

SPRINT 5

Index

GUÍA PER ACCEDIR ALS SERVEIS DEL CRAAX	4
Ports per accedir:	4
Web:	5
API	5
Podem accedir des del navegador:	5
Podem accedir des d'un client visual:	6
MongoDB	7
Des d'una màquina virtual:	7
Des d'un client visual:	8
Backups	9
Direccions IP de les màquines del CRAAX:	9
Contenidors	9
Cloud	10
Fog	10
Leader	11
Exportem les imatges en la màquina dels backups	11
Fitxers	12
Crea backup de la base de dades:	12
Cloud	13
Fog	13
Restaura la base de dades	14
Mesura extrema de seguretat:	16
Instal·lar MongoDB a la màquina de Backups	16
Esquema de comunicació	19
Comunicacions dels elements	21
Connexió aplicació mòbil	21
Pàgina web de gestors:	23
Pàgina web dels Clients	24
Pàgina web dels Clients (fora l'aeroport)	25
Comunicació Flota de cotxes:	25
Medi de comunicacions	26
Cotxe autònom:	26
App mòbil:	27
Pàgina web:	28
Disponibilitat	29
Què és l'alta disponibilitat?	29

Alta disponibilitat en el Cloud i el Fog?	30
Solucions per arribar a obtenir una alta disponibilitat?	30
Altres coses a tindre en compte	31
Manteniment de docker-compose	32
Passos previs:	32
Passos per reiniciar un contenidor en concret:	32
Passos per fer un reinici de manteniment:	32
Construint una aplicació Android amb docker-android	33
Requeriments	34
Primers passos	34
Configurar la imatge	35
Construïm un projecte android	35
Controlem el dispositiu android connectat al host (emulador o dispositiu real)	35
Appium i Selenium Grid	36
Controlem el emulador android desde fora el contenidor	36
Simulació de SMS	36
Kubernetes	37
Què és Kubernetes?	37
Burndown	38
Webgrafia	39

GUÍA PER ACCEDIR ALS SERVEIS DEL CRAAX

En aquest document s'explica com accedir als serveis de les màquines del CRAAX.

1. Ports per accedir:

- **CLOUD:**

- Web → 36081
- API → 36301
- MongoDB → 36717

- **FOG:**

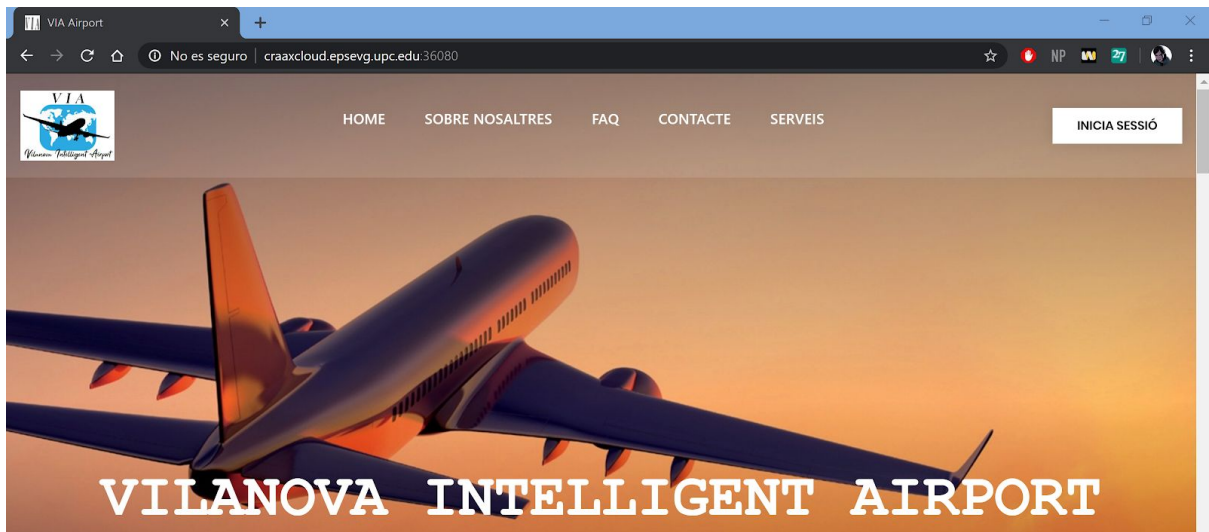
- Web → 36080
- API → 36302
- MongoDB → 36718

- **EDGE:**

- API → 36303

2. Web:

- Cloud: <http://craaxcloud.epsevg.upc.edu:36081/>
- Fog: <http://craaxcloud.epsevg.upc.edu:36080/>



3. API

Per accedir farem:

`http://craaxcloud.epsevg.upc.edu:port/metode`

Hem de tenir en compte els mètodes implementats a cada servidor. Si alguna cosa no funciona, notifiqueu-ho ràpidament.

3.1 Podem accedir des del navegador:

Per exemple:

<http://craaxcloud.epsevg.upc.edu:36301/pasajero>



3.2 Podem accedir des d'un client visual:

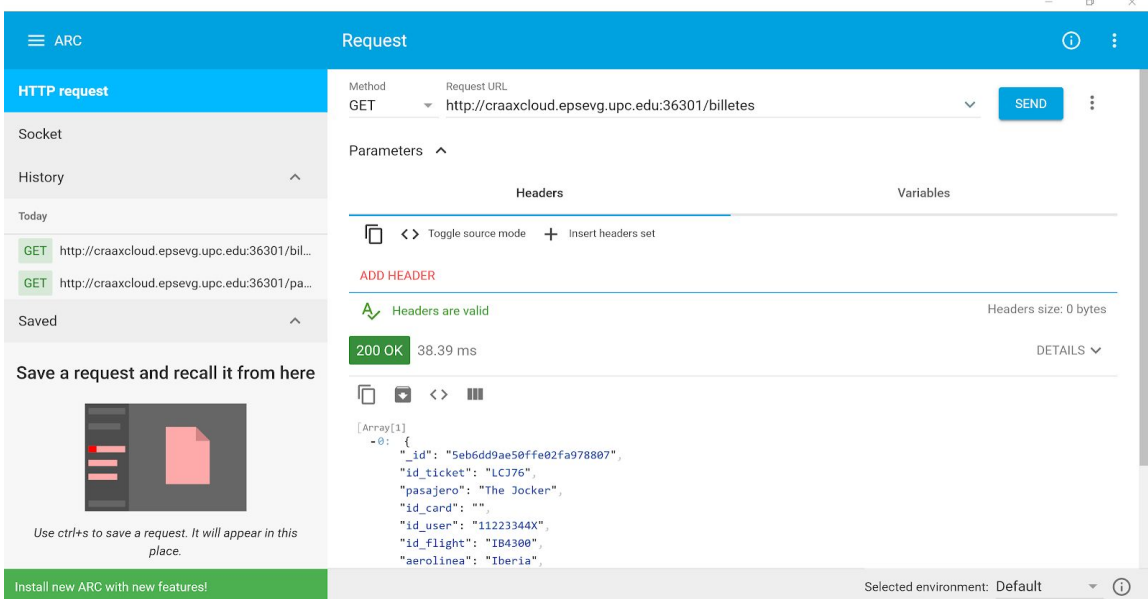
- Advanced Rest Client:

<https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddfdnphfgceellkdfbfbieloo/related?hl=es-419>

- Especifiquem el mètode i posem la direcció.

Per exemple:

<http://craaxcloud.epsevg.upc.edu:36301/billetes>



Qualsevol altre dubte sobre la API, hi ha un vídeo de demostració anomenat

Demo API.mp4 a la carpeta compartida pels equips:

<https://drive.google.com/drive/folders/18D-OhS3VUFGdkSuVmT-BDXq6eTEBFco3>

4. MongoDB

Recordeu canviar el port per l'adient, 36717 per al MongoDB del Cloud i 36718 per al MongoDB del Fog.

Per accedir a MongoDB es pot fer:

4.1 Des d'una màquina virtual:

Podeu descarregar directament la màquina que hi ha a la carpeta compartida, anomenada **PTIN_MongoDB.ova**:

<https://drive.google.com/drive/folders/18D-OhS3VUFGdkSuVmT-BDXq6eTEBFco3>

- Per accedir com admin:

```
# mongo -u admin --authenticationDatabase
admin craaxcloud.epsevg.upc.edu:36717
```

La password de moment és 'admin'.

```
root@debian:~# mongo -u admin --authenticationDatabase admin craaxcloud.epsevg.upc.edu:36717
MongoDB shell version v4.2.5
Enter password:
connecting to: mongodb://craaxcloud.epsevg.upc.edu:36717/test?authSource=admin&compressors=disabled
ServiceName=mongodb
Implicit session: session { "id" : UUID("c66d1a8b-77db-4564-9a33-75ab0d8c767e") }
MongoDB server version: 4.2.6
Server has startup warnings:
2020-05-18T15:05:57.050+0000 I STORAGE [initandlisten]
2020-05-18T15:05:57.050+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is
recommended with the WiredTiger storage engine
2020-05-18T15:05:57.050+0000 I STORAGE [initandlisten] ** See http://dochub.mongodb.
odnotes-filesystem
2020-05-18T15:05:59.592+0000 I CONTROL [initandlisten]
2020-05-18T15:05:59.592+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent
habled is 'always'.
2020-05-18T15:05:59.592+0000 I CONTROL [initandlisten] ** We suggest setting it to 'ne
2020-05-18T15:05:59.592+0000 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin 0.000GB
cloud 0.000GB
config 0.000GB
local 0.000GB
test 0.000GB
> █
```

- Per accedere con qualsiasi altro utente:

```
# mongo -u user_test -p user_test
craaxcloud.epsevg.upc.edu:36717
```

```
root@debian:~# mongo -u user_test -p user_test craaxcloud.epsevg.upc.edu:36717
MongoDB shell version v4.2.5
connecting to: mongodb://craaxcloud.epsevg.upc.edu:36717/test?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6a4af833-63e4-4e51-9484-00d7948268ab") }
MongoDB server version: 4.2.6
> show dbs
test 0.000GB
>
```

