dashboard.tsx

```tsx
import React, {  useCallback, useState } from 'react';
import { View, Text, FlatList, StyleSheet, TouchableOpacity, Alert }
from 'react-native';
import { useNavigation, useFocusEffect, NavigationProp } from
'@react-navigation/native';
import {Product} from '../types/types';
import { RootStackParamList } from '../../App';


const Dashboard: React.FC = () => {
    const [products, setProducts] = useState<Product[]>([]);
    const [yaEjecutado, setYaEjecutado] = useState(false);
    const navigation =
useNavigation<NavigationProp<RootStackParamList>>();

    const fetchProducts = async () => {
        fetch('https://rn1nb289-9000.use2.devtunnels.ms/productos')
            .then((response) => response.json())
            .then((data) => setProducts(data))
            .catch((error) => console.error(error));
            console.log("se hizo la peticion");

    }


    useFocusEffect(
        useCallback(() => {
            if (!yaEjecutado) {
                const fetchData = async () => {
                    await fetchProducts();
                    // Después de la primera ejecución, establece
yaEjecutado a true
                    setYaEjecutado(true);
                };

                fetchData();
            }
            // No es necesario retornar una función de limpieza en este
caso
            // ya que solo queremos ejecutar el efecto una vez.
```

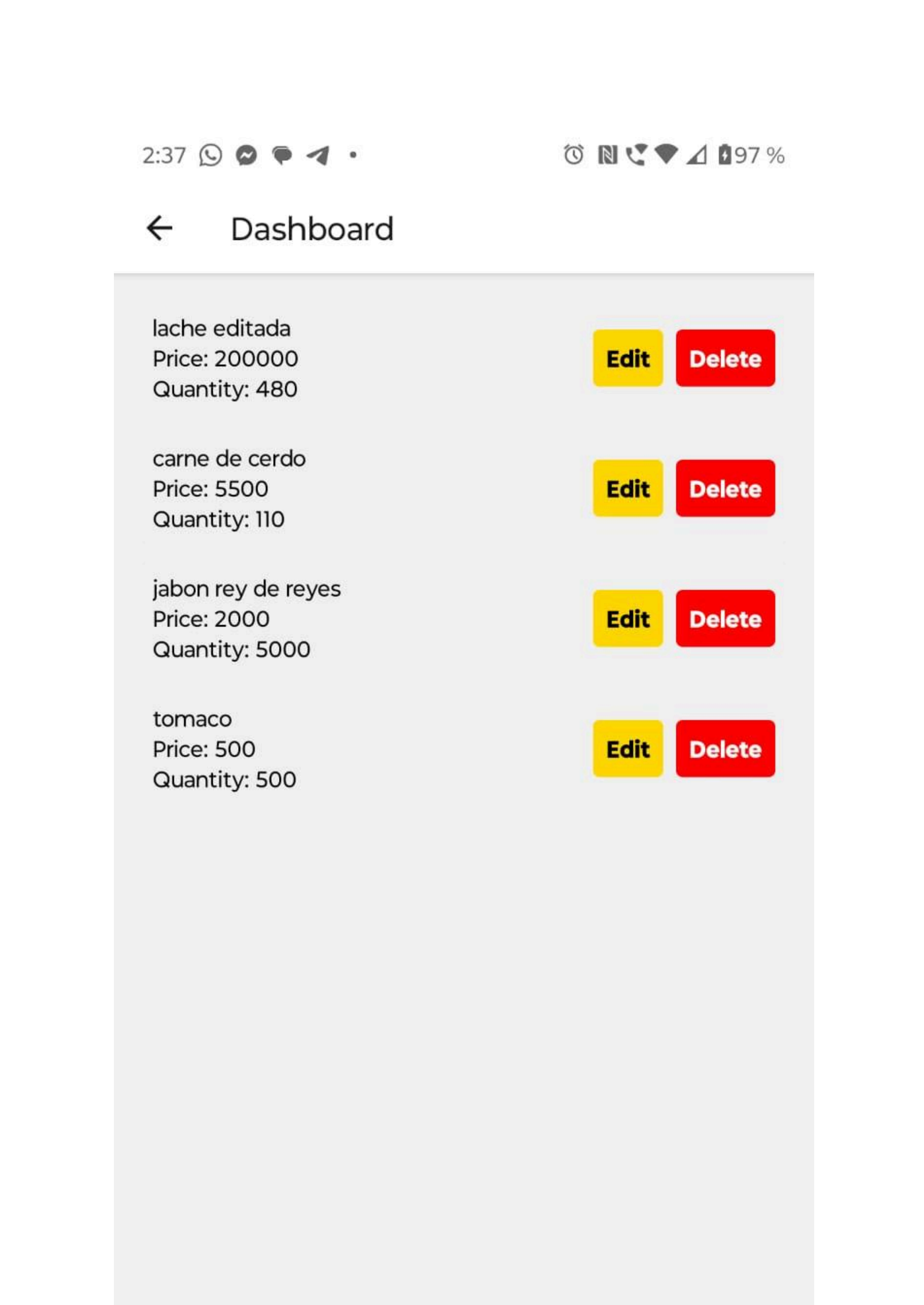
```
        }, [yaEjecutado, fetchProducts]) // Asegúrate de incluir
yaEjecutado en las dependencias
    );

    const renderProductItem = ({ item }: { item: Product }) => (
        <View style={styles.productItem}>
            <View style={styles.productDetails}>
                <Text>{item.Nombre}</Text>
