Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ CARPOOLING ΣΕ ANDROID ΛΕΙΤΟΥΡΓΙΚΟ

Παναγιώτης Ανδρέου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής Carpooling σε Android λειτουργικό

Παναγιώτης Ανδρέου

Επιβλέπων Καθηγητής
Γιώργος Πάλλης

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

Μάιος 2011

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον λέκτορα κ. Γιώργο Πάλλη που μου έδωσε την ευκαιρία να υλοποιήσω αυτή την εφαρμογή αλλά επίσης και για τις πολύτιμες συμβουλές και χρήσιμες πληροφορίες που μου παρείχε στα διάφορα στάδια της διπλωματικής εργασία που είχαν σαν αποτέλεσμα την επιτυχή ολοκλήρωσης της.

Επίσης θα ήθελα να ευχαριστήσω τον λέκτορα κ. Δημήτρη Ζεϊναλιπούρ και το Πανεπιστήμιο Κύπρου για την προσφορά της κινητής συσκευής ούτως ώστε να γίνει κατορθωτή η υλοποίηση της εφαρμογής.

Επίσης θέλω να ευχαριστήσω τον συμφοιτητή και φίλο μου Χάρη Ευσταθιάδη για την άψογη βοήθεια, συνεργασία και ανταλλαγή απόψεων που είχαμε κατά την διάρκεια της υλοποίησης της διπλωματικής εργασίας.

Επιπλέον θα ήθελα να δώσω τις θερμές μου ευχαριστίες στους φίλους και συμφοιτητές μου που δοκίμασαν την εφαρμογή και με βοήθησαν στον έλεγχο και την αξιολόγηση της.

Περίληψη

Μια εφαρμογή Carpooling έχει ως σκοπό να βρίσκει τα κοινά δρομολόγια που υπάρχουν μεταξύ των χρηστών της, και να τους βοηθά να εκτελούν αυτά τα δρομολόγια με ένα αυτοκίνητο. Με τη χρήση της εφαρμογής μπορούμε να συνεισφέρουμε θετικά σε διάφορα προβλήματα που αντιμετωπίζει η ανθρωπότητα στις μέρες μας όπως για παράδειγμα:

i. Στην μείωση του κυκλοφοριακού προβλήματος.
ii. Στην μείωση των καυσαερίων στην ατμόσφαιρα.
iii. Στην εξοικονόμηση ενέργειας.

Επίσης με τη χρήση της εφαρμογής οι χρήστες θα έχουμε και οικονομικό όφελος επειδή θα μοιράζονται τα κοινά δρομολόγια με τους φίλους τους και έτσι θα επωφελούνται από την εξοικονόμηση καυσίμων.

Στα πλαίσια της διπλωματικής εργασίας υλοποιήσαμε την εφαρμογή Hermes σε Android λογισμικό η οποία έχει τα ακόλουθα χαρακτηριστικά:

i. Χρησιμοποιεί δικό της κοινωνικό δίκτυο όπου οι χρήστες μπορούν να ανταλλάσουν μηνύματα με τους φίλους τους και να συμμετέχουν σε δρομολόγια τα οποία προσθέτονται από τους φίλους τους μόνο.
ii. Δίνει την δυνατότητα πρόσθεσης των φίλων του χρήστη από το κοινωνικό δίκτυο Facebook.
iii. Χρησιμοποιεί αλγόριθμο εύρεσης πιθανών δρομολογίων όπου βάσει κάποιων δεδομένων εισόδου, και ενός δρομολογίου βρίσκει τα όμοια δρομολόγια από τους φίλους του χρήστη, τα οποία μπορούν να τον εξυπηρετήσουν.
iv. Δίνεται αμοιβή στους χρήστες οι οποίοι προσθέτουν και εκτελούν δρομολόγια, σύμφωνα με την απόσταση και τον αριθμό των ατόμων που μεταφέρουν.

Ο τρόπος με τον οποίο προσεγγίσαμε το πρόβλημα θα βοηθήσει τους χρήστες να χρησιμοποιούν εύκολα την εφαρμογή από την κινητή τους συσκευή, αλλά επίσης και να μοιράζονται τα δρομολόγια τους με τους υπόλοιπους χρήστες γιατί θα είναι διαθέσιμα μόνο προς τους φίλους τους και θα υπάρχει εμπιστοσύνη μεταξύ τους.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Έξυπνα κινητά (smart phones)

Τα έξυπνα κινητά προσφέρουν τις λειτουργίες ενός ηλεκτρονικού υπολογιστή σε μια μικρότερη του έκδοση και έχουν κατακλύσει την παγκόσμια αγορά κινητής τηλεφωνίας[11]. Τα έξυπνα κινητά δεν έχουν ελκύσει μόνο το ενδιαφέρον της τεχνολογίας στις μέρες μας αλλά επίσης και το ενδιαφέρον των απλών χρηστών. Ο λόγος για τον οποίο το έχουν επιτύχει αυτό είναι οι δυνατότητες που προσφέρουν σε συνδυασμό πάντοτε με την φορητότητα τους, αλλά επίσης και το χαμηλό κόστος αγοράς τους.

Τα πλείστα έξυπνα κινητά χαρακτηρίζονται από υπολογιστική ισχύ, GPS, πρόσβαση στο διαδίκτυο αλλά επίσης και πολλούς αισθητήρες που είναι ενσωματωμένοι με την κινητή συσκευή. Αυτά τα χαρακτηριστικά επιτρέπουν στους κατασκευαστές προγραμμάτων να αναπτύξουν νέες καινοτόμες εφαρμογές ειδικευμένες στα έξυπνα κινητά στις μέρες μας. Αυτό το προνόμιο δεν ήταν εφικτό παλαιότερα γιατί οι κινητές συσκευές περιόριζαν τις διάφορες εφαρμογές κυρίως υπολογιστικά.

1.1.1 Λειτουργικό σύστημα Android

Το android είναι ένα λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού συστήματος Linux. Το λειτουργικό σύστημα Adroid επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας

προγραμματισμού Java, ελέγχοντας την κινητή συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google (Android SDK).

Το λογισμικό του Android το οποίο είναι γραμμένο σε Java επιτρέπει στους κατασκευαστές λογισμικού να το μεταγλωττίσουν και να το εκτελέσουν σε εικονική μηχανή Dalvik, η οποία είναι μια εξειδικευμένη υλοποίηση εικονικής μηχανής, σχεδιασμένη για χρήση από φορητές συσκευές, παρόλο που δεν είναι μια πρότυπη εικονική μηχανή.

Το λειτουργικό σύστημα Android υποστηρίζει τεχνολογίες συνδεσιμότητας GSM, Bluetooth, Wi-Fi, CDMA. Επίσης κάποιες άλλες σημαντικές τεχνολογίες τις οποίες υποστηρίζει είναι το GPS και οι διάφοροι αισθητήρες κίνησης.

Τα πιο πάνω χαρακτηριστικά γνωρίσματα των κινητών συσκευών που υποστηρίζουν λειτουργικό Android, αλλά επίσης και η ευκολία χρήσης και ανάπτυξης εφαρμογών σε αυτό ήταν οι σημαντικότεροι παράγοντες που οδηγηθήκαμε στην επιλογή της συγκεκριμένης πλατφόρμας.

Τέλος να αναφέρουμε πως σύμφωνα με μελέτες που έγιναν τον Ιανουάριο του 2011 για την προτίμηση των χρηστών σε λειτουργικά που χρησιμοποιούνται στις μέρες μας σε έξυπνα κινητά κατέδειξαν πως το λειτουργικό Android κατέχει την πλειοψηφία των κατόχων έξυπνων κινητών με ποσοστό 33% έναντι άλλων λειτουργικών όπως Symbian, Apple, Windows mobile κτλ [5].

1.2 Carpooling

Ο γενικός όρος του Carpooling είναι η πρακτική κατά την οποία κάποιος που ταξιδεύει μόνος του με το αυτοκίνητο του να δέχεται και άλλους επιβάτες μαζί του γνωστούς ή άγνωστους προς αυτόν με σκοπό να μοιραστούν τα έξοδα του ταξιδιού που προκύπτουν όπως για παράδειγμα την βενζίνη, τα διόδια κτλ.

Το Carpooling μπορεί να βοηθήσει την κοινωνία στην αντιμετώπιση σημαντικών προβλημάτων που αντιμετωπίζουμε στις μέρες μας όπως για παράδειγμα:

1. Αντιμετώπιση της περιβαλλοντικής ρύπανσης της ατμόσφαιρας από τα καυσαέρια των οχημάτων.
2. Εξοικονόμηση ενέργειας (πετρελαίου) .

3. Αντιμετώπιση του κυκλοφοριακού προβλήματος που αντιμετωπίζουμε καθημερινά όλοι μας στους πολυσύχναστους δρόμους της πόλης μας.
4. Εξοικονόμηση χρημάτων για τους επιβάτες αλλά και τον οδηγό.
5. Επίσης σε κάποιες χώρες υπάρχει ειδική λωρίδα στους δρόμους η οποία μπορεί να χρησιμοποιείται από αυτοκίνητα στα οποία ταξιδεύουν 2 άτομα και άνω. Με αυτήν την ενέργεια της κοινότητας αναγνωρίζεται η σημαντικότητα της ιδέας του Carpool.
6. Τα καθημερινά δρομολόγια που εκτελούνται με το αυτοκίνητο μας μπορούν να γίνουν πιο διασκεδαστικά με την συμμετοχή και άλλων ατόμων.

Σε κάποιες χώρες μπορεί το Carpooling να δίνει πλεονέκτημα στον οδηγό να παίρνει κάποιους πόντους (Credits) αναλόγως με την απόσταση του ταξιδίου και τον αριθμό των επιβατών, όπου αργότερα μπορεί να χρησιμοποιήσει αυτούς τους πόντους σε διάφορες δημόσιες υπηρεσίες, όπως για παράδειγμα στα διόδια, σε δημόσιους χώρους στάθμευσης ή ακόμη και σε μειωμένο κόστος καύσιμης ύλης. Μια άλλη προσέγγιση του Carpooling είναι οι επιβάτες να μοιράζονται τα έξοδα του ταξιδίου όλοι μαζί με τον οδηγό.

Ο τρόπος υλοποίησης και προσέγγισης της εφαρμογής Carpooling διαφέρει σε κάθε εφαρμογή. Για τον λόγο αυτό μελετήσαμε εφαρμογές Carpooling οι οποίες είναι υλοποιημένες σε Android λειτουργικό ή όχι και μετά από καταγραφή των σημαντικών στοιχείων καταλήξαμε σε μια ολοκληρωμένη άποψη για το πως πρέπει να προσεγγίσουμε την εφαρμογή ούτως ώστε να δώσουμε στον χρήστη τα κατάλληλα κίνητρα για να την χρησιμοποιεί.

1.2.1 Κίνητρα υλοποίησης εφαρμογής Carpooling

Σύμφωνα με μια μελέτη που έγινε στη Νέα - Υόρκη Αμερικής το 2011 έδειξαν πως κατά μέσο όρο το 54% των οδηγών ταξιδεύουν μόνοι τους χωρίς συνεπιβάτες στο αυτοκίνητο τους [6]. Αυτό είναι πολύ μεγάλο ποσοστό και είναι ένα από τα κυριότερα κίνητρα για την δημιουργία μιας εφαρμογής Carpooling.

Επίσης όπως έχουμε αναφέρει και πριν η κάθε υλοποίηση μιας Carpooling εφαρμογής διαφέρει στον τρόπο προσέγγισης της με κάποια άλλη αναλόγως με την χώρα για την οποία αναπτύσσεται, από τους χρήστες στους οποίους απευθύνεται αλλά επίσης και από τα κίνητρα που θέλει να δώσει στους χρήστες της η κάθε μια αναλόγως.

Επίσης μετά από μελέτη διαφόρων εφαρμογών Carpooling παρατηρήσαμε ότι οι πλείστες εφαρμογές επιτρέπουν στους χρήστες της να ταξιδεύουν μαζί χωρίς να λαμβάνουν υπόψη τους αν είναι φίλοι ή όχι (αυτό δεν σημαίνει απαραιτήτως ότι είναι μειονέκτημα των άλλων εφαρμογών), και στις πλείστες περιπτώσεις η αμοιβή του οδηγού είναι χρηματικό ποσό.

Οι πιο πάνω λόγοι αλλά και η σημαντικότητα του Carpool που έχει στις μέρες μας κυρίως από περιβαλλοντική άποψη ήταν τα κύρια κίνητρα που μας οδήγησαν να δημιουργήσουμε μια εφαρμογή Carpooling. Θέσαμε σαν στόχο να δημιουργήσουμε μια αποτελεσματική εφαρμογή η οποία να ενθαρρύνει τον κόσμο όσο το δυνατόν περισσότερο να χρησιμοποιεί την ιδέα του Carpool και, αν είναι δυνατό να επιτύχουμε να υιοθετηθεί αυτός ο τρόπος σκέψης ανεξάρτητα από τις διάφορες χώρες.

Πρωταρχικός μας στόχος είναι ο χρήστης να μπορεί να έχει πρόσβαση στην εφαρμογή από όπου και αν βρίσκεται. Επίσης θέλουμε να ενισχύσουμε την ιδέα του Carpooling και να δώσουμε στους χρήστες της εφαρμογής κίνητρα για να την χρησιμοποιούν.

Για τον λόγο αυτό αποφασίσαμε ότι θα αναπτύξουμε την εφαρμογή σε έξυπνη κινητή συσκευή και πιο συγκεκριμένα σε Android λειτουργικό, όπου με αυτό τον τρόπο θα επιτύχουμε την φορητότητα της εφαρμογής μας.

Ακολούθως θέλαμε να δημιουργήσουμε μια εφαρμογή Carpooling η οποία θα εμπνέει την εμπιστοσύνη και την οικειότητα προς τους χρήστες της. Έτσι αποφασίσαμε πως η εφαρμογή που θα υλοποιήσουμε θα είναι βασισμένη σε κοινωνικό δίκτυο, όπου με αυτή την ιδιαιτερότητα της εφαρμογής μας επιτρέπουμε στους χρήστες να συμμετέχουν σε δρομολόγια τα οποία ανήκουν μόνο στους φίλους τους. Έτσι θα έχουμε το πλεονέκτημα όπου ο χρήστης δεν θα διατρέχει ανασφάλειες για τον οδηγό γιατί θα είναι άτομο το οποίο θα γνωρίζει. Επίσης αυτή η προσέγγιση που ακολουθήσαμε βοηθά τους χρήστες στο να δουν πια δρομολόγια προστίθενται από τους φίλους τους, να δουν ποιοι άλλοι συμμετέχουν μαζί του, και με αυτό τον τρόπο μετατρέπεται το Carpooling σε μια διασκεδαστική διαδικασία.

Επίσης πέραν από αυτό θέλουμε να δώσουμε και στον οδηγό κίνητρα και προσωπικό όφελος για να χρησιμοποιήσει την εφαρμογή μας. Για αυτό τον λόγο καταλήξαμε πως ο οδηγός θα μπορεί να μοιράζεται τα έξοδα του ταξιδίου με τους υπόλοιπους επιβάτες του αυτοκινήτου αν το θέλει χωρίς να το καθιστά υποχρεωτικό. Με αυτή την προσέγγιση ο οδηγός θα μπορεί να απαιτήσει από τους υπόλοιπους επιβάτες να μοιραστούν τα έξοδα, και οι υπόλοιποι επιβάτες θα το γνωρίζουν από πριν.

Επίσης πέραν της χρηματικής αμοιβής που μπορεί να λάβει ο οδηγός αν το θελήσει, παίρνει πάντοτε την αμοιβή των Credits η οποία θα είναι ανάλογη της απόστασης του ταξιδιού, τον αριθμό των επιβατών, αλλά επίσης και της εξυπηρέτησης των επιβατών από τον οδηγό. Τα Credits αυτά μπορούν να χρησιμοποιηθούν ανάλογα με την πολιτική που χρησιμοποιεί κάθε χώρα.

1.3 Συνεισφορά

Η συνεισφορά της πτυχιακής είναι η δημιουργία της εφαρμογής Hermes η οποία έχει τα ακόλουθα χαρακτηριστικά, και προσεγγίζει το πρόβλημα με τον ακόλουθο τρόπο:

1. Είναι βασισμένη σε κοινωνικό δίκτυο όπου ένας χρήστης μπορεί εύκολα να προσθέσει τους φίλους του από την βάση δεδομένων της εφαρμογής. Επίσης πέραν του κοινωνικού δικτύου που έχει η εφαρμογή, ο χρήστης μπορεί εύκολα να φέρει τους φίλους του από το κοινωνικό δίκτυο Facebook, κάτι το οποίο δεν έχουν οι πλείστες εφαρμογές Carpooling. Αυτό το χαρακτηριστικό είναι πολύ σημαντικό για τον χρήστη γιατί μπορεί εύκολα και σε ελάχιστο χρόνο να διαμορφώσει το προφίλ του και να έχει τους φίλους του σε αυτό.

2. Επίσης στην εφαρμογή μας αναπτύξαμε έναν αλγόριθμο ο οποίος χρησιμοποιείται για να γίνει εύρεση δρομολογίων από τους φίλους του χρήστη. Ο αλγόριθμος αυτός ελέγχει κατά πόσο πληρούνται τα κριτήρια που έδωσε ο οδηγός αλλά επίσης και τα κριτήρια που έδωσε ο χρήστης που κάνει την αναζήτηση. Η υλοποίηση αυτού του αλγόριθμου αποτελεί ένα σημαντικό πλεονέκτημα για την εφαρμογή γιατί ο χρήστης μπορεί εύκολα, γρήγορα και αποτελεσματικά να διατρέξει αναζήτηση για ένα δρομολόγιο στο σύνολο όλων των διαθέσιμων δρομολογίων από τους φίλους του και να δει τα προτεινόμενα δρομολόγια που του παρουσιάζονται.

1.4 Τι θα ακολουθήσει

Στο επόμενο κεφάλαιο που ακολουθεί θα μιλήσουμε για κάποιες παρόμοιες εφαρμογές που δημιουργήθηκαν στο παρελθόν, θα συζητήσουμε τις διαφορές που έχουν με την εφαρμογή που αναπτύξαμε, αλλά επίσης θα μιλήσουμε για τον τρόπο με τον οποίο εκτελούν την αναζήτηση δρομολογίων οι άλλες εφαρμογές, σε σύγκριση με την δική μας εφαρμογή, όπου χρησιμοποιούμε έναν αλγόριθμο για να έχουμε πιο αποδοτική και αποτελεσματική αναζήτηση.

Στο τρίτο κεφάλαιο θα αναφερθούμε κυρίως στον αλγόριθμο που χρησιμοποιούμε για να εκτελούμε την αναζήτηση δρομολογίων. Θα αναφέρουμε πιο είναι το πρόβλημα που αντιμετωπίσαμε, και θα αναλύσουμε την πιθανή λύση του προβλήματος που χρησιμοποιήσαμε για την εφαρμογή.

Στο τέταρτο κεφάλαιο που ακολουθεί θα μιλήσουμε για την δική μας εφαρμογή, και θα αναφερθούμε στο τρόπο με τον οποίο προσεγγίσαμε και υλοποιήσαμε το πρόβλημα. Επίσης θα αναλύσουμε την αρχιτεκτονική του συστήματος, όσον αφόρα την εφαρμογή στην κινητή συσκευή αλλά επίσης και την εφαρμογή που είναι εγκατεστημένη στον εξυπηρετητή της εφαρμογής.

Στο κεφάλαιο πέντε θα μιλήσουμε για κάποιες πειραματικές μετρήσεις που κάναμε για να δούμε κυρίως την αποδοτικότητα του αλγόριθμου μας χρησιμοποιώντας κάποια σενάρια. Επίσης θα εξηγήσουμε τα αποτελέσματα των μετρήσεων μας, και θα τα παρουσιάσουμε σε μορφή γραφικών παραστάσεων. Τέλος θα αναφερθούμε στο ερωτηματολόγιο που χρησιμοποιήθηκε για την αξιολόγηση της εφαρμογής, καθώς επίσης και στα αποτελέσματα που εξήχθησαν.

Στο τελευταίο κεφάλαιο θα κάνουμε μια σύνοψη του τι κάναμε, και θα μιλήσουμε για την μελλοντική εργασία που μπορεί να γίνει για ενίσχυση της εφαρμογής.

Τέλος στα παραρτήματα Α και Β μπορούμε να δούμε κάποιες υποδειγματικές οθόνες της εφαρμογής όπου και θα αναλυθούν, αλλά επίσης και κάποια κομμάτια κώδικα που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

Κεφάλαιο 2

Σχετικές εργασίες που μελετήθηκαν

2.1 Σημαντικότητα μελέτης σχετικών εφαρμογών

Για την δημιουργία της εφαρμογής Hermes μελετήθηκαν σχετικές εφαρμογές οι οποίες υλοποιήθηκαν στο παρελθόν. Η μελέτη των σχετικών εφαρμογών βοήθησε στο να εμβαθύνουμε και να κατανοήσουμε περισσότερο το πρόβλημα το οποίο αντιμετωπίζαμε και έπρεπε να λύσουμε. Επίσης μας βοήθησε στο να μελετήσουμε διάφορους τρόπους προσέγγισης του προβλήματος, και καταγράψαμε κύρια χαρακτηριστικά από την κάθε εφαρμογή, με απώτερο σκοπό να τα συμπεριλάβουμε ή να τα αποφύγουμε από την εφαρμογή που αναπτύχθηκε.

Για τον σκοπό αυτό μελετήθηκαν εφαρμογές οι οποίες είναι ανεπτυγμένες σε κινητές συσκευές, όπου παρατηρήσαμε τα σημαντικά τους σημεία, και τον τρόπο με τον οποίο εκμεταλλεύονται οι δυνατότητες της κινητής συσκευής. Επίσης μελετήσαμε εφαρμογές οι οποίες είναι ανεπτυγμένες για την χρήση τους κυρίως από τον ηλεκτρονικό υπολογιστή, όπου μας βοήθησαν να εμβαθύνουμε σε κάποια σημαντικά σημεία της εφαρμογής.

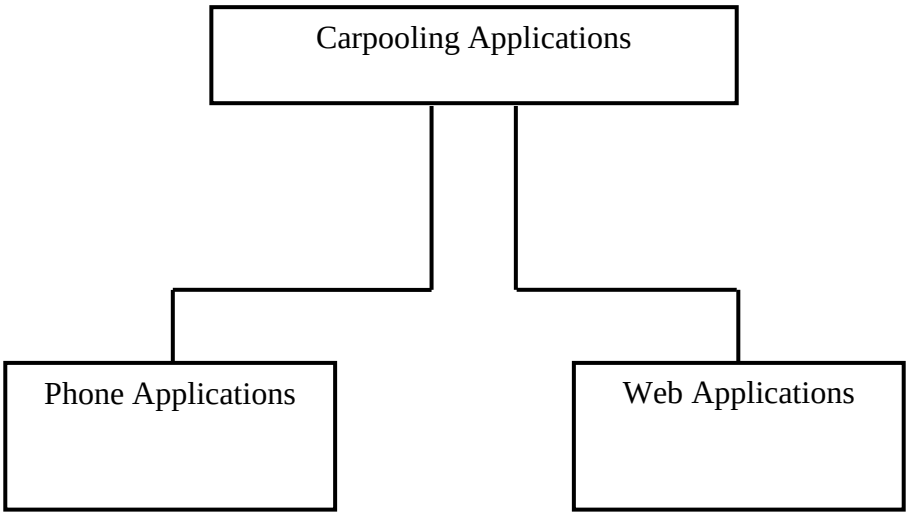
Τέλος από την μελέτη σχετικών εφαρμογών μελετήσαμε τον τρόπο υλοποίησης του αλγορίθμου εύρεσης που χρησιμοποιεί η κάθε εφαρμογή, με απώτερο σκοπό να δημιουργήσουμε ένα δικό μας αλγόριθμο.

2.2 Σύγκριση σχετικών εφαρμογών

Σε αυτό το υποκεφάλαιο θα μιλήσουμε για τις εφαρμογές που μελετήσαμε πριν να υλοποιήσουμε την δική μας, και θα τις συγκρίνουμε μεταξύ τους για να δούμε τις κύριες

διαφορές τους. Επίσης θα δούμε τις δύο σημαντικές ομάδες εφαρμογών που αποτελούν τις Carpooling εφαρμογές που μελετήσαμε.

Ομαδοποίηση Εφαρμογών Carpooling:



Εικόνα 2.2.1 Κατηγοριοποίηση εφαρμογών Carpooling.

Στην πιο πάνω εικόνα παρουσιάζονται οι δύο σημαντικότερες κατηγορίες που διαχωρίζονται οι εφαρμογές Carpooling, σύμφωνα με τα σημαντικότερα χαρακτηριστικά τους.

Όπως παρατηρούμε οι εφαρμογές χωρίζονται σε δύο κατηγορίες:

1. **Phone Applications**

   Οι εφαρμογές οι οποίες είναι ανεπτυγμένες και εξειδικευμένες για κινητές συσκευές και χαρακτηρίζονται κυρίως από την φορητότητα τους. Οι εφαρμογές αυτές μπορούν να χρησιμοποιούν χαρακτηριστικά της κινητής συσκευής όπως για παράδειγμα το Notification Area, GPS, Vibration και άλλα χαρακτηριστικά της συσκευής.

2. **Web Applications**

   Οι εφαρμογές οι οποίες είναι ανεπτυγμένες σε μορφή ιστοσελίδων και μπορούμε να έχουμε πρόσβαση σε αυτές μέσω ενός φυλλομετρητή (browser). Οι εφαρμογές αυτές είναι ανεπτυγμένες κυρίως για την χρήση τους από τον ηλεκτρονικό υπολογιστή, παρόλο που σε κάποιες από αυτές μπορούμε να έχουμε πρόσβαση και από την κινητή συσκευή.

Σημεία Σύγκρισης	Hermes	Car-pooling	Carpool	Carpool Hunter	Facebook4 CarPooling	Ride Remedy	ZimRide
Κοινωνικό δίκτυο της εφαρμογής	✓						✓
Αλγόριθμος εύρεσης δρομολογίων	✓	✓		✓		✓	✓
Χρήση κοινωνικού δικτύου Facebook	✓				✓		✓
Credits	✓			✓			
Δυνατότητα χρήσης χρημάτων	✓	✓	✓	✓		✓	✓
Χρήση χαρτών	✓			✓		✓	✓
Εφαρμογή κινητής συσκευής	✓	✓	✓	✓		✓	
Feedback	✓		✓				✓
Google Network				✓		✓	
Google Talk				✓			
Δυναμική εύρεση τρέχων δρομολογίων	✓						
Δυνατότητα χρήσης ταξί						✓	

Εικόνα 2.2.2 Πίνακας σύγκρισης σχετικών εφαρμογών Carpooling

Επεξήγηση σημαντικών εφαρμογών

Carpool Hunter: είναι μια εφαρμογή ανεπτυγμένη για χρήστες Android λειτουργικού [11]. Χρησιμοποιεί το Google Network όπου ο χρήστης μπορεί να εισέρθει στην εφαρμογή χρησιμοποιώντας το Google Id του. Το κύριο χαρακτηριστικό της εφαρμογής είναι το ότι χρησιμοποιεί τα χρήματα ως αμοιβή του οδηγού. Επίσης οι χρήστες οι οποίοι θέλουν να συμμετέχουν σε ένα δρομολόγιο που προστέθηκε από έναν άλλο χρηστή μπορούν να κάνουν χρηματική προσφορά σε αυτόν και στο τέλος η εφαρμογή αποδέχεται τους χρήστες που έκαναν την μεγαλύτερη προσφορά και απορρίπτει τους υπόλοιπους. Επίσης η εφαρμογή επιτρέπει στους συμμετέχοντες ενός δρομολογίου να μιλήσουν μεταξύ τους χρησιμοποιώντας το Google Talk.

Ride Remedy: Είναι μια εφαρμογή η οποία είναι ανεπτυγμένη για κινητές συσκευές με Apple λειτουργικό [8]. Χρησιμοποιεί το Google Network για πρόσβαση της στην εφαρμογή, όπου οι χρήστες μπορούν να προσθέσουν ένα δρομολόγιο, να αναζητήσουν ένα διαθέσιμο δρομολόγιο από τους άλλους χρήστες αλλά επίσης μπορούν να βρουν κοινά δρομολόγια με άλλους χρήστες και να πάρουν μαζί ταξί. Επίσης η εφαρμογή επιτρέπει την συνομιλία των χρηστών μέσω μηνυμάτων. Ένα σημαντικό στοιχείο της εφαρμογής είναι ο αλγόριθμος αναζήτησης που χρησιμοποιεί για την εύρεση όμοιων δρομολογίων μεταξύ των χρηστών. Ο αλγόριθμος είναι αρκετά ενδιαφέρον και θα χρησιμοποιηθεί σε μετέπειτα στάδιο για σύγκριση του με τον αλγόριθμο αναζήτηση που χρησιμοποιήσαμε.

ZimRide: Είναι μια εφαρμογή η οποία είναι ανεπτυγμένη για χρήση κυρίως από ηλεκτρονικό υπολογιστή [13]. Η εφαρμογή αυτή χρησιμοποιείται από το κοινωνικό δίκτυο Facebook, αλλά επίσης έχει το δικό της κοινωνικό δίκτυο. Στην εφαρμογή μπορείς να επιλέξεις από μια λίστα διαθέσιμων περιοχών την περιοχή σου και ακολούθως να δεις όλα τα διαθέσιμα δρομολόγια που αναφέρονται σε αυτή την περιοχή. Επίσης ο χρήστης μπορεί να κάνει αναζήτηση δρομολογίου, όπου θα παρουσιαστούν τα διαθέσιμα δρομολόγια τα οποία ικανοποιούν το δρομολόγιο, και το χρηματικό ποσό που απαιτεί ο συγκεκριμένος χρήστης για το δρομολόγιο. Επίσης οι χρήστες μπορούν να δουν τα στοιχεία άλλου χρήστη αν επιτρέπεται μέσω του Facebook, ή να δουν κάποια σχόλια που πρόσθεσαν άλλοι χρήστες για αυτόν.

Σχολιασμός χαρακτηριστικών που έχουν χρησιμοποιηθεί:

1. **Κοινωνικό δίκτυο:** Από την σχετική μελέτη των εφαρμογών παρατηρήσαμε πως οι περισσότερες εφαρμογές δεν στηρίζονταν σε δικό τους κοινωνικό δίκτυο, όπου οι χρήστες της εφαρμογής να μπορούν να συμμετέχουν σε δρομολόγια τα οποία να εκτελούνται από τους φίλους τους μόνο. Αυτό είναι ένα σημαντικό κριτήριο για την ανάπτυξη της εφαρμογής μας γιατί η χρήση του κοινωνικού δικτύου για την εφαρμογή θα μπορεί να εμπνέει την εμπιστοσύνη στους χρήστες της εφαρμογής, γιατί οι συμμετέχοντες θα είναι γνωστά άτομα προς αυτούς και δεν θα τους δημιουργούνται οποιαδήποτε διλλήματα.
2. **Αλγόριθμος εύρεσης όμοιων δρομολογίων:** Επίσης παρατηρήσαμε πως ο αλγόριθμος εύρεσης σχετικών δρομολογίων δεν ήταν τόσο αξιόπιστος όσον αφορά τα αποτελέσματα του.

3. **Credits:** Ένα άλλο σημαντικό σημείο που παρατηρήθηκε είναι τα κίνητρα που δίνει η εφαρμογή στους χρήστες της στο να προσθέτουν νέα δρομολόγια. Όπως παρατηρήσαμε οι περισσότερες εφαρμογές έχουν την δυνατότητα χρήσης χρημάτων ως αμοιβή για τον οδηγό και επίσης για να μοιράζονται τα έξοδα του ταξιδίου. Εμείς πέραν από αυτό θέλαμε να δώσουμε στους χρήστες επιπλέον κίνητρα για να χρησιμοποιήσουν την εφαρμογή. Για αυτό και αποφασίσουμε να χρησιμοποιήσουμε τα Credits στην εφαρμογή μας όπου ο χρήστης μπορεί να τα χρησιμοποιήσει αναλόγως με τις δυνατότητες που του δίνει η πολιτεία του. Μπορεί για παράδειγμα να χρησιμοποιηθούν σε χώρους στάθμευσης, στα διόδια ή ακόμα και για μειωμένο κόστος καυσίμων.
4. **Χρήση χαρτών:** Η χρήση χαρτών για απεικόνιση των δρομολογίων είναι ένα σημαντικό στοιχείο γιατί ο χρήστης θα μπορεί να δει τα δρομολόγια λεπτομερώς με την χρήση τους.
5. **Εφαρμογή κινητής συσκευής:** Επίσης θέλαμε ο χρήστης να μπορεί να έχει πρόσβαση στην εφαρμογή μας από οποιοδήποτε σημείο. Για αυτό το λόγο θέσαμε σαν σημαντικό παράγοντα η εφαρμογή μας να αναπτυχθεί σε κινητή συσκευή.
6. **Feedback:** Ένα άλλο σημείο το οποίο προσέξαμε ήταν η αξιοπιστία τον χρηστών και η προστασία των χρηστών από μη αξιόπιστα άτομα. Για το λόγο αυτό προσθέσαμε στην εφαρμογή μας την δυνατότητα χρήσης του Feedback την οποία έχουν και αρκετές άλλες εφαρμογές όπως φαίνεται και στον πίνακα πιο πάνω.
7. **Δυναμική εύρεση τρέχων δρομολογίων:** Τέλος θέσαμε σαν σημαντικό στοιχείο την δυναμική εύρεση τρεχόντων δρομολογίων. Με τον όρο δυναμική εύρεση τρεχόντων δρομολογίων εννοούμε τα δρομολόγια τα οποία διεξάγονται αυτή την στιγμή.

2.3 Σχετικοί αλγόριθμοι εύρεσης δρομολογίων

Μετά από μελέτη των σχετικών εφαρμογών είδαμε πως ο αλγόριθμος εύρεσης σε αρκετές εφαρμογές δεν ήταν τόσο αποτελεσματικός, δηλαδή επέστρεφε αποτελέσματα τα οποία αναφέρονταν σε δρομολόγια τα οποία θα εκτελεστούν σχετικά κοντά στην περιοχή για την οποία διέτρεξε την αναζήτηση ο χρήστης σύμφωνα με το δικό του δρομολόγιο. Οι πλείστες εφαρμογές που μελετήσαμε δεν συνδύαζαν στοιχεία όπως:

1. Ευελιξία δρομολογίου (Route Flexibility), όπου ο χρήστης δηλώνει την μέγιστη απόσταση η οποία είναι αποδεκτή στο να παρακάμψει το δρομολόγιο του για να ικανοποιήσει κάποιον άλλον χρήστη.

2. Ευελιξία χρόνου (Time Flexibility), όπου ο χρήστης μπορεί να δει τα διαθέσιμα δρομολόγια που θα εκτελεστούν από τους φίλους του την ημερομηνία και ώρα που έχει δώσει σε συνδυασμό με την ευελιξία χρόνου όπου μπορεί να χρησιμοποιηθεί είτε θετικά είτε αρνητικά.

Για τους λόγους αυτούς που αναφέραμε πιο πάνω θα πρέπει να αναπτυχθεί ένας αλγόριθμος εύρεσης πιθανών δρομολογίων για τον οποίο δεν θα είναι αναγκαίο τα δρομολόγια να έχουν παρόμοιο σημείο έναρξης ή τερματισμού. Επίσης θα πρέπει ο αλγόριθμος να λαμβάνει υπόψη του στοιχεία όπως για παράδειγμα ευελιξία δρομολόγιου (Route Flexibility), αλλά επίσης και ευελιξίας χρόνου (Time Flexibility) και να επιστρέφει σαν αποτέλεσμα τα ακριβείς δρομολόγια αντιστοίχισης σύμφωνα με τα δεδομένα εισόδου που του δόθηκαν.

Σχετικός Αλγόριθμος εφαρμογής Ride Remedy

Ο αλγόριθμος εύρεσης της εφαρμογής Ride Remedy που μελετήθηκε χρησιμοποιεί παρόμοια χαρακτηριστικά με πιο πάνω για την σύγκριση δύο δρομολογίων [8].

Ορισμοί αλγορίθμου:

1. **Departure Distance:** είναι η μέγιστη απόσταση μεταξύ των σημείων έναρξης των δύο δρομολογίων.
2. **Destination Distance:** είναι η μέγιστη απόσταση μεταξύ των σημείων τερματισμού των δύο δρομολογίων.
3. **Departure Time Difference:** είναι η μέγιστη χρονική διαφορά μεταξύ των δρομολογίων.

Οι είσοδοι του αλγορίθμου αυτού είναι:

1. Το δρομολόγιο που θέλουμε να βρούμε όμοιο του.
2. Ένα σύνολο δρομολογίων από την βάση δεδομένων και όπου η χρονική τους διαφορά είναι μικρότερη ή ίση από το Departure Time Difference.
3. Το Departure Distance.
4. Το Destination Distance.

Ο αλγόριθμος αυτός συγκρίνει το δρομολόγιο για το οποίο θέλουμε να βρούμε παρόμοια του με τα υπόλοιπα δρομολόγια που βρίσκονται στην βάση δεδομένων, με σκοπό να βρει δρομολόγια τα οποία να μπορούν να το εξυπηρετήσουν.

Για να υπάρχει ταύτιση μεταξύ δύο δρομολογίων πρέπει να ικανοποιούνται οι πιο κάτω συνθήκες:

1. Η απόσταση μεταξύ των σημείων έναρξης να είναι μικρότερη ή ίση με το Departure Distance.
2. Η απόσταση μεταξύ των σημείων τερματισμού να είναι μικρότερη ή ίση με το Destination Distance.

Εικόνα 2.3.1 Παράδειγμα υλοποίησης του αλγορίθμου εύρεσης της εφαρμογής Ride Remedy.

Στην πιο πάνω εικόνα βλέπουμε το δρομολόγιο για το οποίο θέλουμε να βρούμε όμοιο του από το σύνολο δρομολογίων που υπάρχει στη βάση δεδομένων. Για να γίνει ταύτιση του δρομολογίου με κάποιο άλλο που έχουμε αποθηκευμένο στη βάση δεδομένων θα πρέπει, τα σημεία έναρξης και τερματισμού του δρομολόγιο να βρίσκονται εντός των δύο πράσινων κύκλων που αφορούν το αντίστοιχο σημείο του συγκεκριμένου δρομολογίου. Οι δύο κύκλοι σχηματίζονται βάσει του Departure Distance και Destination Distance τα οποία χρησιμοποιούνται ως ακτίνα για τον σχηματισμό του, και μπορούν να είναι διαφορετικού μεγέθους μεταξύ τους.

Κεφάλαιο 3

Αλγόριθμος εύρεσης όμοιων δρομολογίων Hermes

3.1 Διατύπωση του προβλήματος

Έστω ότι έχουμε ένα δρομολόγιο από το Α στο Β, όπου Α το αρχικό σημείο και Β το τελικό σημείο του δρομολογίου. Θέλουμε να βρούμε όμοια του από ένα σύνολο αποθηκευμένων δρομολογίων τα οποία αναφέρονται στην ίδια ώρα με το ζητούμενο δρομολόγιο με συνδυασμό του Time Flexibility. Παίρνουμε κάθε ένα δρομολόγιο από το σύνολο αυτό και το συγκρίνουμε με το ζητούμενο δρομολόγιο. Ελέγχουμε αν η απόσταση μεταξύ Α και Α΄ (δύο αρχικών σημείων) είναι μικρότερη ή ίση με το Route Flexibility (ΑΑ΄ <= Route Flexibility) , και η απόσταση μεταξύ Β και Β΄ (δύο τελικών σημείων) είναι μικρότερη ή ίση από το Route Flexibility (ΒΒ΄ <= Route Flexibility). Αν οι δύο πιο πάνω περιορισμοί ικανοποιούνται προσθέτεται το δρομολόγιο στο σύνολο των όμοιων δρομολογίων.

Ορισμοί που θα χρησιμοποιηθούν:

- **Δρομολόγιο:** είναι ένα σύνολο γεωγραφικών σημείων τα οποία αποτελούν την διαδρομή που προκύπτει μεταξύ του αρχικού και του τελικού σημείου.
- **Route Flexibility:** είναι η μέγιστη αποδεκτή απόσταση από τον χρήστη για να παρακάμψει το δρομολόγιο του ούτως ώστε να εξυπηρετήσει έναν άλλο χρήστη και υπολογίζεται σε μέτρα.
- **Time Flexibility:** είναι η ευελιξία στο χρόνο που έδωσε ο χρήστης όπου μπορεί να χρησιμοποιηθεί είτε θετικά είτε αρνητικά.

Το πρόβλημα που αντιμετωπίζουμε σε αυτό το σημείο είναι ότι ο αριθμός των δρομολόγιων που βρίσκονται στην βάση μπορεί να είναι υπερβολικά μεγάλος με αποτέλεσμα η αναζήτηση να χρειάζεται αρκετή ώρα για να ανταποκριθεί. Ο σημαντικότερος παράγοντας για τον οποίο η αναζήτηση μπορεί να χρειάζεται αρκετό χρόνο είναι το ότι για κάθε δρομολόγιο θα πρέπει να γίνεται request στο Google Maps για να πάρουμε το σύνολο των γεωγραφικών σημείων που το αποτελούν.

Για να επιτύχουμε την δημιουργία του αλγορίθμου θα πρέπει με κάποιο τρόπο να παίρνουμε ένα αντιπροσωπευτικό υποσύνολο των δρομολογίων της βάσης δεδομένων βάσει κάποιων κριτήριων και ακολούθως να τα εξετάζουμε αυτά μόνο.

Το υποσύνολο αυτό θα πρέπει να λαμβάνει υπόψη του κριτήρια όπως την ημερομηνία, την ώρα, το Time Flexibility, τα γεωγραφικά σημεία που βρίσκονται τα δρομολόγια, αλλά επίσης και τον αριθμό διαθέσιμων θέσεων του δρομολογίου.

3.2 Προτεινόμενη λύση

Ο αλγόριθμος εύρεσης πιθανών δρομολογίων χωρίζεται σε δύο τμήματα τα οποία εκτελούνται σε διαφορετικές χρονικές στιγμές. Με αυτό τον τρόπο περιορίζουμε την πολυπλοκότητα και τον χρόνο εκτέλεσης του.

Πρώτο τμήμα του αλγορίθμου

Το πρώτο τμήμα του αλγορίθμου εκτελείται στην φάση όπου ένας χρήστης εισάγει ένα νέο δρομολόγιο στη βάση δεδομένων. Σε αυτό το σημείο ορίζουμε μια περιοχή για το συγκεκριμένο δρομολόγιο η οποία καθορίζει την περιοχή στην οποία πρέπει να βρίσκεται ένα δρομολόγιο για να μπορεί να το εξυπηρετήσει. Αν δηλαδή το αρχικό και τελικό σημείο του δρομολογίου βρίσκεται μέσα σε αυτή την περιοχή τότε μπορεί να εξυπηρετηθεί από το συγκεκριμένο δρομολόγιο αν όχι τότε δεν ικανοποιεί τα κριτήρια και δεν θα συμπεριλαμβάνεται στο σύνολο ομοίων δρομολογίων.

Να αναφέρουμε πως το κάθε δρομολόγιο ορίζεται από διαφορετική περιοχή, επίσης η περιοχή εξαρτάται από το Route Flexibility που δίνει ο χρήστης κατά την προσθήκη του δρομολογίου.

