Έγγραφο απαιτήσεων λογισμικού (SRS)

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

[WattErloo]

# Εισαγωγή

## 1.1 Εισαγωγή: σκοπός του λογισμικού

*Το σύστημα που αναπτύσσουμε συνιστά μια πλατφόρμα στην οποία ο χρήστης μπορεί να λαμβάνει δεδομένα σχετικά με την κατανάλωση ηλεκτρικής ενέργειας στην Ευρώπη. Στόχος του συστήματος είναι η μεγάλη ανάγκη για την επισκόπηση και την κατανόηση της χρήσης της ηλεκτρικής ενέργειας, τόσο από οργανισμούς, όσο και από ιδιώτες. Με την αλλαγή των συνθηκών στον τομέα της παραγωγής ενέργειας και την εμφάνιση νέων μορφών ηλεκτρικής ενέργειας καθίσταται σημαντικό ο καθένας να γνωρίζει από που προέρχεται η παραγωγή για την περιφέρειά του. Την ανάγκη αυτή στοχεύει να καλύψει η εφαρμογή μας.*

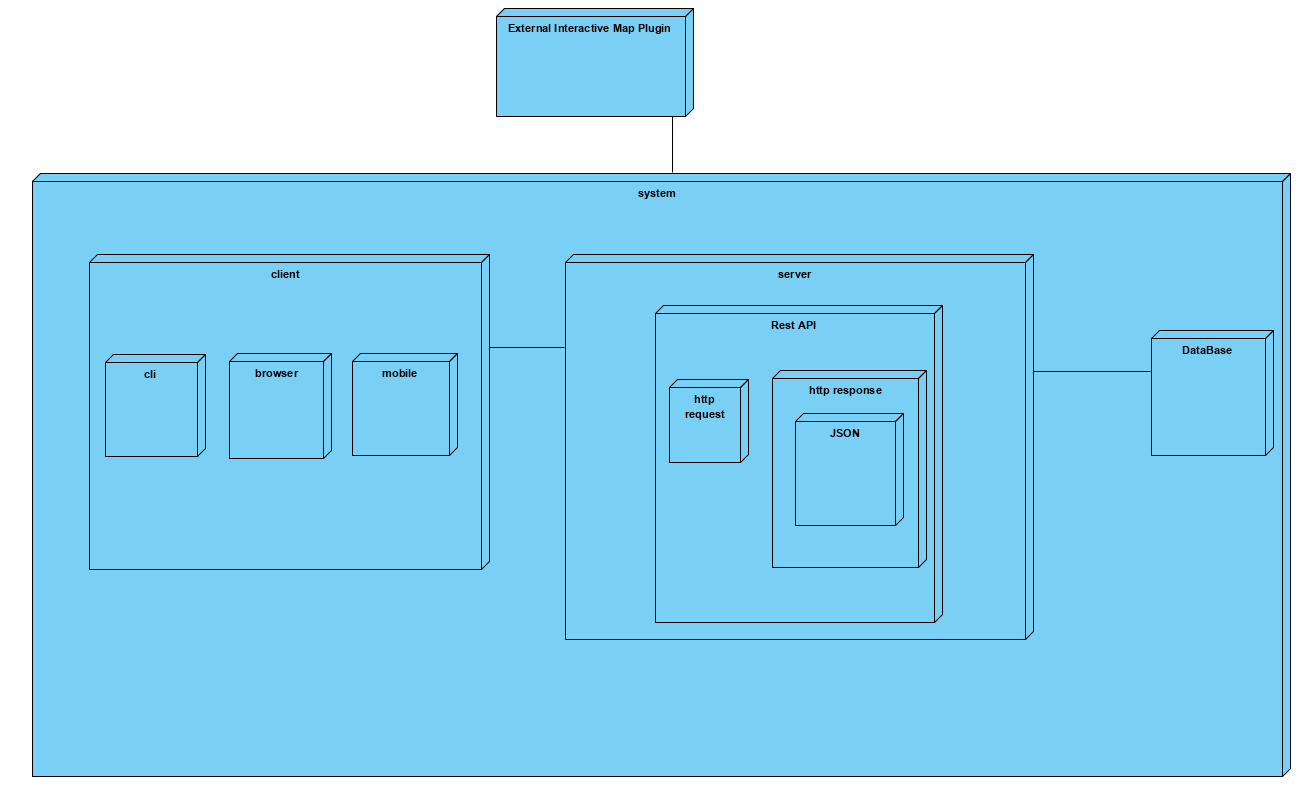
## 1.2 Διεπαφές

### 1.2.1 Διεπαφές με εξωτερικά συστήματα και εφαρμογές λογισμικού

*Ακολουθόντας σε μεγάλο βαθμό το μοντέλο MVC και ενσωματώνοντας την αρχιτεκτονική REST, μια επισκόπηση της δομής του λογισμικού μας φαίνεται από το Deployment Diagram της εικόνας 2. Οι εξωτερικές διεπαφές του συστήματος αφορούν:*

* *Τα άκρα της σύνδεσης στον browser με τον εξυπηρετητή που φιλοξενεί το API της εφαρμογής.*
* *Την επικοινωνία ανάμεσα στο User Interface και REST API για τη σωστή ενημέρωση του περιεχομένου που ζητείται από το χρήστη.*
* *Τις εξόδους του REST API προς τη βάση δεδομένων.*
* *Η επικοινωνία μέσω του CLI.*
* *Η επικοινωνία μέσω του mobile app.*

