TECNOLÓGICO SUPERIOR DE PURÍSIMA DEL RINCÓN

Nombres del Alumno (a):

Alva Zavala Gabriela – RS21110105

Botello Santos Rufino – RS21110220

Márquez Robles Emmanuel – RS21110007

Medina Martínez Oscar Eduardo - RS21110201

Quintana Salas Abraham de Jesús – RS21110200

Carrera:

Ing. Informática

Grado y grupo:

6-A

Materia:

Programación en Ambiente Cliente Servidor

Nombre del docente:

Valentín Calzada Ledesma

Tema:

Proyecto 3er Parcial: Aplicación WEB en Angular.

Purísima del Rincón, Gto. A 27 de mayo de 2024

1. Introducción

El área de la informática está enfocada en muchas ramas, desde la programación de algoritmos básicos y complejos hasta la parte de las redes de telecomunicación, lo cual incluye también el protocolo de red Internet. Gracias a este protocolo es que podemos comunicarnos actualmente a través de páginas de lenguaje de hipertexto, páginas como Google, YouTube, Netflix o Facebook. En años pasados, las primeras páginas web que se montaron para ser consumidas fueron creadas y diseñadas en base a 3 cosas: lenguaje HTML, estilo en cascada o CSS y el lenguaje de programación como JavaScript o PHP. Con la evolución de la tecnología fueron saliendo nuevos “marcos de referencia” o frameworks, que son un conjunto de módulos y herramientas que permiten el desarrollo ágil mediante la aportación de librerías y funcionalidades ya creadas.

La materia de Programación en Ambiente Cliente-Servidor abarca diversos temas que van desde la introducción a JavaScript hasta el uso de frameworks para el Desarrollo WEB. Específicamente los temas son: introducción a JS, uso de NodeJS, programación básica con JS, introducción a HTML y CSS, desarrollo web con HTML y CSS, diseño de UX con Bootstrap, eventos de HTML con JS, introducción a JQuery, uso de JQuery, introducción al framework Angular, desarrollo básico en Angular, introducción a TypeScript, desarrollo básico en TypeScript, implementación de API’s, uso de Express, etc.

El proyecto tiene como propósito desarrollar un gestor de películas utilizando el Framework Angular y consumiendo API’s utilizando Express. Además se debe implementar el aspecto de desarrollo de backend y frontend. Para el backend toda la parte del consumo de las API’s, programación de la Playlist, entre otras funcionalidades; y para el frontend se hará uso de elementos de Bootstrap para aplicar UX en la aplicación. Harán uso de los conocimientos obtenidos durante el curso de la materia. Además, se debe ser autodidacta a la hora de programar tanto el Frontend como el Backend.

2. Objetivo general y específicos

Objetivo General:

- Desarrollar una página web en Angular.

Objetivos Específicos:

- Adaptar código HTML al nuevo framework.
- Adaptar código de JS a TS.
- Crear API's y consumirlas.
- Ejecutar frontend y backend.
- Utilizar Express.js

3. Marco teórico

3.1 HTML. *Es el lenguaje con el que se define el contenido de las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web, como imágenes, listas, vídeos, etc. HTML es el componente más básico de la Web. Un elemento HTML se distingue de otro texto en un documento mediante "etiquetas", que consisten en el nombre del elemento rodeado por "<" y ">"* (Álvarez, 2001). Según el autor, el lenguaje de hipertexto es una parte importante y esencial de una página web, ya que la manera de programar se basa en palabaras reservadas encerradas en símbolos de mayor y menor que le dan la estructura a la página web.

3.2 CSS. *Es un lenguaje que maneja el diseño y presentación de las páginas web, es decir, cómo lucen cuando un usuario las visita. Se les denomina hojas de estilo en cascada porque puedes tener varias y una de ellas con las propiedades heredadas (o en cascada) de otras. Con CSS, se pueden crear reglas para decirle a tu sitio web cómo se quiere mostrar la información y guardar los comandos para elementos de estilo (como fuentes, colores, tamaños) separados de los que configuran el contenido* (González, 2023). Según la

autora, el CSS es un archivo en el cual los desarroladores web van insertando el diseño solicitado y deseado para que la página HTML tome más forma y sea más accesible para los usuarios.

3.3 Framework. *Es una herramienta de desarrollo web que se define como una aplicación o conjunto de módulos que permiten el desarrollo ágil de aplicaciones mediante la aportación de librerías y funcionalidades ya creadas. Intenta aliviar el exceso de carga asociado con actividades comunes usadas en desarrollos web. Por ejemplo, muchos framework proporcionan bibliotecas para acceder a bases de datos, estructuras para plantillas y gestión de sesiones, y con frecuencia facilitan la reutilización de código* (Salazar, 2023). Según el autor un framework nos proporciona muchas herramientas para el desarrollo web, además permite reducir el exceso de cargas de páginas web.

3.4 Angular. *Es una plataforma y un framework para crear aplicaciones de una sola página en el lado del cliente usando HTML y TypeScript. Angular está escrito en TypeScript. Implementa la funcionalidad básica y opcional como un conjunto de bibliotecas TypeScript que importas en tus aplicaciones. La arquitectura de una aplicación en Angular se basa en ciertos conceptos fundamentales. Los bloques de construcción básicos son los NgModules, que proporcionan un contexto de compilación para los componentes* (Angular, 2022). Según la definición de la empresa Angular es un framework que permite desarrollar aplicaciones web utilizando TypeScript como lenguaje principal y separando la aplicación en módulos para que cargue mejor la página.

3.5 TypeScript. *Es un superconjunto de JavaScript que añade tipado estático opcional y funciones avanzadas a JavaScript. Ha sido desarrollado por Microsoft y se publicó por primera vez en octubre de 2012. Desde su lanzamiento en 2012, se ha extendido rápidamente entre la comunidad de desarrolladores web* (Kinsta, 2023). Según el autor TypeScript es una versión tipada de JavaScript,

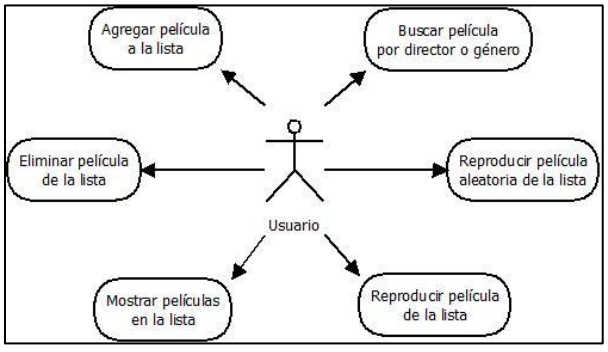
es decir, conserva cierta sintaxis de JS pero hace que sea necesario declarar el tipo de dato de las variables y funciones.

3.6 API. *API significa "interfaz de programación de aplicaciones". En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas. La documentación de su API contiene información sobre cómo los desarrolladores deben estructurar esas solicitudes y respuestas* (Amazon, 2023). Según Amazon, una API es una interfaz que se refiere a un contrato entre el servicio de dos aplicaciones, a través de solicitudes y respuestas para recibir información.

3.7 Express.js. *Es un framework de backend Node.js minimalista, rápido y similar a Sinatra, que proporciona características y herramientas robustas para desarrollar aplicaciones de backend escalables. Te ofrece el sistema de enrutamiento y características simplificadas para ampliar el framework con componentes y partes más potentes en función de los casos de uso de tu aplicación* (Kinsta, 2022). Según el autor, Express es un framework de backend que permite escalar las aplicaciones, además de ofrecer servicio de enrutamiento y ampliación de proyectos con componentes.

4. Metodología

Diagrama de Casos de Uso.



Comandos en CMD.

npm install bootstrap: Instala el paquete Bootstrap para agregar estilos y componentes.

npm install express: Instala Express para crear un servidor web en Node.js y definir rutas y middleware.

npm install cors: Agrega el paquete CORS para manejar solicitudes entre dominios diferentes.

Inicializar proyecto Node.js

npm init -y: Inicializa un proyecto Node.js con configuraciones predeterminadas.

Para el servidor

node index.js: Ejecuta el servidor Node.js.

Agregar servicio

ng generate service api: Genera un servicio en Angular para manejar llamadas a APIs.

Para agregar un componente

ng generate component: Crea un nuevo componente Angular.

Para ejecutar el Proyecto

ng serve: Inicia el servidor de desarrollo de Angular.

Capturas de código.

Main.ts

En Main.ts importaremos todas las funciones y configuraciones esenciales para iniciar nuestra aplicación en Angular. Importaremos bootstrapApplication de @angular/platform-browser para arrancar la aplicación, provideHttpClient de @angular/common/http para peticiones HTTP. Configuramos desde app.config y el componente principal AppComponent desde app.component.

```
TS main.ts
Proyecto3P > src > TS main.ts > ...
1  import { bootstrapApplication } from '@angular/platform-browser';
2  import { appConfig } from './app/app.config';
3  import { AppComponent } from './app/app.component';
4  import { provideHttpClient } from '@angular/common/http';
5
6
7
8  bootstrapApplication(AppComponent, {
9    providers: [provideHttpClient(), ...appConfig.providers]
10 })
11 .catch((err) => console.error(err));
```

API.

