LARAUSESENA - ceet

DESARROLLO DE APPS CON REACT NATIVE

INTERGRAR CRUD CON LA APLICACIÓN

1. Modifique server.js asi :

const express = require('express');

const passport = require('passport');

const app = express();

const http = require('http');

const server = http.createServer(app);

const logger = require('morgan');

const cors = require('cors');

const mysql = require('mysql2');

const db  = require('./config/config.js');

/\*\*

 \* Importar las rutas

 \*/

const userRoutes = require('./routes/userRoutes');

const port = process.env.PORT || 3000;

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extendd: true}));

app.use(cors());

app.use(passport.initialize());

app.use(passport.session());

require('./config/passport')(passport);

app.disable('x-powered-by');//Disable the x-powered-by header in the response

app.set('port',port);

/\*\*

 \* Llamado las rutas

 \*/

userRoutes(app);

// Iniciando el servidor

// direccion ip V4 de la maquina consultar ipconfig

server.listen(3000, '192.168.1.13' || 'localhost', function() {

    console.log('Aplicacion de NodeJs' + process.pid + 'ejecutando en ' + server.address().address + ' : ' + server.address().port);

});

/\*\*Rutas \*/

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

    res.send('estas en la Ruta raiz de Backend');

});

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

    res.send('estas en la Ruta test');

});

//Manejo de errores

//Error handler

app.use((err, req, res, next) => {

    console.log(err);

    res.status(err.status || 500).send(err.stack);

});

// CRUD endpoints

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

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

      if (err) throw err;

      res.json(results);

    });

  });

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

    const { id } = req.params;

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

      if (err) throw err;

      res.json(results[0]);

    });

  });

  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. Cree el archivo api.js en Data/sourse/remote/api queda así:

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;

};

export const getProductById = async (id) => {

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

  return response.data;

};

1. En la carpeta views/profile/info modifique profileInfo.tsx asi:

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

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

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

import { RootStackParamList } from '../../../../../App';

import useViewModel from './ViewModel';

import { getProducts, createProduct, updateProduct, deleteProduct, getProductById } from '../../../../Data/sourse/remote/api/api';

import Modal from 'react-native-modal';

import styles from './Styles';

interface Product {

  id: number;

  name: string;

  price: number;

}

interface Props extends StackScreenProps<RootStackParamList, 'ProfileInfoScreen'> { };

export const ProfileInfoScreen = ({ navigation, route }: Props) => {

  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);

  const [searchId, setSearchId] = useState('');

  const [searchedProduct, setSearchedProduct] = useState<Product | null>(null);

  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 handleSearchProduct = async () => {

    const product = await getProductById(parseInt(searchId));

    setSearchedProduct(product);

  };

  const toggleModal = () => {

    setModalVisible(!isModalVisible);

  };

    const { removeSession } = useViewModel();

    return (

        //<View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>

        <View style={{ flex: 1, justifyContent: 'flex-end',  }}>

          <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}>

              <TouchableOpacity style={styles.button} onPress={() => {

                setName(item.name);

                setPrice(item.price.toString());

                setSelectedProductId(item.id);

              }}>

                <Text style={styles.buttonText}>Modificar</Text>

              </TouchableOpacity>

              <TouchableOpacity style={styles.button} onPress={() => handleDeleteProduct(item.id)}>

                <Text style={styles.buttonText}>Eliminar</Text>

              </TouchableOpacity>

            </View>

          </View>

        )}

      />

      <TextInput

        style={styles.input}

        placeholder="Nombre"

        value={name}

        onChangeText={setName}

      />

      <TextInput

        style={styles.input}

        placeholder="Precio"

        value={price}

        onChangeText={setPrice}

        keyboardType="numeric"

      />

      <TouchableOpacity style={styles.addButton} onPress={selectedProductId ? handleUpdateProduct : handleAddProduct}>

        <Text style={styles.addButtonText}>{selectedProductId ? "Modificar Producto" : "Crear Producto"}</Text>

      </TouchableOpacity>

      <TextInput

        style={styles.input}

        placeholder="Numero de ID"

        value={searchId}

        onChangeText={setSearchId}

        keyboardType="numeric"

