Universidad Politécnica de Chiapas.

Cuatrimestre.	Septiembre- Diciembre 2025
Grupo.	Noveno B
Asignatura.	SOA
Corte	C2
Actividad.	Patrones de diseño y comunicación adecuados
Fecha	
Matricula.	221189 231208 221214 231198
Nombre.	Luis Alberto Batalla. Willber Hernandez. Esduardo Palomeque Roblero. Maria Fernanda Sanchez

Patrones de Diseño y Comunicación: Reconexión Humana

Versión: 1.0 Fecha: 23 de Octubre de 2025

1. Patrones de Arquitectura y Diseño Aplicados

La arquitectura de "Reconexión Humana" se basa en un conjunto de patrones de diseño probados para construir sistemas de microservicios robustos y escalables.

1.1. API Gateway

- Descripción: Todas las peticiones de los clientes (aplicaciones móviles y web) pasan a través de un único punto de entrada, el API Gateway.
- Aplicación: El Gateway es responsable de:
  - Enrutamiento: Dirigir las peticiones al microservicio correspondiente (ej. /api/posts a SocialConnect, /api/auth/login a AuthIdentity).
  - Autenticación: Orquestar la validación de tokens JWT llamando al endpoint /token/validate de AuthIdentity antes de pasar la petición al servicio de destino.
  - Agregación: (Opcional) Componer respuestas a partir de múltiples servicios.
- Justificación: Simplifica la lógica del cliente, proporciona un punto central para la seguridad (rate limiting, WAF) y oculta la topología interna de los microservicios.

1.2. Database per Service

- Descripción: Cada microservicio es el único propietario de sus datos y su esquema de base de datos. Ningún otro servicio puede acceder directamente a la base de datos de otro.
- Aplicación: AuthIdentity gestiona su base de datos PostgreSQL, SocialConnect gestiona su conjunto de bases de datos políglotas (MongoDB, Neo4j, etc.), y así sucesivamente.
- Justificación: Garantiza un bajo acoplamiento. Permite que cada servicio evolucione su esquema de datos de forma independiente y elija la tecnología de base de datos más adecuada para su tarea específica, como se ve en el enfoque de persistencia políglota de SocialConnect.

1.3. Arquitectura Orientada a Eventos (Event-Driven Architecture)

- Descripción: La comunicación asincrónica entre microservicios se realiza mediante la publicación y consumo de eventos a través de un broker de mensajes (RabbitMQ).
- Aplicación:
  - AuthIdentity publica UserRegistered para que otros servicios reaccionen.
  - SocialConnect publica eventos de actividad (PostCreated, CommentAdded) que son consumidos por RiskMitigation.
  - RiskMitigation publica InterventionRequired para desacoplar la lógica de riesgo de la acción de notificar.
- Justificación: Promueve el desacoplamiento total, la resiliencia (si un consumidor está caído, el evento puede ser procesado más tarde) y la escalabilidad (se pueden añadir nuevos consumidores sin modificar a los productores).

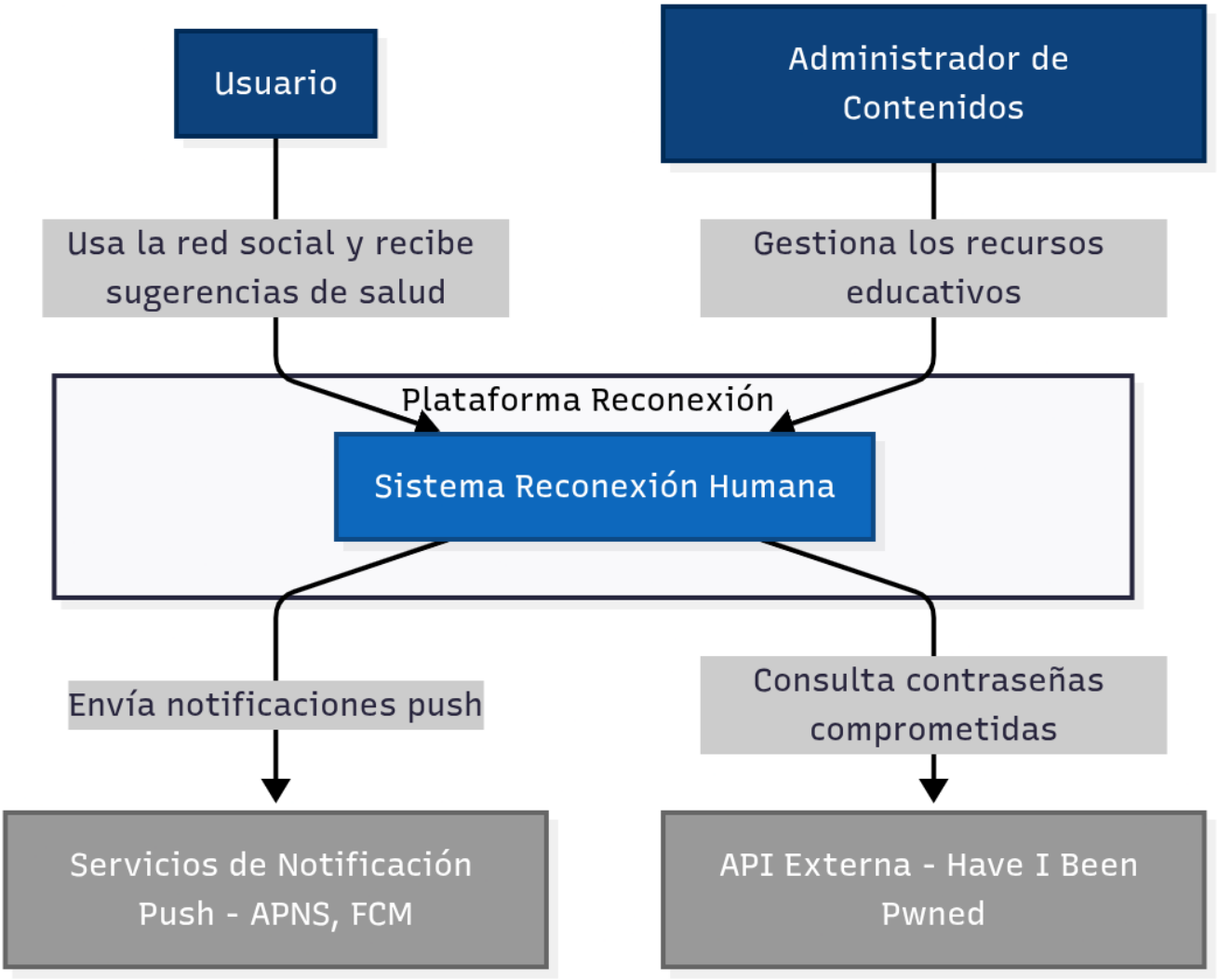
1.4. CQRS (Command Query Responsibility Segregation)

