Έγγραφο απαιτήσεων λογισμικού (SRS)

ChargeNET

# Εισαγωγή

## 1.1 Εισαγωγή: σκοπός του λογισμικού

Σκοπός του λογισμικού είναι να υποστηρίξει ολόκληρη την αλυσίδα των διαδικασιών φόρτισης ηλεκτρικών οχημάτων. Το πληροφοριακό σύστημα θα δίνει τη δυνατότητα σε όλους τους εμπλεκόμενους στη διαδικασία αυτή να μοιράζονται μεταξύ τους σημαντικές πληροφορίες, να οργανώνουν καλύτερα τις μεταξύ τους σχέσεις και να έχουν ανά πάσα στιγμή μια σύνοψη των σημαντικών πληροφοριών τους. Στόχος είναι να αυξηθεί η διαφάνεια στην αγορά, να ξεπεραστούν τα εμπόδια που έχουν να κάνουν με τον αυξημένο χρονικό παράγοντα του αυξημένου ανεφοδιασμού και να υποστηριχθεί ο χρονοπρογραμματισμός και η οργάνωση των ανανεωμένων διαδικασιών ανεφοδιασμού με το ελάχιστο δυνατό κόστος προσαρμογής από όλους τους εμπλεκόμενους.

## 1.2 Διεπαφές (interfaces)

### 1.2.1 Διεπαφές με εξωτερικά συστήματα

Η εφαρμογή μας, λόγω του γεγονότος ότι έχει ως βάση ένα Restful API δίνει τη δυνατότητα σε τρίτα - εξωτερικά συστήματα να επικοινωνούν με αυτή στέλνοντας αιτήματα προς εξυπηρέτηση στα κατάλληλα endpoints.

Ακόμα, η εφαρμογή μας χρησιμοποιεί και αυτή τρίτα συστήματα APIs για να εκτελέσει κάποιες λειτουργίες της. Συγκεκριμένα, χρησιμοποιούνται τα:

* **Google Maps API**: με σκοπό την προβολή των χαρτών στο front-end της εφαρμογής μας.
* **Payments API:** το API αυτό βοηθά στην εκτέλεση της χρέωσης των χρηστών της εφαρμογής και τη διευθέτηση των πληρωμών σε ένα ασφαλές περιβάλλον. Δεν έχει επιλεγεί επακριβώς το ποιο API με αυτή τη λειτουργία θα χρησιμοποιηθεί για αυτό και χρησιμοποιούμε το γενικό αυτό όρο.

### 1.2.2 Διεπαφές με το χρήστη

Προδιαγραφή διεπαφών με το χρήστη. Μοντέλο Use Case (UML).

Η αλληλεπίδραση με το χρήστη συντελείται μέσω είτε του cli-client, είτε του front end web-app. Όλες οι μορφές διεπαφής υποστηρίζουν, με διαφορετική διαδραστικότητα η κάθεμια, όλες τις δυνατές ενέργειας κάθε ομάδας χρηστών. Τέλος, σημειώνεται πως τρίτες εφαρμογές μπορούν να χρησιμοποιήσουν απευθείας τις λειτουργίες του API μας, στέλνοντας απευθείας αιτήματα στα κατάλληλα endpoints.

Συγκεκριμένα, ανάλογα με το ρόλο τους στην αλυσίδα των διαδικασιων οι χρήστες χωρίζονται σε:

* Μη εγγεγραμμένοι χρήστες
* Εγγεγραμμένοι ιδιοκτήτες οχημάτων
* Εγγεγραμμένοι διαχειριστές (ΣΕΟ) σε αυτοκινητόδρομους
* Διαχειριστές του συστήματος

Οι **μη εγγεγραμμένοι χρήστες** έχουν δυνατότητα να ενημερωθούν για τις δυνατότητες της εφαρμογής και να εγγραφούν σε αυτή ακολουθώντας την διαδικασία που χρειάζεται για κάθε τύπο χρήστη. Ιδιαίτερα για τους μη εγγεγραμμένους ιδιοκτήτες οχημάτων, υπάρχει η δυνατότητα απλής προβολής των σημείων φόρτισης στο χάρτη.

Οι **εγγεγραμμένοι χρήστες ως ιδιοκτήτες οχημάτων** έχουν τη δυνατότητα από τις διάφορες διεπαφές :

* Να προβάλουν στο χάρτη της θέσης φόρτισης και τα διαθέσιμα σε αυτές προγράμματα
* Να προχωρήσουν σε προκράτηση θέσης φόρτισης για συγκεκριμένη ώρα.
* Να προβάλουν το ιστορικό φορτίσεών τους.
* Να προχωρήσουν σε ηλεκτρονικές πληρωμές για τι;ς φορτίσεις τους και να δέχονται τα τιμολόγιά τους
* Να διατηρούν πόντους επιβράβευσης από τις φορτίσεις τους.
* Να διαχειρίζονται και να αποθηκεύουν τα οχήματά τους

Οι **εγγεγραμμένοι χρήστες ως διαχειριστές ΣΕΟ σε αυτοκινητόδρομους** έχουν μέσω των διεπαφών της εφαρμογής τη δυνατότητα για τις παρακάτω λειτουργίες:

* Σύνδεση / Αποσύνδεση
* Διαχείριση θέσεων φόρτισης στο σταθμό που διαχειρίζονται (προβολή, μεταβολή, προσθήκη, διαγραφή)
* Διαχείριση προγραμμάτων φόρτισης στο σταθμό που διαχειρίζονται (προβολή, μεταβολή, προσθήκη, διαγραφή)
* Διαχείριση προγραμμάτων επιβράβευσης στο σταθμό που διαχειρίζονται (προβολή, μεταβολή, προσθήκη, διαγραφή)
* Διαχείριση κρατήσεων φόρτισης για το σταθμό που διαχειρίζονται.
* Προβολή real-time κατάστασης θέσεων
* Προβολή συνολικής ενέργειας που καταναλώθηκε για χρονικό διάστημα και αντίστοιχης πρόβλεψης.
* Προβολή προτιμήσεων χρηστών.
* Προβολή ιστορικού φορτίσεων.

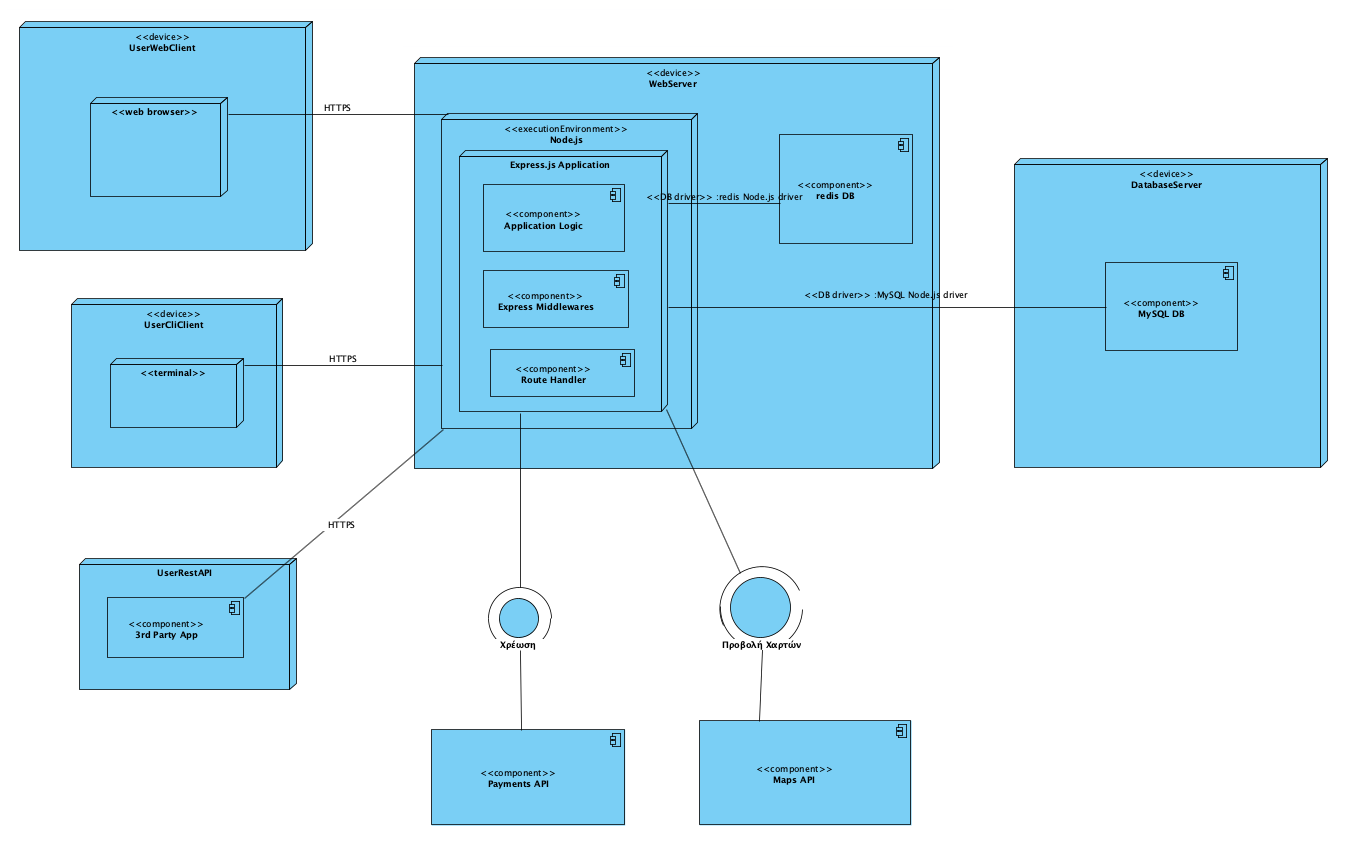
Τέλος, οι χρήστες που είναι **γενικοί διαχειριστές της εφαρμογής** έχουν το μεγαλύτερο βαθμό δικαιωμάτων και μπορούν:

