ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

«Δρομολόγηση με χρήση δεδομένων OpenStreetMap»

Πτυχιακή εργασία του φοιτητή

Παναγιώτη Καρκάνη

Α.Μ 21013

Επιβλέπων καθηγητής:

Δημήτριος Μιχαήλ

Αθήνα 2015

Περιεχόμενα

Περίληψη

Η δρομολόγηση σε οδικά δίκτυα είναι ένα ζήτημα το οποίο έχει αποτελέσει αντικείμενο μελέτης και συνεχίζει να αναπτύσσεται. Στην παρούσα πτυχιακή εργασία παρουσιάζονται τα σημαντικότερα στοιχεία για την μοντελοποίηση του προβλήματος, καθώς και διάφορες τεχνικές και προσεγγίσεις για την επίλυση του. Τέλος, παρουσιάζεται αναλυτικά μια διαδικτυακή εφαρμογή η οποία παρέχει την υπηρεσία της δρομολόγησης, βρίσκοντας την συντομότερη διαδρομή από άποψη χρόνου.

Εισαγωγή

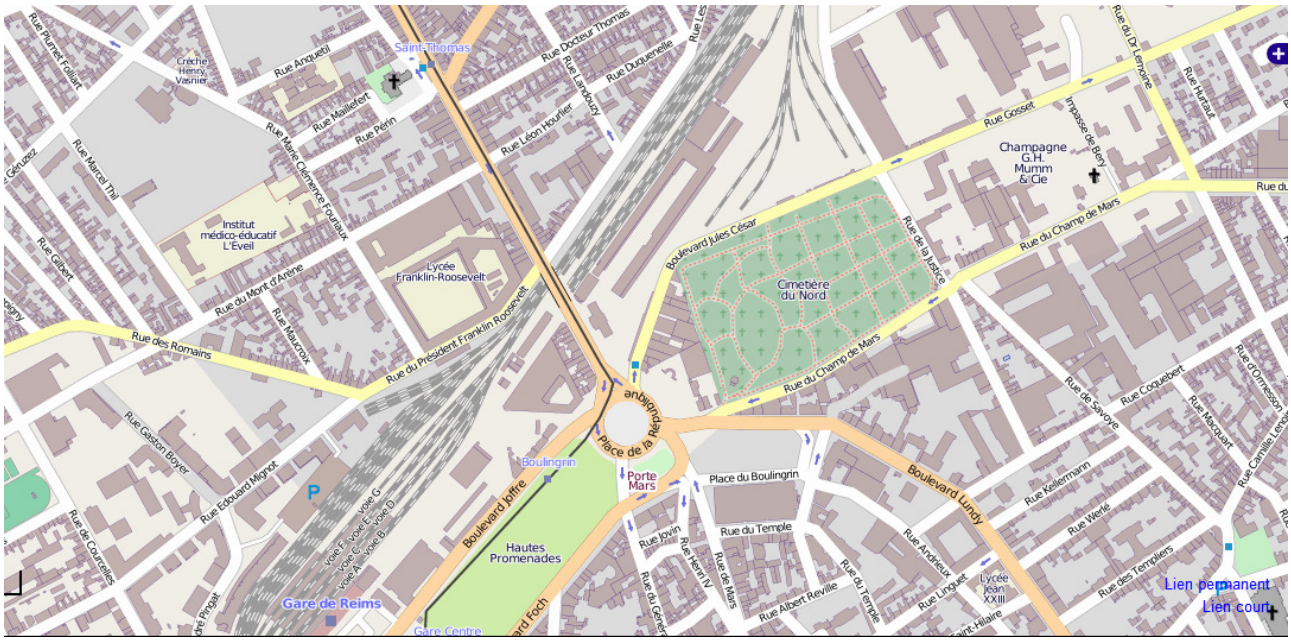
Ο όρος δρομολόγηση αναφέρεται στην διαδικασία εύρεσης διαδρομής από ένα αρχικό σημείο ενδιαφέροντος προς ένα τελικό σημείο ενδιαφέροντος. Πιο συγκεκριμένα, στα οδικά δίκτυα η δρομολόγηση αφορά την εύρεση της διαδρομής, δηλαδή των οδών που παρεμβάλλονται για την μετάβαση από ένα σημείο του χάρτη σε ένα άλλο. Η διαδικασία της δρομολόγησης σε οδικά δίκτυα είναι ένα θεμελιώδες πρόβλημα και έχει αποτελέσει αντικείμενο μελέτης και έρευνας κυρίως από τον κλάδο της πληροφορικής, διότι προσφέρει λύσεις σε ένα ευρύ φάσμα προβλημάτων. Μερικές από αυτές είναι πιο απλές, όπως απλά η ένδειξη πορείας σε περιπτώσεις που ο χρήστης δεν είναι εξοικειωμένος με την περιοχή, αλλά και πιο σύνθετες, όπως η εύρεση της πιο απλής ή πιο γρήγορης ή πιο οικονομικής διαδρομής. Επιπλέον, αξίζει να σημειωθεί ότι η δρομολόγηση είναι αρκετά σημαντική και σε επίπεδο επιχειρήσεων και οργανισμών, καθώς προσφέρει λύσεις που επιφέρουν κέρδος, όπως για παράδειγμα την επιλογή της βέλτιστης πορείας για την κάλυψη των δικτύων διανομής τους.

Η δημοφιλέστερη προσέγγιση για την δόμηση των δεδομένων έτσι ώστε να επιτευχθεί στην συνέχεια δρομολόγηση, είναι η δημιουργία γραφήματος. Ο λόγος που η συγκεκριμένη προσέγγιση είναι τόσο δημοφιλής είναι λόγω του πλήθους των τεχνικών και αλγορίθμων που έχουν αναπτυχθεί για αναζήτηση και εύρεση μονοπατιών σε γραφήματα, γεγονός που καθιστά την διαδικασία ευκολότερη. Ένα γράφημα αποτελείται απο ένα σύνολο κόμβων και ακμών. Στην περίπτωση της μοντελοποίησης οδικών δικτύων συγκεκριμένα, χρησιμοποιείται ένα κατευθυνόμενο γράφημα με μη αρνητικές ακμές. Οι κόμβοι του γραφήματος αναπαριστούν σημεία στο χάρτη ενώ οι κατευθυνόμενες ακμές τους δρόμους που ενώνουν τα σημεία αυτά και διαθέτουν ένα βάρος. Το βάρος αυτό, εξαρτάται απο την φύση του προβλήματος. Δηλαδή, για παράδειγμα, αν ο σκοπός της δρομολόγησης είναι η εύρεση της λιγότερο χρονοβόρας διαδρομής τότε το βάρος αναπαριστά το χρόνο προσπέλασης του δρόμου, ενώ αν ο σκοπός είναι η εύρεση της οικονομικότερης διαδρομής από άποψη κατανάλωσης καυσίμου, το βάρος αναπαριστά την κατανάλωση καυσίμου που χρειάζεται για την προσπέλαση του συγκεκριμένου δρόμου. Με την συγκεκριμένη δομή, μπορούν να εκτελεστούν διάφοροι αλγόριθμοι εύρεσης διαδρομής μεταξύ κόμβων του γραφήματος. Ιδιαίτερη σημασία παρουσιάζει επίσης και η τεχνική με την οποία η δρομολόγηση μπορεί να επιτευχθεί από και προς οπουδήποτε. Στην ουσία ο χρήστης δεν περιορίζεται στην εκκίνηση και στον τερματισμό της δρομολόγησης μόνο σε κόμβους του γραφήματος. Αντιθέτως, επιλέγει τα σημεία εκκίνησης και τερματισμού και στην συνέχεια αναζητούνται κόμβοι του γραφήματος που βρίσκονται πιο κοντά στα δοθέντα σημεία.

Παλαιότερα, με την έλλειψη των ηλεκτρονικών χαρτών και της δρομολόγησης ο μόνος τρόπος γνώσης της πορείας που θα πρέπει να ακολουθηθεί ήταν με τους έγγραφους χάρτες. Η διαδικασία αυτή ήταν εξαιρετικά δύσκολη και χρονοβόρα ενώ καθιστούσε αδύνατη τη γνώση για την βέλτιστη διαδρομή απο οποιαδήποτε άποψη. Με την πάροδο του χρόνου, ξεκίνησε σταδιακά η συλλογή δεδομένων και η ηλεκτρονική χαρτογράφηση περιοχών, γεγονός που είχε ως αποτέλεσμα την σταδιακή ανάπτυξη αλγορίθμων και τεχνικών για την δρομολόγηση στα οδικά δίκτυα αλλά και πιο γενικά την χρήση των δεδομένων αυτών.

Στις μέρες μας, η ύπαρξη και η χρήση γεωγραφικών δεδομένων έχουν γνωρίσει μεγάλη ανάπτυξη. Μεγάλες εταιρίες όπως η Google και η Microsoft, έχουν ασχοληθεί με γεωγραφικά

δεδομένα, πραγματοποιώντας μια αξιοσημείωτη προσπάθεια χαρτογράφησης του μεγαλύτερου μέρους του πλανήτη, παρέχοντας πληθώρα υπηρεσιών σε εταιρίες και οργανισμούς αλλά και σε απλούς χρήστες. Πιο συγκεκριμένα, πλέον ο κάθε χρήστης έχει την δυνατότητα να περιηγηθεί στο μεγαλύτερο μέρος του πλανήτη, βλέποντας ακόμα και φωτογραφίες. Επιπλέον, παρέχονται υπηρεσίες δρομολόγησης από οποιοδήποτε προς οποιοδήποτε σημείο του πλανήτη, με πλήρη πλοήγηση και με χρόνο απόκρισης μόλις λίγα δευτερόλεπτα ενώ ταυτόχρονα παρουσιάζεται η δυνατότητα επιλογής μέσου μεταφοράς αλλά και γενικά άλλων επιλογών. Επιπλέον, πέρα από εταιρίες και επιχειρήσεις, υπάρχουν και οργανισμοί οι οποίοι έχουν ασχοληθεί με γεωγραφικά δεδομένα παρέχοντας ένα αξιοσημείωτο μέγεθος ανοικτών δεδομένων, καθώς και πληροφοριών και υπηρεσιών που βασίζονται σε αυτά. Ο πιο σημαντικός τέτοιος οργανισμός είναι το OpenStreetMap [19] ο οποίος με την σταδιακή και συνεχόμενη ανάπτυξή του έχει καταφέρει να συγκεντρώσει ένα αρκετά σημαντικό μέγεθος ανοιχτών δεδομένων, με αποτέλεσμα να έχουν αναπτυχθεί αρκετά εργαλεία και εφαρμογές κάνοντας χρήση αυτών.



Εικόνα 1: Παράδειγμα ηλεκτρονικού χάρτη [11]

OpenStreetMap

Το OpenstreetMap είναι ένα collaborative project που δημιουργήθηκε και αναπτύχθηκε από εθελοντές με σκοπό την δημιουργία ενός ανοιχτού, τροποποιήσιμου και επεκτάσιμου χάρτη, με ανοιχτά γεωγραφικά δεδομένα προς οποιαδήποτε χρήση. Εμπνεύστηκε και ιδρύθηκε από τον Steve Coast το 2004, ενώ στην αρχή είχε εμβέλεια μόνο στην περιοχή του Ηνωμένου Βασιλείου.

Πλέον, το OpenStreetMap παρέχει πληροφορίες για όλο τον κόσμο και έχουν αναπτυχθεί αρκετές εφαρμογές που κάνουν χρήση των δεδομένων και των υπηρεσιών του. Τα δεδομένα αυτά συγκεντρώνονται από την συνεχή συνεισφορά των χρηστών που συμβάλλουν στην χαρτογράφηση περιοχών που δεν υπάρχουν ήδη καταχωρημένες πληροφορίες, αλλά και στην βελτίωση και ενημέρωση των υπαρχόντων περιοχών. Επιπλέον, περιλαμβάνονται δεδομένα ανοικτής άδειας από εθνικές υπηρεσίες χαρτογράφησης και άλλες πηγές. Τα δεδομένα του είναι αδειοδοτημένα υπό την Open Data Commons Open Database License(ODbL) ενώ το OpenStreetMap υποστηρίζεται από το OpenStreetMap foundation το οποίο αποτελεί έναν μη κερδοσκοπικό οργανισμό που έχει σκοπό την υποστήριξη ανοιχτών γεωγραφικών δεδομένων.

Μοντελοποίηση Προβλήματος

Το πρόβλημα εύρεσης της συντομότερης διαδρομής (shortest path problem) σε οδικά δίκτυα αλλά και γενικά σε όλους τους τύπους δικτύων, αποτελεί ένα θεμελιώδες πρόβλημα της σύγχρονης επιστήμης των υπολογιστών και είναι αντικείμενο μελέτης, έρευνας και χρήσης. Για την επίλυση προβλημάτων σε οδικά δίκτυα, ιδιαίτερα σημαντική είναι η συμβολή της θεωρίας των γράφων, διότι η σύσταση και η δομή τους είναι κατάλληλη για την αναπαράσταση, την μοντελοποίηση και την προσομοίωση των οδικών δικτύων, των οδών, των σημείων ενδιαφέροντος, καθώς και την σύνδεση όλων των παραπάνω.

Στο κεφάλαιο αυτό αναφέρονται και εξετάζονται ορισμένα στοιχεία που αφορούν τα γραφήματα και την θεωρία τους, καθώς και το πρόβλημα της συντομότερης διαδρομής. Επιπλέον, παρουσιάζεται η χρήση τους για την μοντελοποίηση του προβλήματος της δρομολόγησης στα οδικά δίκτυα καθώς και τεχνικές που χρησιμοποιούνται για την μοντελοποίηση των οδικών δικτύων.

Ένα γράφημα G(V,E) είναι μια δομή δεδομένων που αποτελείται από ένα σύνολο κόμβων V μεγέθους n και ένα σύνολο ακμών E μεγέθους m. Σε κάθε ακμή (u,υ) ανατίθεται και ένα βάρος w(u,υ) το οποίο προέρχεται από μια συνάρτηση βάρους. Ένα μονοπάτι P στο γράφημα G από έναν κόμβο u_1 προς έναν κόμβο u_k ορίζεται ως μια ακολουθία ακμών (u_1,u_2), (u_2,u_3),…,(u_{k-1},u_k). Συχνά, το μονοπάτι εκφράζεται και ως μια ακολουθία κόμβων $<u_1,u_2,\ldots,u_k>$ ή ως ένα σύνολο κόμβων $\{u_1,u_2,\ldots,u_k\}$. Ως μήκος w(P) ενός μονοπατιού P ορίζεται το άθροισμα των βαρών των ακμών που ανήκουν στο μονοπάτι P. Επιπλέον, ένα μονοπάτι $P^*=\{s,\ldots,t\}$ αναφέρεται ως συντομότερο αν δεν υπάρχει άλλο μονοπάτι P’ από το s προς το t που να ισχύει $w(P')<w(P^*)$. Η απόσταση $d_G(s,t)$, είναι το μήκος του συντομότερου μονοπατιού από το s στο t ή ∞ σε περίπτωση που δεν υπάρχει μονοπάτι από το s στο t.

Τα γραφήματα χωρίζονται σε μερικές κατηγορίες σύμφωνα με διάφορα κριτήρια. Όσον αφορά τον αριθμό των ακμών που ενώνουν δύο κόμβους τα γραφήματα χωρίζονται σε απλά, πολύ-γραφήματα και ψευδό-γραφήματα. Στα απλά γραφήματα, δύο κόμβοι u1, u2 μπορούν να συνδέονται μεταξύ τους με μέχρι μόνο μια ακμή. Αντίθετα, τα πολύ-γραφήματα επιτρέπουν παραπάνω από μια ακμές μεταξύ δύο κόμβων. Τέλος, τα γραφήματα που επιτρέπουν μια ακμή (t,t), δηλαδή μια ακμή που συνδέει έναν κόμβο με τον εαυτό του, ονομάζονται ψευδό-γραφήματα. Τα γραφήματα χωρίζονται επιπλέον σε κατευθυνόμενα και μη κατευθυνόμενα. Τα γραφήματα που το σύνολο των ακμών τους δεν περιέχει κάποια κατεύθυνση ονομάζονται μη κατευθυνόμενα. Σε διαφορετική περίπτωση, όταν σε ένα γράφημα υπάρχουν ακμές που διαθέτουν κατεύθυνση, τότε αποκαλείται κατευθυνόμενο.

Όσον αφορά την χρήση γραφημάτων για την αναπαράσταση οδικών δικτύων, συνήθως χρησιμοποιείται ένα κατευθυνόμενο γράφημα. Κάθε κόμβος αναπαριστά μια διασταύρωση οδών και κάθε ακμή μια οδό ή ένα τμήμα της. Το γράφημα είναι κατευθυνόμενο διότι δίνει την δυνατότητα αναπαράστασης της επιτρεπόμενης πορείας στον δρόμο. Δηλαδή αν είναι διπλής κατεύθυνσης, ή μονόδρομος, ενώ σε περίπτωση που είναι μονόδρομος αναπαρίσταται και η επιτρεπόμενη κατεύθυνση. Επιπλέον, το βάρος της κάθε ακμής αναπαριστά το κόστος προσπέλασης της. Το κόστος εξαρτάται από την φύση του προβλήματος. Πιο συγκεκριμένα, αν ο σκοπός της δημιουργίας του γραφήματος είναι η εύρεση της πιο οικονομικής, από άποψη κατανάλωσης, διαδρομής τότε το βάρος της κάθε ακμής αναπαριστά την κατανάλωση καυσίμου

που θα χρειαστεί για την προσπέλαση της. Αν ο σκοπός είναι η εύρεση της γρηγορότερης, από άποψη χρόνου, διαδρομής το βάρος αφορά τον χρόνο προσπέλασης της ακμής.

Το πρόβλημα της συντομότερης διαδρομής αφορά την εύρεση ενός μονοπατιού μεταξύ δύο κόμβων σε ένα γράφημα, του οποίου το άθροισμα των βαρών είναι το μικρότερο που μπορεί να επιτευχθεί. Στην θεωρία των γράφων, το παραπάνω αποτελεί θεμελιώδες πρόβλημα και παρουσιάζεται με διάφορες παραλλαγές. Οι σημαντικότερες παρουσιάζονται παρακάτω:

- point-to-point: υπολογισμός του συντομότερου μονοπατιού από έναν κόμβο εκκίνησης s προς ένα κόμβο τερματισμού t.
- single-source: για έναν συγκεκριμένο κόμβο εκκίνησης, υπολογισμός του συντομότερου μονοπατιού για όλους τους υπόλοιπους κόμβους του γραφήματος.
- All-pairs shortest paths: υπολογισμός του συντομότερου μονοπατιού από κάθε κόμβο προς κάθε κόμβο του γραφήματος.

Σχετικά με την εύρεση της συντομότερης διαδρομής στα οδικά δίκτυα χρησιμοποιείται η παραλλαγή point-to-point. Πιο συγκεκριμένα, δοθέντος ενός κόμβου εκκίνησης και ενός κόμβου τερματισμού υπολογίζεται στο γράφημα η συντομότερη διαδρομή/μονοπάτι μεταξύ των δύο αυτών κόμβων. Η διαδρομή που βρίσκεται στο γράφημα, αναπαριστά την διαδρομή που πρέπει να ακολουθηθεί στο οδικό δίκτυο για την μετάβαση από το σημείο εκκίνησης προς το σημείο τερματισμού.

Σε αυτό το σημείο αξίζει να παρουσιαστεί και η προσέγγιση που χρησιμοποιείται για την μετάβαση από ένα σημείο ενδιαφέροντος του χάρτη σε έναν κόμβο του γραφήματος, καθώς και η διαδικασία επιλογής του συγκεκριμένου κόμβου. Όπως αναφέρθηκε και στα παραπάνω, το πρόβλημα της συντομότερης διαδρομής αφορά την εύρεση του συντομότερου μονοπατιού στο γράφημα ανάμεσα σε έναν αρχικό και έναν τελικό κόμβο. Συνεπώς, πριν την εκτέλεση του αλγορίθμου, είναι απαραίτητος ο καθορισμός των δύο αυτών κόμβων. Οι χρήστες που χρησιμοποιούν εφαρμογές που παρέχουν υπηρεσίες δρομολόγησης δεν έχουν καμία επαφή με το γράφημα, ούτε γνωρίζουν την ύπαρξη του. Συνήθως, εισάγουν τα σημεία εκκίνησης και τερματισμού δίνοντας τις συντεταγμένες τους ή χρησιμοποιώντας κάποιον χάρτη γραφικά. Έτσι, χρίζεται απαραίτητη η διαδικασία εύρεσης των κόμβων που θα χρησιμοποιηθούν ανάλογα με την επιλογή των χρηστών. Πιο συγκεκριμένα, για κάθε σημείο που παρέχεται από τον χρήστη ακολουθεί μια διαδικασία εύρεσης του πιο κοντινού κόμβου του γραφήματος ως προς το σημείο αυτό. Έτσι, και για το σημείο εκκίνησης και για το σημείο τερματισμού, βρίσκεται ο πιο κοντινός κόμβος και στην συνέχεια πραγματοποιείται η διαδικασία εύρεσης της διαδρομής για τους δύο κόμβους που έχουν επιλεχθεί.

Για την επίτευξη της παραπάνω διαδικασίας, δηλαδή την διαχείριση γεωγραφικών δεδομένων αποδοτικά και την εκτέλεση γεωγραφικών ερωτημάτων, στην συγκεκριμένη περίπτωση την εύρεση του κοντινότερου κόμβου, συνήθως γίνεται χρήση τεχνικών ευρετηριοποίησης, όπως αυτής των δέντρων R (R-trees [6][7]) αλλά και των Quadtree[5]. Τα δέντρα R αποτελούν μια επέκταση των δέντρων B σε παραπάνω από μια διαστάσεις. Τα Quadtrees αποτελούν επίσης μια δομή δεδομένων δέντρου όπου κάθε εσωτερικός κόμβος έχει ακριβώς τέσσερα παιδιά. Και οι δυο

τεχνικές προσφέρουν καλύτερη και αποδοτικότερη λύση στην αναπαράσταση και αποθήκευση γεωγραφικών δεδομένων καθώς επίσης και στις διαδικασίες αναζήτησης και τροποποίησης τους.

