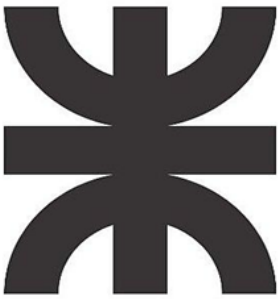
DESARROLLO DE SOFTWARE

Trabajo Práctico Final



UNIVERSIDAD TECNOLÓGICA NACIONAL

FACULTAD REGIONAL RESISTENCIA

Integrantes:

- Vallejos Kehmeier, Adriel
  - Mail: adrielvallejoskehmeier@gmail.com
- Niveiro, Gianfranco
  - Mail: gianfranconiveiro@hotmail.com
- Beron De Astrada, Santiago Agustin
  - Mail: santiagobdea@gmail.com
- Morel, Francisco Ezequiel
  - Mail: morelfrancisco18@hotmail.com
- Bregant, Joaquin Conrado
  - Mail: joabresper@gmail.com
- Leiva Romero, Agustín Nazareno
  - Mail: agustinleiva627@gmail.com

Grupo: 7

Carrera: ISI

- AÑO 2024 -

ÍNDICE

Escenario

La organización de la Bienal Internacional de Escultura del Chaco, se a contactado son su empresa para planificar, analizar, desarrollar e implementar un sistema de gestión que soporte el registro de los eventos, escultores como así también aplicaciones satélites para que los ciudadanos/ publico en general pueda realizar comentarios y votación durante el evento.

