Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Υλοποίηση και μελέτη εφαρμογών για δρομολόγηση με μέσα πολλαπλής τροχιάς (Multimodal routing)

Διπλωματικη Εργασια

της

Μαρίας Ράλλη

Επιβλέπων: Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Εργαστηριο Τεχνολογιας Λογισμικου

Αθήνα, Ιούνιος 2015

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Τεχνολογίας Λογισμικού

Υλοποίηση και μελέτη εφαρμογών για δρομολόγηση με μέσα πολλαπλής τροχιάς (Multimodal routing)

Διπλωματικη Εργασια

της

Μαρίας Ράλλη

Επιβλέπων: Ιωάννης Βασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Ιουνίου, 2015.

.....	.....	.....
Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης	Ιωάννης Σταύρακας
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Ερευνητής ΙΠΣΥ

Αθήνα, Ιούνιος 2015

.....
Μαρια Ραλλη
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια, έχει αυξηθεί το ενδιαφέρον για την υλοποίηση εφαρμογών δρομολόγησης με Μέσα Μαζικής Μεταφοράς. Οι συγκεκριμένες εφαρμογές είναι ιδιαίτερα χρήσιμες τόσο για τους κατοίκους μιας πόλης αλλά και για τους τουρίστες που την επισκέπτονται. Οι εφαρμογές δέχονται ως είσοδο τα δρομολόγια των Μέσων Μαζικής Μεταφοράς και το οδικό δίκτυο της περιοχής ενδιαφέροντος, τα μοντελοποιούν κατάλληλα και εκτελούν πάνω στο παραγόμενο μοντέλο κατάλληλους αλγόριθμους δρομολόγησης. Οι περισσότερες από αυτές τις εφαρμογές είναι διαδικτυακές προκειμένου να διευκολύνουν την ευκολότερη πρόσβαση στους τελικούς χρήστες (π.χ. τουρίστες).

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη και ανάπτυξη τεχνολογιών και εργαλείων που αφορούν την αυτόματη ανάλυση και δρομολόγηση σε δίκτυα Μέσων Μαζικής Μεταφοράς. Στα πλαίσια της παρούσας διπλωματικής μελετήθηκαν υπάρχουσες διαδικτυακές εφαρμογές εύρεσης της βέλτιστης διαδρομής σε δίκτυα πολλαπλών Μέσων και αναπτύχθηκαν εργαλεία τόσο για τον έλεγχο της δρομολόγησης που πραγματοποιείται από τις συγκεκριμένες εφαρμογές, τη συγκριτική αξιολόγηση τους (benchmarking), τη μελέτη της απόδοσής και της κλιμάκωσής τους σε αυξανόμενο φόρτο και μεγαλύτερους αριθμούς χρηστών, καθώς και για την ανάλυση των ίδιων των δικτύων πολλαπλών Μέσων.

Λέξεις Κλειδιά

Αλγόριθμοι δρομολόγησης, Πολλαπλή τροχιά, Σχεδιασμός ταξιδιών, Γράφοι οδικών δικτύων, Οπτικοποίηση στατιστικών

Abstract

In the last few years, recent research focused on the development of routing applications operating on public transportation networks. This type of applications are very useful for a variety of users, including not only the citizens of a city but also the visiting tourists. The input data needed for such an application is the public transportation network data and the underlying road network of the area of interest. Then the routing application models this data properly, so that suitable routing algorithms may operate on the created model-representation. Most of those multimodal, routing applications are web-based to facilitate easier access and faster adoption from end users (e.g. tourists).

The main objective of this thesis is to study and develop novel technologies and tools for the automated analysis and routing of public transportation networks. In this thesis, we focused on an existing open-source, web-application for multimodal routing and we developed novel tools and applications that assess, evaluate, compare and benchmark the routing results performed by this application, as well as stress-testing its performance and scalability. Moreover, we have created an online application tool for easier analysis and visualization of the public transportation networks used in our evaluation.

Keywords

Routing algorithms, Multimodal, Trip planning, Road network graphs, Statistics visualization

Ευχαριστίες

Θα ήθελα να πω ένα μεγάλο ευχαριστώ στον κύριο Ιωάννη Βασιλείου για την διαρκή υποστήριξη και πολύτιμη καθοδήγηση, η οποία συνέβαλε καθοριστικά στη διαμόρφωση αυτής της διπλωματικής εργασίας, καθώς και για την εμπιστοσύνη και σεβασμό τον οποίο μου έδειξε. Επίσης, θα ήθελα να ευχαριστήσω τον Αλέξανδρο Εφεντάκη για την πολύ σημαντική βοήθεια την οποία μου προσέφερε.

Χρωστάω ένα μεγάλο ευχαριστώ στους γονείς μου για την υποστήριξη που μου έχουν προσφέρει και την εμπιστοσύνη που έδειξαν σε κάθε επιλογή μου.

Τέλος, θέλω να πω ένα μεγάλο ευχαριστώ σε όλους τους φίλους μου που μου στάθηκαν τα τελευταία χρόνια.

Μαρία Ράλλη

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κατάλογος παραθέσεων

Κεφάλαιο 1

Εισαγωγή

Τις τελευταίες δύο δεκαετίες υπάρχει σημαντικό ερευνητικό ενδιαφέρον που εστιάζει στην ανάπτυξη εφαρμογών που αφορούν τα συστήματα πλοήγησης. Τέτοιες εφαρμογές ευνοούν τόσο τους κατοίκους μιας πόλης για την οποία έχουν αναπτυχθεί όσο και τους τουρίστες που μπορεί να επισκεφθούν μια άγνωστη πόλη. Αρχικά το ενδιαφέρον της ερευνητικής κοινότητας επικεντρώθηκε στην υλοποίηση αποτελεσματικών αλγόριθμων για το πρόβλημα της εύρεσης της συντομότερης διαδρομής μεταξύ δύο σημείων σε οδικά δίκτυα (η δημοσίευση [28] παρουσιάζει την τελευταία σχετική επισκόπηση). Παράλληλα πρόσφατες ερευνητικές δουλειές εστίασαν στη βελτίωση προηγούμενων αλγόριθμων δρομολόγησης στα πλαίσια δυναμικών οδικών δικτύων (όπως η [33] που βελτιστοποίησε το σημαντικό αλγόριθμο ALT [37] για δυναμικά οδικά δίκτυα) ή επεκτάθηκαν σε καινούρια ενδιαφέροντα προβλήματα, όπως ο διαμοιρασμός της προπεξεργασίας των οδικών δικτύων σε πολλούς web-clients [36], στην επίλυση πιο σύνθετων ερωτημάτων ένας-προς-όλους, εύρους και ένας-προς-πολλούς [34], στα ερωτήματα k-πλησιεστέρων γειτόνων [35] καθώς και στη δημιουργία ολοκληρωμένων συστημάτων [31] και εφαρμογών [32] που καλύπτουν όλο των κύκλο ζωής GPS δεδομένων κίνησης για την εκτίμηση του κυκλοφορικού φόρτου και την υλοποίηση αλγόριθμων δρομολόγησης που χρησιμοποιούν τα ζωντανά δεδομένα κίνησης για τον υπολογισμό της συντομότερης διαδρομής ή ισοχρονικών καμπύλων.

Στην περίπτωση που θέλουμε να βρούμε τη βέλτιστη διαδρομή χρησιμοποιώντας Μέσα Μαζικής Μεταφοράς (ΜΜΜ), το πρόβλημα γίνεται σημαντικά πιο πολύπλοκο, όχι μόνο γιατί μπορεί να έχουμε πολλαπλά κριτήρια για τον ορισμό της βέλτιστης διαδρομής (λιγότερες μετεπιβιβάσεις, συντομότερος χρόνος, φθηνότερο εισιτήριο ή μικρότερη διαδρομή περπατήματος), αλλά και γιατί στα δίκτυα ΜΜΜ "*εκτός από τη γεωγραφική πληροφορία έχουμε και να λάβουμε υπόψη μας τα δρομολόγια των μέσων*" [27]. Συνεπώς, παρόλο που η μοντελοποίηση ενός τέτοιου δικτύου ως γράφου είναι εφικτή, ο παραγόμενος γράφος είναι σημαντικά πιο πολύπλοκος, "*αφού οι κόμβοι του παραγόμενου γράφου αναφέρονται στην άφιξη ή αναχώρηση ενός μέσου από το σταθμό ενώ οι ακμές αντιπροσωπεύουν χρόνους αναμονής ή χρόνος μετάβασης ενός μέσου μεταξύ δύο σταθμών*" [27]. Επιπλέον, η δυσκολία στην ανάπτυξη εφαρμογών δρομολόγησης με Μέσα Μαζικής Μεταφοράς οφείλεται όχι μόνο στους πολύπλοκους αλγορίθμους εύρεσης της βέλτιστης διαδρομής, αλλά και στην έλλειψη δημόσιων δεδομένων ΜΜΜ από αντίστοιχους φορείς. Στα πλαίσια της παρούσας διπλωματικής θα μελετήσουμε ένα σημαντικό εργαλείο ανοικτού κώδικα για δρομολόγηση με ΜΜΜ και θα αναπτύξουμε επιπλέον συμπληρωματικά εργαλεία που θα μελετούν την απόδοσή και κλιμάκωση του για διαφορετικές πόλεις, αλλά θα αναπτύξουμε και μια επιπλέον διαδικτυακή εφαρμογή που θα οπτικοποιεί τα στατιστικά χαρακτηριστικά των δικτύων ΜΜΜ που χρησιμοποιήσαμε.

1.1 Αντικείμενο της διπλωματικής

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη και ανάπτυξη τεχνολογιών και εργαλείων που αφορούν την αυτόματη ανάλυση και και δρομολόγηση σε δίκτυα Μέσων Μαζικής Μεταφοράς. Συγκεκριμένα, μελετήθηκε και εκτελέστηκε μια εφαρμογή ανοικτού κώδικα για την δρομολόγηση σε δίκτυα πολλαπλών μέσων και αναπτύχθηκαν εργαλεία για τόσο για την μελέτη της ίδιας της εφαρμογής όσο και για τα δίκτυα πολλαπλών μέσων που δέχεται ως είσοδο.

Η εφαρμογή που μελετήθηκε είναι το OpenTripPlanner [19] το οποίο αποτελεί την πιο διαδεδομένη εφαρμογή ανοικτού κώδικα για την δρομολόγηση με πολλαπλά Μέσα και βασίζεται στο πρότυπο GTFS [8] και την υπηρεσία OpenStreetMaps [39]. Αρχικά, μελετήθηκε η αρχιτεκτονική της εφαρμογής. Συγκεκριμένα, αναλύθηκαν οι αλγόριθμοι δρομολόγησης που χρησιμοποιεί καθώς και ο τρόπος μοντελοποίησης των δεδομένων του δικτύου πολλαπλών Μέσων. Καθώς η εφαρμογή OpenTripPlanner είναι διαδικτυακή μελετήθηκαν τα συστατικά της εφαρμογής, τα δεδομένα που δέχεται ως είσοδο, τα αποτελέσματα που παράγει καθώς και η διαδικασία που πρέπει να ακολουθηθεί για την εκτέλεσή της.

Επιλέχθηκε η εκτέλεση της εφαρμογής OpenTripPlanner για τη δρομολόγηση σε τρεις πόλεις την Αθήνα, το Παρίσι και το Πόρτλαντ. Καθώς η εφαρμογή δέχεται το δεδομένα των δικτύων Μέσων Μαζικής Μεταφοράς σε ένα καθορισμένο πρότυπο της Google (GTFS) φροντίσαμε να λάβουμε τα δεδομένα αυτά από τους κατάλληλους φορείς. Στη συνέχεια, έγινε εισαγωγή των δεδομένων αυτών σε μια βάση δεδομένων για τον έλεγχό τους και διόρθωση πιθανών σφαλμάτων. Έπειτα, πραγματοποιήθηκε παραμετροποίηση και εκτέλεση της εφαρμογής OpenTripPlanner για τις επιλεγμένες πόλεις σε ένα μηχάνημα που στεγάζεται στην υπηρεσία του Okeanos [15].

Αφού εκτελέστηκε η εφαρμογή και κατέστη δυνατή η σχεδίαση ταξιδιών στην Αθήνα, το Παρίσι και το Πόρτλαντ μέσω της γραφικής διεπαφής χρήστη παρουσιάστηκε η ανάγκη εκτέλεσης πειραμάτων σε αυτή. Η εκτέλεση πειραμάτων στην εφαρμογή ήταν σημαντική τόσο για τον έλεγχο της ορθότητας των αποτελεσμάτων που επιστέφει, όσο και τη μέτρηση του χρόνου απόκρισης. Για το λόγο αυτό αναπτύχθηκε ένα εργαλείο γραμμής εντολών σε γλώσσα Java [12] το οποίο εκτελεί πειράματα στην εφαρμογή, ελέγχει και αποθηκεύει τα αποτελέσματα. Τα πειράματα που εκτελεί μπορεί είτε να καθορίζονται από το χρήστη, είτε να είναι αυτοματοποιημένα από το εργαλείο κάτι που επιτυγχάνεται με τη σύνδεση του εργαλείου στη βάση δεδομένων.

Το εργαλείο όμως που αναπτύχθηκε εκτελεί πειράματα στην εφαρμογή προσομοιώνοντας ένα μεμονωμένο χρήστη που κάνει αιτήματα σειριακά. Έτσι, ο χρόνος απάντησης που μετράται δεν περιλαμβάνει τις περιπτώσεις πολλών ταυτόχρονων αιτημάτων από πολλούς και διαφορετικούς χρήστες. Η μέτρηση του συγκεκριμένου χρόνου είναι αρκετά σημαντική και αντιπροσωπευτική για την απόκριση μιας διαδικτυακής εφαρμογής καθώς το πιο πιθανό σενάριο χρήσης της είναι ακριβώς αυτό, δηλαδή η αποδοχή αιτημάτων από πολλούς ταυτόχρονους χρήστες. Για το λόγο αυτό εκτελέστηκαν πειράματα μέτρησης της απόκρισης της εφαρμογής προσομοιώνοντας μεγάλο αριθμό ταυτόχρονων χρηστών με τη χρήση του Apache JMeter [1].

Τέλος, στα πλαίσια της παρούσας διπλωματικής υλοποιήθηκε μια Web εφαρμογή για την εξαγωγή και οπτικοποίηση στατιστικών πληροφοριών που αφορούν τα δεδομένα GTFS των πόλεων. Όπως αναφέρθηκε και πιο πάνω, η μορφή των δεδομένων που δέχονται οι περισσότερες εφαρμογές δρομολόγησης είναι το πρότυπο GTFS. Καθώς από τα δεδομένα αυτά μπορεί να εξαχθεί πολύτιμη πληροφορία για ένα δίκτυο πολλαπλών Μέσων και κατ' επέκταση για

τη δρομολόγηση σε αυτό, κρίθηκε αναγκαία η ανάπτυξη μιας τέτοιας εφαρμογής. Η βασική λειτουργία της διαδικτυακής εφαρμογής που αναπτύχθηκε είναι η παρουσίαση μέσω διαγραμμάτων στατιστικών που αφορούν το δίκτυο Μέσων Μαζικής Μεταφοράς μιας ή περισσότερων πόλεων.

1.2 Οργάνωση κειμένου

Η εργασία ακολουθεί την παρακάτω δομή:

- Στο δεύτερο κεφάλαιο περιγράφονται και αναλύονται τα εργαλεία και τα πρότυπα τα οποία έχουν αναπτυχθεί και σχετίζονται με τη δρομολόγηση με πολλαπλά Μέσα. Τέτοια εργαλεία είναι το πρότυπο GTFS, το πρότυπο KML [14] και η υπηρεσία OpenStreetMaps.

- Στο τρίτο κεφάλαιο περιγράφεται η εφαρμογή OpenTripPlanner και παρουσιάζεται η δομή της και η αρχιτεκτονική της. Έπειτα, αναλύεται η διαδικασία που ακολουθήθηκε για την παραμετροποίηση και εκτέλεσή της για τις τρεις πόλεις που επιλέχθηκαν, την Αθήνα, το Παρίσι και το Πόρτλαντ. Συγκεκριμένα, περιγράφεται η διαδικασία ελέγχου, διόρθωσης και εισαγωγής των δεδομένων στην εφαρμογή και παρουσιάζονται τα αποτελέσματα εκτέλεσής της, τόσο τα γραφικά όσο και οι απαντήσεις της προγραμματιστικής διεπαφής.

- Στο τέταρτο κεφάλαιο περιγράφεται το εργαλείο γραμμής εντολών otp-client το οποίο αναπτύχθηκε για την εκτέλεση πειραμάτων στην προγραμματιστική διεπαφή του OpenTripPlanner μέσω της εκτέλεσης HTTP Requests. Αναλύεται η δομή του και παρουσιάζονται οι διάφορες λειτουργίες του όπως εκτέλεση πειραμάτων από το χρήστη, εκτέλεση αυτοματοποιημένων τυχαίων πειραμάτων και λειτουργία εξαγωγής αρχείων για το Google Earth. Τέλος, παρουσιάζονται τα αποτελέσματα μέτρησης χρόνων για τις τρεις πόλεις του OpenTripPlanner που μετρήθηκαν με τη χρήση του εργαλείου και δίνεται μια ερμηνεία αυτών.

- Στο πέμπτο κεφάλαιο αναλύεται η διαδικασία που ακολουθήθηκε για την εκτέλεση πειραμάτων προσομοίωσης ταυτόχρονου φορτίου στην προγραμματιστική διεπαφή του OpenTripPlanner. Συγκεκριμένα, αφού γίνει μια εισαγωγή στο εργαλείο Apache JMeter, παρουσιάζεται ο τρόπος οργάνωσης των πειραμάτων με που εκτελεί και αναλύονται τα αποτελέσματα που λάβαμε από την εκτέλεση των πειραμάτων με σύγκριση των χρόνων τόσο ανάμεσα στους γράφους των διαφορετικών πόλεων όσο ανάμεσα και στους διάφορους αλγόριθμους δρομολόγησης.

- Στο έκτο κεφάλαιο περιγράφεται η διαδικτυακή υπηρεσία οπτικοποίησης GTFS δεδομένων που αναπτύχθηκε. Συγκεκριμένα, παρουσιάζεται η δομή της και η αρχιτεκτονική της με περιγραφή των στοιχείων που τη συνθέτουν και των τεχνολογιών πάνω στις οποίες αναπτύχθηκε και γίνεται μια σύντομη περιγραφή της λειτουργίας της.

- Στο έβδομο κεφάλαιο περιγράφεται ένας σύντομος οδηγός για τη χρήση της υπηρεσίας οπτικοποίησης GTFS δεδομένων και παρουσιάζονται τα αποτελέσματα από τη χρήση της με τα συμπεράσματα που μπορούν να διεξαχθούν.

Κεφάλαιο 2

Σχετικές εφαρμογές και πρότυπα

Όπως αναφέρθηκε και στην εισαγωγή, το OpenTripPlanner βασίζεται σε εργαλεία και πρότυπα ανοικτού κώδικα όπως το GTFS [8] και το OpenStreetMaps [39]. Συγκεκριμένα, το GTFS χρησιμοποιείται για την κατασκευή και μοντελοποίηση των δεδομένων των δρομολογίων των Μέσων Μαζικής Μεταφοράς βάση των οποίων εκτελείται ο αλγόριθμος δρομολόγησης της εφαρμογής. Το OpenStreetMaps από την άλλη, χρησιμοποιείται τόσο για την μοντελοποίηση των γεωγραφικών δεδομένων σε συνδυασμό με τα δεδομένα των δρομολογίων όσο και για την γραφική αναπαράσταση των αποτελεσμάτων της εφαρμογής πάνω σε ένα χάρτη. Στο κεφάλαιο αυτό περιγράφονται αναλυτικά τα πρότυπα αυτά με τα χαρακτηριστικά τους. Επιπλέον, καθώς στα πλαίσια της παρούσας διπλωματικής χρειάστηκε η δημιουργία αρχείων σύμφωνα με το πρότυπο KML [14], στο τρίτο εδάφιο του κεφαλαίου περιγράφεται το πρότυπο KML με τα βασικότερα στοιχεία του. Οι περιγραφές των προτύπων στο κεφάλαιο αυτό βασίζονται σε μεγάλο βαθμό στις αντίστοιχες επίσημες διαδικτυακές πηγές που τα αναφέρουν.

2.1 GTFS

Το πρότυπο GTFS (General Transit Feed Specification) ορίζει μία ενιαία μορφολογία των δεδομένων που αφορούν τις γεωγραφικές πληροφορίες και το πρόγραμμα των δρομολογίων των Μέσων Μαζικής Μεταφοράς και διευκολύνει τόσο τη δημοσιοποίηση των δεδομένων από τους κατάλληλους οργανισμούς όσο και τη διαχείριση των πληροφοριών αυτών. Οποιοδήποτε πρακτορείο συγκοινωνιών έχει τη δυνατότητα να παράγει μία GTFS διανομή των δεδομένων του και να τα μοιραστεί με προγραμματιστές που αναπτύσσουν εργαλεία που ενσωματώνουν αυτά τα GTFS δεδομένα. Έτσι το πρότυπο GTFS μπορεί να χρησιμοποιηθεί στο σχεδιασμό ταξιδιών, στον προγραμματισμό δρομολογίων και σε ένα μεγάλο πλήθος εφαρμογών που βασίζονται στις πληροφορίες των δρομολογίων για τα Μέσα Μαζικής Μεταφοράς.

Τα δεδομένα δίνονται σε αρχεία με κατάλληλη ονομασία και το καθένα περιέχει Comma Separated Values (CSV), με την πρώτη γραμμή να ορίζει τα ονόματα των πεδίων που ακολουθούν και τις υπόλοιπες να περιέχουν τις τιμές τους. Κάθε αρχείο μοντελοποιεί ένα συγκεκριμένο στοιχείο της πληροφορίας για τα δρομολόγια των μέσων. Τα αρχεία που συνιστούν το πρότυπο πρέπει να ακολουθούν αυστηρά τις παρακάτω προϋποθέσεις:

Αρχείο	Ορισμός
agency.txt	Ένας ή περισσότεροι οργανισμοί που παρέχουν τα δεδομένα των δρομολογίων.
stops.txt	Τοποθεσίες για την επιβίβαση και αποβίβαση επιβατών με πληροφορίες για το όνομά τους και τις γεωγραφικές τους συντεταγμένες.
routes.txt	Ένα route είναι ένα σύνολο από trips με κοινό δρομολόγιο.
trips.txt	Ένα trip είναι ένα σύνολο διαδοχικών stops που συμβαίνουν μια συγκεκριμένη ώρα. Ένα route περιλαμβάνει πολλά trips.
stop_times.txt	Οι ώρες άφιξης και αναχώρησης από μία στάση για ένα συγκεκριμένο trip.
calendar.txt	Το πρόγραμμα ενός συγκεκριμένου service για όλες τις ημέρες της εβδομάδος καθώς και η ημερομηνία έναρξης και λήξης του.

Πίνακας 2.1: Ορισμός των βασικών αρχείων που απαρτίζουν το πρότυπο GTFS

Πεδίο	Ορισμός
agency_name	Το όνομα του πρακτορείου συγκοινωνιών
agency_url	Το URL του πρακτορείου συγκοινωνιών το οποίο θα πρέπει να ξεκινάει από "http://" ή "https://" και να είναι πλήρως συμβατό με τις προδιαγραφές του προτύπου URL
agency_timezone	Η ζώνη ώρας για την γεωγραφική τοποθεσία του πρακτορείου. Αν δηλωθούν πολλαπλά πρακτορεία το κάθε ένα θα πρέπει να έχει την ζώνη ώρας που του αντιστοιχεί

Πίνακας 2.2: Ορισμός των βασικών πεδίων του αρχείου agency.txt

- Όλα τα αρχεία πρέπει να περιέχουν κείμενο οριοθετημένο με κόμματα.
- Η πρώτη γραμμή κάθε αρχείου πρέπει να περιέχει τα ονόματα των πεδίων. Τα ονόματα των πεδίων που αντιστοιχούν σε κάθε αρχείο ορίζονται αυστηρά για το συγκεκριμένο αρχείο.
- Τα ονόματα των πεδίων διακρίνουν τα πεζά από τα κεφαλαία γράμματα
- Οι τιμές των πεδίων δεν πρέπει να περιέχουν ειδικούς χαρακτήρες όπως tabs ή enter.
- Οι τιμές των πεδίων που μπορεί να περιέχουν εισαγωγικά ή κόμματα πρέπει να περικλείονται σε εισαγωγικά. Επιπλέον, κάθε τιμή πεδίου που ξεκινάει με εισαγωγικά πρέπει να τελειώνει και με αυτά.
- Γενικά η μορφοποίηση ενός αρχείου είναι συμβατή με τη μορφή του CSV που εξάγει το πρόγραμμα Microsoft Excel [4].
- Η κωδικοποίηση των αρχείων πρέπει να υποστηρίζει το UTF-8 σχήμα κωδικοποίησης χαρακτήρων.
- Όλα τα αρχεία πρέπει να έχουν συμπιεστεί σε ένα αρχείο Zip.

Τα βασικά αρχεία που συνιστούν το πρότυπο φαίνονται στον πίνακα 2.1 όπως αυτά ορίζονται.

Για κάθε αρχείο ξεχωριστά ορίζονται στο πρότυπο και τα πεδία που τους αντιστοιχούν. Στους πίνακες 2.2, 2.3, 2.4, 2.5, 2.6, 2.7 περιγράφονται τα υποχρεωτικά πεδία που απαιτούνται για κάθε αρχείο.

Πεδίο	Ορισμός
stop_id	Ένα μοναδικό αναγνωριστικό που χαρακτηρίζει την στάση. Πολλαπλά δρομολόγια μπορούν να χρησιμοποιήσουν την ίδια στάση
stop_name	Το όνομα της στάσης. Πρέπει να γίνεται εύκολα κατανοητό από ταξιδιώτες και τουρίστες
stop_lat	Το γεωγραφικό πλάτος της στάσης
stop_lon	Το γεωγραφικό μήκος της στάσης

Πίνακας 2.3: Ορισμός των βασικών πεδίων του αρχείου stops.txt

Πεδίο	Ορισμός
route_id	Το αναγνωριστικό του δρομολογίου που πρέπει να είναι μοναδικό.
route_short_name	Η σύντομη ονομασία του δρομολογίου. Η ονομασία αυτή χρησιμοποιείται για την αναγνώριση του δρομολογίου από τους ταξιδιώτες αλλά δεν δίνει επιπλέον πληροφορία για τις περιοχές που αυτό εξυπηρετεί. Γενικά συνίσταται το πεδίο αυτό να μην αντιστοιχεί σε κενή συμβολοσειρά.
route_long_name	Το πλήρες όνομα του δρομολογίου. Το όνομα αυτό είναι πιο περιγραφικό από τη σύντομη ονομασία και πολλές φορές περιέχει τον προορισμό ή αφετηρία του δρομολογίου. Αν και επιτρέπεται να είναι η κενή συμβολοσειρά δεν συνίσταται. Γενικά, πρέπει τουλάχιστον ένα από τα δύο ονόματα του δρομολογίου(σύντομο ή πλήρες) να μην είναι κενό.
route_type	Ο τύπος του μεταφορικού μέσου που χρησιμοποιείται στο συγκεκριμένο δρομολόγιο. Οι τιμές που μπορεί να λάβει το πεδίο αυτό είναι ακέραιες με εύρος από 0 έως 7 και ενδεικτικά αναφέρεται ότι 0 αντιστοιχεί σε Τραμ, το 2 σε Μετρό, το 2 σε τραίνο και το 3 σε λεωφόρο.

Πίνακας 2.4: Ορισμός των βασικών πεδίων του αρχείου routes.txt

Πεδίο	Ορισμός
route_id	Αναφορά στο αναγνωριστικό ενός δρομολογίου που βρίσκεται στο αρχείο routes.txt.
service_id	Αναφορά στο αναγνωριστικό της υπηρεσίας που βρίσκεται στο αρχείο calendar.txt.
trip_id	Το αναγνωριστικό που αντιστοιχεί στο συγκεκριμένο ταξίδι. Το πεδίο αυτό πρέπει να είναι μοναδικό για κάθε ταξίδι που αναφέρεται στο αρχείο.

Πίνακας 2.5: Ορισμός των βασικών πεδίων του αρχείου trips.txt

GTFS-realtime [10]

Το GTFS-realtime αποτελεί μια επέκταση του προτύπου GTFS και επιτρέπει στα πρακτορεία συγκοινωνιών παρέχουν ενημερώσεις πάνω στα δρομολόγια τους σε πραγματικό χρόνο. Το πρότυπο αυτό σχεδιάστηκε ώστε να να διευκολύνει τη διαλειτουργικότητα του GTFS προτύπου και να εστιάσει στην άμεση και έγκαιρη ενημέρωση του επιβάτη. Υποστηρίζει τις αλλαγές που αφορούν ένα ταξίδι όπως καθυστερήσεις, ακυρώσεις και αλλαγές, τις αλλαγές σε μια υπηρεσία όπως την μεταφορά της θέσης μιας στάσης καθώς και τις τοποθεσίες των οχημάτων. Τα δεδομένα προσφέρονται μέσω μιας HTTP αίτησης από τον εξυπηρετητή του πρακτορείου συγκοινωνιών που υλοποιεί το πρότυπο αυτό.

Πεδίο	**Ορισμός**
trip_id	Αναφορά στο αναγνωριστικό ενός ταξιδιού που βρίσκεται στο αρχείο trips.txt.
arrival_time	Το πεδίο αυτό αντιστοιχεί στην ώρα άφιξης στην συγκεκριμένη στάση ενός συγκεκριμένου ταξιδιού. Η ώρα έχει τη μορφή HH:MM:SS και αντιστοιχεί σε ώρες μετά τα μεσάνυχτα (όπως μετράται συμβατικά η ώρα). Αξίζει να σημειωθεί η ιδιαιτερότητα της μέτρησης της ώρας, καθώς για κάποιο ταξίδι που επεκτείνεται και στην επόμενη μέρα (δηλαδή μετά τα μεσάνυχτα) η ώρα δεν μηδενίζεται τα μεσάνυχτα αλλά συνεχίζει να αυξάνεται ως το τέλος του ταξιδιού και παίρνει τιμές μεγαλύτερες της 24:00:00.
departure_time	Η ώρα αναχώρησης του ταξιδιού από τη συγκεκριμένη στάση. Για το πεδίο αυτό ισχύει ότι και για το πεδίο arrival_time. Αν ένα ταξίδι αναχωρεί απείθειας από τη συγκεκριμένη στάση τότε τα δύο πεδία λαμβάνουν την ίδια τιμή.
stop_id	Το αναγνωριστικό της στάσης από την οποία διέρχεται το συγκεκριμένο ταξίδι. Το πεδίο αυτό αναφέρεται στο αρχείο stops.txt.
stop_sequence	Το πεδίο αυτό ορίζει τη σειρά της στάσης στο συγκεκριμένο ταξίδι. Οι τιμές του πεδίου αυτού θα πρέπει να είναι θετικοί ακέραιοι και να αυξάνονται κατά το μήκος του ταξιδιού.

Πίνακας 2.6: Ορισμός των βασικών πεδίων του αρχείου stop_times.txt

Πεδίο	**Ορισμός**
service_id	Το πεδίο αυτό περιέχει ένα μοναδικό αναγνωριστικό το οποίο προσδιορίζει ένα σύνολο ημερών κατά τις οποίες η υπηρεσία είναι διαθέσιμη για ένα ή περισσότερα δρομολόγια. Το συγκεκριμένο αναγνωριστικό πρέπει να είναι μοναδικό για το αρχείο αυτό.
monday	Το πεδίο αυτό λαμβάνει λογικές τιμές (0 ή 1) και υποδεικνύει αν η υπηρεσία είναι διαθέσιμη κάθε Δευτέρα για το εύρος των ημερών που προσδιορίζεται στο αρχείο.
tuesday	Λογική τιμή που αφορά τη διαθεσιμότητα της υπηρεσίας κάθε Τρίτη.
wednesday	Λογική τιμή που αφορά τη διαθεσιμότητα της υπηρεσίας κάθε Τετάρτη.
thursday	Λογική τιμή που αφορά τη διαθεσιμότητα της υπηρεσίας κάθε Πέμπτη.
friday	Λογική τιμή που αφορά τη διαθεσιμότητα της υπηρεσίας κάθε Παρασκευή.
saturday	Λογική τιμή που αφορά τη διαθεσιμότητα της υπηρεσίας κάθε Σάββατο.
sunday	Λογική τιμή που αφορά τη διαθεσιμότητα της υπηρεσίας κάθε Κυριακή.
start_date	Η ημερομηνία έναρξης της υπηρεσίας σε μορφή YYYYMMD.
end_date	Η ημερομηνία λήξης της υπηρεσίας σε μορφή YYYYMMD.

Πίνακας 2.7: Ορισμός των βασικών πεδίων του αρχείου calendar.txt

2.2 OpenStreetMaps

Το OpenStreetMaps είναι μία διαδικτυακή υπηρεσία πληθοπορισμού που δημιουργεί και διανέμει ελεύθερα γεωγραφικά δεδομένα για ολόκληρο τον κόσμο. Έχει αναπτυχθεί με εθελοντές και διανέμεται με άδεια ανοιχτού περιεχομένου. Η άδεια του χάρτη επιτρέπει ελεύθερη πρόσβαση τόσο στις εικόνες που απαρτίζουν το χάρτη όσο και στα γεωγραφικά δεδομένα των περιοχών.
Το Λονδίνο του οποίου και ο χάρτης φαίνεται στην εικόνα 2.1 αποτέλεσε τη γενέτειρα του OpenStreetMaps. Τα πρώτα του γεωγραφικά στοιχεία καταγράφηκαν το καλοκαίρι του 2004 και χάρις την μεγάλη κοινότητα υποστήριξης και ανάπτυξης του εργαλείου, μέχρι σήμερα έχει γίνει εξαιρετική πρόοδος όχι μόνο για τα δεδομένα του Λονδίνου αλλά και ολόκληρου του κόσμου. Παρ' όλα αυτά, καθώς ο κόσμος είναι πολύ μεγάλος, ο χάρτης δεν είναι απόλυτα πλήρης ακόμα. Η συνεχής συνεισφορά στο έργο όμως, ιδιωτών, εταιριών και κρατών καθιστά το OpenStreetMaps μια πολύ καλή επιλογή παρόχου χάρτη και γεωγραφικών δεδομένων.

Σχήμα 2.1: Χάρτης του Λονδίνου

Το OpenStreetMaps προσφέρει τη δυνατότητα λήψης, αποθήκευσης και επαναχρησιμοποίησης από ποικίλες εφαρμογές των γεωγραφικών του δεδομένων. Η λήψη των δεδομένων για μια μικρή γεωγραφική περιοχή μπορεί να γίνει με τη χρήση της λειτουργικότητας της "εξαγωγής" που προσφέρεται από το γραφικό περιβάλλον του χάρτη. Για δεδομένα μεγαλύτερων γεωγραφικών περιοχών θα πρέπει να χρησιμοποιηθεί η διεπαφή που προσφέρει το εργαλείο και παρέχει τα δεδομένα που ζητούνται με κατάλληλη αίτηση για την περιοχή που περικλείεται από συγκεκριμένες γεωγραφικές συντεταγμένες. Τα δεδομένα δίνονται σε μορφή XML και αποτελούνται από τα εξής βασικά στοιχεία:

- Το στοιχείο *node*, που περιέχει τις γεωγραφικές συντεταγμένες μιας τοποθεσίας στο σύστημα αναφοράς WGS84.
- Το στοιχείο *way*, που περιέχει μια διατεταγμένη λίστα από nodes.
- Το στοιχείο *relation*, το οποίο μπορεί να περιέχει nodes και ways και χρησιμοποιείται για να εκφράσει τις λογικές και γεωγραφικές σχέσεις των υπόλοιπων στοιχείων μεταξύ τους.

Στην παράθεση 2.1 φαίνεται ένα παράδειγμα ενός αρχείου σε μορφή OSM XML.

2.3 KML

Το KML αποτελεί ένα πρότυπο για χρήση και αναπαράσταση γεωγραφικών πληροφοριών σε κάποιον ψηφιακό χάρτη όπως το Google Earth και το Google Maps. Η μορφή KML χρησιμοποιεί μια δομή βασισμένη στο standard της XML. Παρακάτω αναλύονται τα βασικά στοιχεία ενός αρχείου KML:

2.3.1 Point

Το στοιχείο *Point* αποτελεί μια γεωγραφική τοποθεσία η οποία ορίζεται από με γεωγραφικό μήκος, γεωγραφικό πλάτος, και (προαιρετικά) το υψόμετρο. Όταν ένα *Point* περιέχεται σε ένα *Placemark* το ίδιο το *Point* χρησιμοποιείται για να ορίσει τη θέση του *Placemark*. Τα στοιχεία που προσδιορίζουν ένα *Point* είναι τα εξής:

- *coordinates* (απαιτείται)
  Μια τριάδα που αποτελείται από δεκαδικές τιμές για το γεωγραφικό μήκος, γεωγραφικό πλάτος, και το υψόμετρο(με αυτή τη σειρά). Το γεωγραφικό μήκος και το πλάτος δίνονται σε γεωγραφικές συντεταγμένες ενώ το υψόμετρο δίνεται σε μέτρα από την επιφάνεια της θάλασσας και είναι προαιρετικό.
- *extrude*
  Καθορίζει αν θα συνδέεται το Point με το έδαφος μέσω μιας ευθείας γραμμής και δέχεται λογικές τιμές (boolean values).
- *altitudeMode*
  Καθορίζει το πως θα ερμηνεύονται οι συντεταγμένες που δίνονται. Μπορεί να λάβει τρεις πιθανές τιμές:
  - *clampToGround* (προεπιλεγμένη)
    Αγνοεί το υψόμετρο που δίνεται για τις συντεταγμένες.
  - *RelativeToGround*
    Θέτει το υψόμετρο του του στοιχείου σε σχέση με το επίπεδο του εδάφους στο συγκεκριμένο σημείο. Για παράδειγμα αν το επίπεδο του εδάφους σε ένα σημείο είναι ίσο με το επίπεδο της θάλασσας και το υψόμετρο έχει τεθεί στα 9 μέτρα τότε το ύψος του *Point* για το συγκεκριμένο σημείο θα είναι 9 μέτρα. Ωστόσο, αν το υψόμετρο ενός *Point* έχει τεθεί 9 μέτρα σε ένα σημείο του οποίου το επίπεδο είναι 10 μέτρα πάνω από αυτό της θάλασσας τότε το σημείο θα τοποθετηθεί στα 19

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<osm version=''0.6'' generator=''CGImap 0.0.2''>
    <bounds minlat=''54.0889580'' minlon=''12.2487570'' maxlat=''54.0913900''
            maxlon=''12.2524800''/>
    <node id=''298884269'' lat=''54.0901746'' lon=''12.2482632'' user=''SvenHRO''
          uid=''46882'' visible=''true'' version=''1'' changeset=''676636''
          timestamp=''2008-09-21T21:37:45Z''/>
    <node id=''261728686'' lat=''54.0906309'' lon=''12.2441924'' user=''PikoWinter''
          uid=''36744'' visible=''true'' version=''1'' changeset=''323878''
          timestamp=''2008-05-03T13:39:23Z''/>
    <node id=''1831881213'' version=''1'' changeset=''12370172'' lat=''54.0900666''
          lon=''12.2539381'' user=''lafkor'' uid=''75625'' visible=''true''
          timestamp=''2012-07-20T09:43:19Z''>
        <tag k=''name'' v=''Neu Broderstorf''/>
        <tag k=''traffic_sign'' v=''city_limit''/>
    </node>
    ...
    <node id=''298884272'' lat=''54.0901447'' lon=''12.2516513'' user=''SvenHRO''
          uid=''46882'' visible=''true'' version=''1'' changeset=''676636''
          timestamp=''2008-09-21T21:37:45Z''/>
    <way id=''26659127'' user=''Masch'' uid=''55988'' visible=''true'' version=''5''
         changeset=''4142606'' timestamp=''2010-03-16T11:47:08Z''>
        <nd ref=''292403538''/>
        <nd ref=''298884289''/>
        ...
        <nd ref=''261728686''/>
        <tag k=''highway'' v=''unclassified''/>
        <tag k=''name'' v=''Pastower Straße''/>
    </way>
    <relation id=''56688'' user=''kmvar'' uid=''56190'' visible=''true'' version=''28''
              changeset=''6947637'' timestamp=''2011-01-12T14:23:49Z''>
        <member type=''node'' ref=''294942404'' role=''''/>
        ...
        <member type=''node'' ref=''364933006'' role=''''/>
        <member type=''way'' ref=''4579143'' role=''''/>
        ...
        <member type=''node'' ref=''249673494'' role=''''/>
        <tag k=''name'' v=''Küstenbus Linie 123''/>
        <tag k=''network'' v=''VVW''/>
        <tag k=''operator'' v=''Regionalverkehr Küste''/>
        <tag k=''ref'' v=''123''/>
        <tag k=''route'' v=''bus''/>
        <tag k=''type'' v=''route''/>
    </relation>
    ...
</osm>
```

Παράθεση 2.1: OSM XML

μέτρα από το επίπεδο της θάλασσας. Αυτός ο τρόπος ερμηνείας του υψομέτρου χρησιμοποιείται συχνά στην τοποθέτηση τηλεφωνικών κεραιών ή τελεφερίκ.

– *Absolute*
Θέτει το υψόμετρο του στοιχείου σε σχέση με το επίπεδο της θάλασσας ανεξάρτητα από την ανύψωση του εδάφους στο συγκεκριμένο σημείο. Για παράδειγμα αν τεθεί το υψόμετρο ενός στοιχείου στα 10 μέτρα με αυτόν τον τρόπο, το σημείο θα είναι ορατό μόνο αν η ανύψωση του εδάφους είναι κάτω από 10 μέτρα. Μια τυπική

χρήση του συγκεκριμένου τρόπου ερμηνείας υψομέτρου είναι στην τοποθέτηση αεροσκαφών.

2.3.2 Placemarks

Το στοιχείο *Placemark* αποτελεί ένα από τα βασικότερα συστατικά ενός KML αρχείου και χρησιμοποιείται πάρα πολύ στους χάρτες του Google Earth. Τα *Placemarks* που χρησιμοποιούνται για σημειακές γεωγραφικές συντεταγμένες είναι τα σημειακά *Placemarks* (*Point Placemarks*) και είναι τα πιο ευρέως χρησιμοποιήσιμα σε ένα χάρτη Google Earth, Ένα σημειακό *Placemark* υποδεικνύει μία τοποθεσία πάνω στην επιφάνεια της γης, χρησιμοποιώντας συνήθως ένα κίτρινο δείκτη σε μορφή πινέζας ως εικονίδιο. Δίνεται η δυνατότητα να οριστεί το όνομα και το εικονίδιο για ένα σημειακό *Placemark* και καθώς και επιπλέον γεωμετρικά στοιχεία σε αυτό. Υπάρχουν τριών ειδών *Placemarks* ως προς την τοποθέτηση τους σε σχέση με το έδαφος το *simple*, το *floating* και το *extruded*. Το *simple Placemark* τοποθετείται στο ίδιο επίπεδο με το έδαφος, το *floating Placemark* τοποθετείται πάνω από το έδαφος σε απόσταση που μπορεί να προσδιοριστεί και το *extruded Placemark* τοποθετείται πάνω από το έδαφος αλλά είναι προσδεδεμένο σε αυτό. Στην εικόνα 2.2 φαίνονται οι τρεις αυτοί τύποι *Placemarks*.

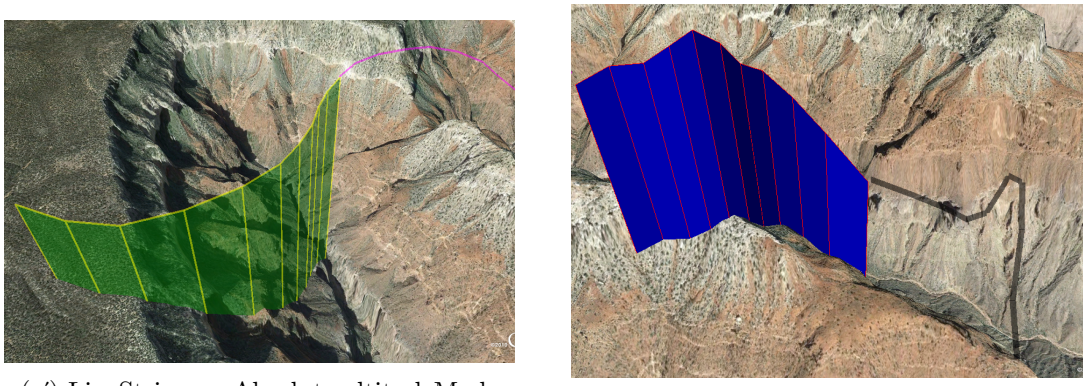
Σχήμα 2.2: Placemarks

Στην παράθεση 2.2 φαίνεται ένα σημειακό *Placemark* όπως αυτό περιγράφεται σε ένα αρχείο KML. Μπορούμε να διακρίνουμε τα εξής στοιχεία-παιδιά(child elements):

- name
  Το όνομα που χρησιμοποιείται στο *Placemark* ως ετικέτα
- description
  Η περιγραφή που περιλαμβάνεται μέσα στο "μπαλόνι" περιγραφής που αντιστοιχεί στο συγκεκριμένο *Placemark*

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Παράθεση 2.2: Σημειακό *Placemark* σε αρχείο KML



(α′) LineString με Absolute altitudeMode

(β′) LineString με Relative altitudeMode

Σχήμα 2.3: Εμφάνιση του στοιχείου LineString στον χάρτη του Google Maps

- Point
  Ένα γεωγραφικό σημείο που περιλαμβάνει συγκεκριμένες συντεταγμένες (γεωγραφικό μήκος, πλάτος, και προαιρετικά ύψος)

Το *Placemark* δεν είναι απαραίτητο να περιλαμβάνει μόνο ένα *Point* ως παιδί. Μπορεί να περιλαμβάνει πολλά γεωγραφικά στοιχεία ως παιδιά όπως ένα *LineString*, *Polygon* ή *Model*. Μόνο όμως αν έχει ένα στοιχείο *Point* ως παιδί μπορεί να απεικονιστεί με ένα εικονίδιο και μια περιγραφή.

2.3.3 LineString

Το στοιχείο *LineString* καθορίζει ένα σύνολο από ευθύγραμμα τμήματα. Μπορεί να χρησιμοποιηθεί σε συνδυασμό με το στοιχείο *LineStyle* για τον καθορισμό του χρώματος, του πλάτους και άλλων χαρακτηριστικών της γραμμής. Όταν ένα στοιχείο *LineString* δηλώνεται ως *extruded*, η γραμμή εκτείνεται ως το έδαφος "τρυπώντας" το κάθετα σχηματίζοντας έναν τοίχο. Το στοιχείο *LineString* μπορεί να περιέχει ως παιδί το στοιχείο *altitudeMode* που αναφέρθηκε παραπάνω και να το ερμηνεύει αναλόγως όπως φαίνεται και στην παράθεση 2.3. Στην εικόνα 2.3 φαίνεται η διαφορά της εμφάνισης του στοιχείου *LineString* ανάλογα με την επιλογή του *altitudeMode*.