Η πιο συνηθισμένη μεθοδολογία για την εύρεση του κοντινότερου κόμβου ως προς τον σημείο που έδωσε ως είσοδο ο χρήστης, είναι η αναζήτηση των κοντινότερων γειτόνων. Η συγκεκριμένη τακτική, λαμβάνει υπόψιν το σημείο εισόδου και αναζητεί τους κοντινότερους κόμβους ως προς το σημείο αυτό. Στην συνέχεια επιλέγεται ο καταλληλότερος από τους κόμβους που έχουν βρεθεί και χρησιμοποιείται για την εύρεση της διαδρομής.

Αλγόριθμοι Δρομολόγησης

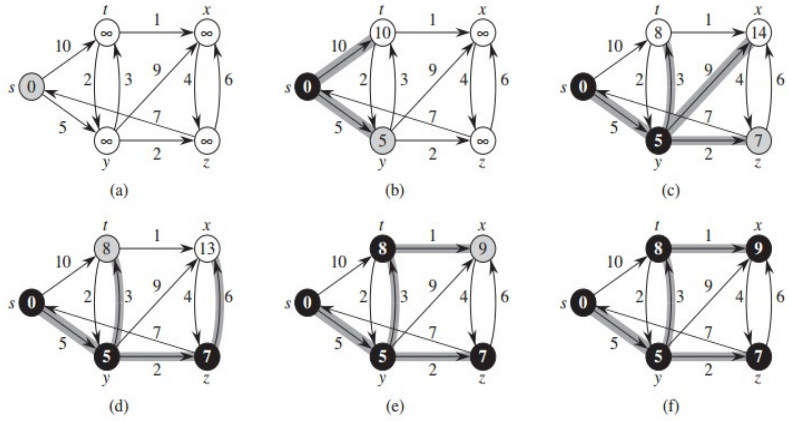
Το πρόβλημα υπολογισμού της βέλτιστης διαδρομής είναι ένα ιδιαίτερο ζήτημα και αποτελεί αντικείμενο ενασχόλησης για πολλά χρόνια. Αρκετοί αλγόριθμοι και υλοποιήσεις έχουν αναπτυχθεί για την αντιμετώπιση του προβλήματος. Η πιο γνωστή προσέγγιση είναι αυτή του αλγορίθμου Dijkstra[4]. Όμως, λόγω της συνεχόμενης αύξησης των απαιτήσεων απόκρισης και των δεδομένων, τέτοιες τεχνικές έχουν ξεπεραστεί λόγω της αδυναμίας τους να παρέχουν λύση στο απαιτούμενο χρονικό πλαίσιο. Για τον λόγο αυτό αναπτύχθηκε μια πληθώρα τεχνικών και αλγορίθμων, καθώς και συνδυασμός τους στην προσπάθεια επιτάχυνσης της διαδικασίας και της επίτευξης όλο και γρηγορότερων χρόνων απόκρισης. Παρακάτω, παρατίθενται μερικοί αλγόριθμοι και τεχνικές, από πιο κλασσικές όπως ο αμφίδρομος Dijkstra και ο A-άστρο, καθώς και πιο σύγχρονες, όπως το ιεραρχικό μοντέλο και η χρήση landmarks [2][3][8][9][10].

Σε αυτό το σημείο αξίζει να σημειωθεί ότι για την επίτευξη της δρομολόγησης, όλες οι τεχνικές χρησιμοποιούν ένα κατευθυνόμενο γράφημα G=(V,E) όπου V είναι το σύνολο των κόμβων και E το σύνολο των κατευθυνόμενων ακμών.

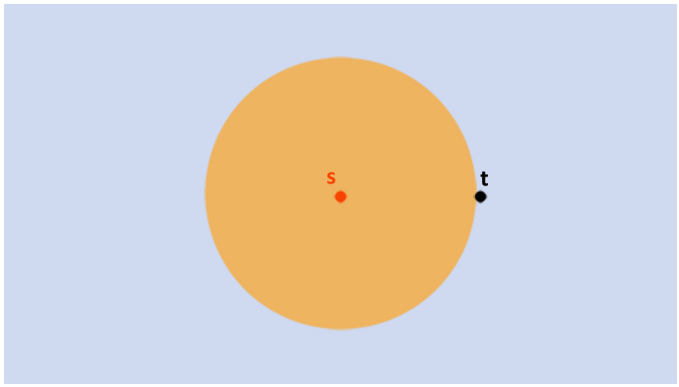
Αλγόριθμος Dijkstra

Ο κλασσικός αλγόριθμος για τον υπολογισμό του συντομότερου μονοπατιού μεταξύ ενός αρχικού και ενός τελικού κόμβου του γραφήματος είναι ο αλγόριθμος Dijkstra[3][8]. Ο αλγόριθμος διατηρεί για κάθε κόμβο u, ένα πίνακα δοκιμαστικών αποστάσεων distance[u] με την απόσταση του κόμβου u από τον αρχικό κόμβο. Μια ουρά προτεραιότητας Q περιέχει όλους του κόμβους που αποτελούν το τρέχον μέτωπο αναζήτησης. Σε κάθε βήμα του, ο αλγόριθμος διαγράφει απο το μέτωπο αναζήτησης τον κόμβο u με την μικρότερη απόσταση απο τον αρχικό κόμβο s. Στην συνέχεια, χαλαρώνονται, δηλαδή επισκέπτονται, οι εξερχόμενες ακμές (u,υ) του κόμβου που ανακαλύφθηκε. Έπειτα ελέγχεται η συνθήκη d(s,u) + length(u,υ) < distance[υ]. Στην ουσία δηλαδή, ελέγχεται αν διαμέσω του νέου κόμβου που ανακαλύφθηκε βρέθηκε κάποια καλύτερη διαδρομή προς κάποιον κόμβο υ. Αν ισχύει η παραπάνω συνθήκη, βρέθηκε καλύτερη διαδρομή προς τον κόμβο υ. Ο αλγόριθμος Dijkstra τερματίζεται, όταν έχουν αφαιρεθεί όλοι οι κόμβοι απο το μέτωπο αναζήτησης.

Όπως φαίνεται απο την σύντομη περιγραφή του, ο αλγόριθμος Dijkstra επισκέπτεται όλους τους κόμβους του μετώπου αναζήτησης, γεγονός που τον χρίζει ακατάλληλο για αναζητήσεις σε γραφήματα που τα δεδομένα του μπορούν να περιλαμβάνουν πληροφορίες για όλο τον πλανήτη. Παρόλα αυτά, η σημασία του αλγορίθμου είναι τεράστια διότι αποτέλεσε την βάση για την ανάπτυξη αποδοτικότερων τεχνικών και αλγορίθμων που χρησιμοποιούνται σήμερα.



Εικόνα 2: Παράδειγμα εκτέλεσης αλγορίθμου Dijkstra που παρουσιάζεται ο τρόπος προσπέλασης των κόμβων.[1]



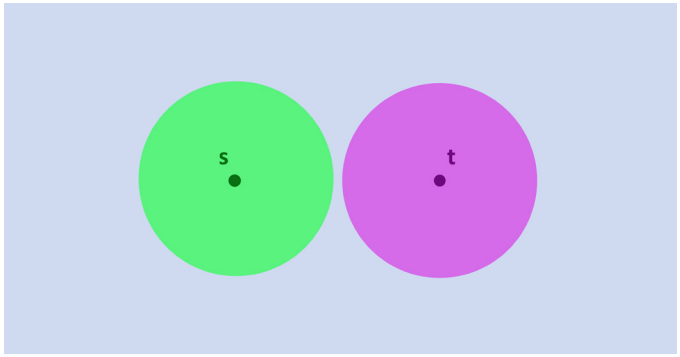
Εικόνα 3:Σχηματική αναπαράσταση του μετώπου αναζήτησης του αλγορίθμου Dijkstra.

Αμφίδρομη Προσέγγιση-Αμφίδρομος Dijkstra

Η πιο απλή τεχνική επιτάχυνσης του αλγορίθμου Dijktra είναι αυτή του αμφίδρομου Dijkstra[3][8][9][10]. Πιο συγκεκριμένα, ο αλγόριθμος Dijkstra εκτελείται ταυτόχρονα και από τον κόμβο εκκίνησης προς τον κόμβο τερματισμού και από τον κόμβο τερματισμού προς τον κόμβο εκκίνησης. Η διαδικασία σταματάει όταν κάποιος κόμβος προσπελαστεί και απο τις δύο κατευθύνσεις. Η τελική διαδρομή σχηματίζεται απο τις πληροφορίες που έχουν μαζευτεί απο τις δύο αναζητήσεις.

Αξίζει να σημειωθεί, ότι για την εκτέλεση του αμφίδρομου Dijkstra και συγκεκριμένα για την αναζήτηση από τον κόμβο τερματισμού προς τον κόμβο εκκίνησης, το γράφημα πρέπει να αντιστραφεί. Ουσιαστικά το αντιστραμμένο γράφημα αποτελείται από το ίδιο σύνολο κόμβων V, αλλά αντίστροφο σύνολο ακμών E'={ (u, υ) | (υ, u) ανήκει στο E }.

[1] Από Cormen et al. (2001) σελ 596

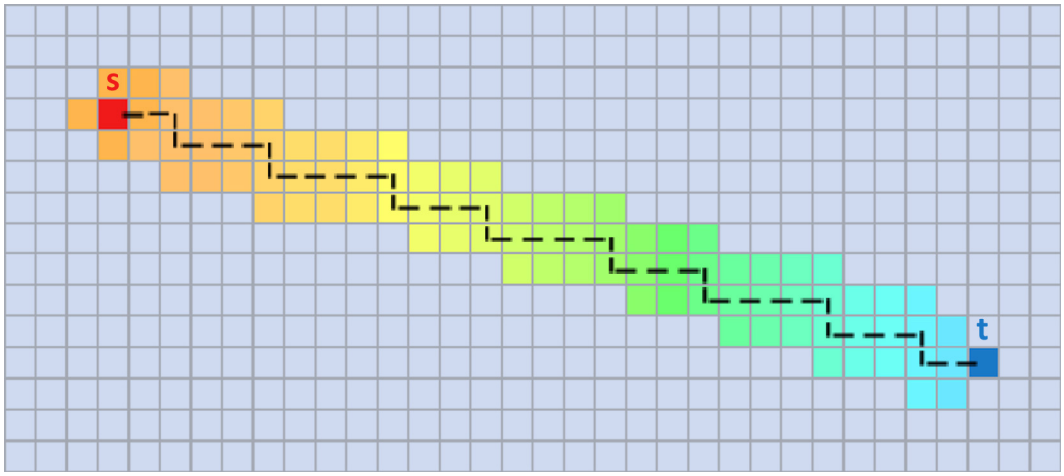


Εικόνα 4: Σχηματική αναπαράσταση του μετώπου αναζήτησης του αμφίδρομου αλγορίθμου Dijkstra.

Αλγόριθμοι οδηγούμενοι από τον στόχο-Αλγόριθμος Α-Άστρο(Α*)

Η χρήση του αλγορίθμου Α-άστρο [8][9][10] είναι μια τεχνική, εμπνευσμένη από την τεχνητή νοημοσύνη. Ο αλγόριθμος ανήκει στην κατηγορία των αλγόριθμων ευριστικής αναζήτησης διότι αλλάζει την προτεραιότητα προσπέλασης των κόμβων με την συμβολή ενός ευριστικού μηχανισμού. Πιο συγκεκριμένα, ο αλγόριθμος προσθέτει στην υπάρχουσα απόσταση κάθε κόμβου distance[u] μία τιμή $p_t : V \rightarrow R^+$ η οποία εξαρτάται από τον προορισμό t. Έτσι, πλέον κάθε κόμβος έχει απόσταση ίση με το άθροισμα distance(u) + p_t(u). Με την σωστή επιλογή του ευριστικού μηχανισμού και κατ' επέκταση της ευριστικής συνάρτησης που θα χρησιμοποιηθεί για την τιμή p_t(u) κάθε κόμβου u, ο χρόνος εκτέλεσης του αλγορίθμου μπορεί να μειωθεί σε ιδιαίτερα σημαντικό βαθμό, παρέχοντας τη συντομότερη διαδρομή. Μια ιδιαίτερα γνωστή πρακτική για την υλοποίηση του ευριστικού μηχανισμού είναι αυτή της ευκλείδειας απόστασης ή της απόστασης Μανχάταν. Πιο συγκεκριμένα, στην προσέγγιση αυτή, η τιμή που απορρέει από την ευριστική συνάρτηση κάνει χρήση του τύπου της ευκλείδειας απόστασης ή της απόστασης Μανχάταν με στόχο να δοθεί μεγαλύτερη προτεραιότητα σε κόμβους που είναι πιο κοντά γεωμετρικά στον κόμβο προορισμού.

Στην εικόνα 5 παρουσιάζεται γραφικά μια εκτέλεση αναζήτησης που κάνει χρήση του αλγορίθμου Α-άστρο και του ευριστικού μηχανισμού απόστασης Manhattan. Είναι φανερό, ότι η αναζήτηση γίνεται γεωμετρικά μόνο προς το μέρος του προορισμού μειώνοντας σημαντικά το πλήθος των κόμβων που επισκέπτονται.



Εικόνα 5:Παράδειγμα εκτέλεσης αλγορίθμου Α-Άστρου με χρήση απόστασης Manhattan.

Ορόσημα – Landmarks

Η χρήση landmarks[2][9][10] είναι μια τεχνική που μελετήθηκε και χρησιμοποιήθηκε πολύ πιο πρόσφατα σε σχέση με τις τεχνικές που παρουσιάστηκαν παραπάνω. Ένα από τα σημαντικότερα στοιχεία της είναι ότι προ-επεξεργάζεται τα δεδομένα. Συγκεκριμένα, η τεχνική των landmarks αποτελείται από την επιλογή ενός υποσυνόλου κόμβων του αρχικού γραφήματος. Μετά την επιλογή των κόμβων αυτών ακολουθεί ο προ-υπολογισμός των αποστάσεων προς τους επιλεγμένους κόμβους, από όλους τους κόμβους του αρχικού γραφήματος. Με την χρήση της παραπάνω τεχνικής, διευκολύνεται η διαδικασία υπολογισμού της απόστασης μεταξύ των ζητούμενων κόμβων, συνδυάζοντας απλά τις προ-υπολογισμένες αποστάσεις. Όπως είναι κατανοητό, στην συγκεκριμένη μεθοδολογία ιδιαίτερο ενδιαφέρον παρουσιάζει η επιλογή των d κόμβων που θα θεωρηθούν ως landmarks. Για την επιλογή αυτή ακολουθούνται κυρίως οι παρακάτω στρατηγικές.

- Τυχαία επιλογή: Επιλέγονται τυχαία d κόμβοι από το γράφημα
- Βαθμός (Degree) του κόμβου: Ταξινομούνται με αύξουσα σειρά οι κόμβοι του γραφήματος με βάση τον βαθμό τους, δηλαδή τον αριθμό των ακμών που διέρχονται από αυτόν, και επιλέγονται οι n μεγαλύτεροι κόμβοι. Η λογική της στρατηγικής αυτής είναι, ότι όσο πιο συνδεδεμένος είναι ο κόμβος, τόσο σε πιο πολλά συντομότερα μονοπάτια θα συμπεριλαμβάνεται
- Κεντρικότητα(Closeness Centrality) του κόμβου: Επιλέγονται ως landmarks οι d κόμβοι που έχουν την μικρότερη closeness centrality. Η λογική της στρατηγικής αυτής είναι ότι όσο πιο κοντά βρίσκεται ένας κόμβος στους υπόλοιπους κόμβους του γραφήματος, τόσο πιθανότερο είναι να αποτελεί τμήμα περισσότερων συντομότερων διαδρομών. Σε αυτό το σημείο αξίζει να αναφερθεί ότι ο όρος closeness centrality ενός κόμβου, εκφράζει το πόσο κοντά είναι ο συγκεκριμένος κόμβος στους υπόλοιπους κόμβους του γραφήματος .

Με την χρήση Landmarks, όταν τίθεται το ερώτημα εύρεσης ενός συντομότερου μονοπατιού, η διάσχιση του γραφήματος γίνεται στο πλαίσιο των επιλεγμένων κόμβων, γεγονός που μειώνει σε τεράστιο βαθμό τον χρόνο αναζήτησης. Στην συνέχεια, η διαδρομή από τους επιλεγμένους κόμβους προς τους κόμβος εκκίνησης και τερματισμού είναι ήδη υπολογισμένη και προστίθεται σε αυτή που υπολογίστηκε κατά την εκτέλεση του ερωτήματος.

Ιεραρχικό μοντέλο

Μία άλλη τεχνική η οποία παρουσιάστηκε επίσης πρόσφατα είναι αυτή που κάνει χρήση του ιεραρχικού μοντέλου [2][3][8][9][10]. Το ιεραρχικό μοντέλο προ-επεξεργάζεται το υπάρχον γράφημα και το εμπλουτίζει με επιπλέον ακμές δημιουργώντας μια ιεραρχία, η οποία σε μεταγενέστερο στάδιο θα επιταχύνει τα ερωτήματα εύρεσης της συντομότερης διαδρομής. Η βασική ιδέα είναι η αναζήτηση να πραγματοποιείται αρχικά σε έναν <<αφαιρετικό>> χώρο και όχι σε ολόκληρο το χώρο του προβλήματος. Αναλυτικότερα, υπάρχουν δύο κυριαρχούσες τεχνικές που κάνουν χρήση του ιεραρχικού μοντέλου.

- Πολύ-επίπεδη(Multi-level) προσέγγιση: Η προσέγγιση αυτή αποσυνθέτει το γράφημα και δημιουργεί k επίπεδα . Σε κάθε επίπεδο i επιλέγει ορισμένους κόμβους Si, οι οποίοι ονομάζονται διαχωριστές. Έτσι, στο επίπεδο 0 οι διαχωριστές αποτελούνται από όλους του

κόμβους V του γραφήματος. Στο επίπεδο 1 από ένα υποσύνολο των κόμβων που αποτελούν το επίπεδο 0 κτλ. Ιδιαίτερη σημασία παρουσιάζει η επιλογή των κόμβων που θα θεωρηθούν ως διαχωριστές και η επιλογή τους εξαρτάται από διάφορα κριτήρια που ξεφεύγει από τα πλαίσια της παρούσας πτυχιακής εργασίας. Στην πολύ-επίπεδη προσέγγιση προστίθενται τρία νέα είδη ακμών που χρησιμοποιούνται για την μετάβαση από προς τα διάφορα επίπεδα. Υπάρχουν οι προς-τα-επάνω ακμές οι οποίες ενώνουν κόμβους που δεν έχουν επιλεχθεί, με κόμβους που έχουν επιλεχθεί ως διαχωριστές, οι προς-τα-κάτω ακμές οι οποίες προέρχονται από επιλεγμένους κόμβους και κατευθύνονται σε μη επιλεγμένους κόμβους, καθώς επίσης και οι ακμές επιπέδου οι οποίες ενώνουν επιλεγμένους κόμβους στο ίδιο επίπεδο.

- Ιεραρχία κύριων δρόμων(Highway Hierarchy): Η συγκεκριμένη προσέγγιση βασίζεται στην λογική, ότι μόνο ένα δίκτυο αυτοκινητόδρομων, η γενικά μεγάλων δρόμων, πρέπει να λαμβάνεται υπόψιν, όταν η διαδικασία της εύρεσης διαδρομής αφορά κόμβους που δεν βρίσκονται στην ίδια γειτονιά. Η προσέγγιση κάνει χρήση δέντρων συντομότερων μονοπατιών για να δημιουργήσει μια ιεραρχία δρόμων. Αναλυτικότερα, χρησιμοποιεί μια τροποποίηση του αλγορίθμου Dijkstra, έτσι ώστε κάθε υπό-μονοπάτι, από ένα συντομότερο μονοπάτι, να είναι πάντα και αυτό συντομότερο μονοπάτι. Με την χρήση των παραπάνω πληροφοριών, δηλαδή με την εύρεση και αποθήκευση των συντομότερων διαδρομών για κάθε κόμβο, οικοδομείται μια ιεραρχία όπου κάθε επίπεδο αποτελεί υπό-γράφημα του παρακάτω επιπέδου. Έτσι, όταν ζητηθεί ένα συντομότερο μονοπάτι, για παράδειγμα μεταξύ δύο δρόμων που βρίσκονται σε διαφορετικές χώρες, τα υψηλότερα επίπεδα διαθέτουν πληροφορίες για την μετάβαση από την μια χώρα στην άλλη. Στην συνέχεια, και όσο η αναζήτηση κατευθύνεται σε χαμηλότερα επίπεδα συντίθεται σταδιακά η διαδρομή, ανακαλύπτοντας αρχικά την διαδρομή προς την ζητούμενη πόλη και στην συνέχεια προς την ζητούμενη γειτονιά.

Στο σημείο αυτό, αξίζει να σημειωθεί, ότι είναι πολύ συνηθισμένο φαινόμενο, ο συνδυασμός τεχνικών και αλγορίθμων με σκοπό την καλύτερη απόδοση. Αρχικά, η αμφίπλευρη αναζήτηση είναι μια τεχνική η οποία μπορεί να συνδυαστεί με όλες τις τεχνικές. Επίσης, μία συνηθισμένη τεχνική είναι ο συνδυασμός του ιεραρχικού μοντέλου και τεχνικών καθοδηγούμενων από τον στόχο, όπως ο αλγόριθμος Α-άστρο. Συνάμα, ο αλγόριθμος Α-άστρο μπορεί να χρησιμοποιηθεί κάλλιστα σε συνδυασμό με την τεχνική των landmarks μαζί με την τεχνική highway hierarchy.

