

# Μείωση κατανάλωσης ενέργειας ηλεκτροφωτισμού χώρων στάθμευσης

*Project του μαθήματος: Διαδίκτυο Των Πραγμάτων 2023-24*



## Ομάδα 01

ΒΛΑΣΣΗ ΣΤΥΛΙΑΝΗ 1072792

ΠΟΤΟΓΛΟΥ ΕΛΕΝΗ-ΙΩΑΝΝΑ 1072682

## ΠΕΡΙΕΧΟΜΕΝΑ

<b>1. ΠΕΡΙΓΡΑΦΗ.....</b>	<b>2</b>
1.1 ΠΕΡΙΛΗΨΗ.....	2
1.2 ΠΡΟΒΛΗΜΑΤΑ.....	2
1.3 ΛΥΣΗ.....	3
1.4 ΕΝΑΛΛΑΚΤΙΚΕΣ ΥΛΟΠΟΙΗΣΕΙΣ.....	3
1.5 ΚΑΙΝΟΤΟΜΙΕΣ.....	3
1.6 ΟΦΕΛΗ.....	4
1.7 ΕΞΟΠΛΙΣΜΟΣ.....	4
<b>2. ΠΡΟΣΕΓΓΙΣΗ.....</b>	<b>5</b>
2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ.....	5
2.1.1 Αρχιτεκτονική Συστήματος.....	5
Device Level.....	5
Edge Level.....	6
Cloud Level.....	6
2.1.2 Ροή Δεδομένων.....	6
Ενεργοποίηση Αισθητήρα Parking (MQTT).....	6
Ενεργοποίηση Φωτός (MQTT).....	7
Ενεργοποίηση Φωτός (HTTP).....	8
2.2 DATABASE.....	9
2.3 BOOKING APPLICATION.....	14
2.4 ΥΛΟΠΟΙΗΣΗ ΡΟΗΣ ΔΕΔΟΜΕΝΩΝ.....	17
1. Connect to MQTT Broker.....	18
2. Subscribe to MQTT Broker.....	18
3. Έλεγχοι.....	20
4. Άναμμα Φωτός.....	21
5. Publish to MQTT Broker.....	22
6. Patch Context Broker.....	23
2.5 TESTING.....	25
Fake Data.....	25
Real Data.....	26
2.6 GRAFANA.....	27
Parking Spots: Availability Map.....	27
Bookings.....	29
<b>3. ΕΡΓΑΛΕΙΑ.....</b>	<b>30</b>
Gantt Chart.....	30
Github.....	32
<b>ΑΝΑΦΟΡΕΣ.....</b>	<b>32</b>

## 1. ΠΕΡΙΓΡΑΦΗ

### 1.1 ΠΕΡΙΛΗΨΗ

Το σενάριο λειτουργίας υποχρεώνει ένα αυτοκίνητο να καλύψει μια προκαθορισμένη θέση στάθμευσης. Εάν φέρει το κατάλληλο bluetooth tag, τότε το φωτιστικό στοιχείο που αντιστοιχεί στη θέση πάρκινγκ, θα ενεργοποιηθεί μέσω ενός αντίστοιχου αισθητήρα ελέγχου για ορισμένο χρονικό διάστημα.

Η ομάδα καλείται να χρησιμοποιήσει υπάρχοντες τοποθετημένους αισθητήρες, καθώς και να τους μοντελοποιήσει για την παραγωγή εικονικών δεδομένων. Επιπλέον, στα πλαίσια της εργασίας θα υλοποιηθεί εφαρμογή web-app μέσω της οποίας ένας χρήστης θα έχει τη δυνατότητα να “κλείσει” τη θέση πάρκινγκ της επιλογής του.

Τέλος, αναμένεται η σχεδίαση συστήματος εποπτείας των θέσεων στάθμευσης και των φωτιστικών στοιχείων μέσω dashboards.

### 1.2 ΠΡΟΒΛΗΜΑΤΑ

Τα προβλήματα που επιθυμούμε να επιλύσουμε με το πρότζεκτ μας μπορούν να χωριστούν σε 2 κατηγορίες: σε προσωπικό επίπεδο, όσον αφορά τον εκάστοτε χρήστη, και σχετικά με την προστασία του περιβάλλοντος.

#### **Σε προσωπικό επίπεδο:**

- Η δυσκολία εύρεσης κενών θέσεων στάθμευσης:
  - Προκαλεί στρες και πίεση στους οδηγούς που δεν βρίσκουν κενό χώρο να παρκάρουν το αυτοκίνητό τους και αναγκάζονται να κάνουν κύκλους και να αργούν στον προορισμό τους
  - Αυξάνει τα λειτουργικά έξοδα του αυτοκινήτου (π.χ. κατανάλωση καυσίμων, φθορές στα λάστιχα κλπ)
- Όταν δεν υπάρχει φως στη θέση πάρκινγκ είναι πιο πιθανό ο οδηγός να μην υπολογίσει καλά τις αποστάσεις και να τρακάρει το αυτοκίνητό του στην προσπάθειά του να παρκάρει
- Η ασφάλεια των ανθρώπων απειλείται σε σκοτεινούς χώρους

*“One-third of all criminal acts take place in parking lots – including 22% of car thefts and 10% of all violent crimes – including assaults, kidnappings, and murders.”*

*-Increasing Public and Private Parking Lot Safety | MCA FAMILY OF COMPANIES*

## Προστασία του περιβάλλοντος

- Τα αυτοκίνητα που αναζητούν κενές θέσεις στάθμευσης προκαλούν κίνηση, ηχορύπανση και ατμοσφαιρική ρύπανση.
- Στην εναλλακτική που τα φώτα παραμένουν συνεχώς αναμμένα, έχουμε περιττή κατανάλωση ενέργειας και φωτορύπανση.

### 1.3 ΛΥΣΗ

Οι οδηγοί αυτοκινήτων, εξοπλισμένοι με το bluetooth tag του συστήματος, μπορούν να κλείσουν εκ των προτέρων μία από τις προκαθορισμένες θέσεις στάθμευσης μέσω της εφαρμογής του συστήματος. Όταν φτάσουν στο καθορισμένο σημείο, εάν το σύστημα αναγνωρίσει το bluetooth tag, τα φώτα πάνω και γύρω από το χώρο στάθμευσης θα ανάψουν για να επιβεβαιώσουν ότι είναι το σωστό μέρος αλλά και να αφήσουν τον οδηγό να παρκάρει με ασφάλεια. Στη συνέχεια, το φως θα παραμείνει αναμμένο για 5 λεπτά, δίνοντας χρόνο στον οδηγό να απομακρυνθεί με ασφάλεια.

### 1.4 ΕΝΑΛΛΑΚΤΙΚΕΣ ΥΛΟΠΟΙΗΣΕΙΣ

1. Τα φώτα να μένουν διαρκώς αναμμένα.

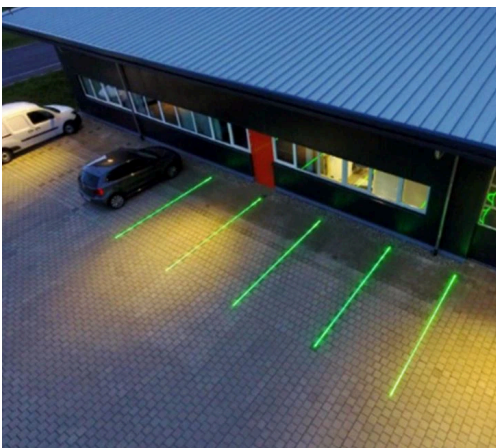
Η λύση απορρίφθηκε διότι προκαλεί περιττή κατανάλωση ενέργειας και φωτορύπανση.

2. Τα φώτα να ανάβουν ανεξάρτητα από το αν αναγνωρίζεται το σωστό bluetooth tag.

Η λύση απορρίφθηκε γιατί έτσι δεν θα ήταν ξεκάθαρο ότι έχει σταθμεύσει το σωστό αυτοκίνητο.

### 1.5 ΚΑΙΝΟΤΟΜΙΕΣ

- Οι οδηγοί μπορούν να κλείσουν τη δική τους προσωπική θέση στάθμευσης.
- Το σύστημα αναγνωρίζει το αυτοκίνητο και ανάβει τα φώτα όταν παρκάρει.
- Κάθε θέση στάθμευσης έχει φώτα led στο έδαφος γύρω του που ανάβουν μαζί με το φως πάνω από το πάρκινγκ.
- Όταν ο οδηγός εισέρχεται στην περιοχή πάρκινγκ, τα φώτα της θέσης που έχει δεσμεύσει ανάβουν, ώστε να αναγνωρίζεται από μακριά πού πρέπει να κατευθυνθεί.



Εικόνα 1.1 - Προσομοίωση LED Φωτών

Για το project μας, έπειτα από έρευνα και βάσει των διαθέσιμων αισθητήρων, αποφασίσαμε να εστιάσουμε στις 2 πρώτες καινοτομίες.

## 1.6 ΟΦΕΛΗ

### 1. Άνεση

Οι οδηγοί μπορούν να βρουν εύκολα θέση στάθμευσης και να πάνε κατευθείαν εκεί χωρίς να χάνουν χρόνο αναζητώντας κενές θέσεις στάθμευσης.

### 2. Ασφάλεια

Οι οδηγοί παρκάρουν με μικρότερο ρίσκο σύγκρουσης. Έπειτα, απομακρύνονται από τη θέση πάρκινγκ με ασφάλεια καθώς ο χώρος είναι φωτεινός.

### 3. Εξοικονόμηση Ενέργειας

Η εφαρμογή είναι φιλική προς το περιβάλλον. Από τη μία, μειώνει το λειτουργικό κόστος των αυτοκινήτων που κάνουν περιττές διαδρομές για αναζήτηση πάρκινγκ και ταυτόχρονα μειώνει τη ρύπανση που προκαλούν αυτές οι μετακινήσεις. Από την άλλη, μειώνει τη φωτορύπανση καθώς στα παρκινγκ, τα φώτα τείνουν να μένουν συνεχώς αναμμένα.

