Objetivo

Desarrollar una aplicación web que implemente operaciones CRUD (Create, Read, Update, Delete) utilizando la API pública de Pokémon (https://pokeapi.co) o la API pública de Rick and Morty (https://rickandmortyapi.com) y almacenamiento local.

Descripción general

Los desarrolladores deberán crear una aplicación que permita gestionar una colección de personajes de Pokémon o personajes de Rick and Morty, combinando datos de la API con funcionalidades de gestión local, demostrando el conocimiento de lit-element, web components y javascript.

La aplicación deberá contar con una estructura de carpetas similar a la vista en las practicas, como:

```
|- src
|   | - components
|      | - mi-componentes
|         |- mi-componente.js
|         |- mi-componenteStyle.js
| - index.html
```

Funcionalidades Requeridas

1. READ - Listado de Personajes

- Consumir la API para obtener una lista de Personajes.
- Mostrar los datos en una tabla con las siguientes columnas:

Para la API de Pokemon:

- ID
- Nombre
- Tipo(s)
- Peso
- Altura
- Botones de acción (Editar / Eliminar)

Para la API de Rick and Morty:

- ID
- Nombre
- Estado
- Tipo

- Género
- Botones de acción (Editar/Eliminar)

2. CREATE - Agregar Personaje

Como las APIs mencionadas son solo lectura, simularemos la creación a través de agregar personajes por medio de un formulario:

Formulario para crear un "Personaje personalizado" con campos:

Para la API de Pokemon:

- Nombre
- Tipo(s) (selección múltiple)
- Peso
- Altura

Para la API de Rick and Morty:

- Nombre
- Estado (Booleano)
- Tipo
- Género

Nota: recuerda que puedes agregar validaciones a los campos.

3. UPDATE - Edición

Para personajes personalizados:

- Editar todos los campos del formulario de creación.
- Validación de datos obligatorios.

4. DELETE - Eliminación

- Eliminar personajes personalizados de la colección local.
- Confirmación antes de eliminar.
- No afectar los datos de la API original.

Especificaciones Técnicas

Tecnologías Sugeridas

- Lit-element, web components (https://www.webcomponents.org).
- Almacenamiento:

    - LocalStorage (https://www.freecodecamp.org/espanol/news/como-usar-localstorage-en-javascript/).
    - Escritura sobre archivos (https://www.browserstack.com/guide/write-files-using-fs-writefilesync-in-node-js).
- Estilo: CSS.
- HTTP Client:
    - Fetch API (https://www.freecodecamp.org/espanol/news/javascript-fetch-api-para-principiantes/).
    - Axios (https://www.arsys.es/blog/axios).
    - O usar herramientas similares.

Criterios de Evaluación

Funcionalidad

- Correcta integración con API.
- CRUD completo funcionando.
- Validación de formularios.

Código

- Estructura y organización del proyecto.
- Buenas prácticas de programación.

UI/UX

- Uso de webcomponents.
- Manejo de estados vacíos.

Entregables

1. Código fuente en repositorio Git.
2. README con instrucciones de instalación y uso.
3. Presentación con el equipo.