Έγγραφο απαιτήσεων λογισμικού (SRS)

*ΠΡΟΣΑΡΜΟΓΗ ΤΟΥ ΑΝΤΙΣΤΟΙΧΟΥ ΕΓΓΡΑΦΟΥ ΤΟΥ ΠΡΟΤΥΠΟΥ ISO/IEC/IEEE 29148:2011*

*ΣΗΜΕΙΩΣΗ: Το στυλ κειμένου (style) "Description" χρησιμοποιείται για να σας δώσει σύντομες οδηγίες για κάθε ενότητα. Το κείμενο που θα συμπληρώσετε πρέπει να γραφεί με το στυλ "Normal"*

GridNest

# Εισαγωγή

## 1.1 Εισαγωγή: σκοπός του λογισμικού

Το πρωτεύον εγχείρημα του λογισμικού της ομάδας GridNest είναι η ανάπτυξη διαδικτυακής πλατφόρμας, το οποίο θα προσφέρει πληροφορίες για τον ιδιοκτήτη ηλεκτρικού οχήματος και θα επιτρέπει αλληλεπίδραση μεταξύ των χρηστών και των τραπεζών-παρόχων υπηρεσιών πληρωμών . Βασικός στόχος των πληροφοριών αυτών αποτελεί η βελτιστοποίηση της διαδικασίας φόρτισης του αυτοκινήτου κάθε χρήστη, αλλά και η δυνατότητα βελτιστοποίησης της διαδικασίας των πληρωμών, με γρήγορη πρόσβαση σε πλατφόρμες πληρωμών, με δυνατότητα επιλογής ανάμεσα σε ποικιλία τρόπων πληρωμής και ανάμεσα σε εύρος οικονομικών δεδομένων (π.χ. μηνιαίος περιοδικός λογαριασμός).

Πιο συγκεκριμένα:

1. Ως προς τον ιδιοκτήτη οχήματος με μπαταρία, θα παρέχεται η δυνατότητα, μέσω της ιστοσελίδας και ενός προσωπικού προφίλ, να επιλέξει ανάμεσα σε διάφορα προγράμματα χρέωσης για τη φόρτιση του οχήματος του, τον τρόπο πληρωμής αλλά και να δύναται να προσδιορίσει σε ζωντανό χρόνο το κόστος φόρτισης του οχήματος του και το χρόνο αναμονής. Επίσης, παρέχονται διάφορα στατιστικα΄γραφήματα τα οποία απεικονίζουν αρκετά δεδομένα με συνεπή-λογικό τρόπο ώστε να μπορεί να εξαχθεί ποιοτική πληροφορία για τον ιδιοκτήτη άμεσα.
2. Ως προς τις τράπεζες και τους παρόχους υπηρεσιών πληρωμών, η ιστοσελίδα θα καθιστά δυνατή την πρόσβαση σε αυτές, για πραγματοποίηση συναλλαγών με τους χρήστες και ως αποτέλεσμα την συλλογή οικονομικών δεδομένων. Ωστόσο, το κυριότερο πλεονέκτημα είναι αμφίδρομη πρόσβαση σε οικονομικά δεδομένα, καθώς και οι ίδιοι χρήστες θα μπορούν να παρακολουθούν τις συναλλαγές τους συνολικά σε χρονικά διαστήματα.

Αξίζει να αναφερθεί, ότι για να εξασφαλιστούν οι παραπάνω επιδιώξεις του συγκεκριμένου λογισμικού κρίνεται αναγκαία η προσαρμογή του, ώστε να επιτυγχάνεται βέλτιστη και εξατομικευμένη αλληλεπίδραση με τον εκάστοτε χρήστη. Οπότε, μείζονος σημασίας σκοπός είναι η δημιουργία σύγχρονου περιβάλλοντος, με την επίκληση καλών πρακτικών ανάπτυξης λογισμικού, που θα διευκολύνουν την περιήγηση του χρήστη στην πλατφόρμα.

Παράλληλα, μέσω της δυνατότητας αξιολόγησης του εκάστοτε σταθμού φόρτισης από κάθε επισκέπτη, θα εξασφαλιστεί η μέγιστη ικανοποίηση κάθε ιδιοκτήτη οχήματος. Αυτό συμβαίνει διότι, από τη μία πλευρά θα καταστεί δυνατή η επιλογή του καλύτερου δυνατού σταθμού για να φορτιστεί κάποιο όχημα και από την άλλη πλευρά, θα διαμορφωθεί ένα δυναμικό περιβάλλον για το δίκτυο παροχής φόρτισης, που θα προσαρμόζεται στις ανάγκες του ευρύτερου κοινού(ιδιοκτήτες ηλ. οχημάτων).

