Desplegament del servei

Web i definició de protocols

Nil Blanca

Roger Tarres

Javier Delgado

Josep Marches

Héctor Montesinos

Índex…

[**Descripció general Sprint**](#_17dp8vu)2

[**Desplegament**](https://docs.google.com/document/d/18GOvh1pQIfcA4pKVHEFYUkSF3lFcm7FOBqWDVQa7C2M/edit#heading=h.1y810tw) **del servei web 2**

[Descripció general](#_26in1rg) 2

[Web](#_lnxbz9) Usuari 3

[Web](#_35nkun2) Administrador 4

[API](#_1ksv4uv) - Pagina Administrador 6

**Definició de protocols 10**

[Descripció general](#_z337ya) 11

[Explicació/Justificació](https://docs.google.com/document/d/18GOvh1pQIfcA4pKVHEFYUkSF3lFcm7FOBqWDVQa7C2M/edit#heading=h.4i7ojhp) 12

[Conclusions](#_3j2qqm3) 13

# 

# 

# 

# 

# 

# 

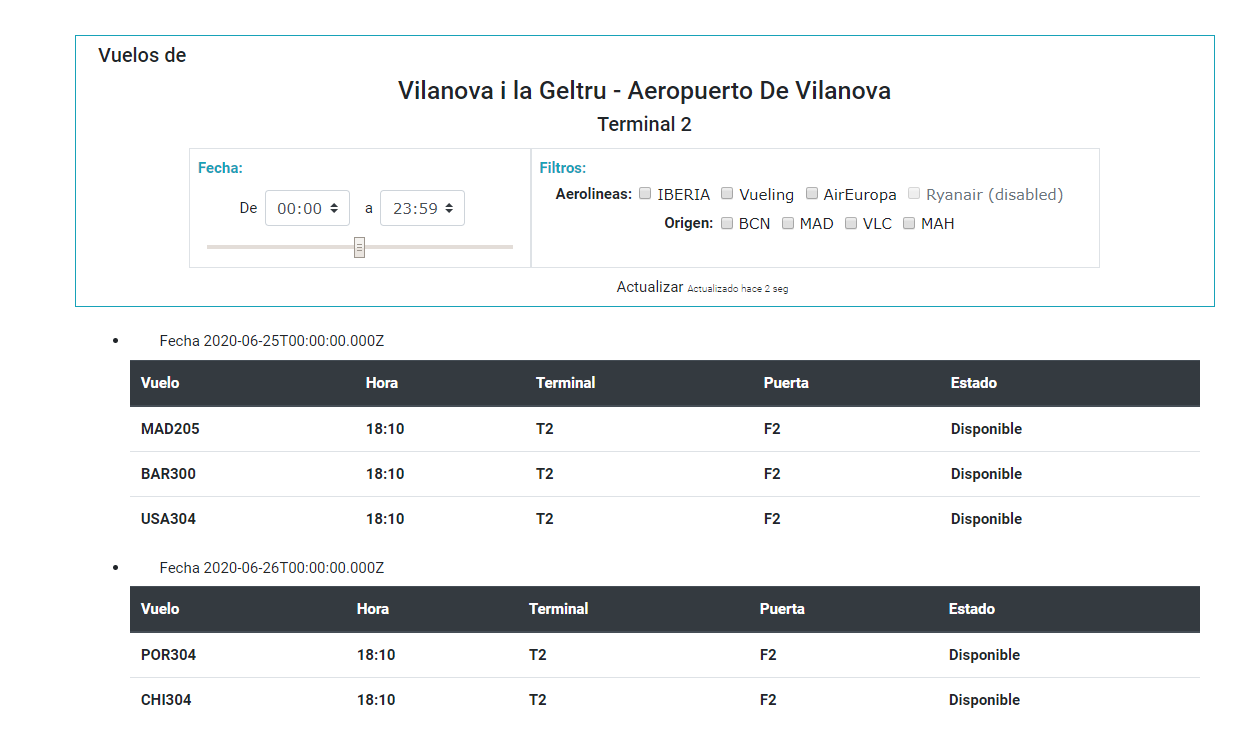
# Descripció general Sprint 3

//Descripció global de les tasques realitzades i l’organització del grup. Part del projecte realitzada, estructuració del treball...

