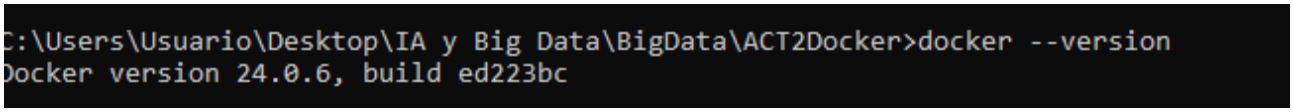
Práctica Docker

Ejercicio 1.

EJ1. (10 mins.) Repaso conceptos básicos: Responde a las siguientes cuestiones, adjunte una o varias capturas de pantalla para demostrar la validez de su respuesta.

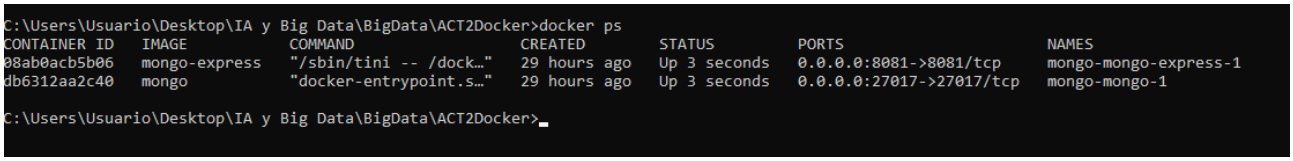
a) ¿Cómo sabemos cuál es la versión de Docker que tenemos instalada?

Para comprobar la version de Docker que tenemos instalada ejecutamos el comando **`docker --version`** en el terminal.



b) ¿Cuántos contenedores hay funcionando en su PC?

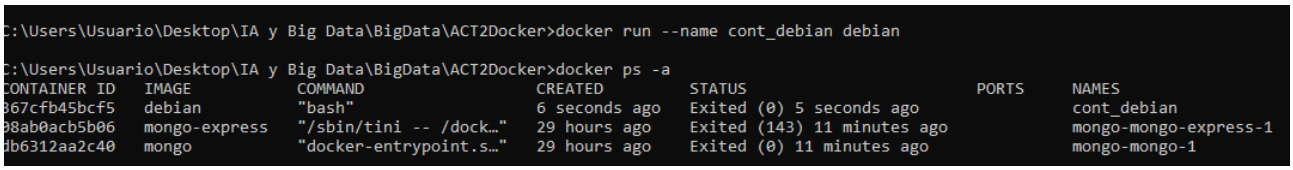
Con el comando **`docker ps`**.



c) Pon a funcionar un contenedor usando la imagen de debian

Para crear un contenedor con la imagen de debian se debe introducir en la terminal **`docker run --name [nombre contenedor] debian`**.

El contenedor debian, al no tener ningún proceso asignado se detiene automáticamente. Para que esto no ocurra sería necesario añadirle algún comando como **`sleep infinity`** al final.



d) Para el contenedor que has creado

Un contenedor se para desde el terminal con el comando **`docker stop [nombre contenedor]`**.

```
C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker ps
CONTAINER ID   IMAGE     COMMAND            CREATED          STATUS          PORTS     NAMES
32b196fb8875   debian    "sleep infinity"   27 seconds ago   Up 26 seconds             cont_debian

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker stop cont_debian
cont_debian

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>
```

e) Elimina el contenedor que has creado

Para eliminar el contenedor utilizamos el comando `docker rm [nombre contenedor]`.Es necesario también que el contenedor este detenido.

```
C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS                       PORTS     NAMES
32b196fb8875   debian          "sleep infinity"         2 minutes ago   Exited (137) 2 minutes ago             cont_debian
08ab0acb5b06   mongo-express   "/sbin/tini -- /dock…"   29 hours ago    Exited (143) 19 minutes ago            mongo-mongo-express-1
db6312aa2c40   mongo           "docker-entrypoint.s…"   29 hours ago    Exited (0) 19 minutes ago              mongo-mongo-1

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker rm cont_debian
cont_debian

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS                       PORTS     NAMES
08ab0acb5b06   mongo-express   "/sbin/tini -- /dock…"   29 hours ago    Exited (143) 19 minutes ago            mongo-mongo-express-1
db6312aa2c40   mongo           "docker-entrypoint.s…"   29 hours ago    Exited (0) 19 minutes ago              mongo-mongo-1

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>
```

Ejercicio 2.

