Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

Σχεδιασμός και ανάπτυξη εφαρμογής Mobile Smart Parking ("Έξυπνο Παρκάρισμα") με χρήση στατιστικής καταγραφής θέσεων

Ονοματεπώνυμο : Στυλιανός Αλμπάνης
Αριθμός Μητρώου : ΜΕ/08073
Ημερομηνία : 28/06/2010

Περιεχόμενα

1. Εισαγωγή

Η χρήση ενός αυτοκινήτου έχει πολλά πλεονεκτήματα αλλά και μειονεκτήματα. Ένα σημαντικό μειονέκτημα είναι η έλλειψη χώρων παρκαρίσματος. Αυτό συναντάται σε μεγαλύτερο βαθμό σε πόλεις με αυξημένο πληθυσμό και έλλειψη υποδομών παρκαρίσματος. Οι ψηφιακές τεχνολογίες δεν έχουν τη δυνατότητα να επιλύσουν ολοκληρωτικά το πρόβλημα, όμως μπορούν να βοηθήσουν στον μετριασμό του.

Είναι γνωστό πόσο ακριβή, περιζήτητη και χρονοβόρα μπορεί να είναι η διαδικασία παρκαρίσματος. Μετά από πρωταρχική έρευνα που διεξήχθη, δεν βρέθηκε κάτι που να ικανοποιεί τις ανάγκες των χρηστών σχετικά με την εύρεση εύκολου παρκαρίσματος. Γι' αυτό το λόγο θεωρήθηκε αναγκαία η ανάπτυξη ενός συστήματος από το μηδέν, καθώς και η διεξοδική επεξήγηση του συστήματος που αναπτύχθηκε για τις ανάγκες της παρούσας εργασίας. Πιο συγκεκριμένα, πραγματοποιήθηκε εστιασμός στην τεχνική μεριά (προγραμματιστική κυρίως) του προβλήματος, και αναπτύχθηκε ένα σύστημα το οποίο θα μπορούσε να βοηθήσει στην προσέγγιση μιας λύσης σχετικά με την ανεύρεση εύκολου παρκαρίσματος. Το σύστημα που αναπτύχθηκε ανήκει στην κατηγορία των λογισμικών που βασίζονται αρκετά στην τροφοδότηση με δεδομένα από τον ανθρώπινο παράγοντα. Επίσης θα αναλυθούν και θα αξιολογηθούν υπάρχοντα συστήματα που εξυπηρετούν παρόμοιους σκοπούς αλλά όμως λειτουργούν με διαφορετικό τρόπο.

Αφού εξηγήσαμε τον στόχο αυτής της εργασίας παρακάτω θα παρουσιάσουμε την δομή που θα ακολουθήσουμε.

- **Κεφάλαιο 2ο,** θα αναλύσουμε τα διάφορα είδη έξυπνων συστημάτων πάρκινγκ που υπάρχουν και στη συνέχεια θα αναλυθούν οι διάφορες λύσεις που υπάρχουν στην αγορά και ασχολούνται με το εύκολο παρκάρισμα. Αργότερα θα δούμε τι παραπάνω προσφέρει η λύση της παρούσας εργασίας σε σχέση με αυτές που αναλύθηκαν
- **Κεφάλαιο 3ο,** θα δούμε τους στόχους ολοκλήρωσης της εφαρμογής, την αρχιτεκτονική του συστήματος που θέλουμε να υλοποιήσουμε και τι εξοπλισμό θα χρειαστούμε
- **Κεφάλαιο 4ο,** θα παρουσιάσουμε λύσεις ανοιχτού λογισμικού τις οποίες μπορούμε να χρησιμοποιήσουμε στην εργασία μας, apis που μπορεί να μας φανούν χρήσιμα, frameworks, τα διάφορα εργαλεία καθώς και τις γλώσσες προγραμματισμού
- **Κεφάλαιο 5ο,** θα δούμε την λειτουργικότητα της εφαρμογής. Παρουσιάζονται οι πιο βασικές λειτουργίες της εφαρμογής, οι λειτουργίες του web service που έχουμε, καθώς και το σχεσιακό μοντέλο, το μοντέλο οντοτήτων συσχετίσεων, οι περιπτώσεις χρήσης και τα διαγράμματα ακολουθίας μηνυμάτων
- **Κεφάλαιο 6ο,** θα αναλύσουμε ποιες τεχνολογίες,apis, εργαλεία,frameworks χρησιμοποιήσαμε τελικά από αυτά που είδαμε στο κεφάλαιο 4 για να δημιουργήσουμε την εφαρμογή μας. Ακόμη θα δείξουμε τα plugins που χρησιμοποιήσαμε για το symfony [13], τα libs του android [11] client καθώς και τα modules του python [31] client. Επίσης θα αναφέρουμε και το hardware που χρησιμοποιήσαμε. Θα παρουσιαστεί το interface της εφαρμογής καθώς και κάποια σενάρια χρήσης, η εγκατάσταση του android emulator, η εγκατάσταση του gps receiver στο linux σύστημα μας και τέλος το εργαλείο soap ui για να δείξουμε πώς εκτελούνται οι λειτουργίες του web service μας
- **Κεφάλαιο 7ο,** θα παρουσιάσουμε τα πλεονεκτήματα και τα μειονεκτήματα της εφαρμογής που αναπτύχθηκε, όπως αυτά προέκυψαν από την χρήση και τη δημιουργία της. Τέλος θα παρουσιάσουμε τα σχέδια μας για μελλοντικές αναβαθμίσεις της υλοποίησης μας
- **Κεφάλαιο 8ο,** θα εκφράσουμε τα συμπεράσματα που προέκυψαν

2. Βιβλιογραφική Επισκόπηση

Σε αυτό το κεφάλαιο θα δούμε στο πρώτο τμήμα μια γενική αναφορά στα είδη έξυπνων συστημάτων πάρκινγκ που υπάρχουν, και στη συνέχεια συστήματα που προέκυψαν από έρευνες που διεξήχθησαν. Κάθε έρευνα προσεγγίζει το θέμα από την δική της μεριά, και προσπαθεί με τον δικό της τρόπο να δώσει λύση στο πρόβλημα του παρκαρίσματος. Στο τέλος αυτού του κεφαλαίου θα παρουσιάσουμε την διαφορετικότητα της λύσης μας.

2.1 Είδη έξυπνων συστημάτων παρκαρίσματος

Η Ευρώπη, η Μεγάλη Βρετανία και η Ιαπωνία αποτελούν τους πρωτοπόρους στην κατασκευή έξυπνων συστημάτων παρκαρίσματος. Σήμερα μπορούμε να βρούμε αρκετές εγκαταστάσεις έξυπνων συστημάτων παρκαρίσματος σε όλες τις μεγάλες πόλεις. Η τεχνολογία του έξυπνου παρκαρίσματος έχει πλεονεκτήματα και για τις δυο πλευρές, τα οποία και περιγράφουμε παρακάτω:

- Ο πελάτης μπορεί να δει τη διαθεσιμότητα σε θέσεις ενός πάρκινγκ προτού εισέλθει σε αυτό, καθώς και για τους ορόφους του
- Ο πελάτης μπορεί χρησιμοποιήσει τα μέσα μαζικής μεταφοράς για την μεταφορά του καθώς τέτοιου είδους συστήματα βρίσκονται κοντά σε γραμμές τρένων ή διαδρομές λεωφορείων
- Ο διαχειριστής του πάρκινγκ μπορεί αναλύοντας τα στατιστικά που έχει συγκεντρώσει να αλλάξει την τιμολογιακή του πολιτική
- Ο διαχειριστής με την χρήση των δεδομένων μπορεί να προβλέψει μελλοντικά μοτίβα παρκαρίσματος ή τάσεις που θα προκύψουν
- Ο διαχειριστής μπορεί να αποτρέψει κλοπές αυτοκινήτων
- Ο διαχειριστής μπορεί να κάνει περικοπές στο προσωπικό του πάρκινγκ, που ασχολείται με την διαχείριση της κυκλοφορίας
- Το σύστημα οδηγεί στην μείωση της κίνησης καθώς και των εκπομπών που γίνονται από τα αυτοκίνητα, μειώνοντας τον χρόνο αναζήτησης θέσεων ή αναμονής μέχρι να ελευθερωθεί κάποια

Τα έξυπνα συστήματα μπορούν να χωριστούν σε πέντε κατηγορίες, οι οποίες περιγράφονται παρακάτω:

2.1.1 Κατευθυντήρια πληροφοριακά συστήματα πάρκινγκ (PGI)

Η PGI τεχνολογία [5] κατευθύνει και παρέχει πληροφορίες για τη διαθεσιμότητα των θέσεων πάρκινγκ που βρίσκονται στις μεγάλες πόλεις. Αισθητήρες οχημάτων είναι εγκατεστημένοι στις εισόδους, εξόδους και στους διάφορους χώρους του πάρκινγκ για να μπορούν να υπολογίζουν τις διαθέσιμες καθώς και τις μη διαθέσιμες θέσεις. Οι πιο κοινοί αισθητήρες είναι οι αισθητήρες επανάληψης, οι τεχνητής οράσεως, οι υπερηχητικοί, οι υπέρυθροι, οι ηλεκτρομαγνητικού κύματος και οι λέιζερ. Η πληροφορία που μας παρέχει ποικίλει από άδειο ή γεμάτο, τον αριθμό των διαθέσιμων θέσεων ή την τοποθεσία στην οποία βρίσκονται οι ελεύθερες θέσεις, και η οποία πρέπει να εμφανίζεται σε διάφορα σημεία για να μπορεί ο χρήστης να πάρει την καλύτερη δυνατή απόφαση.

2.1.2 Συστήματα βασισμένα στα μεταφορικά συστήματα

Τα συστήματα [5] αυτά παρέχουν πληροφορίες για τις θέσεις πάρκινγκ, καθώς και τα δρομολόγια των μέσων μαζικής μεταφοράς. Ο βασικός του στόχος είναι να ενθαρρύνει τους οδηγούς να παρκάρουν τα οχήματα τους και να χρησιμοποιήσουν τα τρένα ή τα λεωφορεία για την μεταφορά τους. Το οποίο με την σειρά του έχει ως αποτέλεσμα την μείωση της κίνησης στους δρόμους, την μόλυνση του περιβάλλοντος και τέλος την κατανάλωση βενζίνης. Και εδώ χρησιμοποιούνται οι αισθητήρες οχημάτων ακριβώς με τον ίδιο τρόπο που χρησιμοποιούνται στην PGI τεχνολογία [5]. Τα μηνύματα που προβάλλονται στις πινακίδες των δρόμων, κατευθύνουν τους οδηγούς προς αυτούς τους χώρους.

2.1.3 Έξυπνα συστήματα πληρωμής

Τα έξυπνα συστήματα πληρωμής [5] χρησιμοποιούν προηγμένες τεχνολογίες, και αποτελούν τους αντικαταστάτες των παραδοσιακών μετρητών. Επιτρέπουν γρήγορη και άνετη πληρωμή, βελτιώνουν τα ποσοστά συλλογής των προστίμων και μειώνουν τα ποσοστά επίθεσης σε υπαλλήλους του πάρκινγκ. Οι τεχνολογίες που χρησιμοποιούνται περιλαμβάνουν μεθόδους επικοινωνίας (χρωστικές, πιστωτικές κάρτες), μεθόδους που δεν περιέχουν κάποιο είδος επικοινωνίας (έξυπνες κάρτες και RFID) και τέλος μεθόδους επικοινωνίας που περιέχουν κινητές συσκευές (υπηρεσίες κινητής τηλεφωνίας). Με την χρήση της πληρωμής για μια θέση πάρκινγκ μπορούμε να υπολογίσουμε ποιες θέσεις καταλαμβάνονται εκείνη την στιγμή. Σε αυτό το σύστημα δεν χρησιμοποιούνται αισθητήρες.

2.1.4 Ε-Πάρκινγκ

Το Ε-Πάρκινγκ [5] χρησιμοποιεί προηγμένες τεχνολογίες για να ενοποιήσει την λειτουργία δέσμευσης θέσεων πάρκινγκ με τις λειτουργίες των συστημάτων πληρωμής. Χρησιμοποιώντας αυτό το σύστημα ο χρήστης μπορεί να ρωτήσει για τη διαθεσιμότητα, να κάνει κράτηση για μια θέση την οποία μπορεί να πληρώσει φεύγοντας. Σε αυτό το σύστημα μπορούμε να έχουμε πρόσβαση από το ίντερνετ, pdas ή το κινητό μας τηλέφωνο. Και σε αυτό το σύστημα χρησιμοποιούνται οι κλασσικοί αισθητήρες οι οποίοι χρειάζονται για να αναγνωρίσουν οχήματα τα οποία πλησιάζουν. Ωστόσο το σύστημα θα πρέπει να αναγνωρίζει τους πελάτες ή τα οχήματα που έχουν κάνει την κράτηση για να τους επιτρέπει την είσοδο στην θέση που έχουν δεσμεύσει. Μπορούμε χρησιμοποιώντας κωδικό επιβεβαίωσης να αναγνωρίσουμε έναν πελάτη, τον οποίο ο χρήστης έχει λάβει στο κινητό του.

2.1.5 Αυτόματο Πάρκινγκ

Πρόκειται για ένα αυτόματο σύστημα μηχανικού πάρκινγκ [5], του οποίου η διαχείριση γίνεται από υπολογιστή. Οι πελάτες μπορούν να κατευθύνουν το αμάξι τους στις υπάρχουσες αποβάθρες, να το κλειδώσουν και ο υπολογιστής αναλαμβάνει να παρκάρει το αυτοκίνητο τους. Για να παραλάβουν το αμάξι τους χρειάζεται να πληκτρολογήσουν τον κωδικό τους, καθώς και το password τους. Εφόσον επιβεβαιωθούν τα στοιχεία τους, παραλαμβάνουν το αυτοκίνητο τους. Το αυτόματο πάρκινγκ επιτρέπει την αποδοτική χρησιμοποίηση ενός πάρκινγκ με λίγες θέσεις. Σε αυτό το σύστημα βρίσκουμε αρκετά είδη αισθητήρων.

2.2 Έρευνες και προϊόντα

Στα κεφάλαια που ακολουθούν θα γίνει αναφορά σε διάφορα προϊόντα που κυκλοφορούν ήδη στην αγορά, και αναπτύχθηκαν είτε για πιλοτικά projects είτε για έρευνες που διεξήχθησαν. Τα συστήματα που θα αναλύσουμε χρησιμοποιούνται σε χώρους πάρκινγκ επί πληρωμή.

2.2.1 Navigon

Μία από τις εταιρείες που προσφέρουν συστήματα έξυπνου παρκαρίσματος είναι η Nanigon [27]. Στον χάρτη της εφαρμογής εμφανίζονται μπλε σήματα P (το σήμα του parking). Ο χρήστης μπορεί να επιλέξει κάποιο στίγμα parking από την οθόνη της συσκευής πλοήγησης του και να δει πληροφορίες που αφορούν το συγκεκριμένο parking ή επιλέγοντας από την οθόνη να δει τα διαθέσιμα parkings της περιοχής στην οποία βρίσκεται. Αποτελεί μια ακριβή λύση, καθώς μια θέση parking μπορεί να ξεκινήσει από πέντε ευρώ και να φτάσει έως και δεκαπέντε ευρώ. Η λύση μας σε σχέση με το προϊόν της Navigon σίγουρα προσφέρει λιγότερες λειτουργίες, καθώς τα προϊόντα της Navigon δημιουργούνται για εμπορική χρήση. Θεωρούμε ότι η λύση της Navigon καλύπτει εν μέρη αυτό που θέλουμε να επιτύχουμε, δηλαδή το έξυπνο παρκάρισμα σε χαμηλό κόστος. Ωστόσο δεν μπορούμε να χρησιμοποιήσουμε κάποιο μέρος του κώδικα της Navigon καθώς αποτελεί εμπορικό κλειστό προϊόν στου οποίου τον κώδικα δεν έχουμε πρόσβαση.

Στην εικόνα 1 βλέπουμε το interface της συσκευής, στο οποίο φαίνεται η διαδρομή που ακολουθεί ο χρήστης. Ακόμα στην οθόνη εμφανίζεται η επιλογή που μας δείχνει τα διαθέσιμα parkings στην περιοχή, αν υπάρχει κάποιο το οποίο βρίσκεται κοντά εμφανίζεται ως στίγμα.

Εικόνα 1: Βλέπουμε το gui του προϊόντος, στο οποίο φαίνεται ο χάρτης της εφαρμογής [27]

Παρακάτω βλέπουμε την εικόνα 2, στην οποία ο χρήστης ψάχνει ανάμεσα στα διαθέσιμα parkings που υπάρχουν στην περιοχή. Δίπλα από κάθε parking υπάρχει αξιολόγηση και μπορούμε επίσης να δούμε

και το κόστος ανά ώρα. Επιλέγοντας από τις διαθέσιμες επιλογές ο χρήστης μπορεί να δει περαιτέρω πληροφορίες για την επιλογή του.

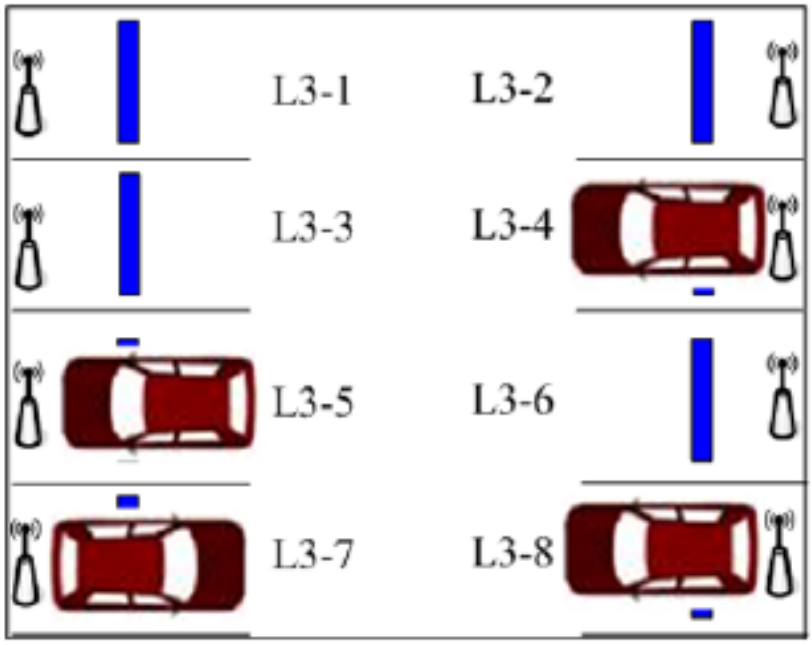
Εικόνα 2: Βλέπουμε αναλυτικές πληροφορίες για τα πάρκινγκ που βρίσκονται στην περιοχή [27]

2.2.2 SmartParking: Ένα ασφαλές και έξυπνο σύστημα παρκαρίσματος

Το SmartParking [1] προτάθηκε από τους Gongjun Yan, Stephan Olariu, Michele C. Weigle, Mahmoud Abuelela σε μια σύσκεψη της IEEE στην Beijing, Κίνα τον Οκτώβριο του 2008. Αποτελεί μια ολοκληρωμένη πολύπλοκη λύση, καθώς εκτός από το λογισμικό χρησιμοποιεί και αισθητήρες, πομποδέκτες για να δουλέψει.

Ο οδηγός που χρησιμοποιεί αυτή την υπηρεσία χρειάζεται να έχει μαζί του έναν ασύρματο πομποδέκτη - μικρό επεξεργαστή, του οποίου η εκπομπή φτάνει το ένα μέτρο. Το parking που θέλει να προσφέρει τις υπηρεσίες του με την χρήση της συγκεκριμένης λύσης πρέπει να έχει και αυτό έναν ασύρματο πομποδέκτη ή ασύρματο δίκτυο, αισθητήριες ζώνες παρκαρίσματος, συσκευές υπερύθρων και έναν κεντρικό υπολογιστή που να διαχειρίζεται τα αιτήματα που γίνονται στο σύστημα μας. Η μετάδοση μηνυμάτων μας παρέχει πληροφορίες για την κατάσταση του συστήματος:

- Τις συνολικές θέσεις του parking
- Ποιες από αυτές είναι δεσμευμένες και ποιες όχι
- Τις κρατήσεις για θέσεις που έχουν γίνει

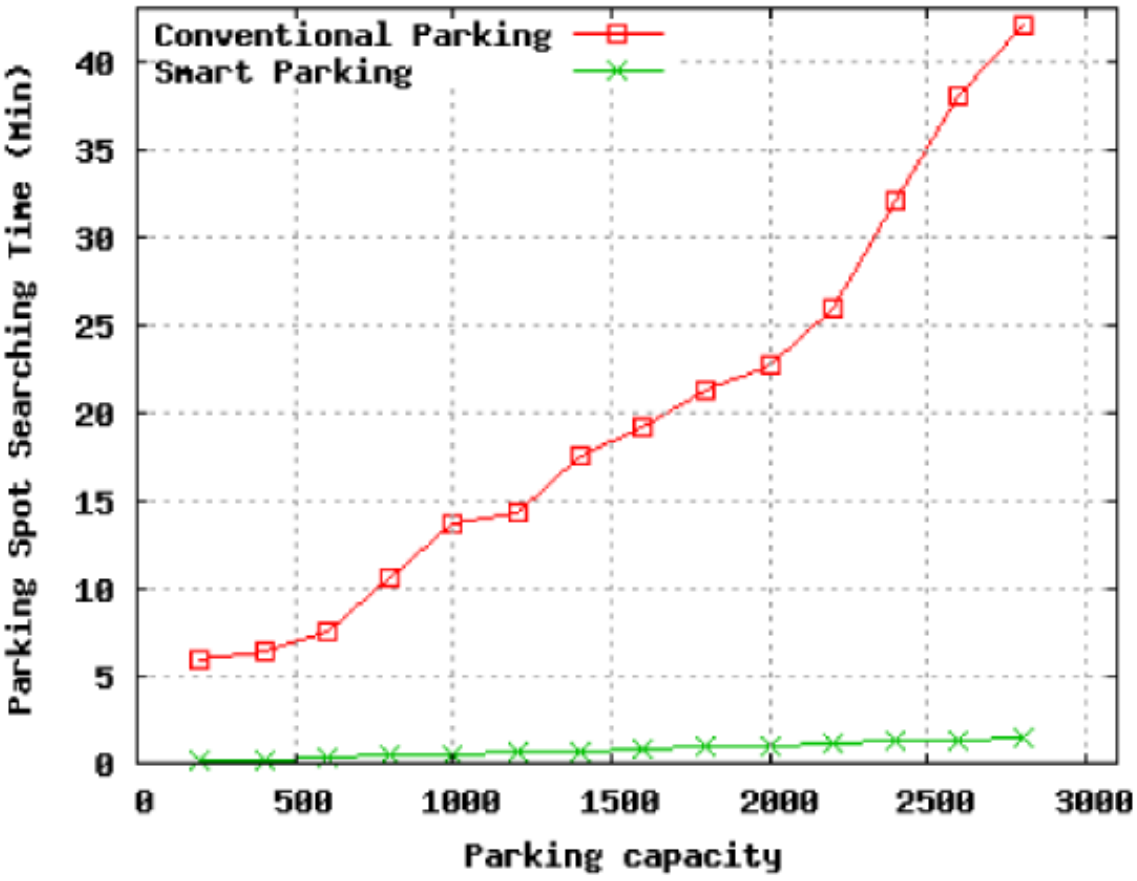


Εικόνα 3: Η αρχιτεκτονική της εφαρμογής [1]

Οι αισθητήριες ζώνες σε συνδυασμό με τις υπέρυθρες συσκευές είναι αυτές που αναγνωρίζουν εάν μια θέση καταλαμβάνεται από κάποιο αμάξι ή όχι, και φροντίζουν να τροφοδοτήσουν το δίκτυο με αυτή την πληροφορία. Αυτό γίνεται για να μειώσουμε όσο το δυνατόν γίνεται τα λάθη ανίχνευσης μιας θέσης. Σε αυτούς οι οποίοι δεν ακολουθούν στους κανόνες του συστήματος ή κάνουν λάθη στο παρκάρισμα τους (καταλαμβάνουν δύο θέσεις πάρκινγκ), επιβάλλονται οικονομικές κυρώσεις

Το σύστημα μπορεί να στέλνει μηνύματα από το αμάξι προς μια ζώνη, από ζώνη σε ζώνη και είτε και από αμάξι σε αμάξι, και έτσι να ενημερώνονται οι χρήστες για όλες τις πιθανές θέσεις που μπορούν να δεσμεύσουν. Ακόμα έχουν τη δυνατότητα, εάν αυτοί το θελήσουν, να δεσμεύσουν μια θέση (δηλώνουν τη θέση που θέλουν και το χρονικό διάστημα παραμονής). Οι χρήστες μπορούν να ακυρώσουν την κράτηση που έκαναν για κάποια θέση. Σε περίπτωση που δεν καταλάβουν μια θέση την οποία έχουν κρατήσει και δεν έχουν ακυρώσει την κράτηση μπορεί να τους επιβληθεί κάποιο πρόστιμο. Οι χρήστες απαγορεύεται να καταλάβουν άλλη θέση από αυτή που δέσμευσαν για οποιοδήποτε λόγο. Μια θέση η οποία δεν είναι δεσμευμένη εμφανίζεται με ένα μπλε σημείο (με κίτρινο εμφανίζονται οι δεσμευμένες και με κόκκινο οι θέσεις στις οποίες έχει γίνει κάποιο λάθος από την μεριά του χρήστη).

Παρακάτω βλέπουμε τον χρόνο στον οποίο ο οδηγός ψάχνει για parking, με την χρήση του συστήματος και χωρίς. Όσο αυξάνονται οι θέσεις αυξάνεται και ο χρόνος κατά τον οποίο κάποιος ψάχνει για parking. Παρατηρούμε ότι σε πάρκινγκ 2500 – 3000 θέσεων κάποιος μπορεί να ψάχνει για μια θέση πάρκινγκ έως και σαράντα λεπτά, σε αντίθεση με κάποιον που χρησιμοποιεί το σύστημα που δεν χρειάζεται να ψάξει πάνω από ένα λεπτό.



Εικόνα 4: Χρόνος εύρεσης παρκαρίσματος [1]

Πρόκειται για μια αξιόλογη λύση, η οποία μπορεί και ενημερώνει τον χρήστη για τις ελεύθερες θέσεις που υπάρχουν και που μπορεί να κρατήσει, εφόσον αυτός το επιθυμεί. Αυτή η λύση είναι διαφορετική σε σχέση με την δική μας, καθώς δίνεται η δυνατότητα στον χρήστη να κάνει κράτηση για μια θέση. Παρουσιάζει το ίδιο μειονέκτημα με το προηγούμενο σύστημα, εφαρμόζεται σε θέσεις πάρκινγκ επί πληρωμή και είναι πολύ ακριβό όσον αφορά την εγκατάσταση και την συντήρησή του.

2.2.3 Έξυπνο σύστημα παρκαρίσματος στον διεθνή αερολιμένα Thurgood Marshall στην Βαλτιμόρη

Το αεροδρόμιο Thurgood Marshall αποτελεί το πρώτο αεροδρόμιο της χώρας που αποφάσισε να εγκαταστήσει ένα έξυπνο σύστημα παρκαρίσματος [28] στο πάρκινγκ που παρέχει στους πελάτες του, συνολικής χωρητικότητας 13.200 θέσεων. Χρησιμοποιεί αρκετά είδη σένσορων, οι οποίοι ελέγχουν για το αν μία θέση είναι ελεύθερη ή όχι.

Οι ηλεκτρονικές πινακίδες που υπάρχουν κοντά στο αεροδρόμιο ενημερώνουν τους οδηγούς για τη

διαθεσιμότητα αλλά και την κατεύθυνση στην οποία μπορούν να πάνε για να βρουν ελεύθερες θέσεις. Εμφανίζεται το σύνολο των ελεύθερων θέσεων στην είσοδο αλλά και ο αριθμός αυτών ανά επίπεδο. Σε κάθε επίπεδο υπάρχουν πινακίδες που ενημερώνουν τους οδηγούς για την διαθεσιμότητα των θέσεων ανά γραμμή. Τέλος, υπάρχει φωτεινή ένδειξη που δηλώνει τη διαθεσιμότητα μιας θέσης, πράσινο για μια διαθέσιμη θέση και κόκκινο για μια μη διαθέσιμη.

Το συγκεκριμένο σύστημα επιτρέπει στους υπεύθυνους του αεροδρομίου να έχουν ακριβή εικόνα για όλες τις θέσεις παρκαρίσματος που διαθέτουν. Όπως αναφέρει ο υπεύθυνος, πάρκινγκ που δεν χρησιμοποιούν ένα έξυπνο σύστημα παρκαρίσματος αναγκάζονται να μην παρέχουν άλλο τις υπηρεσίες τους προσωρινά όταν είναι γεμάτα κατά 70% με 80%, γιατί θεωρούν ότι έχουν γεμίσει πλήρως. Με το συγκεκριμένο σύστημα το αεροδρόμιο μπορεί να παρέχει συνέχεια το 100% της χωρητικότητας του. Το κόστος εγκατάστασης των αισθητήρων ανά θέση είναι 450 δολάρια, το οποίο δεν είναι αρκετά μεγάλο αν αναλογιστεί κανείς το οικονομικό όφελος που μπορεί να έχει.

Στην εικόνα 5 οποία φαίνεται η φωτεινή ένδειξη η οποία μας ενημερώνει για την κατεύθυνση που μπορεί να πάει ο οδηγός για να βρει ελεύθερες θέσεις.

Εικόνα 5: κατευθυντήριες οδηγίες προς ελεύθερες θέσεις πάρκινγκ [28]

Ενδιαφέρουσα λύση αν και πρόκειται για parking επί πληρωμή το οποίο χρησιμοποιείται από το αεροδρόμιο. Επομένως δεν μπορούμε να επωφεληθούμε κάπως από τη συγκεκριμένη υλοποίηση.

2.2.4 Πρωτότυπο έξυπνο παρκάρισμα με την χρήση δικτυακών ασύρματων αισθητήρων (SPARK)

Το συγκεκριμένο σύστημα [4] προτάθηκε από τους S. V. Srikanth, Pramod P. J, Dileep K. P, Tapas S, Mahesh U. Patil, Sarat Chandra Babu N του C-DAC,οι οποίοι προσπάθησαν να δημιουργήσουν ένα αυτοματοποιημένο, οικονομικό, ενημερωμένο σε πραγματικό χρόνο και εύκολο στη χρήση σύστημα. Αποφεύγουν να χρησιμοποιήσουν τους inductive loop αισθητήρες, οι οποίοι παρουσιάζουν προβλήματα συντήρησης, λειτουργίας και χρησιμοποιούν ένα ασύρματο δίκτυο αισθητήρων (WSN).

