**SQLite in Flutter**

If you are seeking a way to use mobile device storage as a database in your Flutter application, then you can obtain it by just integrating the [sqflite](https://pub.dev/packages/sqflite) package.In this article, we are going to learn about CRUD operations with SQLite and Flutter. Our final application works as follows:

In this project, we are going to save the title and description in our database.

**Step 1. Add the dependencies**

You need to add the sqflite package in the pubsec.yaml file of your project.

* **sqflite**: The SQLite package to integrate SQLite database functions.

dependencies:  
 ...  
 sqflite: ^2.0.2+1

**Step 2. Create a database**

Create a file named ***database\_helper.dart*** Inside, we then create the database and tables.

Import library:

import 'package:flutter/foundation.dart';  
import 'package:sqflite/sqflite.dart'as sql;

**Add create table function**

**A screenshot of a computer code

Description automatically generated**

**Step 3. Open the database**

**A computer screen shot of a code

Description automatically generated**

**Step 4. Perform the CRUD operations**

1. **Adding/Create Item**

Next, we write the method to insert our items into the database. This method takes two parameters one is the title and the other is the description.

A screen shot of a computer code

Description automatically generated

**2. Read Item**

A computer code with text

Description automatically generated

**3. Edit/Update Item**

A screen shot of a computer code

Description automatically generated

**4. Delete item**

A computer code with text

Description automatically generated

The **database\_helper.dart** file looks as follows:

A screenshot of a computer

Description automatically generated

**Step 5. Creating the form and displaying**

Create a file named **homepage.dart** where we create the form and display our data in lists.

import 'package:flutter/material.dart';  
import 'database\_helper.dart';  
  
class HomePage extends StatefulWidget {  
 const HomePage({Key? key}) : super(key: key);  
  
 @override  
 \_HomePageState createState() => \_HomePageState();  
}  
  
class \_HomePageState extends State<HomePage> {  
 *// All data* List<Map<String, dynamic>> myData = [];  
  
 bool \_isLoading = true;  
 *// This function is used to fetch all data from the database* void \_refreshData() async {  
 final data = await DatabaseHelper.*getItems*();  
 setState(() {  
 myData = data;  
 \_isLoading = false;  
 });  
 }  
  
 @override  
 void initState() {  
 super.initState();  
 \_refreshData(); *// Loading the data when the app starts* }  
  
 final TextEditingController \_titleController = TextEditingController();  
 final TextEditingController \_descriptionController = TextEditingController();  
  
 *// This function will be triggered when the floating button is pressed  
 // It will also be triggered when you want to update an item* void showMyForm(int? id) async {  
 if (id != null) {  
 *// id == null -> create new item  
 // id != null -> update an existing item* final existingData =  
 myData.firstWhere((element) => element['id'] == id);  
 \_titleController.text = existingData['title'];  
 \_descriptionController.text = existingData['description'];  
 }  
  
 showModalBottomSheet(  
 context: context,  
 elevation: 5,  
 isScrollControlled: true,  
 builder: (\_) => Container(  
 padding: EdgeInsets.only(  
 top: 15,  
 left: 15,  
 right: 15,  
 *// prevent the soft keyboard from covering the text fields* bottom: MediaQuery.*of*(context).viewInsets.bottom + 120,  
 ),  
 child: Column(  
 mainAxisSize: MainAxisSize.min,  
 crossAxisAlignment: CrossAxisAlignment.end,  
 children: [  
 TextField(  
 controller: \_titleController,  
 decoration: const InputDecoration(hintText: 'Title'),  
 ),  
 const SizedBox(  
 height: 10,  
 ),  
 TextField(  
 controller: \_descriptionController,  
 decoration: const InputDecoration(hintText: 'Description'),  
 ),  
 const SizedBox(  
 height: 40,  
 ),  
 ElevatedButton(  
 onPressed: () async {  
 *// Save new data* if (id == null) {  
 await addItem();  
 }  
  
 if (id != null) {  
 await updateItem(id);  
 }  
  
 *// Clear the text fields* \_titleController.text = '';  
 \_descriptionController.text = '';  
  
 *// Close the bottom sheet* Navigator.*of*(context).pop();  
 },  
 child: Text(id == null ? 'Create New' : 'Update'),  
 )  
 ],  
 ),  
 ));  
 }  
  
*// Insert a new data to the database* Future<void> addItem() async {  
 await DatabaseHelper.*createItem*(  
 \_titleController.text, \_descriptionController.text);  
 \_refreshData();  
 }  
  
 *// Update an existing data* Future<void> updateItem(int id) async {  
 await DatabaseHelper.*updateItem*(  
 id, \_titleController.text, \_descriptionController.text);  
 \_refreshData();  
 }  
  
 *// Delete an item* void deleteItem(int id) async {  
 await DatabaseHelper.*deleteItem*(id);  
 ScaffoldMessenger.*of*(context).showSnackBar(const SnackBar(  
 content: Text('Successfully deleted!'),  
 backgroundColor:Colors.*green* ));  
 \_refreshData();  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: const Text('Sqlite CRUD'),  
 ),  
 body: \_isLoading  
 ? const Center(  
 child: CircularProgressIndicator(),  
 )  
 : myData.isEmpty?const Center(child: Text("No Data Available!!!")): ListView.builder(  
 itemCount: myData.length,  
 itemBuilder: (context, index) => Card(  
 color:index%2==0?Colors.*green*: Colors.*green*[200],  
 margin: const EdgeInsets.all(15),  
 child:ListTile(  
 title: Text(myData[index]['title']),  
 subtitle: Text(myData[index]['description']),  
 trailing: SizedBox(  
 width: 100,  
 child: Row(  
 children: [  
 IconButton(  
 icon: const Icon(Icons.*edit*),  
 onPressed: () => showMyForm(myData[index]['id']),  
 ),  
 IconButton(  
 icon: const Icon(Icons.*delete*),  
 onPressed: () =>  
 deleteItem(myData[index]['id']),  
 ),  
 ],  
 ),  
 )),  
 ),  
 ),  
 floatingActionButton: FloatingActionButton(  
 child: const Icon(Icons.*add*),  
 onPressed: () => showMyForm(null),  
 ),  
 );  
 }  
}

main.dart

A screenshot of a computer program

Description automatically generated

Run app:

A screenshot of a cell phone

Description automatically generated