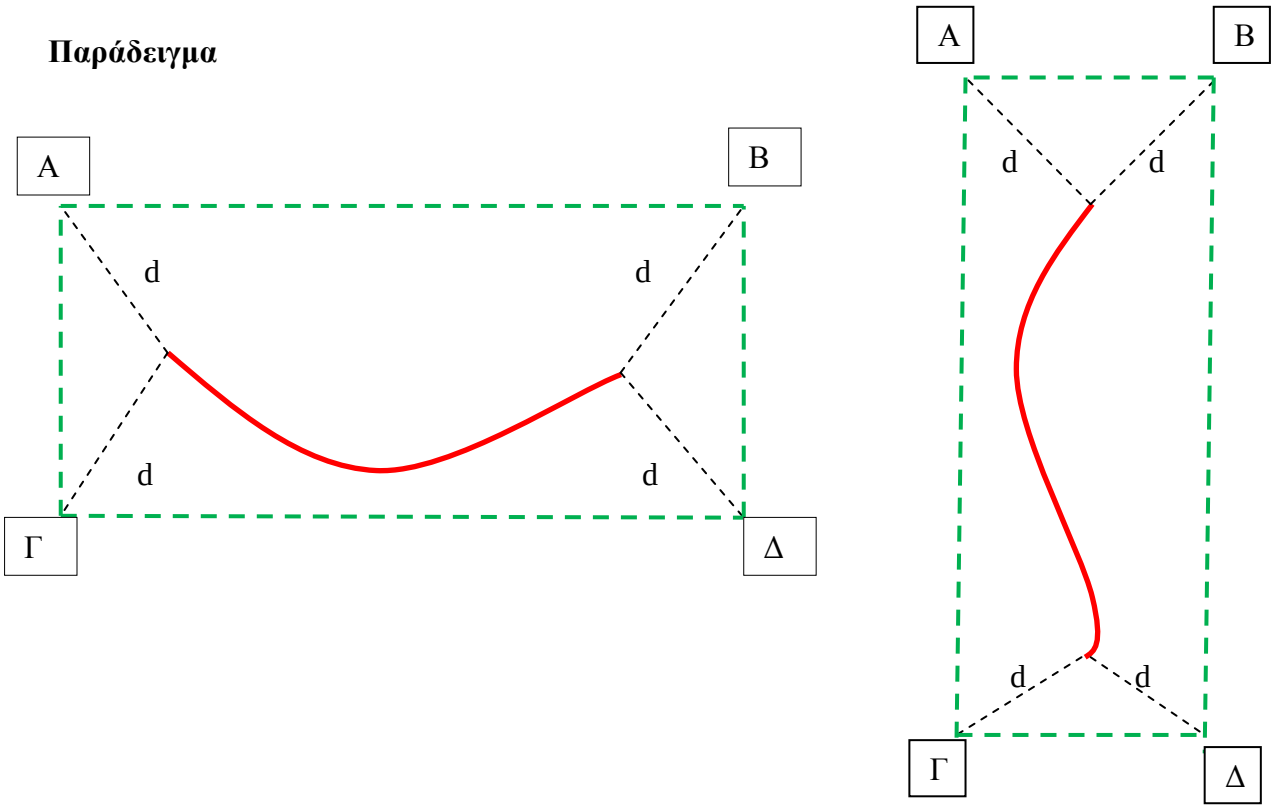
Είσοδοι αλγορίθμου πρώτου τμήματος:

1. Το δρομολόγιο
2. Το Route Flexibility

Για να βρούμε αυτή την περιοχή που ορίζει ένα δρομολόγιο βρίσκουμε πρώτα την απόσταση του δρομολογίου σε ευθεία και σε πραγματική απόσταση. Ακολούθως βρίσκουμε την κλήση του δρομολογίου χρησιμοποιώντας το σημείο έναρξης και το σημείο τερματισμού του δρομολογίου. Επίσης γνωρίζουμε το Route Flexibility του συγκεκριμένου δρομολογίου που έδωσε ο χρήστης.

Με την χρήση αυτών των δεδομένων ορίζουμε 4 γεωγραφικά σημεία τα οποία αποτελούν τα 4 άκρα της περιοχής που θέλουμε να ορίσουμε για το δρομολόγιο. Τώρα θα δώσουμε ένα παράδειγμα του τι κάνει το πρώτο κομμάτι του αλγορίθμου για καλύτερη κατανόηση του.

Παράδειγμα



Στο πιο πάνω παράδειγμα βλέπουμε δυο διαφορετικά δρομολόγια τα οποία προστέθηκαν από τους χρήστες και συμβολίζονται με την κόκκινη γραμμή. Εμείς θέλουμε να βρούμε τα τέσσερα γεωγραφικά σημεία (Α, Β, Γ, Δ) όπως φαίνονται πιο πάνω. Τα 4 αυτά σημεία έχουν απόσταση d από το τελικό και αρχικό σημείο αναλόγως.

Για να βρούμε την απόσταση d χρησιμοποιούμε τα ακόλουθα:

1. DistanceReal: Η πραγματική απόσταση του δρομολογίου
2. DistanceStr: Η απόσταση του δρομολόγιου σε ευθεία γραμμή
3. RouteFlexibility: Η μέγιστη απόσταση σε μέτρα η οποία είναι αποδεχτεί από τον χρήστη να βγει από το δρομολόγιο του για να εξυπηρετήσει έναν άλλο χρήστη.

DistanceReal: το παίρνουμε από Request που κάνουμε στο Google Maps.

DistanceStr: Υπολογίζεται με την μέθοδο Distance [14]. Οι μέθοδος αυτή έχει σαν δεδομένα εισόδου το γεωγραφικός πλάτος και γεωγραφικό μήκος των δύο γεωγραφικών σημείων που θέλουμε να βρούμε την απόσταση τους.

```
DistanceStr (double latitude1, double longitude1, double latitude2, double longitude2) {
        dLat = Math.toRadians(latitude2 – latitude1)
        dLon = Math.toRadians(longitude2 - longitude1)

        a = Math.sin(dLat / 2) * Math.sin(dLat / 2)+ Math.cos(Math.toRadians(latitude1))*
        Math.cos(Math.toRadians(latitude2)) * Math.sin(dLon / 2)* Math.sin(dLon / 2)

        c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a))
        DistanceStr = 6371 *  c
}
```

Route Flexibility: δίνεται από τον χρήστη που πρόσθεσε το δρομολόγιο.

Η απόσταση d (DistanceRange) υπολογίζεται από την πιο κάτω φόρμουλα:

$$Distance Range = (Distance Real - Distance Str + Route Flexibility) \times 1000$$

Όπως βλέπουμε και πιο πάνω στο παράδειγμα με τα δύο διαφορετικά δρομολόγια τα τέσσερα αυτά σημεία (Α, Β, Γ, Δ) είναι ανάλογα από την κλίση του δρομολογίου την οποία υπολογίζουμε χρησιμοποιώντας το αρχικό και το τελικό σημείο του δρομολογίου. Για να βρούμε την κλίση του δρομολογίου χρησιμοποιούμε την συνάρτηση Bearing [14]. Η συνάρτηση Bearing έχει σαν είσοδο το γεωγραφικός πλάτος και γεωγραφικό μήκος των δύο γεωγραφικών σημείων που θέλουμε να βρούμε την κλίση τους.

Επίσης από την πιο πάνω φόρμουλα παρατηρούμε πως όσο αυξάνεται η κυρτότητα του δρόμου τόσο αυξάνεται η διαφορά της πραγματικής απόστασης και της απόστασης σε

ευθεία, όπου στην συνέχεια θα αυξηθεί και το Distance Range και σαν αποτέλεσμα το συγκεκριμένο δρομολόγιο θα ορίζεται από μεγαλύτερη περιοχή.

```
Bearing (double latitude1, double longitude1, double latitude2, double longitude2) {
    lat1 = Math.toRadians(latitude1)
    long1 = Math.toRadians(longitude1)
    lat2 = Math.toRadians(latitude2)
    long2 = Math.toRadians(longitude2)
    delta_long = long2 - long1
    a = Math.sin(delta_long) * Math.cos(lat2)

    b = Math.cos(lat1) * Math.sin(lat2)  - Math.sin(lat1)  * Math.cos(lat2) *
        Math.cos(delta_long)

    bearing = Math.toDegrees(Math.atan2(a, b))
    BearingRoad = (bearing + 360) % 360
}
```

Τέλος θέλουμε να βρούμε τα 4 γεωγραφικά σημεία τα οποία η απόσταση τους από το αρχικό και το τελικό σημείο είναι ίση με το Distance Range και έχουν κλίση ίση με το Bearing Road. Για να το επιτύχουμε αυτό χρησιμοποιούμε το Destination point given distance and bearing from start point [14]. Για να βρούμε το νέο γεωγραφικό σημείο χρειαζόμαστε το γεωγραφικό πλάτος και μήκος του αρχικού σημείου, την κλίση και την απόσταση που θέλουμε μεταξύ των δύο σημείων.

```
NewPoint (double latitude, double longitude, double BearingRoad, Double DistanceRange) {
        RADIUS_EARTH_METERS = 6378137
        DEG2RAD = Math.PI / 180
        distance = DistanceRange / RADIUS_EARTH_METERS
        bearing = DEG2RAD * BearingRoad
        lat1 = DEG2RAD * latitude
        lon1 = DEG2RAD * longitude

        lat2 = Math.asin(Math.sin(lat1)  * Math.cos(distance) + Math.cos(lat1) *
        Math.sin(distance) * Math.cos(brng))
        lon2 = lon1 + Math.atan2(Math.sin(bearing)  * Math.sin(distance)  * Math.cos(lat1),
        Math.cos(distance) - Math.sin(lat1) * Math.sin(lat2))

         lat2deg = lat2 / DEG2RAD
         lon2deg = lon2 / DEG2RAD
         Return new Point (lat2deg) ,lon2deg)
}
```

Όταν βρούμε τα τέσσερα αυτά γεωγραφικά σημεία και τα ενώσουμε μπορούμε να δούμε την περιοχή που ορίζει το συγκεκριμένο δρομολόγιο και για την οποία μπορούν να εξυπηρετηθούν αλλά δρομολόγια τα οποία το αρχικό αλλά και το τελικό τους σημείο βρίσκονται εντός αυτής της περιοχής.

Εικόνα 3.2.1 Παράδειγμα εκτέλεσης πρώτου τμήματος του αλγορίθμου.

Στην πιο πάνω εικόνα βλέπουμε την περιοχή που χαρακτηρίζει το συγκεκριμένο δρομολόγιο με πράσινη γραμμή μετά από εκτέλεση του πρώτου τμήματος του αλγορίθμου από την κινητή συσκευή. Ακολούθως για κάθε δρομολόγιο αποθηκεύουμε τα τέσσερα αυτά γεωγραφικά σημεία στην βάση δεδομένων όπου και θα χρησιμοποιηθούν στο δεύτερο τμήμα του αλγόριθμου (αποθηκεύουμε δηλαδή το γεωγραφικό πλάτος και το γεωγραφικό μήκος για το κάθε ένα από τα σημεία).

Επίσης για τον ορισμό της περιοχής ενός δρομολογίου θα μπορούσαμε να χρησιμοποιήσουμε κύκλο ή έλλειψη. Ο λόγος για τον οποίον χρησιμοποιήσαμε τετράγωνο είναι γιατί θέλαμε η περιοχή του κάθε δρομολογίου να είναι όσο το δυνατό πιο μικρή, αλλά επίσης και να καλύπτει όλους τους πιθανούς συνδυασμούς. Επίσης η χρήση του τετραγώνου μας επιτρέπει

να ελέγχουμε απευθείας από την βάση δεδομένων αν ένα γεωγραφικό σημείο είναι εντός ή εκτός της περιοχής, χωρίς να χρειάζεται να γίνει επίλυση κάποιας μαθηματικής εξίσωσης.

Δεύτερο τμήμα του αλγορίθμου

Το δεύτερο τμήμα του αλγορίθμου εκτελείται στην φάση της αναζήτησης, όπου ένας χρήστης εισάγει τα ακόλουθα δεδομένα:

1. Το δρομολόγιο για το οποίο θέλει να εκτελέσει την αναζήτηση. Το δρομολόγιο αυτό αποτελείται από δύο γεωγραφικά σημεία, το σημείο έναρξης και το σημείο τερματισμού του δρομολογίου.
2. Ημερομηνία και ώρα εκτέλεσης του δρομολογίου.
3. Το Time Flexibility.

Το δεύτερο τμήμα του αλγορίθμου εκτελείται στον εξυπηρετητή γιατί είναι πολύ πιο γρήγορος από την κινητή συσκευή αλλά επίσης και γιατί έχει πολύ καλύτερη σύνδεση με το διαδίκτυο. Αυτό είναι ένας πολύ σημαντικός παράγοντας για τον αλγόριθμο γιατί κατά την εκτέλεση του γίνονται Request στο Google Maps για να πάρουμε το ακριβές δρομολόγιο ως ένα σύνολο γεωγραφικών σημείων για τα οποία θα διατρέξουμε τον έλεγχο.

Βήματα εκτέλεσης δεύτερου τμήματος του αλγορίθμου:

1. Από την βάση δεδομένων παίρνουμε όλα τα διαθέσιμα δρομολόγια τα οποία θα εκτελεστούν από τους φίλους του χρήστη την συγκεκριμένη ημερομηνία και ώρα που έδωσε ο χρήστης σε συνδυασμό με το Time Flexibility. Επίσης ελέγχουμε αν το δρομολόγιο, για το οποίο θέλουμε να βρούμε άλλα δρομολόγια τα οποία το ικανοποιούν, το αρχικό και το τελικό του σημείο να βρίσκονται εντός της γεωγραφικής περιοχής που δημιουργείται από τα τεσσάρα γεωγραφικά σημεία που βρήκαμε και αποθηκεύσαμε στη βάση στο πρώτο τμήμα του αλγορίθμου όπως εξηγήσαμε πιο πάνω.
2. Ακολούθως παίρνουμε δρομολόγια που πήραμε από την εκτέλεση του Query στην βάση και ελέγχουμε το κάθε ένα ξεχωριστά. Στη συνέχεια κάνουμε Request στο Google Maps για να μας δώσει το καλύτερο δρόμο για το συγκεκριμένο δρομολόγιο καθώς επίσης και το σύνολο όλων των γεωγραφικών σημείων τα οποία αποτελούν την ακριβή διαδρομή του.
3. Στη συνέχεια παίρνουμε τα γεωγραφικά σημεία του δρομολογίου και ελέγχουμε αν η απόσταση τους σε ευθεία γραμμή μεταξύ του γεωγραφικού σημείου και του σημείου

έναρξης του δευτέρου δρομολογίου είναι μικρότερη από το Route Flexibility που έδωσε ο χρήστης.

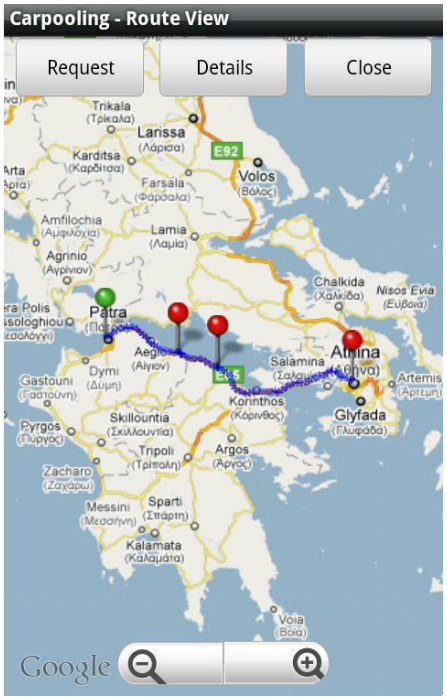
4. Αν αυτή η απόσταση είναι μικρότερη από το Route Flexibility κάνουμε Request στο Google Maps και παίρνουμε την πραγματική απόσταση μεταξύ του γεωγραφικού σημείου και του δεύτερου δρομολόγιου και την συγκρίνουμε με το Route Flexibility αν είναι μικρότερη σταματούμε την εκτέλεση του βήματος (3).

Σημείωση:

Αν περάσουμε από όλα τα γεωγραφικά σημεία του δρομολογίου και δεν βρούμε κάποιο γεωγραφικό σημείο το οποίο να ικανοποιεί τον περιορισμό της απόσταση, τότε το συγκεκριμένο δρομολόγιο δεν θα είναι στα πιθανά δρομολόγια και θα αποκλειστεί.

5. Ακολούθως εκτελούμε το βήμα (3) και (4) για να δούμε αν ικανοποιείται ο περιορισμός απόστασης και το σημείο τερματισμού του δεύτερου δρομολογίου.
6. Αν ικανοποιείται ο περιορισμός απόσταση για το σημείο έναρξης αλλά επίσης και το σημείο τερματισμού του δευτέρου δρομολογίου τότε το συγκεκριμένο δρομολόγιο το οποίο το πήραμε από την βάση δεδομένων και εξετάστηκε είναι ένα πιθανό δρομολόγιο και θα παρουσιαστεί στον χρήστη.

Να αναφέρουμε ότι τα βήματα (3) μέχρι (4) θα εκτελεστούν για όλα τα δρομολόγια που πήραμε από την βάση δεδομένων μετά την εκτέλεση του Query.



Εικόνα 3.2.2 Παράδειγμα εκτέλεσης δευτέρου τμήματος του αλγορίθμου.

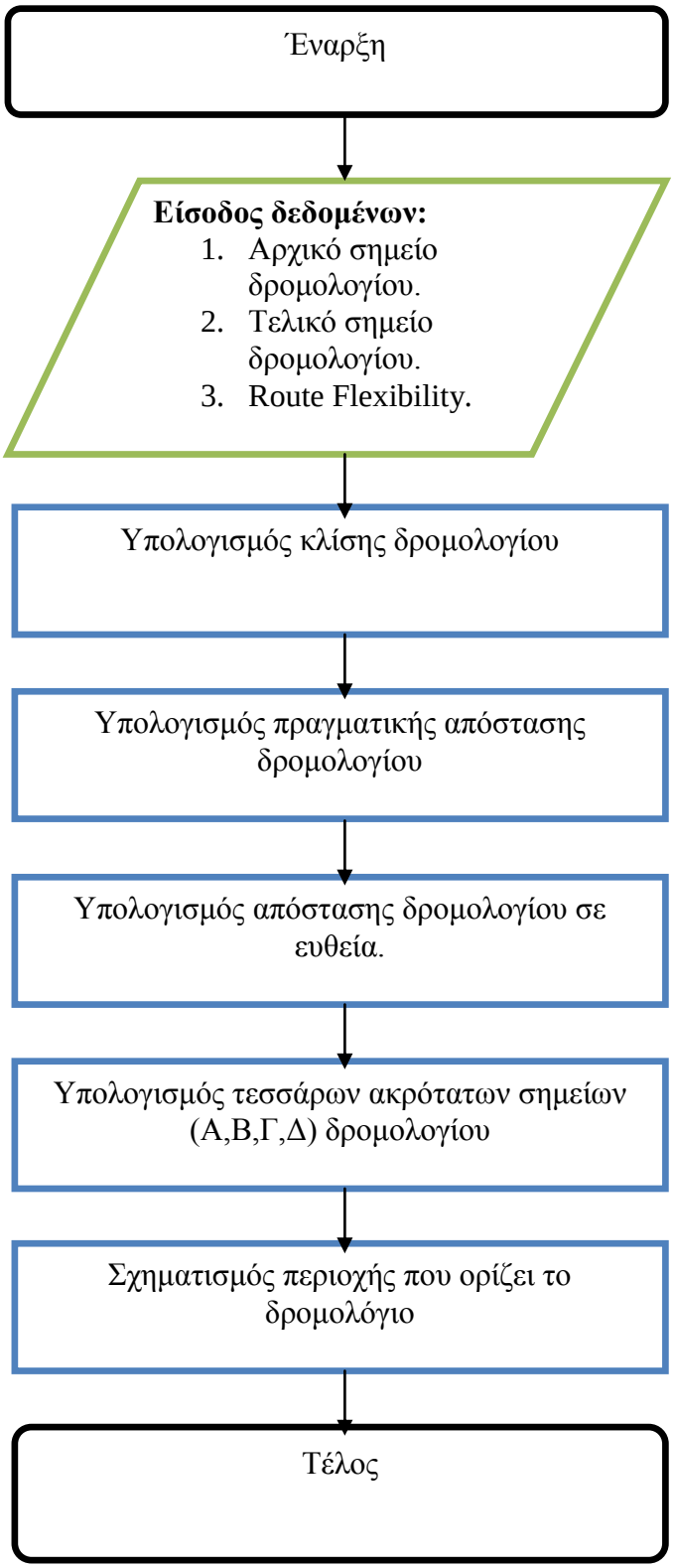
Στην πιο πάνω εικόνα βλέπουμε ένα από τα αποτελέσματα εκτέλεσης του δεύτερου τμήματος του αλγορίθμου εύρεσης δρομολογίων. Για την εκτέλεση αυτού του σεναρίου είχαμε στην βάση δεδομένων αποθηκευμένο το δρομολόγιο Αθήνα – Πάτρα και εκτελέσαμε την αναζήτηση εύρεση πιθανών δρομολογίων για το ενδιάμεσο δρομολόγιο, όπως φαίνεται στην πιο πάνω εικόνα.

Ο αλγόριθμος εύρεσης πιθανών δρομολογίων επέστρεψε το δρομολόγιο Αθήνα – Πάτρα παρόλο που έχουν εντελώς διαφορετικά σημεία έναρξης και τερματισμού. Ο λόγος για τον οποίο θεωρεί πως είναι πιθανό δρομολόγιο είναι γιατί το δρομολόγιο που είχαμε αποθηκευμένο στη βάση δεδομένων περνά κοντά από το αρχικό και τελικό σημείο του δρομολόγιου που εκτελέσαμε την αναζήτηση χωρίς να ξεπερνά το Route Flexibility.

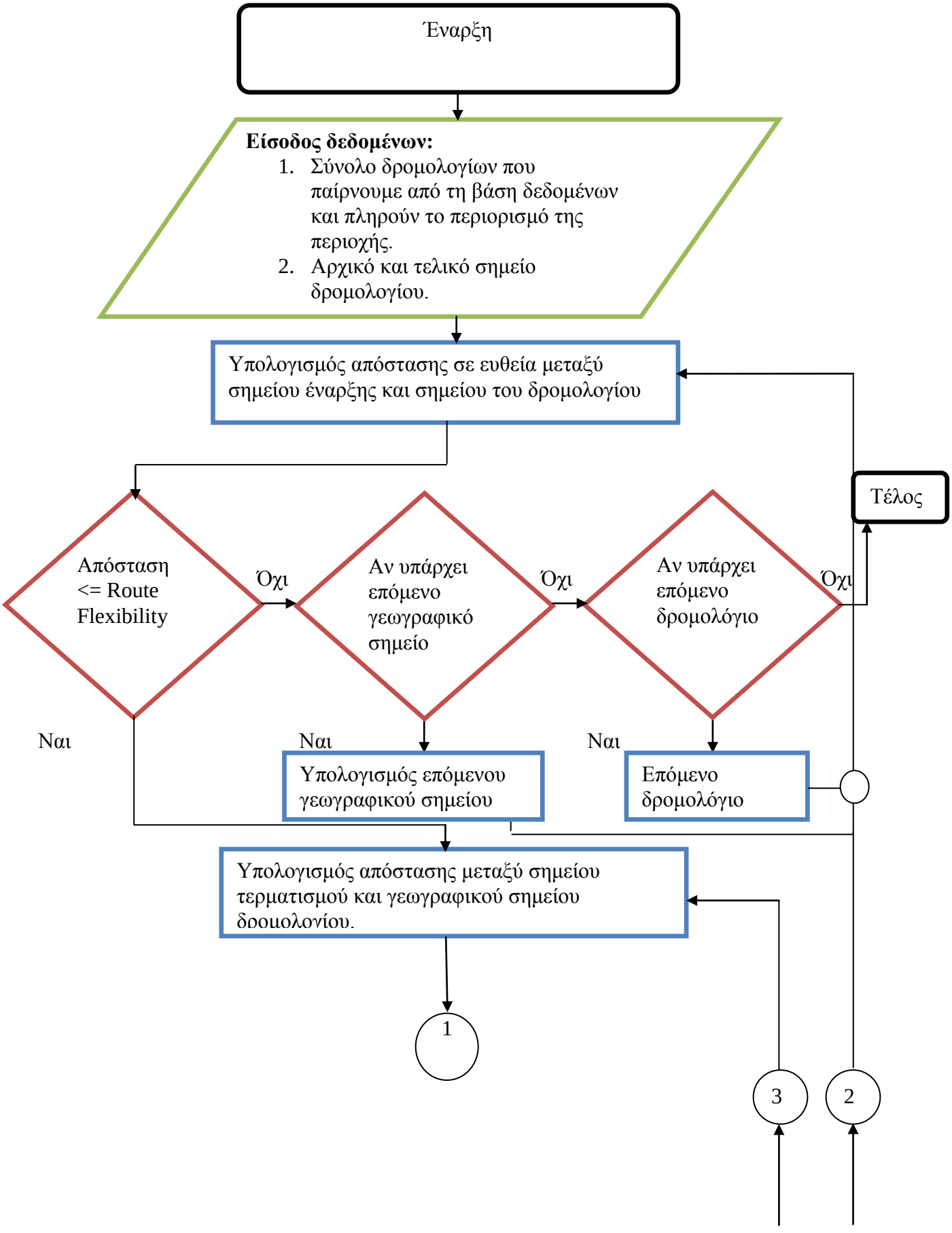
3.3 Διάγραμμα ροής δεδομένων αλγορίθμου

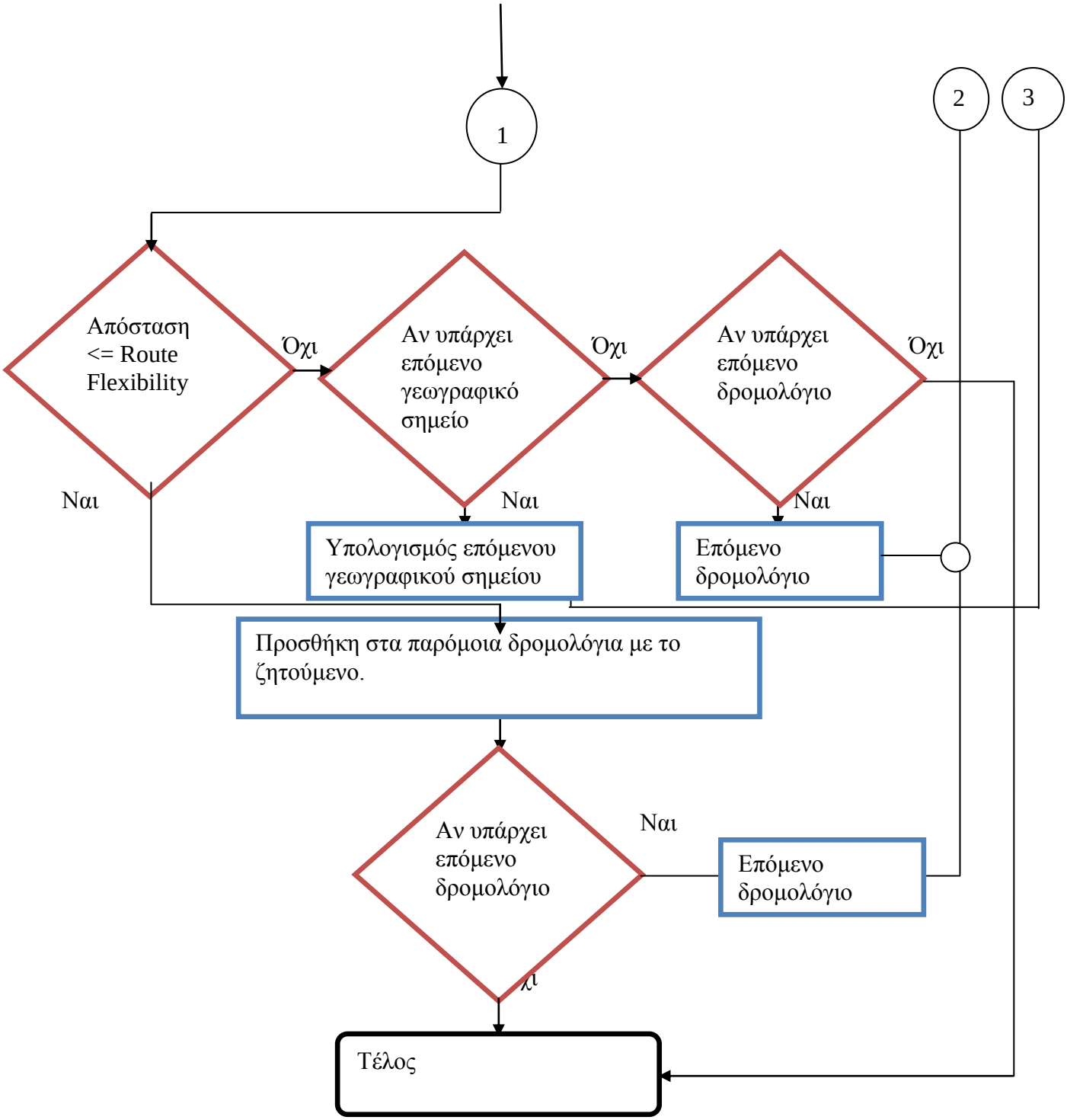
Σε αυτό το υποκεφάλαιο θα δούμε το διάγραμμα ροής δεδομένων του αλγορίθμου για το πρώτο και για το δεύτερο του τμήμα.

3.3.1 Διάγραμμα ροής δεδομένων πρώτου τμήματος.



3.3.2 Διάγραμμα ροής δεδομένων δευτέρου τμήματος.




1
2
3
Απόσταση <= Route Flexibility
Όχι
Αν υπάρχει επόμενο γεωγραφικό σημείο
Όχι
Αν υπάρχει επόμενο δρομολόγιο
Όχι
Ναι
Ναι
Ναι
Υπολογισμός επόμενου γεωγραφικού σημείου
Επόμενο δρομολόγιο
Προσθήκη στα παρόμοια δρομολόγια με το ζητούμενο.
Αν υπάρχει επόμενο δρομολόγιο
Ναι
Επόμενο δρομολόγιο
χι
Τέλος

Κεφάλαιο 4

Περιγραφή εφαρμογής

4.1 Περιγραφή εφαρμογής

Η εφαρμογή που αναπτύχθηκε διασπάται σε 2 τμήματα:

1. Το τμήμα της εφαρμογής το οποίο είναι εγκατεστημένο στην κινητή συσκευή και συγκεκριμένα σε Android πλατφόρμα.
2. Το τμήμα της εφαρμογής το οποίο είναι εγκατεστημένο στον εξυπηρετητή (Server).

Πρώτο τμήμα: Εφαρμογή κινητής συσκευής

Είναι το τμήμα της εφαρμογής που είναι εγκατεστημένο στην κινητή συσκευή. Σε αυτό το τμήμα ο χρήστης έχει άμεση αλληλεπίδραση με το σύστημα, και μπορεί να εκτελεί τις διάφορες διαδικασίες που του προσφέρονται και να λαμβάνει τα ανάλογα αποτελέσματα τους. Ο χρήστης μπορεί να εισέλθει στη εφαρμογή χρησιμοποιώντας το User Name και το Password του λογαριασμού του. Ακολούθως μπορεί να δει τα διαθέσιμα δρομολόγια που εκτελούνται από τους φίλους του, να προσθέσει δρομολόγια και να επικοινωνήσει με άλλους χρήστες μέσο μηνυμάτων. Η επικοινωνία μεταξύ της κινητής συσκευής και του εξυπηρετητή επιτυγχάνεται με τη χρήση του πρωτοκόλλου TCP/IP. Επίσης τα δεδομένα που στέλνονται από την κινητή συσκευή στον εξυπηρετητή στέλνονται αμέσως χωρίς χρονική καθυστέρηση. Ο λόγος για τον οποίο θέλουμε να στέλλονται αμέσως είναι κυρίως γιατί πρέπει να κρατάμε ενημερωμένο το σύστημα όσον αφορά τα δρομολόγια, όπου κάποιοι χρήστες δεσμεύουν θέσεις. Επίσης πρέπει να γνωρίζει το σύστημα αν είναι διαθέσιμα κάποια δρομολόγια για άλλους χρήστες. Επίσης ο χρήστης αλληλεπιδρά άμεσα με το σύστημα, και εκτελεί κάποια

αιτήματα όπου περιμένει να λάβει τα αποτελέσματα τους αμέσως χωρίς την ύπαρξη χρονικής καθυστέρησης. Τα δεδομένα που στέλλονται από την κινητή συσκευή στον εξυπηρετητή είναι μικρού μεγέθους. Για αυτό το λόγο δεν χρειάστηκε να χρησιμοποιηθεί κάποιος αλγόριθμος συμπίεσης των δεδομένων για γρηγορότερη αποστολή τους.

Εφαρμογή εξυπηρετητή

Το τμήμα της εφαρμογής που είναι εγκατεστημένο στον εξυπηρετητή είναι το τμήμα όπου εκτελούνται οι διάφορες λειτουργίες που επιλέγει ο χρήστης και του αποστέλλονται τα αποτελέσματα. Σε αυτό το τμήμα της εφαρμογής αποθηκεύονται και ανακαλούνται τα δεδομένα των χρηστών με τη χρήση βάσης δεδομένων.

Η επικοινωνία που υπάρχει μεταξύ κινητής συσκευής και εξυπηρετητή είναι υλοποιημένη με την χρήση του πρωτοκόλλου TCP/IP (Transmission Control Protocol). Ο λόγος για τον οποίο χρησιμοποιήσαμε αυτό το πρωτόκολλο είναι ότι υπάρχει ισχυρή και άμεση αλληλεπίδραση μεταξύ client – server και θέλουμε τα δεδομένα να στέλνονται άμεσα χωρίς χρονική καθυστέρηση στα δύο τμήματα. Το πρωτόκολλο TCP εγγυάται την αξιόπιστη αποστολή και λήψη δεδομένων, χωρίς να χάνονται πακέτα, όπου και παραδίδονται στην σωστή σειρά. Σε αντίθεση με το πρωτόκολλο επικοινωνίας UDP/IP το οποίο αποστέλλει τα δεδομένα πιο γρήγορα από το πρωτόκολλο TCP/IP, μπορούν να χαθούν κάποια πακέτα χωρίς να ξανασταλθούν. Για το λόγο αυτό το πρωτόκολλο TCP/IP είναι το καταλληλότερο πρωτόκολλο για την εφαρμογή που αναπτύχθηκε.

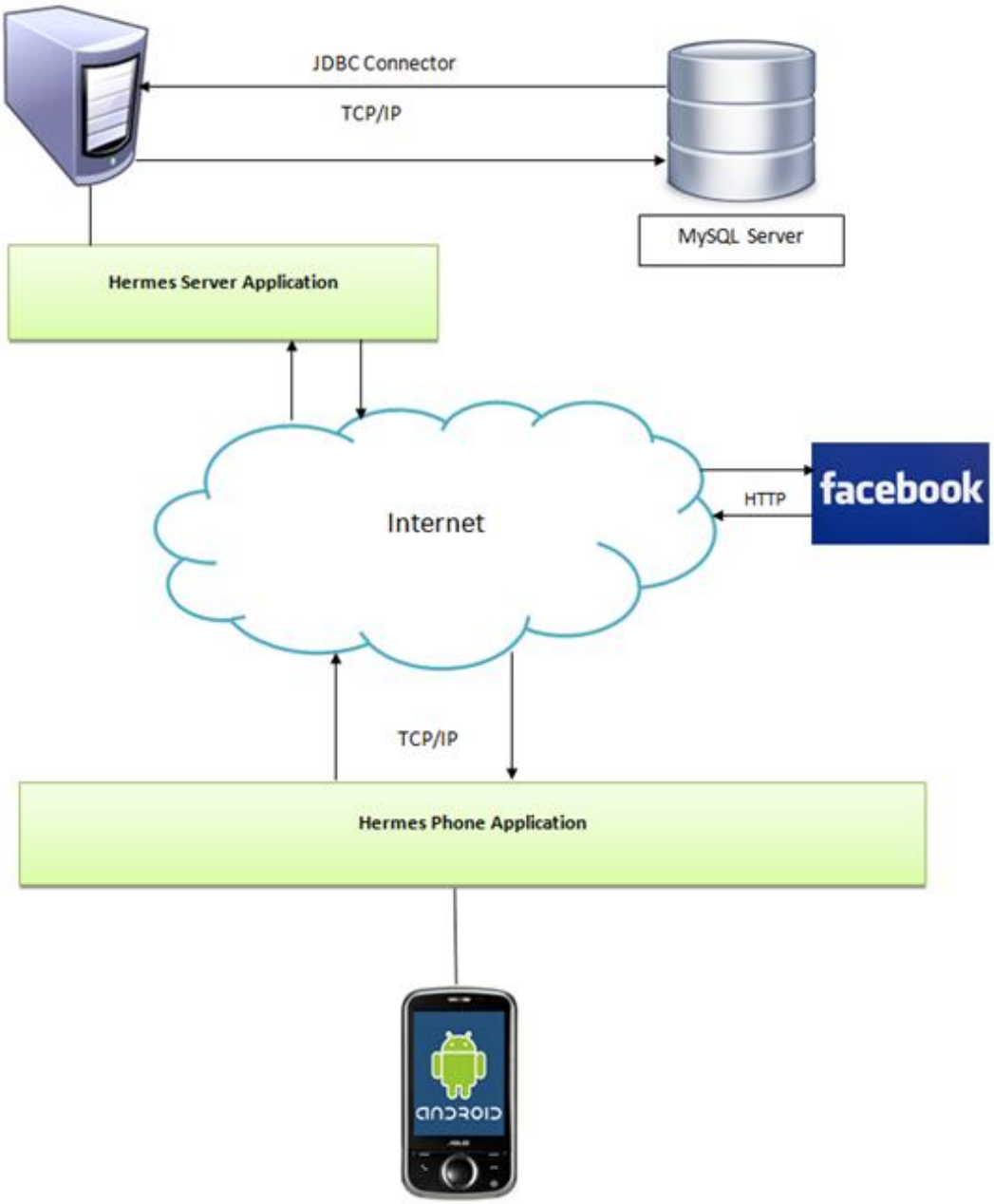
Επίσης το τμήμα της εφαρμογής που βρίσκεται εγκατεστημένο στον εξυπηρετητή πρέπει να είναι σε θέση να εκτελεί παράλληλα πολλές διαδικασίες, αιτήσεις από τους χρήστες και να αποστέλλει τα αποτελέσματα τους στους σωστούς τελικούς χρήστες. Για αυτόν τον λόγο σε αυτό το τμήμα χρησιμοποιούνται νήματα για να έχουμε παράλληλα εκτελούμενα έργα και να μπορούν να εναλλάσσονται γρήγορα και να μοιράζονται τους πόρους του συστήματος όπως την μνήμη, υπολογιστική ισχύ κτλ. Όταν έρχονται αιτήματα από διάφορες κινητές συσκευές δημιουργούνται νέα νήματα, όπου το κάθε ένα από αυτά αντιπροσωπεύει ένα αίτημα, και εξυπηρετούνται με τη χρήση ουρών FIFO.

Για την αποθήκευση και ανάκλαση δεδομένων των χρηστών χρησιμοποιήσαμε μια βάση δεδομένων, όπου θα μας παρείχε αυτή την δυνατότητα, αλλά επίσης και την δυνατότητα εκτέλεσης υποερωτήσεων (Query) σε αυτή όπου ο χρόνος απόκρισης της είναι ελάχιστος.

Επομένως στην βάση δεδομένων που χρησιμοποιήσαμε αποθηκεύονται διάφορα στοιχεία, όπως τα δεδομένα του χρήστη, τα δρομολόγια, τα διάφορα μηνύματα που ανταλλάσσονται μεταξύ των χρηστών, τα αιτήματα των χρηστών αλλά επίσης υλοποιεί και το κοινωνικό δίκτυο στο οποίο στηρίζεται η εφαρμογή.

Για την κατασκευή της βάσης δεδομένων χρησιμοποιήσαμε MySQL η οποία είναι ανοικτού κώδικα και διατίθεται δωρεάν για χρήση.

4.2 Γενικότερη αρχιτεκτονική εφαρμογής



Εικόνα 4.2.1 Γενικότερη αρχιτεκτονική εφαρμογής.

Στο πιο πάνω σχήμα παρουσιάζεται η γενική αρχιτεκτονική του συστήματος, η οποία χωρίζεται σε δύο τμήματα.

Πρώτο τμήμα (Hermes Phone Application):
Όπως αναφέραμε και προηγουμένως αυτό το τμήμα είναι εγκατεστημένο στην κινητή συσκευή και αλληλεπιδρά άμεσα με τον χρήστη. Όταν ο χρήστης εκτελέσει μια διαδικασία στην εφαρμογή της κινητής συσκευής δημιουργείται μια νέα σύνδεση (TCP/IP Connection) με τον εξυπηρετητή χρησιμοποιώντας το IP του εξυπηρετητή και το port number στο οποίο ακούει η εφαρμογή μας, όπου και στέλλονται τα κατάλληλα δεδομένα στον εξυπηρετητή.

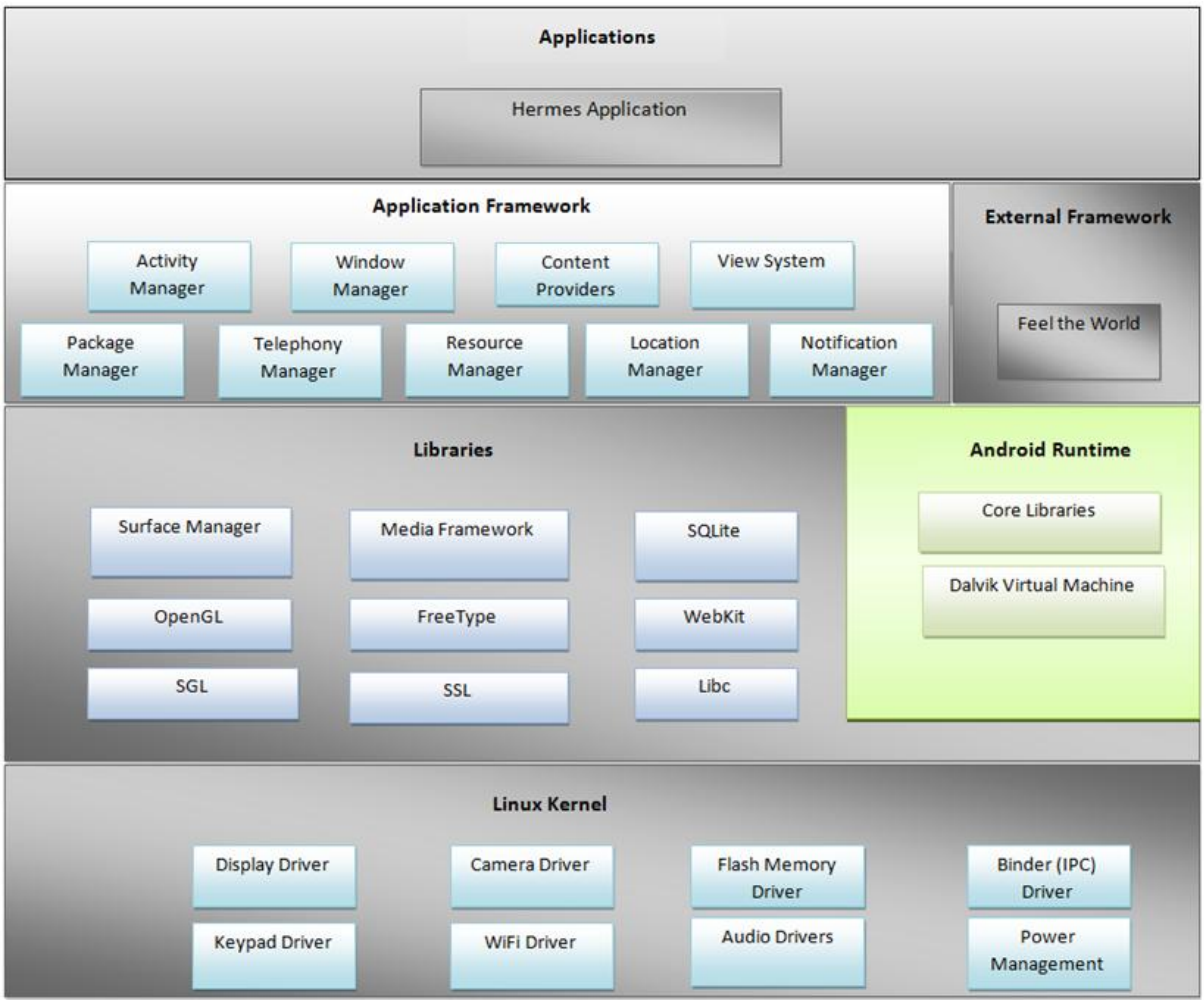
Δεύτερο τμήμα (Hermes Server Application):
Στο δεύτερο τμήμα του συστήματος λαμβάνονται τα δεδομένα από την κινητή συσκευή και εκτελείται η κατάλληλη διαδικασία χρησιμοποιώντας τις κατάλληλες υποερωτήσεις από την βάση δεδομένων. Τέλος συλλέγονται τα αποτελέσματα της διαδικασίας, παράγεται το απαιτούμενο αποτέλεσμα και αποστέλλονται στο IP και το port number της κινητής συσκευής. Τέλος κλείνεται η σύνδεση TCP/IP Connection και παρουσιάζονται τα αποτελέσματα στον χρήστη.

