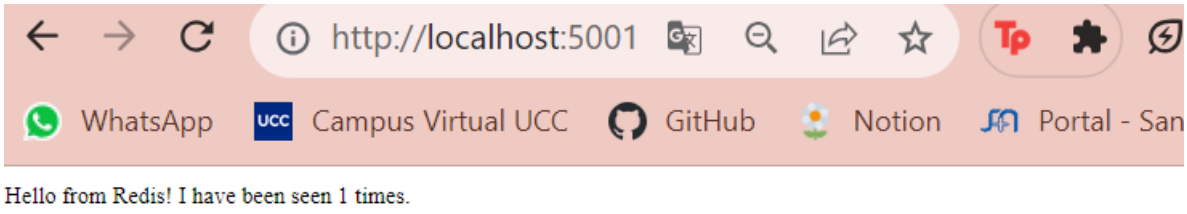
Practico 15/08

1. Sistema Distribuido Simple




```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker ps -a
CONTAINER ID   IMAGE                      COMMAND                  CREATED
 STATUS          PORTS                     NAMES
b3fa6d3bdfa6   alexisfr/flask-app:latest   "python /app.py"         44 seconds ago
 Up 42 seconds   0.0.0.0:5001->5000/tcp   web
98187fb66030   redis:alpine               "docker-entrypoint.s…"   6 minutes ago
 Up 6 minutes    6379/tcp                 db
```

---

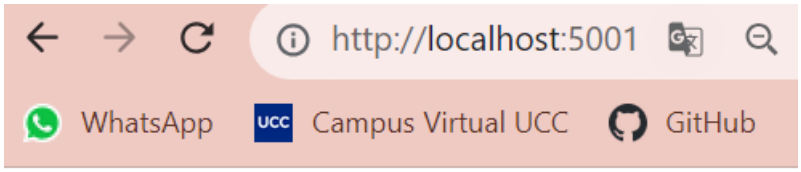
2. Análisis del Sistema

Fuerzas el borrado del contenedor

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker rm -f web
web

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS
  PORTS      NAMES
98187fb66030   redis:alpine   "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes
  6379/tcp   db
```

- `docker run -d --net mybridge -e REDIS_HOST=db -e REDIS_PORT=6379 -p 5001:5000 --name web alexisfr/flask-app:latest`



Hello from Redis! I have been seen 2 times.

Aunque lo eliminé al contenedor sigue el contenido en la pagina web porque no necesariamente eliminamos el servicio que se ejecutaba dentro del contenedor.

> 💡 La eliminacion del contenedor no elimina automaticamente los recursos que el contenedor pudo haber dejado atras, como volumenes o imagenes.

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker rm -f db
db

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker ps
CONTAINER ID   IMAGE                      COMMAND            CREATED              S
TATUS              PORTS                      NAMES
4570711c52a2   alexisfr/flask-app:latest   "python /app.py"   About a minute ago   U
p About a minute   0.0.0.0:5001->5000/tcp   web
```

> 💡 Vuelvo a tener el contenedor web y hay un **volumen** para que puedan persistir los datos

Los volúmenes en Docker permiten separar los datos del contenedor, lo que significa que los datos persisten incluso si el contenedor se detiene o se elimina.

Ahora limpio todo

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker rm -f db
Error: No such container: db

