Ατομική Διπλωματική Εργασία

ΒΕΛΤΙΣΤΗ ΧΡΗΣΗ ΑΣΤΙΚΩΝ ΣΥΓΚΟΙΝΩΝΙΩΝ ΜΕ ΧΡΗΣΗ ΚΙΝΗΤΗΣ ΠΛΑΤΦΟΡΜΑΣ ANDROID

Φαίδων Ρούσου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2012

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Βέλτιστη Χρήση Αστικών Συγκοινωνιών Με Χρήση Κινητής Πλατφόρμας Android

Φαίδων Ρούσου

Επιβλέπων Καθηγητής
Γιώργος Πάλλης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2012

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου λέκτορα κ. Γιώργο Πάλλη, ο οποίος υπήρξε ο επιβλέποντας καθηγητής της διπλωματικής εργασίας, για την συνεχή υποστήριξη και καθοδήγηση που μου παρείχε καθ' όλη την διάρκεια της εκπόνησης της διπλωματικής εργασίας. Συγκεκριμένα θα ήθελα να τον ευχαριστήσω για τον χρόνο που διέθεσε στις διάφορες συναντήσεις που προγραμματίσαμε ούτως ώστε να γίνει ο έλεγχος της προόδου μου και για την ειλικρινή ανατροφοδότηση που μου έδινε. Τον ευχαριστώ για την κατανόηση που έδειξε όταν υπήρχαν διάφορες δυσκολίες και για τις συμβουλές του για τα διάφορα προβλήματα που προέκυπταν. Είχαμε μια εξαιρετική συνεργασία καθ' όλη την διάρκεια του χρόνου. Τέλος, θα ήθελα να τον ευχαριστήσω για την ευκαιρία που μου έδωσε να δουλέψω σε ένα ερευνητικό εργαστήριο και που μου προσέφερε αυτή διευκόλυνση στην διεκπεραίωσης της διπλωματικής μου εργασίας.

Θα ήθελα επίσης να ευχαριστήσω τα άτομα από το ερευνητικό εργαστήριο LINC για τη βοήθεια που μου παρείχαν κατά τα διάφορα στάδια της διπλωματικής μου εργασίας. Συγκεκριμένα θα ήθελα να ευχαριστήσω τον Νικόλα Λουλούδη, υποψήφιο διδάκτορα, για τις μακρές συζητήσεις που είχαμε οι οποίες αποδείχτηκαν πολύτιμες καθώς και την Andoena Balla για την άμεση ενέργειά της σχετικά με οποιαδήποτε ανάγκη προέκυπτε σχετικά με τον εξυπηρετητή. Ευχαριστώ επίσης τον φίλο και συμφοιτητή Γιώργο Χριστοδούλου για τις καρποφόρες συζητήσεις που κάναμε καθ' όλη την διάρκεια του χρόνου.

Θα ήθελα να ευχαριστήσω την εταιρεία GEOMATIC Technologies για την βοήθεια που μου παρείχε δίνοντας μου ένα δείγμα από τα GTFS αρχεία για την περιοχή της Λευκωσίας. Χωρίς αυτά δεν θα ήταν εφικτή η δοκιμή και αξιολόγηση της εφαρμογής.

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου για την υποστήριξή και την κατανόηση που έδειξαν κατά την διάρκεια αυτής της δύσκολης περιόδου.

Περίληψη

Τα δημόσια μέσα μεταφοράς διαδραματίζουν σημαντικό ρόλο σε ανθρώπους που αναζητούν λύσεις για να διευκολύνουν τον καθημερινό τρόπο μετακίνησής τους, την ανεξαρτικοποίηση τους από τα ιδιωτικά τους αυτοκίνητα ή ίσως την ελαχιστοποίηση της επίπτωσης που προκαλούν στον περιβαλλοντικό τομέα. Επιπλέον, βοηθούν στην καταπολέμηση της κυκλοφοριακής συμφόρησης, τη μείωση των εκπομπών του διοξειδίου του άνθρακα και την προώθηση βιώσιμων αστικών κοινοτήτων.

Υπάρχουν πολλές υπηρεσίες οι οποίες παρέχουν οδηγίες για την μετακίνηση μέσα στο οδικό δίκτυο (GPS navigation) όπως για παράδειγμα οι συσκευές Garmin που παρέχουν οδική καθοδήγηση με IX αυτοκίνητα . Οι υπηρεσίες αυτές στηρίζονται σε συστήματα GIS (Geographic Information System) , παρέχουν οδηγίες σχετικά με την μετακίνηση στο οδικό δίκτυο και αφορούν αποκλειστικά το οδικό δίκτυο. Συνήθως είναι ανεξάρτητες του αστικού δικτύου. Εμείς θέλουμε να αναπτύξουμε μια εφαρμογή αποκλειστικά για το αστικό δίκτυο.

Η ευχρηστία των δημόσιων μέσων μεταφοράς μπορεί να ενισχυθεί σημαντικά με την παροχή κατάλληλων μηχανισμών πληροφόρησης. Υπάρχουν πολλά εργαλεία τόσο στο διαδίκτυο όσο και σε κινητές συσκευές που αποσκοπούν στην παροχή έξυπνων μεθόδων για την βέλτιστη χρήση των αστικών μέσων μεταφοράς. Ένα από αυτά τα εργαλεία είναι και το EasyTransit, μία Android εφαρμογή που αναπτύξαμε με σκοπό την εύκολη, αποτελεσματική και αποδοτική πληροφόρηση των χρηστών της σχετικά με τα δημόσια μέσα μεταφοράς.

Στόχος λοιπόν της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη εφαρμογής σε κινητή πλατφόρμα Android χρησιμοποιώντας τα ήδη υπάρχοντα δεδομένα αστικών συγκοινωνιών μίας πόλης (GTFS), η οποία θα παρουσιάζει στον χρήστη της κινητής συσκευής τη βέλτιστη (λιγότερες μετεπιβιβάσεις, ταχύτερη) διαδρομή μεταξύ δύο σημείων, χρησιμοποιώντας αποκλειστικά αστικές συγκοινωνίες. Η ανάπτυξη της εφαρμογής στηρίζεται στην ήδη υλοποιημένη server υποδομή του Open Trip Planner και είναι διαθέσιμη για τις περιοχές Tampa, Porland και Cyprus (μερικώς).

Table of Contents

Κεφάλαιο 1. Εισαγωγή 1
1.1 Έξυπνα κινητά τηλέφωνα (smart phones) 1
1.2 Λειτουργικό σύστημα Android 2
1.3 Αστικό δίκτυο συγκοινωνίας 4
1.4 Κίνητρα διπλωματικής εργασίας 5
1.5 Στόχοι διπλωματικής εργασίας 7

Κεφάλαιο 2. Παρουσίαση συναφούς βιβλιογραφίας 8
2.1 Πρόβλημα του πλανόδιου πωλητή (TSP) 8
2.2 Αλγόριθμοι που μελετήθηκαν 10
2.2.1 h_εu-optimal algorithm 10
2.2.2 Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns 14
2.2.3 Alternative Route Graphs in Road Networks 17
2.3 Σχετικές εφαρμογές 19
2.4 Open Trip Planner 23
2.5 GTFS δεδομένα 31
2.6 Client-Server Based VS Client Based-Only 33
2.8.1 Client-Server Based 33
2.8.2 Client Based Only 35
2.8.3 Συμπεράσματα 36

Κεφάλαιο 3. Αλγόριθμος Δρομολόγησης 39
3.1. Διατύπωση του προβλήματος 39
3.2. A* Αλγόριθμος 41
3.3 Χρονική πολυπλοκότητα 48

Κεφάλαιο 4. Περιγραφή Συστήματος 49
4.1 Τεχνολογίες-Εργαλεία που χρησιμοποιήθηκαν 49
4.1.1 Server-Side 49
4.1.2 Client-Side 50
4.2 Συσκευές που χρησιμοποιήθηκαν 52
4.3 Αρχιτεκτονική συστήματος 53
4.3.1 Γενική περιγραφή συστήματος 53
4.3.2 Ανάλυση αρχιτεκτονικής 55
4.4 Λειτουργίες του συστήματος 68
4.4.1 Δρομολόγηση Διαδρομής 69
4.4.2 Αναζήτηση Σταθμού/Στάσης 75
4.4.3 Οδηγός χρήσης λειτουργιών 76

Κεφάλαιο 5 Αξιολόγηση Συστήματος 77
5.1 Σκοπιμότητα των πειραμάτων 77
5.2 Χρόνος αποκρισιμότητας συστήματος κατά τον υπολογισμό μιας διαδρομής 77
5.3 Αξιολόγηση Ευχρηστίας 92
5.3.1 Διαδικασία αξιολόγησης 92
5.3.2 Στατιστικές μετρήσεις 94
5.4 Ενδεικτική σύγκριση με Google Maps 103

Κεφάλαιο 6. Συμπεράσματα 105
6.1 Σύνοψη-Συμπεράσματα 105
6.2 Μελλοντική εργασία 106

Βιβλιογραφία 107

Table of Figures

Διάγραμμα 1.1. Αρχιτεκτονική Android....3
Διάγραμμα 1.2. Στατιστικά πωλήσεων smartphone παγκοσμίως[7]....4
Εικόνα 2.1. Κριτήρια ταξιδιού που θέτουν οι χρήστες της εφαρμογής ([13],fig 3)....11
Διάγραμμα 2.2. heu-optimal Expansion of the Search Space 9 ([12], figure 5)....12
Διάγραμμα 2.6. Αρχιτεκτονική OTP....25
Διάγραμμα 2.7. Ένας γράφος δρόμων συνδεδεμένος με ένα γράφο διέλευσης (Μανχάταν-Νέα Υόρκη)....26
Διάγραμμα 2.8. Υφιστάμενο Μοντέλο Γράφου OTP....27
Σχήμα 2.9. Οπτική αναπαράσταση γράφου για περιοχή Portland....28
Σχήμα 2.10. Οπτική αναπαράσταση γράφου για περιοχή Λευκωσίας....29
Σχήμα 2.11. Οπτική αναπαράσταση γράφου για περιοχή Tampa....30
Διάγραμμα 3.1. A* υπολογισμός κόστους....41
Διάγραμμα 3.2. Παράδειγμα A* (Εξέταση Ημιεξαμήνου 2009-2010 Τεχνητή Νοημοσύνη-Ελπίδα Κεραυνού)....43
Διάγραμμα 4.3. Γενική περιγραφή συστήματος....53
Διάγραμμα 4.4. Αρχιτεκτονική server-side....55
Εικόνα 4.5 Cron job Καθορισμός χρονοπρογράμματος....58
Διάγραμμα 4.6. Αρχιτεκτονική client-side....61
Εικόνα 4.8 Αρχική Οθόνη Εφαρμογής....68
Εικόνα 4.9 Επιλογή Εξυπηρετητή....69
Εικόνα 4.10 Χειροκίνητη επιλογή εξυπηρετητή....70
Εικόνα 4.11 Μέθοδοι δρομολόγησης....70
Εικόνα 4.12 Προτιμήσεις χρήστη....71
Εικόνα 4.13 Παρουσίαση διαδρομής και οδηγιών....71
Εικόνα 4.15 Επιπλέον επιλογές....72
Εικόνα 4.16 Ρυθμίσεις....73
Εικόνα 4.17 Πρόσφατες διαδρομές....74
Εικόνα 4.18 Λειτουργία αναζήτησης....75
Εικόνα 4.19 Οδηγός Χρήσης Λειτουργιών....76
Σχήμα 5.6 Γραφική παράσταση χρόνου απόκρισης σε σχέση με την απόσταση....85
Εικόνα 5.12 QR Code για την εφαρμογή EasyTransit....93
Εικόνα 5.13 Στατιστικά στοιχεία φύλου....94
Εικόνα 5.14 Στατιστικά στοιχεία ηλικίας....94
Εικόνα 5.15 Στατιστικά στοιχεία ακαδημαϊκής γνώσης....95
Εικόνα 5.16 Στατιστικά στοιχεία χρηστών android OS....95
Εικόνα 5.17 Στατιστικά στοιχεία διαπροσωπείας χρήσης....96
Εικόνα 5.18 Στατιστικά στοιχεία ευκολίας εκμάθησης λειτουργιών....97
Εικόνα 5.19 Στατιστικά στοιχεία ευκολίας χρήσης λειτουργιών....97
Εικόνα 5.20 Στατιστικά στοιχεία ομαλής μετάβασης οθονών....98
Εικόνα 5.21 Στατιστικά στοιχεία αποκρισιμότητας εφαρμογής....98
Εικόνα 5.22 Στατιστικά στοιχεία κατανόησης μηνυμάτων....99
Εικόνα 5.23 Στατιστικά στοιχεία βαθμού αρέσκειας εφαρμογής....100
Εικόνα 5.24 Στατιστικά στοιχεία βαθμού χρήσης εφαρμογής....100
Εικόνα 5.25 Στατιστικά στοιχεία βαθμού ικανοποίησης λειτουργιών....101
Εικόνα 5.26 Στατιστικά στοιχεία βαθμού πρότασης εφαρμογής σε άλλους....101
Εικόνα 5.26 Ενδεικτική σύγκριση με google maps....103
Εικόνα 5.27 Ενδεικτική σύγκριση με google maps....104

Table of Tables

Πίνακας 2.3. Χρονοδιάγραμμα τρένων 15
Πίνακας 2.4. Δισδιάστατος πίνακας διαδρομών μιας γραμμής 16
Πίνακας 2.5. Σύγκριση σχετικών εφαρμογών 22
Πίνακας 2.12. Χαρακτηριστικά Nexus-S 36
Πίνακας 4.1. Κυρίως τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν 51
Πίνακας 4.2. Χαρακτηριστικά συσκευών που χρησιμοποιήθηκαν 52
Πίνακας 4.7. Παράδειγμα Polyline Encoder 64
Πίνακας 4.14 Αντιστοίχιση Μέσων Μεταφοράς με τα χρώματα 72
Πίνακας 5.1 Χαρακτηριστικά συσκευών πειράματος 78
Πίνακας 5.2 Αποτελέσματα πειράματος 82
Πίνακας 5.3 Μέσοι όροι πειραμάτων 83
Πίνακας 5.4 Διάμεσοι πειραμάτων 83
Πίνακας 5.5 Χαρακτηριστικά γράφων 84
Πίνακας 5.7 Αποτελέσματα ANOVA test για 1 km 88
Πίνακας 5.8 Αποτελέσματα ANOVA test για 5 km 89
Πίνακας 5.9 Αποτελέσματα ANOVA test για 10 km 89
Πίνακας 5.10 Αποτελέσματα ANOVA test για 20 km 90
Πίνακας 5.11 Αποτελέσματα ANOVA test για 30 km 91

Κεφάλαιο 1. Εισαγωγή

1.1 Έξυπνα κινητά τηλέφωνα (smart phones) 1
1.2 Λειτουργικό σύστημα Android 2
1.3 Αστικό δίκτυο Συγκοινωνίας 4
1.4 Κίνητρα διπλωματικής εργασίας 5
1.5 Στόχοι διπλωματικής εργασίας 7

Στο κεφάλαιο αυτό κάνουμε μια εισαγωγή στο πεδίο με το οποίο ασχοληθήκαμε στην διπλωματική εργασία. Αρχικά αναφερόμαστε στην ιστορία του πρώτου φορητού τηλεφώνου και στην εξέλιξη των φορητών τηλεφώνων εξηγώντας επίσης την ορολογία «Έξυπνα κινητά τηλέφωνα». Στη συνέχεια αναφερόμαστε στο λειτουργικό σύστημα Android της Google. Ακολούθως εξηγούμε τι εννοούμε με τον όρο Αστικό δίκτυο συγκοινωνιών. Τέλος, αναφερόμαστε εν συντομία στα κίνητρα και τους στόχους της παρούσας διπλωματικής εργασίας.

1.1 Έξυπνα κινητά τηλέφωνα (smart phones)

Μέχρι το 1973, η τεχνολογία της κινητής τηλεφωνίας περιοριζόταν μόνο σε τηλέφωνα εγκατεστημένα σε αυτοκίνητα και άλλα οχήματα. Στις 3 Απριλίου 1973, ο Martin Cooper δουλεύοντας για την εταιρεία Motorola, εφηύρε το πρώτο φορητό κινητό τηλέφωνο με το οποίο πραγματοποίησε το πρώτο αναλογικό φορητό τηλεφώνημα χρησιμοποιώντας ένα βαρύ πρωτότυπο μοντέλο ([1]). Το συγκεκριμένο μοντέλο ζύγιζε 2.5 κιλά, είχε μέγεθος 9x5x1.75 ίντσες με μέγιστη ώρα ομιλίας 30 λεπτά και συνολικό χρόνο φόρτισης 10 ώρες. Αργότερα, το 1983, αφού έγιναν αρκετές βελτιστοποιήσεις στο πρωτότυπο μοντέλο, το Motorola DynaCat 8000s βγήκε στην αγορά με αρχική τιμή $3,995 ([2]).

Από τότε, η τεχνολογία των κινητών τηλεφώνων έχει ακολουθήσει μια συνεχόμενη ανοδική εξέλιξη. Την δεκαετία του ’90 έγινε ψηφιοποίηση των δικτύων (GSM [3]) ενώ παράλληλα οι κινητές συσκευές άρχισαν να παίρνουν μια πιο προσιτή μορφή (πιο ελαφριές, πιο μικρές) οδηγώντας έτσι στα κινητά 2ας γενεάς (2G) που παρουσίασαν πρωτοποριακές λειτουργίες όπως αποστολή μηνυμάτων SMS και λήψη ψηφιακών φωτογραφιών.

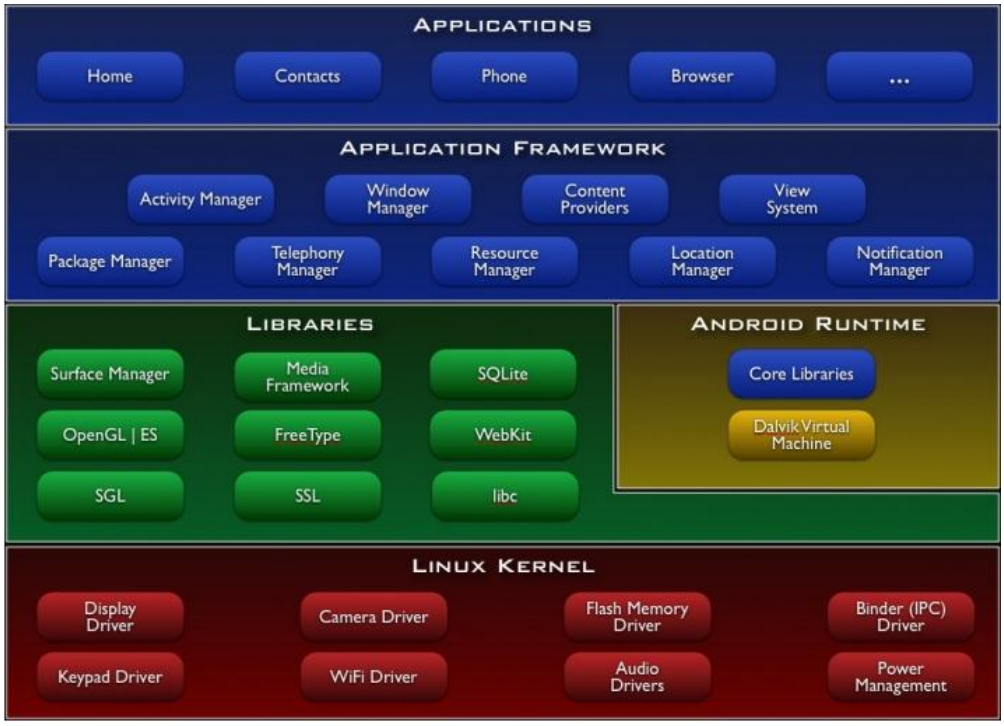
Πλέον, με τα σημερινά δεδομένα και με την ραγδαία ανάπτυξη της τεχνολογίας , εισάχθηκε μια νέα ορολογία, αυτή των έξυπνων συσκευών (κινητά τηλέφωνα 3ης γενεάς-3G). Οι συσκευές αυτές έχουν χαρακτηριστικά ενός ηλεκτρονικού υπολογιστή, συνήθως με δυνατότητες μικρότερης κλίμακας, και έχουν γίνει επίκεντρο της παγκόσμια αγοράς κινητής τηλεφωνίας ([4]). Τα έξυπνα κινητά είναι κατασκευασμένα σε κινητές υπολογιστικές πλατφόρμες και παρέχουν μεγαλύτερη υπολογιστική ισχύ και συνδεσιμότητα σε σχέση με τα απλά κινητά τηλέφωνα. Η ουσιαστική όμως διαφορά είναι ότι οι έξυπνες συσκευές έχουν προηγμένες διεπαφές προγραμματισμού εφαρμογών (APIs) με τις οποίες είναι δυνατόν να τρέχουν 3rd party applications που επιτρέπουν σε αυτές τις εφαρμογές να έχουν καλύτερη ενσωμάτωση στο λειτουργικό σύστημα και στο υλικό του τηλεφώνου ([5]). Έτσι, οι προγραμματιστές μπορούν πολύ εύκολα και γρήγορα να δημιουργήσουν τις δικές τους εφαρμογές ούτως ώστε να τρέχουν σε κινητές πλατφόρμες.

Τα έξυπνα κινητά παρέχουν πολλαπλές δυνατότητες. Τα πλείστα διαθέτουν ενσωματωμένο δέκτη GPS για την αναγνώριση της παρούσας γεωγραφικής θέσης του χρήστη και WIFI δέκτη που επιτρέπει την άμεση χρήση του διαδικτύου σε υψηλές ταχύτητες. Διαθέτουν αισθητήρες όπως το επιταχυνσιόμετρο και το γυροσκόπιο με το οποίο αναγνωρίζουμε σε πια στάση βρίσκεται το τηλέφωνο και φωτοαισθητήρες για την αυτόματη ρύθμιση της φωτεινότητας της οθόνης και εξοικονόμηση ενέργειας. Επιπλέον, παρέχουν μεγάλη υπολογιστική ισχύ για να χρησιμοποιηθεί από εφαρμογές με αυξημένη απαίτηση σε υπολογιστική ισχύ (παράδειγμα: games). Όλες αυτές οι δυνατότητες είναι διαθέσιμες προς χρήση από τους προγραμματιστές για την ανάπτυξη εφαρμογών.

1.2 Λειτουργικό σύστημα Android

Το Android είναι ένα linux-βασισμένο λειτουργικό σύστημα για κινητές συσκευές όπως τα έξυπνα κινητά και οι ταμπλέτες (tablets). Αρχικά αναπτύχθηκε το 2003 από τους Andy Rubin, Rich Miner και Chris White ιδρύοντας την εταιρεία Android Inc.και έπειτα το 2005 αγοράστηκε από την Google. Από τότε βρίσκεται σε συνεχή ανέλιξη. Σημαντικό χαρακτηριστικό του Android λειτουργικού συστήματος είναι ο ανοικτός πηγαίος κώδικας με τον οποίο οποιοσδήποτε μπορεί να συνεισφέρει στην ανάπτυξή του και να τον χρησιμοποιήσει με όποιο τρόπο θέλει. Το Android είναι γραμμένο σε γλώσσα προγραμματισμού Java και παρέχει ένα πλήθος βιβλιοθηκών λογισμικού μέσω του Android JDK ([6]).

Android Runtime: Το Android περιλαμβάνει ένα σύνολο από βιβλιοθήκες πυρήνα, που περιλαμβάνουν τις πλείστες λειτουργίες που περιλαμβάνουν και οι βιβλιοθήκες πυρήνα της java. Κάθε εφαρμογή Android, τρέχει στην δική της διεργασία (process) μέσα σε στιγμιότυπο της εικονικής μηχανής Dalvik . Η μηχανή Dalvik είναι μια εικονική μηχανή εξειδικευμένη για κινητές συσκευές και έχει υλοποιηθεί με τέτοιο τρόπο ώστε μία συσκευή να μπορεί να χρησιμοποιεί πολλαπλές εικονικές μηχανές με αποδοτικό τρόπο.



Διάγραμμα 1.1. Αρχιτεκτονική Android

Το διάγραμμα 1.1 παρουσιάζει τα κύρια συστατικά του λειτουργικού συστήματος Medoid ([6]).

Το **Applications module** αφορά ένα σύνολο από έτοιμες εφαρμογές πυρήνα όπως SMS λειτουργία, email client, maps, browser, contacts.

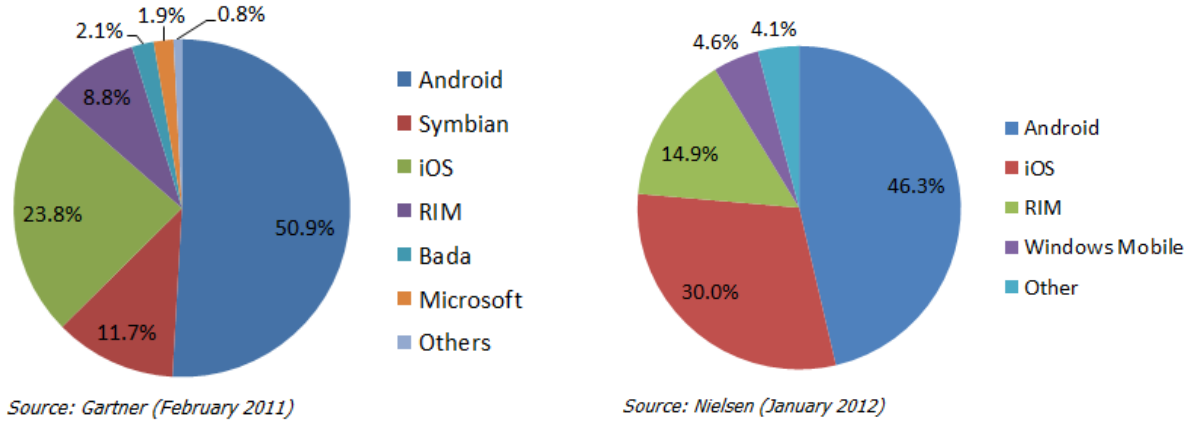
Το **Application Framework module** προσφέρει την δυνατότητα στους προγραμματιστές να χρησιμοποιήσουν το υλικό της συσκευής, να έχουν πρόσβαση σε υπηρεσίες τοποθεσίας, να τρέξουν background υπηρεσίες, να διαχειριστούν τους πόρους της συσκευής.

Το **Libraries module** περιλαμβάνει κάποιες βιβλιοθήκες C/C++ που χρησιμοποιούνται από πολλά τμήματα του Android OS. Αυτές οι βιβλιοθήκες είναι προσβάσιμες από τους προγραμματιστές μέσω του Android Application framework.

Το **Android runtime module** περιλαμβάνει τις βιβλιοθήκες πυρήνα και την Dalvik VM με τα οποία οι προγραμματιστές μεταγλωττίζουν το πρόγραμμά τους και το εκτελούν σε εικονική μηχανή Dalvik.

Το **Linux Kernel module** αφορά υπηρεσίες πυρήνα όπως ασφάλεια,διαχείριση μνήμης, διαχείριση διεργασιών, διαχείριση ενέργειας. Επίσης λειτουργεί ως ένα αφαιρετικό επίπεδο μεταξύ του υλικού και του λογισμικού.

Οι λόγοι που μας ώθησαν στο να αναπτύξουμε την εφαρμογή μας σε λειτουργικό σύστημα Android είναι κυρίως δύο. Ο 1ος αφορά την ευκολία χρήσης του Android JDK. Η γλώσσα προγραμματισμού που χρησιμοποιείται είναι Java, η οποία είναι αντικειμενοστραφής γλώσσα προγραμματισμού και είναι ευρέως διαδεδομένη. Επιπλέον, παρέχεται αρκετή βοήθεια στο διαδίκτυο σχετικά με Android μέσα από το Android documentation αλλά και μέσα από διάφορα forums.Ο 2^{ος} λόγος και ο πιο σημαντικός είναι ότι στατιστικές έρευνες που έγιναν τα τελευταία 2 χρόνια έδειξαν ότι το μεγαλύτερο ποσοστό χρηστών έξυπνων κινητών χρησιμοποιεί Android OS ([7]).



Διάγραμμα 1.2. Στατιστικά πωλήσεων smartphone παγκοσμίως[7]

1.3 Αστικό δίκτυο συγκοινωνίας

Με την ορολογία αστικό δίκτυο συγκοινωνίας εννοούμε τα δημόσια μεταφορικά μέσα (ΔΜΜ). Είναι μια κοινή υπηρεσία μεταφοράς επιβατών η οποία είναι διαθέσιμη για χρήση από το ευρύ κοινό, σε αντιδιαστολή με τους τρόπους μεταφοράς όπως ταξί, αυτοκίνητο, carpooling ή ενοικιαζόμενα λεωφορεία τα οποία δεν συμμερίζονται με αγνώστους ούτε αφορούν ιδιωτικές συμφωνίες.

Τα δημόσια μέσα μεταφοράς περιλαμβάνουν τα λεωφορεία, τα τρόλεϊ, τα τραμ, τα τρένα, ταχεία μέσα μεταφοράς όπως metro, subways, undergrounds και τα ferries. Οι δημόσιες συγκοινωνίες μεταξύ πόλεων κυριαρχείται από αεροπορικές εταιρείες, coaches και υπεραστικές σιδηρογραμμές. Υψηλών ταχυτήτων σιδηροδρομικά δίκτυα αναπτύσσονται σε πολλά μέρη του κόσμου.

Οι αστικές δημόσιες συγκοινωνίες συνήθως παρέχονται από μία ή περισσότερες ιδιωτικές επιχειρήσεις μεταφορών ή από κάποια αρχή συγκοινωνιών. Οι δημόσιες υπηρεσίες μεταφορών συνήθως χρηματοδοτούνται από τις κρατικές επιδοτήσεις με κόμιστρα που εφαρμόζουν σε κάθε επιβάτη.

Τα περισσότερα δημόσια μέσα μεταφοράς ακολουθούν κάποιο προγραμματισμένο χρονοδιάγραμμα. Το χρονοδιάγραμμα αυτό γνωστοποιείται προς το ευρύ κοινό ούτως ώστε να επιτρέψει σε πιθανούς χρήστες να προσχεδιάσουν τα ταξίδια τους και συνήθως μαζί με το χρονοδιάγραμμα δίνονται οι κατάλληλοι χάρτες και η διαβάθμιση των ναύλων για να βοηθήσει τους ταξιδιώτες να οργανώσουν καλύτερα το ταξίδι τους. Επιπλέον, τα τελευταία χρόνια η τεχνολογία του διαδικτύου έκανε δυνατό τον σχεδιασμό διαδρομών (trip planning) μέσω διαδικτυακών υπηρεσιών δρομολόγησης όπως είναι το Google Maps, Open Trip Planner, Bing Maps διευκολύνοντας με αυτό τον τρόπο ακόμα περισσότερο τους χρήστες .

1.4 Κίνητρα διπλωματικής εργασίας

Όπως έχουμε αναφέρει και στην περίληψη στην αρχή του εγγράφου αυτού, τα δημόσια μέσα μεταφοράς αποτελούν λύση για πολλά προβλήματα που μαστίζουν την κοινωνία στην οποία ζούμε. Συγκεκριμένα πιο κάτω θα δούμε ποιους τομείς επηρεάζουν τα ΔΜΜ.

α. Τα δημόσια μέσα μεταφοράς ενισχύουν τις προσωπικές ευκαιρίες.

Παρέχουν την προσωπική κινητικότητα και την ελευθερία μετακίνησης σε όλα τα στάδια της ζωής μας. Επιπλέον, μας δίνουν την δυνατότητα να πάμε στην δουλειά μας, να επισκεφθούμε φίλους και δίνουν πρόσβαση σε υπηρεσίες που αφορούν την υγεία μας (γιατροί, νοσοκομεία) . Τέλος, σε μεγάλες περιοχές όπως είναι η Αμερική, τα ΔΜΜ δίνουν την ευκαιρία σε εκατομμύρια ανθρώπους να έχουν πρόσβαση σε ευκαιρίες εργοδότησης.

β. Εξοικονόμηση καυσίμων, μείωση κυκλοφοριακής συμφόρησης.

Έρευνες που έγιναν έδειξαν ότι η πρόσβαση σε λεωφορεία και σιδηροδρομικές γραμμές μειώνει την οδήγηση μέχρι 4,400 μίλια ανά σπίτι ετησίως. Επιπλέον, έρευνες που έγιναν στην Αμερική έδειξαν ότι σε περιοχές όπου υπάρχουν καλά δίκτυα δημόσιων μέσων μεταφοράς υπάρχει εξοικονόμηση 785 εκατομμύριων ωρών σε χρόνο μετακίνησης και εξοικονόμηση 640 εκατομμυρίων γαλονιών καυσίμου ετησίως. Έχουν υπολογιστεί και τα κόστη της κυκλοφοριακής συμφόρησης εάν δεν υπήρχαν τα ΔΜΜ και η τιμή ανέρχεται στα 19 δισεκατομμύρια δολάρια ([8]).

γ. Εξοικονόμηση Χρημάτων.

Ένα μέσο σπίτι ξοδεύει 18 σεντς του δολαρίου για μετακίνηση, από το οποίο το 94% αυτού πηγαίνει στην αγορά, συντήρηση και λειτουργία των ιδιωτικών οχημάτων. Επίσης, τα ΔΜΜ αποτελούν μια οικονομική και για πολλούς απαραίτητη εναλλακτική λύση για μεταφορά αφού το κόστος των ναύλων είναι αρκετά πιο φτηνό από το συνολικό κόστος ενός ιδιωτικού οχήματος..

δ. Μείωση ρύπανσης του περιβάλλοντος-Υγεία

Είναι ευρέως γνωστό ότι οι εξατμίσεις των μηχανοκίνητων οχημάτων προκαλούν ρύπανση στο περιβάλλον. Στην Κίνα και στις ΗΠΑ παράγεται σχεδόν το μισό διοξείδιο του άνθρακα όλου του κόσμου το οποίο προέρχεται κυρίως από μηχανοκίνητα. Η ελευθέρωση αυτού του αερίου προκαλεί διάφορα προβλήματα. Επεκτείνει το φαινόμενο του Θερμοκηπίου με συνέπεια τις ακραίες καιρικές μεταβολές, παγκόσμια αύξηση της θερμοκρασίας και σταδιακή απώλεια των οικοσυστημάτων. Επίσης, σύμφωνα με μια μελέτη που έγινε το 1996 (EPA) τα μηχανοκίνητα οχήματα είναι αιτία για δημιουργία πάνω από 3000 περιπτώσεις καρκίνου. Μια πρόσφατη μελέτη διαπίστωσε ότι η έκθεση σε καυσαέρια από τα αυτοκίνητα αυξάνει σημαντικά την πιθανότητα κάποιο άτομο να υποστεί καρδιακή προσβολή ([9]). Εάν σκαθημερινά ένα άτομο στραφεί προς τα ΔΜΜ οι εκπομπές του διοξειδίου του άνθρακα μπορούν να μειωθούν σε 20 pounds ή περισσότερο από 4800 pounds ετησίως. Επομένως, η χρήση ΔΜΜ παρόλο που δεν θα επιλύσει τα πιο πάνω προβλήματα θα συντελέσει σημαντικά στην μείωσή τους.

Το μεγαλύτερο πρόβλημα όμως που προκύπτει είναι η άγνοια των ανθρώπων σχετικά με τη χρήση των ΔΜΜ. Πολλές χώρες έχουν πλήρη ανεπτυγμένα συστήματα δημόσιας συγκοινωνίας αλλά πολλοί πολίτες είτε δεν γνωρίζουν πώς να τα χρησιμοποιούν είτε δεν γνωρίζουν πως υπάρχουν είτε δεν θέλουν να μπουν στον κόπο να ψάξουν και να οργανώσουν το ταξίδι τους γιατί ίσως το θεωρούν χρονοβόρο. Γι’ αυτό η ευχρηστία των δημόσιων μέσων μεταφοράς μπορεί να ενισχυθεί σημαντικά με την παροχή κατάλληλων μηχανισμών πληροφόρησης. Υπάρχουν πολλά εργαλεία τόσο στο διαδίκτυο όσο και σε κινητές συσκευές που αποσκοπούν στην παροχή έξυπνων μεθόδων για την βέλτιστη χρήση των αστικών μέσων μεταφοράς. Και σε αυτό το σημείο ερχόμαστε και εμείς να αναπτύξουμε μια Android εφαρμογή η οποία θα είναι εύχρηστη, απλή και αποδοτική για να βοηθήσει τους χρήστες της να οργανώσουν εύκολα τις διαδρομές τους και να τους παρέχει βήμα-προς-βήμα οδηγίες μέχρι τον στόχο τους. Επιπλέον, η εφαρμογή μας βρίσκει απήχηση και σε ξένους ταξιδιώτες, σε τουρίστες που επισκέπτονται άγνωστες χώρες με άγνωστη γλώσσα ούτως ώστε να μπορούν να χρησιμοποιήσουν τα ΔΜΜ για να μετακινηθούν στις τοποθεσίες που αυτοί επιθυμούν.

1.5 Στόχοι διπλωματικής εργασίας

Στόχος της διπλωματικής εργασίας είναι η ανάπτυξη μια κινητής πλατφόρμας Android χρησιμοποιώντας τα ήδη υπάρχοντα δεδομένα αστικών συγκοινωνιών μίας πόλης (GTFS) και τη ήδη υλοποιημένη server υποδομή του Open Trip Planner, θα μπορεί να παρουσιάζει στον χρήστη της κινητής συσκευής τη βέλτιστη (λιγότερες μετεπιβιβάσεις, ταχύτερη) διαδρομή μεταξύ δύο σημείων, χρησιμοποιώντας αποκλειστικά αστικές συγκοινωνίες. Η ανάπτυξη της πλατφόρμας θα στηρίζεται, επίσης, στις εξής προδιαγραφές:

- Η παρουσίαση των αποτελεσμάτων θα γίνεται τόσο σε χάρτες (Open Street Maps), όσο και με μορφή κειμένου, χρησιμοποιώντας τις δυνατότητες της κινητής συσκευής.
- Θα είναι δυνατή η αποθήκευση των αποτελεσμάτων των αναζητήσεων του χρήστη, ώστε ο χρήστης να έχει πρόσβαση σε αυτά ακόμα και όταν δεν είναι συδνεδεμένος με το διαδίκτυο.

Τα καθήκοντα της εργασίας είναι τα παρακάτω:

- Υλοποίησης ενός διαδραστικού εργαλείου χαρτών, προκειμένου ο χρήστης να ορίσει αφετηρία και τέλος διαδρομής, καθώς και τις λεπτομέρειες της διαδρομής του (ώρα, μέσο κτλ)
- Επικοινωνία του εργαλείου με το Open Trip Planner προκειμένου να διαβιβαστεί το αίτημα του χρήστη
- Λήψη της απάντησης από το Open Trip Planner και κατάλληλη παρουσίαση των αποτελεσμάτων (χάρτες – κείμενο)
- Δυνατότητα αποθήκευσης των αναζητήσεων του χρήστη στην κινητή συσκευή, ώστε να μπορεί να έχει πρόσβαση σε αυτές, ακόμα και όταν δεν είναι συνδεδεμένος με το διαδίκτυο.
- Παρουσίαση πληροφοριών σχετικά με το δίκτυο αστικών συγκοινωνιών (στάσεις, timetables)

Κεφάλαιο 2. Παρουσίαση συναφούς βιβλιογραφίας

2.1 Πρόβλημα του πλανόδιου πωλητή 8
2.2 Αλγόριθμοι που μελετήθηκαν 10
2.3 Σχετικές εφαρμογές 19
2.4 Open Trip Planner 23
2.5 GTFS Δεδομένα 31
2.6 Client Server Based vs Client Based Only 33

Στο κεφάλαιο αυτό θα μιλήσουμε για την βιβλιογραφία την οποία μελετήσαμε που βοήθησε στην διεκπεραίωση της παρούσας διπλωματικής εργασίας. Αρχικά θα μιλήσουμε για το πρόβλημα του πλανόδιου πωλητή και πως αυτό συνδέεται με το θέμα μας. Έπειτα θα αναφερθούμε στα NP δύσκολα προβλήματα και θα εξηγήσουμε γιατί το πρόβλημα του πλανόδιου πωλητή ανήκει σε αυτού του είδους τα προβλήματα. Στη συνέχεια θα αναφερθούμε στους αλγόριθμους δρομολόγησης και στις σχετικές εφαρμογές που μελετήσαμε. Θα αναφερθούμε επίσης στο Open Trip Planner στο οποίο βασιστήκαμε για την διεκπεραίωση της παρούσης διπλωματικής εργασίας και θα αναλύσουμε την αρχιτεκτονική του συστήματος. Έπειτα θα αναφερθούμε στην εταιρεία Geomatic Technologies Ltd από την οποία πήραμε τα GTFS δεδομένα για το αστικό δίκτυο της Κύπρου. Τέλος θα αναφερθούμε στα συστήματα πελάτη-εξυπηρετητή και στα συστήματα πελάτη και κάνοντας κάποια σύγκριση θα αποφασίσουμε τελικά ποια από τις 2 αρχιτεκτονικές θα χρησιμοποιήσουμε στην εφαρμογή μας.

2.1 Πρόβλημα του πλανόδιου πωλητή (TSP)

Το πρόβλημα του πλανόδιου πωλητή είναι ένα ευρέως διαδεδομένο πρόβλημα τόσο στην θεωρητική επιστήμη των υπολογιστών (θεωρία υπολογιστικής πολυπλοκότητας) όσο και σε πολλούς άλλους τομείς όπως είναι οι ερευνητικές επιχειρήσεις ή ακόμα και στην ιατρική για την μελέτη της αλληλουχίας του DNA.

Το πρόβλημα διατυπώθηκε για πρώτη φορά ως ένα μαθηματικό πρόβλημα το 1930 από τον K.Menger[10], και είναι ένα από τα πιο εντατικά μελετημένα προβλήματα προς βελτιστοποίηση. Το πρόβλημα είναι το εξής: Έχοντας ως δεδομένα ένα σύνολο από πόλεις και τις αποστάσεις τους ανά ζεύγη, ο στόχος είναι να βρεθεί η συντομότερη δυνατή διαδρομή

στην οποία γίνεται επίσκεψη σε κάθε πόλη ακριβώς μια φορά και επιστροφή πίσω στην πόλη προέλευσης. Το TSP μπορεί να μοντελοποιηθεί ως ένα μη-κατευθυνόμενο γράφημα με βάρη όπου οι πόλεις είναι οι κορυφές του γράφου και οι αποστάσεις μεταξύ 2 πόλεων είναι αντίστοιχα τα βάρη πάνω στις ακμές που ενώνουν 2 κορυφές. Πρόκειται για ένα πρόβλημα ελαχιστοποίησης ξεκινώντας και καταλήγοντας σε μια συγκεκριμένη κορυφή αφού ήδη έχουμε επισκεφθεί όλες τις άλλες κορυφές ακριβώς μια φορά.

Να σημειώσουμε ότι κανένας ακριβής αλγόριθμος δεν μπορεί να εγγυηθεί ότι θα δώσει βέλτιστες λύσεις μέσα σε λογικό υπολογιστικό χρόνο όταν ο αριθμός των κόμβων είναι μεγάλος λόγω του ότι είναι NP δύσκολο (Non-Deterministic polynomial time) πρόβλημα.

Όπως έχουμε αναφέρει στους στόχους της διπλωματικής εργασίας (Κεφ. 1) χρειαζόμαστε μια εφαρμογή που να υπολογίζει την βέλτιστη διαδρομή (λιγότερες μετεπιβιβάσεις, συντομότερη διαδρομή) μεταξύ 2 γεωγραφικών σημείων. Έτσι το πρόβλημά που καλούμαστε να επιλύσουμε μπορεί να ενταχθεί στο πρόβλημα του πλανόδιου πωλητή με την διαφορά ότι δεν είναι απαραίτητο όλοι οι κόμβοι να επισκεφθούν και επίσης δεν θα επιστρέψουμε πίσω στο σημείο εκκίνησης.

<u>NP-δύσκολα προβλήματα:</u>

Τα NP δύσκολα προβλήματα είναι προβλήματα για τα οποία δεν γνωρίζουμε κάποιο αποδοτικό αλγόριθμο που να βρίσκει λύση σε πολυωνυμικό χρόνο. Είναι προβλήματα δυσεπίλυτα, όσο μεγαλώνουν τα στιγμιότυπα γίνεται αδύνατο να τα επιλύσουμε σε πολυωνυμικό χρόνο. Ο χρόνος που απαιτείται να λυθεί το πρόβλημα χρησιμοποιώντας οποιοδήποτε γνωστό αλγόριθμο αυξάνεται εκθετικά όσο το μέγεθος του προβλήματος αυξάνεται.

Ακόμα ένας πιο σωστός ορισμός είναι: Ένα πρόβλημα είναι NP-δύσκολο εάν ένας αλγόριθμος που το λύνει μπορεί να μεταφραστεί σε ένα αλγόριθμο που λύνει οποιοδήποτε NP-πρόβλημα). Το NP-δύσκολο λοιπόν σημαίνει ότι "είναι τουλάχιστον τόσο δύσκολο όσο κάθε άλλο NP-πρόβλημα" και μάλιστα μπορεί να είναι ακόμα πιο δύσκολο.

