Ing. Mauricio Lara ParraSENA - ceet

DESARROLLO DE APPS CON REACT NATIVE

CREAR UN CRUD

1. instala Expo CLI globalmente: npm install -g expo-cli
2. Ejecuta el siguiente comando para crear un nuevo proyecto con Expo y TypeScript:

npx create-expo-app nombre de mi app --template blank-typescript

1. **Navegar al directorio del proyecto**

cd nombre de mi app

1. Crea una carpeta para el backend y navega a ella:

cd backend

1. **Inicializar el proyecto de Node.js ejecute desde la terminal**

npm init -y

1. **Instalar dependencias necesarias**

npm install express mysql2 cors

1. **Crear el archivo index.js**

const express = require('express');

const mysql = require('mysql2');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(express.json());

const db = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: '',

  database: 'products\_db'

});

db.connect((err) => {

  if (err) throw err;

  console.log('Connected to MySQL');

});

// CRUD endpoints

app.get('/products', (req, res) => {

  db.query('SELECT \* FROM products', (err, results) => {

    if (err) throw err;

    res.json(results);

  });

});

app.post('/products', (req, res) => {

  const { name, price } = req.body;

  db.query('INSERT INTO products (name, price) VALUES (?, ?)', [name, price], (err, results) => {

    if (err) throw err;

    res.json({ id: results.insertId, name, price });

  });

});

app.put('/products/:id', (req, res) => {

  const { id } = req.params;

  const { name, price } = req.body;

  db.query('UPDATE products SET name = ?, price = ? WHERE id = ?', [name, price, id], (err, results) => {

    if (err) throw err;

    res.json({ id, name, price });

  });

});

app.delete('/products/:id', (req, res) => {

  const { id } = req.params;

  db.query('DELETE FROM products WHERE id = ?', [id], (err, results) => {

    if (err) throw err;

    res.json({ id });

  });

});

app.listen(3000, () => {

  console.log('192.168.1.13 port 3000');

});

1. Conectar el Frontend con el Backend
2. Ingrese a la carpeta del proyecto desde la terminal
3. Instalar Axios en el proyecto de React Native

npm install axios

1. Crear el archivo api.js desde la raíz del proyecto

import axios from 'axios';

const api = axios.create({

  baseURL: 'http://192.168.1.13:3000'

});

export const getProducts = async () => {

  const response = await api.get('/products');

  return response.data;

};

export const createProduct = async (product) => {

  const response = await api.post('/products', product);

  return response.data;

};

export const updateProduct = async (id, product) => {

  const response = await api.put(`/products/${id}`, product);

  return response.data;

};

export const deleteProduct = async (id) => {

  const response = await api.delete(`/products/${id}`);

  return response.data;

};

1. Para ver reporte de productos instalar la librería

npm install react-native-modal

1. Crear el archivo ProductsScreen.tsx

import React, { useState, useEffect } from 'react';

import { View, Text, FlatList, Button, TextInput, StyleSheet,  TouchableOpacity } from 'react-native';

import { getProducts, createProduct, updateProduct, deleteProduct } from './api';

import Modal from 'react-native-modal';

interface Product {

  id: number;

  name: string;

  price: number;

}

const ProductsScreen = () => {

  const [products, setProducts] = useState<Product[]>([]);

  const [name, setName] = useState('');

  const [price, setPrice] = useState('');

  const [selectedProductId, setSelectedProductId] = useState<number | null>(null);

  const [isModalVisible, setModalVisible] = useState(false);

  useEffect(() => {

    fetchProducts();

  }, []);

  const fetchProducts = async () => {

    const data = await getProducts();

    setProducts(data);

  };

  const handleAddProduct = async () => {

    const newProduct = await createProduct({ name, price });

    setProducts([...products, newProduct]);

    setName('');

    setPrice('');

  };

  const handleUpdateProduct = async () => {

    if (selectedProductId !== null) {

      const updatedProduct = await updateProduct(selectedProductId, { name, price });

      setProducts(products.map(product => product.id === selectedProductId ? updatedProduct : product));

      setSelectedProductId(null);

      setName('');

      setPrice('');

