Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Μηχανικών Πληροφορικής & Τηλεπικοινωνιών

# Κατασκευή Πληροφοριακού Συστήματος Διαχείρισης Στόλου Οχημάτων

Κεχαγιάς Απόστολος

Επιβλέπων: Δασυγένης Μηνάς

Κοζάνη, 10 Ιουνίου 2013

# Περιεχόμενα

# Κατάλογος σχημάτων

# Κατάλογος πινάκων

# Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον κ. Δασυγένη Μηνά για την εισήγηση της εργασίας, την εμπιστοσύνη που έδειξε στον πρόσωπό μου για την ανάληψή της και την πολύτιμη βοήθεια που μου πρόσφερε όταν χρειάστηκε.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου και όλους τους δικούς μου ανθρώπους για την ηθική υποστήριξη και συμπαράσταση καθ' όλη τη διάρκεια των σπουδών μου και κατά την εκτέλεση της εργασίας αυτής.

**Κεχαγιάς Απόστολος**

# Περίληψη

Διαχείριση Στόλου (Fleet management) καλείται το σύνολο των τεχνολογιών και των συστημάτων, το οποίο επιτρέπει σε μία επιχείρηση να έχει πλήρη έλεγχο των οχημάτων της. Ο τρόπος λειτουργίας ενός τέτοιου συστήματος αποσκοπεί στη βελτιστοποίηση αρκετών επιμέρους επιχειρηματικών διαδικασιών, στο διαχειριστικό έλεγχο και κατά συνέπεια, στη μείωση του κόστους και την καλύτερη κατανομή των πόρων της επιχείρησης.

Σκοπός αυτής της διπλωματικής εργασίας είναι να δημιουργηθεί ένα πληροφοριακό σύστημα διαχείρισης στόλου οχημάτων που θα αποτελείται από:

α) τις συσκευές καταγραφής, που τοποθετούνται στο όχημα και έχουν την δυνατότητα καταγραφής των συντεταγμένων και αποστολής αυτών σε κάποιο εξωτερικό σύστημα

β) τον εξυπηρετητή, που λαμβάνει τα δεδομένα των συσκευών καταγραφής και τα αποθηκεύει σε μια βάση δεδομένων

γ) την διαδικτυακή εφαρμογή, που επεξεργάζεται τα δεδομένα και εμφανίζει στον χρήστη πληροφορίες σχετικές με την λειτουργία των οχημάτων του

Αρχικά, γίνεται μια εισαγωγή στο θέμα της εργασίας παρουσιάζοντας τις τεχνολογίες που χρησιμοποιούνται στην διαχείριση στόλου. Το επόμενο βήμα είναι ο σχεδιασμός του συστήματος και η παρουσίαση της αρχιτεκτονικής του. Στην συνέχεια, πραγματοποιείται η υλοποίηση της διαδικτυακής εφαρμογής και ο προγραμματισμός των συσκευών καταγραφής. Τέλος, γίνεται αναφορά στα συμπεράσματα που προέκυψαν και στις μελλοντικές επεκτάσεις του συστήματος.

**Λέξεις κλειδιά:** διαχείρηση στόλου, όχημα, εντοπισμός οχήματος, παρακολούθηση οχήματος, χάρτης, GPS, GPRS

# Abstract

Fleet Management is the sum of technologies and systems which allows a company to have full control of its vehicles. The operation of such a system is designed to optimize several individual business processes to audit and consequently, reduce costs and better allocate the company's resources.

The scope of this thesis is to create a fleet management information system consisting of:

a) the recording apparatus fitted on a vehicle which has the ability to record coordinates and send them to an external system

b) the server that receives the data from recording devices and stores them in a database

c) the web application that processes the data and displays information το the user relevant to the operation of his vehicles

Initially, an introduction is made to the topic of work presenting the technologies used in fleet management. The next step is to design the system and present its architecture. Then we implement the web application and we program the recording devices. Finally, a reference is made to the conclusions and future extensions of the system.

**Keywords:** fleet management, vehicle, locate vehicle, track vehicle, map, GPS, GPRS

# Διάρθρωση κειμένου

**Κεφάλαιο 1**

Γίνεται μια εισαγωγή στο θέμα της διπλωματικής. Περιγράφεται η διαχείριση στόλου οχημάτων και όλες οι τεχνολογίες οι οποίες βοηθούν ώστε να πραγματοποιηθεί.

**Κεφάλαιο 2**

Το κεφάλαιο αυτό παρέχει πληροφορίες για τον σχεδιασμό της εφαρμογής, της βάσης δεδομένων και την μεταξύ τους επικοινωνία. Περιγράφονται οι απαιτήσεις για τον σχεδιασμό της βάσης και δίνεται λεπτομερής περιγραφή των οντοτήτων, δεδομένων και των συσχετίσεων που έχουν δημιουργηθεί.

**Κεφάλαιο 3**

Στο κεφάλαιο αυτό παρουσιάζεται η λειτουργία της διαδικτυακής εφαρμογής που αναπτύχθηκε και οι δυνατότητες που προσφέρει η εφαρμογή στον χρήστη και διαχειριστή.

**Κεφάλαιο 4**

Στο 4ο κεφάλαιο γίνεται η παρουσίαση των συσκευών καταγραφής. Αναλύεται ο τρόπος λειτουργίας τους και ο προγραμματισμός τους.

**Κεφάλαιο 5**

Στο κεφάλαιο αυτό παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ολοκλήρωση της εργασίας και τα κριτήρια που οδήγησαν στην επιλογή τους.

**Κεφάλαιο 6**

Στο 6ο κεφάλαιο γίνεται μια σύνοψη για την εργασία. Αναφέρονται οι μελλοντικές επεκτάσεις του συστήματος που αναπτύχθηκε τόσο σε υλικό, όσο και σε λογισμικό. Επίσης, αναφέρονται τα συμπεράσματα τα οποία προέκυψαν μετά την δημιουργία του.

# Κεφάλαιο 1

# Εισαγωγή

Το κεφάλαιο αυτό αποτελεί την εισαγωγή στην διαχείριση στόλου οχημάτων. Περιγράφεται ο τρόπος λειτουργίας και οι τεχνολογίες που χρησιμοποιούνται σε αυτή. Δηλαδή, η λειτουργία του GPS και του GSM. Έτσι, στο επόμενο κεφάλαιο μπορεί γίνει η ανάλυση και ο σχεδιασμός του συστήματος.

## 1.1 Διαχείριση στόλου οχημάτων

Η διαχείριση στόλου οχημάτων είναι η δυνατότητα που δίνεται σε μια επιχείριση ή οργανισμό, για την από απόσταση παρακολούθηση της λειτουργίας των οχημάτων τους. Ουσιαστικά δημιουργήθηκε για τις ανάγκες βελτιστοποίησης της εφοδιαστικής αλυσίδας δηλαδή, της πορείας ενός προϊόντος από μια επιχείριση στον καταναλωτή. Αρχικά, η παρακολούθηση περιοριζόταν μόνο στην θέση των οχημάτων, όμως γρήγορα προστέθηκαν και άλλες δυνατότητες όπως ταχύτητα, καθορισμός δρομολογίων, ανίχνευση στάσεων, παρακολούθηση κινητήρα, κατανάλωση του οχήματος και παρακολούθηση της στάθμης των καυσίμων.

Ο σκοπός δηλαδή της διαχείρισης στόλου είναι να αξιοποιηθούν όλα τα δεδομένα των οχημάτων, ώστε να βελτιστοποιηθεί γενικότερα ο τρόπος λειτουργίας της επιχείρησης. Αυτό επιτυγχάνεται με την παροχή αποδοτικότερων τρόπων υπολογισμού κρίσιμων μεγεθών της επιχείρησης, με ταυτόχρονη αξιοποίηση σύγχρονων και αποτελεσματικών εργαλείων.

Οι λύσεις διαχείρισης στόλου, βρίσκουν εφαρμογή σε πλήθος επιχειρήσεων, ανεξάρτητα από το μέγεθός τους. Εφαρμογές υπάρχουν σε μεταφορικές και τουριστικές επιχειρήσεις, μέσα μεταφοράς, ασθενοφόρα, σώματα ασφαλείας, ένοπλες δυνάμεις ή ακόμα και σε ιδιωτικές εταιρείες ασφάλειας. Σε εταιρείες ενοικίασης αυτοκινήτων και σε εταιρείες ταχυμεταφορών.

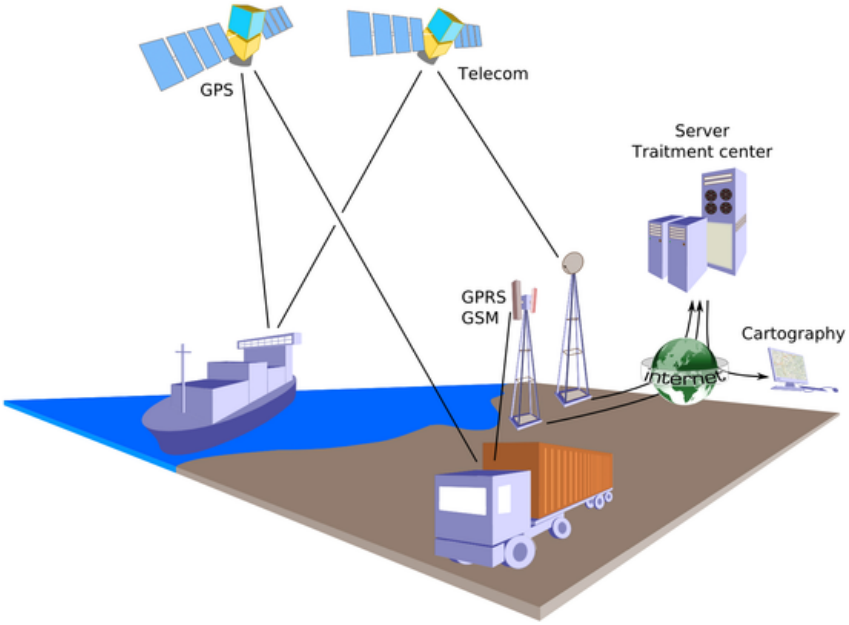
### 1.1.1 Αρχιτεκτονική συστημάτων

Όλα αυτά δεν θα ήταν δυνατό να συμβούν, αν δεν υπήρχαν οι κατάλληλες τεχνολογίες για να υποστηρίξουν το εγχείρημα. Για την βασική λειτουργία ενός τέτοιου συστήματος,

απαιτείται η συλλογή πληροφοριών που αφορά τα οχήματα και η μετάδοση τους σε ένα ειδικά διαμορφωμένο λογισμικό, για την επεξεργασία και την εξαγωγή αποτελεσμάτων.

Τα δεδομένα μεταβιβάζονται συνήθως μέσω του δικτύου GSM, είτε με την υπηρεσία μηνυμάτων SMS, είτε με την χρήση του GPRS. Βέβαια, υπάρχουν περιπτώσεις στις οποίες δεν γίνεται να χρησιμοποιηθεί το δίκτυο GSM. Για παράδειγμα, ένα πλοίο στον ειρηνικό ωκεανό είναι αδύνατο να έχει πρόσβαση στο δίκτυο εκτός και αν βρίσκεται κοντά σε ξηρά. Σε τέτοιες περιπτώσεις χρησιμοποιούνται δορυφορικές συνδέσεις.

Ο προσδιορισμός της θέσης γίνεται συνήθως χρησιμοποιώντας ένα δίκτυο δορυφόρων. Στην πλειονότητα το περιπτώσεων χρησιμοποιείται το δίκτυο GPS, ωστόσο υπάρχουν και άλλα παρόμοια συστήματα για τον ίδιο σκοπό (GLONASS, Compass, IRNSS).




Σχήμα 1.1: Η αρχιτεκτονική ενός συστήματος διαχείρισης στόλου [18]

### 1.1.2 Πλεονεκτήματα

Η όλο και μεγαλύτερη ενσωμάτωση των συστημάτων διαχείρισης στόλου στον επιχειρηματικό κλάδο, υποδηλώνει ότι τα πλεονεκτήματα της χρησιμοποίησής του, είναι πολλαπλά και σημαντικά. Τα κυριότερα από αυτά είναι:

- αυτοματοποιημένος εντοπισμός των οχημάτων
- αποδοτικότερη δρομολόγηση των οχημάτων
- δυνατότητα προσθήκης περιορισμών ασφαλείας (φράκτης, ακινητοποίηση οχήματος σε περίπτωση κλοπής)

- αυτόματη εξαγωγή δεδομένων για την λειτουργία του στόλου
- μείωση κατανάλωσης καυσίμων
- αποτελεσματικότερη αντιμετώπιση έκτακτων καταστάσεων
- παροχή αισθήματος ασφάλειας στους οδηγούς
- αύξηση της απόδοσης του στόλου οχημάτων
- μείωση του κόστους λειτουργίας

## 1.2 GPS

Το παγκόσμιο σύστημα προσδιορισμού θέσης (GPS) είναι ένα δίκτυο δορυφόρων που κινούνται σε τροχιά γύρω από τη γη, σε σταθερά σημεία πάνω από τον πλανήτη και μεταδίδουν σήματα προς την γη. Τα σήματα αυτά, φέρουν κώδικα χρόνου και σημείο γεωγραφικών δεδομένων, που παρέχουν στους χρήστες τη δυνατότητα να εντοπίζουν την ακριβή τους θέση, την ταχύτητα και την ώρα σε οποιοδήποτε σημείο του πλανήτη.

### 1.2.1 Ιστορικό

Το GPS σχεδιάσθηκε για στρατιωτικές και κατασκοπευτικές εφαρμογές κατά την περίοδο της κορύφωσης του Ψυχρού Πολέμου, τη δεκαετία του 1960. Η ιδέα προέκυψε μετά την εκτόξευση του Σοβιετικού διαστημόπλοιου Sputnik το 1957. Αναπτύχθηκε από το Υπουργείο Άμυνας των Ηνωμένων Πολιτειών και η επίσημη ονομασία του είναι NAVSTAR GPS, την οποία έδωσε ο Mr. John Walsh. Τη διαχείριση του συνόλου των δορυφόρων έχει η Αμερικανική Αεροπορία και το συνολικό κόστος συντήρησης του συστήματος ανέρχεται σε $400 εκατομμύρια το χρόνο.

Το πρώτο σύστημα που τέθηκε σε τροχεία ήταν το Transit στις αρχές τις δεκαετίας του 60. Αυτό αποτελούνταν από πέντε δορυφόρους και αρχικά δοκιμάσθηκε από το πολεμικό ναυτικό των ΗΠΑ. Με μόλις πέντε δορυφόρους σε τροχιά γύρω από τη γη, δόθηκε στα πλοία η δυνατότητα να προσδιορίζουν τη θέση τους, μία φορά κάθε ώρα. Το 1967, το Transit διαδέχθηκε ο δορυφόρος Timation που απέδειξε ότι στο διάστημα μπορούσαν λειτουργούν εξαιρετικά ακριβή ατομικά ρολόγια. Στη συνέχεια, το σύστημα GPS αναπτύχθηκε γρήγορα, με συνολικά 11 δορυφόρους "Block I", που τέθηκαν σε τροχιά σε μια περίοδο μεταξύ του 1978 και του 1985.

Ωστόσο, η κατάρριψη του κορεατικού επιβατικού αεροσκάφους της πτήσης 007 το 1983 από την ΕΣΣΔ οδήγησε την κυβέρνηση Reagan στις ΗΠΑ, να διαθέσει το σύστημα για πολιτικές εφαρμογές, έτσι ώστε αεροσκάφη, πλοία και μέσα μεταφοράς σε ολόκληρο τον κόσμο να μπορούν να προσδιορίζουν τη θέση τους και να αποφεύγουν την τυχαία εκτροπή τους σε απαγορευμένες ξένες επικράτειες.

Η χρήση του NAVSTAR επιτράπηκε το 1983 και σε πολιτικούς χρήστες. Η ακρίβεια για τους πολιτικούς χρήστες ήταν σκόπιμα υποβαθμισμένη σε περίπου 100 μέτρα, χρη-

σιμοποιώντας ένα σύστημα, γνωστό ως επιλεκτική διαθεσιμότητα, η οποία και απενεργοποιήθηκε στα τέλη του 2000.

Οι ΗΠΑ αναβάθμιζαν συνεχώς το σύστημα προσθέτοντας περισσότερους δορυφόρους σε τροχιά. Αυτή η διαδικασία καθυστέρησε, μετά την καταστροφή του διαστημικού λεωφορείου SS Challenger το 1986. Μόλις το 1989 τέθηκαν σε τροχιά οι πρώτοι δορυφόροι Block II. Το καλοκαίρι του 1993, οι ΗΠΑ έθεσαν σε τροχιά τον 24ο δορυφόρο Navstar, ο οποίος ολοκλήρωσε τη σύγχρονη ομάδα δορυφόρων GPS, ένα δίκτυο 24 δορυφόρων, γνωστό σήμερα ως το παγκόσμιο σύστημα προσδιορισμού θέσης ή GPS. Οι 21 από τους δορυφόρους αυτής της ομάδας ήταν ενεργοί ανά πάσα στιγμή, ενώ οι άλλοι 3 λειτουργούσαν ως εφεδρεία. Το σημερινό δίκτυο GPS διαθέτει 32 ενεργούς δορυφόρους στην ομάδα GPS. Οι επιπρόσθετοι αυτοί δορυφόροι βελτίωσαν την ακρίβεια των υπολογισμών των δεκτών GPS παρέχοντας πλεονάζουσες μετρήσεις αλλά και μεγαλύτερη αξιοπιστία και διαθεσιμότητα.

Σήμερα, το GPS χρησιμοποιείται για πολλές εφαρμογές πλοήγησης, κατάρτισης δρομολογίων για οδηγούς, χαρτογράφησης, σεισμογραφικής έρευνας, κλιματικών μελετών και παιχνιδιών αναζήτησης θησαυρών που είναι γνωστά ως γεωαναζήτηση.

### 1.2.2 Το λειτουργικά μέρη του συστήματος

Το GPS [20] αποτελείται από τρία βασικά λειτουργικά μέρη: το διαστημικό τμήμα, το επίγειο τμήμα και το τμήμα των χρηστών του συστήματος.

#### Το διαστημικό τμήμα

Το διαστημικό τμήμα του συστήματος περιλαμβάνει τους δορυφόρους που βρίσκονται σε τροχιά. Στο σχεδιασμό του συστήματος ορίστηκε ότι θα λειτουργούν 24 δορυφόροι, 8 για κάθε ένα από τα 3 τροχιακά επίπεδα, αλλά αυτό τροποποιήθηκε σε 6 τροχιές με 4 δορυφόρους η κάθε μία. Οι τροχιές αυτές έχουν για κέντρο τους το κέντρο της γης και κλίση περίπου 55° αναφορικά με τον γήινο ισημερινό. Ολοκληρώνουν μια τροχιά σε περίοδο 12 ωρών (δύο τροχιές ανά ημέρα) σε ύψος περίπου 11.500 μιλίων, ταξιδεύοντας με ταχύτητα 9.000 μίλια/ώρα (3,9 χιλιόμετρα ανά δευτερόλεπτο ή 14.000 χλμ. την ώρα). Οι τροχιές είναι έτσι σχεδιασμένες, ώστε τουλάχιστον 6 δορυφόροι να είναι πάντα ορατοί από σχεδόν κάθε σημείο της γήινης επιφάνειας.

#### Το επίγειο τμήμα ελέγχου

Το επίγειο τμήμα ελέγχου του GPS αποτελείται από ένα δίκτυο σταθμών οι οποίοι βρίσκονται διασκορπισμένοι ανά την υφήλιο. Ο ρόλος τους είναι ο έλεγχος της εύρυθμης λειτουργίας των δορυφόρων καθώς και η διόρθωση των σφαλμάτων που προκύπτουν σε αυτούς. Οι έλεγχοι που πραγματοποιούνται αφορούν τη σωστή τους ταχύτητα και υψόμετρο και την κατάσταση της επάρκειάς τους σε ηλεκτρική ενέργεια. Παράλληλα, εφαρμόζονται όλες οι διορθωτικές ενέργειες που αφορούν στο σύστημα χρονομέτρησης των δορυφόρων, ώστε να αποτρέπεται η παροχή λανθασμένων πληροφοριών στους χρήστες του συστήματος.

Το τμήμα αποτελείται από ένα επανδρωμένο κέντρο και ακόμα τέσσερα, μη επανδρωμένα, που βρίσκονται διάσπαρτα στον πλανήτη, στις περιοχές

- Κολοράντο (Ηνωμένες Πολιτείες της Αμερικής)
- Χαβάη (Ανατολικός Ειρηνικός Ωκεανός)
- Ascension Island (Ατλαντικός Ωκεανός)
- Diego Garcia (Ινδικός Ωκεανός)
- Kwajalein (Δυτικός Ειρηνικός Ωκεανός)


Σχήμα 1.2: Οι επίγειοι σταθμοί του GPS

Ο κυριότερος σταθμός βάσης είναι αυτός του Κολοράντο, ο οποίος είναι μάλιστα και ο μοναδικός που βρίσκεται στην ξηρά. Αυτός έχει αναλάβει τον έλεγχο της σωστής λειτουργίας των υπολοίπων τεσσάρων σταθμών, καθώς και τον συντονισμό τους. Όλοι είναι εξοπλισμένοι με υψηλής ακρίβειας δέκτες GPS και χρονόμετρα ατομικών ταλαντωτών κεσίου. Τρεις από αυτούς έχουν και επίγειες κεραίες για την αποστολή πληροφοριών στους δορυφόρους. Η διάταξή των σταθμών αυτών δεν είναι τυχαία. Ακολουθούν μια γραμμή παράλληλη με τα γεωγραφικά μήκη της γης.

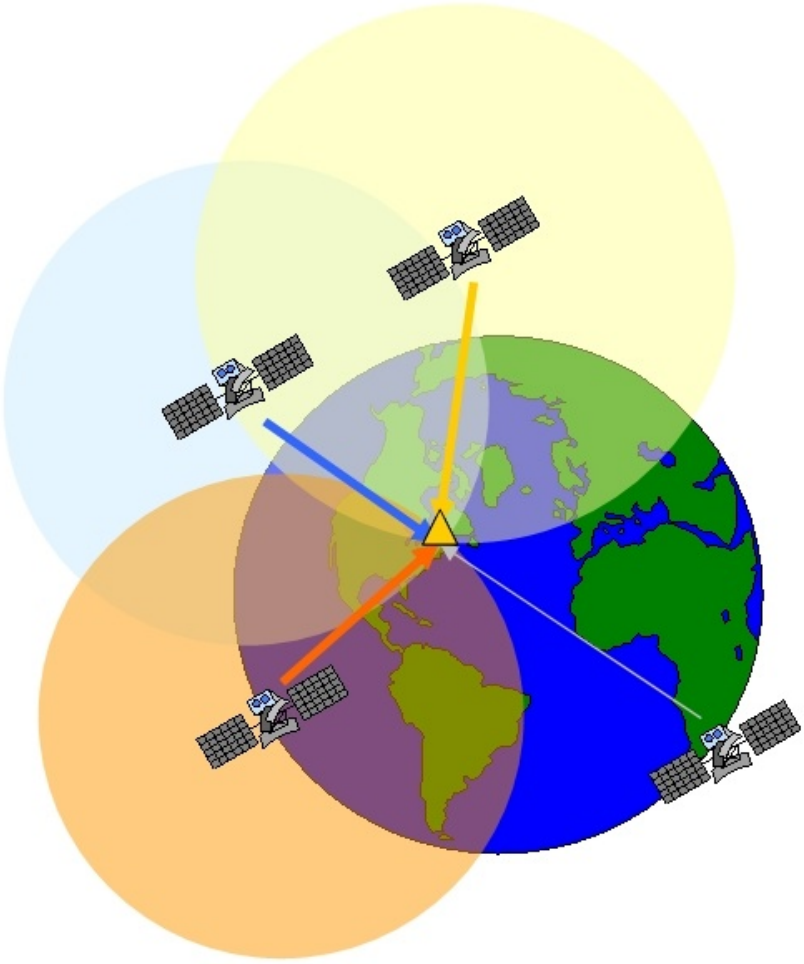
### Το τμήμα των χρηστών

Το τμήμα των χρηστών απαρτίζεται από τους χιλιάδες χρήστες των δεκτών GPS παγκοσμίως. Γενικά οι δέκτες αποτελούνται από μια κεραία, που συντονίζεται στις συχνότητες που εκπέμπουν οι δορυφόροι, από επεξεργαστές λήψης σημάτων και από ένα υψηλής σταθερότητας χρονόμετρο (συχνά πρόκειται για έναν ταλαντωτή κρυστάλλου). Μπορεί ακόμη να είναι εφοδιασμένοι με οθόνη, για την επίδειξη στο χρήστη, πληροφοριών θέσης και ταχύτητας. Για να προσφέρουν όσο το δυνατόν περισσότερες πληροφορίες, οι δέκτες συνδυάζονται με ειδικό λογισμικό, που προβάλλει ένα χάρτη στην οθόνη της συσκευής GPS. Πρόκειται δηλαδή, για λογισμικό που λαμβάνει από τους δορυφόρους τις πληροφορίες για το στίγμα του σημείου στο οποίο βρίσκεται ο δέκτης και τις μετατρέπει σε κατανοητή μορφή, πληροφορώντας το χρήστη για την ακριβή γεωγραφική του θέση.

### 1.2.3 Προσδιορισμός θέσης

Ένας χρήστης GPS χρησιμοποιεί τον τριπλευρισμό, [8] ώστε να προσδιορίσει τη θέση του στην επιφάνεια της γης. Η θέση προκύπτει χρονομετρώντας τα σήματα τουλάχιστον τριών δορυφόρων του παγκόσμιου συστήματος προσδιορισμού θέσης.

Κάθε δορυφόρος της ομάδας GPS μεταδίδει περιοδικά σήματα μαζί με ένα σήμα χρόνου. Τα σήματα αυτά λαμβάνονται από συσκευές GPS, οι οποίες τα χρησιμοποιούν για να υπολογίσουν την απόσταση μεταξύ της συσκευής και κάθε δορυφόρου. Αυτό γίνεται με βάση τη διαφορά χρόνου μεταξύ της ώρας αποστολής και της ώρας λήψης του σήματος. Η διαφορά αυτή προκύπτει από την καθυστέρηση λήψης του σήματος, λόγω της απόστασης των δορυφόρων από την γη, οι οποίοι βρίσκονται σε ύψος δεκάδων χιλιάδων χιλιομέτρων.



Σχήμα 1.3: Η μέθοδος τριπλευρισμού για τον προσδιορισμό θέσης

Όταν μια συσκευή GPS έχει προσδιορίσει τις αποστάσεις τουλάχιστον τριών δορυ-

φόρων, μπορεί να εκτελέσει υπολογισμούς τριπλευρισμού για τον προσδιορισμό θέσης. Αυτή η διαδικασία μπορεί να παραλληλιστεί με την εύρεση της θέσης ενός σημείου χρησιμοποιώντας διαβήτη, όταν υπάρχει ακριβής γνώση των αποστάσεων από τρία διαφορετικά ορόσημα. Αν λοιπόν, σχεδιαστούν κύκλοι με κέντρο κάθε ένα από τα ορόσημα και ακτίνα την απόσταση από αυτά, δημιουργείται μια κοινή περιοχή μεταξύ τους. Στο σημείο αλληλοκάλυψης των τριών αυτών κύκλων, εντοπίζεται η ζητούμενη θέση.

Οι υπολογισμοί πραγματοποιούνται σε τρεις διαστάσεις με ένα εικονικό τρισδιάστατο διαβήτη, έτσι ώστε η θέση να είναι στο σημείο αλληλοκάλυψης των τριών σφαιρών με ακτίνα που προσδιορίζεται από την απόσταση του καθένα από τους τρεις δορυφόρους. Εάν η συσκευή GPS μπορεί να εντοπίσει έναν τέταρτο δορυφόρο, τότε παρέχεται η δυνατότητα επιβεβαίωσης των μετρήσεων.

### 1.2.4 Κώδικες PRN

Το αρχικό σχέδιο GPS περιλαμβάνει δύο κώδικες μέτρησης ψευδοαποστάσεων: τον Coarse/Acquisition (C/A), ο οποίος είναι ελεύθερα διαθέσιμος στο κοινό και τον κώδικα (P), που είναι διαθέσιμος μόνο σε εξουσιοδοτημένους χρήστες.

#### Κώδικας C/A

Ο C/A κώδικας είναι μια ντετερμινιστική ακολουθία μήκους 1.023 bit που ονομάζεται ψευδοτυχαίος θόρυβος (pseudorandom noise) ή ψευδοτυχαία δυαδική ακολουθία (pseudorandom binary sequence) PN ή PRN, η οποία μεταδίδεται με ρυθμό 1.023 megabits ανά δευτερόλεπτο (Mbit/s) και επαναλαμβάνεται κάθε χιλιοστό του δευτερολέπτου. Αυτές οι ακολουθίες ταιριάζουν, ή σχετίζονται στενά, μόνο όταν είναι ακριβώς ευθυγραμμισμένες. Κάθε δορυφόρος εκπέμπει ένα μοναδικό κώδικα PRN, ο οποίος δεν συσχετίζεται καλά με τον κώδικα PRN οποιουδήποτε άλλου δορυφόρου. Με άλλα λόγια, οι κώδικες PRN είναι ιδιαίτερα ορθογώνιοι μεταξύ τους. Αυτή είναι μια μορφή της διαίρεσης κώδικα πολλαπλής πρόσβασης (CDMA), επιτρέποντας στο δέκτη να αναγνωρίζει πολλούς δορυφόρους στην ίδια συχνότητα.

#### Κώδικας P

Ο κώδικας P είναι επίσης κώδικας PRN. Ωστόσο, η ακολουθία PRN κάθε δορυφόρου έχει μήκος 6,1871 × 1012 bits (6,187,100,000,000 bits, 720.213 gigabytes) και επαναλαμβάνεται μια φορά την εβδομάδα με ρυθμό 10,23 Mbit/s. Η ακολουθία αυτή, ήταν τόσο μεγάλη και πολύπλοκη, που πίστευαν ότι ο δέκτης δεν μπορεί να την αποκτήσει και να συγχρονιστεί άμεσα, χρησιμοποιώντας μόνο αυτό το σήμα. Ήταν αναμενόμενο ότι ο δέκτης θα πρέπει πρώτα να κλειδώσει, χρησιμοποιώντας τον σχετικά απλό C/A κώδικα και στη συνέχεια, μετά από την λήψη της τρέχουσας ώρας και μιας θέσης κατά προσέγγιση, να γίνει συγχρονισμός με το κώδικα P.

Μολονότι τα PRNs του κώδικα C/A είναι μοναδικά για κάθε δορυφόρο, στην πραγματικότητα πρόκειται για τμήματα του κώδικα P. Σε κάθε δορυφόρο ανατίθεται ένα τέτοιο τμήμα και αυτός τον εκπέμπει επανειλημμένα.

| Subframes | Words | Description |
|---|---|---|
| 1 | 1-2 | TLM and HOW |
| | 3-10 | Satellite clock, GPS time relationship |
| 2/3 | 1-2 | TLM and HOW |
| | 3-10 | Ephemeris (precise satellite orbit) |
| 4/5 | 1-2 | TLM and HOW |
| | 3-10 | Almanac component |

Πίνακας 1.1: Δομή μηνύματος πλοήγησης

Για να αποτραπεί σε μη εξουσιοδοτημένους χρήστες να τον χρησιμοποιούν ή να παρεμβαίνουν στο στρατιωτικό σήμα, μέσω μιας διαδικασίας που ονομάζεται spoofing, αποφασίστηκε η κρυπτογράφηση του. Για τον σκοπό αυτό, ο κώδικας διαμορφώνεται με τον κώδικα W, μια ειδική ακολουθία κρυπτογράφησης, για τη δημιουργία του κώδικα Y. Ο κώδικας Y είναι αυτός που μεταδίδεται από τους δορυφόρους από τότε που εφαρμόστηκε αυτή η δικλείδα ασφαλείας. Το κρυπτογραφημένο σήμα αναφέρεται ως κώδικας P(Y).

Οι λεπτομέρειες του κώδικα W διατηρούνται μυστικές, αλλά είναι γνωστό ότι εφαρμόζεται στον κώδικα P στην συχνότητα περίπου 500 kHz. Έτσι ο ρυθμός του σήματος επιβραδύνεται κατά ένα συντελεστή περίπου 20. Αυτό επέτρεψε τις εταιρείες να αναπτύξουν προσεγγίσεις για την παρακολούθηση του P(Y) σήματος, χωρίς την γνώση του κώδικα W.

### 1.2.5 Το μήνυμα πλοήγησης

Το μήνυμα πλοήγησης είναι ουσιαστικά ένα πλαίσιο 1500 bit. Αυτό χωρίζεται σε πέντε υποπλαίσια των 300 τα οποία μεταδίδονται με ρυθμό 50 bit/s. Έτσι απαιτούνται 6 δευτερόλεπτα για την μετάδοση του καθενός. Κάθε πλαίσιο φέρει την ώρα GPS.

Το υποπλαίσιο 1 περιέχει τον αριθμό της εβδομάδας και πληροφορίες για την διόρθωση χρονικών διαφορών καθώς και την κατάσταση του δορυφόρου. Τα υποπλαίσια 2 και 3 περιέχουν τις πληροφορίες των εφημερίδων (ephemeris data). Τα τελευταία δύο υποπλαίσια 4 και 5 περιέχουν στοιχεία από το ημερολόγιο (almanac). Το συνολικό ημερολόγιο προκύπτει από την λήψη 25 πλαισίων αφού κάθε πλαίσιο έχει το 1/25 από αυτό. Άρα, με αυτό τον ρυθμό η λήψη του ημερολογίου από έναν δορυφόρο μπορεί να πραγματοποιηθεί σε 12.5 λεπτά.

Όπως φαίνεται και στον πίνακα 1.1, κάθε υποπλαίσιο μπορεί να διαιρεθεί σε 10 λέξεις. Η πρώτη λέξη είναι το TLM (Telemetry Word), η οποία επιτρέπει στο δέκτη, να ανιχνεύσει την αρχή ενός υποπλαισίου και να καθορίσει την ώρα του ρολογιού του δέκτη κατά την οποία το υποπλαίσιο αρχίζει. Η αμέσως επόμενη λέξη είναι η HOW (Handover Word), η οποία δίνει το χρόνο GPS (στην πραγματικότητα τη στιγμή που το πρώτο κομμάτι από το επόμενο υποπλαίσιο θα μεταδοθεί) και προσδιορίζει το συγκεκριμένο υποπλαίσιο μέσα σε ένα ολοκληρωμένο πλαίσιο. Οι υπόλοιπες οκτώ λέξεις του υποπλαισίου περιέχουν τα πραγματικά δεδομένα που αφορούν το υποπλαίσιο.

### 1.2.6 Πηγές σφάλματος

Υπάρχουν αρκετοί παράγοντες οι οποίοι μπορεί να υποβαθμίσουν το σήμα GPS και επομένως, να επηρεάσουν την ακρίβεια.

Το δορυφορικό σήμα επιβραδύνεται καθώς περνά μέσα από την ατμόσφαιρα. Προκαλούνται καθυστερήσεις από την ιονόσφαιρα και την τροπόσφαιρα. Το σύστημα GPS χρησιμοποιεί ένα ενσωματωμένο μοντέλο, που υπολογίζει το μέσο ποσό της καθυστέρησης, για την εν μέρει διόρθωση του σφάλματος.

Οι πολλαπλοί αντικατοπτρισμοί μπορούν επίσης να δημιουργήσουν πρόβλημα. Το σήμα GPS μπορεί να αντικατοπτρίζεται σε αντικείμενα, όπως τα ψηλά κτίρια ή μεγάλοι βράχοι και άλλες επιφάνειες, πριν φτάσει στον δέκτη. Αυτό το φαινόμενο αυξάνει το χρόνο λήψης του σήματος, προκαλώντας σφάλματα στον υπολογισμό του χρόνου, άρα της απόστασης του δορυφόρου και τελικά της θέσης.

Η ακρίβεια της μέτρησης του χρόνου πάντα ήταν πρόβλημα και αυτό δεν θα μπορούσε να μην επηρεάσει το GPS, του οποίου η λειτουργία βασίζεται σε χρονικά δεδομένα. Το ενσωματωμένο ρολόι του δέκτη μπορεί να μην είναι τόσο ακριβές όσο τα ατομικά ρολόγια των δορυφόρων GPS. Ως εκ τούτου, μπορεί να υπάρξουν πολύ μικρά χρονικά λάθη.

