🎬 EXAMEN: MovieFlix - Tu Buscador de Películas

- Añade tu nombre y apellidos a **TODOS** los ficheros que entregues de javascript

🌐 API Local Disponible

Durante el examen tendrás acceso a una API REST local:

- **URL**: `http://localhost:1492/api/movies`
- **Método**: GET
- **Respuesta**: JSON con array de 200 películas

📁 Estructura de cada película (JSON):

```
{
  "id": 1084242,
  "title": "Zootrópolis 2",
  "original_title": "Zootopia 2",
  "overview": "Descripción de la película...",
  "poster_path": "/images/poster.jpg",
  "backdrop_path": "/images/backdrop.jpg",
  "release_date": "2025-11-26",
  "vote_average": 7.7,
  "vote_count": 280,
  "popularity": 577.2135,
  "genre_ids": [16, 10751, 35, 12, 9648]
}
```

🏗️ ESTRUCTURA DEL PROYECTO

Debes crear un proyecto con **Vite + JavaScript Vanilla** con esta estructura:

```
examen-javascript/
├── index.html          ← Raíz del proyecto
├── package.json        ← Generado por Vite
└── src/                ← Todo tu código aquí
```

```
├── main.js           ← Entry point
├── app.js            ← Cerebro de la aplicación
├── style.css         ← Proporcionado por el profesor
├── services/
│   └── movieService.js
├── components/
│   ├── MovieCard.js
│   └── MovieList.js
└── utils/
    └── storage.js
```

📦 ARCHIVOS PROPORCIONADOS

Solo recibirás:

- ✅ Este enunciado
- ✅ `style.css` (con comentarios de qué clase usar en cada elemento)

🚫 RESTRICCIONES OBLIGATORIAS

Prohibido:

- ❌ Usar frameworks o librerías (jQuery, React, etc.)
- ❌ Añade tu nombre y apellidos a **TODOS** los ficheros que entregues de javascript
- ❌ Hacer múltiples peticiones fetch (solo 1 al inicio)

Obligatorio:

- ✅ **JavaScript Vanilla puro**
- ✅ `createElement()` **y** `appendChild()` para crear elementos DOM
- ✅ **Promesas** para peticiones API
- ✅ **Caché de datos en memoria** (1 sola petición)
- ✅ **Factory Pattern** en componentes
- ✅ **Closures** para encapsular lógica

📝 PARTES DEL EXAMEN

🔧 Parte 1: Configuración inicial con Vite (0.5 puntos)

Tareas:

1. **Crear proyecto Vite:**
   - Ejecutar `npm create vite@latest examen-javascript`
   - Seleccionar: **Vanilla** + **JavaScript**
   - Instalar dependencias
   - Crear un script para levantar el proyecto llamado "examen-javascript" en package.json
   - La estructura de carpetas es la que se indica en el enunciado.
   - El archivo index.html solo debe tener un contenedor vacío: `<div id="app"></div>`
   - El archivo main.js debe importar el CSS y ejecutar createApp().

🌐 Parte 2: Servicio API con Caché (2 puntos)

Archivo: `src/services/movieService.js`

Concepto clave: CACHÉ EN MEMORIA

⚠ **MUY IMPORTANTE:** Para evitar saturar la API, debes:

- Hacer **UNA ÚNICA petición fetch** al cargar la aplicación
- Guardar las películas en **una variable dentro de un closure**
- Todas las búsquedas/filtros deben trabajar sobre ese array en memoria

Funcionalidades requeridas:

1. **Variable privada** (dentro del closure) que almacene todas las películas
2. **Función** `fetchAllMovies()`**:**
   - Hace fetch a la API **UNA SOLA VEZ**
3. **Función** `getMovies()`**:**
   - Devuelve las películas **desde memoria** (NO hace fetch)
4. **Función** `getMovieById(id)`**:**
   - Busca y retorna una película por ID **desde memoria**
5. **Usar patrón Pattern Factory o Closure** para encapsular

📱 Parte 3: main.js - Entry Point (0.5 puntos)

Archivo: `src/main.js`

Funcionalidad:

Este archivo debe ser **MUY simple**:

- Importar el CSS y el archivo app.js

🧠 Parte 4: app.js - Cerebro de la Aplicación (2.5 puntos)

Archivo: `src/app.js`

Responsabilidades:

1. **Crear toda la estructura HTML con JavaScript** (0.5 pts)
    - Header con título "MovieFlix y tu NOMBRE"
    - Contenedor principal
    - Contenedor para input de búsqueda
    - Contenedor para select de ordenación
    - Contenedor para las películas
2. **Crear input de búsqueda** (0.5 pts)
    - Tipo text
    - Placeholder adecuado
    - Aplicar clase CSS correspondiente
3. **Crear select de ordenación** (0.5 pts)
    - Opciones:
        - Sin ordenar (por defecto)
        - Título A-Z
        - Título Z-A
        - Valoración (más alta primero)
    - Aplicar clase CSS correspondiente
4. **Cargar datos iniciales** (0.25 pts)
5. **Inicializar componentes** (0.25 pts)
6. **Gestionar estado de la aplicación** (0.25 pts)
    - Variable que guarda el filtro de búsqueda actual
    - Variable que guarda la ordenación actual
7. **Función para aplicar filtros + ordenación** (0.5 pts)

- Obtener películas desde el caché
- Filtrar por título si hay búsqueda
- Ordenar según selección
- Renderizar con MovieList
- Restaurar estado de películas vistas (localStorage)

8. **Event listeners** (0.5 pts)
   - Input de búsqueda
   - Select de ordenación

Funcionamiento esperado:

- **Búsqueda:** Filtrar películas en tiempo real por título
- **Ordenación:** Cambiar orden según criterio seleccionado
- **Combinación:** Ambos filtros deben poder aplicarse simultáneamente
- **Caché:** Todo debe funcionar sobre los datos en CACHE

Criterios de evaluación:

- ✅ Estructura HTML completa creada con JavaScript y manejo del DOM
- ✅ Todos los elementos tienen las clases CSS correctas
- ✅ Input y select funcionales
- ✅ Búsqueda filtra en tiempo real
- ✅ Ordenación aplica los 4 criterios
- ✅ Filtros se pueden combinar
- ✅ NO se hacen peticiones fetch al filtrar
- ✅ Código organizado y legible

🎴 Parte 5: MovieCard - Componente Tarjeta (2 puntos)

Archivo: `src/components/MovieCard.js`

Patrón: Factory Function

Debe devolver un objeto con método `create(movie)` que genere una tarjeta.

Funcionalidades:

1. **Crear estructura de tarjeta** (1 punto)
   - Contenedor principal con clase `movie-card`

- Imagen del póster (`poster_path`)
- Contenedor de información
- Título de la película
- Valoración formateada (ej: "⭐ 7.7")
- Fecha de estreno
- Atributo `data-movie-id` con el ID de la película (IMPORTANTE)

2. **Gestionar eventos interactivos** (1 punto)

   Click izquierdo (0.3 pts):

   - Toggle clase `movie-watched` (difumina la película)
   - Guardar/quitar ID en localStorage

   Click derecho (0.4 pts):

   - Prevenir menú contextual del navegador
   - Quitar clase `movie-watched`
   - Actualizar localStorage

   Doble click (0.3 pts):

   - Eliminar tarjeta del DOM completamente

Estructura DOM esperada de cada tarjeta:

```
<div class="movie-card" data-movie-id="123">
  <img class="movie-poster" src="..." alt="...">
  <div class="movie-info">
    <h3 class="movie-title">Título</h3>
    <div class="movie-rating">⭐ 7.7</div>
    <p class="movie-date">2025-11-26</p>
  </div>
</div>
```

📚 Parte 6: MovieList - Componente Contenedor (1 punto)

Archivo: `src/components/MovieList.js`

Patrón: Closure

Debe usar closure para encapsular la lógica del contenedor.