- Descripción: Este patrón segrega las operaciones que modifican datos (Comandos) de las que leen datos (Consultas).
- Aplicación (Sugerida para SocialConnect):
  - Modelo de Escritura (Comandos): Las operaciones como POST /posts, POST /comments se dirigen a un modelo y base de datos optimizados para escrituras rápidas y consistentes (ej. PostgreSQL o MongoDB). Al completarse, publican un evento (ej. PostWasCreated).
  - Modelo de Lectura (Consultas): Las operaciones complejas como GET /feed leen de un modelo de datos desnormalizado y optimizado para lecturas rápidas (ej. una tabla en Cassandra o un documento en Elasticsearch). Este modelo de lectura se actualiza de forma asíncrona escuchando los eventos del modelo de escritura.
- Justificación: Permite escalar los modelos de lectura y escritura de forma independiente. Un feed de noticias tiene muchas más lecturas que escrituras, por lo que optimizar su modelo de datos por separado mejora drásticamente el rendimiento y la experiencia del usuario.

2. Diagramas de Arquitectura

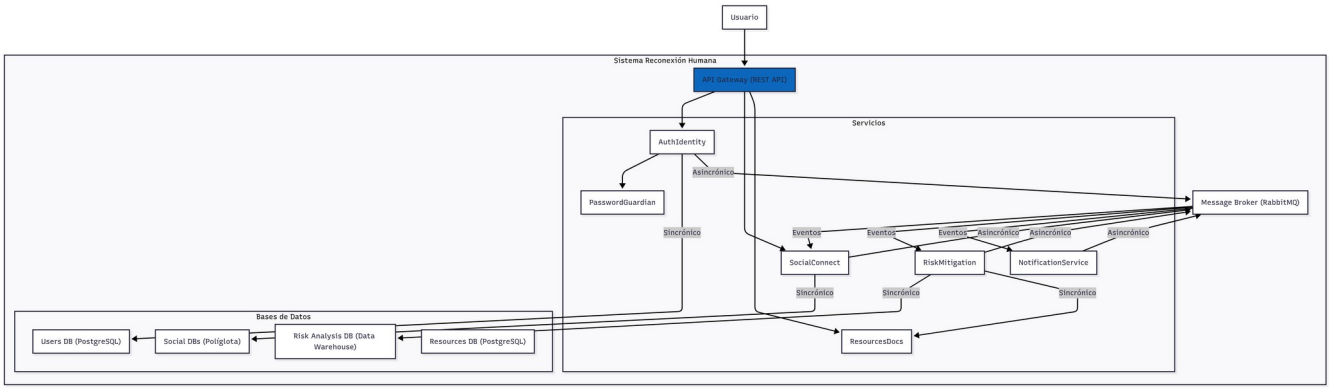
2.1. Diagrama de Contexto (C4 - Nivel 1)

Este diagrama muestra una vista de alto nivel del sistema, sus usuarios y las interacciones principales.



2.2. Diagrama de Contenedores (C4 - Nivel 2)

Este diagrama descompone el sistema en sus microservicios principales ("contenedores") y muestra cómo interactúan.



2.3. Diagrama de Secuencia: Flujo de Registro de Usuario