Σε αυτό το σημείο θα εξετάσουμε γιατί το TSP ανήκει στα NP-δύσκολα προβλήματα. Υπάρχουν "εύκολες" λύσεις για κάποια προβλήματα TSP. Με τον όρο "εύκολες" λύσεις εννοούμε τα TSP που περιλαμβάνουν μόνο μερικούς κόμβους. Εάν υπάρχουν μόνο 2 κόμβοι τότε μόνο 1 μονοπάτι υπάρχει. Το TSP όμως αφορά οποιοδήποτε αριθμό κόμβων. Γενικά

υπάρχουν (n-1)!/2 μονοπάτια διότι έχουμε συμμετρικούς κόμβους δλδ η απόσταση του Α από τον Β είναι η ίδια με την απόσταση του Β από τον Α. Το (n-1) οφείλεται στο ότι ξεκινούμε από τον αρχικό κόμβο και απομένει να διερευνήσουμε ακόμα n-1 κόμβους.
Παρατηρούμε λοιπόν ότι όσο αυξάνεται ο αριθμός των κόμβων n, τα μονοπάτια αυξάνονται υπερβολικά. Εφόσον δεν υπάρχει κάποια μηχανή με τέτοια υπολογιστική δύναμη που να υπολογίζει «αστραπιαία» όλα τα μονοπάτια, αναπτύσσονται αλγόριθμοι που βρίσκουν πιθανές λύσεις. Δεν μπορούμε όμως να πούμε ότι αυτές οι λύσεις είναι βέλτιστες διότι θα πρέπει να έχουμε από πριν υπολογίσει τα κόστη όλων των μονοπατιών και να τα συγκρίνουμε με τη λύση μας.
Επομένως, εάν δεν υπολογίσουμε όλα τα μονοπάτια για να βρούμε το ελάχιστο κόστος , δεν μπορούμε να πούμε ότι ο αλγόριθμός μας είναι βέλτιστος.
Με αυτά που αναφέραμε προκύπτει το συμπέρασμα ότι το TSP πρόβλημα ανήκει στα NP δύσκολα προβλήματα.

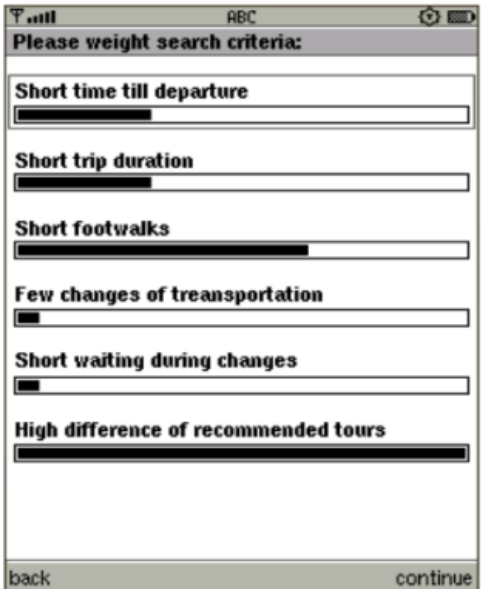
2.2 Αλγόριθμοι που μελετήθηκαν

2.2.1 $h_{\varepsilon}u$-optimal algorithm

Τα [11,12,13] αφορούν άρθρα ενός έργου που ονομάζεται Rose. Το Rose είναι μία εφαρμογή κινητής συσκευής που εισηγείται τοποθεσίες και συμβάντα (events) στους χρήστες του με πραγματικού χρόνου καθοδήγηση δημόσιας μεταφοράς. Αυτά τα χαρακτηριστικά είναι που το κάνουν να ξεχωρίζει από άλλες προϋπάρχων εφαρμογές. Το Rose αντιδρά σε πραγματικού χρόνου καθυστερήσεις στο σύστημα δημόσιας μεταφοράς και υπολογίζει εναλλακτικές διαδρομές εάν χρειάζεται. Χρησιμοποιεί client-server αρχιτεκτονική για να ελαφρύνει τις συσκευές με αδύνατο υλικό. Ο πελάτης στέλλει το αίτημα με http στο εξυπηρετητή. Στον εξυπηρετητή γίνεται επεξεργασία των δεδομένων χρησιμοποιώντας διαφορετικούς παροχείς , όπως για παράδειγμα live public transport data, event και location αρχεία ή map services. Το Rose αποτελείται από κάποια βασικά modules:

- Recommendation module: οι χρήστες τοποθετούν κάποια ενέργεια για παράδειγμα, eat pizza , που στο server εκτελείται υπό την μορφή query, και επιστρέφεται πίσω στο χρήστη μία λίστα με εισηγήσεις (εστιατόρια). Λόγω του ότι τα δεδομένα χρησιμοποιούνται από διαφορετικούς providers, γίνεται χρήση του αλγόριθμου OKAPI measure για να γίνει κάποια κανονικοποίηση των αρχείων διαφορετικού μεγέθους .
- Route Generation module: αφού οι χρήστες εισαγάγουν τα δεδομένα εισόδου , σημείο εκκίνησης σημείο προορισμού κτλ, το σύστημα υπολογίζει τη διαδρομή από μέσω κάποιου ευρετικού αλγόριθμου $h_{\varepsilon}u$.

- Navigation module: αφορά το πως παρουσιάζεται η διαδρομή στο χάρτη της κινητής συσκευής μαζί με timetables για τα δημόσια μέσα μεταφοράς.



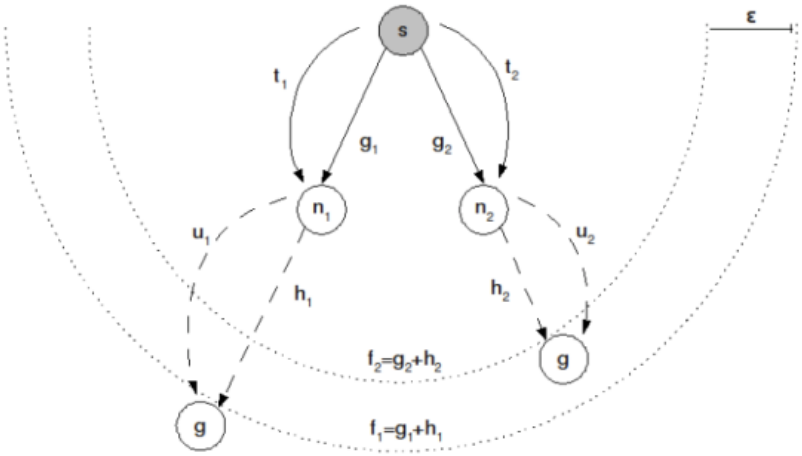
Εικόνα 2.1. Κριτήρια ταξιδιού που θέτουν οι χρήστες της εφαρμογής ([13],fig 3)

Έρευνες που έγιναν έδειξαν ότι οι χρήστες δεν αξιολογούν μία διαδρομή κοιτάζοντας σε μονοδιάστατη κλίμακα , για παράδειγμα την διαδρομή που εκτελείται σε πιο λίγη ώρα αλλά προσπαθούν να βρουν ένα συμβιβασμό μεταξύ πολλαπλών χαρακτηριστικών.

Στην εικόνα 2.1 διαφαίνονται τα κριτήρια που επιλέγουν οι χρήστες βάση των οποίων γίνεται ο υπολογισμός της βέλτιστης διαδρομής. Τα κριτήρια που κρίθηκαν ως πιο σημαντικά είναι:

- No long waiting time until departure.
- Short duration of the trip.
- Short foot walks.
- Few changes of transportation.
- No long waiting time during changes.
- High difference of recommended tours.

Για το Route Generation module χρησιμοποιείται ο **ευρετικός αλγόριθμος $\mathbf{h_{\varepsilon}u}$**. Για να καταλάβουμε πως δουλεύει αυτός ο αλγόριθμος θα κοιτάξουμε το πιο κάτω σχήμα.



Διάγραμμα 2.2. heu-optimal Expansion of the Search Space 9 ([12], figure 5)

Στο διάγραμμα 2.2 βλέπουμε 2 πιθανά μονοπάτια από κάποιο αρχικό κόμβο s σε κάποιο τελικό κόμβο g.

- s: αρχικός κόμβος
- g: τελικός κόμβος
- g_i: το πραγματικό κόστος από τον s στον κόμβο n_i που εξετάζεται τώρα
- h_i: το υπολογιζόμενο κόστος από τον κόμβο n_i που εξετάζουμε τώρα μέχρι τον τελικό κόμβο g
- t_i: Η πραγματική αποτίμηση από το s στον κόμβο n_i που είναι υπό εξέταση (αφορά τις προτιμήσεις του χρήστη)
- u_i: Η υπολογιζόμενη αποτίμηση στο g βάση της ευρετικής συνάρτησης u (user preferences)

Επεξήγηση αλγορίθμου:

- Εάν υπάρχει ακριβώς ένας βέλτιστος διάδοχος τ.ω $|g_1 + h_1 – g_2 –h_2| > \varepsilon$,

η διαδικασία αναζήτησης δουλεύει ως συνήθως δηλαδή παίρνουμε το πιο σύντομο μονοπάτι.

- Στην περίπτωση $|g_1 + h_1 – g_2 –h_2| \leq \varepsilon$, και τα 2 μονοπάτια από το s στο g είναι δυσδιάκριτα στο h

Γενικά εάν ισχύει περίπτωση $|g_1 + h_1 – g_2 –h_2| \leq \varepsilon$, τότε άλλα κριτήρια λαμβάνονται υπόψη για τη λήψη αποφάσεων (user preferences).

Έστω 2 διαδρομές;

Α διαδρομή: 30 λεπτά , 3 εναλλαγές

Β διαδρομή: 25 λεπτά , 6 εναλλαγές

Για να καταλάβουμε την έννοια του ε, ας υποθέσουμε ότι οι 2 διαδρομές από το s στο g παίρνουν 30 λεπτά και 25 λεπτά αντίστοιχα. ‘Ομως, η πρώτη διαδρομή χρειάζεται πιο λίγες εναλλαγές μεταφορικών μέσων από ότι η $2^{η}$. Εάν ο χρήστης απεχθάνεται τις εναλλαγές, η πρώτη διαδρομή είναι βέλτιστη σύμφωνα με τις προτιμήσεις του χρήστη u κάτω από την υπόθεση ενός ε-tollernance ε≥ 3 λεπτά, δλδ το ε μπορεί να θεωρηθεί κάτι σαν κατώφλι (threshold).

Για να διαλέξουμε ένα μονοπάτι που να είναι βέλτιστο ως προς το h και ταυτόχρονα ως προς το u, υπολογίζουμε το $p_1= t_1 + u_1$ και $p_2= t_2+u_2$ των 2 μονοπατιών.
Τα p_1 και p_2 είναι διανύσματα που αντιπροσωπεύουν όλα τις προτιμήσεις του χρήστη στο u. Κάθε διάσταση αντιπροσωπεύει μία προτίμηση, από αυτές που είδαμε πριν.

Λόγω του ότι είναι ο χώρος προτιμήσεων του p_1 και p_2 είναι μερικώς διατεταγμένος (δεν μπορείς να βάλεις προτεραιότητες πάνω στις προτιμήσεις), γενικά δεν υπάρχει κάποιο ευδιάκριτο ελάχιστο στοιχείο. Για το λόγω αυτό υπολογίζεται κάποιο pareto-optimal set που περιλαμβάνει όλα τα μονοπάτια που **δεν μπορούν να διακριθούν ούτε από το h ούτε από το u.**
Η τελική επιλογή του μονοπατιού που είναι και hε-optimal και uε-optimal, γίνεται με κάποιο decision procedure. Για να γίνει ο υπολογισμός αυτός χρησιμοποιείται MADL (Multi Attributive Decision Language) γλώσσα προγραμματισμού.

```
decision procedure(MADL- )
ParetoDecision doWhat{
        ALT [bus293]
        ALT{Wait,bus288}
        ALT{walk}
        GOAL MIN!(waiting time)
        GOAL MIN!(changes)
        GOAL MIN!(tripDuration)
}
```

Πιο πάνω έχουμε ένα παράδειγμα της λογικής του αλγορίθμου. Το doWhat αποτελείται από 3 εναλλακτικές επιλογές: να πάρουμε το λεωφορείο 293, να περιμένουμε λίγο και να πάρουμε το λεωφορείο 288 ή να περπατήσουμε. Ο στόχος είναι να ελαχιστοποιήσουμε τρία

χαρακτηριστικά που είναι χρόνος αναμονής, διάρκεια ταξιδιού και ο αριθμός εναλλαγών. Και αυτό δίνει την τελική επιλογή μονοπατιού.

Για να βελτιστοποιήσουμε τα αποτελέσματά τρέχουμε τον he-u αλγόριθμο n φορές χρησιμοποιώντας n διαφορετικά ευρετικά. Ταξινομούμε τις n καλύτερες λύσεις

Χρονική πολυπλοκότητα αλγορίθμου: Εξαρτάται από τις ευρετικές συναρτήσεις που χρησιμοποιούνται (οι οποίες δεν είναι γνωστές).

2.2.2 Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns

Στο άρθρο [14] προτείνεται ένας αλγόριθμος που υπολογίζει βέλτιστες διαδρομές σε ένα μεγάλου μεγέθους δίκτυο δημόσιας μεταφοράς μέσα σε ένα μέσο χρόνο μερικών χιλιοστών του δευτερολέπτου.
Ο αλγόριθμος αυτός λαμβάνει υπόψη τις μέρες που υπάρχει κίνηση, το περπάτημα μέχρι τους σταθμούς και τα κόστη πολλαπλών κριτηρίων (συνδυασμός κριτηρίων).
Ο αλγόριθμος βασίζεται σε 2 παρατηρήσεις:

α. Πολλά από τα πιο σύντομα μονοπάτια έχουν ένα κοινό μοτίβο εναλλαγών (transfer pattern) Παράδειγμα. Έως ένα σημείο θα περάσουμε σίγουρα από κάποιους κοινούς σταθμούς

β. Οι απευθείας διαδρομές χωρίς αλλαγή μέσου μπορούν να υπολογιστούν γρήγορα (10 ms)

Ουσιαστικά κατάφεραν να επιταχύνουν τον υπολογισμό βέλτιστης διαδρομής της υπηρεσίας δημόσιας μεταφοράς σε Google Maps.

Γενική Ιδέα:

- Έστω το query A@t->B όπου A(Freiburg) η περιοχή εκκίνησης, B (Zurich) ο προορισμός και t(10:00) ο χρόνος αναχώρησης.
- Έστω ότι κάθε βέλτιστο μονοπάτι από το A στο B είναι είτε απευθείας (χωρίς εναλλαγές μέσου) είτε με μία ακριβώς αλλαγή σε κάποιο σταθμό C (Basel).
- Έστω ότι έχουμε στην διάθεσή μας και τα χρονοδιαγράμματα των τραίνων.

Path	Station	Arrival	Departure
1) Freiburg-Zurich	Freiburg	-	12.55
	Zurich	14:52	-
2)Freiburg-Basel-Zurich	Freiburg	10:02	-
	Basel	10:47	11:07
	Zurich	14:52	-

Πίνακας 2.3. Χρονοδιάγραμμα τρένων

Τότε εύκολα μπορούμε να υπολογίσουμε την επόμενη απευθείας σύνδεση από μια δεδομένη στάση στην επόμενη. Βρίσκουμε την επόμενη άμεση σύνδεση από το Freiburg μέχρι το Zurich. Έστω ότι φεύγει στις 12.55 και φτάνει στις 14:52.
Επίσης βρίσκουμε την επόμενη άμεση σύνδεση από το Freiburg στο Basel και επίσης από το Basel στο Zurich. Έστω ότι φεύγει στις 10:02 από το Freiburg, φτάνει στις 10:47 στο Basel, φεύγει στις 11:07 από το Basel και φτάνει στο Zurich στις 14:52 (Πίνακας 2.3).
Αυτές οι 2 διαδρομές δεν μπορούν να συγκριθούν μεταξύ τους , δλδ δεν μπορούμε πια διαδρομή θα είναι η πιο βέλτιστη διότι αυτό εξαρτάται από τις προτιμήσεις του χρήστη (λιγότερες εναλλαγές ή πιο γρήγορη διαδρομή) και υπολογίστηκαν με μόνο 3 queries άμεσης σύνδεσης.
Το σύνολο όλων των optimal transfer patterns μεταξύ όλων των δυάδων των σταθμών είναι πολύ μεγάλο για να προϋπολογιστεί και να αποθηκευτεί. Γι' αυτό προϋπολογίζεται ένα σύνολο από μέρη/κομμάτια των transfer patterns έτσι ώστε όλα τα optimal transfer patterns να μπορούν να προκύψουν από αυτά τα κομμάτια.
Μάλιστα για τα 3 πιο πάνω data sets που αναφέραμε προϋπολογίζονται τέτοια κομμάτια σε 20-40 core hours ανά 1 εκατομμύριο departure/arrival events και μπορούν να αποθηκευτούν σε 10-50 MB ανά 1000 σταθμούς.
Αφού έχουν προϋπολογιστεί αυτές οι πληροφορίες τότε υπολογίζονται όλοι οι συνδυασμοί που παράγουν ένα transfer pattern από το A στο B. Δημιουργείται έτσι ένα query graph. Παράδειγμα: για το πρόβλημα που είδαμε πιο πάνω το query graph έχει 3 κόμβους , τον A=Freiburg τον B=Zurich και τον C=Basel και 3 ακμές A->B, A->C, C->B.
Γενικά όμως, λόγω του ότι δημιουργούνται και μη-βέλτιστα transfer patterns από τον

συνδυασμό των προϋπολογισμένων κομματιών, τα query graphs αποτελούνται από εκατοντάδες ακμές. Αυτό όμως δεν μας εμποδίζει αφού οι απευθείας συνδέσεις μπορούν να υπολογιστούν σε περίπου 10 μs. Άρα η μέθοδος αυτή δουλεύει σε μεγάλου δίκτυα.

Βασικός αλγόριθμος:

Έστω ότι δεν υπάρχουν διαδρομές με εναλλαγές μέσου.

1. Προϋπολογίζονται όλες οι διαδρομές και τις ομαδοποιούμε σε γραμμές L1,L2,.. Ln έτσι ώστε όλες οι διαδρομές μίας γραμμής να έχουν κοινή αλληλουχία σταθμών και το ίδιο (σχετικά) penalty μεταξύ 2 σταθμών. Penalty είναι η χρονική απόσταση μεταξύ 2 σταθμών.
Οι διαδρομές μιας γραμμής ταξινομούνται βάση του χρόνου και αποθηκεύονται σε ένα δισδιάστατο πίνακα.

Line L17	**S154**	**S097**	**S987**	**S111**	**...**
Trip 1	8:15	8:22 8:23	8:27 8:29	8:38 8:39	...
Trip2	9:14	9:21 9:22	9:28 9:28	9:37 9:38	...
…	...	...	...	...	...

Πίνακας 2.4. Δισδιάστατος πίνακας διαδρομών μιας γραμμής

2. Για κάθε σταθμό, υπολογίζεται η θέση του σε κάθε line δλδ εάν εμφανίζεται σε κάποια γραμμή τότε θα δούμε σε ποια θέση του πίνακα εμφανίζεται.

Παράδειγμα: S097: (L8,4) (L17,2) (L34,5) (L87,17) …

S111: (L9,1) (L13,5) (L17,4) (L55,16)…

3. Για τον υπολογισμό του query A@t->B βρίσκουμε από τους πιο πάνω πίνακες το νωρίτερο δυνατό μονοπάτι και μετά διαλέγουμε αυτό με τα βέλτιστα κόστη.

2.2.3 Alternative Route Graphs in Road Networks

Αυτό το άρθρο[15] ασχολείται με τον υπολογισμό εναλλακτικών διαδρομών μεταξύ αφετηρίας και προορισμού πέραν από την βέλτιστη-πιο σύντομη διαδρομή.
Το πρόβλημα ανήκει στα NP δύσκολα προβλήματα και για την λύση του προτείνονται κάποιοι ευρετικοί μέθοδοι.

Γενικά υπάρχουν πολλές εναλλακτικές διαδρομές μεταξύ της αφετηρία και προορισμού που πλησιάζουν την βέλτιστη λύση (ως προς την απόσταση, την ώρα μέχρι τον προορισμό…). Είναι πολλοί λόγοι για τους οποίους ένας άνθρωπος να χρειάζεται να έχει επιλογές στην διάθεσή του. Για παράδειγμα κάποιος μπορεί να έχει κάποια προσωπική προτίμηση δρόμου, ξέρει καλά τον εναλλακτικό δρόμο ή μπορεί να μην θέλει να πάρει την διαδρομή που οδηγεί μέσα από μια πόλη και να προτιμά τον αυτοκινητόδρομο. Επιπλέον, οι διαδρομές μπορεί να έχουν διαφορετικά χαρακτηριστικά, δλδ μπορεί μια διαδρομή να είναι πιο οικονομική από μια άλλη, από άποψη κατανάλωσης βενζίνης , φόρων στα διόδια κτλ.
Με το να υπολογιστούν εναλλακτικές διαδρομές , κάποιος μπορεί να επιλέξει την διαδρομή που γι’ αυτόν είναι η ιδανική.

Alternative graphs (AG)

Ορισμός: Ένας εναλλακτικός γράφος είναι γράφος που αφορά την ένωση πολλών μονοπατιών από την εκκίνηση στον προορισμό.

Μαθηματικός ορισμός:

- G=(V,E) γράφος με συνάρτηση βάρους w: E -> R_+
- s source, t destination
- AG H=(V’,E’), V’ $\subseteq$ V, e $\in$ E’
- $\exists$ path s-t τ.ω το e να ανήκει σε αυτό το path.

Για κάποιο σημείο εκκίνησης s και κάποιο προορισμό t ένας alternative γράφος AG H=(V’,E’) είναι ένας γράφος του οποίου οι κόμβοι V’ είναι υποσύνολο των κόμβων V και για κάθε ακμή e που ανήκει στο σύνολο των ακμών E’, υπάρχει ένα απλό μονοπάτι από το s στο t που περιλαμβάνει το e.

Μεθόδοι για τον υπολογισμό εναλλακτικών μονοπατιών-Χρήση ευρετικών

k-shortest paths

Η ιδέα είναι να υπολογιστούν τα k μικρότερα μονοπάτια . Αυτό όμως οδηγεί στον υπολογισμό μονοπατιών που απέχουν ελάχιστο από το βέλτιστο μονοπάτι χωρίς ουσιαστικές διαφορές μεταξύ τους. Έτσι, τα εναλλακτικά μονοπάτια που υπολογίζονται είναι τόσο κοντά το ένα στο άλλο που από κάποιο άνθρωπο μπορεί να θεωρηθούν ως μη διακριτά. Για να

υπολογιστούν διακριτά μονοπάτια πρέπει ο αριθμός του k να είναι αρκετά μεγάλος. Αυτό όμως θα οδηγήσει στην εύρεση πολλών εναλλακτικών μονοπατιών , πράγμα που δεν το θέλουμε διότι θα συγχύζει τον χρήστη. Επίσης, πρακτικά δεν υπάρχουν αλγόριθμοι που να υπολογίζουν "γρήγορα" τα k-μικρότερα μονοπάτια.

Pareto method

Υπάρχουν πολλά διαφορετικά ήδη βαρών που μπορούμε να λάβουμε υπόψη μας όπως ώρα ταξιδιού, κατανάλωση καυσίμων , γραφικές διαδρομές κτλ .Ακόμα όμως και αν περιοριστούμε σε 1 κυρίως(primary) βάρος μπορούμε να προσθέσουμε μία 2η συνάρτηση βάρους που να έχει τιμή 0 για τις ακμές που βρίσκονται έξω από τον AG και για τις ακμές που βρίσκονται μέσα στο AG να έχουν τιμή ίση με το primary βάρος.

Ένα μονοπάτι είναι pareto optimal εάν δεν υπάρχει άλλο μονοπάτι που να είναι καλύτερο, σε σχέση με τα 2 βάρη. Όλα τα Pareto optimal paths μπορούν να υπολογιστούν με ένα γενικευμένο αλγόριθμο του Dijkstra.

Επειδή όμως πιθανόν να υπάρχουν πολλά pareto optimal μονοπάτια, θα πρέπει μειώσουμε τον αριθμό αυτό κρατώντας τα μονοπάτια αυτά που διαφέρουν επαρκώς μεταξύ τους.

Γι' αυτό, όλα τα μονοπάτια που είναι 1+ε φορές μεγαλύτερα από το μικρότερο μονοπάτι τα κρατάμε.

Επίσης, όλα τα μονοπάτια που το γινόμενο του primary και secondary βάρους είναι 1/ r φορές μεγαλύτερο από κάποιο άλλο μονοπάτι τα κρατάμε.

Το ε και το r είναι ρυθμιστικές μεταβλητές. Υπολογίζουμε λιγότερες διαδρομές με μικρό ε και μεγάλο r.

Με τη μέθοδο αυτή βρίσκουμε μονοπάτια που είναι εντελώς ασύνδετα μεταξύ τους.

Plateau method

Η μέθοδος αυτή αναγνωρίζει γρήγορα highways (plateaus=οροπέδια) και διαλέγει τις καλύτερες διαδρομές βασισμένος στο μήκος της διαδρομής και στο μήκος του highway.

Συγκεκριμένα, εκτελείται ο Dijkstra αλγόριθμος από την εκκίνηση σε όλους τους κόμβους και ένα ανάποδο Dijkstra από τον προορισμό προς την εκκίνηση. Μετά γίνεται μια διασταύρωση των shortest path tree edges των 2 Dijkstra.

Το αποτέλεσμα είναι κάποια μονοπάτια που τα ονομάζουμε plateau.

Penalty method

1. υπολογίζεται η συντομότερη διαδρομή
2. Προσθέτουμε την διαδρομή αυτή στη λίστα με τις λύσεις
3. Αυξάνουμε τα βάρη σε αυτό το μονοπάτι
4. Επιστρέφουμε πίσω στο βήμα 1

Η βασική ιδέα είναι να υπολογιστεί η συντομότερη διαδρομή, να την προσθέσουμε στην λύση μας, να αυξήσουμε τα βάρη των ακμών σε αυτό το μονοπάτι και να ξεκινήσουμε από την αρχή μέχρι να είμαστε ευχαριστημένοι με την λύση.

2.3 Σχετικές εφαρμογές

Κάναμε μια έρευνα όσο αφορά κάποια applications που σχετίζονται άμεσα με την εφαρμογή που θέλουμε να αναπτύξουμε. Η σύγκριση έγινε ως προς τις δικές μας απαιτήσεις, εάν δηλαδή περιλαμβάνει αυτά που θέλουμε να πετύχουμε.

- **Transport Urban** (online έκδοση http://transporturban.ro/ro/)

Πρόκειται για μια εφαρμογή που αναπτύχθηκε σε Android OS για την περιοχή της Ρουμανίας Βουκουρέστι και αφορά αποκλειστικά αστικό δίκτυο συγκοινωνίας. Βοηθά τους χρήστες του να ταξιδέψουν γρήγορα και με ασφάλεια χρησιμοποιώντας τις δημόσιες συγκοινωνίες στο Βουκουρέστι. Οι χρήστες αλληλεπιδρούν με το χάρτη της πόλης του Βουκουρεστίου και έχουν στη διάθεσή τους κάποια εργαλεία που θα τους βοηθήσουν να οργανώσουν τα ταξίδια τους. Οι χρήστες επιλέγουν σημείο εκκίνησης και σημείο προορισμού και τους παρουσιάζεται ο καλύτερος τρόπος να πάνε από το ένα σημείο στο άλλο μέσω οδηγιών στην οθόνη και μέσω γραφικής αναπαράστασης στο χάρτη. Επιπλέον, η εφαρμογή προσφέρει στους χρήστες του αναζήτηση διεύθυνσης και σημείων ενδιαφέροντος ενώ με το πάτημα ενός κουμπιού εμφανίζεται και η γεωγραφική θέση του χρήστη πάνω στον χάρτη.

Σχολιασμός:
Η εφαρμογή αυτή έχει κάποια βασική υλοποίηση ενός Trip Planner. Παρόλο όμως που η επιλογή σημείου εκκίνησης και προορισμού γίνεται με την αλληλεπίδραση με τον χάρτη(Pin στο χάρτη) εντούτοις η εφαρμογή δεν προσφέρει δυνατότητα δημιουργίας διαδρομής μέσω διεύθυνσης (Geolocation). Αυτό προκαλεί ερωτήματα καλής ευχρηστίας. Για παράδειγμα, ένας τουρίστας δεν θα έχει ιδέα του τι βρίσκεται και που βρίσκεται στον χάρτη της Ρουμανίας άρα θα του ήταν άχρηστη η εφαρμογή αυτή. Θα ήταν πιο εύκολο για αυτόν να κάνει αναζήτηση διαδρομής βάση αρχικής και τελικής διεύθυνσης. Επίσης, η παρουσίαση των οδηγιών στον χρήστη δεν είναι αρκετά λεπτομερής ενώ παράλληλα τα κόστη επιβίβασης δεν είναι διαθέσιμα. Άλλο ένα μειονέκτημα της εφαρμογής αυτής είναι ότι δεν δουλεύει offline. Απαραίτητη προϋπόθεση είναι να υπάρχει σύνδεση με το διαδίκτυο . Αυτό επιβαρύνει τον χρήστη εφόσον θα πρέπει για κάθε προγραμματισμό επανειλημμένης

διαδρομής να υπάρχει σύνδεση με το διαδίκτυο. Τέλος, δεν υπάρχει αποθήκευση των πρόσφατων διαδρομών που αναζήτησε ο χρήστης.

- **Maps on Android**

Η εφαρμογή αυτή είναι προεγκατεστημένη στις κινητές πλατφόρμες που τρέχουν λειτουργικό σύστημα Android. Πρόκειται για μια εφαρμογή βασισμένη σε Google Maps, η οποία αναπτύχθηκε από την Google Inc. Παρέχει κάποιο βασικό Trip Planner στο οποίο η αναζήτηση γίνεται αποκλειστικά βάση αρχικής και τελικής διεύθυνσης ενώ τα αποτελέσματα εμφανίζονται τόσο στο χάρτη όσο και σε μορφή κειμένου και είναι αρκετά λεπτομερής. Η εφαρμογή προσφέρει υπηρεσίες σε παγκόσμιο επίπεδο, δηλαδή δεν αφορά αποκλειστικά μια περιοχή.

Σχολιασμός:
Η εφαρμογή εκ πρώτης όψεως φαίνεται να εκτελεί τον σκοπό για τον οποίο έχει αναπτυχθεί, που είναι να παρέχει οδική καθοδήγηση στους χρήστες της και φαίνεται να είναι αρκετά μελετημένη. Παρόλα αυτά όμως υστερεί σε κάποιες βασικές λειτουργίες. Πρώτον, η εφαρμογή αυτή αδυνατεί να λειτουργήσει εάν δεν υπάρχει σύνδεση με το διαδίκτυο. Υπάρχει caching για τους χάρτες όταν είναι offline, εντούτοις δεν υπάρχει δυνατότητα αποθήκευσης πρόσφατων διαδρομών που αναζήτησε ο χρήστης. Δεύτερον, παρόλο που προσφέρεται η δυνατότητα δρομολόγησης βάση διευθύνσεων, δεν προσφέρεται η δυνατότητα δρομολόγησης βάση επιλογής σημείου εκκίνησης και προορισμού πάνω στο χάρτη. Τρίτον, παρόλο που προσφέρεται η δυνατότητα χρήσης δημόσιων μέσων μεταφοράς, εντούτοις οι χρήστες δεν μπορούν να αναζητήσουν τα δρομολόγια και χρονοπρογραμματισμό των υπηρεσιών αυτών. Τέλος, η εφαρμογή δεν προσφέρει υπηρεσίες για notifications. Για παράδειγμα εάν χρησιμοποιώ υπηρεσία λεωφορείων και πρέπει να κατεβώ στην επόμενη στάση, τότε καλό θα ήταν η εφαρμογή να με ειδοποιεί.

- **TripPlanner**

Πρόκειται για μια εφαρμογή που αναπτύχθηκε σε Android OS και αφορά την περιοχή New South Wales της Αυστραλίας. Περιλαμβάνει την επιλογή για Trip Planning στην οποία ο χρήστης δίνει σημείο εκκίνησης και σημείο προορισμού βάση διεύθυνσης. Τα αποτελέσματα που δίνονται στο χρήστη είναι σε μορφή κειμένου μόνο. Η εφαρμογή αυτή

δίνει την δυνατότητα στον χρήστη να δει το ιστορικό του, δηλαδή τις διαδρομές που σχεδίασε στο παρελθόν και για αυτή την λειτουργία η εφαρμογή δουλεύει offline.

Σχολιασμός:
Αυτή η εφαρμογή ξεχωρίζει από τις άλλες που είδαμε διότι επιτρέπει στον χρήστη να έχει πρόσβαση στα πρόσφατα ταξίδια που σχεδίασε. Αποθηκεύονται λοιπόν locally τα ταξίδια που σχεδίασε και επομένως η πρόσβασή τους μπορεί να γίνει offline. Πάραυτα όμως, υστερεί σε θέματα ευχρηστίας. Η όλη εφαρμογή δεν εμφανίζει κάποια διαπροσωπεία χάρτη με την οποία ο χρήστης μπορεί να αλληλεπιδράσει και να δει την διαδρομή πάνω στον χάρτη. Επιπλέον, παρόλο που εμφανίζονται οι οδηγίες ταξιδιού, εντούτοις δεν εμφανίζονται πουθενά πληροφορίες σχετικά με το χρηματικό κόστος της διαδρομής.

- **BayTripper**

Πρόκειται για μια εφαρμογή που αναπτύχθηκε σε iOS και αφορά την περιοχή του San Francisco. Αναπτύχθηκε από 2 φοιτητές στο Πανεπιστήμιο του Berkeley με σκοπό να παροτρύνουν τους ανθρώπους να ξεφύγουν από τα ιδιωτικά τους αυτοκίνητα και να χρησιμοποιήσουν τα πόδια τους, τα ποδήλατα και τα δημόσια μέσα μεταφοράς.

Σχολιασμός:
Περιλαμβάνει λειτουργίες trip planning με δυνατότητες επιλογής start point και end point με την βοήθεια του χάρτη (Google Maps) και με επιλογή διεύθυνσης. Οι οδηγίες που εμφανίζονται στους χρήστες είναι αρκετά λεπτομερής και περιλαμβάνει πληροφορίες σχετικά με τα χρηματικό κόστος του ταξιδιού. Ένα από τα μειονεκτήματα της εφαρμογής αυτής είναι ότι δεν παρέχει την δυνατότητα για ανάληψη πρόσφατων δρομολογίων που σχεδίασε ο χρήστης.

Συνοψίσαμε τα αποτελέσματα στον πιο κάτω πίνακα

Characteristics	Transport Urban	Maps (Android)	TripPlanner	Bay tripper	EasyTransit
Look up Schedules	✖	✖	✖	✔	✔
Fares	✖	✔	✖	✔	✔
Map Interface	✔	✔	✖	✔	✔
Trip Planner	✔	✔	✔	✔	✔
Map Based Trip planner (pins on map)	✔	✖	✖	✔	✔
Address Based Trip Planner	✖	✔	✔	✔	✔
Recent Trips Offline Access	✖	✖	✔	✖	✔
Many Areas Covered on single App	✖	✔	✖	✖	✔
Notifications based On GPS Location	✖	✖	✖	✖	✔
Voice Directions for people with reading problems	✖	✖	✖	✖	✔
Step By Step Instructions	✖	✔	✔	✖	✔
Multiple languages	✔	✖	✖	✖	✖
Multimodal trip planning	✖	✖	✖	✖	✖
POI trip planning	✔	✔	✖	✖	✖
Real Time Information	✖	✖	✖	✔	✖

Πίνακας 2.5. Σύγκριση σχετικών εφαρμογών

Συνοψίζοντας, παρατηρούμε ότι σχεδόν όλες οι εφαρμογές που μελετήσαμε δεν περιλαμβάνουν offline υποστήριξη. Επίσης, καμιά από τις εφαρμογές δεν υποστηρίζει (ειδοποιήσεις) notifications και φωνητικές οδηγίες. Η διαπροσωπεία του χάρτη είναι άκρως απαραίτητη για τέτοιου είδους εφαρμογές και γι' αυτό οι πλείστες εφαρμογές περιλαμβάνουν αυτή την δυνατότητα .

2.4 Open Trip Planner

Το Open Trip Planner (OTP [16]) είναι μία υπηρεσία που βοηθά τους χρήστες του να υπολογίσουν τον καλύτερο τρόπο για να πάνε από ένα σημείο στο άλλο χρησιμοποιώντας κυρίως Δημόσια Μέσα Μεταφοράς. Οι χρήστες μπορούν να χρησιμοποιήσουν πολλαπλά μέσα για να σχεδιάσουν το ταξίδι τους, για παράδειγμα, μπορεί να γίνει με τα πόδια, ποδήλατο, τρένο, λεωφορείο, ή κάποιο συνδυασμό αυτών. Η υπηρεσία αυτή επιτρέπει εύκολα να προσαρμοστεί το ταξίδι ανάλογα με τις προτιμήσεις του καθενός, για παράδειγμα κάποιος μπορεί να μην θέλει να κάνει πολλές μεταβιβάσεις για να φτάσει στον προορισμό του, ή κάποιος μπορεί να θέλει να πάει σε με περιοχή περπατώντας σε ένα πιο γραφικό δρόμο ή μεταφορά με λεωφορεία που διαθέτουν ειδικούς αποθηκευτικούς χώρους για ποδήλατα. Το Open Trip Planner δέχεται ως είσοδο GTFS δεδομένα (υποκεφάλαιο 2.6) ή shapefiles ([17]) και αρχεία .osm (Open Street maps λήψη από [18]) και δημιουργεί ένα γράφο βάση του οποίου γίνεται η αναζήτηση πληροφοριών για την εμφάνιση μιας διαδρομής.

Το OTP περιέχει open source code ([19]) τον οποίο πολλοί developers χρησιμοποιούν για την ανάπτυξη δικών τους εφαρμογών. Συνδυάζει δουλειά από ένα αριθμό από υφιστάμενα open source projects όπως Graphserver, OneBusAway και FivePoints.

Στόχος του Open trip planner εκτός από την βασική λειτουργία που προσφέρει είναι να είναι ευκολόχρηστο προς τους χρήστες ,να είναι ευέλικτο, αξιόπιστο, και γρήγορο καθώς και εύκολα επεκτάσιμο.

Αρχιτεκτονική συστήματος

Για την ανακάλυψη και κατανόηση της γενικής αρχιτεκτονικής του συστήματος OTP έγινε μία μελέτη του εγγράφου [20] που αποτελεί μια τελική αναφορά στην οποία περιγράφεται η όλη διαδικασία ανάπτυξης του συστήματος, οι σκοποί του συστήματος καθώς και οι λόγοι που το οδήγησαν σε ανοικτό πηγαίο κώδικα.

Εμείς θα επικεντρωθούμε στην υποενότητα που αναφέρεται στην αρχιτεκτονική του συστήματος ([20],σελ. 22-24).

Λόγω του ότι η δρομολόγηση διαδρομών γενικά θεωρείται ως ένα αρκετά πολύπλοκο σύστημα και λόγω του ότι μπορεί να χρησιμοποιηθεί υπό διαφορετικές συνθήκες (κατά την οδήγηση ή στο σπίτι) θεωρήθηκε σωστό διασπαστεί σε διάφορα μικρότερα καθιστώντας έτσι υψηλό βαθμό modularity. Έτσι, τα βασικά στοιχεία του συστήματος αναπτύσσονται ξεχωριστά ενώ ταυτόχρονα αποτελούν κομμάτι του μεγαλύτερου συστήματος.

Περιγραφή βασικών τμημάτων

Core Routing Engine

Το Core Route Engine αποτελείται από 2 sub-modules, το graph manager και το route server. Αυτό το component περιλαμβάνει τον κώδικα που επεξεργάζεται τα αποθηκευμένα δεδομένα ,δημιουργεί το γράφο και τρέχει τον αλγόριθμο που παράγει την καλύτερη διαδρομή στο δίκτυο σύμφωνα με τα δεδομένα εισόδου.

Narrative Engine

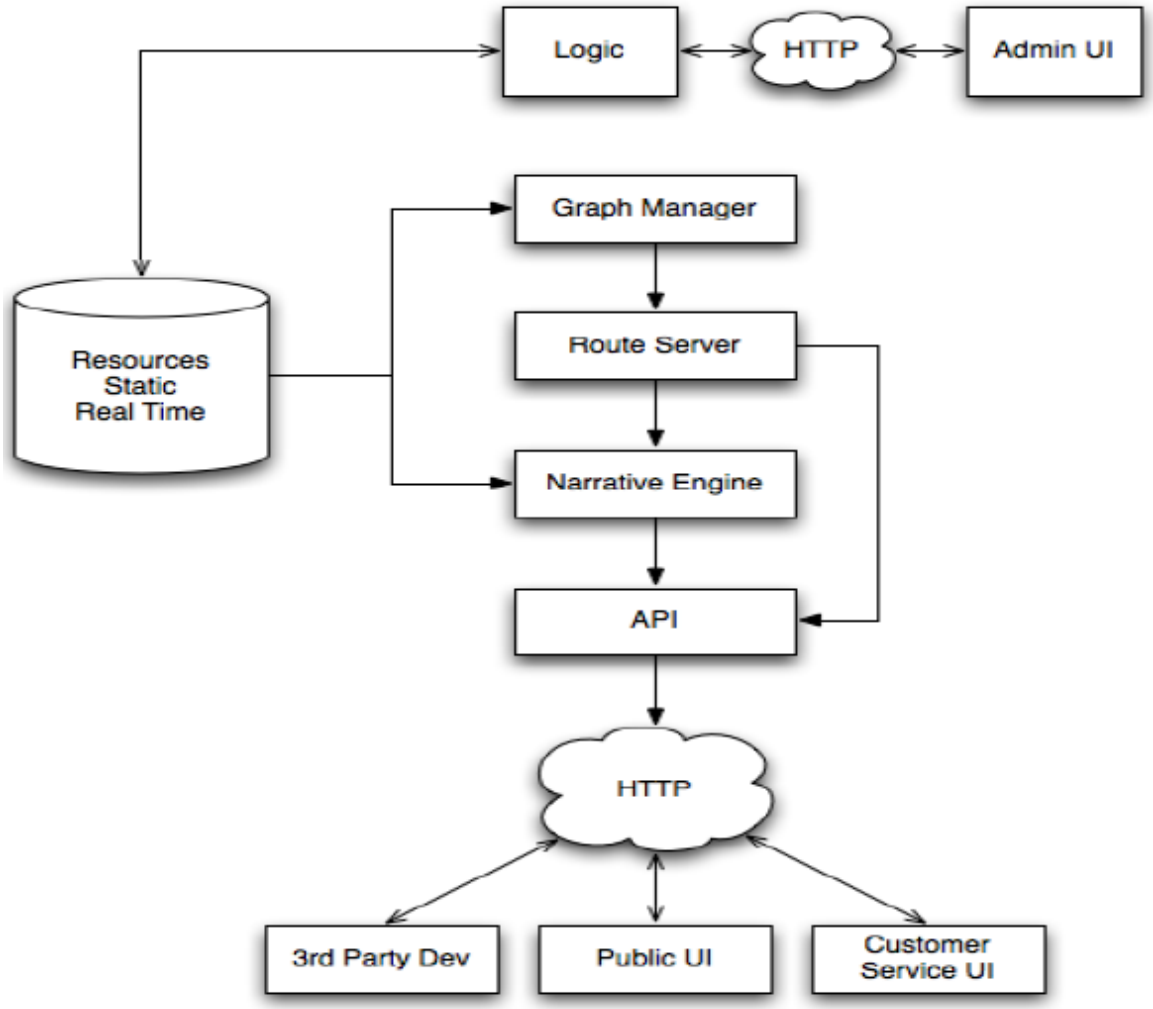
Αυτό το component δημιουργεί την τελική παρουσίαση/αναπαράσταση διαδρομής από την διαδρομή που δημιουργήθηκε στο γράφο έτσι ώστε ο χρήστης να μπορεί να την αντιληφθεί οπτικά.

“Resources” (data store and manager) and logic box

Αυτό το κομμάτι επιτρέπει την φόρτωση και την χρήση υφιστάμενων datasets όπως transit graphs και schedules. Το datastore επικοινωνεί με το Narrative Engine επιτρέποντας έτσι την υπηρεσία συμβουλών (advisor service) να γίνει overlayed στην τελική αναπαράσταση της διαδρομής. Περιλαμβάνει κυρίως στατικά δεδομένα (GTFS schedule files). Πρόσφατα έγινε η ενσωμάτωση και real-time δεδομένων.

Front-end user interface (widgets)

Αυτό το κομμάτι αφορά την προσθήκη δυνατότητας να χρησιμοποιηθεί το σύστημα (το querying) από μια ιστοσελίδα. Ο σκοπός αυτού του module είναι στο μέλλον να μπορεί να δημιουργηθεί ένα off-the-shelf toolbox για τους προγραμματιστές για να μπορούν να κατασκευάζουν εύκολα εφαρμογές σχετιζόμενες με τα δημόσια μέσα μεταφοράς.



