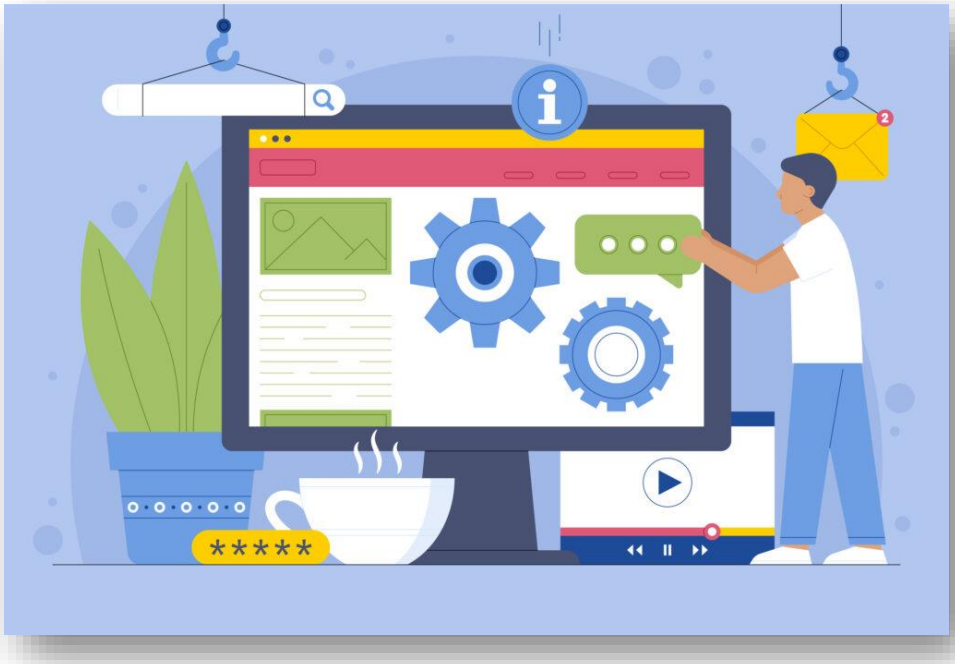
Instituto Politécnico Nacional
Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

upiita-ipn

- Integrantes:
  - Cisneros Rios Julio Cesar
  - Sánchez Cortés José Ángel

- Grupo: 2TM4
- Carrera: Ingeniería Telemática
- Materia: Ingeniería Web
- Maestro: Polanco Montelongo Francisco Antonio
- Arquitectura de sistema



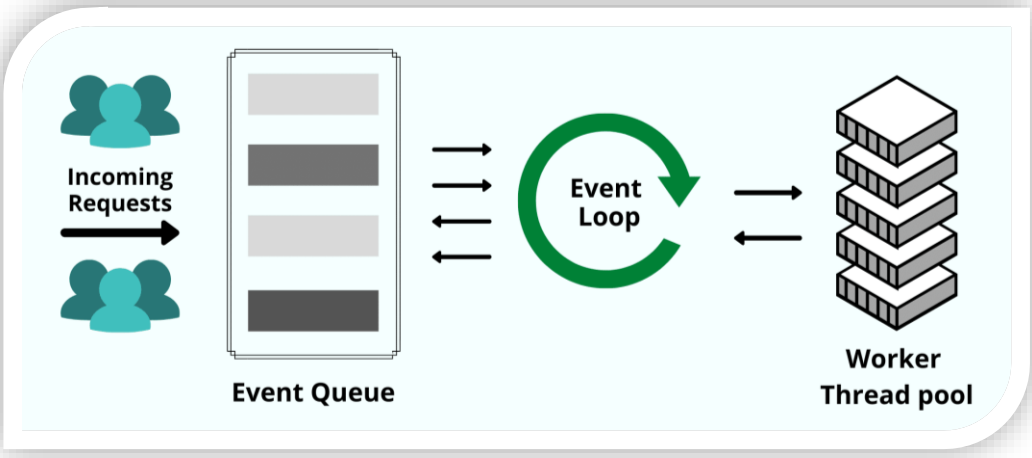
Arquitectura de sistema

En el proyecto utilizamos el framework Node.js. Node.js es, en esencia, un entorno de ejecución de JavaScript del lado del servidor. Se trata de un entorno de código abierto, multiplataforma y que se utiliza para ejecutar código JavaScript en el servidor.

La arquitectura de Node. js es de un solo hilo y basada en eventos. En un modelo multi-hilo, el servidor procesa cada una de las distintas solicitudes concurrentes de los clientes con múltiples hilos, asignando uno a cada petición, por lo que se consumen más recursos y memoria. Node.js funciona como un proceso único, por lo que puede manejar múltiples conexiones simultáneamente sin crear hilos adicionales, utilizando menos recursos y logrando mayor velocidad de ejecución.

