BUAP

Benemérita Universidad

Autónoma de Puebla

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

CURSO: SERVICIOS WEB

DOCENTE: GUSTAVO EMILIO MENDOZA OLGUIN

ALUMNOS:

JAVIER AGUILAR MACIAS 202142536

JOSÉ ANTONIO COYOTZI JUAREZ 202118536

CONTENIDO

INTRODUCCIÓN

Este proyecto implementa una arquitectura orientada a servicios que integra bases de datos SQL (PostgreSQL) y NoSQL (MongoDB) mediante un sistema de comunicación basado en mensajería RPC con RabbitMQ. La solución está diseñada para manejar operaciones CRUD, consultas complejas (JOIN) y agregaciones, proporcionando una interfaz web desarrollada en Flask que permite a los usuarios enviar solicitudes de manera intuitiva. El diseño aprovecha microservicios desacoplados para garantizar escalabilidad y flexibilidad, donde cada componente (interfaz, API Gateway, servicios de base de datos) se comunica a través de colas de mensajes, asegurando resiliencia frente a picos de demanda. La arquitectura refleja mejores prácticas modernas, como el uso de brokers de mensajería para orquestación y la capacidad de cambiar entre motores de base de datos sin modificar el código cliente.

ARQUITECTURA PROPUESTA

El sistema se compone de cuatro capas principales:

Interfaz Web (Flask): Una aplicación HTML/Bootstrap que captura las consultas del usuario, ya sea en SQL directo o JSON para MongoDB, y las envía al API Gateway mediante HTTP. Esta capa oculta la complejidad del backend, permitiendo al usuario seleccionar el motor de base de datos y tipo de operación (CREATE, READ, etc.) desde un formulario simplificado.

API Gateway y Message Broker: El Gateway, implementado en Flask, recibe las solicitudes HTTP, valida su estructura y las publica en una cola RabbitMQ (db_rpc_queue). RabbitMQ actúa como intermediario, enrutando cada mensaje al servicio correspondiente (SQL o NoSQL) mediante un patrón RPC. Aquí se garantiza el desacoplamiento: los servicios no conocen el origen de las peticiones, solo procesan mensajes y devuelven respuestas a colas temporales vinculadas a cada cliente.

Servicios de Base de Datos:

SQL Service: Un worker dedicado que recibe mensajes de RabbitMQ, ejecuta consultas dinámicas en PostgreSQL (con soporte para múltiples bases de datos mediante el parámetro dbname), y maneja transacciones con commit/rollback. Soporta operaciones como JOIN y agregaciones con SQL puro.

NoSQL Service: Procesa mensajes para MongoDB, realizando inserciones, consultas con filtros, actualizaciones y pipelines de agregación. Usa colecciones flexibles y devuelve resultados en formato JSON.

Flujo de Respuestas: Las respuestas de los workers viajan de vuelta a través de RabbitMQ hasta el API Gateway, que las renderiza en la interfaz web. Este ciclo síncrono (solicitud-respuesta) asegura que el usuario reciba feedback inmediato, mientras que el uso de colas garantiza que ninguna petición se pierda, incluso si un servicio está temporalmente sobrecargado.

