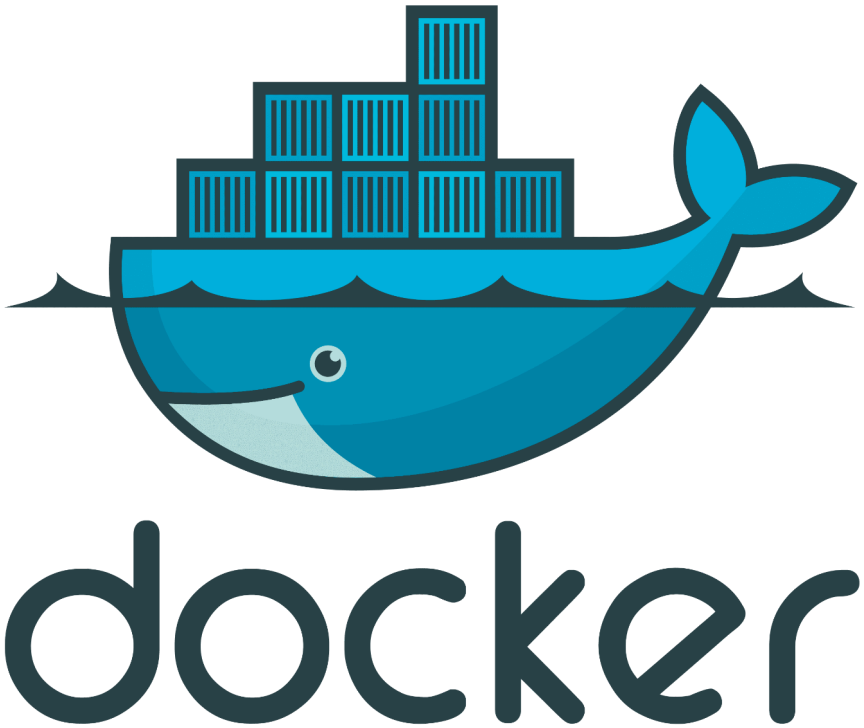
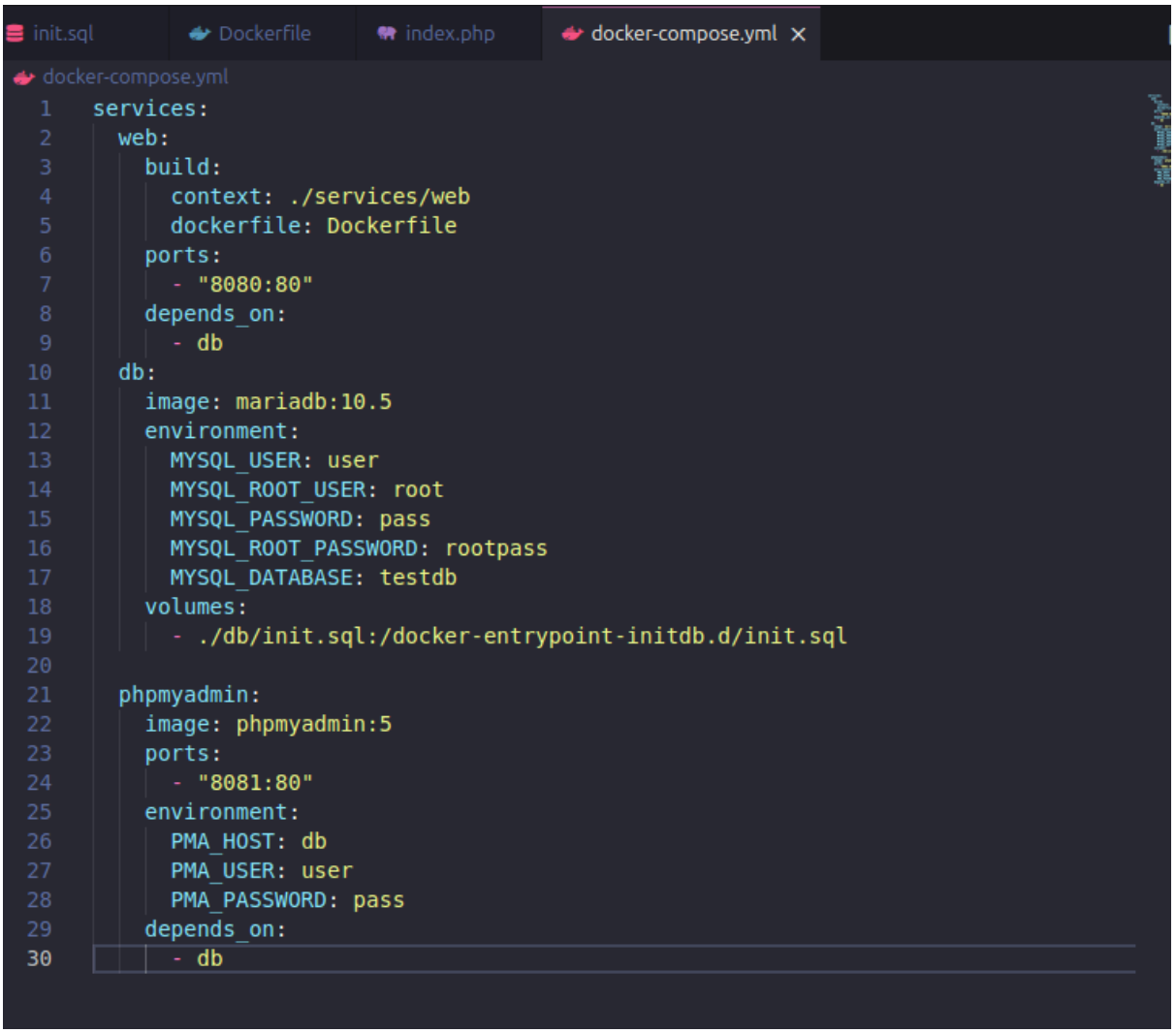
Práctica Docker