Τα λάθη στις τροχιές των δορυφόρων είναι πολύ συνηθισμένα. Για τον λόγο αυτό οι τροχιές τους παρακολουθούνται και διορθώνονται ανά τακτά χρονικά διαστήματα. Τα τροχιακά λάθη μπορούν να δημιουργήσουν ανακρίβειες για τις θέσεις των δορυφόρων από την εφημερίδα (ephemeris data) και αυτό μπορεί να εισάγει σφάλμα στον υπολογισμό της θέσης.

Ο αριθμός ορατών δορυφόρων παίζει πολύ μεγάλο ρόλο στην ακρίβεια του υπολογισμού θέσης. Όσο περισσότερους δορυφόρους μπορεί να εντοπίσει ένας δέκτης GPS, τόσο καλύτερο θα είναι και το αποτέλεσμα τον υπολογισμών.

Το περιβάλλον χρήσης. Κτίρια, μορφολογία εδάφους, ηλεκτρονικές παρεμβολές, και πυκνά φυλλώματα μπορούν να εμποδίσουν τη λήψη του σήματος, προκαλώντας προβλήματα. Οι μονάδες GPS συνήθως δεν θα λειτουργήσουν σε εσωτερικούς χώρους, υποβρύχια ή υπόγεια.

Η γεωμετρία των δορυφόρων μπορεί σε μερικές περιπτώσεις να προκαλέσει προβλήματα. Υπάρχουν δυο πιθανές καταστάσεις:

- ιδανική γεωμετρία
- κακή γεωμετρία

Το πρόβλημα της ιδανικής γεωμετρίας συναντάται, όταν ένας δορυφόρος βρίσκεται σε ευρεία γωνία σε σχέση με άλλους. Κακή γεωμετρία υπάρχει, όταν οι δορυφόροι βρίσκονται σε μια γραμμή ή πολύ κοντά μεταξύ τους.

Η σκόπιμη υποβάθμιση του δορυφορικού σήματος μεσώ της επιλεκτικής διαθεσιμότητας, ήταν μια σημαντική παράμετρος για ανακρίβειες στον προσδιορισμό της θέσης. Ωστόσο μετά την απενεργοποίησή του, βελτιώθηκε σημαντικά η ακρίβεια των πολιτικών δεκτών GPS.

## 1.3 GSM

Το GSM [21] είναι ένα κυψελοειδές ψηφιακό σύστημα κινητής τηλεφωνίας, το οποίο χρησιμοποιεί ηλεκτρομαγνητικά σήματα, για την παροχή ασύρματων υπηρεσιών. Χρησιμοποιεί την τεχνική πολλαπλής πρόσβασης, με την οποία γίνεται διαχωρισμός του διαθέσιμου φάσματος συχνοτήτων σε έναν αριθμό καναλιών. Αυτά τα κανάλια διαιρούνται σε χρονοθυρίδες, που χρησιμοποιούνται για την μετάδοση σημάτων.

### 1.3.1 Ιστορικό

Η ιδέα της ανάπτυξης κυψελοειδών κινητών τηλεπικοινωνιών δεν είναι νέα. Στα τέλη της δεκαετίας του 70 τα εργαστήρια Bell δημιούργησαν τα πρώτα πειραματικά δίκτυα κινητών επικοινωνιών, χωρίς όμως εμπορικές εφαρμογές. Η εμπορική εκμετάλλευση ξεκίνησε τα πρώτα χρόνια της δεκαετίας του 80, όταν άρχισαν να λειτουργούν τα πρώτα αναλογικά κινητά τηλέφωνα. Η κινητή τηλεφωνία αναπτύχθηκε ταχύτατα, αρχικά στη Σκανδιναβία και στη Μεγάλη Βρετανία, και κατόπιν στην υπόλοιπη Ευρώπη. Σήμερα, οι κινητές τηλεπικοινωνίες βρίσκονται στην αιχμή της τεχνολογίας και αποτελούν χώρο ταχύτατης ανάπτυξης.

Κατά τα πρώτα στάδια της κινητής τηλεφωνίας, κάθε χώρα δημιουργούσε τα δικά της πρότυπα και συστήματα επικοινωνίας, κάτι που δυσχέρανε αρκετά την εξάπλωση των κινητών, αφού ήταν απαραίτητη η ενοποίηση των διεθνών αγορών, προκειμένου η χρήση των κινητών τηλεφώνων να μην περιορίζεται σε συγκεκριμένες γεωγραφικές περιοχές. Με την ύπαρξη ενιαίων προτύπων στην κινητή τηλεφωνία, θα άνοιγε ο δρόμος, τόσο για τη δυνατότητα διεθνών κλήσεων, όσο και για τη μεγαλύτερη εξάπλωση των συσκευών, αφού δεν θα χρειαζόταν οι εταιρείες να κατασκευάζουν τηλέφωνα που να λειτουργούν σε μία μόνο χώρα.

Το φαινόμενο κατακερματισμού των προτύπων και των αγορών έπρεπε να εξαλειφθεί. Έτσι, ιδρύθηκε το 1982 το Groupe Special Mobile (GSM), που ανέλαβε να θεσπίσει πανευρωπαϊκά πρότυπα στην κινητή επικοινωνία. Αρχικά, αποτελούνταν από μια ομάδα ειδικών τηλεπικοινωνιακών μηχανικών που μελετούσαν την ένωση των επιμέρους συστημάτων που προϋπήρχαν. Το 1989 η ευθύνη του GSM ανατέθηκε στο Ευρωπαϊκό Τηλεπικοινωνιακό Ινστιτούτο Προτύπων (ETSI) και το 1990 ανακοινώθηκαν επίσημα για πρώτη φορά το πρότυπο και τα χαρακτηριστικά του GSM. Το 1991 άρχισε η εμπορική του διάθεση στην Ευρώπη, ενώ στην Ελλάδα το σύστημα χρησιμοποιήθηκε το 1993 από την WIND Hellas (πρώην TIM ή πρώην TELESTET).

Το πρότυπο GSM μπορεί να αναπτύχθηκε από ευρωπαϊκό οργανισμό, αλλά τελικά υιοθετήθηκε από πολλές άλλες χώρες των υπόλοιπων ηπείρων, εκμεταλλευόμενο διάφορες ζώνες συχνοτήτων. Σήμερα το GSM είναι το ακρωνύμιο για το το παγκόσμιο σύστημα των κινητών τηλεπικοινωνιών. Οι αρχικοί στόχοι που έθεσε όσον αφορά στην ποιότητα και στις υπηρεσίες του συστήματος είναι:

- καλή ποιότητα ήχου
- χαμηλό κόστος τερματικών συσκευών (τηλεφώνων) και σταθμών βάσης

- διεθνής περιαγωγή
- συμβατότητα με άλλα συστήματα επικοινωνιών
- δυνατότητα επέκτασης και σε νέες υπηρεσίες

Το GSM είναι ψηφιακό σύστημα και κατά συνέπεια πλεονεκτεί στη λειτουργία του, αν το συγκρίνει κανείς με τα προγενέστερα αναλογικά συστήματα. Συγκεκριμένα, η ψηφιακή του λειτουργία επιτρέπει την εξυπηρέτηση μεγαλύτερου αριθμού συνδρομητών, συμβατότητα με άλλα συστήματα, επεκτασιμότητα και καλύτερη ποιότητα στις διάφορες υπηρεσίες που υποστηρίζει.

### 1.3.2 Αρχιτεκτονική

Το GSM δίκτυο αποτελείται από τρία βασικά μέρη. Από τον κινητό σταθμό (Mobile Station), το βασικό υποσύστημα σταθμού (Base Station Subsystem) και το υποσύστημα δικτύου μεταγωγής (Network Switching Subsystem).

Ο κινητός σταθμός είναι μια συσκευή κινητού τηλεφώνου και φέρει οπωσδήποτε πομπό-δέκτη, κεραία, οθόνη και την κάρτα SIM. Η κάρτα SIM, μπορεί να μεταφερθεί εύκολα από κινητό προς κινητό και να χρησιμοποιηθεί σε οποιαδήποτε συσκευή και αν τοποθετηθεί. Κάθε συσκευή με δυνατότητα σύνδεσης στο δίκτυο, διαθέτει έναν προσωπικό χαρακτηριστικό κωδικό που ονομάζεται IMEI (International Mobile Station Equipment Identity). Η κάρτα SIM διαθέτει επίσης έναν κωδικό IMSI (International Mobile Subscriber Identity), ο οποίος περιέχει κωδικό αναγνώρισης και πληροφορίες για τον συνδρομητή. Η κάρτα SIM, μπορεί να κλειδωθεί με την χρήση ενός κωδικού (PIN).

Το βασικό υποσύστημα σταθμού χωρίζεται στο βασικό σταθμό πομπό-δέκτη (BTS) και στο βασικό σταθμό ελέγχου (BSC). Το BTS είναι υπεύθυνο για τον έλεγχο τις επικοινωνίας μεταξύ του δικτύου GSM και του κινητού σταθμού. Όταν ένας χρήστης Α θέλει να πραγματοποιήσει μια κλήση σε έναν άλλο συνδρομητή Β, ο σταθμός βάσης μεταβιβάζει το σήμα με το αίτημά του Α για αναζήτηση και εντοπισμό του άλλου συνδρομητή Β στο τηλεπικοινωνιακό κέντρο της εταιρείας του Α. Το κέντρο της εταιρείας εντοπίζει την κυψέλη στην οποία βρίσκεται ο Β και στέλνει το σήμα στον πλησιέστερο σταθμό βάσης. Από εκεί, πάλι με τη χρήση των διαθέσιμων συχνοτήτων, στέλνεται το σήμα στο κινητό του Β κι έτσι μπορεί να επικοινωνήσει μαζί του ο Α.

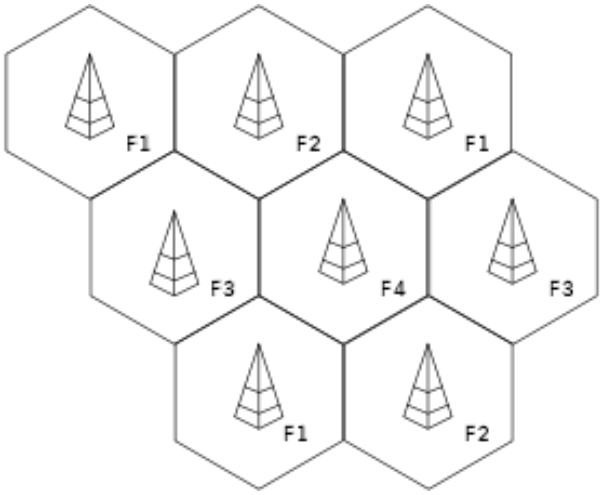
Ο βασικός σταθμός ελέγχου (BSC), ελέγχει τα σήματα από ένα η περισσότερα BTS και τα κατευθύνει ανάλογα. Από ένα BTS πραγματοποιείται και η μετατροπή (αν χρειαστεί) των 16Kbps φωνής που χρησιμοποιούν τα κινητά τηλέφωνα, προς το πρότυπο των 64kbps που χρησιμοποιείται από τα σταθερά.

Το υποσύστημα δικτύου μεταγωγής (NSS) χωρίζεται στο κέντρο διανομής (MSC) και στο κέντρο πιστοποίησης (AC). Το MSC είναι υπεύθυνο για την λειτουργία του κάθε συνδρομητή στο δίκτυο, όπως η καταχώρηση, η πιστοποίηση, η ενημέρωση της θέσης του κ.α. Στο MSC υπάρχουν καταχωρητές όπως ο Visitor Locator Register, ο Home Locator Register και καταχωρητές ασφαλείας, όπως ο Equipment Identify Register ο οποίος ελέγχει αν το IMEI του κινητού σταθμού είναι πιστοποιημένο και μπορεί να λειτουργήσει

σωστά στο δίκτυο. Τέλος, στο κέντρο πιστοποίησης (Authentication Center – AC) γίνεται η διαχείριση δεδομένων, για την πιστοποίηση της ταυτότητας του χρήστη.

### 1.3.3 Η κυψελοειδής δομή του δικτύου

Η αύξηση της εμβέλειας ενός δικτύου GSM σε μια περιοχή, επιτυγχάνεται με διαίρεση της περιοχής αυτής σε μικρότερα τμήματα, που ονομάζονται κυψέλες. Αυτές οι περιοχές, εφάπτονται μεταξύ τους και έχουν από ένα σταθμό βάσης (Base Station). Έτσι γίνεται σύνθεση ενός δικτύου κυψελών, το οποίο μπορεί να επαναληφθεί όσες φορές χρειάζεται, για να πραγματοποιηθεί η απαιτούμενη κάλυψη σε μια περιοχή, κάνοντας επαναχρησιμοποίηση των συχνοτήτων. Με αυτή την μέθοδο αυξάνεται η χωρητικότητα του δικτύου, αλλά πρέπει η ισχύς κάθε κυψέλης να είναι όση χρειάζεται ώστε να μην ξεπερνάει τα όριά της και να επικαλύπτει άλλες κυψέλες της ίδιας δομής.




Σχήμα 1.4: Η επαναχρησιμοποίηση συχνοτήτων σε ένα κυψελοειδές δίκτυο

Η επαναχρησιμοποίηση των συχνοτήτων μπορεί όμως, να δημιουργήσει ενδοκαναλικές παρεμβολές. Πρέπει να σχεδιάζεται, έτσι ώστε οι κυψέλες που λειτουργούν στις ίδιες συχνότητες, να απέχουν αρκετά η μια από την άλλη. Η ενδοκαναλική παρεμβολή μειώνεται όσο αυξάνει ο αριθμός των κυψελών της δομής. Η ακτίνα κάθε κυψέλης σε αραιοκατοικημένες περιοχές είναι έως και 35Km ενώ σε πυκνοκατοικημένες περιοχές δεν ξεπερνά τα 300 μέτρα. Σε περιοχές με πολύ μεγάλη ζήτηση χωρητικότητας δικτύου όπως σε αστικά κέντρα, οι σταθμοί βάσης υπερφορτώνονται και έτσι υπάρχει ανάγκη για μεγαλύτερη χωρητικότητα του δικτύου. Έτσι για να επιτευχθεί αυτός ο σκοπός γίνεται διάσπαση των υπαρχόντων κυψελών σε μικρότερες, ενώ γι' αυτές χρησιμοποιούνται κεραίες μικρότερης ισχύος (macro bs - micro- bs - pico bs) όπως σε κτήρια, στο μετρό, Δημόσιους Οργανισμούς, οδικές αρτηρίες κτλ.

### 1.3.4 Συχνότητες

**GSM 900**

Όταν άρχισε η λειτουργία των πρώτων GSM δικτύων το 1990, χρησιμοποιήθηκε η ζώνη των 900 MHz. Συγκεκριμένα, χρησιμοποιήθηκαν οι συχνότητες 890 - 915 MHz για την βάση και οι συχνότητες 935 - 960 MHz για τον κινητό σταθμό. Αυτή η ζώνη συχνοτήτων, ήταν πολύ μικρή για να εξυπηρετήσει έναν ικανοποιητικό αριθμό χρηστών. Με αυτό το σκεπτικό, χρησιμοποιήθηκε ένας συνδυασμός διαίρεσης χρόνου και συχνοτήτων - πολλαπλής πρόσβασης. (Time and Frequency Division - Multiple Access TDMA/FDMA). Οι ζώνες των 25 MHz υποδιαιρούνται η καθεμία σε 124 (+ 1 ελεύθερο) κανάλια συχνότητας και κάθε κανάλι έχει εύρος ζώνης 200 KHz. Όλο αυτό το σύστημα ονομάστηκε GSM 900 ή Standard GSM.

**GSM 1800**

Η γρήγορη εξάπλωση των κινητών επικοινωνιών δημιούργησε την ανάγκη για χρήση του GSM σε διαφορετικές συχνότητες. Αυτό έγινε κυρίως, γιατί σε κάποιες περιπτώσεις οι ζώνες του GSM 900 στην Ευρώπη ήταν πιασμένες από άλλους παροχείς κινητής τηλεφωνίας. Ουσιαστικά πρόκειται για την μεταφορά του GSM σε διαφορετικές ζώνες συχνοτήτων αφού οι προδιαγραφές του είναι ίδιες. Οι σημαντικότερες διαφοροποιήσεις αφορούν τη συχνότητα λειτουργίας που είναι η περιοχή των 1.800 MHz και τη στάθμη εκπομπής που είναι αρκετά χαμηλότερη. Οι ζώνες που χρησιμοποιούνται είναι 1710 - 1785 MHz για την βάση και 1805 - 1880 MHz για τον κινητό σταθμό. Οι περιοχές των 75 MHz που προκύπτουν, υποδιαιρούνται η καθεμία σε 374 (+ 1 ελεύθερο) κανάλια και κάθε κανάλι έχει εύρος ζώνης 200 KHz. Το σύστημα αρχικά ονομάστηκε DCS 1800 και αργότερα μετονομάστηκε σε GSM 1800, για λόγους προώθησης του GSM ως παγκόσμιο σύστημα.

**GSM 1900**

Στην Αμερική αρχικά χρησιμοποιήθηκε η ζώνη συχνοτήτων των 1900 MHz. Διατηρήθηκε η δομή του GSM όπως και στο GSM 1800, όμως χρησιμοποιήθηκαν και πάλι διαφορετικά ζεύγη συχνοτήτων. Συγκεκριμένα για τον σταθμό βάσης 1850 - 1910 MHz και 1930 - 1990 MHz για τον κινητό σταθμό. Κάθε μια από αυτές τις ζώνες υποδιαιρείται σε 299 (+ 1 ελεύθερο) κανάλια συχνότητας όπου κάθε κανάλι έχει εύρος ζώνης 200 KHz. Το σύστημα αυτό αρχικά ονομάστηκε PCS 1900, όμως στην συνέχεια μετονομάστηκε σε GSM 1900, για τους ίδιους λόγους με το GSM 1800.

**E-GSM**

Στα τέλη της δεκαετίας του 1990 αποφασίστηκε η αντικατάσταση του GSM 900 με το E-GSM. Το νέο σύστημα διατήρησε την ήδη υπάρχουσα δομή με ταυτόχρονη αύξηση των ζωνών συχνοτήτων. Για τον σταθμό βάσης ορίστηκαν οι συχνότητες 880 - 915 MHz και για τον κινητό σταθμό οι συχνότητες 925 - 960 MHz.

### 1.3.5 Πιστοποίηση και ασφάλεια

Οι ραδιοσυχνότητες είναι διαθέσιμες σε όλους, οπότε η πιστοποίηση των χρηστών κρίνεται αναγκαία υπόθεση. Η πιστοποίηση ενός χρήστη γίνεται χρησιμοποιώντας την κάρτα SIM που βρίσκεται στο κινητό και το κέντρο πιστοποίησης (AC). Κάθε συνδρομητής χαρακτηρίζεται από ένα κρυμμένο κλειδί, το οποίο βρίσκεται αποθηκευμένο στην κάρτα SIM και στο κέντρο πιστοποίησης. Για να πραγματοποιηθεί η πιστοποίηση, το κέντρο πιστοποίησης δημιουργεί έναν τυχαίο αριθμό και τον στέλνει στο χρήστη. Ο αριθμός αυτός χρησιμοποιείται τόσο από το κινητό αλλά και το κέντρο πιστοποίησης, σε συνδυασμό με το κρυφό κλειδί και έναν αλγόριθμο κρυπτογράφησης που καλείται A3. Το αποτέλεσμα είναι η δημιουργία ενός αριθμού, ο οποίος στέλνεται πίσω στο κέντρο πιστοποίησης. Αν ο αριθμός που υπολογίστηκε από το κινητό, είναι ο ίδιος με αυτόν που υπολογίστηκε από το κέντρο, ο συνδρομητής έχει πιστοποιηθεί.

Για περισσότερη ασφάλεια, ο αριθμός που προέκυψε από την παραπάνω διαδικασία, χρησιμοποιείται μαζί με τον αριθμό πλαισίου TDMA σε έναν αλγόριθμο με το όνομα A5. Αυτό γίνεται για την κωδικοποίηση των δεδομένων που στέλνονται στην ραδιοζεύξη, ώστε να γίνει αδύνατη η παρακολούθησή τους από τρίτους.

Κατά την προτυποποίηση του GSM, έγινε κατανοητό ότι πρέπει να προστεθεί και ασφάλεια σε επίπεδο συσκευής. Γι' αυτό τον λόγο, κάθε τερματικό του GSM έχει την δική του ταυτότητα (IMEI). Το δίκτυο διατηρεί μια λίστα με όλους τους αριθμούς IMEI των τερματικών του και αποδίδει στο καθένα μια από τις τρεις παρακάτω καταστάσεις.

- λευκή λίστα - φυσιολογική λειτουργία
- γκρι λίστα - λίστα συσκευών υπό παρακολούθηση λόγω προβλημάτων
- μαύρη λίστα - λίστα συσκευών που έχουν δηλωθεί ως κλεμμένα από τους κατόχους

### 1.3.6 GPRS

Το GPRS [10] ή αλλιώς General Packet Radio Service, είναι μια υπηρεσία προστιθέμενης αξίας που εστιάζει στην ανταλλαγή δεδομένων μέσω του δικτύου κινητής τηλεφωνίας GSM. Το GPRS επιτρέπει τη χρήση του κινητού για τη μεταφορά δεδομένων, συνήθως από το διαδίκτυο, γρήγορα και εύκολα, ενώ παράλληλα παρέχει το πλεονέκτημα της αδιάκοπης σύνδεσης με αυτό.

Πριν από την ένταξη του GPRS στις προδιαγραφές του GSM (1997), η μετάδοση των δεδομένων πραγματοποιούνταν με την αποκλειστική χρήση κυκλωμάτων CSD (Circuit Switched Data), ωστόσο η ταχύτητα περιορίζονταν στα 9,6kbits/s. Επιπρόσθετα, το κύκλωμα δεσμεύονταν καθ' όλη τη διάρκεια της χρήσης, ανεξάρτητα από το αν πραγματοποιούνταν μεταφορά δεδομένων, με αποτέλεσμα την σπατάλη των διαθέσιμων πόρων του δικτύου.

Αντίθετα, στο GPRS επιτρέπεται η ταυτόχρονη χρήση των ίδιων κυκλωμάτων από πολλούς χρήστες, αφού αυτά αξιοποιούνται, μόνο όταν πραγματοποιείται μεταφορά δεδομένων.

### Λειτουργία

Η λειτουργία του GPRS παρουσιάζει αρκετές ομοιότητες με τον τρόπο λειτουργίας του διαδικτύου. Η πληροφορία, σε κάθε περίπτωση, διαιρείται σε πακέτα δεδομένων, τα οποία μεταδίδονται στον προορισμό τους και στη συνέχεια συνδυάζονται για να δημιουργήσουν ένα ακριβές αντίγραφο της αρχικής πληροφορίας. Αναλόγως λειτουργεί και το πρωτόκολλο IP (Internet Protocol) που χρησιμοποιείται στο διαδίκτυο. Το GPRS εκμεταλλεύεται σε μέγιστο βαθμό τους διαθέσιμους πόρους του δικτύου GSM για να επιτραπεί ο κατακερματισμός των πληροφοριών και η ασύρματη μεταφορά τους.

Οι συχνότητες λειτουργίας του GSM περιέχουν κανάλια πλάτους 200KHz, το καθένα από τα οποία χωρίζεται σε 8 χρονοθυρίδες. Για παράδειγμα, για τη πραγματοποίηση μιας φωνητικής κλήσης, δεσμεύεται μια από αυτές τις χρονοθυρίδες, η οποία απελευθερώνεται μετά τον τερματισμό της κλήσης. Η κάθε χρονοθυρίδα επιτρέπει και τη μετάδοση πληροφοριών στη ταχύτητα των 9,6kbps. Στα δίκτυα GPRS ωστόσο, επιτρέπεται η ταυτόχρονη χρήση πολλών χρονοθυρίδων, ώστε να επιτυγχάνεται η ταχύτερη μετάδοση των πληροφοριών. Παράλληλα, οι χρονοθυρίδες δεσμεύονται μόνο όταν απαιτείται η αποστολή ή λήψη πακέτων δεδομένων και αποδεσμεύονται μετά το τέλος της μετάδοσης. Αυτό έχει ως αποτέλεσμα μια πολύ πιο αποδοτική χρήση των διαθέσιμων πόρων.

### Ταχύτητα σύνδεσης

Υποστηρίζονται 4 διαφορετικά σχήματα κωδικοποίησης για την μεταφορά δεδομένων (CS-1, CS-2, CS-3, CS-4). Κάθε ένα από αυτά προσφέρει διαφορετική ποιότητα και διαφορετικό ρυθμό μετάδοσης ανά χρονοθυρίδα. Με τη χρήση του πρώτου σχήματος κωδικοποίησης CS-1 ο ρυθμός μετάδοσης είναι 9,05 kbits/s, με τη χρήση του CS-2 είναι 13,4 kbits/s, με τη χρήση του CS-3 είναι 15,6 kbits/s και με τη χρήση του CS-4 είναι 21,4 kbits/s. Εφόσον χρησιμοποιηθούν ταυτόχρονα 8 χρονοθυρίδες και το τέταρτο σχήμα κωδικοποίησης το συνολικό data rate μπορεί να φθάσει τα 171,2 kbits ανά δευτερόλεπτο.

Στην πράξη όμως, τα περισσότερα δίκτυα χρησιμοποιούν κάθε σχήμα για διαφορετικό σκοπό. Το CS-1 χρησιμοποιείται για σηματοδοσία και το CS-2 για τη μεταφορά πληροφοριών, ενώ τα κινητά τηλέφωνα συνήθως επιτρέπουν τη δέσμευση έως και 4 χρονοθυρίδων για τη λήψη. Δηλαδή το πραγματικό data rate είναι 4 x 13,4 = 53,6 kbits/s ή περίπου 6,7Kb ανά δευτερόλεπτο.

# Κεφάλαιο 2

# Ανάλυση και σχεδίαση

Το προηγούμενο κεφάλαιο αποτέλεσε την εισαγωγή, ώστε να μπορεί να γίνει κατανοητό από τον αναγνώστη, το σύστημα που αναπτύσσεται. Σε αυτό το κεφάλαιο, γίνεται η ανάλυση του συστήματος που έγινε πριν το στάδιο της υλοποίησης. Γίνεται γνωστή η αρχιτεκτονική του συστήματος που θα υλοποιηθεί. Επίσης, καταγράφονται οι ρόλοι οι απαιτήσεις και οι λειτουργίες του συστήματος. Σαν συνέπεια όλων αυτών σχεδιάζεται και η βάση δεδομένων.

## 2.1 Αρχιτεκτονική

Η αρχιτεκτονική του συστήματος βασίζεται στην λογική του client-server με μονόπλευρη επικοινωνία. Δηλαδή, ένα όχημα μπορεί να μεταδώσει πληροφορία στην εφαρμογή, αλλά όχι το αντίθετο. Η μόνη επικοινωνία που συμβαίνει προς την αντίθετη κατεύθυνση, είναι η επιβεβαίωση της σύνδεσης.

Το σύστημα υποστηρίζει πολλαπλούς χρήστες και κάθε χρήστης μπορεί να προσθέτει οχήματα. Η βασική ιδέα είναι ότι ένα όχημα φέρει μια συσκευή, η οποία μπορεί να επικοινωνήσει με την εφαρμογή. Η συσκευή έχει την δυνατότητα εξάγει πληροφορίες σχετικές με την κίνηση του οχήματος (Θέση, υψόμετρο, ταχύτητα) και να τις στείλει στην εφαρμογή. Τα οχήματα αναγνωρίζονται μοναδικά από το IMEI τους.

Η επικοινωνία πραγματοποιείται χρησιμοποιώντας την υπηρεσία GPRS του δικτύου GSM. Ουσιαστικά, πρόκειται για μια αίτηση GET μέσω ασφαλούς σύνδεσης SSL με τις παρακάτω παραμέτρους.

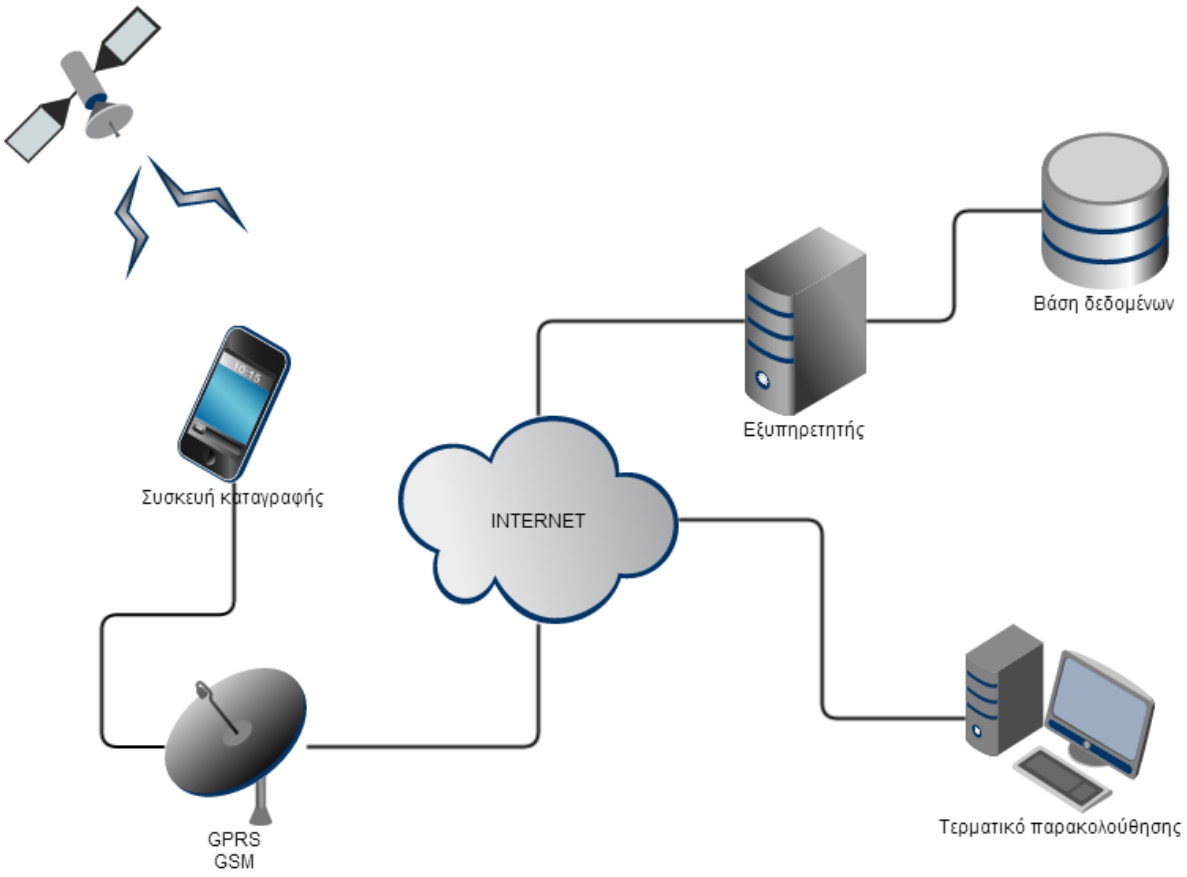
- key: το μοναδικό κλειδί της συσκευής (IMEI)
- lat: γεωγραφικό πλάτος
- lng: γεωγραφικό μήκος
- spd: ταχύτητα σε χλμ/ώρα
- alt: υψόμετρο σε μέτρα

- tim: χρόνος καταγραφής των παραπάνω δεδομένων σε millisecond
- alm: ειδοποίηση (προαιρετική)

Η αίτηση GET έχει την εξής μορφή:

**http://domain.com/upload?key=key&lat=40&lng=22&spd=120&alt=100&tim=1365791030317**

Θα μπορούσε να χρησιμοποιηθεί και POST, αλλά δεν θα προσέφερε κάποια επιπλέον ασφάλεια, μιας και το SSL λειτουργεί με τον ίδιο τρόπο και στις δυο περιπτώσεις. Η επιλογή του GET έγινε περισσότερο για λόγους απλότητας υλοποίησης.




Σχήμα 2.1: Σχηματικά η αρχιτεκτονική

Στο Σχήμα 2.1 φαίνεται σχηματικά η λειτουργία του συστήματος. Οι συσκευές λαμβάνουν σήματα GPS και μεταδίδουν δεδομένα μέσω του internet. Τα δεδομένα αυτά αποθηκεύονται σε μια βάση δεδομένων και είναι διαθέσιμα μέσω του ιστοχώρου.

## 2.2 Ρόλοι του συστήματος

Η βασική αρχή για να γίνει σωστός σχεδιασμός, είναι να βρεθούν οι χρήστες του συστήματος. Δηλαδή να βρεθούν οι οντότητες οι οποίες θα αλληλεπιδρούν με αυτό. Έτσι εντοπίστηκαν τέσσερις:

- αυτή που αναπαριστά ένα όχημα
- αυτή του εγγεγραμμένου χρήστη
- αυτή του ανώνυμου χρήστη
- και αυτή του διαχειριστή

Ο ρόλος της καθεμιάς από τις παραπάνω οντότητες αναλύεται στις επόμενες παραγράφους και οι λεπτομέρειες σχετικά με αυτές, θα συμπληρώνονται κατά την ανάπτυξη του συστήματος.

Ο ρόλος ενός οχήματος ως οντότητα που αλληλεπιδρά με το σύστημα είναι περιορισμένος. Επικεντρώνεται στην ενημέρωση των οντοτήτων, που έχουν να κάνουν με την αποθήκευση πληροφοριών, σχετικά με την θέση ενός οχήματος μια δεδομένη χρονική στιγμή. Η ενημέρωση αυτή απαιτείται να γίνεται περιοδικά ούτως ώστε το σύστημα να είναι πάντα ενήμερο για την θέση ενός οχήματος. Έτσι, προσφέρεται όσο το δυνατόν καλύτερη πληροφορία σε μια άλλη οντότητα, τον χρήστη.

Ο κύριος ρόλος του συστήματος είναι αυτός του χρήστη. Ουσιαστικά, είναι αυτός ο οποίος θα μπορεί μέσω ενός γραφικού περιβάλλοντος, να επιδρά στο σύστημα και να αντλεί πληροφορίες από αυτό.

Ο ανώνυμος χρήστης είναι η περίπτωση του χρήστη που δεν έχει εγγραφεί στο σύστημα. Αντίθετα, η περίπτωση του διαχειριστή είναι αυτή του χρήστη, ο οποίος κατέχει επιπρόσθετες λειτουργίες και δικαιώματα.

## 2.3 Απαιτήσεις

Αφού πλέον έχουν καθοριστεί οι ρόλοι του συστήματος, μπορούν να εκφραστούν και οι απαιτήσεις που έχουν αυτοί από την εφαρμογή.

Έτσι ένας ανώνυμος χρήστης μπορεί:

- να δημιουργήσει έναν νέο λογαριασμό, καταχωρώντας τα στοιχεία του στο σύστημα
- να συνδεθεί στο σύστημα

Ένας εγγεγραμμένος χρήστης μπορεί:

- να διαχειρίζεται οχήματα (προσθήκη/επεξεργασία/διαγραφή)
- να διαχειρίζεται στόλους οχημάτων

- να παρακολουθεί τα οχήματα και στόλους του σε πραγματικό χρόνο
- να μπορεί να εμφανίζει διαδρομές που έχουν πραγματοποιηθεί
- να ορίζει όρια ταχύτητας, περιοχής κίνησης, συντήρησης
- να προσθέτει δεδομένα σχετικά με τον ανεφοδιασμό των οχημάτων
- να λαμβάνει και να προβάλει ειδοποιήσεις
- να λαμβάνει περιοδικές αναφορές για τα οχήματά του
- να βλέπει στατιστικά των οχημάτων/στόλων
- να τροποποιεί τον λογαριασμό του

Ένας διαχειριστής μπορεί:

- να διαχειρίζεται όλα τα οχήματα
- να διαχειρίζεται τους στόλους όλων των χρηστών
- να διαχειρίζεται όλα τα δεδομένα ανεφοδιασμών
- να διαχειρίζεται όλους τους χρήστες
- να βλέπει ανά πάσα στιγμή την κατάσταση του συστήματος

Ένα όχημα μπορεί:

- να καταγράφει πληροφορίες σχετικές με την θέση και την κίνηση του
- να στέλνει αυτές τις πληροφορίες στο σύστημα προς αποθήκευση

## 2.4 Περιπτώσεις χρήσης

Οι περιπτώσεις χρήσης [1] του συστήματος είναι αρκετές και η ανάλυση όλων σε αυτό το κείμενο θα ήταν υπερβολική. Έτσι, για την πληρότητα του κειμένου, αναλύονται παρακάτω μερικές από αυτές.

