ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
INTRODUCCIÓN AL DESARROLLO WEB
2025 II
LABORATORIO 19 – JAVASCRIPT: CONSUMO DE APIS

I

OBJETIVOS

- Comprender la importancia de JavaScript en el desarrollo web
- Comprender los fundamentos del consumo de APIs en JavaScript
- Solucionar ejercicios aplicando los conocimientos obtenidos

TIEMPO ESTIMADO: 2 horas

II

CONSIDERACIONES DE EVALUACIÓN

- Se deberán utilizar los conocimientos impartidos en las clases teóricas
- Deberá utilizar nombre de variables significativos
- Deberá realizar pruebas adicionales
- El alumno deberá indicar en su código con quien colaboró, así sea la IA
- El alumno será requerido de realizar modificaciones en su código y responder a preguntas sobre el mismo
- Los ejercicios deberán realizarse en el laboratorio, a su ritmo y subirlos al aula virtual como AVANCE antes de finalizar la clase
- Todos los ejercicios completos, deberán ser subidos al aula virtual como TAREA, para lo cual se les dará un tiempo adecuado y deben cumplir el deadline estipulado. Esto se deberá cumplir, aunque en el laboratorio ya se hayan terminado todos los ejercicios
- El formato a usar para los avances y tareas será .zip, que contenga todos los archivos requeridos
- Utilizar Git con GitHub para el manejo de versiones de su trabajo, ocasionalmente se le solicitará que compartan sus repositorios

III

POLITICA DE COLABORACION

La política del curso es simple, a menos que se exprese lo contrario en el laboratorio, siéntase libre de colaborar con sus compañeros en todos los laboratorios, pero debe notificar expresamente con quien ha colaborado. La colaboración con alumnos, que no están matriculados en el curso está prohibida. Los laboratorios y asignaciones han sido desarrollados para ayudarlo a comprender el material. Conozca su código y esté preparado para revisiones individuales de código. Durante las revisiones es probable que se le pida realizar modificaciones y justificar sus decisiones de programación. Cada uno de sus ejercicios debe iniciar de la siguiente forma:

```
// Laboratorio  Nro x - Ejerciciox
// Autor: mi nombre
// Colaboró : el nombre
// Tiempo :
```

INDICACIONES GENERALES

a. En cada sesión de laboratorio, los ejercicios propuestos deberán ser guardados en la misma carpeta/directorio
b. La carpeta deberá tener el nombre del Laboratorio y el nombre del alumno, así por ejemplo:
Laboratorio 11 – Juan Perez
c. Utilice nombres significativos
d. Su código deberá estar correctamente indentado y preferentemente documentado
e. Deberá ser debidamente probado

MARCO TEORICO

1. ¿Qué es API?

Una API (Application Programming Interface) es un conjunto de reglas, definiciones y protocolos que permiten que dos sistemas o aplicaciones se comuniquen entre sí.
Una API es un puente que permite que una aplicación/sistema/programa use funciones o datos de otro, sin conocer su funcionamiento interno.

2. ¿Qué hace exactamente una API?

Es como un manual o contrato que indica cómo interactuar correctamente con un sistema

- Qué puedes pedir: explica qué operaciones están permitidas (ejemplo: obtener datos, enviarlos, actualizar algo)
- Cómo debes pedirlo: define cómo deben enviarse las solicitudes (formato, estructura, parámetros)
- Qué recibirás a cambio: Especifica cómo serán las respuestas que devuelve

3. Metáfora

- El cliente (tu programa) pide un plato
- El menú (API) te dice qué platos puedes pedir y cómo pedirlos
- El mesero (API) entrega tu pedido y lleva la comida de la cocina hacia ti
- La cocina (servidor) prepara la comida

4. API Web (HTTP/REST)

Son las más usadas hoy (hay otros tipos). Se consumen mediante URL.
PokéAPI es una API REST.

Ejemplo:

```
https://pokeapi.co/api/v2/pokemon/ditto
```

5. Partes principales de una API

1) Endpoints

Son las direcciones URL a las que se hace solicitudes.

Ejemplos:

```
https://pokeapi.co/api/v2/pokemon/pikachu
https://pokeapi.co/api/v2/pokemon/25
https://pokeapi.co/api/v2/type/fire
https://pokeapi.co/api/v2/type/water
https://pokeapi.co/api/v2/ability/overgrow
https://pokeapi.co/api/v2/move/thunder-punch
https://pokeapi.co/api/v2/pokemon-species/bulbasaur
https://pokeapi.co/api/v2/pokemon-form/pikachu
```

2) Métodos HTTP

Indican la acción que se quiere realizar:

- GET → obtener datos
- POST → enviar datos
- PUT → actualizar
- DELETE → eliminar

3) Formato de respuesta

La mayoría de APIs usan JSON, un formato de texto fácil de leer y enviar.