* Να διαχειρίζονται όλους τους χρήστες της πλατφόρμας, δημιουργώντας νέους, αλλάζοντας τα στοιχεία των υπάρχοντων και διαγράφοντας χρήστες αν κάτι τέτοιο απαιτείται.
* Να διαχειρίζονται όλους τους πίνακες της βάσης δεδομένων, δημιουργώντας, μεταβάλλοντας και διαγράφοντας καταχωρήσεις.

Η αναλυτική παρουσίαση με Use Case UML πρόκειται να υλοποιηθεί σε επόμενη έκδοση του παρόντος.

Παρακάτω παρουσιάζεται το UML Deployment Diagram που παρουσιάζει το σύνολο της εφαρμογής και των εξωτερικών διεπαφών της, με ένα μεγάλο βαθμό αφαίρεσης.

UML Deployment Diagram



# Προδιαγραφές απαιτήσεων λογισμικού

## 2.1 Περιπτώσεις χρήσης

### 2.1.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: Προγραμματισμός φόρτισης από εγγεγραμμένο ιδιοκτήτη οχήματος.

#### 2.1.1.1 Χρήστες (ρόλοι) που εμπλέκονται

* Εγγεγραμμένος χρήστης - Ιδιοκτήτης Οχήματος
* Front - End Web app
* Back - end
* MySQL Database
* redis Database

#### 2.1.1.2 Προϋποθέσεις εκτέλεσης

Για να μπορεί να εκτελεστεί η παραπάνω ενέργεια θα πρέπει αρχικά ο χρήστης που θέλει να κάνει την κράτηση για φόρτιση να είναι εγγεγραμμένος στο σύστημά μας ως ιδιοκτήτης ηλεκτρικού οχήματος και να έχει ταυτοποιηθεί από το σύστημα, να είναι δηλαδή logged-in.

#### 2.1.1.3 Περιβάλλον εκτέλεσης

Η εκτέλεση του συγκεκριμένου σεναρίου χρήσης ξεκινάει από το **front end web-app client**, ο οποίος εξυπηρετεί το χρήστη- ιδιοκτήτη αυτοκινήτου. Το αίτημα του χρήστη αρχικά για προβολή σταθμών στο χάρτη,, αλλά και τα επιμέρους αιτήματα, όταν αυτός ζητά να προβάλει επιπλέον πληροφορίες για έναν σταθμό, όπως διαθέσιμα προγράμματα φόρτισης, διαβιβάζεται από το front-end client στο **back-end**, το οποίο αναλαμβάνει την εξυπηρέτησή του. Εμπλέκεται επίσης και η **MySQL Database**, από την οποία λαμβάνονται τα δεδομένα. Τέλος πριν ο χρήστης καταφέρει να προσπελάσει δεδομένα τα οποία βρίσκονται στη MySQL Database, γίνεται επαλήθευση πως ο χρήστης είναι ταυτοποιημένος μέσω της **redis Database**.

#### 2.1.1.4 Δεδομένα εισόδου

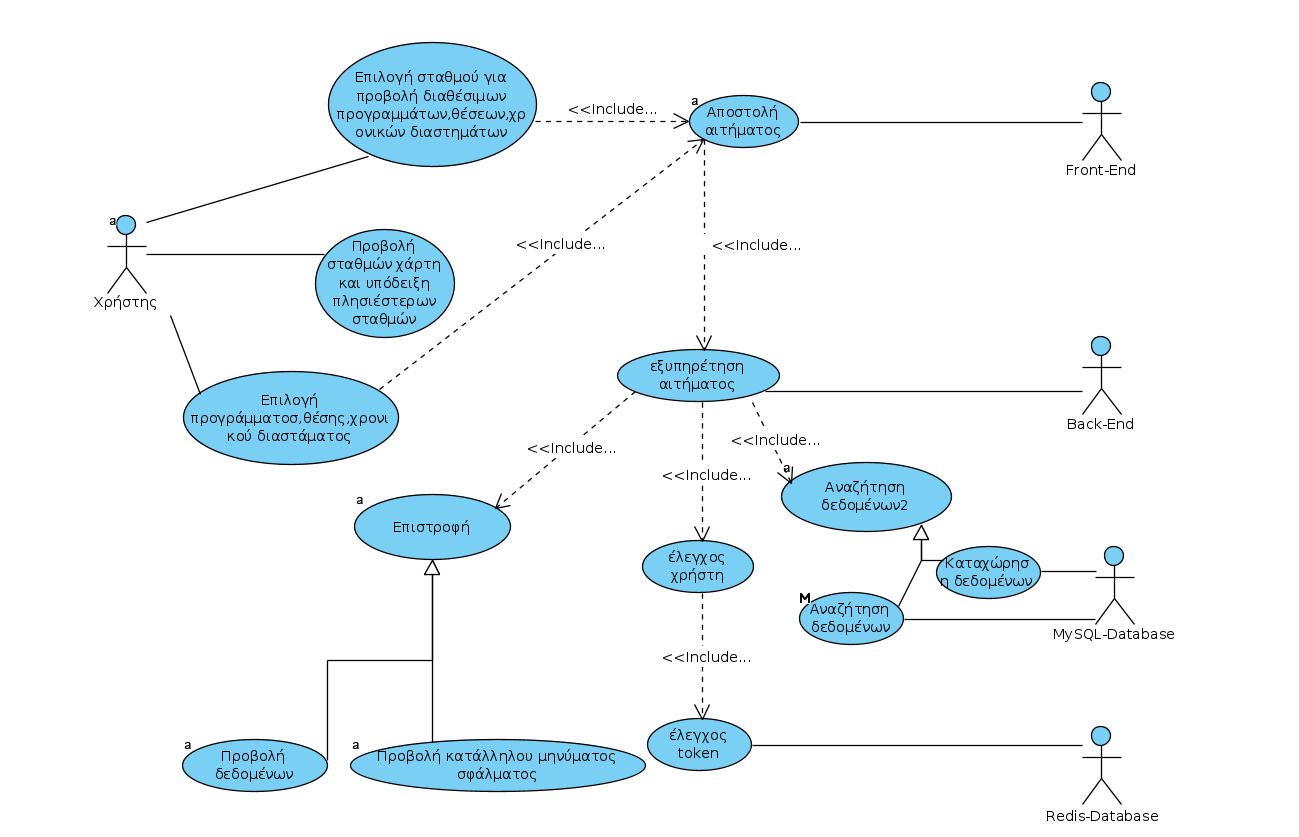
Δε λαμβάνουμε δεδομένα εισόδου, καθώς όλες οι απαραίτητες παράμετροι περνούν μέσω του URL.