Lo siguiente es la API donde se utiliza Node.js y Express.js para crear una API que ofrece información sobre las películas y series de televisión que se encuentran en nuestro proyecto. Utiliza los módulos express y cors para configurar el servidor y permitir solicitudes desde un origen específico el cual es: (http://localhost:4200). Define rutas para obtener detalles de las películas y series almacenadas, mediante solicitudes GET a /api/movies/ o /api/series/. Si no se encuentra la película o serie solicitada, devuelve un código de estado 404 (no encontrado).

```
JS index.js
Proyecto3P > backend > JS index.js > ...
1  const express = require('express');
2  const cors = require('cors');
3  const app = express();
4  const port = 3000;
```

```
const series = {
  truedetective: {
    title: 'True Detective',
    duration:'Proximamente, Suspenso, Misterio',
    description:'Prior investiga las conexiones entre la estación
    image: '/assets/images/Truedetectiveperfil.jpg',
    linkWatch: 'https://www.max.com/mx/es/shows/true-detective/9a
    linkTrailer: 'https://www.youtube.com/watch?v=fVQUcaO4AvE',
    idImage: 'panelFS11',
    images: [
      '/assets/images/Truedetective1.jpg',
      '/assets/images/Truedetective2.jpg',
      '/assets/images/truedetective.jpg'
    ]
  },
```

```
const movies = {
  dune2: {
    title: 'Dune Parte II',
    durationandgenres:'115min. Acción, Aventura, Ciencia ficción.
    duration:'115min',
    genres:'Accion, Aventura, Ciencia ficcion',
    director:'Denis Villeneuve',
    year:'2024',
    description:'Paul Atreides se une a Chani y a los Fremen mien
    image: '/assets/images/duneperfil.jpg',
    linkWatch: 'https://www.youtube.com/watch?v=Way9Dexny3w&pp=yg
    linkTrailer: 'https://www.youtube.com/watch?v=TTgk_iT8Uts',
    idImage: 'panelF1',
    images: [
      '/assets/images/dune2.jpeg',
      '/assets/images/dune2.jpeg',
      '/assets/images/dune3.jpg'
    ]
  },
```

```
app.use(cors({
  origin: 'http://localhost:4200' // Allow requests from this origin
}));

app.get('/api/movies/:id', (req, res) => {
  const movieId = req.params.id;
  const movie = movies[movieId];
  if (movie) {
    res.json(movie);
  } else {
    res.status(404).send({ error: 'Movie not found' });
  }
});
```

```
app.get('/api/movies', (req, res) => {
  res.json(movies);
});

app.get('/api/series/:id', (req, res) => {
  const seriesId = req.params.id;
  const serie = series[seriesId];
  if (serie) {
    res.json(serie);
  } else {
    res.status(404).send({ error: 'Series not found' });
  }
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

movie.service

MovieService, obtiene información sobre las películas desde la API creada posteriormente, obteniendo solicitudes GET. Utiliza el módulo HttpClient. Con ayuda del método 'getMovies' devuelve los datos de la pelicula correspondiente al ID proporcionado.

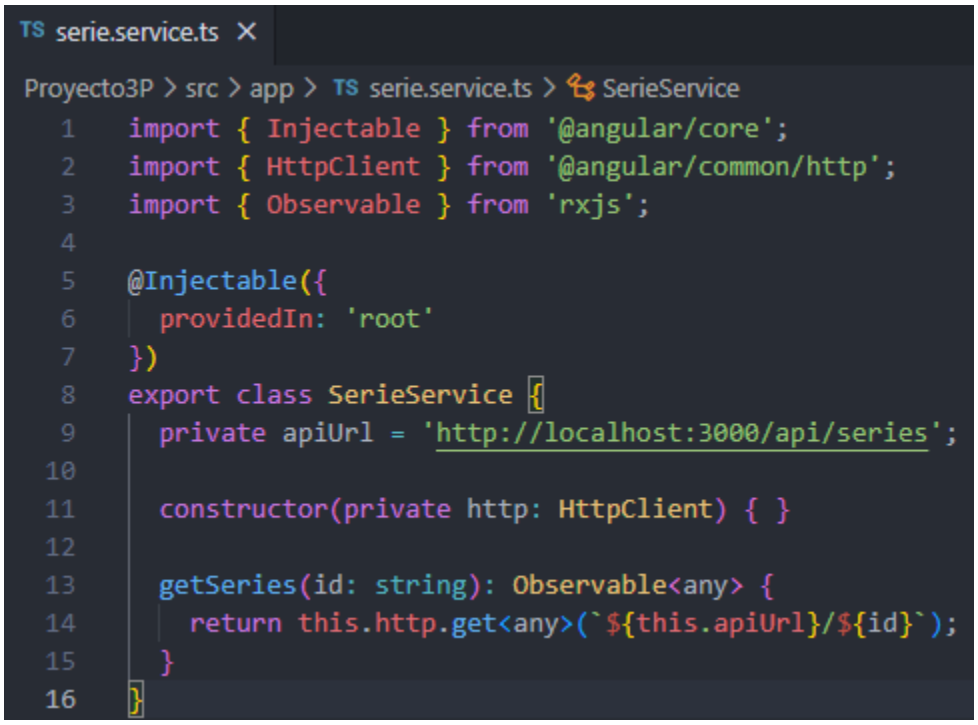
TS movie.service.ts ×

Proyecto3P > src > app > TS movie.service.ts > ...

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class MovieService {

  private apiUrl = 'http://localhost:3000/api/movies';

  constructor(private http: HttpClient) { }

  getMovie(id: string): Observable<any> {
    return this.http.get<any>(`${this.apiUrl}/${id}`);
  }

  getAllMovies(): Observable<any> {
    return this.http.get<any>(this.apiUrl);
  }
}
```

serie.service

SeriesService, obtiene información sobre las series desde la AP, obteniendo solicitudes GET. También utiliza el módulo HttpClient. Con ayuda del método 'getSeries' devuelve los datos de la serie correspondiente al ID.



movie.html

movie.html nos ayuda a visualizar la información de las películas. Estructurando el html para Angular donde controla la visibilidad de los elementos basados en los datos de la película. Tambien se muestra el título, duración, descripción y botones para ver la película y el tráiler. Además, se incluye una galería de imágenes con botones para navegar entre ellas.

```
<div class="container col-xxl-8 px-4 py-5" *ngIf="movieData">
  <div class="row flex-lg-row-reverse align-items-center g-5 py-5" id="{{ movieData.idImage }}">
    <div class="col-10 col-sm-8 col-lg-6">
      <img src="{{ movieData.image }}" class="d-block mx-lg-auto img-fluid" alt="Bootstrap Themes" width
    </div>
    <div class="col-lg-6" >
      <h1 class="display-5 fw-bold text-body-emphasis lh-1 mb-3" id="lead2">{{ movieData.title }}</h1>
      <p class="lead text-white" id="lead2">{{ movieData.durationandgenres }}</p>
      <p class="lead text-white" id="lead2">{{ movieData.description }}</p>
      <div class="d-grid gap-2 d-md-flex justify-content-md-start">
        <button type="button" id="mirarBtn" class="btn btn-primary btn-lg px-4 me-md-2" target="_blank"
        <a class="btn btn-lg btn-primary" target="_blank" href="{{ movieData.linkTrailer }}">Ver Trailer
      </div>
    </div>
  </div>
</div>
```

```
<div id="gallery" *ngIf="movieData">
  <p class="lead1" style ="text-align: center; color: #FFFFFF">Galería</p>

  <img src="{{ path }}" alt="Imagen">

  <div id="botones">
    <button (click)="prevImage()">Anterior</button>
    <button (click)="nextImage()">Siguiente</button>
  </div>
</div>
```

movie.ts

Movie.ts el cual es un componente Angular se encarga de mostrar detalles de las películas. Utiliza el servicio MovieService para obtener los datos de la película a partir de su ID, que se recibe a través de la ruta. Luego, muestra la información de la película y permite navegar entre las imágenes asociadas a la misma. Las funciones prevImage() y nextImage() permiten cambiar entre las imágenes disponibles. La función openLink() abre un enlace en una nueva pestaña del navegador.

TS movie.component.ts ×

Proyecto3P > src > app > movie > TS movie.component.ts > ...

```ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { MovieService } from '../movie.service';
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-movie',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './movie.component.html',
  styleUrl: './movie.component.css'
})
```

```ts
export class MovieComponent implements OnInit {
  movieId: string='';
  movieData: any = {};
  currentIndex: number = 0;
  path: string = "";

  constructor(private route: ActivatedRoute, private movieService: MovieService) { }

  openLink(url: string): void {
    window.open(url, '_blank');
  }

  ngOnInit(): void {
    this.route.params.subscribe(params => {
      this.movieId = params['id'];
      console.log('Movie ID:', this.movieId);
      this.movieService.getMovie(this.movieId).subscribe((data : any )=> {
        console.log('Movie Data:', data);
        this.movieData = data;

        if (this.movieData) {
          //this.initializeGallery();
          this.currentImage();
        }
      });
    });
  }
```

```
currentImage(): void {
  if (this.movieData && this.movieData.images && this.movieData.images.length > 0) {
      this.path = this.movieData.images[this.currentIndex];
      console.log("HOLAAAAAAAAAAAAAAAAA "+this.path);
  }
}

prevImage(): void {
  if (this.movieData && this.movieData.images) { // Verifica si this.movieData y this
    this.currentIndex = (this.currentIndex - 1 + this.movieData.images.length) % this
    this.currentImage();
  }
}

nextImage(): void {
  if (this.movieData && this.movieData.images) { // Verifica si this.movieData y this
    this.currentIndex = (this.currentIndex + 1) % this.movieData.images.length;
    this.currentImage()
  }
}
```

serie.html

serie.html nos ayuda a visualizar la información de las series. Estructurando el html para Angular donde controla la visibilidad de los elementos basados en los datos de la serie. Tambien se muestra el título, duración, descripción y botones para ver la serie y el tráiler. Además, se incluye una galería de imágenes con botones para navegar entre ellas.

```
<div class="container col-xxl-8 px-4 py-5" *ngIf="serieData">
  <div class="row flex-lg-row-reverse align-items-center g-5 py-5" id="{{ serieData.idImage }}">
    <div class="col-10 col-sm-8 col-lg-6">
      <img src="{{ serieData.image }}" class="d-block mx-lg-auto img-fluid" alt="Bootstrap Themes"
    </div>
    <div class="col-lg-6" >
      <h1 class="display-5 fw-bold text-body-emphasis lh-1 mb-3" id="lead2">{{ serieData.title }}<
      <p class="lead text-white" id="lead2">{{ serieData.duration }}</p>
      <p class="lead text-white" id="lead2">{{ serieData.description }}</p>
      <div class="d-grid gap-2 d-md-flex justify-content-md-start">
        <button type="button" id="mirarBtn" class="btn btn-primary btn-lg px-4 me-md-2" (click)="o
        <a class="btn btn-lg btn-primary" target="_blank" href="{{ serieData.linkTrailer }}">Ver T
      </div>
    </div>
  </div>
</div>
```

```
<div id="gallery" *ngIf="serieData">
  <p class="lead1" style ="text-align: center; color: #FFFFFF">Galería</p>

  <img src="{{ path }}" alt="Imagen">

  <div id="botones">
    <button (click)="prevImage()">Anterior</button>
    <button (click)="nextImage()">Siguiente</button>
  </div>
</div>
```

serie.ts

Este componente de Angular, SerieComponent, es responsable de mostrar los detalles de una serie específica, incluyendo la navegación entre las imágenes de la serie. Utiliza el servicio SerieService para obtener los datos de la serie basada en el ID proporcionado en la ruta y ofrece métodos para navegar entre las imágenes de la serie.

```
export class SerieComponent implements OnInit {
  serieId: string='';
  serieData: any;
  currentIndex: number = 0;
  path: string = "";

  constructor(private route: ActivatedRoute, private serieService: SerieService) { }

  openLink(url: string): void {
    window.open(url, '_blank');
  }
  ngOnInit(): void {
    this.route.params.subscribe(params => {
      this.serieId = params['id'];
      console.log('Serie ID:', this.serieId);
      this.serieService.getSeries(this.serieId).subscribe((data : any )=> {
        console.log('Serie Data:', data);
        this.serieData = data;

        if (this.serieData) {
          //this.initializeGallery();
          this.currentImage();
        }
      });
    });
  }
```

```
currentImage(): void {
  if (this.serieData && this.serieData.images && this.serieData.images.length > 0) {
      this.path = this.serieData.images[this.currentIndex];
      console.log("HOLAAAAAAAAAAAAAAAA "+this.path);
  }
}

prevImage(): void {
  if (this.serieData && this.serieData.images) { // Verifica si this.movieData y this
    this.currentIndex = (this.currentIndex - 1 + this.serieData.images.length) % this
    this.currentImage();
  }
}

nextImage(): void {
  if (this.serieData && this.serieData.images) { // Verifica si this.movieData y this
    this.currentIndex = (this.currentIndex + 1) % this.serieData.images.length;
    this.currentImage();
  }
}
```

app.html

App.html crea un encabezado con una barra de navegación fija en la parte superior de la página. Utiliza Bootstrap para los estilos y Angular para la navegación. Los enlaces en la barra de navegación permiten a los usuarios navegar entre diferentes vistas (Películas, Series, Playlist) sin recargar la página, gracias al enrutamiento de Angular. El <router-outlet> es el lugar donde se renderizan estos componentes según la ruta activa.