```
<Placemark>
      <name>Absolute Extruded</name>
      <description>Transparent green wall with yellow outlines</description>
      <LineString>
        <extrude>1</extrude>
        <tessellate>1</tessellate>
        <altitudeMode>absolute</altitudeMode>
        <coordinates> -112.2550785337791,36.07954952145647,2357
          -112.2549277039738,36.08117083492122,2357
          -112.2552505069063,36.08260761307279,2357
          -112.2564540158376,36.08395660588506,2357
          -112.2580238976449,36.08511401044813,2357
          -112.2595218489022,36.08584355239394,2357
          -112.2608216347552,36.08612634548589,2357
          -112.262073428656,36.08626019085147,2357
          -112.2633204928495,36.08621519860091,2357
          -112.2644963846444,36.08627897945274,2357
          -112.2656969554589,36.08649599090644,2357
        </coordinates>
      </LineString>
```

Παράθεση 2.3: Το στοιχείο LineString

Στο κεφάλαιο αυτό έγινε περιγραφή των βασικότερων εργαλείων και προτύπων που χρησιμοποιούνται από εφαρμογές δρομολόγησης με πολλαπλά Μέσα και συγκεκριμένα από την εφαρμογή OpenTripPlanner. Στο επόμενο κεφάλαιο περιγράφεται αναλυτικά η αρχιτεκτονική της εφαρμογής OpenTripPlanner καθώς και η διαδικασία παραμετροποίησης και εκτέλεσής της.

Κεφάλαιο 3

Εκτέλεση της εφαρμογής OpenTripPlanner

Στα πλαίσια της παρούσας διπλωματικής εκτελέστηκε η εφαρμογή OpenTripPlanner [19] τόσο για τα δεδομένα της Αθήνας όσο και για δύο ακόμα πόλεις, το Παρίσι και το Πόρτλαντ. Στο παρόν κεφάλαιο περιγράφεται η εφαρμογή και η αρχιτεκτονική της και αναλύεται η διαδικασία που ακολουθήθηκε για την παραμετροποίηση και εκτέλεσή της.

3.1 Εισαγωγή στο OpenTripPlanner

Το OpenTripPlanner (OTP) είναι μία εφαρμογή ανοιχτού κώδικα για το σχεδιασμό ταξιδιών με σύστημα μεταφορών πολλαπλών Μέσων, η οποία μπορεί να εκτελεστεί σε πολλά λειτουργικά συστήματα όπως Linux, Mac και Windows ή σε οποιαδήποτε πλατφόρμα με Java Virtual Machine. Παρέχει τόσο μια γραφική διεπαφή (web interface) για χρήσης από τους ταξιδιώτες που θέλουν να σχεδιάσουν το ταξίδι τους όσο και μια προγραμματιστική διεπαφή (API) που παρέχει τη δυνατότητα ανάπτυξης ανεξάρτητων εφαρμογών, επιτρέποντας στους χρήστης της να ψάξουν για δρομολόγια με αυτοκίνητο, Μέσα Μαζικής Μεταφοράς, περπάτημα ή ποδήλατο.

Η εφαρμογή ξεκίνησε να αναπτύσσεται το 2009 και από τότε έχει προσελκύσει μια ολοένα και αυξανόμενη κοινότητα χρηστών και προγραμματιστών. Έχει λάβει χρηματοδοτήσεις από δημόσιους φορείς, εταιρίες και συμβουλευτικές υπηρεσίες συγκοινωνιών. Αν και η αρχική της υποστήριξη ήταν από το πρακτορείο TriMet του Πόρτλαντ, το οποίο αυτή τη στιγμή χρησιμοποιεί το OpenTripPlanner ως τη βάση του σχεδιασμού της τοπικής του δρομολόγησης, μέχρι σήμερα έχουν αναπτυχθεί πολλαπλές διαδικτυακές εφαρμογές δρομολόγησης για διάφορες πόλεις με βάση την μηχανή δρομολόγησης του OpenTripPlanner.

Η εφαρμογή βασίζεται σε εργαλεία και πρότυπα ανοιχτού κώδικα και συγκεκριμένα στο πρότυπο General Transit Feed Specification (GTFS) [8] για τα δεδομένα των δρομολογίων, την υπηρεσία OpenStreetMaps (OSM) [39] για τις γεωγραφικές πληροφορίες των δρόμων και την παρουσίαση των αποτελεσμάτων στο χάρτη και το πρότυπο GTFS-Realtime [10] για την τοποθέτηση των οχημάτων και τον υπολογισμό των καθυστερήσεων.

Το OpenTripPlanner ακολουθεί το μοντέλο πελάτη-εξυπηρετητή (client-server) με αρκετές γραφικές διεπαφές βασισμένες σε χάρτη για τους χρήστες και παροχή ενός REST API που

μπορεί να χρησιμοποιηθεί από εφαρμογές τρίτων. Το βασικό τμήμα του OpenTripPlanner είναι μια βιβλιοθήκη υλοποιημένη σε Java η οποία βρίσκει τις βέλτιστες διαδρομές σε δίκτυα μεταφορών πολλαπλών μέσων. Τα δεδομένα που δέχεται ως είσοδο έχουν δημιουργηθεί χρησιμοποιώντας τις γεωγραφικές πληροφορίες από το OpenStreetMaps σε συνδυασμό με κατάλληλα αρχεία GTFS. Με βάση τη βιβλιοθήκη αυτή έχουν υλοποιηθεί κάποιες υπηρεσίες με βασικότερη το API του OpenTripPlanner για την εύρεση της βέλτιστης δρομολόγησης.

Η επιλογή της εφαρμογής OpenTripPlanner για εκτέλεση αλλά και χρήση ως βάση για την ανάπτυξη εφαρμογής στα πλαίσια της παρούσας διπλωματικής έγινε για τα χαρακτηριστικά που προαναφέρθηκαν και συνοψίζονται ως εξής:

- Εφαρμογή ανοικτού κώδικα με τη δυνατότητα ελέγχου του κώδικα που εκτελείται και προσαρμογής του στις απαιτήσεις κάθε προγραμματιστή.
- Παροχή προγραμματιστικής διεπαφής (API) [16] για την ανάπτυξη οποιασδήποτε εφαρμογής τρίτων βασισμένης σε αυτή.
- Χρησιμοποίηση εργαλείων και προτύπων ανοιχτού κώδικα όπως το GTFS και το OpenStreetMaps και οποία είναι ευρέως διαδεδομένα καθιστώντας εύκολη την εύρεση γεωγραφικών πληροφοριών για κάθε περιοχή.
- Ευρέως χρησιμοποιούμενη από προγραμματιστές και χρήστες και μεγάλη κοινότητα υποστήριξης και ανάπτυξής της.
- Πλήρως τεκμηριωμένη με παροχή καλογραμμένου documentation προς τους προγραμματιστές.
- Αρκετές δυνατότητες παραμετροποίησης ως προς την επιλογή των αλγορίθμων δρομολόγησης που χρησιμοποιούνται κάθε φορά.

3.2 Θεωρητικό υπόβαθρο

Στο παρόν εδάφιο θα αναλυθούν τα μοντέλα δεδομένων και οι αλγόριθμοι που χρησιμοποιούνται από εφαρμογές δρομολόγησης σε δίκτυα πολλαπλών Μέσων, ανάμεσα στις οποίες συμπεριλαμβάνεται και το OpenTripPlanner.

3.2.1 Μοντελοποίηση δικτύων μεταφοράς

"*Η μοντελοποίηση των απλών οδικών δικτύων είναι αρκετά απλή. Σε κάθε τμήμα δρόμου μπορεί να αντιστοιχηθεί μια ακμή και σε κάθε διασταύρωση δύο ή περισσότερων τμημάτων ένας κόμβος. Το κόστος κάθε ακμής θα μπορούσε να αντιπροσωπεύει την ώρα που χρειάζεται για την διάσχιση του συγκεκριμένου τμήματος δρόμου. Έτσι, η βέλτιστη διαδρομή από ένα σημείο Α σε ένα σημείο Β δηλαδή η διαδρομή με τον ελάχιστο συνολικό χρόνο θα μπορούσε να υπολογιστεί με την εφαρμογή ενός αλγορίθμου εύρεσης βέλτιστων μονοπατιών στο γράφο*" [27]. Ένας τέτοιος αλγόριθμος είναι ο γνωστός Dijkstra είτε αυτούσιος είτε με τη χρήση κάποιας ευρετικής μεθόδου όπως συμβαίνει με τον A*.

"*Τα δίκτυα των Μέσων Μαζικής Μεταφοράς από την άλλη, μπορούν να μοντελοποιηθούν με παρόμοιο τρόπο, με τη διαφορά ότι εκτός από τις γεωγραφικές πληροφορίες πρέπει να ληφθούν*

επίσης υπόψιν και τα χρονικά προγράμματα των δρομολογίων" [27].Σε μια παραλλαγή του μοντέλου για τα απλά οδικά δίκτυα, υπάρχει περίπτωση το κόστος μιας ακμής να μην είναι μια σταθερά αλλά μια μεταβλητή εξαρτώμενη από το χρόνο. Για παράδειγμα, το κόστος της ακμής σε πολλές περιπτώσεις μπορεί να είναι μια γραμμική συνάρτηση που αντιστοιχίζει κάθε πιθανή ώρα άφιξης στον κόμβο της ακμής σε ένα διαφορετικό κόστος διαδρομής κατά μήκος της ακμής αυτής. Το μοντέλο αυτό θα μπορούσε να να χρησιμοποιηθεί για τα δίκτυα Μέσων Μαζικής Μεταφοράς και έχει αποδειχθεί ότι μια απλή παραλλαγή του αλγορίθμου του Dijkstra μπορεί να υπολογίσει τις ελάχιστες διαδρομές σε ένα τέτοιο μοντέλο.

Τα βασικά μοντέλα αναπαράστασης των δρομολογίων είναι δύο, το μοντέλο επέκτασης του χρόνου (time-expanded model) και το μοντέλο εξάρτησης χρόνου(time-depended model).

Μοντέλο επέκτασης χρόνου

Στο μοντέλο επέκτασης χρόνου για κάθε κάθε γεγονός άφιξης και αναχώρησης δημιουργείται ένας κόμβος, οι κόμβοι ομαδοποιούνται σε σταθμούς και οι ακμές αντιστοιχίζονται είτε σε ακμές αναμονής ανάμεσα σε δύο κόμβους του ίδιου σταθμού είτε σε ακμές μεταφοράς ανάμεσα σε δυο διαφορετικούς σταθμούς. Το μοντέλο όμως αυτό δεν λαμβάνει υπόψιν του ότι η αλλαγή μέσου παίρνει χρόνο και επιπλέον οι πολλές αλλαγές μέσων καλό θα είναι να αποφευχθούν (πολλές φορές κοστίζει και επιπλέον χρήματα). Ένας απλός τρόπος για να λυθεί αυτό το πρόβλημα είναι η δημιουργία όχι ενός αλλά δύο κόμβων για κάθε γεγονός άφιξης και αναχώρησης από τους οποίους ο ένας θα αναπαριστά την κατάσταση όπου ο επιβάτης βρίσκεται στο όχημα και ο άλλος την κατάσταση όπου ο επιβάτης βρίσκεται στο σταθμό.

Σε μια απλή μορφή, η αναζήτηση θα ξεκινά από ένα δεδομένο σταθμό κατά την ώρα της νωρίτερης αναχώρησης και θα καταλήγει σε ένα σταθμό προορισμό. Πιο ρεαλιστικά όμως, η αφετηρία και ο προορισμός δεν θα είναι απλά σταθμοί αλλά γεωγραφικές περιοχές από τις οποίες ο ταξιδιώτης θα πρέπει να περπατήσει μέχρι την κοντινότερη στάση. Αυτό είναι πολύ σημαντικό ειδικά για δημοτικές περιοχές με αραιά μέσα όπου μάλιστα δεν είναι ξεκάθαρο προς πια στάση θα πρέπει να περπατήσει ο ταξιδιώτης πρώτα. Η εύρεση της βέλτιστης αρχικής ή ακόμα και της τελικής στάσης αποτελεί μέρος του προβλήματος και δημιουργεί ομάδες από στάσεις αφετηρίας και προορισμού. Το παραπάνω πρόβλημα δεν αποτελεί θέμα στα οδικά δίκτυα χωρίς μέσα μεταφοράς καθώς τα οδικά δίκτυα είναι αρκετά πυκνά και η επιλογή του κοντινότερου τμήματος δρόμου για αφετηρία δίνει σωστά αποτελέσματα.

Μοντέλο εξάρτησης χρόνου

Στο μοντέλο εξαρτώμενο από το χρόνο κάθε στάση αναπαριστάται από έναν κόμβο και η ακμή από μία στάση σε μια άλλη έχει κόστος d-x +t όπου d>x αντιπροσωπεύει την επόμενη αναχώρηση του μέσου από τον πρώτο κόμβο στον δεύτερο και t η διάρκεια του ταξιδιού.

Μια μη τετριμμένη επέκταση του μοντέλου είναι η χρήση συναρτήσεων πολλαπλών κριτηρίων. Η επέκταση αυτή είναι λογική για τα οδικά δίκτυα και υποχρεωτική για τα δίκτυα μαζικής μεταφοράς. Για παράδειγμα, οι χρήστες συνήθως ενδιαφέρονται τόσο για την διάρκεια του ταξιδιού όσο και για τον αριθμό τον μετεπιβιβάσεων αλλά συχνά δεν μπορούν να ελαχιστοποιηθούν και τα δύο ταυτόχρονα: μπορεί να υπάρχει μια διαδρομή που διαρκεί δύο ώρες και δεν απαιτεί καμία μετεπιβίβαση και μπορεί να υπάρχει μια άλλη διαδρομή που διαρκεί μια ώρα και απαιτεί δύο μετεπιβιβάσεις. Κάποιοι χρήστες μπορεί να προτιμήσουν την πιο γρήγορη διαδρομή, ενώ κάποιοι άλλοι την διαδρομή που απαιτεί τα λιγότερα μέσα μεταφοράς και επομένως

θα πρέπει να υπολογίσουμε και να παρουσιάσουμε και τις δυο επιλογές.

Ένα άλλο πρακτικό πρόβλημα που περιπλέκει αρκετά τον υπολογισμό των διαδρομών σε δίκτυα μαζικής μεταφοράς είναι το γεγονός ότι κάποια δρομολόγια διεξάγονται μόνο ορισμένες μέρες και ώρες.

Τα δυο παραπάνω προβλήματα είναι συνδεμένα μεταξύ τους καθώς πλέον δεν μπορούμε να θεωρήσουμε ότι κάθε κόμβος του γραφήματος προσδιορίζεται μονάχα από ένα κόστος αλλά από ένα σύνολο από ασύνδετα μεταξύ τους κόστη. Ο αλγόριθμος του Dijkstra μπορεί εύκολα να επεκταθεί ώστε να αντιμετωπίσει την παραπάνω περίπτωση. Τα αντικείμενα στην ουρά είναι πλέον ανεξάρτητα κόστη (από τα οποία αποτελείται ένας κόμβος) και όταν θέτουμε ένα καινούργιο κόστος πρέπει να λάβουμε υπόψιν και όλα τα υπόλοιπα που είναι συνδεμένα με τον συγκεκριμένο κόμβο.

3.2.2 Αλγόριθμοι δρομολόγησης

Ο υπολογισμός της βέλτιστης διαδρομής ενός δικτύου μεταφοράς μεταξύ μιας αφετηρίας και ενός προορισμού αποτελεί ένα πολύ βασικό πρόβλημα σε πολλές σύγχρονες εφαρμογές. Πολλές φορές είναι αναγκαίος ο υπολογισμός της βέλτιστης διαδρομής, τόσο από ιδιώτες που θέλουν να σχεδιάσουν ένα ταξίδι με Μέσα Μαζικής Μεταφοράς όσο και από εφαρμογές που υπολογίζουν τη δρομολόγηση σε ένα δίκτυο, την κίνηση ή το κόστος των διαδρομών [30].

Αλγόριθμος του Dijkstra

Η πιο διαδεδομένη μέθοδος εύρεσης της ελάχιστης διαδρομής από ένα αρχικό σε ένα τελικό κόμβο σε ένα δεδομένο γράφο είναι ο αλγόριθμος του Dijkstra ο οποίος χρονολογείται από το 1950. Εν συντομία, ο αλγόριθμος του Dijkstra λειτουργεί ως εξής. Σε κάθε κόμβο αντιστοιχίζεται ένα εκτιμώμενο κόστος ελάχιστης διαδρομής το οποίο είναι αρχικά 0 για τον κόμβο αναχώρησης και άπειρο για όλους τους υπόλοιπους κόμβους του γράφου. Ο αλγόριθμος ξεκινά από τον κόμβο αναχώρησης και επισκέπτεται όλες τις εξερχόμενες ακμές. Για κάθε ακμή,ελέγχει αν μπορεί χρησιμοποιώντας τη να προσεγγίσει το κόμβο με τον οποίο συνδέεται με μικρότερο κόστος από το ήδη υπάρχον. Αν μπορεί (το οποίο συμβαίνει πάντα την πρώτη φορά προσπέλασης του κόμβου), το ενδεικτικό κόστος του κόμβου αναβαθμίζεται. Η διαδικασία αυτή ονομάζεται "χαλάρωση" ακμής. Όταν όλες οι εξερχόμενες ακμές ενός κόμβου έχουν χαλαρώσει ο κόμβος αυτός θεωρείται "εξερευνημένος". Στην επόμενη επανάληψη, επιλέγουμε τον κόμβο με την μικρότερη εκτίμηση κόστους μέχρι στιγμής και χαλαρώνουμε τις εξερχόμενες ακμές. Η διαδικασία επαναλαμβάνεται μέχρι να γίνουν "εξερευνημένοι" όλοι οι κόμβοι του γράφου.

Για τον υπολογισμό της πολυπλοκότητας του αλγορίθμου θεωρούμε ότι για κάθε επανάληψη βρίσκουμε ανάμεσα σε όλους τους κόμβους αυτόν με τη μικρότερη εκτίμηση κόστους χρησιμοποιώντας μια ουρά προτεραιότητας σε χρόνο O(log n) όπου n ο αριθμός των κόμβων. Η διαδικασία αυτή θα πραγματοποιηθεί και για τους n κόμβους άρα συνολικά θα εκτελεστεί σε χρόνο O(nlogn). Η "χαλάρωση" κάθε ακμής απαιτεί σταθερό χρόνο και θα εκτελεστεί για m φορές σε χρόνο O(m) όπου m το άθροισμα των εξερχόμενων ακμών των κόμβων του γράφου το οποίο φράζεται από τον αριθμό των ακμών. Άρα συνολικά ο αλγόριθμος θα έχει πολυπλοκότητα O(nlogn + m) όπου n ο αριθμός ο κόμβων και m ο αριθμός των ακμών.

Ο αλγόριθμος του Dijkstra, ακόμα και 50 χρόνια από την ανακάλυψή του, δεν έχει αποδειχθεί ότι είναι ο βέλτιστος για την εύρεση της ελάχιστης διαδρομής όπως επίσης και αν

```
## V: το σύνολο των κόμβων
## s: ο αρχικός κόμβος
## t: ο κόμβος κόμβος προορισμού
## d[]: αποστάσεις από αφαιτηρία
## w[][]: βάρος ακμής που ενώνει δύο κόμβους
## prev[]: προηγούμενος κόμβος στο βέλτιστο μονοπάτι