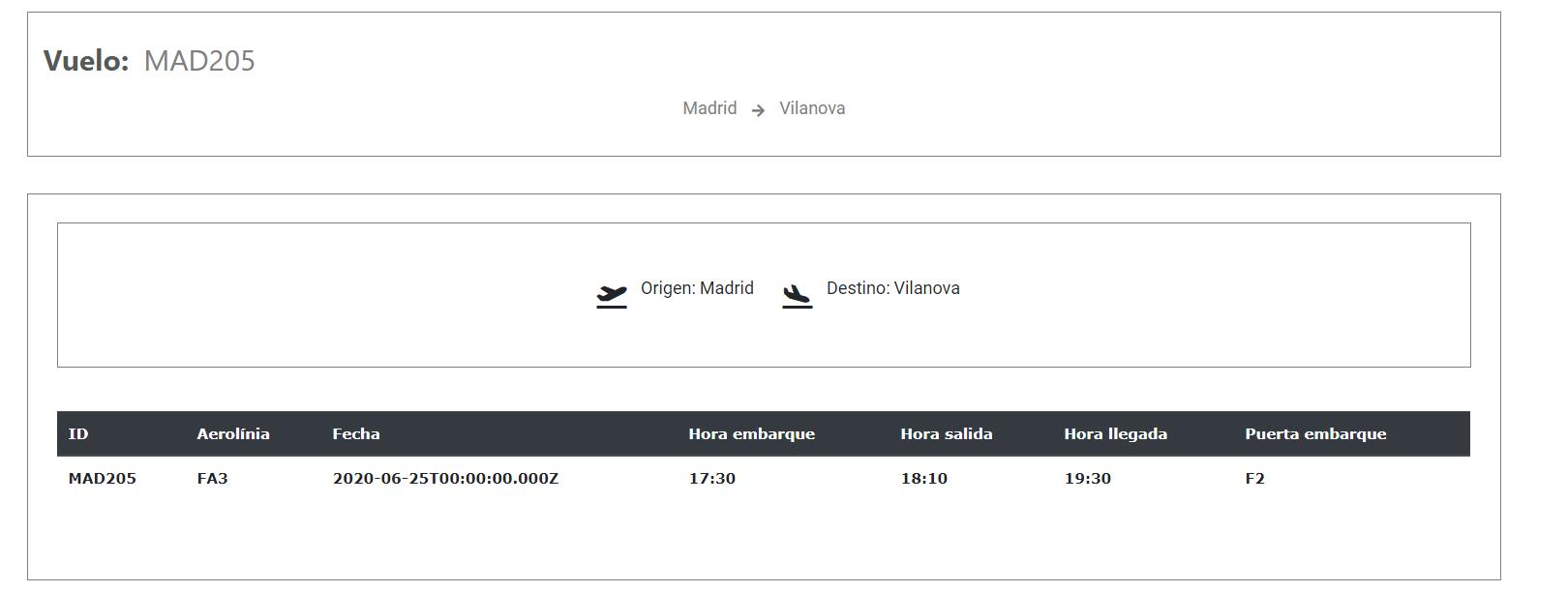
* El principal increment d’aquest sprint a sigut l'adaptació de l’API per al correcte funcionament amb tots els equips del projecte. S’han realitzat canvis com:
  + -Adaptar el socket en la API per connectar els cotxes.
  + -Hem adaptat els atributs de la API en funció de la necessitat de la APP.
  + -Reformat els atributs de Flights – vols en la API.
* L’increment de la web d’administrador es tracta de noves funcionalitats.
  + -Ja es pot interactuar més amb al BD: afegir, modificar i eliminar des de la pagina d’administrador.
  + -Es poden veure el cotxes temps real des de la pagina d’administrador.
  + -S’ha fet una millora en el tocken.
  + -Es pot definir un treballador “jefe” que pot gestionar a la resta de treballadors que poden accedir a la pagina d’administrador.
* L’increment de la web d’usuari es centra en que comenci a interactuar amb la API fent peticions. Es mostren els vols de que tinguem en la BD als usuaris.

