιλαμβάνει ένα σύνολο από βιβλιοθήκες C/C++ που χρησιμοποιούνται από τα διάφορα υποσυστήματα του. Αυτές οι βιβλιοθήκες εκτίθενται στους προγραμματιστές μέσω του Android Application Framework που περιγράφηκε παραπάνω. Μερικές από τις βασικές βιβλιοθήκες είναι οι εξής:

- **System C library** (C βιβλιοθήκη συστήματος) – είναι μια υλοποίηση της standard βιβλιοθήκης C (libc) προερχόμενη από το BSD και παραμετροποιημένη για embedded Linux-based συσκευές
- **Media Libraries** (Βιβλιοθήκες πολυμέσων) – βασισμένες στο OpenCore της PacketVideo. Υποστηρίζουν αναπαραγωγή και εγγραφή πολλών μορφών ήχου, βίντεο και στατικής εικόνας, συμπεριλαμβάνοντας τα MPEG4, H.264, MP3, AAC, AMR, JPG και PNG
- **Surface Manager** – διαχειρίζεται της πρόσβαση στο υποσύστημα της οθόνης και συνθέτει 2D και 3D στρώματα γραφικών από πολλαπλές εφαρμογές
- **LibWebCore** – μία μηχανή web browser που χρησιμοποιείται τόσο από το browser του Android όσο και από την ενσωματώσιμη web view που αναφέρθηκε και πιο πάνω
- **SGL** – η μηχανή για τα 2D γραφικά
- **3D libraries** – μία υλοποίηση βασισμένη στα OpenGL ES 1.0 APIs. Οι βιβλιοθήκες χρησιμοποιούν είτε hardware 3D acceleration (όταν παρέχεται), είτε ένα βελτιστοποιημένο σύστημα software 3D acceleration
- **FreeType** – μηχανή για τη σχεδίαση των γραμμάτων στην οθόνη
- **SQLite** – μία ισχυρή και ελαφριά μηχανή σχεσιακής βάσης δεδομένων που είναι διαθέσιμη σε όλες τις εφαρμογές

4.5.1.4 *Android Runtime*

Το Android περιλαμβάνει ένα σύνολο από κεντρικές βιβλιοθήκες που παρέχουν σχεδόν όλες τις λειτουργίες που παρέχουν οι βιβλιοθήκες της Java.

Κάθε εφαρμογή Android «τρέχει» τη δική της διεργασία (process), με το δικό της instance της Dalvik virtual machine. Η μηχανή Dalvik γράφτηκε έτσι ώστε να μπορεί να εκτελεί αποτελεσματικά διαφορετικές Virtual Machines ταυτόχρονα. Η Dalvik VM εκτελεί αρχεία σε μορφή .dex (Dalvik Executable) τα οποία είναι βελτιστοποιημένα για ελάχιστο αποτύπωμα μνήμης (memory footprint).

Η Dalvik VM στηρίζεται στο Linux Kernel για βαθύτερες λειτουργίες όπως threading και διαχείριση μνήμης σε χαμηλό επίπεδο.

4.5.1.5 *Linux Kernel*

Το Android στηρίζεται στην έκδοση Linux 2.6 για κεντρικές υπηρεσίες συστήματος όπως ασφάλεια, διαχείριση μνήμης, διαχείριση διεργασιών, στοίβες δικτύου και οδηγούς συστήματος. Ο kernel επίσης χρησιμοποιείται και ως ένα επίπεδο ανάμεσα στο υλικό και την υπόλοιπη στοίβα λογισμικού.

4.5.2 Android SDK

Στα παραπάνω περιγράψαμε με σχετική συντομία το λειτουργικό σύστημα Android και τα βασικά χαρακτηριστικά στα οποία στηρίζει τη λειτουργία του καθώς και τη λειτουργία των εφαρμογών που γράφονται για αυτό. Εκτός από το λειτουργικό όμως, το Android προσφέρει και κάποια εργαλεία που βοηθούν την ανάπτυξη εφαρμογών που εκτελούνται στο περιβάλλον του. Το πιο βασικό από αυτά είναι το Android SDK (Software Development Kit).

Το Android SDK παρέχει μια σειρά από εργαλεία με τη μορφή προγραμμάτων, που μπορούν να εγκατασταθούν σαν ένα πακέτο σε όλα τα ευρέως χρησιμοποιούμενα λειτουργικά συστήματα (Windows, Mac OS, Linux) και τα οποία παρέχουν όλα όσα χρειάζεται ένα προγραμματιστής για να αρχίσει να προγραμματίζει για το Android. Στο πακέτο περιλαμβάνονται εργαλεία για επικοινωνία με συσκευές, δημιουργία εικονικών συσκευών, παραδείγματα εφαρμογών, απαραίτητες βιβλιοθήκες κ.α..

Ο προγραμματισμός σε αυτό γίνεται με τη γλώσσα JAVA. Υπάρχει, βέβαια, και η δυνατότητα να γράψει κάποιος σε C «κατεβαίνοντας» σε ένα επίπεδο πιο κοντά στο υλικό της συσκευής και αυτό επιτυγχάνεται χρησιμοποιώντας το Android NDK (Native Development Kit). Στην παρούσα εργασία δεν κρίθηκε απαραίτητο αυτό και επομένως η ανάπτυξη της εφαρμογής έγινε με χρήση του SDK και παραγραμματίζοντας σε JAVA.

Το Android SDK, καθώς και άλλα βοηθητικά εργαλεία, που χρησιμοποιήθηκαν κατά την ανάπτυξη της εφαρμογής, θα αναλυθούν εκτενέστερα σε επόμενη ενότητα.

Στην επόμενη ενότητα θα αναλύσουμε τις λειτουργικές απαιτήσεις της εφαρμογής που θα κατασκευαστεί στα πλαίσια αυτής εδώ της εργασίας.

5

Ανάλυση απαιτήσεων συστήματος

Σκοπός της διπλωματικής αυτής, όπως παρουσιάστηκε στα προηγούμενα κεφάλαια, είναι η δημιουργία μια εφαρμογής για κινητές συσκευές Android, που δίνει τη δυνατότητα στους χρήστες της να πληροφορηθούν για τα μέσα μαζικής μεταφοράς της περιοχής της Αθήνας, αλλά και να αναζητήσουν τη βέλτιστη διαδρομή για τον προορισμό τους χρησιμοποιώντας τις αστικές συγκοινωνίες. Στις παρακάτω παραγράφους περιγράφονται οι απαιτήσεις της εφαρμογής για αυτές τις δύο βασικές λειτουργίες και αναλύονται τα σενάρια χρήσης για κάθε μία από αυτές.

5.1 Δρομολόγηση με χρήση των αστικών συγκοινωνιών

Το πρώτο κομμάτι της εφαρμογής, δηλαδή αυτό που αφορά την **αναζήτηση βέλτιστων διαδρομών**, θα χρησιμοποιήσει έναν αλγόριθμο δρομολόγησης που βρίσκεται σε απομακρυσμένο server. Για το λόγο αυτό, η χρήση του απαιτεί τη σύνδεση στο internet, είτε μέσω Wi-Fi είτε με χρήση του δικτύου κινητής τηλεφωνίας.

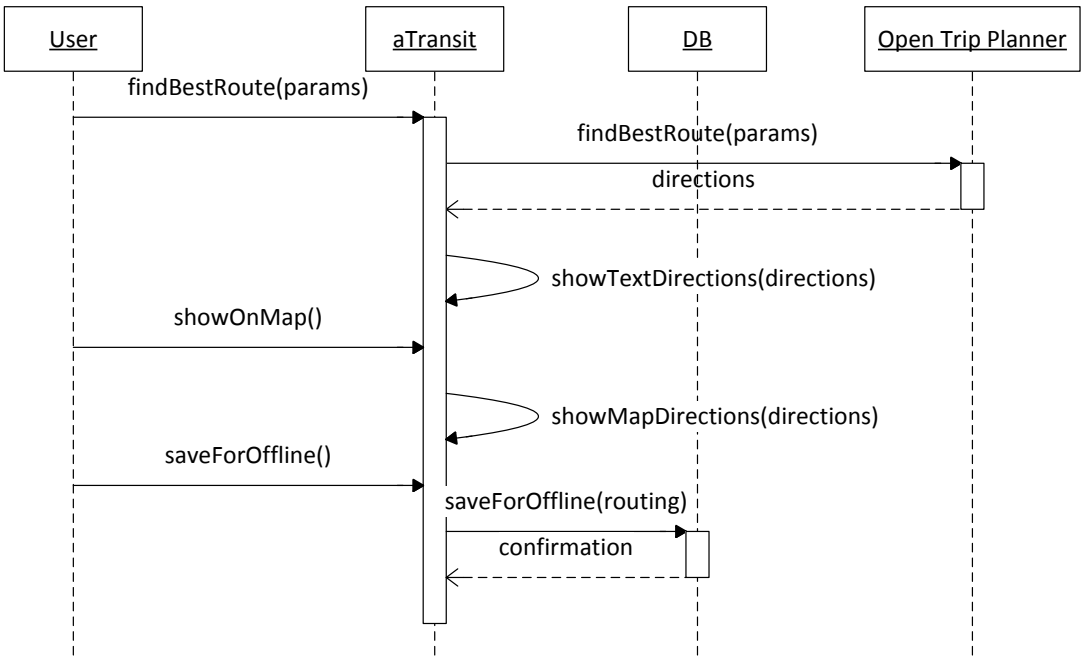
Το βασικό σενάριο χρήσης αυτής της λειτουργίας, περιλαμβάνει την πληκτρολόγηση διεύθυνσης εκκίνησης και προορισμού (ή χρήση της τρέχουσας τοποθεσίας), την παρουσίαση, με κείμενο, της προτεινόμενης διαδρομής και τέλος την παράθεση των δεδομένων αυτών στο χάρτη της εφαρμογής. Τα σημεία αφετηρίας και προορισμού θα πρέπει να μπορούν να επιλεγούν και μέσα από χάρτη, ενώ ο χρήστης θα έχει τη δυνατότητα να παραμετροποιήσει τον αλγόριθμο δρομολόγησης βάση συγκεκριμένων επιλογών. Τέλος, τα

αποτελέσματα της δρομολόγησης θα πρέπει να μπορούν να αποθηκευτούν στην τοπική μνήμη για μετέπειτα offline χρήση.

Πιο αναλυτικά το παραπάνω σενάριο χρήσης περιλαμβάνει τις εξής λειτουργίες:

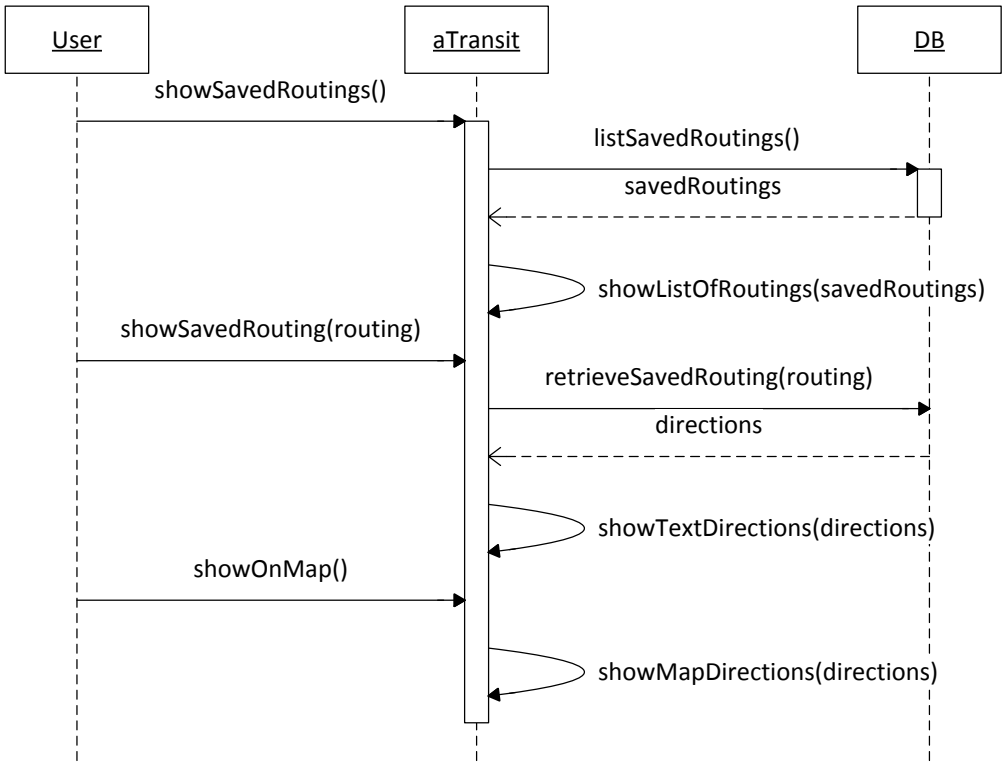
- Επιλογή αφετηρίας και προορισμού είτε πληκτρολογώντας τις αντίστοιχες διευθύνσεις, είτε επιλέγοντας σημεία στο χάρτη ή ακόμη χρησιμοποιώντας την τρέχουσα τοποθεσία ως κάποιο από τα δύο πεδία
- Καθορισμός επιθυμητής ημερομηνίας και ώρας εκκίνησης
- Παραμετροποίηση του αλγορίθμου με βάση τις επιλογές του χρήστη. Συγκεκριμένα ο χρήστης πρέπει να μπορεί να επιλέξει τον τρόπο βελτιστοποίησης της δρομολόγησης, με βάση το αν επιθυμεί το «ταξίδι με τις λιγότερες μετεπιβιβάσεις», το «γρηγορότερο ταξίδι» ή το «ασφαλέστερο ταξίδι». Ακόμη θα μπορεί να επιλέξει του τύπους των μέσων που τον εξυπηρετούν, αλλά και να θέσει ένα όριο στην απόσταση που προτίθεται να περπατήσει
- Αποστολή όλων των παραπάνω δεδομένων στο server προκειμένου να εκτελεστεί ο αλγόριθμος και λήψη του αποτελέσματος μέσω του διαδικτύου
- Παρουσίαση των αποτελεσμάτων σε κείμενο ή χάρτη
- Δυνατότητα αποθήκευσης των αποτελεσμάτων για μετέπειτα χρήση offline

Στο Διάγραμμα 5.1 φαίνεται εποπτικά το παραπάνω σενάριο χρήσης.

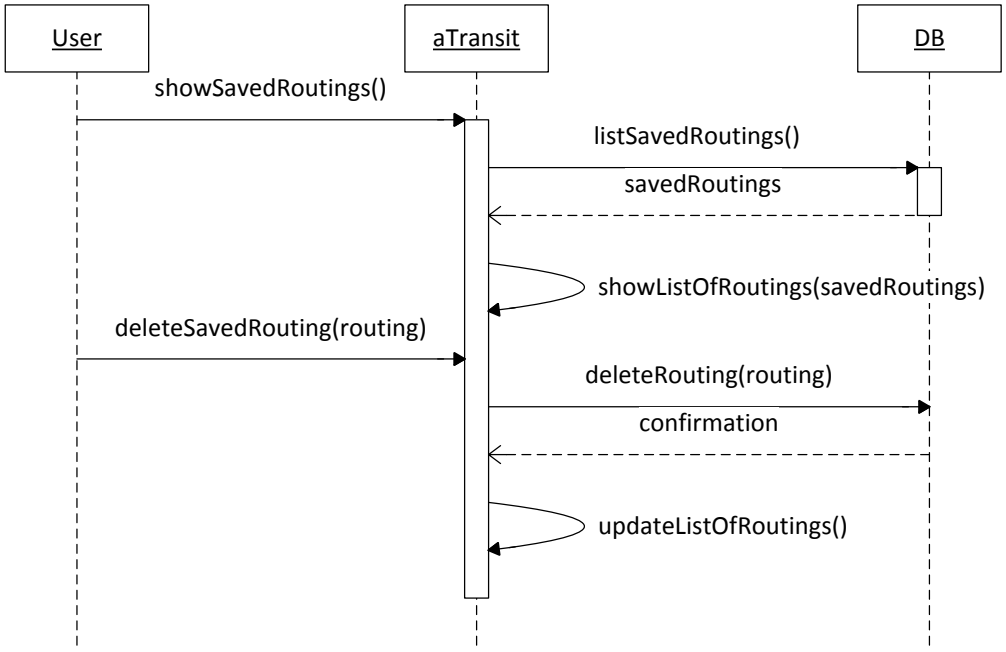


Διάγραμμα 5.1: Αναζήτηση και προβολή βέλτιστης διαδρομής

Επιπλέον, η εφαρμογή θα δίνει τη δυνατότητα στο χρήστη να έχει πρόσβαση στα αποθηκευμένα αποτελέσματα δρομολόγησης, είτε για να τα προβάλλει στο χάρτη, είτε για να τα διαγράψει. Τα δύο αυτά σενάρια χρήσης παρουσιάζονται καλύτερα στο Διάγραμμα 5.2 και στο Διάγραμμα 5.3 αντίστοιχα.



Διάγραμμα 5.2: Προβολή αποθηκευμένης δρομολόγησης

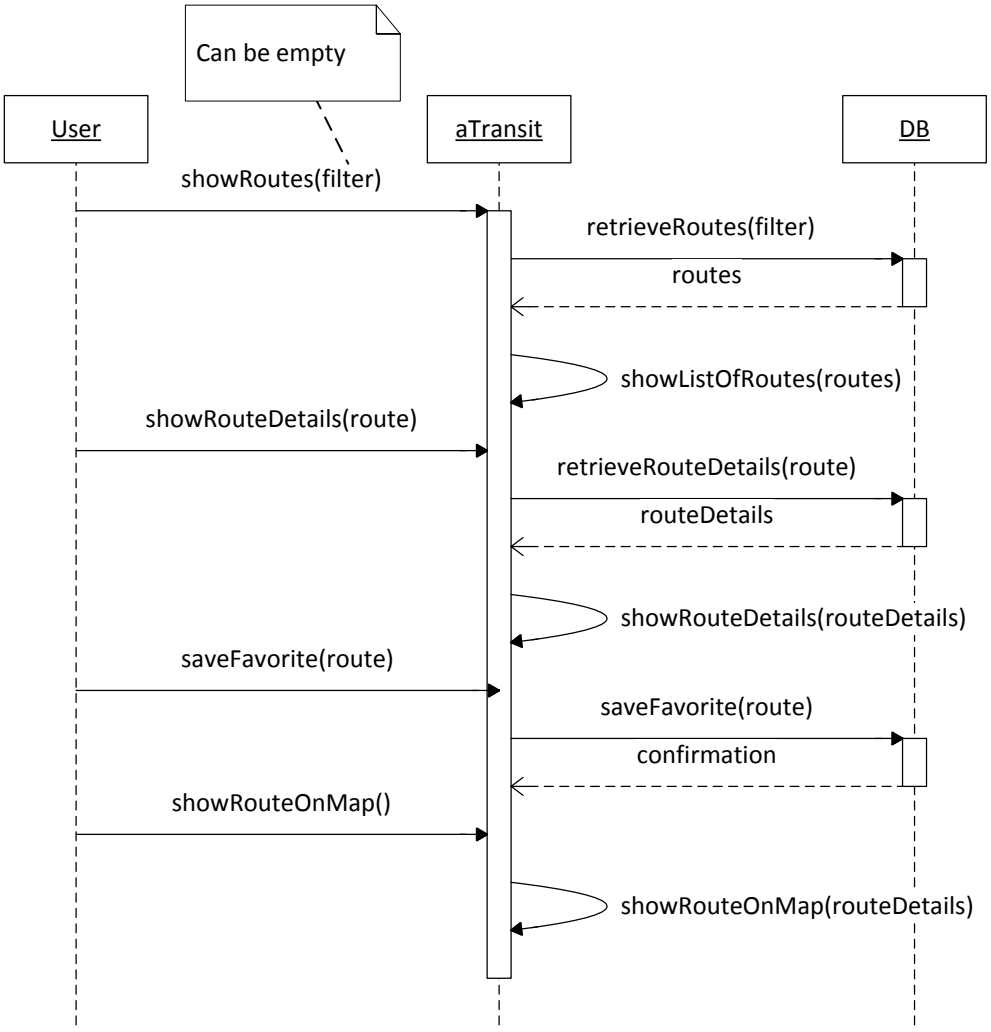


Διάγραμμα 5.3: Διαγραφή αποθηκευμένης δρομολόγησης

5.2 Παρουσίαση πληροφοριών για τις αστικές συγκοινωνίες

Σε αυτή την εργασία δόθηκε έμφαση, όπου ήταν δυνατό, στη χρήση offline πληροφοριών. Για αυτό το λόγο, το δεύτερο κομμάτι της εφαρμογής (η παρουσίαση στατικών πληροφοριών) έχει σκοπό την πλήρη ανεξαρτητοποίησή του από τη χρήση διαδικτύου.

Χρησιμοποιώντας μία τοπική βάση δεδομένων ο χρήστης θα έχει τη δυνατότητα να περιηγηθεί στις διαθέσιμες διαδρομές/στάσεις και να εντοπίσει λεπτομέρειες για αυτές, όπως συχνότητες δρομολογίων, ώρες έναρξης και τερματισμού κλπ.. Επιπρόσθετα θα του δίνεται η δυνατότητα να αποθηκεύσει «αγαπημένες» διαδρομές καθώς και να αναζητήσει διαδρομές και στάσεις. Τέλος, προσθέτοντας ένα υποσύστημα offline χαρτών, όλες αυτές οι πληροφορίες θα πρέπει να μπορούν να υπερτεθούν πάνω στο χάρτη, χωρίς να είναι απαραίτητη, και σε αυτή την περίπτωση, η χρήση διαδικτύου.

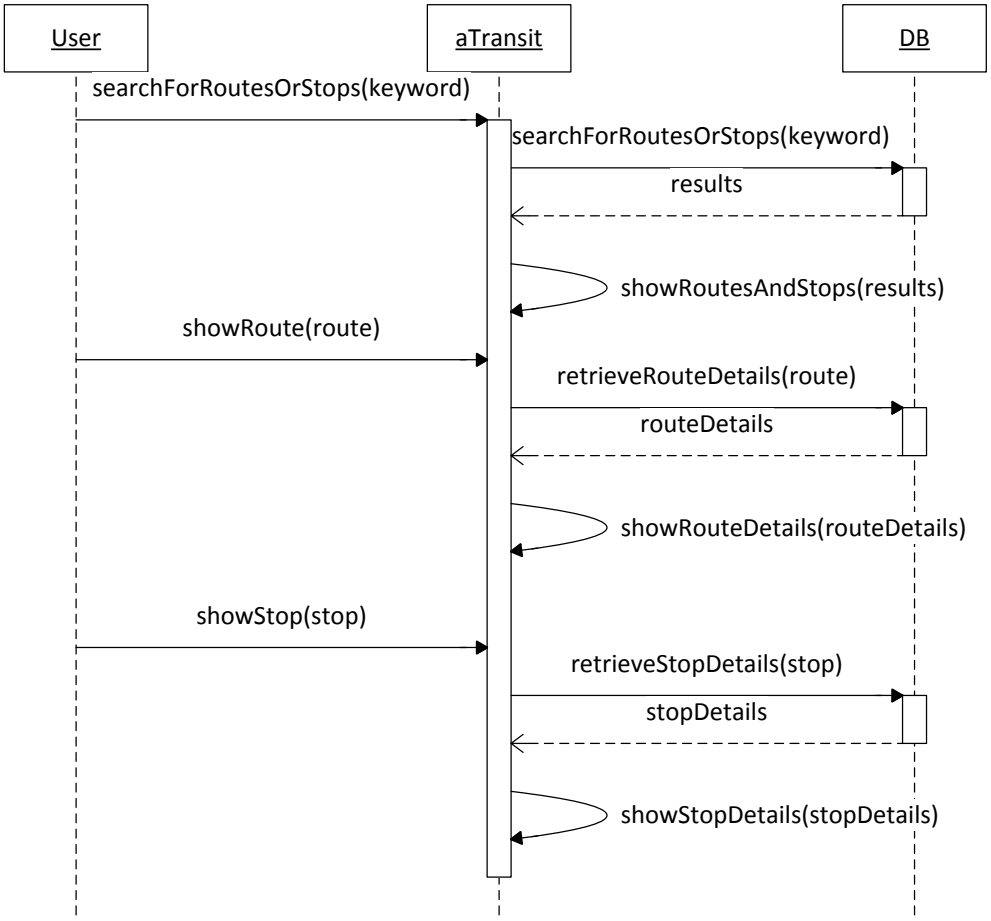


Διάγραμμα 5.4: Περιήγηση στις διαθέσιμες διαδρομές και στάσεις

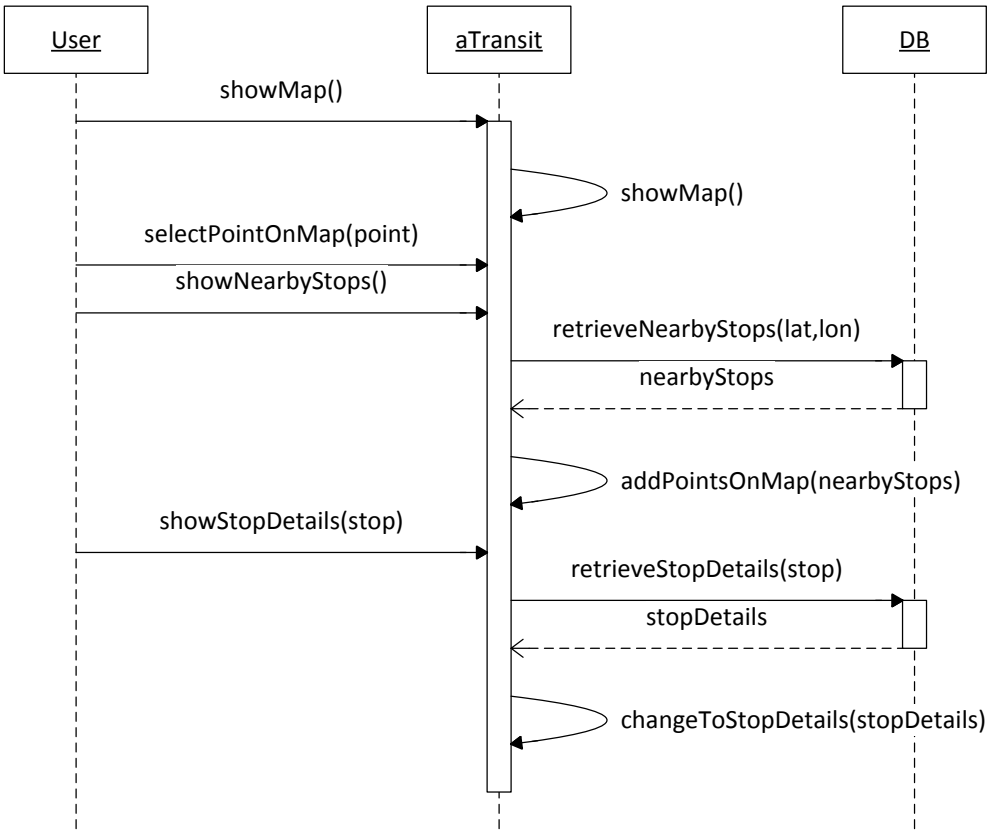
Αναλυτικά τα σενάρια χρήσης για αυτό το κομμάτι της εφαρμογής είναι τα εξής:

- **Περιήγηση στις διαθέσιμες διαδρομές και στάσεις**. Προβολή των λεπτομερειών τους ή αποθήκευση τους στα «Αγαπημένα» (Διάγραμμα 5.4)
- **Αναζήτηση διαδρομών και στάσεων** (Διάγραμμα 5.5)
- **Προβολή κοντινών στάσεων στο χάρτη** (Διάγραμμα 5.6)

Στα αντίστοιχα διαγράμματα μπορεί κανείς να δει πιο αναλυτικά τι περιλαμβάνει το κάθε ένα από αυτά τα σενάρια.



Διάγραμμα 5.5: Αναζήτηση διαδρομών και στάσεων



Διάγραμμα 5.6: Προβολή κοντινών στάσεων στο χάρτη

Σε αυτή την ενότητα είδαμε σύντομα με τη βοήθεια κατάλληλων σχημάτων τις λειτουργικές απαιτήσεις της εφαρμογής aTransit. Στο κεφάλαιο που ακολουθεί θα παρουσιαστεί η σχεδίαση του συστήματος με βάση τις απαιτήσεις που δόθηκαν εδώ. Θα σχεδιαστούν τα διάφορα υποσυστήματα της εφαρμογής που θα υλοποιούν τις απαιτήσεις και θα προδιαγραφεί ο τρόπος επικοινωνίας με τον server του OpenTripPlanner για τη δρομολόγηση.

6

Σχεδίαση συστήματος

Πριν προχωρήσουμε στην υλοποίηση της εφαρμογής σχεδιάζουμε το σύστημα με βάση της λειτουργικές απαιτήσεις που αναπτύχθηκαν στο προηγούμενο κεφάλαιο. Το παρόν κεφάλαιο, λοιπόν, είναι αφιερωμένο στην περιγραφή της εφαρμογής aTransit και στην μακροσκοπική παρουσίαση των υποσυστημάτων από τα οποία αποτελείται. Αρχικά παρουσιάζεται η αρχιτεκτονική του συστήματος, στη συνέχεια παρουσιάζονται εκτενέστερα οι υπομονάδες από τις οποίες δομείται και τέλος παρατίθεται το μοντέλο Οντοτήτων Συσχετίσεων στο οποίο θα βασιστεί η τοπική βάση δεδομένων.

