**Instrucciones parciales 3: Orquestación de Servicios con Docker Compose**

**Objetivo**

Levantar 8 servicios diferentes usando Docker Compose, algunos conectados entre sí, simulando un entorno empresarial con servicios web, base de datos, email, administración y proxy inverso.

**🔧 Servicios a implementar**

1. **MySQL** – Base de datos relacional.
2. **phpMyAdmin** – Interfaz web para administrar MySQL.
3. **WordPress** – CMS que usará MySQL como backend.
4. **Node.js API** – API desarrollada por ti, que también se conecta a MySQL.
5. **Mailhog** – Servicio para pruebas de envío de correo
6. **Nginx** – Reverse proxy para WordPress y la API.
7. **MongoDB** – Base de datos NoSQL independiente.
8. **AdminMongo** – Interfaz web para MongoDB.

**🗂️ Estructura de archivos sugerida**

docker-compose-taller/

├── backend/

│ ├── Dockerfile

│ └── index.js

├── nginx/

│ └── default.conf

└── docker-compose.yml

**📋 Paso a paso**

**1. Crea la estructura de carpetas mencionada.**

**2. Dentro de docker-compose.yml define los servicios mencionados:**

* Usa mysql:8 y define usuarios, contraseña y base de datos.
* Conecta phpMyAdmin a MySQL
* WordPress debe usar los datos del servicio de MySQL.
* El backend en Node.js debe construirse desde un Dockerfile.
* Mailhog usa el puerto 8025 por defecto.
* Nginx debe enrutar:
  + / hacia WordPress
  + /api hacia tu backend
* MongoDB debe exponerse en el puerto 27017.
* AdminMongo debe conectarse a MongoDB y exponerse en el puerto 8082.

**3. Crea el archivo Dockerfile para tu API en ./backend/.**

**4. Escribe un index.js simple para tu API que responda a GET /.**

**5. Configura default.conf en Nginx para enrutar hacia WordPress y el backend.**

**6. Usa volúmenes para persistencia de MySQL y MongoDB.**

**7. Declara todas las redes necesarias (una red compartida es suficiente para este taller).**

**8. Usa docker-compose up -d --build para levantar todo.**

**9. Verifica el funcionamiento accediendo a:**

* WordPress: http://localhost:8080
* phpMyAdmin: http://localhost:8081
* Backend (vía Nginx): http://localhost/api
* Mailhog: http://localhost:8025
* AdminMongo: http://localhost:8082

**🧪 Extras**

* Agrega funcionalidad de envío de correos desde tu API a Mailhog.
* Agrega operaciones CRUD desde el backend hacia MySQL.
* Usa MongoDB para almacenar logs o registros.