Presentación Final - 2da Evaluación Tecnologías Web
Nombre del Estudiante: Jose Antonio Favian Yujra Yana

---

1. Introducción

Este proyecto es un portal interactivo para consultar personajes de **Star Wars** utilizando la API de **swapi.tech**. La aplicación permite a los usuarios ver la lista de personajes, navegar entre páginas y consultar detalles como nombre, altura, género, y fecha de nacimiento. La aplicación maneja la carga de datos, la paginación, y los posibles errores al consumir la API.

---

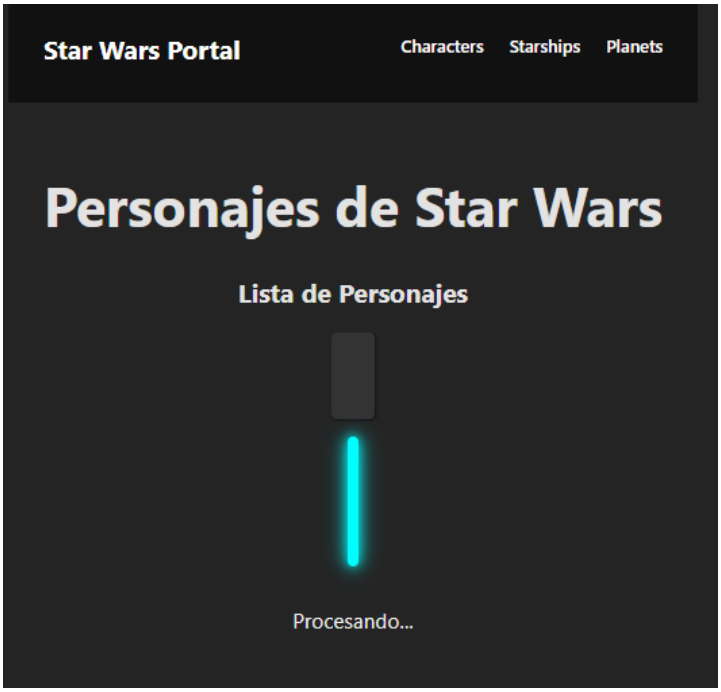
2. Demostración Funcional

2.1 Flujo de la Aplicación

1. **Pantalla de carga**: Cuando el portal se está cargando, se muestra un **indicador de carga** para que el usuario sepa que los datos están siendo procesados.
2. **Lista de personajes**: La aplicación carga la lista de personajes desde la API y permite al usuario navegar entre diferentes páginas de resultados.
3. **Manejo de errores**: Si hay un problema al cargar los datos, la aplicación muestra un mensaje de error al usuario.

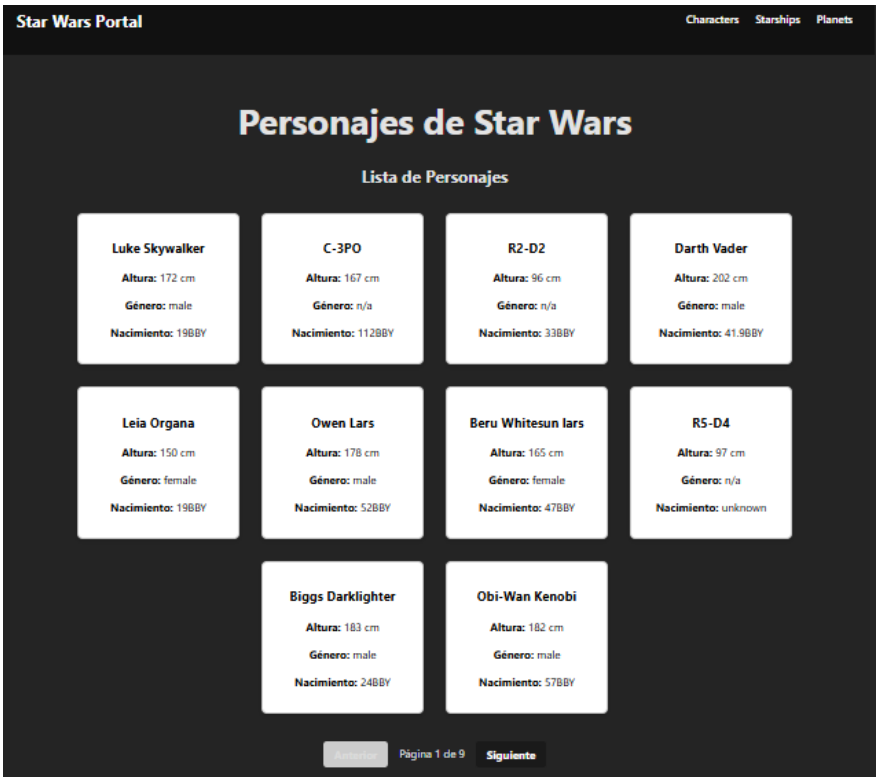
2.2 Capturas de Pantalla

Carga:

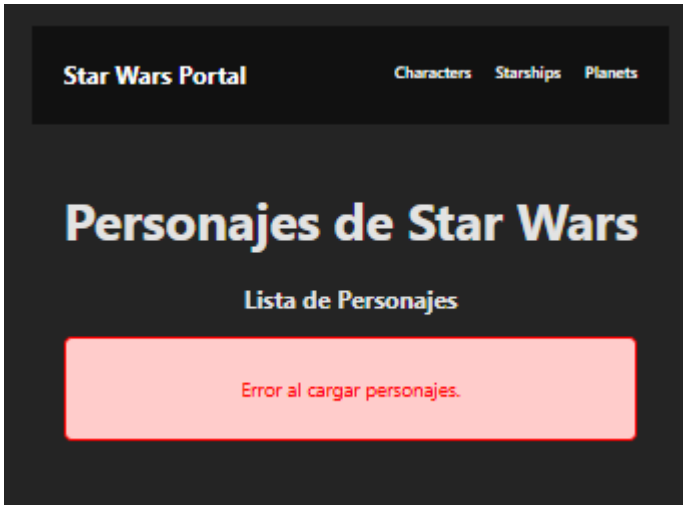


Animación de Sable de Luz

Paginación y despliegue de información de personajes:



Mensaje de Error:



---

3. Explicación Técnica

3.1 Tecnologías Utilizadas

- **Vue.js**: Framework de frontend que se utiliza para estructurar la interfaz de usuario y manejar los datos.
- **Axios**: Librería para realizar solicitudes HTTP y consumir la API de **swapi.tech**.
- **GitHub**: Herramienta de control de versiones y repositorio remoto para almacenar el código.
- CSS puro,
- Vue Router

3.2 Estructura del Proyecto

El proyecto está organizado en varias carpetas y archivos principales que facilitan la modularidad y el mantenimiento.

- **src/: Contiene la mayor parte del código fuente del proyecto.**
    - **components/: Carpeta que contiene los componentes reutilizables de la aplicación:**
        - **ErrorMessage.vue: Muestra un mensaje de error cuando ocurre un problema al cargar los datos.**
        - **LoadingIndicator.vue: Muestra un indicador de carga mientras se obtienen los datos de la API.**

        - **Navbar.vue: Componente de la barra de navegación que permite al usuario navegar entre las diferentes secciones de la aplicación.**
        - **PeopleList.vue: Componente donde se gestionan la lista de personajes y la paginación. Aquí se hace la llamada a la API para obtener los personajes y se muestra la información de cada uno.**
        - **PersonCard.vue: Componente que muestra los detalles de un personaje individual. Se usa dentro de PeopleList.vue.**
        - **PlanetCard.vue: Similar a PersonCard.vue, pero para mostrar detalles de planetas.**
        - **PlanetsList.vue: Muestra una lista de planetas obtenidos de la API.**
        - **StarshipCard.vue: Componente que muestra los detalles de una nave espacial.**
        - **StarshipsList.vue: Muestra una lista de naves espaciales obtenidas de la API.**
    - **views/: Contiene las vistas principales que representan las diferentes páginas de la aplicación.**
        - **Home.vue: Página principal de la aplicación, que puede incluir una bienvenida o resumen de las funcionalidades.**
    - **App.vue: Componente raíz de la aplicación, que contiene la estructura general y el enrutamiento.**
    - **main.js: Archivo principal de entrada donde se inicializa la aplicación y se monta.**
    - **router.js: Archivo donde se configuran las rutas de la aplicación, permitiendo la navegación entre las vistas (por ejemplo, personajes, planetas, naves).**
    - **style.css: Archivo de estilos globales para la aplicación.**

3.3 Consumo de la API

La API de **swapi.tech** proporciona datos sobre los personajes de Star Wars. Usamos **Axios** para hacer solicitudes GET y obtener los datos.

Código de ejemplo:

javascript

Copiar

```
const fetchPeople = async () => {
 loading.value = true;
 try {
  const res = await
axios.get(`https://www.swapi.tech/api/people?page=${currentPage.value}`);
  totalPages.value = res.data.total_pages;
  const detailedPeople = await Promise.all(
   res.data.results.map(async (item) => {
    const detail = await axios.get(item.url);
    return detail.data.result.properties;
   })
  );
  people.value = detailedPeople;
  error.value = null;
 } catch (err) {
  error.value = 'Error al cargar personajes.';
  console.error(err);
 } finally {
  loading.value = false;
 }
};
```

Este fragmento de código muestra cómo cargamos los datos de personajes y manejamos la paginación.

3.4 Explicación de la Paginación

La paginación en la aplicación se maneja con el estado **currentPage**, que se actualiza al hacer clic en los botones "Anterior" y "Siguiente". Dependiendo de la página actual, se hace una nueva solicitud a la API para obtener los personajes correspondientes.

3.5 Manejo de Errores

Si ocurre un error durante la solicitud a la API, el sistema muestra un mensaje de error usando el componente **ErrorMessage.vue**. Este manejo de errores asegura que el usuario tenga una experiencia de usuario fluida, incluso cuando los datos no pueden ser cargados.

---

4. Conclusión

La aplicación es una herramienta sencilla pero funcional que permite a los usuarios interactuar con la base de datos de personajes de Star Wars. El proyecto fue desarrollado utilizando **Vue.js** para el frontend y **Axios** para consumir la API. La paginación y manejo de errores están implementados para asegurar una experiencia de usuario sólida y sin interrupciones.