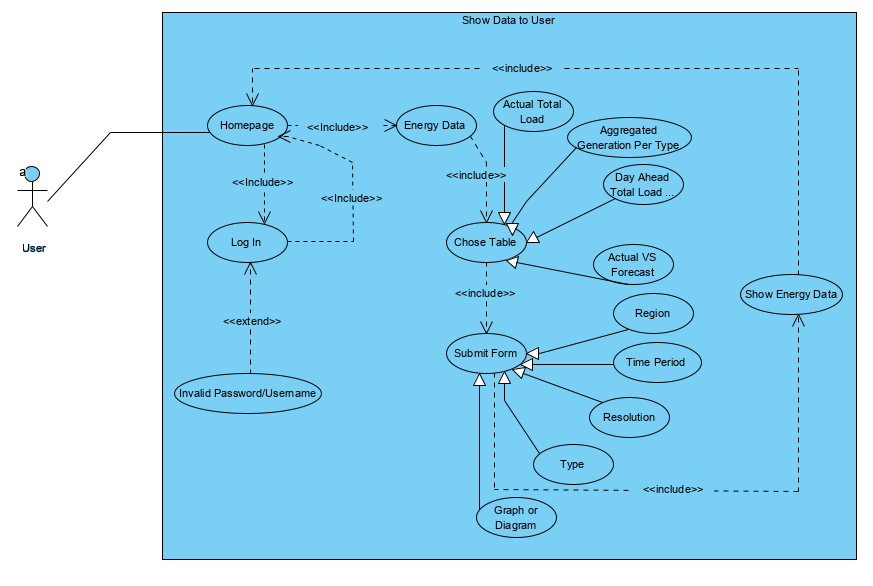
*Ο Client επικοινωνεί μέσω http requests με το API. Δέχεται απάντηση από το API http responses σε μορφή JSON με τις πληροφορίες που ζητήθηκαν. Το API από τη μεριά του είναι υπεύθυνο για την επικοινωνία με τη βάση δεδομένων μέσω SQL requests and responses. Ανάλογα με τα αιτήματα του χρήστη πραγματοποιεί την απαραίτητη επεξεργασία δεδομένων, ενημερώνει ή ζητά αποτελέσματα από τη βάση και, τέλος, επιστρέφει αποτελέσματα στον φυλλομετρητή του Client με μορφοποίηση JSON.*

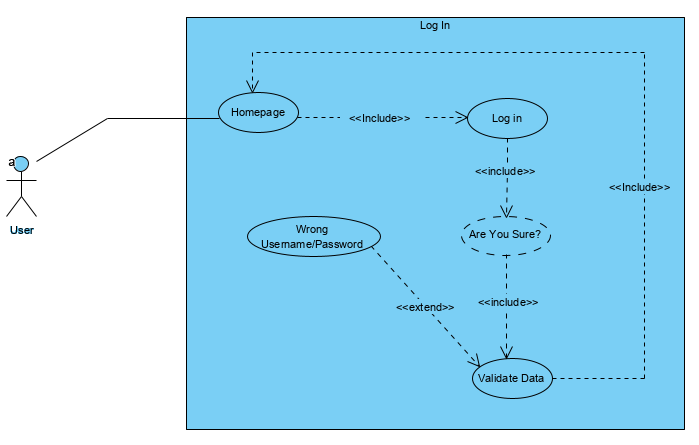
**

*Εικόνα 1. Deployment Diagram του Συστήματος*

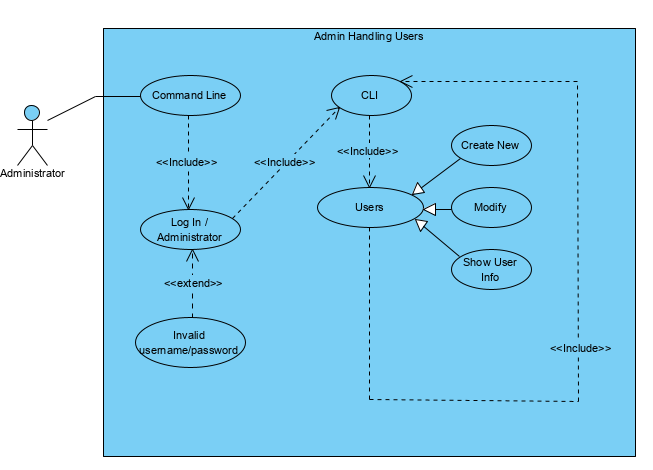
### 1.2.2 Διεπαφές με το χρήστη

*Στις επόμενες εικόνες περιγράφουμε τα σενάρια διεπαφής του χρήστη με την εφαρμογή.*

*Εικόνα 2. Εμφάνιση δεδομένων στο χρήστη από το Web App.*

**

*Εικόνα 3. Είσοδος χρήστη από το Web App.*

**

*Εικόνα 4. Επεξεργασία χρηστών από τον Administrator.*

# Προδιαγραφές απαιτήσεων λογισμικού

## 3.1 Εξωτερικές διεπαφές

*Στην προηγούμενη ενότητα 1.3.1 περιγράψετε τη δομή των διεπαφών της εφαρμογής μας. Σε αυτή την ενότητα θα περιγράψουμε την εμπειρία του χρήστη μέσα από τις γραφικές διεπαφές (Browser UI και Mobile UI) καθώς και τις μη γραφικές διεπαφές (CLI) με τις οποίες αλληλεπιδρά.*

*Web App*

*Αναφορικά με το web app UI, ο πρώτος κόμβος επίσκεψης κάθε χρήστη καθορίζεται δυναμικά με το εάν έχει συνδεθεί ή όχι. Αν ο χρήστης δεν είναι συνδεδεμένος, σε όποια σελίδα της εφαρμογής προσπαθήσει να αποκτήσει πρόσβαση θα γίνει redirect στη σελίδα σύνδεσης (Login Page). Αντίθετα, αν ο χρήστης είναι συνδεδεμένος ή συνδέεται, μεταφέρεται στην κεντρική σελίδα της εφαρμογής (Home Page). Αυτή είναι η κύρια σελίδα της εφαρμογής και αποτελείται από τα 3 βασικά μέρη : Header, Body, Footer. Στο header της σελίδας εμφανίζεται το logo της ομάδας, καθώς και links για μετάβαση στην αρχική σελίδα και για Log Out. Στο footer, έχουμε το όνομα της ομάδας μας.*

