



Desplegament de la Infraestructura i Arquitectura, l'aplicació mòbil, i la Base de Dades



Índex...

Descripció general Sprint	3
Desplegament de la Infraestructura i Arquitectura	4
Descripció general	4
Explicació software/hardware utilitzat	4
Manual d'usuari	4
Conclusions	4
L'aplicació mòbil	5
Descripció general	5
Explicació software/hardware utilitzat	5
Manual d'usuari	5
Conclusions	4
Base de Dades	6
Descripció general	6
Explicació software/hardware utilitzat	6
Manual d'usuari	6
Conclusions	6

Descripció general Sprint

En aquest sprint hem intentat començar a unir els punts de la terminal amb els altres equips. Es important que una vegada pasada la fase inicial del projecte, començem a treballar els equips en conjunt, per aixó ha sigut necessari moltes hores de reunions amb altres equips, y una bona utilització de les eines de comunicació per tal de que aquesta sigui lo més entenedora posible.

Desplegament de la Infraestructura i Arquitectura

Descripció general

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

Explicació software/hardware utilitzat

Manual d'usuari

DOCUMENTACIÓ DOCKER - PART DEL FOG

<https://docs.docker.com/engine/install/debian/>

<https://docs.docker.com/engine/install/linux-postinstall/>

1) Desinstalar versiones antiguas de Docker por si las moscas pican.

- `sudo apt-get remove docker docker-engine docker.io containerd runc`
- `sudo apt-get update`

```
jaume@Debian:~$ su
Contraseña:
root@Debian:/home/jaume# apt-get update
Obj:1 http://security.debian.org/debian-security buster/updates InRelease
Obj:2 http://deb.debian.org/debian buster InRelease
Obj:3 http://deb.debian.org/debian buster-updates InRelease
Leyendo lista de paquetes... Hecho
root@Debian:/home/jaume#
```

2) Install using the repository

- `sudo apt install apt-transport-https ca-certificates curl gnupg2 software-properties-common`

```
jaume@Debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
Seleccionando el paquete apt-transport-https previamente no seleccionado.
(Leyendo la base de datos ... 133067 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar .../apt-transport-https_1.8.2_all.deb ...
Desempaquetando apt-transport-https (1.8.2) ...
Seleccionando el paquete curl previamente no seleccionado.
Preparando para desempaquetar .../curl_7.64.0-4+deb10u1_amd64.deb ...
Desempaquetando curl (7.64.0-4+deb10u1) ...
Seleccionando el paquete gnupg2 previamente no seleccionado.
Preparando para desempaquetar .../gnupg2_2.2.12-1+deb10u1_all.deb ...
Desempaquetando gnupg2 (2.2.12-1+deb10u1) ...
Configurando gnupg2 (2.2.12-1+deb10u1) ...
Configurando apt-transport-https (1.8.2) ...
Configurando curl (7.64.0-4+deb10u1) ...
Procesando disparadores para man-db (2.8.5-2) ...
root@Debian:/home/jaume#
```

3) Add Docker's official GPG Key

- `curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -`

```
root@Debian:/home/jaume# curl -fsSL https://download.docker.com/linux/debian/gpg
| sudo apt-key add -
OK
root@Debian:/home/jaume#
```

- `sudo apt-key fingerprint 0EBFCD88`

```
root@Debian:/home/jaume# sudo apt-key fingerprint 0EBFCD88
pub  rsa4096 2017-02-22 [SCEA]
    9DC8 5822 9FC7 DD38 854A  E2D8 8D81 803C 0EBF CD88
uid  [desconocida] Docker Release (CE deb) <docker@docker.com>
sub  rsa4096 2017-02-22 [S]

root@Debian:/home/jaume#
```

