Έγγραφο απαιτήσεων λογισμικού (SRS)

[ΤΙΤΛΟΣ ΕΦΑΡΜΟΓΗΣ ΛΟΓΙΣΜΙΚΟΥ]

# Εισαγωγή

## 1.1 Εισαγωγή: σκοπός του λογισμικού

Η εφαρμογή που αναπτύσσουμε αποτελεί ένα σύστημα διαχείρισης φόρτισης ηλεκτρικών οχημάτων μέσω Web Application. Η φόρτιση σε σχέση με την τροφοδότηση καυσίμων σε συμβατικά αυτοκίνητα, αποτελεί μια χρονοβόρα διαδικασία. Ταυτόχρονα η έλλειψη σταθμών φόρτισης καθιστά αναγκαία την ύπαρξη μιας καθολικής εφαρμογής, η οποία θα διευκολύνει τόσο την ίδια τη φόρτιση, όσο και την αναζήτηση σταθμών. Στόχος λοιπόν του συστήματος είναι η διευκόλυνση και απλούστευση της διαδικασίας της φόρτισης προσφέροντας στον οδηγό την δυνατότητα αναζήτησης σταθμών φόρτισης, παρακολούθησης προόδου, ηλεκτρονική πληρωμή κλπ.

## 1.2 Διεπαφές (interfaces)

### 1.2.1 Διεπαφές με εξωτερικά συστήματα

Το σύστημα μας χρησιμοποιεί τα εξής API’s τρίτων εφαρμογών:

* **OpenStreetMaps API**: Το συγκεκριμένο API μας δίνει την δυνατότητα να παρουσιάζουμε τους σταθμούς φόρτισης επάνω στον χάρτη, κάνοντας την αναζήτηση γρηγορότερο και πιο φιλική προς τον χρήστη.
* **Facebook/Google API**: Το συγκεκριμένο API δίνει την δυνατότητα στον χρήστη να κάνει   
  sign up με τον υπάρχων λογαριασμό του στο Facebook.
* **Speech Recognition API**: Δίνει την δυνατότητα στον χρήστη να εισάγει κείμενο χρησιμοποιώντας την φωνή του.
* **Σύστημα Πληρωμής**: Το συγκεκριμένο API αξιοποιείται στο ασφαλές σύστημα πληρωμής.

### 1.2.2 Διεπαφές με το χρήστη

Οι χρήστες διαχωρίζονται στις εξής κατηγορίες:

* **Εγγεγραμμένος χρήστης**: Ο εν λόγω χρήστης διαθέτει προσωπικό λογαριασμό και μπορεί να αξιοποιήσει όλες τις λειτουργίες της εφαρμογής, όπως πληρωμή.
* **Guest χρήστης**: Ο guest χρήστης μπορεί να χρησιμοποιήσει το κύριο μέρος της υπηρεσίας, πχ αναζήτηση σταθμών, αλλά για διάφορες άλλες λειτουργίες όπως πληρωμή ή αξιολόγηση σταθμού απαιτείται η δημιουργία λογαριασμού.
* **Διαχειριστής:** Ο διαχειριστής είναι ο αυτός που χειρίζεται το back-end σύστημα, έχει πρόσβαση στην βάση δεδομένων της εφαρμογής και την δυνατότητα να επεξεργάζεται, να εισάγει ή να διαγράφει στοιχεία από αυτήν.

# Αναφορές - πηγές πληροφοριών

Ν/Α

# Προδιαγραφές απαιτήσεων λογισμικού

## 3.1 Περιπτώσεις χρήσης

 Λεπτομερής προδιαγραφή των λειτουργιών του λογισμικού σε επίπεδο περιπτώσεων χρήσης. Ο αριθμός των περιπτώσεων χρήσης ανάλογα με τον αριθμό των μελών της ομάδας σύμφωνα με την εκφώνηση.