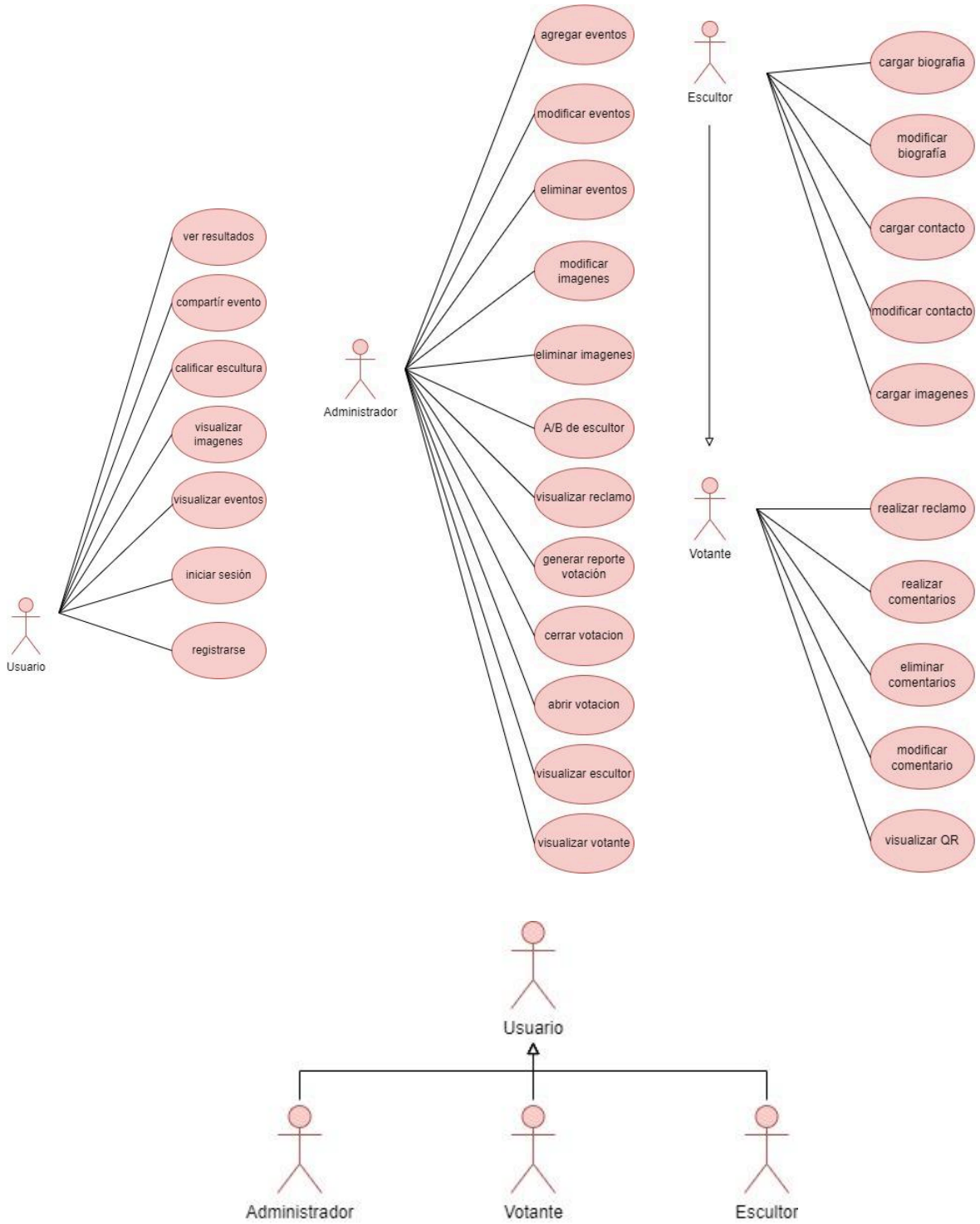
Analisis y Planificacion

Para el desarrollo de este trabajo práctico se siguió un enfoque estructurado que permitió planificar y modelar el sistema antes de pasar a su implementación. Se comenzó con la elaboración del **Modelo de Casos de Uso**, lo que permitió modelar las funcionalidades necesarias para la interacción entre los usuarios y el sistema.

A continuación se realizó el **Diagrama de Clases**, una herramienta fundamental para modelar la estructura estática del