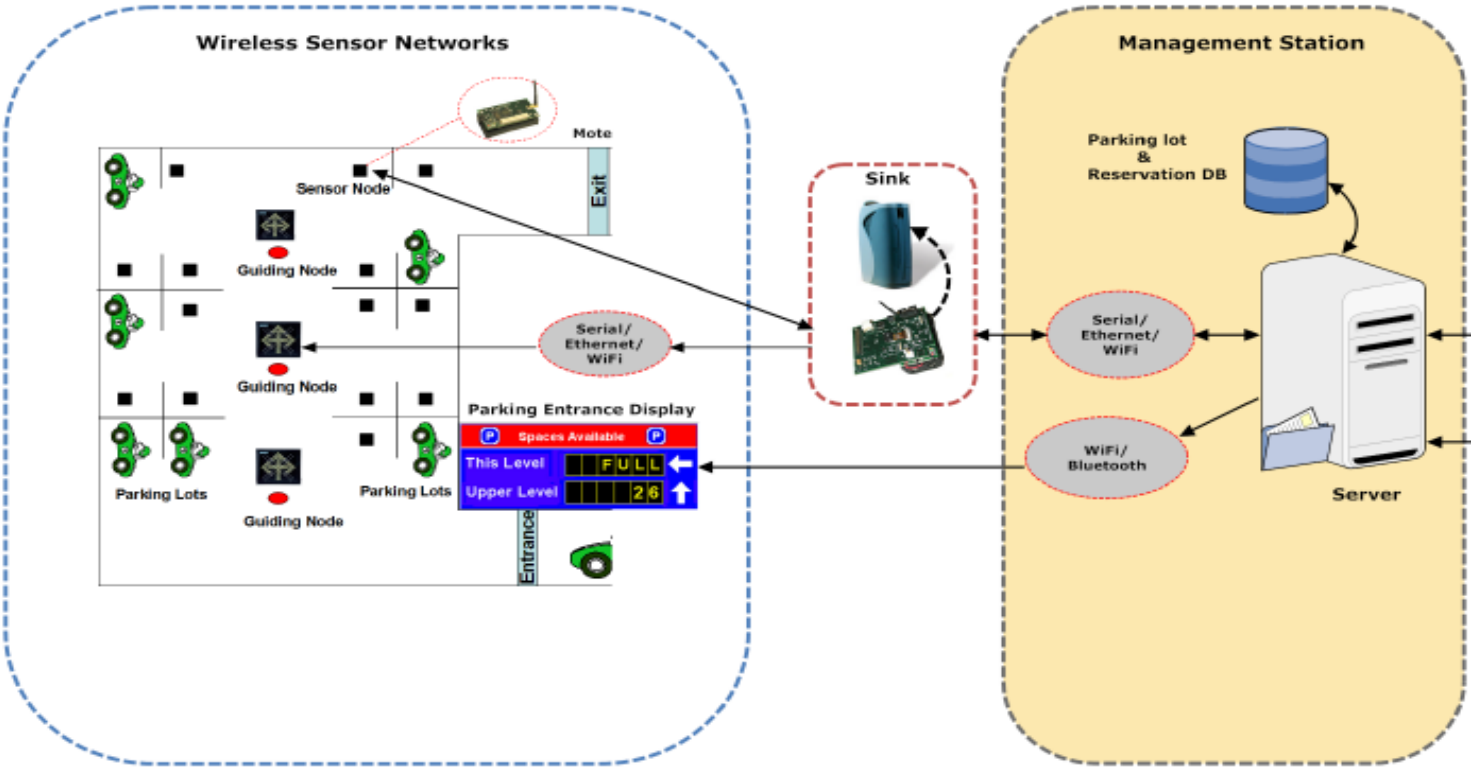
Το σύστημα αυτό παρέχει τις εξής υπηρεσίες, παρακολούθηση και διαχείριση κάθε θέσης ξεχωριστά,

παρέχει οδηγίες στους οδηγούς για να μπορέσουν να κατευθυνθούν στις ελεύθερες θέσεις και τέλος τους δίνει τη δυνατότητα να επιλέξουν μια θέση μέσω του κινητού τους και να την κρατήσουν εφόσον αυτοί το επιθυμούν. Για να δεσμεύσουν μια θέση χρειάζεται να δώσουν τον αριθμό της πινακίδας τους, καθώς και το χρονικό διάστημα για το οποίο την χρειάζονται (τα μηνύματα που στέλνονται μεταφέρονται μέσω του GSM/GPRS δικτύου). Τα μηνύματα μεταφέρονται στον server του συστήματος, ο οποίος απαντάει στον χρήστη αν το αίτημα του έγινε δεκτό.

Το σύστημα είναι χωρισμένο σε υποσυστήματα, τα οποία καλύπτουν μια ξεχωριστή λειτουργικότητα το καθένα.

- **Υποσύστημα WSN,** ελέγχει τις θέσεις του συστήματος για τυχόν αλλαγές, τις οποίες και στέλνει μέσω του RF
- **Υποσύστημα Sink,** αποτελεί το gateway που μεταφέρει πληροφορίες από το WSN στο εξωτερικό δίκτυο (κυρίως στο υποσύστημα διαχείρισης παρκαρίσματος). Επίσης μεταφέρει πληροφορίες για την αλλαγή της κατάστασης των θέσεων από το υποσύστημα διαχείρισης στο υποσύστημα καθοδήγησης
- **Υποσύστημα διαχείρισης parking,** εδώ καταλήγουν όλα τα μηνύματα τα οποία και αποθηκεύονται στη βάση (αποτελεί την καρδιά του συστήματος). Είναι υπεύθυνο είτε για την κράτηση των θέσεων είτε για να δώσει σήμα στο υποσύστημα καθοδήγησης . Ακόμα λαμβάνει πληροφορίες για τη διαθεσιμότητα των θέσεων τις οποίες και εμφανίζει στο interface του πελάτη. Τέλος ενημερώνει το υποσύστημα εισόδου, το οποίο είναι αυτό που εμφανίζει τις διαθέσιμες θέσεις που υπάρχουν στο πάρκινγκ
- **Αυτόματο υποσύστημα καθοδήγησης,** εμφανίζει πληροφορίες για να κατευθυνθεί ο οδηγός εύκολα σε μια μη κατειλημμένη θέση, όπως και πληροφορίες κατεύθυνσης για να εντοπίσει ελεύθερες θέσεις
- **Υποσύστημα ενημέρωσης εισόδου**, ενημερώνει τους χρήστες σχετικά με τη διαθεσιμότητα των θέσεων που διαθέτει το parking
- **Υποσύστημα client,** αποτελεί το υποσύστημα που διαχειρίζεται ο χρήστης και μέσω αυτού μπορεί να δει τις διαθέσιμες θέσεις και να κρατήσει κάποια εφόσον αυτός το επιθυμεί. Για να κρατήσει κάποια θέση χρειάζεται να δώσει τον αριθμό πινακίδας του, καθώς και το χρονικό διάστημα για το οποίο επιθυμεί να δεσμεύσει τη θέση

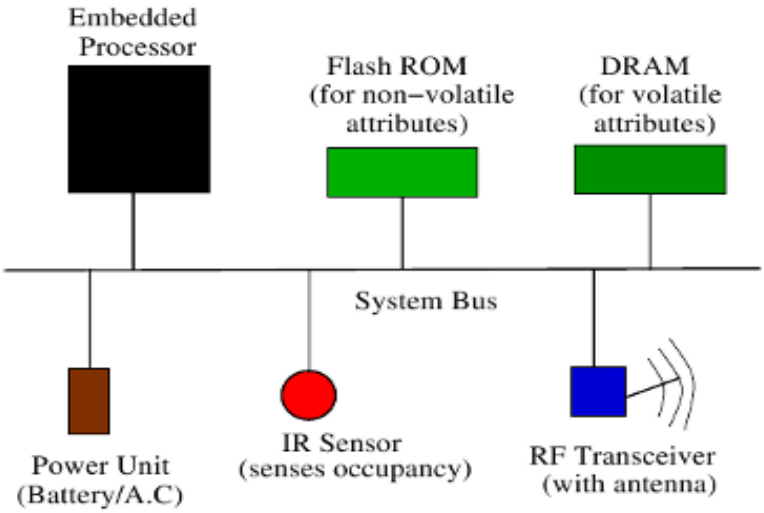
Στην εικόνα 6 φαίνεται η αρχιτεκτονική του συστήματος, όπως αυτή έχει προταθεί. Πρόκειται για μια ενδιαφέρουσα λύση, αλλά και αυτή βασίζεται σε χώρους parking επί πληρωμή.



Εικόνα 6: Αρχιτεκτονική συστήματος SPARK [4]

2.2.5 Δικτυωμένες Θέσεις Παρκαρίσματος

Οι δικτυωμένες θέσεις παρκαρίσματος [2] προτάθηκαν από τους Prithwish Basu και Thomas D.C. Little του πανεπιστημίου της Βοστώνης. Ο χρήστης μπορεί να βρει τη θέση παρκαρίσματος που εκείνος πραγματικά επιθυμεί, εκτελώντας ερωτήματα μέσω του κινητού του. Για παράδειγμα ο χρήστης επιθυμεί να βρει μια θέση που βρίσκεται 50 μέτρα μακριά του και κοστίζει 2 ευρώ, εφόσον εντοπιστεί αυτή η θέση και αυτός το επιθυμεί μπορεί να την κρατήσει. Το σύστημα υποστηρίζει διάφορα ερωτήματα, τα οποία μπορεί να εκτελέσει ο χρήστης όσον αφορά τις θέσεις.



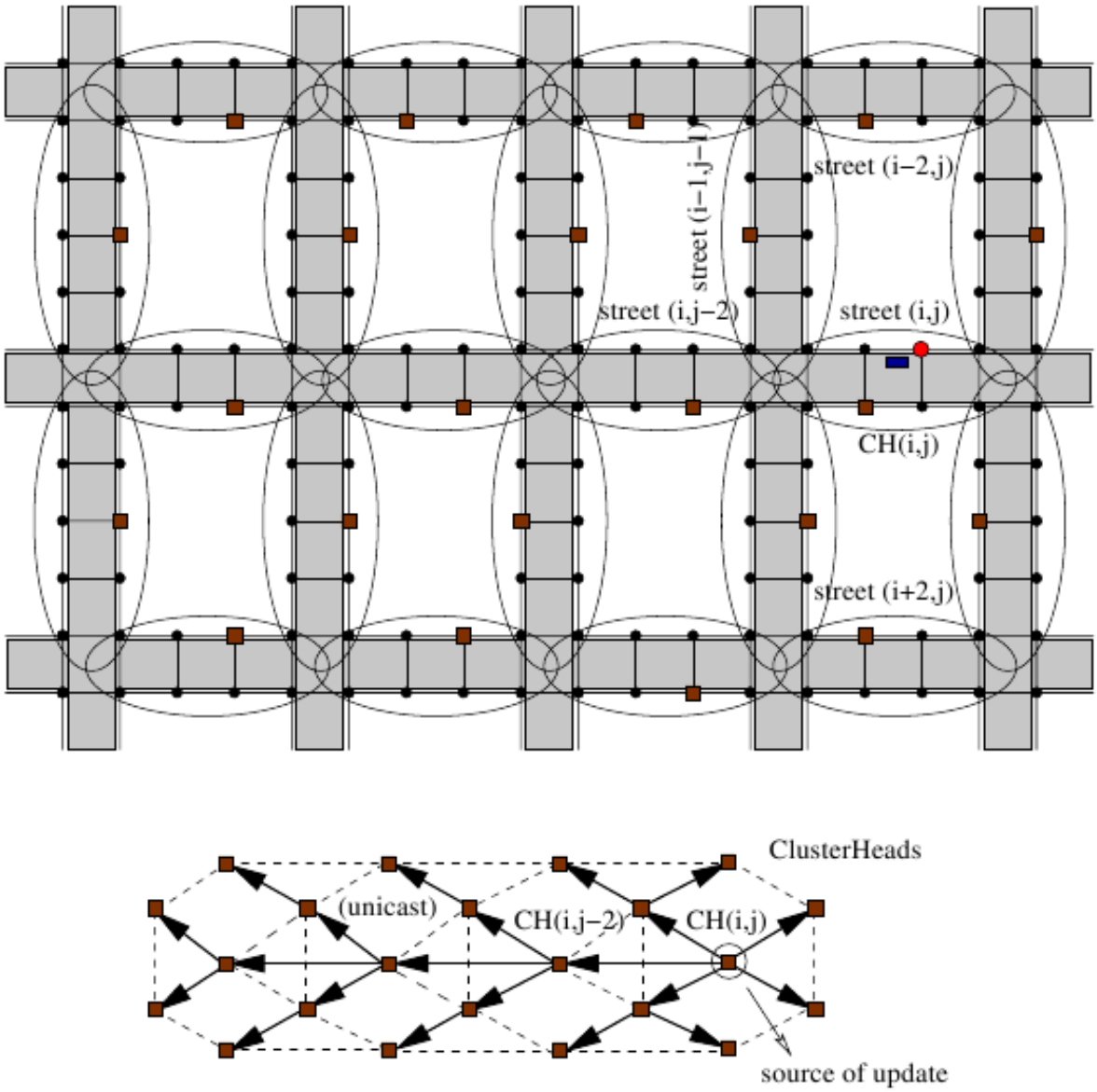
Εικόνα 7: Αρχιτεκτονική αισθητήρα – μετρητή [2]

Ο βασικός κορμός του δικτύου είναι ένα ad hoc δίκτυο. Το συγκεκριμένο δίκτυο αποτελείται από αυτόνομες μονάδες – μετρητές, οι οποίες επικοινωνούν μεταξύ τους για να δώσουν απάντηση στα

ερωτήματα που υποβάλει ο χρήστης. Στην εικόνα 7 βλέπουμε ότι οι συγκεκριμένοι μετρητές διαθέτουν μπαταρία, επεξεργαστή, μνήμη (εγγράψιμη και μιας εγγραφής), υπέρυθρο αισθητήρα (αναγνωρίζει αν η θέση είναι κατειλημμένη) και ασύρματη κάρτα για να μπορούν να επικοινωνούν μεταξύ τους. Ο κάθε μετρητής διαθέτει διάφορες πληροφορίες, από τις οποίες άλλες είναι σταθερές και άλλες αλλάζουν. Για παράδειγμα, η τοποθεσία του μετρητή είναι μη μεταβαλλόμενη πληροφορία, ενώ η μέτρηση της στάθμης της μπαταρίας του είναι είναι μια μεταβαλλόμενη πληροφορία.

Σε ένα αρχικό σενάριο όλοι οι μετρητές επικοινωνούν μεταξύ τους και ο κάθε μετρητής γνωρίζει τι γίνεται με την κατάσταση των υπολοίπων έτσι ώστε να μπορεί να απαντήσει σε ερωτήματα του χρήστη για εύρεση ελεύθερων θέσεων. Παρ' όλα αυτά, έτσι θα σπαταλιόνταν αρκετοί πόροι του συστήματος που δεν χρειάζεται να γίνεται. Για να μην υπάρχει συνεχής επικοινωνία μεταξύ των μετρητών, αποφάσισαν να χωρίσουν τους μετρητές - αισθητήρες σε ομάδες (Clusters). Βασικό πλεονέκτημα των συγκεκριμένων μετρητών - αισθητήρων είναι ότι το δίκτυο δεν αλλάζει, καθώς οι αισθητήρες είναι σταθεροί (μόνο σε δύο περιπτώσεις μπορεί να αλλάξει, είτε να σταματήσει να είναι διαθέσιμος ένας αισθητήρας, είτε να εγκαταστήσουμε έναν καινούριο).

Στην εικόνα 8 που ακολουθεί, βλέπουμε πως ενώνονται οι αισθητήρες και δημιουργούν ομάδες (Clusters). Η κάθε ομάδα διαθέτει μια κεφαλή (Cluster Head), η οποία είναι υπεύθυνη για να γνωρίζει τι γίνεται με τους αισθητήρες της δικιάς της ομάδας. Η επιλογή της κεφαλής γίνεται τυχαία, αυτό που μπορούμε να ελέγξουμε είναι η υπολογιστική ισχύ που καταναλώνει σε συνδυασμό με το υπόλοιπο της μπαταρίας που διαθέτει. Σε περίπτωση που λάβουν κάποιο ερώτημα, αυτές είτε απαντούν με βάση κάποιο μήνυμα που είχαν λάβει πιο πριν, είτε χρειάζεται να ρωτήσουν τις άλλες κεφαλές μέσα στο δίκτυο για να πάρουν την πληροφορία που θέλουν. Επίσης όταν κάποιος αισθητήρας (δεν χρειάζεται να είναι κεφαλή) παραλάβει κάποιο μήνυμα, αυτός αναλαμβάνει να το δρομολογήσει στην κεφαλή της ομάδας του, η οποία με τη σειρά της είτε το κρατάει, αν αφορά την δική της ομάδα, είτε το προωθεί στην κεφαλή που πρέπει.



Εικόνα 8: Αρχιτεκτονική δικτύου [2]

Βασικό χαρακτηριστικό που πρέπει να ληφθεί υπόψιν, είναι ότι ο χρήστης βρίσκεται σε συνεχή κίνηση . Όπως είπαμε οι μετρητές - αισθητήρες έχουν πληροφορίες μόνο για την δική τους ομάδα. Αυτό σημαίνει ότι όταν ο χρήστης ρωτήσει κάτι για την περιοχή στην οποία βρίσκεται, πρέπει να λάβουμε υπόψιν ότι η απάντηση που θα στείλει το σύστημα στον χρήστη, εάν αυτή καθυστερήσει μπορεί να μην του είναι χρήσιμη.

Ακόμα το σύστημα υπολογίζει και τις συγκρούσεις ενδιαφέροντος που δημιουργούνται όταν δύο οδηγοί ενδιαφέρονται να καταλάβουν την ίδια θέση. Στην ουσία το σύστημα ενημερώνει τον χρήστη που είναι πιο μακριά, ότι η θέση για την οποία ενδιαφερόταν δεν είναι πια διαθέσιμη και έτσι αυτός

δεν χρειάζεται να κατευθυνθεί προς αυτή. Επίσης τον ενημερώνει και για άλλες διαθέσιμες θέσεις στην περιοχή. Τέλος, εάν κάποιος χρήστης παραβιάζει το χρόνο παραμονής σε μια θέση, το σύστημα τον χρεώνει με κάποιο πρόστιμο στον λογαριασμό του ανάλογα με το πόσο παραπάνω κρατάει τη θέση.

Πρόκειται για την πρώτη λύση που εντοπίσαμε και χρησιμοποιεί εξωτερικούς αισθητήρες-παρκόμετρα, καθώς όλες οι προηγούμενες λύσεις παρουσιάζουν συστήματα για κλειστούς χώρους parking. Αρκετά ενδιαφέρουσα πρόταση, καθώς οι μετρητές - αισθητήρες από μόνοι τους και χωρίς την χρήση κάποιου κεντρικού server αποτελούν το σύστημα και μπορούν να ενημερώσουν τον χρήστη για οποιαδήποτε θέση που βρίσκεται εντός του δικτύου τους. Παρ' όλα αυτά, και σε αυτή τη λύση μιλάμε για θέσεις επί πληρωμή.

2.2.6 Πιλοτικό πρόγραμμα με ασύρματους αισθητήρες στο Σαν Φρανσισκο για την παρακολούθηση των θέσεων παρκαρίσματος

Σε μια έρευνα που ξεκίνησε τον Αύγουστο του 2006 το λιμάνι του ΣΦ σε συνεργασία με το υπουργείο μεταφορών του ΣΦ εγκατέστησε δίκτυο από ασύρματους αισθητήρες [29] για να υπολογίσει ποιες θέσεις χρησιμοποιούνται και για ποιο χρονικό διάστημα.

Το σύστημα εγκαταστάθηκε σε 250 αισθητήρες γύρω από την προβλήτα των ψαράδων και το Embarcadero. Το σύστημα κατέγραφε κάθε πότε άφηνε αυτοκίνητο την θέση του και σύγκρινε τις πληροφορίες κράτησης της θέσης με τις πληρωμές που είχαν γίνει για αυτήν. Τον Ιούνιο του 2007 εγκαταστάθηκαν άλλοι 30 αισθητήρες σε θέσεις parking επί πληρωμή στο 500 block της πλατείας Colombus. Το πιλοτικό πρόγραμμα με τους 250 αισθητήρες τελείωσε τον Μάρτιο του 2007 και λίγο αργότερα τελείωσε και το πρόγραμμα για τους υπόλοιπους 30 που είχαν εγκατασταθεί.

Πρόκειται για ασύρματες συσκευές, των οποίων το μέγεθος είναι ίσο με το μέγεθος των ανακλαστήρων που βρίσκουμε συχνά στους δρόμους. Η συχνότητα στην οποία τρέχουν είναι 2.4 έως 2.5 GHz ή 902 έως 928 MHz και διαθέτουν μια μπαταρία, η οποία μπορεί να καλύψει τη συσκευή για διάστημα ενός χρόνου. Οι συσκευές στέλνουν συνέχεια τη θέση τους, στοιχεία για τη θέση πάρκινγκ στην οποία αντιστοιχούν καθώς και τον μοναδικό τους αριθμό.

Όταν ένας αισθητήρας αναγνωρίσει ότι ένα αμάξι καταλαμβάνει μια θέση, στέλνει τις πληροφορίες μέσω του δικτύου των αισθητήρων μέχρι αυτό να φτάσει σε μια gateway, που συνήθως είναι πάνω σε ένα φανάρι και η οποία με την σειρά της στέλνει τα δεδομένα αυτά σε έναν server χρησιμοποιώντας μια κινητή σύνδεση. Ακριβώς η ίδια διαδικασία γίνεται και όταν αντιληφθεί ο αισθητήρας ότι ένα αμάξι αποχωρεί από μια θέση. Οι ίδιοι αισθητήρες μπορούν να αναγνωρίσουν την κίνηση σε παρακείμενες λωρίδες και έτσι να παρέχουν συνέχεια πληροφορίες για την κίνηση που υπάρχει στους κοντινούς δρόμους.

Με την χρήση των στατιστικών αυτών, οι αρχές μπορούν να αποφασίσουν την τιμολογιακή τους πολιτική καθώς και αν θα αλλάξουν τα χρονικά όρια παρκαρίσματος . Για κάθε σένσορα χρειάζονται 300 δολάρια και για κάθε μόνιτορ του σένσορα 150. Ο υπεύθυνος της εταιρείας που το παρέχει αναφέρει ότι στα μελλοντικά τους σχέδια θέλουν να δημιουργήσουν ένα σύστημα, στο οποίο οι χρήστες θα μπορούν να σημειώνουν μέσω του gps τους τις κενές θέσεις που αυτοί εντοπίζουν.

Και αυτή η υλοποίηση βασίζεται σε εξωτερικούς μετρητές - αισθητήρες πάρκινγκ. Είναι η πρώτη φορά

όμως που εμφανίζεται η δικιά μας αρχική ιδέα (έστω και ως απλή αναφορά).

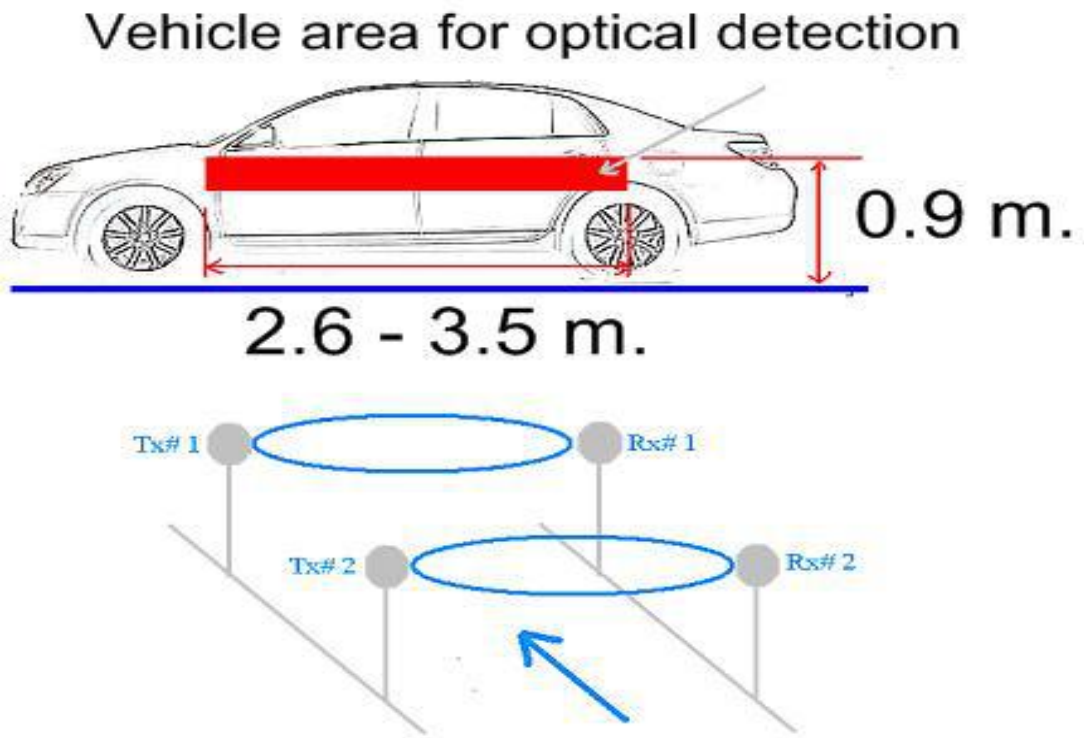
2.2.7 Έξυπνο παρκάρισμα: Μια εφαρμογή οπτικού δικτύου ασύρματων αισθητήρων

Ένα σημαντικό κομμάτι του έξυπνου παρκαρίσματος είναι οι αισθητήρες οχημάτων, με τους αισθητήρες επανάληψης να είναι αυτοί που χρησιμοποιούνται περισσότερο. Τα βασικά τους μειονεκτήματα είναι η δυσκολία εγκατάστασης τους και η συντήρησή τους, απαιτούν σκάψιμο της επιφάνειας του δρόμου που μπορεί να αποτρέψει την κανονική λειτουργία ενός υπόγειου πάρκινγκ. Για να αποφευχθούν αυτές οι δυσκολίες, οι ερευνητές προτείνουν ένα δίκτυο από ασύρματους αισθητήρες , που παρακολουθεί την διέλευση των οχημάτων. Πρόκειται για ένα απλό WSN [5], το οποίο χρησιμοποιεί την τοπολογία αστέρα. Το δίκτυο αποτελείται από έναν κεντρικό κόμβο με όλους τους υπόλοιπους να αποτελούν τους αισθητήρες κόμβους. Ο κεντρικός κόμβος είναι εγκατεστημένος σε ένα κεντρικό σημείο και λαμβάνει δεδομένα κυκλοφορίας από όλους τους υπόλοιπους κόμβους, οι οποίοι είναι εγκατεστημένοι σε διαφορετικά σημεία παρακολούθησης. Ο αισθητήριος κόμβος είναι εξοπλισμένος με έναν οπτικό αισθητήρα για να ανιχνεύει διερχόμενα οχήματα, ο οποίος μας δίνει μια ακριβή μέτρηση για τη θέση ενός οχήματος.

Χρειάστηκε η εγκατάσταση του συστήματος για να γίνεται παρακολούθηση του υπόγειου πάρκινγκ. Οι κόμβοι – αισθητήρες μπορούν εύκολα να εγκατασταθούν, καθώς δεν χρειάζονται ειδική καλωδίωση ή εκσκαφή του δρόμου. Η εγκατάσταση τους γίνεται σε όλες τις εισόδους, εξόδους καθώς και σημεία κλειδιά για τη μέτρηση οχημάτων, από τα οποία έχουμε συνεχή διέλευση οχημάτων. Οι αισθητήρες προωθούν αυτά τα δεδομένα στον κεντρικό κόμβο, ο οποίος με τη σειρά του υπολογίζει τον αριθμό των διαθέσιμων και μη διαθέσιμων θέσεων πάρκινγκ. Το σύστημα εμφανίζει τον αριθμό των ελεύθερων θέσεων ή κατευθύνει προς τοποθεσίες, στις οποίες υπάρχουν ελεύθερες θέσεις.

Το αρχικό WSN χρησιμοποιούσε αισθητήρες οπτικών κεφαλών για να αναγνωρίζει διερχόμενα οχήματα. Το βασικό του μειονέκτημα είναι ότι δεν μπορεί να αναγνωρίσει τη διάφορα ανάμεσα σε πεζούς, αντικείμενα ή οχήματα. Στην αρχή αυτό δεν θεωρήθηκε πρόβλημα, καθώς πίστευαν ότι το σύστημα θα εγκατασταθεί σε σημεία από τα οποία δεν θα διέρχονταν πεζοί όπως δρόμοι ταχείας κυκλοφορίας, ωστόσο τα πάρκινγκ είναι ένα μέρος στο οποίο δεν ισχύει αυτό. Για να λυθεί αυτό το πρόβλημα εγκατέστησαν άλλη μια οπτική κεφαλή-αισθητήρα σε κάθε κόμβο και έτσι οι δύο οπτικές κεφαλές αισθητήρες μπορούσαν πλέον να αναγνωρίσουν τη διαφορά μεταξύ των μεγεθών των αντικειμένων και να καταλάβουν αν ένα αντικείμενο είναι όχημα και όχι πεζός ή κάτι άλλο.

Οι κεφαλές τοποθετούνται σε ύψος 0.9 μέτρα από το έδαφος, που είναι το ύψος που αντιστοιχεί στο κεντρικό μέρος ενός αυτοκινήτου και βρίσκεται λίγο πιο πάνω από τα λάστιχα. Οι δυο σένσορες τοποθετούνται σε μια απόσταση από 2.6 έως 3.5 μέτρα. Αυτή η απόσταση μας επιτρέπει να αναγνωρίσουμε οχήματα και όχι πεζούς ή μοτοσυκλέτες. Μια άλλη ενδιαφέρουσα πληροφορία που μπορούν να μας δώσουν είναι η κατεύθυνση ενός οχήματος, αρκετά χρήσιμο σε γκαράζ που είσοδοι μπορεί να είναι ταυτόχρονα και έξοδοι.