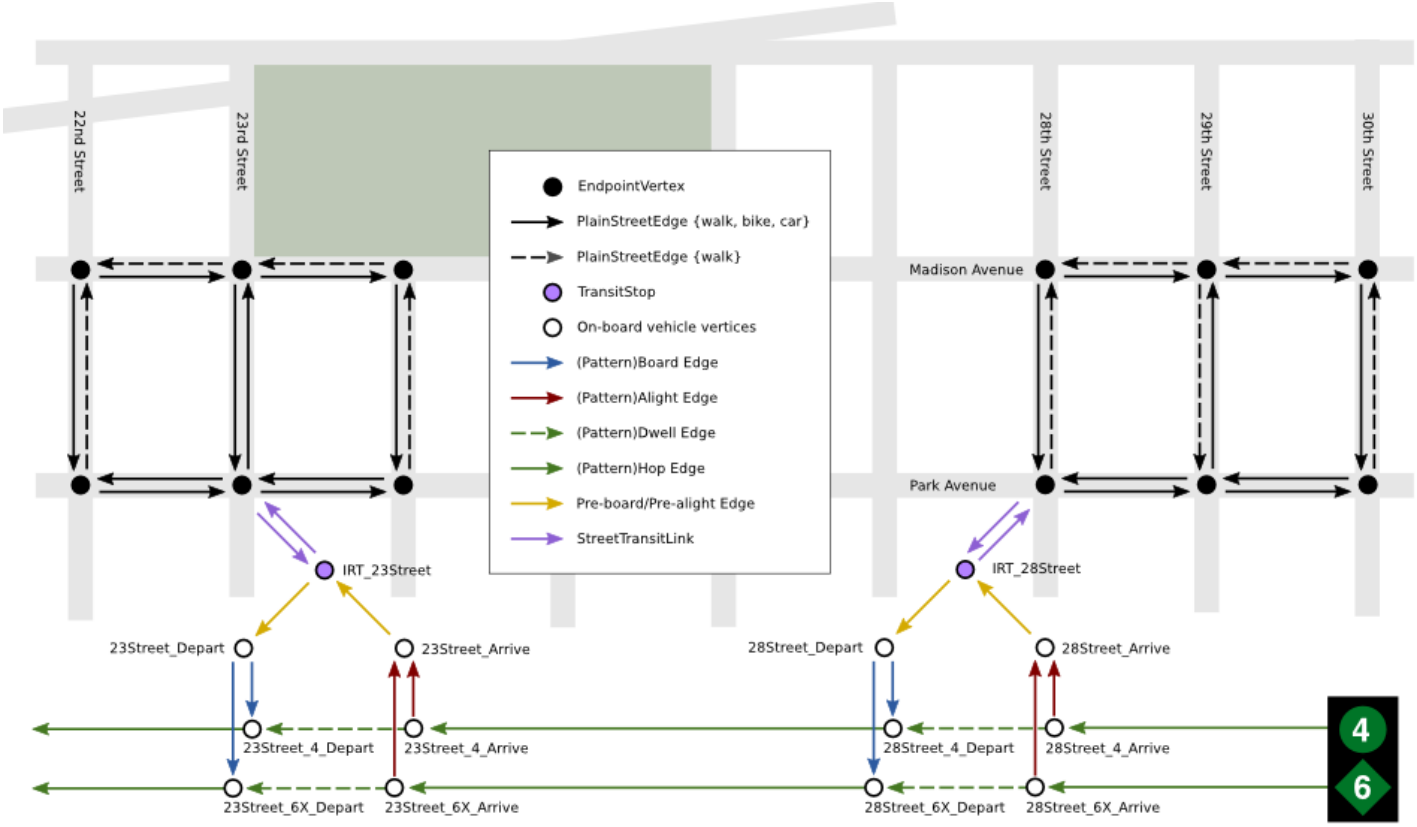
Διάγραμμα 2.6. Αρχιτεκτονική OTP

Τα δεδομένα για την κατασκευή του γράφου λαμβάνονται από την βάση δεδομένων που περιλαμβάνει στατικά δεδομένα (GTFS) και πραγματικού χρόνου δεδομένα. Η ενημέρωση αυτών των δεδομένων γίνεται από κάποιον administrator ο οποίος επικοινωνεί μέσω http με το Logic box και έχει την δυνατότητα να κάνει αλλαγές πάνω στα δεδομένα. Ο route server ανάλογα με τα queries (API), τρέχει κάποιο αλγόριθμο εύρεσης καλύτερης διαδρομής και τα αποτελέσματα παρουσιάζονται στο χρήστη με τη χρήση του module Narrative Engine. Το narrative engine αντλεί επίσης δεδομένα από τη βάση δεδομένων όπως schedules για την υπηρεσία συμβουλών (Παράδειγμα: παρουσιάζεται το δρομολόγιο μιας διαδρομής πάνω στο χάρτη). Τέλος, οι διάφοροι ενδιαφερόμενοι του OTP μπορούν να χρησιμοποιήσουν το σύστημα ή να θέσουν οποιεσδήποτε απορίες διαμέσου Http μέσω του API που παρέχεται.

Γράφος OTP:

Όπως αναφέραμε και προηγουμένως, για την δημιουργία του γράφου, το OTP δέχεται ως είσοδο τα αρχεία GTFS και ένα αρχείο .osm που περιέχει πληροφορίες σχετικά με μια περιοχή. Αρχικά τα 2 αυτά δεδομένα φορτώνονται στο σύστημα σαν 2 εντελώς ασύνδετοι υπογράφοι. Στη συνέχεια κατασκευάζεται ένα χωρικό ευρετήριο για όλες τις ακμές δρόμων από το αρχείο .osm. Για αυτό το λόγο, για μεγάλα αρχεία .osm χρειάζεται να δεσμευτεί αρκετή μνήμη RAM αλλιώς υπάρχει το γνωστό πρόβλημα με την μνήμη του σωρού της java (java heap exception). Τότε, διασχίζονται μέσα σε ένα βρόγχο οι στάσεις από το αρχείο GTFS και ανευρίσκονται όλοι οι κοντινότεροι δρόμοι για κάθε στάση, χρησιμοποιώντας το ευρετήριο που κατασκευάστηκε. Έπειτα συνδέονται οι στάσεις με τις ακμές των δρόμων με μια κάθετη γραμμή, και η γραμμή αυτή μοιράζεται ανάλογα ούτως ώστε να συνδεθούν τα 2 άκρα με ακμή χαμηλού βάρους.

Στο [21] περιγράφεται με λεπτομέρεια η δομή που ακολουθεί ο γράφος του OTP.



Διάγραμμα 2.7. Ένας γράφος δρόμων συνδεδεμένος με ένα γράφο διέλευσης (Μανχάταν-Νέα Υόρκη)

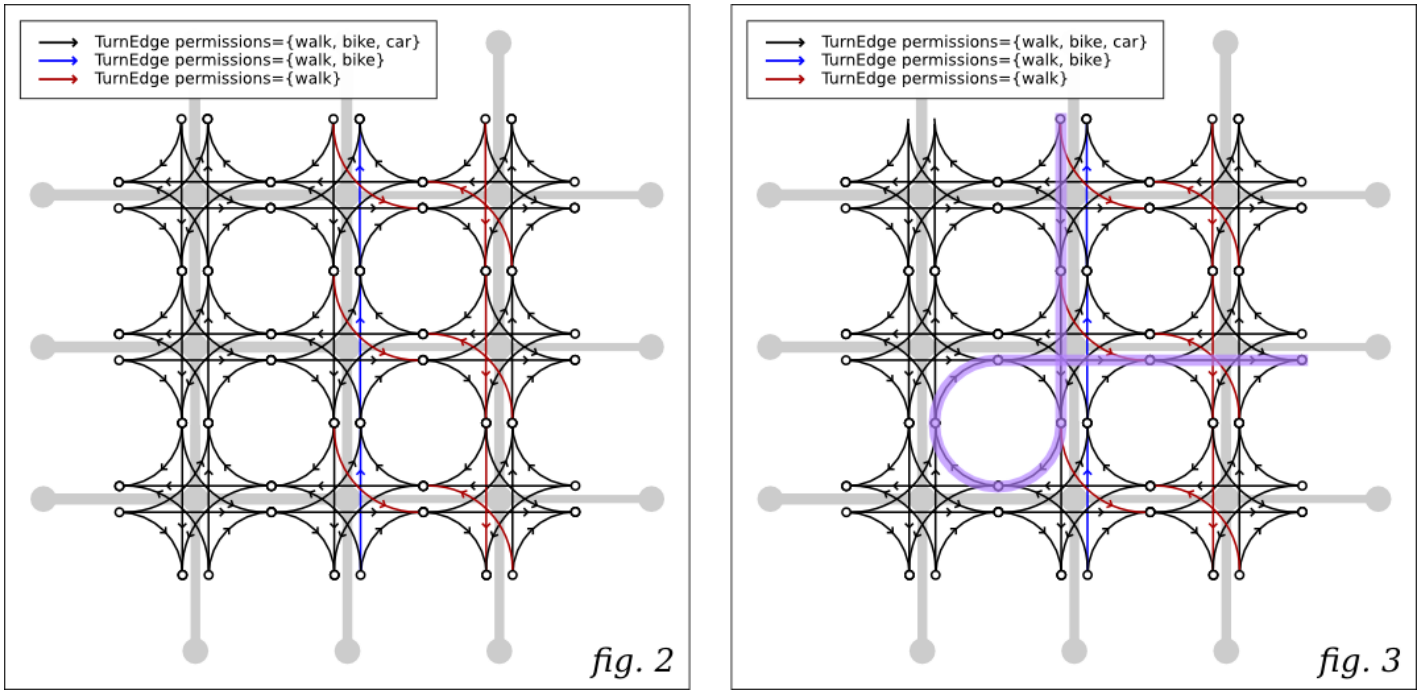
Το πιο πάνω σχήμα παρουσιάζει ένα OTP γράφο για μια κεντρική περιοχή του Μανχάταν και περιλαμβάνει πληροφορίες από τα αρχεία .osm και GTFS. Το μοντέλο που χρησιμοποιείται εδώ έχει εξελιχθεί (βλέπε πιο κάτω) αλλά θα το αναφέρουμε ούτως ώστε να καταλάβουμε από πού προέκυψε το καινούριο μοντέλο. Σε αυτό το μοντέλο, οι κόμβοι (end point vertices)

αναπαριστούν διασταυρώσεις δρόμων και οι ακμές αναπαριστούν τμήματα δρόμων μεταξύ των διασταυρώσεων. Λόγω του ότι τα δρομολόγια των ΔΜΜ μπορεί να έχουν 2 κατευθύνσεις (διαφορετικά θα πάμε από το X στο Y και διαφορετικά θα πάμε από το Y στο X) ο γράφος λέμε ότι είναι κατευθυνόμενος, δλδ για κάθε σύνδεση από ένα κόμβο Α σε ένα κόμβο Β υπάρχει ζεύγος ακμών με αντίθετη κατεύθυνση.

Κάθε σταθμός μπορεί να έχει πολλαπλούς κόμβους στην περίπτωση που υποστηρίζει πολλές διαφορετικές πλατφόρμες. Κάθε ένας από αυτούς τους κόμβους ενώνεται με το δίκτυο δρόμων (street network) . Στο πιο πάνω σχήμα φαίνονται 2 σταθμοί που ενώνονται με ακμή StreetTransitLink με τους κόμβους που βρίσκονται στις διασταυρώσεις.

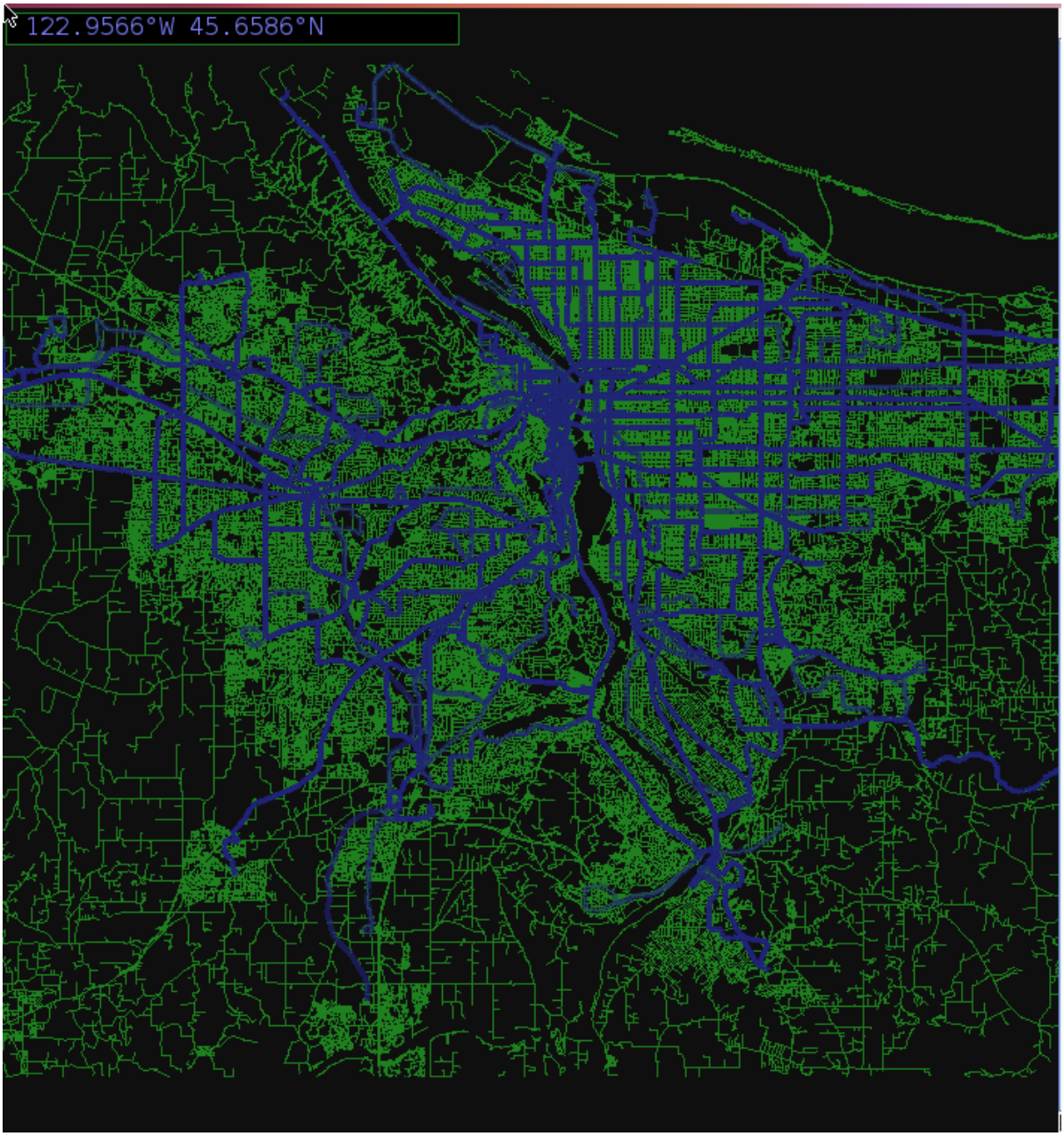
Υπάρχουν διαφορετικού είδους ακμές (Board,Alight,Dwell,Hop) και αντιπροσωπεύουν δρομολόγια. Πολλά δρομολόγια μπορούν να ομαδοποιηθούν μαζί σε πίνακες χρόνου PatternBoard, PatternAlight, PatternDwell και PatternHop εάν έχουν την ίδια σειρά σταθμών. Το βάρος αυτών των ακμών είναι ανάλογο του χρόνου που χρειάζονται για να τις διασχίσουμε καθώς και άλλων παραμέτρων όπως κόστος μετεπιβίβασης, στροφές, περπάτημα ή αναμονή .

Το πρόβλημα που προκύπτει με αυτό το μοντέλο είναι ότι δεν υποστηρίζει περιορισμούς στις στροφές, για παράδειγμα τους μονόδρομους. Έτσι χρειάστηκε να αναπτυχθεί ένα άλλο μοντέλο που ένα είναι υποστηρίζει πιο πολύπλοκη συμπεριφορά.



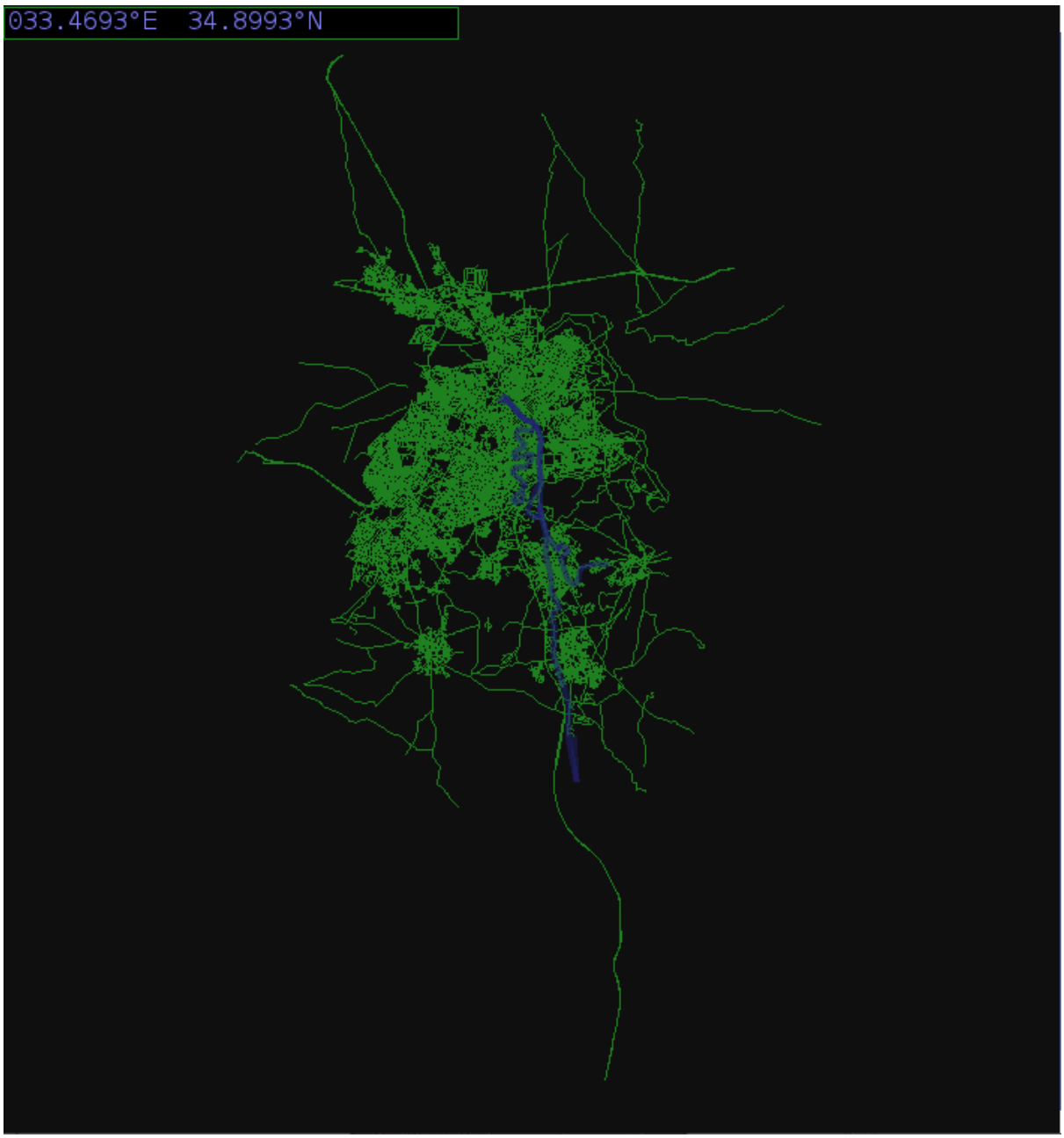
Διάγραμμα 2.8. Υφιστάμενο Μοντέλο Γράφου OTP

Το σύστημα αναπαράστασης του δικτύου των δρόμων του σχήματος 2.7 αντικαταστάθηκε με το μοντέλο αναπαράστασης με ακμές όπως φαίνεται στο σχήμα 2.8. Με γκρίζο χρώμα παρουσιάζεται το παλιό μοντέλο αναπαράστασης και πάνω σε αυτό βρίσκεται το νέο μοντέλο. Κάθε ακμή του παλιού μοντέλου έχει αντικατασταθεί με ένα κόμβο που τώρα αντιπροσωπεύει ένα τμήμα του δρόμου. Κάθε ζεύγος ακμών του παλιού μοντέλου έχει αντικατασταθεί με μία μόνο ακμή και η ακμή αυτή αντιπροσωπεύει την μετακίνηση από ένα τμήμα του δρόμου σε ένα άλλο γειτονικό τμήμα του δρόμου. Το transit network παραμένει ως έχει.

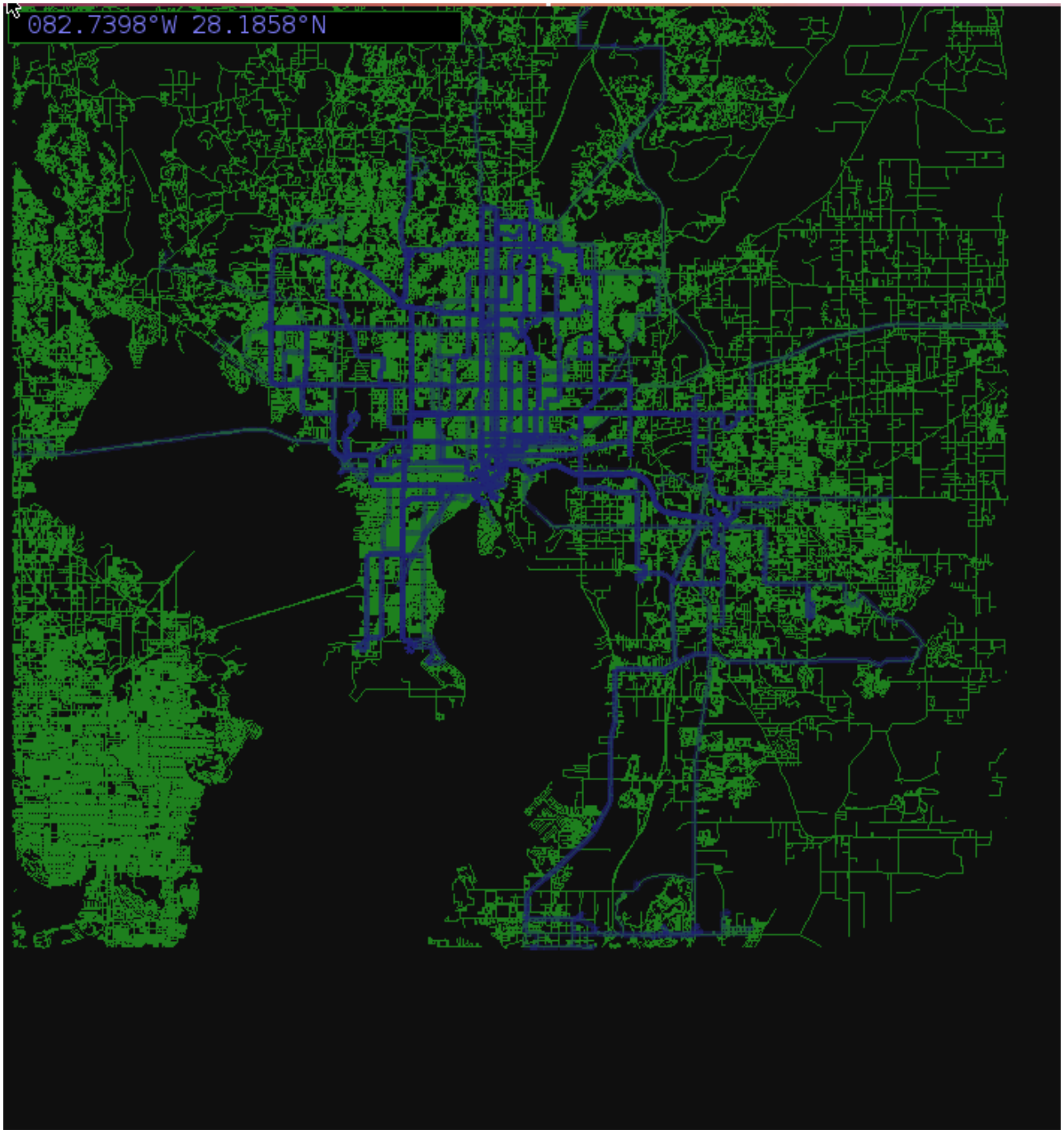


Σχήμα 2.9. Οπτική αναπαράσταση γράφου για περιοχή Portland

Με μπλε είναι το αστικό δίκτυο ενώ με πράσινο το δίκτυο δρόμων



Σχήμα 2.10. Οπτική αναπαράσταση γράφου για περιοχή Λευκωσίας



Σχήμα 2.11. Οπτική αναπαράσταση γράφου για περιοχή Tampa

Υφιστάμενος κώδικας OTP για Android OS

Το 2010 η Marcy Gordon, μία προγραμματίστρια του OTP, ξεκίνησε να γράφει κώδικα για να μεταφέρει το OTP σε Android πλατφόρμα. Ο κώδικας βρίσκεται στο [22]. Το πρόβλημα με τον συγκεκριμένο κώδικα, πέραν του ότι δεν ήταν ολοκληρωμένος, ήταν ότι δεν συμβάδιζε με την εξέλιξη του OTP. Το OTP είναι ένα έργο το οποίο αναπτύσσεται και εξελίσσεται συνεχώς με αποτέλεσμα ο κώδικας που ανέπτυξε η Marcy Gordon να είναι out of date. Εμείς πήραμε αυτό τον κώδικα, τον εκσυγχρονίσαμε και συνεχίσαμε αναπτύσσοντας τις δικές μας ανεξάρτητες λειτουργίες.

2.5 GTFS δεδομένα

Τα **GTFS** που είναι η συντομογραφία του General (ή Google) Transit Data Feed, είναι κάποια αρχεία που αφορούν προγράμματα (schedules) διακίνησης του πληθυσμού με δημόσιες υπηρεσίες συσχετιζόμενα με γεωγραφικές πληροφορίες. Κάποιες πληροφορίες που περιλαμβάνουν αυτά τα αρχεία είναι οι στάσεις που κάνει το κάθε μέσο διακίνησης (γεωγραφικές συντεταγμένες, ονομασίες), οι ώρες άφιξης και αναχώρησης σε κάθε στάση, τα ναύλα για να ταξιδέψει κάποιος με αυτό το μέσο διακίνησης.

Ένα GTFS αρχείο ([23]),παράδειγμα http://www.gtfs-data-exchange.com/agency/city-of-seattle/latest.zip) αποτελείται από διάφορα δεδομένα όπως :

agency.txt : Περιλαμβάνει κάποιες πληροφορίες σχετικά με την εταιρεία των λεωφορείων
stops.txt: Περιλαμβάνει πληροφορίες σχετικά με περιοχές από όπου ένα μεταφορικό μέσο παίρνει ή αφήνει επιβάτες
routes.txt: Περιλαμβάνει πληροφορίες σχετικά με τις διαδρομές μιας υπηρεσίας. Ένα route είναι μια ομάδα από διαδρομές που εμφανίζονται στους χρήστες σαν μία υπηρεσία.
trips.txt: Περιλαμβάνει όλα τα trips και τα routes τους. Ένα trip είναι μία ακολουθία από 2 ή περισότερες στασεις που συμβαίνουν μία συγκεκριμένη χρονική στιγμή.
stop_times.txt: Περιλαμβάνει τις ώρες που ένα μεταφορικό μέσο φτάνει και αναχωρεί από μία περιοχή για κάθε trip.
calendar.txt : Εβδομαδιαίο προγραμμα σταθμών. Προσδιορίζεται πότε ξεκινά μία υπηρεσία, πότε τελειώνει και ποιες μέρες την εβδομάδα είναι διαθέσιμη
calendar_dates.txt: Αργίες/Εξαιρέσεις των υπηρεσιών που προσδιορίζονται στο calendar.txt.
fare_attributes.txt: Προσδιορίζονται τα ναύλα των υπηρεσιών

fare_rules.txt: Προσδιορίζονται κάποιες πληροφορίες για τα ναύλα
shapes.txt: Περιλαμβάνει τους κανόνες για να δημιουργηθούν οι γραμμές για να σχηματιστεί μια διαδρομή.

Τα αρχεία GTFS είναι συνήθως διαθέσιμα δωρεάν (open source) στο διαδίκτυο ([24]) και πολλοί προγραμματιστές τα χρησιμοποιούν για την ανάπτυξη διαφόρων εφαρμογών.

Να σημειώσουμε ότι θα χρειαστούμε αυτού του είδους δεδομένα για την δρομολόγηση των διαδρομών μέσω του Open Trip Planner.

Geomatic Technologies-GTFS Data for Cyprus

Η Geomatic Technologies Ltd ([25]) είναι μια εταιρεία εγκατεστημένη στην Κύπρο που προσφέρει προϊόντα και υπηρεσίες για ολοκληρωμένες λύσεις τηλεματικής. Είναι μια από τις μεγαλύτερες εταιρείες που ασχολούνται με χαρτογραφία στην Κύπρο. Για την ικανοποίηση των αναγκών της Κυπριακής αγοράς, η Geomatic υλοποιεί καινοτόμα εργαλεία διαχείρισης και παρέχει λύσεις τεχνολογικής ακμής που συνδυάζουν το Global Positioning System (GPS) και το Γεωγραφικό Σύστημα Πληροφοριών (GIS). Η εταιρεία παρέχει GIS και εξειδικευμένες υπηρεσίες σε ένα ευρύ φάσμα επαγγελματικών και επιστημονικών περιοχών, μέσω ενός δικτύου έμπειρων συνεργατών, συμβούλων και ειδικών επιστημόνων. Επιπλέον, το τμήμα Έρευνας και Ανάπτυξης της εταιρείας επεξεργάζεται και αναπτύσσει εφαρμογές και εργαλεία για τον δημόσιο τομέα.

Τον Φεβρουάριο του 2012 ήρθαμε σε επαφή με την συγκεκριμένη εταιρεία για να ζητήσουμε την συνεργασία και την βοήθειά τους σχετικά με τη απόκτηση κάποιων δεδομένων. Λόγω του ότι η εταιρεία αυτή είναι η πιο γνωστή στην Κύπρο, κρίναμε σωστό να απευθυνθούμε σε αυτή για την απόκτηση δεδομένων σχετικά με τις γραμμές λεωφορείων όπως είναι οι γεωγραφικές συντεταγμένες των στάσεων, τα ωρολόγια προγράμματα και τα ναύλα διακίνησης, δεδομένα τα οποία οι διάφορες εταιρείες λεωφορείων δεν είχαν στην διάθεσή τους. Με την απόκτηση αυτών των δεδομένων θα μπορούσαμε να κτίσουμε τα δικά μας GTFS αρχεία για το σύστημα δημόσιας διακίνησης στην Κύπρο, πράγμα που δεν υπήρχε πουθενά μέχρι τώρα.

Στην επικοινωνία που είχαμε με την εταιρεία μέσω τηλεφώνου, κανονίσαμε συνάντηση σε δικό τους χώρο, εκεί που είναι εγκατεστημένα τα γραφεία τους. Στο χώρο εκείνο, αφού τους

εξήγησα τον ακριβή σκοπό της διπλωματικής εργασίας και την ανάγκη δημιουργίας των GTFS αρχείων για πειραματισμό και έλεγχο της εφαρμογής, μου απάντησαν ότι είχαν ήδη δημιουργήσει αυτά τα GTFS αρχεία (μόλις 2 μήνες πριν να επικοινωνήσω μαζί τους) και ότι και η ίδια η εταιρεία είχε βλέψεις να φτιάξει παρόμοια εφαρμογή. Λόγω του ότι όμως αυτά τα δεδομένα ήταν πολύ σημαντικά για την εταιρεία, διότι προέκυψαν μετά από χρόνια εντατικής δουλειάς (μου ανέφεραν ότι οι άνθρωποι της Geomatic ήταν από την αρχή παρόντες στην απόφαση για την τοποθεσία των στάσεων των λεωφορείων) και έτσι δεν ήταν σε θέση να μου παρέχουν τέτοιου είδους δεδομένα. Για το λόγο αυτό, το μόνο που μπορούσαν να κάνουν ήταν να μου παρέχουν ένα μικρό δείγμα αυτών των δεδομένων (2-3 δρομολόγια λεωφορείων) με την προϋπόθεση ότι δεν θα δημοσιοποιούνταν. Ευχαρίστως αποδέχτηκα τους όρους τους και τους ευχαρίστησα θερμά για την συνεργασία τους. Σε επόμενο στάδιο δημιούργησα τον γράφο που αφορούσε το δίκτυο της Κύπρου στο server του πανεπιστημίου μέσω του Open Trip Planner.

2.6 Client-Server Based VS Client Based-Only

Η μελέτη αυτή έγινε για να εξακριβωθούν και να απαντηθούν κάποια βασικά ερωτήματα σχετικά με το πώς θα σχεδιαστεί και θα υλοποιηθεί το θέμα της διπλωματικής εργασίας. Συγκεκριμένα θα απαντηθούν κάποια ερωτήματα σχετικά με το αν η υλοποίηση θα είναι client-server based ή εάν θα είναι client-based only. Να σημειωθεί ότι ένα υβριδικό σύστημα (κάποια να τρέχουν στον server και κάποια local στη συσκευή) δεν είναι δυνατόν να υλοποιηθεί διότι το πλείστο υλικό θα χρησιμοποιηθεί από το Open Trip Planner. Συγκεκριμένα θα αναφερθούμε στα χαρακτηριστικά του client-Server based και στα πλεονεκτήματα-μειονεκτήματα του. Στη συνέχεια θα αναφερθούμε στα χαρακτηριστικά του client-based-only και στα πλεονεκτήματα-μειονεκτήματα που έχει. Τέλος, θα κάνουμε μια μικρή σύγκριση και θα αποφασίσουμε τελικά τι συμφέρει να ακολουθήσουμε.

2.8.1 Client-Server Based

Προτού προχωρήσουμε, οφείλουμε να εξηγήσουμε τον όρο client-server architecture. Είναι λοιπόν μία δικτυακή αρχιτεκτονική στην οποία κάθε μηχανή ή διεργασία είναι είτε client είτε server. Οι servers είναι ισχυρές μηχανές αφοσιωμένες σε διεργασίες οι οποίες είναι υπεύθυνες για την διαχείριση διαφόρων πόρων όπως disk drives (disk servers), printers (print servers) ή network traffic(network servers). Οι clients είναι συνήθως προσωπικοί υπολογιστές(PCs) ή αλλιώς σταθμοί εργασίας όπου οι χρήστες τρέχουν κάποιες εφαρμογές. Ο client ξεκινάει πάντα μια σύνδεση με τον server, ενώ ο server περιμένει πάντα για

αιτήματα από κάθε client. Οι clients βασίζονται στους servers για πόρους όπως αρχεία, συσκευές και κυρίως για υπολογιστική δύναμη. Η client-server αρχιτεκτονική έχει γίνει ένα από τα βασικά μοντέλα των υπολογιστών του διαδικτύου. Πολλά είδη εφαρμογών έχουν σχεδιαστεί χρησιμοποιώντας το client-server μοντέλο. Τυπικές λειτουργίες δικτύου, όπως η ανταλλαγή e-mail, πρόσβαση στο διαδίκτυο και πρόσβαση σε βάσεις δεδομένων έχουν σχεδιαστεί με βάση το μοντέλο client-server. Για παράδειγμα, ένα πρόγραμμα περιήγησης στο Web είναι ένα πρόγραμμα-πελάτη στον υπολογιστή των χρηστών που μπορούν να έχουν πρόσβαση σε πληροφορίες σε οποιονδήποτε web server στον κόσμο.

Πλεονεκτήματα Client-Server αρχιτεκτονικής

1. Στις περισσότερες περιπτώσεις, μία client-server αρχιτεκτονική επιτρέπει στους ρόλους και τις ευθύνες ενός πληροφοριακού συστήματος να κατανεμηθεί μεταξύ πολλών ανεξάρτητων υπολογιστών που είναι γνωστοί ο ένας με τον άλλο μόνο μέσω ενός δικτύου. Πλεονέκτημα λοιπόν αυτού του μοντέλου είναι μεγαλύτερη ευκολία συντήρησης. Για παράδειγμα, είναι δυνατή η αντικατάσταση, επισκευή, αναβάθμιση, ή ακόμα και επανεγκατάσταση δεδομένων σε ένα server, ενώ οι clients του βρίσκονται σε άγνοια και δεν επηρεάζονται από την αλλαγή αυτή (encapsulation).
2. Το Modular Infrastructure έχει ως σκοπό να βελτιώσει την ευχρηστία, την ευελιξία, την λειτουργικότητα και την επεκτασιμότητα σε σχέση με άλλες αρχιτεκτονικές όπως είναι η Centralized, η Mainframe και η time sharing computing.
3. Όλα τα δεδομένα είναι αποθηκευμένα στους servers, στους οποίους υπάρχει καλός έλεγχος ασφάλειας. Οι servers μπορούν να ελέγχουν καλύτερα την πρόσβαση και πόρους, ώστε να εξασφαλίζεται ότι μόνο οι clients με τα κατάλληλα δικαιώματα μπορούν να έχουν πρόσβαση και να τροποποιήσουν τα δεδομένα.
4. Έχουν ειδικά πρωτόκολλα για την μείωση του traffic στο δίκτυο.
5. Υπάρχουν πολλές σύγχρονες client-server τεχνολογίες οι οποίες είναι και διαθέσιμες και έχουν ως σκοπό να διασφαλίσουν την ασφάλεια, να παρέχουν φιλική προς τον χρήστη διεπαφή και εύκολη στη χρήση.
6. Είναι ευέλικτη διότι μπορεί να λειτουργήσει με πολλαπλούς διαφορετικούς πελάτες διαφορετικών προδιαγραφών.

Μειονεκτήματα Client-Server αρχιτεκτονικής

1. Ένα από τα πιο συχνά προβλήματα που προκαλείται σε client-server αρχιτεκτονικές είναι το πρόβλημα του network traffic που προκαλεί κατάρρευση του server. Αυτό συμβαίνει διότι ο server υπερφορτώνεται λόγω της ταυτόχρονης αποστολής αιτημάτων από πολλούς clients.

2. Εάν ένας server τεθεί εκτός λειτουργίας (βλάβη ή διακοπή τροφοδοσίας) τότε τα αιτήματα των clients δεν μπορούν να διεκπεραιωθούν.
3. Στις περισσότερες Client-Server αρχιτεκτονικές, για να επικοινωνήσει ο client με το server απαιτείται η σύνδεση με το διαδίκτυο.

2.8.2 Client Based Only

Με τον όρο client-based-only εννοούμε την δυνατότητα μιας μηχανής να τρέχει διάφορες διεργασίες, χρησιμοποιώντας τους δικούς της πόρους, χωρίς οποιαδήποτε επικοινωνία με οποιαδήποτε μορφή δικτύου. Δεν έχουμε την έννοια του server καθώς για την διεκπεραίωση κάποιας διεργασίας μιας εφαρμογής, γίνεται χρήση της υπολογιστικής ισχύς της μηχανής στην οποία τρέχει η συγκεκριμένη εφαρμογή. Εφαρμογές που ακολουθούν τέτοιου είδους αρχιτεκτονική τρέχουν locally διάφορες εντολές και locally παίρνουν τα διάφορα αποτελέσματα.

Πλεονεκτήματα Client Based Only αρχιτεκτονικής

1. Περισσότερη ασφάλεια που παρέχεται λόγω του ότι δεν υπάρχει άμεση επικοινωνία με τον έξω κόσμο (εξωτερικές συνδέσεις).
2. Καλύτερος έλεγχος και διαμοιρασμός των διεργασιών που εκτελούνται, από τον ίδιο τον χρήστη.
3. Δεν επηρεάζεται από οποιοδήποτε network traffic.
4. Βασίζεται μόνο στις δυνατότητες της μηχανής και όχι από εξωτερικούς παράγοντες (για παράδειγμα στην περίπτωση των servers εάν ο server χαλούσε τότε η δουλειά μας δεν θα μπορεί να διεκπεραιωθεί).

Μειονεκτήματα Client Based Only αρχιτεκτονικής

1. Οι πόροι που προσφέρει η μηχανή μπορεί να μην επαρκούν για τις απαιτήσεις του συστήματος.
2. Για συστήματα που απαιτούν μεγάλη υπολογιστική ισχύ καταναλώνεται αρκετή ενέργεια και σε κινητές συσκευές αυτό είναι αρκετά περιοριστικό.

2.8.3 Συμπεράσματα

Εφόσον το θέμα που θα υλοποιήσουμε απαιτεί την χρήση κινητής συσκευής συγκεκριμένα συσκευής που τρέχει λογισμικό Android θα πρέπει να δούμε τι υπολογιστική ισχύ υποστηρίζει μια μέση συσκευή καθώς και ποια είναι τα υπόλοιπα χαρακτηριστικά της. Συγκεκριμένα θα χρησιμοποιήσουμε την κινητή συσκευή της Google με μοντέλο Nexus-S.

Talk Time:	6 hours
Stand-By time	428 hours
Ram	512MB
Internal Storage	16Gb
Wi-Fi	802.11 b/g/n
GPS	Yes
OS Version	Android 2.3
CPU	1GHz Cortex A8 (Hummingbird) processor

Πίνακας 2.12. Χαρακτηριστικά Nexus-S

(Σημείωση: Θα μπορούσαμε να χρησιμοποιήσουμε και συσκευές που έχουν μεγαλύτερες υπολογιστικές και χωρικές δυνατότητες αλλά για το θέμα που θα ασχοληθούμε δεν θα υπάρξει μεγάλη διαφορά. Εξάλλου η εφαρμογή θέλουμε να δουλεύει και στις πιο απλές κινητές συσκευές.)

Όπως έχουμε αναφέρει, θα ασχοληθούμε με κινητές συσκευές τύπου Android. Οι πλείστες από αυτές τις συσκευές δεν έχουν και μεγάλη υπολογιστική ισχύ γιατί εξάλλου οι διεργασίες που εκτελούν δεν απαιτούν μεγάλη υπολογιστική ισχύ. Εφαρμογές οι οποίες απαιτούν περισσότερους πόρους είναι συνήθως υλοποιημένες σε client server αρχιτεκτονική όπως για παράδειγμα τα Google maps . Γίνεται από τον client ένα request το οποίο εμφανίζεται στον server υπό την μορφή query . Ο server τότε εκτελεί αυτό το query και στέλλει τα αποτελέσματα στον client. Γενικά, η αρχιτεκτονική client-server είναι αρκετά διαδεδομένη σε κινητές συσκευές και χρησιμοποιείται σε πολλές εφαρμογές λόγω των πλεονεκτημάτων που αναφέραμε προηγουμένως.

Ας καταγράψουμε περιληπτικά τις διάφορες διαδικασίες που πρέπει να συμπεριληφθούν για να πετύχουμε το σκοπό μας:

1. Άνοιγμα εφαρμογής
2. Λήψη συντεταγμένων τοποθεσίας συσκευής
3. Συγκέντρωση πληροφορίας σχετικά με τις συντεταγμένες: Θα πρέπει να κατασκευαστεί ανάλογος γράφος που θα περιέχει τα δεδομένα από πολλαπλά αρχεία GTFS.

4. Εμφάνιση χάρτη βάση συντεταγμένων .
5. Επιλογή σημείου εκκίνησης και σημείου προορισμού από τον χρήστη
6. Επεξεργασία ζητήματος χρήστη.
7. Αλγόριθμος εύρεσης διαδρομής
8. Επεξεργασία αποτελέσματος αλγορίθμου
9. Εμφάνιση αποτελέσματος

Όπως παρατηρούμε από τα πιο πάνω σημεία, υπάρχουν πολλές διαδικασίες που πρέπει να εκτελεστούν και σίγουρα δεν θα είναι σε τόσο απλή μορφή όπως τα παρουσιάζουμε αλλά σε πιο πολύπλοκη μορφή.

Συγκεκριμένα, το σημείο 3 είναι αυτό που θα χρειαστεί περισσότερους πόρους από την συσκευή. Ένα τυπικό GTFS αρχείο για μια περιοχή από συγκεκριμένο agent έχει μέγεθος 20 MB περίπου. Χώρες όπως η Αμερική έχουν αρκετούς φακέλους GTFS από διαφορετικούς agents. Αυτό όμως προϋποθέτει την ανάγκη για μεγάλο αποθηκευτικό χώρο. Επιπλέον, για την κατασκευή του γράφου θα χρειαστεί κάποια σημαντική υπολογιστική ισχύ γιατί θα πρέπει να γίνει κάποια επεξεργασία των GTFS αρχείων προτού κατασκευαστεί.