### 2.4.1 Δημιουργία λογαριασμού

#### Σύντομη περιγραφή

Η περίπτωση χρήσης "Δημιουργία λογαριασμού" επιτρέπει σε έναν χειριστή να δημιουργήσει έναν νέο λογαριασμό και να καταχωρίσει τις απαραίτητες πληροφορίες και προτιμήσεις για αυτόν.

**Χειριστές**

Ανώνυμος χρήστης

**Βασική ροή**

Αυτή η περίπτωση χρήσης ξεκινά, μόλις ο χρήστης επιλέξει τη λειτουργία "Δημιουργία νέου λογαριασμού"

- Το σύστημα εμφανίζει μια φόρμα με τις απαιτούμενες πληροφορίες προς εισαγωγή.
- Ο χρήστης εισάγει τις απαραίτητες πληροφορίες και ζητά από το σύστημα να τις επικυρώσει.
- Το σύστημα επικυρώνει τις πληροφορίες και ξεκινά την περίπτωση χρήσης "Είσοδος στο σύστημα".

**Εναλλακτική ροή 1 - Ακύρωση δημιουργίας λογαριασμού**

Σε οποιοδήποτε στάδιο δημιουργίας του λογαριασμού, ο χρήστης μπορεί να ακυρώσει την δημιουργία. Το αποτέλεσμα είναι ότι δεν δημιουργείται λογαριασμός.

**Εναλλακτική ροή 2 - Ο χρήστης εισάγει λανθασμένη πληροφορία**

Το σύστημα αναγνώρισε ότι ο χρήστης έχει εισαγάγει λανθασμένη πληροφορία.

- Το σύστημα παρουσιάζει σχετικό μήνυμα λάθους για κάθε λανθασμένο πεδίο.
- Το σύστημα επιτρέπει την επανεισαγωγή του πεδίου.
- Ο χρήστης εισάγει την πληροφορία.
- Το σύστημα ελέγχει αν τα στοιχεία που δόθηκαν είναι σωστά. Αν ναι τότε το σύστημα συνεχίζει με το βήμα 3 της βασικής ροής. Αν όχι, τότε το σύστημα επανέρχεται στην εναλλακτική ροή 2.

**Κατάσταση εισόδου**

Δεν υπάρχει.

**Κατάσταση εξόδου**

- Ο λογαριασμός έχει δημιουργηθεί.
- Ο λογαριασμός δεν δημιουργήθηκε λόγω εσφαλμένων δεδομένων, ή γιατί ο χρήστης ακύρωσε την ενέργεια.

### 2.4.2 Είσοδος στο σύστημα

**Σύντομη περιγραφή**

Η περίπτωση χρήσης "Είσοδος στο σύστημα", επιτρέπει σε έναν χειριστή να εισέλθει στο σύστημα χρησιμοποιώντας τα στοιχεία του λογαριασμού του.

**Χειριστές**

Ανώνυμος χρήστης

**Βασική ροή**

Αυτή η περίπτωση χρήσης ξεκινά, μόλις ο χρήστης επιλέξει τη λειτουργία "Είσοδος στο σύστημα"

- Το σύστημα εμφανίζει μια φόρμα με δύο πεδία. Το email και το συνθηματικό του χρήστη.
- Ο χρήστης εισάγει τις απαραίτητες πληροφορίες και ζητά από το σύστημα να τις επικυρώσει.
- Το σύστημα επικυρώνει τις πληροφορίες και μεταφέρει τον χρήστη στο προσωπικό του προφίλ.

**Εναλλακτική ροή 1 - Ακύρωση εισόδου στο σύστημα**

Σε οποιοδήποτε στάδιο διαδικασίας εισόδου, ο χρήστης μπορεί να διακόψει. Το αποτέλεσμα είναι ότι ο χρήστης παραμένει ανώνυμος και δεν εισάγεται στο σύστημα.

**Εναλλακτική ροή 2 - Ο χρήστης εισάγει λανθασμένη πληροφορία**

Το σύστημα αναγνώρισε ότι ο χρήστης έχει εισαγάγει λανθασμένη πληροφορία.

- Το σύστημα παρουσιάζει εμφανίζει μήνυμα λανθασμένων στοιχείων.
- Το σύστημα επιτρέπει την επανεισαγωγή των πεδίων.
- Ο χρήστης εισάγει την πληροφορία.
- Το σύστημα ελέγχει αν τα στοιχεία που δόθηκαν είναι σωστά. Αν ναι, τότε το σύστημα συνεχίζει με το βήμα 3 της βασικής ροής. Αν όχι, τότε το σύστημα επανέρχεται στην εναλλακτική ροή 2.

**Κατάσταση εισόδου**

Να έχει πραγματοποιηθεί εγγραφή μέσω της περίπτωσης χρήσης "Δημιουργία λογαριασμού".

**Κατάσταση εξόδου**

- Ο χρήστης έχει εισέλθει στο σύστημα.
- Η είσοδος δεν πραγματοποιήθηκε λόγω εσφαλμένων δεδομένων ή γιατί ο χρήστης ακύρωσε την ενέργεια.

### 2.4.3 Παρακολούθηση οχημάτων

**Σύντομη περιγραφή**

Η περίπτωση χρήσης "Παρακολούθηση οχημάτων" επιτρέπει σε ένα χειριστή να παρακολουθήσει τα οχήματα πάνω σε ένα χάρτη σε πραγματικό χρόνο.

**Χειριστές**

Εγγεγραμμένος χρήστης

**Βασική ροή**

Αυτή η περίπτωση χρήσης ξεκινά, μόλις ο χρήστης επιλέξει τη λειτουργία "Παρακολούθηση οχημάτων"

- Το σύστημα εμφανίζει έναν χάρτη και μια φόρμα επιλογής οχημάτων.
- Ο χρήστης επιλέγει το όχημα που θέλει να παρακολουθήσει.
- Το σύστημα αναζητά την θέση του οχήματος και το εμφανίζει στον χάρτη χρησιμοποιώντας κάποιο σύμβολο.

**Εναλλακτική ροή 1 - Δεν υπάρχει διαθέσιμη θέση**

Η εφαρμογή δεν βρήκε διαθέσιμη θέση για το επιλεγμένο όχημα. Εμφανίζει μήνυμα στον χρήστη και επιστρέφει στο βήμα 1 της βασικής ροής.

**Κατάσταση εισόδου**

Δεν υπάρχει.

**Κατάσταση εξόδου**

- Το όχημα που επιλέχθηκε εμφανίζεται στον χάρτη.
- Δεν υπάρχει κάποιο εμφανιζόμενο όχημα γιατί δεν βρέθηκε θέση στην βάση δεδομένων.

### 2.4.4 Δημιουργία γεωφράκτη

#### Σύντομη περιγραφή

Η περίπτωση χρήσης "Δημιουργία γεωφράκτη" επιτρέπει σε ένα χειριστή να περιορίσει τα οχήματα του σε μια περιοχή.

#### Χειριστές

Εγγεγραμμένος χρήστης

#### Βασική ροή

Αυτή η περίπτωση χρήσης ξεκινά, μόλις ο χρήστης επιλέξει τη λειτουργία "Δημιουργία γεωφράκτη"

- Το σύστημα εμφανίζει έναν χάρτη και μια φόρμα επιλογής οχημάτων.
- Ο χρήστης επιλέγει το όχημα που θέλει να περιορίσει.
- Το σύστημα εμφανίζει μια εργαλειοθήκη για την δημιουργία του γεωφράκτη.
- Ο χρήστης επιλέγει μια περιοχή και επιλέγει αποθήκευση.
- Το σύστημα αποθηκεύει τον περιορισμό και εμφανίζει ενημερωτικό μήνυμα.

#### Εναλλακτική ροή 1 - Υπάρχει ήδη περιορισμός

Το σύστημα οπτικοποιεί τον περιορισμό και επιστρέφει στο βήμα 3 της βασικής ροής.

#### Εναλλακτική ροή 2 - Ακύρωση γεωφράκτη

Σε οποιοδήποτε σημείο της διαδικασίας, ο χρήστης μπορεί να επιλέξει την ακύρωση των ενεργειών που έκανε για περιορισμό. Το αποτέλεσμα είναι ότι δεν αποθηκεύεται ο φράκτης.

#### Κατάσταση εισόδου

Πρέπει να υπάρχουν διαθέσιμα οχήματα για την εισαγωγή των περιορισμών.

#### Κατάσταση εξόδου

- Το όχημα που επιλέχθηκε, έχει περιοριστεί στην επιλεγμένη περιοχή.
- Δεν έχει γίνει κάποια αλλαγή στα οχήματα, γιατί ο χρήστης ακύρωσε την ενέργεια.

### 2.4.5 Προσθήκη οχήματος

#### Σύντομη περιγραφή

Η περίπτωση χρήσης "Προσθήκη οχήματος" επιτρέπει σε ένα χειριστή να προσθέσει οχήματα στην εφαρμογή.

#### Χειριστές

Εγγεγραμμένος χρήστης

#### Βασική ροή

Αυτή η περίπτωση χρήσης ξεκινά, μόλις ο χρήστης επιλέξει τη λειτουργία "Προσθήκη οχήματος".

- Το σύστημα εμφανίζει μια φόρμα με τις απαιτούμενες πληροφορίες προς εισαγωγή.
- Ο χρήστης εισάγει τις απαραίτητες πληροφορίες και ζητά από το σύστημα να τις επικυρώσει.
- Το σύστημα επικυρώνει τις πληροφορίες και εμφανίζει ενημερωτικό μήνυμα επιτυχίας.

#### Εναλλακτική ροή 1 - Ο χρήστης εισάγει λανθασμένη πληροφορία

Το σύστημα αναγνώρισε ότι ο χρήστης έχει εισαγάγει λανθασμένη πληροφορία.

- Το σύστημα παρουσιάζει σχετικό μήνυμα λάθους για κάθε λανθασμένο πεδίο.
- Το σύστημα επιτρέπει την επανεισαγωγή του πεδίου.
- Ο χρήστης εισάγει την πληροφορία.
- Το σύστημα ελέγχει αν τα στοιχεία που δόθηκαν είναι σωστά. Αν ναι, τότε το σύστημα συνεχίζει με το βήμα 3 της βασικής ροής. Αν όχι, τότε το σύστημα επανέρχεται στην εναλλακτική ροή 1.

#### Εναλλακτική ροή 2 - Ακύρωση προσθήκης οχήματος

Σε οποιοδήποτε σημείο της διαδικασίας, ο χρήστης μπορεί να επιλέξει την ακύρωση της προσθήκης οχήματος. Το αποτέλεσμα είναι ότι δεν προστίθεται το όχημα.

#### Κατάσταση εισόδου

Δεν υπάρχει.

**Κατάσταση εξόδου**

- Το όχημα έχει προστεθεί.
- Δεν έχει προστεθεί κάποιο όχημα γιατί ο χρήστης ακύρωσε την διαδικασία.

## 2.5 Βάση δεδομένων

Ο σχεδιασμός της βάσης δεδομένων είναι πολύ σημαντικός για την σωστή και χωρίς προβλήματα ανάπτυξη του συστήματος. Πρέπει να γίνει, χρησιμοποιώντας όλες τις πληροφορίες που συλλέχθηκαν στα προηγούμενα στάδια με τον βέλτιστο δυνατό τρόπο. Δηλαδή, να μην υπάρχουν πεδία που δεν χρησιμοποιούνται, πεδία που δεσμεύουν περισσότερο χώρο απ' ότι χρειάζεται και πεδία που επαναλαμβάνονται.

Αρχικά, έγινε μια προσπάθεια, ώστε να αποτυπωθούν σε ένα σχήμα οι ανάγκες σε δεδομένα. Δηλαδή, μια συνοπτική περιγραφή της λειτουργίας του συστήματος με προτεραιότητα στα δεδομένα. Έτσι προέκυψε το σχήμα 2.2. Σε αυτό παρουσιάζονται έξι οντότητες (χρήστης, όχημα, στόλος, καύσιμα, θέση, ειδοποίηση). Ένας χρήστης δηλαδή, μπορεί να διαθέτει οχήματα και στόλους οχημάτων. Οι στόλοι οχημάτων δημιουργούνται από οχήματα που ανήκουν σε μια συγκεκριμένη κατηγορία. Τέλος, τα οχήματα μπορούν να βρίσκονται σε μια θέση, να ανεφοδιάζονται και η λειτουργία τους να δημιουργεί ειδοποιήσεις.

Βέβαια, αυτό το σχήμα εμπλουτιζόταν συνεχώς κατά την διάρκεια της ανάπτυξης της εφαρμογής, ανάλογα με τις ανάγκες. Έτσι, το τελικό σχήμα της βάσης απαρτίζεται από τους παρακάτω πίνακες.

- Vehicle
- Fleet
- User
- Notification
- Position
- Fuel

### 2.5.1 Vehicle

Ο πίνακας Vehicle χρησιμοποιείται για την αποθήκευση των οχημάτων και των χαρακτηριστικών τους στην βάση δεδομένων.

**id**

Είναι το πρωτεύον κλειδί του πίνακα. Για κάθε εγγραφή που εισάγεται, δημιουργείται και ένα καινούριο, που είναι η τιμή του κλειδιού της προηγούμενης εγγραφής, αυξημένη κατά ένα. Έτσι χρησιμοποιήθηκε ο τύπος δεδομένων integer.

| Στήλη | Τύπος | Κενό | Προεπιλογή |
|---|---|---|---|
| id | int(11) | όχι | καμία |
| name | varchar(120) | ναι | καμία |
| user_id | int(11) | όχι | καμία |
| key | varchar(15) | όχι | καμία |
| fleet_id | int(11) | ναι | καμία |
| odometer | float | - | 0 |
| service | float | - | 0 |
| speed | float | ναι | καμία |
| stasis | int(11) | - | 0 |
| fence | text | ναι | καμία |
| stasis_not_int | tinyint(1) | - | 1 |
| fence_not_int | tinyint(1) | - | 1 |
| speed_not_int | tinyint(1) | - | 1 |
| service_not_int | tinyint(1) | - | 1 |
| email_not | tinyint(1) | - | 1 |
| report | tinyint(1) | - | 1 |
| celery_stasis_id | varchar(40) | ναι | καμία |

Πίνακας 2.1: Πίνακας Vehicle

**name**

Αποθηκεύεται το όνομα του οχήματος. Επιλέχθηκε ο τύπος varchar(120) γιατί το πεδίο είναι κατά κανόνα αλφαριθμητικό, ενώ το μήκος 120 χαρακτήρων είναι αρκετό, ώστε να καλύψει κάθε ανάγκη ονοματοδοσίας.

**user_id**

Είναι ξένο κλειδί του πίνακα User. Πρόκειται για το αντίστοιχο πρωτεύον κλειδί που αντιστοιχεί στον κάτοχο του οχήματος, οπότε χρησιμοποιείται ο ίδιος τύπος δεδομένων.

**key**

Για την επικοινωνία των οχημάτων με το σύστημα χρησιμοποιείται ένα κλειδί. Αυτό το κλειδί αποφασίστηκε ότι πρέπει να είναι το IMEI της συσκευής. Το μήκος του IMEI είναι 15 χαρακτήρες και γι' αυτό χρησιμοποιείται ο τύπος δεδομένων varchar(15).

**fleet_id**

Κάθε όχημα μπορεί να ανήκει σε ένα στόλο οχημάτων ή όχι. Ο στόλος στον οποίο ανήκει, αποθηκεύεται σε αυτό το πεδίο, χρησιμοποιώντας σύνδεση ξένου κλειδιού με τον πίνακα fleet. Έτσι ο τύπος του πεδίου ορίζεται όπως αυτόν του αντίστοιχου του πίνακα Fleet.

**odometer**

Σε αυτό το πεδίο αποθηκεύεται η τιμή του οδομέτρου κάθε οχήματος σε χιλιόμετρα. Με κάθε κίνηση του οχήματος, αλλάζει και η τιμή του πεδίου, ανάλογα με την απόσταση που διανύθηκε. Η απόσταση που προστίθεται κάθε φορά μπορεί να είναι μερικά μέτρα, πράγμα που σημαίνει δεκαδικό αριθμό. Έτσι επιλέχθηκε τύπος δεδομένων float.

**service**

Καταγράφεται μια τιμή στην οποία θα πρέπει να γίνουν εργασίες συντήρησης του οχήματος. Αποθηκεύεται αριθμός, οπότε επιλέχθηκε τύπος float.

**speed**

Αποθηκεύει την τιμή της ταχύτητας στην οποία περιορίζεται το όχημα. Αποθηκεύεται αριθμός, οπότε επιλέχθηκε τύπος float.

**stasis**

Περιέχει το χρονικό διάστημα σε δευτερόλεπτα, μετά το πέρας του οποίου, θα πρέπει να ειδοποιηθεί ο χρήστης για ακινησία. Έτσι επιλέχθηκε ο τύπος δεδομένων integer.

**fence**

Σε αυτό το πεδίο αποθηκεύονται όλες οι συντεταγμένες, οι οποίες περιγράφουν έναν γεωφράκτη. Ο τύπος που χρησιμοποιήθηκε είναι text, γιατί επιτρέπει την επέκταση των σημείων που απαρτίζουν τον φράκτη. Θα ήταν προτιμότερη η χρησιμοποίηση τύπου polygon, όμως δεν υποστηρίζεται από όλες τις βάσεις δεδομένων.

**stasis_not_int**

Αποθηκεύεται η τρέχουσα κατάσταση των ειδοποιήσεων στάσης. Χρησιμοποιείται για την αποφυγή επαναλαμβανόμενων ειδοποιήσεων. Ο τύπος των δεδομένων είναι boolean αλλά στην πράξη η MySQL αποθηκεύει μια εγγραφή boolean σε πεδίο tinyint. Δηλαδή, είναι συνώνυμοι τύποι δεδομένων. Έτσι, κάθε εγγραφή που κανονικά θα έπρεπε να οριστεί σαν boolean γιατί έχει δύο καταστάσεις, ορίζεται σαν tinyint(1).

**fence_not_int**

Αποθηκεύεται η τρέχουσα κατάσταση των ειδοποιήσεων παραβίασης του γεωφράκτη. Χρησιμοποιείται για την αποφυγή επαναλαμβανόμενων ειδοποιήσεων. Το πεδίο υποστηρίζει δύο καταστάσεις οπότε επιλέχθηκε ο τύπος tinyint(1).

| Στήλη | Τύπος | Κενό | Προεπιλογή |
|---|---|---|---|
| id | int(11) | όχι | καμία |
| name | varchar(120) | όχι | καμία |
| user_id | int(11) | όχι | καμία |
| description | varchar(120) | ναι | καμία |

Πίνακας 2.2: Πίνακας Fleet

**speed_not_int**

Αποθηκεύεται η τρέχουσα κατάσταση των ειδοποιήσεων παραβίασης του ορίου ταχύτητας. Χρησιμοποιείται για την αποφυγή επαναλαμβανόμενων ειδοποιήσεων. Το πεδίο υποστηρίζει δύο καταστάσεις, οπότε επιλέχθηκε ο τύπος tinyint(1).

**email_not**

Περιέχει την επιλογή του χρήστη για παροχή ή μη παροχή ειδοποιήσεων μέσω ηλεκτρονικού ταχυδρομείου. Το πεδίο υποστηρίζει δύο καταστάσεις, οπότε επιλέχθηκε ο τύπος tinyint(1).

**report**

Περιέχει την επιλογή του χρήστη για την προσθήκη ή όχι του συγκεκριμένου οχήματος στις περιοδικές ειδοποιήσεις. Το πεδίο υποστηρίζει δύο καταστάσεις, οπότε επιλέχθηκε ο τύπος tinyint(1).

**celery_stasis_id**

Αυτό το πεδίο χρησιμοποιείται για την αποθήκευση του id ενός νήματος που έχει αναλάβει την ανίχνευση στάσης. Με κάθε νέα θέση του οχήματος, πρέπει να γίνει διακοπή του συγκεκριμένου νήματος και για να γίνει αυτό, απαιτείται να είναι γνωστό το id του. Το id έχει μήκος 40 χαρακτήρων και γι' αυτό χρησιμοποιείται ο τύπος δεδομένων varchar(40).

### 2.5.2 Fleet

Στον πίνακα Fleet αποθηκεύονται όλοι οι στόλοι οχημάτων των χρηστών. Για λόγους εξοικονόμησης χώρου έχει αποδοθεί σε κάθε όχημα το χαρακτηριστικό fleet_id σύμφωνα με το οποίο καθορίζεται και σε ποιον στόλο ανήκει.

**id**

Είναι το πρωτεύον κλειδί του πίνακα. Για κάθε εγγραφή που εισάγεται, δημιουργείται και ένα καινούριο, που είναι η τιμή του κλειδιού της προηγούμενης εγγραφής, αυξημένη

| Στήλη | Τύπος | Κενό | Προεπιλογή |
|---|---|---|---|
| id | int(11) | όχι | καμία |
| name | varchar(120) | όχι | καμία |
| surname | varchar(120) | όχι | καμία |
| email | varchar(120) | όχι | καμία |
| telephone | varchar(15) | όχι | καμία |
| address | varchar(120) | όχι | καμία |
| status | tinyint(1) | - | 0 |
| role | tinyint(1) | - | 1 |
| password | varchar(40) | όχι | καμία |
| registered | datetime | όχι | καμία |
| report | tinyint(1) | όχι | καμία |
| email_not | tinyint(1) | όχι | καμία |
| int_not | tinyint(1) | όχι | καμία |
| last_reported | datetime | όχι | καμία |

Πίνακας 2.3: Πίνακας User

κατά ένα. Έτσι χρησιμοποιήθηκε ο τύπος δεδομένων integer.

#### name

Είναι το όνομα του στόλου οχημάτων. Είναι αλφαριθμητικό και γι' αυτό χρησιμοποιήθηκε ο τύπος varchar(120).

#### description

Σε αυτό το πεδίο αποθηκεύεται προαιρετικά μια σύντομη περιγραφή του στόλου. Είναι αλφαριθμητικό και γι' αυτό χρησιμοποιήθηκε ο τύπος varchar(120).

### 2.5.3 User

Ο πίνακας User χρησιμοποιείται για την αποθήκευση όλων των χρηστών αλλά και των ρυθμίσεών τους στην εφαρμογή.

#### id

Είναι το πρωτεύον κλειδί του πίνακα. Για κάθε εγγραφή που εισάγεται, δημιουργείται και ένα καινούριο, που είναι η τιμή του κλειδιού της προηγούμενης εγγραφής αυξημένη κατά ένα. Έτσι χρησιμοποιήθηκε ο τύπος δεδομένων integer.

**name**

Το μικρό όνομα του χρήστη. Το πεδίο είναι κατά κανόνα αλφαριθμητικό, όποτε επιλέγεται τύπος varchar με ένα ικανοποιητικό μέγεθος για όλες τις περιπτώσεις. Το μήκος 120 χαρακτήρων θεωρείται υπεραρκετό.

**surname**

Το επίθετο του χρήστη. Το πεδίο αυτό όπως και το όνομα του χρήστη είναι κατά κανόνα αλφαριθμητικό, όποτε επιλέγεται τύπος varchar με ένα ικανοποιητικό μέγεθος για όλες τις περιπτώσεις. Το μήκος 120 χαρακτήρων θεωρείται υπεραρκετό και σε αυτή την περίπτωση.

**telephone**

Το τηλέφωνο του χρήστη. Αποθηκεύεται σε μορφή varchar γιατί μπορεί να περιέχονται και σύμβολα (-, +). Το μήκος του πεδίου ορίστηκε σε 15, γιατί σύμφωνα με το E.164 [17] του ITU-T, το μέγιστο μήκος ενός τηλεφώνου περιορίζεται στα 15 ψηφία.

**address**

Η διεύθυνση του χρήστη. Το πεδίο είναι κατά κανόνα αλφαριθμητικό, όποτε επιλέγεται τύπος varchar με ένα ικανοποιητικό μέγεθος για όλες τις περιπτώσεις. Το μήκος 120 χαρακτήρων θεωρείται υπεραρκετό.

**status**

Ένας χρήστης μπορεί να είναι, είτε ενεργός, είτε ανενεργός. Αυτό το πεδίο χρησιμοποιείται για αυτή την λειτουργιά. Η πληροφορία αυτή αποθηκεύεται ως εξής:

0: σημαίνει ότι ο χρήστης είναι ενεργός

1: σημαίνει ότι ο χρήστης είναι δεν είναι ενεργός

Υπάρχουν μόνο δύο δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**role**

Περιέχει την πληροφορία που φανερώνει το επίπεδο του χρήστη. Η εφαρμογή υποστηρίζει δύο επίπεδα χρηστών. Τους διαχειριστές και του απλούς χρήστες. Έτσι αυτό το πεδίο μπορεί να πάρει δυο τιμές.

0: σημαίνει ότι ο χρήστης είναι διαχειριστής

1: σημαίνει ότι είναι απλός χρήστης

Υπάρχουν μόνο δύο δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**password**

Σε αυτό το πεδίο αποθηκεύεται ο προσωπικός κωδικός του χρήστη. Για λόγους ασφαλείας, ο κωδικός που υπάρχει στην βάση είναι κρυπτογραφημένος με την συνάρτηση sha1. Η επιλογή του μεγέθους και του τύπου varchar(40) έγινε γιατί η συνάρτηση sha1 έχει ως έξοδο ένα αλφαριθμητικό μήκους 40 χαρακτήρων.

**email_not**

Ο χρήστης μπορεί να λαμβάνει ειδοποιήσεις και μέσω email. Σε αυτό το πεδίο αποθηκεύεται η προτίμηση για χρησιμοποίηση αυτής της λειτουργίας ή όχι.

Υπάρχουν μόνο δύο δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**int_not**

Ένας χρήστης μπορεί να θέλει να απενεργοποιήσει τις ειδοποιήσεις που δημιουργούνται εσωτερικά και είναι διαθέσιμες μέσω της ιστοσελίδας. Αυτό το πεδίο αποθηκεύει την προτίμηση του.

Υπάρχουν μόνο δύο δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**report**

Η εφαρμογή δίνει την δυνατότητα για λήψη περιοδικών αναφορών. Σε αυτό το πεδίο αποθηκεύεται η προτίμηση του χρήστη να τις λαμβάνει ή όχι και πότε. Συγκεκριμένα, μπορεί να πάρει τις εξής τιμές:

0: απενεργοποιημένο
1: ημερήσιες
2: εβδομαδιαίες
3: μηνιαίες
4: ετήσιες

Υπάρχουν μόνο πέντε δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**last_reported**

Για την λειτουργία των περιοδικών αναφορών, απαιτείται να είναι γνωστή η τελευταία ημερομηνία που έγινε αναφορά. Κάθε νέα αναφορά, ενημερώνει αυτό το πεδίο με την τρέχουσα ημερομηνία και ώρα. Χρησιμοποιείται πεδίο datetime, αφού πρόκειται για ημερομηνία.

| Στήλη | Τύπος | Κενό | Προεπιλογή |
|---|---|---|---|
| id | int(11) | όχι | καμία |
| dev_id | int(11) | όχι | καμία |
| latitude | numeric(12,8) | όχι | καμία |
| longitude | numeric(12,8) | όχι | καμία |
| speed | float | όχι | καμία |
| altitude | float | όχι | καμία |
| stasis | tinyint(1) | - | 0 |
| timestamp | datetime | όχι | καμία |

Πίνακας 2.4: Πίνακας Location

### 2.5.4 Location

Ο πίνακας Location χρησιμοποιείται για την αποθήκευση των θέσεων όλων των οχημάτων. Η θέση ενός οχήματος μπορεί να χαρακτηριστεί από τα παρακάτω στοιχεία.

#### id

Είναι το πρωτεύον κλειδί του πίνακα. Για κάθε εγγραφή που εισάγεται, δημιουργείται και ένα καινούριο, που είναι η τιμή του κλειδιού της προηγούμενης εγγραφής, αυξημένη κατά ένα. Έτσι χρησιμοποιήθηκε ο τύπος δεδομένων integer.

#### dev_id

Το όχημα στο οποίο αποδίδεται η εγγραφή. Με βάση το αναγνωριστικό κλειδί επικοινωνίας του κάθε οχήματος, αποθηκεύεται το id του για εξοικονόμηση χώρου. Πρόκειται για ξένο κλειδί του πίνακα Vehicle, οπότε χρησιμοποιείται ο ίδιος τύπος δεδομένων.

#### latitude

Το γεωγραφικό πλάτος του οχήματος. Επιλέχθηκε numeric, γιατί απαιτείται μεγάλη ακρίβεια στον αριθμό που αποθηκεύεται. Συγκεκριμένα χρησιμοποιήθηκε μήκος αριθμού 12 με 8 δεκαδικά ψηφία. Δηλαδή 4 ψηφία δεσμεύονται για το ακέραιο μέρος. Αν επιλεγόταν float, θα υπήρχε η περίπτωση να υπάρξει σφάλμα μετατροπής των δεδομένων και στρογγυλοποιήσεις.

#### longitude

Το γεωγραφικό μήκος του οχήματος. Επιλέχθηκε numeric(12,8) για τους ίδιους λόγους με την περίπτωση του γεωγραφικού πλάτους.

| Στήλη | Τύπος | Κενό | Προεπιλογή |
|---|---|---|---|
| id | int(11) | όχι | καμία |
| message | tinyint(1) | όχι | καμία |
| data_id | int(11) | όχι | καμία |
| read | tinyint(1) | - | 0 |

Πίνακας 2.5: Πίνακας Notification

**altitude**

Το υψόμετρο στο οποίο βρισκόταν το οχήματος. Επιλέχθηκε float, γιατί υπάρχει η περίπτωση να εισαχθούν δεκαδικές τιμές υψομέτρου.

**speed**

Η ταχύτητα του οχήματος. Επιλέχθηκε float, γιατί υπάρχει η περίπτωση να εισαχθούν δεκαδικές τιμές ταχύτητας.

**stasis**

Αποθηκεύεται η κατάσταση του οχήματος. Κάθε όχημα μπορεί να βρίσκεται σε κίνηση ή όχι. Αν ανιχνευθεί στάση, τότε αυτό το πεδίο ενημερώνεται αναλόγως.

0: το όχημα κινείται 1: το όχημα είναι σταματημένο

Υπάρχουν μόνο δύο δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**timestamp**

Αποθηκεύεται η στιγμή της καταγραφής όλων των παραπάνω δεδομένων. Χρησιμοποιείται πεδίο datetime αφού πρόκειται για ημερομηνία.

### 2.5.5 Notification

Οι ειδοποιήσεις που δημιουργούνται στην ιστοσελίδα αποθηκεύονται στον πίνακα Notification. Μια ειδοποίηση αποτελείται από τα παρακάτω πεδία:

**id**

Είναι το πρωτεύον κλειδί του πίνακα. Για κάθε εγγραφή που εισάγεται, δημιουργείται και ένα καινούριο, που είναι η τιμή του κλειδιού της προηγούμενης εγγραφής, αυξημένη κατά ένα. Έτσι χρησιμοποιήθηκε ο τύπος δεδομένων integer.

| Στήλη | Τύπος | Κενό | Προεπιλογή |
|---|---|---|---|
| id | int(11) | όχι | καμία |
| dev_id | int(11) | όχι | καμία |
| bill | float | - | 0 |
| liters | float | - | 0 |
| timestamp | datetime | όχι | καμία |

Πίνακας 2.6: Πίνακας Fuel

**message**

Μια ειδοποίηση μπορεί να έχει δημιουργηθεί για 5 διαφορετικούς λόγους. Οι τιμές που μπορεί να πάρει είναι:

0: παραβίαση γεωφράκτη

1: παραβίαση ορίου ταχύτητας

2: υπέρβαση χιλιομέτρων για συντήρηση

3: ανίχνευση στάσης

4: έκτακτη ανάγκη

Υπάρχουν μόνο πέντε δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

**data_id**

Η εγγραφή του πίνακα Location που δημιούργησε την ειδοποίηση. Πρόκειται για ξένο κλειδί του πίνακα Location, οπότε χρησιμοποιείται ο ίδιος τύπος δεδομένων.

**read**

Αποθηκεύει την κατάσταση της ειδοποίησης. Δηλαδή αν έχει διαβαστεί ή όχι από τον χρήστη.

0: δεν έχει διαβαστεί

1: έχει διαβαστεί

Υπάρχουν μόνο δυο δυνατές περιπτώσεις, οπότε μπορεί να χρησιμοποιηθεί πεδίο tinyint με προβολή ενός στοιχείου.

### 2.5.6 Fuel

Η κατανάλωση των οχημάτων ενός χρήστη ελέγχεται μέσω χειροκίνητης εισαγωγής των δεδομένων ανεφοδιασμού. Τα δεδομένα αυτά αποθηκεύονται στον πίνακα Fuel και αναλύονται παρακάτω.

**id**

Είναι το πρωτεύον κλειδί του πίνακα. Για κάθε εγγραφή που εισάγεται, δημιουργείται και ένα καινούριο, που είναι η τιμή του κλειδιού της προηγούμενης εγγραφής αυξημένη κατά ένα. Έτσι χρησιμοποιήθηκε ο τύπος δεδομένων integer.

**dev_id**

Κάθε εγγραφή ανεφοδιασμού συνδέεται άμεσα με το όχημα που ανεφοδιάστηκε. Σε αυτό το πεδίο αποθηκεύεται το id του οχήματος. Είναι ξένο κλειδί του πίνακα Vehicle οπότε χρησιμοποιείται ο ίδιος τύπος δεδομένων.

**bill**

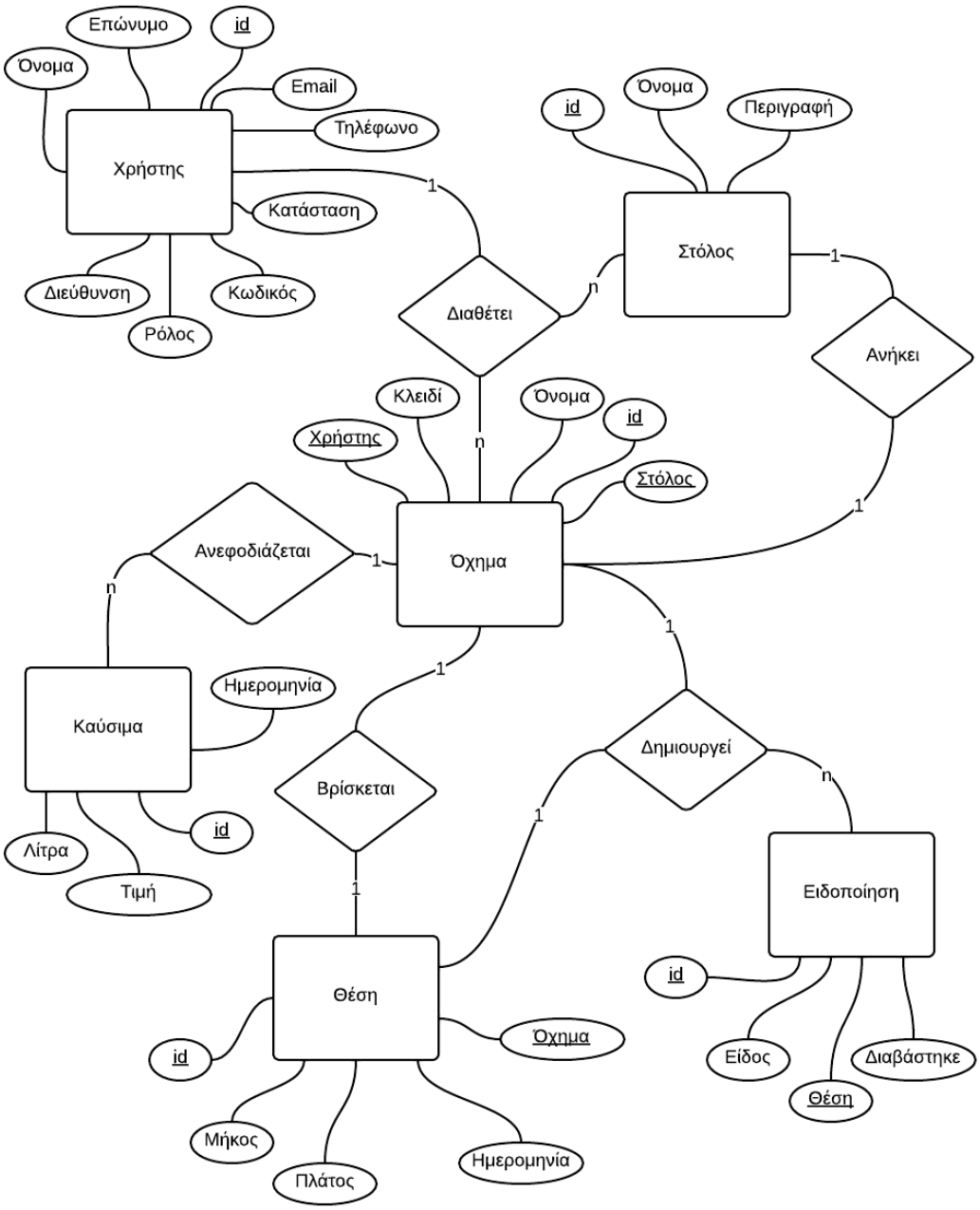
Αποθηκεύεται το ποσό που πληρώθηκε για τον ανεφοδιασμό. Επιλέχθηκε float γιατί υπάρχει η περίπτωση να υπάρχουν δεκαδικές τιμές λογαριασμών.