DIAGRAMA DE ARQUITECTURA

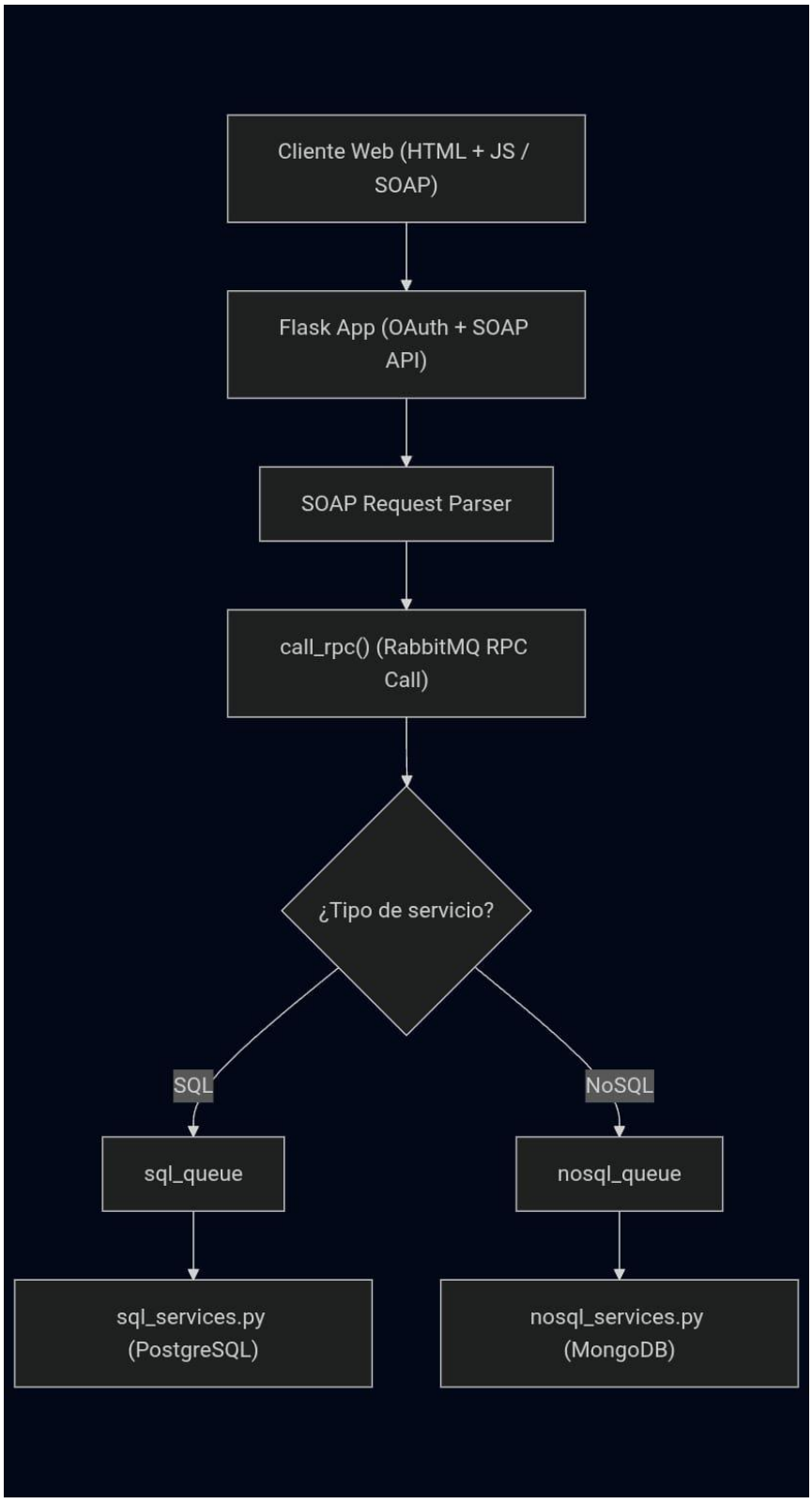

Figura 1. Diagrama de bloques y componentes de la arquitectura.

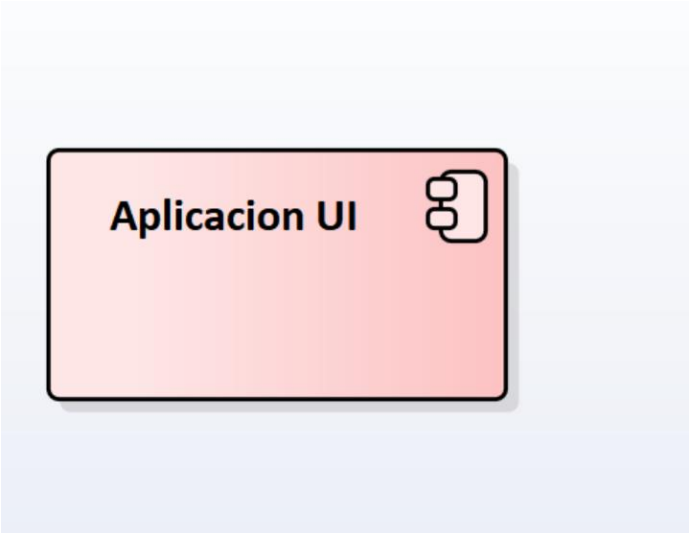

Figura 3. Componente "Cliente" de la aplicación UI y sus integraciones.