Στο σημείο 4 απαιτείται η χρήση χάρτη. Οι χάρτες συνήθως γίνονται access online διότι είναι μεγάλου μεγέθους και δεν βολεύει να είναι αποθηκευμένοι locally. Για παράδειγμα, οι open street maps έχουν μέγεθος 250GB που είναι ένα τεράστιο μέγεθος. Υπάρχει φυσικά και η δυνατότητα διαχωρισμού του χάρτη ανά χώρα για να πάρουμε την χώρα που μας βολεύει αλλά πάλι το μέγεθος θα είναι αρκετά μεγάλο.

Στο σημείο 6 και 7 αφού ο χρήστης τοποθετήσει τα σημεία εκκίνησης και προορισμού γίνεται κάποια επεξεργασία τους (μετατρέπονται σε συντεταγμένες) και θα πρέπει να γίνει κάποια επεξεργασία στο γράφο που δημιουργήθηκε ούτως ώστε να βρεθεί με βάση κάποιο αλγόριθμο η συντομότερη διαδρομή. Αυτή η διαδικασία κοστίζει σε υπολογιστική δύναμη.

Σύνοψη:

Από τις πιο πάνω παρατηρήσεις καταλήγουμε στις πιο κάτω απαιτήσεις για την επίτευξη του στόχου μας:

1. Απαιτείται μεγάλος αποθηκευτικός χώρος για αποθήκευση χάρτη, GTFS αρχείων.
2. Απαιτείται κάποια σχετικά μεγάλη υπολογιστική ισχύ για σημαντικές διεργασίες.

Για την Client-based-only αρχιτεκτονική, έχοντας τα χαρακτηριστικά της κινητής συσκευής που είδαμε πιο πάνω και τις απαιτήσεις που αναφέραμε, παρατηρούμε ότι καμιά από τις απαιτήσεις που θέσαμε δεν μπορεί να ικανοποιηθεί. Εάν τρέχουμε locally τις διάφορες

διεργασίες μας τότε απαιτείται μεγάλη υπολογιστική ισχύ και τεράστιος αποθηκευτικός χώρος για την κάλυψη αυτών των διεργασιών πράγμα που είναι αδύνατο με τις παρούσες κινητές συσκευές. Ακόμα και εάν ήταν δυνατόν, τότε η κατανάλωση ενέργειας της κινητής συσκευής θα ήταν τεράστια με αποτέλεσμα να καταναλώνεται αρκετά γρήγορα η μπαταρία. Συνεπώς θα έπαυε να ήταν λειτουργική και εύχρηστη πράγμα που είναι το σημαντικότερο χαρακτηριστικό μιας κινητής συσκευής.

Εάν συγκρίνουμε με μια ματιά τα πλεονεκτήματα των 2 αρχιτεκτονικών που αναφέραμε προηγουμένως, παρατηρούμε ότι υπερέχει η client-server αρχιτεκτονική. Με την αρχιτεκτονική αυτή ο αποθηκευτικός χώρος που είναι αναγκαίος για τις διάφορες διεργασίες δεν αποτελεί πρόβλημα αφού τα απαραίτητα αρχεία θα βρίσκονται αποθηκευμένα σε ένα server με τεράστιο αποθηκευτικό χώρο. Επιπλέον, η υπολογιστική δύναμη του server είναι τεράστια αφού είναι καθαρά dual-core ή quad-core ενώ η μνήμη σε ram είναι τουλάχιστον 2 GB. Παρόλα τα πλεονεκτήματα του μοντέλου client-server εντούτοις υπάρχουν κάποια σημαντικά μειονεκτήματα που δεν μπορούμε να αγνοήσουμε. Πρώτο και σημαντικότερο, είναι το γεγονός ότι για να υπάρχει επικοινωνία μεταξύ της κινητής συσκευής και του Server θα πρέπει να υπάρχει συνδεσιμότητα με το διαδίκτυο. Αυτό για μια κινητή συσκευή σημαίνει ότι είτε θα πρέπει να υπάρχει σύνδεση με κάποιο τοπικό wi-fi access point είτε να υπάρχει σύνδεση internet με κάποιο παροχέα κινητής τηλεφωνίας μέσω της sim card. Σε χώρες όπως η Αμερική αυτό είναι πλέον ένα συνηθισμένο φαινόμενο που δεν κοστίζει σχεδόν τίποτα άρα δεν αποτελεί πρόβλημα. Στην Κύπρο και σε άλλες χώρες, η μέθοδο με την Sim Card δεν είναι ευρέως διαδεδομένη (λόγω υψηλών τιμών στο mobile internet) και μπορεί να θεωρηθεί ως πρόβλημα που θα ξεπεραστεί με την πάροδο του χρόνου . Επιπλέον, σε πολλές χώρες συμπεριλαμβανομένου και της Κύπρου, τα δημόσια μέσα μεταφοράς όπως το λεωφορείο παρέχουν την δυνατότητα σύνδεσης με το Internet μέσω wi-fi spots που βρίσκονται εγκαταστημένα στα λεωφορεία (παράδειγμα: γραμμή Πέρα Χωρίο-Λευκωσία). Εκμεταλλευόμενη αυτή την δυνατότητα, οι χρήστες της εφαρμογής μας θα μπορούν εύκολα να αλληλεπιδράσουν με το σύστημα μας χωρίς να έχουν την λειτουργία του mobile internet. Η κατανάλωση ενέργειας (λόγω σύνδεσης με το διαδίκτυο) δεν θα είναι μεγάλη διότι η διεργασίες που εκτελούνται στην κινητή συσκευή αφορούν μόνο τη σύνδεση με το server μέσω της αποστολής πακέτων. Ουσιαστικά όλες οι διεργασίες θα εκτελούνται στο server ο οποίος θα λαμβάνει requests από την κινητή συσκευή και θα απαντά. Πέραν από αυτό το μειονέκτημα, τα υπόλοιπα μειονεκτήματα που έχουμε αναφέρει παραμένουν.

Παρόλα τα μειονεκτήματα του client-server και σύμφωνα με όλα αυτά που έχουμε αναφέρει καταλήγουμε στο συμπέρασμα ότι **θα ακολουθήσουμε client-server αρχιτεκτονική** γιατί η άλλη μέθοδος δεν μπορεί να ανταποκριθεί στις απαιτήσεις του συστήματός μας.

Κεφάλαιο 3. Αλγόριθμος Δρομολόγησης

3.1 Διατύπωση του προβλήματος 39
3.2 A* αλγόριθμος 41
3.3 Χρονική Πολυπλοκότητα 48

3.1. Διατύπωση του προβλήματος

Ως δεδομένα εισόδου έχουμε ένα κατευθυνόμενο γράφο, και είναι κατευθυνόμενος διότι εάν πάρουμε για παράδειγμα τα λεωφορεία μπορεί να μην έχουν διαδρομές που πηγαίνουν από τη μία στάση στην άλλη και πίσω, άμεσα, δλδ φανταστείτε 2 στάσεις κάποιας γραμμής λεωφορείου, A,B. Από την A στάση μπορείς να πας ευθύδρομα στην B, αλλά ανάδρομα προς τα πίσω δεν μπορείς να πας.

Στο πρόβλημά υποθέτουμε ότι υπάρχουν κυκλικές διαδρομές δλδ ξεκινώντας από μία στάση, για να πάμε σε μία στάση που βρίσκεται πιο πίσω , μπορεί να χρειαστεί να πάρω μία διαδρομή που κάνει κύκλο, εάν δεν υπάρχει πιο σύντομη. Κόμβο εκκίνησης τον συμβολίζουμε με s, και τον κόμβο προορισμού, δλδ εκεί που θέλει να πάει ο χρήστης το συμβολίζουμε με d. Έτσι, με τους όρους που θέσαμε, σίγουρα θα υπάρχει λύση από τον αρχικό κόμβο στον κόμβο προορισμού.

Επίσης ο χρήστης χρειάζεται να δώσει κάποια άλλα στοιχεία, κάποια χρονικά όρια, που αφορούν το χρόνο που θα ξεκινήσει το ταξίδι.

Inputs:

- G(V,E) : κατευθυνόμενος γράφος όπου V οι κόμβοι και E οι ακμές
- s κόμβος εκκίνησης και d κόμβος προορισμού, $s,d \in V$ και $s \neq d$
- sTime: ο χρόνος που θέτει ο χρήστης για το ταξίδι.

Θα χρειαστούμε την ακόλουθη σημειογραφία για να αναλύσουμε το πρόβλημα

- $V=\{v_0,v_1,v_2,\ldots,v_n\}$ το σύνολο των κόμβων που αντιπροσωπεύουν τις στάσεις
- $E_{ij}=(v_i,v_j)$ δυάδα κόμβων με ακμή από τον κόμβο v_i στο v_j, $1\leq i \leq n$, $1\leq j \leq n$, $i \neq j$
- time(Eij) ο χρόνος μετάβασης από το v_i στο v_j, $i \neq j$
- switch(Eij) η εναλλαγή μέσου μεταφοράς από τον κόμβο v_i στο v_j , $i \neq j$.
  Μπορεί να πάρει τιμές 0 ή 1:
  - 0 δεν υπάρχει εναλλαγή
  - 1 υπάρχει εναλλαγή

- Μονοπάτι: είναι ένα σύνολο (ακολουθία) από κόμβους που ενώνονται μεταξύ τους με ακμές.
- P_{sd}^{i} ={E_{sx},E_{xr},…, E_{jy}, E_{yd}} είναι ένα μονοπάτι που αποτελείται από ένα σύνολο από δυάδες κόμβων με αρχικό κόμβο τον s και τελικό τον d.

- P_{sd} = { P_{sd}^{1} , P_{sd}^{2}, …P_{sd}^{t} } σύνολο από μονοπάτια

- α,β ≥ 0 κάποιες σταθερές τιμές (για κανονικοποίηση)

Υπολογισμός κόστους:

MinCost= min $\sum_{i=1}^{|P|}$ ([α * time(P_{sd}^{j} (i)) + β * switch(P_{sd}^{j} (i))])

$\forall P_{sd}^{j} \in P_{sd,}$ 1≤j≤t

Το μονοπάτι είναι μία ακολουθία από κόμβους που ενώνονται μεταξύ τους με ακμές

Το P_{sd}^{i} , συμβολίζει ένα μονοπάτι από τον αρχικό κόμβο στον τελικό

Οι σταθερές τιμές α,β αφορούν κάποιες σταθερές τιμές. Τις χρησιμοποιούμε για να κανονικοποιήσουμε τις τιμές των κόστων.

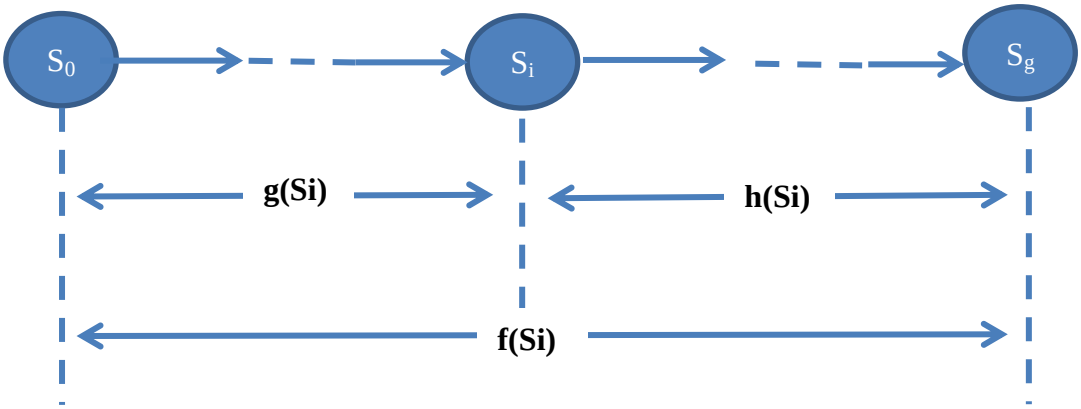
Ο υπολογισμός του ελάχιστου κόστους υπολογίζεται ως εξής:

Παίρνουμε μία μία τις διαδρομές και για κάθε διαδρομή βρίσκουμε το άθροισμα των κόστων για κάθε περίπτωση κόστους, δλδ για το κόστος στο χρόνο και στον αριθμό εναλλαγών. Να σημειώσουμε ότι κάθε φορά πολλαπλασιάζουμε το άθροισμα κόστους με κάποιο σταθερό αριθμό για να κανονικοποιήσουμε τις τιμές.

3.2. A* Αλγόριθμος

(Σημείωση: κάποιες πληροφορίες αντλήθηκαν από τις σημειώσεις της Καθηγήτριας Ελπίδας Κεραυνού από το μαθήματος ΕΠΛ 341. Τεχνητή Νοημοσύνη από το Κεφ 3. «Επίλυση προβλημάτων μέσω αναζήτησης.»)

Ο A* είναι αλγόριθμος δρομολόγησης που προκύπτει από τον αλγόριθμο του Dijkstra. Είναι αρκετά δημοφιλής σε προβλήματα δρομολόγησης διότι μπορεί να βελτιώσει τον χρόνο αναζήτησης.



Διάγραμμα 3.1. A* υπολογισμός κόστους

Ο αλγόριθμος A* χρησιμοποιεί την συνάρτηση αξιολόγησης **f(Si)=g(Si)+h(Si)** όπου:

g(Si): αφορά το πραγματικό κόστος του συγκεκριμένου κόμβου Si (που ελέγχεται την δεδομένη στιγμή) σε σχέση με την αρχική κατάσταση S_0. Είναι δηλαδή η απόσταση από την ρίζα (κόμβος εκκίνησης) στην Si.

h(Si): πρόκειται για το κόστος που υπολογίζει (μαντεύει) η ευρετική συνάρτηση. Προσπαθεί να μαντέψει το κόστος για να πάμε από την κατάσταση Si στην τελική κατάσταση S_g.

Η αξιοπιστία της συνάρτησης f εξαρτάται από την αξιοπιστία της συνάρτησης h. Εάν το κόστος που υπολογίζει η h υπερεκτιμηθεί τότε υπάρχει ο κίνδυνος παραγνώρισης της Si. Εάν για οποιαδήποτε κατάσταση η ευρετική συνάρτηση δεν υπερεκτιμά το πραγματικό κόστος της μετάβασης τότε η χρήση της μας εγγυάται λύση και η ευρετική συνάρτηση λέμε ότι είναι παραδεκτή.

Ορισμοί:

ανοικτή κατάσταση: είναι μη-τελική κατάσταση η οποία δεν έχει ακόμα διερευνηθεί ή για την οποία ενδείκνυται η εκ νέου διερεύνηση.

κλειστή κατάσταση: κατάσταση η οποία επί του παρόντος θεωρείται ότι έχει διερευνηθεί.

προκάτοχος της κατάστασης s: η κατάσταση s΄, η οποία οδηγεί απευθείας, δλδ μέσω μιας μοναδικής ενέργειας, στην κατάσταση s και επί του παρόντος η διαδρομή από την αρχική κατάσταση s διαμέσου της s΄ θεωρείται η καλύτερη.

διάδοχος της κατάστασης s: οι καταστάσεις για τις οποίες μια κατάσταση s΄ αποτελεί προκάτοχό τους

αδιέξοδο: μη-τελική κατάσταση η οποία δεν έχει διαδόχους ή όλες οι διάδοχοί της οδηγούν σε αδιέξοδο

Ψευδοκώδικας Α* αλγορίθμου:

1. ΑΝΟΙΚΤΕΣ := [so], ΚΛΕΙΣΤΕΣ := [ ]

2. Εάν ΑΝΟΙΚΤΕΣ = [ ] τότε τερμάτισε. Δεν υπάρχει λύση.

3. Αφαίρεσε την κατάσταση, si, από την λίστα ΑΝΟΙΚΤΕΣ, για την οποία f(si) <= f(sj) για όλες τις άλλες ανοικτές καταστάσεις sj και πρόσθεσε την στις ΚΛΕΙΣΤΕΣ.

4. Δημιούργησε τις διαδόχους της si, και δώσε στην κάθε διάδοχο ένα δείκτη προς την si, ως την προκάτοχό της.

5. Εάν η sg ανήκει στις διαδόχους της si, τότε τερμάτισε και επέστρεψε τη διαδρομή από την sg στην so.

6. Διαφορετικά επανέλαβε τα ακόλουθα για κάθε διάδοχο, sj, της si:

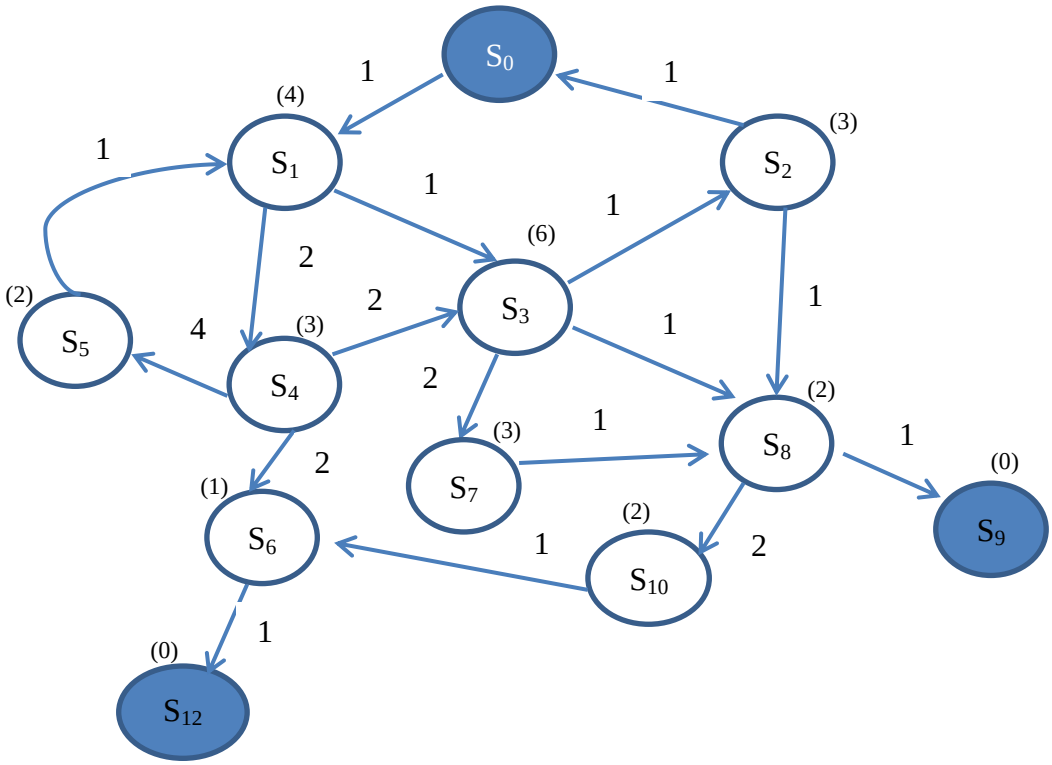
6.1 Υπολόγισε την τιμή f(sj).

6.2 Εάν η sj δεν ανήκει ούτε στις ΑΝΟΙΚΤΕΣ, ούτε στις ΚΛΕΙΣΤΕΣ καταστάσεις, τότε πρόσθεσε την sj στις ΑΝΟΙΚΤΕΣ με τιμή αξιολόγησης, f(sj).

6.3 Εάν η sj ήδη ανήκει στις ΑΝΟΙΚΤΕΣ ή ΚΛΕΙΣΤΕΣ τότε σύγκρινε τη νέα τιμή αξιολόγησής της, έστω νέα, με την παλαιότερή της, έστω παλαιά. Εάν παλαιά <= νέα, τότε απέβαλε το νέο κόμβο με την κατάσταση sj. Διαφορετικά, αφαίρεσε το στοιχείο (sj, παλαιά) από την λίστα στην οποία ανήκει (ΑΝΟΙΚΤΕΣ ή ΚΛΕΙΣΤΕΣ) και πρόσθεσε το στοιχείο (sj, νέα) στις ΑΝΟΙΚΤΕΣ.

7. Επανέλαβε από την εντολή 2.

Παράδειγμα :



Διάγραμμα 3.2. Παράδειγμα Α* (Εξέταση Ημιεξαμήνου 2009-2010 Τεχνητή Νοημοσύνη-Ελπίδα Κεραυνού)

Στο πιο πάνω διάγραμμα η αρχική κατάσταση είναι η S_0 και οι ικανοποιητικές τελικές καταστάσεις η S_{12} και S_9 . Ο κάθε ένας από τους παραπάνω κόμβους συνοδεύεται από την τιμή δεδομένης ευρετικής συνάρτησης (ο αριθμός που δίνεται σε παρένθεση) και ο αριθμός πάνω στις ακμές αφορά το βάρος της ακμής δλδ το κόστος να πάμε από τον ένα κόμβο στον άλλο.

Εφαρμογή αλγορίθμου Α*

	Ανοικτές καταστάσεις (Κατάσταση,f(Si),Προκάτοχος)	**f(Si)=g(Si)+h(Si)**	**Κλειστές καταστάσεις (Κατάσταση,f(Si),Προκάτοχος)**
1.	[(S_0,_,_)]	**f(S_1)=1+4=5**	[ ]
2.	[**(S_1,5, S_0)**]	f(S_3)=2+6=8 f(S_4)=3+3=6	[(S_0,_,_)]
3.	[**(S_4,6, S_1)** , (S_3,8, S_1)]	f(S_5)=7+2=9 f(S_6)=5+1=6	[(S_0,_,_), **(S_1,5, S_0)**]
4.	[**(S_6,6, S_4)**, (S_5,9, S_4) , (S_3,8, S_1)]	f(S_{12})=6+0=6	[(S_0,_,_), (S_1,5, S_0)**,(S_4,6, S_1)**]
5.	[(S_{12},6, S6),(S_5,9, S_4) , (S_3,8, S_1)]		[(S_0,_,_), (S_1,5, S_0)**,** (S_4,6, S_1), **(S_6,6, S_4)**]
	//		

Καταλήγω σε λύση στον κόμβο S_{12} .

Εφαρμογή αλγορίθμου στο OTP:

Ο A* αλγόριθμος που υλοποιείται στο OTP εκτελεί ακριβώς την ίδια διαδικασία με την διαδικασία που περιγράψαμε πιο πάνω. Αυτό όμως που αξίζει να αναφέρουμε είναι η ανάθεση των βαρών πάνω στις ακμές. Όπως έχουμε αναφέρει και πριν, η κατάσταση (κόμβος) η οποία θα εισαχθεί στην λίστα με την μικρότερη διαδρομή εξαρτάται από την συνάρτηση αξιολόγησης **f(Si)=g(Si)+h(Si).** Στο κεφάλαιο 2.4 έχουμε περιγράψει λεπτομερώς την δομή του γράφου και όπως έχουμε αναφέρει υπάρχουν διαφορετικά ήδη ακμών. Υπάρχουν ακμές BoardEdge, AlightEdge, DwellEdge, HopEdge, StreetTransitEdge. Το βάρος σε κάθε μια από αυτές τις ακμές ανατίθεται διαφορετικά. Πιο κάτω θα δούμε πως ανατίθενται τα βάρη σε κάθε μια από το είδος ακμών που αναφέραμε στην περίπτωση του g(Si) που αντιπροσωπεύει το πραγματικό βάρος μιας ακμής. Να αναφέρουμε ότι το βάρος αυτό εξαρτάται από το είδος της ακμής και προκύπτει από διάφορους παράγοντες όπως χρόνος αναμονής στον σταθμό, χρόνος επιβίβασης, περιορισμοί στις στροφές κτλ.

BoardEdge, AlightEdge cost: αφορά το κόστος για να ανεβούμε σε ένα μεταφορικό μέσο και το κόστος για κατέβουμε από ένα μεταφορικό μέσο. Το κόστος υπολογίζεται με τον ίδιο τρόπο και για τις 2 ακμές.
Πιο κάτω βλέπουμε τον κώδικα στον οποίο γίνεται η ανάθεση του βάρους.

```
StateEditor s1 = s0.edit(this);
S1.incrementTimeInSeconds (wait);
s1.incrementWeight (wait*options.waitReluctance + options.boardCost);
```

Το waitReluctance (απροθυμία αναμονής) πρόκειται για μια σταθερή τιμή (συντελεστής) η οποία εκφράζει πόσο χειρότερο είναι να περιμένει κανείς ένα όχημα δημόσιας μεταφοράς σε σχέση με το να είναι μέσα στο όχημα.

```
    public double waitReluctance = 0.95;
```

Το boardCost εξαρτάται από το είδος του μεταφορικού μέσου που επιθυμεί να χρησιμοποιήσει ο χρήστης κατά τον σχεδιασμό της διαδρομής (user prferences).

```
protected int walkBoardCost = 60 * 5;
protected int bikeBoardCost = 60 * 10; //cyclists hate loading their bike a second
time

public int getBoardCost(TraverseMode mode) {
        if (mode == TraverseMode.BICYCLE)
            return bikeBoardCost;
        // I assume you can't bring your car in the bus
        return walkBoardCost;
    }
```

DwellEdge cost: Πρόκειται για το κόστος του να περιμένει κανείς σε ένα μέσο μεταφοράς όταν το μέσο μεταφοράς κάνει μια στάση.

```
//seconds
int elapsed = stopTime.getDepartureTime() - stopTime.getArrivalTime();
  .
  .

  state1.incrementWeight(elapsed);
```

HopEdge cost: Πρόκειται για το κόστος μεταξύ δύο στάσεων. Αφορά τον χρόνο που χρειάζεται για να πάει το συγκεκριμένο μεταφορικό μέσο από μια στάση στην άλλη. Εκφράζεται σε δευτερόλεπτα.

```
int elapsed = end.getArrivalTime() - start.getDepartureTime();
s1.incrementWeight(elapsed);
```

StreetTransitEdge cost: Αφορά το κόστος(χρόνος) για να μεταβούμε από κόμβο σταθμό/στάση σε ένα κόμβο δρόμου(και το ανάποδο).

```
s1.incrementWeight(transitStop.getStreetToStopTime());
```

Ευρετική Συνάρτηση :

Για να υπολογίσουμε το **f(Si)** χρειαζόμαστε και το εκτιμώμενο βάρος **h(Si)** που υπολογίζει η ευρετική συνάρτηση. Πιο κάτω παραθέτουμε την ευρετική συνάρτηση που χρησιμοποιεί το OTP.

```
public double computeForwardWeight(State s, Vertex target) {

        Vertex sv = s.getVertex();
        double euclidianDistance = sv.fastDistance(target);

         if (useTransit) {
            double speed = options.getSpeedUpperBound();
            if (s.isAlightedLocal()) {
                if (euclidianDistance + s.getWalkDistance() > options.getMaxWalkDistance())
                {
                    return -1;
                }
                return options.walkReluctance * euclidianDistance / speed;
            } else {
                int boardCost;
                if (s.isOnboard()) {
                    boardCost = 0;
                } else {
                    boardCost = options.getBoardCostLowerBound();
                }
                if (s.isEverBoarded()) {
                    boardCost += options.transferPenalty;
```

```
            }
            if (euclidianDistance < target.getDistanceToNearestTransitStop()) {
                if (euclidianDistance + s.getWalkDistance() >
                    options.getMaxWalkDistance()) {
                    return -1;
                }
                return options.walkReluctance * euclidianDistance / speed;
            } else {
                double mandatoryWalkDistance = target.getDistanceToNearestTransitStop()
                        + sv.getDistanceToNearestTransitStop();
                if (mandatoryWalkDistance + s.getWalkDistance() >
                    options.getMaxWalkDistance()) {
                    return -1;
                }
                double distance = (euclidianDistance - mandatoryWalkDistance) / maxSpeed
                        + mandatoryWalkDistance * options.walkReluctance / speed
                        + boardCost;
                return Math.min(distance, options.walkReluctance * euclidianDistance
                        / speed);
            }
        }
    }
    else {
        // This was missing from the original AStar, but it only makes sense
        return options.walkReluctance * euclidianDistance / maxSpeed;
    }
}
```

Σχολιασμός:

```
double euclidianDistance = sv.fastDistance(target);
```

Η πρώτη παρατήρηση που έχουμε να κάνουμε σχετικά με την πιο πάνω συνάρτηση είναι ότι υπολογίζει την ορθροδρομική ή αλλιώς μεγάλου-κύκλου απόσταση (orthodromic/grate-circle distance) μεταξύ του κόμβου που ελέγχουμε και του τελικού κόμβου. Η τιμή αυτή δίνει την μικρότερη απόσταση μεταξύ 2 σημείων στην σφαίρα. Λόγω του ότι ασχολούμαστε με σφαιρική επιφάνεια και όχι με επίπεδη επιφάνεια δεν θα ήταν ορθό να χρησιμοποιήσουμε την γνωστή ευκλείδεια απόσταση για τον υπολογισμό της απόστασης μεταξύ 2 γεωγραφικών σημείων. Και επειδή η σφαιρική γεωμετρία είναι διαφορετική από τη ευκλείδεια γεωμετρία, οι εξισώσεις για την απόσταση παίρνουν μια διαφορετική μορφή. Η απόσταση μεταξύ δύο σημείων στον ευκλείδειο χώρο είναι το μήκος μιας ευθείας γραμμής από το ένα σημείο στο άλλο. Στην σφαιρική επιφάνεια όμως, δεν υπάρχουν ευθείες γραμμές. Γι' αυτό οι ευθείες γραμμές αντικαθίστανται με γεωδαισιακές. Γεωδαισιακές στην σφαίρα είναι οι κύκλοι στην σφαίρα των οποίων τα κέντρα συμπίπτουν με το κέντρο της σφαίρας.

Συνάρτηση υπολογισμού ορθροδρομικής απόστασης

$$\Delta\hat{\sigma} = \arctan\left(\frac{\sqrt{\left(\cos\phi_f \sin\Delta\lambda\right)^2 + \left(\cos\phi_s \sin\phi_f - \sin\phi_s \cos\phi_f \cos\Delta\lambda\right)^2}}{\sin\phi_s \sin\phi_f + \cos\phi_s \cos\phi_f \cos\Delta\lambda}\right).$$

```
//implementation of orthodromic function
public double distance(double lat1, double lon1, double lat2, double lon2,double
radius) {

        lat1 = toRadians(lat1); // Theta-s
        lon1 = toRadians(lon1); // Lambda-s
        lat2 = toRadians(lat2); // Theta-f
        lon2 = toRadians(lon2); // Lambda-f

        double deltaLon = lon2 - lon1;

        double y = sqrt(DistanceLibrary.p2(cos(lat2) * sin(deltaLon))
                + DistanceLibrary.p2(cos(lat1) * sin(lat2) - sin(lat1) * cos(lat2)
* cos(deltaLon)));
        double x = sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(deltaLon);

        return radius * atan2(y, x);
    }
```

Όταν επιλέγουμε μια ευρετική συνάρτηση για τον A* αλγόριθμο, ένα σημαντικό κριτήριο είναι ότι δεν πρέπει να υπερεκτιμά το υπολειπόμενο βάρος μέχρι να φτάσουμε στον στόχο. Δηλαδή τα βήματα που χρειάζεται να κάνω για να πάω από μια κατάσταση στην τελική κατάσταση πρέπει να μην είναι περισσότερα από τα πραγματικά βήματα. Στις διάφορες περιπτώσεις που υπάρχουν στον κώδικα παρατηρούμε ότι η ορθοδρομική απόσταση διαιρείται με την μέγιστη δυνατή ταχύτητα που έχει ένα συγκεκριμένο μεταφορικό μέσο. Η ορθοδρομική απόσταση δεν είναι η πραγματική απόσταση που θα διανύσει κανείς ούτως ώστε να φτάσει στον προορισμό του διότι η πραγματική απόσταση περιλαμβάνει στροφές, λόφους κτλ. Επιπλέον, διαιρώντας την ορθοδρομική απόσταση με την μέγιστη δυνατή ταχύτητα που έχει ένα συγκεκριμένο μεταφορικό μέσο το αποτέλεσμα που προκύπτει είναι χρόνος (απόσταση/ταχύτητα= χρόνος). Αυτό μαζί με κάποιους άλλους παράγοντες (Παράδειγμα: walk reluctance) δίνουν και το βάρος της ακμής και καθιστούν την ευρετική συνάρτηση παραδεκτή (υποεκτιμά το υπολειπόμενο βάρος).

Γενικά όμως, όσο πιο πολύ απέχει η τιμή που δίνει η ευρετική συνάρτηση από την πραγματική τιμή τόσοι περισσότεροι κόμβοι πρέπει να διερευνηθούν για να βρεθεί βέλτιστη λύση και άρα τόση περισσότερη ώρα θα πάρει μέχρι να υπολογιστεί το μονοπάτι.

Ευρετική συνάρτηση-ανάθεση βαρών:

1. Εάν το ταξίδι περιλαμβάνει δημόσια μεταφορικά μέσα (user preferences)
    i. Εάν δεν είμαστε σε κάποιο μεταφορικό μέσο (όσο αφορά τον κόμβο).
    υπάρχουν 2 τρόποι να υπολογίσουμε το υπολειπόμενο βάρος και παίρνουμε όποιο από τα δύο είναι το πιο μικρό:
        α.κόστος=walkReluctance*απόστασηΜέχριΠροορισμό/ταχύτητα περπατήματος
        β. κόστος =κόστος επιβίβασης + κόστος περπατήματος που θα χρειαστεί μέχρι να φτάσουμε στον προορισμό /ταχύτητα περπατήματος+ (απόστασηΜέχριπροορισμό - κόστος περπατήματος που θα χρειαστεί μέχρι να φτάσουμε στον προορισμό)/μέγιστηΤαχύτητα
    ii. Βρισκόμαστε ήδη πάνω σε κάποιο μεταφορικό μέσο(δεν έχουμε το κόστος επιβίβασης)
        κόστος = walkReluctance*απόστασηΜέχριΠροορισμό/ταχύτητα
2. Εάν το ταξίδι δεν περιλαμβάνει δημόσια μεταφορικά μέσα
        κόστος = walkReluctance*απόστασηΜέχριΠροορισμό/ταχύτητα

3.3 Χρονική πολυπλοκότητα

Η χρονική πολυπλοκότητα του αλγορίθμου A* εξαρτάται από την ευρετική συνάρτηση που χρησιμοποιείται. Στη χειρότερη περίπτωση, ο αριθμός των κόμβων που διερευνούνται είναι εκθετικός σε σχέση με το μήκος της βέλτιστης λύσης (η συντομότερη διαδρομή), αλλά είναι πολυωνυμικός, όταν ο χώρος αναζήτησης είναι ένα δέντρο, υπάρχει μόνο ένας στόχος, και η ευρετική συνάρτηση h πληρεί τον ακόλουθο όρο:

$$|h(x) - h^*(x)| = O(\log h^*(x))$$

όπου h* είναι η βέλτιστη ευρετική συνάρτηση που παρέχει πάντα το ακριβής κόστος για να πάμε στο στόχο. Με άλλα λόγια, το σφάλμα του h δεν θα αυξηθεί γρηγορότερα από το λογάριθμο του "βέλτιστου ευρετικού" h* που επιστρέφει την πραγματική απόσταση από το x προς το στόχο.

Κεφάλαιο 4. Περιγραφή Συστήματος

4.1 Τεχνολογίες-Εργαλεία που χρησιμοποιήθηκαν 49
4.2 Συσκευές που χρησιμοποιήθηκαν 52
4.3 Αρχιτεκτονική συστήματος 53
4.4 Λειτουργίες συστήματος 68

Στο κεφάλαιο αυτό θα περιγραφούν συνοπτικά οι τεχνολογίες που εφαρμόστηκαν για την ανάπτυξη του συστήματος και οι συσκευές που χρησιμοποιήθηκαν για τον έλεγχο και διεκπεραίωση της εφαρμογής. Στη συνέχεια θα αναφερθούμε διαγραμματικά και περιγραφικά στην αρχιτεκτονική του συστήματος και τέλος θα περιγράψουμε τις λειτουργίες που αναπτύχθηκαν για τους σκοπούς της παρούσης διπλωματικής εργασίας.

4.1 Τεχνολογίες-Εργαλεία που χρησιμοποιήθηκαν

4.1.1 Server-Side

Σε αρχικό στάδιο, χρησιμοποιήσαμε τον κώδικα του Open Trip Planner ([19]) για να δημιουργήσουμε πάνω στον δικό μας server στην ηλεκτρονική διεύθυνση http://euclid.grid.ucy.ac.cy:8081/opentripplanner-webapp ένα λειτουργήσιμο παράδειγμα του Open Trip Planner. Για να γίνει αυτό εγκαταστήσαμε την πλατφόρμα Eclipse στον Euclid server του πανεπιστημίου, που τρέχει λειτουργικό σύστημα Ubuntu, κάτω από το account μας και ακολουθήσαμε την διαδικασία που περιγράφεται στο [16].

Όπως αναφέραμε στο κεφ 2.5 , το Open Trip Planner για την δημιουργία του γράφου παίρνει ως είσοδο τα GTFS αρχεία και ένα .osm αρχείο που δηλώνουν την περιοχή που θέλουμε να καλύψουμε. Στις διάφορες δοκιμές που κάναμε, για μεγάλες περιοχές όπως είναι η Νέα Υόρκη τα .osm αρχεία είναι τεράστια και αυτό οδήγησε σε java heap exception λόγω της έλλειψης μνήμης RAM. Επομένως, χρειαστήκαμε κάποιο εργαλείο για να κάνουμε extract από την περιοχή αυτή το κομμάτι εκείνο που μας ενδιέφερε. Χρησιμοποιήσαμε λοιπόν το εργαλείο OSMOSIS ([26]) για να δημιουργήσουμε ένα bounding box γύρω από την περιοχή που μας ενδιαφέρει και να την αποκόψουμε από την υπόλοιπη περιοχή.

Στο παρόν στάδιο στον server χρησιμοποιούμε τα δικά μας Gtfs αρχεία (2.7) και το .osm αρχείο της Κύπρου για να δημιουργήσουμε ένα γράφο που να αφορά αποκλειστικά το δίκτυο δημόσιας συγκοινωνίας της Κύπρου.

Στην πλατφόρμα eclipse εγκαταστήσαμε το plugin Maven για να διαχειριστούμε εύκολα τα dependencies του OTP καθώς και τον Apache Tomcat server 6.0 για να δημοσιεύσουμε το OTP . Επιπλέον, χρησιμοποιούμε php αρχεία που έχουν τον ρόλο ενός web service τα οποία εκτελούν queries σε μια βάση δεδομένων. Τα php αρχεία καλούνται από τον client και είναι δημοσιευμένα μέσω apache server . Στον client η απάντηση δίνεται σε μορφή json. Η βάση δεδομένων υλοποιήθηκε με MySql server και εκεί αποθηκεύουμε τα GTFS δεδομένα για την χρήση τους από το Client-side. Για εύκολη διαχείριση της βάσης δεδομένων εγκαταστήσαμε το εργαλείο phpMyAdmin ([27}). Για να κρατάμε τα GTFS δεδομένα στην τελευταία τους-πιο πρόσφατη μορφή υλοποιήσαμε με php κώδικα μια υπηρεσία η οποία συγκρίνει τα GTFS δεδομένα του server με τα δεδομένα στο διαδίκτυο και εάν παρατηρηθεί αλλαγή τότε αυτόματα φορτώνει τα νέα δεδομένα στην βάση του server. Αυτό γίνεται με την χρήση cron jobs όπου μέσα στο αρχείο cron tab ορίσαμε να εκτελείται αυτόματα ο κώδικας php κάθε μέρα.

4.1.2 Client-Side

Αφού αποφασίσαμε να χρησιμοποιήσουμε λειτουργικό σύστημα Android για την υλοποίηση της εφαρμογής (κεφ 1.2), σε πρώτο στάδιο χρειάστηκε να εγκαταστήσουμε την πλατφόρμα **Eclipse**. Η πλατφόρμα αυτή προσφέρει δυνατότητες προγραμματισμού σε Java και είναι συμβατή με το Android JDK. Σε επόμενο στάδιο χρειάστηκε να εγκαταστήσουμε το **Android SDK** για τον προγραμματισμό σε Android OS.

Για την εφαρμογή χρησιμοποιήσαμε τεχνολογία GPS για την ανάκτηση πληροφοριών σχετικά με την γεωγραφική τοποθεσία του χρήστη. Το GPS πρόκειται για ένα δέκτη που είναι αρκετά ακριβής αφού χρησιμοποιεί τεχνολογία ηλεκτρομαγνητικών κυμάτων καθώς και δορυφορική υποστήριξη. Ο δέκτης GPS είναι εφαρμοσμένος στην κινητή συσκευή Nexus-S που χρησιμοποιήσαμε για την υλοποίηση και έλεγχο της εφαρμογής.

Επιπλέον, χρησιμοποιήσαμε SQLite βάση δεδομένων για να αποθηκεύσουμε locally στην συσκευή συγκεκριμένα δεδομένα (για εύκολο offline access). Όσο αφορά την διαπροσωπεία

του χάρτη, χρησιμοποιούμε διάφορους Map Tile Providers (OSMarender, Mapnik, CycleMap, OSMPublicTransport …)

Tool-Technology	Functionality	Use on project
Eclipse	Develop applications in Java	Project implementation
Android SDK	Develop Android applications	Android Interface design and function implementation
MySQL server	Handling Database Data	Query submission and data retrieval
PhpMyAdmin	Handle MySQL server	Handle MySQL server
php	server-side scripting language	Data retrieval and placement from/in database Retrieve the most updated GTFS data
Cron jobs	Run commands automatically (e.g every week)	Run every day the php script for retrieving the most updated GTFS data
osmosis	Open Street Map tool	Crop area from .osm
SQLite	Database tool for android	Save Data for offline use
Apache Tomcat Server	Use as a server	To deploy OTP for external use

Πίνακας 4.1. Κυρίως τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν

4.2 Συσκευές που χρησιμοποιήθηκαν

Για την υλοποίηση και έλεγχο της εφαρμογής χρησιμοποιήθηκαν οι πιο κάτω συσκευές. Να σημειώσουμε ότι η εφαρμογή προσαρμόστηκε ούτως ώστε να υποστηρίζει πολλαπλές οθόνες (4 ,7 και 10.1 ίντσες) .

Samsung Nexus-S

Κυρίως Χαρακτηριστικά

Talk Time:	6 hours
Stand-By time	428 hours
Ram	512MB
Display	Screen size: 4.0 (inches) Screen resolution: WVGA (800 x 480)
Internal Storage	16Gb
Wi-Fi	802.11 b/g/n
GPS	Yes
OS Version	Android 2.3
CPU	1GHz Cortex A8 (Hummingbird) processor

Samsung Galaxy-tab

Ram	1 GB
Display	Screen size: 7.0 (inches) Widescreen Screen resolution: WVGA (1024 x 600)
Internal Storage	16 GB
Wi-Fi	802.11 b/g/n
GPS	Yes
OS Version	2.3
CPU	1GHz Application Processor with Power Vr SGX540

Motorola Xoom tablet

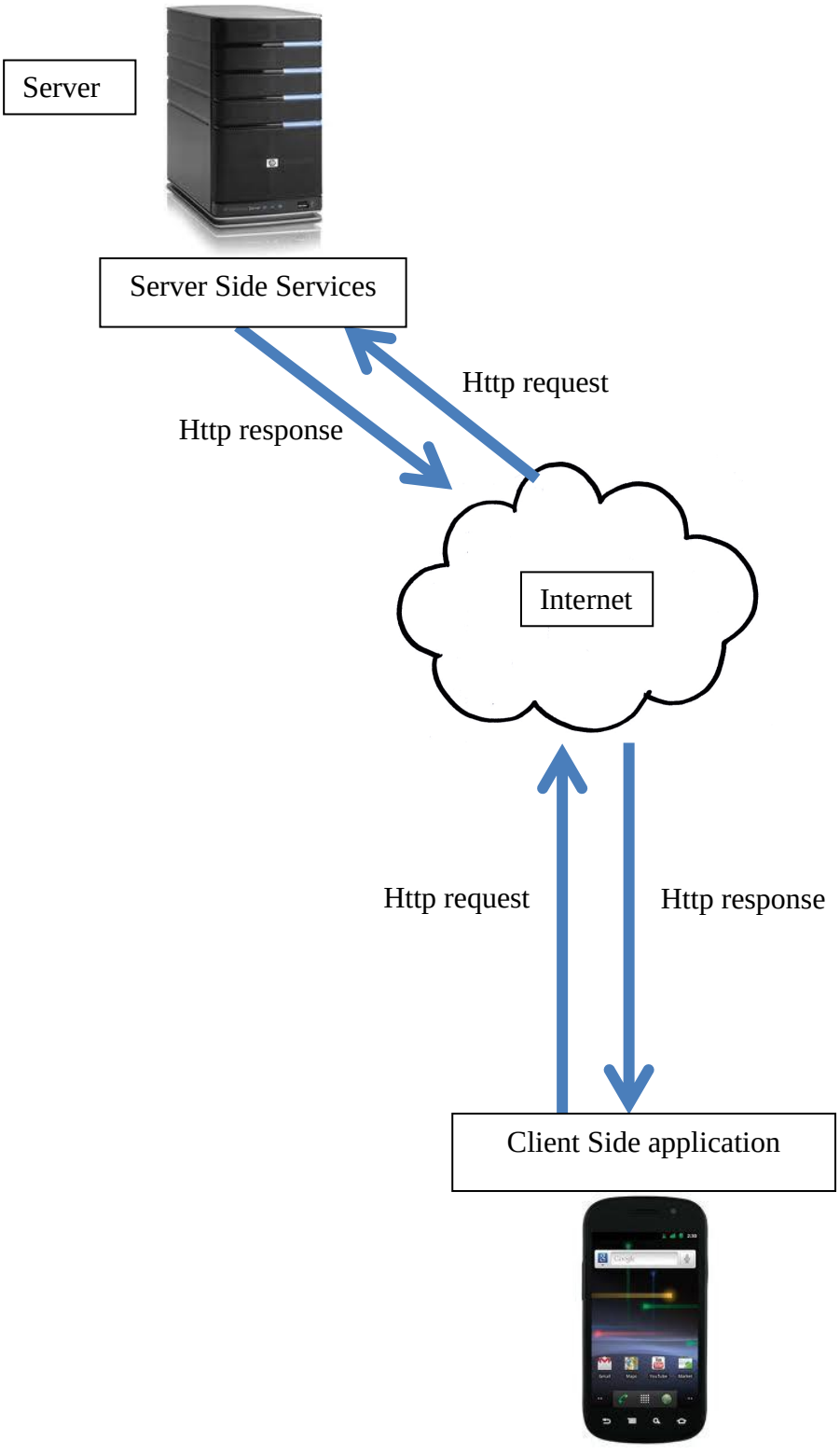
Ram	1 GB
Display	10.1-in.; WXGA (1280x 800 pixels; 150 pixels / inch), HD 720p
Internal Storage	32 GB
Wi-Fi	802.11 b/g/n
GPS	Yes
OS Version	3.2
CPU	1GHz Dual Core

Πίνακας 4.2. Χαρακτηριστικά συσκευών που χρησιμοποιήθηκαν

4.3 Αρχιτεκτονική συστήματος

Αρχικά θα περιγράψουμε την γενική αρχιτεκτονική που ακολουθεί το σύστημά μας και στην συνέχεια θα μιλήσουμε πιο συγκεκριμένα δλδ από ποια τμήματα και υποτμήματα αποτελείται.