*Το κυρίως σώμα της εφαρμογής χωρίζεται σε 2 μέρη: Στα πεδία αναζήτησης που βρίσκονται στο αριστερό μέρος υπάρχουν τα πεδία αναζήτησης που επιτρέπουν στο χρήστη να επιλέξει τα δεδομένα που θέλει να δει : Χώρα, Ημερομηνία Από/Έως, Χρονικό Διάστημα, Είδος εμφάνισης (Πίνακας/ Γράφημα) και ένα κουμπί Submit, με το πάτημα του οποίου εμφανίζονται τα δεδομένα. Στο δεξί μέρος υπάρχει ένας χάρτης, πάνω στον οποίο ο χρήστης μπορεί να επιλέξει τη χώρα για την οποία θέλει τα αποτελέσματα(αντί της χρήσης του αντίστοιχου πεδίου.*

*Μετά την επιλογή αυτή, μεταφερόμαστε στη σελίδα Results, στην οποία εμφανίζονται τα αποτελέσματα με τη μορφή που τα ζήτησε ο χρήστης. Και πάλι στον header υπάρχει η επιλογή για ‘logout’ και για επιστροφή στο home page.*

*Mobile App*

*Ο χρήστης όταν ανοίγει την mobile εφαρμογή που έχουμε δημιουργήσει η πρώτη σελίδα που βλέπει είναι η σελίδα εισόδου του χρήστη στο σύστημα ( Log In Activity ). Για να μπορέσει στο κυρίως σώμα της εφαρμογής θα πρέπει να βάλει τα στοιχεία εισόδου του ( Username και Password ) και αν είναι σωστά και υπάρχουν στην βάση δεδομένων μας του επιτρέπουμε να κάνει access και να προχωρήσει στον κορμό του app ( Main Activity). Η κύρια σελίδα αποτελείται στο πάνω μέρος της από εισαγωγικές πληροφορίες προς τον χρήστη και από την συμπλήρωση των απαραίτητων πεδίων από την σκοπιά του προκυμμένου να μπορέσει να πραγματοποιηθεί σωστά η αναζήτηση που επιθυμεί και να εντοπίσει τα σωστά αποτελέσματα αναφορικά με την ενέργεια ( Πεδία συμπλήρωσης: Country, Date From, Date To, Form, Time ).*

*Αφού συμπληρωθούν όλα τα προαπαιτούμενα κενά πατάμε το «Search» και ανάλογα με την επιλογή που έχουμε κάνει στο πεδίο «Form» κατευθυνόμαστε στην σελίδα Table Activity ή Ravdo Activity που περιέχουν την ανάλυση των στοιχείων που αντλήσαμε από την βάση μας με τα δοθέντα στοιχεία του χρήστη σε μορφή Πίνακα στοιχείων ή Ραβδογράμματος αντίστοιχα. Σε κάθε μία από αυτές τις σελίδες υπάρχει η δυνατότητα από το πλαίσιο στο κάτω μέρος να μεταπηδήσουμε από την λειτουργία Table στην Ravdo και αντίστροφα καθώς επίσης και να πραγματοποιήσουμε μια καινούργια αναζήτηση με το πάτημα του πλήκτρου «New Search».*

*Σε κάθε μία από της παραπάνω σελίδες του app μας υπάρχει η δυνατότητα να κάνει ο χρήστης Log Out και να επιστρέψει στην αρχική σελίδα ( Log In Activity ) του συστήματος προκυμμένου να κάνει εκ νέου Log In με άλλα στοιχεία.*

3.2 Λειτουργίες: περιπτώσεις χρήσης

### 3.2.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: (Web log in)

#### 3.2.1.1 Χρήστες (ρόλοι) που εμπλέκονται

*Αυτή η χρήση αφορά τον χρήστη που θέλει να συνδεθεί μέσω του Web App ώστε να καταναλωσει τα δεδομένα του API.*

#### 3.2.1.2 Προϋποθέσεις εκτέλεσης

*Για να μπορεί ο χρήστης να συνδεθεί, πρέπει να είναι ήδη εγγεγραμμένος στο σύστημα της βάσης δεδομένων. Η λειτουργία αυτή γίνεται από τον διαχειριστή.*

#### 3.2.1.3 Περιβάλλον εκτέλεσης

*Για τη συγκεκριμένη περίπτωση χρήσης, θα εξετάσουμε μόνο το περιβάλλον της εφαρμογής Web.*

#### 3.2.1.4 Δεδομένα εισόδου

*Τα δεδομένα εισόδου είναι τα ατομικά στοιχεία του εγγεγραμμένου χρήστη (username, password).*

#### 3.2.1.5 Παράμετροι

*Για την επιτυχή είσοδο του χρήστη απαιτείται σωστός συνδυασμός username και password.*

#### 3.2.1.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

*Βήμα 1: Είσοδος στο login page*

*Βήμα 2: Εισαγωγή username και password από τον χρήστη*

*Βήμα 3: Καταχώριση της φόρμας και επιβεβαίωση στο popup window*

*Βήμα 4: Αν είναι σωστά τα credentials, ο user συνδέεται και μεταφέρεται στο homepage, αλλιώς βήμα 4.*