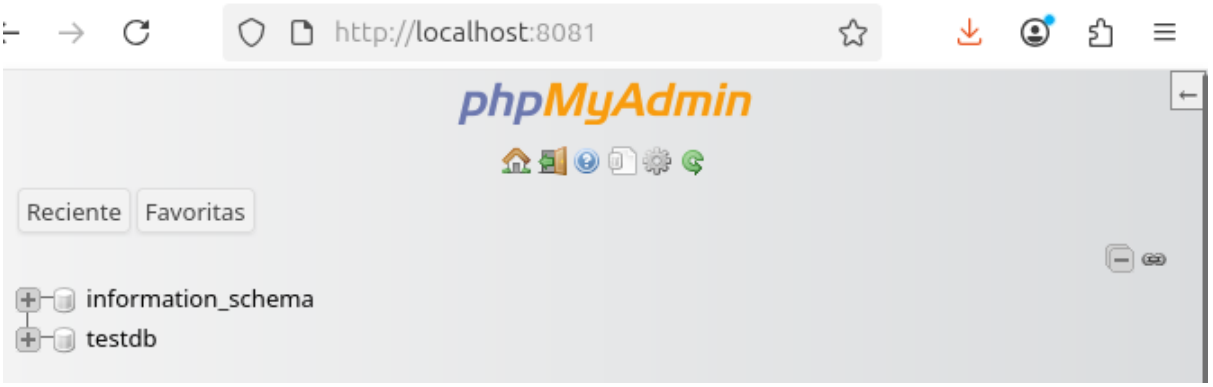


Manuel Morente Díaz

Añadimos la interfaz a la BD

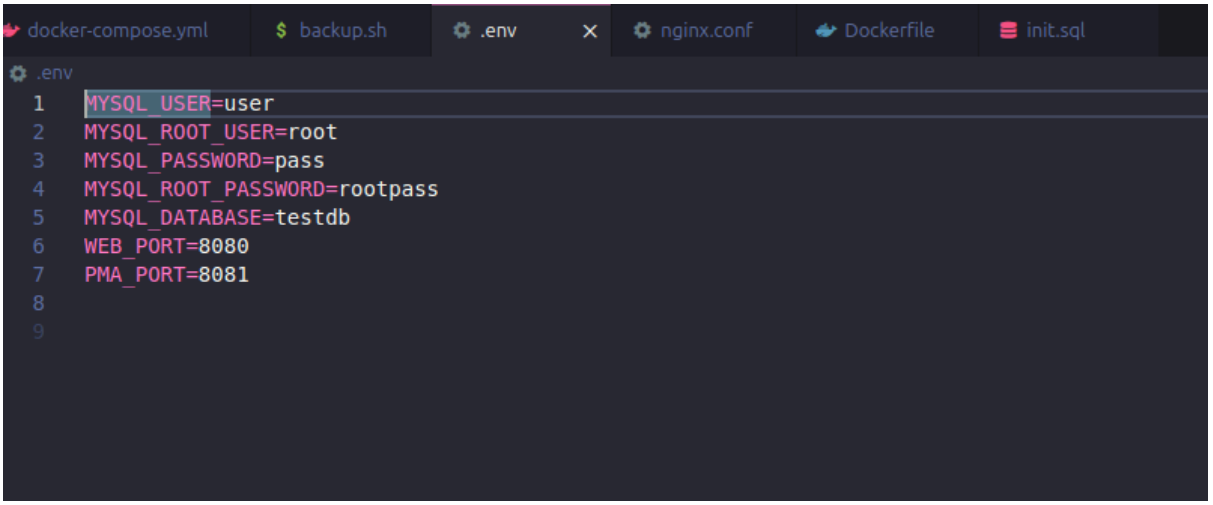
Se añade phpmyadmin en el docker-compose.yml para acceder a la web en cuanto a su interfaz y poder gestionar la base de datos.