app.component.html

Proyecto3P > src > app > app.component.html > router-outlet

```
<header data-bs-theme="dark">
  <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand" href="index.html">TecFlix</a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" da
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarCollapse">
        <ul class="navbar-nav me-auto mb-2 mb-md-0">
          <li class="nav-item">
            <a routerLink="/peliculas" routerLinkActive="active">Peliculas</a>
          </li>
          <li class="nav-item">
            <a routerLink="/series" routerLinkActive="active">Series</a>
          </li>
          <li class="nav-item">
            <a routerLink="/playlist" routerLinkActive="active">Playlist</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
</header>

<router-outlet></router-outlet>
```

app.routes

App.routes configura el enrutamiento de la aplicación Angular para que las URL específicas muestren los componentes correspondientes. También incluye redireccionamientos para manejar rutas no definidas y la raíz de la aplicación.

app.routes.ts

Proyecto3P > src > app > app.routes.ts > ...

```ts
import { Routes } from '@angular/router';
import { PeliculasComponent } from './peliculas/peliculas.component';
import { SeriesComponent } from './series/series.component';
import { PlaylistComponent } from './playlist/playlist.component';
import { MovieComponent } from './movie/movie.component';
import { SerieComponent } from './serie/serie.component';

export const routes: Routes = [
    { path: 'peliculas', component: PeliculasComponent },
    { path: 'series', component: SeriesComponent },
    { path: 'playlist', component: PlaylistComponent },
    { path: 'movie/:id', component: MovieComponent },
    { path: 'serie/:id', component: SerieComponent },
    { path: '', redirectTo: '/peliculas', pathMatch: 'full' },
    { path: '**', redirectTo: '/peliculas' }
];
```

playlist.html

Playlist.html nos ayuda a agregar películas a una lista de reproducción en la cual podemos elegir una película y agregarla a la cola de reproducción, podemos filtrar la playlist creada por director o genero, para solo mostrar la o las películas de dicho director o género, también contamos con un botón para volver a visualizar la playlist completa y otro botón para reproducir la playlist aleatoriamente, también contamos con un botón para ver o eliminar una película de la playlist.

6A ING. INF

Materia: Programación en ambiente cliente servidor

Tema: Proyecto 3er parcial: Aplicación WEB en Angular.

```
<div class="px-4 pt-5 my-5 text-center border-bottom">
    <h1 class="display-4 fw-bold text-body-emphasis">Movie Playlist</h1>
    <div class="overflow-hidden" style="max-height: 30vh;">
      <div class="container px-5">
        <img id="logo" src="/assets/images/movieLogo.png"  class="img-fluid bor
      </div>
    </div>

    <form id="formulario">
        <label for="busqueda">Filtrar Playlist por director o género: </label>
        <input type="text" id="busqueda" name="busqueda" >

        <button type="button" (click)="filterByDirector()">Director</button>
        <button type="button" (click)="filterByGenre()">Género</button>
    </form><br>
```

```
<table class="tabla" id="tablaPeliculas">
    <thead>
      <tr>
        <th >Título</th>
        <th >Director</th>
        <th >Géneros</th>
        <th >Año</th>
        <th >Duración</th>
        <th ></th>
      </tr>
    </thead>
    <tbody >

    </tbody>
</table>
```

```
  <button type="button" (click)="insertMovie()">Ver Playlist Completa</button><br><br>
  <button type="button" (click)="randomMovie()">Película de Playlist Aleatoria</button><br>

  <br>
  <form id="formulario2">
      <label for="peliculasSelector">Agregar película:</label>
      <select id="peliculasSelector" name="peliculasSelector">
          <option *ngFor="let movie of movies" [value]="movie.id">{{ movie.title }}</option>
        </select>
        <button type="button" (click)="addMovieToTable()">Agregar Película</button>
    </form>


  <br><br><br><br><br>
</div>
```

playlist.ts

Este código es un componente de Angular que forma parte de una aplicación para gestionar una lista de reproducción de películas. Utiliza el servicio MovieService para obtener los datos de la película a partir de su ID, se utiliza el addMovieToTable elcual Añade una película seleccionada a la lista de reproducción si no está ya incluida. El filterByDirector nos filtra las películas en la lista de reproducción por director y actualiza la tabla con los resultados. El insertMovie nos actualiza la tabla HTML con la lista de reproducción actual. Añade botones para ver y eliminar cada película. Con el directorTable nos actualiza la tabla HTML con las películas filtradas por director. genreTable actualiza la tabla HTML con las películas filtradas por género. Después el verPeli nos abre una nueva ventana del navegador con el enlace de la película seleccionada y muestra una alerta con el título. Por ultimo randomMovie Selecciona una película aleatoria de la lista de reproducción y la reproduce.

```
export class PlaylistComponent implements OnInit {
  movies: any[] = [];
  addedMovies: string[] = [];
  myPlayList: any[] = [];
  moviesDirector: any[] = [];
  moviesGenre: any[] = [];

  constructor(private movieService: MovieService) { }

  ngOnInit(): void {
    this.movieService.getAllMovies().subscribe((data: any) => {
      this.movies = Object.keys(data).map(key => ({ id: key, ...data[key] }));
    });
  }
```

```
addMovieToTable(): void {
  const selectElement = document.getElementById('peliculasSelector') as HTMLSelectElement;
  const selectedMovieId = selectElement.value;
  const selectedMovie = this.movies.find(movie => movie.id === selectedMovieId);

  if (selectedMovie && !this.addedMovies.includes(selectedMovieId)) {
    this.addedMovies.push(selectedMovieId);
    this.myPlayList.push(selectedMovie);

    this.insertMovie();
  } else {
    alert("¡YA TIENES ESA PELÍCULA EN LA LISTA!");
  }
}
```

```
filterByDirector(): void {
  this.moviesDirector = [];

  const directorInput = (document.getElementById('busqueda') as HTMLInputElement).value.toLowerCase();
  this.moviesDirector = this.myPlayList.filter(movie => movie.director.toLowerCase().includes(directorInput))

  if (this.moviesDirector.length > 0) {
    this.directorTable();
  } else {
    alert("Director: " + directorInput + ", no encontrado o nombre mal escrito.");
  }
}
```

```
filterByGenre(): void {
  this.moviesGenre = [];

  const genreInput = (document.getElementById('busqueda') as HTMLInputElement).value.toLowerCase();
  this.moviesGenre = this.myPlayList.filter(movie => movie.genres.toLowerCase().includes(genreInput));

  if (this.moviesGenre.length > 0) {
    this.genreTable();
  } else {
    alert("Género: " + genreInput + ", no encontrado o mal escrito.");
  }
}
```

```
insertMovie(): void {
  const tableElement = document.getElementById('tablaPeliculas');
  if (tableElement) {
    const tableBody = tableElement.getElementsByTagName('tbody')[0];
    if (tableBody) {
      tableBody.innerHTML = '';
      this.myPlayList.forEach(movie => {
        const newRow = tableBody.insertRow();
        newRow.insertCell(0).textContent = movie.title;
        newRow.insertCell(1).textContent = movie.director;
        newRow.insertCell(2).textContent = movie.genres;
        newRow.insertCell(3).textContent = movie.year;
        newRow.insertCell(4).textContent = movie.duration;

        const viewButton = document.createElement('button');
        viewButton.textContent = 'Ver';
        viewButton.addEventListener('click', () => {
          this.verPeli(movie.title, movie.linkWatch);
        });
        newRow.insertCell(5).appendChild(viewButton);

        const deleteButton = document.createElement('button');
        deleteButton.textContent = 'Eliminar';
        deleteButton.addEventListener('click', () => {
          newRow.remove();
          const index = this.addedMovies.indexOf(movie);
          if (index !== -1) {
            this.addedMovies.splice(index, 1);
          }
        });
        newRow.insertCell(6).appendChild(deleteButton);
```

6A ING. INF

Materia: Programación en ambiente cliente servidor

Tema: Proyecto 3er parcial: Aplicación WEB en Angular.

```
directorTable(): void {
  const tableElement = document.getElementById('tablaPeliculas');
  if (tableElement) {
    const tableBody = tableElement.getElementsByTagName('tbody')[0];
    if (tableBody) {
      tableBody.innerHTML = '';
      this.moviesDirector.forEach(movie => {
        const newRow = tableBody.insertRow();
        newRow.insertCell(0).textContent = movie.title;
        newRow.insertCell(1).textContent = movie.director;
        newRow.insertCell(2).textContent = movie.genres;
        newRow.insertCell(3).textContent = movie.year;
        newRow.insertCell(4).textContent = movie.duration;

        const viewButton = document.createElement('button');
        viewButton.textContent = 'Ver';
        viewButton.addEventListener('click', () => {
          this.verPeli(movie.title, movie.linkWatch);
        });
        newRow.insertCell(5).appendChild(viewButton);
      });
    }
  }
}
```

```
genreTable(): void {
  const tableElement = document.getElementById('tablaPeliculas');
  if (tableElement) {
    const tableBody = tableElement.getElementsByTagName('tbody')[0];
    if (tableBody) {
      tableBody.innerHTML = '';
      this.moviesGenre.forEach(movie => {
        const newRow = tableBody.insertRow();
        newRow.insertCell(0).textContent = movie.title;
        newRow.insertCell(1).textContent = movie.director;
        newRow.insertCell(2).textContent = movie.genres;
        newRow.insertCell(3).textContent = movie.year;
        newRow.insertCell(4).textContent = movie.duration;

        const viewButton = document.createElement('button');
        viewButton.textContent = 'Ver';
        viewButton.addEventListener('click', () => {
          this.verPeli(movie.title, movie.linkWatch);
        });
        newRow.insertCell(5).appendChild(viewButton);
      });
    }
  }
}
```

```
verPeli(titulo: string, link: string): void {
  alert("REPRODUCIENDO:\n"+titulo);
  window.open(link, '_blank');
}

randomMovie(): void {
  const tabla = document.getElementById('tablaPeliculas') as HTMLTableElement;
  const cuerpoTabla = tabla.tBodies[0]; // Accedemos al tbody de la tabla
  const FilasPlaylist = cuerpoTabla.rows.length; // Vemos cantidad de filas
  // Igual se puede verificar si hay películas agregadas verificando si el arr

  if (FilasPlaylist === 0) { // Si no hay filas
      alert('¡La playlist está vacía!'); // Mostrar mensaje
      return;  // Nos salimos
  } else {  // Si hay filas
    const RandomMovie = Math.floor(Math.random() * FilasPlaylist); // Hacemos
    // alert("Random" + RandomMovie)
    const datospelicula = this.myPlayList[RandomMovie]; // Accedemos a los dat
    const link = datospelicula.linkWatch; // Obtenemos el link
    const titulo = datospelicula.title;
    this.verPeli(titulo, link);
  }
}
```