Εικόνα 9: Αισθητήρες τοποθετημένοι σε απόσταση μεταξύ τους για να αναγνωρίζουν τη διαφορά μεταξύ των αντικειμένων [5]

Και σε αυτό το σύστημα βλέπουμε αισθητήρες παρκαρίσματος εσωτερικού χώρου και πρόκειται για θέσεις πάρκινγκ επί πληρωμή.

2.3 Διαφορετικότητα της λύσης μας

Αναλύσαμε και παρουσιάσαμε συστήματα τα οποία έχουν ως στόχο το έξυπνο παρκάρισμα. Οι συγκεκριμένες λύσεις ήταν αρκετά αξιόλογες και καθεμία από αυτές είχε τα δικά της πλεονεκτήματα και μειονεκτήματα. Ωστόσο δεν είναι εύκολο να συγκρίνουμε όλες αυτές τις λύσεις με την δικιά μας καθώς δεν θα χρησιμοποιήσουμε την ίδια τεχνική αλλά ούτε και το ίδιο εύρος θέσεων, αφού εμείς δεν θα παρατηρούμε θέσεις παρκαρίσματος επί πληρωμή ή κλειστούς χώρους πάρκινγκ, αλλά τις υπόλοιπες θέσεις που βρίσκονται στον δρόμο. Η τεχνική που θα χρησιμοποιήσουμε για να δώσουμε λύση στο πρόβλημα που δημιουργείται είναι η καταγραφή θέσεων. Σημαντικό μας πλεονέκτημα είναι η πλήρης καταγραφή της κατάστασης των θέσεων ανεξαρτήτως τοποθεσίας.

3. Απαιτήσεις

Σε αυτό το κεφάλαιο θα περιγράψουμε την αρχιτεκτονική της εφαρμογής που θέλουμε να υλοποιήσουμε, τις υλοποιήσεις που χρειάζεται να γίνουν και τον εξοπλισμό που χρειάζεται να προμηθευτούμε.

3.1 Γενικός Σχεδιασμός

Σκοπός μας είναι να αναπτυχθεί μια πλατφόρμα η οποία θα είναι απολύτως φιλική προς τον χρήστη και θα δουλεύει σε όσο το δυνατόν γίνεται περισσότερες συσκευές. Γύρω από αυτή την πλατφόρμα θέλουμε να χτίσουμε μια κοινότητα η οποία θα την χρησιμοποιεί συνέχεια,δίνοντας και λαμβάνοντας πληροφορίες. Βέβαια δεν χρειάζεται να προσθέτει κάποιος δικά του markers στην πλατφόρμα, μπορεί για π.χ να συνεισφέρει με το να ορίζει ως μη έγκυρα κάποια markers. Αυτός είναι και ο λόγος που προσπαθούμε να κατασκευάσουμε μια οικονομική λύση και για να το επιτύχουμε αυτό πρέπει να χρησιμοποιήσουμε ότι μας προσφέρει η τεχνολογία του σήμερα.

Χρειάζεται να υλοποιηθεί το σύστημα από την αρχή, μιας και δεν υπάρχει κάτι παρόμοιο προς το παρόν το οποίο να ικανοποιεί τις ανάγκες μας. Βέβαια στην διάθεση μας έχουμε διάφορα frameworks και apis τα οποία και μπορούμε να χρησιμοποιήσουμε και τα οποία θα μας βοηθήσουν στο να οδηγηθούμε σε ένα πιο ποιοτικό αποτέλεσμα. Το αποτέλεσμα αυτό αποτελεί και τον λόγο για τον οποίο δεν προσπαθούμε να γράψουμε την εφαρμογή από το μηδέν.

Θα χρησιμοποιήσουμε κάποιο framework για να φτιάξουμε τον κεντρικό κορμό της εφαρμογής, ο οποίος θα διασυνδεθεί με μια βάση της επιλογής μας για να μπορούμε να αποθηκεύουμε τα δεδομένα. Ακόμα θα χρειαστούμε κάποιο api που να προσφέρει χάρτες, το οποίο και θα ενσωματωθεί στην υλοποίηση μας. Κάτι που θα πρέπει να προσέξουμε, είναι η εφαρμογή μας (ο χάρτης) να μπορεί να πάρει αυτόματο μέγεθος, και ακόμα να αναπροσαρμόζει το μέγεθος του σε τυχόν αλλαγές του μεγέθους και του πλάτους του browser.

Οι πιο σημαντικές δυνατότητες της εφαρμογής πρέπει να γραφούν ως web services για να μπορούν να χρησιμοποιηθούν ευρέως. Στην περίπτωση μας θα χρησιμοποιούνται από την εφαρμογή, τον android client και τον pc client, τα οποία θα γραφτούν σε διαφορετικές γλώσσες.

Για την client android συσκευή θα γραφτεί μια εφαρμογή, η οποία θα τρέχει ως service και θα καλεί συνέχεια το web service του server, το οποίο και θα ενημερώνει την θέση του συγκεκριμένου χρήστη στην βάση. Χρειάζεται να βρούμε τι libs μας προσφέρονται για το android όσον αφορά τους web service clients. Το πρόγραμμα μας θα περιλαμβάνει και σύστημα login, το οποίο θα χρησιμοποιεί και αυτό το web service μας.

Ακόμα θα χρειαστεί να γράψουμε ένα script για τις άλλες συσκευές που έχουμε σκοπό να υποστηρίξουμε, το οποίο θα καλεί το web service συνέχεια και θα κάνει update την θέση του χρήστη. Για να μπορεί να τρέχει αυτόματα από το σύστημα θα πρέπει να περαστεί στο cron του χρήστη.

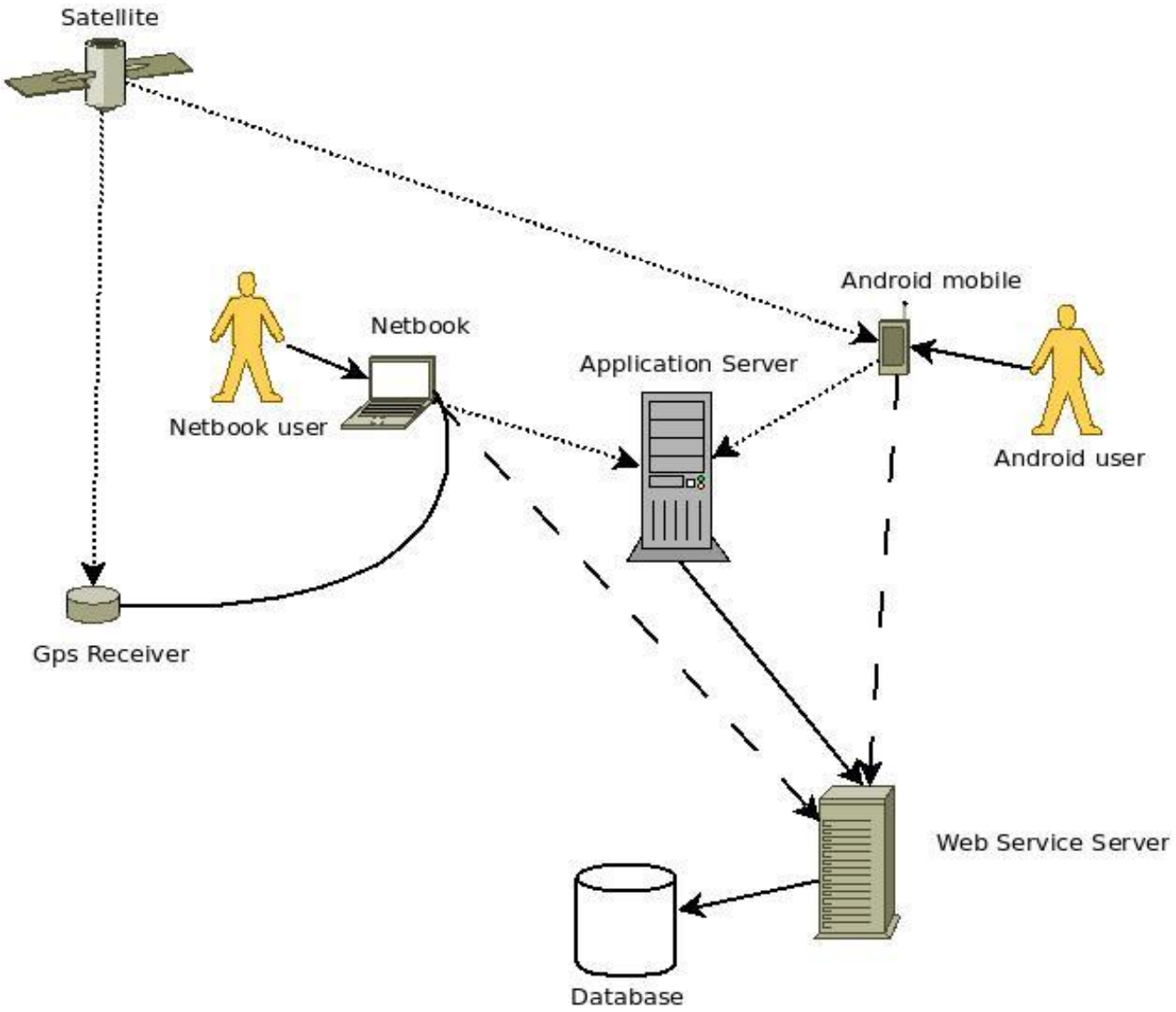
Για την εφαρμογή θα χρειαστεί να γράψουμε έναν αλγόριθμο είτε με την μορφή sql queries είτε με την μορφή κώδικα, ο οποίος με βάση το στίγμα του χρήστη θα υπολογίζει τις πιο κοντινές και διαθέσιμες για εκείνη την στιγμή θέσεις παρκαρίσματος. Ο τύπος που θα χρησιμοποιήσουμε είναι ο τύπος του haversaine. [8]

3.2 Αρχιτεκτονική εφαρμογής

Η αρχιτεκτονική που θέλουμε να επιτύχουμε είναι η εξής

- **Κεντρικός server,** ο κεντρικός server ο οποίος φροντίζει για να εμφανίζει την εφαρμογή στον χρήστη
- **Web Service server,** ο web service server ο οποίος είναι υπεύθυνος για να εκτελεί τα αιτήματα που λαμβάνει από τον server, τον client του android και τον client του pc. Οι περισσότερες λειτουργίες εκτελούν queries στην κεντρική βάση της εφαρμογής
- **Βάση δεδομένων,** είναι υπεύθυνη για την αποθήκευση όλων των δεδομένων και χρησιμοποιείται από τις λειτουργίες του web service
- **Android συσκευή**, ο χρήστης ο οποίος χρησιμοποιεί την android κινητή συσκευή του για να δει την εφαρμογή
- **Pc συσκευή,** ο χρήστης που χρησιμοποιεί το netbook - laptop του για να δει την εφαρμογή

Η αρχιτεκτονική που θέλουμε να επιτύχουμε φαίνεται στην εικόνα 10.



Εικόνα 10: Αρχιτεκτονική συστήματος Gipi

3.3 Εύρεση κατάλληλου hardware

Για την συγκεκριμένη υλοποίηση εκτός από τις λύσεις λογισμικού που θα χρησιμοποιήσουμε θα χρειαστούμε και ορισμένα κομμάτια hardware.

- **Server,** ο κεντρικός server στον οποίο θα τρέχει η εφαρμογή, δεν χρειαζόμαστε ένα δυνατό μηχάνημα, πρέπει όμως να πληρεί τις προϋποθέσεις για να εκτελέσει λογισμικό linux
- **Gps receiver**, στην παρούσα υλοποίηση θα χρειαστούμε έναν gps receiver, ο οποίος θα επικοινωνεί με τους δορυφόρους που υπάρχουν και θα μας επιστρέφει την θέση του χρήστη. Η διασύνδεση του λογισμικού με τον gps receiver γίνεται όπως είπαμε, με το gpsd [18]. Αυτό σημαίνει ότι θα πρέπει να βρούμε μια συσκευή που να υποστηρίζεται από το gpsd
- **Netbook ή laptop**, θα χρειαστούμε έναν φορητό υπολογιστή, πρέπει όμως να πληρεί τις

προϋποθέσεις για να εκτελέσει λογισμικό linux

- **Android συσκευή,** θα χρειαστεί να βρούμε μια android συσκευή για να κάνουμε τις δοκιμές μας, καθώς και να εγκαταστήσουμε την εφαρμογή μας

4. Τεχνολογικές Υλοποιήσεις

Σε αυτή την εργασία θα χρησιμοποιηθούν λύσεις ανοιχτού κώδικα, για να μπορούμε σε περίπτωση που χρειαστεί, να επέμβουμε για να πάρουμε το επιθυμητό αποτέλεσμα.

Στην παρούσα εργασία έχουμε στην διάθεση μας αρκετές λύσεις ανοιχτού λογισμικού:

1) **Linux λογισμικό,** υπάρχουν πολλά διαφορετικά releases και όλα έχουν τον ίδιο κορμό. Ο λόγος που διαλέγουμε το συγκεκριμένο είδος λογισμικού είναι γιατί χρειαζόμαστε όλα τα εργαλεία που μπορεί να μας προσφέρει και θα μας οδηγήσουν σε ένα καλύτερο αποτέλεσμα Μερικά από τα λειτουργικά αυτά είναι τα arch, fedora, debian, suse, centos, red hat, ubuntu και slackware
2) **Βάση δεδομένων,** στην διάθεση μας έχουμε τις sqlite, mysql, postgresql και oracle. Για να ολοκληρωθεί η εφαρμογή μας, σίγουρα θα χρειαστούμε μια βάση δεδομένων έτσι ώστε να αποθηκεύουμε δεδομένα, καθώς και να εκτελούμε ερωτήματα στην βάση
3) **Android sdk**, για την διασύνδεση με το android κινητό θα χρειαστεί το sdk της google, που μας επιτρέπει να γράψουμε εφαρμογές για το android os. Για το android μπορούμε να γράψουμε προγράμματα στην γλώσσα java,[32] χωρίς αυτό να σημαίνει οτι δεν μπορούμε να γράψουμε και σε scripting γλώσσες (ASE)
4) **Gpsd λογισμικό,** αξιόλογο λογισμικό το οποίο βοηθάει ώστε διασυνδέσουμε του δέκτη gps με τον υπολογιστή μας
5) **Web server,** Apache server ή lighttpd. Ο web server θα μας χρειαστεί για μπορεί να ερμηνευθεί η σελίδα μας από τους browsers
6) **Γλώσσα προγραμματισμού,** php [31] ,ruby. Διαλέγουμε τις δυο αυτές γλώσσες γιατί πρόκειται για μια web εφαρμογή και θεωρούμε ότι μπορούν να μας φέρουν πιο κοντά στο αποτέλεσμα που προσπαθούμε να επιτύχουμε. Στην διάθεση μας για τον κεντρικό κορμό είχαμε και άλλες γλώσσες όπως c και java, θεωρούμε ότι δεν θα μας χρειαστούν για την υλοποίηση της εφαρμογής. Παρ' όλα αυτά η java χρησιμοποιείται στον client της android συσκευής
7) **Frameworks γραμμένα σε php,** όπως symfony, zend, code igniter και cakephp. Θεωρούμε ότι η χρήση ενός framework θα μας βοηθήσει να προχωρήσουμε με γρήγορους ρυθμούς όσον αφορά την υλοποίηση της εφαρμογής μας, επίσης προτιμάμε να χρησιμοποιήσουμε ένα από τα δοκιμασμένα frameworks για να αποφύγουμε όσο το δυνατόν γίνετε τα bugs
8) **Soap τεχνολογία,** μπορεί να χρησιμοποιηθεί στα web services. Τα web services βοηθούν ώστε οι λειτουργίες να είναι διαθέσιμες από παντού ανεξαρτήτως πλατφόρμας ή γλώσσας. Αυτό μας χρειάζεται, καθώς έχουμε ως σκοπό να χρησιμοποιήσουμε διαφορετικές τεχνολογίες
9) **Ajax τεχνολογία,** για να είναι όσο το δυνατόν πιο λειτουργική η εφαρμογή μας, αφού με την συγκεκριμένη τεχνολογία (δεν αποτελεί κάτι το καινούργιο αλλά μια ένωση υπαρχόντων πρωτοκόλλων - τεχνολογιών) μπορούμε να μειώσουμε τα περιττά submits και να αλλάζουμε πολύ εύκολα το content της σελίδας χωρίς ο χρήστης να παρατηρεί διαφορά στον browser του αλλά μόνο στο περιεχόμενο της σελίδας
10) **Scripting γλώσσες,** python και perl. Η χρήση του gps receiver μας οδηγεί στην χρήση μιας από τις δυο scripting γλώσσες, καθώς θα μας χρειαστούν τα bindings που υπάρχουν μεταξύ του gpsd και των perl – python, για να μπορούμε να διαβάζουμε τα latitude και longitude που έχει ο χρήστης εκείνη την στιγμή και να τα εισάγουμε στην βάση
11) **Api με χάρτες,** google maps api [24] ή yahoo maps api. Από την στιγμή που βασιζόμαστε σε έναν χάρτη, σίγουρα θα χρειαστούμε ένα api που να μας δίνει πρόσβαση σε αυτόν. Πρόκειται για δύο πολύ αξιόλογα apis όσον αφορά τις λειτουργικότητες που μας προσφέρουν. Βασικοί παράγοντες επιλογής θα είναι η ευκολία ενσωμάτωσης στην εφαρμογή καθώς και η λειτουργίες

που προσφέρουν

12) **Javascript framework,** jquery [23] ή mootools. Η χρήση ενός javascript framework θα μειώσει αρκετά τα προβλήματα που θα αντιμετωπίσουμε με την χρήση της γλώσσας javascript στους browsers, κάτι που μας απασχολεί καθώς θέλουμε η εφαρμογή μας να μπορεί να παίζει σε όσο το δυνατόν γίνετε σε περισσότερους browsers
13) **Έλεγχος των webservices,** ο οποίος μπορεί να πραγματοποιηθεί με το πρόγραμμα soapui. Με αυτό το πρόγραμμα μπορούμε να καλέσουμε τα functions του web service αφού εισάγουμε τις παραμέτρους και να πάρουμε απαντήσεις από το κάλεσμα τους.
14) **Dia,** αποτελεί ένα σχεδιαστικό πρόγραμμα το οποίο έχουμε στην διάθεση μας για να κατασκευάσουμε τα διαγράμματα που χρειαζόμαστε
15) **Gimp,** αποτελεί ένα πρόγραμμα επεξεργασίας εικόνων το οποίο έχουμε ως σκοπό να χρησιμοποιήσουμε
16) **Inkscape,** χρησιμοποιείται για την δημιουργία svg εικόνων και εμείς μπορούμε να το χρησιμοποιήσουμε για να δημιουργήσουμε εικονίδια για την εφαρμογή μας

5. Σχεδιασμός / Μοντελοποίηση λύσης

Η λύση μας περιλαμβάνει λειτουργίες, με τις πιο σημαντικές να εκτελούνται μέσω των web services για να είναι εύκολα προσβάσιμες.

5.1 Λειτουργικότητα εφαρμογής

Το functionality αποτελείται από τις παρακάτω λειτουργίες:

- **Είσοδος στο σύστημα,** ο χρήστης γίνεται authenticated με βάση τα username και password τα οποία εισάγει, εφόσον επιβεβαιωθεί ως αυθεντικός χρήστης του συστήματος μπορεί να εισέλθει σε αυτό
- **Προσθήκη Marker,** ο χρήστης έχει την δυνατότητα να κάνει κλικ στον χάρτη και να αποθηκεύσει έναν marker σε εκείνο το σημείο, ο οποίος εμφανίζετε στον χάρτη εφόσον τηρεί τις προϋποθέσεις
- **Διαγραφή Marker,** δίνεται η δυνατότητα να διαγραφεί ένας marker κλικάροντας πάνω του από τον χάρτη. Ο marker διαγράφεται από την βάση και σβήνεται από τον χάρτη. Μόνο ο ιδιοκτήτης του marker μπορεί να τον σβήσει
- **Ορισμός Marker ως μη έγκυρου,** για τους markers στους οποίους ο χρήστης δεν είναι ιδιοκτήτης δίνεται η δυνατότητα να οριστούν ως μη έγκυροι. Επίσης ορίζονται μαζί τους ως μη έγκυροι και οι markers που βρίσκονται στην ίδια θέση και στην ίδια χρονική ζώνη (προστίθεται +1 στην μη εγκυρότητα των markers που βρέθηκαν). Οι markers που έχουν περισσότερα ή και 3 μη έγκυρα διαγράφονται από την βάση και σβήνονται από τον χάρτη
- **Αυτόματο μέγεθος του χάρτη,** ο χάρτης χρησιμοποιώντας τεχνικές css και javascript (jquery) έχει την δυνατότητα να κάνει auto-resize σε οποιαδήποτε μεγέθους οθόνη. Το 100% width επιτυγχάνεται με το css και το ύψος της οθόνης δίνεται από λειτουργίες της γλώσσας javascript. Επίσης χρησιμοποιείται event το οποίο διαβάζει για πιθανές τροποποιήσεις στο ύψος του παραθύρου και κάνει resize το παράθυρο αυτόματα
- **Τρέχουσα θέση,** ο χρήστης μπορεί να δει την θέση που έχει εκείνη την στιγμή, η οποία λαμβάνεται από την βάση για τον κάθε χρήστη. Η συγκεκριμένη θέση μπορεί να γίνει update με 2 τρόπους όπως είπαμε είτε από την συσκευή android με την χρήση του service είτε με την χρήση του script της python για τον υπολογιστή. Στον χάρτη εμφανίζεται ένα μπλε εικονίδιο το οποίο δείχνει την θέση που έχει ο χρήστης εκείνη την στιγμή. Επίσης μας δείχνει και τις 3 πιο κοντινές διαδρομές για τον χρήστη σε σχέση με την θέση που έχει εκείνη την στιγμή. Εμφανίζονται τρεις γραμμές. Με πράσινο η πιο κοντινή σε αυτόν, κίτρινο η δεύτερη σε απόσταση και κόκκινο η πιο μακρινή. Οι διαδρομές που παρέχονται, προέρχονται από το google maps api και είναι κανονικές διαδρομές που να χρησιμοποιήσει ο χρήστης για να φτάσει στον προορισμό του. Δίνετε η δυνατότητα στον χρήστη να ορίζει κάθε πότε θέλει να ανανεώνεται η θέση του καθώς και επιλογή απενεργοποίησης
- **Markers του χάρτη,** ο χρήστης με την είσοδο του στο σύστημα θα δει τα markers που αντιστοιχούν στην χρονική ζώνη που βρίσκεται. Θα παρατηρήσει εικονίδια (πράσινα , κίτρινα, κόκκινα και μπλε). Όπως είπαμε, το μπλε δείχνει την τρέχουσα θέση του χρήστη. Τα υπόλοιπα εικονίδια παρουσιάζουν τον βαθμό δυσκολίας παρκαρίσματος της θέσης με το πράσινο να αποτελεί ένα εύκολο παρκάρισμα, το κίτρινο ένα μέτριο και το κόκκινο ένα δύσκολο. Για να μπορεί εύκολα ο χρήστης να παρατηρεί τα στίγματα, χρησιμοποιείται μαι τεχνική του css (το z-

index) για να εμφανίζονται πιο ψηλά τα στίγματα σε σχέση με τα άλλα. Τα πράσινα στίγματα εμφανίζονται πιο ψηλά σε σχέση με τα κίτρινα και τα κόκκινα. Ενώ τα κίτρινα εμφανίζονται πιο χαμηλά σε σχέση με τα πράσινα και πιο ψηλά σε σχέση με τα κόκκινα. Τα κόκκινα στίγματα εμφανίζονται πιο χαμηλά σε σχέση με τα υπόλοιπα. Το μπλε στίγμα εμφανίζεται πιο ψηλά όσον αφορά όλα τα υπόλοιπα. Και εδώ δίνετε στον χρήστη η δυνατότητα να ορίζει κάθε πότε επιθυμεί να ανανεώνεται ο χάρτης με τα νέα στίγματα καθώς και επιλογή απενεργοποίησης

- **Χρήση φίλτρων,** δίνεται η δυνατότητα στον χρήστη να χρησιμοποιήσει φίλτρα όσον αφορά τους markers του χάρτη (τα φίλτρα αυτά εφαρμόζονται και στις τρεις θέσεις που επιστρέφει η τρέχουσα θέση). Στα φίλτρα ο χρήστης μπορεί να ορίσει το επίπεδο δυσκολίας της θέσης (όπως αναφέραμε εύκολο, μέτριο,δύσκολο) καθώς και αν θέλει να δει όλα τα επίπεδα δυσκολίας. Επίσης μπορεί να ορίσει ποιους markers θέλει να δει σε σχέση με τον ιδιοκτήτη που έχουν (όλους, μόνο τους δικούς του, με διαφορετικό ιδιοκτήτη). Ακόμα μπορεί να επιλέξει σε ποια χρονική ζώνη θέλει να αναζητήσει markers (όλες, την συγκεκριμένη που βρίσκεται καθώς και το φάσμα όπως εμείς το έχουμε χωρίσει). Για να προκύψουν οι χρονικές ζώνες χωρίσαμε την ημέρα ανά διαστήματα των τεσσάρων ωρών
- **Εμφάνιση / Απόκρυψη λειτουργιών,** δίνεται η δυνατότητα στον χρήστη να αποκρύπτει τις μπάρες που περιέχουν τις λειτουργικότητες του χάρτη, και με αυτό τον τρόπο μπορεί να βλέπει τον χάρτη στο μέγιστο μέγεθος που του επιτρέπει η οθόνη του.

5.2 Λειτουργικότητα Web Services

Παρακάτω θα περιγράψουμε τις λειτουργικότητες που μας δίνει το web service της εφαρμογής. Όλα τα κομμάτια που απαρτίζουν αυτή την εργασία καλούν τις λειτουργίες του web service:

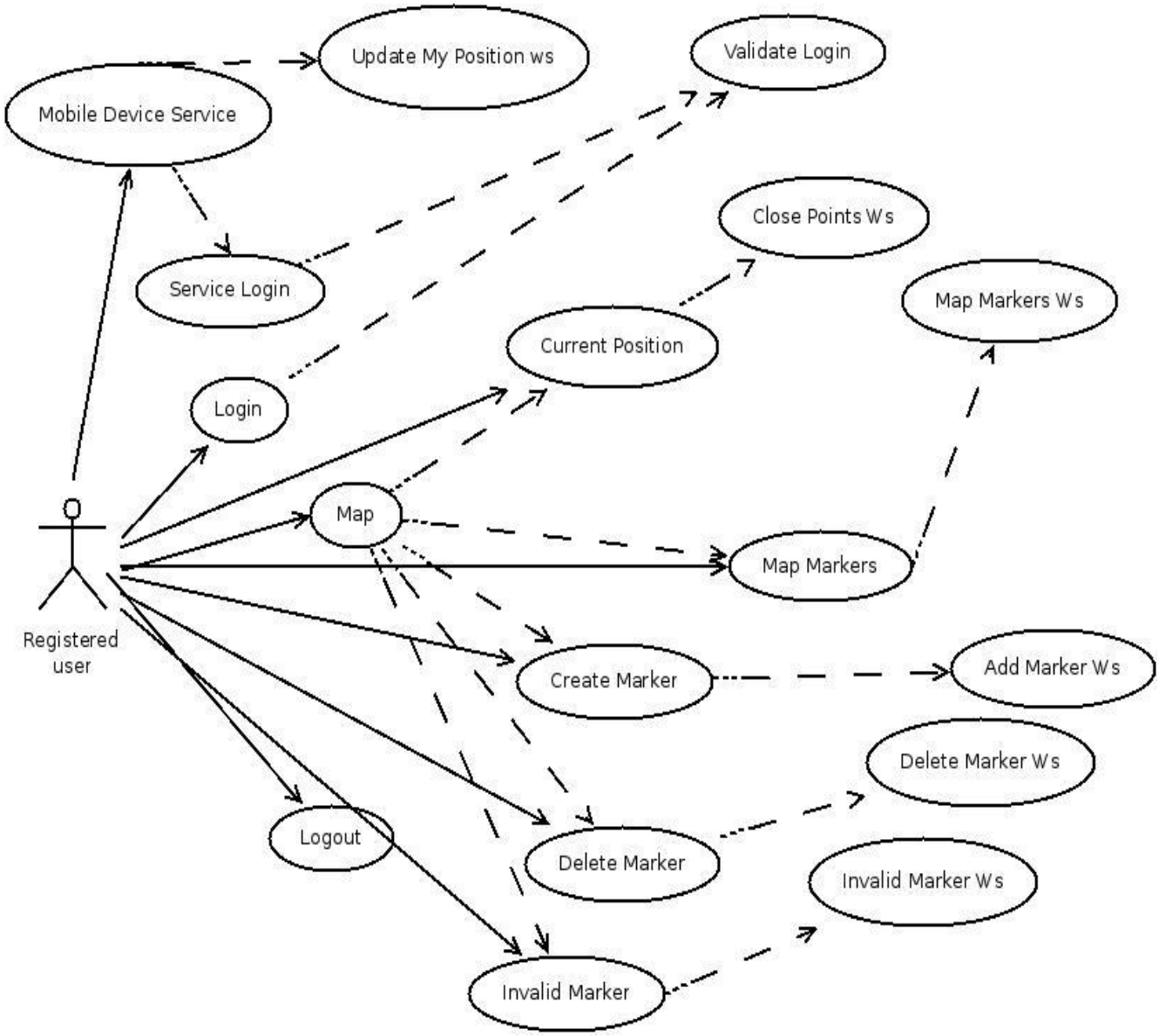
- **MapMarkers,** επιστρέφει τα markers που ζητάει ο χρήστης. Οι επιλογές που δίνονται στον χρήστη έχουν να κάνουν με τα φίλτρα που εισάγει και δίνεται η δυνατότητα να οριστούν οι παρακάτω επιλογές: το id του χρήστη (αποτελεί ένα από τα φίλτρα), το επίπεδο δυσκολίας παρκαρίσματος (αποτελεί και αυτό ένα από τα φίλτρα), τη χρονική ζώνη που θέλουμε να δούμε (και αυτό αποτελεί ένα από τα φίλτρα), τον τύπο ιδιοκτήτη ανά marker που θέλουμε (και αυτό αποτελεί ένα από τα φίλτρα)
- **AddMarker,** προσθέτει τον marker στην βάση και μας τον επιστρέφει. Τα ορίσματα που εισάγονται έχουν να κάνουν με τον marker που θα προστεθεί στην βάση. Τα ορίσματα είναι το username του χρήστη και το password του, χρειάζονται για να αυθεντικοποιηθεί ο χρήστης από το σύστημα (από αυτά τα δύο προκύπτει ο ιδιοκτήτης του marker), το επίπεδο δυσκολίας που ορίζει ο χρήστης, η χρονική στιγμή κατά την οποία θα είναι διαθέσιμη η θέση (από αυτό προκύπτει η χρονική ζώνη που είναι διαθέσιμη η θέση), το latitude και το longitude που έχει η θέση. Με την χρήση των δύο τελευταίων, εμφανίζεται η θέση πάνω στον χάρτη
- **ValidateLogin,** ελέγχει αν ο χρήστης είναι ένας έγκυρος χρήστης για το σύστημα. Τα ορίσματα που δέχεται είναι το username του χρήστη καθώς και το password του χρήστη. Το σύστημα πραγματοποιεί τους ελέγχους που χρειάζονται και μας επιβεβαιώνει το αν ο συγκεκριμένος χρήστης είναι ένας από τους ενεργούς, έγκυρους χρήστες του συστήματος ή όχι
- **DeleteMapMarker**, διαγράφει έναν συγκεκριμένο marker από το σύστημα. Για την διαγραφή χρειάζεται το id του marker καθώς και ορίσματα για να κάνει αυθεντικοποίηση του χρήστη εφόσον είναι έγκυρα τα στοιχεία που έλαβε μας επιστρέφει το id του marker που διαγράφτηκε
- **InvalidMapMarker,** ορίζει έναν marker ως μη έγκυρο. Το όρισμα που παίρνει είναι ποιον

marker να ορίσει ως μη έγκυρο καθώς και τα ορίσματα για να κάνει αυθεντικοποίηση, όπως τα περιγράψαμε προηγουμένως. Ένας μη έγκυρος marker διαγράφεται στις τρεις φορές. Επίσης μαζί του ορίζει ως μη έγκυρα και τα markers που έχουν το ίδιο latitude,longitude και ίδια χρονική ζώνη. Αυτά που έχουν τον δείκτη μη εγκυρότητας πάνω από τρεις φορές διαγράφονται. Η λειτουργία εκτελείται μόνο αν τα στοιχεία αυθεντικοποίησης είναι έγκυρα και μας επιστρέφει τα ids των markers που διαγράφτηκαν