Dijstra(Graph,s,t) //Αρχικοποίηση
Για κάθε κόμβο v ∈ Graph {
    d[v] = ∞
    prev[v] = ∅
}
d[s] = 0
Q = V
S = ∅
Όσο Q ≠ ∅ {
    u = εξαγωγή_ελαχίστου(Q)
    Αν u = t
        Τερματισμός
    S = S ∪ {u}
    Για κάθε κόμβο v, γείτονα του κόμβου u, όπου v ∉ S{
        Αν d[v] > d[u] + w[u,v] τότε
            d[v] = d[u] + w[u,v]
            prev[v] = u
    }
}
```

Παράθεση 3.1: Ψευδοκώδικας του αλγορίθμου Dijkstra

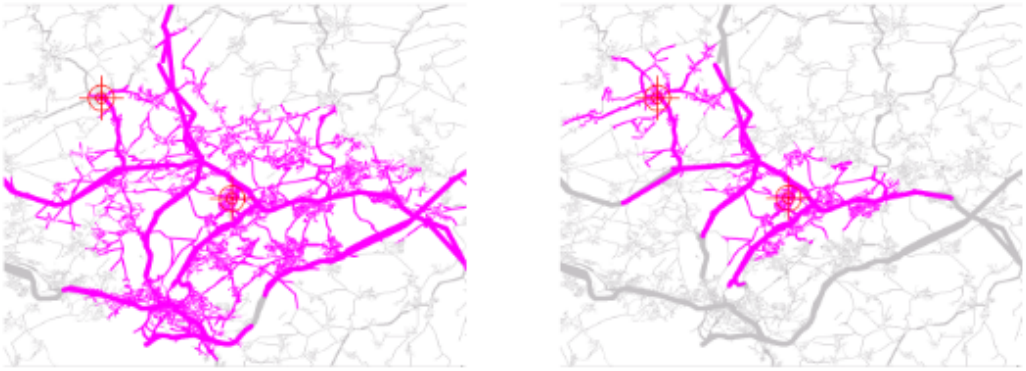
υπάρχει κάποιος αλγόριθμος που λύνει το πρόβλημα σε γραμμικό χρόνο. Ακόμα και ένας τέτοιος αλγόριθμος θα έπρεπε να επισκεφτεί κάθε ακμή και κόμβο τουλάχιστον μία φορά, κάτι που είναι αρκετά χρονοβόρο όταν το δίκτυο είναι πολύ μεγάλο. Για παράδειγμα, το οδικό δίκτυο της δυτικής Ευρώπης περιλαμβάνει περίπου 20 εκατομμύρια κόμβους και 50 εκατομμύρια ακμές οπότε η εύρεση της βέλτιστης διαδρομής θα χρειαστεί αρκετά δευτερόλεπτα σε έναν συμβατικό υπολογιστή. Στα δίκτυα μέσων μαζικής μεταφοράς έχουμε ακόμα περισσότερους κόμβους καθώς η προσθήκη εκατομμυρίων γεγονότων άφιξης και αναχώρησης για κάθε δίκτυο επεκτείνει το γράφο με εκατοντάδες εκατομμύρια κόμβους.

Αμφίδρομη αναζήτηση (bidirectional search)

Ο αλγόριθμος αμφίδρομης αναζήτησης (bidirectional search) εκτελεί τον αλγόριθμο του Dijkstra ταυτόχρονα από τον κόμβο αφετηρίας και προορισμού. Όταν κάποιος κόμβος έχει προσπελαστεί και από τις δύο κατευθύνσεις το συντομότερο δρομολόγιο μπορεί να προκύψει από τις πληροφορίες που έχουν ήδη υπολογιστεί. Ο αλγόριθμος αμφίδρομης αναζήτησης δεν προσφέρει από μόνος του μεγάλη βελτίωση στο χρόνο αναζήτησης αλλά αποτελεί βάση για πολλές εξελιγμένες τεχνικές αναζήτησης.

Στα δίκτυα Μέσων Μαζικής Μεταφοράς η αμφίδρομη αναζήτηση είναι ακόμα πιο περίπλοκη, καθώς γνωρίζουμε τον σταθμό προορισμό αλλά όχι τον συγκεκριμένο κόμβο του σταθμού στον οποίο θέλουμε να φτάσουμε. Πολλές φορές μάλιστα δεν μπορεί να υπολογιστεί εκ των προτέρων ούτε ο ίδιος ο σταθμός-προορισμός καθώς ο προορισμός μπορεί να δίνεται σε γεωγραφικές συντεταγμένες.Στην πραγματικότητα, η εύρεση του κόμβου-προορισμού αποτελεί

ένα βασικό στοιχείο του αρχικού προβλήματος. Γι' αυτό το λόγο στην αμφίδρομη αναζήτηση θα πρέπει η εκτέλεση του αλγορίθμου από τον προορισμό να πραγματοποιηθεί από ένα σύνολο πιθανών κόμβων προορισμού. Έτσι, κατά την "προς τα πίσω" αναζήτηση θα υπολογίζεται για κάθε κόμβο το κόστος του μονοπατιού από τον κόμβο που αντιστοιχεί στην συντομότερη πιθανή άφιξη για τον συγκεκριμένο σταθμό 3.1.



Σχήμα 3.1: Το εύρος αναζήτησης του αλγορίθμου Dijkstra σε ένα οδικό δίκτυο. Αριστερά: Ο Dijkstra. Δεξιά: Αμφίδρομη αναζήτηση

Αλγόριθμος A*

Ένας απλός τρόπος βελτίωσης του αλγορίθμου του Dijstra όταν είναι γνωστός ο κόμβος προορισμού είναι η επαύξηση του αλγορίθμου με μία ευρετική συνάρτηση η οποία για κάθε κόμβο θα υπολογίζει προσεγγυστικά το κόστος προς τον προορισμό. Έτσι, κάθε κόμβος επιλέγεται από την ουρά προτερεώτητας της συνάρτησης $F(S) = g(S) + h(S)$ όπου η $g(S)$ δίνει την απόσταση της S από την αρχική κατάσταση, η οποία είναι πραγματική και γνωστή, και η $h(S)$ δίνει την εκτίμηση της απόστασης της S από την τελική κατάσταση μέσω μιας ευρετικής συνάρτησης.

Ο αλγόριθμος A* εξασφαλίζει ότι η πρώτη λύση που θα βρει θα είναι η βέλτιστη στην περίπτωση που η ευρετική συνάρτηση που χρησιμοποιεί δίνει πάντα υποεκτιμήσεις, δηλαδή εκτιμήσεις μικρότερες ή ίσες της πραγματικής απόστασης κάθε κατάστασης από την κοντινότερη τελική. Μια τέτοια ευρετική συνάρτηση ονομάζεται αποδεκτή. Η αποδοτικότητα του αλγορίθμου εξαρτάται από την ποιότητα της ευρετικής συνάρτησης. Δύο ευρέως χρησιμοποιούμενες ευρετικές συναρτήσεις είναι η μέθοδος Manhattan η οποία υπολογίζει την απόσταση δύο σημείων ως το άθροισμα των διαφορών των συντεταγμένων τους και η γνωστή ευκλείδεια απόσταση.

Multiobjective A*

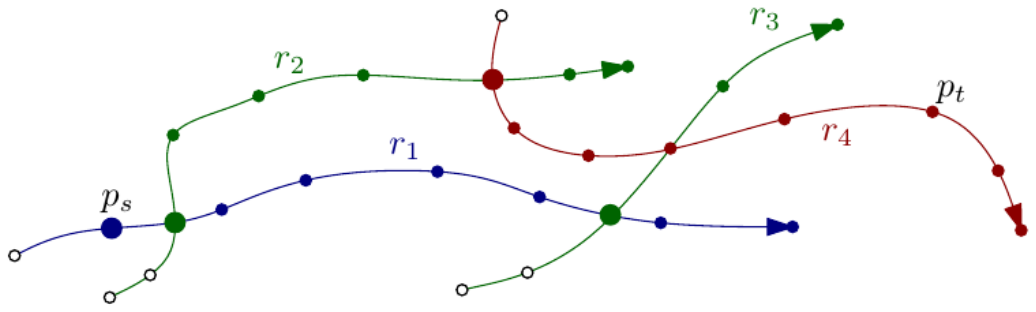
Μια διαδικασία που βρίσκει εφαρμογή σε δρομολόγηση με Μέσα Μαζικής Μεταφοράς είναι η βελτιστοποίηση με πολλαπλά κριτήρια που σημαίνει την ταυτόχρονη βελτιστοποίηση δυο ή περισσότερων αντικρουόμενων ζητημάτων με διαφόρους περιορισμούς. Η βελτιστοποίηση με πολλαπλά κριτήρια είναι πολύ σημαντική σε αυτόν το τομέα καθώς πολλές φορές χρειάζεται να

ελαχιστοποιηθεί τόσο ο χρόνος του ταξιδιού από ένα σημείο σε ένα άλλο όσο και ο αριθμός των μετεπιβιβάσεων από το ένα μέσο στο άλλο ή η απόσταση που διανύεται με τα πόδια. Για παράδειγμα, θα μπορούσαν να υπολογιστούν δύο διαφορετικές δρομολογήσεις για την μετεπιβίβαση από ένα σημείο σε άλλο, μία που διαρκεί δύο ώρες και δεν περιλαμβάνει καμία αλλαγή Μέσου και μία που διαρκεί μία ώρα με δύο μετεπιβιβάσεις από το ένα Μέσο σε άλλο. Οπότε θα πρέπει να χρησιμοποιηθεί ένας αλγόριθμος που επιλέγει το βέλτιστο για το χρήστη κάθε φορά.

Ένας βασικός αλγόριθμος βελτιστοποίησης πολλαπλών κριτηρίων σε προβλήματα δρομολόγησης είναι ο αλγόριθμος Multiobjective A* (MOA*) [40]. Ο αλγόριθμος MOA* επεκτείνει τον A* εισάγοντας την έννοια των πολλαπλών κριτηρίων. Η ποιοτική διαφοροποίησή του έγκειται στο ότι κάθε κόμβος που εξάγεται από την ουρά προτεραιότητας ως ελάχιστος για κάποιο κριτήριο υπάρχει περίπτωση να επανατοποθετηθεί για επανεξέταση και έλεγχο ως προς κάποιο άλλο κριτήριο.

Raptor [29]

Ο Raptor είναι ένας αλγόριθμος που υπολογίζει τη βέλτιστη δρομολόγηση για ένα δίκτυο Μέσων Μαζικής Μεταφοράς χρησιμοποιώντας δύο κριτήρια, τον χρόνο άφιξης και τον αριθμό των μετεπιβιβάσεων. Αντίθετα με πολλούς αλγόριθμους εύρεσης βέλτιστης δρομολόγησης, ο Raptor δεν βασίζεται καθόλου στον αλγόριθμο του Dijkstra και ελέγχει κάθε δρομολόγιο (route) το πολύ μια φορά σε κάθε επανάληψη εκτέλεσής του. Συγκεκριμένα, ξεκινώντας από τον κόμβο αφετηρίας, διασχίζει τα δρομολόγια που τον περιέχουν και όταν συναντήσει κάποιο κόμβο από τον οποίο περνά κάποιο εναλλακτικό δρομολόγιο διασχίζει και το δρομολόγιο αυτό στον επόμενο γύρο εκτέλεσής του. Γενικά επεκτείνεται το εύρος του πολύ σε κάθε γύρο εκτέλεσής του καλύπτοντας μεγάλο εύρος του δικτύου. Στην εικόνα 3.2 φαίνονται τα δρομολόγια που εξετάζει σε κάθε γύρο για την εύρεση της βέλτιστης διαδρομής από τον κόμβο ps στον κόμβο pt. Το βασικό πλεονέκτημα του αλγορίθμου αυτού είναι η δυνατότητα παραλληλοποίησής του που προσφέρει κλιμακωσιμότητα όταν εκτελεστεί σε σύστημα με υπολογιστικούς πυρήνες.



Σχήμα 3.2: Εκτέλεση του αλγορίθμου Raptor. Στον πρώτο γύρο εξετάζεται το r1, στο δεύτερο το r2 και r3 και στον τρίτο το r4.

3.3 Τεχνικά χαρακτηριστικά εφαρμογών δρομολόγησης

Όπως αναφέρεται και στην εισαγωγή, οι περισσότερες εφαρμογές δρομολόγησης (όπως και το OpenTripPlanner) είναι διαδικτυακές για την διευκόλυνση πρόσβασης σε αυτές από τους τελικούς χρήστες. Συνήθως, παρέχουν τόσο ένα γραφικό περιβάλλον για το σχεδιασμό των ταξιδιών από τους χρήστες, όσο και μια προγραμματιστική διεπαφή για την ανάπτυξη σχετικών εφαρμογών από τρίτους με τη βοήθειά της. Στην παρούσα ενότητα γίνεται μια εισαγωγή στα Restful web services πάνω στα οποία είναι βασισμένη η αρχιτεκτονική πολλών εφαρμογών δρομολόγησης συμπεριλαμβανομένου και του OpenTripPlanner.

3.3.1 REST APIs

Το API του OpenTripPlanner αποτελεί ένα RESTful web service δηλαδή ακολουθεί τις προδιαγραφές της αρχιτεκτονικής REST (Representational State Transfer). Στο αρχιτεκτονικό μοντέλο REST τα δεδομένα και οι λειτουργίες πάνω σε αυτά θεωρούνται πόροι στο διαδίκτυο (resources) οι οποίοι αναπαρίστανται με κατάλληλα διαδικτυακά αναγνωριστικά (Uniform Resource Identifiers). Το αρχιτεκτονικό μοντέλο REST περιορίζει την αρχιτεκτονική του συστήματος σε μοντέλο πελάτη-εξυπηρετητή (Client–Server) και έχει σχεδιαστεί ώστε να χρησιμοποιεί κάποιο stateless πρωτόκολλο επικοινωνίας όπως είναι το HTTP. Στο αρχιτεκτονικό μοντέλο REST ο πελάτης επικοινωνεί με τον εξυπηρετητή χρησιμοποιώντας τυποποιημένες διεπαφές και πρωτόκολλα. Πολλές από τις υπηρεσίες REST δέχονται ένα απλό HTTP GET URI ως στοιχεία εισόδου. Οι πιο σύνθετες υπηρεσίες δέχονται στοιχεία εισόδου JSON μέσω HTTP GET (για ανάκτηση), POST (για δημιουργία) ή PUT (για ενημέρωση). Τα αποτελέσματα όπως και τα μηνύματα σφαλμάτων ή οι ενδείξεις κατάστασης συνήθως επιστρέφονται σε μορφή JSON ή XML. Η μορφή JSON μπορεί να αναλυθεί και να χρησιμοποιηθεί εύκολα από τη JavaScript και άλλα προϊόντα, εργαλεία και γλώσσες, προσφέροντας έτσι μεγάλη ευελιξία για την αξιοποίησή της.

Μια διαδικτυακή υπηρεσία βασισμένη στην αρχιτεκτονική REST ακολουθεί τις εξής βασικές αρχές:

- Αναγνώριση πόρων και υπηρεσιών μέσω Uniform Resource Identifiers (URIs).
- Διαχείριση πόρων μέσω τεσσάρων τυποποιημένων διαδικασιών:
  - Δημιουργία καινούριου πόρου με χρήση της αίτησης PUT.
  - Διαγραφή ενός πόρου με χρήση της αίτησης DELETE.
  - Ανάκτηση της τρέχουσας κατάστασης ενός πόρου σε κάποια μορφή αναπαράστασης με χρήσης της αίτησης GET.
  - Επεξεργασία και αλλαγή της τρέχουσας κατάστασης ενός πόρου με χρήση της αίτησης POST.
- Ανεξαρτησία πόρων από την αναπαράστασή τους ώστε να υπάρχει δυνατότητα περιγραφής της κατάστασής τους σε μια ποικιλία μορφών όπως HTML, XML, απλό κείμενο, PDF, JPEG, JSON και άλλες.

3.4 Αρχιτεκτονική OpenTripPlanner

Στα δύο προηγούμενα εδάφια έγινε μια σύντομη εισαγωγή στις τεχνολογίες και τους αλγορίθμους που χρησιμοποιούνται από εφαρμογές δρομολόγησης με πολλαπλά Μέσα. Στην παρούσα ενότητα περιγράφεται αναλυτικά η αρχιτεκτονική της εφαρμογής OpenTripPlanner που περιλαμβάνει:

- Τα δεδομένα εισόδου της εφαρμογής
- Τους αλγόριθμους δρομολόγησης που χρησιμοποιεί
- Τα ερωτήματα που δέχεται και τις απαντήσεις που προσφέρει στο χρήστη

3.4.1 Δεδομένα εισόδου

Το OpenTripPlanner δέχεται ως είσοδο τα δεδομένα των δρομολογίων σε μορφή GTFS τοποθετημένα σε κατάλληλα CSV αρχεία. Για την εύρεση των μετεπιβιβάσεων με τα πόδια και για την αποτύπωση των αποτελεσμάτων στη γραφική διεπαφή χρησιμοποιεί τα δεδομένα από το OpenStreetMaps.

3.4.2 Μοντέλα δεδομένων και αλγόριθμοι δρομολόγησης

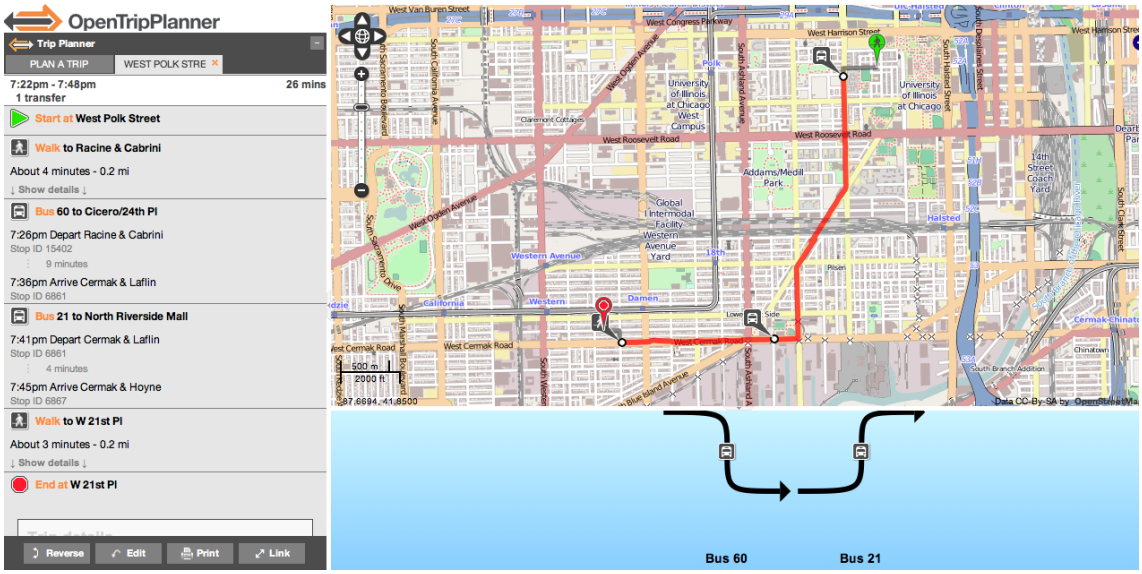
Η εφαρμογή OpenTripPlanner χρησιμοποιεί ένα χρονικά εξαρτημένο γράφο ο οποίος συνδυάζει τις γεωγραφικές πληροφορίες με το δίκτυο των Μέσων Μαζικής Μεταφοράς. Ο προκαθορισμένος αλγόριθμος της εφαρμογής είναι ο A* πολλαπλών κριτηρίων (MOA*) με κριτήρια το χρόνο άφιξης και τον αριθμό των μετεπιβιβάσεων. Ο αλγόριθμος που χρησιμοποιείται από το OpenTripPlanner για τις διαδρομές που περιλαμβάνουν μόνο περπάτημα είναι ο A* με χρήση ως ευρετικής μεθόδου την ευκλείδεια απόσταση. Επιπλέον, προσφέρεται η δυνατότητα εκτέλεσής του με τον εναλλακτικό αλγόριθμο Raptor ο οποίος έχει αποδειχθεί ότι είναι πιο γρήγορος σε μεγάλα δίκτυα. Η υλοποίηση του Raptor έχει σχεδιαστεί ώστε να είναι συμβατή με την υπόλοιπη λειτουργικότητα της εφαρμογής οπότε συνεχίζεται να χρησιμοποιείται ο Dijkstra για τα τμήματα του ταξιδιού που περιλαμβάνουν περπάτημα.

3.4.3 Χρήση γραφικής διεπαφής

Το OpenTripPlanner προσφέρει μια γραφική διεπαφή χρήστη η οποία είναι υλοποιημένη σε JavaScript και προσφέρει στο χρήστη τη δυνατότητα να σχεδιάζει τα ταξίδια του με την βοήθεια ενός γραφικού περιβάλλοντος με χάρτη. Στην εικόνα 3.3 φαίνεται η γραφική αυτή διεπαφή.

3.4.4 Χρήση προγραμματιστικής διεπαφής

Το API του OpenTripPlanner [16] αποτελεί μια διαδικτυακή υπηρεσία (web service) η οποία βασίζεται στο αρχιτεκτονικό μοντέλο του REST (Representational State Transfer) και απαντάει σε αιτήματα για σχεδιασμό ταξιδιών με διαδρομές στη μορφή JSON ή XML. Χρησιμοποιώντας το API αυτό, δίνεται η δυνατότητα υλοποίησης εφαρμογών που επικοινωνούν απ'



Σχήμα 3.3: Γραφική διεπαφή του OpenTripPlanner

Όνομα (τύπος)	Περιγραφή
date (dateTime)	Η ημερομηνία και ώρα του ταξιδιού
from (place 3.2)	Η προέλευση
to (place 3.2)	ο προορισμός
itineraries/itinerary (itinerary 3.3)	Μια λίστα με τις πιθανές διαδρομές.

Πίνακας 3.1: Η μορφή του στοιχείου tripPlan της απάντησης του API

ευθείας με αυτό και χρησιμοποιούν τις απαντήσεις που επιστρέφονται ανάλογα με το σκοπό κάθε εφαρμογής.

Η διεπαφή δέχεται ερωτήματα από το χρήστη με τη μορφή με τη μέθοδο GET του πρωτοκόλλου HTTP. Τα ερωτήματα έχουν ως παραμέτρους τα στοιχεία που αφορούν τη δρομολόγηση όπως ο χρόνος αναχώρησης, οι γεωγραφικές συντεταγμένες των σημείων αφετηρίας και προορισμού και πολλές ακόμα παραμέτρους του ταξιδιού. Οι παράμετροι αυτοί τεκμηριώνονται αναλυτικά στο documentation της εφαρμογής [17].

Η απάντηση [18] που λαμβάνεται από το API του OpenTripPlanner περιλαμβάνει την οντότητα *tripPlan*. Το *tripPlan* είναι ένα σύνολο από διαδρομές με τις οποίες μπορούμε να μεταβούμε από το σημείο A στο σημείο B μια δεδομένη χρονική στιγμή T. Η μορφή της οντότητας *tripPlan* φαίνεται στον πίνακα 3.1. Τα βασικά στοιχεία που περιλαμβάνει η οντότητα *tripPlan* είναι ένα σύνολο από *itineraries* και δύο οντότητες τύπου *place*. Το *place* είναι το σημείο από το οποίο ένα ταξίδι αρχίζει ή τελειώνει, ή μια στάση ενός ταξιδιού με μορφή που φαίνεται στον πίνακα 3.2. Το *itinerary* είναι ένα ολοκληρωμένο δρομολόγιο για την μετάβαση από την αρχική τοποθεσία στην τελική και αναλύεται στον πίνακα 3.3.

3.5 Εκτέλεση εφαρμογής

Στην ακόλουθη ενότητα γίνεται η περιγραφή της διαδικασίας που ακολουθήθηκε για την εκτέλεση της εφαρμογής. Πιο συγκεκριμένα, περιγράφουμε τα εργαλεία που χρησιμοποιήθηκαν καθώς και την αρχιτεκτονική τους. Στη συνέχεια, αναλύεται η διαδικασία που ακολουθήθηκε

Όνομα (τύπος)	**Περιγραφή**
orig (string)	(δεν υπάρχει περιγραφή)
zoneId (string)	(δεν υπάρχει περιγραφή)
stopIndex (int)	Ο αύξων αριθμός της στάσης (ξεκινάει από το μηδέν από την έναρξη του ταξιδιού)
name (string)	Για στάση μιας διαδρομής, το όνομα της στάσης. Για σημεία ενδιαφέροντος, το όνομα του σημείου.
stopId (agencyAndId)	Το μοναδικό αναγνωριστικό της στάσης. Αυτό συνήθως δεν ενδιαφέρει τους χρήστες.
stopCode (string)	Ο "κωδικός" της στάσης. Ανάλογα με το εκάστοτε ταξιδιωτικό πρακτορείο, οι χρήστες συνήθως ενδιαφέρονται για αυτό το πεδίο.
lon (double)	Το γεωγραφικό μήκος της τοποθεσίας.
lat (double)	Το γεωγραφικό πλάτος της τοποθεσίας.
arrival (dateTime)	Η ώρα άφιξης στην τοποθεσία.
departure (dateTime)	Η ώρα αναχώρησης από την τοποθεσία.

Πίνακας 3.2: Η μορφή του στοιχείου place της απάντησης του API

Όνομα (τύπος)	**Περιγραφή**
duration (long)	Διάρκεια του ταξιδιού, σε χιλιοστά του δευτερολέπτου.
startTime (dateTime)	Ώρα αναχώρησης για το ταξίδι.
endTime (dateTime)	Ώρα άφιξης στον προορισμό του ταξιδιού.
walkTime (long)	Ο χρόνος που δαπανάται περπατώντας, σε δευτερόλεπτα.
transitTime (long)	Ο χρόνος που δαπανάται ταξιδεύοντας μέσα στα οχήματα, σε δευτερόλεπτα.
waitingTime (long)	Ο χρόνος που δαπανάται περιμένοντας κάποιο όχημα, σε δευτερόλεπτα.
walkDistance (double)	Πόσο χρειάζεται να περπατήσει ο χρήστης, σε μέτρα.
elevationLost (double)	Πόση ανύψωση χάνεται, συνολικά, κατά τη διάρκεια του ταξιδιού, σε μέτρα. Για παράδειγμα, ένα ταξίδι από την κορυφή του Έβερεστ προς το επίπεδο της θάλασσας, στη συνέχεια, στην κορυφή του K2 και έπειτα πάλι κάτω θα έχει elevationLost ίσο με Έβερεστ + K2.
elevationGained (double)	Πόση ανύψωση κερδίζεται, σε μέτρα.
transfers (int)	Ο αριθμός των μετεπιβιβάσεων για αυτό το ταξίδι.
fare (fare)	Το κόστος του συγκεκριμένου ταξιδιού.
legs/leg (leg)	Μία λίστα από Legs. Ένα Leg είναι είτε ένα μέρος της διαδρομής που γίνεται με τα πόδια (ποδήλατο, αμάξι κτλ) είτε ένα μέρος της διαδρομής που γίνεται με ένα συγκεκριμένο όχημα.
tooSloped (boolean)	Αυτή η διαδρομή έχει μεγαλύτερη κλίση από την μέγιστη που ζήτησε ο χρήστης, αλλά δεν υπάρχει διαδρομή με καλή κλίση).

Πίνακας 3.3: Η μορφή του στοιχείου itinenary της απάντησης του API

για την παραμετροποίηση και εκτέλεση του OpenTripPlanner καθώς και η επεξεργασία που έγινε στα δεδομένα που δέχεται ως είσοδο.

3.5.1 Εργαλεία που χρησιμοποιήθηκαν

Το OpenTripPlanner είναι μια εφαρμογή ανοιχτού κώδικα υλοποιημένο σε γλώσσα προγραμματισμού Java. Για την εκτέλεσή του χρησιμοποιήθηκε το προγραμματιστικό εργαλείο Maven και το servlet container Apache Tomcat. Επιπλέον, για την διόρθωση των δεδομένων εισόδου της εφαρμογής και την εξαγωγή βασικών στατιστικών μετρήσεων σε αυτά, χρησιμο-

ποιήθηκε η βάση δεδομένων PostgreSQL. Στη συνέχεια δίνεται μια σύντομη περιγραφή των εργαλείων που χρησιμοποιήθηκαν. Για περισσότερες πληροφορίες ο αναγνώστης προτρέπεται στην σχετική τεκμηρίωση κάθε εργαλείου.

Java

Η Java [12] είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems, η οποία είναι πλέον θυγατρική εταιρεία της Oracle. Επιτρέπει την ανάπτυξη εφαρμογών οι οποίες μπορούν να λειτουργήσουν ανεξάρτητα πλατφόρμας, σε ένα πολύ μεγάλο εύρος συσκευών, όπως personal computers, servers, κινητά τηλέφωνα, και μικροσυσκευές. Αυτό είναι δυνατό λόγω του ότι η εκτέλεση του κώδικα δεν πραγματοποιείται άμεσα από τον υπολογιστή, αλλά μέσω του Java Virtual Machine (JVM).

Όλα τα εργαλεία που χρειάζονται ώστε να γράψει κάποιος Java προγράμματα έρχονται δωρεάν, από το περιβάλλον ανάπτυξης μέχρι εργαλεία build όπως το Apache Ant [24] και βιβλιοθήκες, ενώ υπάρχουν πολλές διαφορετικές υλοποιήσεις της Εικονικής Μηχανής και του μεταγλωττιστή (όπως the GNU Compiler for Java [25]) της Java.

Apache Maven

Πρόκειται για προγραμματιστικό εργαλείο για την διαχείριση projects λογισμικού και την αυτοματοποίηση διαφόρων εργασιών όπως το compiling, το packaging του εκτελέσιμου, την εκτέλεση δοκιμών, την εγκατάσταση ή ακόμα και την δημιουργία τεκμηρίωσης.

Το Maven [2] κατεβάζει δυναμικά τις βιβλιοθήκες Java και τα plug-ins του ίδιου που απαιτούνται, από ένα ή περισσότερα repositories. Αν και θεωρητικά το σύστημα αρχιτεκτονικής των plug-ins επιτρέπει την χρήση του για οποιαδήποτε γλώσσα προγραμματισμού, εν τούτοις στην πράξη χρησιμοποιείται σχεδόν αποκλειστικά για την Java. Το ίδιο το Maven μπορεί να χρησιμοποιηθεί και ως plug-in στο περιβάλλον του Eclipse.

Apache Tomcat

Ο Apache Tomcat [3] είναι ένας ανοικτού λογισμικού web server και servlet container που αναπτύσσεται από το Apache Software Foundation (ASF). Ο Tomcat υλοποιεί τις Java Servlet και JavaServer Pages (JSP) προδιαγραφές της Sun Microsystems. Τα Java Servlets είναι μια web τεχνολογία βασισμένη στη Java με την οποία μπορούν να αναπτυχθούν server-side εφαρμογές για το διαδίκτυο. Τα Servlets είναι προγράμματα που φορτώνονται στον διακομιστή Web και εκτελούνται όταν μια κατάλληλη εντολή HTTP που ζητά την εκτέλεσή τους λαμβάνεται από τον διακομιστή. Τα Servlets βοηθούν στην ανάπτυξη πολύ ισχυρών Web εφαρμογών χάρη στις εξής ιδιότητές τους:

- Μπορούν να εκτελεστούν χωρίς αλλαγές σε διάφορους τύπους διακομιστών.
- Βασίζονται και αναπτύσσονται στην γλώσσα προγραμματισμού Java και όχι σε μια script γλώσσα. Έτσι οι προγραμματιστές των Servlets έχουν πρόσβαση σε διάφορα εργαλεία της Java όπως η CORBA, RMI, εργαλεία ασφαλείας της Java και εργαλεία σύνδεσης βάσεων δεδομένων.
- Δίνουν την δυνατότητα για stateful αιτήματα. Το HTTP είναι ένα stateless πρωτόκολλο. Αυτό σημαίνει πως σε κάθε αίτημα που κάνει ο χρήστης ενός browser για μια ιστοσελίδα, για τον web server είναι σαν να ξεκινάει από την αρχή. Τα Servlets μπορούν να διατηρήσουν την κατάστασή τους, να διατηρήσουν κάποιο είδος μνήμης, ανάμεσα στα αιτήματα κάτι που σημαίνει ότι μπορούν να θυμηθούν δεδομένα και λεπτομέρειες ενός προηγούμενου αιτήματος.
- Παρέχουν ανεξαρτησία από το Λειτουργικό Σύστημα, μια και εκτελούνται από την Java Virtual Machine.
- Προσφέρουν καλύτερη επίδοση από άλλες τεχνολογίες, όπως για παράδειγμα το CGI (Common Gateway Interface), διότι το μοντέλο διεργασιών που χρησιμοποιείται για τα Servlets είναι αισθητά καλύτερο έναντι των άλλων τεχνολογιών. Με τον προγραμματισμό CGI σε γλώσσες όπως η Perl, κάθε φορά που ένα καινούργιο αίτημα προωθείται από τον Web browser μια καινούργια επεξεργασία πρέπει να ξεκινήσει και να τερματίσει με την λήξη σύνδεσης και το κατάλληλο πρόγραμμα πρέπει να φορτωθεί και να ξεφορτωθεί από τη μνήμη. Αντίθετα τα Servlets είναι μονίμως εγκατεστημένα στην μνήμη. Αυτό σημαίνει ότι το φορτίο είναι μικρότερο για τον διακομιστή Web.

PostgreSQL Database

Το σύστημα διαχείρισης που χρησιμοποιήθηκε για τη δημιουργία των βάσεων δεδομένων που χρησιμοποιήθηκαν στην παρούσα διπλωματική είναι η PostgreSQL [21]. Η PostgreSQL είναι μια σχεσιακή βάση δεδομένων ανοικτού κώδικα με πολλές δυνατότητες. Αναπτύσσεται ενεργά εδώ και περισσότερα από 15 χρόνια και η αρχιτεκτονική της έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία. Η PostgreSQL έχει τη δυνατότητα να εκτελεστεί σε όλα τα βασικά λειτουργικά συστήματα, στα οποία περιλαμβάνονται το Linux, το UNIX και τα Windows. Η διαχείρισή της μπορεί να γίνει είτε από τη γραμμή εντολών είτε από το εργαλείο PgAdmin [20].

Τα κυριότερα χαρακτηριστικά της PostgreSQL (πολλά από τα οποία αξιοποιήθηκαν στην παρούσα διπλωματική) είναι τα εξής:

- Πλήρης συμβατότητα με το πρότυπο ANSI-SQL:2008 [23].
- Υποστήριξη πληθώρας τύπων δεδομένων μεταξύ των οποίων είναι οι: `numberic`, `decimal`, `smallint`, `integer`, `bigint`, `real`, `double`, `serial`, `char`, `varchar`, `bit`, `text`, `date`, `time`, `timestamp`, `interval`, `boolean`, `network address`, `JSON`.
- Πλήρης υποστήριξη συναρτήσεων συγκεντρωτικών αποτελεσμάτων (`GROYP_BY`) όπως `COUNT`, `SUM`, `AVG`, `MIN`, `MAX`, `STDDEV` και `VARIANCE`.
- Υποστήριξη όλων των τύπων ενώσεων (`cross`, `inner`, `outer`, `left`, `right`, `full`, `natural`).

- Διεπαφή προγραμματισμού εφαρμογών (API) για πολλές γλώσσες όπως`Perl`, `Python`, `PHP`, και `Java`.
- Βιβλιοθήκη συναρτήσεων και τελεστών με ορισμένες προεγκατεστημένες συναρτήσεις όπως `math`, `date/time`, `string`, `geometric` και `formating`.

3.5.2 Έλεγχος και διόρθωση δεδομένων

Το OpenTripPlanner δέχεται ως είσοδο τα δεδομένα των δρομολογίων σε μορφή GTFS [8]. Για την εκτέλεση της εφαρμογής για την Αθήνα προμηθευτήκαμε τα κατάλληλα GTFS αρχεία από τον επίσημο Οργανισμό Αστικών Συγκοινωνιών της Αθήνας (ΟΑΣΑ). Για τις δύο άλλες πόλεις τα δεδομένα αποκτήθηκαν από την ιστοσελίδα GTFS Data Exchange [9] η οποία αποτελεί μια πλατφόρμα ανταλλαγής GTFS δεδομένων από τους επίσημους οργανισμούς και τους χρήστες.

Τα δεδομένα ελέγχθηκαν τόσο συντακτικά (ώστε να είναι σύμφωνα με το πρότυπο GTFS) όσο και σημασιολογικά (ώστε να δίνουν έγκυρα αποτελέσματα). Για αυτό το σκοπό χρησιμοποιήθηκε το εργαλείο FeedValidator [6] με τη βοήθεια του οποίου βρέθηκαν δύο ειδών λάθη. Συγκεκριμένα, εντοπίστηκαν αρκετές αχρησιμοποίητες στάσεις δηλαδή στάσεις οι οποίες δεν περιέχονται σε κανένα ταξίδι (stops without trips) καθώς και παλιά ημερομηνία ισχύος των δρομολογίων στο αρχείο `calendar.txt`. Τα παραπάνω λάθη δεν ήταν μέγιστης σημασίας και θα μπορούσαν να αγνοηθούν αλλά κρίθηκε χρήσιμο να διορθωθούν για την όσο το δυνατόν καλύτερη λειτουργία της εφαρμογής.

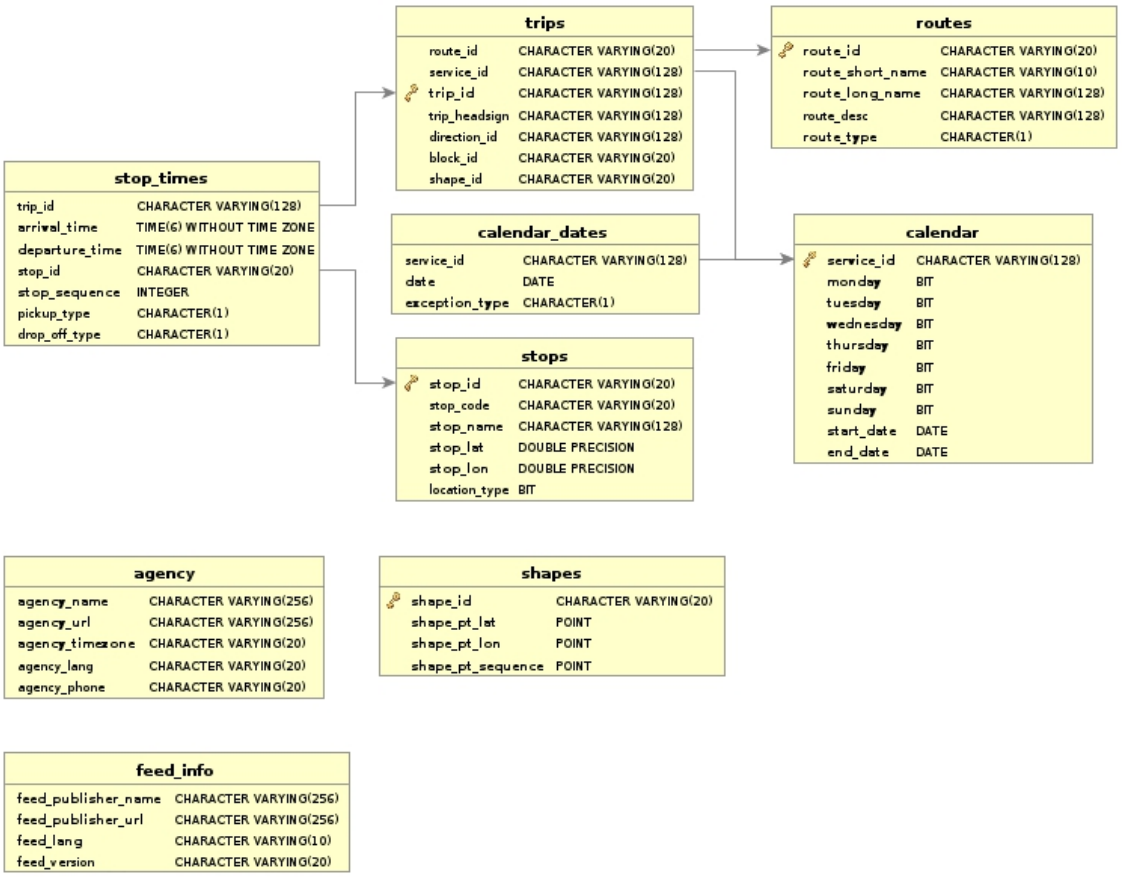
Για την διαγραφή των αχρησιμοποίητων στάσεων χρειάστηκε η εισαγωγή των δεδομένων σε μια βάση δεδομένων καθώς η εύρεση και η διαγραφή των στάσεων χωρίς ταξίδια επιτυγχάνεται εύκολα με χρήση κατάλληλων ερωτημάτων (queries). Από τα τρία διαφορετικά δεδομένα GTFS για τις πόλεις που επιλέχθηκαν, δημιουργήθηκαν αντίστοιχα τρεις βάσεις δεδομένων, μία για κάθε πόλη. Το σχεσιακό διάγραμμα της βάσης που δημιουργήθηκε με τα δεδομένα της Αθήνας φαίνεται στο σχήμα 3.4.

Τέλος, καθώς τα διαθέσιμα GTFS δεδομένα αναφερόντουσαν στην περσινή χρονιά χρειάστηκε να αλλάξουμε την ημερομηνία λήξης των δρομολογίων ώστε να ισχύουν και για την τρέχουσα ημερομηνία. Αυτό πραγματοποιήθηκε με την χρήση ενός προγράμματος υλοποιημένου σε γλώσσα python που κάνει συντακτική ανάλυση και αλλαγή του αρχείου `calendar.txt`.

3.5.3 Κατασκευή γράφου

Για να εκτελεστεί το OpenTripPlanner χρειάζεται το εργαλείο maven καθώς η εφαρμογή χρησιμοποιεί το προγραμματιστικό περιβάλλον maven για την ανάπτυξη και διαχείρισή της. Αν δεν είναι αναγκαία η αλλαγή του πηγαίου κώδικα της εφαρμογής το build μπορεί να γίνει χωρίς τη χρήση κάποιας προγραμματιστικής πλατφόρμας (π.χ. Eclipse) αλλά από τη γραμμή εντολών.

Η εφαρμογή δέχεται ως είσοδο ένα γράφο που περιέχει τις στάσεις όπως αυτές περιγράφονται στα αρχεία GTFS σε συνδυασμό με τις γεωγραφικές πληροφορίες από το OpenStreetMaps για την περιοχή που καλύπτουν. Το OpenTripPlanner παρέχει ένα εκτελέσιμο το οποίο κατασκευάζει τον γράφο λαμβάνοντας ως είσοδο τα αρχεία GTFS και OpenStreetMaps και παράγοντας ως έξοδο ένα σελιδοποιημένοι Java αντικείμενο. Τα δεδομένα του OpenStreetMaps μπορούν είτε να κατέβουν αυτόματα κατά την εκτέλεση της κατασκευής του γράφου



Σχήμα 3.4: Σχεσιακό διάγραμμα της βάσης δεδομένων με τα GTFS δεδομένα της Αθήνας

είτε να προϋπάρχουν στο μηχάνημα ώστε να γίνει απευθείας πρόσβαση σε αυτά μέσω του path που έχουν αποθηκευτεί. Στην πρώτη περίπτωση αναλαμβάνει το εκτελέσιμο πρόγραμμα να κατεβάσει τα αρχεία του OpenStreetMaps χρησιμοποιώντας τις συντεταγμένες των στάσεων που δίνονται από τα αρχεία GTFS. Το αυτόματο κατέβασμα των αρχείων μας απαλλάσσει από το να τα κατεβάσουμε χειροκίνητα αλλά καθιστά την κατασκευή του γράφου πολύ χρονοβόρα κάθε φορά καθώς πραγματοποιείται εκ νέου το κατέβασμα όλων των OpenStreetMaps αρχείων που αφορούν τη γεωγραφική περιοχή. Για το λόγο αυτό προτιμήθηκε το χειροκίνητο κατέβασμα και αποθήκευση των δεδομένων χρησιμοποιώντας το API που προσφέρει το OpenStreetMaps για την ανάκτηση και αποθήκευση γεωγραφικών δεδομένων από τη βάση δεδομένων του. Για το σκοπό αυτό αναπτύχθηκε μια απλή Java εφαρμογή που συνδέεται με τη βάση δεδομένων των GTFS αρχείων της κάθε πόλης, ανακτά τα ελάχιστα και μέγιστα γεωγραφικά μήκη και πλάτη που αφορούν τα εν λόγω δεδομένα και εκτελεί request στο API του OpenStreetMaps με την αντίστοιχη γεωγραφική περιοχή. Η απάντηση που λαμβάνει είναι σε μορφή XML και αποθηκεύονται τοπικά για τη μετέπειτα προσπέλασή της από τον κατασκευαστή του γράφου κατά τη διαδικασία της κατασκευής του. Το αποτέλεσμα της εκτέλεσης του κατασκευαστή του γράφου είναι ένα object file το οποίο χρησιμοποιείται κατά την εκτέλεση του OpenTripPlanner.

Πόλη	Δρομολόγια	Ταξίδια	Στάσεις	Εύρος δικτύου
Αθήνα	290	52.373	7808	60 km
Παρίσι	173	21.978	530	212 km
Πόρτλαντ	92	44.827	6915	66 km

Πίνακας 3.4: Μέγεθος δεδομένων GTFS για τις τρεις πόλεις

Πόλη	Αριθμός κόμβων-|V|	Αριθμός ακμών-|E|	Χρόνος δημιουργίας (sec)
Αθήνα	225.664	628.391	1039
Παρίσι	632.970	1.682.471	4738
Πόρτλαντ	433.203	1.203.929	750

Πίνακας 3.5: Μέγεθος των γράφων μοντελοποίησης δεδομένων για τις τρεις πόλεις

3.5.4 Στατιστικά των δεδομένων εισόδου

Όπως αναφέρεται και πιο πάνω η εφαρμογή OpenTripPlanner εκτελέστηκε για τα δεδομένα τριών πόλεων. Ο χρόνος υπολογισμού και αποστολής της απάντησης για τη βέλτιστη διαδρομή σε κάθε πόλη βασίζεται τόσο στην πολυπλοκότητα όσο και στο μέγεθος των δεδομένων εισόδου. Για να έχουμε μια γενική ιδέα των δεδομένων αυτών μετρήθηκε ο αριθμός των οντοτήτων που περιέχουν με κατάλληλα ερωτήματα στη βάση δεδομένων. Στον πίνακα 3.4 φαίνονται τα αποτελέσματα εκτέλεσης ερωτημάτων στη βάση αναφορικά με το μέγεθος των δεδομένων. Σημειώνεται ότι η στήλη "εύρος δικτύου" αντιπροσωπεύει την μέγιστη απόσταση δύο στάσεων του δικτύου.

Παρατηρούμε ότι η Αθήνα και το Πόρτλαντ έχουν παρόμοιο μέγεθος δεδομένων με την Αθήνα να υπερισχύει στον αριθμό των οντοτήτων που μετρήθηκαν αν και το εύρος του δικτύου της είναι ελαφρώς μικρότερο. Το Παρίσι από την άλλη, ενώ έχει ένα πολύ μεγάλο εύρος δικτύου περιέχει σχεδόν τα μισά ταξίδια από τις δύο άλλες πόλεις και αρκετά λιγότερες στάσεις. Αυτό συμβαίνει γιατί τα δεδομένα του Παρισιού αφορούν τους σιδηροδρόμους του Παρισιού οι οποίοι αποτελούν το βασικό μέσο μεταφοράς της πόλης. Για τον λόγο αυτό, φαίνεται τόσο μεγάλη η απόσταση που καλύπτεται από τα μέσα και οι στάσεις εμφανώς λιγότερες.

Με τα παραπάνω δεδομένα περιμένουμε οι γράφοι που θα δημιουργηθούν για την Αθήνα και το Πόρτλαντ να είναι παρόμοιοι σε μέγεθος ενώ ο γράφος του Παρισιού να είναι μεγαλύτερος καθώς καλύπτει σχεδόν την τριπλάσια περιοχή από τους δύο άλλους. Το μέγεθος των γράφων που δημιουργεί το OpenTripPlanner είναι αρκετά καθοριστικό σε ότι αφορά το χρόνο εύρεσης της βέλτιστης δρομολόγησης σε κάθε πόλη. Στον πίνακα 3.5 φαίνεται το μέγεθος του γράφου που δημιουργήθηκε από την εφαρμογή καθώς και ο χρόνος που δαπανήθηκε για τη δημιουργία του από το OpenTripPlanner.

Παρατηρώντας τις πληροφορίες για τους γράφους διαπιστώνεται ότι όπως περιμέναμε ο γράφος για το Παρίσι είναι αρκετά μεγαλύτερος από τους δύο άλλους. Αυτό συμβαίνει καθώς το OpenTripPlanner δεν δημιουργεί απλώς ένα γράφο με τα GTFS δεδομένα, αλλά προσθέτει και γεωγραφικά δεδομένα από το OpenStreetMaps τα οποία είναι αρκετά πυκνά και συμβάλουν στη μεγέθυνση του γράφου ιδιαίτερα όταν καλύπτουν μια μεγάλη περιοχή.

3.5.5 Παραμετροποίηση

Ο χρήστης έχει την δυνατότητα να ορίσει μέσω κατάλληλου αρχείου XML (`datasources.xml`) το γράφο τον οποίο θα χρησιμοποιήσει η εφαρμογή για την εκτέλεση του αλγορίθμου δρο-

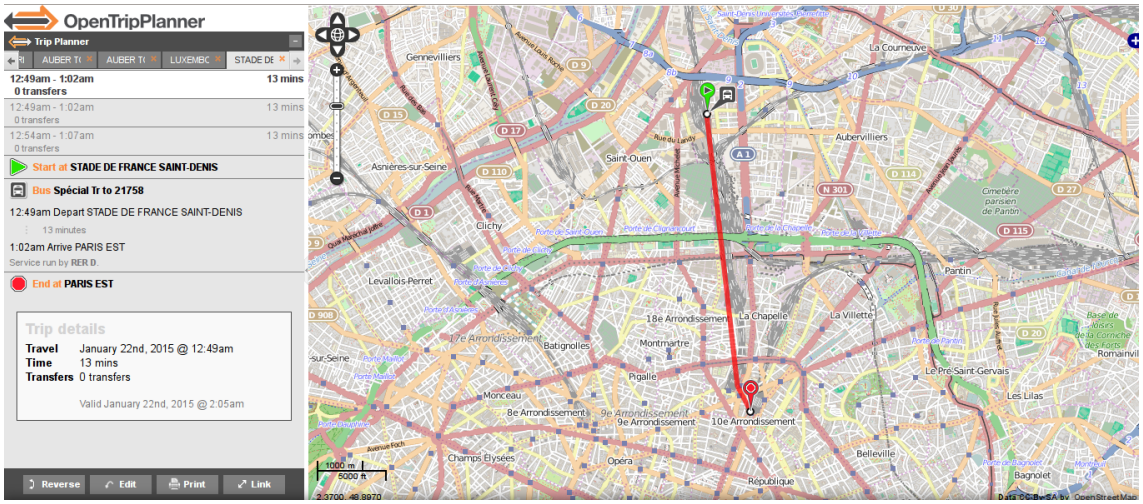
Σχήμα 3.5: Εκτέλεση του OpenTripPlanner για την περιοχή της Αθήνας

μολόγησης και τη μετέπειτα οπτικοποίησή της. Καθώς πολλές φορές προκύπτει η ανάγκη εκτέλεσης της εφαρμογής για διαφορετικές γεωγραφικές περιοχές παρέχεται η δυνατότητα της ταυτόχρονης ενσωμάτωσης περισσότερων του ενός γράφου ρυθμίζοντας κατάλληλα το αρχείο XML. Επιπλέον εκτός από τον προκαθορισμένο αλγόριθμο δρομολόγησης μπορεί να χρησιμοποιηθεί και ο εναλλακτικός αλγόριθμος Raptor. Για την επιλογή αυτού του αλγορίθμου χρειάζεται η αναφορά της κλάσης στην οποία υλοποιείται στο `application-context.xml` αρχείο.

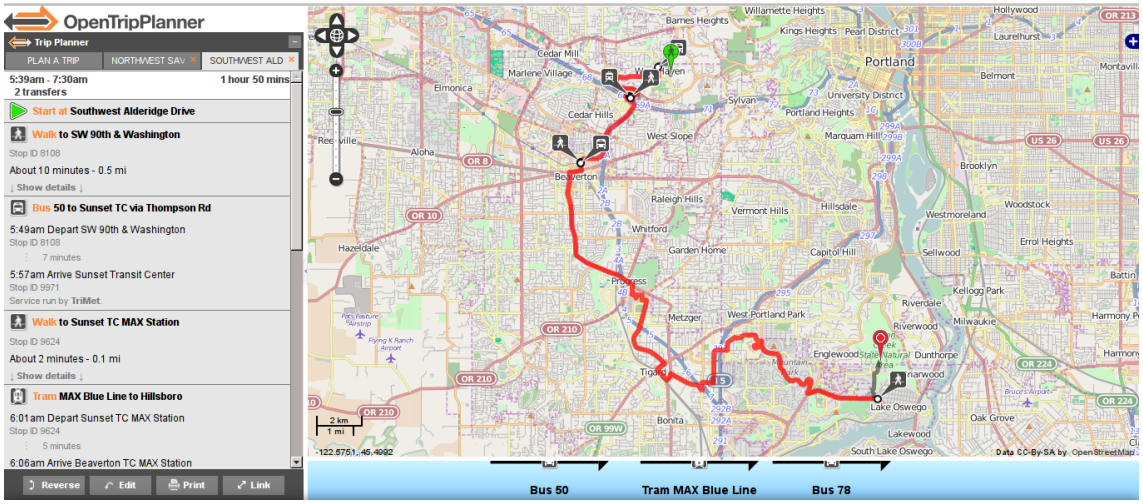
3.5.6 Εκτέλεση σε Servlet container

Η εκτέλεση της εφαρμογής έγινε σε ένα μηχάνημα που στεγάζεται στην υπηρεσία του Okeanos [15]. Το συγκεκριμένο μηχάνημα διαθέτει μνήμη μεγέθους 6 GB και 2 επεξεργαστικούς πυρήνες. Για την εκτέλεση της εφαρμογής επιλέχθηκε το Servlet container Tomcat. Δίνοντας τα κατάλληλα ορίσματα στα αρχεία παραμετροποίησης του Tomcat, ορίστηκε το μέγεθος της της χρησιμοποιούμενης μνήμης από αυτόν στα 5 GB. Το μέγεθος της μνήμης που χρησιμοποιείται από τον Tomcat και κατ'επέκταση από την εφαρμογή OpenTripPlanner τέθηκε αρκετά υψηλά ώστε να καταστήσει την εφαρμογή όσο το δυνατόν πιο αποκρίσιμη σε πολλά και ταυτόχρονα αιτήματα και στην εκτέλεση πειραμάτων.

Τα war αρχεία δημιουργούνται με τη χρήση του εργαλείου maven και τοποθετούνται μέσα στα webapps του Tomcat. Αν χρειαζόμαστε μόνο το API του OpenTripPlanner για να απαντάει σε requests με απαντήσεις XML ή JSON αρκεί η εκτέλεση του webapp opentripplanner-api-webapp. Για την προσθήκη web GUI πάνω στο οποίο θα επιλέγεται η αφετηρία και ο τερματισμός του δρομολογίου από το χρήστη, είναι αναγκαία η εκτέλεση μιας επιπλέον web εφαρμογής, του opentripplanner-webapp το οποίο επικοινωνεί με το API και αναπαριστά την απάντηση πάνω στο χάρτη του OpenStreetMaps. Στις παραθέσεις 3.2 και 3.3 φαίνεται η απάντηση του API του OpenTripPlanner σε μορφή XML και JSON αντίστοιχα. Στα σχήματα 3.5, 3.6 και 3.7 φαίνεται η εκτέλεση του webapp για την περιοχή της Αθήνας, του Παρισιού και του Πόρτλαντ αντίστοιχα.



Σχήμα 3.6: Εκτέλεση του OpenTripPlanner για την περιοχή του Παρισιού

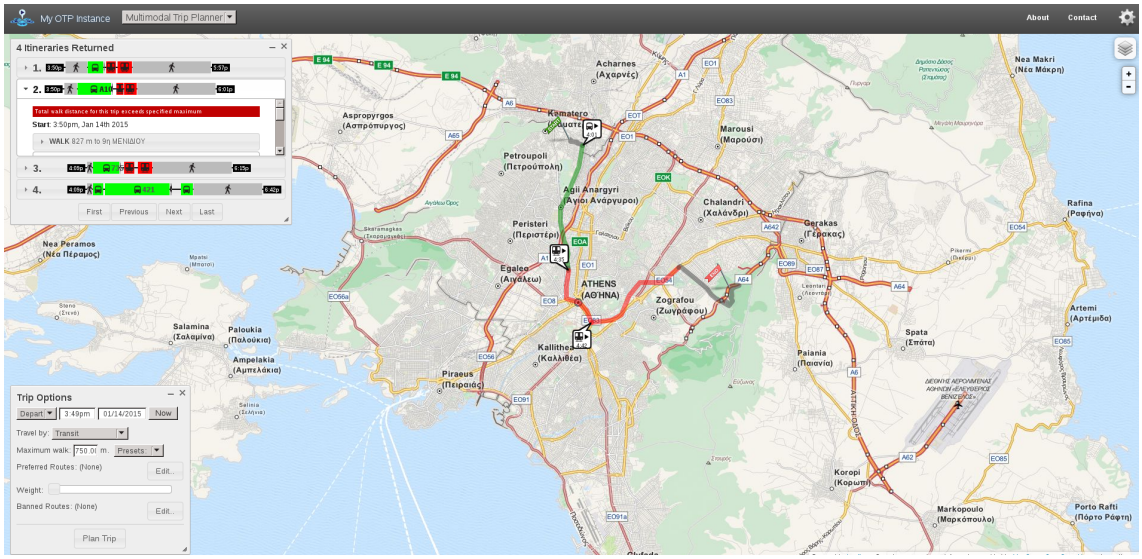


Σχήμα 3.7: Εκτέλεση του OpenTripPlanner για την περιοχή του Πόρτλαντ

3.5.7 Αυτόνομη εκτέλεση του OpenTripPlanner

Το OpenTripPlanner από την έκδοση 0.11 ενσωματώνει ένα επίπεδο HTTP Server. Χάρις σε αυτόν τον ενσωματωμένο HTTP Server το OpenTripPlanner έχει τη δυνατότητα να εκτελεστεί ως ένα αυτόνομο αρχείο JAR. Ο χρήστης μπορεί να εκτελέσει το συγκεκριμένο αρχείο και να έχει μια πλήρη λειτουργική έκδοση του OpenTripPlanner χωρίς την ανάγκη της εγκατάστασης και της διαχείρισης κάποιου Servlet container (όπως ο Tomcat). Για την υλοποίηση του ενσωματωμένου HTTP Server χρησιμοποιείται το Project Grizzly [22] ενώ για το RESTful Web Service χρησιμοποιείται το εργαλείο Jersey [13]. Στο σχήμα 3.8 φαίνεται η εκτέλεση του αλγορίθμου χρησιμοποιώντας τον ενσωματωμένο Grizzly Server.

Στο παρόν κεφάλαιο εξετάστηκε η εφαρμογή OpenTripPlanner, αναλύθηκε η αρχιτεκτονική της και περιγράφηκε η διαδικασία εκτέλεσής της για τα δεδομένα των τριών πόλεων. Καθώς προέκυψε η ανάγκη διεξαγωγής πειραμάτων στην προγραμματιστική διεπαφή της εφαρμογής, αναπτύχθηκε ένα εργαλείο γραμμής εντολών. Στο επόμενο κεφάλαιο περιγράφεται το εργαλείο αυτό και παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν με τη



Σχήμα 3.8: Εκτέλεση του OpenTripPlanner σε ενσωματωμένο Server

βοήθειά του στις τρεις πόλεις για τις οποίες εκτελείται η εφαρμογή.

