

VideoClub - Documentación

Descripción General

Aplicación web que permite a los usuarios explorar, buscar y gestionar películas. Los usuarios pueden registrarse mediante email o Google para acceder a funcionalidades como guardar favoritos y escribir reseñas.

La aplicación ofrece una interfaz elegante y moderna, con un diseño en tonos oscuros y acentos en morado y rosa. Cuenta con una página principal que muestra películas populares, un sistema de búsqueda avanzado con múltiples filtros, y páginas detalladas para cada película donde se puede ver información completa, trailers y reseñas.

La aplicación está construida con React, utiliza un backend Node.js que hace peticiones a la API TMDB para la información de películas.



Tecnologías Utilizadas

Frontend

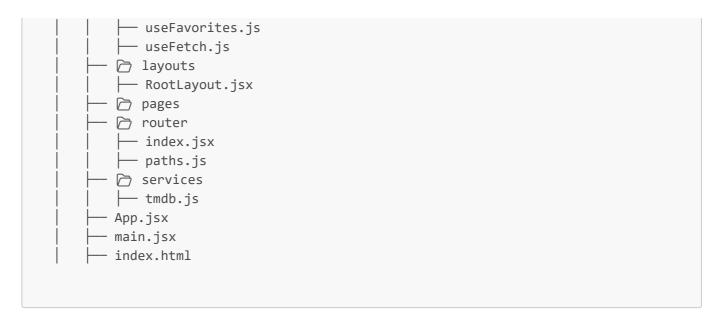
- React.js (con Vite y React Router)
- Context API (para gestión de estado global de autenticación y favoritos)
- **Fetch API** (para comunicarse con la API de TMDB y el backend)
- \(\sum \) **Sonner** (para notificaciones visuales)

Funcionalidades Clave

- **Q** Buscar películas por nombre o género.
- Agregar y gestionar favoritos.
- Escribir y ver reseñas de películas.
- Registro e inicio de sesión con JWT y OAuth (Google).
- Explorar películas populares y de distintos géneros.
- Modo protegido para rutas de usuario autenticado.

S Estructura del Proyecto





Puntos destacables de este frontend

Gestión de estados

Autenticación local

- Manejo de estado global con Context API
- Sincronización con el backend
- Manejo de errores y estados de carga

Sistema de Búsqueda Avanzado

```
// Search.jsx - Búsqueda con filtros dinámicos
const performSearch = async (searchQuery, currentFilters) => {
  const searchOptions = {
    ...currentFilters,
    page: 1,
  };
```

```
const response = await searchMovies(searchQuery, searchOptions);
setMovies(response.results || []);
};
```

- Búsqueda en tiempo real
- Filtros combinados (género, año, puntuación)
- Optimización y centralización de llamadas a la API

Custom Hooks Reutilizables

```
// useFetch.js - Hook personalizado para peticiones
export const useFetch = (fetchFunction, dependencies = []) => {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(false);

useEffect(() => {
  const abortController = new AbortController();
  fetchData(abortController.signal);
  return () => abortController.abort();
}, dependencies);
```

- Manejo automático de estados de carga
- Cancelación de peticiones al desmontar
- Reutilizable en toda la aplicación

Sistema de Autenticación Completo

```
// AuthContext.jsx - Manejo de autenticación
const handleLogin = async (email, password) => {
   try {
     const data = await login(email, password);
     const userData = await fetchUser();
     setUser(userData);
     setIsAuthenticated(true);
     router.navigate("/");
   } catch (error) {
     throw error;
   }
};
```

- Múltiples métodos de autenticación
- Persistencia de sesión
- Manejo de rutas protegidas

Arquitectura Escalable

- Organización modular del código
- Configuración centralizada
- Fácil mantenimiento y extensión

% Autor y Créditos

🔊 **Desarrollado por:** Ana María García García.