|  |
| --- |
|  |
| MO 6 Desenvolupament Web Client |
| UF2 - Projecte React |

|  |
| --- |
| Mohamed Bouirig Ait Bourig  27-05-2024 |

INDICE

[1 - Objetivos: 2](#_Toc167712419)

[2 - Estructura del proyecto: 2](#_Toc167712420)

[3 - Componentes: 3](#_Toc167712421)

[4 - Funcionalidades: 8](#_Toc167712422)

[5 - Tecnologías utilizadas: 11](#_Toc167712423)

[6 - Conclusión: 11](#_Toc167712424)

# 1 - Objetivos:

(1 punt) La informació de les pel·lícules s’ha d’agafar d’un fitxer local i emmagatzemar-se en un context. I ser accessible des de tots els components de l’aplicació.

• (1 punt) S’ha d’utilitzar un Router per a definir les rutes de la nostra aplicació.

• (1 punt) Ha d'haver-hi una barra de navegació que tingui almenys enllaços a la pàgina principal, mostrar pel·lícules, inserir una pel·lícula i nosaltres.

• (2 punts) La pàgina principal (Catalog) ha de mostrar totes les pel·lícules en format targeta (PeliculaCard) i en cada targeta de pel·lícula hi haurà un botó per veure la informació detallada de cada pel·lícula.

• (2 punts) En la vista individual de cada pel·lícula (PeliculaDetall) ha d'haver-hi la informació completa de la pel·lícula i botons per llogar (i retornar), modificar i eliminar cada pel·lícula.

• (2 punts) Per a inserir i modificar les pel·lícules s’ha de fer servir el Hook useForm per crear els formularis.

• L’estructura mínima de components ha de ser: ◦ Calatog: Component que obtindrà l’array de pel·lícules del context i les mostrarà en format targeta. ◦ PeliculaCard: Component que mostrarà la informació bàsica d’una pel·lícula amb un botó per mostrar la informació detallada de cada pel·lícula. ◦ PeliculaDetall: Component per mostrar tota la informació completa de cada pel·lícula amb els botons per llogar (i retornar), modificar i eliminar la pel·lícula en concret.

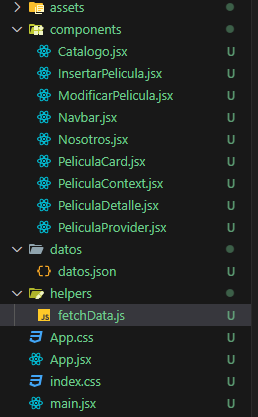
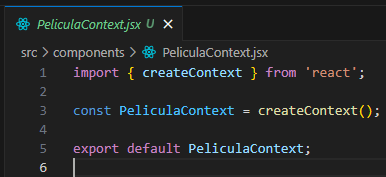
• (1 punt) Memòria.

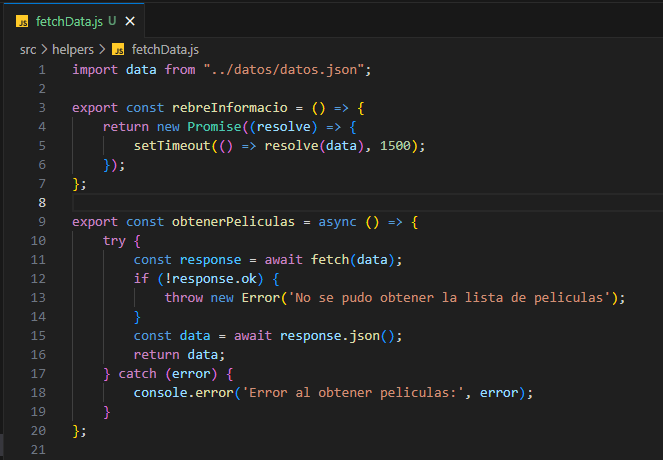
# 2 - Estructura del proyecto:

El proyecto está estructurado en varios componentes React, cada uno encargado de una funcionalidad específica.

Además, se utiliza un contexto (PeliculaContext) para compartir el estado global de las películas entre los componentes.

