CONSUMO DE API

Consumir una API (Application Programming Interface) es el proceso de interactuar con una API para acceder a sus datos y utilizarlos en el desarrollo de aplicaciones.

Las API Permiten a las aplicaciones comunicarse entre sí, Permiten aprovechar el código ya existente, Aumentan la velocidad de desarrollo. Cómo consumir una API Iniciar sesión en un portal o mercado de API Agregar la información de la aplicación Enviar solicitudes a la API Consideraciones al consumir una API
Los códigos de estado indican el estado de la petición Los métodos HTTP permiten interactuar con la información de la API Es importante documentar bien la API Si la API es privada, es importante implementar medidas de seguridad Algunos ejemplos de API Google Maps, PayPal, PokéAPI, Random User. Herramientas para consumir API Axios, Requests.

APIs útiles y populares

https://ed.team/blog/las-mejores-apis-publicas-para-practicar

https://www.xn--apaados-6za.es/146-40-apis-utiles-para-disenadores-y-desarrolladores-web

Guía Paso a Paso para Construir una Aplicación de Pokémon con la Poke API

1. Preparación del Entorno

1. **Editor de Código:**
   - Asegúrate de que los estudiantes tengan un editor de código instalado, como Visual Studio Code, Sublime Text o Atom.
2. **Navegador Web:**
   - Utilizar un navegador moderno como Chrome, Firefox o Edge para visualizar la aplicación y depurar errores.
3. **Archivos del Proyecto:**
   - Crea una carpeta para el proyecto y dentro de ella, crea tres archivos:
     - `index.html` (Estructura de la página)
     - `poke.css` (Estilos visuales)

- `poke.js` (Lógica de la aplicación)

2. Estructura HTML (index.html)

1. **Estructura Básica:**
    - Comienza con la estructura básica de un documento HTML5:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Pokemon</title>
   <link rel="stylesheet" href="poke.css">
</head>
<body>
   <form action="" class="formulario">
   <input type="text" class="campo-busqueda">
   <button class="boton-buscar">Buscar pokemon</button>
   </form>

<div class="seccion-pokemon">



</div>

   <script src="poke.js"></script>
</body>
</html>
```

- **Explicación**:
    - Se incluye un `input` para que el usuario ingrese el nombre o número del Pokémon.
    - Un `button` para iniciar la búsqueda.
    - Un div con id="pokemonInfo", para mostrar la información del pokemon obtenido de la api.
    - Se enlaza el archivo `style.css` para los estilos y `script.js` para la lógica.

3. Estilos CSS (style.css)

1. **Estilos Básicos:**
    - Agrega estilos para mejorar la apariencia de la aplicación:

CSS

```
.formulario{
padding: 2em;
background-color: fuchsia;
}

.campo-busqueda{
width: 300px;

}

.//-boton-buscar{
background-color: dodgerblue;
color: floralwhite;
cursor: pointer;
padding: 5px;
}

.seccion-pokemon{
padding: 10px;
background-color: goldenrod;
}

.pokemon-info{
color: red;

}
```

- **Explicación**:
  - Se centra el contenido en la página.
  - Se da estilo al área donde se mostrará la información del Pokémon.
  - Se limita el tamaño de las imagenes de los pokemon.

4. Lógica JavaScript (script.js)

1. **Obtener Elementos del DOM:**
   - Captura los elementos HTML que se utilizarán:

JavaScript

```
const campoBusqueda=traerElemento('.campo-busqueda'),
botonBuscar=traerElemento('.boton-buscar'),
container=traerElemento('.seccion-pokemon');

const URL_API = 'https://pokeapi.co/api/v2/pokemon/';

var pokeName,
```

`recibePokemon;`

2. **Función para Buscar Pokémon:**
    - Crea una función que realice la llamada a la API y muestre la información:

JavaScript

```
 function traerElemento(element){

return document.querySelector(element);

}



function solicitarInfo(URL_API, name){

    fetch(URL_API + name)

    .then(response => response.json())

    .then(data => {

     recibePokemon = data;

    })

    .catch(err => console.log(err));



}
```

    - Se utiliza `fetch` para hacer la llamada a la API.
    - Se convierte la respuesta a formato JSON.
    - Se maneja los errores con un bloque try catch.

3. **Función para Mostrar Información:**
    - Crea una función que muestre los datos del Pokémon en el HTML:

JavaScript

```
function imprimir(){

 container.innerHTML =`
```

```
<div class= "pokemon-picture">
<img src="${recibePokemon.sprites.front_default}">
</div>

<div class= "pokemon-info">
<h1>Nombre: ${recibePokemon.name}</h1>
<h2>Cod: ${recibePokemon.id}</h2>
<h3>Altura: ${recibePokemon.height/10} mts</h3>
</div>
    `
}

function iniciar(pokeName){
solicitarInfo(URL_API, pokeName);
imprimir();
}

    botonBuscar.addEventListener('click', event => {
    event.preventDefault();
    pokeName=campoBusqueda.value.toLowerCase();
    iniciar(pokeName);

});
```

Explicación:

- Se extraen el nombre, la imagen y los tipos del Pokémon.
- Se construye el HTML con la información.

4. **Evento del Botón:**
   - Agrega un evento al botón para que llame a la función `buscarPokemon`:

JavaScript

```
botonBuscar.addEventListener('click', event => {

    event.preventDefault();

    pokeName=campoBusqueda.value.toLowerCase();

    iniciar(pokeName);
```

5. Pruebas y Depuración

1. **Abrir el HTML en el Navegador:**
   - Abre el archivo `index.html` en tu navegador web.
2. **Probar la Aplicación:**
   - Ingresa el nombre o número de un Pokémon y haz clic en "Buscar".
3. **Depurar Errores:**
   - Utiliza las herramientas de desarrollador del editor o navegador (presiona F12) para revisar la consola en busca de errores y depurar el código.