Este archivo php lo que nos permitirá es conectarnos a la base de datos para poder obtener la información de esta y poder mostrar los datos por pantalla.

index.php

```php
<?php
$mysqli = @new mysqli('db', 'root', 'rootpass', 'demo');
if ($mysqli->connect_error) {
    die("Error de conexión: " . $mysqli->connect_error);
}
$result = $mysqli->query("SELECT * FROM users");
while ($row = $result->fetch_assoc()) {
    echo $row['id'] . " - " . $row['nombre'] . "<br>";
}
?>

```

Aqui con el archivo yml tendremos dos servicios, el de la base de datos y el de apache, luego usaremos una red interna para que se puedan ver entre ellos, aparte tambien crearemos un volumen para los datos de la base de datos persistan. Definiremos los perfiles dev y prod y por ultimo añadiremos el healthcheck en la base de datos con un depends_on con un service_healthy.

docker-compose.yml

```yaml
version: '3.9'

services:
  db:
    image: mysql:8.0
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: demo
    volumes:
      - db_data:/var/lib/mysql
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    networks:
      - internal
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 5s
      retries: 5
    profiles: ["dev"]

  web:
    image: php:8.2-apache
    volumes:
      - ./src:/var/www/html
    networks:
      - internal
    depends_on:
      db:
        condition: service_healthy
    profiles: ["dev", "prod"]

volumes:
  db_data:

networks:
  internal:
```

Aqui creamos el el archivo sql para poder crear la base de datos al ejecutar los comandos pertinentes.

init.sql

```sql
CREATE DATABASE IF NOT EXISTS demo;
USE demo;

CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL
);

INSERT INTO users (nombre) VALUES ('Mon3tr'), ('Laevatain'), ('Kakekuri');

```

--profile dev: levanta db y web (porque ambos tienen el profile dev).

--profile prod: levanta solo web (la BD no tiene profile prod).

-d: modo detached (en segundo plano).

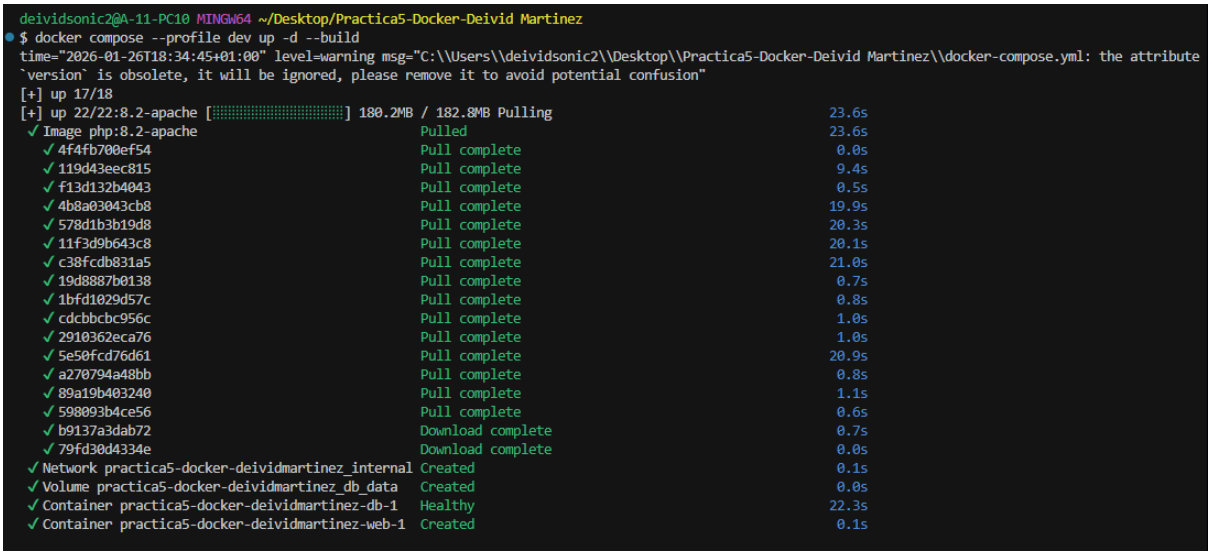
--build: fuerza reconstrucción si cambias algo (aunque aquí usamos imágenes oficiales, sigue siendo válido).