*Στις επόμενες εικόνες διατίθεται διάγραμμα UML για το παραπάνω Use Case.*

#### 3.2.1.7 Δεδομένα εξόδου

*Δεν υπάρχει κάποιο δεδομένο εξόδου προς τον χρήστη, παρατηρεί πως η σύνδεση επιτεύχθηκε από την αλλαγή του Navigation Bar, στο οποίο τώρα αναφέρεται η ένδειξη Logout.*

(UML 1ΟΥ USE CASE ΕΔΩ)

### 3.2.2 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: (Admin Handling Users)

#### 3.2.2.1 Χρήστες (ρόλοι) που εμπλέκονται

*Η συγκεκριμένη περίπτωση χρήσης αφορά τους Administrators.*

#### 3.2.2.2 Προϋποθέσεις εκτέλεσης

*O χρήστης πρέπει να έχει δικαιώματα admin και το σωστό password ώστε να μπορέσει να συνδεθεί.*

#### 3.2.2.3 Περιβάλλον εκτέλεσης

*Η επεξεργασία των χρηστών γίνεται μόνο μέσω του command line interface, και όχι μέσω web app ή mobile app.*

#### 3.2.2.4 Δεδομένα εισόδου

*Τα δεδομένα εισόδου είναι το username (admin) και ο κωδικός πρόσβασης password. Επιπλέον, απαιτείται η επιλογή του είδους της επεξεργασίας (Create New/ Modify/ Show User Info).*

#### 3.2.2.5 Παράμετροι

*Για την επιτυχή είσοδο του διαχειριστή απαιτείται ο σωστός κωδικός πρόσβασης από αυτόν.*

#### 3.2.2.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

*Βήμα 1. Είσοδος στο CLI.*

*Βήμα 2. Login ως admin και εισαγωγή του κωδικού πρόσβασης. Αν δοθεί λανθασμένος κωδικός, εμφανίζεται μήνυμα λάθους και επιστρέφουμε στο βήμα 2.*

*Βήμα 3. Επιλογή του τύπου επεξεργασίας. Create New/ Modify/ Show User Info. Για τις πρώτες δύο περιπτώσεις, ακολουθούμε στο βήμα 4, για την Τρίτη στο βήμα 5 .*

*Βήμα 4. Εισαγωγή δεδομένων προς επεξεργασία. Επιστροφή στο βήμα 3.*

*Βήμα 5. Εμφάνιση δεδομένων του χρήστη, επιστροφή στο βήμα 3.*

#### 3.2.2.7 Δεδομένα εξόδου

*Ως δεδομένα εξόδου στην περίπτωση Create New/ Modify έχουμε μόνο Confirmation πως οι αλλαγές έγιναν σωστά. Στην περίπτωση Show User Info, έχουμε ως έξοδο τα δεδομένα του χρήστη.*

*(UML SEQUENCE ΕΔΩ)*

### 3.2.3 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 2: (User Fetching Data)

#### 3.2.3.1 Χρήστες (ρόλοι) που εμπλέκονται

*Η συγκεκριμένη περίπτωση χρήσης αφορά τους Users που είναι εγγεγραμμένοι.*

#### 3.2.3.2 Προϋποθέσεις εκτέλεσης

*Ο χρήστης πρέπει να είναι εγγεγραμμένος και να διαθέτει το σωστό συνδυασμό username/ password.*

#### 3.2.3.3 Περιβάλλον εκτέλεσης

*Το περιβάλλον εκτέλεσης είναι η web εφαρμογή, αλλά ίδια λογική ακολουθείται και στη mobile εφαρμογή.*

#### 3.2.3.4 Δεδομένα εισόδου

*Τα δεδομένα εισόδου είναι το username και ο κωδικός πρόσβασης password. Επιπλέον, απαιτείται η επιλογή του είδους της επεξεργασίας επιλογή του τύπου των δεδομένων που ζητάει (Πρόβλεψη/ Τύπος ενέργειας/ Πραγματικό Φορτίο). Τέλος, πρέπει να προσδιοριστούν οι παράμετροι που αναφέρονται στο επόμενο ερώτημα.*

#### 3.2.3.5 Παράμετροι

*Οι παράμετροι είναι ο συνδυασμός username/password, ο οποίος πρέπει να είναι σωστός, ο τύπος δεδομένων που ζητείται όπως αναφέρθηκε στο 3.2.3.4, καθώς και οι παράμετροι Region, Time Period (From – Until), Resolution (διαστήματα χρόνου) και επιλογή Graph/ Table.*

#### 3.2.3.6 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

*Βήμα 1. Είσοδος στο login Page.*

*Βήμα 2. Login με εισαγωγή του ονόματος χρήστη και του κωδικού πρόσβασης. Αν δοθεί λανθασμένος κωδικός, εμφανίζεται μήνυμα λάθους και επιστρέφουμε στο βήμα 2.*

*Βήμα 3. Μετάβαση στο Homepage.*

*Βήμα 4. Επιλογή των παραμέτρων όπως προσδιορίζονται στην παράγραφο 3.2.3.5 και επιλογή Submit .*

*Βήμα 5. Αναμονή για απάντηση, αν το αίτημα είναι δεκτό, πήγαινε στο βήμα 6, αλλιώς πήγαινε στο βήμα 7.*

*Βήμα 6. Εμφάνιση των αποτελεσμάτων που ζητήθηκαν. Επιλογή να κάνει ο χρήστης reload, οπότε έχουμε επιστροφή στο βήμα 6, είτε να πατήσει back, οπότε έχουμε επιστροφή στο βήμα 3.*

*Βήμα 7. Εμφάνιση μηνύματος λάθους στον χρήστη και μετάβαση στο βήμα 3.*

#### 3.2.3.7 Δεδομένα εξόδου

*Ως δεδομένα εξόδου σε επιτυχή περίπτωση έχουμε τα ζητούμενα δεδομένα στην μορφή που ζητήθηκαν. Αλλιώς έχουμε ως έξοδο το κατάλληλο μήνυμα σφάλματος προς τον χρήστη.*

## 3.3 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)
* 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.

## 3.4 Απαιτήσεις οργάνωσης δεδομένων

### 3.4.1 Τεχνική περιγραφή των δεδομένων που διαχειρίζεται το λογισμικό και των σχετικών μετρικών φορτίου δεδομένων εισόδου, επεξεργασίας κ.λπ.