      />

      <TouchableOpacity style={styles.searchButton} onPress={handleSearchProduct}>

        <Text style={styles.searchButtonText}>Buscar Producto</Text>

      </TouchableOpacity>

      {searchedProduct && (

        <View style={styles.productContainer}>

          <Text style={styles.productText}> Producto Buscado: {searchedProduct.name} - ${searchedProduct.price}</Text>

        </View>

      )}

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

        <Text style={styles.reportButtonText}>Generar Reporte</Text>

      </TouchableOpacity>

      <Modal isVisible={isModalVisible}>

        <View style={styles.modalContent}>

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

          </TouchableOpacity>

        </View>

      </Modal>

    </View>

            <Button

                onPress={() => {

                    removeSession();

                    navigation.navigate('HomeScreen');

                }}

                title="Cerrar Sesion"

            />

        </View>

    )

}

1. En la carpeta views/profile/info cree el archivo Styles.tsx y queda así :

import { StyleSheet } from "react-native";

const ProfileInfostyles = 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',

    },

    button: {

      backgroundColor: '#2196F3',

      padding: 10,

      borderRadius: 5,

      marginHorizontal: 5,

    },

    buttonText: {

      color: 'white',

      fontWeight: 'bold',

    },

    input: {

      height: 40,

      borderColor: '#ccc',

      borderWidth: 1,

      borderRadius: 5,

      marginBottom: 10,

      paddingHorizontal: 10,

      backgroundColor: '#fff',

    },

    addButton: {

      backgroundColor: '#4CAF50',

      padding: 15,

      borderRadius: 5,

      alignItems: 'center',

      marginBottom: 10,

    },

    addButtonText: {

      color: 'white',

      fontWeight: 'bold',

    },

    searchButton: {

      backgroundColor: '#FF9800',

      padding: 15,

      borderRadius: 5,

      alignItems: 'center',

      marginBottom: 10,

    },

    searchButtonText: {

      color: 'white',

      fontWeight: 'bold',

    },

    reportButton: {

      backgroundColor: '#9C27B0',

      padding: 15,

      borderRadius: 5,

      alignItems: 'center',

      marginBottom: 10,

    },

    reportButtonText: {

      color: 'white',

      fontWeight: 'bold',

    },

    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',

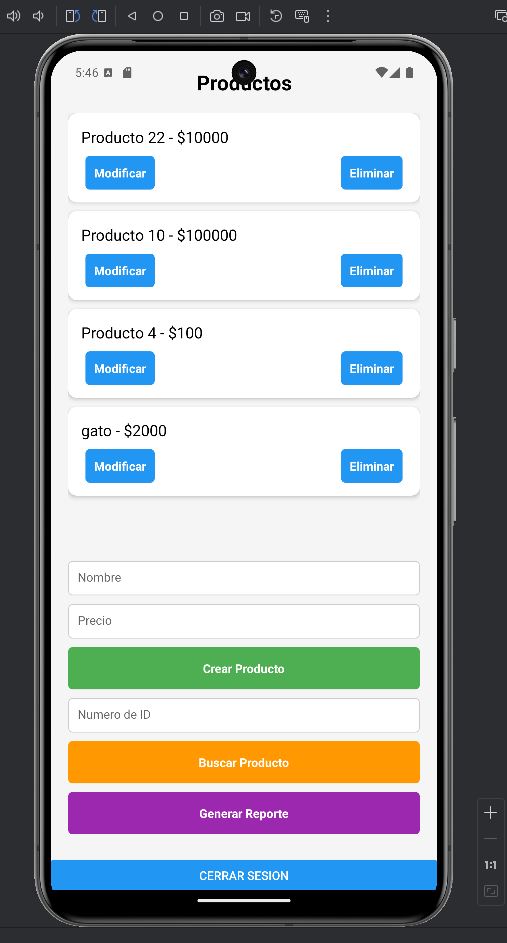
      fontWeight: 'bold',

    },

  });

  export default ProfileInfostyles;

1. Debe quedar asi:



1. Realice las pruebas por backend por postman de la misma manera que las guias anteriores para garantizar la api y después por el emulador