- **UpdateMyPosition,** κάνει update την τρέχουσα θέση του χρήστη. Τα ορίσματα που παίρνει είναι το username και το password του χρήστη που θα ανανεωθεί. Το τρέχον latitude καθώς και το τρέχον longitude που έχει. Επιστρέφει το αποτέλεσμα του update της θέσης, εφόσον ο χρήστης αυθεντικοποιηθεί. Ενημερώνει το profile του χρήστη με την ip από την οποία έγινε το update. για λόγους ασφαλείας
- **ClosePoints,** επιστρέφει τα κοντινότερα σημεία στο στίγμα που δώσαμε (ορίζουμε εμείς το πλήθος στιγμάτων που θα επιστρέφονται). Τα ορίσματα που δέχεται είναι τα εξής: το latitude της θέσης, το longitude της θέσης, το id του χρήστη (αποτελεί ένα από τα φίλτρα), το επίπεδο δυσκολίας παρκαρίσματος (αποτελεί ένα από τα φίλτρα), τη χρονική ζώνη που θέλουμε να δούμε (και αυτό αποτελεί ένα από τα φίλτρα), τον τύπο ιδιοκτήτη ανά marker που θέλουμε (και αυτό αποτελεί ένα από τα φίλτρα)
- **FindMarkers,** βρίσκει την απόσταση μεταξύ δυο σημείων σε χιλιόμετρα ή μίλια. Τα ορίσματα που παίρνει είναι τα εξής: το latitude του σημείου α, το longitude του σημείου α,το latitude του σημείου β, το longitude του σημείου β, τον τύπο της επιστροφής (χιλιόμετρα ή μίλια). Επιστρέφει την απόσταση των δυο σημείων στις μονάδες που ζητήσαμε

5.3 Περιπτώσεις Χρήσεις (Use Cases)

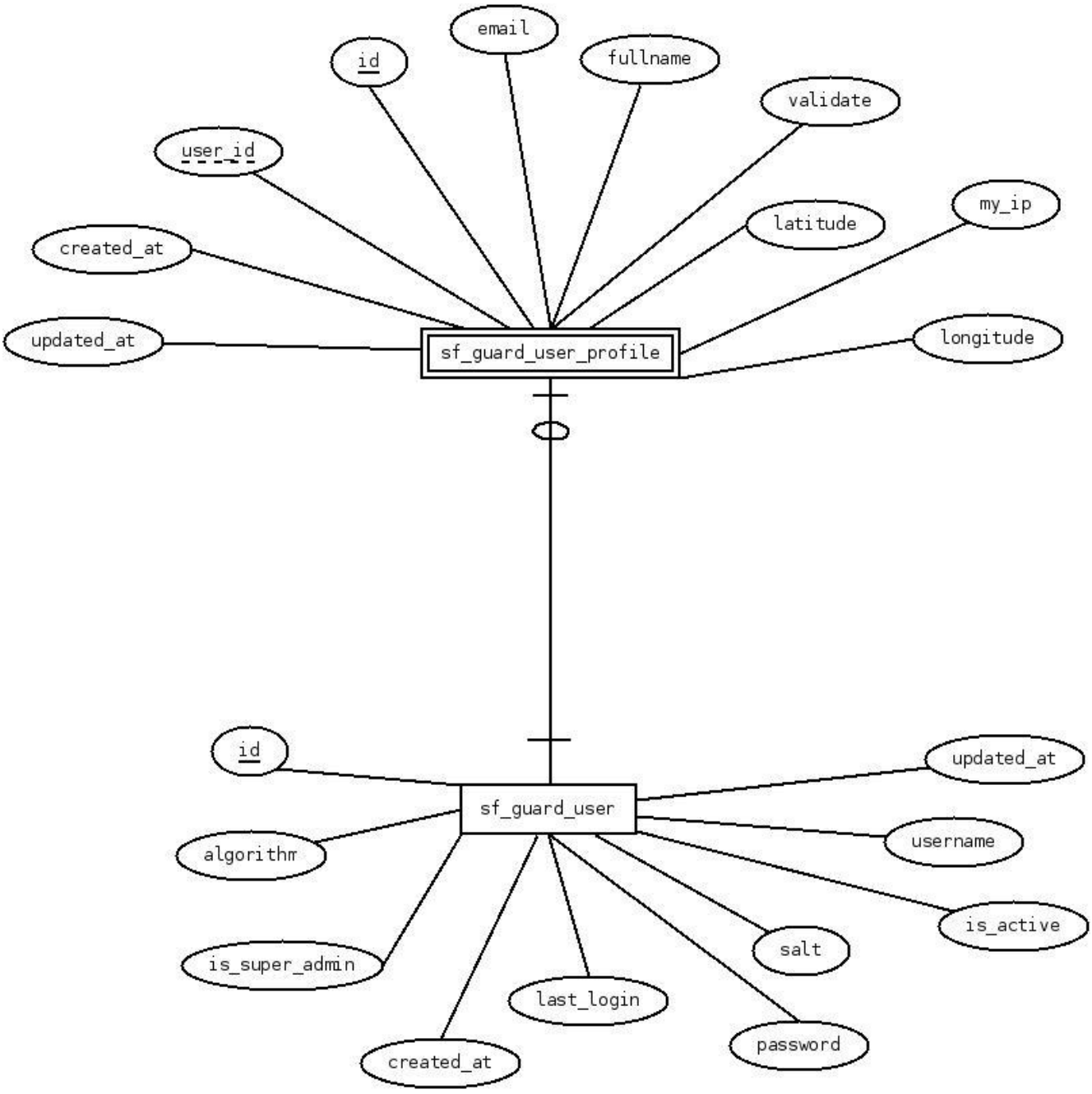
Στην εικόνα 11 που ακολουθεί βλέπουμε το διάγραμμα περίπτωσης χρήσης που αντιστοιχεί στην εφαρμογή. Στο διάγραμμα εμφανίζονται όλες οι λειτουργίες που μπορεί να εκτελέσει ένας αυθεντικοποιημένος χρήστης του συστήματος.



Εικόνα 11: Use case διάγραμμα

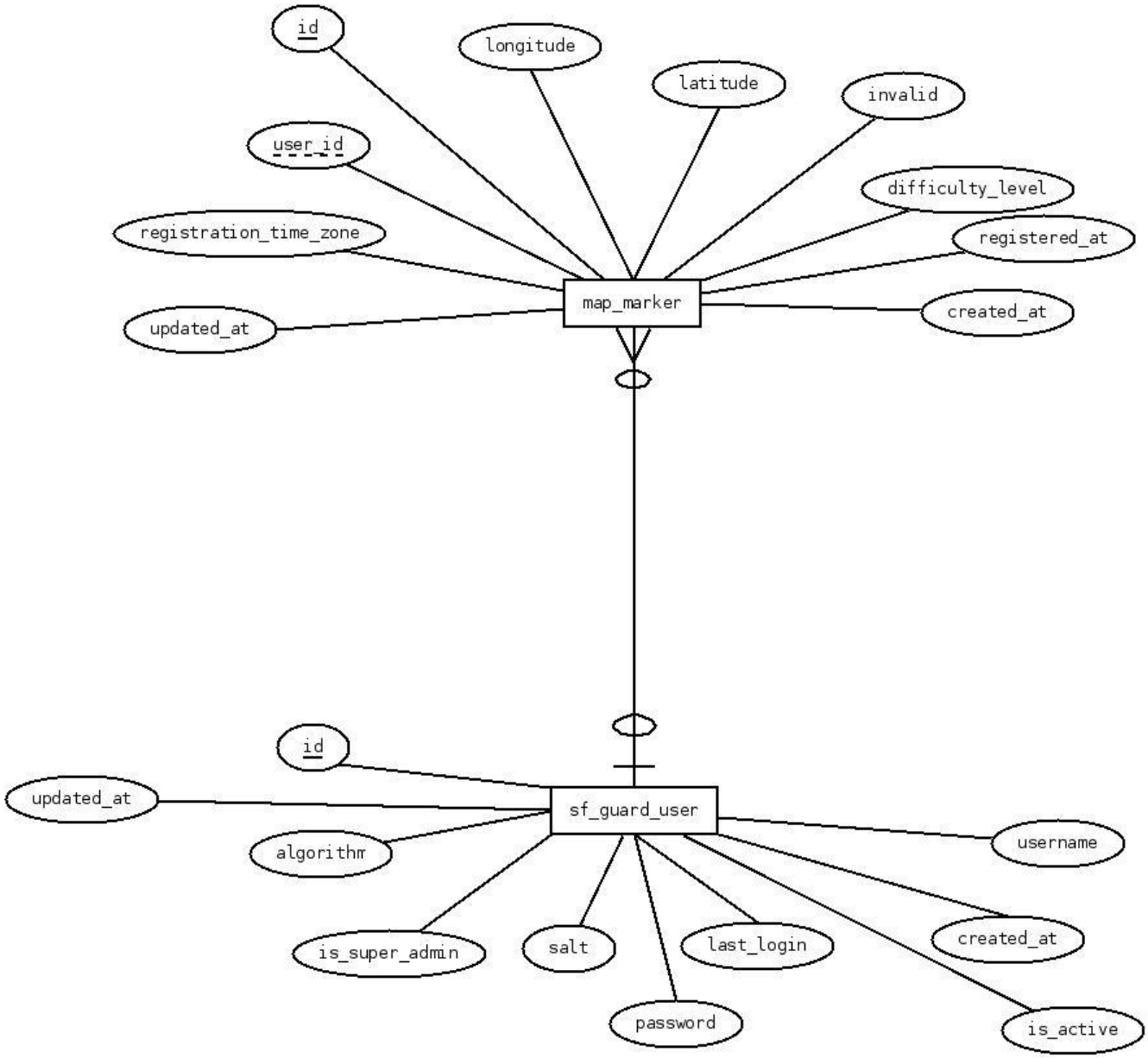
5.4 Μοντέλο Οντοτήτων Συσχετίσεων (Entity Relationship Model)

Στο μοντέλο οντοτητών συσχετίσεων στην εικόνα 12 φαίνεται η διασύνδεση μεταξυ των tables χρήστη (sf_guard_user) και του προφίλ του χρήστη(sf_guard_user_profile).



Εικόνα 12: ER σχήμα α

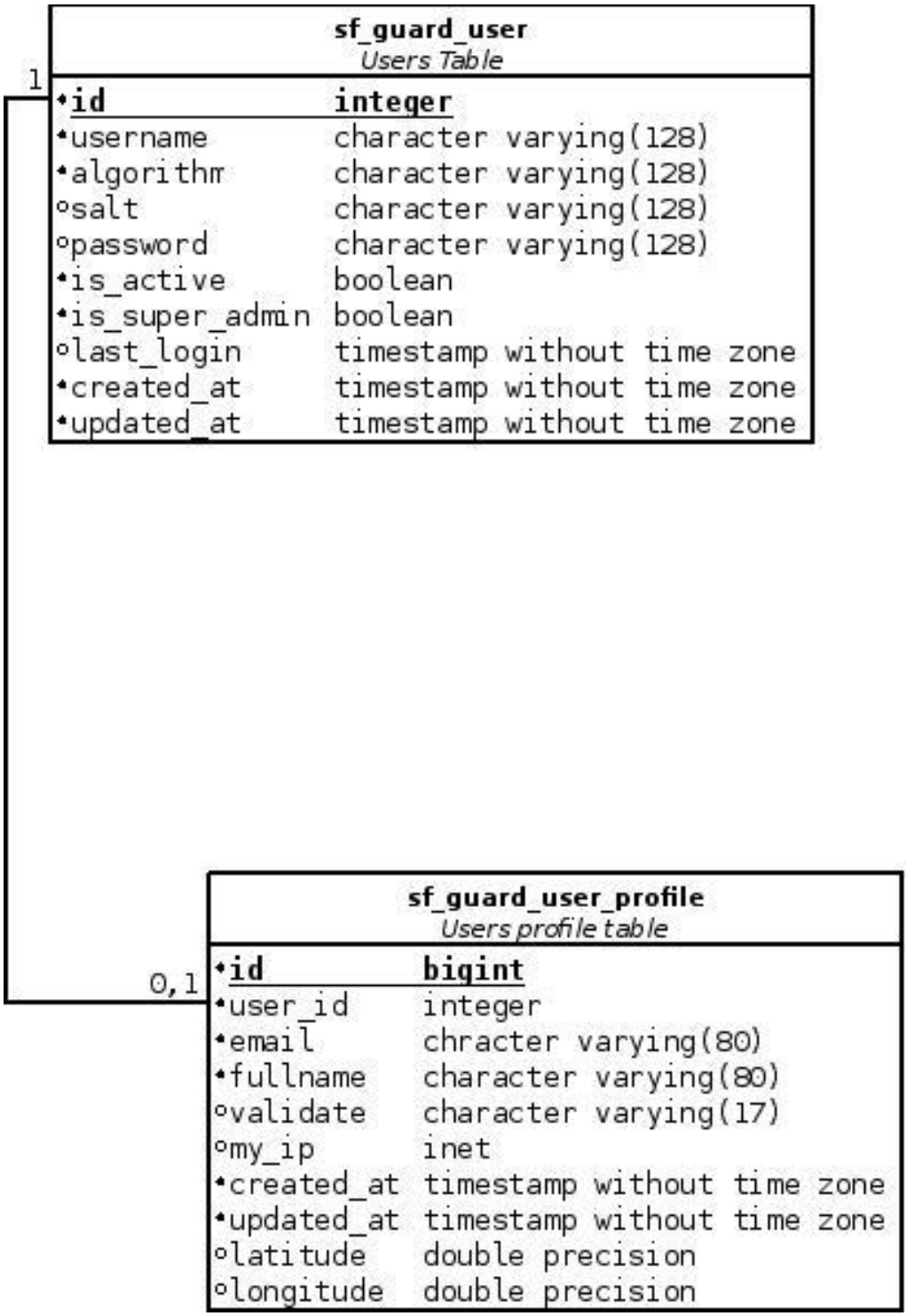
Στο μοντέλο οντοτήτων συσχετίσεων στην εικόνα 13 φαίνεται η διασύνδεση μεταξύ των tables χρήστη (sf_guard_user) και των markers του συστήματος(map_marker).



Εικόνα 13: ER σχήμα β

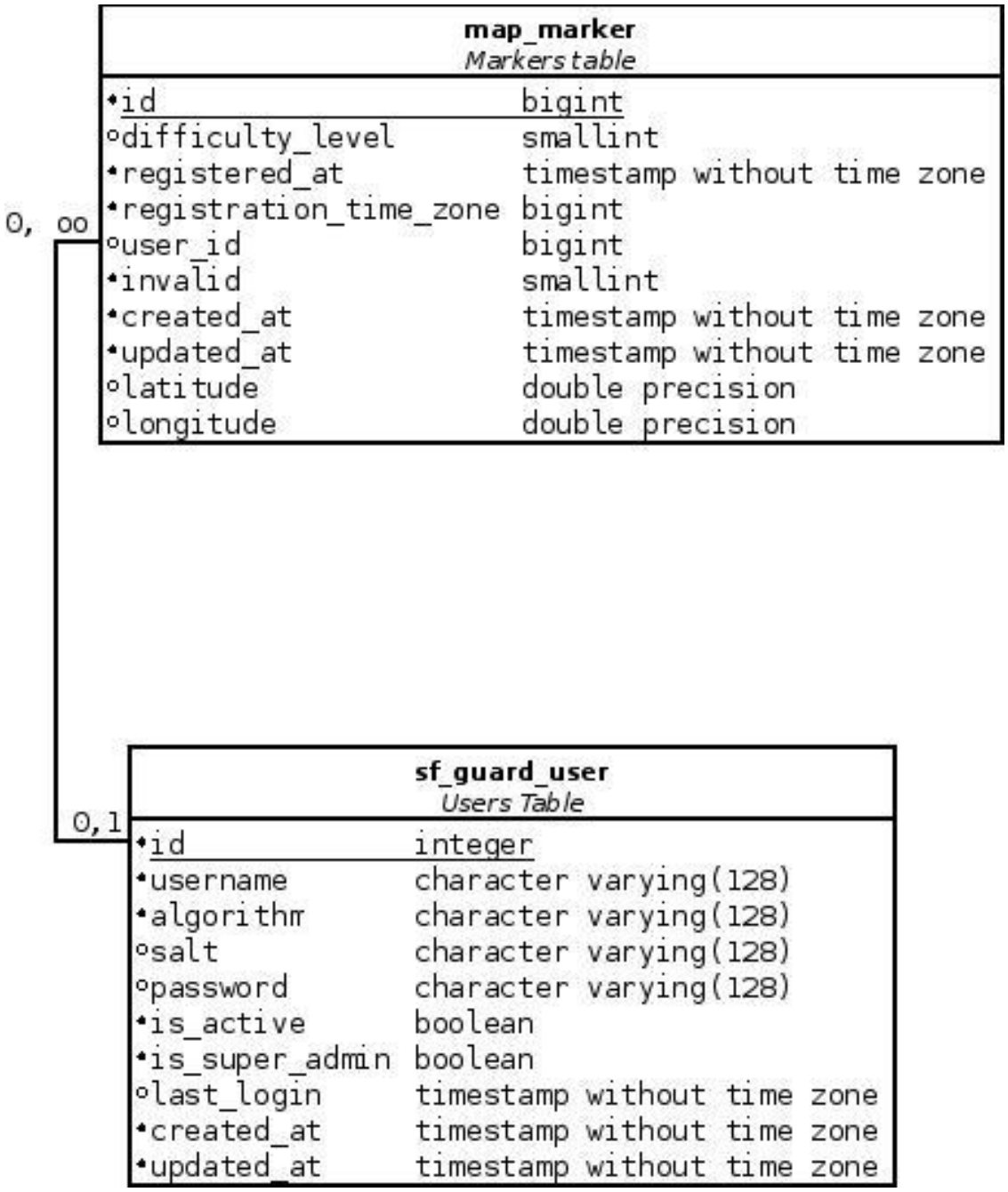
5.5 Σχεσιακό Μοντέλο (Relational Model)

Στο σχεσιακό μοντέλο στην εικόνα 14 φαίνεται η διασύνδεση μεταξύ των tables χρήστη (sf_guard_user) και του προφίλ του χρήστη(sf_guard_user_profile).



Εικόνα 14: Σχεσιακό μοντέλο σχήμα α

Στο σχεσιακό μοντέλο στην εικόνα 15 φαίνεται η διασύνδεση μεταξύ των tables χρήστη (sf_guard_user) και των markers του συστήματος(map_marker).



Εικόνα 15: Σχεσιακό μοντέλο σχήμα β

5.6 Διαγράμματα ακολουθίας μηνυμάτων (Message sequence charts)

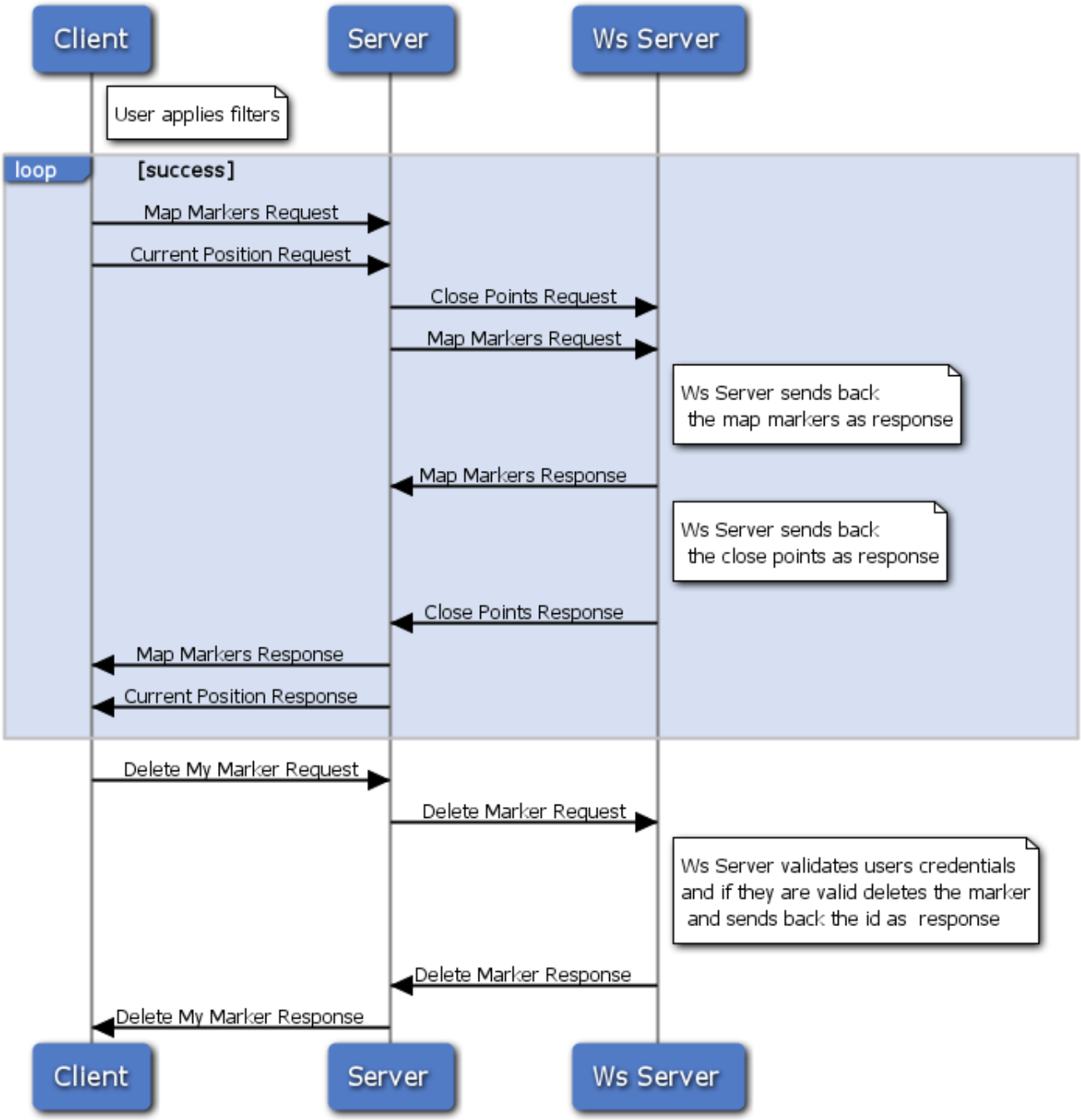
Στα κεφάλαια που ακολουθούν θα παρουσιάσουμε τα διαγράμματα ακολουθίας μηνυμάτων για τις βασικές λειτουργικότητες του συστήματος μας. Οι οποίες είναι οι εξής:

- Αίτημα για εμφάνιση του χάρτη από τον χρήστη
- Προσθήκη ενός map marker
- Έλεγχος για έγκυρο χρήστη
- Αίτημα για λήψη τρέχουσας θέση
- Διαγραφή marker από το σύστημα
- Διαδικασία ορισμού marker ως μη έγκυρου
- Αίτημα για λήψη των map markers
- Ανανέωση της τρέχουσας θέσης του χρήστη

5.6.1 Αίτημα για εμφάνιση του χάρτη από τον χρήστη

Στο διάγραμμα ακολουθίας μηνυμάτων της εικόνας 16 ο χρήστης αφού δει την κεντρική οθόνη της εφαρμογής, εισάγει τα φίλτρα που επιθυμεί. Αργότερα αφού δει τον χάρτη αποφασίζει να διαγράψει

μια από αυτές.

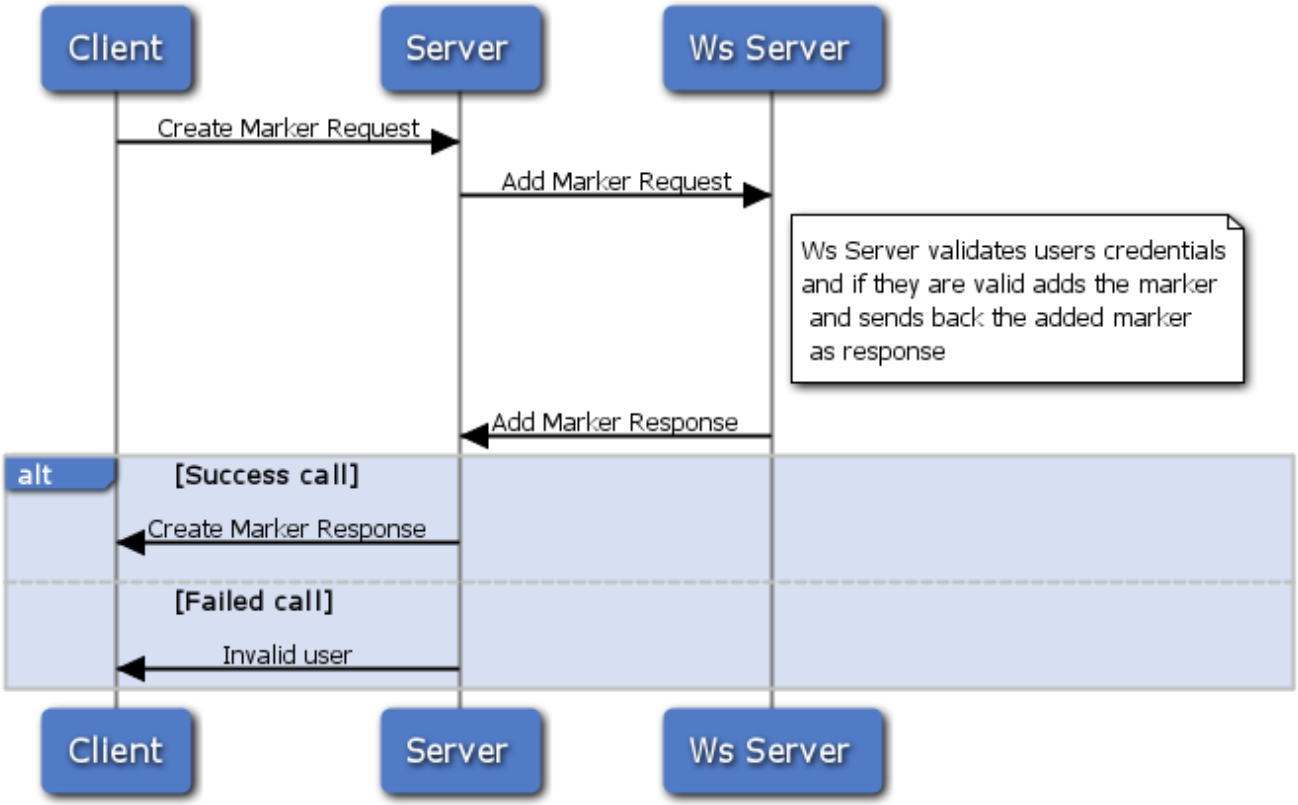


Εικόνα 16: Msc σχήμα διαδικασίας εμφάνισης του χάρτη στον χρήστη

5.6.2 Προσθήκη ενός map marker

Στο διάγραμμα ακολουθίας μηνυμάτων της εικόνας 17, ο χρήστης επιλέγει την θέση που επιθυμεί πάνω

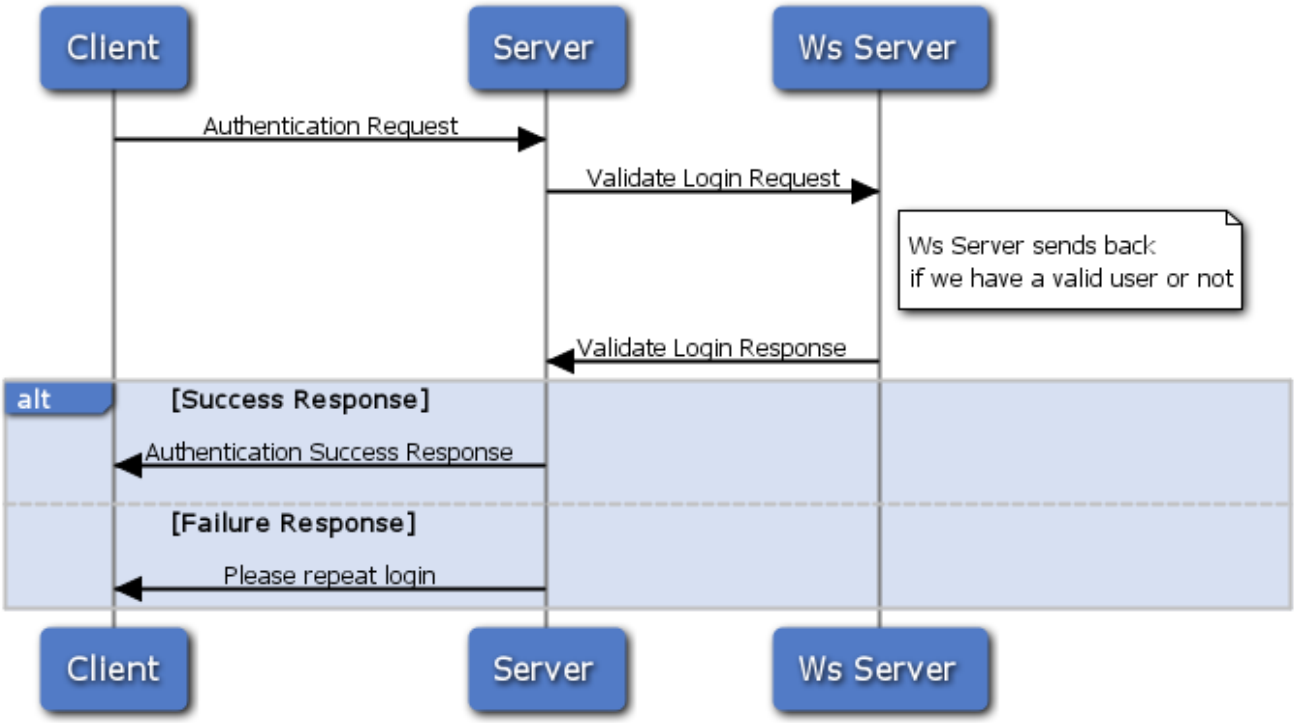
στον χάρτη και αποθηκεύει τον marker του, ο server στέλνει πίσω την απάντηση που έλαβε από τον web service server.