Επίσης υπάρχει επικοινωνία της εφαρμογής με το κοινωνικό δίκτυο Facebook, όπου ο χρήστης μπορεί να προσθέσει τους φίλους του από το κοινωνικό δίκτυο στους φίλους του στην εφαρμογή αν χρησιμοποιούν την εφαρμογή και έχουν λογαριασμό. Για να προστεθούν φίλοι κάποιου χρήστη στο λογαριασμό του από το κοινωνικό δίκτυο Facebook θα πρέπει, να εισέλθουν στο κοινωνικό δίκτυο από την εφαρμογή τουλάχιστο μια φορά. Έτσι θα καταγραφεί το Social Id τους που χρησιμοποιείται από το κοινωνικό δίκτυο, και είναι πρότυπο για τον κάθε χρηστή στην εφαρμογή, και ακολούθως θα μπορεί να προσθέσει τον συγκεκριμένο χρήστη.

4.3 Ανάλυση αρχιτεκτονικής

Σε αυτό το υποκεφάλαιο θα αναλύσουμε την αρχιτεκτονική της εφαρμογής για τα δύο τμήματα που την αποτελούν.

4.3.1 Hermes Phone Application

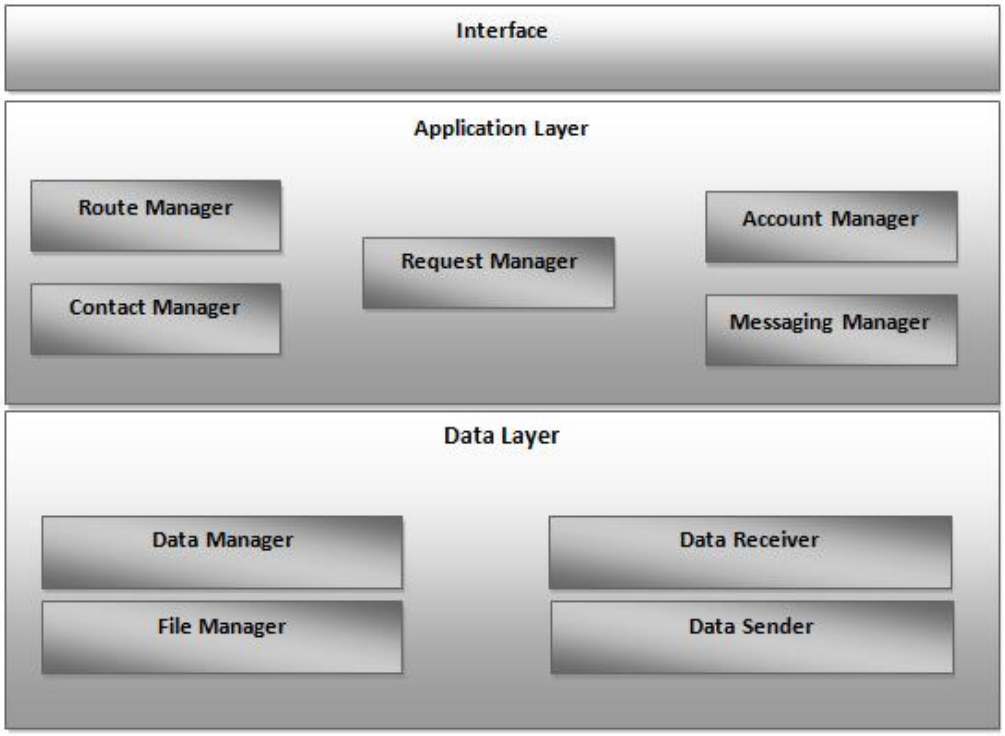


Εικόνα 4.3.1 Αρχιτεκτονική εφαρμογής της κινητής συσκευής.

Στην πιο πάνω εικόνα περιγράφεται η αρχιτεκτονική της εφαρμογής που βρίσκεται στην κινητή συσκευή. Παρατηρούμε πως το Carpooling Application βρίσκεται στο υψηλότερο Layer. Επίσης η εφαρμογή που αναπτύχθηκε χρησιμοποιεί το Android SDK το οποίο αποτελείται από αρκετά Frameworks που χρησιμοποιήθηκαν σε κάθε σημείο της. Τα διάφορα Frameworks είναι διαθέσιμα από το Android SDK βοήθησαν στην διαχείριση των διαφόρων notification, των λειτουργιών, την εναλλαγή δεδομένων μεταξύ των διαφορετικών δραστηριοτήτων, την χρήση των διαθέσιμων χαρτών από το Google Maps και σε άλλα σημεία.

Στο ακριβώς επόμενο Layer από το Application Framework έχουμε τις διάφορες βιβλιοθήκες που χρησιμοποιούν τα Frameworks για να εκτελέσουν τις διάφορες δραστηριότητες. Στο τελευταίο Layer έχουμε τα Drivers τα οποία χειρίζονται το Hardware τις κινητής συσκευής.

Επίσης για την υλοποίηση της εφαρμογής χρησιμοποιήσαμε ένα εξωτερικό Framework Feel The World [7]. Το Framework αυτό αποτελείται από χρήσιμες διαδικασίες όπως συμπίεση δεδομένων, εκτέλεση υποερωτήσεων (Query) στη βάση δεδομένων του κοινωνικού δικτύου Facebook, διαδικασίες αναφοράς τρέχουσας τοποθεσίας της κινητής συσκευής, συλλογή δεδομένων από τους αισθητήρες της κινητής συσκευής κτλ. Από το Framework αυτό χρησιμοποιήσαμε την διαδικασία εκτέλεσης υποερωτήσεων στην βάση δεδομένων του κοινωνικού δικτύου Facebook για να μπορέσουμε να πάρουμε τους φίλους ενός χρήστη που έχει στο αναφερόμενο κοινωνικό δίκτυο.



Εικόνα 4.3.2 Αναλυτικότερη αρχιτεκτονική εφαρμογής της κινητής συσκευής.

Στη πιο πάνω εικόνα αναλύεται περισσότερο η αρχιτεκτονική της εφαρμογής της κινητής συσκευής. Παρατηρούμε πως η εφαρμογή διασπάται σε τρία τμήματα όπου στο ψηλότερο επίπεδο έχουμε το Interface, ακολούθως το Application Layer και στο χαμηλότερο επίπεδο το Data Layer.

Interface:

Το Interface βρίσκεται στο υψηλότερο επίπεδο της εφαρμογής και είναι υπεύθυνο για την άμεση αλληλεπίδραση της εφαρμογής με το χρήστη. Ο χρήστης εκτελεί κάποιες διαδικασίες και με την βοήθεια των χαμηλότερων επιπέδων παράγονται τα αναμενόμενα αποτελέσματα, όπου και παρουσιάζονται στον χρήστη από αυτό το επίπεδο.

Application Layer:

Σε αυτό το επίπεδο έχουμε κάποιες ομάδες διαδικασιών οι οποίες διαχειρίζονται αναλόγως τα σχετικά δεδομένα του χρήστη.

- **Route Manager**

Το Route Manager είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες να διαχειρίζονται τα δρομολόγια του χρήστη.

1. Προσθήκη και διαγραφή δρομολογίων, όπου ο χρήστης προσθέτει ένα δρομολόγιο στη βάση δεδομένων με τα χαρακτηριστικά γνωρίσματα που το αποτελούν, ούτως ώστε να είναι διαθέσιμο στους φίλους του, ή το διαγράφει όπου και δεν θα είναι πλέον διαθέσιμο.
2. Εύρεση δρομολογίου, που βάσει κάποιων δεδομένων εκτελείται, αναζήτηση πιθανών δρομολογίων από τους φίλους του.
3. Κράτηση ή απελευθέρωση θέσης σε ένα δρομολόγιο. Γίνεται κράτηση μια θέσης σε ένα δρομολόγιο το οποίο θα εκτελεστεί ή αναλόγως ακυρώνεται μια θέση.
4. Εμφάνιση του ακριβές δρομολογίου που θα εκτελεστεί με την χρήση των Google Maps.
5. Εμφάνιση των πιο πρόσφατων δρομολογίων που προστέθηκαν από τους φίλους του χρήστη.
6. Εμφάνιση των δρομολογίων όπου συμμετείχε ο χρήστης στο παρελθόν.

- **Contact Manager**

Το Contact Manager είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για τη διαχείριση των φίλων του χρήστη.

1. Εμφάνιση των φίλων του συγκεκριμένου χρήστη.
2. Προσθήκη και διαγραφή φίλων από τον λογαριασμό του.
3. Αναζήτηση χρηστών της εφαρμογής από την βάση δεδομένων.
4. Εμφάνιση προσωπικών στοιχείων των φίλων του χρήστη.

- **Request Manager**

Το Request Manager είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες να διαχειρίζονται τα αιτήματα του χρήστη. Τα αιτήματα που μπορεί να εκτελέσει ένας χρήστης είναι δύο ειδών. Αίτημα για να κρατήσει μια θέση σε ένα δρομολόγιο και αίτημα για να προσθέσει ένα χρήστη στους φίλους του.

1. Δημιουργία νέου αιτήματος.
2. Εμφάνιση εισερχόμενων και εξερχομένων αιτημάτων στον χρήστη.

3. Αποδοχή ενός αιτήματος από τον χρήστη όπου εκτελείται η κατάλληλη διαδικασία αναλόγως αν είναι αίτημα για κράτηση θέσης σε ένα δρομολόγιο ή αν είναι αίτημα για πρόσθεση ενός χρήστη (Friend Request).
4. Μη αποδοχή ενός αιτήματος.

- **Account Manager**

Το Account Manager είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για την διαχείριση των πληροφοριών του λογαριασμού του χρήστη.

1. Δημιουργία νέου λογαριασμού.
2. Τροποποίηση προσωπικών στοιχείων χρήστη.
3. Διαγραφή λογαριασμού.

- **Messaging Manager**

Το Messaging Manager είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για την διαχείριση των μηνυμάτων του χρήστη.

1. Αποστολή μηνύματος.
2. Διαγραφή ενός μηνύματος.
3. Εμφάνιση των εισερχομένων και εξερχομένων μηνυμάτων του χρήστη.
4. Αποστολή αυτοματοποιημένων μηνυμάτων από την εφαρμογή.

Data Layer

Το Data Layer είναι το χαμηλότερο επίπεδο της εφαρμογής. Το Data Layer είναι υπεύθυνο για την επεξεργασία των δεδομένων και για την αποστολή και αποδοχή τους από τον εξυπηρετητή. Το Data Layer αποτελείται από τέσσερεις ομάδες διαδικασιών.

- **Data Manager**

Το Data Manager είναι υπεύθυνο για την συλλογή των δεδομένων από το Application Layer. Ακολούθως τα τροποποιεί αναλόγως αν χρειαστεί και τα στέλνει στο Data Sender για να αποσταλούν στον εξυπηρετητή ή στο File Manager για να αποθηκευτούν ή να διαγράφουν από κάποιο αρχείο. Επίσης το Data Manager είναι υπεύθυνο να παίρνει τα δεδομένα από το Data Receiver να τα τροποποιεί αν χρειαστεί και ακολούθως να τα στέλνει πίσω στο Application Layer.

```
DataManager (Array Data[], Bit  type) {
        Array Results []
```

```
        if (type == 1) {
                DataSender( Data[])
                Results = DataReciever ()
                return Results[]
        }
        else {
                Results = FileManager(Data[])
                return Results []
        }
}
```

- **File Manager**

Το File Manager είναι υπεύθυνο να διαχειρίζεται την πρόσβαση, την εγγραφή και την διαγραφή στοιχείων από το αρχείο Users. Σε αυτό το αρχείο είναι αποθηκευμένο το User Name των χρηστών που χρησιμοποίησαν την συγκεκριμένη κινητή συσκευή, ούτως ώστε να μην χρειάζεται ο χρήστης να το γράφει κάθε φορά για να εισέρθει στην εφαρμογή.

```
FileManager (Array Data[]) {
        Users = File.open (users.txt)
        if (Data[0] == 1)
                Users.write(Data[1])
        else if (Data[0] == 2) {
                Array Results []
                Results = Users.getAll ()
                return Results[]
        }
        else
                Users.clear()
        return 1
}
```

- **Data Sender**

Το Data Sender είναι υπεύθυνο για την διαχείριση των δεδομένων που αποστέλλονται από το Data Manager, όπου ακολούθως δημιουργεί νέα σύνδεση με τον εξυπηρετητή και τα αποστέλλει για την εκτέλεση της αναμενόμενης διαδικασίας.

```
DataSender (Array Data[]) {
        connection = New Connection (IP, PORT)
        size = Data.size
        while (size > 0) {
                Data[size].toSting
                connection.send( Data[size])
                size = size - 1
        }
}
```

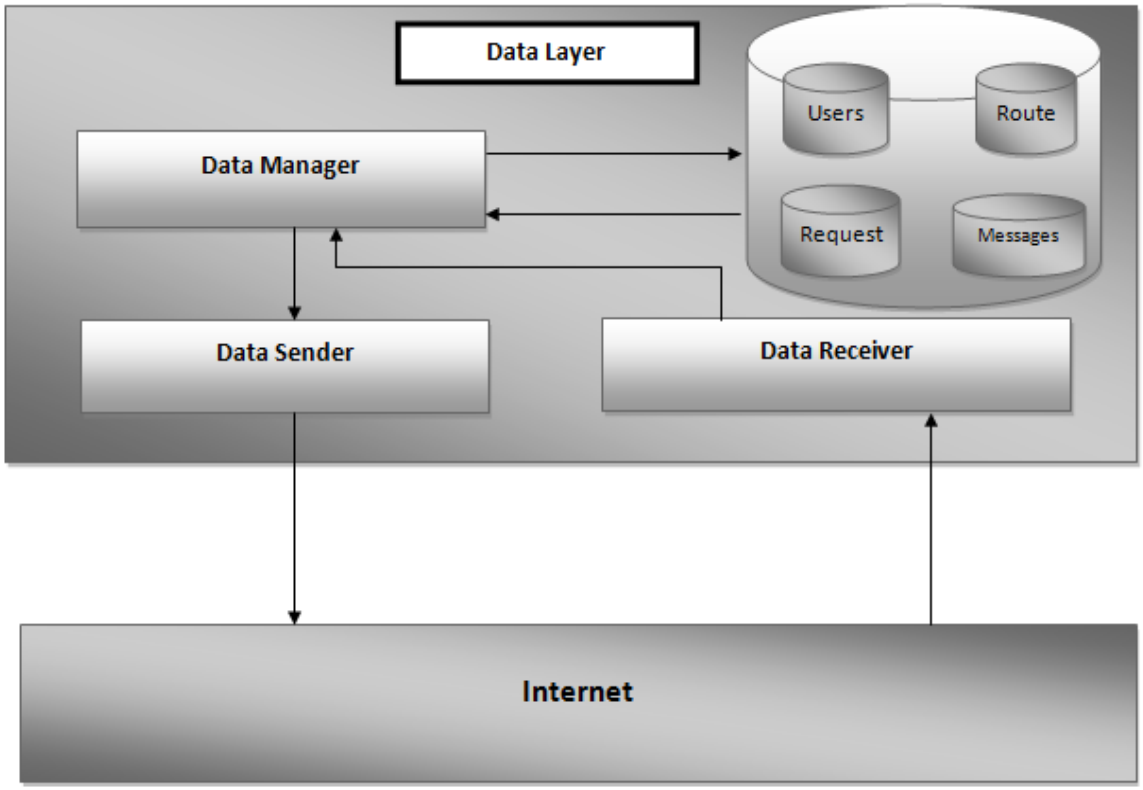
- **Data Receiver**

Το Data Receiver είναι υπεύθυνο να αποδέχεται τα δεδομένα που αποστέλλονται από τον εξυπηρετητή της εφαρμογής και να τα στέλνει στο Data Manager για περαιτέρω ανάλυση και τροποποίηση τους.

```
DataReceiver () {
        Array Data[]
        temp = connection.get()
        pos = 0
        While (temp) {
                Data[pos] = temp
                temp = connection.get()
                pos = pos + 1
        }
        Return Data[]
}
```

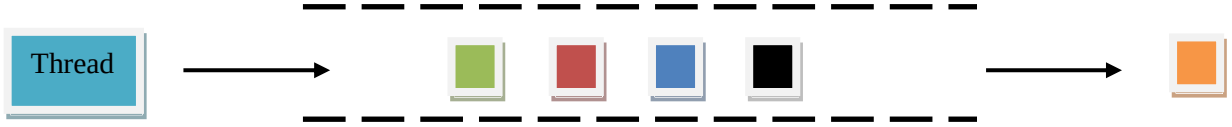
4.3.2 Hermes Server Application

Τώρα θα αναλύσουμε την αρχιτεκτονική της εφαρμογή που βρίσκεται εγκατεστημένη στον εξυπηρετητή. Θα δούμε τα τμήματα τα οποία είναι υπεύθυνα να λαμβάνουν τα δεδομένα από την κινητή συσκευή, να εκτελέσουν την επιθυμητή διαδικασία, να διαχειριστούν και να ενημερώσουν την βάση δεδομένων. Επίσης θα δούμε τους πίνακες που χρησιμοποιήσαμε για την δημιουργία της βάσης δεδομένων.



Εικόνα 4.3.2.1 Αρχιτεκτονική εφαρμογής εξυπηρετητή.

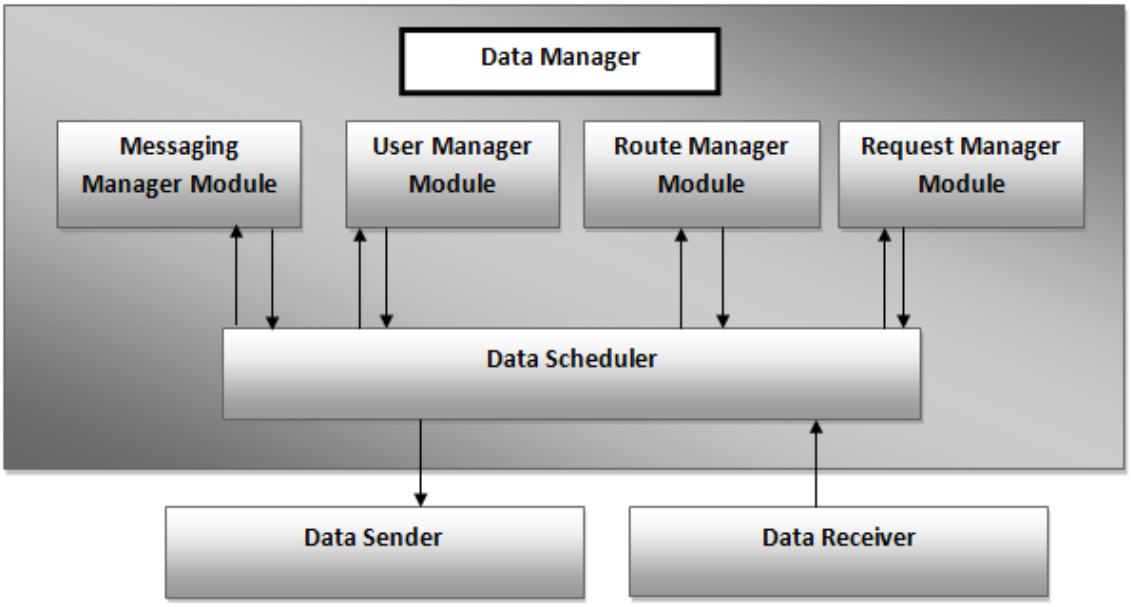
Όταν αποσταλούν δεδομένα από την κινητή συσκευή στον εξυπηρετητή τα δέχεται ο Data Receiver όπου δημιουργείται ένα νέο νήμα για την εκτέλεση του συγκεκριμένου αιτήματος του χρήστη. Τα νήματα που δημιουργούνται ικανοποιούνται σύμφωνα με το πιο έφτασε πρώτο στο Data Receiver, όπου κάποια νήματα περιμένουν στην ουρά για εκτέλεση προηγούμενων.



Ακολούθως για την εκτέλεση κάποιου νήματος στέλνονται τα δεδομένα στο Data Manager όπου αναγνωρίζεται ποία διαδικασία θα εκτελέσει και καλεί τα κατάλληλα Query από την βάση δεδομένων. Επίσης τροποποιεί τα δεδομένα όπου χρειάζεται, ή τα επεξεργάζεται περισσότερο και ακολούθως τα στέλνει στο Data Sender.

Το Data Sender μετατρέπει τα δεδομένα στην κατάλληλη μορφή όπου στην συνέχεια δρομολογούνται με την σωστή σειρά στην κινητή συσκευή. Μετά την αποστολή των δεδομένων στην κινητή συσκευή τερματίζεται το νήμα που δημιουργήθηκε από τον Data Receiver.

Σε αυτό το σημείο θα αναλύσουμε και να κατανοήσουμε την λειτουργία του Data Manager και να δούμε πια είναι τα καθήκοντα του, και πως υλοποιείται η επικοινωνία του με το Data Receiver και Data Sender.



Εικόνα 4.3.2.2 Αρχιτεκτονική του Data Manager στον εξυπηρετητή.

- **Data Receiver**

Στο Data Receiver στέλνονται δεδομένα από κινητές συσκευές. Μια κινητή συσκευή χρησιμοποιεί το IP και το Port Number της εφαρμογής για να στείλει τα δεδομένα της. Το Data Receiver αποδέχεται το αίτημα για δημιουργία νέας σύνδεσης που στέλνεται από μια κινητή συσκευή και ακολούθως δημιουργεί ένα νέο νήμα για το συγκεκριμένο αίτημα. Όταν αρχίσει η εκτέλεση του συγκεκριμένου νήματος το Data Receiver αποδέχεται τα δεδομένα που στάλθηκαν από την κινητή συσκευή και ακολούθως στέλνονται στο Data Scheduler.

```
DataReceiver (){
        main() {
                While (true) {
                        con = connection.accept()
                        connectionHandler (con)
                }
        }
connectionHandler( Connection con) {
        Thread t = new Thread ()
        t.run ()
}
run () {
        Array Data []
        Data = con.getAllData()
        DataScheduler (Data [])
        }
}
```

- **Data Scheduler**

Σημαντικό κομμάτι του Data Manager είναι το Data Scheduler. Το Data Scheduler είναι υπεύθυνο στο να αναγνωρίζει βάση κάποιων πρότυπων χαρακτηριστικών σε ποία ομάδα διεργασιών αναφέρονται τα συγκεκριμένα δεδομένα που έλαβε. Ακολούθως στέλνει τα δεδομένα σε αυτή την ομάδα. Μετά την εκτέλεση της διαδικασία από την συγκεκριμένη ομάδα στέλνονται τα αποτελέσματα στο Data Receive για να αποσταλούν πίσω στην κινητή συσκευή που έκανε το αίτημα.

```
DataScheduler (Array Data []) {
        if (Data[0] == 1) {
                Data = MessagingManager(Data)
                DataSender(Data)
        }
        else if (Data[0] == 2) {
                 Data = UserManager(Data)
                DataSender (Data)
        }
```

```
        else if (Data[0] == 3) {
                Data = RouteManager(Data)
                DataSender (Data)
        }
        else {
                Data = RequestManager(Data)
                DataSender (Data)
        }
}
```

- **Data Sender**

Το Data Sender είναι υπεύθυνο να αποδέχεται τα δεδομένα από το Data Scheduler και να τα αποστέλλει πίσω στην κινητή συσκευή μετατρέποντας τα στο κατάλληλο τύπο. Μετά την αποστολή των δεδομένων το νήμα τερματίζεται.

```
DataSender (Array Data []) {
        size = Data.size
        while (size> 0 ) {
                Data[size].toString
                con.send(Data[size])
                size = size – 1
        }
        con.close
}
```

Από την πιο πάνω εικόνα 3.3.2.2 αναγνωρίζουμε πως υπάρχουν 4 ομάδες διαδικασιών. Η κάθε ομάδα είναι υπεύθυνη να αναγνωρίσει την διαδικασία που πρέπει να εκτελέσει και ακολούθως να της στείλει τα κατάλληλα δεδομένα. Η κάθε ομάδα αποτελείται από διαδικασίες οι οποίες αναφέρονται για τον συγκεκριμένο τομέα. Οι ομάδες που έχουμε είναι οι ακόλουθες:

- **Messaging Manager Module**

Το Messaging Manager Module είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για την διαχείριση των μηνυμάτων των χρηστών.

1. Αποστολή ενός μηνύματος, όπου προστίθεται το μήνυμα στην βάση δεδομένων με τα κατάλληλα χαρακτηριστικά.

2. Εύρεση μηνυμάτων των χρηστών, όπου εντοπίζει τα εισερχόμενα και εξερχόμενα μηνύματα του χρήστη από την βάση δεδομένων.
3. Ανάκληση πληροφοριών ενός μηνύματος, όπου παίρνουμε τις σχετικές πληροφορίες που το αποτελούν.
4. Διαγραφή μηνύματος από την βάση δεδομένων.

- **User Manager Module**

Το User Manager Module είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για την διαχείριση των πληροφοριών των λογαριασμών των χρηστών.

1. Δημιουργία νέου λογαριασμού στην βάση δεδομένων.
2. Τροποποίηση προσωπικών στοιχείων του χρήστη.
3. Διαγραφή λογαριασμού.

- **Route Manager Module**

Το Route Manager Module είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για την διαχείριση των δρομολόγιων του χρήστη.

1. Προσθήκη και διαγραφή δρομολογίων.
2. Εύρεση δρομολόγιου. Σε αυτό το σημείο εκτελείται ο αλγόριθμος που υλοποιήσαμε, όσον αφορά την ταύτιση δρομολογίων.
3. Κράτηση ή απελευθέρωση θέσης από ένα δρομολόγιο.
4. Υπολογισμός των Credits που θα πάρει ο οδηγός ενός δρομολογίου.

- **Request Manager Module**

Το Request Manager Module είναι μια κατηγορία από διαδικασίες οι οποίες είναι υπεύθυνες για την διαχείριση των αιτημάτων του χρήστη. Τα αιτήματα που μπορεί να εκτελέσει ένας χρήστης είναι δύο ειδών. Αίτημα για να κρατήσει μια θέση σε ένα δρομολόγιο και αίτημα για να προσθέσει ένα χρήστη στους φίλους του.

1. Δημιουργία νέου αιτήματος.
2. Εμφάνιση αιτημάτων στον χρήστη.
3. Αποδοχή αιτήματος.
4. Διαγραφή αιτήματος (μη αποδοχή του αιτήματος).

Βάση δεδομένων

Για την υλοποίηση του Hermes Server Application χρησιμοποιήσαμε MySQL βάση δεδομένων με τους ακόλουθους πίνακες:

1. **Users:** αποθηκεύονται τα στοιχεία των χρηστών της εφαρμογής.
2. **Contacts:** αποθηκεύονται το ποιος χρήστης είναι φίλος με ποιο.
3. **Routes:** αποθηκεύονται τα δρομολόγια που προστίθενται από τους χρήστες.
4. **Route Passengers:** αποθηκεύονται ποιοι χρήστες λαμβάνουν μέρος σε ένα συγκεκριμένο δρομολόγιο.
5. **Messages:** αποθηκεύονται τα μηνύματα που αποστέλλονται μεταξύ των χρηστών.
6. **Request:** αποθηκεύονται τα αιτήματα των χρηστών, όπου μπορεί να είναι είτε για δρομολόγια είτε για να προσθέσουν ένα χρήστη στο λογαριασμό τους.

1. Πίνακας Users

User Id	User Name	Password	Name	Surname	Address	Mobile	Credits	Feedback	Last Login	Social Id	Car Type

User Id	IDENTITY(1,1)	Not Null
User Name	NVARCHAR (30)	Not Null
Password	NVARCHAR (20)	Not Null
Name	NVARCHAR (30)	Not Null
Address	NVARCHAR (50)	Not Null
Mobile	NVARCHAR (20)	Not Null
Credits	INTEGER	Not Null
Feedback	INTEGER	Not Null
Last Login	SMALLDATETIME	
Social Id	INTEGER	
Car Type	NVARCHAR (20)	

Στον πίνακα Users αποθηκεύεται ο κάθε χρήστης που έχει λογαριασμό στην εφαρμογή. Αποθηκεύονται τα στοιχεία του χρήστη όπως φαίνεται πιο πάνω. Επίσης για το κάθε χρήστη αποθηκεύεται ο αριθμός των Credits που πήρε από την εφαρμογή, το Feedback που μπορεί να του αφήσει κάθε χρήστης μετά το τερματισμό ενός δρομολογίου όπου ήταν ο οδηγός. Το Last Login είναι η ημερομηνία και η ώρα για την οποία ο χρήστης χρησιμοποίησε την εφαρμογή για τελευταία φορά και αποθηκεύεται για να μπορεί να δει ο χρήστης αν παραβιάστηκε ο λογαριασμός του. Το Social Id είναι το id του συγκεκριμένου χρήστη από το Facebook αν το σύνδεσε με την εφαρμογή.

2. Πίνακας Contacts

User Id1	User Id2

User Id1	INTEGER	Not Null
User Id2	INTEGER	Not Null

Στον πιο πάνω πίνακα χρησιμοποιούμε δύο πεδία. Το User Id1 και το User Id2 τα οποία είναι Foreign Key από τον πίνακα Users. Τα δυο αυτά πεδία δηλώνουν πως ο χρήστης με το User Id1 είναι φίλος με τον χρήστη με το User Id2.

3. Πίνακας Routes

Route Id	Src Lat	Src Lng	Dest Lat	Dest Lng	User Id	Date Time	Route Flexibility	Available Seats	Area1 Lat	Area1 Lng	Area2 Lat	Area2 Lng	Note	Cur Lat	Cur Lng

Route Id	IDENTITY(1,1)	Not Null
Src Lat	DOUBLE	Not Null
Src Lng	DOUBLE	Not Null
Dest Lat	DOUBLE	Not Null
Dest Lng	DOUBLE	Not Null
User Id	INTEGER	Not Null
Date Time	SMALLDATETIME	Not Null
Route Flexibility	DOUBLE	Not Null
Available Seats	INTEGER	Not Null
Area1 Lat	DOUBLE	Not Null
Area1 Lng	DOUBLE	Not Null
Area2 Lat	DOUBLE	Not Null
Area2 Lng	DOUBLE	Not Null
Note	NVARCHAR (30)	
Cur Lat	DOUBLE	
Cur Lng	DOUBLE	

Στο πιο πάνω πίνακα αποθηκεύουμε τα στοιχεία ενός δρομολογίου, όπως το Id του, το γεωγραφικό μήκος και πλάτος των σημείων έναρξης και τερματισμού του δρομολογίου (Src Lat, Src Lng κτλ), τον οδηγό (User Id) όπου είναι Foreign Key από τον πίνακα Users, την ημερομηνία που θα διεξαχθεί, το Route Flexibility, τις διαθέσιμες θέσεις που υπάρχουν και την περιοχή που ορίζει το συγκεκριμένο δρομολόγιο. Η περιοχή αυτή είναι τα γεωγραφικά σημεία που βρήκαμε με την εκτέλεση του πρώτου τμήματος του αλγορίθμου εύρεσης. Στο πεδίο Note αποθηκεύουμε τις σημειώσεις που πρόσθεσε ο χρήστης για το συγκεκριμένο δρομολόγιο, ούτως ώστε να είναι διαθέσιμες προς στους άλλους χρήστες. Το πεδίο Cur Lat και Cur Lng χρησιμοποιούνται για να αποθηκεύσουμε την τρέχον θέση του οδηγού όταν ξεκινήσει η εκτέλεση του δρομολογίου από τον χρήστη. Αυτά τα δύο πεδία είναι χρήσιμα στην υλοποίηση της δυναμικής εύρεσης δρομολογίων, όπου οι χρήστες μπορούν να κάνουν

αιτήματα για δρομολόγια τα οποία εκτελούνται την συγκεκριμένη στιγμή. Σε αυτή την περίπτωση η τρέχουσα θέση του οδηγού λειτουργεί σαν αρχικό σημείο του δρομολογίου για να ελεγχθεί κατά πόσον μπορεί ο χρήστης να εξυπηρετήσει ένα άλλο δρομολόγιο.

4. Πίνακας Route Passengers

Route Id	User Id	Src Lat	Src Lng	Dest Lat	Dest Lng

Route Id	IDENTITY(1,1)	Not Null
User Id	INTEGER	Not Null
Src Lat	DOUBLE	Not Null
Src Lng	DOUBLE	Not Null
Dest Lat	DOUBLE	Not Null
Dest Lng	DOUBLE	Not Null

Στον πιο πάνω πίνακα αποθηκεύονται οι επιβάτες του δρομολογίου με το συγκεκριμένο Id. Το Route Id είναι Foreign Key από τον πίνακα Routes και το User Id είναι Foreign Key από τον πίνακα Users, όπου δηλώνουν ποιος χρήστης συμμετέχει στο συγκεκριμένο δρομολόγιο. Επίσης για τον κάθε χρήστη που συμμετέχει σε ένα δρομολόγιο αποθηκεύουμε το γεωγραφικό πλάτος και μήκος για το σημείο έναρξης και τερματισμού του δρομολογίου, γιατί οι χρήστες που συμμετέχουν σε ένα δρομολόγιο μπορούν να έχουν διαφορετικά σημεία έναρξης και τερματισμού μεταξύ τους.

5. Πίνακας Messages

Message Id	User Id1	User Id2	Read	Message	Date Time	Subject

Message Id	IDENTITY(1,1)	Not Null
User Id1	INTEGER	Not Null
User Id2	INTEGER	Not Null
Read	BIT	Not Null
Message	TEXT	Not Null
Date Time	SMALLDATETIME	Not Null
Subject	NVARCHAR (30)	

Στον πιο πάνω πίνακα αποθηκεύουμε τα χαρακτηριστικά ενός μηνύματος. Το User Id1 και User Id2 είναι Foreign Key από τον πίνακα Users και δηλώνουν το ποιος χρήστης είναι ο αποστολέας του μηνύματος και ποιος ο παραλήπτης. Το πεδίο Read είναι τύπου Bit και παίρνει την τιμή 1 αν έχει διαβαστεί από τον χρήστη ή 0 αν δεν έχει διαβαστεί.

6. Πίνακας Request

Request Id	User Id1	User Id2	Type	Src Lat	Date Time	Src Lng	Dest Lat	Dest Lng

Request Id	IDENTITY(1,1)	Not Null
User Id1	INTEGER	Not Null
User Id2	INTEGER	Not Null
Type	BIT	Not Null
Date Time	SMALLDATETIME	Not Null
Src Lat	DOUBLE	
Src Lng	DOUBLE	
Dest Lat	DOUBLE	
Dest Lng	DOUBLE	

Στον πιο πάνω πίνακα αποθηκεύουμε τα χαρακτηριστικά ενός Request. Το User Id1 και User Id2 είναι Foreign Key από τον πίνακα Users και απεικονίζουν το ποιος χρήστης έκανε το Request και σε ποιον. Το πεδίο Type είναι τύπου Bit και παίρνει την τιμή 1 αν είναι Friend Request ή 0 αν είναι Route Request. Τα πεδία Src Lat, Src Lng, Dest Lat, και Dest Lng χρησιμοποιούνται όταν το Request είναι τύπου Route Request, όπου αποθηκεύεται το γεωγραφικό πλάτος και μήκος του σημείου έναρξης και τερματισμού. Να αναφέρουμε πως μετά την απάντηση του χρήστη για κάποιο Request διαγράφεται από την βάση δεδομένων.

Κεφάλαιο 5

Αξιολόγηση

5.1 Σημαντικότητα πειραμάτων και μετρήσεων

Σε αυτό το κεφάλαιο θα αναφερθούμε σε κάποιες πειραματικές μετρήσεις που κάναμε με πολλούς χρήστες και πολλά δρομολόγια. Επίσης θα σχολιάσουμε την σημαντικότητα των πειραμάτων και θα παρουσιάσουμε τα αποτελέσματα τους. Τέλος θα αναφερθούμε στους λόγους που οφείλονται οι μετρήσεις που πήραμε.

Στο τελευταίο υποκεφάλαιο θα συγκρίνουμε τον αλγόριθμο εύρεσης δρομολογίων που αναπτύχθηκε με έναν άλλο αλγόριθμο ο οποίος χρησιμοποιείται από την εφαρμογή Ride Remedy. Επίσης θα κάνουμε σύγκριση του αλγορίθμου που υλοποιήσαμε με το χειρότερο σενάριο υλοποίησης του παρόντος προβλήματος.

5.2 Πειράματα που διατρέξαμε

Για τον λόγο που αναφέραμε διατρέξαμε διάφορα πειράματα τα οποία αναφέρονται στα κυριότερα σημεία της εφαρμογής.

Για την εκτέλεση των πειραμάτων χρησιμοποιήσαμε:

Κινητή συσκευή HTC Desire [21]
CPU
1Gz Processing Speed
Platform
Android 2.2, Kernel Version 2.6.32.15
Storage
Rom 512 Mb
Ram 576 Mb
Wi-Fi
IEEE 802.11 b/g

Υπολογιστής
CPU
Intel Xeon E5440 2.83GHz [22]
Number of cores 4
Number of threads 4
Cache 12 Mb
Operating System
Ubuntu 8.04 Kernel Version 2.6.24
Memory
25 GB

Μετρικές πειραμάτων:

1. Χρόνος εκτέλεσης (Execution Time ms).
2. Απόσταση δρομολογίου σε χιλιόμετρα (Distance km).
3. Αριθμός επιβατών (Passengers)
4. Αριθμός δρομολογίων που επέστρεψε η αναζήτηση (Similar Routes)

Πρώτο Πείραμα

Το πρώτο πείραμα που διατρέξαμε αναφέρεται στον αλγόριθμο ελέγχου και σύγκρισης δύο δρομολογίων. Να αναφέρουμε πως ο χρήστης μπορεί να δει τα πρόσφατα δρομολόγια τα οποία προστεθήκαν από τους φίλους του και να κάνει αίτηση για να κρατήσει θέση σε αυτά αν υπάρχει διαθέσιμη, νοουμένου ότι έχει δώσει το σημείο έναρξης και τερματισμού για τα οποία θέλει να περάσει ο οδηγός για να τον εξυπηρετήσει αλλά επίσης και στο ότι ο χρήστης συμφωνεί με την ώρα που θα εκτελεστεί το δρομολόγιο που έχει δώσει ο οδηγός. Το πρώτο πείραμα που αναφέρεται σε αυτό το σημείο της εφαρμογής και ελέγχουμε τον χρόνο εκτέλεσης του σε διάφορα σενάρια.

Δεύτερο Πείραμα

Το δεύτερο πείραμα που διατρέξαμε αναφέρεται στο σημείο όπου σχηματίζουμε το ακριβές δρομολόγιο, χρησιμοποιώντας Google Maps. Σε αυτό το σημείο θέλουμε να βρούμε τον πιο καλό δρόμο τον οποίο θα ακολουθήσει ο χρήστης αλλά επίσης και ποίο χρήστη θα περάσει να πάρει πρώτα, ποιον ακολούθως, αλλά και πως θα εξυπηρέτηση τους χρήστες με κύριο κριτήριο να ακολουθήσει τη συντομότερη διαδρομή. Αυτό το πείραμα αναφέρεται σε δρομολόγια τα οποία έχουν 2 ή περισσότερους χρήστες με διαφορετικά σημεία έναρξης και τερματισμού ο κάθε ένας. Αυτό το οποίο ελέγχουμε σε αυτό το πείραμα είναι ο χρόνος ο οποίος χρειάζεται για να παρουσιάσουμε στον χάρτη το δρομολόγιο που θα ακολουθηθεί δεδομένου του αρχικού σημείου και του τελικού του δρομολογίου αλλά επίσης και των αρχικών και τελικών σημείων των υπόλοιπων χρηστών που συμμετέχουν στο συγκεκριμένο δρομολόγιο.

Τρίτο Πείραμα

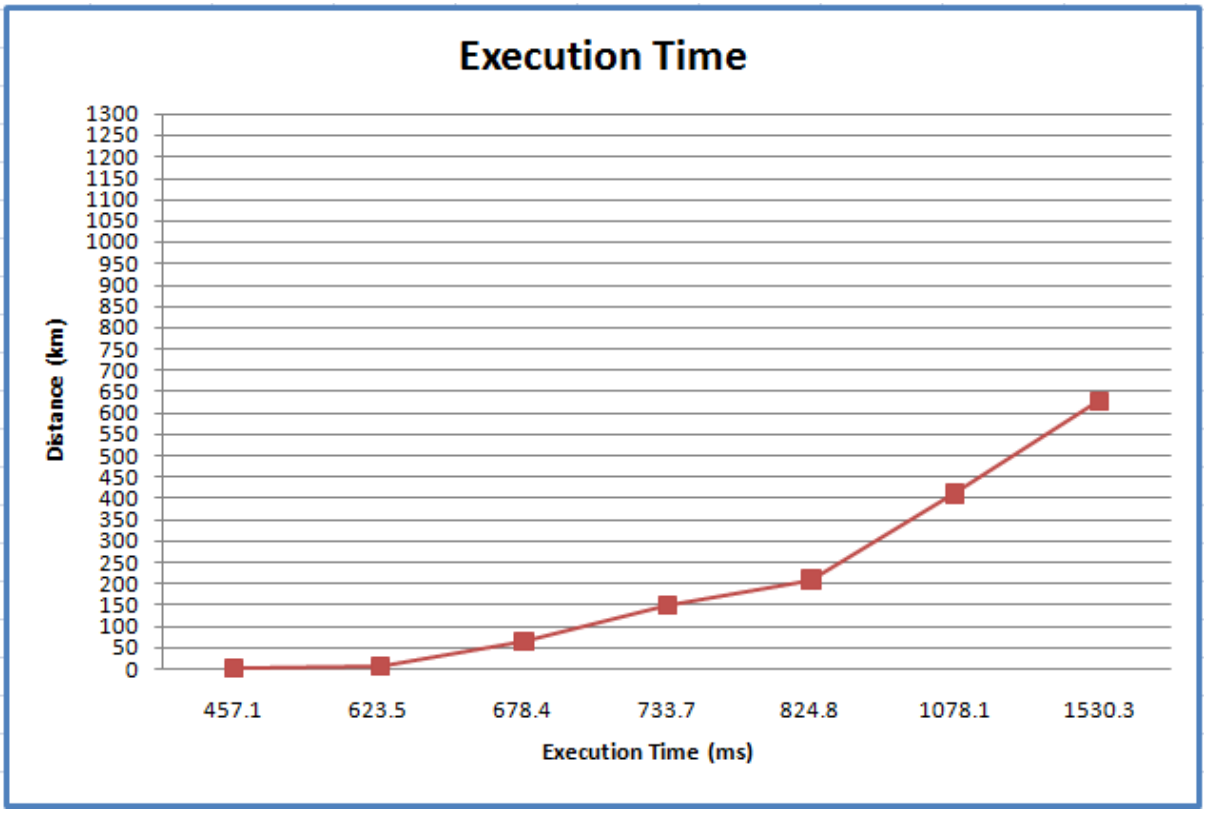
Το τρίτο πείραμα που διατρέξαμε αναφέρεται στον αλγόριθμο αναζήτησης που αναλύσαμε στο κεφάλαιο τρία. Σε αυτό το πείραμα θέλουμε να δούμε τον χρόνο εκτέλεσης, σε διάφορα σενάρια όπου έχουμε έναν μεγάλο αριθμό δρομολογίων αποθηκευμένα στη βάση δεδομένων. Επίσης συγκρίναμε τον αλγόριθμο εύρεσης δρομολογίων με τον αλγόριθμο που χρησιμοποιεί η εφαρμογή Ride Remedy, αλλά επίσης και με το χειρότερο σενάριο υλοποίησης του.

5.3 Παρουσίαση πειραματικών αποτελεσμάτων και αξιολόγηση

Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε γραφικά τα αποτελέσματα που πήραμε από τις πειραματικές μετρήσεις που κάναμε και ακολούθως θα τα σχολιάσουμε και θα επεξηγήσουμε που οφείλονται τα αποτελέσματα.

Πρώτο πείραμα

Το πρώτο πείραμα που εκτελέσαμε αναφέρεται στον αλγόριθμο ελέγχου και σύγκρισης δύο δρομολογίων, όπου έχει σαν είσοδο τα αρχικά και τελικά σημεία των δύο δρομολογίων, αλλά επίσης το Route Flexibility που έχει δώσει ο οδηγός. Ο αλγόριθμος αυτός ελέγχει κατά πόσο μπορεί να υπάρχει ένα σημείο του πρώτου δρομολογίου όπου να ικανοποιεί το δεύτερο δρομολόγιο αλλά επίσης να τηρείται και ο περιορισμός απόσταση (Route Flexibility). Να αναφέρουμε πως αυτό τον έλεγχο τον κάνουμε για το αρχικό και τελικό σημείο και πρέπει να ικανοποιούνται και τα δύο για να μπορεί ο χρήστης να κρατήσει θέση στο συγκεκριμένο δρομολόγιο.

