UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desenvolupament de carreteres autònomes

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jordi Urios i Llorens

Tutor: Joan Fons i Cors

2016-2017

Resum

La constant millora i aparició de nous components de maquinari acompanyat amb noves tecnologies que permeten la interconnexió d'aquests ha donat peu a tota una plataforma de serveis destinats a millorar les ciutats i la relació que el ciutadà té amb ella.

Un dels punts de millora a destacar d'aquest nou paradigma és la possibilitat d'obtenir amb aquesta tecnologia una millora substancial de l'eficiència dels recursos que despesa una ciutat en els serveis que ofereix.

Aquest projecte pretén presentar una solució que, emprant aquest ecosistema tecnològic, propose una nova forma de gestionar i millorar l'eficiència energètica en l'enllumenat a les carreteres redissenyant la infraestructura i proposant noves polítiques adaptades a les noves capacitats d'aquesta.

Paraules clau: ciutats intel·ligents, carreteres intel·ligents, eficiència energètica, internet de les coses.

Resumen

La constante mejora y la aparición de nuevos componentes acompañado por las nuevas tecnologías que permiten la interconexión de éstos han llevado a cabo toda una nueva plataforma de servicios destinados a mejorar las ciudades y la relación de los ciudadanos con éstas.

Uno de los puntos de mejora a destacar de este nuevo paradigma es la posibilidad de obtener con esta tecnología una mejora sustancial en la eficiencia de los recursos que gasta una ciudad en los servicios que ofrece.

Este proyecto pretende presentar una solución que, utilizando esta tecnología, proponga una nueva forma de gestionar y mejorar la eficiencia energética en el alumbrado de las carreteras rediseñando la infraestructura y proponiendo nuevas políticas adaptadas a las nuevas capacidades de ésta

Palabras clave: ciudades inteligentes, carreteras inteligentes, eficiencia energética, internet de las cosas.

Abstract

As a result persistence betters and the appearance of new hardware components together with new technologies which allow the connexion of these components, a new services platform has been launched to improve the cities and citizens relationships.

One of the outlined aims to improve this new paradigm is the possibility to obtain a substantial improvement in the efficiency of the resources used by cities.

This project is intended to present a solution which, using this technology, suggests a new way of managing and bettering the energy efficiency of the street lighting of roads. Redesigning the infrastructure plus proportioning new policies adapted to new capacities

Keywords : smart cities, smart roads, energy efficiency, internet of things.

Tabla de contenidos

1. Introducció

La convergència de múltiples tecnologies - com ara les connexions sense fil, el *big data*, el *machine learning* així com també la varietat de sensors i dispositius incrustats en el mercat - que han anat evolucionat al pas del temps - han permès la interconnexió, a través d'Internet, d'objectes físics quotidians més enllà dels ordinadors. Electrodomèstics, mobles, entre altres, a l´àmbit de la llar formen les cases intel·ligents. Sensors, edificis, mòbils, vehicles, entre altres, a les nostres urbs formen les ciutats intel·ligents. En un futur més que present, tot dispositiu capaç de rebre o enviar informació, i la tendència és que hi seran molts, formaran part d'una xarxa ambiciosa de comunicació que permetrà als mateixos dispositius compartir informació de l'entorn. Aquests dispositius, dotats de capacitat per analitzar i entendre tot allò que passa al seu voltant podran prendre decisions basades en informació real i actual sense la necessitat de cap interacció humana. Seran, doncs, objectes intel·ligents en una xarxa d' Internet de les Coses (en avant, *IoT*).

Sense cap dubte, aquesta nova tecnologia permet millorar molts aspectes del món que ens envolta. Amb la ingent quantitat d'informació al nostre abast produïda pels milers i milers de dispositius escampats per les ciutats podem - o millor dit, el núvol *IoT* de dispositius i serveis poden- fer una anàlisi més exhaustiva i a temps real de les necessitats de les ciutats sabent quan és necessari un servei i quan no ho és.

Això obri les portes, entre altres, a iniciar un procés de millora, optimització i eficiència energètica a les ciutats i als pobles per tal d'escometre la necessitat mediambiental de convertir les nostres urbs en ciutats eficients i respectuoses en el medi ambient.

El present projecte pretén dissenyar i prototipar una solució autònoma en l'àmbit de les ciutats intel·ligents. A l'inici del projecte vam proposar diferents enfocaments que podria tenir el projecte - gestió de residus, gestió de l'aigua, control del consum energètic, etc...- i vam optar per centrar-nos en la gestió lumínica, donada la seua importància en les despeses energètiques actuals i l'impacte mediambiental.

1.1 Motivació

La il·luminació urbana juga un paper molt important en les nostres ciutats i pobles. També ho fa a les nostres carreteres. Imaginem-nos per un instant com seria el nostre dia a dia sense il·luminació a les cases o als carrers. La il·luminació artificial permet estendre les nostres activitats més enllà de la posta de sol. A les societats prèvies a l'aparició de l'enllumenat públic la societat estava forçada a acceptar un horari actiu limitat a les hores de sol. Instal·lar enllumenat públic permet estendre el nostre horari amb els beneficis i les conseqüències que això implica.

A més, l'enllumenat públic és element fonamental en la seguretat vial. Resulta més segur per a la integritat física d'un vianant, per exemple, caminar per una carretera ben il·luminada que no pas anar per una carretera fosca o incorrectament il·luminada.

Això ha fet que, junt al creixement dels propis centres urbans, la tendència als excessos de la societat de consum i la bombolla immobiliària, sobretot a la nostra zona, ha fet també créixer l'enllumenat públic. Malauradament, aquest creixement imparable té unes conseqüències. Per una banda, la conseqüència més tangible és el cost econòmic de mantenir aquesta il·luminació.

Per altra banda, no totes les carreteres són igual de transitades. Cada una té diferents necessitats lumíniques i, a certes hores de la nit, la majoria de les carreteres no són pràcticament transitades (sobretot si parlem de pobles i carreteres interurbanes) pel que s'està gastant energia per una il·luminació innecessària que, a més a més, afecta directament en el temps de vida de les faroles. I finalment, i no menys important, aquesta ingent quantitat d'il·luminació, molta d'ella mal aprofitada, genera nombroses conseqüències en el medi ambient. No només en l'emissió de gasos d'efecte hivernacle que s'emeten al generar aquesta energia, sinó també per la degradació dels ecosistemes nocturns i els efectes que en la salut pot tenir la contaminació lumínica.

En un punt de no retorn davant el canvi climàtic en el que ens trobem, qualsevol solució -per molt xicoteta que siga- que ajude a mitigar el problema mediambiental al que ens afrontem dia a dia, sempre serà benvinguda.

1.1.2 Un problema mundial és també un problema local

Segons Europapress[1] al 2017, un terç de la població mundial no pot veure la Via Làctia degut als greus nivells de contaminació lumínica dels nuclis urbans.

Però tot problema global té uns factors locals i en aquest cas a València en tenim alguns. Un article del País[2] al 2009, remarcava que València és la ciutat amb major índex de contaminació lumínica d'Europa suposant això una despesa superior als 13,5 milions d'euros al 2007. A la ciutat del Túria es consumeixen 127 quilovats/hora per habitant front als 61,5 a Madrid o als 57 a Barcelona.

A més, segons l'informe del departament d'Astrofísica i Ciències de l'Atmosfera de la Universitat Complutense de Madrid, segons l'article, afirma que Espanya és el país de la Unió Europea que més consumeix per farola. Castelló i Alacant també hi són per damunt de la mitja europea. En el mateix article, Lydia Freire - presidenta del centre d'Investigació Astronòmica d'Alacant - afirma que des d'Alacant es pot veure la llum que emet València.

Sense entrar a qüestionar sobre la necessitat o no de tantes faroles - sobre això en parlaré a la valoració personal-, és imperatiu cercar una solució. Per a la nostra economia, per a la nostra salut i per a la salut dels ecosistemes en contacte.

En aquesta punt, el IoT i els dispositius intel·ligents tenen un paper summament important que jugar. Poden ser part d'una solució necessària per tal de construir ciutats tolerants en el medi ambient, òptimes i eficients on, al mateix temps, la seguretat de les seues carreteres no es veja compromesa, ni minvats els serveis que presta als seus ciutadans.

Imatge nocturna de la ciutat de València per la NASA

1.2 Objectius

L'objectiu principal d'aquest projecte és presentar una infraestructura capaç d'autoregular el consum energètic en funció de les necessitats del moment. Fent ús de les tecnologies que envolten les ciutats intel·ligents i el paradigma del IoT i amb la intenció prioritària de reduir el consum energètic i dotar a la xarxa de noves capacitats per produir un servei eficient.

Per tal de fer front a aquest objectiu, s'han de realitzar tres fites remarcables:

1. **Dotar a la infraestructura actual d'autonomia**. Per tal d'implantar la solució del projecte es requereix desenvolupar certes millores tecnològiques en la infraestructura actual que il·luminen les nostres carreteres. Aquestes millores permetran un desenvolupament de normatives i polítiques molt més flexibles focalitzant-ho a un ús de la llum més eficient sense que això implique una deficiència en el servei lumínic.
2. **Desplegar les noves polítiques i normatives que permeten un ús eficient i responsable de la xarxa lumínica**. Implantar les millores tecnològiques en la infraestructura permetrà gestionar amb més precisió l'ús

que fem de la il·luminació. Les normatives i les polítiques actuals - que s'empren en la infraestructura actual-, són excessivament rígides i no tenen en compte aquestes millores implantades. Caldrà aleshores redissenyar de nou una normativa i la forma de gestionar les polítiques en aquesta nova infraestructura.

3 **Prototipar un escenari que exemplifique i mostre les capacitats de la nova xarxa**. Una vegada dissenyats els punts anteriors, desenvolupar un escenari que mostre com funcionarà aquesta nova infraestructura amb el nou sistema de polítiques i la normativa.

1.3 Metodologia

La metodologia emprada per al desenvolupament està basada per fases.

- **Cas d'estudi**: La temàtica del projecte podia abastir diferents aspectes a tractar. Després de veure totes les possibilitats vaig optar per tractar la eficiència energètica a les carreteres autònomes. En aquesta fase vaig cercar la normativa actual i quins aspectes a millorar es podrien aplicar per obtenir una millora substancial en aspectes relacionats en l'eficiència energètica.
- **Anàlisi i disseny de la infraestructura**: En aquest punt vaig determinar quines característiques havia de tenir la infraestructura per suportar la solució a implementar. Aquesta fase va estar sustentada per un procés iteratiu on plantejava una solució, llistava les millores i les contres que sorgien per a, finalment, replantejar una nova solució que intentava solventar els problemes. Un cop ja tenia dissenyades diferents solucions, cada una amb les seues virtuts i problemàtiques, vaig dissenyar com s'aplicarien polítiques en aquesta nova infraestructura.
- **Aprenentatge de la tecnologia**: Per a la següent fase vaig optar per emprar una nova tecnologia que encara no havia gastat: *AngularJS*. En aquesta fase vaig estar aprenent com fer servir aquesta eina i poder adaptar-la a les altres tecnologies que anava a utilitzar.
- **Implementació**: Per tal de "mostrar" aquesta nova infraestructura vaig optar per fer un prototip d'aquesta. No obstant, degut a les limitacions tècniques - no tenia un grup de faroles per provar-ho - el resultat que obtenia no era el desitjat. Per tant, vaig optar per desenvolupar un petit simulador on es pot veure com s'apliquen les polítiques segons el disseny implementat.
- **Proves**: Finalment, la última fase la vaig destinar a realitzar proves de funcionalitat en el simulador. Vaig preparar un escenari que mostrara l'ús de les noves polítiques.

Totes aquestes fases han estat supervisades pel meu tutor. Des d'un inici, vaig plantejar al meu tutor abordar aquest projecte a distància. Jo anava a estar desenvolupant el meu projecte al meu poble i em resultava interessant poder treballar amb el meu tutor a distància. Per una banda anar a València implica un cost econòmic important. Per altra banda, em suposava un repte interessant d'assolir el de treballar a distància tenint en compte que, dia a dia, són més freqüents els treballs remots.

Vaig optar per treballar amb un *Gitflow* fent servir els beneficis de Git per al treball col·lectiu remot i les opcions que té per poder detectar les modificacions entre versions del projecte per aportar-li al meu tutor un resum de les meues modificacions.

Tot i així, també acordàrem algunes reunions presencials per tal de tancar certs fils que des de la plataforma de GIT ens resultaven contraproduents tancar-los.

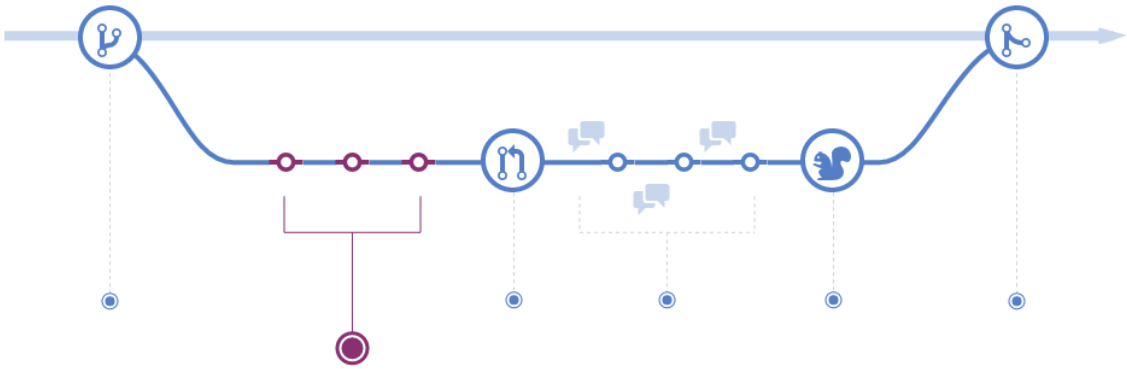
1.3.1 Git, Gitflow, Issues i Merge Requests

Git és un sistema de control de versions distribuït i escalable. És un projecte de programari lliure i gratuït que permet treballar en diferents versions del codi i tindre un *tracking* dels canvis realitzats.

A més, Git és àmpliament utilitzat i compta amb molts serveis web gratuïts. El que ens permetia, al meu tutor i a mi, treballar amb una aplicació web visual sobre les meues modificacions.

La flexibilitat de Git i el seu disseny descentralitzat i basat en rames ha donat peu a milers de maneres de treballar amb ell. Cada servei web de Git, cada empresa, cada desenvolupador té la seua manera (*flow*) de fer-lo servir i d'organitzar-lo.

Nosaltres vam optar per la forma de treballar que recomana GitHub per als seus repositoris[3]:



Imatge del flow que recomana GitHub en el que em vaig basar

En tot moment hi havia una rama principal que contenia una versió del projecte corregida i validada pel meu tutor. Des d'aquest punt s'iniciava una iteració:

Creació d'una *Issue*

La iteració s'inicia amb el plantejament d'un nou punt a desenvolupar. Creava, o el meu tutor ho feia, un Issue (un fil de discussió en el nostre servei Git web) on debatia amb el meu tutor com abordar certs aspectes del projecte. Després de la conversa i quan ja ho tenia clar, s'iniciava el desenvolupament.

Una de les issues del projecte

Creació d'una rama específica

Un cop definits la tasca en l'Issue, creava una rama que versionava el projecte on desenvolupava allò comentat en el *Issue*.

Merge Request

Un cop finalitzat el desenvolupament, li enviava una petició al meu tutor per unificar les meues modificacions en la rama principal. El meu tutor, a través del *Merge Request*, podia veure tots els meus canvis. En el *Merge Request* també hi ha un fil de discussió on poder conversar sobre les modificacions i buscar un consens abans d'acceptar-les.

Merge a principal

Un cop el tutor em donava el vist i plau de les modificacions, acceptava el *Merge Request* i s'aplicaven els canvis realitzats sobre la rama principal. I s'iniciava una nova iteració.

Després d'uns mesos fent servir aquesta metodologia, vaig poder extraure la següent taula dels beneficis i els contres de fer servir aquesta metodologia front a la tradicional de les tutories i correus electrònics.

