UNIVERSIDAD DEL VALLE

INGENIERIA DE SISTEMAS INFORMATICOS

CINE-JMMV

ESTUDIANTE: JOSE MARIO MACUCHAPI VILLA

DOCENTE: ING. MIRANDA ORDOÑEZ HENRY

MATERIA: PROGRAMACION WEB I

GRUPO: C

La Paz-Bolivia
2025

1. Introducción

Este documento describe el funcionamiento técnico de mi proyecto web "CINE JMMV", que es una página para ver películas populares usando información de The Movie Database (TMDB). El proyecto lo hice como parte de mi aprendizaje en desarrollo web y para demostrar mis conocimientos en HTML, CSS y JavaScript.

1.1 Objetivo del Proyecto

Crear una página web funcional donde los usuarios puedan:

- Ver un catálogo de películas actuales
- Buscar películas por nombre
- Filtrar por género (acción, terror, comedia, etc.)
- Ver información detallada de cada película
- Navegar entre páginas de resultados

2. Tecnologías Usadas

Para este proyecto usé las siguientes tecnologías que aprendí durante el curso:

- HTML5: Para la estructura de la página
- CSS3: Para los estilos y diseño
- JavaScript: Para la lógica y consumo de la API
- TMDB API: Para obtener los datos de las películas

No usé ningún framework como React o Vue porque quise hacerlo con JavaScript puro para entender mejor cómo funciona todo.

3. Estructura de Archivos

Mi proyecto tiene 4 archivos principales:

CINE-JMMV/

index.html → Página principal con toda la estructura

style.css → Todos los estilos y colores

script.js → Código JavaScript para la funcionalidad

CJMMV.png → Logo que diseñé para el proyecto

Es una estructura simple porque así es más fácil de mantener y entender.

4. Explicación del HTML (index.html)

4.1 Estructura General

Organicé mi página en 5 secciones principales:

1. Header (Cabecera): Aquí puse el logo, el título y el menú de navegación. También incluí el buscador y el filtro de géneros.

2. Contenedor de Películas: Es donde se muestran todas las películas en forma de tarjetas (cards). Esta parte se genera dinámicamente con JavaScript.

3. Paginador: Los botones para cambiar de página (anterior, números, siguiente).

4. Footer (Pie de página): Información sobre mí, mis datos de contacto y créditos.

5. Modal: Una ventana emergente que se abre cuando le das clic a "Ver descripción" en una película.

4.2 Partes Importantes del Código

El Header:

html

```
<header class="barra-superior">
  <div class="logo-titulo">
    <img src="CJMMV.png" alt="Logo">
    <h1>CINE - JMMV</h1>
  </div>

  <button class="hamburguesa" id="btnMenu">☰</button>

  <nav class="menu" id="menu">
    <!-- Aquí van los enlaces y controles -->
  </nav>
```

```
</header>
```

El botón hamburguesa (☰) lo puse para que en celulares el menú se pueda abrir y cerrar, así se ve mejor en pantallas pequeñas.

El Selector de Géneros:

html

```
<select id="filtroGenero">
  <option value="">Todos</option>
  <option value="28">Acción</option>
  <option value="35">Comedia</option>
  <!-- etc... -->
</select>
```

Cada género tiene un número (ID) que es el que usa la API de TMDB. Por ejemplo, Acción es el 28 y Terror es el 27.

El Modal:

html

```
<div id="modal" class="modal">
  <div class="modal-contenido">
    <span id="cerrarModal" class="cerrar">&times;</span>
    <h3 id="modalTitulo"></h3>
    <p id="modalDesc"></p>
    <p id="modalFecha"></p>
  </div>
</div>
```

El modal está oculto por defecto y se muestra cuando haces clic en el botón de ver descripción.

5. Explicación del CSS (style.css)

5.1 Colores que Elegí

Hice una paleta de colores que me pareció que se ve bien para un sitio de películas:

css

```
:root {
  --naranja: #E97A35;          /* Para botones y títulos */
  --naranja-oscuro: #C65B1A;   /* Para hover */
  --azul-petroleo: #2A4B5A;    /* Para la barra superior */
  --azul-verdoso: #4F7A82;     /* Para las tarjetas */
  --crema: #F2E3C6;            /* Para el texto */
  --azul-profundo: #1A2F36;    /* Para el fondo */
}
```

Usé variables CSS para que sea más fácil cambiar los colores después si quiero.

