Introducción al desarrollo Web con Ant Design

DELETE

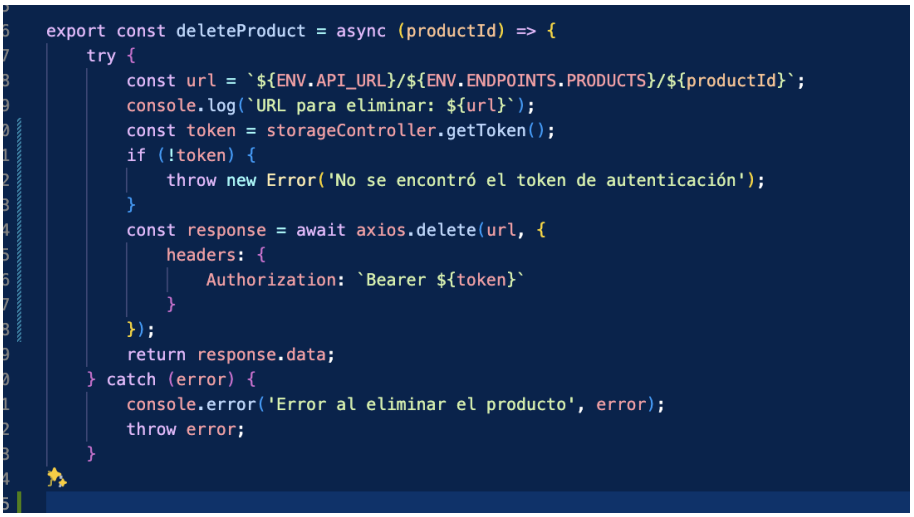
Para comenzar con el CRUD de productos, vamos a crear el componente CrudButton, para este componente use la propiedad Row y Col para cada uno de los botones que componen las acciones a realizar para la gestión de usuarios, cada columna renderiza un botón que muestra un icono para agregar, editar o eliminar, las propiedades de gutter y span es para manejar el tamaño y espaciado entre columna y fila.

En el Botón de eliminar en onClick se pasa la función onDelete que vamos a crear más adelante.

```jsx
import React from 'react';
import { Col, Row, Button } from 'antd';
import { PlusOutlined, EditOutlined, DeleteOutlined } from '@ant-design/icons';
import './CrudButton.css';

const CrudButton = ({ }) => {
    return (
        <div className='content-crud'>
            <Row gutter={32}>
                <Col span={6}>
                    <Button type="primary" icon={<PlusOutlined />} />
                </Col>
                <Col span={6}>
                    <Button
                        type="primary"
                        icon={<EditOutlined />}

                    />
                </Col>
                <Col span={6}>
                    <Button
                        type="primary"
                        icon={<DeleteOutlined />}

                        onClick={onDelete}
                    />
                </Col>
            </Row>
        </div>
    );
}


export default CrudButton;
```

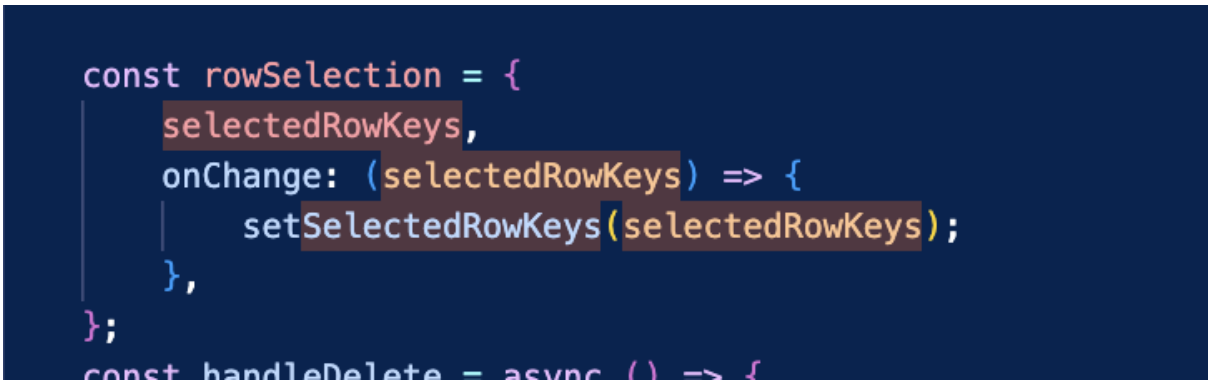
Lo que sigue es construir la solicitud a nuestro endpoint para eliminar un producto esto lo podemos hacer en el archivo de services/products.js

La función la llamamos deleProduct que recibe como parámetro el identificador, dentro de un try catch construimos la url de la solicitud concatenando nuestras variables de entorno más el productId, como es una solicitud que está protegida, nos traemos el token del storage y hacemos la validación de que si no se encuentra el token mande error de autenticación, una vez que se valide el token, realizamos la solicitud por medio de axios y enviando token en los Headers, manejo del error con el catch



```
export const deleteProduct = async (productId) => {
    try {
        const url = `${ENV.API_URL}/${ENV.ENDPOINTS.PRODUCTS}/${productId}`;
        console.log(`URL para eliminar: ${url}`);
        const token = storageController.getToken();
        if (!token) {
            throw new Error('No se encontró el token de autenticación');
        }
        const response = await axios.delete(url, {
            headers: {
                Authorization: `Bearer ${token}`
            }
        });
        return response.data;
    } catch (error) {
        console.error('Error al eliminar el producto', error);
        throw error;
    }
}
```

Ahora en mi vista de Productos, actualiza la función rowSelection



```
const rowSelection = {
    selectedRowKeys,
    onChange: (selectedRowKeys) => {
        setSelectedRowKeys(selectedRowKeys);
    },
};
const handleDelete = async () => {
```

Importamos la función **deleteProduct** y construimos handleDelete, dentro de esta función vamos a validar que exista un elemento seleccionado a través de **selectedRowKeys,** por ahora estamos mostrando un mensaje al usuario, pero más adelante vamos a deshabilitar el botón, una vez hecha la validación, construimos la solicitud extrayendo el id de la celda seleccionada y llamando a la **deleteProduct** que recibe como parámetro el id, observa que hay console.log para debug y verificar que efectivamente esté llegando el id de la celda seleccionada, para finalizar maneja el error adecuadamente.

```
59
60      const handleDelete = async () => {
61          if (selectedRowKeys.length !== 1) {
62              message.error('Selecciona exactamente un producto para eliminar.');
63              return;
64          }
65
66          try {
67              const productId = selectedRowKeys[0];
68              console.log('ID para eliminar:', productId);
69              await deleteProduct(productId);
70              message.success('Producto eliminado exitosamente.');
71              setProducts(products.filter(product => product.key !== productId));
72              setSelectedRowKeys([]);
73          } catch (error) {
74              console.error('Error al eliminar el producto', error);
75              message.error('Error al eliminar el producto.');
76          }
77      };
78
```

Ahora solo queda pasarle por props estas funciones creadas en el componente productos al componente CrudButton

El llamado que del componente quedaría de la siguiente manera, donde puedes observar que añadimos la propiedad **isSingleSelection** para asegurarnos que solo se seleccione un registro.

```
{user && (
    <CrudButton
        isSingleSelection={selectedRowKeys.length === 1}
        hasSelected={selectedRowKeys.length > 0}
        onDelete={handleDelete}
    />
)}
```

Y ahora mi componente CrudButton, recibe estas funciones por props, en el evento onClick llamamos la propiedad OnDelete, y en disabled **isSingleSelection** para que el botón se deshabilite.

```jsx
const CrudButton = ({ isSingleSelection, hasSelected, onDelete }) => {
    return (
        <div className='content-crud'>
            <Row gutter={32}>
                <Col span={6}>
                    <Button type="primary" icon={<PlusOutlined />} />
                </Col>
                <Col span={6}>
                    <Button
                        type="primary"
                        icon={<EditOutlined />}
                        disabled={!hasSelected}
                    />
                </Col>
                <Col span={6}>
                    <Button
                        type="primary"
                        icon={<DeleteOutlined />}
                        disabled={!isSingleSelection}
                        onClick={onDelete}
                    />
                </Col>
            </Row>
        </div>
    );
}

export default CrudButton;
```

La funcionalidad de eliminar ya está lista, solo, como experiencia de usuario tenemos que hacer la validación que le pida al usuario confirmar los cambios, vamos a utilizar **Popconfirm** de Antd, no olvides importarlo, crea dos estados , uno para abrir el Pop y otro para gestionar la carga

```jsx
const [open, setOpen] = useState(false);
const [confirmLoading, setConfirmLoading] = useState(false);
```

Seguido crea las siguientes funciones asociadas al Pop, abrir, cancelar y enviar la solicitud cuando el usuario da clic en Ok (más adelante cambia por eliminar).

handleOk activa el estado de carga, llama la función onDelete, y pasa el estado de abrir el Pop a false para cerrarlo, todo esto está envuelto en un setTimeout.

```
const showPopconfirm = () => {
    setOpen(true);
};
const handleCancel = () => {
    console.log('Clicked cancel button');
    setOpen(false);
};
const handleOk = () => {
    setConfirmLoading(true);
    setTimeout(() => {
        setOpen(false);
        setConfirmLoading(false);
        onDelete()
    }, 1000);

};
```

Tu botón de eliminar quedaría de la siguiente manera

```
<Col span={6}>
    <Popconfirm
        title="Eliminar producto"
        description="¿Estás seguro que quieres eliminar este producto?"
        open={open}
        onConfirm={handleOk}
        okButtonProps={{
            loading: confirmLoading,
        }}
        onCancel={handleCancel}
    >
        <Button
            type="primary"
            icon={<DeleteOutlined />}
            disabled={!isSingleSelection}
            onClick={showPopconfirm}
        />
    </Popconfirm>

</Col>
```