Node.js utiliza la arquitectura “Single Threaded Event Loop” para manejar múltiples clientes al mismo tiempo. Utiliza el modelo de entrada y salida sin bloqueo (operaciones E/S no bloqueantes) y controlado por eventos. Con el Event Loop (bucle de eventos), se recogen las solicitudes y se genera un evento para cada petición, que se gestiona de manera asíncrona e independiente, sin interferencias entre eventos. Así, el hilo no se bloquea a la espera de respuesta y, mientras, pasa a procesar otros eventos.

Por ello, Node es recomendable para el desarrollo de aplicaciones escalables y de alta concurrencia, pues permite dar respuesta a muchas peticiones simultáneas de manera estable.



También se utilizó ReactJS. Este es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Creada por Facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI) llamadas componentes.

Es importante señalar que ReactJS no es un framework de JavaScript. Esto porque sólo es responsable de renderizar los componentes de la capa de vista de una aplicación. React es una alternativa a frameworks como Angular y Vue, que permiten crear funciones complejas.

React se ha vuelto muy popular en el desarrollo web debido a su enfoque en la construcción de interfaces de usuario eficientes y su flexibilidad para trabajar con otras tecnologías y bibliotecas. Algunas de las características clave de React son:

- Componentes: React organiza la interfaz de usuario en componentes reutilizables, que pueden ser de distinto tipo y tamaño.
- Estados y Desarrollo Declarativo: React permite al desarrollador describir cómo debe verse la interfaz dependiendo del estado de la aplicación. De este modo, cuando un componente cambia de estado, React hace un seguimiento de esos cambios y actualiza la interfaz de usuario para que aparezca acorde al nuevo estado.
- Virtual DOM: React aplica las modificaciones primero en una representación virtual del DOM, de forma que posteriormente solo se apliquen al DOM real los cambios necesarios. Así, se reduce la carga en el navegador y se optimiza el rendimiento.
- JSX: React utiliza JSX (JavaScript XML), que permite escribir código que se asemeja al marcado HTML. Gracias a ello, la creación de componentes se simplifica y mejora su legibilidad.
- Unidireccionalidad de datos: esto significa que los datos solo fluyen en una dirección a través de la aplicación. Esto facilita el seguimiento y el mantenimiento, con lo que se consigue un código más predecible y fácil de depurar.

Para validar datos utilizamos Zod, el cual es una biblioteca de declaración y validación de esquemas que prioriza TypeScript y está diseñada para proporcionar una forma segura de validar objetos JavaScript. Ayuda a los desarrolladores a definir la forma de los datos esperados y a generar automáticamente tipos TypeScript a partir de estos esquemas, lo que garantiza la validación tanto en tiempo de compilación como en tiempo de ejecución. Esta doble capacidad convierte a Zod en una herramienta invaluable para los desarrolladores TypeScript que buscan reforzar la integridad de los datos en sus aplicaciones.

Ventajas de usar Zod:

- Seguridad de tipos : Zod se integra perfectamente con TypeScript, lo que le permite definir esquemas fuertemente tipados. Esto garantiza que sus datos se adhieran a la estructura esperada, lo que reduce el riesgo de errores en tiempo de ejecución.
- Validación integral : Zod admite una amplia gama de tipos de datos y reglas de validación listas para usar, desde primitivos simples como cadenas y números hasta objetos anidados y matrices complejos.
- Extensibilidad : Zod le permite crear una lógica de validación personalizada, lo que le permite adaptar el proceso de validación a sus necesidades específicas.
- Seguridad : al validar los datos entrantes, Zod ayuda a evitar que se procesen cargas maliciosas. Esto es fundamental para proteger su aplicación de amenazas de seguridad comunes, como la inyección SQL y otras formas de ataques basados en datos.
- Integración : Zod se puede integrar fácilmente con varios marcos y bibliotecas, incluidos React, Express y más, lo que lo convierte en una opción versátil para diferentes tipos de proyectos.

Para autentificar usuarios se utilizó JWT. JSON Web Token, comúnmente conocido como JWT, es un estándar abierto (RFC 7519) para transmitir información de forma segura entre partes como un objeto JSON. El token está firmado digitalmente, lo que garantiza su autenticidad e integridad. Los JWT se utilizan principalmente para autenticar usuarios, autorizar el acceso a determinados recursos e intercambiar información de forma segura.

Cuando un usuario inicia sesión o intenta acceder a un recurso protegido, el servidor genera un token JWT después de una autenticación exitosa. Luego, el cliente almacena este token, generalmente en el almacenamiento local o en una cookie. Para cada solicitud posterior que requiera autenticación, el cliente envía el token JWT en los encabezados de la solicitud. El servidor valida el token verificando la firma y decodificando la carga útil para garantizar la autenticidad y la autorización del usuario.

Los reclamos en un JWT están codificados como un objeto JSON y normalmente están firmados digitalmente con un Código de autenticación de mensajes (MAC). El escenario más común para utilizar un JWT es la autenticación. Cuando el usuario inicia sesión, cada solicitud posterior incluye el JWT, que permite al usuario acceder a los servicios permitidos por ese token.