Change Password

Para realizar el cambio de contraseña es necesario crear un nuevo endpoint en tu Api, te lo dejo aquí para mayor eficiencia, recuerda que este va en tu Api en el archivo userController.js si tienes la misma estructura de archivos.

```
export const changePassword = async (req, res) => {
    try {
        const { currentPassword, newPassword } = req.body;
        const userId = req.params.userId;

        // Obtener el usuario por ID
        const user = await User.findById(userId);
        if (!user) {
            return res.status(404).json({ message: 'Usuario no encontrado' });
        }

        // Verificar la contraseña actual
        const isMatch = await bcrypt.compare(currentPassword, user.password);
        if (!isMatch) {
            return res.status(400).json({ message: 'Contraseña actual incorrecta' });
        }

        // Encriptar la nueva contraseña
        const encryptedPassword = await User.encryptPassword(newPassword);

        // Actualizar la contraseña del usuario
        user.password = encryptedPassword;
        await user.save();

        res.status(200).json({ message: 'Contraseña actualizada correctamente' });
    } catch (error) {
        res.status(500).json({ message: error.message });
    }
};
```

Ahora en el archivo user.routes.js crea la ruta para cambiar la contraseña, no te olvides de importar la función anterior

```
router.put('/:userId/change-password', authJwt.verifyToken, changePassword);
```

Crea la solicitud a este endPoint en services/users.js

```
const changePassword = async (token, currentPassword, newPassword) => {
    try {
        const decoded = jwtDecode(token);
        const userId = decoded.id;
        const url = `${ENV.API_URL}/${ENV.ENDPOINTS.USERS}/${userId}/change-password`;


        const response = await authFetch(url, {
            method: 'PUT',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ currentPassword, newPassword }),
        });

        if (!response.ok) {

            throw new Error(`HTTP error! status: ${response.status}`);
        }

        return await response.json();
    } catch (error) {
        console.error('Error en changePassword', error);
        throw error;
    }
}
```

Ahora vamos a crear los cambios necesarios en la interfaz, lo que accionara este nuevo evento será el Botón de cambiar contraseña que está en el Drawer.

Crearemos el componente ***ChangePassword*** que contendrá el el botón que abrirá el modal con el formulario para actualizar la contraseña, traé de Antd todas las funciones asociadas al modal, que permiten abrir y cerrar.

Como puedes ver el modal renderiza el componente ***FormChangePassword*** que también tendrás que crear e importar en este componentes, le pasamos por props la función handleCancel para cerrar el modal desde la solicitud en ***FormChangePassword.***

```
rc > components > ChangePassword > index.jsx > default
1   import React, { useState } from 'react';
2   import { Modal, Button } from 'antd';
3   import FormChangePassword from '../FormChangePassword';
4
5
6   const ChangePassword = () => {
7       const [isModalOpen, setIsModalOpen] = useState(false);
8
9       const showModal = () => {
10          setIsModalOpen(true);
11      };
12      const handleOk = () => {
13          setIsModalOpen(false);
14      };
15      const handleCancel = () => {
16          setIsModalOpen(false);
17      };
18
19      return (
20
21          <>
22              <Button type="primary" onClick={showModal}>
23                  Cambiar contraseña
24              </Button>
25              <Modal title="Cambiar contraseña"
26                  open={isModalOpen}
27                  onOk={handleOk}
28                  onCancel={handleCancel}
29                  footer={null}
30                  width={650}
31              >
32
33                  <FormChangePassword closeModal={handleCancel} />
34              </Modal>
35          </>
36      );
37  }
38
39  export default ChangePassword;
```

Dentro del ya mencionado ***FormChangePassword*** vamos a comenzar con los imports necesarios, incluye componentes de Antd, y funciones ya creadas anteriormente, como la validación de la contraseña, para asegurarnos que sean iguales, **userService** para acceder a la solicitud y storageController para llamar el token.

```
1  import React, { useState, useRef } from 'react';
2  import { Form, Input, Button, message } from 'antd';
3  import { validatePassword } from '../../utils/validation.js';
4  import { usersService } from '../../services/users.js';
5  import { storageController } from '../../services/token';
6
```

Seguido vamos a crear el estado de error y carga, así mismo formRef que nos servirá para limpiar el formulario crea las funciones onFinish que realizará la solicitud, por ahora solo muestra un mensaje con los valores del formulario y onFinishFailed para el manejo del error.

```
const formRef = useRef(null);

// Estado de error de cambio
const [changeError, setChangeError] = useState(false);
// Estado de carga
const [loading, setLoading] = useState(false);

const onFinish = async (values) => {
   console.log(values);
};

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

Dentro del formulario, estoy considerando tres campos, contraseña actual, contraseña nueva y repetir contraseña nueva, esta última usa la función de **validatePassword** para asegurarnos que sea igual al campo contraseña nueva, como lo hicimos anteriormente en registro

Entonces mi formulario con algunas modificaciones en cuanto estilos, debería de quedarte así .

```
<Form
    ref={formRef}
    onFinish={onFinish}
    onFinishFailed={onFinishFailed}
    labelCol={{ span: 8 }}
    wrapperCol={{ span: 14 }}
    layout="horizontal"
    initialValues={{}}
    style={{ maxWidth: 800, marginTop: 40 }}
>
    <Form.Item
        name="password-actual"
        label="Contraseña actual"
        rules={[{ required: true, message: 'Por favor ingresa tu contraseña actual' }]}
    >
        <Input.Password

            type="password"

        />
    </Form.Item>
    <Form.Item
        name="password-new"
        label="Contraseña nueva"
        rules={[
            {
                required: true,
                message: 'Por favor ingrese su contraseña',
            },
        ]}
    >
        <Input.Password

            type="password"

        />
    </Form.Item>

    <Form.Item
        name="password-repet"
        label="Repetir contraseña nueva"
        rules={[
            {
                required: true,
                message: 'Por favor repita su contraseña',
            },
            ({ getFieldValue }) => validatePassword({ getFieldValue }),
        ]}
    >
        <Input.Password

            type="password"

        />
    </Form.Item>
    {changeError && (
        <Form.Item wrapperCol={{ offset: 8, span: 14 }}>
            <div style={{ color: 'red' }}>Error al cambiar la contraseña. Por favor, inténtelo de nuevo.</div>
        </Form.Item>
    )}
    <Form.Item

        style={{ display: 'flex', justifyContent: 'center', marginTop: 40 }}
    >
        <Button type="primary" htmlType="submit" style={{ width: 250, fontSize: 16, height: 35 }} loading={loading}>
            Guardar
        </Button>
    </Form.Item>
</Form>
```

Ahora solo queda modificar la función onFinish para realizar la solicitud.

Observa que la función construida, primero llama y actualiza el estado de carga y error, dentro del try está la solicitud, donde se obtiene el token y se crea la constante con los datos que se envían en la solicitud currentPassword y newPassword una vez creados los parámetros que se envían en la solicitud se crea la constante response con la solicitud a **changePassword** se manda un mensaje de confirmación si la solicitud se llama la función closeModal que se recibe por props (no olvides pasarla al componente) y se limpia el formulario, al final se maneja el error.

```
const onFinish = async (values) => {
    setLoading(true);
    setChangeError(false);

    try {
        const token = await storageController.getToken();

        const { "password-actual": currentPassword, "password-repet": newPassword } = values;

        const response = await usersService.changePassword(token, currentPassword, newPassword);
        console.log('Password changed successfully', response);
        message.success('Contraseña cambiada correctamente');
        closeModal()
        formRef.current.resetFields();

    } catch (error) {
        console.error('Error changing password', error);
        setChangeError(true);
    } finally {
        setLoading(false);
    }
};
```