peliculas.html

Películas.html nos ayuda a visualizar los titulos de las películas, duración, y géneros, también se tiene un botón de ver más para ver la descripción de la película, la película y su trailer, al igual que un carrusel donde se muestran las películas sugeridas o en tendencia, en dicho carrusel se tienen los botones de ver más (en dónde se muestra más información de la película) y el botón ver trailer (dónde te dirijira al trailer de la película en cuestión)

```
<div id="myCarousel" class="carousel slide mb-6" data-bs-ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="0" class="active" aria-cur
    <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="1" aria-label="Slide 2"></
    <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="2" aria-label="Slide 3"></
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active" id="panel1">
      <img src="assets/images/dunetwo.jpg" class="d-block w-100" alt="...">
      <svg class="bd-placeholder-img" width="100%" height="100%" xmlns="http://www.w3.org/2000/svg"
      <div class="container">
        <div class="carousel-caption text-end">
          <h1 style="color: white;" id="hero">Dune Parte Dos</h1>
          <p style="color: white;" id="hero1">115min. Accion, aventura y ciencia ficcion</p>
          <p style="display: inline-block; margin-right: 10px;"><a class="btn btn-lg btn-primary" [
          <p style="display: inline-block;"><a class="btn btn-lg btn-primary" target="_blank" href=

        </div>
      </div>
    </div>
```

```
<main >
  <div class="d-md-flex flex-md-equal w-100 my-md-3 ps-md-3" >
    <div class="text-bg-dark me-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center over
      <div class="my-3 py-3">
        <h2 class="display-5">Jack In The Box 3: El Ascenso</h2>
        <p class="lead">132m. Terror.</p>
      </div>
      <div class="bg-body-tertiary shadow-sm mx-auto" style="width: 80%; height
        <br>
        <a class="btn btn-lg btn-primary" [routerLink]="['/movie/', 'jackintheb
    </div>
```

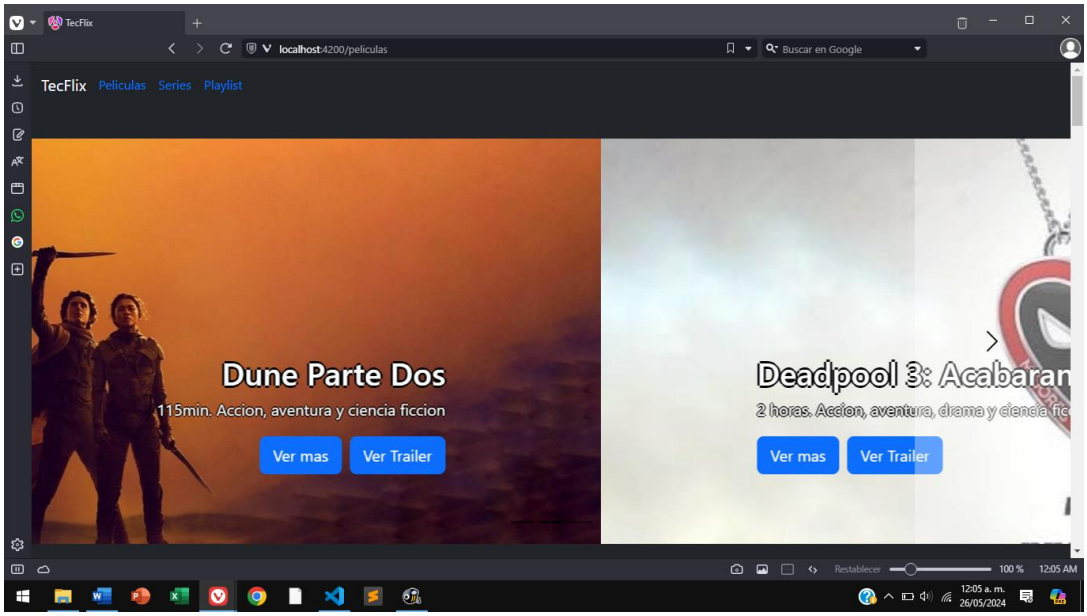
series.html

Series.html nos ayuda a visualizar los titulos de las series, temporadas, y géneros, también se tiene un botón de ver más para ver la descripción de la serie, la serie y su trailer, al igual que un carrusel donde se muestran las series que serán agregadas próximamente en dicho carrusel se tienen los botones de ver más (en dónde se muestra más información de la serie) y el botón ver trailer (dónde te dirijira al trailer de la serie en cuestión)

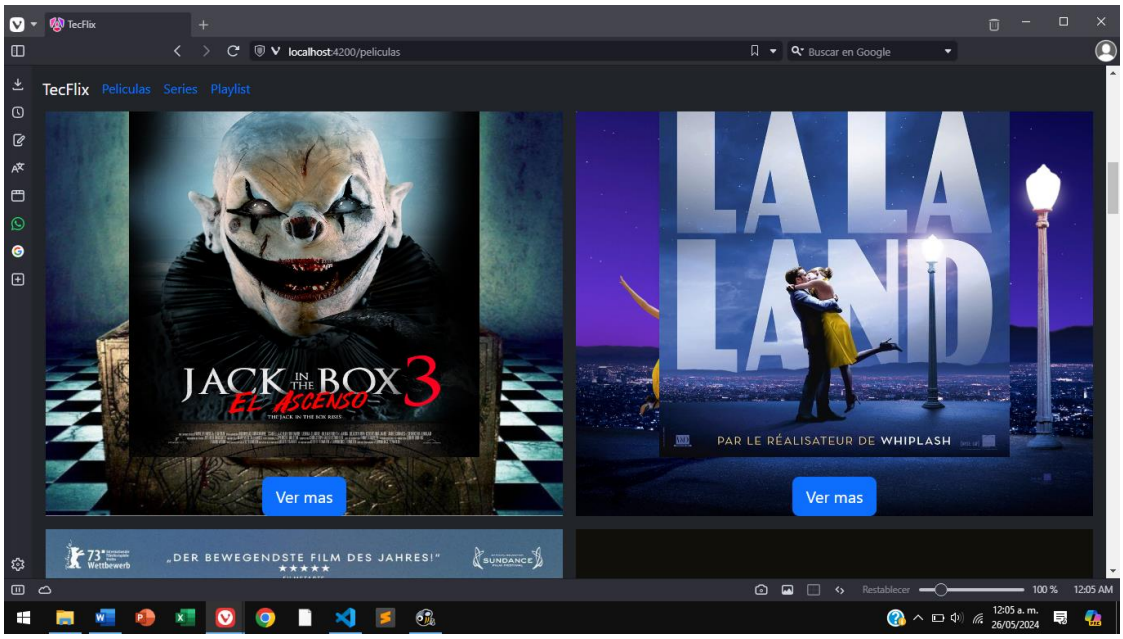
```
<div id="myCarousel" class="carousel slide mb-6" data-bs-ride="carousel">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="0" class="active" aria-cur
    <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="1" aria-label="Slide 2"></
    <button type="button" data-bs-target="#myCarousel" data-bs-slide-to="2" aria-label="Slide 3"></
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="/assets/images/truedetective.jpg" class="d-block w-100" alt="...">
      <svg class="bd-placeholder-img" width="100%" height="100%" xmlns="http://www.w3.org/2000/svg"
      <div class="container">
        <div class="carousel-caption text-start">
          <h1 style="color: white;" id="hero">True Detective</h1>
          <p style="color: white;" id="hero1">Proximamente. Drama, Misterio, Acción</p>
          <p style="display: inline-block; margin-right: 10px;"><a class="btn btn-lg btn-primary" [
          <p style="display: inline-block;"><a class="btn btn-lg btn-primary" target="_blank" href=

        </div>
      </div>
    </div>
```

```
<main>
  <div class="d-md-flex flex-md-equal w-100 my-md-3 ps-md-3" >
    <div class="text-bg-dark me-md-3 pt-3 px-3 pt-md-5 px-md-5 text-center overflow-hidden"
      <div class="my-3 py-3">
        <h2 class="display-5">Dark</h2>
        <p class="lead">Temporadas: 3, Suspenso, Misterio, Thriller.</p>
      </div>
      <div class="bg-body-tertiary shadow-sm mx-auto" style="width: 80%; height: 500px; bor
        <br>
        <a class="btn btn-lg btn-primary" [routerLink]="['/serie/', 'dark']">Ver mas</a>
    </div>
```

