REACT NATIVE-SQLITE- SELECT

A biblioteca expo-sqlite permite o acesso ao banco de dados através da API WebSQL-like. O banco de dados SQLite persiste os dados mesmo depois do celular ser reiniciado.

Vamos continuar o tutorial anterior de Inserção para consultar os dados cadastrados no banco de dados 'nomes' criado para guardar nomes completos.

Nesta aula vamos consultar os dados de forma total, sem critérios, depois caminharemos para o refinamento da pesquisa utilizando a cláusula WHERE do SELECT.

Documentação oficial:

https://docs.expo.dev/versions/latest/sdk/sqlite/

Este tutorial é continuação do React Native – Hook Form SQLite - Inserção.

Instale o SQLite da Expo com a execução da linha abaixo:

```
expo install expo-sqlite
```

Vamos criar uma nova janela (componente) para realizar a consulta, então é necessário algum mecanismo de navegação para trocar de janela e solicitar o carregamento desta que iremos criar.

Estude a aula **React Native - Navegação Stack - Troca de Janela 1** já disponibilizada para entender o funcionamento do **React Native Navigation**.

Este é o esqueleto da janela **ListarNomes.js** que deve ser criada:

```
import React, { useEffect, useState } from "react";
import {
  SafeAreaView,
  View,
  VirtualizedList,
  StyleSheet,
  Text,
  StatusBar,
} from "react-native";

const ListarNomes = () => {

};

const styles = StyleSheet.create({
```

```
  container: {
    flex: 1,
    marginTop: StatusBar.currentHeight,
  },
  item: {
    backgroundColor: "#f9c2ff",
    height: 150,
    justifyContent: "center",
    marginVertical: 8,
    marginHorizontal: 16,
    padding: 20,
  },
  nome: {
    fontSize: 32,
    color: "black"
  },
});

export default ListarNomes;
```

Vamos inserir a importação do SQLite:

```
import * as SQLite from "expo-sqlite";
```

Dentro do componente ListarNomes vamos criar o objeto db para acessar o banco de dados criado na janela de cadastro:

```
  // Abrir ou criar banco de dados SQLite
  const db = SQLite.openDatabase("dados.db");
```

Criar um estado para receber a lista de informações da tabela ‘nomes’:

```
  const [DATA, setDATA] = useState([]);
```

Novamente, para garantir que a tabela exista, caso não tenha sido criada, vamos utilizar o useEffect para preparar a janela para acessar a tabela ‘nomes’. Se a tabela estiver preenchida com dados então o objeto DATA receberá as informações através de um objeto que será serializado para JSON.

```
  // Criação da tabela de nomes, se ela ainda não foi
criada
  useEffect(() => {
    db.transaction((tx) => {
      tx.executeSql(
        "create table if not exists nomes (id integer
primary key not null, nome text, sobrenome text);"
      );
      tx.executeSql("select * from nomes", [], (_, {
rows }) => {
        console.log(JSON.stringify(rows));
        setDATA(rows);
        console.log(JSON.stringify(DATA._array[0].nome
));
      });
    });
  }, []);
```

Os próximos 3 métodos (componentes) são importantes, pois serão carregados na lista de itens (VirtualizedList).

O getItem realiza o carregamento de cada item da lista de nomes:

```
  const getItem = (data, index) => ({
      id: JSON.stringify(DATA._array[index].id),
        nome:
JSON.stringify(DATA._array[index].nome).replace("\"","
")
        .replace("\"",""),
        sobrenome:
JSON.stringify(DATA._array[index].sobrenome
          .replace("\"","")
        .replace("\"",""))});
```

o getItemCount define o tamanho da lista, isto é, a quantidade de registros existentes na tabela 'nomes':

```
  const getItemCount = (data) => data.length;
```

O componente Item é visual e define a forma como as informações serão carregadas:

```
const Item = ({ nome, sobrenome }) => (
  <View style={styles.item}>
    <Text style={styles.nome}>{nome}</Text>
    <Text style={styles.nome}>{sobrenome}</Text>
  </View>
);
```

Vamos criar um método atualizarDados() para recarregar a lista com novos itens:

```
const atualizarDados = () => {
  db.transaction((tx) => {
    tx.executeSql("select * from nomes", [], (_, { rows }) => {
      console.log(JSON.stringify(rows));
      setDATA(rows);
      //
console.log(JSON.stringify(DATA._array[0].nome));
    });
  });
}
```

Abaixo temos o return do componente ListarNomes com dois componentes:

- SafeAreaView – é uma evolução do componente View e procura respeitar as áreas limites do aparelho celular (smartphone) iOS a partir da versão 11. Ver documentação: https://docs.expo.dev/versions/latest/react-native/safeareaview/
- VirtualizedList – é uma das opções de listagem de itens de uma lista do tipo JSON além dos componentes FlatList e SectionList. Este componente gerencia melhor a memória e é a melhor opção para lista com uma massa de dados expressiva.
    - data: define a lista de dados (JSON) que será carregada.
    - getItem: obtém cada item da lista de dados, permitindo o tratamento de cada informação que será apresentada.
    - getItemCount: define a quantidade de itens que será carregada pela lista.