Καταληκτικά, θεμελιώδης στόχος της συγκεκριμένης εφαρμογής τεχνολογίας λογισμικού είναι η δυνατότητα διαφοροποίησης της από παρεμφερείς πλατφόρμες. Αυτό, θα εξασφαλιστεί, κυρίως, μέσω της ευρείας ποικιλίας των δεδομένων που διατίθενται, το οποίο συνεπάγεται περισσότερο προσωποποιημένη και ολοκληρωμένη παροχή υπηρεσιών για κάθε ιδιοκτήτη ηλ. οχήματος, αλλά και μέσω της προσεκτικής υλοποίησης της εφαρμογής, σε συνολικό επίπεδο, ώστε να την προτιμήσει ο χρήστης. Ακόμα, θα είναι η μόνη εφαρμογή που συνδέει φυσικά πρόσωπα(ιδιοκτήτες ηλ.οχημάτων) και νομικά πρόσωπα(τράπεζες και πάροχοι υπηρεσιών πληρωμών) με τόσο αποτελεσματικό τρόπο, διευκολύνοντας την αποπεράτωση των επιμέρους εργασιών τους. Ολοκληρώνοντας, θα δομείται με τέτοιο τρόπο, που θα είναι εύκολα επεκτάσιμη για να ικανοποιήσει ανάγκες και άλλων φορέων που εμπλέκονται στη διαδικασία φόρτισης ενός ηλεκτρικού οχήματος (π.χ. Υπουργείο περιβάλλοντος και Ενέργειας, Κατασκευαστές ηλ. οχημάτων), γεγονός που την καθιστά εξελίξιμη σε σχέση με οτιδήποτε άλλο κυκλοφορεί στην αγορά.

## 1.2 Διεπαφές (interfaces)

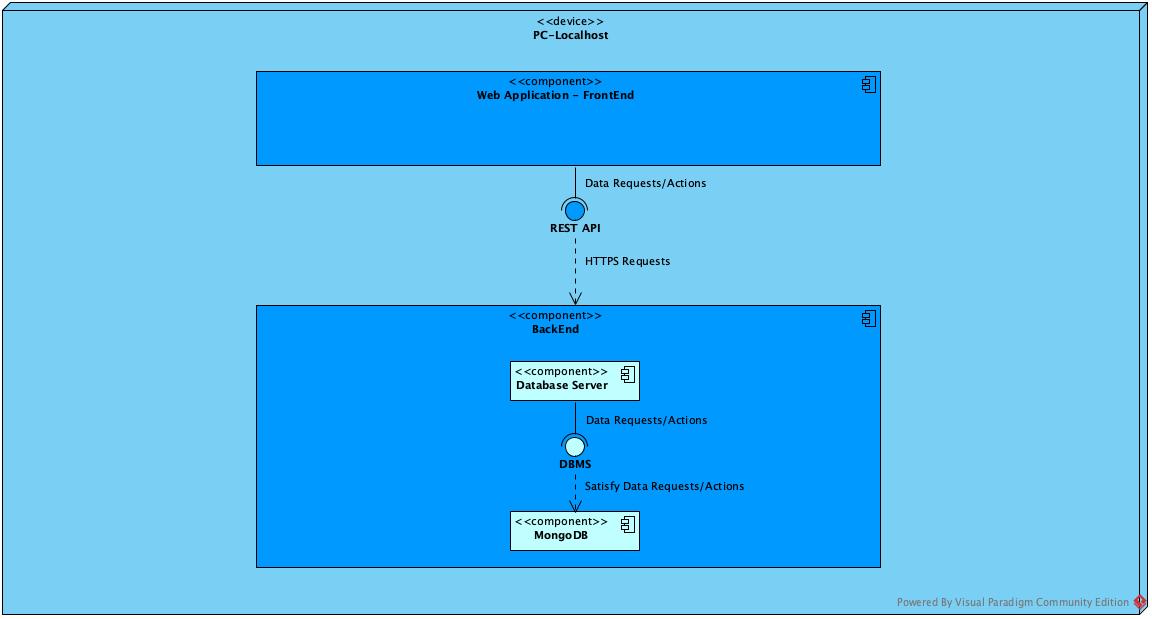
### 1.2.1 Διεπαφές με εξωτερικά συστήματα

*Προδιαγραφή διεπαφών με εξωτερικά συστήματα και λογισμικό, με αναφορά σε πρότυπα ανταλλαγής δεδομένων και κλήσης υπηρεσιών. Χρήση διαγραμμάτων deployment UML.*

Το σύστημα μας για να λειτουργήσει επιτυχώς βασίζεται στις παρακάτω εσωτερικές διεπαφές.

1. **API για BackEnd-MongoDB RDBMS Server επικοινωνία:** H συγκεκριμένη διεπαφή αφορά την επικοινωνία, μέσω DBMS statements, του BackEnd συστήματος με τη NoSQL βάση. Πιο συγκεκριμένα, έχοντας αποθηκεύσει τα περιορισμένα δεδομένα στη βάση, καθίσταται δυνατό μέσω εντολών που ήδη υπάρχουν στη βιβλιοθήκη της mongodb, είτε μέσω εξατομικευμένων queries που δημιουργήσαμε για τους σκοπούς της εφαρμογής, να ανακτηθούν ή να τροποποιηθούν.
2. **REST API:** Ο στόχος μας είναι η πρόσβαση του FrontEnd συστήματος στα κληθέντα δεδομένα από τον εκάστοτε χρήστη, ανάλογα με τα δικαιώματα πρόσβασης του σε αυτά. Για αυτό το λόγο, χρησιμοποιείται ένα RESTful API(REpresentational State Transfer Application Programming Interface), ως δίαυλος επικοινωνίας μεταξύ της BackEnd και FrontEnd εφαρμογής. Ειδικότερα, το REST API, δια μέσου HTTPS(S stands for security) requests, υποστηρίζει τις λειτουργίες get, put, post, delete, patch στα δεδομένα. Ωστόσο, για να καταστούν αυτές δυνατές από κάποιο χρήστη πρέπει, όπως αναφέρθηκε προηγουμένως, να ελεγχθούν τα δικαιώματα του. Οπότε, αρχικά γίνεται ένα post request με το όνομα και τον κωδικό του χρήστη και αν ληφθεί authorization token πραγματοποιούνται οι επιθυμητές ενέργειες πάνω στα δεδομένα της βάσης.