                <Text>Price: {item.Precio}</Text>
                <Text>Quantity: {item.Cantidad}</Text>
            </View>

            <View style={styles.buttonContainer}>
                <TouchableOpacity style={styles.editButton} onPress={()
=> handleEdit(item)}>
                    <Text style={styles.buttonTextEdit}>Edit</Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.deleteButton}
onPress={() => handleDelete(item)}>
                    <Text style={styles.buttonText}>Delete</Text>
                </TouchableOpacity>
            </View>
        </View>
    );

    const handleEdit = (item: Product) => {
        // Handle edit logic here
        console.log('Edit', item);
        setYaEjecutado(false);
        navigation.navigate('ModalEdit', { product: item});
    };

    const handleDelete = (item: Product) => {
        // Handle delete logic here
        console.log('Delete', item);

        Alert.alert(
            'Confirm Delete',
            'Está segur@ de borrar este produto?',
            [
                {
                    text: 'Cancel',
                    onPress: () => console.log('Cancel Pressed'),
```

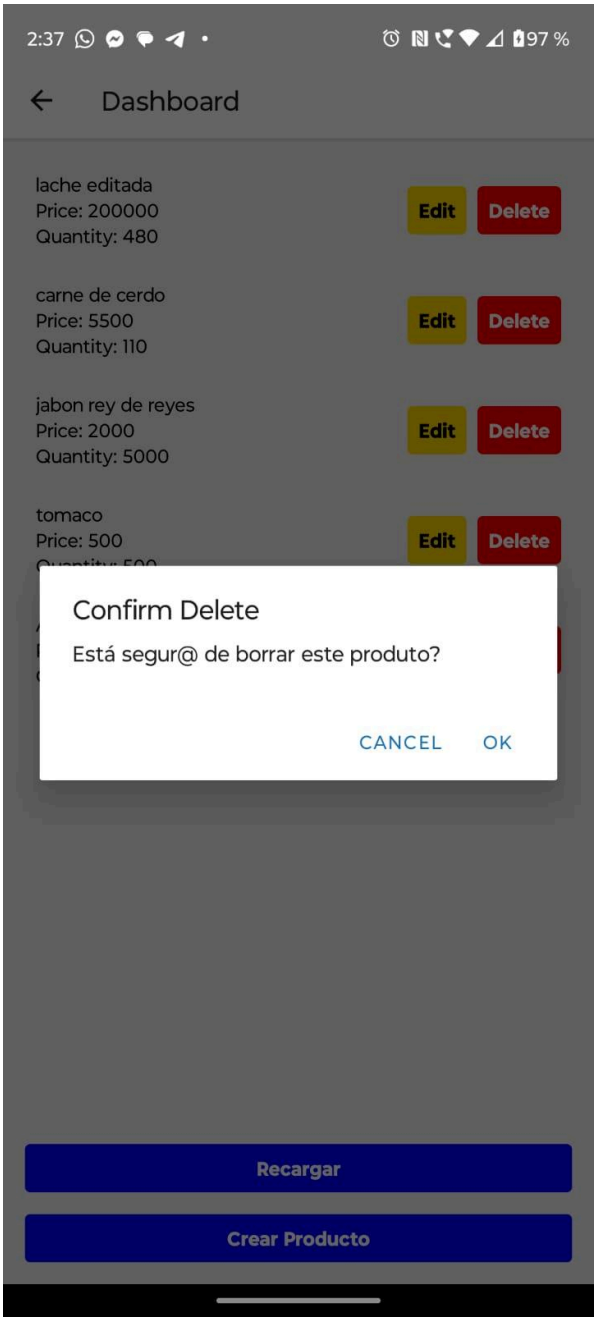
```
                    style: 'cancel',
                },
                {
                    text: 'OK',
                    onPress: () => {

fetch(`https://rn1nb289-9000.use2.devtunnels.ms/productos/del/${item.ID
_Producto}`, {
                            method: 'DELETE',
                        })
                        .then((response) => response.json())
                        .then(() => {
                            // Item deleted successfully
                            // You can update the state or perform any
other necessary actions
