ProjectManager | MERN

Confirmar cuenta de usuario

1. Importar en el componente `<ConfirmAccount/>` las siguientes dependencias y componentes:

```
import React, {useEffect, useState} from 'react';
import { clientAxios } from "../config/clientAxios";
import {Link, useParams, useNavigate} from 'react-router-dom';
import {Alert} from '../components/Alert';
import Swal from 'sweetalert2';
```

2. Obtener el *token* de la URL utilizando el hook **useParams** de *react router dom*

```
const params = useParams();
const {token} = params;
```

- IMPORTANTE: verificar que el link que se envía en el email de confirmación tenga el *token* Nota: El hook **useParams** devuelve un objeto de pares clave/valor de los parámetros dinámicos de la URL actual que coincidieron con la `<Route path>`. Ver más info

3. Crear el estado para manejar las alertas

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

4. Guadar a la ejecución del hook `useNavigate()` de *react router dom* para redirigir a la página de login

```
  const navigate = useNavigate()
```

4. Crear la función `handleShowAlert()` que se encargará mostrar los errores

```
  setAlert({
    msg
  });
```

5. Utilizando **useEffect** crear la petición a la API para confirmar la cuenta por medio de **axios**:

```
  useEffect(() => {

  const confirmAccount = async () => {
      try {
        const url = `/auth/checked?token=${token}`;
        const { data } = await clientAxios.get(url);

        Swal.fire({
          title: 'Felicitaciones',
          text : "Tu registración se ha completado con éxito",
          confirmButtonText: 'Iniciá sesión',
        }).then((result) => {
          if (result.isConfirmed) {
            navigate('/')
          }
        })

      } catch (error) {
        console.error(error.response);
        handleShowAlert(error.response.data.msg);
      }
    };

    confirmAccount();
  }, []);
```

- Más info:
  - **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
  - **useEffect** Acepta una función que contiene código imperativo, posiblemente código efectivo. La función pasada a useEffect se ejecutará después de que el renderizado es confirmado en la pantalla. Por defecto, los efectos se ejecutan después de cada renderizado completado, pero puede elegir ejecutarlo solo cuando ciertos valores han cambiado.
    - Documentación useEffect()

6. Agregar la siguiente lógica a la estructura del componente para que muestre las alertas

```
{alert.msg && (
    <div>
        <>
          <Alert {...alert} />
          <nav>
            <Link
              to={"/register"}
            >
```

```
              ¿No tenés una cuenta? Registrate
            </Link>
            <Link
              to={"/"}
            >
              ¿Estás registrado? Iniciá sesión
            </Link>
          </nav>
        </>
    </div>
)}
```

Resetear la contraseña

1. Importar en el componente `<ForgetPassword/>` las siguientes dependencias y componentes:

```
import React, {useState} from "react";
import { clientAxios } from "../config/clientAxios";
import { Link } from "react-router-dom";
import {Alert} from '../components/Alert';
import Swal from "sweetalert2";
```

2. Crear los estados para manejar las alertas y los datos recibidos en el formulario.

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

3. Crear la función (manejador) `handleSubmit()` con las validaciones pertinentes e implementarla en el evento `onSubmit={handleSubmit}` del formulario.

```
  const handleSubmit = async (e) => {
    e.preventDefault();
    if(!email){
      handleShowAlert("El email es obligatorio")
      return null
    }
  }

  const handleShowAlert = (msg) => {
    setAlert({
      msg
    });

    setTimeout(() => {
      setAlert({});
```

```
    }, 3000);
  }
```

4. Agregar los atributos `value` y `onChange` con sus repectivos valores al `<input/>` del formulario

```
  value={email}
  onChange={(e) => setEmail(e.target.value)}
```

5. Agregar en el manejador `handleSubmit()` la solicitud al servidor, con su correspondiente manejo de errores, y así obtener el token y para proceder a la recuperación de la contraseña.

```
    try {
      const {data} = await
clientAxios.post(`${import.meta.env.VITE_URL_BACKEND}/auth/send-token`,{
        email
      });

      Swal.fire({
        title: 'Revisá tu casilla de correo',
        text : data.msg,
        confirmButtonText: 'Entendido',
      })

      setEmail("")

    } catch (error) {
      console.error(error);
      handleShowAlert(error.response.data.msg);
      setEmail("")
    }
```

- *Nota: Desde el backend se enviará el mail con el token para resetar la contraseña.*

6. Importar en el componente `<RecoverPassword/>` las siguientes dependencias y componentes:

```
import React, {useState, useEffect} from 'react';
import {Link, useParams} from 'react-router-dom';
import { clientAxios } from "../config/clientAxios";
import {Alert} from '../components/Alert';
```

7. Crear los estados para manejar las alertas y la verificación del token, password y actualiación del mismo.

```
    const [alert, setAlert] = useState({});
    const [tokenChecked, setTokenChecked] = useState(false);
    const [password, setPassword] = useState("");
    const [changePassword, setChangePassword] = useState(false);
```

8. Mediante el hook `useParams()` obtener el token de la URL y guardar la ejecución del hook `useNavigate()` para utilizarla en la redirección

```
  const {token} = useParams()
  const navigate = useNavigate();
```

9. Crear el manejador para mostrar las alertas

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

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

10. Crear la petición a la API utilizando **useEffect()** y **Axios** para validar el token y mostrar el formulario

```
  useEffect(() => {

    const checkToken =  async () => {
      try {

        const {data} = await clientAxios.get(`/auth/reset-password?
token=${token}`);
        setTokenChecked(true)

      } catch (error) {
        console.log(error.response);
        handleShowAlert(error.response?.data.msg)
      }
    }

    checkToken()

  }, []);
```

11. Crear la función `handleSubmit()` e implementarla en el evento onSubmit del formulario: `onSubmit={handleSubmit}`

```
  const handleSubmit = async (e) => {
    e.preventDefault();

    if(!password) {
      handleShowAlert('El password es requerido')
      return null
    }

    try {
      const {data} = await clientAxios.post(`/auth/reset-password?token=${token}`,
{
        password
      });

      Swal.fire({
        icon: 'info',
        title: 'Contraseña reseteada!',
        text: data.msg,
        confirmButtonText : "Iniciá sesión",
        allowOutsideClick : false
      }).then(result => {
        if(result.isConfirmed){
          setPassword("");
          navigate('/')
        }
      })
    } catch (error) {
      console.error(error)
      handleShowAlert(error.response?.data.msg)
      setPassword("");

    }

  }
```

12. Refactorizar la estructura del componente para que el usuario pueda ingresar una nueva contraseña

```
 <>
      <h1>
        Reestablecé tu contraseña
      </h1>

      {
      alert.msg && <Alert {...alert}/>
    }
    {
```

```
        tokenChecked ?
        (
          <>
          <form action=""  noValidate onSubmit={handleSubmit}>
          <div className="my-5">
            <label htmlFor="password" >Nueva contraseña</label>
            <input
              id="password"
              type="password"
              placeholder="Escribí tu nueva contraseña"
              name={password}
              onChange={(e) => setPassword(e.target.value)}
              />
          </div>
          <button
            type="submit"
          >
            Resetear contraseña
          </button>
      </form>
           </>
        )
        :
        <nav className="md:flex md:justify-between">
        <Link
          to={"/register"}
          className=" text-sky-700 block text-center my-3 text-sm uppercase "

        >
          ¿No tenés una cuenta? Registrate
        </Link>
        <Link
          to={"/"}
          className=" text-sky-700 block text-center my-3 text-sm uppercase "

        >
          ¿Estás registrado? Iniciá sesión
        </Link>
      </nav>
      }
      </>
```