Αρχιτεκτονική - Συνοπτική περιγραφή της παρούσας εργασίας

Η παρούσα πτυχιακή εργασία, είναι ουσιαστικά μια διαδικτυακή εφαρμογή, server - client για την εύρεση της συντομότερης, όσον αφορά τον χρόνο, διαδρομής από ένα σημείο στο χάρτη προς ένα άλλο με την χρήση οχήματος. Οι παράγοντες που επηρεάζουν και διαμορφώνουν την καλύτερη διαδρομή, και στην ουσία την διαδρομή η οποία θα επιλεχθεί είναι οι εξής:

- Απόσταση
- Κατηγορία δρόμου (εθνική οδός, επαρχιακή οδός, οδός που κατοικείται)
- Ποιότητα δρόμου (άσφαλτος, χώμα)
- Μέγιστη επιτρεπόμενη ταχύτητα

Η υλοποίηση της αποτελείται από τέσσερα κύρια βήματα.

- Συλλογή και επεξεργασία των δεδομένων

Για την επίτευξη της συλλογής πληροφορίας σχετικά με τα οδικά δίκτυα και γενικά της χαρτογράφησης της περιοχής που αποσκοπείται να χρησιμοποιηθεί για την δρομολόγηση, χρησιμοποιούνται δεδομένα από το OpenStreetMap. Τα συγκεκριμένα δεδομένα ανακτώνται, αναλύονται, επεξεργάζονται και αποθηκεύονται κατάλληλα.

- Δημιουργία γραφήματος

Η διαδικασία δημιουργίας του γραφήματος περιλαμβάνει την συλλογή της κατάλληλης πληροφορίας και την αποθήκευση της σε μορφή γραφήματος

- Εύρεση διαδρομής στο γράφημα

Στην διαδικασία εύρεσης διαδρομής, ο χρήστης δίνει ως είσοδο δύο σημεία, ένα εκκίνησης και ένα τερματισμού και επιστρέφεται η γρηγορότερη διαδρομή στο γράφημα.

- Αποτύπωση της διαδρομής

Στην αποτύπωση της διαδρομής, εμφανίζεται στον χρήστη γραφικά η διαδρομή που έχει προκύψει από τον αλγόριθμο.

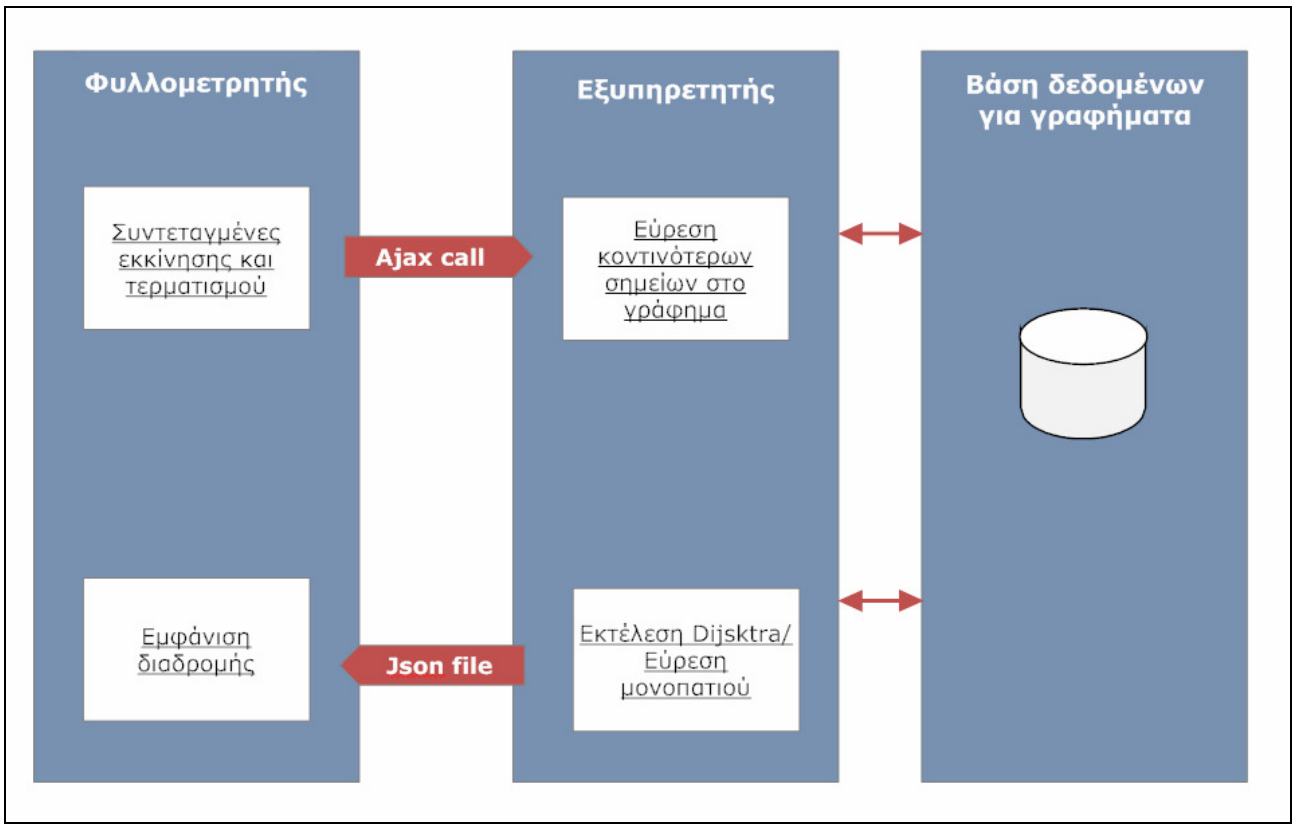
Όσον αφορά την αρχιτεκτονική, η εφαρμογή, σε γενικά πλαίσια, κάνει χρήση του εργαλείου osm2pgsql[24], της βάσης δεδομένων PostgreSQL[21], της βάσης δεδομένων για γραφήματα Neo4j[20] καθώς και μια επέκταση της για διαχείριση γεωγραφικών δεδομένων[17], της βιβλιοθήκης javascript Leaflet[18] και της γλώσσας προγραμματισμού Java. Επιπλέον χρησιμοποιεί και έναν Glassfish application-server[23]. Πιο αναλυτικά, η εφαρμογή αποτελείται απο δύο μεγάλα κομμάτια. Την δημιουργία του γραφήματος και την εκτέλεση ερωτημάτων στο υπάρχον γράφημα για την εύρεση της καλύτερης διαδρομής.

Σχετικά με την δημιουργία του γραφήματος, η αρχική πληροφορία, είναι ένα αρχείο xml το οποίο εμπεριέχει τα δεδομένα που αποσπάστηκαν από το OpenStreetMap. Το αρχείο αυτό, δίνεται ως είσοδος στο εργαλείο osm2pgsql, του οποίου η έξοδος τοποθετεί τα δεδομένα του xml αρχείου σε μια βάση δεδομένων PostgreSQL. Έπειτα τα δεδομένα διαβάζονται από την βάση

δεδομένων αυτήν, επεξεργάζονται κατάλληλα, και στην συνέχεια αποθηκεύονται στην βάση δεδομένων Neo4j με την μορφή γραφήματος. Η παραπάνω διαδικασία, δηλαδή η πορεία από το xml αρχείο μέχρι την αποθήκευση του γραφήματος αναπαρίσταται στην εικόνα 6.

Εικόνα 6: Διαδικασία φόρτωσης δεδομένων

Το δεύτερο μεγάλο κομμάτι της εφαρμογής είναι άμεσα συνδεδεμένο με την εκτέλεση της εφαρμογής. Δηλαδή, αφορά την αλληλεπίδραση του χρήστη με το σύστημα και την εκτέλεση της εφαρμογής. Πιο συγκεκριμένα, ο χρήστης, με τον φυλλομετρητή του, εκτελεί την διαδικτυακή εφαρμογή και του εμφανίζεται στην οθόνη του ένας χάρτης της περιοχής που έχει επεξεργαστεί. Η διαχείριση του χάρτη γίνεται με την χρήση της βιβλιοθήκης javascript Leaflet[18]. Με την βοήθεια του χάρτη, και πιο συγκεκριμένα με κλικ, δίνει ως είσοδο στο σύστημα τα σημεία εκκίνησης και τερματισμού της διαδρομής που επιθυμεί. Στην συνέχεια, όταν ο χρήστης πατήσει το κουμπί για τον υπολογισμό της διαδρομής, αποστέλλεται στον εξυπηρετητή, ασύγχρονα, ένα αίτημα που εμπεριέχει την είσοδο του χρήστη. Έπειτα, ο εξυπηρετητής, για κάθε ένα απο τα δύο σημεία που έδωσε ο χρήστης ψάχνει να βρει τον κόμβο του γραφήματος που να βρίσκεται πιο κοντά στο σημείο εισόδου του χρήστη. Κατόπιν, και αφού έχουν βρεθεί οι δύο κόμβοι εκκίνησης και τερματισμού, εκτελείται η διαδικασία της εύρεσης διαδρομής με την χρήση του αλγορίθμου Dijkstra. Το αποτέλεσμα, δηλαδή το σύνολο των κόμβων και των ακμών που αποτελούν την διαδρομή, αποθηκεύεται σε ένα αρχείο JSON και επιστρέφεται στον φυλλομετρητή Εκεί, αναλύεται και αποτυπώνεται στον χάρτη, με την βοήθεια της βιβλιοθήκης Leaflet, που ήδη έχει εμφανιστεί στον χρήστη. Η παραπάνω διαδικασία φαίνεται και σχηματικά στην εικόνα 7 όπου παρουσιάζεται η αρχιτεκτονική του συστήματος.



Εικόνα 7:Αρχιτεκτονική Συστήματος

Φόρτωση Δεδομένων

Σύνολο Δεδομένων OpenStreetMap

Τα δεδομένα από το OpenStreetMap βρίσκονται σε διάφορες μορφές. Οι πιο σημαντικές είναι οι OSMXML, PBFFORMAT, o5m, OSMJSON και η LEVELOL. Από τις παραπάνω μελετήθηκε αρκετά η μορφή OSMXML, η οποία παρέχει μια XML αναπαράσταση των δεδομένων η οποία περιλαμβάνει μια λίστα με τα βασικά elements node, way και relation.

Το element node αναπαριστά ένα σημείο στο χάρτη ορισμένο από τις συντεταγμένες του καθώς και πληροφορίες σχετικά με το σημείο αυτό. Το element way αποτελείται από ένα σύνολο κόμβων τα οποία σχηματίζουν μια γραμμή. Στο πλαίσιο της δρομολόγησης είναι το πιο σημαντικό element καθώς με την χρήση του αναπαριστούνται οι δρόμοι των οδικών δικτύων ενώ αποθηκεύονται και πληροφορίες σχετικές με αυτούς. Το element relation αποτελείται από ένα σύνολο node και way και η χρήση του είναι για να εκφράζει λογικές και γεωγραφικές συνδέσεις μεταξύ των εμπεριεχομένων elements. Παρακάτω παρατίθεται ένα παράδειγμα ενός αρχείου OSM XML για την περαιτέρω διασαφήνιση.

```
      <?xml version="1.0" encoding="UTF-8"?>
      <osm     version="0.6"     generator="CGImap     0.3.3     (17076     thorn-
02.openstreetmap.org)"      copyright="OpenStreetMap      and      contributors"
attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/">
       <bounds    minlat="50.8561000"    minlon="4.3253000"    maxlat="50.8638000"
maxlon="4.3452000"/>
       <node   id="145316"   visible="true"   version="3"   changeset="11958828"
timestamp="2012-06-20T13:32:34Z"   user="eMerzh"   uid="15399"   lat="50.8637412"
lon="4.3294861"/>
      …....................
      <node  id="2524655885"  visible="true"  version="1"  changeset="18788958"
timestamp="2013-11-08T20:21:13Z"   user="eMerzh"   uid="15399"   lat="50.8571535"
lon="4.3353904">
        <tag k="addr:housenumber" v="34"/>
        <tag k="addr:street" v="Rue Tazieaux - Tazieauxstraat"/>
        <tag k="ref:UrbIS" v="8154458"/>
       </node>
       <node  id="2524655886"  visible="true"  version="1"  changeset="18788958"
timestamp="2013-11-08T20:21:13Z"   user="eMerzh"   uid="15399"   lat="50.8570885"
lon="4.3351327">
        <tag k="addr:housenumber" v="38"/>
      <tag k="addr:street" v="Rue Tazieaux - Tazieauxstraat"/>
        <tag k="ref:UrbIS" v="8156288"/>
       </node>
      …....................................
      <way   id="8517018"   visible="true"   version="10"   changeset="18167832"
timestamp="2013-10-03T19:37:15Z" user="eMerzh" uid="15399">
        <nd ref=”36754862”/>
        <nd ref="2481636750"/>
        <nd ref="36754834"/>
        <tag k="highway" v="residential"/>
        <tag k="name" v="Re de l'Armistice - Wapenstilstandstraat"/>
        <tag k="name:fr" v="Rue de l'Armistice"/>
        <tag k="name:nl" v="Wapenstilstandstraat"/>
       </way>
      <way   id="14608576"   visible="true"   version="7"   changeset="20741429"
timestamp="2014-02-23T21:53:26Z" user="JanFi" uid="672253">
        <nd ref="144006114"/>
        <nd ref="144002984"/>
```

```
        <tag k="highway" v="residential"/>
        <tag k="name" v="Rue de la Prospérité - Voorspoedstraat"/>
        <tag k="name:fr" v="Rue de la Prospérité"/>
        <tag k="name:nl" v="Voorspoedstraat"/>
        <tag k="oneway" v="yes"/>
        <tag k="oneway:bicycle" v="no"/>
       </way>
      …..................................
       <relation id="2788026" visible="true" version="33" changeset="27276390"
timestamp="2014-12-05T22:25:35Z" user="escada" uid="436365">
        <member type="way" ref="226919708" role=""/>
        <member type="way" ref="199776513" role=""/>
      …..................
        <member type="node" ref="502959019" role="platform"/>
        <member type="node" ref="492356703" role="platform"/>
        <member type="node" ref="1322952502" role="platform"/>
        <tag k="bus" v="express"/>
        <tag k="colour" v="#dd0077"/>
        <tag k="from" v="Aalst Station Perron 5"/>
        <tag k="name" v="De Lijn 212 Snelbus Aalst Station -Brussel
Noordstation/Bruxelles Gare du Nord"/>
        <tag k="operator" v="De Lijn"/>
        <tag k="public_transport:version" v="2"/>
        <tag k="ref" v="212"/>
        <tag k="route" v="bus"/>
        <tag k="to" v="Brussel Noord Perron 2"/>
        <tag k="type" v="route"/>
       </relation>
      </osm>
```

Ο λόγος που έγινε εμπεριστατωμένη μελέτη και ανάλυση της συγκεκριμένης μορφής ήταν η προσπάθεια κατανόησης της πληροφορίας που παρέχεται μέσω του OpenStreetMap, καθώς και του τρόπου αποθήκευσης της. Στο πρακτικό κομμάτι και στο πλαίσιο χρησιμοποίησης και φόρτωσης των δεδομένων χρησιμοποιήθηκε η μορφή PBFFORMAT η οποία αναπαριστά τα δεδομένα σε μια πιο συμπιεσμένη μορφή.

Όσον αφορά την διαδικασία εύρεσης των δεδομένων OpenstreetMap, αυτά μπορούν να βρεθούν κατεβάζοντας είτε ολόκληρο τον χάρτη του πλανήτη, είτε κάποιο υποσύνολό του π. χ μια χώρα ή μια ήπειρο, είτε την οριοθέτηση της ενδιαφέρουσας περιοχής π.χ ένα κομμάτι μιας πόλης.

Βάση δεδομένων PostgreSQL και Εργαλείο Osm2pgsql

Το osm2pgsql είναι ένα πρόγραμμα, που χρησιμοποιείται μέσω της γραμμής εντολών, το οποίο διευκολύνει την διαδικασία ανάγνωσης, μετατροπής και γενικά της διαχείρισης δεδομένων που παρέχονται από το OpenStreetMap. Πιο συγκεκριμένα, ο ρόλος του είναι η μετατροπή και η αποθήκευση των παρεχόμενων δεδομένων σε βάση δεδομένων PostgresSQL, η οποία έχει την επέκταση postGIS[22] ενεργοποιημένη. Η είσοδος του προγράμματος είναι αρχεία OSM XML ή PBF FORMAT και η έξοδος του είναι η τοποθέτηση των δεδομένων των εισαγόμενων αρχείων στην βάση δεδομένων. Με λίγα λόγια διαβάζει τα OpenStreetMap δεδομένα και τα αποθηκεύει στην προαναφερθείσα βάση δεδομένων η οποία είναι κατάλληλη για την αποθήκευση γεωγραφικών δεδομένων. Ο σκοπός της χρήσης του συγκεκριμένου εργαλείου είναι η διευκόλυνση στην διαδικασία της προ-επεξεργασίας των δεδομένων. Στην ουσία η έξοδος του osm2pgsql, δηλαδή η καταχώρηση των δεδομένων στην βάση, παρέχει μια πιο δομημένη και ευκολότερη ως προς την διαχείριση και χρησιμοποίηση, αναπαράσταση των δεδομένων από αυτήν που μπορεί να επιτευχθεί από τα αρχεία που προέρχονται από το OpenStreetMap.

Η εντολή που χρησιμοποιήθηκε για την εκτέλεση του προγράμματος είναι η παρακάτω:

osm2pgsql -l -s -d postgres -d nameofdatabase --hstore /file/path/toosm/fileorpbf/name.osm

Η παράμετρος –s αφορά την εκτέλεση του προγράμματος σε slimmode, το οποίο έχει ως αποτέλεσμα την επιβράδυνση στην εκτέλεση του προγράμματος αλλά ταυτόχρονα την αποθήκευση ενδιάμεσων πινάκων στην βάση δεδομένων.

Η παράμετρος –hstore αφορά την βάση δεδομένων postgresql και πιο συγκεκριμένα την αποθήκευση των δεδομένων σε μορφή κλειδί-τιμή(key-value).

Η παράμετρος –l αφορά την μορφή που θα αποθηκευτούν οι συντεταγμένες στην βάση δεδομένων.

Η παράμετρος –d υποδεικνύει το όνομα της PostgreSQL βάσης δεδομένων που θα γίνει η σύνδεση και η αποθήκευση των δεδομένων.

Το αποτέλεσμα του osm2pgsql προγράμματος είναι η καταχώρηση των δεδομένων OpenStreetMap στην βάση δεδομένων PostgreSQL. Η μορφή και το σχήμα της PostreSQL σύμφωνα με το πώς έχει παραχθεί από το osm2pgsql περιλαμβάνει τέσσερις κύριους πίνακες, καθώς και τρεις βοηθητικούς και ενδιάμεσους οι οποίοι επίσης διαδραματίζουν σημαντικό ρόλο στην διαδικασία της προ-επεξεργασίας.

Το σχήμα της βάσης δεδομένων είναι το εξής:

Κύριοι πίνακες

- **planet_osm_line:** Περιλαμβάνει όλες τις γραμμές (δρόμους) που έχουν εισαχθεί. Οι γραμμές αποτελούνται από συνεχόμενα σημεία ενωμένα μεταξύ τους.
- **planet_osm_point:** Περιλαμβάνει όλα τα σημεία που έχουν εισαχθεί, τα οποία περιέχουν επιπλέον πληροφορία πέραν των συντεταγμένων τους,.
- **planet_osm_polygon:** Περιλαμβάνει όλα τα πολύγωνα που έχουν εισαχθεί. Στα πλαίσια της προ-επεξεργασίας και γενικά της συγκεκριμένης εφαρμογής δεν γίνεται καθόλου χρήση του, καθώς δεν περιέχει χρήσιμη πληροφορία.
- **planet_osm_roads: :** Περιλαμβάνει ένα υποσύνολο του πίνακα planet_osm_line και περιέχει πληροφορίες για απεικόνιση δρόμων που βρίσκονται σε πιο κοντινά zoom. Επίσης δεν γίνεται κάποια χρήση του.

Ενδιάμεσοι πίνακες

- **planet_osm_nodes:** Περιλαμβάνει όλους τους κόμβους που έχουν εισαχθεί, ανεξάρτητα αν περιλαμβάνουν επιπλέον πληροφορία, ή απλά μόνο τις συντεταγμένες τους.
- **planet_osm_rels**: Περιλαμβάνει όλες τις συσχετίσεις (relations) που έχουν εισαχθεί.
- **planet_osm_ways:** Περιλαμβάνει όλες τις γραμμές (δρόμους) που έχουν εισαχθεί.

Από τους παραπάνω πίνακες οι πιο σημαντικοί και αυτοί που περιέχουν χρήσιμες πληροφορίες για την αναπαράσταση του οδικού δικτύου και κατ' επέκταση την απεικόνιση του στον χάρτη είναι οι πίνακας planet_osm_line και planet_osm_ways, διότι με τις κατάλληλες επερωτήσεις μπορούν να ανακτηθούν μόνο οι πλειάδες που αναφέρονται σε δρόμους προσπελάσιμους με αυτοκίνητο, καθώς επίσης και ο πίνακας planet_osm_nodes, διότι περιλαμβάνει πληροφορίες για τα σημεία που εμπεριέχονται σε έναν δρόμο και που τον σχηματίζουν.

Βάση Δεδομένων Για Γραφήματα Neo4j

Το σημαντικότερο βήμα για την επίτευξη του τελικού σκοπού, δηλαδή της πραγματοποίησης δρομολόγησης, είναι η δημιουργία ενός γραφήματος το οποίο θα αναπαριστά την μορφή και την σύσταση των δρόμων καθώς και θα εμπεριέχει πληροφορίες για αυτούς. Ο λόγος που επιλέχθηκε γράφημα για την αναπαράσταση των δρόμων είναι διότι αποτελεί μονόδρομο και γενικά καθιστά ευκολότερη την διαδικασία της δρομολόγησης. Για την δημιουργία και την αποθήκευση του γραφήματος χρησιμοποιήθηκε η βάση δεδομένων για γραφήματα Neo4j[20], η οποία ύστερα από αρκετή έρευνα θεωρήθηκε ότι είναι η καλύτερη για την συγκεκριμένη περίπτωση καθώς προσφέρει μια έτοιμη δομή για την αποθήκευση του γραφήματος, εμπεριέχονται έτοιμες βιβλιοθήκες με αλγόριθμους εύρεσης διαδρομής και συνάμα, λόγω της ύπαρξης μιας επέκτασης για την αποθήκευση γεωγραφικών δεδομένων.