#### 2.1.1.5 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

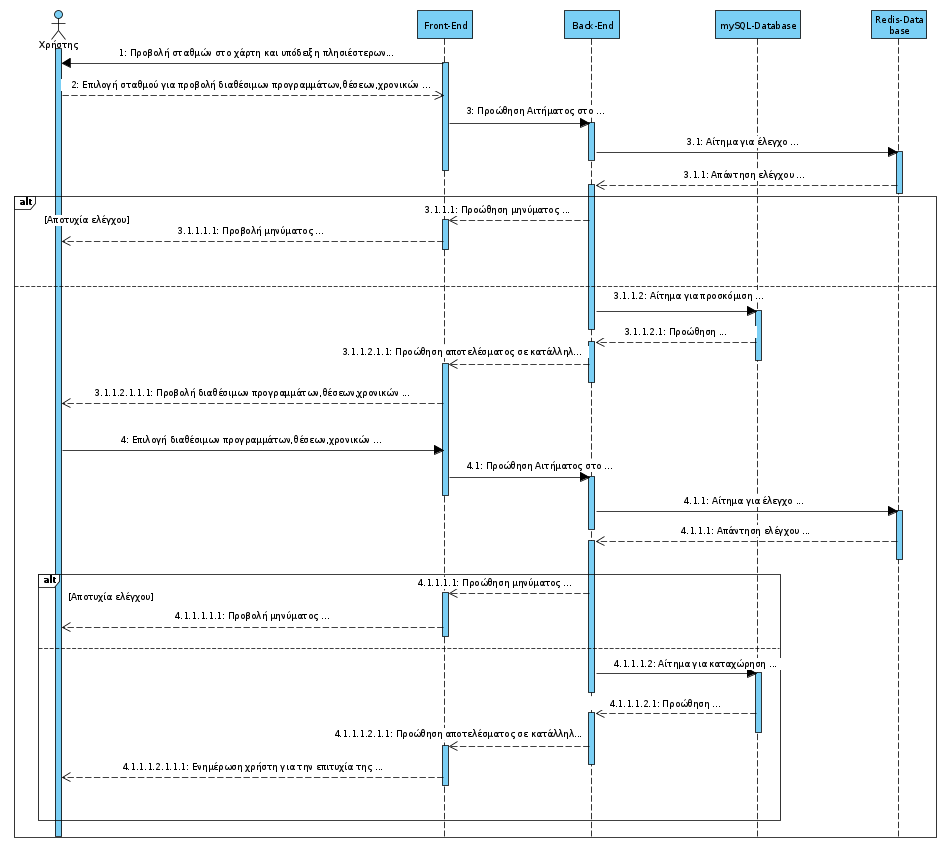
Η αλληλουχία ενεργειών φαίνεται στα παρακάτω βήματα:

1. Προβολή σταθμών στο χάρτη και υπόδειξη πλησιέστερων σταθμών σε λίστα.
2. Επιλογή σταθμού είτε από το χάρτη, είτε από λίστα, για προβολή διαθέσιμων προγραμμάτων,θέσεων και χρονικών διαστημάτων..
3. Προώθηση αιτήματος για προβολή δεδομένων σταθμού από το front-end στο back-end για εξυπηρέτηση.
4. Αίτημα από το back-end στη redis Database για επιβεβαίωση πως ο χρήστης είναι συνδεδεμένος.
5. Αν όχι, αποστολή κατάλληλου μηνύματος αποτυχίας.
6. Αν ναι, αίτημα από το back-end στη βάση δεδομένων για προσκόμιση των δεδομένων που θέλει να προβάλει ο χρήστης.
7. Προώθηση αποτελέσματος αναζήτησης από τη βάση στο back-end, επεξεργασία τους ώστε να έρθουν στην κατάλληλη μορφή και προώθησή τους στο front-end.
8. Προβολή του αποτελέσματος στο χρήστη από το front-end με κατάλληλο τρόπο.
9. Επιλογή προγράμματος, θέσης και ώρας από το χρήστη μέσω του front-end.
10. Προώθηση αιτήματος για καταχώρηση δεδομένων από το front-end στο back-end για εξυπηρέτηση.
11. Αίτημα από το back-end στη redis Database για επιβεβαίωση πως ο χρήστης είναι συνδεδεμένος.
12. Αν όχι, αποστολή κατάλληλου μηνύματος αποτυχίας.
13. Αν ναι, αίτημα από το back-end στη βάση δεδομένων για καταχώρηση των δεδομένων που έχει επιλέξει ο χρήστης.
14. Προώθηση αποτελέσματος καταχώρησης από τη βάση στο back-end, επεξεργασία τους ώστε να έρθουν στην κατάλληλη μορφή και προώθησή τους στο front-end.
15. Ενημέρωση χρήστη για την επιτυχία της κράτησης του σταθμού του για το επιλεγμένο χρονικό διάστημα με σχετικό μήνυμα.

