**DOCUMENTACION**

**Breve resumen que argumente cómo se han conseguido los diferentes objetivos/logros, con capturas del código y explicación.**

**DAW**

Instalamos el Docker-Desktop:

[Descargar docker](https://www.docker.com/products/docker-desktop/)  
  
Una vez instalado accedemos e iniciamos sesión, nos saldrán los contenedores vacíos.A screenshot of a computer

Description automatically generated

Vamos a crear un contenedor con MySQL:

Vamos al PowerShell:

Ejecutamos: **docker pull mysql**

Esto nos descarga la imagen del mysql

Ahora ejecutamos: **docker run --name mysql -e MYSQL\_ROOT\_PASSWORD=123456 -p 3307:3306 - d mysql**

Esto crea el contenedor

A continuacion con este comando accedemos a MySQL desde la terminal:

**docker exec -it mysql mysql -u root –p123456**  
A screenshot of a computer program

Description automatically generated

Para acceder de manera grafica vamos al laragon > Base de datos > Nueva

Y rellenamos de esta forma recuerda que la contraseña es: 123456 por el puerto 3307

A screenshot of a computer

Description automatically generated

Le damos abrir y listo: A screenshot of a computer

Description automatically generated

Comandos para Docker:

docker start mysql // arranca el contenedor

Docker stop mysql // para el contenedor

Docker ps // muestra los contenedores arrancados

Docker ps –a // muestra los todos los contenedores

Así se crea con docker composer

Primero creamos esta estructura:

A computer screen shot of a program

Description automatically generated

sDentro de la raiz creamos el .yml llamado docker-compose.yml

A screenshot of a computer

Description automatically generated

El .env:

A screenshot of a computer

Description automatically generated

En el directorio apache creamos este archivo 000-default.confA screen shot of a computer

Description automatically generated

Con esto vamos al powerShell al directorio donde tienes el docker-compose y levantamos el servicio con: docker-compose up –d

A screenshot of a computer

Description automatically generated

Ahora comprobamos.

Con el nombre de dominio.

A screenshot of a computer

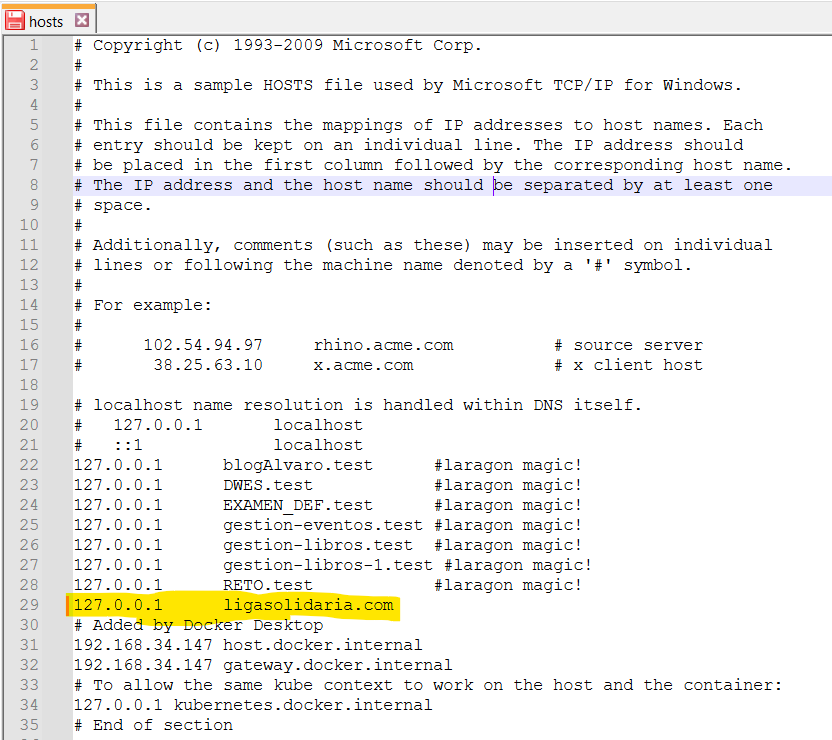
AI-generated content may be incorrect.