EJ2. (20 mins.) Crear un contenedor debian, acceder a él desde dos puntos simultáneamente, luego ejecutar un comando sobre él, detenerlo y eliminarlo. Usar los comandos: create, start, attach, exec, stop y rm.

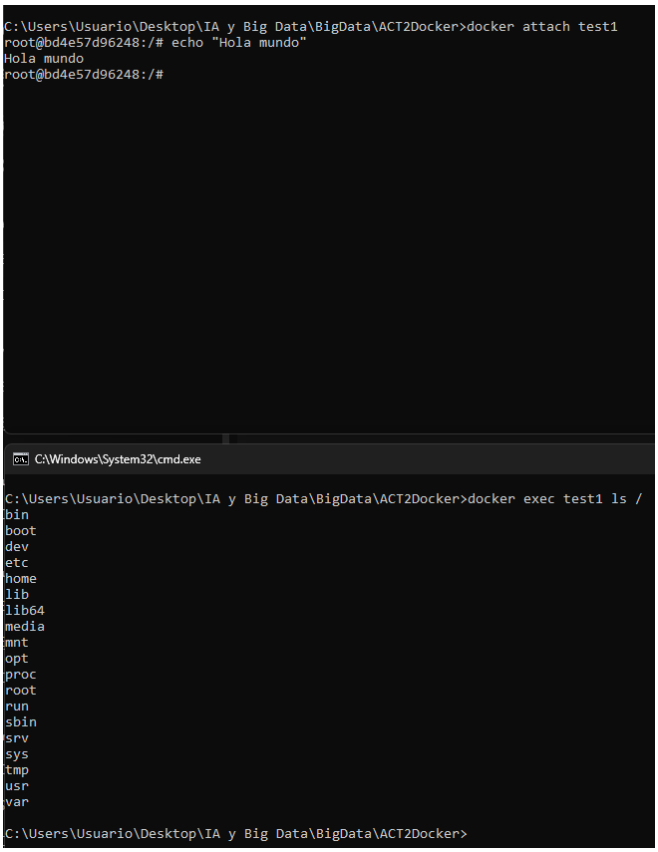
Nombre el contenedor como "test1" con la opción --name.

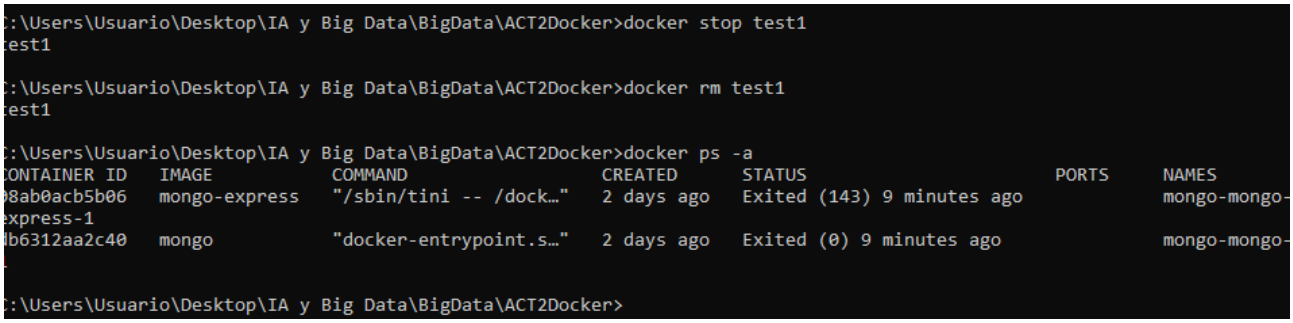
Publique esta imagen en su DockerHub.