## 1.7 ΕΞΟΠΛΙΣΜΟΣ

Ο απαραίτητος εξοπλισμός για τη διεκπεραίωση του project είναι ο εξής:

- Bluetooth Tag
- Αισθητήρες:

- Cicicom S-LG3T (parking)
- inteliLIGHT FRE-220-NEMA-L (lightning)
- Φωτισμός:
  - PLS Serie της Energy Plus

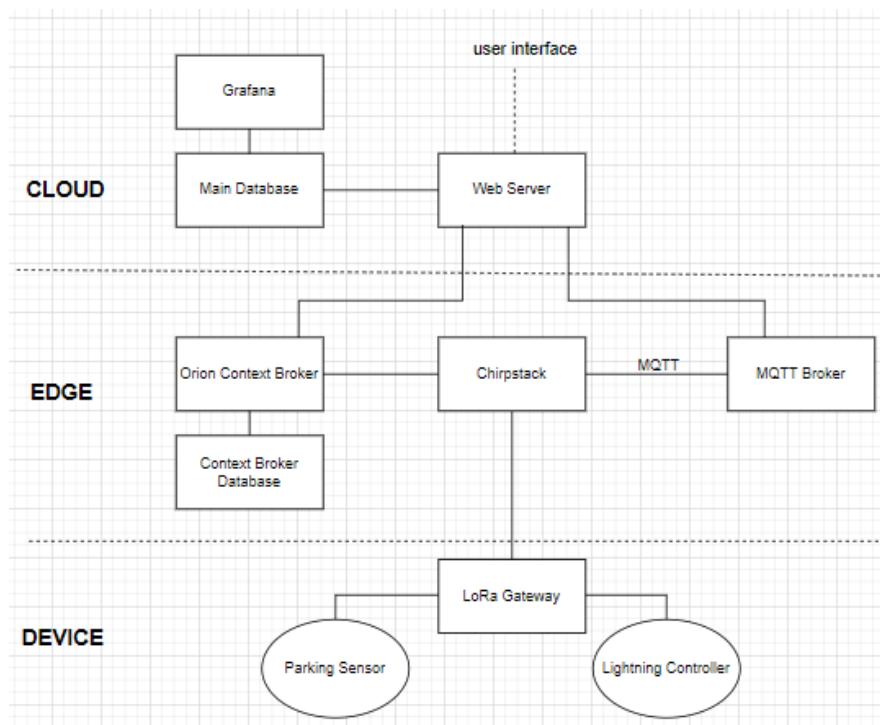
## 2. ΠΡΟΣΕΓΓΙΣΗ

Για τη διεκπεραίωση της εργασίας ακολουθήσαμε συγκεκριμένα βήματα ώστε να φτάσουμε στο τελικό αποτέλεσμα.

### 2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ

#### 2.1.1 Αρχιτεκτονική Συστήματος

Στην Εικόνα 2.1 φαίνονται τα βασικά στοιχεία που χρησιμοποιήθηκαν για την εφαρμογή μας, κατηγοριοποιημένα στα 3 επίπεδα του Internet Of Things (Device Level - Edge Level - Cloud Level).



Εικόνα 2.1.1 Αρχιτεκτονική Συστήματος

#### Device Level

- Parking Sensor: Ο αισθητήρας πάρκινγκ αναγνωρίζει όταν παρκάρει κάποιο αμάξι σε

μία συγκεκριμένη θέση parking.

- Lighting Controller: Ο ελεγκτής φωτός ενεργοποιεί το φωτιστικό στοιχείο, στο οποίο είναι συνδεδεμένος, όταν λάβει μία τέτοια εντολή.
- LoRa Gateway: Η πύλη LoRa λειτουργεί σαν διαμεσολαβητής μεταξύ των end devices (αισθητήρες, ενεργοποιητές) και του network server.

## Edge Level

- ChirpStack: Μία πλατφόρμα (network server) που μεταδίδει δεδομένα μεταξύ του LoRa Gateway και των υπόλοιπων συσκευών στις οποίες είναι συνδεδεμένη.
- MQTT Broker: Το σύστημα το οποίο μεταδίδει μηνύματα μεταξύ των συνδεδεμένων συσκευών χρησιμοποιώντας το MQTT πρωτόκολλο (publish-subscribe).
- Orion Context Broker: Είναι μία υλοποίηση NGSIv2 server της πλατφόρμας FIWARE που διαχειρίζεται δεδομένα (όμοια με τον MQTT Broker αλλά με το HTTP πρωτόκολλο)

## Cloud Level

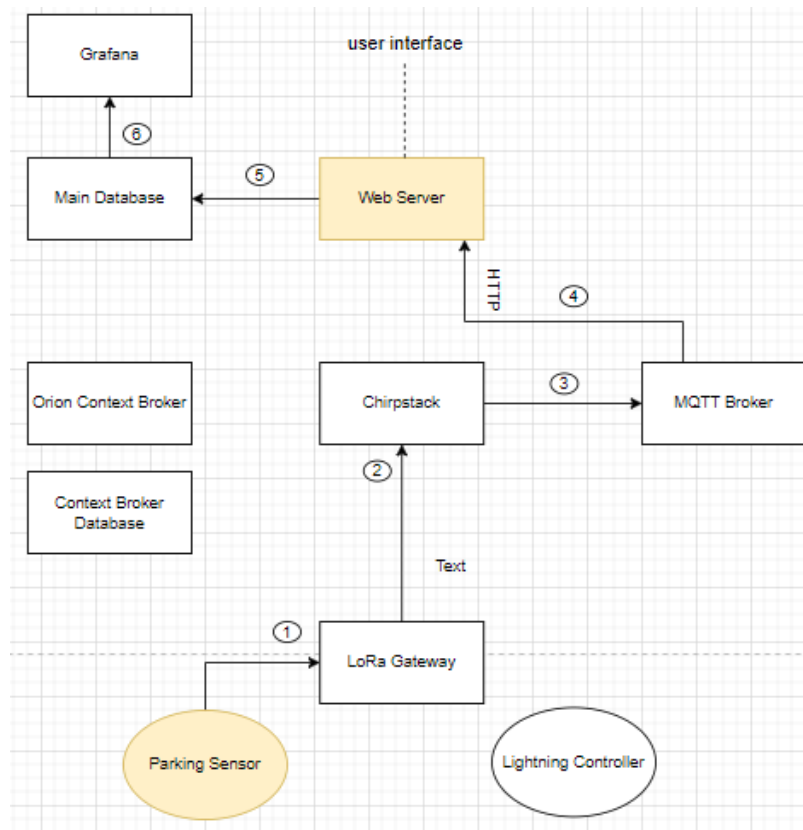
- Web Server: Διαχειρίζεται την εφαρμογή κρατήσεων καθώς και την επικοινωνία με τους αισθητήρες.
- Main Database: Αποθηκεύει τα δεδομένα των κρατήσεων, των αισθητήρων, και των θέσεων πάρκινγκ.
- Grafana: Είναι ένα dashboard στο οποίο κάνουμε visualize τα δεδομένα, στη μεριά του admin.

### 2.1.2 Ροή Δεδομένων

Κατά τη διάρκεια του project, ασχοληθήκαμε με 2 end devices. Παρακάτω φαίνονται 3 διαφορετικές ροές δεδομένων οι οποίες υπάγονται στην εργασία μας.

### Ενεργοποίηση Αισθητήρα Parking (MQTT)

Ο αισθητήρας πάρκινγκ, αφού πάρει σήμα ότι ένα αυτοκίνητο βρίσκεται εν κινήση στη θέση πάρκινγκ, στέλνει ένα σήμα ενεργοποίησης στο LoRa Gateway. Στη συνέχεια το gateway, συνδεδεμένο με το Chirpstack φτάνει τα δεδομένα του αισθητήρα στον MQTT Broker του πανεπιστημίου στο οποίο έχουμε subscribe. Από εκεί, τα δεδομένα περνάνε μέσω του Web Server στην Βασική Database της εργασίας η οποία είναι σε sqlite. Η database με τη σειρά της συνδέεται με το γραφικό περιβάλλον του Grafana.

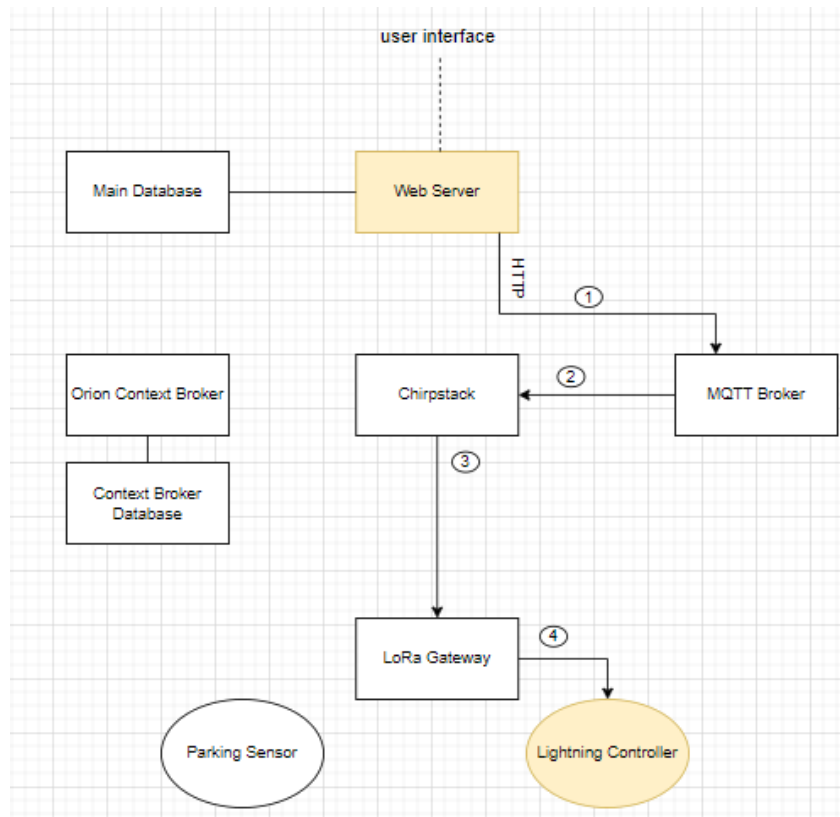


Εικόνα 2.1.2 : Αρχιτεκτονική της ενεργοποίησης του αισθητήρα parking

### Ενεργοποίηση Φωτός (MQTT)

Αφού μεταφερθεί μήνυμα στην βάση δεδομένων ότι έγινε ενεργοποίηση του αισθητήρα πάρκινγκ, ο Web Server προχωράει σε ορισμένες διαδικασίες ελέγχου ([Κεφάλαιο 2.4.3](#)). Αν έχει παρκάρει στη θέση το αυτοκίνητο που περιμένει η εφαρμογή, προχωράει στην ενεργοποίηση του φωτός, μέσω ενός μηνύματος publish στο σωστό topic του MQTT Broker του πανεπιστημίου. Από εκεί το φως δέχεται ένα σήμα να ανάψει και μετά από 5 λεπτά δέχεται ένα ακόμα σήμα με τον ίδιο τρόπο ώστε να κλείσει το φως.