**liters**

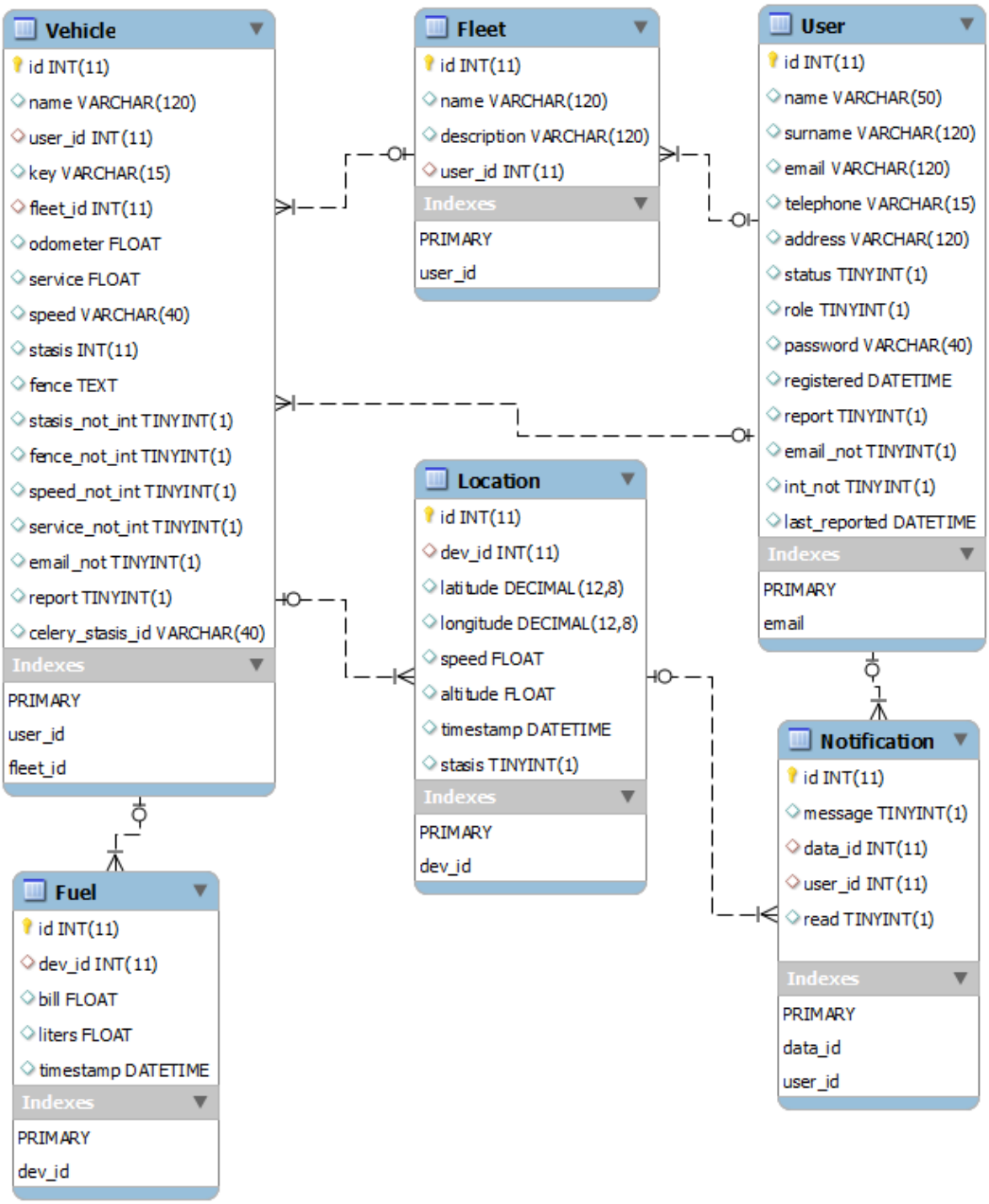
Ο αριθμός των λίτρων καυσίμου που πληρώθηκαν. Επιλέχθηκε float γιατί υπάρχει η περίπτωση να υπάρχουν δεκαδικές τιμές λίτρων.

**timestamp**

Αποθηκεύεται η ημερομηνία που έγινε ο ανεφοδιασμός. Χρησιμοποιείται πεδίο date-time αφού πρόκειται για ημερομηνία.




Σχήμα 2.2: Προσχέδιο βάσης δεδομένων




Σχήμα 2.3: Διάγραμμα οντοτήτων-συσχετίσεων

# Κεφάλαιο 3

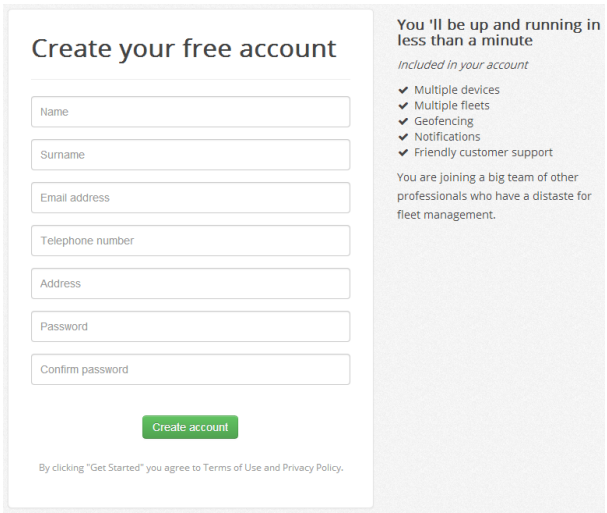
# Σχεδιασμός του ιστοχώρου

Στο προηγούμενο κεφάλαιο έγινε ο σχεδιασμός του συστήματος και ορίστηκε η αρχιτεκτονική του. Στο τρέχον κεφάλαιο θα παρουσιαστεί η λειτουργία και υλοποίηση της διαδικτυακής εφαρμογής.

## 3.1 Λειτουργίες χρήστη

### 3.1.1 Εγγραφή χρήστη

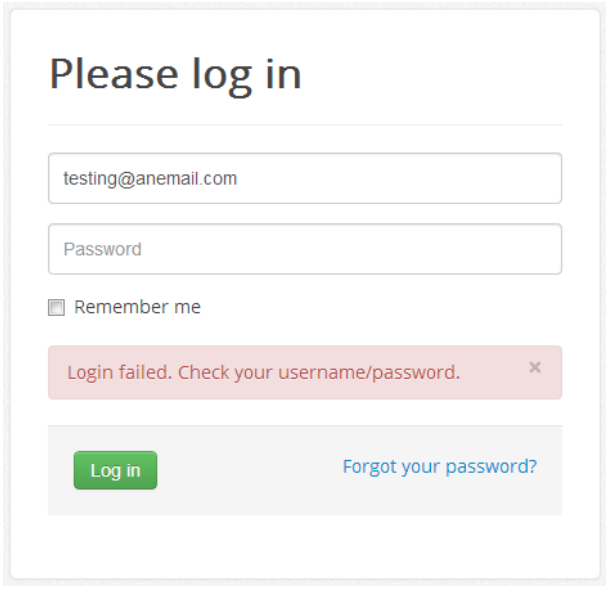
Η χρήση της εφαρμογής γίνεται αυστηρά από εγγεγραμμένους χρήστες. Η εγγραφή ενός χρήστη είναι ελεύθερη και πραγματοποιείται από την αρχική σελίδα επιλέγοντας τον σύνδεσμο Login. Κατά την εγγραφή ενός χρήστη απαιτούνται το ονοματεπώνυμο, ένα email, τηλέφωνο, και κωδικός πρόσβασης. Συμπληρωματικά, υπάρχει και η επιλογή για εισαγωγή της διεύθυνσης του χρήστη.




Σχήμα 3.1: Εγγραφή νέου χρήστη

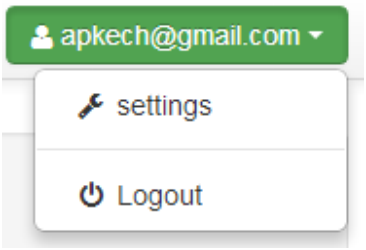
### 3.1.2 Σύνδεση και αποσύνδεση χρήστη

Η είσοδος ενός χρήστη στην εφαρμογή μπορεί να γίνει χρησιμοποιώντας την φόρμα στον υποκατάλογο login. Τα πεδία που απαιτούνται είναι το email και το συνθηματικό που έχει δηλώσει ο χρήστης κατά την εγγραφή. Πριν από οποιαδήποτε επεξεργασία των δεδομένων, γίνεται έλεγχος στο περιβάλλον του χρήστη με Javascript. Συγκεκριμένα, απαιτείται και τα δύο πεδία να μην είναι κενά, ενώ το email πρέπει να είναι και έγκυρο. Οι ίδιοι έλεγχοι πραγματοποιούνται και σε δεύτερη φάση από τον εξυπηρετητή.




Σχήμα 3.2: Αποτυχιά σύνδεσης

Όταν τα στοιχεία του χρήστη δεν είναι έγκυρα, εμφανίζεται ενημερωτικό μήνυμα σφάλματος, ενώ σε αντίθετη περίπτωση, ο χρήστης ανακατευθύνεται στον προσωπικό του λογαριασμό.




Σχήμα 3.3: Αποσύνδεση χρήστη

Η έξοδος από την εφαρμογή πραγματοποιείται από το σταθερό μενού στο πάνω μέρος της σελίδας, με την επιλογή Logout (Σχήμα 3.3). Κατά την έξοδο, διαγράφεται η συνεδρία του χρήστη και ακυρώνονται τα cookies που είχαν δημιουργηθεί.

### 3.1.3 Υπενθύμιση κωδικού

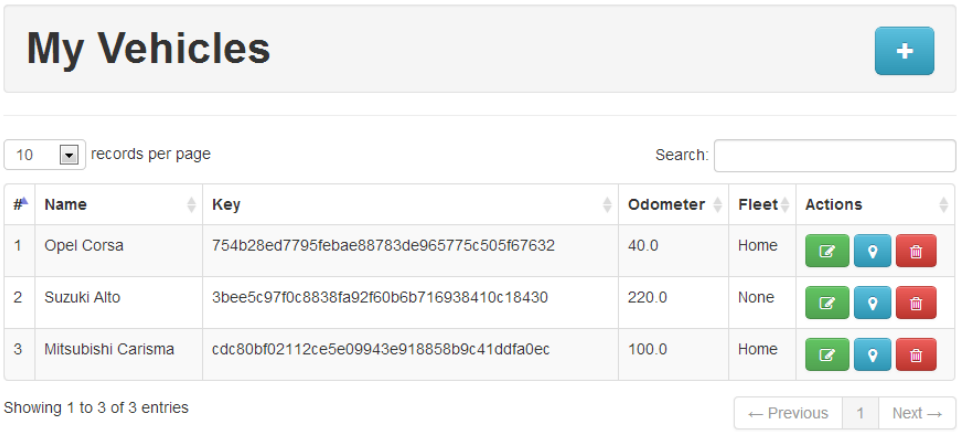
Σε περίπτωση απώλειας του κωδικού πρόσβασης, μπορεί να γίνει η ανάκτησή του από την επιλογή "Forgot your password?". Η εφαρμογή αλλάζει τον κωδικό με έναν τυχαία παραγόμενο και στέλνει τον νέο κωδικό στο προσωπικό email του χρήστη.

Ο κωδικός που δημιουργείται είναι ένα τμήμα της εξόδου μιας συνάρτησης κατακερματισμού. Στην συνέχεια μέσω του αλγορίθμου sha1 αποθηκεύεται στην βάση δεδομένων.

```
1 password = hashlib.sha1(os.urandom(10)).hexdigest()[:10]
```

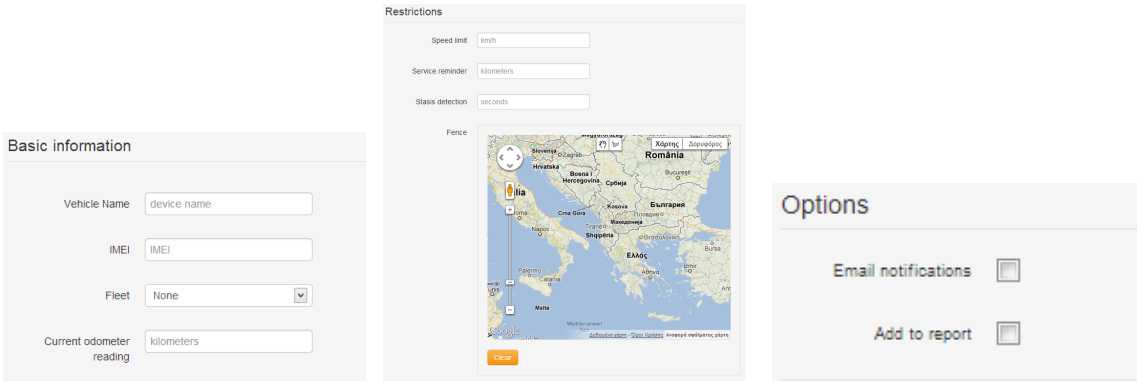
### 3.1.4 Διαχείριση οχημάτων

Η κεντρική διαχείριση των οχημάτων ενός χρήστη γίνεται από την επιλογή Vehicles στο αριστερό κεντρικό μενού. Ουσιαστικά, μπορεί να γίνει προσθήκη, επεξεργασία και διαγραφή οχημάτων. Εφόσον υπάρχουν ήδη οχήματα, εμφανίζεται μια λίστα με αυτά (Σχήμα 3.4) δίνοντας πληροφορίες σχετικά με το όνομα του οχήματος, το χαρακτηριστικό κλειδί, το οδόμετρο, και τον στόλο οχημάτων στον οποίο ανήκει. Στο ίδιο σημείο υπάρχουν και οι επιλογές για την επεξεργασία, την παρακολούθηση καθώς και την διαγραφή ενός οχήματος.




Σχήμα 3.4: Επισκόπηση οχημάτων

Κατά την προσθήκη ενός οχήματος ο χρήστης μπορεί να αρχικοποιήσει όλες τις διαθέσιμες παραμέτρους και λειτουργίες. Αυτές κατηγοριοποιούνται ως βασικές πληροφορίες, περιορισμούς και επιπλέον επιλογές. Τα μόνα απαιτούμενα πεδία είναι το όνομα του οχήματος και το IMEI, όμως μπορούν να οριστούν και ο στόλος στον οποίο θα ενταχθεί το όχημα, το οδόμετρο του, ένα όριο ταχύτητας, υπενθυμίσεις για συντήρηση, ανίχνευση στάσεων, γεωφράκτης, αν οι ειδοποιήσεις θα στέλνονται με email και αν το όχημα συμπεριλαμβάνεται στην περιοδική αναφορά (αν είναι ενεργοποιημένη στις ρυθμίσεις του χρήστη).




(α') βασικές πληροφορίες (β') περιορισμοί (γ') πρόσθετες επιλογές

Σχήμα 3.5: Προσθήκη-επεξεργασία οχήματος

Η επεξεργασία-προβολή ενός οχήματος περιέχει ακριβώς τα ίδια πεδία με την προσθήκη. Τέλος, η διαγραφή ενός οχήματος είναι οριστική. Αυτό σημαίνει ότι όταν ένα όχημα διαγραφεί, διαγράφονται και όλες οι πληροφορίες οι οποίες έχουν συλλεχθεί από τις συσκευές καταγραφής για αυτό.

### 3.1.5 Διαχείριση στόλου

Τα οχήματα μπορούν να ανήκουν ή να μην ανήκουν σε έναν στόλο οχημάτων. Όταν ένα όχημα δεν ανήκει σε κάποιον στόλο τότε η εφαρμογή δείχνει σαν προεπιλογή "None". Η δημιουργία ενός στόλου μπορεί να γίνει πολύ εύκολα χρησιμοποιώντας την επιλογή Fleets από το μενού. Για την δημιουργία απαιτείται το όνομα του στόλου και μια επεξήγηση. Στην ίδια φόρμα μπορούν να επιλεγούν και οι συσκευές που θα αποτελέσουν μέρος του στόλου (Σχήμα 3.6).

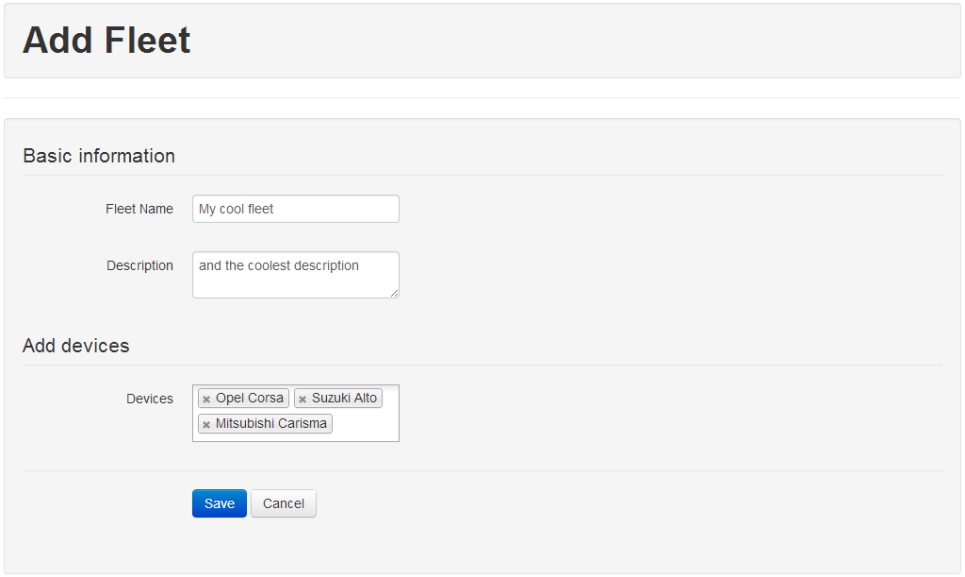
Η επεξεργασία του στόλου γίνεται με τον ίδιο τρόπο. Η προσθήκη και αφαίρεση οχημάτων από έναν στόλο, μπορεί να γίνει κάνοντας την επιθυμητή ενέργεια στην λίστα οχημάτων που εμφανίζεται και επιλέγοντας αποθήκευση. Αν δεν επιλεγεί αποθήκευση, δεν γίνεται κάποια αλλαγή στον στόλο.

Διαγραφή ενός στόλου μπορεί να γίνει από την κεντρική σελίδα της διαχείρισης στόλου. Στο ίδιο σημείο υπάρχει και επιλογή για την γρήγορη εμφάνιση των οχημάτων ενός στόλου (Σχήμα 3.8).

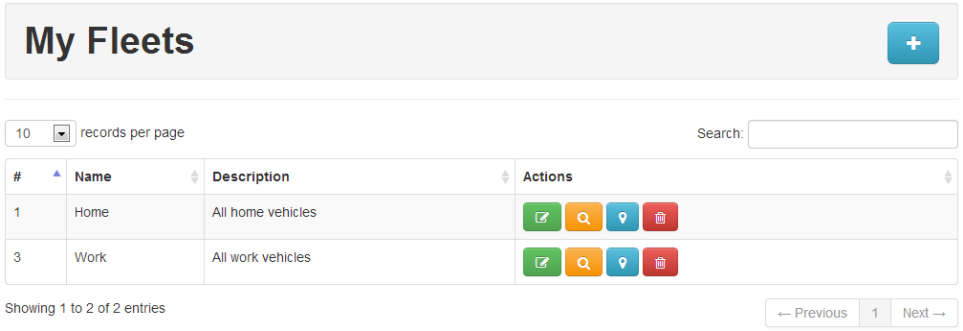
### 3.1.6 Ρυθμίσεις χρήστη

Ο χρήστης μέσω αυτής της επιλογής μπορεί να επεξεργαστεί και να προβάλει τις προσωπικές του πληροφορίες. Αυτή η επιλογή είναι διαθέσιμη σε δύο σημεία. Από το αριστερό κεντρικό μενού ή από το μενού στο πάνω δεξιό μέρος της σελίδας (Σχήμα 3.3).

Συγκεκριμένα, όλες οι πληροφορίες εμφανίζονται σε μια φόρμα στην οποία υπάρχει η δυνατότητα για επεξεργασία. Με την επιλογή για αποθήκευση, όλα τα στοιχεία που έχουν




Σχήμα 3.6: Προσθήκη στόλου οχημάτων

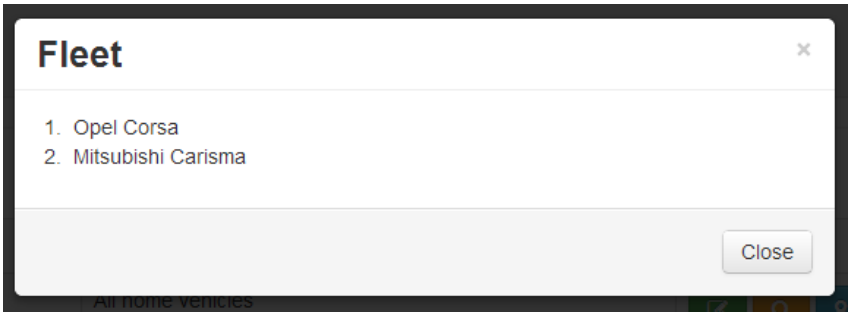



Σχήμα 3.7: Επισκόπηση διαθέσιμων στόλων

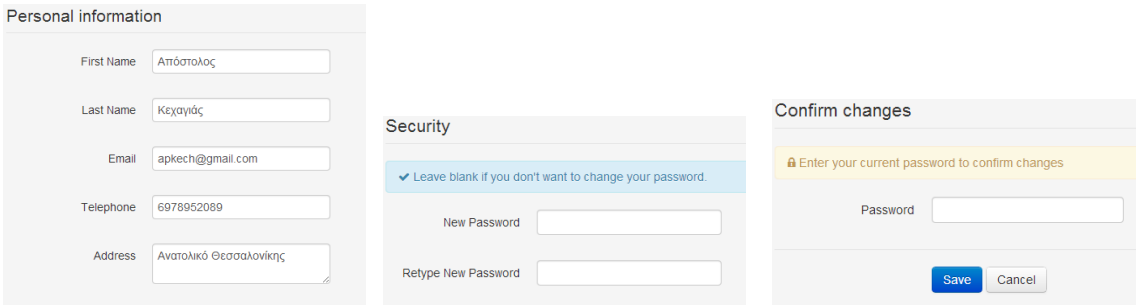
εισαχθεί, ελέγχονται για την εγκυρότητα τους. Εάν υπάρχουν σφάλματα, τότε εμφανίζεται στον πάνω μέρος της φόρμας μια λίστα με όλα τα σφάλματα. Από την ίδια φόρμα μπορεί να γίνει και αλλαγή του κωδικού πρόσβασης, ενεργοποίηση ή απενεργοποίηση των ειδοποιήσεων (εσωτερικών ή με email) και ο καθορισμός των περιοδικών αναφορών. Για λόγους ασφαλείας κάθε αλλαγή σε αυτά τα στοιχεία απαιτεί την εισαγωγή του ισχύοντος κωδικού. Τέλος, με την επιλογή ακύρωσης ο χρήστης μεταφέρεται στην κεντρική σελίδα και οι όποιες αλλαγές έχουν γίνει δεν αποθηκεύονται.

### 3.1.7 Περιοδικές αναφορές

Από τις ρυθμίσεις του, ο χρήστης μπορεί να ορίσει αν θέλει να λαμβάνει περιοδικές αναφορές για τα οχήματά του. Αυτές οι αναφορές μπορούν να είναι σε ημερήσια, εβδο-




Σχήμα 3.8: Επισκόπηση οχημάτων στόλου




(α΄) βασικές πληροφορίες (β΄) αλλαγή κωδικού (γ΄) επιβεβαίωση αλλαγών

Σχήμα 3.9: Ρυθμίσεις χρήστη

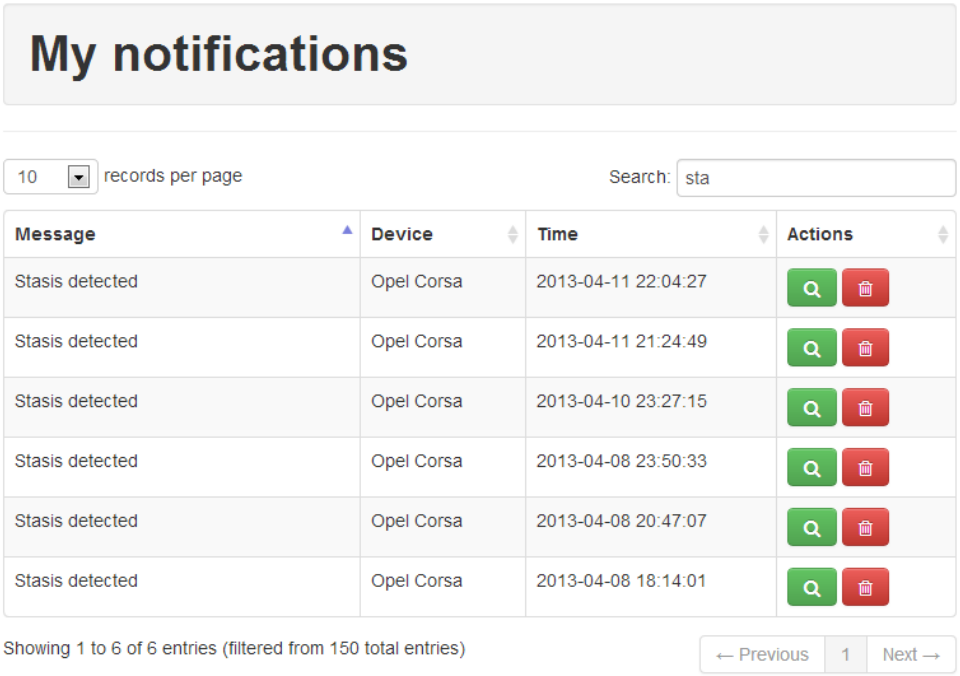
μαδιαία, μηνιαία και ετήσια βάση. Βέβαια, μπορούν να απενεργοποιηθούν και τελείως.

Οι περιοδικές αναφορές στέλνονται με email και ουσιαστικά πρόκειται για ένα αρχείο excell, που περιέχει όλες τις καταγεγραμμένες θέσεις των επιλεγμένων οχημάτων του χρήστη. Επίσης περιλαμβάνεται ο αριθμός χιλιομέτρων που διανύθηκαν, οι στάσεις, καθώς και η διάρκεια αυτών.

### 3.1.8 Ειδοποιήσεις

Η λειτουργία των ειδοποιήσεων είναι διαθέσιμη από το κεντρικό μενού στα αριστερά της σελίδας. Τα δεδομένα που παρέχονται μέσω αυτής, εξαρτώνται άμεσα από τις ρυθμίσεις που έχουν οριστεί για κάθε όχημα. Αφορούν την υπέρβαση ορίου ταχύτητας, την παραβίαση του γεωφράκτη, την ανίχνευση στάσης και υπενθυμίσεις σχετικά με την συντήρηση των οχημάτων.

Στις περιπτώσεις της ταχύτητας και του γεωφράκτη μπαίνει χρονικός περιορισμός στην συχνότητα των ειδοποιήσεων και ορίζεται από τον διαχειριστή του συστήματος. Αυτό έγινε γιατί, αν σε κάποιο σημείο ο οδηγός ενός οχήματος βγει εκτός φράκτη ή ξεπεράσει το όριο ταχύτητας, είναι πολύ πιθανό μελλοντικά δεδομένα να προξενήσουν την ίδια ακριβώς ειδοποίηση μέσα σε λίγα δευτερόλεπτα. Κάτι τέτοιο θα επιβάρυνε το σύστημα χωρίς ιδιαίτερο νόημα για τον χρήστη.




Σχήμα 3.10: Παράδειγμα ειδοποιήσεων

Σε κάθε περίπτωση, μια ειδοποίηση αποτελείται από ένα μήνυμα, το όχημα που την προκάλεσε, και την ακριβή ημερομηνία που δημιουργήθηκε (Σχήμα 3.10). Χρησιμοποιώντας την επιλογή των περισσότερων λεπτομερειών, εμφανίζεται ένας χάρτης σημειώνοντας το ακριβές σημείο του οχήματος την δεδομένη χρονική στιγμή. Επιπλέον, παρέχεται η δυνατότητα για διαγραφή των ειδοποιήσεων καθώς και για φιλτράρισμα και ταξινόμηση ανά χρόνο, μήνυμα και όχημα.

Συμπληρωματικά, ο χρήστης μπορεί να επιλέξει να λαμβάνει και ειδοποιήσεις μέσω ηλεκτρονικού ταχυδρομείου. Η λειτουργία και αυτών των ειδοποιήσεων είναι ανάλογη με τις εσωτερικές (Σχήμα 3.19).

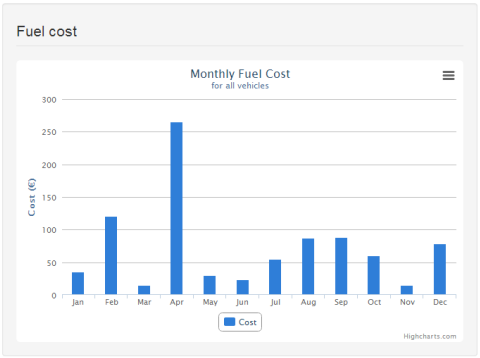
### 3.1.9 Λειτουργία στατιστικών

Η εφαρμογή παρέχει στατιστικά σχετικά με την λειτουργία των οχημάτων και στόλων. Η επιλογή προβολής στατιστικών είναι διαθέσιμη, στα σημεία που είναι διαθέσιμο και το όνομα του οχήματος ή στόλου.

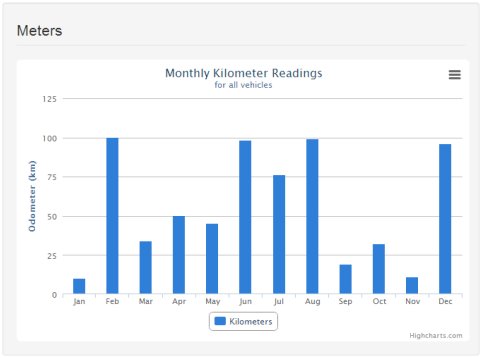
Τα στατιστικά αφορούν την κατανάλωση καυσίμων και τα χιλιόμετρα που διανύθηκαν σε χρονικό διάστημα ενός έτους, με ομαδοποίηση κατά μήνα.

### 3.1.10 Παρακολούθηση οχημάτων

Η παρακολούθηση των οχημάτων γίνεται από την επιλογή "Tracking". Μέσω αυτής, ο χρήστης μπορεί να παρακολουθήσει ένα η περισσότερα οχήματα και στόλους οχημάτων.

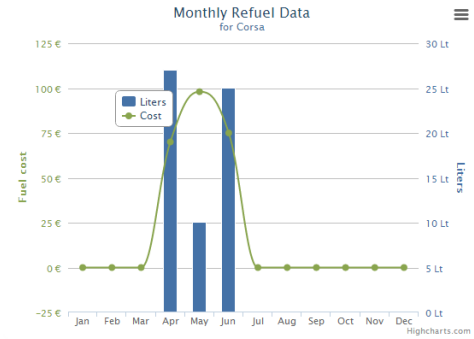



(α') Διάγραμμα κατανάλωσης ανά μήνα

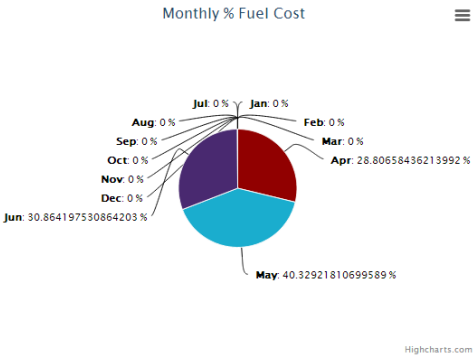



(β') Διάγραμμα χιλιομέτρων ανά μήνα

Σχήμα 3.11: Στατιστικά για όλα τα οχήματα

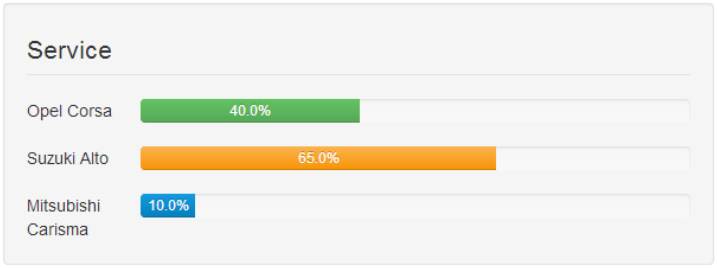



(α') Διάγραμμα κατανάλωσης ανά μήνα



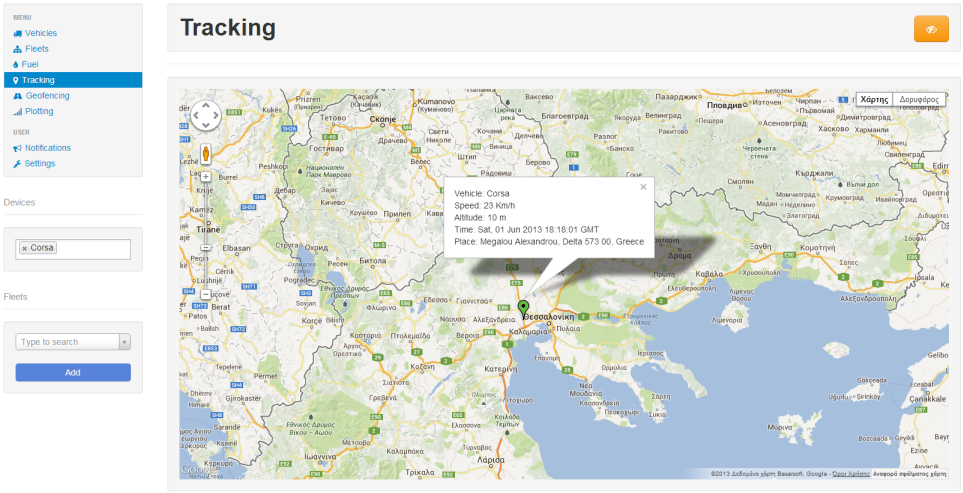

(β') Ποσοστιαίο διάγραμμα κατανάλωσης

Σχήμα 3.12: Στατιστικά οχήματος




Σχήμα 3.13: Ένδειξη για οχήματα προς συντήρηση

Στα αριστερά υπάρχει της σελίδας υπάρχει μια κεντρική επιλογή με την οποία μπορούν να προστίθενται και να αφαιρούνται οχήματα. Κάθε ενεργό όχημα εμφανίζεται σε αυτό το τμήμα με την κατάλληλη επιλογή για να αφαιρεθεί. Ακριβώς κάτω από αυτή την επιλογή, υπάρχει μια συμπληρωματική, με την οποία μπορούν να προστεθούν μαζικά οχήματα με βάση τον στόλο στον οποίο ανήκουν.

