

# Trabajo Práctico - Sprint 2: Watchlist

---

## Objetivo

---

Desarrollar una aplicación en React + Vite donde los usuarios puedan visualizar una lista de películas y agregar o eliminar películas de su **watchlist** (lista de películas favoritas). La watchlist debe almacenarse en el **Local Storage**, de manera que no se pierda al recargar la página.

## Requerimientos

---

### 1 Configuración inicial

- Crear un nuevo proyecto en **Vite** con React.
- Instalar y configurar **Tailwind CSS** siguiendo la documentación oficial.

### 2 Estructura del Proyecto

El código debe organizarse en **componentes reutilizables**. Si quieres puedes usar una estructura similar a esta:

```
/src
├── components
│   ├── MovieCard.jsx
│   ├── WatchlistModal.jsx
│   ├── MovieList.jsx
│   ├── Header.jsx
│   └── Button.jsx
├── pages
│   └── Home.jsx
├── hooks
│   └── useWatchlist.js
├── App.jsx
└── main.jsx
```

Pero siéntete libre al hacer tu estructura de archivos

### 3 Funcionalidades Principales

#### ✓ Mostrar un listado de películas

- Renderizar una lista de películas en la pantalla principal.
- Cada película debe mostrarse en una tarjeta (**MovieCard**) con su imagen, título y un botón para agregar a la watchlist.

#### ✓ Agregar películas a la Watchlist

- Al hacer clic en “Agregar a mi lista”, la película debe almacenarse en un estado global de la aplicación ( `useState` ).
- Guardar la watchlist en **Local Storage** para que no se pierda al recargar la página.

#### ✓ Ver la Watchlist

- Implementar un **modal** o **sidebar** que se abra al hacer clic en “Ver mi lista”.
- Mostrar la lista de películas guardadas.
- Cada película en la watchlist debe tener un botón para **removerla**.

#### ✓ Persistencia de datos

- Usar `useEffect` para cargar la **watchlist desde Local Storage** cuando se monta el componente principal.

### 4 Estilos

- Utilizar **Tailwind CSS** para el diseño.
- La interfaz debe ser **responsive** y de fácil uso.



## Recomendaciones y Ayudas

---

### ◆ ¿Cómo manejar el estado de la Watchlist?

Puedes crear un estado en el componente principal ( `Home.jsx` o `App.jsx` ):

```
const [watchlist, setWatchlist] = useState([]);
```

Para agregar una película:

```
const addToWatchlist = (movie) => {
  setWatchlist([...watchlist, movie]);
  localStorage.setItem("watchlist", JSON.stringify([...watchlist, movie]));
};
```

Para eliminar una película:

```
const removeFromWatchlist = (id) => {
  const updatedList = watchlist.filter(movie => movie.id !== id);
  setWatchlist(updatedList);
  localStorage.setItem("watchlist", JSON.stringify(updatedList));
};
```

Para cargar los datos desde Local Storage cuando la página se recarga:

```
useEffect(() => {
  const savedWatchlist = JSON.parse(localStorage.getItem("watchlist")) || [
    setWatchlist(savedWatchlist);
  ], []);
```

## ◆ ¿Cómo hacer el modal para la Watchlist?

Puedes usar un `useState` para manejar la visibilidad del modal:

```
const [isModalOpen, setIsModalOpen] = useState(false);
```

Dentro del JSX:

```
<button onClick={() => setIsModalOpen(true)}>Ver mi lista</button>
{isModalOpen && <WatchlistModal watchlist={watchlist} onClose={() => setIsM
```

## ◆ ¿Cómo estructurar la información de una película?

Puedes usar un array de objetos con esta estructura:

```
const movies = [  
  { id: 1, title: "Inception", image: "URL de la imagen" },  
  { id: 2, title: "Interstellar", image: "URL de la imagen" },  
  { id: 3, title: "The Dark Knight", image: "URL de la imagen" },  
];
```

## ◆ ¿Cómo pasar props?

Cuando renderizas `MovieCard`, pásale la función `addToWatchlist` como prop:

```
<MovieCard movie={movie} onAdd={addToWatchlist} />
```




En `MovieCard.jsx`:

```
const MovieCard = ({ movie, onAdd }) => (  
  <div>  
    <h3>{movie.title}</h3>  
    <button onClick={() => onAdd(movie)}>Agregar a mi lista</button>  
  </div>  
);
```

## Evaluación

---

Se evaluará:  Correcta implementación de los componentes y uso de props. 

Manejo de estado con **useState** y efectos con **useEffect**.  Uso de **Local Storage** para persistencia de la watchlist.  Diseño coherente y estilos con **Tailwind CSS**. 

Implementación de la lógica para agregar y eliminar películas (realizadas en un Hooks personalizado).

---

 **¡Manos a la obra!** Recuerden que pueden usar la plantilla HTML como referencia:

[Watchlist en HTML](#)

Cualquier duda, ¡pregunten en clase! 