Pros	Contres
• Llibertat horària. No cal comunicació directa, tot va per fils de discussió • Discussions organitzades per aspectes del desenvolupament i directament enllaçades en versions del codi (Una issue i un Merge Request estan relacionats directament amb una iteració del desenvolupament) • Facilitat per al tutor per corregir modificacions. Git detecta les modificions pel que és menys tediós per al tutor cercar les modificacions	• Paciència. Fer entendre't quan escrius no és tan fàcil com quan parles. • Temps d'espera entre missatges. No és el mateix que una conversa directa • Adaptar-se a la nova metodologia implica cert temps d'aprenentatge

Taula de valoració

1.4 Pla de treball

El desenvolupament del projecte va seguir el següent pla de treball:

	Novembre	Desembre	Gener	Febrer	Març	Abril	Maig	Juny
Cas d'estudi	X	X						
Anàlisi i disseny de la infraestructua			X	X	X			
Aprenentatge de la tecnologia						X		
Implementació					X	X	X	X
Proves								X

Pla de treball

2. Context tecnològic

2.1 Programari

2.1.2 Git, GitLab i Markdown

Git[4] és un sistema de control de versions distribuït i escalable. És un projecte de software lliure i gratuït que permet treballar en diferents versions del codi i tindre un *tracking* dels canvis realitzats.

Des d'un inici, sabia que anava a fer servir un sistema de control de versions. Primer, per a poder tindre una copia de seguretat actualitzada del meu codi en un servidor evitant així desastres majors com la perduda del projecte. Per altra banda, em resulta interessant poder treballar en diferents versions del meu codi, permetent tirar arrere quan alguna nova característica no s'integrara correctament en el projecte. A més, m'interessava poder "obtenir" el treball realitzat entre dos punts temporals diferents per a que el meu tutor li resultarà més fàcil veure el meu progrés, com he explicat al punt anterior.

Per tal de fer servir Git, vaig investigar els diferents serveis Git-web que hi han actualment. Entre la gran varietat que hi vaig trobar vaig optar per *GitLab* que ofereix repositoris privats de forma gratuïta.

Tot i que Git inicialment està pensat per al control de versions de codi, molts serveis de Git-web ofereixen un llenguatge de marcat lleuger anomenat *Markdown*[5] per poder escriure documentació. Em resultava interessant aquesta característica doncs es podria escriure la memòria fent servir els beneficis de Git.

D'aquesta forma era relativament senzill poder saber quines diferències hi havia en la memòria entre les diferents versions que li manava al meu tutor per corregir. Facilitant-li així la feina de correcció.

2.1.2 Atom[6]

Per al desenvolupament del simulador he treballat, majoritàriament, en llenguatges interpretats. Això m'ha permès certa flexibilitat a l'hora d'escollir el meu entorn de desenvolupament doncs no necessitava executar processos complicats com pot ser una compilació. A més, el simulador no és un projecte de gran extensió pel que no necessitava les característiques que pot oferir un IDE.

Per al desenvolupament de la memòria, també he fet servir Atom doncs té un *renderitzador* de Markdown, el que em permetia escriure i veure el text renderitzat a temps real.

2.1.3 Postman[7]

Postman és una aplicació d'interfície d'escriptori que permet construir peticions HTTP i enviar-les a un *endpoint*.

Aquesta eina l'he gastada per provar l'*api* del *backend* del simulador. Una de les característiques que ofereix Postman, entre moltes altres, és que el programa manté un històric de peticions, que fa que pugues, amb rapidesa, provar el funcionament de la teua API.

2.1.4 Nginx[8] i Passenger[9]

Per tal de llançar el simulador públicament per a que el meu tutor puguera poder accedir a ell, i com el meu simulador empra tecnologies web, vaig instal·lar un servidor web amb *Nginx* i *Passenger*. *Nginx* és l'encarregat de gestionar les peticions dels clients i encaminar-les. Per una banda, encamina les peticions cap a l'*api* a Passenger, i les peticions referents al client web les gestiona el propi *Nginx*.

Passenger és el servidor d'aplicacions web que llança el *backend* del simulador(desenvolupat amb Ruby on Rails).

2.2 Maquinari

En aquest projecte es va utilitzar un servidor web per a publicar les diferents versions del simulador.

2.2.1 DigitalOcean[10]

Per a provar el simulador, vaig contractar un servidor al núvol durant les etapes finals del desenvolupament del TFG (quan ja tenia un simulador mostrable).

Ho vaig fer a DigitalOcean, una empresa proveïdora d'infraestructures *cloud* que facilita i simplifica el *deployment* de màquines virtuals.

Com que només volia llançar un petit simulador, sense grans requisits d'espai ni de potència vaig optar per la segona opció més barata (la primera no tenia suficient memòria RAM per mantenir tot el simulador en peu).

Característiques del servidor	
Sistema Operatiu	Ubuntu 16.04 x86_64
CPU	1
Memòria RAM	1 GB
SSD	30 GB
Transferència	2 TB

Taula de característiques del servidor

2.3 Llenguatges de programació

2.3.1 Ruby[11]

Ruby és un llenguatge interpretat, dinàmic, reflectiu i orientat a objectes. La idea principal de Ruby és centrar-se en millorar la productivitat del desenvolupament de codi.

2.3.1.1 Ruby on Rails[12]

Ruby on Rails és un *framework* Model-Vista-Controlador (en avant, MVC) escrit amb Ruby per al desenvolupament d'aplicacions *backend* web. Els motius pels que vaig optar per treballar en Ruby on Rails es poden extraure dels principis darrere del *framework* que són:

- **"Convenció per damunt de configuració"**: El programador només necessita especificar els aspectes no convencionals de l'aplicació.
  En el meu cas volia realitzar una API prou convencional i bàsica per al simulador.

- **No et repetisques**: La informació està localitzada en un lloc únic i no ambigu.
  No volia perdre molt de temps dissenyant la Base de dades. En aquest sentit Ruby on Rails treballa amb el patró *'Active Record*' que relaciona un objecte en una taula de la base de dades i una fila de la taula en una instància de l'objecte.

- **Grans models, petits controladors**: Tota la lògica de de l'aplicació deuria estar ubicada en el model i no en el controlador. Pràcticament als controladors del simulador no hi ha codi. No obstant, als models hi estan definides totes les relacions i validacions de la Base de dades.

Aquestes característiques ajudaven en prou mesura a realitzar un desenvolupament del *backend* ràpid.

2.3.2 Tecnologies web

2.3.2.1 HTML5[13], CSS[14]

Per a la part del client del simulador, inicialment vaig optar per realitzar una aplicació web fent servir els llenguatges habituals i estandarditzats per al desenvolupament web: HTML, CSS i Javascript.

No obstant, durant el procés de disseny vaig veure la necessitat de fer servir algun *framework* que treballara amb el patró MVC. La idea era construir una aplicació web fluida que semblara una aplicació d'escriptori. Vaig optar, doncs, per treballar amb Angular2.

2.3.2.2 TypeScript[15]

En lloc de treballar en JavaScript, Angular2 utilitza TypeScript. TypeScript és una superposició a JavaScript oferint algunes característiques que mancaven com el *tipat* opcional estàtic i la programació orientada a objectes. Al final, TypeScript és un llenguatge que compilat genera un JavaScript.

2.3.2.3 Angular2[16]

Angular2 és un *framework* basat en TypeScript per al desenvolupament de *frontend* web liderat per Google i una extensa comunitat de desenvolupadors. La idea darrere d'Angular és la de desenvolupar webs anomenades SPA (*Single Pages Applications*) fent servir el patró MVC.

Fer servir Angular2 em permetia construir una web més dinàmica amb un codi més net i mantenible gràcies a TypeScript.

2.3.2.4 JSON[17]

JSON és un format de text lleuger per a l'intercanvi de dades àmpliament emprat en entorns web. A més és un format integrat en Javascript pel que treballar en JSON amb Javascript, i per tant, en TypeScript, és molt senzill.

2.3.3 Base de dades SQLite[18]

Per a l'emmagatzemament de dades vaig optar per gastar SQLite. SQLite és un sistema de gestió de base de dades relacional. La diferència que té SQLite respecte a altres sistemes de gestió de base de dades relacional és que SQLite no segueix un patró

“servidor – client”. SQLite, en canvi, es llança amb l’aplicació passant a ser part integral d’aquesta.

Per a realitzar el simulador vaig optar per SQLite davant l’alternativa directa, MySQL, perquè no veia necessari configurar i muntar un servidor per a una aplicació que no requereix de grans quantitats de clients.

3. Cas d'estudi: Estat actual

Com he explicat en la introducció, aquest projecte tracta de resoldre el problema de la gestió energètica a les carreteres autònomes. La idea, a grans trets, és transformar el servei d'il·luminació a les carreteres d'un servei constant i continu en el temps a un servei baix demanda.

Què vol dir això? Doncs que les faroles estiguen engegades única i exclusivament quan algú necessite que estiguen engegades.

Durant anys, a les carreteres, el que s'ha implantat és una política basada per hores on els operaris (o els encarregats de gestionar la il·luminació de les carreteres) han determinat una hora d'apertura de les llums i una altra d'aturada. Als darrers anys, amb les noves exigències d'eficiència energètica, algunes faroles a certes hores intempestives de la nit baixen la potència. Per una xarxa d'il·luminació utilitzant dispositius no-intel·ligents és probablement la millor solució que es pot oferir per millorar la eficiència energètica.

No obstant, ara tenim un nou paradigma. Nous dispositius, faroles intel·ligents, una xarxa de milers i milers de dispositius representant la realitat amb dades que són capaços d'entendre l'entorn. A més, aquests dispositius poden comunicar-se amb serveis del núvol, i prendre decisions. Amb aquest nou context, aquesta política basada exclusivament per hores és una política rígida que no explota les noves capacitats de la xarxa.

Per tant, el que es cerca en el projecte és com modificar una xarxa d'il·luminació actual, basada en il·luminació no-intel·ligent, en una nova xarxa amb capacitat de poder gestionar eficientment els recursos fent ús del *IoT* i dels dispositius intel·ligents.

En aquest punt vaig analitzar la infraestructura i la normativa actual per la que es regeixen la majoria de les xarxes d'enllumenat públic.

3.1 Infraestructura física

Actualment, gairebé totes les ciutats segueixen un organigrama similar per organitzar i capacitar a la urbs d'enllumenat públic. Generalment, es divideix la ciutat en sectors (habitualment barris, o en funció del procés de creixement i les noves demandes d'il·luminació). Cada sector compta amb un "quadre de comandament" on s'han connectat totes les faroles i dispositius d'enllumenat. Aquest "quadre de comandament" és força important en la instal·lació doncs és el lloc físic on està ubicat el centre "lògic" de la xarxa al sector.

Aquest quadre prèviament ha estat configurat pel projectista en el moment de la instal·lació del sector. Cada sector compta amb uns requisits distints d'il·luminació en funció, com veurem, de la normativa. El projectista, a través d'un simulador d'il·luminació calcula els paràmetres i la quantitat de faroles en cada sector necessari

per complir amb la normativa vigent i configura al quadre de comandament aquests paràmetres per a les faroles.

Quadre de comandament vist des de fora i per dins

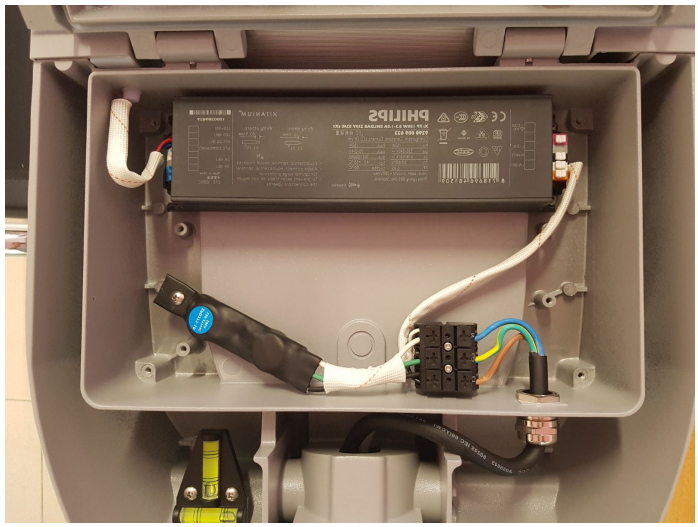
El quadre de comandament, a més, compta amb un sistema d'activació/desactivació de les faroles que permeten l'engegat d'aquestes automàticament a una hora determinada. Habitualment es fa a través d'un rellotge astronòmic tot i que existeixen altres mètodes com un sensor d'il·luminació (encara que aquesta solució és més problemàtica perquè si es fa malbé el sensor, tot el sector funcionarà malament).

El quadre també compta amb un sistema de regulació de flux que permet reduir la despesa energètica a certes hores. Aquest sistema és una solució efectiva si la instal·lació del sector ha estat correctament dissenyada. Si no és així, es pot donar el cas de que la baixada de tensió produïda per aquest sistema apague algunes faroles (les situades més lluny on la baixada de tensió és més sensible). Això fa que aquesta solució no siga sempre pràctica.

Sistema de regulació de flux en el quadre de comandament

Actualment, les instal·lacions més modernes compten amb un sistema que s'anomena la "maniobra del 50%". És un dispositiu que es fica al cap de cada farola i permet reduir la intensitat de la farola de forma individual. Això evita el problema esmentat en el sistema anterior. Es diu maniobra del 50% perquè es pot reduir la intensitat de la farola un 50%.

Amb la aparició de les faroles LED, es pot permetre reduir en 5 nivells diferents la intensitat de la farola, el que permet jugar millor amb les intensitats.



Sistema de regulació de flux a 5 nivells en el cap de la farola

Finalment, i és poc habitual perquè és una tecnologia cara, hi han algunes instal·lacions recents que el quadre de comandament compta amb un sistema de telegestió que permet modificar els paràmetres del quadre en remot i ja no cal anar físicament al quadre per modificar el comportament de les faroles.

Salvant l'aparició de les noves tecnologies, com puga ser la "maniobra del 50%"i la de 5 nivells de les faroles LED, generalment la instal·lació de l'enllumenat sol ser una infraestructura bastant rígida i que no permet jugar amb les faroles.

S'ha de tenir en compte que aquestes instal·lacions solen ser heretades de molts anys enrere i que, si no s'ha invertit en renovar-les, probablement no compten amb aquestes noves tecnologies d'eficiència energètica.

3.2 Normativa actual

La normativa que regula l'eficiència energètica en les instal·lacions exteriors està descrita en el Butlletí Oficial de l'Estat (Reial decret 1890/2008)[19]. Els punts rellevants estan relacionats amb l'article 6 de l'esmentat decret.

3.2.1 Article 6

S'han de complir els nivells màxims de luminància o il· luminació, i d'uniformitat mínima permesa, en funció dels diferents tipus de l'enllumenat exterior, segons el que disposa la ITC-EA-02.

3.2.2 ITC-EA-02

Es tracta d'una instrucció tècnica complementària en la que es determinen els nivells d'il·luminació que ha de tenir una zona exterior amb la intenció de garantir el servei d'il·luminació mínim eficient.

En aquesta secció de la normativa, es fa una classificació de les diferents vies i determina uns valors mínims d'il·luminació per a cada via.

Classificació	Tipus de via	Velocitat del trànsit rodat (km/h)
A	d'alta velocitat	$v > 60$
B	de velocitat moderada	$30 < v \leq 60$
C	carrils bici	--
D	de baixa velocitat	$5 < v \leq 30$
E	vies de vianants	$v \leq 5$

Taula de classificació segons el tipus de via

Per a cada via, hi ha una segona classificació en funció de les característiques físiques de la via i la densitat de tràfic mitjà(IMD).

Si ens fixem, per exemple, en les vies de tipus A (màxima velocitat), aquesta seria la classificació de la via.

Situacions de projecte	Tipus de vies	Classe d'enllumenat(*)
A1	• ***Carreteres de calçades separades amb encreuaments a diferent nivell i accessos controlats (autopistes i autovies).***	
	Intensitat de trànsit	
	Alta (IMD) ≥ 25.000	ME1
	Mitjana (IMD) ≥ 15.000 i < 25.000	ME2
	Baixa (IMD) < 15.000	ME3a
	• ***Carreteres de calçada única amb doble sentit de circulació i accessos limitats (vies ràpides).***	
	Intensitat de trànsit	
	Alta (IMD) > 15.000	ME1
	Mitjana i baixa (IMD) < 15.000	ME2
A2	• ***Carreteres interurbanes sense separació de voreres o carrils bici.***	
	• ***Carreteres locals en zones rurals sense via de servei.***	
	Intensitat de trànsit	
	IMD ≥ 7.000	ME1 / ME2
	IMD < 7.000	ME3a / ME4a
A3	• ***Vies col·lectores i rondes de circumval·lació.***	
	• ***Carreteres interurbanes amb accessos no restringits.***	
	• ***Vies urbanes de trànsit important, ràpides radials i de distribució urbana a districtes.***	
	• ***Vies principals de la ciutat i travessia de poblacions.***	
	Intensitat de trànsit i complexitat del traçat de la carretera.	
	IMD ≥ 25.000	ME1
	IMD ≥ 15.000 i < 25.000	ME2
	IMD ≥ 7.000 i < 15.000	ME3b
	IMD < 7.000	ME4a / ME4b

(*) Per a totes les situacions de projecte (A1, A2 i A3), quan les zones pròximes siguin clares (fons clars), totes les vies de trànsit han d'incrementar les exigències a les de la classe d'enllumenat immediatament superior.