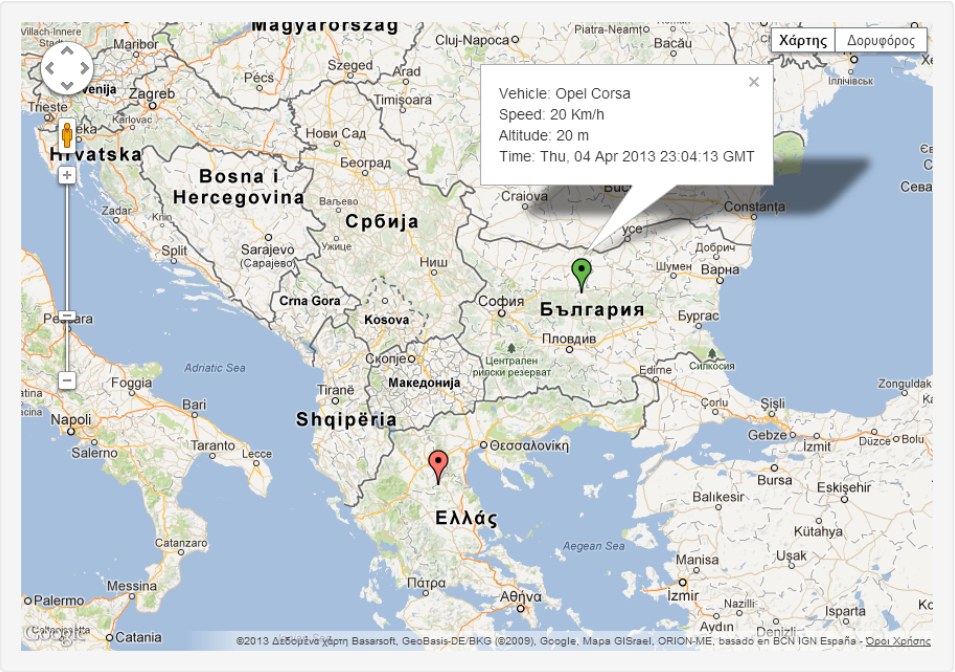



Σχήμα 3.14: Παρακολούθηση οχήματος

Τα οχήματα που είναι ενεργά, εμφανίζονται στον χάρτη [4] και μπορεί να βρίσκονται σε δύο καταστάσεις. Όταν η ένδειξη θέσης είναι κόκκινη, τότε το όχημα βρίσκεται σε στάση. Αντίθετα, όταν η ένδειξη είναι πράσινη, το όχημα βρίσκεται σε κίνηση. Περισσότερες πληροφορίες σχετικά με το όχημα παρέχονται μεσώ αναδυόμενων παραθύρων, έπειτα από την επιλογή του οχήματος.

### Υλοποίηση

Για την λειτουργία της παρακολούθησης οχημάτων χρησιμοποιείται η τεχνολογία AJAX. Με την προσθήκη ενός οχήματος, ενεργοποιείται η συνάρτηση loadMarker με όρισμα το id του.




Σχήμα 3.15: Κατάσταση οχημάτων

```
function loadMarker(id) {
  $.get('{{ url_for('get_device_position') }}' + id, function(result) {
    if(result.data)
      addMarker(
       new google.maps.LatLng(result.data.latitude,
       result.data.longitude),
       result.data.name,
       result.data.speed,
       result.data.altitude,
       result.data.timestamp,
       id,
       result.data.stasis,
       result.data.address);
    });
}
```

Μέσω αυτής, πραγματοποιείται μια ασύγχρονη αίτηση στην εφαρμογή. Η εφαρμογή επιστρέφει την τελευταία εγγραφή (εφόσον υπάρχει) του πίνακα Location σε μορφή JSON.

```
@app.route("/user/devices/position/", methods = ['GET', 'POST'])
@app.route("/user/devices/position/<int:id>", methods = ['GET', 'POST'])
@login_required
def get_device_position(id = 0):
    device = Device.query.filter(Device.user_id == g.user.id)
                         .filter(Device.id == id).first()
    if device:
        result = Data.query.filter(Data.dev_id == id)
                           .order_by(Data.timestamp.desc()).first()
        if result:
            address = geocode(result.latitude, result.longitude)
            result.address = ""

            if address:
                result.address = address

            result.name = device.name
            return jsonify(data=result.serialize())

    return jsonify(data=None)
```

Όπως φαίνεται στο παραπάνω τμήμα κώδικα, γίνεται και μια κλήση στην συνάρτηση geocode. Η συνάρτηση geocode χρησιμοποιεί το API της google για την ληψη της διεύθυνσης, χρησιμοποιώντας συντεταγμένες. Αν βρέθηκε διεύθυνση επιστρέφεται, αλλιώς επιστρέφεται μια κενή συμβολοσειρά.

```
def geocode(lat, lng):
    latlng = str(lat) + ',' + str(lng)
    url = """http://maps.googleapis.com/maps/api/geocode/json?latlng=%s&sensor
             =false&language=en""" % latlng

    jsondata = json.load(urllib2.urlopen(url))

    if jsondata['status'] == 'OK':
        return jsondata['results'][0]['formatted_address']

    return ""
```

Σε κάθε περίπτωση, η απάντηση του εξυπηρετητή εφόσον υπάρχουν δεδομένα, έχει την μορφή του παρακάτω παραδείγματος.

```
{
  "data": {
    "dev_id": 1,
    "name": "Opel Corsa",
    "timestamp": "2013-06-04T15:28:14",
    "altitude": 97.0,
    "stasis": true,
    "longitude": 22.9875,
    "address": "Konstantinou Paleologou 32, Pylaia-Chortiatis 555 35, Greece",
    "latitude": 40.5991,
    "speed": 8.70365
  }
}
```

Αν υπάρχουν έγκυρα δεδομένα, τότε γίνεται η κλήση της συνάρτησης addMarker. Αυτή ουσιαστικά προσθέτει το σύμβολο στον χάρτη και δημιουργεί το αναδυόμενο παράθυρο. Κάθε όχημα που εισάγεται στον χάρτη, προστίθεται στον πίνακα markers, ώστε να μπορεί να γίνει η αφαίρεση και ο επαναπροσδιορισμός της θέσης του.

```
function addMarker(location,title,speed,altitude,time,id,stasis,address) {
    if(stasis)
      icon = 'http://maps.google.com/mapfiles/marker.png';
    else
      icon = 'http://maps.google.com/mapfiles/marker_green.png';
    marker = new google.maps.Marker({
      position: location, map: map,
      title: title, icon: icon });
    marker['id'] = id;
    marker['infowindow'] = new google.maps.InfoWindow({
            content: "Vehicle: " + title  + "<br>" +
                     "Speed: " + speed + " Km/h" +
                     "<br>Altitude: " + altitude + " m" +
                     "<br>Time: " + time +
                     "<br>Place: " + address });
    google.maps.event.addListener(marker, 'click', function() {
        this['infowindow'].open(map, this);});
    markers.push(marker);
}
```

Όταν υπάρχουν στοιχεία στον πίνακα markers, ενεργοποιείται η συνάρτηση repositionMarkers, η οποία ανανεώνει την θέση του οχήματος και το περιεχόμενο του αναδυόμενου παραθύρου.

```
function repositionMarkers(){
 if(markers.length > 0){
   for(i in markers){
    $.get(url + markers[i]['id'], function(result) {
    position = new google.maps.LatLng(result.data.latitude,
                                      result.data.longitude);
    markers[i].setPosition(position);
    markers[i]['infowindow'].setContent(
                "Vehicle: " + result.data.name + "<br>" +
                "Speed: " + result.data.speed + " Km/h" +
                "<br>Altitude: " + result.data.altitude + " m" +
                "<br>Time: " + new Date(result.data.timestamp).toUTCString() +
                "<br>Place: " + result.data.address);
    if(result.data.stasis)
      markers[i].setIcon('http://maps.google.com/mapfiles/marker.png');
    else
      markers[i].setIcon('http://maps.google.com/mapfiles/marker_green.png');
    });
  }
 }
}
```

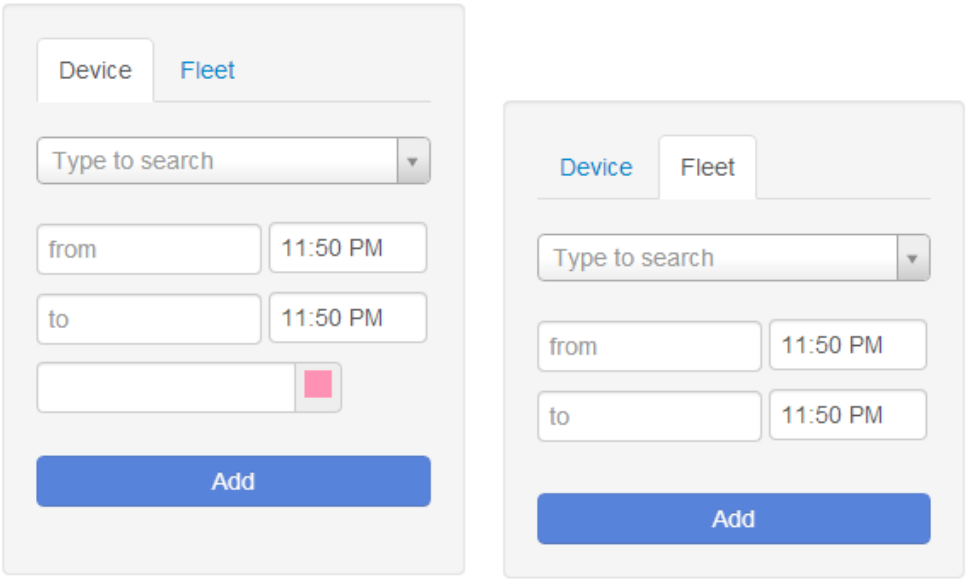
Αυτή η διαδικασία πραγματοποιείται περιοδικά, χρησιμοποιώντας την συνάρτηση setInterval.

### 3.1.11 Παραλαβή δεδομένων

Η παραλαβή των δεδομένων είναι ίσως το σημαντικότερο τμήμα της εφαρμογής. Λόγω της φύσης των συσκευών, αυτή η διαδικασία πρέπει να γίνεται όσο το δυνατόν γρηγορότερα. Σε αυτή την διαδικασία όμως πρέπει να εκτελεστούν αλγόριθμοι, οι οποίοι ανάλογα με την περίπτωση, μπορούν να καθυστερήσουν αυτήν την διαδικασία.

Έτσι η υλοποίηση ακολούθησε την εξής λογική. Ένα όχημα στέλνει τα δεδομένα προς αποθήκευση στην εφαρμογή. Η εφαρμογή αποθηκεύει αμέσως τα δεδομένα και δημιουργεί τις εργασίες που απαιτούνται προς εκτέλεση στο παρασκήνιο [11]. Με αυτόν τον τρόπο, αποδεσμεύεται η συσκευή από την συγκεκριμένη επικοινωνία όσο το δυνατόν γρηγορότερα και μπορεί να μεταδώσει ξανά, ακόμη και αν οι εργασίες που υπάρχουν στο παρασκήνιο, δεν έχουν ακόμη εκτελεστεί.

Αρχικά, γίνεται ένα ερώτημα στην βάση δεδομένων, για να επιβεβαιωθεί ότι υπάρχει όχημα με το κλειδί που χρησιμοποιήθηκε. Αν υπάρχει, τότε γίνεται έλεγχος εγκυρότητας σε όλα τα υπόλοιπα δεδομένα και γίνεται αποθήκευση. Αν δεν βρεθεί όχημα με το συγκεκριμένο κλειδί ή τα δεδομένα κριθούν άκυρα, δεν γίνεται αποθήκευση και τερματίζεται η σύνδεση.




(α') προσθήκη οχήματος (β') προσθήκη στόλου

Σχήμα 3.16: Επιλογές διαδρομών

### 3.1.12 Λειτουργία ανεφοδιασμών

Η λειτουργία ανεφοδιασμών προσφέρει στον χρήστη την δυνατότητα να κρατά αρχείο με τους ανεφοδιασμούς που έχει πραγματοποιήσει. Τα στοιχεία που απαιτούνται για την προσθήκη μιας εγγραφής είναι το όχημα, το ποσό πληρωμής, τα λίτρα καυσίμου και η ημερομηνία που πραγματοποιήθηκε.

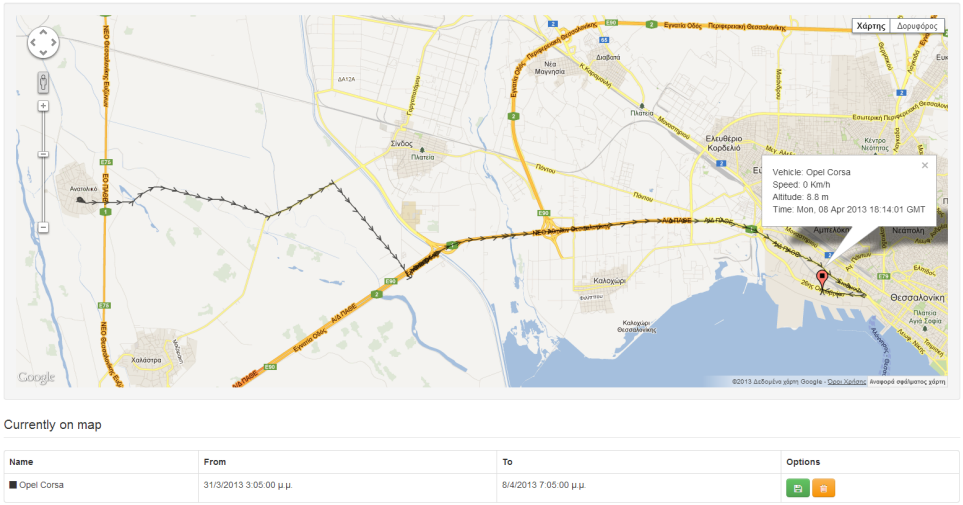
Οι εγγραφές που υπάρχουν ήδη στο σύστημα, εμφανίζονται στον χρήστη σε μια λίστα. Μέσω αυτής, ο χρήστης μπορεί να επεξεργαστεί ή να διαγράψει μια εγγραφή.

### 3.1.13 Λειτουργία προβολής διαδρομών

Μέσω αυτής της λειτουργίας, ο χρήστης μπορεί να προβάλλει σε χάρτη τις διαδρομές που έχουν πραγματοποιήσει τα οχήματά του. Μπορεί να επιλέξει μεμονωμένα οχήματα αλλά και στόλους οχημάτων (Σχήμα 3.16α', 3.16β'). Δηλαδή, δίνεται η δυνατότητα για απεικόνιση πολλαπλών οχημάτων και πολλαπλών διαδρομών ενός οχήματος.

Για την προσθήκη μιας διαδρομής πρέπει να επιλεγεί ένα όχημα, να οριστεί το χρονικό διάστημα που θέλει ο χρήστης να εμφανίσει και ένα χρώμα που χρησιμοποιείται για τον χρωματισμό της διαδρομής. Κατά την προσθήκη ενός στόλου το χρώμα των οχημάτων επιλέγεται αυτόματα.

Σε κάθε περίπτωση, αν υπάρχουν διαθέσιμα δεδομένα για προβολή, εμφανίζονται τα οχήματα στο κάτω μέρος του χάρτη. Εκεί φαίνεται ο χρωματισμός που χρησιμοποιείται για κάθε διαδρομή, σε ποιο όχημα ανήκει και σε πιο χρονικό διάστημα αναφέρεται. Από το ίδιο σημείο, δίνεται η δυνατότητα να μεταφορτωθούν τα δεδομένα ενός οχήματος σε




Σχήμα 3.17: Παράδειγμα διαδρομής

μορφή CSV και να αφαιρεθούν διαδρομές από τον χάρτη.

Στο παράδειγμα του Σχήματος 3.17 μπορούμε να δούμε την διαδρομή που έχει ακολουθήσει το όχημα με όνομα Opel Corsa. Τα δεδομένα αφορούν την περίοδο από 31/3/2013 και 3:05 μ.μ μέχρι 8/4/2013 και 7:05 μ.μ ενώ το χρώμα που έχει επιλεγεί είναι το μαύρο. Επίσης μπορούμε να δούμε ότι υπάρχει ένα σημείο με το σύμβολο της παύσης. Με αυτό τον τρόπο εμφανίζονται τα σημεία στα οποία έχει ανιχνευθεί στάση.

### Υλοποίηση

Η υλοποίηση της προβολής διαδρομών λειτουργεί με ασύγχρονες κλήσεις για την παραλαβή των δεδομένων. Αυτό γίνεται κυρίως για την βελτίωση της εμπειρίας χρήστη. Επιπλέον, η προσθήκη πολλαπλών διαδρομών σε έναν χάρτη, θα ήταν δύσκολο να πραγματοποιηθεί με διαφορετικό τρόπο.

Όταν επιλέγεται η προσθήκη ενός οχήματος, προετοιμάζονται τα δεδομένα ώστε να πραγματοποιηθεί μια αίτηση AJAX.

```
$("a#add_device").on( "click", function(e) {
    id = $("#select_device").val();
    fd = $("#from_dev").val();
    td = $("#to_dev" ).val();
    color = $( "#cpd" ).val();
    if(fd != "" && td != "" && id != ""){
        f_date = new Date(fd).getTime();
        t_date = new Date(td).getTime();
            loadLine(id , f_date, t_date, color);
        $("#select_device").select2("val","0");
        $("a#add_device").addClass('disabled');
    }
});
```

Με αυτό το javascript event συλλέγονται όλα τα δεδομένα( id, αρχή σε ms, τέλος σε ms, χρώμα). Έπειτα, καλείται η συνάρτηση loadLine, η οποία και πραγματοποιεί την κλήση για την παραλαβή των δεδομένων προς σχεδιασμό. Αν στα δεδομένα που έχουν ληφθεί, υπάρχουν σημεία με την μεταβλητή stasis σε αληθή κατάσταση, τότε γίνεται προσθήκη των σημείων στον χάρτη ως markers.

```
function loadLine(id, f, t, color) {
  $.get(url + id + '/' + f + '/' + t, function(result) {
      if(result.data.length > 0) addLine(result.data, color);
          for(i in result.data){
           if(result.data[i].stasis == true){
                marker = new google.maps.Marker({
                  position: new google.maps.LatLng(result.data[i].latitude,
                                                   result.data[i].longitude),
                  map: map,
                  icon: 'http://maps.google.com/mapfiles/dd-end.png',
                });
                if(!markers[lines.length - 1]){
                    markers[lines.length - 1] = [];
                }
                markers[lines.length - 1].push(marker);
            }
        }
    });
}
```

Η απάντηση του εξυπηρετητή είναι μια λίστα από δεδομένα για κάθε σημείο που βρέθηκε, μορφοποιημένη σε JSON.

```
{
  "data": [
        {
          "name": "Corsa",
          "timestamp": "2013-06-01T18:18:01",
          "altitude": 10.0,
          "stasis": false,
          "longitude": 22.7122,
          "address": "Megalou Alexandrou, Delta 573 00, Greece",
          "latitude": 40.6599,
          "speed": 23.0
        },
        {
          "name": "Corsa",
          "timestamp": "2013-06-01T18:18:08",
          "altitude": 12.0,
          "stasis": true,
          "longitude": 22.7128,
          "address": "Megalou Alexandrou, Delta 573 00, Greece",
          "latitude": 40.66,
          "speed": 66.0
        }
    ]
}
```

Όταν υπάρχουν δεδομένα προς εμφάνιση, καλείται η συνάρτηση addLine με όρισμα τα δεδομένα και το επιλεγμένο χρώμα. Εκεί προστίθενται όλα τα σημεία στην global μεταβλητή points και δημιουργείται μια γραμμή με βάση όλα τα σημεία που βρέθηκαν.

```
function addLine(data, color){
    var points = [];
    for( i in data ){
     points.push(new google.maps.LatLng(data[i].latitude, data[i].longitude));
    }
    var line = new google.maps.Polyline({
          path: points,
          strokeColor: color,
          strokeOpacity: 0.7,
          strokeWeight: 1.4,
    });
    line.setMap(map);
    lines.push(line);
}
```

Η ίδια διαδικασία πραγματοποιείται και για την προσθήκη ενός στόλου οχημάτων.

Η μόνη διαφορά είναι, ότι οι κλήσεις για δεδομένα γίνονται επαναληπτικά για κάθε όχημα του επιλεγμένου στόλου. Ο χρωματισμός στην προσθήκη στόλου γίνεται αυτόματα χρησιμοποιώντας μια συνάρτηση παραγωγής τυχαίων χρωμάτων.

```
function get_random_color() {
    var letters = '0123456789ABCDEF'.split('');
    var color = '#';

    for (var i = 0; i < 6; i++ ) {
        color += letters[Math.round(Math.random() * 15)];
    }
    return color;
}
```

### 3.1.14 Λειτουργία οδομέτρου

Η λειτουργία του οδομέτρου ενεργοποιείται με την παραλαβή δεδομένων. Συγκεκριμένα, κάθε φορά που αποθηκεύονται δεδομένα, εκτελείται μια εργασία στο παρασκήνιο. Αυτή η εργασία χρησιμοποιεί την μέθοδο Haversine [6] [22] για να υπολογιστεί η απόσταση μεταξύ των συντεταγμένων που παραλήφθηκαν με τις αμέσως προηγούμενες τιμές.

$$a = \sin^2(\Delta\phi/2) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2(\Delta\lambda/2) \tag{3.1}$$

$$c = 2 \cdot \arctan \frac{\sqrt{a}}{1 + \sqrt{1-a}} \tag{3.2}$$

$$d = R \cdot c \tag{3.3}$$

Όπου $\phi$ είναι το γεωγραφικό πλάτος, $\lambda$ το γεωγραφικό μήκος, $R$ η μέση τιμή της ακτίνας της γης.

```
class Haversine:
    earthRadius = 6371
    d2r = (math.pi / 180.0)

    def HaversineInKM(self, lat_f, lng_f, lat_t, lng_t):
        dlng = (lng_t - lng_f) * self.d2r
        dlat = (lat_t - lat_f) * self.d2r

        lat_t = lat_t * self.d2r
        lat_f = lat_f * self.d2r

        a = math.sin(dlat / 2.0) * math.sin(dlat / 2.0)
                                 + math.sin(dlng / 2.0)
                                 * math.sin(dlng / 2.0)
                                 * math.cos(lat_f)
                                 * math.cos(lat_t)
        c = 2.0 * math.atan2(math.sqrt(a), math.sqrt(1.0 - a))
        d = self.earthRadius * c;

        return d;
```

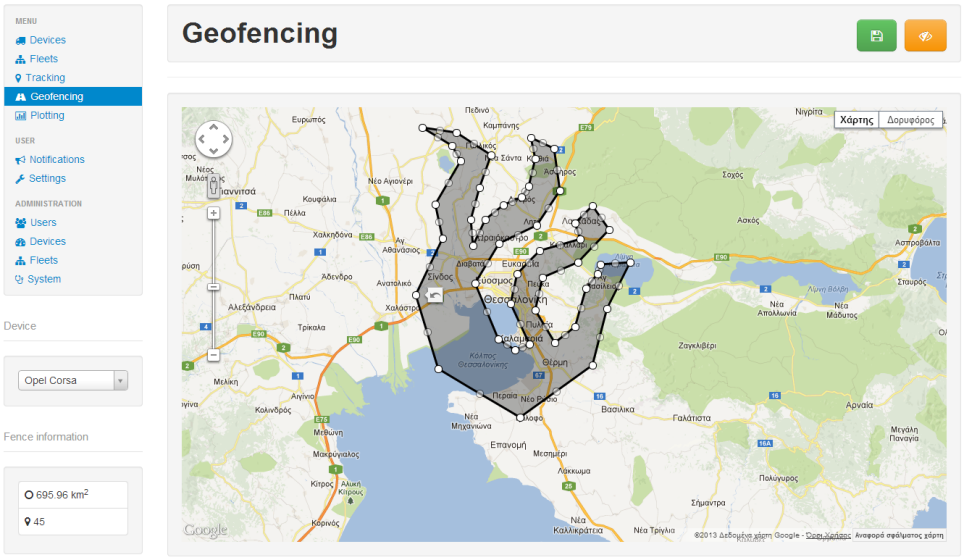
Ο υπολογισμός της απόστασης με αυτή την μέθοδο δεν είναι απόλυτα ακριβής. Γίνεται με την παραδοχή ότι η γη είναι σφαιρική. Στην πραγματικότητα είναι ελλειψοειδής ή καλύτερα ένα πεπλατυσμένο σφαιροειδές, με μια ακτίνα που κυμαίνεται από 6378 χλμ (ισημερινός) μέχρι και 6357 χλμ (πόλοι). Η γενικά αποδεκτή μέση τιμή για την ακτίνα της γης είναι 6371 χιλιόμετρα. Αυτό σημαίνει ότι υποθέτοντας σφαιρική γεωμετρία εισάγουμε σφάλμα 0.55% στον υπολογισμό μας. Η τιμή αυτή όμως δεν ξεπερνά το 0.3% στις περισσότερες περιπτώσεις, δηλαδή, ακρίβεια καλύτερη από 3 μέτρα σε κάθε 1 χιλιόμετρο.

### 3.1.15 Λειτουργία γεωφράκτη

Ο χρήστης μπορεί να ορίσει μόνο μια περιοχή στην οποία μπορεί να κινείται το όχημα. Η σχεδίαση του φράκτη γίνεται με πολύ απλό τρόπο, τοποθετώντας σημεία στο χάρτη. Δεν υπάρχει περιορισμός σημείων αλλά ούτε και σχήματος που μπορεί να δημιουργηθεί. Ο γεωφράκτης αφορά αποκλειστικά συσκευές-οχήματα και όχι στόλους οχημάτων. Παρ' όλα αυτά, υπάρχει η δυνατότητα να οριστεί ένας φράκτης και να αντιγραφεί για όλες τις συσκευές ενός στόλου οχημάτων.

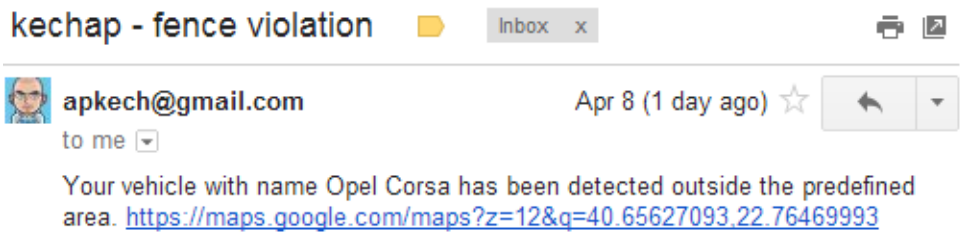
Ο φράκτης μπορεί να οριστεί κατά την προσθήκη ενός οχήματος ή από την επιλογή επεξεργασίας ή από το μενού, επιλέγοντας την επιθυμητή συσκευή. Εφόσον υπάρχει ήδη ορισμένος φράκτης, αυτός εμφανίζεται στον χρήστη και δίνεται η δυνατότητα για μετακίνηση των σημείων που τον καθορίζουν. Επίσης, μπορούν να προστεθούν σημεία σέρνοντας τα σημεία που εμφανίζονται με σκουρότερο χρώμα.

Όταν ένα όχημα βρεθεί εκτός της περιοχής που έχει οριστεί, δημιουργείται μια εσω-




Σχήμα 3.18: Ένα παράδειγμα γεωφράκτη

τερική ειδοποίηση και στέλνεται email (αν έχει επιλεγεί στις ρυθμίσεις του οχήματος) στον χρήστη. Στο email, υπάρχει σύνδεσμος που δείχνει το ακριβές σημείο στο οποίο εντοπίστηκε η παραβίαση, χωρίς να απαιτείται είσοδος στην εφαρμογή.




Σχήμα 3.19: Email παραβίασης γεωφράκτη

Ο έλεγχος για το αν το όχημα βρίσκεται εντός ή εκτός φράκτη, πραγματοποιείται με κάθε νέο ζεύγος συντεταγμένων. Χρησιμοποιείται και σε αυτήν την περίπτωση, μια εργασία που εκτελείται στο παρασκήνιο, για τους λόγους που αναφέρθηκαν παραπάνω.

### Σχεδίαση γεωφράκτη

Για την σχεδίαση του φράκτη χρησιμοποιείται η βιβλιοθήκη drawing. Η βιβλιοθήκη συμπεριλαμβάνεται στο Google Maps API v3, όμως πρέπει να δηλωθεί ότι χρησιμοποιείται. Αυτό γίνεται ως εξής.

```
1 https://maps.googleapis.com/maps/api/js?v=3.exp&sensor=false&libraries=drawing
```

Με την αρχικοποίηση του χάρτη, ορίζεται ένα αντικείμενο της κλάσης DrawingManager. Σε αυτό ορίζεται η κατάσταση της εργαλειοθήκης, το που αυτή θα εμφανιστεί και το είδος του σχήματος που θα σχεδιαστεί.

```
drawingManager = new google.maps.drawing.DrawingManager({
  drawingControl: true,
  drawingControlOptions: {
    position: google.maps.ControlPosition.TOP_CENTER,
    drawingModes: [
      google.maps.drawing.OverlayType.POLYGON
    ]
  },
  polygonOptions
});
```

Σε αυτή την περίπτωση η εργαλειοθήκη είναι ενεργή (drawingControl: true), εμφανίζεται στο πάνω μέρος και στο κέντρο του χάρτη με σχήμα πολύγωνο (OverlayType POLYGON).

Για την ενεργοποίηση του αντικειμένου DrawingManager, πρέπει να γίνει η σχετική ενημέρωση του αντικειμένου που χρησιμοποιείται για την εμφάνιση του χάρτη.

```
drawingManager.setMap(map);
```

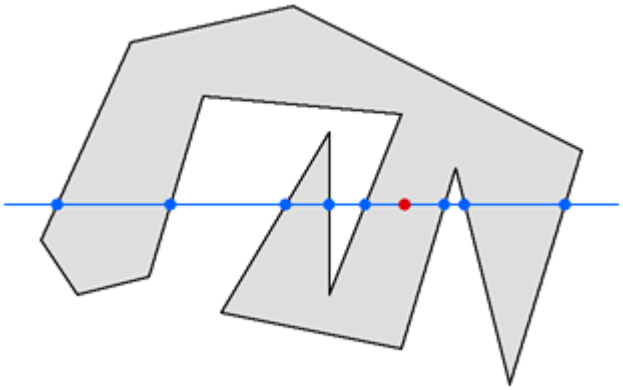
Έπειτα, πρέπει να οριστούν οι ενέργειες που θα πραγματοποιηθούν μετά το τέλος του σχεδιασμού. Αυτό γίνεται με την χρησιμοποίηση ενός listener στο event overlaycomplete, με όρισμα το αντικείμενο DrawingManager.

```
google.maps.event.addListener(drawingManager, 'overlaycomplete', function(e) {
    if (e.type != google.maps.drawing.OverlayType.MARKER) {

        drawingManager.setDrawingMode(null);
        drawingManager.setOptions({drawingControl: false});

        points = e.overlay.getPath().getArray();

        var newShape = e.overlay;
        newShape.type = e.type;

        setSelection(newShape);
    }
});
```

Με την ενεργοποίηση του event γίνεται αλλαγή σε κατάσταση προβολής και όχι σχεδιασμού. Έπειτα, συλλέγονται όλα σημεία που απαρτίζουν τον φράκτη και αποθηκεύονται σε μια global μεταβλητή (points). Τελικά, επιλέγεται το αντικείμενο που σχεδιάστηκε, ώστε να μπορεί να τροποποιηθεί και να διαγραφεί από τον χρήστη.

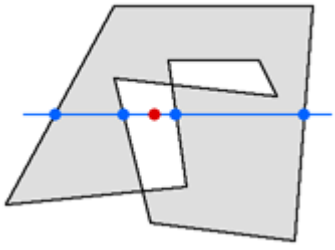
### Ο αλγόριθμος απόφασης

Ο αλγόριθμος [2] που χρησιμοποιήθηκε, λειτουργεί χρησιμοποιώντας ένα όριο $Y$ και το πόσες φορές αυτό συναντάται, με τις πλευρές του πολυγώνου που θέλουμε να ελέγξουμε. Συγκεκριμένα, δημιουργούμε έναν οριζόντιο νοητό άξονα στο σημείο $Y$ και καταρτίζουμε μια λίστα από κόμβους, όπου κάθε κόμβος είναι ένα σημείο που μία πλευρά του πολυγώνου τέμνει τον άξονα.



Σχήμα 3.20: Σημείο εντός πολυγώνου

Στο παράδειγμα που φαίνεται στο Σχήμα 3.20, οκτώ πλευρές του πολυγώνου τέμνονται με το όριο Y, ενώ οι άλλες έξι πλευρές δεν το κάνουν. Έτσι, εάν ο αριθμός των κόμβων σε κάθε πλευρά του σημείου δοκιμής είναι μονός, το σημείο είναι εσωτερικό του πολυγώνου. Αντίθετα, εάν υπάρχει άρτιος αριθμός κόμβων σε κάθε πλευρά του σημείου δοκιμής, τότε το σημείο είναι εξωτερικό του πολυγώνου. Στο παράδειγμα της εικόνας, υπάρχουν πέντε κόμβοι στα αριστερά και τρεις στα δεξιά του σημείου. Οι αριθμοί 5 και 3 είναι μονοί που σημαίνει ότι το σημείο βρίσκεται στο εσωτερικό του πολυγώνου.



Σχήμα 3.21: Σημείο εκτός πολυγώνου

Η ίδια λογική λειτουργεί ακόμη και αν το πολύγωνο είναι πιο πολύπλοκο, όπως στο Σχήμα 3.21. Ο άξονας στο σημείο δοκιμής τέμνει σε δύο πλευρές από δεξιά και δύο από αριστερά. Ο αριθμός των κόμβων είναι άρτιος που σημαίνει ότι το σημείο δοκιμής είναι εκτός του πολυγώνου.

Η υλοποίηση του αλγορίθμου σε γλώσσα Python:

```
class Geofence:
    points = list()

    def __init__(self, points):
        self.points = points

    def check(self, x, y):
        sides = len(self.points)
        j = sides - 1
        pointStatus = False
        for i in range(0, sides):
            if(self.points[i][1] < y
               and self.points[j][1] >= y
               or self.points[j][1] < y
               and self.points[i][1] >= y):
                if(self.points[i][0]
                   + (y - self.points[i][1])
                   / (self.points[j][1] - self.points[i][1])
                   * (self.points[j][0] - self.points[i][0])
                   < x):
                    pointStatus = not pointStatus
            j = i
        return pointStatus
```

## 3.2 Λειτουργίες διαχειριστή

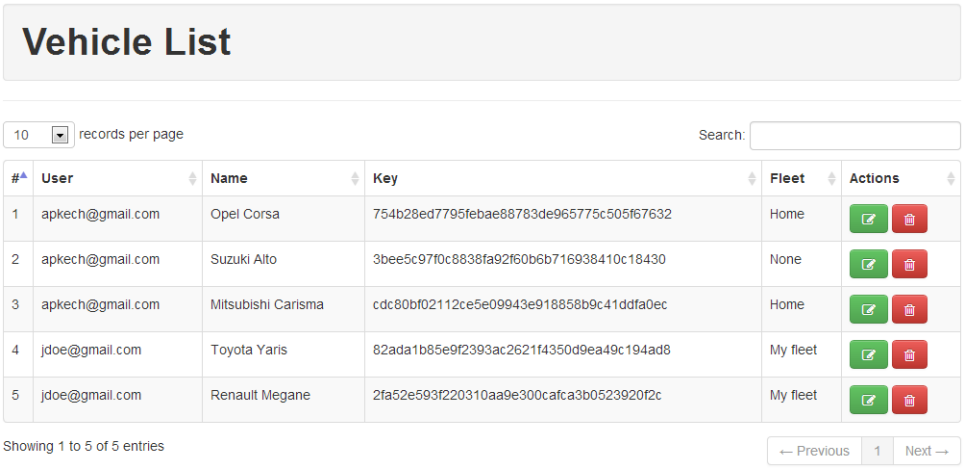
Ο ρόλος του διαχειριστή υλοποιείται σαν υπερσύνολο του ρόλου του χρήστη. Άρα, είναι ένας απλός χρήστης με κάποιες παραπάνω δυνατότητες και όλες οι λειτουργίες που αναφέρονται ως λειτουργίες χρήστη, ισχύουν και για τον διαχειριστή.