Βέβαια, πριν την δημιουργία του γραφήματος και την αποθήκευση του στην Neo4j, πρέπει να γίνει η επιλογή των δεδομένων που θα χρησιμοποιηθούν και επιπλέον να γίνει μια επεξεργασία για την εξαγωγή επιπλέον πληροφορίας, καθώς και την συμπλήρωση δεδομένων που χρειάζονται αλλά δεν παρέχονται από τα υπάρχοντα δεδομένα. Για την σύνδεση με την βάση δεδομένων PostgreSQL, για την επεξεργασία των δεδομένων, καθώς και την δημιουργία του γραφήματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού JAVA.

Επιλογή Δεδομένων

Αρχικά για την ανάγνωση των δεδομένων, πρέπει να πραγματοποιηθεί η σύνδεση της βάσης δεδομένων PostgreSQL με την JAVA. Για τον λόγο αυτό συγγράφηκε και χρησιμοποιήθηκε μια κλάση, η οποία αναλαμβάνει εξ' ολοκλήρου την σύνδεση με την PostgreSQL, καθώς και την ανάγνωση των δεδομένων από αυτήν. Ο τρόπος υλοποίησης της συγκεκριμένης κλάσης ακολουθεί τους κανόνες του design pattern singleton, έτσι ώστε να υπάρχει μόνο ένα αντικείμενο το οποίο θα είναι υπεύθυνο για την σύνδεση και την επικοινωνία με την βάση, με αποτέλεσμα την αποφυγή υπερφόρτωσης της βάσης δεδομένων.

Το επόμενο στάδιο, αφορά την διαδικασία διαλογής των δεδομένων και την ενασχόληση με αυτά τα οποία θα χρησιμοποιηθούν στην συνέχεια. Ο τελικός σκοπός είναι η παραμονή μόνο των δεδομένων που έχουν χρήσιμη πληροφορία και θα βοηθήσουν στην σύσταση του γραφήματος.

Αρχικά, πρέπει να σημειωθεί ότι τα δεδομένα αυτά, στο συγκεκριμένο στάδιο, βρίσκονται ακόμα ακατέργαστα στην βάση δεδομένων PostgreSQL. Όπως έχει προαναφερθεί στο κεφάλαιο που αφορούσε το σχήμα της PostgreSQL, η πληροφορία που αφορά τους δρόμους

βρίσκεται στον πίνακα planet_osm_line. Έτσι για την απόσπαση της συγκεκριμένης πληροφορίας, ήταν απαραίτητη η ανάγνωση δεδομένων από τον πίνακα αυτόν. Στην παρακάτω εικόνα (Εικόνα 8) φαίνεται το schema του πίνακα planet_osm_line όπως ορίζεται από το wiki του osm2pgsql.

Column	Type	Description
osm_id		ID in OSM
tag	*as specified in style file*	One column for each tag specified in the style file along with the *line* flag, contains the tag value
z_order		Z order (if specified in style file), calculated automatically
way_area		Area (if specified in the style file), calculated automatically
way	LINESTRING	Way geometry (coordinates of all points)

Εικόνα 8: Schema πίνακα planet_osm_line

Όπως αναφέρεται και στην εικόνα, η στήλη tag αφορά όλα τα πιθανά πεδία/χαρακτηριστικά που θα μπορούσε να έχει κάποιος δρόμος. Για την κατανόηση της δομής της βάσης του πίνακα planet_osm_line θα ήταν χρήσιμο η παρουσίαση μιας εικόνας με όλη τη δομή του πίνακα. Όμως, το πλήθος των πεδίων του συγκεκριμένου πίνακα το καθιστά αδύνατο. Συγκεκριμένα, η εκτέλεση του παρακάτω ερωτήματος το οποίο υπολογίζει το πλήθος των πεδίων του συγκεκριμένου πίνακα, επιστρέφει τον αριθμό 70.

```
SELECT COUNT(COLUMN_NAME)

FROM INFORMATION_SCHEMA.COLUMNS

WHERE

TABLE_CATALOG = '****'

AND TABLE_SCHEMA = '****'

AND TABLE_NAME = 'planet_osm_line'
```

Αξίζει να παρατηρηθεί, ότι ο πίνακας planet_osm_line δεν αφορά μόνο δρόμους αλλά περιέχει επίσης δεδομένα άσχετα με τον δικό μας σκοπό. Για τον λόγο αυτό, κρίθηκε απαραίτητη η απαλοιφή των συγκεκριμένων δεδομένων κατά την διάρκεια φόρτωσης τους. Ο απώτερος σκοπός είναι η επιλογή των πλειάδων οι οποίες είναι σχετικές μόνο με δρόμους. Η συγκεκριμένη διαδικασία έγινε σύμφωνα με το πρότυπο που παρέχεται από το wiki του OpenStreetMap [12]. Επίσης, αφαιρέθηκαν και επιπλέον πλειάδες, όπως για παράδειγμα πλειάδες που αφορούσαν πεζόδρομους ή μονοπάτια, με αποτέλεσμα η εναπομένουσα πληροφορία να αφορά μόνο δρόμους οι οποίοι είναι προσπελάσιμοι από οχήματα, διότι αυτός είναι ο σκοπός της συγκεκριμένης εφαρμογής.

Έχοντας καταφέρει την απομόνωση μόνο των πλειάδων με τους επιθυμητούς δρόμους, σειρά έχει η επιλογή μόνο των επιθυμητών γνωρισμάτων από τις εναπομένουσες πλειάδες του πίνακα. Στον πίνακα planet_osm_line υπάρχει πληθώρα γνωρισμάτων σχετικά με τους δρόμους. Παρ' όλα αυτά, επιλέχθηκαν τα γνωρίσματα τα οποία αξιολογήθηκαν ως πιο χρήσιμα και σχετικά

για τον σχηματισμό του γραφήματος και την διαδικασία της δρομολόγησης. Τα γνωρίσματα που επιλέχθηκαν από τον πίνακα planet_osm_line είναι τα εξής:

1. osm_id: το αναγνωριστικό του δρόμου
2. name: το όνομα του δρόμου
3. highway: η κατηγορία του δρόμου (εθνική οδός, επαρχιακός κτλ)
4. oneway: η ένδειξη για την κατεύθυνση του δρόμου
5. surface: η ποιότητα δρόμου (άσφαλτος, χωματόδρομος κτλ)
6. tags: περιέχει περαιτέρω πληροφορίες σε μορφή key-value σχετικά με τον δρόμο. Από το συγκεκριμένο γνώρισμα αποσπάται η μέγιστη επιτρεπόμενη ταχύτητα (maxspeed).

Παρακάτω ακολουθεί το ερώτημα προς τον πίνακα planet_osm_line για την επιλογή μόνο των ζητούμενων δεδομένων και στην συνέχεια μια εικόνα (Εικόνα 9) που παρουσιάζει ένα τμήμα από τα δεδομένα που ανακτώνται από το ερώτημα αυτό, για λόγους αποσαφήνισης. Όπως φαίνεται και στο SQL ερώτημα στο τμήμα της προβολής του ερωτήματος χρησιμοποιούνται μόνο τα προαναφερθέντα γνωρίσματα ενώ στο τμήμα της επιλογής, απορρίπτονται οι πλειάδες που δεν αφορούν δρόμους προσπελάσιμους από όχημα.

```
SELECT osm_id,name,highway,tags,oneway,surface
FROM planet_osm_line
WHERE (highway<>''
AND highway<>'footway'
AND highway<>'pedestrian'
AND highway<>'track'
AND highway<>'bus_guideway'
AND highway<>'raceway'
AND highway<>'footway'
AND highway<>'cycleway'
AND highway<>'bridleway'
AND highway<>'steps'
AND highway<>'path'
AND highway<>'service'
AND highway<>'platform') limit "+ BUCKET_SIZE+" offset "+offset+" ;
```

	osm_id bigint	name text	highway text	tags hstore	oneway text	surface text
1	50054588	Berliner Straße	primary	"lanes"=>"1", "maxspeed"=>"50	yes	asphalt
2	145554440	Menzelstraße	residential	"maxspeed"=>"50", "postal cod		
3	50054449	Berliner Straße	primary	"lanes"=>"1", "cycleway"=>"tr	yes	asphalt
4	43029489	Berliner Straße	primary	"lanes"=>"1", "maxspeed"=>"50	yes	asphalt
5	100833935	Berliner Straße	primary	"lanes"=>"1", "cycleway"=>"la	yes	asphalt
6	96874145	Berliner Straße	primary	"lanes"=>"1", "cycleway"=>"tr	yes	asphalt
7	93466596	Berliner Straße	primary	"lanes"=>"2", "cycleway"=>"la		asphalt
8	220634239	Schwanenallee	living street	"postal code"=>"14467"		
9	50054448	Berliner Straße	primary	"lanes"=>"2", "cycleway"=>"la		asphalt
10	93466597	Krampnitzer Straße	residential	"maxspeed"=>"30", "smoothness		asphalt
11	4425411	Glienicker Brücke	primary	"image"=>"http://commons.wiki		asphalt
12	23045211	Königstraße	primary	"lanes"=>"2", "cycleway"=>"la		
13	23045210	Königstraße	primary	"lanes"=>"2", "cycleway"=>"la		
14	9887761	Allee nach Glienicke	tertiary	"maxspeed"=>"50", "postal cod		asphalt
15	4906363	Parzivalstraße	residential	"postal code"=>"14476"		paved
16	24185057	Karl-Liebknecht-Straße	tertiary	"psv"=>"yes", "maxspeed"=>"30		asphalt
17	97329978	Fährstraße	residential	"smoothness"=>"intermediate"		asphalt
18	79565566	Krampnitzer Straße	residential	"maxspeed"=>"30", "smoothness		asphalt
19	155033905	Königstraße	primary	"lanes"=>"2", "cycleway"=>"la		asphalt
20	33904081	Weinmeisterweg	residential			
21	79565574	Fährstraße	residential	"smoothness"=>"intermediate"		asphalt
22	34068538	Am Waldfrieden	residential	"postal code"=>"14476"		unpaved
23	220634105	Park Babelsberg	residential	"maxspeed"=>"30", "postal cod		asphalt
24	4898647	Bergstraße	residential	"maxspeed"=>"30", "smoothness		paving stones
25	43159028	Allee nach Glienicke	tertiary	"maxspeed"=>"50", "postal cod		asphalt
26	10370005	Hoher Weg	residential	"maxspeed"=>"30", "postal cod		
27	10422945	Wilhelm-Leuschner-Straße	residential	"maxspeed"=>"30", "postal cod		cobblestone

Εικόνα 9: Τμήμα Αποτελεσμάτων από την εκτέλεση του παραπάνω SQL ερωτήματος

Τα παραπάνω δεδομένα περιέχουν ορισμένες πληροφορίες για τους δρόμους, παρ' όλα αυτά όμως, δεν είναι αρκετά για την πλήρη αναπαράστασή τους. Αυτό που λείπει είναι τα δεδομένα για την γεωγραφική θέση των δρόμων. Για την ανάκτηση των δεδομένων αυτών, γίνεται χρήση δύο βοηθητικών πινάκων από την βάση δεδομένων PostgreSQL, τον πίνακα planet_osm_ways και τον πίνακα planet_osm_nodes. Όπως είχε αναφερθεί και στο κεφάλαιο περιγραφής του σχήματος της PostgreSQL, ένα line, στην συγκεκριμένη περίπτωση ένας δρόμος, αποτελείται από ένα σύνολο συνεχόμενων κόμβων, οι οποίοι ενώνονται μεταξύ τους. Συνεπώς, είναι αναγκαία η ανάκτηση των κόμβων για κάθε δρόμο ξεχωριστά. Η πληροφορία αυτή βρίσκεται στον πίνακα planet_osm_ways. Έτσι, με την χρήση του αναγνωριστικού του κάθε δρόμου, ανακτήθηκαν τα αναγνωριστικά των κόμβων που τον αποτελούν.

Παρακάτω παρατίθεται μια εικόνα (Εικόνα 10) που παρουσιάζει την δομή του πίνακα planet_osm_ways για περαιτέρω διασαφήνιση.

	id bigint	nodes bigint[]	tags text[]	pending boolean
1	4045150	{1234120411,262876417,262877047,	{postal code,15732,maxspeed,30,surface,asphalt,highway,residential,name,Waldstraße}	f
2	4045194	{21432547,354484060}	{postal code,10318,description,"Ursula Goetze (1907-1943), member of the German Resistence, sentec	f
3	4045220	{28394961,553334163,3423764025,5	{sidewalk,both,maxspeed,30,surface,cobblestone,highway,residential,lanes,1,name,"Hönower Straße"}	f
4	4045223	{29808846,21432567,21432565,1550	{postal code,10318,sidewalk,both,maxspeed,30,highway,residential,oneway,yes,lanes,1,name,"Gundelfi	f
5	4045243	{1822620447,2845478638,284547863	{postal code,10247,sidewalk,right,maxspeed,50,cycleway,track,surface,asphalt,highway,primary,oneway	f
6	4045247	{21432333,1383533497,1383533517,	{smoothness,bad,surface,cobblestone,highway,secondary,name,Seestraße,ref,"L 401",lit,yes}	f
7	4045248	{56192557,296233577,21432342,308	{smoothness,bad,maxspeed,30,surface,cobblestone,highway,secondary,name,Goethestraße,ref,"L 401",li	f
8	4045655	{21441709,21441710,2167575487,21	{postal code,13158,highway,residential,name,"Am Wiesengrund"}	f
9	4045656	{30432649,30432575,21441709,2236	{postal code,13158,maxspeed,50,cycleway,track,surface,asphalt,highway,secondary,access,yes,lanes,2	f
10	4054007	{21432147,262868706,32495419,324	{smoothness,bad,surface,cobblestone,highway,residential,name,"Grünauer Straße"}	f
11	4054010	{21432148,248687191,1335417715,2	{postal code,15732,surface,cobblestone,highway,residential,name,"Gosener Straße"}	f
12	4054013	{28795932,385448884,534019714,38	{parking:lane:both,none,postal code,12439,maxspeed,50,cycleway,track,surface,asphalt,highway,prima	f
13	4067882	{27560124,282898608,282898609,16	{postal code,14052,maxspeed,50,highway,secondary,oneway,yes,name,"Olympische Straße",lit,yes}	f
14	4067883	{29030447,29030449,29030445,1221	{postal code,14052,maxspeed,50,cycleway,track,highway,secondary,name,Jafféstraße,lit,yes}	f
15	4067899	{21508792,26985242,21508793,2979	{note:name,"Der reg name, sowie ggf. der loc name sind dem zugehörigen Wikipediaartikel entnommen.	f
16	4067902	{21441726,29686212}	{postal code,13405,maxspeed,50,highway,primary,oneway,yes,lanes,2,name,Scharnweberstraße}	f
17	4067906	{32135264,695595039,2799872752,2	{note:name,"Der reg name, sowie ggf. der loc name sind dem zugehörigen Wikipediaartikel entnommen.	f
18	4067919	{28933012,476614441,3259495666}	{postal code,12437,sidewalk,right,maxspeed,50,cycleway,track,surface,asphalt,highway,primary,oneway	f
19	4067922	{530309,530283}	{postal code,13407,maxspeed,50,highway,residential,name,Alt-Reinickendorf}	f
20	4067932	{160503256,1472911019,3463838884	{postal code,10178,wikipedia,de:Liebknechtbrücke,sidewalk,right,maxspeed,50,cycleway,track,highway	f
21	4067937	{18244407,18244411}	{postal code,14163,highway,living street,name,Barkenhof}	f
22	4067941	{27177336,27177335,137549028,271	{postal code,13403,maxspeed,30,surface,asphalt,highway,residential,bicycle,yes,name,Kienhorststraß	f
23	4067944	{16541488,275725722,501547006,36	{postal code,14055,old name,Jafféstraße,highway,secondary,lanes,2,name,"Alte Jafféstraße",lit,yes}	f

Εικόνα 10: Δομή πίνακα planet_osm_ways

Όμως, αυτό που χρειάζεται είναι οι συντεταγμένες του κάθε κόμβου. Για τον λόγο αυτό, με την χρήση του αναγνωριστικού του κάθε κόμβου ανακτώνται οι συντεταγμένες του, από τον πίνακα planet_osm_nodes. Παρακάτω παρατίθεται εικόνα (Εικόνα 11) με την δομή του πίνακα planet_osm_nodes.

	id bigint	lat integer	lon integer	tags text[]
14	172587	525736414	133521444	{highway,traffic signals}
15	172589	525732190	133589950	{highway,traffic signals}
16	172590	525729714	133594305	
17	172594	525703422	133608532	{highway,traffic signals}
18	458320	525027176	132812225	
19	458321	525028858	132814361	
20	458322	525037711	132824173	
21	458323	525042930	132827047	
22	458324	525051202	132828139	
23	458325	525054972	132828394	
24	458326	525062206	132830150	
25	458327	525064340	132831319	
26	458328	525067215	132833130	
27	458329	525086169	132846003	
28	458330	525112015	132861468	{ref,7,name,Kaiserdamm,highway,
29	458331	525121595	132863692	
30	458332	525140104	132865211	
31	458333	525144243	132864310	

Εικόνα 11: Δομή Πίνακα planet_osm_nodes

Για περαιτέρω διευκρίνιση και αποφυγή λάθος συμπερασμάτων, μέχρι στιγμής έχει επιτευχθεί η συγκράτηση δεδομένων μόνο για δρόμους που είναι προσπελάσιμοι από οχήματα. Τα δεδομένα αυτά αφορούν για κάθε δρόμο, αρχικά μια αλληλουχία συντεταγμένων οι οποίες σχηματίζουν την γεωγραφική υπόσταση του δρόμου, και επιπλέον ορισμένα χαρακτηριστικά του δρόμου τα οποία είναι, ένα μοναδικό αναγνωριστικό, το όνομα, η κατηγορία, η κατεύθυνση, η ποιότητα και η μέγιστη επιτρεπόμενη ταχύτητα.

Σε αυτό το σημείο αξίζει να σημειωθεί ότι οι πλειάδες του πίνακα planet_osm_line διαβάζονται και επεξεργάζονται σε ομάδες των 100 (buckets). Η διαδικασία αυτή γίνεται για την επίτευξη καλύτερης διαχείρισης της μνήμης και καλύτερης απόδοσης, ενώ ταυτόχρονα για την αποφυγή σπατάλης όλης της διαθέσιμης μνήμης, δεδομένου ότι τα δεδομένα είναι πολύ μεγάλα σε μέγεθος. Σε επίπεδο υλοποίησης, στο ερώτημα για την ανάκτηση των δεδομένων προστέθηκε στο τέλος το παρακάτω κομμάτι κώδικα SQL το οποίο ευθύνεται για την ανάγνωση των πλειάδων ανά τιμή BUCKET_SIZE, το οποίο έχει τιμή 100.

```
limit "+Initialization.BUCKET_SIZE+" offset "+offset+"
```

Προεπεξεργασία Δεδομένων

Έχοντας αποσπάσει τα απαραίτητα δεδομένα, σειρά έχει η επεξεργασία τους, με σκοπό να έρθουν σε μορφή όπου θα μπορούν να σχηματίσουν το γράφημα. Η διαδικασία αυτή περιλαμβάνει την διάσπαση των δρόμων σε μικρότερα τμήματα για την καλύτερη διαχείριση τους,

την ανάλυση των δεδομένων που έχουν αναγνωστεί από την βάση δεδομένων PostgreSQL, την πρόβλεψη τιμών σε περίπτωση απουσίας τους από τα υπάρχοντα δεδομένα καθώς και τον συνδυασμό των δεδομένων για την δημιουργία περαιτέρω πληροφορίας. Επιπλέον, περιλαμβάνει και την δημιουργία μιας κλάσης για την οργανωμένη διατήρηση της συγκεκριμένης πληροφορίας στην μνήμη μέχρι αυτή να αποθηκευτεί με την μορφή γραφήματος.

Το πρώτο στάδιο της επεξεργασίας των δεδομένων, αφορά την ανάλυση των γνωρισμάτων που διαβάζονται και την διαχείριση τους. Παρακάτω παρουσιάζονται λεπτομερειακά τα βήματα επεξεργασίας του κάθε γνωρίσματος ξεχωριστά. Αρχικά, τα γνωρίσματα που απασχολούν λιγότερο την διαδικασία επεξεργασίας των δεδομένων είναι το αναγνωριστικό του δρόμου και το όνομα του δρόμου. Τα δύο αυτά γνωρίσματα απλώς διαβάζονται και αποθηκεύονται διότι δεν κρίθηκε απαραίτητη η τροποποίηση τους.

Στην συνέχεια, σειρά έχει το γνώρισμα που αφορά την κατηγορία του δρόμου, δηλαδή αν ο δρόμος που αναλύεται είναι εθνική οδός, ή επαρχιακός, ή δρόμος που βρίσκεται σε κατοικημένη περιοχή. Στο γνώρισμα αυτό, υπάρχουν αρκετές τιμές που να καθορίζουν την κατηγορία/είδος του. Για την καλύτερη διαχείριση και αποθήκευση της συγκεκριμένης πληροφορίας έγινε μια ομαδοποίηση σε ορισμένες πιο γενικές κατηγορίες δρόμων. Ο παρακάτω πίνακας παρουσιάζει όλες τις κατηγορίες για το είδος των δρόμων, καθώς και σε ποια μορφή δρόμων αναφέρεται η κάθε κατηγορία σύμφωνα με το wiki του openstreetmap [13].