Y en la base de datos que crean las tablas.

A screenshot of a computer

AI-generated content may be incorrect.

Definimos host virtual con DNS.

****

Aplicamos SSL en el docker, con este comando creando en la raiz un repositotio ssl y nos movemos a ese repositorio ‘ssl’:

docker run --rm -v ${PWD}:/certs -w /certs alpine/openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout server.key -out server.crt -subj "/C=US/ST=State/L=City/O=Organization/OU=Department/CN=ligasolidaria.com"

Esto nos descargara la clave pública y privada, modificamos el archivo de configuración de apache.

<VirtualHost \*:80>

ServerName ligasolidaria.com

Redirect permanent / <https://ligasolidaria.com/>

</VirtualHost>

<VirtualHost \*:443>

ServerName ligasolidaria.com

DocumentRoot /var/www/html

SSLEngine on

SSLCertificateFile /etc/apache2/ssl/server.crt

SSLCertificateKeyFile /etc/apache2/ssl/server.key

<Directory /var/www/html>

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

Ahora también el .yml.

version: '3.7'

services:

web:

image: php:apache

container\_name: apache\_server

restart: always

ports:

- "80:80"

- "443:443"

volumes:

- ./www:/var/www/html

- ./apache/000-default.conf:/etc/apache2/sites-available/000-default.conf

- ./ssl:/etc/apache2/ssl

environment:

- APACHE\_DOCUMENT\_ROOT=/var/www/html

depends\_on:

- db

command: >

bash -c "a2enmod ssl &&

a2ensite 000-default &&

apache2-foreground"

db:

image: mariadb

container\_name: mariadb\_server

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: root\_password

MYSQL\_DATABASE: tu\_app

MYSQL\_USER: usuario

MYSQL\_PASSWORD: contraseña

volumes:

- ./mysql/init.sql:/docker-entrypoint-initdb.d/init.sql

ports:

- "3308:3306"

Archivo de certificados dentro del proyecto.

A screenshot of a computer

AI-generated content may be incorrect.

Volvemos a construir nuestra web:

docker-compose down

docker-compose up -d

Contenedores utilizados para la aplicación, en Docker Desktop.

A screenshot of a computer

AI-generated content may be incorrect.

Para el desarrollo y control de versiones de la aplicación hemos usado un repositorio GitHub.

A screenshot of a computer

AI-generated content may be incorrect.

**DIW**

Para nuestra aplicación, hemos empleado imágenes sin licencia.

El formato elegido para las imágenes es WebP, ya que pesa poco y tiene mayor calidad que otros formatos.

Se importan imágenes a la hora de añadir retos a la web.

El diseño de la web es responsive.

**DWEC**

DOM:

Filtro para ver la lista de centros:Texto

El contenido generado por IA puede ser incorrecto.   
  
Este script usa jQuery para eliminar un reto en una aplicación web:

* **Funcionalidad:**
  + Se activa al hacer clic en un elemento con la clase .delete-reto.
  + Obtiene el id del reto que se desea eliminar mediante el atributo data-id.
  + Muestra un cuadro de confirmación para asegurarse de que el usuario desea eliminar el reto.
  + Si el usuario confirma, realiza una petición AJAX con el método DELETE a una URL específica (/retos/${retoId}).
  + Si la eliminación es exitosa, el reto se elimina del DOM con una animación.

Texto

El contenido generado por IA puede ser incorrecto.

Carga dinámica de retos y actualización de filtros, usa AJAX para:

* Cargar los retos con base en el torneo y el centro seleccionados.
* Actualizar el <select> de centros cuando se cambia el torneo.
* Genera el HTML para los retos dinámicamente y lo agrega a un contenedor (#retos-container).

Texto

El contenido generado por IA puede ser incorrecto.

**DWES**

A white and black text on a white background

Description automatically generated

A screen shot of a computer

Description automatically generated

A white rectangular box with black text

Description automatically generated

A white square with black text

Description automatically generated