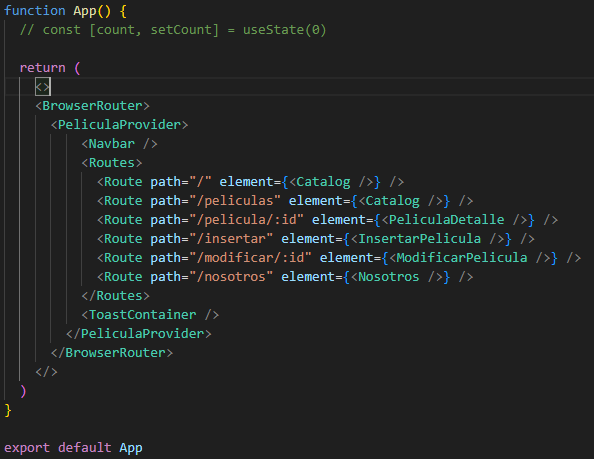
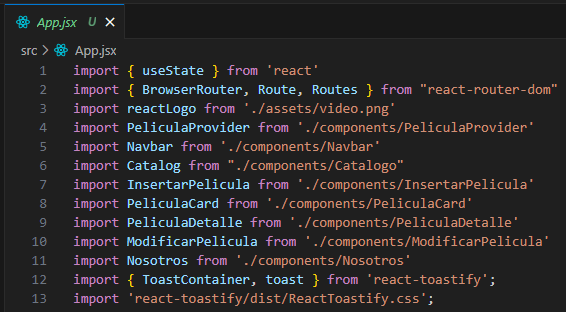
El archivo fetchData.js contiene funciones para obtener y procesar los datos de las películas.

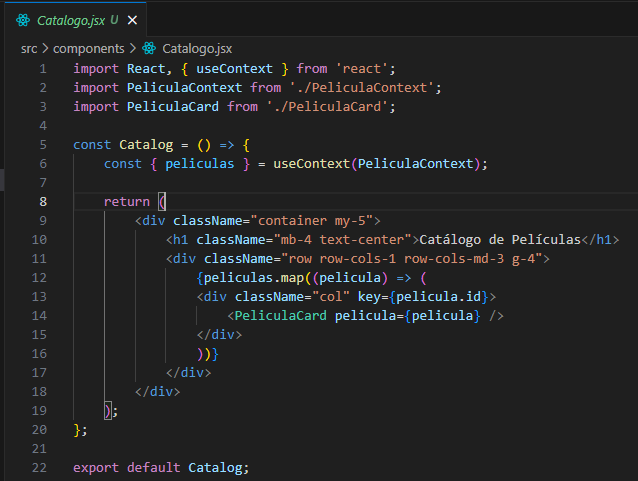


# 3 - Componentes:

App.jsx: Este es el componente principal que define las rutas y renderiza los demás componentes. Aquí se configura el enrutamiento de la aplicación utilizando React Router, se importan los otros componentes y se envuelven en el proveedor de contexto PeliculaProvider.

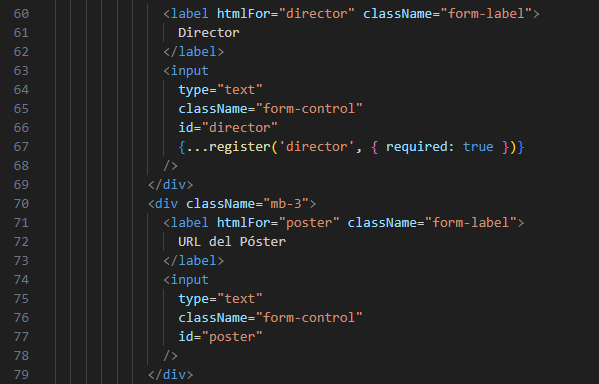


Catalog.jsx: Este componente muestra el catálogo de películas. Utiliza el contexto PeliculaContext para obtener la lista de películas y las renderiza usando el componente PeliculaCard.