### 3.2.1 Διαχείριση οχημάτων

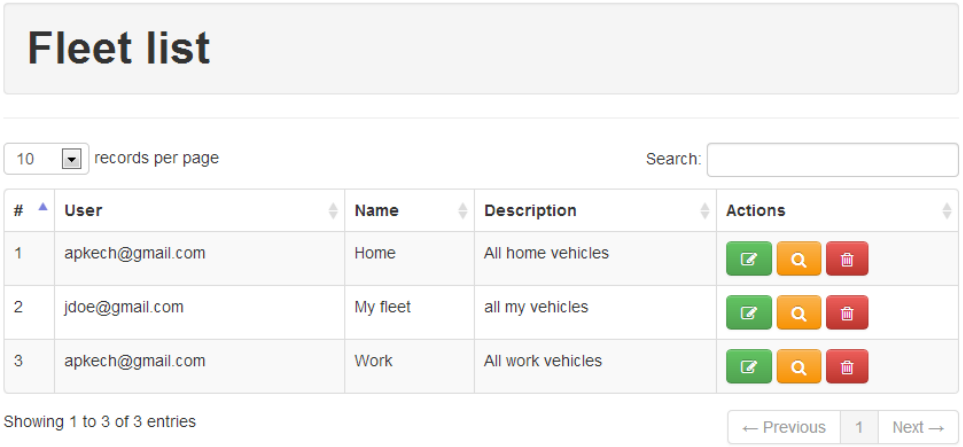
Η διαχείριση των οχημάτων στο περιβάλλον του διαχειριστή δεν διαφέρει πολύ από αυτή στο περιβάλλον του χρήστη. Η μόνη διαφορά είναι ότι έχει πρόσβαση σε όλα τα οχήματα του συστήματος.

### 3.2.2 Διαχείριση στόλων

Με αυτή την λειτουργία ο διαχειριστής μπορεί να εμφανίσει να επεξεργαστεί και να διαγράψει κάθε στόλο οχημάτων που υπάρχει στο σύστημα σε οποιονδήποτε χρήστη και αν ανήκει αυτός.




Σχήμα 3.22: Λίστα οχημάτων



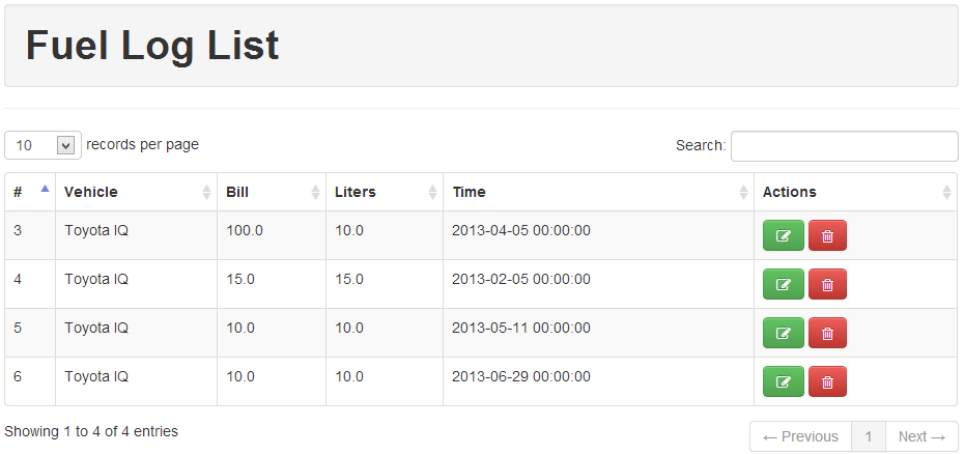

Σχήμα 3.23: Λίστα στόλων

### 3.2.3 Διαχείριση ανεφοδιασμών

Με αυτή την λειτουργία ο διαχειριστής έχει πρόσβαση στα αρχεία ανεφοδιασμών των χρηστών. Ο διαχειριστής μπορεί να επεξεργαστεί και να διαγράψει οποιαδήποτε εγγραφή.

### 3.2.4 Διαχείριση χρηστών

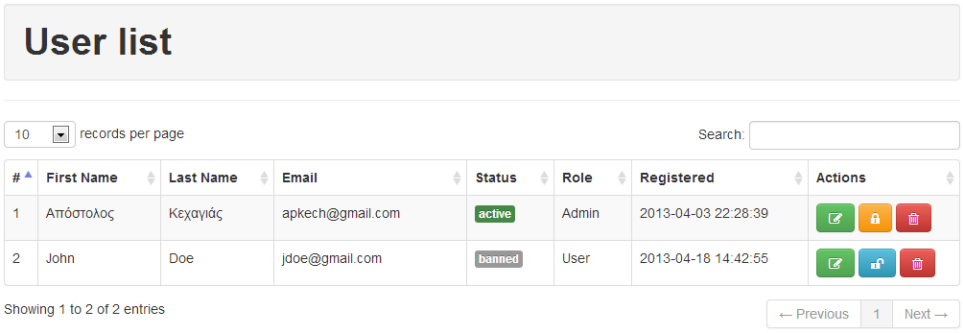
Ένας διαχειριστής μπορεί να επεξεργαστεί, να διαγράψει και να απενεργοποιήσει έναν χρήστη. Εκτός από τα βασικά δεδομένα τα οποία μπορεί να επεξεργαστεί (όνομα, επώνυμο, κωδικό πρόσβασης) υπάρχουν και άλλα δύο βασικά πεδία. Είναι ο ρόλος του χρήστη και η κατάσταση του λογαριασμού του. Αυτά τα δύο πεδία είναι επεξεργάσιμα μόνο από διαχειριστές.




Σχήμα 3.24: Λίστα ανεφοδιασμών

Ένας χρήστης μπορεί να είναι ενεργός η ανενεργός. Όλοι οι ενεργοί χρήστες χαρακτηρίζονται ως active ενώ οι ανενεργοί ως banned. Από την στιγμή που θα απενεργοποιηθεί ο λογαριασμός ενός χρήστη, αυτός δεν μπορεί να συνδεθεί στην εφαρμογή. Επίσης, τα δεδομένα που παραλαμβάνονται από την εφαρμογή για τυχόν ενεργά οχήματα, απορρίπτονται.

Οι διαθέσιμοι ρόλοι που μπορούν να αποδοθούν είναι αυτός του διαχειριστή και αυτός του χρήστη.



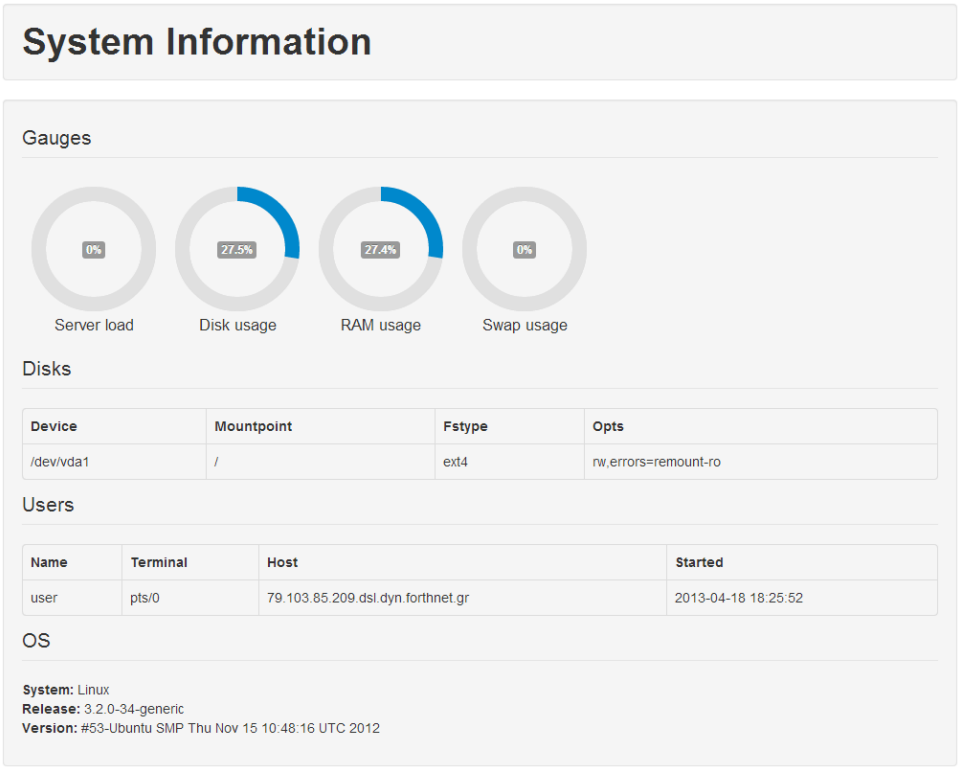

Σχήμα 3.25: Λίστα χρηστών

Όταν ένας χρήστης διαγραφεί, τότε διαγράφονται και όλα τα δεδομένα που έχουν παραχθεί από αυτόν (οχήματα, στόλοι, ειδοποιήσεις, κτλ).

### 3.2.5 Πληροφορίες συστήματος

Ο διαχειριστής πρέπει να είναι σε θέση να ελέγχει ανά πάσα στιγμή την κατάσταση του συστήματος. Είναι τέτοιος ο όγκος των δεδομένων αλλά και των αιτήσεων από οχήματα-συσκευές, που γρήγορα μπορούν να προκαλέσουν πρόβλημα στην εύρυθμη λειτουργία

της εφαρμογής. Έτσι προσφέρεται η δυνατότητα να παρακολουθούνται σε πραγματικό χρόνο, το ποσοστό χρήσης επεξεργαστικής ισχύος, η διαθέσιμη εικονική και πραγματική μνήμη και ο αποθηκευτικός χώρος. Συμπληρωματικά, προσφέρονται και πληροφορίες




Σχήμα 3.26: Πληροφορίες συστήματος

σχετικά με το λειτουργικό σύστημα, τους συνδεδεμένους χρήστες σε αυτό και λεπτομέρειες σχετικά με τις συσκευές αποθήκευσης.

## 3.3 Ασφάλεια

Η ανάπτυξη της εφαρμογής έγινε δίνοντας μεγάλη έμφαση στην ασφάλεια των δεδομένων. Έτσι όλα τα στοιχεία που εισάγονται από χρήστες, υποβάλλονται σε ελέγχους εγκυρότητας τόσο στο περιβάλλον του χρήστη, όσο και στο περιβάλλον του εξυπηρετητή. Ακόμη και αν κάποιος καταφέρει να παρακάμψει αυτούς τους ελέγχους, δεν θα μπορέσει να πετύχει κάτι, γιατί γίνεται καθαρισμός των ειδικών χαρακτήρων σε όλα τα δεδομένα που προέρχονται από φόρμες και χρησιμοποιούνται για ερωτήματα στην βάση δεδομένων. Αυτό σημαίνει ότι δεν μπορούν να παρερμηνευτούν με εξειδικευμένες τεχνικές, δεδομένα ως εντολές, μιας και τα πάντα αντιμετωπίζονται ως χαρακτήρες ή ομάδες χαρακτήρων.

Η εφαρμογή δίνει την δυνατότητα στον χρήστη να μην εισάγει κάθε φορά τον κωδικό

μέσω της επιλογής Remember me. Ένας απλός χρήστης θα μπορούσε να αλλάξει για παράδειγμα το id του χρήστη και να εισέλθει στο σύστημα ως διαχειριστής. Τέτοιου είδους επιθέσεις απορρίπτονται, γιατί ο εξυπηρετητής ελέγχει αν το cookie που χρησιμοποιείται για την συνεδρία είναι έγκυρο, συγκρίνοντας το υπάρχον με αυτό που θα δημιουργούσε ο ίδιος (hash). Μια άλλη επίθεση θα μπορούσε να συμβεί από κάποιον που παρακολουθεί την επικοινωνία ενός χρήστη με την εφαρμογή. Αυτός, θα μπορούσε να υποκλέψει το cookie που χρησιμοποιείται για την συγκεκριμένη συνεδρία και να εισέλθει στο σύστημα με τον λογαριασμό του θύματος. Αυτό το πρόβλημα λύθηκε χρησιμοποιώντας ασφαλή σύνδεση σε όλη την δομή της εφαρμογής.

Με τον ίδιο ακριβώς τρόπο διασφαλίστηκε και η παραλαβή δεδομένων. Η διαδικασία της παραλαβής των δεδομένων πρέπει να είναι επίσης ασφαλής καθώς μεταφέρεται η τρέχουσα θέση ενός οχήματος αλλά και το κλειδί της συσκευής. Η υποκλοπή του κλειδιού θα μπορούσε να δημιουργήσει πρόβλημα, γιατί ο κάτοχός του μπορεί να το χρησιμοποιήσει για να εισάγει δεδομένα.

Το ανθρώπινο λάθος έχει επίσης προβλεφθεί. Τα στοιχεία του χρήστη δεν μπορούν να αλλάξουν αν δεν χρησιμοποιηθεί ο ισχύον κωδικός πρόσβασης. Αυτό κάνει βέβαιο ότι κάποιος που βρέθηκε με κάποιον τρόπο στον λογαριασμό ενός χρήστη, δεν μπορεί να αλλάξει κωδικό πρόσβασης ούτε και email. Μια αλλαγή στο email μαζί με μια αλλαγή κωδικού, θα έκανε αδύνατη την ανάκτηση του λογαριασμού.

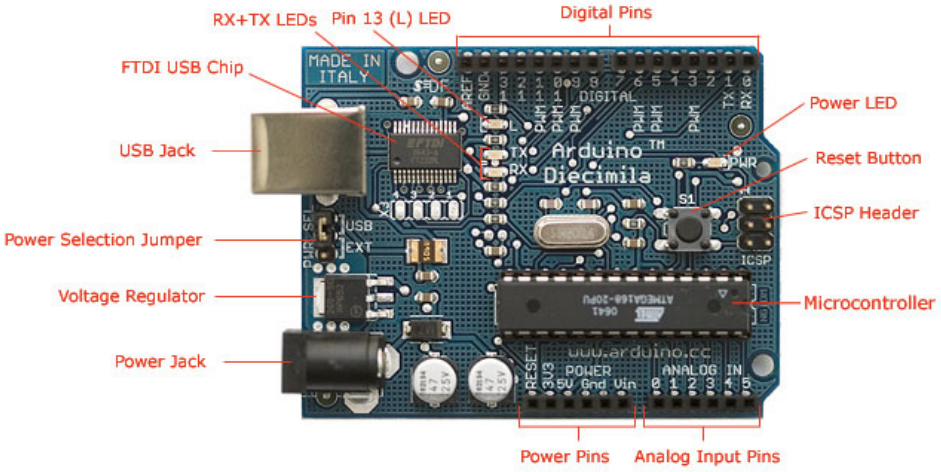
# Κεφάλαιο 4

# Οι συσκευές καταγραφής

Σε αυτό το κεφάλαιο παρουσιάζονται οι συσκευές καταγραφής που χρησιμοποιήθηκαν στα οχήματα για την καταγραφή των δεδομένων. Περιγράφονται τα χαρακτηριστικά τους, ο προγραμματισμός τους και ο τρόπος λειτουργίας τους.

## 4.1 Συσκευή βασισμένη στο Arduino

Η συσκευή που είναι βασισμένη στο Arduino χρησιμοποιεί μια αναπτυξιακή πλακέτα Arduino Diecimila, ένα GPRS και ένα GPS shield. Η επιλογή της πλατφόρμας Arduino έγινε, γιατί δίνει την δυνατότητα εύκολης και γρήγορης ανάπτυξης ενός συστήματος, κυρίως λόγω των βιβλιοθηκών και των shields.




Σχήμα 4.1: Arduino Diecimila

| | |
|---|---|
| Μικροεπεξεργαστής | ATmega168 |
| Τάση λειτουργίας | 5V |
| Τάση εισόδου | 7-12 V |
| Τάση εισόδου | 6-20 V |
| Ψηφιακά I/O Pins | 14 (6 provide PWM output) |
| Αναλογικά Pins εισόδου | 6 |
| DC ένταση για κάθε I/O Pin | 40 mA |
| DC ένταση για κάθε 3.3V Pin | 50 mA |
| Flash Memory | 16 KB (2 KB bootloader) |
| SRAM | 1 KB |
| EEPROM | 512 bytes |
| Ταχύτητα ρολογιού | 16 MHz |

Πίνακας 4.1: Χαρακτηριστικά του Arduino Diecimila

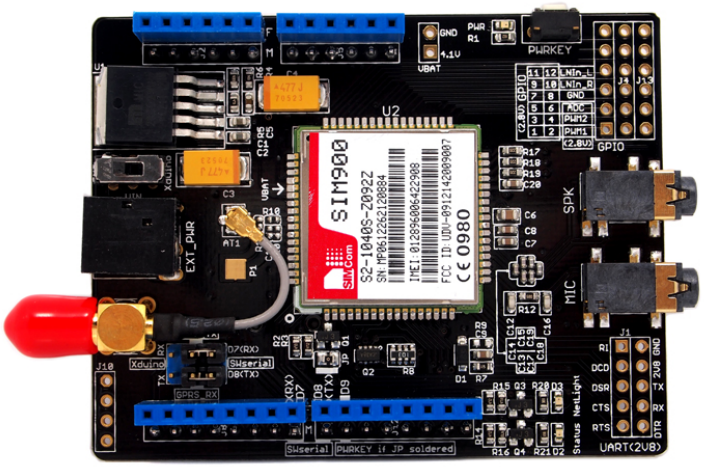
**GPRS shield**

Η συσκευή που επιλέχθηκε για την επικοινωνία των οχημάτων με τον εξυπηρετητή, είναι το Seeedstudio GPRS Shield. Στην συγκεκριμένη εφαρμογή μπορεί να χρησιμοποιείται μόνο το GPRS, αλλά υπάρχει η δυνατότητα και για κλήσεις, μηνυμάτα SMS και FAX. Η επιλογή του συγκεκριμένου shield έγινε γιατί:

α) βασίζεται στο chip SIM900, το οποίο και υπόσχεται εξαιρετικά χαμηλή κατανάλωση ενέργειας

β) διατίθεται μαζί με αυτό κεραία

γ) είναι συμβατό με την πλατφόρμα Arduino

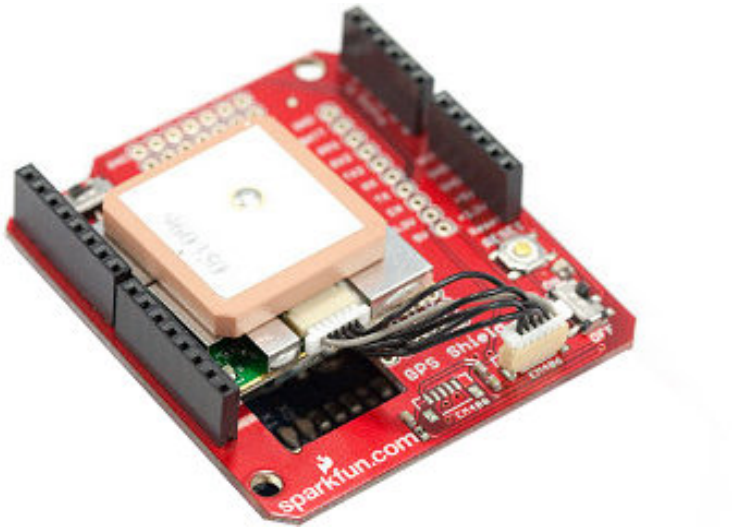



Σχήμα 4.2: GPRS shield

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Συμβατό με το GSM phase 2/2+
- Class 4 (2 W @850/ 900 MHz)
- Class 1 (1 W @ 1800/1900MHz)
- Λειτουργεί με AT εντολές (GSM 07.07 ,07.05 και SIMCOM)
- Χαμηλή κατανάλωση: 1.5mA
- Θερμοκρασία λειτουργίας: από -40°C μέχρι +85 °C

**GPS shield**

Η επιλογή του GPS shield έγινε με παρόμοια κριτήρια. Την συμβατότητα δηλαδή με την πλατφόρμα Arduino. Ένας ακόμη λόγος που επιλέχθηκε, είναι ότι δίνει ένα μικρό χώρο για δημιουργία κάποιου κυκλώματος. Έτσι θα μπορούσαν να προστεθούν εύκολα LED, που θα ενημερώνουν τον χρήστη για την κατάσταση λειτουργίας της συσκευής.




Σχήμα 4.3: GPS shield

- Υποδοχή EM-406
- UP501 chip
- Περιοχή προτυποποίησης

(α΄) (β΄)

Σχήμα 4.4: Συσκευή βασισμένη στο Arduino

- Κουμπί επανεκκίνησης
- Διακόπτης εναλλαγής λειτουργίας DLINE/UART
- ON/OFF διακόπτης

Επίσης, χρησιμοποιήθηκε ένα τροφοδοτικό αυτοκινήτου με έξοδο 12V 2A και τρία ενημερωτικά LED, για την λειτουργία της συσκευής. Τα shields τοποθετήθηκαν το ένα πάνω στο άλλο στο Arduino και όλα μαζί μπήκαν σε ένα κουτί. Τέλος προστέθηκε και ένας διακόπτης για την ενεργοποίηση και απενεργοποίηση της συσκευής. Το ολοκληρωμένο αποτέλεσμα φαίνεται στο Σχήμα 4.4.

### 4.1.1 Προγραμματισμός

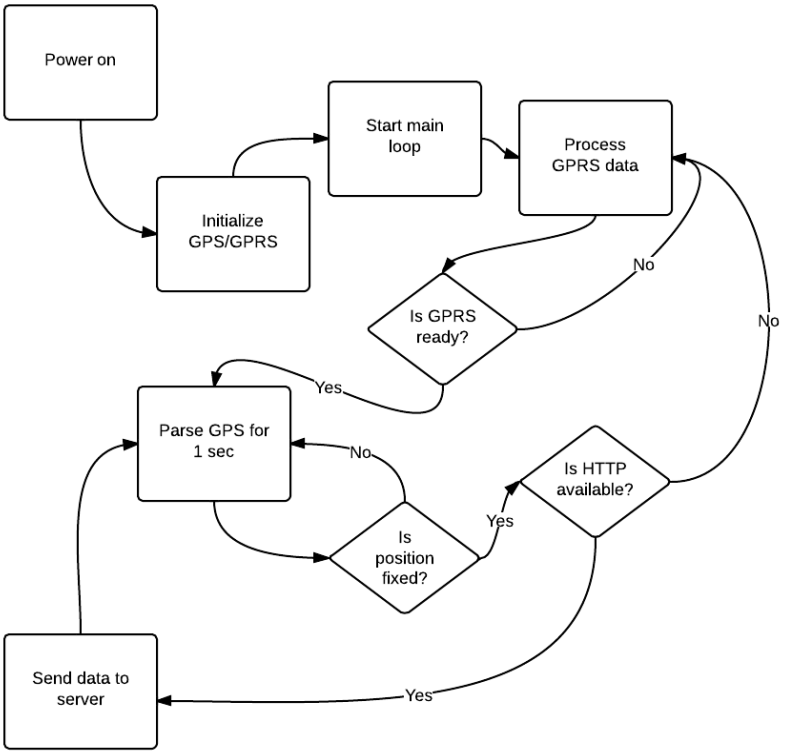
Ο προγραμματισμός της συσκευής χωρίζεται στο τμήμα που αφορά την αρχικοποίηση και το τμήμα που επαναλαμβάνεται συνεχώς μόλις αυτό ολοκληρωθεί.

Στο τμήμα της αρχικοποίησης ξεκινά η λειτουργία των δύο υποσυστημάτων GPS, GPRS. Αμέσως μετά εκτελείται το κυρίως πρόγραμμα. Εκεί γίνεται έλεγχος για το αν υπάρχει συνδεσιμότητα με το δίκτυο και δυνατότητα μεταφοράς δεδομένων. Εφόσον υπάρχει δυνατότητα μεταφοράς δεδομένων, ξεκλειδώνει το τμήμα του κώδικα που αφορά τις πληροφορίες θέσης. Αν η συσκευή έχει βρει με επιτυχία μια έγκυρη θέση, τότε τα δεδομένα αποστέλλονται στον εξυπηρετητή.

Η αρχικοποίηση των υποσυστημάτων πραγματοποιείται ως εξής:

```
1 Serial.begin(4800);
2 cell.begin(19200);
3 powerUpOrDown();
```

Το GPS shield συνδέεται χρησιμοποιώντας την μοναδική σειριακή οδό που έχει το Arduino Diecimila. Για το GPRS shield χρησιμοποιείται μέθοδος που βασίζεται σε λογισμικό. Η ταχύτητα που χρησιμοποιήθηκε για το GPS shield είναι η μέγιστη δυνατή. Το

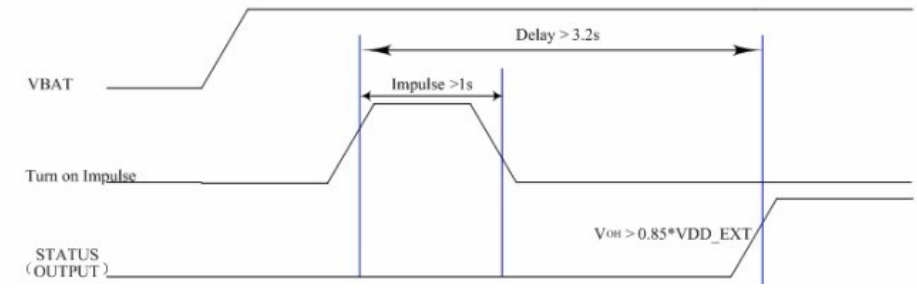



Σχήμα 4.5: Διάγραμμα ροής Arduino

GPRS shield υποστηρίζει και υψηλότερους ρυθμούς δεδομένων, όμως πρέπει ενημερωθεί αναλόγως το υλικό. Αυτό μπορεί να γίνει χρησιμοποιώντας την εντολή "AT+IPR=X" όπου X είναι ο ρυθμός baud προς ορισμό.

Η συνάρτηση powerUpOrDown χρησιμοποιείται για την ενεργοποίηση του GPRS shield. Η ενεργοποίηση δεν γίνεται αυτόματα με την παροχή τάσης. Μπορεί να γίνει, είτε χρησιμοποιώντας το κουμπί που βρίσκεται επάνω στην πλακέτα, είτε χρησιμοποιώντας λογισμικό. Προφανώς, είναι περισσότερο λειτουργικό, να ενεργοποιείται η συσκευή αυτόματα, χωρίς περαιτέρω ενέργειες από τον χρήστη.

Ο οδηγός [12] (Σχήμα - 4.6) της πλακέτας αναφέρει ότι ένας παλμός μεγαλύτερος του ενός δευτερολέπτου αρκεί, ώστε να πραγματοποιήσει την ενεργοποίηση ή απενεργοποίηση, ενώ χρειάζεται και μια καθυστέρηση μεγαλύτερη από 3.2 δευτερόλεπτα, για να μην υπάρχουν προβλήματα χρονισμού. Η ίδια διαδικασία γίνεται και για την απενεργοποίηση. Δηλαδή, η συσκευή πηγαίνει στην αντίθετη κατάσταση από αυτή που βρισκόταν πριν την εφαρμογή του παλμού.




Σχήμα 4.6: Ενεργοποίηση/απενεργοποίηση GPRS shield

```
void powerUpOrDown(){
  pinMode(9, OUTPUT);

  digitalWrite(9,LOW);
  delay(1000);

  digitalWrite(9,HIGH);
  delay(2000);

  digitalWrite(9,LOW);
  delay(3000);
}
```

Το κυρίως μέρος της εφαρμογής εκτελείται αμέσως μετά την αρχικοποίηση. Ωστόσο, δεν γίνεται καμία παραπάνω ενέργεια, εφόσον δεν υπάρχει καλό σήμα και σύνδεση δεδομένων. Αυτό γίνεται χρησιμοποιώντας σημαίες, οι οποίες ενεργοποιούνται όταν πληρούνται τα κριτήρια που έχουν τεθεί.

```
if(firstLoop){
    firstLoop = 0;

    while(GPRS_SHIELD_READY == 0){
        readATString();
        processATString();
    }

    while(GPRS_HTTP_READY == 0){
        setupHTTP();
    }
}
```

Υπάρχει ένας έλεγχος για την πρώτη εκτέλεση αυτού του τμήματος. Εφόσον ισχύει, διαβάζονται δεδομένα από το GPRS shield χρησιμοποιώντας την συνάρτηση readATString. Όλα τα δεδομένα αποθηκεύονται σε έναν πίνακα και έπειτα ερμηνεύονται χρησι-

μοποιώντας την συνάρτηση processATString. Όταν επιστραφεί το αλφαριθμητικό "Call Ready", η συσκευή είναι έτοιμη και η σημαία GPRS_SHIELD_READY τίθεται σε κατάσταση 1.

```
void setupHTTP()
{
  cell.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");
  delay(1000);

  cell.println("AT+SAPBR=3,1,\"APN\",\"internet\"");
  delay(4000);

  cell.println("AT+SAPBR=1,1");
  delay(2000);

  cell.println("AT+HTTPSSL=1");
  delay(2000);

  cell.println("AT+HTTPINIT");
  delay(2000);

  GPRS_HTTP_READY = 1;
}
```

Η σημαία GPRS_HTTP_READY τίθεται σε κατάσταση 1 μετά την αρχικοποίηση της σύνδεσης μέσω HTTP. Όταν και οι δύο σημαίες είναι σε κατάσταση 1, τότε γίνεται προσπάθεια προσδιορισμού της θέσης. Αυτό το στάδιο κρατάει ένα δευτερόλεπτο.

```
for(long start = millis(); millis() - start < 1000;){
    while (Serial.available()){
        char c = Serial.read();

        if (gps.encode(c))
        dataReady = true;
    }
}
```

Αν σε αυτό το διάστημα υπάρχουν έγκυρα δεδομένα, τότε στέλνονται χρησιμοποιώντας την συνάρτηση sendDataToServer.

```
if (dataReady){
    float flat, flon, fspd, falt;
    unsigned long age;
    gps.f_get_position(&flat, &flon, &age);
    fspd = gps.f_speed_kmph();
    falt = gps.f_altitude();

    tmElements_t tm;
    int year;
    gps.crack_datetime(&year, &tm.Month, &tm.Day, &tm.Hour,
                       &tm.Minute, &tm.Second, NULL, NULL);
    tm.Year = year - 1970;
    time_t time = makeTime(tm);
    time = time + (offset * SECS_PER_HOUR);

    sendDataToServer(flat, flon, fspd, falt, time);
}
```

Για την ενημέρωση του χρήστη σχετικά με την λειτουργία της συσκευής, αποφασίστηκε να προστεθούν κάποια LEDs. Συγκεκριμένα τοποθετήθηκαν τρία LED. Το πρώτο χρησιμοποιείται για την κατάσταση της συσκευής και είναι αναμμένο όταν αυτή λειτουργεί.

```
void sendDataToServer(float lat, float lng, float spd, float alt, time_t tim)
{
  cell.print("AT+HTTPPARA=\"URL\"");
  cell.print(",\"localhost/upload?key=0&lat=");
  cell.print(lat, 8);
  cell.print("&lng=");
  cell.print(lng, 8);
  cell.print("&alt=");
  cell.print(alt, 8);
  cell.print("&spd=");
  cell.print(spd, 8);
  cell.print("&tim=");
  cell.print(tim);
  cell.println("\"");
  delay(1000);

  cell.println("AT+HTTPACTION=0");
  delay(2000);
}
```

Όταν είναι σβηστό, η συσκευή δεν λειτουργεί. Το δεύτερο χρησιμοποιείται για την ένδειξη της αποστολής δεδομένων. Κάθε φορά που στέλνονται δεδομένα αναβοσβήνει. Όταν

είναι δηλαδή σβηστό για μεγάλο χρονικό διάστημα σημαίνει ότι η σύνδεση παρουσιάζει πρόβλημα. Το τρίτο LED ενημερώνει τον χρήστη για τα δεδομένα θέσης. Κάθε φορά που αναβοσβήνει, σημαίνει ότι έχει εντοπιστεί μια νέα θέση.

## 4.2 Συσκευή βασισμένη σε smartphone

Η χρήση των έξυπνων κινητών τηλεφώνων έχει αυξηθεί κατακόρυφα τα τελευταία χρόνια. Το μερίδιο αγοράς τους συνεχώς αυξάνεται και προβλέπεται ότι σε λίγα χρόνια όλοι θα έχουν από ένα. Έτσι πάρθηκε η απόφαση να αναπτυχθεί μια εφαρμογή, η οποία να κάνει χρήση όλων των τεχνολογιών που υπάρχουν ενσωματωμένες σε μια τέτοια συσκευή. Αυτό έγινε κυρίως για δύο λόγους. Ο πρώτος λόγος είναι οικονομικός. Δηλαδή κάποιος που έχει μια τέτοια συσκευή, μπορεί απλά με την εγκατάσταση μιας εφαρμογής, να κάνει χρήση της υπηρεσίας. Ο δεύτερος λόγος είναι η υποστήριξη πολλών τρόπων πρόσβασης στην υπηρεσία.

Το έξυπνο κινητό τηλέφωνο που χρησιμοποιήθηκε είναι το LG E900. Χρησιμοποιεί το λειτουργικό σύστημα Windows Phone 7 και συγκεκριμένα την έκδοση 7.8.


Σχήμα 4.7: LG E900

Τα χαρακτηριστικά της συσκευής περιγράφονται αναλυτικά παρακάτω.

- Λειτουργικό: Windows Phone 7

- Δίκτυα: GSM Quad Band (850/900/1800/1900) / HSDPA
- Τύπος οθόνης: Capacitive WVGA LCD 3,8 ιντσών
- Ανάλυση οθόνης: 800×480 pixels
- Κάμερα: 5 Megapixels LED Flash με Auto Focus
- Μνήμη: Εσωτερική 16GB
- Αναπαραγωγή βίντεο: H.263, H.264, MPEG4, MPEG4@HD 720p 30fps
- Εγγραφή βίντεο: H.263, H.264, MPEG4, MPEG4@HD 720p 24fps
- Αναπαραγωγή αρχείων μουσικής: MP3, AAC, AAC+, e AAC+, WMA pro, WAV, AMR-NB, MIDI
- Συνδεσιμότητα: WiFi 802.11b/g/n / Bluetooth 2.1 EDR / USB 2.0
- Stereo FM Radio
- Αισθητήρες: Accelerometer / Proximity / Ambient Light Sensor / Digital Compass
- SMS / MMS / Email (IMAP 4, SMTP, POP3) / SNS
- Τύπος μπαταρίας: LiOn 1500mAh
- Διαστάσεις (χιλ.): 125×59,8×11,5
- Βάρος: 157 γρ.

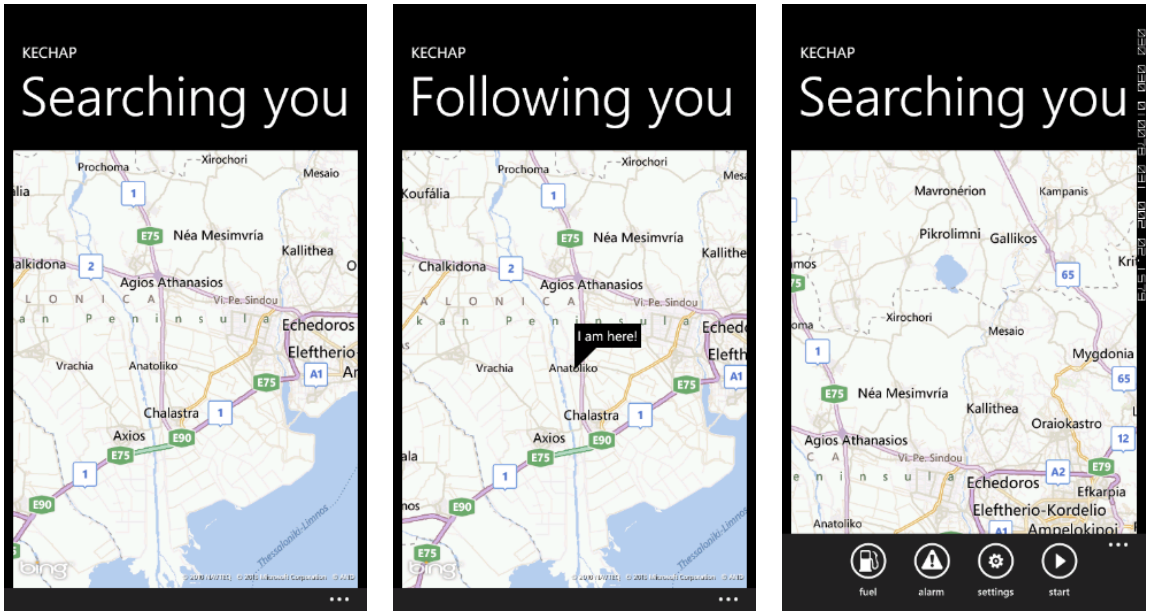
### 4.2.1 Προγραμματισμός

Η διαφορά στην λειτουργία των συσκευών έφερε και αλλαγές στην υλοποίηση. Το κινητό τηλέφωνο έχει την δυνατότητα απεικόνισης, οπότε κρίθηκε σκόπιμο να προβάλλεται και η θέση του χρήστη εκτός από την απλή αποστολή δεδομένων.