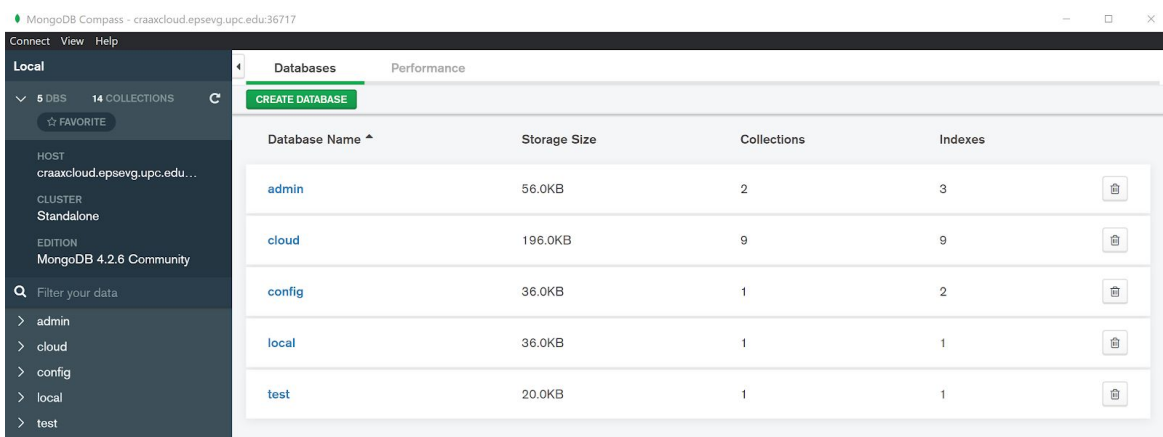
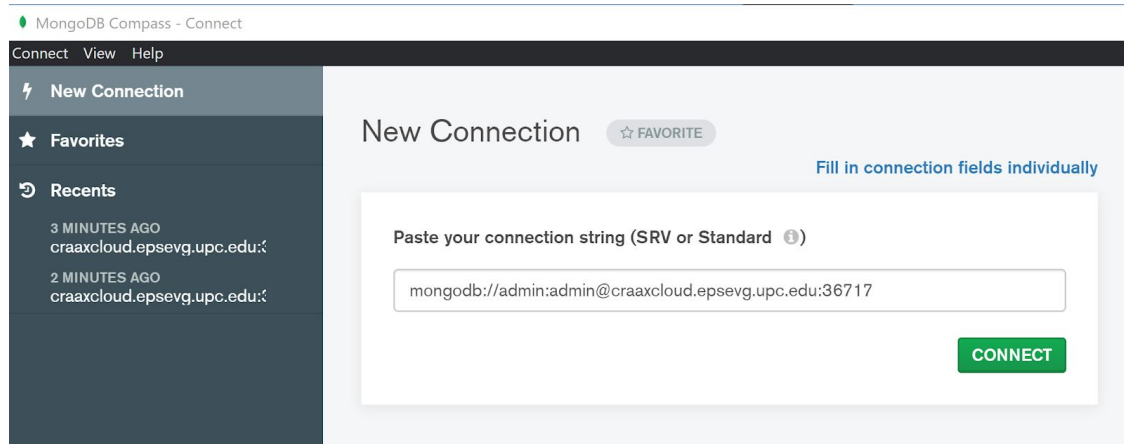
4.2 Des d'un client visual:

Per exemple MongoDB compass: <https://www.mongodb.com/products/compass>

```
# mongodb://user:password@craaxcloud.epsevg.upc.edu:port
```

Per exemple, per accedir com a admin:

```
# mongodb://admin:admin@craaxcloud.epsevg.upc.edu:36717
```



Backups

Direccions IP de les màquines del CRAAX:

Cloud:	22 → 36022 80 → 36081 443 → 36444 3000 → 36301 27017 → 36717	10.0.5.5
Backups:	22 → 36026	10.0.5.10
Fog:	22 → 36023 53 → 36053 80 → 36080 443 → 36443 3000 → 36300	10.0.5.15
Leaders:	22 → 36024 3000 → 36302 27017 → 36718	10.0.5.20
Edge:	22 → 36025 3000 → 36303	10.0.5.25

Contenidors

Cloud

1- Mirem les IDs dels contenidors que exportarem.

```
ptin@cloud:~/backups_imatges$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND
MES
e371f58e2cc9        digitalocean.com/php "docker-p
p
0fdb19721d07        mongo:latest       "docker-e
_mongo
1a06d42c82bf        nginx:alpine        "nginx -g
bserver
bdc31d32a6a5        cloud_nodejs        "docker-e
dejs
```

2- Exportarem els contenidors

```
ptin@cloud:~/backups_contenidors$ sudo docker export e371f58e2cc9 > cloud_php.tar
ptin@cloud:~/backups_contenidors$ sudo docker export 0fdb19721d07 > cloud_mongo.tar
ptin@cloud:~/backups_contenidors$ sudo docker export 1a06d42c82bf > cloud_nginx.tar
ptin@cloud:~/backups_contenidors$ sudo docker export bdc31d32a6a5 > cloud_nodeJS.tar
ptin@cloud:~/backups_contenidors$ ls
cloud_mongo.tar  cloud_nginx.tar  cloud_nodeJS.tar  cloud_php.tar  mongo.tar
```

3- Extraïem els .tar

```
judit@judit-VirtualBox:~$ scp -P 36022 ptin@craaxcloud.epsevg.upc.edu:/home/ptin/backups_contenidors/* .
```

Fog

1- Mirem les IDs dels contenidors que exportarem:

```
ptin@fog:~/backups_contenidors$ sudo docker ps
[sudo] password for ptin:
CONTAINER ID        IMAGE               COMMAND
NAMES
9145de1b4dc8        digitalocean.com/php "docker-p
app
bafcaac523f1        networkboot/dhcpd   "/entrypc
dhcp_ptin
24210615e303        nginx:alpine        "nginx -g
433/tcp  webserver
ptin@fog:~/backups_contenidors$
```

2- Exportarem els contenidors

```
ptin@fog:~/backups_contenidors$ sudo docker export 9145de1b4dc8 > fog_php.tar
ptin@fog:~/backups_contenidors$ sudo docker export bafcaac523f1 > dhcp.tar
ptin@fog:~/backups_contenidors$ sudo docker export 24210615e303 > nginx.tar
ptin@fog:~/backups_contenidors$ ls
dhcp.tar  fog_php.tar  nginx.tar
```

3- Extraïem els .tar

```
judit@judit-VirtualBox:~$ scp -P 36023 ptin@craaxcloud.epsevg.upc.edu:/home/ptin/backups_contenidors/* .
dhcp.tar 100% 78MB 17.5MB/s 00:04
fog_php.tar 100% 557MB 24.7MB/s 00:22
nginx.tar 100% 98MB 27.2MB/s 00:03
```

Leader

1- Mirem les IDs dels contenidors que exportarem:

```
ptin@leaders:~/backups_contenidors$ sudo docker ps
[sudo] password for ptin:
CONTAINER ID        IMAGE               COMMAND                  CREATED            STATUS              PORTS                               NAMES
6d2c22a9e5a2       leader_nodejs      "docker-entrypoint.s..." 6 days ago        Up 6 days          0.0.0.0:3000->3000/tcp             nodejs
3af752ce24bc       mongo:latest       "docker-entrypoint.s..." 8 days ago        Up 8 days          0.0.0.0:27017-27019->27017-27019/tcp my_mongo
```

2- Exportarem els contenidors

```
ptin@leaders:~/backups_contenidors$ sudo docker export 6d2c22a9e5a2 > lider_NodeJS.tar
ptin@leaders:~/backups_contenidors$ sudo docker export 3af752ce24bc > lider_mongo.tar
ptin@leaders:~/backups_contenidors$ ls
lider_mongo.tar  lider_NodeJS.tar
```

3- Extraïem els .tar

```
judit@judit-VirtualBox:~$ scp -P 36024 ptin@craaxcloud.epsevg.upc.edu:/home/ptin/backups_contenidors/* .
lider_mongo.tar 100% 370MB 22.6MB/s 00:16
lider_NodeJS.tar 100% 903MB 25.1MB/s 00:35
```

Exportem les imatges en la màquina dels backups

```
ptin@backups:~$ scp ptin@10.0.5.5:/home/ptin/backups_contenidors/* .
cloud_mongo.tar 100% 370MB 27.9MB/s 00:13
cloud_nginx.tar 100% 20MB 18.0MB/s 00:01
cloud_nodeJS.tar 100% 903MB 18.1MB/s 00:49
cloud_php.tar 100% 556MB 17.3MB/s 00:32
```

```
ptin@backups:~/imagenes$ scp ptin@10.0.5.15:/home/ptin/backups_contenidors/* .
The authenticity of host '10.0.5.15 (10.0.5.15)' can't be established.
ECDSA key fingerprint is SHA256:pZF0ncs81Ih+oBwZneH9rgf1zJj82iPNMIuXbQ09edw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.5.15' (ECDSA) to the list of known hosts.
dhcp.tar 100% 78MB 42.7MB/s 00:01
fog_php.tar 100% 557MB 19.9MB/s 00:28
nginx.tar 100% 98MB 16.7MB/s 00:05
```

```
ptin@backups:~/imagenes$ scp ptin@10.0.5.20:/home/ptin/backups_contenidors/* .
The authenticity of host '10.0.5.20 (10.0.5.20)' can't be established.
ECDSA key fingerprint is SHA256:pZF0ncs81Ih+oBwZneH9rgf1zJj82iPNMIuXbQ09edw.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.5.20' (ECDSA) to the list of known hosts.
lider_mongo.tar 100% 370MB 26.2MB/s 00:14
lider_NodeJS.tar 100% 903MB 17.6MB/s 00:51
```

```
ptin@backups:~/imagenes$ ls
cloud_mongo.tar cloud_nginx.tar cloud_nodeJS.tar cloud_php.tar dhcp.tar fog_php.tar lider_mongo.tar lider_NodeJS.tar nginx.tar
```