```
<?xml version=''1.0'' encoding=''UTF-8''?>
<response>
  <plan>
    <from>
      <name>corner of Παράσχου and I. Βαρβάκη</name>
      <lon>23.745856408</lon>
      <lat>37.990234042000004</lat>
      <geometry>{''type'': ''Point'', ''coordinates'':
          [23.745856408,37.990234042000004]}
      </geometry>
    </from>
    <to>
      <name>corner of way 193380281 from 1 and way 31954355 from 8</name>
      <lon>23.501847805</lon>
      <lat>38.072035201</lat>
      <geometry>{''type'': ''Point'', ''coordinates'':
          [23.501847805,38.072035201]}
      </geometry>
    </to>
    <itineraries>
      <itinerary>
        <walkDistance>517.0468858949516</walkDistance>
        <transfers>2</transfers>
        <legs>
          <leg agencyTimeZoneOffset=''10800000'' mode=''WALK'' route="">
            <startTime>2013-07-07T16:29:32+03:00</startTime>
            <endTime>2013-07-07T16:32:59+03:00</endTime>
            <distance>256.1409590649571</distance>
            <from orig="">
              <name>I. Βαρβάκη</name>
              <lon>23.7459265</lon>
              <lat>37.990068</lat>
              <geometry>{''type'': ''Point'', ''coordinates'':
                  [23.7459265,37.990068]}
              </geometry>
            </from>
            <to>
              <name>ΙΠΠΟΚΡΑΤΟΥΣΑΦΕΤΗΡΙΑ()</name>
              <stopId>
                  <agencyId>Οργανισμός ΑστικώνΣυγκοινωνιώνΟΑΣΑ   -A
                      thens Urban Transport Organisation
                  </agencyId>
                <id>060024</id>
              </stopId>
              <stopCode>060024</stopCode>
              <lon>23.746830817</lon>
              <lat>37.988430122</lat>
              <arrival>2013-07-07T16:33:00+03:00</arrival>
              <departure>2013-07-07T16:33:00+03:00</departure>
              <geometry>{''type'': ''Point'', ''coordinates'':
                  [23.746830817,37.988430122]}
              </geometry>
            </to>
          </leg>
        </legs>
      </itinerary>
    </itineraries>
  </plan>
</response>
```

Παράθεση 3.2: API response σε μορφή XML

```
{
  ''plan'': {
    ''to'': {
      ''lon'': 23.838400553,
      ''name'': ''corner of Ksanthou and Melinas Merkourh'',
      ''lat'': 38.029238447999994,
    },
    ''from'': {
      ''lon'': 23.741030985000002,
      ''name'': ''Xarilaou Trikouph'',
      ''lat'': 37.98694948,
    },
    ''itineraries'': [
      {
        ''transfers'': 2,
        ''walkTime'': 549,
        ''endTime'': 1428001713000,
        ''startTime'': 1427999399000,
        ''duration'': 2314000,
        ''waitingTime'': 205,
        ''legs'': [
          {
            ''to'': {
              ''lon'': 23.756879095,
              ''name'': ''STATHMOS AMPELOKHPOI'',
              ''departure'': 1427999915000,
              ''arrival'': 1427999915000,
              ''stopCode'': ''060073'',
              ''lat'': 37.987198533,
            },
            ''routeId'': ''A7-20'',
            ''from'': {
              ''lon'': 23.741130985,
              ''name'': ''TSIMISKH'',
              ''stopCode'': ''060013'',
              ''lat'': 37.98684948,
            },
            ''endTime'': 1427999820000,
            ''agencyId'': A''thens Urban Transport Organisation'',
            ''mode'': ''BUS'',
            ''startTime'': 1427999400000,
            ''distance'': 1472.9640220557226,
            ''headsign'': ''KANIGKOS - KHFISIA'',
            ''routeShortName'': ''A7'',
            ''agencyUrl'': ''http:\/\/www.oasa.gr'',
            ''tripId'': ''7047084-CALEND-A7-Daily-01'',
            ''routeLongName'': ''KANIGKOS - KHFISIA'',
            ''agencyName'': A''thens Urban Transport Organisation'',
            ''duration'': 420000,
            ''route'': ''A7'',
            ''legGeometry'': {
              ''length'': 6,
              ''points'': ''whzfFa}{oCmHgQyC_LfA_LrCuWdDm\\''
            }
          }
        ]
      }
    ]
  }
}
```


Παράθεση 3.3: API response σε μορφή JSON

Κεφάλαιο 4

Εργαλείο γραμμής εντολών

Στο προηγούμενο κεφάλαιο αναλύθηκε η εφαρμογή OpenTripPlanner και παρουσιάστηκε η δομή και αρχιτεκτονική της. Επίσης, περιγράφηκε η διαδικασία που ακολουθήθηκε για την παραμετροποίηση και εκτέλεσή της για τα δεδομένα της Αθήνας, του Παρισιού και του Πόρτλαντ. Με την εκτέλεση της εφαρμογής δίνεται η δυνατότητα σε κάθε χρήστη να έχει πρόσβαση τόσο στην γραφική διεπαφή της εφαρμογής όσο και στην προγραμματιστική διεπαφή. Μπορεί λοιπόν, να εκτελεί requests στην προγραμματιστική διεπαφή του OpenTripPlanner και να λαμβάνει αποτελέσματα σε μορφή XML ή JSON. Για την αυτοματοποίηση της διαδικασίας αυτής αναπτύχθηκε ένα εργαλείο γραμμής εντολών το οποίο θα περιγραφεί στο παρόν κεφάλαιο.

Στα πλαίσια της διπλωματικής υλοποιήθηκε ένα command-line εργαλείο σε Java, με όνομα otp-client, το οποίο εκτελεί HTTP Requests στο API του OpenTripPlanner και αποθηκεύει τις απαντήσεις για περαιτέρω ανάλυση και επεξεργασία. Η αναγκαιότητα ανάπτυξης του συγκεκριμένου εργαλείου προέκυψε από τις παρακάτω ανάγκες:

- Ανάγκη ακρίβειας στα αποτελέσματα των requests κάτι που δεν προσφέρεται από τη διαμεσολάβηση του Web application.
- Εκτέλεση πολλαπλών πειραμάτων για διαφορετικά γεωγραφικά δεδομένα.
- Έλεγχος της ορθότητας των αποτελεσμάτων με αυτοματοποιημένο τρόπο.
- Οπτικοποίηση των αποτελεσμάτων σε σε κάποιο διαφορετικό χάρτη από το OpenStreetMaps (π.χ. Google Earth).

4.1 Εργαλεία που χρησιμοποιήθηκαν

Για την ανάπτυξη του εργαλείου χρησιμοποιήθηκε η προγραμματιστική πλατφόρμα Eclipse που περιγράφεται στο παρόν εδάφιο. Η έκδοση της Java που χρησιμοποιήθηκε για την μεταγλώττιση του εργαλείου είναι η Java 7 της Oracle.

Eclipse

Το Eclipse [5] είναι ένα περιβάλλον ανάπτυξης λογισμικού για διάφορες γλώσσες προγραμματισμού, συνδυάζοντας ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) κι ένα σύστημα πρόσθετων plug-ins. Για να γράψει κάποιος κώδικα Java δεν είναι αρκεί ένας απλός επεξεργαστής κειμένου. Ωστόσο ένα ολοκληρωμένο περιβάλλον ανάπτυξης βοηθάει πολύ, ιδιαίτερα στον εντοπισμό σφαλμάτων (debugging).

Το Eclipse είναι ένα λογισμικό ανοικτού κώδικα και η ανάπτυξή του επιβλέπεται από το Eclipse Foundation, έναν μη κερδοσκοπικό οργανισμό.

4.2 Περιγραφή λειτουργίας

4.2.1 Βασική λειτουργία

Το εργαλείο δέχεται μέσω της γραμμής εντολών τις παραμέτρους που αφορούν το request που θα εκτελεστεί στο API του OpenTripPlanner όπως αυτές περιγράφονται στο documentation [16]. Επιπλέον, δέχεται και κάποιες ειδικές επιλογές που αφορούν το μηχάνημα στο οποίο θα αποσταλεί το request, την θύρα(port), τον τύπο απάντησης (JSON, XML, KML) καθώς και το αρχείο στο οποίο θα αποθηκευτεί. Αφού διαβαστούν οι παράμετροι της γραμμής εντολών το εργαλείο κατασκευάζει το κατάλληλο URL για την εκτέλεση του request, το αποστέλλει και λαμβάνει την απάντηση στη μορφή που ζητήθηκε. Κατόπιν, την επεξεργάζεται ανάλογα με την περίπτωση και την αποθηκεύει σε κατάλληλο αρχείο.Τα πιο βασικά options του εργαλείου φαίνονται στο listing 4.1.

4.2.2 Λειτουργία εκτέλεσης πειραμάτων

Ο client που υλοποιήθηκε εκτός από τη βασική λειτουργικότητα, δηλαδή να κάνει ένα request στο OpenTripPlanner με τις παραμέτρους που δέχεται από τον χρήστη και να αποθηκεύει την απάντηση έχει και μια επιπλέον λειτουργία για εκτέλεσης πολλαπλών πειραμάτων. Συγκεκριμένα, έχει τη δυνατότητα να εκτελεί ένα μεγάλο αριθμό requests με τυχαίες παραμέτρους και να αποθηκεύει τα αποτελέσματα σε κατάλληλα αρχεία συμπεριλαμβανομένου του χρόνου απάντησης για την κάθε μία. Η χρησιμότητα της συγκεκριμένης λειτουργικότητας του εργαλείου προκύπτει τόσο από την ανάγκη για έλεγχο της ορθότητας των αποτελεσμάτων για τους διαφορετικούς αλγορίθμους δρομολόγησης, όσο και από την ανάγκη σύγκρισης του χρόνου απάντησης για ένα πλήθος διαφορετικών παραμέτρων όπως τα δεδομένα εισόδου (διαφορετικοί GTFS γράφοι), ο χρησιμοποιούμενος αλγόριθμος δρομολόγησης και η επιλογή της μορφής της απάντησης(JSON ή XML).

Για να αποσταλεί το request για ένα τυχαίο ζεύγος συντεταγμένων ο client συνδέεται στη βάση δεδομένων, η οποία περιλαμβάνει τα δρομολόγια και τις στάσεις για μια συγκεκριμένη πόλη, και κάνει τυχαία επιλογή δύο στάσεων από τη βάση. Αφού ανακτήσει τα γεωγραφικά μήκη και πλάτη για το ζεύγος των στάσεων, εισάγει σε αυτά μια απόσταση κατά ± 0.0003