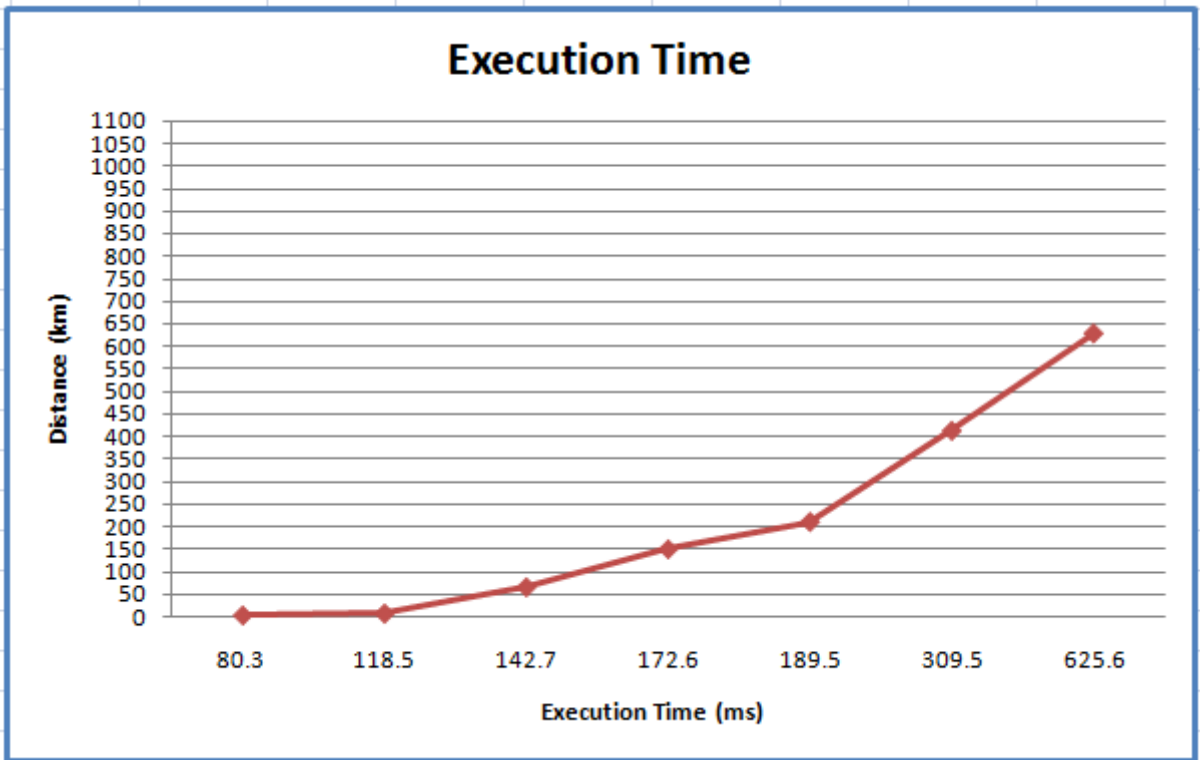


Εικόνα 5.3.1 Γραφική παράσταση ταύτισης σημείου έναρξης και τερματισμού των δύο δρομολογίων.

Στην πιο πάνω γραφική παράσταση βλέπουμε τους χρόνους εκτέλεσης του αλγορίθμου εκτέλεσης και σύγκρισης δύο δρομολογίων. Στον άξονα Χ βλέπουμε την απόσταση του δρομολογίου σε Km, και στον άξονα Υ βλέπουμε τον χρόνο που χρειάστηκε για να εκτελεστεί ο αλγόριθμος σε millisecond. Στα πιο πάνω δρομολόγια ταυτίστηκε το σημείο έναρξης αλλά επίσης και το σημείο τερματισμού των δύο δρομολογίων. Με τον όρο ταυτίστηκαν εννοούμε πως ο οδηγός που πρόσθεσε το πρώτο δρομολόγιο μπορεί να εξυπηρετήσει το δεύτερο δρομολόγιο που αναφέρεται στον χρήστη που εκτέλεσε αυτόν τον έλεγχο.

Από την πιο πάνω γραφική παράσταση παρατηρούμε πως ο χρόνος εκτέλεσης του αλγόριθμου εξαρτάται από την πραγματική απόσταση του δρομολογίου. Βλέπουμε πως ένα δρομολόγιο απόστασης 3,522 Km χρειάστηκε περίπου 463 ms (0,46 δευτερόλεπτα) για να ταυτιστεί με το άλλο δρομολόγιο.

Επίσης παρατηρούμε πως για ένα μεγάλο δρομολόγιο 628 Km περίπου χρειάστηκε 1530 ms (1,5 δευτερόλεπτα) . Επίσης να αναφέρουμε πως για δρομολόγια μέσης απόστασης (δηλ. 20 km – 100 Km) χρειάζονται περίπου 600 ms – 750 ms για να ταυτιστούν.



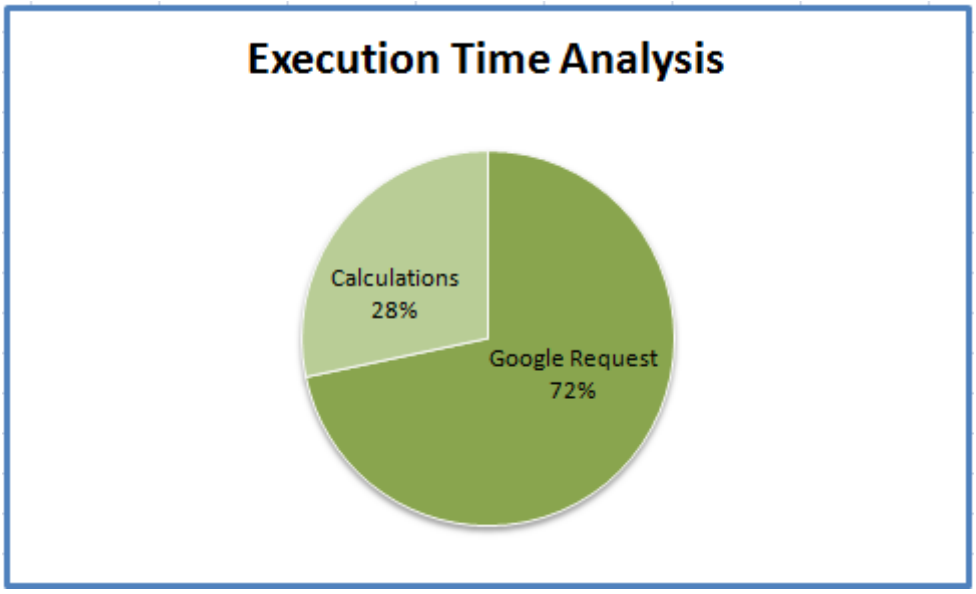
Εικόνα 5.3.2 Γραφική παράσταση ταύτισης σημείου έναρξης και όχι του σημείου τερματισμού των δύο δρομολογίων.

Στη πιο πάνω γραφική παράσταση βλέπουμε τους χρόνους εκτέλεσης του ίδιου αλγόριθμου με την προηγούμενη γραφική παράσταση με την διαφορά ότι τα δρομολόγια ταυτίστηκαν μόνο στο σημείο έναρξης. Δηλαδή το σημείο τερματισμού του δεύτερου δρομολογίου δεν τηρούσε το κριτήριο απόστασης (Route Flexibility) που έχει δώσει ο οδηγός όταν πρόσθεσε το δρομολόγιο.

Από την πιο πάνω γραφική παράσταση όπως και προηγουμένως παρατηρούμε πως ο χρόνος εκτέλεση του αλγορίθμου εξαρτάται και πάλι από την πραγματική απόσταση του δρομολογίου. Επίσης παρατηρούμε πως οι χρόνοι εκτέλεσης σε αυτό το σενάριο είναι πολύ μικρότεροι από τους προηγούμενους. Ο λόγος για τον οποίο συμβαίνει αυτό είναι το ότι ο αλγόριθμός καταλαβαίνει μέσω της απόστασης σε ευθεία, του σημείου τερματισμού με το πρώτο δρομολόγιο πως είναι μεγαλύτερος από το Route Flexibility που έδωσε ο χρήστης και δεν χρειαζόταν να κάνει Requeststο Google Maps για να πάρει την πραγματική απόσταση που απέχουν τα δύο γεωγραφικά σημεία. Όπως θα δούμε και στην επόμενη γραφική παράσταση τα Request που γίνονται στο Google Maps κοστίζουν γιατί χρειάζεται σημαντικός χρόνος για να πάρουμε τα αποτελέσματα τους.

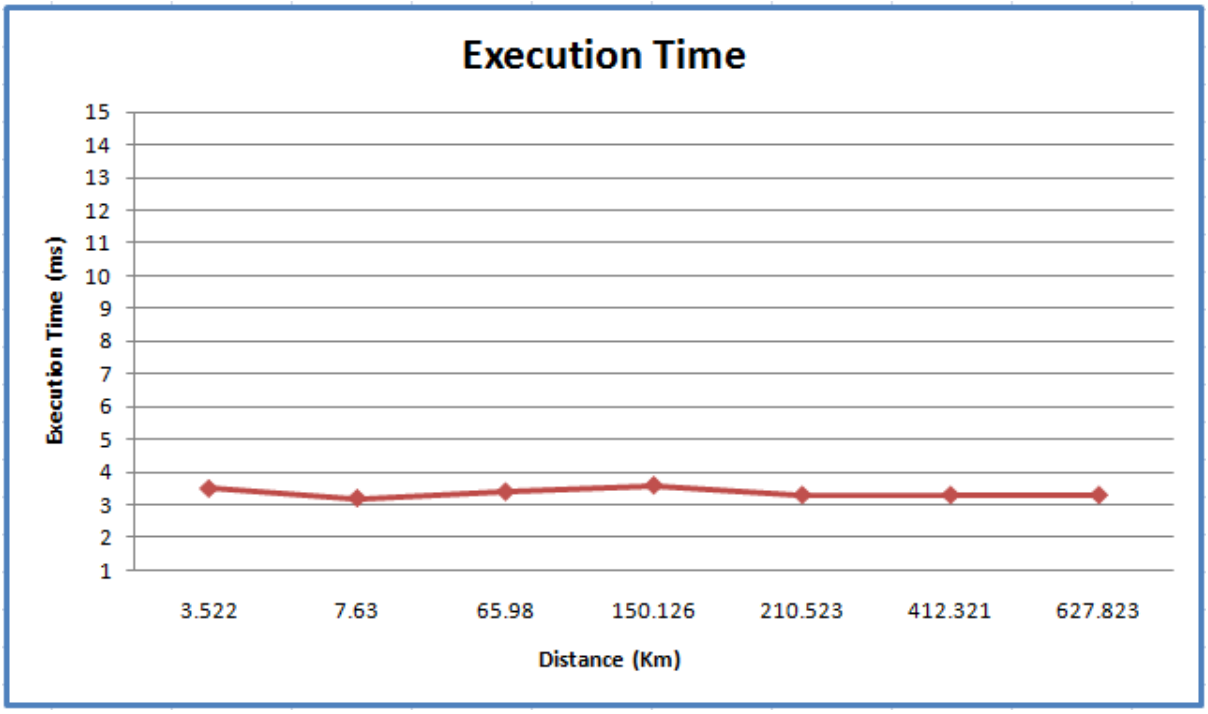
Επίσης από την πιο πάνω γραφική παράσταση παρατηρούμε πως ο αλγόριθμος χρειάζεται περίπου 100 ms – 150 ms για δρομολόγια μέσης απόστασης. Επίσης παρατηρούμε πως στο

χειρότερο σενάριο που δώσαμε όπου το δρομολόγιο έχει απόσταση 628 Km χρειάζονται περίπου 625 ms(0,62 δευτερόλεπτα).



Εικόνα 5.3.3 Γραφική παράσταση διαμερισμού χρόνου εκτέλεσης για σύγκριση δυο δρομολογίων .

Στην πιο πάνω γραφική παράσταση βλέπουμε τον διαμερισμό του χρόνου εκτέλεσης που πήραμε κατά μέσο όρο. Βλέπουμε πως το 72% κατά μέσο όρο του συνολικού χρόνου εκτέλεσης του αλγορίθμου οφείλεται σε Request που κάνουμε στο Google Maps. Το ποσοστό αυτό είναι πολύ σημαντικό για τον αλγόριθμο γιατί όπως βλέπουμε ξεπερνά κατά πολύ το 50% του μέσου όρου του χρόνου εκτέλεσης. Για τον λόγο αυτό πρέπει να περιορίζονται τα Request όσο το δυνατό περισσότερο για να επιτευχτεί η μείωση του χρόνου εκτέλεσης. Ο λόγος για τον οποίο χρειάζεται αυτός ο χρόνος είναι γιατί δημιουργείται σύνδεση με το Google Maps και παίρνουμε από αυτό μια μεγάλη σε μέγεθος λίστα που περιέχει αρκετά χαρακτηριστικά του δρομολογίου για το οποίο κάναμε το Request. Επίσης παρατηρούμε πως οι υπόλοιπες υπολογιστικές πράξεις που εκτελεί ο αλγόριθμος παίρνουν μόλις το 28% κατά μέσο όρο του συνολικού χρόνου εκτέλεσης.



Εικόνα 5.3.4 Γραφική παράσταση όπου δεν ταυτίζεται το σημείο έναρξης ούτε το σημείο τερματισμού των δύο δρομολογίων.

Στη πιο πάνω γραφική παράσταση βλέπουμε όπως και πριν τους χρόνους εκτέλεσης του ίδιου αλγόριθμου με τις προηγούμενες δύο γραφικές παραστάσεις με την διαφορά ότι τα δρομολόγια δεν ταυτίστηκαν ούτε στο σημείο έναρξης, ούτε στο σημείο τερματισμού.

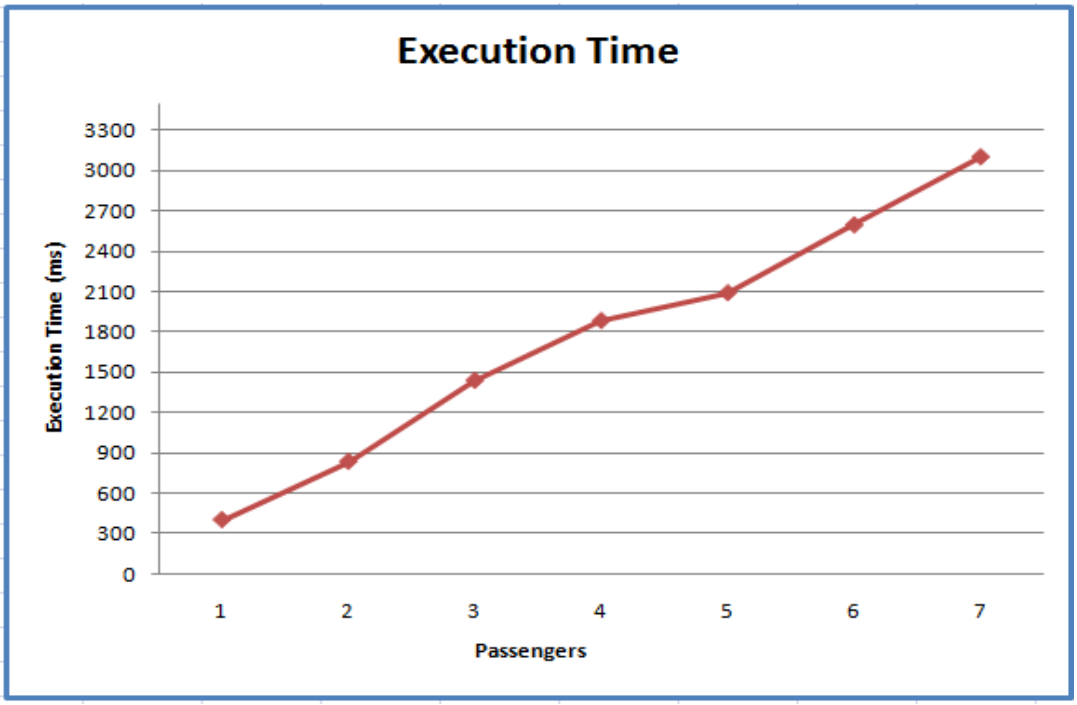
Από την πιο πάνω γραφική παράσταση βλέπουμε πως ο χρόνος εκτέλεση του αλγορίθμου σε αυτή την περίπτωση σε αντίθεση με τις δύο προηγούμενες, δεν εξαρτάται από την πραγματική απόσταση του δρομολογίου. Επίσης παρατηρούμε πως οι χρόνοι εκτέλεσης σε αυτό το σενάριο είναι πάρα πολύ μικρότεροι από τις δύο προηγούμενες περιπτώσεις και είναι σχεδόν μηδαμινοί. Ο λόγος για τον οποίο συμβαίνει αυτό είναι το ότι ο αλγόριθμος αντιλαμβάνεται μέσω της περιοχής που ορίζει το πρώτο δρομολόγιο πως δεν συμπίπτει με το δεύτερο δρομολόγιο και πως είναι εντελώς διαφορετικά μεταξύ τους. Έτσι με την χρήση του αλγορίθμου που παράγει την περιοχή που καθορίζει το κάθε ένα δρομολόγιο δεν χρειάζεται να γίνει κάποιο Request στο Google Maps για να πάρουμε την πραγματική απόσταση που απέχουν τα γεωγραφικά σημεία μεταξύ των δύο δρομολογίων.

Επίσης από την πιο πάνω γραφική παράσταση βλέπουμε πως ο χρόνος εκτέλεσης του αλγόριθμου στα διάφορα σενάρια είναι περίπου 3 ms που είναι σχεδόν μηδενικός χρόνος εκτέλεσης. Αυτό το χαρακτηριστικό του αλγορίθμου είναι πολύ σημαντικό για μετέπειτα στάδιο όταν θα εκτελούμε αναζήτηση πάνω σε όλα τα διαθέσιμα δρομολόγια που υπάρχουν στην βάση δεδομένων την συγκεκριμένη ημερομηνία και ώρα, όπου μπορεί σε μερικές

περιπτώσεις να είναι αρκετές εκατοντάδες. Με την βοήθεια του αλγορίθμου που ορίζει την περιοχή του δρομολογίου θα μπορέσουμε να αποκλείσουμε τα άσχετα μεταξύ τους δρομολόγια σε ελάχιστο χρονικό διάστημα και θα εκτελούνται πράξεις μόνο στα παρόμοια μεταξύ τους δρομολόγια μειώνοντας έτσι τις υπολογιστικές πράξεις που θα εκτελεστούν που θα έχουν σαν αποτέλεσμα και μείωση του χρόνου εκτέλεσης.

Δεύτερο πείραμα

Το δεύτερο πείραμα που εκτελέσαμε αναφέρεται στον σημείο της εφαρμογής όπου σχηματίζουμε το ακριβές δρομολόγιο, χρησιμοποιώντας Google Maps. Σε αυτό το σημείο θέλουμε να βρούμε το καλύτερο μονοπάτι το οποίο θα ακολουθήσει ο χρήστης αλλά επίσης και ποιόν χρήστη θα περάσει να πάρει πρώτα ποιόν ακολούθως αλλά και πως θα εξυπηρετήσει τους χρήστες με κύριο κριτήριο να ακολουθήσει τη συντομότερη διαδρομή.



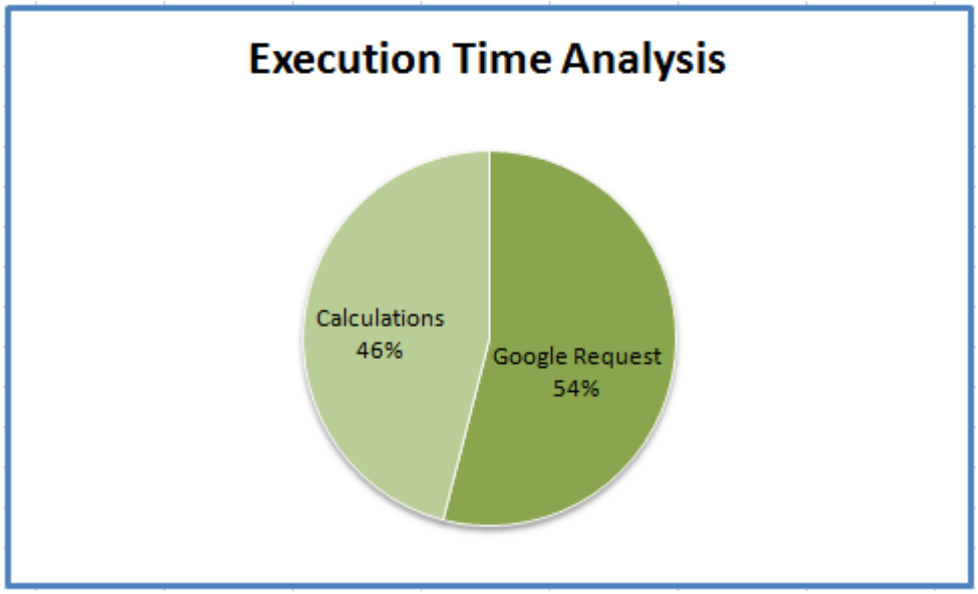
Εικόνα 5.3.5 Γραφική παράσταση χρόνου εκτέλεσης για σχηματισμό του ακριβές δρομολογίου.

Στη πιο πάνω γραφική παράσταση παρατηρούμε τον χρόνο εκτέλεσης που χρειάζεται για σχηματισμό του ολοκληρωμένου δρομολόγιου το οποίο θα ακολουθήσει ο οδηγός από όλα τα αρχικά και τελικά γεωγραφικά σημεία των επιβατών.

Στον άξονα Χ βλέπουμε τον αριθμό των χρηστών που συμμετέχουν στο δρομολόγιο, και στον άξονα Υ βλέπουμε τον χρόνο εκτέλεσης που χρειάστηκε σε κάθε ένα από τα σενάρια. Παρατηρούμε πως ο χρόνος εκτέλεσης εξαρτάται από τον αριθμό των επιβατών. Αυτό το

φαινόμενο ήταν αναμενόμενο για το λόγο ότι όσοι περισσότεροι επιβάτες υπάρχουν σε ένα δρομολόγιο με διαφορετικό αρχικό και τελικό σημείο για τον κάθε ένα τόσο περισσότερες υπολογιστικές πράξεις και Request στο Google Maps θα εκτελεστούν για την σωστή δημιουργία του δρομολογίου.

Στο πιο πάνω πείραμα πήραμε μετρήσεις για τον χρόνο εκτέλεσης του αλγόριθμου για 1 μέχρι 7 επιβάτες σε ένα δρομολόγιο και βλέπουμε ότι για 3 μέχρι 5 επιβάτες χρειάζονται περίπου 1,5 με 2 δευτερόλεπτα. Στο χειρότερο σενάριο που φαίνεται στη πιο πάνω γραφική παράσταση όπου έχουμε 7 επιβάτες βλέπουμε ότι χρειάζονται περίπου 3 δευτερόλεπτα για να εκτελεστεί ο αλγόριθμος.



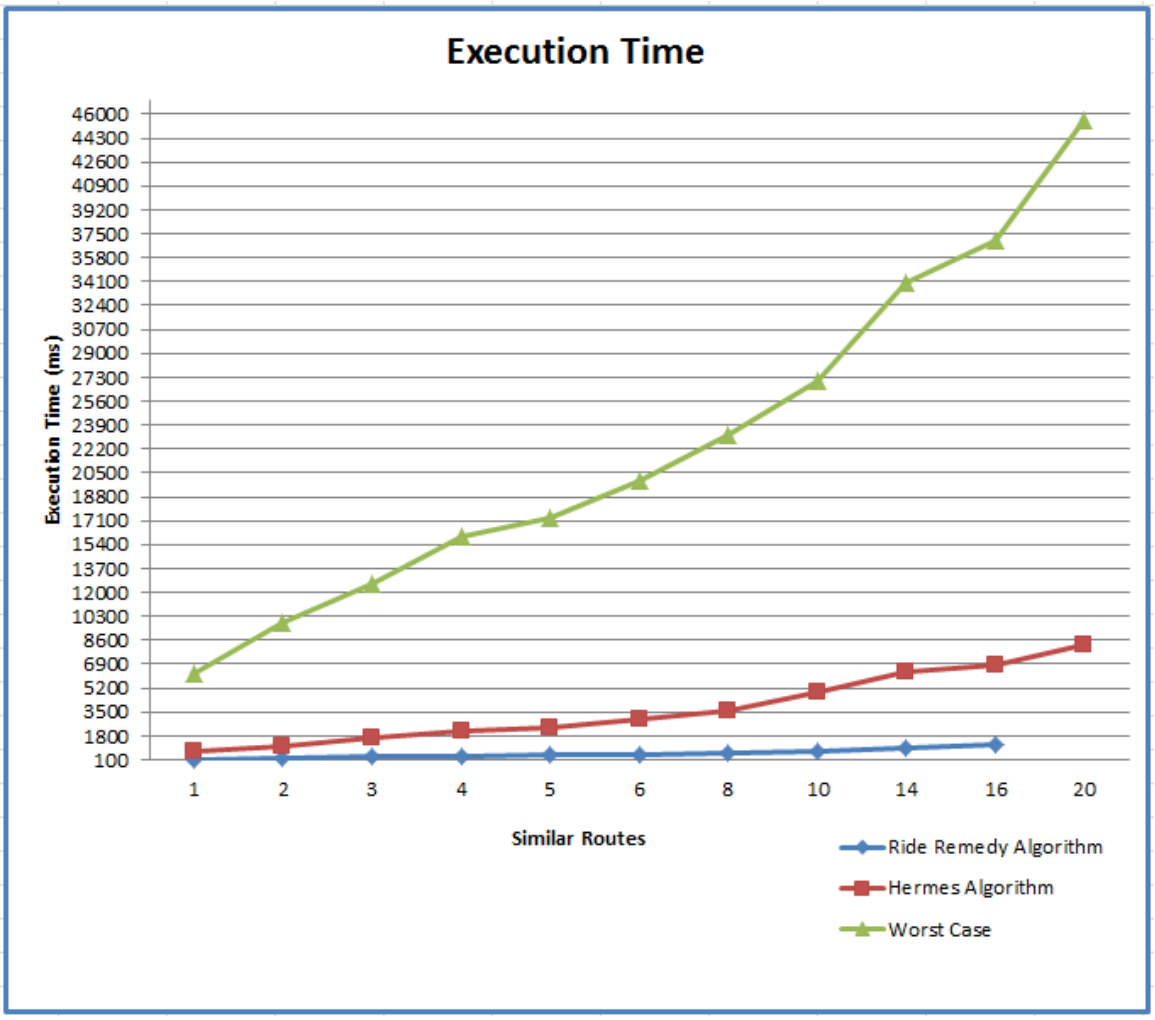
Εικόνα 5.3.6 Γραφική παράσταση κατανομής του χρόνου εκτέλεσης για σχηματισμό του ακριβές δρομολογίου.

Σε αυτή την γραφική παράσταση βλέπουμε την κατανομή του χρόνου για τον αλγόριθμο που μιλήσαμε πιο πάνω. Παρατηρούμε πως η κατανομή του χρόνου χωρίζεται σε δύο τμήματα. Όπως βλέπουμε το 54% κατά μέσο όρο του χρόνου που απαιτείται για να εκτελεστεί ο αλγόριθμος οφείλεται σε Request που γίνονται στο Google Maps όπως και προηγουμένως για να πάρουμε το δρομολόγιο μεταξύ δύο γεωγραφικών σημείων. Επίσης από την πιο πάνω γραφική παράσταση παρατηρούμε πως το υπόλοιπο 45% οφείλεται σε υπολογιστικές πράξεις οι οποίες εκτελούνται για να βρούμε την σειρά των αρχικών και τελικών σημείων με την οποία θα περάσει ο οδηγός για να εξυπηρετήσει τους επιβάτες του.

Τρίτο πείραμα

Το τρίτο πείραμα που εκτελέστηκε αναφέρεται στον αλγόριθμο αναζήτησης που αναλύσαμε στο κεφάλαιο τρία. Σε αυτό το πείραμα ελέγχεται ο χρόνος εκτέλεσης του αλγορίθμου σε διάφορα σεμινάρια όπου έχουμε έναν μεγάλο αριθμό δρομολογίων αποθηκευμένα στη βάση δεδομένων.

Επίσης θα συγκρίνουμε τον αλγόριθμο εύρεσης σχετικών δρομολόγιων με τον αλγόριθμο εύρεσης δρομολογίων που χρησιμοποίει η εφαρμογή Ride Remedy αλλά επίσης και με το χειρότερο σενάριο υλοποίησης του αλγορίθμου.



Εικόνα 5.3.7 Γραφική παράσταση χρόνου εκτέλεσης αλγορίθμου εύρεσης πιθανών δρομολογίων και σύγκριση του με τον αλγόριθμο εύρεσης της εφαρμογής Ride Remedy και με το χειρότερο σενάριο υλοποίησης του.

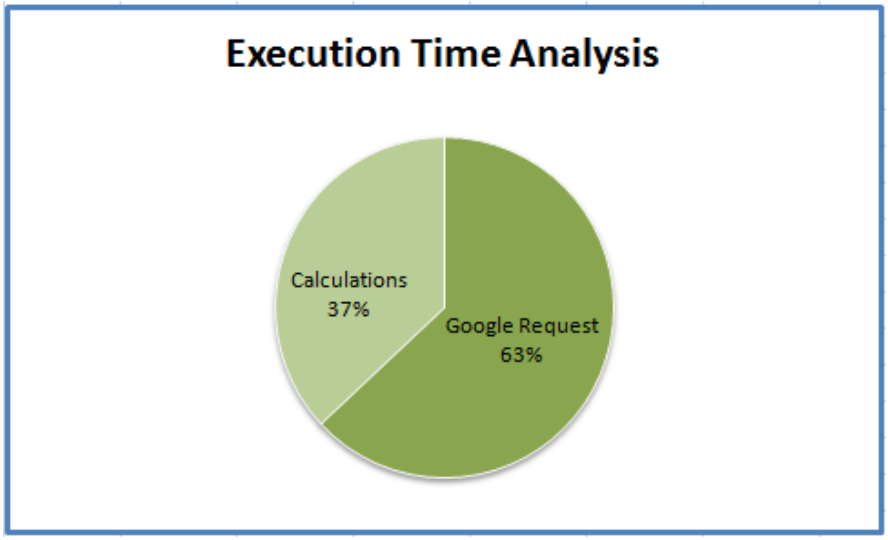
Σχολιασμός αλγορίθμου εύρεσης Hermes

Η πιο πάνω γραφική παράσταση μας δείχνει το χρόνο εκτέλεσης που χρειάζεται ο αλγόριθμος αναζήτησης που επεξηγήσαμε στο κεφάλαιο 3, όπου ένας χρήστης εκτελεί μια αναζήτηση για να βρει τα διαθέσιμα δρομολόγια από τους φίλους του δεδομένου του δρομολογίου που θέλει να εκτελέσει, της ημερομηνίας και ώρας σε συνδυασμό με την ευελιξία χρόνου (Time Flexibility) όπου μπορεί να χρησιμοποιηθεί είτε θετικά είτε αρνητικά στην ώρα που έχει δώσει και την ευελιξία του δρομολογίου (Route Flexibility).

Στον άξονα Χ βλέπουμε τον αριθμό των δρομολογίων που ικανοποιούν τους περιορισμούς του δρομολόγιου, και στον άξονα Υ βλέπουμε τον χρόνο που χρειάστηκε για να εκτελεστεί ο αλγόριθμος αναζήτηση σε κάθε ένα από τα σενάρια. Παρατηρούμε πως ο χρόνος εκτέλεσης εξαρτάται από τον αριθμό των δρομολογίων που ικανοποιούν τον χρήστη. Αυτό το φαινόμενο ήταν αναμενόμενο για το λόγο ότι όσα περισσότερα δρομολόγια ικανοποιούν το χρήστη τόσο περισσότερες υπολογιστικές πράξεις και Request στο Google Maps θα εκτελεστούν.

Να αναφέρουμε πως στην βάση δεδομένων είχαμε αποθηκευμένα 200 δρομολόγια όπου η πλειοψηφία αυτών αποκλείστηκε με την χρήση του αλγορίθμου που καθορίζει την περιοχή κάθε δρομολογίου όπως αναλύθηκε στο κεφάλαιο 3.

Από την γραφική παράσταση παρατηρούμε πως όσο αυξάνονται τα δρομολόγια τα οποία πληρούν τα κριτήρια τα οποία έδωσε ο χρήστης τόσο περισσότερο αυξάνεται και η καθυστέρηση του τερματισμού του αλγόριθμου. Για τον λόγο αυτό θεώρησα χρήσιμο να προσθέσω μια προαιρετική συνθήκη στον αλγόριθμο όπου να δηλώνει τον μέγιστο αριθμό αποτελεσμάτων που θέλουμε να έχει ο αλγόριθμος. Επομένως με την χρήση αυτής της προαιρετικής συνθήκης μπορεί ο χρήστης να δηλώσει πως αναμένει για παράδειγμα το πολύ μέχρι 30 δρομολόγια μετά τον τερματισμό του αλγορίθμου. Φυσικά ο χρήστης μπορεί να δει όλα τα ολοκληρωμένα αποτελέσματα του αλγόριθμου αν το θελήσει χωρίς την χρήση της προαιρετικής συνθήκης.



Εικόνα 5.3.8 Γραφική παράσταση κατανομής του χρόνου εκτέλεσης αλγορίθμου εύρεσης πιθανών δρομολογίων (Hermes Algorithm).

Από την πιο πάνω γραφική παράσταση βλέπουμε την κατανομή του χρόνου για τον αλγόριθμο αναζήτησης όπως έχουμε μιλήσει και πιο πάνω. Παρατηρούμε πως η κατανομή του χρόνου χωρίζεται σε δύο μεγάλα κομμάτια. Όπως βλέπουμε το 63% κατά μέσο όρο του χρόνου που απαιτείται για να εκτελεστεί ο αλγόριθμος οφείλεται σε Request που γίνονται στο Google Maps όπως και προηγουμένως για να πάρουμε το δρομολόγιο μεταξύ δύο γεωγραφικών σημείων και να υπολογίσουμε την πραγματική απόσταση του δρομολογίου. Επίσης παρατηρούμε πως το υπόλοιπο 37% οφείλεται σε υπολογιστικές πράξεις οι οποίες εκτελούνται για να βρούμε κυρίως την πραγματική απόσταση δύο γεωγραφικών σημείων αλλά επίσης και για να βρούμε το σημείο όπου το πρώτο δρομολόγιο θα παρακάμψει το δρομολόγιο του για να εξυπηρετήσει άλλους χρήστες.

Σύγκριση σχετικών αλγορίθμων εύρεσης δρομολογίων

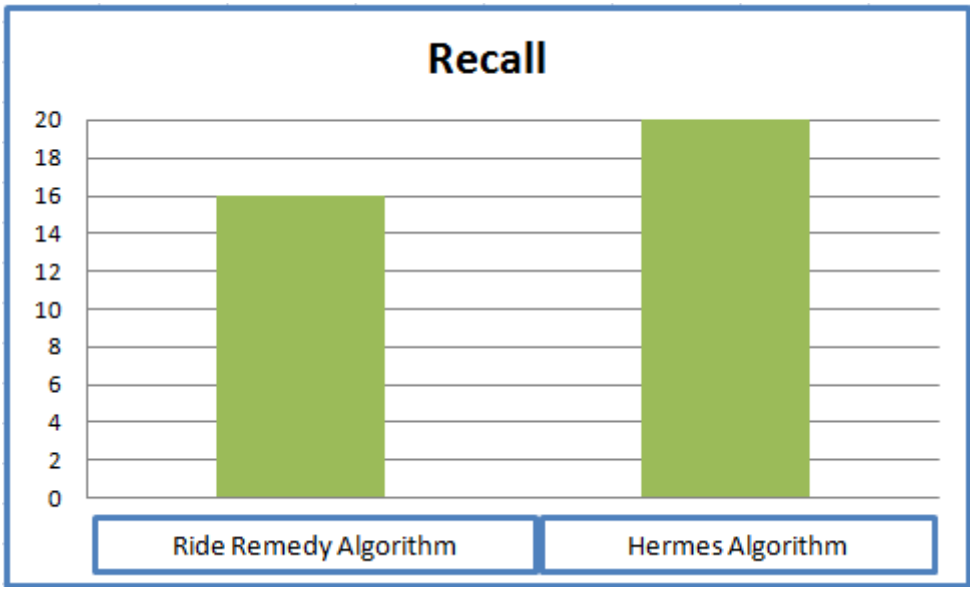
Σε αυτό το σημείο θα συγκρίνουμε τον αλγόριθμο εύρεσης σχετικών δρομολόγιων που υλοποιήσαμε με τον αλγόριθμο εύρεσης που χρησιμοποιεί η εφαρμογή Ride Remedy αλλά επίσης και με το χειρότερο σενάριο υλοποίησης του αλγορίθμου.

1. **Σύγκριση Hermes Algorithm με Ride Remedy Algorithm**

Στην πρώτη σύγκριση που διατρέξαμε συγκρίναμε τον αλγόριθμο που υλοποιήσαμε με τον αλγόριθμο της εφαρμογής Ride Remedy [8], όσο αφορά τον χρόνο εκτέλεσης τους αλλά επίσης και της ανάκλησης τους.

Όπως βλέπουμε στην εικόνα 5.3.7 παρουσιάζονται οι πειραματικές μετρήσεις που πήραμε από τους δύο αλγόριθμους. Παρατηρούμε πως ο αλγόριθμος της εφαρμογής Ride Remedy

έχει πολύ λιγότερο χρόνο εκτέλεσης σε σύγκριση με τον αλγόριθμο όπου υλοποιήσαμε. Ο λόγος για τον οποίο συμβαίνει αυτό είναι γιατί ο αλγόριθμος της εφαρμογής Ride Remedy χρησιμοποίει μόνο υπολογιστικές πράξεις για την ταύτιση δύο δρομολογίων χρησιμοποιώντας το σημείο έναρξης και τερματισμού τους. Δηλαδή δεν εκτελεί καθόλου Request στο Google Maps, το οποίο χρησιμοποιείται για την λειτουργία του αλγορίθμου που αναπτύξαμε.



Εικόνα 5.3.9 Γραφική παράσταση ανάκλησης αλγορίθμου εύρεσης Hermes και Ride Remedy.

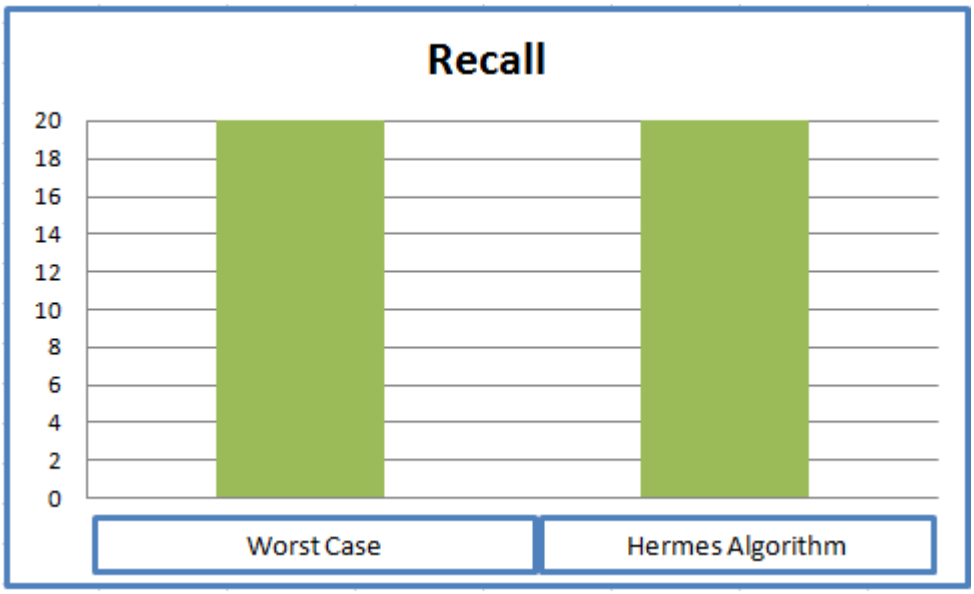
Επίσης από την πιο πάνω γραφική παράσταση παρατηρούμε πως η ανάκληση του αλγορίθμου της εφαρμογής Ride Remedy είναι 16/20 δρομολόγια. Ο λόγος για τον οποίο δεν επιστρέφει 20/20 δρομολόγια είναι γιατί ο αλγόριθμος ταυτίζει δρομολόγια τα οποία έχουν παρόμοιο σημείο έναρξης και τερματισμού σε συνδυασμό με δύο μεταβλητές που δίνει ο χρήστης. Το Departure Distance και Destination Distance, όπου ορίζουν την ακτίνα από το σημείο έναρξης και σημείο τερματισμού αντίστοιχα. Αυτό το χαρακτηριστικό του αλγορίθμου δεν του επιτρέπει να ταυτίσει δρομολόγια τα οποία έχουν εντελώς διαφορετικό σημείο έναρξης και τερματισμού άλλα συναντώνται σε ενδιάμεσο σημείο του δρομολογίου. Για να γίνει αυτό εφικτό θα πρέπει να κάνει Request για να πάρει τα σημεία από τα οποία περνούν τα δύο δρομολόγια και αναλόγως να τα συγκρίνει μεταξύ τους.

2. Σύγκριση Hermes Algorithm με την χειρότερη υλοποίηση του αλγορίθμου

Στη δεύτερη σύγκριση που διατρέξαμε συγκρίναμε τον αλγόριθμο που υλοποιήσαμε με την χειρότερη υλοποίηση του αλγόριθμου, όσον αφορά τον χρόνο εκτέλεσης τους αλλά επίσης και της ανάκλησης τους. Να αναφέρουμε πως στην χειρότερη υλοποίηση της εύρεσης

παίρνουμε το κάθε ένα δρομολόγιο από την βάση δεδομένων και το συγκρίνουμε με το ζητούμενο δρομολόγιο, χωρίς να αποκλείουμε κάποια αρχικά.

Όπως βλέπουμε στην εικόνα 5.3.7 παρουσιάζονται οι πειραματικές μετρήσεις που πήραμε από τις δύο υλοποιήσεις του αλγορίθμου εύρεσης. Παρατηρούμε ότι η χειρότερη υλοποίηση του αλγορίθμου έχει πολύ μεγαλύτερο χρόνο εκτέλεσης σε σύγκριση με τον αλγόριθμο όπου υλοποιήσαμε. Ο λόγος για τον οποίο συμβαίνει αυτό είναι γιατί συγκρίνει όλα τα δρομολόγια που βρίσκονται στην βάση δεδομένων με το ζητούμενο δρομολόγιο. Σε αντίθεση με τον αλγόριθμο που υλοποιήσαμε όπου αποκλείονται αρκετά δρομολόγια βάσει της διαφορετικής περιοχής μεταξύ των δύο δρομολογίων και ελέγχονται μόνο αυτά που αναφέρονται στην ίδια περιοχή.



Εικόνα 5.3.10 Γραφική παράσταση ανάκλησης αλγορίθμου εύρεσης Hermes και Worst case.

Από την πιο πάνω γραφική παράσταση παρατηρούμε πως η ανάκληση και στις δύο υλοποιήσεις του αλγορίθμου είναι η ορθή. Επιστρέφουν και τα δύο 20/20 δρομολόγια που βρίσκονται αποθηκευμένα στην βάση δεδομένων. Όμως διαφέρουν κατά πολύ στον χρόνο εκτέλεσης.

5.4 Ερωτηματολόγιο αξιολόγησης εφαρμογής

Μετά το τέλος του συστήματος και την πρακτική δοκιμή του έχουμε δώσει ερωτηματολόγια σε αυτούς που έχουν λάβει μέρος θέλοντας να εξάγουμε κάποια αποτελέσματα. Με αυτό τον τρόπο θα δούμε πιο εύκολα τις αδυναμίες του συστήματος που χρίζουν διόρθωσης ή βελτίωσης.