El módulo Cliente se refiere a la Aplicación UI, que puede ser web o móvil. Es la interfaz con la que interactúa el usuario final. Su función principal es capturar datos, visualizar resultados y desencadenar acciones en el sistema a través del API Gateway. Aunque parece un componente simple, es fundamental para la experiencia de usuario y debe estar bien integrado con los servicios de autenticación y lógica del backend. Su diseño debe tener en cuenta la asincronía y los posibles tiempos de espera por la naturaleza serverless del sistema.

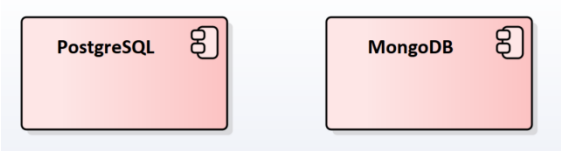

Figura 4. Componente Almacenamiento de datos: PostgerSQL, y MongoDB

El paquete Almacenamiento de Datos agrupa dos servicios gestionados en la nube: PostgreSQL Managed y MongoDB Atlas. PostgreSQL aporta un motor relacional probado, capaz de manejar consultas complejas, joins y operaciones de agregación con alto rendimiento y consistencia transaccional. MongoDB Atlas, por su parte, brinda flexibilidad de esquemas y escalado horizontal para datos semiestructurados, facilitando modelos de documento dinámicos. Al delegar esto a servicios gestionados, reducimos la carga operativa y obtenemos alta disponibilidad y seguridad integrada sin invertir en infraestructura propia.

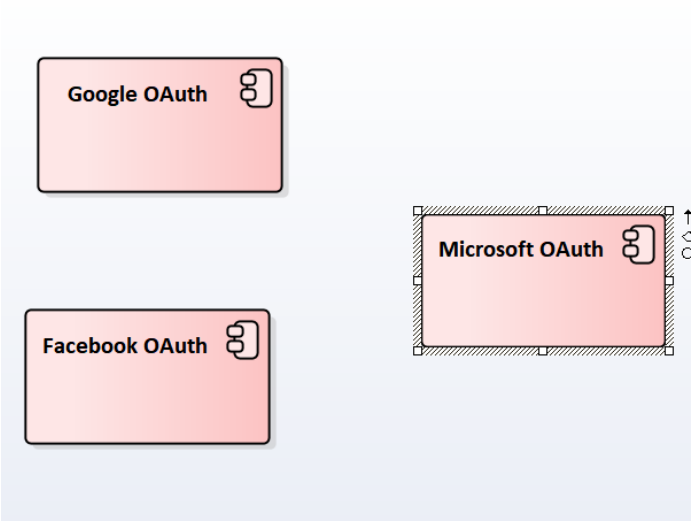

Figura 5. Integraciones Externas.

Integraciones Externas

Las Integraciones Externas permiten ampliar las capacidades de autenticación del sistema utilizando proveedores conocidos como

Facebook, Google y Microsoft a través de OAuth. Estas integraciones facilitan el inicio de sesión social, reduciendo la fricción para los usuarios y evitando la necesidad de gestionar credenciales directamente. También ayudan a aumentar la adopción al ofrecer confianza y conveniencia. Desde una perspectiva de arquitectura, estas integraciones requieren la configuración adecuada de Cognito como federador de identidades, así como ajustes en la interfaz cliente.

DIAGRAMA DE DE FLUJO(Comunicación SOAP)