- Afegim el corresponent repository

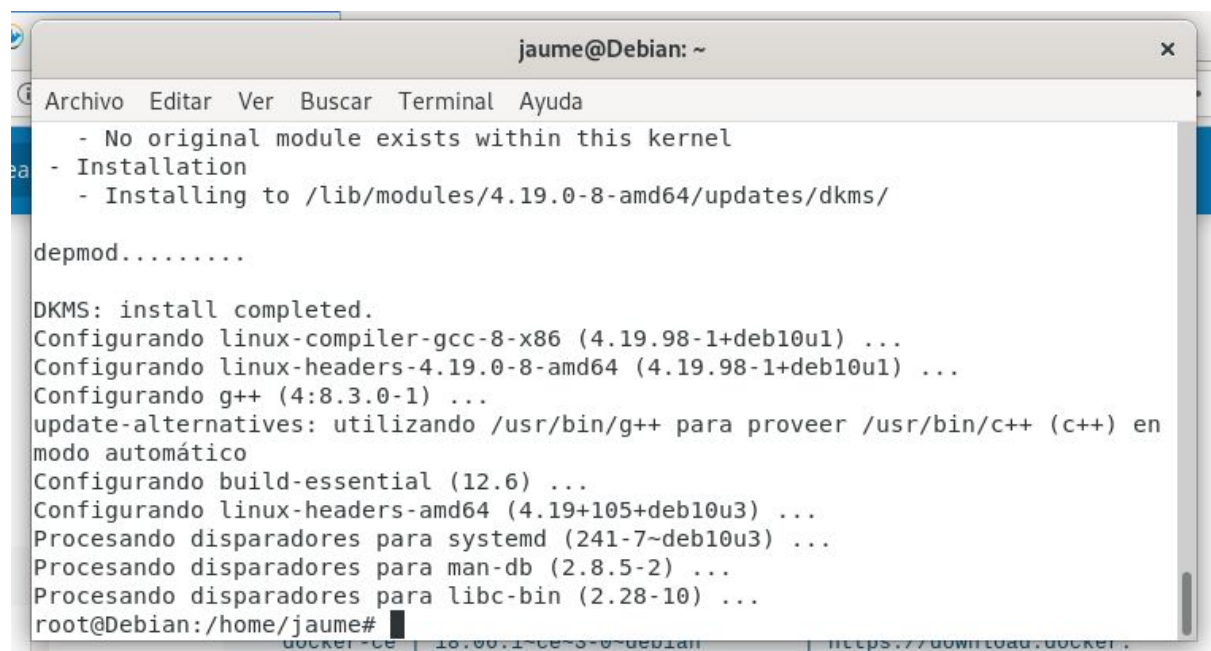
```
root@Debian:/home/jaume# sudo add-apt-repository \
> "deb [arch=amd64] https://download.docker.com/linux/debian \
> $(lsb_release -cs) \
> stable"
root@Debian:/home/jaume#
```

INSTAL·LACIÓ DE DOCKER

1) `sudo apt-get update`

```
root@Debian:/home/jaume# sudo apt-get update
Obj:1 http://deb.debian.org/debian buster InRelease
Obj:2 http://deb.debian.org/debian buster-updates InRelease
Des:3 https://download.docker.com/linux/debian buster InRelease [44,4 kB]
Des:4 https://download.docker.com/linux/debian buster/stable amd64 Packages [11,5 kB]
Obj:5 http://security.debian.org/debian-security buster/updates InRelease
Descargados 55,9 kB en 1s (46,7 kB/s)
Leyendo lista de paquetes... Hecho
root@Debian:/home/jaume#
```

2) `sudo apt-get install docker-ce docker-ce-cli containerd.io`



```
jaume@Debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
- No original module exists within this kernel
- Installation
- Installing to /lib/modules/4.19.0-8-amd64/updates/dkms/

depmod.....

DKMS: install completed.
Configurando linux-compiler-gcc-8-x86 (4.19.98-1+deb10u1) ...
Configurando linux-headers-4.19.0-8-amd64 (4.19.98-1+deb10u1) ...
Configurando g++ (4:8.3.0-1) ...
update-alternatives: utilizando /usr/bin/g++ para proveer /usr/bin/c++ (c++) en
modo automático
Configurando build-essential (12.6) ...
Configurando linux-headers-amd64 (4.19+105+deb10u3) ...
Procesando disparadores para systemd (241-7~deb10u3) ...
Procesando disparadores para man-db (2.8.5-2) ...
Procesando disparadores para libc-bin (2.28-10) ...
root@Debian:/home/jaume#
```