Εικόνα 17: Msc σχήμα Διαδικασίας προσθήκης map marker στο σύστημα

5.6.3 Έλεγχος για έγκυρο χρήστη

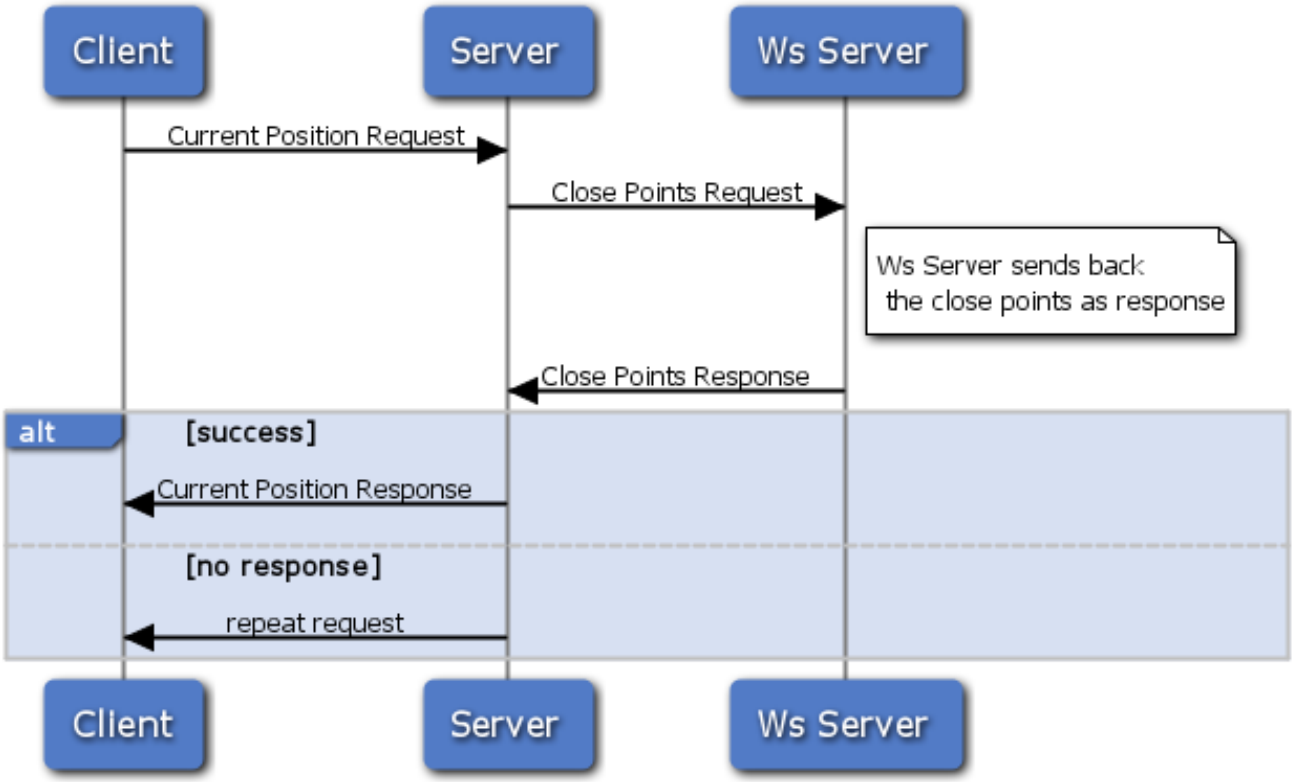
Στο διάγραμμα ακολουθίας μηνυμάτων της εικόνας 18, ο χρήστης εισάγει τα στοιχεία του και ελέγχεται αν πρόκειται για έγκυρο η μη έγκυρο χρήστη του συστήματος.



Εικόνα 18: Msc σχήμα διαδικασίας ελέγχου για έγκυρο χρήστη

5.6.4 Αίτημα για λήψη τρέχουσας θέση

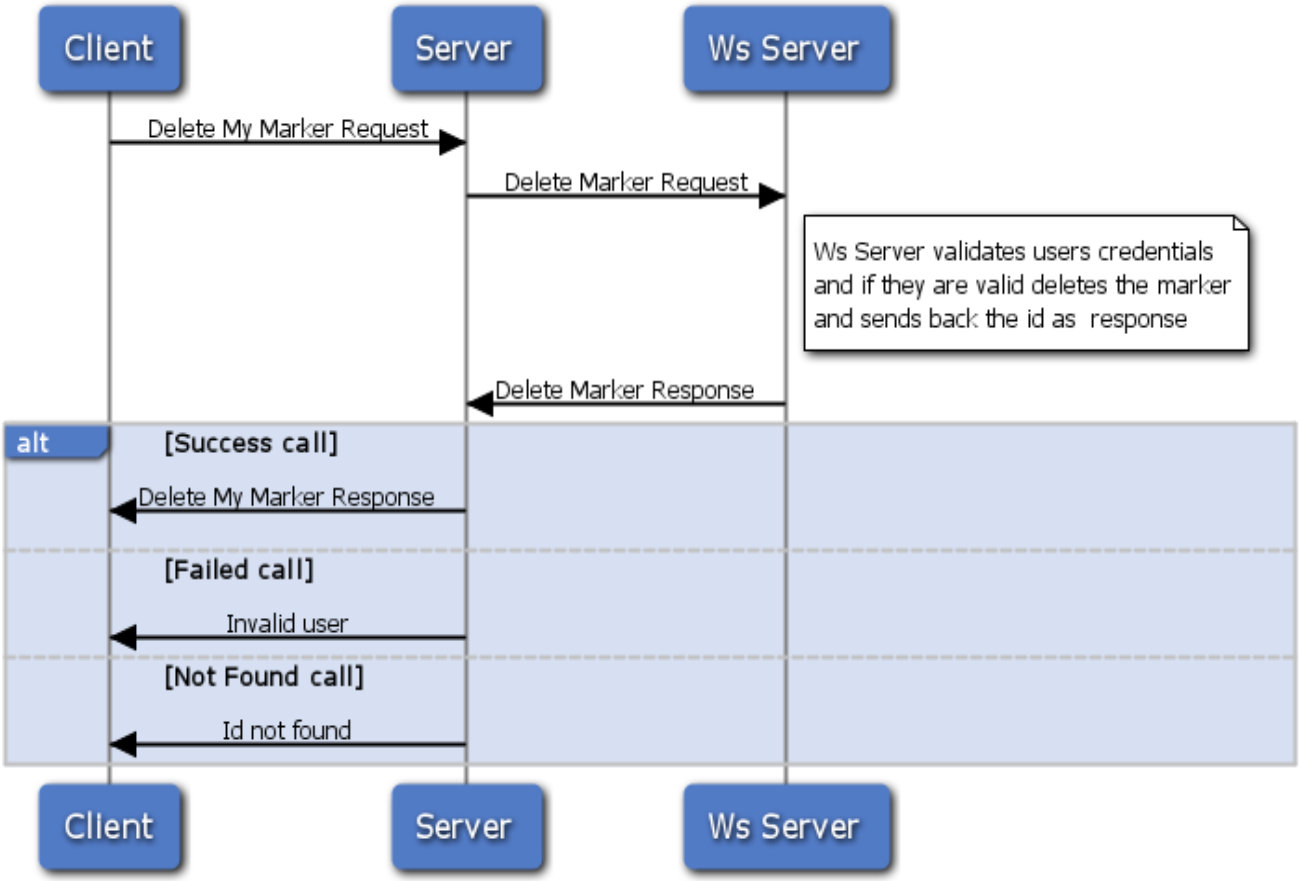
Στο διάγραμμα ακολουθίας της εικόνας 19, ο χρήστης στέλνει αίτημα για να παραλάβει την τρέχουσα θέση του καθώς και τα πιο κοντινά σημεία σε αυτήν.



Εικόνα 19: Msc σχήμα διαδικασίας αιτήματος για τρέχουσα θέση

5.6.5 Διαγραφή marker από το σύστημα

Στο διάγραμμα ακολουθίας της εικόνας 20, ο χρήστης στέλνει αίτημα για να διαγραφεί ο marker που έχει επιλέξει, ο server στέλνει στον χρήστη την απάντηση που έλαβε από τον web service server.

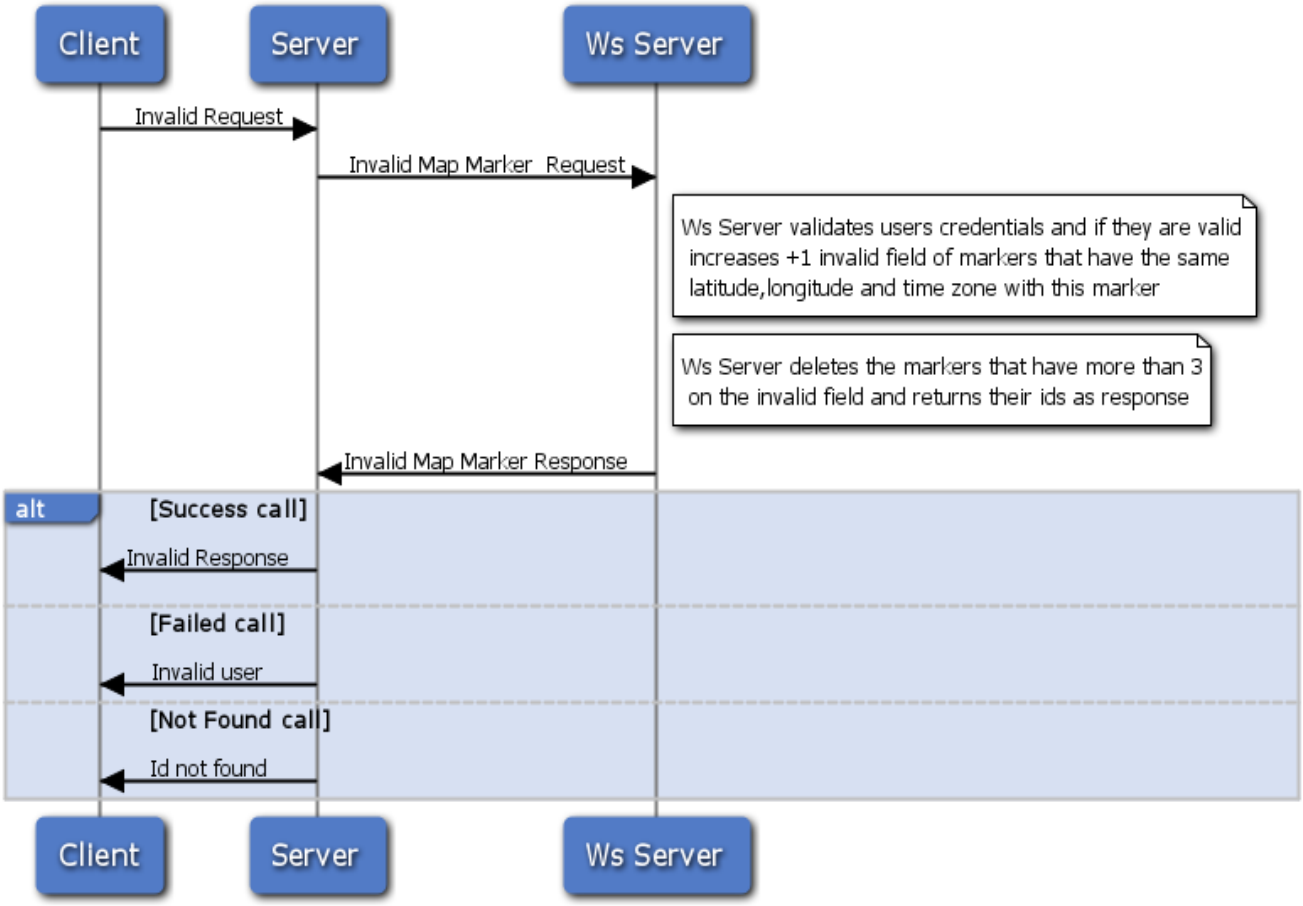


Εικόνα 20: Msc σχήμα διαδικασίας διαγραφής marker από το σύστημα

5.6.6 Διαδικασία ορισμού marker ως μη έγκυρου

Στο διάγραμμα ακολουθίας της εικόνας 21, ο χρήστης στέλνει αίτημα για να οριστεί ο marker που έχει

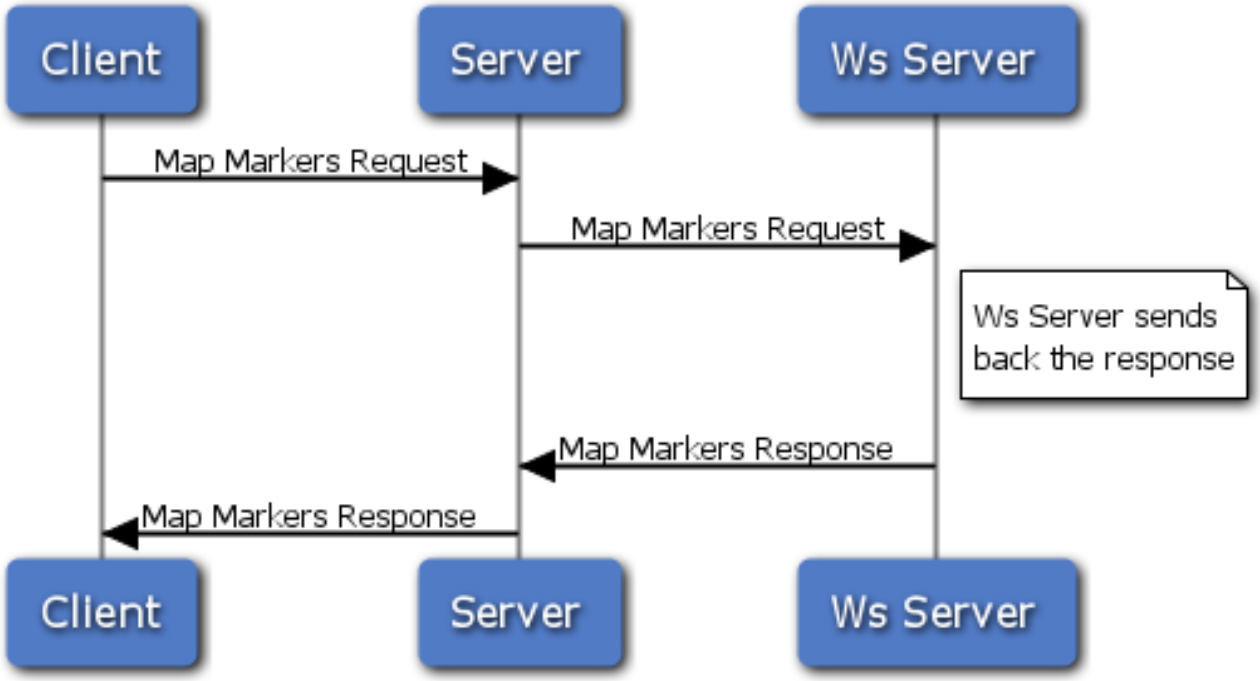
επιλέξει ως μη έγκυρος,, ο server στέλνει πίσω στον χρήστη την απάντηση που έλαβε από τον web service server.



Εικόνα 21: Msc σχήμα διαδικασίας μη έγκυρου marker

5.6.7 Αίτημα για λήψη των map markers

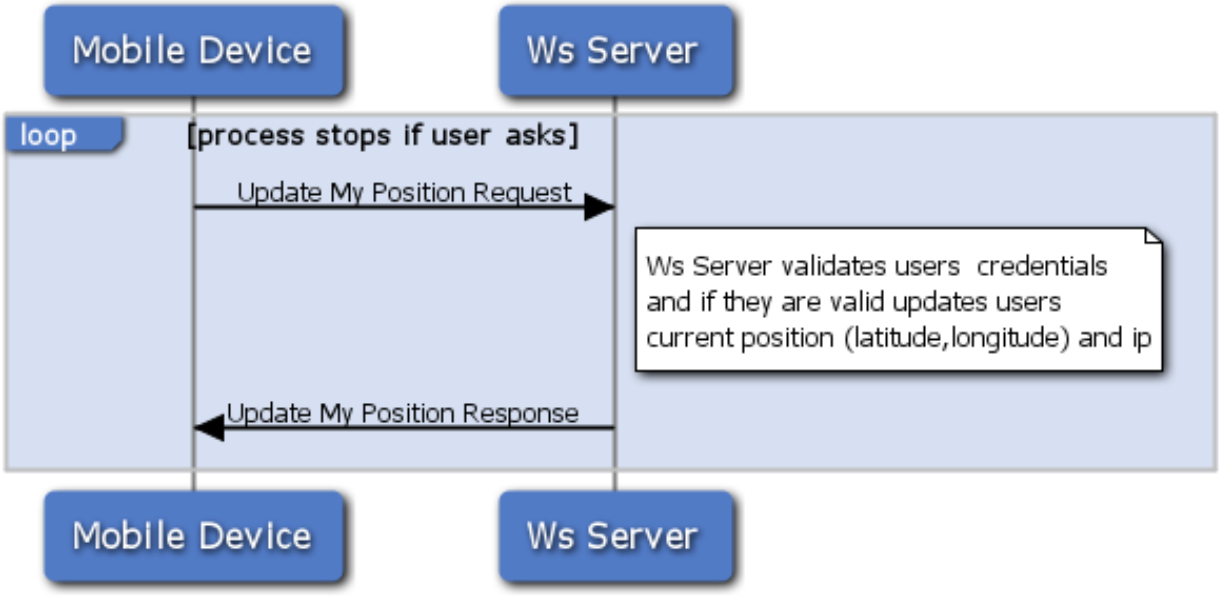
Στο διάγραμμα ακολουθίας της εικόνας 22, ο χρήστης στέλνει αίτημα για λήψη των markers και ο server απαντάει στέλνοντας πίσω την απάντηση που έλαβε από τον web service server.



Εικόνα 22: Msc σχήμα εμφάνισης map markers

5.6.8 Ανανέωση της τρέχουσας θέσης του χρήστη

Στο διάγραμμα ακολουθίας της εικόνας 23, ο χρήστης στέλνει αίτημα για να ανανεώσει την τρέχουσα θέση του.



Εικόνα 23: Msc σχήμα ανανέωσης της θέσης

6. Υλοποίηση

Παρακάτω παρουσιάζουμε τις τεχνολογίες καθώς και τα εργαλεία ή plugins που χρησιμοποιήσαμε. Ακόμα θα δώσουμε οδηγίες εγκατάστασης της εφαρμογής στις συσκευές που υποστηρίζουμε, και τέλος θα κάνουμε μια παρουσίαση της εφαρμογής.

6.1 Τεχνολογίες, γλώσσες, εργαλεία που χρησιμοποιήθηκαν στην υλοποίηση μας

Στην παρούσα εργασία χρησιμοποιήθηκαν αρκετές λύσεις ανοιχτού λογισμικού. Οι λύσεις αυτές περιγράφονται εκτενώς παρακάτω.

6.1.1 Τεχνολογίες, γλώσσες και εργαλεία που χρησιμοποιήθηκαν

Παρακάτω θα δούμε τις λύσεις που χρησιμοποιήσαμε:

1) Το linux λογισμικό το οποίο χρησιμοποιήσαμε είναι το arch linux
2) Ο κεντρικός κορμός της εργασίας γράφτηκε στο framework symfony v1.5.4 καθώς μετά από σύγκριση ανάμεσα στα υποψήφια frameworks, αποφασίσαμε ότι είναι αυτό που καλύπτει περισσότερο τις ανάγκες μας
3) Η βάση που χρησιμοποιήσαμε είναι η postgresql v8.4.4 καθότι υπερίσχυσε στο τέλος όσον αφορά την mysql
4) Το android sdk 1-r06 καθώς αποτελεί μονόδρομο για να γράψουμε εφαρμογές για το android
5) Το λειτουργικό σύστημα του android που χρησιμοποιήσαμε ήταν το 1.6 για να είμαστε συμβατοί με το λειτουργικό της κινητής μας συσκευής
6) Η χρήση του gpsd v2.39 αποτέλεσε μονόδρομο στο linux
7) Χρησιμοποιήσαμε τον apache server v2.2.15 επειδή αυτός καλύπτει περισσότερο τις ανάγκες μας
8) Php v5.3.2 για να γράψουμε ότι χρειάζεται και να εμπλουτίσουμε τον κεντρικό κορμό της εφαρμογής μας
9) Soap τεχνολογία σε συνδυασμό με τα web services
10) Χρησιμοποιήσαμε την scripting γλώσσα python v2.6.5
11) Ajax για να μπορούμε να κάνουμε calls χωρίς να αλλάζει το content της σελίδας
12) Προτιμήσαμε το google maps api σε σχέση με το yahoo maps api για την υλοποίηση μας, καθώς υπάρχουν και οι αντίστοιχες βιβλιοθήκες στο android. Ακόμα διαλέξαμε το google maps api v3 σε σχέση με το v2. Το v3 είχε λιγότερο functionality σε σχέση με το v2 αλλά το προτιμήσαμε καθότι ήταν το πιο καινούργιο
13) Αντί να γράψουμε raw javascript προτιμήσαμε το javascript framework jquery v1.4.2 το οποίο μας βοηθάει με τα compatibility issues όσον αφορά τους browsers
14) Χρησιμοποιήθηκε soapui v3.5 για να ελέγξουμε τα web services τα οποία χρησιμοποιούμε
15) Dia v0.97 για την σχεδίαση των μοντέλων της βάσης (σχεσιακό μοντέλο, οντοτήτων συσχετίσεων, περιπτώσεις χρήσης και διαγράμματα ακολουθίας μηνυμάτων)
16) Το inkscape v0.47 για την δημιουργία εικονιδίων
17) Το gimp v2.6.8 για την επεξεργασία εικόνων

6.1.2 Plugins και βιβλιοθήκες που χρησιμοποιήθηκαν για το symfony

framework

Για την κεντρική εφαρμογή χρησιμοποιήσαμε ορισμένα plugins και βιβλιοθήκες, για ότι δεν υπήρχε κάτι έτοιμο υλοποιήσαμε τα δικά μας. Παρακάτω παρουσιάζουμε τις λύσεις που χρησιμοποιήσαμε – υλοποιήσαμε :

- **sfXMapsPlugin, π**ρόκειται για το plugin το οποίο υλοποιήσαμε και εκτελεί όλες τις βασικές λειτουργίες της εφαρμογής μας που έχουν να κάνουν με τον χάρτη
- **ckWebServicePlugin,** το συγκεκριμένο plugin χρησιμοποιήθηκε για να δημιουργήσουμε τις web service λειτουργίες της εφαρμογής. Ο κώδικας του περιλάμβανε ορισμένα bugs τα οποία και χρειάστηκε να επιλύσουμε προκειμένου να δουλέψει ο web service server
- **SfXSoap,** χρειάστηκε να υλοποιήσουμε την συγκεκριμένη κλάση από το μηδέν για να έχουμε διασύνδεση με τον web service server μέσα από την εφαρμογή
- **sfDoctrineApplyPlugin,sfDoctrineGuardPlugin,** τα συγκεκριμένα plugins χρησιμοποιήθηκαν και είναι υπεύθυνα για την διαχείριση των λογαριασμών των χρηστών, καθώς και την είσοδο και την έξοδο από το σύστημα
- **sfJqueryReloadedPlugin,** προσφέρει ενσωμάτωση με το jquery api καθώς και διάφορες βασικές λειτουργίες
- **sfXHelpersPlugin,** δικό μας plugin που περιλαμβάνει γενικές λειτουργίες και κλάσεις που χρησιμοποιούνται μέσα στην εφαρμογή

6.1.3 Modules που χρησιμοποιήθηκαν για την εφαρμογή python του υπολογιστή

Για την python εφαρμογή, χρειάστηκαν ορισμένα modules τα οποία και παρουσιάζουμε παρακάτω :

- **gps,** πρόκειται για το module του gpsd που περιέχει τα python bindings για να διασυνδέσουμε την συσκευή μας με τον gpsd daemon
- **optparse**, χρησιμοποιήθηκε για να διαβάσουμε τα ορίσματα που στέλνει ο χρήστης στην εφαρμογή
- **suds**, μας παρέχει έναν ελαφρύ client για web services της εφαρμογής μας

6.1.4 Βιβλιοθήκες που χρησιμοποιήθηκαν για την εφαρμογή client της android συσκευής

Για την android εφαρμογή, χρειάστηκε να χρησιμοποιήσουμε τις εξής βιβλιοθήκες :

- **ksoap2-android,** πρόκειται για μια βιβλιοθήκη που χρησιμοποιήσαμε στο android και περιλαμβάνει functions για να μπορέσουμε να καλέσουμε τις λειτουργίες του web service server
- **kxml2,** περιλαμβάνει κλάσεις και λειτουργίες για την δημιουργία xml elements από την java. Χρησιμοποιήθηκε για να δημιουργήσουμε την κεφαλή αυθεντικοποίησης

6.2 Hardware που χρησιμοποιήθηκε

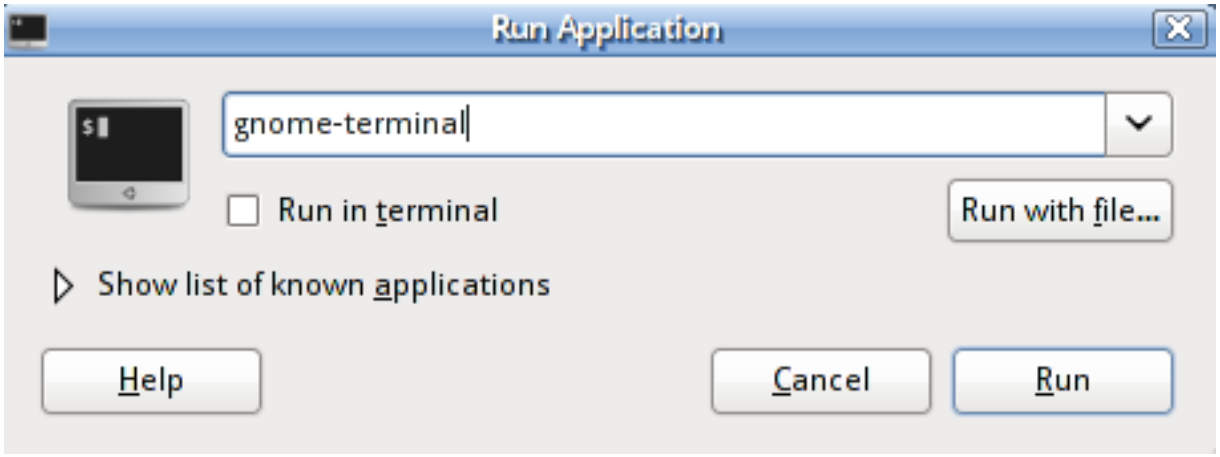
Για την συγκεκριμένη υλοποίηση χρειαστήκαμε ορισμένες συσκευές και τις παρουσιάζουμε παρακάτω:

- **Asus EEE PC 1000HE,** πρόκειται για το netbook που χρησιμοποιήσαμε για να κάνουμε τις δοκιμές μας
- **GlobalSat BU-353 USB Δέκτης GPS,** πρόκειται για τον gps δέκτη που χρησιμοποιήσαμε. Ο συγκεκριμένος έχει υποδοχή usb και όχι bluetooth καθώς αυτήν προτιμήσαμε
- **Android emulator,** χρησιμοποιήσαμε τον android emulator με την πλατφόρμα android 1.6 που μας παρέχεται από το sdk του android.
- **Pentium 4,** πρόκειται για τον κεντρικό μας υπολογιστή ο οποίος χρησιμοποιήθηκε ως web service server και server για την εφαρμογή μας

6.3 Εγκατάσταση εφαρμογής client και usb gps receiver σε υπολογιστή

Βήμα 1ο

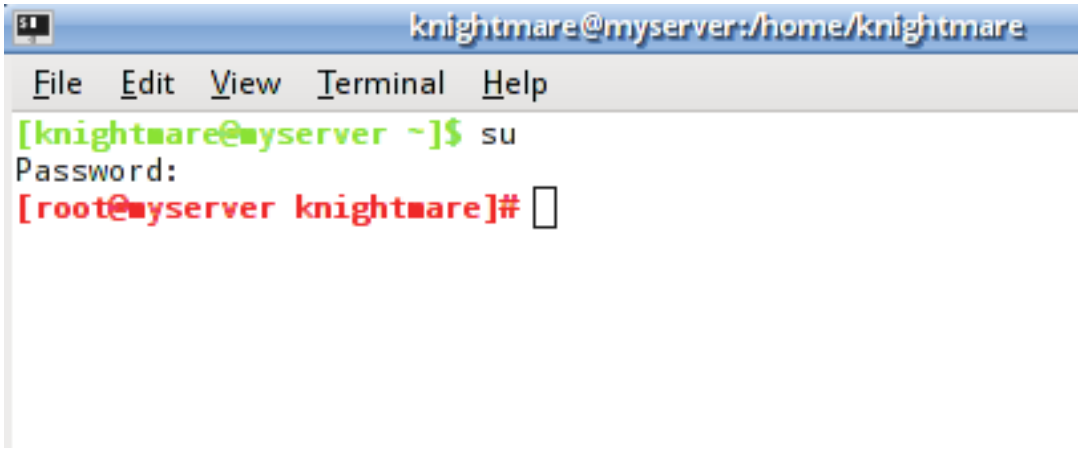
Πατάμε alt+f2 στο keyboard εμφανίζεται το run application στο οποίο γράφουμε gnome-terminal αν είμαστε σε gnome ή konsole αν είμαστε σε kde.



