PARCIAL 3 – SERVICIOS CON DOCKER COMPOSE

PRESENTADO POR:

CRISTIAN CAMILO VALLEJOS BASTIDAS

JOHAN FERNANDO BURBANO CALPA

INSTITUTO TECNOLOGICO DEL PUTUMAYO

FACULTAD DE INGENIERIA Y CIENCIAS BASICAS

INGENIERIA DE SISTEMAS

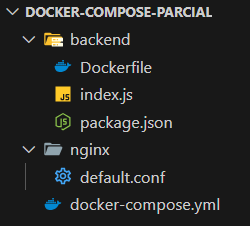
MOCOA - PUTUMAYO

2025

Servicios Con Docker Compose

**Objetivo:** Implementar un entorno empresarial compuesto por 8 servicios orquestados con Docker Compose, que integren funcionalidades de base de datos, administración, backend, frontend, pruebas de correo y proxy inverso.

Estructura de Proyecto:

****

**Crear docker-compose.yml**

Este archivo le dice a Docker:

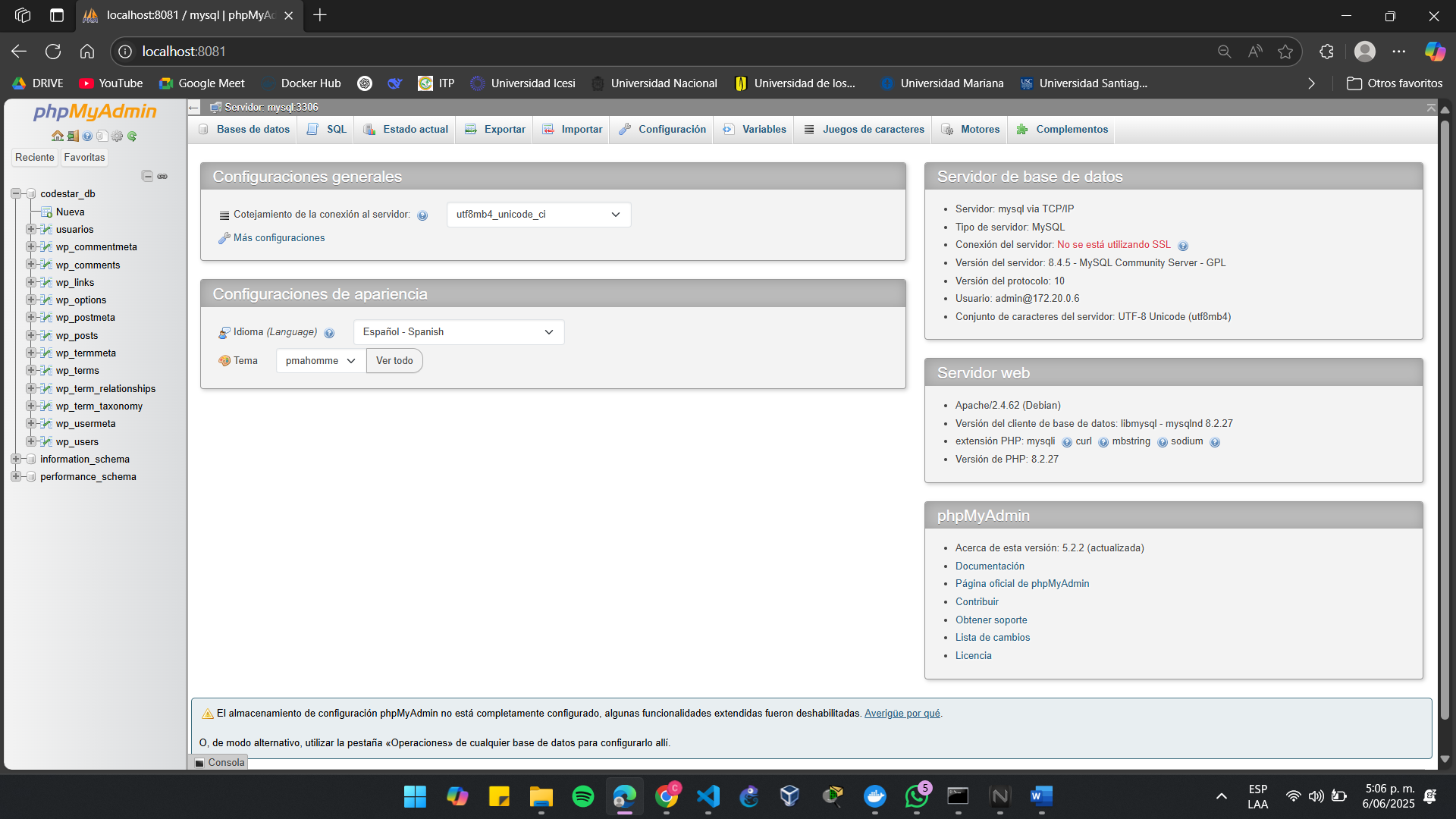
* Qué servicios levantar.
* Qué imagen usar para cada uno.
* Cómo se comunican entre sí.
* Qué puertos exponer.
* Qué volúmenes usar para persistencia.