3) Ara hem de triar una versió del docker, primer les llistem

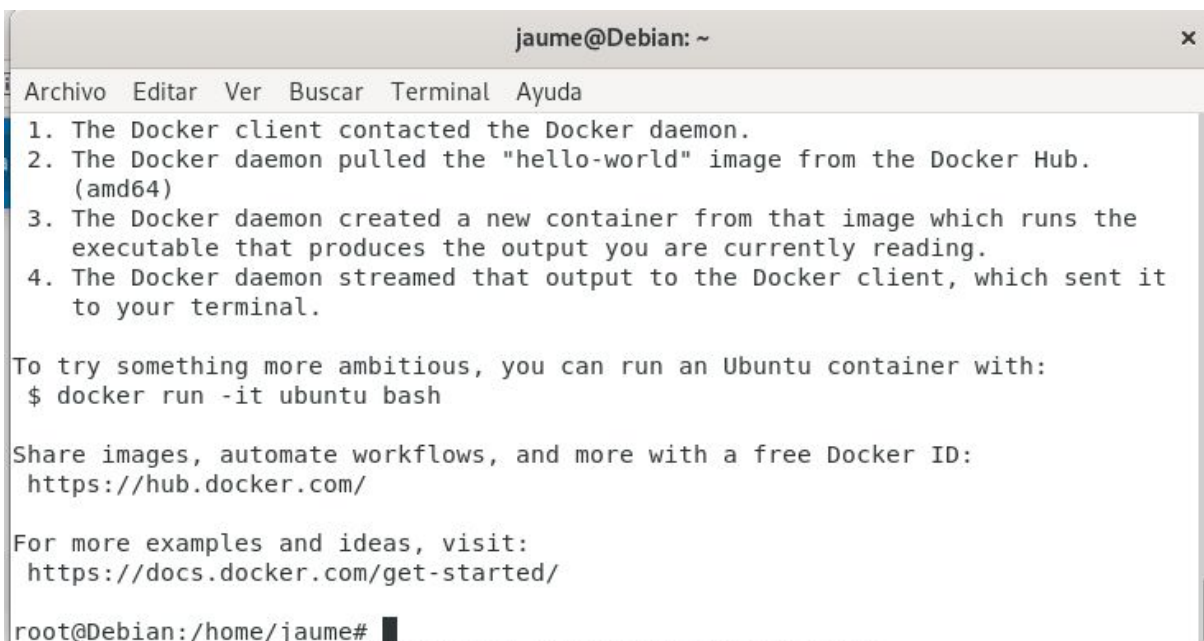
- `apt-cache madison docker-ce`

```
root@Debian:/home/jaume# apt-cache madison docker-ce
docker-ce | 5:19.03.8~3-0~debian-buster | https://download.docker.com/linux/debian buster/stable amd64 Packages
docker-ce | 5:19.03.7~3-0~debian-buster | https://download.docker.com/linux/debian buster/stable amd64 Packages
docker-ce | 5:19.03.6~3-0~debian-buster | https://download.docker.com/linux/debian buster/stable amd64 Packages
docker-ce | 5:19.03.5~3-0~debian-buster | https://download.docker.com/linux/debian buster/stable amd64 Packages
docker-ce | 5:19.03.4~3-0~debian-buster | https://download.docker.com/linux/debian buster/stable amd64 Packages
docker-ce | 5:19.03.3~3-0~debian-buster | https://download.docker.com/linux/debian buster/stable amd64 Packages
```

- instal·lem la versió que volem utilitzar

```
root@Debian:/home/jaume# apt-get install docker-ce=5:19.03.8~3-0~debian-buster docker-ce-cli=5:19.03.8~3-0~debian-buster containerd.io
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
containerd.io ya está en su versión más reciente (1.2.13-1).
docker-ce-cli ya está en su versión más reciente (5:19.03.8~3-0~debian-buster).
docker-ce ya está en su versión más reciente (5:19.03.8~3-0~debian-buster).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
root@Debian:/home/jaume#
```

4) `sudo docker run hello-world`



```
jaume@Debian: ~
Archivo Editar Ver Buscar Terminal Ayuda
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

root@Debian:/home/jaume#
```

5) Revisem que efectivament el servei docker està funcionant

- `sudo service docker status`



jaume@Debian: ~

Archivo Editar Ver Buscar Terminal Ayuda

Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

Active: **active (running)** since Fri 2020-05-01 20:57:21 CEST; 4min 46s ago

Docs: <https://docs.docker.com>

Main PID: 508 (dockerd)

Tasks: 9

Memory: 127.9M

CGroup: /system.slice/docker.service

└─508 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

may 01 20:57:12 Debian dockerd[508]: time="2020-05-01T20:57:12.493284028+02:00" level=warning msg="