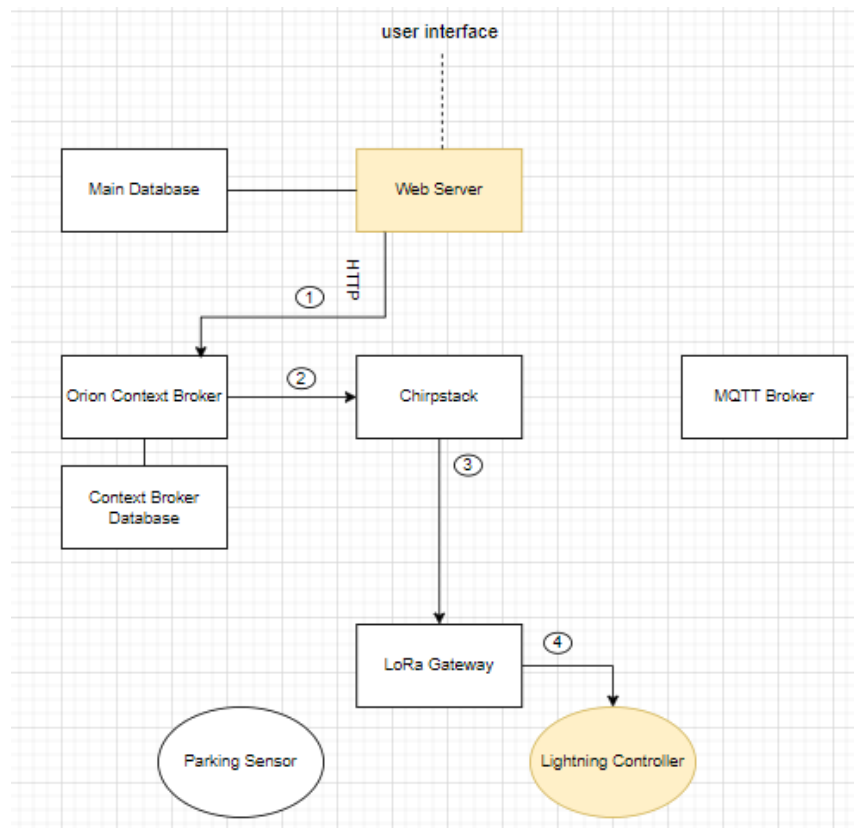




Εικόνα 2.1.3: Αρχιτεκτονική ενεργοποίησης του φωτός μέσω MQTT Broker

### Ενεργοποίηση Φωτός (HTTP)

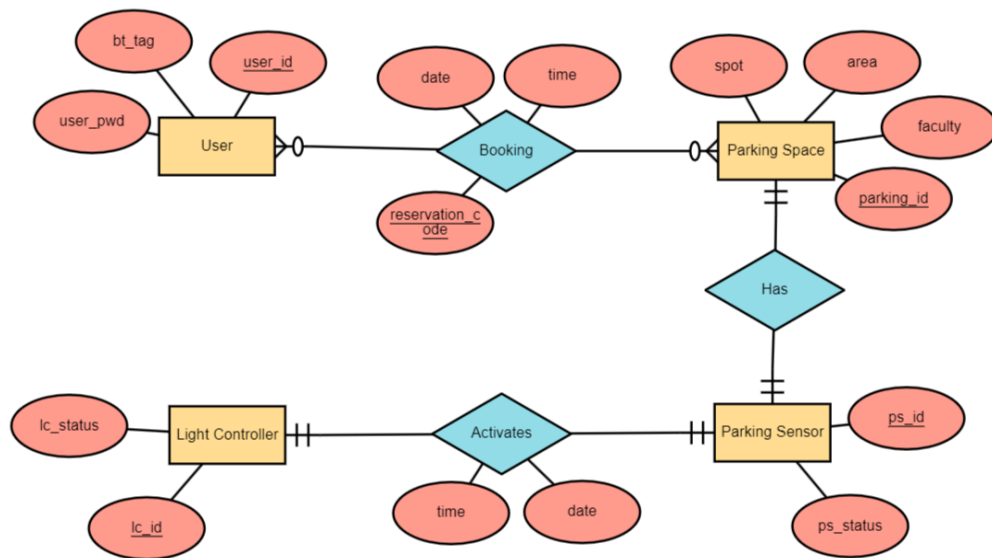
Αντίστοιχα με τη παραπάνω αρχιτεκτονική, στα πλαίσια της εργασίας αποφασίσαμε να κάνουμε υλοποίηση του light activation και μέσω του context broker. Έτσι όπως αναφέρθηκε παραπάνω, αφού γίνουν οι ελεγχοι, η πληροφορία της ενεργοποίησης του φωτός στέλνεται μέσω του Context Broker με PATCH, δηλαδή, πρωτόκολλο HTTP.



Εικόνα 2.1.4: Αρχιτεκτονική ενεργοποίησης του φωτός μέσω Context Broker

## 2.2 DATABASE

Σχεδιάσαμε ένα αρχικό ERD (Entity Relationship Diagram) και το αντίστοιχο Database Design που περιλαμβάνει τις απαραίτητες οντότητες, τα χαρακτηριστικά και τις συσχετίσεις μεταξύ τους που κρίθηκαν απαραίτητες προσθήκες στη βάση δεδομένων.



Εικόνα 2.2.1: Entity Relationship Diagram - Stage 1



Εικόνα 2.2.2: Database Design - Stage 1

Στη συνέχεια, προκειμένου να συμβαδίσουμε με τα [Smart Data Models](#), τροποποιήσαμε τα παραπάνω διαγράμματα ώστε να χρησιμοποιήσουμε τις κατάλληλες οντότητες και τις κατάλληλες ιδιότητες (όσες ήταν απαραίτητες βάσει κανονισμού και όσες κρίναμε απαραίτητες για τη συγκεκριμένη εφαρμογή). Συγκεκριμένα χρησιμοποιήθηκαν οι εξής:

### 1. OffStreetParking

Προσδιορίζει την περιοχή parking.

Σύμφωνα με το documentation:

A site, off street, intended to park vehicles, managed independently and with suitable and clearly marked access points (entrances and exits).

Properties:

- id[varchar]: Unique identifier of the entity
- location[varchar]: Geojson reference to the item
- type[string]: "OffStreetParking" by default
- name[string]: the name of this item
- images[array]: an URL containing a photo of this parking site

## 2. Entity: ParkingSpot

Προσδιορίζει την θέση parking.

Σύμφωνα με το documentation:

The aim of this entity type is to monitor the status of parking spots individually. Thus, an entity of type ParkingSpot cannot exist without a containing entity of type (OnStreetParking, OffStreetParking).

Properties:

- category[array]: Category(ies) of the parking spot. The parking spot belongs to an offStreet parking site
- id[varchar]: Unique identifier of the entity
- location[varchar]: Geojson reference to the item
- status[string]: Status of the parking spot from the point of view of occupancy.  
Enum: 'closed, free, occupied, unknown'
- type[string]: "ParkingSpot" by default
- refParkingSite[varchar]: Parking site to which the parking spot belongs to
- refDevice[array]: The device representing the physical sensor used to monitor this parking spot
- name[string]: The name of this item

## 3. Entity: Device

Προσδιορίζει τον αισθητήρα parking.

Σύμφωνα με το documentation:

A Device is a tangible object which contains some logic and is producer and/or consumer of data. A Device is always assumed to be capable of communicating electronically via a network.

Properties:

- id[varchar]: Unique identifier of the entity
- Type[string]: "Device" by default

- controlledProperty[array]: Anything that can be sensed, measured or controlled by. Enum:'airPollution, atmosphericPressure, averageVelocity, batteryLife, batterySupply, cdom, conductance, conductivity, depth, eatingActivity, electricityConsumption, energy, fillingLevel, freeChlorine, gasConsumption, gateOpening, heading, humidity, light, location, milking, motion, movementActivity, noiseLevel, occupancy, orp, pH, power, precipitation, pressure, refractiveIndex, salinity, smoke, soilMoisture, solarRadiation, speed, tds, temperature, trafficFlow, tss, turbidity, waterConsumption, waterFlow, waterLevel, waterPollution, weatherConditions, weight, windDirection, windSpeed' -> στην εφαρμογή μας είναι: "movementActivity"
- deviceCategory[array] -> στην εφαρμογή μας είναι "Sensor"
- value[string]: An observed or reported value -> στην εφαρμογή μας, αν αναγνωρίσει bluetooth tag παίρνει την τιμή του bluetooth tag, αλλιώς κενό string (" ")
- dateLastValueReported[date-time]: A timestamp which denotes the last time when the device successfully reported data to the cloud

#### 4. Entity: Streetlight

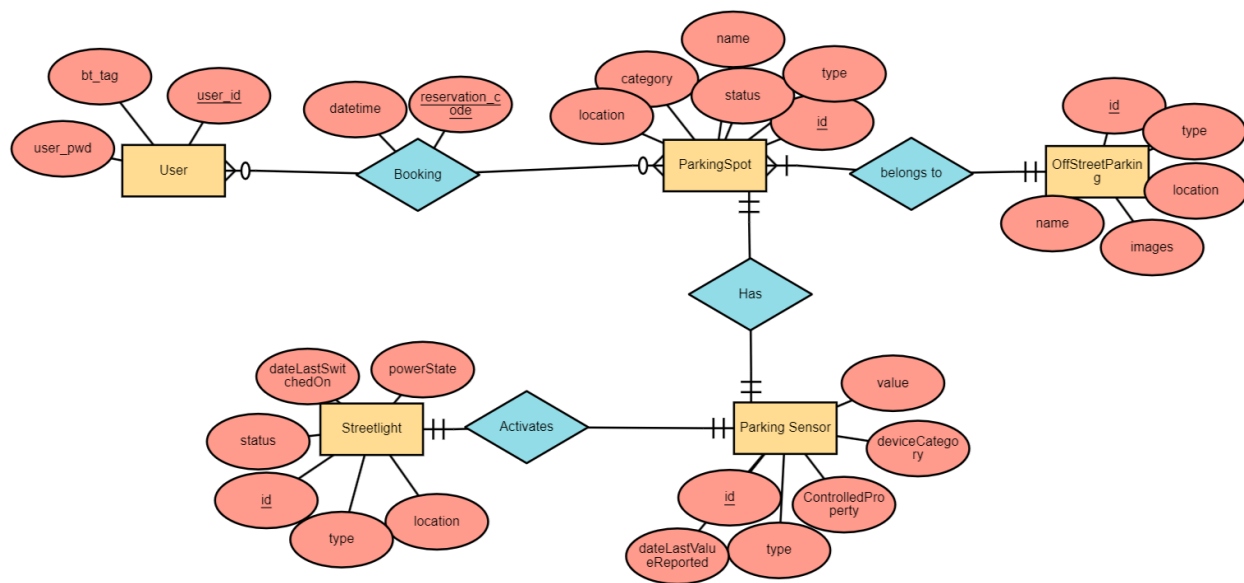
Προσδιορίζει το φωτιστικό στοιχείο.