6.1 Αρχιτεκτονική

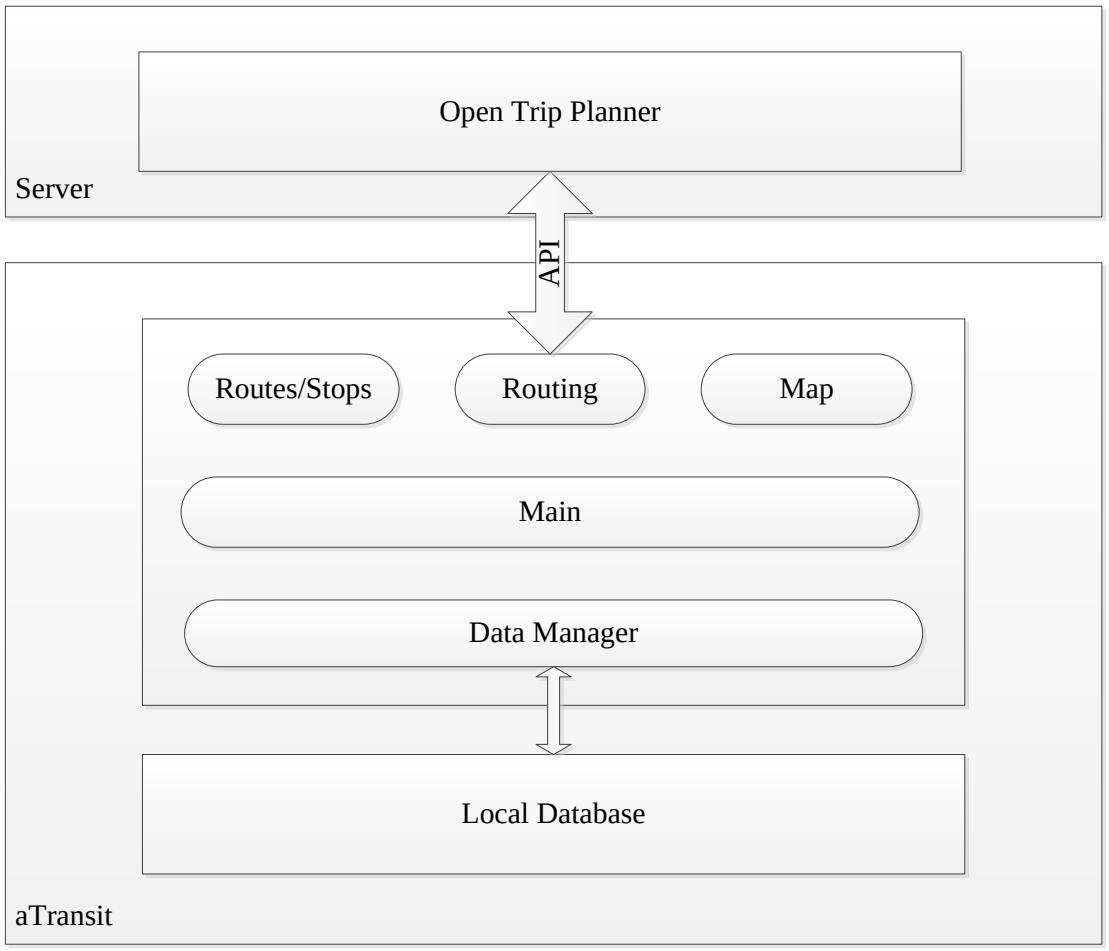
Οι απαιτήσεις του συστήματος υλοποιούνται από τα παρακάτω υποσυστήματα που το αποτελούν:

- **Main**: Είναι το κεντρικό υποσύστημα της εφαρμογής. Εμφανίζεται με την εκκίνηση της εφαρμογής και δίνει τη δυνατότητα στο χρήστη να εκκινήσει τις διάφορες λειτουργίες που υλοποιούνται από αυτό ή άλλα υποσυστήματα. Ακόμη δίνει πρόσβαση στις ρυθμίσεις της εφαρμογής.
- **Routes/Stops**: Αναλαμβάνει την παρουσίαση των διαδρομών/δρομολογίων των αστικών συγκοινωνιών. Επιτρέπει την επιλογή διαδρομών για περεταίρω πληροφορίες (στάσεις, συχνότητα δρομολογίων), την αποθήκευση διαδρομών στα «Αγαπημένα», ενώ στο ίδιο υποσύστημα κατατάσσουμε και την προβολή των «αγαπημένων».
- **Routing**: Αναλαμβάνει την επικοινωνία με τον απομακρυσμένο server και τον αλγόριθμο δρομολόγησης. Παραλαμβάνει τα δεδομένα τα οποία επεξεργάζεται και

παρουσιάζει σε κείμενο. Δίνει την επιλογή της αποθήκευσης της διαδρομής ή της προβολής της στο χάρτη.

- **Map**: Αναλαμβάνει την προβολή πληροφοριών στο χάρτη. Λειτουργεί ως συμπληρωματικό των άλλων υποσυστημάτων αναλαμβάνοντας την παράθεση διαδρομών και στάσεων στο χάρτη, αλλά και αυτόνομα αποτελώντας σημείο έναρξης για την αναζήτηση βέλτιστης διαδρομής ή κοντινών στάσεων.
- **Data Manager**: Είναι το υποσύστημα που αναλαμβάνει την επικοινωνία με την τοπική βάση δεδομένων.

Στο παρακάτω block diagram παρουσιάζεται συνοπτικά η αρχιτεκτονική του συστήματος:



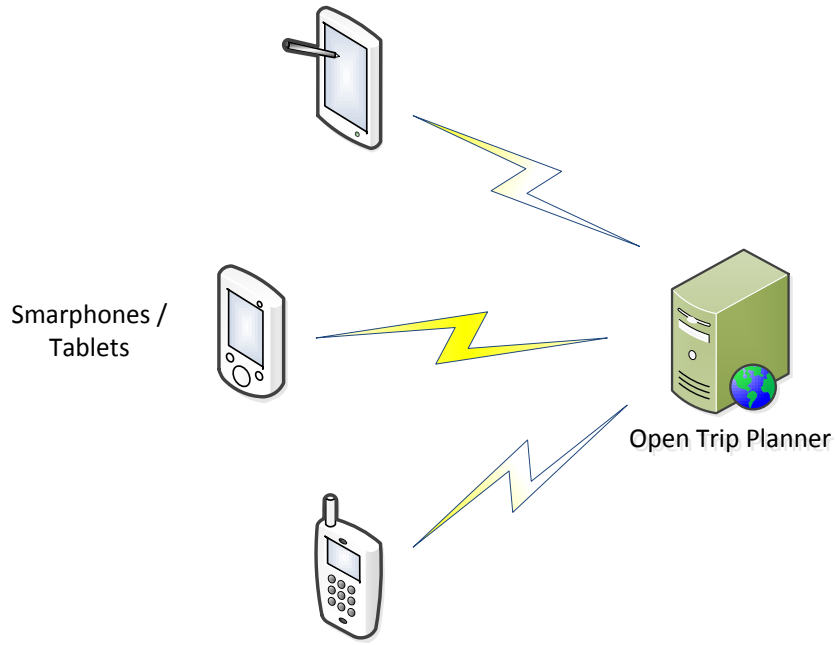
Εικόνα 6.1: block diagram αρχιτεκτονικής

Όπως φαίνεται και από την εικόνα, το υποσύστημα Routing είναι το μοναδικό που απαιτεί σύνδεση στο διαδίκτυο αφού επικοινωνεί με τον εξωτερικό server που αναλαμβάνει την επεξεργασία των δεδομένων, εντοπίζει τη βέλτιστη διαδρομή και αποστέλλει τα αποτελέσματα πίσω στην εφαρμογή.

Σε αυτό το σημείο αξίζει να δοθεί έμφαση στην ανεξαρτησία μεταξύ του client, που αποτελεί ουσιαστικά το σκοπό αυτής της διπλωματικής, και του server που είναι ανεξάρτητη δουλειά που έχει γίνει στο παρελθόν και βασίζεται στο OpenTripPlanner.

Το μοναδικό σημείο όπου χρειάζεται συμβατότητα μεταξύ των δύο συστημάτων είναι η έκδοση των GTFS αρχείων στο οποίο βασίστηκε η βάση δεδομένων τους. Γενικά ο client βλέπει το server σαν ένα **μαύρο κουτί (black box)** και θα μπορούσε να συνεργαστεί με οποιοδήποτε server χρησιμοποιεί αυτά τα δεδομένα και υλοποιεί το OpenTripPlanner API. Αυτό είναι ένα σημαντικό στοιχείο της αρχιτεκτονικής του client, αφού επιτρέπει περεταίρω βελτιστοποίηση των αλγορίθμων δρομολόγησης, ανεξάρτητα από τον τρόπο παρουσίασης.

Από τη μεριά του **ο client δεν αποθηκεύει απολύτως τίποτα στο server,** αλλά τον χρησιμοποιεί μόνο για εύρεση βέλτιστης διαδρομής στέλνοντας του τις απαιτήσεις του χρήστη και λαμβάνοντας το αποτέλεσμα. Το αποτέλεσμα αυτό μπορεί μετά να το αποθηκεύσει στην τοπική του βάση δεδομένων προκειμένου να το χρησιμοποιήσει αργότερα για offline παρουσίαση της βέλτιστης διαδρομής.



Εικόνα 6.2: Επικοινωνία πολλών clients με το OpenTripPlanner

Για τους λόγους που εξηγήθηκαν παραπάνω **δεν υπάρχει κάποιος σημαντικός περιορισμός στον αριθμό των χρηστών** που μπορούν να χρησιμοποιούν την εφαρμογή. Τα περισσότερα υποσυστήματα λειτουργούν χωρίς σύνδεση, ενώ για το server **δεν αποτελεί μεγάλο φόρτο** ο υπολογισμός των διαδρομών, μιας και έχει χτιστεί ο γράφος υπολογισμού κατά την εγκατάσταση. Από εκεί και πέρα οι αλγόριθμοί του είναι αρκετά «φτηνοί» και δεν αναμένεται να υπάρξει πρόβλημα ανταπόκρισης για αρκετά requests ταυτόχρονα.

Στην παράγραφο που ακολουθεί θα γίνει μια πιο λεπτομερείς περιγραφή των υποσυστημάτων που παρουσιάστηκαν.

6.2 Περιγραφή Λειτουργιών

6.2.1 Main

Η Main είναι το πρώτο υποσύστημα με το οποίο έρχεται σε επαφή ο χρήστης όταν ανοίγει την εφαρμογή. Από αυτή μπορεί να εκκινήσει τις διάφορες υπηρεσίες που υλοποιεί η εφαρμογή και περιγράφονται στη συνέχεια. Ουσιαστικά πρόκειται για εκκίνηση των άλλων υποσυστημάτων, για αυτό και η Main αποτελεί το κεντρικό υποσύστημα της εφαρμογής aTransit.

6.2.1.1 *Υπηρεσία εύρεσης βέλτιστης διαδρομής*

Η υπηρεσία εύρεσης της βέλτιστης διαδρομής αποτελεί ένα πολύ σημαντικό κομμάτι της εφαρμογής και για αυτό η έναρξη της γίνεται από το κύριο υποσύστημα της εφαρμογής, δηλαδή τη Main. Ο χρήστης μπορεί εδώ να επιλέξει σημείο αφετηρίας και προορισμού ή να χρησιμοποιήσει την τρέχουσα τοποθεσία του. Μπορεί ακόμη να επιλέξει την ώρα και την ημερομηνία στην οποία επιθυμεί να ταξιδέψει. Οι υπόλοιπες ρυθμίσεις του τρόπου δρομολόγησης γίνονται μέσα από το μενού των ρυθμίσεων της εφαρμογής που θα αναλυθούν παρακάτω, στην παρουσίαση της καρτέλας ρυθμίσεων.

Αφού ο χρήστης κάνεις της επιλογές του σχετικά με τις λεπτομέρειες τις επιθυμητής δρομολόγησης και εκκινήσει τη διαδικασία αναζήτησης, τότε η εφαρμογή αναλαμβάνει την κωδικοποίηση των διευθύνσεων αφετηρίας προορισμού σε συντεταγμένες και μαζί με τις υπόλοιπες επιλογές τα μεταφέρει στο υποσύστημα της δρομολόγησης (Routing) για την περεταίρω επεξεργασία και παρουσίαση.

6.2.1.2 *Καρτέλα ρυθμίσεων*

Η καρτέλα ρυθμίσεων αφορά το σύνολο της εφαρμογής και για αυτό κατατάσσεται στο υποσύστημα Main. Παρόλα αυτά η καρτέλα είναι προσβάσιμη και από άλλα υποσυστήματα της εφαρμογής μέσα από το μενού κάθε οθόνης. Με αυτό τον τρόπο δίνεται στο χρήστη η δυνατότητα γρήγορης αλλαγής ρυθμίσεων χωρίς να χρειάζεται να μεταβεί στην αρχική οθόνη.

Οι επιλογές που υπάρχουν στο μενού των ρυθμίσεων χωρίζονται σε 2 κατηγορίες και αφορούν το χάρτη και την υπηρεσία δρομολόγησης.

- **Χάρτης**: Ο χρήστης έχει τη δυνατότητα να επιλέξει τον τρόπο παρουσίασης του χάρτη. Μία από τις επιλογές αφορά τον offline χάρτη ο οποίος προκειμένου να χρησιμοποιηθεί θα πρέπει να «κατέβει» στη συσκευή. Η επιλογή αυτή δίνεται επίσης μέσα στο μενού των ρυθμίσεων. Άλλες επιλογές που αφορούν το χάρτη, είναι η

παρουσίαση της κλίμακας μέσα στο χάρτη και η αλλαγή του μεγέθους της γραμματοσειράς σε αυτόν.

- **Δρομολόγηση**: Ο χρήστης έχει 3 βασικές επιλογές όσο αφορά τον τρόπο λειτουργίας του αλγορίθμου δρομολόγησης του OpenTripPlanner. Αυτοί είναι: ο τρόπος υπολογισμού της διαδρομής (λιγότερες αλλαγές μέσων, γρηγορότερο ταξίδι, ασφαλέστερο ταξίδι), τρόπο ταξιδιού (συνδυασμός των διάφορων μέσων) και μέγιστη απόσταση περπατήματος.

6.2.1.3 *Εκκίνηση άλλων υποσυστημάτων*

Τέλος μέσα από την αρχική οθόνη της εφαρμογής ο χρήστης έχει τη δυνατότητα, πατώντας τα σχετικά εικονίδια, να εκκινήσει κάποια από τα υποσυστήματα της εφαρμογής, όπως το υποσύστημα της offline παρουσίασης διαδρομών/στάσεων (Routes/Stops) ή το χάρτη (Map) για να εντοπίσει κοντινές στάσεις ή να επιλέξει σημεία δρομολόγησης μέσα από αυτόν.

6.2.2 Online υποσυστήματα

6.2.2.1 *Routing*

Το υποσύστημα Routing αναλαμβάνει την αναζήτηση και παρουσίαση μιας βέλτιστης διαδρομής. Η λειτουργία του αρχίζει μετά την κωδικοποίηση των δεδομένων που εισήχθησαν από το χρήστη και περιλαμβάνει την επικοινωνία με το server δρομολόγησης, την παρουσίαση της βέλτιστης διαδρομής με κείμενο ή χάρτη και την αποθήκευση της αναζήτησης.

Αναλυτικότερα, όταν ο χρήστης επιλέξει σημείο αφετηρίας και προορισμού το υποσύστημα δρομολόγησης αναλαμβάνει δράση, συγκεντρώνοντας τις επιλογές του χρήστη, από τα πεδία αναζήτησης και από τις ρυθμίσεις της εφαρμογής, και αποστέλλοντας τες στο απομακρυσμένο σύστημα δρομολόγησης μέσω δικτύου. Ο χρήστης ενημερώνεται ότι αναμένεται απάντηση από το σύστημα. Όταν η απάντηση επιστρέψει στην κινητή συσκευή, το σύστημα αναλαμβάνει την αποκωδικοποίησή της την παρουσίαση της σε μορφή κειμένου στο χρήστη. Στη συνέχει δίνεται η δυνατότητα προβολής των λεπτομερειών της διαδρομής στο χάρτη, με χρήση του υποσυστήματος OpenStreetMap, ενώ υπάρχει και η επιλογή αποθήκευσης για μετέπειτα offline χρήση.

6.2.3 Offline υποσυστήματα

6.2.3.1 *Routes/Stops*

Το υποσύστημα αυτό υλοποιεί την απαίτηση της εφαρμογής για παροχή offline πληροφοριών για τις αστικές συγκοινωνίες της περιοχής. Ο χρήσης μέσα από αυτό μπορεί να περιηγηθεί στις διαθέσιμες διαδρομές/γραμμές αστικών συγκοινωνιών, να τις φιλτράρει ανάλογα με το

είδος τους και να τις αποθηκεύσει στα «Αγαπημένα» ώστε να έχει εύκολη πρόσβαση σε αυτές αργότερα.

Επιπλέον επιλέγοντας κάποια διαδρομή, ο χρήστης μπορεί να δει την αναλυτική περιγραφή της. Εκεί προβάλλονται οι στάσεις της διαδρομής με τη σειρά με την οποία τις επισκέπτεται. Εδώ υπάρχει η επιλογή της αλλαγής κατεύθυνσης (τέλος προς αρχή), καθώς πολλές διαδρομές περάνε από διαφορετικούς δρόμους κατά την επιστροφή τους. Ακόμη δίνεται η επιλογή της παρουσίασης στο χάρτη, μέσα από το υποσύστημα OpenStreetMap, ενώ παρουσιάζονται και λεπτομέρειες όπως συχνότητα δρομολογίων και ώρες εκκίνησης και τερματισμού αυτών.

Στο υποσύστημα αυτό κατατάσσουμε και την ανάλυση των στάσεων. Καθώς ο χρήστης περιηγείται σε μία διαδρομή μπορεί να επιλέξει κάποια από τις στάσεις της. Αυτό που γίνεται τότε είναι εμφάνιση των διαδρομών που διέρχονται από αυτή τη στάση, έτσι ώστε να δοθεί η πληροφορία για τα δρομολόγια που συνδέονται με τη διαδρομή αυτή. Σε αυτή την οθόνη επίσης μπορεί κάποιος να βρεθεί μετά από αναζήτηση για στάσεις μέσα από το σύστημα αναζήτησης που αναλύεται παρακάτω.

6.2.3.1.1 Αναζήτηση διαδρομών/στάσεων

Όπως και η καρτέλα ρυθμίσεων, έτσι και η αναζήτηση μπορεί να ξεκινήσει από διάφορα σημεία της εφαρμογής αλλά, λόγω του ότι τα αποτελέσματα που παρέχει αφορούν την κατηγορία offline πληροφοριών για διαδρομές και στάσεις, την κατατάσσουμε στο υποσύστημα Routes/Stops.

Εδώ ο χρήστης καλείται να πληκτρολογήσει τον τίτλο ή κομμάτι από τον τίτλο ή των αριθμό κάποιας γραμμής ή στάσης προκειμένου να έχει γρήγορη πρόσβαση σε αυτές. Το αποτέλεσμα περιλαμβάνει δύο λίστες. Μία με τις διαδρομές και μία με τις στάσεις που ταίριαξαν με τη συγκεκριμένη αναζήτηση. Ανάλογα με την επόμενη επιλογή του χρήστη γίνεται αναλυτική παρουσίαση μιας διαδρομής ή εμφάνιση των διερχόμενων γραμμών από κάποια στάση.

6.2.3.2 *Map*

Το υποσύστημα χαρτών αναλαμβάνει την προβολή στο χάρτη οποιασδήποτε πληροφορίας της εφαρμογής μπορεί να παρουσιαστεί με αυτό τον τρόπο. Αποτελεί έτσι σημαντικό κομμάτι της εφαρμογή και, εκτός από συμπληρωματικό στοιχείο, μπορεί να εκτελεστεί και αυτόνομα προσφέροντας δικές του υπηρεσίες ή αποτελώντας σημείο εκκίνησης για υπηρεσίες άλλον υποσυστημάτων.

Ως αυτόνομο σύστημα, ο χάρτης, παρέχει μία άποψη της περιοχής του χρήστη και προσφέρει μια σειρά από υπηρεσίες:

- **Εντοπισμός θέσης.** Μέσα από το μενού της οθόνης, ο χρήστης μπορεί να επιλέξει την εμφάνιση της θέσης του στο χάρτη. Ο εντοπισμός της θέσης γίνεται τόσο με χρήση του GPS της συσκευής, όσο και μέσω των κεραιών κινητής τηλεφωνίας. Εφόσον κάνει αυτή την επιλογή ο χρήστης θα μπορεί να παρακολουθεί τη θέση του στο χάρτη μέχρι να επιλέξει την παύση του εντοπισμού ή να κλείσει το υποσύστημα χαρτών.
- **Εντοπισμός κοντινών στάσεων**. Με την επιλογή ενός σημείου πάνω στο χάρτη ο χρήστης έχει τη δυνατότητα κάποιον επιλογών για αυτό. Μία από αυτές είναι η εύρεση κοντινών στάσεων. Μέσω του Data Manager γίνεται μία αναζήτηση με βάση την ακτίνα γύρω από το σημείο που επιλέχθηκε και οι στάσεις που βρίσκονται μέσα σε αυτή εμφανίζονται στο χάρτη. Επιλέγοντας κάποια από αυτές ο εμφανίζεται μία λίστα με τις διαδρομές που διέρχονται από αυτή.
- **Καθορισμός σημείου αφετηρίας/τερματισμού για δρομολόγηση**. Οι άλλες δύο επιλογές που έχει ο χρήστης στο μενού επιλογών για ένα σημείο, είναι ο καθορισμός σημείου αφετηρίας και τερματισμού για εύρεση της βέλτιστης διαδρομής μεταξύ τους. Όταν ο χρήστης έχει καθορίσει και τα δύο μπορεί να εκκινήσει τη διαδικασία δρομολόγησης απ' όπου αναλαμβάνει το υποσύστημα Routing.

6.2.3.3 *Data Manager*

Ο Data Manager αναλαμβάνει την επικοινωνία με την τοπική βάση δεδομένων προκειμένου να φέρει στην οθόνη του χρήστη δεδομένα που έχουν αποθηκευτεί από αυτόν ή από την ίδια την εφαρμογή τοπικά στη συσκευή του.

Μερικές από τις βασικές υπηρεσίες που προσφέρει είναι:

- **Εύρεση διαδρομών**: Μία βασική λειτουργία του Data Manager είναι η λήψη από τη βάση δεδομένων της λίστας μες τις διαθέσιμες διαδρομές για την περιοχή της Αθήνας. Αυτό μπορεί να γίνει προσθέτοντας και κάποια φίλτρα για το είδος των διαδρομών (όπως μετρό, λεωφορείο κλπ.).
- **Συγκέντρωση πληροφοριών διαδρομής**: Με την επιλογή μίας διαδρομής από το χρήστη ο Data Manager αναλαμβάνει τη συγκέντρωση των πληροφοριών για τη διαδρομή αυτή (στάσεις, συχνότητα δρομολογίων, πρώτο/τελευταίο δρομολόγιο κλπ)
- **Αναζήτηση διαδρομών/στάσεων**: Ο χρήστης μπορεί να αναζητήσει διαδρομές και στάσεις πληκτρολογώντας κομμάτι του τίτλου ή του αριθμού τους. Ο Data Manager αναλαμβάνει την ερώτηση αυτή στη βάση.
- **Αποθήκευση/ανάγνωση «Αγαπημένων»**: Μέσα από το υποσύστημα Routes/Stops ο χρήστης έχει τη δυνατότητα να αποθηκεύσει τις διαδρομές που χρησιμοποιεί συχνά ως «αγαπημένες». Αυτή η διαδικασία, καθώς και η μετέπειτα εύρεσή τους στη βάση, γίνεται μέσω του Data Manager.
- **Αποθήκευση/ανάγνωση δρομολογήσεων**: Μετά από μία επιτυχή αναζήτηση βέλτιστης διαδρομής ο χρήστης μπορεί να αποθηκεύσει τα αποτελέσματα για μετέπειτα χρήση χωρίς την ανάγκη δικτύου. Αυτή τη λειτουργία αναλαμβάνει ο Data Manager σε συνεργασία με την τοπική βάση δεδομένων.

Ως το μέσω επικοινωνίας με τη βάση δεδομένων, ο Data Manager αναλαμβάνει και άλλες μικρές λειτουργίες του συστήματος. Παραπάνω έγινε μία παρουσίαση των σημαντικότερων

από αυτές που παίζουν μεγάλο ρόλο στη λειτουργία των υπόλοιπων υπομονάδων του συστήματος.

6.3 Μοντέλο Οντοτήτων Συσχετίσεων

Η βάση δεδομένων που χρησιμοποιεί η εφαρμογή αποτελείται κυρίως από τα δεδομένα που προέρχονται από τα GTFS αρχεία, αλλά υπάρχουν και κάποιες προσθήκες που αφορούν την αποθήκευση των «αγαπημένων» δρομολογίων, αλλά και των δρομολογήσεων για μετέπειτα offline χρήση.

Όσο αφορά τα GTFS δεδομένα, η μορφή των οποίων αναλύθηκε στην παράγραφο 4.1, κάθε αρχείο αποτελεί ουσιαστικά ένα πίνακα. Δεν ενσωματώθηκαν όμως όλα τα αρχεία, αφού ο ΟΑΣΑ δεν χρησιμοποιεί όλο το πρότυπο GTFS, αλλά κυρίως τους απαιτούμενους πίνακες και πεδία.

Στη συνέχεια παρουσιάζεται με συντομία το σύνολο τον οντοτήτων και των συσχετίσεων που προδιαγράφουν τη βάση δεδομένων που θα δημιουργηθεί και θα αποθηκευτεί στην εσωτερική μνήμη του τηλεφώνου.

6.3.1 Οντότητες

Καθεμία από τις παρακάτω οντότητες διαθέτει από ένα κλειδί (**_id**) που είναι ένας ακέραιος που αντιπροσωπεύει κάθε εγγραφή. Αυτό το πεδίο είναι απαραίτητο για να λειτουργήσει ομαλά η βάση δεδομένων με τις διαθέσιμες κλάσεις και μεθόδους του Android.

agency: Αντιπροσωπεύει τις διάφορες εταιρίες που εκτελούν τα δρομολόγια των ΜΜΜ (ΕΘΕΛ, ΗΛΠΑΠ, ΗΣΑΠ κλπ). Περιλαμβάνει τα ακόλουθα πεδία:

- **αgency_id**: το αναγνωριστικό της εταιρίας
- **agency_name**: το όνομα της εταιρίας
- **agency_url**: η ιστοσελίδα της εταιρίας
- **agency_timezone**: η ζώνη ώρας όπου βρίσκεται η εταιρία
- **agency_lang**: η βασική γλώσσα που χρησιμοποιεί η εταιρία
- **agency_phone**: το τηλέφωνο της εταιρίας

stops: Αντιπροσωπεύει όλες τις στάσεις που υπάρχουν στην περιοχή της Αθήνας. Κάθε εγγραφή της διαθέτει τα εξής πεδία:

- **stop_id**: το αναγνωριστικό της στάσης
- **stop_code**: ο κωδικός της διαδρομής (μπορεί να είναι διαφορετικός από το stop_id και να γίνεται εμφανής και στους επιβάτες)
- **stop_name**: το όνομα της στάσης
- **stop_desc**: μία περιγραφή της στάσης
- **stop_lat**: γεωγραφικό πλάτος της στάσης
- **stop_lon**: γεωγραφικό μήκος της στάσης
- **zone_id**: ο κωδικός της ζώνης στην οποία ανήκει η στάση

- **stop_url**: ηλεκτρονική διεύθυνση που αφορά τη στάση
- **location_type**: ένας ακέραιος που δείχνει αν η στάση βρίσκεται σε κάποιο σταθμό ή είναι απλή στάση
    - **0 ή null** – Απλή στάση
    - **1** - Σταθμός
- **parent_station**: ο κωδικός του σταθμού στον οποίο ανήκει η στάση (συμπληρώνεται μόνο για εγγραφές που το location_type είναι 1)

routes: Αντιπροσωπεύει τις διαδρομές που υπάρχουν στην περιοχή της Αθήνας. Διαθέτει, εκτός από το _id τα εξής πεδία:

- **route_id**: το αναγνωριστικό της διαδρομής
- **agency_id**: το αναγνωριστικό της εταιρίας που εκτελεί τη διαδρομή
- **route_short_name**: ένα σύντομο αναγνωριστικό της διαδρομής
- **route_long_name**: ένα πιο πλήρες αναγνωριστικό της διαδρομής
- **route_desc**: μια περιγραφή της διαδρομής
- **route_type**: ένας ακέραιος που δείχνει τον τύπο της διαδρομής. Στη συγκεκριμένη εφαρμογή χρησιμοποιούνται τα εξής:
    - **0** – Τραμ – Προαστιακός
    - **1** – Μετρό – ΗΣΑΠ
    - **3** – Λεωφορείο – Τρόλεϊ
- **route_url**: ιστοσελίδα της διαδρομής
- **route_color**: χρώμα διαδρομής
- **route_text_color**: χρώμα κειμένου διαδρομής (τέτοιο ώστε να ταιριάζει με το χρώμα της διαδρομής)

trips: Αντιπροσωπεύει συγκεκριμένα δρομολόγια για τις διαδρομές που ανήκουν στον πίνακα routes. Διαθέτει τα εξής πεδία:

- **route_id**: το αναγνωριστικό της διαδρομής στην οποία αντιστοιχεί αυτό το δρομολόγιο
- **service_id**: ορίζει τις ημερομηνίες που λειτουργεί το δρομολόγιο
- **trip_id**: το αναγνωριστικό του δρομολογίου
- **trip_headsign**: ο τίτλος της διαδρομής (όπως αυτή φαίνεται στους επιβάτες)
- **trip_short_name**: ένα σύντομο όνομα για το δρομολόγιο
- **direction_id**: ένας ακέραιος που δείχνει την κατεύθυνση του δρομολογίου. Μπορεί να είναι:
    - **0** – Κανονική κατεύθυνση
    - **1** – Αντίθετη κατεύθυνση
- **block_id**: ένα μπλοκ δρομολογίων που συνδέονται μεταξύ τους (εκτελούνται από το ίδιο όχημα). Κάθε block_id μπορεί να αναφέρεται από πολλές εγγραφές στον πίνακα trips.
- **shape_id**: το σχήμα της διαδρομής που πρέπει να σχεδιαστεί στο χάρτη

stop_times: Συνδέει στάσεις με δρομολόγια και συμπληρώνει έτσι το βασικό πίνακα δρομολογίων των ΜΜΜ της περιοχής. Περιέχει τα εξής πεδία:

- **trip_id**: ο κωδικός του δρομολογίου στο οποίο αναφέρεται η εγγραφή (πρέπει να υπάρχει το δρομολόγιο στον πίνακα **trips**)

- **arrival_time**: η ώρα άφιξης του συγκεκριμένου δρομολογίου στη συγκεκριμένη στάση
- **departure_time**: η ώρα αναχώρησης του συγκεκριμένου δρομολογίου από τη συγκεκριμένη στάση
- **stop_id**: ο κωδικός της στάσης στην οποία αναφέρεται αυτή η εγγραφή
- **stop_sequence**: η σειρά αυτής της στάσης στο συγκεκριμένο δρομολόγιο
- **pickup_type**: ορίζει τον τρόπο με τον οποίο μπορεί κάποιος να επιβιβαστεί στο δρομολόγιο από αυτή τη στάση
    - **0** – Προγραμματισμένη επιβίβαση
    - **1** – Δεν επιτρέπεται η επιβίβαση
    - **2** – Πρέπει να προηγηθεί τηλέφωνο στην εταιρία που εκτελεί το δρομολόγιο
    - **3** – Πρέπει να προηγηθεί συνεννόηση με τον οδηγό
- **drop_off_type**: ορίζει τον τρόπο με τον οποίο μπορεί κάποιος να αποβιβαστεί από το δρομολόγιο σε αυτή τη στάση (οι δυνατές τιμές είναι αντίστοιχες με αυτές του pickup_type)

frequencies: Περιλαμβάνει στοιχεία που αφορούν τις συχνότητες των διάφορων δρομολογίων. Διαθέτει τα παρακάτω πεδία:

- **trip_id**: το αναγνωριστικό του δρομολογίου για το οποίο ισχύει η συγκεκριμένη συχνότητα
- **start_time**: ώρα έναρξης δρομολογίου
- **end_time**: ώρα τελευταίου δρομολογίου
- **headway_secs**: συχνότητα μεταξύ των δρομολογίων (σε δευτερόλεπτα)

favorite_routes: Αντιπροσωπεύει της «αγαπημένες» διαδρομές του χρήστη της εφαρμογής. Το μόνο πεδίο του, εκτός από το _id του, είναι το **route_rowid** που αντιστοιχεί στο _id μιας εγγραφής στον πίνακα routes

saved_routings: Αντιπροσωπεύει τις δρομολογήσεις που έχει αποθηκεύσει ο χρήστης για offline χρήση. Περιέχει τα εξής πεδία:

- **routing_title**: ένας χαρακτηριστικός τίτλος για τη δρομολόγηση
- **date_added**: η ημερομηνία αποθήκευσης της δρομολόγησης
- **routing_origin**: η αφετηρία της δρομολόγησης
- **routing_dest**: ο προορισμός της δρομολόγησης
- **routing_xml**: ένα string που περιέχει το XML Response του OpenTripPlanner για τη συγκεκριμένη δρομολόγηση. Εδώ βρίσκονται όλες οι απαραίτητες πληροφορίες για την επίδειξη των αποτελεσμάτων στο χρήστη.