UML Deployment Diagram

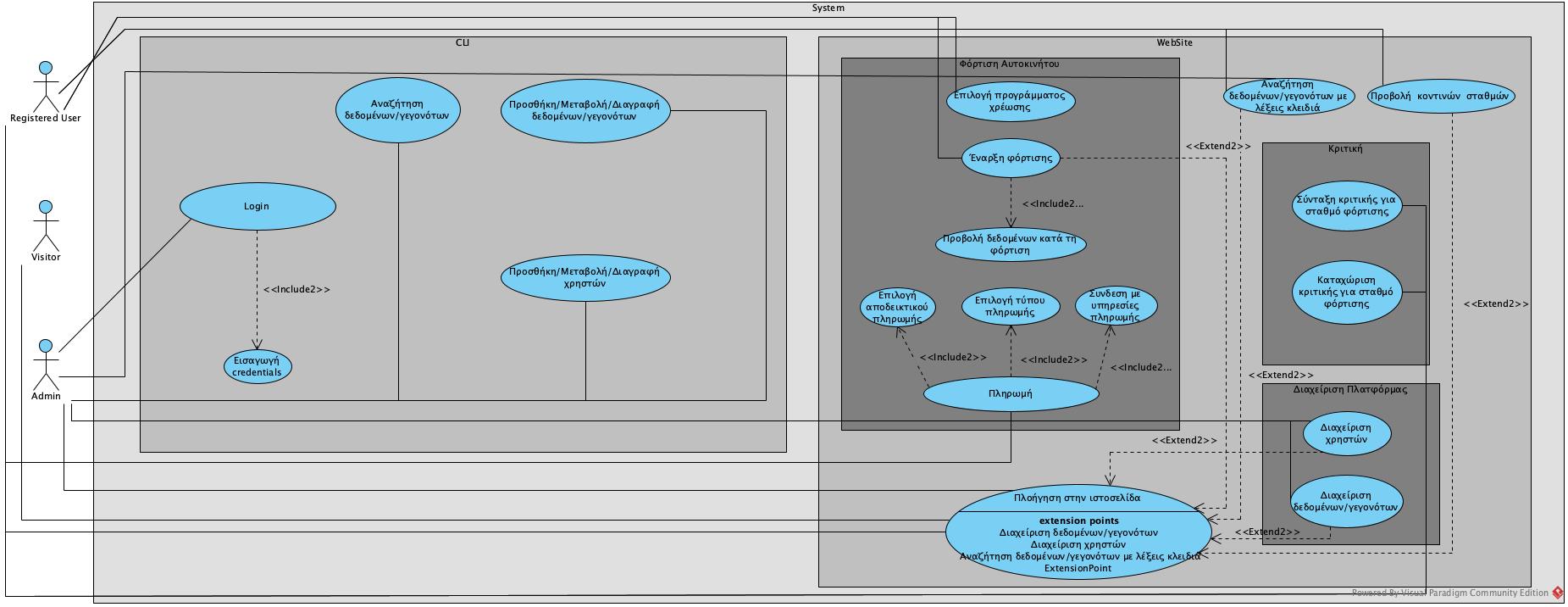


### 1.2.2 Διεπαφές με το χρήστη

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

1. **Eγγεγραμμένος χρήστης:** Αυτός ο τύπος χρήστη της εφαρμογής έχει πρόσβαση στο WebSite-FrontEnd σύστημα. Εκεί, έχει τη δυνατότητα πλοήγησης όπου μπορεί να αναζητήσει και να καταχωρήσει πληροφορίες-δεδομένα(π.χ. εύρεση κοντινότερου σταθμού- σύνταξη κριτικής για τους σταθμούς - συνολική κατανάλωση ενέργειας από κάποιο σταθμό σε ένα χρονικο διάστημα κλπ.). Ακόμα, καθίσταται δυνατή, η εκτέλεση ορισμένων λειτουργιών αυτοματοποίησης της φόρτισης του οχήματος (όπως επιλογή τύπου πληρωμής, επιλογή προγράμματος χρέωσης κλπ.).
2. **Μη εγγεγραμμένος χρήστης:** Ο συγκεκριμένος τύπος χρήστη ενώ έχει τη δυνατότητα πλοήγησης και αναζήτησης γενικών πληροφοριών, δεν έχει πρόσβαση στις προαναφερθείσες υπηρεσίες αυτοματοποίησης της διαδικασίας φόρτισης, ενώ παράλληλα, δεν επιτρέπεται η καταχώρηση κριτικών για οτιδήποτε.
3. **Διαχειριστής (Admin):** Έχει πρόσβαση, πέρα από την ιστοσελίδα, στο CLI(BackEnd) σύστημα. Σε αυτά δύναται να διαχειριστεί τους λογαριασμούς των χρηστών και μπορεί να τροποποιήσει καταχωρήσεις στη βάση δεδομένων. Ουσιαστικά, έχει αυτές τις δύο επιπλέον δυνατότητες μαζί με τις λειτουργίες των δύο προαναφερθέντων τύπων εγγεγραμμένων χρηστών.