```
usage: otp options
 -0,--out <file name>     File to write output
 -H,--host <name or ip>   Host name or ip for request (default: localhost)
 -h,--help                Print this message
 -p,--port <number>       Port to send request (default:8080)
 -r,--response <type>     Request for xml response or json(default)
 -tm,--test               Test mode for this tool
usage: request options
 -d,--date <arg>                        The date that the trip should
                                        depart (or arrive, for requests
                                        where arriveBy is true)
 -f,--fromPlace <arg>                   The start location -- either
                                        latitude,longitude pair in degrees
                                        or a Vertex label. For example,
                                        40.714476,-74.005966 or
                                        mtanyctsubway_A27_S
 -rI,--routerId <arg>                   Router ID used when in multiple
                                        graph mode. Unused in singleton
                                        graph mode
 -t,--toPlace <arg>                     The end location (see fromPlace
                                        for format)
```

Παράθεση 4.1: Τα βασικότερα options του OpenTripPlanner client

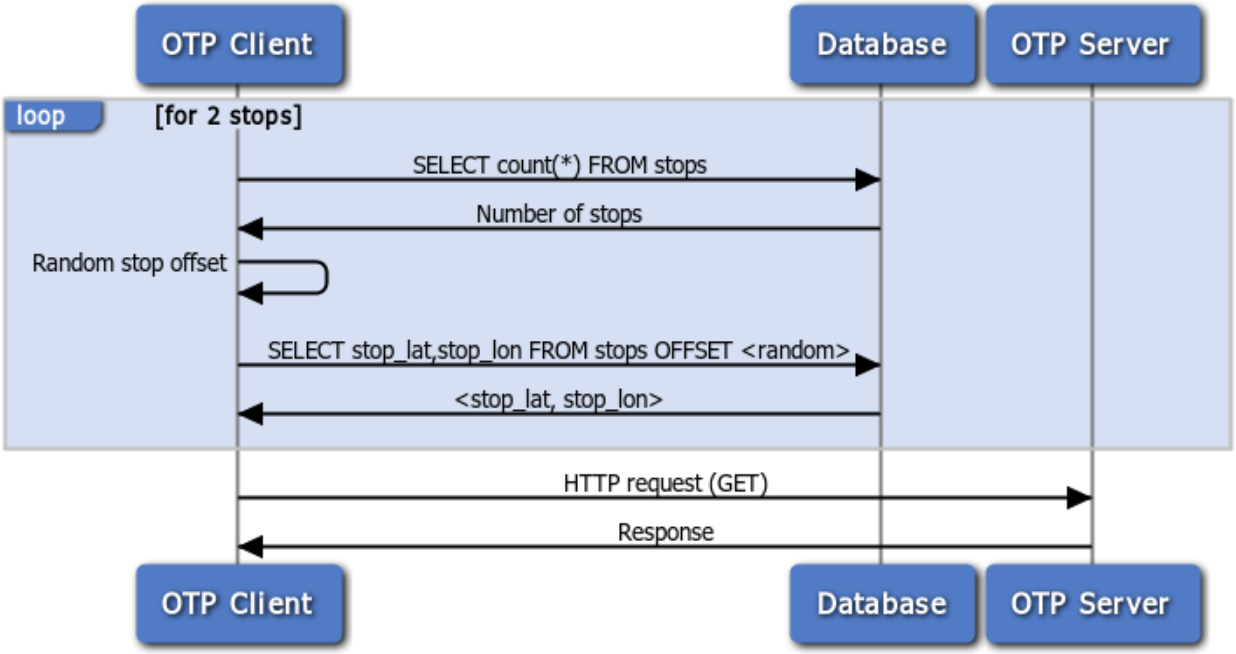
γεωγραφικές μοίρες(περίπου 30 μέτρα), ώστε να υπάρξει τουλάχιστον μία μετάβαση περπατώντας ως τις στάσεις, και εκτελεί το request στο API του OpenTripPlanner αποθηκεύοντας τη ληφθείσα απάντηση καθώς και το χρόνο που μεσολάβησε μέχρι τη λήψη της. Η διαδικασία που ακολουθείται για την εκτέλεση ενός τυχαίου request για μια πόλη είναι η παρακάτω:

- Σύνδεση του Client στη βάση δεδομένων που περιέχει τα δρομολόγια και τις στάσεις για τη συγκεκριμένη πόλη.
- Μέτρηση του αριθμού τον στάσεων της βάσης.
- Δημιουργία 2 ψευδοτυχαίων αριθμών με 2 seed με όριο τον αριθμό των στάσεων.
- Επιλογή των 2 στάσεων από τη βάση με offset τους δύο τυχαίους αριθμούς.
- Αποθήκευση των πεδίων latitude,longitude του ζεύγους στάσεων.
- Εκτέλεση του request στον Server με τις συντεταγμένες των στάσεων προσθέτοντας ± 0.0003 στην γεωγραφική τοποθεσία της κάθε στάσης.
- Αποθήκευση του αποτελέσματος συμπεριλαμβανομένου του χρόνου απάντησης σε κατάλληλο αρχείο.

Στο σχήμα 4.1 αναπαριστάται ένα ακολουθιακό διάγραμμα UML για τη λειτουργία εκτέλεσης πειραμάτων του εργαλείου.

4.2.3 Εξαγωγή KML αρχείων

Ο Client έχει τη δυνατότητα να κάνει εξαγωγή τους αποτελέσματος και σε KML μορφή ώστε να μπορεί να οπτικοποιηθεί και σε κάποιο άλλο χάρτη (εκτός του OpenStreetMaps) όπως



Σχήμα 4.1: Ακολουθιακό διάγραμμα UML για την εκτέλεση ενός τυχαίου request για τα δεδομένα μιας πόλης

είναι το Google Earth ή Google Maps.

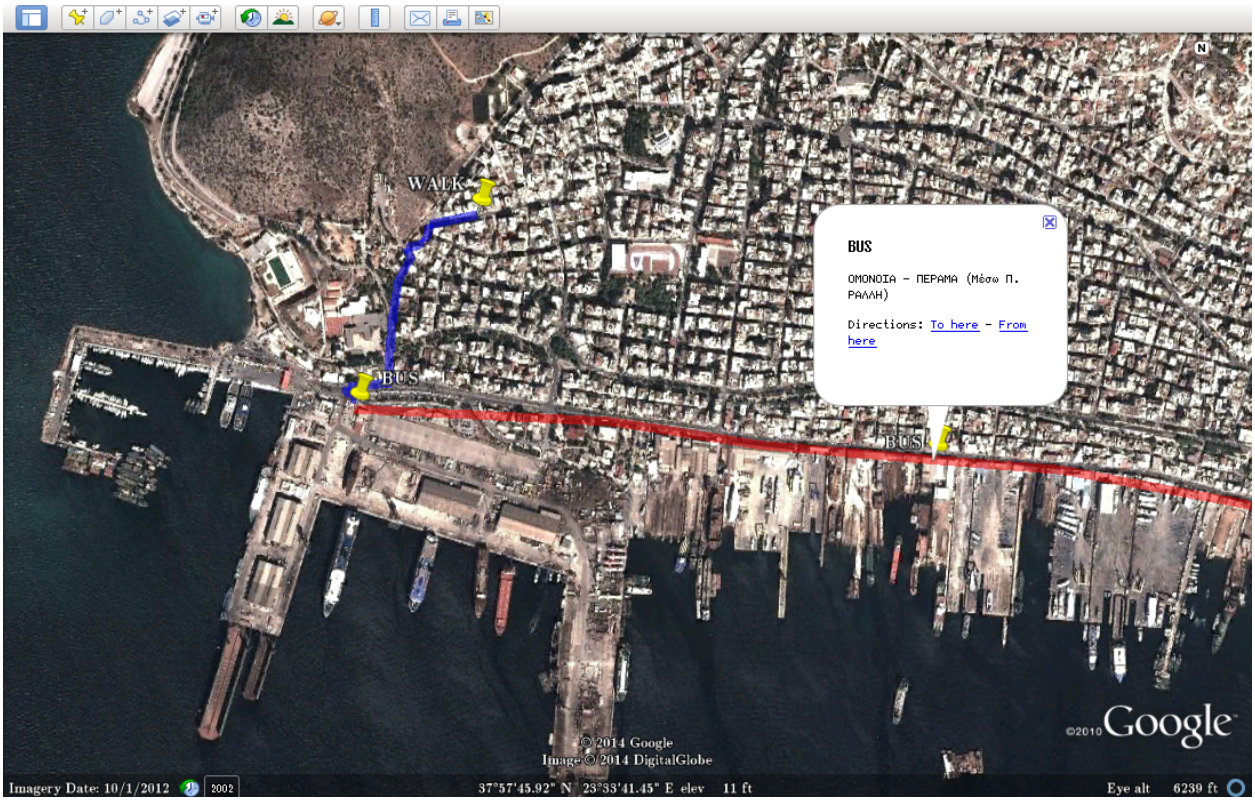
Καθώς η απάντηση από το API του OpenTripPlanner είναι σε μορφή JSON ή XML το εργαλείο μας θα πρέπει να την επεξεργαστεί με κάποιο τρόπο ώστε να τη φέρει στην επιθυμητή KML μορφή. Για να επιτευχθεί αυτό, ο Client εκτελεί αρχικά request στο Server με επιλογή απάντησης σε μορφή JSON. Όταν ληφθεί απάντηση αναλύεται συντακτικά(parsing), επιλέγονται οι συντεταγμένες της διαδρομής που υπολογίστηκε και ως αποτέλεσμα δημιουργείται ένα αρχείο σύμφωνο με το πρότυπο KML που αποτυπώνει τη διαδρομή. Για καλύτερη αναπαράσταση των τρόπων μεταβάσεως από ένα σημείο σε άλλο χρησιμοποιήθηκαν διαφορετικά χρώματα καθώς και κατάλληλα points με αντίστοιχη περιγραφή. Συγκεκριμένα, χρησιμοποιήθηκε κόκκινο χρώμα για τις μεταβάσεις με Μέσα Μαζικής Μεταφοράς και μπλε χρώμα για τις μεταβάσεις με τα πόδια. Στο σχήμα 4.2 απεικονίζεται το αποτέλεσμα ενός KML response για την Αθήνα.

4.3 Περιγραφή κλάσεων

Στην ενότητα αυτή δίνεται μια σύντομη περιγραφή των βασικών κλάσεων του client

- public class CmdOptions
  Η κλάση αυτή είναι υπεύθυνη για τη δημιουργία των κατάλληλων παραμέτρων που θα δέχεται η εφαρμογή καθώς και για την ανάλυση αυτών που θα διαβάζει από τη γραμμή εντολών.

- public class Client
  Η κλάση αυτή συνδέεται με τον server στον οποίο εκτελείται το OpenTripPlanner και στέλνει το request που έχει κατασκευαστεί. Επιπλέον, μετρά το χρόνο που μεσολάβησε



Σχήμα 4.2: KML response για την Αθήνα αποτυπωμένο στο Google Earth

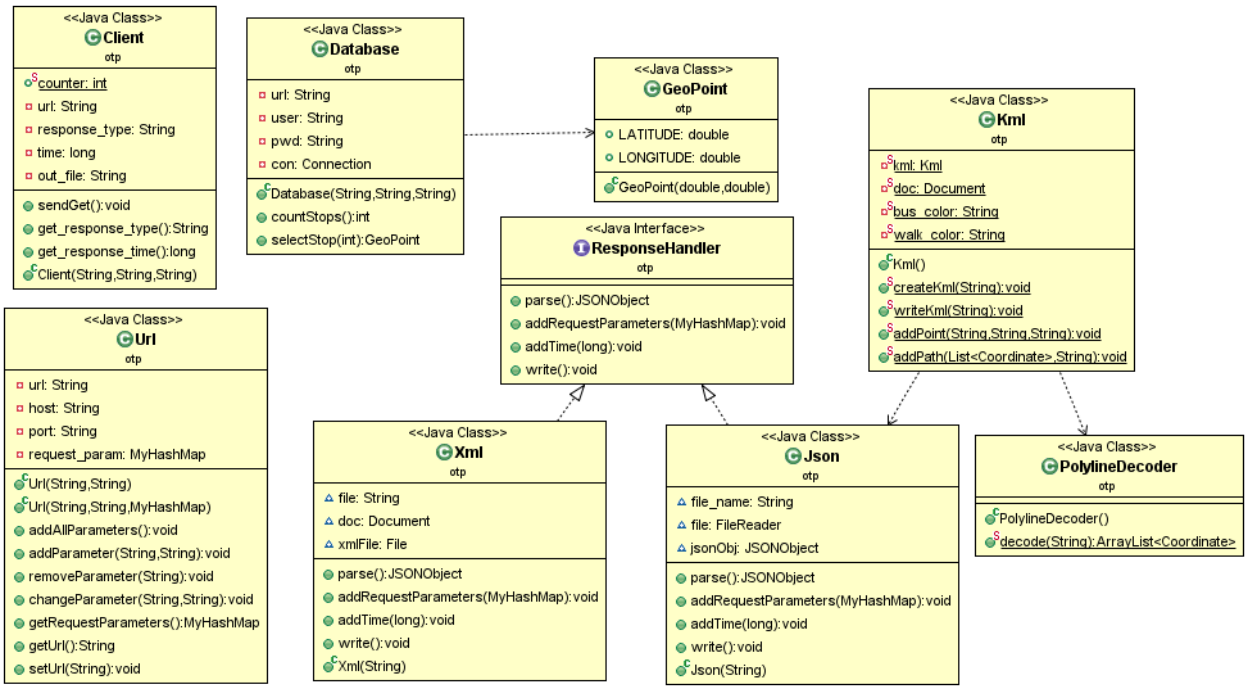
μέχρι να ληφθεί απάντηση και αποθηκεύει το response σε κατάλληλο αρχείο. Για τη μείωση του χρόνου που δαπανάται στο δίκτυο χρησιμοποιήθηκε gzip συμπίεση της απάντησης. Συγκεκριμένα, ενεργοποιήθηκε η επιλογή στον tomcat server για gzip compression από τα αρχεία ρυθμίσεών του και προστέθηκε στον client το κατάλληλο request property ώστε να ζητάει το συγκεκριμένο compression.

- public class Url
  Η κλάση αυτή κατασκευάζει το Url με το οποίο θα γίνει το request καθώς και αποθηκεύει τις παραμέτρους σε ένα hashtable ώστε να τις παρέχει αν ζητηθούν.

- public class Json
  Η κλάση αυτή επεξεργάζεται την απάντηση σε περίπτωση που είναι σε μορφή JSON ελέγχοντας την συντακτική ορθότητά της και προσθέτοντας τον χρόνο που μεσολάβησε για να ληφθεί ως μια επιπλέον εγγραφή στο JSON αρχείο.

- public class Xml
  Η κλάση αυτή είναι υπεύθυνη για την επεξεργασία της απάντησης όταν έχει μορφή XML και επιτελεί τις ίδιες λειτουργίες με την κλάση Json. Επιπλέον, καθώς το API του OpenTripPlanner δεν επιστρέφει τις παραμέτρους του requests ενσωματωμένες στην XML απάντηση, η κλάση Xml τις τοποθετεί στο XML δέντρο με κατάλληλη ετικέτα για μελλοντική χρήση.

- public class Database
  Η κλάση αυτή είναι υπεύθυνη για την επικοινωνία με τη βάση δεδομένων καθώς και για την εκτέλεση σε αυτή των ζητούμενων queries.



Σχήμα 4.3: Διάγραμμα Κλάσεων του OpenTripPlanner Client

- public class RandomStop
  Η κλάση αυτή δημιουργεί τυχαίους αριθμούς και καλεί την Database με αυτούς ώστε να επιλεγεί τυχαία μια εγγραφή από τη βάση δεδομένων. Αξίζει να σημειωθεί ότι οι αριθμοί που δημιουργούνται δεν είναι απόλυτα τυχαίοι αλλά ψευδοτυχαίο, δημιουργούνται δηλαδή με seed με σκοπό να επιλεγεί ξανά η ίδια εγγραφή αν δοθεί ως είσοδος το ίδιο seed.

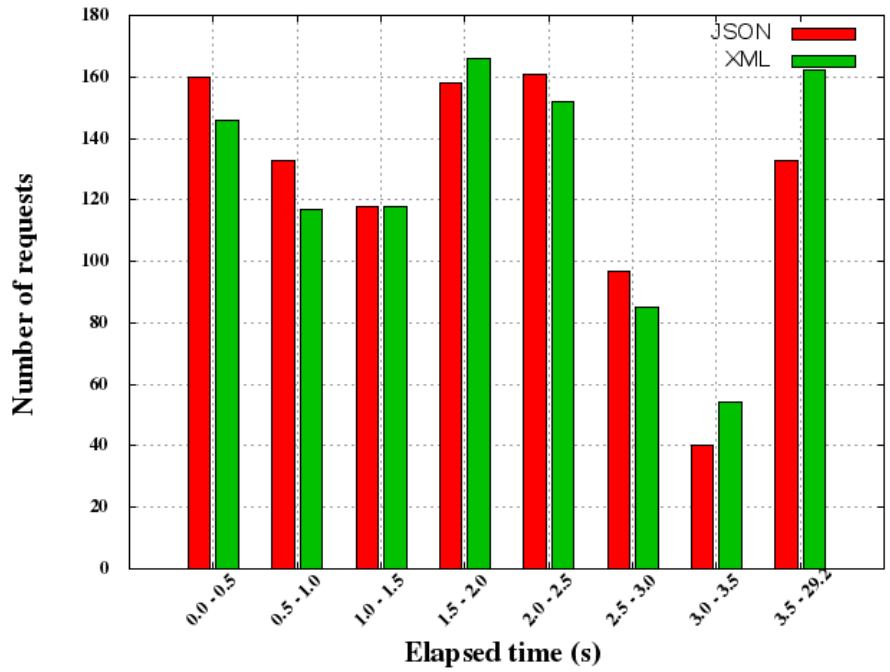
- public class Kml
  Η κλάση αυτή είναι υπεύθυνη για την επεξεργασία της απάντησης JSON και την δημιουργία του κατάλληλου αρχείου KML ώστε να μπορεί να δοθεί ως είσοδος στο Google Maps.

- public class PolylineDecoder
  Η κλάση αυτή είναι υπεύθυνη για την αποκωδικοποίηση των τιμών που ακολουθούν τη μορφή "Encoded Polyline" σε μορφή λίστας από συντεταγμένες. Η μετατροπή αυτή κρίνεται απαραίτητη καθώς ο OpenTripPlanner Server αναπαριστά τις απαντήσεις στην προαναφερθείσα μορφή κάτι που δεν μπορεί να χρησιμοποιηθεί άμεσα σε ένα KML αρχείο.

Στο σχήμα 4.3 φαίνεται το UML διάγραμμα των βασικών κλάσεων του εργαλείου.

4.4 Εκτέλεση πειραμάτων

Χρησιμοποιώντας τη λειτουργία του εργαλείου για την εκτέλεση πειραμάτων όπως περιγράφεται παραπάνω διεξάγαμε κάποια πειράματα από τα οποία μπορούμε να εξάγουμε συμπεράσματα τόσο για την ορθότητα των αποτελεσμάτων που επιστρέφονται από τον Server όσο και για το χρόνο απόκρισης του Server για κάθε περίπτωση.



Μέσος χρόνος JSON: 2.21 sec. Μέσος χρόνος XML: 2.44 sec
Σχήμα 4.4: Χρόνοι απάντησης για τα δεδομένα της Αθήνας

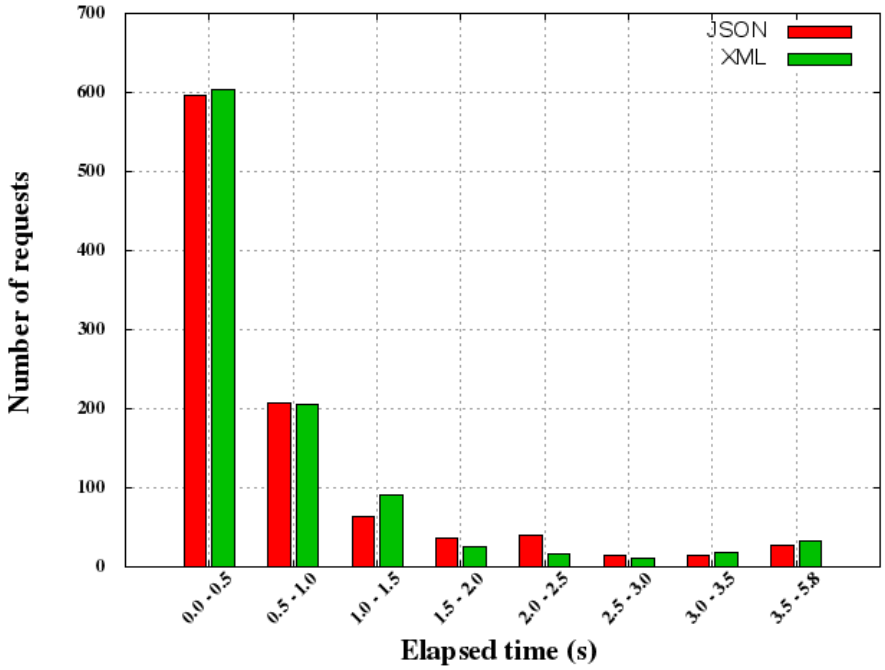
4.4.1 Οργάνωση πειραμάτων

Τα πειράματα εκτελέστηκαν για 3 ξεχωριστές πόλεις την Αθήνα, το Παρίσι και την Πόρτλαντ από ένα τοπικό μηχάνημα προς τον OpenTripPlanner server στον Okeanos [15]. Το μηχάνημα από το οποίο εκτελέστηκαν τα πειράματα έχει μνήμη 4 G και 2 επεξεργαστικούς πυρήνες. Τα δεδομένα των δρομολογίων για τις τρεις πόλεις τοποθετήθηκαν σε τρεις διαφορετικές βάσεις δεδομένων το καθένα οι οποίες έχουν τη μορφή που περιγράφεται στο προηγούμενο κεφάλαιο. Επιπλέον, τοποθετήθηκε στο servlet container Tomcat το OpenTripPlanner API ρυθμισμένο να υποστηρίζει και τους τρεις γράφους των πόλεων. Για κάθε πόλη εκτελέστηκαν 1000 requests όπου το κάθε ένα αντιστοιχεί σε ένα ζευγάρι τυχαίων στάσεων. Για κάθε πόλη τα αποτελέσματα ζητήθηκαν τόσο σε XML όσο και σε JSON μορφή. Για κάθε πόλη υπολογίστηκε ο χρόνος απάντησης τόσο για τη μορφή XML όσο και για μορφή JSON.

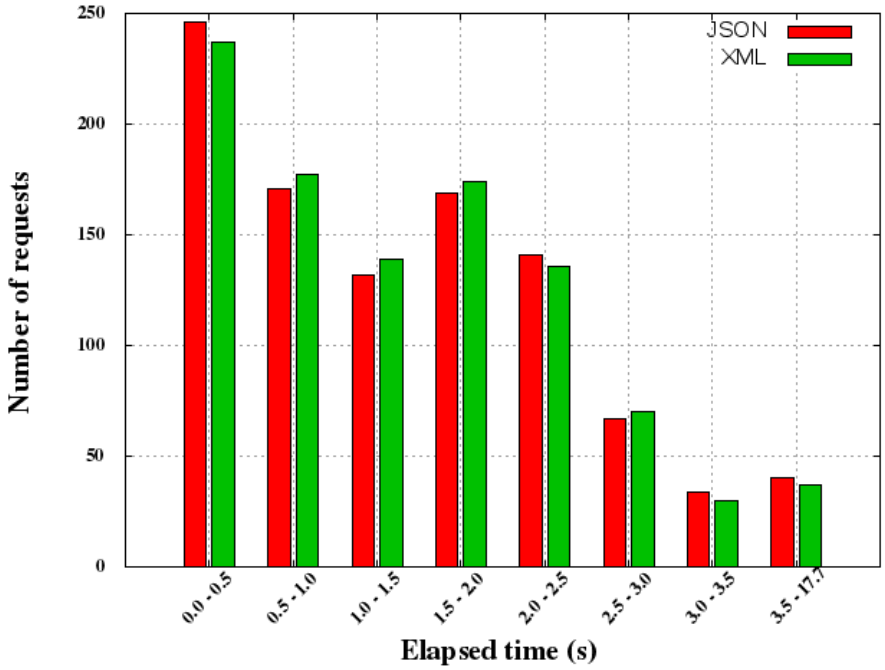
4.4.2 Αποτελέσματα

Για κάθε πόλη υπολογίστηκε ο αριθμός των requests που αντιστοιχούν σε ένα εύρος χρόνου που αφορά το χρόνο αποστολής απάντησης καθώς και ο μέσος χρόνο απάντησης. Στα σχήματα 4.4, 4.5, 4.6 και 4.7 παρουσιάζονται τα αποτελέσματα των μετρήσεων με μορφή γραφικών παραστάσεων.

Στα διαγράμματα 4.8, 4.9 και 4.10 παρουσιάζονται τα αποτελέσματα μέτρησης χρόνων όταν η εφαρμογή εκτελείται με τον αλγόριθμο Raptor ως αλγόριθμο δρομολόγησης. Τα αποτελέσματα αυτά συγκρίνονται με τα αντίστοιχα αποτελέσματα που λάβαμε με την χρήση του αλγορίθμου MOA*. Για μεγαλύτερη κατανόηση των αποτελεσμάτων στα διαγράμματα 4.12, 4.13, 4.14 , 4.15 και 4.16 παρουσιάζονται οι μετρήσεις των χρόνων αθροιστικά.

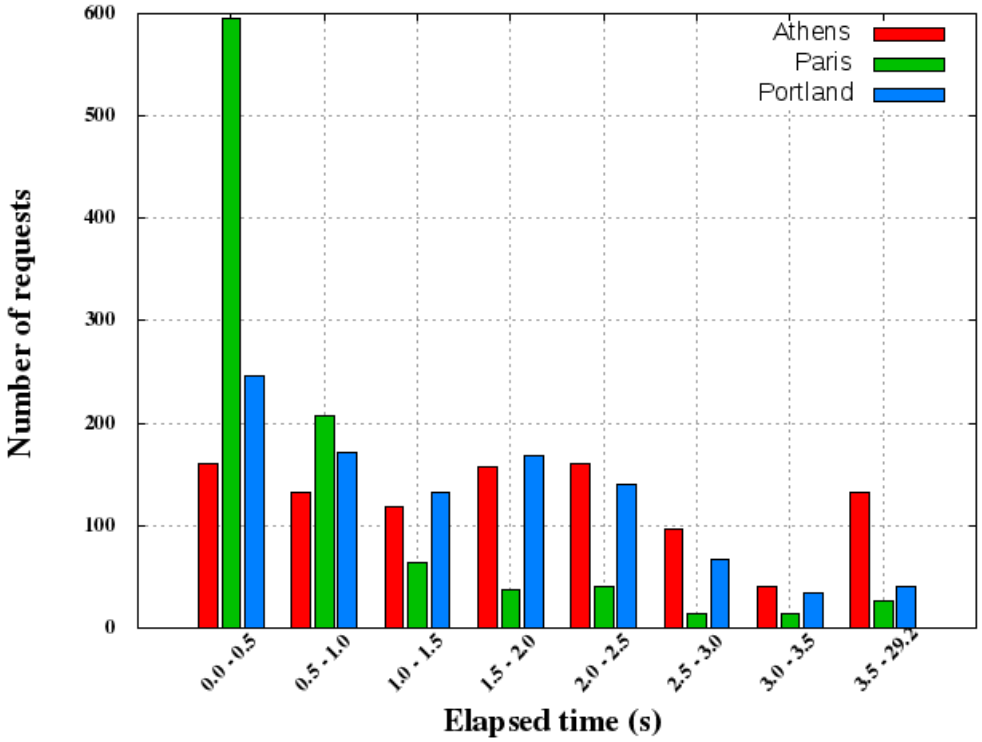


Μέσος χρόνος JSON: 0.7 sec. Μέσος χρόνος XML: 0.68 sec
Σχήμα 4.5: Χρόνοι απάντησης για τα δεδομένα του Παρισιού

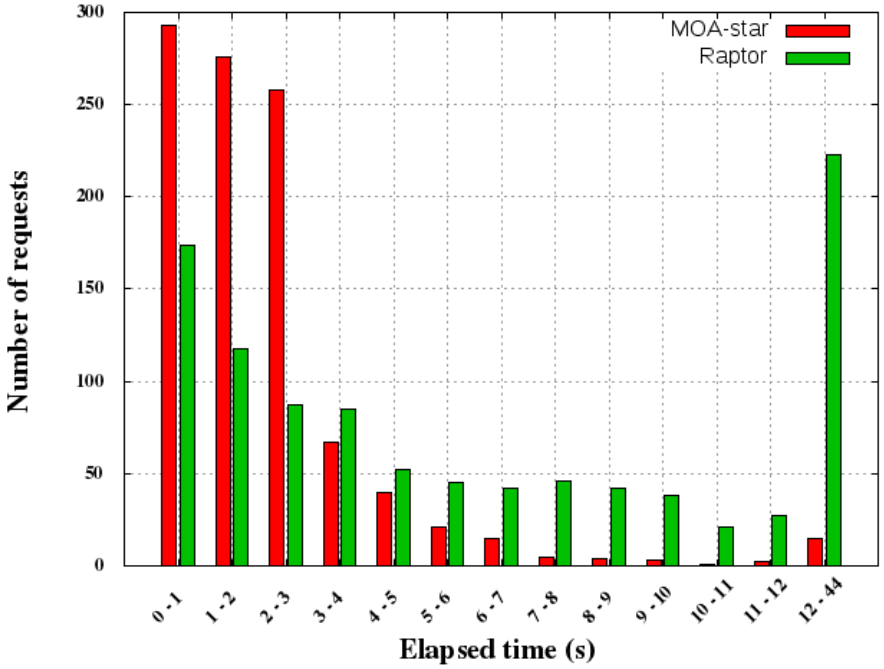


Μέσος χρόνος JSON: 1.48 sec. Μέσος χρόνος XML: 1.48 sec
Σχήμα 4.6: Χρόνοι απάντησης για τα δεδομένα του Πόρτλαντ

Επιπλέον για κάθε απάντηση υπολογίστηκε ο μέσος χρόνος απάντησης συναρτήσει της απόστασης των δύο σημείων αφετηρίας και προορισμού. Τα αποτελέσματα παρουσιάζονται στα διαγράμματα 4.17, 4.18, 4.19, 4.20 και 4.21.

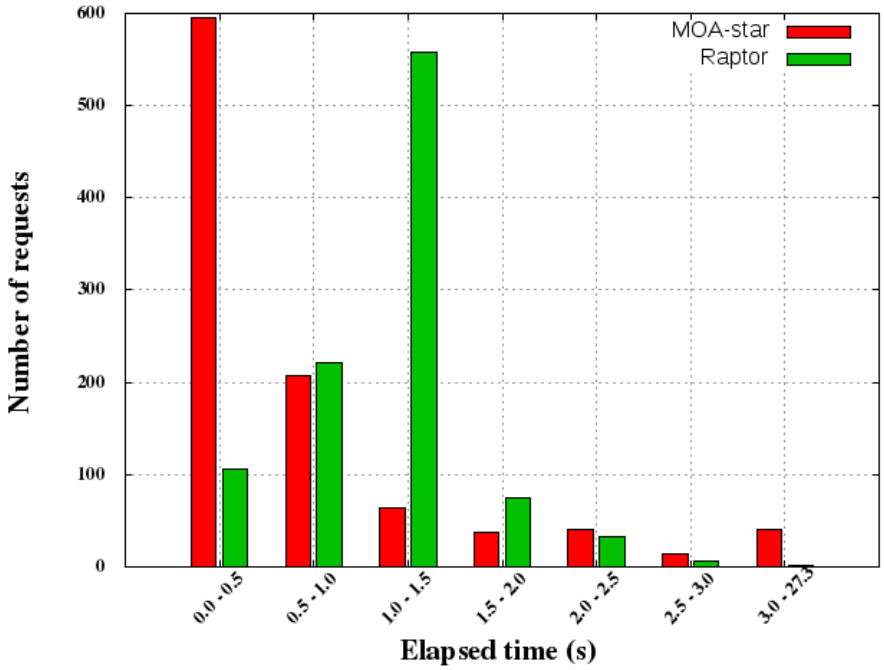


Σχήμα 4.7: Σύγκριση χρόνων εκτέλεσης του MOA* για τις 3 πόλεις

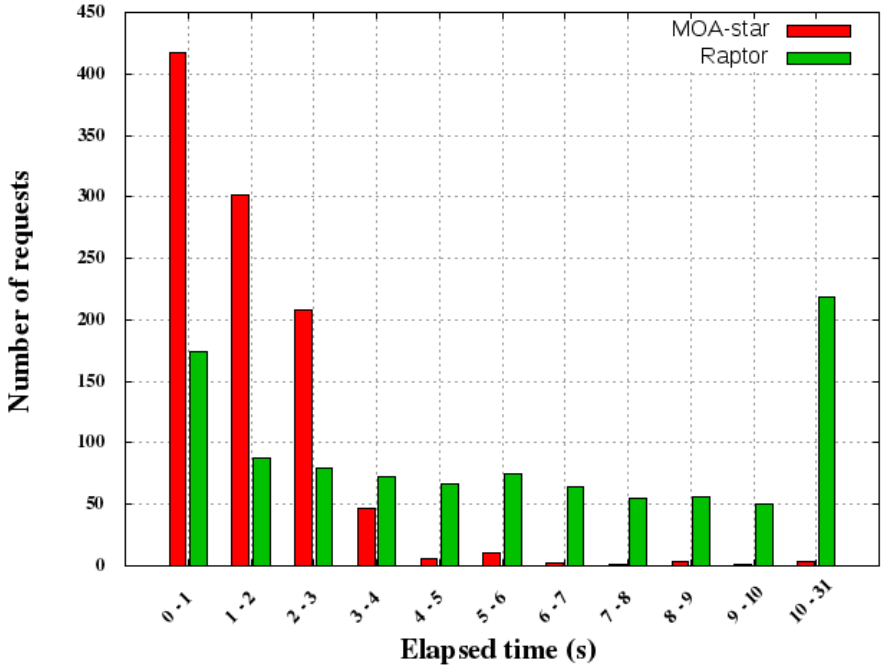


Μέσος χρόνος MOA*: 2.21 sec. Μέσος χρόνος Raptor: 7.39 sec

Σχήμα 4.8: Χρόνοι για την εκτέλεση των δύο αλγορίθμων στα δεδομένα της Αθήνας



Μέσος χρόνος MOA*: 0.70 sec. Μέσος χρόνος Raptor: 1.11 sec
Σχήμα 4.9: Χρόνοι για την εκτέλεση των δύο αλγορίθμων στα δεδομένα του Παρισιού

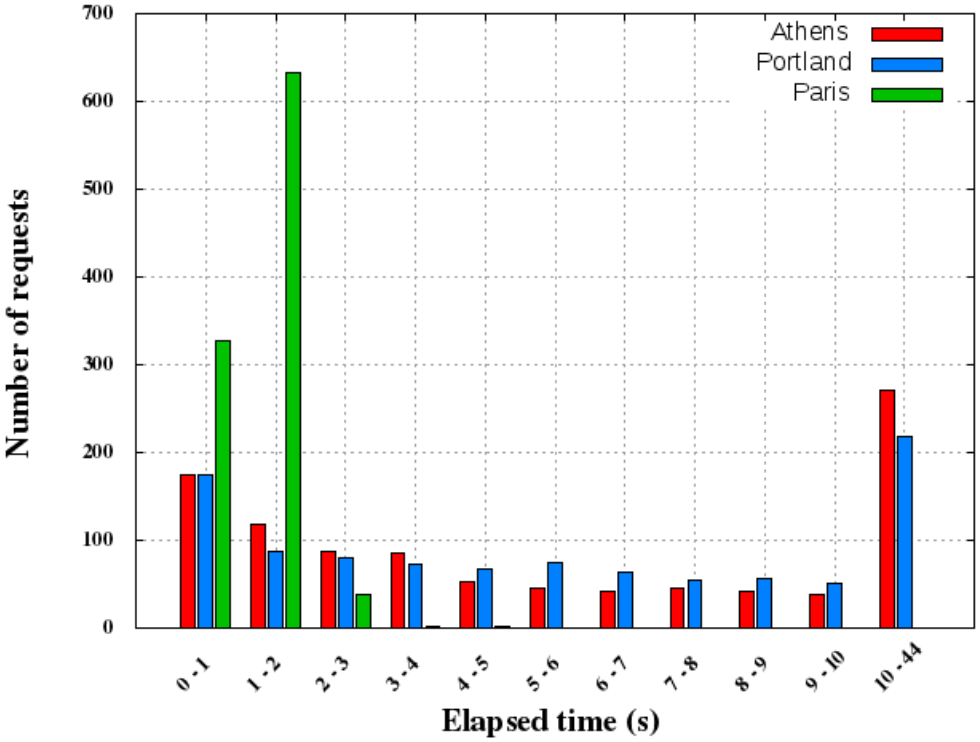


Μέσος χρόνος MOA*: 1.48 sec. Μέσος χρόνος Raptor: 6.29 sec
Σχήμα 4.10: Χρόνοι για την εκτέλεση των δύο αλγορίθμων στα δεδομένα του Πόρτλαντ

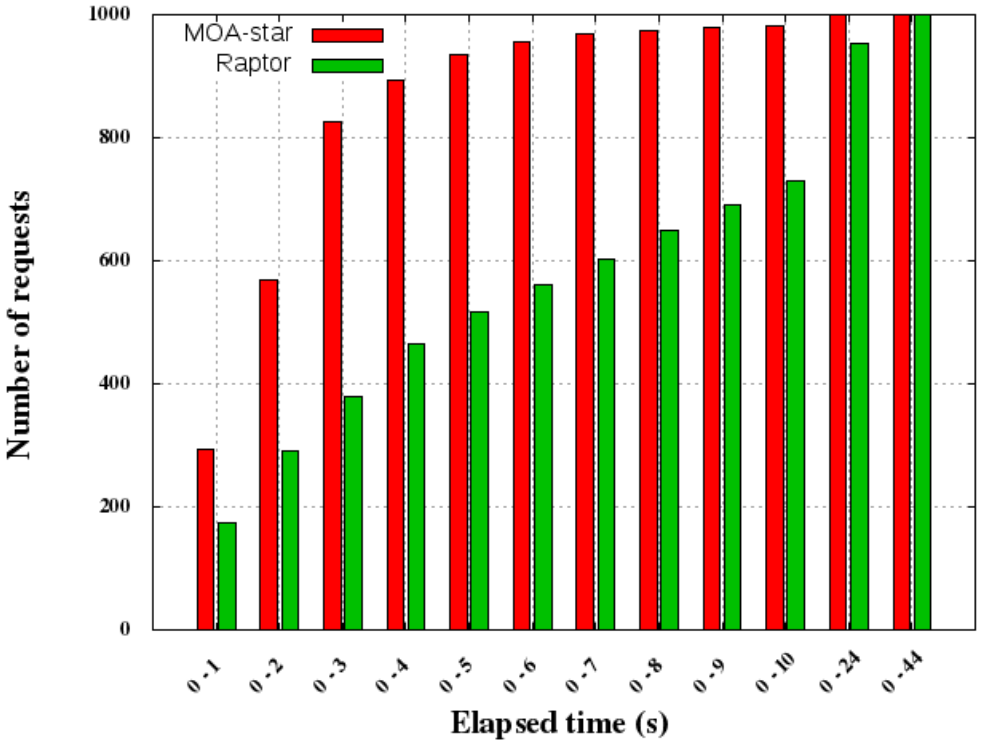
4.4.3 Συμπεράσματα

Κωδικοποίηση απάντησης

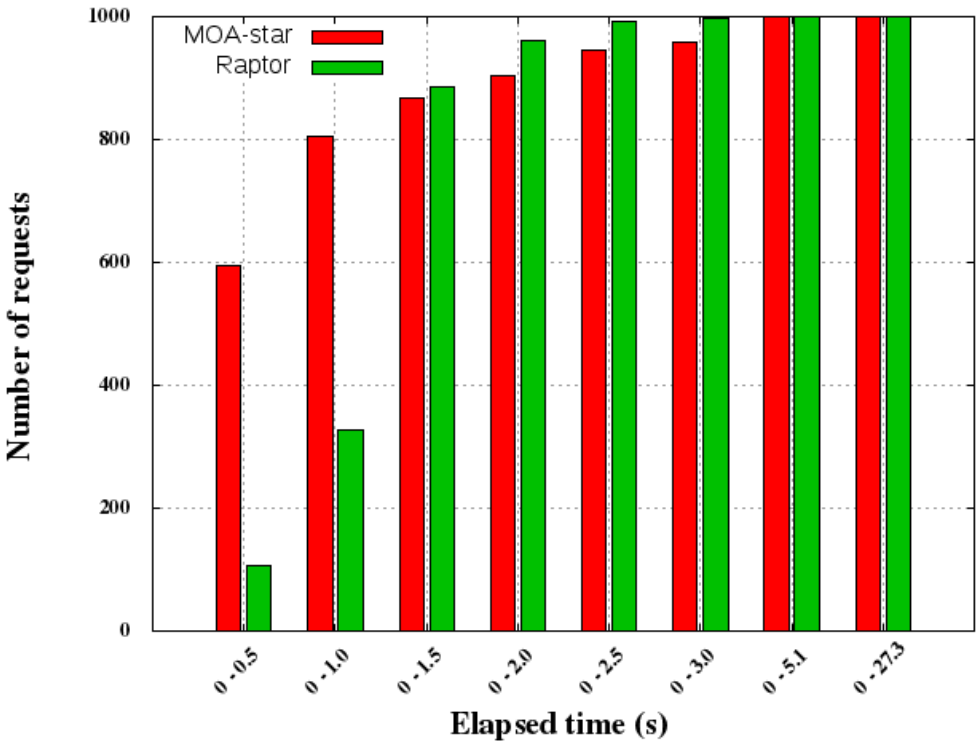
Συγκρίνοντας την καθυστέρηση μεταξύ των δύο μορφών απάντησης (JSON και XML) παρατηρούμε ότι κατά βάση η απάντηση JSON έχει μικρότερο μέσο χρόνο κάτι το οποίο οφείλεται



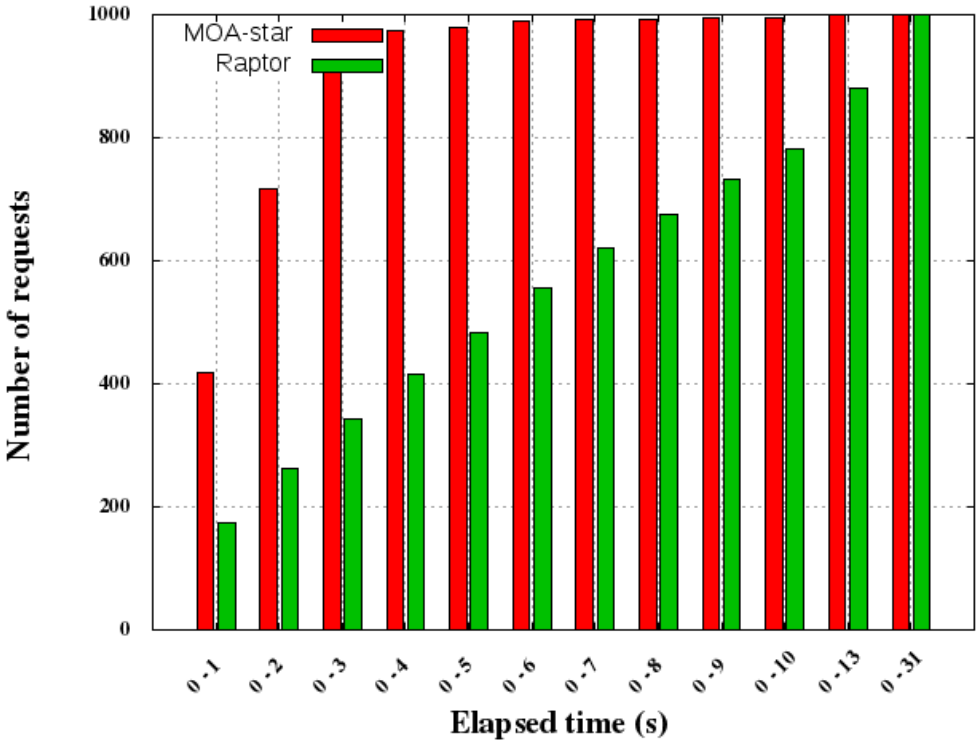
Σχήμα 4.11: Σύγκριση χρόνων εκτέλεσης του Raptor για τις 3 πόλεις



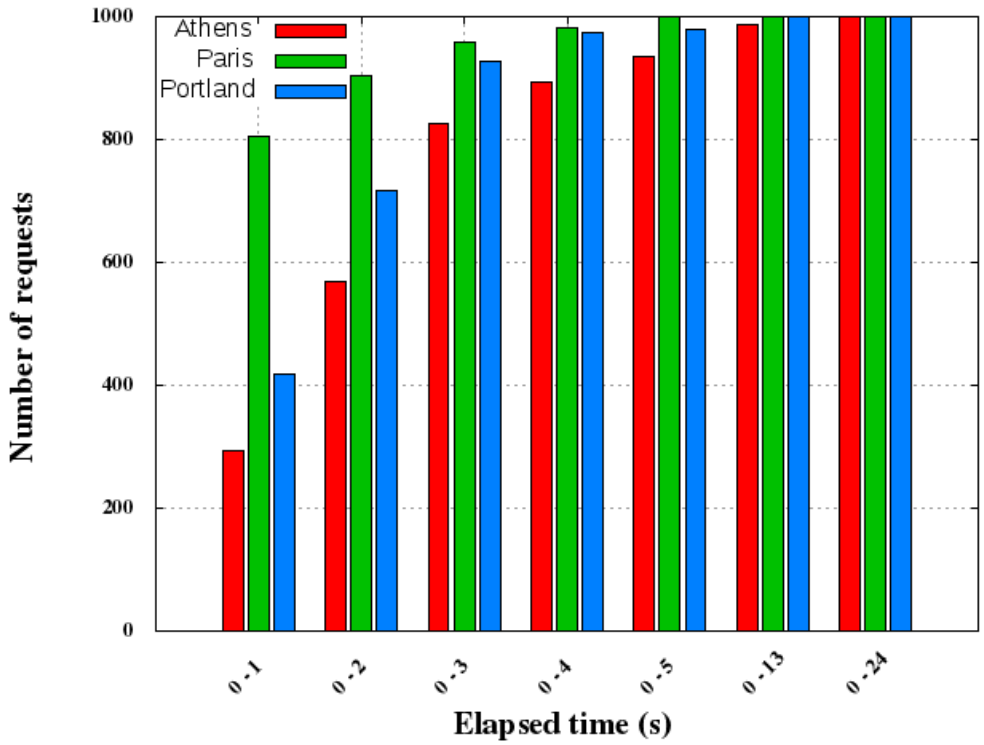
Σχήμα 4.12: Αθροιστικό διάγραμμα χρόνων για την Αθήνα



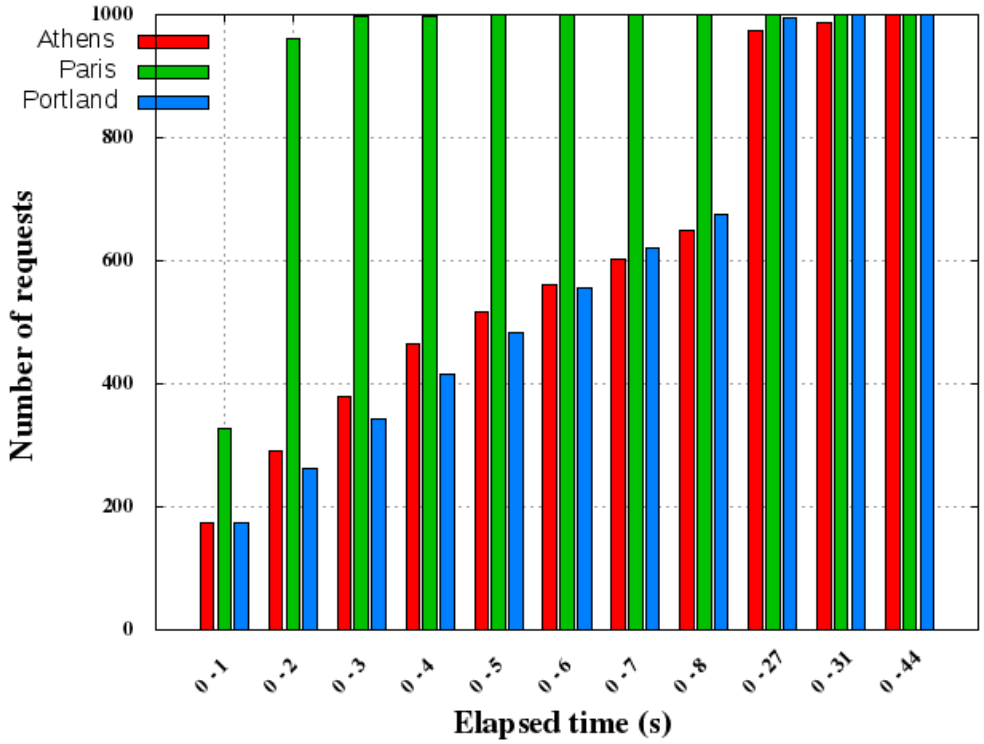
Σχήμα 4.13: Αθροιστικό διάγραμμα χρόνων για το Παρίσι



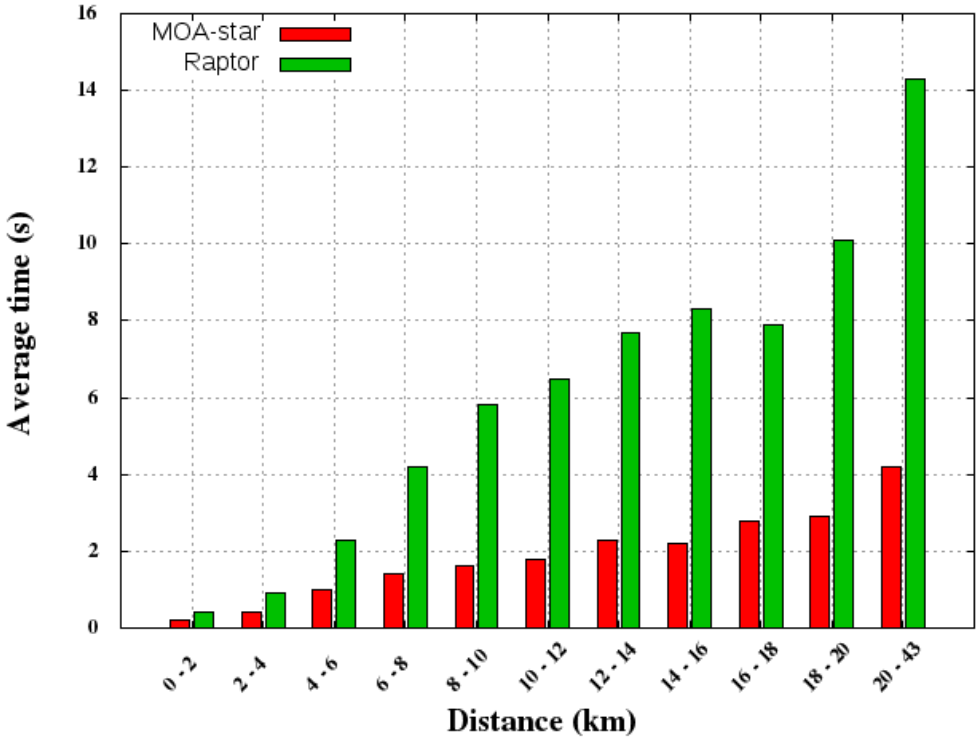
Σχήμα 4.14: Αθροιστικό διάγραμμα χρόνων για το Πόρτλαντ



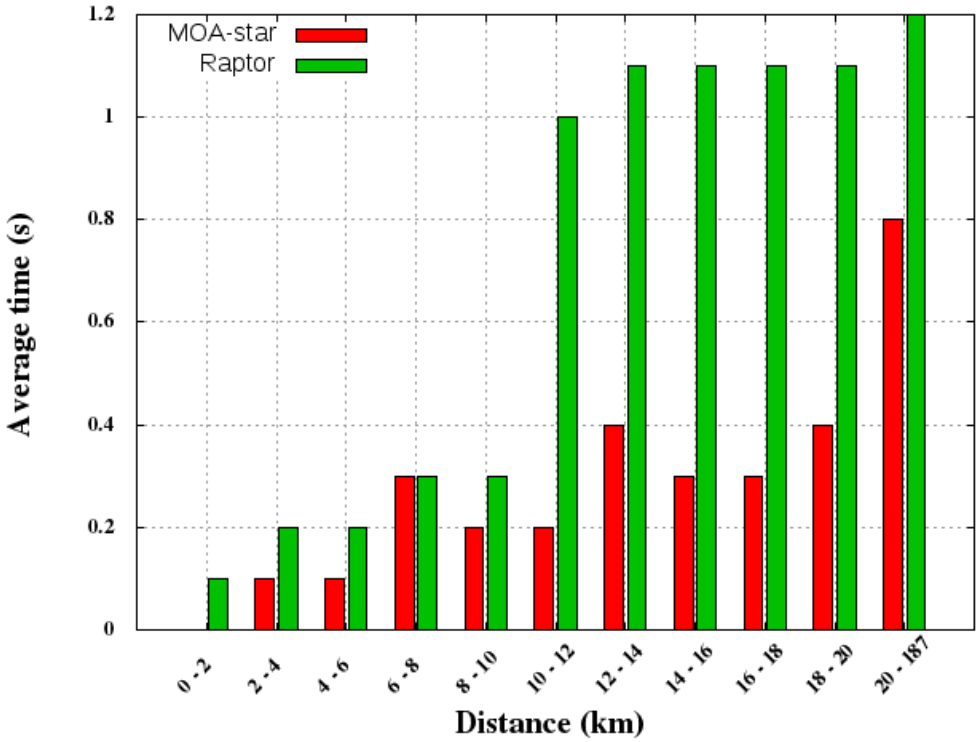
Σχήμα 4.15: Αθροιστικό διάγραμμα χρόνων για την εκτέλεση του MOA* στις 3 πόλεις



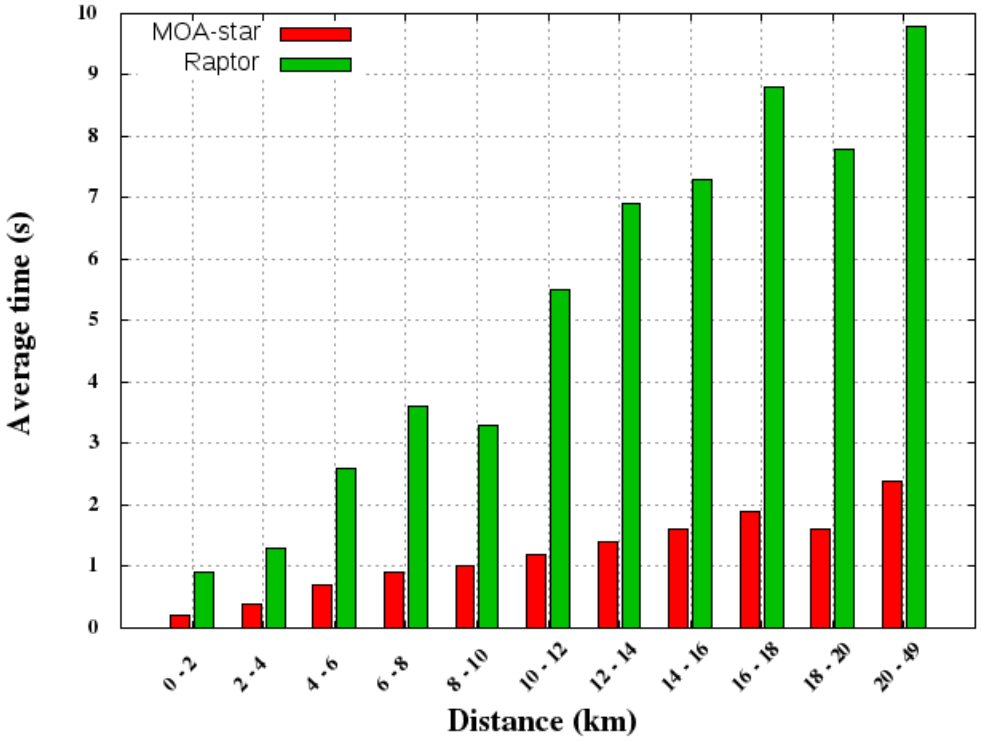
Σχήμα 4.16: Αθροιστικό διάγραμμα χρόνων για την εκτέλεση του Raptor στις 3 πόλεις



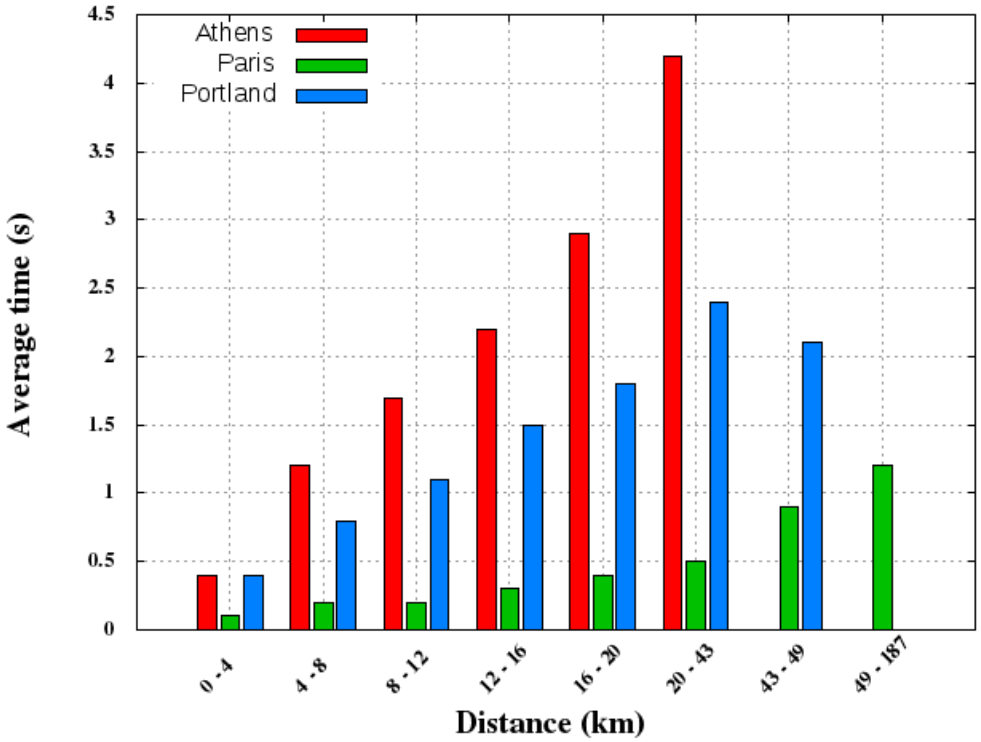
Σχήμα 4.17: Διάγραμμα χρόνων συναρτήσει των αποστάσεων για την Αθήνα



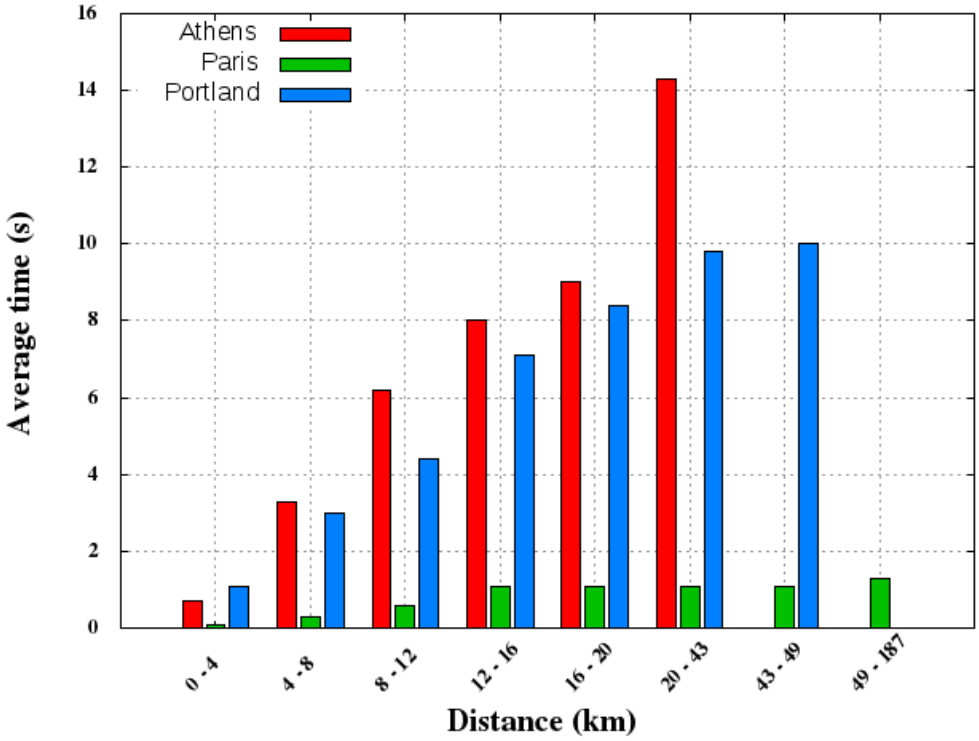
Σχήμα 4.18: Διάγραμμα χρόνων συναρτήσει των αποστάσεων για το Παρίσι



Σχήμα 4.19: Διάγραμμα χρόνων συναρτήσει των αποστάσεων για το Πόρτλαντ



Σχήμα 4.20: Διάγραμμα χρόνων συναρτήσει των αποστάσεων για τον αλγόριθμο MOA*



Σχήμα 4.21: Διάγραμμα χρόνων συναρτήσει των αποστάσεων για τον αλγόριθμο Raptor

ίσως στο μικρότερο μέγεθος των αρχείων όταν έχουν μορφή JSON άρα και λιγότερη καθυστέρηση στο δίκτυο κατά την αποστολή. Η συγκεκριμένη διαφορά στο χρόνο βέβαια μπορεί να θεωρηθεί αμελητέα καθώς το μεγαλύτερο μέρος του χρόνου δαπανάται στον υπολογισμό της βέλτιστης δρομολόγησης και όχι στην αποστολή της απάντησης.

Σύγκριση χρόνων για τις πόλεις

Αθήνα

Για την εκτέλεση του αλγορίθμου δρομολόγησης MOA* στην πόλη της Αθήνας παρατηρούμε ότι το μεγαλύτερο πλήθος των απαντήσεων επιστρέφονται σε χρόνο που κυμαίνεται από 1 έως 3 δευτερόλεπτα. Επιπλέον, αναλύοντας το αθροιστικό διάγραμμα των χρόνων απάντησης φαίνεται πως οι μισές απαντήσεις έχουν ληφθεί σε χρόνο μικρότερο των 2 δευτερολέπτων. Οι απαντήσεις αυτές περιμένουμε να αφορούν πιο κοντινές αποστάσεις από τις υπόλοιπες σε ότι αφορά τα σημεία αναχώρησης και άφιξης. Τέλος, μπορούμε να διακρίνουμε ότι ένα μικρό ποσοστό απαντήσεων επιστέφεται σε μεγάλο χρόνο (κοντά στα 10 δευτερόλεπτα). Οι απαντήσεις αυτές αφορούν είτε μεγάλες αποστάσεις των δύο σημείων αρχής και τέλους του ταξιδιού, είτε ταξίδια τα οποία περιλαμβάνουν αρκετές αλλαγές Μέσων λόγω τις μορφολογίας του δικτύου.

Για τον αλγόριθμο δρομολόγησης Raptor για την πόλη της Αθήνας, παρατηρούμε ότι το μεγαλύτερο πλήθος των απαντήσεων επιστρέφονται σε χρόνο που κυμαίνεται έως 8 δευτερόλεπτα. Από το αθροιστικό γράφιμα των χρόνων παρατηρούμε πως οι μισές απαντήσεις έχουν επιστραφεί σε χρόνο 8 δευτερολέπτων ενώ απαιτούνται 44 δευτερόλεπτα για την λήψη

της πιο αργής απάντησης. Οι χρόνοι για την εκτέλεση του αλγορίθμου Raptor είναι σαφώς μεγαλύτεροι από αυτούς που αφορούν στον αλγόριθμο MOA*.

Παρατηρώντας τα γραφίματα των χρόνων συναρτήσει των αποστάσεων των σημείων αφετηρίας και προορισμού φαίνεται ότι πράγματι ο χρόνος απάντησης της εφαρμογής αυξάνεται σχεδόν αναλογικά με την απόσταση των δύο σημείων. Εκτός κάποιων μικρών διαφοροποιήσεων, παρατηρείται ότι όσο αυξάνεται η απόσταση των δύο σημείων τόσο αυξάνεται και ο χρόνος απάντησης της εφαρμογής. Ενδεικτικά, παρατηρούμε πως για τον αλγόριθμο MOA* ο μέσος χρόνος για μια μικρή απόσταση (μέχρι 6 χιλιόμετρα) είναι 1 δευτερόλεπτο ενώ για τη μεγαλύτερη απόσταση (περίπου 40 χιλιόμετρα) είναι 4 δευτερόλεπτα. Παρατηρώντας το αντίστοιχο διάγραμμα για την εκτέλεση του Raptor βλέπουμε πως για την απόσταση των 6 χιλιομέτρων οι χρόνοι κυμαίνονται στα 2.2 δευτερόλεπτα ενώ για 40 χιλιόμετρα οι μέσοι χρόνοι είναι περίπου 14 δευτερόλεπτα. Από τις παραπάνω παρατηρήσεις, επιβεβαιώνουμε ότι οι μεγάλες αποστάσεις των σημείων αφετηρίας και προορισμού επιδρούν σημαντικά στο χρόνο εκτέλεσης του αλγορίθμου δρομολόγησης.

Παρίσι

Παρατηρώντας τους χρόνους εκτέλεσης της εφαρμογής για το Παρίσι, φαίνεται πως το μεγαλύτερο μέρος των απαντήσεων επιστρέφονται σε χρόνο που κυμαίνεται μέχρι 0.5 δευτερόλεπτα. Αναλύοντας το αθροιστικό γράφιμα χρόνων απάντησης διακρίνουμε πως το μεγαλύτερο μέρος των απαντήσεων έχει επιστραφεί σε χρόνο 0.5 δευτερολέπτων. Για τον Raptor από την άλλη, παρατηρούμε πως οι περισσότερες απαντήσεις έχουν επιστραφεί σε χρόνο 1.5 δευτερολέπτων. Ο αλγόριθμος Raptor και σε αυτή την περίπτωση παρουσιάζει μεγάλη επιβράδυνση σε σχέση με τον MOA* με αρκετά μεγαλύτερη καθυστέρηση με την αύξηση των αποστάσεων.

Πόρτλαντ

Για το Πόρτλαντ παρατηρείται πως το μεγαλύτερο πλήθος των απαντήσεων αντιστοιχεί στο εύρος χρόνων 1 με 2 δευτερόλεπτα ενώ το μεγαλύτερο ποσοστό των απαντήσεων είχε ληφθεί σε χρόνο 2 δευτερολέπτων. Ο αλγόριθμος του Raptor παρουσιάζει και σε αυτή την πόλη επιβράδυνση με χρόνο 5 δευτερολέπτων για τη λήψη των μισών απαντήσεων.

Συγκρίνοντας τους χρόνους μεταξύ των τριών διαφορετικών πόλεων παρατηρούμε ότι η Αθήνα φαίνεται να έχει την μεγαλύτερη καθυστέρηση σε χρόνους και το Παρίσι τους μικρότερους. Η Αθήνα και το Πόρτλαντ φαίνεται να έχουν παρόμοιους χρόνους απάντησης και αυτό οφείλεται στην ομοιότητα των γράφων με τα δεδομένα εισόδου.

Παρατηρούμε επίσης, πως αν και το Παρίσι έχει αρκετά μεγαλύτερο γράφο δεδομένων εισόδου και μεγαλύτερο σε έκταση δίκτυο Μέσων Μαζικής μεταφοράς, ο χρόνος εκτέλεσης των αλγορίθμων δρομολόγησης είναι μικρότερος. Αυτό συμβαίνει λόγω της πολυπλοκότητας του δικτύου των Μέσων του Παρισιού καθώς όπως αναφέρθηκε και στο δεύτερο κεφάλαιο, αποτελείται μόνο από διαδρομές με τρένα και περπάτημα.

Συμπεραίνουμε λοιπόν, πως εκτός από το εύρος του δικτύου, μεγάλης σημασίας για την ταχύτητα εκτέλεσης των αλγορίθμων δρομολόγησης με πολλαπλά Μέσα είναι και ο αριθμός των διαφορετικών τύπων Μέσων που χρησιμοποιούνται. Κάτι τέτοιο είναι λογικό και αποδεικνύει τη σημαντικότητα του αριθμού των πιθανών μετεπιβιβάσεων σε ένα δίκτυο. Δείχνει δηλαδή, ότι η δρομολόγηση σε ένα δίκτυο με λιγότερες μετεπιβιβάσεις, όπως για παράδειγμα στο Παρίσι όπου οι μετεπιβιβάσεις αφορούν τις γραμμές του ίδιου Μέσου είναι ευκολότερη

από κάποιο με περισσότερες επιλογές ανά στάση, ακόμα και στην περίπτωση που το δεύτερο έχει μικρότερες αποστάσεις. Περισσότερες λεπτομέρειες πάνω στα χαρακτηριστικά και τη μορφολογία κάθε δικτύου Μέσων Μαζικής Μεταφοράς θα αναλυθούν στο κεφάλαιο 5.

Σύγκριση αλγορίθμων

Συγκρίνοντας τον αλγόριθμο MOA* με τον αλγόριθμο Raptor παρατηρούμε ότι ο Raptor είναι πολύ πιο αργός από τον MOA* σε όλες τις περιπτώσεις. Κάτι τέτοιο ήταν αναμενόμενο καθώς ο Raptor παρουσιάζει επιτάχυνση μόνο σε πολύ μεγάλα δίκτυα (όπως της Νέας Υόρκης) και με τη χρήση πολλαπλών επεξεργαστών. Επιπλέον, η ίδια η εφαρμογή δεν ευνοεί την υλοποίηση της παραλληλοποίησης του αλγορίθμου καθώς έχει αναπτυχθεί σε γλώσσα προγραμματισμού Java η οποία δεν παρέχει εξειδικευμένες βιβλιοθήκες για την παραλληλοποίηση αλγορίθμων.

Σε αυτό το κεφάλαιο αναλύθηκε το εργαλείο γραμμής εντολών otp-client που αναπτύχθηκε για την εκτέλεση πειραμάτων στην προγραμματιστική διεπαφή του OpenTripPlanner. Αφότου αναλύθηκε η δομή του και οι λειτουργίες του, παρουσιάστηκαν τα αποτελέσματα των μετρήσεων των χρόνων που πραγματοποιήθηκαν με τη βοήθειά του. Στο κεφάλαιο που ακολουθεί, περιγράφεται μια επέκταση της διαδικασίας εκτέλεσης πειραμάτων που πραγματοποιήθηκε στο παρόν κεφάλαιο. Συγκεκριμένα, αναλύεται η διαδικασία εκτέλεσης πειραμάτων όχι μόνο για ένα χρήστη αλλά για πολλούς ταυτόχρονους χρήστες δηλαδή με προσομοίωση ταυτόχρονου φορτίου.

Κεφάλαιο 5

Εκτέλεση πειραμάτων προσομοίωσης ταυτόχρονου φορτίου

Στο προηγούμενο κεφάλαιο περιγράφεται η εκτέλεση πειραμάτων στον εξυπηρετητή που εκτελείται η εφαρμογή OpenTripPlanner με τη βοήθεια του εργαλείου γραμμής εντολών otp-client που αναπτύχθηκε. Με τον τρόπο αυτό έγινε έλεγχος των αποτελεσμάτων που επιστρέφονται από την εφαρμογή τόσο ως προς την ορθότητα όσο και ως προς το χρόνο απάντησης σε κάθε διαφορετική περίπτωση. Ο χρόνος απάντησης που μετράται όμως αναφέρεται σε ένα μεμονωμένο χρήστη και δεν περιλαμβάνει τις περιπτώσεις πολλών ταυτόχρονων αιτημάτων από πολλούς και διαφορετικούς χρήστες. Η μέτρηση του συγκεκριμένου χρόνου είναι αρκετά σημαντική και αντιπροσωπευτική για την απόκριση μιας διαδικτυακής εφαρμογής καθώς το πιο πιθανό σενάριο χρήσης της είναι ακριβώς αυτό, δηλαδή η αποδοχή αιτημάτων από πολλούς ταυτόχρονους χρήστες. Για το λόγο αυτό εκτελέστηκαν πειράματα μέτρησης της απόκρισης της εφαρμογής προσομοιώνοντας μεγάλο αριθμό ταυτόχρονων χρηστών. Στο παρόν κεφάλαιο περιγράφεται η η διαδικασία που ακολουθήθηκε για την εκτέλεση πειραμάτων προσομοίωσης ταυτόχρονου φορτίου στην προγραμματιστική διεπαφή του OpenTripPlanner. Τέλος, αναλύονται τα αποτελέσματα που λάβαμε από την εκτέλεση των πειραμάτων για σύγκριση των χρόνων και διεξαγωγή συμπερασμάτων.

Για την εκτέλεση των συγκεκριμένων πειραμάτων έγινε χρήση του εργαλείου Apache JMeter το οποίο έχει αναπτυχθεί για το σκοπό αυτό, προσομοιώνει δηλαδή πολλά ταυτόχρονα αιτήματα στον εξυπηρετητή που θέλουμε να ελέγξει και μετράει διάφορα στατιστικά που αντιπροσωπεύουν την απόκριση του κάτω από ένα τέτοιο φορτίο. Στα πλαίσια της παρούσας διπλωματικής εκτελέστηκαν πειράματα στη διεπαφή του εργαλείου OpenTripPlanner που αφορά στις τρεις πόλεις, Αθήνα, Παρίσι και Πόρτλαντ. Τα αιτήματα που εκτελέστηκαν ήταν τύπου HTTP Request με τη μέθοδο GET και για την δημιουργία των αιτημάτων με τις σωστές παραμέτρους για κάθε πόλη χρησιμοποιήθηκε το εργαλείο otp-client που παρουσιάσθηκε στο προηγούμενο κεφάλαιο. Στον παρόν κεφάλαιο αναλύεται ο τρόπος χρήσης του εργαλείου για τη διεξαγωγή των πειραμάτων καθώς και τα αποτελέσματα εκτέλεσης αυτών.

5.1 Εισαγωγή στο εργαλείο Apache JMeter

Η εφαρμογή Apache JMeter είναι ένα εργαλείο ανοικτού κώδικα το οποίο έχει αναπτυχθεί στη γλώσσα προγραμματισμού Java και έχει σχεδιαστεί για τον έλεγχο εφαρμογών κάτω από βαρύ δοκιμαστικό φορτίο και για τη μέτρηση της απόδοσής τους. Ξεκίνησε να αναπτύσσεται από την Apache Software Foundation και μέχρι σήμερα έχει καταφέρει να γίνει ένα από τα σημαντικότερα εργαλεία για τον έλεγχο της απόκρισης μεγάλων εφαρμογών σε βαρύ ταυτόχρονο φορτίο.

Η εφαρμογή μπορεί να χρησιμοποιηθεί για τον έλεγχο της απόδοσης που αφορά τόσο σε στατικούς όσο και σε δυναμικούς πόρους, όπως υπηρεσίες διαδικτύου (SOAP/REST), δυναμικές γλώσσες προγραμματισμού (PHP, Java, ASP.NET), βάσεις δεδομένων, FTP εξυπηρετητές και πολλές ακόμα υπηρεσίες. Μπορεί να χρησιμοποιηθεί για την προσομοίωση βαρέων ταυτόχρονων φορτίων σε ένα εξυπηρετητή, ένα σύνολο εξυπηρετητών ή ένα δίκτυο ώστε να εξεταστεί η απόδοσή του πόρου τόσο γενικά όσο και ως προς συγκεκριμένους τύπου φορτίων. Επιπλέον, το εργαλείο προσφέρει τη δυνατότητα γραφικής οπτικοποίησης της απόδοσης του συγκεκριμένου πόρου που θέλουμε να ελέγξουμε για καλύτερη αναπαράσταση και ανάλυση των αποτελεσμάτων.

Το εργαλείο περιλαμβάνει τις εξής δυνατότητες:

- Δυνατότητα εκτέλεσης περαμάτων και ελέγχων σε εξυπηρετητές που υλοποιούν τους εξής τύπους πρωτοκόλλων:
  - Web - HTTP, HTTPS
  - SOAP / REST
  - FTP
  - Database via JDBC
  - LDAP
  - Message-oriented middleware (MOM) via JMS
  - Mail - SMTP(S), POP3(S) and IMAP(S)
  - MongoDB (NoSQL)
  - Native commands or shell scripts
  - TCP
- Πλήρη φορητότητα και δυνατότητα εκτέλεσης σε οποιοδήποτε λειτουργικό σύστημα καθώς είναι αναπτυγμένο εξολοκλήρου στη γλώσσα Java.
- Για συστήματα με δυνατότητα εκτέλεσης πολλών νημάτων (threads) επιτρέπεται η ταυτόχρονη δειγματοληψία από διαφορετικά νήματα τα οποία μπορούν να επιτελούν διαφορετικές λειτουργίες σε ομάδες.
- Προσεκτικά σχεδιασμένο γραφικό περιβάλλον το οποίο συντελεί σε ταχύτερο σχεδιασμό των πειραμάτων και αποσφαλμάτωση αυτών.
- Δυνατότητα αποθήκευσης των αποτελεσμάτων για την επανεξέταση και περαιτέρω ανάλυσή τους.

- Δυνατότητες περαιτέρω επέκτασης του πυρήνα του εργαλείου, όπως επέκταση των τύπων των στατιστικών που μπορούν να συλλεχθούν και προσαρμογής του τρόπου αναπαράστασης των αποτελεσμάτων στις προτιμήσεις κάθε χρήστη.

5.2 Σχεδιασμός πειράματος προσομοίωσης

Για το σχεδιασμό ενός πειράματος χρησιμοποιούνται κάποια βασικά στοιχεία τα οποία συνιστούν κάθε πείραμα. Όσο αφορά τα πειράματα σε διαδικτυακές εφαρμογές τα πιο βασικά στοιχεία που τα απαρτίζουν είναι τα εξής:

5.2.1 Thread Group

Τα στοιχεία Thread Group είναι τα αρχικά συστατικά οποιουδήποτε πειράματος. Τα στοιχεία Controllers και Samplers πρέπει να τοποθετηθούν κάτω από ένα Thread Group. Όπως φαίνεται και από το όνομα, το στοιχείο αυτό είναι υπεύθυνο για τα νήματα (threads) τα οποία θα δημιουργηθούν από το εργαλείο κατά τη διάρκεια της εκτέλεσης ενός πειράματος. Οι ρυθμίσεις ενός Thread Group επιτρέπουν στο χρήστη:

- Την επιλογή του αριθμού νημάτων
- Τον ορισμό της περιόδου εκκίνησης (ramp-up period) για τα νήματα
- Τον ορισμό του αριθμού των επαναλήψεων για το πείραμα

Κάθε νήμα θα εκτελέσει το πείραμα εντελώς ανεξάρτητα από τα υπόλοιπα νήματα. Ουσιαστικά τα νήματα χρησιμοποιούνται για την προσομοίωση ταυτόχρονων συνδέσεων στον εξυπηρετητή της εφαρμογής.

Η περίοδος εκκίνησης (ramp-up period) τίθεται από τον χρήστη και ορίζει το χρόνο μέσα στον οποίο θα δημιουργηθούν όλα τα νήματα που έχουν επιλεχθεί. Για παράδειγμα, αν έχει επιλεχθεί η δημιουργία 10 νημάτων και έχει οριστεί περίοδος εκκίνησης 100 δευτερολέπτων, τότε το Apache JMeter θα πρέπει να δημιουργήσει 10 νήματα σε 100 δευτερόλεπτα. Έτσι, κάθε νήμα θα ξεκινά 10 (100/10) δευτερόλεπτα μετά από το προηγούμενο, δηλαδή ο χρόνος διαμεσολάβησης ανάμεσα στη δημιουργία των 10 νημάτων θα είναι 10 δευτερόλεπτα. Η περίοδος εκκίνησης θα πρέπει να είναι αρκετά μεγάλη ώστε να αποφεύγεται ο πολύ μεγάλος φόρτος εργασίας στην αρχή του πειράματος αλλά και αρκετά μικρό ώστε να μην έχει τελειώσει την εκτέλεσή του το προηγούμενο νήμα πριν προλάβει να δημιουργηθεί το επόμενο. Μια προτεινόμενη τακτική είναι η εκκίνηση του πειράματος με επιλογή της περιόδου εκκίνησης ίσης με τον αριθμό των νημάτων και μετέπειτα προσαρμογής της.

5.2.2 Controllers

Οι Controllers είναι υπεύθυνοι για την διαδικασία εκτέλεσης του πειράματος καθώς και για τη ροή του. Το Jmeter έχει δύο τύπους Controllers, τους Samplers και τους Logical Controllers.

Samplers

Οι Samplers καθορίζουν τις αιτήσεις που θα στείλει το Apache JMeter στον εξυπηρετητή και περιμένουν για την απάντηση. Υπάρχουν αρκετοί τύποι Samplers και περιλαμβάνουν τα εξής είδη αιτημάτων:

- FTP Request
- HTTP Request
- JDBC Request
- Java object request
- LDAP Request
- SOAP/XML-RPC Request
- WebService (SOAP) Request

Κάθε Sampler έχει τη δυνατότητα παραμετροποίησης αρκετών ιδιοτήτων του. Για περαιτέρω παραμετροποίηση χρησιμοποιείται ένα Configuration στοιχείο που προστίθεται στο πείραμα. Επιπλέον, πολλές φορές είναι αναγκαίος ο λογικός έλεγχος της απάντησης που λαμβάνεται ιδιαίτερα σε HTTP Requests καθώς μπορεί η απάντηση να επιστρέψει απάντηση επιτυχίας (OK Status Code) αλλά η ίδια η σελίδα που επιστράφηκε να περιέχει λάθη ή πεδία που λείπουν. Για το λόγο αυτό, υπάρχει η δυνατότητα προσθήκης ενός στοιχείου Assertion στον Sampler το οποίο θα μπορεί να ελέγχει για τυχόν λάθη στην απάντηση μέσω της χρήσης κανονικών εκφράσεων (regular expressions).

HTTP Request Sampler

Ένας πολύ χρήσιμος και ευρέως χρησιμοποιούμενος Sampler είναι ο HTTP Request Sampler ο οποίος επιτρέπει την αποστολή μιας HTTP/HTTPS αίτησης σε ένα διαδικτυακό εξυπηρετητή (web server). Μέσω της απάντησης μπορούν να ανακτηθούν οι ενσωματωμένοι πόροι που μπορεί να είναι εικόνες, applets, stylesheets ή ήχος. Ο προεπιλεγμένος συντακτικός αναλυτής για την απάντηση είναι ο htmlparser ο οποίος μπορεί να αλλάξει από τις ρυθμίσεις.

Επιπλέον, στην περίπτωση που ο χρήστης θέλει να στείλει πολλαπλές αιτήσεις στον ίδιο εξυπηρετητή υπάρχει η δυνατότητα ορισμού ενός στοιχείου HTTP Request Defaults έτσι ώστε να μην είναι αναγκαία η προσθήκη της ίδιας πληροφορίας σε κάθε HTTP Request ξεχωριστά. Στον πίνακα 5.1 φαίνονται οι βασικοί παράμετροι που μπορούν να οριστούν για τον HTTP Request Sampler.

HTTP Request Defaults

Το HTTP Request Defaults αποτελεί ένα βασικό συστατικό των πειραμάτων που περιέχουν HTTP Request Samplers. Χρησιμοποιούνται για την αποφυγή της δημιουργίας πολλαπλών HTTP Request Samplers με τις ίδιες παραμέτρους καθώς οι παράμετροι της αίτησης μπορούν να καθοριστούν στο HTTP Request Defaults στοιχεία και θα κληρονομηθούν αυτόματα από τους αντίστοιχους HTTP Request Samplers.

Παράμετρος	Περιγραφή
Name	Περιγραφικό όνομα για τον Sampler έτσι όπως θα εμφανίζεται στο δέντρο του γραφικού περιβάλλοντος.
Server	Το όνομα ή η IP διεύθυνση του εξυπηρετητή στον οποίο θα γίνει η αίτηση.
Port	Ο αριθμός της θύρας στην οποία "ακούει" ο εξυπηρετητής
Connect Timeout	Ο μέγιστος χρόνος αναμονής για επίτευξη σύνδεσης σε milliseconds.
Response Timeout	Ο μέγιστος χρόνος αναμονής για απάντηση από τον εξυπηρετητή. Το όριο του συγκεκριμένου χρόνου εφαρμόζεται για κάθε ξεχωριστή απάντηση.
Protocol	Το πρωτόκολλο βάση του οποίου εκτελείται το αίτημα ανάμεσα στο HTTP, HTTPS ή FILE με το HTTP να είναι το προεπιλεγμένο.
Method	Η μέθοδος της αίτησης. Οι βασικοί μέθοδοι είναι GET, POST, HEAD, TRACE, OPTIONS, PUT και DELETE.
Path	Η διαδρομή για τον πόρο στον οποίο θέλουμε να εκτελεστεί το αίτημα (για παράδειγμα, /servlets/myServlet). Στην περίπτωση που απαιτούνται επιπλέον παράμετροι για την αίτηση προστίθενται με την επιλογή προσθήκης παραμέτρων που δίνεται από το εργαλείο. Στην ειδική περίπτωση όπου το Path ξεκινάει με "http://" ή "https://" τότε το ίδιο το Path χρησιμοποιείται ως το πλήρες URL.
Send Parameters With the Request	Οι επιπλέον παράμετροι που επιθυμούμε να στείλουμε με την αίτηση. Κάθε παράμετρος περιέχει όνομα και τιμή καθώς και τις επιλογές κωδικοποίησης της. Από την λίστα των παραμέτρων δημιουργείται ένα query string το οποίο σε περίπτωση χρήσης GET ή DELETE μεθόδου προστίθεται στο URL και στην περίπτωση POST ή PUT μεθόδου στέλνεται ξεχωριστά.

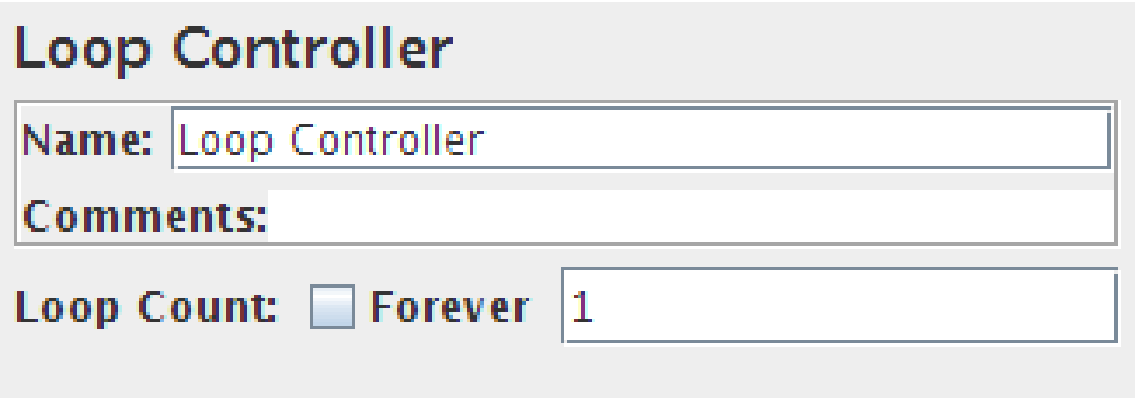
Πίνακας 5.1: Ορισμός των βασικών παραμέτρων που απαρτίζουν τον HTTP Request Sampler

Logic Controllers

Οι Logic Controllers καθορίζουν την λογική την οποία χρησιμοποιεί το Apache JMeter για την αποστολή των αιτημάτων. Δίνουν τη δυνατότητα αλλαγής της σειράς αποστολής των αιτήσεων, το περιεχόμενο τους και τον αριθμό των επαναλήψεων εκτέλεσής τους. Για παράδειγμα, Logic Controller είναι ο Once Only Controller ο οποίος υποδεικνύει στο Apache JMeter να επεξεργαστεί το περιεχόμενο του Controller μόνο μία φορά ανά νήμα, και ο Interleave Controller ο οποίος επιλέγει την εκτέλεση ενός από τα περιεχόμενά του (παράδειγμα, HTTP request) κάθε φορά σειριακά. Δύο άλλοι βασικοί Logic Controllers είναι ο Loop Controller και ο If Controller που εξηγούνται παρακάτω.

Loop Controller

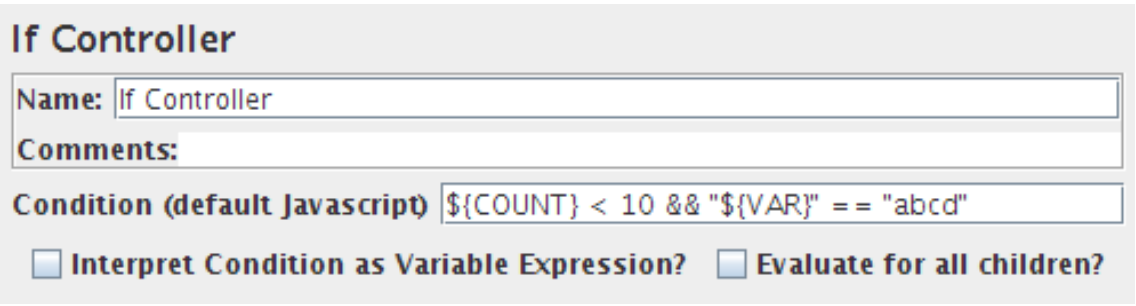
Με την προσθήκη ενός Loop Controller στο πείραμα όπως φαίνεται και στην εικόνα 5.1, το Apache JMeter θα εκτελέσει επαναλήψεις στο εσωτερικό του όσες φορές έχουν προσδιοριστεί. Οι επαναλήψεις αυτές προστίθενται στον αριθμό των επαναλήψεων που έχουν προσδιοριστεί ήδη κατά τη δημιουργία του Thread Group. Για παράδειγμα, αν προστεθεί ένα HTTP Request σε ένα Loop Controller με αριθμό επαναλήψεων ίσο με δύο, ενώ το Thread Group έχει ρυθμιστεί να εκτελεστεί τρεις φορές, τότε το Apache JMeter θα στείλει συνολικά 2 * 3 = 6 HTTP Requests.



Σχήμα 5.1: Παράδειγμα Loop Controller

If Controller

Ένας If Controller όπως φαίνεται και στην εικόνα 5.2 επιτρέπει στο χρήστη να ελέγξει και να καθορίσει αν τα στοιχεία που περιέχει θα εκτελεστούν ή όχι. Η συνθήκη ελέγχου ερμηνεύεται ως κώδικας JavaScript που επιστρέφει "true" ή "false".



Σχήμα 5.2: Παράδειγμα If Controller

5.2.3 Listeners

Οι Listeners παρέχουν πρόσβαση στις πληροφορίες που συλλέγει το Apache JMeter για τα πειράματα που εκτελούνται και δίνουν τη δυνατότητα προβολής, αποθήκευσης και επαναχρησιμοποίησης των αποτελεσμάτων των πειραμάτων. Τα αποτελέσματα ενός πειράματος αποθηκεύονται σε μορφή XML ή CSV . Υπάρχουν πολλών ειδών listeners και διαφέρουν ως προς τον τρόπο αναπαράστασης των αποτελεσμάτων. Εδώ αναφέρονται οι πιο σημαντικοί.