- renderItem: Obtém o item da prop data e carrega na lista.
- keyExtractor: Esta prop define a ordem de apresentação dos itens na lista. O melhor candidato é o item.id para fazer o papel do item.key e auxiliar na ordenação da lista.
- initialNumToRender: Define a quantidade inicial de itens que serão carregados na lista.
- refreshing: Define se o componente será atualizado com novos dados..
- scrollEnabled: Define se o efeito de rolagem pode ser acionado por toque.
- onScroll: executa um método durante o efeito de rolagem.
- Mais informações na documentação oficial: https://docs.expo.dev/versions/latest/react-native/virtualizedlist/

```
  return (
    <SafeAreaView style={styles.container}>
      <VirtualizedList
        data={DATA}
        initialNumToRender={4}
        renderItem={({ item }) => <Item
nome={item.nome}
                                  sobrenome={item.sobrenome}
/>}
        keyExtractor={(item) => item.id}
        getItemCount={getItemCount}
        getItem={getItem}
        refreshing={true}
        scrollEnabled={true}
        onScroll={atualizarDados}

      />
    </SafeAreaView>
  );
};
```

Segue o código completo:

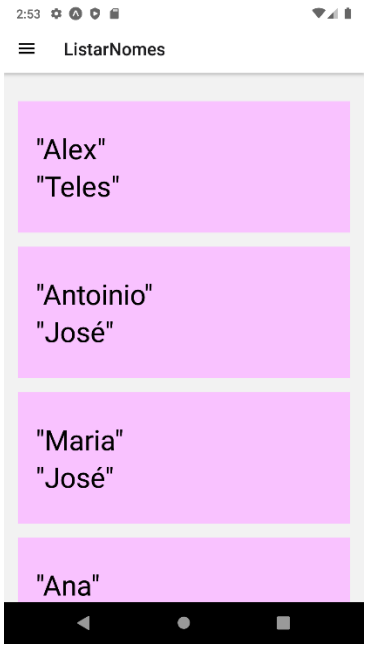
```
import React, { useEffect, useState } from "react";
import {
  SafeAreaView,
  View,
  VirtualizedList,
  StyleSheet,
  Text,
  StatusBar,
} from "react-native";
import * as SQLite from "expo-sqlite";

const ListarNomes = () => {
  // Abrir ou criar banco de dados SQLite
  const db = SQLite.openDatabase("dados.db");

  const [DATA, setDATA] = useState([]);

  // Criação da tabela de nomes, se ela ainda não foi
criada
  useEffect(() => {
    db.transaction((tx) => {
      tx.executeSql(
        "create table if not exists nomes (id integer
primary key not null, nome text, sobrenome text);"
      );
      tx.executeSql("select * from nomes", [], (_, {
rows }) => {
        console.log(JSON.stringify(rows));
        setDATA(rows);
        //
console.log(JSON.stringify(DATA._array[0].nome));
      });
    });
```

```
  }, []);

  const getItem = (data, index) => ({
      id: JSON.stringify(DATA._array[index].id),
        nome: JSON.stringify(DATA._array[index].nome),
        sobrenome:
JSON.stringify(DATA._array[index].sobrenome)});

  const getItemCount = (data) => data.length;

  const Item = ({ nome, sobrenome }) => (
    <View style={styles.item}>
      <Text style={styles.nome}>{nome}</Text>
      <Text style={styles.nome}>{sobrenome}</Text>
    </View>
  );

  const atualizarDados = () => {
    db.transaction((tx) => {
      tx.executeSql("select * from nomes", [], (_, {
rows }) => {
        console.log(JSON.stringify(rows));
        setDATA(rows);
        //
console.log(JSON.stringify(DATA._array[0].nome));
      });
    });
  }

  return (
    <SafeAreaView style={styles.container}>
      <VirtualizedList
        data={DATA}
        initialNumToRender={4}
```

```
        renderItem={({ item }) => <Item
nome={item.nome}
                                  sobrenome={item.sobrenome}
/>}
        keyExtractor={(item) => item.id}
        getItemCount={getItemCount}
        getItem={getItem}
        refreshing={true}
        scrollEnabled={true}
        onScroll={atualizarDados}
      />
    </SafeAreaView>
  );
};



const styles = StyleSheet.create({
  container: {
    flex: 1,
    marginTop: StatusBar.currentHeight,
  },
  item: {
    backgroundColor: "#f9c2ff",
    height: 150,
    justifyContent: "center",
    marginVertical: 8,
    marginHorizontal: 16,
    padding: 20,
  },
  nome: {
    fontSize: 32,
    color: "black"
  },
```

```
});

export default ListarNomes;
```

Execute o projeto, e verifique a lista de nomes cadastradas no banco de dados interno SQLite:



Referência Bibliográfica

[1] https://docs.expo.dev/versions/latest/sdk/sqlite/. Acessado em 24/11/2021.

[2] https://www.w3.org/TR/webdatabase/. Acessado em 24/11/2021.

[3] https://github.com/expo/examples/blob/master/with-sqlite/App.js. Acessado em 24/11/2021.