5.2 El Grid de Películas

Para mostrar las películas usé CSS Grid porque aprende que es la mejor forma de hacer layouts:

css

```
.grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(220px, 1fr));
  gap: 20px;
}
```

Esto hace que las tarjetas se acomoden automáticamente según el tamaño de la pantalla. Si hay espacio para 5, muestra 5, si solo hay para 3, muestra 3.

5.3 Animaciones que Agregué

Me gustó aprender sobre animaciones CSS y agregué algunas:

Cuando aparecen las tarjetas:

css

```
@keyframes aparecer {
```

```
  from {
    opacity: 0;
    transform: translateY(20px);
  }
  to {
    opacity: 1;
    transform: translateY(0);
  }
}
```

Esto hace que las tarjetas aparezcan desde abajo con un efecto suave.

Efecto hover en las tarjetas:

css

```
.item:hover {
  transform: translateY(-10px);
  box-shadow: 0 10px 25px rgba(233, 122, 53, 0.3);
}
```

Cuando pasas el mouse sobre una película, se levanta un poquito y se le pone una sombra naranja.

5.4 Diseño Responsive

Hice que la página se vea bien en celulares usando media queries:

Para tablets (768px):

- Aparece el botón hamburguesa
- El menú se pone vertical
- Los elementos ocupan el 90% del ancho

Para celulares (480px):

- El logo se hace más pequeño
- El título también se reduce
- Los botones del paginador se ajustan mejor

6. Explicación del JavaScript (script.js)

Esta es la parte más importante porque aquí está toda la lógica de mi página.

6.1 Variables Principales

Al inicio declaré todas las variables que necesito:

javascript

```javascript
const API_KEY = "912dd0733c76e627da6361f7bc7a8797";
const PELICULAS_POR_PAGINA = 15;
const MAX_PELICULAS = 200;

let paginaActual = 1;
let genero = "";
let todasPeliculas = [];
```

- API_KEY: Es mi clave para usar la API de TMDB (la saqué gratis registrándome en su página)
- PELICULAS_POR_PAGINA: Decidí mostrar 15 películas por página
- MAX_PELICULAS: Cargo máximo 200 películas para que no sea tan pesado
- todasPeliculas: Aquí guardo todas las películas que descargo

6.2 Función para Cargar Películas Populares

Esta es la función que se ejecuta cuando abres la página:

javascript

```javascript
async function cargarPopulares() {
  todasPeliculas = [];
  let paginaTMDB = 1;

  while (todasPeliculas.length < MAX_PELICULAS) {
    const r = await fetch(
```

```
        `https://api.themoviedb.org/3/movie/popular?api_key=${API_KEY}&language=es-ES&page=${paginaTMDB}`
      );
      const resultados = (await r.json()).results;
      if (!resultados || resultados.length === 0) break;
      todasPeliculas = todasPeliculas.concat(resultados);
      paginaTMDB++;
    }

    aplicarBusqueda();
}
```

¿Qué hace?

1. Limpia el array de películas
2. Hace peticiones a la API página por página
3. Va agregando los resultados hasta tener 200 películas
4. Al final llama a aplicarBusqueda() para mostrarlas

Usé async/await porque aprendí que es mejor que usar .then() para promesas.

6.3 Función para Mostrar las Películas

Esta función es la que realmente pone las películas en la página:

javascript

```
function mostrarPagina(lista, pagina) {
  contenedor.innerHTML = "";

  if (!lista?.length) {
    contenedor.innerHTML = "<p>No se encontraron películas.</p>";
    return;
  }

  const inicio = (pagina - 1) * PELICULAS_POR_PAGINA;
  const fin = inicio + PELICULAS_POR_PAGINA;
```

```
    const subLista = lista.slice(inicio, fin);

    subLista.forEach(p => {
        const poster = p.poster_path
            ? `https://image.tmdb.org/t/p/w500${p.poster_path}`
            : "https://via.placeholder.com/500x750?text=Sin+Imagen";

        const item = document.createElement("div");
        item.className = "item";
        item.innerHTML = `
            <a
href="https://www.youtube.com/results?search_query=${encodeURIComponent(p.title + "
trailer")}" target="_blank">
                <img src="${poster}" class="poster">
            </a>
            <button class="btn-desc">Ver descripción</button>
        `;

        item.querySelector(".btn-desc").onclick = () => {
            modalTitulo.textContent = p.title;
            modalDesc.textContent = p.overview || "Sin descripción disponible.";
            modalFecha.innerHTML = `<strong>Estreno:</strong> ${p.release_date || "Fecha no
disponible"}`;
            modal.style.display = "flex";
        };

        contenedor.appendChild(item);
    });
}
```