Desplegament del Cloud

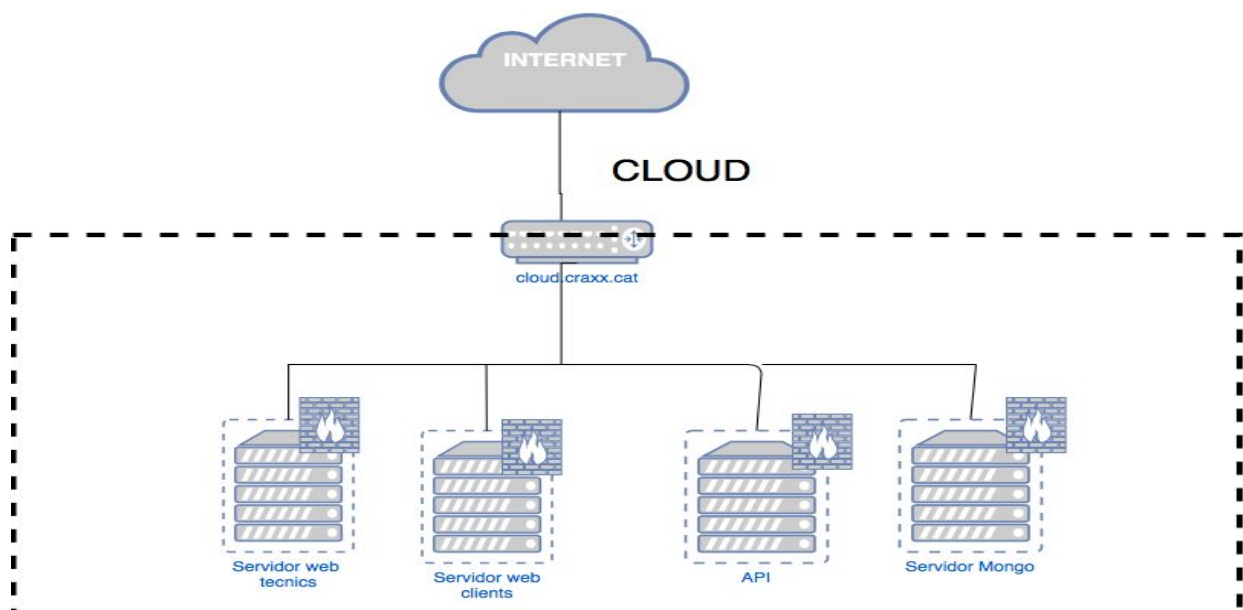
A continuació, implementarem un esquema definitiu (suposadament) del cloud, el qual ha rebut diverses modificacions respecte al anterior sprint, tant després de parlat amb el client tècnic, com debatre amb la resta de grups.

Les principals diferències respecte al cloud realitzat en el anterior sprint son:

- Ara les màquines mongo.craxx.cat i www.craxx.cat, les qual allotjen el servidor mongoDB i la pàgina web per a clients, respectivament, funcionen en dockers.
- S'han afegit dos dockers addicionals, un on introduïrem la API del grup B3, i un altre per allotjar la pàgina web per a tècnics.
- El router cloud.craxx.cat queda representat com la pròpia màquina host.

A més, utilitzant les funcions que docker-compose ens permet, introduïrem els 4 contenidors en una mateixa red virtual per facilitar la comunicació entre els dockers.

El esquema queda de la següent manera:





Desplegament del Servei

Per facilitar el desplegament de la infraestructura Cloud, creem un arxiu `docker-compose.yml` en la ruta `/Desplegament`.

Servei viawebclients

Per crear aquest servei, agafem la imatge `httpd:2.4`, la qual es una imatge apache públic de dockerhub. La descarga es realitzara automaticament en cas de no dispondre d'aquesta en la màquina local.

Anomenarem al contenidor creat `viawebclients`, a més, mapearem el port `8080` de la màquina host al `80` del docker

A més, crearem un volum per introduir la pàgina web del grup B3 en la imatge del servidor apache, aixó permet un desplegament senzill de la web, a més de que a l'hora de realitzar actualitzacions a aquesta, simplement tindrem que canviar el contingut de la carpeta origen

Afegirem el docker a la red `cloud_net` amb la ip `10.0.0.2`

Servei viawebtecnicos

Els pasos aseguir son identics als anteriors, simplement canviarem la carpeta origen del volum, mapejarem el port `8081` en comptes del `8080`, i assignarem la ip `10.0.0.3` al contenidor dins la `cloud_net`.

Servei db

Per crear aquest servei, afegirem la imatge `mongo`, la qual és una imatge apache pública de dockerhub. La descarga es realitzara automaticament en cas de no dispondre d'aquesta ne la maquina local.

Anomenarem al contenidor creat `db`, a més, mapejarem el port `27017` de la maquina host al `27017` del docker.

Declarem una variable d'entorn amb el nom de la base de dades, i mapejarem el arxiu `.js` que crea les tables e introdueix les seves dades a una ruta dins del docker.

Al introduir el arxiu `.js` en aquesta carpeta en qüestió, al iniciar el docker de Mongo automàticament s'executa el arxiu `.js` en qüestió i ens queda la base de dades ya creada.