Taula de classes d'enllumenat per a les vies A

Podem veure que en aquesta taula s'assigna a cada cas una classe d'enllumenat. Això ens porta a una tercera classificació de classe d'enllumenat on s'especifica els requisits mínims d'il·luminació de la carretera.

En les vies de la taula 2 s'assignen un enllumenat ME*. A la taula següent podem veure els paràmetre d'il·luminació mínima per a classes d'enllumenat ME*

Classe d'enllumenat	Luminància de la superfície de la calçada en condicions seques			Enlluernament pertorbador	Il·luminació dels voltants
	Luminància (4) mitjana L_m (cd/m^2)(1)	Uniformitat global U_o [mínima]	Uniformitat longitudinal $U_\square$ [mínima]	Increment llindar TI (%)(2) [màxim]	Relació entorn SR (3) [mínima]
ME1	2,00	0,40	0,70	10	0,50
ME2	1,50	0,40	0,70	10	0,50
ME3a	1,00	0,40	0,70	15	0,50
ME3b	1,00	0,40	0,60	15	0,50
ME3c	1,00	0,40	0,50	15	0,50
ME4a	0,75	0,40	0,60	15	0,50
ME4b	0,75	0,40	0,50	15	0,50
ME5	0,50	0,35	0,40	15	0,50
ME6	0,30	0,35	0,40	15	Sense requisits

(1) Els nivells de la taula són valors mínims en servei amb manteniment de la instal·lació d'enllumenat, excepte (TI), que són valors màxims inicials. A fi de mantenir aquests nivells de servei, s'ha de considerar un factor de manteniment (f_m) elevat que depèn de la làmpada adoptada, tipus de llum, grau de contaminació de l'aire i modalitat de manteniment preventiu.

(2) Quan s'utilitzin fonts de llum de baixa luminància (làmpades fluorescents i de vapor de sodi a baixa pressió), es pot permetre un augment del 5% de l'increment llindar (TI).

(3) La relació entorn SR s'ha d'aplicar en les vies de trànsit rodat on no hi hagi altres àrees contigües a la calçada que tinguin els seus propis requisits. L'amplada de les bandes adjacents per a la relació entorn SR ha de ser igual com a mínim a la d'un carril de trànsit, i es recomana, si és possible, 5 m d'amplada.

(4) Els valors de luminància donats es poden convertir en valors d'il·luminació, multiplicant els primers pel coeficient R (segons CIE) del paviment utilitzat, i prenen un valor de 15 quan aquest no es conegui.

Sèries ME de classe d'enllumenat per a vials secs de tipus A i B

La instrucció tècnica segueix amb un apartat que també ens afecta directament en el disseny de la solució. Després de determinar aquests paràmetres, obri una nova secció anomenada enllumenats específics.

Segons la normativa, es consideren enllumenats específics els llums que corresponen a passarel·les de vianants, escales i rampes, passos subterranis de vianants, enllumenat addicional de passos de vianants, parcs i jardins, passos a nivell de ferrocarril, cul-de-sacs, glorietes, túnels i passos inferiors, aparcaments de vehicles a l'aire lliure i àrees de treball exteriors, així com qualsevol altre que puga assimilar a aquests.

Cada un d'aquests llums té assignada una classe d'enllumenat específic. Per exemple, la llum d'un parc (vials d'accessos, glorietes etc) se li assigna una il·luminació de classe E.

En resum, cada farola està regulada per una política que defineixen els paràmetres d'il·luminació en funció, o bé del rol que juga la farola en la ciutat, o bé pel lloc -tipus de carretera- on il·lumina.

En certa forma, la definició de la normativa està estretament lligada a les limitacions que hi ha en la infraestructura. Si compten, generalitzant, amb infraestructures rígides, les normatives a les que estan sotmeses seran rígides també.

Actualment, i el decret 1890/2008 és la mostra de que hi ha intenció, s'ha millorat la eficiència energètica. En aquest decret es parla de l'obligatorietat de que existisca a la xarxa un rellotge astronòmic per a engegar o aturar la xarxa quan s'estiga entrant a la nit. És un pas substancial a la situació prèvia on s'engegaven les llums a una hora que, en funció de quin mes i quin dia fóra, no era necessari.

Però, malgrat això, hi han alguns punts a millorar en aquest escenari:

- **Rigidesa de les polítiques**: Una farola només té un comportament durant el servei que ofereix. No sempre ha de ser així. Una farola pot estar actuant com "il·luminació activa" en certs moments. En altres, com a "il·luminació presencial"...
  Per tant, hem de cercar una solució que permeta assignar una llista de polítiques a una farola i no només una política única.

- **Flexibilitat de les polítiques**: Millorar les polítiques per a que s'apliquen en funció de condicions donades a temps real. Per exemple, que una política s'aplique si hi ha un vehicle passant prop, o si hi ha certes condicions climatològiques...

- **Visualitzar l'estat de les faroles**: La possibilitat de que la infraestructura permitisca un panel per poder veure, a temps real, l'estat de les faroles i les polítiques que s'estan aplicant en cada moment.

- **Facilitat de modificar el comportament**: Poder, en el mateix panel que he citat al punt anterior, seleccionar una farola o una carretera i modificar les polítiques que s'estan aplicant, afegir de noves o eliminar altres sense que això implique estar físicament en la instal·lació de les faroles.

Per portar a terme aquests canvis cal afegir les capacitats tècniques a la infraestructura per poder realitzar les següents accions:

- Crear, per a cada farola, un llistat de polítiques.
- Definir un ordre de prioritat en el llistat de polítiques que s'apliquen a cada farola.
- Manar ordres a la farola, en funció del resultat d'aplicar el llistat de polítiques de cadascuna.
- Rebre informació de l'estat de la farola.

Com a resultat d'aquesta anàlisi, vaig dissenyar el següent diagrama de classes que representa els conceptes del domini:

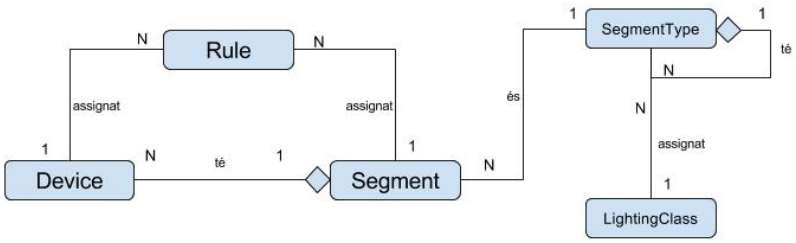


Diagrama del domini

En l'anterior diagrama de domini, tenim les següents entitats:

- **Device**: Representa un dispositiu lumínic. Generalment faroles.
- **Segment**: Representa una divisió de la carretera de mida variable.
- **Rule**: Representa una política.
- **SegmentType**: Representa un tipus de segment segons la normativa.
- **LightingClass**: Representa un tipus - paràmetres- d'il·luminació - segons la normativa.

Quan a les relacions, un dispositiu està relacionat en un segment i, a més a més, té assignades unes polítiques. El segment, a l'hora, també té assignades unes polítiques i és d'un tipus de segment. Els tipus de segment tenen una relació recíproca perquè hi ha subtipus de segments. I finalment, les classes d'enllumenat s'assignen als tipus de segments.

4. Disseny de la infraestructura

Per obtenir la solució que estem cercant, necessitem inicialment realitzar un canvi en la infraestructura actual d'una xarxa de faroles. Hem de tenir en compte que la capacitat actual d'una xarxa, com hem vist anteriorment, no permet una gestió de polítiques variada i flexible com la volem fer nosaltres.

La idea és que, al final, puguem definir un comportament raonablement complexe a les faroles a través de diferents polítiques per a que la farola realitze el mateix servei que està oferint ara mateixa però d'una forma més eficient.

Per aconseguir aquesta situació, hem de traslladar tota la lògica que hi ha en el quadre de comandament, instal·lat físicament en la xarxa, al núvol IoT. Aquesta transició té certes implicacions i moltes decisions que prendre generant diferents tipus de solucions.

En aquest procés de disseny de la solució vaig optar per seguir una metodologia cíclica de forma que proposava una solució, analitzava els beneficis i els contres de la solució i iniciava un altra iteració amb una nova proposta que complementava l'obtinguda amb millores. Totes les solucions que ara presentaré són igual de vàlides. Cada una és més completa que l'anterior. Però totes elles tenen en comú l'objectiu: Traslladar la lògica de les polítiques al núvol.

4.1 Solució base

La xarxa de carreteres està dotada de dispositius lumínics amb capacitat per enviar peticions i rebre a través d'una API. Quan una farola s'instal·la físicament a la carretera es donarà d'alta en el sistema enviant una petició a la API del servei d'altes i baixes del sistema (signin/signout).

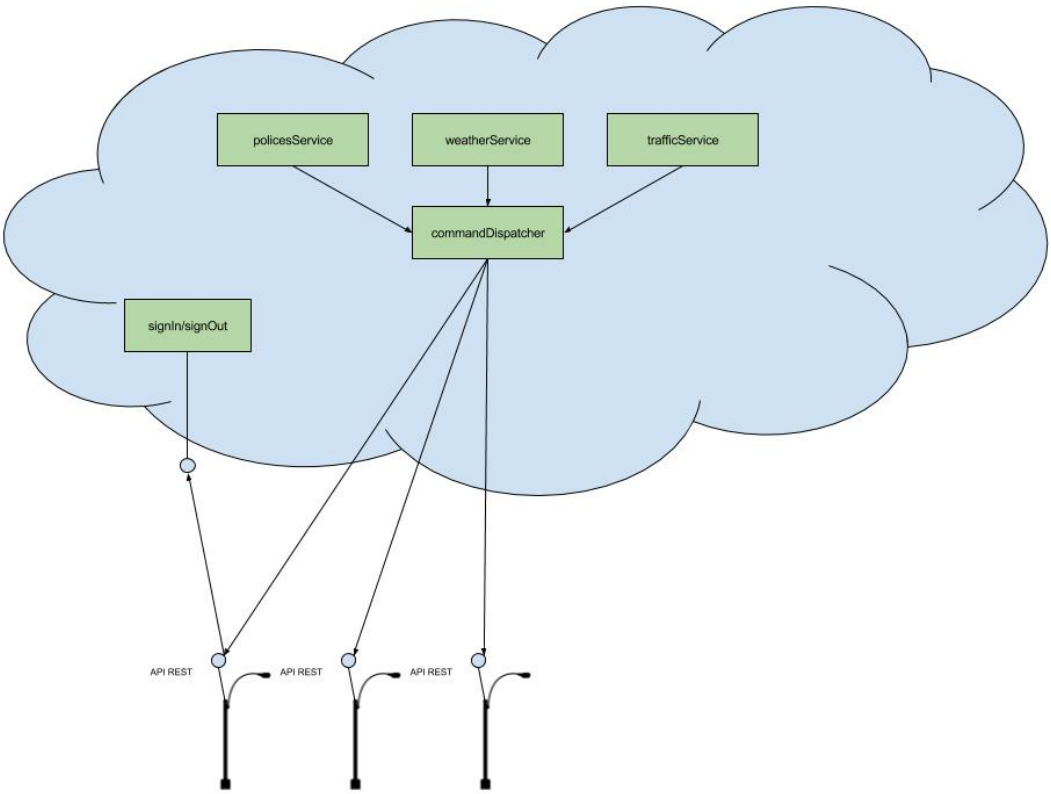
Aquest procés de petició d'alta serveix per a que el sistema d'altes en particular, però tot el sistema en general, obtinga dades importants de la farola:

- Endpoint de la farola
- Fabricant: Protocol de comunicació amb la farola
- Geolocalització: Dades importants per assignar-li polítiques en funció d'on il·lumina

La farola queda, aleshores, en espera de rebre ordres del servei encarregat d'enviar ordres: el *commandDispatcher*.

El commandDispatcher està sempre subscrit a tots els events que s'emeten dins del núvol *IoT* que afecten directa o indirectament a les polítiques: events de tràfic, events climatològics, hora actual...

Quan hi ha un event, per a cada segment i farola s'apliquen les polítiques tenint en compte les noves condicions de l'entorn. Aquest procés generà, al final, una ordre. Una ordre de que la farola s'ature, s'engegue, canvie la intensitat, el color... al cap i a la fi envia els paràmetres que abans enviava el quadre de comandament.



Solució base

Aquesta solució té els seus pros i contres que passe a descriure:

PROS	CONTRES
Infraestructura lògica senzilla.	La instal·lació del nou sistema és extremadament costós. Requereix que totes les faroles de la xarxa siguen intel·ligents
	Obligatorietat d'adquisició de faroles intel·ligents augmentat el cost d'implantació del sistema i el manteniment a llarg termini.
	Poc escalable. El commandDispatcher ha de manar ordres farola a farola.
	Delegació de responsabilitats inexistent. El commandDispatcher ha d'encarregar-se de localitzar cada dispositiu i enviar l'ordre de la forma que ho mana el fabricant.

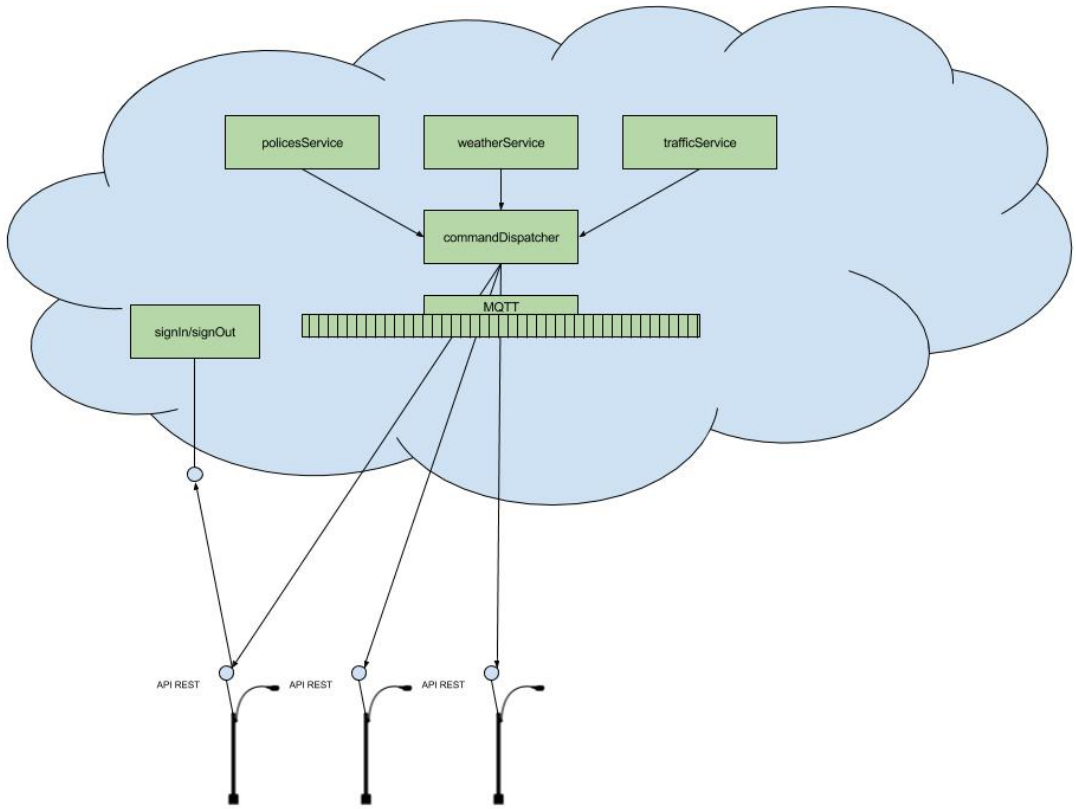
Taula de característiques de la solució base

4.2 Segona iteració: Millora de l'escalabilitat

Un dels punts desfavorables de l'anterior solució és que la infraestructura era poc escalable. El punt més crític està en la missatgeria que envia el commandDispatcher. S'ha de tenir en compte que ara mateix el commandDispatcher ha d'enviar a cada farola el missatge corresponent quan una política canvia. Si envia aquest missatge a 5 faroles, és clarament assumible. No ho és tant si ha d'enviar un missatge a milers i milers de faroles. Aquesta solució pretén solventar aquest problema.

La idea és fer servir un sistema de missatgeria que implemente el patró de missatgeria publicador/subscriptor com podria ser MQTT[20]. En aquest cas, el rol de publicador seria el commandDispatcher i els subscriptors serien les faroles.