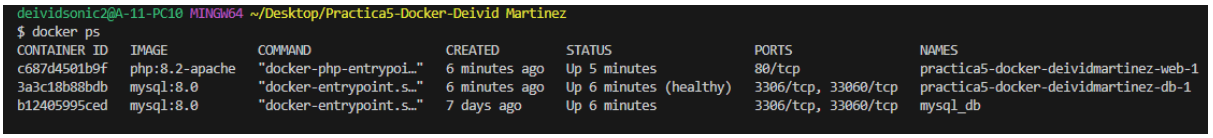
docker ps: para capturas:

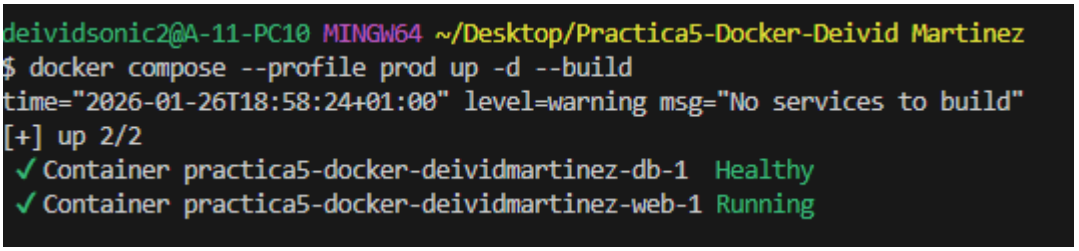
En dev: deben salir 2 contenedores (db y web).

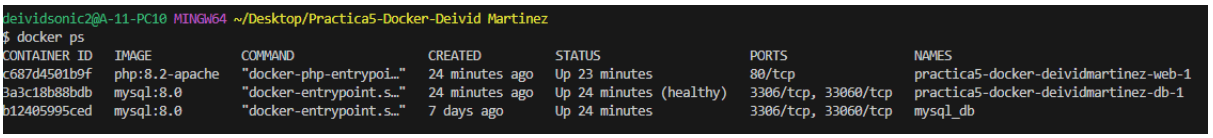
En prod: solo web (y la web debe fallar al conectar a la BD).

docker compose down: para parar y limpiar contenedores y red (el volumen persiste).









```
deividsonic2@A-11-PC10 MINGW64 ~/Desktop/Practica5-Docker-Deivid Martinez
$ docker compose down
[+] down 2/2
 ✓ Container practica5-docker-deividmartinez-db-1   Removed
 ! Network practica5-docker-deividmartinez_internal Resource is still in use
```

```
deividsonic2@A-11-PC10 MINGW64 ~/Desktop/Practica5-Docker-Deivid Martinez
$ docker compose --profile prod up -d --build
service "web" depends on undefined service "db": invalid compose project
```

Explicación del docker-compose.yml

El archivo docker-compose.yml define dos servicios: db, que ejecuta MySQL y crea la base de datos con datos iniciales, y web, que ejecuta PHP con Apache para mostrar la información en el navegador. El volumen db_data sirve para que los datos de MySQL no se pierdan aunque el contenedor se elimine. La red interna permite que los servicios se comuniquen entre sí usando sus nombres. El healthcheck comprueba si MySQL está listo antes de permitir que la aplicación web arranque en el perfil de desarrollo. Finalmente, los profiles controlan qué servicios se levantan: en dev se inician db y web para trabajar normalmente, mientras que en prod solo se inicia web, provocando intencionadamente un error de conexión al no existir la base de datos.

Conclusion Personal

Una práctica como esta enseña mucho en poco tiempo porque combina varios conceptos del mundo profesional. En esencia, te permite entender cómo se organizan varios servicios con Docker Compose. También aprendes a separar entornos mediante profiles. Aprendes a garantizar que un servicio no arranque hasta que otro esté listo usando healthchecks. Aprendes cómo una aplicación web se comunica con una base de datos dentro de una red interna. Además, trabajas con volúmenes para mantener datos persistentes.