### 3.1.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: Φόρτιση

#### 3.1.1.1 Χρήστες (ρόλοι) που εμπλέκονται

Αναφορά στους ρόλους που αφορά η περίπτωση χρήσης

* Εγγεγραμμένοι χρήστες
* Ιδιοκτήτες σταθμών φόρτισης

#### 3.1.1.2 Προϋποθέσεις εκτέλεσης

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

Για να μπορέσει να πραγματοποιηθεί φόρτιση:

* Θα πρέπει αρχικά ένας χρήστης να διαθέτει προσωπικό λογαριασμό(εγγεγραμμένος χρήστης).
* Θα πρέπει να έχει επιλέξει τον κατάλληλο σταθμό φόρτισης (αυτόν στον οποίο βρίσκεται εκείνη τη στιγμή και είναι συμβατός με το όχημα του).
* Θα πρέπει να έχει συνδέσει την χρεωστική/πιστωτική του κάρτα με το σύστημα πληρωμών.

#### 3.1.1.3 Περιβάλλον εκτέλεσης

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

* Web app.
* CLI.
* Σελίδα διαχείρισης.
* Έναρξη φόρτισης
* Καταγραφή γεγονότος
* Συλλογή δεδομένων
* Υπολογισμός αναμενόμενου κόστους
* Υπολογισμός κόστους φόρτισης που ολοκληρώνεται

#### 3.1.1.4 Δεδομένα εισόδου

Καταγραφή δεδομένων εισόδου και εξόδου και συνθηκών εγκυρότητας αυτών.

* Προσωποποιητικό χρήστη.
* kWh που παρέχει ο σταθμός φόρτισης.
* Ιδιότητες αυτοκινήτου (πχ. Χωρητικότητα μπαταρίας, μοντέλο αυτοκινήτου, κλπ).

#### 3.1.1.5 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: Είσοδος στο σύστημα (Τα σφάλματα και οι απρόβλεπτες συμπεριφορές περιγράφονται στο αντίστοιχο use case)
* Βήμα 2: Επιλογή καταχωρημένου οχήματος.
* Βήμα 3: Έλεγχος διαθεσιμότητας σταθμού.
  + - Αν δεν υπάρχει, να υπολογίζει χρόνο αναμονής και άλλους κοντινούς σταθμούς
* Βήμα 4: Κατοχύρωση θέσης δια σύρματου φόρτισης.
* Βήμα 5: Έναρξη φόρτισης με τα προτεινόμενα προγράμματα χρέωσης και δέσμευση θέσης.
  + - Αν δεν πετύχει, να προβάλλει τις αιτίες σφάλματος και αναφορά χρήστη
* Βήμα 5: Καταγραφή γεγονότων και συλλογή δεδομένων.
* Βήμα 6: Υπολογισμός αναμενόμενου κόστους.
* Βήμα 7: Προβολή λογαριασμού και επιλογή τρόπου χρέωσης.
* Βήμα 8: Checkout.

#### 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)

* Εκτιμώμενος χρόνος και κόστος.
* Τελικό κόστος.
* Στατιστικά και διαγράμματα φόρτισης.
* Αναφορά σε πιθανά σφάλματα.
* Απελευθέρωση θέσης.

#### 3.1.1.8 Παρατηρήσεις

Δεν υπάρχουν παρατηρήσεις

### 3.1.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: Σύνδεση/Εγγραφή

#### 3.1.2.1 Χρήστες (ρόλοι) που εμπλέκονται

Αυτή η περίπτωση χρήσης, αφορά τον χρήστη εγγεγραμμένο ή μη, ο οποίος θέλει να συνδεθεί στον λογαριασμό του ή αν δεν έχει, να δημιουργήσει έναν ή να συνδεθεί ως guest.

#### 3.1.2.2 Προϋποθέσεις εκτέλεσης

Για να συνδεθεί ο πελάτης πρώτη φορά θα πρέπει να διαθέτει διεύθυνση ηλεκτρονικού ταχυδρομείου ώστε να μπορεί να δημιουργήσει έναν λογαριασμό και να τον επιβεβαιώσει. Από εκεί και πέρα για να συνδεθεί ο πελάτης του αρκούν τα στοιχεία username και password που έθεσε κατά την εγγραφή.

Στην περίπτωση που θέλει να συνδεθεί απλως σαν guest (περιοερισμένες λειτουργίες) δεν χρειάζονται τα παραπάνω.