Si la aplicació de les polítiques implica un canvi en el comportament de totes les faroles d'un segment, s'enviarà el missatge per el *exchange* del segment fent que l'ordre aplegue a totes les faroles del segment. Si, per altra banda, l'ordre va destinada a una única farola, s'emprarà *l'exchange* de la farola. D'aquesta forma, les demés faroles no se n'adonen de res.



Solució 2

PROS	CONTRES
Infraestructura lògica senzilla.	La instal·lació del nou sistema és extremadament costós. Requereix que totes les faroles de la xarxa siguen intel·ligents
Millora de l'escalabilitat en el commandDispatcher	Obligatorietat d'adquisició de faroles intel·ligents augmentant el cost d'implantació del sistema i el manteniment a llarg termini
	Delegació de responsabilitats inexistent. El commandDispatcher ha d'encarregar-se de localitzar els dispositius, manar l'ordre de la forma que ho mana el fabricant.

Taula de característiques de la solució 2

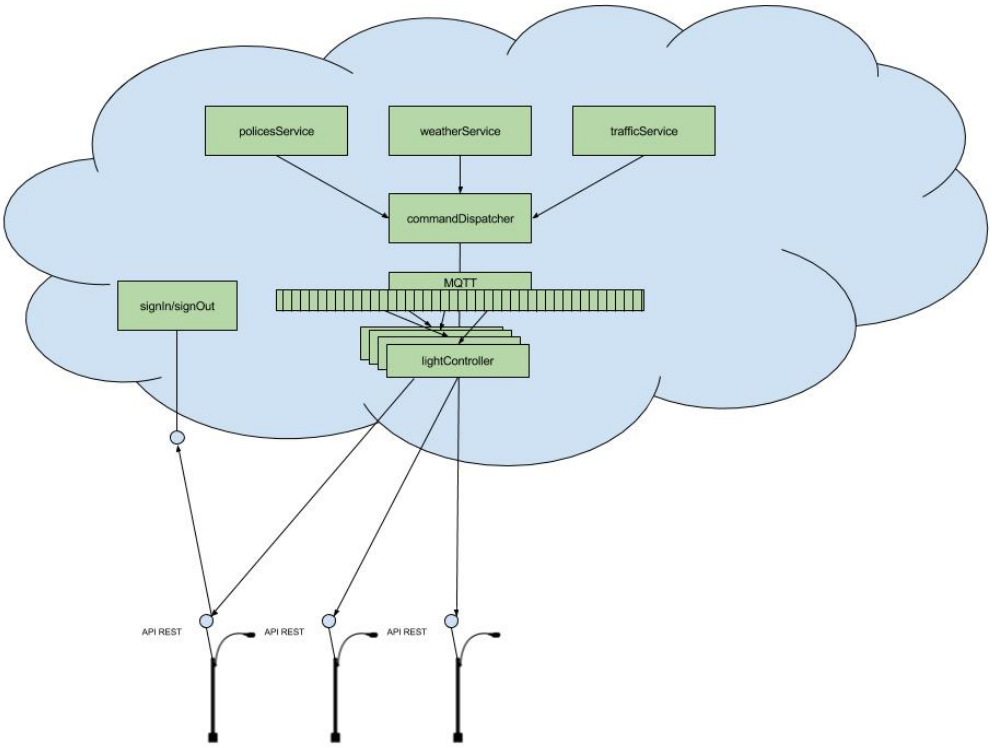
4.3 Tercera iteració: Delegació de responsabilitats

Per a afrontar el problema de la diferència de protocols en funció de les faroles vist en les dues solucions anteriors, i per permetre una major compatibilitat entre els diferents tipus de dispositius, és necessaria la creació d'un controlador.

El controlador, anomenat *lightController* actuarà com un intermediari lògic entre el *commandDispatcher* (més bé, el MQTT del *commandDispatcher*) i el dispositiu físic.

Cada *lightController* estarà configurat per treballar sobre un protocol/especificació. Per tant, cada dispositiu que es done d'alta al sistema haurà d'especificar en quin protocol/especificació treballa i se li assignarà un *lightController* adient.

El *lightController* pot estar en el núvol com a servei per part del fabricant, o bé, pot estar físicament a la xarxa ja siga al cap de les faroles o en el quadre de comandament.



Solució 3

PROS	CONTRES
Infraestructura jerarquitzada.	Augmenta el cost dels recursos lògics
Millora de l'escalabilitat en el commandDispatcher	La instal·lació del nou sistema és extremadament costós. Requereix que totes les faroles de la xarxa siguen intel·ligents
Delegació de responsabilitats adient. El commandDispatcher ja no ha de tenir en compte el tipus de faroles i protocols	Obligatorietat d'adquisició de faroles intel·ligents augmentant el cost d'implantació del sistema i el manteniment a llarg termini.
Cada fabricant publica el seu lightController com si es tractés d'un controlador d'un component de maquinari	

Taula de característiques de la solució 3

4.4 Quarta iteració: Compatibilitat cap enrere

Totes les propostes anteriors tenen un requisit indispensable: les faroles han de poder:

- Enviar peticions a una API per a donar-se d'alta/baixa en un sistema
- Fer ús de geolocalització
- Subscriure's a una cua MQTT
- Processar missatges per aplicar ordres concretes

Fins ara totes les solucions donades suposen que la flota de faroles té la capacitat de fer-ho. Però la realitat és que actualitzar tota la flota de faroles és, de per sí,

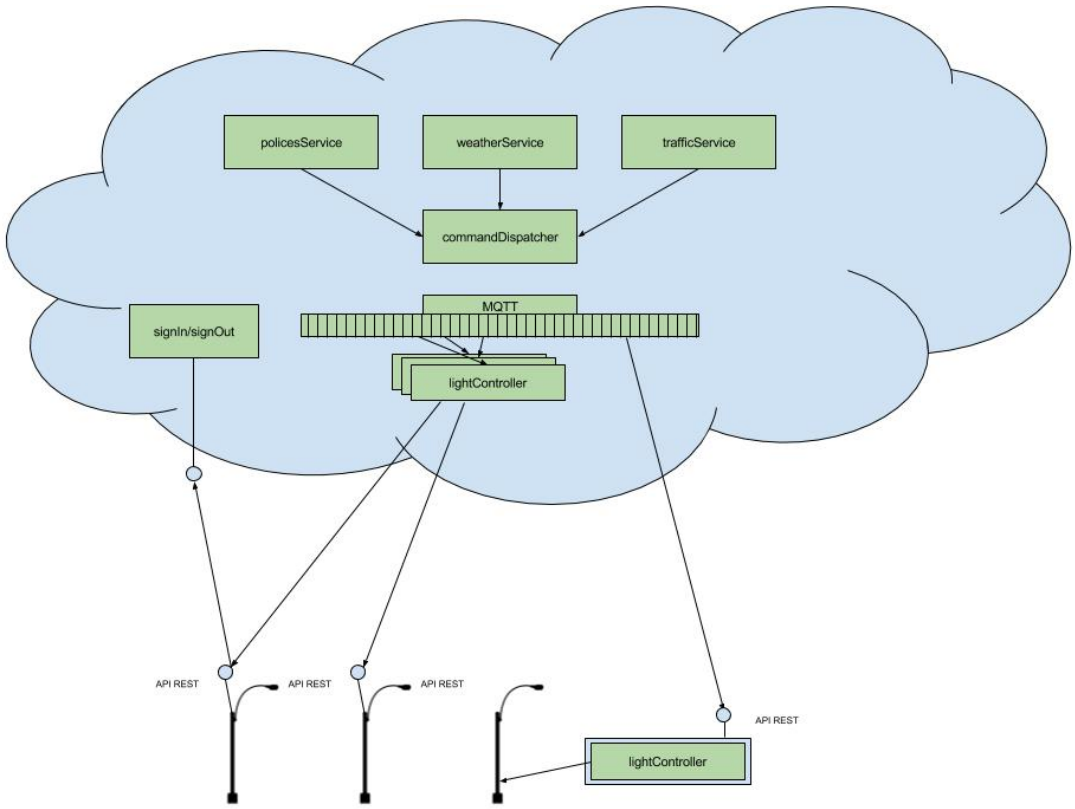
tremendament car. Però si, a més a més, els models que hem d'instal·lar són models "intel·ligents" el cost de l'actualització pot ascendir a preus astronòmics.

Per reutilitzar totes les faroles que ja tenim, aquesta solució proposa afegir un nou actor (físic-lògic) anomenat *'gateway'*.

Els *gateways* domòtics són dispositius físics situats estratègicament arreu de la xarxa de carreteres i que permeten als dispositius que no compleixen els requisits abans expostats rebre i enviar informació.

Els *gateways* seran un intermediari entre els *lightControllers* i els dispositius antics. Bàsicament traslladem la intel·ligència d'una farola al *gateway* (que serà qui envie les peticions o es subscriga a un lightController). I serà el *gateway* qui, al rebre una ordre, la transmeta al dispositiu.

Aquesta aproximació, a més, permet que el propi *gateway* tinga el lightController instal·lat en el propi dispositiu físic o servir un lightController en el núvol.



Solució 4

PROS	CONTRES
Infraestructura jerarquitzada.	Augmenta el cost dels recursos lògics
Millora de l'escalabilitat en el commandDispatcher	Cost dels gateways físics
Delegació de responsabilitats adient. El commandDispatcher ja no ha de tenir en compte el tipus de faroles i protocols	
Cada fabricant publica el seu lightController com si es tractés d'un controlador d'un component de maquinari	
Sistema compatible cap enrere. Implantació còmoda i menys costosa	

Taula de característiques de la solució 4

4.5 Patrons de disseny

Al final, cada solució ha anat emprant diferents patrons de disseny. Per una banda, he optat per solucions basades en microserveis en el núvol amb un sistema de missatgeria entre serveis. Aquesta forma d'abstraure responsabilitats fa el núvol més robust i cada servei més independent.

Per tal de mantenir una missatgeria massiva, he optat per sistemes de cues amb un patró de publicador subscriptor, doncs generalment l'escenari era un emissor i molts destinataris.

5. Disseny del sistema de polítiques

En aquest punt explicaré el nou sistema de polítiques fent ús de la nova capacitat de la infraestructura.

Un cop traslladades les polítiques de la xarxa física d'un quadre de comandaments al núvol, cal redissenyar les polítiques per a que puguem definir comportaments de les faroles més complexos.

Als inicis del projecte, quan sabia que anàvem a tractar l'eficiència energètica en la il·luminació de les carreteres intel·ligents vaig observar que les faroles tenien unes ordres molt simples: engegat a certa hora, redueix la teua intensitat a aquesta altra hora i aturat a aquesta altra.

El que volia aconseguir és que aquest comportament siguera la base per defecte del comportament d'una farola. I sobre això, anar definint comportaments excepcionals. L'objectiu és dir-li a una farola: Comporta't com les faroles del teu segment, però si passa una cosa "A" aplica un comportament "B".

Per aquest motiu, vaig començar a dissenyar un sistema basat en un llistat de polítiques ordenades per prioritats. Per una banda, les polítiques per defecte tindrien una prioritat baixa. Les polítiques de comportament excepcional tindrien una prioritat alta. Quan s'apliquen les polítiques ho farien aquelles que tenen una prioritat més alta.

Però això tenia un element contraproduent: És tediós definir tot el comportament d'una farola cada vegada que volem canviar un aspecte, per molt minúscul que siga. Per aquest motiu, vaig optar per utilitzar un sistema de llistat superposada de polítiques.

Si tenim unes polítiques per defecte que defineixen tots els paràmetres possibles d'una farola i anem superposant sobre aquesta política noves polítiques amb més prioritat que van alterant només uns paràmetres aconseguim que, un cop superposades totes les polítiques actives, el resultat de la superposició és un estat de la farola complet, on cada paràmetre ha estat modificat per una política concreta.

Però seguim tenint punts contraproduents. I si vull modificar el comportament de tot un segment de faroles? He d'anar farola per farola del segment definint el nou comportament? No és gens interessant a nivell d'usuari aquest disseny. Per això vaig crear els grups de prioritat.

Una farola té 4 grups de prioritat. Cada grup de prioritat té una llista de polítiques amb prioritat independent. Els grups de prioritat són:

1. Grup de preferència de polítiques de farola
2. Grup de preferència de polítiques de segment
3. Grup de polítiques de farola

4. Grup de polítiques del segment

Cada grup és una llista de polítiques ordenada per un valor de prioritat de l'1 al 100. En el moment de la superposició de les polítiques es fa en el següent ordre:

1. Grup de polítiques del segment
2. Grup de polítiques de farola
3. Grup de preferència de polítiques de segment
4. Grup de preferència de polítiques de farola

D'aquesta forma ens assegurem que sempre va a complir-se que una política de farola és més prioritària que una política del segment, però a l'hora menys prioritària que les polítiques de preferència.

Els grups de preferència són casos de polítiques excepcionals que volem assegurar-nos que tenen màxima prioritat. La idea és que habitualment aquest grup estiga buit i només s'afegisquen polítiques temporals en casos excepcionals. Per exemple: Hi ha un concert en un carrer i volem que les faroles estiguen engegades. És un cas d'una política de segment –doncs volem que tot el segment estiga il·luminat, però no obstant si una farola donada té una política de farola que indique el contrari, la farola no es veurà afectada per la nova política. Per evitar aquests casos, afegim la política en el grup de polítiques de preferència del segment i per tant la política de la farola es veurà sobreescrita per la nova política del segment.

En definitiva, aquests grups estan per simplificar el sistema de prioritats de les polítiques.

Finalment, aquest seria el diagrama del resultat final aplicant l'última iteració de les solucions amb el nou sistema de polítiques:

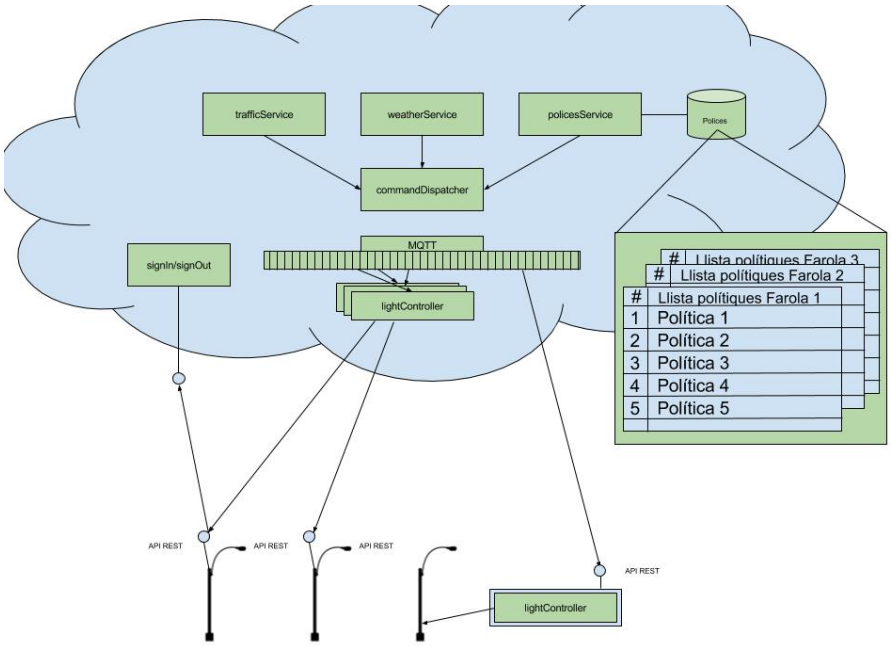


Diagrama final

5.1 El problema de quantificar la il·luminació d'una farola

L'objectiu, al cap i a la fi, que ací estem proposant, es tindre la capacitat de poder definir quina il·luminació volem en una carretera en determinades circumstàncies. I aparentment sembla trivial aquesta operació: Modifiquem la intensitat de la farola.

No obstant, no és tant simple. La il·luminació d'una carretera no ve determinada per la intensitat a la que està il·luminant una farola, sinó a un càlcul molt més complex. S'ha de tenir en compte que la il·luminació en un punt de la carretera ve determinada per la suma de les il·luminacions de les diferents faroles pròximes, el tipus de material de la carretera, la il·luminació ambiental i, fins i tot, la posició de la lluna (en el cas de que estiguem parlant d'un escenari on és de nit). Per tant, sent completament realistes, determinar una política només indicant un valor d´intensitat de la farola no s'adapta amb molta precisió a les necessitats reals de la carretera.