Ejemplo JSON:

```
{
  "nombre": "Pikachu",
  "tipo": "eléctrico"
}
```

6. ¿Qué significa "consumir una API"?

Consumir una API significa hacer solicitudes (requests) desde tu aplicación para obtener datos, enviar información o activar acciones en un servidor externo.

En JavaScript normalmente se realiza con:

- fetch()
- Librerías como Axios (opcional)

7. Ejemplos

1) Obtener un Pokémon por nombre (básico)

```
fetch("https://pokeapi.co/api/v2/pokemon/pikachu")
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error("Error:", err));
```

2) Mostrando solo un dato

```
fetch("https://pokeapi.co/api/v2/pokemon/charizard")
  .then(res => res.json())
  .then(data => console.log("Altura:", data.height))
  .catch(err => console.error("Error:", err));
```

3) Usando async/await

```
async function obtenerPokemon() {
    const res = await fetch("https://pokeapi.co/api/v2/pokemon/charizard");

    const data = await res.json();
    console.log("Altura:", data.height);
}

obtenerPokemon();
```

4) Usando async/await con manejo de errores

```
async function obtenerPokemon() {
  try {
    const res = await fetch("https://pokeapi.co/api/v2/pokemon/charizard");

    // Verificar si hubo error HTTP (404, 500, etc.)
    if (!res.ok) {
      throw new Error("Error en la solicitud: " + res.status);
    }

    const data = await res.json();
    console.log("Altura:", data.height);

  } catch (err) {
    console.error("Error:", err);
  }
}

obtenerPokemon();
```

5) Mostrar nombre, peso y habilidades

```
async function infoPokemon(nombre) {
  try {
    const res = await fetch(`https://pokeapi.co/api/v2/pokemon/${nombre.toLowerCase()}`);

    // Verifica errores HTTP (como 404)
    if (!res.ok) {
      throw new Error(`Pokémon "${nombre}" no encontrado (Código: ${res.status})`);
    }

    const data = await res.json();

    console.log("Nombre:", data.name);
    console.log("Peso:", data.weight);

    const habilidades = data.abilities.map(a => a.ability.name);
    console.log("Habilidades:", habilidades);

  } catch (error) {
    console.error("Error:", error.message);
  }
}

infoPokemon("bulbasaur");
```

6) Mostrar en un html el nombre, ID, peso, altura y habilidades de Pikachu

HTML

```
<h1>Datos del Pokémon</h1>
<div id="card"></div>

<script src="script.js"></script>
```

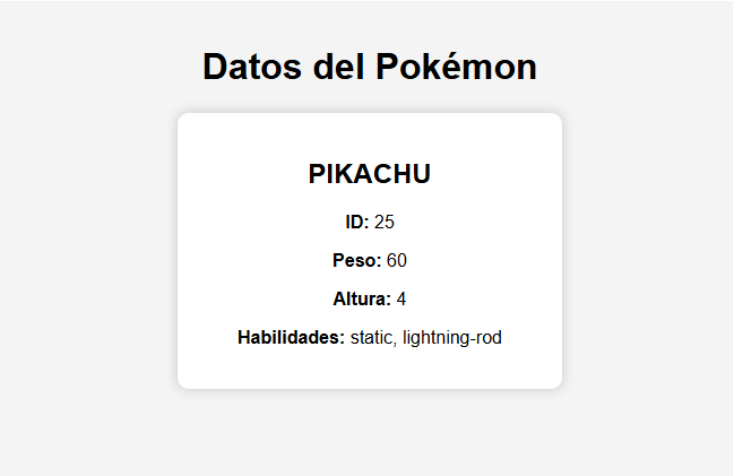
CSS

```css
body {
    font-family: Arial;
    background: #f4f4f4;
    padding: 20px;
    text-align: center;
}
#card {
    margin-top: 20px;
    background: white;
    padding: 20px;
    border-radius: 10px;
    width: 300px;
    margin-left: auto;
    margin-right: auto;
    box-shadow: 0 0 10px rgba(0,0,0,0.2);
}
```

JS

```js
async function mostrarPokemon() {
    const card = document.getElementById("card");
    const nombre = "pikachu";  // ← CAMBIAR AQUÍ EL POKÉMON

    try {
        const res = await fetch(`https://pokeapi.co/api/v2/pokemon/${nombre}`);
        const data = await res.json();

        const habilidades = data.abilities.map(a => a.ability.name).join(", ");

        card.innerHTML = `
            <h2>${data.name.toUpperCase()}</h2>
            <p><strong>ID:</strong> ${data.id}</p>
            <p><strong>Peso:</strong> ${data.weight}</p>
            <p><strong>Altura:</strong> ${data.height}</p>
            <p><strong>Habilidades:</strong> ${habilidades}</p>
        `;
    } catch (error) {
        card.innerHTML = "<h3>Error al obtener datos</h3>";
    }
}

// Ejecutar automáticamente al cargar la página
mostrarPokemon();
```