Graph Results

Ο Graph Results Listener δημιουργεί ένα απλό γράφημα που αναπαριστά όλους τους χρόνους των δειγμάτων όπως φαίνεται και στην εικόνα 5.3. Αναπαριστά το πως εξελίσσονται στο χρόνο τα εξής στατιστικά στοιχεία:

- Οι πραγματικές τιμές των χρόνων (Data)
- Ο μέσος όρος των χρόνων

Σχήμα 5.3: Παράδειγμα Graph Results Listener

- Η διάμεσος των χρόνων
- Η τυπική απόκλιση των χρόνων
- Η ρυθμοαπόδοση ως τον αριθμό των δειγμάτων ανά μονάδα χρόνου

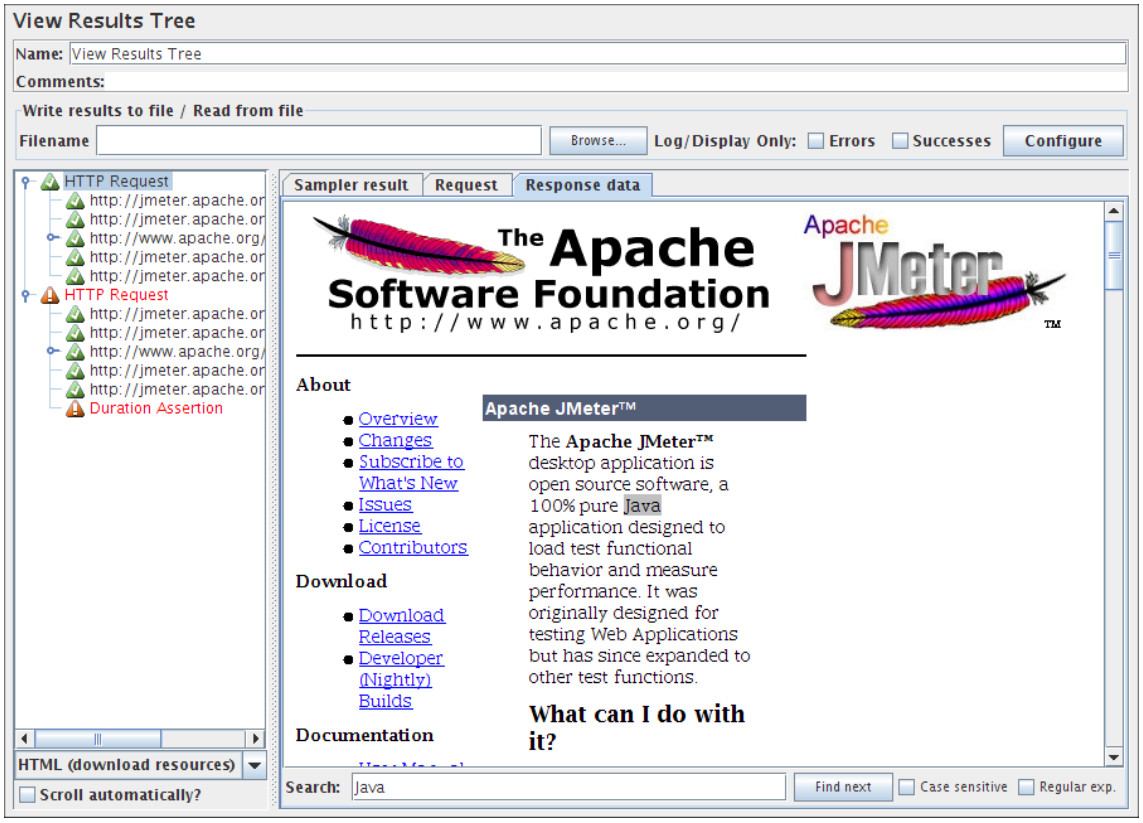
Γενικά, σε πειράματα ελέγχου απόκρισης με μεγάλο φορτίο δεν ενδείκνυται η χρήση του συγκεκριμένου Listener καθώς καταναλώνει πολλούς πόρους (μνήμη και υπολογιστική ισχύ).

View Results Tree

Ο Listener αυτός αναπαριστά τα αποτελέσματα σε μορφή γραφικού δέντρου επιτρέποντας στο χρήστη να εξετάσει ξεχωριστά κάθε απάντηση. Στην εικόνα 5.4 φαίνεται ένα παράδειγμα τέτοιου Listener. Εκτός από την ίδια την απάντηση προβάλλεται και ο χρόνος που μεσολάβησε για να ληφθεί. Επίσης, παρέχεται η δυνατότητα επιλογής του τρόπου εμφάνισης της απάντησης όπως HTML, Text και XML. Λόγω της μεγάλης κατανάλωσης πόρων ο Listener αυτός δεν ενδείκνυται σε πειράματα ελέγχου απόκρισης με μεγάλο φορτίο.

View Results In Table

Στην περίπτωση του συγκεκριμένου Listener δημιουργείται μία γραμμή του πίνακα για κάθε ξεχωριστό αποτέλεσμα. Ένα παράδειγμα τέτοιου πίνακα φαίνεται στην εικόνα 5.5.



Σχήμα 5.4: Παράδειγμα View Results Tree Listener

Response Time Graph

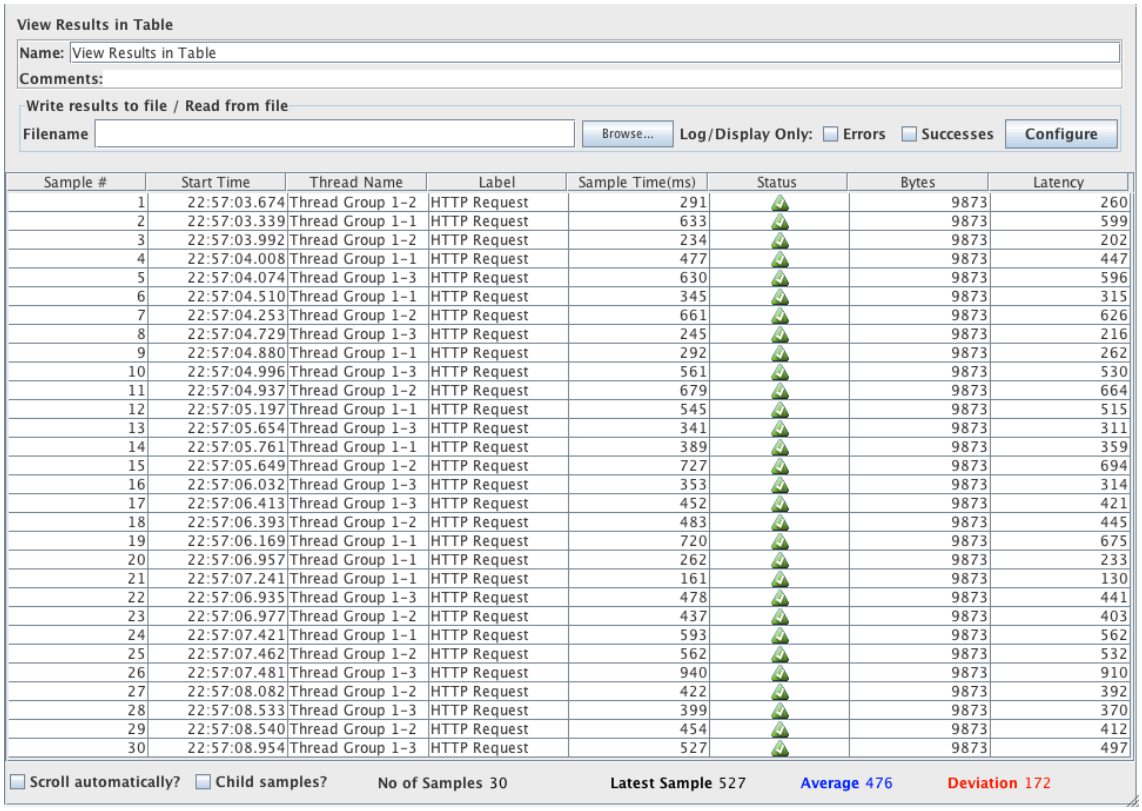
Όπως φαίνεται και στην εικόνα 5.6, ο Response Time Graph σχεδιάζει ένα γραμμικό διάγραμμα που αναπαριστά την εξέλιξη του χρόνου απόκρισης κατά τη διάρκεια εκτέλεσης του πειράματος, για κάθε αίτημα που έχει label.

Aggregate Report

Ο Aggregate Report Listener δημιουργεί μια γραμμή ενός πίνακα για κάθε αίτημα με διαφορετική ονομασία που δημιουργήθηκε στο πείραμα. Για κάθε αίτημα, αθροίζει τις πληροφορίες από την απάντηση και υπολογίζει τον αριθμό αιτημάτων, τον ελάχιστο, μέγιστο και μέσο χρόνο, το ποσοστό σφαλμάτων και τη ρυθμοαπόδοση.

Summary Report

Ο Listener Summary Report είναι παρόμοιος με τον Listener Aggregate Report, καθώς δημιουργεί μια γραμμή ενός πίνακα για κάθε αίτημα, μόνο που καταναλώνει λιγότερη μνήμη. Στον πίνακα 5.2 φαίνονται αναλυτικά τα πεδία που περιέχει και η περιγραφή τους.



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Latency
1	22:57:03.674	Thread Group 1-2	HTTP Request	291		9873	260
2	22:57:03.339	Thread Group 1-1	HTTP Request	633		9873	599
3	22:57:03.992	Thread Group 1-2	HTTP Request	234		9873	202
4	22:57:04.008	Thread Group 1-1	HTTP Request	477		9873	447
5	22:57:04.074	Thread Group 1-3	HTTP Request	630		9873	596
6	22:57:04.510	Thread Group 1-1	HTTP Request	345		9873	315
7	22:57:04.253	Thread Group 1-2	HTTP Request	661		9873	626
8	22:57:04.729	Thread Group 1-3	HTTP Request	245		9873	216
9	22:57:04.880	Thread Group 1-1	HTTP Request	292		9873	262
10	22:57:04.996	Thread Group 1-3	HTTP Request	561		9873	530
11	22:57:04.937	Thread Group 1-2	HTTP Request	679		9873	664
12	22:57:05.197	Thread Group 1-1	HTTP Request	545		9873	515
13	22:57:05.654	Thread Group 1-3	HTTP Request	341		9873	311
14	22:57:05.761	Thread Group 1-1	HTTP Request	389		9873	359
15	22:57:05.649	Thread Group 1-2	HTTP Request	727		9873	694
16	22:57:06.032	Thread Group 1-3	HTTP Request	353		9873	314
17	22:57:06.413	Thread Group 1-3	HTTP Request	452		9873	421
18	22:57:06.393	Thread Group 1-2	HTTP Request	483		9873	445
19	22:57:06.169	Thread Group 1-1	HTTP Request	720		9873	675
20	22:57:06.957	Thread Group 1-1	HTTP Request	262		9873	233
21	22:57:07.241	Thread Group 1-1	HTTP Request	161		9873	130
22	22:57:06.935	Thread Group 1-3	HTTP Request	478		9873	441
23	22:57:06.977	Thread Group 1-2	HTTP Request	437		9873	403
24	22:57:07.421	Thread Group 1-1	HTTP Request	593		9873	562
25	22:57:07.462	Thread Group 1-2	HTTP Request	562		9873	532
26	22:57:07.481	Thread Group 1-3	HTTP Request	940		9873	910
27	22:57:08.082	Thread Group 1-2	HTTP Request	422		9873	392
28	22:57:08.533	Thread Group 1-3	HTTP Request	399		9873	370
29	22:57:08.540	Thread Group 1-2	HTTP Request	454		9873	412
30	22:57:08.954	Thread Group 1-3	HTTP Request	527		9873	497

Σχήμα 5.5: Παράδειγμα View Results in Table Listener

Παράμετρος	**Περιγραφή**
Label	Μια ετικέτα που χαρακτηρίζει το δείγμα.
# Samples	Ο αριθμός των samples με το ίδιο Label.
Average	Ο μέσος χρόνος αναμονής για την λήψη μιας απάντησης για ένα σύνολο αποτελεσμάτων.
Min	Ο ελάχιστος χρόνος αναμονής για ένα σύνολο αποτελεσμάτων.
Max	Ο μέγιστος χρόνος αναμονής για ένα σύνολο αποτελεσμάτων.
Std. Dev.	Η τυπική απόκλιση του χρόνου αναμονής.
Error %	Το ποσοστό των αιτημάτων με errors.
Throughput	Αριθμός αιτημάτων ανά μονάδα χρόνου.
Avg. Bytes	Μέσο μέγεθος της απάντησης σε bytes.

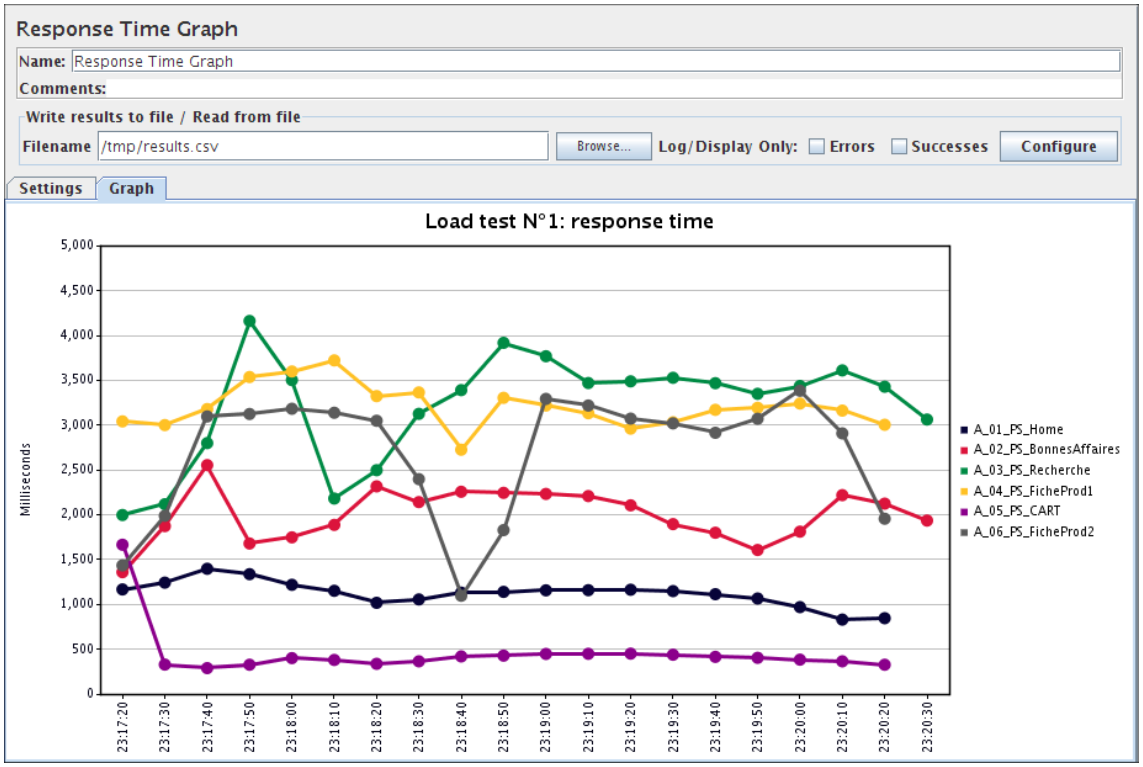
Πίνακας 5.2: Ορισμός των πεδίων του Summary Report Listener

Aggregate Graph

Ο Listener αυτός είναι όμοιος με τον Aggregate Report με τη μόνη διαφορά ότι ο Aggregate Graph παράγει ένα γράφημα σε μορφή μπάρας από τα αποτελέσματα και μπορεί να αποθηκευτεί ως ένα αρχείο εικόνας. Ένα παράδειγμα τέτοιου Listener φαίνεται στην εικόνα 5.7.

5.2.4 Timers

Καθώς ένα νήμα του Apache JMeter στέλνει αιτήματα χωρίς τη μεσολάβηση ενός διαστήματος ανάμεσα σε αυτά, συστήνεται ο προσδιορισμός μιας καθυστέρησης με την προσθήκη ενός



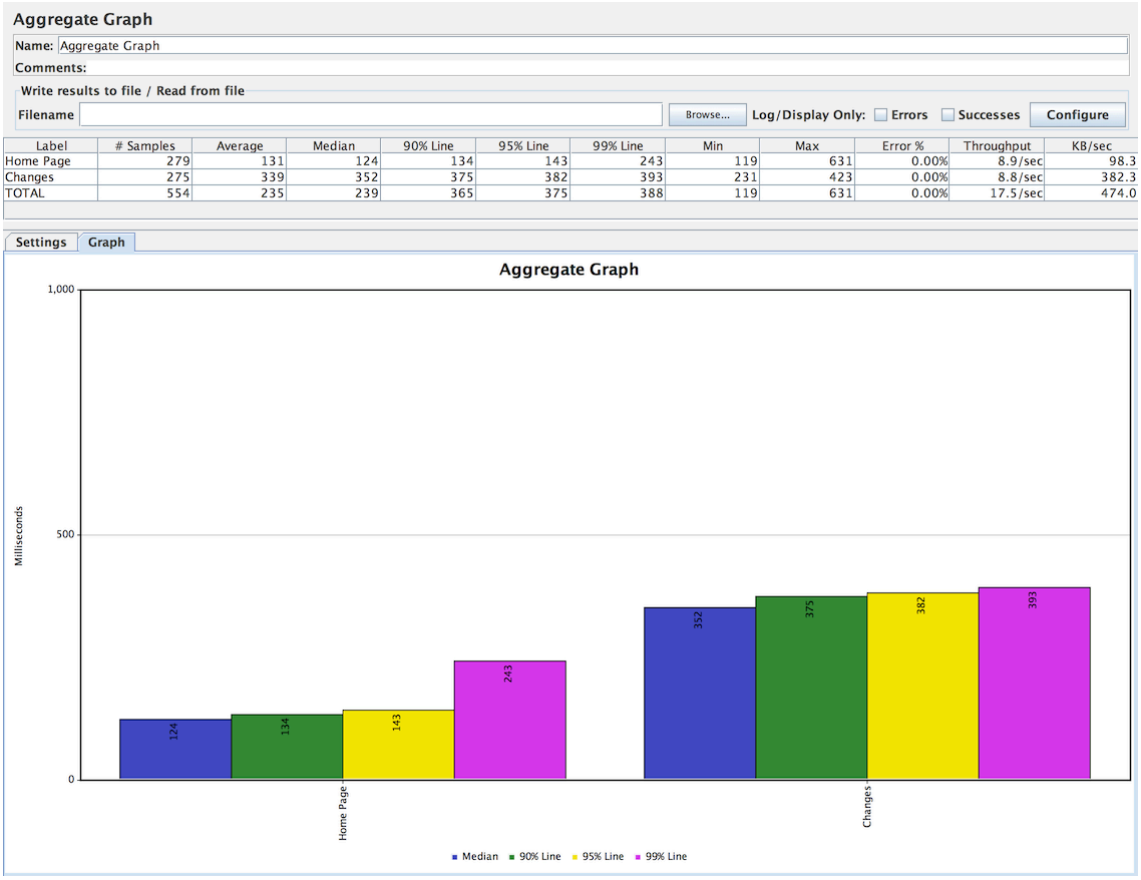
Σχήμα 5.6: Παράδειγμα Response Time Graph Listener

Timer στο Thread Group. Αν παραληφθεί ο Timer υπάρχει περίπτωση το Apache JMeter να κατακλύσει τον εξυπηρετητή με πολλαπλά αιτήματα σε μικρό χρονικό διάστημα.

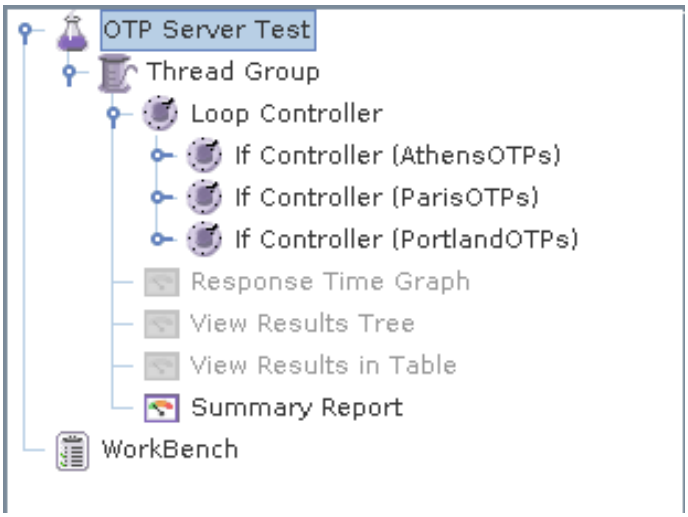
5.3 Διαδικασία εκτέλεσης πειραμάτων

Για τις ανάγκες της διπλωματικής εκτελέστηκαν πειράματα προσομοίωσης πολλών ταυτόχρονων συνδέσεων στο API του OpenTripPlanner για τις τρεις πόλεις για τις οποίες εκτελείται δηλαδή για την Αθήνα, το Παρίσι και το Πόρτλαντ. Τα αιτήματα που έγιναν στον εξυπηρετητή ήταν HTTP Requests με τη μέθοδο GET. Όπως αναφέρθηκε και στην περιγραφή των παραμέτρων που δέχεται το εργαλείο Apache JMeter, όταν γίνεται χρήση της μεθόδου GET μπορεί να δοθεί ως τιμή της παραμέτρου Path του HTTP Request το πλήρες URL του αιτήματος που περιλαμβάνει και τις ειδικές παραμέτρους. Για την εύρεση των σωστών URLs που ανταποκρίνονται σε κάθε πόλη χρησιμοποιήθηκε η λειτουργία εξαγωγής URLs του otp-client εργαλείου. Τα URL αυτά εισήχθησαν στο Apache JMeter για την επίτευξη των πειραμάτων της κάθε πόλης. Αναλυτικά, η διαδικασία που ακολουθήθηκε περιγράφεται παρακάτω.

Η γενικότερη δομή κάθε πειράματος που εκτελέστηκε φαίνεται στην εικόνα 5.8 και είναι η εξής:



Σχήμα 5.7: Παράδειγμα Aggregate Graph Listener



Σχήμα 5.8: Βασική δομή των πειραμάτων που εκτελέστηκαν στο Apache JMeter

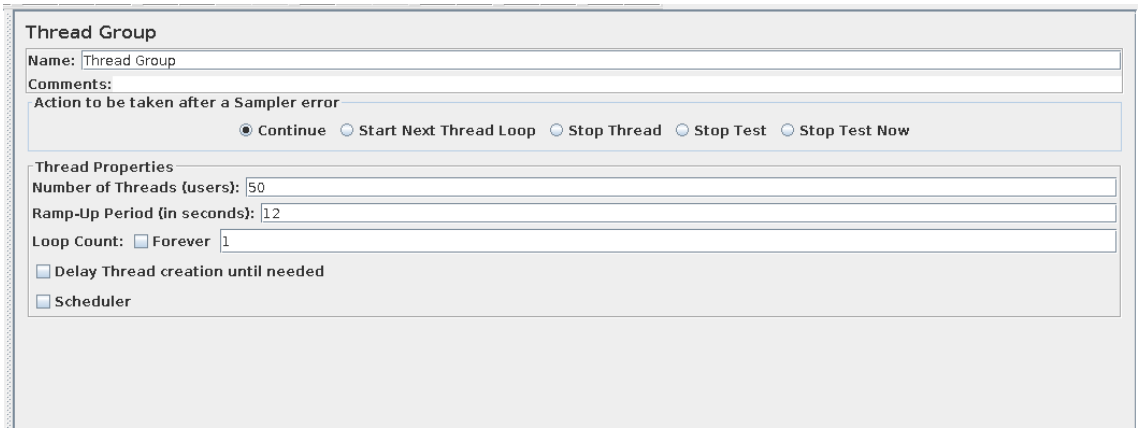
- Δημιουργία ενός Thread Group που περιέχει τον αριθμό των χρηστών που θέλουμε να προσομοιώσουμε.
- Τοποθέτηση των Samplers που στη συγκεκριμένη περίπτωση είναι HTTP Requests Samplers.

- Τοποθέτηση κατάλληλων Listeners για την μελέτη των αποτελεσμάτων.

Τα επιμέρους αυτά βήματα περιγράφονται αναλυτικά παρακάτω.

5.3.1 Δημιουργία Thread Group

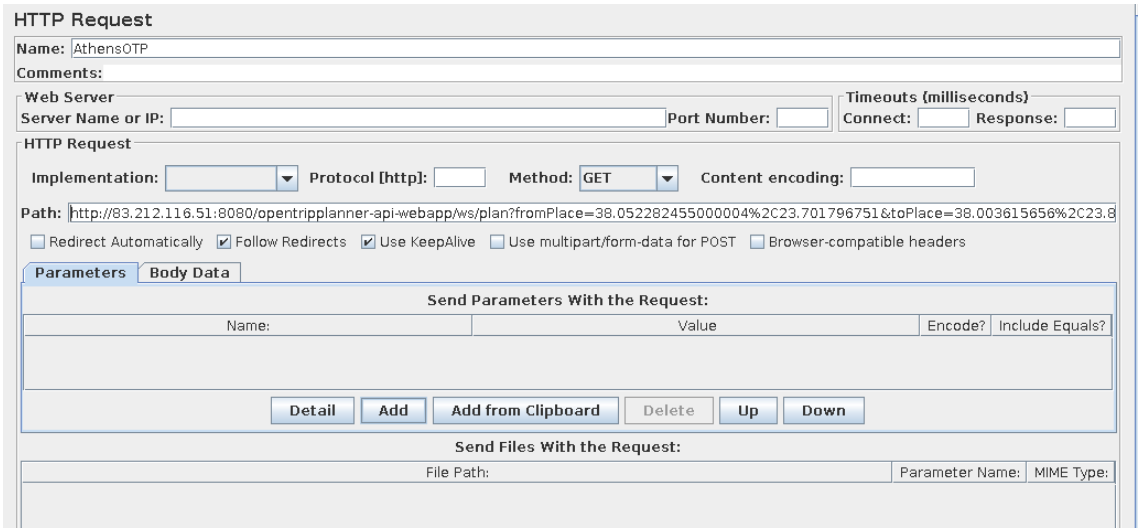
Το Thread Group, όπως αναφέρθηκε και πιο πάνω, προσομοιώνει τους χρήστες που εκτελούν αιτήματα στην εφαρμογή με κατάλληλα νήματα. Για τις ανάγκες των πειραμάτων εGπιλέχθηκαν οι δοκιμές με αριθμό χρηστών από 10 μέχρι 150 με βήμα 10. Επιπλέον, ο χρόνος εκκίνησης των νημάτων επιλέχθηκε να είναι περίπου 250 milliseconds για κάθε νήμα. Αυτό σημαίνει ότι για 50 νήματα-χρήστες ο χρόνος εκκίνησης θα είναι περίπου 12 δευτερόλεπτα. Στην εικόνα 5.9 φαίνεται ένα παράδειγμα Thread Group που δημιουργήθηκε για τις ανάγκες των πειραμάτων.



Σχήμα 5.9: Το συστατικό Thread Group των πειραμάτων του Apache JMeter

5.3.2 Τοποθέτηση Samplers

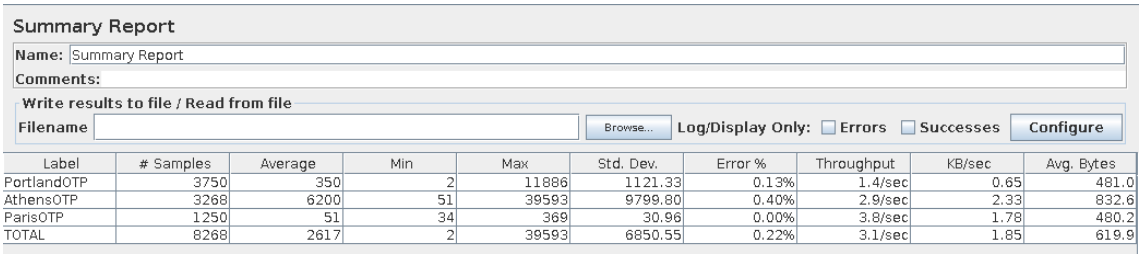
Οι Samplers που επιλέχθηκαν είναι οι HTTP Requests samplers καθώς η διεπαφή της εφαρμογής OpenTripPlanner απαντάει σε HTTP Requests. Η μέθοδος που επιλέχθηκε για την εκτέλεση του request είναι η GET καθώς θέλαμε να ελέγξουμε την απόκριση της εφαρμογής όταν ζητείται η βέλτιστη διαδρομή μεταξύ δύο σημείων. Καθώς χρειάστηκε ο έλεγχος της εφαρμογής και για τις τρεις πόλεις, τα διαφορετικά HTTP Requests ομαδοποιήθηκαν ανά πόλη, δίνοντάς τους διαφορετική ονομασία και τοποθετώντας τα κάτω από έναν If Controller ο οποίος έχει ως συνθήκη true αλλά μπορεί να αλλαχθεί σε περίπτωση που θελήσουμε να παραλείψουμε μια πόλη. Όλα συνολικά τα HTTP Requests τοποθετήθηκαν κάτω από ένα Loop Controller για τη δυνατότητα εκτέλεσής τους πάνω από μια φορά. Τέλος, σε κάθε Sampler τοποθετήθηκε ένας Timer με καθυστέρηση 10 δευτερολέπτων ανάμεσα σε κάθε ξεχωριστό request για την αποφυγή της συμφόρησης του εξυπηρετητή με πολλαπλά αιτήματα σε μικρό χρονικό διάστημα. Στην εικόνα 5.10 φαίνεται ένα παράδειγμα HTTP Request Sampler που χρησιμοποιήθηκε.



Σχήμα 5.10: HTTP Request Sampler που χρησιμοποιήθηκε στα πειράματα του Apache JMeter

5.3.3 Τοποθέτηση Listeners

Κάτω από το Thread Group τοποθετήθηκαν Listeners για την εμφάνιση των αποτελεσμάτων. Συγκεκριμένα, τοποθετήθηκε ο Response Time Graph, ο View Results Tree, ο View Results in Table και ο Summary Report. Οι τρεις πρώτοι χρησιμοποιήθηκαν για μια πρόχειρη εποπτεία των αποτελεσμάτων και αποσφαλμάτωση των πειραμάτων αλλά απενεργοποιήθηκαν κατά την τελική μέτρηση όλων των χρόνων καθώς όπως αναφέρεται και πιο πάνω η κατανάλωσή τους σε πόρους είναι απαγορευτική. Ο Summary Report Listener από την άλλη καταναλώνει λιγότερη μνήμη και δίνει μια καλή εικόνα των χρόνων απόκρισης μέσω της εξαγωγής στατιστικών. Στην εικόνα 5.11 φαίνεται ο Summary Report Listener που χρησιμοποιήθηκε.

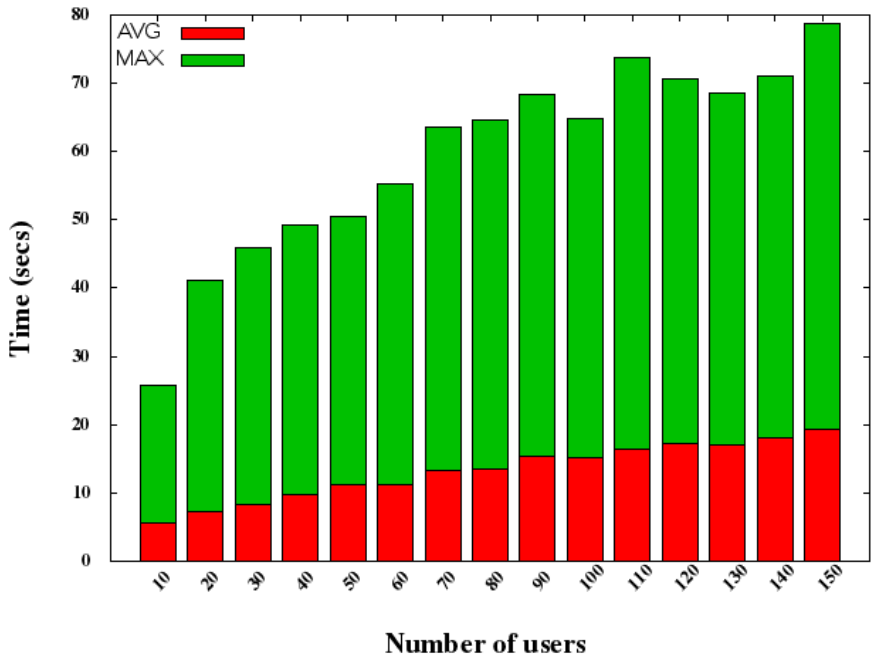


Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
PortlandOTP	3750	350	2	11886	1121.33	0.13%	1.4/sec	0.65	481.0
AthensOTP	3268	6200	51	39593	9799.80	0.40%	2.9/sec	2.33	832.6
ParisOTP	1250	51	34	369	30.96	0.00%	3.8/sec	1.78	480.2
TOTAL	8268	2617	2	39593	6850.55	0.22%	3.1/sec	1.85	619.9

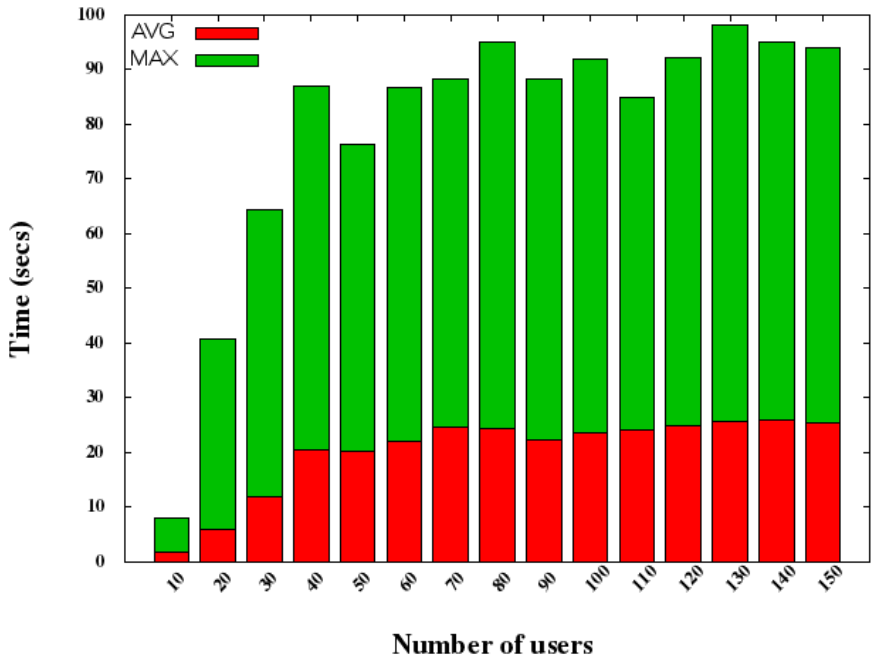
Σχήμα 5.11: Summary Report Listener που χρησιμοποιήθηκε στα πειράματα του Apache JMeter

5.4 Παρουσίαση αποτελεσμάτων και συμπεράσματα

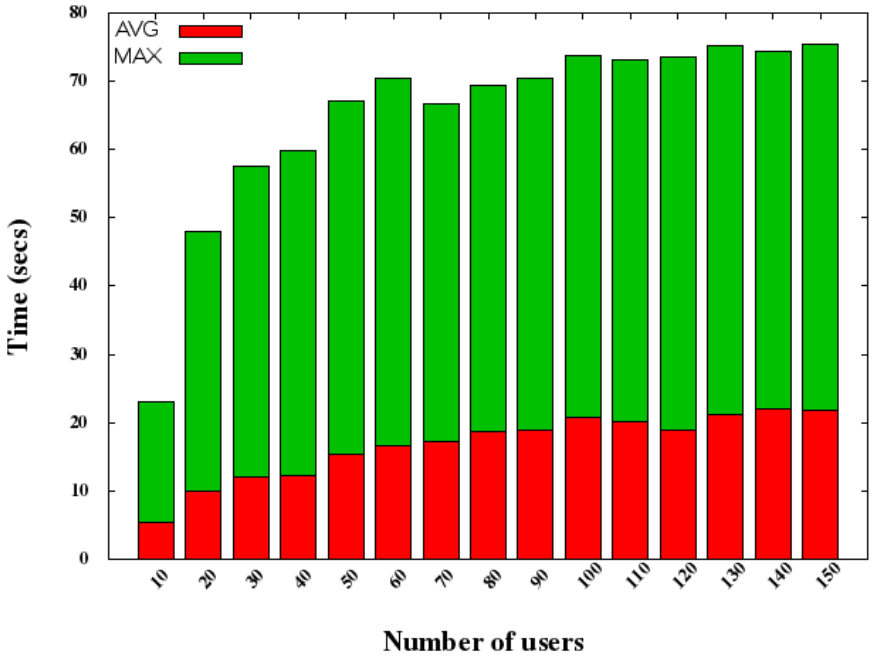
Τα πειράματα αρχικά εκτελέστηκαν για την εκτέλεση του OpenTripPlanner με αλγόριθμο δρομολόγησης τον MOA* από το ίδιο μηχάνημα που εκτελέστηκαν τα αντίστοιχα πειράματα με τον otp-client. Στα διαγράμματα 5.12, 5.13 και 5.14 παρουσιάζονται τα αποτελέσματα για τις τρεις πόλεις και την εκτέλεση του αλγορίθμου MOA*. Τα αποτελέσματα αφορούν τους μέσους και μέγιστους χρόνους απάντησης συναρτήσει του αριθμού των χρηστών.



Σχήμα 5.12: Αποτελέσματα της εκτέλεσης Apache JMeter για MOA* στην Αθήνα



Σχήμα 5.13: Αποτελέσματα της εκτέλεσης Apache JMeter για MOA* στο Παρίσι



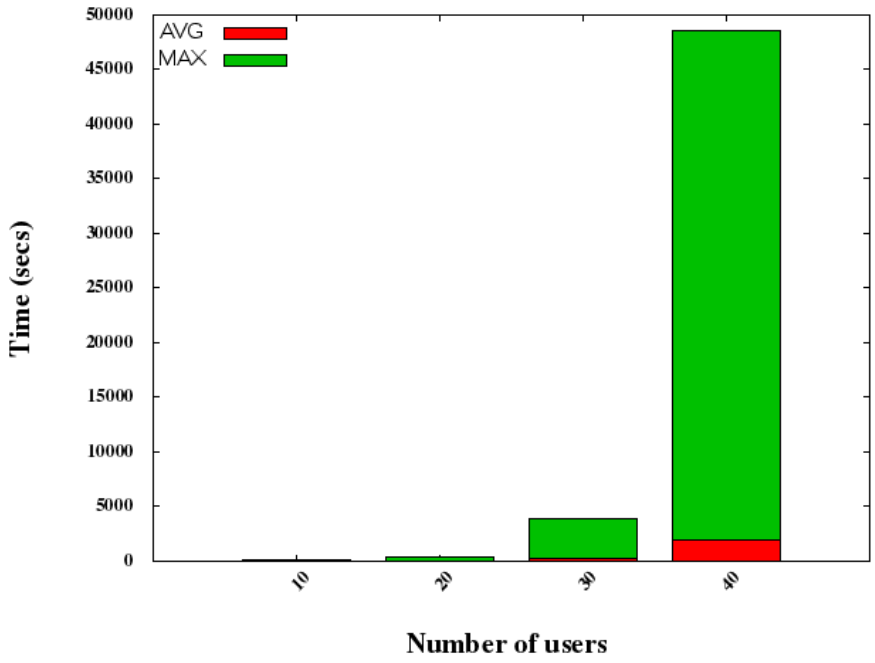
Σχήμα 5.14: Αποτελέσματα της εκτέλεσης Apache JMeter για MOA* στο Πόρτλαντ

Παρατηρούμε ότι για για της τρεις πόλεις τόσο ο μέσος όσο και ο μέγιστος χρόνος απάντησης αυξάνονται αναλογικά με τον αριθμό των χρηστών κάτι που ήταν αναμενόμενο.

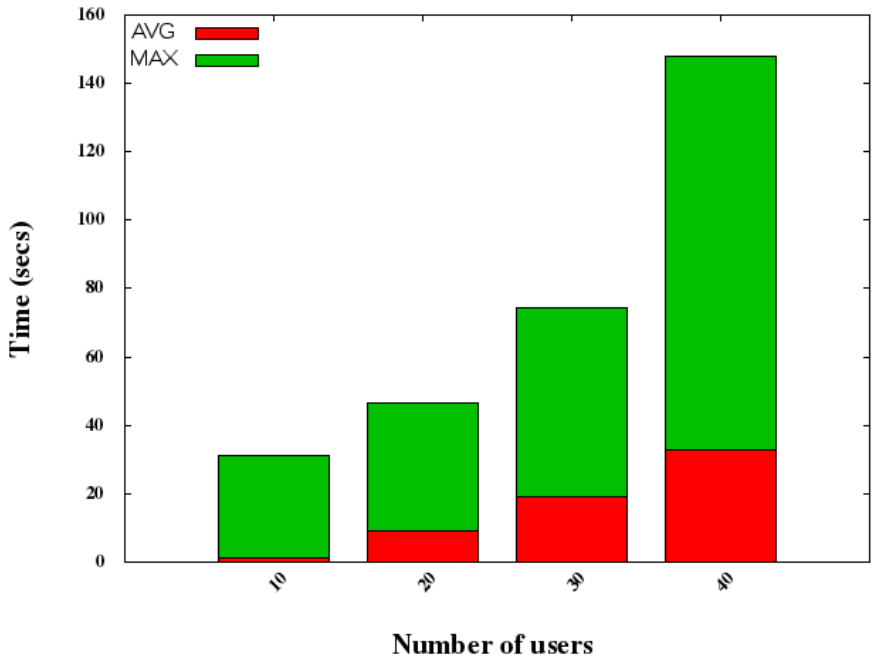
Συγκεκριμένα, για την Αθήνα φαίνεται πως ο μέσος χρόνος απάντησης κυμαίνεται από τα 5 δευτερόλεπτα (που αντιστοιχούν σε 10 χρήστες) στα 20 δευτερόλεπτα που αντιστοιχούν σε 150 χρήστες. Για το Παρίσι, φαίνεται πως ο μέσος χρόνος απάντησης κυμαίνεται από 2 δευτερόλεπτα μέχρι 25 ενώ για το Πόρτλαντ παρατηρείται πως ο μικρότερος χρόνος απάντησης είναι τα 5 δευτερόλεπτα και ο μεγαλύτερος τα 22.

Συγκρίνοντας τα διαγράμματα για τις τρεις πόλεις μεταξύ τους παρατηρούμε πως δεν υπάρχει κάποια μεγάλη διαφοροποίηση ανάμεσα στις πόλεις όσο αυξάνεται ο αριθμός των χρηστών. Ενώ δηλαδή, μέχρι τους 30 χρήστες οι χρόνοι έχουν την εικόνα που παρουσιάζεται και στο προηγούμενο κεφάλαιο, με το Παρίσι για έχει μειωμένους χρόνους, από σαράντα χρήστες και μετά, παρατηρείται μια ομοιομορφία στα αποτελέσματα και για τις τρεις πόλεις. Αυτό σημαίνει πως ο εξυπηρετητής σε μεγάλο φόρτο αιτημάτων παρουσιάζει μια σχετικά σταθερή καθυστέρηση καθώς ο χρόνος καθυστέρησης λόγω φορτίου είναι αρκετά μεγαλύτερος από το χρόνο υπολογισμού της βέλτιστης διαδρομής.

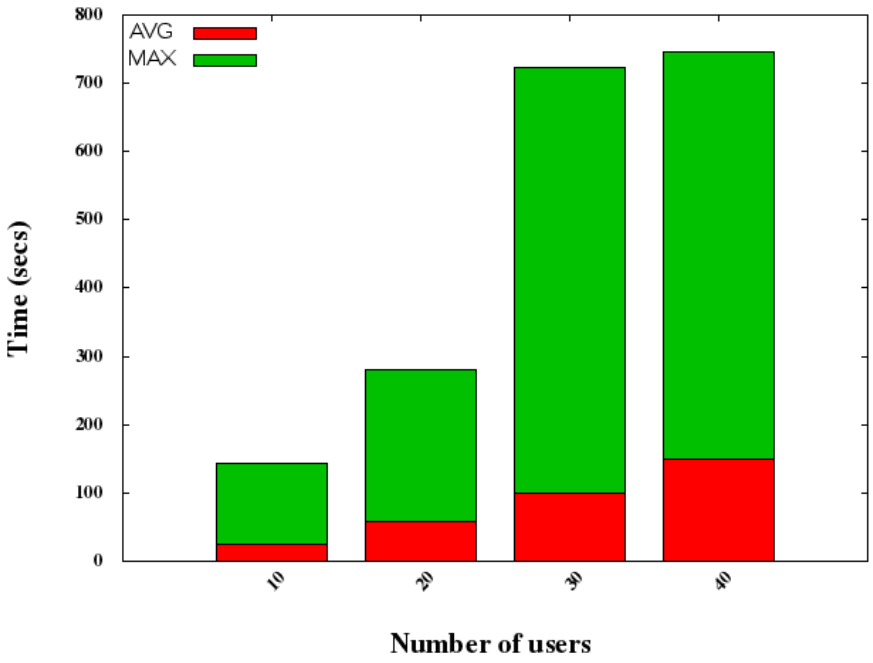
Τα πειράματα πραγματοποιήθηκαν και για την εκτέλεση της εφαρμογής με χρήση του αλγορίθμου Raptor. Τα αποτελέσματα παρουσιάζονται για διαγράμματα 5.15, 5.16 και 5.17.



Σχήμα 5.15: Αποτελέσματα της εκτέλεσης Apache JMeter για Raptor στην Αθήνα



Σχήμα 5.16: Αποτελέσματα της εκτέλεσης Apache JMeter για Raptor στο Παρίσι



Σχήμα 5.17: Αποτελέσματα της εκτέλεσης Apache JMeter για Raptor στο Πόρτλαντ

Τα πειράματα για τον αλγόριθμο Raptor δεν εκτελέστηκαν για μέχρι 150 χρήστες καθώς οι χρόνοι απόκρισης του εξυπηρετητή αυξάνονταν πάρα πολύ με τη αύξηση του αριθμού των χρηστών. Για το Παρίσι, ο μικρότερος χρόνος, που αντιστοιχούσε σε 10 χρήστες, ήταν 2 δευτερόλεπτα ενώ ο χρόνος για τους 40 χρήστες ήταν 35 δευτερόλεπτα. Ο αντίστοιχος χρόνος για τον MOA* ήταν 20 δευτερόλεπτα. Θα μπορούσαμε να πούμε πως στο δίκτυο του Παρισιού ο Raptor ήταν αισθητά πιο αργός αλλά όχι απαγορευτικός.

Αντίθετα, για τις δύο άλλες πόλεις μπορούμε να συμπεράνουμε πως ο αλγόριθμος Raptor καθιστά την εφαρμογή απαγορευτικά πιο αργή. Για το Πόρτλαντ ο μέσος χρόνος που αντιστοιχεί σε 40 χρήστες είναι 150 δευτερόλεπτα ενώ για την Αθήνα 2000 δευτερόλεπτα. Παρατηρήθηκε μάλιστα ότι ο εξυπηρετητής σε κάποιες περιπτώσεις επέστρεφε μήνυμα λάθους (τα δεδομένα του αιτήματος είχαν ελεγχθεί ως προς την ορθότητά τους). Το ποσοστό των μηνυμάτων λάθους ήταν της τάξεως του 3αλλά υποδεικνύει πως ο αλγόριθμος Raptor δεν είχε την αναμενόμενη απόδοση. Μια πιθανή εξήγηση για τη διαφοροποίηση του Παρισιού είναι η μορφολογία του δικτύου, καθώς δεν περιλαμβάνει πολλά διαφορετικά Μέσα σε αντίθεση με αυτό της Αθήνας.

Από τα παραπάνω, μπορούμε να συμπεράνουμε πως η εφαρμογή OpenTripPlanner που εκτελέσαμε είχε ικανοποιητική απόκριση για την εκτέλεση του αλγορίθμου MOA*, αλλά για τον αλγόριθμο Raptor, η χρήση της εφαρμογής OpenTripPlanner καθίσταται απαγορευτική κάτω από βαρύ φορτίο.

Συνοψίζοντας, στο κεφάλαιο αυτό αναλύθηκε η διαδικασία που ακολουθήθηκε για την εκτέλεση πειραμάτων προσομοίωσης μεγάλου ταυτόχρονου φορτίου στον εξυπηρετητή με τη βοήθεια του εργαλείου Apache JMeter. Τα αποτελέσματα εκτέλεσης των πειραμάτων αυτών αποτυπώθηκαν σε διαγράμματα και διαπιστώθηκε ότι για τις πόλεις με πιο πολύπλοκο δίκτυο Μέσων οι χρόνοι απόκρισης της εφαρμογής είναι αρκετά μεγαλύτεροι. Για την περαιτέρω διερεύνηση των δικτύων των Μέσων Μαζικής Μεταφοράς αναπτύχθηκε μια διαδικτυακή εφαρμογή που χρησιμοποιεί τις βάσεις με τα δεδομένα GTFS των τριών πόλεων και οπτικοποιεί

στατιστικές πληροφορίες για τα δίκτυα των Μέσων. Η εφαρμογή αυτή περιγράφεται στο επόμενο κεφάλαιο.

Κεφάλαιο 6

Σχεδιασμός διαδικτυακής υπηρεσίας για οπτικοποίηση GTFS δεδομένων

Στα πλαίσια της παρούσας διπλωματικής υλοποιήθηκε μια Web εφαρμογή για την εξαγωγή και οπτικοποίηση στατιστικών πληροφοριών που αφορούν τα δεδομένα GTFS των πόλεων. Η βασική της λειτουργία είναι η παρουσίαση μέσω διαγραμμάτων στατιστικών που αφορούν το δίκτυο Μέσων Μαζικής Μεταφοράς μιας ή περισσότερων πόλεων.

6.1 Αναγκαιότητα ανάπτυξης της υπηρεσίας

Η εφαρμογή αυτή συνεισφέρει στην εξαγωγή συμπερασμάτων για την πολυπλοκότητα των δρομολογίων των Μέσων Μαζικής Μεταφοράς βάση πολλαπλών παραμέτρων καθώς και στη σύγκριση της πολυπλοκότητας ανάμεσα στα δεδομένα διαφορετικών πόλεων. Τα GTFS αρχεία περιέχουν ένα μεγάλο όγκο πληροφοριών για το πρόγραμμα δρομολογίων και τις γεωγραφικές πληροφορίες για μια πόλη και με κατάλληλη επεξεργασία τους μπορεί να εξαχθεί πολύτιμη πληροφορία για το δίκτυο μιας πόλης. Οι πληροφορίες αυτές αφορούν τη συνδεσιμότητα του δικτύου των Μέσων Μαζικής Μεταφοράς, το εύρος του σε χιλιόμετρα, το μέσο χρόνο που δαπανάται στις μετακινήσεις και πολλούς άλλους παραμέτρους που είναι ενδεικτικοί του δικτύου των Μέσων μιας πόλης. Οι πληροφορίες αυτές συνοδευόμενες με έναν φιλικό προς το χρήστη τρόπο οπτικοποίησής τους θα μπορούσαν να συνεισφέρουν σημαντικά στην αξιολόγηση, τον εντοπισμό ελαττωμάτων και τη βελτίωση του δικτύου Μέσων Μαζικής Μεταφοράς μιας πόλης.

6.2 Βασικές αρχές σχεδίασης της υπηρεσίας

Η εφαρμογή επιλέχθηκε να είναι διαδικτυακή (web application) για την όσο το δυνατό πιο εύκολη και απλή χρήση της. Ο χρήστης που επιθυμεί να δει τα GTFS δεδομένα για μια πόλη αρκεί να χρησιμοποιήσει τον περιηγητή ιστού που έχει στον υπολογιστή χωρίς κανένα άλλο προαπαιτούμενο όπως η εγκατάσταση κάποιου προγράμματος στον τοπικό του υπολογιστή.

Επιπλέον, άλλο ένα γεγονός που χρειάστηκε να ληφθεί υπόψιν κατά την ανάπτυξη της εφαρμογής είναι η συνεχής ανανέωση των δρομολογίων όπως η προσθήκη στάσεων, η κατάργηση γραμμών ή η αλλαγή των ωραρίων που γίνεται από το υπεύθυνο πρακτορείο κάθε πόλης. Για τον λόγο αυτό τα δεδομένα που λαμβάνει η εφαρμογή για επεξεργασία και προβολή δεν αρκεί να είναι στατικά αλλά πρέπει να προσαρμόζονται στις αλλαγές και να ανανεώνονται κάθε φορά που χρειάζεται. Για το λόγο αυτό κρίθηκε αναγκαία η ύπαρξη μιας βάσης δεδομένων που τα περιέχει και τα προσφέρει δυναμικά στην εφαρμογή όταν ζητούνται.

Μια ακόμα συνιστώσα του σχεδιασμού της υπηρεσίας είναι ο τρόπος εμφάνισης των δεδομένων. Τα δεδομένα GTFS τόσο στα αρχεία όσο και στη βάση δεδομένων υπάρχουν σε αναλυτική μορφή και περιέχουν τις πληροφορίες για κάθε οντότητα ξεχωριστά (όπως στάση, δρομολόγιο και ταξίδι). Μια τέτοια αναπαράσταση δεν είναι χρήσιμη για το χρήστη καθώς δεν προσφέρει από μόνη της μια γενική και καθολική εικόνα για το δίκτυο Μέσων Μαζικής Μεταφοράς. Από την άλλη, η εύρεση και παρουσίαση των μέσων όρων των στατιστικών πληροφοριών ενός δικτύου (όπως μέσος χρόνος δρομολογίων) δεν είναι αρκετά περιγραφικός από μόνος του και μπορεί να αγνοεί αρκετή χρήσιμη πληροφορία. Για τους παραπάνω λόγους επιλέχθηκε η μέτρηση και παρουσίαση στατιστικών για τα δεδομένα GTFS σε μια μορφή ομαδοποίησης των μετρήσεων γύρω από ένα εύρος τιμών. Με την επιλογή αυτή επιτυγχάνουμε μια πιο πλούσια και εκφραστική αναπαράσταση της πληροφορίας αλλά όχι τόσο αναλυτική ώστε να μην μπορούν να εξαχθούν γενικά συμπεράσματα.

Για την οπτικοποίηση των αποτελεσμάτων επιλέχθηκε η χρήση διαγραμμάτων καθώς τα διαγράμματα μπορούν να αποτυπώσουν με καλαίσθητο και κατανοητό τρόπο στατιστικά αποτελέσματα σε σχέση με την απλή παρουσίαση αριθμών. Επιπλέον, η χρήση διαγραμμάτων διευκολύνει στην σύγκριση των αποτελεσμάτων τόσο σε επίπεδο ενός διαγράμματος όσο και σε επίπεδο ομοίων διαγραμμάτων που απεικονίζουν διαφορετική πληροφορία.

Τέλος, μια πολύ σημαντική απόφαση για το σχεδιασμό της υπηρεσίας είναι η επιλογή του τύπου των στατιστικών που θέλουμε να μετρώνται και να παρουσιάζονται. Οι οντότητες που υπάρχουν στη βάση δεδομένων είναι και αυτές που δίνονται στα αρχεία GTFS των πρακτορειών και αυτές για τις οποίες μπορούν να εξαχθούν στατιστικά που τις αφορούν είναι οι εξής:

- Στάσεις (Stops)
- Ταξίδια (Trips)
- Δρομολόγια (Route)

Πάνω στις παραπάνω οντότητες αποφασίστηκαν τα στατιστικά που θα μετρηθούν και τις αφορούν. Για τις στάσεις, μια μέτρηση που κρίθηκε ενδεικτική είναι το πόσα δρομολόγια τις έχουν σαν αφετηρία, προορισμό ή ενδιάμεσο σταθμό όπως και πόσα ταξίδια τις διασχίζουν σε εικοσιτετράωρη βάση. Για τα ταξίδια κρίθηκε σημαντικό να μετρηθούν ο αριθμός των στάσεων που περιλαμβάνουν η χρονική διάρκειά τους καθώς και η απόσταση που διανύουν. Όμοια στατιστικά μετρήθηκαν και για τα δρομολόγια με επιπλέον στατιστικό που αφορά τα ταξίδια που εκτελούν σε ένα εικοσιτετράωρο.

6.3 Περιγραφή λειτουργίας

Η εφαρμογή που υλοποιήθηκε είναι ένα Web application το οποίο εκτελείται σε οποιονδήποτε Web browser και παρουσιάζει στο χρήστη τις στατιστικές πληροφορίες για το δίκτυο

των Μέσων Μαζικής Μεταφοράς των πόλεων που έχει επιλέξει. Ο χρήστης έχει τη δυνατότητα να επιλέξει τις πόλεις για τις οποίες επιθυμεί να δει τις πληροφορίες ανάμεσα σε τρεις διαφορετικές πόλεις την Αθήνα, το Παρίσι και το Πόρτλαντ είτε για μία είτε για πολλαπλές πόλεις ταυτόχρονα ώστε να γίνεται σύγκριση. Επιπλέον, έχει την επιλογή εμφάνισης των πληροφοριών ανάμεσα με τρεις μορφές γραφικής παράστασης το γράφημα στηλών (Bars), το γράφημα πίτας (Pie) και το γράφημα δακτυλίου (Donut) για την καλύτερη αναπαράσταση κάθε τύπου πληροφορίας. Οι στατιστικές πληροφορίες από τις οποίες καλείται να επιλέξει για παρουσίαση από την εφαρμογή είναι οι εξής:

- Trips per Route 6.1
  Ο αριθμός των ταξιδιών(trips) που αντιστοιχούν σε ένα συγκεκριμένο δρομολόγιο(route).

- Stops per Route 6.2
  Ο αριθμός των στάσεων που αντιστοιχούν σε ένα δρομολόγιο(route) από την αφετηρία του ως το τέρμα.

- Duration per Route 6.3
  Ο μέσος χρόνος διάρκειας ενός δρομολογίου(route). Ο συγκεκριμένος χρόνος αντιστοιχεί στο μέσο όρο της διάρκειας όλων των ταξιδιών(trips) που αντιστοιχούν στο συγκεκριμένο δρομολόγιο.

- Absolute distance per Route 6.4
  Η ευκλείδεια απόσταση ανά δρομολόγιο από την αφετηρία ως το τέρμα. Η συγκεκριμένη απόσταση μετράται σε χιλιόμετρα και εκφράζει την απόλυτη απόσταση μεταξύ των δύο γεωγραφικών σημείων όπου βρίσκονται οι στάσεις της αφετηρίας και του τέρματος για ένα δρομολόγιο. Καθώς οι γεωγραφικές πληροφορίες εκφράζονται στα αρχεία GTFS σε γεωγραφικές μοίρες χρειάστηκε η μετατροπή τους από μοίρες σε χιλιόμετρα με χρήση της εξίσωσης haversine [11]

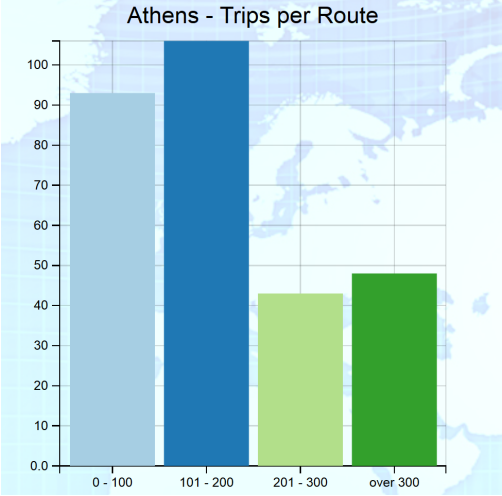
- Distance per Route(degrees) 6.5
  Η απόσταση ανά δρομολόγιο από την αφετηρία ως το τέρμα εκφρασμένο σε γεωγραφικές μοίρες. Η διαφορά της μέτρησης αυτής από την προηγούμενη έγκειται στο ότι στην συγκεκριμένη μέτρηση λαμβάνεται υπόψιν η απόσταση όλων των διαδοχικών στάσεων ενός δρομολόγιο. Η μέτρηση αυτή κρίθηκε αναγκαία καθώς η απόλυτη ευκλείδεια απόσταση από την αφετηρία ως το τέρμα πολλές φορές δεν είναι από μόνη της αντιπροσωπευτική της απόστασης ενός δρομολογίου. Συγκεκριμένα, βρέθηκαν αρκετά παραδείγματα όπου ένα μέσο μπορεί να κάνει κυκλική διαδρομή και καθώς η αφετηρία και το τέρμα αποτελούν την ίδια στάση η ευκλείδεια απόσταση του δρομολογίου να βρεθεί ίση με το μηδέν κάτι που δεν προσφέρει χρήσιμη πληροφορία για το δρομολόγιο.

- Distance per Route(km) 6.6
  Η απόσταση ανά δρομολόγιο από την αφετηρία ως το τέρμα μετρημένο σε χιλιόμετρα υπολογίζοντας την απόσταση όλων των διαδοχικών στάσεων μεταξύ τους. Η διαφορά με την προηγούμενη μέτρηση έγκειται μόνο στις μονάδες μέτρησης.

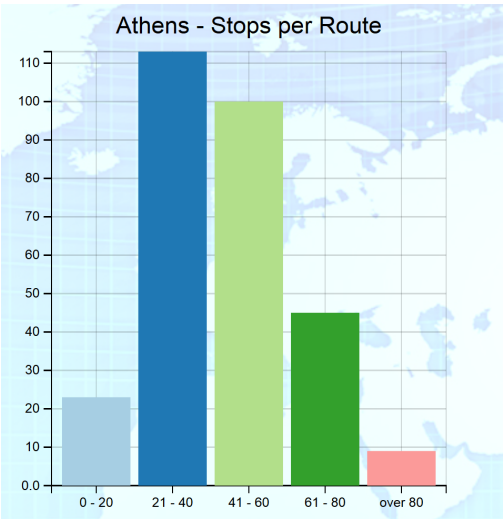
- Trips per Stop 6.7
  Ο αριθμός των ταξιδιών(trips) που αντιστοιχούν σε κάθε στάση(stop). Πρακτικά, αυτή η μέτρηση αντιπροσωπεύει το πόσες φορές θα σταματήσει ένα μέσο σε μία στάση σε ένα εικοσιτετράωρο.

- Routes per Stop 6.8
  Ο αριθμός των δρομολογίων(routes) που αντιστοιχούν σε μία στάση. Πρακτικά, αυτή

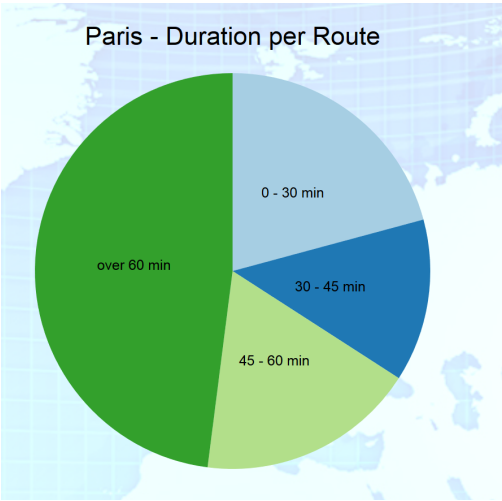
η μέτρηση αναπαριστά τον αριθμό των διαφορετικών γραμμών που περνάνε από μία στάση.



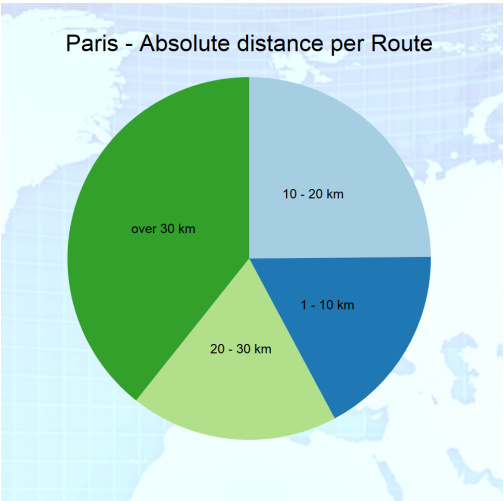
Σχήμα 6.1: Στατιστική μέτρηση Trips per Route



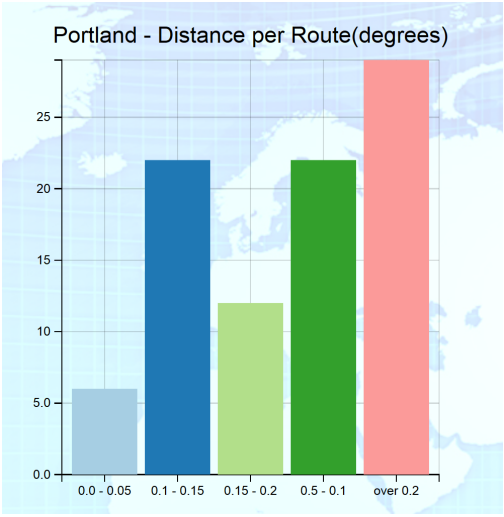
Σχήμα 6.2: Στατιστική μέτρηση Stops per Route



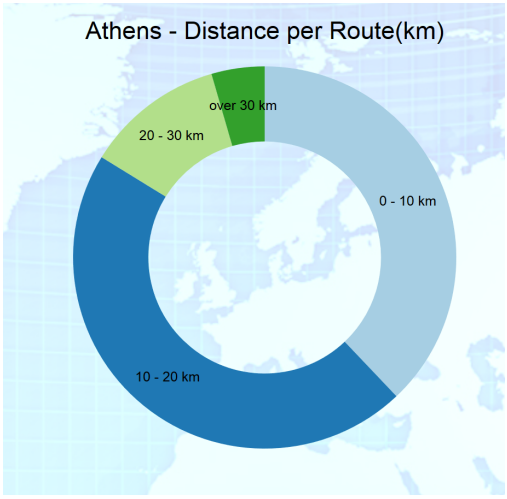
Σχήμα 6.3: Στατιστική μέτρηση Duration per Route



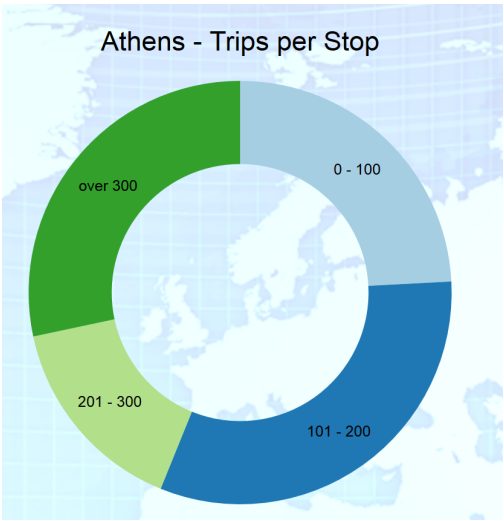
Σχήμα 6.4: Στατιστική μέτρηση Duration per Route (Absolute)



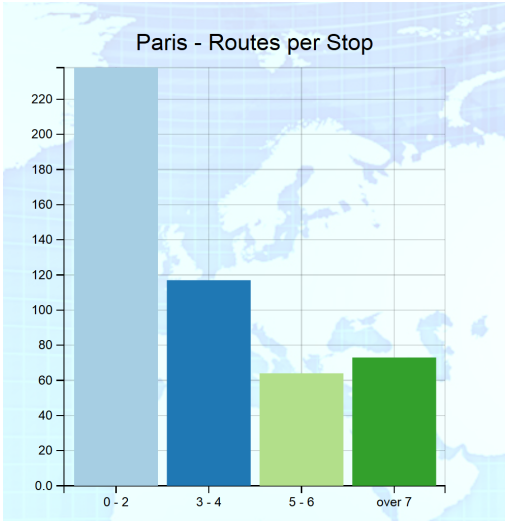
Σχήμα 6.5: Στατιστική μέτρηση Duration per Route (in geographical degrees)



Σχήμα 6.6: Στατιστική μέτρηση Duration per Route (in km)



Σχήμα 6.7: Στατιστική μέτρηση Trips per Stop



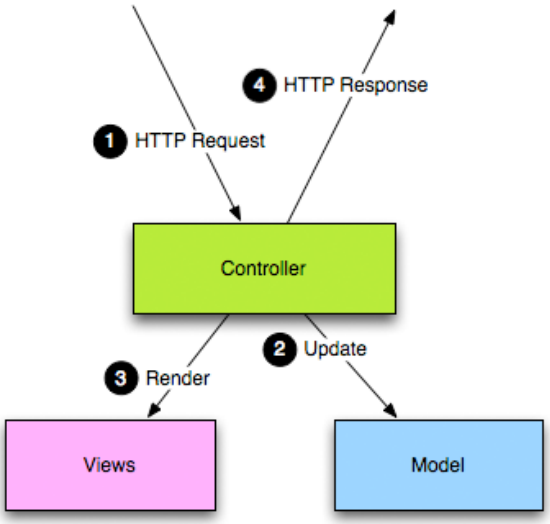
Σχήμα 6.8: Στατιστική μέτρηση Routes per Stop

6.4 Εργαλεία που χρησιμοποιήθηκαν

Στην παρούσα ενότητα αναφέρονται τα βασικά εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Εκτός από τα εργαλεία που αναφέρονται και παρακάτω χρησιμοποιήθηκε και η βάση δεδομένων postgres αν και παραλείπεται η περιγραφή της στην παρούσα ενότητα καθώς έχει αναφερθεί σε προηγούμενη.

6.4.1 Play Framework

Μία εφαρμογή Play ακολουθεί την αρχιτεκτονικό μοντέλο Model-View-Controller εφαρμοσμένο στην αρχιτεκτονική Web όπως παρουσιάζεται στο σχήμα 6.9. Η εφαρμογή χωρίζεται στο επίπεδο Μοντέλου (Model layer) και το επίπεδο Παρουσίασης (Presentation layer) το οποίο με τη σειρά του περιλαμβάνει το επίπεδο Όψεως(View layer) και το επίπεδο Ελέγχου (Controller layer).



Σχήμα 6.9: Η βασική αρχιτεκτονική μιας Web εφαρμογής που έχει αναπτυχθεί με το Play Framework

Το επίπεδο Ελέγχου αποκρίνεται σε γεγονότα που προκαλεί ο χρήστης και αποστέλλει εντολές στο επίπεδο Μοντέλου ώστε να αλλάξει την κατάστασή του, όπως για παράδειγμα η ανανέωση της βάσης δεδομένων ή κάποιου αρχείου. Επιπλέον, ενημερώνει το επίπεδο Όψεων ώστε να ανανεωθεί η αναπαράσταση των ζητούμενων πληροφοριών. Τα γεγονότα στα οποία αποκρίνεται σε μια Web εφαρμογή είναι συνήθως HTTP requests από τα οποία το επίπεδο Ελέγχου αντλεί τη σχετική πληροφορία από τα HTTP headers ή HTTP parameters και εφαρμόζει τις ανάλογες αλλαγές στα άλλα δυο επίπεδα.

Το επίπεδο Μοντέλου διαχειρίζεται την αναπαράσταση των δεδομένων που χρησιμοποιούνται στην εφαρμογή όπως για παράδειγμα τη βάση δεδομένων ή κάποιο αρχείο και ενημερώνει τις συσχετιζόμενα επίπεδα για την κατάστασή του. Όταν αλλάξει η κατάσταση του το επίπεδο Ελέγχου θα πρέπει να παράγει μία ανανεωμένη έξοδο και το επίπεδο Ελέγχου να προσαρμόσει τις σχετικές διαθέσιμες εντολές που περιέχει.

Το επίπεδο όψεων λαμβάνει τις πληροφορίες για το επίπεδο Μοντέλου και τις χρησιμοποιεί για να παράγει την έξοδο προς παρουσίαση για τον χρήστη. Μπορεί να υπάρχουν πολλαπλές Όψεις για ένα μοντέλο ανάλογα με την περίπτωση. Σε μία Web εφαρμογή το επίπεδο Όψεων

περιλαμβάνει αρχεία σε μορφή HTML, XML ή JSON.

Ο κύκλος ζωής μιας εφαρμογής που αναπτύχθηκε με το Play Framework που λαμβάνει HTTP Requests είναι ο εξής:

- Λήψη ενός HTTP Request από την εφαρμογή
- Το μέρος της εφαρμογής που είναι υπεύθυνο για τη μετάφραση των εισερχόμενων HTTP Requests (Router) επιλέγει την ενέργεια που θα εκτελεστεί από το επίπεδο ελέγχου (Controller).
- Εκτελείται ο κατάλληλος κώδικας της εφαρμογής
- Δημιουργείται μια ανάλογη Όψη της εφαρμογής
- Το αποτέλεσμα αποστέλλεται σε μορφή HTTP Response.

Στο διάγραμμα 6.10 φαίνεται η διαδρομή του HTTP Request.

6.4.2 JavaScript

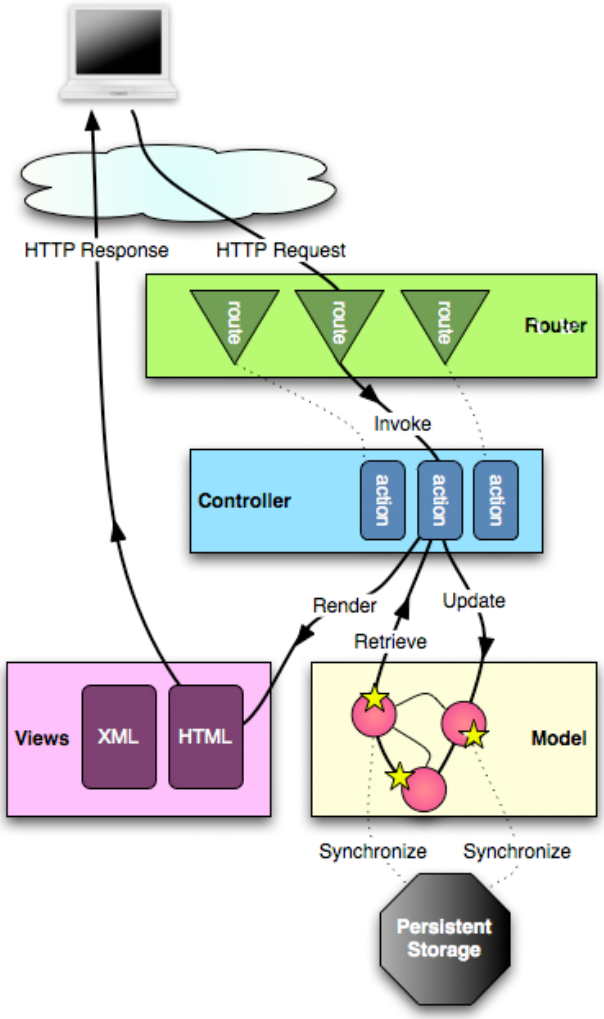
Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές με τη βασική της λειτουργικότητα να αφορά κατασκευή ιστοσελίδων. Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό στυλ προγραμματισμού.

6.4.3 HTML

Η HTML (ακρωνύμιο του αγγλικού HyperText Markup Language είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων.

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.



Σχήμα 6.10: Ο κύκλος ζωής ενός HTTP Request σε μία Web εφαρμογή του Play Framework

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

6.4.4 WIGeoChart

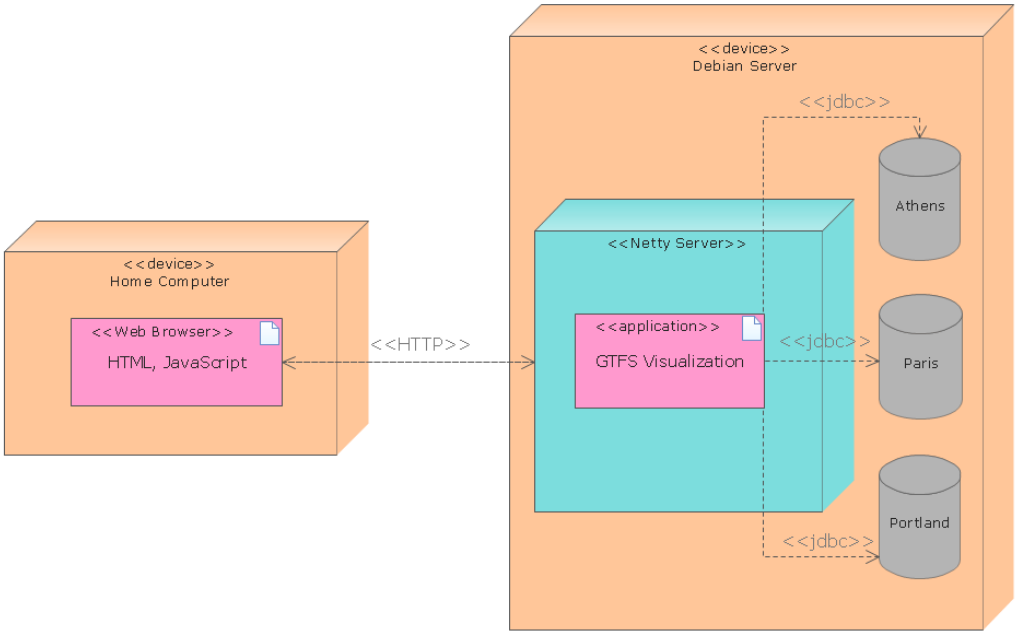
Το WIGeoChart αποτελεί μια βιβλιοθήκη υλοποιημένη σε γλώσσα JavaScript για την δημιουργία διαγραμμάτων από δεδομένα που δίνονται σε μορφή JSON και επιτρέπει την εύκολη ενσωμάτωσή του σε διαδικτυακές εφαρμογές που χρησιμοποιούν HTML. Δίνει τη δυνατότητα αναπαράστασης πολλών ειδών διαγραμμάτων τόσο απλών και ευρέως γνωστών όπως οι μπάρες και η πίτα όσο και πιο σύνθετων και διαδραστικών.

6.4.5 Flat-UI

Το Flat-UI είναι ένα στυλ μορφοποίησης CSS που ορίζει αρκετές κλάσεις στοιχείων της HTML προσφέροντας μια ενιαία μορφοποίηση και εμφάνιση της σελίδας HTML που χρησιμοποιείται. Ακολουθεί το σχεδιαστικό πρότυπο Flat design το οποίο αποτελεί μια μινιμαλιστική σχεδιαστική προσέγγιση που δίνει έμφαση στην χρηστικότητα της γραφικής διεπαφής.

6.5 Αρχιτεκτονική εφαρμογής

Για την ανάπτυξη και εκτέλεση της εφαρμογής χρησιμοποιήθηκε η πλατφόρμα Play Framework με χρήση της γλώσσας προγραμματισμού Java. Η πλατφόρμα Play Framework επιλέχθηκε έναντι άλλων τεχνολογιών ανάπτυξης web εφαρμογών (όπως το Servlet) για την φιλική προς τον χρήστη αρχιτεκτονική της και τη δυνατότητα που προσφέρει για ανάπτυξη επεκτάσιμων (scalable) web εφαρμογών με μικρή κατανάλωση υπολογιστικών πόρων (επεξεργαστική ισχύ, μνήμη). Στην εικόνα 6.11 φαίνεται η αρχιτεκτονική της εφαρμογής σε μορφή διαγράμματος UML.



Σχήμα 6.11: UML διάγραμμα της αρχιτεκτονικής για την Web εφαρμογή που αναπτύχθηκε

6.5.1 Web εφαρμογή

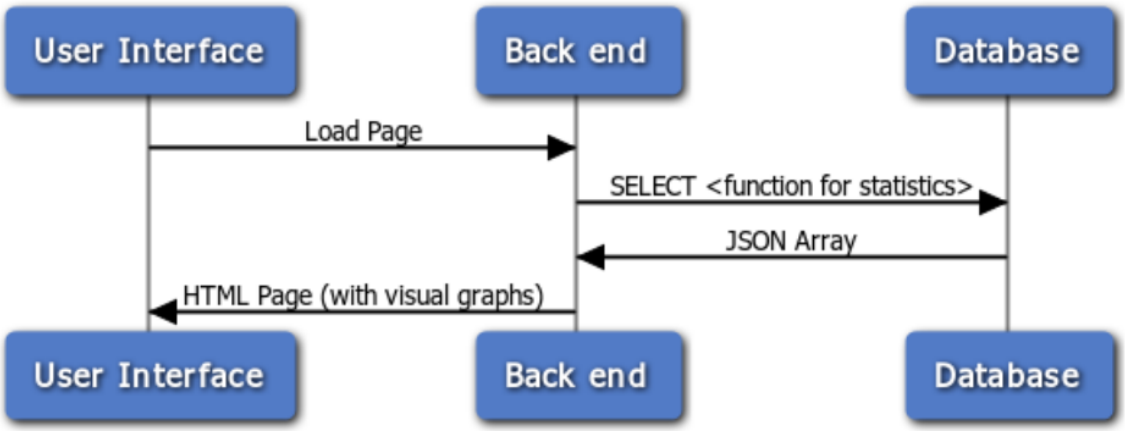
Η web εφαρμογή επικοινωνεί με τη βάση και παρουσιάζει στον χρήστη τα αποτελέσματα σε μορφή γραφικών παραστάσεων. Για τη δημιουργία του διεπαφής χρήστη χρησιμοποιήθηκε η μεταγλώσσα HTML και JavaScript ενώ για την επικοινωνία με τη βάση δεδομένων και διαχείριση των πληροφοριών που λαμβάνονται χρησιμοποιήθηκε η γλώσσα Java.

Η γραφική διεπαφή χρήστη είναι υλοποιημένη στη μεταγλώσσα HTML. Για την ομοιομορφία των στοιχείων(elements) της εφαρμογής επιλέχθηκε η χρήση του CSS Flat UI [7] το οποίο περιέχει στιλιστικές κλάσεις για διάφορα στοιχεία της HTML που χρησιμοποιήθηκαν. Για την επιλογή των στατιστικών πληροφοριών προς εμφάνιση, την επιλογή των πόλεων και των γραφημάτων αναπαράστασης χρησιμοποιήθηκε JavaScript. Τέλος, για την ίδια την αναπαράσταση της πληροφορίας μέσω διαγραμμάτων χρησιμοποιήθηκε η βιβλιοθήκη WIGeoChart [26] η οποία δέχεται δεδομένα σε μορφή JSON Array και δημιουργεί ένα γράφημα αναπαράστασης των δεδομένων αυτών.

Στο διάγραμμα 6.12 φαίνεται ένα απλοποιημένο διάγραμμα ακολουθίας μηνυμάτων UML για την εφαρμογή. Αξίζει να σημειωθεί πως η εφαρμογή επικοινωνεί και αποστέλλει queries σε τρεις διαφορετικές βάσεις δεδομένων(μια για κάθε πόλη).

6.5.2 Βάση δεδομένων

Για να εξαχθούν οι στατιστικές πληροφορίες χρησιμοποιήθηκαν οι βάσεις δεδομένων με το σχήμα όπως περιγράφηκε στο κεφάλαιο 3 και με το σχεσιακό διάγραμμα που φαίνεται στην εικόνα 3.4. Για τον υπολογισμό όμως των απαιτούμενων πληροφοριών χρειάστηκε να δημιουργηθούν επιπλέον όψεις(views) που υπολογίζουν τις αντίστοιχες στατιστικές μετρήσεις.



Σχήμα 6.12: UML διάγραμμα ακολουθίας μηνυμάτων για την Web εφαρμογή που αναπτύχθηκε