Fitxers

Cloud

```
ptin@cloud:~/backups_fitxers$ ls
cloud_fitxers.tar.gz
ptin@cloud:~/backups_fitxers$ sudo tar -czvf cloud_fitxers.tar.gz ../CLOUD
```

Fog

```
ptin@fog:~/backups_fitxers$ ls
fog_fitxers.tar.gz
ptin@fog:~/backups_fitxers$ sudo tar -czvf fog_fitxers.tar.gz ../FOG
```

Líder

```
tin@leaders:~/backups_fitxers$ ls
lider_fitxers.tar.gz
tin@leaders:~/backups_fitxers$ sudo tar -czvf lider_fitxers.tar.gz ../Leader
```

Exportem els fitxers en la màquina dels backups

```
ptin@backups:~/fitxers$ scp ptin@10.0.5.5:/home/ptin/backups_fitxers/* .
cloud_fitxers.tar.gz                                100% 58MB 53.1MB/s 00:01
ptin@backups:~/fitxers$ scp ptin@10.0.5.15:/home/ptin/backups_fitxers/* .
fog_fitxers.tar.gz                                  100% 64MB 49.2MB/s 00:01
ptin@backups:~/fitxers$ scp ptin@10.0.5.20:/home/ptin/backups_fitxers/* .
lider_fitxers.tar.gz                                100% 34MB 42.6MB/s 00:00
ptin@backups:~/fitxers$ ls
cloud_fitxers.tar.gz fog_fitxers.tar.gz lider_fitxers.tar.gz
```

Instal·lació del programari per fer el backups

```
ptin@backups:~$ sudo apt-get install rsync
```

Crea backup de la base de dades:

mongodump -h direcció:puerto-d base_de_dades -c colecció -u usuari -p
contraseña -o ruta_de_exportació


```
ptin@backups:~/BDD$ ls -la
total 16
drwx----- 4 ptin ptin 4096 may 30 12:55 .
drwxr-xr-x 7 ptin ptin 4096 may 30 11:53 ..
drwxr-xr-x 3 ptin ptin 4096 may 30 12:51 cloud_db
drwxr-xr-x 3 ptin ptin 4096 may 30 12:55 fog_db
ptin@backups:~/BDD$ _
```

Cloud

```
ptin@backups:~/BDD$ mongodump -h 10.0.5.5:27017 -d cloud -u admin -p admin --authenticationDatabase admin -o cloud_db
```

Ara ja tenim totes les dades de les col·leccions de la base de dades del cloud.

```
ptin@backups:~/BDD$ ls
cloud_db
ptin@backups:~/BDD$ ls cloud_db/
cloud
ptin@backups:~/BDD$ ls cloud_db/cloud/
billetes.bson          negocios.bson          persona.bson
billetes.metadata.json negocios.metadata.json persona.metadata.json
ciudades.bson          ofertas.bson          tarjetas.bson
ciudades.metadata.json ofertas.metadata.json tarjetas.metadata.json
intereses.bson         pasajero.bson         vuelos.bson
intereses.metadata.json pasajero.metadata.json vuelos.metadata.json
ptin@backups:~/BDD$
```

Fog

```

ptin@backups:~/BDD$ mongodump -h 10.0.5.20:27017 -d fog -u admin -p admin --authenticationDatabase admin -o fog_db
2020-05-30T12:55:01.933+0200    writing fog.nodos to
2020-05-30T12:55:02.011+0200    done dumping fog.nodos (103 documents)
2020-05-30T12:55:02.052+0200    writing fog.listaespera to
2020-05-30T12:55:02.054+0200    writing fog.pospasajero to
2020-05-30T12:55:02.055+0200    writing fog.coches to
2020-05-30T12:55:02.056+0200    done dumping fog.listaespera (0 documents)
2020-05-30T12:55:02.057+0200    done dumping fog.pospasajero (1 document)
2020-05-30T12:55:02.058+0200    done dumping fog.coches (2 documents)
ptin@backups:~/BDD$ ls fog_db/fog/
coches.bson                listaespera.metadata.json    pospasajero.bson
coches.metadata.json       nodos.bson                   pospasajero.metadata.json
listaespera.bson           nodos.metadata.json
ptin@backups:~/BDD$

```

Restaura la base de datos

mongorestore -h direccion:puerto -d base_de_datos -u usuario -p contraseña
 ruta_del_fichero_.bson_exportado

```

root@debian:/media/usb/db_cloud/cloud# ls
billetes.bson                intereses.metadata.json      pasajero.bson                tarjetas.metadata.json
billetes.metadata.json       negocios.bson                pasajero.metadata.json      vuelos.bson
ciudades.bson                negocios.metadata.json       persona.bson                vuelos.metadata.json
ciudades.metadata.json       ofertas.bson                 persona.metadata.json
intereses.bson               ofertas.metadata.json        tarjetas.bson
root@debian:/media/usb/db_cloud/cloud# mongorestore -h craaxcloud.epsevg.upc.edu:36717 -d cloud -u admin -p admin --authenticationDatabase admin billetes.bson
2020-05-30T13:18:21.392+0200    checking for collection data in billetes.bson
2020-05-30T13:18:21.404+0200    reading metadata for cloud.billetes from billetes.metadata.json
2020-05-30T13:18:21.541+0200    restoring cloud.billetes from billetes.bson
2020-05-30T13:18:21.709+0200    no indexes to restore
2020-05-30T13:18:21.709+0200    finished restoring cloud.billetes (4 documents, 0 failures)
2020-05-30T13:18:21.709+0200    4 document(s) restored successfully. 0 document(s) failed to restore
.
root@debian:/media/usb/db_cloud/cloud# mongorestore -h craaxcloud.epsevg.upc.edu:36717 -d cloud -u admin -p admin --authenticationDatabase admin ciudades.bson
2020-05-30T13:18:30.637+0200    checking for collection data in ciudades.bson
2020-05-30T13:18:30.647+0200    reading metadata for cloud.ciudades from ciudades.metadata.json
2020-05-30T13:18:30.724+0200    restoring cloud.ciudades from ciudades.bson
2020-05-30T13:18:30.866+0200    no indexes to restore
2020-05-30T13:18:30.866+0200    finished restoring cloud.ciudades (5 documents, 0 failures)
2020-05-30T13:18:30.867+0200    5 document(s) restored successfully. 0 document(s) failed to restore
.
root@debian:/media/usb/db_cloud/cloud# mongorestore -h craaxcloud.epsevg.upc.edu:36717 -d cloud -u admin -p admin --authenticationDatabase admin intereses.bson
2020-05-30T13:18:42.890+0200    checking for collection data in intereses.bson
2020-05-30T13:18:42.899+0200    reading metadata for cloud.intereses from intereses.metadata.json
2020-05-30T13:18:42.947+0200    restoring cloud.intereses from intereses.bson
2020-05-30T13:18:43.079+0200    no indexes to restore
2020-05-30T13:18:43.079+0200    finished restoring cloud.intereses (9 documents, 0 failures)
2020-05-30T13:18:43.079+0200    9 document(s) restored successfully. 0 document(s) failed to restore
.
root@debian:/media/usb/db_cloud/cloud# _

```

```
> show dbs
admin      0.000GB
cloud      0.000GB
config     0.000GB
local      0.000GB
test       0.000GB
> use cloud
switched to db cloud
> show collections
billetes
ciudades
intereses
negocios
ofertas
pasajero
persona
tarjetas
vuelos
>
```

```
> show dbs
admin      0.000GB
config     0.000GB
fog         0.000GB
local      0.000GB
persona    0.000GB
test       0.000GB
> use fog
switched to db fog
> show collections
coches
listaespera
nodos
pospasajero
>
```

Avís:

És recomanable fer-ho un a un perquè cada x connexions, impedeix temporalment fer més accessos.

Mesura extrema de seguretat:

Han quedat guardats tots els backups(versió antiga en un .tar.gz)

```
ptin@backups:~$ ls
BDD  fitxers  imatges  mogobackups.tar.gz
ptin@backups:~$
```

Instal·lar MongoDB a la màquina de Backups

1. Accedim a la màquina:

```
# ssh -p 36026 ptin@craaxcloud.epsevg.upc.edu
```

2. Instal·lem allò necessari pel següent pas:

```
# sudo apt-get install gnupg
```

```
# sudo apt-get install wget
```

3. Importem la clau pública:

```
# wget -qO -
```

```
https://www.mongodb.org/static/pgp/server-4.2.asc | sudo  
apt-key add -
```

Si ens dona un **OK** es que tot ha anat correctament.

4. Creem una font d'apt per a MongoDB:

```
# echo "deb http://repo.mongodb.org/apt/debian  
buster/mongodb-org/4.2 main" | sudo tee  
/etc/apt/sources.list.d/mongodb-org-4.2.list
```


5. Actualitzem el repositori:

```
# sudo apt-get update
```

6. Instal·lem el paquet de mongo:

```
# sudo apt-get install -y mongodb-org
```

7. Comprovem que tenim el nou servei:

```
# systemctl status mongod
```

```
ptin@backups:~$ systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor prese
   Active: inactive (dead)
     Docs: https://docs.mongodb.org/manual
```

Veurem que està inactiu.