*Τα δεδομένα που διαχειρίζεται το λογισμικό μας είναι κατά κύριο λόγο (σχεδόν αποκλειστικά να εξαιρεθούν τοπικά αποθηκευμένα δεδομένα που αφορούν π χ τα credentials του χρήστη σε ένα session) τα δεδομένα που τελικά θα αποθηκευτούν στη βάση δεδομένων μας.*

*Τα δεδομένα της βάσης παρουσιάζονται στο E-R διάγραμμα που παρατέθηκε προηγουμένως. Συγκεκριμένα, τα δεδομένα μας αναφέρονται σε:*

* *Δεδομένα που αφορούν τους χρήστες (admin και εγγεγραμμένοι)*
* *Δεδομένα που αφορούν την πραγματική κατανάλωση ενέργειας*
* *Δεδομένα που αφορούν προβλέψεις κατανάλωσης της επόμενης ημέρας*
* *Δεδομένα που αφορούν την παραγωγή ενέργειας ανά είδος*

*Τα δεδομένα αυτά παρέχονται στο λογισμικό μας είτε από τον οργανισμό entso, είτε από τον admin, ο οποίος μπορεί να δημιουργεί νέους χρήστες και να επεξεργάζεται τα δεδομένα που αφορούν την ενέργεια.*

*Τέλος, κάποιες μετρικές που μας ενδιαφέρουν για τη διαδικασία του storage capacity planning είναι οι:*

* *Το μέγεθος των δεδομένων που αποθηκεύουμε στη βάση δεδομένων. Πρόκειται για την πιο σημαντική μετρική, η οποία είναι εύκολο και σημαντικό να καταγράφεται κατά την εξέλιξη της εφαρμογής, ώστε να έχουμε μια καλή εικόνα για τις απαιτήσεις του λογισμικού σε αποθηκευτικό χώρο.*
* *Το πλήθος των εσφαλμένων δεδομένων, το οποίο δύσκολα αναγνωρίζεται και είναι δυνατόν μόνο μέσω feedback από τους χρήστες.*
* *Το σύνολο των άκυρων/ελλιπών δεδομένων, δηλαδή αυτών που δεν διαθέτουν όλα τα απαιτούμενα πεδία που μας χρειάζονται. Αν και δεν περιμένουμε να έχουμε τέτοια δεδομένα από τον οργανισμό entso, λόγω της δυνατότητας επεξεργασίας τους μπορεί να δημιουργηθούν κενά.*

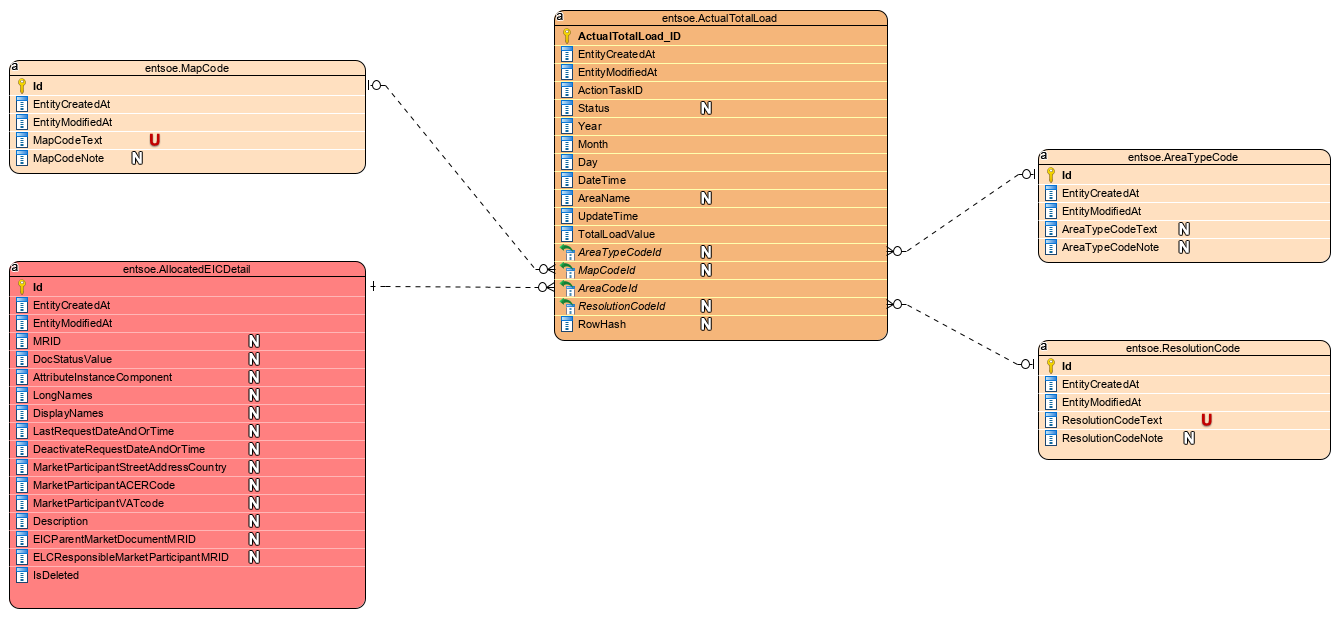
### 3.4.2 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

