Informe Trabajo Final:

POKEDEX

Desarrollo Frontend, uso de API y librería de Pinia

Materia: Programación 3
Año: 2025
Alumna: Josefina Agüero Cerutti
Rol: Fullstack

Introducción

Este proyecto es una aplicación web que permite a los usuarios seleccionar Pokemones para formar equipos, el cuál será guardado en una base de datos.

Partes del Proyecto

El proyecto se divide principalmente en tres partes: Frontend, Backend y Base de Datos.

Frontend: Desarrollado en Vue.js 3, incluye la interfaz de usuario y la lógica de consumo de una API (PokeAPI).

Backend: Un servidor Node.js que utiliza el framework 'Express' que procesa la interacción del usuario con la base de datos.

Base de Datos: Utiliza el motor de MySQL, almacenando los equipos.

Herramientas y el flujo de datos

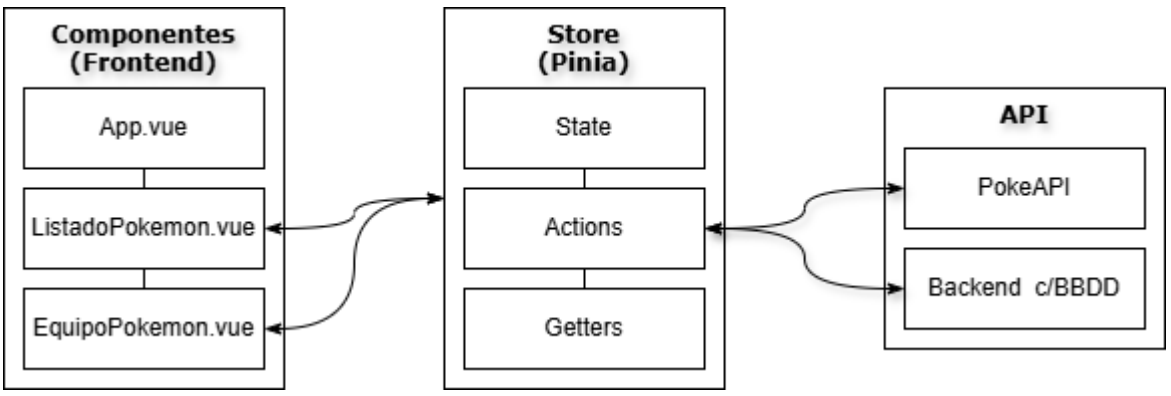
PokeAPI: La fuente externa de datos para obtener nombres e imágenes de Pokémon.

Axios: Cliente HTTP utilizado tanto para obtener datos de la API externa (GET) como para enviar datos al servidor local (POST).

Pinia: Sistema de gestión de estado global que permite que la lista de Pokémon elegidos esté disponible en cualquier parte de la aplicación sin perderse al navegar.

Diseño y Estructura

La estructura de la aplicación está basada en componentes. Cada parte de la interfaz tiene su propia función y se comunica con la Base de Datos.



Creación y ejecución del proyecto

En este primer paso nos centramos en la Base de Datos:

Necesitamos iniciar un servidor, para eso utilizamos **XAMPP** en el cual levantamos **MySQL**.

Accedemos a la Base de Datos por medio del navegador utilizando la herramienta **PhpMyAdmin** y procedemos a crear la Base de Datos y el modelo de datos (estructura).

Consultas ejecutadas en PHPMyAdmin:

```
CREATE DATABASE pokemon;
USE pokemon;

CREATE TABLE equipos (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nombre VARCHAR(100) NOT NULL,
  pokemones JSON NOT NULL,
  fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

La conexión a la Base de Datos se configura dentro del Backend con las credenciales predeterminadas:

```
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'pokemon'
});
```

Componentes

App.vue: Gestiona qué componente se muestra en pantalla, como el Listado de Pokémon o el Equipo.

ListadoPokemon.vue: Se encarga de la interfaz de búsqueda y visualización. Muestra los datos.

EquipoPokemon.vue: Se encarga de mostrar la selección del usuario y gestionar el formulario de envío al backend.

Pinia Store: El almacén de datos global. (Persistencia de datos)

Conexión API y Base de Datos

A través de funciones asincrónicas aseguramos que la aplicación no se congele mientras espera respuestas externas.

Llamada a la API externa: Utilizamos Axios para realizar una solicitud GET a la URL de PokeAPI, como es función asincrónica, esto permite que la aplicación siga funcionando mientras se descargan los datos de los Pokémon. Una vez que se reciben, los datos aparecen en pantalla.

Manejo de Solicitud en MySQL: Una vez que el usuario guarda el equipo de Pókemon, se crea un objeto en JavaScript que contiene el nombre del equipo y el vector de los Pokémon de Pinia. Mediante axios.post, se envían estos datos al servidor Node.js. Una vez que se reciben los datos, el backend utiliza JSON.stringify(), lo que transforma la lista en una cadena de texto con formato JSON, compatible con la columna de la tabla. Una vez realizado todo se devuelve una respuesta exitosa del servidor y el frontend muestra un mensaje de confirmación al usuario.

Despliegue: Para que el sistema funcione se deben garantizar tres estados de ejecución: El servicio de **MySQL** iniciado, El backend con el servidor **Node.js** activo que actua como API interna para gestionar el acceso a la base de datos y el frontend con **Vue.js** para la interfaz de la aplicación.

Conclusión

El desarrollo de este proyecto me ha permitido integrar las partes que inicialmente estaban aisladas en un ecosistema de manera práctica.

El mayor desafío fue entender la comunicación asincrónica; comprender la espera por parte del frontend a la respuesta del servidor sin detenerse y cómo el backend solicita los datos a la base de datos.

A través de esta experiencia, se logró comprender la importancia de tener todo organizado en módulos, donde el **Frontend (Vue.js)** se encarga de la interacción, la importancia de **Store (Pinia)** para evitar que la información se pierda al navegar, el **Backend (Node.js)** gestiona la lógica y seguridad, y la **Base de Datos (MySQL)** garantiza la persistencia de la información.

Además, este trabajo me permitió aprender la importancia de la documentación técnica mediante la creación de un archivo `README.md`. Descubrí el lenguaje **Markdown** y utilicé herramientas online para su edición, lo que me facilitó estructurar las instrucciones de instalación y despliegue de forma clara. Entender que un buen código debe ir acompañado de una guía que permita a otros ejecutarlo fue uno de los aprendizajes más valiosos de esta etapa final.

En resumen, este trabajo me permitió ver cómo funciona una aplicación **Fullstack** y cómo el código que escribimos impacta directamente en una base de datos real.