8. L'habilitem per a que arranqui junt a Debian 10 amb:

```
# sudo systemctl enable mongod
```

```
ptin@backups:~$ sudo systemctl enable mongod
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service →
/lib/systemd/system/mongod.service.
```

9. Arrenquem el servei manualment per a no haver de rebootar la màquina i consultem el seu estat, que ara veurem que està active(running).

```
# sudo systemctl start mongod
```

```
# systemctl status mongod
```

```
ptin@backups:~$ sudo systemctl start mongod
ptin@backups:~$ systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset
   Active: active (running) since Sat 2020-05-30 13:56:10 CEST; 4s ago
     Docs: https://docs.mongodb.org/manual
   Main PID: 2688 (mongod)
    Memory: 117.2M
    CGroup: /system.slice/mongod.service
            └─2688 /usr/bin/mongod --config /etc/mongod.conf

lines 1-8/8 (END)
```


10. Podem accedir ara al shell mongo:

```
# mongo
```

```
ptin@backups:~$ mongo
MongoDB shell version v4.2.7
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("38ffb9f1-b7db-49de-9c63-6d07110f99a7") }
MongoDB server version: 4.2.7
Server has startup warnings:
2020-05-30T13:56:11.539+0200 I STORAGE [initandlisten]
2020-05-30T13:56:11.539+0200 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
2020-05-30T13:56:11.539+0200 I STORAGE [initandlisten] ** See http://dochub.mongodb.org/core/prodnotes-filesystem
2020-05-30T13:56:12.577+0200 I CONTROL [initandlisten]
2020-05-30T13:56:12.577+0200 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-05-30T13:56:12.577+0200 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2020-05-30T13:56:12.577+0200 I CONTROL [initandlisten]
2020-05-30T13:56:12.590+0200 I CONTROL [initandlisten]
2020-05-30T13:56:12.590+0200 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
2020-05-30T13:56:12.590+0200 I CONTROL [initandlisten] ** We suggest setting it to 'never'
2020-05-30T13:56:12.590+0200 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

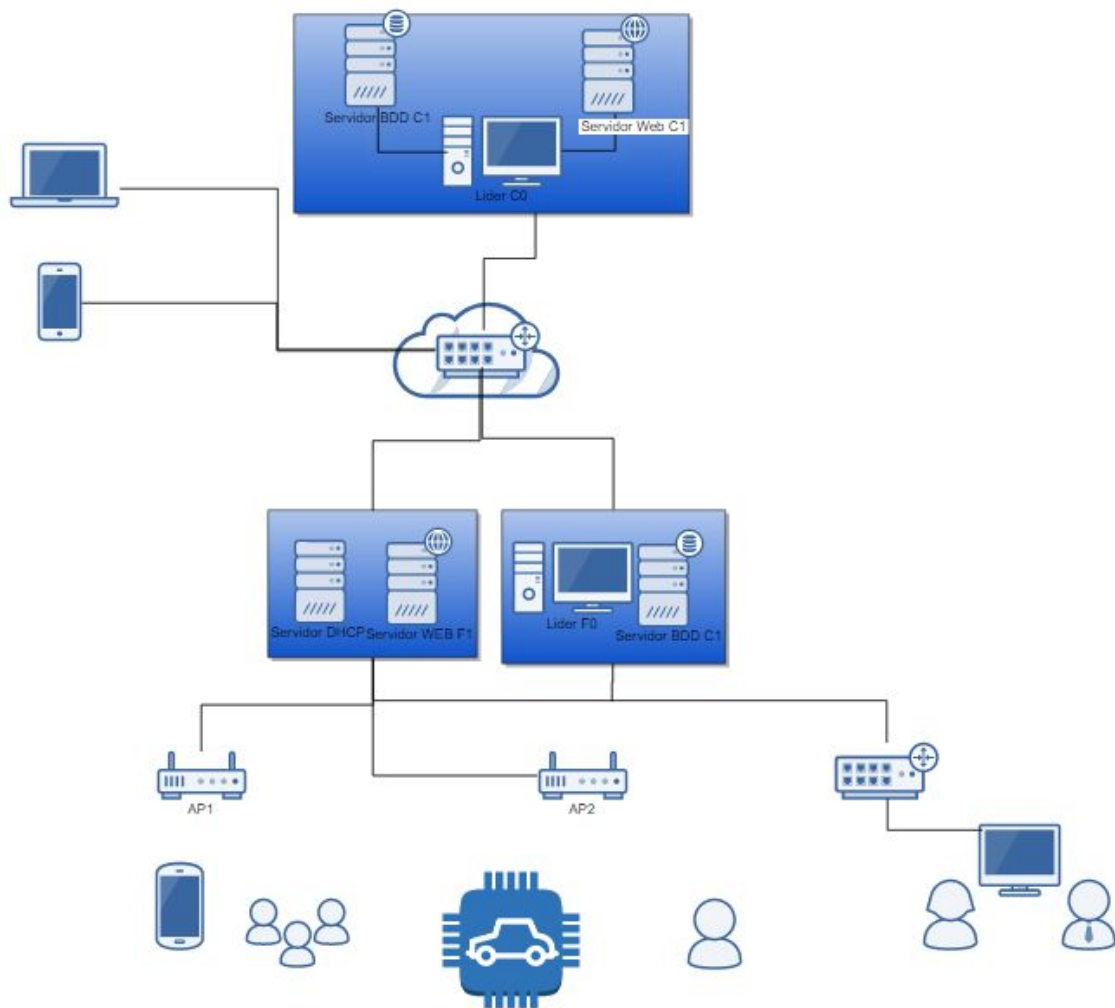
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

> 

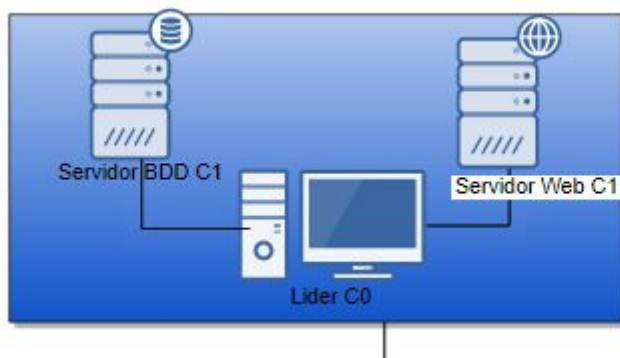
Esquema de comunicació

En aquest document es podrà observar l'arquitectura de tipus F2C de l'aeroport VIA i totes les seves comunicacions. Tot això es veurà de forma esquemàtica per facilitar l'enteniment del funcionament.

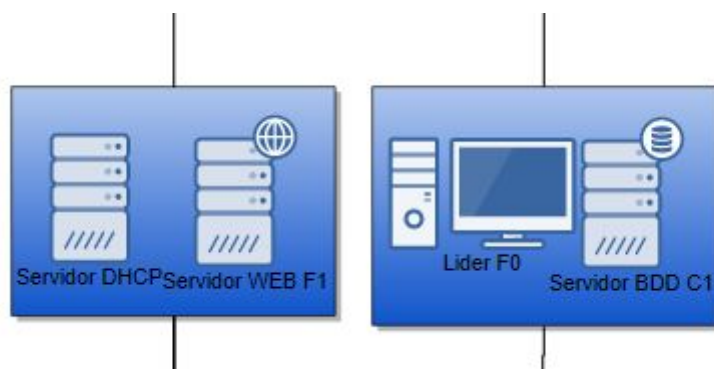
El següent esquema correspon al de tota l'arquitectura



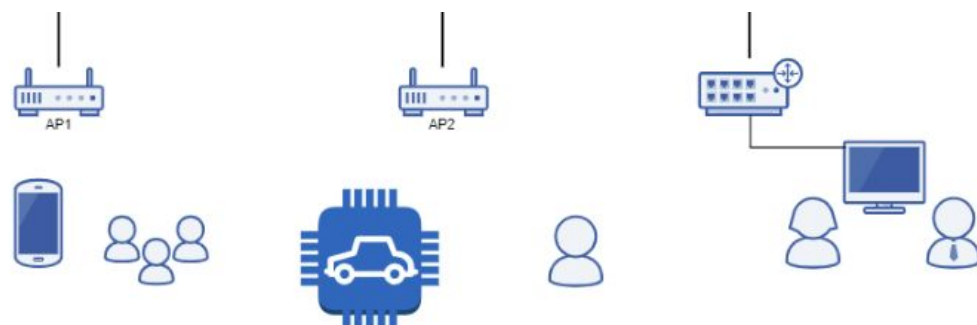
Esquema del cloud



Esquema del Fog

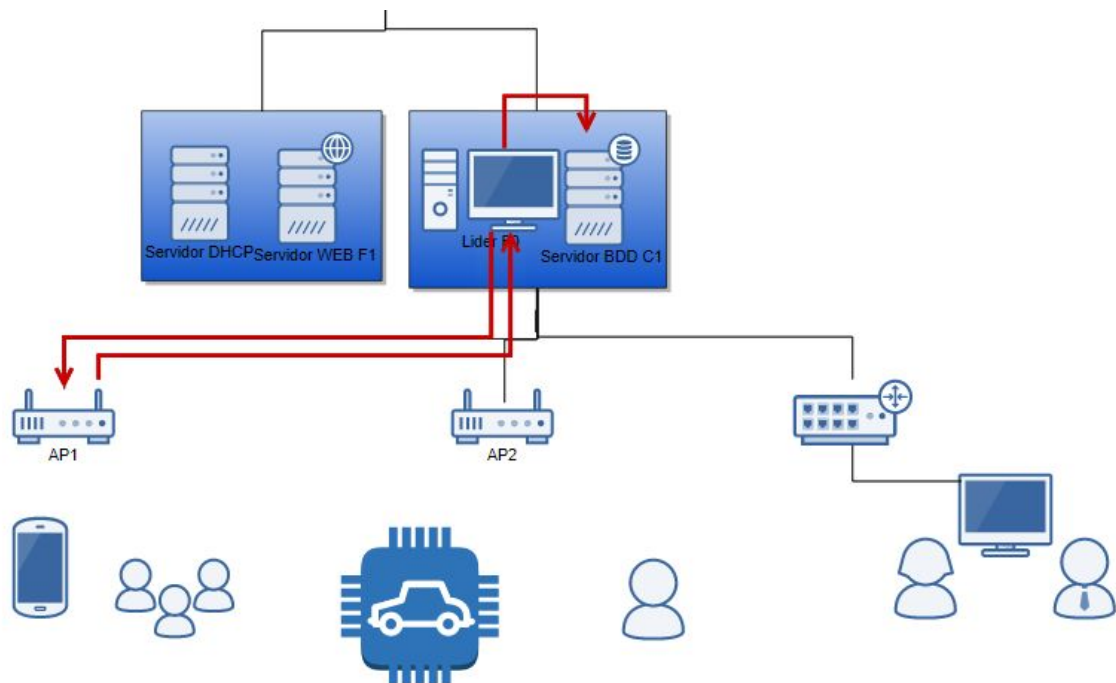


Esquema de l'edge

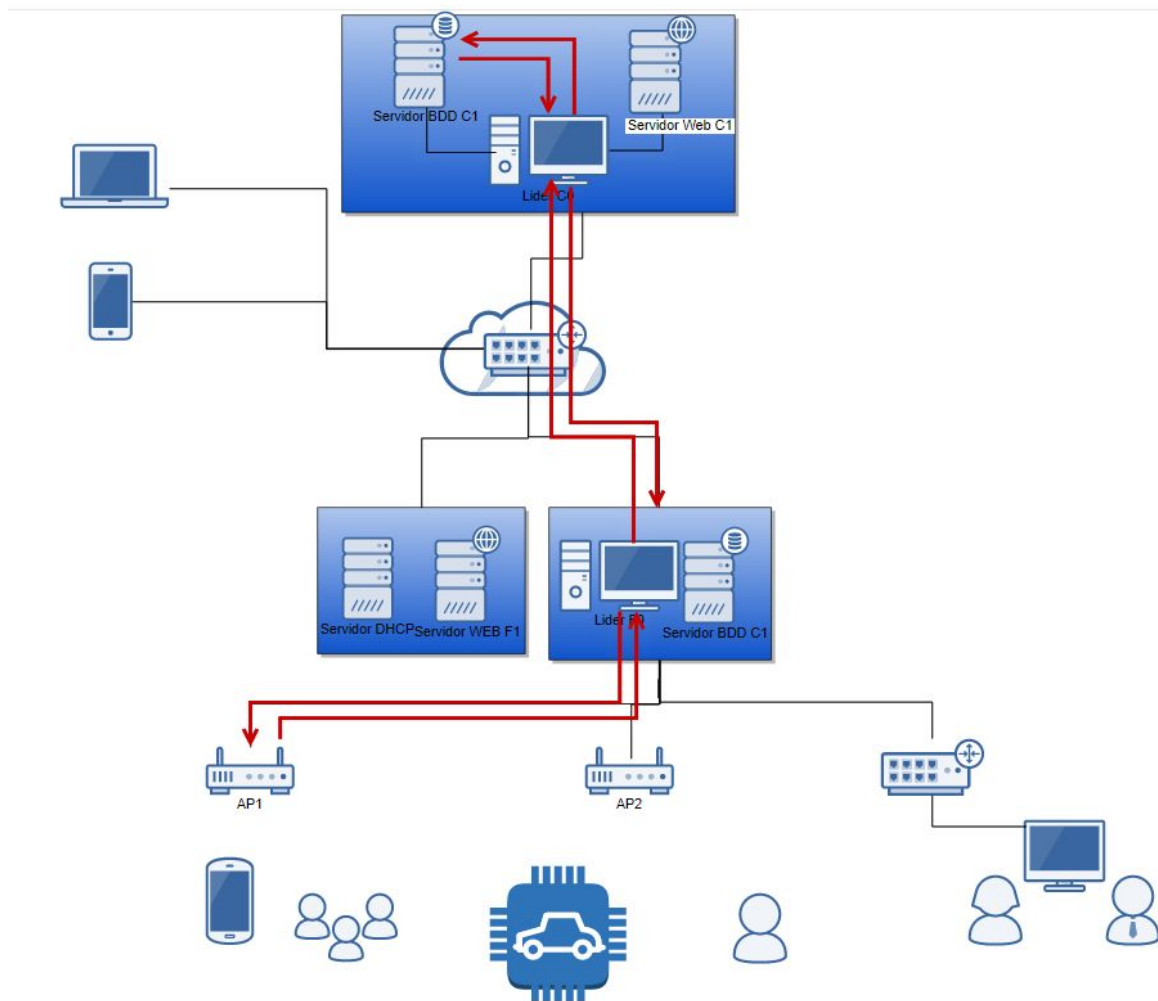


El cas de l'any mòbil necessitarà ser revisat en la seva fase i en la cloud

Per demanar un cotxe, l'aplicació es connectarà a l'API de la capa per fer les diverses peticions i també enviarà la posició dels usuaris. Es pot observar aquesta comunicació en el següent esquema:

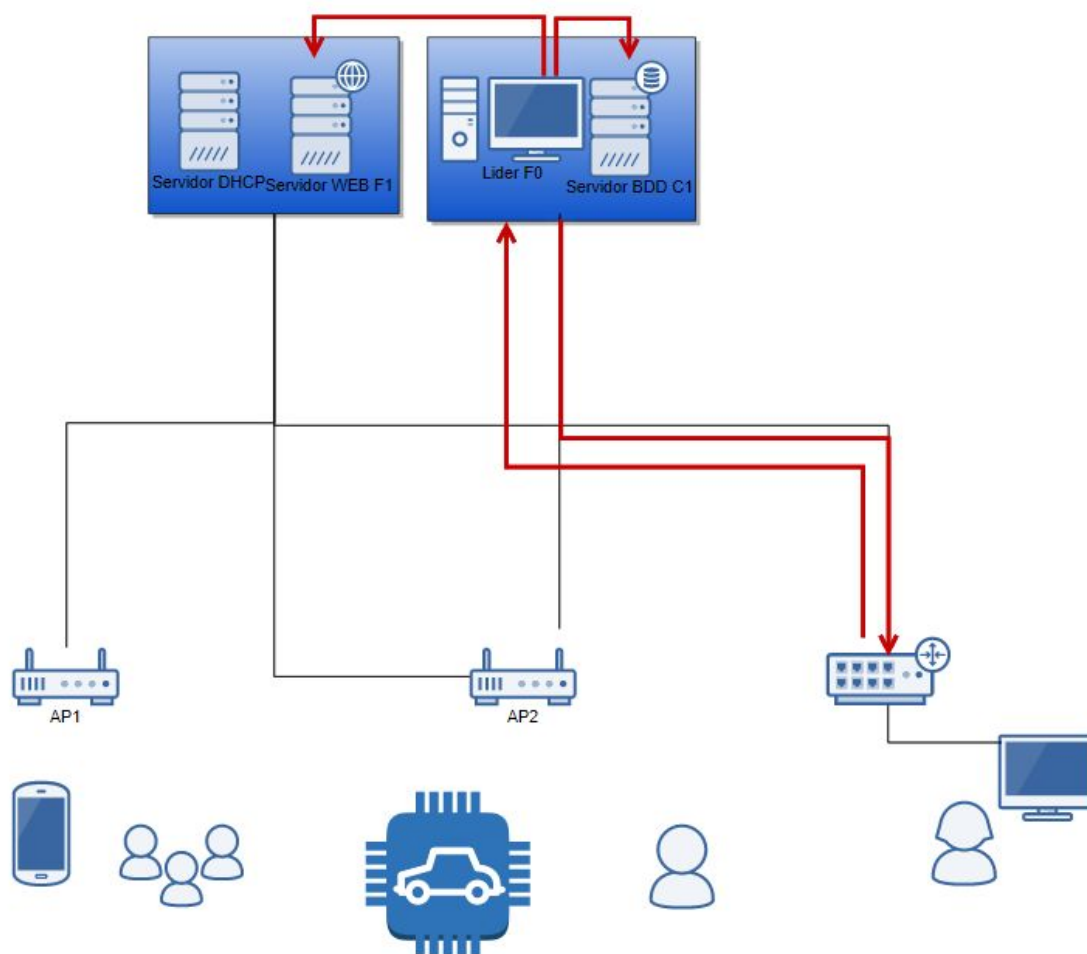


Per altre banda en comunicarà amb l'API del cloud per obtenir les diferents ofertes personalitzades. I l'API ja s'encarregarà d'accedir a la base de dades. Es pot observar aquesta comunicació en el següent esquema:



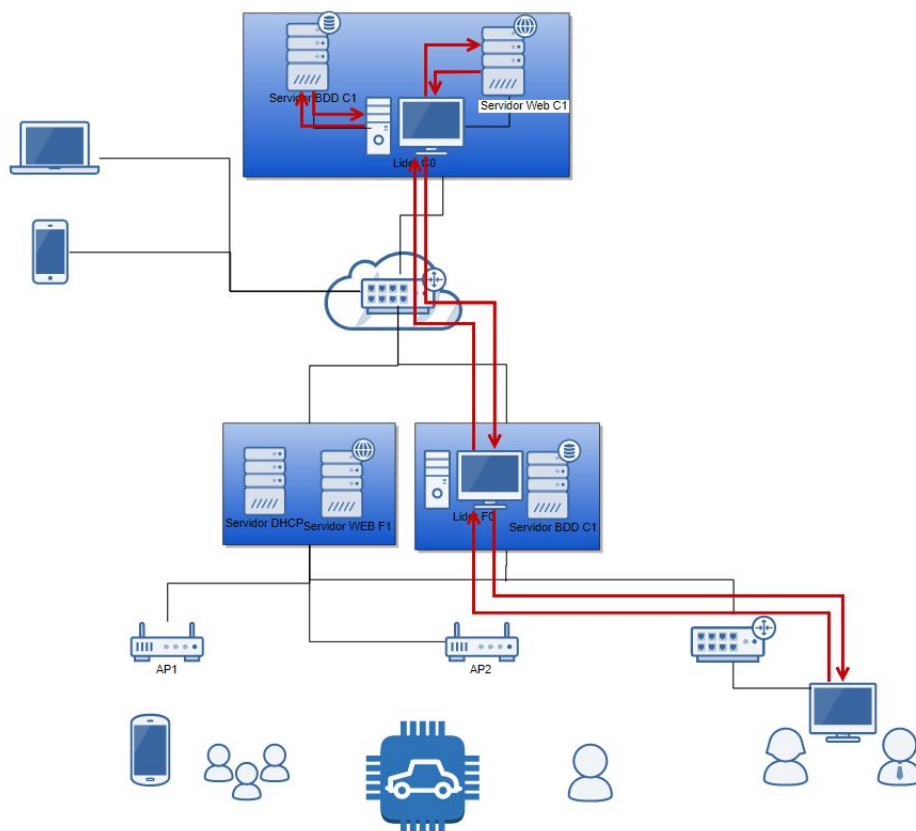
Pàgina web de gestors:

Per accedir a aquesta web, farà falta connectar-se a la capa fog i a l'API del fog per realitzar les diferents peticions per gestionar els gestors que es registren o es loguejen i per fer les peticions per obtenir les diferents ubicacions dels clients i la flota de cotxes i la seva informació pertinent. Es pot observar aquesta comunicació en el següent esquema:



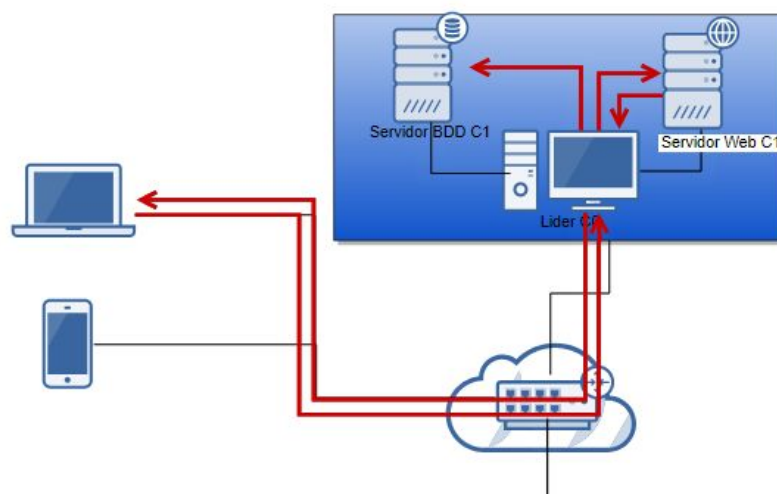
Pàgina web dels Clients

Per accedir a aquesta web, farà falta connectar-se a la capa cloud i a l'API del cloud per realitzar les diferents peticions per gestionar els clients que es registren o es loguejen i per fer les peticions per obtenir les diferents ubicacions dels clients i la flota de cotxes i la seva informació pertinent. Es pot observar aquesta comunicació en el següent esquema:



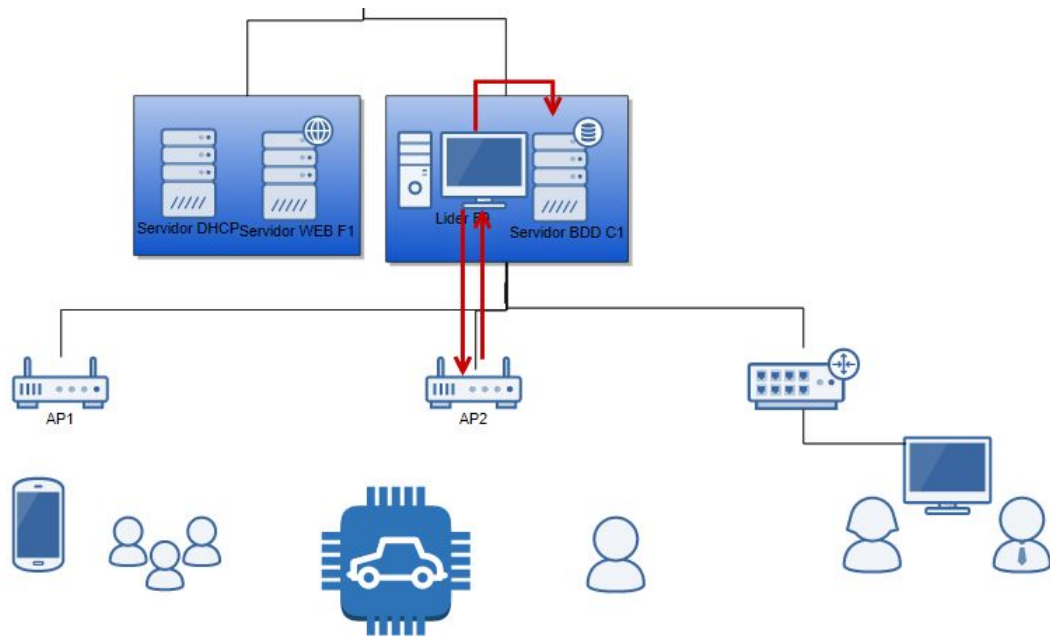
Pàgina web dels Clients (fora l'aeroport)

Per accedir a aquesta web, farà falta connectar-se a la capa cloud i a l'API del cloud per realitzar les diferents peticions dels clients que es registren o es loguejen i obtenir la informació dels perfils. Es pot observar aquesta comunicació en el següent esquema:



Comunicació Flota de cotxes:

El cotxe es comunicarà amb l'API del fog per enviar la seva ubicació cada cop que canvia de punt i farà peticions repetidament per obtenir un client per recollir de la llista d'espera. També la usarà estar pendent de la ubicació dels clients. Es pot observar aquesta comunicació en el següent esquema:

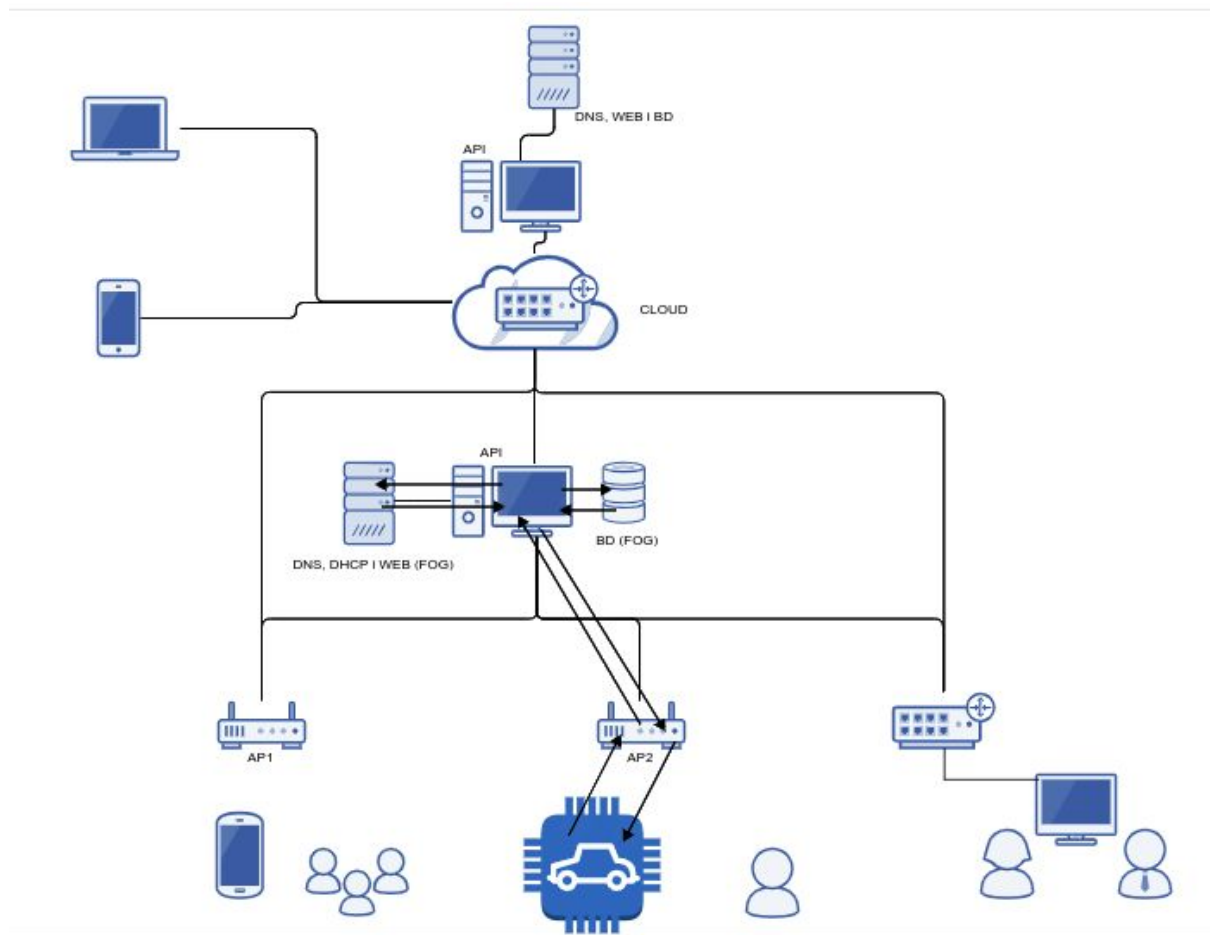


Medi de comunicacions

A continuació expliquem que i com estan connectats cada element com el cotxe, l'app mòbil, la pàgina web... a l'arquitectura, aquesta explicació el fem donant diferents casos de comunicació.

Cotxe autònom:

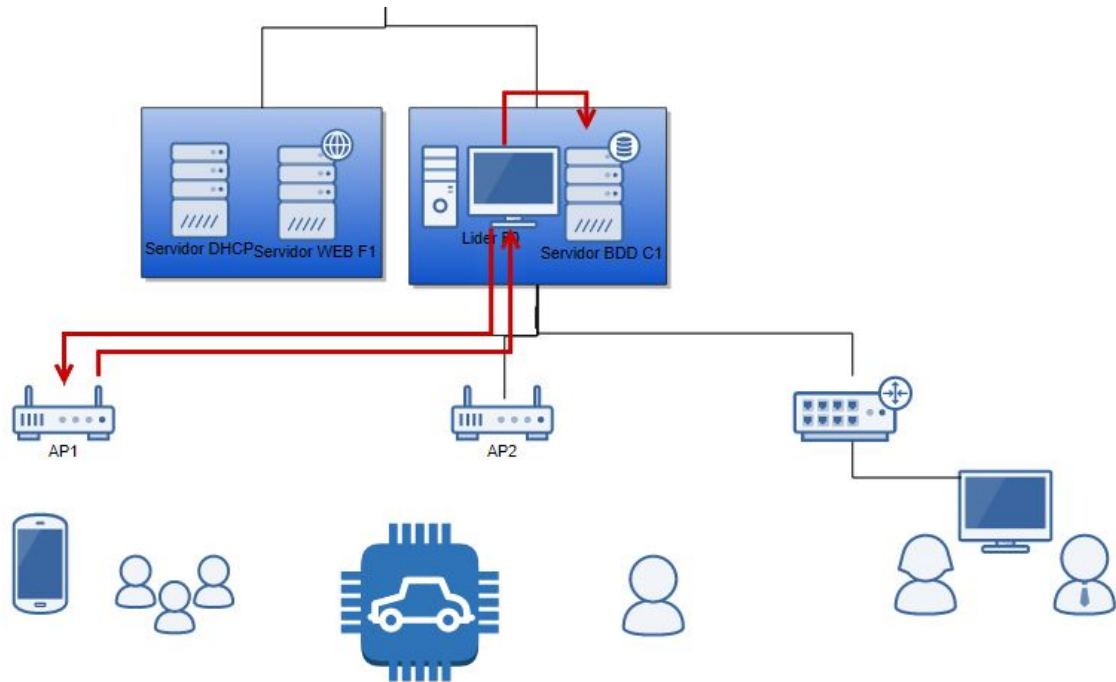
El cotxe autònom està sempre fent peticions GET a l'API mitjançant un script, quan es detecta que hi ha hagut un canvi per exemple: hi ha un passatger per portar a una destinació, llavors es posa en marxa i també envia les seves pròpies dades (com la geolocalització) a l'API fent peticions POST. Aquesta informació el podem observar en el següent esquema:



App mòbil:

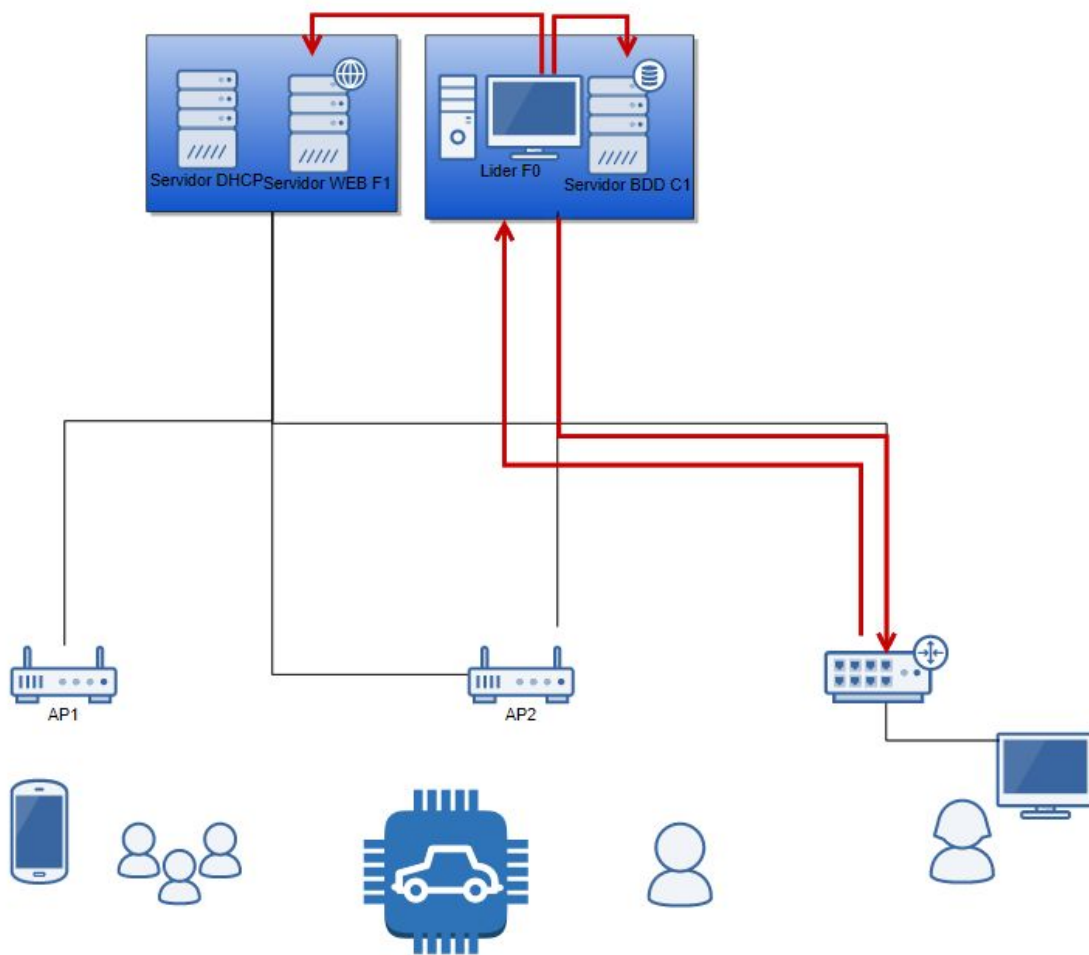
