Arquitectura del Sistema

Esta sección describe los componentes técnicos que conforman la solución DevOps del proyecto final. El objetivo es diseñar una arquitectura moderna, modular y preparada para escalar, aplicando buenas prácticas de desarrollo y despliegue continuo.

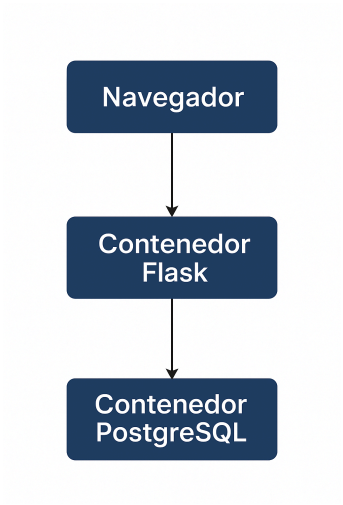
Componentes Principales

- Lenguaje de programación: Python 3.9
- Framework web: Flask
- Contenerización: Docker
- Orquestación local: Docker Compose
- Base de datos: PostgreSQL (contenedorizado)
- Gestión de dependencias: requirements.txt
- Pruebas unitarias: pytest + pytest-cov (mínimo 80% de cobertura)
- Estilo de código: flake8
- Configuración: Variables de entorno gestionadas en docker-compose.yml
- Balanceador de carga: Nginx
- Certificados SSL: ACM de AWS
- Telemetría y logs: logging (desarrollo) / CloudWatch (producción)

Diagrama Descriptivo de la Arquitectura Local

La arquitectura local de desarrollo está compuesta por dos servicios principales contenidos en Docker: la aplicación Flask y una base de datos PostgreSQL. Ambos servicios se levantan simultáneamente utilizando Docker Compose.

Diagrama lógico:



Arquitectura Cloud (en AWS)

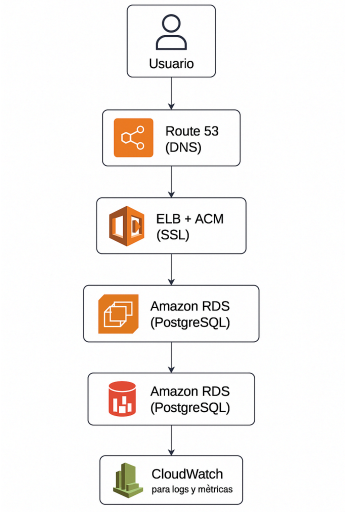
Esta sección describe cómo se traslada la solución actual a una arquitectura Cloud utilizando servicios de Amazon Web Services (AWS). La elección de servicios responde a la necesidad de escalar, automatizar y garantizar disponibilidad y seguridad.

Servicios de AWS

Servicio	Descripción
EC2 (Elastic Compute Cloud)	Para desplegar la aplicación Flask en contenedores o instancias virtuales.
ECS (Elastic Container Service)	Alternativa moderna para desplegar contenedores gestionados.
RDS (Relational Database Service)	Para alojar la base de datos PostgreSQL de forma segura y escalable.
Elastic Load Balancer (ELB)	Balanceador de carga para distribuir tráfico entre instancias EC2 o tareas ECS.
ACM (AWS Certificate Manager)	Para gestionar y aplicar certificados SSL/TLS.
CloudWatch	Para recopilar logs, métricas y configurar alarmas del sistema.
ECR (Elastic Container Registry)	Repositorio privado de imágenes Docker.

IAM (Identity and Access Management)	Gestión de permisos y roles de los recursos.
Route 53	Para gestionar DNS y enrutar tráfico hacia el balanceador de carga.

Diagrama Lógico de Arquitectura en AWS



Consideraciones de Seguridad y Escalabilidad

- Uso de grupos de seguridad para restringir tráfico entrante/saliente
- Roles de IAM para acceso limitado a servicios
- Escalado automático (Auto Scaling Groups o ECS Fargate)
- Monitorización de salud y alertas vía CloudWatch
- Certificados SSL gestionados automáticamente con ACM

Descripción del ciclo de vida del software

Esta sección describe cómo se gestiona el ciclo completo de vida del software, desde el desarrollo local hasta su despliegue en producción, incluyendo flujos de trabajo, estrategias de control de versiones, tipos de entornos y operaciones de mantenimiento.

Modelo de Desarrollo

Se utiliza un modelo basado en ramas de Git para organizar el desarrollo colaborativo:
- Rama `main`: versión estable para producción.
- Rama `develop`: rama de integración para pruebas.
- Ramas `feature/<nombre>`: desarrollo de nuevas funcionalidades.
- Ramas `fix/<nombre>`: corrección de errores.

Tipos de Entornos

Se distinguen tres entornos principales:
- Local: desarrollo con Docker Compose.
- Staging: pruebas internas antes del paso a producción (opcional).
- Producción: entorno definitivo desplegado en AWS.

Tipos de Pruebas Obligatorias

- Pruebas unitarias automatizadas con `pytest`.
- Objetivo mínimo del 80% de cobertura.
- Revisión de código manual en cada Pull Request.
- Linting con `flake8` para mantener el estilo PEP8.

Modelo de Operaciones y Despliegue

El despliegue se realizará utilizando contenedores Docker alojados en EC2 o ECS.
Se aplicará una estrategia de despliegue rolling update para minimizar tiempo de inactividad.
Los despliegues están vinculados a versiones etiquetadas (`tags`) o ramas específicas (`main`).

Logs y Monitoreo

En producción, los logs y métricas se enviarán a AWS CloudWatch para su centralización.
En desarrollo, los logs se imprimen por consola para facilitar la depuración.