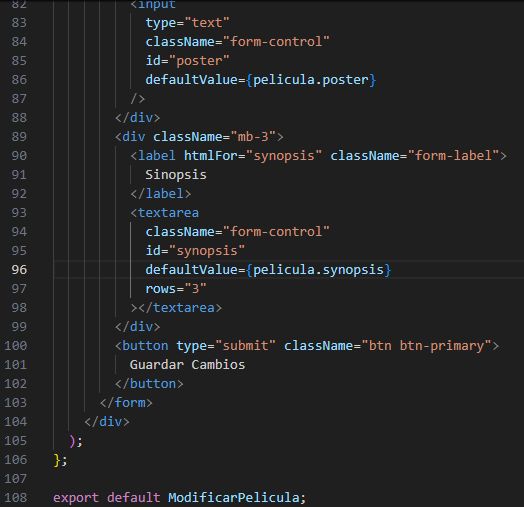
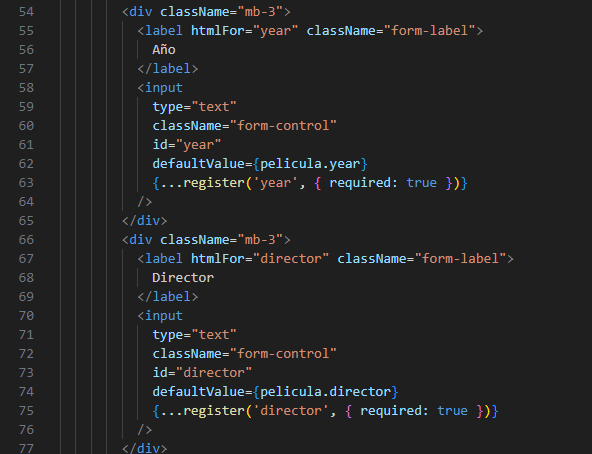
InsertarPelicula.jsx: Este componente permite a los usuarios insertar una nueva película. Utiliza el hook useForm de la biblioteca react-hook-form para manejar el formulario y el contexto PeliculaContext para actualizar la lista de películas.



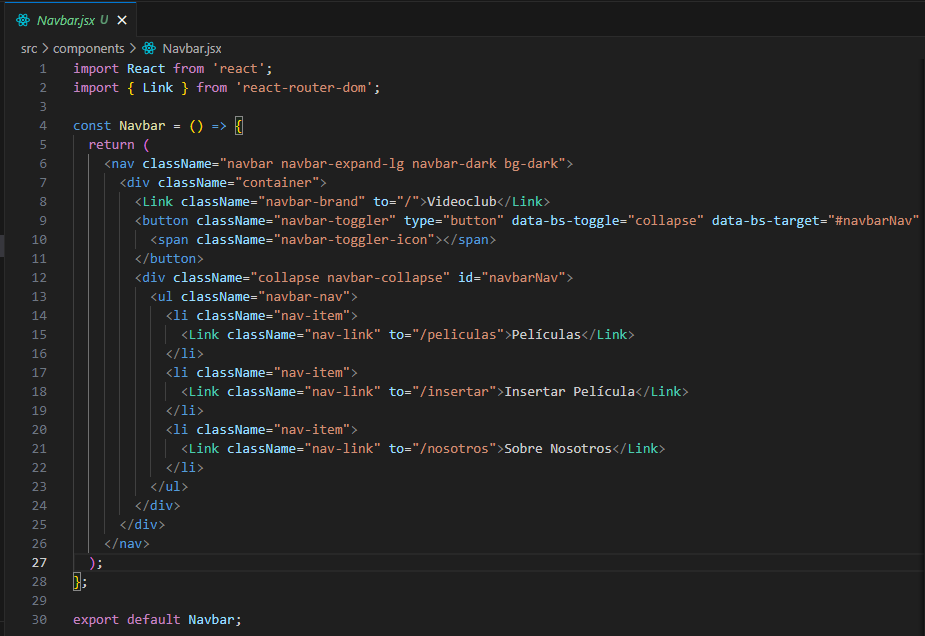


ModificarPelicula.jsx: Este componente permite a los usuarios modificar una película existente. Al igual que InsertarPelicula.jsx, utiliza useForm y PeliculaContext para actualizar la información de la película.