sistema, definiendo las clases, sus atributos, métodos y las relaciones entre ellas. Esto ayudó a organizar y entender mejor los componentes del sistema desde una perspectiva orientada a objetos.

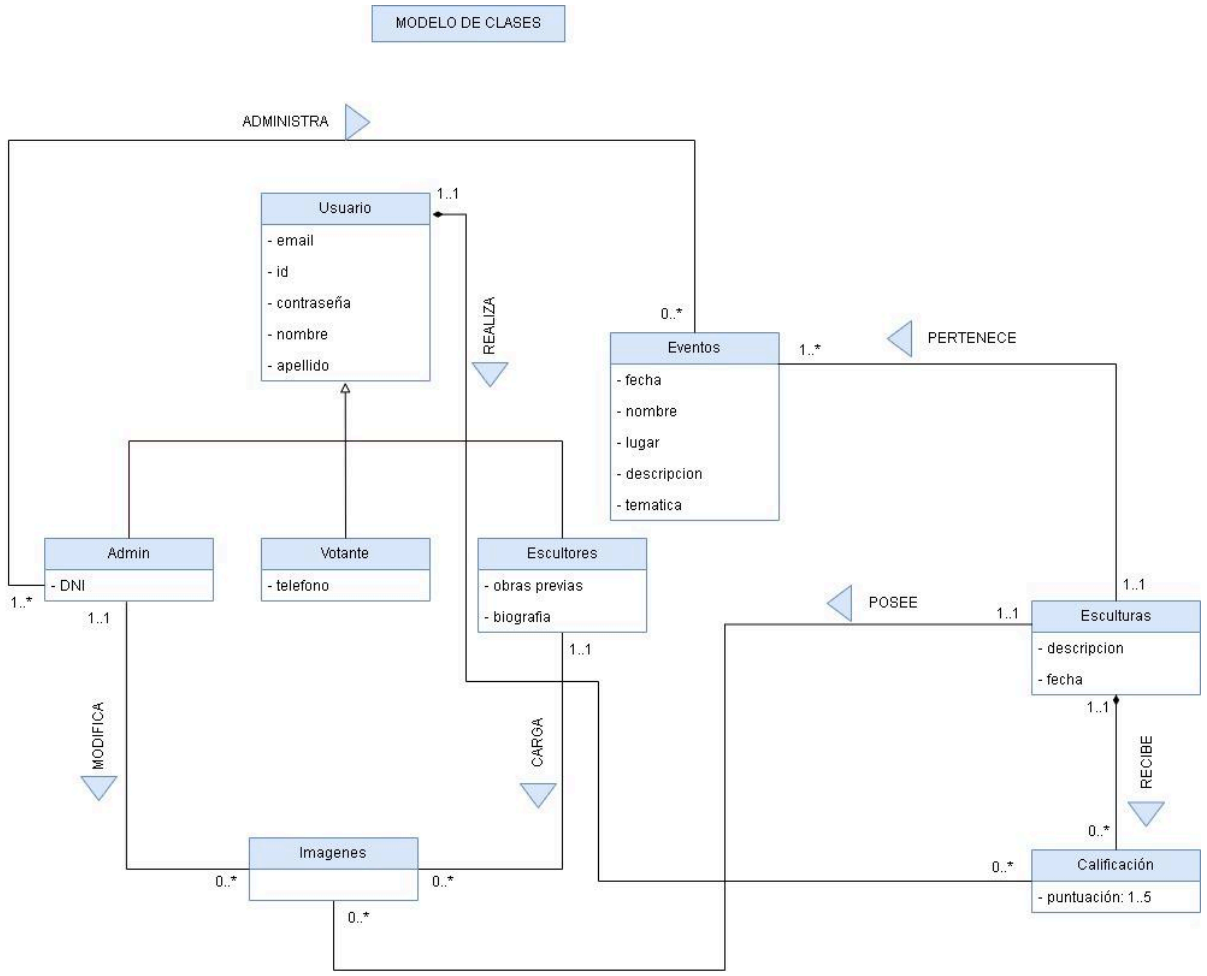
Finalmente, se construyó el **Diagrama Entidad-Relación (DER)**, para detallar de manera clara la estructura de la base de datos, estableciendo las entidades, atributos y relaciones necesarias para garantizar una gestión eficiente y la consistencia de los datos en el sistema.