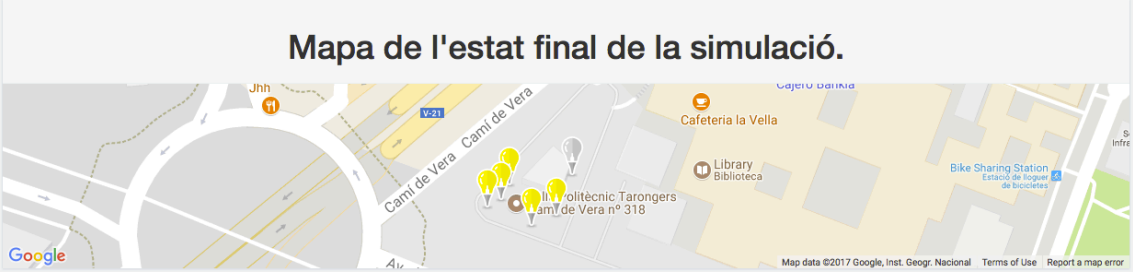
En aquest projecte s'ha simplificat el simulador perquè no era assumible, temporalment parlant, de tractar aquest tema.

Però una solució teòrica seria equipar les carreteres amb sensors situats estratègicament – això és, en posicions equidistants de les faroles- de forma que, a temps real, la xarxa sàpiga quina és la il·luminació de la carretera. Si dotem d'aquesta capacitat, les polítiques no haurien de definir el paràmetre d'intensitat de la farola sinó el paràmetre de "il·luminació" (lúmens) que ha de tenir la carretera. Com cada punt de la carretera tindrà probablement un valor diferent però aproximat –s'ha de tenir en compte que tot afecta a la il·luminació en un punt: faroles, lluna, edificis prop, ombres….- caldrà treballar en rangs de lúmens .

D'aquesta forma, si el sistema detecta que la carretera no té el valor mínim d'il·luminació, ha d'augmentar la intensitat de les faroles. En el cas contrari, si la carretera està sobre-il·luminada respecte al rang definit per la política aplicada, aleshores el sistema ha de reduir les intensitats de les faroles.

6. Implementació del simulador

Després de dissenyar com funcionarien les polítiques a la nostra nova infraestructura vaig optar per desenvolupar un simulador que demostrés aquest nou sistema en funcionament.



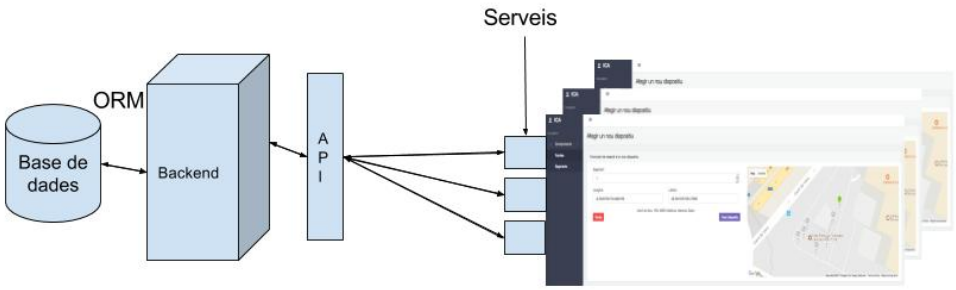
Mapa d'una simulació

En aquest punt vaig a presentar les diferents parts que composa el simulador per a després, en els pròxims capítols, entrar en més precisió en cada un dels components.

6.1 Arquitectura del programari

Per al desenvolupament del simulador vaig optar per dividir l'aplicació en dos components independents:

- Un *backend* que s'encarregaria d'oferir els diferents recursos (faroles, segments i polítiques) i persistir-los a una base de dades[21].
- Una aplicació client on mostraria la informació adient i portaria a terme la simulació. Aquesta aplicació del client estaria orientada a serveis que s'encarregarien de fer peticions a l'API per emmagatzemar o sol·licitar els diferents recursos de l'aplicació.



6.2 Entitats de l'aplicació

Les entitats de l'aplicació són emmagatzemades en el *backend* i a través d'un API són sol·licitades per la aplicació.

6.2.1 Device

És la classe que representa una farola. Amb la intenció de que aquest projecte no es limités únicament a faroles sinó que també es poguera adaptar, en un futur, amb altres dispositius - com poden ser, per exemple, senyals de trànsit basats en LED-, vaig optar per nombrar a la classe Device. Amb aquest nom genèric es dóna peu a poder fer servir diferents tipus de dispositius.

Camps de la classe:

- **serial_id**: Valor únic per identificar una farola i fabricant
- **longitude**: Longitud on està ubicat el dispositiu
- **latitude**: Latitud on està ubicat el dispositiu
- **address**: Direcció física del dispositiu
- **intensity**: Intensitat a la que està emetent la farola actualment. S'ha de tenir en compte la simplificació que s'ha fet respecte a la intensitat comentada en el punt 6.1

6.2.2 Segment

Una carretera pot tenir una extensió variable. Probablement la carretera que creua el nostre barri no serà tan gran com una autopista. Les autopistes, per regla general, tenen una extensió molt llarga i durant aquest recorregut els requisits d'il·luminació poden variar. És per això que vaig optar per dividir les carreteres per segments. Un segment és una divisió de la carretera variable que té com a objectiu agrupar faroles que tindran un comportament similar. D'aquesta forma es poden assignar polítiques a un segment i totes les faroles del segment heretaran aquestes polítiques.

Com hem vist al capítol anterior, a banda de les polítiques del segment, existeixen polítiques de la farola. Això ens permet determinar el comportament específic de la farola, per sobre el comportament del segment, malgrat que la farola pertanya al segment.

Camps de la classe:

- **segment_id**: Identificador del segment
- **min_lumens**: Tot i que el simulador no ho gasta perquè no hem aplegat a aquest nivell de detall, inicialment sí que vaig afegir aquests valors. Com hem comentat, cada segment pertany a una carretera i, com hem vist en la normativa, cada carretera segons el tipus de carretera té uns paràmetres d'il·luminació. Aquests paràmetres es resumeixen en dictar una franja de lúmens. Aquest camp defineix quin és la il·luminació mínima permesa en el segment.
- **max_lumens**: Aquest camp defineix quin és la il·luminació màxima permesa en el segment.

6.2.3 SegmentType

Representa els diferents tipus de segments que hi ha. Cada tipus de segment té uns paràmetres i valors lumínics que deriven en una llista de polítiques per defecte. Cada segment té assignat un segmentType i és aquesta associació qui aporta les polítiques per defecte al segment.

Camps de la classe:

- **name**: Nom del tipus de Segment
- **velocity_min**: Velocitat mínima a la que els vehicles poden circular (s'ha de tenir en compte que segons la normativa els tipus de carreteres es classifiquen inicialment per la velocitat de circulació).
- **velocity_max**: Velocitat màxima a la que els vehicles poden circular (s'ha de tenir en compte que segons la normativa els tipus de carreteres es classifiquen inicialment per la velocitat de circulació).
- **min_lumens**: Valor mínim de lúmens que pot emetre una farola del segment assignat a aquest tipus de segment.
- **max_lumens**: Valor màxim de lúmens que pot emetre una farola del segment assignat a aquest tipus de segment.

6.2.4 Rule

Representa una política. Aquesta política és assignable tant a una farola com a un segment.

Camps de la classe:

- **preference**: Indica si la política pertany al grup de preferència
- **priority**: Prioritat de la política
- **inherit_hibernate**: Indica si el camp d'hibernació està definit o és heretat de la política prioritàriament inferior en la superposició de polítiques.
- **allow_hibernate**: Si la hibernació no es heretada, indica si se li permet hibernar a la farola o no.
- **total_hibernate**: Si la hibernació no és heretada i es permet hibernar, indica si es pot hibernar totalment. És a dir, si pot aturar-se la farola quan no hi haja activitat al carrer que il·lumina.
- **delay_hibernate**: Indica el temps d'espera fins entrar en estat d'hibernació.
- **inherit_intensity**: Indica si el camp d'intensitat és heretat.
- **intensity**: Si el camp d'intensitat no està heretat, indica la intensitat que ha d'assumir la farola.
- **intensity_value**: Indica si el valor d'intensitat és un valor relatiu o absolut.
- **intensity_sign**: Si el valor de la intensitat és absolut, indica si l'increment és positiu o negatiu.
- **enabled**: Indica si la política està activada
- **inherit_schedule**: Indica si l'horari d'activació de la política s'hereta de la política prioritàriament inferior en la superposició de polítiques
- **start_rule_hours**: Si no s'hereta l'horari, hora a que s'activarà la política
- **start_rule_minutes**: Si no s'hereta l'horari, minuts als que s'activarà la política
- **end_rule_hours**: Si no s'hereta l'horari, hora a la que es desactivarà la política
- **end_rule_minutes**: Si no s'hereta l'horari, minuts als que es desactivarà la política
- **inherit_weather**: Indica si els paràmetres referents a les condicions climatològiques s'hereten de la política prioritàriament inferior en la superposició de polítiques.
- **raining**: Si no s'hereta, indica si s'activa quan plou, quan no plou o indiferència.
- **fog**: Si no s'hereta, indica si s'activa quan hi ha boira, quan no hi ha o indiferent.

6.3 Disseny de la Base de Dades

Per a la Base de dades vaig optar per fer servir SQLite. Per a estats inicials del desenvolupament és la forma més fàcil d'emprar una base de dades SQL sense haver de configurar cap servidor.

Aquest és el diagrama de la Base de Dades que vaig implementar:

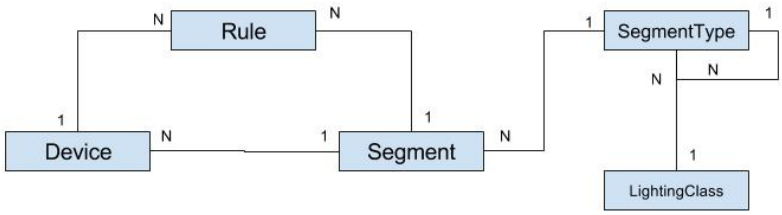


Diagrama de la base de dades

- **Device**: Representa un dispositiu lumínic. Generalment faroles.
- **Segment**: Representa una divisió de la carretera de mida variable.
- **Rule**: Representa una política.
- **SegmentType**: Representa un tipus de segment segons la normativa.
- **LightingClass**: Representa un tipus – paràmetres- d'il·luminació segons la normativa.

Quan a les relacions, sabem que una farola té unes polítiques assignades per tant hi ha una relació 1 a N amb *Rule*. A més a més, un segment també té polítiques assignades – i que indirectament s'assignen a les faroles -, pel que també hi ha una relació 1 a N de Segment amb *Rule*.

A més, *SegmentType*, que representa el tipus de Segment estarà relacionat òbviament amb Segment i amb la classe d'enllumenat. Aquesta última determina les polítiques per defecte del tipus de segment i que, per tant, seran les polítiques per defecte del segment associat. Finalment, *SegmentType* també té una relació recíproca amb sí mateix degut a que existeixen subtipus en els tipus de segments.

7. Implementació del servidor

Com Ruby on Rails - framework que he gastat per desenvolupar la API- fa servir un ORM[22], no vaig haver de definir les taules a la base de dades. No obstant, aquesta definició ve implícita en el moment en que crees les classes que vas a fer servir a la API. A la definició de les classes es crea un fitxer, anomenat "*migration*", on es van definint també els canvis a l' esquema de la base de dades.

7.1 Definicions de les classes

7.1.1 Device (farola)

Definició de la classe

```
class Device < ApplicationRecord
  belongs_to :segment, optional: true
  has_many :rules, dependent: :destroy

  validates :serial_id, uniqueness: true, presence: true
  validates :min_lumens, numericality: true, allow_nil: true
  validates :max_lumens, numericality: true, allow_nil: true
  validates :lumens, numericality: true, allow_nil: true
  validate :lumens

  def lumens
    if self.max_lumens != nil && self.min_lumens != nil
      if self.max_lumens < self.min_lumens
        self.errors.add_to_base("min_lumens must be lower or
                                 equivalent than max_lumens")
      end
    end
  end
end
```

En la classe device vaig definir certes validacions d'integritat de la Base de Dades. En aquesta classe vaig validar:

- **serial_id**: Camp amb unicitat i no NULL (forçant a que sempre estiga present)
- **min_lumens**: Camp numèric que pot ser NULL
- **max_lumens**: Camp numèric que pot ser NULL
- **lumens**: Camp numèric que pot ser NULL

A més, vaig implementar una validació que comprova que el valor **max_lumens** siga major o igual a **min_lumens**.

Definició de l'esquema de la base de dades (migration)

```
class CreateDevices < ActiveRecord::Migration[5.0]
  def change
    create_table :devices do |t|
```

```
      t.string :serial_id, default: nil

      t.float :max_lumens, default: nil
      t.float :min_lumens, default: nil
      t.float :lumens, default: nil

      t.float :longitude, default: nil
      t.float :latitude, default: nil
      t.string :address, default: nil

      t.float :intensity, default: 0

      t.belongs_to :segment, index: true

      t.timestamps
    end
  end
end
```

A la definició de l'esquema, es defineixen els camps de la classe i les relacions que impliquen claus alienes en la classe. En aquest cas, definisc una relació un a molts amb segment (1 segment té N devices). Aquesta relació implica una clau aliena en la taula de devices i és per això que la definició es fa ací amb `t.belongs_to :segment, index: true`.

7.1.2 Segment

Definició de la classe

```
class Segment < ApplicationRecord
  has_many :devices, dependent: :nullify
  has_many :rules, dependent: :destroy

  validates :segment_id, presence: true, uniqueness: true
  validates :min_lumens, numericality: true, allow_nil: true
  validates :max_lumens, numericality: true, allow_nil: true
  validate :lumens

  def lumens
    if self.max_lumens != nil && self.min_lumens != nil
      if self.max_lumens < self.min_lumens
        self.errors.add_to_base("min_lumens must be lower or
                                 equivalent than max_lumens")
      end
    end
  end
end
```

En l'entitat device havíem definit una relació un a molts amb la classe segment. Això implicava una clau aliena en la taula de device. Això permet que des de l'àmbit d'una instància de device es puga accedir al segment associat.

En aquest cas també volem l'accés recíproc. Per això definim els `has_many` en la classe segment. D'aquesta forma, des de segment podem accedir al *array* de dispositius i de polítiques (rules).

El camp `dependent:` defineix el tipus de dependència de la relació. Aquest paràmetre regula què passa quan s'elimina un segment. En el cas del device, la clau aliena es posa a NULL. En el cas de les polítiques, s'elimina la política.

A més, definisc algunes validacions d'integritat de la Base de Dades:

- **segment_id**: Camp amb unicitat i no NULL (forçant a que sempre estiga present)
- **min_lumens**: Camp numèric que pot ser NULL
- **max_lumens**: Camp numèric que pot ser NULL

A més, vaig implementar una validació que comprova que el valor max_lumens siga major o igual a min_lumens.

Definició de l'esquema de la base de dades (migration)

```
class CreateSegments < ActiveRecord::Migration[5.0]
  def change
    create_table :segments do |t|

      t.string :segment_id, default: nil

      t.float :max_lumens, default: nil
      t.float :med_lumens, default: nil

      t.belongs_to :segment_type, index: true

      t.timestamps
    end
  end
end
```

A la definició de l'esquema, es defeneixen els camps de la classe i les relacions que impliquen claus alienes en la classe. En aquest cas, definisc una relació un a molts amb segmentType (1 segment té N devices). Aquesta relació implica una clau aliena en la taula de segment i és per això que la definició es fa ací amb `t.belongs_to :segment_type, index: true`.

7.1.3 SegmentType

Definició de la classe

```
class SegmentType < ApplicationRecord
  has_many :child_types, class_name: "SegmentType",
                         foreign_key: "parent_id"
  belongs_to :parent_type, class_name: "SegmentType",
                                       optional: true
  has_many :segments

  validates :min_lumens, numericality: true, allow_nil: true
  validates :max_lumens, numericality: true, allow_nil: true
  validate :lumens

  def lumens
    if self.max_lumens != nil && self.min_lumens != nil
      if self.max_lumens < self.min_lumens
        self.errors.add_to_base("min_lumens must be lower or
                                 equivalent than max_lumens")
```

```
      end
    end
  end
end
```

En aquest cas, un tipus de segment s'han de validar:

- **min_lumens**: Camp numèric que pot ser NULL
- **max_lumens**: Camp numèric que pot ser NULL

A més, vaig implementar una validació que comprova que el valor max_lumens siga major o igual a min_lumens.

El camp `has_meny :child_types` fa referència a la reciprocitat d'una relació recíproca amb sí mateixa. Ho explicaré amb més detall en la definició de l'esquema de la base de dades.