**-Web Usuari:**

Pagina de vols mostrant els vols per dies:



Vol mostrat en detall:

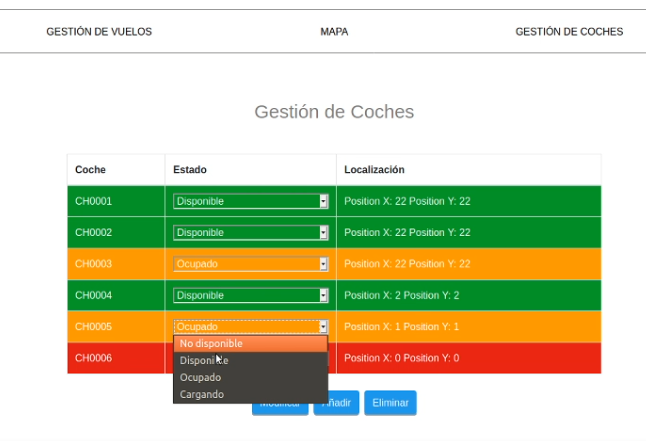


Tot això tret de la Base de dades utilitzant la API.

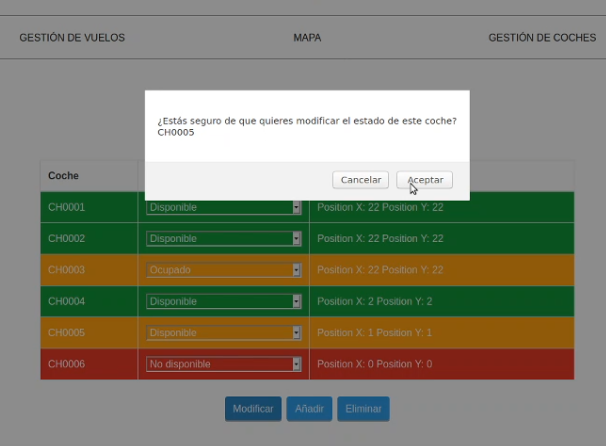
**Web Administrador:**

S’ha implementat poder modificar, eliminar i afegir cotxes.

Tenim diferents estats que podem modificar de forma manual:



Sempre volem anticipar possibles errors de manera que afegim els següent missatge:



I ja ens quedaria modificat:



En la gestió de vols també disposem de les mateixes funcions:



Seguint amb la pàgina d’administrador parlarem de les millores al tocken. Centrades en 2 coses:

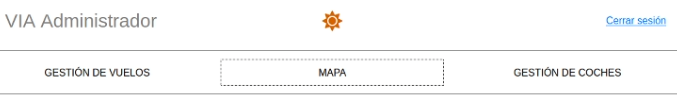
-Temps d’expiració.

-Distingir entre treballadors.

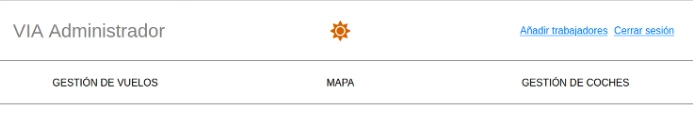
El temps d’expiració, es un temps X en el qual si no interactues amb la pagina d’administrador automàticament et traurien els privilegis de utilitzar la pagina i hauries de repetir el login en cas que vulguis tornar a entrar.

Per altra banda volíem afegir una gestió dintre de la pagina d’admin sobre els treballadors que puguin accedir aquesta. Així hem fet que el tocken distingeixi entre un treballador normal i un “jefe”, aquest últim seria el únic que veuria la funció de gestionar treballador.