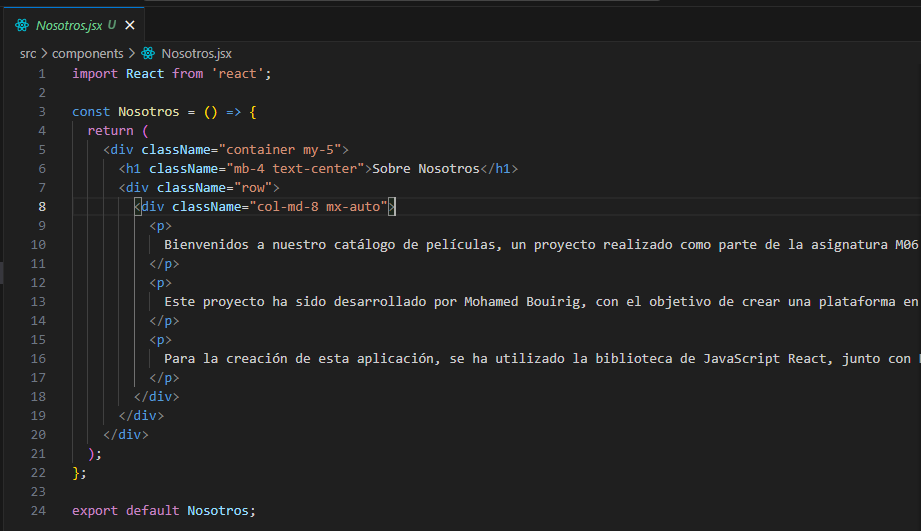




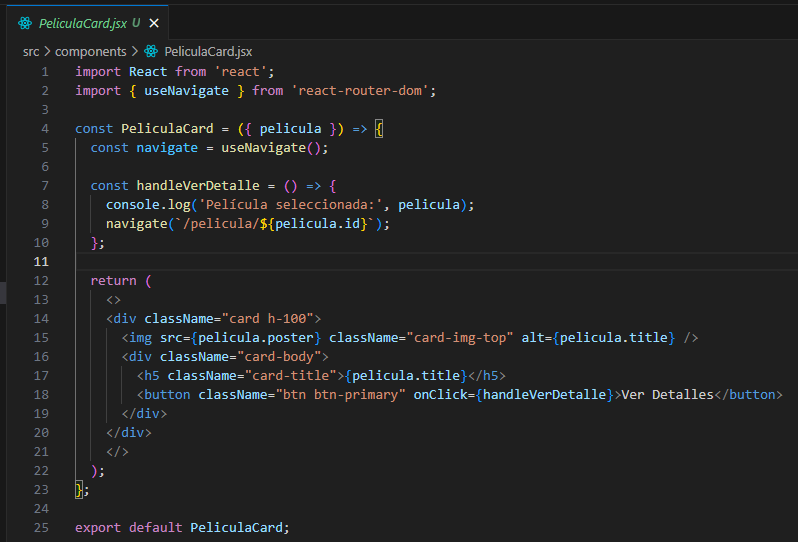
Navbar.jsx: Este componente proporciona la barra de navegación de la aplicación, con enlaces a las diferentes secciones (Peliculas “Catálogo”, Insertar Película, Sobre Nosotros).



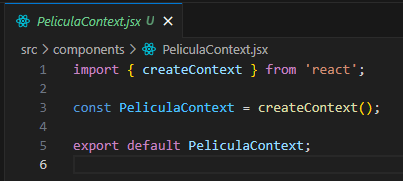
Nosotros.jsx: Este componente muestra información sobre el proyecto y los desarrolladores.



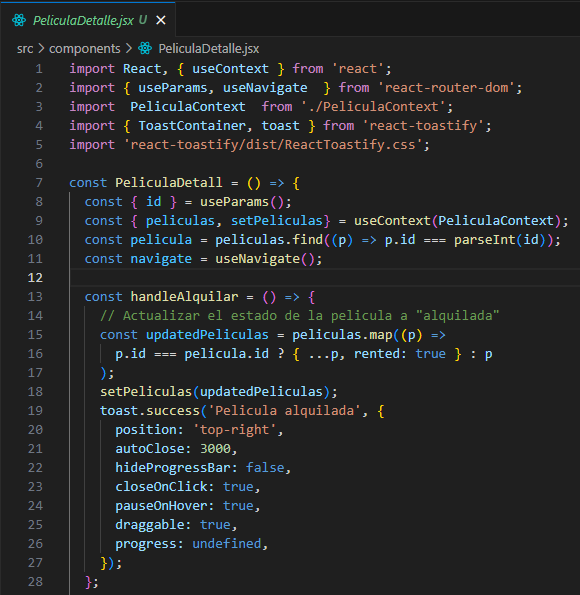
PeliculaCard.jsx: Este componente representa una tarjeta individual de película en el catálogo. Cuando se hace clic en el botón "Ver Detalles", redirige al usuario a la página de detalles de la película.