Definició de l'esquema de la base de dades (migration)

```
class CreateSegmentTypes < ActiveRecord::Migration[5.0]
  def change
    create_table :segment_types do |t|

      t.string :name, default: nil
      t.integer :velocity_min, default: 0
      t.integer :velocity_max, default: 0

      t.float :med_lumens, default: nil
      t.float :min_lumens, default: nil

      t.belongs_to :lighting_class, index: true
      t.references :parent_type, index: true

      t.timestamps
    end
  end
end
```

Quan a la definició de l'esquema, es defeneixen tots els atributs de la classe.

Amb `t.belongs_to :lighting_class, index: true` definim la relació amb "classe d'enllumenat" que serà un a molts.

A més, aquesta classe es relaciona amb sí mateixa perquè existeixen subtipus de segment que pertanyen a un tipus de segment. Si ens fixem en la normativa actual, un segment de tipus "alta velocitat" després es divideix en altres tipus de vies en funció de la fisonomia de la via.

Aquesta relació amb sí mateixa té unes implicacions en la taula que s'especifiquen amb `t.references :parent_type, index: true`. Per a que aquesta associació siga recíproca en la definició de la classe també hi ha un `has_meny` d'aquesta associació.

7.1.4 Rule

Definició de la classe

```
class Rule < ApplicationRecord
  belongs_to :segment, optional: true
  belongs_to :device, optional: true

  validates :intensity, numericality: true, allow_nil: true
  validate :intensity_value_validation
  validate :intensity_sign_validation

  def intensity_value_validation
    if self.intensity_value != nil
      if self.intensity_value != "absolute" &&
         self.intensity_value != "percentual"
        self.errors.add_to_base("intensity_value must be
                                 [absolute|percentual|nil]")
      end
    end
  end


  def intensity_sign_validation
    if self.intensity_sign != nil
      if self.intensity_sign != "+" &&
         self.intensity_sign != "-"
        self.errors.add_to_base("intensity_sign must be [+|-
                                  |nil]")
      end
    end
  end
end
```

Fins ara només haviem definit relacions tipus un a molts. No obstant, la classe *rule* respecte a *device* i segment té una relació molts a molts. Això té certes implicacions a la base de dades. Per definir una relació molts a molts has de definir una taula nova que emmagatzeme aquestes relacions. Això ho veurem en el següent punt on definim l'esquema.

Però, a l'igual que passa amb les relacions un a molts bidireccionals, ací també volem que l'accés a l'entitat associada siga recíproc. Per això, a la definició de la classe especifiquem aquesta associació amb `belongs_to :segment, optional: true` i `belongs_to :device, optional: true`. Ambdues relacions són opcionals pel que pot existir, obviament, segments i dipositius sense polítiques.

En la classe rule vaig definir certes validacions d'integritat de la Base de Dades.En aquesta classe vaig validar

- **intensity**: Camp numèric que pot ser NULL

I vaig definir dues validacions que comproven el següent:

- **intensity_value**: Si no és NULL, que tinga el valor "absolute" o "percentual".
- **intensity_sign**: Si no és NULL, que tinga el valor "+" o "-".

Definició de l'esquema de la base de dades (migration)

```
class CreateRules < ActiveRecord::Migration[5.0]
  def change
    create_table :rules do |t|

      t.boolean :preference, default: false
      t.integer :priority, default: 100

      t.boolean :inherit_hibernate, default: true
      t.boolean :allow_hibernate, default: false
      t.boolean :total_hibernate, default: false
      t.integer :delay_hibernate, default: 0

      t.boolean :inherit_intensity, default: true
      t.float :intensity, default: nil
      t.string :intensity_value, default: "absolute"
      t.string :intensity_sign, default: "+"

      t.boolean :inherit_schedule, default: true
      t.integer :start_rule_hours, default: nil
      t.integer :start_rule_minutes, default: nil
      t.integer :end_rule_hours, default: nil
      t.integer :end_rule_minutes, default: nil

      t.boolean :inherit_weather, default: true
      t.boolean :raining, default: nil
      t.boolean :fog, default: nil

      t.boolean :enabled, default: true

      t.belongs_to :segment, index: true
      t.belongs_to :device, index: true

      t.timestamps
    end
  end
end
```

En la taula de polítiques, he definit tots els atributs i l'altre pas que mancava de la relació molts a molts amb `t.belongs_to :segment, index: true` i `t.belongs_to :device, index: true`.

En totes les definicions de taules de la base de dades hi ha un `t.timestamps`. El timestamps són dos camps de temps automàticament emmagatzemats en la base de dades que fan referència a la data de creació de l'entitat (created_at) i la data de l'última actualització (updated_at).

7.2 Controladors

Quan als controladors, i emprant la base del disseny de Ruby on Rails que diu que els controladors han de ser el més menuts possible, gairebé no tenen cap lògica important més enllà d'obtenir recursos de la Base de Dades i convertir-los a JSON per retornar-los com a resposta o bé agafar les dades de la petició i emmagatzemar-les a la base de dades.

Com a exemple, adjunte el Controlador de **device**:

```
class DevicesController < ApplicationController
  before_action :set_device, only: [
    :show,
    :update,
    :destroy,
    :rules]

  # GET /devices
  def index
    @devices = Device.all

    render json: @devices
  end

  # GET /devices/1
  def show
    render json: @device
  end

  # GET /devices/1/rules
  def rules
    @rules = Rule.where(device_id:params[:id]).or(Rule.where(segment_id:
@device.segment_id))
    render json: @rules
  end

  # POST /devices
  def create
    @device = Device.new(device_params)

    if @device.save
      render json: @device, status: :created, location: @device
    else
      render json: @device.errors, status:
                                  :unprocessable_entity
    end
  end

  # PATCH/PUT /devices/1
  def update
    if @device.update(device_params)
      render json: @device
    else
      render json: @device.errors, status:
                                  :unprocessable_entity
    end
  end

  # DELETE /devices/1
  def destroy
    @device.destroy
  end

  private
    # Use callbacks to share common setup or constraints between actions.
    def set_device
      @device = Device.find(params[:id])
    end

    # Only allow a trusted parameter "white list" through.
    def device_params
      params.require(:device).permit(
        :serial_id,
        :segment_id,
        :longitude,
        :latitude,
        :address
      )
    end
end
```


Amb `before_action :set_device, only: [:show, :update, :destroy, :rules]` el que instanciem és un *hook* que crida a *set_device* a l'inici de les funcions show, update, destroy i rules.

La funció `set_device` el que fa és obtenir el dispositiu que té la *id* que es passa com a paràmetre.

Com a detalls a destacar, la funció `def device_params`. Aquesta funció és una llista blanca de valors permesos en les peticions al controlador. Això filtrarà tots aquells valors que no s'esperen evitant així possibles atacs d'injecció.

7.3 Peticions del backend

```
Method  URI                                  CONTROLLER#ACTION
GET     /rules(.:format)                     rules#index
POST    /rules(.:format)                     rules#create
GET     /rules/:id(.:format)                 rules#show
PATCH   /rules/:id(.:format)                 rules#update
PUT     /rules/:id(.:format)                 rules#update
DELETE  /rules/:id(.:format)                 rules#destroy
GET     /lighting_classes(.:format)          lighting_classes#index
POST    /lighting_classes(.:format)          lighting_classes#create
GET     /lighting_classes/:id(.:format)      lighting_classes#show
PATCH   /lighting_classes/:id(.:format)      lighting_classes#update
PUT     /lighting_classes/:id(.:format)      lighting_classes#update
DELETE  /lighting_classes/:id(.:format)      lighting_classes#destroy
GET     /segment_types/parent(.:format)      segment_types#parent
GET     /segment_types/:id/children(.:format) segment_types#children
GET     /segment_types(.:format)             segment_types#index
POST    /segment_types(.:format)             segment_types#create
GET     /segment_types/:id(.:format)         segment_types#show
PATCH   /segment_types/:id(.:format)         segment_types#update
PUT     /segment_types/:id(.:format)         segment_types#update
DELETE  /segment_types/:id(.:format)         segment_types#destroy
GET     /segments(.:format)                  segments#index
POST    /segments(.:format)                  segments#create
GET     /segments/:id(.:format)              segments#show
PATCH   /segments/:id(.:format)              segments#update
PUT     /segments/:id(.:format)              segments#update
DELETE  /segments/:id(.:format)              segments#destroy
GET     /segments/:id/rules(.:format)        segments#rules
GET     /devices(.:format)                   devices#index
POST    /devices(.:format)                   devices#create
GET     /devices/:id(.:format)               devices#show
PATCH   /devices/:id(.:format)               devices#update
PUT     /devices/:id(.:format)               devices#update
DELETE  /devices/:id(.:format)               devices#destroy
GET     /devices/:id/rules(.:format)         devices#rules
```

En aquesta llista es veu les rutes del backend i quin controlador#mètode es llança a cada ruta.

8. Implementació del client

Per al desenvolupament del client vaig optar per fer servir AngularJS com a *framework* MVC[23]. AngularJS fa servir components web. Un component web és un *widget* que es pot reutilitzar permetent l'encapsulació i la interoperabilitat de diferents elements HTML. Les 4 principals característiques de l'ús de components són:

- **Elements personalitzats**: Creació d'una API per instanciar i crear elements HTML
- **Shadow DOM**: Cada component té encapsulat els seus DOM i estils.
- **HTML Imports**: Capacitat d'importar documents HTML dins d'altres documents HTML
- **HTML Tempates**: Permet crear *templates* que seran instanciats en temps d'execució del *javascript*.

8.1 Definició de classes

Per al client he creat gairebé totes les classes que ho he fet a la API. En aquest cas, només he necessitat crear 3 classes: device, rule i segment.

8.1.1 Device

```
export class Device {
  id: number;
  serial_id: string;
  longitude: number;
  latitude: number;
  address: string;
  segment_id: number;
}
```

8.1.2 Segment

```
import { Device } from '../models/device';

export class Segment {
  id: number;
  segment_id: string;
  devices_ids: number[];
  devices: Device[];
  segment_type_id: number;
}
```

8.1.3 Rule

```
export class Rule {
  id: number;
  preference: boolean;
  priority: number;

  inherit_hibernate: boolean;
  allow_hibernate: boolean;
  total_hibernate: boolean;
  delay_hibernate: number;

  inherit_intensity: boolean;
  intensity: number;
  intensity_value: string;
  intensity_sign: string;

  enabled: boolean;

  segment: Segment;
  segment_id: number;

  device: Device;
  device_id: number;

};
```

8.2 Definició de serveis

Per a cada recurs anterior, vaig crear un servei que seria el responsable de contactar amb la API per fer les peticions. En aquest cas, aquest és el servei de **Device**.

```
export class DeviceService {
  private endpoint: string = _config.api.url+'/devices';

  constructor(private http: Http) { }

  getDevices(): Observable<Device[]> {
    return this.http.get(this.endpoint)
                    .map(this.extractData)
                    .catch(this.handleError);
  }

  getDevice(id: number): Observable<Device> {
    let url: string = this.endpoint+'/'+id;

    return this.http.get(url)
                    .map(this.extractData)
                    .catch(this.handleError);
  }

  getDeviceRules(device: Device): Observable<Rule[]> {
    let id = device.id;
    var url: string = this.endpoint + '/'+id+'/rules';
    return this.http.get(url)
                    .map(this.extractData)
                    .catch(this.handleError);
  }

  createDevice(device: Device): Observable<Device> {
    let headers = new Headers({ 'Content-Type': 'application/json' });
    let options = new RequestOptions({ headers: headers });

    return this.http.post(this.endpoint, { device }, options)
                    .map(this.extractData)
                    .catch(this.handleError);
```

```
  }

  deleteDevice(device: Device): Observable<any> {
    let id = device.id;
    let url: string = this.endpoint + '/'+id;
    return this.http.delete(url)
                    .map(this.extractData)
                    .catch(this.handleError);
  }

  private extractData(res: Response) {
    let body = res.json();
    return body || { };
  }

  private handleError (error: Response | any) {
    // In a real world app, you might use a remote logging infrastructure
    let errMsg: string;
    if (error instanceof Response) {
      const body = error.json() || '';
      const err = body.error || JSON.stringify(body);
      errMsg = `${error.status} - ${error.statusText || ''} ${err}`;
    } else {
      errMsg = error.message ? error.message : error.toString();
    }
    console.error(errMsg);
    return Observable.throw(errMsg);
  }
}
```

En aquest servei, i cada servei que respon a una entitat del model de negoci (deviceService, segmentService i ruleService), bàsicament implementen la funcionalitat d'enviar una petició al backend de cadascuna de les operacions que es pot fer sobre la entitat al *backend*.

En aquest cas, s'implementa:

- `getDevices()`: Obtenir tots els devices de la base de dades
- `getDevice(id)`: Obtenir el device amb l'id indicat
- `createDevice(device)`: Crear el device a la base de dades amb els paràmetres de la classe device passada com argument.
- `deleteDevice(device)`: Esborrar el device passat per argument a la base de dades.

En els serveis faig servir `Observables`, per tal de garantir l'asincronisme que permet *javascript*. Quan una funció té una operació costosa (temporalment parlant), esperar a que aquesta funció acabe per obtenir el resultat seria bloquejant per a l'aplicació. Per evitar aquest problema, es pot retornar un *observable*. Amb aquest observable, hom pot subsciure un *callback* per a que s'execute el codi adient quan el resultat de l'operació costosa finalitze.

En aquest punt, sabem que, tot i que no hauria de ser molt costós, hi ha un retard degut a la latència de la xarxa entre que es fa la petició a la API i la API ens contesta. Si no gastàrem una aproximació asíncrona, el programa quedaria bloquejat durant l'execució d'aquesta petició.

Ens interessa que tot seguixca funcionant malgrat no tenir la resposta de la API i que una vegada tenim la resposta la API es mostre el resultat en la vista.

Per exemple, en el següent codi podem veure el mètode que fa la petició GET a la API per obtenir una farola concreta. Retorna un `Observable` de tipus Device:

```
getDevice(id: number): Observable<Device> {
  let url: string = this.endpoint+'/'+id;

  return this.http.get(url)
                  .map(this.extractData)
                  .catch(this.handleError);
}
```

El component que crida a aquest mètode, es subscriu al `observable` amb una funció (*callback*) que s'executarà un cop l'`observable` retorne un *device*.

```
getDevice(id: number): Observable<Device> {
this.route.params
    .switchMap((params: Params) => this.deviceService.getDevice(+params['id']))
    .subscribe(device => {
      this.device = device; //En aquest punt ja tenim un device
      ...
    });
```

8.3 Components

8.3.1 Component de mapa

Un dels components més reutilitzats en el client és el component de mapes. Aquest component és l'encarregat de:

1. Mostar un mapa
2. Aplicar els estils del mapa que s'indica en la instanciació del component
3. Mostrar els punts del mapa de la llista de punts indicats en la instanciació del component
4. Emetre events quan hi es fa *click* al mapa per a que els altres components puguen subscriure's.

Dades d'entrada del component

```
@Input() greenLights: {latitude: number, longitude: number}[] = [];
@Input() disabledLights: {latitude: number, longitude: number}[] = [];
@Input() offLights: {latitude: number, longitude: number}[] = [];
@Input() onLights: {latitude: number, longitude: number}[] = [];

@Input() mapPosition: {latitude: number, longitude: number} = {
  latitude: 39.3877689,
  longitude: -0.3755075
};
@Input() readOnly: boolean = true;
@Input() uiMapTypeControl: boolean = false;
@Output() clickMapEvent: EventEmitter<{
  latitude: number,
  longitude: number,
  address: string,
}> = new EventEmitter<{
  latitude: number,
  longitude: number,
  address: string,
}>();
```

- **@input() greenLights**: Vector de geoposicions. El mapa mostrarà una "bombeta verda" en el mapa.
- **@input() disabledLights**: Vector de geoposicions. El mapa mostrarà una "bombeta amb un simbol de prohibit" en el mapa.
- **@input() offLights**: Vector de geoposicions. El mapa mostrarà una "bombeta apagada" en el mapa.
- **@input() onLights**: Vector de geoposicions. El mapa mostrarà una "bombeta engegada" en el mapa.
- **@input() mapPosition**: Si s'indica, el mapa es centrarà en aquesta punt. Si no, es centrarà en funció de tots els punts geolocalitzats anteriors
- **@input() readOnly**: Si és *true* no s'emetrà cap event de que s'ha fet click sobre el mapa
- **@input() uiMapTypeControl**: Si és *true* apareixerà en la vista l'opció de poder canviar el tipus de mapa (satèl·lit, mapa de carretera etc)
- **@Output() clickMapEvent**: Event on subscriure's quan es fa click en el mapa.