Α. Σχετικά με την εφαρμογή

1. Σε τι βαθμό σας άρεσε η εφαρμογή;
   Καθόλου 1 2 3 4 5 Πάρα πολύ

2. Σε τι βαθμό ευχρηστίας κατατάσσετε την εφαρμογή;
   Καθόλου 1 2 3 4 5 Πάρα πολύ

3. Πόσος χρόνος χρειάστηκε για την εξοικείωση σας με την εφαρμογή;
   Ελάχιστος 1 2 3 4 5 Αρκετός

4. Πόσο συχνά θα την χρησιμοποιούσατε;
   Καθόλου 1 2 3 4 5 Πολύ Συχνά

5. Θα προτείνατε την χρήση της εφαρμογής; Ναι ☐ Όχι ☐

6. Θέλετε να αναφέρετε κάποιες βελτιώσεις; Ναι ☐ Όχι ☐
   _____
   _____
   _____

7. Αντιμετωπίσατε κάποιες δυσκολίες; Ναι ☐ Όχι ☐
   _____
   _____
   _____

8. Σχόλια
   _____
   _____
   _____

Β. Δημογραφικά χαρακτηριστικά

9. Φύλο Άντρας ☐ Γυναίκα ☐

10. Δίπλωμα οδήγησης Ναι ☐ Όχι ☐

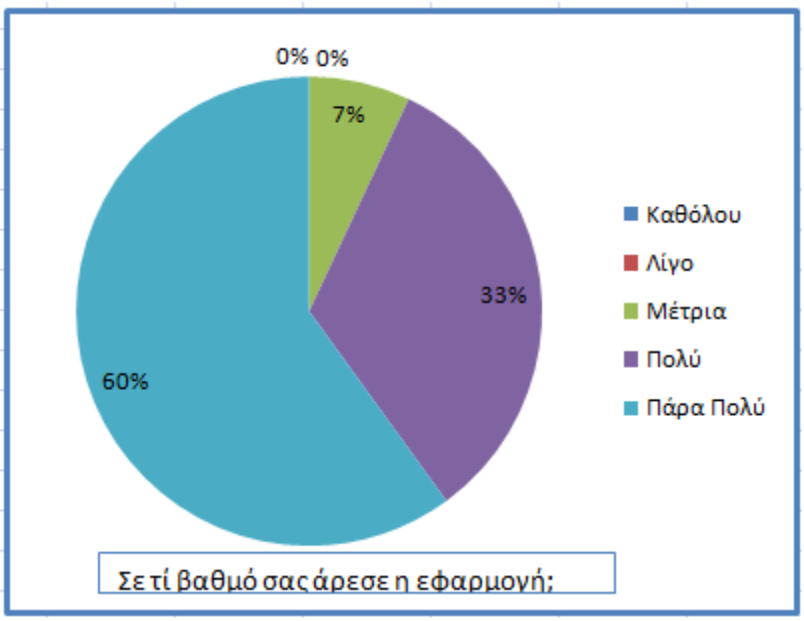
11. Χρήστης Android λειτουργικού Ναι ☐ Όχι ☐

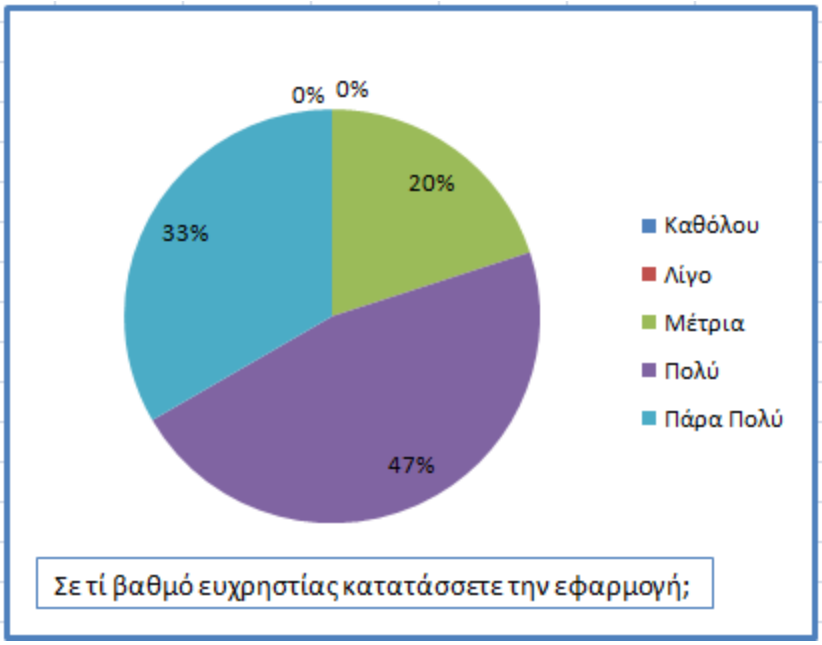
12. Γνώσεις ηλεκτρονικών υπολογιστών
    Καθόλου 1 2 3 4 5 Πάρα πολύ

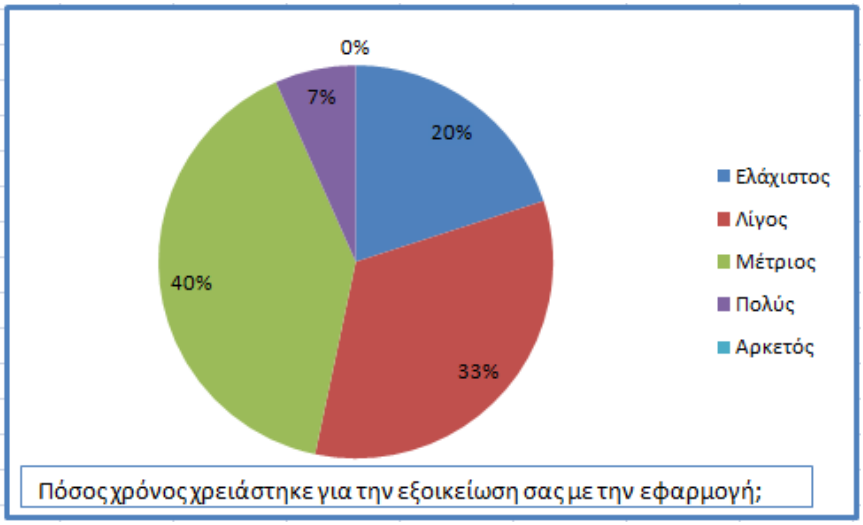
5.4.1 Αποτελέσματα ερωτηματολογίων

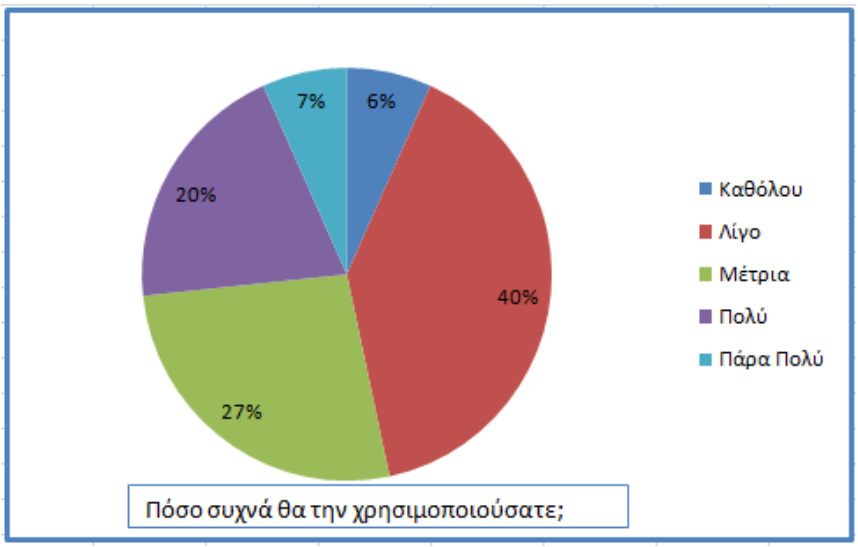
Το σύνολο των χρηστών οι οποίοι χρησιμοποίησαν την εφαρμογή αποτελείτο από 15 άτομα, όπου εκτέλεσαν κάποια σενάρια χρήσης της εφαρμογής όπως:

1. Δημιουργία λογαριασμού στην εφαρμογή.
2. Εύρεσης κάποιου χρήστη από το κοινωνικό δίκτυο.
3. Προσθήκη του χρήστη στους φίλους του.
4. Αποστολή μηνύματος.
5. Προσθήκη νέου δρομολογίου.
6. Εύρεση δρομολογίου και κράτηση θέσης.









Ερωτήσεις 5 - 8

Σχεδόν όλοι οι χρήστες ήταν ικανοποιημένοι από την εφαρμογή και θα πρότειναν την χρήση της. Επίσης από το ερωτηματολόγιο πήραμε κάποιες ιδέες των χρηστών για βελτίωση της εφαρμογής όπως για παράδειγμα:

1. Αν υπάρχει δυνατότητα κάποια πεδία να είναι υπό μορφή επιλογής.
2. Αν υπάρχει η δυνατότητα συνομιλίας μεταξύ των χρηστών (όχι μόνο μέσω μηνυμάτων).
3. Αν υπάρχει η δυνατότητα να προστεθεί η χρήσης φωτογραφίας στα προσωπικά στοιχεία των λογαριασμών των χρηστών.
4. Το Route Flexibility που δίνεται όταν προσθέσουμε ένα δρομολόγιο να μπορεί να αποθηκεύεται και να μην το δίνουμε κάθε φορά.

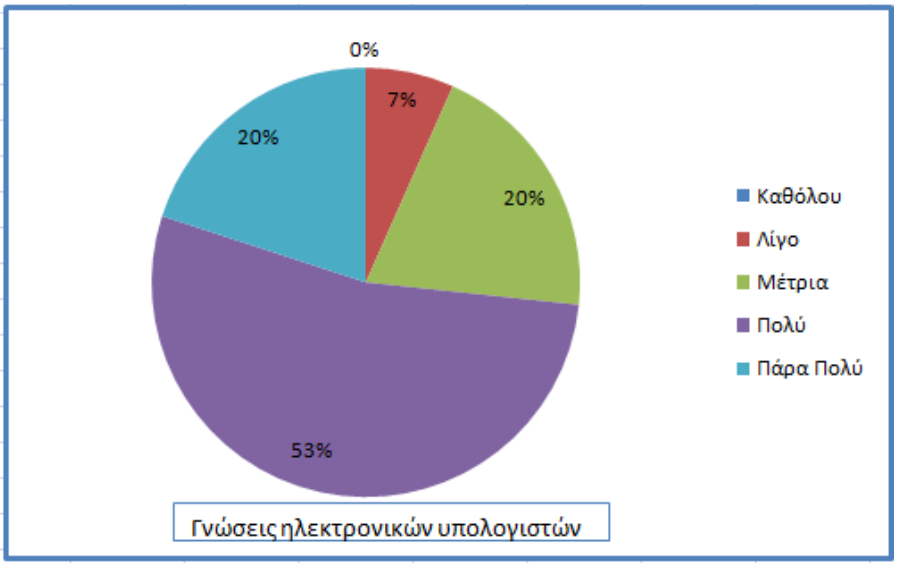
Οι δυσκολίες που αντιμετωπίσθηκαν ήταν κυρίως από χρήστες οι όποιοι δεν χρησιμοποίησαν στον παρελθόν κινητή συσκευή με Android λειτουργικό και είναι οι ακόλουθες:

1. Δυσκολία εύρεσης του μενού της εφαρμογής, όπου πρέπει ο χρήστης να πατήσει το κουμπί menu που διαθέτουν οι κινητές συσκευές με Android λειτουργικό.
2. Δυσκολία εύρεσης τον ειδοποιήσεων (Notifications) όπου βρίσκονται στο Notification Area της κινητής συσκευής.
3. Δυσκολία χρήσης του κουμπιού back που διαθέτει η κινητή συσκευή για να μεταφερθούν στην προηγούμενη οθόνη.

Τα σχόλια που λάβαμε για την εφαρμογή είναι κυρίως ενθαρρυντικά και κάποιοι χρήστες ανάφεραν, αν γίνεται να υπάρχει η χρήση της και από άλλες κινητές συσκευές με διαφορετικό λειτουργικό.

Ερωτήσεις 9 – 11

Το σύνολο των ατόμων αποτελείτο από 12 άντρες και 3 γυναίκες. Επίσης τα 12 άτομα κατέχουν δίπλωμα οδήγησης, και 4 άτομα είναι χρήστες Android λειτουργικού.



Κεφάλαιο 6

Σύνοψη

6.1 Σύνοψη

Στα πλαίσια της διπλωματικής εργασίας ασχοληθήκαμε με την υλοποίησης μιας εφαρμογής Carpooling. Πρώτα μελετήσαμε παρόμοιες εφαρμογές, όπου και καταγράφηκαν από αυτές τα σημαντικότερα χαρακτηριστικά τους τα οποία μας βοήθησαν στην δημιουργία της δικής μας εφαρμογής. Επίσης μελετήσαμε τον τρόπο που υλοποιούν τον αλγόριθμο εύρεσης δρομολογίων.

Μετά την μελέτη των παρόμοιων εφαρμογών αναπτύξαμε την δική μας εφαρμογή Hermes,η οποία είναι ειδικευμένη για κινητή συσκευή με Android λειτουργικό. Η εφαρμογή αυτή έχει τα ακόλουθα χαρακτηριστικά:

1. Είναι βασισμένη σε δικό της κοινωνικό δίκτυο, όπου επιτρέπει στους χρήστες της να ανταλλάσουν μηνύματα με τους φίλους τους, αλλά επίσης και να συμμετέχουν σε δρομολόγια τα οποία προστέθηκαν από φίλους τους.
2. Πέραν του κοινωνικού δικτύου η εφαρμογή παρέχει την δυνατότητα στους χρήστες της να μπορούν να φέρουν τους φίλους τους από το κοινωνικό δίκτυο Facebook, όπου αν κάποιοι από τους φίλους τους έχουν λογαριασμό στην εφαρμογή θα προστεθούν.
3. Η εφαρμογή αμείβει τους χρήστες της οι οποίοι προσθέτουν δρομολόγια και τα ικανοποιούν τις ανάγκες άλλων χρηστών. Η αμοιβή αυτή είναι ανάλογη της απόστασης, αλλά επίσης και των επιβατών του δρομολογίου.
4. Επίσης η εφαρμογή δίνει την δυνατότητα στους χρήστες της να μπορούν να αφήσουν Feedback για την υπηρεσία που τους παρείχε ένας άλλος χρήστης. Έτσι οι χρήστες θα μπορούν να δουν την αξιοπιστία κάποιου άλλου χρήστη.

5. Δίνεται η δυνατότητα δυναμικής εύρεσης δρομολογίων τα οποία έχουν ξεκινήσει την εκτέλεση τους.
6. Τέλος αναπτύξαμε ένα αλγόριθμο εύρεσης πιθανών δρομολογίων ο οποίος συνδυάζει τα χαρακτηριστικά του κάθε αποθηκευμένου δρομολογίου που βρίσκεται στην βάση δεδομένων, και επιστρέφει στον χρήστη όμοια δρομολόγια με το δρομολόγιο που διέτρεξε την αναζήτηση.

Μετά την υλοποίηση της εφαρμογής Hermes διατρέξαμε κάποιες πειραματικές μετρήσεις, όπου ελέγχθηκε κυρίως ο χρόνος απόκρισης κρίσιμων σημείων της εφαρμογής. Επίσης συγκρίναμε τον αλγόριθμο εύρεσης που αναπτύχθηκε με όμοιο αλγόριθμο που χρησιμοποιεί η εφαρμογή Ride Remedy, και με το χειρότερο σενάριο υλοποίησης του αλγορίθμου. Συγκρίναμε τον αλγόριθμο όσον αφορά τον χρόνο απόκρισης του αλλά επίσης και της ανάκλησης που είχαμε σε κάθε ένα σενάριο ξεχωριστά. Από τις μετρήσεις που διατρέξαμε καταλήξαμε στο συμπέρασμα ότι ο αλγόριθμος μας είχε μεγαλύτερο χρόνο εκτέλεσης αλλά η ανάκληση του ήταν η σωστή και κάλυπτε όλα τα πιθανά σενάρια.

Επίσης για την σωστή αξιολόγηση της εφαρμογής δημιουργήσαμε ένα ερωτηματολόγιο το οποίο δώσαμε σε ένα σύνολο χρηστών. Τα αποτελέσματα των ερωτηματολογίων είναι κυρίως θετικά, και οι χρήστες οι οποίοι χρησιμοποίησαν την εφαρμογή έμειναν απολύτως ικανοποιημένοι. Επίσης λάβαμε από αυτούς θετικά σχόλια αλλά και ιδέες για βελτιστοποίηση και προέκταση της εφαρμογής σε κάποια σημεία.

6.3 Μελλοντική εργασία

Σε αυτό το υποκεφάλαιο θα δούμε πως μπορούμε να επεκτείνουμε την εφαρμογή που δημιουργήσαμε στο μέλλον, ούτως ώστε να διευρύνουμε το σύνολο των χρηστών που μπορεί να έχει.

Ιδέες για μελλοντική εργασία:

1. Δημιουργία ιστοσελίδας όπου οι χρήστες θα μπορούν να συνδεθούν με το λογαριασμό τους και να προσθέσουν δρομολόγια, να αναζητήσουν δρομολόγια από τους φίλους τους βάσει κάποιων κριτηρίων, να κάνουν κράτηση σε κάποιο δρομολόγιο και να συνομιλήσουν με άλλους χρήστες.

2. Τροποποίηση της εφαρμογής Hermes για την λειτουργία της σε κινητές συσκευές με διαφορετικό λειτουργικό όπως για παράδειγμα σε Apple, Symbian, Windows Mobile κτλ.
3. Σύνδεση και άλλων κοινωνικών δικτύων με την εφαρμογή εκτός από το Facebook όπου υπάρχει ήδη.
4. Δημιουργία δυνατότητας για τους χρήστες οι οποίοι θέλουν να εκτελέσουν παρόμοια ή ίδια δρομολόγια να χρησιμοποιήσουν το ίδιο ταξί.

Βιβλιογραφία

[1] Satya Komatineni, Dave Maclean and Sayed Hashimi, “Android 3 platform SDK techniques for developing Smartphone and tablet apps”.

[2] George Pallis, Demetrios Zeinalipour–Yazti, Marios D. Dikaiakos, “Online Social Networks: Status and Trends”.

[3] Anna-Kaisa Pietilainen, Earl Oliver, Jason LeBrun, “MobiClique: Middleware for Mobile Social Networking”.

[4] Stephen Smaldone, Lu Han, Pravin Shankar, Liviulftode, “RoadSpeak: Enabling Voice Chat on Roadways using Vehicular Social Networks”.

[5] Canalys, “http://www.canalys.com/pr/2011/r2011013.html”.

[6] Country Health Rankings, “http://www.countyhealthrankings.org/new-york/bronx/67”

[7] Χάρης Ευσταθιάδης, “Feel The World Framework”.

[8] Ride Remedy, “http://www.rideremedy.com”.

[9] carpoolWorld, “http://www.carpoolworld.com”.

[10] Android Developers, “http://developer.android.com/index.html”.

[11] Email Marketing Reports, “http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm”.

[12] Carpool Hunter, “http://video.google.com/videoplay?docid=7731234258848132620#”.

[13] ZimRide, “http://www.zimride.com/”.

[14] Movable Type Scripts. “http://www.movable-type.co.uk/scripts/latlong.html”.

[15] Android Developers, “http://developer.android.com/index.html”.

[16] Mark L. Murphy, “Beginning Android 2”.

[17] Jerome (J. F.) DiMarzio, “Android A Programmer’s Guide”.

[18] Jeff Friesen, “Learn Java For Android Development”.

[19] Exploring Java Chapter 6 Threads,
“http://oreilly.com/catalog/expjava/excerpt/index.html”.

[20] Sockets: Basic Client-Server Programming in Java,
“http://edn.embarcadero.com/article/31995”.

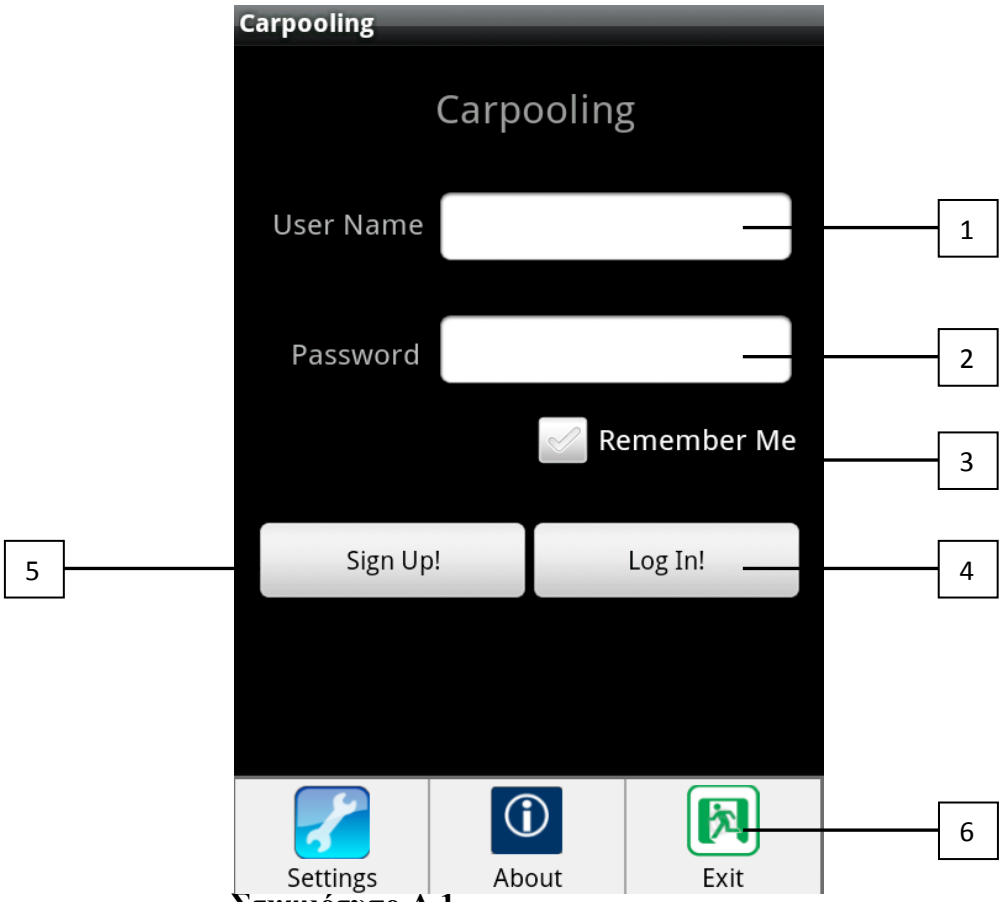
[21] HTC, “http://www.htc.com/europe/product/desire/overview.html”.

[22] Intel, “http://ark.intel.com/Product.aspx?id=33082”.

[23] The international conference on Mobile Systems, Applications and services,
“http://www.sigmobile.org/mobisys/”.

Παράρτημα Α

Σε αυτό το παράρτημα θα παρουσιάσουμε κάποια στιγμιότυπα της εφαρμογής και θα μιλήσουμε για την χρησιμότητα και τις λειτουργίες τους.

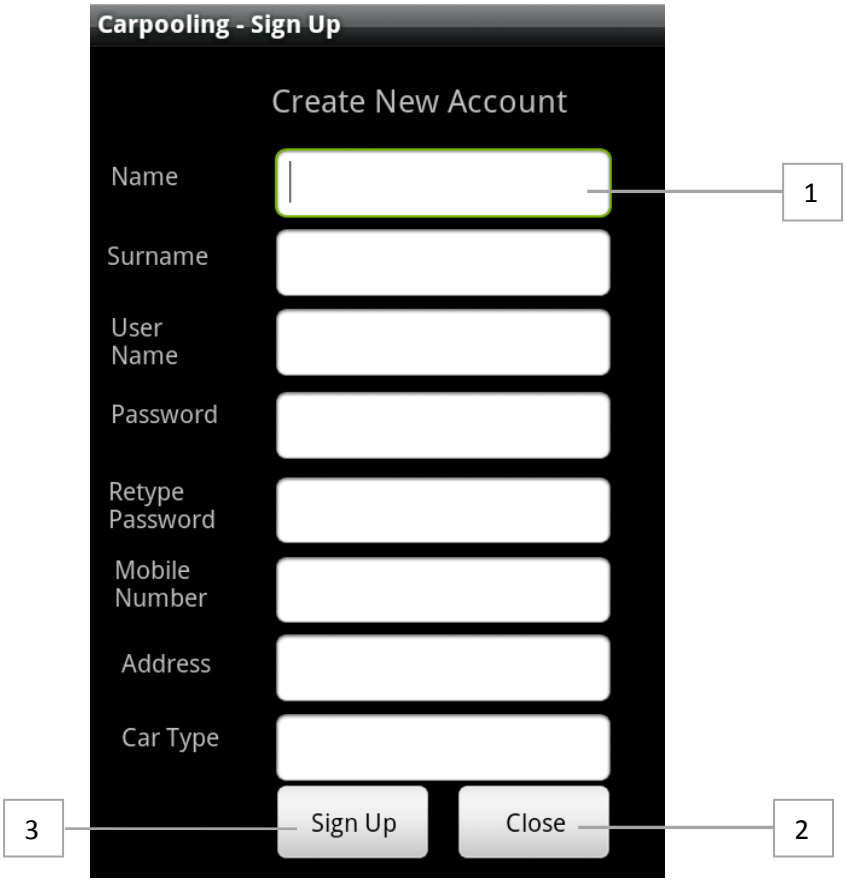


Στιγμιότυπο Α.1

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη μόλις τρέξει την εφαρμογή και έχει τα ακόλουθα χαρακτηριστικά:

1. Εισαγωγή του User Name.
2. Εισαγωγή του Κωδικού πρόσβασης .
3. Επιλογή αποθήκευσης του User Name.
4. Γίνεται έλεγχος του User Name και του κωδικού και αν είναι ορθά ο χρήστης συνδέεται με την εφαρμογή.
5. Ο χρήστης ο οποίος δεν έχει λογαριασμό μπορεί μέσω αυτού να οδηγηθεί σε μια άλλη οθόνη και να δημιουργήσει καινούριο λογαριασμό.
6. Οι επιλογές αυτές εμφανίζονται μετά το πάτημα του κουμπιού menu που διαθέτουν οι κινητές συσκευές με Android λειτουργικό. Στο πιο πάνω μενού έχουμε τρεις επιλογές:
    i. Να δούμε τις ρυθμίσεις της εφαρμογής.
    ii. Να διαβάσουμε κάποια λόγια για την εφαρμογή.

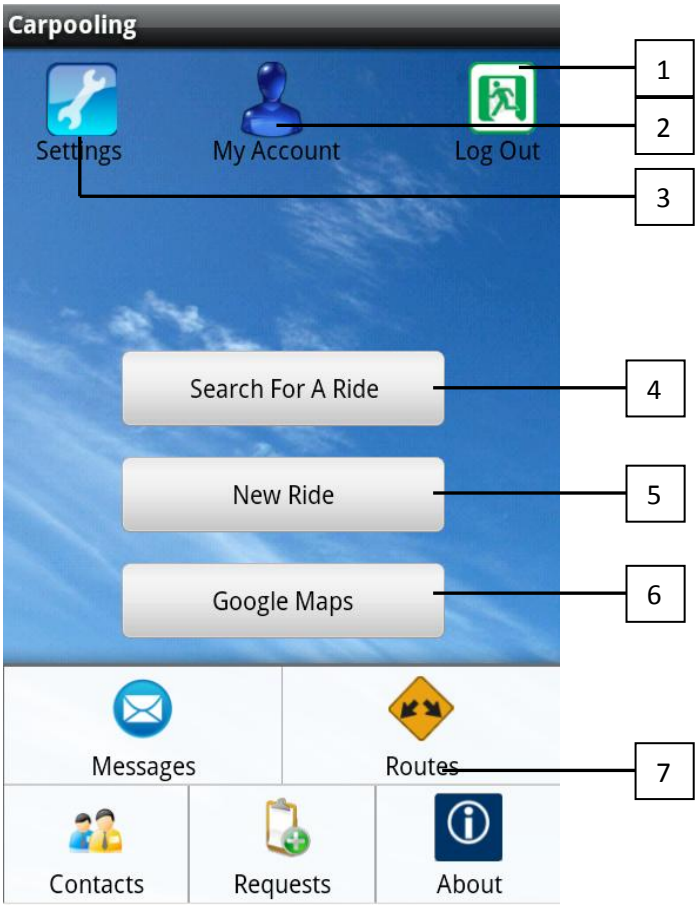
iii. Να κλείσουμε την εφαρμογή.



Στιγμιότυπο Α.2

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη μόλις πατήσει το κουμπί Sign Up και έχει τα ακόλουθα χαρακτηριστικά:

1. Εισαγωγή προσωπικών στοιχείων που απαιτούνται από την εφαρμογή όπως για παράδειγμα Όνομα, Επίθετο, User Name, Κωδικός πρόσβασης, Αριθμός κινητού τηλεφώνου κτλ.
2. Με το πάτημα του κουμπιού Close τερματίζεται η διαδικασία χωρίς να γίνει εγγραφή του νέου χρήστη.
3. Με το πάτημα του κουμπιού Sign Up πρώτα ελέγχεται αν έχουν συμπληρωθεί όλα τα πεδία εκτός του Car Type το οποίο δεν είναι υποχρεωτικό, ακολούθως ελέγχεται ο κωδικός πρόσβασης να είναι τουλάχιστο 4 ψηφία για λόγους ασφαλείας και να είναι ο ίδιος με την επανατύπωση του κωδικού, τέλος ελέγχεται το User Name το οποίο πρέπει να είναι πρότυπο στην βάση δεδομένων.



Στιγμιότυπο Α.3

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη μόλις συνδεθεί με την εφαρμογή και έχει τα ακόλουθα χαρακτηριστικά:

1. Με το πάτημα του κουμπιού Log Out ο χρήστης αποσυνδέεται από τον λογαριασμό του και γίνεται αποθήκευση της ημερομηνίας και της ώρας για σκοπούς ασφαλείας.
2. Με το πάτημα του κουμπιού My Account ο χρήστης βλέπει τα χαρακτηριστικά του λογαριασμού του και τα προσωπικά του στοιχεία, και μπορεί να τα τροποποιήσει.
3. Με το πάτημα του κουμπιού Settings ο χρήστης μπορεί να τροποποιήσει κάποιες διαθέσιμες ρυθμίσεις της εφαρμογής.
4. Με το πάτημα του κουμπιού Search For A Ride ο χρήστης μπορεί να αναζητήσει διαθέσιμα δρομολόγια από τις επαφές του βάσει κάποιων χαρακτηριστικών.
5. Με το πάτημα του κουμπιού New Ride ο χρήστης μπορεί να προσθέσει ένα νέο δρομολόγιο το οποίο θα είναι διαθέσιμο στους φίλους του.
6. Με το πάτημα του κουμπιού Google Maps ο χρήστης μπορεί να έχει άμεση πρόσβαση σε χάρτες που προσφέρονται από το Google Maps.
7. Οι επιλογές αυτές εμφανίζονται μετά το πάτημα του κουμπιού menu. Στο πιο πάνω μενού έχουμε πέντε επιλογές:

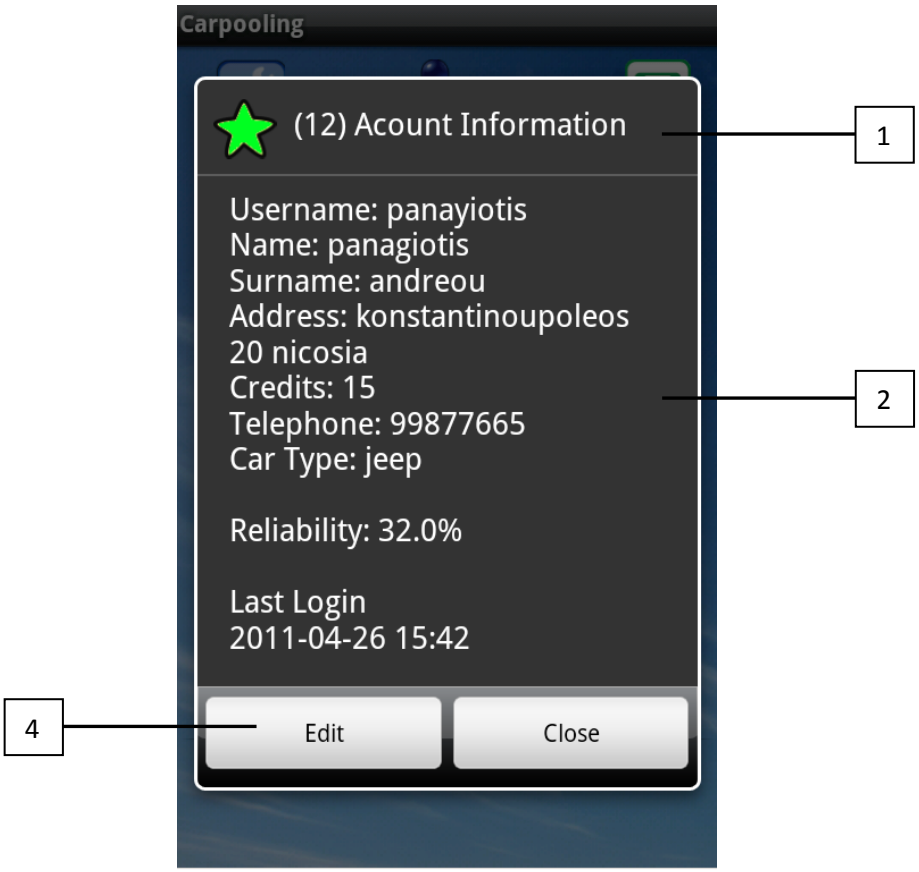
i. Ο χρήστης να δει τα μηνύματα του.

ii. Ο χρήστης να δει τα δρομολόγια που διατίθενται από τους φίλους του ή να δει τα δρομολόγια τα οποία πρόσθεσε ο ίδιος.

iii. Ο χρήστης να δει τους φίλους του.

iv. Ο χρήστης να δει αν έχει κάποιο νέο Request ή να δει τα Request που έχει στείλει και είναι αναπάντητα.

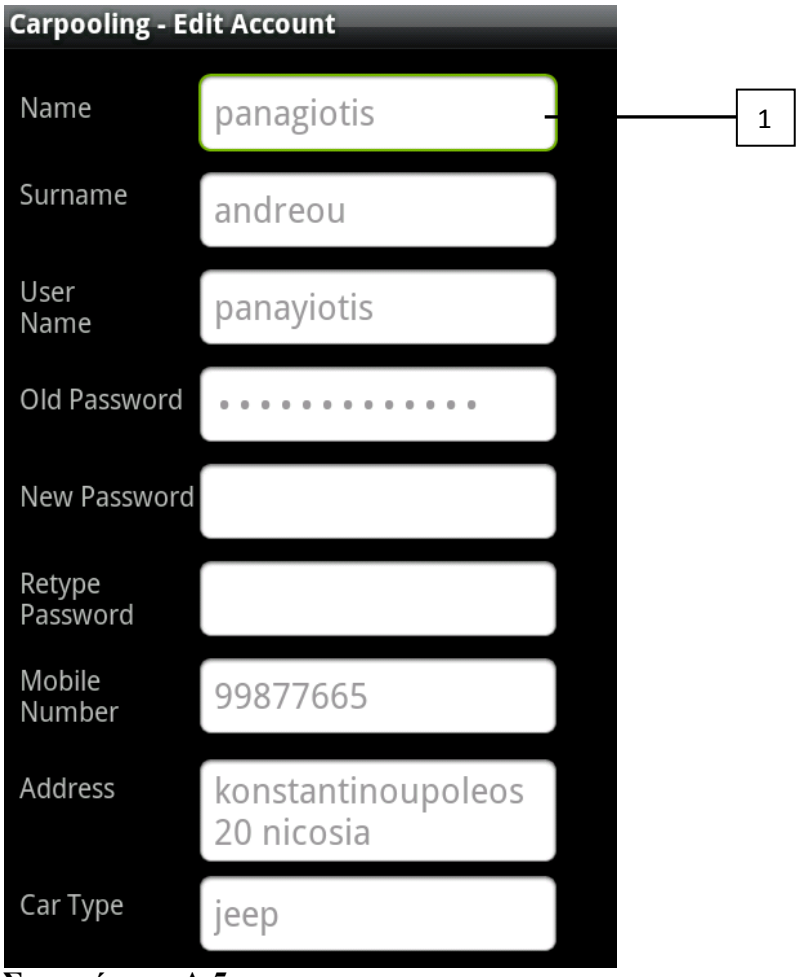
v. Ο χρήστης να διαβάσει κάποια λόγια για την εφαρμογή.



Στιγμιότυπο Α.4

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν πατήσει το κουμπί My Account και έχει τα ακόλουθα χαρακτηριστικά:

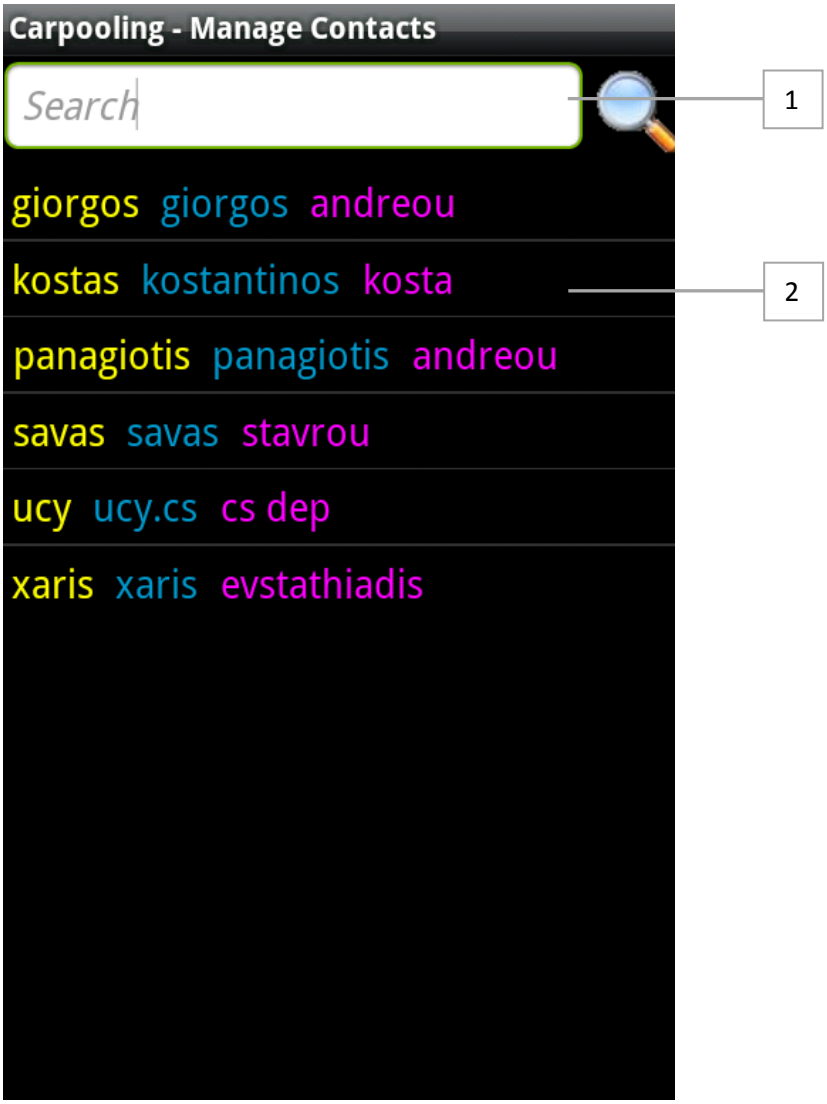
1. Ο χρήστης βλέπει τον αριθμό των δρομολογίων τα οποία εκτέλεσε, και αναλόγως του δίνεται το κατάλληλο αστέρι.
2. Ο χρήστης βλέπει τα προσωπικά του στοιχεία όπως Όνομα, Επίθετο, Διεύθυνση κτλ. Επίσης βλέπει την αξιοπιστία του (Reliability) σε ποσοστό επί τις εκατό και την τελευταία ημερομηνία που χρησιμοποίησε την εφαρμογή (Last Login) για λόγους ασφαλείας.
3. Με το πάτημα του κουμπιού Edit εμφανίζεται στο χρήστη μια νέα οθόνη όπου μπορεί να τροποποιήσει τα προσωπικά του στοιχεία.



Στιγμιότυπο Α.5

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν πατήσει το κουμπί Edit και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης βλέπει τα προσωπικά στοιχεία του λογαριασμού του, και μπορεί να τα τροποποιήσει αναλόγως, και αν είναι έγκυρες οι τροποποιήσεις του θα αποθηκευτούν. Επίσης ο χρήστης μπορεί να τερματίσει αυτή την λειτουργία χωρίς να κάνει οποιαδήποτε αλλαγή.



Στιγμιότυπο Α.6

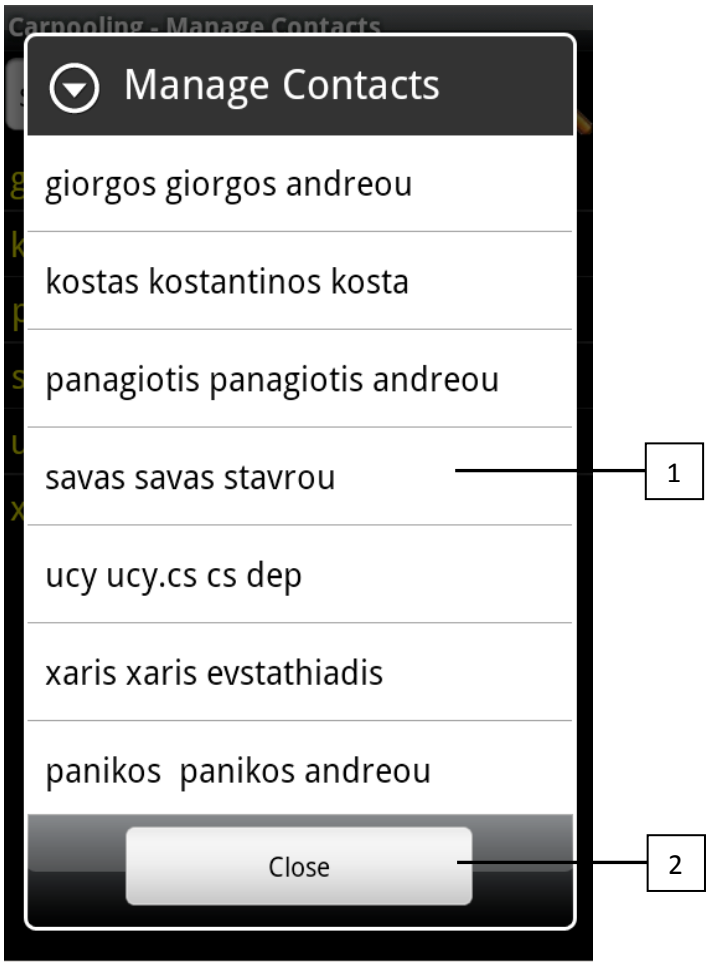
Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν πατήσει το κουμπί Contacts και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης μπορεί να εκτελέσει αναζήτηση σε όλους τους χρήστες που είναι αποθηκευμένοι στην βάση δεδομένων με οποιοδήποτε χαρακτηριστικό γνώρισμα.
2. Ο χρήστης μπορεί να δει όλους τους φίλους που έχει στον λογαριασμό του με αλφαβητική σειρά. Βλέπει το User Name του, το όνομα και το επίθετο του κάθε χρήστη με αυτή την σειρά. Επίσης ο χρήστης μπορεί να πατήσει πάνω σε ένα φίλο του και ακολούθως να δει περαιτέρω πληροφορίες σχετικά με τον συγκεκριμένο χρήστη.