|  |  |
| --- | --- |
| Bloque | Explicación |
| version: '3.8' | Define la versión del esquema de Docker Compose. |
| services: | Contiene todos los contenedores que se van a levantar. |
| mysql, phpmyadmin, etc. | Cada uno es un contenedor distinto con su configuración. |
| environment: | Variables de entorno (como contraseñas o configuraciones necesarias). |
| volumes: | Para persistencia de datos (ej. bases de datos). |
| ports: | Define los puertos expuestos en tu máquina (host:contenedor). |
| depends\_on: | Indica que un servicio debe levantarse después de otro. |
| networks: | Todos los servicios están en una red compartida para poder comunicarse entre sí. |

Servicios Implementados

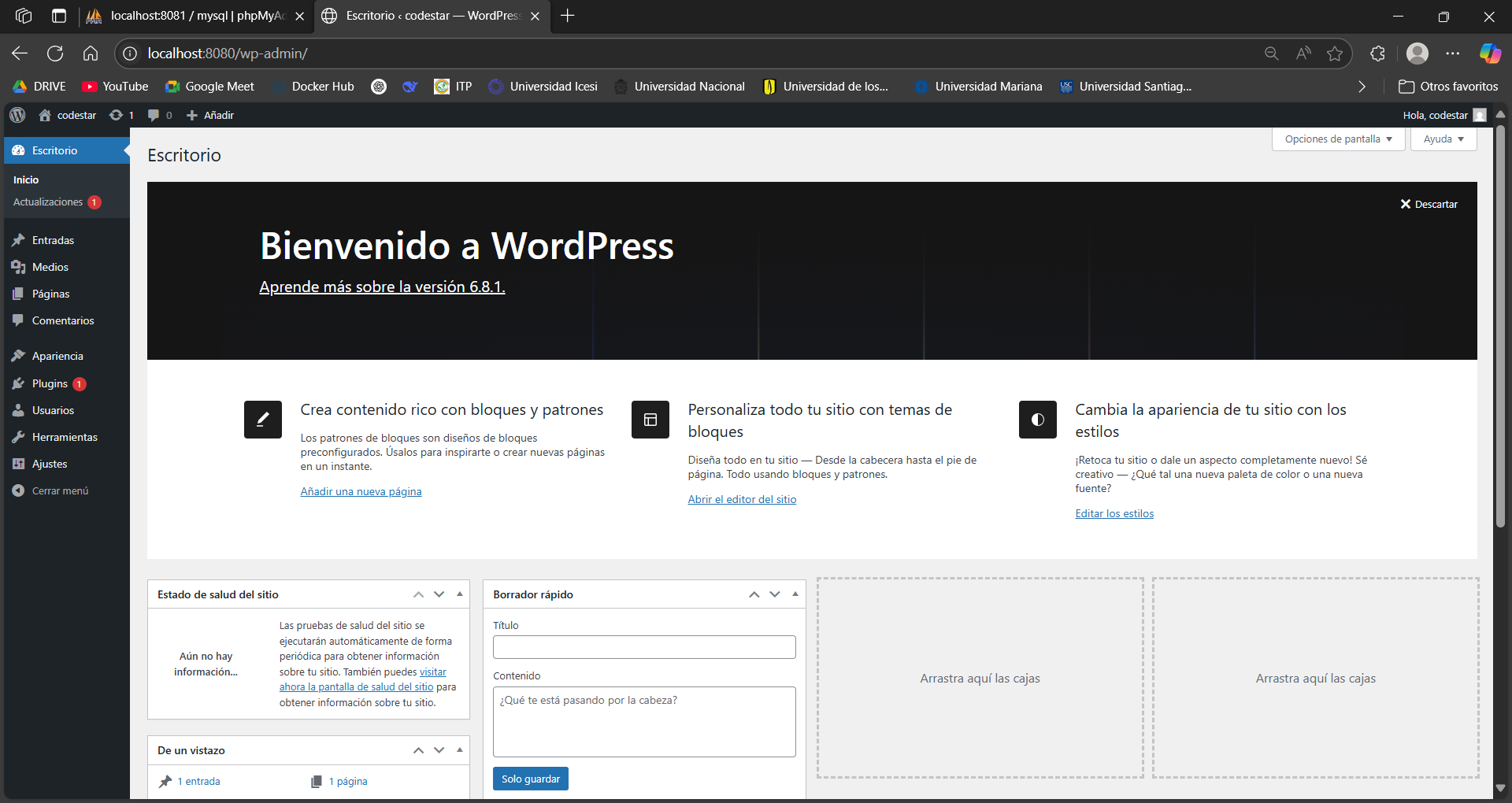
2. phpMyAdmin

* Imagen utilizada: phpmyadmin/phpmyadmin
* Conectado al servicio mysql
* Accesible desde: <http://localhost:8081>