```
--------Stops per Route----------

CREATE OR REPLACE VIEW stops_per_route (route_id,stop_count) AS
    SELECT route_id, COUNT(stop_id)
    FROM
        (SELECT DISTINCT ON (routes.route_id) routes.route_id, trip_id
         FROM routes INNER JOIN trips ON trips.route_id=routes.route_id) AS t
         INNER JOIN stop_times AS st ON t.trip_id = st.trip_id
    GROUP BY route_id;
```

Παράθεση 6.1: Δημιουργία όψεως Stops per Route

Database Views

Για τον υπολογισμό των στατιστικών πληροφοριών χρειάστηκε να δημιουργηθούν κατάλληλες όψεις(views) στις βάσεις δεδομένων οι οποίες επιστρέφουν τα αποτελέσματα των αντίστοιχων queries. Αρχικά, υπολογίζεται για κάθε ξεχωριστή οντότητα η αντίστοιχη τιμή της μέτρησης του στατιστικού που επιθυμούμε. Για παράδειγμα, για τον υπολογισμό των ταξιδιών(trips) ανά δρομολόγιο(route) υπολογίζεται ο αριθμός των ταξιδιών για κάθε δρομολόγιο και δημιουργείται μία όψη η οποία περιέχει εγγραφές με όλα τα δρομολόγια και τον αριθμό των ταξιδιών που αντιστοιχεί στο καθένα. Στα σχήματα 6.2 και 6.1 φαίνονται μερικά παραδείγματα από τη δημιουργία των όψεων για κάποιες στατιστικές μετρήσεις.

Για την οπτικοποίηση των παραπάνω όψεων από κάποια γραφική παράσταση χρειάστηκε συσσωμάτωση των αποτελεσμάτων με κάποιον τρόπο και εύρεση του συνόλου των οντολογιών που αντιστοιχούν σε ένα εύρος τιμών. Έτσι, κάθε γραφική παράσταση που θα δημιουργηθεί αργότερα από των εφαρμογή θα μπορεί να δίνει πληροφορία για την κατανομή των οντολογιών στις τιμές που μετρήθηκαν όπως για παράδειγμα το πόσα δρομολόγια από το σύνολο των δρομολογίων περιέχουν αριθμό στάσεων μέσα σε ένα εύρος. Τα εύρη τιμών για κάθε περίπτωση επιλέχθηκαν βάση του μεγίστου και του ελαχίστου των τιμών σε κάθε όψη και είναι ίδια σε κάθε βάση δεδομένων (για κάθε πόλη) ώστε να επιτυγχάνεται άμεση σύγκριση των ίδιων μετρήσεων ανάμεσα στις πόλεις. Για την ευκολία πρόσβασης στα συσσωρευτικά αποτελέσματα δημιουργήθηκαν και πάλι όψεις που ερωτούν τις προηγούμενες, με εγγραφές