#### 3.1.2.3 Περιβάλλον εκτέλεσης

Η σύνδεση και η εγγραφή του χρήστη γίνεται κατά την περιήγηση του σε συγκεκριμένη σελίδα

του web app και χρειάζεται η επικοινωνία με το back-end για να γίνει ο έλεγχος των στοιχείων ή η καταχώρηση τους στο σύστημα .

#### 3.1.2.4 Δεδομένα εισόδου

Τα δεδομένα εισόδου για την εγγραφή είναι

Όνομα, Επίθετο, Username, Password, email, Ηλικία, Φύλλο, Πληροφορίες Αυτοκινήτου/ων (μοντέλο, κυβισμός κ . λ .π).

Για την σύνδεση: Username, Password

#### 3.1.2.5 Παράμετροι

Κατά το register:

Username: Να μην συμπίπτει με ήδη υπάρχον.

Password: Πάνω από 8 χαρακτήρες.

Email: Υπάρχον email στο οποίο έχετε πρόσβαση.

Οι υπόλοιποι παράμετροι δεν έχουν περιορισμούς

Κατά το login:

Username: Υπάρχον username.

Password: Σε αντιστοιχία με το username.

#### 3.1.2.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Βήμα 1 : Μετάβαση στην αρχική σελίδα

Βήμα 2 : Επιλογή Log in/Sign up/ Log in as guest

Log in

Bήμα 3: Εισαγωγή username,password

Bήμα 4: Πάτημα submit

Βήμα 5: Έλεγχος των στοιχείων. Αν είναι σωστα , η σύνδεση είναι επιτυχής, αν είναι λάθος επιστροφή στο Βήμα 3 του Log in.

Alternate: Στην περίπτωση που ο χρήστης έχει ξεχάσει τον κωδικό του

Βήμα 3: Επιλογή αντίστοιχου κουμπιού

Βήμα 4: Συμπλήρωση e-mail και ξεκινάει διαδικασία ανάκτησης

Register

Βήμα 3: Εισαγωγή στοιχείων

Βήμα 4: Έλεγχος για σφάλμα(Υπαρξη ίδιου username, εισαγωγή μη έγκυρου χαρακτήρα)

Βήμα 5:Αν υπάρχει σφάλμα πήγαινε στο βήμα 3, αλλιώς επιτυχής εγγραφή.

Log In as Guest

Αρκεί το βήμα 1 και βήμα 2 (επιλογή κουμπιού Log in as Guest)

#### 3.1.2.7 Δεδομένα εξόδου

Κατά την διαδικασία Log in/Sign Up , παράγονται κάποια δεδομένα εξόδου. Τέτοια δεδομένα είναι τα μηνύματα σφαλμάτων τα οποία μπορούν να προκύψουν σε περιπτώσεις που αναφέρθηκαν παραπάνω αλλά και μήνυμα επιτυχούς εγγραφής σε περίπτωση σωστής καταχώρησης στοιχείων.

#### 3.1.2.8 Παρατηρήσεις

Δεν υπάρχουν

### 3.1.3 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 3: Πληρωμή

*3.1.3.1 Χρήστες (ρόλοι) που εμπλέκονται*

*Η περίπτωση της πληρωμής αφορά τον εγγεγραμμένο χρήστη, ο οποίος θέλει είτε να εξοφλήσει τον μηνιαίο λογαριασμό του, είτε να πληρώσει μία ολοκληρωμένη φόρτιση. Τράπεζα*

*3.1.3.2 Προϋποθέσεις εκτέλεσης*

*Για να εκτελέσει κάποια πληρωμή ο χρήστης απαιτείται αρχικά να είναι συνδεδεμένος στον προσωπικό του λογαριασμό, να έχει επιτυχώς εισάγει στο σύστημα τα στοιχεία της χρεωστικής ή πιστωτικής του κάρτας και το υπόλοιπο του τραπεζικού του λογαριασμού να είναι αρκετό ώστε να πραγματοποιηθεί η πληρωμή. Να έχει εκδοθεί λογαριασμός,*