4.3.1 Γενική περιγραφή συστήματος



Διάγραμμα 4.3. Γενική περιγραφή συστήματος

Όπως έχουμε ήδη προαναφέρει, το σύστημα που αναπτύξαμε αποτελείται από 2 τμήματα.

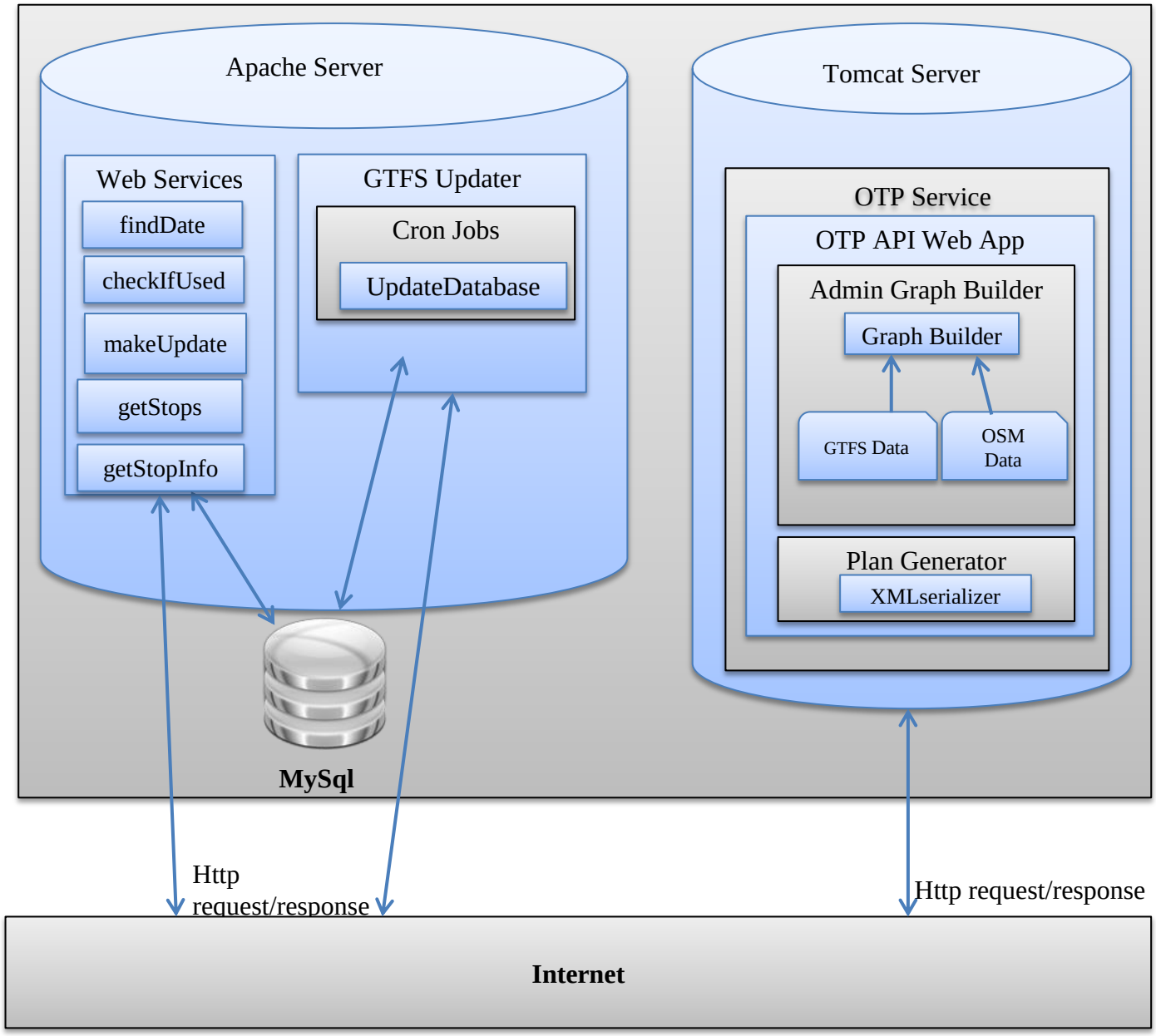
1) Το τμήμα που βρίσκεται εγκατεστημένο στο server.
2) Το τμήμα που αναπτύχθηκε στην κινητή πλατφόρμα.

Το τμήμα που αναπτύχθηκε στον εξυπηρετητή περιλαμβάνει διάφορα web services τόσο για την επικοινωνία με την κινητή συσκευή για την ανταλλαγή δεδομένων όσο και για την αυτόματη ανανέωση των αρχείων GTFS. Στον εξυπηρετητή επίσης βρίσκεται αποθηκευμένος ο γράφος από τον οποίο αντλούμε τις πληροφορίες για την δρομολόγηση των ταξιδιών. Ο γράφος είναι προσβάσιμος μέσω TomCat server.

Στην μεριά του πελάτη βρίσκεται η εφαρμογή EasyTransit. Η εφαρμογή επικοινωνεί με τον server μέσω HTTP. Αποστέλλονται τα διάφορα αιτήματα και λαμβάνονται οι ανάλογες απαντήσεις.

4.3.2 Ανάλυση αρχιτεκτονικής

4.3.2.1 Τμήμα Εξυπηρετητή



Διάγραμμα 4.4. Αρχιτεκτονική server-side

Περιγραφή Server Side Architecture

Under Apache Server

Στον εξυπηρετή που χρησιμοποιούμε εγκαταστήσαμε τον Apache Server ο οποίος υποστηρίζει php γλώσσα προγραμματισμού ούτως ώστε να υλοποιήσουμε τα διάφορα web services και να τα κάνουμε δημοσίευση μέσω http για να είναι εφικτό από την πλευρά του πελάτη να καλεστούν αυτά τα php scripts. Κάτω λοιπόν από τον Apache Server έχουμε υλοποιήσει 2 βασικά τμήματα. Το πρώτο αφορά τα διάφορα web services με τα οποία θα επικοινωνεί ο πελάτης για διάφορες αιτήσεις και το δεύτερο αφορά την αυτόματη ενημέρωση των GTFS αρχείων για την λειτουργία της αναζήτησης στάσεων.

MySql: Στον εξυπηρετητή εγκαταστήσαμε MySql βάση δεδομένων ούτως ώστε να αποθηκεύσουμε τα αρχεία GTFS για να επιτευχθεί η λειτουργία αναζήτησης στάσεων. Στο παρόν στάδιο υπάρχουν αποθηκευμένα στην MySql τα GTFS αρχεία για τις περιοχές Portland, Tampa και Κύπρο (Λευκωσία). Λόγω του μεγάλου μεγέθους των GTFS αρχείων (το αρχείο που αφορά τις στάσεις μπορεί να έχει εκατομμύρια γραμμές) κρίναμε πως θα ήταν σωστό να ορίσουμε indexes μέσα στην βάση ούτως ώστε η προσπέλαση διαφόρων πληροφοριών να είναι πιο γρήγορη από ότι θα ήταν χωρίς τα indexes. Με την βάση δεδομένων **δεν** επικοινωνεί απευθείας ο πελάτης αλλά μέσω των web Services γίνεται η ανταλλαγή πληροφοριών.

Για κάθε περιοχή στην MySql έχουμε ορίσει 2 βάσεις δεδομένων. Για παράδειγμα για την περιοχή Portland έχουμε ορίσει τις βάσεις δεδομένων Portland και Portland2. Μέσα σε κάθε μια από αυτές τις βάσεις, πέραν από τους πίνακες των GTFS δεδομένων, έχουμε δημιουργήσει ένα άλλο πίνακα τον Administrator.

Database Porltand

Tbl Administrator

isUsed	dateTimeModified	OtherDbName
1	07/05/12 12:00:00	Portland2

Database Porltand2

Tbl Administrator

isUsed	dateTimeModified	OtherDbName
0	04/05/12 12:00:00	Portland

Ο πελάτης μέσω του web service θα κάνει αιτήματα στην βάση στην οποία η τιμή του isUsed του πίνακα Administrator έχει τιμή ίση με 1. Η βάση αυτή είναι η πιο ανανεωμένη. Η $2^{η}$ βάση χρειάζεται ούτως ώστε την ώρα που γίνεται η ανανέωση των δεδομένων (GTFS Updater) να γίνεται η χρήση της βάσης η οποία έχει τιμή isUsed=0 και να μην επηρεάζει την αναζήτηση του χρήστη (να μην υπάρχει ταυτόχρονη προσπέλαση των πινάκων από τον χρήστη και από το πρόγραμμά μας).

GTFS Updater

Τα αρχεία GTFS δημιουργούνται και τροποποιούνται από τις εταιρείες που είναι υπεύθυνες για τις υπηρεσίες δημόσιων μέσων μεταφοράς. Οι τροποποιήσεις που συμβαίνουν σε αυτά τα αρχεία μπορεί να είναι σπάνιες (1 φορά τον μήνα) ή και συχνές (καθημερινά). Το πρόβλημα λοιπόν που προκύπτει είναι ότι τα GTFS αρχεία που έχουμε αποθηκευμένα στην βάση δεν θα είναι πάντα στην τελευταία τους μορφή και αυτό αποτελεί ένα σημαντικό πρόβλημα διότι με «ληγμένα» δεδομένα η παρουσίαση πληροφοριών σχετικά με τις στάσεις θα ήταν εντελώς λανθασμένη. Για αυτό το λόγο χρειάστηκε να υλοποιήσουμε μια υπηρεσία η οποία συμβαίνει μόνο στην πλευρά του εξυπηρετητή και αυτόματα κρατά ενήμερη την βάση δεδομένων σε καθημερινή βάση ούτως ώστε τα δεδομένα να είναι pre-fetched, έτοιμα προς χρήση από τον χρήστη.

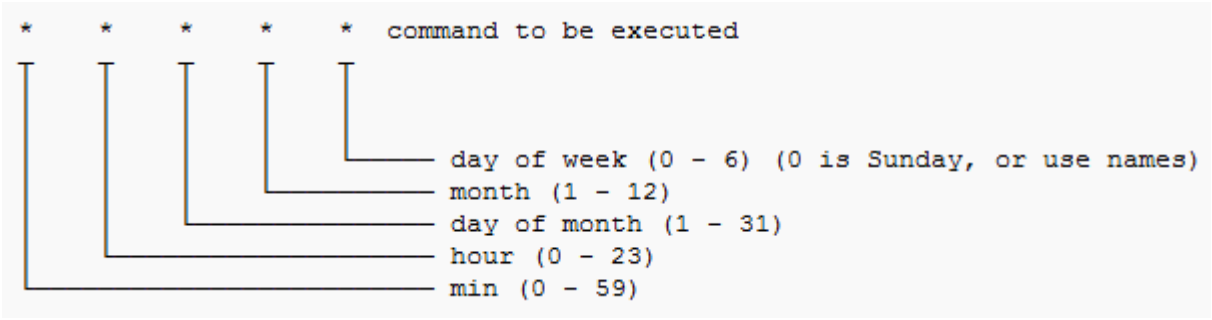
Για να το πετύχουμε αυτό υλοποιήσαμε ένα php script ανά περιοχή, το «UpdateDatabase».
Σε αυτό το αρχείο συμβαίνουν τα εξής.

1. Κατεβάζουμε τα GTFS δεδομένα (.zip) από την πηγή (αναλόγως περιοχής).
2. Κάνουμε extract τα δεδομένα.
3. Διαβάζουμε τα εξαγμένα περιεχόμενα του φακέλου
4. Συγκρίνουμε τα παλιά δεδομένα με τα καινούρια ως προς:
    A. Το μέγεθος τους
    B. Τα περιεχόμενά τους
5. Εάν οποιοδήποτε από τα A ή B είναι διαφορετικό τότε ελέγχουμε ποια βάση από τις 2 (Portland,Portland2) δεν χρησιμοποιείται.
6. Τότε κάνουμε drop όλους τους πίνακες και τους ξαναδημιουργούμε με τα νέα δεδομένα , αναθέτοντας ταυτόχρονα τα indexes σε συγκεκριμένες στήλες.

Για να γίνεται ο έλεγχος για πιο πρόσφατα GTFS δεδομένα σε καθημερινή βάση, χρειάστηκε να χρησιμοποιήσουμε cron jobs. Τα cron jobs επιτρέπουν στους χρήστες unix-based λειτουργικών συστημάτων να χρονοπρογραμματίζουν εντολές,scripts και προγράμματα

ούτως ώστε να τρέχουν σε/ανά συγκεκριμένες χρονικές περιόδους. Εμείς αποφασίσαμε να τρέξουμε το Php script για ανανέωση των δεδομένων στην βάση ανά καθημερινή βάση διότι δεν ξέρουμε την ακριβής χρονική στιγμή που θα αλλάξουν αυτά τα δεδομένα. Για να το πετύχουμε αυτό τροποποιήσαμε το αρχείο crontab και προσθέσαμε την ακόλουθη εντολή για να τρέχει καθημερινά το πρόγραμμά μας η ώρα 12 τα μεσάνυκτα.

```
0 0 * * * Desktop/updateService/Portland/updateDatabase
```



Εικόνα 4.5 Cron job Καθορισμός χρονοπρογράμματος

Web Services

Για την επικοινωνία της βάσης δεδομένων (για την αναζήτηση στάσεων) με τον πελάτη δημιουργήσαμε ενδιάμεσες υπηρεσίες. Αυτές οι υπηρεσίες καλούνται αποκλειστικά από τον πελάτη . Κάθε μια από τις υπηρεσίες αυτές, εάν επιστρέφει δεδομένα, τα κωδικοποιεί σε μορφή json.

findDate: Η υπηρεσία GTFS update που αναφέραμε πιο πάνω έχει το εξής μειονέκτημα. Όπως είπαμε, γίνεται αυτόματα update κάθε μέρα η ώρα 12 το βράδυ. Στην περίπτωση που τα δεδομένα αλλάξουν η ώρα 2 το μεσημέρι, θα υπάρχει ένα κενό 10 ωρών. Μέσα σε αυτές τις ώρες, ένας χρήστης μπορεί να θελήσει να κάνει αναζήτηση μιας στάσης. Τα νέα δεδομένα όμως, δεν έχουν φορτωθεί ακόμα στην βάση δεδομένων διότι ο έλεγχος θα γίνει η ώρα 12 την νύκτα.

Επειδή δεν θέλουμε να δίνουμε λανθασμένες-μη ανανεωμένες πληροφορίες στον πελάτη δημιουργήσαμε αυτό το επιπλέον web service στο οποίο ανευρίσκεται το time-stamp του αρχείου GTFS στον server όπου βρίσκεται ανεβασμένο, και γίνεται σύγκριση με το πεδίο dateTimeModified που υπάρχει στον πίνακα Administrator. Εάν υπάρχει διαφορά στον χρόνο δλδ το dateTimeModified<GTFS timestamp τότε θα εκτελεστεί το php αρχείο **makeUpdate**

που ακολουθεί την ίδια διαδικασία για ανανέωση της βάσης που δείξαμε προηγουμένως. Αλλιώς τα δεδομένα θα μείνουν ως έχουν.

Να αναφέρουμε ότι η υπηρεσία αυτή καλείται κάθε φορά που ο χρήστης κάνει αναζήτηση στάσης αλλά για κάθε ανανέωση δεδομένων που προκύπτει (dateTimeModified<GTFS timestamp) μόνο ένας χρήστης θα επιβαρυνθεί (με λίγα λεπτά μέχρι να γίνει η ανανέωση στην βάση) ενώ οι υπόλοιποι, αφού γίνει η ανανέωση, θα έχουν πρόσβαση στα ανανεωμένα δεδομένα. .

checkIfUsed: Η υπηρεσία αυτή δέχεται ως είσοδο το όνομα μιας βάσης δεδομένων και επιστρέφει κατά πόσο χρησιμοποιείται ή όχι. Αυτό το χρησιμοποιούμε για να διαχωρίσουμε ποια βάση έχει τα ανανεωμένα δεδομένα ούτως ώστε οι διάφορες αιτήσεις να γίνουν στην ανανεωμένη βάση.

getStops,getStopInfo: Η υπηρεσία getStops δέχεται ως είσοδο μία ακολουθία από χαρακτήρες που αντιπροσωπεύει την αναζήτηση στάσης. Συγκεκριμένα εκτελείται το πιο κάτω query:

```
SELECT  s.stop_name, s.stop_id
FROM    stops s
WHERE   SOUNDEX(s.stop_name)=SOUNDEX("stopName") ||
            s.stop_name LIKE '% "stopName”  %'
```

Η αναζήτηση στην βάση δεδομένων γίνεται με 2 τρόπους. Ο πρώτος αφορά την συνάρτηση SOUNDEX η οποία επιστρέφει ένα αριθμό ανάλογα με το πώς ακούονται 2 συμβολοσειρές.
Ο δεύτερος αφορά την συνάρτηση LIKE η οποία επιστρέφει ονόματα στάσεων τα οποία περιέχουν την συμβολοσειρά που αναζήτησε ο χρήστης.
Τα αποτελέσματα του αιτήματος αυτού κωδικοποιούνται σε μορφή json και αποστέλλονται στον πελάτη από όπου αποκωδικοποιούνται και εμφανίζονται ως μια λίστα. Στη συνέχεια ο χρήστης επιλέγει σταθμό μέσα από τη λίστα με τους σταθμούς. Τότε καλείται η υπηρεσία getStopInfo η οποία αναζητά πληροφορίες για τον συγκεκριμένο σταθμό βάση του id που έχει. Οι πληροφορίες που επιστρέφονται στον πελάτη είναι οι ώρες άφιξης και αναχώρησης μιας διαδρομής από τον σταθμό, οι γεωγραφικές συντεταγμένες της στάσης , το είδος μεταφορικού μέσου και το id και το όνομα των γραμμών που περνούν από την συγκεκριμένη στάση.

```
SELECT    st.arrival_time, st.departure_time, r.route_long_name,r.route_id,r.route_type
 FROM     stop_times st, trips t, routes r
WHERE     st.stop_id='".$_REQUEST['stopId']."' AND
          st.trip_id=t.trip_id AND
          t.route_id=r.route_id
ORDER BY r.route_id,st.departure_time
```

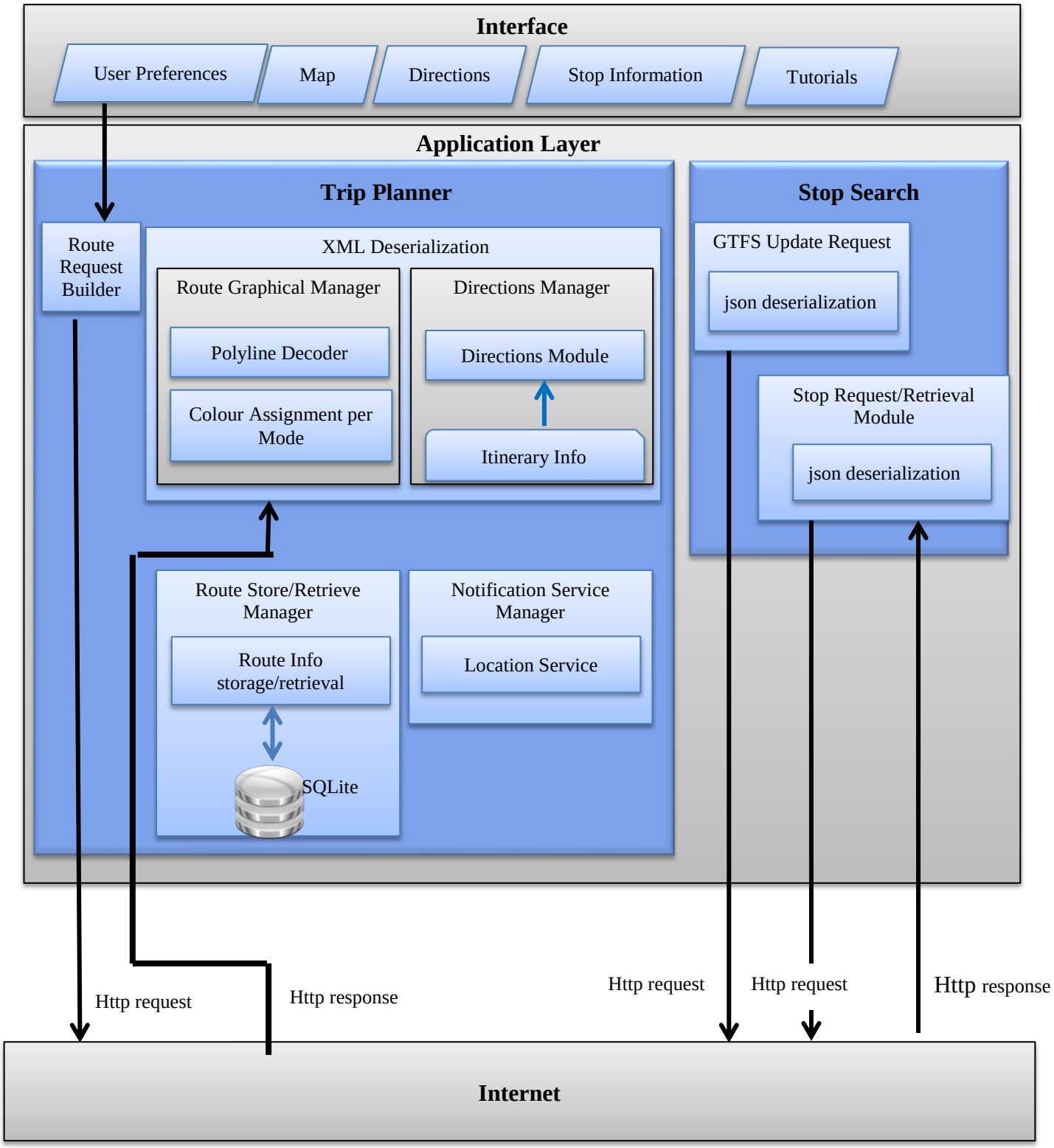
Under TomCat Server

Κάτω από τον Tomcat server βρίσκεται εγκατεστημένη η εφαρμογή του OTP. Αρχικά για την εγκατάσταση της υπηρεσίας που προσφέρει το OTP εγκαταστήσαμε στον εξυπηρετητή μας το eclipse και ακολουθήσαμε τις οδηγίες που περιγράφονται στο [16]. Τα σημαντικότερα modules του OTP που χρησιμοποιήσαμε είναι το module για την κατασκευή του γράφου (graph builder) και το module για να είναι προσβάσιμη η εφαρμογή μέσω του διαδικτύου (api-web app) που περιλαμβάνει και την διαπροσωπεία της εφαρμογής. Για την κατασκευή του γράφου χρειάζεται να δημιουργήσουμε ένα αρχείο xml (graph-config.xml) στο οποίο δίνουμε στοιχεία σχετικά με τα gtfs αρχεία και σχετικά με .osm αρχεία τα οποία είναι απαραίτητα για την δημιουργία του γράφου. Λόγω του ότι είναι δύσκολο για κάποιον που δεν έχει μελετήσει σε λεπτομέρεια το αρχείο αυτό, δλδ δεν ξέρει πως πρέπει να είναι η δομή αυτού του αρχείου ούτως ώστε να το τροποποιήσει ανάλογα και να δημιουργήσει τον γράφο βάση των δικών του δεδομένων, κρίναμε πως ήταν σωστό να υλοποιήσουμε ένα πρόγραμμα που θα διευκολύνει τον administrator να δημιουργήσει και να τρέξει τον γράφο στον tomcat server. Έτσι δημιουργήσαμε το java αρχείο createGraphAndRunServer.java το οποίο καθοδηγεί τον χρήστη να δώσει τα απαραίτητα στοιχεία και έπειτα δημιουργεί το αρχείο graph-config.xml και τρέχει τον γράφο και έπειτα τον tomcat server (ακούει στο port 8080,για περισσότερες λεπτομέρειες στο παράρτημα Α αυτού του εγγράφου)

Αφού τρέξουμε τον γράφο και ανεβάσουμε την εφαρμογή στο tomcat server υπάρχει ένα τμήμα του OTP που διαχειρίζεται τις αιτήσεις του πελάτη.

Το **plan generator** είναι το τμήμα που δέχεται την αίτηση του πελάτη μέσω http και αναλόγως των παραμέτρων που βρίσκονται μέσα σε αυτή την αίτηση (ώρα, ημερομηνία, τύπος ταξιδιού, μέγιστη απόσταση περπατήματος, σημείο εκκίνησης, σημείο τερματισμού) τρέχει τον αλγόριθμο A*, βρίσκει τα δρομολόγια (itineraries) της αίτησης και τα κάνει serialize μέσω των java κλάσεων που υπάρχουν δημιουργώντας έτσι ένα response.xml το οποίο αποστέλλεται στον πελάτη για deserialization.

4.3.2.2 Τμήμα Πελάτη



Διάγραμμα 4.6. Αρχιτεκτονική client-side

Περιγραφή Client Side Architecture

Η εφαρμογή που υλοποιήσαμε χωρίζεται σε 2 βασικά τμήματα. Το τμήμα της διαπροσωπείας που αφορά το τι βλέπει ο χρήστης και τα διάφορα στοιχεία με τα οποία αλληλεπιδρά και το τμήμα εφαρμογής που αποτελεί τον πυρήνα της εφαρμογής, , την υλοποίηση από κάτω ώστε να δουλεύει το σύστημά μας.

Τμήμα διαπροσωπείας

Η διαπροσωπεία αποτελείται από 5 βασικά χαρακτηριστικά.

1. Το πρώτο αφορά τις προτιμήσεις του χρήστη βάση τω οποίων γίνεται ο σχεδιασμός της διαδρομής. Τα χαρακτηριστικά αυτά είναι :

- Το σημείο εκκίνησης και προορισμού
- Η ώρα και ημερομηνία αναχώρησης
- Η προτίμηση σε μεταφορικό μέσο
- Η προτίμηση σε είδος διαδρομής (γρήγορη, πιο λίγες εναλλαγές)
- Η μέγιστη απόσταση περπατήματος
- Η δυνατότητα χρήσης μέσων που να υποστηρίζουν αναπηρικό καροτσάκι

2. Το δεύτερο αφορά την διαπροσωπεία του χάρτη δηλαδή την επιλογή σημείου εκκίνησης και σημείο προορισμού απευθείας πάνω στον χάρτη, την θέαση της διαδρομής , την μεγέθυνση – ελαχιστοποίηση του χάρτη, την προβολή της θέσης του χρήστη πάνω στον χάρτη και την κατεύθυνση της συσκευής.

3. Το τρίτο αφορά την θέαση των οδηγιών στο άνω σημείο της οθόνης. Ο χρήστης μπορεί να πλοηγηθεί στις διάφορες οδηγίες βλέποντας τα διάφορα στάδια των οδηγιών.

4. Με την αναζήτηση στάσεων ο χρήστης μπορεί να ενημερωθεί σχετικά με πληροφορίες στάσεων όπως οι ώρες άφιξης και αναχώρησης μιας διαδρομής από τον σταθμό, οι γεωγραφικές συντεταγμένες της στάσης , το είδος μεταφορικού μέσου και το id και το όνομα των γραμμών που περνούν από την συγκεκριμένη στάση.

5. Οι οδηγίες χρήσης (tutorials) των διαφόρων λειτουργιών παρέχουν οδηγίες με κείμενο και εικόνες σχετικά με το πώς χρησιμοποιούνται οι διάφορες λειτουργίες της εφαρμογής.

Τμήμα εφαρμογής

Στο τμήμα εφαρμογής βρίσκονται υλοποιημένες οι λειτουργίες της εφαρμογής.

A. Route Request Builder

Στο τμήμα αυτό δημιουργείται η αίτηση για τον σχεδιασμό δρομολογίου βάση των προτιμήσεων του χρήστη η οποία αίτηση αποστέλλεται στο plan generator στην πλευρά του εξυπηρετητή μέσω http.

XML Deserialization

Το τμήμα αυτό λαμβάνει την απάντηση response.xml του αιτήματος που θέσαμε μέσω http από τον εξυπηρετητή και το αποκωδικοποιεί με deserialization, δημιουργεί δλδ ένα αντικείμενο response το οποίο περιλαμβάνει το δρομολόγιο της διαδρομής μαζί τις πληροφορίες που το διέπουν. Το XML deserialization χωρίζεται σε 2 βασικά τμήματα:

1. Route Graphical Manager: Το υποτμήμα αυτό είναι υπεύθυνο για τον σχεδιασμό της διαδρομής πάνω στον χάρτη.

Το αρχείο response.xml που επιστρέφεται στην εφαρμογή για να γίνει deserialized, περιλαμβάνει τον κώδικα για ένα συγκεκριμένο leg. Το leg αφορά ένα συγκεκριμένο κομμάτι της διαδρομής, για παράδειγμα το δρομολόγιο ενός λεωφορείου για να πάμε από την μια στάση στην άλλη ή τις οδηγίες για να περπατήσουμε από ένα σημείο σε άλλο.

```
<legGeometry>
      <length>15</length>
      <points>

o_reFxtcjVbH`KlHrJ|LrMhGoInFoHvFyHvFsHhEqHrDsPbPu@dHUdOq@`Oo@|H]
      </points>
</legGeometry>
```

Μέσα στο στοιχείο points συμπεριλαμβάνεται ένα κωδικοποιημένο μήνυμα που είναι ένα κωδικοποιημένο πολύγραμμο (encoded polyline). Στο [28] περιγράφεται με λεπτομέρεια ο αλγόριθμος αυτός. Γενικά το κωδικοποιημένο πολύγραμμο αποθηκεύει 2 ειδών πληροφορίες για οποιοδήποτε σύνολο σημείων.

1. Το γεωγραφικό πλάτος και μήκος αυτών των σημείων
2. Τα μέγιστα σημεία μεγέθυνσης για να εμφανιστούν αυτά τα σημεία.

Για την κωδικοποίηση των γεωγραφικών συντεταγμένων θα δούμε το πιο κάτω παράδειγμα:

1. Ας πάρουμε για παράδειγμα την αρχική τιμή:

   -179.9832104

2. Παίρνουμε το δεκαδικό κομμάτι και το πολλαπλασιάζουμε με 1e5,

στρογγυλοποιώντας το αποτέλεσμα

-17998321

3. Μετατρέπουμε το αποτέλεσμα σε δυαδικό . Να σημειώσουμε ότι για αρνητικές τιμές πρέπει να υπολογίζουμε το συμπλήρωμα του 2 με το να αντιστρέψουμε την δυαδική τιμή και να προσθέσουμε 1 bit.

00000001 00010010 10100001 11110001 (δυαδικό αποτέλεσμα)

11111110 11101101 01011110 00001110 (αντιστροφή αποτελέσματος)

11111110 11101101 01011110 00001111 (προσθέτουμε 1 bit)

4. Κάνουμε Left-shift το δυαδικό αποτέλεσμα κατά 1 bit

11111101 11011010 10111100 00011110

5. Εάν η αρχική τιμή ήταν αρνητική τότε αντιστρέφουμε τα ψηφία

00000010 00100101 01000011 11100001

6. Σπάζουμε την δεκαδική τιμή σε κομμάτια των 5 bit ξεκινώντας από τα δεξιά προς τα αριστερά.

00001 00010 01010 10000 11111 00001

7. Τοποθετούμε τα κομμάτια των 5 bit σε αντίστροφη σειρά

00001 11111 10000 01010 00010 00001

8. Κάνουμε OR κάθε τιμή με 0x20 για κάθε ένα κομμάτι Bit που ακολουθεί

100001 111111 110000 101010 100010 000001

9. Μετατρέπουμε κάθε τιμή σε δεκαδικό αριθμό

33 63 48 42 34 1

10.Προσθέτουμε 63 σε κάθε τιμή

96 126 111 105 97 64

11. Μετατρέπουμε κάθε τιμή στο αντίστοιχο Ascii κώδικα.

`~oia@

Latitude	Longitude	Change In Latitude	Change In Longitude	Encoded Latitude	Encoded Longitude	Encoded Point
38.5	-120.2	+3850000	-12020000	`_p~iF`	`~ps|U`	`_p~iF~ps|U`
40.7	-120.95	+220000	-75000	`_ulL`	`nnqC`	`_ulLnnqC`
43.252	-126.453	+255200	-550300	`_mqN`	`vxq`@`	`_mqNvxq`@`

Πίνακας 4.7. Παράδειγμα Polyline Encoder

Στην εφαρμογή μας έχουμε τοποθετήσει μια συνάρτηση αποκωδικοποίησης που αποκωδικοποιεί τους πιο πάνω asccii code χαρακτήρες σε γεωγραφικές συντεταγμένες. Δίνουμε ως είσοδο μια την κωδικοποιημένη συμβολοσειρά και επιστρέφει πίσω τις συντεταγμένες.

```
//Borrowed from http://jeffreysambells.com/posts/2010/05/27/decoding-polylines-
from-google-maps-direction-api-with-java/
   public static List<GeoPoint> decodePoly(String encoded) {
           List<GeoPoint> poly = new ArrayList<GeoPoint>();
           int index = 0, len = encoded.length();
           int lat = 0, lng = 0;

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

               shift = 0;
               result = 0;
               do {
                   b = encoded.charAt(index++) - 63;
                   result |= (b & 0x1f) << shift;
                   shift += 5;
               } while (b >= 0x20);
               int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
               lng += dlng;
               GeoPoint p = new GeoPoint((int) (((double) lat / 1E5) * 1E6),
                    (int) (((double) lng / 1E5) * 1E6));
               poly.add(p);
           }
           return poly;
       }
```

Color Assignment per mode

Ανάλογα με το mode (μεταφορικό μέσο) του leg χρωματίζουμε την διαδρομή ανάλογα.

Directions Manager

Στο υποτμήμα αυτό κάνουμε deserialize το response.xml που προέρχεται από τον εξυπηρετητή και δημιουργούμε μία λίστα από ένα struct η οποία περιλαμβάνει τα διάφορα βήματα μέχρι να φτάσουμε στον προορισμό μας.

```
public class DirectionsStruct implements Serializable{

       String str;
       double startLon;
       double startLat;
       double endLon;
```

```
        double endLat;
        String mode;
        String generalInfo;
        double fares;
        String dropOff;
        String dropOffId;
.
.
}
```

Το struct αυτό περιλαμβάνει πληροφορίες σχετικά με τις γενικές πληροφορίες του ταξιδιού όπως είναι τα μέσα μεταφοράς, οι γεωγραφικές συντεταγμενες εκκίνησης και τερματισμού για κάθε leg, τα ναύλα του ταξιδιού, οι ώρες εκκίνησης και τερματισμού .

Στη συνέχει βάση αυτού του αντικειμένου παρουσιάζουμε στον χρήστη με κάποιο μηχανισμό πλοήγησης τα διάφορα βήματα των οδηγιών.

Route Store retrieve manager

Βασική λειτουργία της εφαρμογής μας είναι ο χρήστης να μπορεί να χρησιμοποιήσει την λειτουργία σχεδιασμού διαδρομής όταν είναι offline. Για την δρομολόγηση όμως της διαδρομής χρειάζεται να υπάρχει επικοινωνία με τον εξυπηρετητή που συνεπώς χρειάζεται σύνδεση με το διαδίκτυο. Για τον λόγο αυτό αποφασίσαμε να προσθέσουμε την λειτουργία για τοπική αποθήκευση των πρόσφατων δρομολογίων που σχεδίασε ο χρήστης ούτως ώστε να έχει πρόσβαση σε αυτά όταν δεν υπάρχει διασύνδεση με το διαδίκτυο.

Για να το πετύχουμε αυτό χρησιμοποιήσαμε την βάση δεδομένων SQlite που προσφέρει το Android SDK.

Κάθε φορά που ο χρήστης κάνει σχεδιασμό ενός ταξιδιού πάμε και αποθηκεύουμε το response.xml που επιστρέφει ο χρήστης μέσα στην τοπική βάση δεδομένων μαζί με την ημερομηνία και ώρα εισαγωγής. Κρατάμε μόνο τις τελευταίες 20 πιο πρόσφατες εγγραφές. Έτσι ο χρήστης όταν είναι offline μπορεί να πλοηγηθεί στην επιλογή recent trips και να επιλέξει ένα από αυτά τα ταξίδια.

Notification Service Manager

Το τμήμα αυτό της εφαρμογής αφορά την εμφάνιση μηνύματος notification στον χρήστη όταν έχει σχεδιασμένο ένα ταξίδι . Το μήνυμα εμφανίζεται όταν ο χρήστης φτάσει στην προτελευταία στάση από την στάση που πρέπει να κατεβεί.

Βασικό στοιχείο αυτής της λειτουργίας είναι ότι πρέπει να δουλεύει ακόμα και όταν η κινητή συσκευή είναι locked, πρέπει δλδ να τρέχει στο υπόβαθρο. Για τον λόγο αυτό υλοποιήσαμε ένα Service το οποίο κάνει extend το location listener.

Υπάρχουν 2 περιπτώσεις:

1. Ο χρήστης χρησιμοποιεί κάποιο μεταφορικό μέσο που δεν επηρεάζει την λήψη του GPS όπως για παράδειγμα τα λεωφορεία, τα πλοία, τα υπέργεια τρένα. Σε αυτή την περίπτωση το μήνυμα αποστέλλεται βάση της γεωγραφικής συντεταγμένης του χρήστη. Κρατάμε την γεωγραφική συντεταγμένη της προτελευταίας στάσης και ελέγχουμε συνεχώς με την παρούσα γεωγραφική θέση του χρήστη. Όταν οι 2 γεωγραφικές συντεταγμένες συμπέσουν, τότε αποστέλλεται το ανάλογο notification στον χρήση με ήχο και δόνηση (εάν το υποστηρίζει η συσκευή).

Σε περίπτωση που το σήμα του GPS χαθεί την ώρα που πρέπει να σταλεί το μήνυμα στον χρήστη τότε το χειριζόμαστε με κάποια άλλη πολιτική. Όταν ξαναέρθει το σήμα του GPS παρακολουθούμε την απόσταση του χρήστη από τις συντεταγμένες της προτελευταίας στάσης και εάν η απόσταση μεγαλώνει τότε σημαίνει ότι απομακρυνόμαστε από την στάση αυτή. Σε αυτή την περίπτωση αποστέλλουμε το μήνυμα που έπρεπε να έχει σταλεί. Στην άλλη περίπτωση δεν κάνουμε τίποτα.

2. Ο χρήστης χρησιμοποιεί κάποιο μεταφορικό μέσο στο οποίο δεν είναι δυνατή η λήψη σήματος GPS όπως για παράδειγμα το μετρό. Σε τέτοια περίπτωση έχουμε αποθηκευμένη την ώρα άφιξης στην προτελευταία στάση και ελέγχουμε με την ώρα της κινητής συσκευής. Στην περίπτωση που η διαφορά που παρατηρείται είναι μικρότερη ή ίση του ενός λεπτού τότε στέλλουμε στον χρήστη το μήνυμα να κατεβεί στην επόμενη στάση.

B. Stop Search

Στο τμήμα αυτό ο χρήστης μπορεί να αναζητήσει πληροφορίες σχετικά με τις στάσεις δεδομένων. Αρχικά ο χρήστης επιλέγει περιοχή από την οποία θέλει να κάνει την αναζήτηση στάσεων.

GTFS Update Request

Αφού ο χρήστης επιλέξει περιοχή τότε γίνεται μια http κλήση στο web service findDate.php που βρίσκεται στον εξυπηρετητή ούτως ώστε να ελέγξουμε εάν υπάρχουν πιθανά updates στα GTFS αρχεία. Η υπηρεσία στον εξυπηρετητή κωδικοποιεί την απάντηση σε json μορφή και το αποστέλλει στον πελάτη. Χρησιμοποιούμε json deserializer για να αποκωδικοποιήσουμε την απάντηση αυτή. Εάν η απάντηση είναι θετική τότε κάνουμε ένα δεύτερο http αίτημα στη υπηρεσία makeUpdate η οποία είναι υπεύθυνη για την ανανέωση της βάσης δεδομένων στον server.

Stop Request/Retrieval module

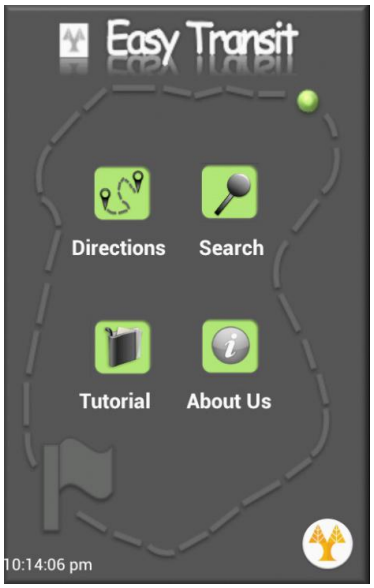
Ο χρήστης μπορεί να αναζητήσει πληροφορίες στάσεων βάση του ονόματός τους. Με την πληκτρολόγηση του ονόματος που επιθυμεί ο χρήστης και με το πάτημα του κουμπιού αναζήτησης, η εφαρμογή στέλλει ένα http αίτημα στην υπηρεσία getStops η οποία αναζητά

στην βάση δεδομένων για στάσεις που το όνομά τους περικλείεται στην συμβολοσειρά που έδωσε ο χρήστης. Το αποτέλεσμα που επιστρέφεται από τον εξυπηρετητή αποκωδικοποιείται με τη χρήση json deserializer και παρουσιάζονται τα αποτελέσματα (όνομα στάσης και id) στον χρήστη μέσα σε μια λίστα.
Στη συνέχεια ο χρήστης επιλέγει ένα σταθμό από την λίστα και αποστέλλεται μία http αίτηση στην υπηρεσία getStopInfo στον εξυπηρετητή . Ανάλογα με το id του σταθμού επιστρέφεται πίσω μία απάντηση κωδικοποιημένη σε json μορφή που περικλείει πληροφορίες σχετικά με την συγκεκριμένη στάση λεωφορείων.

4.4 Λειτουργίες του συστήματος

Το σύστημα παρουσιάζει διάφορες λειτουργίες και θα προσπαθήσουμε να εξηγήσουμε την κάθε μια από αυτές με όσο το μεγαλύτερη λεπτομέρεια γίνεται, παραθέτοντας όπου χρειάζεται σχετικές εικόνες.

Με το που ανοίγει ο χρήστης την εφαρμογή, του παρουσιάζεται μια αρχική οθόνη που περιλαμβάνει τις επιλογές για Δρομολόγηση Διαδρομής (Directions), για Αναζήτηση Στάσης (Search), για εξερεύνηση των λειτουργιών του συστήματος μέσω οθονών εκμάθησης (Tutorials) και την επιλογή να δει πληροφορίες σχετικά με την ανάπτυξη της εφαρμογής (About us). Πιο κάτω θα δούμε συγκεκριμένα κάθε μια από αυτές τις επιλογές.