Añadimos variables de entorno en un .env

Creo el archivo .env que se encuentra en la carpeta raíz, en este se encuentran claves y entornos ya usados. (Es tan solo un .env, sin nombre ya que de lo contrario no lo reconoce como tal)



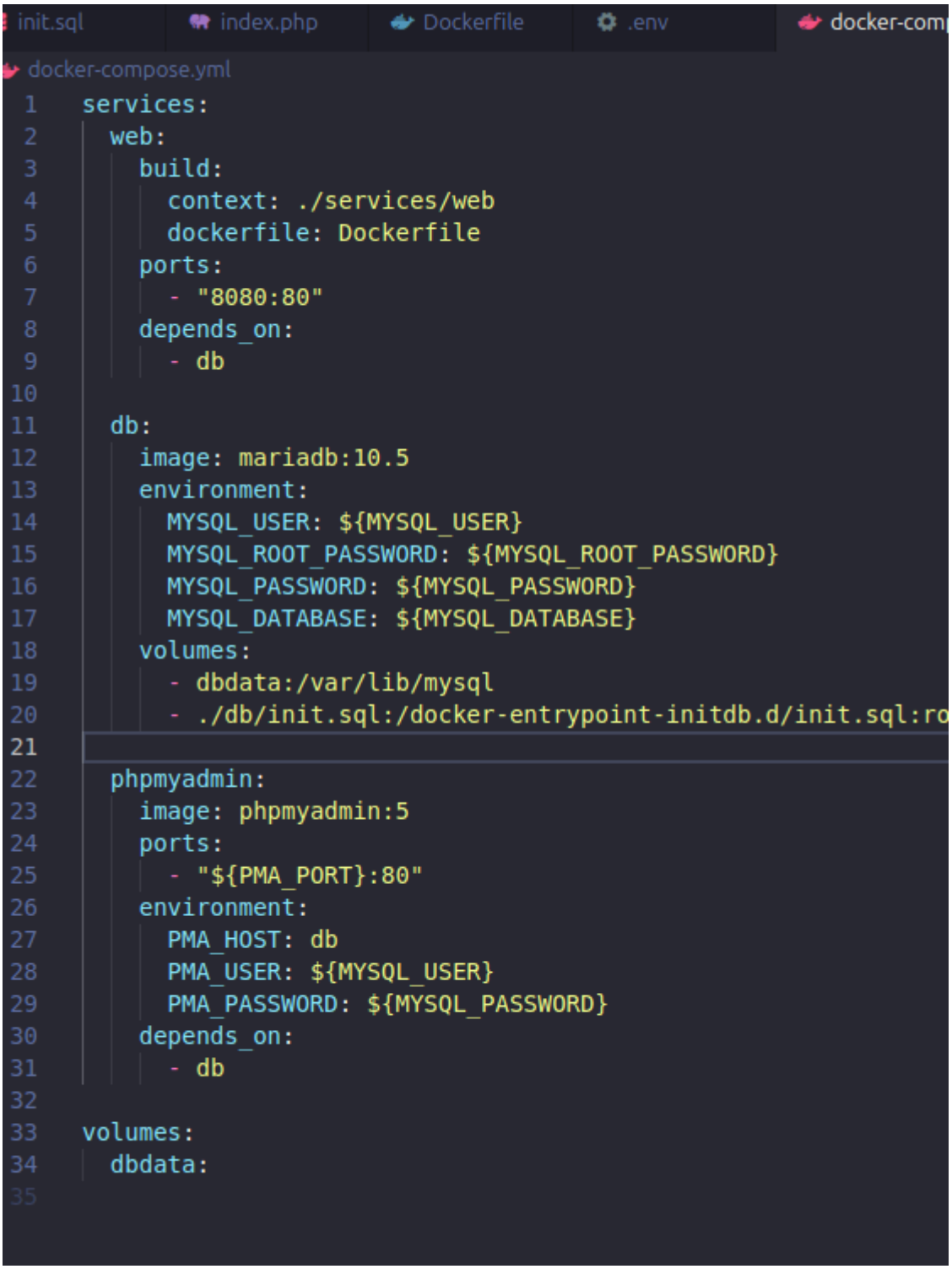
Modifico el docker-compose.yml para usar las variables del .env.

```yaml
services:
  web:
    build:
      context: ./services/web
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    depends_on:
      - db

  db:
    image: mariadb:10.5
    environment:
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    volumes:
      - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro

  phpmyadmin:
    image: phpmyadmin:5
    ports:
      - "${PMA_PORT}:80"
    environment:
      PMA_HOST: db
      PMA_USER: ${MYSQL_USER}
      PMA_PASSWORD: ${MYSQL_PASSWORD}
    depends_on:
      - db
```