                            console.log('Item deleted successfully');
                            fetchProducts();
                        })
                        .catch((error) => console.error(error));
                    },
                },
            ],
            { cancelable: false }
        );
    };

    const handleCreate = () => {
        // Handle create logic here
        console.log('Create');
        setYaEjecutado(false);
        navigation.navigate('ModalCreate');
    }


    return (
        <View style={styles.container}>
            <FlatList
                data={products}
                renderItem={renderProductItem}
                keyExtractor={(item) => item.ID_Producto.toString()}
            />
            <View>
```

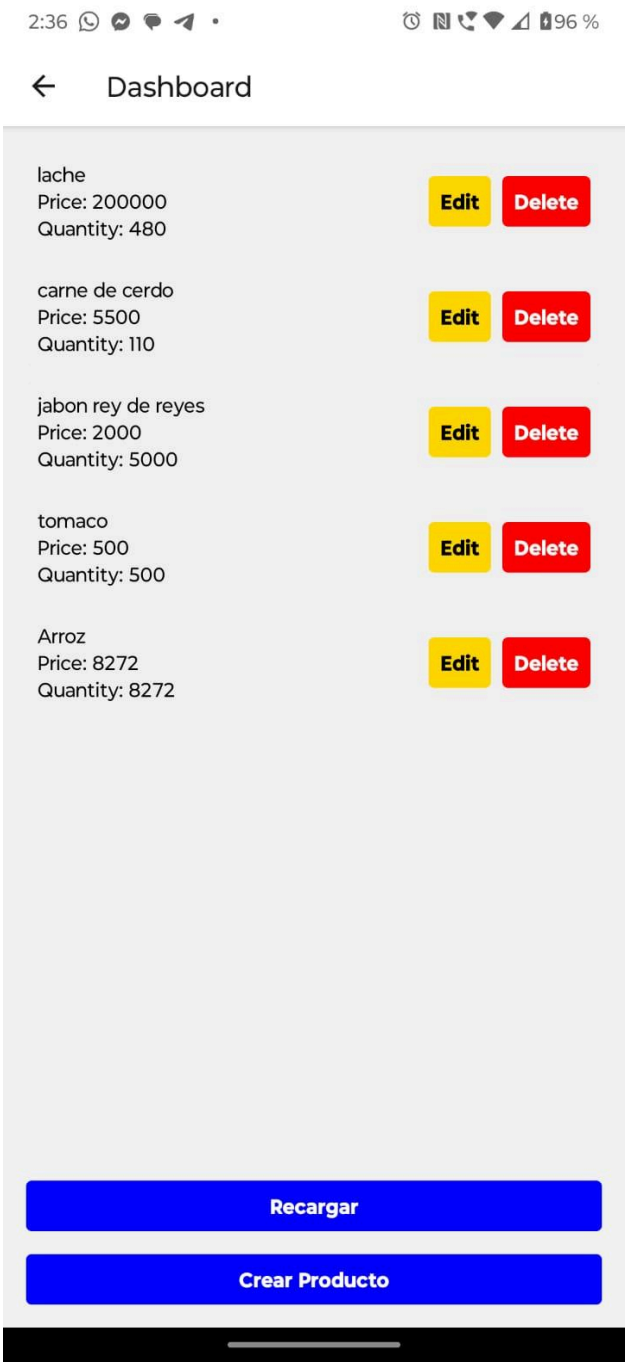
```
                <TouchableOpacity style={styles.reloadButton}
onPress={fetchProducts}>
                    <Text style={styles.buttonText}>Recargar</Text>
                </TouchableOpacity>
                <TouchableOpacity style={styles.reloadButton}
onPress={handleCreate}>
                    <Text style={styles.buttonText}>Crear
Producto</Text>
                </TouchableOpacity>
            </View>
        </View>

    );
};

const styles = StyleSheet.create({
    container: {
        flex: 3,
        padding: 16,
    },
    productDetails: {
        flex: 2, // Ajuste para ocupar el espacio restante
    },
    productItem: {
        flexDirection: 'row', // Cambio aquí para alinear los elementos
en fila
        justifyContent: 'space-between', // Asegura que los elementos
se distribuyan en el espacio disponible
        marginBottom: 8,
        padding: 8,
        backgroundColor: '#f0f0f0',
        borderRadius: 4,