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker rm -f web
web

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker network rm mybridge
mybridge
```

---

3. Docker compose

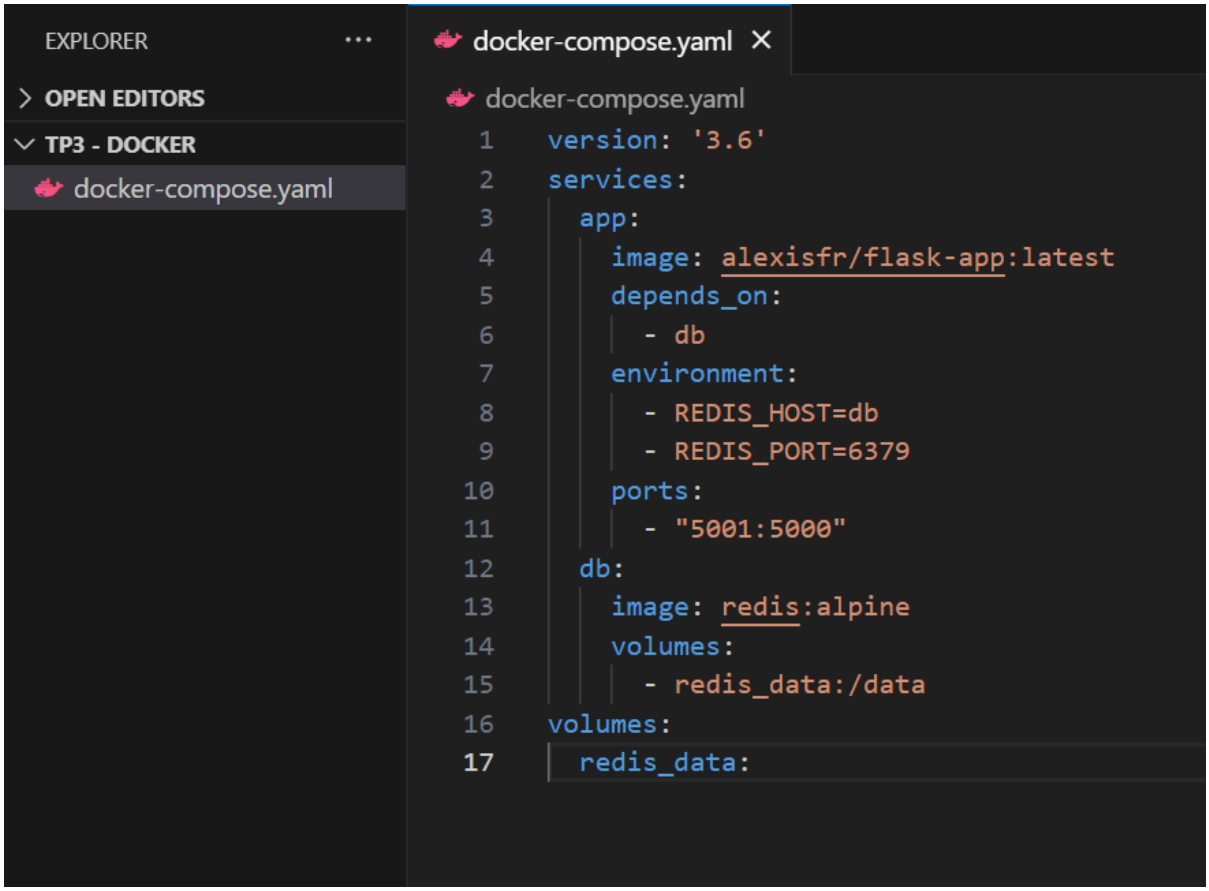
La version del docker (version es el formato del docker compose)

Vamos a levantar 2 servicios (2 contenedores)

1. app
2. db

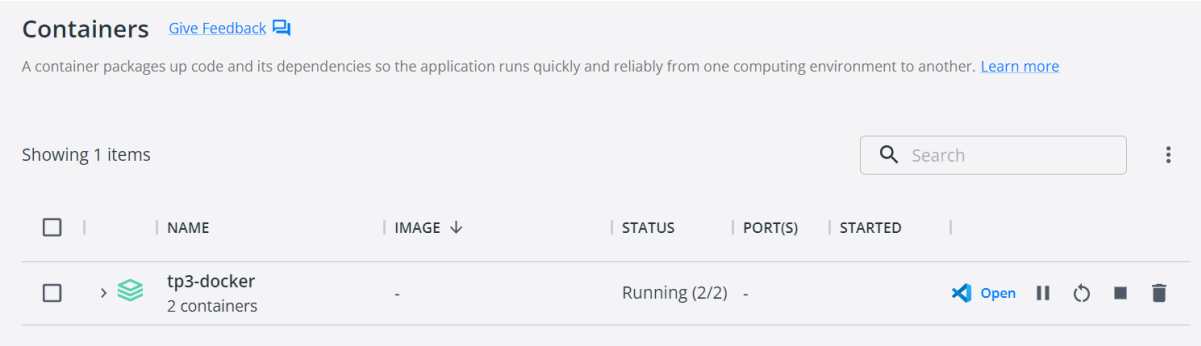
Los datos que poniamos en el comando estaran explicados en ele docker compose

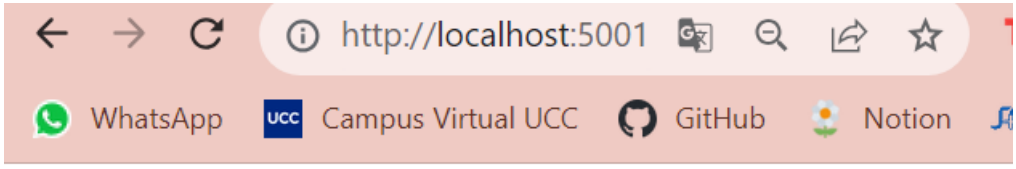
> 💡 ***depends on:*** dice que no levantes tal contenedor hasta que este levantado el que digo aca (hace que siga un orden)


EXPLORER
OPEN EDITORS
TP3 - DOCKER
docker-compose.yaml
docker-compose.yaml
docker-compose.yaml
1 version: '3.6'
2 services:
3 app:
4 image: alexisfr/flask-app:latest
5 depends_on:
6 - db
7 environment:
8 - REDIS_HOST=db
9 - REDIS_PORT=6379
10 ports:
11 - "5001:5000"
12 db:
13 image: redis:alpine
14 volumes:
15 - redis_data:/data
16 volumes:
17 redis_data:

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker-compose up
Creating network "tp3-docker_default" with the default driver
Creating volume "tp3-docker_redis_data" with default driver
Creating tp3-docker_db_1 ... done
Creating tp3-docker_app_1 ... done
Attaching to tp3-docker_db_1, tp3-docker_app_1
db_1   | 1:C 15 Aug 2023 17:51:06.071 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0O
o
db_1   | 1:C 15 Aug 2023 17:51:06.071 # Redis version=7.0.12, bits=64, commit=000000
00, modified=0, pid=1, just started
db_1   | 1:C 15 Aug 2023 17:51:06.071 # Warning: no config file specified, using the
 default config. In order to specify a config file use redis-server /path/to/redis.c
onf
db_1   | 1:M 15 Aug 2023 17:51:06.072 * monotonic clock: POSIX clock_gettime
db_1   | 1:M 15 Aug 2023 17:51:06.073 * Running mode=standalone, port=6379.
db_1   | 1:M 15 Aug 2023 17:51:06.073 # Server initialized
db_1   | 1:M 15 Aug 2023 17:51:06.073 # WARNING Memory overcommit must be enabled! W
ithout it, a background save or replication may fail under low memory condition. Bei
ng disabled, it can can also cause failures without low memory condition, see https:
//github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_mem
ory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommi
t_memory=1' for this to take effect.
db_1   | 1:M 15 Aug 2023 17:51:06.074 * Ready to accept connections
app_1  |  * Serving Flask app "app" (lazy loading)
app_1  |  * Environment: production
app_1  |    WARNING: Do not use the development server in a production environment.
app_1  |    Use a production WSGI server instead.
app_1  |  * Debug mode: on
app_1  |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
app_1  |  * Restarting with stat
app_1  |  * Debugger is active!
app_1  |  * Debugger PIN: 236-306-022
```