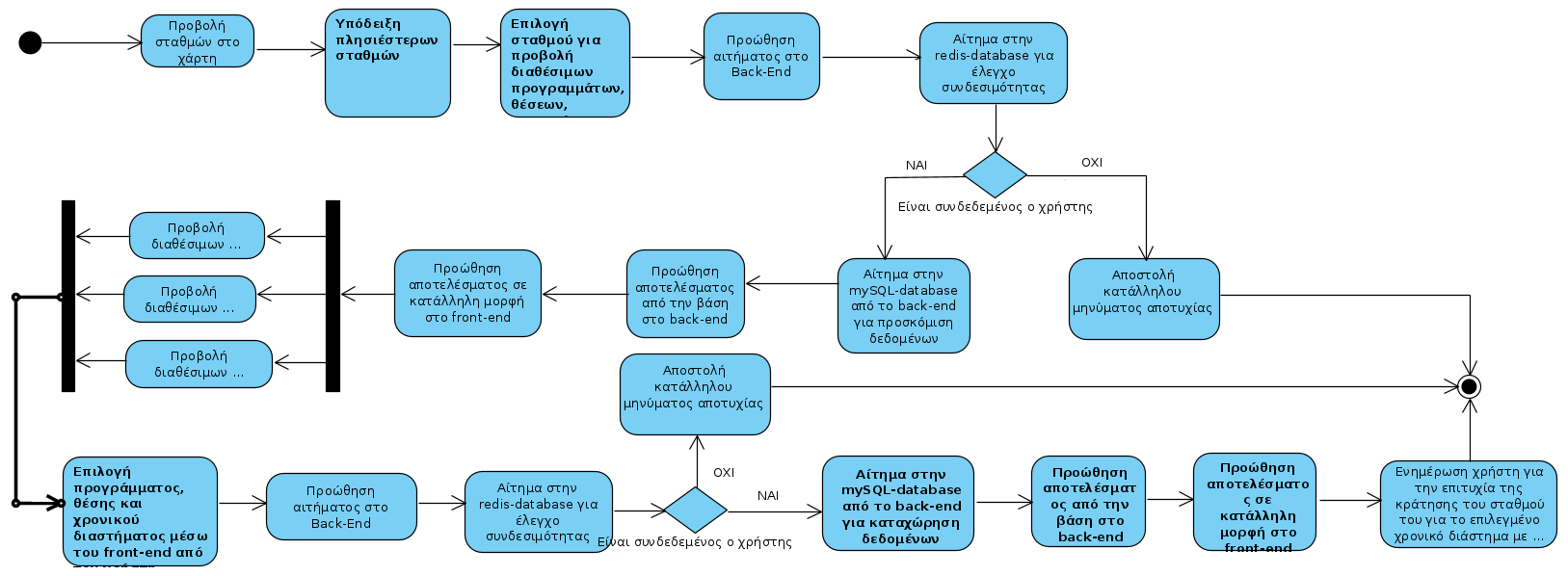
##### Use Case Diagram



##### Sequence Diagram

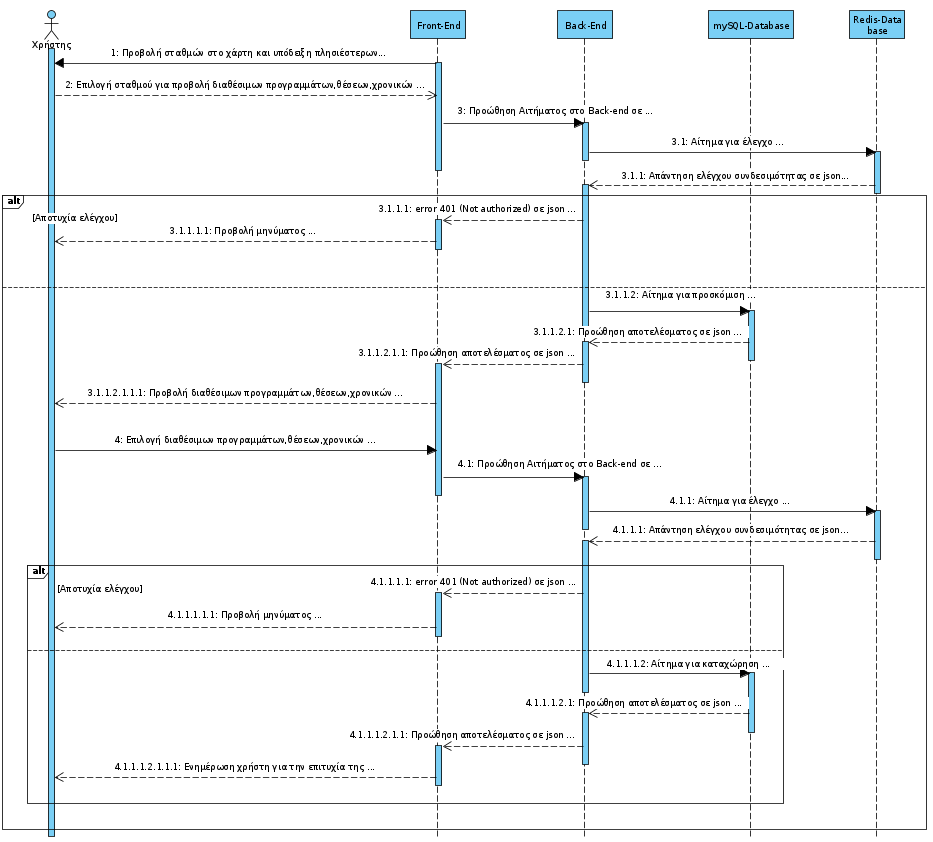


##### Activity Diagram



#### 3.1.1.7 Δεδομένα εξόδου

##### Sequence Diagram with exit



### 2.1.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: Επιλογή προγράμματος φόρτισης από ιδιοκτήτη οχήματος.

#### 2.1.2.1 Χρήστες (ρόλοι) που εμπλέκονται

* Εγγεγραμμένος χρήστης - Ιδιοκτήτης Οχήματος
* Front - End Web app
* Back - end
* MySQL Database
* redis Database
* API πληρωμών

#### 2.1.2.2 Προϋποθέσεις εκτέλεσης

Για να μπορεί να εκτελεστεί η παραπάνω ενέργεια θα πρέπει αρχικά ο χρήστης που θέλει να κάνει την κράτηση για φόρτιση να είναι εγγεγραμμένος στο σύστημά μας ως ιδιοκτήτης ηλεκτρικού οχήματος και να έχει ταυτοποιηθεί από το σύστημα, να είναι δηλαδή logged-in.

#### 2.1.2.3 Περιβάλλον εκτέλεσης

Η εκτέλεση του συγκεκριμένου σεναρίου χρήσης ξεκινάει από το **front end web-app client**, ο οποίος εξυπηρετεί το χρήστη- ιδιοκτήτη αυτοκινήτου. Το αίτημα του χρήστη αρχικά για προβολή διαθέσιμων προγραμμάτων φόρτισης, διαβιβάζεται από το front-end client στο **back-end**, το οποίο αναλαμβάνει την εξυπηρέτησή του. Εμπλέκεται επίσης και η **MySQL Database**, από την οποία λαμβάνονται τα δεδομένα. Τέλος πριν ο χρήστης καταφέρει να προσπελάσει δεδομένα τα οποία βρίσκονται στη MySQL Database, γίνεται επαλήθευση πως ο χρήστης είναι ταυτοποιημένος μέσω της **redis Database**.

#### 2.1.2.4 Δεδομένα εισόδου

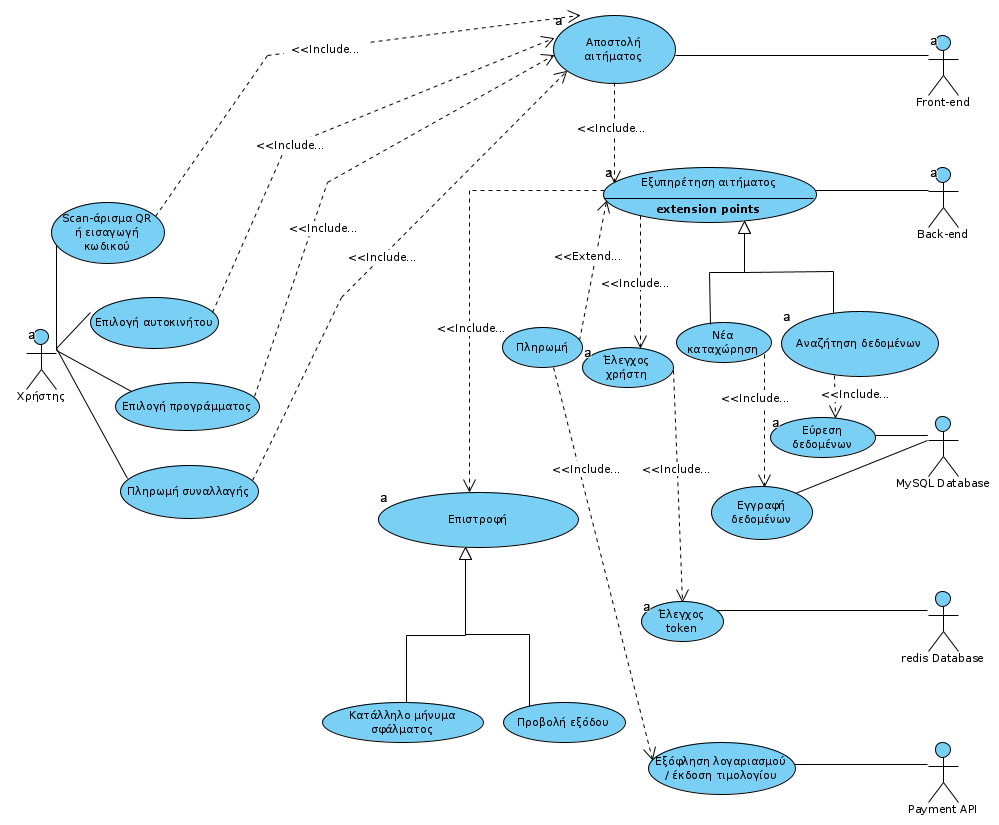
Δε λαμβάνουμε δεδομένα εισόδου, καθώς όλες οι απαραίτητες παράμετροι περνούν μέσω του URL.

#### 2.1.2.5 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Η αλληλουχία ενεργειών φαίνεται στα παρακάτω βήματα:

1. Scan-άρισμα QR code που βρίσκεται στο σταθμό από το χρήστη ή χειροκίνητη εισαγωγή κωδικού στο front-end web app.
2. Back έλεγχος διαθεσιμότητας και αναζήτηση στην βάση για τα οχήματα του χρήστη
3. Επιλογή οχήματος που θέλει να φορτίσει ο χρήστης και αίτημα για προβολή προγραμμάτων.
4. Διαβίβαση αιτήματος από το front-end web app στο back-end της εφαρμογής για εξυπηρέτηση.
5. Έλεγχος ότι ο χρήστης είναι συνδεδεμένος μέσω αιτήματος του back-end στη redis Database.
6. Αίτημα στη MySQL από το back-end για προσκόμιση των δεδομένων που θέλει να προβάλει ο χρήστης.
7. Εξυπηρέτηση αιτήματος από τη MySQL στο back-end και προώθηση του αποτελέσματος από το back-end στο front-end μετά από την κατάλληλη μορφοποίηση.
8. Προβολή δυνατοτήτων φόρτισης στο χρήστη από το front-end.
9. Επιλογή προγράμματος από το χρήστη.
10. Μεταβίβαση αιτήματος για κατάληψη θέσης από το χρήστη από το front-end στο back-end
11. Έλεγχος ότι ο χρήστης είναι συνδεδεμένος μέσω αιτήματος του back-end στη redis Database.
12. Έλεγχος αιτήματος και προώθησή του στη βαση για δημιουργία καταχώρησης συνεδρίας φόρτισης.
13. Προβολή μηνύματος στο χρήστη για την έκβαση του αιτήματός του και αν είναι θετικό κατάληψη της θέσης φόρτισης
14. Αίτημα του χρήστη για πληρωμή της συναλλαγής με κάρτα
15. Διαβίβαση του αιτήματος από το front-end στο back-end
16. Αίτημα από το back-end στο το API πληρωμών για εξόφληση της συναλλαγής
17. Εξυπηρέτηση του αιτήματος από το API πληρωμών, προώθηση μηνύματος στο back-end
18. Αίτημα στη MySQL από το back-end για ανανέωση των πόντων του χρήστη
19. Εξυπηρέτηση αιτήματος από τη MySQL στο back-end και προώθηση του αποτελέσματος από το back-end στο front-end μετά από την κατάλληλη μορφοποίηση.