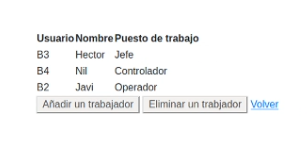
Per a un treballador normal:



Per al “jefe”:



Que trobem dins d’aquesta pestaña “añadir trabajadores”



Es pot veure una taula amb els treballadors actuals, el seu usuari, i la seva funció en el aeroport.

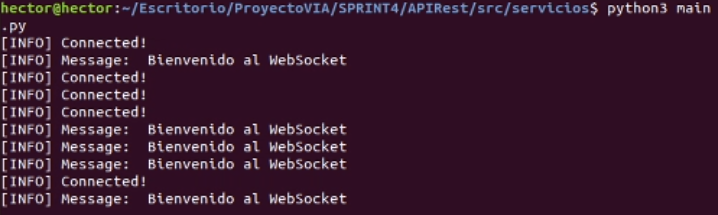
I varies funcionalitats una d’afegir un treballador nou, i un altre d’eliminar-ne algun. Així podem decidir i veure fàcilment qui te accés a la pagina d’admin.

**API - Pagina Administrador**

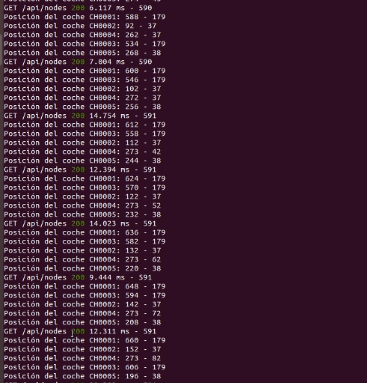
Un dels increments més importants es la de la connexió amb el cotxes autònoms. Aquesta millora s’ha fet a la api però es veu reflectida des de la pagina d’administrador.

Com funciona?

Utilitzant el programa main.py facilitat per l’equip B1. Es “simulen” 5 cotxes que tenim a la base de dades.



Aquest cotxes segueixen una ruta establerta. I es comunicarien a traves de la api donant les severs coordenades. Això ho farien cada 3 segons.



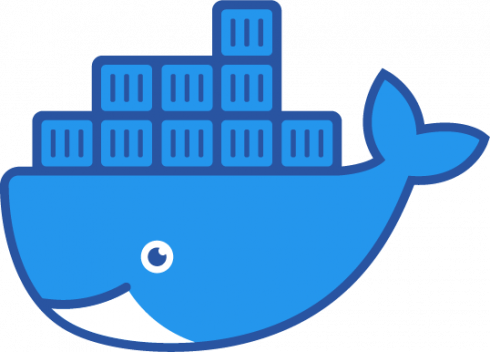
Des de el mapa a temps real de la pagina d’administrador veuríem cada 3 segons el cotxes actualitzant la seva posició.



Canvis en atributs de la API.

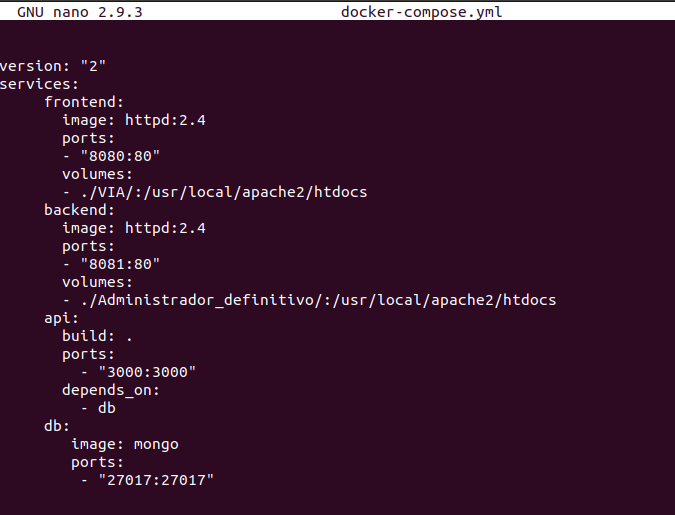
**Contenidor Docker.**

Novament aquest sprint tot està connectat mitjançant contenidors Docker.