    },
    buttonContainer: {
        flex: 1, // Ajuste para ocupar el espacio restante y empujar
los botones hacia la derecha
        flexDirection: 'row', // Asegura que los botones se alineen en
fila
        justifyContent: 'flex-end', // Alinea los botones hacia el
final (derecha)
        alignItems: 'center', // Centra los botones verticalmente
    },
    editButton: {
```

```
        backgroundColor: 'gold',
        padding: 8,
        borderRadius: 4,
        marginLeft: 8, // Añade un margen para separar los botones si
es necesario
    },
    deleteButton: {
        backgroundColor: 'red',
        padding: 8,
        borderRadius: 4,
        marginLeft: 8, // Añade un margen para separar los botones si
es necesario
    },
    buttonTextEdit: {
        color: 'black',
        fontWeight: 'bold',
    },
    buttonText: {
        color: 'white',
        fontWeight: 'bold',
    },
    reloadButton: {
        backgroundColor: 'blue',
        padding: 8,
        borderRadius: 4,
        alignItems: 'center',
        marginTop: 16,
    },
});

export default Dashboard;
```

se agregaron botones y funciones de editar, eliminar y crear productos y se actualiza haciendo peticiones a la api
en este mismo codigo se elimina


2:37
97 %
Dashboard
lache editada
Price: 200000
Quantity: 480
Edit
Delete
carne de cerdo
Price: 5500
Quantity: 110
Edit
Delete
jabon rey de reyes
Price: 2000
Quantity: 5000
Edit
Delete
tomaco
Price: 500
Quantity: 500
Edit
Delete


2:37
97 %
Dashboard
lache editada
Price: 200000
Quantity: 480
Edit
Delete
carne de cerdo
Price: 5500
Quantity: 110
Edit
Delete
jabon rey de reyes
Price: 2000
Quantity: 5000
Edit
Delete
tomaco
Price: 500
Edit
Delete
Confirm Delete
Está segur@ de borrar este produto?