Añadimos persistencia al contenedor

Dentro de db, añado un nuevo volumen, el cual es dbdata:/var/lib/mysql, indicando la ruta de guardado.

Debajo de todo el archivo, al final de este en volumes, se llama el volumen creado, dígase dbdata

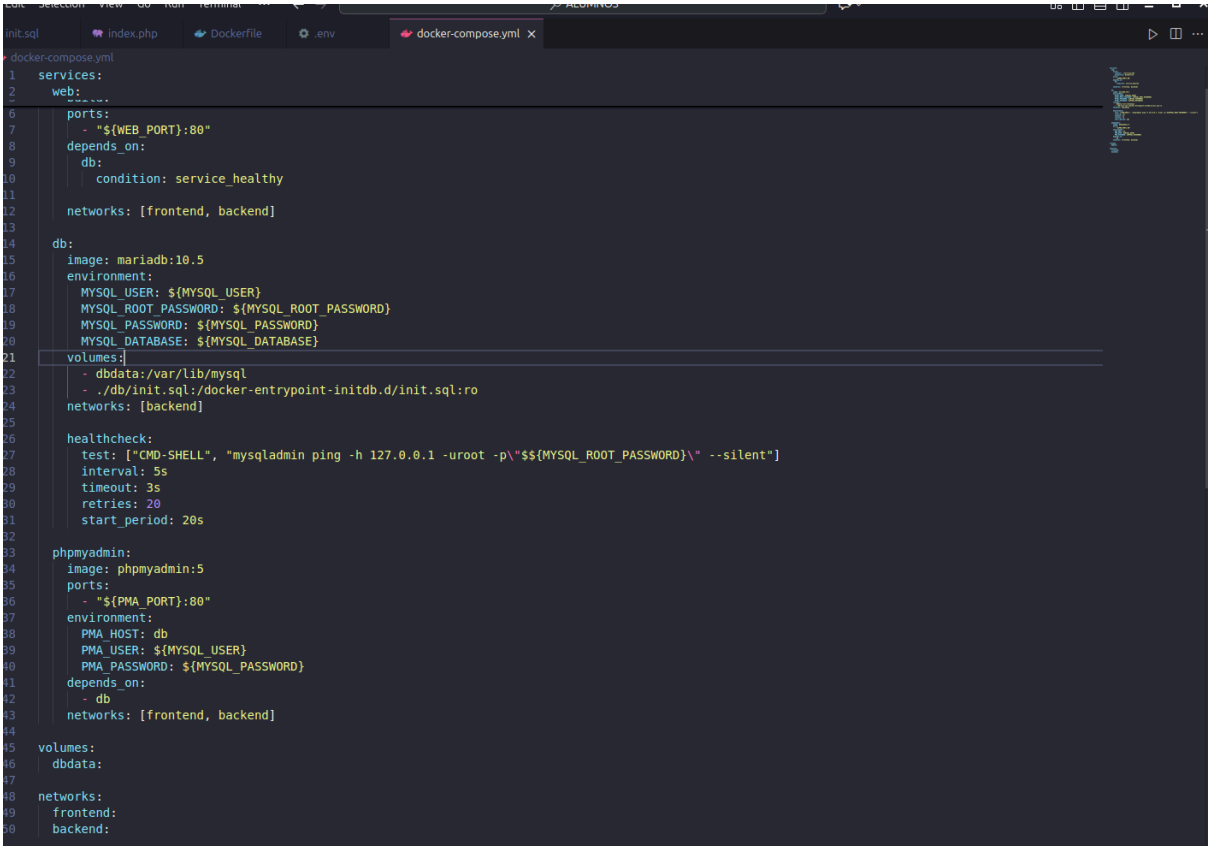


Distinguimos entre servicios

Creo en network las redes backend y frontend, db en este caso está solo en el backend y phpmyadmin requiere de ambos, e indico ambas redes al final de todo.

Check de la base de datos

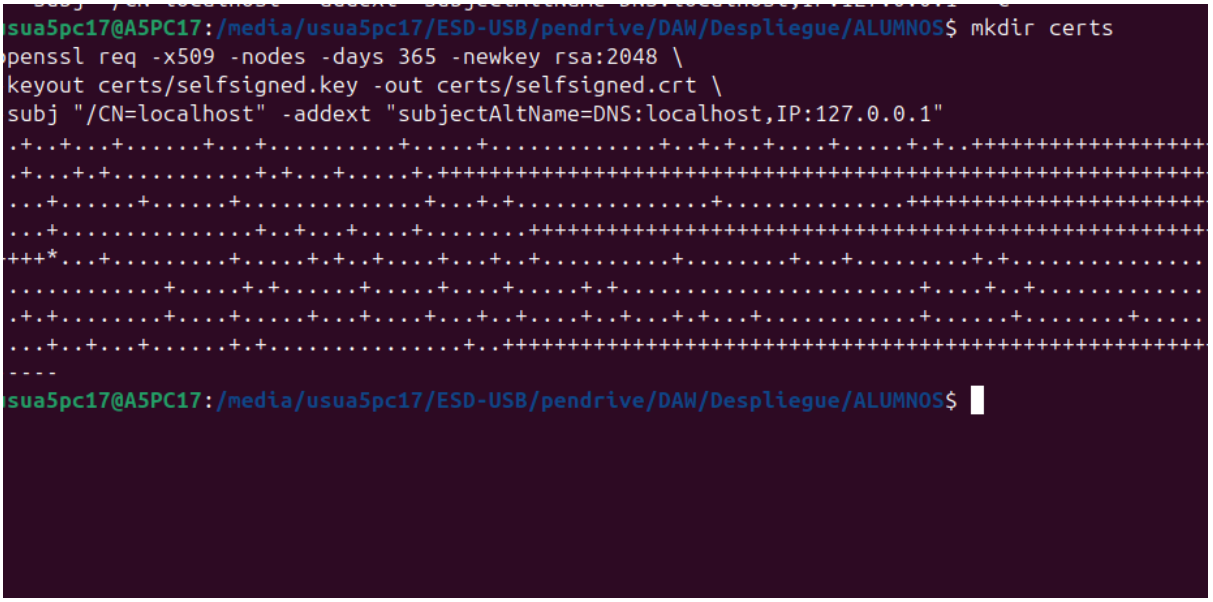
Con healthcheck, dentro de la db, ejecutamos mysqladmin ping, indicando este mismo. En el caso de la web, se define una condición en el que service_healthy esté disponible antes de arrancar la db.



```yaml
services:
  web:
    ports:
      - "${WEB_PORT}:80"
    depends_on:
      db:
        condition: service_healthy

    networks: [frontend, backend]

  db:
    image: mariadb:10.5
    environment:
      MYSQL_USER: ${MYSQL_USER}
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
      MYSQL_DATABASE: ${MYSQL_DATABASE}
    volumes:
      - dbdata:/var/lib/mysql
      - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
    networks: [backend]

    healthcheck:
      test: ["CMD-SHELL", "mysqladmin ping -h 127.0.0.1 -uroot -p\"${MYSQL_ROOT_PASSWORD}\" --silent"]
      interval: 5s
      timeout: 3s
      retries: 20
      start_period: 20s

  phpmyadmin:
    image: phpmyadmin:5
    ports:
      - "${PMA_PORT}:80"
    environment:
      PMA_HOST: db
      PMA_USER: ${MYSQL_USER}
      PMA_PASSWORD: ${MYSQL_PASSWORD}
    depends_on:
      - db
    networks: [frontend, backend]

volumes:
  dbdata:

networks:
  frontend:
  backend:
```

