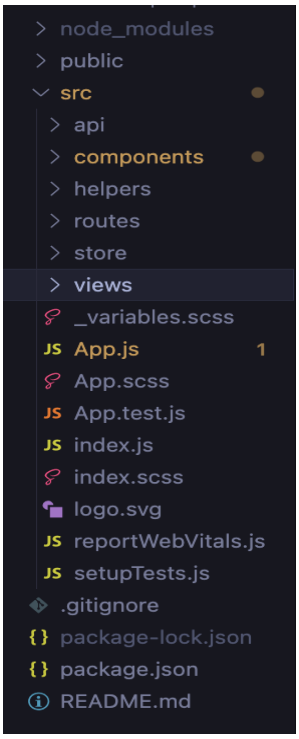
MANUAL TÉCNICO PROYECTO 2 FRONTEND 202003654

1. DESCIPCION GENERAL DE LA SOLUCION
    A. Para el desarrollo del frontend se utilizó la librería React, estructurando principalmente el uso de React+Redux+Axios para el manejo de datos, actualización y consumo del api.

2. REQUERIMIENTOS DEL SISTEMA
    A. NodeJs v14.21.1

3. ESTRUCTURA DEL PROYECTO

4. 

A. src
   i. Contiene todos los directorios principales.
B. api
   i. Cuenta con las configuraciones del api con axios, endpoints a consumir y un index para exportar todas las llamadas.
C. helpers
   i. Contiene las funciones básicas reutilizables en diferntes partes del proyecto.
D. store
   i. Guardar y modifica la estructura del reducer de la aplicación con sus reducers, su store y las acciones a ejecutar.
E. views
   i. Guarda los componentes que actúan como vistas y que contienen y orquestan los componentes de la aplicación.
F. components
   i. Almacena los componentes en su mayoría genéricos para diferentes vistas en la aplicación.
G. _variables.scss
   i. Almacena las variables globales a aplicar en las hojas de estilos
H. App.js
   i. Engloba las variables, estados y cualquier cosa que afecte directamente a toda la aplicación
I. App.scss
   i. Modifica directamente cualquier estilo en toda la aplicación.

5. INICIALIZAR EL PROYECTO

A. Instalar las librerías con npm install.
B. Validar la correcta ejecución del backend.
C. Ejecutar el comando npm start

6. LIBRERÍAS UTILIZADAS

A. Redux
   i. Para almacenar un estado global que sea accesible para todas las vistas y componentes y sean notificados de cambios con base a las llamadas realizadas.
B. React router

   i. Para el correcto manejo, redirección y protección de las rutas y sus vistas.

C. Axios
   i. Para manejar las peticiones realizadas por el usuario y manejo de posibles .errores.

D. Material UI
   i. Para el diseño de vistas y componentes previamente estilizados.

E. React-toastify
   i. Para la notificación al usuario de cambios realizados y su estado.

F. moment
   i. Para el manejo de fechas.

G. React-youtube
   i. Para la reproducción de los videos.

H. Saas y saas-loader
   i. Para el uso y lectura de hojas de estilos escritas en saas.

I. Papaparse
   i. Para la lectura y manejo de archivos ingresados por el usuario

J. Markdown-to-jsx
   i. Para la conversión y lectura de archivos jsx.

7. RUTAS

```jsx
const AppRoutes = () => {
  return (
    <Router>
      <MainMenu />
      <Routes>
        <Route path="/" element={<MainView />} />
        <Route path="/login" element={<LoginView />} />
        <Route path="/signup" element={<SignUpView />} />
        <Route path="/recover-password" element={<RecoverPasswordView />} />
        <Route element={<ProtectedRoute />}>
          <Route path="/dashboard" element={<DashboardView />} />
          <Route path="/my-account/:id" element={<MyAccountView />} />
          <Route path="/movie/:id" element={<MovieView />} />
          <Route path="/playlist" element={<MoviesPlaylistView />} />
        </Route>
        <Route element={<AdminProtectedRoute />}>
          <Route path="/create-movie" element={<CreateMovieView />} />
          <Route path="/edit-movie/:id" element={<EditMovieView />} />
          <Route path="/users" element={<UsersTableView />} />
          <Route path="/create-user" element={<AdminCreateUser />} />
        </Route>
      </Routes>
    </Router>      fernando.morales, 4 days ago • Fixing components routes and
  );
};
```

A. El router

   i. encapsula todas las rutas a utilizar.

B. ProtectedRoute

   i. encapsula las rutas que serán protegidas y visibles únicamente si hay una sesión iniciada y almacenada correctamente.

C. AdminProtectedRoute

   i. Encapsula las rutas que serán únicamente visibles si hay una sesión de administrador activa.

8. REDUCER

```
Fernando Morales Ralda, 14 hours ago | 1 author (Fernando Morales Ralda)
import { configureStore } from '@reduxjs/toolkit';
import loggedInReducer from './reducers/loggedInReducer';
import authReducer from './reducers/authReducer';
import moviesReducer from './reducers/moviesReducer';
import userMoviesPlaylistReducer from './reducers/userMoviesPlaylistReducer';

const store = configureStore({
  reducer: {
    loggedIn: loggedInReducer,
    auth: authReducer,
    movies: moviesReducer,
    userMoviesPlaylist: userMoviesPlaylistReducer,
  },
});

export default store;      Fernando Morales Ralda, 2 days ago • Adding login api call wi
```

A. LoggedIn
   i. Almacenará un booleano que indica si la sesión tiene una cuenta loggeada.
B. Auth
   i. almacenará toda la información del usuario
C. Movies
   i. Almacena las películas registradas en el sistema
D. userMoviesPlaylist
   i. Almacena la playlist de películas del usuario

9. REDUCERS

A. La estructura recomendada inicial para el estado reducer es, lo que almacenará, un estado de cargando de tipo booleano y un error en el que se guardará el error devuelto por el api.

B. Se recomienda utilizar un switch con las acciones deseadas para el reducer, agregar un default que retornará el state del reducer.

10. CONEXIÓN AL API

A. Para la conexión y estandarización del api se definió una constante que setea el base url para la instancia de axios y se retorna la instancia de axios con su configuración para el uso en diferntes endpoints y archivos