Les dues webs estan cadascuna en un contenidor Docker on corre un servidor web Apache al seu interior i tant la API com la base de dades estan en un contenidor Docker independent.

Tots els contenidors estan connectats entre si mitjançant Docker-Compose. El seu fitxer de configuració es docker-compose.yml i el contingut es el següent:



Aquesta connexió anirà dins del servidor Debian que properament es migrarà al CRAAX, amb una configuració més avançada en quant a paràmetres de xarxa.

Desplegament del servei web i API

## Descripció general

//Descripció més detallada i concreta d’aquesta part del treball.

Web Administrador (Back-End):

**Pàgina d’inici**: Modificada amb un fons comú amb el de la pàgina d’usuaris.

**Login**: Funcional amb dades reals, detecció de majúscules, contrasenya oculta i opció de visualitzar-la amb un botó que fa d’ull.

**Gestió de cotxes**: Veiem les dades dels cotxes presents a la base de dades i podem afegir-ne nous, properament, també modificar-los.

**Mapa**: Continua en el mateix estat que l’SPRINT2, properament mostrarà dades en temps real de passatgers.

**Gestió de vols**: Veiem les dades dels vols presents a la base de dades i podem afegir-ne nous, properament, també modificar-los.

Web Usuaris (Front-End):

**Pàgina principal**: S’han fet millores al codi CSS per a una interacció més àgil.

Informació de vols: Nova pestanya que mostra els vols, properament, mostrarà dades reals de la base de dades, ara mateix, merament disseny.

**Localitzador**: Nova pestanya que permet cercar un vol concret, properament, farà una cerca a la base de dades, ara mateix, merament disseny.

**Serveis**: Continua en el mateix estat que l’anterior Sprint, on estava molt avançada, apareixen els comerços de l’aeroport.

**Mapa**: Apareix el mapa de l’aeroport de les dues plantes, continua en el mateix estat que l’anterior Sprint. Properament es situaran els comerços.

**Accessos**: S’ha prescindit de GMaps, ara mostra diverses opcions d’arribada i localització de l’aeroport. Cada localització està mapejada a la web de GMaps per a que l’usuari la pugui ubicar i arribar a ella.

**Contacte**: Pendent d’implementar per a que mostri un formulari de contacte.

API Rest:

**Funcionalitats**: S’han adaptat peticions GET, POST, PUT i DELETE per a les col·leccions reals de la base de dades *terminal2*.

**Seguretat**: S’ha implementat la gestió de Tokens mitjançant JWT, creant un middleware d’autenticació i està funcionant per a fer un login.

Connexió dels serveis:

S’han interconnectat tots els nostres serveis mitjançant Docker-Compose.

Cada servei es troba a un contenidor Docker mapejat a un port concret.

Les dues webs estan als ports 8080 i 8081, la API al port 3000 i la base de dades al port 27017.

## Explicació software/hardware utilitzat

**Node.js**: Entorn que treballa en temps real d'execució, de codi obert, multi plataforma

que permet al programador crear tota classe d'eines del costat servidor (BackEnd) i

aplicacions en Javascript.

**Docker:** Projecte de codi obert que automatitza el desplegament d'aplicacions dins de contenidors software.

**POSTMAN**: Software que simula peticions a la API de rutes, des d'una aplicació web/mòbil.

**MongoDB:** Sistema de base de dades NoSQL, orientat a documents i de codi obert.

Mòduls necessaris de Node.js:

**Express**: Framework de Node, crea l'estructura del servidor i ens permet

escriure codi de manera senzilla.

**Body-Parser**: Mòdul que ens permet convertir les dades que ens arriben en les peticions al servidor, en objectes amb format JSON.

**Mongoose**: Mòdul que ens proveeix mètodes i funcionalitats per a treballar millor amb MongoDB.

**Nodemon**: Mòdul que farà que amb qualsevol canvi en el servidor, aquest es reiniciï (fa com una compilació), a més de que mostra per consola totes les peticions HTTP (GET, POST, PUT, DELETE…) que rep el servidor (200, 404, 500, etc.)