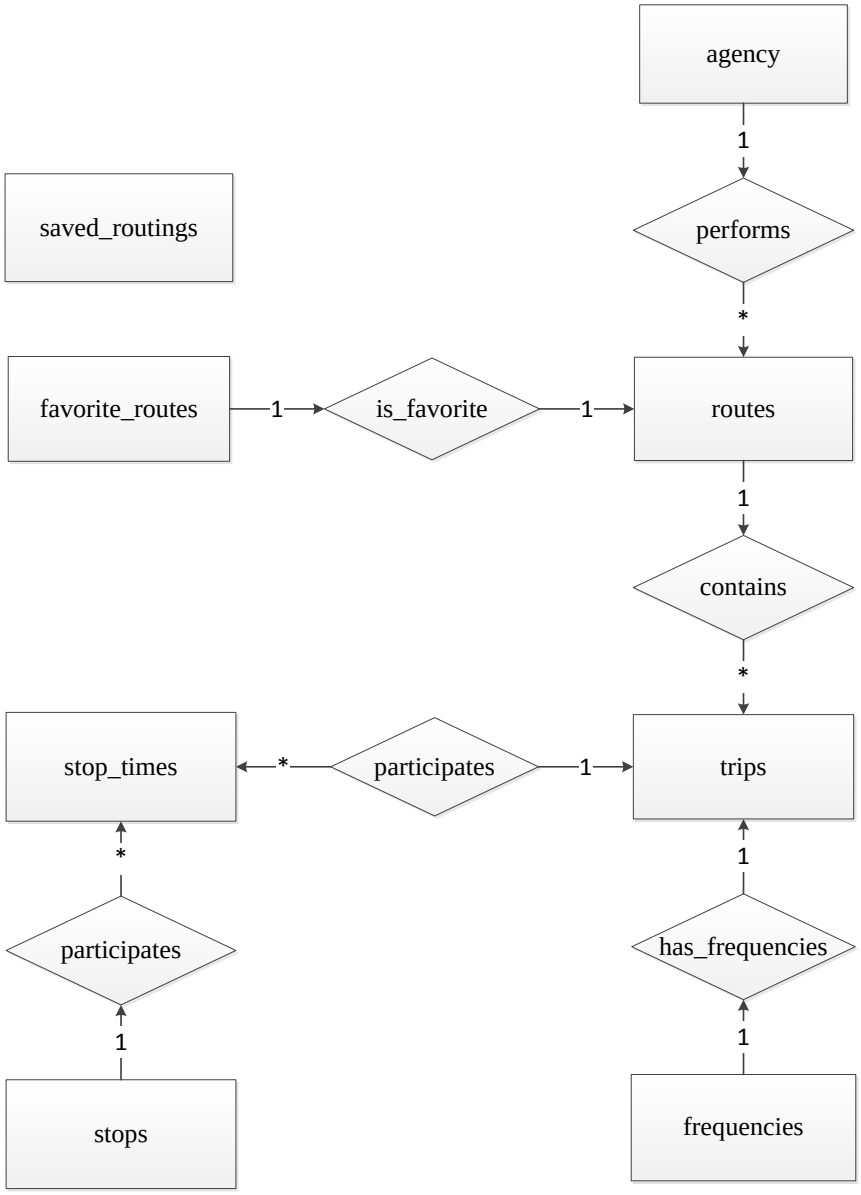
6.3.2 Συσχετίσεις

Οι οντότητες συνδέονται μεταξύ τους μέσω των παρακάτω συσχετίσεων:

- **performs:** συνδέει τις οντότητες **agency** και **routes**. Μία εταιρία μπορεί να εκτελεί μία ή παραπάνω διαδρομές
- **is_favorite:** συνδέει τις οντότητες **favorite_routes** και **routes**. Κάθε «αγαπημένη» διαδρομή αντιστοιχεί σε μία πραγματική διαδρομή στην οντότητα routes.

- **contains:** συνδέει τις οντότητες **routes** και **trips**. Κάθε διαδρομή μπορεί να περιέχει ένα ή περισσότερα δρομολόγια.
- **participates:** συνδέει τις οντότητες **trips** και **stop_times**. Επειδή ουσιαστικά η οντότητα stop_times υλοποιεί τον πίνακα δρομολογίων, λέμε ότι ένα δρομολόγιο συμμετέχει στον πίνακα δρομολογίων μία ή περισσότερες φορές.
- **participates:** συνδέει τις οντότητες stops και stop_times. Αντίστοιχα με προηγουμένως μία στάση μπορεί να συμμετέχει στον πίνακα δρομολογίων μία ή περισσότερες φορές
- **trips:** συνδέει τις οντότητες **trips** και **frequencies**. Κάθε δρομολόγιο μπορεί να διαθέτει μία εγγραφή που αφορά τη συχνότητά του.

6.3.3 Διάγραμμα Οντοτήτων-Συσχετίσεων



Εικόνα 6.3: Διάγραμμα οντοτήτων συσχετίσεων

Στο κεφάλαιο αυτό αναλύσαμε της σχεδίαση του συστήματος σε ένα μακροσκοπικό επίπεδο, διακρίνοντας τα διάφορα κομμάτια στα οποία θα χωρίσουμε το σύστημα μας. Στην ενότητα που ακολουθεί θα δούμε πως υλοποιείται αυτή η σχεδίαση σε μια εφαρμογή Android που είναι και ο τελικός σκοπός της εργασίας. Θα αναλύσουμε, με όσο το δυνατό πιο κατανοητό τρόπο τις βασικές κλάσεις της εφαρμογής, τα απαραίτητα αρχεία που περιλαμβάνει, αλλά και την τοπική βάση δεδομένων.

7

Υλοποίηση

Στα προηγούμενα κεφάλαια έγινε μια παρουσίαση του στόχου της εφαρμογής, αναλύθηκαν οι τεχνολογίες στις οποίες βασίστηκε, έγινε μια ανάλυση των απαιτήσεων πάνω στις οποίες στηρίχθηκε η υλοποίησή της, ενώ παρουσιάστηκε και η σχεδίαση του συστήματος σε αφηρημένο επίπεδο αρχιτεκτονικής. Σε αυτό το κεφάλαιο, θα παρουσιαστεί αναλυτικά ο τρόπος υλοποίησης της εφαρμογής σε επίπεδο κώδικα, αναλύοντας τις κλάσεις, τα ειδικά αρχεία, αλλά και δίνοντας το τελικό σχήμα της τοπικής βάσης δεδομένων. Επιπλέον ανάλυση συγκεκριμένων τεχνικών υλοποίησης θα γίνει στο επόμενο κεφάλαιο.

7.1 Αρχιτεκτονική

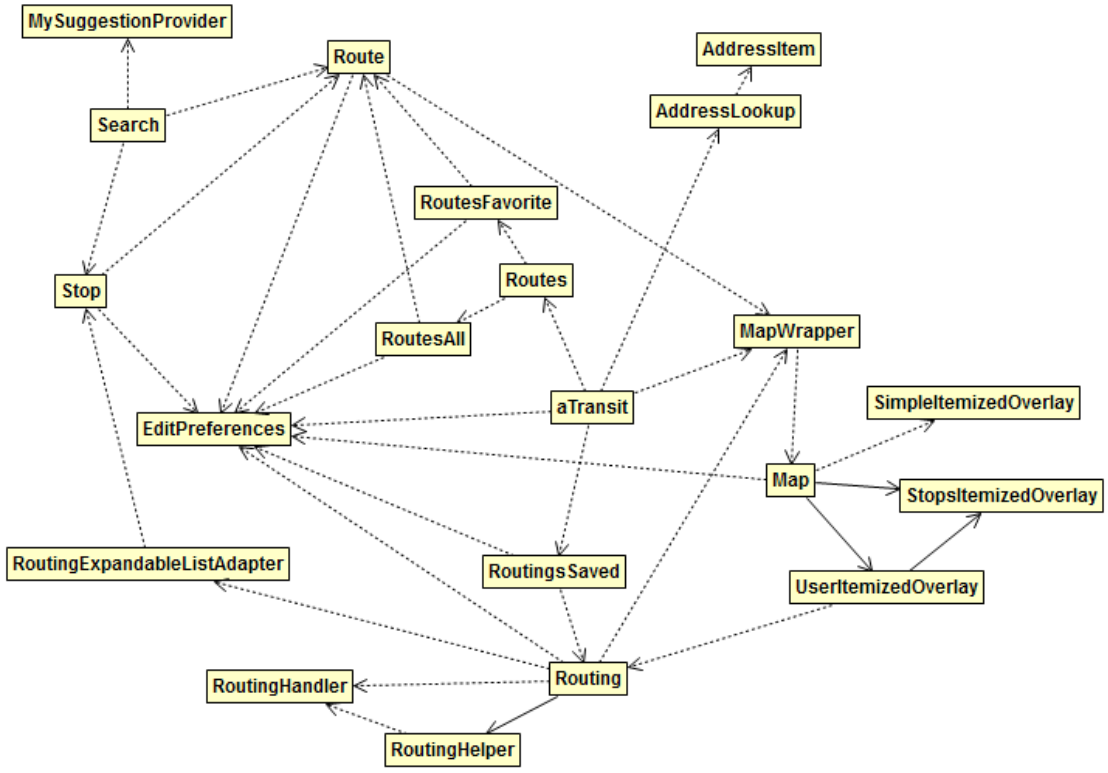
Ο προγραμματισμός για το Android OS γίνεται συνήθως, όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, στη γλώσσα JAVA, εκτός αν κανείς θέλει να έρθει πιο κοντά στις εγγενείς λειτουργίες του συστήματος οπότε μπορεί να χρησιμοποιήσει C. Σε αυτή την εφαρμογή δε χρειάστηκε κάτι τέτοιο, οπότε έχει γραφτεί εξολοκλήρου σε JAVA.

Στο **Διάγραμμα 7.1** φαίνεται το διάγραμμα κλάσεων της εφαρμογής. Για να γίνει καλύτερα ευανάγνωστο έχουν παραληφθεί μερικές βοηθητικές κλάσεις, αλλά και η πολύ σημαντική κλάση DbAdapter για λόγους που θα εξηγηθούν παρακάτω.

Στο σχήμα μπορεί κανείς να διακρίνει την άμεση σύνδεση των κλάσεων με τα υποσυστήματα που περιγράφηκαν στο προηγούμενο κεφάλαιο κατά τη σχεδίαση του συστήματος. Στο κέντρο περίπου του διαγράμματος βρίσκεται η κλάση aTransit που υλοποιεί το υποσύστημα **Main**. Στην δεξιά πλευρά του διακρίνονται οι κλάσεις που συνθέτουν το υποσύστημα χαρτών, ενώ στο κάτω μέρος του βρίσκονται οι κλάσεις που υλοποιούν το υποσύστημα

δρομολόγησης (**Routing**). Τέλος στο πάνω αριστερά κομμάτι του διαγράμματος διακρίνουμε τα υποσυστήματα παρουσίασης διαδρομών και στάσεων (**Routes/Stops**), αλλά και το υποσύστημα αναζήτησης (**Search**).

Ένα πολύ σημαντικό κομμάτι της εφαρμογής λείπει από αυτό το σχήμα και αυτό έγινε γιατί συνδέεται σχεδόν με κάθε άλλη κλάση του συστήματος δημιουργώντας έτσι δυσχέρεια στην ανάγνωσή του. Αυτό είναι η κλάση DbAdapter που είναι η κύρια κλάση του υποσυστήματος που συνδιαλέγεται με τη βάση δεδομένων της εφαρμογής (**Data Manager**).



Διάγραμμα 7.1: Διάγραμμα κλάσεων aTransit

Στις επόμενες παραγράφους θα δούμε πως υλοποιούνται αυτές οι κλάσεις. Επιπλέον θα δούμε τις βασικές κλάσεις του Android API και των άλλων βιβλιοθηκών που χρησιμοποιούνται.

7.2 Περιγραφή Κλάσεων

Σε αυτή την παράγραφο θα παρουσιαστούν οι κλάσεις από τις οποίες αποτελείται η εφαρμογή που χτίστηκε με βάση τις προδιαγραφές του προηγούμενου κεφαλαίου. Θα παρουσιαστούν όλες οι κλάσεις που παράχθηκαν για χάρη αυτής της διπλωματικής καθώς και οι βασικές τους μέθοδοι. Για καλύτερη κατανόηση της υλοποίησης θα παρουσιαστούν ξεχωριστά κάποιες βασικές κλάσεις που επεκτείνονται επανειλημμένα και ανήκουν στις βιβλιοθήκες που χρησιμοποιήθηκαν. Στη συνέχεια θα παρουσιαστούν οι κλάσεις που γράφτηκαν αποκλειστικά για την εφαρμογή χωρισμένες με βάση το αν δημιουργούν γραφικό περιβάλλον ή όχι.

7.2.1 Βασικές κλάσεις του Android API

Μπορεί η γλώσσα της εφαρμογής να είναι η JAVA, όμως αφορά μια εφαρμογή Android και αυτό σημαίνει ότι χρησιμοποιήθηκαν αρκετές κλάσεις προδιαγεγραμμένες από τις βιβλιοθήκες που συμπεριλαμβάνονται στο Android SDK και αποτελούν γενικά το Android API. Αυτές οι κλάσεις χρησιμοποιούνται κατά κόρον από την εφαρμογή συνήθως επεκτείνοντας τες και άλλες φορές παραλαμβάνοντας ένα στιγμιότυπό τους από το σύστημα και χρησιμοποιώντας τις μεθόδους τους.

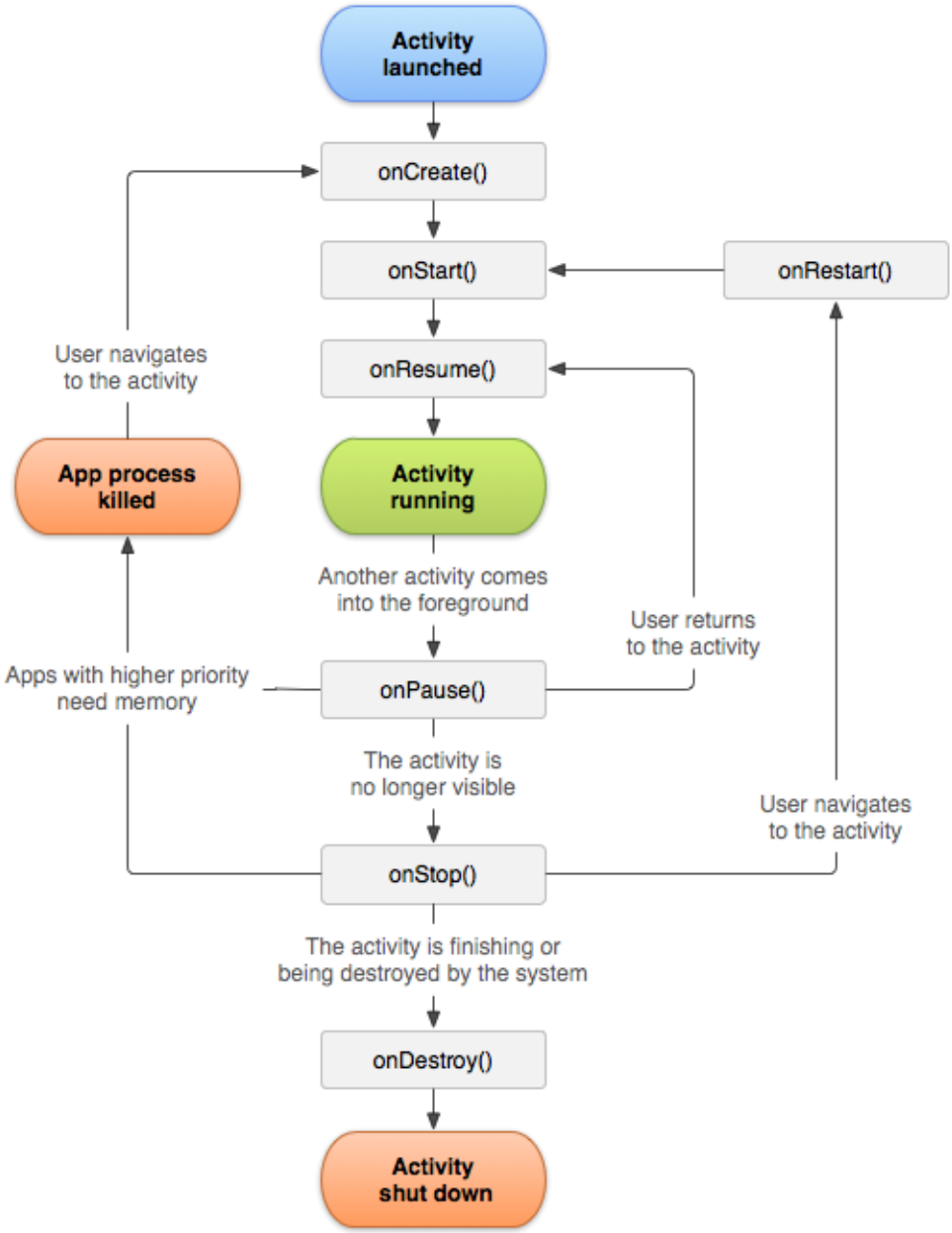
7.2.1.1 *Activity*

Ένα Activity είναι μια κλάση που παρέχει στην εφαρμογή μία οθόνη με την οποία ο χρήστης μπορεί να αλληλεπιδράσει για να επιτύχει κάτι, όπως για παράδειγμα να καλέσει ένα τηλέφωνο, να τραβήξει μία φωτογραφία, να στείλει ένα μήνυμα ή να δει ένα χάρτη. Κάθε activity παραλαμβάνει ένα παράθυρο στο οποίο μπορεί να σχεδιάσει το user interface του. Το παράθυρο αυτό συνήθως καλύπτει όλη την οθόνη, αλλά μπορεί να καταλαμβάνει και μέρος αυτής [20]. Εν συντομία μπορούμε να πούμε ότι όλες οι οθόνες που δείχνουν κάτι ανήκουν σε ένα Activity. Αυτό μπορεί να μην είναι απόλυτα σωστό, αλλά ισχύει στη γενική περίπτωση και σίγουρα ισχύει στην παρούσα εφαρμογή.

Κάθε activity μπορεί να εκκινήσει άλλα activities. Αυτό τότε θα σταματήσει προσωρινά και στο προσκήνιο θα έρθει το επιθυμητό activity. Το προηγούμενο όμως δεν διαγράφεται, αλλά παραμένει σε μία στοίβα, που στο Android είναι γνωστή ως “back stack”. Όταν ο χρήστης θέλει να επιστρέψει στο προηγούμενο activity, απλά πατάει το πλήκτρο «πίσω» της συσκευής του και το προηγούμενο activity επανέρχεται στο προσκήνιο και συνεχίζει τη λειτουργία του.

Κάθε activity έχει τρεις βασικές καταστάσεις με κριτήριο το αν βρίσκεται στο προσκήνιο ή το πόσο βαθιά στο παρασκήνιο βρίσκεται. Αυτές είναι οι **active**, **paused** και **stopped**. Ανάλογα με την κατάσταση του συνήθως καθορίζονται και οι πόροι που καταναλώνει. Και στις τρεις αυτές καταστάσεις η εφαρμογή παραμένει στο back stack και διατηρεί τους πόρους της. Ένα activity τερματίζεται είτε ηθελημένα από τον προγραμματιστή ή από το λειτουργικό σε περίπτωση που χρειαστεί πόρους.

Στην **Εικόνα 7.1** φαίνεται ο κύκλος ζωής μιας activity. Κάθε εναλλαγή κατάστασης σημαίνει και την κλήση μιας μεθόδου callback. Αυτές οι μέθοδοι συνήθως υλοποιούνται από τους προγραμματιστές προκειμένου να συμβεί αυτό που χρειάζεται σε κάθε περίπτωση. Για παράδειγμα μία καλή τακτική είναι να δεσμεύονται οι απαραίτητοι πόροι στην onResume() και να παραδίδονται στην onPause().



Εικόνα 7.1: Κύκλος ζωής ενός Activity

Κάθε κλάση που επεκτείνει την κλάση Activity κληρονομεί και τις μεθόδους τις και διατηρεί τον παραπάνω κύκλο ζωής. Υπάρχουν όμως και ειδικότερες περιπτώσεις activity. Ουσιαστικά πρόκειται για επεκτάσεις της κλάσης Activity που προϋπάρχουν στο Android API προκειμένου να βοηθήσουν στο γρήγορο χτίσιμο τύπων activity που χρησιμοποιούνται συχνά. Μερικές τέτοιες είναι οι **ListActivity**, η **TabActivity** και άλλες. Αυτές με της σειρά τους μπορούν να επεκταθούν από τον προγραμματιστεί για να επιτύχει το επιθυμητό αποτέλεσμα.

7.2.1.2 *View*

Η κλάση View αποτελεί το ελάχιστο δομικό στοιχείο για να δημιουργηθούν γραφικά στην οθόνη της συσκευής. Κάθε στιγμιότυπο της κλάσης μπορεί να «ζωγραφιστεί» πάνω στην οθόνη ενός Activity και να καλέσει μία σειρά από μεθόδους ανάλογα με την αλληλεπίδραση του χρήστη με την οθόνη και τις άλλες εισόδους της συσκευής. Μερικά παραδείγματα υλοποιήσεών της είναι τα κουμπιά, οι εικόνες, τα πεδία εισόδου κειμένου κ.α.

7.2.1.3 *SQLiteOpenHelper*

Πρόκειται για μία βοηθητική abstract κλάση που χρησιμοποιείται για την δημιουργία μιας βάσης δεδομένων, την επικοινωνία με αυτή και της διαχείριση των δεδομένων και των εκδόσεων της. Μέσω τον μεθόδων της επιτρέπει τη έναρξη επικοινωνίας με τη βάση, τη δημιουργία της, αν δεν υπάρχει, και την αναβάθμιση της, αν χρειάζεται.

Όπως θα δούμε παρακάτω αυτή η κλάση υλοποιείται από μια εσωτερική κλάση του Database Manager για να εκτελέσει όλες αυτές τις λειτουργίες που προαναφέρθηκαν.

7.2.2 Βασικές κλάσεις της βιβλιοθήκης mapsforge

Όπως το σύνολο βιβλιοθηκών του Android API μας βοηθάει σε ότι αφορά το χτίσιμο user interfaces, στην δημιουργία και επικοινωνία με τη βάση δεδομένων κλπ., έτσι και η βιβλιοθήκη mapsforge μας βοηθάει στην δημιουργία χαρτών, την παράθεση δεδομένων πάνω σε αυτούς και την αλληλεπίδρασή τους με το χρήστη. Οι πιο βασικές από τις κλάσεις που χρησιμοποιούμε περιγράφονται στις επόμενες παραγράφους.

7.2.2.1 *MapActivity*

Η **MapActivity** είναι μία κλάση που πρέπει να επεκταθεί προκειμένου να εμφανιστούν στην οθόνη της συσκευής οι χάρτες που χτίζονται με τη βιβλιοθήκη mapsforge. Διαθέτει όλες τις μεθόδους μιας κλασικής activity και προσθέτει τη δυνατότητα δημιουργίας μιας **MapView** που ουσιαστικά φιλοξενεί το χάρτη [21].

7.2.2.2 *MapView*

Η κλάση **MapView** δείχνει ένα χάρτη στην οθόνη της συσκευής και διαχειρίζεται όλα τα γεγονότα και τις εισόδους που έχουν να κάνουν με το χάρτη, όπως π.χ. κινήσεις τον δακτύλων για μετακίνηση ή zoom in/out στο χάρτη. Διαθέτει επίσης μία κλίμακα μεγέθους και κουμπιά για έλεγχο του zoom. Μπορεί να μας δώσει και ένα στιγμιότυπο μιας άλλης κλάσης, της **MapController**, που μας δίνει ένα παραπάνω έλεγχο πάνω στο χάρτη [22].

Η ίδια κλάση χρησιμοποιείται είτε μας αφορά offline είτε online rendering του χάρτη. Ο τύπος αυτός του χάρτη καθορίζεται εσωτερικά, ενώ καθορίζεται και το αρχείο που πρέπει να χρησιμοποιηθεί, αν πρόκειται για offline χρήση.

7.2.2.3 *Overlay*

Ένα Overlay είναι μία «επίστρωση» πάνω στο χάρτη που μας επιτρέπει να προσθέσουμε διάφορα αντικείμενα, συνήθως **OverlayItems** και **OverlayWays**.Με αυτό τον τρόπο μπορούμε να δείξουμε συγκεκριμένα σημεία πάνω στο χάρτη (π.χ. καφετέριες ή στάσεις λεωφορείου) και να σχεδιάσουμε γραμμές (π.χ. τα όρια ενός δήμου ή τη διαδρομή ενός τρένου).

7.2.3 Κλάσεις εφαρμογής

Ο κώδικας της εφαρμογής για να γίνεται πιο εύκολα κατανοητός χωρίστηκε σε **κλάσεις με γραφικό περιβάλλον** και **κλάσεις χωρίς γραφικό περιβάλλον**. Οι πρώτες βασίζονται κυρίως στις κλάσεις του Android API και του Mapsforge που αναφέρθηκαν παραπάνω για να σχεδιάσουν τα γραφικά της οθόνες μέσω activities, views, overlays κ.α. Στη γενική περίπτωση κάθε οθόνη που βλέπει ο χρήστης στην οθόνη του είναι μία κλάση, αλλά αυτό δεν αποτελεί κανόνα. Οι κλάσεις της δεύτερης κατηγορίας αναλαμβάνουν επιτελικές λειτουργίες της εφαρμογής όπως επικοινωνία με τη βάση δεδομένων, ανάθεση τιμών σε views του γραφικού περιβάλλοντος κλπ.

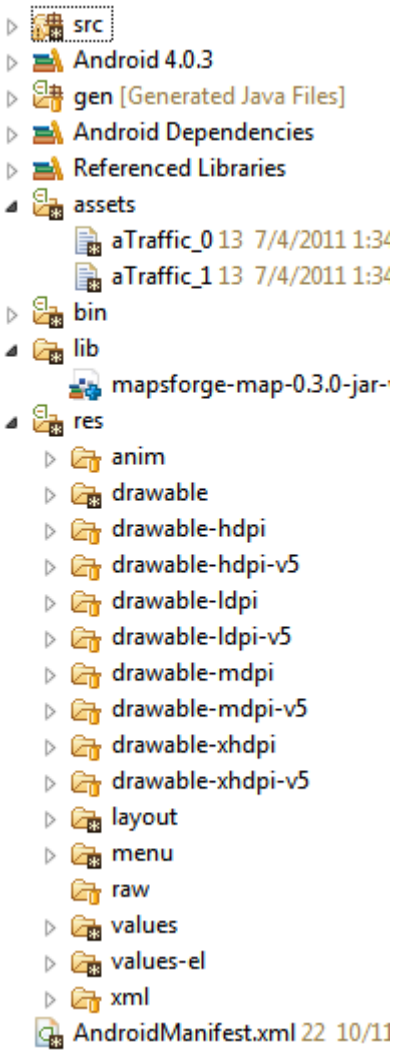
Για λόγους διατήρησης τη ροής του κειμένου η ανάλυση όλων των κλάσεων της εφαρμογής γίνεται στο **κεφάλαιο 12 (παράρτημα)**.

7.3 Κωδικοποίηση αρχείων

Εκτός από τις κλάσεις της εφαρμογής, μία εφαρμογή Android γενικά περιλαμβάνει και άλλα αρχεία τα οποία είναι πολύ βασικά για τη διαδικασία του προγραμματισμού και για την τελική εικόνα της εφαρμογής. Αυτά τα αρχεία γενικά περιλαμβάνουν εικόνες που χρησιμοποιούνται στα διάφορα γραφικά στοιχεία της εφαρμογής, εξωτερικές βιβλιοθήκες, αρκετά αρχεία XML και διάφορα άλλα αρχεία που μπορεί να χρειαστεί να ενσωματωθούν στην εφαρμογή.

Το σύστημα αρχείων μιας Android εφαρμογής περιλαμβάνει κάποιους βασικού καταλόγους όπου αποθηκεύονται τα ανάλογα αρχεία. Στην Εικόνα 7.2φαίνεται η ιεραρχία των αρχείων της εφαρμογής aTransit που σχεδιάστηκε για την παρούσα εφαρμογή. Θα παρουσιάσουμε τους πιο σημαντικούς καταλόγους:

- **src**: περιέχει όλα τα αρχεία κώδικα (τις κλάσεις που περιγράφηκαν παραπάνω)
- **assets**: εδώ περιέχονται τα αρχεία που έρχονται με την εφαρμογή. Στη συγκεκριμένη περίπτωση πρόκειται για την προκατασκευασμένη βάση δεδομένων που αφορά τα δεδομένα GTFS και χωρίζεται σε 2 αρχεία γιατί λόγω περιορισμών πρέπει κάθε αρχείο να είναι μικρότερο από 1MByte.
- **lib**: αυτός ο κατάλογος προστέθηκε από εμάς και περιέχει την εξωτερική βιβλιοθήκη που είναι απαραίτητη για τη λειτουργία των χαρτών της εφαρμογής.
- **res**: περιλαμβάνει τα διάφορα resources της εφαρμογής (βασικά εικόνες και αρχεία XML)



Εικόνα 7.2: Ιεραρχία αρχείων

Θα αναλύσουμε σε περισσότερο βάθος τα περιεχόμενα του φακέλου **res** γιατί περιέχει τα πιο άξια αναφοράς αρχεία, τα οποία ακολουθούν και μια συγκεκριμένη κωδικοποίηση.

Όλοι οι φάκελοι **drawable** περιέχουν εικόνες που χρησιμοποιούνται από την εφαρμογή. Υπάρχουν διαφορετικοί φάκελοι για κάθε βασική ανάλυση οθόνης προκειμένου να χρησιμοποιήσουμε τα σωστά assets ανάλογα με το μέγεθος. Αντίστοιχα στο φάκελο **anim** περιέχονται αρχεία xml που περιγράφουν animations εικόνων μέσω αρχείων XML.

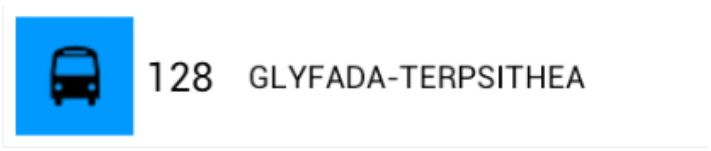
Όπως γίνεται φανερό, τα XML αρχεία παίζουν σημαντικό ρόλο στην υλοποίηση μιας εφαρμογής για το Android. Οι υπόλοιποι φάκελοι περιέχουν γενικά αρχεία XML. Στον κατάλογο **layout** τα αρχεία αυτά περιγράφουν τη διάταξη των διάφορων Views σε ένα Activity. Γενικά ο πιο εύκολος τρόπος να δημιουργήσουμε γραφικό περιβάλλον είναι δηλώνοντας το με τέτοια αρχεία. Ένα παράδειγμα τέτοιου αρχείου φαίνεται στην **Εικόνα 7.3**, ενώ το αποτέλεσμα του στην οθόνη (μετά από προγραμματιστικό «γέμισμα» των στοιχείων του) φαίνεται στην **Εικόνα 7.4**. Αντίστοιχα αρχεία υπάρχουν στον κατάλογο **menu** για να περιγράψουν τα menu διαφόρων Activities.