Κατηγορία για Κατηγορία/Είδος Δρόμου	Περιγραφή
MOTORWAY	**εθνική οδός**
TRUNK	**δρόμοι που δεν είναι εθνικοί οδοί. Π.χ ταχεία κυκλοφορίας**
PRIMARY	**η επόμενη κατηγορία πιο σημαντικών δρόμων (συχνά συνδέουν μεγάλες πόλεις)**
SECONDARY	**η αμέσως επόμενη κατηγορία πιο σημαντικών δρόμων (συχνά συνδέουν πόλεις)**
TERTIARY	**η αμέσως επόμενη κατηγορία πιο σημαντικών δρόμων (συχνά συνδέουν μικρότερες πόλεις ή χωριά)**
ROAD	**οι δρόμοι που είναι λιγότερο σημαντικοί (συχνά συνδέουν χωριά)**
RESIDENTAL	**οι δρόμοι που χρησιμοποιούνται για την πρόσβαση σε κατοικίες**
MOTORWAY_LINK	**οι συνδετικοί δρόμοι για την είσοδο ή την έξοδο από δρόμους κατηγορίας MOTORWAY**

TRUNK_LINK	**οι συνδετικοί δρόμοι για την είσοδο ή την έξοδο από δρόμους κατηγορίας TRUNK**
PRIMARY_LINK	**οι συνδετικοί δρόμοι για την είσοδο ή την έξοδο από δρόμους κατηγορίας PRIMARY**
SECONDARY_LINK	**οι συνδετικοί δρόμοι για την είσοδο ή την έξοδο από δρόμους κατηγορίας SECONDARY**
TERTIARY_LINK	**οι συνδετικοί δρόμοι για την είσοδο ή την έξοδο από δρόμους κατηγορίας TERTIARY**
OTHER	**οποιαδήποτε άλλη κατηγορία δρόμου**

Το επόμενο γνώρισμα που επεξεργάστηκε αφορούσε την ποιότητα του δρόμου. Δηλαδή το υλικό που είναι φτιαγμένος ο δρόμος. Όπως και στο γνώρισμα της κατηγορίας του δρόμου έγινε μια ομαδοποίηση των τιμών της ποιότητας του δρόμου σε πιο γενικές κατηγορίες. Ο παρακάτω πίνακας παρουσιάζει το είδος την ποιότητας του δρόμου που αναφέρεται η κάθε τιμή του enumeration. Σε αυτό το σημείο αξίζει να σημειωθεί ότι για την κατηγοριοποίηση των τιμών έγινε μελέτη του wiki του OpenStreetMap[14].

Κατηγορία για ποιότητα δρόμου	Περιγραφή
ASPHALT	**αναφέρεται σε δρόμους που αποτελούνται κυρίως από άσφαλτο**
CONCRETE	**αναφέρεται σε δρόμους που αποτελούνται κυρίως από τσιμέντο**
COBBLESTONE	**αναφέρεται σε δρόμους που αποτελούνται από πλάκες/πέτρες (πλακόστρωτοι)**
COBBLESTONE, DIRT_ROAD	**αναφέρεται σε χωματόδρομους**
OTHER	**δρόμοι φτιαγμένοι από οποιοδήποτε άλλο υλικό**

Για περαιτέρω διευκρίνηση, όσον αφορά τα γνωρίσματα κατηγορίας και ποιότητας του δρόμου, η παραπάνω διαδικασία έγινε για την ομαδοποίηση της πληροφορίας σε διακριτές τιμές για την καλύτερη διαχείριση τους. Σε επίπεδο υλοποίησης, και για τις δύο κατηγοριοποιήσεις έγινε

χρήση enumeration. Έτσι, στο σημείο αυτό, κάθε δρόμος χαρακτηρίζεται από το είδος του και την ποιότητα του ανάλογα με την τιμή που έχει το κάθε enumeration.

Το επόμενο γνώρισμα που αναλύθηκε αφορά την μέγιστη ταχύτητα που επιτρέπεται. Η διαχείριση και ανάγνωση της παρεχόμενης μέγιστης ταχύτητας γίνεται σύμφωνα με το πρότυπο που παρέχεται από το wiki του OpenStreetMap[15]. Ο σκοπός της επεξεργασίας του γνωρίσματος αυτού είναι η μετατροπή του σε μονάδα μέτρησης χιλιόμετρα ανά ώρα. Στην συνέχεια, γίνεται η ανάλυση του γνωρίσματος που αφορά την κατεύθυνση του δρόμου, με σκοπό την εξαγωγή πληροφορίας για το αν ο δρόμος είναι διπλής κατεύθυνσης ή μονής. Στην περίπτωση της μονής κατεύθυνσης, αποσπάται και πληροφορία σχετικά με το ποια κατεύθυνση είναι η επιτρεπόμενη.

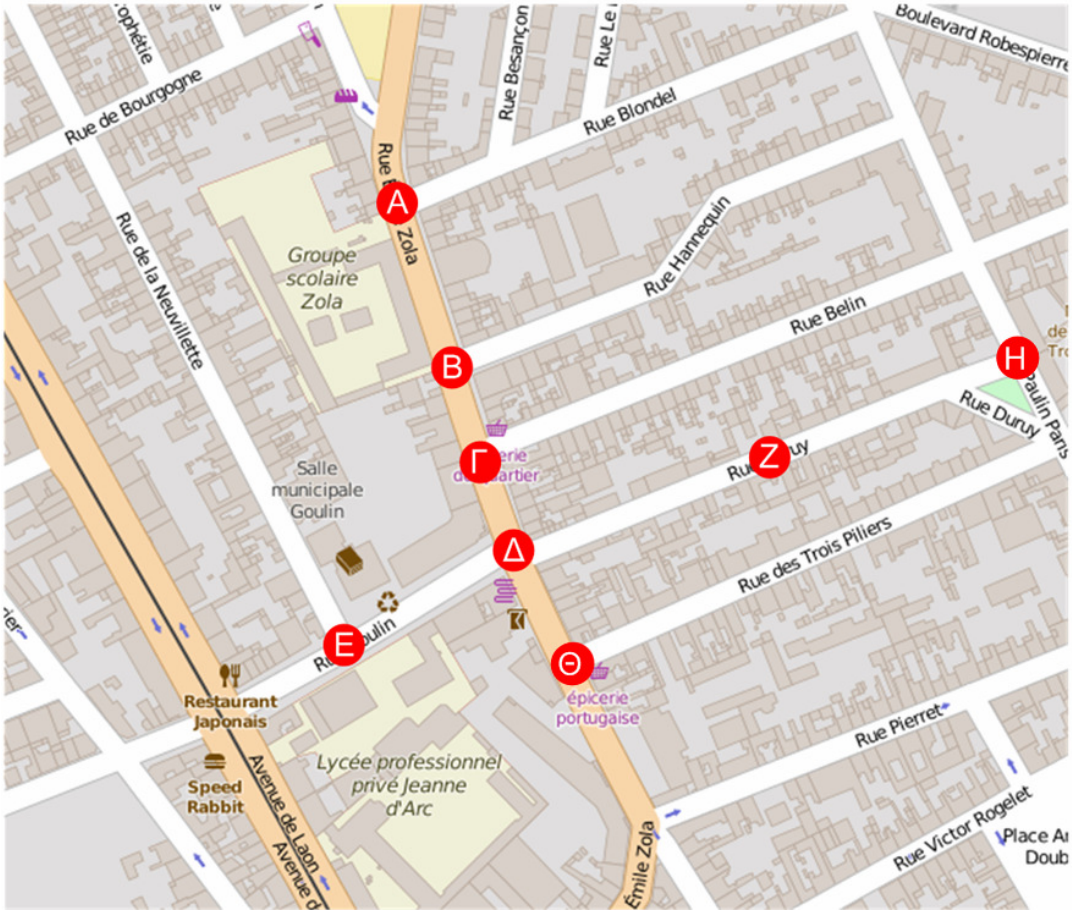
Στο σημείο αυτό, αξίζει να παρατηρηθεί ότι κατά την διάρκεια ανάγνωσης αλλά και επεξεργασίας των δεδομένων πολλές φορές παρουσιάζεται το ζήτημα έλλειψης των αναγκαίων δεδομένων. Δηλαδή να μην υπάρχουν όλα τα απαραίτητα δεδομένα για όλους τους δρόμους από το OpenStreetMap. Για τον λόγο αυτό είναι αδήριτη αναγκαιότητα η συμπλήρωση των τιμών που απουσιάζουν με πιθανές τιμές συμφωνά με ορισμένα κριτήρια που θα αναλυθούν παρακάτω.

Τα γνωρίσματα που παρουσιάζουν το φαινόμενο της έλλειψης των τιμών τους είναι το γνώρισμα της ποιότητας του δρόμου, καθώς και αυτό της μέγιστης επιτρεπόμενης ταχύτητας. Όσον αφορά την ποιότητα του δρόμου, για την πρόβλεψη και την συμπλήρωση της πιθανής τιμής χρησιμοποιείται το γνώρισμα της κατηγορίας/είδους του δρόμου, διότι προσφέρει μια έγκυρη σε γενικές γραμμές εικόνα για την ποιότητα και σύσταση του δρόμου. Πιο αναλυτικά, οι δρόμοι που είναι της κατηγορίας MOTORWAY, TRUNK, PRIMARY, MOTORWAY_LINK, TRUNK_LINK και PRIMARY_LINK ως μεγάλοι και κεντρικοί δρόμοι παίρνουν την τιμή ASPHALT όσον αφορά την ποιότητα τους, δηλαδή ότι αποτελούνται από άσφαλτο. Με αντίστοιχη λογική, δρόμοι της κατηγορίας SECONDARY, TERTIARY, ROAD, RESIDENTAL, SECONDARY_LINK και TERTIARY_LINK παίρνουν την τιμή CONCRETE που αναπαριστά δρόμο φτιαγμένο από τσιμέντο.

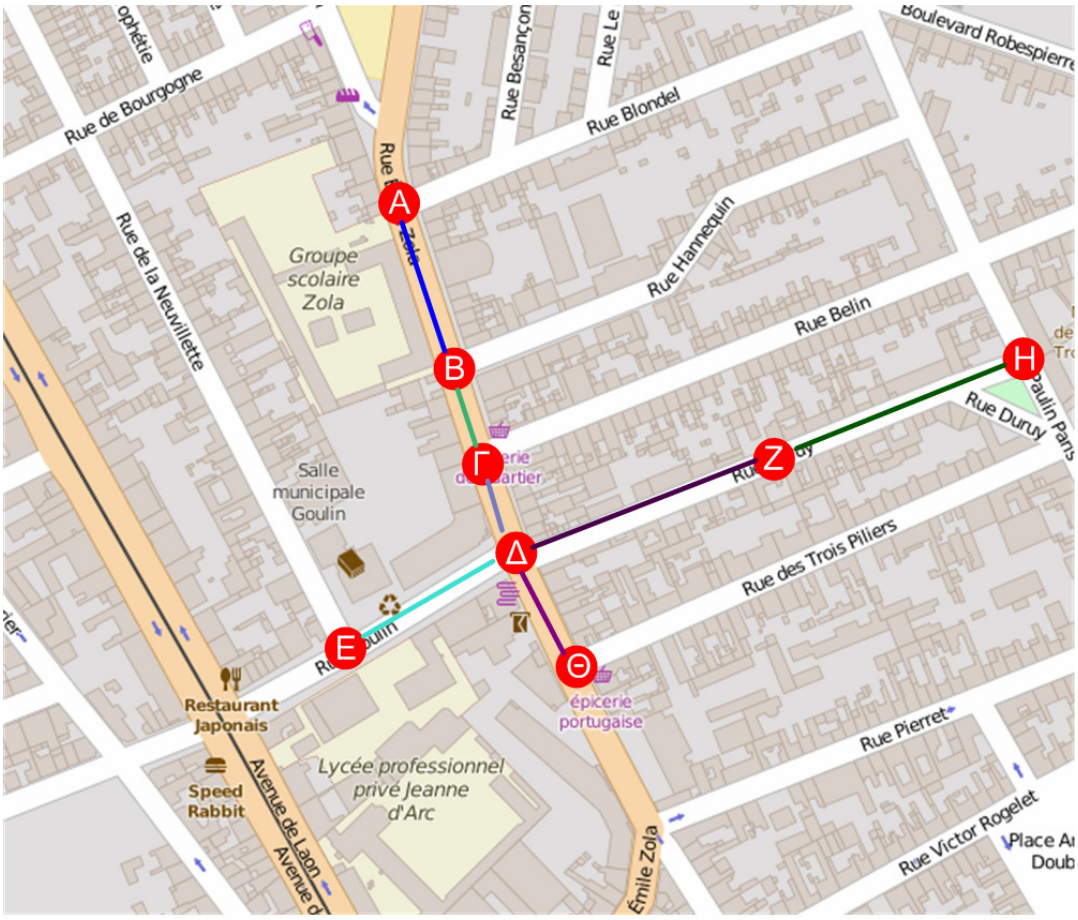
Όσον αφορά την μέγιστη επιτρεπόμενη ταχύτητα, για την πρόβλεψη και την συμπλήρωση τιμών χρησιμοποιούνται τα γνωρίσματα της κατηγορίας του δρόμου και της ποιότητας του δρόμου. Αρχικά, με την χρήση της κατηγορίας του δρόμου, καθώς και με την μελέτη του wiki του OpenStreetMap [16] σχετικά με την επιτρεπομένη ταχύτητα ανά κατηγορία δρόμου ορίζεται μια αρχική μέγιστη επιτρεπόμενη ταχύτητα. Στην συνέχεια, ελέγχεται και η ποιότητα του δρόμου. Σε περίπτωση που ο δρόμος είναι άσφαλτος, η τιμή αυτή παραμένει σταθερή. Διαφορετικά, αν ο δρόμος αποτελείται από τσιμέντο, ή είναι πλακόστρωτος, ή χωματόδρομος η τιμή αυτή μειώνεται κατά δέκα, ή είκοσι, ή τριάντα χιλιόμετρα αντίστοιχα.

Όπως έχει τονιστεί και παραπάνω, τα δεδομένα που αφορούν την γεωγραφική υπόσταση ενός δρόμου είναι ένα σύνολο συνεχόμενων κόμβων (στην ουσία ένα σύνολο συντεταγμένων), των οποίων η ένωση, σχηματίζει μια γραμμή που αναπαριστά έναν δρόμο. Για την καλύτερη διαχείριση ενός δρόμου, κρίθηκε απαραίτητη η διάσπασή του σε μικρότερα τμήματα, όπως αυτά ορίζονται από τους κόμβους που τον αποτελούν. Πιο συγκεκριμένα, το κομμάτι δρόμου που ενώνει δύο συνεχόμενους κόμβους, αποτελεί πλέον έναν αυτόνομο δρόμο. Ένα παράδειγμα για να κάνει το αποτέλεσμα της παραπάνω διαδικασίας πιο σαφές είναι, ότι αν έχουμε έναν δρόμο που σχηματίζεται από τους κόμβους Α,Β,Γ,Δ, αυτός ο δρόμος πλέον μετασχηματίζεται σε 3 αυτόνομους δρόμους ΑΒ, ΒΓ,ΓΔ. Ο σημαντικότερος λόγος της πραγματοποίησης της συγκεκριμένης ενέργειας

είναι διότι βοηθάει σε τεράστιο βαθμό την δημιουργία του γραφήματος καθώς και την ένωση των δρόμων μεταξύ τους στα σημεία που διασταυρώνονται. Αυτό συμβαίνει, επειδή κόμβοι στους δρόμους εμφανίζονται κυρίως στα σημεία που συνδέονται με άλλους δρόμους. Έτσι πρέπει να διασφαλιστεί ότι αυτό το σημείο θα είναι ένας ανεξάρτητος κόμβος που θα εισέρχονται και θα εξέρχονται αρκετές ακμές, προσομοιώνοντας έτσι την δυνατότητα πρόσβασης από έναν δρόμο σε έναν άλλον. Στις 2 παρακάτω εικόνες (Εικόνα 12 και Εικόνα 13), απεικονίζεται γραφικά το αποτέλεσμα της παραπάνω διαδικασίας. Στην εικόνα 12 φαίνονται οι κόμβοι Α, Β, Γ, Δ, Θ που σχηματίζουν έναν δρόμο. Επίσης οι κόμβοι Ε, Δ, Ζ, Η σχηματίζουν έναν δρόμο. Μετά την προαναφερθείσα διαδικασία, τα κομμάτια δρόμων που σχηματίζονται είναι τα ΑΒ, ΒΓ, ΓΔ, ΔΘ, ΕΔ, ΔΖ, ΖΗ. Το αποτέλεσμα φαίνεται στην εικόνα 13.



Εικόνα 12: Ο δρόμος πριν τη διάσπασή του σε μικρότερα κομμάτια



Εικόνα 13: Ο αρχικός δρόμος έχει διασπαστεί σε μικρότερα αυτόνομα κομμάτια δρόμου

Συνολικά, με όλα τα παραπάνω βήματα, έχουμε καταλήξει στο τα δεδομένα μας να είναι ένα σύνολο από κομμάτια δρόμων, τα οποία εμπεριέχουν ορισμένες πληροφορίες. Οι πληροφορίες αυτές είναι:

1. Το αναγνωριστικό του δρόμου που βρίσκεται το κομμάτι του δρόμου
2. Το όνομα του δρόμου
3. Το αναγνωριστικό του ενός κόμβου που αποτελείται το κομμάτι δρόμου (από το αναγνωριστικό βρίσκουμε τις συντεταγμένες του)
4. Το αναγνωριστικό του δεύτερου κόμβου που αποτελείται το κομμάτι δρόμου (από το αναγνωριστικό βρίσκουμε τις συντεταγμένες του)
5. Το αναγνωριστικό του κομματιού του δρόμου
6. Η κατηγορία/είδος του δρόμου ανάλογα με την τιμή που έχει από το enumeration
7. Η μέγιστη επιτρεπόμενη ταχύτητα
8. Η κατεύθυνσή του δρόμου
9. Η ποιότητα του δρόμου ανάλογα με την τιμή που έχει από το enumeration

Για την συγκράτηση και την διαχείριση της παραπάνω πληροφορίας δημιουργήθηκε μια κλάση. Η συγκεκριμένη κλάση παρατίθεται παρακάτω:

```
public class Road {
    private long id;
    private String name;
    private long source_id;
    private long target_id;
    private long highway_id;
    private highway_type_enum highway_type;
    private int maxspeed;
    private boolean oneway;
    private surface_type_enum surface_type;

    public  enum highway_type_enum{
        MOTORWAY,TRUNK,PRIMARY,SECONDARY,TERTIARY,UNCLASSIFIED,ROAD,RESIDENTAL,
        MOTORWAY_LINK,TRUNK_LINK,PRIMARY_LINK,SECONDARY_LINK,TERTIARY_LINK,OTHER
    }
    public enum surface_type_enum{
        ASPHALT,CONCRETE,COBBLESTONE,DIRT_ROAD,OTHER
    }

    public Road(){}
    public Road(long id, String name, long source_id
    ,long target_id,long highway_id,
    highway_type_enum highway_type, int maxspeed,
    boolean oneway, surface_type_enum surface_type) {
        super();
        this.id = id;
        this.name = name;
        this.source_id = source_id;
        this.target_id = target_id;
        this.highway_id = highway_id;
        this.highway_type = highway_type;
        this.maxspeed = maxspeed;
        this.oneway = oneway;
        this.surface_type = surface_type;
    }

    // getters -- setters

}
```

Παρακάτω παρατίθεται το τμήμα κώδικα(μέθοδος) το οποίο είναι υπεύθυνο για την ανάγνωση από την βάση δεδομένων PostgreSQL, την επεξεργασία και την αποθήκευση στην μνήμη των δρόμων με την μορφή λίστας για την καλύτερη διαχείριση τους. Όπως φαίνεται, για κάθε δρόμο ξεχωριστά, διαβάζονται όλα τα γνωρίσματα από την βάση δεδομένων, στην συνέχεια αναλύεται το κάθε ένα ξεχωριστά όπως έχει παρουσιαστεί παραπάνω με την χρήση διαφορετικών μεθόδων για το κάθε ένα και στην συνέχεια αποθηκεύονται όλα μαζί σε ένα αντικείμενο της κλάσης Road που παρουσιάστηκε παραπάνω. Ιδιαίτερο ενδιαφέρον παρουσιάζει το κομμάτι της επανάληψης for το οποίο είναι υπεύθυνο για την διαδικασία διάσπασης του δρόμου σε μικρότερα τμήματα καθώς και την δημιουργία ενός ξεχωριστού αναγνωριστικού σε καθένα από αυτά.

```
private static ArrayList<Road> parseHighway(ResultSet roads_rs){
    ArrayList<Road> roads=new ArrayList<Road>();
    try{
```

```
        long highway_id=roads_rs.getLong("osm_id");
        String name=DataTransform.parseName(roads_rs.getString("name"));
        highway_type_enum
highway_type=DataTransform.parseHighwayType(roads_rs.getString("highway"));
        int oneway=DataTransform.parseDirection(roads_rs.getString("oneway"));
        surface_type_enum
surface_type=DataTransform.parseSurfaceType(roads_rs.getString("surface"),highwa
y_type);
        int
maxspeed=DataTransform.parseMaxSpeed(roads_rs.getString("tags"),highway_type,sur
face_type);
        ArrayList<Long> nodes=getNodes(highway_id);
        boolean isOneway;
        for(int i=0;i<nodes.size()-1;i++){
            long source_id=nodes.get(i);
            long target_id=nodes.get(i+1);
            if(oneway==0){
                isOneway=false;
            }else if(oneway==-1){
                long tmp=source_id;
                source_id=target_id;
                target_id=tmp;
                isOneway=true;
            }else{
                isOneway=true;
            }
            long road_id=0;
            try{
                String _id=Long.toString(highway_id);
                String _index=Integer.toString(i);
                String string_road_id=_id+_index;
                road_id=Long.parseLong(string_road_id);
            }catch(Exception ex){
                ex.printStackTrace();
            }
            Road road=new
Road(road_id,name,source_id,target_id,highway_id,highway_type,maxspeed,isOneway,
surface_type);
            roads.add(road);
        }
    }catch(Exception ex){
        ex.printStackTrace();
    }
    return roads;
}
```

Με την διάσπαση των δρόμων σε μικρότερα τμήματα καθώς και με την απόσπαση και την ανάλυση της ζητούμενης πληροφορίας η διαδικασία της επεξεργασίας δεδομένων έχει σχεδόν ολοκληρωθεί. Αυτό που λείπει είναι η δημιουργία της πληροφορίας, με βάση την οποία θα πραγματοποιηθεί η δρομολόγηση. Όπως είχε αναφερθεί και παραπάνω ο σκοπός της συγκεκριμένης εφαρμογής είναι η εύρεση της συντομότερης από άποψη χρόνου διαδρομής. Για τον λόγο αυτό είναι αναγκαία η γνώση του χρόνου που χρειάζεται για την διάσχιση κάθε τμήματος δρόμου που έχει δημιουργηθεί, καθώς θα αποτελεί το βάρος των ακμών του γραφήματος. Για τον υπολογισμό της πληροφορίας αυτής, είναι απαραίτητη η γνώση της απόστασης σε χιλιόμετρα ανά ώρα, καθώς και η μέγιστη επιτρεπόμενη ταχύτητα.