Η φιλοσοφία ανάπτυξης σε πλατφόρμα WP επικεντρώνεται στο UI. Έτσι, αρχικά σχεδιάστηκε το περιβάλλον του χρήστη και έπειτα προγραμματίστηκε η λειτουργία του. Το περιβάλλον περιλαμβάνει τρεις οθόνες. Την βασική οθόνη παρακολούθησης του οχήματος, την οθόνη ρυθμίσεων και την οθόνη προσθήκης ανεφοδιασμού.

Στην αρχική οθόνη υπάρχει ένας χάρτης, στον οποίο εμφανίζεται ένα σύμβολο που υποδηλώνει την τρέχουσα θέση του χρήστη. Εφόσον υπάρχει αυτό το σύμβολο, στο πάνω μέρος της εφαρμογής εμφανίζεται η ένδειξη "Following you". Όταν δεν έχει βρεθεί το στίγμα του χρήστη, εμφανίζεται η ένδειξη "Searching you". Στο κάτω μέρος της οθόνης υπάρχει αναδυόμενο μενού με τις επιλογές για πρόσβαση στις υπόλοιπες επιλογές (Σχήμα 4.8) αλλά και την εκκίνηση της αποστολής δεδομένων.

Στην οθόνη των ρυθμίσεων υπάρχουν όλες οι απαραίτητες ρυθμίσεις για την λειτουργία της εφαρμογής. Μπορεί να οριστεί το κλειδί της συσκευής, η ακρίβεια του στίγματος




(α') Αναζήτηση (β') Παρακολούθηση (γ') Μενού

Σχήμα 4.8: Αρχική οθόνη εφαρμογής

και συχνότητα αποστολής των δεδομένων σε σχέση με τον χρόνο αλλά και την απόσταση που διανύθηκε.

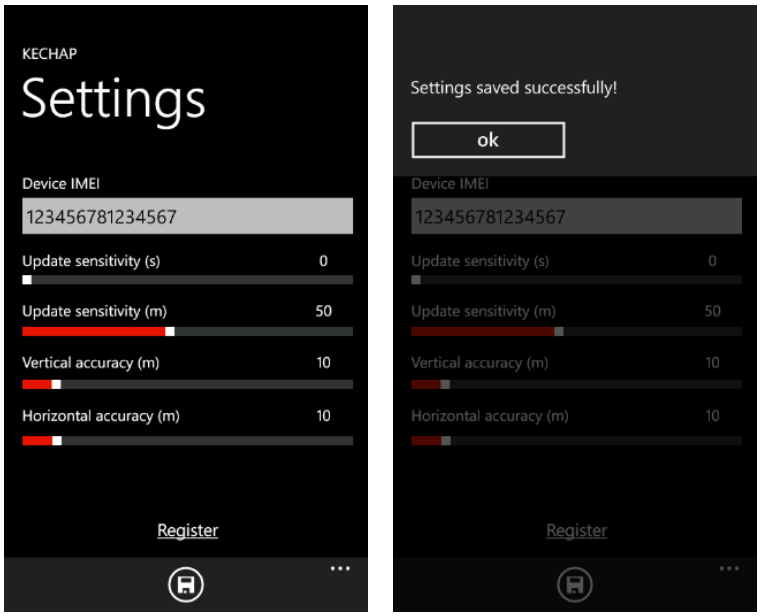
Οι επιλογές του χρήστη για την εφαρμογή αποθηκεύονται σε έναν ειδικό χώρο που μπορεί να δεσμευτεί από κάθε εφαρμογή.

```
1 settings = IsolatedStorageSettings.ApplicationSettings;
2 settings["key"] = keyTextBox.Text;
```

Αυτός ο χώρος είναι διαθέσιμος μόνο από την συγκεκριμένη εφαρμογή και παραμένει ίδιος, ακόμη και μετά από ενημερώσεις που μπορεί να γίνουν στην εφαρμογή.

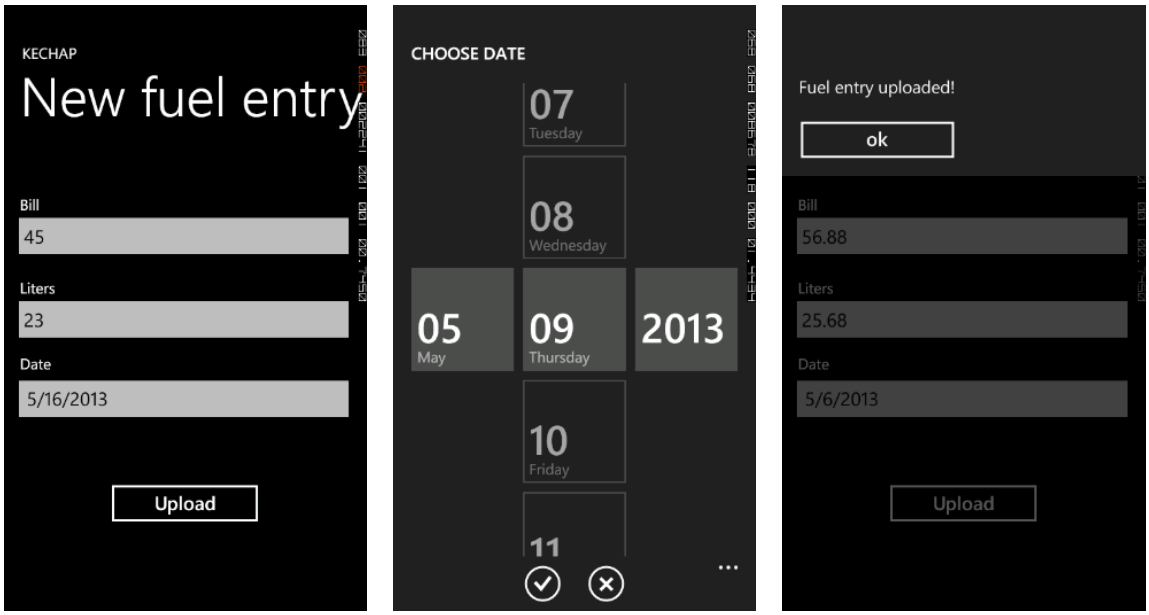
Η οθόνη προσθήκης ανεφοδιασμού είναι ουσιαστικά μια φόρμα που περιέχει τα στοιχεία του ανεφοδιασμού. Δηλαδή το ποσό που δαπανήθηκε, τα λίτρα καυσίμου που αγοράστηκαν και τη χρονική στιγμή που πραγματοποιήθηκε.

Η αποστολή των δεδομένων πραγματοποιείται με αίτηση GET, παρέχοντας όλες της πληροφορίες του χρήστη ως παραμέτρους. Η επιτυχής καταχώρηση εξακριβώνεται μέσω του πεδίου Entry-Added, που εισάγεται στην επικεφαλίδα της απάντησης του εξυπηρετητή.




(α΄) Φόρμα επιλογών (β΄) Επιβεβαίωση

Σχήμα 4.9: Επιλογές εφαρμογής




(α΄) Φόρμα (β΄) Επιλογή ημερομηνίας (γ΄) Επιβεβαίωση προσθήκης

Σχήμα 4.10: Προσθήκη ανεφοδιασμού

```
wc.DownloadStringCompleted += (ss, ee) => {
    bool found = false;
    if (!ee.Cancelled && ee.Error == null)
    {
        if (new List<string>(wc.ResponseHeaders.AllKeys).
            Contains("Entry-Added"))
        {
            found = true;
            MessageBox.Show("Fuel entry uploaded!");
        }
    }
    if (!found)
    {
        MessageBox.Show("An error occured! Try again.");
    }
};
```

Ο προσδιορισμός της θέσης του οχήματος γίνεται χρησιμοποιώντας την κλάση GeoCoordinateWatcher [9].

```
gw = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
```

Η παράμετρος GeoPositionAccuracy.High είναι απαραίτητη γιατί με αυτό τον τρόπο χρησιμοποιείται μόνο ο δέκτης GPS. Διαφορετικά, μπορεί να χρησιμοποιηθεί η μέθοδος προσδιορισμού θέσης μέσω διαδικτύου που είναι εξαιρετικά ανακριβής. Η συγκεκριμένη κλάση απαιτεί την υλοποίηση του event PositionChanged.

```
gw.PositionChanged += (s, e) =>
{
  if((DateTime.Now - e.Position.Timestamp.DateTime) < TimeSpan.FromSeconds(20)
    && e.Position.Location.VerticalAccuracy <= verAccuracy
    && e.Position.Location.HorizontalAccuracy <= horAccuracy && isTimerFired)
    {
        if (e.Position.Location.IsUnknown != true)
        {
            if (NetworkInterface.GetIsNetworkAvailable())
            {
                SendGeoDataToServer(e.Position.Location,
                e.Position.Timestamp.DateTime);
                isTimerFired = false;
            }
        }
    }
};
```

Έτσι κάθε φορά που υπάρχει αλλαγή θέσης, ενεργοποιείται το συγκεκριμένο τμήμα

κώδικα. Κάθε φορά εξετάζεται αν το στίγμα είναι σχετικά πρόσφατο. Αυτός ο έλεγχος απαιτείται, γιατί με τις δοκιμές που πραγματοποιήθηκαν, διαπιστώθηκε ότι το λειτουργικό έχει την τάση να κρατά παλιές τιμές και να τις προωθεί ως αποτέλεσμα σε αυτό το event. Επίσης, ελέγχονται οι περιορισμοί σχετικά με την ακρίβεια και την συχνότητα αποστολής δεδομένων, που έχουν οριστεί στις ρυθμίσεις της εφαρμογής. Εφόσον ικανοποιούνται όλες οι συνθήκες, γίνεται έλεγχος για συνδεσιμότητα στο διαδίκτυο και αποστέλλονται τα δεδομένα στον εξυπηρετητή.

```
public void SendGeoDataToServer(GeoCoordinate location, DateTime time)
{
    DateTime unixEpoch = new DateTime(1970, 1, 1);
    DateTime currentDate = time;
    long totalMS = (currentDate.Ticks-unixEpoch.Ticks)/10000;

    WebClient wc = new WebClient();
    string url = web_service_url
          + "?key=" + key
          + "&lat=" + location.Latitude.ToString().Replace(",", ".")
          + "&lng=" + location.Longitude.ToString().Replace(",", ".")
          + "&alt=" + location.Altitude.ToString().Replace(",", ".")
          + "&spd=" + (location.Speed*3.6).ToString().Replace(",", ".")
          + "&tim=" + totalMS;
    if (alarm)
    {
        url += "&alm=1";
    }

    wc.DownloadStringAsync(new Uri(url));
    wc.DownloadStringCompleted += (ss, ee) => {
        if (alarm)
        {
            MessageBox.Show("Alarm sent!");
            alarm = false;
        }
    };
}
```

Στην συνάρτηση αυτή γίνεται ο υπολογισμός του χρόνου συλλογής των δεδομένων. Ο χρόνος μετριέται από την αρχή που χρησιμοποιούν οι υπολογιστές, δηλαδή την 1/1/1970 και υπολογίζεται η διαφορά σε millisecond από την τρέχουσα ημερομηνία. Έπειτα, δημιουργείται μια ασύγχρονη αίτηση GET στον εξυπηρετητή με όλα τα δεδομένα. Η ίδια συνάρτηση χρησιμοποιείται και για την αποστολή ειδοποίησης. Όταν πατηθεί το κουμπί 'alarm', καταγράφεται η τελευταία γνωστή θέση του οχήματος, τίθεται η σημαία alarm σε αληθή κατάσταση και εκτελείται η συνάρτηση.

| | |
|---|---|
| Διαστάσεις | 2.10"(L)x1.51"(W)x0.56"(H) |
| Βάρος | 50g |
| Δίκτυο | GSM/GPRS |
| Band | 850/900/1800/1900Mhz |
| GSM ευαισθησία | <-102dBm |
| GPS chipset | SirF III chipset |
| GPS ευαισθησία | -159dBm |
| GPS ακρίβεια | 5m |
| Εκκίνηση | 35s(cold)/2s(hot) |
| CPU | ARM |
| Μπαταρία | 3.7V DC Li-Ion Battery,860mA |
| Αυτονομία | 48 ώρες |
| Θερμοκρασία λειτουργίας | από -20°C μέχρι +55°C |

Πίνακας 4.2: Χαρακτηριστικά του TK106

## 4.3 Συσκευή εμπορίου

Κατά την διάρκεια των δοκιμών της εφαρμογής βρέθηκε διαθέσιμη η συσκευή TK106. Έτσι αναπτύχθηκε το κατάλληλο λογισμικό, ώστε να λειτουργήσει με την εφαρμογή.

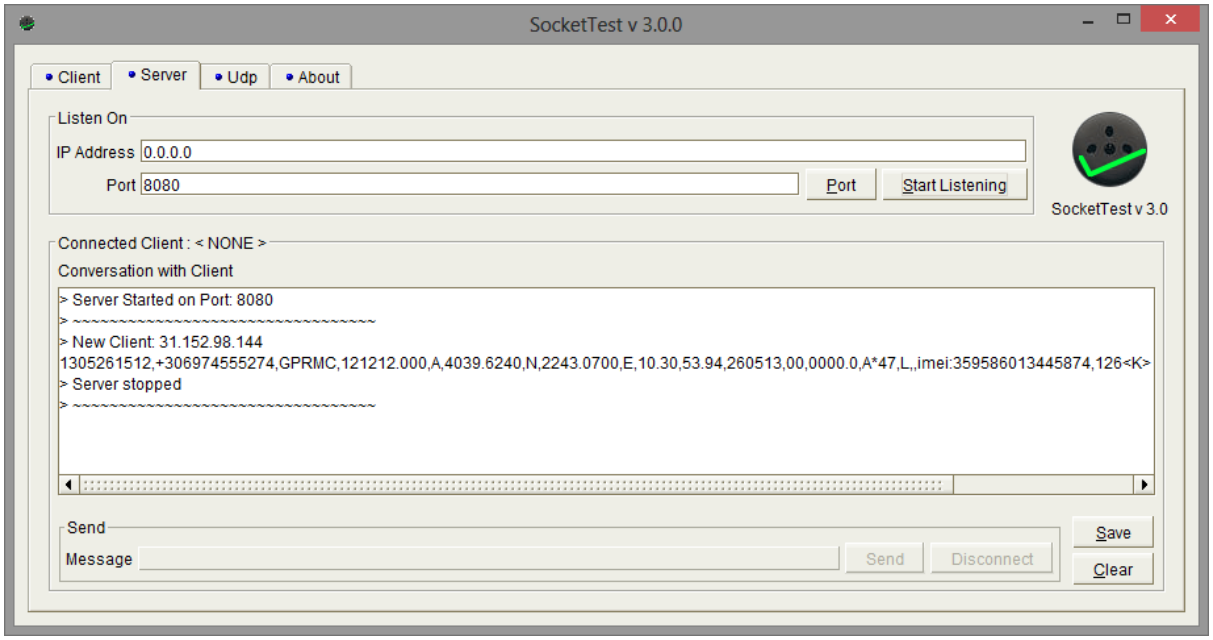

Σχήμα 4.11: TK106

### 4.3.1 Προγραμματισμός

Η συσκευή λειτουργεί προεπιλεγμένα χρησιμοποιώντας SMS. Μπορεί όμως να γίνει παραμετροποίηση, ώστε να λειτουργεί χρησιμοποιώντας GPRS. Η διαδικασία πραγματοποιείται στέλνοντας μηνύματα SMS στην συσκευή.

- "begin123456" - αρχικοποιεί την συσκευή, ο κωδικός είναι εξ αρχής 123456
- "apn123456 internet" - ορίζει το APN σε internet
- "GPRS123456" - ενεργοποίηση υπηρεσίας GPRS
- "adminip123456 <IP> 8080" - ορίζεται η IP διεύθυνση και η θύρα στην οποία θα στέλνονται τα δεδομένα
- "t010s***n123456" - ορίζεται η αποστολή δεδομένων κάθε 10 δευτερόλεπτα

Έπειτα από αναζήτηση στο διαδίκτυο, βρέθηκε ότι η συγκεκριμένη συσκευή χρησιμοποιεί tcp socket για την αποστολή των δεδομένων. Για να γίνει γνωστή η μορφή τους, χρησιμοποιήθηκε το πρόγραμμα SocketTest.




Σχήμα 4.12: SocketTest

Η μορφή των πακέτων ακολουθεί συγκεκριμένο πρότυπο και πλέον μπορεί να προγραμματιστεί η παραλαβή και αποθήκευσή τους.

**<serial datetime local>, <admin cell number>, GPRMC, <time utc>, <status> <lat>, <N or S>, <lng>, <E or W>, <speed>, <course>, <date utc>, <not used>, <not used>, <mode - A, D, E, N, S - with checksum>, <signal quality F or L>, <alarm>, <IMEI>, <firmware info>**

Η εφαρμογή που αναπτύχθηκε, χρησιμοποιεί web service για την παραλαβή των δεδομένων. Γι' αυτό τον λόγο, χρειάστηκε η υλοποίηση ενός socket server, που θα πραγματοποιεί την ίδια εργασία για την συγκεκριμένη συσκευή.

```
class TkServer(protocol.Protocol):
    def connectionMade(self):
        print "Connected from", self.transport.client

    def connectionLost(self, reason):
        print "Disconnected from", self.transport.client

    def dataReceived(self, data):
        pass

class TkServerFactory(protocol.Factory):
    protocol = TkServer

if __name__ == "__main__":
    factory = TkServerFactory()
    reactor.listenTCP(8080, factory)
    reactor.run()
```

Τα δεδομένα παραλαμβάνονται με την συνάρτηση dataReceived. Επειδή η μορφή τους δεν είναι ίδια με αυτή που χρησιμοποιείται από την εφαρμογή, απαιτείται η μετατροπή τους.

```
def dataReceived(self, data):
    data = data.split(',')
    tim = int(datetime.combine(datetime.strptime(data[11], '%d%m%y').date(),
                           datetime.strptime(data[3], '%H%M%S.%f').time()).
                           strftime("%s")) * 1000
    lat = int(data[5][:2]) + (float(data[5][2:]) / 60.0)
    if data[6] == 'S':
        lat = - lat
    lng = int(data[7][:2]) + (float(data[7][2:]) / 60.0)
    if data[8] == 'W':
        lng = - lng

    spd = float(data[9]) * 1.852
    alt = 0
    alm = data[16]

    key = data[17][5:]

    if len(key) == 15:
       device = Device.query.filter(Device.key == key).first()
       if device:
           d = Data(device.id, lat, lng, spd, alt, tim)
```

Μετατρέπονται οι συντεταγμένες σε μορφή αναγνωρίσιμη από τους χάρτες, η ημερομηνία σε αντικείμενο datetime και η ταχύτητα από κόμβους σε χιλιόμετρα. Έπειτα από δοκιμές, βρέθηκε ότι η συσκευή στέλνει το μήνυμα 'help me' όταν πατηθεί το κουμπί SOS. Για αυτό τον λόγο δόθηκε η ίδια λειτουργικότητα με το κουμπί ειδοποίησης της εφαρμογής WP.

# Κεφάλαιο 5

# Τεχνολογίες που χρησιμοποιήθηκαν

Στα δύο προηγούμενα κεφάλαιο παρουσιάστηκε η εφαρμογή και οι συσκευές καταγραφής. Αντικείμενο του κεφαλαίου αυτού είναι η παρουσίαση και η περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξή τους. Οι πληροφορίες αφορούν τόσο το λογισμικό όσο και το υλικό.

## 5.1 Python

Η Python [24] είναι μια αντικειμενοστραφής διερμηνευόμενη γλώσσα γενικού σκοπού. Είναι γλώσσα υψηλού επιπέδου και η δημιουργία της βασίστηκε στην αναγνωσιμότητα του κώδικα. Αυτό που γίνεται άμεσα αντιληπτό από κάποιον που προγραμματίζει σε αυτή, είναι η εκφραστικότητά της. Δηλαδή, η δυνατότητα που έχει κάποιος να επιτυγχάνει τους στόχους που έχει θέσει, με το μικρότερο δυνατό κόστος, από την πλευρά της πολυπλοκότητας και της έκτασης του παραγόμενου κώδικα.

Έχει μια μεγάλη κύρια βιβλιοθήκη που καλύπτει ένα μεγάλο εύρος πεδίων, κάνοντάς την ιδανική για χρήση σχεδόν σε οποιοδήποτε εφαρμογή. Είναι εύκολη στην εκμάθηση, στην αναγνωσιμότητα αλλά και στην συντήρηση των εφαρμογών. Βοηθά την γρήγορη ανάπτυξη αλλά δεν υστερεί σε λειτουργίες που προσφέρουν άλλες γλώσσες προγραμματισμού. Υποστηρίζει πολύ υψηλού επιπέδου δομές δεδομένων, είναι ώριμη, παρέχει αυτόματη διαχείριση μνήμης, είναι ανοιχτού κώδικα και εκτελείται σε οποιοδήποτε λειτουργικό σύστημα. Όλα αυτά την έχουν κάνει μια πολύ δημοφιλή γλώσσα.

### 5.1.1 Ιστορικά στοιχεία

Η σύλληψη της ιδέας για την δημιουργία της έγινε στα τέλη του 1980 και άρχισε να υλοποιείται τον Δεκέμβριο του 1989 από τον Guido van Rossum. Αρχικά προοριζόταν για το λειτουργικό σύστημα Amoeba και αναπτύχθηκε στο ερευνητικό κέντρο CWI της Ολλανδίας, χρησιμοποιώντας ως οδηγό την ήδη επιτυχημένη συνταγή της γλώσσας ABC.

Οι μεγαλύτερες αλλαγές έγιναν με την δημοσίευση των εκδόσεων 2.0 και 3.0. Στην Python 2.0 προστέθηκε ο συλλέκτης απορριμάτων, αλλά και η επίσημη υποστήριξη για Unicode χαρακτήρες. Η κυκλοφορία της Python 3.0 έφερε πολλές αλλαγές, οι οποίες δημιούργησαν και προβλήματα συμβατότητας για εφαρμογές που είχαν αναπτυχθεί για προηγούμενες εκδόσεις της γλώσσας. Γι' αυτό τον λόγο υπάρχουν δύο επίσημα υποστηριζόμενες εκδόσεις. Μια για την υποστήριξη κώδικα που έχει γραφτεί με το παλαιότερο πρότυπο και μια για το νεότερο. Τελευταία έκδοση είναι η Python 3.3.0


Σχήμα 5.1: Το λογότυπο της Python

### 5.1.2 Zen of Python

Το Zen of Python [5] περιγράφει τις βασικές αρχές με τις οποίες δομήθηκε και εξελίχθηκε η Python.

- Όμορφο είναι καλύτερο από άσχημο.
- Άμεσο είναι καλύτερο από έμμεσο.
- Απλό είναι καλύτερο από σύνθετο.
- Σύνθετο είναι καλύτερο από περίπλοκο.
- Επίπεδο είναι καλύτερο από εμφωλευμένο.
- Αραιό είναι καλύτερο από πυκνό.
- Η αναγνωσιμότητα μετράει.
- Οι ειδικές περιπτώσεις δεν είναι αρκετά ειδικές, ώστε να σπάνε τους κανόνες.
- Ωστόσο η πρακτικότητα υπερτερεί της αγνότητας.
- Τα λάθη δεν θα πρέπει ποτέ να αποσιωπούνται.

- Εκτός αν αποσιωπούνται ρητά.
- Όταν αντιμετωπίζεις την αμφιβολία, αρνήσου τον πειρασμό να μαντέψεις.
- Θα πρέπει να υπάρχει ένας και προτιμητέα μόνο ένας προφανής τρόπος να το κάνεις.
- Αν και αυτός ο τρόπος μπορείς να μην είναι προφανής, εκτός αν είσαι Ολλανδός.
- Τώρα είναι καλύτερα από ποτέ.
- Αν και ποτέ είναι συχνά καλύτερα από ακριβώς τώρα.
- Αν η υλοποίηση είναι δύσκολο να εξηγηθεί, τότε είναι κακή ιδέα.
- Αν η υλοποίηση είναι εύκολο να εξηγηθεί, τότε ίσως είναι καλή ιδέα.
- Τα ονόματα χώρου (namespaces) είναι μια φοβερά καλή ιδέα – ας κάνουνε περισσότερα από αυτά!

Το ακριβές κείμενο υπάρχει μέσα στην ίδια την γλώσσα και μπορεί να το δει κάποιος πληκτρολογώντας:

```
1 import this
```

## 5.2 Flask

Το Flask (φλασκί) είναι μια μικρή βιβλιοθήκη, η οποία βοηθά την συγγραφή Python εφαρμογών για το διαδίκτυο. Βασίζεται στα εργαλεία Werkzeug, και Jinja2 και διατίθεται με την άδεια BSD.


Σχήμα 5.2: Το λογότυπο του Flask

- περιέχει λειτουργία αποσφαλμάτωσης
- εξυπηρετητή για την διαδικασία της ανάπτυξης
- ενσωματωμένη λειτουργία για σενάρια δοκιμών της εφαρμογής
- RESTful requests
- χρησιμοποιεί Jinja2 templating
- sessions
- 100% συμβατό με το WSGI 1.0
- Unicode-based
- εκτενής τεκμηρίωση
- συμβατό με το Google App Engine
- δυνατότητα εισαγωγής πρόσθετων

Αξίζει να σημειωθεί ότι η ανάπτυξή του άρχισε από τον Armin Ronacher, εμπνευσμένος από το Sinatra μετά από ένα πρωταπριλιάτικο αστείο. Όπως αναφέρει όμως και ο δημιουργός του, αποδείχθηκε αρκετά καλό, ώστε να χρησιμοποιηθεί και σε σοβαρές εφαρμογές.

## 5.3 SQLAlchemy

Η βιβλιοθήκη SQLAlchemy [25] μας προσφέρει μια πληθώρα εργαλείων ανοιχτού κώδικα για την δημιουργία ερωτημάτων σε βάσεις δεδομένων. Συγκεκριμένα, δίνεται η δυνατότητα στον προγραμματιστή να αποφύγει την συγγραφή ερωτημάτων SQL και να χρησιμοποιήσει τις κλάσεις της βιβλιοθήκης για την έκφραση των ερωτημάτων.

Ο χρήστης έχει δύο επιλογές. Μπορεί να χρησιμοποιήσει την κλασική μέθοδο, με την οποία τα ερωτήματα απλά γράφονται με την μορφή κώδικα Python, είτε να δημιουργήσει κλάσεις που αναπαριστούν τα δεδομένα, ώστε να τα διαχειρίζεται σαν κανονικά αντικείμενα της γλώσσας (ORM - Object Relational Mapper).

Η βιβλιοθήκη δημιουργήθηκε για να προσφέρει αποτελεσματική και αποδοτική πρόσβαση στα δεδομένα μια βάσης, αλλά και να μικρύνει το κενό ανάμεσα στην γλώσσα προγραμματισμού, τα δεδομένα και το πως αυτά αποθηκεύονται. Υπάρχει δηλαδή η δυνατότητα, να δημιουργηθεί μια εφαρμογή, η οποία μπορεί να λειτουργεί με τον ίδιο ακριβώς τρόπο, ανεξάρτητα από την βάση δεδομένων που χρησιμοποιείται.

Η βιβλιοθήκη έκανε την εμφάνισή της τον Φεβρουάριο του 2006 και έκτοτε έχει χρησιμοποιηθεί με επιτυχία σε πολλές εφαρμογές όπως:

- Yelp!

- reddit
- Mozilla
- Fedora Project

Χρησιμοποιώντας ORM, η δημιουργία ενός πίνακα μπορεί να οριστεί με τον εξής τρόπο.

```
# defining a database table
class User(db.Model):
    __tablename__ = 'user'

    id = db.Column(db.Integer, autoincrement=True,
                   primary_key=True)
    name = db.Column(db.String(50))
    surname = db.Column(db.String(120))
    email = db.Column(db.String(120), unique=True)
    telephone = db.Column(db.String(20))
    address = db.Column(db.String(120))
    status = db.Column(db.SmallInteger, default=ACTIVE)
    role = db.Column(db.SmallInteger, default=USER)
    password = db.Column(db.String(40))
    registered = db.Column(db.DateTime(),
                           default=datetime.datetime.now)
```

Έπειτα, η δημιουργία ερωτημάτων, καθώς και η αλλαγή των δεδομένων, γίνεται πολύ εύκολη όπως φαίνεται και στο παρακάτω τμήμα κώδικα.

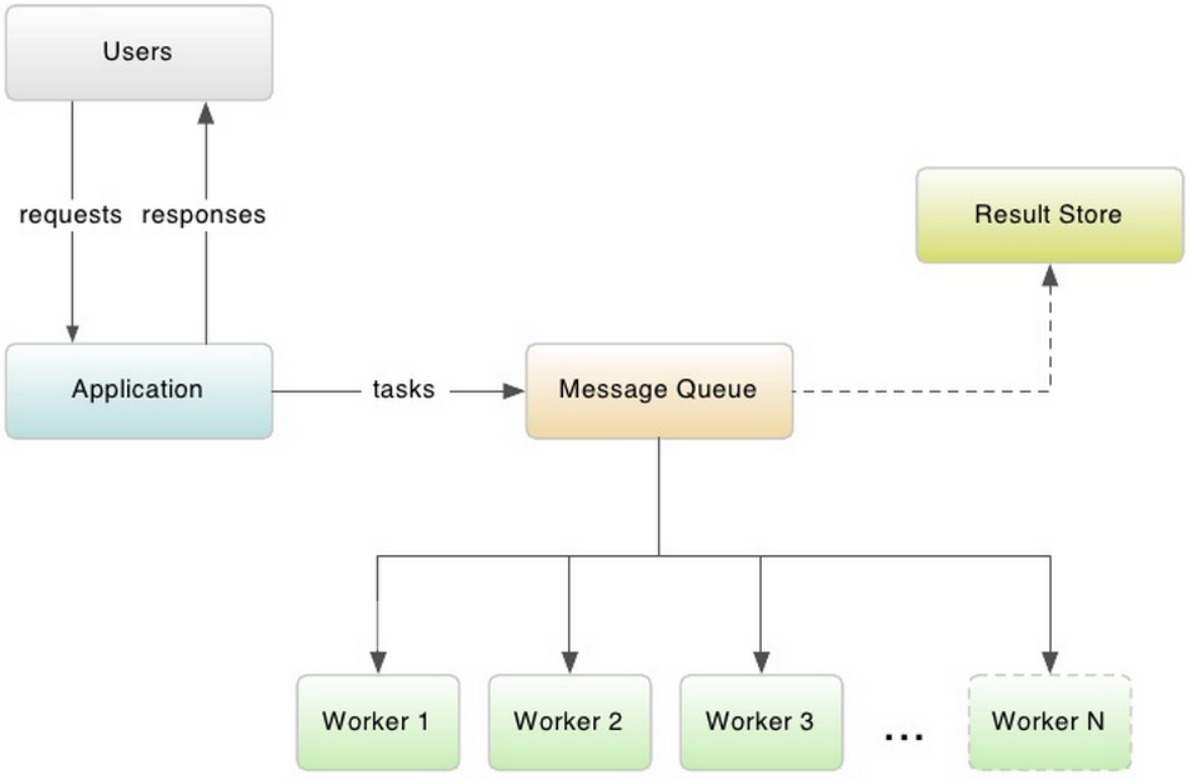
```
# quering the database
user = User.query.filter(User.id == 1).first()

# and updating data
user.name = "new name"
```

## 5.4 Celery

Το Celery [11] (Σέλινο) είναι μια βιβλιοθήκη, η οποία έχει αναπτυχθεί στην γλώσσα Python και μας παρέχει την υλοποίηση μια ουράς για ασύγχρονη εκτέλεση κώδικα. Οι εργασίες προς εκτέλεση ονομάζονται tasks και εκτελούνται ταυτόχρονα σε μια ή περισσότερες οντότητες (workers). Η λειτουργία αυτή επιτυγχάνεται χρησιμοποιώντας ανταλλαγή μηνυμάτων και επικεντρώνεται στην εκτέλεση σε πραγματικό χρόνο. Συμπληρωματικά, παρέχεται και η δυνατότητα για προγραμματισμένη εκτέλεση (scheduling). Η ανταλλαγή

των μηνυμάτων πραγματοποιείται από brokers και το Celery δεν έχει ενσωματωμένη αυτήν την λειτουργία. Έτσι οι επιλογές είναι οι εξής:

- RabbimMQ
- Redis
- Database
- Amazon SQS
- MongoDB




Σχήμα 5.3: Το μοντέλο λειτουργίας του Celery

Όπως φαίνεται στην εικόνα 2.3, ο χρήστης αλληλεπιδρά με την εφαρμογή και η εφαρμογή δημιουργεί εργασίες προς εκτέλεση. Αυτές μεταφέρονται στον μεσάζοντα (broker) και εκτελούνται μέσω των εργατών (workers).
Ένα παράδειγμα δημιουργίας και εκτέλεσης μιας εργασίας.

```python
# defining a task
from celery.decorators import task

@task
def add(x, y):
    return x + y
```

```python
# executing a task
from tasks import add

add.delay(1, 2)
```

## 5.5 AMQP

### 5.5.1 Το πρωτόκολλο AMQP

Το πρωτόκολλο AMPQ [7, 14] (Advanced Message Queueing Protocol) είναι ένα ανοιχτό πρότυπο στρώματος εφαρμογής για την ανταλλαγή μηνυμάτων. Ουσιαστικά πρόκειται για ένα πρωτόκολλο που προορίζεται για χρήση στο διαδίκτυο, όπως ακριβώς και τα HTTP, TCP αλλά ασύγχρονο.

Η λογική είναι ότι ένα μήνυμα πρέπει να μπει σε μια ουρά, ώστε να μπορέσει να φτάσει στον προορισμό του. Έτσι οι καταναλωτές δημιουργούν ουρές, ώστε να επεξεργάζονται την εισερχόμενη ροή μηνυμάτων. Οι καταναλωτές ενημερώνουν τις ουρές για να προσκολληθούν σε κάποια συγκεκριμένη ροή, χρησιμοποιώντας ένα κλειδί. Οι δημιουργοί των μηνυμάτων χρησιμοποιώντας το ίδιο κλειδί, μπορούν να στέλνουν μηνύματα. Οι ανταλλαγές (exchanges) δρομολογούν αυτά τα μηνύματα, χρησιμοποιώντας τα κλειδιά που έχουν επιλεγεί για την επικοινωνία.

Άρα, τα σημαντικότερα χαρακτηριστικά που προσφέρει το AMQP, είναι η διευθυνσιοδότηση του μηνύματος, η δυνατότητα αναμονής, η δρομολόγηση, η αξιοπιστία και η ασφάλεια για την μετάδοση μηνυμάτων.