Diagrama de CU



Aclaración: Esto es un esquema inicial, no se han implementado todos los CU identificados.
Link:
https://drive.google.com/file/d/1cPXULaryPRm3UTepvE9GI8up7fNYgpvO/view?usp=sharing

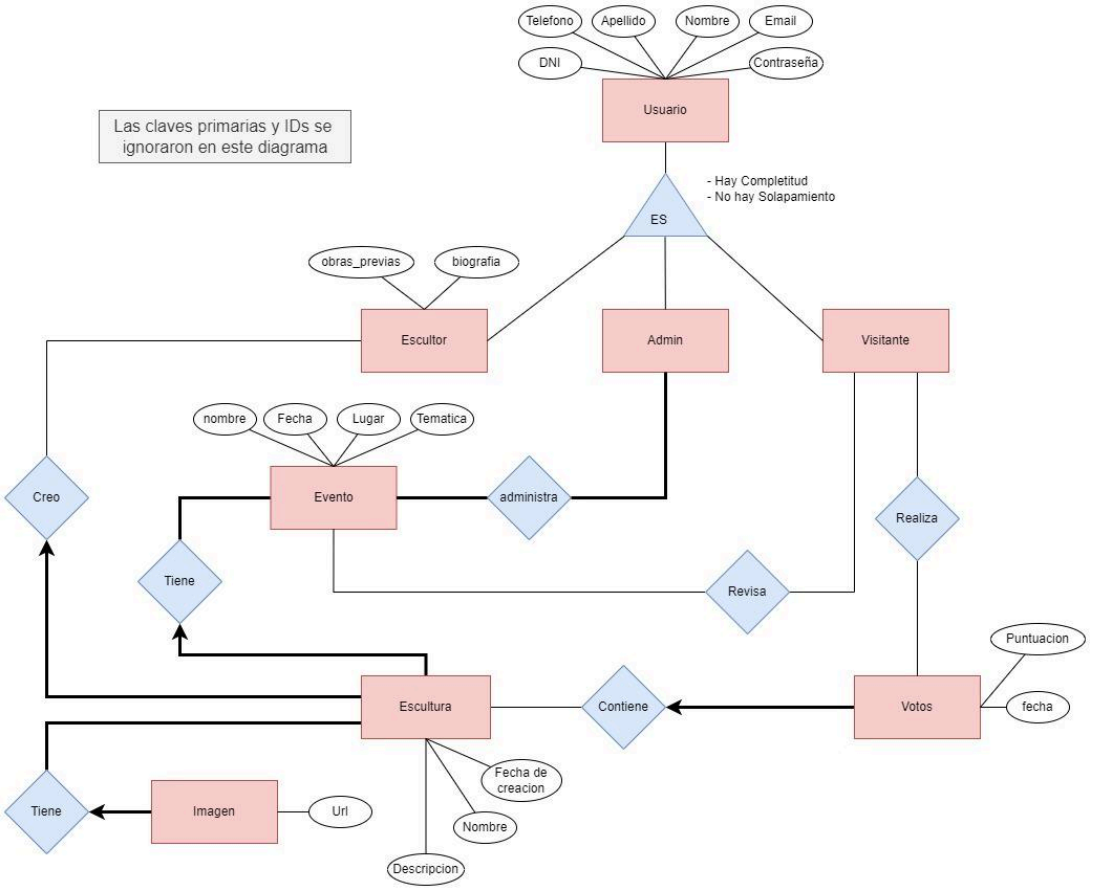
Diagrama de Clases



Link: https://drive.google.com/file/d/1cPXULaryPRm3UTepvE9GI8up7fNYgpvO/view?usp=sharing

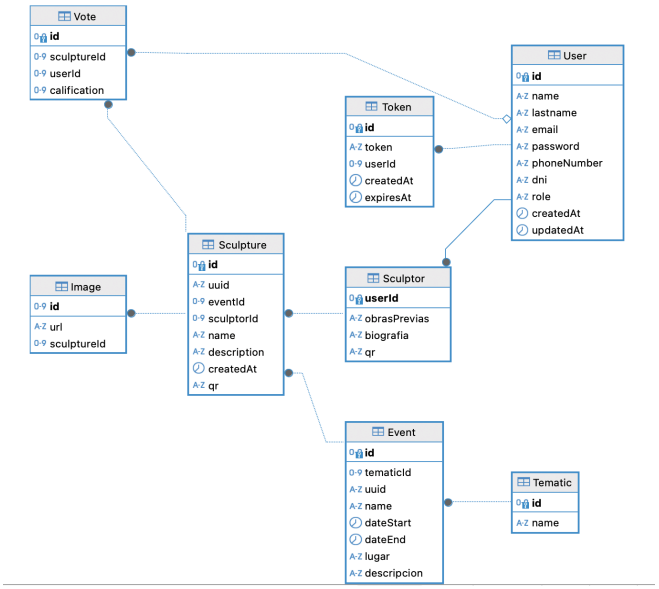
Diagrama Entidad-Relación

Diagrama de Análisis



Link:
https://drive.google.com/file/d/1cPXULaryPRm3UTepvE9Gl8up7fNYgpvO/view?usp=sharing