Η μέγιστη επιτρεπόμενη ταχύτητα είναι ήδη καταχωρημένη, είτε από τα δεδομένα του OpenStreetMap, είτε από τον συμψηφισμό κατηγορίας και ποιότητας δρόμου. Έτσι, αυτό που απομένει να υπολογιστεί, είναι η γεωγραφική απόσταση του κάθε κομματιού δρόμου, δηλαδή το

μήκος του. Για τον υπολογισμό της απόστασης, συμβάλουν τα δύο ζευγάρια συντεταγμένων που ορίζουν το κομμάτι δρόμου. Πιο συγκεκριμένα, αυτό υλοποιείται με την χρήση του τύπου Harvesine ο οποίος με είσοδο δύο ζευγαριών συντεταγμένων υπολογίζει την απόσταση για την μετάβαση από το ένα σημείο στο άλλο. Παρακάτω παρατίθεται το κομμάτι κώδικα(μέθοδος) που συγγράφηκε για την υλοποίηση του τύπου Harvesine.

```
private static final double EARTH_RADIUS=6371*1000;
public static double calculateDinstanceCost(double lon1,double lat1,double
lon2,double lat2){
     double dLat = Math.toRadians(lat2-lat1);
     double dLng = Math.toRadians(lon2-lon1);
     double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
               Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))
*
               Math.sin(dLng/2) * Math.sin(dLng/2);
     double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
     float dist = (float) (EARTH_RADIUS * c);
     return dist;
}
```

Στην συνέχεια, με απόσταση και ταχύτητα γνωστές, υπολογίζεται ο χρόνος που χρειάζεται να διασχιστεί το κομμάτι δρόμου με την χρήση του πρώτου νόμου του Νεύτωνα(t=x/u).

Με την ολοκλήρωση της παραπάνω διαδικασίας ολοκληρώνεται ουσιαστικά και το βήμα της επεξεργασίας δεδομένων. Συνεπώς, το βήμα που ακολουθεί, είναι αυτό της δημιουργίας του γραφήματος.

Δημιουργία Γραφήματος

Η διαδικασία δημιουργίας του γραφήματος, στην ουσία αποτελείται από την μετατροπή και αποθήκευση των δεδομένων σε μορφή γραφήματος. Ένα γράφημα αποτελείται από ένα σύνολο κόμβων και ακμών. Για τον λόγο αυτό και στη συγκεκριμένη περίπτωση, κάθε κομμάτι δρόμου πρέπει να μετατραπεί σε ακμή του γραφήματος και οι κόμβοι που αποτελούν και σχηματίζουν το κομμάτι του δρόμου να μετατραπούν σε κόμβους της συγκεκριμένης ακμής στο γράφημα. Συνεπώς, η διαδικασία που γίνεται για κάθε κομμάτι δρόμου αποτελείται από δύο μέρη. Την εισαγωγή στην βάση δεδομένων Neo4j των δύο κόμβων του δρόμου και στην συνέχεια την σύνδεση τους με μια ακμή, καθώς και την αποθήκευση των απαραίτητων πληροφοριών σε αυτά.

Πιο αναλυτικά, για κάθε κομμάτι δρόμου που εξετάζεται, αρχικά δημιουργείται ένας νέος κόμβος στο γράφημα. Στην συνέχεια, αποθηκεύονται οι απαραίτητες πληροφορίες σχετικά με τον κόμβο αυτόν. Αυτές οι πληροφορίες είναι, το αναγνωριστικό του κόμβου, καθώς και οι συντεταγμένες του. Επιπλέον, το αναγνωριστικό του κάθε κόμβου καθώς και οι πληροφορίες του εισάγονται σε ένα ευρετήριο που παρέχεται από την γεωγραφική επέκταση (plugin) που διαθέτει το Neo4j. Η διαδικασία αυτή, είναι χρήσιμη για την εκτέλεση γεωγραφικών ερωτημάτων στην βάση δεδομένων κατά την διαδικασία της δρομολόγησης στην συνέχεια. Τα βήματα που παρουσιάζονται παραπάνω εκτελούνται επίσης και για τον δεύτερο κόμβο που αφορά το κομμάτι δρόμου που επεξεργάζεται.

Στην συνέχεια, σειρά έχει η δημιουργία και η αποθήκευση της ακμής που ενώνει τους δύο κόμβους στο γράφημα. Αρχικά, ανάλογα με την κατεύθυνση του κομματιού του δρόμου που

έχει καθοριστεί από προηγούμενο στάδιο, επιλέγεται και η κατάλληλη κατεύθυνση της ακμής στο γράφημα. Δηλαδή, αν το κομμάτι δρόμου είναι διπλής κατεύθυνσης, δημιουργείται μια ακμή από τον πρώτο κόμβο προς τον δεύτερον και μια ακμή από τον δεύτερο κόμβο προς τον κόμβο. Διαφορετικά, δημιουργείται μια μόνο ακμή ανάλογα προφανώς με την επιτρεπόμενη κατεύθυνση. Έπειτα, αποθηκεύονται στην ακμή αυτή, όλες οι απαραίτητες πληροφορίες που περιγράφουν ένα κομμάτι δρόμου. Οι πληροφορίες αυτές είναι, το αναγνωριστικό του δρόμου, το αναγνωριστικό του κομματιού του δρόμου, το όνομα του δρόμου, η κατηγορία/είδος του δρόμου, η ποιότητα του δρόμου και η μέγιστη επιτρεπόμενη ταχύτητα. Επιπλέον, αποθηκεύεται ο χρόνος που χρειάζεται για να διασχιστεί το κομμάτι δρόμου, το οποίο αποτελεί και το βάρος της ακμής, με την χρήση του οποίου στην συνέχεια θα εκτελεστεί ο αλγόριθμος εύρεσης την βέλτιστης διαδρομής. Παρακάτω παρατίθεται το κομμάτι κώδικα(μέθοδος) το οποίο είναι υπεύθυνο για την δημιουργία του γραφήματος και πιο συγκεκριμένα για την εισαγωγή των δεδομένων σε αυτό. Αναλυτικότερα, για κάθε κομμάτι δρόμου που βρίσκεται στη λίστα αρχικά αποκτώνται και οι δύο κόμβοι του. Η διαδικασία απόκτησης των κόμβων αναλύεται επιγραμματικά σε παρακάτω στάδιο. Στη συνέχεια, ανάλογα με την κατεύθυνση του δρόμου που εξετάζεται δημιουργείται μια ακμή η οποία είτε θα είναι δύο κατευθύνσεων, είτε μίας. Στην συνέχεια αποθηκεύονται τα γνωρίσματα της ακμής στο γράφημα, από τα οποία το πιο σημαντικό είναι το γνώρισμα static_cost, το οποίο αποτελεί το βάρος της ακμής και εκφράζει τον χρόνο προσπέλασης του κομματιού του δρόμου που αναπαριστά η ακμή.

```
private static void addNodesRels(ArrayList<Road> roads){
  for(Road r:roads){
    Node source=retrieveNode(r.getSource_id());
    Node target=retrieveNode(r.getTarget_id());

    try(Transaction tx=graphDb.beginTx()){
        Relationship rel;
        if(r.isOneway()){
            rel=source.createRelationshipTo(target,ExampleTypes.ONE_WAY);
        }else{
            rel=source.createRelationshipTo(target,ExampleTypes.BIDIRECTIONAL);
        }

        rel.setProperty("road_id", r.getId());
        rel.setProperty("name", r.getName());
        rel.setProperty("highway_id", r.getHighway_id());
        rel.setProperty("highway_type", r.getHighway_type().name());
        rel.setProperty("surface_type", r.getSurface_type().name());
        rel.setProperty("maxspeed", r.getMaxspeed());
        double dinstance= CostCalculation.calculateDinstanceCost(source,
target);
        rel.setProperty("dinstance", dinstance);
        double
static_cost=CostCalculation.CalculateStaticCost(r.getMaxspeed(),dinstance );
        rel.setProperty("static_cost", static_cost);
        tx.success();
        }

    }
 }
```

Παρακάτω παρατίθεται το κομμάτι κώδικα (μέθοδος) που αφορά την απόκτηση του κόμβου. Όπως έχει εξηγηθεί και παραπάνω, δύο κομμάτια δρόμων μπορούν να μοιράζονται έναν

κόμβο. Αυτός είναι άλλωστε και ο τρόπος σύνδεσης τους, δηλαδή η διασταύρωση. Για τον λόγο αυτό και όπως φαίνεται και στην παρακάτω μέθοδο για την απόκτηση του κόμβου αρχικά ελέγχεται αν ο συγκεκριμένος κόμβος με το συγκεκριμένο αναγνωριστικό, έχει εισαχθεί ήδη στο γράφημα. Σε περίπτωση που έχει εισαχθεί ήδη, ανακτάται για την σύνδεση της νέας ακμής σε αυτόν. Διαφορετικά, δημιουργείται ένας νέος κόμβος, αποθηκεύεται στο γράφημα αλλά και ευρετηριοποιείται με βάση τις συντεταγμένες του από το ειδικό plugin του Neo4j έτσι ώστε στην συνέχεια να λαμβάνεται υπόψιν από τα γεωγραφικά ερωτήματα που θα εκτελεστούν από το συγκεκριμένο plugin.

```
private static Node retrieveNode(long node_id){
    if(nodeExists(node_id)){
        try(Transaction tx=graphDb.beginTx()){
        //return the node
        Node existing_node=nodeIndex.get("id",node_id).getSingle();
        tx.success();
        return existing_node;
        }
    }else{
        try(Transaction tx=graphDb.beginTx()){
            //create and return the node
            Coordinate coo=FetchDataFromDB.fetchCoordinates(node_id);
            SpatialDatabaseRecord
record=runningLayer.add(runningLayer.getGeometryFactory().createPoint(coo));
            Node created_node=record.getGeomNode();
            created_node.setProperty("id", new Long(node_id));
            created_node.setProperty("lon",new Double(coo.x));
            created_node.setProperty("lat",new Double(coo.y));
            nodeIndex.add(created_node, "id", new Long(node_id));
            tx.success();
            return created_node;

   }
  }
 }
```

Στο σημείο αυτό, τονίζεται η σημαντικότητα της επέκτασης(plugin) του Neo4j για την διαχείριση γεωγραφικών δεδομένων [17]. Η συγκεκριμένη επέκταση παρέχει μια βιβλιοθήκη η οποία διαθέτει μια πληθώρα λειτουργιών και ερωτημάτων που μπορούν να εκτελεστούν σε γεωγραφικά δεδομένα, κάνοντας έτσι πολύ ευκολότερη και αποδοτικότερη την χρήση τους καθώς οι αναζητήσεις είναι αρκετά γρήγορες εξαιτίας των R-δέντρων που χρησιμοποιούνται για την αποθήκευση και την αναπαράσταση των δεδομένων από το plugin. Ιδιαίτερη σημασία, η συγκεκριμένη επέκταση παρουσιάζει στο στάδιο εκτέλεσης ερωτημάτων στο γράφημα, διαδικασία που εξετάζεται επιγραμματικά παρακάτω.

Συνοψίζοντας, ύστερα από την περαίωση της διαδικασίας που περιγράφεται παραπάνω, έχει σχηματιστεί ένα γράφημα, το οποίο αποτελείται από κόμβους και ακμές. Το σύνολο των κόμβων αποτελείται από το σύνολο των συντεταγμένων που ορίζουν σημεία πάνω στους δρόμους. Το σύνολο των ακμών αποτελείται από τμήματα δρόμων που ενώνουν τους συγκεκριμένους κόμβους. Επιπλέον, κάθε κόμβος έχει ευρετηριοποιηθεί για την εκτέλεση γεωγραφικών ερωτημάτων, ενώ ταυτόχρονα σε κάθε ακμή υπάρχει ένα γνώρισμα το οποίο αναπαριστά το κόστος της ακμής, του οποίου η φυσική του σημασία έχει να κάνει με τον χρόνο που χρειάζεται για να διασχιστεί το συγκεκριμένο κομμάτι δρόμου. Σύμφωνα με τα παραπάνω, στο σημείο αυτό, έχει δημιουργηθεί πλέον, ένα γράφημα το οποίο μπορεί να χρησιμοποιηθεί για την διαδικασία της δρομολόγησης με την χρήση του αλγορίθμου Dijktra.

Εκτέλεση ερωτημάτων-Επίτευξη δρομολόγησης

Το συγκεκριμένο κεφάλαιο αφορά την παρουσίαση των τεχνικών και τεχνολογιών που χρησιμοποιήθηκαν για το γραφικό κομμάτι της εφαρμογής, καθώς και της επικοινωνίας του φυλλομετρητή(browser) με τον εξυπηρετητή. Επίσης, αναλύονται και οι διαδικασίες εύρεσης του κοντινότερου κόμβου στο γράφημα και εκτέλεσης του αλγορίθμου. Πιο συγκεκριμένα ο φυλλομετρητής, στον οποίο εκτελείται η εφαρμογή διαβάζει την είσοδο του χρήστη και την αποστέλλει στον εξυπηρετητή. Ο εξυπηρετητής, βρίσκει τους κοντινότερους κόμβους στο γράφημα ως προς την είσοδο εκκίνησης και τερματισμού του χρήστη και στην συνέχεια εκτελείται η διαδικασία εύρεσης του συντομότερου μονοπατιού ανάμεσα στους δύο αυτούς κόμβους. Έπειτα το μονοπάτι που έχει βρεθεί, επιστρέφεται στον φυλλομετρητή, όπου εμφανίζεται γραφικά στον χρήστη.

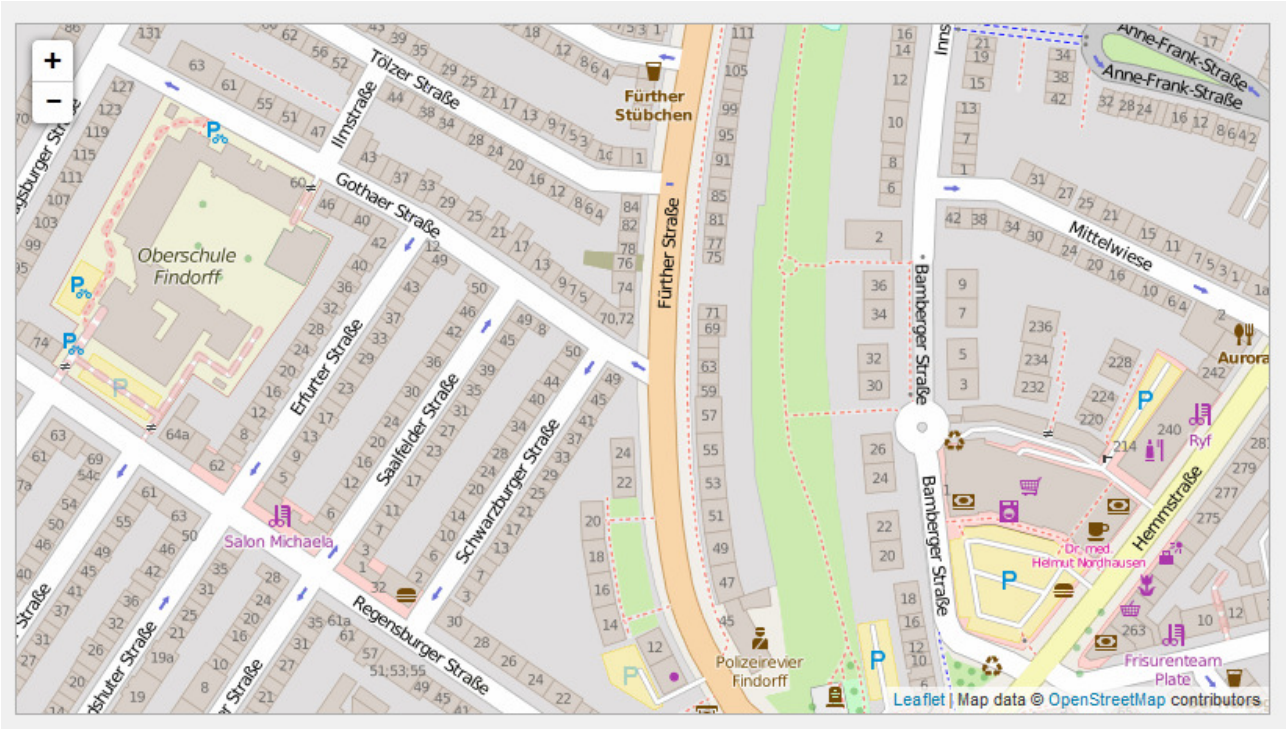
Είσοδος Χρήστη

Η διαδικασία την δρομολόγησης απαιτεί την ύπαρξη αρχικού και τελικού σημείου τα οποία καθορίζονται από την είσοδο του χρήστη. Για την διευκόλυνση του χρήστη, ως προς την παροχή δεδομένων εισόδου, δηλαδή δύο ζευγαριών συντεταγμένων, γίνεται χρήση γραφικού χάρτη. Αναλυτικότερα, ο χρήστης με δύο συνεχόμενα κλικ ορίζει τα σημεία εκκίνησης και τερματισμού. Το πρώτο κλικ αφορά το σημείο εκκίνησης, ενώ το δεύτερο, το σημείο τερματισμού. Στην συνέχεια, από τα επιλεγμένα σημεία, ανακτώνται οι συντεταγμένες τους.

Για την γραφική απεικόνιση του χάρτη καθώς και την διαχείριση του χρησιμοποιήθηκε η βιβλιοθήκη Leaflet.js [18]. Πιο συγκεκριμένα, αρχικά φορτώνεται ο χάρτης της περιοχής που ενδιαφέρεται να εμφανιστεί δηλώνοντας επίσης και το αρχικό zoom στον χάρτη. Επιπρόσθετα, η συγκεκριμένη περιοχή οριοθετείται, δηλαδή ορίζεται το μέγιστο zoom-out και zoom-in που επιτρέπεται στον χάρτη καθώς και το πόσο επιτρέπεται στον χρήστη να απομακρυνθεί από αυτήν. Το παρακάτω κομμάτι κώδικα javascript αναλαμβάνει την παραπάνω διαδικασία.

```
map = new L.Map('map');
// create the tile layer with correct attribution
var osmUrl='http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';
var osmAttrib='Map data © <a href="http://openstreetmap.org">OpenStreetMap</a>
contributors';
var osm = new L.TileLayer(osmUrl, {minZoom: 10, maxZoom: 20, maxNativeZoom: 18,
attribution: osmAttrib});
map.setView(new L.LatLng(53.0966258, 8.8067699),17 );
var northEast = new L.LatLng(53.2192808, 8.9501943);
var southWest = new L.LatLng(53.0274496, 8.4934559);
var bounds = new L.LatLngBounds(southWest, northEast);
map.setMaxBounds(bounds);
map.addLayer(osm);
```

Παρακάτω ακολουθεί εικόνα (Εικόνα 14) με τον χάρτη που εμφανίζεται. Αξίζει να σημειωθεί, ότι ο χάρτης προέρχεται από το OpenStreetMap.



Εικόνα 14:Ο χάρτης της εφαρμογής

Στην συνέχεια, αφού έχει εμφανιστεί ο χάρτης στον χρήστη, ο χρήστης πρέπει να δώσει την είσοδο του. Δηλαδή, πρέπει να ενημερώσει το σύστημα από ποιο σημείο, μέχρι ποιο σημείο θέλει να επιτευχθεί η δρομολόγηση. Η διαδικασία αυτή, γίνεται με δύο συνεχόμενα κλικ από τον χρήστη στον χάρτη. Πιο συγκεκριμένα, το πρώτο κλικ του χρήστη εμφανίζει στον χάρτη ένα πράσινο σημάδι και το δεύτερο κλικ ένα κόκκινο σημάδι που υποδηλώνουν τα σημείο εκκίνησης και τερματισμού αντίστοιχα. Αφού ο χρήστης έχει δώσει πλέον την είσοδό του πατάει το αντίστοιχο κουμπί που υποδηλώνει την ολοκλήρωση του σταδίου που αφορά την είσοδο στο σύστημα και την εκκίνηση της υπόλοιπης διαδικασίας. Παρακάτω παρατίθεται ο κώδικας javascript που αφορά την διαχείριση του χάρτη, των σημαδιών και της εισόδου του χρήστη. Αρχικά φορτώνονται οι εικόνες των σημαδιών. Έπειτα, διακρίνεται η μέθοδος onMapClick, η οποία εκτελείται μετά από κάθε κλικ πάνω στον χάρτη. Αρχικά ελέγχεται αν το κλικ που έγινε είναι το πρώτο ή το δεύτερο, αλλιώς αγνοείται. Αν είναι το πρώτο κλικ, στο σημείο που έγινε το κλικ τοποθετείται το πράσινο σημάδι και αποθηκεύονται η τετμημένη και τεταγμένη εκκίνησης του σημείου σε μεταβλητές. Αν είναι το δεύτερο κλικ, τοποθετείται το κόκκινο σημάδι και αποθηκεύονται οι συντεταγμένες του σημείου προορισμού σε μεταβλητές.