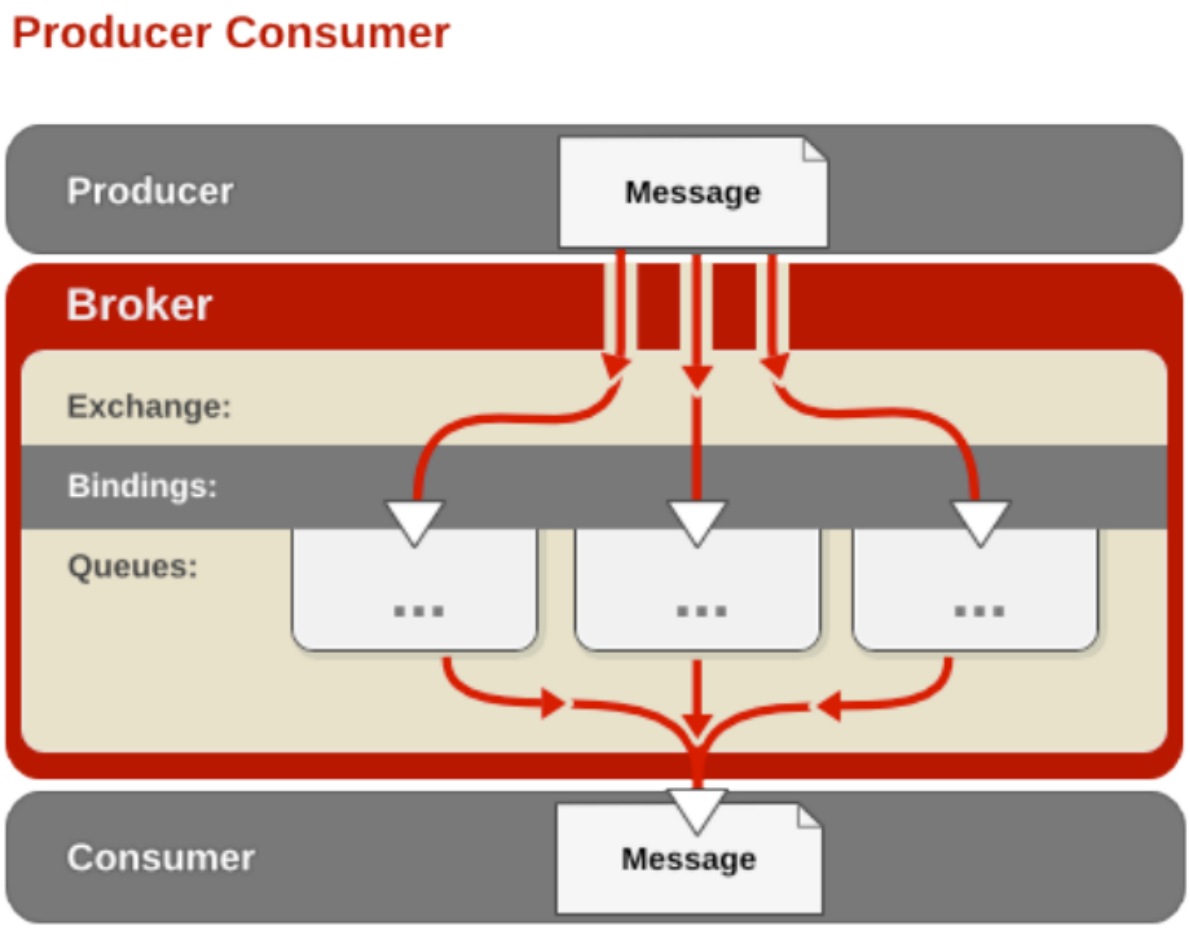
### 5.5.2 Το RabbitMQ

Το RabbitMQ είναι ένα λογισμικό ανοιχτού κώδικα που λειτουργεί ως μεσάζοντας στην ανταλλαγή μηνυμάτων και ουσιαστικά υλοποιεί το πρωτόκολλο AMQP.

Το RabbitMQ έχει γραφτεί στην γλώσσα Erlang και ο πηγαίος κώδικας διατίθεται στο κοινό υπό την άδεια Mozilla Public License..

## 5.6 Bootstrap

Το Bootstrap είναι μια συλλογή εργαλείων η οποία δίνει την δυνατότητα στον προγραμματιστή να αναπτύξει γρήγορα εφαρμογές διαδικτύου δίνοντας περισσότερη έμφαση




Σχήμα 5.4: Η ροή των δεδομένων στο πρωτόκολλο AMQP

στην λειτουργικότητα της εφαρμογής και όχι στην εμφάνιση. Αναπτύχθηκε από το Twitter για την δημιουργία του ιστοχώρου της υπηρεσίας και έπειτα ο κώδικας έγινε διαθέσιμος προς όλους με την άδεια Apache License v2.0.

Η βασική λογική με την οποία ο χρήστης καλείται να διαχειριστεί το περιεχόμενο είναι αυτή του πλέγματος (grid). Δηλαδή, υπάρχει μια σειρά από κλάσεις, οι οποίες καταλαμβάνουν συγκεκριμένο χώρο σε μια ιστοσελίδα και μπορούν να χρησιμοποιηθούν, ώστε να επιτευχθεί το επιθυμητό χωρικό αποτέλεσμα. Βέβαια, εκτός από αυτή την βασική λογική, παρέχονται και άλλες βοηθητικές κλάσεις οι οποίες χρησιμοποιούνται πολύ συχνά (π.χ εμφάνιση αναδυόμενων παραθύρων, σφαλμάτων κλπ).

Η δημιουργία του έγινε με γνώμονα τους νέους περιηγητές και έτσι κάποιες λειτουργίες ενδέχεται να μην είναι διαθέσιμες σε παλαιότερους. Οι επίσημα υποστηριζόμενοι περιηγητές είναι:

- Chrome

- Firefox
- Safari
- Opera
- Internet Explorer

## 5.7 Windows Phone

### 5.7.1 Γενικά για τα Windows Phone

Το Windows Phone [26] είναι ένα λειτουργικό σύστημα το οποίο αναπτύχθηκε από την Microsoft για κινητές συσκευές. Αποτελεί τον συνεχιστή των επιτυχημένων Windows Mobile, χωρίς όμως να υπάρχει προς τα πίσω συμβατότητα των εφαρμογών. Αυτό συνέβη κυρίως, γιατί η εταιρία αποφάσισε να πραγματοποιήσει μεγάλες αλλαγές στο λειτουργικό της σύστημα, λόγω ανταγωνισμού από τα Android και IOS.

Η παρουσίαση του νέου λειτουργικού συστήματος, έγινε τον Οκτώβριο του 2010 με την έκδοση 7. Από τότε η Microsoft έχει προχωρήσει σε τρεις σημαντικές ενημερώσεις του λειτουργικού συστήματος, με τις εκδόσεις 7.5, 7.8 και 8. Στην τελευταία έκδοση έγινε και η προσθήκη για υποστήριξη της αρχιτεκτονικής x86, αφού μέχρι τότε το λειτουργικό σύστημα μπορούσε να εκτελεστεί μόνο σε αρχιτεκτονική ARM.

### 5.7.2 Silverlight

Το Silverlight [23] είναι μια σουίτα εργαλείων για την δημιουργία και την εκτέλεση εφαρμογών διαδικτύου. Η λειτουργία του είναι παρόμοια με το Adobe Flash. Το περιβάλλον εκτέλεσης του Silverlight είναι διαθέσιμο ως πρόσθετο για τους περισσότερους περιηγητές στην πλειονότητα των λειτουργικών συστημάτων.

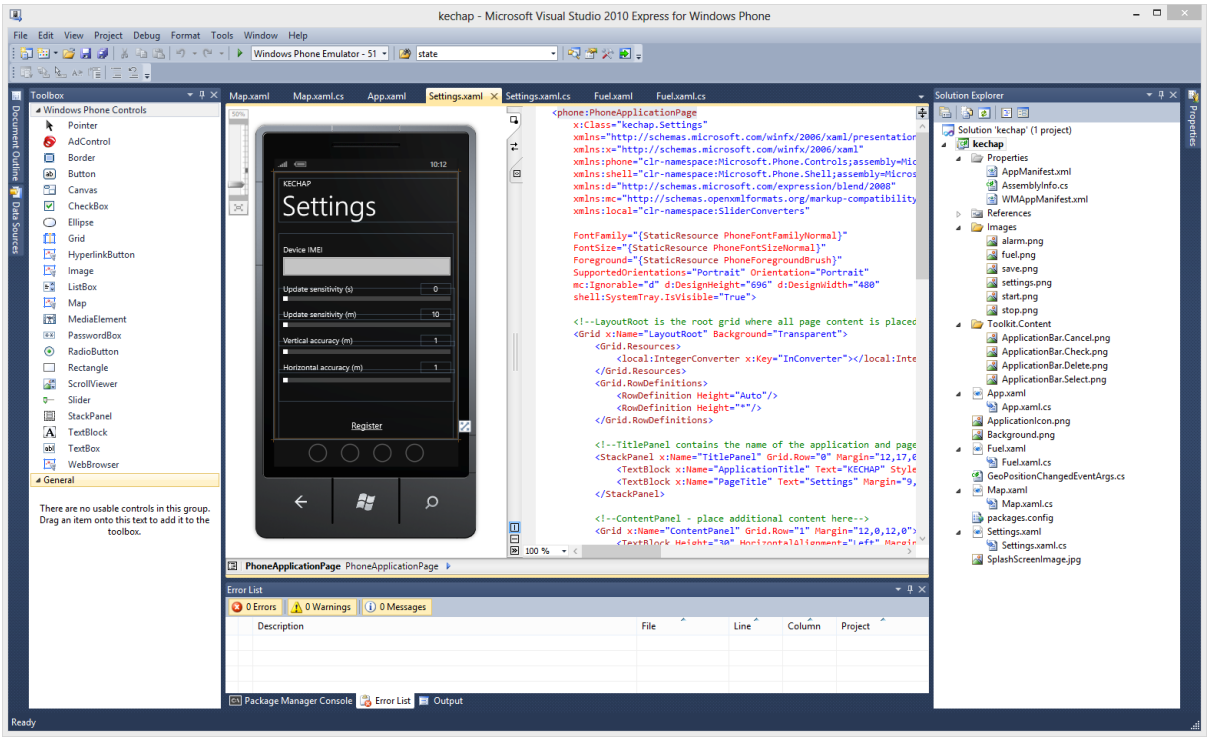
Αρχικά ο προσανατολισμός του ήταν το streaming, αλλά στην συνέχεια προστέθηκαν και άλλες δυνατότητες (γραφικά, animation, κλπ). Επίσης, έγινε η κύρια πλατφόρμα για την ανάπτυξη εφαρμογών σε Windows Phone με την έκδοση 4.

Μετά την συμφωνία συνεργασίας της Microsoft με την Nokia, το Silverlight πλέον μπορεί να εκτελεστεί και σε λειτουργικό σύστημα Symbian (S40, S60), με περιορισμένες ωστόσο δυνατότητες, καθώς υποστηρίζεται μόνο η έκδοση Silverlight 2.

### 5.7.3 Περιβάλλον ανάπτυξης

Η ανάπτυξη των εφαρμογών γίνεται αποκλειστικά με την χρήση του Visual Studio. Το εργαλείο αυτό παρέχεται δωρεάν από την Microsoft και οποιοσδήποτε μπορεί να το κατεβάσει και να ξεκινήσει να δημιουργεί εφαρμογές για την πλατφόρμα Windows Phone.

Το Visual Studio προσφέρει ένα εύκολο εργαλείο για την γρήγορη δημιουργία του περιβάλλοντος της εφαρμογής με επιλογές για φόρμες, κουμπιά, εικόνες κ.α. Εναλλακτικά ο χρήστης μπορεί να χρησιμοποιήσει XML για να δημιουργήσει το επιθυμητό αποτέλεσμα.




Σχήμα 5.5: Το περιβάλλον ανάπτυξης εφαρμογών Visual Studio

Μαζί με το περιβάλλον ανάπτυξης, υπάρχει και προσομοιωτής δίνοντας έτσι την δυνατότητα στον προγραμματιστή να δοκιμάσει σε πραγματικές συνθήκες την εφαρμογή του.

## 5.8 Google Maps

### 5.8.1 Γενικά

Το Google Maps [19] είναι μια διαδικτυακή πλατφόρμα προβολής πληροφοριών πάνω σε γεωγραφικούς χάρτες. Η εφαρμογή προσφέρει οδικούς χάρτες, δυνατότητα για χάραξη διαδρομών ανάλογα με το μεταφορικό μέσω (ποδήλατο, αυτοκίνητο, ΜΜΜ) και σύστημα εντοπισμού σημείων ενδιαφέροντος για όλο τον κόσμο. Η πλατφόρμα έγινε διαθέσιμη στο κοινό τον Φεβρουάριο του 2005. Το αρχικό όνομα ήταν Google Local, αλλά αυτό άλλαξε σε Google Maps, για να αντιπροσωπεύει καλύτερα τις υπηρεσίες που προσφέρει και φυσικά έχουν να κάνουν στο μεγαλύτερο μέρος τους με χάρτες.

### 5.8.2 Google Maps API

Πέντε μήνες μετά την επίσημη παρουσίαση του Google Maps, η Google ξεκίνησε το Google Maps API, επιτρέποντας τους προγραμματιστές να ενσωματώσουν την υπη-

ρεσία στις ιστοσελίδες τους. Πρόκειται για μια δωρεάν υπηρεσία και επί του παρόντος δεν περιέχει διαφημίσεις, ωστόσο οι όροι χρήσης αναφέρουν ότι μελλοντικά μπορεί να υπάρξουν.

Το Google Maps είναι δωρεάν για εμπορική χρήση, με την προϋπόθεση ότι δεν χρεώνεται η χρήση των χαρτών σε τρίτους και ότι οι προσβάσεις στους χάρτες δεν ξεπερνούν τις 25.000 ανά ημέρα. Σε περιπτώσεις που δεν μπορούν να ικανοποιηθούν αυτά τα κριτήρια, η Google παρέχει το Google Maps API for Business.

Η επιτυχία του Google Maps API έχει δημιουργήσεις μια σειρά ανταγωνιστικών εναλλακτικών λύσεων. Μερικές από αυτές είναι, το Yahoo! Maps API, Bing Maps Platform, MapQuest Development Platform, και OpenLayers.

Το API προσφέρει στις προγραμματιστές διάφορες υπηρεσίες που έχουν να κάνουν με τους χάρτες της Google. Οι πιο σημαντικές είναι, η δυνατότητα δημιουργίας στρωμάτων με περιεχόμενο που ορίζεται από τον χρήστη, η εισαγωγή στοιχείων στους χάρτες αλλά και η υποβολή ερωτημάτων, σχετικά με δεδομένα που υπάρχουν σε αυτούς.

## 5.9 Arduino

Το Arduino [15] είναι μια αναπτυξιακή πλακέτα που, φέρει έναν ενσωματωμένο μικροελεγκτή και παρέχει στον χρήστη την δυνατότητα να πειραματιστεί με τις εισόδους και τις εξόδους του, με μεγάλη ευκολία. Ο προγραμματισμός μπορεί να γίνει χρησιμοποιώντας την γλώσσα C, είτε μια τροποποιημένη έκδοση αυτής που ονομάζεται Processing.

Το Arduino είναι έργο ανοιχτού κώδικα, που σημαίνει ότι οποιοσδήποτε μπορεί να δει τον σχεδιασμό της πλακέτας, να τον αντιγράψει και να τον εμπορευθεί και ο ίδιος. Γι' αυτόν τον λόγο υπάρχουν διάφορες εναλλακτικές προτάσεις, που είναι απόλυτα συμβατές με την επίσημη συσκευή. Το μεγαλύτερο πλεονέκτημα που προσφέρει η πλατφόρμα, είναι η ευκολία με την οποία κάποιος μπορεί να βρει υλικό που θα ενσωματωθεί πάνω στην συσκευή και θα της προσφέρει νέες δυνατότητες (shields).

Οι μικροελεγκτές που χρησιμοποιούνται είναι συνήθως οι atmega168 και atmega328. Όλες οι πλακέτες ενσωματώνουν ρυθμιστή τάσης στα 5V και κρυσταλλικό ταλαντωτή 16 MHz. Επίσης, κάθε μικροελεγκτής είναι από κατασκευής προγραμματισμένος με τον Arduino bootloader, έτσι ώστε να μπορεί να γίνει ο προγραμματισμός, χωρίς την χρησιμοποίηση εξωτερικής συσκευής. Έτσι ο προγραμματισμός γίνεται χρησιμοποιώντας την θύρα USB, μέσω εικονικής σειριακής σύνδεσης. Βέβαια, μπορεί να χρησιμοποιηθεί και εξωτερική συσκευή, αφού υποστηρίζεται η σύνδεση με προγραμματιστή SPI.

Οι περισσότερες επαφές εισόδου/εξόδου της πλακέτας είναι εκτεθειμένες για σύνδεση εξωτερικών κυκλωμάτων. Οι περισσότερες εκδόσεις του Arduino παρέχουν 14 ψηφιακές επαφές εισόδους/εξόδους και 6 αναλογικές. Από τις ψηφιακές, 6 μπορούν να παράξουν και σήματα PWM.

Για την συγγραφή των προγραμμάτων παρέχεται το Arduino IDE. Έχει αναπτυχθεί στην γλώσσα προγραμματισμού Java και ως εκ τούτου μπορεί να εκτελεστεί σε πολλές πλατφόρμες. Εκτός από επεξεργαστή κώδικα και μεταγλωττιστή, περιλαμβάνει και εργαλεία για την φόρτωση των προγραμμάτων στο Arduino.

## 5.10 Η επιλογή των τεχνολογιών

Η επιλογή των περισσότερων τεχνολογιών έγινε, με βάση το ότι η εφαρμογή θα βασιστεί στην γλώσσα Python. Η Python επιλέχθηκε γιατί πρόκειται για μια γλώσσα, η οποία έχει πολύ καλή υποστήριξη, μεγάλη κοινότητα προγραμματιστών και εξαιρετικά εργαλεία, τα οποία πολλές φορές είναι αδύνατο να βρεθούν συγκεντρωμένα σε μια γλώσσα σεναρίων (scripting).

| Project | Language | Ajax | MVC framework | MVC push-pull | i18n & L10n? | ORM | Testing framework(s) | DB migration framework(s) | Security framework(s) | Template framework(s) | Caching framework(s) | Form validation framework(s) | Python 3.* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CherryPy | Python | Yes | controller & URL dispatching | | Yes | ORM agnostic | use stdlib's unittest and doctest | depends on ORM | | Templating engine agnostic | Yes | Form validation engine agnostic | Yes |
| CubicWeb | Python | Yes | controller & URL dispatching | | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No |
| Flask | Python | Yes | Yes | Push | Yes | Yes | Yes | Yes | Yes | Jinja2 | Yes | Yes | No |
| Grok | Python | Yes | Yes | Pull | Yes | OODBMS called ZODB, SQLAlchemy, Storm | Unit tests, functional tests | ZODB Generations | Yes | Yes | Yes | Yes | |
| Pyjamas | Python, JavaScript | Yes | Use PureMVC Python version (compiled to JavaScript) | | Yes | ??, no direct data access | | No | | | | | No |
| Pylons | Python | helpers for Prototype and script.aculo.us | controller | Push | Yes | ORM-agnostic | via nose | depends on ORM | | pluggable: Mako, Genshi, Myghty, Kid, more | Beaker cache (memory, memcached, file, databases) | preferred formencode | No |
| Pyramid | Python | Yes | Yes | Push | Yes | ORM-agnostic | Yes | depends on ORM | Yes | pluggable: Chameleon, Genshi, Mako, more | Beaker cache (memory, memcached, file, databases) | preferred formencode | Yes |
| Django | Python | Yes | Full stack | Push | Yes | Django ORM | Yes | Provided by South | ACL-based | Django Template Language | Cache Framework | Django Forms API | Yes |
| TurboGears | Python | Toolkit-independent, provides support via JSON | Full stack, best-of-breed based | Push | Yes | SQLAlchemy | nose | SQLAlchemy-Migrate | Repoze.what & Repoze.who | pluggable: Genshi, more | Support for memcached, and any WSGI compliant system | ToscaWidgets, utilizing FormEncode | No |
| web2py | Python | Yes | Yes | Push | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No |
| Webware | Python | No | Optional | Pull | No | Yes | Yes | No | Yes | Yes | No | No | No |
| BlueBream (Zope 3) | Python | via add-on products, e.g. Plone w/KSS | Yes | Pull | Yes | ZODB, SQLObject, SQLAlchemy | Unit tests, functional tests | ZODB generations | ACL-based | Yes | Yes | Yes | No |
| Zope 2 | Python | | Yes | Pull | Yes | ZODB, SQLObject, SQLAlchemy | Unit tests | | ACL-based | Yes | Yes | CMFFormController | No |
| mnml | Python | No | No | | No | No | only development server | No | | Templating engine agnostic | No | Form validation engine agnostic | No |

Σχήμα 5.6: Σύγκριση βιβλιοθηκών Python για διαδικτυακές εφαρμογές [16]

Η επιλογή του Flask έγινε γιατί δίνει την ευχέρεια στον χρήστη να προσθέσει μόνο τις λειτουργίες που του χρειάζονται και να συνδυάσει με ευκολία διαφορετικά υποσυστήματα. Όλα αυτά, κάνουν την εφαρμογή πολύ εύκολα επεκτάσιμη και ευέλικτη, από άποψη συντήρησης.

Η SQLAlchemy επιλέχθηκε γιατί η χρησιμοποίηση της, σημαίνει ότι η εφαρμογή μπορεί να λειτουργήσει με όλες τις δημοφιλείς βάσεις δεδομένων, χωρίς καμία αλλαγή στον κώδικα. Επίσης, προτιμήθηκε από την μοναδική εναλλακτική (Peewee), γιατί κρίθηκε πιο ώριμη και προσφέρει μεγαλύτερη ευελιξία επιλογής βάσης δεδομένων.

Η επιλογή του Celery ήταν υποχρεωτική, αφού δεν υπάρχει κάποια άλλη εναλλακτική πρόταση στην Python. Η επιλογή του RabbitMQ εξαρτήθηκε άμεσα από την χρησιμοποίηση του Celery. Μπορεί να υπάρχουν και άλλα συστήματα που προσφέρουν τις ίδιες δυνατότητες, αλλά το RabbitMQ είναι η λύση που συστήνεται από το ίδιο το Celery Project. Επίσης, ανταποκρίνεται πολύ καλά σε μεγάλο φόρτο και εγγυάται την σωστή επικοινωνία με το Celery, χωρίς απώλεια μηνυμάτων.

Το API της Google για την απεικόνιση των δεδομένων, χρησιμοποιήθηκε γιατί υπήρχε προηγούμενη εμπειρία με αυτό. Ο κύριος λόγος επιλογής του όμως ήταν, γιατί οι χάρτες των ανταγωνιστικών προϊόντων δεν εμφανίζουν την ίδια πληρότητα.

Το Bootstrap επιλέχθηκε περισσότερο για πειραματισμό με εργαλεία τέτοιου είδους.

Τα Windows Phone και Arduino επιλέχθηκαν γιατί το υλικό ήταν ήδη διαθέσιμο, οπότε δεν εξετάστηκαν άλλες επιλογές λόγω κόστους.

# Κεφάλαιο 6

# Επίλογος

Στο κεφάλαιο αυτό γίνεται μια σύνοψη της διπλωματικής εργασίας και των αποτελεσμάτων που προέκυψαν από την δημιουργία της. Γίνεται επίσης αναφορά στους τρόπους, με τους οποίους μπορούν να χρησιμοποιηθούν τα αποτελέσματα αυτά, για την μελλοντική επέκταση του συστήματος.

## 6.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία δημιουργήθηκε ένα πληροφοριακό σύστημα διαχείρισης στόλου οχημάτων. Αυτό βασίζεται στις τεχνολογίες GPS και GPRS και αποτελείται από:

α) τις συσκευές καταγραφής, που τοποθετούνται στο όχημα και έχουν την δυνατότητα καταγραφής των συντεταγμένων και αποστολής αυτών σε ένα εξωτερικό σύστημα

β) τον εξυπηρετητή, που λαμβάνει τα δεδομένα των συσκευών καταγραφής και τα αποθηκεύει σε μια βάση δεδομένων

γ) την διαδικτυακή εφαρμογή, που επεξεργάζεται τα δεδομένα και εμφανίζει στον χρήστη πληροφορίες σχετικές με την λειτουργία των οχημάτων του

Σαν συσκευές χρησιμοποιήθηκαν ένα κινητό τηλέφωνο με το λειτουργικό σύστημα Windows Phone, μια συσκευή arduino με GPS, GPRS shields και μια συσκευή του εμπορίου (TK106). Οι δύο πρώτες συσκευές προγραμματίστηκαν από την αρχή για να επιτευχθεί το επιθυμητό αποτέλεσμα. Για την τρίτη, αναπτύχθηκε ενδιάμεσο λογισμικό (middleware) στον εξυπηρετητή για την παραλαβή των δεδομένων. Για την αξιοποίηση των δεδομένων που λαμβάνονται από τις συσκευές αναπτύχθηκε μια διαδικτυακή εφαρμογή. Αυτή έχει ως κύριο στόχο την παρακολούθηση των οχημάτων. Επίσης, αναπτύχθηκαν και άλλες υπηρεσίες, ώστε το αποτέλεσμα να βρίσκεται όσο το δυνατόν πιο κοντά σε ένα πραγματικό σύστημα:

- καθορισμός γεωφράκτη
- προβολή διαδρομών
- λήψη περιοδικών αναφορών

- στατιστικά στόλων - οχημάτων
- προγραμματισμός συντήρησης

Κατά την ανάπτυξη, όπως ήταν αναμενόμενο, προέκυψαν κάποια προβλήματα, τα οποία λύθηκαν σε ικανοποιητικό βαθμό. Η παρακολούθηση της θέσης ενός οχήματος, πραγματοποιείται με περιοδικές αιτήσεις ανανέωσης περιεχομένου από τον web brower. Η ιδανική περίπτωση θα ήταν να εμφανίζεται μια νέα θέση του οχήματος, όταν αυτή γίνει διαθέσιμη. Κάτι τέτοιο όμως δεν στάθηκε δυνατό.

Επίσης, εκ του αποτελέσματος φάνηκε ότι υπάρχουν μικρά προβλήματα με την ακρίβεια του γεωφράκτη και του οδομέτρου. Στην πρώτη περίπτωση διαπιστώθηκε ότι τα αποτελέσματα του αλγορίθμου είναι τυχαία, όταν ένα όχημα κινείται μερικά μέτρα εντός ή εκτός του ορίου που έχει τεθεί. Στην περίπτωση του οδομέτρου, διαπιστώθηκε ότι οι αποστάσεις που μετρώνται είναι ανακριβείς, όταν υπάρξει μια πολύ μικρή διαφορά μεταξύ συντεταγμένων.

Η σκέψη για τον υπολογισμό κατανάλωσης ήταν διαφορετική εξ' αρχής. Γινόταν αυτόματα χρησιμοποιώντας το οδόμετρο, την μέση τιμή κατανάλωσης καυσίμου ανά 100χλμ και την μέση τιμή του καυσίμου. Αυτή η προσέγγιση όμως δεν είχε κάποια ιδιαίτερη αξία για τον χρήστη, αφού η κατανάλωση εξαρτάται από πολλές παραμέτρους, που δεν είναι δυνατό να εισαχθούν στο σύστημα χωρίς την προσθήκη κάποιου αισθητήρα μέτρησης. Έτσι, αποφασίστηκε να αφαιρεθεί αυτή η επιλογή και να προστεθεί η δυνατότητα για χειροκίνητη εισαγωγή των δεδομένων ανεφοδιασμού.

Το σύστημα που αναπτύχθηκε, δοκιμάσθηκε εκτενώς σε πραγματικές συνθήκες. Υπήρξαν τουλάχιστον 3 οχήματα και 4 χρήστες που δημιούργησαν περίπου 20.000 εγγραφές στην βάση δεδομένων σε ένα διάστημα δύο μηνών. Το συμπέρασμα είναι ότι η εφαρμογή μπορεί να λειτουργήσει χωρίς προβλήματα και ότι με μερικές προσθήκες, θα μπορούσε να ανταγωνιστεί εφαρμογές που διατίθενται με αμοιβή.

## 6.2 Μελλοντικές επεκτάσεις

### 6.2.1 GPRS και SMS

Το σύστημα λειτουργεί αποκλειστικά χρησιμοποιώντας το GPRS. Σε μερικές περιπτώσεις μπορεί να είναι οικονομικότερη ή ακόμη και πιο εύκολη η χρήση μηνυμάτων SMS. Σε κάθε περίπτωση, θα ήταν καλό να δίνεται στον χρήστη η δυνατότητα επιλογής υπηρεσίας. Θα μπορούσε ακόμη και το σύστημα να παίρνει αυτή την απόφαση, χρησιμοποιώντας τα δεδομένα που έχει στην διάθεσή του (υπόλοιπο κάρτας, χρόνος μεταξύ αποστολής δεδομένων).

Όλες οι συσκευές που χρησιμοποιήθηκαν, έχουν δυνατότητα αποστολής SMS. Μπορεί δηλαδή, να προστεθεί αυτή η δυνατότητα πολύ εύκολα. Το μόνο πρόβλημα με αυτή την προσέγγιση, είναι η συλλογή των μηνυμάτων SMS και η αποθήκευσή τους στην βάση. Για την επίλυση αυτού του προβλήματος μπορεί να χρησιμοποιηθεί, είτε κάποια online υπηρεσία, είτε κάποιο κινητό ή 3G modem, σε συνδυασμό με μια βιβλιοθήκη για την

επικοινωνία υπολογιστή και συσκευής. Πραγματοποιήθηκε λοιπόν μια δοκιμή με την συσκευή One Touch X211S και την βιβλιοθήκη gammu, η οποία ήταν και επιτυχημένη.

### 6.2.2 Μεγαλύτερος έλεγχος στο όχημα

Τα συστήματα που απευθύνονται στον επιχειρηματικό κλάδο έχουν καλύτερο έλεγχο στο όχημα. Για παράδειγμα, υπάρχουν αισθητήρες για την πίεση των ελαστικών, για την στάθμη του καυσίμου, για την κατάσταση κινητήρα κ.α. Επίσης στις περισσότερες περιπτώσεις, δίνεται και η δυνατότητα για κλείδωμα ή ακινητοποίηση του οχήματος.

Γίνεται άμεσα αντιληπτό ότι η προσθήκη αυτών των χαρακτηριστικών, θα έκανε την εφαρμογή πληρέστερη και άρα πιο ανταγωνιστική. Η υλοποίηση αυτού του εγχειρήματος όμως είναι πιο δύσκολη απ' ότι φαίνεται αρχικά. Για παράδειγμα, στην συσκευή TK106 δεν υπάρχει δυνατότητα παρέμβασης. Η καλύτερη λύση είναι να χρησιμοποιηθούν εξειδικευμένες συσκευές που παρέχουν την δυνατότητα προσθήκης αισθητήρων.

### 6.2.3 Υποστήριξη περισσότερων συσκευών

Υπάρχουν πάρα πολλές συσκευές διαθέσιμες στην αγορά που μπορούν να παραμετροποιηθούν, ώστε να υποστηριχθούν από την εφαρμογή. Έτσι, ένας μεγάλος όγκος χρηστών που έχει ήδη κάποια συσκευή, θα έχει πρόσβαση στην εφαρμογή. Η σκέψη είναι ότι θα μπορούν να χρησιμοποιηθούν πιο πολύπλοκες συσκευές, που έχουν υποστήριξη του CAN bus και OBD, ώστε να επιτευχθεί και καλύτερος έλεγχος για το όχημα που αναφέρθηκε προηγουμένως.

Η διαδικασία της προσθήκης μια συσκευής είναι ανάλογη με την συσκευή TK106 που έγινε στο αμέσως προηγούμενο κεφάλαιο. Γίνεται αρχικά έλεγχος της δομής των δεδομένων και έπειτα προγραμματίζεται η μετατροπή (αν χρειάζεται) τους, ώστε να είναι στην ίδια μορφή με αυτή που χρησιμοποιεί η εφαρμογή.

### 6.2.4 Επικοινωνία συστήματος και οχήματος

Το σύστημα που αναπτύχθηκε είναι σχεδόν μονόπλευρο. Το ιδανικό θα ήταν να μπορεί να υπάρχει και επικοινωνία από το σύστημα προς το όχημα, προς όφελος του οδηγού. Για παράδειγμα, αποστολή δρομολογίων, ανταλλαγή μηνυμάτων και άλλων πληροφοριών που μπορεί να χρειάζεται να μεταδοθούν από έναν χρήστη του συστήματος προς ένα όχημα.

Για να υλοποιηθούν αυτές οι λειτουργίες, θα έπρεπε να καθοριστεί ο τρόπος επικοινωνίας και το είδος των δεδομένων που θα ανταλλάσσονται μεταξύ των οντοτήτων. Η επικοινωνία θα μπορούσε να γίνει χρησιμοποιώντας ένα socket, στο οποίο θα μπορούν να γράφουν και να διαβάζουν και τα δυο μέρη του συστήματος.

# Συντομογραφίες

| | |
|---|---|
| GSM | Global System for Mobile communications |
| GPRS | General Packet Radio Service |
| SMS | Short Message Service |
| GPS | Global Positioning System |
| CAN | Contoller Area Network |
| OBD | On-board Diagnostics |
| IRNSS | Indian Regional Navigational Satellite System |
| GLONASS | Global Navigation Satellite System |
| SA | Selective Availability |
| PRN | PseudoRandom Noise |
| TLM | TeLeMetry word |
| HOW | HandOver Word |
| ETSI | European Telecommunications Standards Institute |
| BSS | Base Station Subsystem |
| SIM | Subscriber Identity Module |
| PIN | Personal Identification Number |
| BTS | Base Transceiver Station |
| BSC | Base Station Controller |
| MSC | Mobile Switching Center |
| TDMA | Time Division Multiple Access |
| DCS | Digital Cellular Service |
| PCS | Personal Communication Service |
| AC | Authentication Center |
| IMEI | International Mobile Station Equipment Identity |
| IMSI | International Mobile Subscriber Identity |
| CSD | Circuit Switched Data |
| PSTN | Public Switched Telephone Network |
| AMQP | Advanced Message Queuing Protocol |
| API | Application Programming Interface |
| LED | Light-Emitting Diode |
| UI | User Interface |
| WP | Windows Phone |

| | |
|---|---|
| IP | Internet Protocol |
| TCP | Transmission Control Protocol |
| SPI | Serial Peripheral Interface |
| ORM | Object-Relational Mapping |
| AJAX | Asynchronous JavaScript and XML |
| JSON | JavaScript Object Notation |

# Βιβλιογραφία


[1] Β. Γερογιάννης Γ. Κακαρόντζας Α. Καμέας Γ. Στάμελος Π. Φιτσίλης. *Αντικειμενοστρεφής Ανάπτυξη Λογισμικού με τη UML.* Εκδόσεις Κλειδάριθμος, 2006.

[2] Darel Rex Finley. Point-in-polygon algorithm. http://alienryderflex.com/polygon/, April 1998,2006,2007.

[3] R. Ramakrishnan J. Gehrke. *Συστήματα Διαχείρισης Βάσεων Δεδομένων.* Εκδόσεις Τζιόλα, 2000.

[4] Google. Google maps api v3. https://developers.google.com/maps/documentation/javascript/, May 2013.

[5] Dimitris A. Leventeas. *Οδηγός Εκμάθησης Python Βήμα Βήμα.* python.org.gr, 2000.

[6] Movable Type Ltd. Calculate distance, bearing and more between latitude/longitude points. http://www.movable-type.co.uk/scripts/latlong.html, April 2013.

[7] M. Mahendra. The distributed task queue. PyCon, November 2010.

[8] Mio. Περιγραφή gps τεχνολογίας. http://eu.mio.com/el_gr/global-positioning-system.htm, May 2013.

[9] MSDN. Geocoordinatewatcher. http://msdn.microsoft.com/en-us/library/system.device.location.geocoordinatewatcher.aspx, May 2013.

[10] Myphone. Gprs. http://www.myphone.gr/library/article-33.html, May 2013.

[11] Celery Project. Celery: Distributed task queue. http://www.celeryproject.org/, April 2013.

[12] SeeedStudio. Gprs shield v1.0. http://www.seeedstudio.com/wiki/GPRS_Shield_V1.0, May 2013.

[13] Jayesh Sukumaran. Real time mobile gps tracker with google maps. http://jayeshprojects.blogspot.gr/2010/04/real-time-mobile-gps-tracker-with.html, April 2013.

[14] Wikipedia. Advanced message queuing protocol. http://en.wikipedia.org/wiki/AMQP, April 2013.

[15] Wikipedia. Arduino. http://el.wikipedia.org/wiki/Arduino, April 2013.

[16] Wikipedia. Comparison of web application frameworks. http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks, April 2013.

[17] Wikipedia. E.164. http://en.wikipedia.org/wiki/E.164, May 2013.

[18] Wikipedia. Fleet management. http://en.wikipedia.org/wiki/Fleet_management, May 2013.

[19] Wikipedia. Google maps. http://en.wikipedia.org/wiki/Google_maps, April 2013.

[20] Wikipedia. Gps. http://el.wikipedia.org/wiki/GPS, May 2013.

[21] Wikipedia. Gsm. http://el.wikipedia.org/wiki/GSM, May 2013.

[22] Wikipedia. Haversine formula. http://en.wikipedia.org/wiki/Haversine_formula, April 2013.

[23] Wikipedia. Microsoft silverlight. http://en.wikipedia.org/wiki/Microsoft_Silverlight, April 2013.

[24] Wikipedia. Python (programming language), April 2013.

[25] Wikipedia. Sqlalchemy. http://en.wikipedia.org/wiki/SQLAlchemy, April 2013.

[26] Wikipedia. Windows phone. http://en.wikipedia.org/wiki/Windows_phone, April 2013.