RENDERIZADO



7) Mostrar en un html la imagen, el nombre, ID, peso, altura y habilidades de Pikachu

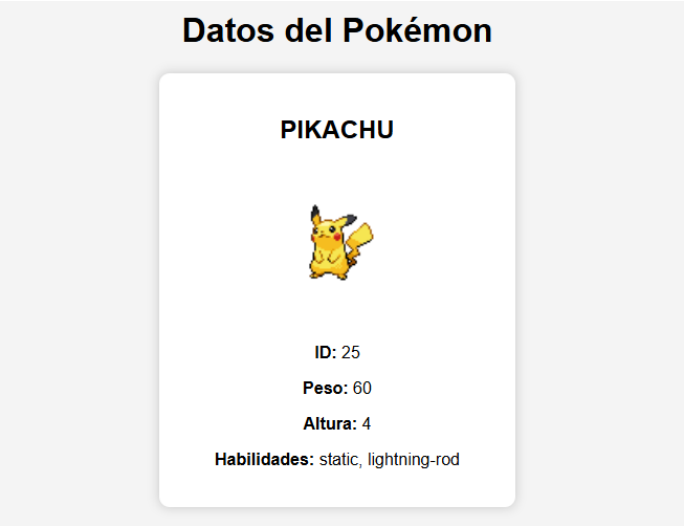
En CSS aumentar:

```
img {
    width: 150px;
}
```

En JS aumentar en el innerHTML creado:

```
<img src="${data.sprites.front_default}" alt="${data.name}">
```

RENDERIZADO



1. Crear un directorio que tenga su nombre y un subdirectorio laboratorio19. Recomendación: usar minúsculas, sin espacios, sin tildes ni "ñ" y con guiones medios o bajos
2. Utilizar los atajos de teclado o combinación de teclas para agilizar su trabajo
3. Pide al usuario un ID de Pokémon y muestra en consola su name
4. Usa .then para obtener altura y peso de Pikachu
5. Usa async/await para obtener altura y peso de Pikachu
6. Muestra en consola la URL de sprites.front_default de Charizard
7. Listar los primeros 20 Pokémon. Usar `https://pokeapi.co/api/v2/pokemon?limit=20`
8. Obtener un Pokémon aleatorio. Genera un número del 1 al 898 y busca ese Pokémon
9. Buscador visual con HTML + JS. Ingresar el ID de un Pokémon y mostrar sus datos principales en el html (imagen, el nombre, ID, peso, altura y habilidades)
10. Crear un html que, al cargar, obtenga los Pokémon del 1 al 10 y muestre:
    Nombre, imagen e ID
    Cada Pokémon debe aparecer en una tarjeta (div) distinta.
    Tips:
    Usa un for para pedir varios Pokémon
    Usa data.sprites.front_default para las imágenes
    Almacena todos los resultados en una lista y muéstralos en el HTML
11. Crear un programa donde el usuario escriba el nombre o ID de un Pokémon en un input, y el sistema muestre:
    Nombre, imagen, tipos (fire, water, grass, electric, etc.)
    Ejemplo:
    Charizard → tipos: fire, flying
    Tips:
    Los tipos vienen en data.types[0].type.name
    Un Pokémon puede tener 1 o 2 tipos → mostrar ambos
12. Mostrar estadísticas base de un Pokémon
13. Crear un html que pida por input un Pokémon y muestre sus stats: hp, attack, defense, speed, special-attack, special-defense
    Muestra cada estadística en una lista o tabla
    Tips: Los stats vienen en data.stats[i].stat.name y en data.stats[i].base_stat
14. Crea una aplicación web que muestre información básica de los primeros 12 Pokémon (IDs del 1 al 12). La aplicación debe cumplir exactamente lo siguiente:
    - Mostrar tarjetas de Pokémon
    - La aplicación debe mostrar solo 3 Pokémon a la vez
    - Cada Pokémon debe aparecer dentro de una tarjeta que muestre:
      Imagen, nombre, ID
    - Navegación entre grupos de 3 Pokémon
    - Agrega dos botones:
      Anterior y Siguiente cuyo comportamiento debe ser:
      Botón "Siguiente"
        Al presionarlo, debe mostrar los siguientes 3 Pokémon
      Botón "Anterior"
        Debe volver al grupo de 3 Pokémon anterior

    Tips:
    Para cargar la información primero debes obtener los Pokémon del 1 al 12 (puedes cargarlos todos al iniciar)
    Las tarjetas mostradas deben provenir de esa lista ya cargada
15. Crear un repositorio remoto en GitHub y subir tu repositorio local. Compartir URL y pdf con captura de pantalla del código de los archivos js y de la ejecución

I. TAREA PARA LA CASA: Complete todos los ejercicios.

Crear un documento docx/pdf con la solución de los ejercicios.

Subir el documento a la tarea **Tarea 19** del Aula Virtual respetando las fechas indicadas.