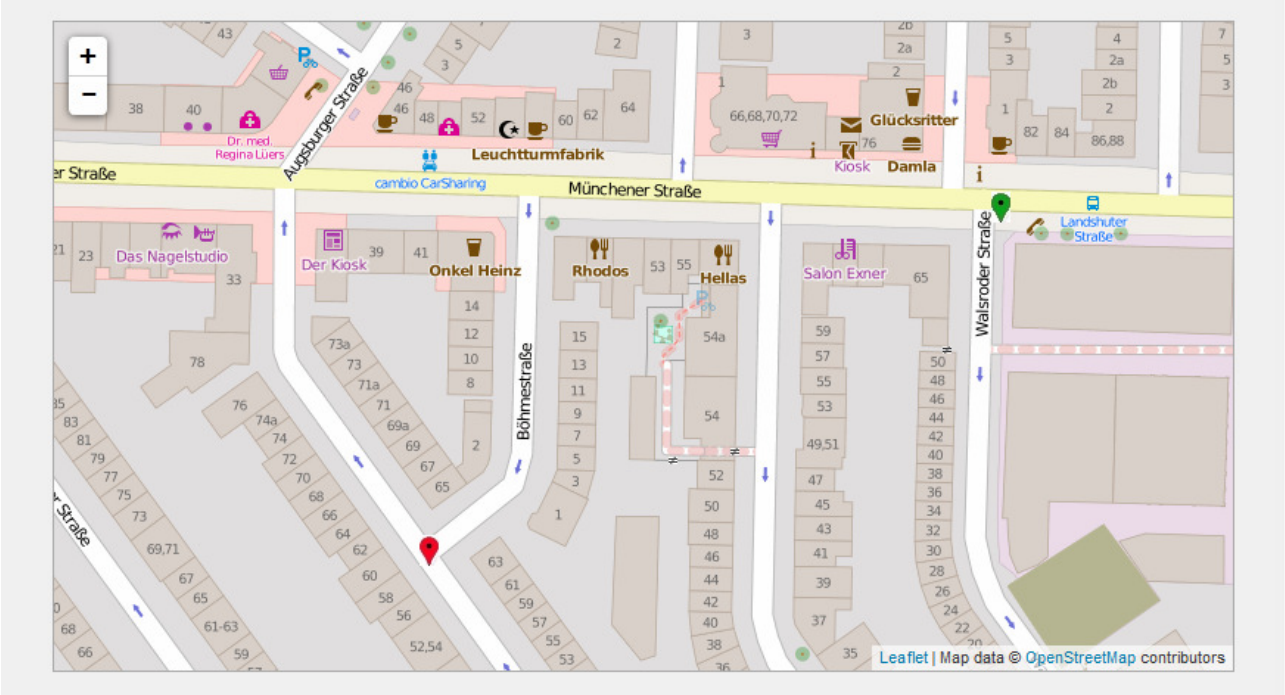
```
var startIcon = L.icon({
    iconUrl: 'path/to/images/green.png'
});

var endIcon = L.icon({
    iconUrl: 'path/to/images/red.png'
});

function onMapClick(e) {
    if(start_lon==undefined && start_lat==undefined){
        start_lon=e.latlng.lng;
        start_lat=e.latlng.lat
```

```
        marker_start = L.marker([start_lat,start_lon], {icon:
startIcon}).addTo(map);
    }else if(end_lon==undefined && end_lat==undefined){
        end_lon=e.latlng.lng;
        end_lat=e.latlng.lat
        marker_end = L.marker([end_lat,end_lon], {icon: endIcon}).addTo(map);
    }
}
map.on('click', onMapClick);
```

Παρακάτω παρατίθεται μια εικόνα (Εικόνα 15) με την ύπαρξη του χάρτη καθώς και με τα σημάδια εκκίνησης και τερματισμού.



Εικόνα 15:Ο χάρτης της εφαρμογής με τα σημάδια εκκίνησης και τερματισμού

Αποστολή δεδομένων στο εξυπηρετητή

Αφού πλέον τα δεδομένα από την είσοδο του χρήστη έχουν συγκεντρωθεί, το επόμενο βήμα αφορά την αποστολή τους στο εξυπηρετητή για την εύρεση των κοντινότερων κόμβων σε αυτά, σε πρώτο στάδιο, καθώς και την εύρεση του συντομότερου μονοπατιού στην συνέχεια. Όπως έχει αναφερθεί στο κεφάλαιο της εισόδου του χρήστη οι συντεταγμένες εκκίνησης και τερματισμού βρίσκονται αποθηκευμένες σε μεταβλητές στην javascript. Συνεπώς, το στάδιο αυτό περιγράφει την αποστολή τους στον εξυπηρετητή.

Η αποστολή των δεδομένων στον εξυπηρετητή γίνεται ασύγχρονα, δηλαδή με την κλήση AJAX. Η συγκεκριμένη τεχνική χρησιμοποιήθηκε, διότι κρίθηκε προτιμότερη η διατήρηση της υπάρχουσας σελίδας, δηλαδή η ύπαρξη του χάρτη μαζί με τα σημάδια εκκίνησης και τερματισμού, και στην συνέχεια η αποτύπωση της διαδρομής στον συγκεκριμένο χάρτη. Η συγκεκριμένη προσέγγιση κάνει την εφαρμογή απλούστερη καθώς δεν χρειάστηκε δημιουργία νέας σελίδας αλλά και κατανοητότερη από τον χρήστη, διότι μετά την αποτύπωση της διαδρομής, παρουσιάζεται στον χρήστη ο χάρτης, τα σημεία που επέλεξε και η διαδρομή που βρέθηκε μαζί.

Όσον αφορά το πρακτικό κομμάτι της κλήσης AJAX, χρησιμοποιήθηκε η βιβλιοθήκη javascript jQuery καθώς διαθέτει μια σειρά από συναρτήσεις που καθιστούν την διαδικασία ευκολότερη. Παρακάτω παρατίθεται το κομμάτι κώδικα(συνάρτηση) που πραγματοποιεί την ασύγχρονη μεταφορά δεδομένων από και προς τον server. Η παρακάτω συνάρτηση καλείται όταν ο χρήστης πατήσει το κουμπί για την εκτέλεση της δρομολόγησης. Αναλυτικά, αποστέλλει στο αρχείο routing.jsp του εξυπηρετητή ένα αίτημα της μορφής GET με τέσσερις μεταβλητές οι οποίες αποτελούν τις τετμημένες και τεταγμένες των σημείων εκκίνησης και τερματισμού. Επίσης, περιμένει να παραλάβει ένα αρχείο json το οποίο όπως φαίνεται από την συνάρτηση που καλείται στην περίπτωση επιτυχίας της διαδικασίας αποστέλλεται σε μια αυτόνομη συνάρτηση για την απομονωμένη ανάγνωση και ανάλυσή του.

```
function buttonCalculateFunction(){
    $.ajax({
        type : 'GET',
        url : 'routing.jsp',
        dataType : 'json',
        data: {
            startX : start_lon,
            startY : start_lat,
            endX : end_lon,
            endY : end_lat
        },
        success : function(json){
            jString=JSON.stringify(json,null,2);
            parseJson(jString);
        },
        error: function(){
            console.log("error during the procedure");
        },
        complete: function(){
            console.log("procedure completed");
        }
    });
}
```

Εύρεση γειτονικότερων σημείων

Μέχρι στιγμής, έχουν βρεθεί τα σημεία εκκίνησης και τερματισμού και έχουν αποσταλεί στον εξυπηρετητή. Το επόμενο στάδιο αφορά την αναζήτηση του κόμβου εκκίνησης και του κόμβου τερματισμού που είναι πιο κοντά στα σημεία εκκίνησης και τερματισμού αντίστοιχα.

Στην πραγματοποίηση της προαναφερθείσας διαδικασίας, ιδιαίτερο ρόλο διαδραματίζει η συμβολή της γεωγραφικής επέκτασης(plugin) που παρέχεται από το Neo4j. Όπως αναφέρθηκε στο κεφάλαιο που αφορά την δημιουργία του γραφήματος, κάθε κόμβος έχει ευρετηριοποιηθεί με την χρήση της συγκεκριμένης επέκτασης. Έτσι, πλέον μπορεί να εκτελεστεί αναζήτηση με βάση τα γεωγραφικά χαρακτηριστικά του κόμβου, και πιο συγκεκριμένα τις συντεταγμένες του.

Στα πλαίσια της υλοποίησης, αρχικά συλλέγονται οι συντεταγμένες του σημείου εκκίνησης και του σημείου τερματισμού. Στην συνέχεια, και για κάθε σημείο ξεχωριστά γίνεται αναζήτηση για την εύρεση των κόμβων του γραφήματος κοντά στο σημείο. Το κομμάτι κώδικα(μέθοδος) που παρατίθεται παρακάτω είναι υπεύθυνο για την διαδικασία αυτή. Αρχικά, στο τμήμα κώδικα που βρίσκεται εκτός της μεθόδου ενεργοποιείται/ανακτάται/ετοιμάζεται για χρήση το layer που χρησιμοποιείται για την αναπαράσταση των γεωγραφικών δεδομένων, έτσι ώστε να

γίνει αναζήτηση σε αυτό. Στην συνέχεια, όσον αφορά την μέθοδο, αξίζει να αναφερθεί ο τρόπος λειτουργίας της. Αρχικά, με βάση το σημείο εκκίνησης αναζητεί σε μια ακτίνα μήκους ενός μέτρου κόμβους του γραφήματος που να βρίσκονται γεωγραφικά μέσα στην ακτίνα αυτή. Αν δεν βρεθεί κανένας κόμβος, τότε το μέγεθος της ακτίνας διπλασιάζεται και συνεχίζει η ίδια διαδικασία αναζήτησης με διπλάσιο μήκος ακτίνας. Αυτή η διαδικασία συνεχίζεται μέχρι να βρεθεί ένας ή περισσότεροι κόμβοι. Τα αναγνωριστικά των κόμβων που έχουν βρεθεί, επιστρέφονται κατά φθίνουσα σειρά με την μορφή λίστας και επιλέγεται το πρώτο στοιχείο της λίστας σε μεταγενέστερο στάδιο ως κοντινότερος κόμβος. Προφανώς η διαδικασία διπλασιασμού της ακτίνας σταματάει όταν φτάσει σε ακραίες τιμές.

```
private static final double MAX_LENGTH=10000.0;

static GraphDatabaseService graphDb=Initialation.getDBConnection();
static SpatialDatabaseService spatialDb = new SpatialDatabaseService(graphDb);
static SimplePointLayer runningLayer = (SimplePointLayer)
spatialDb.getOrCreateLayer("running", SimplePointEncoder.class,
SimplePointLayer.class, "lon:lat");

private static ArrayList<Long> findPointsNearCoordinate(Coordinate
search_coordinate){
    double radius=0.001;// radius in km
    ArrayList<Long> nearestPoints=new ArrayList<Long>();
    List<GeoPipeFlow> results
=runningLayer.findClosestPointsTo(search_coordinate,radius);

    while(results.size()==0){
        if(radius>MAX_LENGTH){
            System.out.println("Point too far");
            return null;
        }
        radius=radius*2;
        results =runningLayer.findClosestPointsTo(search_coordinate,radius);
    }


    for(int i=0;i<results.size();i++){
        nearestPoints.add((Long)
results.get(i).getRecord().getGeomNode().getProperty("id"));
    }
    return nearestPoints;
 }
```

Υπολογισμός Συντομότερης Διαδρομής

Με την εύρεση των κοντινότερων κόμβων στα σημεία εκκίνησης και τερματισμού, το βήμα που ακολουθεί, είναι αυτό της εκτέλεσης του αλγορίθμου Dijkstra. Παρακάτω παρατίθεται το τμήμα κώδικα(κλάση) που έχει αναλάβει την εκτέλεση του αλγορίθμου. Ιδιαίτερη σημασία παρουσιάζουν τα αντικείμενα τύπου PathExpander και CostEvaluator. Το αντικείμενο PathExpander έχει ως σκοπό την συγκεκριμενοποίηση του τρόπου που ο αλγόριθμος διάσχισης του γραφήματος θα διαχειρίζεται και θα αντιμετωπίζει τις ακμές. Πιο συγκεκριμένα, καθορίζεται ότι στην διαδικασία διάσχισης οι ακμές που έχουν ορισθεί ως ONE_WAY θα αντιμετωπίζονται ως εξερχόμενες μόνο, ενώ οι ακμές που έχουν ορισθεί ως BIDIRECTIONAL είναι διπλής κατεύθυνσης. Συνάμα, το αντικείμενο CostEvaluator διαχειρίζεται το κόστος/βάρος της ακμής. Στην ουσία, η υλοποίηση του MyCostEvaluator απλά περιλαμβάνει την ανάγνωση του βάρους της εκάστοτε εξεταζόμενης ακμής για την απόφαση αν θα την προσπελάσει ή όχι. Στο κύριο τμήμα της υλοποίησης ανακτώνται οι κόμβοι εκκίνησης και τερματισμού, που έχουν βρεθεί από το

προηγούμενο στάδιο, με την χρήση των αναγνωριστικών τους. Στην συνέχεια εκτελείται ο αλγόριθμος dijkstra για την εύρεση της συντομότερης διαδρομής. Όπως φαίνεται και στον κώδικα, για την εκτέλεση του αλγορίθμου απαραίτητα είναι οι κόμβοι εκκίνησης και τερματισμού, ένα αντικείμενο CostEvaluator για τον υπολογισμό του κόστους της κάθε ακμής και ένα αντικείμενο PathExpander για τον καθορισμό του επιτρεπόμενου τρόπου προσπέλασης των ακμών. Ο αλγόριθμος επιστρέφει ένα σταθμισμένο μονοπάτι το οποίο αποτελείται από κόμβους και ακμές, τα οποία αποθηκεύονται μαζί σε μια λίστα η οποία επιστρέφεται.

```
public class DijkstraImplementation{
    int numberOfNodesCrossed;
    static GraphDatabaseService graphDb = Initialation.getDBConnection();
    private final PathExpander expander =
PathExpanders.forTypesAndDirections(ExampleTypes.ONE_WAY,
Direction.OUTGOING,ExampleTypes.BIDIRECTIONAL, Direction.BOTH);
    private final CostEvaluator<Double> costEvaluator = new MyCostEvaluator();

    public ArrayList <GenericObj> runDijkstra(long source_id, long target_id) {
        try{
            Node source = Querries.getNodeFromId(source_id);
            Node target = Querries.getNodeFromId(target_id);
            numberOfNodesCrossed=0;
            System.out.println("Calculating route for the below nodes: ");
            System.out.println("source_id: "+source_id);
            System.out.println("target_id: "+target_id);

            try(Transaction tx = graphDb.beginTx()){
                PathFinder<WeightedPath> finder =
GraphAlgoFactory.dijkstra(expander, costEvaluator);
                WeightedPath path = finder.findSinglePath(source, target);
                if(path==null){
                    return null;
                }
                ArrayList <GenericObj> paths = new ArrayList <GenericObj>();

                for(PropertyContainer pc:path){


                    if(pc instanceof Node){
                        long id=(Long)((Node) pc).getProperty("id");
                        double lat=(Double)((Node) pc).getProperty("lat");
                        double lon=(Double)((Node) pc).getProperty("lon");
                        GenericObj obj=new
com.mycompany.mavenproject1.Node(lon,lat,id);
                        paths.add(obj);
                        numberOfNodesCrossed++;
                        System.out.println(((Node) pc).getProperty("id"));
                    }else{
                        long id=(Long)((Relationship)
pc).getProperty("road_id");
                        String name=(String)((Relationship)
pc).getProperty("name");
                        String highwayType=(String)((Relationship)
pc).getProperty("highway_type");
                        String surfaceType=(String)((Relationship)
pc).getProperty("surface_type");
                        int maxspeed=(Integer)((Relationship)
pc).getProperty("maxspeed");

                        GenericObj obj=new
com.mycompany.mavenproject1.Relationship(name,highwayType,surfaceType,maxspeed,i
d);
```

```
                        paths.add(obj);

                    }

                }
                tx.success();

                System.out.println("Number of nodes crossed:
"+numberOfNodesCrossed);
                return paths;
            }
        }catch(Exception e){
            System.out.println(e);
            e.printStackTrace();
        }
        return null;
    }
}
```

Μετά τον υπολογισμό του συντομότερου μονοπατιού, σειρά στην διαδικασία έχει η αποστολή της ευρεθείσας διαδρομής πίσω στον εξυπηρετητή για την εμφάνιση της διαδρομής στον χάρτη. Η αποστολή της διαδρομής στον εξυπηρετητή, γίνεται με την αποθήκευση των δεδομένων που αποτελούν την διαδρομή, σε ένα αρχείο json και την αποστολή του αρχείου αυτού. Παρακάτω παρατίθεται ένα παράδειγμα json αρχείου μιας μικρής διαδρομής για περαιτέρω διασαφήνιση. Η δομή του αρχείου περιλαμβάνει τον κόμβο εκκίνησης. Στην συνέχεια ακολουθεί το κομμάτι δρόμου που ενώνει τον αρχικό κόμβο με τον επόμενο κόμβο. Η διαδικασία αυτή, δηλαδή κόμβος-κομμάτι δρόμου-κόμβος συνεχίζεται μέχρι τον κόμβο τερματισμού.


```
[
  {
    "id":29037641,
    "lon":8.8039122,
    "lat":53.0960439
  },
  {
    "id":858771676,
    "name":"Regensburger Straße",
    "maxspeed":30,
    "surface_type":"ASPHALT",
    "highway_type":"RESIDENTAL"
  },
  {
    "id":29073591,
    "lon":8.8032967,
    "lat":53.0962967
  },
  {
    "id":45985430,
    "name":"Kulmbacher Straße",
    "maxspeed":30,
    "surface_type":"COBBLESTONE",
    "highway_type":"RESIDENTAL"
  },
  {
    "id":1920927931,
    "lon":8.8020519,
    "lat":53.0951908
  }
]
```

Εμφάνιση διαδρομής

Ύστερα από την αποστολή των δεδομένων της διαδρομής από τον εξυπηρετητή στον φυλλομετρητή, σειρά έχει η διαχείριση των δεδομένων αυτών και η εμφάνιση της διαδρομής στον χρήστη γραφικά.

Όσον αφορά την ανάγνωση των δεδομένων, αρχικά γίνεται ανάγνωση και ανάλυση των δεδομένων που βρίσκονται στο αρχείο json που επιστρέφεται από τον εξυπηρετητή με σκοπό την εξαγωγή μόνο της χρήσιμης πληροφορίας από αυτό. Η διαδικασία αυτή πραγματοποιείται στο παρακάτω κομμάτι κώδικα στην συνάρτηση parseJson. Αναλυτικότερα, το αρχείο json διαβάζεται και συγκρατείται μόνο πληροφορία σχετικά με τους κόμβους(συντεταγμένες) από τους οποίους διέρχεται η διαδρομή. Την εξέλιξη της διαδικασίας, για την τύπωση της διαδρομής αναλαμβάνει η συνάρτηση renderRoute που θα αναλυθεί παρακάτω.

```
function parseJson(jstring){
    parsed_data=JSON.parse(jstring);
    nodes=[];

    for (var i=0; i<parsed_data.length; i++){
        if(parsed_data[i].hasOwnProperty("lat")){
            new_node={keyword_id:parsed_data[i].id,
keyword_lon:parsed_data[i].lon ,keyword_lat:parsed_data[i].lat};
            nodes.push(new_node);
            console.log(parsed_data[i].id);
            console.log(parsed_data[i].lat);
            console.log(parsed_data[i].lon);
        }else{
            console.log(parsed_data[i].id);
            console.log(parsed_data[i].name);
        }

    }
    renderRoute();
}
```

Σε αυτό το σημείο, αξίζει να σημειωθεί, ότι ο λόγος που μεταφέρονται στον φυλλομετρητή πληροφορίες που εν τέλει δεν χρησιμοποιούνται όπως για παράδειγμα τα ονόματα των δρόμων που περνά η διαδρομή, είναι για λόγους πιθανής χρησιμοποίησης τους σε περίπτωση επέκτασης των λειτουργιών της εφαρμογής.