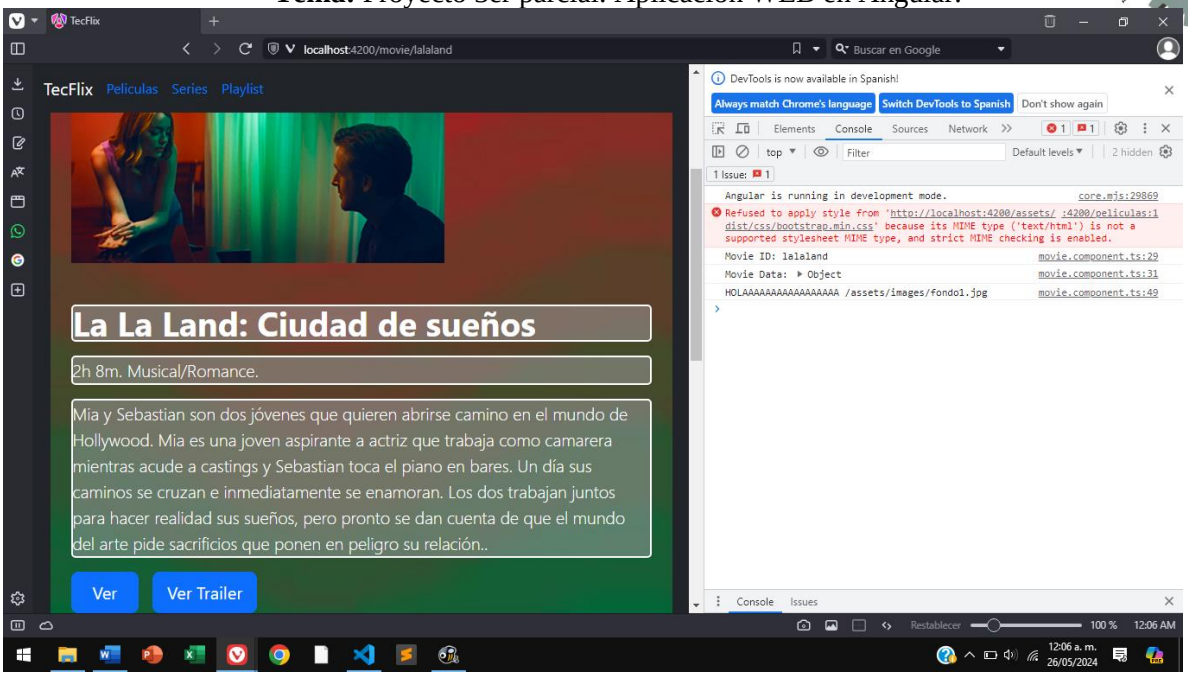
5. Reporte de resultados



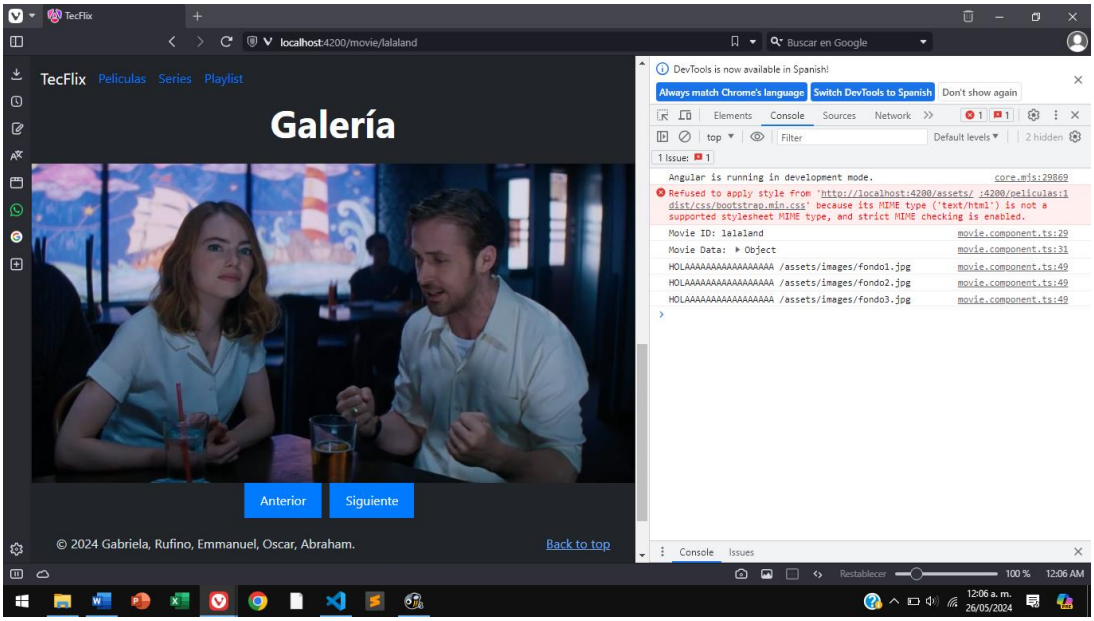
Este es el carrusel de imágenes de la página principal de película.



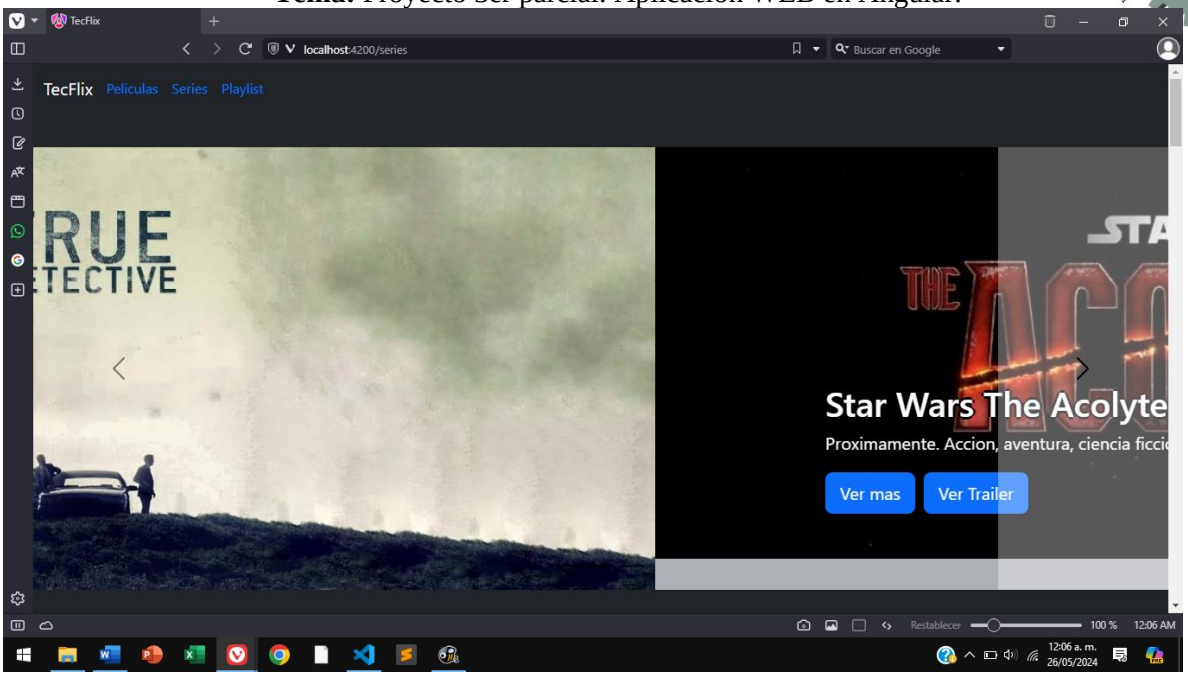
Si bajamos un poco, podemos ver todas las películas agregadas a la página.



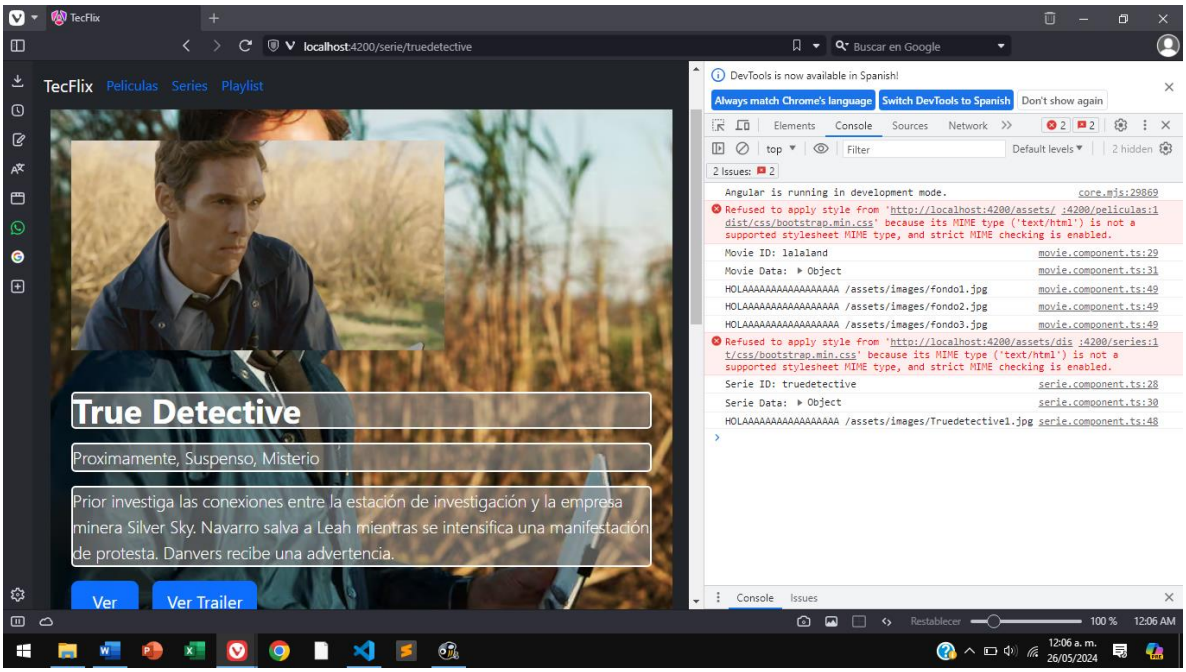
Cuando hacemos clic en una, consumimos la API que creamos. En la consola del navegador se imprimen algunos datos.



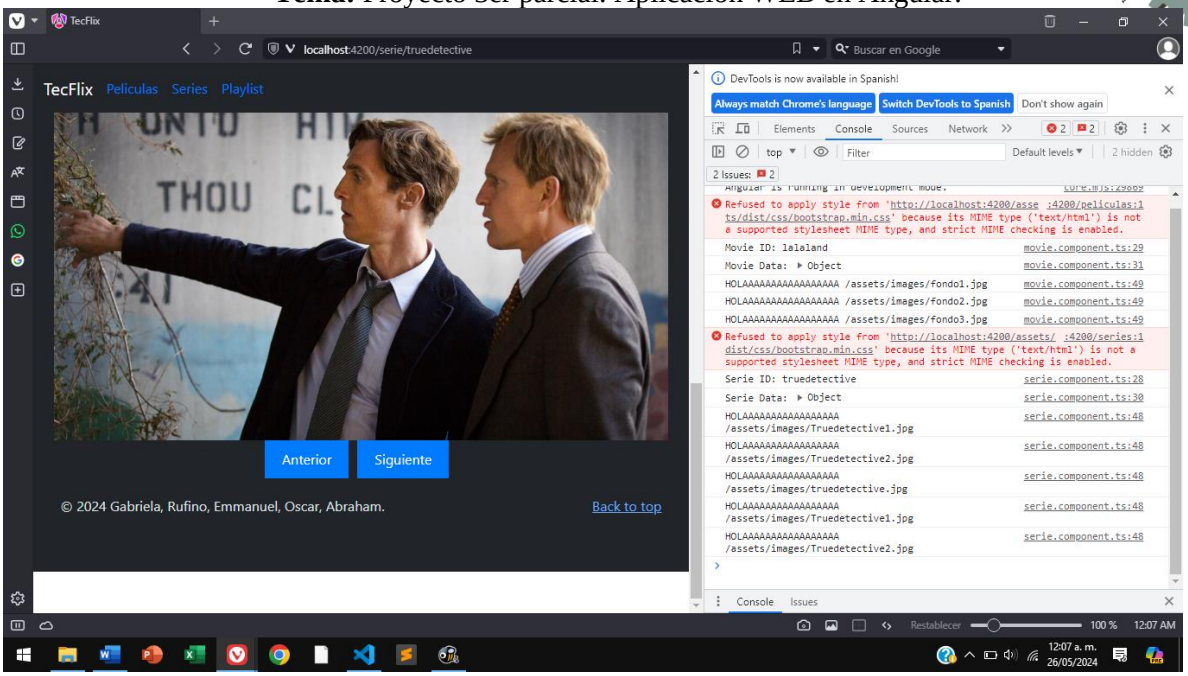
Para el carrusel de imágenes de JQuery (ahora en typeScript) se imprime la dirección de las imágenes.