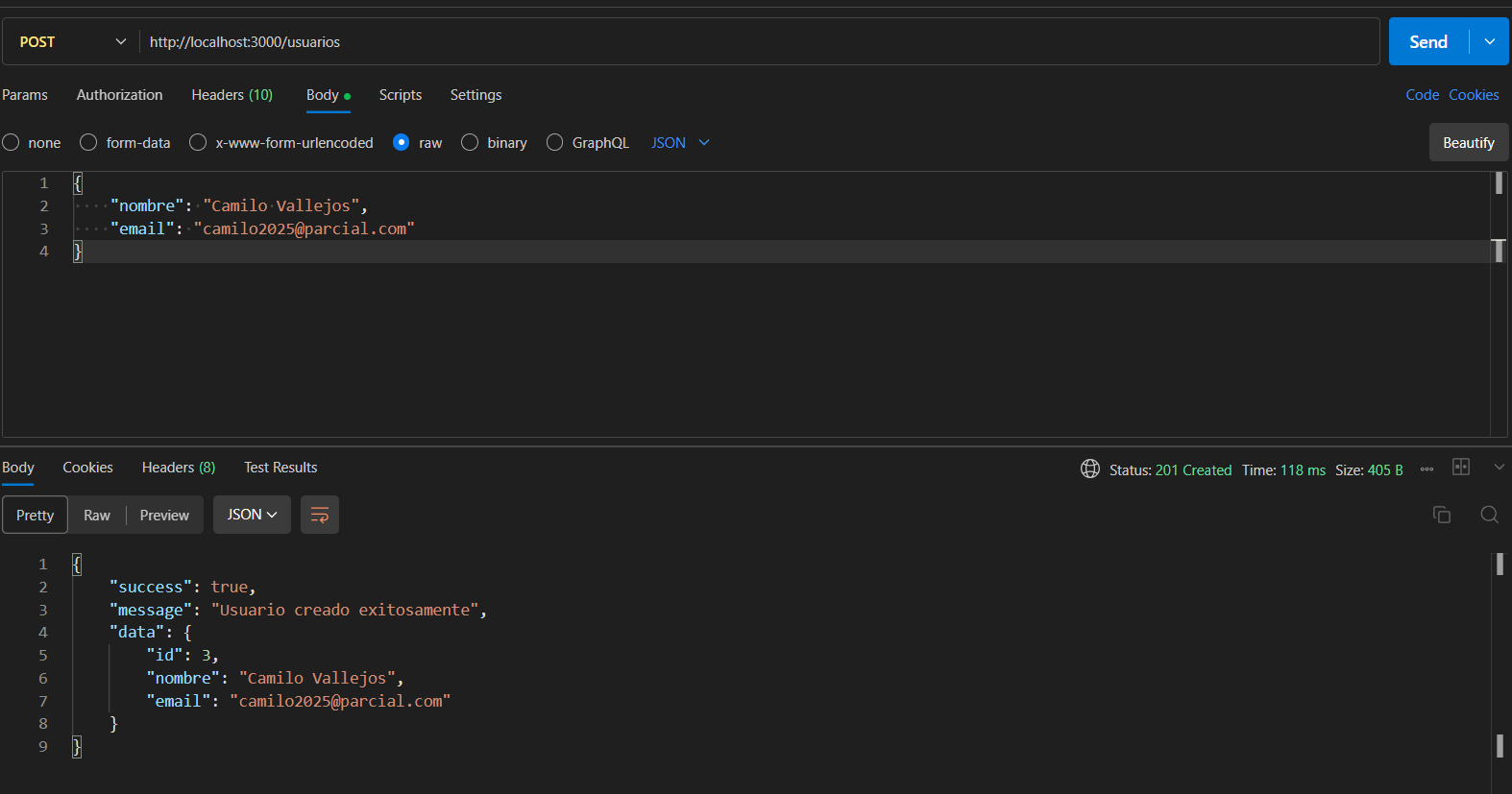
3. WordPress

* Imagen: wordpress:latest
* Conectado al mismo contenedor de MySQL.
* Proxy inverso configurado en Nginx para redirigir desde /.
* Accesible desde: <http://localhost:8080>



