Manual de Arquitectura - Sistema EagleView+

1. Introducción

Este documento detalla la arquitectura de software del sistema **EagleView+**, una plataforma orientada a facilitar la búsqueda y recomendación inteligente de contenido audiovisual. Su propósito es proporcionar una visión integral de los componentes del sistema, las decisiones de diseño adoptadas, y las tecnologías que lo componen.

Está dirigido a desarrolladores, arquitectos de software y responsables técnicos involucrados en el desarrollo, mantenimiento y evolución futura de la plataforma.

La arquitectura se basa en un enfoque de tres capas lógicas:

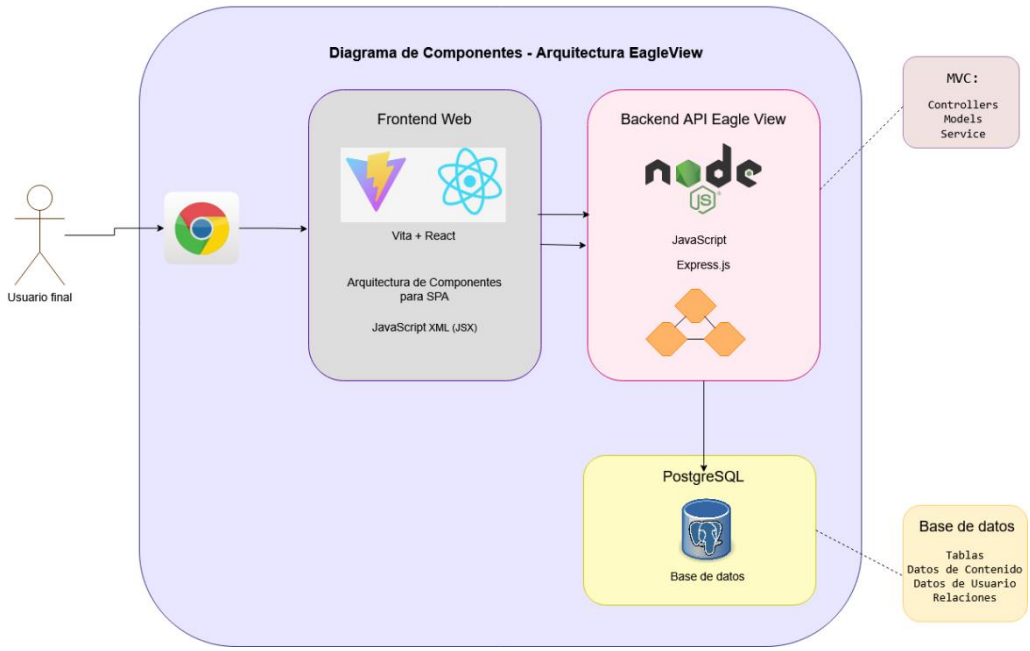
1. Interfaz de usuario (*Frontend*)
2. Lógica de negocio y servicios (*Backend*)
3. Persistencia de datos (*Base de Datos*)

2. Visión General de la Arquitectura

EagleView+ adopta una arquitectura multicapa, permitiendo escalabilidad, mantenibilidad y una clara separación de responsabilidades.

Estructura General

- **Frontend**: Aplicación SPA React que consume una API REST.
- **Backend**: Servidor Node.js con arquitectura MVC + capa de servicios.
- **Base de Datos**: PostgreSQL con modelo entidad-relación.



3. Capas y Componentes

- **Capa de Presentación (Frontend)**

La interfaz gráfica del usuario está implementada como una **Single Page Application (SPA)** utilizando **React** con **Vite** para el entorno de desarrollo rápido y moderno.

Características:

- **Tecnologías**: React, JavaScript (ES6+), JSX, Vite, React Router.
- **Patrón**: Arquitectura basada en **componentes reutilizables y modulares**.
- **Estructura de carpetas**:
    - components/: componentes reutilizables (botones, tarjetas, listas).
    - api.js: centraliza las llamadas HTTP a la API del backend.

Uso de .jsx

Los archivos .jsx combinan JavaScript con sintaxis similar a HTML (JSX), permitiendo crear interfaces declarativas con React. Esta extensión mejora la legibilidad del código y el soporte de herramientas de desarrollo.

- **Capa de Lógica (Backend)**

El backend expone una API RESTful creada con Node.js y Express.js, que orquesta la lógica, seguridad, validación de datos y la comunicación con la base de datos.

Estructura Arquitectónica:

- Controladores (Controller): manejan las solicitudes HTTP y delegan en los servicios.
- Rutas (Routes): encapsulan la lógica de negocio de cada módulo.
- Modelos (Model): definen esquemas y operaciones directas sobre la base de datos.

- **Capa de Persistencia (Base de Datos)**

El sistema utiliza **PostgreSQL** como motor de base de datos relacional para asegurar integridad y rendimiento en el almacenamiento de información.

Modelo Relacional:

- Tablas normalizadas: users, contents, ratings, etc.
- Relaciones mediante claves primarias y foráneas.
- Integridad referencial y restricciones a nivel de esquema.

Acceso a datos:

- Conexión a través del cliente ORM (Sequelize).

4. Justificación Técnica

La elección tecnológica y estructural responde a objetivos clave del sistema:

REQUISITO	SOLUCIÓN TÉCNICA
ESCALABILIDAD Y MANTENIMIENTO	Arquitectura por capas
AGILIDAD EN EL DESARROLLO	JavaScript full-stack (React + Node.js)
MODULARIDAD	Componentes y servicios desacoplados
BUENAS PRÁCTICAS	MVC, control de errores, autenticación
DESACOPLAMIENTO TOTAL	API RESTful independiente del cliente

5. Módulos Adicionales y Consideraciones

Conectividad

El frontend y backend se comunican mediante **Axios** con base en endpoints /api/.

Escalabilidad y despliegue

- El sistema es compatible con despliegues en contenedores Docker.
- Se despliegp en plataformas **Render y Railway.**
- Arquitectura adaptable a microservicios en el futuro.

6. Conclusión

EagleView+ ha sido diseñado con principios de claridad estructural, separación de responsabilidades y uso de tecnologías modernas, lo que permite un desarrollo ágil, un mantenimiento organizado y una base sólida para crecer en funcionalidades, rendimiento y escalabilidad.