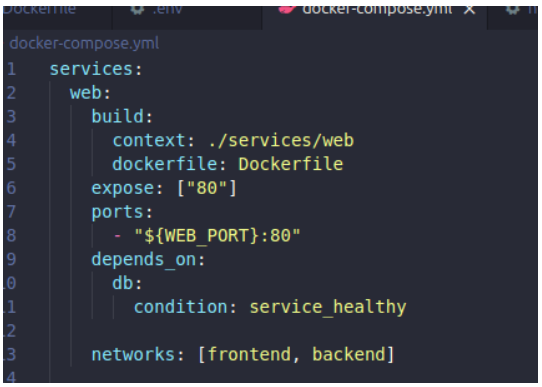
Certificado autofirmado

Con este comando genero el certificado, creando los archivos dentro de la carpeta certs, es decir, selfsigned.key y selfdigned.crt.



```
sua5pc17@A5PC17:/media/usua5pc17/ESD-USB/pendrive/DAW/Despliegue/ALUMNOS$ mkdir certs
penssl req -x509 -nodes -days 365 -newkey rsa:2048 \
keyout certs/selfsigned.key -out certs/selfsigned.crt \
subj "/CN=localhost" -addext "subjectAltName=DNS:localhost,IP:127.0.0.1"
.+..+...+......+...+..........+.....+.............+..+.+..+....+.....+.+..++++++++++++++++++
.+...+.+...........+.+...+.....+.++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
...+......+......+.............+...+.+.............+.............++++++++++++++++++++++++++
...+.............+..+...+....+........++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++*...+.........+.....+.+..+....+...+..+..........+........+...+.........+.+..............
............+.....+.+......+.....+....+.....+.+.....................+....+..+.............
.+.+........+....+.....+...+....+...+..+....+..+...+.+...+............+......+........+.....
...+..+...+......+.+.............+..++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
-----
sua5pc17@A5PC17:/media/usua5pc17/ESD-USB/pendrive/DAW/Despliegue/ALUMNOS$
```

En el apartado web dentro de docker-compose, cambio los puertos por expose, dejando el 80, en el proxy, defino el archivo y carpeta nginx, que sirve TLS con los certificados y hacer reverse proxy hacia la web.

