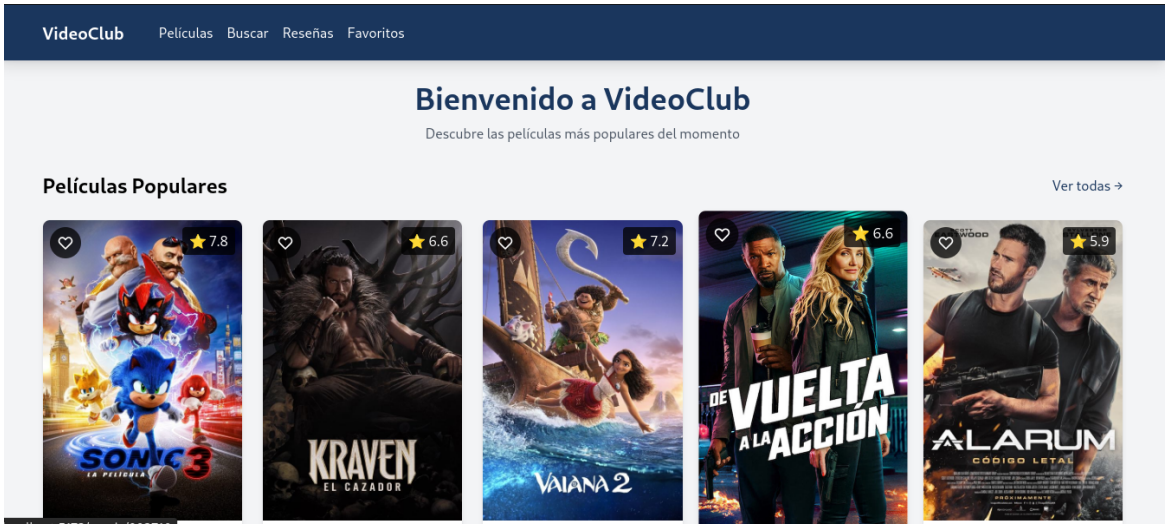
📻 ¡Tu Videoclub con React! 🍿



Objetivo: Construir una aplicación web de catálogo de películas utilizando React, la API de The Movie Database (TMDB), y las mejores prácticas de desarrollo que hemos tratado durante todo el curso 2024/2025.

Requisitos Previos:

- ✅ Conocimientos adquirido en este curso de DWEC.
- ✅ Una cuenta gratuita en The Movie Database (TMDB) y una API key.

Pasos a Seguir:

1. 🚀 **Preparación del Proyecto:**
    - Crear un nuevo proyecto React utilizando Vite:
    - Instalar Tailwind CSS para el estilizado:
    - Instalar react router dom
    - Crear la estructura de carpetas necesaria dentro de `src`: `components`, `contexts`, `hooks`, `layouts`, `pages`, `router`, `services`.

- Crear un archivo `.env` en la raíz del proyecto para almacenar la API key de TMDB (ej: `VITE_API_TOKEN=tu_api_key`).

2. 🌐 **Conexión a la API de TMDB:**
   - Crear el archivo `services/tmdb.js`.
   - Definir constantes para la API key, la URL base, la URL base de imágenes y los tamaños de imagen.
   - Implementar las siguientes funciones:
     - `getImageUrl(path, size)`: Construye la URL completa de una imagen.
     - `fetchFromApi(endpoint, options)`: Realiza peticiones a la API.
     - `getPopularMovies(page)`: Obtiene películas populares (paginadas).
     - `getMovieDetails(id)`: Obtiene detalles de una película.
     - `searchMovies(query, page)`: Busca películas por término.
     - `getMovieVideos(id)`: Obtener los videos de una película.
3. 🎣 **Creación del Hook `useFetch`:**
   - Crear el archivo `hooks/useFetch.js`.
   - Implementar el hook `useFetch` para manejar las peticiones a la API, incluyendo:
     - Estado de carga (`loading`).
     - Estado de error (`error`).
     - Datos recibidos (`data`).
     - Debe aceptar una función de fetching y un array de dependencias.
4. 🏠 **Creación de la Página Principal (`Home`):**
   - Crear el archivo `pages/Home.jsx`.

- Utilizar `useFetch` y `getPopularMovies` para obtener las películas populares.
- Mostrar un spinner de carga mientras se obtienen los datos (crear el componente `LoadingSpinner`).
- Mostrar un mensaje de error si la petición falla.
- Mostrar una lista de películas (inicialmente, solo los títulos).
- Implementar paginación como extra.