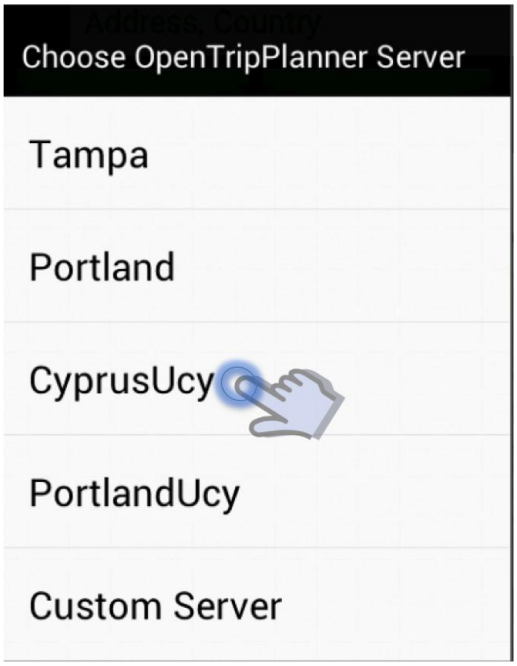


Εικόνα 4.8 Αρχική Οθόνη Εφαρμογής

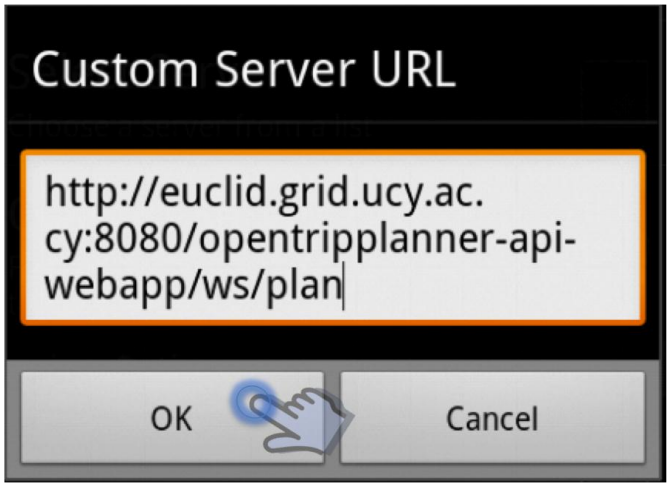
4.4.1 Δρομολόγηση Διαδρομής

4.4.1.1 Επιλογή εξυπηρετητή

Η λειτουργία δρομολόγησης διαδρομής αφορά την βασική λειτουργία του συστήματος. Ο χρήστης με το που επιλέγει το εικονίδιο Directions μεταφέρεται σε μια άλλη οθόνη που στο λειτουργικό σύστημα android μεταφράζεται ως Activity. Στην οθόνη αυτή, εάν υπάρχει σύνδεση με το διαδίκτυο, τότε εμφανίζεται στον χρήστη μία λίστα με τους πιθανούς OTP servers (ανά περιοχή) με τους οποίους μπορεί να ενωθεί ο χρήστης. Η λίστα αυτή ελέγχεται από τον διαχειριστή από την πλευρά του εξυπηρετητή και πρόκειται για ένα .xls αρχείο το οποίο περιλαμβάνει πληροφορίες σχετικά με τα URLs των OTP εξυπηρετητών καθώς και κάποιες άλλες πληροφορίες σχετικά με επικοινωνία με αρμόδια άτομα σε περίπτωση που υπάρχει κάποιο πρόβλημα. Ο χρήστης έχει την δυνατότητα, εάν θελήσει, να πληκτρολογήσει την ηλεκτρονική διεύθυνση ενός OTP εξυπηρετητή που έχει υπόψη του και δεν βρίσκεται στην λίστα που του παρέχεται. Στην περίπτωση που δεν υπάρχει σύνδεση με το διαδίκτυο η εφαρμογή εμφανίζει ενημερωτικό μήνυμα.



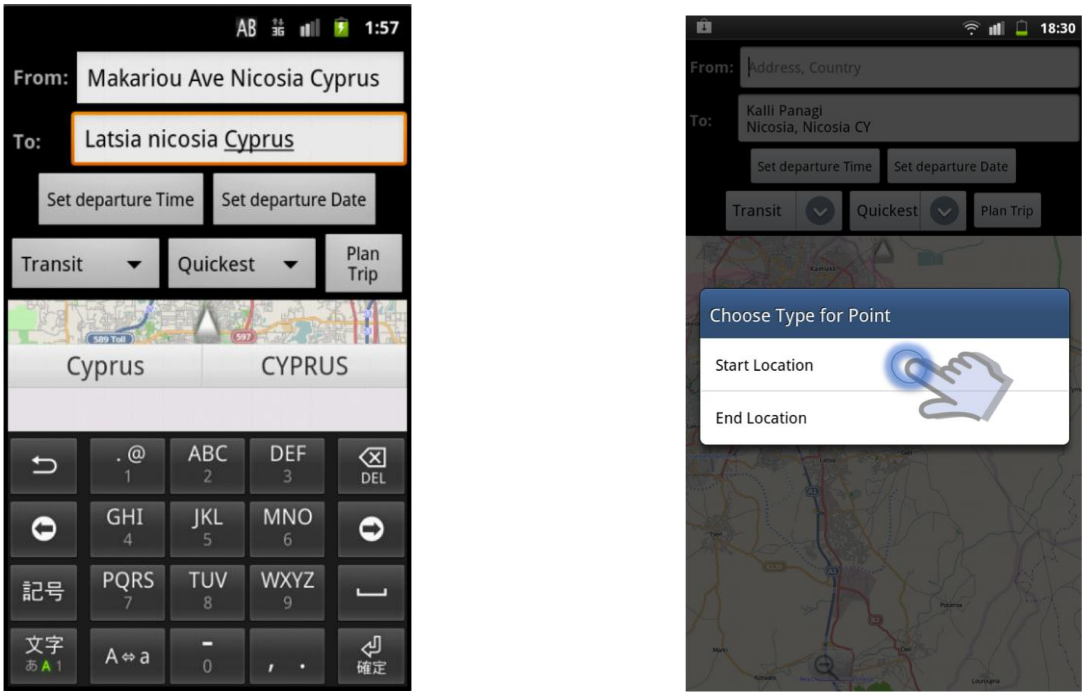
Εικόνα 4.9 Επιλογή Εξυπηρετητή



Εικόνα 4.10 Χειροκίνητη επιλογή εξυπηρετητή

4.4.1.2 Σχεδιασμός διαδρομής

Αφού ο χρήστης επιλέξει τον εξυπηρετητή που επιθυμεί, τότε είναι έτοιμος να δρομολογήσει το ταξίδι του. Η εφαρμογή προσφέρει στον χρήστη την δυνατότητα να επιλέξει μεταξύ 2 διαφορετικών τρόπων δρομολόγησης. Ο πρώτος αφορά την τοποθέτηση της διεύθυνσης μέσα στα πεδία ‘From’ και ‘To’. Αυτόματα μία υπηρεσία της Google που ονομάζεται Google Geocoder μεταφράζει την διεύθυνση σε γεωγραφικές συντεταγμένες. Ο δεύτερος τρόπος αφορά την τοποθέτηση πινέζας προορισμού και εκκίνησης στον χάρτη. Αυτό φυσικά προϋποθέτει ότι ο χρήστης είναι γνώστης της περιοχής και γνωρίζει που βρίσκεται και που θέλει να πάει.

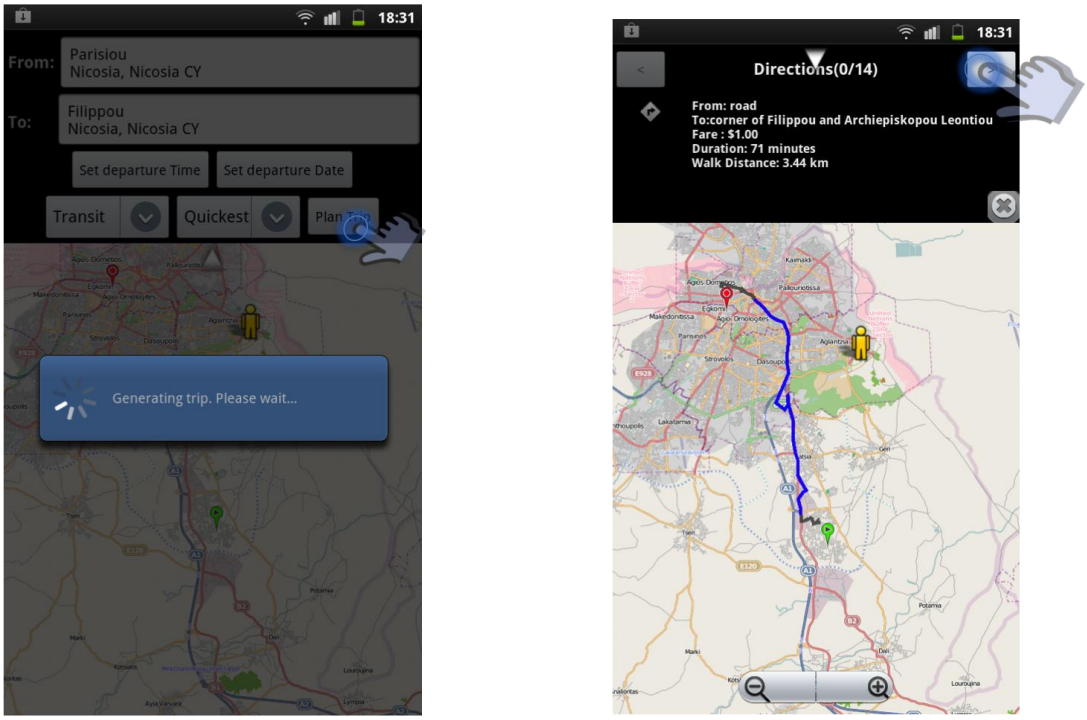


Εικόνα 4.11 Μέθοδοι δρομολόγησης

Ο χρήστης κατά τον σχεδιασμό διαδρομής μπορεί να επιλέξει ώρα και μέρα αναχώρησης, μεταφορικό μέσο (λεωφορείο, τρένο, ποδήλατο, περπάτημα) καθώς και να επιλέξει ανάμεσα στην πιο γρήγορη διαδρομή ή στην διαδρομή που επιφέρει τις λιγότερες μετεπιβιβάσεις.

Εικόνα 4.12 Προτιμήσεις χρήστη

Στην συνέχεια πατώντας το κουμπί Plan Trip, στέλλεται ένα http request στον εξυπηρετητή που επέλεξε ο χρήστης και υπολογίζεται η διαδρομή βάση των προτιμήσεων του. Η διαδρομή τότε μεταφράζεται σε ένα XML αρχείο το οποίο αποστέλλεται στον πελάτη. Στην πλευρά του πελάτη το αρχείο αυτό αποκωδικοποιείται και παρουσιάζεται στον χρήστη τόσο σε γραφική αναπαράσταση (χάρτης) όσο και σε μορφή κειμένου (οδηγίες που πρέπει να ακολουθηθούν για να φτάσει στον προορισμό του).



Εικόνα 4.13 Παρουσίαση διαδρομής και οδηγιών

Η γραφική αναπαράσταση της διαδρομής στον χάρτη περιλαμβάνει 2 κυρίως χαρακτηριστικά. Περιλαμβάνει το σημείο εκκίνησης και προορισμού και την διαδρομή. Η

διαδρομή ανάλογα με το είδος μεταφορικού μέσου έχει διαφορετικό χρώμα. Συγκεκριμένα πιο κάτω βλέπουμε ένα πίνακα με τα χρώματα και τι σημαίνει το κάθε χρώμα.

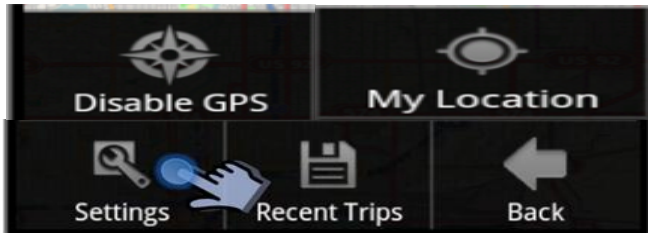
Μέσο Μεταφοράς	Χρώμα
Περπάτημα	Γκρίζο
Λεωφορείο	Μπλε
Τρένο/Τραμ	Πράσινο
Ποδήλατο	Μωβ
Άλλο είδος	Άσπρο

Πίνακας 4.14 Αντιστοίχιση Μέσων Μεταφοράς με τα χρώματα

Οι οδηγίες που εμφανίζονται στον χρήστη αρχικά περιγράφουν τα γενικά στοιχεία της διαδρομής όπως το σημείο εκκίνησης, το σημείο προορισμού, τα ναύλα για την μεταφορά, την διάρκεια του ταξιδιού και την συνολική απόσταση που θα περπατήσει για να φτάσει στον προορισμό του.

Ο χρήστης μπορεί να πλοηγηθεί στις οδηγίες πατώντας τα κουμπιά που βρίσκονται στο πάνω μέρος της οθόνης αριστερά και δεξιά. Με το πάτημα του κουμπιού γίνεται αυτόματα μεγέθυνση του χάρτη στο σημείο εκείνο που αναφέρονται οι οδηγίες.

4.4.1.3 Επιπλέον επιλογές



Εικόνα 4.15 Επιπλέον επιλογές

Οι επιπλέον επιλογές μιας εφαρμογής στις συσκευές με λειτουργικό σύστημα Android εμφανίζονται με το πάτημα ενός συγκεκριμένου κουμπιού. Η εφαρμογή μας παρέχει τις δικές τις επιλογές που εξυπηρετούν διάφορους σκοπούς.

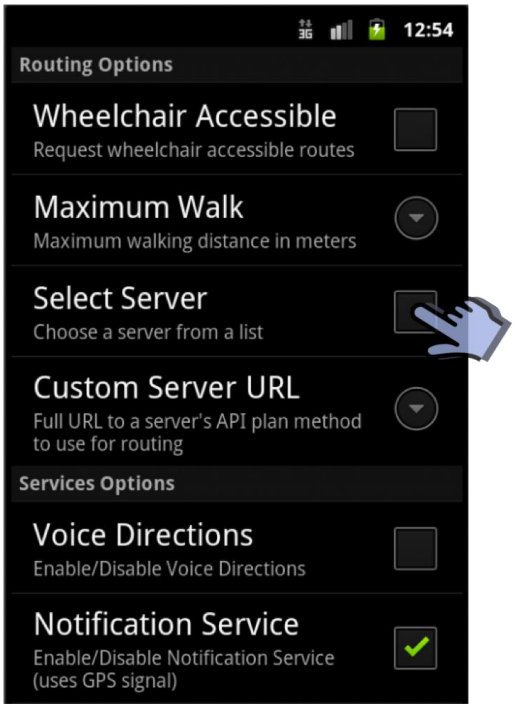
Συγκεκριμένα η εφαρμογή μας έχει τις πιο κάτω επιλογές :

1. **Enable/Disable GPS:** Επιλέγοντας αυτό το κουμπί μεταφερόμαστε στις ρυθμίσεις του κινητού τηλεφώνου και συγκεκριμένα στο σημείο εκείνο των ρυθμίσεων που αφορά την ενεργοποίηση/απενεργοποίηση του GPS δέκτη. Αυτή η επιλογή είναι απαραίτητη εφόσον η εφαρμογή για να δουλέψει σωστά σε κινητή συσκευή χρειάζεται να υπάρχει

ενεργοποιημένος ο δέκτης GPS. Έτσι ο χρήστης χωρίς να χρειαστεί να βγει από την εφαρμογή, μπορεί να ενεργοποιήσει το δέκτη.

2. **My location:** Επιλέγοντας αυτό το κουμπί και με την προϋπόθεση ότι ο δέκτης GPS ή η αναγνώριση τοποθεσίας χρήστη βάση wifi είναι ενεργοποιημένη, ο χάρτης κεντράρεται ανάλογα με τις γεωγραφικές συντεταγμένες του χρήστη.

3. **Settings:** Πρόκειται για μια λίστα από διάφορες ρυθμίσεις με τις οποίες ο χρήστης αλληλεπιδρά για να αλλάξει διάφορες υπηρεσίες που του παρέχονται.



Εικόνα 4.16 Ρυθμίσεις

3.1. **Wheelchair Accessible:** με την επιλογή αυτή ο χρήστης ζητά διαδρομές που να υποστηρίζουν δημόσια μέσα μεταφοράς που να είναι προσιτά σε αναπηρικά καροτσάκια.

3.2. **Maximum Walk:** Ο χρήστης έχει την δυνατότητα να αλλάξει την μέγιστη απόσταση περπατήματος σε ένα ταξίδι. Η μονάδα μέτρησης της τιμής είναι σε μέτρα.

3.3 **Select Server:** Ο χρήστης με αυτή την επιλογή μπορεί να αλλάξει τον OTP εξυπηρετητή στον οποίο θα κάνει τα αιτήματα για την δρομολόγηση μιας διαδρομής.

3.4. **Custom Server URL**: Όπως αναφέραμε και πριν, ο χρήστης έχει την δυνατότητα να βάλει ένα OTP server που δεν υπάρχει στην λίστα που του παρέχει η εφαρμογή.

3.5. **Voice Directions:** Με την έννοια «Φωνητικές Οδηγίες» εννοούμε την δυνατότητα μετατροπής του κειμένου που αφορά τις οδηγίες του ταξιδιού σε ακουστική αναπαράσταση (text-to-speech). Η λειτουργία αυτή βρίσκει απήχηση σε άτομα τα οποία έχουν κάποια προβλήματα όρασης (για παράδειγμα ηλικιωμένα άτομα) και έτσι θα τους

διευκολύνουμε στην κατανόηση αυτών των οδηγιών. Στο παρόν στάδιο οι φωνητικές οδηγίες υποστηρίζουν μόνο την αγγλική γλώσσα.

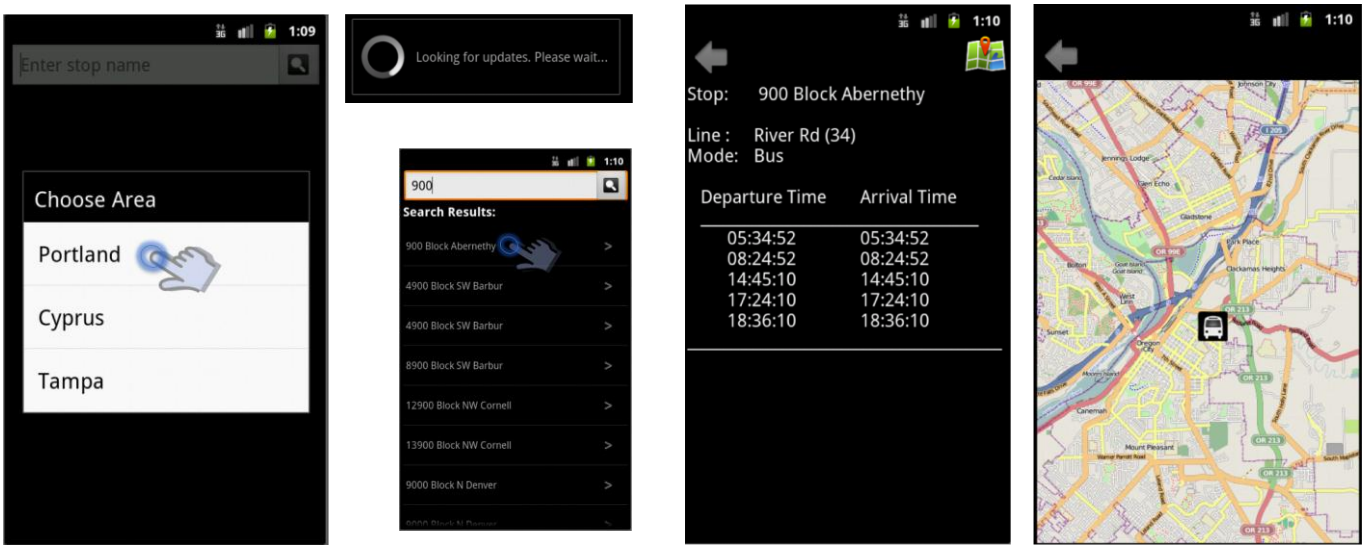
3.6 **Notification Service:** Η υπηρεσία ειδοποίησης είναι μια αρκετά σημαντική λειτουργία αυτής της εφαρμογής και είναι μάλιστα η κύρια λειτουργία που την κάνει να ξεχωρίζει από άλλες υφιστάμενες εφαρμογές. Η υπηρεσία αυτή έχει ουσία όταν ο χρήστης σχεδιάσει μια διαδρομή. Καθώς ο χρήστης κινείται , για παράδειγμα βρίσκεται μέσα σε ένα λεωφορείο, η υπηρεσία χρησιμοποιεί το δέκτη GPS για να λάβει τις γεωγραφικές συντεταγμένες του. Όταν θα πρέπει να κατεβεί σε μια στάση, η υπηρεσία ειδοποιεί τον χρήστη στην αμέσως προηγούμενη στάση (προ-τελευταία στάση) ενημερώνοντας τον (με δόνηση και notification) ότι θα πρέπει να κατεβεί στην x στάση σε n μέτρα. Έτσι ο χρήστης θα μπορεί να ταξιδεύει με τα δημόσια μέσα μεταφοράς χωρίς να ανησυχεί συνεχώς για το πότε πρέπει να κατέβει. Στην περίπτωση που το GPS δεν λαμβάνει σήμα, όπως για παράδειγμα στα μετρό, η εφαρμογή συγκρίνει την ώρα της δεδομένης στιγμής με την ώρα που καταφθάνει το τρένο στην προτελευταία στάση και εάν η διαφορά είναι μικρότερη ή ίση του ενός λεπτού τότε αποστέλλεται μήνυμα. Σημαντικό χαρακτηριστικό αυτής της λειτουργίας είναι ότι δουλεύει ακόμα και εάν ο χρήστης πλοηγηθεί εκτός της εφαρμογής, δουλεύει δλδ και όταν το τηλέφωνο είναι κλειδωμένο.

4. **Recent Trips:** Η επιλογή αυτή παρέχει την δυνατότητα στον χρήστη να δει προηγούμενες διαδρομές που έχει σχεδιάσει. Συγκεκριμένα, η εφαρμογή φυλάει τοπικά τις 20 προηγούμενες διαδρομές που έχει σχεδιάσει ο χρήστης και τις ταξινομεί ανά πιο πρόσφατη ημερομηνία. Η χρησιμότητα αυτής της λειτουργίας είναι ότι ο χρήστης θα μπορεί να δει μια προηγούμενη διαδρομή ακόμα και όταν δεν είναι συνδεδεμένος με το διαδίκτυο.

Εικόνα 4.17 Πρόσφατες διαδρομές

4.4.2 Αναζήτηση Σταθμού/Στάσης

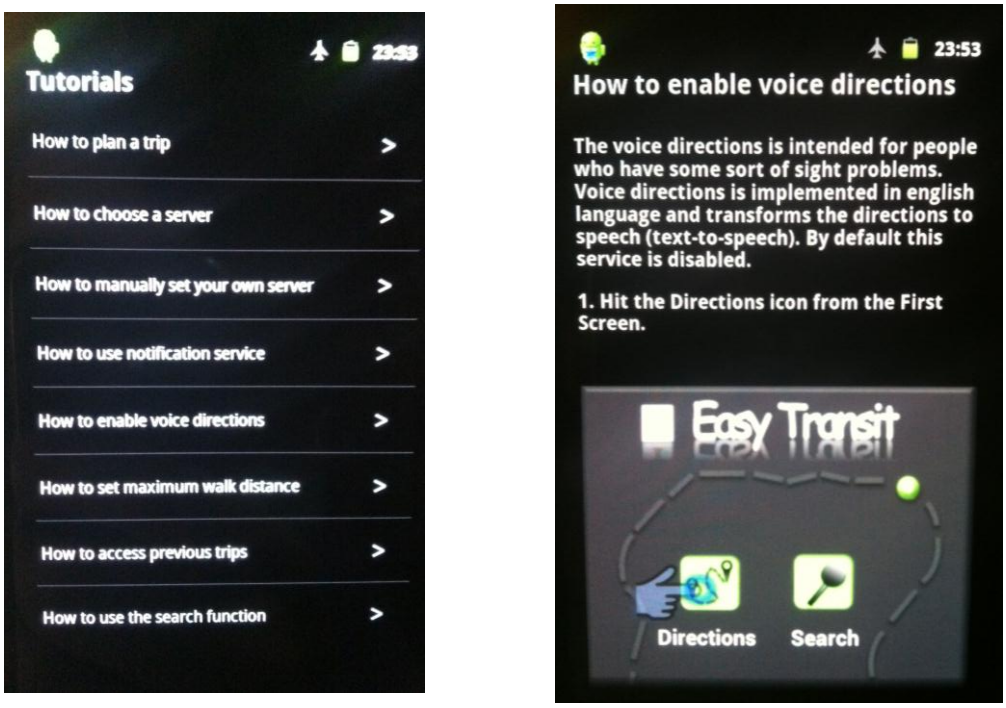
Ο χρήστης έχει την δυνατότητα να αναζητήσει μία στάση/σταθμό βάση του ονόματος του. Αρχικά επιλέγει μία περιοχή από αυτές που του παρέχονται στην λίστα. Προς το παρόν υποστηρίζονται οι περιοχές Portland, Tampa και Κύπρος (Λευκωσία - μικρό δείγμα). Αφού επιλέξει περιοχή, η εφαρμογή επικοινωνεί με τον εξυπηρετητή (με web service) για να ελέγξει κατά πόσο υπάρχουν ανανεωμένα GTFS αρχεία για την συγκεκριμένη περιοχή. Στη συνέχεια ο χρήστης πρέπει να πληκτρολογήσει τουλάχιστον 3 γράμματα για να γίνει η αναζήτηση. Πατώντας είτε το enter από το πληκτρολόγιο της συσκευής είτε το κουμπί της αναζήτησης (πάνω δεξιά) η εφαρμογή επικοινωνεί με τον εξυπηρετητή μέσω ενός web service και ζητά το id στάσεων και το όνομά τους ανάλογα με τα δεδομένα εισόδου του χρήστη. Τα αποτελέσματα με τα ονόματα των στάσεων εμφανίζονται στην ίδια οθόνη υπό την μορφή λίστας. Επιλέγοντας κάποιο από αυτά τα ονόματα γίνεται ξανά κάποια αίτηση στον εξυπηρετητή για να φέρει πληροφορίες σχετικά με την συγκεκριμένη στάση. Οι πληροφορίες αυτές περιλαμβάνουν το όνομα/τα της γραμμής, το είδος του μεταφορικού μέσου και τις ώρες άφιξης και αναχώρησης. Επιπλέον ο χρήστης έχει την δυνατότητα να δει την τοποθεσία μιας στάσης στον χάρτη.



Εικόνα 4.18 Λειτουργία αναζήτησης

4.4.3 Οδηγός χρήσης λειτουργιών

Για την πιο εύκολη και γρήγορη εκμάθηση των διαφόρων λειτουργιών του συστήματος, κρίναμε σωστό να υλοποιήσουμε ένα οδηγό (tutorial). Ο οδηγός αυτός είναι προσβάσιμος επιλέγοντας το κουμπί Tutorials που βρίσκεται στην αρχική οθόνη. Από εκεί μεταφερόμαστε σε νέα οθόνη όπου εμφανίζονται υπό την μορφή λίστας οι διάφοροι οδηγοί χρήσης. Επιλέγοντας ένα από αυτούς τους οδηγούς εμφανίζονται πληροφορίες τόσο σε μορφή κειμένου όσο και σε μορφή εικόνων που αφορούν την συγκεκριμένη λειτουργία.



Εικόνα 4.19 Οδηγός Χρήσης Λειτουργιών

Κεφάλαιο 5 Αξιολόγηση Συστήματος

5.1 Σκοπιμότητα των πειραμάτων 77
5.2 Χρόνος αποκρισιμότητας Συστήματος κατά τον υπολογισμό μιας διαδρομής 77
5.3 Αξιολόγηση Ευχρηστίας 92
5.4 Ενδεικτική σύγκριση με Google Maps 103

5.1 Σκοπιμότητα των πειραμάτων

Για την αξιολόγηση της εφαρμογής αποφασίσαμε να αξιολογήσουμε τα 2 πιο σημαντικά χαρακτηριστικά της εφαρμογής μας που είναι η αποκρισιμότητα ως προς τον σχεδιασμό μιας διαδρομής και η ευχρηστία της εφαρμογής.
Με την αξιολόγηση της εφαρμογής θα προκύψουν συμπεράσματα σχετικά με το πόσο γρήγορη είναι η εφαρμογή και από ποιους παράγοντες εξαρτάται αυτός ο χρόνος.
Η αξιολόγηση της ευχρηστίας της συσκευής θα φανερώσει πόσο ικανοποιημένοι είναι οι χρήστες από τις λειτουργίες που προσφέρει η εφαρμογή, από την διαπροσωπεία της εφαρμογής, από την ευκολία εκμάθησης των διαφόρων λειτουργιών ενώ ταυτόχρονα θα προκύψουν τα διάφορα προβλήματα που πιθανόν να έχει και τυχόν βελτιώσεις που μπορούν να γίνουν.

5.2 Χρόνος αποκρισιμότητας συστήματος κατά τον υπολογισμό μιας διαδρομής

Όταν λέμε χρόνος αποκρισιμότητας συστήματος κατά τον υπολογισμό μιας διαδρομής εννοούμε τον χρόνο που χρειάζεται το σύστημα να υπολογίσει την διαδρομή βάση των προτιμήσεων του χρήστη και να αποστείλει την απάντηση πίσω στον πελάτη. Δεν υπολογίζουμε δηλαδή πόσο χρόνο χρειάζεται να εμφανιστεί η διαδρομή στην κινητή συσκευή. Αυτό που μας ενδιαφέρει είναι πόσο γρήγορα υπολογίζεται η διαδρομή. Ο στόχος μας εδώ είναι να βγάλουμε κάποια συμπεράσματα ανάλογα με το είδος του γράφου (πυκνός, αραιός, μεγάλος, μικρός) σχετικά με το πόσο γρήγορα υπολογίζεται μια διαδρομή. Για αυτό τρέξαμε το πιο κάτω πείραμα για τρεις περιοχές με διαφορετικού μεγέθους γράφο. Περιοχές Portland, Tampa και Cyprus.

Χαρακτηριστικά πειράματος:

Κινητή συσκευή: Android Nexus S

Ram	**512MB**
Display	**Screen size: 4.0 (inches)** **Screen resolution: WVGA (800 x 480)**
Internal Storage	**16Gb**
Wi-Fi	**802.11 b/g/n**
GPS	**Yes**
OS Version	**Android 2.3**
CPU	**1GHz Cortex A8 (Hummingbird) processor**

Εξυπηρετητής: Euclid

Ram	**23.6 GiB**
CPU	**8 processors Intel Xeon 2.83Ghz**
OS	**Ubuntu**

Πίνακας 5.1 Χαρακτηριστικά συσκευών πειράματος

Περιγραφή πειράματος

Για την λήψη μετρήσεων σχετικά με τον χρόνο αποκρισιμότητας **χρησιμοποιήσαμε τρεις περιοχές** όπου το μέγεθος των γράφων των περιοχών αυτών ήταν διαφορετικό . Για κάθε μια από τις περιοχές αυτές **μεταβάλαμε την απόσταση του σημείου εκκίνησης από το σημείο προορισμού.** Κρατούσαμε το σημείο εκκίνησης σταθερό ενώ μεταβάλλαμε το σημείο προορισμού κατά x χιλιόμετρα. **Συγκεκριμένα χρησιμοποιήσαμε 5 διαφορετικές αποστάσεις , 1 km, 5 km, 10 km, 20 km και 30 km. Για κάθε απόσταση τρέξαμε 20 φορές το πείραμα** για να πάρουμε πιο σωστά αποτελέσματα και βγάλαμε τον μέσο όρο και τον διάμεσο (median) των χρόνων αποκρισιμότητας.

Υπολογισμός χρόνου απόκρισης

```
try {…

        long startTime = System.currentTimeMillis();
        //request for trip
        result = Http.get(u).use(client).header("Accept","application/xml").
                 header("Keep-Alive","timeout=60,max=100").
                 charset("UTF-8").followRedirects(true).asString();
        long endTime   = System.currentTimeMillis();
        long responseTime = endTime - startTime;
        respTime=responseTime;
        Log.d("Response Time",responseTime+"");

…}
```

Πριν κάνουμε το http αίτημα στον εξυπηρετητή αποθηκεύουμε την υφιστάμενη ώρα και αφού λάβουμε το αποτέλεσμα από τον εξυπηρετητή βλέπουμε την διαφορά της ώρας. Να σημειώσουμε ότι η τιμή της ώρας που παίρνουμε είναι milliseconds.

Για να είναι σωστό το πείραμά μας πρέπει οι παράμετροι που στέλλουμε στον εξυπηρετητή, πέραν του σημείου προορισμού που μεταβάλλεται, να είναι σταθεροί σε όλες τις περιπτώσεις που εξετάζουμε. Έτσι κρατάμε σταθερό το σημείο εκκίνησης (σε κάθε περιοχή είναι διαφορετικό αλλά καθώς μεταβάλλουμε τις αποστάσεις το κρατάμε σταθερό), την ώρα και ημερομηνία εκκίνησης, το είδος του ταξιδιού (quickest) καθώς και το μεταφορικό μέσο (transit).

Να σημειώσουμε ότι τα πειράματα τα τρέξαμε στον ίδιο χώρο, στον χώρο του εργαστηρίου LINC διότι η ταχύτητα του διαδικτύου παίζει σημαντικό ρόλο αφού κάνουμε http request και λαμβάνουμε http response. Επιπλέον, στον εξυπηρετητή ευκλείδη δημιουργήσαμε ξεχωριστά τους τρεις γράφους και δεν χρησιμοποιήσαμε τους υφιστάμενους εξυπηρετητές του OTP για να ληφθούν οι μετρήσεις κάτω από την ίδια υπολογιστική δύναμη. Τέλος, τα πειράματα φροντίσαμε να γίνουν την ίδια μέρα και σιγουρευτήκαμε ότι η μηχανή του εξυπηρετητή δεν είχε μεγάλο υπολογιστικό φόρτο από άλλες διεργασίες την στιγμή της διεξαγωγής των πειραμάτων που να επηρεάσουν τους χρόνους απόκρισης.

Υπολογισμός σημείου προορισμού:

Για να υπολογίσουμε το σημείο προορισμού χρειαζόμαστε 3 δεδομένα. Το πρώτο είναι οι συντεταγμένες του σημείου εκκίνησης, το δεύτερο είναι η απόσταση που θα έχει το σημείο προορισμού από το σημείο εκκίνησης και το τρίτο είναι η κατεύθυνση – το αζιμούθιο (bearing). Το αζιμούθιο το θέσαμε ως 200 δηλαδή πήραμε την κατεύθυνση βορειοδυτικά του σημείου εκκίνησης. Για αυτό σε κάθε περιοχή τοποθετούσαμε το αρχικό σημείο στο κάτω όριο του χάρτη και δεξιά. Πιο κάτω βλέπουμε την συνάρτηση υπολογισμού του σημείου προορισμού.

```
//function for calculating a geoPoint depending on a startPoint, on the
bearing(azimouth=direction) and on preferred distance

public static GeoPoint FindPointAtDistanceFrom(GeoPoint startPoint, double
initialBearingRadians, double distanceKilometres)
{
    double radiusEarthKilometres = 6371.01;
    double distRatio = distanceKilometres / radiusEarthKilometres;
    double distRatioSine = Math.sin(distRatio);
    double distRatioCosine = Math.cos(distRatio);
```

```
    double startLatRad = DegreesToRadians((startPoint.getLatitudeE6())/1E6);
    double startLonRad = DegreesToRadians((startPoint.getLongitudeE6())/1E6);

    double startLatCos = Math.cos(startLatRad);
    double startLatSin = Math.sin(startLatRad);

    double endLatRads = Math.asin((startLatSin * distRatioCosine) + (startLatCos *
distRatioSine * Math.cos(initialBearingRadians)));

    double endLonRads = startLonRad
        + Math.atan2(
            Math.sin(initialBearingRadians) * distRatioSine * startLatCos,
            distRatioCosine - startLatSin * Math.sin(endLatRads));

    return new
GeoPoint(RadiansToDegrees(endLatRads),RadiansToDegrees(endLonRads));

}
```

Αποτελέσματα:

	Portland	Tampa	Cyprus
Distance(km)	ResponseTime (ms)		
1	240	205	220
1	199	251	167
1	143	192	153
1	208	220	201
1	158	180	199
1	160	201	227
1	188	161	120
1	208	189	160
1	195	173	241
1	148	178	148
1	164	157	251
1	170	175	162
1	190	153	187
1	183	188	157
1	190	182	152
1	211	173	201
1	122	172	160
1	154	173	214
1	137	141	156
1	151	150	206
5	502	216	277
5	545	222	340
5	520	296	220
5	606	362	290

5	485	325	267
5	672	318	241
5	568	280	231
5	532	292	447
5	543	284	329
5	562	285	313
5	506	245	270
5	532	327	276
5	622	223	232
5	562	252	350
5	477	266	260
5	532	333	236
5	564	313	243
5	610	335	357
5	525	305	207
5	668	183	200
10	1596	317	486
10	1237	339	436
10	1190	214	655
10	1314	322	390
10	1152	329	391
10	1292	369	383
10	1141	311	353
10	1250	283	357
10	1163	299	519
10	1075	342	368
10	1126	341	418
10	1172	368	379
10	1272	319	319
10	1143	362	325
10	1187	289	299
10	1198	367	558
10	1540	349	509
10	1422	318	475
10	1337	385	313
10	1607	263	328
20	1529	1354	665
20	1951	1429	476
20	1449	1413	547
20	1467	1477	444
20	1541	1509	550
20	1480	1359	468

20	1408	1603	495
20	1669	1516	498
20	1497	1615	685
20	1537	1773	593
20	1849	1461	631
20	2085	1552	612
20	1577	1454	516
20	1606	1558	497
20	1571	1539	427
20	2179	1477	518
20	1513	1551	577
20	2558	1627	634
20	1544	1514	845
20	1474	1842	798
30	6876	2561	669
30	6893	2381	798
30	6662	2380	719
30	7619	2301	1024
30	7196	2368	743
30	7166	2745	674
30	7472	2552	743
30	7236	2916	650
30	7328	2755	686
30	6914	2733	778
30	6717	2552	666
30	6825	2432	633
30	6952	2325	627
30	6887	2246	625
30	6750	2260	758
30	7000	2697	758
30	6713	2582	328
30	6715	2734	377
30	6981	2628	357
30	6946	2606	440

Πίνακας 5.2 Αποτελέσματα πειράματος

Μέσος όρος

	Porland	Tampa	Cyprus
1	**175.95**	**180.7**	**184.1**
5	**556.65**	**283.1**	**279.3**
10	**1270.7**	**324.3**	**413.05**
20	**1674.2**	**1531.15**	**573.8**
30	**6992.4**	**2537.7**	**652.65**

Πίνακας 5.3 Μέσοι όροι πειραμάτων

Διάμεσος

	Porland	Tampa	Cyprus
1	**176.5**	**176.5**	**177**
5	**544**	**288.5**	**268.5**
10	**1217.5**	**325.5**	**386.5**
20	**1542.5**	**1515**	**548.5**
30	**6930**	**2556.5**	**671.5**

Πίνακας 5.4 Διάμεσοι πειραμάτων

Όπως παρατηρούμε οι μέσοι όροι σε σχέση με τους διάμεσους δεν διαφέρουν κατά πολύ

Θεωρία Γράφων :

Πυκνότητα:

Η πυκνότητα ενός γράφου G = (V,E) μετρά πόσες ακμές βρίσκονται σε ένα σετ ακμών E σε σχέση με τις μέγιστο πιθανό αριθμό ακμών μεταξύ των κόμβων V. Ο μέγιστος αριθμός που μπορεί να προκύψει είναι 1 για πλήρης γράφους (πυκνός γράφος) ενώ το ελάχιστο είναι 0 για άδειους γράφους (αραιός γράφος).

Η πυκνότητα για ένα κατευθυνόμενο γράφο υπολογίζεται ως εξής:

$$\mathbf{Density} = \frac{\mathbf{E}}{\mathbf{V(V-1)}}$$

Μέγεθος:

Το μέγεθος ενός γράφου G είναι ο αριθμός των κόμβων V

Σειρά(Order):

Η σειρά ενός γραφήματος είναι ο αριθμός των ακμών E

-Ελάχιστος αριθμός είναι 0 (άδειος γράφος)

-Μέγιστος αριθμός είναι E(E-1)/2 (πλήρης γράφος)

Στοιχεία γράφων

	Porland	**Tampa**	**Nicosia**
Αριθμός Κόμβων	***455672***	**265327**	**41101**
Αριθμός Ακμών	**1068276**	**577034**	**76592**
Πυκνότητα	**0.00000514493 ($5.14493318 \times 10^{-6}$)**	**0.00000819672 ($8.19672089 \times 10^{-6}$)**	**0.0000453408 ($4.53408022 \times 10^{-5}$)**

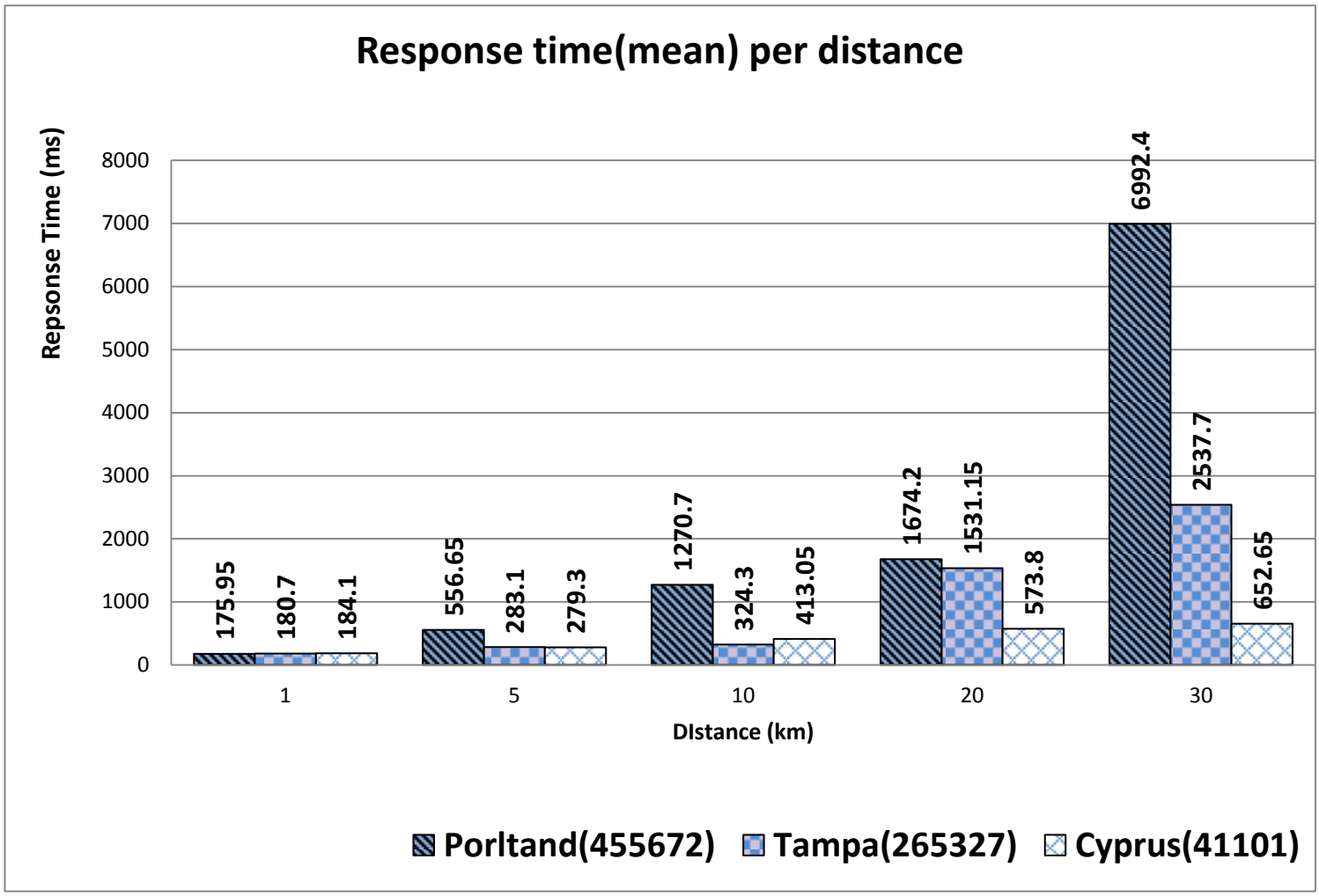
Πίνακας 5.5 Χαρακτηριστικά γράφων

Παρατηρήσεις:

Παρατηρούμε ότι ο μεγαλύτερος γράφος (μέγεθος=αριθμός κόμβων) είναι αυτός της Portland ενώ ακολουθεί αυτός της Tampa και τέλος αυτός της Κύπρου.

Όσο αφορά την πυκνότητα των γράφων η Λευκωσία έχει την μεγαλύτερη πυκνότητα, ακολουθεί ο γράφος της Tampa και τέλος ο γράφος της Portland.

Γραφική Παράσταση



Σχήμα 5.6 Γραφική παράσταση χρόνου απόκρισης σε σχέση με την απόσταση

Η πιο πάνω γραφική παράσταση παρουσιάζει τους μέσους χρόνους απόκρισης ανά απόσταση στις διάφορες περιοχές. Σε κάθε περιοχή αναγράφεται στην παρένθεση το μέγεθος του γράφου. Η Portland παρουσιάζει τον μεγαλύτερο γράφο με αριθμό κόμβων ίσο με 455672, ακολουθεί η Tampa με αριθμό κόμβων 265327 και τέλος η Κύπρος με αριθμό κόμβων 41191.