CANCEL
OK
Recargar
Crear Producto



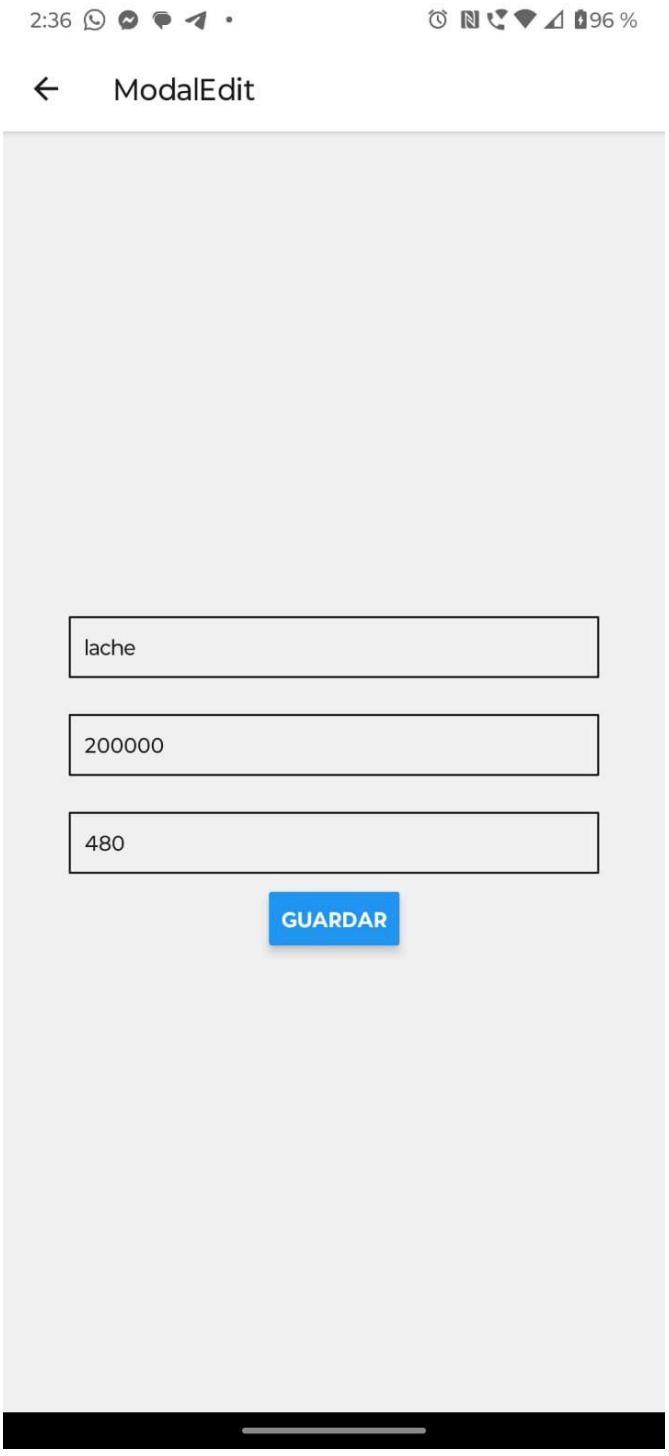
antes estaba el arroz y se elimino

codigo para editar

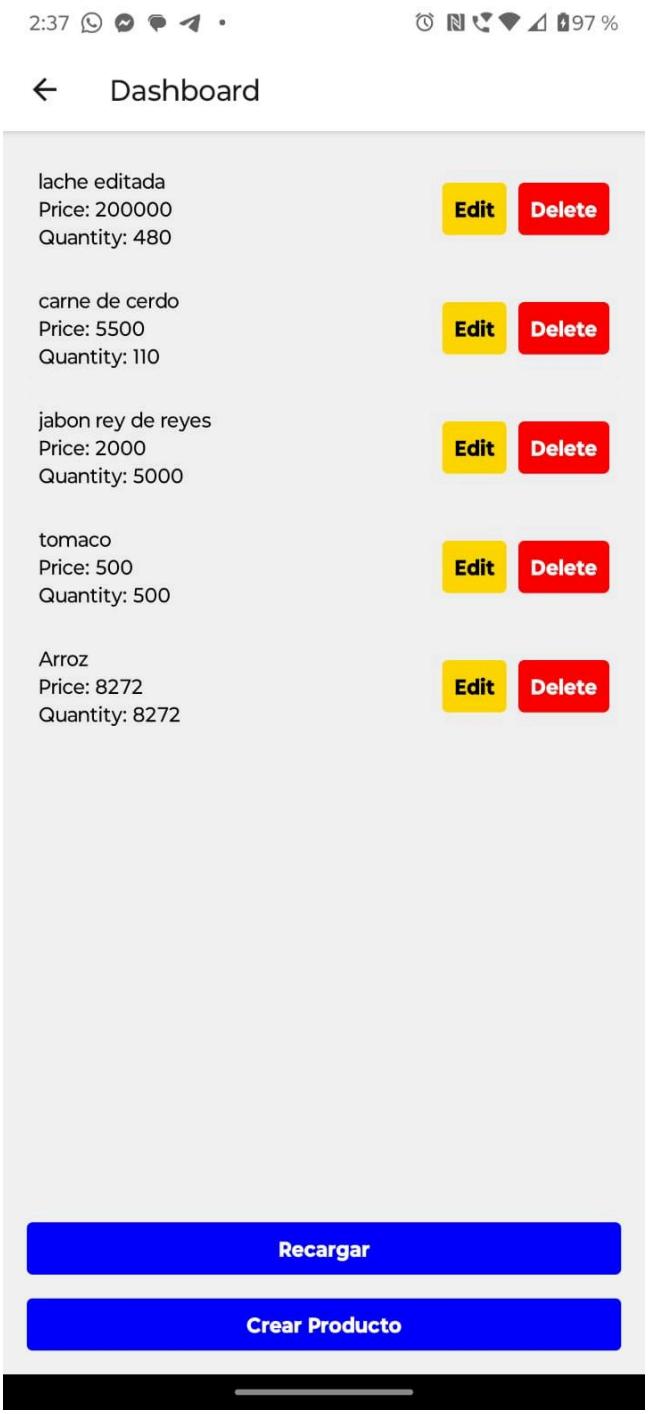
```
import React,{useState} from 'react';
import { View, TextInput, Button, StyleSheet } from 'react-native';
import { RouteProp, NavigationProp } from '@react-navigation/native';
 import { RootStackParamList } from '../../App';


type ModalEditRouteProp = RouteProp<RootStackParamList, 'ModalEdit'>;
type ModalEditNavigationProp = NavigationProp<RootStackParamList,
'ModalEdit'>;

type Props = {
    route: ModalEditRouteProp;
    navigation: ModalEditNavigationProp;
};

const ModalEdit: React.FC<Props> = ({ route, navigation }) => {
    const [product, setProduct] = useState(route.params.product);

    console.log('Producto:', product);



    const handleSave =async () => {
        console.log('Producto guardado', product);

fetch(`https://rn1nb289-9000.use2.devtunnels.ms/productos/editproduct/$
{product.ID_Producto}`, {
            method: 'PUT',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ producto: product}),
        }).then((response) => response.json()).then((data) => {
            console.log('Success:', data);
            navigation.goBack();
        }).catch((error) => {
            console.error('Error:', error);
        });
```

```
        // Aquí deberías implementar la lógica para guardar el producto