Σύμφωνα με το documentation:

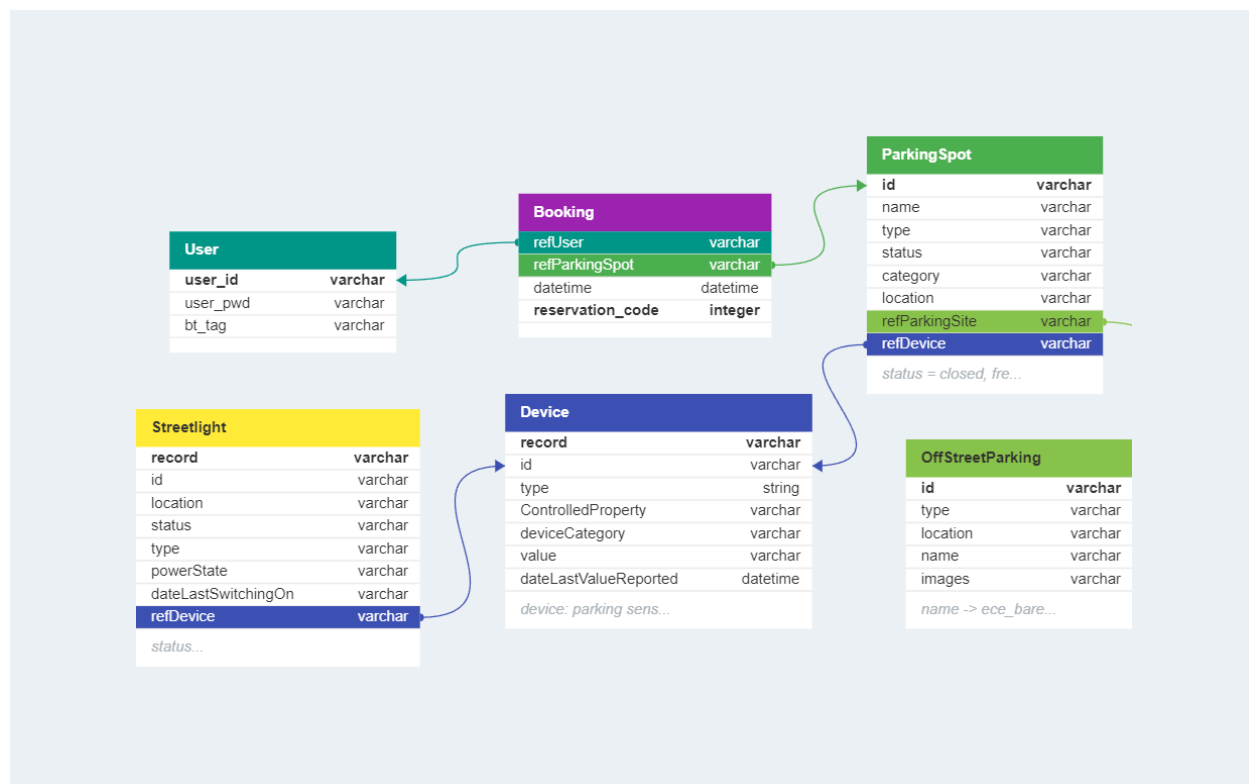
An entity of type Streetlight represents an urban streetlight.

Properties:

- id[varchar]: Unique identifier of the entity
- location[varchar]: Geojson reference to the item.
- status[string]: The overall status of this street light. Enum:'brokenLantern, columnIssue, defectiveLamp, ok'
- type[string]: NGSI Entity type. It has to be "Streetlight"
- powerState[string]: Streetlight's power state. Enum:'bootingUp, low, off, on'
- dateLastSwitchingOn[date-time]: Timestamp of the last switching on
- refDevice[array]: Reference to the device(s) used to monitor this streetlight.



Εικόνα 2.2.3: Entity Relationship Diagram - Stage 2 (SMDs)

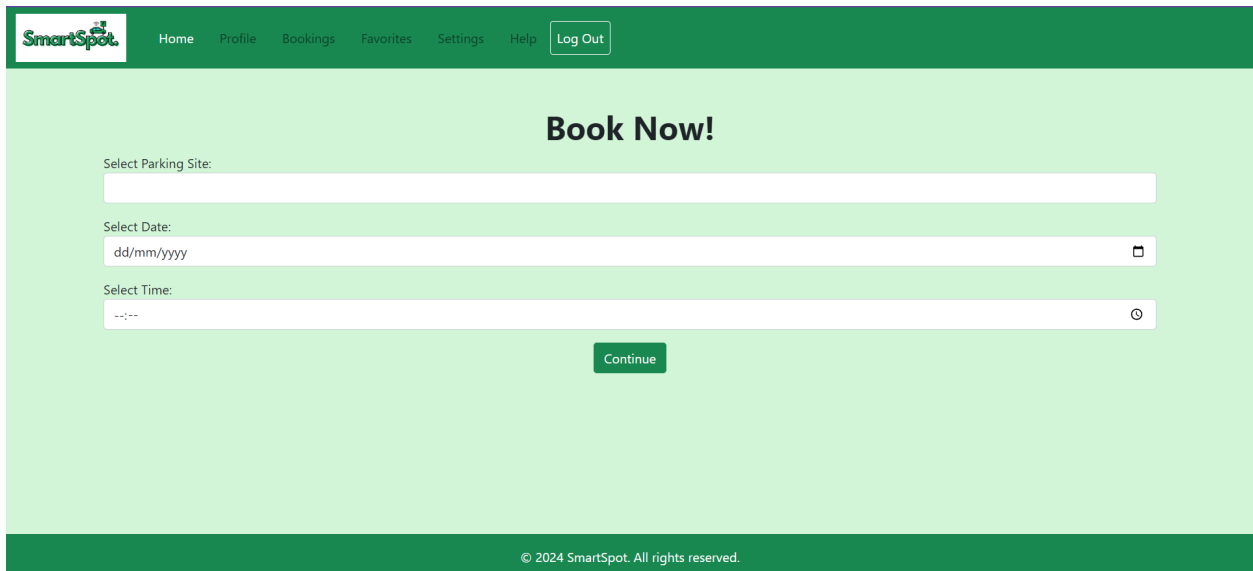


Εικόνα 2.2.4: Database Design - Stage 2 (SMDs)

Τα entities “User” και “Booking” παρέμειναν ίδια με την πρώτη υλοποίηση καθώς δεν συσχετίζονται με τα Smart Data Models.

## 2.3 BOOKING APPLICATION

Για το κλείσιμο των θέσεων πάρκινγκ από τους επισκέπτες του πανεπιστημίου, δημιουργήσαμε μια web application.

The screenshot shows the 'Book Now!' page of the SmartSpot application. At the top is a dark green navigation bar with the 'SmartSpot' logo and links for Home, Profile, Bookings, Favorites, Settings, Help, and a Log Out button. The main content area has a light green background. It features three input fields: 'Select Parking Site:', 'Select Date:' (with a date format 'dd/mm/yyyy' and a calendar icon), and 'Select Time:' (with a time format '--:--' and a clock icon). Below these fields is a green 'Continue' button. The footer is a dark green bar with the text '© 2024 SmartSpot. All rights reserved.'

Εικόνα 2.3.1: Αρχική σελίδα της web application

Η εφαρμογή είναι end-to-end δηλαδή συνδέεται μέσω router και controller στην database.

Παρακάτω φαίνεται η δομή της :

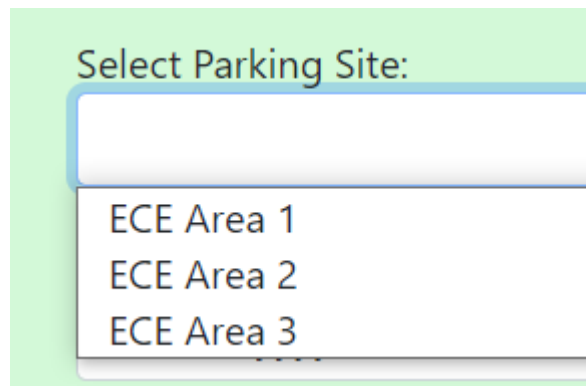
```
> .vscode
> controller
> model
> mqtt_broker
> public
> routes
> views
JS app.mjs
```

Εικόνα 2.3.2: Φάκελοι κώδικα του web application

Η ιστοσελίδα ενεργοποιείται μέσω nodejs και εμφανίζει στο <http://localhost:3001/home> την αρχική σελίδα.

Ο χρήστης έχει μετά την επιλογή να διαλέξει το Parking Site, Date και Time που θέλει να κλείσει μια θέση παρκινγκ.

Τα Parking Sites που εμφανίζονται προέρχονται από τη βάση και συγκεκριμένα είναι τα names από τον πίνακα **OffStreetParking**.