```
---Total distance per route counting every stop(in kilometers)---

CREATE OR REPLACE FUNCTION haversin(double precision) RETURNS double precision
    AS 'select power(sin($1/2),2)'
    LANGUAGE SQL
    RETURNS NULL ON NULL INPUT;


CREATE OR REPLACE FUNCTION distance(double precision,double precision,
    double precision,double precision) RETURNS double precision
    AS 'select 2*6371*asin(sqrt(haversin(radians($3-$1))+
    cos(radians($1))*cos(radians($1))*haversin(radians($4-$2))))'
    LANGUAGE SQL
    RETURNS NULL ON NULL INPUT;

CREATE OR REPLACE VIEW distance_per_route (route_id,distance) AS
SELECT stops1.route_id,
SUM(distance(stops1.stop_lat,stops1.stop_lon,stops2.stop_lat,stops2.stop_lon))
FROM (
SELECT route_id, stop_sequence, stop_lat, stop_lon
FROM stops, stop_times,
    (SELECT DISTINCT ON (route_id) * FROM trips) tr
    WHERE tr.trip_id = stop_times.trip_id
    AND stop_times.stop_id = stops.stop_id ) AS stops1,
(SELECT route_id, stop_sequence, stop_lat, stop_lon
    FROM stops, stop_times,
    (SELECT DISTINCT ON (route_id) * FROM trips) tr
    WHERE tr.trip_id = stop_times.trip_id
    AND stop_times.stop_id = stops.stop_id ) AS stops2
WHERE stops1.stop_sequence = stops2.stop_sequence+1
AND stops1.route_id=stops2.route_id
GROUP BY stops1.route_id;
```

Παράθεση 6.2: Δημιουργία όψεως Duration per Route με τη βοήθεια της εξίσωσης haversine για τη μετατροπή των γεωγραφικών μοιρών σε χιλιόμετρα

το εύρος των τιμών της μέτρησης και τον αριθμό των οντολογιών που τους αντιστοιχεί. Καθώς υπάρχει η ανάγκη άμεσης απόκρισης της web εφαρμογής όταν ζητείται από το χρήστη η οπτικοποίηση ενός στατιστικού, οι όψεις αυτές επιλέχθηκε να είναι materialized. Με αυτόν τον τρόπο αποφεύγεται η αναμονή κάποιον δευτερολέπτων πριν την απεικόνιση των στατιστικών πληροφοριών κάτι που θα καθιστούσε την εφαρμογή αργή στην απόκριση. Καθώς βέβαια θέλουμε οι όψεις να ανταποκρίνονται πάντα στα τρέχοντα δεδομένα της βάσης δεδομένων θα πρέπει να φροντίζουμε να ανανεώνονται και αυτές όταν γίνει update σε κάποιον πίνακα πάνω στον οποίο εκτελούν queries. Στο σχήμα 6.3 φαίνεται ένα παράδειγμα από τη δημιουργία των όψεων αυτών.

JSON αποτέλεσμα

Καθώς η εφαρμογή απαιτεί τα αποτελέσματα σε μορφή JSON για την δημιουργία της γραφικής παράστασης, προτιμήθηκε να επιστρέφεται από τη βάση δεδομένων ένα JSON Array ώστε να μην υπάρχει η ανάγκη δημιουργίας του προγραμματιστικά από την εφαρμογή. Συγκεκριμένα, η βάση δεδομένων δημιουργεί ένα JSON αντικείμενο για κάθε γραμμή της συσσωρευτικής

```
--------Stops per Route Aggregation--------

CREATE MATERIALIZED VIEW stops_per_route_agg (label,value) AS
    SELECT
    CASE WHEN stop_count BETWEEN 0 AND 20 THEN '0 - 20'
         WHEN stop_count BETWEEN 21 AND 40 THEN '21 - 40'
         WHEN stop_count BETWEEN 41 AND 60 THEN '41 - 60'
         WHEN stop_count BETWEEN 61 AND 80 THEN '61 - 80'
         ELSE 'over 80'
    END AS RouteRange,COUNT(route_id) AS RoutesTotal
    FROM stops_per_route
    GROUP BY
    CASE WHEN stop_count BETWEEN 0 AND 20 THEN '0 - 20'
         WHEN stop_count BETWEEN 21 AND 40 THEN '21 - 40'
         WHEN stop_count BETWEEN 41 AND 60 THEN '41 - 60'
         WHEN stop_count BETWEEN 61 AND 80 THEN '61 - 80'
         ELSE 'over 80'
     END
     ORDER BY
     RouteRange;
```

Παράθεση 6.3: Δημιουργία όψεως που συσσωρεύει τα αποτελέσματα για τη μέτρηση Stops per Route και τα ομαδοποιεί βάσει κάποιου εύρους τιμών

```
--SQL function
CREATE OR REPLACE FUNCTION stops_per_route() RETURNS json AS
'SELECT array_to_json(array_agg(row_to_json(t)))
    FROM (
    SELECT label,value
    FROM stops_per_route_agg
) t;'
LANGUAGE SQL;
```

Παράθεση 6.4: Δημιουργία συνάρτησης που επιστρέφει ένα JSON Array από τα δεδομένα μιας όψεως

όψης και τοποθετεί το κάθε αντικείμενο σε ένα Array δημιουργώντας έτσι ένα JSON Array στη μορφή που ζητά η εφαρμογή. Το JSON Array επιστρέφεται από τη βάση δεδομένων με μία συνάρτηση που δημιουργήθηκε για κάθε στατιστική μέτρηση για μεγαλύτερη ευκολία απόκτησης των στατιστικών από τη βάση. Στα σχήματα 6.4 φαίνεται ένα παράδειγμα δημιουργίας μιας τέτοιας συνάρτησης στη βάση.

Συνοψίζοντας, σε αυτό το κεφάλαιο παρουσιάστηκε η διαδικτυακή υπηρεσία που αναπτύχθηκε για την εξαγωγή και οπτικοποίηση στατιστικών πληροφοριών που αφορούν τα δεδομένα GTFS των τριών πόλεων. Συγκεκριμένα, αναλύθηκε η δομή και η λειτουργία της. Στο επόμενο κεφάλαιο δίνεται ένας σύντομος οδηγός για τη χρήση της υπηρεσίας και παρουσιάζονται τα αποτελέσματα που προκύπτουν από τη χρήση της και συμπεράσματα σε αυτά.

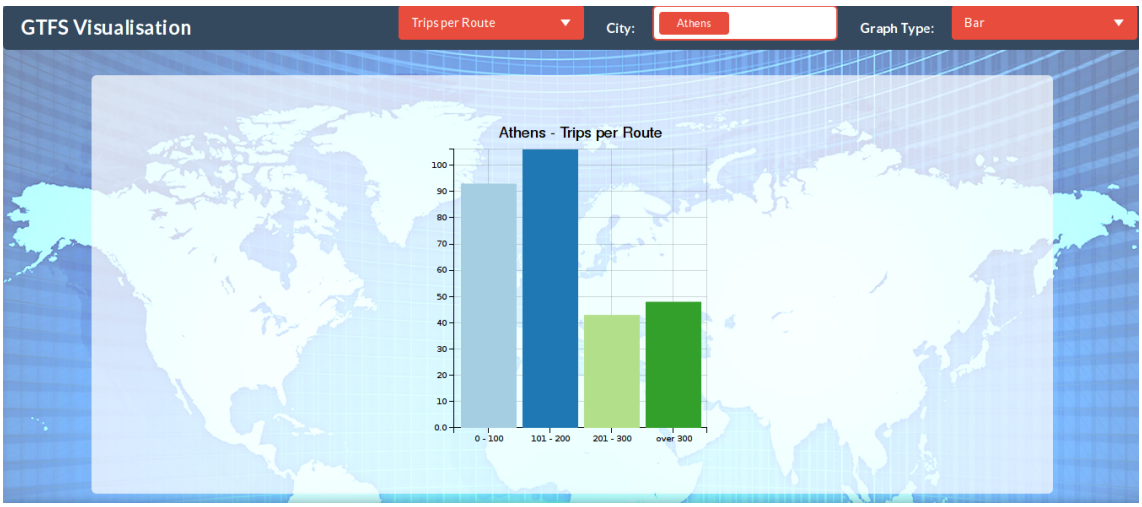
Κεφάλαιο 7

Παραδείγματα χρήσης της εφαρμογής και συμπεράσματα

Στο παρόν κεφάλαιο παρουσιάζεται ένας σύντομος οδηγός χρήσης της εφαρμογής για τους χρήστες της συνοδευμένος από παραδείγματα. Παρουσιάζονται επίσης τα συμπεράσματα από την εκτέλεση της εφαρμογής για τις επιλεγμένες πόλεις.

7.1 Παράδειγμα χρήσης της εφαρμογής

Όταν ο χρήστης επισκεφθεί την αρχική σελίδα της εφαρμογής βλέπει ένα διάγραμμα για το στατιστικό Trips per Route που αφορά την Αθήνα και είναι προεπιλεγμένο. 7.1

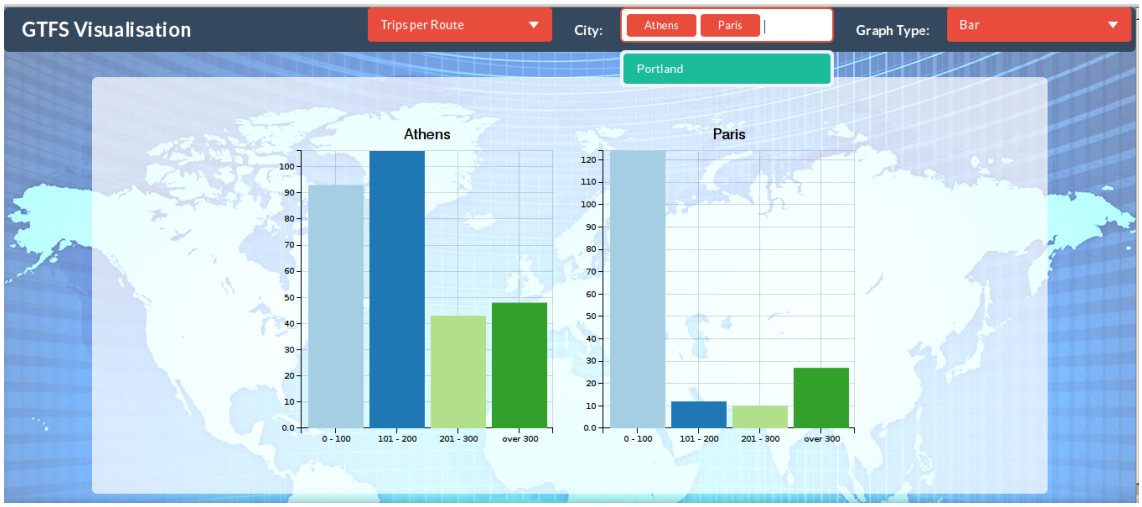


Σχήμα 7.1: Αρχική σελίδα της εφαρμογής

Μετά την εμφάνιση της αρχικής σελίδας ο χρήστης έχει τις εξής επιλογές:

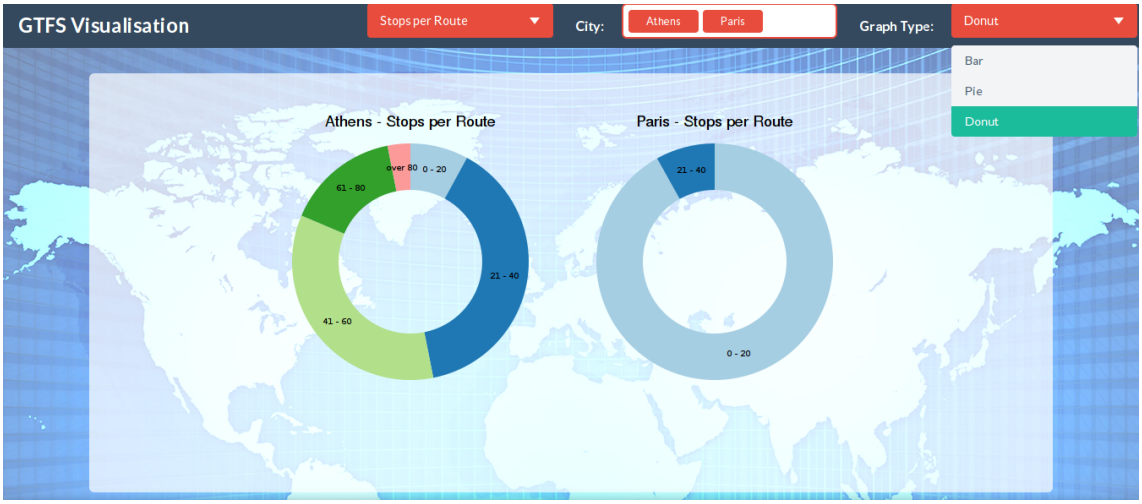
- Προσθήκη ή αφαίρεση πόλεων. 7.2
- Αλλαγή του στατιστικού που παρουσιάζεται. 7.3

- Αλλαγή του τύπου του γραφίματος απεικόνισης των αποτελεσμάτων. 7.4



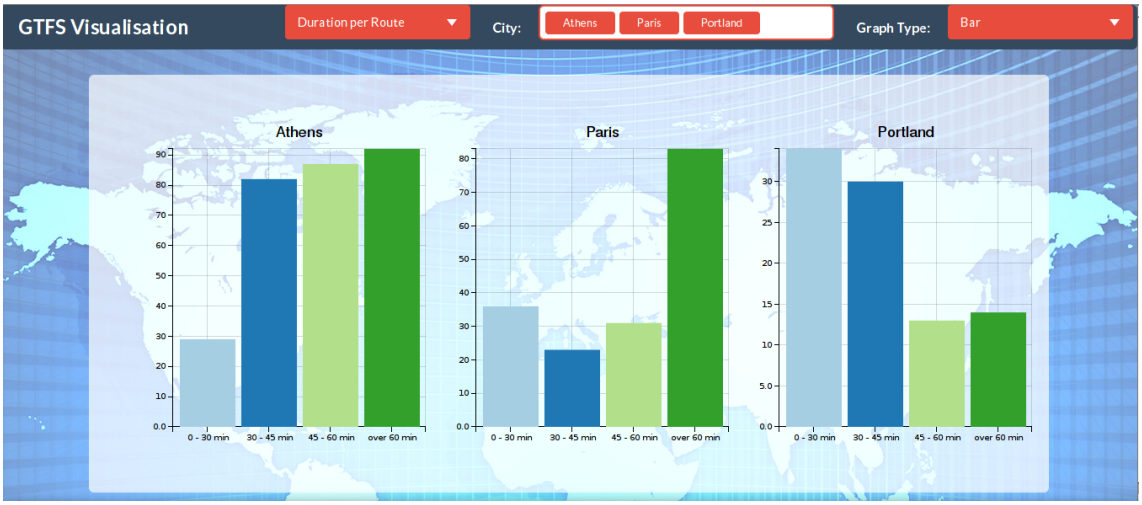
Σχήμα 7.2: Επιλογή πόλεων για την παρουσίαση των στατιστικών μετρήσεων

Σχήμα 7.3: Επιλογή στατιστικής μέτρησης

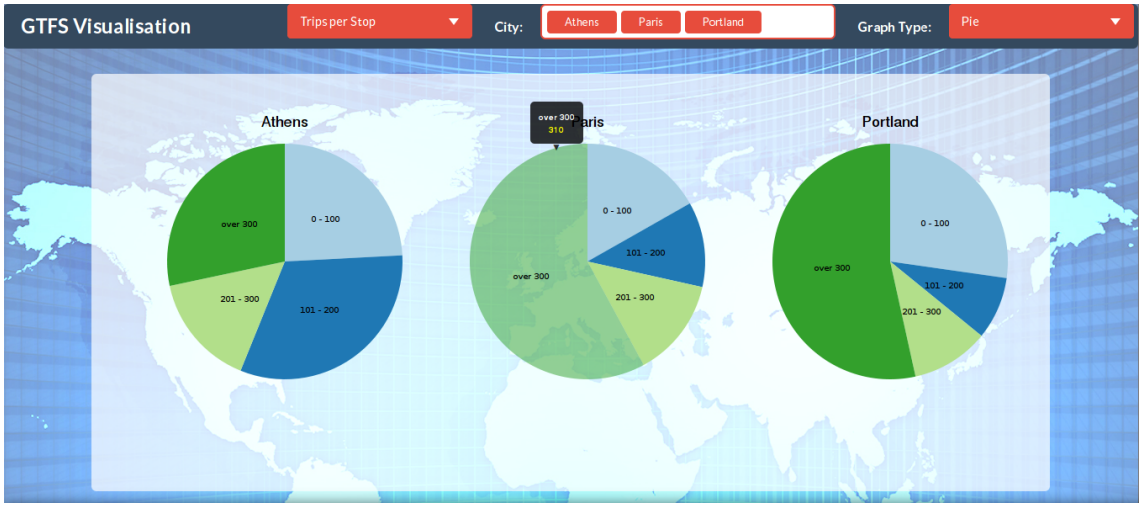


Σχήμα 7.4: Επιλογή τύπου γραφίματος

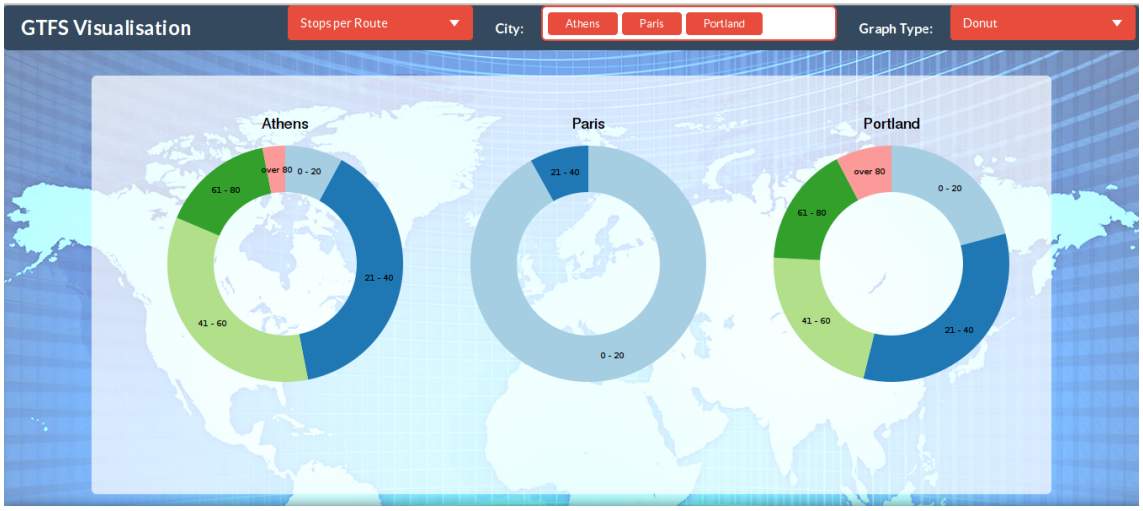
Ο χρήστης, με κατάλληλη επιλογή από τη διεπαφή, μπορεί να δει τα διαγράμματα που τον ενδιαφέρουν. Στις εικόνες 7.5, 7.6, 7.7 φαίνονται κάποια στιγμιότυπα χρήσης της web εφαρμογής.



Σχήμα 7.5: Σύγκριση των μετρήσεων για τη χρονική διάρκεια ανά δρομολόγιο (Duration per Route) αποτυπωμένη σε μορφή μπάρας



Σχήμα 7.6: Σύγκριση των μετρήσεων για τον αριθμό των ταξιδιών ανά στάση (Trips per Stop) αποτυπωμένη σε μορφή πίτας



Σχήμα 7.7: Σύγκριση των μετρήσεων για τον αριθμό των στάσεων ανά δρομολόγιο (Stops per Route) αποτυπωμένη σε μορφή donut

7.2 Συμπεράσματα

Από την εκτέλεση της εφαρμογής για τις τρεις πόλεις διεξάχθηκαν κάποια συμπεράσματα που αφορούν την πολυπλοκότητα ανάμεσα στα δεδομένα διαφορετικών πόλεων. Με μία πρόχειρη ματιά στα αποτελέσματα όπως παρουσιάζονται από την εφαρμογή παρατηρούμε τα εξής:

- Το Πόρτλαντ και η Αθήνα περιέχουν πιο πολλά ταξίδια ανά δρομολόγιο και στάσεις ανά δρομολόγιο από το Παρίσι. Το Παρίσι από την άλλη μεριά έχει αρκετά μεγαλύτερη απόσταση σε κάθε διαδρομή, μεγαλύτερο χρόνο και περισσότερα ταξίδια ανά στάση. Ως συμπέρασμα θα μπορούσαμε να πούμε ότι το δίκτυο Μέσων Μαζικής Μεταφοράς του Παρισιού είναι ένα δίκτυο με λιγότερες στάσεις αλλά μεγαλύτερο σε έκταση και με πιο πολλά ταξίδια από κάθε στάση από τα άλλα δύο. Αυτό το μοντέλο δικτύου παραπέμπει σε

δίκτυο με βασικό στοιχείο τα τρένα και τα μετρό έναντι των λεωφορείων ή των τρόλεϊ. Με πιο εκτενή ανάλυση από κάποιο εξειδικευμένο συγκοινωνιολόγο θα μπορούσαν να εξαχθούν ακόμα πιο χρήσιμα και λεπτομερή συμπεράσματα για τα χαρακτηριστικά και την ποιότητα ενός δικτύου Μέσων Μαζικής Μεταφοράς.

- Το δίκτυο της Αθήνας περιέχει αρκετά δρομολόγια με μηδενική ευκλείδεια απόσταση δηλαδή μηδενική απόσταση από την αφετηρία ως το τέρμα. Αυτό υποδεικνύει αρκετά κυκλικά δρομολόγια έναντι των άλλων δύο πόλεων που δεν περιέχουν κανένα. Βέβαια, όπως αναφέρθηκε και πιο πάνω μετρήθηκε και η απόσταση κάθε δρομολογίου υπολογίζοντας όλες τις διαδοχικές στάσεις ώστε να υπάρξει και συμπέρασμα για τη συνολική απόσταση του διανύει ένα Μέσο στις κυκλικές αυτές διαδρομές.

- Τα συμπεράσματα αυτά επιβεβαιώνουν τα αποτελέσματα των μετρήσεων που διεξήχθησαν στο κεφάλαιο 3 καθώς δίνουν μια εικόνα για την πολυπλοκότητα ενός δικτύου και κατ' επέκταση για το χρόνο που χρειάζεται ο αλγόριθμος εύρεσης της βέλτιστης δρομολόγησης, τουλάχιστον όσο αφορά το κομμάτι των Μέσων. Συγκεκριμένα, παρατηρούμε ότι Παρίσι μπορεί να έχει μεγαλύτερες αποστάσεις και περισσότερα ταξίδια ανά στάση αλλά ο μικρός αριθμός των στάσεων ανά δρομολόγιο και ο μικρότερος αριθμός δρομολογίων γενικά συνθέτει ένα μικρότερο γράφο συνδεσιμότητας του δικτύου από τις άλλες δύο πόλεις και συντελεί σε μικρότερο χρόνο εκτέλεσης του αλγορίθμου βέλτιστης δρομολόγησης.

Κεφάλαιο 8

Συμπεράσματα και μελλοντικές επεκτάσεις

8.1 Συμπεράσματα

Στα πλαίσια της παρούσας εργασίας μελετήθηκαν και αναλύθηκαν τεχνολογίες που αφορούν την ανάλυση και δρομολόγηση σε δίκτυα πολλαπλών Μέσων. Η μελέτη και ανάπτυξη τέτοιων εφαρμογών είναι αρκετά σημαντική λόγω της ραγδαίας αύξησής τους και της σπουδαιότητάς τους.

Μια αρκετά σημαντική εφαρμογή που μελετήθηκε και αφορά τον τομέα της δρομολόγησης με Μέσα Μαζικής Μεταφοράς είναι το OpenTripPlanner. Αφού μελετήθηκε εκτενώς η εφαρμογή και η αρχιτεκτονική της επιλέχθηκε η εκτέλεσή της σε δικό μας μηχάνημα που στεγάζεται στην υπηρεσία του Okeanos. Επιλέχθηκε η εκτέλεσή της για τρεις πόλεις την Αθήνα, το Παρίσι και το Πόρτλαντ. Για την εκτέλεση της εφαρμογής χρειάστηκε η απόκτηση και επεξεργασία των δεδομένων εισόδου της που δίνονται σε μορφή GTFS. Τα δεδομένα εισόδου ελέγχθηκαν και εισήχθησαν σε μια βάση δεδομένων για την διόρθωση τυχόν λαθών και εξαγωγή κάποιων στατιστικών αναφορικά με το μέγεθός τους. Έπειτα παραμετροποιήθηκε και εκτελέστηκε η εφαρμογή OpenTripPlanner για αυτά τα δεδομένα στο μηχάνημά μας. Καθώς η εφαρμογή αποτελεί μια διαδικτυακή υπηρεσία, είδαμε τα αποτελέσματα εκτέλεσής της τόσο γραφικά στο χάρτη του OpenStreetMaps όσο και σε μορφή XML και JSON από την προγραμματιστική διεπαφή που προσφέρει.

Ο έλεγχος όμως της εφαρμογής από το γραφικό περιβάλλον ή η λήψη ενός μεμονωμένου αποτελέσματος από την προγραμματιστική διεπαφή δεν κρίθηκε ικανή από μόνη της για τον έλεγχο των αποτελεσμάτων της εφαρμογής ή τη μέτρηση του χρόνου απόκρισής της. Για τον λόγο αυτό, αναπτύχθηκε ένα εργαλείο γραμμής εντολών που εκτελεί πολλαπλά πειράματα στην προγραμματιστική διεπαφή του OpenTripPlanner. Με τη βοήθεια του εργαλείου που αναπτύχθηκε εκτελέστηκαν πειράματα στο API του OpenTripPlanner και για τις τρεις πόλεις για τις οποίες εκτελείται. Συγκεκριμένα, εκτελέστηκαν 1000 τυχαία αιτήματα για κάθε πόλη για κάθε ένα από τα οποία αποθηκεύτηκε η απάντηση και ο χρόνος που μεσολάβησε για τη λήψη της. Από τα αποτελέσματα αυτά δημιουργήθηκαν διαγράμματα αναφορικά με τους χρόνους λήψης απάντησης. Με βάση τα διαγράμματα που δημιουργήθηκαν, αναλύθηκε η συμπεριφορά της εφαρμογής με βάση κάποιες παραμέτρους όπως ο αλγόριθμος δρομολόγησης που χρησιμοποιείται, το μέγεθος των δεδομένων της πόλης για την οποία εκτελείται, η

απόσταση των σημείων αρχής και τέλους και η κωδικοποίηση της απάντησης που αποστέλλεται. Βρέθηκε ότι ο αλγόριθμος με βάση τον A* που χρησιμοποιείται είναι σαφώς ταχύτερος από τον Raptor. Τέλος, διαπιστώθηκε ότι πράγματι για μεγαλύτερες αποστάσεις και για πιο πολύπλοκους γράφους δικτύων η εφαρμογή παρουσιάζει καθυστέρηση.

Εκτός από τα πειράματα με το εργαλείο γραμμής εντολών που αναπτύχθηκε, εκτελέστηκαν και πειράματα προσομοίωσης μεγάλου ταυτόχρονου φορτίου στον εξυπηρετητή, δηλαδή πολλών ταυτόχρονων χρηστών. Για τα πειράματα αυτά χρησιμοποιήθηκε η εφαρμογή JMeter η οποία, με δεδομένα τις τυχαίες αιτήσεις που δημιουργούνται με το εργαλείο γραμμής εντολών, εκτέλεσε πειράματα στη διεπαφή του OpenTripPlanner για τις τρεις πόλεις που εκτελέστηκε. Τα αποτελέσματα των πειραμάτων αυτών οποτικοποιήθηκαν σε διαγράμματα και διαπιστώθηκε ότι για τις πόλεις με πιο πολύπλοκο δίκτυο Μέσων οι χρόνοι απόκρισης της εφαρμογής είναι αρκετά μεγαλύτεροι. Τέλος, διαπιστώθηκε πως η υπερβολικά μεγάλη καθυστέρηση της εφαρμογής για μεγάλο αριθμό χρηστών, όταν εκτελείται με τον αλγόριθμο Raptor καθιστά τη χρήση του απαγορευτική σε περιπτώσεις μεγάλου φορτίου.

Τέλος, υλοποιήθηκε μια Web εφαρμογή για την εξαγωγή και οπτικοποίηση στατιστικών πληροφοριών που αφορούν τα δεδομένα GTFS των τριών πόλεων. Η εφαρμογή μας προσφέρει πολλές σημαντικές πληροφορίες για τα δίκτυα των Μέσων κάθε πόλης με τη μορφή διαγραμμάτων, όπως τον αριθμό των στάσεων ανά ταξίδι, την απόσταση ανά δρομολόγιο, το χρόνο ανά ταξίδι και αρκετά ακόμα στατιστικά. Με τη χρήση της εφαρμογής μπορέσαμε να διεξάγουμε συμπεράσματα για τη μορφολογία, την πολυπλοκότητα και τα χαρακτηριστικά του δικτύου πολλαπλών Μέσων τις κάθε πόλης.

8.2 Μελλοντικές επεκτάσεις

8.2.1 Εκτέλεση της εφαρμογής OpenTripPlanner

Οι πόλεις για τις οποίες επιλέχθηκε η εκτέλεση της εφαρμογής επιλέχθηκαν με τις εξής κριτήρια. Η Αθήνα επιλέχθηκε για τους προφανείς λόγους, το Παρίσι επιλέχθηκε λόγω του μεγάλου εύρους αλλά μικρής πολυπλοκότητας του δικτύου του και το Πόρτλαντ λόγω της προέλευσης της εφαρμογής OpenTripPlanner και της διαρκής ανανέωσης των δεδομένων GTFS από τους τοπικούς φορείς. Παρόλα αυτά, η εφαρμογή θα ήταν ενδιαφέρουσα να εκτελεστεί και για το τεράστιο δίκτυο Μέσων Μαζικής Μεταφοράς της Νέας Υόρκης, για το οποίο ίσως να υπάρχουν μεγάλες διαφοροποιήσεις για την εκτέλεση των διάφορων αλγορίθμων δρομολόγησης.

Επιπλέον, θα μπορούσε να εξεταστεί η περαιτέρω παραμετροποίηση της εφαρμογής OpenTripPlanner όπως η χρήση διαφορετικών ευρετικών μεθόδων κατά την εκτέλεση του αλγορίθμου A*, ο συνδυασμός πολλαπλών δεδομένων GTFS για την ίδια πόλη ή η χρήση του προτύπου GTFS-realtime για τις ανανεώσεις των δρομολογίων σε πραγματικό χρόνο.

8.2.2 Εργαλείο otp-client

Το εργαλείο γραμμής εντολών otp-client που αναπτύχθηκε θα μπορούσε να αποτελέσει βάση για τη δημιουργία αρκετών εφαρμογών εκτέλεσης πειραμάτων στη προγραμματιστική διεπαφή του OpenTripPlanner. Μέχρι τώρα τα πειράματα που εκτελεί επικεντρώνονται κυρίως στην μέτρηση των χρόνων απάντησης από την εφαρμογή και στον έλεγχο της συντακτικής

ορθότητας των αποτελεσμάτων. Τα αποτελέσματα όμως που αποθηκεύει περιέχουν πλούσια σημασιολογική πληροφορία για τις διαδρομές του υπολογίστηκαν οι οποίες θα μπορούσαν να χρησιμοποιηθούν για πολλαπλούς σκοπούς.

Μια πιθανή επέκταση θα ήταν η δημιουργία ενός ακόμα εργαλείου το οποίο χρησιμοποιεί τα αποτελέσματα που έχει συλλέξει το εργαλείο otp-client και συγκρίνει τα μονοπάτια που επιστράφηκαν ως απαντήσεις μεταξύ των διαφόρων αλγορίθμων δρομολόγησης. Με τον τρόπο αυτό θα μπορούσαν να βρεθούν τυχόν διαφοροποιήσεις στο ίδιο το περιεχόμενο των αποτελεσμάτων και ίσως οπτικοποίηση των διαφορών.

Επιπλέον, αν και το otp-client εργαλείο φάνηκε πολύ χρήσιμο στην εκτέλεση πειραμάτων μέσω της γραμμής εντολών, θα μπορούσε να συνδυαστεί με ένα γραφικό περιβάλλον για εκτέλεση πειραμάτων όπου θα επιτρέπει στο χρήστη την εκτέλεσή τους μέσω της γραφικής διεπαφής. Ένα παράδειγμα χρήσης της επέκτασης αυτής θα ήταν ο χρήστης να επιλέγει από τη διεπαφή την πόλη και τον αριθμό των τυχαίων πειραμάτων που επιθυμεί να εκτελεστούν και να λαμβάνει ως αποτέλεσμα τα συγκεντρωτικά διαγράμματα για του χρόνους απάντησης.

8.2.3 Εκτέλεση πειραμάτων προσομοίωσης πολλαπλών χρηστών

Για την εκτέλεση των πειραμάτων πολλαπλών χρηστών, ένα επιπλέον πείραμα που θα μπορούσε να πραγματοποιηθεί και να παρουσιάσει ενδιαφέρον είναι η εκτέλεση πειραμάτων για το OpenTripPlanner με τη χρήση του ενσωματωμένου server. Τα πειράματα που εκτελέστηκαν στα πλαίσια της παρούσας διπλωματικής αφορούσαν σε εξυπηρετητή Tomcat. Καθώς όμως το OpenTripPlanner προσφέρει και την επιλογή της αυτόνομης εκτέλεσης σε δικό του ενσωματωμένο εξυπηρετητή, θα είχε ενδιαφέρον η εκτέλεση πειραμάτων ταυτόχρονου φορτίου σε αυτόν καθώς αναμένουμε οι χρόνοι απάντησης αν είναι μεγαλύτεροι.

8.2.4 Εφαρμογή οπτικοποίησης δεδομένων GTFS

Η εφαρμογή οπτικοποίησης δεδομένων GTFS θα μπορούσε να επεκταθεί και να παρουσιάζει αποτελέσματα για τα δεδομένα περισσότερων πόλεων. Επιπλέον, καθώς τα δεδομένα GTFS περιέχουν πολύ πλούσια πληροφορία, η εφαρμογή θα μπορούσε να εξάγει ακόμα περισσότερα στατιστικά για τα δίκτυα των Μέσων κάθε πόλης, αν και τα περισσότερα και σημαντικότερα στατιστικά υλοποιήθηκαν στα πλαίσια της παρούσας διπλωματικής.

Βιβλιογραφία

[1] Apache JMeter. http://jmeter.apache.org.

[2] Apache Maven. https://maven.apache.org.

[3] Apache Tomcat. https://tomcat.apache.org.

[4] Common Format and MIME Type for Comma-Separated Values (CSV) Files. http://tools.ietf.org/html/rfc4180.

[5] Eclipse. http://www.eclipse.org.

[6] FeedValidator command line tool. https://github.com/google/transitfeed/wiki/FeedValidator.

[7] Flat UI Kit. https://designmodo.github.io/Flat-UI/.

[8] General Transit Feed Specification. https://developers.google.com/transit/gtfs.

[9] GTFS Data Exchange. http://gtfs-data-exchange.com.

[10] GTFS-realtime. https://developers.google.com/transit/gtfs-realtime.

[11] Haversine formula. https://en.wikipedia.org/wiki/Haversine_formula.

[12] Java. http://www.java.com.

[13] Jersey RESTful Web Services in Java. https://jersey.java.net/.

[14] Keyhole Markup Language. https://developers.google.com/kml.

[15] Okeanos cloud service. https://okeanos.grnet.gr.

[16] OpenTripPlanner API documentation. http://dev.opentripplanner.org/apidoc/0.12.0/.

[17] OpenTripPlanner API documentation for planning request. http://dev.opentripplanner.org/apidoc/0.12.0/resource_Planner.html.

[18] OpenTripPlanner API documentation for response. http://dev.opentripplanner.org/apidoc/0.12.0/ns0_response.html.

[19] OpenTripPlanner home page. http://www.opentripplanner.org.

[20] pgAdmin. http://www.pgadmin.org/.

[21] PostgreSQL. http://www.postgresql.org.

[22] Project Grizzly. https://grizzly.java.net/.

[23] SQL standard. http://www.iso.org/iso/catalogue_detail.htm?csnumber=45498.

[24] The Apache Ant Project. https://ant.apache.org.

[25] The GNU Compiler for Java. http://gcc.gnu.org/java/.

[26] WIGeoChart Library. http://webgistu.wigeogis.com/protozone/gm/d3chart/start.php.

[27] H. Bast. Car or public transport—two worlds. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, volume 5760 of *Lecture Notes in Computer Science*, pages 355--367. Springer Berlin Heidelberg, 2009.

[28] H. Bast, D. Delling, A. V. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route planning in transportation networks. *CoRR*, abs/1504.05140, 2015.

[29] D. Delling, T. Pajor, and R. F. Werneck. Round-based public transit routing. In *Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12)*. Society for Industrial and Applied Mathematics, 2012.

[30] D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering route planning algorithms. In J. Lerner, D. Wagner, and K. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117--139. Springer Berlin Heidelberg, 2009.

[31] A. Efentakis, S. Brakatsoulas, N. Grivas, G. Lamprianidis, K. Patroumpas, and D. Pfoser. Towards a flexible and scalable fleet management service. In *Proceedings of the 6th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS '13, pages 79--84, New York, NY, USA, 2013.

[32] A. Efentakis, N. Grivas, G. Lamprianidis, G. Magenschab, and D. Pfoser. Isochrones, traffic and demographics. In Knoblock et al. [38], pages 538--541.

[33] A. Efentakis and D. Pfoser. Optimizing landmark-based routing and preprocessing. In *Proceedings of the 6th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, IWCTS '13, pages 25--30, New York, NY, USA, 2013.

[34] A. Efentakis and D. Pfoser. GRASP. Extending graph separators for the single-source shortest-path problem. In A. Schulz and D. Wagner, editors, *Algorithms - ESA 2014*, volume 8737 of *Lecture Notes in Computer Science*, pages 358--370. Springer Berlin Heidelberg, 2014.

[35] A. Efentakis, D. Pfoser, and Y. Vassiliou. SALT. A unified framework for all shortest-path query variants on road networks. *CoRR*, abs/1411.0257, 2014.

[36] A. Efentakis, D. Theodorakis, and D. Pfoser. Crowdsourcing computing resources for shortest-path computation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12. ACM, 2012.

[37] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *16th ACM-SIAM Symposium on Discrete Algorithms*, pages 156--165, 2004.

[38] C. A. Knoblock, M. Schneider, P. Kröger, J. Krumm, and P. Widmayer, editors. *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*. ACM, 2013.

[39] OpenStreetMap Wiki. Main Page --- OpenStreetMap Wiki,, 2014. [Online; accessed 25-May-2015].

[40] B. S. Stewart and C. C. White, III. Multiobjective a*. *J. ACM*, 38(4):775--814, Oct. 1991.