editado
    };

    return (
        <View style={styles.container}>
            <TextInput
                style={styles.input}
                value={product.Nombre}
                onChangeText={(text) => setProduct({ ...product,
Nombre: text })}
            />
            <TextInput
                style={styles.input}
                value={product.Precio.toString()}
                onChangeText={(text) => setProduct({ ...product,
Precio: Number(text) })}
            />
            <TextInput
                style={styles.input}
                value={product.Cantidad.toString()}
                onChangeText={(text) => setProduct({ ...product,
Cantidad: Number(text) })}
            />
            {/* Repite el TextInput para otras propiedades del producto
que desees editar */}
            <Button title="Guardar" onPress={handleSave} />
        </View>
    );
};

const styles = StyleSheet.create({
    container: {
        flex: 1,
        alignItems: 'center',
        justifyContent: 'center',
    },
    input: {
        height: 40,
        margin: 12,
        borderWidth: 1,
        padding: 10,
        width: '80%',
```

```
    },
});

export default ModalEdit;
```



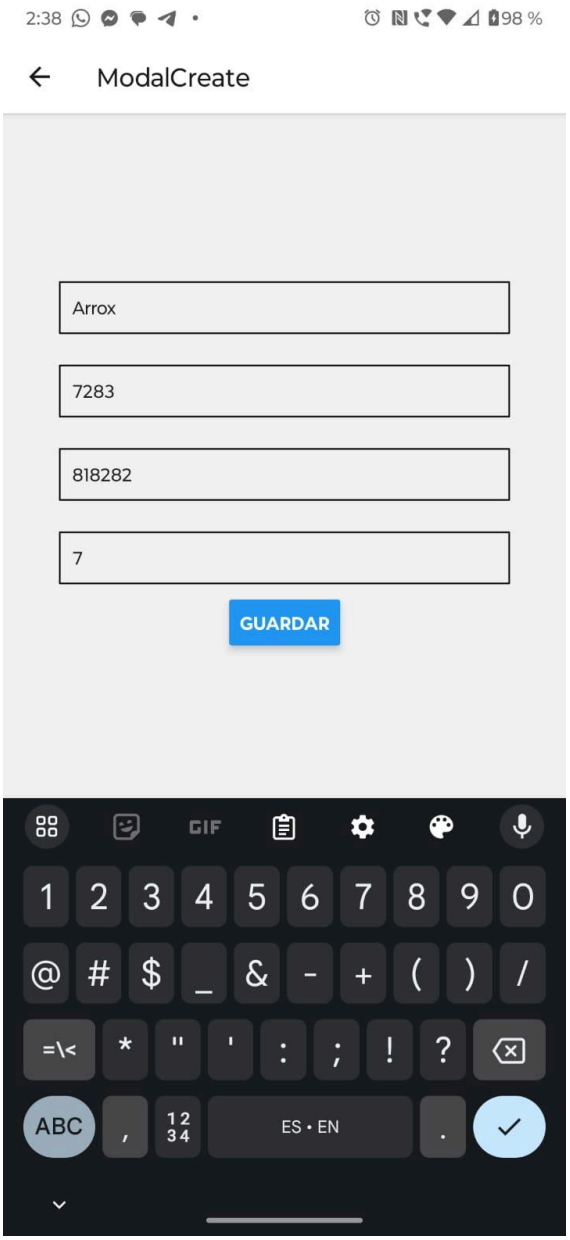
codigo para crear

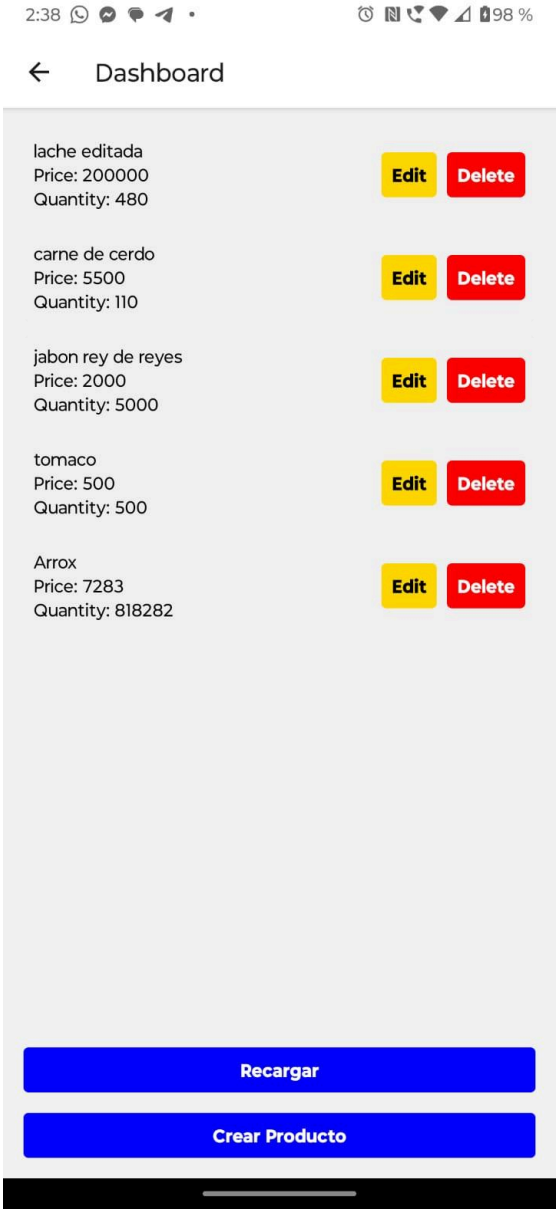



```
import { NavigationProp, useNavigation } from
'@react-navigation/native';
import React,{useState} from 'react';