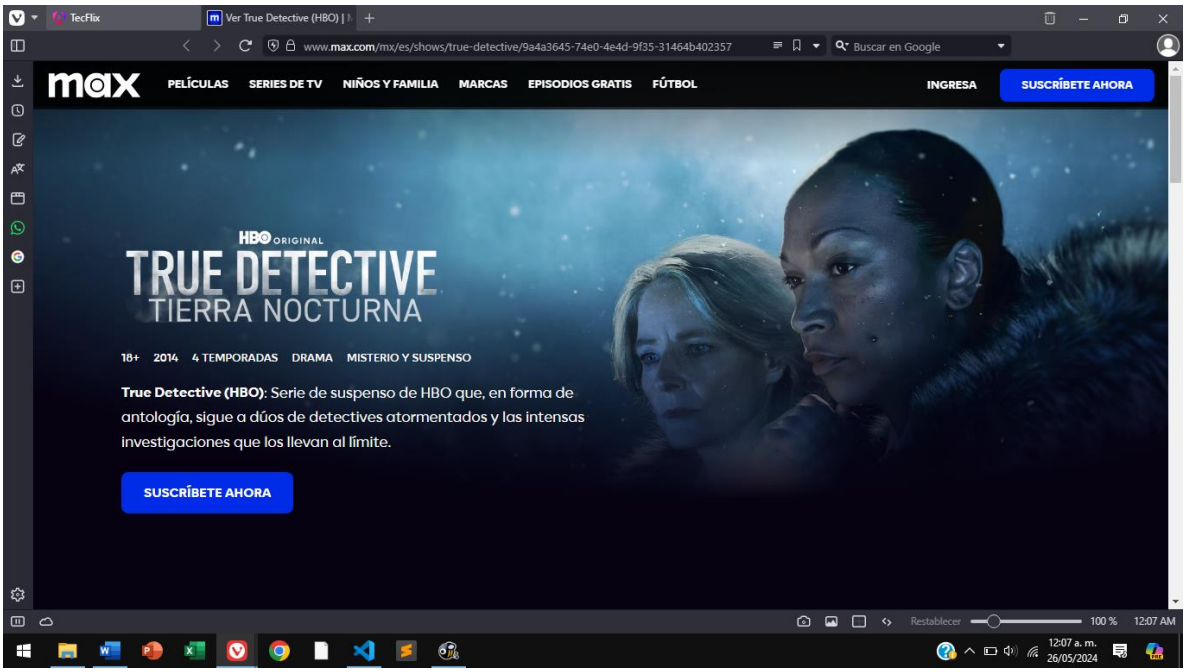
Este es el carrusel de imágenes del apartado de series.



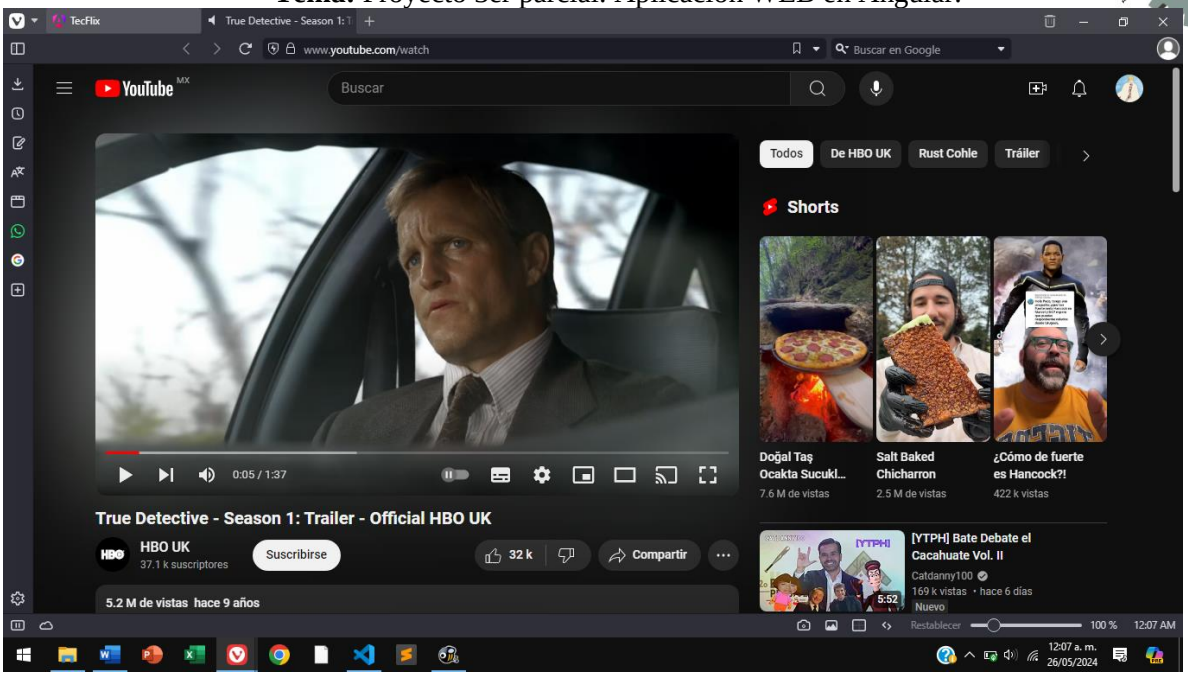
De igual forma, cuando vemos una serie se consume la API.



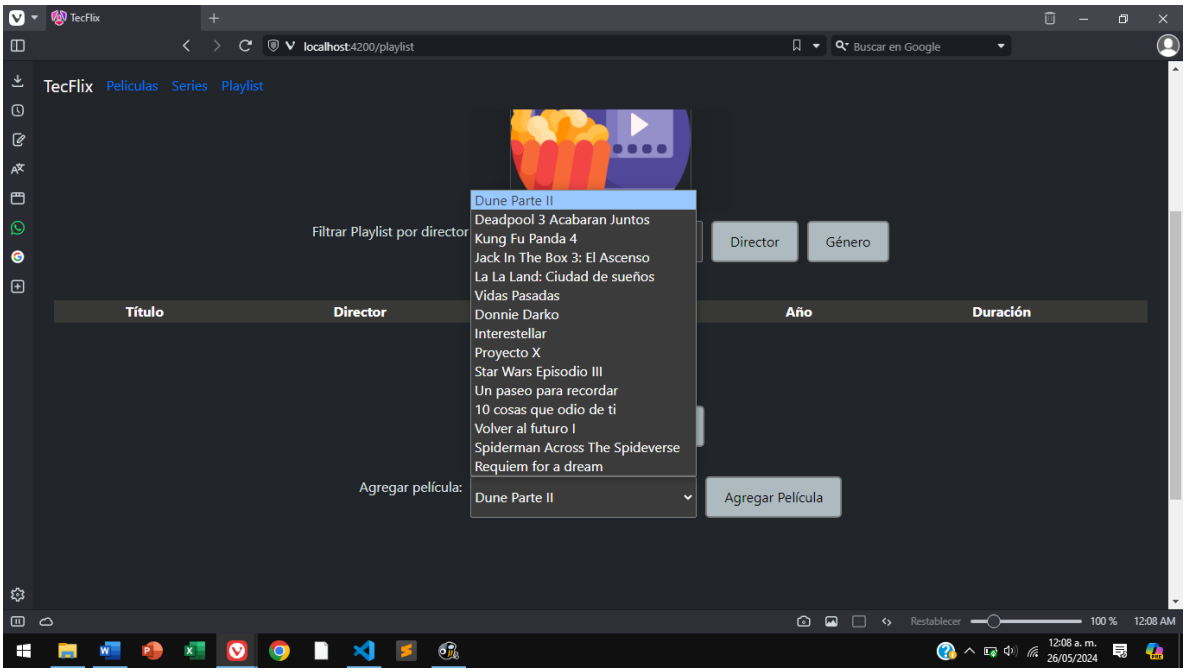
Y el carrusel de imágenes imprime la dirección de la imagen.



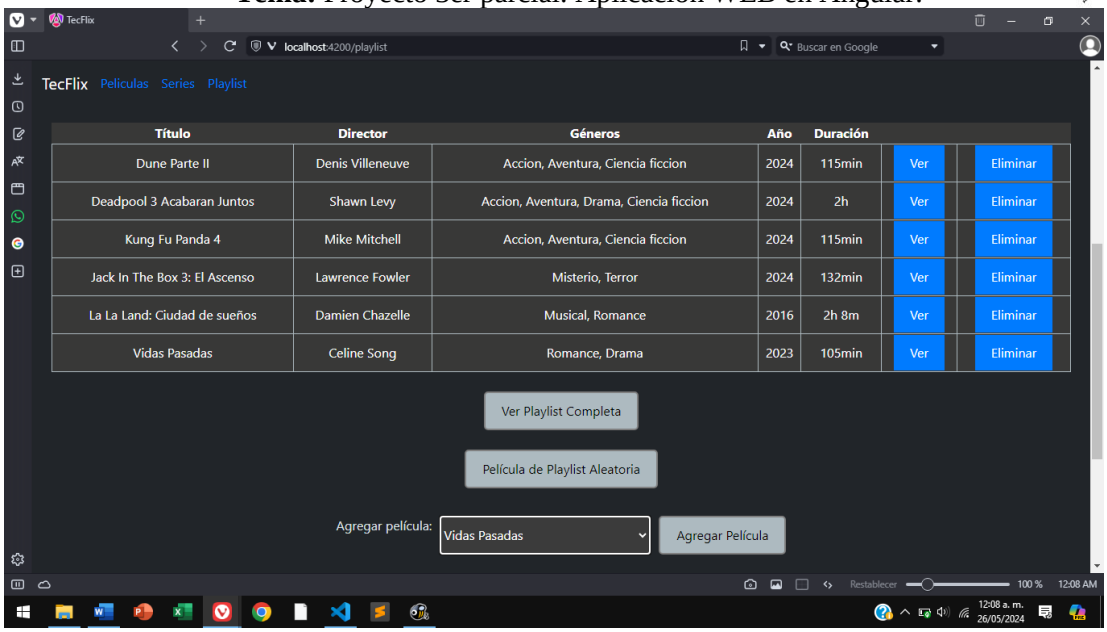
Cuando hacemos clic en “Ver”, extraemos el link de la API y nos reenvía al mismo.