Στον άξονα Χ παρουσιάζονται οι αποστάσεις σε χιλιόμετρα και στον άξονα Υ βρίσκονται οι χρόνοι απόκρισης σε miliseconds.

Εκ πρώτης όψεως παρατηρούμε ότι καθώς αυξάνουμε την απόσταση από το σημείο εκκίνησης ο χρόνος απόκρισης για όλες τις περιοχές παρουσιάζει αύξηση. Η περιοχή Portland φαίνεται να παρουσιάζει την μεγαλύτερη αύξηση ιδιαίτερα στην απόσταση 30 χλμ στην οποία διαφέρει κατά πολύ από την απόσταση 20 χλμ.

Στην απόσταση 1 χλμ, οι χρόνοι απόκρισης των τριών γράφων κυμαίνονται περίπου στις ίδιες τιμές. Αυτό που θα περίμενε κανείς να δει είναι την Portland να κατέχει το μεγαλύτερο χρόνο απόκρισης διότι έχει και το μεγαλύτερο γράφο. Τα αποτελέσματα όμως δείχνουν ότι ο γράφος της Κύπρου παρόλο που είναι ο πιο μικρός από τις τρεις περιοχές, παρουσιάζει τον μεγαλύτερο χρόνο απόκρισης με τιμή 184 ms.Ακολουθεί ο γράφος της Tampa με τιμή 180.7ms και τέλος ο γράφος της Portland με τιμή 175.95 ms. Λαμβάνοντας υπόψη ότι η απόσταση 1 km είναι μία αρκετά μικρή απόσταση, οι κόμβοι που θα πρέπει να διερευνηθούν από τον αλγόριθμο A* για να οδηγηθούμε στον τελικό προορισμό είναι ελάχιστοι σε σχέση με τους κόμβους που θα διερευνηθούν σε απόσταση 5 ,10,20 και 30 χλμ.

Η πιο πιθανή εξήγηση για αυτό το συμβάν είναι ότι από το σημείο εκκίνησης που επιλέξαμε μέχρι το σημείο προορισμού , οι κόμβοι που έπρεπε να διερευνηθούν ήταν περισσότεροι στην Κύπρο σε σχέση με τις άλλες περιοχές. Αυτό οφείλετε σε τυχαίους παράγοντες, για παράδειγμα η συγκέντρωση των κόμβων στο σημείο εκκίνησης που επιλέξαμε για την Κύπρο ήταν μεγαλύτερη. Επιπλέον, για τις άλλες περιοχές παίρναμε ως σημείο εκκίνησης ένα σημείο στο κάτω όριο του χάρτη και δεξιά που πιθανότατα να μην περιλάμβανε πολλούς κόμβους. Για την Κύπρο όμως, λόγω του ότι είχαμε περιορισμένα δεδομένα GTFS (μόνο 2 διαδρομές) πήραμε ως σημείο εκκίνησης ένα κεντρικό τυχαίο σημείο, λίγο έξω από την Λευκωσία.

Σε απόσταση 5 χλμ, παρατηρούμε ότι υπήρξε άνοδος στον χρόνο απόκρισης και στους τρεις γράφους. Συγκεκριμένα ο γράφος της Portland είχε αύξηση 380.7 ms, ο γράφος της Tampa 102.4 ms και ο γράφος της Κύπρου 95.2 ms.

Σε απόσταση 10 χλμ η διαφορά στον χρόνο απόκρισης αυξάνεται ακόμα περισσότερο. Από αυτό το σημείο αρχίζει να ξεχωρίζει ότι ο γράφος της Portland παρουσιάζει και τον μεγαλύτερο χρόνο απόκρισης με αρκετή διαφορά από τις υπόλοιπες περιοχές. Επιπλέον, σε αυτό το σημείο παρατηρούμε ότι ο γράφος της Κύπρου έχει μεγαλύτερη τιμή απόκρισης από τον γράφο της Tampa και αυτό πιθανότατα να οφείλετε στον ίδιο λόγο που εξηγήσαμε πριν.

Σε απόσταση 20 χλμ, είναι αρκετά αισθητή η διαφορά του χρόνου απόκρισης των γράφων Portland και Tampa σε σύγκριση με τον γράφο της Κύπρου με την διαφορά αυτή να ακουμπά τα 1100.4 ms.

Σε απόσταση 30 χλμ είναι πλέον ξεκάθαρο ότι ο μεγαλύτερος γράφος που παρουσιάζει και τον μεγαλύτερο χρόνο εκτέλεσης είναι αυτός της Portland με την τιμή σχεδόν να τετραπλασιάζεται.

ANOVA Test (Analysis of variance)

Το ANOVA test στην πιο απλή του μορφή προσφέρει ένα στατιστικό έλεγχο του κατά πόσο ή όχι οι μέσοι όροι διαφόρων δειγμάτων είναι οι ίδιοι . Κάνει δηλαδή την ίδια δουλειά με το t-test με την διαφορά ότι δεν περιοριζόμαστε μόνο στις 2 ομάδες δειγμάτων αλλά μπορούμε να χρησιμοποιήσουμε περισσότερες ομάδες. Το ζητούμενο είναι να χρησιμοποιηθούν τα δεδομένα από τα δείγματα ώστε να προκύψουν γενικά συμπεράσματα .

Το ANOVA test δέχεται ως είσοδο 2 δεδομένα. Τα δείγματα από τις διάφορες ομάδες και μία σταθερά α. Το α πρόκειται για το **επίπεδο σημαντικότητας** που είναι η μέγιστη πιθανότητα το αποτέλεσμα να οφείλεται σε σφάλματα ή τυχαίους παράγοντες.

Το α θα το θέσουμε 0.05 εννοώντας ότι αποδεχόμαστε 5% σφάλμα στα συμπεράσματά μας.

Εξαρτημένη Μεταβλητή (μεταβλητή που παρατηρώ): Χρόνος απόκρισης προγραμματισμού διαδρομής.

Ανεξάρτητη μεταβλητή (μεταβλητή που ελέγχω): Μέγεθος γράφου (ως προς τους κόμβους)

Διατύπωση υποθέσεων

Η μηδενική υπόθεση είναι μία πρόταση που προβλέπει ότι η αλλαγή της ανεξάρτητης μεταβλητής δεν έχει καμία επίδραση στην εξαρτημένη μεταβλητή

Μηδενική υπόθεση H0: Το μέγεθος του γράφου δεν επηρεάζει το χρόνο απόκρισης.

μPortland=μTampa=μCyprus

(Η μηδενική υπόθεση δηλώνει ότι οι μέσοι όροι των διαφόρων ομάδων είναι οι ίδιοι.)

Εναλλακτική υπόθεση H1: Το μέγεθος του γράφου επηρεάζει το χρόνο απόκρισης.

μPortland≠μTampa≠μCyprus

Ο σκοπός μας είναι να χρησιμοποιήσουμε το ANOVA test για να δούμε κατά πόσο ισχύει η μηδενική υπόθεση ή όχι.

Στατιστικός δείκτης F

$$\text{στατιστικό ελέγχου} = \frac{\text{αποτελέσμα τα οφείλονται στην επίδραση της ανεξάρτητης μεταβλητής}}{\text{αποτελέσμα τα οφείλονται σε τυχαίους παράγοντες και σφάλματα}}$$

Εάν η τιμή του στατιστικού δείκτη ελέγχου είναι > 1 τότε υποδηλώνει ότι η πιθανότητα λήψης των συγκεκριμένων αποτελεσμάτων είναι μεγαλύτερη από την πιθανότητα αυτά τα δεδομένα να οφείλονται σε τυχαίους παράγοντες.

Κρίσιμη τιμή F

Είναι η τιμή η οποία πρέπει να ξεπεραστεί ούτως ώστε να απορρίψουμε την μηδενική υπόθεση.

Πιο κάτω θα δούμε τα αποτελέσματα του ANOVA test για τις αποστάσεις 1km, 5km,10km,20km και 30km.

Ανάλυση διακύμανσης κατά ένα παράγοντα	**Απόσταση 1km**					
ΣΥΜΠΕΡΑΣΜΑ						
Ομάδες	***Πλήθος***	***Άθροισμα***	***Μέσος όρος***	***Διακύμανση***		
Portland	20	3519	175.95	895.944737		
Cyprus	20	3682	184.1	1257.56842		
Tampa	20	3614	180.7	645.8		
ΑΝΑΛΥΣΗ ΔΙΑΚΥΜΑΝΣΗΣ						
Προέλευση διακύμανσης	***SS***	***Df(βαθμοί ελευθερίας)***	***MS***	***F***	***τιμή-P***	***critical F***
Μεταξύ ομάδων	670.3	2	335.15	0.35917739	0.699819437	3.158842719
Μέσα στις ομάδες	53186.95	57	933.10439			
Σύνολο	53857.25	59				

Πίνακας 5.7 Αποτελέσματα ANOVA test για 1 km

Επεξήγηση:

Παρατηρούμε ότι για απόσταση 1km, τον μεγαλύτερο μέσο όρο στον χρόνο απόκρισης κατέχει η Κύπρος (184.1) ενώ ακολουθεί η περιοχή Tampa (180.7) και τέλος η Portland (175.95).

Ο δεύτερος πίνακας που είναι η ανάλυση διακύμανσης είναι αυτός που μας ενδιαφέρει ιδιαίτερα. Μας ενδιαφέρει η τιμή F και η τιμή F κριτήριο. Από ότι παρατηρούμε η τιμή F είναι πιο μικρή από την τιμή του F κριτήριο. Γι' αυτό το λόγο **αποδεχόμαστε την μηδενική υπόθεση,** ότι δλδ το μέγεθος του γράφου δεν επηρεάζει τον χρόνο απόκρισης.

Το συμπέρασμα που προκύπτει από εδώ είναι ότι τα δεδομένα μας δεν παρέχουν επαρκής αποδείξεις που να στηρίζουν την άποψη ότι το μέγεθος του γράφου επηρεάζει τον χρόνο απόκρισης.

Ανάλυση διακύμανσης κατά ένα παράγοντα	**Απόσταση**	**5km**				
ΣΥΜΠΕΡΑΣΜΑ						
Ομάδες	***Πλήθος***	***Άθροισμα***	***Μέσος όρος***	***Διακύμανση***		
Portland	20	11133	556.65	3000.87105		
Cyprus	20	5586	279.3	3760.64211		
Tampa	20	5662	283.1	2237.98947		
ΑΝΑΛΥΣΗ ΔΙΑΚΥΜΑΝΣΗΣ						
Προέλευση διακύμανσης	***SS***	***Df(βαθμοί ελευθερίας)***	***MS***	***F***	***τιμή-P***	***critical F***
Μεταξύ ομάδων	1011780.433	2	505890.22	168.639392	1.15355E-24	3.158842719
Μέσα στις ομάδες	170990.55	57	2999.8342			
Σύνολο	1182770.983	59				

Πίνακας 5.8 Αποτελέσματα ANOVA test για 5 km

Παρατηρούμε ότι για απόσταση 5km, τον μεγαλύτερο μέσο όρο χρόνου απόκρισης κατέχει η Portland (556.65) με μεγάλη διαφορά από την Tampa (283.1) και την Κύπρο (279.3).

Αυτό που μας ενδιαφέρει είναι η τιμή F σε σχέση με το κριτήριο F. Παρατηρούμε ότι η τιμή F είναι κατά πολύ μεγαλύτερη από το κριτήριο F **επομένως θα απορρίψουμε την μηδενική μας υπόθεση** ότι το μέγεθος του γράφου δεν επηρεάζει τον χρόνο απόκρισης και θα αποδεχτούμε την εναλλακτική υπόθεση που λέει ότι το μέγεθος του γράφου επηρεάζει τον χρόνο απόκρισης με 95% πιθανότητα.

Ανάλυση διακύμανσης κατά ένα παράγοντα	**Απόσταση**	**10km**				
ΣΥΜΠΕΡΑΣΜΑ						
Ομάδες	***Πλήθος***	***Άθροισμα***	***Μέσος όρος***	***Διακύμανση***		
Portland	20	25414	1270.7	24695.9053		
Cyprus	20	8261	413.05	8863.10263		
Tampa	20	6486	324.3	1689.27368		
ΑΝΑΛΥΣΗ ΔΙΑΚΥΜΑΝΣΗΣ						
Προέλευση διακύμανσης	***SS***	***Df(βαθμοί ελευθερίας)***	***MS***	***F***	***τιμή-P***	***critical F***
Μεταξύ ομάδων	10927420.3	2	5463710.2	465.019278	5.05721E-36	3.158842719
Μέσα στις ομάδες	669717.35	57	11749.427			
Σύνολο	11597137.65	59				

Πίνακας 5.9 Αποτελέσματα ANOVA test για 10 km

Παρατηρούμε ότι για απόσταση 10km, τον μεγαλύτερο μέσο όρο χρόνου απόκρισης κατέχει και πάλι η περιοχή Portland (1270.7) με μεγάλη διαφορά από την Tampa (324.3) και την Κύπρο (413.05).

Παρατηρούμε ότι η τιμή F είναι κατά πολύ μεγαλύτερη από το κριτήριο F **επομένως θα απορρίψουμε την μηδενική μας υπόθεση** ότι το μέγεθος του γράφου δεν επηρεάζει τον χρόνο απόκρισης και θα αποδεχτούμε την εναλλακτική υπόθεση που λέει ότι το μέγεθος του γράφου επηρεάζει τον χρόνο απόκρισης.

Ανάλυση διακύμανσης κατά ένα παράγοντα	**Απόσταση**	**20km**				
ΣΥΜΠΕΡΑΣΜΑ						
Ομάδες	***Πλήθος***	***Άθροισμα***	***Μέσος όρος***	***Διακύμανση***		
Portland	20	33484	1674.2	90048.48421		
Cyprus	20	11476	573.8	12525.32632		
Tampa	20	30623	1531.15	14803.92368		
ΑΝΑΛΥΣΗ ΔΙΑΚΥΜΑΝΣΗΣ						
Προέλευση διακύμανσης	***SS***	***Df(βαθμοί ελευθερίας)***	***MS***	***F***	***τιμή-P***	***critical F***
Μεταξύ ομάδων	14319083.23	2	7159541.62	182.9872164	1.55769E-25	3.158842719
Μέσα στις ομάδες	2230176.95	57	39125.9114			
Σύνολο	16549260.18	59				

Πίνακας 5.10 Αποτελέσματα ANOVA test για 20 km

Παρατηρούμε ότι για απόσταση 20km, τον μεγαλύτερο μέσο όρο χρόνου απόκρισης κατέχει πάλι η περιοχή Portland (1674.2) με μεγάλη διαφορά από την Tampa (1531.15) και την Κύπρο (573.8).

Παρατηρούμε ότι η τιμή F είναι κατά πολύ μεγαλύτερη από το κριτήριο F επομένως θα απορρίψουμε την μηδενική μας υπόθεση ότι το μέγεθος του γράφου δεν επηρεάζει τον χρόνο απόκρισης και θα αποδεχτούμε την εναλλακτική υπόθεση που λέει ότι το μέγεθος του γράφου επηρεάζει τον χρόνο απόκρισης με 95% πιθανότητα.

Ανάλυση διακύμανσης κατά ένα παράγοντα	**Απόσταση**	**30km**				
ΣΥΜΠΕΡΑΣΜΑ						
Ομάδες	***Πλήθος***	***Άθροισμα***	***Μέσος όρος***	***Διακύμανση***		
Portland	20	139848	6992.4	70403.41053		
Cyprus	20	13053	652.65	28184.45		
Tampa	20	50754	2537.7	37050.43158		
ΑΝΑΛΥΣΗ ΔΙΑΚΥΜΑΝΣΗΣ						
Προέλευση διακύμανσης	***SS***	***Df(βαθμοί ελευθερίας)***	***MS***	***F***	***τιμή-P***	***critical F***
Μεταξύ ομάδων	423934637.7	2	211967319	4688.218546	5.81219E-64	3.158842719
Μέσα στις ομάδες	2577127.55	57	45212.764			
Σύνολο	426511765.3	59				

Πίνακας 5.11 Αποτελέσματα ANOVA test για 30 km

Παρατηρούμε ότι για απόσταση 30km, τον μεγαλύτερο μέσο όρο χρόνου απόκρισης κατέχει πάλι η περιοχή Portland (6992.4) με μεγάλη διαφορά από την Tampa (2537.7) και ακολουθεί η Κύπρος (573.8).

Παρατηρούμε ότι η τιμή F είναι κατά πολύ μεγαλύτερη από το κριτήριο F επομένως **θα απορρίψουμε την μηδενική μας υπόθεση** ότι το μέγεθος του γράφου δεν επηρεάζει τον χρόνο απόκρισης και θα αποδεχτούμε την εναλλακτική υπόθεση που λέει ότι το μέγεθος του γράφου επηρεάζει τον χρόνο απόκρισης με 95% πιθανότητα.

Συμπεράσματα από το Anova Test:

Από τις παρατηρήσεις που κάναμε στα διάφορα Anova Test μόνο σε μία περίπτωση αποδεκτήκαμε την μηδενική υπόθεση ότι δλδ το μέγεθος του γράφου δεν επηρεάζει τον χρόνο απόκρισης άρα υπάρχει 95% πιθανότητα το συμπέρασμα να οφείλετε σε σφάλματα. Η περίπτωση αυτή αφορούσε την απόσταση 1km και μπορούμε να πούμε ότι το αποτέλεσμα που πήραμε μπορεί να θεωρηθεί ως εξαίρεση. Σε απόσταση 1km, ο αριθμός των κόμβων που διερευνήθηκαν το πιο πιθανόν να ήταν ο ίδιος σε όλες τις περιοχές, και γι' αυτό πήραμε παρόμοιους μέσους όρους. Επίσης, ο χρόνος απόκρισης της Portland και την Tampa ήταν μικρότερος από τον χρόνο απόκρισης της Κύπρου. Αυτό σημαίνει ότι σε 1 km οι κόμβοι που χρειάστηκαν να διερευνηθούν ήταν πιο λίγοι από αυτούς της Κύπρου. Όπως αναφέραμε και πριν κατά τον σχολιασμό της γραφικής παράστασης, πιθανόν από το σημείο εκκίνησης που επιλέξαμε μέχρι το σημείο προορισμού , οι κόμβοι που έπρεπε να διερευνηθούν ήταν

περισσότεροι στην Κύπρο σε σχέση με τις άλλες περιοχές. Αυτό οφείλετε σε τυχαίους παράγοντες, για παράδειγμα η συγκέντρωση των κόμβων στο σημείο εκκίνησης που επιλέξαμε για την Κύπρο ήταν μεγαλύτερη.
Στη συνέχεια, όσο αυξάναμε την απόσταση παρατηρήσαμε ότι απορρίπταμε την μηδενική υπόθεση και άρα βγάζαμε το συμπέρασμα ότι κατά 95% το μέγεθος του γράφου επηρεάζει τον χρόνο απόκρισης κατά τον υπολογισμό μιας διαδρομής.

5.3 Αξιολόγηση Ευχρηστίας

Για να αξιολογήσουμε την ευχρηστία της εφαρμογής που υλοποιήσαμε αποφασίσαμε να ζητήσουμε την άποψη απλών χρηστών οι οποίοι μέσω ερωτηματολογίων εξέφρασαν την γνώμη τους.

5.3.1 Διαδικασία αξιολόγησης

Για να κάνει κανείς αξιολόγηση μιας εφαρμογής πρέπει πρώτα να κατεβάσει την εφαρμογή στο κινητό του τηλέφωνο,να την εγκαταστήσει και να ασχοληθεί μαζί της προσπαθώντας να ανακαλύψει τις διάφορες λειτουργίες που προσφέρει. Το ιδανικό θα ήταν να ανεβάσουμε την εφαρμογή στο Android Market αλλά για ευνόητους λόγους αυτό δεν ήταν δυνατό. Για αυτό αποφασίσαμε να ανεβάσουμε την εφαρμογή σε ένα εξυπηρετητή στον οποίο οποιοσδήποτε θα είχε πρόσβαση.
Το πρόβλημα όμως που δημιουργείται σε τέτοιες περιπτώσεις είναι ότι ο χρήστης πρέπει από μόνος του να ανοίξει τον πλοηγό στην κινητή του συσκευή, να πληκτρολογήσει το URL που του παρέχουμε και έπειτα να κατεβάσει την εφαρμογή. Αυτή όλη η διαδικασία για πολλούς χρήστες θεωρείται χρονοβόρα με αποτέλεσμα πολλοί να προτιμήσουν να μην ασχοληθούν καθόλου. Έτσι, έχοντας υπόψη το πρόβλημα αυτό και με απώτερο σκοπό να πείσουμε τους χρήστες να κατεβάσουν την εφαρμογή χρειάστηκε να επινοήσουμε ένα εύκολο τρόπο με τον οποίο οι χρήστες εύκολα και γρήγορα θα μπορούσαν να κατεβάσουν την εφαρμογή.
Χρησιμοποιήσαμε λοιπόν την μέθοδο QRCode (Quick Response Code) με την οποία μετατρέψαμε την ηλεκτρονική διεύθυνση που οδηγούσε στην εφαρμογή, σε QRCode. Το QRCode πρόκειται για μία μορφή κωδικοποίησης στην οποία μπορούμε να κωδικοποιήσουμε διάφορες πληροφορίες. Είναι μια αρκετά γνωστή μέθοδος λόγω της γρήγορης αναγνωσιμότητάς της και του μεγάλου αποθηκευτικού της χώρου. Ο κώδικας αποτελείται από κάποια μαύρα τμήματα (τετράγωνες τελείες) που είναι διατεταγμένα πάνω σε ένα τετράγωνο μοτίβο και βρίσκονται πάνω σε ένα άσπρο φόντο.

.

Εικόνα 5.12 QR Code για την εφαρμογή EasyTransit

Οι χρήστες των Android συσκευών με την χρήση κάποιου απλού προγράμματος που σαρώνει το QRCode μέσω της κάμερας του τηλεφώνου, πολύ εύκολα μπορούν σαρώσουν την εικόνα και να κατεβάσουν την εφαρμογή με μόνο λίγες απλές κινήσεις.

Για να βρούμε υποψήφιους αξιολογητές της εφαρμογής απευθυνθήκαμε στο κοινωνικό δίκτυο Facebook. Συγκεκριμένα δημοσιεύσαμε το θέμα στην σελίδα του τμήματος Πληροφορικής του πανεπιστημίου Κύπρου και από εκεί άλλα άτομα κοινοποίησαν την δημοσίευση μέσω των δικών τους προφίλ. Στην δημοσίευση εξηγήσαμε τον σκοπό της εφαρμογής και την διαδικασία που έπρεπε να ακολουθήσουν οι συμμετέχοντες στην αξιολόγηση.

Για την καταγραφή των ερωτήσεων στο ερωτηματολόγιο βασιστήκαμε στους 10 κανόνες του Nielsen για την αξιολόγηση ευχρηστίας :

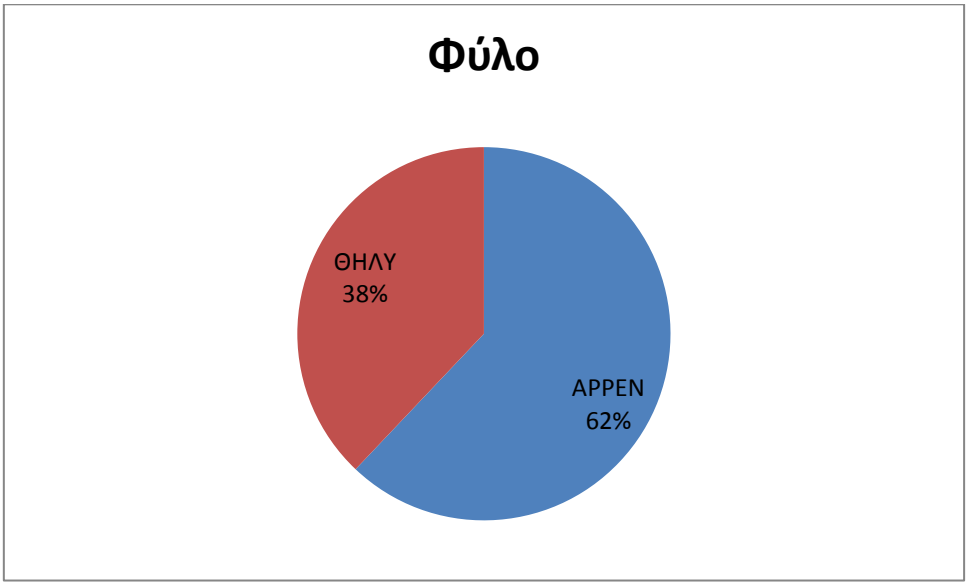
- Ορατότητα της κατάστασης του συστήματος
- Συνοχή (consistency) και συμμόρφωση έναντι προτύπων
- Πρόληψη σφαλμάτων
- Αναγνώριση και όχι ανάκληση (σύγκρινε GUI με command line)
- Προσαρμοστικότητα και αποδοτικότητα χρήσης
- Καλαισθητικός σχεδιασμός (κομψότητα, ικανοποίηση)
- Διευκόλυνση χρηστών ως προς τον εντοπισμό, την διάγνωση και την διόρθωση σφαλμάτων
- Βοήθεια και εγχειρίδια
- Συμφωνία του συστήματος με τον έξω κόσμο
- Έλεγχος και ελευθερία χρήστη

Σύμφωνα λοιπόν με αυτούς τους κανόνες προσπαθήσαμε να δημιουργήσουμε ένα ερωτηματολόγιο το οποίο θα ήταν εύκολο και γρήγορο στην συμπλήρωση. Το ερωτηματολόγιο βρίσκεται στο παράρτημα Β. Να σημειώσουμε ότι το ερωτηματολόγιο δημιουργήθηκε ηλεκτρονικά (http://kwiksurveys.com) ώστε οι χρήστες να το συμπληρώσουν μέσω του διαδικτύου.

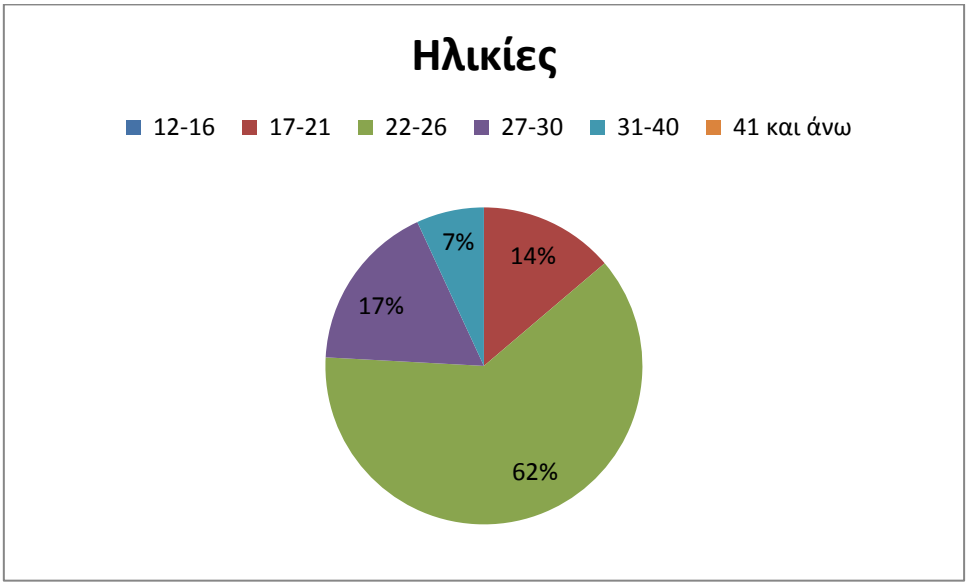
5.3.2 Στατιστικές μετρήσεις

Στην αξιολόγηση της εφαρμογής συμμετείχαν 30 άτομα από τα οποία τα 23 συμπλήρωσαν με επιτυχία το ηλεκτρονικό ερωτηματολόγιο. Τα υπόλοιπα 7 άτομα ξεκίνησαν να συμπληρώνουν το ερωτηματολόγιο αλλά δεν το ολοκλήρωσαν μέχρι το τέλος.

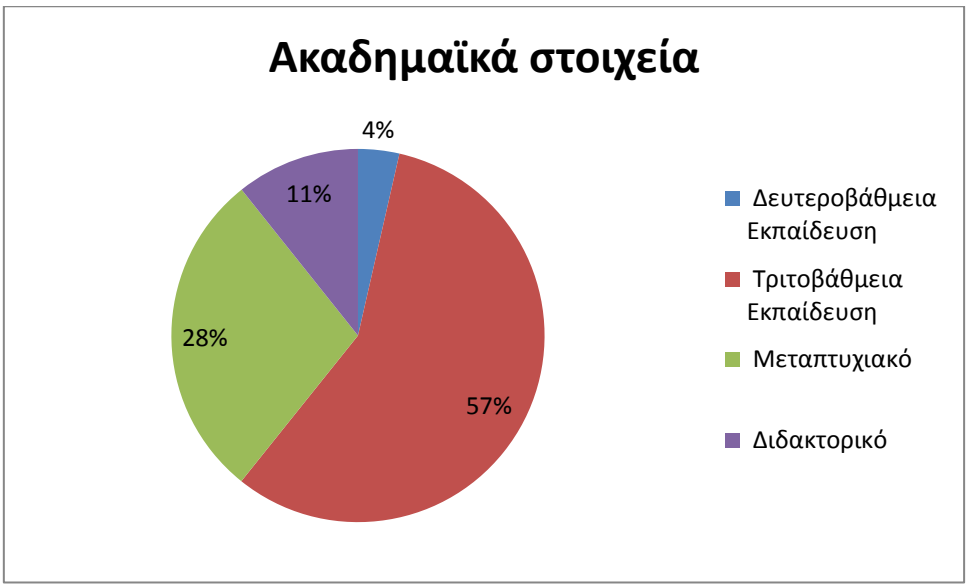
Ερωτήσεις 1 – 4. Δημογραφικά χαρακτηριστικά



Εικόνα 5.13 Στατιστικά στοιχεία φύλου



Εικόνα 5.14 Στατιστικά στοιχεία ηλικίας



Εικόνα 5.15 Στατιστικά στοιχεία ακαδημαϊκής γνώσης

Εικόνα 5.16 Στατιστικά στοιχεία χρηστών android OS

Οι πιο πάνω ερωτήσεις τέθηκαν με σκοπό να κατανοήσουμε το γνωστικό υπόβαθρο των συμμετεχόντων καθώς και κάποια άλλα χαρακτηριστικά που παρουσιάζουν.

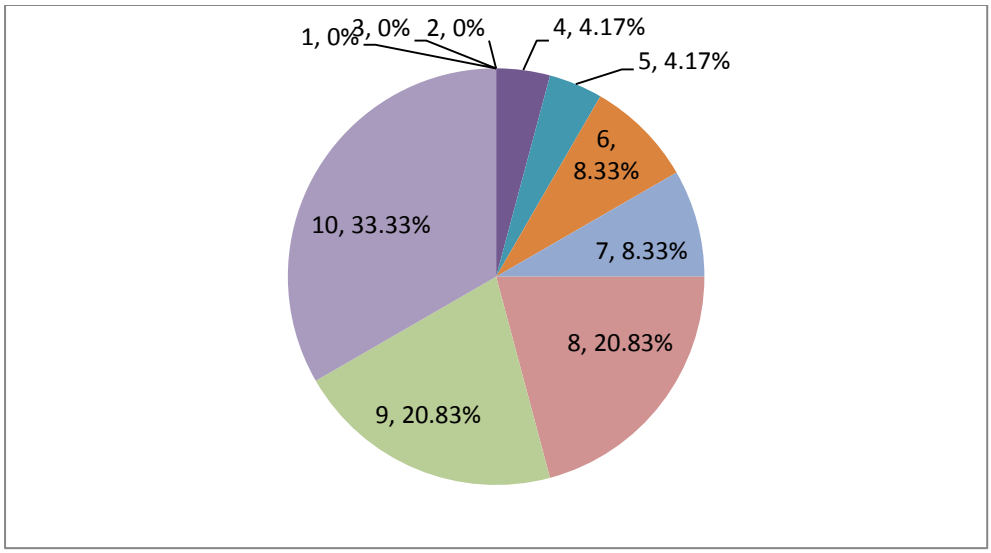
Παρατηρούμε ότι στην έρευνα συμμετείχαν άτομα και από τα 2 φύλα με τους άνδρες να υπερέχουν. Άτομα διαφόρων ηλικιών έχουν συμπληρώσει το ερωτηματολόγιο με το μεγαλύτερο ποσοστό να βρίσκεται μεταξύ 22-26 ενώ δεν παρουσιάστηκε κανένα άτομο ηλικίας 12-16 και 41 και άνω. Όσο αφορά την ακαδημαϊκά στοιχεία των ατόμων αυτών, το μεγαλύτερο ποσοστό που φτάνει στα 60% δήλωσε τριτοβάθμια εκπαίδευση . Ακολουθεί η μεταπτυχιακή εκπαίδευση με ποσοστό 17%, τα άτομα με διδακτορικό με ποσοστό 11% και η

δευτεροβάθμια εκπαίδευση με ποσοστό 4%. Τέλος, στην ερώτηση για την χρήση Android λειτουργικού συστήματος 83% των ατόμων που συμμετείχαν δήλωσαν χρήστες Android ενώ το υπόλοιπο 17% δήλωσε ότι το αντίθετο. Αυτοί που δήλωσαν ότι δεν ήταν χρήστες του Medoid πιθανόν είτε ανήκουν στην κατηγορία των ατόμων που δεν ολοκλήρωσαν το ερωτηματολόγιο είτε δανείστηκαν Android συσκευή από κάποιο άλλο άτομο.

Ερώτηση Σε τι βαθμό θα αξιολογούσατε τα παρακάτω:

(0 αστέρια = μικρός βαθμός, 10 αστέρια= μεγάλος βαθμός)

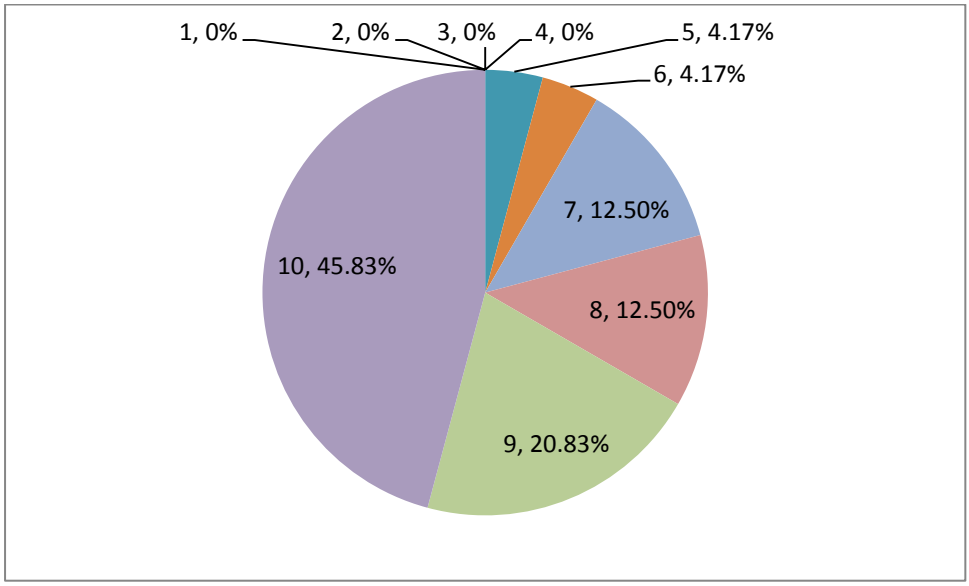
5. Διαπροσωπεία χρήσης (Αλληλεπίδραση με το μενού και τα κουμπιά)



Εικόνα 5.17 Στατιστικά στοιχεία διαπροσωπείας χρήσης

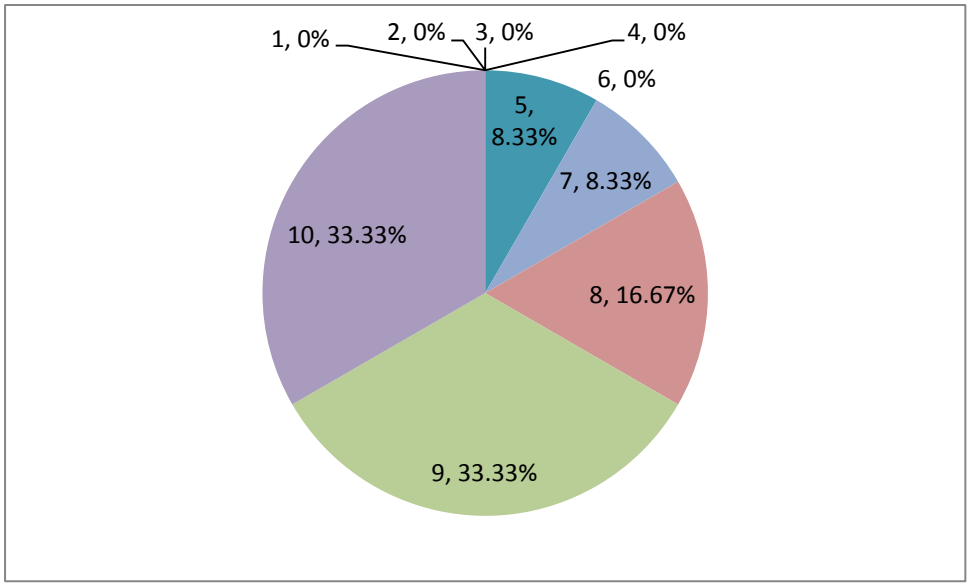
Οι περισσότεροι χρήστες έμειναν ευχαριστημένοι από την διαπροσωπεία χρήσης της εφαρμογής με το μεγαλύτερο ποσοστό των 33.33% να δηλώνει βαθμό 10. Ακολουθούν οι βαθμοί 9 και 8 με ποσοστά 20.83% . Ένα άτομο (4.17%) βαθμολόγησε την διαπροσωπεία με βαθμό 4 που είναι και ο πιο χαμηλός βαθμός που παρουσιάστηκε. Γενικά, τα αποτελέσματα δείχνουν ότι οι χρήστες είναι ευχαριστημένοι με την διαπροσωπεία της εφαρμογής αφού οι πλείστοι βαθμολόγησαν την εφαρμογή με βαθμό πάνω από 5.

6. Ευκολία εκμάθησης λειτουργιών



Εικόνα 5.18 Στατιστικά στοιχεία ευκολίας εκμάθησης λειτουργιών

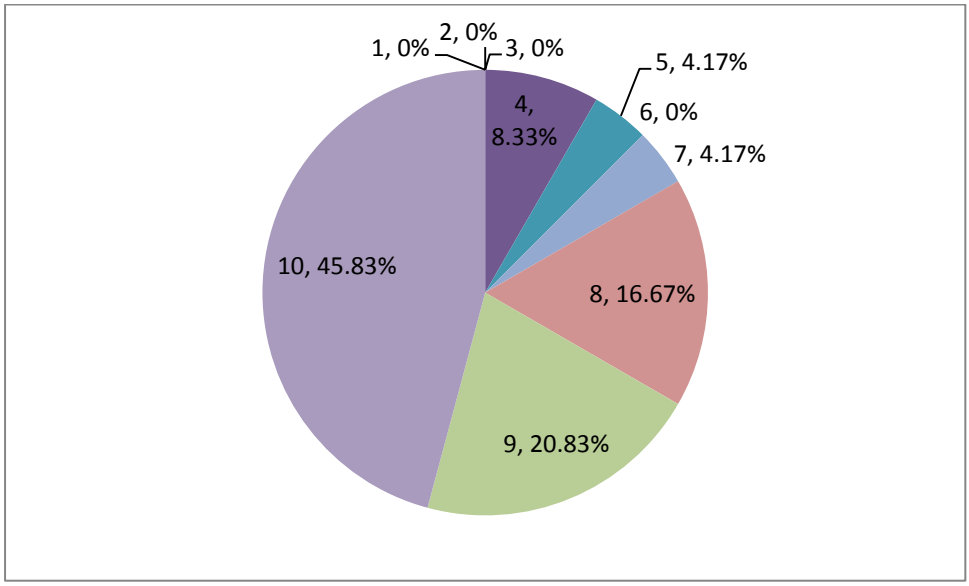
7. Ευκολία χρήσης λειτουργιών



Εικόνα 5.19 Στατιστικά στοιχεία ευκολίας χρήσης λειτουργιών

Τα ποσοστά ευκολίας εκμάθησης λειτουργιών και χρήσης λειτουργιών φανερώνουν ότι τα πλείστα άτομα δεν δυσκολεύτηκαν να μάθουν πώς να χρησιμοποιήσουν τις διάφορες λειτουργίες του συστήματος. Αυτό ίσως να οφείλεται στο γεγονός ότι η εφαρμογή παρείχε οδηγό λειτουργιών από τον οποίο οι χρήστες μπορούσαν να πληροφορηθούν σχετικά με οποιαδήποτε λειτουργία χρειάζονταν να χρησιμοποιήσουν.

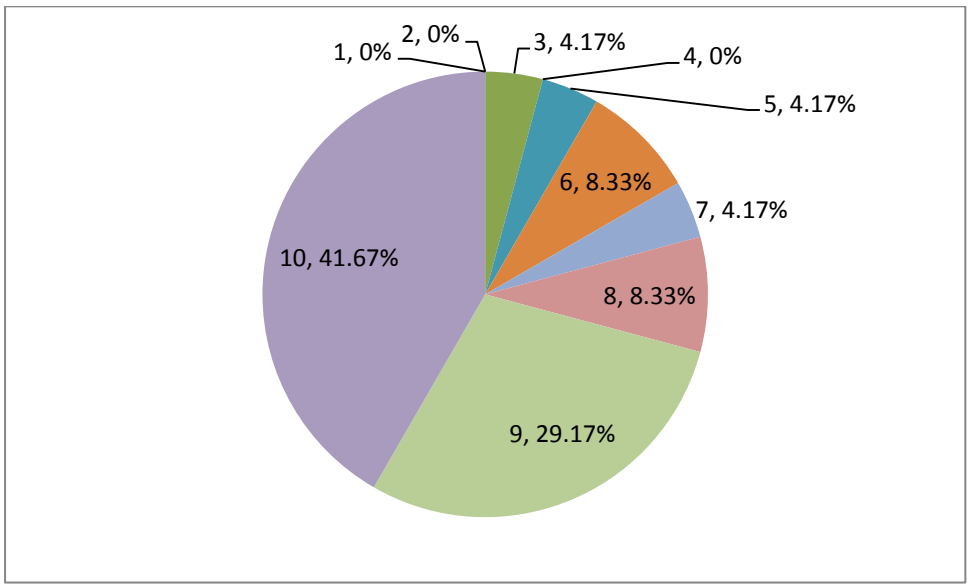
8. Ομαλή μετάβαση μεταξύ των οθονών



Εικόνα 5.20 Στατιστικά στοιχεία ομαλής μετάβασης οθονών

Με την ομαλή μετάβαση μεταξύ των οθονών εννοούμε την πλοήγηση της εφαρμογής σε οθόνες που βρίσκονται κάτω από μια ιεραρχία. Οι συμμετέχοντες δεν φάνηκε να έχουν κάποιο πρόβλημα με αυτό το θέμα καθώς το 82.66% έδωσε βαθμό πάνω από 5.

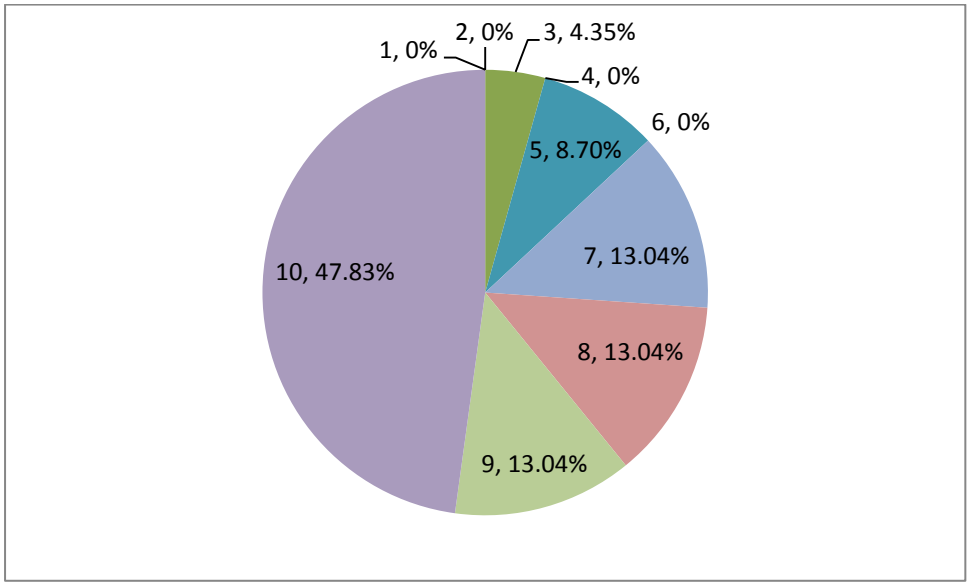
9. Αποκρισιμότητα εφαρμογής



Εικόνα 5.21 Στατιστικά στοιχεία αποκρισιμότητας εφαρμογής

Με τον όρο αποκρισιμότητα εννοούμε το πόσο γρήγορα ανταποκρίνεται η εφαρμογή στις διάφορες ενέργειες που προσπαθεί να εκτελέσει ο χρήστης. Το 41.67% των συμμετεχόντων δήλωσε βαθμό αποκρισιμότητας 10 και το 29.17% δήλωσε βαθμό αποκρισιμότητας 9. Ένα ποσοστό δλδ 70% δήλωσαν ευχαριστημένοι με την αποκρισιμότητα της εφαρμογής. Υπήρχε και μια περίπτωση όπου ένα άτομο δήλωσε βαθμό 3.