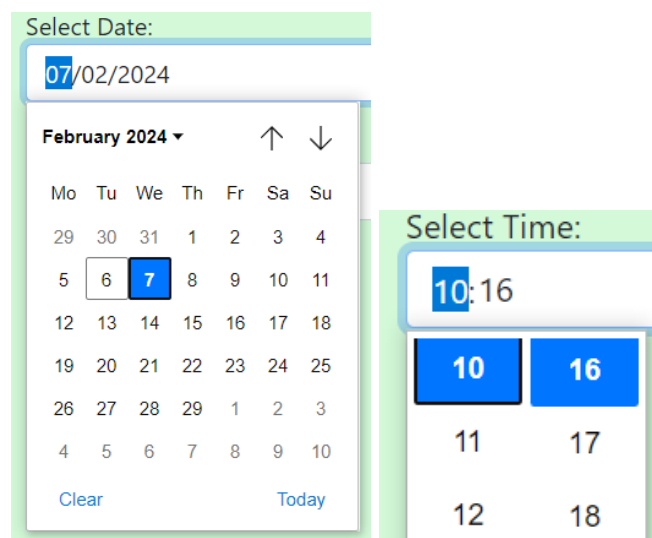


Select Parking Site:

- ECE Area 1
- ECE Area 2
- ECE Area 3

Εικόνα 2.3.3: Select Parking Site, επιλογές χρήστη

Το date και το time έχουν συγκεκριμένο format:



Select Date:

07/02/2024

February 2024

Mo	Tu	We	Th	Fr	Sa	Su
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	1	2	3
4	5	6	7	8	9	10

Clear Today

Select Time:

10:16

10	16
11	17
12	18

Εικόνα 2.3.4 Date and Time Selection

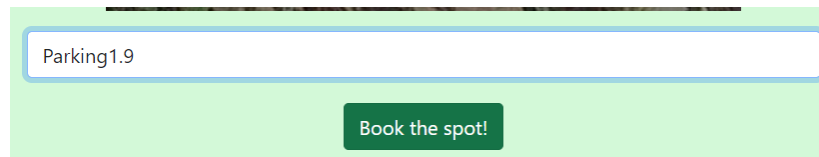
Αφού επιλέξει τα πεδία, ο χρήστης μεταφέρεται στην σελίδα του Parking Site. Εκεί μπορεί να επιλέξει τις θέσεις που είναι "free" για το διάστημα που ζητάει να παρκάρει.





Εικόνα 2.3.5: Σελίδα [http://localhost:3001/home\\_site](http://localhost:3001/home_site).

Σαν επικεφαλίδα φαίνεται το parking site που διάλεξε ο χρήστης και από κάτω η δορυφορική εικόνα του με την αρίθμηση των θέσεων. Στο drop-down menu εμφανίζονται μόνο οι θέσεις που είναι διαθέσιμες.



Εικόνα 2.3.6.: Επιλογή Parking Spot

Αφού ο χρήστης πατήσει το "Book the Spot", μεταφέρεται στην σελίδα του reservation confirmed όπου μπορεί να δει τα Booking Details της θέσης του.

**Your reservation is complete!**

**Booking Details**

User: *Stellini*

Datetime: *2024-02-07 at 10:16*

Parking Spot: *Parking1.9 in ECE Area 1*

Reservation Code: *85636205*

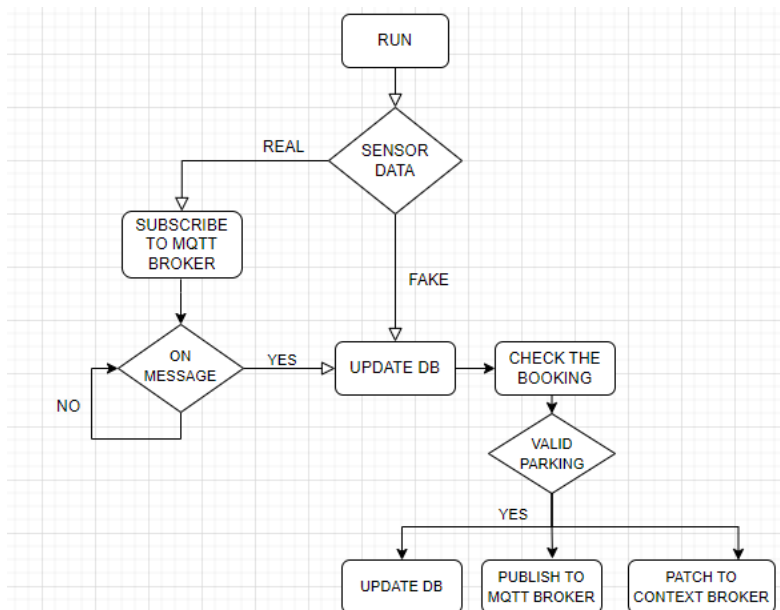
Εικόνα 2.3.7: Σελίδα <http://localhost:3001/reservation>

Ο χρήστης επιβεβαιώνει τα στοιχεία του Reservation του που έχει ήδη εγγραφεί στην βάση δεδομένων. Πλέον, το αντίστοιχο εικονικό Parking Sensor Device θα αναγνωρίσει το bluetooth tag του και αν τα δεδομένα αντιστοιχούν, θα ανοίξει το αντίστοιχο εικονικό Streetlight Device.

Από εκεί ο χρήστης έχει τη δυνατότητα να ξαναγυρίσει στην αρχική σελίδα.

## 2.4 ΥΛΟΠΟΙΗΣΗ ΡΟΗΣ ΔΕΔΟΜΕΝΩΝ

Για να υλοποιήσουμε τη ροή δεδομένων που φαίνεται στο κεφάλαιο 2.1.2 κάναμε τα παρακάτω βήματα, χρησιμοποιώντας τη γλώσσα προγραμματισμού python, στο αρχείο "mqtt\_code.py".



Εικόνα 2.4.1 Γενικό Διάγραμμα Ροής

## 1. Connect to MQTT Broker

Σε πρώτο στάδιο, συνδεθήκαμε με τον MQTT Broker του εργαστηρίου μέσω του σωστού IP address και port.

```
def connect_mqtt():
    def on_connect(client, userdata, flags, rc):
        if rc == 0:
            print("Connected to MQTT Broker!")
        else:
            print("Failed to connect, return code %d\n", rc)

    broker = '150.140.186.118'
    port = 1883
    client_id = "elianna"
    client = mqtt_client.Client(client_id)
    client.on_connect = on_connect
    client.connect(broker, port)
    return client
```

## 2. Subscribe to MQTT Broker

Έπειτα, κάναμε εγγραφή ("subscribe") στο σωστό topic του MQTT Broker ώστε να δεχόμαστε μηνύματα από τον αισθητήρα parking.

```
def subscribe(client):
    payload_dict = None

    def on_message(client, userdata, msg):
        nonlocal payload_dict
        # Decode bytes to string
        payload_str = msg.payload.decode('utf-8')
        payload_str = payload_str.replace("'", '"')
        try:
            # Parse JSON data
            payload_dict = json.loads(payload_str)
        except json.decoder.JSONDecodeError:
            print("json.decoder.JSONDecodeError")
            pass

    topic = "json/Parking/cicicom-s-lg3t:2"
    client.subscribe(topic)
    client.on_message = on_message

    while payload_dict is None:
        client.loop() # Process incoming messages
        time.sleep(0.1)

    return payload_dict
```

Το payload που λαμβάναμε είχε αυτή τη μορφή:

```

{
  "deduplicationId": "cb5abfd0-14f5-419d-bb54-fc1385d5f057",
  "time": "2024-02-02T16:24:40.086118906+00:00",
  "deviceInfo": {
    "tenantId": "063a0ecb-e8c2-4a13-975a-93d791e8d40c",
    "tenantName": "Smart Campus",
    "applicationId": "f3b95a1b-d510-4ff3-9d8c-455c59139e0b",
    "applicationName": "Parking",
    "deviceProfileId": "1f6e3708-6d76-4e0f-a5cb-30d27bc78158",
    "deviceProfileName": "Cicicom S-LG3T",
    "deviceName": "cicicom-s-lg3t:2",
    "devEui": "0004a30b00e95f14",
    "tags": {
      "model": "S_LG3T",
      "apiKey": "4jggokgpesnvfb2uv1s40d73ov",
      "manufacturer": "Cicicom",
      "deviceId": "cicicom-s-lg3t:1"
    }
  },
  "devAddr": "0133177d",
  "adr": true,
  "dr": 5,
  "fCnt": 981,
  "fPort": 1,
  "confirmed": true,
  "data": "NzMuMjcwAAAAEysxNy4w",
  "object": {
    "batteryVoltage": "3.27",
    "tag": "",
    "carStatus": 0.0,
    "temperature": "+17.0"
  },
  "rxInfo": [
    {
      "gatewayId": "1dee1a0843acf826",
      "uplinkId": 12699,
      "rssi": -87,
      "snr": -8.2,
      "channel": 5,
      "location": {
        "latitude": 38.288395556071336,
        "longitude": 21.788930292281066
      },
      "context": "aMIM7A==",
      "metadata": {
        "region_common_name": "EU868",
        "region_config_id": "eu868"
      },
      "crcStatus": "CRC_OK"
    }
  ],
  "txInfo": {
    "frequency": 867500000,
    "modulation": {
      "lorawan": {
        "bandwidth": 125000,
        "spreadingFactor": 7,
        "codeRate": "CR_4_5"
      }
    }
  }
}

```

Κάθε φορά που έρχεται ένα νέο μήνυμα από τον αισθητήρα, ανανεώνουμε τον πίνακα "Device" της database μας με τα παρακάτω δεδομένα:

- Object.tag: το bluetooth tag του αυτοκινήτου του πάρκαρε ("value")
- deviceInfo.devEui: το unique ID του συγκεκριμένου Device ("id")
- time: η ακριβής ημερομηνία και ώρα που ήρθε το τελευταίο μήνυμα

