Desarrollo Web
Mayo- Agosto 2024
Ing. Ana Sol Arteaga

Introducción al desarrollo Web con Ant Design

Context

Para comenzar a trabajar con la sesión del usuario, vamos a separar un poco las responsabilidades en el formulario de Login y Register.

Crea src/services/auth.js

Dentro de ese archivo vamos a crear las solicitudes a la API de login y registro, de esta manera te debe de quedar tu archivo

src > services > auth.js > default

```
1   import axios from 'axios';
2
3   const register = async (username, email, password) => {
4       return axios.post('http://localhost:3000/api/auth/signup', {
5           username,
6           email,
7           password,
8           roles: ['user'],
9       });
10  };
11
12  const loginF = async (email, password) => {
13      return axios.post('http://localhost:3000/api/auth/signin', {
14          email,
15          password,
16      });
17
18  };
19
20  export default {
21      register,
22      loginF,
23  };
```

Después en cada uno de los formularios, has uso de estas funciones. Tanto en login como en register Importa → **import authService from '../../services/auth';**

onFinish en Login quedaría así

```
const onFinish = async (values) => {
    setLoading(true);
    setLoginError(false);
    try {
        const response = await authService.loginF(values.username, values.password);
        if (response && response.data) {
            localStorage.setItem('token', response.data.token);
            console.log(response.data.token)
            navigate('/');
        } else {
            console.error('Error en el inicio de sesión: Respuesta inesperada');
            setLoginError(true);
        }
    } catch (error) {
        console.error('Error en el inicio de sesión:', error.response ? error.response.data : error.message);
        setLoginError(true);
    } finally {
        setLoading(false);
    }
};
```

Y en register

```
17
18      const onFinish = async (values) => {
19          setLoading(true);
20          try {
21              await authService.register(values.username, values.email, values.password);
22              console.log('Registro exitoso');
23              navigate('/login');
24          } catch (error) {
25              console.error('Error en el registro:', error.response.data);
26              setRegisterError(true);
27          } finally {
28              setLoading(false);
29          }
30      };
31
```

Para tener aún más limpio nuestro formulario de Registro crea la carpeta utils/validation.js y lleva la función **validatePassword**

```
src > utils > JS validation.js > ...
1   export const validatePassword = ({ getFieldValue }) => ({
2       validator(_, value) {
3           if (!value || getFieldValue('password') === value) {
4               return Promise.resolve();
5           }
6           return Promise.reject(new Error('Las contraseñas no coincide
7       },
8   });
9
```

Importa en formRegister → **import { validatePassword } from '../../utils/validation.js';**

Context Auth

Crea src/context/AuthContext.jsx

Importa los siguientes elementos

```
import React, { useState, useEffect, createContext} from 'react';
```

Crea AuthContext

```
export const AuthContext = createContext();
```

Crea el provider

```
1   import React, { useState, useEffect, createContext } from 'react';
2
3
4   export const AuthContext = createContext();
5
6   export const AuthProvider = (props) => {
7       const { children } = props
8
9       const data = {
10          user: null,
11          login: () => console.log('login'),
12          logout: () => console.log('logout'),
13          upDateUser: () => console.log('update user')
14      }
15      return (
16          <AuthContext.Provider value={data}>
17              {children}
18          </AuthContext.Provider>)
19
20  }
21  ```
```

En App.js importa el AuthProvider

```
import { AuthProvider } from './src/context/AuthContext';
```

Y envuelve tu aplicación

```
import { AuthProvider } from './context/AuthContext'
import './App.css'

function App() {

  return (
    <AuthProvider>
      <ConfigProvider
        theme={{
          token: {
            colorPrimary: '#7A83E1',
          },
        }}
      >

        <BrowserRouter >
          <AppRoutes />
        </BrowserRouter >
      </ConfigProvider>
    </AuthProvider>
  )
}
```

Crear src/hooks/useAuth.js

src > hooks > JS useAuth.js > ...

```
import React, { useContext } from 'react';
import { AuthContext } from '../context/AuthContext';

export const useAuth = () => useContext(AuthContext);
```

Probar que useAuth esté disponible para todos los componentes hijos del contexto
En FormLogin

importar → **import { useAuth } from '../../hooks/useAuth';**

```
//manejo de estado de autenticación
const useAuthData = useAuth();
console.log(useAuthData);
```

Realiza un login y observa la consola

```
Context                                          index.jsx:16
▼ {user: null, login: f, logout: f, updateUser: f} i
  ▶ login: () => console.log("login")
  ▶ logout: () => console.log("logout")
  ▶ updateUser: () => console.log("updateUser")
    user: null
  ▶ [[Prototype]]: Object
```

En AuthProvider reemplaza el valor de user:null y asignar un valor por ejemplo ‘Ana’, modifica el componente **AppRoutes** importa useAuth, crea user de useAuth y crea la condicional.