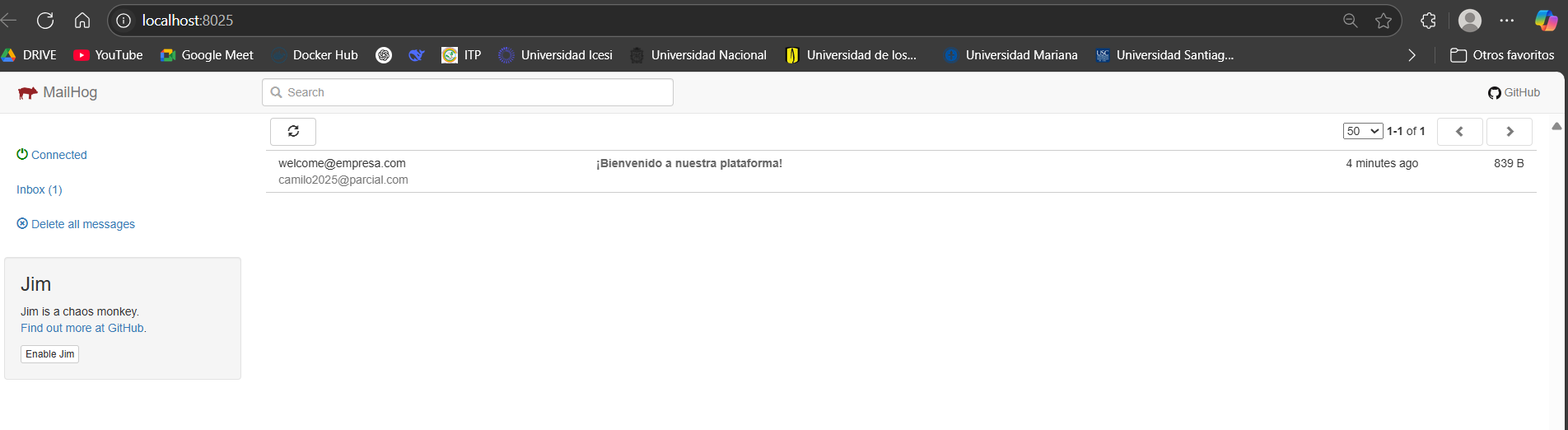
4. Node.js API

* API creada desde cero en ./backend/index.js.
* Construida con un Dockerfile.
* Se conecta a MySQL y realiza operaciones CRUD.