Per últim, assignem al docker a la red `cloud_net` amb la ip `10.0.0.4`



Servei apirest

Per aquest servei, no utilitzarem una imatge de dockerhub com en les anteriors, sino que utilitzarem la api que ens ha passat directament el grup B3.

Construirem el servei a partir del Dockerfile que també ens ha pasat el grup B3 i anomenarem al contenedor api rest.

Mapejarem en el port 3000 i indicarem que te dependencia de la base de dades db, per últim, la introduïrem en la red cloud_net amb la ip 10.0.0.4

Docker-Compose

Per últim, mostrem el resultat final.

```
version: '3'

services:
  viawebclients:
    image: httpd:2.4
    container_name: viawebclients
    ports:
      - "8080:80"
    volumes:
      - ./webClients:/usr/local/apache2/htdocs/
    networks:
      cloud_net:
        ipv4_address: 10.0.0.2

  viawebtecnicos:
    image: httpd:2.4
    container_name: viawebtecnicos
    ports:
      - "8081:80"
    volumes:
      - ./webTecnicos:/usr/local/apache2/htdocs/
    networks:
      cloud_net:
        ipv4_address: 10.0.0.3

  db:
    image: mongo
    container_name: db
    environment:
      # MONGO_INITDB_ROOT_USERNAME: admin
      # MONGO_INITDB_ROOT_PASSWORD: admin
      MONGO_INITDB_DATABASE: terminal2
```

```
# MONGO_INITDB_ROOT_PASSWORD: admin
MONGO_INITDB_DATABASE: terminal2
volumes:
  - ./mongoDades/MongoDB.js:/docker-entrypoint-initdb.d/init.js
ports:
  - "27017:27017"
networks:
  cloud_net:
    ipv4_address: 10.0.0.4

apiREST:
  build:
    context: ./apiREST
    dockerfile: ./dockerfile
  image: apiREST
  container_name: apiREST
  ports:
    - "3000:3000"
  depends_on:
    - db
  networks:
    cloud_net:
      ipv4_address: 10.0.0.5

networks:
  cloud_net:
    ipam:
      driver: default
      config:
        - subnet: 10.0.0.0/24
```



