Front-End

1. Vue

Vue.js es un framework progresivo de JavaScript que permite construir interfaces web de forma declarativa y reactiva. Su principal ventaja es su curva de aprendizaje amigable, ya que puedes integrarlo en una sola página HTML o usarlo en proyectos más complejos con herramientas como Vue CLI. Vue usa un sistema de componentes que divide la UI en piezas reutilizables, lo cual mejora la organización y el mantenimiento del código.

2. Vuetify

Vuetify es una librería de componentes basada en Material Design para Vue.js. Proporciona componentes visuales listos como botones, formularios, barras de navegación, tarjetas, tablas, entre muchos otros. Lo más destacable es que es muy personalizable y responsivo, permitiendo crear interfaces modernas con menos esfuerzo en diseño, enfocándote más en la lógica que en la apariencia.

3. Axios

Axios es una librería HTTP basada en promesas que permite hacer solicitudes al servidor desde el navegador. Es ampliamente usada en Vue para consumir APIs REST. Puedes hacer solicitudes `GET`, `POST`, `PUT`, `DELETE`, entre otras. Además, Axios permite configurar cabeceras (headers), interceptores para manejar errores globalmente, y trabajar con `async/await` para un código más limpio.

4. i18n en Vuetify

i18n significa "internacionalización" (por sus 18 letras entre la "i" y la "n"). Se refiere a la capacidad de traducir una aplicación a múltiples idiomas. En combinación con Vuetify, se puede usar `vue-i18n` para traducir textos dinámicamente, mostrar fechas o monedas locales, e incluso cambiar el idioma en tiempo real según el navegador del usuario o una configuración personalizada.

5. Pinia

Pinia es la nueva solución oficial para el manejo de estado global en Vue 3. Reemplaza a Vuex por ser más simple, ligera y compatible con TypeScript. Permite definir "stores" que contienen el estado de tu aplicación, junto con funciones para mutarlo y getters computados. Se integra fácilmente con Composition API, es modular y soporta persistencia del estado con plugins.

6. Options API en Vue

Es la forma clásica de escribir componentes en Vue, donde la lógica se organiza por opciones (`data`, `methods`, `computed`, etc.) en lugar de funciones. Aunque Vue 3 introdujo la Composition API, la Options API sigue siendo válida y más intuitiva para principiantes, separando claramente los datos, las funciones y las propiedades computadas.

 Back-End

1. Métodos HTTP

Son los verbos que definen la acción que el cliente quiere ejecutar sobre un recurso. Los más comunes son:

`GET`: recuperar información sin modificar nada.

`POST`: enviar nuevos datos al servidor.

`PUT`: reemplazar un recurso existente.

`PATCH`: modificar parcialmente un recurso.

`DELETE`: eliminar un recurso.

Cada uno tiene su propósito y es clave usarlos correctamente al diseñar una API REST.

2. Clases

Una clase es una plantilla que permite crear objetos con atributos (propiedades) y comportamientos (métodos). En backend orientado a objetos (como en Java o TypeScript), las clases se usan para representar entidades del dominio (como Usuario, Producto, etc.), encapsulando su lógica y sus datos.

3. Interfaces

Una interfaz define qué estructura debe seguir un objeto o clase, sin preocuparse por cómo lo hace. Es decir, establece un contrato que garantiza que un objeto tenga ciertas propiedades o métodos. Muy útil para mantener el código claro, especialmente cuando se trabaja con TypeScript o lenguajes estáticos.

4. Mappers

Los mappers son capas intermedias que convierten objetos de un tipo a otro. Por ejemplo, transformar una entidad de base de datos a un DTO (objeto que viaja a través de la red) o viceversa. Esto ayuda a separar responsabilidades, evitando acoplar directamente la base de datos con la lógica de negocio o la API.

5. Autenticación con JWT

JWT (JSON Web Token) es un método seguro de autenticación que no depende de sesiones almacenadas en el servidor. En lugar de eso, el servidor genera un token firmado digitalmente y lo envía al cliente. El cliente lo guarda (normalmente en localStorage) y lo usa para cada solicitud. El servidor verifica la validez del token en cada petición protegida, permitiendo o negando el acceso.

6. CORS

CORS (Cross-Origin Resource Sharing) es una política de seguridad implementada por los navegadores para evitar que una página web haga peticiones a un dominio diferente sin autorización. Si el servidor no permite solicitudes externas, el navegador bloqueará la respuesta. Configurar correctamente CORS es fundamental para que el frontend y el backend puedan comunicarse en desarrollo y producción.

7. Clean Architecture

Propuesta por Robert C. Martin, busca mantener el código limpio y desacoplado. Divide el sistema en capas (Entidad, Caso de Uso, Interfaz, Infraestructura), cada una con responsabilidades específicas. Permite probar fácilmente el dominio sin preocuparte por detalles externos como bases de datos o interfaces.

8. Arquitectura Hexagonal

También conocida como “Ports and Adapters”. Propone que el núcleo de la aplicación esté aislado y pueda conectarse con el exterior (bases de datos, API, interfaces gráficas) mediante puertos y adaptadores. Esta estructura permite sustituir fácilmente componentes externos sin afectar la lógica principal.

9. Patrones de diseño (Criteria, Singleton)

Criteria: permite construir consultas dinámicas y reutilizables para filtrar o buscar datos según criterios personalizados, Singleton: patrón que garantiza que una clase tenga una única instancia en toda la aplicación. Útil para cosas como controladores de logs, configuración o conexiones de base de datos.

GIT Y GITHUB

1. Comandos básicos de GIT

Git es el sistema de control de versiones más usado. Algunos comandos esenciales:

- `git init`: inicializa un repositorio local.
- `git clone <url>`: clona un repositorio remoto.
- `git add <archivo>`: agrega archivos al área de preparación.
- `git commit -m "mensaje"`: guarda los cambios localmente.
- `git push`: sube los commits al repositorio remoto.
- `git pull`: actualiza tu código con cambios remotos.

2. ¿Qué es un PR en GitHub?

Un Pull Request (PR) es una solicitud para que se integren los cambios de una rama (por lo general de un fork o feature branch) a la rama principal de un repositorio. Es el paso clave en el trabajo colaborativo, ya que permite revisión de código, comentarios y pruebas antes de aceptar los cambios.

3. ¿Qué es un Fork en GitHub?

Un fork es una copia de un repositorio en tu cuenta personal. Es muy usado en proyectos open source, ya que permite hacer cambios sin tocar el repositorio original. Luego, puedes hacer un PR desde tu fork para proponer mejoras o correcciones.

Es una solicitud de aceptación

Mark down es un lenguaje en Git hub Project quien se utilizar en las tarjetas

-Git hub Project

. Llevar la gestión del proyecto

-tareas

METODOLOGIA AGIL SCRUM

-Metodología Agil

-Permita la colaboración de diferentes personas de forma suficientes

Product owner

Scrum master

Team developer

Sprint no debe perdurar en más de 3 semanas