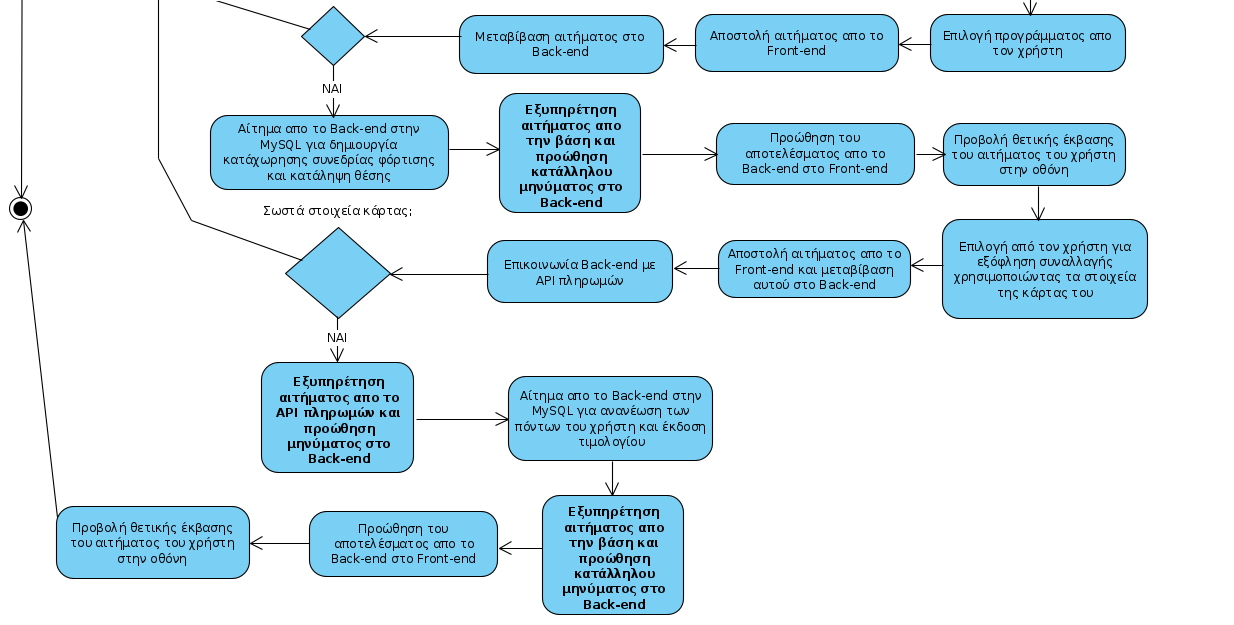
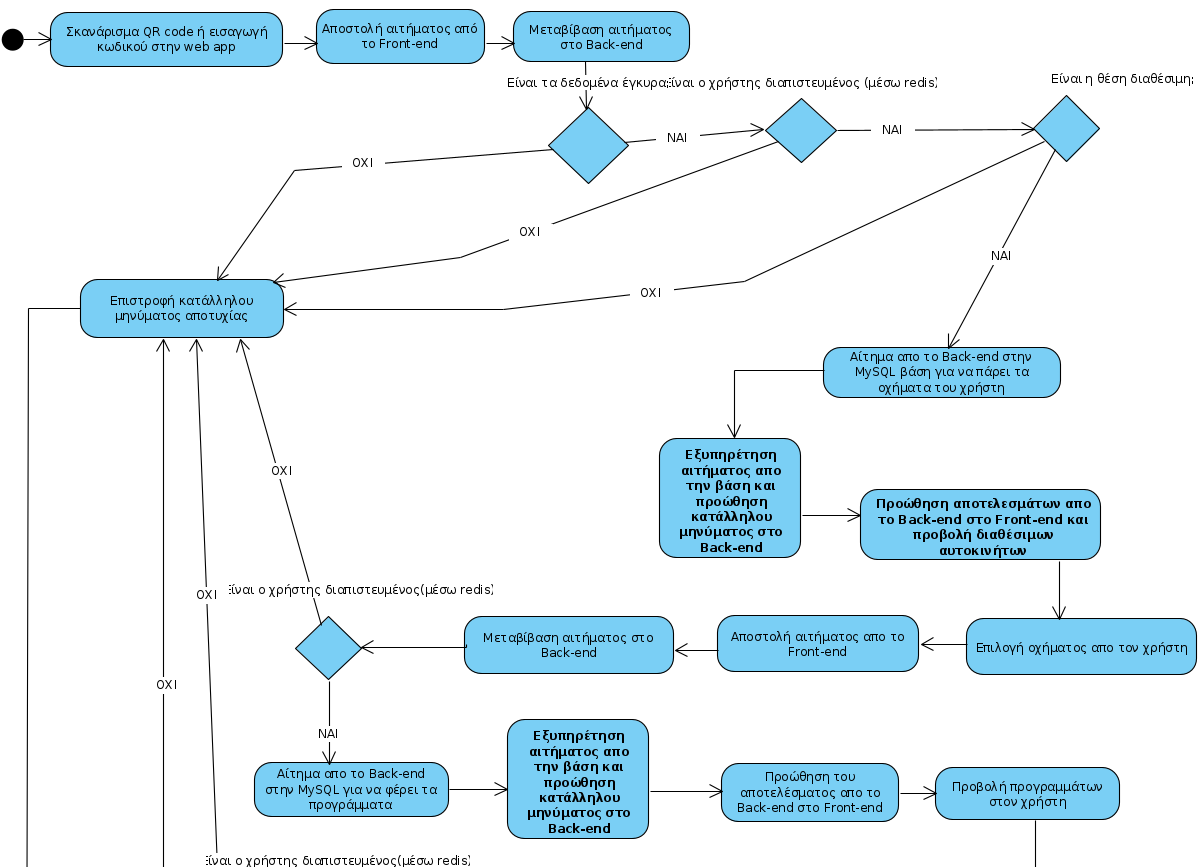
##### Use Case Diagram



##### Sequence Diagram

##### 

##### Activity Diagram



#### 2.1.2.7 Δεδομένα εξόδου

##### Sequence Diagram with exit

### 2.1.3 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 3: Προβολή overview από το διαχειριστή σταθμού για το σταθμό που διαχειρίζεται.

#### 2.1.3.1 Χρήστες (ρόλοι) που εμπλέκονται

* Εγγεγραμμένος χρήστης - Διαχειριστής ΣΕΟ
* Front - End Web app
* Back - end
* MySQL Database
* redis Database

#### 2.1.3.2 Προϋποθέσεις εκτέλεσης

Για να μπορεί να εκτελεστεί η παραπάνω ενέργεια θα πρέπει αρχικά ο χρήστης που θέλει να κάνει την κράτηση για φόρτιση να είναι εγγεγραμμένος στο σύστημά μας ως διαχειριστής σταθμού ΣΕΟ και να έχει ταυτοποιηθεί από το σύστημα, να είναι δηλαδή logged-in.

#### 2.1.3.3 Περιβάλλον εκτέλεσης

Η εκτέλεση του συγκεκριμένου σεναρίου χρήσης ξεκινάει από το **front end web-app client**, ο οποίος εξυπηρετεί το χρήστη- ιδιοκτήτη αυτοκινήτου. Το αίτημα του χρήστη αρχικά για προβολή overview δεδομένων, διαβιβάζεται από το front-end client στο **back-end**, το οποίο αναλαμβάνει την εξυπηρέτησή του. Εμπλέκεται επίσης και η **MySQL Database**, από την οποία λαμβάνονται τα δεδομένα. Τέλος πριν ο χρήστης καταφέρει να προσπελάσει δεδομένα τα οποία βρίσκονται στη MySQL Database, γίνεται επαλήθευση πως ο χρήστης είναι ταυτοποιημένος μέσω της **redis Database**.