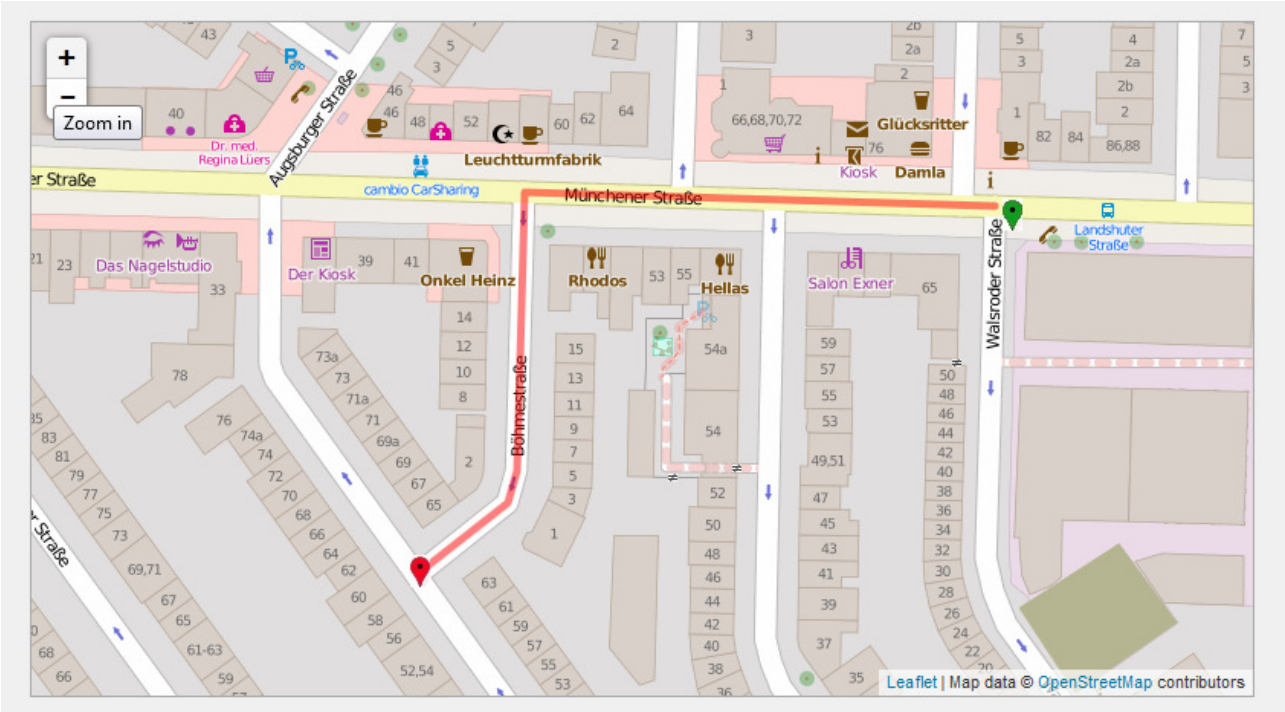
Το τελευταίο στάδιο αφορά την αποτύπωση της διαδρομής, δηλαδή την εμφάνιση στον χρήστη της διαδρομής γραφικά. Η προσέγγιση που χρησιμοποιήθηκε για την σχεδίαση της γραμμής στον χάρτη είναι η συλλογή όλων των συντεταγμένων που σχηματίζουν την διαδρομή και η αποθήκευση τους σε έναν πίνακα. Στην συνέχεια, η ένωση όλων των διαδοχικών συντεταγμένων με μία γραμμή θα οδηγήσει στην εμφάνιση όλης της διαδρομής στον χάρτη. Για την αποτύπωση της διαδρομής στον χάρτη έγινε χρήση της βιβλιοθήκης Leaflet.js. Το κομμάτι κώδικα(συνάρτηση) που χρησιμοποιείται για την εμφάνιση της διαδρομής παρατίθεται παρακάτω. Όπως φαίνεται και στον κώδικα όλες οι συντεταγμένες αποθηκεύονται σε έναν πίνακα. Σε αυτό το σημείο αξίζει να σημειωθεί ότι οι συντεταγμένες που βρίσκονται στον πίνακα αποτελούν ένα πολύγραμμο(Polyline). Η βιβλιοθήκη Leaflet.js παρέχει μια έτοιμη υλοποίηση για την διαχείριση πολύγραμμων. Έτσι, όπως φαίνεται και στον κώδικα, ο πίνακας δίνεται απλά σαν όρισμα στην έτοιμη υλοποίηση της

βιβλιοθήκης, σε συνδυασμό με το χρώμα της γραμμή(στην προκειμένη περίπτωση κόκκινο) με αποτέλεσμα την εκτύπωση της διαδρομής.

```
function renderRoute(){
    var lastNodeIndex;
    polylinelatlong=[];
    for(var n=0, count=nodes.length;n<count;n++){
        polylinelatlong.push([nodes[n].keyword_lat, nodes[n].keyword_lon]);
        lastNodeIndex=n;
    }
    lastNodeIndex=lastNodeIndex-1;
    polyline = L.polyline(polylinelatlong,{color:'red'}).addTo(map);

}
```

Παρακάτω ακολουθεί εικόνα (Εικόνα 16) με τον χάρτη και την διαδρομή εκτυπωμένη.

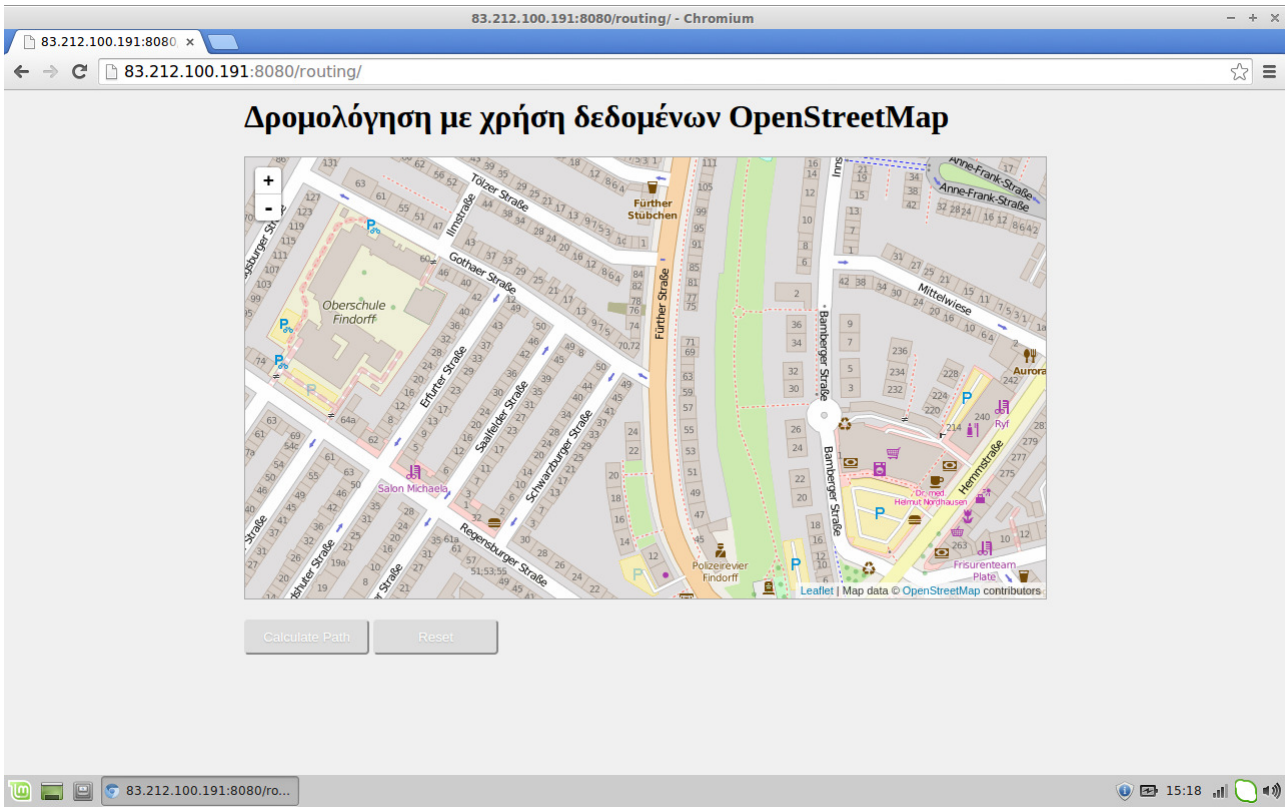


Εικόνα 16:Ο χάρτης της εφαρμογής με τα σημάδια εκκίνησης και τερματισμού, καθώς και της αποτυπωμένης διαδρομής

Εκτέλεση Εφαρμογής

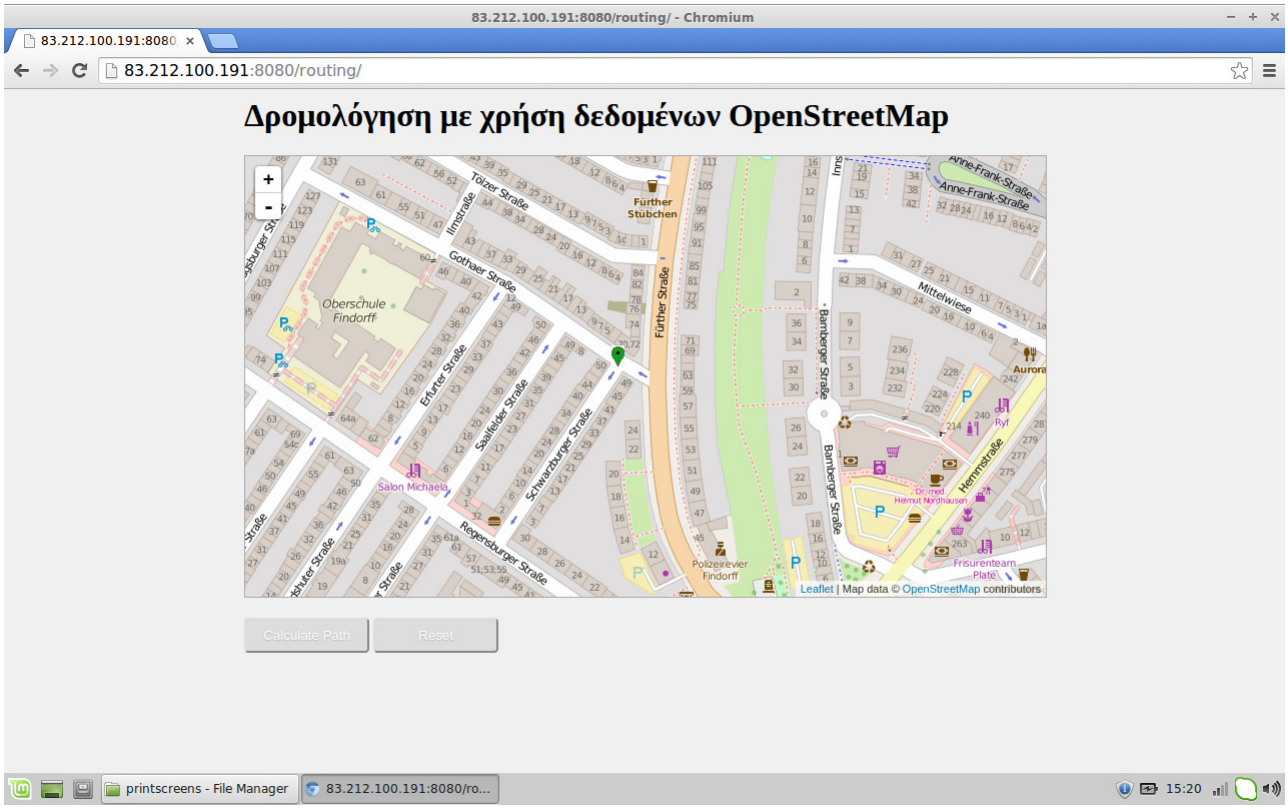
Παρακάτω ακολουθεί ένα σύνολο φωτογραφιών οθόνης (prinstscreens) που αναπαριστούν σταδιακά την διαδικασία της εκτέλεσης της εφαρμογής.

Στην εικόνα 17 παρουσιάζεται η εφαρμογή κατά την εκκίνηση της. Στο σημείο αυτό, απλά εμφανίζεται ένας χάρτης και κάτω από τον χάρτη εμφανίζονται δύο κουμπιά. Ένα με τίτλο "Calculate Path" και ένα με τίτλο "Reset".



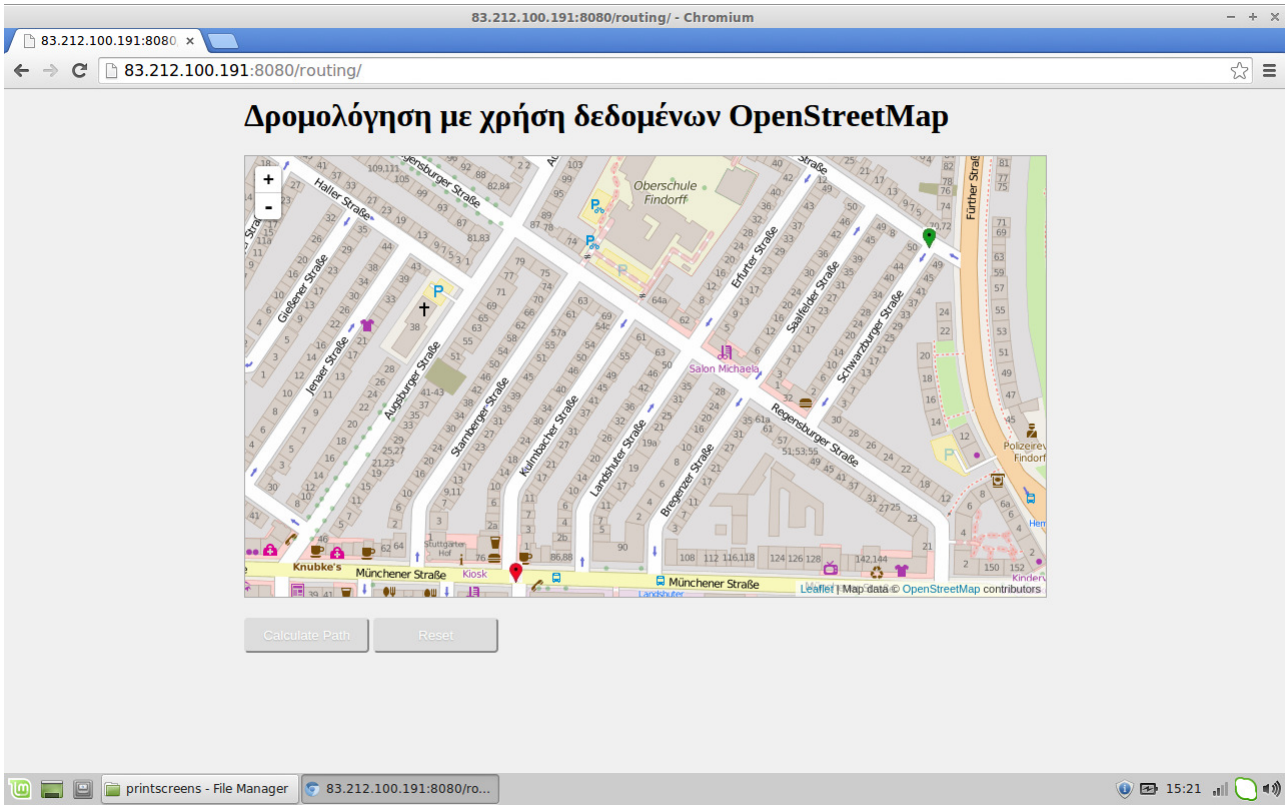
Εικόνα 17: Η οθόνη της εφαρμογής κατά την εκκίνησή της

Στην εικόνα 18 ο χρήστης έχει κάνει κλικ στον χάρτη και έχει εμφανιστεί στο σημείο που έκανε κλικ ένα πράσινο σημάδι, το οποίο υποδηλώνει ότι ο χρήστης έχει θέσει σημείο εκκίνησης για την διαδικασία της εύρεσης διαδρομής.



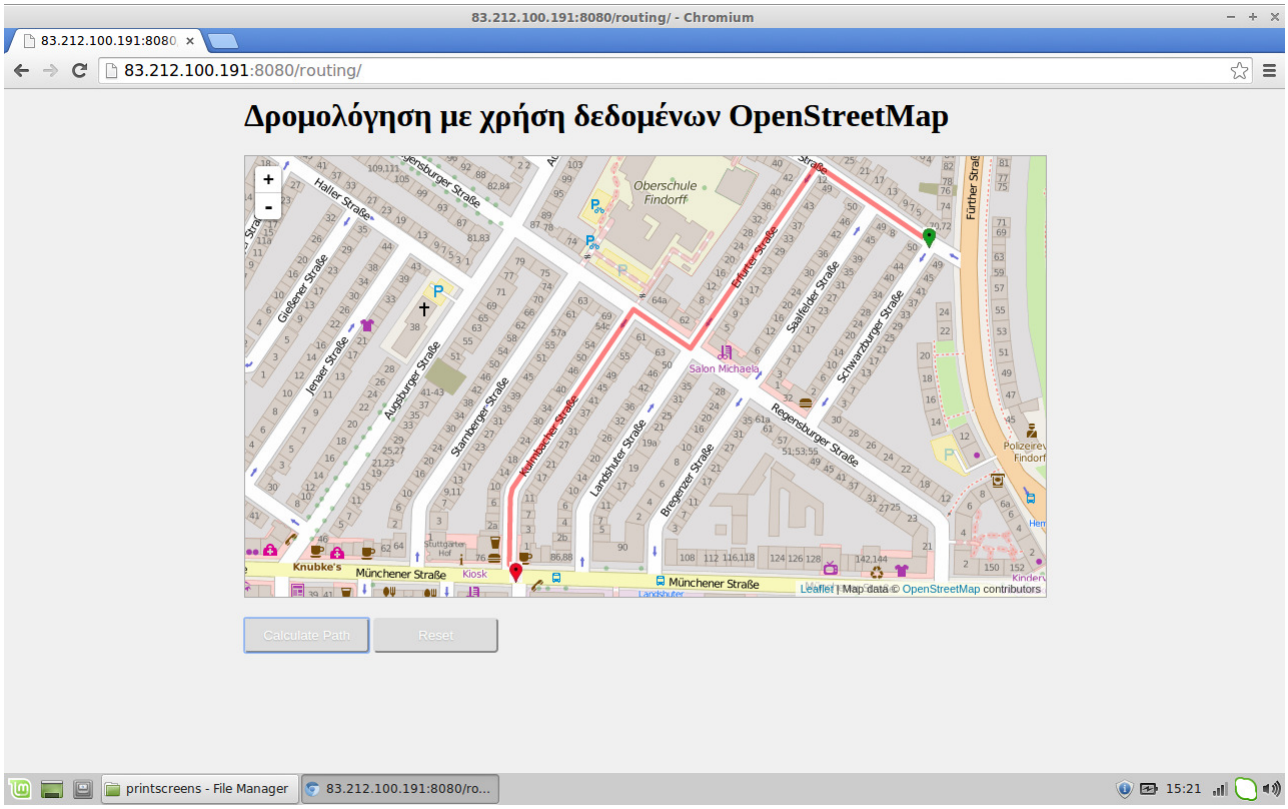
Εικόνα 18: Η οθόνη της εφαρμογής μετά την επιλογή του σημείου εκκίνησης

Στην εικόνα 19 ο χρήστης έχει κάνει άλλο ένα κλικ στον χάρτη και έχει εμφανιστεί στο σημείο που έκανε κλικ ένα κόκκινο σημάδι, ενώ ταυτόχρονα συνεχίζει να υπάρχει το πράσινο σημείο. Το κόκκινο σημάδι υποδηλώνει ότι το σημείο εκείνο αποτελεί το σημείο τερματισμού της δρομολόγησης. Έτσι, η τρίτη εικόνα παρουσιάζει τον χάρτη, μετά από το δεύτερο κλικ του χρήστη, όπου υπάρχει ένα πράσινο σημάδι στο σημείο εκκίνησης και ένα κόκκινο σημάδι στο σημείο τερματισμού.



Εικόνα 19: Η οθόνη της εφαρμογής μετά την επιλογή και του σημείου τερματισμού

Στην εικόνα 20 παρουσιάζεται η εφαρμογή μετά το πάτημα του κουμπιού “Calculate Path”. Στην εικόνα φαίνεται ότι έχει εμφανιστεί η διαδρομή που ζητήθηκε από τον χρήστη σύμφωνα με τα σημεία εκκίνησης και τερματισμού. Η διαδρομή σχηματίζεται από το κόκκινο πολύγραμμο.



Εικόνα 20: Η οθόνη της εφαρμογής μετά την εμφάνιση της ευρεθείσας διαδρομής

Σε αυτό το σημείο αξίζει να σημειωθεί ότι ο χρήστης σε οποιοδήποτε σημείο της αλληλεπίδρασης του με την εφαρμογή μπορεί να πατήσει το κουμπί “Reset” το οποίο είναι υπεύθυνο για την επανεκκίνηση της όλης διαδικασίας. Το συγκεκριμένο κουμπί χρειάζεται για περιπτώσεις λανθασμένης εισόδου από τον χρήστη όπου επιθυμεί να ξανά-εισάγει τα σημεία που τον ενδιαφέρουν ή σε περίπτωση που ο χρήστης θέλει να υπολογίσει νέα διαδρομή μετά την εμφάνιση κάποιας διαδρομής.

Βιβλιογραφία

1. Cormen, T.H., Leiserson, C.E., Rivest, R.L & Stein, C. (2001) Introduction to Algorithms. MIT Press Από Cormen et al.(2001), σελ 596
2. Delling, D. (2009). *Engineering and augmenting route planning algorithms* (Doctoral dissertation, Karlsruhe Institute of Technology). (Πρόσβαση 22/7/2015)
3. Delling, D., Sanders, P., Schultes, D., & Wagner, D. (2009). Engineering route planning algorithms. In Algorithmics of large and complex networks (pp. 117-139). Springer Berlin Heidelberg. (Πρόσβαση 20/7/2015)
4. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269-271.(Πρόσβαση 15/6/2015)
5. Finkel, R. A., & Bentley, J. L. (1974). Quad trees a data structure for retrieval on composite keys. Acta informatica, 4(1), 1-9. (Πρόσβαση 25/7/2015)
6. Guttman, A. (1984). *R-trees: a dynamic index structure for spatial searching* (Vol. 14, No. 2, pp. 47-57). ACM. (Πρόσβαση 25/7/2015)
7. Roussopoulos, N., Kelley, S., & Vincent, F. (1995, June). Nearest neighbor queries. In *ACM sigmod record* (Vol. 24, No. 2, pp. 71-79). ACM. (Πρόσβαση 25/7/2015)
8. Sanders, P., & Schultes, D. (2007). Engineering fast route planning algorithms. In *Experimental Algorithms* (pp. 23-36). Springer Berlin Heidelberg. (Πρόσβαση 20/7/2015)
9. Schultes, D. (2008, February). Route Planning in Road Networks. In Ausgezeichnete Informatikdissertationen (pp. 271-280). (Πρόσβαση 15/6/2015)
10. Wagner, D., & Willhalm, T. (2007). Speed-up techniques for shortest-path computations. In STACS 2007 (pp. 23-36). Springer Berlin Heidelberg. (Πρόσβαση 22/7/2015)
11. http://www.openstreetmap.org/?lat=49.2625&lon=4.02934&zoom=16&layers=M
12. https://wiki.openstreetmap.org/wiki/OSM_tags_for_routing#Roads
13. http://wiki.openstreetmap.org/wiki/Key:highway#Values
14. http://wiki.openstreetmap.org/wiki/Key:surface
15. http://wiki.openstreetmap.org/wiki/Key:maxspeed#Parser
16. http://wiki.openstreetmap.org/wiki/OSM_tags_for_routing/Maxspeed
17. https://github.com/neo4j-contrib/spatial
18. leafletjs.com/
19. www.openstreetmap.org
20. http://neo4j.com/
21. http://www.postgresql.org/
22. http://postgis.net/
23. https://glassfish.java.net/
24. http://wiki.openstreetmap.org/wiki/Osm2pgsql