import { View, TextInput, Button, StyleSheet } from 'react-native';
import { RootStackParamList } from '../../App';
```

```
const ModalCreate: React.FC = () => {
    const navigation =
useNavigation<NavigationProp<RootStackParamList>>();
    const [product, setProduct] = useState({
        Nombre: '',
        Precio: 0,
        Cantidad: 0,
        ID_categoria:0,
        // Agrega aquí otras propiedades del producto que desees editar
    });
    const handleSave =async () => {
        console.log('Producto guardado', product);
        fetch(`https://rn1nb289-9000.use2.devtunnels.ms/productos/`, {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ producto: product}),
        }).then((response) => response.json()).then((data) => {
            console.log('Success:', data);
            navigation.goBack();
        }).catch((error) => {
            console.error('Error:', error);
        });

        // Aquí deberías implementar la lógica para guardar el producto
editado
    };

    return (
        <View style={styles.container}>
            <TextInput
                style={styles.input}
                value={product.Nombre}
                onChangeText={(text) => setProduct({ ...product,
Nombre: text })}
            />
            <TextInput
                style={styles.input}
                value={product.Precio.toString()}
                onChangeText={(text) => setProduct({ ...product,
Precio: Number(text) })}
```

```
            />
            <TextInput
                style={styles.input}
                value={product.Cantidad.toString()}
                onChangeText={(text) => setProduct({ ...product,
Cantidad: Number(text) })}
            />
            <TextInput
                style={styles.input}
                value={product.ID_categoria.toString()}
                onChangeText={(text) => setProduct({ ...product,
ID_categoria: Number(text) })}
            />
            {/* Repite el TextInput para otras propiedades del producto
que desees editar */}
            <Button title="Guardar" onPress={handleSave} />
        </View>
    );
};

const styles = StyleSheet.create({
    container: {
        flex: 1,
        alignItems: 'center',
        justifyContent: 'center',
    },
    input: {
        height: 40,
        margin: 12,
        borderWidth: 1,
        padding: 10,
        width: '80%',
    },
});

export default ModalCreate;
```


2:38
98 %
ModalCreate
Arrox
7283
818282
7
GUARDAR
GIF
1 2 3 4 5 6 7 8 9 0
@ # $ _ & - + ( ) /
=\< * " ' : ; ! ?
ABC
ES • EN


2:38
98 %
Dashboard
lache editada
Price: 200000
Quantity: 480
Edit
Delete
carne de cerdo
Price: 5500
Quantity: 110
Edit
Delete
jabon rey de reyes
Price: 2000
Quantity: 5000
Edit
Delete
tomaco
Price: 500
Quantity: 500
Edit
Delete
Arrox
Price: 7283
Quantity: 818282
Edit
Delete
Recargar
Crear Producto