FRONT-END

Es la parte visual de una aplicación o página web: lo que el usuario ve y con lo que interactúa.

Vue

Un framework de JavaScript para crear interfaces web dinámicas.

- **Ventaja**: Fácil de aprender, flexible y muy usado.
- **Ejemplo**: Mostrar en tiempo real una lista de productos sin recargar la página.

Vuetify

Una librería de componentes visuales para Vue, con un estilo moderno basado en Material Design.

- **Ejemplo**: Botones, menús, tablas y formularios listos para usar.

Axios

Una herramienta para que tu aplicación se comunique con un servidor y obtenga o envíe datos.

- **Ejemplo**: Consultar precios de criptomonedas desde una API.

i18N en Vuetify

Significa “internationalization” (internacionalización). Sirve para mostrar tu aplicación en varios idiomas.

- **Ejemplo**: Cambiar de español a inglés con un clic.

Pinia

Una forma de manejar datos compartidos entre varios componentes en Vue.

- **Ejemplo**: Guardar el usuario que inició sesión y mostrarlo en distintas partes de la app.

Options API en Vue

Una manera tradicional de organizar el código en Vue usando secciones como data, methods y computed.

- **Ejemplo**: Tener todo el código de un componente estructurado en un solo bloque.

BACKEND

Es la parte “invisible” que maneja la lógica, datos y conexión con la base de datos.

Métodos HTTP

Formas en que el navegador y el servidor se comunican:

- **GET**: obtener datos.
- **POST**: enviar datos.
- **PUT**: actualizar datos.
- **DELETE**: borrar datos.

Clases

Estructuras para crear objetos con propiedades y funciones.

- **Ejemplo**: Clase Usuario con nombre, correo y método para iniciar sesión.

Interfaces

Contratos que definen qué propiedades y métodos debe tener un objeto (comunes en TypeScript y Java).

- **Ejemplo**: “Todo usuario debe tener un nombre y un correo”.

Mappers

Transformadores de datos: convierten información de un formato a otro.

- **Ejemplo**: Pasar datos de una base de datos a un formato que entienda tu app.

Autenticación con JWT

Forma segura de verificar quién eres usando un “token” (llave digital).

- **Ejemplo**: Inicias sesión y recibes un token que te da acceso a las funciones privadas.

CORS

Permiso que da un servidor para que otros sitios web puedan pedirle información.

- **Ejemplo**: Evita que cualquier web extraña acceda a tus datos sin autorización.

Clean Architecture

Organizar el código para que sea fácil de entender, probar y mantener.

- **Ejemplo**: Separar la lógica de negocio de los detalles técnicos.

Arquitectura Hexagonal

Un estilo para que la aplicación sea independiente de herramientas externas.

- **Ejemplo**: Que tu app funcione igual, aunque cambies de base de datos.

Patrones de diseño

- **Criteria**: filtrar datos según condiciones.
- **Singleton**: garantizar que solo exista una instancia de algo (por ejemplo, una conexión a base de datos).

OTRAS HERRAMIENTAS Y CONCEPTOS

Comandos básicos de Git

- git init: iniciar un proyecto.
- git add: preparar cambios.
- git commit: guardar cambios.
- git push: subir cambios a GitHub.

Pull Request (PR) en GitHub

Propuesta de cambios para que alguien los revise antes de unirlos al proyecto.

Fork en GitHub

Copia de un proyecto para trabajar de forma independiente.

Usar API REST desde Postman

Postman permite probar y enviar peticiones a APIs sin necesidad de programar.

Markdown

Lenguaje sencillo para dar formato a texto (títulos, listas, enlaces).

GitHub Projects

Tablero para organizar tareas de un proyecto.

SCRUM

Metodología ágil para trabajar en equipo con entregas rápidas y revisiones constantes.

RECOMENDACIONES DE HERRAMIENTAS

- **Postman** → probar APIs.
- **DBeaver** → manejar bases de datos.
- **Spring Tool Suite** → programar en Java.
- **Visual Studio Code** → editor de código versátil.
- **Brave** → navegador rápido y privado.
- **GitKraken** → interfaz visual para manejar Git.