*Από την αρχή του έργου έχουμε αποφασίσει τη σχεδίαση και υλοποίηση ενός REST API. Επομένως, έχουμε ήδη προδιαγράψει πως κάποιος εξωτερικός χρήστης δεν θα έχει ποτέ άμεση πρόσβαση στα δεδομένα που διατηρεί η βάση δεδομένων και το λογισμικό. Η πρόσβαση σε αυτά γίνεται μόνο από τα γνωστά endpoints του API.*

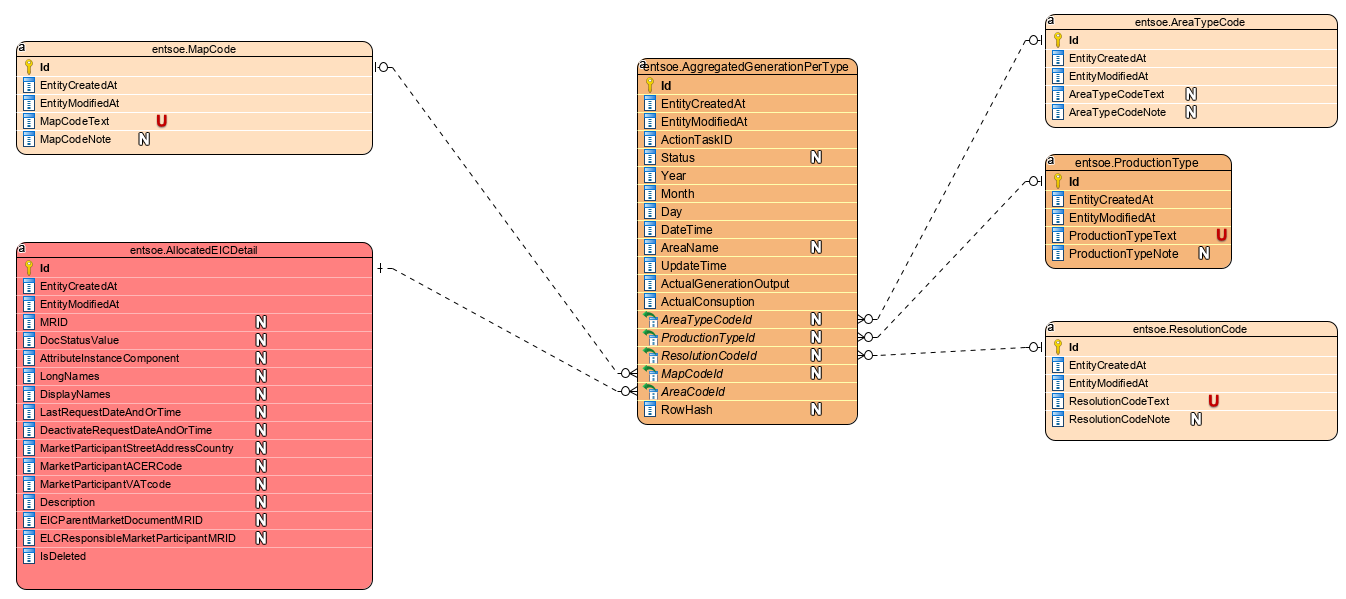
*Εκτός από αυτό, είναι απαραίτητο να μην υπάρχει πρόσβαση στα δεδομένα που αφορούν τον χρήστη από άλλους. Για το λόγο αυτό, κάθε χρήστης της εφαρμογής απαιτείται να έχει ένα username και ένα κωδικό πρόσβασης.*

### 3.4.3 Μοντέλο δεδομένων (μοντέλο κλάσεων UML ή/και μοντέλο ER)

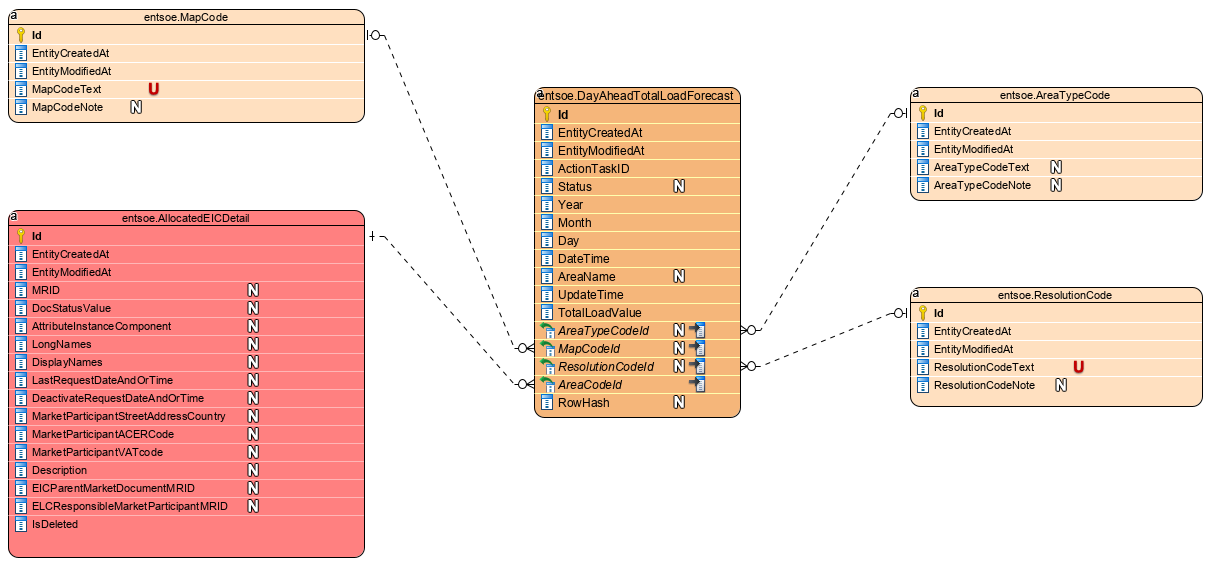
*Ακολουθούν τα μοντέλα δεδομένων που αφορούν των 3 είδους δεδομένων που παρέχει η εφαρμογή : ActualTotalLoad, DayAheadTotalLoadForecast και AggregateGeneratedByType.*