tengo agrupados todos los contenedores que puse dentro del compose





Hello from Redis! I have been seen 6 times.

```
app_1  |  * Restarting with stat
app_1  |  * Debugger is active!
app_1  |  * Debugger PIN: 236-306-022
app_1  | 172.20.0.1 - - [15/Aug/2023 17:52:45] "GET / HTTP/1.1" 200 -
app_1  | 172.20.0.1 - - [15/Aug/2023 17:52:53] "GET / HTTP/1.1" 200 -
app_1  | 172.20.0.1 - - [15/Aug/2023 17:52:53] "GET / HTTP/1.1" 200 -
app_1  | 172.20.0.1 - - [15/Aug/2023 17:52:54] "GET / HTTP/1.1" 200 -
app_1  | 172.20.0.1 - - [15/Aug/2023 17:52:56] "GET / HTTP/1.1" 200 -
app_1  | 172.20.0.1 - - [15/Aug/2023 17:52:57] "GET / HTTP/1.1" 200 -
```

6 veces que recargue la pagina

Después puedo darlo de baja si quiero

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ docker-compose down
Removing tp3-docker_app_1 ... done
Removing tp3-docker_db_1  ... done
Removing network tp3-docker_default
```

---

4. Análisis de otro sistema distribuido

Un poco mas complejo que el anterior porque implica mas servicios

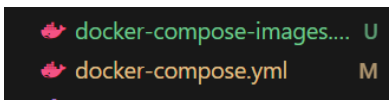
A cada contenedor que corre de manera independiente no le interesa el lenguaje en el que está programado.

Clono el repo

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeSoftware3/T
P3 - Docker (main)
$ git clone https://github.com/dockersamples/example-voting-app.git
Cloning into 'example-voting-app'...
remote: Enumerating objects: 1088, done.
remote: Total 1088 (delta 0), reused 0 (delta 0), pack-reused 1088 eceiving objects:
  93% (1012/1088), 1.09 MiB | 1.08 MiB/s
Receiving objects: 100% (1088/1088), 1.14 MiB | 1.10 MiB/s, done.
Resolving deltas: 100% (408/408), done.
```