UML Use Case Diagram



# Αναφορές - πηγές πληροφοριών

*Αναφορά σε πηγές πληροφοριών στο μέτρο της αναγκαιότητας για την κατανόηση του συστήματος*

[*https://www.mulesoft.com/resources/api/what-is-rest-api-design*](https://www.mulesoft.com/resources/api/what-is-rest-api-design)

[*https://www.mongodb.com/what-is-mongodb*](https://www.mongodb.com/what-is-mongodb)

# Προδιαγραφές απαιτήσεων λογισμικού

## 3.1 Περιπτώσεις χρήσης

H εφαρμογή αυτή μπορεί να χρησιμοποιηθεί με 2 διαφορετικούς τρόπους. O πρώτος τρόπος αφορά την διαδικασία φόρτισης ενός ηλεκτρικού οχήματος. Κατά την περίπτωση αυτή ο χρήστης ο οποίος βρίσκεται σε ενα σταθμό φόρτισης και αλληλεπιδρά με την εφαρμογή αρχικά κάνει sign-in στο σύστημα.Έπειτα επιλέγει τον τύπο οχήματος του, το πρόγραμμα χρέωσης το οποίο δημιουργείται δυναμικά και τον τρόπο πληρωμής.Ταυτόχρονα με την έναρξη της φόρτισης εμφανίζεται στην διεπαφή το κόστος φόρτισης που ολοκληρώνεται.

Η δεύτερη περίπτωση χρήσης αφορά την έκδοση περιοδικού λογαριασμού ανα όχημα.Ο χρήστης αφού κάνει sign in απο οποιαδήποτε web διεπαφή επιλέγει το όχημα και την χρονική περίοδο για την οποία επιθυμεί και ως αποτέλεσμα εκδίδεται ο περιοδικός λογαριασμός.

### 3.1.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: (Φόρτιση ηλεκτρικού οχήματος)

#### 3.1.1.1 Χρήστες (ρόλοι) που εμπλέκονται

*Αναφορά στους ρόλους που αφορά η περίπτωση χρήσης*

Οι ρόλοι που εμπλέκονται είναι αυτοί των εγγεγραμμένων χρηστων.

#### 3.1.1.2 Προϋποθέσεις εκτέλεσης

*Καταγραφή των συνθηκών που πρέπει να ισχύουν ώστε να μπορεί να εκτελεστεί η περίπτωση χρήσης*

Να είναι οι χρήστες εγγεγραμμένοι.

Να βρίσκονται οι χρήστες σε έναν σταθμό φόρτισης.

#### 3.1.1.3 Περιβάλλον εκτέλεσης

*Αναφορά στο περιβάλλον στο οποίο εκτελείται η περίπτωση χρήσης. Πχ "διαδικτυακή διεπαφή χρήστη", "DBMS" κλπ*

Το περιβάλλον εκτέλεσης είναι η διαδικτυακή διεπαφή του σταθμού φόρτισης.

#### 3.1.1.4 Δεδομένα εισόδου *Καταγραφή δεδομένων εισόδου και εξόδου και συνθηκών εγκυρότητας αυτών.*

Δεδομένα εισόδου:

⦁ Όνομα χρήστη, κωδικός

⦁ Εισαγωγή τύπου αυτοκινήτου / Επιλογή τύπου αυτοκινήτου από λίστα

⦁ Επιλογή προγράμματος χρέωσης

⦁ Επιλογή τρόπου πληρωμής

⦁ Αξιολόγηση σταθμού

Συνθηκες εγκυροτητας:

⦁ Όνομα χρήστη και κωδικός να μην είναι κενά

⦁ Όνομα χρήστη να υπάρχει στην βάση

⦁ Σωστός κωδικός

⦁ Ο τύπος του αμαξιού να υπάρχει στην βάση

Δεδομένα εξόδου:

⦁ Απόδειξη φόρτισης

⦁ Μήνυμα λάθους

⦁ Πιθανά προγράμματα χρέωσης

⦁ Κόστος φόρτισης που ολοκληρώνεται

*3.1.2.5 Παράμετροι*

Καταγραφή παραμέτρων και συνθηκών εγκυρότητας αυτών, εφόσον υπάρχουν παράμετροι.

Παράμετρος : Τύπος αυτοκινήτου Τύπος: String Περιορισμοί : Να υπάρχει στην βάση

Παράμετρος : Αξιολόγηση Τύπος: List of String Περιορισμοί : <= 5000 characters

#### 3.1.1.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

*Περιγραφή με κείμενο (Βήμα 1, Βήμα 2 κλπ) και διαγράμματα UML αλληλουχίας (Sequence) και δραστηριοτήτων (Activity). Περιλαμβάνεται η συμπεριφορά σε απρόβλεπτες καταστάσεις και σφάλματα (εναλλακτικές ροές).*