Lo que hace paso por paso:

1. Limpia el contenedor (borra lo que había antes)
2. Calcula qué películas mostrar según la página actual
3. Para cada película crea una tarjeta (div) con:
    - El póster (imagen)

- Un enlace a YouTube para buscar el trailer
- Un botón para ver la descripción

4. Le pone un evento al botón para abrir el modal

6.4 El Buscador

Para el buscador implementé un "debounce" que aprendí investigando:

javascript

```
let tiempoEspera;
buscar.oninput = () => {
  clearTimeout(tiempoEspera);
  tiempoEspera = setTimeout(() => {
    paginaActual = 1;
    aplicarBusqueda();
  }, 300);
};
```

¿Por qué hice esto? Si buscas la película cada vez que escribes una letra, sería muy lento. Entonces espero 300 milisegundos (0.3 segundos) después de que dejas de escribir para hacer la búsqueda. Así es más eficiente.

6.5 El Paginador

Esta fue la parte más difícil. Quería que se viera como los paginadores profesionales:

javascript

```
function crearPaginador(lista) {
  paginador.innerHTML = "";
  if (!lista) lista = todasPeliculas;

  const totalPaginas = Math.ceil(lista.length / PELICULAS_POR_PAGINA);
  if (totalPaginas <= 1) return;

  const maxVisibles = 5;
```

```
    // Botón anterior
    addBtn("<", paginaActual - 1, false, paginaActual === 1);

    // Primera página
    addBtn("1", 1, paginaActual === 1);

    // Puntos suspensivos si estamos lejos del inicio
    if (paginaActual > 3)
        addBtn("...", null, false, true);

    // Páginas del medio
    const inicio = Math.max(2, paginaActual - 1);
    const fin = Math.min(totalPaginas - 1, paginaActual + 1);

    for (let i = inicio; i <= fin; i++) {
        addBtn(i, i, i === paginaActual);
    }

    // Puntos suspensivos si estamos lejos del final
    if (paginaActual < totalPaginas - 2)
        addBtn("...", null, false, true);

    // Última página
    if (totalPaginas > 1)
        addBtn(totalPaginas, totalPaginas, paginaActual === totalPaginas);

    // Botón siguiente
    addBtn(">", paginaActual + 1, false, paginaActual === totalPaginas);
}
```

Esto hace que si tienes muchas páginas (por ejemplo 20), no muestra todos los botones sino solo:

- Botón anterior

- Página 1
- ...
- Páginas cercanas a la actual
- ...
- Última página
- Botón siguiente

Por ejemplo: < 1 ... 8 9 10 ... 20 >

6.6 Menú Hamburguesa para Móviles

Para que el menú funcione en celulares agregué este código simple:

javascript

```
btnMenu.onclick = function() {
  menu.classList.toggle("abierto");
};
```

Cuando haces clic en el botón ☰, le agrega o quita la clase "abierto" al menú, y en el CSS ya está definido que con esa clase se muestre.

7. Cómo Funciona la API de TMDB

7.1 ¿Qué es una API?

Una API (Application Programming Interface) es como un mesero en un restaurante. Tú le pides algo (una solicitud) y él te trae lo que pediste (una respuesta). En este caso, yo le pido películas a TMDB y me devuelve información sobre ellas.

7.2 Endpoints que Uso

Para películas populares:

https://api.themoviedb.org/3/movie/popular?api_key=TU_CLAVE&language=es-ES&page=1

Para películas por género:

https://api.themoviedb.org/3/discover/movie?api_key=TU_CLAVE&language=es-ES&with_genres=28&page=1

7.3 Los Géneros y sus IDs

Investigué y estos son los IDs que usa TMDB para cada género:

ID	Género
28	Acción
12	Aventura
16	Animación
35	Comedia
80	Crimen
18	Drama
27	Terror
878	Ciencia Ficción

7.4 7.4 Ejemplo de Respuesta

Cuando hago una petición, la API me devuelve algo así:

json


```
{
 "results": [
  {
   "id": 12345,
   "title": "Spider-Man",
   "overview": "Peter Parker es un estudiante...",
```

```
    "poster_path": "/ruta-del-poster.jpg",
    "release_date": "2025-06-15"
  }
 ]
}
```


Yo solo uso algunos campos como title, overview, poster_path y release_date.

8. Problemas que Tuve y Cómo los Resolví

    8.1 Problema 1: Las imágenes no cargaban

El problema: Al principio no se veían los pósters de las películas.