Στιγμιότυπο Α.7

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν πατήσει ένα από τους χρήστες που έχει στο λογαριασμό του και έχει τα ακόλουθα χαρακτηριστικά:

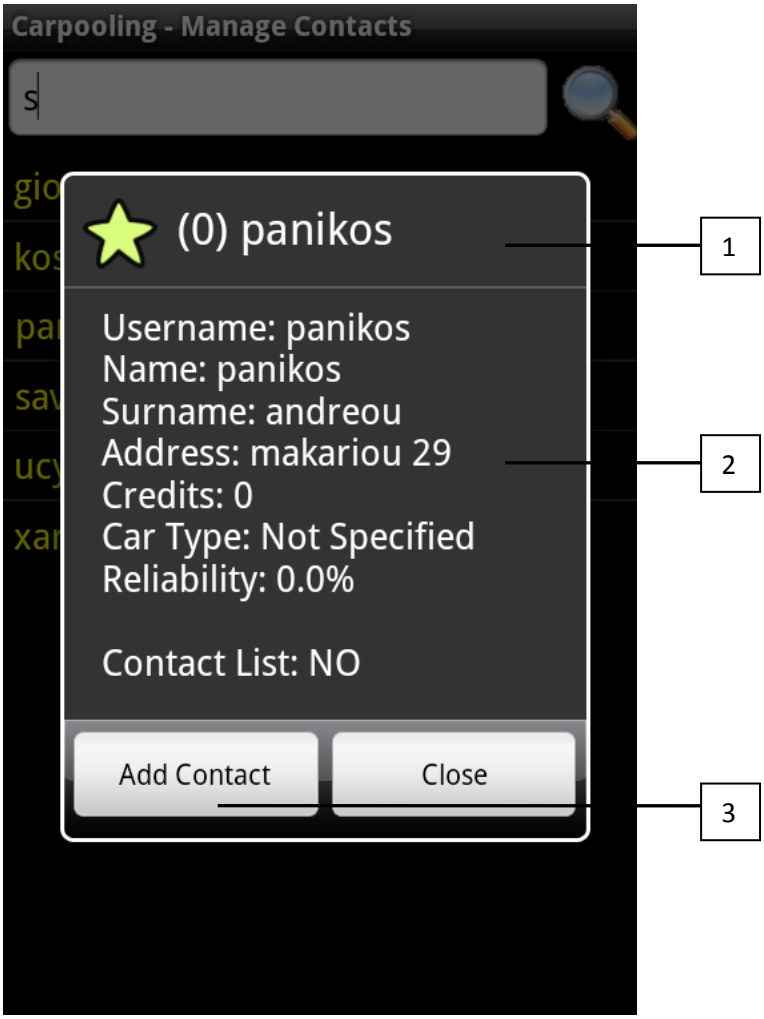
1. Ο χρήστης βλέπει το User Name του φίλου του που επέλεξε και βλέπει αριθμητικά τα δρομολόγια τα οποία εκτέλεσε, όπου παίρνει και τα ανάλογο αστέρι.
2. Ο χρήστης βλέπει κάποια προσωπικά στοιχεία του φίλου του όπως Όνομα, Επίθετο, Διεύθυνση, είδος του αυτοκινήτου του, την αξιοπιστία του και αν είναι στη λίστα με τους φίλους του.
3. Πατώντας το κουμπί Send Message ο χρήστης μπορεί να στείλει μήνυμα στον συγκεκριμένο φίλο του και θα οδηγηθεί σε μια άλλη οθόνη.
4. Πατώντας το κουμπί Delete User ο χρήστης μπορεί να διαγράψει την συγκεκριμένη επαφή από τους φίλους του.



Στιγμιότυπο Α.8

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν εκτελέσει μια αναζήτηση ενός χρήστη βάσει οπουδήποτε χαρακτηριστικού γνωρίσματος και έχει τα ακόλουθα χαρακτηριστικά:

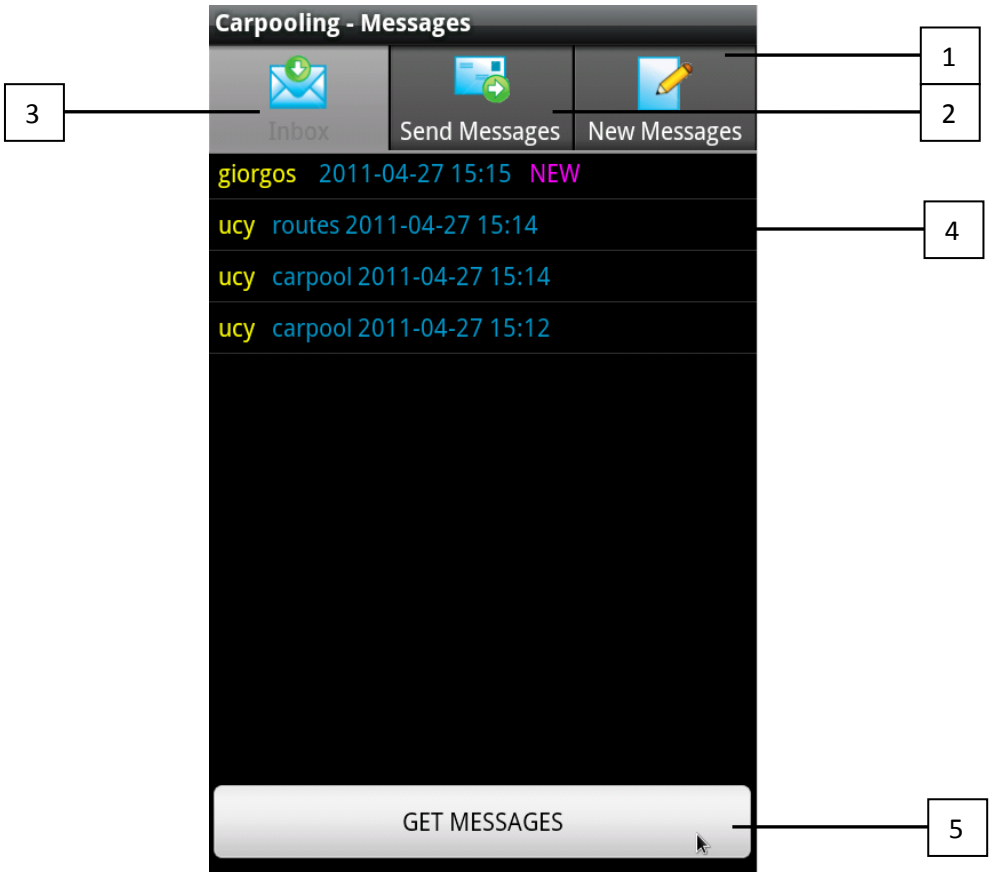
1. Ο χρήστης μπορεί να δει τα αποτελέσματα της αναζήτησης όπου βλέπει το User Name το όνομα και το επίθετο του συγκεκριμένου χρήστη. Επίσης οι χρήστες εμφανίζονται με αλφαβητική σειρά και πρώτοι εμφανίζονται οι χρήστες οι οποίοι είναι στους φίλους του συγκεκριμένου χρήστη και ακολούθως οι χρήστες οι οποίοι δεν είναι στους φίλους του. Ο χρήστης μπορεί να πατήσει πάνω σε ένα άλλο χρήστη και να εμφανιστούν σε νέα οθόνη περαιτέρω πληροφορίες και δυνατότητες.
2. Πατώντας το κουμπί Close ο χρήστης μπορεί να τερματίσει την διαδικασία.



Στιγμιότυπο Α.9

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει ένα από τα αποτελέσματα της αναζήτησης που εκτέλεσε στο προηγούμενο βήμα και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης βλέπει το User Name του συγκεκριμένου χρήστη που επέλεξε, τα δρομολόγια τα οποία εκτέλεσε αριθμητικά και το σχετικό αστέρι που αντιστοιχεί.
2. Ο χρήστης βλέπει διάφορα στοιχεία του χρήστη που επέλεξε όπως User Name, Όνομα, Επίθετο, Διεύθυνση κτλ. Επίσης ο χρήστης μπορεί να δει την αξιοπιστία του συγκεκριμένου χρήστης όπως επίσης και αν βρίσκεται στην λίστα των φίλων του. Ο συγκεκριμένος χρήστης δεν βρίσκεται στην λίστα των φίλων του χρήστη και για αυτό μπορεί να τον προσθέσει αν το θέλει. Αν ο χρήστης ο οποίος επέλεγε βρισκόταν στην λίστα των φίλων του θα μπορούσε να του στείλει μήνυμα ή να τον διαγράψει.
3. Πατώντας το κουμπί Add Contact ο χρήστης μπορεί να στείλει αίτημα στον συγκεκριμένο χρήστη για να τον προσθέσει στην λίστα με τους φίλους του.



Στιγμιότυπο Α.10

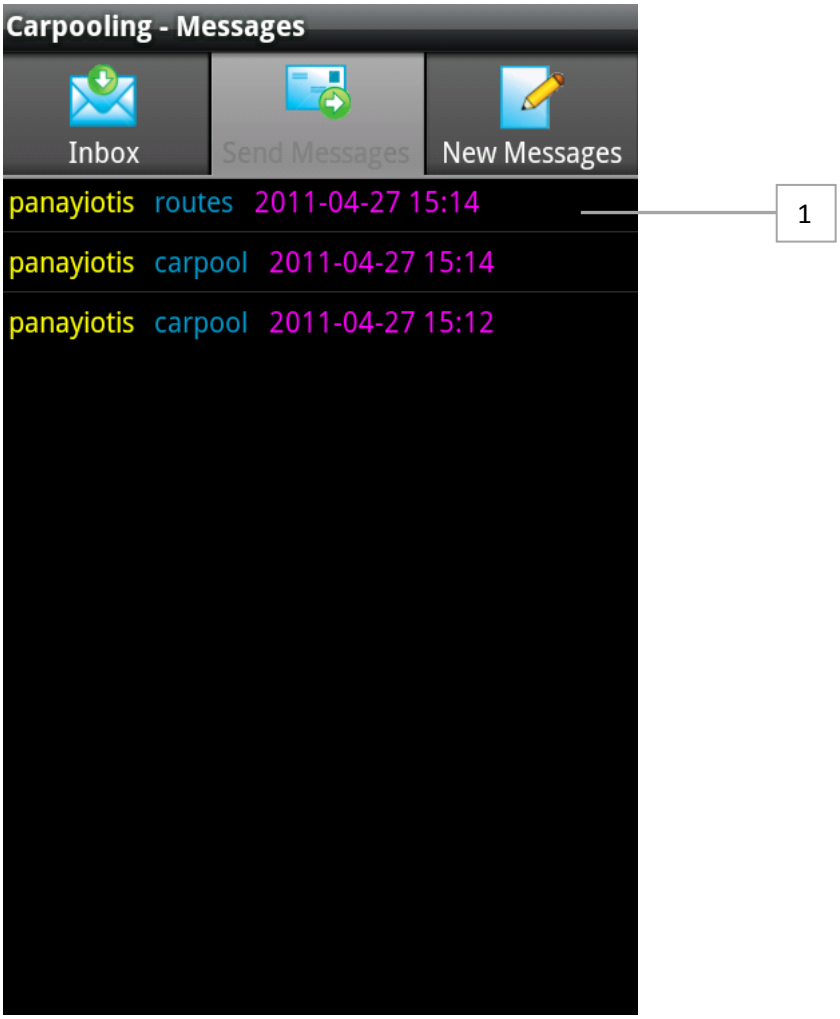
Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα μηνύματα του και έχει τα ακόλουθα χαρακτηριστικά:

1. Πατώντας το tab New Message μπορεί να γράψει ένα νέο μήνυμα και να το στείλει σε έναν από του φίλους του.
2. Πατώντας το tab Send Message μπορεί να δει τα εξερχόμενα μηνύματα του.
3. Πατώντας το tab Inbox μπορεί να δει τα εισερχόμενα μηνύματα του όπως στην οθόνη που βλέπουμε πιο πάνω.
4. Ο χρήστης βλέπει τα εισερχόμενα μηνύματα του, όπου βλέπει το User Name του χρηστή που του έχει στείλει το μήνυμα, το θέμα, αν υπάρχει, και την ημερομηνία και ώρα που στάλθηκαν. Επίσης τα μηνύματα τα οποία δεν έχουν διαβαστεί έχουν δίπλα τους την επιγραφή NEW.
5. Ο χρήστης πατώντας το κουμπί Get Messages μπορεί να ενημερώσει τα εισερχόμενα μηνύματα του αν υπάρχουν νέα μηνύματα στο λογαριασμό του.

Στιγμιότυπο Α.11

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να διαβάσει ένα από τα εισερχόμενα μηνύματα του και έχει τα ακόλουθα χαρακτηριστικά:

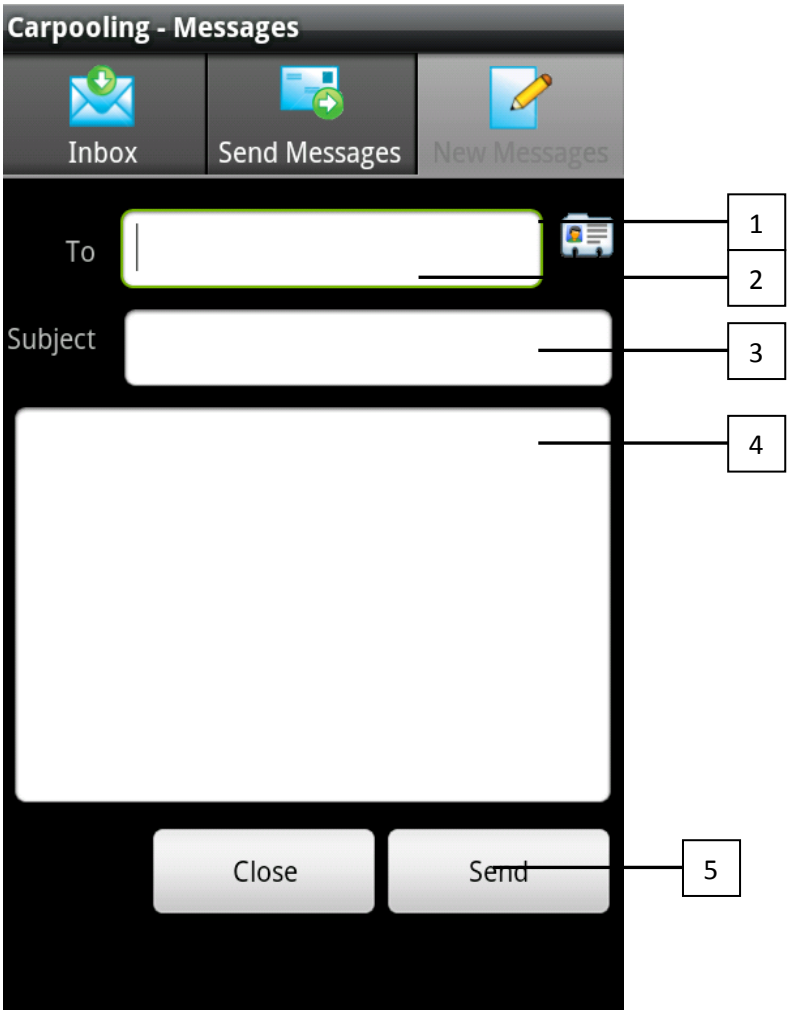
1. Ο χρήστης βλέπει τον αποστολέα του μηνύματος.
2. Ο χρήστης βλέπει το θέμα του μηνύματος αν υπάρχει.
3. Ο χρήστης βλέπει το μήνυμα που του έχει αποσταλεί.
4. Πατώντας το κουμπί Reply ο χρήστης θα οδηγηθεί σε άλλη οθόνη για να απαντήσει στο μήνυμα του φίλου του.
5. Πατώντας το κουμπί Delete ο χρήστης μπορεί να διαγράψει από τον λογαριασμό του το συγκεκριμένο μήνυμα.



Στιγμιότυπο Α.12

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα εξερχόμενα μηνύματα του και έχει τα ακόλουθα χαρακτηριστικά:

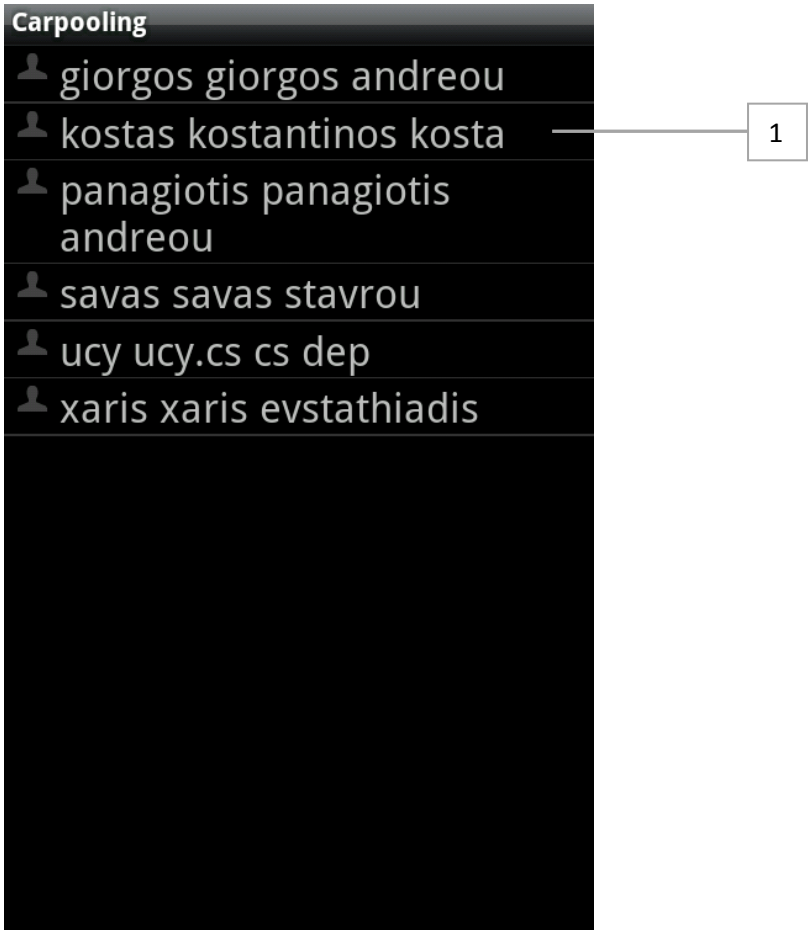
1. Ο χρήστης βλέπει τα εξερχόμενα μηνύματα του. Μπορεί να δει τον χρήστη στον οποίο έχει στείλει το μήνυμα, το θέμα του μηνύματος και την ημερομηνία που το έχει στείλει. Επίσης ο χρήστης μπορεί να ανοίξει και να διαβάσει το μήνυμα το οποίο έχει στείλει.



Στιγμιότυπο Α.13

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν θα στείλει ένα νέο μήνυμα και έχει τα ακόλουθα χαρακτηριστικά:

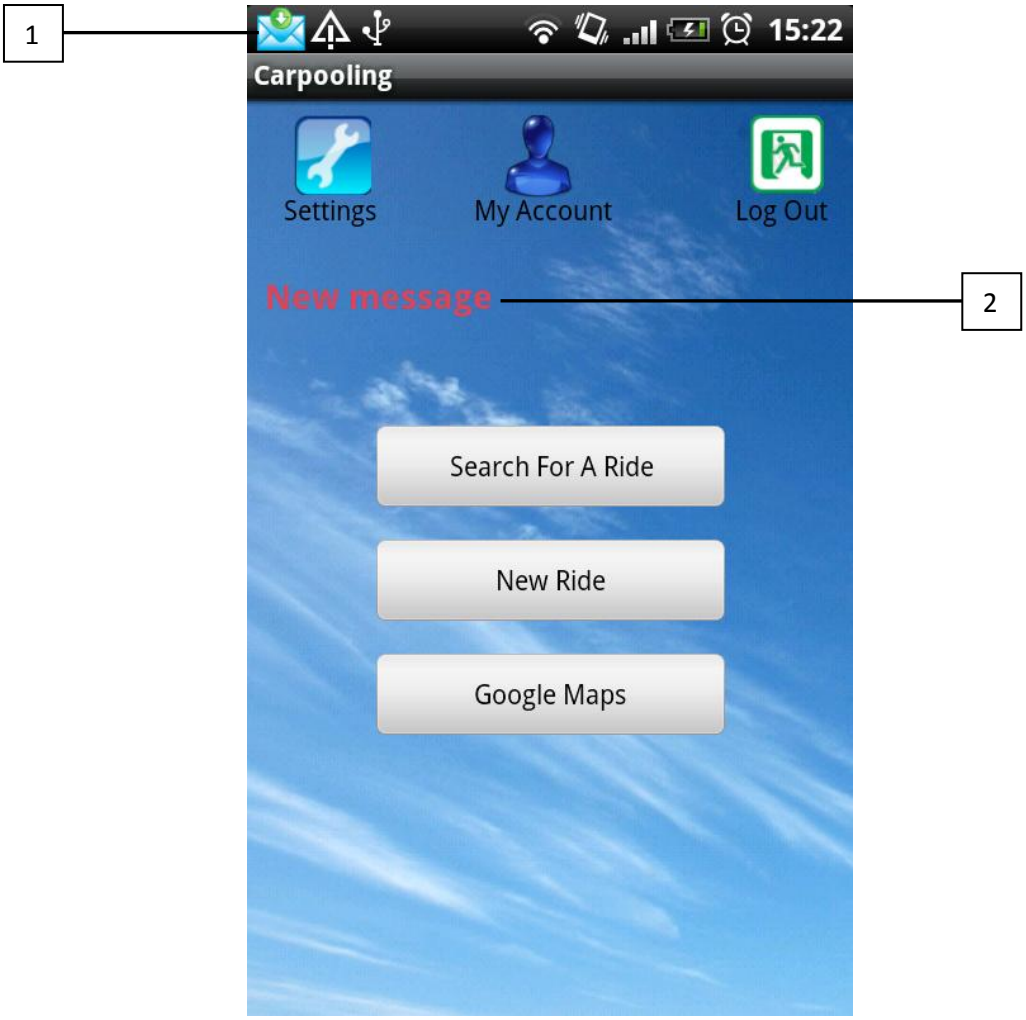
1. Ο χρήστης πατώντας αυτό το κουμπί θα οδηγηθεί σε μια άλλη οθόνη όπου μπορεί να δει και να επιλέξει σε πιο από τους φίλους του να στείλει το μήνυμα.
2. Ο χρήστης μπορεί να γράψει το User Name του χρήστη στον οποίο θα στείλει το μήνυμα. Το πεδίο αυτό είναι Auto complete και δεν χρειάζεται να γράψει ολόκληρο το User Name αλλά μόνο τα αρχικά.
3. Ο χρήστης γράφει το θέμα του μηνύματος. Επίσης μπορεί να το αφήσει κενό.
4. Ο χρήστης γράφει το μήνυμα του.
5. Με το πάτημα του κουμπιού Send ο χρήστης θα στείλει το μήνυμα στον χρήστη που επέλεξε.



Στιγμιότυπο Α.14

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τους διαθέσιμους φίλους του όπου μπορεί να στείλει το μήνυμα και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης βλέπει τους φίλους του ταξινομημένους σε αλφαβητική σειρά, όπου βλέπει το User Name, το όνομα και το επίθετο του κάθε χρήστη. Επιλέγοντας έναν από τους φίλους του θα οδηγηθεί πίσω στην οθόνη για αποστολή του μηνύματος στον επιλεγόμενο χρήστη.

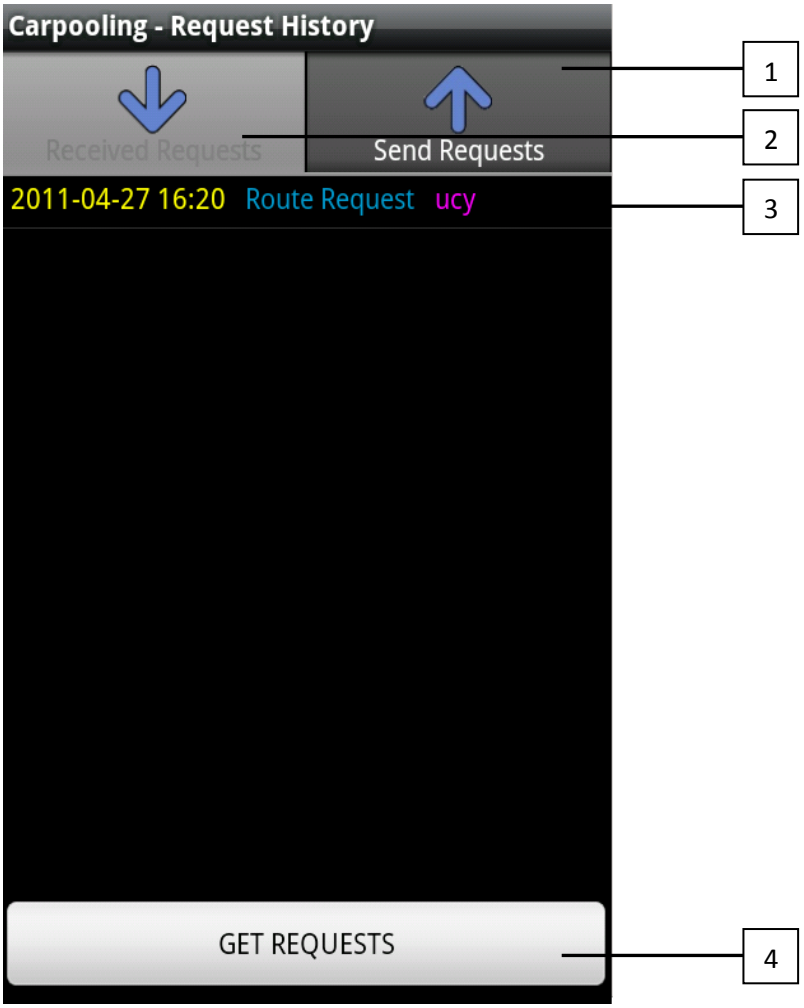


Στιγμιότυπο Α.15

Το πιο πάνω στιγμιότυπο είναι η κύρια οθόνη της εφαρμογής του χρήστη όπου στείλαμε το μήνυμα και έχει τα ακόλουθα χαρακτηριστικά:

1. Έρχεται ειδοποίηση στον χρήστη στην περιοχή των ειδοποιήσεων που χρησιμοποιούν τα Android λειτουργικά και ακλουθείτε με δόνηση τις κινητής συσκευής για ένα δευτερόλεπτο περίπου.
2. Έρχεται ειδοποίηση στην κύρια οθόνη της εφαρμογής του χρήστη πως έχει νέο μήνυμα στον λογαριασμό του.

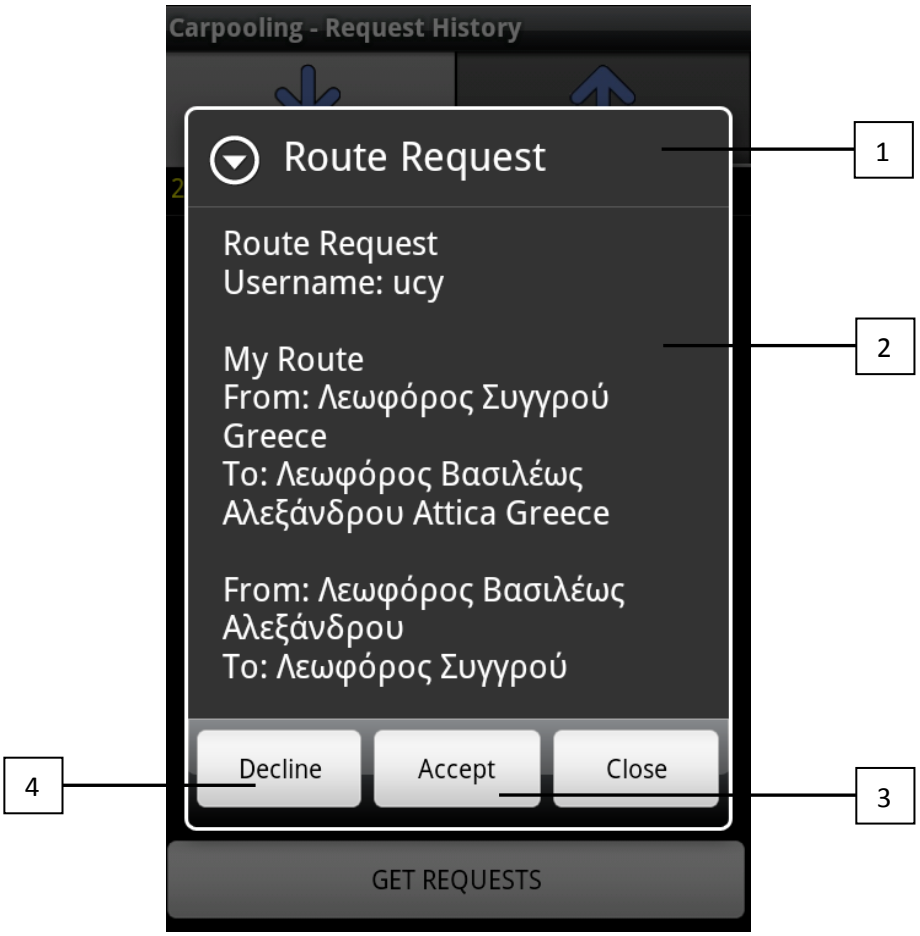
Να αναφέρουμε πως άλλες ειδοποιήσεις έρχονται παρόμοια με την πιο πάνω περίπτωση. Οι ειδοποιήσεις που χρησιμοποιούνται είναι όταν έχει νέο αίτημα από άλλο χρήστη, όταν πρέπει να ξεκινήσει την εκτέλεση ενός δρομολογίου και όταν πρέπει να αφήσει Feedback σε ένα χρήστη μετά τον τερματισμό ενός δρομολογίου.



Στιγμιότυπο Α.16

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα εισερχόμενα και εξερχόμενα αιτήματα του και έχει τα ακόλουθα χαρακτηριστικά:

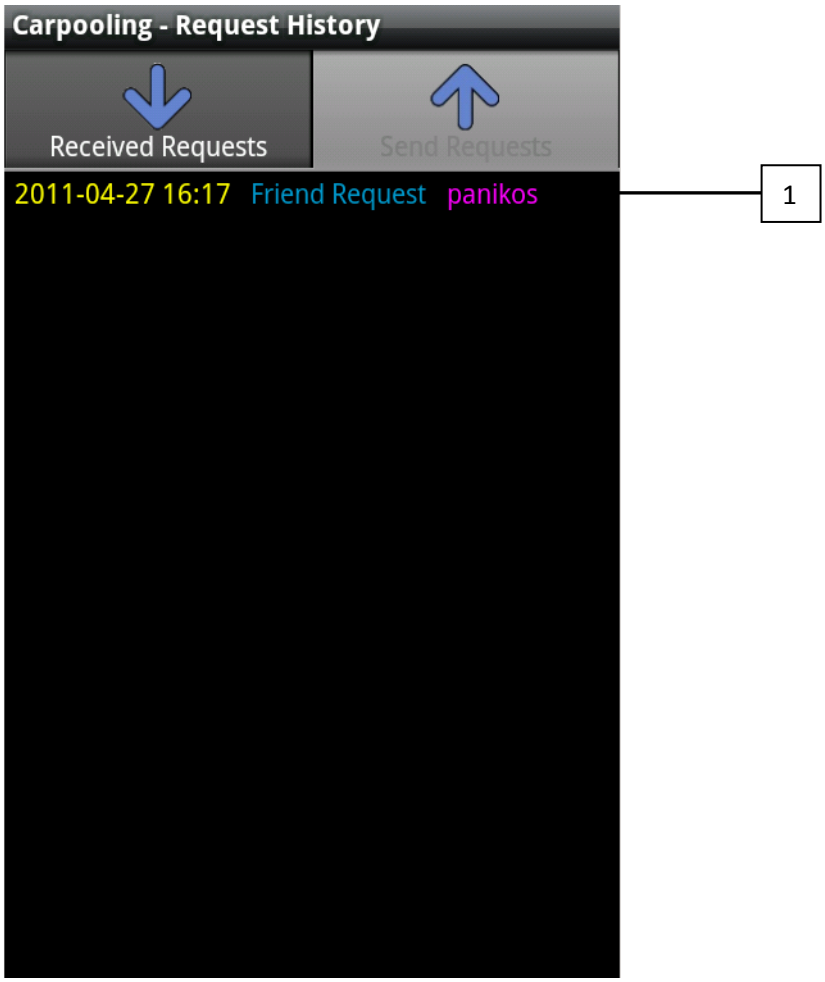
1. Πατώντας το tab Send Requests ο χρήστης μπορεί να δει τα εξερχόμενα αναπάντητα αιτήματα του.
2. Πατώντας το tab Received Requests ο χρήστης μπορεί να δει τα εισερχόμενα αναπάντητα αιτήματα του όπου είναι και η οθόνη που βλέπουμε πιο πάνω. Να πούμε πως τα αιτήματα είναι ταξινομημένα με την ημερομηνία που έχουν σταλθεί από τα νεότερα στα παλαιότερα.
3. Ο χρήστης μπορεί να δει τα εισερχόμενα αιτήματα του όπου βλέπει την ημερομηνία που έχουν σταλεί το είδος του αιτήματος, δηλαδή αν είναι Route Request ή Friend Request και τέλος βλέπει τον χρήστη που του έχει στείλει το συγκεκριμένο αίτημα. Τέλος ο χρήστης μπορεί να πατήσει σε ένα από τα εισερχόμενα αιτήματα του για να δει περεταίρω λεπτομέρειες και να απαντήσει στο αίτημα.
4. Πατώντας το κουμπί Get Requests ο χρήστης θα λάβει τα νέα αιτήματα στο λογαριασμό του αν υπάρχου.



Στιγμιότυπο Α.17

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα εισερχόμενα αιτήματα του και έχει τα ακόλουθα χαρακτηριστικά:

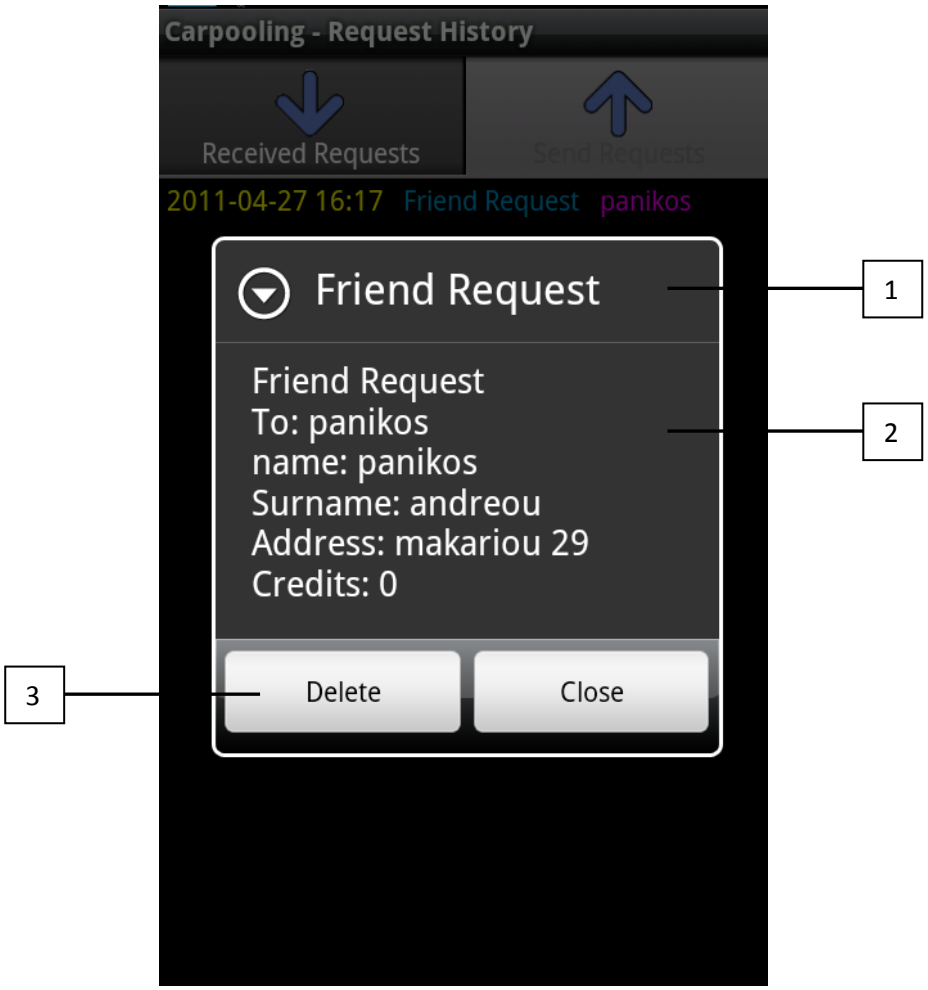
1. Ο χρήστης βλέπει αν είναι Route ή Friend Request.
2. Ο χρήστης βλέπει διάφορα στοιχεία για το συγκεκριμένο αίτημα, όπως για παράδειγμα ποιος χρήστης το έχει στείλει, για πιο δρομολόγιο αναφέρεται κτλ.
3. Πατώντας το κουμπί Accept ο χρήστης αποδέχεται το συγκεκριμένο αίτημα του χρήστη.
4. Πατώντας το κουμπί Decline ο χρήστης απορρίπτει το συγκεκριμένο αίτημα.



Στιγμιότυπο Α.18

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα εξερχόμενα αιτήματα του και έχει τα ακόλουθα χαρακτηριστικά:

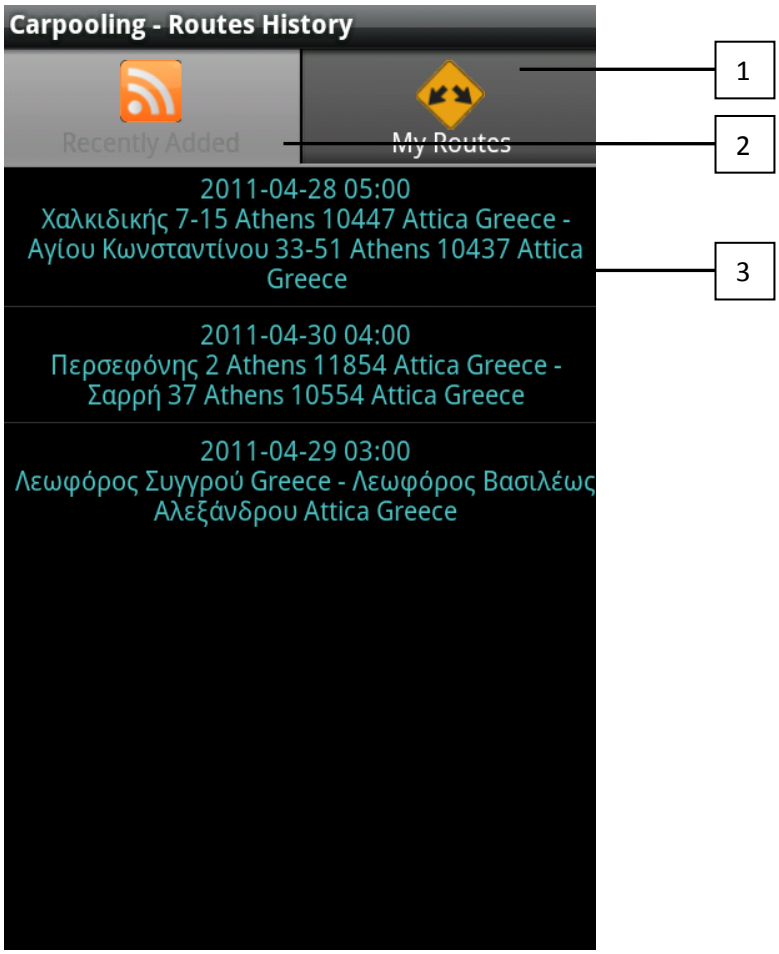
1. Ο χρήστης βλέπει τα εξερχόμενα αιτήματα του, βλέπει την ημερομηνία που τα έχει σταλεί, αν είναι Friend ή Route Request και τον χρήστη στον οποίο στάλθηκε το αίτημα. Ο χρήστης μπορεί να πατήσει πάνω στο αίτημα που έχει κάνει και να δει περαιτέρω λεπτομέρειες.



Στιγμιότυπο Α.19

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει ένα από τα εξερχόμενα αιτήματα του και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης βλέπει αν είναι Friend ή Route Request.
2. Ο χρήστης βλέπει σχετικές λεπτομέρειες όσο αφορά το αίτημα που έχει στείλει, όπως το User Name του χρήστη που το έστειλε, το όνομα του, το επίθετο του, την διεύθυνση του και τον αριθμό των Credits του.
3. Πατώντας το κουμπί Delete ο χρήστης μπορεί να διαγράψει το συγκεκριμένο αίτημα που έκανε.



Στιγμιότυπο Α.20

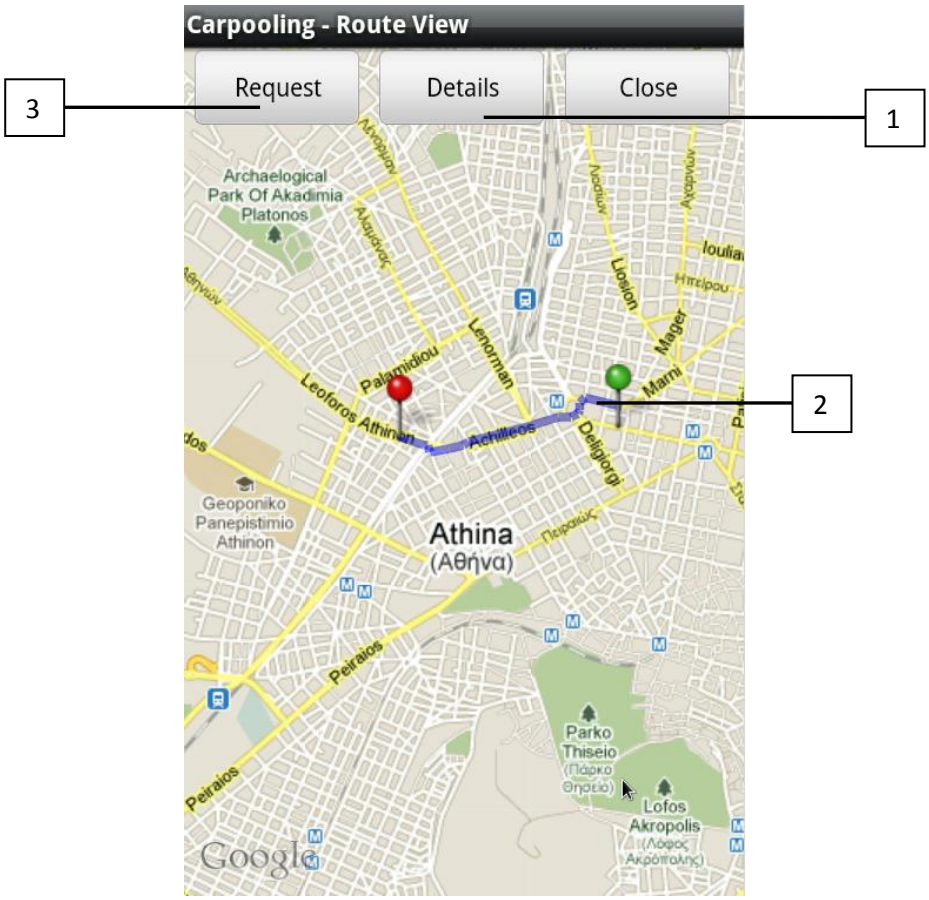
Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα πιο πρόσφατα δρομολόγια που προστέθηκαν από τους φίλους του ή όταν θέλει να δει τα δικά του δρομολόγια και έχει τα ακόλουθα χαρακτηριστικά:

1. Πατώντας το tab My Routes ο χρήστης θα δει τα δικά του δρομολόγια ή τα δρομολόγια στα οποία συμμετέχει σαν επιβάτης.
2. Πατώντας το tab Recently Added ο χρήστης θα δει τα πιο πρόσφατα δρομολόγια που προστέθηκαν από τους φίλους του ταξινομημένα βάσει της ημερομηνίας που προστέθηκαν όπως φαίνεται στο πιο πάνω στιγμιότυπο.
3. Ο χρήστης βλέπει τα διαθέσιμα δρομολόγια. Βλέπει την ημερομηνία που θα διεξαχθούν την αφετηρία και τον προορισμό τους. Επίσης επιλέγοντας ένα από αυτά μπορεί να δει περαιτέρω σχετικές πληροφορίες για το συγκεκριμένο δρομολόγιο.