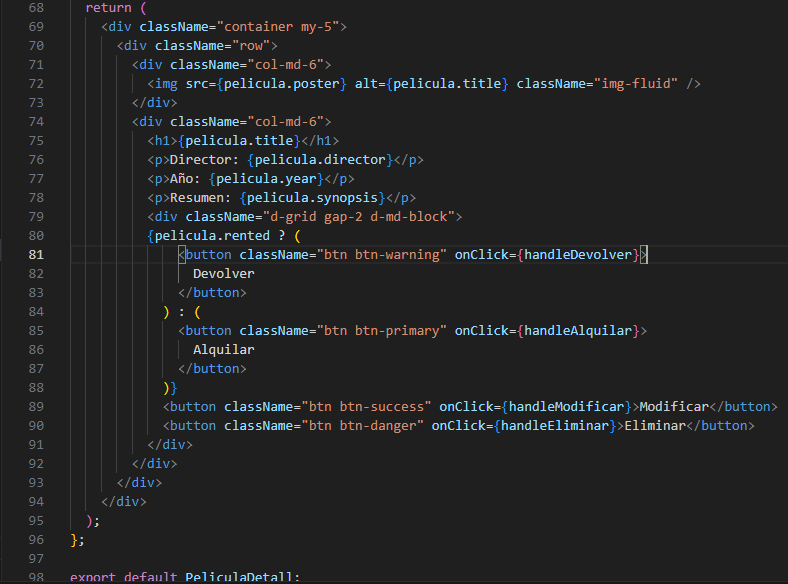


PeliculaContext.jsx: Este archivo define el contexto PeliculaContext, que se utiliza para compartir el estado de las películas entre los diferentes componentes de la aplicación.

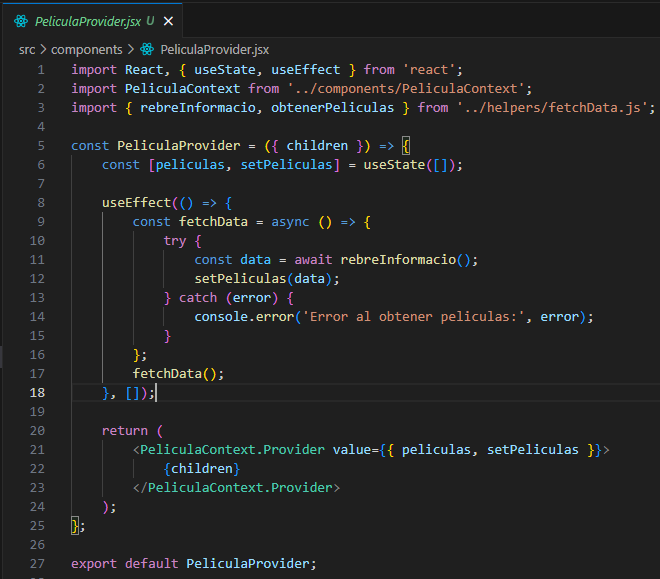


PeliculaDetalle.jsx: Este componente muestra los detalles de una película seleccionada, incluyendo la posibilidad de alquilar, devolver, modificar y eliminar la película.



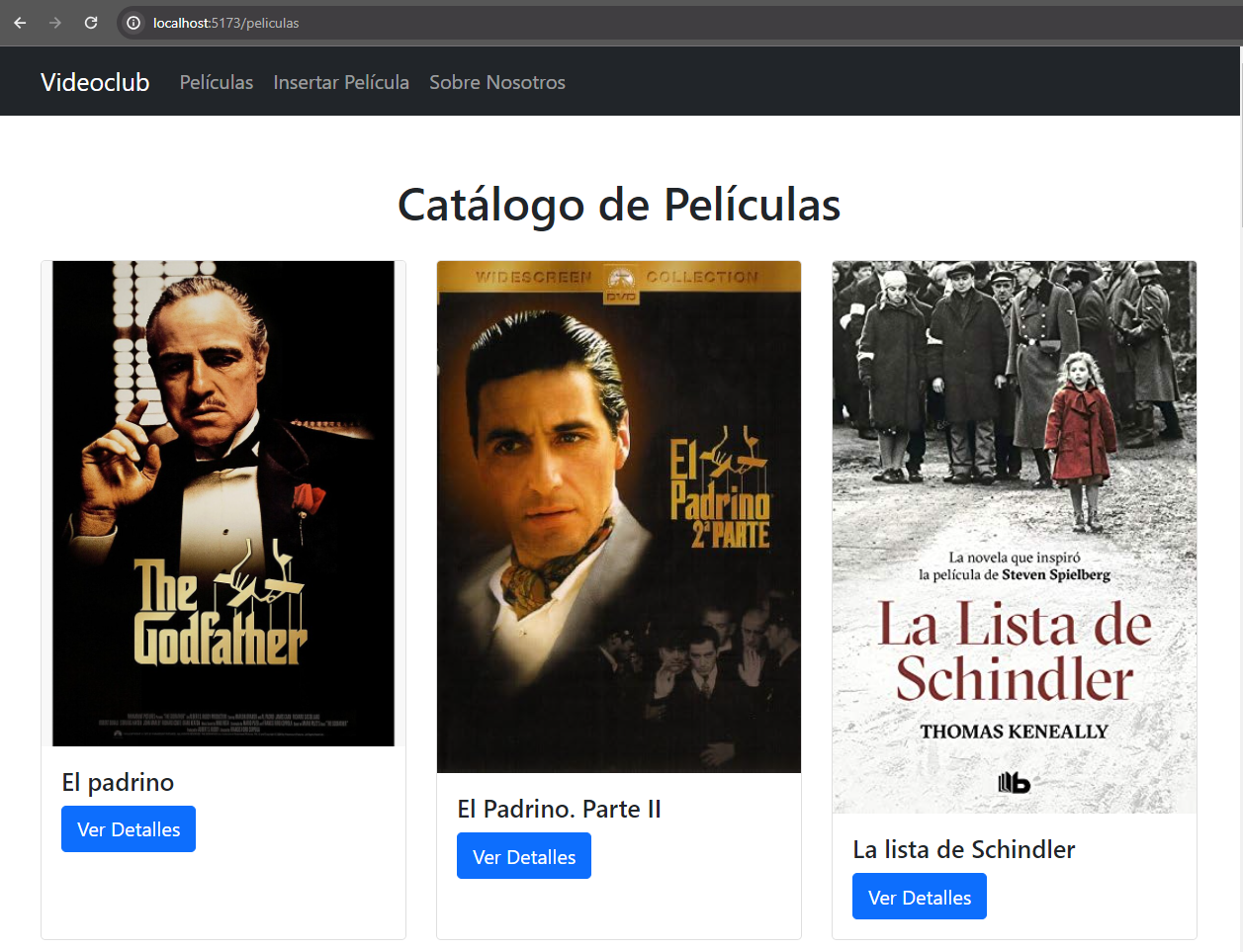


PeliculaProvider.jsx: Este componente proporciona el proveedor de contexto para las películas, cargando los datos iniciales desde un archivo JSON gracias a las funciones de fetchData.js.

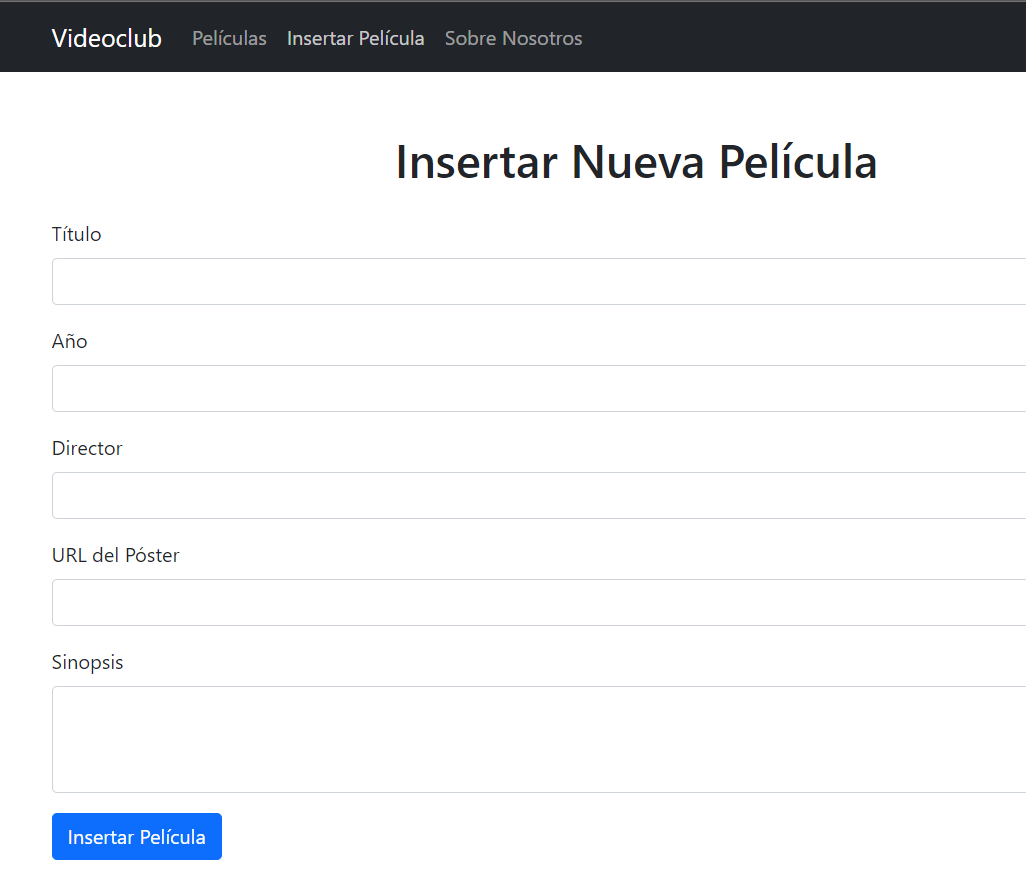


# 4 - Funcionalidades:

Visualización del catálogo de películas: El usuario puede ver la lista de películas disponibles en el catálogo.



Inserción de nuevas películas: El usuario puede agregar una nueva película al catálogo a través del formulario de inserción.



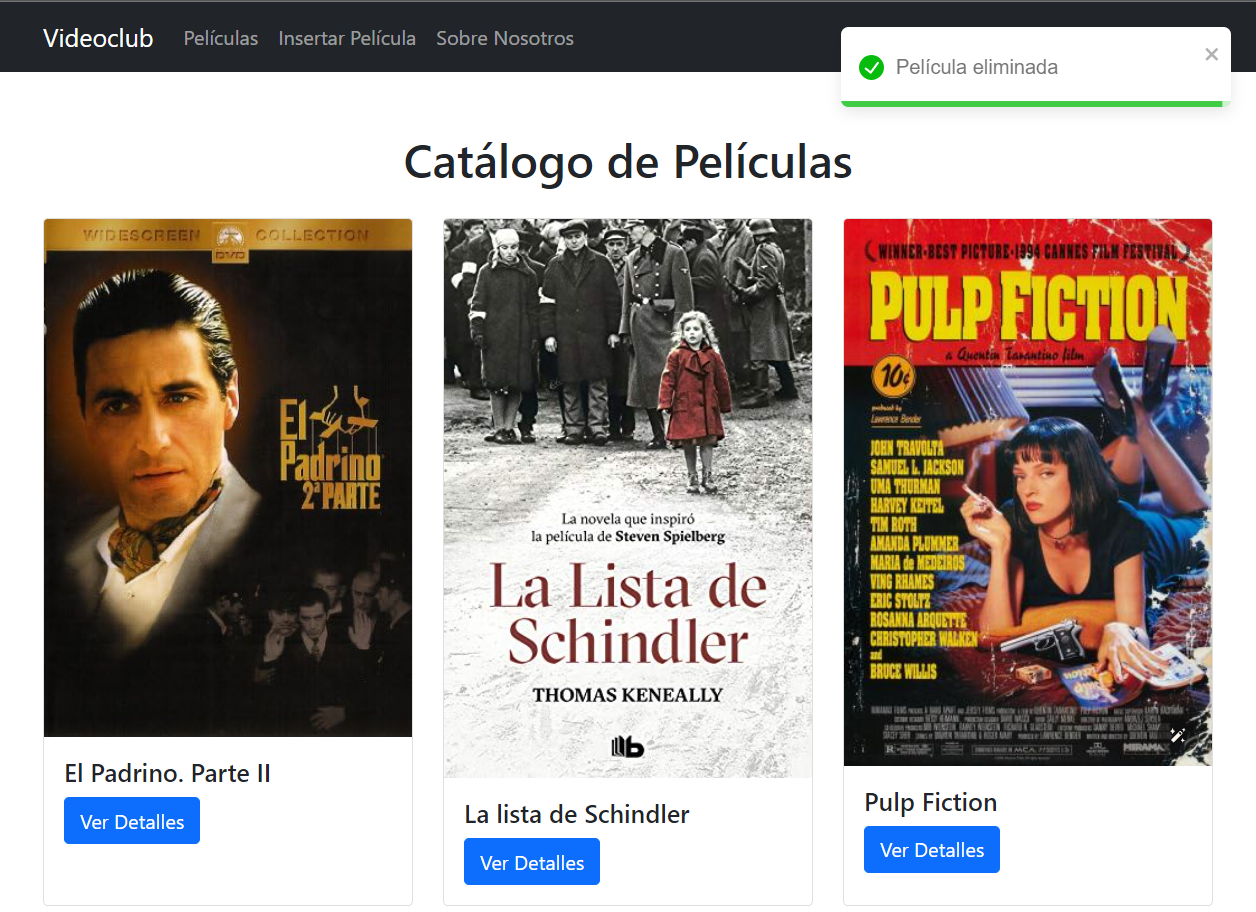
Modificación de películas existentes: El usuario puede editar la información de una película existente a través del formulario de modificación.



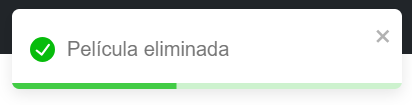
Alquiler y devolución de películas: El usuario puede alquilar o devolver una película desde la página de detalles de la película.

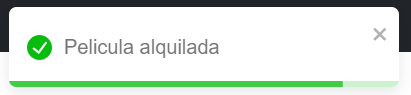


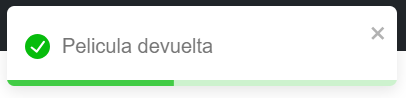
Eliminación de películas: El usuario puede eliminar una película desde la página de detalles de la película.

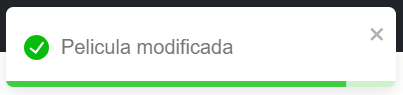


Notificaciones de éxito: Se utilizan notificaciones (toasts) para informar al usuario sobre las operaciones exitosas, como la inserción, modificación, alquiler o eliminación de una película.











# 5 - Tecnologías utilizadas:

React: Librería de JavaScript para la construcción de interfaces de usuario.

React Router: Biblioteca para el manejo de enrutamiento en aplicaciones React.

React Hook Form: Biblioteca para la gestión de formularios en React.

React Toastify: Biblioteca para mostrar notificaciones. (<https://www.npmjs.com/package/react-toastify>)

Bootstrap: Framework CSS para el diseño y la interfaz de usuario.

JSON: Formato de intercambio de datos utilizado para almacenar y transmitir la información de las películas.

# 6 - Conclusión:

Ha sido un proyecto bastante interesante que me ha ayudado a comprender mucho mejor React y de sus ventajas. Al principio he tenido dificultades a la hora de realizar la lógica para alquilar, modificar y eliminar, pero con algo de investigación he encontrado varias formas de hacerlo.

También he encontrado una librería toastify (parecida o igual a toastr, que te enseñé en la anterior uf pero que no pude implementar) para React para el tema de las notificaciones y no hacerlo con alert ya que era demasiado intrusivo.