("dateLastValueReported")

```
#get variables from payload
bluetooth_tag=payload_dict.get('object', {}).get('tag')
device_ID= payload_dict["deviceInfo"]["devEui"]
datetime from payload = payload dict['time']

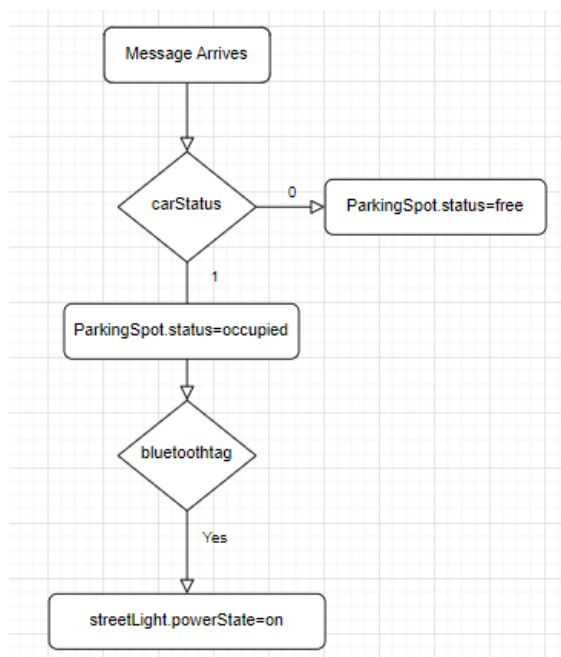
def save_payload_to_db():
    #Update Device table
    cursor.execute("UPDATE Device SET (dateLastValueReported,value)=(?,?) WHERE id=?", (datetime_from_payload,bluetooth_tag,device_ID,))
    conn.commit()
```

### 3. Έλεγχοι

Ο αισθητήρας parking στέλνει μηνύματα σε 3 περιπτώσεις:

- όταν παρκάρει κάποιο αμάξι
- όταν ξεπαρκάρει κάποιο αμάξι
- μετά από τακτά χρονικά διαστήματα (~4 ώρες) για να δείξει ότι λειτουργεί κανονικά

Η λειτουργία του συστήματος ανά περίπτωση φαίνεται συνοπτικά στο παρακάτω διάγραμμα ροής:



Εικόνα 2.4.2: Διάγραμμα Ροής Ελέγχου

### Αναλυτικά:

Κάθε φορά που λαμβάνουμε ένα payload, πρέπει πρώτα να ελέγξουμε αν έχει παρκάρει κάποιο αμάξι, χρησιμοποιώντας την πληροφορία "carStatus" από το payload. Στη συνέχεια, ανανεώνουμε στον πίνακα "ParkingSpot" το "status" ως εξής:

- Αν το Object.carStatus είναι 1, το status του parkingSpot γίνεται occupied
- Αν το Object.carStatus είναι 0, το status του parkingSpot γίνεται free

```
def check_if_car_parked():
    carStatus=payload_dict.get('object', {}).get('carStatus')
    if (carStatus==1):
        print("A car parked in the parking spot with ID: "+parking_spot)
        #Update ParkingSpot's status to occupied
        cursor.execute("UPDATE ParkingSpot SET status='occupied' WHERE refDevice=?", (device_ID,))
        conn.commit()
    elif (carStatus==0):
        print("There is no car parked in the parking spot with ID: "+parking_spot)
        #Update ParkingSpot's status to free
        cursor.execute("UPDATE ParkingSpot SET status='free' WHERE refDevice=?", (device_ID,))
        conn.commit()
    else:
        print("There has been an error in loading the car status")
    return carStatus
```

Στη συνέχεια, είναι απαραίτητο να γίνει ένας έλεγχος για το αν πάρκαρα ο σωστός χρήστης, στη σωστή θέση πάρκινγκ, στη σωστή ημερομηνία και ώρα, βάσει των κρατήσεων της εφαρμογής. Συγκεκριμένα για την ώρα, επιτρέπεται μία απόκλιση 10 λεπτών από την ώρα που έγινε η κράτηση.

```
def check_if_parking_valid():
    on_time=False
    #Get times that match to the user,date and parking spot
    cursor.execute("SELECT time FROM Booking WHERE (refUser=? and date=? and refParkingSpot=?)", (user_ID,date_from_payload,parking_spot))
    times_that_match_list=cursor.fetchall()
    if (times_that_match_list!=[]):
        times_that_match = [item[0] for item in times_that_match_list]
        # Check if the driver parked on time (payload's time = booking's time +- 10 mins)
        time_from_payload_formatted = datetime.strptime(time_from_payload, "%H:%M")
        ten_minutes = timedelta(minutes=10)
        for time_that_match in times_that_match:
            time_formatted = datetime.strptime(time_that_match, "%H:%M")
            if (time_from_payload_formatted - ten_minutes <= time_formatted <= time_from_payload_formatted + ten_minutes):
                on_time=True
                break
        else:
            on_time=False
    return on_time
```

## 4. Αναμμα Φωτός

Αν τα στοιχεία είναι σωστά, τότε πρέπει το φως να ανάψει και μετά από 5 λεπτά να σβήσει. Για να γίνει αυτό, ανανεώνουμε τη τιμή του powerState στο Streetlight της βάσης δεδομένων μας σε "on" και "off" αντίστοιχα.

Για να ανάψουμε/σβήσουμε το φως έχουμε 2 επιλογές:

- να κάνουμε publish στον MQTT Broker με το σωστό topic
- να κάνουμε patch στο Context Broker με το σωστό URL

Στα πλαίσια αυτής της εργασίας, δοκιμάσαμε και τις 2 διαδρομές. Φυσικά σε πραγματικά πλαίσια αρκούσε η μία.

```
try:
    save_payload_to_db()
    parking_spot=get_parkingspot_id()
    if (parking_spot):
        carStatus = check_if_car_parked()
        if (carStatus):
            user_ID=get_user_id()
            if(user_ID):
                on_time=check_if_parking_valid()
                if (on_time):
                    print("The car matches the one from the booking. The light is ON for 5 minutes")
                    print()
                    open_the_light()
            if (not(user_ID) or not(on_time)):
                print("There's no parking match based on the app. The light stays OFF")
                print()

def open_the_light():
    #Get Streetlight's ID
    cursor.execute("SELECT * FROM Streetlight WHERE refDevice==?", (device_ID,))
    streetlight_list=cursor.fetchall()
    if (streetlight_list==[]):
        print("This Device does not refer to any Streetlight.")
        print()
    else:
        streetlight_info=streetlight_list[0]
        #Set Streetlight's powerState to 'on'
        cursor.execute("UPDATE Streetlight SET powerState='on',dateLastSwitchingOn=? WHERE id=?", (datetime_from_payload,streetlight_info[0]),)
        conn.commit()
        #Publish To MQTT Broker
        publish(client,streetlight_info,'on',datetime_from_payload)
        #Patch Context Broker
        patch('on',datetime_from_payload)
        time.sleep(300) # 5 minutes = 300 seconds
        print("5 minutes passed. The light is OFF")
        print()
        #Set Streetlight's powerState to 'off'
        cursor.execute("UPDATE Streetlight SET powerState='off' WHERE id=?", (streetlight_info[0],))
        conn.commit()
        publish(client,streetlight_info,'off',datetime_from_payload)
        patch('off',datetime_from_payload)
```

## 5. Publish to MQTT Broker

Σε πραγματικές συνθήκες, για να αλλάξουμε την κατάσταση του φωτός θα έπρεπε να κάνουμε publish στον MQTT Broker της εταιρείας που έχει το inteliLIGHT FRE-220-NEMA-L το κατάλληλο μήνυμα στο κατάλληλο topic. Παρόλα αυτά, επειδή δημιουργήθηκε τεχνικό πρόβλημα με τα licenses της εταιρείας, δεν μπόρεσε να υλοποιηθεί.

Για αυτό, πραγματοποιήσαμε μία προσομοίωση όπου στέλνουμε σαν μήνυμα ένα json αρχείο

με τις πληροφορίες που έχουμε στη βάση μας για το Streetlight σε ένα δικό μας topic. Ουσιαστικά τα πεδία που αλλάζουμε είναι το status το οποίο παίρνει τις τιμές on και off όταν θέλουμε να ανάψουμε και να σβήσουμε το φως αντίστοιχα και το dateLastSwitchingOn που παίρνει την τιμή της ώρας που έφτασε το μήνυμα από τον αισθητήρα πάρκινγκ.

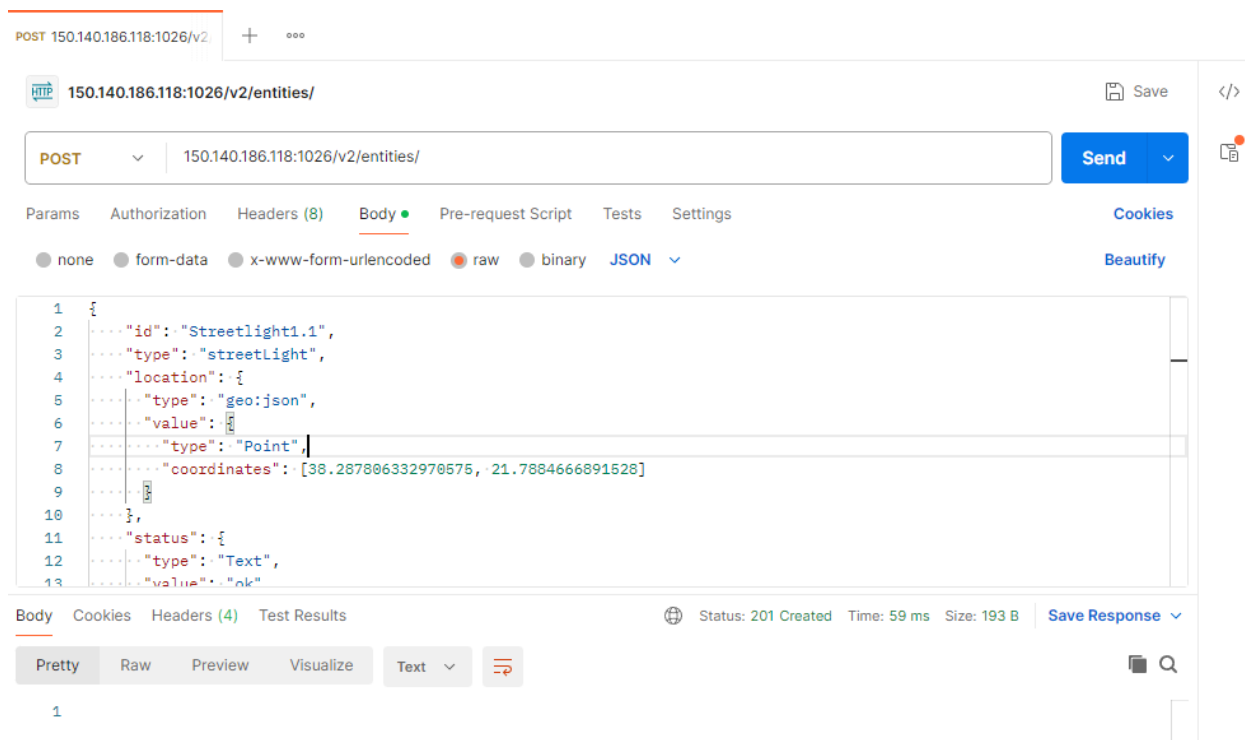
```
def publish(client,streetlightInfo,powerState,dateLastSwitchingOn):
    topic = "json/Parking/inteliLIGHT-FRE-220-NEMA-L"
    msg = f'{{"id":{streetlightInfo[0]}, "location":{streetlightInfo[1]}, "status":{streetlightInfo[2]}, "type":{streetlightInfo[3]},
    ["powerState":{powerState},"dateLastSwitchingOn":{dateLastSwitchingOn},"refDevice":{streetlightInfo[6]} }}'
    result = client.publish(topic, msg)
    status = result[0]
    if status == 0:
        print(f"[MQTT Broker] Send `{msg}` to topic `{topic}`")
        print()
    else:
        print(f"[MQTT Broker] Failed to send message to topic {topic}")
        print()
```