    }

  };

  const handleDeleteProduct = async (id: number) => {

    await deleteProduct(id);

    setProducts(products.filter(product => product.id !== id));

  };

  const toggleModal = () => {

    setModalVisible(!isModalVisible);

  };

  return (

    <View style={styles.container}>

      <Text style={styles.title}>Productos</Text>

      <FlatList

        data={products}

        keyExtractor={(item) => item.id.toString()}

        renderItem={({ item }) => (

          <View style={styles.productContainer}>

            <Text style={styles.productText}>{item.name} - ${item.price}</Text>

            <View style={styles.buttonContainer}>

              <Button title="Modificar" onPress={() => {

                setName(item.name);

                setPrice(item.price.toString());

                setSelectedProductId(item.id);

              }} />

              <Button title="Eliminar" onPress={() => handleDeleteProduct(item.id)} />

            </View>

          </View>

        )}

      />

      <TextInput

        style={styles.input}

        placeholder="Nombre"

        value={name}

        onChangeText={setName}

      />

      <TextInput

        style={styles.input}

        placeholder="Precio"

        value={price}

        onChangeText={setPrice}

        keyboardType="numeric"

      />

      <Button

        title={selectedProductId ? "Modificar Producto" : "Crear Producto"}

        onPress={selectedProductId ? handleUpdateProduct : handleAddProduct}

      />

      <Button title="Crear Reporte" onPress={toggleModal} />

      <Modal isVisible={isModalVisible}>

        <View style={styles.modalContent}>

          <Text style={styles.modalTitle}>Productos Reporte</Text>

          {products.map(product => (

            <Text key={product.id} style={styles.modalText}>{product.name} - ${product.price}</Text>

          ))}

          <TouchableOpacity onPress={toggleModal} style={styles.closeButton}>

            <Text style={styles.closeButtonText}>Close</Text>

          </TouchableOpacity>

        </View>

      </Modal>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    padding: 20,

    backgroundColor: '#f5f5f5',

  },

  title: {

    fontSize: 24,

    fontWeight: 'bold',

    marginBottom: 20,

    textAlign: 'center',

  },

  productContainer: {

    padding: 15,

    backgroundColor: '#fff',

    borderRadius: 10,

    marginBottom: 10,

    shadowColor: '#000',

    shadowOffset: { width: 0, height: 2 },

    shadowOpacity: 0.1,

    shadowRadius: 5,

    elevation: 3,

  },

  productText: {

    fontSize: 18,

    marginBottom: 10,

  },

  buttonContainer: {

    flexDirection: 'row',

    justifyContent: 'space-between',

  },

  input: {

    height: 40,

    borderColor: '#ccc',

    borderWidth: 1,

    borderRadius: 5,

    marginBottom: 10,

    paddingHorizontal: 10,

    backgroundColor: '#fff',

  },

  modalContent: {

    backgroundColor: 'white',

    padding: 20,

    borderRadius: 10,

    alignItems: 'center',

  },

  modalTitle: {

    fontSize: 20,

    fontWeight: 'bold',

    marginBottom: 10,

  },

  modalText: {

    fontSize: 16,

    marginBottom: 5,

  },

  closeButton: {

    marginTop: 20,

    padding: 10,

    backgroundColor: '#2196F3',

    borderRadius: 5,

  },

  closeButtonText: {

    color: 'white',

    fontWeight: 'bold',

  },

});

export default ProductsScreen;

1. Crear la base de datos

CREATE DATABASE products\_db;

1. Luego, crea la tabla products dentro de la base de datos:

USE products\_db;

CREATE TABLE products (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255) NOT NULL,

price DECIMAL(10, 2) NOT NULL

);

1. Modificar App.tsx

import React from 'react';

import { NavigationContainer } from '@react-navigation/native';

import { createStackNavigator } from '@react-navigation/stack';

import ProductsScreen from './ProductsScreen';

const Stack = createStackNavigator();

const App = () => {

  return (

    <NavigationContainer>

      <Stack.Navigator initialRouteName="Products">

        <Stack.Screen name="Productos" component={ProductsScreen} />

      </Stack.Navigator>

    </NavigationContainer>

  );

};

export default App;

1. Para ver el reporte de productos instalar la librería