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

Conclusions



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

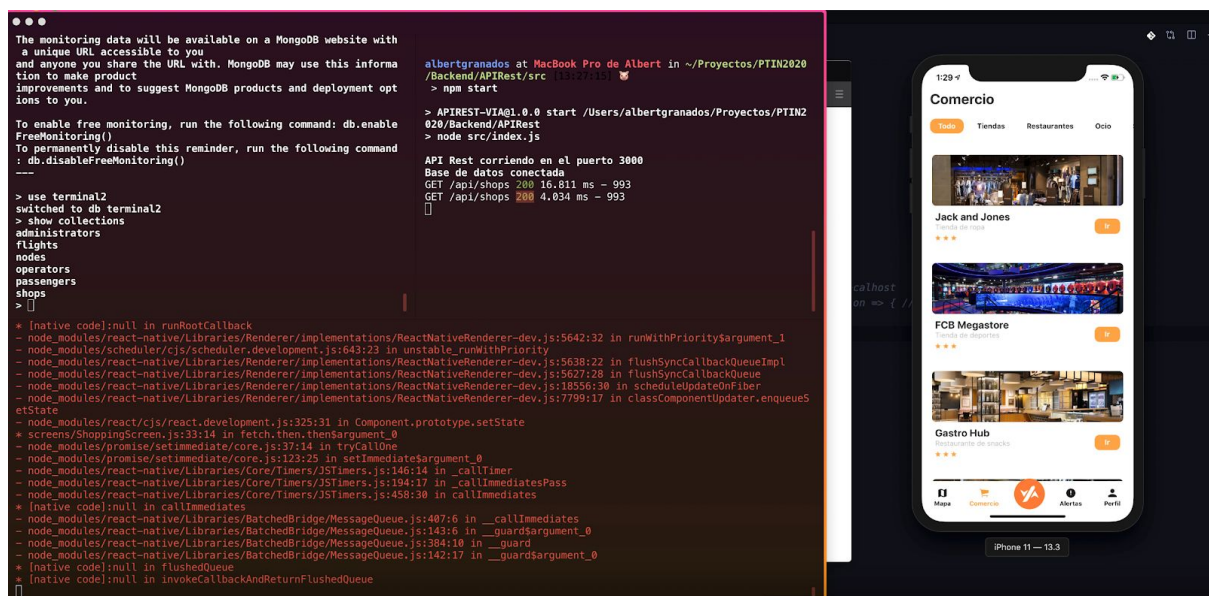
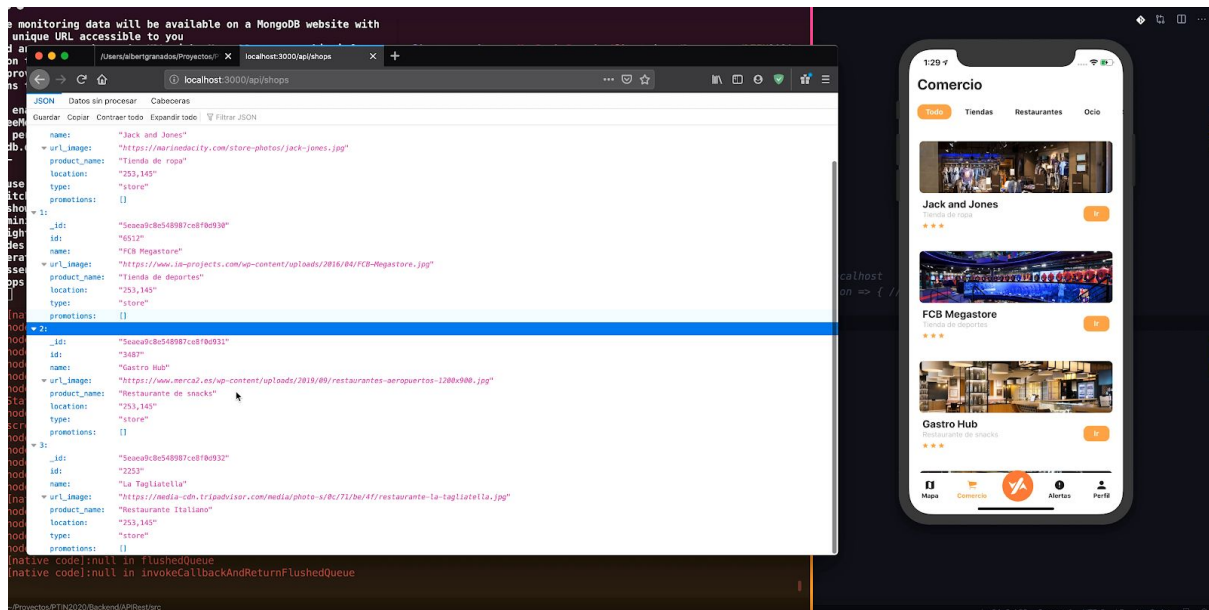
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

L'aplicació mòbil

Descripció general

S'ha desplegat la API del grup B1 en local, i hem adaptat l'aplicació per a que la informació de la bdd sigui accessible desde la app connectant amb la API.

També s'han arreglat errors de la app amb la navegació i els filtres.



Explicació software/hardware utilitzat

Hem utilitzat la API que ens ha proporcionat el grup B1 i hem fet alguns imports d'exemple a la base de dades amb mongo shell. Hem desplegat la api en local i hem fet alguna petició per comprovar la connexió entre la app i la API.

```
// Creacion de colecciones
db.createCollection("administrators");
db.createCollection("operators");
db.createCollection("passengers");
db.createCollection("nodes");
db.createCollection("flights");
db.createCollection("shops");

db.administrators.insert({
  id: "12123",
  name: "Josep M. Roig",
  email: "jmroig@gmail.com",
  birthdate: new Date("1990-05-18T16:00:00Z"),
  phone: "+34 9856233255",
  password: "aBnsizl"
});

db.operators.insert({
  id: "256845",
  name: "Daniel Ortiz",
  email: "dortiz@gmail.com",
  birthdate: new Date("1985-02-10T16:00:00Z"),
  phone: "+34 923568426",
  password: "m12Asml",
  airline: "Swissport"
});

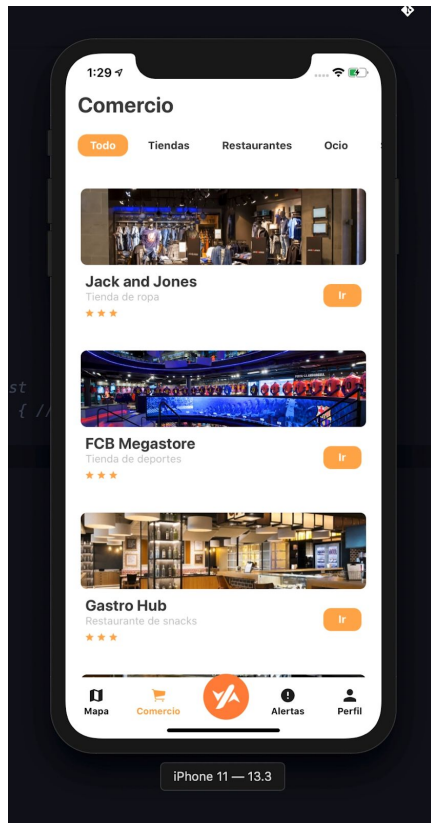
db.passengers.insert({
  id: "SD4564",
  name: "Antonio Delgado",
  email: "adelgado@gmail.com",
  birthdate: new Date("1993-01-26T16:00:00Z"),
  phone: "+34 9839565956",
  password: "1IMad5NVgs",
  country: "Spain",
  City: "Barcelona",
  location: "1.2,1.2"
});

db.passengers.insert({
  id: "A334i499J",
  name: "Jordi Gasol",
  email: "jgasol@gmail.com",
  birthdate: new Date("1999-01-02T16:00:00Z"),
  phone: "+34 681339900",
  password: "sA3iskps",
  country: "Spain",
  City: "Barcelona",
  location: "14,55"
});
```

```
> use terminal2
switched to db terminal2
> show collections
administrators
flights
nodes
operators
passengers
shops
> 
```