*3.1.3.3 Περιβάλλον εκτέλεσης*

*Η πληρωμή εκτελείται στο Web App της εφαρμογής μέσω του ασφαλούς τραπεζικού περιβάλλοντος πληρωμής που διαθέτει το σύστημα. Αν ο χρήστης επιθυμεί να πιστώσει την πληρωμή στον μηνιαίο του λογαριασμό τότε θα μεταφερθεί στο αντίστοιχο περιβάλλον το οποίο έχει επικοινωνία με το back-end για να εισαχθεί η ανανεωμένη τιμή περιοδικού λογαριασμού.*

*3.1.3.4 Δεδομένα εισόδου*

*Τα δεδομένα εισόδου που αφορούν την πληρωμή είναι η επιλογή τρόπου πληρωμής, τα προσωπικά στοιχεία του χρήστη και το ποσό πληρωμής.*

*3.1.3.5 Παράμετροι*

*N/A*

*3.1.3.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά*

Basic flow

*Βήμα 1: Η περίπτωση χρήσης ξεκινάει με την σύνδεση του χρήστη στον προσωπικό του λογαριασμό*

1. *Αν προκύψει σφάλμα στην ταυτοποίηση ζητείται να επαναληφθεί η διαδικασία.*
2. *Αν ο χρήστης έχει ξεχάσει τα στοιχεία σύνδεσης του (username ή κωδικό) μεταφέρεται στο περιβάλλον ανάκτησης λογαριασμού.*
3. *Αν ο χρήστης δεν διαθέτει λογαριασμό τότε μεταφέρεται στο περιβάλλον δημιουργίας λογαριασμού*

*Βήμα 2: O χρήστης στο περιβάλλον των πληρωμών, εισάγει τα στοιχεία της χρεωστικής/πιστωτικής του κάρτας.*

1. *Αν προκύψει σφάλμα η διαδικασία επαναλαμβάνεται.*

*Βήμα 3: Ο χρήστης πατάει το κουμπί που αφορά τον μηνιαίο λογαριασμό.*

*Βήμα 4: Ο χρήστης βρίσκεται στο περιβάλλον επισκόπησης του μηνιαίου λογαριασμού και εφόσον υπάρχει ανεξόφλητος λογαριασμός (ή πολλοί) ο χρήστης διαλέγει τον λογαριασμό που τον ενδιαφέρει και αυτόματα μεταφέρεται στο περιβάλλον πληρωμής.*

*Βήμα 5: Ο χρήστης πατάει το κουμπί πληρωμής.*

1. *Αν η πληρωμή αποτύχει τότε εμφανίζεται το κατάλληλο μήνυμα σφάλματος.*

*Alternate Flow:*

* + - * 1. *Αν ο χρήστης έχει ήδη καταχωρίσει τα στοιχεία της κάρτας με την οποία θέλει να πληρώσει, τότε το βήμα 2 του basic flow παραλείπεται.*
        2. *Αν ο χρήστης επιθυμεί να πληρώσει μια φόρτιση μεμονωμένα και όχι μέσω του περιοδικού λογαριασμού, τότε τα βήματα 3, 4, 5 παραλείπεται και η πληρωμή γίνεται αυτόματα κατά τον τερματισμό της φόρτισης.*

*3.1.3.7 Δεδομένα εξόδου*

*Η διαδικασία της πληρωμής σχετίζεται με τα παρακάτω δεδομένα εξόδου.*

* *Διάφορα Μηνύματα (π.χ. Μήνυμα σφάλματος)*
* *Οφειλές περιοδικού λογαριασμού*
* *Αναλυτικός περιοδικός λογαριασμός*
* *Κόστος φόρτισης (κατά την μεμονωμένη πληρωμή)*