Στιγμιότυπο Α.21

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει ένα από τα πιο πρόσφατα δρομολόγια που προστέθηκαν από τους φίλους του και έχει τα ακόλουθα χαρακτηριστικά:

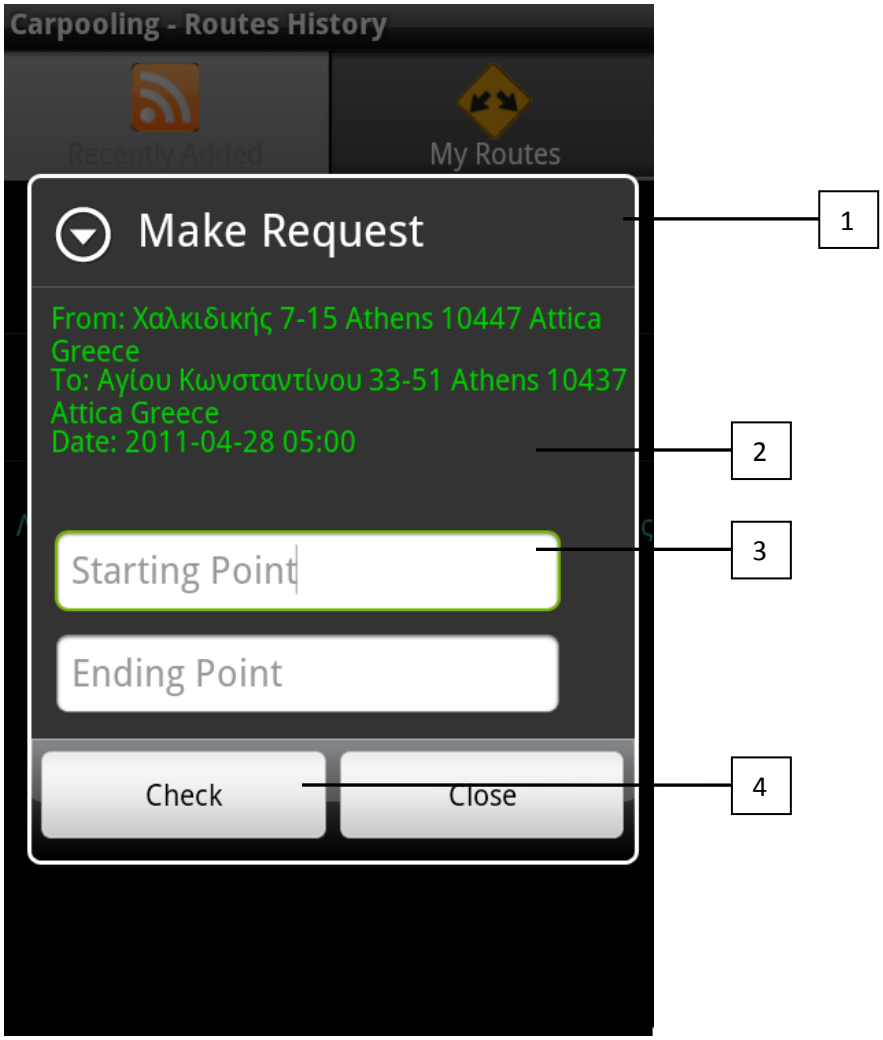
1. Ο χρήστης μπορεί να δει σχετικές πληροφορίες για το δρομολόγιο, όπως την ημερομηνία και ώρα που θα διεξαχθεί, την αφετηρία και τον προορισμό τους, αλλά επίσης και των οδηγό, την αξιοπιστία του, τις διαθέσιμες θέσεις που υπάρχουν και τους επιβάτες αν υπάρχουν.
2. Πατώντας το κουμπί Map View ο χρήστης μπορεί να δει το δρομολόγιο σε χάρτη με την βοήθεια των Google Maps.
3. Πατώντας το κουμπί Request θα εμφανιστεί μια νέα οθόνη, όπου ο χρήστης μπορεί να κάνει αίτημα για το συγκεκριμένο δρομολόγιο.



Στιγμιότυπο Α.22

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει σε χάρτη το δρομολόγιο που επίλεξε στο πιο πάνω βήμα και έχει τα ακόλουθα χαρακτηριστικά:

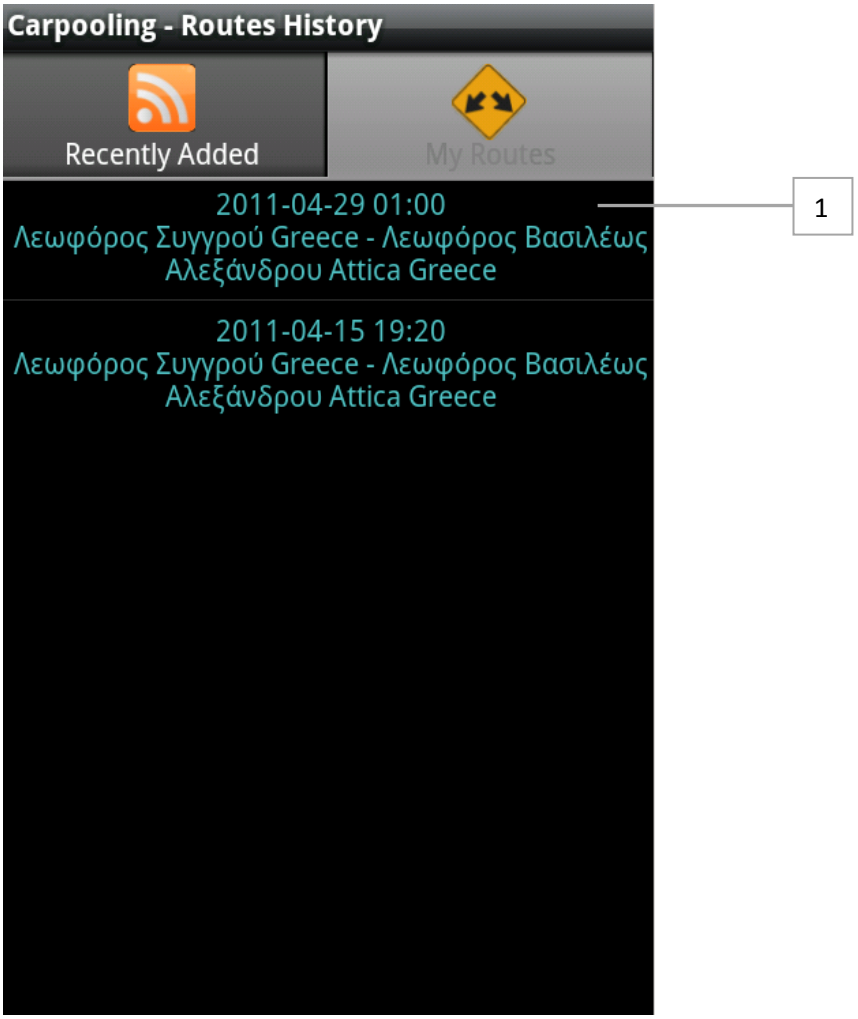
1. Πατώντας το κουμπί Details ο χρήστης μπορεί να δει σχετικές πληροφορίες για το δρομολόγιο.
2. Ο χρήστης βλέπει το ακριβές δρομολόγιο από την αφετηρία η οποία συμβολίζεται με κόκκινη πινέζα μέχρι τον τερματισμό (πράσινη πινέζα).
3. Πατώντας το κουμπί Request ο χρήστης μπορεί μέσω μια άλλης οθόνης να κάνει αίτηση για το συγκεκριμένο δρομολόγιο.



Στιγμιότυπο Α.23

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να κάνει αίτηση στο συγκεκριμένο δρομολόγιο που επέλεξε στο προηγούμενο βήμα και έχει τα ακόλουθα χαρακτηριστικά:

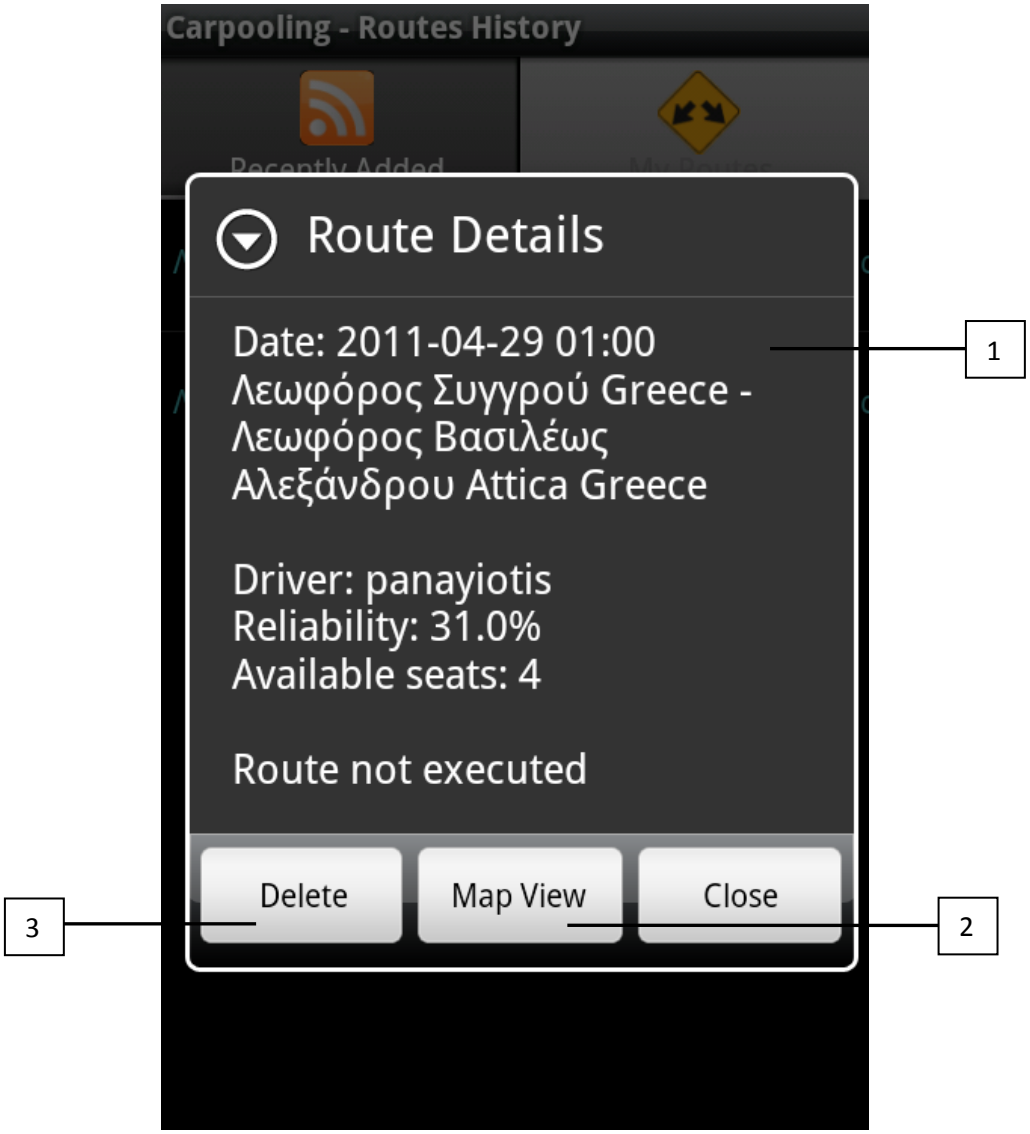
1. Ο χρήστης βλέπει σχετικές πληροφορίες όσον αφορά το συγκεκριμένο δρομολόγιο.
2. Ο χρήστης γράφει το αρχικό σημείο που θέλει να περάσει ο οδηγός.
3. Ο χρήστης γράφει το τελικό σημείο που θέλει να περάσει ο οδηγός.
4. Πατώντας το κουμπί Check ελέγχεται βάσει του αλγορίθμου που υλοποιήσαμε για την ταύτιση δρομολόγιων αν τα δρομολόγια των δυο χρηστών ταυτίζονται ούτως ώστε να γίνει το αίτημα του χρήστη.



Στιγμιότυπο Α.24

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τα δικά του δρομολόγια ή τα δρομολόγια τα οποία συμμετέχει και έχει τα ακόλουθα χαρακτηριστικά:

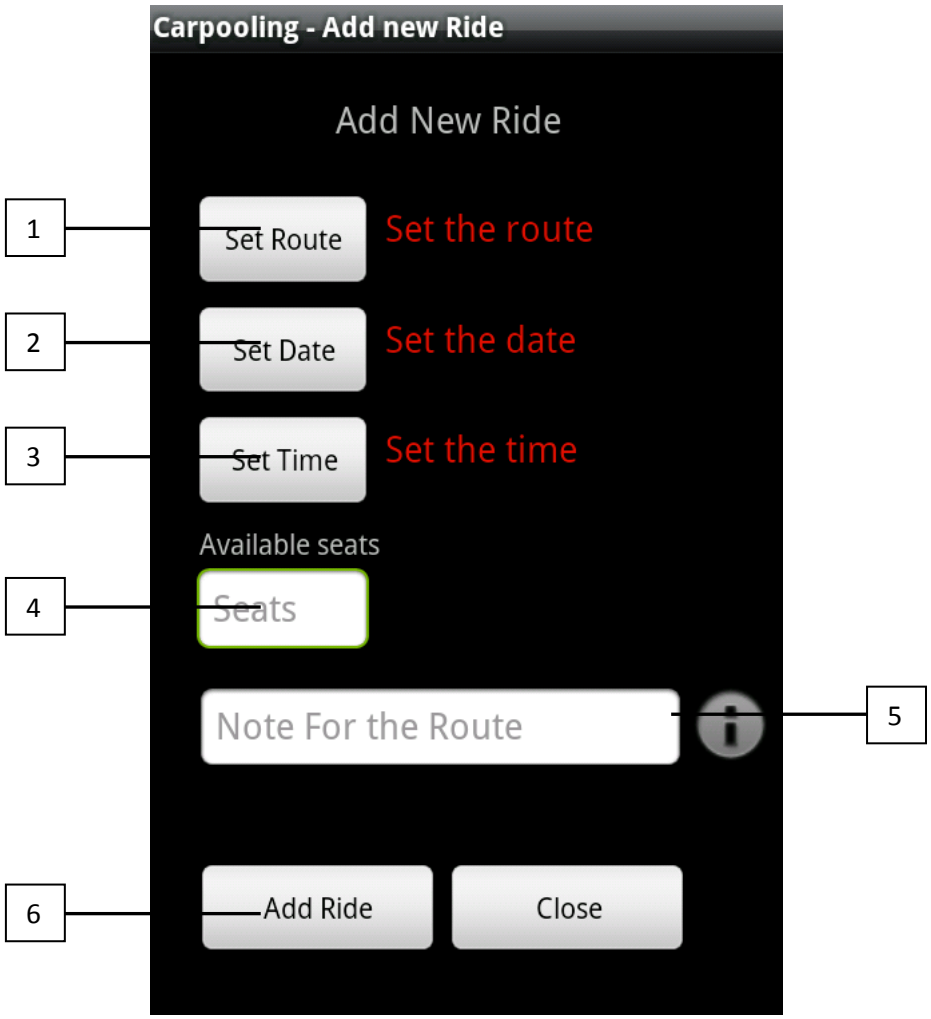
1. Ο χρήστης βλέπει τα δικά του δρομολόγια τα οποία είναι ταξινομημένα βάσει της ημερομηνίας, όπου παρουσιάζονται πρώτα τα πιο πρόσφατα. Επίσης ο χρήστης βλέπει την ημερομηνία και ώρα του δρομολογίου, την αφετηρία και τον προορισμό και μπορεί επιλέγοντας ένα από αυτά να δει περαιτέρω πληροφορίες για αυτό.



Στιγμιότυπο Α.25

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τις λεπτομέρειες ενός δρομολογίου από την προηγούμενη οθόνη και έχει τα ακόλουθα χαρακτηριστικά:

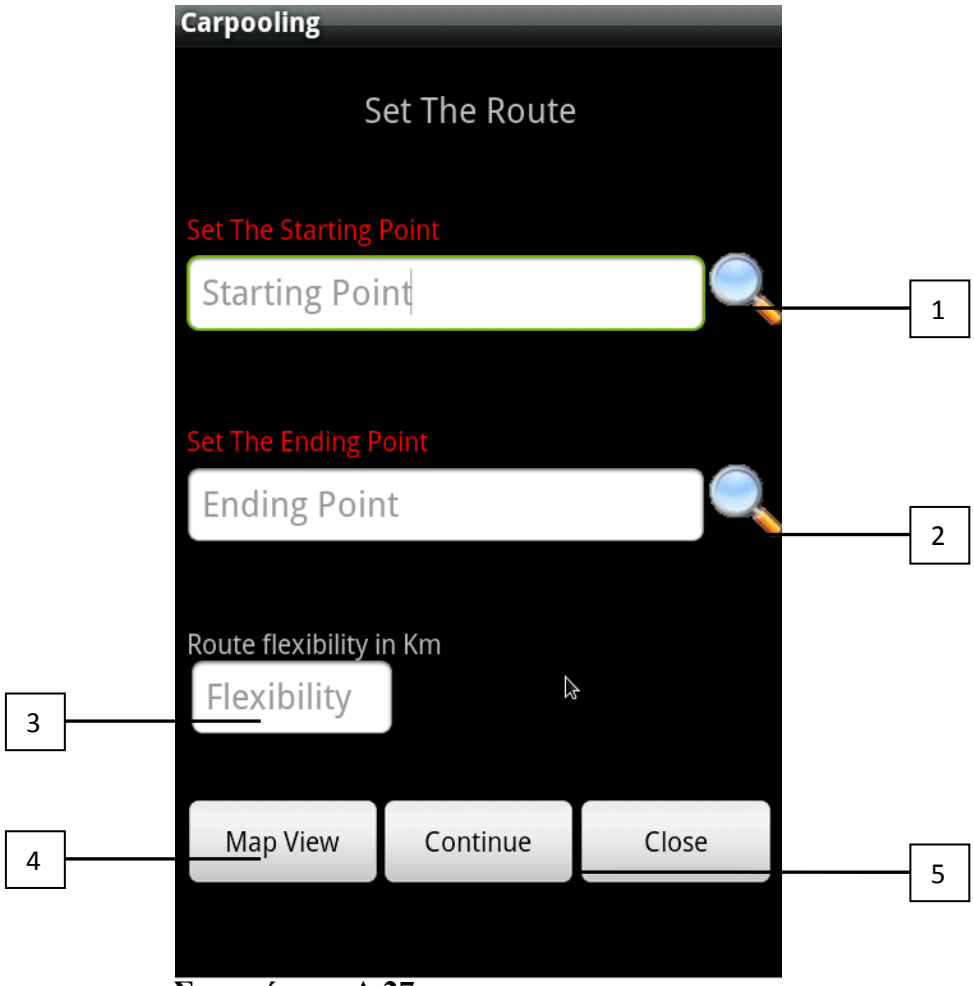
1. Ο χρήστης βλέπει σχετικές πληροφορίες για το δρομολόγιο, όπως ημερομηνία, ώρα, αφετηρία, τερματισμό κτλ. Επίσης ο χρήστης μπορεί να δει τον οδηγό και τους επιβάτες του συγκεκριμένου δρομολογίου.
2. Πατώντας το κουμπί Map View ο χρήστης μπορεί να δει το δρομολόγιο με την χρήση χαρτών που διατίθενται από το Google Maps.
3. Πατώντας το κουμπί Delete ο χρήστης μπορεί να διαγράψει το συγκεκριμένο δρομολόγιο.



Στιγμιότυπο Α.26

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να προσθέσει ένα νέο δρομολόγιο από την κύρια οθόνη της εφαρμογής και έχει τα ακόλουθα χαρακτηριστικά:

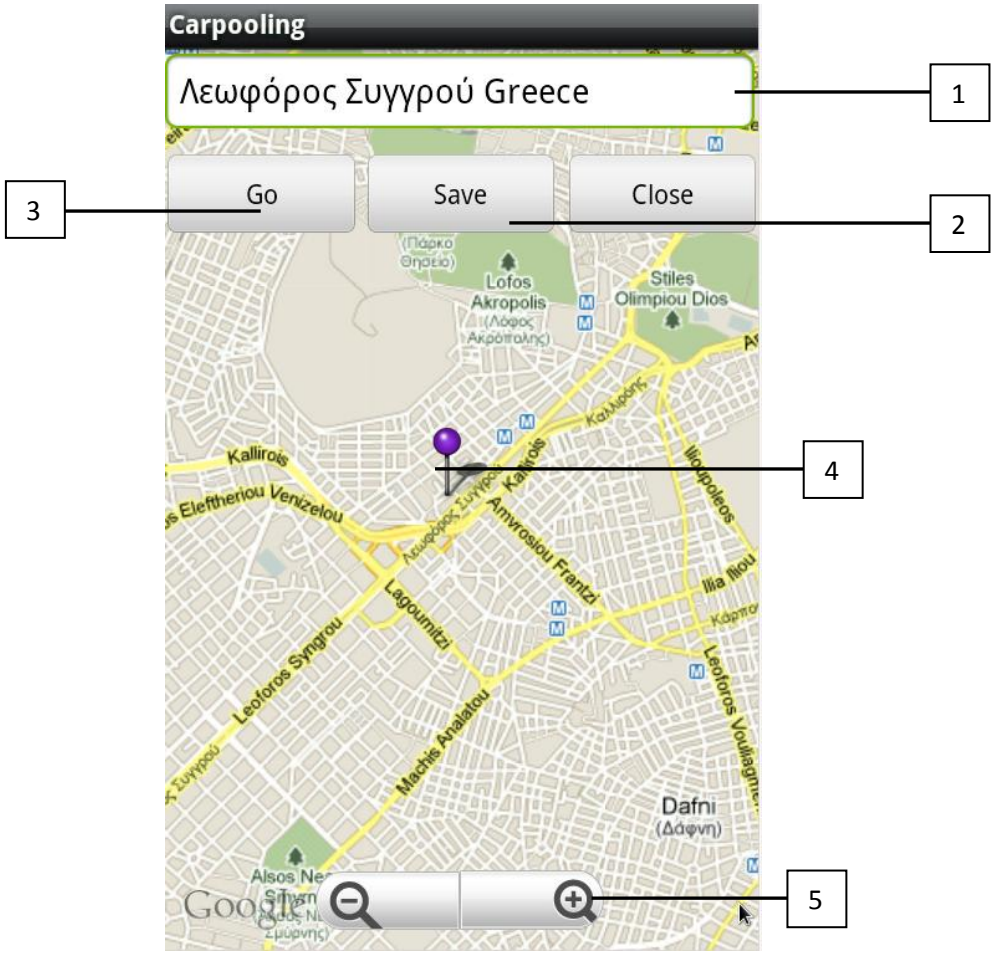
1. Πατώντας το κουμπί Set Route ο χρήστης οδηγείται σε μια άλλη οθόνη όπου ορίζει το σημείο έναρξης και τερματισμού του δρομολογίου.
2. Πατώντας το κουμπί Set Date ο χρήστης ορίζει την ημερομηνία που θέλει να πραγματοποιηθεί το δρομολόγιο.
3. Πατώντας το κουμπί Set Time ο χρήστης ορίζει την ώρα του δρομολογίου.
4. Στο πεδίο με αριθμό 4 ο χρήστης ορίζει τις διαθέσιμες θέσεις που διαθέτει στο αυτοκίνητο του.
5. Στο πεδίο με αριθμό 5 ο χρήστης ορίζει οποιαδήποτε σημείωση θέλει να δουν οι άλλοι χρήστες όπως για παράδειγμα να μοιραστούν τα έξοδα του ταξιδιού.
6. Πατώντας το κουμπί Add Ride ελέγχεται η εγκυρότητα των πεδίων και προστίθεται το δρομολόγιο ούτως ώστε να είναι διαθέσιμο στους φίλους του χρήστη.



Στιγμιότυπο Α.27

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει το κουμπί Set Route από την προηγούμενη οθόνη και έχει τα ακόλουθα χαρακτηριστικά:

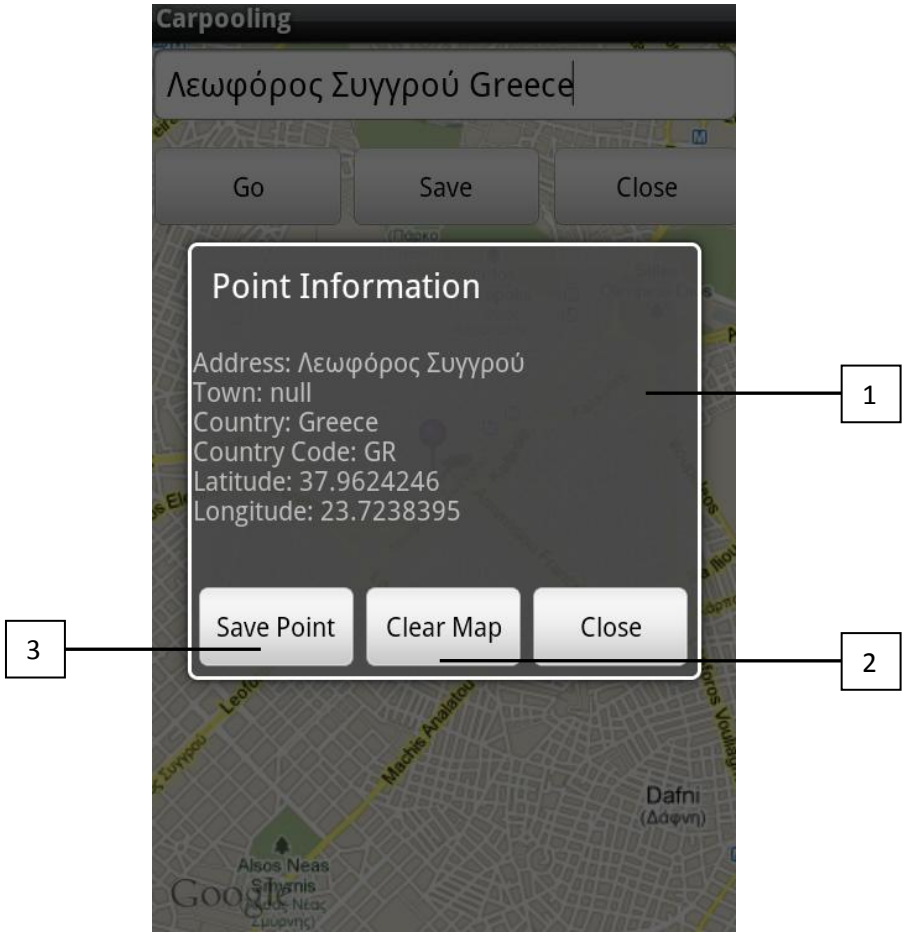
1. Ο χρήστης ορίζει το σημείο έναρξης του δρομολογίου.
2. Ο χρήστης ορίζει το σημείο τερματισμού του δρομολογίου.
3. Ο χρήστης ορίζει την ευελιξία του δρομολογίου, δηλαδή την μέγιστη απόσταση που αποδέχεται να παρακάμψει το δρομολόγιο του για να εξυπηρετήσει έναν άλλο χρήστη.
4. Πατώντας το κουμπί Map View ο χρήστης μπορεί να δει το δρομολόγιο σε χάρτη.
5. Πατώντας το κουμπί Continue ο χρήστης μπορεί να συνεχίσει στο επόμενο βήμα της διαδικασία προσθήκης ενός δρομολογίου.



Στιγμιότυπο Α.28

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να ορίσει το αρχικό ή το τελικό σημείο του δρομολογίου από την προηγούμενη οθόνη που παρουσιάσαμε και έχει τα ακόλουθα χαρακτηριστικά:

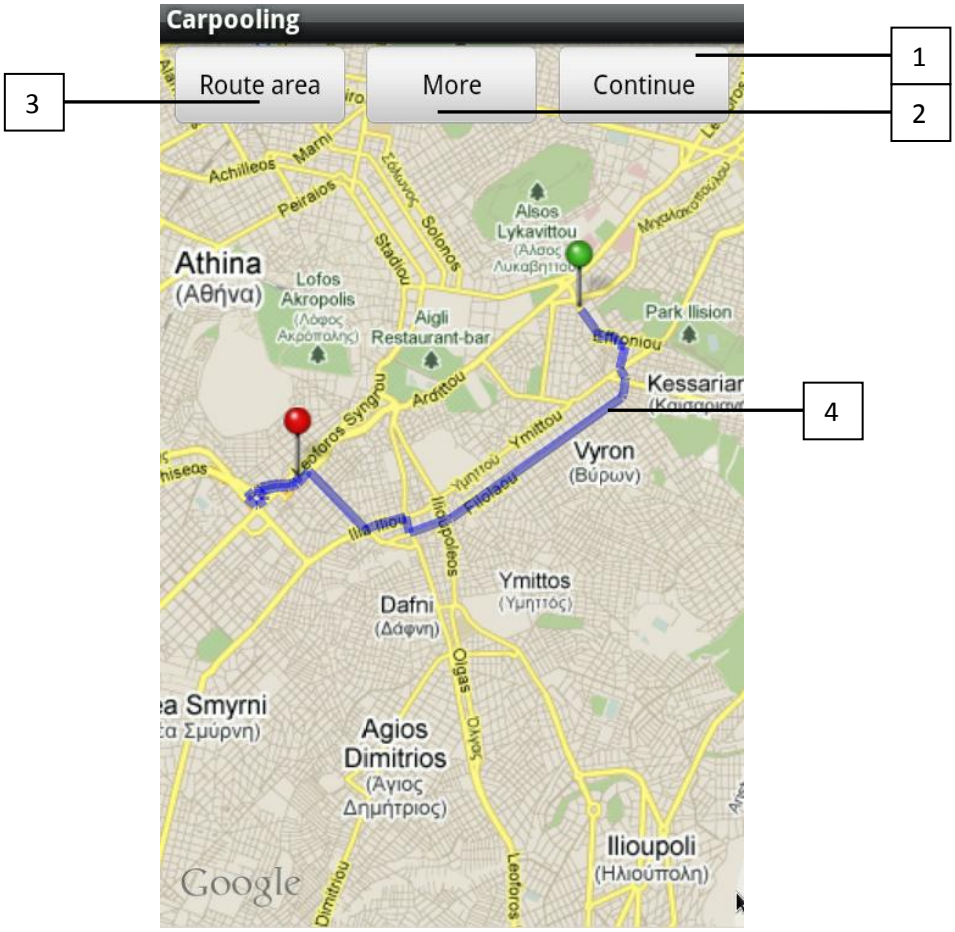
1. Ο χρήστης γράφει την διεύθυνση που θέλει να ορίσει το αρχικό ή το τελικό σημείο του δρομολόγιου του.
2. Πατώντας το κουμπί Save ο χρήστης μπορεί να δει περισσότερες πληροφορίες σχετικά με την τοποθεσία που έθεσε και να την αποθηκεύσει.
3. Πατώντας το κουμπί Go ο χρήστης οδηγείται στην διεύθυνση που όρισε πιο πάνω αν είναι έγκυρη.
4. Ο χρήστης βλέπει την ακριβή τοποθεσία της διεύθυνσης που όρισε και μπορεί να μετακινήσει την πινέζα με το δάκτυλο του.
5. Ο χρήστης μπορεί να μεγεθύνει ή να σμικρύνει τον χάρτη.



Στιγμιότυπο Α.29

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να αποθηκεύσει το αρχικό ή το τελικό σημείο του δρομολογίου από την προηγούμενη οθόνη που παρουσιάσαμε και έχει τα ακόλουθα χαρακτηριστικά:

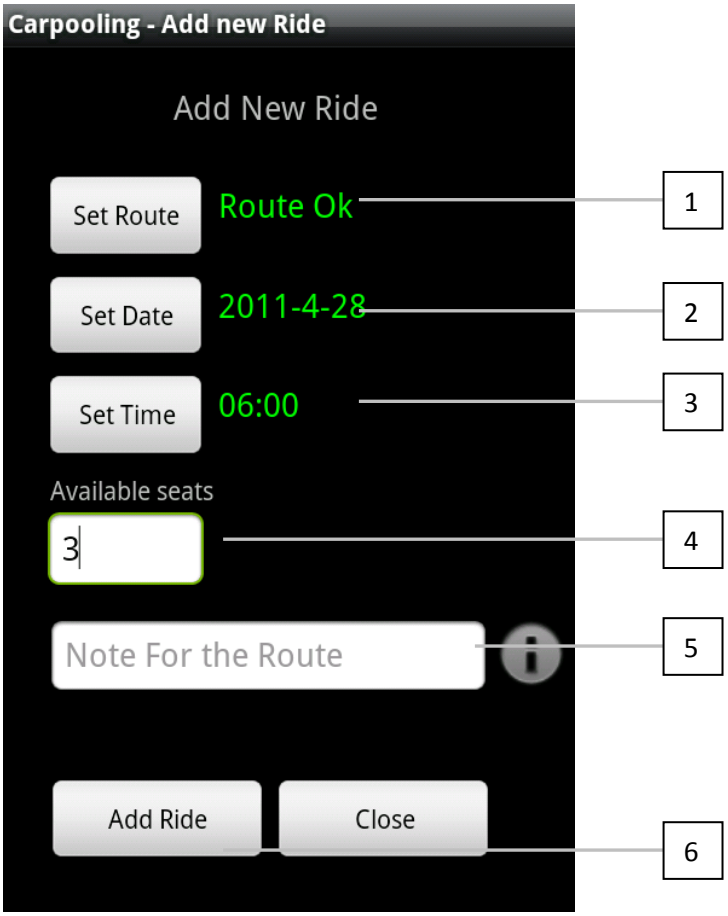
1. Ο χρήστης βλέπει κάποιες πληροφορίες σχετικά με την διεύθυνση που όρισε.
2. Πατώντας το κουμπί Clear Map ο χρήστης διαγράφει όλα τα σημεία από τον χάρτη.
3. Πατώντας το κουμπί Save Point ο χρήστης αποθηκεύει το αρχικό ή το τελικό σημείο του δρομολογίου αναλόγως.



Στιγμιότυπο Α.30

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν ορίσει το αρχικό και τελικό σημείο του δρομολογίου του και επιλέξει να δει την διαδρομή με την χρήση χαρτών και έχει τα ακόλουθα χαρακτηριστικά:

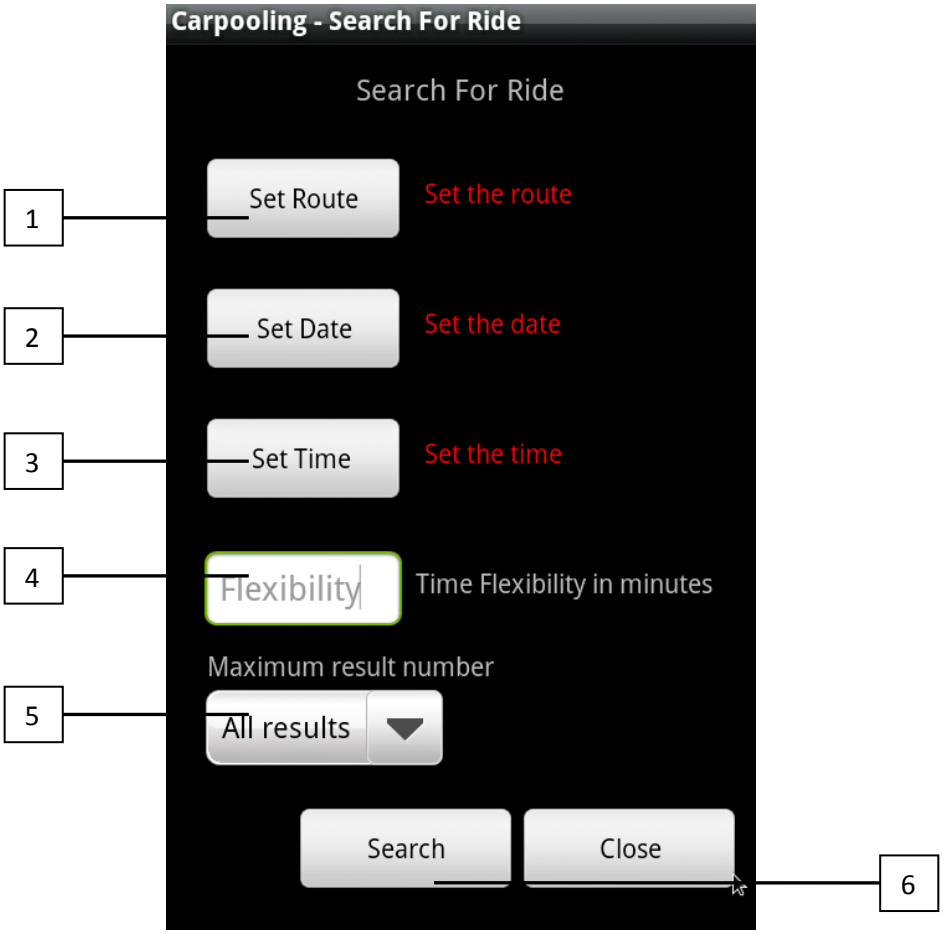
1. Ο χρήστης μπορεί να συνεχίσει στο επόμενο βήμα για να προσθέσει το δρομολόγιο πατώντας το κουμπί Continue.
2. Πατώντας το κουμπί More ο χρήστης μπορεί να δει περισσότερες πληροφορίες για το δρομολόγιο όπως την διεύθυνση του αρχικού και τελικού σημείο, την απόσταση τους κτλ.
3. Πατώντας το κουμπί Route Area ο χρήστης μπορεί να δει την χαρακτηριστική περιοχή που ορίζει το δρομολόγιο του.
4. Ο χρήστης βλέπει το δρομολόγιο του με την χρήστη χαρτών Google Maps.



Στιγμιότυπο Α.31

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν ορίσει όλα τα χαρακτηριστικά ενός δρομολόγιου και έχει τα ακόλουθα χαρακτηριστικά:

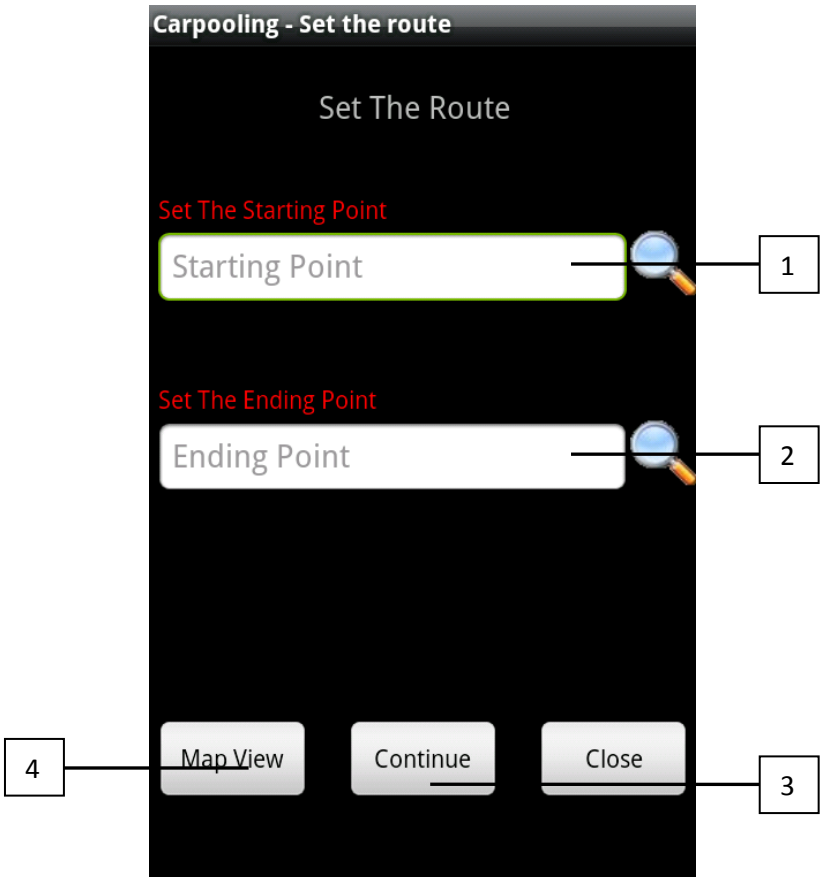
1. Ο χρήστης όρισε σωστά το αρχικό και το τελικό σημείο του δρομολογίου
2. Ο χρήστης όρισε την ημερομηνία του δρομολογίου.
3. Ο χρήστης όρισε την ώρα του δρομολογίου.
4. Ο χρήστης δίνει τις διαθέσιμες θέσεις που έχει στο αυτοκίνητο του.
5. Ο χρήστης μπορεί να γράψει οποιαδήποτε σημείωση θέλει να δουν οι άλλοι χρήστες όπως για παράδειγμα να μοιραστούν τα έξοδα του ταξιδίου.
6. Πατώντας το κουμπί Add Ride ο χρήστης μπορεί να προσθέσει το δρομολόγιο στην βάση δεδομένων και να είναι διαθέσιμο στους φίλους του, εφόσον όλα τα χαρακτηριστικά γνωρίσματα του είναι έγκυρα.



Στιγμιότυπο Α.32

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη επιλέξει να εκτελέσει μια αναζήτηση στα διαθέσιμα δρομολόγια που υπάρχουν στους φίλους του και έχει τα ακόλουθα χαρακτηριστικά:

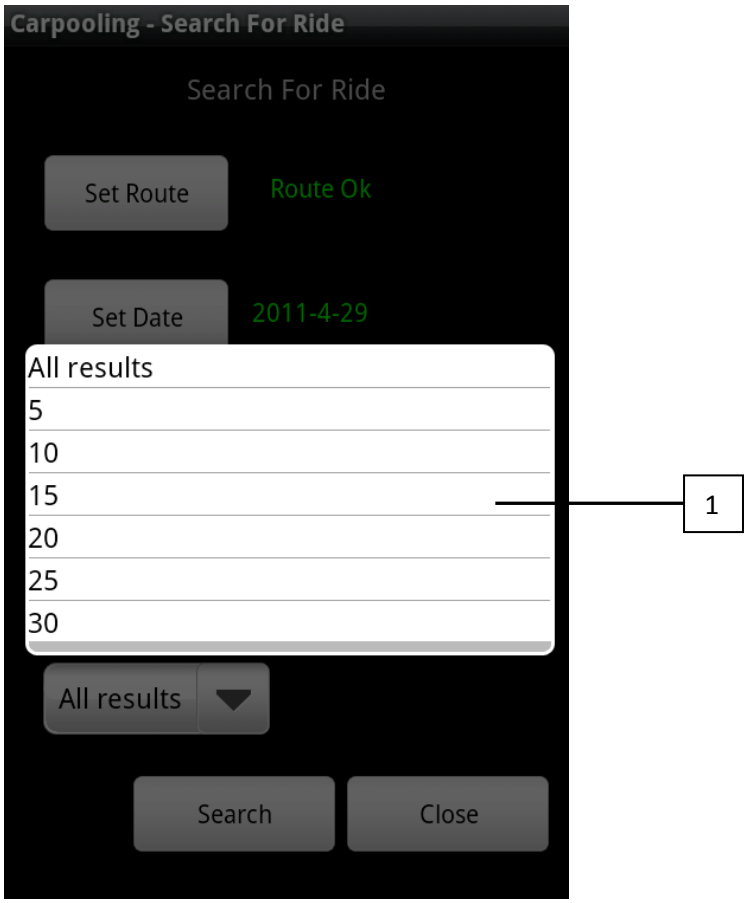
1. Πατώντας το κουμπί Set Route ο χρήστης μπορεί να ορίσει το σημείο έναρξης και τερματισμού του δρομολογίου.
2. Πατώντας το κουμπί Set Date ο χρήστης μπορεί να θέσει την ημερομηνία που θέλει να εκτελέσει το δρομολόγιο.
3. Πατώντας το κουμπί Set Time ο χρήστης μπορεί να θέσει την ώρα που θέλει να εκτελέσει το δρομολόγιο.
4. Ο χρήστης μπορεί να θέσει την ευελιξία στην ώρα που έθεσε, όπου μπορεί να χρησιμοποιηθεί είτε θετικά είτε αρνητικά.
5. Ο χρήστης μπορεί να επιλέξει τον μέγιστο αριθμό αποτελεσμάτων που αναμένει από την αναζήτηση.
6. Πατώντας το κουμπί Search ελέγχονται τα χαρακτηριστικά των πεδίων αν είναι έγκυρα και εκτελείται η αναζήτηση.



Στιγμιότυπο Α.33

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη για να ορίσει το σημείο έναρξης και τερματισμού του δρομολογίου και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης ορίζει το σημείο έναρξης του δρομολογίου.
2. Ο χρήστης ορίζει το σημείο τερματισμού του δρομολογίου.
3. Πατώντας το κουμπί Continue ο χρήστης προχωρεί στο επόμενο βήμα της αναζήτησης.
4. Πατώντας το κουμπί Map View ο χρήστης μπορεί να δει το δρομολόγιο του με την χρήση χαρτών.