L'aplicació mòbil farà peticions (GET, POST, PUT, DELETE i UPDATE) a la base de dades mitjançant l'API, aquestes peticions es portaran a terme fent servir missatges json internament. La connexió a l'API REST des d'Android comporta el mateix funcionament que si féssim una connexió des d'Android a qualsevol recurs que funcioni amb el protocol http, en l'aplicació mòbil en lloc de fer les peticions GET, POST, PUT, DELETE i UPDATE es faran les peticions HttpGet, HttpPost, HttpPut i HttpDelete. Per exemple: quan des de l'aplicació mòbil se sol·licita un cotxe, en aquest cas internament es farà una petició httpget a l'API que retornarà la localització del cotxe però en aquest cas també s'haurà d'enviar a l'API amb una

petició httput la localització de l'usuari qui fa la petició mitjançant l'aplicació. A continuació podem veure l'esquema d'aquesta comunicació:



Pàgina web:

La pàgina web també farà peticions (GET, POST, PUT, DELETE i UPDATE) a la base de dades mitjançant l'API, però aquestes peticions es faran quan a la pàgina web hi hagi alguna interacció que ho requereixi. Per exemple: quan els gestors volen saber informació relacionada amb els vols, en aquest cas la pàgina web (html) farà una petició, i rebrà aquesta informació en format json i llavors la convertirà en un objecte javascript i ho mostrarà per pantalla. Aquesta comunicació el podem observar en el següent esquema:



Disponibilitat

Què és l'alta disponibilitat?

L'alta disponibilitat s'encarrega de dissenyar i garantir la disponibilitat i el funcionament dels serveis en tot moment.

Alta disponibilitat en el Cloud i el Fog?

L'alta disponibilitat és un dels factors clau i un dels més importants del núvol. La idea de tenir un accés en qualsevol moment i en qualsevol lloc, a un servei, eina o dades, és el que dóna sentit a la idea de Cloud i Fog. També es pot afirmar que si un servei té una alta disponibilitat, el sistema serà també molt fiable.

Però els sistemes no poden garantir una disponibilitat del 100% en qualsevol infraestructura. Però sí que es poden donar solucions perquè el percentatge sigui el més pròxim a la perfecció.

Solucions per arribar a obtenir una alta disponibilitat?

- Redundància o duplicació de recursos:

Per fer que el sistema sigui segur, s'ha de fer que tingui la capacitat de ser tolerant a falles. Per fer-ho s'ha de duplicar els recursos crítics.

- Suport de recursos crítics:

Els **elements de hardware** redundants, hauran de ser capaços de ser **reemplaçats en calent**, sense interrompre el servei.

- Sistema de Clústers:

Conjunt de dos o més equips que tenen un sèrie de serveis compartits i estan constantment monitorats entre ells.

En el moment que un dels equips caigui, algun software serà capaç d'arrancar els serveis en una altra màquina operativa, sense que els usuaris detectin aquesta caiguda.

Hi ha 2 tipus de software per detectar les caigudes:

- Alta disponibilitat d'infraestructura: detecta els errors de hardware.
- Alta disponibilitat d'aplicació: detecta els errors de hardware i d'aplicacions.

Altres coses a tindre en compte

- Balanceig de càrrega:

Minimiza el temps de resposta i evita la saturació del servei.

- Monitorització:

Si vigilem constantment el nostre sistema, al detectar qualsevol error, podrem actuar.

Manteniment de docker-compose

Passos previs:

1. Hem de informar els equips que estan treballant actualment per evitar confusions en cas que estiguin fent proves
2. Ens situem al directori del docker-compose on volem reconstruir

Passos per reiniciar un contenidor en concret:

1. Saber com es diu el contenidor que volem reiniciar (primera columna)
`sudo docker-compose ps`
2. Parar i esborrar el contenidor de docker
`sudo docker-compose stop nom_contenidor`
`sudo docker-compose rm nom_contenidor`
3. (Opcional) Esborrar el contenidor (fitxer)
`sudo docker system prune -a`
4. Tornar a construir i aixecar el contenidor
`sudo docker-compose build nom_contenidor`
`sudo docker-compose up (-d)`

Si quan construïm el contenidor (build) **tots** els passos (si el contenidor es construeix amb un Dockerfile) han sigut carregats de cache és possible que o no hi hagi canvis, o s'hagi de fer un reinici de manteniment

Passos per fer un reinici de manteniment:

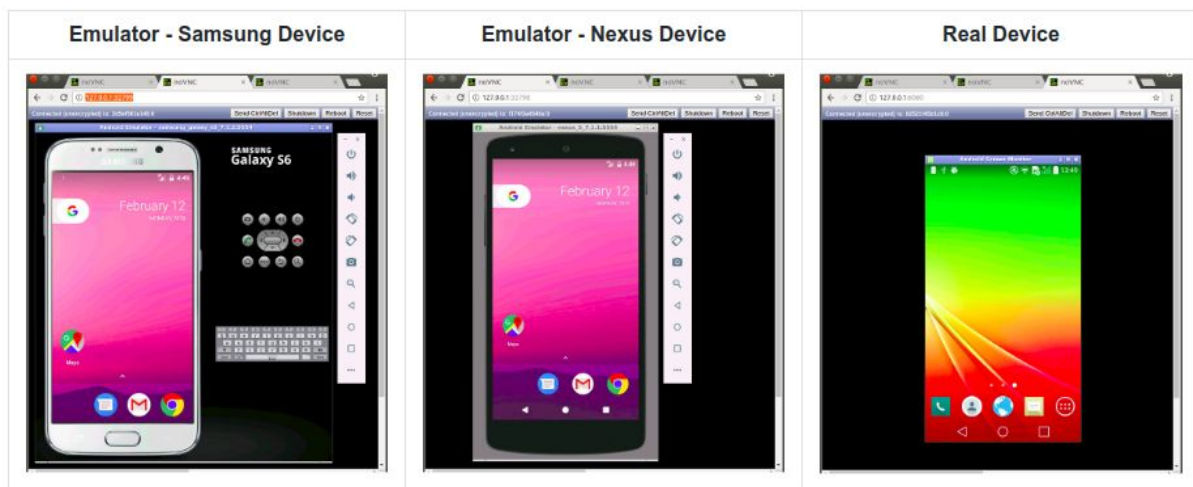
1. Aturar tots els contenidors i volums
`sudo docker-compose down -v`
2. Esborrem tots els contenidors, volums, imatges i xarxes que estan parats o que ja no s'utilitzen (tots en teoria)
`sudo docker system prune -a`
3. Tornem a aixecar els contenidors (com hem esborrat tot, l'"up" construeix tot una altra vegada)
`sudo docker-compose up`

Finalment

No ens hem d'oblidar de fer un “`sudo docker-compose ps`” per veure si tot funciona bé

Construint una aplicació Android amb docker-android

Docker-android és una imatge docker per utilitzar tot allò relacionat amb “mobile website testing” i “Android project”. Uns exemples de com emular una aplicació en diferents dispositius que permet aquesta imatge és el següent:



Amb aquesta imatge podem aconseguir:

1. Fer proves UI per pàgines web per mòbil amb appium
2. Construir projectes Android i executar proves unit amb les eines de construcció més actualitzades
3. Fer proves UI per aplicacions mòbils amb diferents frameworks com appium, espresso, robotium, etc.
4. Proves SMS

Avantatges que té aquesta imatge:

1. Té el programa noVNC per saber que passa a l'interior del contenidor docker.

2. Aporta un emulador per diferents dispositius mòbil com Samsung Galaxy S6, LG Nexus 4, HTC Nexus,etc.
3. Té l'habilitat de connectar amb Selenium Grid.
4. Té l'habilitat per controlar el emulador desde fora del contenidor fent servir adb connect.

La llista de dispositius per els quals hi ha un emulador és el següent:

Type	Device Name
Phone	Samsung Galaxy S10
Phone	Samsung Galaxy S9
Phone	Samsung Galaxy S8
Phone	Samsung Galaxy S7 Edge
Phone	Samsung Galaxy S7
Phone	Samsung Galaxy S6
Phone	Nexus 4
Phone	Nexus 5
Phone	Nexus One
Phone	Nexus S
Tablet	Nexus 7

Requeriments

Tenir el docker instal·lat al sistema.

Primers passos

1. Executem l'imatge docker-android:

```
docker run --privileged -d -p 6080:6080 -p 5554:5554 -p
5555:5555 -e DEVICE="Samsung Galaxy S6" --name
android-container budtmo/docker-android-x86-8.1
```

2. Comprovem la direcció ip del host de docker

```
docker-machine ip default
```

3. Posem <http://docker-host-ip-address:6080> al navegador.

Configurar la imatge

Si volem configurar la imatge docker-android ho podem fer seguint aquesta [documentació](#).

Construïm un projecte android

1. Clonem:

```
git clone git@github.com:android/testing-samples.git
```

2. Construïm el projecte

```
docker run -it --rm -v
```

```
$PWD/testing-samples/ui/espresso/BasicSample:/tmp -w /tmp
```

```
budtmo/docker-android-x86-8.1 /tmp/gradlew build
```

Controlem el dispositiu android connectat al host (emulador o dispositiu real)

1. Creem un contenidor docker amb aquesta comanda:

```
$ docker run --privileged -d -p 6080:6080 -p 5554:5554 -p
5555:5555 -p 4723:4723 --name android-container-appium
budtmo/docker-android-real-device
```

2. Obrim noVNC accedint <http://localhost:6080>

3. Obrim terminal fent clic sobre noVNC window >> Terminal emulator

4. Per connectar al programa adb del host fem:

```
$ adb -H host.docker.internal devices
```

Per especificar el port fem:

```
$ adb -H host.docker.internal -P 5037 devices
```

5. Ara el nostre contenidor pot accedir als dispositius hosts. Però, abans tenim que afegir a . i a . les capacitats desitjades per fer que Appium reconeixi aquests dispositius.

Appium i Selenium Grid

Podem seguir aquesta [documentació](#) si es vol fer servir Appium i Selenium Grid. La [documentació](#) exemples i casos simples.

Controlem el emulador android desde fora el contenidor

Executem la següent comanda:

```
adb connect <docker-machine-ip-address>:5555
```

Simulació de SMS

1. Fent servir telnet

Cal buscar la variable auth_token i copiar-ho:

```
docker exec -it android-container cat  
/root/.emulator_console_auth_token
```

Accedim al emulador amb telnet i fem el login amb auth_token

```
telnet <docker-machine-ip-address> 5554
```

Fem el login amb auth_token

```
auth <auth_token>
```

Enviem el SMS

```
sms send <phone_number> <message>
```

2. Fent servir adb, executem la següent comanda:

```
docker exec -it android-container adb emu sms send  
<phone_number> <message>
```

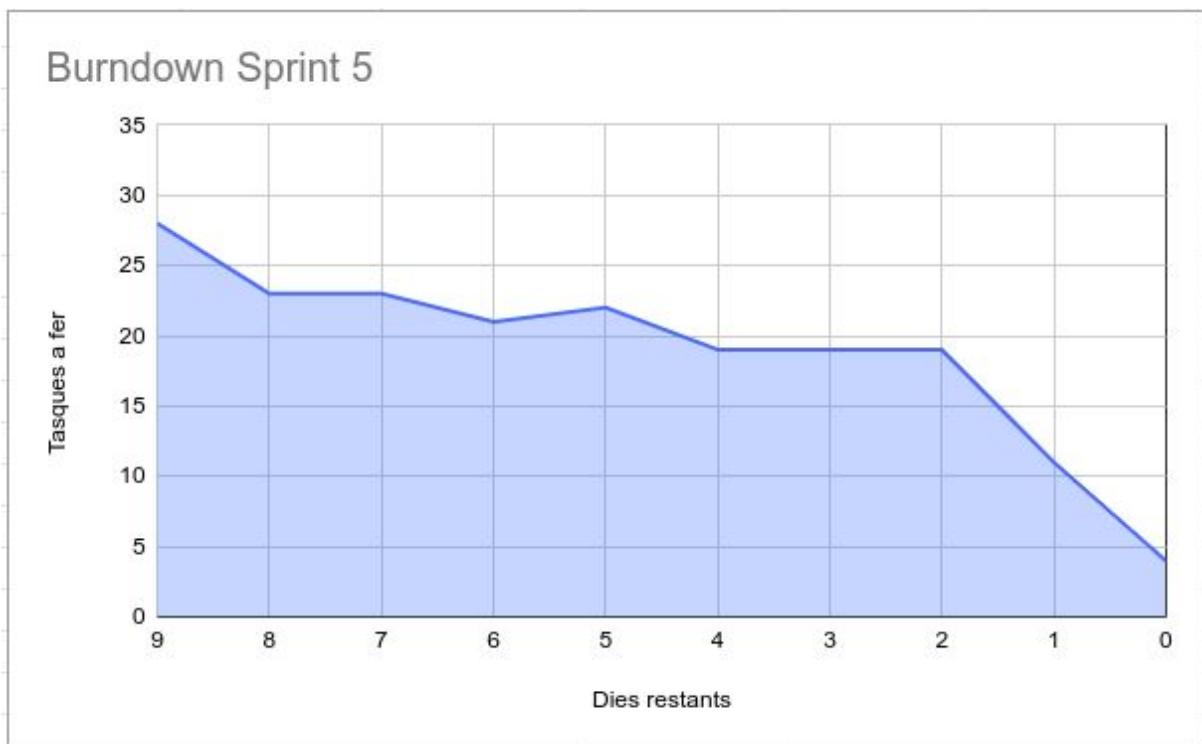
Kubernetes

Què és Kubernetes?

És un “orquestrador de contenidors”, és una plataforma de codi obert. Per una descripció més detallada podeu consultar la següent pàgina web:

<https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>

Burndown



Webgrafía

<https://nodeymongo.wordpress.com/2014/06/12/exportacion-e-importacion-de-datos-en-mongodb/>

<http://rcg-comunicaciones.com/disponibilidad-tus-redes-sistemas-comunicaciones/>

<https://github.com/budtmo/docker-android>

<https://appium.io/>

https://github.com/budtmo/docker-android/blob/master/README_CUSTOM_CONFIG.md

https://github.com/budtmo/docker-android/blob/master/README_APPIUM_AND_SELENIUM.md

<https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>