#### 2.1.3.4 Δεδομένα εισόδου

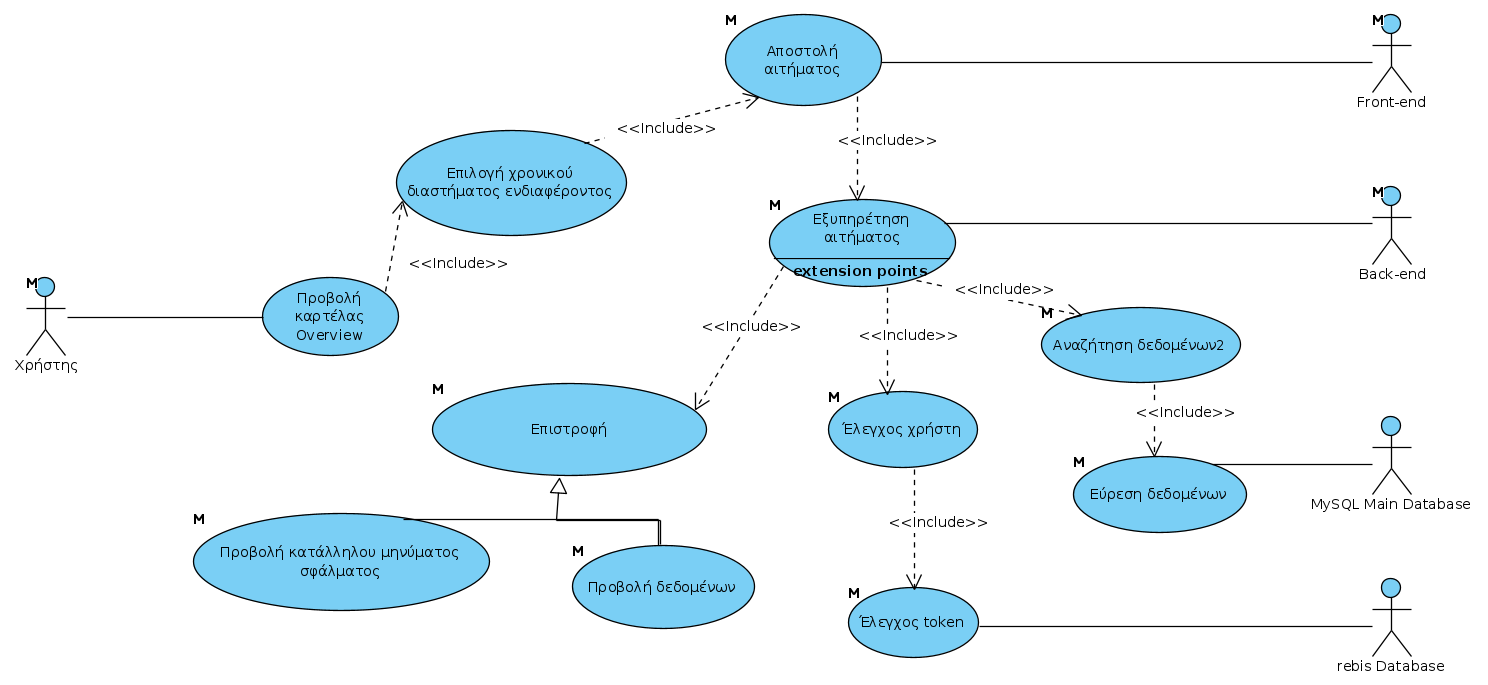
Δε λαμβάνουμε δεδομένα εισόδου, καθώς όλες οι απαραίτητες παράμετροι περνούν μέσω του URL.

#### 2.1.3.5 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

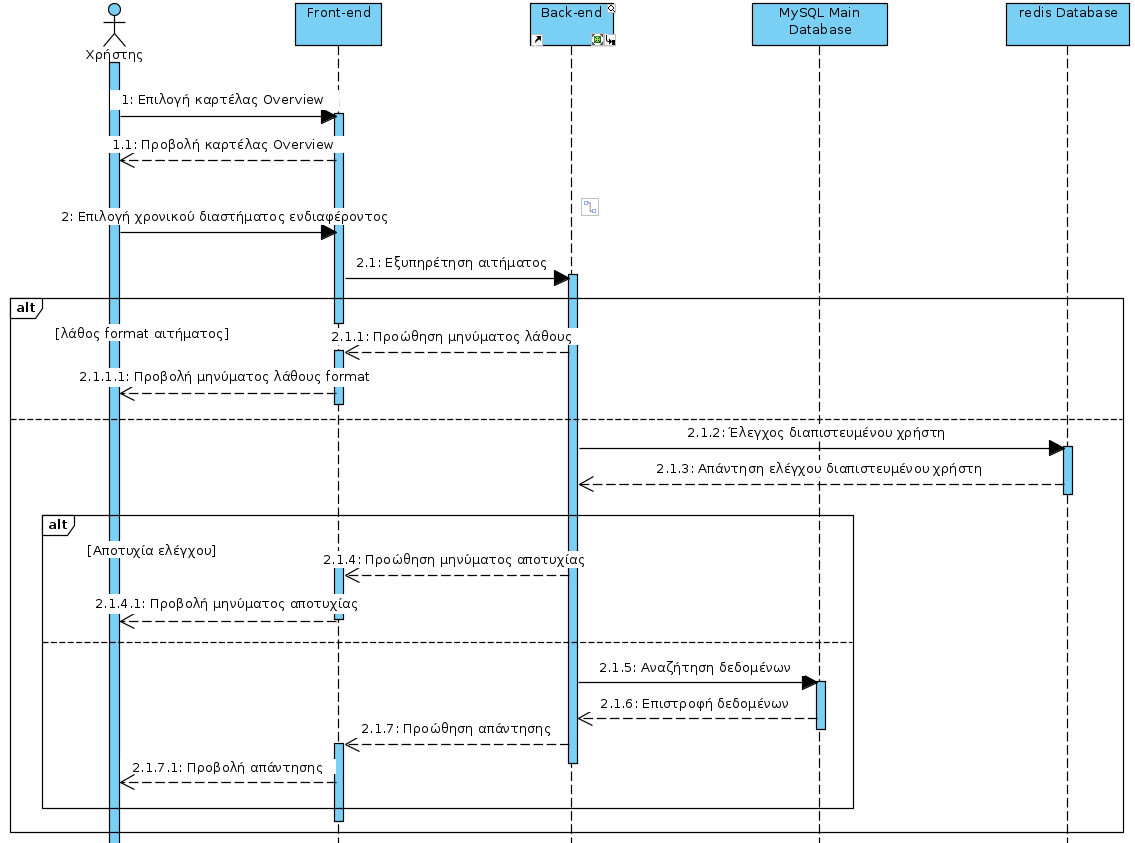
Η αλληλουχία ενεργειών φαίνεται στα παρακάτω βήματα:

1. Επιλογή της Καρτέλας Overview από τον χρήστη - Διαχειριστή ΣΕΟ στο front-end web app.
2. Επιλογή του χρονικού διαστήματος ενδιαφέροντος στο front-end και πάτημα πλήκτρου προβολή.
3. Αποστολή αιτήματος από το front-end στο back-end για εξυπηρέτηση.
4. Έλεγχος για ταυτοποίηση χρήστη από το back-end στη redis Database.
5. Αναζήτηση των κατάλληλων δεδομένων από το back-end στη MySQL Database.
6. Μορφοποίηση δεδομένων που επιστράφηκαν από τη βάση κατάλληλα από το back-end και προώθησή τους στο front-end..
7. Προβολή των δεδομένων από το front-end σε μορφή διαδραστικών γραφημάτων που οπτικοποιούν την πληροφορία (π.χ. heatmaps για τις ώρες αιχμής).