Manual d'usuari

L'usuari ara pot interactuar amb la llista de comerços.



Conclusions

Ha sigut un Sprint on s'ha prioritzat la comunicació entre l'equip de la app i BDD i la API i la Web. La comunicació ha sigut bona i s'ha aconseguit l'objectiu que era començar a tancar el cercle de connexió entre els elements de la terminal. En el següent sprint s'implementarà la API al servidor. i tancarem el cercle de connexió entre API, app i base de dades.

Prueba conexión App-API-BD en local

Descripció general

Se ha buscado información de como hacer una conexión entre la App - API - BD y posteriormente a su implementación en local porque aún no tenemos acceso a las máquinas virtuales , se ha tenido que probar primero con una app de prueba porque la App real aun le faltaban algunas cosas por detallar, pero luego de esta prueba se probó en la App que es la explicación de la parte de la App

Explicació software/hardware utilitzat

React : Es una biblioteca de javascript de código abierto. Esta sería como una prueba de app no es react native pero es lo que más asemeja en cuanto a la programación.

MongoDB: Es un sistema de base de datos no relacional. Es el que usamos para nuestra base de datos.

Node: es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript

npm: es el sistema de gestión de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript, con esto podemos hacer que nuestra Appi funcione en local y también la prueba de la App.

Para poder hacer la prueba, primero que nada hay que inicializar nuestra base de datos mongo, dependiendo de en qué sistema operativo estés los comandos podrían variar, en el caso de debian es:

mongod para iniciar el servidor

mongo para interactuar con la base de datos.

una vez que estemos dentro de mongo para poder interactuar podemos ver las base de datos que tenemos, usar una base de datos y poder ver sus colecciones.

```
2020-05-05T12:59:02.950702000 I CONTROL [mongo@cs-ten]
> show dbs
admin            0.000GB
config           0.000GB
local            0.000GB
restapi-local    0.000GB
term2            0.000GB
> use term2
switched to db term2
> show collections
administrators
flights
nodes
operators
passengers
shops
>
```

Una vez hecho ya la parte de la BD pasamos a la parte de la APPI que una vez que la tengamos nos vamos a su carpeta contenedora y ponemos el siguiente comando:

npm start

```
cesar@cesar-ZenBook:~/Escritorio/Web/APIRest$ npm start

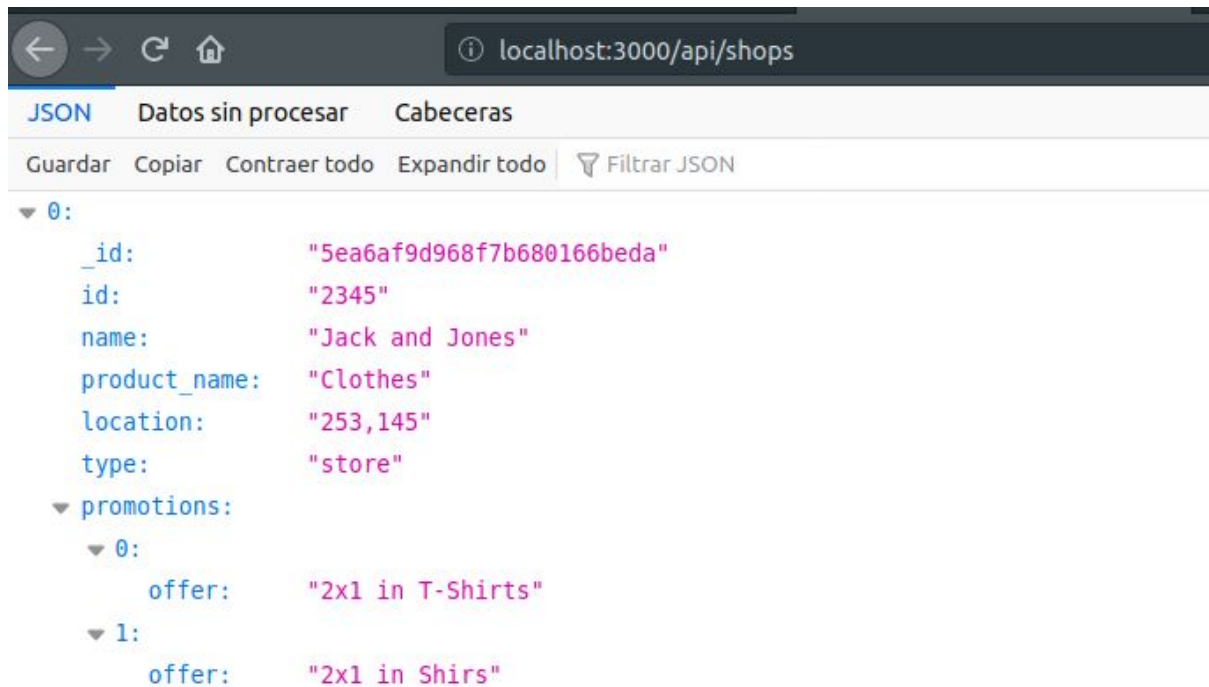
> APIREST-VIA@1.0.0 start /home/cesar/Escritorio/Web/APIRest
> node src/index.js

API Rest corriendo en el puerto 3000
Base de datos conectada
```

Como podemos observar la nos dice que ya está en funcionamiento y por el puerto 3000.

Podemos ver su funcionamiento si nos vamos a un navegador cualquiera y ponemos los siguiente:

<http://localhost:3000/api/shops>



Podemos observar que si nos muestra las tiendas que hay, en este caso solo hay uno pero porque es de prueba.

Una vez hecho esto ya podemos decir que la conexión APPI-BD ya está hecha, ahora solo nos queda la app, que éste caso es casi una web pero es solo de prueba, es decir conectarnos a la appi desde la prueba de la app

Para eso usamos React, lo más parecido a React native

```
getShops(){
  fetch('/shops') // URL de la API, en nuestro caso sera localhost
  .then(res => res.json()) // Cuando recibamos los datos, se convierten en json
  .then(json => { // cuando ya esté en formato json
    this.setState({
      isLoading: true, // decimos que ya ha cargado
      items: json, // guardamos los datos en una lista llamada items
    })
  });
}

render() {
  var {isLoading, items} = this.state;

  if(!isLoading){
    return <div> Loading... </div>
  }
  else {
    return (
      <div className="App">
        <ul>

          {items.map( item => (
            <li key={item.id} >
              Name: {item.name}
            </li>
          ))};

        </ul>
      </div>
    );
  }
}
```

Este es un código muy sencillo que hemos hecho de una app, la parte crucial es la fetch quien es el que hace la llamada a la appi, como podemos ver sólo tiene '/shops', eso es porque en el package.json que es lo que se ejecutara antes que el código que se ve aquí.

En el archivo package.js tenemos que añadir lo siguiente para que podamos probarlo en local:

```
"proxy": "http://localhost:3000/api"
```

Esto es la dirección de nuestra api, es decir donde se encuentra, en este caso en localhost, pero podría ir perfectamente el nombre de un dominio.

Una vez hecho todo esto ya podemos probar nuestra app de prueba usando el comando: npm start, dentro de la carpeta contenedora de nuestra app de prueba.



• Name: Jack and Jones

;



Como podemos observar nos muestra solo el nombre del comercio pero es suficiente para saber que nos hace la conexión App-API-BD

Conclusions

En un principio decidimos probarla directamente en la App pero nos faltaban algunas cosas por eso es que la probamos con una App de prueba para que al menos pudiéramos hacer la prueba.

Se puede hacer la conexión App-API-BD, con esto ya podemos pasar a hacer las pruebas con la App verdadera.