Funcionalidades:

1. **Método** `render(movies)` (0.6 pts)
   - Limpiar contenedor
   - Si no hay películas, mostrar mensaje "No se encontraron películas"
   - Recorrer array de películas
   - Crear cada tarjeta con `MovieCard.create()`
   - Añadir tarjetas al contenedor
2. **Método** `clear()` (0.2 pts)
   - Vaciar completamente el contenedor
3. **Gestión del contenedor** (0.2 pts)
   - Obtener referencia al contenedor por ID
   - Mantenerla en el closure

💾 Parte 7: Storage - Persistencia con LocalStorage (1 punto)

Archivo: `src/utils/storage.js`

Funcionalidad:

Sistema para persistir las películas marcadas como "vistas".

Métodos requeridos:

1. `getWatchedMovies()` (0.25 pts)
   - Obtener array de IDs desde localStorage
   - Si no existe, retornar array vacío
2. `addWatchedMovie(movieId)` (0.25 pts)
   - Añadir ID al array (si no existe ya)
   - Guardar en localStorage
3. `removeWatchedMovie(movieId)` (0.25 pts)
   - Quitar ID del array
   - Actualizar localStorage
4. `isWatched(movieId)` (0.25 pts)

- Verificar si un ID está en el array
- Retornar boolean

Datos a guardar:

```
// Ejemplo: Array de IDs de películas vistas
[123, 456, 789, 1011]
```

🎁 BONUS OPCIONAL (+0.5 puntos)

Implementa UNO o MÁS de estos extras:

A) Contador de películas (0.2 pts)

Mostrar: "Mostrando X de Y películas"

B) Botón "Limpiar filtros" (0.2 pts)

Resetear búsqueda y ordenación

C) Estadísticas (0.3 pts)

- Total de películas
- Películas vistas
- Promedio de valoración actual

D) Animaciones CSS (0.2 pts)

- Transiciones al filtrar
- Hover en tarjetas
- Animación al eliminar

📊 RESUMEN DE PUNTUACIÓN

Parte	Archivo	Puntos
1	Setup Vite + estructura	0.5
2	`movieService.js` (API + caché)	2.0
3	`main.js` (entry point)	0.5
4	`app.js` (cerebro)	2.5
5	`MovieCard.js` (tarjetas + eventos)	2.0
6	`MovieList.js` (contenedor)	1.0
7	`storage.js` (localStorage)	1.0
Bonus	Extras opcionales	+0.5
TOTAL		**10 pts**

✅ CHECKLIST PRE-ENTREGA

Verifica antes de entregar:

Funcionalidad:

- ☐ El proyecto arranca sin errores
- ☐ Se hace solo 1 petición a la API
- ☐ Las películas se muestran correctamente
- ☐ La búsqueda funciona en tiempo real
- ☐ La ordenación funciona (4 opciones)
- ☐ Click izquierdo marca/desmarca película
- ☐ Click derecho desmarca película
- ☐ Doble click elimina película
- ☐ Las películas vistas se guardan en localStorage
- ☐ Al recargar, las películas vistas siguen marcadas
- ☐ Búsqueda + ordenación funcionan combinadas

Técnico:

- ☐ Estructura de carpetas correcta
- ☐ Todos los archivos exportan/importan correctamente
- ☐ Se usa `createElement` y `appendChild`, etc.
- ☐ Se usa Factory Pattern en MovieCard
- ☐ Se usa Closure en MovieService y MovieList

- ☐ El CSS está aplicado correctamente
- ☐ No hay errores en la consola del navegador

🆘 CONSEJOS FINALES

Durante el examen:

1. **Lee TODO el enunciado** antes de empezar
2. **Empieza por lo básico**: setup → servicio → componentes
3. **Prueba cada parte** antes de continuar
4. **Usa console.log()** para depurar
5. **Revisa el style.css**: ahí están las clases que necesitas
6. **Gestiona tu tiempo**: deja bonus para el final

🎯 CONCEPTOS EVALUADOS

Este examen evalúa tu dominio de:

✅ **JavaScript Vanilla**
✅ **Closures y scope**
✅ **Factory Pattern**
✅ **Promesas y async/await**
✅ **Manipulación del DOM**
✅ **Event listeners**
✅ **Array methods** (filter, sort, map)
✅ **LocalStorage**
✅ **Módulos ES6** (import/export)
✅ **Estructura de proyecto Vite**
✅ **Arquitectura de componentes**