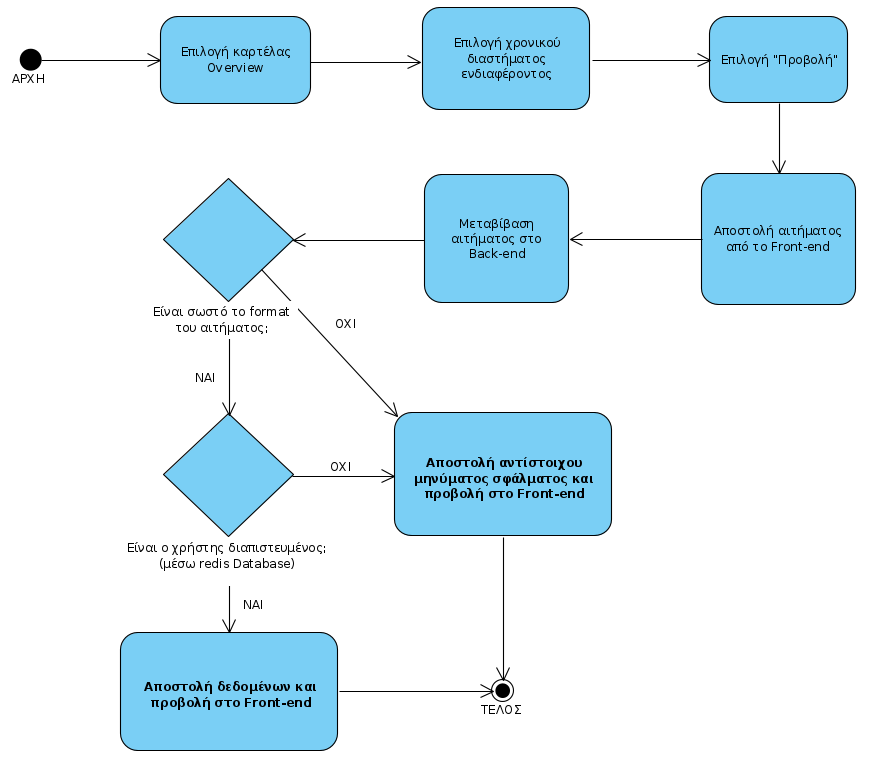
##### Use Case Diagram



##### Sequence Diagram

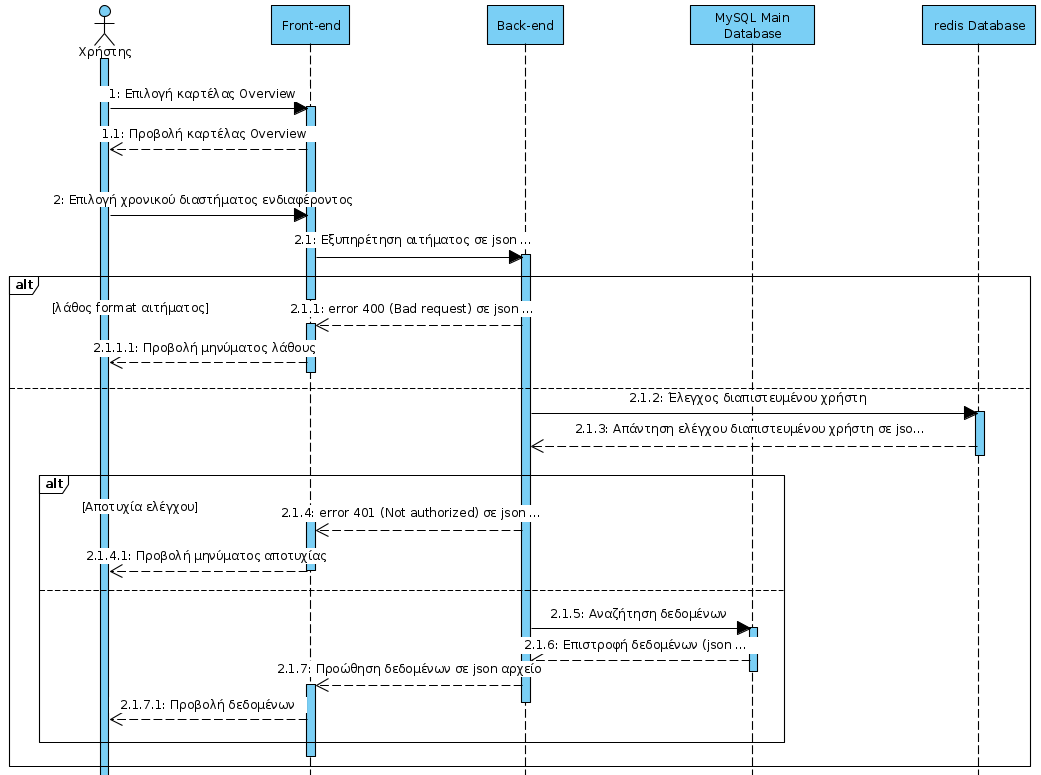


##### Activity Diagram



#### 2.1.2.7 Δεδομένα εξόδου

##### Sequence Diagram with exit



## 2.2 Απαιτήσεις επιδόσεων

Οι επιλογές που κάναμε κατά το σχεδιασμό και την υλοποίηση της εφαρμογής έγιναν με σκοπό της μεγιστοποίηση της επίδοσης, όσον αφορά τη δυνατότητα εξυπηρέτησης μεγάλου φορτίου αιτημάτων σε εύλογη ταχύτητα.

Οι σχεδιαστικές επιλογές που έγιναν με βάση αυτό το κριτήριο ήταν:

* Η επιλογή μιας ασύγχρονης γλώσσας, όπως η NodeJS (Javascript), για την υλοποίηση του back-end. Η γλώσσα αυτή μας επιτρέπει να εκτελούμε παράλληλα τμήματα κώδικα, ενώ λειτουργεί με ένα format με requests και responses που ταιριάζει απόλυτα στις απαιτήσεις κατασκευής ενός REST API.
* Η επιλογή των json web tokens (JWT), ως τρόπο ταυτοποίησης των χρηστών της εφαρμογής μας. H stateless φύση τους μας επιτρέπει να κάνουμε οικονομία χώρου και χρόνου, αφού δεν υπάρχει ανάγκη αποθήκευσής τους σε κάποια SQL βάση, η ανάκτηση από την οποία θα απαιτούσε queries προς σκληρό δίσκο.
* Η επιλογή της redis, μιας nonSQL database για την αποθήκευση των invalidated tokens. Η redis είναι in-memory, πράγμα που σημαίνει πως λειτουργεί στη RAM βελτιώνοντας τρομερά την ταχύτητα του ελέγχου.
* Η υλοποίηση της εφαρμογής με βάση τα πρότυπα της Model - View - Controller (MVC) αρχιτεκτονικής, ώστε η πρόσβαση στη βάση δεδομένων να γίνεται μόνο όταν αυτό είναι απολύτως απαραίτητο.

Όλα τα παραπάνω συντελούν στο να μπορεί η εφαρμογή μας να μπορεί να επεξεργαστεί ταυτόχρονα μεγάλο όγκο αιτημάτων και να ανταποκρίνεται σε αυτά με εύλογη ταχύτητα, αφού αποφεύγεται η πρόσβαση σε “αργό” σκληρό δίσκο, όπου αυτό είναι δυνατό.

