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

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

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

import { db } from '../database/AbreConexao';

export default function ListarProdutos({ navigation }) {

const [produtos, setProdutos] = useState([]);

const [produtoSelecionado, setProdutoSelecionado] = useState(null);

const [itemSelecionado, setSelectedItem] = useState(null);

const [totalProdutos, setTotalProdutos] = useState(0);

const [totalEstoqueMinimo, setTotalEstoqueMinimo] = useState(0);

const [filtroNomeProduto, setFiltroNomeProduto] = useState('');

const [filtroEstoqueMinimo, setFiltroEstoqueMinimo] = useState(false);

useEffect(() => {

carregarProdutos();

}, [filtroEstoqueMinimo, filtroNomeProduto]);

useFocusEffect(

React.useCallback(() => {

carregarTotais();

setFiltroEstoqueMinimo(false);

setFiltroNomeProduto('');

carregarProdutos();

}, [])

);

const carregarTotais = () => {

db.transaction((transaction) => {

transaction.executeSql(

`SELECT COUNT(\*) AS total FROM produtos`,

[],

(\_, { rows }) => {

const totalProdutos = rows.\_array[0].total;

setTotalProdutos(totalProdutos);

},

(\_, error) => {

console.log('Erro ao carregar total de produtos: ' + error);

}

);

transaction.executeSql(

`SELECT COUNT(\*) AS total FROM produtos

INNER JOIN estoque ON produtos.id\_produto = estoque.id\_produto

WHERE estoque.estoque\_atual <= produtos.estoque\_minimo`,

[],

(\_, { rows }) => {

const totalEstoqueMinimo = rows.\_array[0].total;

setTotalEstoqueMinimo(totalEstoqueMinimo);

},

(\_, error) => {

console.log('Erro ao carregar total de produtos com estoque mínimo: ' + error);

}

);

});

};

const carregarProdutos = () => {

db.transaction((transaction) => {

let query = `SELECT p.id\_produto, p.nome\_produto, p.estoque\_minimo, e.estoque\_atual

FROM produtos AS p

LEFT JOIN estoque AS e ON p.id\_produto = e.id\_produto`;

if (filtroEstoqueMinimo) {

query += ' WHERE e.estoque\_atual <= p.estoque\_minimo';

}

if (filtroNomeProduto) {

query += ` AND p.nome\_produto LIKE '%${filtroNomeProduto}%'`;

}

query += ' ORDER BY p.nome\_produto';

transaction.executeSql(

query,

[],

(\_, { rows }) => {

const produtosList = rows.\_array;

setProdutos(produtosList);

},

(\_, error) => {

console.log('Erro ao listar produtos: ' + error);

}

);

});

};

const handleSelecionarProduto = (produto) => {

if (produto.id\_produto === itemSelecionado) {

setSelectedItem(null);

setProdutoSelecionado(null);

} else {

setSelectedItem(produto.id\_produto);

setProdutoSelecionado(produto);

}

};

const handleAtualizarEstoque = () => {

if (produtoSelecionado && itemSelecionado) {

navigation.navigate('AtualizarEstoque', { produto: produtoSelecionado });

setSelectedItem(null);

}

};

const handleEditarProduto = () => {

if (produtoSelecionado && itemSelecionado) {

navigation.navigate('EditarProduto', { produto: produtoSelecionado });

setSelectedItem(null);

}

};

const handleExcluirProduto = () => {

if (produtoSelecionado && itemSelecionado) {

navigation.navigate('ExcluirProduto', { produto: produtoSelecionado });

setSelectedItem(null);

}

};

const handleEstoqueMinimo = () => {

setFiltroEstoqueMinimo(!filtroEstoqueMinimo);

};

const handleLimparFiltro = () => {

setFiltroNomeProduto('');

setFiltroEstoqueMinimo(false);

};

const renderItem = ({ item }) => (

<TouchableOpacity

onPress={() => handleSelecionarProduto(item)}

style={[styles.item, { backgroundColor: item.id\_produto === itemSelecionado ? '#aaa' : '#f9c2ff' }]}

>

<View style={styles.item\_Nome}>

<Text style={styles.nomeProduto1}>Nome:</Text>

<Text style={styles.nomeProduto2}> {item.nome\_produto}</Text>

</View>

<View style={styles.item\_ID}>

<Text style={styles.ID1}>ID:</Text>

<Text style={styles.ID2}> {item.id\_produto}</Text>

</View>

<View style={styles.item\_estAtual}>

<Text style={styles.estoqueText2}>Estoque Atual:</Text>

<Text style={styles.estoqueText3}> {item.estoque\_atual}</Text>

</View>

<View style={styles.item\_estMinimo}>

<Text style={styles.estoqueText2}>Estoque Mínimo:</Text>

<Text style={styles.estoqueText3}> {item.estoque\_minimo}</Text>

</View>

</TouchableOpacity>

);

return (

<View style={styles.container}>

<View style={styles.opcoesContainer1}>

<TouchableOpacity

style={[styles.opcaoButton, filtroEstoqueMinimo ? styles.filtroSelecionado : null]}

onPress={handleEstoqueMinimo}

>

<Text style={[styles.opcaoText, filtroEstoqueMinimo ? styles.opcaoSelecionadaText : null]}>

Estoque Mínimo ({totalEstoqueMinimo})