5. Mailhog

* Imagen: mailhog/mailhog
* Puerto: 8025
* Permite visualizar correos enviados desde la API.
* Accesible desde: <http://localhost:8025/>

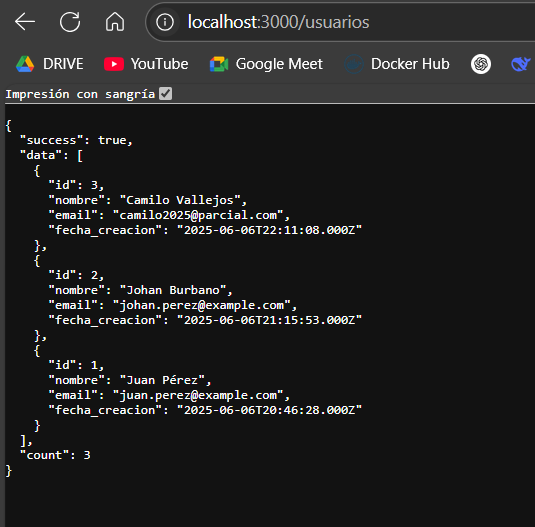


6. Nginx

* Imagen: nginx:latest
* Configurado como proxy inverso para:

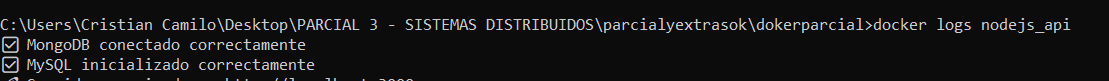
1. / → WordPress
2. /api → Node.js

* Usa el archivo nginx/default.conf.



