Desarrollo Web
Mayo- Agosto 2024
Ing. Ana Sol Arteaga

**Introducción al desarrollo Web con Ant Design**

**Consumo de API para SignUp y Login**

Para llevar a cabo esta práctica e implementar registro e inicio de sesión consumiendo la API que creamos el cuatrimestre pasado, vamos a instalar la librería de **Axios** por medio de ***npm install axios*** una vez que se instala vamos a llamar este módulo en nuestro componente FormRegister para comenzar con el registro.

Vamos a necesitar dos manejos de estado, uno para en caso de error y el otro para gestionar el tiempo de carga del proceso, no olvides importar useState.

```
// Estado de error de registro
const [registerError, setRegisterError] = useState(false);
// Estado de carga
const [loading, setLoading] = useState(false);
```

Recuerda que dentro del formulario de Login, debes de tener el input asegure al usuario registrar la contraseña deseada, por eso tenemos que hacer una validación, para ver que ambos input sean iguales.

La constante validatePassword recibe como argumento getFieldValue que se utiliza para obtener el valor de un campo específico del formulario, dentro de la condicional **if (!value || getFieldValue('password') === value)** se verifican dos cosas:

- Si value es falso (es decir, si el campo de confirmación de contraseña está vacío).
- Si el valor del campo de confirmación de contraseña (value) es igual al valor del campo de contraseña principal (getFieldValue('password')).
- Si esta condición no se cumple, retorna un rejet de la promesa (lo que garantiza que esperara a que se cumpla el proceso anterior) con el mensaje de error

```
const validatePassword = ({ getFieldValue }) => ({
    validator(_, value) {
        if (!value || getFieldValue('password') === value) {
            return Promise.resolve();
        }
        return Promise.reject(new Error('Las contraseñas no coinciden'));
    },
});
```

El cuerpo de la solicitud que recibe la API es el siguiente, por lo que tendrás que agregar los campos necesarios en FormRegister para que cumpla con los parámetros necesarios.

```
{
    "username":"Moderator29",
    "email": "moderatorgera@test.com",
    "password":"88888888",
    "roles":[ "moderator"]

}
```

Entonces nuestro formulario de registro, agregar el campo Nombre y los campos Contraseña y repetir contraseña, quedaría de la siguiente manera, ya validando la función anterior, no olvides pasarle por parámetro la función **onFinishFailed.**

```
<Form
    name="normal_register"
    className="register-form"
    initialValues={{
        remember: true,
    }}
    onFinish={onFinish}
    onFinishFailed={onFinishFailed}


>
```

```
34    const onFinishFailed = (errorInfo) => {
35        console.log('Failed:', errorInfo);
36        setRegisterError(true);
37    };
```

```
<Form.Item
    name="username"
    rules={[
        {
            required: true,
            message: 'Por favor ingrese su usuario',
        },
    ]}
>
    <Input prefix={<UserOutlined className="site-form-item-icon" />} placeholder="Nombre" />
</Form.Item>
<Form.Item
    name="email"
    rules={[
        {
            required: true,
            message: 'Por favor ingrese su email',
        },
    ]}
>
    <Input prefix={<UserOutlined className="site-form-item-icon" />} placeholder="Email" />
</Form.Item>
<Form.Item
    name="password"
    rules={[
        {
            required: true,
            message: 'Por favor ingrese su contraseña',
        },
    ]}
>
    <Input.Password
        prefix={<LockOutlined className="site-form-item-icon" />}
        type="password"
        placeholder="Contraseña"
    />
</Form.Item>

<Form.Item
    name="password-repet"
    rules={[
        {
            required: true,
            message: 'Por favor repieta su contraseña',
        },
        ({ getFieldValue }) => validatePassword({ getFieldValue }),
    ]}
>
    <Input.Password
        prefix={<LockOutlined className="site-form-item-icon" />}
        type="password"
        placeholder="Repetir contraseña"
    />
</Form.Item>
```