Εικόνα 24: Πρόγραμμα εκτέλεσης λειτουργιών

Βήμα 2ο

Πατάμε su και βάζουμε το password του root (ο administrator στο linux).



Εικόνα 25: Αλλαγή σε χρήστη root

Βήμα 3ο

Κάνουμε add το γκρουπ gps (ή το group της επιλογής μας).

```
[root@myserver knightmare]# groupadd gps
```

Βήμα 4ο

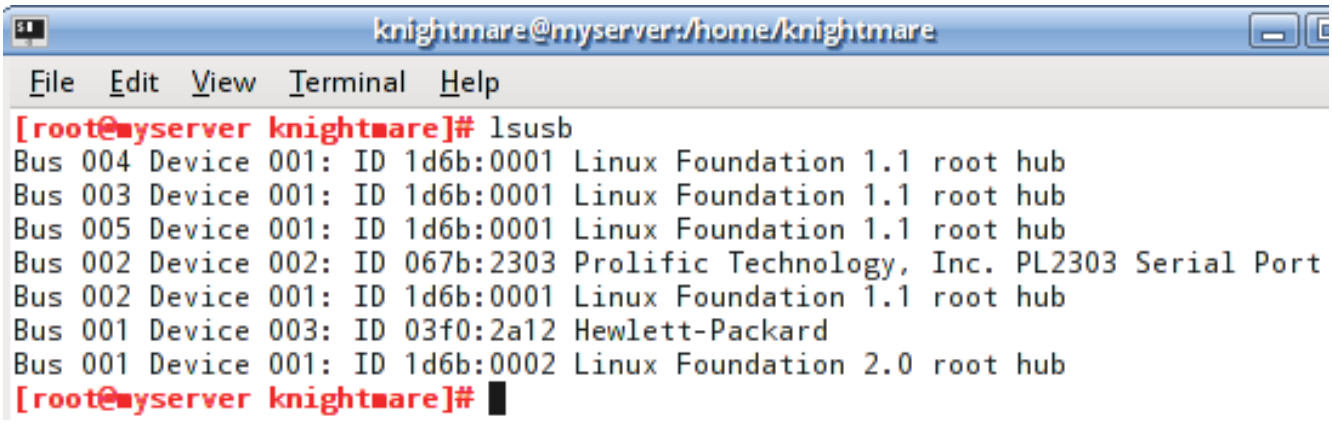
Στην περίπτωση που δεν έχουμε εγκατεστημένο το gpsd το κάνουμε install.

```
[root@myserver knightmare]# pacman -S gpsd
```

Πατάμε y στο proceed with installation

Βήμα 5ο

Τρέχουμε lsusb για να πάρουμε τις πληροφορίες των συνδεδεμένων usb συσκευών



Εικόνα 26: Εμφάνιση usb συσκευών του συστήματος

Στην περίπτωση μας η συσκευή μας είναι η εξής

Bus 002 Device 002: ID 067b:2303 Prolific Technology, Inc. PL2303 Serial Port

- Ο κωδικός 067b αποτελεί τον vendor κωδικό
- Ο κωδικός 2303 αποτελεί τον product κωδικό

και οι δυο κωδικοί θα μας χρειαστούν παρακάτω

Βήμα 6ο

Ανοίγουμε το conf file των gpsd usb rules

```
[root@myserver knightmare]# nano -w /etc/udev/rules.d/99-gpsd-usb.rules
```

```
# udev rules for the gpsd USB hotplugging

SUBSYSTEM!="tty", GOTO="gpsd-usb_rules_end"


ATTRS{idVendor}=="067b", ATTRS{idProduct}=="2303", SUBSYSTEM=="tty", ACTION=="add",MODE="664" , GROUP="gps", KERNEL=="ttyUSB[0-9]*", SYMLINK="gps%n", RUN+="/etc/hotplug/usb/gpsd add $root/%k"

ATTRS{idVendor}=="067b", ATTRS{idProduct}=="2303", SUBSYSTEM=="tty", ACTION=="remove", MODE="664" , GROUP="gps",KERNEL=="ttyUSB[0-9]*", RUN+="/etc/hotplug/usb/gpsd remove $root/%k"
```

```
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SUBSYSTEM=="tty", ACTION=="add", KERNEL=="ttyUSB[0-9]*", SYMLINK="gps%n", RUN+="/etc/hotplug/usb/gpsd add $root/%k"

ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SUBSYSTEM=="tty", ACTION=="remove", KERNEL=="ttyUSB[0-9]*", RUN+="/etc/hotplug/usb/gpsd remove $root/%k"

ATTRS{idVendor}=="1163", ATTRS{idProduct}=="0100", SUBSYSTEM=="tty", ACTION=="add", KERNEL=="ttyUSB[0-9]*", SYMLINK="gps%n", RUN+="/etc/hotplug/usb/gpsd add $root/%k"

ATTRS{idVendor}=="1163", ATTRS{idProduct}=="0100", SUBSYSTEM=="tty", ACTION=="remove", KERNEL=="ttyUSB[0-9]*", RUN+="/etc/hotplug/usb/gpsd remove $root/%k"

ATTRS{idVendor}=="067b", ATTRS{idProduct}=="aaa0", SUBSYSTEM=="tty", ACTION=="add", KERNEL=="ttyUSB[0-9]*", SYMLINK="gps%n", RUN+="/etc/hotplug/usb/gpsd add $root/%k"

ATTRS{idVendor}=="067b", ATTRS{idProduct}=="aaa0", SUBSYSTEM=="tty", ACTION=="remove", KERNEL=="ttyUSB[0-9]*", RUN+="/etc/hotplug/usb/gpsd remove $root/%k"

ATTRS{idVendor}=="091e", ATTRS{idProduct}=="0003", SUBSYSTEM=="tty", ACTION=="add", KERNEL=="ttyUSB[0-9]*", SYMLINK="gps%n", RUN+="/etc/hotplug/usb/gpsd add $root/%k"

ATTRS{idVendor}=="091e", ATTRS{idProduct}=="0003", SUBSYSTEM=="tty", ACTION=="remove", KERNEL=="ttyUSB[0-9]*", RUN+="/etc/hotplug/usb/gpsd remove $root/%k"

#ATTRS{idVendor}=="", ATTRS{idProduct}=="", SUBSYSTEM=="tty", ACTION=="add", KERNEL=="ttyUSB[0-9]*", SYMLINK="gps%n", RUN+="/etc/hotplug/usb/gpsd add $root/%k"

#ATTRS{idVendor}=="", ATTRS{idProduct}=="", SUBSYSTEM=="tty", ACTION=="remove", KERNEL=="ttyUSB[0-9]*", RUN+="/etc/hotplug/usb/gpsd remove $root/%k"

LABEL="gpsd-usb_rules_end"
```

α) Ψάχνουμε για idVendor = 067b (το δικό μας)
β) ψάχνουμε για idProduct = 2303 (το δικό μας)

Προσθέτουμε την γραμμή μετά το ACTION και πριν το KERNEL

```
 MODE="664" , GROUP="gps",
```

Όπου group το group της επιλογής μας, στην περίπτωση μας gps.

Βήμα 7ο

Αποθηκεύουμε και κλείνουμε το αρχείο που ανοίξαμε

```
ctrl+o για να σώσουμε
ctrl+x για να κλείσουμε
```

Βήμα 8ο

Κάνουμε add τον χρήστη στο γκρουπ της επιλογής μας (στην περίπτωση μας gps), η εντολή που πρέπει να εκτελέσουμε είναι της μορφής gpasswd -a username group

```
[root@myserver knightmare]# gpasswd -a knightmare gps
```

Βήμα 9ο

Κάνουμε restart την εφαρμογή gpsd

```
[root@myserver knightmare]# /etc/rc.d/gpsd start
```

Βήμα 10ο

Αν όλα πήγαν καλά, παίρναμε το service έτσι ώστε να μπορεί να εκτελείται αυτόματα.

```
[root@myserver knightmare]# nano -w /etc/rc.conf
```

Εντοπίζουμε την γραμμή DAEMONS και γράφουμε την λέξη gpsd μέσα στις παρενθέσεις.

Βήμα 11ο

Εκτελούμε την παρακάτω εντολή από την κονσόλα

```
[root@myserver python]# crontab -e
```

Βήμα 12ο

Προσθέτουμε την παρακάτω γραμμή στο crontab, στην περίπτωση μας το script gipi.py βρίσκεται στο path /home/programming/python. Στο --username admin βάζουμε το username του χρήστη που μας έχει αποδοθεί. πχ admin και στο –password το password που μας έχει αποδοθεί. Πχ unipi.

```
/home/programming/python/gipi.py --username admin --password unipi
```

Βήμα 13ο

Κρατώντας πατημένο το shift, πατάμε το κουμπί : (άνω και κάτω τελεία) και γράφουμε wq (αποθήκευση και κλείσιμο αρχείου).

6.4 Εγκατάσταση εφαρμογής client στον android emulator

Για να εκτελεστεί ο android client που έχουμε δημιουργήσει για την εφαρμογή χρειάστηκε να εγκαταστήσουμε το android-sdk. Παρακάτω θα περιγράψουμε τα βήματα που χρειάζεται να εκτελεστούν

Βήμα 1ο

Γινόμαστε root και εγκαθιστούμε το android-sdk.

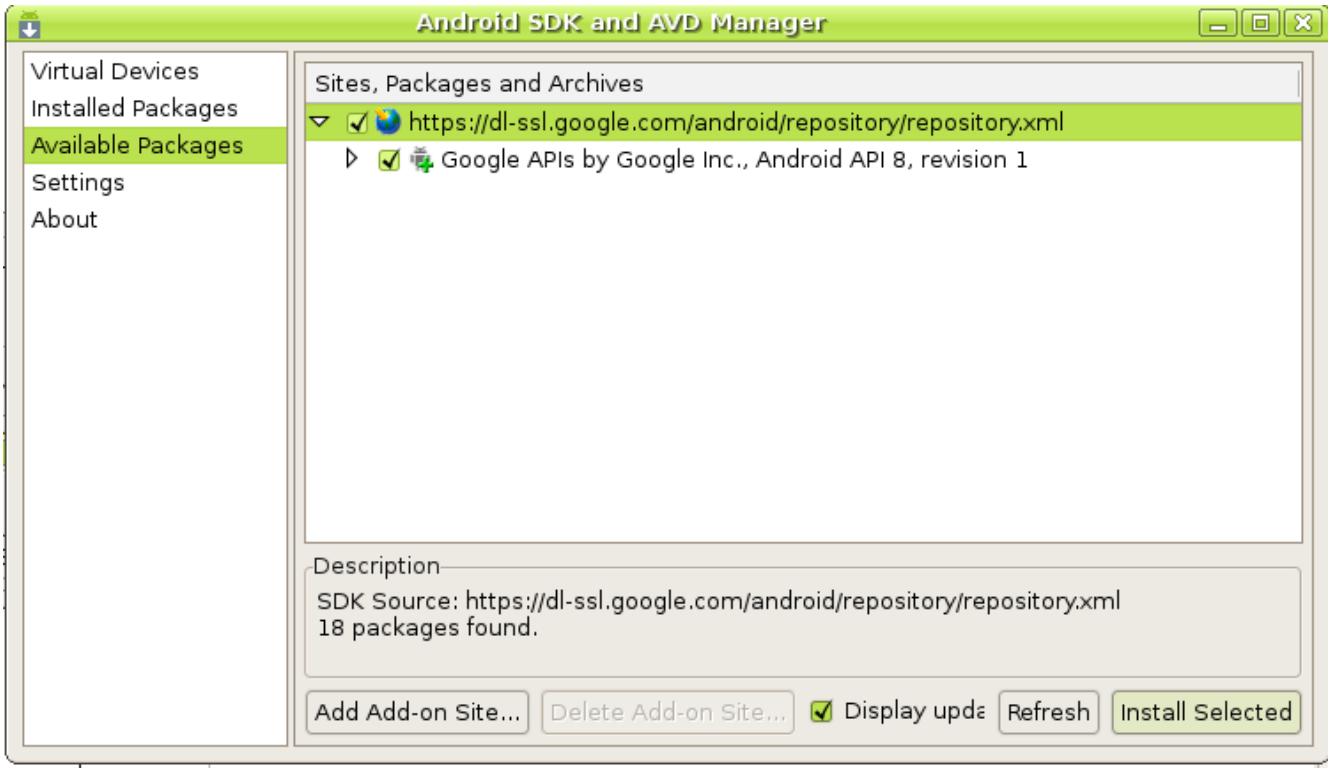
Εάν όλα πήγαν καλά και η εγκατάσταση μας είναι έτοιμη περνάμε στο επόμενο βήμα

Βήμα 2ο

Εμφανίζουμε την κονσόλα μας, πατώντας alt+f2 και εκτελώντας την εντολή android εκτελείται ο android SDK and AVD Manager

Βήμα 3ο

Αφού εμφανιστεί το gui του android-sdk πατάμε στα αριστερά του μενού το Available Packages. Κάνουμε tick το κεντρικό κουτάκι κάτω από το Sites,Packages and Archives (δίπλα από τις αναβαθμίσεις επιλέγονται όλες οι επιλογές) και πατώντας τέλος το install selected (accept και install στις επιλογές που εμφανίζονται) ξεκινάει η αναβάθμιση του manager.



Εικόνα 27: Ο manager του android

Όταν τελειώσει η συγκεκριμένη διαδικασία ο android manager μας θα είναι ανανεωμένος και είμαστε έτοιμοι να δημιουργήσουμε το πρώτο μας εικονικό android που θα χρειαστούμε παρακάτω.

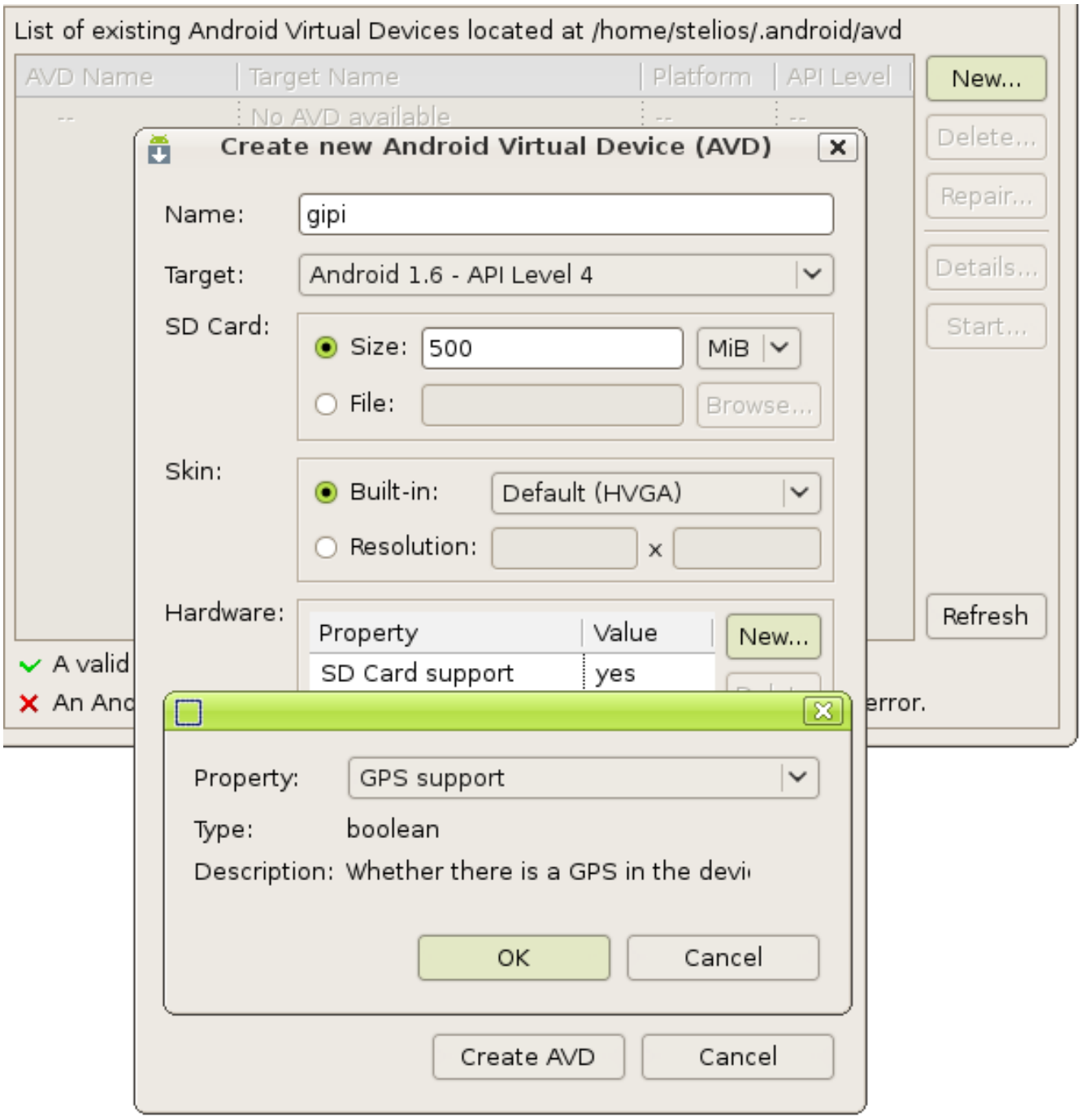
Βήμα 4ο

Πηγαίνουμε στα virtual devices και επιλέγουμε new, οι επιλογές που ορίζουμε είναι οι εξής

- Στο name βάζουμε gipi (ή το όνομα της επιλογής μας)
- Στο target διαλέγουμε το android 1.6 (η έκδοση android που χρησιμοποιούμε)
- Στο SD Card size βάζουμε 500 ,αφήνουμε την επιλογή δίπλα του σε Mib
- Από κάτω στο hardware πατάμε new βρίσκουμε το property GPS support το επιλέγουμε και πατάμε ok
- Ακολουθούμε την ακριβώς ίδια διαδικασία και επιλέγουμε το SD Card Support

Τέλος πατάμε Create AVD και εμφανίζετε ένα παράθυρο που μας ενημερώνει ότι η εικονική συσκευή μας είναι έτοιμη και μπορούμε να την ενεργοποιήσουμε.

Τα βήματα φαίνονται και στην εικόνα 28 που ακολουθεί.



Εικόνα 28: Δημιουργία συσκευής στον manager του android

Βήμα 5ο

Εγκαθιστούμε το maven για να κάνουμε build την εφαρμογή.

```
[root@myhost stelios]# pacman -S maven
```

Βήμα 6ο

Εκτελούμε τις παρακάτω εντολές για να μπορεί η εφαρμογή να έχει πρόσβαση στο android.jar

Πηγαίνουμε στον φάκελο της επιλογής μας και αντιγράφουμε το tar.gz που μας δίνετε .

Για να το κάνουμε extract

```
tar -xvzf MyGipiAndroidProject.tar.gz
```

κάνουμε export τα παρακάτω variable PATHS από κονσόλα για τον χρήστη με τον οποίο είμαστε συνδεδεμένοι.

```
export ANDROID_HOME=/opt/android-sdk
export PATH=$PATH:$ANDROID_HOME
```

Δημιουργούμε το μονοπάτι του φακέλου για να αντιγράψουμε τα αρχεία

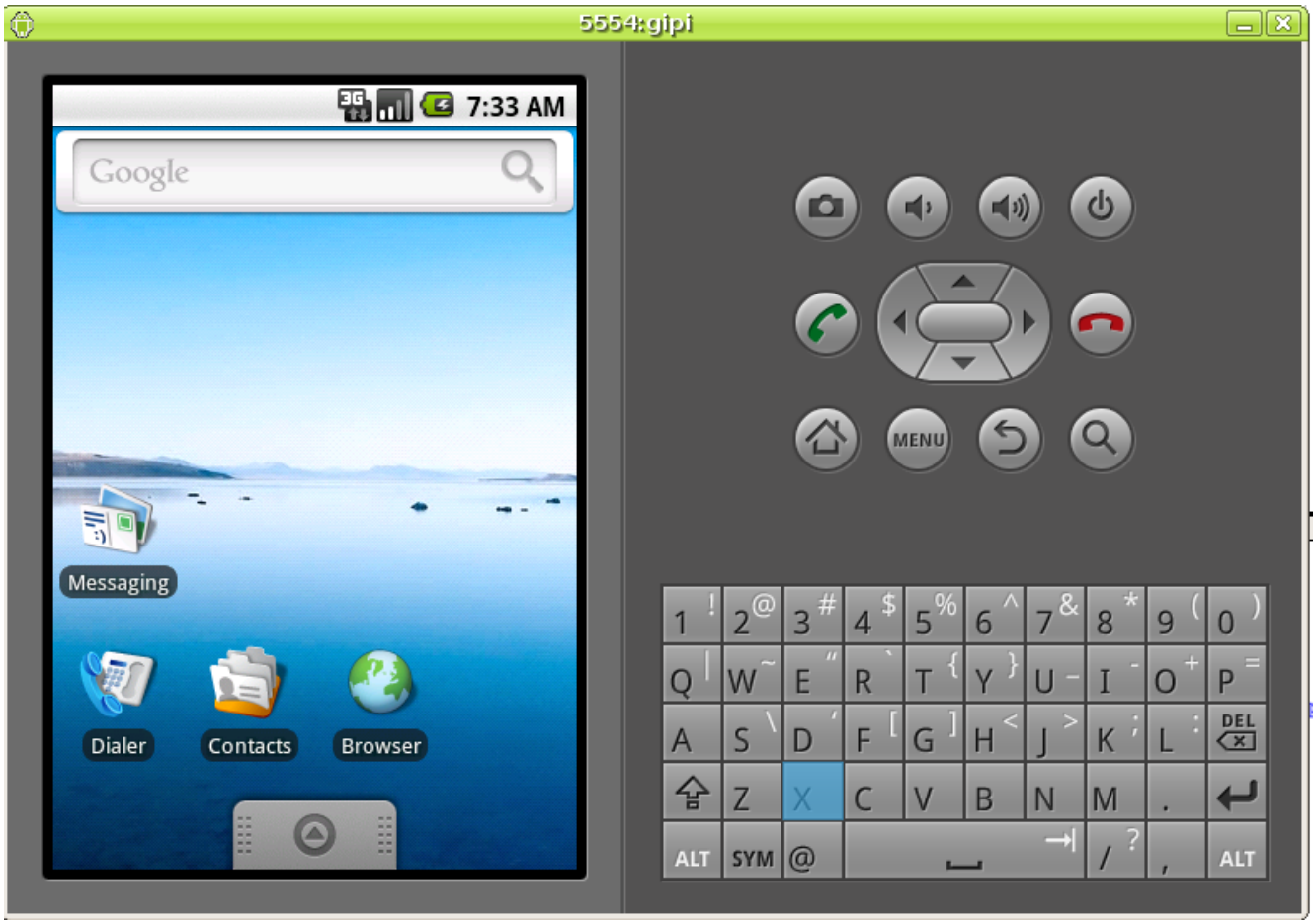
```
mkdir -p ~/.m2/repository/android/android/1.6_r3
```

Αντιγράφουμε την βιβλιοθήκη του android

```
cp android-1.6_r3.jar ~/.m2/repository/android/android/1.6_r3/
```

Βήμα 7ο

Πηγαίνουμε στα Virtual Devices (Android manager), επιλέγουμε την συσκευή που μόλις δημιουργήσαμε και πατάμε Start και χωρίς να επιλέξουμε κάτι launch. Περιμένουμε μέχρι να φορτώσει το android λειτουργικό.



Εικόνα 29: Η αρχική οθόνη της εικονικής συσκευής μας

Περιμένουμε μέχρι να ανοίξει η εικονική μας συσκευή και μπαίνουμε στο path της εφαρμογής μας.

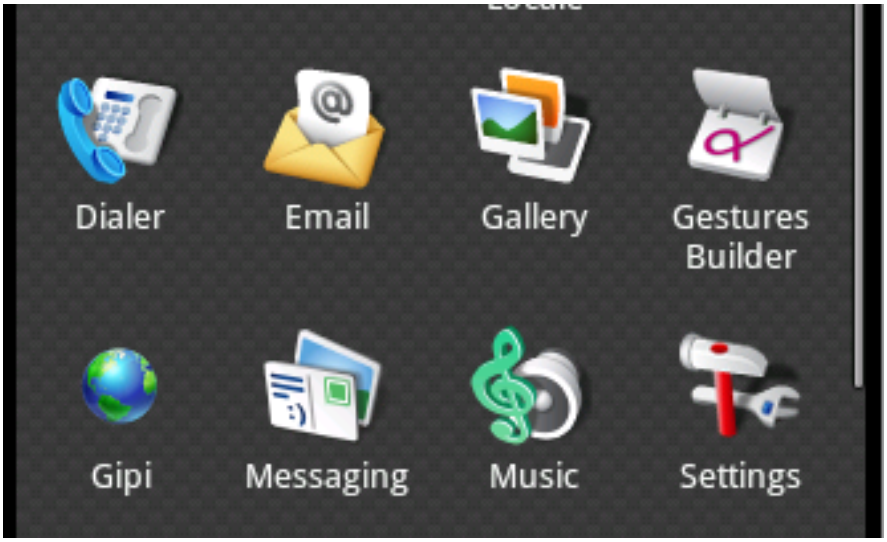
Βήμα 8ο

```
cd MyGipiAndroidProject
```

Και εκτελούμε

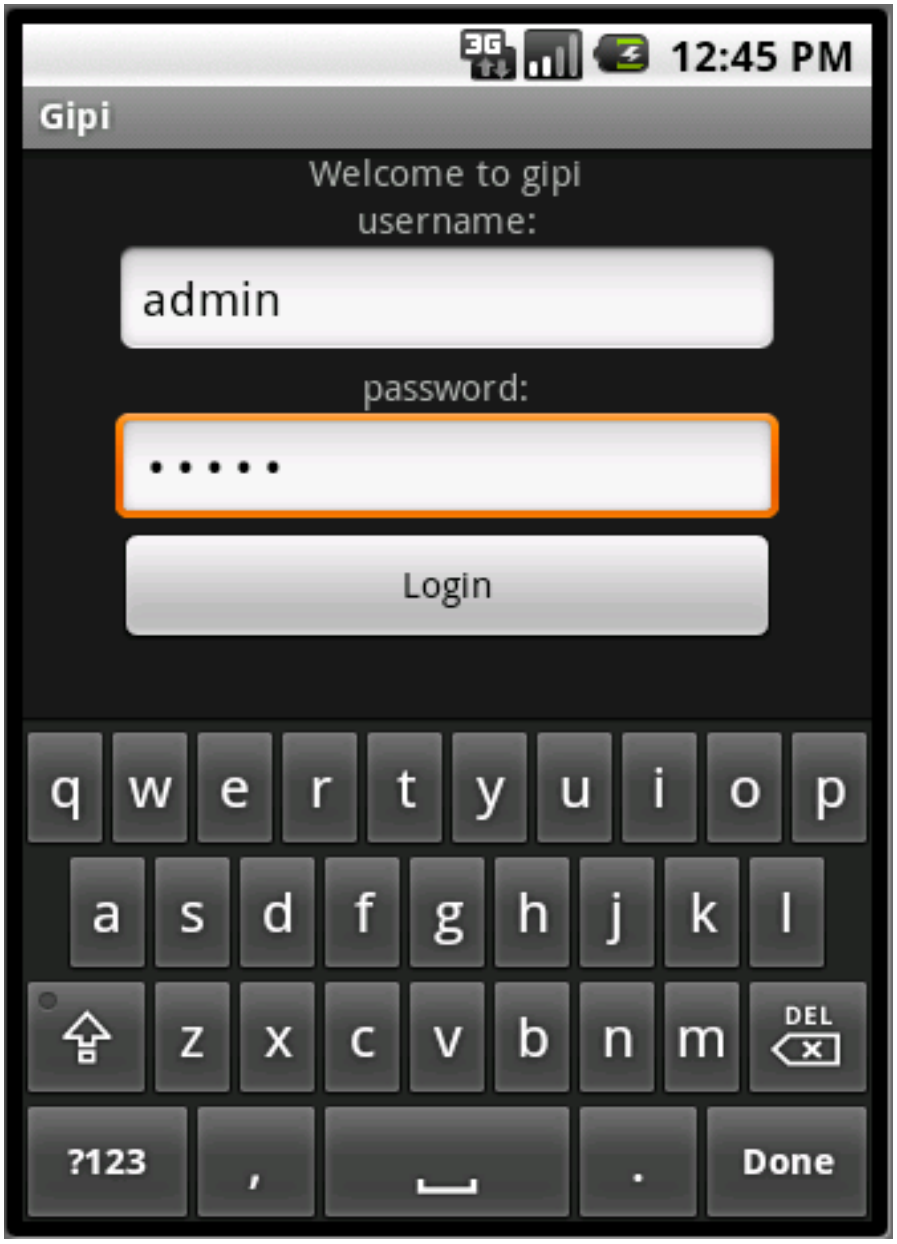
```
./deploy
```

Εάν όλα πήγαν καλά στην εικονική μας συσκευή πατώντας το βέλος (το γκρι κουμπί με το βέλος) θα δούμε την εφαρμογή μας (Gipi).