Στους καταλόγους **values** και **values-el** υπάρχουν σε μορφή XML όλα τα strings που χρησιμοποιούνται από την εφαρμογή. Κάθε φορά που θέλουμε να εμφανίσουμε ένα κείμενο στην οθόνη χρησιμοποιούμε αναφορά ενός έτοιμου string αντί να το φτιάχνουμε επιτόπου. Αυτός ο τρόπος λειτουργίας έχει δύο σημαντικά πλεονεκτήματα. Το πρώτο είναι ότι τα λάθη διορθώνονται εύκολα και σε περίπτωση που επαναχρησιμοποιούμε strings, χρειάζεται να

διορθωθούν μία φορά. Το δεύτερο λύνει τα χέρια του προγραμματιστή σε ότι αφορά της μετάφραση της εφαρμογής του. Όπως αναφέραμε, στην εφαρμογή μας διαθέτουμε δύο καταλόγους. Ο κατάλογος values περιέχει τα strings στα Αγγλικά, ενώ ο κατάλογος values-el περιέχει τα strings με τα ίδια ονόματα στα Ελληνικά. Με αυτό τον τρόπο αναφέροντας απλά το όνομα του string, το σύστημα ταιριάζει το κατάλληλο string ανάλογα με τη γλώσσα συστήματος. Ελληνικά αν είναι στα Ελληνικά και Αγγλικά αν είναι σε οτιδήποτε άλλο. Η μετάφραση γίνεται εύκολη αφού όταν έχουμε τα strings σε κάποια άλλη γλώσσα απλά δημιουργούμε ένα ακόμα κατάλογο για αυτή.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/route"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:fadingEdge="horizontal"
    android:gravity="center_vertical"
    android:padding="5dp" >
    <ImageView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/route_type"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:padding="10dp" />
    <TextView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/route_short_name"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:gravity="center_vertical"
        android:padding="5dp"
        android:textColor="@color/text"
        android:textSize="18sp" />
    <TextView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/route_long_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center_vertical"
        android:padding="10dp"
        android:textColor="@color/text"
        android:textSize="14sp" />
</LinearLayout>
```

Εικόνα 7.3: Αρχείο XML που ορίζει γραφικό περιβάλλον



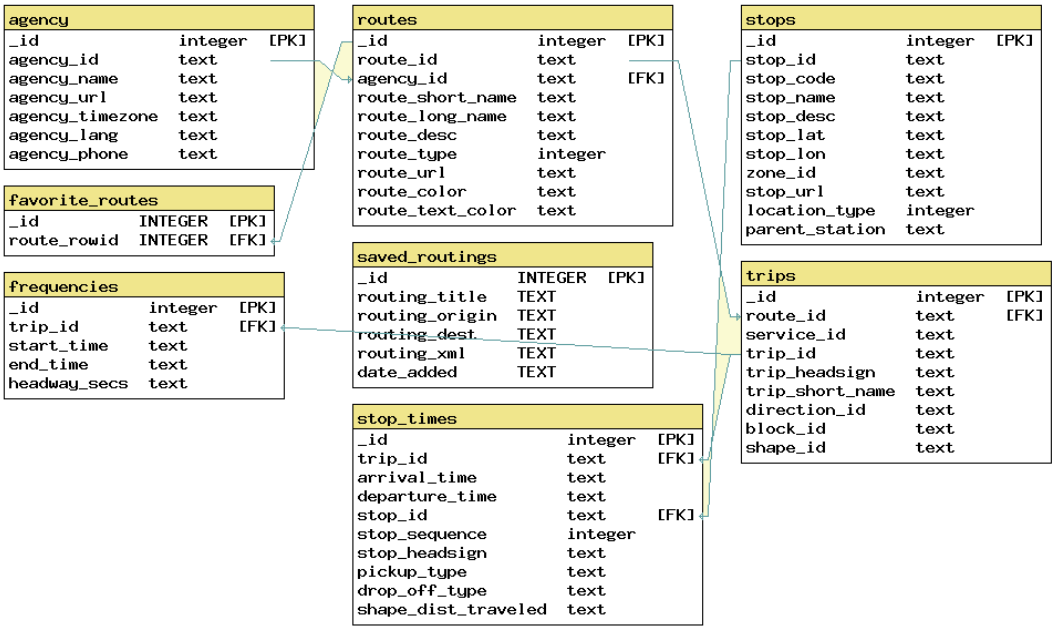
Εικόνα 7.4: Γραφικό για μία διαδρομή, χτισμένο από XML στην Εικόνα 7.3

Με παρόμοιο τρόπο περιγράφονται και οι ρυθμίσεις της εφαρμογής από ένα αρχείο που βρίσκεται στον κατάλογο **xml**. Το κτίσιμο του activity ρυθμίσεων της εφαρμογής γίνεται πολύ απλά με την κλήση μιας μεθόδου που το κτίζει διαβάζοντας το συγκεκριμένο αρχείο.

Όπως φαίνεται και στην Εικόνα 7.2, στη ρίζα της εφαρμογής βρίσκεται το αρχείο **AndroidManifest.xml**. Αυτό είναι το βασικότερο αρχείο για τη σωστή λειτουργία της εφαρμογής αφού σε αυτό αναφέρονται όλες οι λεπτομέρειες υλοποίησής της προς το σύστημα. Εδώ δηλώνονται όλα τα Activities και συγκεκριμένες συμπεριφορές αυτών. Σε αυτό το αρχείο επίσης, δηλώνονται οι άδειες που χρειάζονται από το χρήστη (π.χ. η χρήση internet, ο εντοπισμός θέσης και η πρόσβαση σε αρχεία της εξωτερικής μνήμης). Τέλος εδώ δηλώνεται η έκδοση Android για την οποία έχει κτιστεί η εφαρμογή, αλλά και η μικρότερη έκδοση στην οποία έχει ελεγχθεί ότι λειτουργεί σωστά.

7.4 Βάση Δεδομένων

Στην υλοποίηση του συστήματος περιλαμβάνεται και η τοπική βάση δεδομένων. Στην **Εικόνα 7.5** παρουσιάζεται το σχήμα της βάσης δεδομένων όπως προκύπτει από το μοντέλο οντοτήτων συσχετίσεων που προδιαγράφηκε στην **παράγραφο 6.3**. Στην επόμενη ενότητα θα αναλυθεί και η διαδικασία που ακολουθήθηκε για την κατασκευή της.



Εικόνα 7.5: Διάγραμμα της βάσης δεδομένων

Σε αυτό το κεφάλαιο παρουσιάσαμε, αρκετά αναλυτικά, τον τρόπο με τον οποίο υλοποιήθηκε η σχεδίαση του συστήματος σε επίπεδο κώδικα και αρχείων και είδαμε το τελικό σχήμα της βάσης δεδομένων που συμπεριλαμβάνεται στο πακέτο της εφαρμογής. Είδαμε τις ιδιαιτερότητες του Android κατά τον προγραμματισμό για αυτό, πολλές από τις οποίες είναι βοηθητικές για τον προγραμματιστή. Για παράδειγμα η προτροπή της χρήσης xml αρχείων για τον ορισμό τον strings, τελικά βοηθά πολύ στην μετάφραση της εφαρμογής. Ακόμη είδαμε σύντομα κάποιες βασικές κλάσεις της βιβλιοθήκης mapsforgeπου μας βοήθησε σημαντικά στην υλοποίηση του υποσυστήματος χαρτών για offline χρήση.

Στην επόμενη ενότητα θα δούμε αναλυτικότερα μερικά σημεία της ανάπτυξης που είναι άξια ειδικής αναφοράς. Θα παρουσιάσουμε τον τρόπο επικοινωνίας με το OpenTripPlanner, και θα δούμε τον τρόπο δημιουργίας του offline χάρτη, αλλά και της τοπικής βάσης δεδομένων.

8

Τεχνικές λεπτομέρειες υλοποίησης

Στα προηγούμενα κεφάλαια παρουσιάστηκε εκτενώς η σχεδίαση του συστήματος και η υλοποίηση της εφαρμογής με ένα σχετικά μακροσκοπικό τρόπο. Σε αυτό το κεφάλαιο θα συζητηθεί η υλοποίηση τους σε πιο μικροσκοπικό επίπεδο. Θα παρουσιαστούν, αρχικά, κάποια σημεία της υλοποίησης που χρήζουν ιδιαίτερης προσοχής και στη συνέχεια τα εργαλεία που χρησιμοποιήθηκαν κατά την υλοποίηση του συστήματος καθώς και ο τρόπος εγκατάστασης τους και εγκατάστασης της εφαρμογής.

8.1 Λεπτομέρειες υλοποίησης

Όπως αναφέρθηκε παραπάνω, μερικά κομμάτια του συστήματος, αξίζουν ιδιαίτερη αναφορά προκειμένου να γίνει κατανοητός ο τρόπος υλοποίησής τους καθώς και οι λόγοι για τους οποίους ακολουθήθηκαν συγκεκριμένες διαδικασίες. Σκοπός αυτής της παραγράφου είναι αυτή ακριβώς η ανάλυση των ιδιαίτερων πτυχών του συστήματος.

Στις επόμενες παραγράφους θα περιγραφούν τα κομμάτια της διπλωματικής που χρήζουν περαιτέρω εξήγησης και ανάλυσης. Αυτά είναι:

1. Ο τρόπος επικοινωνίας της εφαρμογής με τον server μέσω του OpenTripPlanner API
2. Ο τρόπος με τον οποίο δημιουργήθηκε ο χάρτης που χρησιμοποιείται για το offline rendering
3. Ο τρόπος με τον οποίο προέκυψε η τοπική βάση δεδομένων από τα αρχεία GTFS

8.1.1 OpenTripPlanner API

Όπως έχει αναφερθεί αρκετές φορές στα προηγούμενα κεφάλαια, ο αλγόριθμος δρομολόγησης που χρησιμοποιείται για την εύρεση της βέλτιστης διαδρομής βρίσκεται σε

απομακρυσμένο server και βασίζεται στο OpenTripPlanner. Είναι λοιπόν πολύ χρήσιμο να παρουσιαστούν περισσότερες λεπτομέρειες σχετικά με το API του OTP και αυτός είναι ο σκοπός αυτής της παραγράφου.

8.1.1.1 *Request*

Για να ξεκινήσει η διαδικασία εύρεσης της βέλτιστης διαδρομής είναι απαραίτητο να γίνει ένα αίτημα (request) στον server. Ένα παράδειγμα τυπικού Request στο OpenTripPlanner για την εύρεση βέλτιστης διαδρομής είναι το παρακάτω:

http://geolinux.imis.athena-innovation.gr/opentripplanner-api-webapp/ws/**plan**?_dc=1273478553238&**fromPlace**=37.98236388851004,23.720786873046904&**toPlace**=37.97343331182684,23.776748481933623&**arriveBy**=false&**date**=05/25/2011&**time**=6:19pm&**optimize**=QUICK&**maxWalkDistance**=160&**mode**=TRANSIT,WALK,WALK&**wheelchair**=false&**toCoord**=37.97343331182684,23.776748481933623&**fromCoord**=37.982363888510044,23.720786873046904&**intermediatePlaces**=

Εικόνα 8.1: Τυπικό Request στο OTP

Στο παραπάνω request φαίνονται με bold οι βασικές μεταβλητές οι οποίες χρησιμοποιούνται και στην παρούσα εφαρμογή. Το "**plan**" αφορά τον τύπο του request και δείχνει στο server ότι ζητάμε ένα πλάνο για την βέλτιστη δρομολόγηση. Από εκεί και μετά ακολουθεί μια σειρά από μεταβλητές οι οποίες στην εφαρμογή θέτονται από το χρήστη είτε από την αρχική οθόνη είτε μέσα από τις ρυθμίσεις. Στον **Πίνακας 8.1** αναλύονται οι διάφορες μεταβλητές [23].

8.1.1.2 *Response*

Η φυσική συνέχεια του request στον OTP server, με την προϋπόθεση φυσικά ότι στον client υπάρχει σύνδεση στο διαδίκτυο και ο server είναι προσπελάσιμος, είναι ένα response με τις απαραίτητες πληροφορίες για τη βέλτιστη διαδρομή σύμφωνα με την είσοδο. Στην Εικόνα 8.2 φαίνεται ένα σημαντικό κομμάτι ενός τυπικού response. Λόγω συντομίας αφαιρέθηκαν κομμάτια που αφορούν γενικά το ταξίδι, όπως διάρκεια και απόσταση συνολικού ταξιδιού, αφετηρία και προορισμός, η είσοδος στην οποία απάντησε ο server κ.α. Θα επικεντρωθούμε όμως στο σημαντικότερο σημείο που είναι η ανάλυση ενός υποταξιδιού.

Όνομα	Περιγραφή	Default τιμή
fromPlace	Η αφετηρία. Είναι είτε ένα ζευγάρι γεωγραφικού πλάτους/μήκους, είτε ένα Vertex label.	
toPlace	Ο προορισμός (ομοίως με fromPlace)	
intermediatePlaces	Μία μη ταξινομημένη λίστα με τοποθεσίες που επιθυμεί να επισκεφτεί ο χρήστης (ομοίως με fromPlace)	
date	Η ημερομηνία που πρέπει να ξεκινήσει το ταξίδι (ή που θα τελειώσει, αν το arriveBy είναι true).	
time	Η ώρα που πρέπει να ξεκινήσει το ταξίδι (ή που θα τελειώσει, αν το arriveBy είναι true).	
arriveBy	Αν το ταξίδι πρέπει να ξεκινήσει ή να τελειώσει την καθορισμένη ώρα και ημερομηνία	false
wheelchair	Αν θα πρέπει το ταξίδι να υποστηρίζει τη χρήση αναπηρικής καρέκλας	false
maxWalkDistance	Η μεγαλύτερη απόσταση που ο χρήστης δύναται να περπατήσει.	800
optimize	Ένα σετ με χαρακτηριστικά με τα οποία ο χρήστης επιθυμεί να βελτιστοποιήσει. Μπορεί να είναι κάτι μεταξύ των QUICK, TRANSFERS ή SAFE (γρηγορότερο ταξίδι, λιγότερες μετεπιβιβάσεις ή ασφαλέστερο ταξίδι).	QUICK
mode	Ένα σετ με τρόπους που ο επιβάτης επιθυμεί να ταξιδέψει.	TRANSIT,WALK

Πίνακας 8.1: Περιγραφή μεταβλητών OTP request

Στο παρακάτω κομμάτι του response (Εικόνα 8.2) μπορεί κανείς να παρατηρήσει τον τρόπο με τον οποίο περιγράφεται ένα ταξίδι, ενώ διακρίνεται αναλυτικά η περιγραφή ενός υποταξιδιού. Κάθε προτεινόμενο ταξίδι ανήκει σε ένα element με όνομα "**legs**". Αυτό περιλαμβάνει έναν αριθμό από elements ονόματι "**leg**" που καθένα περιγράφει ένα υποταξίδι. Στα attributes του element μπορεί κανείς να διακρίνει τα μετά-δεδομένα του που αναφέρουν τον τρόπο της μετακίνησης (**mode**) και άλλα, ανάλογα με τον τύπο. Αν πρόκειται για περπάτημα αναφέρεται μόνο ο δρόμος (**route**), ενώ αν πρόκειται για μέσο μεταφοράς αναφέρεται ο τίτλος της διαδρομής (**route**) και η εταιρία που την εκτελεί (**agencyId**).

```
<response>
<plan>
……
<legs>
   <leg route="Keramikou" mode="WALK">
      …
   </leg>
   <leg agencyId="AMEL" route="M2" mode="SUBWAY">
      <duration>225000</duration>
      <startTime>2011-05-25T18:25:48+03:00</startTime>
      <endTime>2011-05-25T18:29:33+03:00</endTime>
      <distance>1219.3192220545288</distance>
      <alternateRoutes/>
      <from>
         <name>METAXOURGIO METRO STATION</name>
         <stopId>ST061552</stopId>
         <lon>23.721069</lon>
         <lat>37.985794</lat>
         <geometry>
            {"type": "Point", "coordinates": [23.721069,37.985794]}
         </geometry>
      </from>
      <to>
         <name>PANEPISTIMIO METRO STATION</name>
         <stopId>ST061559</stopId>
         <lon>23.732795</lon>
         <lat>37.980512</lat>
         <geometry>
            {"type": "Point", "coordinates": [23.732795,37.980512]}
         </geometry>
      </to>
      <intermediateStops>
         <stop>
            <name>OMONIA METRO STATION</name>
            <stopId>ST062251</stopId>
            <lon>23.728113</lon>
            <lat>37.984134</lat>
            <geometry>
               {"type": "Point", "coordinates": [23.728113,37.984134]}
            </geometry>
         </stop>
      </intermediateStops>
      <legGeometry>
         <length>4</length>
         <points>ebzfFs_xoCjIak@??rUg\</points>
      </legGeometry>
   </leg>
   <leg route="street transit link" mode="WALK">
      …
   </leg>
   …
</legs>
……
</plan>
……
</response>
```

Εικόνα 8.2: Ένα κομμάτι ενός τυπικού response του OTP

Σε κάθε υποταξίδι (leg) περιλαμβάνεται η διάρκειά του (**duration**), η ώρα εκκίνησης και τερματισμού του (**startTime**, **endTime**), αλλά και η απόσταση που καλύπτει. Στη συνέχεια αναφέρονται αναλυτικά το σημείο έναρξης και τερματισμού (**from**, **to**) του υποταξιδιού με

τοποθεσία (**lat**, **lon**), ονομασία (**name**), αλλά και id (**stopId**) όταν πρόκειται για στάση μέσων μεταφοράς. Στο element **intermediateStops** περιλαμβάνονται, με την ίδια μορφή όπως στα from και to, οι ενδιάμεσες στάσεις που θα εκτελέσει το μέσο.

Τέλος υπάρχει το **legGeometry** element όπου περιγράφεται η γεωμετρία του υποταξιδιού σε μορφή encoded polyline (**points**). Ουσιαστικά πρόκειται για ένα κωδικοποιημένο string που περιγράφει μία γραμμή πάνω στο χάρτη που δείχνει την πορεία του ταξιδιού. Το element **length** εδώ αναφέρει το πλήθος των διαφορετικών κομματιών που απαρτίζουν τη γραμμή αυτή, συνήθως τις ευθείες μεταξύ δύο στάσεων.

8.1.2 Χρήση Osmosis και Map File Writer για την παραγωγή χάρτη

Προκειμένου να χρησιμοποιήσουμε την βιβλιοθήκη mapsforge, που αναφέρθηκε στην παράγραφο 4.4, χωρίς τη χρήση διαδικτύου, πρέπει πρώτα να προετοιμάσουμε το χάρτη της εφαρμογής μας. Οι developers του mapsforge έχουν ετοιμάσει ήδη κάποια αρχεία για πόλεις της Γερμανίας ή άλλες μεγάλες πόλεις της Ευρώπης, αλλά όχι και για την περιοχή της Αθήνας. Αυτό όμως δεν αποτελεί δυσκολία, γιατί οι ίδιοι έχουν αναπτύξει ένα plug-in για το εργαλείο Osmosis που μπορεί να χρησιμοποιηθεί για την παραγωγή οποιουδήποτε αρχείου χάρτη σε σημαντικά συμπυκνωμένο μέγεθος. Το plug-in αυτό ονομάζεται Map File Writer [24].

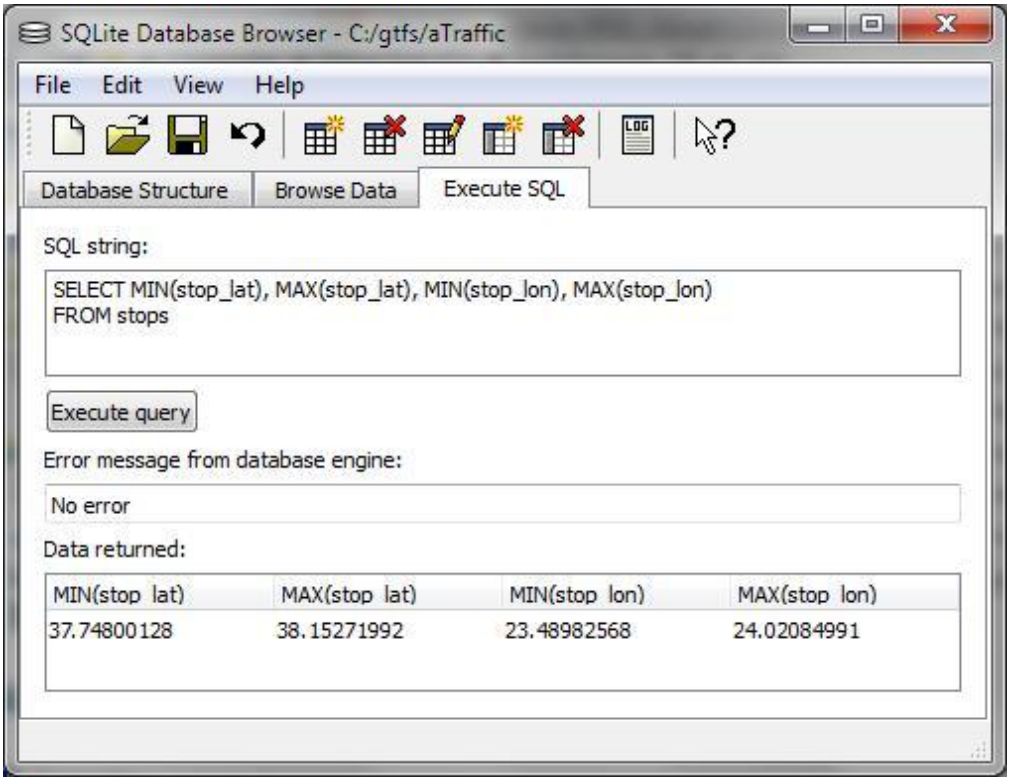
Το Osmosis είναι μία Java εφαρμογή που χρησιμοποιείται για την επεξεργασία δεδομένων του OSM (OpenStreetMap). Το εργαλείο αυτό αποτελείται από μια σειρά από υποσυστήματα που συνεργάζονται μεταξύ τους για να εκτελέσουν μεγαλύτερες λειτουργίες. Έχει γραφτεί με σκοπό να είναι εύκολο να προσθέσει κανείς νέες δυνατότητες χωρίς να χρειάζεται να αλλάξει βασικές λειτουργίες, όπως επεξεργασία αρχείων και βάσεων δεδομένων [17].

Ένα τέτοιο υποσύστημα, το Map File Writer, έχει αναπτυχθεί από την ομάδα του mapsforge και μας επιτρέπει να δημιουργήσουμε ένα συμπυκνωμένου μεγέθους χάρτη όποιας περιοχής θέλουμε, για να το χρησιμοποιήσουμε στη συνέχεια στην εφαρμογή μας.

Η διαδικασία που ακολουθήθηκε για τη δημιουργία του χάρτη της Αθήνας είναι η εξής:

1. **Κατέβασμα του χάρτη της Ελλάδας από το wiki το OSM** [16]. Εκεί μπορεί κανείς να βρει εξαγωγές χαρτών από τον ευρύτερο χάρτη του κόσμου, που ανανεώνονται αρκετά συχνά. Θα μπορούσαμε να κάνουμε βέβαια και δική μας εξαγωγή με χρήση του Osmosis. Το αρχείο που κατεβάστηκε στη δική μας περίπτωση, είχε μέγεθος 45MB και ήταν στη μορφή .pbf.

2. **Εύρεση των ορίων του επιθυμητού χάρτη.** Αυτό έγινε μέσα από τη βάση των δεδομένων που παράχθηκε από τα αρχεία GTFS. Η εντολή που χρησιμοποιήθηκε για αυτό το σκοπό, καθώς και τα αποτελέσματά της, φαίνονται στην παρακάτω εικόνα:



Εικόνα 8.3: Εύρεση ορίων του χάρτη μέσω των στάσεων

3. **Χρήση των παραπάνω ορίων για εξαγωγή του χάρτη της Αθήνας από τον χάρτη της Ελλάδας.** Αυτό γίνεται με χρήση του Osmosis και κάποιων plug-ins που συμπεριλαμβάνονται στη βασική του έκδοση και παράγεται ένα, μικρότερο από το αρχικό, αρχείο .pbf. Η εντολή που χρησιμοποιήθηκε για την εξαγωγή του χάρτη φαίνεται στο παρακάτω πλαίσιο. Το αρχείο που προέκυψε έχει μέγεθος 4.5MB.

```
osmosis --read-pbf file=../maps/greece.osm.pbf --
bounding-box left=23.490 top=38.153 right=24.021
bottom=37.749 --write-pbf file=../maps/athens.osm.pbf
```

4. **Χρήση του Map File Writer για την παραγωγή του αρχείου .map που χρησιμοποιείται από το mapsforge.** Και πάλι με χρήση του Osmosis και του plug-in που αναφέρθηκε παράγουμε το συγκεκριμένο αρχείο χάρτη που γίνεται αποδεκτό από τη βιβλιοθήκη mapsforge προκειμένου να γίνει το offline rendering. Η εντολή φαίνεται στο παρακάτω πλαίσιο. Μέσω αυτής της εντολής καθορίζουμε και την αρχική θέση του χάρτη στο κέντρο της Αθήνας.

```
osmosis --rb file=../maps/athens.osm.pbf --
mapfile-writer file=../maps/athens.map map-start-
position=37.98396,23.728048
```

Μέσα από αυτή τη διαδικασία παράγεται ένα αρχείο athens.map μεγέθους **3MB** που παρέχεται για download μέσα από τις ρυθμίσεις της εφαρμογής και μπορεί να ανανεωθεί, ακολουθώντας την ίδια διαδικασία, όσο συχνά το επιθυμούμε.

8.1.3 Δημιουργία βάσης SQLite από GTFS

Όπως έχει αναφερθεί στα προηγούμενα κεφάλαια, η βάση δεδομένων, στην οποία στηρίζεται η offline λειτουργικότητα της εφαρμογής, προέρχεται από το πρότυπο GTFS. Στην παράγραφο 4.1 έγινε εκτενής αναφορά στο πρότυπο αυτό και τη μορφή των αρχείων που το υλοποιούν, ενώ στην παράγραφο 6.3 προδιαγράφηκε η βάση δεδομένων που χρησιμοποιείται από την εφαρμογή και βασίζεται σε αυτά τα αρχεία. Σε αυτή την παράγραφο θα παρουσιαστεί ο τρόπος με τον οποίο προέκυψε μια βάση δεδομένων SQLite, που είναι αυτή που υποστηρίζει εγγενώς (natively) το Android.

Διατηρώντας την αρχική μορφή των πινάκων, όπως προκύπτουν άμεσα από τα αρχεία GTFS, δεν αποτελεί ιδιαίτερο πρόβλημα το σχήμα της βάσης. Κάθε αρχείο αποτελεί ένα πίνακα και κάθε γραμμή του αρχείου αποτελεί μία εγγραφή, εκτός από την πρώτη που είναι οι τίτλοι των στηλών του πίνακα. Έτσι, η μεταφορά των δεδομένων σε μία βάση δεδομένων δεν αποτελεί ιδιαίτερο πρόβλημα μιας και είναι ουσιαστικά ένα διάβασμα των αρχείων γραμμή-γραμμή και πέρασμα των στοιχείων τους στη βάση. Η εντολή .import της sqlite3 κάνει τα πράγματα ακόμη πιο εύκολα.

Η SQLite μπορεί να υποστηρίζεται εγγενώς από το Android API, έχει όμως κάποιες ιδιαιτερότητες στο σχηματισμό της βάσης προκειμένου να χρησιμοποιηθούν οι βοηθητικές του κλάσεις (όπως List Adapters κ.α.). Μία από αυτές είναι ότι απαιτεί να υπάρχει ένα μοναδικό «κλειδί» για κάθε εγγραφή πίνακα με όνομα πεδίου **_id**. Αυτό το πεδίο δημιουργείται αυτόματα σε πίνακες που κατασκευάζονται από την ίδια την εφαρμογή, αλλά για τη βάση της εφαρμογής αυτής, που κατασκευάζεται εξωτερικά, πρέπει να μεριμνήσουμε ώστε να υπάρχει. Επειδή η εντολή .import δεν μπορεί να παραμετροποιηθεί ώστε να δημιουργήσει αυτό το πεδίο, χρησιμοποιήσαμε ενδιάμεσους πίνακες για την εισαγωγή των στοιχείων και περάσαμε τα στοιχεία του στους τελικούς πίνακες με τη χρήση απλών εντολών **INSERT.**

Στην **Εικόνα 8.4** φαίνεται το πρώτο κομμάτι του script που χρησιμοποιήσαμε. Εδώ γίνεται η δημιουργία των ενδιάμεσων πινάκων της εισαγωγής και στη συνέχεια πραγματοποιείται η εισαγωγή των δεδομένων με χρήστη της εντολής .**import**. Επειδή η εντολή αυτή διαβάζει όλες τις γραμμές του αρχείου .txt, κάνει εισαγωγή και της πρώτης γραμμής που περιλαμβάνεις του τίτλους των πεδίων Για το λόγο αυτό, μετά την εκτέλεσή της σε όλα τα απαραίτητα αρχεία, εκτελούμε διαγραφή (**DELETE**) της συγκεκριμένης γραμμής σε κάθε πίνακα.

```
/*create tables without IDs to import data from files*/
create table agencyImport (agency_id text, agency_name text not null, agency_url
text not null,agency_timezone text not null, agency_lang text, agency_phone text);
create table stopsImport (stop_id text unique not null, stop_code text, stop_name
text not null, stop_desc text, stop_lat text not null, stop_lon text not null,
zone_id text, stop_url text, location_type integer, parent_station text);
create table routesImport (route_id text unique not null, agency_id text,
route_short_name text not null, route_long_name text not null, route_desc text,
route_type integer not null, route_url text, route_color text, route_text_color
text);
create table tripsImport (route_id text not null, service_id text not null, trip_id
text unique not null, trip_headsign text, trip_short_name text, direction_id text,
block_id text, shape_id text);
create table stop_timesImport (trip_id text not null, arrival_time text,
departure_time text, stop_id text not null, stop_sequence integer not null,
stop_headsign text, pickup_type text, drop_off_type text, shape_dist_traveled
text);
create table frequenciesImport (trip_id text not null, start_time text not null,
end_time text not null, headway_secs text);
/*import data*/
.separator ,
.header ON
.import agency.txt agencyImport
.import stops.txt stopsImport
.import routes.txt routesImport
.import trips.txt tripsImport
.import stop_times.txt stop_timesImport
.import frequencies.txt frequenciesImport
/*delete unneded lines*/
delete from agencyImport where agencyImport.agency_id='agency_id';
delete from stopsImport where stopsImport.stop_id='stop_id';
delete from routesImport where routesImport.route_id='route_id';
delete from tripsImport where tripsImport.trip_id='trip_id';
delete from stop_timesImport where stop_timesImport.stop_id='stop_id';
delete from frequenciesImport where frequenciesImport.trip_id='trip_id';
```

Εικόνα 8.4: Script δημιουργίας βάσης SQLite από αρχεία GTFS (α' μέρος)

Στην **Εικόνα 8.5** διακρίνεται πλέον η δημιουργία των τελικών πινάκων όπου περιλαμβάνονται τα πεδία **_id**, ως κλειδιά του πίνακα καθώς και τα ξένα κλειδιά μεταξύ τους. Μετά τη δημιουργία τους εκτελούνται διαδοχικά **INSERT** όλων των τιμών των ενδιάμεσων πινάκων. Ακολουθεί διαγραφή των ενδιάμεσων πινάκων και δημιουργία ενός ευρετηρίου (**index**) στον μεγαλύτερο από τους πίνακες που θα χρησιμοποιήσουμε (stop_times).

Πρέπει να σημειωθεί σε αυτό το σημείο ότι η περιγραφή και το script που παρουσιάστηκε σε αυτή την παράγραφο αφορά μόνο τη βάση που προκύπτει από τα αρχεία GTFS. Η βάση αυτή αντιγράφεται στην συσκευή κατά την πρώτη εκτέλεση της εφαρμογής και στη συνέχεια δημιουργούνται τοπικά οι πίνακες που αφορούν την αποθήκευση αποτελεσμάτων δρομολόγησης και αγαπημένων διαδρομών.

