Instituto Tecnológico de Colima.

“U2”

Asignatura:

DESARROLLO DE APPS MULTIPLATAFORMA

Docente:

ALBERTO VELASCO VERJÁN

Alumno:

Fernando Cárdenas Larios - 16460108
Michell Iván Cázares Martínez – 16460120
Daniel Guadalupe Yocupicio Vazquez - 16460615

“Implementación de base de datos. ”

Introducción

Poder desarrollar para todos los dispositivos de los usuarios con una sola base de código es la tendencia. Permitir agregar una actualización rápida, verdaderas capacidades nativas y creatividad. Expo maneja las partes más importantes de la construcción para cada tienda de aplicaciones, por lo que no necesita Xcode o Android Studio para poner su aplicación en manos de la gente.

Con la Expo, los desarrolladores pueden crear aplicaciones nativas reaccionan sin todas las frustraciones que vienen con la instalación y configuración de las dependencias de software o todas las herramientas necesarias para desarrollar y ejecutar una aplicación nativa de reaccionar.

Expo es un marco y una plataforma para aplicaciones universales de React. Es un conjunto de herramientas y servicios construidos alrededor de React Native y plataformas nativas que lo ayudan a desarrollar, construir, implementar e iterar rápidamente en iOS, Android y aplicaciones web desde la misma base de código JavaScript / TypeScript.

En este documento se podrá apreciar la implementación de un plugin sqlite dentro del framework Expo, para permitirnos implementar una base de datos en nuestra aplicación multiplataforma.

Objetivo

Desarrollar una aplicación multiplataforma que permite conectarse con una base de datos SQL y de esta manera almacenar datos para después ser mostrados en pantalla.

Herramientas utilizadas

1. Expo CLI
2. Expo
3. node.js.
4. react-native.
5. expo-sqlite.
6. expo-constants.

Código de la aplicación

```
import React, { useState } from 'react';
import { ScrollView, StyleSheet, Text, TextInput, TouchableOpacity, View, Button } from 'react-native';
import Constants from 'expo-constants';
import * as SQLite from 'expo-sqlite';

//se crea el objeto base de datos
const db = SQLite.openDatabase("db.db");

function Items({ done: doneHeading, onPressItem }) {
  const [items, setItems] = React.useState(null);

  React.useEffect(() => {
    db.transaction(tx => {
      tx.executeSql(
        `select * from items where done = ?;`,
        [doneHeading ? 1 : 0],
        (_, { rows: { _array } }) => setItems(_array)
      );
    });
  }, []);

  const heading = doneHeading ? "Completed" : "Todo";
```

```
    if (items === null || items.length === 0) {
      return null;
    }

    return (
      <View style={styles.sectionContainer}>
        <Text style={styles.sectionHeading}>{heading}</Text>
        {items.map(({ id, done, value }) => (
          <TouchableOpacity
            key={id}
            onPress={() => onPressItem && onPressItem(id)}
            style={{
              backgroundColor: done ? "#1c9963" : "#fff",
              borderColor: "#000",
              borderWidth: 1,
              padding: 8
            }}
          >
            <Text style={{ color: done ? "#fff" : "#000" }}>{value}</Text>
          </TouchableOpacity>
        ))}
      </View>
    );
  }

  export default function App() {
    const [text, setText] = React.useState(null)
    const [forceUpdate, forceUpdateId] = useForceUpdate()

    React.useEffect(() => {
      db.transaction(tx => {
        tx.executeSql(
          "create table if not exists items (id integer primary key not null,
done int, value text);"
        );
      });
    }, []);

    const add = (text) => {
      // is text empty?
      if (text === null || text === "") {
        return false;
      }

      db.transaction(
        tx => {
          tx.executeSql("insert into items (done, value) values (0, ?)",
[text]);
```

```
          tx.executeSql("select * from items", [], (_, { rows }) =>
            console.log(JSON.stringify(rows))
          );
        },
        null,
        forceUpdate
      );
    }

    const eliminar = () => {
      db.transaction(
        tx => {
          tx.executeSql("delete from items");
        },
        null,
        forceUpdate
      );
    }

    return (
      <View style={styles.container}>
        <Text style={styles.heading}>Práctica SQLite React Native</Text>
        <View style={styles.flexRow}>
          <TextInput
            onChangeText={text => setText(text)}
            onSubmitEditing={() => {
              add(text);
              setText(null);
            }}
            placeholder="Agregue una cadena a la base de datos"
            style={styles.input}
            value={text}
          />
        </View>
        <ScrollView style={styles.listArea}>
          <Items
            key={`forceupdate-todo-${forceUpdateId}`}
            done={false}
            onPressItem={id =>
              db.transaction(
                tx => {
                  tx.executeSql(`update items set done = 1 where id = ?;`, [
                    id
                  ]);
                },
                null,
                forceUpdate
              )
            }
```

```
          />
          <Items
            done
            key={`forceupdate-done-${forceUpdateId}`}
            onPressItem={id =>
              db.transaction(
                tx => {
                  tx.executeSql(`delete from items where id = ?;`, [id]);
                },
                null,
                forceUpdate
              )
            }
          />
        </ScrollView>
        <View>
            <Button
              title="Eliminar todo"
              style={styles.btn}
              onPress={eliminar}
            />
        </View>
      </View>
    );

  }

  function useForceUpdate() {
    const [value, setValue] = useState(0);
    return [() => setValue(value + 1), value];
  }

  const styles = StyleSheet.create({
    container: {
      backgroundColor: "#fff",
      flex: 1,
      paddingTop: Constants.statusBarHeight
    },
    heading: {
      fontSize: 20,
      fontWeight: "bold",
      textAlign: "center"
    },
    flexRow: {
      flexDirection: "row"
    },
    input: {
      borderColor: "#4630eb",
      borderRadius: 4,
```

```
      borderWidth: 1,
      flex: 1,
      height: 48,
      margin: 16,
      padding: 8
    },
    listArea: {
      backgroundColor: "#f0f0f0",
      flex: 1,
      paddingTop: 16
    },
    sectionContainer: {
      marginBottom: 16,
      marginHorizontal: 16
    },
    sectionHeading: {
      fontSize: 18,
      marginBottom: 8
    },
    btn: {
      color: "red"
    }
  });
```

Desarrollando la aplicación

1.- Se tiene que instalar Expo CLI en la computadora y Expo en el dispositivo móvil. En la computadora se hace ejecutando el comando `npm install expo-cli --global.`

2.- Ejecutamos el comando *expo init bd-ejemplo.*

3.- Vamos a la carpeta del proyecto y ejecutamos el comando *expo start*.

4.- Importamos librerías sqlite, constants y el hook useState en la parte superior del código.

5.- Creamos un objeto que contenga la configuración de la base de datos.

```
const db = SQLite.openDatabase("db.db");
```

6.- Generamos la vista de la aplicación mediante las etiquetas <view> teniendo en cuenta la constante de estilos de la parte inferior.

7-. Creamos las constantes de useState y mediante useEffect() ejecutamos el script para crear la tabla de los datos.

8-. Codificamos los métodos de add y eliminar, en este caso el método eliminar quita todos los datos de la tabla mediante un botón inferior color azul.

9-. Para verificar que la aplicación funcione correctamente vamos a una aplicación previamente instalada en el celular llamada Expo y escaneamos el código Qr y esperamos a que cargue.

10-. Para la compilación de la apk para los dispositivos Android ejecutamos un código expo build:android.

Evidencias

Las siguientes capturas muestran la aplicación corriendo en un dispositivo móvil Android.