**

*Εικόνα . Διάγραμμα E-R του ActualTotalLoad*

******

*Εικόνα . Διάγραμμα E-R του AggregateGenerationPerType*

**

*Εικόνα . Διάγραμμα E-R του DayAheadTotalLoadForecast.*

### 3.4.4 Προδιαγραφές ακεραιότητας δεδομένων

*Η εφαρμογή μας έχει ως στόχο τη σωστή και έγκυρη ενημέρωση των χρηστών σε θέματα που αφορούν την παραγωγή ηλεκτρικής ενέργειας, τόσο την πραγματική, όσο και την πρόβλεψη για βάθος χρόνου. Με βάση το μοντέλο δεδομένων μας, το οποίο παρουσιάστηκε στην προηγούμενη ενότητα, οι βασικές ιδιότητες που πρέπει να υφίστανται ώστε να μπορούμε να ισχυριστούμε την ακεραιότητα και εγκυρότητα των δεδομένων μας είναι οι ακόλουθες:*

* *Οι πληροφορίες που διατηρούμε για την κατανάλωση/ παραγωγή/ ειδών ηλεκτρικής ενέργειας να είναι ορθές και αληθείς.*
* *Όσον αφορά την ακεραιότητα των δεδομένων, όπως είναι λογικό, απαιτούμε για κάθε μας τύπο οντότητας (όπως οι χρήστες) μοναδικά ονόματα και αναγνωριστικά, για την μονοσήμαντη αναγνώρισή τους, και γενικότερα, απαιτούμε να ισχύουν οι περιορισμοί εκείνοι τους οποίους έχουμε ορίσει κατά την κατασκευή της σχεσιακής βάσης μας.*

*Τα παραπάνω, για να επιτευχθούν, απαιτείται όσο το δυνατόν συνεχής επιτήρηση, ενώ βοηθάει και ο ρόλος των administrators, οι οποίοι μπορούν να ελέγχουν αν είτε υπάρχουσες είτε νέες καταχωρήσεις πληρούν τις προϋποθέσεις εγκυρότητας.*

### 3.4.5 Προδιαγραφές διατήρησης δεδομένων

*Εδώ, θα περιγράψουμε πλέον τις απαιτήσεις τις οποίες θα θέλαμε να πληροί το λογισμικό, ως προς τα δεδομένα του, σε μεγαλύτερο βάθος χρόνου. Πιο συγκεκριμένα, κάποιες ιδιότητες που θα θέλαμε να υφίστανται σε μεγάλο χρονικό ορίζοντα είναι οι εξής:*

* *Απαιτείται η σωστή διατήρηση των δεδομένων των χρηστών και γενικότερα τα δεδομένα της βάσης δεδομένων να μην χάνονται με το πέρασμα του χρόνου εάν δεν υπάρχει κάποιος λόγος να χαθούν (π.χ. διαγραφή χρηστών).*
* *Αποφυγή προσθήκης στη βάση διαφορετικών καταχωρήσεων από τον admin για τον ίδιο χρήστη, ή ακόμα και για τα ίδια τα δεδομένα που αφορούν την ενέργεια. Αυτό προφανώς αποφεύγεται με τη χρήση PK στην SQL, αλλά είναι γενικότερη απαίτηση που θέλουμε να ισχύει.*

*Τα παραπάνω, για να επιτευχθούν, απαιτείται όσο το δυνατόν συνεχής επιτήρηση, ενώ βοηθάει και ο ρόλος των administrators, οι οποίοι μπορούν να ελέγχουν αν είτε υπάρχουσες είτε νέες καταχωρήσεις πληρούν τις προϋποθέσεις εγκυρότητας.*

## 3.5 Περιορισμοί σχεδίασης

*Κατά την ανάπτυξη της εφαρμογής προέκυψαν περιορισμοί με βάση σχεδιαστικές επιλογές που λάβαμε κατά τη διάρκεια της σχεδίασης. Αυτές θα αναλύσουμε στην παράγραφο αυτή.*

1. *Αναφορικά με το backend της εφαρμογής, προέκυψαν διάφοροι περιορισμοί με τη σχεδιαστική επιλογή του να δημιουργήσουμε REST API για την εφαρμογή μας. Λόγω της επιλογής αυτής, όλα τα υπόλοιπα μέρη του front end, δεν επικοινωνούν καθόλου με τη βάση, παρά μόνο με το API, το οποίο στη συνέχεια επικοινωνεί με τη βάση. Επιπλέον, το backend είναι υποχρεωμένο να υλοποιεί όλα τα απαραίτητα endpoints, ώστε να δίνει "προς τα έξω" ένα ολοκληρωμένο interface για πλήρη και αποδοτική πρόσβαση στα δεδομένα της βάσης, το οποίο ταυτόχρονα να συμμορφώνεται και στο stateless πρωτόκολλο του REST.*

*Επιπλέον, η χρήση της SQL (και, πιο συγκεκριμένα, της MySQL) για την υλοποίηση της βάσης δεδομένων επιβάλλει και αυτή κάποιους περιορισμούς, οι βασικότεροι από τους οποίους έχουν να κάνουν με το ότι πρόκειται για μία σχεσιακή βάση δεδομένων. Σαν αποτέλεσμα, η βάση μας είναι λίγο πολύ υποχρεωμένη να περιέχει έναν αριθμό "προτύπων" (patterns - οι πίνακες), πολύ μικρό σε σχέση με το πλήθος των δεδομένων, και έτσι οι εγγραφές για χρήστες, για παράδειγμα, θα έχουν ένα σημαντικό βαθμό ομοιομορφίας. Γενικότερα, μια σχεσιακή βάση επιβάλλει κάποια σχετικά αυστηρή δομή στα δεδομένα της.*