Tots els valors anteriors d'entrada són opcionals. En funció de les dades d'entrada, es mostrarà una cosa o altra. La forma de cridar al component seria la següent:

```
<app-map [onLights]="resultMapMarks.on" [offLights]="resultMapMarks.off"></app-map>
```

En aquest cas, està cridada es fa en la vista del simulador quan aquest s'ha executat. En `resultMapMarks` es guarden les faroles en funció de si estan engegades (en `resultMapMarks.on`) o aturades (en `resultMapMarks.off`).

El component està sempre pendent de si alguna dada d'entrada ha canviat en temps d'execució i s'actualitza. D'aquesta forma, per exemple, si de sobte un punt

geolocalitzat passa de `offLight` a `onLights` - cosa que passa sovint quan es simula-, el canvi es veurà reflexat en el mapa immediatament. Aquesta característica s'ha aconseguit utilitzant els *hooks* disponibles dins del cicle de vida que té un component en AngularJS. En el nostre cas vaig implementar un mètode quan el component detecta un canvi en els paràmetres del component (`ngOnChanges`).

```
ngOnChanges(changes: SimpleChanges) {
    if(changes.mapPosition)
      this.mapPosition = changes.mapPosition.currentValue;

    if(changes.greenLights)
      this.greenLights = changes.greenLights.currentValue;

    if(changes.disabledLights)
      this.disabledLights = changes.disabledLights.currentValue;

    if(changes.offLights)
      this.offLights = changes.offLights.currentValue;

    if(changes.onLight)
      this.onLights = changes.onLights.currentValue;

    if(this.isReady){
      this.renderMap(); //Renderitzem de nou el mapa amb els nous canvis
    }
  }
```

Un dels problemes que vaig tindre en aquest component va ser a l'hora de renderitzar diversos instancies del component de mapa en una mateixa vista. Inesperadament, tots els components de mapa mostraven el mateix (el mateix mapa, mateixos punts, mateixa posició) quan clarament no havien de fer-ho.

Després de moltes hores de pegar-li voltes em vaig topar amb el problema. El mapa es situa en la vista HTML cercant un element amb un `id = map`:

```
this.map = new google.maps.Map(document.getElementById("map"))
```

En el template del component del mapa tenia el següent:

```
<div id="map" class="map"></div>
```

D'aquesta forma, allà on es cridara a aquest component, es renderitzaria el mapa dins del div amb `id="map"`.

Com el component es reutilitzava varies vegades en la mateixa vista, hi havien diversos elements HTML amb el mateix `id="map"`. Això feia que l'últim component en renderitzar el mapa ubicara el mateix mapa en tots els elements de mapa.

Per solventar el problema, vaig crear un id que es definira aleatòriament en temps d'execució. En el component de mapa definia un id aleatori:

```
id: string = this.randomString(10);
```

```
randomString(length: number): string {
    return Math.round((Math.pow(36, length + 1) - Math.random() * Math.pow(36,
                                                length))).toString(36).slice(1);
  }
```

I en el template del component del mapa feia un *binding* d'aquest valor amb l'atribut id de l'element HTML:

```
<div [attr.id]="id" class="map"></div>
```

D'aquesta forma, cada mapa tenia un id diferent i a l'hora de renderitzar no hi havien conflictes.

8.3.2 Component del simulador

Un altre component força important en el client és el component del simulador. És l'encarregat d'obtenir el segment a simular, les faroles del segment, la llista de polítiques del segment i la de les faroles. Un cop es simula, és l'encarregat d'aplicar les polítiques segons l'algorisme definit en la solució i mostrar els resultats.

Per llevar responsabilitats al component, el simulador compta amb un servei que realitza les operacions de simulació. D'aquesta forma el component només s'encarrega de mostrar els resultats.

El simulador treballa amb dues classes. Quan es realitza la simulació, cada política s'aplica o no, en funció de les condicions de la simulació. Si no plou, les regles marcades com que només s'apliquen si plou no s'aplicaran, per exemple.

Per tal de representar aquests resultats, he definit la classe **ResultRule**.

```
export class ResultRule {
  rule: Rule;
  active: boolean;
  observation: string;
};
```

Aquesta classe conté informació de si una política, `rule`, està activa (`active` a true). Si no ho està, `active` a false, el camp `observation` és un string que conté el motiu pel que aquesta política no s'ha executat.

8.3.2.1 Servei del simulador

El servei del simulador s'encarrega de les operacions darrere d'una simulació. Ordena les polítiques de cada farola en funció de la seua prioritat, tenint en compte els grups de prioritat, i aplica les polítiques per obtenir el resultat.

```
applyPolices(rules: Rule[], simulatorParameters: {enabled: boolean}): ResultStatus {
    var orderedRules: Rule[] = this.sortRulesByPriority(rules).reverse(); // Ordena les
polítiques per prioritat (de mínima a màxima prioritat)
    var intensity: number = 0.0;
    var result: ResultStatus = new ResultStatus();
    result.resultRules = new Array<ResultRule>();

    orderedRules.forEach(rule => {
      var active = false;
      var observation = null;
      var resultRule: ResultRule = new ResultRule;
      var br: boolean = false;

      resultRule.rule = rule;

      if(simulatorParameters.enabled){

        if(rule.enabled == false){
          resultRule.active = false;
          resultRule.observation = "Motiu: No habilitada";
          br = true;
        }

        if(!br && simulatorParameters.time === true){
          var simulatorTime = new Date()
          simulatorTime.setHours(simulatorParameters.hours, simulatorParameters.minutes,
0);
          var startTimeRule = new Date()
          startTimeRule.setHours(rule.start_rule_hours, rule.start_rule_minutes, 0);
          var endTimeRule = new Date()
          endTimeRule.setHours(rule.end_rule_hours, rule.end_rule_minutes, 0);

          if(startTimeRule > endTimeRule) {
            if(simulatorTime > endTimeRule && simulatorTime < startTimeRule){
              resultRule.active = false;
              resultRule.observation = "Motiu: Horari";
              br = true;
            }
          } else {
            if(simulatorTime > endTimeRule || simulatorTime < startTimeRule){
              resultRule.active = false;
              resultRule.observation = "Motiu: Horari";
              br = true;
            }
          }
        }

        if(!br && rule.raining !== null){
          if(simulatorParameters.raining !== rule.raining ){
            resultRule.active = false;
            resultRule.observation = "Motiu: Condició climàtica de pluja";
            br = true;
          }
        }

        if(!br && rule.fog !== null){
          if(simulatorParameters.fog !== rule.fog ){
            resultRule.active = false;
            resultRule.observation = "Motiu: Condició climàtica de boira";
            br = true;
          }
        }

        if(!br){
          resultRule.active = true;
```

```
          intensity = this.getRuleIntensity(resultRule, intensity);
          resultRule.observation = null;
        }
      } else {
        resultRule.active = true;
        resultRule.observation = null;
      }

      result.resultRules.push(resultRule);
    });


    result.resultRules = result.resultRules.reverse(); // El resultat el tornem ordenat
per priotat de màxima a mínima prioritat (i així veure les polítiques més prioritaries
al principi de la llista)
    result.intensity = intensity

    return result;
  }
```

La funció `applyPolices` obté com a entrada la llista de polítiques de cada farola del segment i retorna una llista de `ResultRule` (classe que he explicat anteriorment).

La funció itera sobre totes les polítiques i aplica la política analitzant cada camp si és compatible amb els paràmetres del simulador. Si alguna d'aquestes comprovacions que fa no es compatible amb el simulador (i per tant no s'aplica la política), la iteració salta i ignora la política.

Per exemple, en aquest segment de codi comprova si la política està habilitada. Si no ho està, el *flag* `br` canvia a *true* i les següents comprovacions no les fa.

```
if(rule.enabled == false){
          resultRule.active = false;
          resultRule.observation = "Motiu: No habilitada";
          br = true;
}
```

9. Exemple d'ús de l'aplicació

En aquesta secció mostraré com funciona l'aplicació del simulador. Per mostrar l'aplicació del simulador de polítiques en aquesta secció, crearé un segment, assignaré faroles al segment i assignaré una política per a que s'ature una d'elles en la simulació.

9.1 Operacions disponibles del simulador

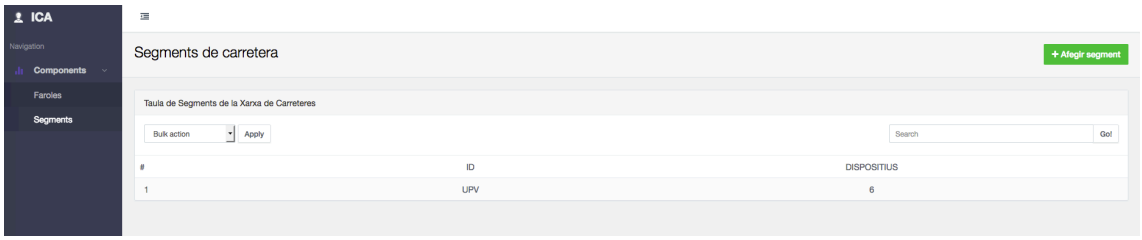
El simulador permet crear els elements necessaris per llançar una petita simulació que mostre un comportament de les faroles basat en polítiques tal i com hem definit a la solució.

Entre les operacions més importants:

- Dispositius
  - Crear dispositius
  - Crear polítiques
- Segment
  - Crear segment
  - Assignar faroles al segment
  - Crear polítiques
  - Simular segment

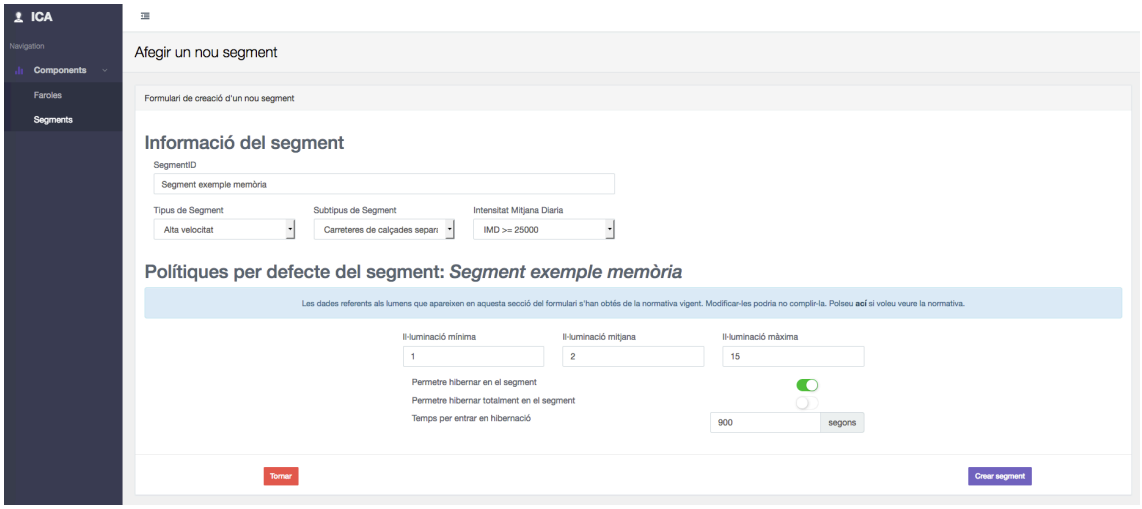
9.2 Creació del segment

Per a crear el segment, accedim a l'aplicació i anem a la secció de segments en el menú lateral esquerre.



En aquesta vista tenim una llista de segments donada d'alta. En aquest cas, només tenim un segment ja creat anomenat UPV amb 6 dispositius (faroles) assignats. Si polsem sobre un element de la llista, apareixerà una secció al lateral dret amb informació del segment (i la posició de les faroles del segment).

Per crear un segment, premem el botó superior verd que diu "Afegir segment". Ens carregarà un formulari.



En aquest formulari hem donat d'alta un segment d'una carretera d'Alta Velocitat que té carreteres de calçades separades amb encreuaments a diferent nivell i accessos controlats (autopistes i autovies). Aquesta classificació està extreta de la normativa vigent.

Al seleccionar aquest tipus de segment, es defineixen uns paràmetres per defecte de il·luminació mínima, màxima i mitjana (aquesta característica no està implementada del tot per qüestions de temps, com ja he comentat. Internament assigna un valor d'intensitat). Aquests valors creen un rang on la intensitat de la llum pot variar.

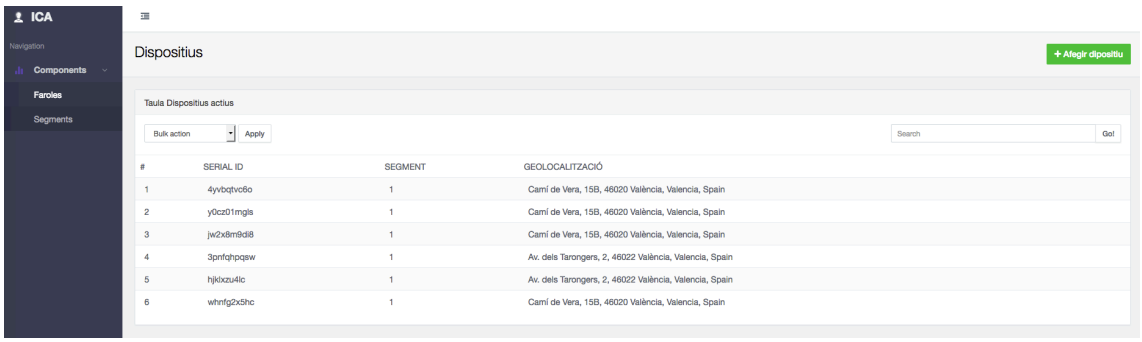
Al formulari, a més, hem marcat l'opció d'hibernar. Això vol dir que quan la farola veja que no hi ha activitat al carrer activarà un mode d'estalvi. Si l'opció d'hibernació total haguera estat marcada, s'aturaria totalment. Si aquesta opció no està marcada, es reduirà a la intensitat mínima seleccionada. Finalment, hem definit 15 minuts d'inactivitat per a entrar en mode hibernació.

Tots aquests paràmetres es tradueixen, en definitiva, en una llista de polítiques per defecte. La idea és que, per configuració, el sistema tinga una llista de polítiques per defecte per a cada tipus diferent de segment. I no només de segments on hi haja una carretera, també en el segments on hi haja un parc, o un monument etc... Al cap i a la fi, els que ens interessa és que la instal·lació d'un nou segment siga el més ràpid i menys tediós possible. I després, si volem un comportament específic del segment, aleshores ja sí, afegir polítiques personalitzades.

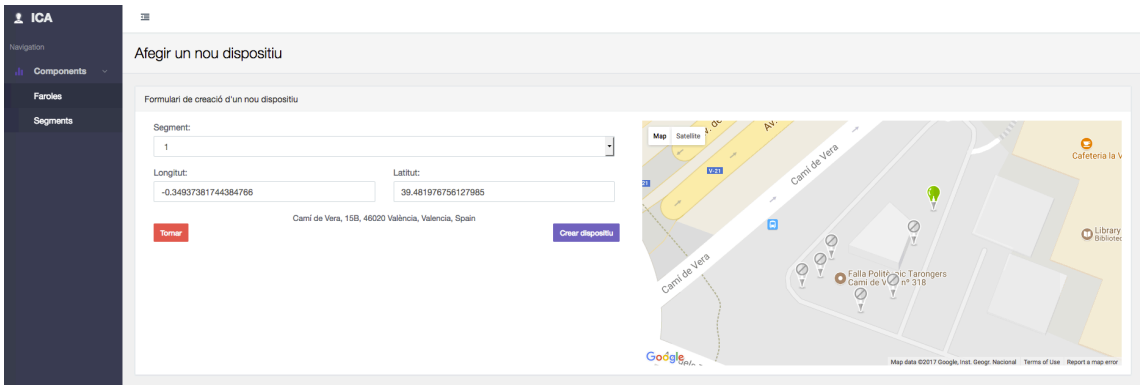
Si premem a Crear segment, tornarem a la llista de segments i podrem veure que s'ha creat el segment. En el meu cas li han assignat un ID 2.

9.3 Creació de faroles

Una vegada creat el segment, falta afegir faroles. Per tal, accedim a la secció de **faroles** en el menú lateral esquerre.



A l'igual que a la vista de segments, ací veiem una llista de dispositius que ja estan donats d'alta. Novament, si premem sobre el botó verd superior que diu "afegir dispositiu" accedirem al formulari de creació d'una nova farola.