**Cors:** Mòdul que permet que es puguin sol·licitar recursos exclusius.

**JsonWebToken:** Mòdul que permet el sistema d’autenticació mitjançant Tokens.

**JavaScript**: Llenguatge de programació.

**HTML5**: Conjunt de tecnologies Web

**CSS3**: Llenguatge de disseny gràfic enfocat en Web.

**Bootstrap**: Conjunt d'eines per a disseny Web.

**AJAX (JavaScript And XML)**: Técnica de desenvolupament web per a crear aplicacions interactives.

**W3.CSS Icons**: Llibreria d’icones gratuïts de W3Schools.

**Ionic icons**: Pàgina web d’icones.

**Flaticons**: Pàgina que ofereix fotos en format png.

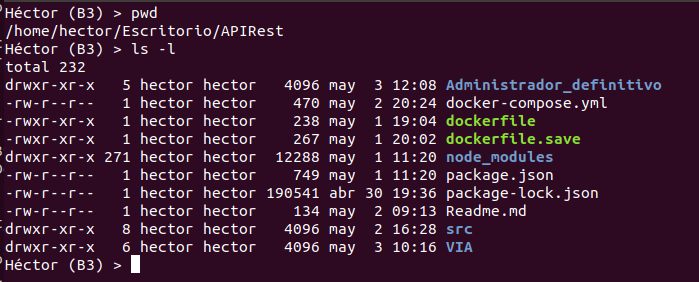
**Pexels**: Servei que ofereix fotos de stock.

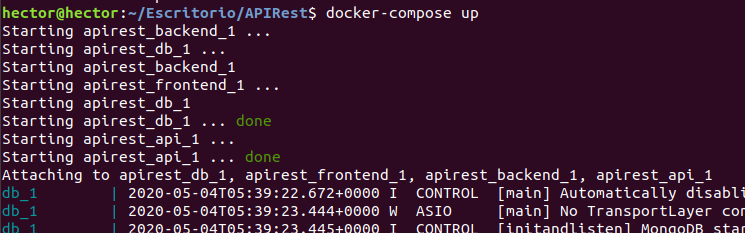
**Weatherwidget**: Widget responsive sobre el clima en temps real de qualsevol ubicació.

## Manual d’usuari

//Interaccions que pot realitzar l’usuari amb el projecte actual.

Amb el codi Web, codi de l’API i els fitxers Dockerfile per muntar la API i docker-compose.yml per interconnectar tots els serveis en un mateix directori, qualsevol podria entrar la comanda *docker-compose up* i poder gaudir dels quatre serveis interconnectats funcionant i poder veure totes les funcionalitats que hem indicat a l’apartat anterior:





## Conclusions

Em pogut unir serveis entre equips de la terminal a pesar de no haver aconseguir tenir-ho tot en cloud. Estem aprenent moltes coses noves que mai havíem tingut contacte abans.

# 

# Definició de protocols

## Descripció general

//Descripció més detallada i concreta d’aquesta part del treball.

Els clients Web es connecten a la API mitjançant peticions AJAX (JavaScript And XML).

La API fa peticions HTTP (Protocol de transport TCP) a la base de dades MongoDB. La base de dades MongoDB retorna dades en format JSON (format de text orientat al intercanvi de dades) i aquest JSON es retorna als clients Web.

En quant a l’aplicació mòbil es quasi bé el mateix, la única diferencia es que les peticions son FETCH.

En quant als cotxes, s’obrirà un WebSocket a la API i es connectaran.

## Explicació/Justificació

//Explicació de la raó per la qual s’utilitzaran els protocols definits

Trobem que son les connexions mes utilitzades per aquests tipus de serveis, i en quant a la connexió mitjançant socket amb els cotxes, creiem que es la millor opció per a fer una connexió entre Node.js (API) i Python (cotxes).

## Conclusions

//\*\*Cal afegir un manual o be un link amb documentació dels protocols utilitzats.

Els protocols utilitzats es poden veure a l’esquema general inclòs al primer apartat.