ProjectManager | MERN

Registración de Usuarios

1. Crear el arhivo `src/hooks/useForm.jsx` con el siguiente código:

```
import React, {useState} from 'react'

export const useForm = (initialState={}) => {
    const [formValues, setFormValues] = useState(initialState);

    const handleInputChange = ({target}) =>{
        setFormValues({
            ...formValues,
            [target.name] : target.value
        })
    };

    const reset = () => {
        setFormValues(initialState)
    }

    return {
        formValues,
        handleInputChange,
        reset
    }
}
```

2. Implementar el `useForm()` en el componente `<Register/>`

```
  const {formValues, handleInputChange, reset} = useForm(
    {
      name: "",
      email: "",
      password: "",
      password2 : ""
    }
  );

  const {name,email,password, password2} = formValues;
```

3. Implementar en cada input los atributos `name`, `value` y `onChange`

- Ejemplo:

```
    name="name"
    value={name}
    onChange={handleInputChange}
```

4. Implementar el manejo de errores

- Completar el componente `<Alert/>` con el siguiente código

```
    <div className="bg-red-600 text-center p-3 rounded-md uppercase text-white
font-bold text-sm my-10">
        {msg}
    </div>
```

- Crear un estado en el componente `<Register/>` para mostrar y setear las alertas

```
  const [alert, setAlert] = useState({});
```

- Crear en el componente `<Register/>` el manejador de alertas

```
   const handleShowAlert = (msg) => {
    setAlert({
      msg
    })
    setTimeout(() => {
      setAlert({})
    }, 3000);
  }
```

- Implementar la validación de errores

```
if([name,email,password,password2].includes("")){
    handleShowAlert( "Todos los campos son obligatorios")
}

const exRegEmail = /^[^@]+@[^@]+\.[a-zA-Z]{2,}/;

if (!exRegEmail.test(email)) {
    handleShowAlert("El email tiene un formato inválido");
    return null;
}

if (password !== password2) {
```

```
    handleShowAlert("Las contraseñas no coinciden");
    return null;
}
```

5. Implementar CORS en el backend

¿Qué es el CORS? Intercambio de Recursos de Origen Cruzado (CORS) es una característica de seguridad del navegador que restringe las solicitudes HTTP de origen cruzado que se inician desde secuencias de comandos que se ejecutan en el navegador. ver más

- Crear la variable de entorno `URL_FRONTEND`
- Instalar **cors** en el *backend* para que el servidor permita peticiones desde otro dominio

```
npm i cors
```

- Requerirlo e implementarlo en el *entrypoint* `app.js`

```
const cors = require('cors');
const whiteList = [process.env.URL_FRONTEND];
const corsOptions = {
  origin : function(origin, callback){
    if(whiteList.includes(origin)){
      callback(null, true)
    }else {
      callback(new Error("Error de Cors"))
    }
  }
}
app.use(cors(corsOptions))
```

6. Implementar AXIOS y enviar el pedido a la API

- **Axios** es un Cliente HTTP basado en promesas para node.js y el navegador. Es isomorfico (= puede ejecutarse en el navegador y nodejs con el mismo código base). En el lado del servidor usa el modulo nativo http de node.js, mientras que en el lado del cliente (navegador) usa XMLHttpRequests.
    - Sitio Web de Axios
    - Documentación de Axios

- Crear la variable de entorno `VITE_URL_BACKEND`

```
VITE_URL_BACKEND=http://localhost:4000/api
```

Nota: Las variables de entorno siempre deben comenzar con VITE. --> ver documentación

- Crear el archivo `src/config/clientAxios.jsx`

```
import axios from 'axios';

const clientAxios = axios.create({
    baseURL : `${import.meta.env.VITE_URL_BACKEND}`
});

export default clientAxios;
```

- Crear un estado para que se deshabilite el botón de registar cuando se haya enviado el formulario, y luego implementar su valor en el atributo `disabled` del button.

```
  const [sending, setSending] = useState(false);
```

Instalar la librería *Sweet Alert 2* para mostrar una alerta informativa luego de que el registro sea exitoso.

```
npm i sweetalert2
```

- Crear un estructura `try... catch` para hacer el pedido a la API

```
    try {

      setSending(true)

      const {data} = await clientAxios.post('/auth/register',{
        name,
        email,
        password
      });

      setSending(false)

      Swal.fire({
        icon: 'info',
        title: 'Gracias por registrate!',
        text: data.msg,
      });

      reset()

    } catch (error) {
      console.error(error);
```

```
        handleShowAlert(error.response.data.msg);
        reset()
      }
```