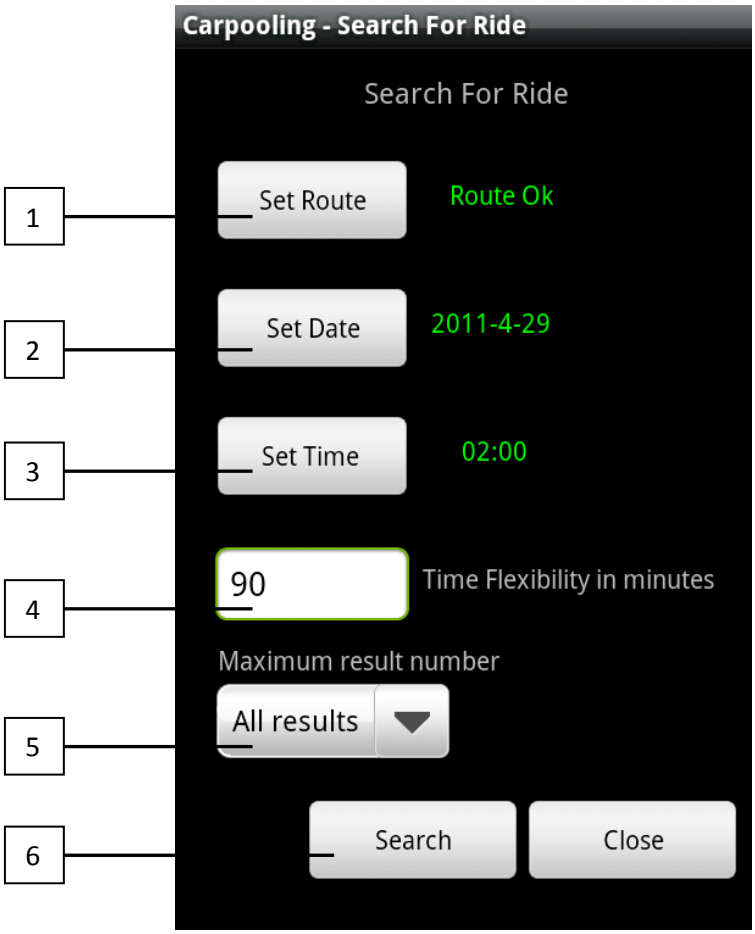


Στιγμιότυπο Α.34

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη για να ορίσει τον μέγιστο αριθμό των αποτελεσμάτων που αναμένει και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης επιλέγει των μέγιστο αριθμό αποτελεσμάτων που αναμένει από την αναζήτηση πιθανών δρομολογίων.

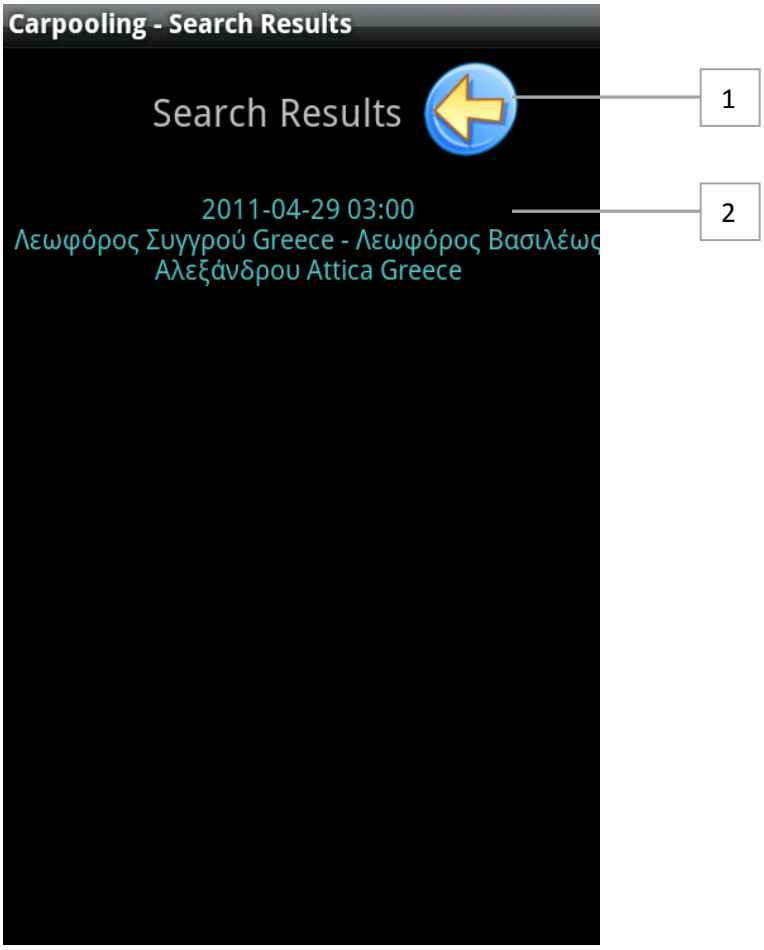
Ο λόγος για τον οποίο χρειάζεται αυτό το χαρακτηριστικό είναι για να παρουσιάζονται στον χρήστη όσο το δυνατό γρηγορότερα τα αποτελέσματα της αναζήτησης αναλόγως με το μέγεθος των αποτελεσμάτων που επιθυμεί να του παρουσιαστούν.



Στιγμιότυπο Α.35

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν ορίσει όλα τα χαρακτηριστικά γνωρίσματα για να εκτελέσει την αναζήτηση πιθανών δρομολογίων και έχει τα ακόλουθα χαρακτηριστικά:

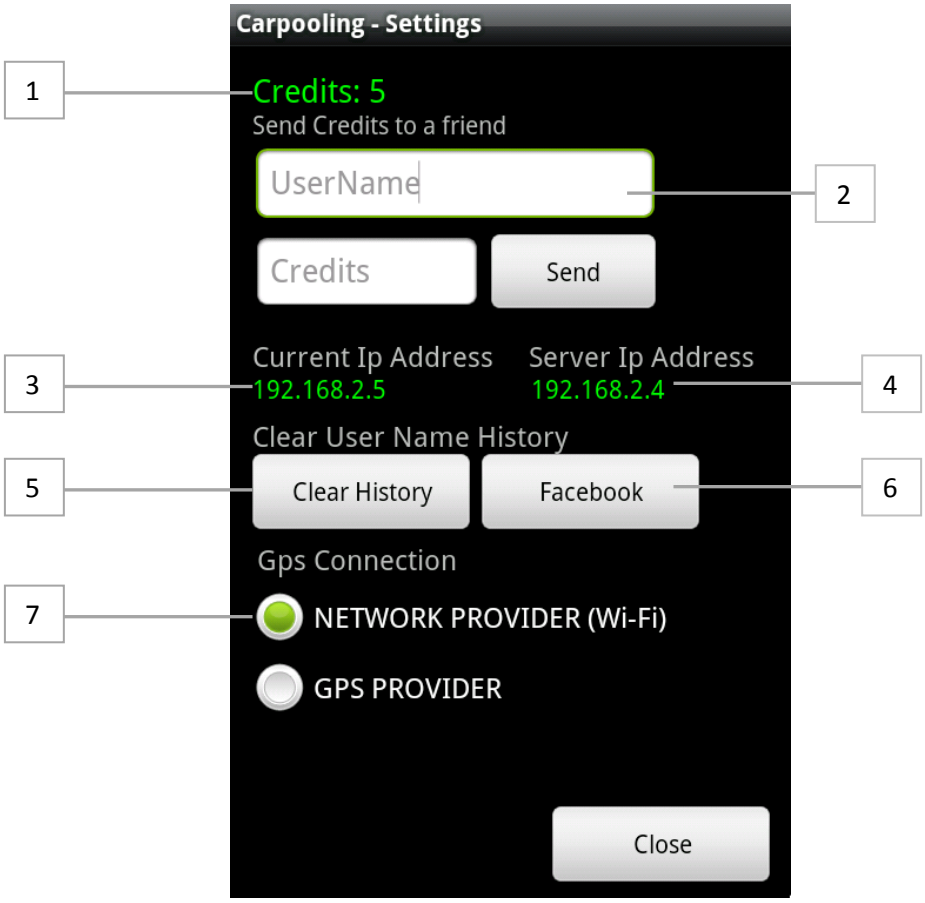
1. Ο χρήστης έθεσε ορθά το αρχικό και τελικό σημείο του δρομολογίου.
2. Ο χρήστης έθεσε ορθά την ημερομηνία για την οποία θέλει να αναζητήσει πιθανά δρομολόγια.
3. Ο χρήστης έθεσε ορθά την ώρα του δρομολογίου.
4. Ο χρήστης έθεσε την ευελιξία του χρόνου που θέλει για το δρομολόγιο του.
5. Ο χρήστης επέλεξε να δει όλα τα διαθέσιμα δρομολόγια που υπάρχουν από τους φίλους του.
6. Πατώντας το κουμπί Search θα εμφανιστούν σε μια άλλη οθόνη τα αποτελέσματα της αναζήτηση.



Στιγμιότυπο Α.36

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν εκτελέσει την αναζήτηση δρομολογίων και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης μπορεί να επιστρέψει στην προηγούμενη οθόνη και να αλλάξει κάποιες παραμέτρους και να ξαναεκτελέσει την αναζήτηση.
2. Ο χρήστης βλέπει τα αποτελέσματα της αναζήτηση όπου μπορεί να επιλέξει ένα από αυτά και να δει περαιτέρω λεπτομέρειες και να κάνει αίτηση για κράτηση θέσης σε ένα δρομολόγιο.

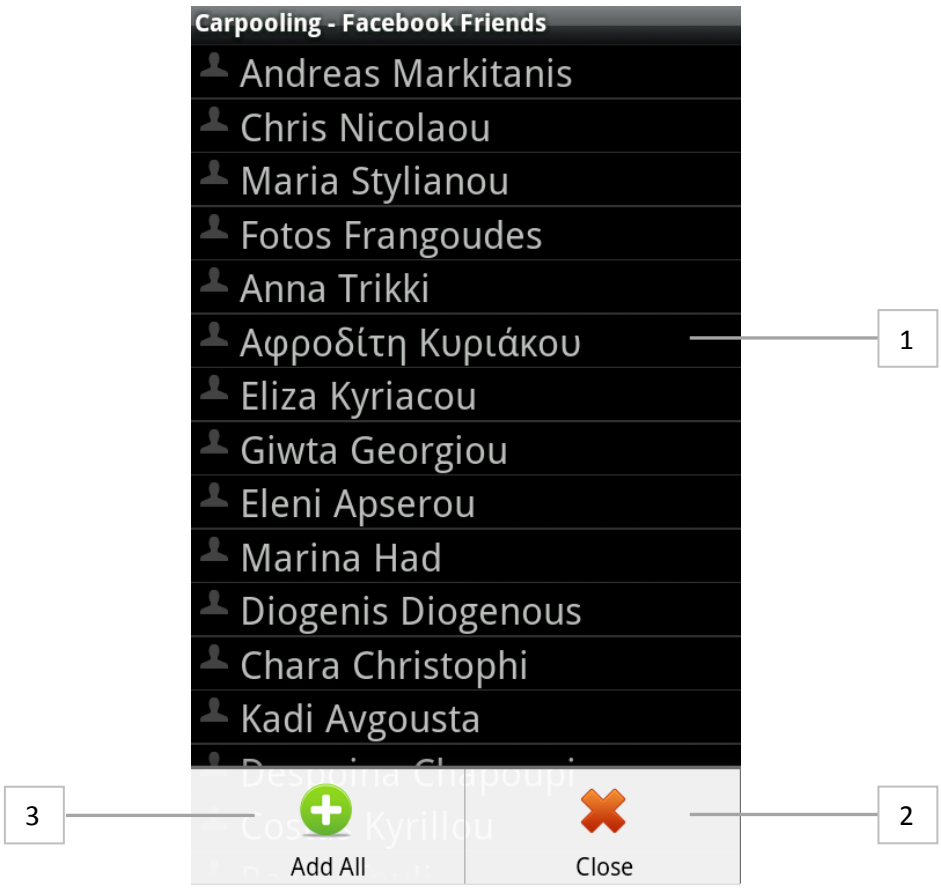


Στιγμιότυπο Α.37

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να δει τις ρυθμίσεις της εφαρμογής από την κύρια οθόνη της εφαρμογής και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης μπορεί να δει το ποσό των Credits που έχει στον λογαριασμό του.
2. Ο χρήστης μπορεί να δώσει το User Name ενός φίλου του, τον αριθμό των Credits που θέλει να του στείλει και ακολούθως πατώντας το κουμπί Send να του σταλούν τα Credits (αν έχει δώσει έγκυρο ποσό).
3. Ο χρήστης μπορεί να δει το IP Address της κινητής συσκευής του για την τρέχουσα στιγμή.
4. Ο χρήστης μπορεί να δει το IP Address του εξυπηρετητή.
5. Πατώντας το κουμπί Clear History ο χρήστης μπορεί να διαγράψει τα User Name που υπήρχαν αποθηκευμένα στην εφαρμογή για σκοπούς εξυπηρέτησης του χρήστη και ταχύτερης πρόσβασης του στην εφαρμογή.
6. Πατώντας το κουμπί Facebook ο χρήστης κάνει Log In στο κοινωνικό δίκτυο Facebook και στην συνέχεια μπορεί να προσθέσει τους φίλους που έχει στο κοινωνικό δίκτυο, στους φίλους που έχει στην εφαρμογή.
7. Ο χρήστης μπορεί να επιλέξει τον τρόπο λειτουργίας του GPS της κινητής συσκευής για την εφαρμογή. Το GPS χρησιμοποιείται όταν ένας χρήστης είναι οδηγός ενός

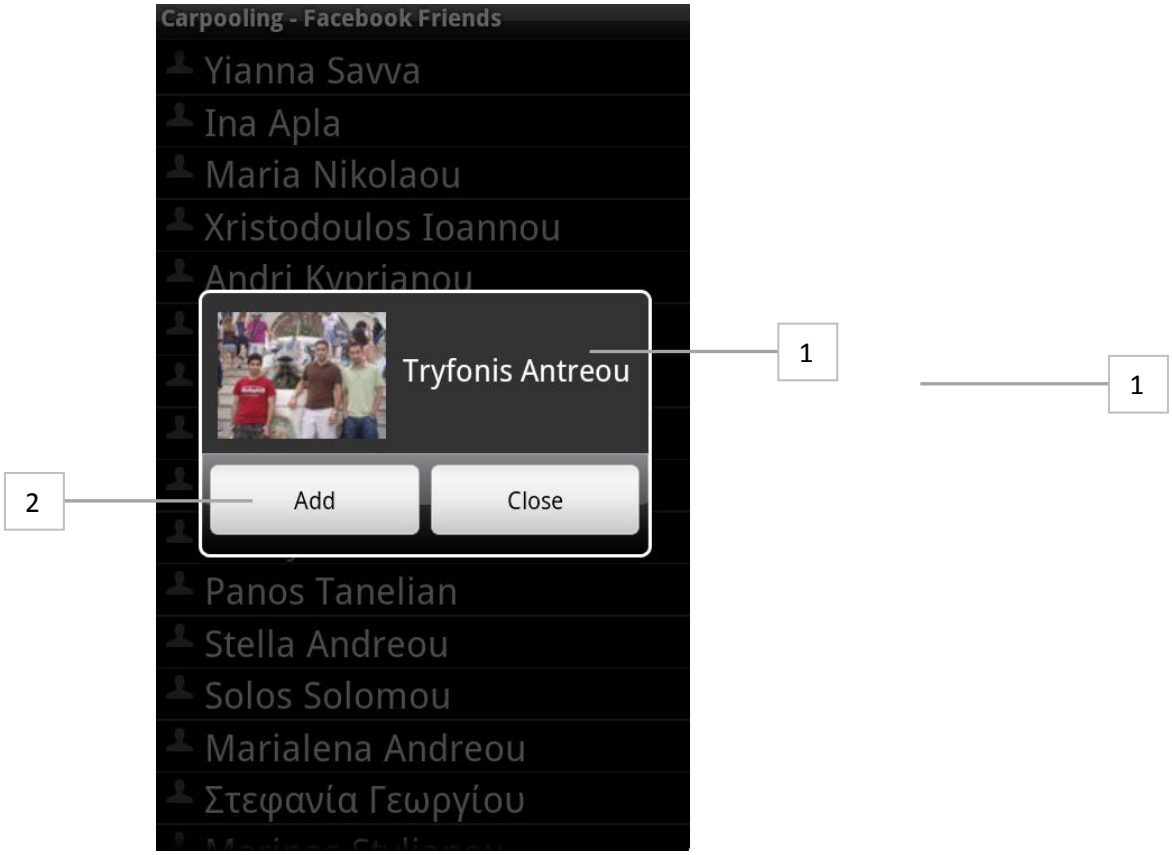
δρομολογίου και μέσω του GPS ενημερώνουμε τους επιβάτες του δρομολόγιου για το που βρίσκεται ο οδηγός την δεδομένη στιγμή.



Στιγμιότυπο Α.38

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει να προσθέσει τους φίλους του από το κοινωνικό δίκτυο Facebook από την προηγούμενη οθόνη που παρουσιάσαμε και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης μπορεί να δει τους φίλους που έχει στο κοινωνικό δίκτυο Facebook όπου και μπορεί να επιλέξει κάποιον από αυτούς.
2. Πατώντας το κουμπί Close ο χρήστης τερματίζει την τρέχουσα διαδικασία.
3. Πατώντας το κουμπί Add All ο χρήστης προσθέτει όλους του φίλους που έχει στο κοινωνικό δίκτυο Facebook στην εφαρμογή αν έχουν λογαριασμό.



Στιγμιότυπο Α.39

Το πιο πάνω στιγμιότυπο εμφανίζεται στον χρήστη όταν επιλέξει ένα από τους φίλους του από το κοινωνικό δίκτυο Facebook από την προηγούμενη οθόνη που παρουσιάσαμε και έχει τα ακόλουθα χαρακτηριστικά:

1. Ο χρήστης βλέπει το όνομα του χρήστη που επέλεξε και την φωτογραφία που έχει στο κοινωνικό δίκτυο.
2. Πατώντας το κουμπί Add ο χρήστης μπορεί να προσθέσει τον συγκεκριμένο φίλο του από το κοινωνικό δίκτυο στην εφαρμογή.

Παράρτημα Β

Σε αυτό παράρτημα θα παρουσιάσουμε κάποια κομμάτια κώδικα από την εφαρμογή που υλοποιήθηκε.

```
package com.example.carpool;
import java.util.ArrayList;
import com.harisThesis.ConnectionClient;
import com.harisThesis.FacebookManager;
import com.harisThesis.FacebookUser;
import net.xeomax.FBRocket.Facebook;
import net.xeomax.FBRocket.LoginListener;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.WindowManager;
import android.widget.Toast;

public class FacebookLogin extends Activity implements LoginListener
{
      static FacebookManager face;
      static boolean connected = false;
      static ArrayList<FacebookUser> friends;
      /** Called when the activity is first created. */
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
      setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRA
      IT);

      face = new FacebookManager();
      getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_I
      NPUT_STATE_ALWAYS_HIDDEN);
            try {
                  face.connectToFacebook(this, R.layout.main,
                  "Facebook Test","6ebd515d435fbc86d806e7e40b256756");
            } catch (Exception e) {
                  Toast.makeText(getApplicationContext(),"There was an
                  ERROR with facebook
                  connection",Toast.LENGTH_SHORT).show();
            }
      }

      @Override
      public void onLoginFail() {
            connected = false;
            face.onLoginFail();
      }

      @Override
      public void onLoginSuccess(Facebook facebook) {
            face.onLoginSuccess(facebook);
            connected = true;
```

```
            FacebookUser me=face.getMyProfile();

            long uid=me.getUid();


            friends=me.getFriends();
            Intent i = new Intent(FacebookLogin.this,
facefriend.class);

            startActivity(i);
            finish();
      }

}


package com.example.carpool;

import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

public class client {
      static String server_ip = "194.42.27.225";

      public static Socket server_con() {
            Socket socket=null;
            try {
             socket = new Socket(server_ip, 8787);

         } catch (UnknownHostException e) {
             e.printStackTrace();
         } catch (IOException e) {
             e.printStackTrace();
         }
         return socket;
      }

}



package com.example.carpool;

import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.Socket;
import java.net.URL;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.xml.sax.SAXException;

import com.google.android.maps.MapActivity;
import android.app.AlertDialog;
import android.app.Dialog;
```

```
import android.location.Geocoder;
import android.os.Bundle;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemSelectedListener;

public class set_route extends MapActivity {
      TextView res;
      EditText location;
      Geocoder geocoder = null;
      int[] routes1 = null;
      int[] routes2 = null;
      String[] names = new String[2];
      String location1 = null, location2 = null, temp = null;
      private int route_flex=0;
      boolean routeS= false;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.set_route);
            final EditText RouteF = (EditText)
             findViewById(R.id.RouteF);
            final EditText fromT = (EditText)
            findViewById(R.id.strP);
            fromT.setOnTouchListener(new View.OnTouchListener() {
                  @Override
                  public boolean onTouch(View v, MotionEvent event) {
                        String locationName =
                        fromT.getText().toString();
                        if (locationName.contains("Starting Point")) {
                              fromT.setText("");
                              fromT.setTextColor(Color.rgb(0, 0, 0));
                        }
                        return false;
                  }
            });

            final EditText toT = (EditText) findViewById(R.id.endP);
            toT.setOnTouchListener(new View.OnTouchListener() {
                  @Override
                  public boolean onTouch(View v, MotionEvent event) {
                        String locationName =
                        toT.getText().toString();
```

```
            if (locationName.contains("Ending Point")) {
                  toT.setText("");
                  toT.setTextColor(Color.rgb(0, 0, 0));
            }
            return false;
      }
});

geocoder = new Geocoder(this);
final Button button1 = (Button)
findViewById(R.id.search1);
button1.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
            EditText loc = (EditText)
            findViewById(R.id.strP);
            String locationName =
            loc.getText().toString();
            if (locationName.length() > 0 &&
            (!locationName.contains("Starting Point"))  &&
            locationName.length() > 0) {
                  location1 = locationName;
                  Intent i = new Intent(set_route.this,
                  point_view.class);
                  i.putExtra("routes", location1);
                  startActivityForResult(i, 1);
            } else {
                  Intent i = new Intent(set_route.this,
                   point_view.class);
                  startActivityForResult(i, 1);
            }

      }
});
final Button button2 = (Button)
findViewById(R.id.search2);
button2.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {

            EditText loc2 = (EditText)
             findViewById(R.id.endP);
            String locationName =
            loc2.getText().toString();
            if (locationName != null&&
            (!locationName.contains("Ending Point"))    &&
            locationName.length() > 0) {
                  location2 = locationName;
                  // names[1]= new String (location2);
                  Intent i = new Intent(set_route.this,
                   point_view.class);
                  i.putExtra("routes", location2);
                  startActivityForResult(i, 2);
            } else {
                  Intent i = new Intent(set_route.this,
                  point_view.class);
                  startActivityForResult(i, 2);
            }

      }
});
```

```
        final Button button3 = (Button)
        findViewById(R.id.mapView);
        button3.setOnClickListener(new View.OnClickListener() {
              public void onClick(View v) {
                    if ((routes1[0] != 0 && routes1[1] != 0)&&
                    (routes2[0] != 0 && routes2[1] != 0)) {
                          try{

route_flex=Integer.parseInt(RouteF.getText().toString());
                          }
                          catch(NumberFormatException nfe){
                                Toast msg =
                          Toast.makeText(getBaseContext(),"Give a
              valid number for Flexibility",Toast.LENGTH_SHORT);
                          msg.setGravity(Gravity.BOTTOM, 0, 60);
                                msg.show();
                                RouteF.setText("");
                          RouteF.setTextColor(Color.rgb(0, 0, 0));
                          }
                          int direction[] = new int[4];
                          direction[0] = routes1[0];
                          direction[1] = routes1[1];
                          direction[2] = routes2[0];
                          direction[3] = routes2[1];
                          Intent i = new Intent(set_route.this,
                           direction_map.class);
                          i.putExtra("routes", direction);
                          i.putExtra("names", names);
                          i.putExtra("flex", route_flex);
                          startActivity(i);

                    } else {
                          AlertDialog.Builder adb = new
                          AlertDialog.Builder(set_route.this);
                          adb.setTitle("Not available address");
                          adb.setMessage("Make sure all street and
                          city names are spelled correctly");
                          adb.setPositiveButton("Close",new
                          DialogInterface.OnClickListener() {
                                public void onClick(DialogInterface
                                dialog,int id) {
                                dialog.cancel();
                                }
                          });
                          adb.show();
                    }

              }
        });
        final Button button4 = (Button) findViewById(R.id.cont);
        button4.setOnClickListener(new View.OnClickListener() {
              public void onClick(View v) {

                    if (routes1==null || routes2==null) {
                          AlertDialog.Builder adb = new
                          AlertDialog.Builder(set_route.this);
                          adb.setTitle("Route Error");
                          adb.setMessage("Please choose the
                          Starting Point and the Ending Point");
```

```
                              adb.setPositiveButton("Close",new
                              DialogInterface.OnClickListener() {
                                    public void onClick(DialogInterface
                                    dialog, int id) {
                                          dialog.cancel();
                                    }
                              });
                              adb.show();
                        }
                        else {

                              try   {

      route_flex=Integer.parseInt(RouteF.getText().toString());
                              if (route_flex>=0)
                                    routeS=true;
                              else {
                                    Toast msg =
                              Toast.makeText(getBaseContext(),"Give a
valid number for Flexibility",Toast.LENGTH_SHORT);
                              msg.setGravity(Gravity.BOTTOM, 0, 60);
                              msg.show();
                              RouteF.setText("");

      RouteF.setTextColor(Color.rgb(0, 0, 0));
                              }
                              }
                              catch(NumberFormatException nfe){
                                    Toast msg =
                  Toast.makeText(getBaseContext(),"Give a valid number
                  for Flexibility",Toast.LENGTH_SHORT);
                  msg.setGravity(Gravity.BOTTOM, 0, 60);
                  msg.show();
                  RouteF.setText("");
                  RouteF.setTextColor(Color.rgb(0, 0, 0));
                  }

                  StringBuilder urlString = new StringBuilder();

      urlString.append("http://maps.google.com/maps?f=d&hl=en");
      urlString.append("&saddr=");// from

      urlString.append(Double.toString(routes1[0] / 1.0E6));
      urlString.append(",");

      urlString.append(Double.toString(routes1[1] / 1.0E6));
      urlString.append("&daddr=");// to

      urlString.append(Double.toString(routes2[0] / 1.0E6));
      urlString.append(",");

      urlString.append(Double.toString(routes2[1] / 1.0E6));

      urlString.append("&ie=UTF8&0&om=0&output=kml");
            Document doc = null;
            HttpURLConnection urlConnection = null;
            URL url = null;
            try {
                  url = new URL(urlString.toString());
```

```
                urlConnection = (HttpURLConnection)
                url.openConnection();

      urlConnection.setRequestMethod("GET");
      urlConnection.setDoOutput(true);
      urlConnection.setDoInput(true);
      urlConnection.connect();
      DocumentBuilderFactory dbf =
      DocumentBuilderFactory.newInstance();
      DocumentBuilder db = dbf.newDocumentBuilder();
      doc = db.parse(urlConnection.getInputStream());
                                 if
(doc.getElementsByTagName("GeometryCollection").getLength() > 0 &&
routeS) {
      int direction[] = new int[4];
      direction[0] = routes1[0];
      direction[1] = routes1[1];
      direction[2] = routes2[0];
      direction[3] = routes2[1];
      Intent mIntent = getIntent();
      mIntent.putExtra("route", direction);
      mIntent.putExtra("flex", route_flex);
      mIntent.putExtra("loc1", location1);
      mIntent.putExtra("loc2", location2);
      setResult(RESULT_OK, mIntent);
      finish();
      }
      } catch (MalformedURLException e) {
            e.printStackTrace();
      } catch (IOException e) {
            e.printStackTrace();
      } catch (ParserConfigurationException e) {
            e.printStackTrace();
      } catch (SAXException e) {
            AlertDialog.Builder adb = new
            AlertDialog.Builder(set_route.this);
            adb.setTitle("Error drawing path");
            adb.setMessage("We could not calculate driving
            directions"+ "\n\n" +"Google maps driving directions not
            provided for this Country");
            EditText loc2 = (EditText) findViewById(R.id.endP);
            EditText loc = (EditText) findViewById(R.id.strP);
            loc.setTextColor(Color.rgb(150, 150, 150));
            loc.setText("Starting Point");
            res = (TextView) findViewById(R.id.setStr);
            res.setTextColor(Color.rgb(255, 0, 0));
            res.setText("Set The Starting Point");
            res = (TextView) findViewById(R.id.endSet);
            res.setTextColor(Color.rgb(255, 0, 0));
            res.setText("Set The Ending Point");
            loc2.setTextColor(Color.rgb(150, 150, 150));
            loc2.setText("Ending Point");
            routes1=null; routes2=null;
            adb.setPositiveButton("Close",new
            DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int id)
                  {
                        dialog.cancel();
                  }
```

```
            });
            adb.show();
            e.printStackTrace();
            }
      }
      }
});

final Button button5 = (Button) findViewById(R.id.close);
button5.setOnClickListener(new View.OnClickListener() {
      public void onClick(View v) {
            finish();
            }
      });

RouteF.setOnTouchListener(new View.OnTouchListener() {
      @Override
      public boolean onTouch(View v, MotionEvent event) {
      String temp = RouteF.getText().toString();
      if (temp.contains("Flexibility")) {
            RouteF.setText("");
            RouteF.setTextColor(Color.rgb(0, 0, 0));
      }
      return false;
      }
});
}

      @Override
      protected void onActivityResult(int requestCode, int
      resultCode, Intent data) {
            super.onActivityResult(requestCode, resultCode, data);

            if (resultCode == RESULT_OK && requestCode == 1) {
                  routes1=null;
                  routes1 = data.getIntArrayExtra("route");

                  temp = new String(data.getStringExtra("location"));
                  names[0] = new String(data.getStringExtra("name"));
                  location = (EditText) findViewById(R.id.strP);
                  location.setTextColor(Color.rgb(000, 255, 000));
                  if (!temp.contains("NOT_SPECIFIED"))
                        location1 = temp;

                  location.setText(location1);
                  res = (TextView) findViewById(R.id.setStr);
                  res.setTextColor(Color.rgb(000, 255, 000));
                  res.setText("Starting Point Ok");
            } else if (resultCode == RESULT_OK && requestCode == 2) {
                  routes2=null;
                  routes2 = data.getIntArrayExtra("route");

                  temp = new String(data.getStringExtra("location"));
                  names[1] = new String(data.getStringExtra("name"));
                  location = (EditText) findViewById(R.id.endP);
                  location.setTextColor(Color.rgb(000, 255, 000));
                  if (!temp.contains("NOT_SPECIFIED"))
                        location2 = temp;
```

```
            location.setText(location2);

            res = (TextView) findViewById(R.id.endSet);
            res.setTextColor(Color.rgb(000, 255, 000));
            res.setText("Ending Point Ok");
      }
}

@Override
protected boolean isRouteDisplayed() {
      // TODO Auto-generated method stub
      return false;
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater = getMenuInflater();
      inflater.inflate(R.menu.menu, menu);
      return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
      switch (item.getItemId()) {
      case R.id.contacts:
            Intent i = new Intent(set_route.this,
             manage_contact.class);
            startActivity(i);
            break;
      case R.id.msgs:
            Intent i2 = new Intent(set_route.this,
            massages.class);
            startActivity(i2);
            break;
      case R.id.routes:
            Intent i3 = new Intent(set_route.this,
            routes_history.class);
            startActivity(i3);
            break;
      case R.id.request:
            Intent i4 = new Intent(set_route.this,
             request_history.class);
            startActivity(i4);
            break;
      case R.id.about: {
            AlertDialog.Builder adb = new
            AlertDialog.Builder(set_route.this);
            adb.setTitle("Carpooling Information");
            adb.setMessage("This application can help you to
            find easily common routes with your friends, and
            also to share the same car");
            adb.setPositiveButton("Close",new
            DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id)
            {
                  dialog.cancel();
            }
            });
            adb.setIcon(R.drawable.carpool);
```

```
                adb.show();
                break;
            }
            }
            return true;
      }

}



package com.example.carpool;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class sign_up extends Activity {

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.sign_up);


      getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_I
      NPUT_STATE_ALWAYS_HIDDEN);

            final Button button1 = (Button) findViewById(R.id.close);
            button1.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                        finish();
                  }
            });
            final Button button2 = (Button) findViewById(R.id.sign);
            button2.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                  EditText name = (EditText) findViewById(R.id.name);
                  EditText surname = (EditText)
                  findViewById(R.id.surname);
                  EditText username = (EditText)
```

```
findViewById(R.id.username);
EditText pass = (EditText) findViewById(R.id.pass);
EditText pass2 = (EditText)
findViewById(R.id.pass2);
EditText mobile = (EditText)
findViewById(R.id.mobilenum);
EditText address = (EditText)
findViewById(R.id.address);
EditText car_type = (EditText)
findViewById(R.id.cat_t);
String sname = name.getText().toString();
String ssurname = surname.getText().toString();
String susername = username.getText().toString();
String spass = pass.getText().toString();
String spass2 = pass2.getText().toString();
String smobile = mobile.getText().toString();
String saddress = address.getText().toString();
String scar_type = car_type.getText().toString();
if (sname.length() > 0 && ssurname.length() > 0
&& susername.length() > 0 && spass.length() > 0
&& spass2.length() > 0 && smobile.length() > 0
&& saddress.length() > 0) {
     if (spass.equals(spass2) &&
     spass2.length()>=4) {
     Socket sok = client.server_con();
     if (sok != null) {
          ObjectOutputStream oos;
          ObjectInputStream ois;
     try {
     oos = new
     ObjectOutputStream(sok.getOutputStream());
     oos.writeObject("signup");
     oos.writeObject(sname);
     oos.writeObject(ssurname);
     oos.writeObject(susername);
     oos.writeObject(spass);
     oos.writeObject(smobile);
     oos.writeObject(saddress);
     if (scar_type.length()>0)
     oos.writeObject(scar_type);
     else
     oos.writeObject("");
     ois = new
     ObjectInputStream(sok.getInputStream());
     try {
     String message = (String) ois.readObject();
     if (message.contentEquals("0")) {
     Toast msg =
     Toast.makeText(getBaseContext(),"Account is
     ready",Toast.LENGTH_SHORT);
     msg.setGravity(Gravity.CENTER, 0, 0);
     msg.show();
     finish();
     sok.close();
} else if (message.contentEquals("2")) {
     Toast msg =
     Toast.makeText(getBaseContext(),"Username in
     use",Toast.LENGTH_SHORT);
```

```
                        msg.setGravity(Gravity.CENTER, 0, 0);
                        msg.show();
                        username.setText("");
                        }
                    } catch (ClassNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    }
                    } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                    }
               } else {
                    Toast msg = Toast.makeText(getBaseContext(),"No
                    connection with the server",Toast.LENGTH_SHORT);
                    msg.setGravity(Gravity.CENTER, 0, 0);
                    msg.show();
               }
               }
               else {
               Toast msg = Toast.makeText(getBaseContext(),"Password is
               not correct Passwod must be at least 4 characters",
               Toast.LENGTH_LONG);
               msg.setGravity(Gravity.CENTER, 0, 0);
               msg.show();
               pass2.setText("");
               pass.setText("");
          }
          } else {
          Toast msg = Toast.makeText(getBaseContext(),"You must supply a
          value for all the fields",Toast.LENGTH_SHORT);
          msg.setGravity(Gravity.CENTER, 0, 0);
          msg.show();
          }
}
});
}


import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.UnsupportedEncodingException;
import java.lang.ClassNotFoundException;
import java.lang.Runnable;
import java.lang.Thread;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Timer;
import java.util.TimerTask;

public class server_connection  extends TimerTask{
     public static Connection con = null;
```

```
private ServerSocket server;
private int port = 8787;


public server_connection() {
      try {
            server = new ServerSocket(port);
      } catch (IOException e) {
            e.printStackTrace();
      }
}

public static void main(String[] args) throws
ClassNotFoundException, InstantiationException,
IllegalAccessException, SQLException,
UnsupportedEncodingException {

      mysql_connect mysql = new mysql_connect();
      con = mysql.connect_database();
      server_connection example = new server_connection();
      InetAddress thisIp = null;
      try {
            thisIp = InetAddress.getLocalHost();
      } catch (UnknownHostException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }
      System.out.println("IP:"+thisIp.getHostAddress());

   Timer timer = new Timer();
   TimerTask task = new Task();

   // delay in milliseconds before task is to be executed.
   long delay = 0;
   long period =  60*60*1000;
   //long period =  60*60;
   timer.schedule(task, delay, period);


      example.handleConnection();

}

public void handleConnection() {
      System.out.println("Waiting for client message...");
      while (true) {
            try {
                  Socket socket = server.accept();
                  new ConnectionHandler(socket);
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
}

public void run() {
      query_case.periodic_event();

}
```

```
}

class ConnectionHandler implements Runnable {
      private Socket socket;

      public ConnectionHandler(Socket socket) {
            this.socket = socket;
            Thread t = new Thread(this);
            t.start();

      }

      public void run() {
            try {

                  ObjectInputStream ois = new
                  ObjectInputStream(socket.getInputStream());
                  String message = (String) ois.readObject();
                  ObjectOutputStream oos = new
                  ObjectOutputStream(socket.getOutputStream());
                  query_case.selection(ois, oos, message);

                  ois.close();
                  oos.close();
                  System.out.println("Waiting for client message...");

            } catch (IOException e) {
                  e.printStackTrace();
            } catch (ClassNotFoundException e) {
                  e.printStackTrace();
            }
      }
}



import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

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

import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class search_algorithm {


      public static boolean isPossibleRoute (double srclat,double
      srclng,double destlat,double destlng,double src_lat,double
      src_lng,double end_lat,double end_lng,int flex, boolean flag3,
      boolean flag4){
             long startT = System.currentTimeMillis();
             long end=0;
            StringBuilder urlString = new StringBuilder();
```

```
urlString.append("http://maps.google.com/maps?f=d&hl=en");
urlString.append("&saddr=");// from
urlString.append(Double.toString((double) srclat));
urlString.append(",");
urlString.append(Double.toString((double) srclng));
urlString.append("&daddr=");// to
urlString.append(Double.toString((double) destlat ));
urlString.append(",");
urlString.append(Double.toString((double) destlng ));
urlString.append("&ie=UTF8&0&om=0&output=kml");
Document doc = null;
HttpURLConnection urlConnection = null;
URL url = null;
boolean flag1=flag3;
boolean flag2=flag4;
boolean iflag=false;
try {
      url = new URL(urlString.toString());
      urlConnection = (HttpURLConnection) url.openConnection();
      urlConnection.setRequestMethod("GET");
      urlConnection.setDoOutput(true);
      urlConnection.setDoInput(true);
      urlConnection.connect();
      DocumentBuilderFactory dbf =
      DocumentBuilderFactory.newInstance();
      DocumentBuilder db = dbf.newDocumentBuilder();
      doc = db.parse(urlConnection.getInputStream());
      end = System.currentTimeMillis();


      if (doc.getElementsByTagName
      ("GeometryCollection").getLength() > 0) {
      String path = doc.getElementsByTagName
      ("GeometryCollection").item(0).getFirstChild().getFirstCh
      ild().getFirstChild().getNodeValue();
      String[] pairs = path.split(" ");
      double lat=0.0;
      double lng=0.0;
      double  starting_str = CalculationByDistance(srclat,
      srclng,src_lat,src_lng);
      double starting_end=CalculationByDistance(srclat,
      srclng,end_lat,end_lng)+flex;
      double temp1=0;
      double temp2=0;
      double min=0;
      System.out.println(pairs.length);
      if (starting_str<=starting_end || (flag1 || flag2)) {
            int i=0;
            int i2=0;
            while(i < pairs.length && (flag1==false ||
            flag2==false)) {
            String[] lngLat = pairs[i].split(",");
            lat = Double.parseDouble(lngLat[1]);
            lng = Double.parseDouble(lngLat[0]);
                  if (flag1==false ) {

            temp1 = CalculationByDistance(lat,
            lng,src_lat,src_lng);
```

```
            if (temp1 <=flex && getDistance(lat,
            lng,src_lat,src_lng)<=flex)
            flag1=true;


            }
      if (flag2==false){
      temp2= CalculationByDistance(lat, lng,end_lat,end_lng);
      if (temp2 <=flex && getDistance(lat,
      lng,end_lat,end_lng)<=flex)
      flag2=true;
      }
      if (flag1 && flag2)
      return true;


      if (temp1<temp2 && flag1==false)
            min=temp1;
            else
            min=temp2;

            if (min!=0){
                  i2=(int) (min/0.35) + i;
            if (iflag)
                  break;
            else {
            if (i2==i)
                  i++;
            else
            i=i2;
      }

      if (i>pairs.length && !iflag) {
            i=pairs.length-1;
            lngLat = pairs[i].split(",");
            lat = Double.parseDouble(lngLat[1]);
            lng = Double.parseDouble(lngLat[0]);
            iflag=true;
            continue;
      }

}
      else
      i++;
}
if (CalculationByDistance(lat, lng,  destlat, destlng)>1) {
      if (isPossibleRoute(lat, lng, destlat, destlng, src_lat,
      src_lng, end_lat, end_lng, flex, flag1, flag2)){
      flag1=true;
      flag2=true;
      }
}
}
}
} catch (MalformedURLException e) {
      e.printStackTrace();
} catch (IOException e) {
      e.printStackTrace();
} catch (ParserConfigurationException e) {
```

```
            e.printStackTrace();
      } catch (SAXException e) {
            e.printStackTrace();
      }
      if (flag1 && flag2)
            return true;
      else
            return false;


      }
      public static double CalculationByDistance(double startP_lat,
      double startP_lng, double endP_lat, double endP_lng) {
            double lat1 = startP_lat  ;
            double lat2 = endP_lat ;
            double lon1 = startP_lng ;
            double lon2 = endP_lng ;
            double dLat = Math.toRadians(lat2 - lat1);
            double dLon = 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(dLon / 2)* Math.sin(dLon / 2);
            double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 -
a));
            return 6371 * c;
      }

      public static int getDistance (double startP_lat, double
      startP_lng, double endP_lat, double endP_lng) {
            double distance=0.0;

            if (startP_lat==endP_lat && startP_lng==endP_lng)
                  return 0;

            StringBuilder urlString = new StringBuilder();

      urlString.append("http://maps.google.com/maps?f=d&hl=en");
      urlString.append("&saddr=");// from
      urlString.append(Double.toString((double) startP_lat));
      urlString.append(",");
      urlString.append(Double.toString((double) startP_lng));
      urlString.append("&daddr=");// to
      urlString.append(Double.toString((double) endP_lat ));
      urlString.append(",");
      urlString.append(Double.toString((double) endP_lng ));
      urlString.append("&ie=UTF8&0&om=0&output=kml");
      Document doc = null;
      HttpURLConnection urlConnection = null;
      URL url = null;
      try {
            url = new URL(urlString.toString());
            urlConnection = (HttpURLConnection) url.openConnection();
            urlConnection.setRequestMethod("GET");
            urlConnection.setDoOutput(true);
            urlConnection.setDoInput(true);
            urlConnection.connect();

            DocumentBuilderFactory dbf =
            DocumentBuilderFactory.newInstance();
```

```
            DocumentBuilder db = dbf.newDocumentBuilder();
            doc = db.parse(urlConnection.getInputStream());
            if ( doc.getElementsByTagName("GeometryCollection")!=null
            && doc.getElementsByTagName(
            "GeometryCollection").getLength() > 0) {
            String path = doc.getElementsByTagName("
            GeometryCollection").item(0).getFirstChild().getFirstChil
            d() .getFirstChild().getNodeValue();
            String[] pairs = path.split(" ");
            String[] lngLat = pairs[0].split(",");

            double lat2 = Double.parseDouble(lngLat[1]);
            double lng2 = Double.parseDouble(lngLat[0]);
            double lat=0.0;
            double lng=0.0;
            int flag1=0;
            int flag2=0;

            for (int i = 1; i < pairs.length; i++)      {

                  lngLat = pairs[i].split(",");
                  lat=lat2;
                  lng=lng2;

                  lat2 = Double.parseDouble(lngLat[1]);
                  lng2 = Double.parseDouble(lngLat[0]);
                  distance = CalculationByDistance(lat, lng , lat2,
                  lng2) + distance;

                  }

            }

            }catch (MalformedURLException e) {
                  e.printStackTrace();
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (ParserConfigurationException e) {
                  e.printStackTrace();
            } catch (SAXException e) {

                  e.printStackTrace();
            }
            return (int) distance;
      }
}
```