Diagrama real de la BD

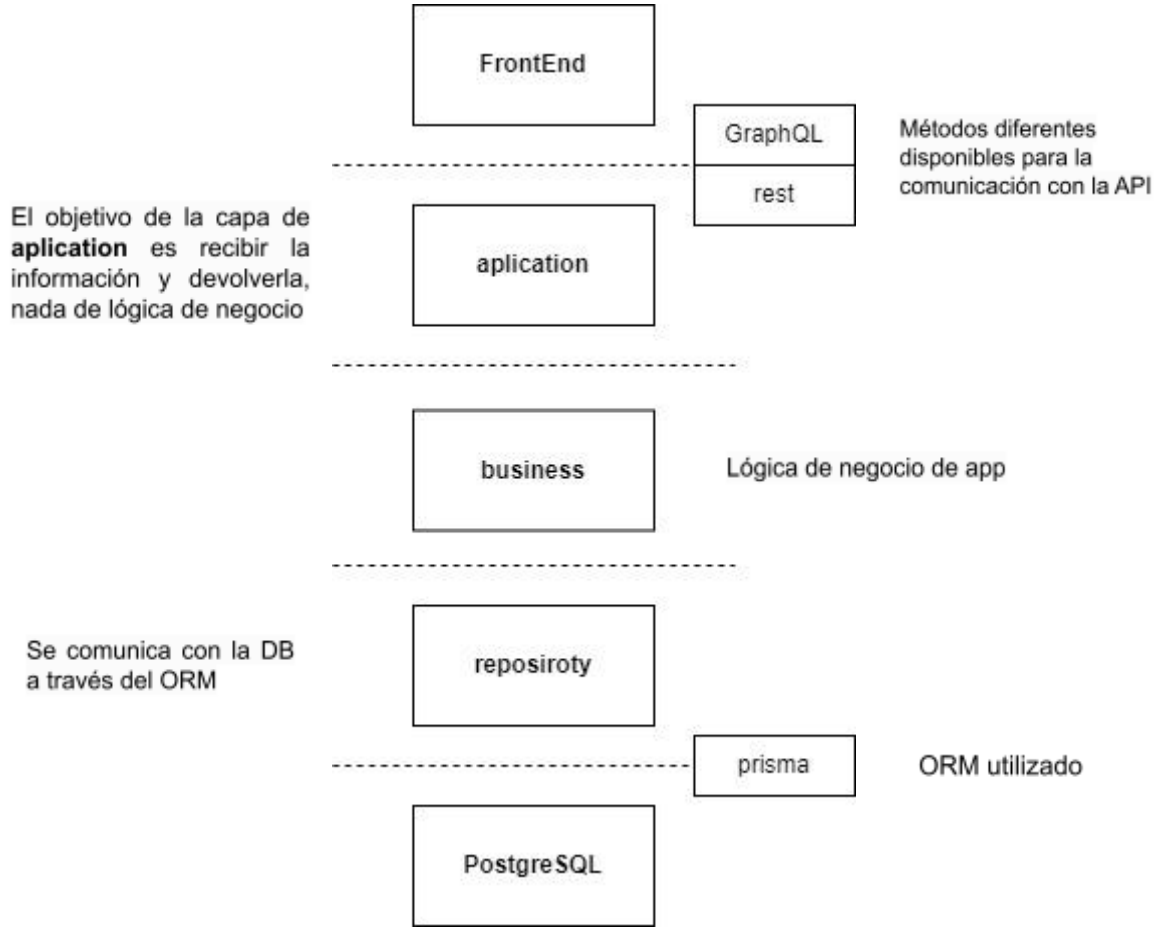


Decisiones de diseño

Diseño del Backend

Para el diseño del backend se utilizó una **arquitectura de capas** ya que esta permitió una organización limpia y organizada, separando responsabilidades, así como también la posibilidad de reemplazar alguna capa en caso de que fuera necesario, sin afectar a las demás.

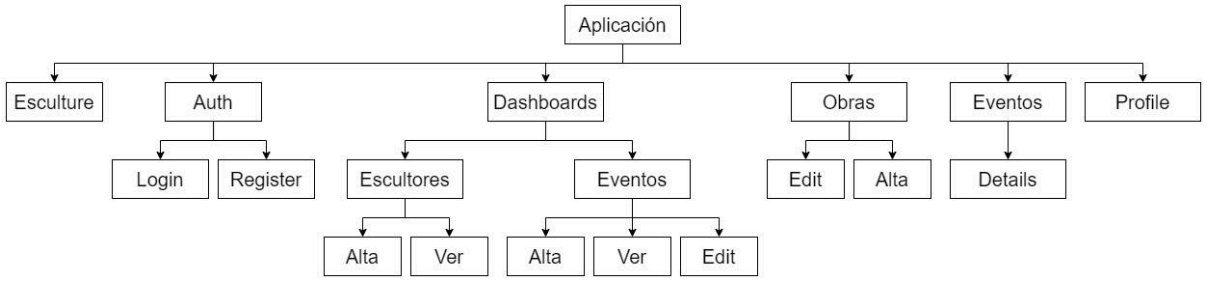
Modelo de capas del sistema



Diseño del Frontend

Para el diseño y desarrollo del frontend se utilizó una **arquitectura modular**, separando cada funcionalidad de la aplicación en módulos independientes que interactúan entre sí. Esto permitió el desarrollo progresivo e incremental de la aplicación, manteniendo un crecimiento controlado y facilitando la detección de errores por parte de los desarrolladores.