Βήμα 1 : Επιλογή φόρτισης ηλεκτρικού οχήματος

Βήμα 2: Sign in

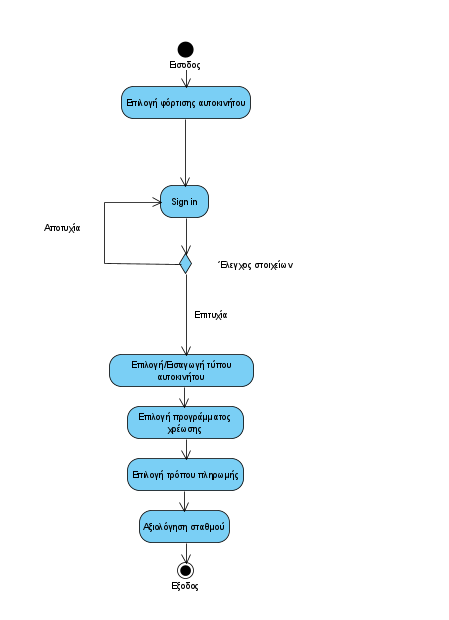
Βήμα 3: Είσοδος/Επιλογή τύπου αυτοκινήτου

Βήμα 4: Επιλογή προγράμματος χρέωσης από λίστα

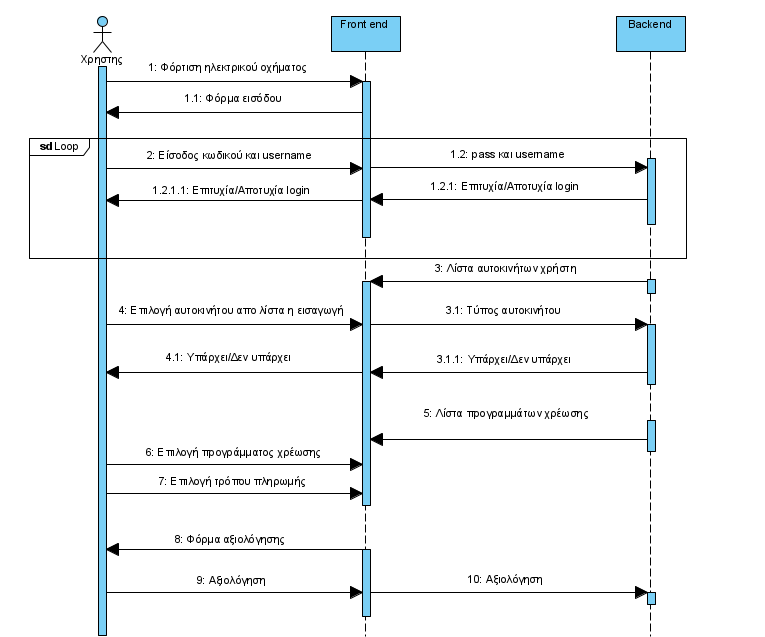
Βήμα 5: Επιλογή τρόπου πληρωμής

Βήμα 6 :Αξιολόγηση του σταθμού

UML Activity Diagram



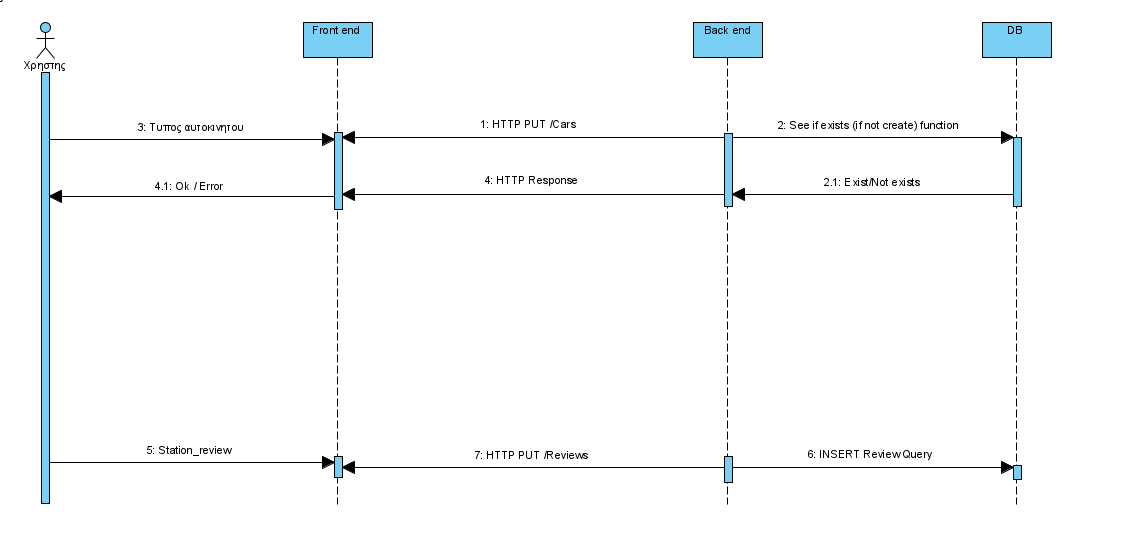
UML Sequence Diagram



#### 3.1.1.7 Δεδομένα εξόδου

*Διαγράμματα UML αλληλουχίας για την παραγωγή δεδομένων εξόδου. Ως δεδομένα εξόδου νοούνται όλα τα δεδομένα του συστήματος τα οποία δημιουργούνται ή μεταβάλλονται κατά την εκτέλεση (αν υπάρχουν τέτοια)*

UML Data Sequence Diagram

**

#### 3.1.1.8 Παρατηρήσεις

N/A.

### 3.1.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: (Έκδοση περιοδικού λογαριασμού ανα όχημα)

#### 3.1.2.1 Χρήστες (ρόλοι) που εμπλέκονται

Οι ρόλοι που εμπλέκονται είναι αυτοί των εγγεγραμμένων χρηστων.