## 3.2 Απαιτήσεις επιδόσεων

Το σύστημά μας διαθέτει βάση δεδομένων η οποία περιέχει πληροφορίες για τους χρήστες, τα καταστήματα, τα προϊόντα και τις τιμές, καθώς και API και UI, ρόλος των οποίων είναι η παροχή πρόσβασης χρηστών, τρίτων εφαρμογών και γενικά εξωτερικών συστημάτων στις λειτουργίες του παρατηρητηρίου. Με βάση, λοιπόν, την αρχιτεκτονική αυτή, τα κύρια μεγέθη που αφορούν το λογισμικό μας είναι τα εξής δύο: Το μέγεθος των δεδομένων που είναι αποθηκευμένα στη βάση μας και αφορούν χρήστες, καταστήματα, προϊόντα και τιμές (το οποίο μέγεθος μετράται σε GB προς το παρόν και λογικά σε TB στο μέλλον) Το πλήθος των ενεργών συνδέσεων με το API (είτε από το δικό μας UI, είτε από τρίτες εφαρμογές) Η ταχύτητα ανταλλαγής δεδομένων μέσω αυτών των συνδέσεων (από και προς το API, από και προς τη βάση δεδομένων) Έχοντας, οπότε, υπόψη μας τα παραπάνω βασικά μεγέθη, κρίσιμες μετρικές της επιθυμητής απόδοσης του λογισμικού μας είναι οι ακόλουθες: Η αύξηση του μεγέθους των δεδομένων μας με το χρόνο (ή, καλύτερα, το ποσοστό αύξησης π.χ. ανά εβδομάδα, μήνα και έτος), η οποία μας βοηθά στη κατανόηση των τεχνικών αναγκών της εφαρμογής. Χρόνος απόκρισης σε ένα αίτημα χρήστη σε ms. Είναι πολύ σημαντικό ο συνολικός χρόνος επεξεργασίας, καθώς και το latency του δικτύου να μην επηρεάζουν αρνητικά την εμπειρία του χρήστη. Throughput εξυπηρέτησης αιτημάτων σε βάση δεδομένων και API. Ενδιαφέρει τόσο το πλήθος των αιτημάτων όσο και το μέγεθός τους (επιτρέπονται κλήσεις στο API που απαιτούν πολλή επεξεργασία και επιστρέφουν μεγάλο πλήθος δεδομένων). Η κλιμακωσιμότητα, θέλουμε δηλαδή η οποιαδήποτε αύξηση των δεδομένων λόγω της δραστηριότητας της εφαρμογής να αντισταθμίζεται γραμμικά από μία αντίστοιχη οριζόντια αύξηση των διαθέσιμων πόρων. Το πλήθος των σφαλμάτων που επιστρέφουν η βάση και το API, καθώς και το ποσοστό σφαλμάτων σε σχέση με το συνολικό πλήθος των αιτημάτων, με τη πάροδο του χρόνου. Περαιτέρω διάκριση των σφαλμάτων είναι επίσης σημαντική για τη συντήρηση αλλά και για τη προστασία του συστήματος από λογικά λάθη της εφαρμογής, λανθασμένη χρήση ή ακόμα και κακόβουλες επιθέσεις. Η διεκπεραιωτική ικανότητα της εφαρμογής, και, πιο συγκεκριμένα, τόσο το bandwidth του ίδιου του API, όσο και της βάσης, καθώς τα δύο αυτά δεν είναι άρρηκτα συνδεδεμένα. Πιο συγκεκριμένα, για κάθε αίτημα στη βάση, σίγουρα προηγήθηκε ένα τουλάχιστον αίτημα στο API (αφού η βάση δεν είναι άμεσα προσβάσιμη), αλλά δεν είναι απαραίτητο ότι για κάθε αίτημα στο API θα γίνεται και ένα αντίστοιχο αίτημα στη βάση (λόγω περιπτώσεων όπως σφάλματα, caching κλπ.)

Το σύστημα μας θα διαθέτει μια βάση δεδομένων η οποία περιέχει πληροφορίες για τους χρήστες , τα αυτοκίνητα τους , τους λογαριασμούς τους σταθμούς φόρτισης και τις διαδικασίες φορτίσεων. Ακόμη διαθέτει UI και API ώστε ο χρήστης ή και κάποιο άλλο εξωτερικό σύστημα να έχει πρόσβαση στις λειτουργίες της εφαρμογής οι οποίες θα αντλούν πληροφορίες από την βάση ή θα ενημερώνουν αυτές.