Módulos del Sistema



Herramientas Utilizadas

Github

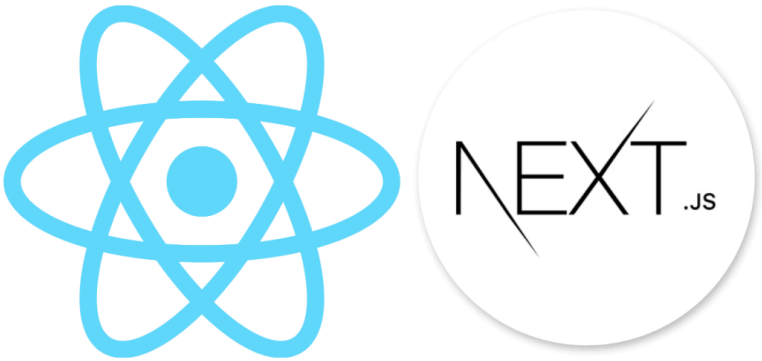
Para el control de versiones utilizamos GitHub

Esta plataforma nos facilitó la gestión y la colaboración de nuestro proyecto, la principal funcionalidad que presenta es que ofrece un entorno para trabajar con el sistema de control de versiones Git, lo que nos permitió visualizar los cambios que se realizan en el código, colaborar en tiempo real y gestionar los proyectos de manera eficiente.

Frontend

NextJS

NextJS es un framework de desarrollo web basado en React, que es una biblioteca de JavaScript diseñada para realizar interfaces de usuario.



Con esta herramienta, el servidor se encarga del proceso de renderizado de cada página, en lugar del navegador del usuario. Como resultado, después de realizar una solicitud, el usuario recibe mucho más rápido una página completamente renderizada.

NextJS utiliza el Renderizado del Lado del Servidor(SSR), donde se genera el HTML en cada solicitud al servidor, o la Generación de Sitios Estáticos(SSG), donde el HTML se genera durante la compilación.

Al poder utilizar SSR y SSG indistintamente, renderiza páginas al momento de su solicitud o al momento de su creación. Esta flexibilidad ofrece gran rendimiento en la carga de las páginas, al igual que para la obtención de datos.

La misma también nos permite crear rutas de API directamente en el directorio, simplificando así la integración backend.

Nos decidimos por usar esta herramienta por las siguientes razones:

- Menor tiempo de carga de la página web.
- Sitios web ajustables, según el tamaño de la pantalla del dispositivo.
- Las aplicaciones y sitios web con NextJS funcionan en cualquier dispositivo, permitiendo ofrecer los servicios a través de muchos canales.
- Las páginas web estáticas creadas con NextJS no tienen acceso directo a bases de datos o datos de usuarios, garantizando así la seguridad de los datos.

Backend

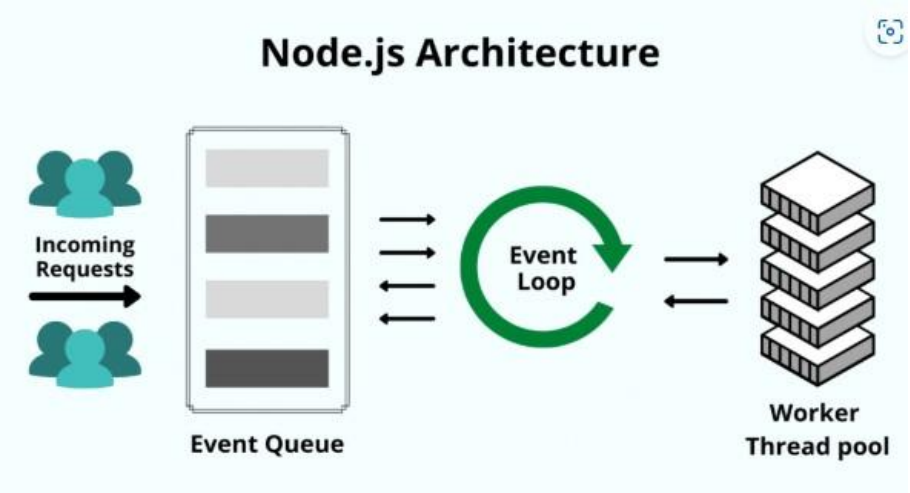
NodeJS

NodeJS es un entorno de ejecución de JavaScript del lado del servidor, construido sobre el motor V8 de Google Chrome.