#### 3.1.2.2 Προϋποθέσεις εκτέλεσης

*Καταγραφή των συνθηκών που πρέπει να ισχύουν ώστε να μπορεί να εκτελεστεί η περίπτωση χρήσης*

Να είναι οι χρήστες εγγεγραμμένοι.

Να έχει χρησιμοποιήσει ο χρήστης την εφαρμογή για φόρτιση στο παρελθόν.

#### 3.1.2.3 Περιβάλλον εκτέλεσης

*Αναφορά στο περιβάλλον στο οποίο εκτελείται η περίπτωση χρήσης. Πχ "διαδικτυακή διεπαφή χρήστη", "DBMS" κλπ*

Το περιβάλλον εκτέλεσης είναι η διαδικτυακή διεπαφή του σταθμού φόρτισης αλλα και η web εφαρμογή από οποιονδήποτε browser.

#### 3.1.2.4 Δεδομένα εισόδου

*Καταγραφή δεδομένων εισόδου και εξόδου και συνθηκών εγκυρότητας αυτών.*

Δεδομένα εισόδου:

* Όνομα χρήστη, κωδικός
* Επιλογή τύπου αυτοκινήτου από λίστα
* Επιλογή χρονικής περιόδου απο λιστα.

Συνθηκες εγκυροτητας:

* Όνομα χρήστη και κωδικός να μην είναι κενά
* Όνομα χρήστη να υπάρχει στην βάση
* Σωστός κωδικός

Δεδομένα εξόδου:

* Περιοδικός λογαριασμός
* Μήνυμα λάθους κατά το sign in
* Λίστα οχημάτων
* Χρονικές περίοδοι σε λίστα

#### 3.1.2.5 Παράμετροι

N/A

#### 3.1.2.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

*Περιγραφή με κείμενο (Βήμα 1, Βήμα 2 κλπ) και διαγράμματα UML αλληλουχίας (Sequence) και δραστηριοτήτων (Activity). Περιλαμβάνεται η συμπεριφορά σε απρόβλεπτες καταστάσεις και σφάλματα (εναλλακτικές ροές).*

Βήμα 1 : Sign in.

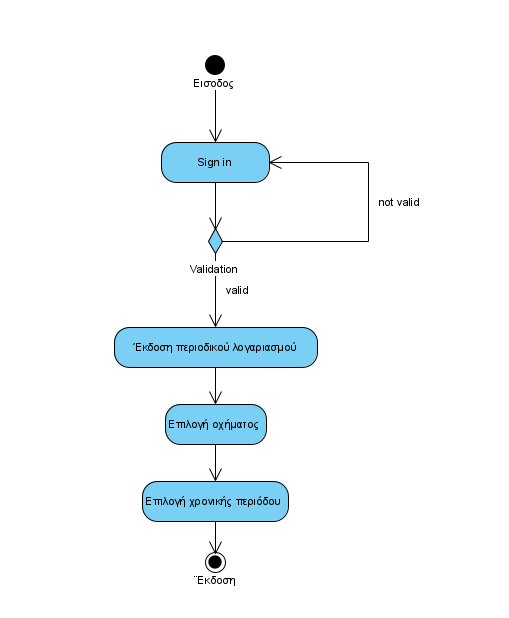
Βήμα 2 : Επιλογή έκδοσης περιοδικού λογαριασμού.

Βήμα 3 : Επιλογή οχήματος από λίστα.

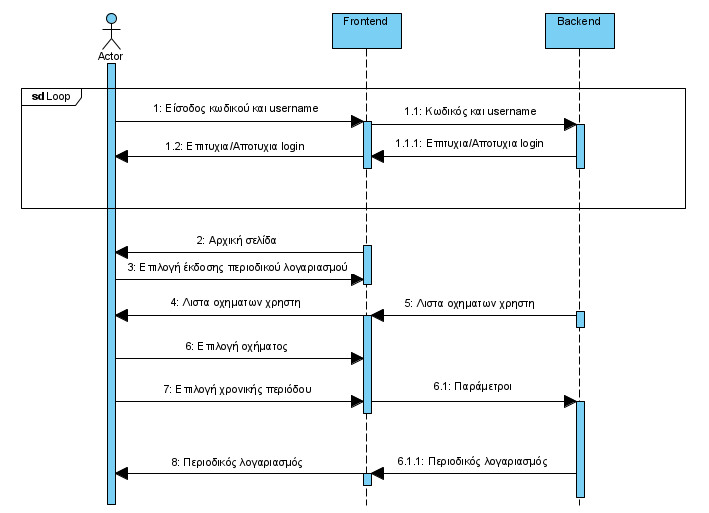
Βήμα 4 : Επιλογή χρονικής περιόδου.

Βήμα 5 : Έκδοση.

UML Activity Diagram



UML Sequence Diagram



#### 3.1.2.7 Δεδομένα εξόδου

*Διαγράμματα UML αλληλουχίας για την παραγωγή δεδομένων εξόδου. Ως δεδομένα εξόδου νοούνται όλα τα δεδομένα του συστήματος τα οποία δημιουργούνται ή μεταβάλλονται κατά την εκτέλεση (αν υπάρχουν τέτοια)*

N/A

#### 3.1.2.8 Παρατηρήσεις

Σε περίπτωση που ο χρήστης δεν έχει φορτίσει το αυτοκίνητο του ποτέ με την εφαρμογή και συνεπώς αυτό δεν βρίσκεται στην βάση επιστρέφεται κενή λίστα και δεν μπορεί αυτός να συνεχίσει.