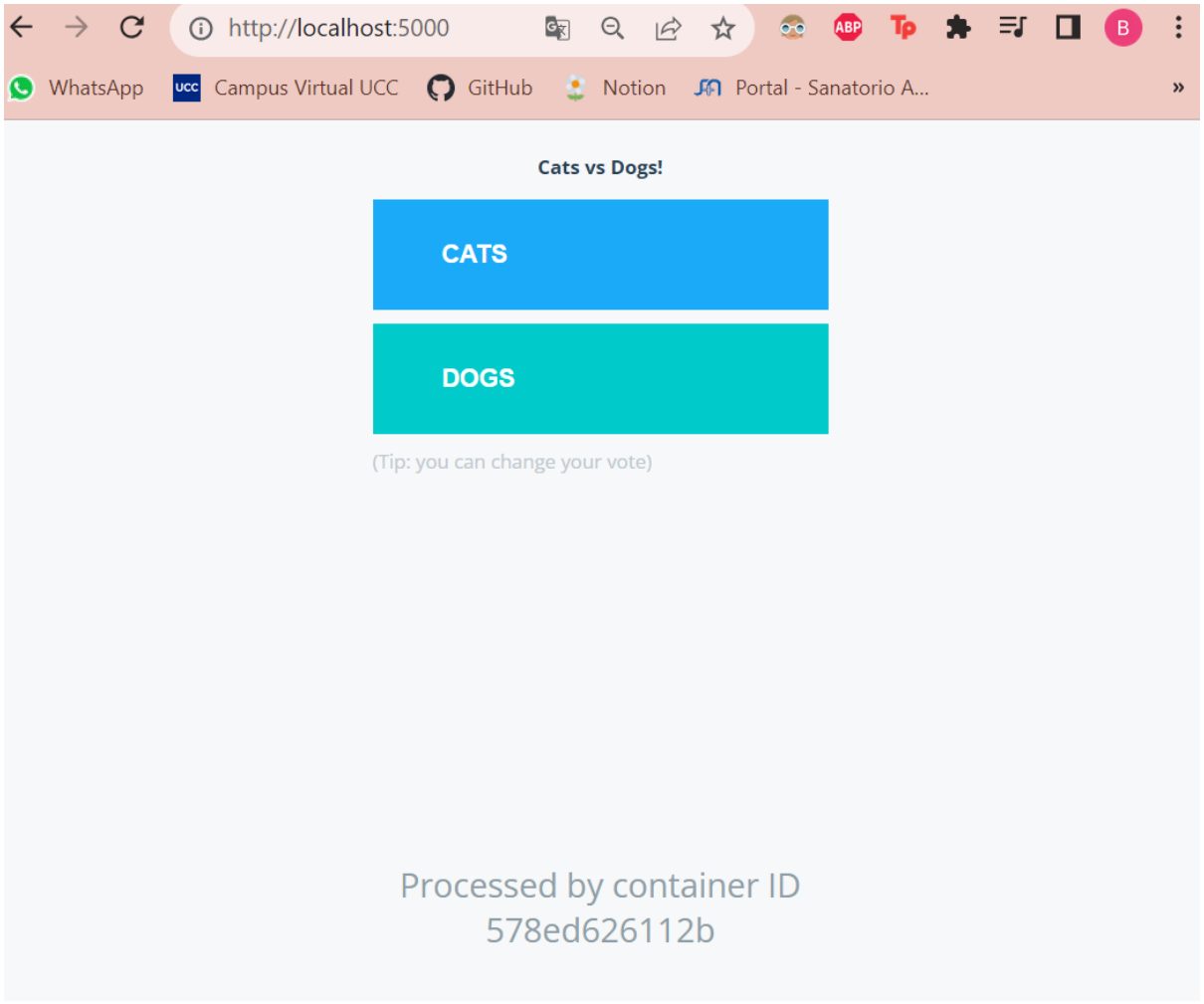
Hago `docker-compose -f docker-compose.yml up -d`