5. 🎬 **Creación del Componente `MovieCard`:**
   - Crear el archivo `components/MovieCard.jsx`.
   - Mostrar la información básica de una película (imagen, título, año, puntuación).
   - Usar Tailwind CSS para el diseño.
   - Preparar el componente para la funcionalidad de favoritos (pero no implementarla todavía).
6. 🎞️ **Creación de la Página de Detalles de Película (`MovieDetail`):**
   - Crear el archivo `pages/MovieDetail.jsx`.
   - Obtener el ID de la película de la URL (usar `useParams` de React Router).
   - Utilizar `useFetch`, `getMovieDetails` y `getMovieVideos` para obtener los detalles y videos de la película.
   - Mostrar la información completa de la película (imagen de fondo, póster, título, sinopsis, géneros, trailer, etc.).
   - Preparar el componente para la funcionalidad de favoritos y reseñas (pero no implementarlas todavía).
7. ⭐ **Creación del Contexto de Favoritos:**
   - Crear el archivo `contexts/FavoritesContext.jsx`.
   - Implementar `FavoritesContext` y `FavoritesProvider` para gestionar el estado global de favoritos:

- Añadir/quitar películas de favoritos.
- Verificar si una película es favorita.
- Obtener la lista de favoritos.
- Persistir los favoritos en `localStorage`.
- Mostrar notificaciones (crear el contexto de Toast en el paso 10).

8. 📝 **Creación del Contexto de Reseñas:**
   - Crear el archivo `contexts/ReviewsContext.jsx`.
   - Implementar `ReviewsContext` y `ReviewsProvider` para gestionar el estado global de reseñas:
     - Añadir reseñas.
     - Eliminar reseñas.
     - Obtener las reseñas de una película.
     - Persistir las reseñas en `localStorage`.
     - Mostrar notificaciones (crear el contexto de Toast en el paso 10).
9. 💬 **Creación de los Componentes de Reseñas:**
   - Crear el archivo `components/ReviewForm.jsx`
   - Crear el componente `components/ReviewItem.jsx`
10. 🔔 **Creación del Contexto de Toast (Notificaciones):**
    - Crear el archivo `contexts/ToastContext.jsx`.
    - Implementar `ToastContext` y `ToastProvider` para mostrar notificaciones al usuario.
11. 🔍 **Creación de la Página de Búsqueda (`Search`):**
    - Crear el archivo `pages/Search.jsx`.
    - Implementar un campo de búsqueda (crear el componente `SearchBox`).

- Utilizar `useFetch` y `searchMovies` para buscar películas.
- Mostrar los resultados de la búsqueda.
- Manejar la paginación como extra.

12. 📜 **Creación de la Página de Listado Completo de Películas (`MovieList`):**
    - Crear el archivo `pages/MovieList.jsx`
    - Obtener y mostrar todas las peliculas populares o por filtro
    - Implementar filtros.
13. ❤ **Creación de la página de favoritos (`Favorites`):**
    - Crear el archivo `pages/Favorites.jsx`
    - Obtener las peliculas favoritas del usuario
    - Mostrar las películas en un listado
14. 📝 **Creación de la página de reseñas (`Reviews`):**
    - Crear el archivo `pages/Reviews.jsx`
    - Obtener y mostrar las películas.
    - Mostrar las reseñas de cada película.
15. 🧭 **Configuración de las Rutas:**
    - Crear el archivo `router/index.jsx` .
    - Utilizar `createBrowserRouter` (de React Router) para definir las rutas de la aplicación:
        - `/` : Página principal (`Home`).
        - `/movie/:id` : Detalles de película (`MovieDetail`).
        - `/search` : Búsqueda (`Search`).
        - `/movies` : Listado completo de películas (`MovieList`).
        - `/favorites` : Películas favoritas (`Favorites`).

        - `/reviews`: Reseñas del usuario (`Reviews`).

16. 🔄 **Integración Final:**
    - En `app.jsx`, envolver la aplicación con los providers de contexto necesarios (`ToastProvider`, `FavoritesProvider`, `ReviewsProvider`).
    - Asegurarse de que todos los componentes utilicen los contextos y hooks correctamente.

¡Desafíos Adicionales! (Opcional)

- ✨ **Mejorar la UI/UX:** Añadir transiciones, animaciones, un diseño más pulido.
- 🔒 **Autenticación:** Añadir un sistema de autenticación de usuarios (esto requeriría un backend).
- 🌍 **Internacionalización:** Traducir la aplicación a otros idiomas.