Επομένως σημαντικό ρόλο θα έχουν το μέγεθος των δεδομένων της βάσης αλλά και η ταχύτητα και το πλήθος των συνδέσεων. Επομένως καίριο ρόλο στην επίδοση του συστήματος μας θα έχουν:

* Η ταχύτητα απόκρισης στα αιτήματα , η οποία θέλουμε να είναι τέτοια ώστε ο χρήστης να μένει ικανοποιημένος και να μην υπάρχουν σημαντικές καθυστερήσεις.
* Ο ρυθμός αύξησης των δεδομένων , καθώς με την πάροδο του χρόνου , θα αυξάνονται τα δεδομένα της βάσης.
* Το πλήθος και το μέγεθος των αιτημάτων που μπορούν να απαντηθούν χωρίς να υπάρχει πρόβλημα.
* Η εμφάνιση σφαλμάτων (bugs) σε αιτήματα. Αυτά θα πρέπει να αναλύονται και να εξειδικεύονται ανάλογα με τον λόγο που προέκυψαν και να αντιμετωπίζονται έγκυρα και για την καλύτερη εμπειρία του χρήστη αλλά και για την ομαλή λειτουργία και ασφάλεια του συστήματος.

## 3.3 Απαιτήσεις οργάνωσης δεδομένων

### 3.3.1 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

Απαιτήσεις πρόσβασης και περιορισμοί.

## 3.4 Περιορισμοί σχεδίασης

Ανάλογα με ποιο τεχνολογικό πρότυπο που θα χρησιμοποιηθεί κατά την ανάπτυξη της εφαρμογής θα προκύψουν διάφοροι περιορισμοί σχεδίασης ώστε να πληρούνται οι προδιαγραφές που έχουν δοθεί. Για την συγγραφή, τον συντονισμό της ομάδας και την σχεδίαση της εφαρμογής θα χρησιμοποιούν τα κατάλληλα πρότυπα. Για το back-end θα χρησιμοποιηθεί Python, Django, MySQL και Django REST Framework. Για το Front-end HTML, CSS, React, Axios και JavaScript.

## 3.5 Λοιπές απαιτήσεις

### 3.5.1 Απαιτήσεις διαθεσιμότητας λογισμικού

Το λογισμικό απευθύνεται στην φόρτιση οχημάτων, είναι προφανές ότι η διαθεσιμότητα του κάθε στιγμή είναι αναγκαία. Ο κύριος λόγος που θα η εφαρμογή θα είναι μη διαθέσιμη είναι προγραμματισμένη συντήρηση λογισμικού ανά διαστήματα αλλά και άλλοι παράγοντες όπως σφάλμα του hosting server, bugs, συμφόρηση στο δίκτυο κ.α. Όπως φαίνεται από τα παραπάνω το downtime της εφαρμογής πρέπει να είναι όσο το δυνατό μικρότερο, οι συντηρήσεις να γίνονται γρήγορα και αποτελεσματικά και άλλοι παράγοντες να αντιμετωπίζονται στιγμιαία.

### 3.5.2 Απαιτήσεις ασφάλειας

Η ασφάλεια των προσωπικών δεδομένων των χρηστών, είναι προτεραιότητα στην ασφάλεια της εφαρμογής μας. Το Django framework παρέχει hashing για τους κωδικούς των χρηστών. Επίσης, θα υπάρχει encryption κατά την μεταφορά των δεδομένων.

Επίσης, ιδιαίτερη προσοχή δίνεται στο σύστημα πληρωμών. Δεν είναι δυνατόν να υποκλαπεί κάποια συναλλαγή εφόσον χρησιμοποιούμε συστήματα των ίδιων των τραπεζών.

Τέλος, θα υπάρχει κάποιο quota στο σύστημα των REST API , το οποίο θα εξασφαλίζει το όριο των API calls που μπορεί να κάνει ένας χρήστης ώστε να αποφύγουμε Injection attacks

### 3.5.3 Απαιτήσεις συντήρησης

* Κατά την επέκταση της εφαρμογής μας, θα πρέπει να συντηρείται η βάση δεδομένων καθώς μπορεί να προκύπτουν σφάλματα.