1. *Αναφορικά με το front end, στο web app χρησιμοποιήθηκε το framework React JS με σκοπό την ανάπτυξη reactive single-page application. Προέκυψαν και εκεί περιορισμοί αναφορικά με το ιδιαίτερο συντακτικό μερικών σημείων του framework (για παράδειγμα το class tag της CSS μεταφράζεται σε className στο React). Όπως και σε κάθε έργο ανάπτυξης λογισμικού ακολουθήθηκαν συγκεκριμένες πολιτικές ονοματολογίες, όπως για παράδειγμα τα ονόματα των μεταβλητών να ακολουθούν camel notation, ενώ τα ονόματα των components να αρχίζουν πάντα με κεφαλαίο. Τέλος, λόγω της χρήσης του Create-React-App, χρησιμοποιήθηκαν αυτόματα frameworks όπως το babel και το webpack τα οποία είναι απαραίτητα για την υλοποίηση της εφαρμογής.*

## 3.6 Λοιπές απαιτήσεις

### 3.6.1 Απαιτήσεις διαθεσιμότητας λογισμικού

*Λόγω της φύσης του λογισμικού μας, και του σκοπού που αυτό εξυπηρετεί, είναι λογικό οι διάφοροι χρήστες μας να αναμένουν μεγάλη ή και πλήρη διαθεσιμότητα, ιδιαίτερα από το API, το οποίο είναι η "καρδιά" της εφαρμογής μας και κατά πάσα πιθανότητα θα χρησιμοποιείται και από τρίτες εφαρμογές, αλλά και από το UI μας σε κάποιο βαθμό, αφού αποτελεί ένα πλήρες Web App, με ό,τι αυτό συνεπάγεται για τις απαιτήσεις διαθεσιμότητάς του.*

*Μπορούμε να διακρίνουμε διάφορους λόγους που είναι δυνατόν να υπάρξει “downtime” της εφαρμογής μας. Το πρώτο είναι η εσκεμμένη από εμάς συντήρηση του λογισμικού. Η δεύτερη δεν εξαρτάται από εμάς, αλλά από αστάθμητους παράγοντες. Τέτοιο παράγοντες μπορεί να είναι σφάλμα του hosting server, συμφόρηση στο δίκτυο κλπ, οι οποίοι μπορούμε να θεωρήσουμε πως μπορούν να αντιμετωπιστούν σε σχετικά σύντομο χρονικό διάστημα.*

*Λόγω της ανεξαρτησίας μεταξύ του REST API και του user interface, όποια μορφή και να έχει αυτό, οι διαθεσιμότητές τους είναι επίσης ανεξάρτητες. Παρόλο δηλαδή που δε θα έχουμε τα απαραίτητα δεδομένα για να δείξουμε στους χρήστες, η εφαρμογή θα “λειτουργεί” αν δεν έχουμε πρόσβαση στο REST API, αλλά πολλές σελίδες θα έχουν ως έξοδο σφάλμα στο server.*

### 3.6.2 Απαιτήσεις ασφάλειας

*Με βάση το μοντέλο των δεδομένων μας (το οποίο έχει αναπτυχθεί σε άλλη υποενότητα), μπορούμε να διαπιστώσουμε ότι τα μόνα ευαίσθητα δεδομένα τα οποία διαχειρίζεται το λογισμικό μας είναι τα στοιχεία των χρηστών: username, password.*

*Στα δεδομένα αυτά, και στον κάθε λογαριασμό, θέλουμε να έχει πρόσβαση μόνο ο αντίστοιχος χρήστης, μέσω του μυστικού κωδικού (password) που μας έχει δώσει κατά τη διαδικασία εγγραφής του (sign up).*

*Για το σκοπό αυτό, έχουμε εφαρμόσει στο λογισμικό μας τις ακόλουθες, ευρύτερα γνωστές, τακτικές:*

*• Διατήρηση του hash κάθε κωδικού στη βάση μας, και όχι του ίδιου του κωδικού, ώστε ο κωδικός του χρήστη να είναι ασφαλής από επιθέσεις στη βάση.*

*• Επικοινωνία που περιλαμβάνει ανταλλαγή οποιωνδήποτε ευαίσθητων πληροφοριών / δεδομένων γίνεται μέσω ασφαλούς σύνδεσης (SSL). Στη φάση του development, αυτό γίνεται με χρήση self-signed certificate, αλλά στο μέλλον σε περιβάλλον production, στόχος είναι να γίνει μέσω CA signed certificate.*

### 3.6.3 Απαιτήσεις συντήρησης

*Αναφορικά με τη συντήρηση του λογισμικού, κινούμαστε γύρω από δύο βασικούς άξονες:*

1. *Η συνεχής επιτήρηση του λογισμικού, για τον εντοπισμό σφαλμάτων. Μπορεί μάλιστα να κριθεί απαραίτητη η διακοπή της λειτουργίας για συντήρηση του λογισμικού σε περιπτωση σοβαρού σφάλματος, το οποίο δεν είναι επιδιορθώσιμο όσο η εφαρμογή είναι “live”. Τέτοια σφάλματα μπορεί να αφορούν τόσο το backend, όσο και το front end.*
2. *Γενικές εργασίες συντήρησης πιθανώς χρειάζονται από τους developers/administrators για την εγκυρότητα των δεδομένων της εφαρμογής, όπως εξηγήθηκαν στο τμήμα 3.4.*