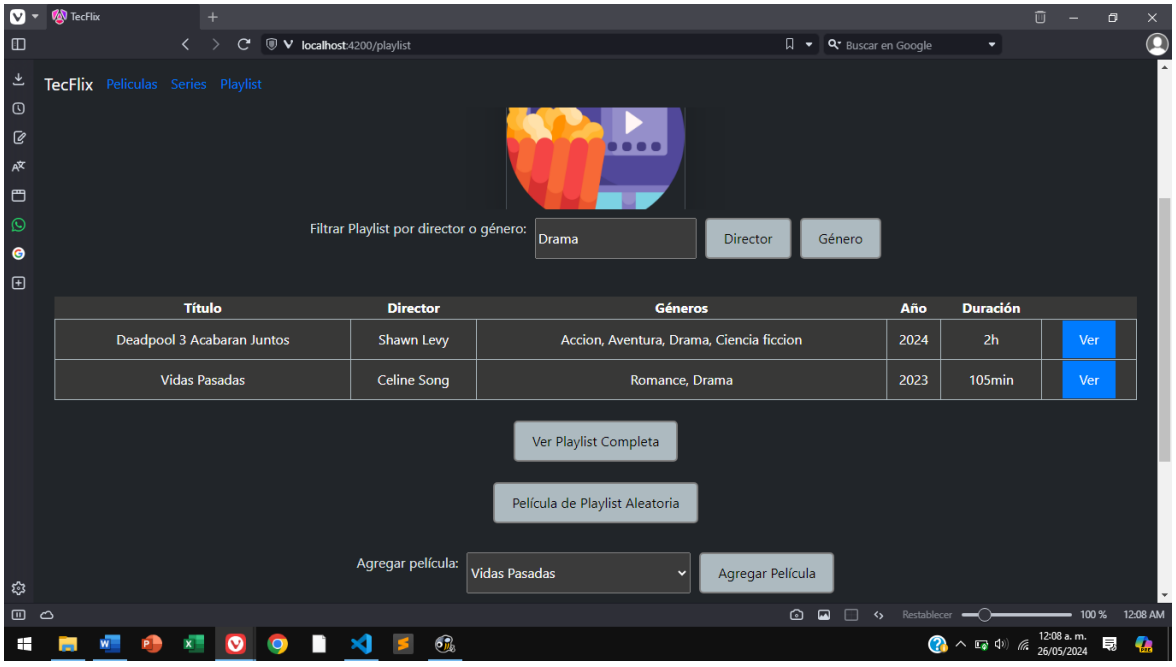
También podemos ver el tráiler con el link de la API.



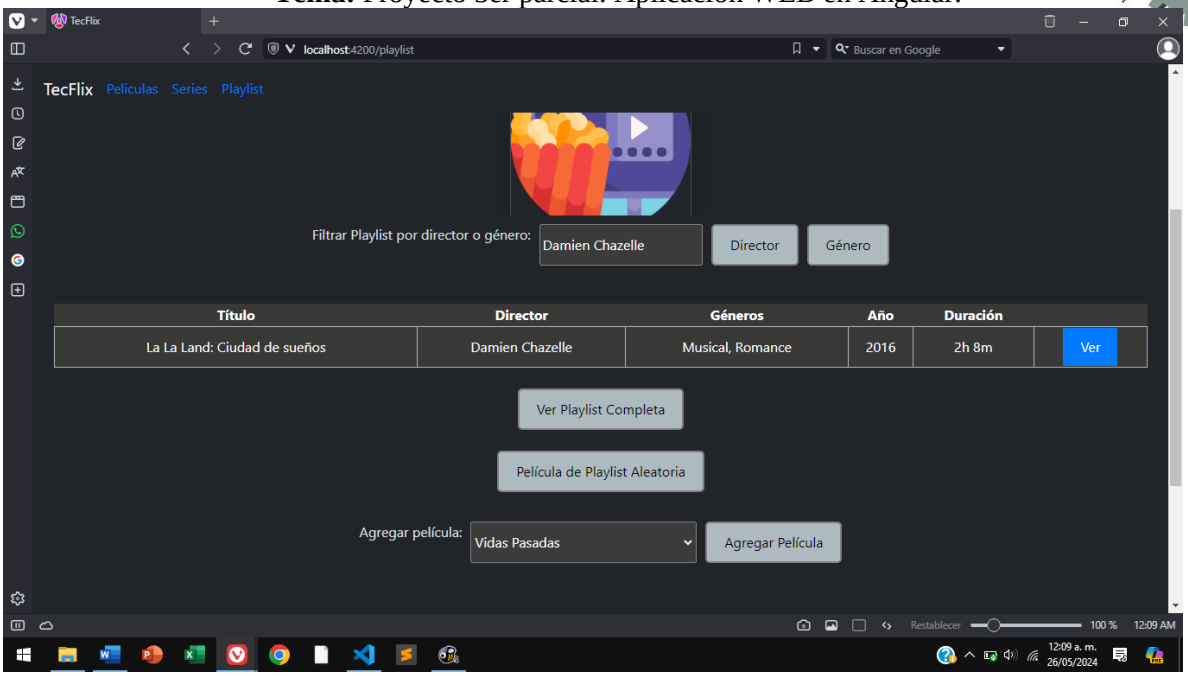
Para el apartado de la PlayList temenos esta página, el combo se llena con el ID de las películas en la API.



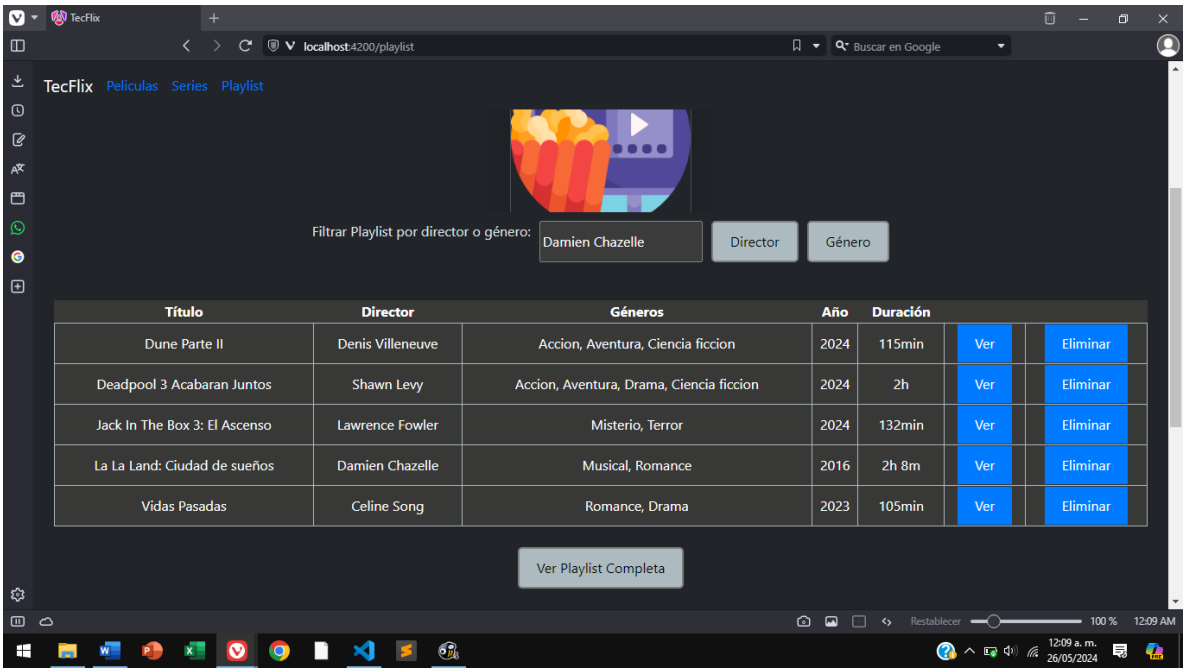
Agregamos varias películas.



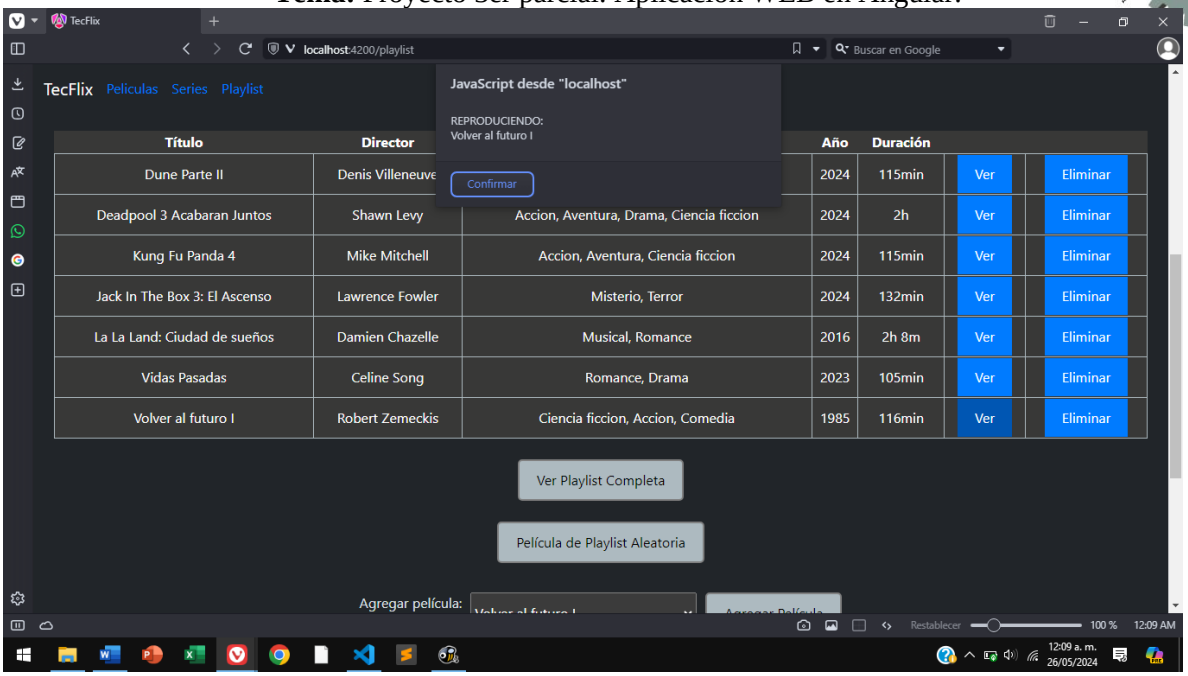
Buscamos las películas en la lista con el género “Drama” y nos arroja dos resultados.