Solución: Me di cuenta que la API solo da la ruta (/abc123.jpg) pero no la URL completa. Tuve que agregar https://image.tmdb.org/t/p/w500 antes de la ruta.

    8.2 Problema 2: El buscador era muy lento

El problema: Cada vez que escribía una letra se hacía una búsqueda, haciendo la página lenta.

Solución: Implementé el debounce con setTimeout() para esperar a que el usuario termine de escribir.

    8.3 Problema 3: El modal no se cerraba

El problema: El modal se abría pero no había forma de cerrarlo.

Solución: Agregué dos formas de cerrar:

1. Haciendo clic en la X
2. Haciendo clic fuera del contenido del modal

    8.4 Problema 4: En celular el menú se veía mal

El problema: Los enlaces y el buscador se amontonaban en pantallas pequeñas.

Solución: Aprendí sobre media queries y hice un menú hamburguesa que se muestra solo en pantallas menores a 768px.

9. Cosas que Aprendí

9.1 Sobre JavaScript

- Cómo usar async/await para APIs
- La diferencia entre innerHTML y textContent
- Cómo funcionan los eventos (onclick, oninput, etc.)
- El concepto de debouncing
- Manipulación del DOM (crear elementos, agregar clases, etc.)

9.2 Sobre CSS

- Variables CSS con :root
- CSS Grid para layouts
- Animaciones con @keyframes
- Transiciones con transition
- Media queries para diseño responsive

9.3 Sobre APIs

- Cómo hacer peticiones con fetch()
- Parsear JSON con .json()
- Parámetros de URL (?api_key=...&language=...)
- Paginación en APIs

10. Limitaciones de mi Proyecto

Soy honesto con las limitaciones que tiene mi proyecto:

1. La API key está visible: Cualquiera puede ver mi clave en el código fuente. Lo ideal sería tener un servidor backend pero no lo aprendí todavía.
2. Solo 200 películas: Puse un límite porque si cargo todas sería muy pesado. La API tiene miles de películas.
3. No hay favoritos: Me hubiera gustado que puedas guardar tus películas favoritas pero tendría que usar localStorage o una base de datos.
4. Solo está en español: La API soporta varios idiomas pero solo configuré español.
5. No hay sistema de usuarios: No puedes crear una cuenta ni guardar preferencias.

11. Mejoras Futuras

Si tuviera más tiempo, me gustaría agregar:

1. Sistema de favoritos usando localStorage
2. Filtros combinados (por ejemplo: Terror + 2024)
3. Ordenar por fecha, rating, etc.
4. Ver trailer en la misma página en lugar de ir a YouTube
5. Modo oscuro/claro para que el usuario elija
6. Backend propio para ocultar la API key
7. Sistema de recomendaciones tipo "Si te gustó esta, te puede gustar..."

12. Cómo Instalar y Usar

12.1 Requisitos

- Navegador web moderno (Chrome, Firefox, Edge, Safari)
- Conexión a internet (para cargar las películas de la API)
- Un editor de código si quieres modificarlo (recomiendo VS Code)

12.2 Pasos para Usarlo

1. Descargar los archivos del proyecto
2. Abrir el archivo index.html en tu navegador
3. ¡Listo! Debería cargar automáticamente

12.3 Si Quieres Modificarlo

1. Abre los archivos en tu editor de código
2. Modifica lo que quieras (colores en CSS, funciones en JS, etc.)
3. Guarda y recarga la página en el navegador

13. Conclusión

Este proyecto me ayudó mucho a entender cómo funcionan las páginas web modernas. Aprendí que no solo es importante que se vea bonito (CSS) sino que funcione bien (JavaScript) y esté bien estructurado (HTML).

Lo más difícil fue el paginador y hacer que todo sea responsive, pero investigando en internet y viendo tutoriales pude resolverlo.

Estoy orgulloso del resultado final aunque sé que tiene cosas que mejorar. Lo importante es que funciona y aprendí mucho haciéndolo.

14. Referencias y Recursos

Estos son los recursos que usé para aprender y hacer el proyecto:

- TMDB API Docs: https://developers.themoviedb.org/3
- MDN Web Docs: Para consultar sobre HTML, CSS y JavaScript
- W3Schools: Para ejemplos de código
- YouTube: Varios tutoriales sobre fetch API y CSS Grid
- Stack Overflow: Para resolver dudas específicas

15. Datos de Contacto

Nombre: José Mario Macuchapi Villa
Email: josemariomacuchapivilla@gmail.com
Teléfono: 79135310
Ubicación: La Paz, Bolivia
Fecha del Proyecto: Noviembre 2025