Εικόνα 30: Η εφαρμογή μας Gipi μαζί με τις υπόλοιπες

Επιλέγουμε την εφαρμογή και εμφανίζετε η login screen του προγράμματος



Εικόνα 31: Η οθόνη σύνδεσης της εφαρμογής μας για το android

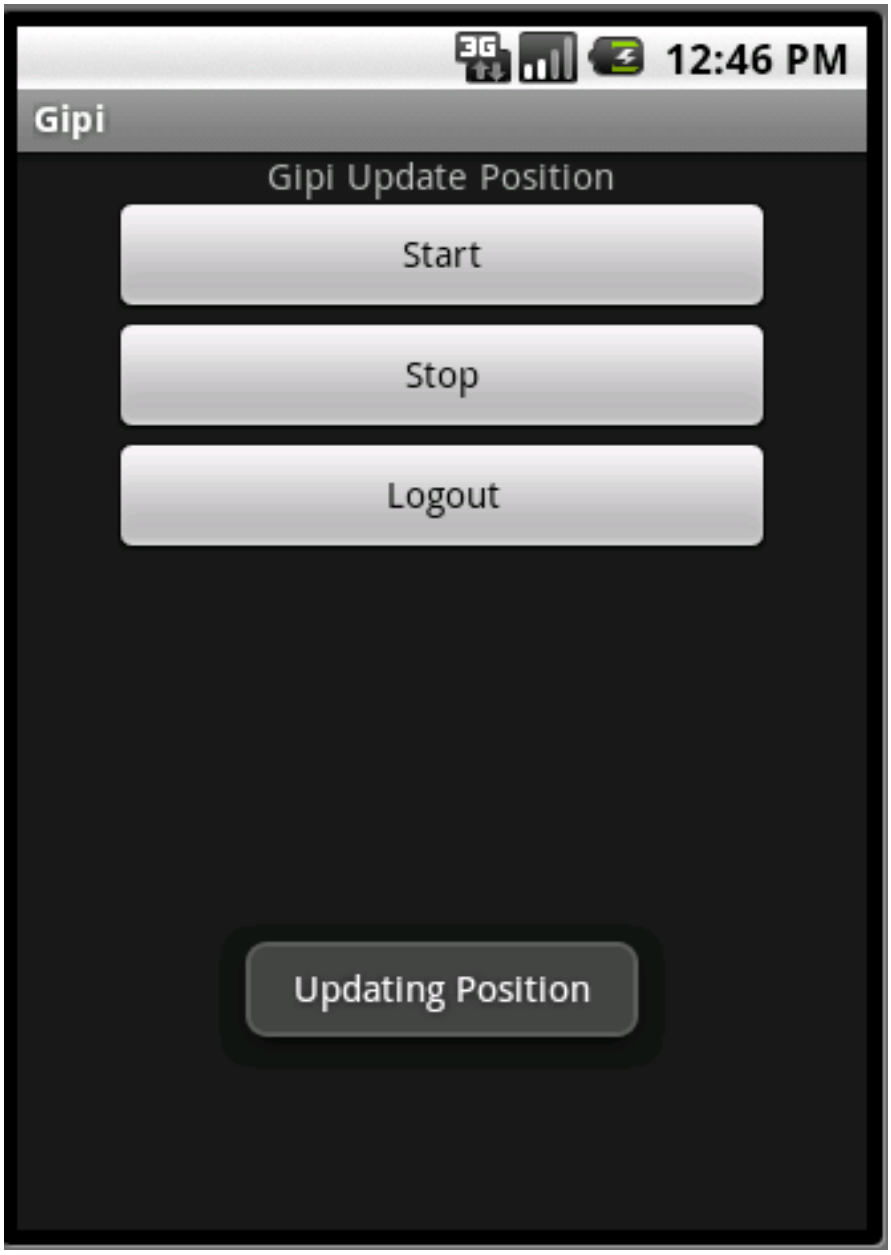
Κάνοντας login η εφαρμογή εκτελεί το web service validate login το οποίο μας ενημερώνει με κατάλληλα μηνύματα για το αν βρέθηκε ο χρήστης ή όχι.

Αφού συνδεθούμε οι επιλογές που μας δίνονται οι εξής

- **Start,** ορίζει ένα interval στο οποίο θα ανανεώνεται η τοποθεσία του χρήστη. Η κλάση που καλείται είναι η UpdateMyPosition που με την σειρά της καλεί την λειτουργία του web service της εφαρμογής την UpdateMyPosition στην οποίο και δίνει τις κατάλληλες μεταβλητές
- **Stop,** σβήνει το interval, Οπότε και σταματάει να ανανεώνεται η θέση του χρήστη

- **Logout,** αποσυνδέει τον χρήστη από τον client

Παρακάτω βλέπουμε την εφαρμογή την στιγμή που ο χρήστης έχει πατήσει το start και έχει ξεκινήσει να εκτελείται η ανανέωση της θέσης.



Εικόνα 32: Εσωτερική οθόνη της εφαρμογής μας για το android

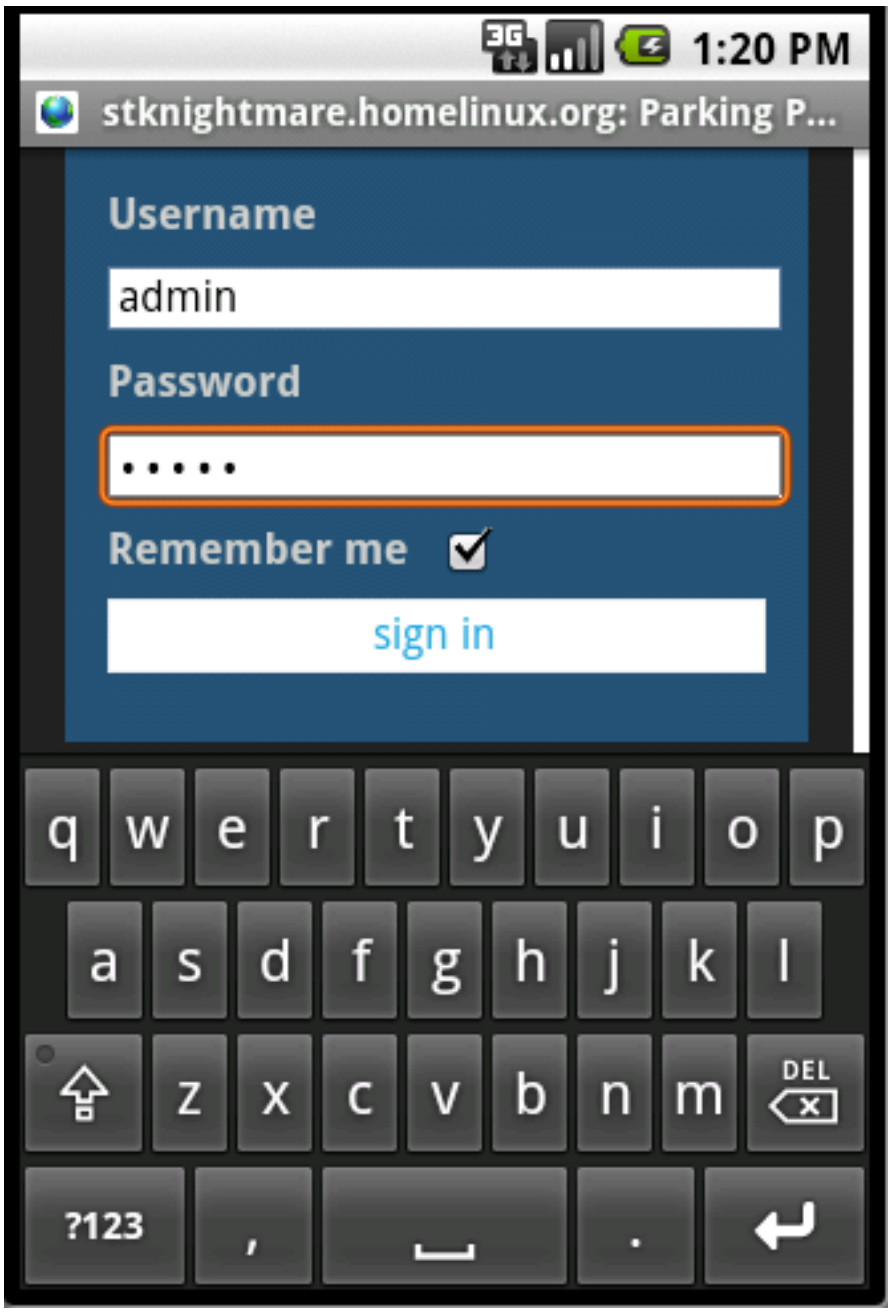
6.5 Παρουσίαση εφαρμογής από το android

Αφήνοντας το service να εκτελείται πατάμε το κουμπί που έχει το εικονίδιο σπίτι από τα δεξιά του

emulator. Μετά πατάμε τον browser. Πάλι από τα δεξιά πατάμε τον μεγεθυντικό φακό, στην μπάρα που προκύπτει βάζουμε το url της εφαρμογής

http://stknightmare.homelinux.org:30000/gipi

και εμφανίζεται η παρακάτω φόρμα στην οποία εισάγουμε τα στοιχεία μας.



Εικόνα 33: Η φόρμα σύνδεσης για την εφαρμογή μας

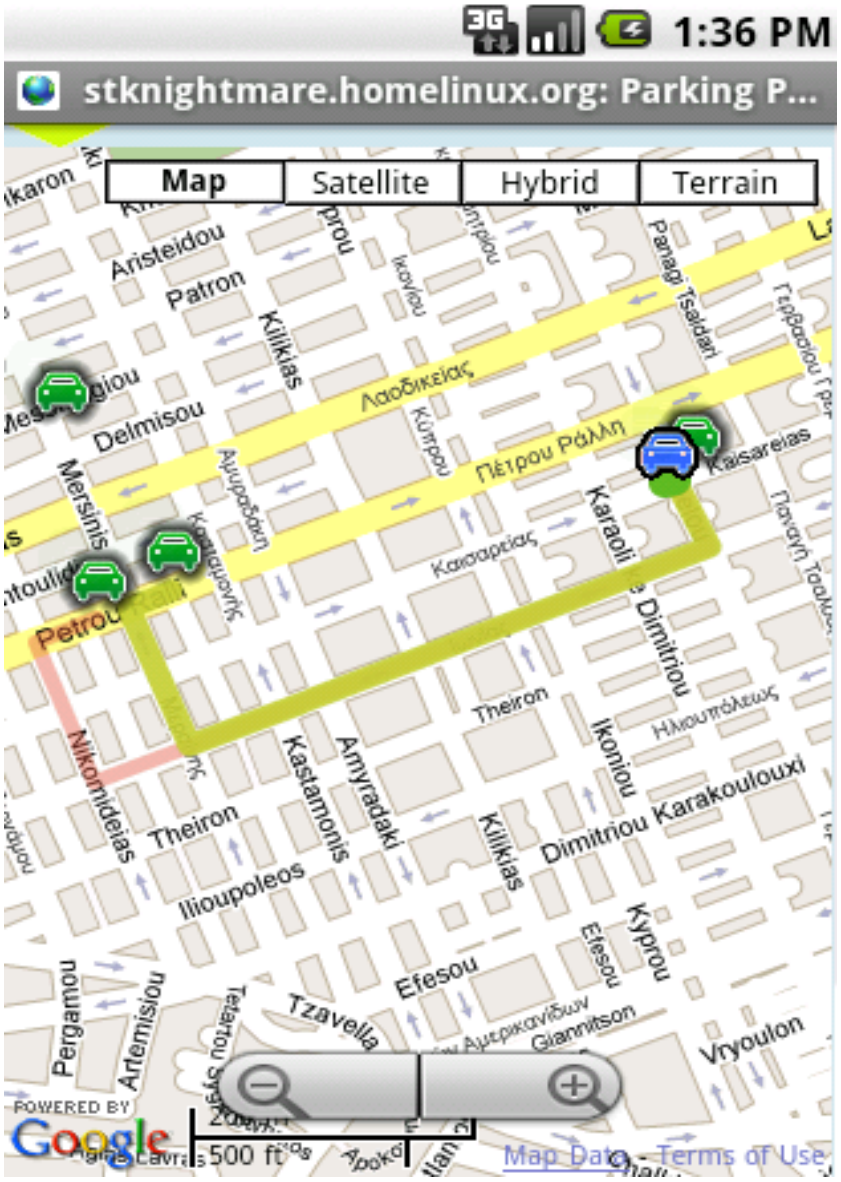
Αφού εισάγουμε έγκυρα στοιχεία, το σύστημα μας κατευθύνει στον χάρτη και στην εικόνα που

ακολουθεί.

Βλέπουμε τις 3 θέσεις που προτείνονται στον χρήστη και είναι πιθανό ότι αποτελούν ελεύθερες θέσεις. Επίσης βλέπουμε τις οδηγίες που δίνονται στον χρήστη για το πως να φτάσει μέχρι εκεί.

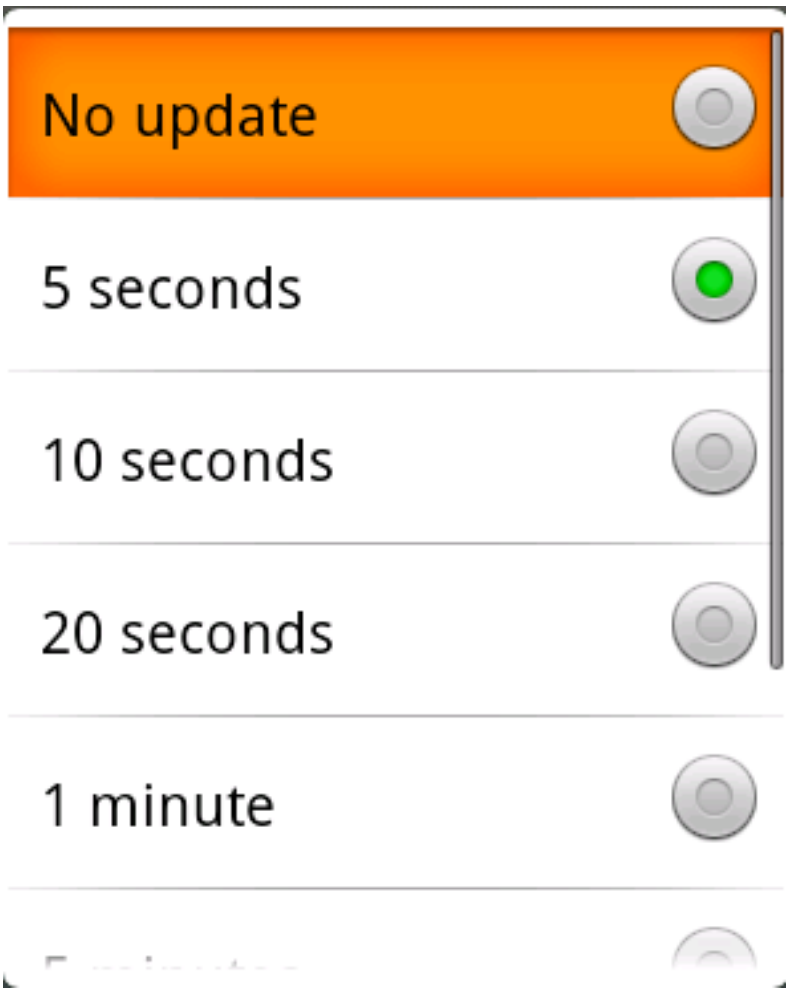
Εικόνα 34: Η εφαρμογή μας όπως εμφανίζεται στην οθόνη του android

Παρακάτω δείχνουμε και μια εικόνα από πιο κοντινό zoom



Εικόνα 35: Κοντινό ζουμ της εφαρμογής μας

Τέλος στην εικόνα 36 δείχνουμε τις επιλογές της εφαρμογής που εμφανίζονται όταν πατήσουμε τα διάφορα κουμπιά επιλογών, όπως είναι το current position



Εικόνα 36: Το σύστημα επιλογών στο android

6.6 Έλεγχος του web service με την χρήση του soapui

Βήμα 1ο

Ανοίγουμε τον browser της επιλογής μας (firefox) και στην διεύθυνση του browser εισάγουμε, την διεύθυνση του web service. η απάντηση που πρέπει να πάρουμε φαίνεται στην εικόνα 37.

```
<wsdl:definitions name="MapMarkersApi" targetNamespace="http://stknightmare.homelinux.org:30000/gipi/">
-<wsdl:types>
  -<xsd:schema targetNamespace="http://stknightmare.homelinux.org:30000/gipi/">
    -<xsd:complexType name="sfXMapMarker">
      -<xsd:sequence>
          <xsd:element name="id" type="xsd:int"/>
          <xsd:element name="difficulty_level" type="xsd:int"/>
          <xsd:element name="registered_at" type="xsd:string"/>
          <xsd:element name="registration_time_zone" type="xsd:int"/>
          <xsd:element name="user_id" type="xsd:int"/>
          <xsd:element name="invalid" type="xsd:int"/>
          <xsd:element name="created_at" type="xsd:string"/>
          <xsd:element name="updated_at" type="xsd:string"/>
          <xsd:element name="latitude" type="xsd:double"/>
          <xsd:element name="longitude" type="xsd:double"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="sfXMapMarkerElement" type="tns:sfXMapMarker"/>
    -<xsd:complexType name="SfXMapMarkerArray">
      -<xsd:sequence>
          <xsd:element name="item" type="tns:sfXMapMarker" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="SfXMapMarkerArrayElement" type="tns:SfXMapMarkerArray"/>
    -<xsd:complexType name="IntArray">
      -<xsd:sequence>
          <xsd:element name="item" type="xsd:int" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="IntArrayElement" type="tns:IntArray"/>
    </xsd:schema>
  </wsdl:types>
-<wsdl:portType name="MapMarkersApiPortType">
  -<wsdl:operation name="MapMarkers" parameterOrder="user_id difficulty_level registration_time_zone user
      <wsdl:input message="tns:MapMarkersRequest"/>
      <wsdl:output message="tns:MapMarkersResponse"/>
    </wsdl:operation>
```

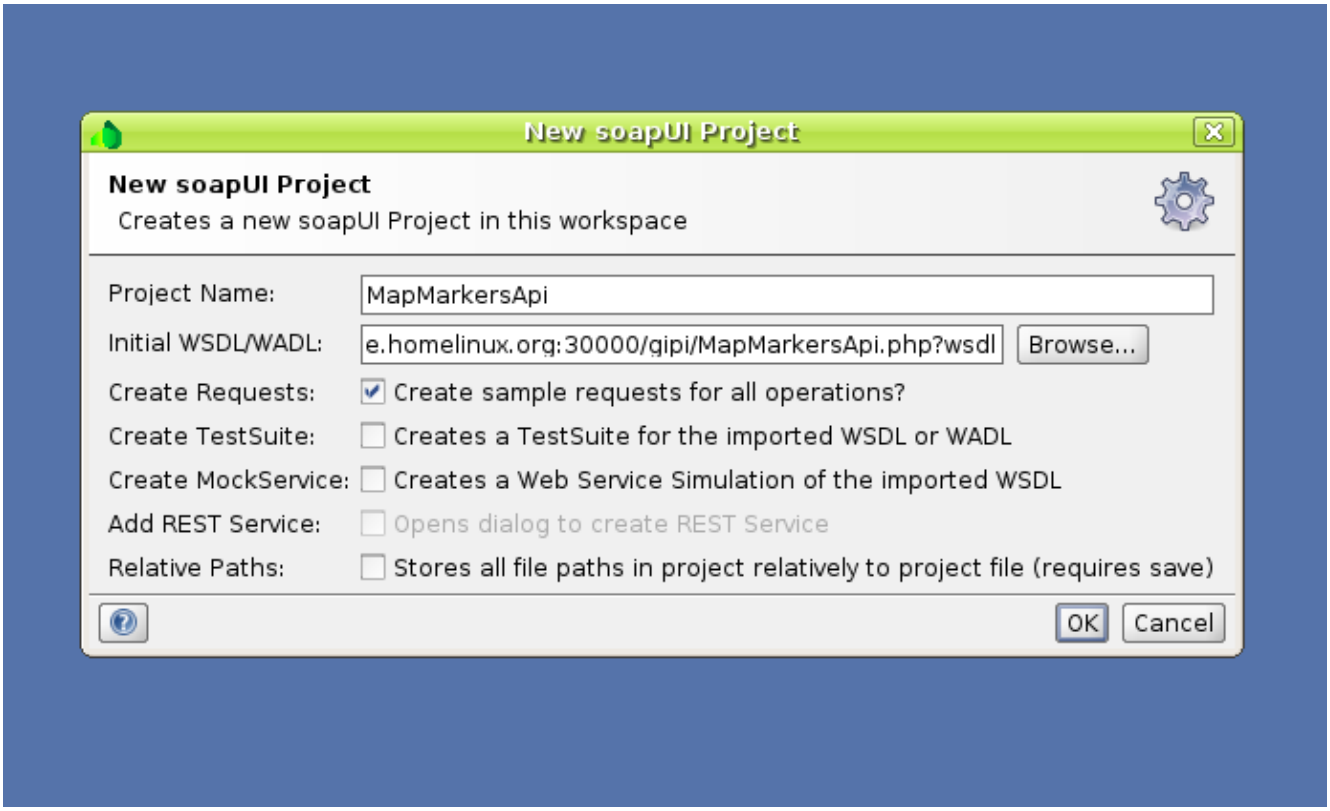
Εικόνα 37: Ένα μέρος του Wsdl του web service μας

Βήμα 2ο

Ανοίγουμε το soapui και πατάμε File->new soapUI Project (Ctrl+N).

Στο πεδίο initial WSDL/WADL

Εισάγουμε την διεύθυνση http://stknightmare.homelinux.org:30000/gipi/MapMarkersApi.php?wsdl



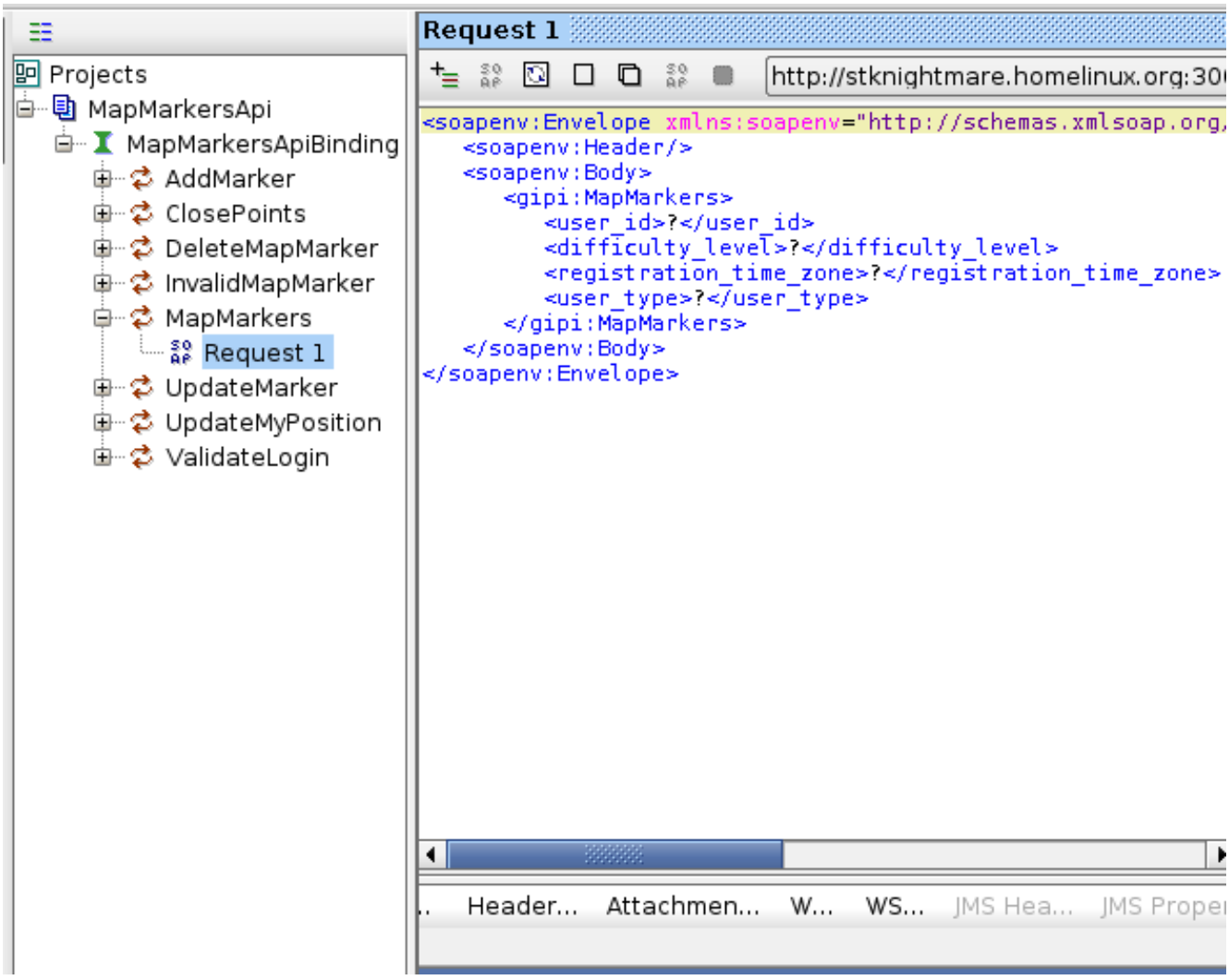
Εικόνα 38: Η οθόνη του soapui για το νέο project

Εφόσον φορτωθεί σωστά, μπορούμε να επιλέξουμε το MapMarkersApi και να το σώσουμε, με δεξί κλικ πάνω στο MapMarkersApi επιλέγουμε Save Project (Ctrl + S).

Από κάτω εμφανίεται το MapMarkersApiBinding το οποίο διαθέτει τις εξής λειτουργίες:

- AddMarker
- ClosePoints
- DeleteMapMarker
- InvalidMapMarker
- MapMarkers
- UpdateMarker
- UpdateMyPosition
- ValidateLogin
- FindDistance

Θα δοκιμάσουμε ορισμένες από τις λειτουργίες αυτές για να δούμε την απάντηση που μας δίνει ο web service server



Εικόνα 39: Οι λειτουργίες του web service μας

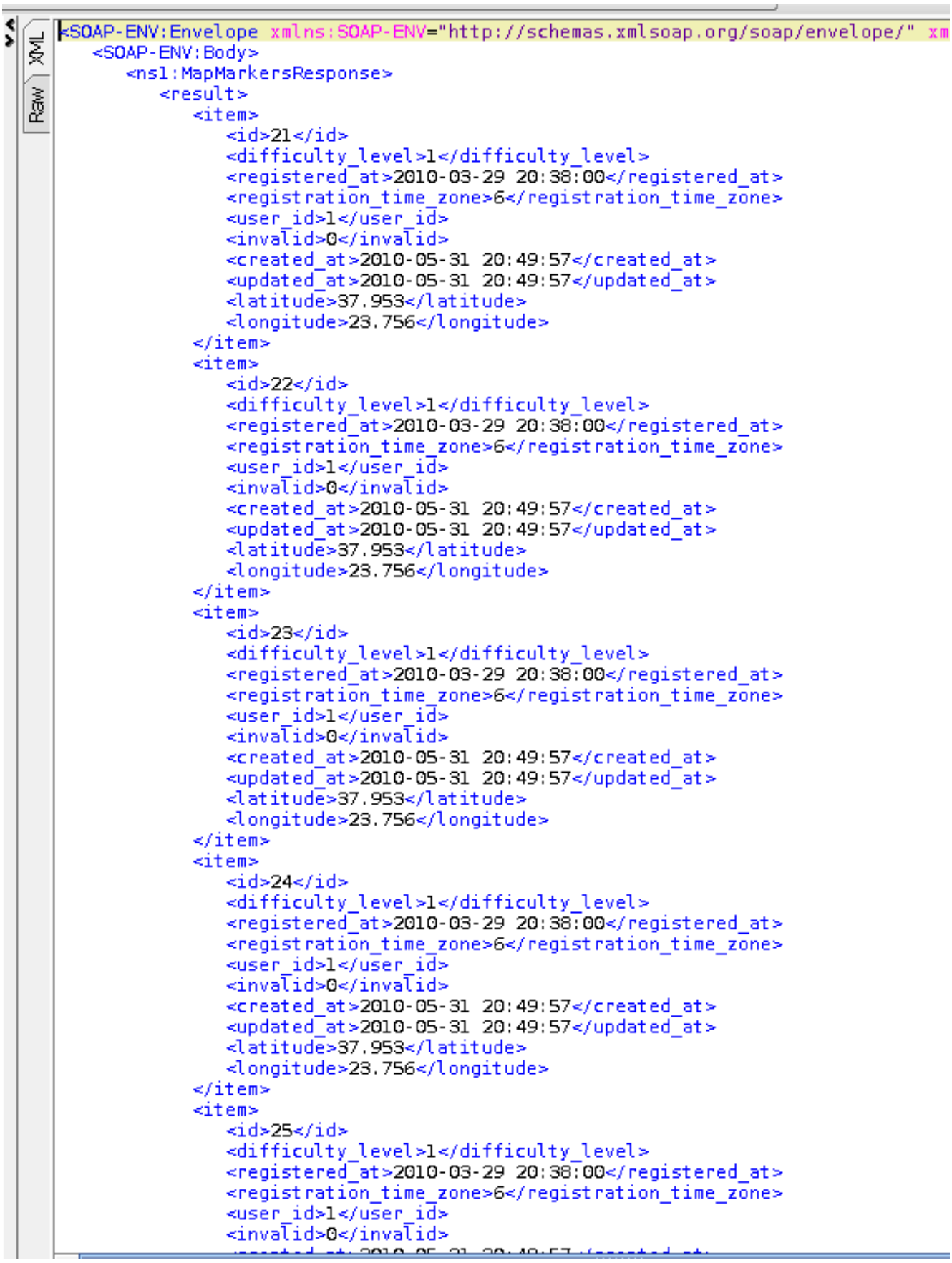
Βήμα 3ο

Κάνουμε κλικ στο + δίπλα από το MapMarkersApi, ακριβώς το ίδιο για τα MapMarkersApiBinding,MapMarkers και τέλος κάνουμε διπλό κλικ πάνω στο Request 1 του MapMarkers. (όπως φαίνεται και στην εικόνα 39)

Η εφαρμογή μας επιστρέφει ένα xml της μορφής που φαίνεται στην εικόνα 39 στο οποίο εισάγουμε τις εξής παραμέτρους

- user_id = 1
- difficulty_level = -1
- registration_time_zone = -1
- user_type = 0

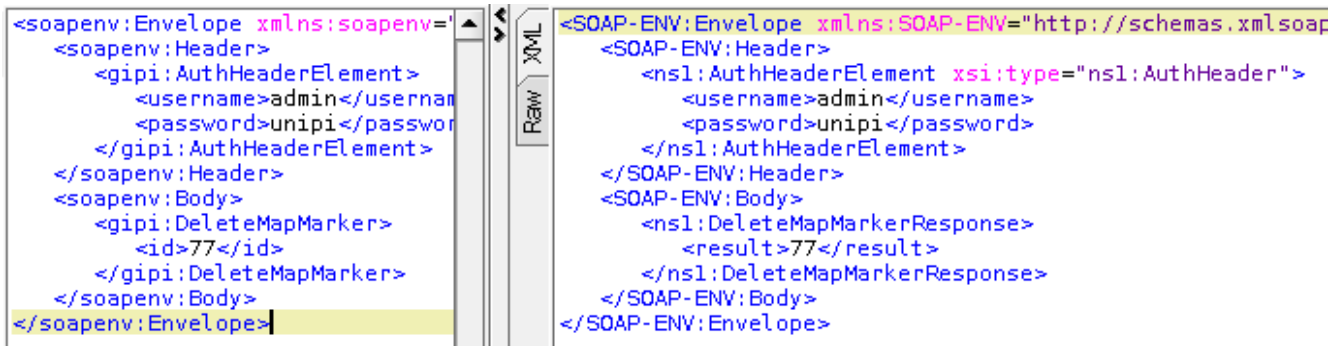
Πατώντας το πράσινο κουμπί, εκτελείται αίτηση στην function του web service να μας επιστρέψει όλες τις θέσεις με οποιοδήποτε βαθμό δυσκολίας παρκαρίσματος, που ανήκουν σε όλες τις χρονικές ζώνες και τέλος έχουν σαν ιδιοκτήτη τον user που ζητήσαμε (1). Το αποτέλεσμα είναι το παρακάτω.