*. . .*

*(η ενότητα 3.1.Χ.1 - 3.1.Χ.8 επαναλαμβάνεται για όλες τις περιπτώσεις χρήσης που συμπεριλαμβάνονται στο έγγραφο, όπως απαιτούνται από τις ομάδες ανάλογα με τον αριθμό των ατόμων)*

## 3.2 Απαιτήσεις επιδόσεων

*Ποσοτική τεκμηρίωση μέτρων και κριτηρίων επιθυμητών επιδόσεων με αναφορά στα ποσοτικά χαρακτηριστικά εισόδων και φορτίου του λογισμικού.*

Οι απαιτήσεις του συστήματος αφορούν, κυρίως, την αποδοτική αλληλεπίδραση του χρήστη με αυτό καθώς και ποσοτικές μετρήσεις για την ταχύτητα εκτέλεσης των λειτουργιών που πλαισιώνονται σε αυτό.

Όσον αφορά την αλληλεπίδραση με το σύστημα, κρίνεται απαραίτητο να είναι εμφανή τα σημεία-buttons που ο χρήστης κάνει εγγραφή ή συνδέεται στο λογαριασμό του, ώστε να επιλεχθούν τάχιστα. Ύστερα, διαδικασίες όπως της πληρωμής, της φόρτισης, της σύνταξης κριτικής για κάποιο σταθμό φόρτισης και η αναζήτηση/καταχώρηση πληροφοριών, θα πρέπει να προβάλλονται με κατανοητό-ευδιάκριτο τρόπο στην οθόνη της συσκευής του χρήστη, πάλι για λόγους επίδοσης. Ειδικότερα, για τη διαδικασία αναζήτησης, θα υφίστανται φίλτρα τα οποία θα διευκολύνουν ικανοποιητικά την ανάπτυξη querries από τους χρήστες, μειώνοντας το χρόνο ολοκλήρωσης της.

Από την άλλη πλευρά, το σύστημα για να λειτουργήσει ομαλά και σε περιορισμένο χρόνο, θα απαιτεί ταχεία σύνδεση στο διαδίκτυο από το χρήστη, ενώ συγχρόνως, το REST API θα οφείλει να υποστηρίζει πληθώρα requests το δευτερόλεπτο, από τη βάση δεδομένων. Αυτο θα συμβαίνει διότι, η αρχιτεκτονική της εφαρμογής θεμελιώνεται στη χρήση διαδικτύου για απόκτηση πρόσβαση στις λειτουργίες της και στους πόρους της, ενώ επίσης, υπάρχει το ενδεχόμενο κλήσης ή καταχώρησης δεδομένων στη βάση από μεγάλο αριθμό ατόμων το δευτερόλεπτο.

## 3.3 Απαιτήσεις οργάνωσης δεδομένων

### 3.3.1 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

*Απαιτήσεις πρόσβασης και περιορισμοί.*

1. Πρόσβαση συνολικά στα δεδομένα της βάσης θα παρέχεται μόνο στους χρήστες με την ιδιότητα του διαχειριστή της εφαρμογής (admin).

Αξίζει να σημειωθεί ότι η πρόσβαση για την παραπάνω περίπτωση θα πραγματοποιείται, ως επί το πλείστον, από το CLI του BackEnd συστήματος. Ωστόσο, η διαχείριση των χρηστών θα καταστεί δυνατή και από το WebSite.

1. Σε μικρότερη κλίμακα, οι εγγεγραμμένοι χρήστες θα μπορούν να επεξεργάζονται περιορισμένο εύρος δεδομένων στη βάση (κυρίως αυτά που αφορούν το προφίλ τους και τους σταθμους φόρτισης), ενώ θα έχουν πρόσβαση σε αρκετά δεδομένα, αναφορικά με την διαδικασία της φόρτισης του οχήματος τους, τα οποία όμως τους αφορούν και συμφωνούν επίσης με το GDPR.
2. Τέλος, οι μη εγγεγραμμένοι χρήστες, θα έχουν πρόσβαση σε πολύ μικρό εύρος δεδομένων (π.χ. προβολή κοντινότερου σταθμού), χωρίς δυνατότητα οποιασδήποτε επεξεργασίας.

Αντίθετα, για τις τελευταίες δύο περιπτώσεις, το WebSite θα αποτελεί το αποκλειστικό μέσο πρόσβασης σε δεδομένα.