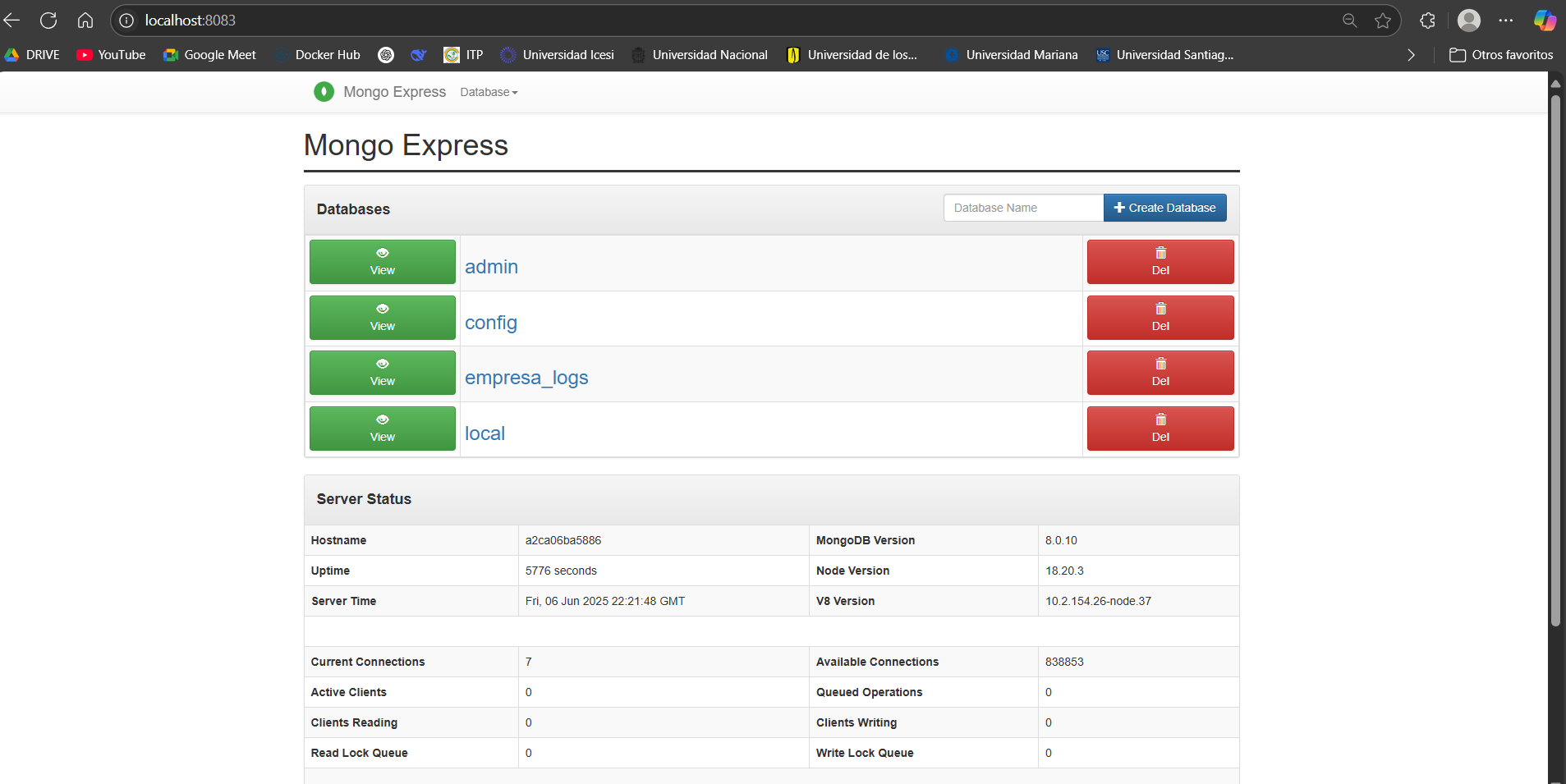
7. MongoDB

* Imagen: mongo:latest
* Puerto expuesto: 27017
* Utilizado para almacenar logs de la API.



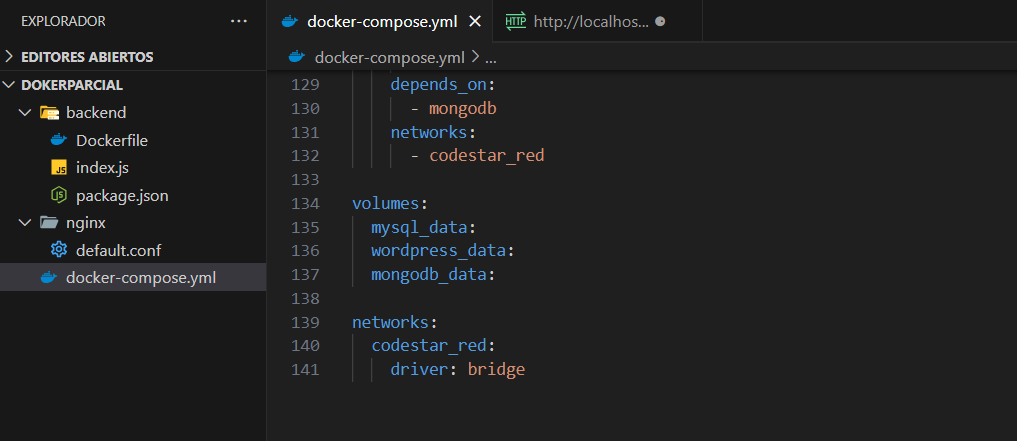
8. AdminMongo

* Imagen: mrvautin/adminmongo
* Conectado a MongoDB.
* Accesible desde: <http://localhost:8083>

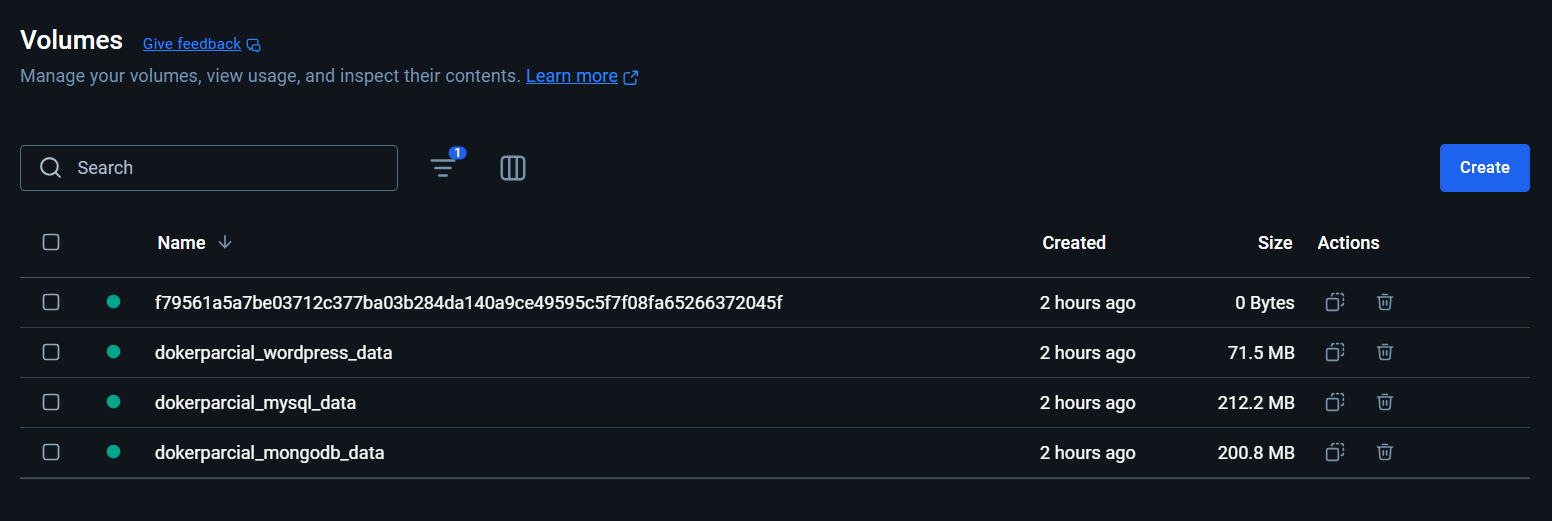


Redes y Volúmenes

* Se utilizó una red Docker bridge personalizada para conectar los servicios.



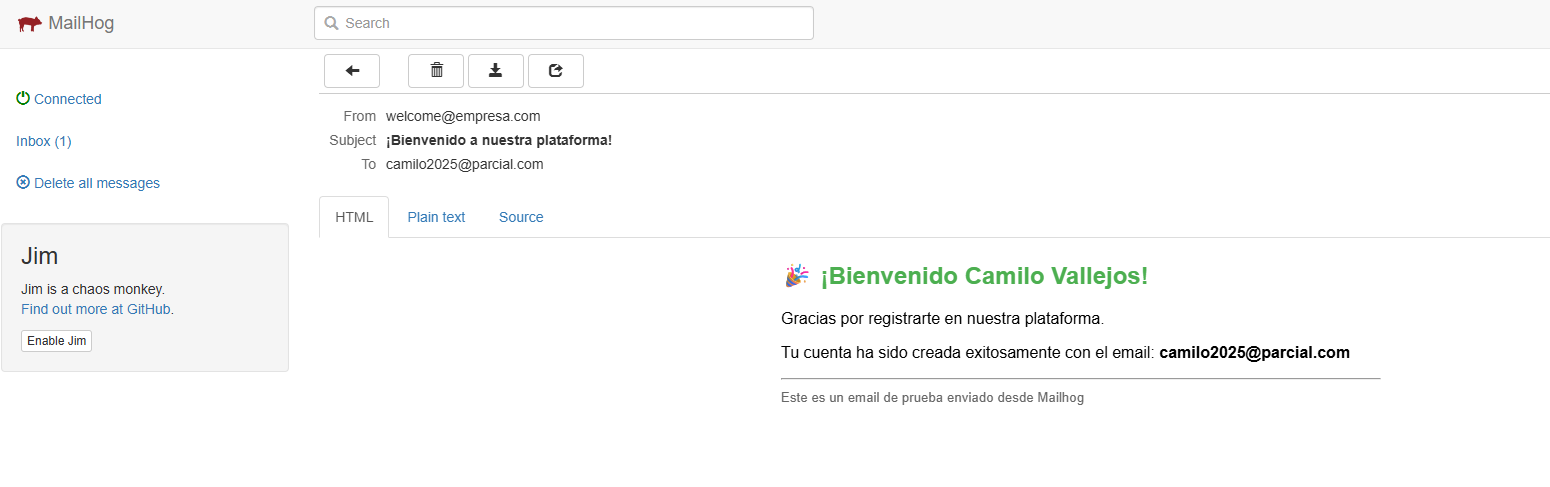
* Se declararon volúmenes persistentes para MySQL y MongoDB.



Funcionalidades Adicionales

Envío de Correos

La API tiene funcionalidad para enviar correos, los cuales se visualizan en Mailhog.



Logs en MongoDB

La API almacena logs de operaciones en MongoDB como respaldo.