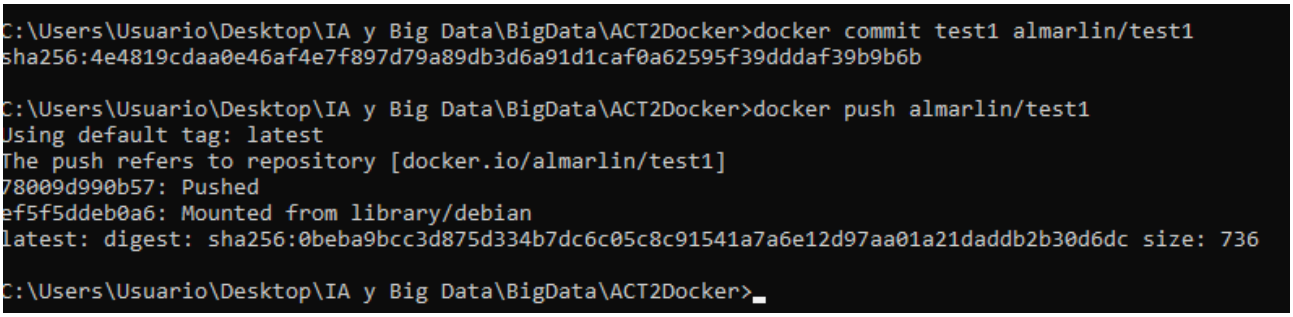
Adjunte una o varias capturas de pantalla para demostrar la validez de su respuesta. Y el enlace a su imagen en DockerHub.

```
C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker run --name test1 -it debian bash
root@bd4e57d96248:/# exit
exit

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker start test1
test1

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED          STATUS         PORTS     NAMES
bd4e57d96248   debian    "bash"    18 seconds ago   Up 1 second              test1

C:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>
```

```
:\Users\Usuario\Desktop\IA y Big Data\BigData\ACT2Docker>docker attach test1
oot@bd4e57d96248:/# echo "Hola mundo"
ola mundo
oot@bd4e57d96248:/#
```





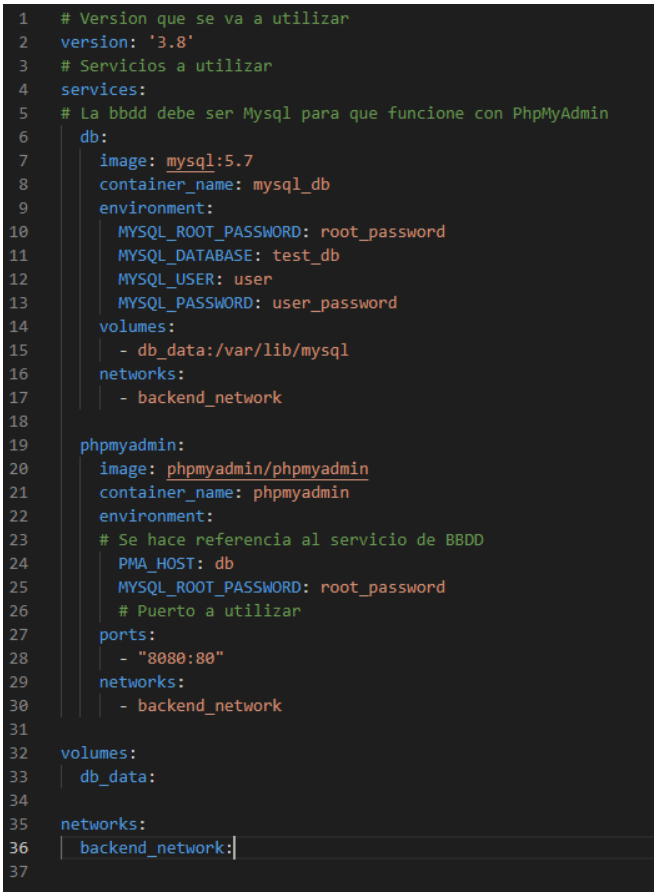


Docker Hub: https://hub.docker.com/repository/docker/almarlin/test1/general

Problema 1.

P1. (20 mins.) Crear un back-end de gestión de BBDD con phpMyAdmin y MySQL. Enlace de utilidad: https://hub.docker.com/r/phpmyadmin/phpmyadmin/

Para crear el contenedor con la capacidad de gestión de BBDD con phpMyAdmin es necesario configurar un **`docker-compose.yml`**. En este archivo se deben indicar los servicios que vayamos a configurar, las imágenes y el nombre de los contenedores. El archivo configurado es el siguiente:

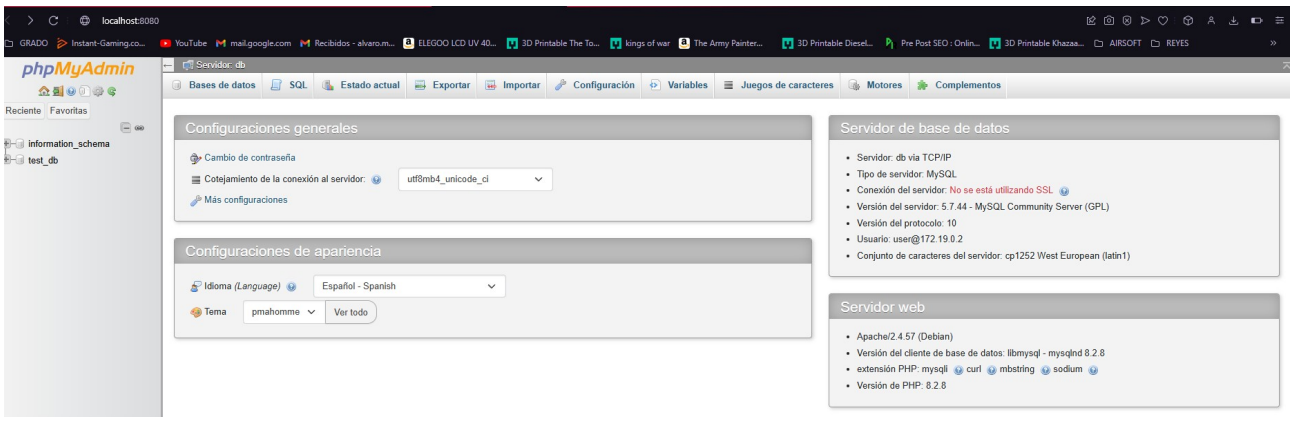


```yaml
# Version que se va a utilizar
version: '3.8'
# Servicios a utilizar
services:
# La bbdd debe ser Mysql para que funcione con PhpMyAdmin
  db:
    image: mysql:5.7
    container_name: mysql_db
    environment:
      MYSQL_ROOT_PASSWORD: root_password
      MYSQL_DATABASE: test_db
      MYSQL_USER: user
      MYSQL_PASSWORD: user_password
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - backend_network

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin
    environment:
    # Se hace referencia al servicio de BBDD
      PMA_HOST: db
      MYSQL_ROOT_PASSWORD: root_password
      # Puerto a utilizar
    ports:
      - "8080:80"
    networks:
      - backend_network

volumes:
  db_data:

networks:
  backend_network:

```

Una vez configurado el archivo es necesario ejecutar el comando **`docker-compose up -d`** en la ruta donde se encuentre este archivo para crear los contenedores ejecutarlos.

```
C:\Users\Usuario\vsworkspace\IAyBigData\BigData\ACT2Docker\Problema1>docker-compose up -d
[+] Running 31/20
 ✔phpmyadmin 18 layers [####################]      0B/0B      Pulled                                                  85.1s
 ✔db 11 layers [############]      0B/0B      Pulled                                                               101.5s

[+] Building 0.0s (0/0)                                                                                  docker:default
[+] Running 4/4
 ✔Network problema1_backend_network  Created                                                                       0.3s
 ✔Volume "problema1_db_data"         Created                                                                       0.0s
 ✔Container mysql_db                 Started                                                                       1.8s
 ✔Container phpmyadmin               Started                                                                       1.8s

C:\Users\Usuario\vsworkspace\IAyBigData\BigData\ACT2Docker\Problema1>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED          STATUS          PORTS                     NAMES
ef9e0bf26f01   phpmyadmin/phpmyadmin  "/docker-entrypoint.…"   36 seconds ago   Up 32 seconds   0.0.0.0:8080->80/tcp      phpmyadmin
c7aa0f230531   mysql:5.7              "docker-entrypoint.s…"   36 seconds ago   Up 32 seconds   3306/tcp, 33060/tcp       mysql_db
08ab0acb5b06   mongo-express          "/sbin/tini -- /dock…"   2 days ago       Up 2 minutes    0.0.0.0:8081->8081/tcp    mongo-mongo-express-1
db6312aa2c40   mongo                  "docker-entrypoint.s…"   2 days ago       Up 2 minutes    0.0.0.0:27017->27017/tcp  mongo-mongo-1
```

Por último, queda comprobar que funcione correctamente en el puerto configurado (en este caso 8080) e iniciar sesión con las credenciales introducidas en el **`docker-compose.yml`**.



Problema 2.

P2. (90 mins.) Crear las pilas (X)AMP y (X)EMP para despliegue de proyectos web usando:

- *Para XAMP: (X=cualquier OS), Apache Server, MySQL con phpMyAdmin y PHP.*
- *Para XEMP: (X=cualquier OS), Nginx, MySQL con phpMyAdmin y PHP*

Enlaces de utilidad: https://openwebinars.net/blog/infraestructura-lamp-con-docker-compose/

- *Documentación de la imagen nginx en: https://hub.docker.com/nginx/*
- *Configuración del servidor nginx: http://nginx.org/en/docs/beginners_guide.html#conf_structure*

XAMP.