NodeJS nos permite ejecutar JavaScript fuera del navegador, lo que ha revolucionado el desarrollo backend al hacer posible el uso de un solo lenguaje(JavaScript) para el desarrollo completo de aplicaciones web.

La misma utiliza la arquitectura Single Theaded Event Loop para manejar múltiples clientes al mismo tiempo. Para entender en qué se diferencia de otros tiempos de ejecución, tenemos que entender cómo se manejan los clientes concurrentes multihilo en lenguajes como Java.

En un Modelo de solicitud respuesta multihilo, varios clientes envían una solicitud y el servidor procesa cada una de ellas antes de devolver la respuesta. Sin embargo, se utilizan múltiples hilos para procesar las llamadas concurrentes. Estos hilos se definen en un pool de hilos, y cada vez que llega una petición se asigna un hilo individual para manejarla.



NodeJS funciona de forma diferente.

NodeJS mantiene un pool de hilos limitado para atender las peticiones.

Cada vez que llega una solicitud, Node.js la coloca en una cola.

Ahora, el bucle de eventos de un solo hilo el componente principal entra en escena. Este bucle de eventos espera las peticiones indefinidamente.

Cuando llega una solicitud, el bucle la recoge de la cola y comprueba si requiere una operación de entrada/salida (E/S) de bloqueo. Si no es así, procesa la solicitud y envía una respuesta.

Si la solicitud tiene una operación de bloqueo que realizar, el bucle de eventos asigna un hilo del pool de hilos internos para procesar la solicitud. Los hilos internos disponibles son limitados. Este grupo de hilos auxiliares se llama grupo de trabajadores.

El bucle de eventos rastrea las solicitudes que se bloquean y las coloca en la cola una vez que se procesa la tarea que se bloquea. Así es como mantiene su naturaleza no bloqueante.

Dado que Node.js utiliza menos hilos, utiliza menos recursos/memoria, lo que resulta en una ejecución más rápida de las tareas. Así que para nuestros propósitos, esta arquitectura de un solo

hilo es equivalente a la arquitectura multihilo. Cuando uno necesita procesar tareas con muchos datos, entonces tiene mucho más sentido utilizar lenguajes multihilo como Java. Pero para aplicaciones en tiempo real, Node.js es la opción obvia.

Decidimos utilizar esta tecnología por las siguientes razones:

- Es bastante fácil de empezar. Es una opción buena para principiantes en el desarrollo web. Con un montón de tutoriales y una gran comunidad.
- Proporciona una gran escalabilidad para las aplicaciones
- La ejecución de hilos sin bloqueo hace que sea aún más rápida y eficiente.
- Existe un amplio conjunto de paquetes de código abierto que pueden simplificar el trabajo.
- El soporte multiplataforma.
- Que tanto el frontend como el backend pueden ser gestionados con JavaScript como un único lenguaje.

NestJS

NestJS es uno de los frameworks de NodeJS de más rápido crecimiento para construir aplicaciones backend eficientes y escalables utilizando JavaScript y TypeScript.

Conclusión

En el desarrollo de este proyecto se desarrolló una aplicación web para la implementación de la solución solicitada, utilizando una arquitectura sólida basada en tecnologías modernas. Como se mencionó en el desarrollo del informe, el backend fue diseñado siguiendo un modelo de capas con **NodeJS** y **NestJS**, empleando **GraphQL** y **REST** como métodos de comunicación con el frontend, y una base de datos en **PostgreSQL** para la gestión de la información y persistencia de los datos. Por su parte, el frontend se desarrolló con un enfoque modular basado en componentes utilizando **NextJS**, lo que garantiza escalabilidad, reutilización de código y una experiencia de usuario eficiente.

Si bien no se lograron implementar todas las funcionalidades previstas en el alcance inicial del proyecto, se priorizaron las más relevantes, brindando las funcionalidades principales necesarias para alcanzar los objetivos de la aplicación. Esto deja una base sólida para la futura finalización de los requerimientos y a la vez, futuras mejoras e incorporación de características adicionales.