## 3.4 Περιορισμοί σχεδίασης

*Λεπτομερής τεχνική τεκμηρίωση των περιορισμών σχεδίασης οι οποίοι επιβάλλονται από απαιτήσεις συμμόρφωσης σε πρότυπα, κανονισμούς, ή άλλους περιορισμούς του έργου. Περιλαμβάνεται η πολιτική ονοματολογίας οντοτήτων δεδομένων και πεδίων. Τέτοιοι περιορισμοί μπορεί να επιβάλλονται από τη χρήση βιβλιοθηκών, frameworks, περιβαλλόντων ανάπτυξης κλπ*

1. Χρήση ονοματολογίας οντοτήτων των δοσμένων δεδομένων, καθώς και διατήρηση σχέσεων μεταξύ αυτών των οντοτήτων.
2. Συμμόρφωση με την απαίτηση το BackEnd σύστημα να έχει πρόσβαση αποκλειστικά στη βάση δεδομένων και το REST API να λειτουργεί ως κόμβος επικοινωνίας μεταξύ FrontEnd και BackEnd.
3. Συντονισμός της ομάδας για την ανάπτυξη του συστήματος μέσω ορισμένων προτύπων συνεισφοράς:
4. GitHub Issues/Branching
5. Visual Paradigm Online - Community Edition v16.2
6. Χρήση των παρακάτω τεχνολογικών προτύπων:
7. BackEnd:

* MongoDB v4.4.1
* Node.JS v15.4.0

1. FrontEnd:

* React.JS v17.0.1

## 3.5 Λοιπές απαιτήσεις

### 3.5.1 Απαιτήσεις διαθεσιμότητας λογισμικού

*Τεκμηρίωση απαιτήσεων διαθεσιμότητας*

Για να εξυπηρετηθούν ανάγκες διαθεσιμότητας του λογισμικού, απαιτείται ένας σέρβερ ο οποίος θα φιλοξενεί το λογισμικό και δεν θα αγνοεί τις απαιτήσεις μας ως προς τις επιδόσεις του συστήματος. Παράλληλα, δεδομένου ότι αρκετές φορές η επικοινωνία με την πλατφόρμα θα αποτυγχάνει, λόγω ανάγκης συντήρησης της, είναι επιτακτική ανάγκη να μειώνεται όσο το δυνατό περισσότερο τόσο το downtime όσο και η ανάγκη διαρκούς συντήρησης. Καταληκτικά, κομβικό σημείο για παροχή του λογισμικού, αποτελεί η πρόσβαση στο διαδίκτυο.

### 3.5.2 Απαιτήσεις ασφάλειας

*Τεκμηρίωση απαιτήσεων ασφαλείας*

Όπως δηλώθηκε προηγουμένως, βασικός στόχος της εφαρμογής μας είναι η εγγραφή του χρήστη σε αυτή. Αυτό γίνεται, μέσω της παροχής κάποιων προσωπικών στοιχείων (όνομα, mail, κωδικός). Ωστόσο, υπάρχει το ενδεχόμενο κάποιας κακόβουλης ενέργειας στο σύστημα, για υποκλοπή στοιχείων και παρακολούθησης των αλληλεπιδράσεων μεταξύ των χρηστών και του συστήματος(requests and responses).

Για αυτό το λόγο, αναπόδραστη απαίτηση για την τήρηση της ασφάλειας στην πλατφόρμα μας, είναι η κρυπτογράφηση των κωδικών (ίσως μέσω hash functions) στη βάση δεδομένων, καθώς και η χρήση HTTPS πρωτοκόλλου.

### 3.5.3 Απαιτήσεις συντήρησης

*Τεκμηρίωση απαιτήσεων συντήρησης*

Κάθε σύστημα που υπάρχει στο διαδίκτυο χαρακτηρίζεται από αναγκαιότητα συντήρησης του. Αυτό, συνήθως, περιλαμβάνει διόρθωση προβλημάτων (bugs), αναθεώρηση δομών επειδή είτε είναι outdated είτε δεν εξυπηρετούν τις νέες συνθήκες της αγοράς ή ενδεχομένως δεν είναι οι βέλτιστες λύσεις στα προβλήματα που επιλύουν. Όλα αυτά, εφαρμόζουν και στο δικό μας λογισμικό και αποτελούν απαιτήσεις για την ομαλή εξέλιξη του.

Βέβαια,γενικότερος στόχος αποτελεί η προσπάθεια επίτευξης των προαναφερθέντων με όσο το δυνατόν πιο βέλτιστη πρακτική. Πιο συγκεκριμένα, οποιαδήποτε επέμβαση επιδιωχθεί, θα πρέπει να σχεδιαστεί με μεγάλη ακρίβεια προηγουμένως, για να μην δημιουργηθεί άλλο μεγάλο σφάλμα στην εφαρμογή. Ταυτόχρονα, διατήρηση αντιγράφου (backup) προηγούμενης έκδοσης θα πρέπει να εξασφαλιστεί, ώστε στη χειρότερη περίπτωση να μπορούμε να επαναφέρουμε το σύστημα σε λειτουργική έκδοση.

Απο την άλλη πλευρά, εφόσον πρόκειται για online πλατφόρμα, τα updates, λόγω συντήρησης, θα πραγματοποιούνται αυτόματα (με κάποιο refresh της ιστοσελίδας), δίνοντας τη δυνατότητα στον χρήστη να λειτουργεί σε παλιά έκδοση μέχρι την εφαρμογή της αναβάθμισης.

(Καταληκτικά, πέρα από το software της εφαρμογής, είναι θέμα μείζονος σημασίας η συντήρηση του σέρβερ που φιλοξενεί η βάση δεδομένων μας. Προφανώς, η συντήρηση δεν περιορίζεται μόνο στη διατήρηση του σέρβερ σε κανονική λειτουργία, αλλά και στη βελτίωση του μέσω μεταβολής του κώδικα του με σκοπό την αναθεώρηση του τρόπου λειτουργίας ώστε να πραγματοποιείται καλύτερη διαχείριση της βάσης δεδομένων μας.) **αυτο μαλλον ειναι βλακεια**