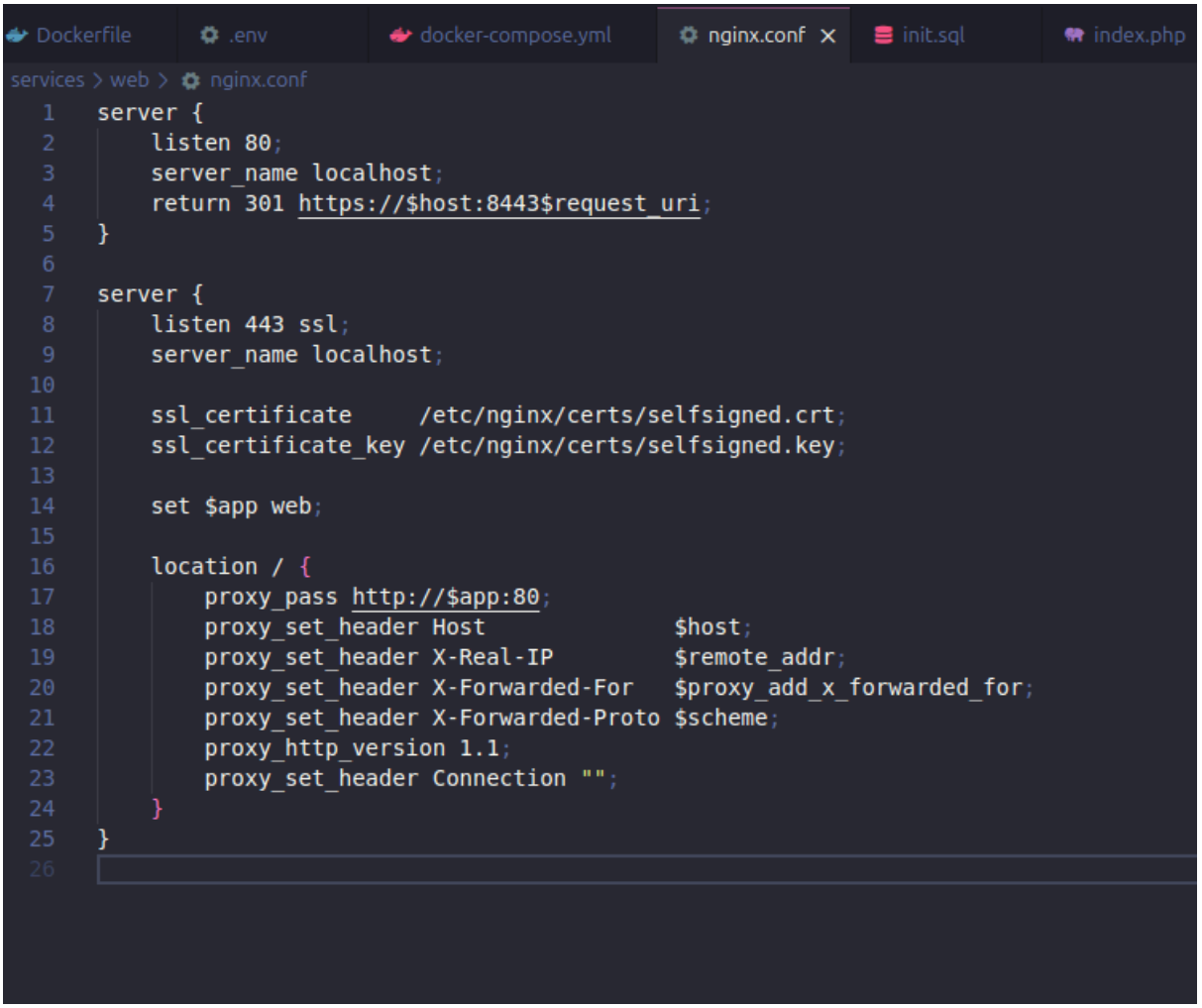


docker-compose.yml

```yaml
services:
  web:
    build:
      context: ./services/web
      dockerfile: Dockerfile
    expose: ["80"]
    ports:
      - "${WEB_PORT}:80"
    depends_on:
      db:
        condition: service_healthy

    networks: [frontend, backend]
```

```yaml
proxy:
  image: nginx:alpine
  ports:
    - "88:80"
    - "8443:443"
  volumes:
    - ./nginx.conf:/etc/nginx/conf.d:r
    - ./certs:/etc/nginx/certs:r
  depends_on:
    web:
      condition: service_started
```

Este es el nuevo archivo creado mediante el proxy anterior, en este lo que hace es redirigir HTTP y HTTPS y el proxy_pass a web, que es el nombre del servicio en Docker Compose



Dockerfile | .env | docker-compose.yml | nginx.conf | init.sql | index.php

services > web > nginx.conf

```
server {
    listen 80;
    server_name localhost;
    return 301 https://$host:8443$request_uri;
}

server {
    listen 443 ssl;
    server_name localhost;

    ssl_certificate     /etc/nginx/certs/selfsigned.crt;
    ssl_certificate_key /etc/nginx/certs/selfsigned.key;

    set $app web;

    location / {
        proxy_pass http://$app:80;
        proxy_set_header Host              $host;
        proxy_set_header X-Real-IP         $remote_addr;
        proxy_set_header X-Forwarded-For   $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }
}
```