```
/*create final tables*/
create table agency (_id integer primary key autoincrement, agency_id text unique
not null, agency_name text not null, agency_url text not null,agency_timezone text
not null, agency_lang text, agency_phone text);
create table stops (_id integer primary key autoincrement, stop_id text unique not
null, stop_code text, stop_name text not null, stop_desc text, stop_lat text not
null, stop_lon text not null, zone_id text, stop_url text, location_type integer,
parent_station text);
create table routes (_id integer primary key autoincrement, route_id text unique
not null, agency_id text, route_short_name text not null, route_long_name text not
null, route_desc text, route_type integer not null, route_url text, route_color
text, route_text_color text, foreign key(agency_id) references agency(agency_id));
create table trips (_id integer primary key autoincrement, route_id text not null,
service_id text not null, trip_id text unique not null, trip_headsign text,
trip_short_name text, direction_id text, block_id text, shape_id text, foreign
key(route_id) references routes(route_id));
create table stop_times (_id integer primary key autoincrement, trip_id text not
null, arrival_time text, departure_time text, stop_id text not null, stop_sequence
integer not null, stop_headsign text, pickup_type text, drop_off_type text,
shape_dist_traveled text, foreign key(trip_id) references trips(trip_id), foreign
key(stop_id) references stops(stop_id));
create table frequencies (_id integer primary key autoincrement, trip_id text not
null, start_time text not null, end_time text not null, headway_secs text, foreign
key(trip_id) references trips(trip_id));
/*move data from import tables to final tables*/
insert into agency(agency_id, agency_name, agency_url,agency_timezone, agency_lang,
agency_phone) select * from agencyImport;
insert into stops(stop_id, stop_code, stop_name, stop_desc, stop_lat, stop_lon,
zone_id, stop_url, location_type, parent_station) select * from stopsImport;
insert into routes(route_id, agency_id, route_short_name, route_long_name,
route_desc, route_type, route_url, route_color, route_text_color) select * from
routesImport;
insert into trips(route_id, service_id, trip_id, trip_headsign, trip_short_name,
direction_id, block_id, shape_id) select * from tripsImport;
insert into stop_times(trip_id, arrival_time, departure_time, stop_id,
stop_sequence, stop_headsign, pickup_type, drop_off_type, shape_dist_traveled)
select * from stop_timesImport;
insert into frequencies(trip_id, start_time, end_time, headway_secs) select * from
frequenciesImport;
/*drop import tables*/
drop table agencyImport;
drop table stopsImport;
drop table routesImport;
drop table tripsImport;
drop table stop_timesImport;
drop table frequenciesImport;
/*create indeces*/
CREATE INDEX stop_times_stop_id on stop_times(stop_id);
vacuum;
```

Εικόνα 8.5: Script δημιουργίας βάσης SQLite από αρχεία GTFS (β' μέρος)

8.2 Πλατφόρμες και προγραμματιστικά εργαλεία

Σε αυτή την παράγραφο θα παρουσιάσουμε τα εργαλεία που χρησιμοποιήθηκαν κατά την υλοποίηση του συστήματος και θα αναλυθούν οι ευκολίες που προσφέρουν στον προγραμματιστή. Στη συνέχεια θα δούμε τις απαιτήσεις σε software και hardware των συσκευών που μπορούν να «τρέξουν» την εφαρμογή, ενώ θα παρέχουμε και έναν οδηγό εγκατάστασης της εφαρμογής σε πραγματική συσκευή.

8.2.1 Προγραμματιστικά εργαλεία

8.2.1.1 *Android SDK*

Το Android SDK (Software Development Kit) [25] αποτελεί ένα σύνολο από εργαλεία και βιβλιοθήκες που είναι απαραίτητα προκειμένου κάποιος να δημιουργήσει μία εφαρμογή για το λειτουργικό σύστημα Android. Διατίθεται για όλα τα κύρια λειτουργικά συστήματα (Windows, Linux και Mac OS) και στη βασική του έκδοση περιλαμβάνει τον Android SDK Manager ο οποίος επιτρέπει το κατέβασμα των επιμέρους πακέτων και μας ενημερώνει για τις ενημερώσεις τους. Στον **Πίνακας 8.2** περιγράφονται όλα τα πακέτα που είναι διαθέσιμα για κατέβασμα μέσω του Android SDK Manager.

Με τη χρήση των βιβλιοθηκών που παρέχονται με το Android SDK γίνεται δυνατός ο **προγραμματισμός** για την πλατφόρμα του Android, ενώ με τα εργαλεία που παρέχονται γίνεται δυνατή η **εγκατάσταση** της εφαρμογής, ο **έλεγχος** της σε κάποια εικονική μηχανή ή σε πραγματική συσκευή και τέλος η **πιστοποίηση** της για την **δημοσίευση** στο Google Play Store.

Στη συνέχεια θα παρουσιαστούν εν συντομία τα σημαντικότερα εργαλεία που συμπεριλαμβάνονται στο Android SDK.

8.2.1.1.1 Android Emulator

Το Android Emulator είναι ένα εργαλείο που λειτουργεί ως προσομοιωτής συσκευών. Επιτρέπει στον προγραμματιστή να αναπτύξει και να δοκιμάσει εφαρμογές χωρίς να έχει στην κατοχή του μία πραγματική συσκευή. Ο προσομοιωτής αυτός βασίζεται στον QEMU, ένα προσομοιωτή επεξεργαστών, για τη λειτουργία του. Προσομοιώνει όλες τις βασικές λειτουργίες μια πραγματικής συσκευής, όπως την πλήρη λειτουργία του λειτουργικού συστήματος και τις βασικές εφαρμογές του. Επιτρέπει τη χρήση του διαδικτύου, ενώ μπορεί να προσομοιώσει τηλεφωνικές κλήσεις, ακόμη και να δεχτεί location based events προσομοιώνοντας έτσι το GPS της συσκευής.

Βέβαια στην πράξη γίνεται κατανοητό ότι μια πραγματική συσκευή είναι απαραίτητη για το σωστό testing μιας εφαρμογής αφού δίνει την πραγματική αίσθηση χρήσης της και παράλληλα κάνει πιο φυσική τη χρήση των αισθητήρων που εκτός από GPS μπορεί να περιλαμβάνουν επιταχυνσιόμετρο, γυροσκόπιο ή άλλου είδους ανιχνευτές.

8.2.1.1.2 Android Virtual Devices

Εκτός από τον ίδιο τον Android Emulator παρέχεται από το SDK και ένα διαχειριστής των εικονικών συσκευών που δημιουργεί ο προγραμματιστής κατά τη διαδικασία των δοκιμών.

Κάθε εικονική συσκευή (AVD) που δημιουργείται είναι εντελώς ανεξάρτητη από τις άλλες. Διαθέτει ξεχωριστή μνήμη, ανάλυση οθόνης και γενικά τις δικές βασικές ρυθμίσεις και χαρακτηριστικά. Επίσης όταν δημιουργείται μια εικονική συσκευή, επιλέγεται η έκδοση του Android API την οποία θα υλοποιήσει, αλλά και το αν θα υλοποιεί ή όχι το Google Maps API που είναι ξεχωριστό.

Πακέτο	**Περιγραφή**
SDK Tools	Εργαλεία για debugging και testing, καθώς και άλλα απαραίτητα εργαλεία για τη δημιουργία εφαρμογών
SDK Platform-tools	Περιλαμβάνει εργαλεία που αφορούν την πλατφόρμα του Android. Ανανεώνεται κάθε φορά που μία καινούργια πλατφόρμα γίνεται διαθέσιμη.
Documentation	Ένα αντίγραφο της τεκμηρίωσης του Android API
SDK Platform	Υπάρχει ένα SDK Platform για κάθε έκδοση του Android. Περιλαμβάνει ένα αρχείο **android.jar** που περιέχει μία πλήρη βιβλιοθήκη για την πλατφόρμα. Για να δημιουργήσει μία εφαρμογή ο προγραμματιστής πρέπει να καθορίσει μία πλατφόρμα ως στόχο (target).
System Images	Κάθε έκδοση της πλατφόρμας προσφέρει και μία ή περισσότερες εικόνες συστήματος. Αυτές χρησιμοποιούνται από τον Android Emulator για χτίσει εικονικές μηχανές Android.
Sources for Android SDK	Ένα αντίγραφο του κώδικα της πλατφόρμας του Android. Μπορεί να φανεί χρήσιμο κατά την εκσφαλμάτωση (debugging)
Samples for SDK	Μία συλλογή από εφαρμογές – παραδείγματα που παρουσιάζουν μια πλειάδα από τα APIs της πλατφόρμας..
Google APIs	Ένα πρόσθετο του SDK περιλαμβάνει μία βιβλιοθήκη και μία εικόνα συστήματος που μπορούν να χρησιμοποιηθούν για εφαρμογές που χρησιμοποιούν τα Google APIs (π.χ. Google Maps).
Android Support	Μία στατική βιβλιοθήκη που μπορεί να χρησιμοποιηθεί για την υποστήριξη νέων δυνατοτήτων σε παλαιότερες πλατφόρμες που δεν τις υποστηρίζουν εγγενώς.
Google Play Billing	Παρέχει τις στατικές βιβλιοθήκες που χρειάζονται για να προσθέσει ο προγραμματιστής υπηρεσίες επί πληρωμή μέσω του Google Play.
Google Play Licencing	Παρέχει τις στατικές βιβλιοθήκες και παραδείγματα που επιτρέπουν στον προγραμματιστή να επιβεβαιώσει τα πιστοποιητικά του προκειμένου να διαθέσει την εφαρμογή του στο Google Play.

Πίνακας 8.2: Διαθέσιμα πακέτα στον Android SDK Manager

8.2.1.1.3 Android Debug Bridge

Το Android Debug Bridge (adb) είναι ένα εργαλείο χωρίς γραφικό περιβάλλον που επιτρέπει στον προγραμματιστή να επικοινωνήσει με μια εικονική ή μία πραγματική συσκευή συνδεδεμένη στον υπολογιστή του.

Λειτουργεί σαν client-server πρόγραμμα που συμπεριλαμβάνει ένα client που τρέχει στον υπολογιστή που γίνεται ο προγραμματισμός, ένα server που τρέχει επίσης στον υπολογιστή και ένα daemon που υπάρχει σαν background process σε κάθε εικονική ή πραγματική συσκευή. Με αυτό τον τρόπο μπορούμε από πολλούς client να εκτελούμε εντολές προς όλες τις διαθέσιμες συσκευές αφού την επικοινωνία αναλαμβάνει ο server.

Μέσω του adb μπορούμε να εγκαταστήσουμε εφαρμογές ("adb install") σε μία συσκευή, να πάρουμε το log της ("adb logcat"), να μεταφέρουμε αρχεία από ("adb pull") και προς αυτή ("adb push") κ.α.. Δίνεται ακόμη και η δυνατότητα να συνδεθούμε με την κονσόλα εντολών της ίδια της συσκευής και να εκτελέσουμε εντολές μέσα από αυτή ("adb shell").

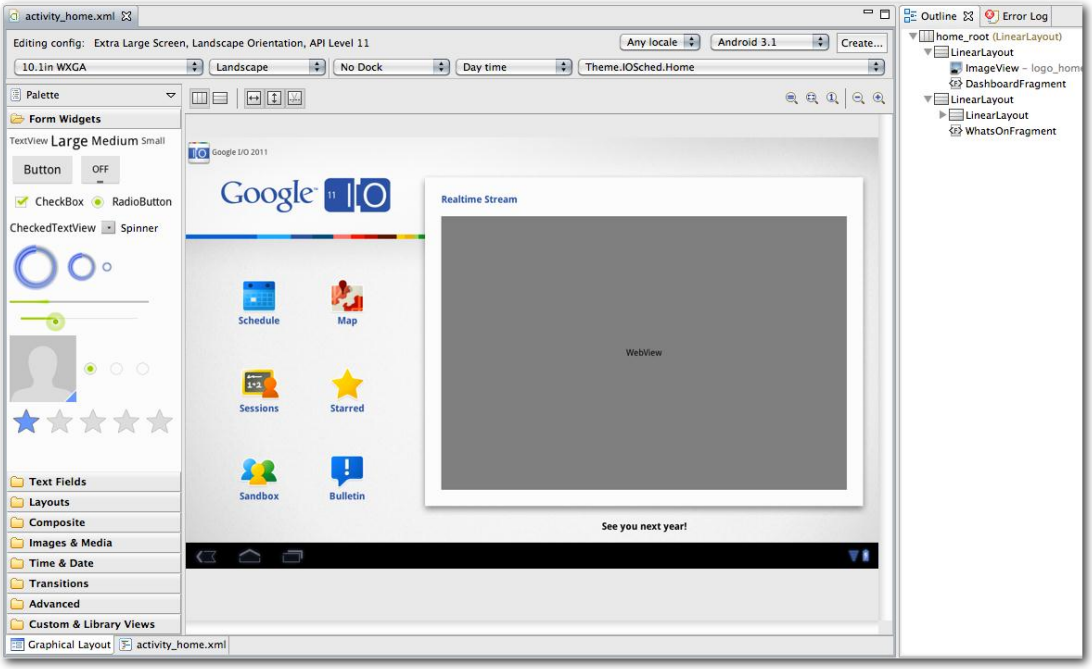
Στην επόμενη παράγραφο θα δούμε πως η ζωή του προγραμματιστή γίνεται πιο εύκολη χρησιμοποιώντας ένα άλλο εργαλείο που δίνεται ως πρόσθετο για το Eclipse IDE.

8.2.1.2 *Android Development Toolkit (Eclipse Plugin)*

Μπορεί το Android SDK να παρέχει όλα όσα χρειάζονται για τη δημιουργία, τον έλεγχο και την διανομή μιας εφαρμογής στην πλατφόρμα του Android σε μορφή ξεχωριστών εργαλείων και βιβλιοθηκών, αλλά οι προγραμματιστές σήμερα έχουν μάθει να δουλεύουν με πολλά βοηθητικά εργαλεία με γραφικό περιβάλλον. Αυτό κάνει τη δουλειά τους πιο εύκολη και τους δίνει χρόνο σκέψης για πιο ουσιώδη πράγματα που αφορούν την ίδια την εφαρμογή και τη λειτουργικότητά της. Ένα τέτοιο εργαλείο είναι το Android Development Toolkit (ADT) [26] που παρέχεται ως πρόσθετο για το Eclipse IDE από την ομάδα του Android.

Μερικά από τα χαρακτηριστικά του ADT είναι:

- Ενσωμάτωση στο Eclipse της **ροής εργασιών (workflow)** που απαιτείται για τη δημιουργία μιας εφαρμογής (δημιουργία project, packaging, εγκατάσταση και debugging).
- Ενσωμάτωση στο γραφικό περιβάλλον του Eclipse πολλών από τα **εργαλεία που προσφέρει το SDK**. Η λειτουργικότητα αυτή πολλές φορές κρύβεται στο παρασκήνιο, όπως για παράδειγμα η επικοινωνία με τη συσκευή και το logging της εφαρμογής γίνονται χωρίς ο προγραμματιστής να χρησιμοποιήσει άμεσα τα εργαλεία **adb** και **logcat**, αφού αυτά χρησιμοποιούνται από το ADT.
- Προσθήκη **ειδικών Java και XML editors** που διευκολύνουν τον προγραμματισμό με ελέγχου σύνταξης, auto completion (αυτόματη ολοκλήρωση κώδικα) και ενσωματωμένη τεκμηρίωση (documentation). Οι XML editors έχουν αυξημένη λειτουργικότητα επιτρέποντας στον προγραμματιστή να επεξεργαστεί τα αρχεία μέσα από φόρμες και να δημιουργήσει γραφικό περιβάλλον για την εφαρμογή του μέσα από ένα περιβάλλον drag and drop (Εικόνα 8.6).



Εικόνα 8.6: Επεξεργαστής γραφικού περιβάλλοντος (ADT)

8.2.2 Απαιτήσεις εφαρμογής

Η εφαρμογή αφορά όλες τις συσκευές που διαθέτουν λειτουργικό **Android στην έκδοση 1.6 (Donut) ή μεγαλύτερη**. Παρόλα αυτά λόγω της σχεδίασης της, δεν ενδείκνυται η χρήση της σε tablets μιας και το user interface δε θα είναι πολύ εύχρηστο λόγω της μεγάλης οθόνης. Δεν υπάρχει βέβαια κανένας λόγος να μην τρέχει η εφαρμογή και σε αυτές τις συσκευές.

Κατά της διάρκεια της εκπόνησης της διπλωματικής εργασίας, η εφαρμογή δοκιμάστηκε σε δύο πραγματικές συσκευές και σε διάφορες εικονικές συσκευές του Android SDK. Οι πραγματικές συσκευές στις οποίες δοκιμάστηκε είναι το **HTC Magic με Android 2.3** και το **Google Nexus S** με διάφορες εκδόσεις Android (**2.3, 4.0 και 4.1**). Το γεγονός ότι πρόκειται για μία αρκετά παλιά και μία σχετικά νεότερη συσκευή αντίστοιχα, δείχνει ότι περιμένουμε από την εφαρμογή να τα πάει αρκετά καλά στις περισσότερες συσκευές που κυκλοφορούν στην αγορά σήμερα.

Η μοναδική απαίτηση της εφαρμογής φαίνεται να είναι σε **μνήμη RAM**, κυρίως λόγω των χαρτών που καταναλώνουν αρκετή μνήμη, αλλά φαίνεται να τα πάει αρκετά καλά στις παραπάνω συσκευές που εκφράζουν σε μεγάλο βαθμό τη μέση συσκευή.

Στην επόμενη παράγραφο θα παρουσιαστεί η διαδικασία εγκατάστασης της εφαρμογής σε μία συσκευή Android.

8.2.3 Εγκατάσταση της εφαρμογής

Γενικά η εγκατάσταση μιας εφαρμογής είναι απλή διαδικασία αν αυτή έχει εκδοθεί στο επίσημο Android Play Store. Για μια εφαρμογή όμως που βρίσκεται ακόμα στο στάδιο υλοποίησης απαιτεί κάποια παραπάνω διαδικασία, που δεν απαιτεί όμως κάποιες ειδικές γνώσεις, μιας και το Android είναι ανοιχτό λειτουργικό και για το λόγο αυτό επιτρέπει εύκολα την εγκατάσταση «εξωτερικών» εφαρμογών.

Για τις οδηγίες εγκατάστασης θεωρούμε ότι ο χρήστης διαθέτει το αρχείο **aTransit.apk** που αποτελεί ουσιαστικά το πακέτο που περιλαμβάνει την εφαρμογή.

8.2.3.1 *Εγκατάσταση μέσω υπολογιστή*

1. **Εγκατάσταση Android SDK**. Για την εγκατάσταση του SDK ο χρήστης μπορεί να κατεβάσει κατάλληλο αρχείο για το σύστημά του (installer για τα Windows ή zip για Linux και Mac) από το http://developer.android.com/sdk/index.html. Στο τέλος της εγκατάστασης θα υπάρχει κάπου ένας κατάλογος android-sdk-<OS>. Στον αρχικό κατάλογο υπάρχει ένας κατάλογος platform-tools όπου βρίσκεται το πρόγραμμα adb που θα χρησιμοποιήσουμε.
2. **Ενεργοποίηση USB Debugging στη συσκευή.** Από τις ρυθμίσεις της συσκευής μας ενεργοποιούμε το USB Debugging ως εξής:
    a. Για Android 2.3 ή παλιότερο η επιλογή βρίσκεται στο Applications → Development → USB Debugging
    b. Για Android 4.0 ή νεότερο η επιλογή βρίσκεται στο Developer options → USB Debugging
3. **Σύνδεση της συσκευής στον υπολογιστή.** Για να ελέγξουμε τη σύνδεση της συσκευής, τη συνδέουμε με καλώδιο USB και από την κονσόλα, στο σημείο που βρίσκεται το πρόγραμμα adb, εκτελούμε την εντολή **adb devices**. Θα πρέπει να δούμε τον κωδικό της συσκευής μας, αλλιώς σημαίνει ότι χρειαζόμαστε επιπλέον drivers για τη συσκευή. Αυτά ισχύουν κυρίως σε windows συστήματα αφού σε Linux και Mac συνήθως η σύνδεση λειτουργεί αμέσως. Στην περίπτωση που διαθέτουμε συσκευή Nexus οι drivers διατίθενται μέσω του SDK και μπορούν να κατέβουν από το εργαλείο διαχείρισης πακέτων. Σε κάθε άλλη περίπτωση οι drivers πρέπει να κατεβούν από το site της εταιρίας που το κατασκευάζει.
4. **Εγκατάσταση και απεγκατάσταση της εφαρμογής.** Για την εγκατάσταση της εφαρμογής **"adb install aTransit.apk"**. Για την απεγκατάσταση της εκτελούμε **"adb uninstall com.baskourtis.aTransit"**.

8.2.3.2 *Εγκατάσταση απευθείας στη συσκευή*

Η διαδικασία μπορεί να γίνει πιο εύκολη αν ο χρήστης διαθέτει κάποιο πρόγραμμα διαχείρισης αρχείων στη συσκευή του. Τέτοια προγράμματα είναι πολύ δημοφιλή στο Android Play Store και μπορούν να εγκατασταθούν και να χρησιμοποιηθούν αρκετά εύκολα. Στην περίπτωση αυτή η διαδικασία είναι η εξής:

1. **Σύνδεση της συσκευής με υπολογιστή για μεταφορά αρχείων**. Η διαδικασία είναι πολύ εύκολη και οι drivers αυτοί είναι συνήθως διαθέσιμοι για κάθε συσκευή σε κάθε σύστημα. Με τη σύνεση της συσκευής στη θήρα USB ένα μήνυμα στην οθόνη μας ρωτάει αν θέλουμε να κάνουμε σύνδεση για μεταφορά αρχείων. Επιλέγουμε τη σύνδεση και εμφανίζεται η εσωτερική μνήμη της συσκευής μας σας εξωτερικός δίσκος. Σε φάκελο της επιλογής μας αποθηκεύουμε το αρχείο aTransit.apk.
2. **Ενεργοποίηση της επιλογής Unknown Sources**. Από τις ρυθμίσεις της συσκευής το ενεργοποιούμε ως εξής:
    a. Για Android 2.3 ή παλιότερο Applications → Development → Unknown Sources.
    b. Για Android 4.0 ή νεότερο Security → Unknown Sources
3. Με το πρόγραμμα διαχείρισης αρχείων ανοίγουμε τον κατάλογο που αποθηκεύσαμε το aTransit.apk και το επιλέγουμε. Αποδεχόμαστε την εγκατάσταση στην επόμενη οθόνη και η εφαρμογή εγκαθίσταται κανονικά στη συσκευή.

Στην ενότητα αυτή, είδαμε αρκετές λεπτομέρειες από την υλοποίηση της εφαρμογής aTransit, καθώς και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξή της. Στην ενότητα που ακολουθεί θα δώσουμε έναν αναλυτικό οδηγό χρήσης της εφαρμογής με πολλές εικόνες από το τελικό αποτέλεσμα.

9

Εγχειρίδιο χρήσης

Στο προηγούμενο κεφάλαιο αναλύσαμε τον τρόπο με τον οποίο δουλεύει ο προγραμματιστής και το πώς κάποιος μπορεί να εγκαταστήσει την εφαρμογή αυτής της διπλωματικής στη συσκευή του. Στην ουσία ο χρήστης χρειάζεται μόνο το εργαλείο adb για να πραγματοποιήσει την εγκατάσταση. Στην πραγματικότητα, οι καθημερινοί χρήστες δεν έρχονται σε επαφή με τέτοιου είδους προγραμματιστικά εργαλεία. Εντοπίζουν τις διαθέσιμες εφαρμογές στο Google Play Store, τις κατεβάζουν μέσα από την συσκευή τους και αρχίζουν αμέσως να τη χρησιμοποιούν. Για το λόγο αυτό, στο κεφάλαιο αυτό θα συζητήσουμε θέματα που αφορούν τη χρήστη της εφαρμογής μετά την εγκατάστασή της. Θα παρουσιαστούν διάφορα σενάρια χρήσης και θα επιδειχθεί ο τρόπος με τον οποίο αυτά υλοποιούνται από την εφαρμογή.