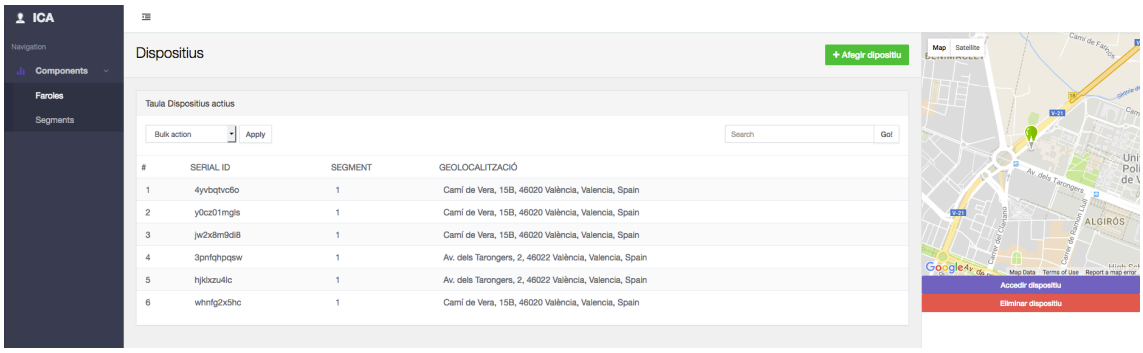
En aquest formulari, podem assignar la farola a un segment que tinguem donat d'alta. Un cop es selecciona el segment, el mapa del costat mostra les faroles que ja hi són donades d'alta en aquest segment (marcades amb una bombeta grisa).

Per indicar la posició de la bombeta podem o bé escriure els seus valors de geoposició als camps de longitud i latitud o bé fer *click* sobre el mapa i apareixerà una bombeta verda.

Un cop localitzada, premem "Crear dispositiu" per confirmar-ho. Això ens tornarà a la vista anterior on podrem veure la nova farola en la llista.

9.4 Definició d'una política de farola

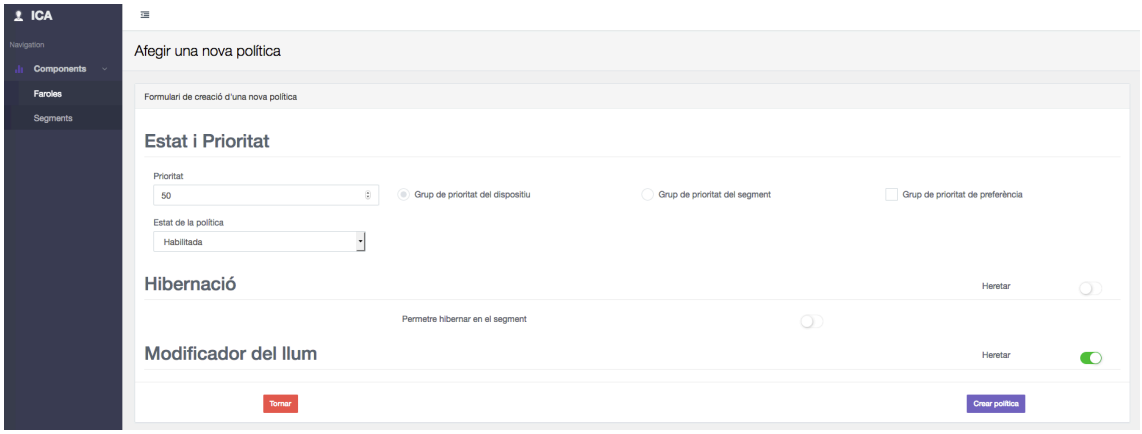
En la vista de llista de dispositius, si premem sobre un element de la llista, apareixerà una secció al lateral dret amb informació de la farola i podem accedir a un panel de configuració de la farola.



Quan es crea la farola, tal i com hem dissenyat en la solució, heretarà automàticament les polítiques del segment a la que pertany. Nosaltres podem modificar el comportament afegint noves polítiques a la farola que tenen una prioritat superior a les polítiques del segment.

Si volem crear una política específica a la farola, en aquest menú accedim a "Accedir al dispositiu", i en aquesta vista podem entrar a la secció de "Polítiques". En aquesta secció ens apareixerà una llista de polítiques. Aquestes polítiques són polítiques que s'han heretat del segment (doncs nosaltres encara no hem donat d'alta cap política a la farola). Ho podem fer ara prement el botó "Afegir una política".

Ens apareixerà un nou formulari per a crear la nova política:



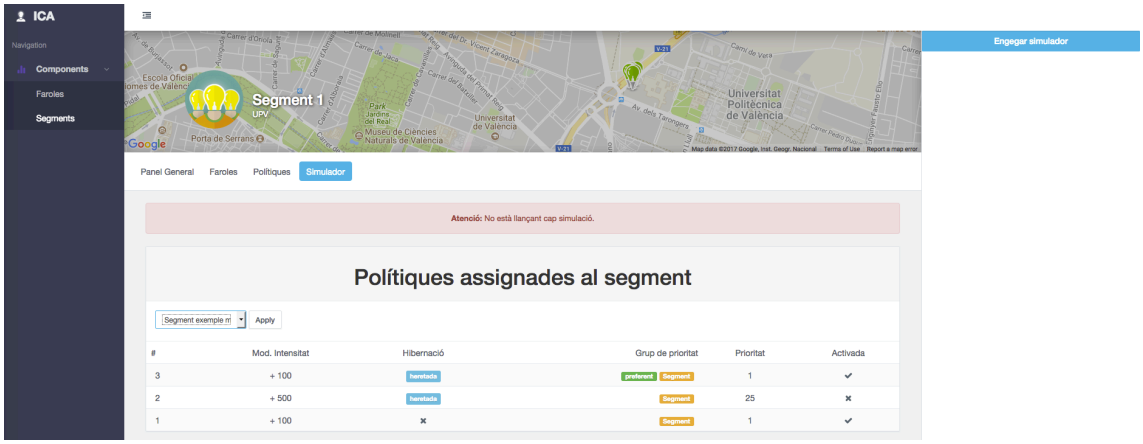
Podem definir la prioritat de la política (en aquest cas la política estarà al grup de prioritat de polítiques de la farola i no del segment), si la política és preferent, si està habilitada. També ens permet definir camps heretats (pel que no definirem el seu valor). En el nostre cas, anem a crear una política inhabilitada de prioritat 1 de preferència. En aquesta política modificarem la llum a 0 (pel que desactivem l'opció de que s'herete). Aquest política que anem a crear no s'aplicarà (doncs l'hem marcada com a inhabilitada).

Crearem un altra, de prioraitat 40 de preferència, la deixarem habilitada i en la secció de modificador de llum ficarem la intensitat a 0, pel que el resultat final és que la farola s'aturarà.

9.5 Simulació del segment

Si tornem a la llista de segments i premem el segment amb ID 1 que és en el que hem estat creant les faroles, apareixerà el menú de la dreta i podrem prémer el botó de "accedir al segment". En aquesta vista podem accedir a la secció del simulador en la pestanya de "Simulador".

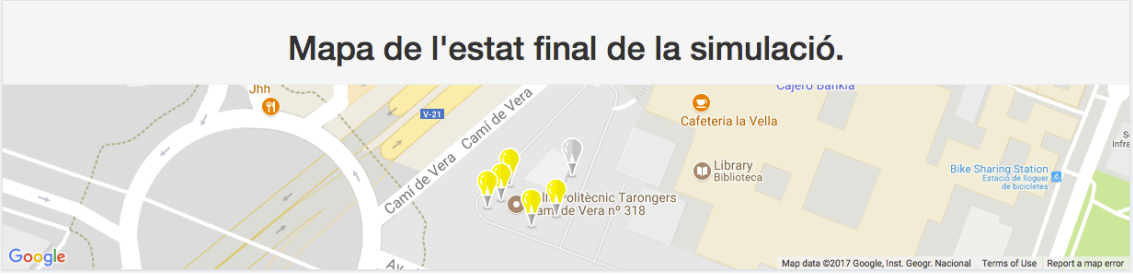
En aquesta vista tenim la llista de polítiques ordenada del segment.



Aquestes polítiques són les polítiques aplicades per defecte al segment i que totes les faroles tindran. En el selector que hi ha a sobre la llista que diu "Segment exemple memòria" ens permet canviar la llista per veure, també, la llista de polítiques de cada farola del segment. Podem comprovar que gairebé totes les faroles tenen la mateixa llista de polítiques a excepció de la farola on hem creat 2 polítiques específiques:

En aquest cas podem veure com les dues polítiques de dispositius que hem creat amb preferència estan al capdamunt de la llista i per tant tenen màxima prioritat.

Si premem sobre el botó "Engegar Simulador" del menú de la dreta, podrem veure quin és el resultat d'aplicar aquestes polítiques. Ens apareixerà un mapa on podem comprovar que, efectivament, l'esmentada farola roman aturada. Les altres, sotmeses a les polítiques del segment sí que hi són engegades.



A més a més, podem veure que a la llista abans esmentada podem veure quines polítiques no han entrat en la llista d'aplicació (perquè, en aquest cas, les hem marcades com a deshabilitades) doncs estan amb menys opacitat:

10. Conclusions

10.1 Valoració personal

El IoT i el món de les ciutats intel·ligents és un camp apassionant amb molt de futur. A més el considere part de la solució per tal de millorar l'impacte en el nostre entorn i l'eficiència energètica a les nostres ciutats, com podem veure en aquest projecte. Però no ens enganyem tampoc, aquesta tecnologia ha de ser element assistencial i no la solució.

Per tal d'aconseguir unes ciutats eficients hem de canviar la nostra societat de consum i els nostres hàbits. I després, recolzar-nos en la tecnologia. No podem continuar amb aquesta deriva consumista amb la justificació de que la tecnologia algun dia ens traurà d'aquesta.

Però no hi ha dubte que podem aconseguir nivells d'eficiència - i no només energètica, també parle d'eficiència als serveis públics a les ciutats, a les nostres cases, als nostres centres de treball...-, que sense aquesta tecnologia seria pràcticament impossibles d'assumir. I per això pense que és un camp interessant per explotar.

Ara bé, quan hom llig sobre el *big data*, el IoT, el *machine learning*, veu el potencial que pot tenir, però també com pot ser de contraproduent per a nosaltres.

En primer lloc, la gran capacitat per gestionar, processar i emmagatzemar dades de càmeres, senyals, comunicacions... posa en greu perill el dret a la privacitat. Si això ho sumem a que molta d'aquesta tecnologia està gestionada per empreses privades encara està en més perill. Crec que abans d'embarcar-nos amb aquestes tecnologies havíem de marcar uns límits clars i contundents de fins on i quines dades es poden recol·lectar i permetre que el propi usuari -i només ell- decidixca quines dades vol compartir. I que mai una tecnologia, per molt que millore el nostre dia a dia, estiga per davant dels nostres drets fonamentals.

En segon lloc, la dependència que poden crear aquestes tecnologies a les persones. Em preocupa que el fet de que la tecnologia treballe per nosaltres acabe per deixar-nos incapacitats per fer res. I això s'heretarà a les generacions venidores incrementant-se l'efecte. I que passa si un dia es col·lapsa tot? No sabrem fer res. De segur que això forma part de novel·les de ciència ficció però no ve gens malament reflexionar sobre això.

En tercer lloc, la falta de seguretat en el IoT. Han existit casos d'atacs sobre xarxes IoT amb èxit. Capacitat de modificar el comportament d'un dispositiu per una persona no autoritzada. O l'obtenció de dades de caràcter privat. S'ha d'anar amb cura amb això.

Respecte al projecte, valore positivament el treball realitzat. Tractar aquest món és interessant i apassionant. He après molt com està dissenyada una infraestructura IoT, els problemes que poden sorgir i com es relacionen tots els dispositius.

10.2 Dificultats i objectius no aconseguits

La dificultat afegida de realitzar el projecte a distància ha estat present sempre. No és gaire fàcil tractar temes de manera escrita i moltes hores es perden escrivint allò que es pot dir cara a cara en pocs segons. A més, interpretar allò escrit és més complicat que allò parlat. Així que una de les dificultats més importants ha estat aquesta.

Per altra banda, inicialment vaig decidir prototipar una infraestructura. Em va portar algun temps crear una solució base ja que volia ser particularment realista amb el prototip. No obstant, després de mostrar-li al meu tutor la solució base vam veure que no hi havia cap resultat interessant per presentar com a projecte. En aquest moment vaig optar per centrar-me en la gestió de polítiques i crear el simulador.

10.3 Experiència

El desenvolupament del simulador, i sobretot la part del client, s'ha realitzat amb tecnologia que pràcticament desconeixia. Per una banda, AngularJS que, malgrat haver sentit parlar d'aquest *framework* no havia tocat res. Va ser tot un repte aprendre a desenvolupar una plataforma amb els requisits que volia en tan poc temps, però ha valgut la pena. Pense que el simulador té el dinamisme que cercava i AngularJS és una nova tecnologia que pose a la motxilla. A més, tampoc havia treballat amb TypeScript i em sembla una millora substancial respecte a JavaScript pel que ha estat tot un encert utilitzar-la.

Com he comentat en punts anteriors, per mostrar el simulador al meu professor vaig haver de fer un *deploy* del simulador en un servidor en producció. Això ha implicat instal·lar diversos servidors web, proves i configuracions. Sense cap dubte, he obtés molta experiència interessant.

10.4 Treballs futurs

10.4.1 Millorar el simulador

El simulador, malgrat ser funcional, se li poden afegir més característiques. A banda de que alguns processos del simulador calen *refactoritzar*, pot ser no és del tot eficient. El simulador només permet simular un segment. És interessant poder simular tota una ciutat.

També seria interessant afegir més condicions d'execució (més climatologia, condicions de trànsit...) en la simulació.

10.4.2 Simulador real

El simulador podria adaptar-se per a que l'execució es realitzara sobre una maqueta amb faroles reals i no sobre un mapa. I qui sap, sobre un escenari de faroles reals.

10.4.3 Machine learning

Una de les deficiències que he explicat en la infraestructura actual és que no hi havia capacitat per definir un comportament de les faroles complex perquè només es podia definir unes poques polítiques simples a la xarxa. Un dels punts que ha desenvolupat aquest projecte ha estat ampliar la capacitat de la xarxa per a que cada farola tinga una llista de polítiques que defineixen un comportament més complex. I així, ampliar la complexitat del comportament de la farola tant en quant s'ampliava el nombre de polítiques.

Encara que semble paradoxal, pense que la solució definitiva és que, al final, no existisca cap política. I que siga la farola, a través del *machine learning* i el *big data* qui sàpiga en quin moment s'ha d'engegar i aturar i a quina intensitat fer-ho. I per tant, és la pròpia farola qui es defineix les seves polítiques.

11. Agraïments

Vull agrair, i de manera especial, el suport tècnic del meu tutor, Joan Fons i Cors, a l'hora de desenvolupar aquest projecte.

A Ignasi Nadal per l'assessorament en la matèria de l'enllumenat públic i els recursos fotogràfics.

Al meu germà, Pau Urios, per la iconografia del simulador.

A la meua família per l'ajuda emocional i econòmica, sempre al meu costat.

Als meus amics per acompanyar-me en aquest viatge.

12. Bibliografia

[1] Notícia d'Europapress - Contaminación lumínica: ¿qué riesgos tiene y cómo se puede evitar?:
http://www.europapress.es/sociedad/noticia-contaminacion-luminica-riesgos-tiene-puede-evitar-20170221132505.html

[2] Notícia El País - Valencia, capital europea en contaminación lumínica
https://elpais.com/diario/2009/05/10/cvalenciana/1241983078_850215.html

[3] Gitflow de GitHub
https://guides.github.com/introduction/flow/

[4] Git
https://git-scm.com

[5] Documentació de Markdown en GitLab
https://gitlab.com/help/user/markdown

[6] Atom
https://atom.io

[7] Postman
https://www.getpostman.com

[8] Nginx
http://nginx.org/

[9] Passenger
https://www.phusionpassenger.com/

[10] Digitalocean
https://www.digitalocean.com/

[11] Ruby
https://www.ruby-lang.org/en/

[12] Ruby on Rails
http://rubyonrails.org/

[13] HTML
https://developer.mozilla.org/en-US/docs/Web/HTML

[14] CSS
https://developer.mozilla.org/en-US/docs/Web/CSS

[15] TypeScript
http://www.typescriptlang.org/

[16] AngularJS
https://angular.io/

[17] JSON
http://json.org/

[18] SQLite
https://www.sqlite.org/

[19] Butlletí Oficial de l'Estat (Reial decret 1890/2008)
https://www.boe.es/boe_catalan/dias/2008/11/19/pdfs/BOE-A-2008-18634-C.pdf

[20] MQTT
http://mqtt.org/

[21] Ruby on Rails com a API
http://guides.rubyonrails.org/api_app.html

[22] ORM
http://guides.rubyonrails.org/active_record_basics.html

[23] MVC
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller