

# MO 6

# Desenvolupament

# Web Client

UF2 - Projecte React

Mohamed Bourig Ait Bourig  
27-05-2024

INDICE

1 - Objetivos: ..... 2

2 - Estructura del proyecto:..... 2

3 - Componentes: ..... 3

4 - Funcionalidades: ..... 8

5 - Tecnologías utilizadas: ..... 11

6 - Conclusión: ..... 11

## 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.

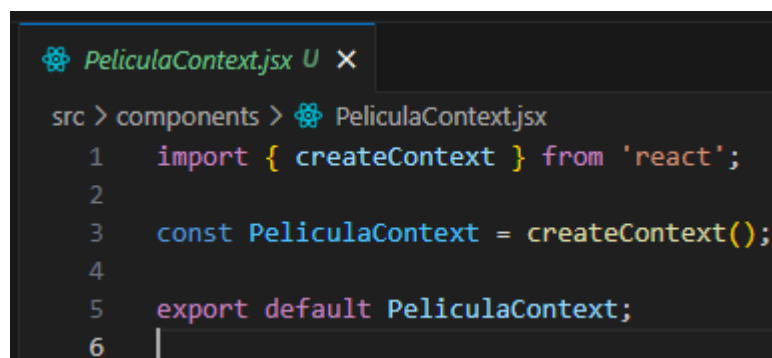
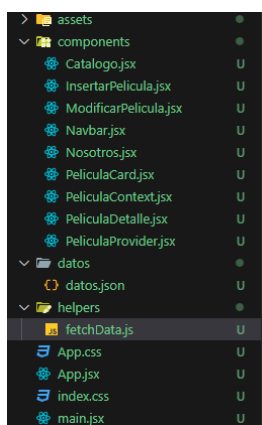
- (1 punt) S'ha d'utilitzar un Router per a definir les rutes de la nostra aplicació.
- (1 punt) Ha d'haver-hi una barra de navegació que tingui almenys enllaços a la pàgina principal, mostrar pel·lícules, inserir una pel·lícula i nosaltres.
- (2 punts) La pàgina principal (Catalog) ha de mostrar totes 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.



```
fetchData.js U x
src > helpers > fetchData.js
1 import data from "../datos/datos.json";
2
3 export const rebreInformacio = () => {
4   return new Promise((resolve) => {
5     setTimeout(() => resolve(data), 1500);
6   });
7 };
8
9 export const obtenerPelículas = async () => {
10   try {
11     const response = await fetch(data);
12     if (!response.ok) {
13       throw new Error('No se pudo obtener la lista de películas');
14     }
15     const data = await response.json();
16     return data;
17   } catch (error) {
18     console.error('Error al obtener películas:', error);
19   }
20 };
21
```

### 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.

```
App.jsx U x
src > App.jsx
1 import { useState } from 'react'
2 import { BrowserRouter, Route, Routes } from 'react-router-dom'
3 import reactLogo from './assets/video.png'
4 import PeliculaProvider from './components/PeliculaProvider'
5 import NavBar from './components/NavBar'
6 import Catalog from './components/Catalogo'
7 import InsertarPelicula from './components/InsertarPelicula'
8 import PeliculaCard from './components/PeliculaCard'
9 import PeliculaDetalle from './components/PeliculaDetalle'
10 import ModificarPelicula from './components/ModificarPelicula'
11 import Nosotros from './components/Nosotros'
12 import { ToastContainer, toast } from 'react-toastify';
13 import 'react-toastify/dist/ReactToastify.css';

function App() {
  // const [count, setCount] = useState(0)

  return (
    <BrowserRouter>
      <PeliculaProvider>
        <NavBar />
        <Routes>
          <Route path="/" element={<Catalog />} />
          <Route path="/películas" element={<Catalog />} />
          <Route path="/película/:id" element={<PeliculaDetalle />} />
          <Route path="/insertar" element={<InsertarPelicula />} />
          <Route path="/modificar/:id" element={<ModificarPelicula />} />
          <Route path="/nosotros" element={<Nosotros />} />
        </Routes>
        <ToastContainer />
      </PeliculaProvider>
    </BrowserRouter>
  )
}

export default App
```

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.

```
Catalogo.jsx U X
src > components > Catalogo.jsx
1  import React, { useContext } from 'react';
2  import PeliculaContext from './PeliculaContext';
3  import PeliculaCard from './PeliculaCard';
4
5  const Catalog = () => {
6    const { peliculas } = useContext(PeliculaContext);
7
8    return (
9      <div className="container my-5">
10        <h1 className="mb-4 text-center">Catálogo de Películas</h1>
11        <div className="row row-cols-1 row-cols-md-3 g-4">
12          {peliculas.map((pelicula) => (
13            <div className="col" key={pelicula.id}>
14              <PeliculaCard pelicula={pelicula} />
15            </div>
16          ))}
17        </div>
18      </div>
19    );
20  };
21
22  export default Catalog;
```

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.

```
InsertarPelicula.jsx U X
src > components > InsertarPelicula.jsx
1  import React, { useContext } from 'react';
2  import { useNavigate } from 'react-router-dom';
3  import PeliculaContext from './PeliculaContext';
4  import { useForm } from 'react-hook-form';
5  import { ToastContainer, toast } from 'react-toastify';
6  import 'react-toastify/dist/ReactToastify.css';
7
8  const InsertarPelicula = () => {
9    const { peliculas, setPeliculas } = useContext(PeliculaContext);
10   const navigate = useNavigate();
11   const { register, handleSubmit, reset } = useForm();
12
13   const guardarPeli = (data) => {
14     const nuevaPelicula = {
15       id: peliculas.length + 1,
16       ...data,
17       rented: false,
18     };
19     setPeliculas([...peliculas, nuevaPelicula]);
20     toast.success('Pelicula insertada', {
21       position: 'top-right',
22       autoClose: 3000,
23       hideProgressBar: false,
24       closeOnClick: true,
25       pauseOnHover: true,
26       draggable: true,
27       progress: undefined,
28     });
29     reset();
30     navigate('/peliculas');
31   };
32
33   return (
34     <div className="container my-5">
35       <h1 className="mb-4 text-center">Insertar Nueva Pelicula</h1>
36       <form onSubmit={handleSubmit(guardarPeli)}>
37         <div className="mb-3">
38           <label htmlFor="title" className="form-label">
39             Título
40           </label>
41           <input
42             type="text"
43             className="form-control"
44             id="title"
45             {...register('title', { required: true })}
46           />
47         </div>
48         <div className="mb-3">
49           <label htmlFor="year" className="form-label">
50             Año
51           </label>
52           <input
53             type="text"
54             className="form-control"
55             id="year"
56             {...register('year', { required: true })}
57           />
58         </div>
59         <div className="mb-3">
60           <label htmlFor="director" className="form-label">
61             Director
```

```

60      <label htmlFor="director" className="form-label">
61        Director
62      </label>
63      <input
64        type="text"
65        className="form-control"
66        id="director"
67        {...register('director', { required: true })}
68      />
69    </div>
70    <div className="mb-3">
71      <label htmlFor="poster" className="form-label">
72        URL del Póster
73      </label>
74      <input
75        type="text"
76        className="form-control"
77        id="poster"
78      />
79    </div>

```

```

74      <input
75        type="text"
76        className="form-control"
77        id="poster"
78      />
79    </div>
80    <div className="mb-3">
81      <label htmlFor="synopsis" className="form-label">
82        Sinopsis
83      </label>
84      <textarea
85        className="form-control"
86        id="synopsis"
87        rows="3"
88      ></textarea>
89    </div>
90    <button type="submit" className="btn btn-primary">
91      Insertar Película
92    </button>
93  </form>
94 </div>
95 );
96 };
97
98 export default InsertarPelícula;

```

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

```

src > components > ModificarPelícula.jsx
1  import React, { useContext } from 'react';
2  import { useNavigate, useParams } from 'react-router-dom';
3  import PelículaContext from './PelículaContext';
4  import { useForm } from 'react-hook-form';
5  import { ToastContainer, toast } from 'react-toastify';
6  import 'react-toastify/dist/ReactToastify.css';
7
8  const ModificarPelícula = () => {
9    const { id } = useParams();
10   const { películas, setPelículas } = useContext(PelículaContext);
11   const navigate = useNavigate();
12   const { register, handleSubmit } = useForm();
13
14   const película = películas.find((p) => p.id === parseInt(id));
15
16   const guardarPeli = (data) => {
17     const updatedPelícula = {
18       id: película.id,
19       ...data,
20       rented: película.rented,
21     };
22     const updatedPelículas = películas.map((p) =>
23       p.id === película.id ? updatedPelícula : p
24     );
25     setPelículas(updatedPelículas);
26     toast.success('Película modificada', {
27       position: 'top-right',
28       autoClose: 3000,
29       hideProgressBar: false,
30       closeOnClick: true,
31       pauseOnHover: true,
32       draggable: true,
33     });
34   };
35   navigate('/películas');
36
37   return (
38     <div className="container my-5">
39       <h1 className="mb-4 text-center">Modificar Película</h1>
40       <form onSubmit={handleSubmit(guardarPeli)}>
41         <div className="mb-3">
42           <label htmlFor="title" className="form-label">
43             Título
44           </label>
45           <input
46             type="text"
47             className="form-control"
48             id="title"
49             defaultValue={película.title}
50             {...register('title', { required: true })}
51           />
52         </div>
53

```

```

54     <div className="mb-3">
55       <label htmlFor="year" className="form-label">
56         Año
57       </label>
58       <input
59         type="text"
60         className="form-control"
61         id="year"
62         defaultValue={película.year}
63         {...register('year', { required: true })}
64       />
65     </div>
66     <div className="mb-3">
67       <label htmlFor="director" className="form-label">
68         Director
69       </label>
70       <input
71         type="text"
72         className="form-control"
73         id="director"
74         defaultValue={película.director}
75         {...register('director', { required: true })}
76       />
77     </div>

```

```

82     <input
83       type="text"
84       className="form-control"
85       id="poster"
86       defaultValue={película.poster}
87     />
88   </div>
89   <div className="mb-3">
90     <label htmlFor="synopsis" className="form-label">
91       Sinopsis
92     </label>
93     <textarea
94       className="form-control"
95       id="synopsis"
96       defaultValue={película.synopsis}
97       rows="3"
98     ></textarea>
99   </div>
100   <button type="submit" className="btn btn-primary">
101     Guardar Cambios
102   </button>
103 </form>
104 </div>
105 );
106 };
107
108 export default ModificarPelícula;

```

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

```
Navbar.jsx U X
src > components > Navbar.jsx
1 import React from 'react';
2 import { Link } from 'react-router-dom';
3
4 const Navbar = () => {
5   return (
6     <nav className="navbar navbar-expand-lg navbar-dark bg-dark">
7       <div className="container">
8         <Link className="navbar-brand" to="/">Videoclub</Link>
9         <button className="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav">
10           <span className="navbar-toggler-icon"></span>
11         </button>
12         <div className="collapse navbar-collapse" id="navbarNav">
13           <ul className="nav navbar-nav">
14             <li className="nav-item">
15               <Link className="nav-link" to="/peliculas">Películas</Link>
16             </li>
17             <li className="nav-item">
18               <Link className="nav-link" to="/insertar">Insertar Película</Link>
19             </li>
20             <li className="nav-item">
21               <Link className="nav-link" to="/nosotros">Sobre Nosotros</Link>
22             </li>
23           </ul>
24         </div>
25       </div>
26     </nav>
27   );
28 };
29
30 export default Navbar;
```

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

```
Nosotros.jsx U X
src > components > Nosotros.jsx
1 import React from 'react';
2
3 const Nosotros = () => {
4   return (
5     <div className="container my-5">
6       <h1 className="mb-4 text-center">Sobre Nosotros</h1>
7       <div className="row">
8         <div className="col-md-8 mx-auto">
9           <p>
10             Bienvenidos a nuestro catálogo de películas, un proyecto realizado como parte de la asignatura M06
11           </p>
12           <p>
13             Este proyecto ha sido desarrollado por Mohamed Boulig, con el objetivo de crear una plataforma en
14           </p>
15           <p>
16             Para la creación de esta aplicación, se ha utilizado la biblioteca de JavaScript React, junto con
17           </p>
18         </div>
19       </div>
20     </div>
21   );
22 };
23
24 export default Nosotros;
```

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.

```
PeliculaCard.jsx U X
src > components > PeliculaCard.jsx
1 import React from 'react';
2 import { useNavigate } from 'react-router-dom';
3
4 const PeliculaCard = ({ pelicula }) => {
5   const navigate = useNavigate();
6
7   const handleVerDetalle = () => {
8     console.log('Película seleccionada:', pelicula);
9     navigate(`/pelicula/${pelicula.id}`);
10   };
11
12   return (
13     <>
14       <div className="card h-100">
15         <img src={pelicula.poster} className="card-img-top" alt={pelicula.title} />
16         <div className="card-body">
17           <h5 className="card-title">{pelicula.title}</h5>
18           <button className="btn btn-primary" onClick={handleVerDetalle}>Ver Detalles</button>
19         </div>
20       </div>
21     </>
22   );
23 };
24
25 export default PeliculaCard;
```

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.

```
PeliculaContext.jsx U X
src > components > PeliculaContext.jsx
1  import { createContext } from 'react';
2
3  const PeliculaContext = createContext();
4
5  export default PeliculaContext;
6
```

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

```
PeliculaDetalle.jsx U X
src > components > PeliculaDetalle.jsx
1  import React, { useContext } from 'react';
2  import { useParams, useNavigate } from 'react-router-dom';
3  import PeliculaContext from '../PeliculaContext';
4  import { ToastContainer, toast } from 'react-toastify';
5  import 'react-toastify/dist/ReactToastify.css';
6
7  const PeliculaDetalle = () => {
8    const { id } = useParams();
9    const { peliculas, setPeliculas } = useContext(PeliculaContext);
10   const pelicula = peliculas.find(p => p.id === parseInt(id));
11   const navigate = useNavigate();
12
13   const handleAlquilar = () => {
14     // Actualizar el estado de la película a "alquilada"
15     const updatedPeliculas = peliculas.map((p) =>
16       p.id === pelicula.id ? { ...p, rented: true } : p
17     );
18     setPeliculas(updatedPeliculas);
19     toast.success('Película alquilada', {
20       position: 'top-right',
21       autoClose: 3000,
22       hideProgressBar: false,
23       closeOnClick: true,
24       pauseOnHover: true,
25       draggable: true,
26       progress: undefined,
27     });
28   };
29
30   const handleDevolver = () => {
31     // Actualizar el estado de la película a "devuelta"
32     const updatedPeliculas = peliculas.map((p) =>
33       p.id === pelicula.id ? { ...p, rented: false } : p
34     );
35     setPeliculas(updatedPeliculas);
36     toast.success('Película devuelta', {
37       position: 'top-right',
38       autoClose: 3000,
39       hideProgressBar: false,
40       closeOnClick: true,
41       pauseOnHover: true,
42       draggable: true,
43       progress: undefined,
44     });
45   };
46
47   const handleModificar = () => {
48     navigate(`/modificar/${pelicula.id}`);
49   };
50
51   const handleEliminar = () => {
52     // Eliminar la película del contexto y de la lista de películas
53     const updatedPeliculas = peliculas.filter(p => p.id !== pelicula.id);
54     setPeliculas(updatedPeliculas);
55     toast.success('Película eliminada', {
56       position: 'top-right',
57       autoClose: 3000,
58       hideProgressBar: false,
59       closeOnClick: true,
60       pauseOnHover: true,
61       draggable: true,
62       progress: undefined,
63     });
64     navigate('/peliculas'); // Redirigir al usuario al catálogo de películas
65   };
66
67   return (
68     <div className="container my-5">
69       <div className="row">
70         <div className="col-md-6">
71           <img src={pelicula.poster} alt={pelicula.title} className="img-fluid" />
72         </div>
73         <div className="col-md-6">
74           <h1>{pelicula.title}</h1>
75           <p>Director: {pelicula.director}</p>
76           <p>Año: {pelicula.year}</p>
77           <p>Resumen: {pelicula.synopsis}</p>
78           <div className="d-grid gap-2 d-md-block">
79             {pelicula.rented ? (
80               <button className="btn btn-warning" onClick={handleDevolver}>
81                 Devolver
82               </button>
83             ) : (
84               <button className="btn btn-primary" onClick={handleAlquilar}>
85                 Alquilar
86               </button>
87             )}
88             <button className="btn btn-success" onClick={handleModificar}>Modificar</button>
89             <button className="btn btn-danger" onClick={handleEliminar}>Eliminar</button>
90           </div>
91         </div>
92       </div>
93     </div>
94   );
95
96   export default PeliculaDetalle;
97
98
```

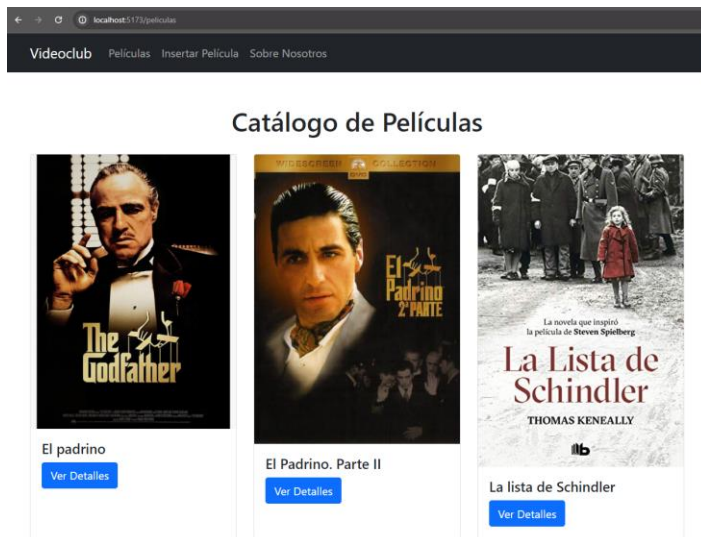


PeliculaProvider.jsx: Este componente proporciona el proveedor de contexto para las películas, cargando los datos iniciales desde un archivo JSON gracias a las funciones de fetchData.js.

```
PeliculaProvider.jsx U x
src > components > PeliculaProvider.jsx
1  import React, { useState, useEffect } from 'react';
2  import PeliculaContext from '../components/PeliculaContext';
3  import { rebreInformacio, obtenerPelículas } from '../helpers/fetchData.js';
4
5  const PeliculaProvider = ({ children }) => {
6    const [películas, setPelículas] = useState([]);
7
8    useEffect(() => {
9      const fetchData = async () => {
10        try {
11          const data = await rebreInformacio();
12          setPelículas(data);
13        } catch (error) {
14          console.error('Error al obtener películas:', error);
15        }
16      };
17      fetchData();
18    }, []);
19
20    return (
21      <PeliculaContext.Provider value={{ películas, setPelículas }}>
22        {children}
23      </PeliculaContext.Provider>
24    );
25  };
26
27  export default PeliculaProvider;
```

## 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.

Videoclub

Películas

Insertar Película

Sobre Nosotros

### Insertar Nueva Película

Título

Año

Director

URL del Póster

Sinopsis

Insertar Película

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

Videoclub

Películas

Insertar Película

Sobre Nosotros

### Modificar Película

Título

El padrino

Año

1972

Director

Francis Ford Coppola

URL del Póster

https://ih1.redbubble.net/image.1275478949.6629/flat,750x,075,f-pad,750x1000,f8f8f8.jpg

Sinopsis

Don Vito Corleone (Marlon Brando) es el respetado y temido jefe de una de las cinco familias de la mafia de Nueva York. Tiene cuatro hijos: Connie (Talia Shire), el impulsivo Sonny (James Caan), el pusiñime Freddie (John Cazale) y Michael (Al Pacino), que no quiere saber nada de los negocios de su padre. Cuando Corleone, en contra de los consejos de 'Il consigliere' Tom

Guardar Cambios

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

Videoclub

Películas

Insertar Película

Sobre Nosotros

Película alquilada



### El padrino

Director: Francis Ford Coppola

Año: 1972

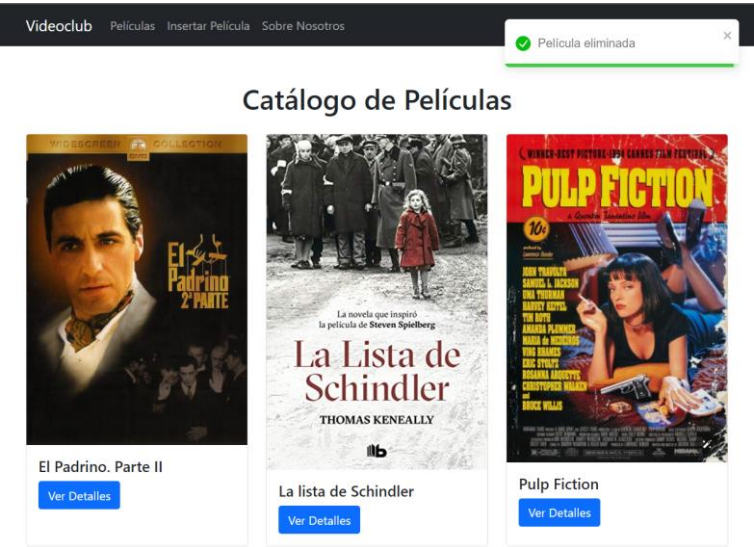
Resumen: Don Vito Corleone (Marlon Brando) es el respetado y temido jefe de una de las cinco familias de la mafia de Nueva York. Tiene cuatro hijos: Connie (Talia Shire), el impulsivo Sonny (James Caan), el pusiñime Freddie (John Cazale) y Michael (Al Pacino), que no quiere saber nada de los negocios de su padre. Cuando Corleone, en contra de los consejos de 'Il consigliere' Tom Hagen (Robert Duvall), se niega a intervenir en el negocio de las drogas, el jefe de otra banda ordena su asesinato. Empieza entonces una violenta y cruenta guerra entre las familias mafiosas.

Devolver

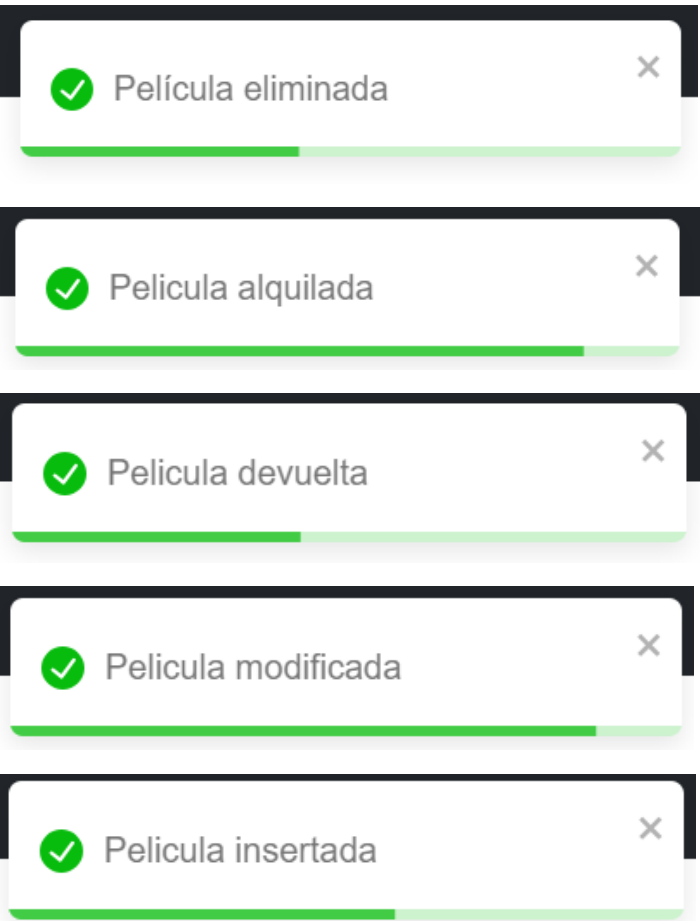
Modificar

Eliminar

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 uí pero que no pude implementar) para React para el tema de las notificaciones y no hacerlo con alert ya que era demasiado intrusivo.