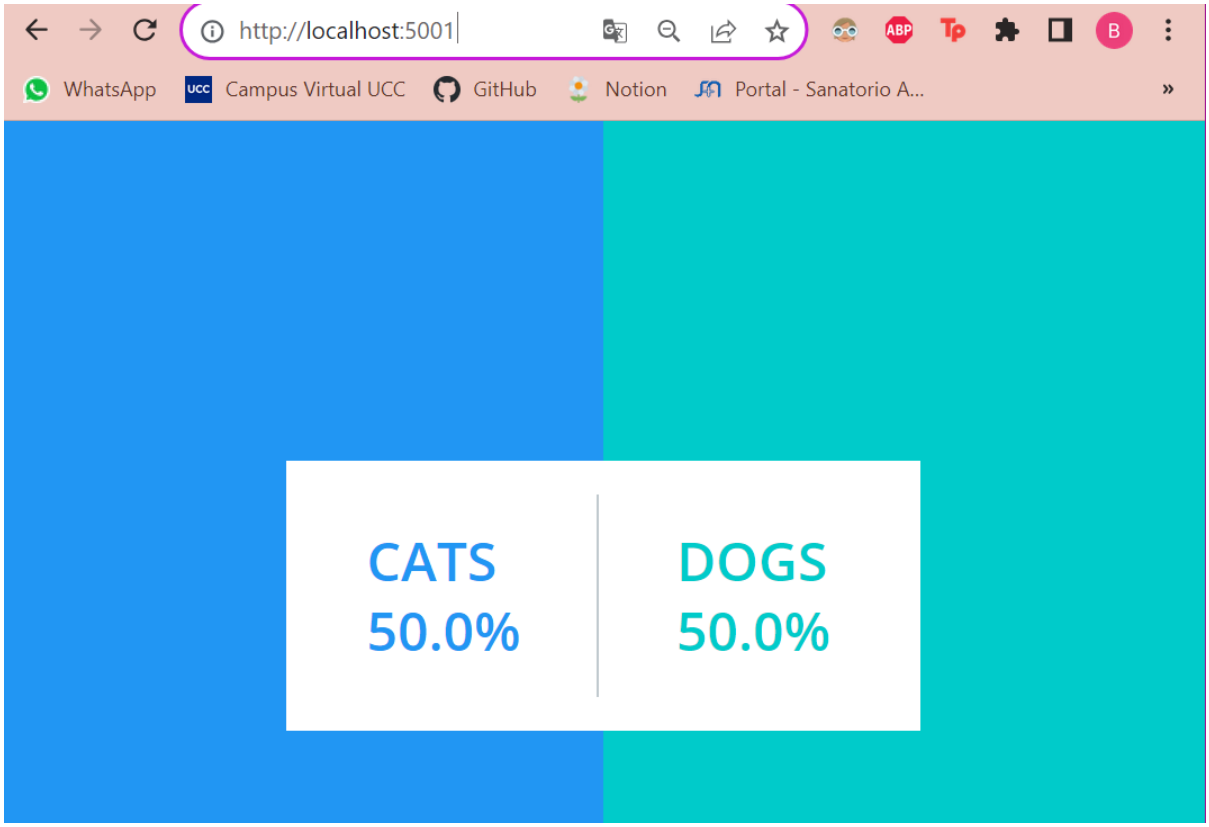
📢 cambié nombres de carpetas



Hago compose…

```
BELU@belenaguilarv MINGW64 ~/go/src/github.com/belenaguilarv/IngenieriaDeS
oftware3/TP3 - Docker/example-voting-app (main)
$ docker-compose -f docker-compose.yml up -d
Creating network "example-voting-app_back-tier" with the default driver
Creating network "example-voting-app_front-tier" with the default driver
Creating example-voting-app_db_1    ... done
Creating example-voting-app_redis_1 ... done
Creating example-voting-app_vote_1   ... done
Creating example-voting-app_worker_1 ... done
Creating example-voting-app_result_1 ... done
```





Voting App (Python): Aplicación web que permite emitir votos entre dos opciones. Se ejecuta en el puerto 5000.

Result App (Node.js): Aplicación web que muestra los resultados de la votación en tiempo real. Se ejecuta en el puerto 5001.

Redis: Cola de mensajes para recolectar votos. Se ejecuta en el puerto 6379.

Worker (Java/.NET): Aplicación que consume votos de Redis y los almacena en la base de datos PostgreSQL.

PostgreSQL: Base de datos respaldada por un volumen Docker.