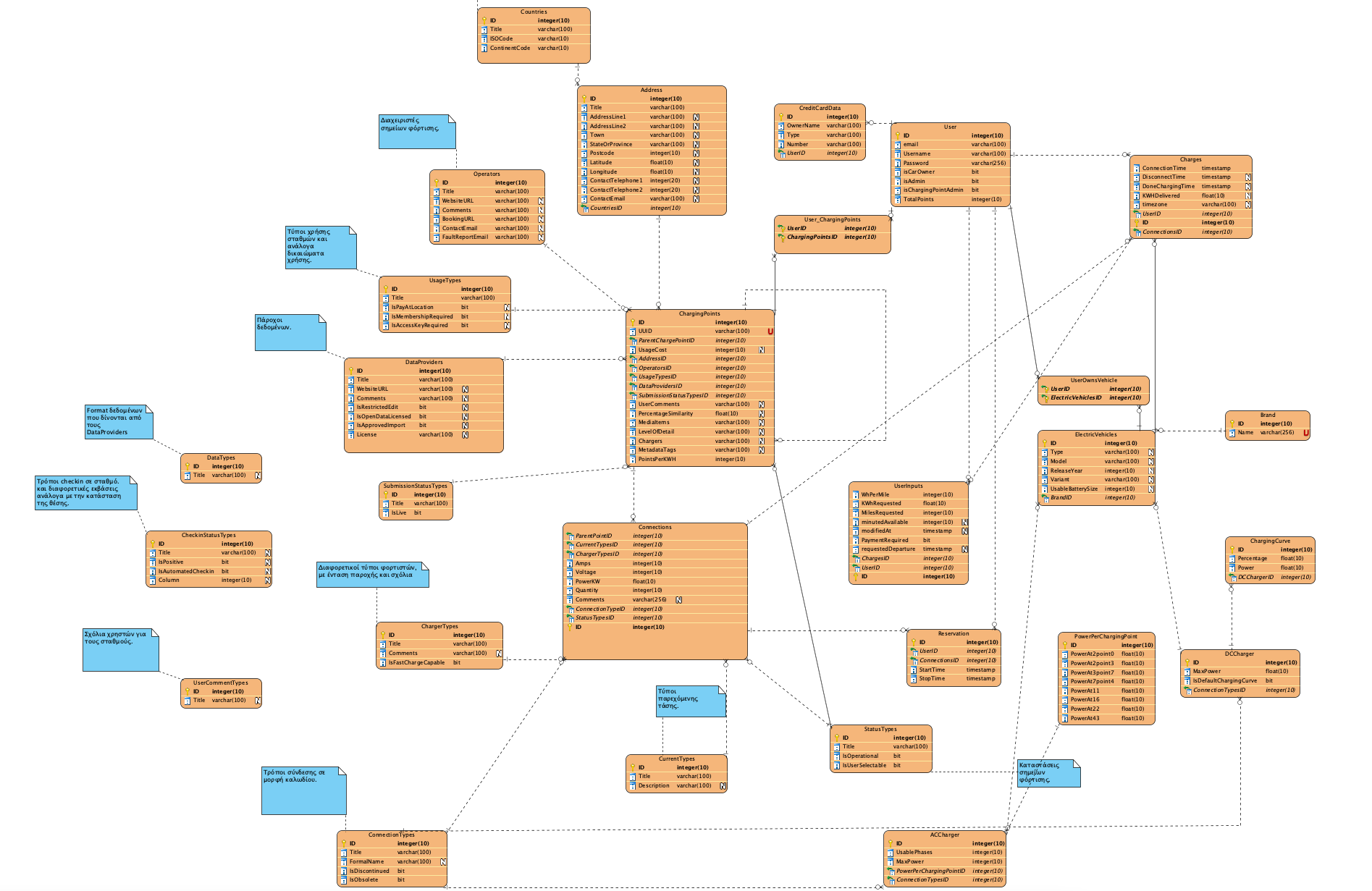
## 2.3 Απαιτήσεις οργάνωσης δεδομένων

### 2.3.1 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

Η εφαρμογή μας έχει 4 διαφορετικούς τύπους χρήστη με διαφορετικά επίπεδα δικαιωμάτων πρόσβασης στα δεδομένα:

* Μη διαπιστευμένος χρήστης: μπορεί να κάνει απόπειρα για διαπίστευση, εγγραφή και να ανακτήσει μόνο βασικά δεδομένα, όπως η θέση των Σταθμών και οι τιμές και τα προγράμματα φόρτισης στους σταθμούς.
* Διαπιστευμένος χρήστης - Ιδιοκήτης Οχήματος : ο χρήστης αυτός έχει όλα τα δικαιώματα του μη διαπιστευμένου χρήστη και επιπλέον τη δυνατότητα προβολής και μεταβολής των δεδομένων του οχήματός του και του λογαριασμού του. Επιπλέον, μπορεί να πραγματοποιήσει πληρωμές και να κλείσει ραντεβού για φόρτιση, ενέργειες που μεταφράζονται σε καταγραφές στη βάση.
* Διαπιστευμένος χρήστης - Διαχειριστής ΣΕΑ : ο χρήστης αυτός έχει όλα τα δικαιώματα του μη διαπιστευμένου χρήστη και επιπλέον τη δυνατότητα προβολής και μεταβολής των δεδομένων του ΣΕΑ που διαχειρίζεται. Επιπλέον, μπορεί να δεχτεί πληρωμές και να κλείσει ραντεβού για φόρτιση, ενέργειες που μεταφράζονται σε καταγραφές στη βάση.
* Διαχειριστής Συστήματος : ο χρήστης αυτός έχει δυνατότητα παρέμβασης σε όλα τα στοιχεία και όλους τους πίνακες της βάσης με σκοπό την εισαγωγή δεδομένων που δεν γίνεται να εισαχθούν αυτόματα.

Τα δεδομένα στη MySQL βάση την οποία χρησιμοποιούμε είναι δομημένα με βάση το παρακάτω διάγραμμα Οντοτήτων-Συσχετίσεων (ERD) :



## Δυστυχώς λόγω της μεγάλης έκτασής του δεν ήταν δυνατή η απεικόνισή του με λεπτομέρεια, αλλά είναι διαθέσιμο και στο .vpp αρχείο που επισυνάπτεται στο SRS.

## 2.4 Περιορισμοί σχεδίασης

Η εφαρμογή μας υλοποιεί στον κορμό της ένα RESTful API και πρέπει να συμμορφώνεται συγκεκριμένα με τα πρότυπα του Open API 3.0 , όπως η επικοινωνία με μέσω λήψης και αποστολής Requests, όπου οι πληροφορίες είναι κωδικοποιημένες σε μορφή json object με προκαθορισμένα πεδία.

## 2.5 Λοιπές απαιτήσεις

### 2.5.1 Απαιτήσεις διαθεσιμότητας λογισμικού

Η εφαρμογή μας είναι μια διαδικτυακή εφαρμογή και η διαθεσιμότητά της είναι στενά συνδεδεμένη με την διαθεσιμότητα του host server. Το γεγονός πως υποστηρίζει νευραλγικές διαδικασίες που συμβαίνουν όλο το 24ωρο απαιτεί την εξασφάλιση συνεχούς διαθεσιμότητας.

### 2.5.2 Απαιτήσεις ασφάλειας

Η εφαρμογή μας οφείλει να τηρεί αυστηρότατες προδιαγραφές ασφαλείας, τόσο λόγω της διαδικτυακής της φύσης που επιτρέπει σε κακόβουλους τρίτους να κάνουν απόπειρες για μη επιθυμητές ενέργειες, όσο και λόγω της ευαισθησίας των πληροφοριών που διαχειρίζεται, όπως τα στοιχεία ηλεκτρονικών πληρωμών κατά την πληρωμή με κάρτα. Τα αντίμετρα που παίρνουμε για να θωρακίσουμε την εφαρμογή μας είναι:

* Η υποστήριξη του πρωτοκόλλου ασφαλείας https, το οποίο αναγνωρίζεται από τους browsers και η χρήση SSL πιστοποιητικών υπογεγραμμένων από γνωστούς 3rd party παρόχουν τέτοιων υπηρεσιών. Η χρήση αυτού του πρωτοκόλλου θωρακίζει την εφαρμογή μας από επιθέσεις phising και spoofing, οι οποίες δεν μπορούν να παρέχουν το trusted πιστοποιητικό.
* Η εφαρμογή hashing στους κωδικούς των χρηστών, κατά την αποθήκευσή τους στη βάση, ώστε κανείς να μην έχει τη δυνατότητα να τους προβάλει σε plain-text μορφή, ανεξαρτήτως των δικαιωμάτων που αυτός έχει στην εφαρμογή. Έτσι ακόμα και στο ατυχές ενδεχόμενο ενός leak δεδομένων από τη βάση μας, οι κωδικοί των χρηστών μας είναι ασφαλείς αφού δεν είναι δυνατή η αντιστροφή της hash-function ως βαρύ υπολογιστικό πρόβλημα που είναι.
* Η διαπίστευση των χρηστών μας μέσω JWT tokens που μας προσφέρουν μια ασφαλή και γρήγορη μέθοδο ταυτοποίησης. Η διαδικασία του token generation περιλαμβάνει την υπογραφή τους από ένα μοναδικό μυστικό κλειδί, το οποίο έχει ο back-end server μας. Η λήξη (invalidation) της εγκυρότητας του token, γίνεται είτε μέσω του logout, είτε αυτόματα έπειτα από πεπερασμένο χρονικό διάστημα που εμείς έχουμε ορίσει. Τότε για να μην μπορεί να ξαναχρησιμοποιηθεί το ίδιο token γράφεται στη βάση redis, η οποία κρατάει όλα τα invalid tokens.
* Οι πληρωμές διευθετούνται μέσω τρίτου διακεκριμένου API για αυτή τη λειτουργία.

### 2.5.3 Απαιτήσεις συντήρησης

Καθ’ όλη τη διάρκεια της υλοποίησης της εφαρμογής και της συγγραφής του κώδικα λήφθηκε υπόψιν η ανάγκη για ευκολία της συντήρησής της. Έτσι λοιπόν, ο κώδικας είναι γραμμένος με βάση το πρότυπο MVC και με συχνή χρήση σχολίων, ώστε να μπορεί κανείς εύκολα να προσδιορίσει τι γίνεται σε κάθε κομμάτι και να μπορεί να επέμβει σε αυτό. Επιπλέον, οι βιβλιοθήκες που χρησιμοποιούμε είναι οι πλέον διαδεδομένες, ώστε να είμαστε βέβαιοι πως ενημερώονονται συχνά.