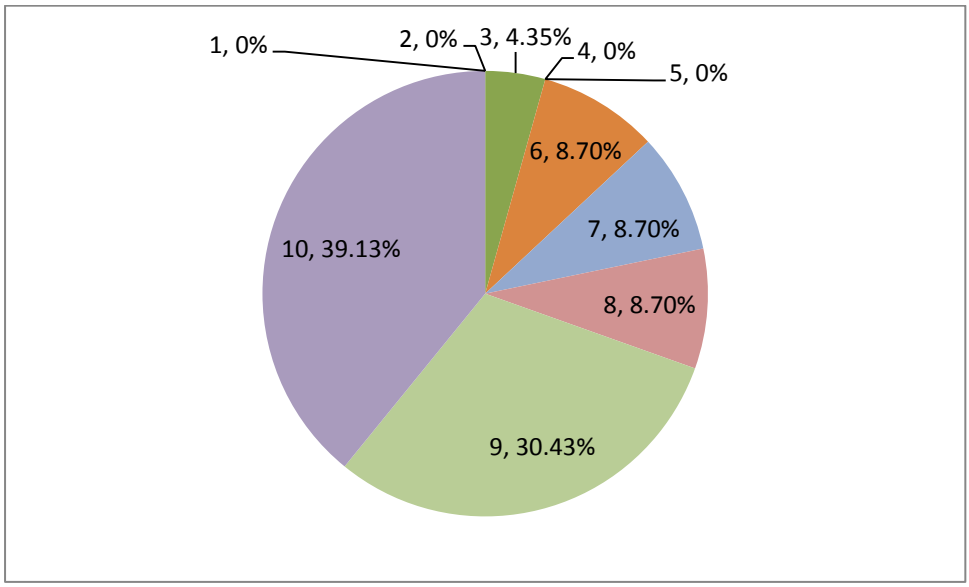
10. Κατανόηση μηνυμάτων που παρουσιάζει η εφαρμογή



Εικόνα 5.22 Στατιστικά στοιχεία κατανόησης μηνυμάτων

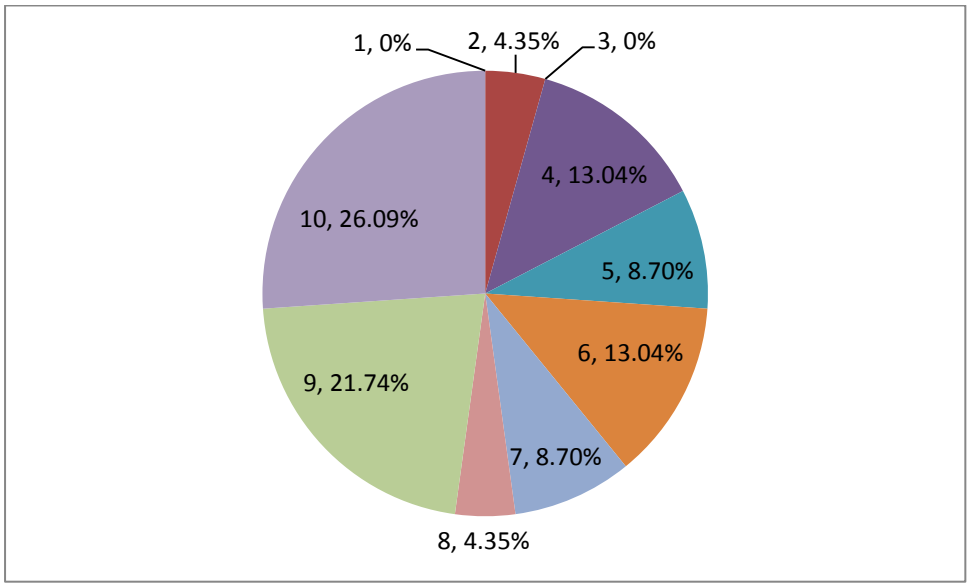
Είναι πολύ σημαντικό για μια εφαρμογή να παρουσιάζει μηνύματα τα οποία να είναι κατανοητά προς τους χρήστες ούτως ώστε οι επόμενες τους ενέργειες να εκτελούνται βάση των οδηγιών αυτών των μηνυμάτων. Για παράδειγμα στην λειτουργία αναζήτησης σταθμού υπάρχει ο περιορισμός να τοποθετηθούν τουλάχιστον 3 γράμματα στο πεδίο αναζήτησης. Η εφαρμογή εμφανίζει ανάλογο μήνυμα. Σκοπός αυτής της αξιολόγησης είναι να δούμε κατά πόσο αυτά τα μηνύματα ήταν κατανοητά προς τους χρήστες. Οι στατιστικές μετρήσεις δείχνουν ότι το υψηλό ποσοστό 47.83%, σχεδόν οι μισοί συμμετέχοντες (11 άτομα περίπου) δήλωσαν βαθμό 10 και άρα τα μηνύματα που παρουσιάζει η εφαρμογή είναι αρκετά κατανοητά. Ένα άτομο δήλωσε βαθμό 3 αλλά γενικά τα πλείστα άτομα επέλεξαν βαθμό πάνω από 5.

11. Σε τι βαθμό σας άρεσε η εφαρμογή



Εικόνα 5.23 Στατιστικά στοιχεία βαθμού αρέσκειας εφαρμογής

12. Σε τι βαθμό θα χρησιμοποιούσατε την εφαρμογή στην καθημερινότητά σας

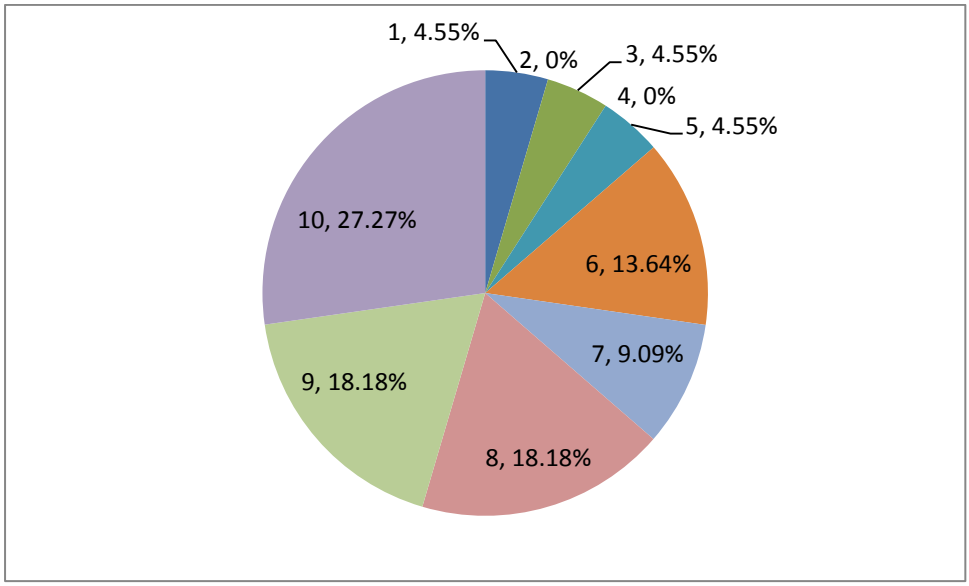


Εικόνα 5.24 Στατιστικά στοιχεία βαθμού χρήσης εφαρμογής

Οι ερωτήσεις 11 και 12 αφορούν την γενική αξιολόγηση της εφαρμογής. Οι ερωτήσεις αυτές στοχεύουν στην μέτρηση της ικανοποίησης του χρήστη. Όσο αφορά την ερώτηση 11, τα ποσοστά που πήραμε ήταν αρκετά ψηλά. Όσο αφορά την ερώτηση 12, οι απαντήσεις που πήραμε ήταν διαμοιρασμένες στου διάφορους βαθμούς σημαντικότητας. Αυτό ίσως να οφείλετε στο γεγονός ότι στην Κύπρο, οι κάτοικοι δεν συνηθίζουν να χρησιμοποιούν τα

δημόσια μέσα μεταφοράς για να μετακινηθούν αλλά έχουν το δικό τους ΙΧ αυτοκίνητο. Επομένως, ο κάθε ένας από τους συμμετέχοντες έβλεπε διαφορετικά την εφαρμογή.

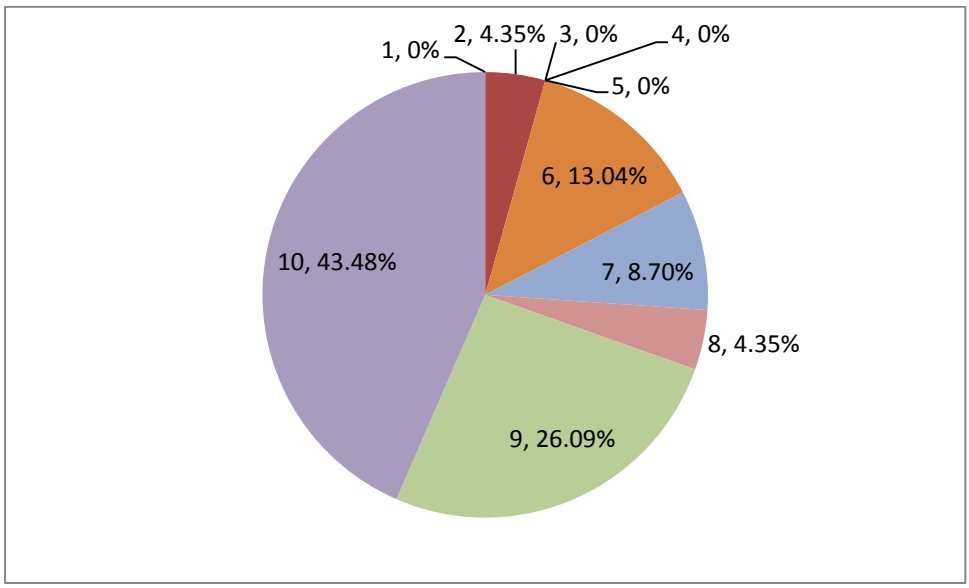
13. Σε τι βαθμό σας ικανοποιεί η εφαρμογή ώστε να εκπληρώσετε τους στόχους σας



Εικόνα 5.25 Στατιστικά στοιχεία βαθμού ικανοποίησης λειτουργιών

Στην ερώτηση αυτή παρουσιάστηκε ένα άτομο που δήλωσε βαθμό 1 που είναι ο πιο χαμηλός βαθμός στην κλίμακα. Το πιο πιθανό αυτό το άτομο να προσπάθησε να σχεδιάσει ένα ταξίδι από το οποίο κανένα μεταφορικό μέσο δεν ήταν διαθέσιμο. Αυτό οφείλετε στο ότι είτε την συγκεκριμένη ώρα δεν υπήρχε διαθέσιμο μεταφορικό μέσο είτε το ταξίδι που προσπάθησε να σχεδιάσει δεν υποστηριζόταν λόγω του μικρού GTFS δείγματος που έχουμε για τις διάφορες στάσεις εδώ στην Κύπρο. Όσο αφορά τα υπόλοιπα άτομα, δήλωσαν ότι μπόρεσαν να χρησιμοποιήσουν την εφαρμογή για να εκπληρώσουν τους στόχους τους.

14. Σε τι βαθμό θα προτείνατε την εφαρμογή σε άλλους



Εικόνα 5.26 Στατιστικά στοιχεία βαθμού πρότασης εφαρμογής σε άλλους

Ένα μεγάλο ποσοστό των συμμετεχόντων απάντησε σε μεγάλο βαθμό ότι θα πρότεινε την εφαρμογή αυτή σε άλλα άτομα.

15. Τι άλλες λειτουργίες θα θέλατε να παρέχει η εφαρμογή;

Η ερώτηση αυτή ζητούσε από τους χρήστες να γράψουν την γνώμη τους σχετικά με το τι άλλες λειτουργίες θα ήθελαν να εκτελεί η εφαρμογή.

Η πιο δημοφιλής απάντηση που πήραμε ήταν να προστεθεί η δυνατότητα για autocomplete δλδ την ώρα που ο χρήστης αναζητά μια περιοχή στον χάρτη, να του εμφανίζονται οι πιθανές περιοχές ούτως ώστε να διευκολύνετε στην επιλογή μιας περιοχής.

Κάποιο άλλο άτομο δήλωσε την επιθυμία να βλέπει την κίνηση που υπάρχει στους δρόμους αλλά αυτό δεν θα αποσκοπούσε σε τίποτα αφού τα μεταφορικά μέσα έχουν σταθερά δρομολόγια (και η εφαρμογή δεν εμφανίζει εναλλακτικές διαδρομές).

Ένα άλλο άτομο φανέρωσε την επιθυμία για σημεία ενδιαφέροντος (POI) ενώ κάποιο άλλο άτομο ζήτησε να του εμφανίζονται οι κοντινότεροι σταθμοί βενζίνης.

16. Τι θα θέλατε να βελτιώσετε στην εφαρμογή;

Στην ερώτηση αυτή κάποια άτομα ανέφεραν ξανά την ανάγκη για autocomplete.

Ένα άλλο άτομο ανάφερε ότι στην αρχική οθόνη τα κουμπιά δεν βρίσκονταν σε σωστή στοίχιση και αυτό οφείλετε στο ότι δεν έγινε έλεγχος για όλα τα μεγέθη οθονών.

Κάποιος άλλος ανέφερε ότι δυσκολεύτηκε να κάνει drag&drop τα σημάδια εκκίνησης και προορισμού στον χάρτη.

17. Τι σας δυσκόλεψε περισσότερο από την εφαρμογή;

Σε αυτή την ερώτηση τα πλείστα άτομα απάντησαν ότι δυσκολεύτηκαν να βρουν διαδρομές οι οποίες χρησιμοποιούν δημόσια μέσα μεταφοράς. Αυτό οφείλετε στο ότι το δίκτυο συγκοινωνιών που δημιουργήσαμε ήταν αρκετά μικρό λόγω του περιορισμού που είχαμε με τα GTFS δεδομένα.

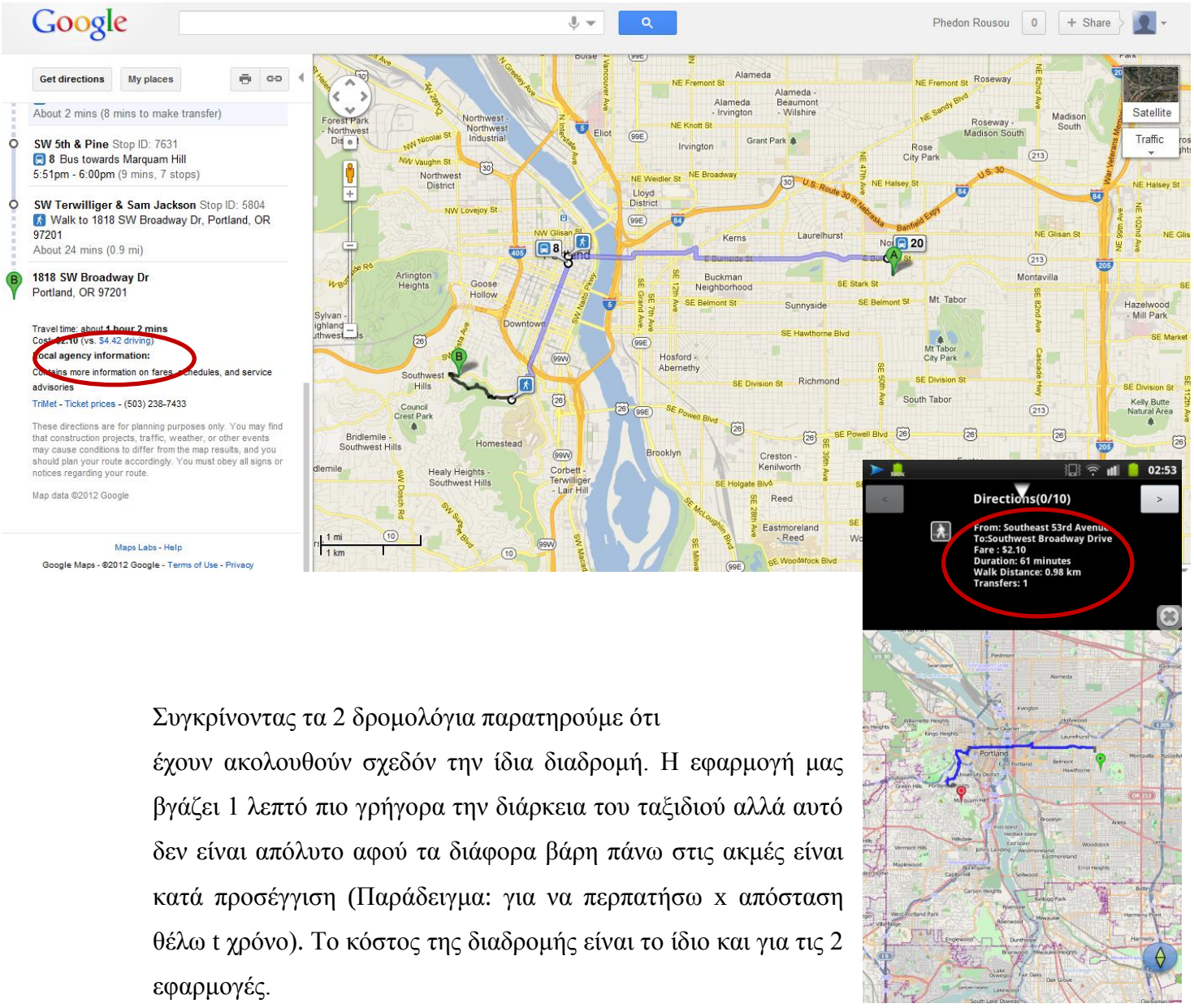
<u>Συμπεράσματα αξιολόγησης ευχρηστίας</u>

Από τις στατιστικές μετρήσεις που πήραμε φάνηκε ότι ένα μεγάλο ποσοστό των συμμετεχόντων έμειναν ικανοποιημένοι από την χρήση της εφαρμογής. Αυτό είναι αρκετά ενθαρρυντικό διότι οι χρήστες κατανόησαν το σκοπό της εφαρμογής, κατάφεραν να την χρησιμοποιήσουν σωστά εκτελώντας τις διάφορες λειτουργίες που προσέφερε και γενικά δεν δυσκόλεψε η χρήση της.

Το γεγονός ότι αρκετοί συμμετέχοντες δήλωσαν ότι δεν κατάφεραν να βρουν κάποια διαδρομή στην οποία γίνεται η χρήση δημόσιου μέσου μεταφοράς δεν οφείλετε σε πρόβλημα της εφαρμογής καθώς έχουμε περιορισμένα δεδομένα όσο αφορά τα δρομολόγια λεωφορείων εδώ στην Κύπρο. *5.4 Ενδεικτική σύγκριση με Google maps*

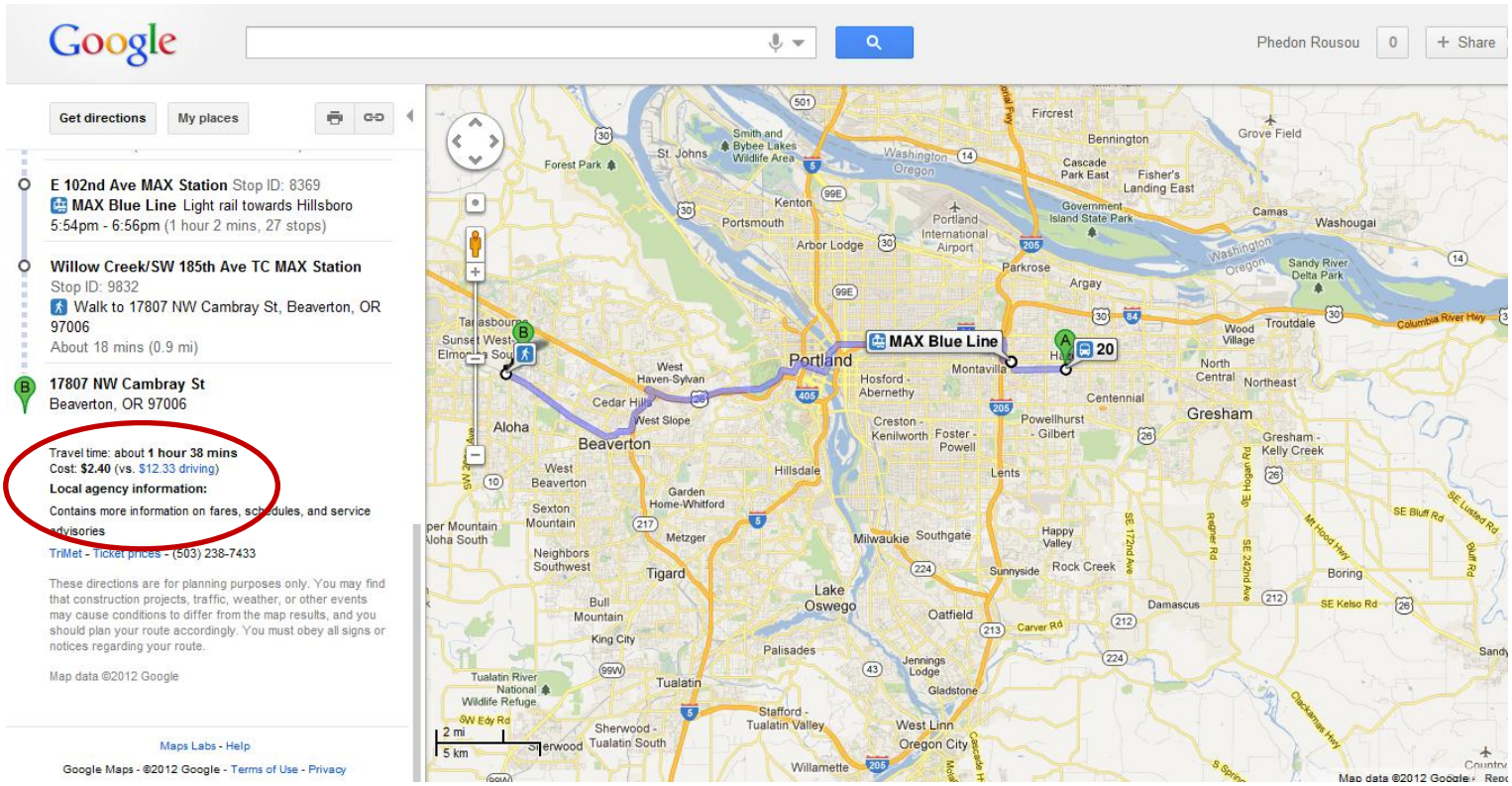
5.4 Ενδεικτική σύγκριση με Google Maps

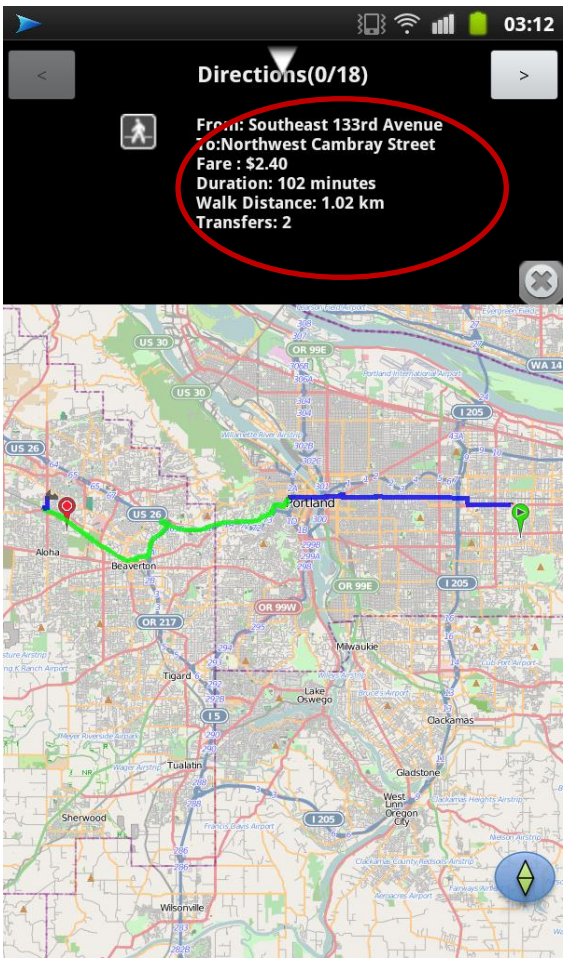
Σε αυτό το υποκεφάλαιο θα κάνουμε μία ενδεικτική σύγκριση των αποτελεσμάτων που παίρνουμε από την εφαρμογή μας με τα αποτελέσματα που παίρνουμε από τους χάρτες της Google (web version). Συγκεκριμένα θα συγκρίνουμε όμοια δρομολόγια στην περιοχή Portland της Αμερικής.



Συγκρίνοντας τα 2 δρομολόγια παρατηρούμε ότι έχουν ακολουθούν σχεδόν την ίδια διαδρομή. Η εφαρμογή μας βγάζει 1 λεπτό πιο γρήγορα την διάρκεια του ταξιδιού αλλά αυτό δεν είναι απόλυτο αφού τα διάφορα βάρη πάνω στις ακμές είναι κατά προσέγγιση (Παράδειγμα: για να περπατήσω x απόσταση θέλω t χρόνο). Το κόστος της διαδρομής είναι το ίδιο και για τις 2 εφαρμογές.

Εικόνα 5.26 Ενδεικτική σύγκριση με google maps





Παρατηρούμε ότι οι 2 διαδρομές ακολουθούν το ίδιο δρομολόγιο. Η εφαρμογή μας παρουσιάζει με διαφορετικά χρώματα το είδος μεταφορικού μέσου ούτως ώστε να είναι ξεκάθαρο και στον χάρτη πέραν της αναπαράστασης με κείμενο. Το κόστος του ταξιδιού είναι το ίδιο ενώ όσο αφορά το χρόνο του ταξιδιού τα google maps βγάζουν 98 λεπτά ενώ η εφαρμογή μας 102 λεπτά. Αυτό όπως είπαμε και πριν εξαρτάται από την εκτίμηση της ώρας για να πάμε περπατώντας από ένα σημείο σε ένα άλλο σημείο.

Εικόνα 5.27 Ενδεικτική σύγκριση με google maps

Κεφάλαιο 6. Συμπεράσματα

6.1 Σύνοψη-Συμπεράσματα 105
6.2 Μελλοντική εργασία 106

6.1 Σύνοψη-Συμπεράσματα

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με τον τομέα δημόσιων συγκοινωνιών και αναπτύξαμε μία εφαρμογή η οποία βοηθά τους χρήστες εύκολα και γρήγορα να σχεδιάσουν διαδρομές με χρήση αποκλειστικά δημόσιων μέσων μεταφοράς.

Αρχικά μιλήσαμε για τα κίνητρα δημιουργίας μιας τέτοιας εφαρμογής ενώ έπειτα θέσαμε τους στόχους της διπλωματικής εργασίας. Αφού μελετήσαμε σχετικές δουλειές που κυκλοφορούν, εντοπίσαμε τα μειονεκτήματα της κάθε εφαρμογής για να τα αποφύγουμε ενώ ταυτόχρονα κοιτάξαμε τις λειτουργίες που προσέφεραν ούτως ώστε να εφαρμόσουμε τέτοιες λειτουργίες και στην εφαρμογή μας.

Στην συνέχεια, αναφερθήκαμε στο Open Trip Planner αναλύοντας την αρχιτεκτονική του και επίσης αναλύσαμε την δομή του γράφου στο OTP.

Στο Κεφ3 αναφερθήκαμε στον αλγόριθμο που χρησιμοποιεί το OTP και στο Κεφ4 αναλύσαμε την αρχιτεκτονική του συστήματος μας και αναφερθήκαμε στις λειτουργίες που υποστηρίζει.

Στο Κεφ.5 περιγράφουμε την μεθοδολογία που ακολουθήσαμε για τα πειράματα και παρουσιάζουμε τα αποτελέσματα που πήραμε ως προς χρόνο απόκρισης κατά τον υπολογισμό της διαδρομής και ως προς την αξιολόγηση της ευχρηστίας του συστήματος.

Από το πείραμα που διατρέξαμε συμπεράναμε ότι ο χρόνος απόκρισης κατά τον υπολογισμό μιας διαδρομής εξαρτάται από το μέγεθος του γράφου (ως προς τον αριθμό των κόμβων). Επιβεβαιώσαμε αυτό το συμπέρασμα με την τεχνική ANOVA test η οποία για τις αποστάσεις 5,10,20 και 30 χλμ έδειξε ότι ο χρόνος απόκρισης κατά 95% οφείλετε στο μέγεθος του γράφου. Για την απόσταση 1 χλμ το ANOVA test απέρριψε αυτή την θεωρία αλλά αυτό οφείλετε στο ότι η απόσταση 1 χλμ είναι αρκετή μικρή και για ένα γράφο δεν θα παίξει και πολύ ρόλο.

Η ανάλυση της ευχρηστίας του συστήματος βασίστηκε πάνω στους 10 κανόνες του Nielsen. Σύμφωνα με αυτούς τους 10 κανόνες δημιουργήσαμε ένα ερωτηματολόγιο το οποίο συμπλήρωσαν διάφορα άτομα που έδειξαν ενδιαφέρον για την εφαρμογή. Τα γενικά συμπεράσματα από την αξιολόγηση είναι ότι το μεγαλύτερο ποσοστό των συμμετεχόντων έμεινε ικανοποιημένο από την εφαρμογή ως προς την ευκολία χρήσης της και ως προς την αποτελεσματικότητά της. Αρκετοί πρότειναν επιπλέον λειτουργίες που θα ήθελαν να παρέχει η εφαρμογή.

6.2 Μελλοντική εργασία

Σε αυτή την διπλωματική εργασία καταφέραμε να φέρουμε εις πέρας όλους τους στόχους που θέσαμε. Πέραν όμως από αυτούς τους στόχους υπάρχουν αρκετά σημεία τα οποία θα μπορούσαμε να προσθέσουμε στο σύστημά μας.

Ως μελλοντική εργασία επιβάλλεται να προσθέσουμε κάποια υπηρεσία στην εφαρμογή που να ενημερώνει άμεσα (πραγματικού χρόνου υπηρεσία) τον χρήστη σχετικά με οποιεσδήποτε αλλαγές προκύπτουν στα χρονοδιαγράμματα των δημόσιων μέσων μεταφοράς. Για παράδειγμα, αυτό που θα μπορούσε να γίνει είναι να διαβάζουμε τα RSS feed διαφόρων εταιρειών δημόσιων μέσων μεταφοράς (εφόσον αυτά προσφέρονται) και να τα προωθούμε μέσω μηνύματος ή ειδοποίησης στην κινητή συσκευή του χρήστη εφόσον ο ίδιος επιθυμεί. Αυτό θα ήταν πολύ χρήσιμο σε περιπτώσεις όπου ο χρήστης σχεδιάζει μία διαδρομή και να ενημερώνεται σχετικά με τυχόν καθυστερήσεις ούτως ώστε να προγραμματίζεται καλύτερα.
Επιπλέον, η εφαρμογή θα μπορούσε να εξελιχθεί ούτως ώστε να παρουσιάζει στον χρήστη points of interest. Αυτό θα ήταν ιδιαίτερα χρήσιμο για άτομα τα οποία επισκέπτονται μία ξένη χώρα ούτως ώστε να συνδυάσουν την μεταφορά με δημόσια μέσα συγκοινωνίας με την επίσκεψη αρχαιολογικών χώρων.
Μία άλλη σημαντική επέκταση του EasyTransit θα ήταν να δίνουμε την δυνατότητα στον χρήστη να επιλέξει ανάμεσα σε διάφορους σχεδιασμούς διαδρομών. Ήδη, το OTP έχει υλοποιημένη την υποδομή για εναλλακτικές διαδρομές αλλά απομένει να εφαρμοστεί στην εφαρμογή.
Όσο αφορά την πλευρά του εξυπηρετητή, ο έλεγχος για την ενημέρωση των GTFS αρχείων στο παρόν στάδιο, γίνεται σε καθημερινή βάση. Θα μπορούσε να υλοποιηθεί κάποιος μηχανισμός ούτως ώστε ανάλογα με το ιστορικό των ενημερώσεων να αποφασίζεται δυναμικά το πότε θα γίνεται ο έλεγχος για ενημέρωση.

Βιβλιογραφία

[1] History of mobile phones, http://en.wikipedia.org/wiki/History_of_mobile_phones

[2] MotorolaDynaTac, http://en.wikipedia.org/wiki/Motorola_DynaTAC

[3] GSM Networks, http://en.wikipedia.org/wiki/GSM

[4] Smartphone Statistics, http://www.guardian.co.uk/technology/appsblog/2011/aug/01/smartphone-stats-2011

[5] Smart Phones, http://en.wikipedia.org/wiki/Smartphone

[6] Android Information, http://developer.android.com/guide/basics/what-is-android.html

[7] Mobile OS Statistics, http://www.email-marketing-reports.com/wireless- mobile/smartphone-statistics.htm

[8] American Public Transportation Association, http://www.apta.com/resources/statistics/Documents/FactBook/APTA_2011_Fact_Book.pdf

[9] Bhaskaran K et al. "The effects of hourly differences in air pollution on the risk of myocardial infarction: case-crossover analysis of the MINAP database", BMJ, 2011

[10] A. Schrijver, "On the history of combinatorial optimization (till 1960)", 2006

[11] Zenker, B. and Ludwig, B. , "ROSE – Assisting Pedestrians to Find Preferred Events and Comfortable Public Transport Connections", ACM Mobility, 2009.

[12] Zenker, B. and Ludwig, B. , "ROSE – An Intelligent Mobile Assistance Discovering Preferred Events and Finding Comfortable Transportation Links", ICAART, 2010.

[13] Ludwig, B., Zenker, B. and Schrader, J., "Recommendation of personalized routes with public transport connections", ACM Mobility, 2009.

[14] Bast, H., Carlsson, E., Eigenwillig, A., Geisberger, R., Harrelson, C., Raychev, V., Viger, F., "Fast Routing in Very Large Public Transportation Networks Using Transfer Patterns", ESA'10 Proceedings of the 18th annual European conference on Algorithms, 2010.

[15] Bader, R., Dees, J., Geisberger, R., Sanders, P., "Alternative Route Graphs in Road Networks", International ICST Conference on Theory and Practice of Algorithms in Computer Systems, 2011.

[16] OTP Project Sites: opentriplanner.org, opentripplanner.com

[17] Shapefiles, http://en.wikipedia.org/wiki/Shapefile

[18] OSM downloads, http://downloads.cloudmade.com/

[19] Open source code https://github.com/openplans/OpenTripPlanner

[20] McHugh, B, "The Open Trip Planner Project", 2011

[21] Γράφος OTP , https://github.com/openplans/OpenTripPlanner/wiki/GraphStructure

[22] Αρχική υλοποίηση OTP σε android OS, https://github.com/marchica/OpenTripPlanner-for-Android

[23] GTFS structure, https://developers.google.com/transit/gtfs/reference

[24] GTFS open source, http://www.gtfs-data-exchange.com/agencies/bylocation

[25] Geomatic Technologies, http://www.geomatic.com.cy/

[26] Osmosis Tool, http://wiki.openstreetmap.org/wiki/Osmosis

[27] PhpMyAdmin Tool http://www.phpmyadmin.net

[28] Encoded Polyline algorithm, https://developers.google.com/maps/documentation/utilities/polylinealgorithm

Παράρτημα Α

Αυτοματοποίηση Δημιουργίας Γράφου

Για να διευκολύνουμε τον administrator σχετικά με την δημιουργία του γράφου γράψαμε ένα Java πρόγραμμα που αυτοματοποιεί αυτή την διαδικασία.
(Σημείωση: Το OTP συνεχώς αναπτύσσεται και μεταβάλλεται γι' αυτό υπάρχει η πιθανότητα το παρακάτω πρόγραμμα να χρειαστεί κάποια τροποποίηση)

Οδηγός Χρήσης

1.Ο administrator πρέπει πρώτα να κατεβάσει την κώδικα του OTP από το διαδίκτυο. Για να το κάνει αυτό πρέπει να ανοίξει ένα terminal και να πληκτρολογήσει την εντολή

git clone git://github.com/openplans/OpenTripPlanner.git

2. Αφού κατεβάσει τον κώδικα θα πρέπει να εγκαταστήσει το maven2 για τις διάφορες εξαρτήσεις του project.

sudo apt-get maven2

3. Στην συνέχεια θα πρέπει να τοποθετήσει το αρχείο CreateGraphAndRunServer.java στο αρχείο git/OpenTripPlanner.

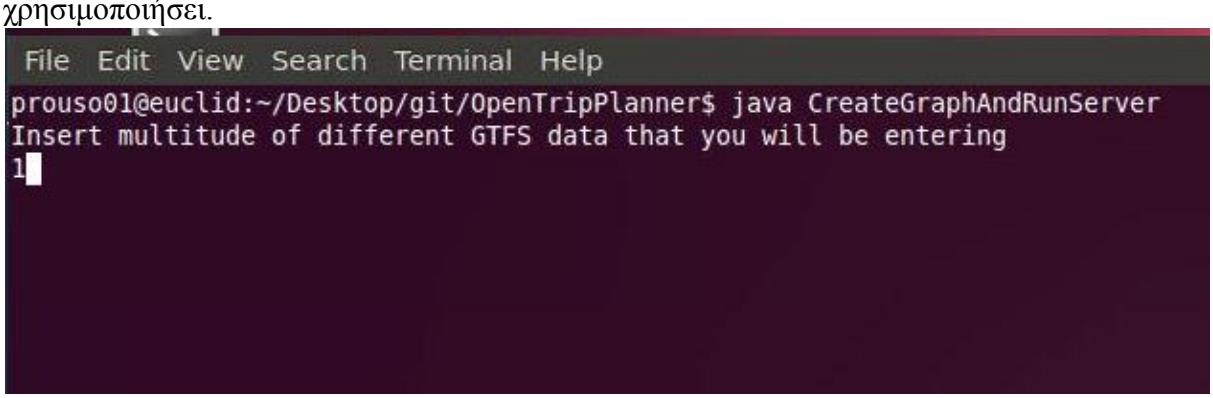
4. Από το terminal θα μεταφερθεί στον φάκελο όπου είναι το git/OpenTripPlanner.και θα τρέξει την εντολή για να δημιουργήσει το .class αρχείο

javac -cp . CreateGraphAndRunServer.java

5. Στη συνέχεια θα τρέξει την εντολή

java CreateGraphAndRunServer

6. Το πρόγραμμα θα του ζητήσει να τοποθετήσει το πλήθος των GTFS δεδομένων που θα χρησιμοποιήσει.



7. Στη συνέχεια το πρόγραμμα θα τον ρωτήσει να επιλέξει εάν τα δεδομένα που θα δώσει θα προέρχονται από το διαδίκτυο ή από το τοπικό σύστημα αρχείων.

8. Έπειτα ζητείται από τον administrator να τοποθετήσει το URL για τα GTFS δεδομένα

9. Ο χρήστης πρέπει να τοποθετήσει το πλήρης μονοπάτι για το αρχείο .osm.

10. Εάν όλα πάνε καλά το πρόγραμμα θα τερματίσει.

11. Αυξάνουμε το java heap space με την ακόλουθη εντολή

export _JAVA_OPTIONS= “-Xmx2g”

12. Στην συνέχεια ο χρήστης το μόνο που απομένει να κάνει είναι να τρέξουμε την εντολή

mvn integration-test -DskipTests -P interactive

Εάν όλα πάνε καλά θα γίνει compile ο γράφος και θα ξεκινήσει ο Tomcat server από όπου μπορεί να γίνει προσβάσιμος στο URL

http://localhost:8080/opentripplanner-webapp

Παράρτημα Β

Στο παράρτημα αυτό παρουσιάζεται το ερωτηματολόγιο το οποίο συμπληρώθηκε από τους χρήστες μέσω του διαδικτύου.

EasyTransit (Phedon Rousou ADE)
ΓΕΝΙΚΕΣ ΠΛΗΡΟΦΟΡΙΕΣ

Σκοπός ερωτηματολογίου:
Το παρόν ερωτηματολόγιο αφορά την αξιολόγηση της demo εφαρμογής EasyTransit που είναι μια εφαρμογή γραμμένη σε Android OS.
Ο σκοπός του ερωτηματολογίου αυτού είναι η λήψη μετρήσεων σχετικά με την ευχρηστία, την ικανοποίηση του χρήστη και την λειτουργικότητα που προσφέρει η εφαρμογή. Είναι εντελώς ανώνυμο και οι στατιστικές μετρήσεις που θα προκύψουν θα χρησιμοποιηθούν για την καταγραφή πορισμάτων.

Η εφαρμογή
Πρόκειτε για μια διαδραστική εφαρμογή που αφορά την δρομολόγηση διαδρομών μέσω δημόσιων μέσων μεταφοράς με την οποία ο χρήστης μπορεί να αλληλεπιδράσει ούτως ώστε:

- Να προσχεδιάσει την μετακίνησή του από ένα σημείο σε άλλο χρησιμοποιώντας αποκλειστικά αστικές συγκοινωνίες
- Να δει πληροφορίες σχετικά με την διαδρομή του όπως απόσταση περπατήματος, αριθμός εναλλαγών στάσεων, κόστος ταξιδιού καθώς και βήμα προς βήμα οδηγίες οπτικά(χάρτης) και ακουστικά μέχρι τον προορισμό.
- Να ειδοποιηθεί από πριν με notifications εάν πρέπει να κατεβεί σε μια στάση(η ενημέρωση γίνεται στην ακριβώς προηγούμενη στάση)
- Να κάνει αναζήτηση στάσεων σχετκά με ώρες άφιξης και αναχώρησης

Προς το παρόν η εφαρμογή είναι διαθέσιμη για τις περιοχές Λευκωσία(δεδομένα από την εταιρεία Geomatic Technologies http://www.geomatic.com.cy/), Portland και Tampa.

Scan with RedLaser

Σκανάρετε την παραπάνω εικόνα με το τηλεφωνό σας για να κατεβάσετε αυτόματα την εφαρμογή.

Η εφαρμογή μπορεί να βρεθεί για κατέβασμα
στο http://euclid.grid.ucy.ac.cy/fedon/easyTransit/easyTransit.apk

Για οποιοδήποτε πρόβλημα επικοινωνήστε μαζι μου (Φαίδων Ρούσου) στο prouso01@cs.ucy.ac.cy

Δημογραφικά στοιχεία

1. Φύλο:

- ΑΡΡΕΝ
- ΘΗΛΥ

Reset

2. Ηλικία:

- 12-16
- 17-21
- 22-26
- 27-30
- 31-40
- 41 και άνω

Reset

3. Ακαδημαϊκά στοιχεία:

- Δευτεροβάθμεια Εκπαίδευση
- Τριτοβάθμεια Εκπαίδευση
- Μεταπτυχιακό
- Διαδκτορικό

Reset

4. Χρήστης Android λειτουργικού συστήματος

- Ναι
- Όχι

Reset

Σχετικά με την εφαρμογή

(0 αστέρια = μικρός βαθμός, 10 αστέρια=μεγάλος βαθμός)

Σε τι βαθμό θα αξιολογούσατε τα παρακάτω:

5. Διαπροσωπεία χρήσης(Αλληλεπίδραση με το μενού και τα κουμπιά)

6. Ευκολία εκμάθησης λειτουργιών

7. Ευκολία χρήσης λειτουργιών

8. Ομαλή μετάβαση μεταξύ των οθονών

9. Αποκρισιμότητα της εφαρμογής (πόσο γρήγορη είναι)

10. Κατανόηση των μηνυμάτων που παρουσιάζει η εφαρμογή

11. Σε τι βαθμό σας άρεσε η εφαρμογή;

12. Σε τι βαθμό θα χρησιμοποιούσατε την εφαρμογή στην καθημερινότητά σας;

13. Σε τι βαθμό σας ικανοποιεί η εφαρμογή ώστε να εκπληρώσετε τους στόχους σας;

14. Σε τι βαθμό θα προτείνατε την χρήση της εφαρμογής σε άλλους;

15. Τι άλλες λειτουργίες θα θέλατε να παρέχει η εφαρμογή;

16. Τι θα θέλατε να βελτιώσετε στην εφαρμογή;

17. Τι σας δυσκόλεψε περισσότερο από την εφαρμογή;