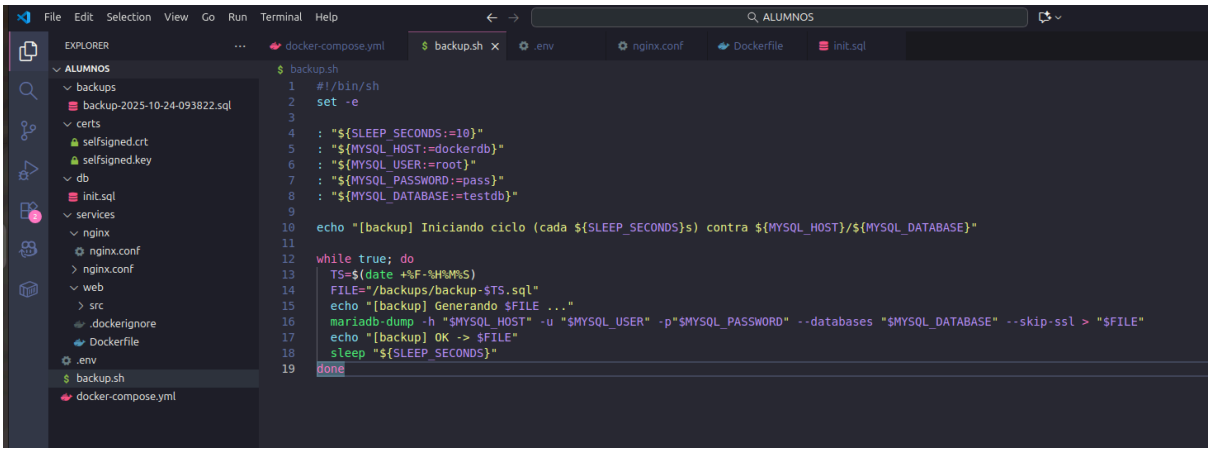
Balanceo de carga

Desde el archivo index.php, añado al final la linea de codigo que muestra el nombre del host o hostname del contenedor, luego con el comando correspondiente lo lanzo:
docker compose up -d --scale web=2

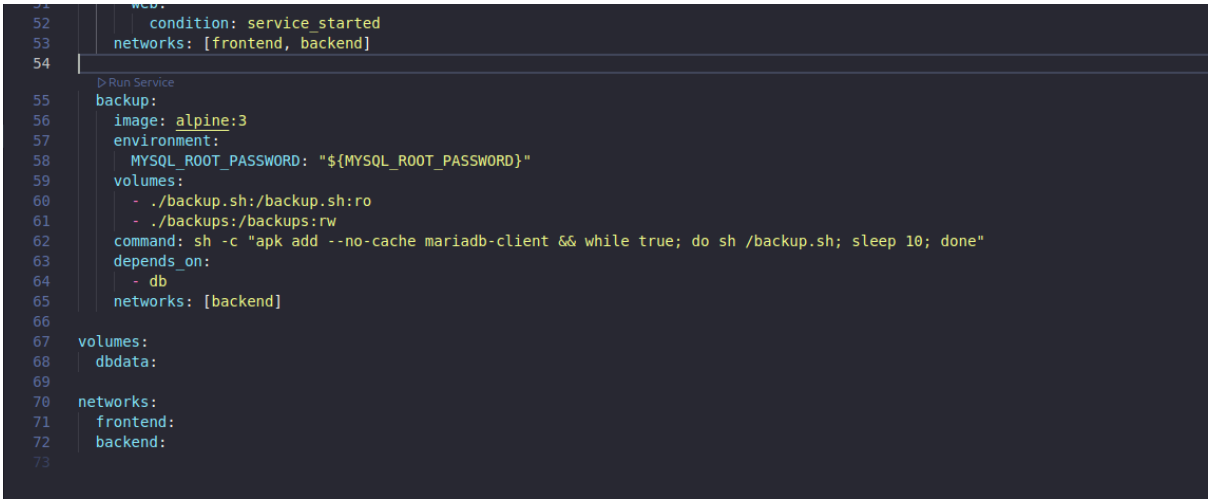
Cada que actualice la pagina, la web muestra distintos caracteres, indicando su correcto funcionamiento.

Backups

Se crea un nuevo archivo, backup.sh, aunque cambiando la cantidad de tiempo en la que cada vez hace el backup, de 60 a 10 segundos.
Esto es un script que cada que pase ese x tiempo, guarda este mismo backup en donde se le indique.



En docker compose, este fragmento monta la carpeta backups para guardar los anteriormente nombrados, y ejecuta el archivo backup.sh



Servicio de aplicaciones

Tras descargar sample.war, dentro de la ruta services/tomcat/webapps/sample.war, en el docker compose, añado el servicio tomcat, que expone el puerto 8080 en la red Docker, se monta el sample war en webapps para que este se despliegue.

```
▷ Run Service
tomcat:
  image: tomcat:9.0
  expose: ["8080"]
  volumes:
    - ./services/tomcat/webapps/sample.war:/usr/local/tomcat/webapps/sample.war
  networks: [backend]

volumes:
  dbdata:

networks:
  frontend:
  backend:
```

Proxy configurado para enrutar sample, a tomcat:8080/sample/ en archivo nginx.conf

```
}

server{
    listen 8080;
    server_name localhost;

    location /sample/ {
        proxy_pass http://tomcat:8080/sample/;
        proxy_set_header Host $host;
    }
}
```

Resultado final tras ejecución: