**Sistema Empresarial con Docker Compose Documentación Completa**

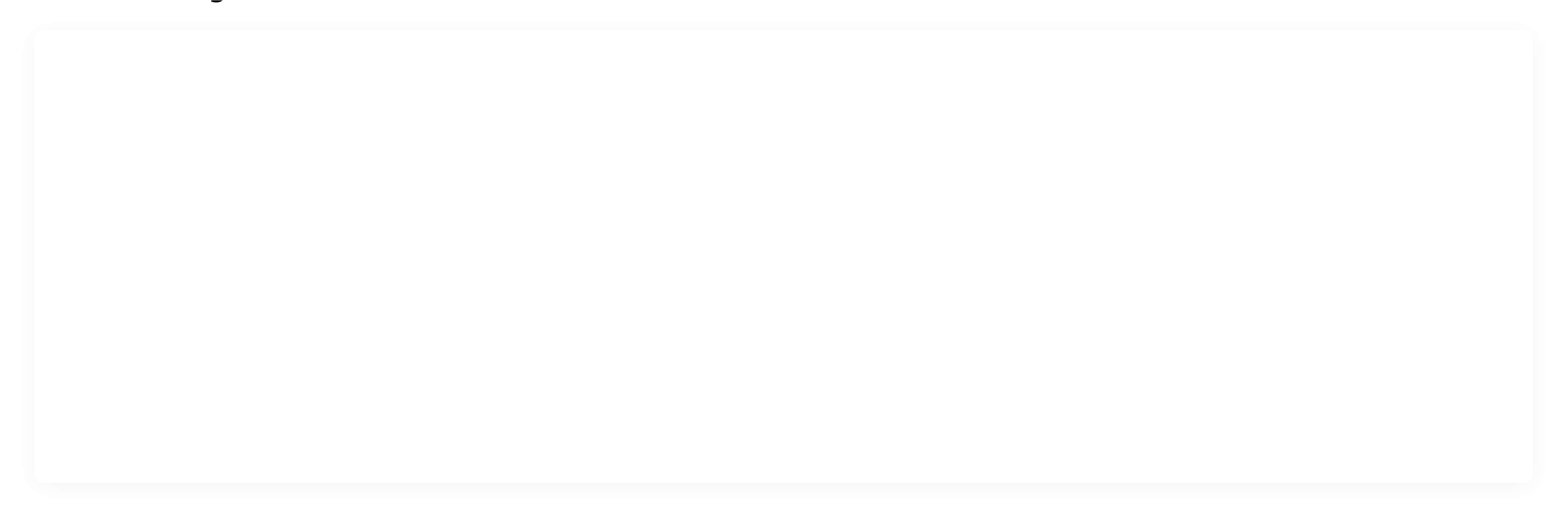
**Desarrollado por:** Jhonier Pasos, Daniel Mora  
**Fecha:** Junio 2025  
**Objetivo:** Orquestación de 8 servicios diferentes usando Docker Compose

**Resumen del Proyecto**

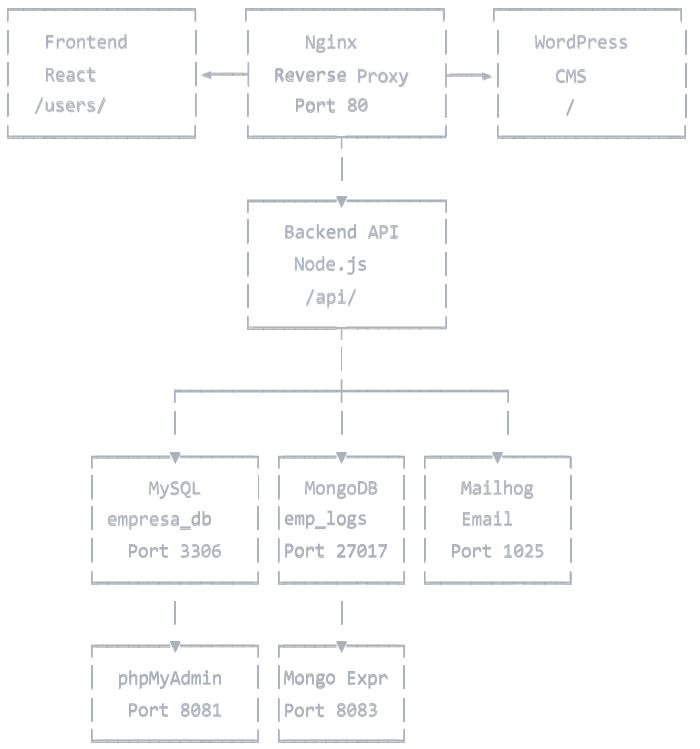
Sistema completo que simula un entorno empresarial con 8 servicios interconectados:

* **WordPress** - CMS principal
* **API Node.js** - Backend con CRUD completo
* **Frontend React** - Interfaz para gestión de usuarios
* **MySQL** - Base de datos principal
* **MongoDB** - Base de datos para logs
* **phpMyAdmin** - Administrador de MySQL
* **Mongo Express** - Administrador de MongoDB
* **Mailhog** - Servidor de pruebas de email
* **Nginx** - Reverse proxy

**URLs y Accesos del Sistema**



**Arquitectura del Sistema**

****

**Funcionalidades Implementadas**

**1. API Node.js (Backend)**

* **CRUD Completo de Usuarios**
  + GET /api/usuarios - Listar usuarios
  + POST /api/usuarios - Crear usuario
  + GET /api/usuarios/:id - Obtener usuario
  + PUT /api/usuarios/:id - Actualizar usuario
  + DELETE /api/usuarios/:id - Eliminar usuario
* **Sistema de Logs**
  + GET /api/logs - Ver logs
  + GET /api/logs/stats - Estadísticas
  + Almacenamiento automático en MongoDB
* **Sistema de Correos**
  + POST /api/send-email - Enviar emails
  + Email automático de bienvenida
  + Integración con Mailhog
* **Health Check**
  + GET /api/health - Estado del sistema

**2. Bases de Datos**

**MySQL (empresa\_db)**

* **Tablas implementadas:**
  + usuarios - Gestión de usuarios
  + configuraciones - Configuraciones del sistema
* **Acceso:** phpMyAdmin (<http://localhost:8081>)

**MongoDB (empresa\_logs)**

* **Colecciones:**
  + logs - Registro de actividades del sistema
* **Acceso:** Mongo Express (<http://localhost:8083>)

**3. Sistema de Correos**

* **Mailhog** para pruebas de envío
* **Emails automáticos** al crear usuarios
* **Interfaz web** para ver emails enviados

**Evidencias de Funcionamiento**

**1. phpMyAdmin - Base de Datos MySQL**

Mostrar imagen

* Conectado a MySQL en puerto 3306
* Base de datos empresa\_db creada
* Tabla logs visible con datos
* Consultas SQL funcionando

**2. Mailhog - Sistema de Correos**

Mostrar imagen

* Servidor SMTP funcionando en puerto 1025
* Interfaz web en puerto 8025
* Email de bienvenida enviado automáticamente
* Destinatario: [juans@test.com](mailto:juans@test.com)
* Asunto: "¡Bienvenido a nuestra plataforma!"

**3. API Backend - Health Check**

Mostrar imagen

* API respondiendo en /api/health
* Estado: "healthy"
* Uptime: 67.68 segundos
* Servicios conectados: MySQL, MongoDB, Mailhog

**4. WordPress - CMS**

Mostrar imagen

* WordPress 6.8.1 instalado
* Panel de administración funcionando
* Usuario: jhonier
* Acceso en puerto 8080

**5. Docker Containers**

Mostrar imagen

* 6 contenedores activos
* Todos los servicios corriendo por 18 horas
* CPU y memoria bajo control
* Puertos correctamente mapeados

**6. Mongo Express - Logs**

Mostrar imagen

* MongoDB conectado en puerto 27017
* Base de datos empresa\_logs
* Colección logs con registros
* Logs de inicio del servidor y health checks

**Configuración Técnica**

**Docker Compose Services**

yaml

- mysql:8 (Puerto 3306)

- phpmyadmin (Puerto 8081)

- wordpress:latest (Puerto 8080)

- node:18-alpine (Puerto 3000)

- mongo:7 (Puerto 27017)

- mongo-express (Puerto 8083)

- mailhog (Puertos 1025, 8025)

- nginx:alpine (Puerto 80)

**Volúmenes Persistentes**

* mysql\_data - Datos de MySQL
* mongodb\_data - Datos de MongoDB
* wordpress\_data - Archivos de WordPress
* nginx\_logs - Logs de Nginx

**Red Docker**

* Red compartida: empresa\_network
* Subnet: 172.20.0.0/16
* Comunicación interna entre servicios

**Endpoints de la API**

**Gestión de Usuarios**

bash

*# Listar usuarios*

GET http://localhost/api/usuarios

*# Crear usuario*

POST http://localhost/api/usuarios

{

"nombre": "Juan Pérez",

"email": "juan@ejemplo.com"

}

*# Obtener usuario específico*

GET http://localhost/api/usuarios/1

*# Actualizar usuario*

PUT http://localhost/api/usuarios/1

{

"nombre": "Juan Carlos",

"email": "juan.carlos@ejemplo.com"

}

*# Eliminar usuario*

DELETE http://localhost/api/usuarios/1

**Sistema de Logs**

bash

*# Ver todos los logs*

GET http://localhost/api/logs

*# Filtrar logs por nivel*

GET http://localhost/api/logs?level=info&limit=10

*# Estadísticas de logs*

GET http://localhost/api/logs/stats

**Sistema de Correos**

bash

*# Enviar email*

POST http://localhost/api/send-email

{

"to": "test@ejemplo.com",

"subject": "Prueba",

"message": "Mensaje de prueba"

}

**Comandos para Ejecutar**

**Iniciar el sistema**

bash

*# Levantar todos los servicios*

docker-compose up -d --build

*# Ver logs*

docker-compose logs -f

*# Ver estado*

docker-compose ps

**Comandos de prueba**

bash

*# Probar API*

curl http://localhost/api/health

*# Crear usuario*

curl -X POST http://localhost/api/usuarios \

-H "Content-Type: application/json" \

-d '{"nombre": "Test User", "email": "test@test.com"}'

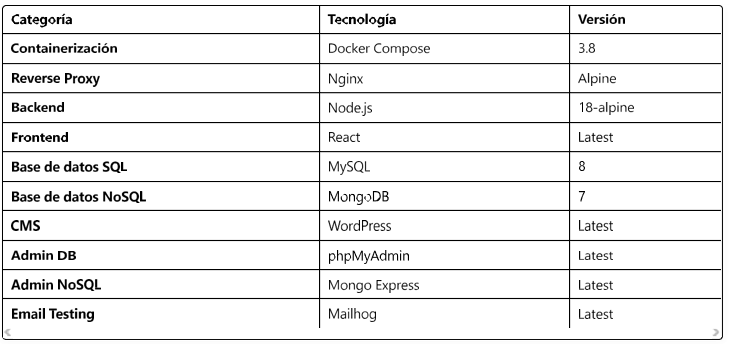
*# Enviar email*

curl -X POST http://localhost/api/send-email \

-H "Content-Type: application/json" \

-d '{"to": "test@test.com", "subject": "Prueba", "message": "Hola!"}'

**Tecnologías Utilizadas**

****