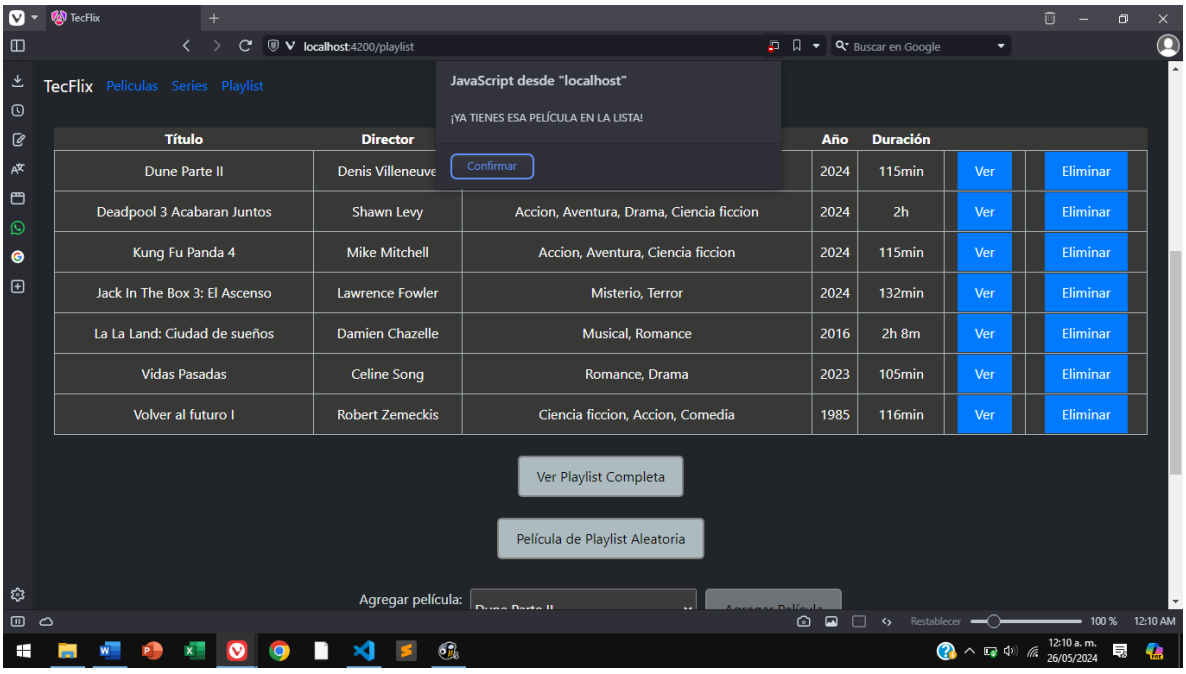
Buscamos una película por director y nos arroja un resultado.



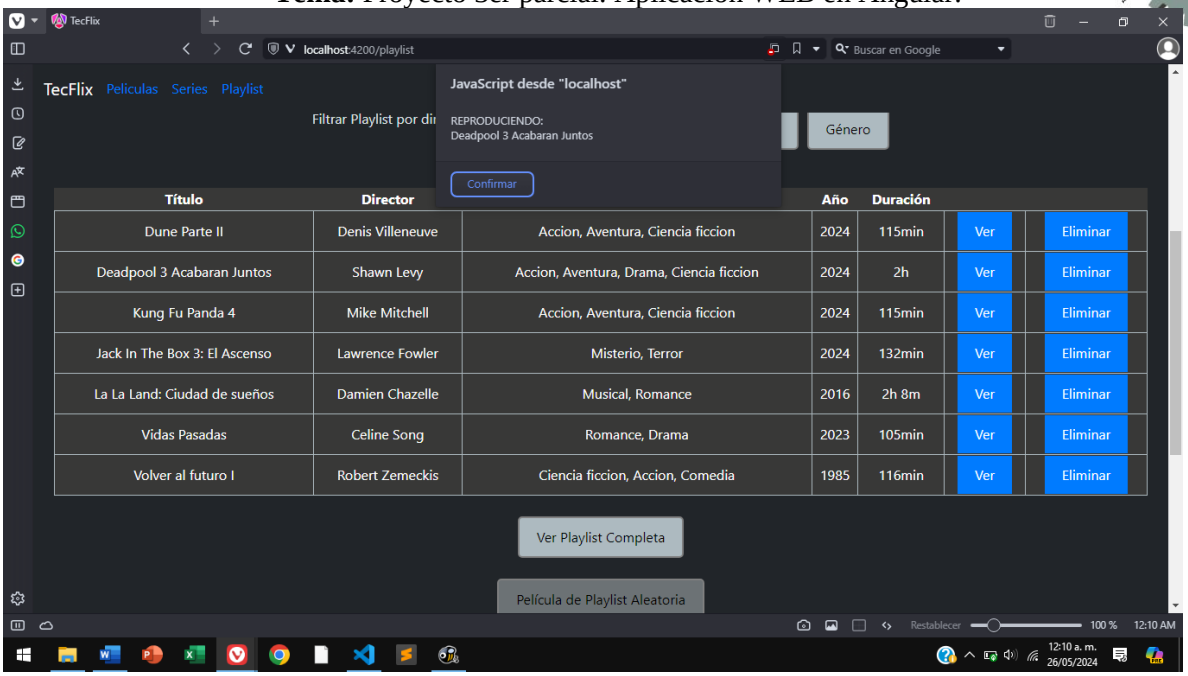
Aquí eliminamos una película (Kung Fu Panda 4) y la lista se recorre.



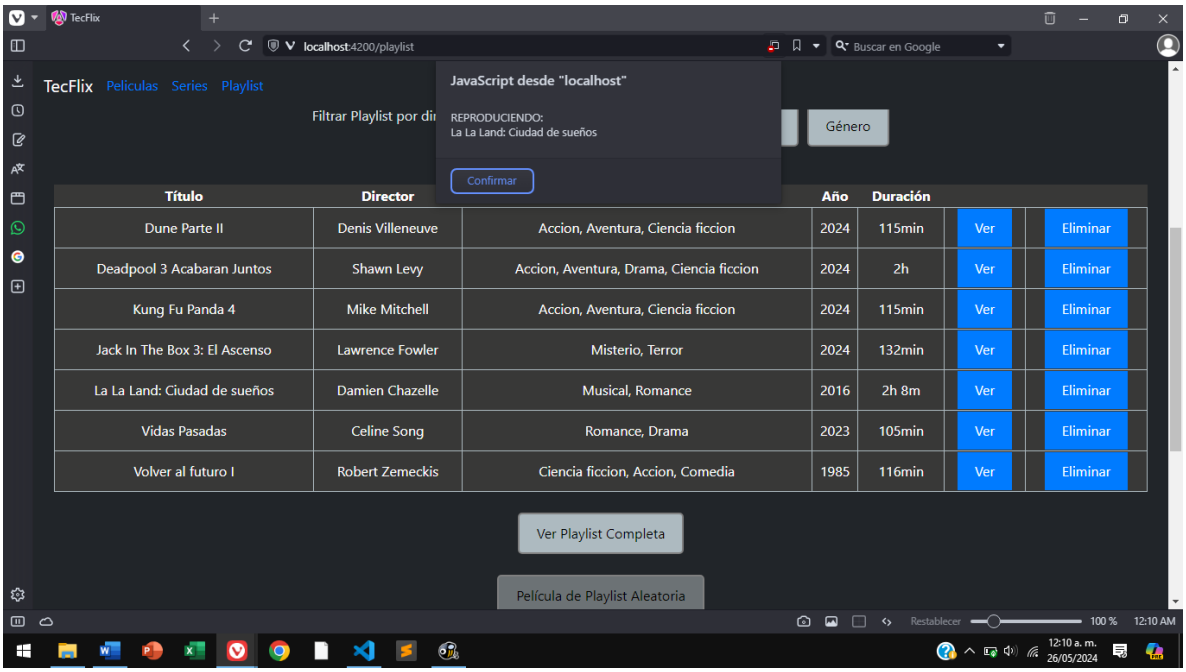
Reproducimos una película y se nos arroja un mensaje de la película que vamos a ver.



Probamos que no se puedan repetir películas en la lista.



Probamos las películas aleatorias, se reproduce Deadpool 3.



Y ahora se reproduce La La Land.

6. Análisis de resultados

Como podemos observar la página web funciona correctamente, incluyendo el traspaso de componentes a componentes, y el consumo y respuesta de la API para mostrar la información de las películas.

7. Conclusión

En conclusión, tras haber realizado una investigación sobre Angular y la creación y consumo de API, pudímos crear con éxito una página WEB con el Cli de Angular. En general, pudimos traspasar las páginas de HTML a componentes de Angular reduciendo a gran escala la cantidad de elementos y la carga en la página. De igual forma, el lenguaje de TypeScript vuelve más entendibles los códigos y maneja eventos de una manera más estructura que JS, la API nos permitió emigrar información de las películas y series a un archivo en el cual solo se hacían consultas, tanto para los componentes de movie y serie, como para el carrusel de imágenes y para imprimir la información en la playlist.

8. Bibliografía

- Álvarez, M. Á. (01 de marzo de 2001). *Desarrollo Web*. Obtenido de Desarrollo Web: https://desarrolloweb.com/articulos/que-es-html.html
- Amazon. (diciembre de 2023). *Amazon*. Obtenido de Amazon: https://aws.amazon.com/es/what-is/api/
- Angular. (04 de marzo de 2022). *Angular*. Obtenido de Angular: https://docs.angular.lat/guide/architecture
- González, M. (28 de julio de 2023). *HubSpot*. Obtenido de HubSpot: https://blog.hubspot.es/website/que-es-css
- Kinsta. (19 de diciembre de 2022). *Kinsta*. Obtenido de Kinsta: https://kinsta.com/es/base-de-conocimiento/que-es-express/

- *Kinsta*. (12 de junio de 2023). Obtenido de Kinsta: https://kinsta.com/es/base-de-conocimiento/que-es-typescript/
- Salazar, M. Á. (21 de junio de 2023). *We Are Marketing*. Obtenido de We Are Marketing: https://www.wearemarketing.com/es/blog/frameworks-en-el-desarrollo-web-las-mejores-practicas-para-tu-negocio-online.html