9.1 Σενάρια χρήσης

Όπως φάνηκε και από τα προηγούμενα κεφάλαια, η συγκεκριμένη εφαρμογή δεν υλοποιεί ένα συγκεκριμένο σκοπό ή σενάριο χρήσης. Στόχος της είναι η βελτιστοποίηση της χρήσης των αστικών συγκοινωνιών της Αθήνας και τον επιτυγχάνει με διαφορετικούς τρόπους, προσφέροντας, για παράδειγμα, online δρομολόγηση, αλλά και offline στατικά δεδομένα για τα αστικά μέσα μεταφοράς.

Χρησιμοποιώντας έναν τέτοιο διαχωρισμό και προσθέτοντας τις διαδικαστικές λειτουργίες της εφαρμογής, όπως τις ρυθμίσεις, καταλήξαμε στα παρακάτω σενάρια χρήστης:

- Δρομολόγηση από σημείο σε σημείο
    - Καθορισμός δεδομένων αναζήτησης
    - Προβολή της προτεινόμενης διαδρομής με κείμενο ή χάρτη

    - Αποθήκευση για offline χρήση
    - Επεξεργασία ρυθμίσεων του αλγορίθμου δρομολόγησης
- Προβολή λεπτομερειών για τις διαθέσιμες διαδρομές και στάσεις
    - Προβολή λίστας με τα διαθέσιμα μέσα
    - Προβολή λεπτομερειών διαδρομής με κείμενο ή χάρτη
    - Προβολή διερχόμενων γραμμών από στάση
    - Αποθήκευση διαδρομής στα «Αγαπημένα»
- Προβολή κοντινών στάσεων γύρω από σημείο
    - Επιλογή σημείου σε χάρτη
    - Εύρεση κοντινών στάσεων
    - Προβολή διερχόμενων γραμμών για κάθε στάση
- Αναζήτηση διαδρομών και στάσεων
- Επιπλέον λειτουργίες που αφορούν το χάρτη
    - Εντοπισμός θέσης του χρήστη
    - Κατέβασμα χάρτη για offline χρήση
    - Ρύθμιση επιλογών που αφορούν την εμφάνιση του χάρτη

Στην επόμενη παράγραφο θα παρουσιαστούν εκτενώς τα παραπάνω σενάρια χρήσης και οι επιμέρους διαδικασίες τους. Θα χρησιμοποιήσουμε αναλυτικές εικόνες της εφαρμογής προκειμένου να δώσουμε όσο γίνεται καλύτερη αίσθηση της πραγματικής χρήσης.

9.2 Αναλυτική παρουσίαση

9.2.1 Δρομολόγηση από σημείο σε σημείο

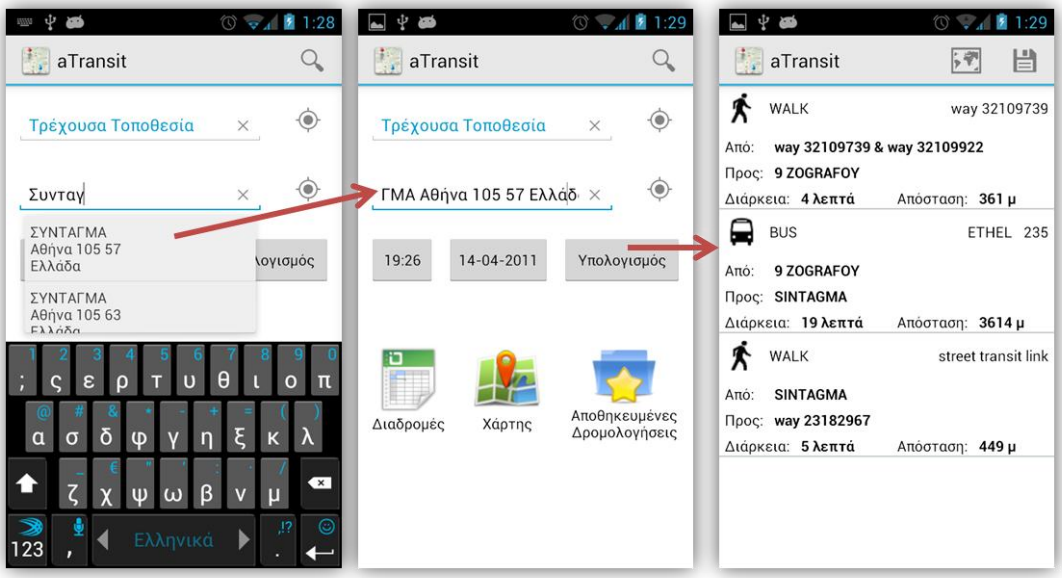
Η δρομολόγηση από σημείο σε σημείο είναι μία από τις βασικότερες λειτουργίες της εφαρμογής. Είναι η μόνη που χρειάζεται πρόσβαση στο διαδίκτυο για να έρθει σε πέρας και μπορεί να εκκινήσει από δύο διαφορετικά σημεία.

Το ένα είναι η κεντρική οθόνη της εφαρμογής όπου ο χρήσης μπορεί να καθορίσει διεύθυνση αφετηρίας και προορισμού καθώς και ώρα και ημερομηνία του ταξιδιού. Το άλλο σημείο είναι ο χάρτης όπου ο χρήστης μπορεί να επιλέξει σημείο αφετηρίας και προορισμού και να εκτελέσει την αναζήτηση βέλτιστης διαδρομής. Οι δύο αυτές περιπτώσεις χρήσης φαίνονται στις **Εικόνα 9.1 και Εικόνα 9.2** αντίστοιχα.

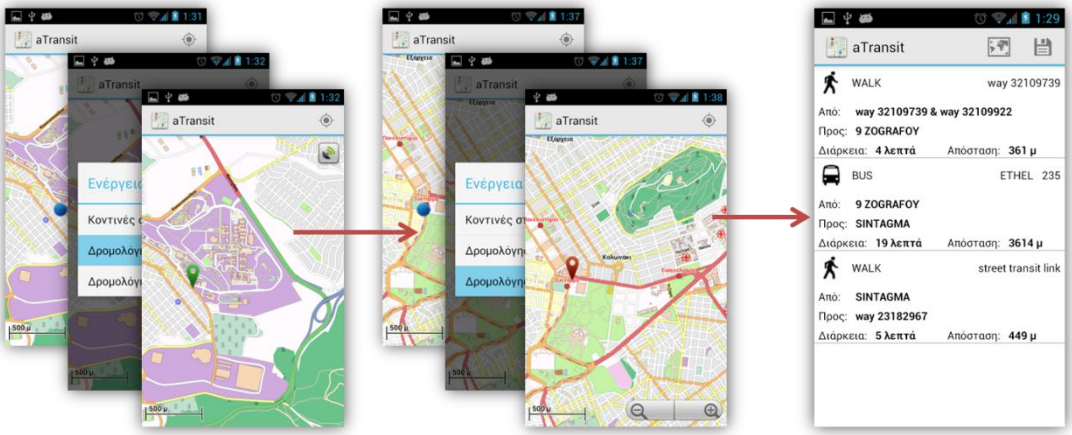
Όπως φαίνεται και από την εικόνα, στην περίπτωση συμπλήρωσης της διεύθυνσης υπάρχει σύστημα υποδείξεων για τη διεύθυνση που προσπαθεί να πληκτρολογήσει ο χρήσης προκειμένου να τον βοηθήσει να επιλέξει ακριβώς τη διεύθυνση που επιθυμεί. Επίσης με το κουμπί που βρίσκεται δίπλα σε κάθε πεδίο μπορεί να επιλέξει την τρέχουσα τοποθεσία του ως σημείο αφετηρίας ή προορισμού.

Αντίστοιχα στην περίπτωση του χάρτη ο χρήστης επιλέγει (με long click πάνω στο χάρτη) ένα σημείο και στη συνέχεια επιλέγοντας αυτό το σημείο μπορεί να το μετατρέψει σε σημείο

αφετηρίας ή προορισμού. Όταν έχουν επιλεγεί και τα δύο σημεία ξεκινάει αυτόματα η διαδικασία δρομολόγησης.

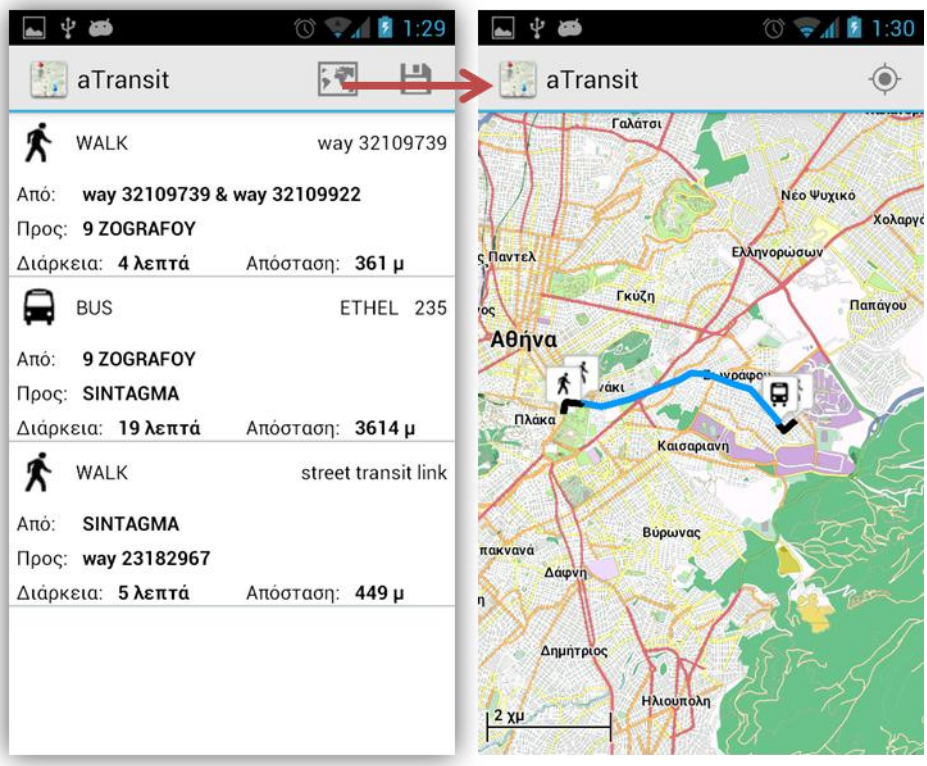


Εικόνα 9.1: Δρομολόγηση με συμπλήρωση διευθύνσεων

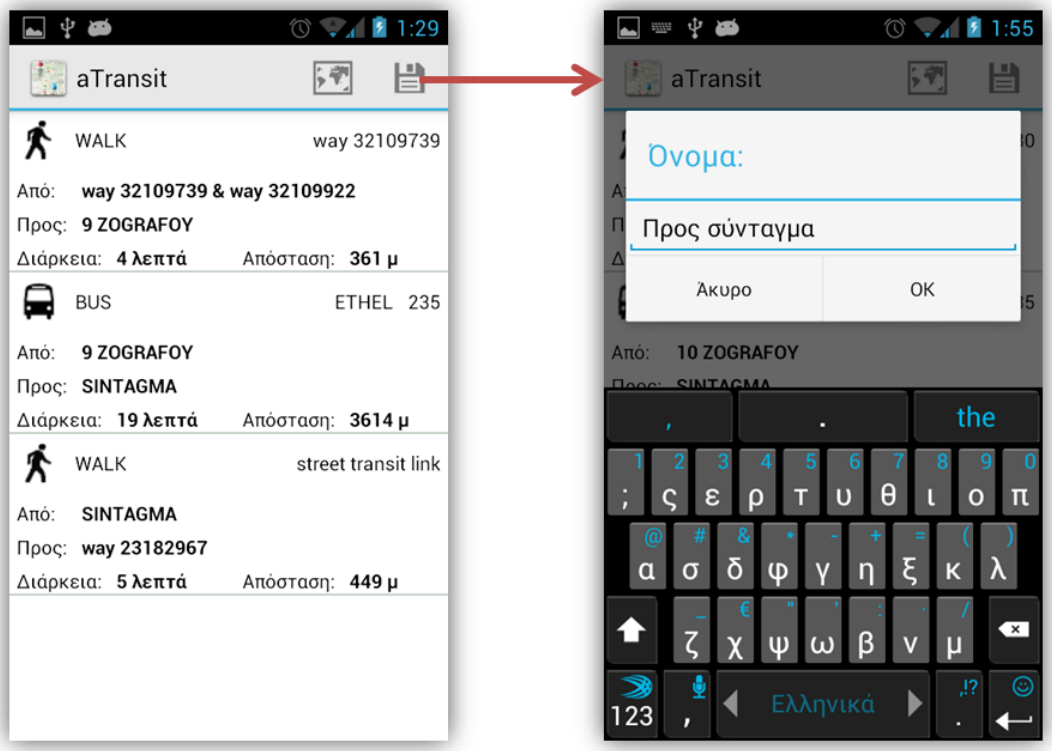


Εικόνα 9.2: Δρομολόγηση με επιλογή σημείων από χάρτη

Όπως επίσης φαίνεται στις εικόνες, η διαδικασία δρομολόγησης καταλήγει σε μία λίστα από τις ενέργειες που πρέπει να ακολουθήσει ο χρήστης για να φτάσει στον προορισμό του. Αυτή η λίστα περιλαμβάνει διαδρομές αστικών συγκοινωνιών αλλά και περπάτημα από σημείο σε σημείο. Πατώντας το πλήκτρο μενού της συσκευής ο χρήσης έχει κάποιες επιπλέον δυνατότητες. Αυτές που αφορούν τη δρομολόγηση είναι η προβολή της προτεινόμενης διαδρομής στο χάρτη και η αποθήκευσή της για μετέπειτα offline προβολή. Αυτές οι δύο ενέργειες επιδεικνύονται χαρακτηριστικά στην **Εικόνα 9.3** και στην **Εικόνα 9.4** αντίστοιχα.

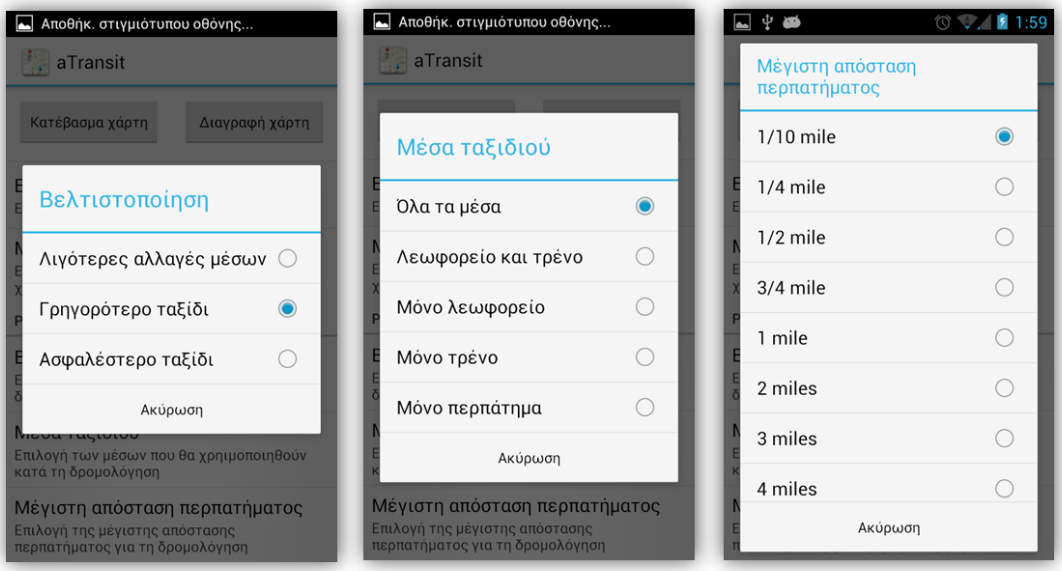


Εικόνα 9.3: Προβολή δρομολόγησης σε χάρτη



Εικόνα 9.4: Αποθήκευση δρομολόγησης

Εκτός από τα δεδομένα της αναζήτησης που μπορεί να θέσει ο χρήστης στην αρχική οθόνη, υπάρχουν και άλλες, πιο εξειδικευμένες, επιλογές δρομολόγησης που μπορεί να επεξεργαστεί μέσα από την κεντρική οθόνη ρυθμίσεων της εφαρμογής. Αυτές αφορούν άμεσα τον αλγόριθμο δρομολόγησης και αλλάζουν τον τρόπο με τον οποίο θα εκτελέσει την αναζήτηση. Στην **Εικόνα 9.5** μπορεί κανείς να διακρίνει αυτές τις επιλογές. Η πρώτη αφορά τη βελτιστοποίηση της δρομολόγησης, η δεύτερη τον επιθυμητό τρόπο ταξιδιού και η τελευταία τη μέγιστη επιθυμητή απόσταση περπατήματος. Είναι φανερό ότι αυτές οι επιλογές επηρεάζουν άμεσα της μεταβλητές του request που θα σταλεί στον OpenTripPlanner όπως παρουσιάστηκαν στην παράγραφο 8.1.1.1.

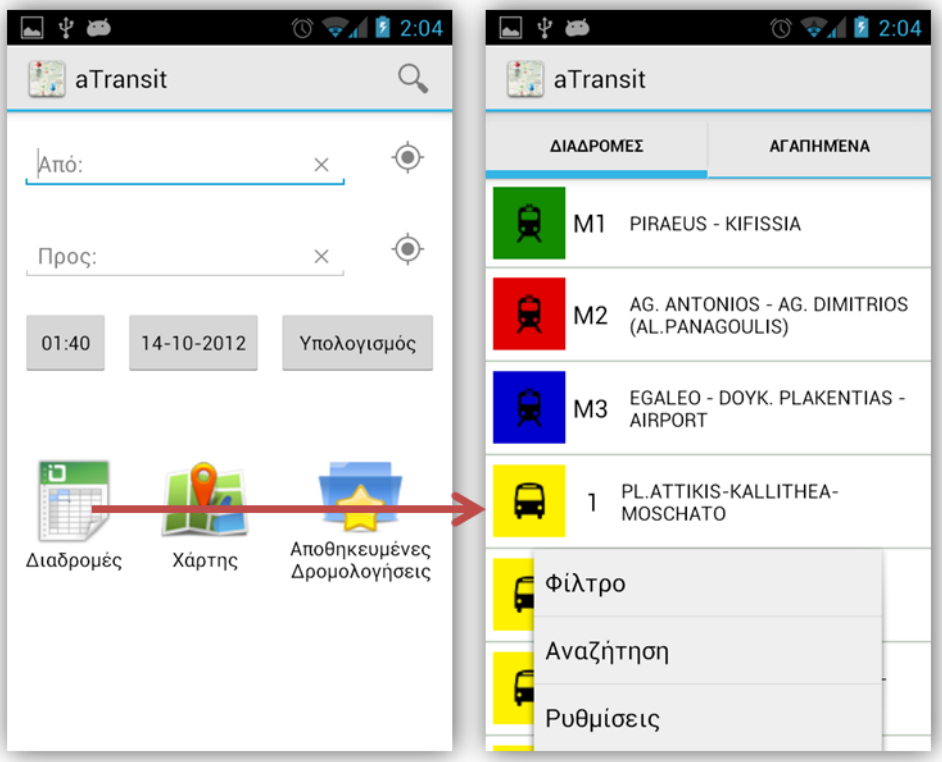


Εικόνα 9.5: Επιπλέον επιλογές δρομολόγησης στις ρυθμίσεις της εφαρμογής

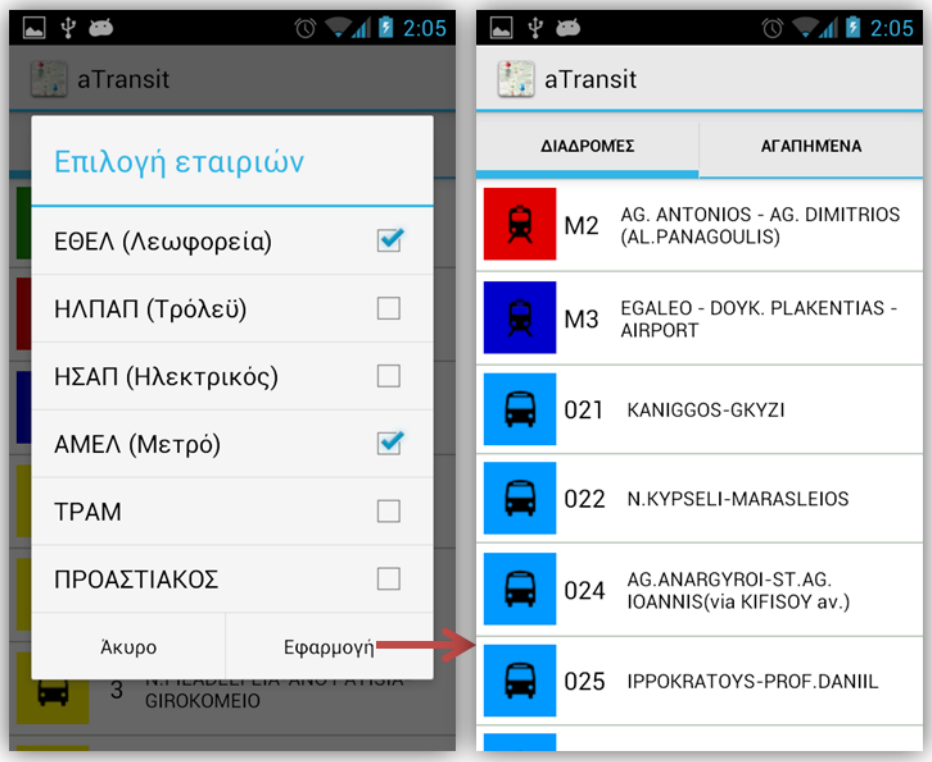
9.2.2 Προβολή λεπτομερειών για τις διαθέσιμες διαδρομές και στάσεις

Για οτιδήποτε άλλο πέρα από την δρομολόγηση από σημείο σε σημείο δε χρειάζεται πρόσβαση σε εξωτερικό server αφού υπάρχει η τοπική βάση δεδομένων. Αυτό φαίνεται καθαρά στο σενάριο εύρεσης λεπτομερειών για διαδρομές και στάσεις. Ο χρήστης έχει πρόσβαση σε αυτές τις λειτουργίες από το εικονίδιο «Διαδρομές» στην κεντρική οθόνη της εφαρμογής. Στις παρακάτω εικόνες γίνονται εμφανείς οι δυνατότητες της εφαρμογής σε αυτόν τον τομέα.

Στην **Εικόνα 9.6** παρουσιάζεται ο τρόπος εισόδου στη λίστα με τις διαθέσιμες εφαρμογές καθώς και οι επιλογές του μενού αυτής της λίστα, ενώ όπως φαίνεται στην **Εικόνα 9.7** δίνεται η δυνατότητα φιλτραρίσματος της λίστας με βάση τις εταιρίες που εκτελούν τα δρομολόγια.

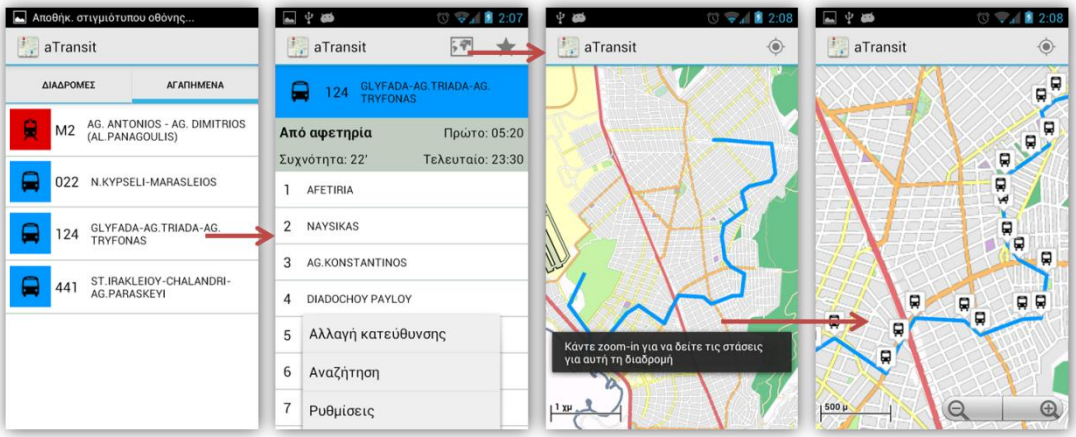


Εικόνα 9.6: Είσοδος στη λίστα διαδρομών

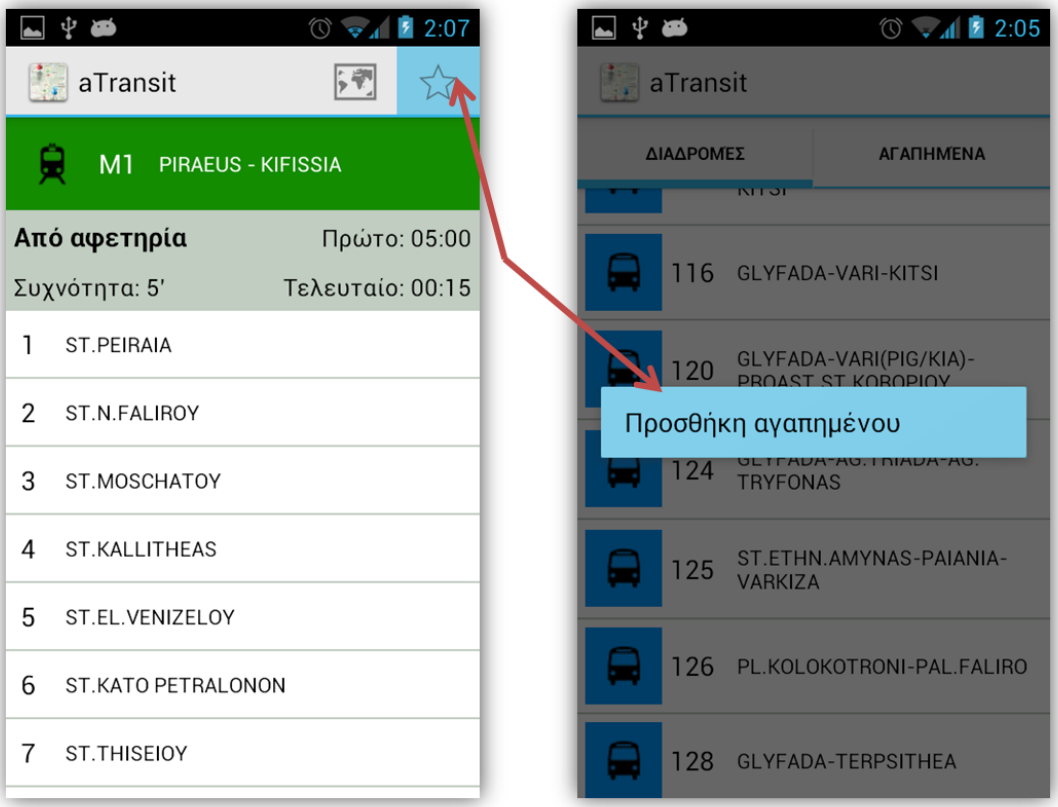


Εικόνα 9.7: Φιλτράρισμα διαδρομών με βάση την εταιρία

Στις επόμενες εικόνες μπορεί κανείς να δε τον τρόπο με τον οποίο μπορεί ο χρήστης να προχωρήσει στις λεπτομέρειες μίας διαδρομής βλέποντας τις στάσεις από τις οποίες διέρχεται, τη συχνότητα εκτέλεσής της, αλλά και να αποκτήσει μία καλύτερη εικόνα για αυτή παρατηρώντας τη στο χάρτη (**Εικόνα 9.8**). Παρατηρούμε ακόμη τον τρόπο με τον οποίο μπορεί ο χρήστης να προσθέσει μία διαδρομή στις «αγαπημένες» του (**Εικόνα 9.9**), είτε μέσα από την ανάλυση διαδρομής είτε από τη λίστα διαδρομών με long-click. Οι «αγαπημένες» διαδρομές είναι προσβάσιμες από την ίδια οθόνη με τη λίστα διαδρομών (δεύτερη καρτέλα).

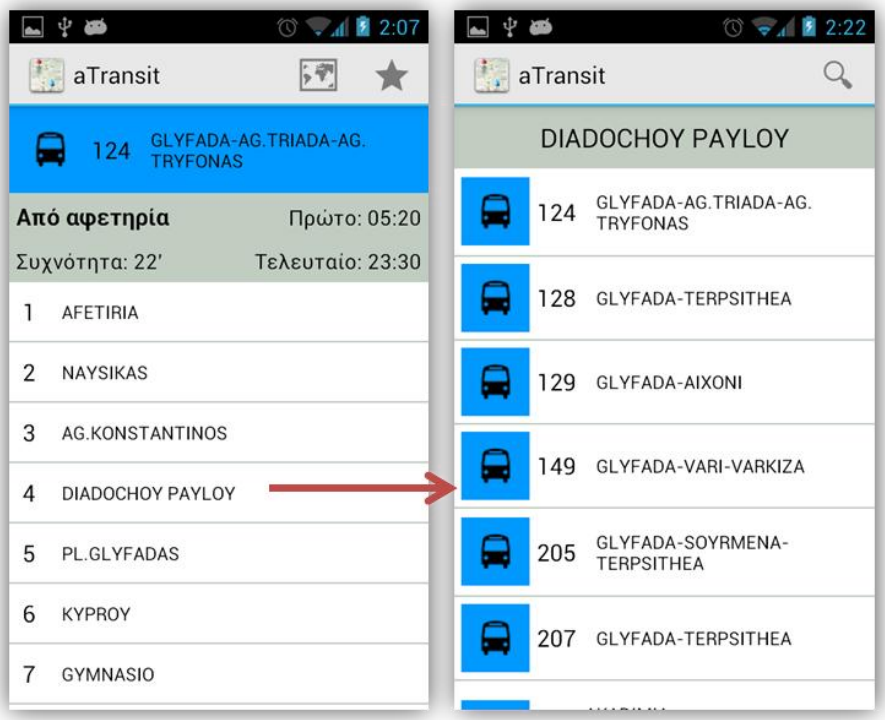


Εικόνα 9.8: Λεπτομέρειες διαδρομής

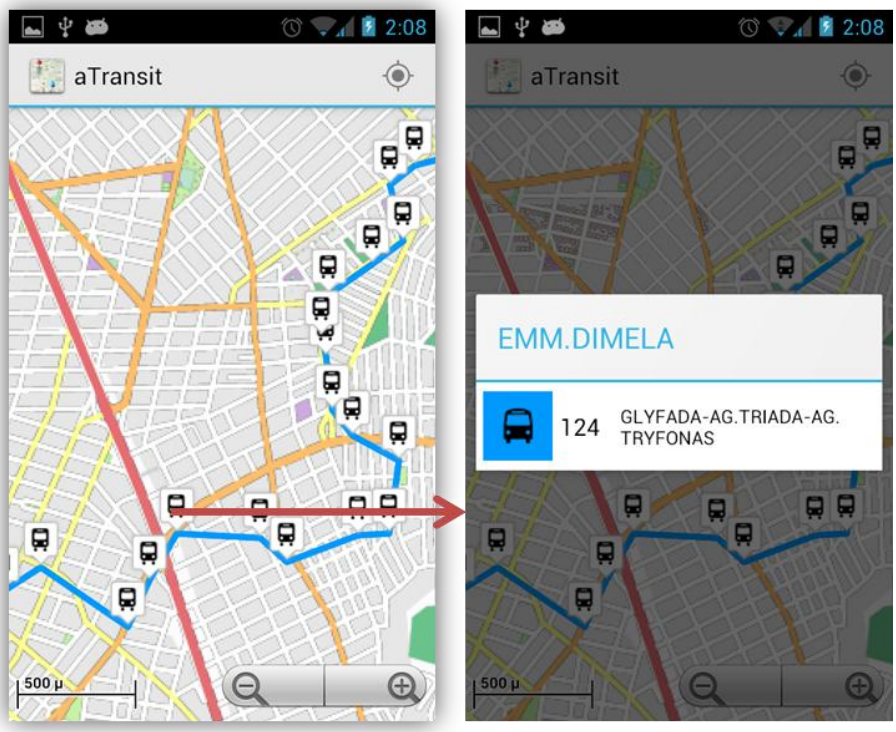


Εικόνα 9.9: Δύο τρόποι προσθήκης διαδρομής στα "Αγαπημένα"

Στις τελευταίες δύο εικόνες, για αυτή την παράγραφο, διακρίνεται ο τρόπος με τον οποίο ο χρήστης μπορεί να δει τις λεπτομέρειες για μίας στάση, δηλαδή τις διαδρομές που διέρχονται από αυτή, απλά πατώντας επιλέγοντας τη στάση, είτε βρίσκεται σε λίστα είτε στο χάρτη.



Εικόνα 9.10: Λεπτομέρειες στάσης (από λίστα)

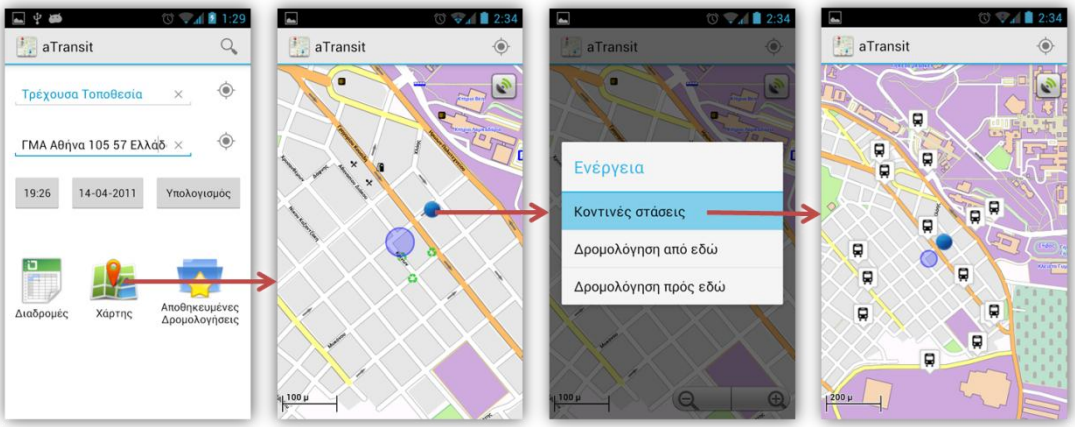


Εικόνα 9.11: Λεπτομέρειες στάσης (από χάρτη)

9.2.3 *Προβολή κοντινών στάσεων γύρω από σημείο (offline)*

Σε αρκετές περιπτώσεις είναι χρήσιμο να δούμε τις στάσεις που βρίσκονται σε μια λογική απόσταση γύρω από το σημείο που βρισκόμαστε. Αυτή η δυνατότητα προσφέρεται από την εφαρμογή και, επειδή τα δεδομένα βρίσκονται αποθηκευμένα στην τοπική βάση, είναι δυνατή η προβολή τους στο χάρτη χωρίς καμία σύνδεση στο διαδίκτυο. Με την προϋπόθεση βέβαια ότι χρησιμοποιείται ο offline χάρτης της εφαρμογής.

Προκειμένου να εντοπίσει ο χρήστης τις κοντινές στάσεις, πρέπει να ανοίξει αρχικά το χάρτη, στη συνέχει να επιλέξει ένα σημείο (με παρατεταμένο πάτημα) και αφού κάνει κλικ στο σημείο αυτό να επιλέξει «κοντινές στάσεις» από το μενού που θα εμφανιστεί[1]. Η διαδικασία που μόλις περιγράφηκε φαίνεται καθαρά στην **Εικόνα 9.12**.



Εικόνα 9.12: Προβολή κοντινών στάσεων

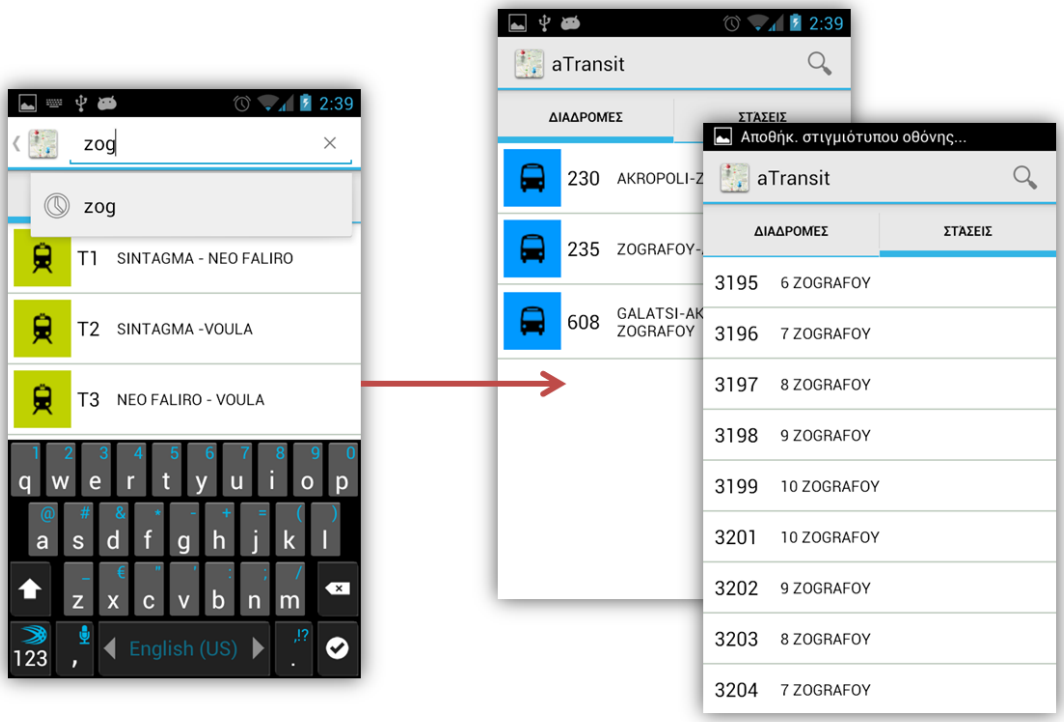
Αν ο χάρτης δεν είναι «κεντραρισμένος» κοντά στην περιοχή του χρήστη, τότε μπορεί να χρησιμοποιήσει την λειτουργία εντοπισμού θέσης του χάρτη ή οποία θα περιγραφεί σε επόμενη παράγραφο.

9.2.4 *Αναζήτηση διαδρομών και στάσεων*

Στις εικόνες τις εφαρμογής, που έχουν παρουσιαστεί ως τώρα, μπορεί κανείς να παρατηρήσει ότι υπάρχει πάντα πάνω δεξιά ένα πλήκτρο αναζήτησης. Σε παλαιότερες εκδόσεις Android αυτό μπορεί να φαίνεται στο μενού της κάθε οθόνης που εμφανίζεται με το πάτημα του πλήκτρου «μενού» της συσκευής. Πατώντας αυτό το πλήκτρο ο χρήστης μπορεί να εκκινήσει τη λειτουργία αναζήτησης διαδρομών και στάσεων. Στην **Εικόνα 9.13** φαίνεται η διαδικασία αναζήτησης: ο χρήστης πληκτρολογεί αριθμό ή κείμενο και στη συνέχεια μεταφέρεται σε μία οθόνη με δύο καρτέλες, μία για τα αποτελέσματα σε διαδρομές και μία για τα αποτελέσματα

[1] Πρόκειται για το ίδιο μενού από το οποίο μπορεί να επιλέξει σημείο αφετηρίας ή προορισμού.

σε στάσεις. Η αναζήτηση επιτυγχάνει ακόμη κι αν ο χρήστης πληκτρολογήσει μέρος της ονομασίας της στάσης ή της διαδρομής, ενώ το ιστορικό αναζητήσεων αποθηκεύεται ώστε να διευκολύνει μελλοντικές αναζητήσεις.

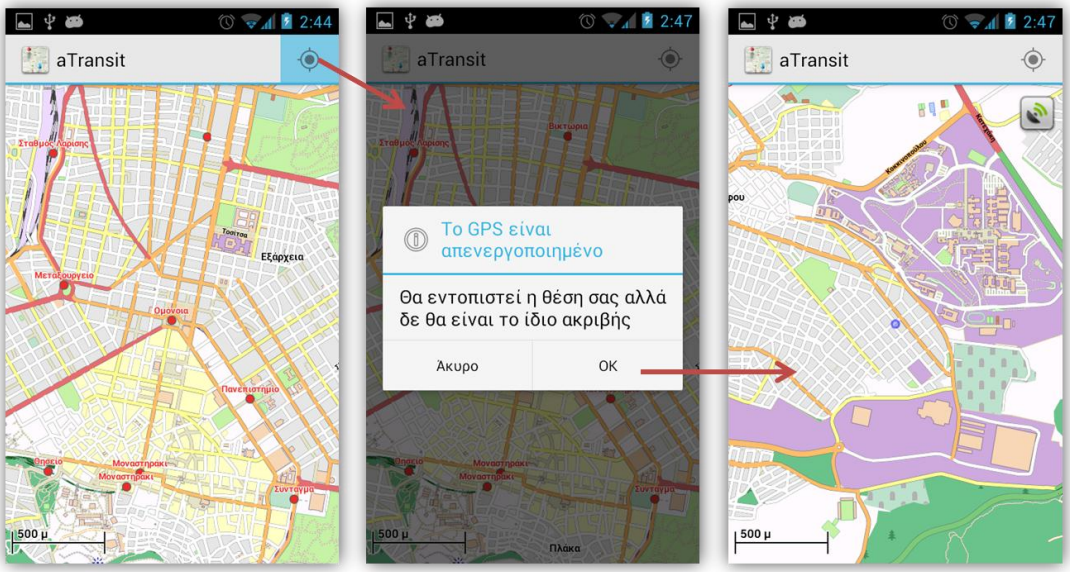


Εικόνα 9.13: Αναζήτηση διαδρομών/στάσεων

9.2.5 Επιπλέον λειτουργίες του χάρτη

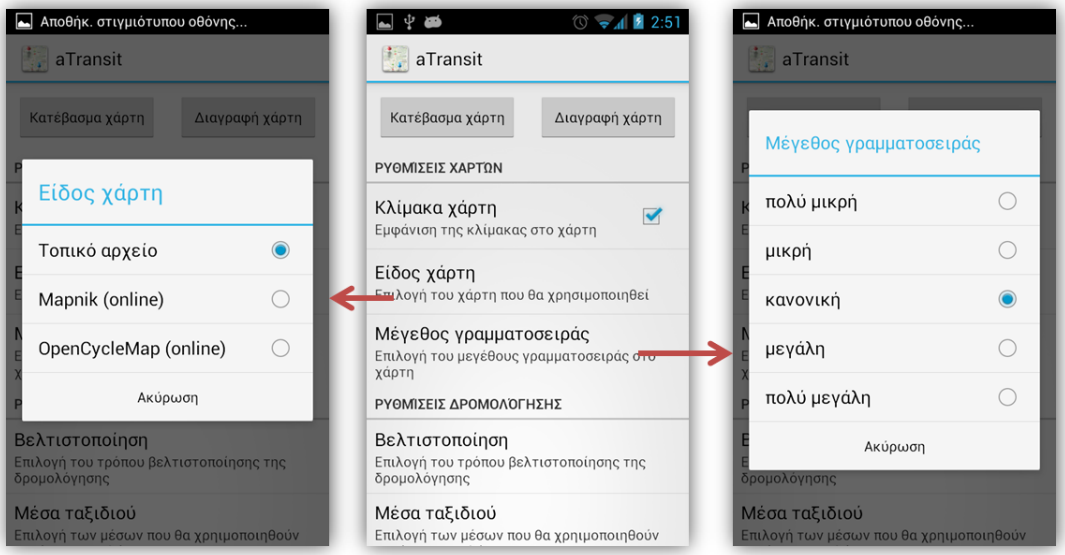
Το υποσύστημα χαρτών της εφαρμογής χρησιμοποιείται από όλα τα άλλα υποσυστήματα. Ως ανεξάρτητο υποσύστημα όμως διαθέτει δικά του χαρακτηριστικά και επιπλέον λειτουργικότητα τα οποία είναι εμφανή σε κάθε του χρήση.

Μία βασική λειτουργιά του χάρτη είναι ότι μπορεί να εντοπίζει τη θέση του χρήστη και να «κεντράρει» τον χάρτη σε αυτή την τοποθεσία. Αυτή η δυνατότητα προσφέρεται σε κάθε στιγμιότυπο του υποσυστήματος χαρτών, είναι προσβάσιμη μέσα από το μενού της οθόνης και ο τρόπος λειτουργίας της φαίνεται στην **Εικόνα 9.14**. Ο χρήστης επιλέγει τον εντοπισμό θέσης από το μενού της οθόνης, στη συνέχεια ένα μήνυμα εμφανίζεται για να τον ενημερώσει (σε περίπτωση που το GPS είναι απενεργοποιημένο) ότι η ποιότητα του εντοπισμού μπορεί να είναι περιορισμένη και τέλος στην οθόνη του εμφανίζεται η θέση του και ο χάρτης έχει «κεντραριστεί» σε αυτή.



Εικόνα 9.14: Εντοπισμός θέσης στο χάρτη

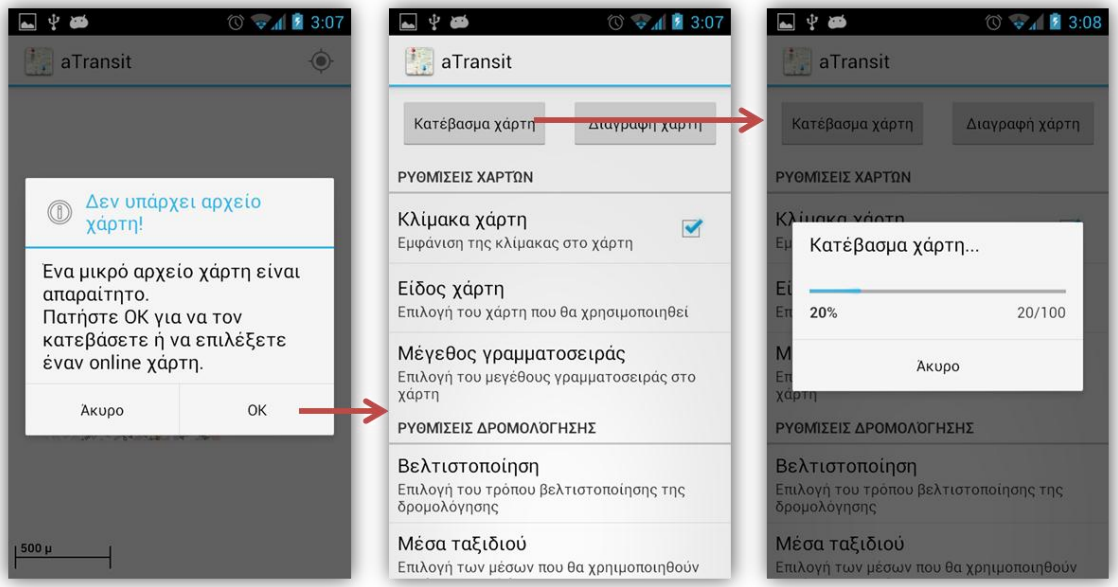
Για το υποσύστημα του χάρτη υπάρχουν ακόμη κάποιες ειδικές ρυθμίσεις που αφορούν τον τρόπο εμφάνισής του, αλλά και το αν θα χρησιμοποιεί το διαδίκτυο. Όπως φαίνεται στην **Εικόνα 9.15** οι ρυθμίσεις αυτές περιλαμβάνουν τον τύπο του χάρτη (τρεις τύποι, δύο εκ των οποίων απαιτούν σύνδεση στο διαδίκτυο), το μέγεθος των γραμματοσειρών στο χάρτη, αλλά και το αν θα φαίνεται ή όχι η κλίμακα του χάρτη.



Εικόνα 9.15: Ρυθμίσεις του χάρτη της εφαρμογής

Αν ο χρήστης επιλέξει Database renderer σαν τύπο χάρτη αυτό σημαίνει ότι επιθυμεί να χρησιμοποιήσει τον offline χάρτη της εφαρμογής. Για λόγους εξοικονόμησης χώρου ο χάρτης αυτός επιλέχθηκε να αποθηκεύεται ξεχωριστά από την εφαρμογή. Για το λόγο αυτό, μέσα από τις ρυθμίσεις της εφαρμογής υπάρχει δυνατότητα κατεβάσματος του αρχείου στην εξωτερική μνήμη της συσκευής. Αν ο χρήστης προσπαθήσει να χρησιμοποιήσει τον offline

χάρτη χωρίς να έχει προηγουμένως κατεβάσει το απαιτούμενο αρχείο, ειδοποιείται σχετικά για να ξεκινήσει τη διαδικασία. Η διαδικασία φαίνεται καθαρά στην **Εικόνα 9.16**. Η πρώτη οθόνη δεν είναι απαραίτητο να εμφανιστεί, αφού η διαδικασία μπορεί να ξεκινήσει κανονικά από τις ρυθμίσεις της εφαρμογής.



Εικόνα 9.16: Κατέβασμα χάρτη για offline χρήση

Για λόγους αποφυγής ανεπιθύμητων χρεώσεων επιλέχθηκε ο offline χάρτης να είναι η προκαθορισμένη επιλογή, έτσι κατά πάσα πιθανότητα ο χρήστης θα αντικρίσει το μήνυμα της πρώτης οθόνης αν προσπαθήσει να χρησιμοποιήσει το χάρτη για πρώτη φορά. Αυτό όμως κρίνεται ότι δεν αποτελεί πρόβλημα, αφού ενημερώνεται κατάλληλα και μπορεί να αλλάξει τη ρύθμιση σε online χάρτη αν επιθυμεί να χρησιμοποιήσει το χάρτη άμεσα.

Όπως μπορεί κανείς να παρατηρήσει από τις εικόνες της καρτέλας ρυθμίσεων, υπάρχει και η επιλογή για **διαγραφή του χάρτη**. Εφόσον κάποιος χρήστης επιλέξει να χρησιμοποιεί κάποιον online χάρτη, είναι περιττό να διατηρεί το αρχείο του χάρτη στη μνήμη του, έστω κι αν αυτό είναι μικρού μεγέθους (περίπου 4MByte). Επιλέγοντας την «Διαγραφή χάρτη» ο χρήστης ερωτάται αν είναι βέβαιος ότι θέλει να συνεχίσει και στη συνέχει πραγματοποιείται η διαγραφή του αρχείου.

10

Επίλογος

10.1 Σύνοψη και συμπεράσματα

Στα προηγούμενα κεφάλαια είδαμε το πρόβλημα της βελτιστοποίησης της χρήσης των αστικών συγκοινωνιών και προτείναμε τη δική μας λύση σε αυτό ορίζοντας τις προδιαγραφές και υλοποιώντας μια εφαρμογή για κινητές συσκευές Android βασιζόμενοι στους ήδη υλοποιημένους αλγορίθμους δρομολόγησης του OpenTripPlanner.

Κατά τη διαδικασία της προδιαγραφής της εφαρμογής παρουσιάστηκε η ιδέα της μεταφοράς, όσο το δυνατόν περισσότερων, υποσυστημάτων της συσκευής τοπικά ώστε να δουλεύουν χωρίς τη χρήση διαδικτύου. Αυτό πραγματοποιήθηκε προσθέτοντας τα διαθέσιμα δεδομένα σε μια τοπική βάση δεδομένων στη συσκευή. Σημαντικό ρόλο σε αυτή την υλοποίηση έπαιξε ο εντοπισμός της κατάλληλης βιβλιοθήκης για χρήση των δεδομένων του OpenStreetMap offline. Με αυτό τον τρόπο γίνεται καλύτερη και πιο άμεση οπτικοποίηση των τοπικών δεδομένων, αλλά και των δεδομένων δρομολόγησης.

Η παρούσα εργασία συνοψίζεται στην ανάπτυξη, από την αρχή ως το τέλος, μιας Android εφαρμογής που διευκολύνει τον κάτοικο ή τον επισκέπτη της πόλης της Αθήνας στην αναζήτησή του βέλτιστου τρόπου μετακίνησης μέσα σε αυτή. Το αποτέλεσμα δείχνει να είναι μία εύχρηστη εφαρμογή, που σέβεται τους οδηγούς σχεδίασης εφαρμογών για το Android OS [27] και επιτυγχάνει το σκοπό της με τρόπο φιλικό προς το χρήση. Τέλος, η χρήση προγραμμάτων και βιβλιοθηκών ανοιχτού κώδικα, αλλά κυρίως η χρήση ανοιχτών προτύπων δεδομένων. όπως το GTFS [2], κάνουν την εφαρμογή πιο ευέλικτη και επεκτάσιμη στο μέλλον.

10.2 Μελλοντικές επεκτάσεις

Η χρήση ανοιχτού λογισμικού και προτύπων κάνει, όπως είπαμε, την εφαρμογή που δημιουργήθηκε στα πλαίσια αυτής της εργασίας, αρκετά ευέλικτη σε αλλαγές. Εκμεταλλευόμενοι αυτή την ευελιξία μπορούμε να προτείνουμε κάποιες επεκτάσεις που θα μπορούσαν να γίνουν βραχυπρόθεσμα χωρίς πολύ μεγάλες αλλαγές στον κώδικα της εφαρμογής:

- **Χρήση εναλλακτικών server.** Λόγω της χρήσης του OpenTripPlanner και το API με XML για την επικοινωνία είναι αρκετά εύκολο να χρησιμοποιήσουμε κάποιο άλλο server που υλοποιεί το ίδιο πρότυπο για τη δρομολόγηση. Αυτό σημαίνει ότι ο χρήστη θα μπορούσε να διαλέγει το server που θα χρησιμοποιεί βλέποντας ποιος τον εξυπηρετεί καλύτερα και δίνονται και το κατάλληλο feedback για βελτιώσεις.
- **Ενημέρωση τοπικής βάσης σε συγχρονισμό με το server.** Θα μπορούσε να υλοποιηθεί ένας τρόπος επικοινωνίας για τον συγχρονισμό των δεδομένων μεταξύ client και server ώστε να μπορεί να γίνει ανανέωση των τοπικών δεδομένων διατηρώντας τη συνέπεια μεταξύ των δύο.
- **Εφαρμογή της ίδια υλοποίησης σε άλλες πόλεις.** Συνδυάζοντας τις δύο προηγούμενες προτάσεις και κάνοντας τις απαραίτητες αλλαγές στον κώδικα ώστε να γίνει όσο το δυνατόν λιγότερο ειδικός για μέσα της Αθήνας, η εφαρμογή μπορεί δυνητικά να χρησιμοποιηθεί οπουδήποτε υπάρχουν διαθέσιμα δεδομένα σε μορφή GTFS.

Τέλος, μία ιδέα για επέκταση είναι η **υλοποίηση του ίδιου του αλγορίθμου δρομολόγησης για λειτουργία τοπικά στη συσκευή**. Αυτή η επέκταση σίγουρα χρειάζεται μεγαλύτερη προσπάθεια από της προηγούμενες, αλλά παρουσιάζει και ένα μεγάλο ερευνητικό ενδιαφέρον και θα άξιζε κανείς να κάνει την κατάλληλη έρευνα για τον εντοπισμό τέτοιων προσπαθειών και να συμβάλει σε αυτές με το δικό του τρόπο.

11

Βιβλιογραφία

[1] OpenTripPlanner, «Homepage,» [Ηλεκτρονικό]. Available: http://opentripplanner.com/learn.

[2] Google Developers, «GTFS Review & Documentation,» [Ηλεκτρονικό]. Available: https://developers.google.com/transit/gtfs/.

[3] Wikipedia, "Dijkstra's algorithm," [Online]. Available: http://en.wikipedia.org/wiki/Dijkstra's_algorithm.

[4] Wikipedia, "A* search_algorithm," [Online]. Available: http://en.wikipedia.org/wiki/A*_search_algorithm.

[5] OpenTripPlanner, "GraphStructure," [Online]. Available: https://github.com/openplans/OpenTripPlanner/wiki/GraphStructure.

[6] Wikipedia, «Google Transit,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Google_Maps#Google_Transit.

[7] Google, «Google Transit,» [Ηλεκτρονικό]. Available: http://www.google.com/transit.

[8] ΙΠΣΥΠ, «Open Trip Planner,» [Ηλεκτρονικό]. Available: http://geolinux.imis.athena-innovation.gr/opentripplanner-webapp/. [Πρόσβαση October 2012].

[9] Opti-Trans, «Opti-Trans,» [Ηλεκτρονικό]. Available: http://www.optitrans-fp7.eu.

[10] «Transportation in Athens,» [Ηλεκτρονικό]. Available:

https://market.android.com/details?id=com.darkpain.athenstransportation.

[11] «OpenMBTA,» [Ηλεκτρονικό]. Available: http://openmbta.org/.

[12] TriMet, «TriMet Homepage,» [Ηλεκτρονικό]. Available: http://trimet.org/.

[13] OpenStreetMap, "Homepage," [Online]. Available: http://www.openstreetmap.org/.

[14] Wikipedia, «OpenStreetMap,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/OpenStreetMap.

[15] OpenStreetMap, "Stats," [Online]. Available: http://wiki.openstreetmap.org/wiki/Stats.

[16] OpenStreetMap, «Planet.osm,» [Ηλεκτρονικό]. Available: http://wiki.openstreetmap.org/wiki/Extract.

[17] Osmosis, «Osmosis,» [Ηλεκτρονικό]. Available: http://wiki.openstreetmap.org/wiki/Osmosis.

[18] Mapsforge, «Mapsforge Homepage,» [Ηλεκτρονικό]. Available: http://code.google.com/p/mapsforge/.

[19] Wikipedia, «Android OS,» [Ηλεκτρονικό]. Available: http://en.wikipedia.org/wiki/Android_(operating_system).

[20] Android, "Activities," [Online]. Available: http://developer.android.com/guide/components/activities.html.

[21] Mapsforge, "Mapsforge Javadoc," [Online]. Available: http://mapsforge.googlecode.com/svn/tags/0.3.0/javadoc/org/mapsforge/android/maps/MapActivity.html.

[22] Mapsforge, "Mapsforge Javadoc," [Online]. Available: http://mapsforge.googlecode.com/svn/tags/0.3.0/javadoc/org/mapsforge/android/maps/MapView.html.

[23] OpenTripPlanner, "API - REST - plan," [Online]. Available: http://www.opentripplanner.org/apidoc/rest.plan.html.

[24] Mapsforge, «Map File Writer Documentation,» [Ηλεκτρονικό]. Available: http://code.google.com/p/mapsforge/wiki/MapFileWriterOsmosis.

[25] Android, "Android SDK," [Online]. Available: http://developer.android.com/sdk/.

[26] Android, "ADT," [Online]. Available: http://developer.android.com/tools/help/adt.html.

[27] Android, "Design," [Online]. Available: http://developer.android.com/design/index.html.

[28] K. Grajek, "Android Series: Download files with Progress Dialog," [Online]. Available: http://www.softwarepassion.com/android-series-download-files-with-progress-dialog/.

[29] PhotoGeolocator, "PhotoGeolocator," [Online]. Available: http://code.google.com/p/photogeolocator/.

[30] Android, [Ηλεκτρονικό]. Available: http://developer.android.com.

[31] S. Winter, "Modeling Costs of Turns in Route Planning," Institute for Geoinformation, Technical University Vienna, Gusshausstrasse 27-29, 1040, Vienna, 2002.

12

ΠΑΡΑΡΤΗΜΑ:

Κλάσεις εφαρμογής

12.1 Κλάσεις με γραφικό περιβάλλον (GUI)

Σε αυτή την παράγραφο θα παρουσιαστούν οι κλάσεις που χρησιμοποιούνται για τη δημιουργία γραφικού περιβάλλοντος και την αλληλεπίδραση με το χρήστη. Οι περισσότερες κλάσεις αυτής τη κατηγορίας επεκτείνουν τη βασική κλάση **Activity** ή υποκλάσεις της, όπως **ListActivity** και **TabActivity**. Όπως αναφέρθηκε και στην παράγραφο **7.2.1.1** η κλάση αυτή διαθέτει αρκετές callback μεθόδους που καλούνται κατά τον κύκλο ζωής της. Αυτές οι μέθοδοι, κατά κανόνα, υλοποιούνται (αντικαθιστούνται) από τις κλάσεις που την επεκτείνουν προκειμένου να κτίσουν το γραφικό περιβάλλον και να παρέχουν τη ζητούμενη λειτουργικότητα. Κρίνεται σκόπιμο λοιπόν να παρουσιαστούν σε αυτό το σημείο οι βασικές μέθοδοι και να αποφευχθεί η αναφορά τους στις παρακάτω κλάσεις όπου θα εμβαθύνουμε σε πιο ειδικές μεθόδους.

- **onCreate**: καλείται κατά την πρώτη δημιουργία του activity. Εδώ δημιουργείται συνήθως το γραφικό περιβάλλον. Δημιουργούνται Views «γεμίζουν» λίστες κλπ. Ακολουθείται από την onStart.
- **onRestart**: καλείται όταν το activity ξεκινάει ξανά, ενώ έχει σταματήσει στο παρελθόν. Ακολουθείται από την onStart.
- **onStart**: καλείται ακριβώς πριν το activity γίνει ορατό στο χρήστη. Ακολουθείται από την onResume ή την onStop αν το activity οριστεί ως κρυφό.

- **onResume**: καλείται ακριβώς πριν το activity αρχίσει να αλληλεπιδρά με το χρήστη. Σε αυτό το σημείο το activity είναι στην κορυφή της στοίβας των activities. Ακολουθείται από την onPause.
- **onPause**: καλείται όταν το σύστημα ξεκινάει ένα άλλο activity. Τυπικά σε αυτή τη μέθοδο αποθηκεύονται δεδομένα που είναι υπό επεξεργασία, σταματάνε διαδικασίες που καταναλώνουν μπαταρία όπως εντοπισμός θέσης ή animations κ.α.. Ακολουθείται από την onResume, αν το activity επανέλθει στην οθόνη, ή την onStop, αν γίνει αόρατο στο χρήστη.
- **onStop**: καλείται όταν το activity δεν είναι πια ορατό στο χρήστη. Αυτό μπορεί να συμβεί είτε γιατί τερματίζεται είτε γιατί ένα άλλο activity το υπερκαλύπτει. Ακολουθείται είτε από την onRestart, αν επανέρθει, ή από την onDestroy, αν τερματίζεται.
- **onDestroy**: καλείται πριν τερματιστεί το activity και είναι η τελευταία μέθοδος που θα εκτελέσει το συγκεκριμένο activity. Καλείται είτε γιατί κάποιος κάλεσε τη finish() είτε γιατί το σύστημα τερματίζει το activity για να ελευθερώσει χώρο.

Αυτές οι μέθοδοι αφορούν τον κύκλο ζωής ενός activity. Υπάρχουν κι άλλες μέθοδοι που μπορούν να χρησιμοποιηθούν από τον προγραμματιστή για να χτίσει γραφικό περιβάλλον ειδικού σκοπού όπως το menu, τα context menus του activity κ.α.. Το menu αφορά τις επιλογές που έχει ο χρήστης όταν πατήσει το πλήκτρο menu της συσκευής του και εκεί τοποθετούνται λειτουργίες που είναι περιττό να καταλαμβάνουν χώρο στην οθόνη. Τα context menus είναι menu που εμφανίζονται για ένα συγκεκριμένο View της activity και εμφανίζονται στο χρήστη μετά από παρατεταμένο πάτημα πάνω στο View. Μερικά παραδείγματα τέτοιων μεθόδων είναι:

- **onCreateOptionsMenu**: καλείται κατά τη δημιουργία του menu του activity.
- **onMenuItemsSelected**: καλείται κατά την επιλογής ενός στοιχείου του menu. Εδώ προστίθεται η λειτουργικότητα των επιλογών του menu.
- **onCreateContextMenu**: καλείται κατά τη δημιουργία του context menu κάθε view.
- **onContextItemSelected**: καλείται κατά την επιλογή ενός στοιχείου ενός context menu.
- **onCreateDialog**: καλείται όταν δημιουργείται ένας διάλογος (dialog) για την επικοινωνία με το χρήστη. Ένα dialog μπορεί να είναι μία λίστα επιλογών, η επιλογή ώρα και ημερομηνίας κ.α.. Σε αυτή τη μέθοδο δημιουργούνται όλοι οι διάλογοι κατά περίπτωση.

Αυτές οι μέθοδοι χρησιμοποιούνται κατά κόρον στις κλάσεις που θα παρουσιαστούν παρακάτω, ενώ υπάρχουν και άλλες, παρόμοιας χρηστικότητας, μέθοδοι που χρησιμοποιούνται λιγότερο. Υπάρχουν ακόμη κάποιες μέθοδοι που δεν αντικαθιστούνται, αλλά χρησιμοποιούνται ως έχουν για την υλοποίηση του γραφικού περιβάλλοντος. Μία από αυτές είναι η setListAdapter που καλείται σε κάθε ListActivity για να «δέσει» τη λίστα του με έναν adapter όπως για παράδειγμα ένα στιγμιότυπο του SimpleCursorAdapter.

Στις επόμενες παραγράφους θα περιγραφούν οι όλες οι κλάσεις που δημιουργούν γραφικό περιβάλλον και οι ειδικές μέθοδοί τους, χωρίς αυτές που περιγράψαμε εδώ.

12.1.1 Main

Είναι η κύρια κλάση της εφαρμογής. Υλοποιεί την κεντρική οθόνη που βλέπει ο χρήστης όταν εκκινεί την εφαρμογή. Επεκτείνει την βασική κλάση του Android, για κλάσεις που αφορούν GUI, που είναι η **Activity.**

Μέσα από την οθόνη της ο χρήστης μπορεί να εκκινήσει μία διαδικασία δρομολόγησης επιλέγοντας σημείο εκκίνησης και προορισμού, αλλά και ημερομηνία και ώρα που επιθυμεί να ταξιδέψει. Μπορεί επίσης να εκκινήσει άλλες Activities της εφαρμογής, όπως το χάρτη, τη λίστα με τις διαθέσιμες διαδρομές και τη λίστα με τις αποθηκευμένες δρομολογήσεις.

Όπως όλες οι Activities της κατηγορίας, μπορεί να εκκινήσει την καρτέλα ρυθμίσεων και την αναζήτηση διαδρομών/στάσεων.

Μέθοδοι:

- **updateTimeButton**: ενημερώνει το κείμενο του κουμπιού που εκκινεί τον διάλογο επιλογής ώρας με την τρέχουσα επιλεγμένη ώρα.
- **updateDateButton**: ενημερώνει το κείμενο του κουμπιού που εκκινεί τον διάλογο επιλογής ημερομηνίας με την τρέχουσα επιλεγμένη ημερομηνία.
- **getPointGoogleGeocodingAPI**: μετατρέπει μία διεύθυνση από μορφή κειμένου σε ένα αντικείμενο τύπου **RoutinPoint** με όνομα και συντεταγμένες, χρησιμοποιώντας το Google Geocoding API.
- **startListeningForLocation**: ξεκινά τη διαδικασία εύρεσης της θέσης του χρήστη και της παρακολούθησης για αλλαγές σε αυτή.
- **stopListeningForLocation**: σταματάει την παραπάνω διαδικασία.

Εσωτερικές κλάσεις:

- **ResolveAddressesAndStart**: επεξεργάζεται τις διευθύνσεις που εισήγαγε ο χρήστης για τη δρομολόγηση. Μετατρέπει τα δεδομένα σε κατάλληλη μορφή, χρησιμοποιώντας και την **getPointGoogleGeocodingAPI** και εκκινεί την κλάση Routing με την κατάλληλη είσοδο.

12.1.2 EditPreferences

Η κλάση αυτή επεκτείνει την κλάση **PreferencesActivity** του Android που επιτρέπει την εύκολη δημιουργία γραφικού περιβάλλοντος για επεξεργασία των ρυθμίσεων της εφαρμογής. Αυτό επιτυγχάνεται συνδέοντας την κλάση με ένα xml αρχείο ρυθμίσεων που έχουμε προετοιμάσει και δηλώνει όλες τις ρυθμίσεις που είναι διαθέσιμες για την εφαρμογή μας.

Εκτός από το βασικό της σκοπό, η κλάση αυτή υλοποιεί και μία επιπλέον δυνατότητα της εφαρμογής που είναι το κατέβασμα του αρχείου του χάρτη που απαιτείται για την χρήση χωρίς την απαίτηση σύνδεσης στο διαδίκτυο. Αυτό γίνεται παρέχοντας δύο κουμπιά στο χρήστη, ένα για κατέβασμα και ένα για διαγραφή του χάρτη αν δεν χρησιμοποιείται. Περιλαμβάνονται και οι κατάλληλοι διάλογοι ενημέρωσης ή επιβεβαίωσης ανάλογα με τις ενέργειες του χρήστη.

Μέθοδοι:

- **checkMapFileAndDownload**: ελέγχει αν ο χάρτης υπάρχει και ενημερώνει τον χρήστη κατάλληλα. Αν ο χάρτης δεν υπάρχει ξεκινά το κατέβασμά του καλώντας την startDownload.
- **startDownload**: ξεκινά το κατέβασμα του χάρτη θέτοντας το URL και δίνοντάς το ως παράμετρο στην εκκίνηση της DownloadFileAsync.

Εσωτερικές κλάσεις:

- **DownloadFileAsync**: επεκτείνει την AsyncTask και επιτελεί το έργο του κατεβάσματος ενός αρχείου, σε ξεχωριστό thread, δείχνοντας παράλληλα την εξέλιξη της διαδικασίας (ποσοστό κατεβάσματος) σε ένα progress dialog στο κυρίως thread της Activity. Η κλάση αυτή γράφτηκε από τον Krzysztof Grajek και ανασύρθηκε από το site softwarepassion.com [28]

12.1.3 Map

Επεκτείνει την κλάση **MapActivity** της βιβλιοθήκης mapsforge και αναλαμβάνει όλη τη λειτουργικότητα της εφαρμογής που αφορά χάρτες. Με βάση την είσοδό της κατά την εκκίνηση υλοποιεί διαφορετικές λειτουργίες, παραθέτοντας στο χάρτη ότι χρειάζεται κάθε φορά. Πολλά κομμάτια της κλάσης αυτής είναι εμπνευσμένα από την εφαρμογή επίδειξης της βιβλιοθήκης mapsforge, που ονομάζεται "AdvancedMapViewer".

Όταν ενεργοποιείται χωρίς κάποια είσοδο, δίνει τη δυνατότητα στο χρήστη να εντοπίσει κοντινές στάσεις γύρω από ένα σημείο και να επιλέξει σημείο εκκίνησης και προορισμού για να εκτελέσει μία αναζήτηση βέλτιστης διαδρομής.

Μέθοδοι:

- **enableFollowGPS**: ενεργοποιεί την κατάσταση "Follow GPS" όπου η θέση του χρήστη εμφανίζεται συνεχώς στο χάρτη. Προσθέτει ένα εικονίδιο στην οθόνη που δείχνει την κατάσταση του σήματος GPS και επιτρέπει στο χρήστη να απενεργοποιήσει την κατάσταση "Follow GPS".
- **disableFollowGPS**: απενεργοποιεί την κατάσταση "Follow GPS" και αφαιρεί το εικονίδιο το GPS από την οθόνη.
- **addRoutingTrips**: παραθέτει στο χάρτη μία σειρά από «ταξίδια» που συνθέτουν μία δρομολόγηση όπως αυτή προέκυψε από το σύστημα δρομολόγησης. Δέχεται ως είσοδο μία λίστα από αντικείμενα τύπου RoutringTrip.
- **decodePoly**: αποκωδικοποιεί μια συμβολοσειρά τύπου encoded polyline σε μία λίστα από αντικείμενα τύπου GeoPoint της βιβλιοθήκης mapsforge.

Εσωτερικές κλάσεις:

- **showTripTask**: επεκτείνει την AsyncTask και αναλαμβάνει να επιδείξει στο χάρτη ένα συγκεκριμένο δρομολόγιο μιας διαδρομής. Δέχεται ως είσοδο το αναγνωριστικό μιας διαδρομής και την κατεύθυνση του δρομολογίου

12.1.4 MapWrapper

Η συγκεκριμένη κλάση επεκτείνει την κλάση **Activity** και χρησιμοποιείται απλώς ως ενδιάμεσος στην εκκίνηση της κλάσης Map. Αυτό χρειάζεται προκειμένου να γίνει δυνατό να αφαιρεθεί την κλάση Map από την στοίβα των Activities του Android και να αποφευχθούν προβλήματα που αφορούν στη διαθέσιμη μνήμη της συσκευής. Χρησιμοποιώντας αυτή την ενδιάμεση κλάση είναι δυνατόν να κρατηθεί η κατάσταση της κλάσης Map ή η είσοδός της και να κληθεί ξανά όταν είναι απαραίτητο.

Μέθοδοι:

- **onActivityResult**: πρόκειται για μέθοδο που κληρονομείται από την κλάση Activity και χρησιμοποιείται για να μας επιτρέψει να ξέρουμε τον τρόπο με τον οποίο τερμάτισε η κλάση Map, η οποία εκκίνησε μέσα από την MapWrapper. Ανάλογα με το αποτέλεσμα η MapWrapper τερματίζει η επανεκκινεί την Map.

12.1.5 Route

Η κλάση Route επεκτείνει την κλάση του Android **ListActivity** και αναλαμβάνει να δείξει στην οθόνη της λεπτομέρειες μιας συγκεκριμένης διαδρομής (route). Παρουσιάζει μία λίστα με τις στάσεις από όπου περνάει η διαδρομή, με τη σειρά με την οποία τις επισκέπτεται, ενώ ταυτόχρονα εμφανίζει και άλλες λεπτομέρειες της διαδρομής, όπως τον τίτλο της, τις ώρες πρώτου και τελευταίου δρομολογίου και τη συχνότητα της. Μέσα από το menu της δίνεται η δυνατότητα αλλαγής κατεύθυνσης (εκτός αν είναι κυκλική διαδρομή), εμφάνισής της στο χάρτη, αλλά και προσθήκης της στα (ή διαγραφή από τα) «Αγαπημένα». Τέλος πατώντας σε μία στάση εκκινεί την κλάση Stop που δείχνει τις λεπτομέρειες τις στάσης.

Μέθοδοι:

- **fillData**: βρίσκει, στη βάση δεδομένων, τις στάσεις τις οποίες επισκέπτεται η συγκεκριμένη διαδρομή με βάση την κατεύθυνσή της και ενημερώνει κατάλληλα της λίστα της Activity.
- **fillRouteData**: ενημερώνει, από τη βάση δεδομένων, τα υπόλοιπα στοιχεία της διαδρομής, όπως τις ώρες πρώτου και τελευταίου δρομολογίου, τη συχνότητα των δρομολογίων και το όνομα της διαδρομής.

12.1.6 Routes

Η κλάση αυτή είναι μια επέκταση της κλάσης **TabActivity** που αναλαμβάνει να φιλοξενήσει διαφορετικά Activities σε διαφορετικές καρτέλες (tabs). Η συγκεκριμένη υλοποίησή της περιλαμβάνει της κλάσεις RoutesAll και RoutesFavorites που η καθεμία δείχνει μια λίστα με διαδρομές. Η πρώτη αφορά όλες τις διαδρομές που υπάρχουν στη βάση δεδομένων και η δεύτερη μόνο αυτές που ο χρήστης έχει αποθηκεύσει ως αγαπημένες.

Η μόνη μέθοδος που υλοποιεί αυτή η κλάση είναι η **onCreate** που κληρονομείται από την υπερκλάση Activity και είναι η πρώτη μέθοδος που εκτελείται κατά τη δημιουργία του Activity.

12.1.7 RoutesAll

Επεκτείνοντας την κλάση **ListActivity**, η RoutesAll αναλαμβάνει την επίδειξη μιας λίστας με όλες τι διαθέσιμες διαδρομές. Κάθε γραμμή αποτελεί μία διαδρομή δείχνοντας, με τη μορφή εικονιδίου, τον τύπο και το χρώμα της, ακολουθούμενο από το όνομα της. Επιλέγοντας μία διαδρομή γίνεται εκκίνηση της κλάσης Route για την επίδειξη των λεπτομερειών της, ενώ με long click πάνω της δίνεται η δυνατότητα προσθήκης της στα (ή αφαίρεσή της από τα) «Αγαπημένα». Μέσα από το μενού δίνεται η δυνατότητα εφαρμογής φίλτρου στις διαδρομές που φαίνονται στην οθόνη με βάση την εταιρία που εκτελεί τη διαδρομή (π.χ. ΕΘΕΛ, ΗΣΑΠ κλπ.)

Μέθοδοι:

- **fillData**: επικοινωνεί με τη βάση δεδομένων και γεμίζει τη λίστα της Activity με όλες τις διαθέσιμες διαδρομές φιλτραρισμένες με βάση τις εταιρίες που έχει επιλέξει ο χρήστης.

12.1.8 RoutesFavorites

Όπως και η RoutesAll, η RoutesFavorites επεκτείνει την κλάση ListActivity με τη διαφορά ότι αυτή αναλαμβάνει την επίδειξη σε λίστα μόνο των διαδρομών που ο χρήστης έχει αποθηκεύσει ως «αγαπημένες». Στη συγκεκριμένη δεν υπάρχει δυνατότητα «φιλτραρίσματος» αφού οι διαδρομές που παρουσιάζονται εδώ αποτελούν ήδη επιλογές του χρήστη. Το κλικ (και το παρατεταμένο κλικ) πάνω σε μία διαδρομή από τη λίστα αντιμετωπίζεται όπως ακριβώς στην κλάση RoutesAll.

Μέθοδοι:

- **fillData**: επικοινωνεί με τη βάση δεδομένων και συμπληρώνει τη λίστα με όλες τις «αγαπημένες» διαδρομές του χρήστη

12.1.9 Search

Η Search αναλαμβάνει την εκτέλεση αναζητήσεων για διαδρομές και στάσεις και την παρουσίαση των αποτελεσμάτων τους. Επεκτείνει την κλάση **TabActivity** υλοποιώντας δύο καρτέλες, μία για αποτελέσματα στην αναζήτηση διαδρομών και μία για τα αποτελέσματα στην αναζήτηση στάσεων.

Η Search ορίζεται στο manifest αρχείο του project ως **singleTop** activity. Αυτό σημαίνει ότι ένα καινούργιο στιγμιότυπο της Search δημιουργείται όταν κληθεί από άλλο activity, αλλά

όχι αν η Search υπάρχει ήδη στο προσκήνιο. Έτσι το ίδιο στιγμιότυπο χρησιμοποιείται για συνεχόμενες αναζητήσεις.

Μέθοδοι:

- **handleIntent**: διαχειρίζεται τις καινούργιες αιτήσεις για αναζήτηση. Χρησιμοποιεί την κλάση **MySuggestionProvider** για να αποθηκεύσει την αναζήτηση στο ιστορικό και στη συνέχει εκτελεί την αναζήτηση καλώντας την **fillData.**
- **fillData**: επικοινωνεί με τη βάση δεδομένων για την εκτέλεση την αναζήτησης και στη συνέχει γεμίζει τις λίστες των αποτελεσμάτων (διαδρομές και στάσεις) με τα αντίστοιχα αποτελέσματα.

12.1.10Stop