Εικόνα 40: Η απάντηση της λειτουργίας του wsdl μας MapMarkers

Κάνουμε κλικ στο + δίπλα από το MapMarkersApi, ακριβώς το ίδιο για τα MapMarkersApiBinding,DeleteMapMarker και τέλος κάνουμε διπλό κλικ πάνω στο Request 1 του DeleteMapMarker.

Εισάγουμε στο username και στο password τα στοιχεία αυθεντικοποίησης μας και στο id πχ το id 77. Το web service μας επιστρέφει το παρακάτω αποτέλεσμα εφόσον είμαστε έγκυροι χρήστες. Αυτό σημαίνει ότι ο mapmarker με το id 77 βρέθηκε και διαγράφτηκε από την βάση.



Εικόνα 41: Η απάντηση της λειτουργίας DeleteMapMarker, καθώς έχουμε εισάγει έγκυρο id

Παρακάτω δείχνουμε και ένα παράδειγμα στο οποίο ο marker που έχουμε εισάγει δεν υπάρχει. Βάζουμε για id το 578 καθώς και στοιχεία αυθεντικοποίησης και το web service μας γυρνάει ως αποτέλεσμα το μηδέν που σημαίνει ότι δεν βρέθηκε ο συγκεκριμένος marker στο σύστημα και δεν μπορεί να διαγραφεί.

Εικόνα 42: Η απάντηση της λειτουργίας DeleteMapMarker, καθώς έχουμε εισάγει μη έγκυρο id

Κάνουμε κλικ στο + δίπλα από το MapMarkersApi, ακριβώς το ίδιο για τα MapMarkersApiBinding, AddMarker και τέλος κάνουμε διπλό κλικ πάνω στο Request 1 του AddMarker.

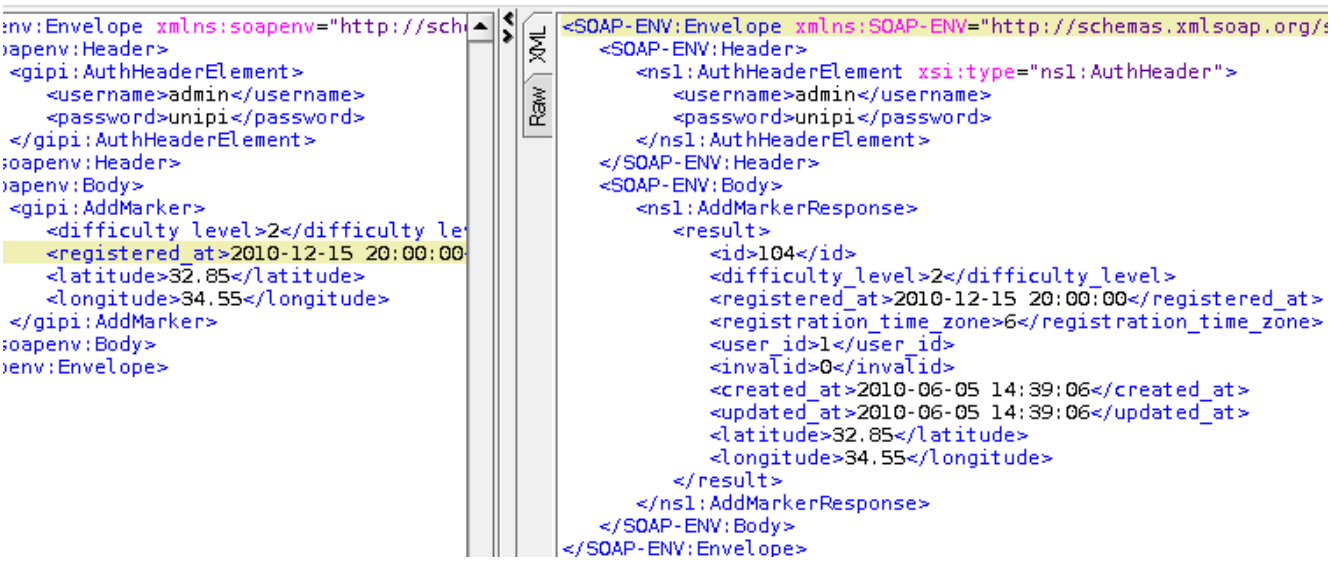
Header

- username = admin
- password = unipi

Body

- difficulty_level = 2
- registered_at = 2010-12-15 20:00:00
- latitude = 32.85
- longitude = 34.55

Παίρνουμε την εξής επιστροφή, που μας δείχνει ότι ο marker προστέθηκε κανονικά με id ίσο με 104 και μπήκε και στην ζώνη 6 στην οποία και αντιστοιχεί κανονικά.

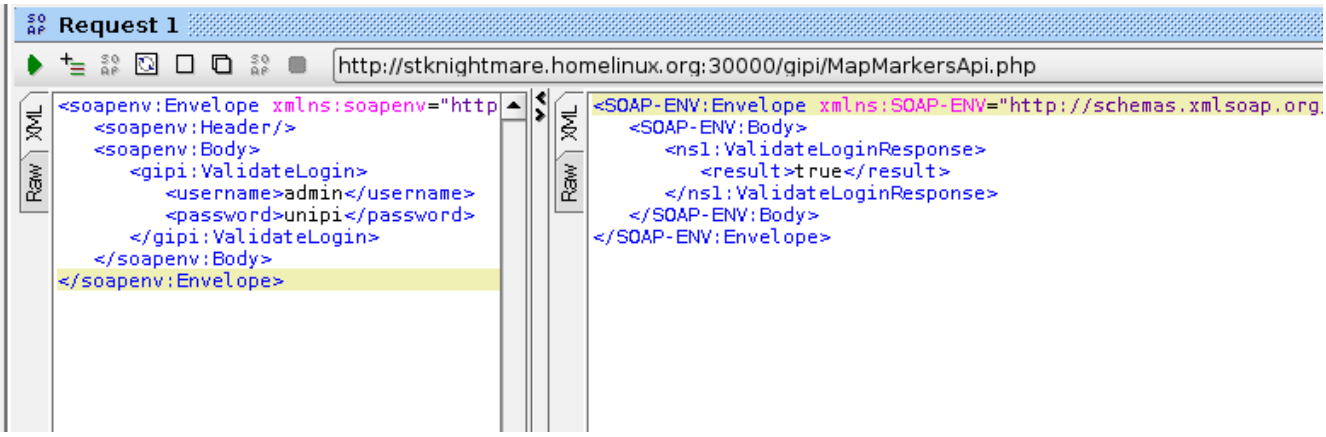


Εικόνα 43: Η απάντηση της λειτουργίας AddMarker , καθώς έχουμε εισάγει έγκυρα στοιχεία αυθεντικοποίησης

Κάνουμε κλικ στο + δίπλα από το MapMarkersApi, ακριβώς το ίδιο για τα MapMarkersApiBinding,ValidateLogin και τέλος κάνουμε διπλό κλικ πάνω στο Request 1 του ValidateLogin.

- username = admin
- password = unipi

Παίρνουμε την εξής επιστροφή, που μας δείχνει ότι ο χρήστης είναι ένας έγκυρος χρήστης για το σύστημα και το password που εισήγαγε είναι το σωστό.



Εικόνα 44: Η απάντηση της λειτουργίας του wsdl μας ValidateLogin, καθώς έχουμε εισάγει έγκυρα στοιχεία αυθεντικοποίησης

7. Πλεονεκτήματα, μειονεκτήματα, Προτάσεις για μελλοντική

επέκταση του συστήματος

Η εφαρμογή μας έχει αρκετά πλεονεκτήματα αλλά και κάποια μειονεκτήματα τα οποία και θα αναλυθούν παρακάτω. Επίσης θα παρουσιάσουμε προτάσεις για μελλοντική επέκταση του συστήματος.

7.1 Πλεονεκτήματα προγραμματιστικής υλοποίησης

Τα πλεονεκτήματα της συγκεκριμένης προγραμματιστικής υλοποίησης είναι τα εξής:

- **Αποτελεί μια λύση ανοιχτού κώδικα**, αυτό σημαίνει ότι ο κώδικας είναι προσβάσιμος και τροποποιήσιμος κάτω από την συγκεκριμένη άδεια από οποιονδήποτε το θελήσει
- **Δομημένος αντικειμενοστραφής προγραμματισμός**, όλος ο κώδικας είναι χωρισμένος σε κλάσεις για να είναι πιο ευκολοχρησιμοποίητος και όσο το δυνατόν γίνετε πιο επεκτάσιμος
- **Αποθήκευση των δεδομένων σε βάση,** τα δεδομένα αποθηκεύονται σε rdbms βάση. Χρησιμοποιείται η postgresql στην οποία εκτελούνται πολλά queries
- **Συνδυασμός διαφορετικών τεχνολογιών**, συνδυάζει αρκετές και διάφορες μεταξύ τους τεχνολογίες και γλώσσες. Ξεκινάμε από php (framework symfony) στην οποία τρέχει η εφαρμογή του server και η οποία χρησιμοποιεί τεχνολογίες όπως json, ajax, javascript (framework jquery), wsdl (web services) καθώς οι σημαντικές λειτουργίες της εφαρμογής περνάνε από τον server web service για να είναι εύκολα προσβάσιμες από παντού. Τέλος έχουμε το service του android το οποίο είναι γραμμένο σε java καθώς και το service που τρέχει στον υπολογιστή το οποίο είναι γραμμένο σε python
- **Οικονομική Λύση**, η συγκεκριμένη λύση έχει αρκετά οικονομικό κόστος υλοποίησης. Στην περίπτωση χρήσης pc (κατά προτίμηση netbook λόγω μεγέθους) μπορεί να χρησιμοποιηθεί οποιαδήποτε συσκευή που μπορεί να εκτελέσει λογισμικό linux και διαθέτει θύρα usb ή bluetooth (ένα netbook κοστίζει το λιγότερο 200 ευρώ), επίσης θα χρειαστούμε και ένα gps receiver (κόστος περίπου 30-50 ευρώ). Η χρήση συσκευής που διαθέτει android λογισμικό είναι επίσης μια ενδιαφέρουσα λύση (κόστος περίπου 220 με 250 ευρώ), ίσως όχι τόσο βολική λόγω μεγέθους της συσκευής
- **Παγκόσμιος Χάρτης,** η εφαρμογή μας χρησιμοποιεί έναν παγκόσμιο χάρτη (google maps) αυτό σημαίνει ότι μπορεί να χρησιμοποιηθεί παντού αρκεί να υπάρχει ο κατάλληλος εξοπλισμός τον οποίο και περιγράψαμε

7.2. Μειονεκτήματα προγραμματιστικής υλοποίησης

Τα μειονεκτήματα της συγκεκριμένης προγραμματιστικής υλοποίησης είναι τα εξής:

- **Ανθρώπινα λάθη,** βασιζόμαστε στα δεδομένα που εισάγονται από τον άνθρωπο. Αυτό σημαίνει ότι μπορούμε να ελέγξουμε τα δεδομένα για λάθη που έχουν να κάνουν με το σύστημα, αλλά όχι για λογικά λάθη που μπορεί να προκύψουν. Ο καθένας μπορεί να δηλώσει ένα στίγμα σε οποιοδήποτε σημείο επιθυμεί πάνω στον χάρτη και αυτό στην παρούσα φάση δεν μπορεί να ελεγχθεί
- **Google maps api,** το google maps api αποτελεί ένα πολύ δυνατό api αλλά η χρήση κάθε api έχει πλεονεκτήματα και μειονεκτήματα, των οποίων όμως η ανάλυση ξεφεύγει από το φάσμα της εφαρμογής μας. Βασικό μειονέκτημα θα ήταν το google maps api να σταματήσει να παρέχει τις υπηρεσίες του. Βέβαια ακόμα και αν για κάποιο λόγο γίνει αυτό, υπάρχουν άλλες λύσεις που μπορούμε να χρησιμοποιήσουμε, η εφαρμογή μας θα χρειαστεί τροποποιήσεις για να δουλέψει με το νέο api
- **Real time feedback,** η εφαρμογή μας όπως είπαμε βασίζεται στον ανθρώπινο παράγοντα. Αυτό σημαίνει ότι πρέπει να υπάρχουν αρκετοί χρήστες οι οποίοι να εισάγουν θέσεις στις διάφορες τοποθεσίες για να μπορεί η εφαρμογή μας να δίνει ακριβής πληροφορίες
- **Συνεχής επικοινωνία με το δίκτυο,** χρειάζεται να υπάρχει συνεχής σύνδεση με το δίκτυο στο οποίο βρίσκεται ο κεντρικός server.
- **Επιτηδευμένα λάθη,** ένα αρκετά τραβηγμένο αλλά όχι απίθανο σενάριο (από την στιγμή που δεν χρησιμοποιούνται αισθητήρες) θα ήταν ένας χρήστης να σημειώνει ανακριβείς θέσεις για να κατευθύνει αυτούς που χρησιμοποιούν την εφαρμογή σε άλλο κομμάτι και μακριά από τις διαθέσιμες θέσεις. Αυτό δεν είναι παράλογο αν αναλογιστούμε ότι ανά σημείο οι αντίστοιχες ελεύθερες θέσεις είναι από 1 – 3.

7.3 Προτάσεις για μελλοντική επέκταση του συστήματος

Με την ανάγνωση αναφορών για άλλα συστήματα οδηγηθήκαμε σε μελλοντικά σχέδια για ένα πιο ολοκληρωμένο σύστημα, τα οποία και παρουσιάζουμε παρακάτω.

Ένα από τα βήματα που χρειάζεται να γίνει είναι ο διαχωρισμός όλων των δρόμων σε ζώνες, χωρισμένες ανά μήκος αυτοκινήτων. Αυτό σημαίνει ότι θα πρέπει να γίνουν διαγραμμίσεις σε όλους τους δρόμους από την αρχή του κάθε δρόμου μέχρι το τέλος του (ένας συγκεκριμένος δρόμος έχει χωρητικότητα 20 αυτοκινήτων μήκους 3.5 μέτρων). Θα μπορούσαμε να χωρίσουμε τους δρόμους σε διαφορετικές θέσεις μήκους για να είναι χρήσιμες από όλους τους οδηγούς χωρίς να χρειάζεται για παράδειγμα ένας οδηγός να πιάσει δύο θέσεις για να παρκάρει. Το πλάτος των θέσεων θα μπορούσαμε να το βρούμε διενεργώντας έναν στατιστικό έλεγχο σε σχέση με τις αντιπροσωπίες καθώς και τα στοιχεία που διαθέτει το υπουργείο για το κάθε αυτοκίνητο. Για παράδειγμα αν ξέρουμε ότι σε έναν δρόμο υπάρχουν μόνο αυτοκίνητα μήκους 4.5 μέτρων θα χωρίζαμε τον δρόμο αντίστοιχα.

Αφού χωριστεί ο δρόμος σε θέσεις, χρειάζεται ένας τρόπος για να αναγνωρίζεται αν μια θέση καταλαμβάνεται από όχημα ή όχι. Για να το επιτύχουμε αυτό χρειάζεται να χρησιμοποιήσουμε αισθητήρες (η συγκεκριμένη τεχνολογία είναι αρκετά διαδεδομένη). Θα μπορούσαμε να χρησιμοποιήσουμε αισθητήριες ζώνες όπως είδαμε (ειδικές ζώνες που ενεργοποιούνται όταν περάσει ένα αντικείμενο από πάνω τους) σε συνδυασμό με αισθητήρες οι οποίοι αναγνωρίζουν αν μια θέση καταλαμβάνεται η όχι από ένα όχημα. Για παράδειγμα, θα μπορούσε κάποιος να πατήσει σε μια ζώνη

άλλης θέσης κάνοντας ελιγμούς έτσι ώστε να μπει σε μια διαφορετική θέση.

Κάθε θέση θα πρέπει να διαθέτει έναν ειδικό μικροϋπολογιστή ο οποίος θα ενώνεται με τον αισθητήρα της θέσης. Αυτός ο υπολογιστής μπορεί να εκτελεί διάφορους υπολογισμούς και να στέλνει μηνύματα μέσω του ασύρματου δικτύου στους υπόλοιπους αισθητήρες. Στην παρούσα υλοποίηση θέλουμε να δημιουργήσουμε ένα ad hoc δίκτυο το οποίο δεν βασίζεται σε κάποιον κεντρικό υπολογιστή για να τρέξει καθώς το σύστημα μπορεί να απαντήσει από μόνο του και με βάση τα μηνύματα που έχει λάβει από το υπόλοιπο δίκτυο στα ερωτήματα τα οποία υποβάλλονται σε αυτό.

Το δίκτυο αυτό πρέπει να χωριστεί σε διαφορετικές ομάδες και κάθε ομάδα να έχει μια κεφαλή, όπως είδαμε στην συγκεκριμένη αρχιτεκτονική πιο πάνω (δίκτυο χωρισμένο σε clusters και ανά cluster πρέπει να υπάρχει μια κεφαλή). Κάθε μικροϋπολογιστής στέλνει τα μηνύματα που λαμβάνει στην κεφαλή του είτε προωθεί σε αυτήν την κατάσταση του. Κάθε κεφαλή μπορεί να ενημερωθεί για την κατάσταση του δικτύου ρωτώντας τις γειτονικές κεφαλές που με την σειρά τους μπορούν να ρωτήσουν τις αντίστοιχες γειτονικές κεφαλές. Το δίκτυο δεν είναι διαρκώς μεταβαλλόμενο αλλά πρόκειται για ένα σταθερό δίκτυο το οποίο μπορεί να αλλάξει με την προσθήκη μιας κεφαλής, είτε με την βλάβη μιας από αυτές. Θα μπορούσαμε όμως να χρησιμοποιήσουμε και έναν κεντρικό υπολογιστή τον οποίο κάθε κεφαλή ενημερώνει ανά τακτά χρονικά διαστήματα για την κατάσταση των αισθητήρων της. Ο χρήστης πρέπει να μπορεί να μπει στο interface και να δει την κατάσταση του δικτύου. Παρ' όλα αυτά το δίκτυο ενημερώνει από μόνο του τον κεντρικό υπολογιστή και δεν χρειάζεται να γίνονται ερωτήσεις σε αυτό (σπατάλη περιττών πόρων), επίσης όπως είπαμε το δίκτυο δεν θα χρειάζεται τον κεντρικό υπολογιστή για να λειτουργήσει αλλά θα είναι αυτόνομο με κατάλληλο πρόγραμμα ελέγχου.

Εφόσον το δίκτυο με τους αισθητήρες μας είναι έτοιμο, θα χρειαστούμε εφαρμογή η οποία θα μπορεί να προσφέρει πολλές και διάφορες λειτουργίες στον χρήστη. Βασικό πλεονέκτημα είναι η κράτηση μιας θέσης (ισχύει για τις θέσεις επί πληρωμή καθώς δεν είναι όλες οι θέσεις του συστήματος μας επί πληρωμή). Ο χρήστης μπορεί να εκφράσει διάφορα ερωτήματα στο δίκτυο μέσω του κινητού του και να βρει την θέση που επιθυμεί (ερώτηση στο σύστημα θα μπορούσε να είναι, εντοπισμός μιας θέσης που βρίσκεται σε απόσταση 100 μέτρων από την τρέχουσα θέση του χρήστη). Ο χρήστης μπορεί μέσω του κατάλληλου λογισμικού να δει ανά πάσα στιγμή τι γίνεται σε ολόκληρο το δίκτυο.

Ο χρήστης για να κάνει κράτηση μιας θέσης εισάγει την πινακίδα του καθώς και το χρονικό διάστημα που επιθυμεί να την κρατήσει. Σε περίπτωση που πρόκειται για θέση επί πληρωμή χρεώνεται ο λογαριασμός του χρήστη. Θα πρέπει να υπάρχουν ειδικά πακέτα χρεώσεων για να διευκολύνεται ο χρήστης καθώς και πρόστιμα για μη επιθυμητά παρκαρίσματα (αυτοκίνητα που καταλαμβάνουν δύο θέσεις αντί για μια), αυτοκίνητα που υπερβαίνουν το χρόνο παραμονής τους σε μια θέση, είτε για οδηγούς που παρκάρουν σε θέση επί πληρωμή την οποία δεν έχουν κατοχυρώσει. Το σύστημα πρέπει να μπορεί να ενημερώνει τον χρήστη ότι λήγει ο χρόνος στάθμευσης του και να πάει να μετακινήσει το αμάξι του αλλιώς θα χρεώνεται με κάποιο είδος πρόστιμου.

Επίσης καλό θα ήταν να υπάρχει και όριο στάθμευσης για τις θέσεις που δεν είναι επί πληρωμή. Αυτό μπορεί να γίνεται είτε αν το αυτοκίνητο δηλώνει την ώρα που θα παραμείνει σε μια θέση είτε αν υπάρχει ανώτατο όριο κατά το οποίο ένα αμάξι μπορεί να παραμείνει σε μια θέση και μετά από αυτό να πρέπει να βρει άλλη θέση.

Το σύστημα ακόμα θα πρέπει να απαντάει γρήγορα στα ερωτήματα του οδηγού, γιατί αυτός μπορεί καθώς υποβάλει τα ερωτήματα να αλλάζει συνέχεια θέση. Επίσης το σύστημα θα μπορεί να υπολογίζει

και την πιθανότητα δυο οδηγοί να ενδιαφέρονται για την ίδια θέση, οπότε θα πρέπει και να ενημερώνει τον χρήστη που βρίσκεται πιο μακριά ότι η θέση για την οποία ενδιαφερόταν δεν είναι πια διαθέσιμη καθώς και να του προτείνει κάποια καινούρια (ο χρήστης γλυτώνει τον δρόμο μέχρι την θέση την οποία θα έβρισκε δεσμευμένη).

Η συγκεκριμένη υλοποίηση είναι αρκετά ενδιαφέρουσα και χρειάζονται πολύπλοκοι αλγόριθμοι υπολογισμού οι οποίοι θα απαντούν στις διάφορα ερωτήματα που θα υποβάλει ο κάθε χρήστης. Το σύστημα μπορεί να ξεκινήσει να δουλεύει πιλοτικά για παράδειγμα στις θέσεις parking που υπάρχουν στο κέντρο της Αθήνας και κοντά στις στάσεις των μέσων (μετρό ή ηλεκτρικός για να ευνοείται η χρήση των μέσων) και μετά να αναπτυχθεί ένα ενιαίο δίκτυο το οποίο θα αναπτύσσεται σιγά σιγά. Με την ανάλυση των στατιστικών θα μπορούσαν να προκύψουν αλλαγές στις τιμές των θέσεων.

Η υλοποίηση ενός τέτοιου συστήματος είναι αρκετά ακριβή (περίπου 450 ευρώ χρειάζονται για κάθε αισθητήρα - συσκευή) αλλά θεωρούμε ότι τα πλεονεκτήματα που μπορεί να επιφέρει στους οδηγούς, οι οποίοι θα γλυτώσουν από άσκοπες περιπλανήσεις προσπαθώντας να βρουν να παρκάρουν, είναι μεγάλα. Ακόμα με την σειρά της, η μείωση χρήσης του αυτοκινήτου (η ποσότητα βενζίνης που σπαταλάει ένα αμάξι όταν ψάχνει για θέση parking για 45 λεπτά είναι αρκετά μεγάλη) θα οδηγήσει σε καλυτέρευση της ατμόσφαιρας από την μείωση των άσκοπων καύσεων. Χωρίς όμως αυτό να μπορούμε να το πούμε με σιγουριά, καθώς η σκέψη ότι δεν θα βρούμε πάρκινγκ καμία φορά μπορεί να μας οδηγήσει να μην πάρουμε το αυτοκίνητο μας. Αντίθετα, αν γνωρίζουμε ότι θα βρούμε να παρκάρουμε με την χρήση της εφαρμογής, μπορεί να πάρουμε το αυτοκίνητο και ως συνέπεια, ενώ έχουμε μείωση του χρόνου παραμονής μέσα στο αυτοκίνητο μπορεί να έχουμε αύξηση των αυτοκινήτων. Αυτά βέβαια αποτελούν θεωρητικές υποθέσεις και μόνο η χρήση μπορεί να μας δώσει μια καθαρή εικόνα.

8. Συμπεράσματα - Ανακεφαλαίωση

Η δυσκολία της υλοποίησης μας ήταν μεγάλη, καθώς η αρχιτεκτονική που χρησιμοποιήσαμε ήταν πολύπλοκη (υποστήριξη διαφορετικών ειδών συσκευών). Παρ' όλα αυτά καταφέραμε να αναλύσουμε

τα προβλήματα που προέκυψαν και να βρούμε λύσεις για αυτά. Σε αυτό βοήθησε ο σκελετός μας, το symfony, όπως και όλες οι σύγχρονες τεχνολογίες που είχαμε στην διάθεση μας.

Παίρνοντας ως βάση την php, πάνω της δημιουργήσαμε τον server του web service μας. Ο κορμός αυτός, περιλαμβάνει όλες τις κατάλληλες λειτουργικότητες, τις οποίες με την σειρά τους καλούν ο python client για τους υπολογιστές και τέλος ο client για τις android συσκευές. Η ανάγνωση αναφορών για άλλες λύσεις που υπάρχουν μας οδήγησε σε σκέψη, από την οποία προέκυψαν κάποιες τροποποιήσεις για την εφαρμογή μας, καθώς και μελλοντικές σκέψεις για να υλοποιήσουμε μια πιο ολοκληρωμένη λύση. Η πιο ολοκληρωμένη λύση θα περιλαμβάνει λογισμικό αλλά και μηχανικό κομμάτι (με αισθητήρες) για να είναι όσο το δυνατόν πιο ακριβής. Η συγκεκριμένη λύση όμως, ακόμα και στην τωρινή της μορφή, θεωρούμε ότι αν χρησιμοποιηθεί σωστά και από πολύ κόσμο μπορεί να βοηθήσει αρκετά όσον αφορά το παρκάρισμα, και έτσι να οδηγήσει και σε μείωση άλλων προβλημάτων, όπως η μείωση της κίνησης, οι εκπομπές αερίων και η κατανάλωση βενζίνης.

9. Βιβλιογραφία

Αναφέρεται παρακάτω αναλυτικά η βιβλιογραφία που χρησιμοποιήθηκε στην πτυχιακή χωρισμένη σε κατηγορίες.

9.1 Papers

1) SmartParking: A Secure and Intelligent Parking System Using NOTICE των Gongjun Yan, Stephan Olariu, Michele C. Weigle, Mahmoud Abuelela, Οκτώβριος 2008
2) Networked Parking Spaces: Architecture and Applications των Prithwish Basu and Thomas D.C. Little
3) Design and Implementation of a prototype Smart
4) PARKing (SPARK) System using Wireless Sensor Networks των S. V. Srikanth, Pramod P. J, Dileep K. P, Tapas S, Mahesh U. Patil, Sarat Chandra Babu N
5) Smart Parking: an Application of optical Wireless Sensor Network των Jatuporn Chinrungrueng, Udomporn Sunantachaikul, Satien Triamlumlerd

9.2 Sites

6) http://en.wikipedia.org/wiki/Latitude
7) http://en.wikipedia.org/wiki/Longitude
8) http://en.wikipedia.org/wiki/Haversine_formula
9) http://www.websequencediagrams.com/
10) http://en.wikipedia.org/wiki/Use_case_diagram
11) http://developer.android.com
12) http://www.symfony-project.org/jobeet/1_4/Doctrine/en/
13) http://www.symfony-project.org/reference/1_4/en/
14) http://www.symfony-project.org/more-with-symfony/1_4/en/
15) http://www.symfony-project.org/gentle-introduction/1_4/en/
16) http://www.doctrine-project.org/projects/orm/1.2/docs/manual/en
17) http://www.doctrine-project.org/projects/orm/1.2/docs/cookbook/en
18) http://gpsd.berlios.de/
19) http://www.perrygeo.net/wordpress/?p=13
20) http://en.wikipedia.org/wiki/Entity-relationship_model
21) http://en.wikipedia.org/wiki/Relational_model
22) http://en.wikipedia.org/wiki/Sequence_diagram
23) http://api.jquery.com/
24) http://code.google.com/apis/maps/documentation/javascript
25) http://college.yukondude.com/2003_09_comp210/html/note-container.php?file=02^Handout^Crow~s_Foot_Entity-Relationship_Diagram_Notation.html

26) http://www.movable-type.co.uk/scripts/latlong.html

27) http://www.navigon.com

28) http://spectrum.ieee.org/green-tech/advanced-cars/smart-parking-systems-make-it-easier-to-find-a-parking-space/1

29) http://www.rfidjournal.com/article/view/3625/2

30) http://php.net/index.php

31) http://www.python.org

32) http://www.java.com/en/