Αν κάνουμε subscribe σε αυτό το topic μπορούμε να δούμε κανονικά αυτό το μήνυμα να έρχεται.

## 6. Patch Context Broker

Για να λειτουργήσουμε και με τον Context Broker, μιάς και τα δεδομένα μας ήταν ήδη στη μορφή Smart Data Models, κάναμε τα εξής βήματα:

1. Κάναμε post στο κατάλληλο url (<http://150.140.186.118:1026/v2/entities/>) ένα json μήνυμα με τις πληροφορίες που είχαμε στη βάση για το StreetLight, μέσω της πλατφόρμας POSTMAN.





Το body του μηνύματος:

```
{  
  "id": "Streetlight1.1",  
  "type": "streetLight",  
  "location": {  
    "type": "geo:json",  
    "value": {  
      "type": "Point",  
      "coordinates": [38.287806332970575, 21.7884666891528]  
    }  
  },  
  "status": {  
    "type": "Text",  
    "value": "ok"  
  },  
  "powerState": {  
    "type": "Text",  
    "value": "off"  
  },  
  "dateLastSwitchingOn": {  
    "type": "DateTime",  
    "value": "2024-01-25T18:23:25.339113601+00:00"  
  },  
  "refDevice": {  
    "type": "Text",  
    "value": "0004a30b00e95f14"  
  }  
}
```

```
}
```

2. Για να αλλάξουμε την κατάσταση του φωτός, κάναμε patch στο κατάλληλο url (<http://150.140.186.118:1026/v2/entities/Streetlight1.1/attrs>) με τα πεδία που θέλουμε να αλλάξουμε:

```
def patch(powerState,dateLastSwitchingOn):
    url="http://150.140.186.118:1026/v2/entities/Streetlight1.1/attrs"
    headers = {"Content-Type": "application/json"}
    payload = {
        "powerState": {
            "type": "Text",
            "value": powerState
        },
        "dateLastSwitchingOn": {
            "type": "DateTime",
            "value": dateLastSwitchingOn
        }
    }
    response = requests.patch(url, headers=headers, data=json.dumps(payload))
    if 200 <= response.status_code < 300:
        print("[Context Broker] Patched ",json.dumps(payload, indent=2)," to URL ",url)
        print()
    else:
        print("[Context Broker] Failed to patch ",json.dumps(payload, indent=2)," to URL ",url)
        print()
```

Αν κάνουμε get αυτό το URL μπορούμε να δούμε κανονικά αυτό το μήνυμα να έρχεται.

## 2.5 TESTING

### *Fake Data*

Επειδή το payload από τον MQTT Broker δεν έρχεται τόσο συχνά, χρειάστηκε να κάνουμε testing με fake data.

```
def run():
    client = connect_mqtt()

    real_data=True #if True: reads data from MQTT Broker, else reads fake data

    if (real_data==True):
        payload=subscribe(client)
    else:
        payload=generate_fake_data()
```

Χρησιμοποιήσαμε τη συνάρτηση generate\_fake\_data για να δώσουμε ένα ενδεικτικό payload το οποίο έχει τα ίδια πεδία με το payload που έρχεται και από τον MQTT Broker. Στα πλαίσια του testing, αλλάζαμε τα πεδία του fake payload για να ελέγξουμε ότι η εφαρμογή λειτουργεί σωστά για όλα τα σενάρια χρήσης, όπως φαίνονται στο διάγραμμα ροής της εικόνας 2.4.1.

Το παρακάτω παράδειγμα δείχνει αυτό που εκτυπώνεται ενδεικτικά στην περίπτωση που το παρκινγκ είναι επιτυχές οπότε πρέπει να ανάψει το φως:

```
New Message from Sensor: 0004a30b00e95f14

A car parked in the parking spot with ID: Parking1.1

The car matches the one from the booking. The light is ON for 5 minutes

[MQTT Broker] Send `{'id':'Streetlight1.1','location':None,'status':'ok', 'type':'streetLight', 'powerState':'on','dateLastSwitchingOn':'2024-01-25T18:22:25.339113601+00:00','refDevice':'0004a30b00e95f14'}` to topic `json/Parking/inteliLIGHT-FRE-220-NEMA-L`

[Context Broker] Patched {
  "powerState": {
    "type": "Text",
    "value": "on"
  },
  "dateLastSwitchingOn": {
    "type": "DateTime",
    "value": "2024-01-25T18:22:25.339113601+00:00"
  }
} to URL http://150.140.186.118:1026/v2/entities/Streetlight1.1/attrs
```

Και έπειτα από 5 λεπτά:

```
5 minutes passed. The light is OFF

[MQTT Broker] Send `{'id':'Streetlight1.1','location':None,'status':'ok', 'type':'streetLight', 'powerState':'off','dateLastSwitchingOn':'2024-01-25T18:22:25.339113601+00:00','refDevice':'0004a30b00e95f14'}` to topic `json/Parking/inteliLIGHT-FRE-220-NEMA-L`

[Context Broker] Patched {
  "powerState": {
    "type": "Text",
    "value": "off"
  },
  "dateLastSwitchingOn": {
    "type": "DateTime",
    "value": "2024-01-25T18:22:25.339113601+00:00"
  }
} to URL http://150.140.186.118:1026/v2/entities/Streetlight1.1/attrs
```

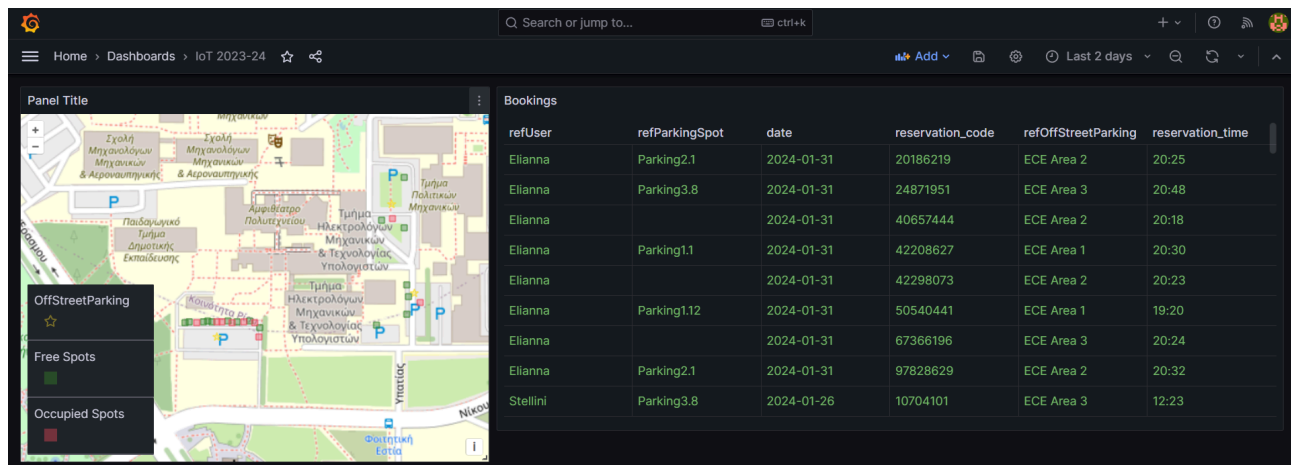
### *Real Data*

Το testing με real data, δηλαδή με το να παρκάρει όντως κάποιο αυτοκίνητο στη θέση πάρκινγκ ώστε να εξετάσουμε τη λειτουργία του συστήματος, δυστυχώς δεν μπόρεσε να υλοποιηθεί για 2 λόγους:

- 1) Όταν ολοκληρώσαμε το τμήμα του κώδικα που αναγνώριζε αν έχει παρκάρει κάποιο αυτοκίνητο και έπαιρνε τις απαραίτητες αποφάσεις, το τμήμα τελούσε υπό κατάληψη.
- 2) Υπήρχαν κάποια Licensing issues με την εταιρεία που είχε το inteliLIGHT FRE-220-NEMA-L οπότε δεν μπορέσαμε να αποκτήσουμε πρόσβαση σε αυτό ώστε να ανάβουμε και να σβήνουμε το φως με δική μας πρωτοβουλία.

## 2.6 GRAFANA

Για την υλοποίηση ενός admin control συστήματος, χρησιμοποίησαμε τα Dashboard της πλατφόρμας "Grafana".



Εικόνα 2.6.1 Admin dashboard in Grafana.