Para la función onFinish que se ejecuta al enviar el formulario de “Registrar”, iniciamos con el estado de carga en true, hacemos uso de un try, catch y finally para el manejo correcto de la solicitud a través de axios, al ser el registro enviaremos la solicitud por el método post a la url donde está almacenada la API, o correrla de manera local, dentro de una función asíncrona, enviando los valores necesarios según la configuración de la API, mostrar en terminal el registro exitoso y navegar hacia ‘/login’ dentro de catch, mostramos el mensaje de error y ejecutamos la función ***setRegisterError(true);*** en finaly cambiamos el estado de carga a false.

En el Botón de registrar, en la propiedad loading ,le pasamos el valor del estado de carga del mismo nombre, y para finalizar, podemos implementar que se muestre el error al usuario en caso de error.

```
const onFinish = async (values) => {
    setLoading(true); // Establece el estado de carga a true al enviar el formulario
    try {
        const response = await axios.post('http://localhost:3000/api/auth/signup', {
            username: values.username,
            email: values.email,
            password: values.password,
            roles: ['user']
        });
        console.log('Registro exitoso:', response.data);
        navigate('/login');
    } catch (error) {
        console.error('Error en el registro:', error.response.data);
        setRegisterError(true);
    } finally {
        setLoading(false); // Establece el estado de carga a false después de recibir la respuesta
    }
};
```

```
<Form.Item>
    {registerError && <p style={{ color: 'red' }}>Falló el registro</p>}
    <Button type="primary" htmlType="submit" className="login-form-button" loading={loading}>
        Registrar
    </Button>
</Form.Item>
```

Implementar las rutas con useNavigate

```
import { useNavigate } from 'react-router-dom';

  const navigate = useNavigate();
```

**Login**

Para implementar el login, es una funcionalidad parecida al registro, lo primero que tienes que hacer es importar **axios** para realizar la solicitud e importar useState para el manejo de estado (error y carga).

```
const navigate = useNavigate();
const [loginError, setLoginError] = useState(false);
const [loading, setLoading] = useState(false); // Estado de carga
```

La función onFinish tiene la misma estructura que register, con la diferencia que solo se envían email y password, de acuerdo a la configuración de la API y se extrae el token de la respuesta y se almacena en localStorage

```
13      const onFinish = async (values) => {
14          setLoading(true); // Establece el estado de carga a true al enviar el formulario
15          try {
16              const response = await axios.post('http://localhost:3000/api/auth/signin', {
17                  email: values.username, // Para este caso, el email actúa como el nombre de usuario
18                  password: values.password
19              });
20              console.log('Inicio de sesión exitoso:', response.data);
21              localStorage.setItem('token', response.data.token); // Guarda el token en el almacenamiento local
22              navigate('/'); // Redirige al usuario a la página principal
23          } catch (error) {
24              console.error('Error en el inicio de sesión:', error.response.data);
25              setLoginError(true);
26          } finally {
27              setLoading(false); // Establece el estado de carga a false después de recibir la respuesta
28          }
29      };
```

No te olvides de actualizar la función **onFinishFailed** y utilizarla en el formulario

```
const onFinishFailed = (errorInfo) => {
    console.log('Failed:', errorInfo);
    setLoginError(true);
};
```

```
<Form
    name="normal_login"
    className="login-form"
    initialValues={{
        remember: true,
    }}
    onFinish={onFinish}
    onFinishFailed={onFinishFailed}
>
```

Para finalizar es necesario agregar la condicional en caso de error al botón de iniciar sesión y activar la propiedad loading del botón de inicio con el estado loading.

```
<Form.Item>
    {loginError && <p style={{ color: 'red' }}>Credenciales incorrectas. Inténtalo de nuevo.</p>}
    <Button type="primary" htmlType="submit" className="login-form-button" loading={loading}>
        Iniciar Sesión
    </Button>
</Form.Item>
```