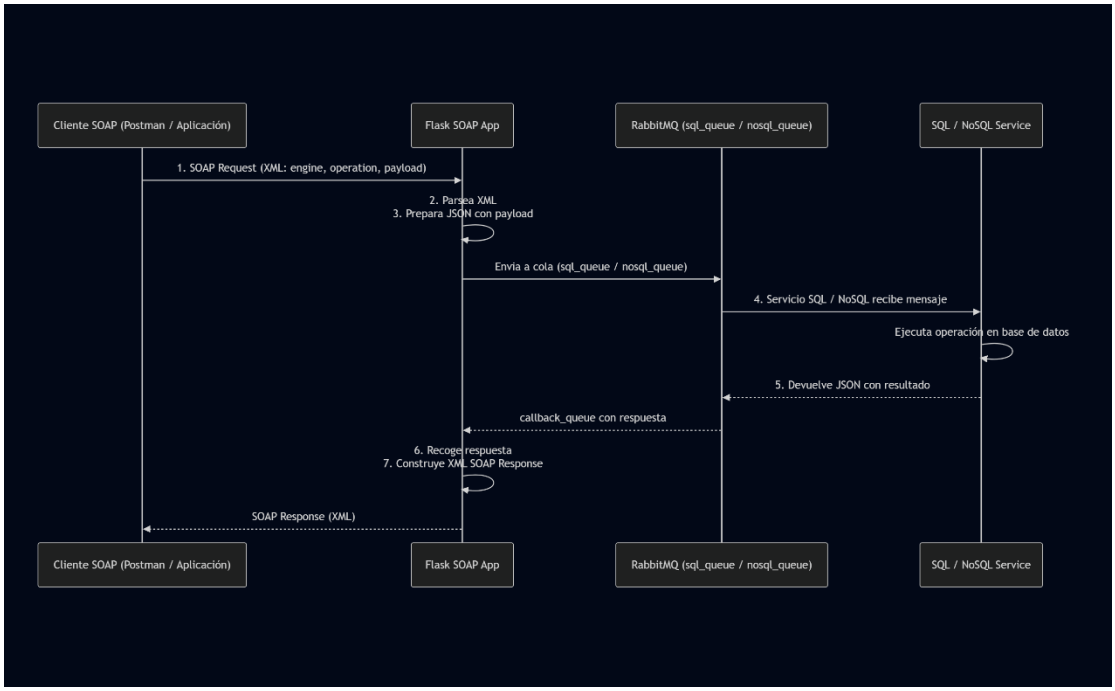

Figura 7. Diagrama de flujo del proceso de autenticación OAuth2 con Cognito, incluyendo ruta alternativa en caso de error, asegurando trazabilidad y control de fallos en el inicio de sesión federado.

El diagrama de flujo del proceso de autenticación con Cognito (OAuth2) representa la interacción entre el cliente, el sistema serverless y los proveedores de identidad (como Facebook, Google o Microsoft). Este flujo asegura que, al momento de iniciar sesión, el usuario sea redirigido a un proveedor de autenticación externa y, al validar correctamente sus credenciales, se emita un token de acceso válido que es procesado por el Lambda Orquestador. En caso de que se reciba un error —ya sea por credenciales inválidas, rechazo del usuario o problemas de red— el sistema captura dicho error, registra el incidente de forma segura y responde al cliente con un mensaje claro y controlado. Este manejo garantiza una experiencia de usuario robusta, segura y tolerante a fallos, cumpliendo con buenas prácticas de autenticación federada en arquitecturas serverless.

DIAGRAMA DE SECUENCIA READ

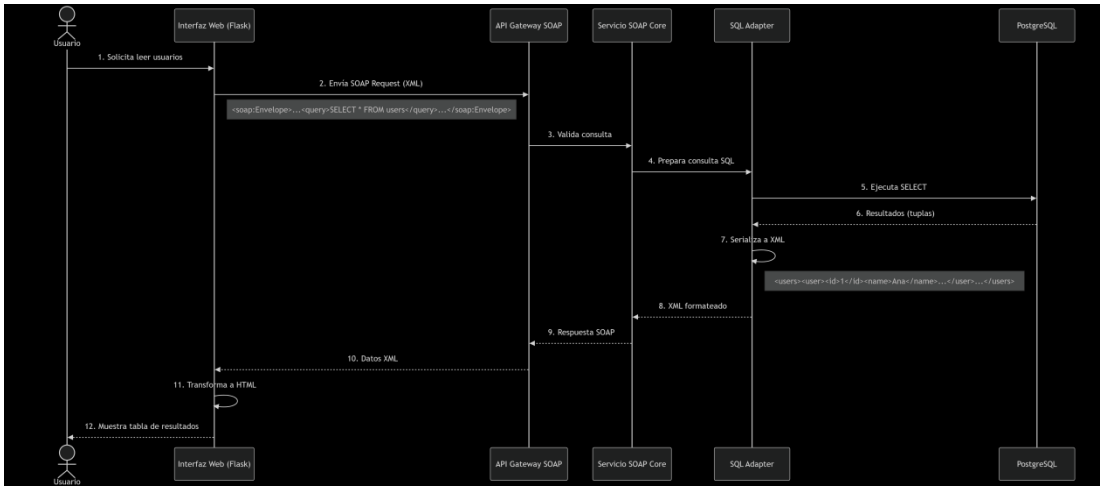

Figura 9. Secuencia de lectura validada por roles y compatible con SQL y NoSQL.