</Text>

</TouchableOpacity>

<TextInput

style={styles.inputText}

placeholder="Digite o nome do produto"

value={filtroNomeProduto}

onChangeText={(text) => setFiltroNomeProduto(text)}

/>

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

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

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

import { db } from '../database/AbreConexao';

export default function ListarProdutos({ navigation }) {

const [produtos, setProdutos] = useState([]);

const [filtroNomeProduto, setFiltroNomeProduto] = useState('');

const [mostrarEstoqueMinimo, setMostrarEstoqueMinimo] = useState(false);

useEffect(() => {

carregarProdutos();

}, [filtroNomeProduto, mostrarEstoqueMinimo]);

useFocusEffect(

React.useCallback(() => {

setMostrarEstoqueMinimo(false);

carregarProdutos();

}, [])

);

const carregarProdutos = () => {

db.transaction((transaction) => {

let query = `SELECT p.id\_produto, p.nome\_produto, p.estoque\_minimo, e.estoque\_atual

FROM produtos AS p

LEFT JOIN estoque AS e ON p.id\_produto = e.id\_produto`;

if (mostrarEstoqueMinimo) {

query += ' WHERE e.estoque\_atual <= p.estoque\_minimo';

}

if (filtroNomeProduto) {

query += ` AND p.nome\_produto LIKE '%${filtroNomeProduto}%'`;

}

query += ' ORDER BY p.nome\_produto';

transaction.executeSql(

query,

[],

(\_, { rows }) => {

const produtosList = rows.\_array;

setProdutos(produtosList);

},

(\_, error) => {

console.log('Erro ao listar produtos: ' + error);

}

);

});

};

const handleEstoqueMinimo = () => {

setMostrarEstoqueMinimo(!mostrarEstoqueMinimo);

};

const handleLimparFiltro = () => {

setFiltroNomeProduto('');

};

const renderItem = ({ item }) => (

<TouchableOpacity

onPress={() => navigation.navigate('DetalhesProduto', { produto: item })}

style={styles.item}

>

<Text style={styles.itemText}>{item.nome\_produto}</Text>

</TouchableOpacity>

);

return (

<View style={styles.container}>

<TouchableOpacity

style={[styles.botaoFiltro, mostrarEstoqueMinimo ? styles.botaoFiltroAtivo : null]}

onPress={handleEstoqueMinimo}

>

<Text style={styles.botaoFiltroText}>Estoque Mínimo</Text>

</TouchableOpacity>

<View style={styles.inputContainer}>

<TextInput

style={styles.inputText}

placeholder="Digite o nome do produto"

value={filtroNomeProduto}

onChangeText={(text) => setFiltroNomeProduto(text)}

/>

<TouchableOpacity onPress={handleLimparFiltro}>

<Text style={styles.limparText}>Limpar</Text>

</TouchableOpacity>

</View>

<FlatList

data={produtos}

renderItem={renderItem}

keyExtractor={(item) => `produto\_${item.id\_produto.toString()}`}

/>

</View>

);

}

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#fff',

padding: 10,

},

botaoFiltro: {

backgroundColor: '#fff',

borderColor: '#3498db',

borderWidth: 1,

borderRadius: 5,

padding: 10,

marginBottom: 10,

},

botaoFiltroAtivo: {

backgroundColor: '#3498db',

},

botaoFiltroText: {

color: '#3498db',

fontSize: 16,

fontWeight: 'bold',

textAlign: 'center',

},

inputContainer: {

flexDirection: 'row',

alignItems: 'center',

marginBottom: 10,

},

inputText: {

flex: 1,

height: 40,

borderWidth: 1,

borderColor: '#ccc',

borderRadius: 5,

paddingHorizontal: 10,

marginRight: 10,

},

limparText: {

color: 'red',

fontSize: 16,

fontWeight: 'bold',

},

item: {

backgroundColor: '#f9c2ff',

padding: 10,

marginVertical: 5,

borderRadius: 5,

},

itemText: {

fontSize: 16,

},

});

const styles = StyleSheet.create({

  container: {

    flex: 1,

    backgroundColor: '#fff',

    paddingHorizontal: 10,

    paddingTop: 10,

  },

  opcoesContainer: {

    flexDirection: 'row',

    justifyContent: 'space-between',

    alignItems: 'center',

    marginBottom: 10,

  },

  opcaoButton: {

    backgroundColor: '#3498db',

    borderRadius: 5,

    paddingVertical: 10,

    paddingHorizontal: 20,

  },

  opcaoText: {

    color: '#fff',

    fontWeight: 'bold',

  },

  opcaoSelecionadaText: {

    color: '#000',

  },

  inputText: {

    flex: 1,

    borderWidth: 1,

    borderColor: '#ccc',

    borderRadius: 5,

    paddingHorizontal: 10,

    marginRight: 10,

  },

  filtrarButton: {

    backgroundColor: '#fff',

    borderWidth: 1,

    borderColor: '#3498db',

    borderRadius: 5,

    justifyContent: 'center',

    alignItems: 'center',

    paddingVertical: 10,

    paddingHorizontal: 20,

  },

  filtrarButtonText: {

    color: '#3498db',

    fontWeight: 'bold',

  },

  filtroSelecionado: {

    backgroundColor: '#3498db',

  },

  filtroSelecionadoText: {

    color: '#fff',

  },

  limparButton: {

    backgroundColor: '#3498db',

    borderRadius: 5,

    justifyContent: 'center',

    alignItems: 'center',

    paddingVertical: 10,

    paddingHorizontal: 20,

  },

  limparButtonText: {

    color: '#fff',

    fontWeight: 'bold',

  },

  item: {

    backgroundColor: '#f9c2ff',

    padding: 20,

    marginVertical: 8,

    marginHorizontal: 16,

    borderRadius: 5,

  },

  nomeProduto: {

    fontWeight: 'bold',

  },

});