Η κλάση αυτή αναλαμβάνει, επεκτείνοντας την **ListActivity**, να αναδείξει τις λεπτομέρειες μια στάσης με έμφαση στις διαδρομές που διέρχονται από αυτή. Αυτές οι διαδρομές αποτελούν και την κύρια λίστα της Activity αυτής.

Μέθοδοι:

- **fillData**: επικοινωνεί με τη βάση δεδομένων για να εντοπίσει τις διαδρομές που διέρχονται από αυτή τη στάση και γεμίζει τη λίστα με αυτές (παρόμοια με τις κλάσεις RoutesAll και RoutesFavorites).
- **fillStopData**: επικοινωνεί με τη βάση δεδομένων για να πάρει περισσότερα στοιχεία για τη στάση, όπως το όνομά της, και να τα προσθέσει στον τίτλο της activity.

12.1.11Routing

Η κλάση αυτή αναλαμβάνει την επίδειξη μιας διαδρομής που προκύπτει ως αποτέλεσμα δρομολόγησης, από τον αλγόριθμο βέλτιστης δρομολόγησης, σε μορφή κειμένου. Επεκτείνει την κλάση του Android **ExpandableListActivity** και αυτό γιατί ουσιαστικά είναι μία λίστα από «ταξίδια» (υποδιαδρομές) που κάθε μία μπορεί να κρύβει μία λίστα από στάσεις (αν πρόκειται για κάποιο μέσο) ή παραπάνω πληροφορίες (αν πρόκειται για περπάτημα).

Η κλάση αυτή, αναλόγως με την είσοδό της, μπορεί να δείξει δρομολογήσεις που έχουν αποθηκευτεί παλιότερα ή να εκτελέσει μία αναζήτηση, με τη βοήθεια της κλάσης **RoutingHelper**, και να δείξει το αποτέλεσμα.

Μέσω του μενού της ο χρήστης μπορεί να αποθηκεύσει την δρομολόγηση για μετέπειτα offline χρήση ή να εκκινήσει το χάρτη για να αποκτήσει μια καλύτερη εικόνα για τη διαδρομή.

Εσωτερικές κλάσεις:

- **RoutingRequest**: επεκτείνει την **AsyncTask** και αναλαμβάνει να εκτελέσει ένα request στο απομακρυσμένο OpenTripPlanner, μέσω της **RoutingHelper**. Καθ' όλη τη διάρκεια του request που συμβαίνει στο background, ο χρήστης βλέπει ένα διάλογο που τον ενημερώνει για τη διαδικασία. Αν η διαδικασία είναι επιτυχής η

ExpandableList γεμίζει μέσω του **RoutingExpandableListAdapter**, ενώ αν είναι ανεπιτυχής ενημερώνεται ο χρήστης.

12.1.12RoutingSaved

Η RoutingSaved είναι μια απλή κλάση που επεκτείνει την **ListActivity** και αναλαμβάνει την επίδειξη μιας λίστας με της δρομολογήσεις που ο χρήστης έχει αποθηκεύσει για offline χρήση. Με την επιλογή μιας αποθηκευμένης δρομολόγησης γίνεται εκκίνηση της κλάσης Routing που αναλαμβάνει την επίδειξη των λεπτομερειών της.

Μέθοδοι:

- **fillData**: επικοινωνεί με τη βάση δεδομένων για να ανακτήσει τις αποθηκευμένες δρομολογήσεις και συμπληρώνει τη λίστα του Activity με τα στοιχεία τους.

12.2 Κλάσεις χωρίς γραφικό περιβάλλον

12.2.1 AddressItem

Η κλάση αυτή υλοποιεί ένα αντικείμενο που περιέχει ένα απλό αντικείμενο Address που περιλαμβάνεται στο Android με σκοπό να συμπεριλάβει μία μέθοδο που μετατρέπει μία τέτοιου τύπου διεύθυνση σε μορφή που μπορεί να διαβαστεί από άνθρωπο.

Η κλάση αυτή αντιγράφηκε από το open source project PhotoGeolocator που μπορεί να βρεθεί στο Google Code [29].

Μέθοδοι:

- **AddressItem**: αποτελεί τον κατασκευαστή της κλάσης και παίρνει ως είσοδο την διεύθυνση σε μορφή αντικειμένου Address.
- **toString**: μετατρέπει τη διεύθυνση σε συμβολοσειρά με τρόπο τέτοιο ώστε να μπορεί να διαβαστεί από άνθρωπο.

12.2.2 AddressLookup

Η κλάση αυτή επεκτείνει την **AsyncTask** και χρησιμοποιείται για να βοηθήσει το χρήστη να πληκτρολογήσει μία διεύθυνση. Αντιστοιχείται σε ένα **AutoCompleteTextView** αντικείμενο και συμπληρώνει τη λίστα των προτάσεων (suggestions) στο χρήστη, καθώς αυτός πληκτρολογεί. Χρησιμοποιεί την κλάση **Geocoder** του Android για να εντοπίσει πιθανές διευθύνσεις και στη συνέχεια την κλάση **AddressItem**, που περιγράφηκε παραπάνω, για να μετατρέψει διευθύνσεις σε κείμενο που διαβάζεται εύκολα από το χρήστη.

Η κλάση αυτή, όπως και η AddressItem πάρθηκε από το open source project PhotoGeolocator, που αναφέρθηκε παραπάνω. Υπέστη μικρές παραμετροποιήσεις για να ταιριάξει στις ανάγκες της εφαρμογής.

Οι μέθοδοι που υλοποιεί ανήκουν στην κλάση AsyncTask και είναι οι **doInBackground** και **onPostExecute**

12.2.3 DbAdapter

Η κλάση αυτή χρησιμοποιείται για τη δημιουργία της βάσης δεδομένων και αναλαμβάνει την επικοινωνία με αυτή, εξυπηρετώντας τις περισσότερες από τις υπηρεσίες που προσφέρει η εφαρμογή. Περιλαμβάνει μεθόδους κατασκευής της βάσης και ανάκτησης πληροφοριών από αυτή.

Μέθοδοι:

- **DbAdapter**: ο κατασκευαστής (constructor) της κλάσης. Παίρνει ως είσοδο το **context** του αντικειμένου που δημιουργεί το στιγμιότυπό της.
- **open:** δημιουργεί ένα αντικείμενο **SQLiteOpenHelper** μέσω εσωτερικής κλάσης και ανοίγει τη βάση μέσω αυτού για γράψιμο και διάβασμα.
- **close**: κλείνει τη βάση δεδομένων.
- **crateDatabase**: δημιουργεί μια κενή βάση δεδομένων και αντιγράφει σε αυτή τη δική μας που έχουμε ενσωματώσει προ-συμπληρωμένη στην εφαρμογή μας. Ακόμη εκτελεί τις δημιουργίες πινάκων που χρησιμοποιούνται επιπλέον από το σύστημα και δεν προ-υπάρχουν στη βάση GTFS.
- **checkDatabase**: ελέγχει την ύπαρξη της βάσης δεδομένων. Χρησιμοποιείται από την crateDatabase προκειμένου να δημιουργήσει την βάση μόνο την πρώτη φορά που θα τρέξει.
- **copyDatabase:** αντιγράφει τη βάση δεδομένων από τα αρχεία που έχουμε συμπεριλάβει στην εφαρμογή στον τοπικό κατάλογο συστήματος της κινητής συσκευής. Χρησιμοποιείται από την createDatabase.
- **fetchAllRoutes:** επιστρέφει όλες τις διαδρομές που υπάρχουν στη βάση δεδομένων.
- **fetchAllTrips:** επιστρέφει όλα τα δρομολόγια που υπάρχουν στη βάση δεδομένων.
- **fetchAllStops:** επιστρέφει όλες τις στάσεις που υπάρχουν στη βάση δεδομένων.
- **fetchRoute:** επιστρέφει μία διαδρομή με βάση το πεδίο **_id**. Παίρνει επίσης ως είσοδο τα πεδία του πίνακα routes τα οποία χρειάζονται. Είναι έτσι ευέλικτη και μπορεί να χρησιμοποιηθεί σε διάφορα σημεία της εφαρμογής χωρίς να δημιουργεί περιττά δεδομένα.
- **fetchTrip:** επιστρέφει ένα δρομολόγιο με βάση το πεδίο **_id**. Ομοίως με την fetchRoute επιλέγει τα πεδία που επιστρέφει με βάση την είσοδο.
- **fetchStop:** επιστρέφει μια στάση με βάση το **_id**. Ομοίως με τις προηγούμενες επιλέγονται τα απαραίτητα πεδία.
- **fetchStopsForRoute:** επιστρέφει τις στάσεις από τις οποίες διέρχεται μία διαδρομή με συγκεκριμένη κατεύθυνση. Οι στάσεις είναι ταξινομημένες με βάση τη σειρά με την οποία η διαδρομή τις επισκέπτεται.
- **fetchRoutesForStop:** επιστρέφει όλες τις διαδρομές που διέρχονται από μία συγκεκριμένη στάση. Η μέθοδος αυτή είναι υπερφορτωμένη, αφού μπορεί να δουλέψει με είσοδο το **_id** ή το **stop_id** της στάσης.
- **fetchRoutesByAgency:** δέχεται ως είσοδο μια λίστα από agencies και επιστρέφει όλες τις διαδρομές που εκτελούνται από αυτές τις εταιρίες.

- **numberOfTripsForRoute:** επιστρέφει το πλήθος των δρομολογίων που υπάρχουν για μία διαδρομή. Μας βοηθάει να αποφανθούμε για το αν μία διαδρομή είναι κυκλική ή όχι.
- **tripFrequencies:** επιστρέφει τη συχνότητα, το πρώτο και το τελευταίο δρομολόγιο μια διαδρομής με συγκεκριμένη κατεύθυνση.
- **routeSearch:** αναλαμβάνει την αναζήτηση διαδρομών με βάση το μικρό ή το μεγαλύτερο όνομά τους ή μέρος αυτού.
- **stopSearch:** αναλαμβάνει την αναζήτηση στάσεων με βάση το όνομά τους ή μέρος αυτού.
- **nearbyStops:** επιστρέφει τις κοντινές στάσεις γύρω από ένα σημείο με συγκεκριμένες συντεταγμένες.
- **typeOfStop:** επιστρέφει τον τύπο μιας στάσης με βάση τον τύπο των διαδρομών που διέρχονται από αυτή.
- **addFavoriteRoute:** προσθέτει μια εγγραφή στον πίνακα favorite_routes με την διαδρομή που της περνάμε ως παράμετρο.
- **deleteFavoriteRoute:** διαγράφει μία διαδρομή από τον πίνακα favorite_routes
- **deleteAllFavoriteRoutes:** διαγράφει όλες τις «αγαπημένες» διαδρομές από τον πίνακα favorite_routes.
- **isFavoriteRoute:** ελέγχει αν μία διαδρομή ανήκει στα «Αγαπημένα».
- **fetchFavoriteRoutes:** επιστρέφει όλες τις διαδρομές που ανήκουν στα «Αγαπημένα».
- **addSavedRouting:** προσθέτει μία καινούργια δρομολόγηση για offline χρήση.
- **deleteSavedRouting:** διαγράφει μία αποθηκευμένη δρομολόγηση από τη βάση.
- **deleteAllSavedRoutings:** διαγράφει όλες τις αποθηκευμένες δρομολογήσεις από τη βάση.
- **fetchAllSavedRoutings:** επιστρέφει όλες τις αποθηκευμένες δρομολογήσεις χωρίς την κυρίως πληροφορία που είναι ή XML περιγραφή της.
- **fetchSavedRouting:** επιστρέφει όλες τις πληροφορίες για μία αποθηκευμένη δρομολόγηση (μαζί με το XML)

Εσωτερικές κλάσεις:

- **DatabaseHelper**: επεκτείνει την **SQLiteOpenHelper** και αναλαμβάνει όλη την επικοινωνία με τη βάση δεδομένων SQLite για λογαριασμό της DbAdapter.

12.2.4 Helper

Η Helper είναι μια κλάση που περιλαμβάνει μεθόδους που είναι χρήσιμες σε διάφορα σημεία τις εφαρμογής.

Μέθοδοι:

- **getImageResource:** επιστρέφει το εικονίδιο που χρειάζεται για μια διαδρομή με βάση κάποιες παραμέτρους. Αυτές είναι η χρήση του εικονιδίου (χάρτης ή λίστα) και ο τύπος της διαδρομής. Η μέθοδος είναι υπερφορτωμένη γιατί σε κάποιες περιπτώσεις (δρομολόγηση) χρειαζόμαστε και εικονίδιο για περπάτημα, οπότε προστίθεται ακόμα μία παράμετρος.

- **getTripColor:** επιστρέφει το χρώμα ενός ταξιδιού. Αυτή η μέθοδος αφορά κυρίως τη δρομολόγηση και επιστρέφει χρώμα ανάλογα με την εταιρία του ταξιδιού ή μαύρο αν πρόκειται για ταξίδι περπατήματος.

12.2.5 MySuggestionProvider

Πρόκειται για τον Suggestion Provider της λειτουργίας Search της εφαρμογής. Χρησιμοποιείται για να προτείνει στο χρήστη πιθανές λέξεις κλειδιά που προσπαθεί να πληκτρολογήσει με βάση το ιστορικό αναζητήσεων. Επεκτείνει την κλάση **SearchRecentSuggestionsProvider** του Android.

12.2.6 RoutesSimpleCursorAdapter

Η κλάση αυτή επεκτείνει την κλάση **SimpleCursorAdapter** του Android και είναι υπεύθυνη για την συμπλήρωση λιστών που αφορούν διαδρομές (routes) όπως στις κλάσεις RoutesAll και RoutesFavorites, αλλά και σε κάποιους διαλόγους και σε άλλα σημεία.

Αντικαθιστώντας την μέθοδο **getView** γίνεται δυνατόν να προστεθεί ένα εικονίδιο που δείχνει τον τύπο της στάσης. Τα υπόλοιπα στοιχεία παρουσιάζονται όπως προκύπτουν από τη βάση δεδομένων και περιλαμβάνουν, εκτός από την ονομασία της διαδρομής, ο αριθμός της, αλλά και το χρώμα της προκειμένου να γίνει πιο εύκολα κατανοητός ο τύπος της.

12.2.7 RoutingPoint

Το **RoutingPoint** είναι ένα αντικείμενο που παίζει το ρόλο της διεύθυνσης, αλλά προσαρμοσμένο στις ανάγκες της εφαρμογής για το υποσύστημα της δρομολόγησης.

Περιλαμβάνει τα πεδία:

- **name:** το όνομα του σημείου
- **stopId:** το stop_id της στάσης
- **lat:** το γεωγραφικό πλάτος του σημείου
- **lon:** το γεωγραφικό μήκος του σημείου

Οι μέθοδοι που περιλαμβάνει είναι απλοί Getters και Setters των παραπάνω πεδίων και είναι οι **getName**, **setName**, **getStopId**, **setStopId**, **getLat**, **setLat**, **getLon**, **setLon**.

12.2.8 RoutingTrip

Είναι ένα αντικείμενο που σκοπό έχει να περιγράψει ένα υποταξίδι ενός ταξιδιού που προκύπτει ως αποτέλεσμα δρομολόγησης.

Περιλαμβάνει τα πεδία:

- **mode:** ο τρόπος ταξιδιού (π.χ. περπάτημα ή λεωφορείο)
- **route:** η διαδρομή (αν δε πρόκειται για περπάτημα)
- **headsign:** ο τίτλος της διαδρομής

- **agencyId:** η εταιρία που εκτελεί το δρομολόγιο (αν δε πρόκειται για περπάτημα)
- **duration:** η διάρκεια του ταξιδιού
- **distnance:** η απόσταση που καλύπτεται κατά το ταξίδι αυτό
- **from:** ένα αντικείμενο τύπου RoutingPoint που δείχνει την αφετηρία του ταξιδιού
- **to:** ένα αντικείμενο τύπου RoutingPoint που δείχνει τον προορισμό του ταξιδιού
- **intermediateStops:** μία λίστα από αντικείμενα τύπου RoutingPoint που δείχνουν τις ενδιάμεσες στάσεις
- **polyline:** η περιγραφή του σχήματος της διαδρομής σε μορφή encoded polyline

Οι μέθοδοι της κλάσης περιλαμβάνουν Getters και Setters για τα παραπάνω πεδία.

12.2.9 RoutingHandler

Πρόκειται για μια κλάση που επεκτείνει τον **DefaultHandler** του SAXParser και αναλαμβάνει το parsing της XML απάντησης του OpenTripPlanner και τη δημιουργία μιας λίστας από αντικείμενα RoutingTrip ως αποτέλεσμα.

Αντικαθιστά τις βασικές μεθόδους της DefaultHandler που είναι οι **startDocument**, **endDocument**, **startElement**, **endElement** και **characters**, ενώ περιλαμβάνει και τη μέθοδο **getParsedData** που επιστρέφει τα δεδομένα σε μορφή λίστας RoutinTrip.

12.2.10RoutingHelper

Η RoutingHelper είναι μια βοηθητική κλάση που μπορεί να προετοιμάσει και να εκτελέσει ένα Request στο OpenTripPlanner, αλλά και να επεξεργαστεί, με τη βοήθεια του RoutingHandler, το Response ενός τέτοιου Request.

Μέθοδοι:

- **openTripPlannerRequest:** προετοιμάζει και στέλνει ένα Request στον OpenTripPlanner Server με βάση την είσοδο της. Στη συνέχει επιστρέφει το Response.
- **xmlParse:** χρησιμοποιεί την RoutingHandler για να επεξεργαστεί ένα XML Response από το OpenTripPlanner και επιστρέφει το στιγμιότυπο του RoutingHandler που κρατάει τα αποτελέσματα.

12.2.11RoutingExpandableListAdapter

Επεκτείνει την **BaseExpandableListAdapter** του Android και χρησιμοποιείται για να συμπληρώσει την ExpandableList της κλάσης Routing με δεδομένα δρομολόγησης που βρίσκονται σε λίστα αντικειμένων RoutingTrip.

Αντικαθιστώντας τις βασικές μεθόδους της BaseExpandableListAdapter, δημιουργεί μια λίστα με χαρακτηριστικά εικονίδια και τα βασικά στοιχεία ενός υποταξιδιού, ενώ περιλαμβάνει και τις ενδιάμεσες στάσεις ενός υποταξιδιού σε μορφή υπολίστας όταν αυτό αφορά ΜΜΜ και όχι περπάτημα.

12.2.12StopsItemizedOverlay

Είναι μια επέκταση της κλάσης ItemizedOverlay της βιβλιοθήκης mapsforge και αφορά την περίπτωση που θέλουμε να προσθέσουμε στάσεις πάνω στο χάρτη. Επεκτείνει τις δυνατότητες της ItemizedOverlay προσθέτοντας την δυνατότητα να επιστρέψει το κέντρο των σημείων που περιλαμβάνει, προκειμένου να μπορεί κανείς να «κεντράρει» το χάρτη ώστε να δείχνει όλες τις στάσεις. Η πιο σημαντική επέκταση όμως είναι αυτή της επίδειξης των διερχόμενων γραμμών από κάθε στάση, κάτι το οποίο συμβαίνει όταν ο χρήστης κάνει «κλικ» πάνω σε μία στάση στο χάρτη.

Μέθοδοι:

- **getCenter**: επιστρέφει το κέντρο όλων των στάσεων που περιλαμβάνονται στο Overlay.
- **updateZoomLevel**: ανανεώνει το τωρινό επίπεδο zoom του χάρτη προκειμένου να χρησιμοποιηθεί σε βοηθητικές μεθόδους που αποτρέπουν λάθος πατήματα πάνω σε στάσεις κατά το zooming με δύο δάχτυλα.
- Αντικαθιστά τις περισσότερες βασικές μεθόδους της ItemizedOverlay όπως **addItem**, **addItems**, **clear**, **removeItem**, **onTap** κ.α.

Εσωτερικές κλάσεις:

- **PopulateRoutesDialog**: αναλαμβάνει την συμπλήρωση του διαλόγου, που ανοίγει μετά από κλικ πάνω σε στάση, με τις διερχόμενες διαδρομές. Επεκτείνει την **AsyncTask** προκειμένου να συμβεί στο background και να ενημερωθεί ο χρήστης για τη διαδικασία.

12.2.13UserItemizedOverlay

Όπως και η προηγούμενη κλάση, έτσι και αυτή επεκτείνει την ItemizedOverlay. Χρησιμοποιείται για την περίπτωση που ο χρήστης ανοίγει το Activity του χάρτη χωρίς σκοπό να παρουσιάσει κάποια διαδρομή και αφορά το στρώμα που αναλαμβάνει την αλληλεπίδραση με το χρήστη. Αυτό το Overlay υπάρχει by default στην κλάση Map, αλλά αφαιρείται όταν πρόκειται να χρησιμοποιηθεί ο χάρτης για να παρουσιάσει κάποια διαδρομή ή δρομολόγηση.

Δίνει τη δυνατότητα στο χάρτη να επιλέξει ένα σημείο πάνω στο χάρτη και να το χρησιμοποιήσει με διαφορετικούς τρόπους, όπως να εντοπίσει κοντινές στάσεις ή να επιλέξει σημεία αφετηρίας και προορισμού για εύρεση βέλτιστης διαδρομής.

Αντικαθιστά τις περισσότερες βασικές μεθόδους της ItemizedOverlay και προσθέτει κάποιες ακόμη χρήσιμες σε αυτή. Ακολουθούν οι άξιες αναφοράς μέθοδοι:

- **onLongPress**: όταν ο χρήστη κάνει ένα παρατεταμένο κλικ πάνω στο χάρτη, προστίθεται ένα σημάδι στο σημείο που άγγιξε. Αυτό αποτελεί το τωρινό επιλεγμένο σημείο.

- **onTap**: όταν ο χρήστης κάνει κλικ πάνω σε αυτό το επιλεγμένο σημείο, εμφανίζεται ένας διάλογος που του δίνει τρεις επιλογές: κοντινές στάσεις, αφετηρία δρομολόγησης και προορισμός δρομολόγησης. Ανάλογα με την επιλογή του, προστίθεται στο χάρτη ένα StopsItemizedOverlay με τις κοντινές στάσεις ή το επιλεγμένο σημείο, αλλάζει σε σημείο αφετηρίας ή προορισμού.
- **startRouting**: καλείται στην περίπτωση που ο χρήστης έχει επιλέξει σημείο αφετηρίας και σημείο τερματισμού. Προετοιμάζει και δημιουργεί ένα νέο στιγμιότυπο της κλάσης Routing που αναλαμβάνει να βρει τη βέλτιστη διαδρομή μεταξύ των δύο σημείων.
- **clearOverlays**: χρησιμοποιείται εσωτερικά για τον καθαρισμό όλων των δευτερευόντων Overlays που χρησιμοποιεί η κλάση.
- **updateZoomLevel**: ανανεώνει το επίπεδο zoom του Overlay που αναλαμβάνει την επίδειξη των κοντινών στάσεων.

12.2.14Utils

Πρόκειται για μια κλάση που περιλαμβάνει στατικές μεθόδους που μπορούν να φανούν χρήσιμες σε διάφορα σημεία της εφαρμογής.

Μέθοδοι:

- **pad**: μετατρέπει έναν ακέραιο σε συμβολοσειρά κάνοντας το διψήφιο, σε περίπτωση που είναι μονοψήφιο, προσθέτοντας ένα «0» στην αρχή.

invalidateOptsMenu: χρησιμοποιείται με είσοδο ένα Activity, προκειμένου να προκαλέσει την ανανέωση του menu της.