Una pila XAMP es un conjunto de software para el desarrollo web donde:

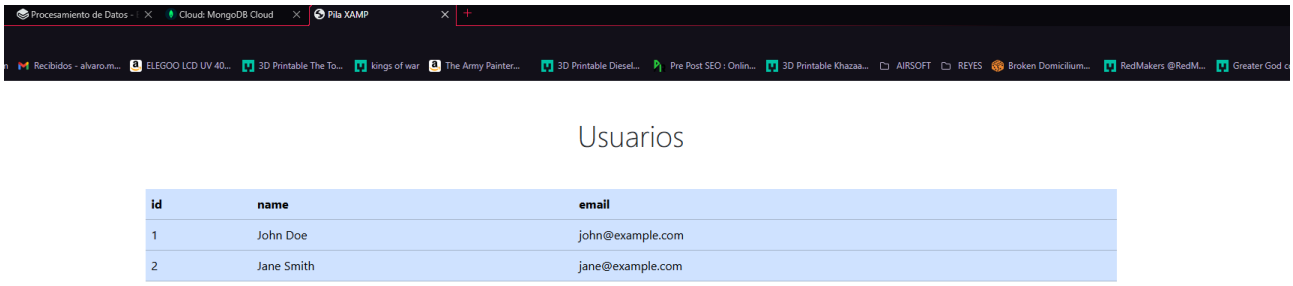
- X: Funciona en distintos sistemas operativos.
- A: Apache, un servidor web para mostrar las páginas creadas.
- M: MySQL, para gestionar la base de datos de la aplicación.
- P: PHP como lenguaje de programación por parte del servidor.

Se ha configurado el **`docker-compose.yml`** de la siguiente forma:

Problema2 > XAMP > docker-compose.yml

```yaml
version: "3.1"
services:
    www:
        image: php:8.0-apache
        build: .
        ports:
            - "80:80"
        volumes:
            - ./www:/var/www/html
        links:
            - db
        networks:
            - default
    db:
        image: mysql:8.0
        ports:
            - "3307:3307"
        command: --default-authentication-plugin=mysql_native_password
        environment:
            MYSQL_DATABASE: dbname
            MYSQL_USER: user
            MYSQL_ROOT_USER: root
            MYSQL_PASSWORD: test
            MYSQL_ROOT_PASSWORD: root
        volumes:
            - ./dump:/docker-entrypoint-initdb.d
            - ./conf:/etc/mysql/conf.d
            - persistent:/var/lib/mysql
        networks:
            - default
    phpmyadmin:
        image: phpmyadmin/phpmyadmin
        links:
            - db:db
        ports:
            - 8000:80
        environment:
            MYSQL_USER: user
            MYSQL_PASSWORD: test
            MYSQL_ROOT_PASSWORD: test
volumes:
    persistent:
```

Cuando ponemos a funcionar los contenedores el resultado es el siguiente:



En el servidor web (Apache) nos aparece el resultado de la consulta a la base de datos de MySQL en un PHP.

XEMP.

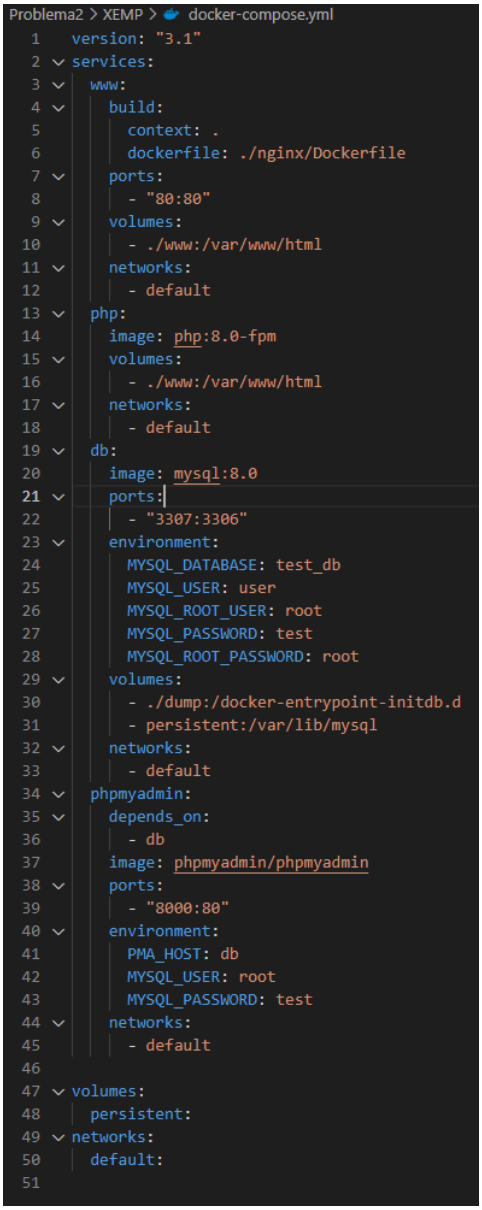
Una pila XEMP es aquella que utiliza una estructura similar a XAMP pero donde nginx reemplaza a Apache. En este caso se ha configurado de la siguiente forma.