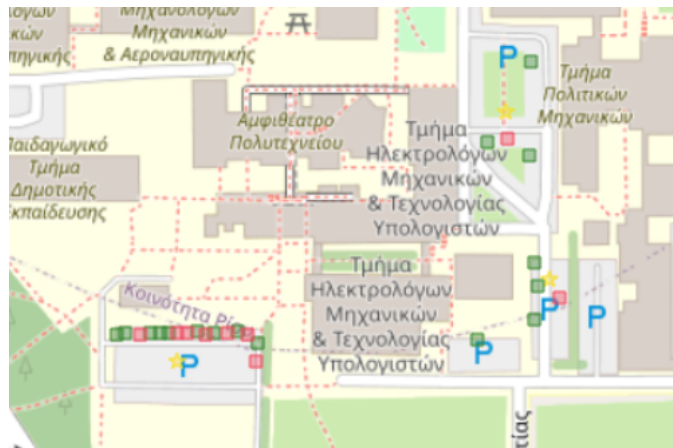
Το dashboard, αποτελείται από 2 panel. Μέσω των panel, ο admin έχει τη δυνατότητα να παρακολουθήσει το status όλων των parking spots αλλά και να δει ποια είναι τα τελευταία reservation θέσεων που θα εκπληρωθούν σύντομα.

### *Parking Spots: Availability Map*

Ο χάρτης δείχνει σε δεδομένα πραγματικού χρόνου, το status των Parking Spots σε όλα τα Parking Sites της εφαρμογής. Συγκεκριμένα:

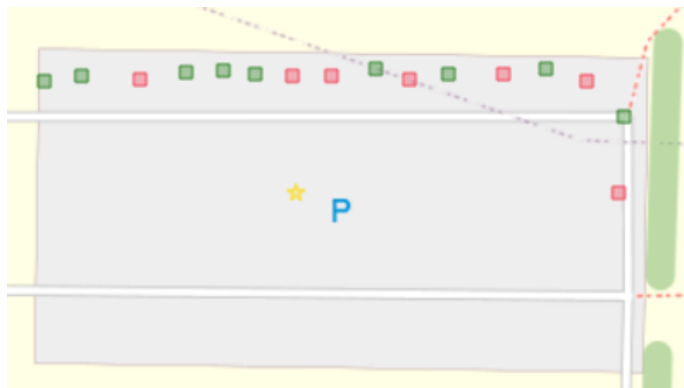
- Με ένα ★ απεικονίζονται τα 3 Parking Sites
- Με ένα ■ απεικονίζονται τα Parking Spots που είναι "free"
- Με ένα ■ απεικονίζονται τα Parking Spots που είναι "occupied"

Τα data αυτών των πληροφοριών λαμβάνονται μέσω της βάσης δεδομένων sqlite που βρίσκεται locally.



Εικόνα 2.6.2: Στιγμιότυπο Grafana

Για τα ECE Parking Site 2 & 3 δημιουργήθηκαν ως παράδειγμα ορισμένα στιγμιότυπα θέσεων. Ένα πλήρες παράδειγμα του συστήματος φαίνεται στο ECE Parking Site 1 παρακάτω.



Εικόνα 2.6.3: ECE Parking Site 1

Φαίνονται όλες οι θέσεις των Parking Sites. Η θέση **Parking 1.1** (πάνω-δεξιά, free) είναι η θέση που βρίσκεται τοποθετημένος ο αισθητήρας του Project και λαμβάνει data από τον σένσορα.

Οι πληροφορίες αυτές εμφανίζονται με το feature του Grafana: **Geomap** το οποίο λαμβάνει δεδομένα από τη database και συγκεκριμένα από τους πίνακες "OffStreetParking" και "ParkingSpot".

	id	type	location	name
	Filter	Filter	Filter	Filter
1	ECE1000	OffStreetParking	[38.287724103241736, 21.788015603191493]	ECE Area 1
2	ECE2000	OffStreetParking	[38.28808994937912, 21.79012209139717]	ECE Area 2
3	ECE3000	OffStreetParking	[38.288838652012664, 21.78987639465963]	ECE Area 3

Εικόνα 2.6.4: Πίνακας OffStreetParking

Μέσω επεξεργασίας στο Grafana, η στήλη “location” μεταφράζεται στις κατάλληλες συντεταγμένες που δείχνουν τη θέση του κάθε πάρκινγκ.

Αντίστοιχη λειτουργία γίνεται και με τον πίνακα ParkingSpot στο οποίο επιπλέον λαμβάνεται υπόψη η στήλη “status”.

	id	name	type	status	category	location	refParkingSite
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Parking1.1	A-1	ParkingSpot	occupied	offStreet	[38.287806332970575, 21.7884666891528]	ECE Area 1
2	Parking1.8	A-8	ParkingSpot	free	offStreet	[38.287852649290784, 21.787959751674855]	ECE Area 1
3	Parking1.10	A-10	ParkingSpot	occupied	offStreet	[38.287850544004144, 21.788064357821096]	ECE Area 1
4	Parking2.1	B-1	ParkingSpot	free	offStreet	[38.28816652159602, 21.790038988088863]	ECE Area 2
5	Parking2.2	B-2	ParkingSpot	free	offStreet	[38.288063640418486, 21.790056594248867]	ECE Area 2
6	Parking2.5	B-5	ParkingSpot	free	offStreet	[38.28791675269953, 21.790044582009774]	ECE Area 2
7	Parking2.15	B-15	ParkingSpot	occupied	offStreet	[38.28801359573345, 21.79018539797587]	ECE Area 2
8	Parking3.1	C-1	ParkingSpot	free	offStreet	[38.28870833546238, 21.789772337806323]	ECE Area 3

Εικόνα 2.6.5: Μέρος του πίνακα ParkingSpot

### Bookings

Ο πίνακας Bookings εμφανίζεται σαν δεύτερο panel στον admin με σκοπό την επισκόπηση των κρατήσεων θέσεων.

Bookings					
refUser	refParkingSpot	date	reservation_code	refOffStreetParking	reservation_time
Elianna	Parking3.8	2024-02-07	21397267	ECE Area 3	17:50
Elianna	Parking2.15	2024-02-07	89264118	ECE Area 2	20:50
Stellini	Parking3.2	2024-02-07	92990411	ECE Area 3	16:52
Stellini	Parking2.5	2024-02-06	84717697	ECE Area 2	19:52
Elianna	Parking1.9	2024-02-05	56925442	ECE Area 1	17:50
Stellini	Parking1.11	2024-02-03	80064564	ECE Area 1	17:51
Stellini	Parking2.5	2024-02-02	44555599	ECE Area 2	18:52
Elianna	Parking2.5	2024-02-02	60645341	ECE Area 2	17:51
Elianna	Parking2.1	2024-01-31	20186219	ECE Area 2	20:25

Εικόνα 2.6.6: Πίνακας Bookings στο Grafana.

Τα entries του πίνακα είναι τοποθετημένα σε σειρά ώστε το πιο σύντομο χρονικά να εμφανίζεται πρώτο. (ORDER BY date DESC)

### 3. ΕΡΓΑΛΕΙΑ

#### Gantt Chart

Για παρακολούθηση κάθε σταδίου της εργασίας χρησιμοποιήσαμε σαν πρωτεύον project management tool το Gantt Chart.

Συνοπτικά, το χρονοδιάγραμμά μας φαίνεται στον παρακάτω πίνακα:

Generic Task	Responsible	November	December	January	February
Pitching	Stella&Elianna				
App Design Figma	Stella&Elianna				
Architecture Design	Stella&Elianna				
Database Design v1	Elianna				
Connect to MQTT Broker	Stella				
App's Frontend	Stella				
Database Design with SMDs	Elianna				
Dummy Data for DB	Stella&Elianna				
App's Backend	Elianna				
Exchange Data with MQTT Broker	Stella				
Data Flow Implementation	Stella				
Grafana	Elianna				
Test with Dummy Data	Stella				
Connect Context Broker	Stella&Elianna				
Exchange Data with Context Broker	Stella				
Test with Real Data					
Report	Stella&Elianna				





					FEBRUARY																					
AREA	TASK	WHO	STATUS	Comment	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F
					19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4	5	6	7	8	9
Meeting	12th Meeting	Elianna & Stella	Done																							
Database	update erd	Elianna	Done																							
Database	update db	Elianna	Done																							
Code	get the final WRITE in the db	Elianna	Done																							
Meeting	13th Meeting	Elianna & Stella	Done																							
Coach	mail to coach to meet	Stella	Done																							
Coach	Meeting with coach #4	Elianna & Stella	Done																							
Code	if logic with prints	Stella	Done	if right driver & right																						
Code	make pop up for reservation success	Elianna	Done																							
Database	insert instances in parkingSpot&device in DB	Stella	Done																							
Code	update if_logic.py so it reads everything from D	Stella	Done																							
Meeting	14th Meeting	Elianna & Stella	Done																							
Code	check if parking spot is reserved (from booking	Stella	Not started	LOW PRIORITY																						
Code	transfer if_logic.py in smartspot_jot & connect t	Stella	Done																							
Code	update if_logic.py so it writes in Streetlight	Stella	Done																							
Code	when Device triggered -> update ParkingSpot t	Stella	Done	in adapt_payload_t																						
Other	connect Grafana to MQTT	Elianna & Stella	Done																							
Database	add more parking spots	Elianna	Done																							
Code	get imgs corresponding to parking area	Elianna	Done																							
Other	add backend in report	Elianna	Not started																							
Other	add mqtt in report	Stella	In Progress																							
Meeting	15th Meeting	Elianna & Stella	Done																							
Meeting	16th Meeting	Elianna & Stella	Done																							
Code	create context broker way	Elianna & Stella	Done																							
Architecture	fix it so that it's 1 way	Stella	Done																							

FINAL MILESTONE

## Github

Προκειμένου να μπορούμε να δουλεύουμε παράλληλα στο πρότζεκτ και να ελέγχουμε καλύτερα τις εκδόσεις του, δημιουργήσαμε ένα [Github repository](#), στο οποίο ανεβάσαμε όλα τα αρχεία που φτιάξαμε.

## ΑΝΑΦΟΡΕΣ

1. [Increasing Public and Private Parking Lot Safety](#) | MCA FAMILY OF COMPANIES
2. [Welcome To Orion Context Broker](#) | FIWARE-ORION
3. [Smart Data Models](#) | Github
4. [Our Project](#) | Github
5. [Grafana: The open observability platform](#) | Grafana Labs
6. [Postman API Platform](#)