- Nginx (www). Se construye a partir de un Dockerfile ubicado en ./nginx/Dockerfile, se mapea al puerto 80.80 y se guardan sus archivos (html, php, etc.) en ./www..
- PHP (php). Se monta igualmente con acceso al directorio ./www para procesar los archivos. También se conecta en la misma red que Nginx y Mysql.
- MySQL (db): Mapea al puerto 3307 del ordenador al 3306 del contenedor y se genera la base de datos con init.sql.
- PhpMyAdmin. Utiliza la imagen de phpmyadmin para tener la interfaz fácil de usar.

Para que XEMP funcione es necesario tener varios puntos en cuenta.

- Para la imagen de Nginx es necesario un **`Dockerfile`** que cree un **`nginx.conf`**.
- Es necesario un archivo **`fastcgi-php.conf`** que defina las configuraciones para PHP.

Este es el `docker-compose.yml`:



Problema 3.

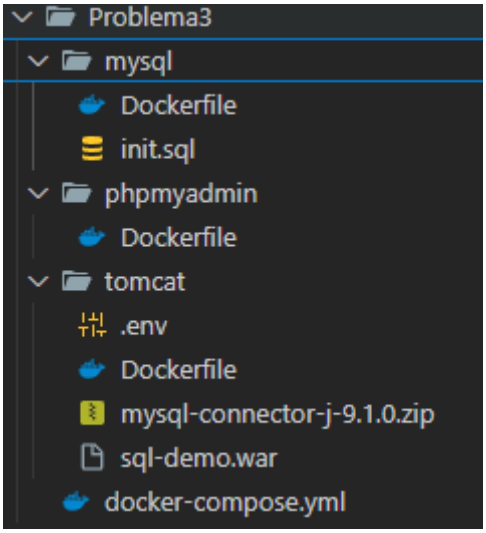
P3. (100 mins.) Crear un servidor Tomcat con MySQL-phpMyAdmin para despliegue de aplicaciones web con Java. Se despliega un .war (necesitaréis uno de prueba).

Enlace de utilidad:

- *https://www.jetbrains.com/help/idea/docker-tutorial-tomcat-debug.html#attach_debugger*
- *https://github.com/Seetha1231/docker-httpd-tomcat-mysql/tree/master/tomcat*

Nota: Debe poderse demostrar el correcto funcionamiento de PHP/Java y MySQL interconectados, es decir, que se debe hacer al menos una consulta simple a la base de datos desde la aplicación, tal y como se describe, en los enlaces de utilidad.

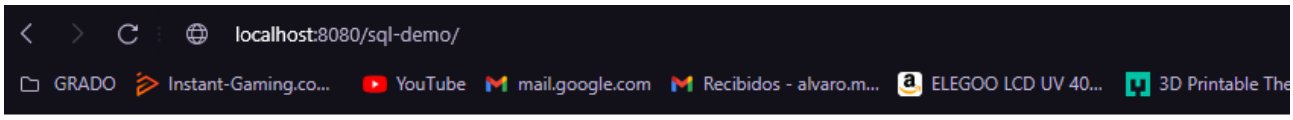
Siguiendo con el **`Github`** proporcionado y utilizando el **`.war`** proporcionado ahí, se ha montado el servidor de la siguiente forma:



En los **`Dockerfile`** de cada carpeta se incluye la imagen de cada servicio a levantar por el **`docker-compose.yml`**. También se ha incluido el conector de mysql y un .env para poder acceder a la base de datos y a la aplicación.

Una vez puesto en funcionamiento los contenedores el resultado se muestra en http://localhost:8080/sql-demo/ .

Resultado final:



Connection status

Name : abc

Name : xyz

Name : pqr
Successfully connected to MySQL server using TCP/IP...