Section : IT Systems Development (DSI)
Module : Framework cross-platform workshop
Teaching unit : Mobile Development and Web
Level : 3rd Year (Applied license: LMD)

Workshop 5: Communicating with a local SQLite database

Objective

Access to a local SQLite database by creating model classes

Technical requirements

In order to start with Flutter, we need a few tools:

- A PC with a recent Windows version, or a Mac with a recent version of the macOS or Linux operating system. we can also use a Chrome OS machine, with a few tweaks.
- An Android/iOS setup.
- The Flutter SDK. It's free, light, and open source.
- Physical device/emulator/simulator
- Android Studio/IntelliJ IDEA or Visual Studio Code

content

We will cover the following recipe:

- Use SQLite in Flutter.
- Create model classes.
- Show data to users of the app.
- Use singletons, and perform Create, Read, Update and Delete (CRUD) actions on a local database.

What to do

By creation an SQLite DB, it will be stored somewhere in the emulator.

1- As sqflite is a package, add the dependency in the pubspec.yaml file.

In order to find the latest version of the dependency, please visit https://pub.dev/packages/sqflite. The dependencies that we are going to use are shown in the following code block:

```
dependencies:
  flutter:
    sdk: flutter
  sqflite: ^2.0.0+4
  path: ^1.8.0
```

```
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.3
```

To avoid nullable error you can use the environment as indicated

```
environment:
  sdk: "…"
```

2- In the lib folder, create a subfolder called util. Here, we'll create a new file: dbuse.dart. This file will contain the methods to create the database, and to retrieve and write data.

3- Import the path and sqflite libraries at the top of the dbuse.dart file, as follows:

```
import 'package:path/path.dart';
import 'package:sqflite/sqflite.dart';
```

4- Create a class dbuse that can be called from other parts of our code: quite predictably, as shown in the following snippet: class dbuse {}

```
class dbuse {
  # This will make it easier to update the database
  final int version = 1;
  Database? db;

  static final dbuse _dbHelper = dbuse._internal();
  dbuse._internal();
  factory dbuse() {
    return _dbHelper;
  }
…
}
```

5- Create a method called openDb that will open the database if it exists, or create it if it doesn't;

```
Future<Database> openDb() async {
  if (db == null) {
    db = await openDatabase(join(await getDatabasesPath(), 'scol.db'),
        onCreate: (database, version) {
      database.execute(
          'CREATE TABLE classes(codClass INTEGER PRIMARY KEY, nomClass
TEXT, nbreEtud INTEGER)');
      database.execute(
          'CREATE TABLE etudiants(id INTEGER PRIMARY KEY, codClass INTEGER,
nom TEXT, prenom TEXT, datNais TEXT, ' +
              'FOREIGN KEY(codClass) REFERENCES classes(codClass))');
    }, version: version);
  }
  return db!;
}
```

Asynchronous operations return Future objects (futures), which is something to be completed at a later time. To suspend execution until a Future completes, we use await in an async function.

6- Create a new method in the dbuse class, called testDb(), that will insert some mock data into our database, and then retrieve the data and print it into the debug console. All database methods are asynchronous, so testDb() returns a Future, and is marked async, as follows:

```
Future testDb() async {
  db = await openDb();
  await db?.execute('INSERT INTO classes VALUES (101, "DSI31", 28)');
  await db?.execute(
      'INSERT INTO etudiants VALUES (101, 10, "Ben Foulen", "Foulen",
"05/10/2023")');
  List lists = await db!.rawQuery('select * from classes');
  List items = await db!.rawQuery('select * from etudiants');
  print(lists[0].toString());
  print(items[0].toString());
}
```

7- Go to the main.dart file and remove the default code created by the framework, leaving only the basic code, as shown in the following block: import 'package:flutter/material.dart';

```
import 'package:useSQLLite/util/dbuse.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    dbuse helper = dbuse();
    helper.testDb();

    return MaterialApp(
        title: 'Class List',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: Container());
  }
}
```

You can see an example of the debug console content:

```
Debug service listening on ws://127.0.0.1:52786/fGsPEpeN5eY=/ws
Syncing files to device Android SDK built for x86...
D/eglCodecCommon(16193): setVertexArrayObject: set vao to 0 (0) 1 0
I/flutter (16193): {codClass: 10, nomClass: DSI31, nbreEtud: 28}
I/flutter (16193): {id: 1, codClass: 10, nom: Ben Foulen, prenom: Foulen, datNais:
10/10/2021}
```

1- - Create a new folder, called models, Create in it a new file called scol_list.dart with the code

```
class ScolList {
  int codClass;
  String nomClass;
  int nbreEtud;
```

```
  ScolList(this.codClass, this.nomClass, this.nbreEtud);

  Map<String, dynamic> toMap() {
    return {
      'codClass': (codClass == 0) ? null : codClass,
      'nomClass': nomClass,
      'nbreEtud': nbreEtud,
    };
  }
}
```

2- Create then in the same folder a new file called list_etudiants.dart

```
class ListEtudiants {
  int id;
  int codClass;
  String nom;
  String prenom;
  String datNais;

  ListEtudiants(this.id, this.codClass, this.nom, this.prenom,
this.datNais);

  Map<String, dynamic> toMap() {
    return {
      'id': (id == 0) ? null : id,
      'codClass': codClass,
      'nom': nom,
      'prenom': prenom,
      'datNais': datNais
    };
  }
}
```

3- - In the dbuse class, add a new method called getClasses() that will return a Future of a Class, containing a ScolList. As usual, this will be asynchronous. The following code illustrates this:

```
Future<List<ScolList>> getClasses() async {
  final List<Map<String, dynamic>> maps = await db!.query('classes');
  return List.generate(maps.length, (i) {
    return ScolList(
      maps[i]['codClass'],
      maps[i]['nomClass'],
      maps[i]['nbreEtud'],
    );
  });
}
```

4- - In the main.dart file, at the top of the _ShListState class, create a ScolList property that will be a List of ScolList items, as follows:

```
Late List<ScolList> scolList;
```

5- - Create a showData() method then in this method call the setState() method to tell our app that the ScolList has changed,

6- - in main.dart we need to show it on the screen

```
import 'package:flutter/material.dart';
import 'package:useSQLLite/models/scol_list.dart';
import 'package:useSQLLite/util/dbuse.dart';
```

```
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        title: 'Classes List',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        home: ShList());
  }
}

class ShList extends StatefulWidget {
  @override
  _ShListState createState() => _ShListState();
}

class _ShListState extends State<ShList> {
  List<ScolList> scolList=[];
  dbuse helper = dbuse();
  @override
  Widget build(BuildContext context) {
    showData();
    return Scaffold(
      appBar: AppBar(
        title: Text(' Classes list'),
      ),
      body: ListView.builder(
          itemCount: (scolList != null) ? scolList.length : 0,
          itemBuilder: (BuildContext context, int index) {
            return ListTile(title: Text(scolList[index].nomClass));
          }),
    );
  }

  Future showData() async {
    …
  }
}
```

Testing the database

7- Test at first the app with the showData method:

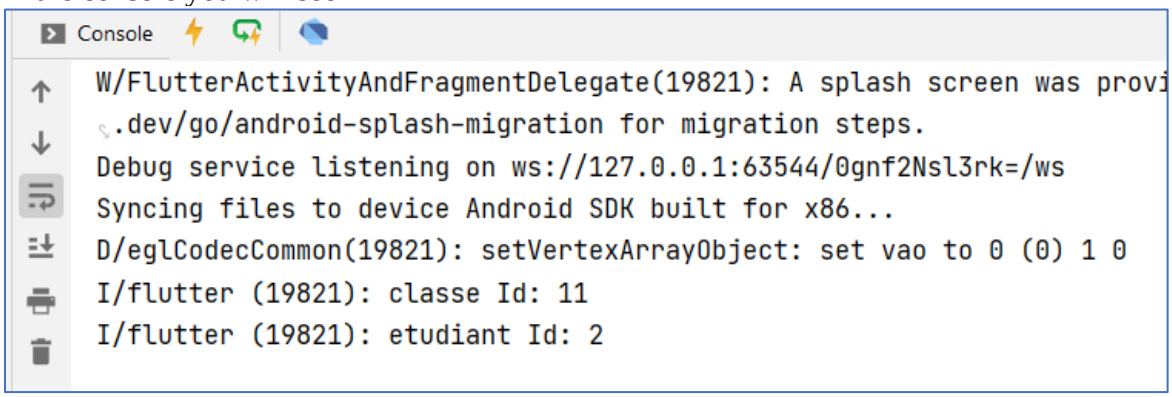
```
Future showData() async {

await helper.openDb();
ScolList list1 = ScolList(11, "DSI31", 30);
int ClassId1 = await helper.insertClass(list1);

ScolList list2 = ScolList(12, "DSI32", 26);
int ClassId2 = await helper.insertClass(list2);

ScolList list3 = ScolList(13, "DSI33", 28);
int ClassId3 = await helper.insertClass(list3);

String dateStart = '22-04-2021';
DateFormat inputFormat = DateFormat('dd-MM-yyyy');
```

```
DateTime input = inputFormat.parse(dateStart);
//String datee = DateFormat('hh:mm a').format(input);
String datee = DateFormat('dd-MM-yyyy').format(input);

final DateTime now = DateTime.now();
final DateFormat formatter = DateFormat('yyyy-MM-dd');
final String formatted = formatter.format(now);

ListEtudiants etud =
    ListEtudiants(1, ClassId1, "Ali", "Ben Mohamed", datee);
int etudId1 = await helper.insertEtudiants(etud);
print('classe Id: ' + ClassId1.toString());
print('etudiant Id: ' + etudId1.toString());

etud = ListEtudiants(2, ClassId2, "Salah", "Ben Salah", datee);
etudId1 = await helper.insertEtudiants(etud);
etud = ListEtudiants(3, ClassId2, "Slim", "Ben Slim", datee);
etudId1 = await helper.insertEtudiants(etud);
etud = ListEtudiants(4, ClassId3, "Foulen", "Ben Foulen", datee);
etudId1 = await helper.insertEtudiants(etud);

}
```

The isertClass and the isertEtudiants are as follow

```
Future<int> insertClass(ScolList list) async {
  int codClass = await this.db.insert(
        'classes',
        list.toMap(),
        conflictAlgorithm: ConflictAlgorithm.replace,
      );
  return codClass;
}

Future<int> insertEtudiants(ListEtudiants etud) async {
  int id = await db.insert(
    'etudiants',
    etud.toMap(),
    conflictAlgorithm: ConflictAlgorithm.replace,
  );
  return id;
}
```

In the console you will see

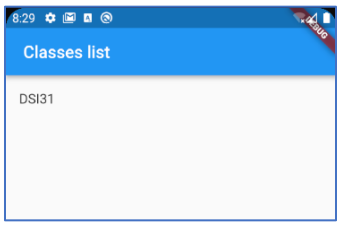


Showing database data to the user

1- And then with code:

```
Future showData() async {
  await helper.openDb();
  scolList = await helper.getClasses();
  setState(() {
    scolList = scolList;
  });
}
```

On the emulator you will see



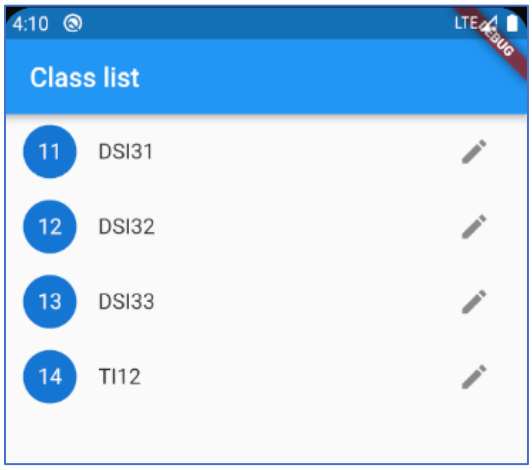
In order to complete this screen, we'll add a few more data for our items, as follows:

2- First, to make the user interface (UI) a bit more appealing, add a CircleAvatar to the leading property the ListTile, containing the priority, as follows:

```
return ListTile(

    title: Text(scolList[index].nomClass),
    leading: CircleAvatar(
      child: Text(scolList[index].codClass.toString()),
    ),
```

3- Then, still in the ListTile, add a trailing icon that we'll use later on to edit the scolList, like this:

```
return Scaffold(
  appBar: AppBar(
    title: Text(' Classes list'),
  ),
  body: ListView.builder(
      itemCount: (scolList != null) ? scolList.length : 0,
      itemBuilder: (BuildContext context, int index) {
        return ListTile(
            title: Text(scolList[index].nomClass),
            leading: CircleAvatar(
              child: Text(scolList[index].codClass.toString()),
            ),
            trailing: IconButton(
              icon: Icon(Icons.edit),
              onPressed: () {},
            ));
      }),
);
```



Now that we can see the Classes List, let's also show the students (etudiants) on each list. For this, we'll need a new file, as follows

4- In order to organize better our code, create a new folder called ui that will contain the UI files of our projects, except main.dart, which will remain in the lib folder.

5- In the ui folder, create a new file called students_screen.dart. Inside the new file, first, import the files that we'll need in order to show the students, as follows:

```
import 'package:flutter/material.dart';
import '../models/list_etudiants.dart';
import '../models/scol_list.dart';
import '../util/dbuse.dart';

class StudentsScreen extends StatefulWidget {
  final ScolList scolList;
  StudentsScreen(this.scolList);
  @override
  _StudentsScreenState createState() =>
_StudentsScreenState(this.scolList);
}

class _StudentsScreenState extends State<StudentsScreen> {
  final ScolList scolList;
  _StudentsScreenState(this.scolList);
  dbuse helper;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: Text(scolList.nomClass),
        ),
        body:Container()
    );
  }
…
```

6- To test the app, call StudentsScreen when the user taps on one of the classes in the ListView of the main screen.

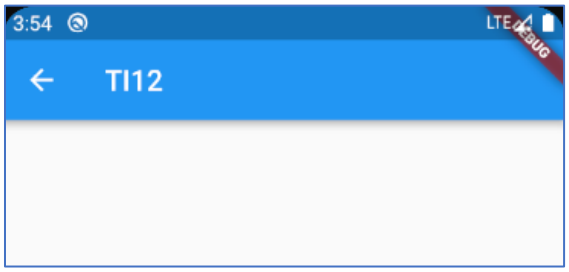
Back to the main.dart file, first, we'll import the students_screen.dart file, like this:

```
import 'package:useSQLLite/ui/students_screen.dart';
```

7- In the build() method of the _ShLstState class, inside the ListTile of the ListView, add an onTap parameter. Inside it, call the Navigator.push() method to call the StudentsScreen, passing the object in the scolList at position index, as follows:

```
return ListTile(
    onTap: () {
      Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => StudentsScreen(scolList[index])),
      );
    },
    title: Text(scolList[index].nomClass),
```

8- If you try this out, when you tap on one of the items in the ListView, you'll get to the second screen, which right now only shows the title of the scolList, as shown in the following screenshot:



9- Write the getEtudiants method in dbuse.dart as follow

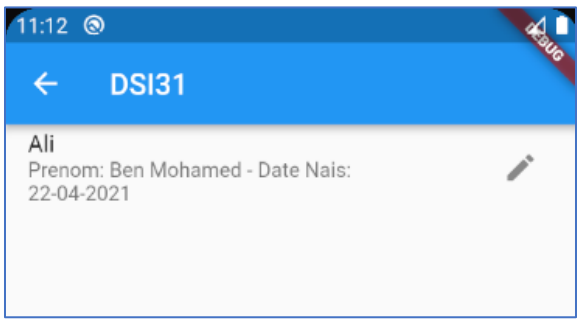
```
Future<List<ListEtudiants>> getEtudiants(code) async {
  final List<Map<String, dynamic>> maps =
      await db.query('etudiants', where: 'codClass = ?', whereArgs:
[code]);
  return List.generate(maps.length, (i) {
    return ListEtudiants(
      maps[i]['id'],
      maps[i]['codClass'],
      maps[i]['nom'],
      maps[i]['prenom'],
      maps[i]['datNais'],
    );
  });
}
```

10- In students_screen.dart we have

```
import 'package:flutter/material.dart';

import '../models/list_etudiants.dart';
import '../models/scol_list.dart';
import '../util/dbuse.dart';

class StudentsScreen extends StatefulWidget {
  final ScolList scolList;
  StudentsScreen(this.scolList);
```

```
  @override
  _StudentsScreenState createState() =>
_StudentsScreenState(this.scolList);
}

class _StudentsScreenState extends State<StudentsScreen> {
  final ScolList scolList;
  _StudentsScreenState(this.scolList);
  dbuse helper;
  List<ListEtudiants> students;

  @override
  Widget build(BuildContext context) {
    helper = dbuse();

    showData(this.scolList.codClass);
    return Scaffold(
        appBar: AppBar(
          title: Text(scolList.nomClass),
        ),
        body: ListView.builder(
            itemCount: (students != null) ? students.length : 0,
            itemBuilder: (BuildContext context, int index) {
              return ListTile(
                title: Text(students[index].nom),
                subtitle: Text(
                    'Prenom: ${students[index].prenom} - Date Nais:
${students[index].datNais}'),
                onTap: () {},
                trailing: IconButton(
                  icon: Icon(Icons.edit),
                  onPressed: () {},
                ),
              );
            }));
  }

  Future showData(int idList) async {
    await helper.openDb();
    students = await helper.getEtudiants(idList);
    setState(() {
      students = students;
    });
  }
}
```

11- If you try this out, you should be able to see each class in the scol(class) list, with a trailing edit icon, as shown in the following screenshot



Inserting and editing data

1- Create the class that will contain the UI for the dialog. We'll call it ScolListDialog, as follows:
class ScolListDialog {}

Import the required dependencies in a new dart filecalled scol_list_dialog.dart:

```
import 'package:flutter/material.dart';
import 'package:useSQLLite/util/dbuse.dart';
import '../models/scol_list.dart';

class ScolListDialog {
  final txtNonClass = TextEditingController();
  final txtNbreEtud = TextEditingController();
  Widget buildDialog(BuildContext context, ScolList list, bool isNew) {
    dbuse helper = dbuse();
    if (!isNew) {
      txtNonClass.text = list.nomClass;
      txtNbreEtud.text = list.nbreEtud.toString();
    }
    return AlertDialog(
        title: Text((isNew) ? 'Class list' : 'Edit class list'),
        shape:
            RoundedRectangleBorder(borderRadius:
BorderRadius.circular(30.0)),
        content: SingleChildScrollView(
          child: Column(children: <Widget>[
            TextField(
                controller: txtNonClass,
                decoration: InputDecoration(hintText: 'Class List Name')),
            TextField(
              controller: txtNbreEtud,
              keyboardType: TextInputType.number,
              decoration:
                  InputDecoration(hintText: 'Class List numbre of
student'),
            ),
            RaisedButton(
              child: Text('Save Class List'),
              onPressed: () {
                list.nomClass = txtNonClass.text;
                list.nbreEtud = int.parse(txtNbreEtud.text);
                helper.insertClass(list);
                Navigator.pop(context);
              },
            ),
          ]),
        ));
  }
}
```

2- At the top of the main.dart file, we'll import the scol_list_dialog.dart file, like this:

```
import 'package:useSQLLite/ui/scol_list_dialog.dart';
```

3- In the _ShListState class, let's first declare a ScolListDialog, called dialog, then override the initState() method to create an instance of the class, as follows:

```
ScolListDialog dialog;
@override
void initState() {
  dialog = ScolListDialog();
  super.initState();
}
```

4- For the edit functionality, in the onPressed parameter of the edit button in the ListTile, call the showDialog method. In its builder, passing the context, the current ShoppingList, and false, as this is an edit and not an insert, as follows:

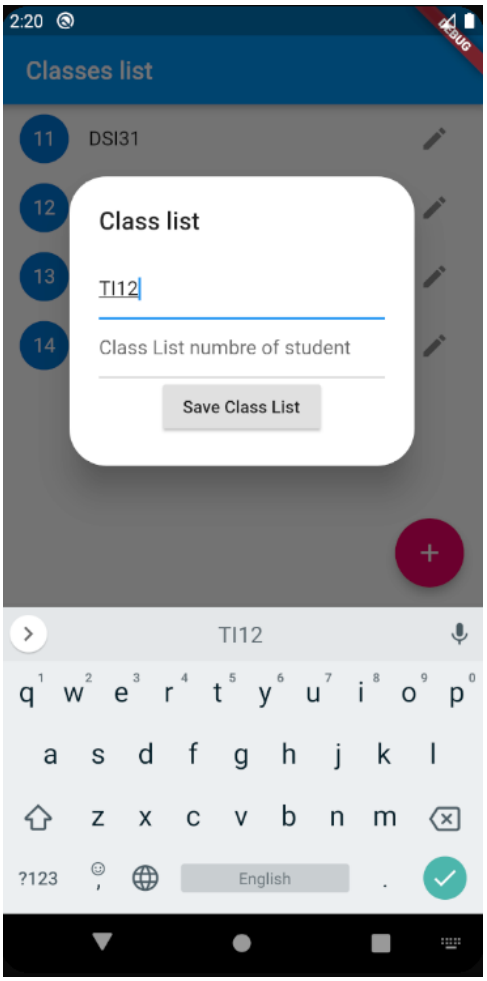
```
onPressed: () {
  showDialog(
      context: context,
      builder: (BuildContext context) =>
          dialog.buildDialog(context, scolList[index], false));
},
```

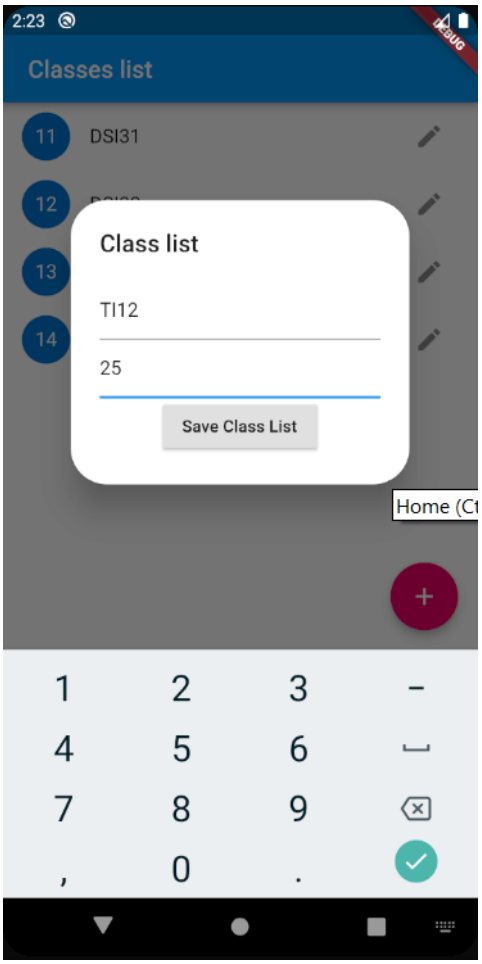
5- Add **`ScolListDialog dialog = ScolListDialog();`** in main.dart

```
@override
Widget build(BuildContext context) {
  ScolListDialog dialog = ScolListDialog();

  showData();
  return Scaffold(
    appBar: AppBar(
      title: Text(' Class list'),
```

6- Under the body of the Scaffold, add a FloatingActionButton

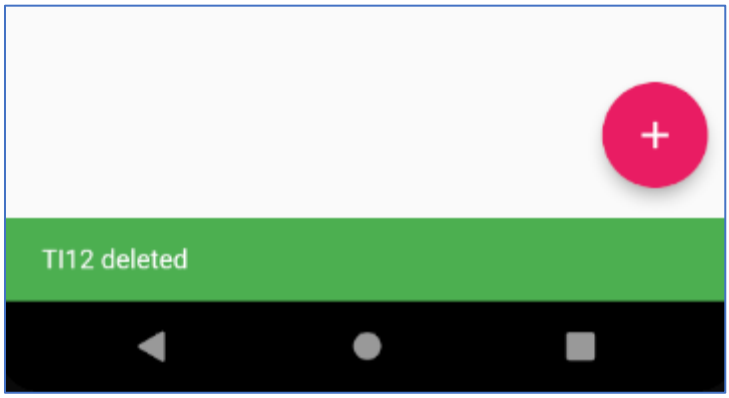
```
floatingActionButton: FloatingActionButton(
  onPressed: () {
    showDialog(
      context: context,
      builder: (BuildContext context) =>
          dialog.buildDialog(context, ScolList(0, '', 0), true),
    );
  },
  child: Icon(Icons.add),
  backgroundColor: Colors.pink,
```





7- In main.dart you need to add the following code for deleting one class

```
body: ListView.builder(
  itemCount: (scolList != null) ? scolList.length : 0,
  itemBuilder: (BuildContext context, int index) {
    return Dismissible(
        key: Key(scolList[index].nomClass),
    onDismissed: (direction) {
    String strName = scolList[index].nomClass;
    helper.deleteList(scolList[index]);
    setState(() {
    scolList.removeAt(index);
    });
    Scaffold.of(context).showSnackBar(
    SnackBar(content: Text("$strName deleted")));
    },
    child:  ListTile(
```



CRUD operations in list_student_dialog.dart Create the ListItemDialog class in the

1- list_student_dialog.dart file,

```
import 'package:flutter/material.dart';
import '../models/list_etudiants.dart';
import '../util/dbuse.dart';

class ListStudentDialog {
  final txtNom = TextEditingController();
  final txtPrenom = TextEditingController();
  final txtdatNais = TextEditingController();
  Widget buildAlert(BuildContext context, ListEtudiants student, bool
isNew) {
    dbuse helper = dbuse();
    helper.openDb();
    if (!isNew) {
      txtNom.text = student.nom;
      txtPrenom.text = student.prenom;
      txtdatNais.text = student.datNais;
    }
    return AlertDialog(
      title: Text((isNew) ? 'New student' : 'Edit student'),
      content: SingleChildScrollView(
        child: Column(
          children: <Widget>[
            TextField(
                controller: txtNom,
                decoration: InputDecoration(hintText: 'Student Name')),
            TextField(
              controller: txtPrenom,
              decoration: InputDecoration(hintText: 'First name'),
            ),
            TextField(
              controller: txtdatNais,
              decoration: InputDecoration(hintText: 'Date naissance'),
            ),
            RaisedButton(
                child: Text('Save dtudent'),
                onPressed: () {
                  student.nom = txtNom.text;
                  student.prenom = txtPrenom.text;
                  student.datNais = txtdatNais.text;
```

```
                    helper.insertEtudiants(student);
                    Navigator.pop(context);
                  },
                  shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(30.0)))
            ],
          ),
        ),
      );
  }
}
```

2- In the build() method in the students_screen.dart file, create an instance of the ListStudentDialog class called dialog, like this:

```
@override
Widget build(BuildContext context) {
  helper = dbuse();
  ListStudentDialog dialog = new ListStudentDialog();
```

3- Update the ListTile in the itemBuilder, in the ListView.builder of the same method, like this:

```
onPressed: () {
  showDialog(
      context: context,
      builder: (BuildContext context) =>
          dialog.buildAlert(context, students[index], false));
},
```

4- In the Scaffold returned by the build() method of the _StudentsScreenState class, add a FloatingActionButton, as follows:

```
floatingActionButton: FloatingActionButton(
  onPressed: () {
    showDialog(
      context: context,
      builder: (BuildContext context) => dialog.buildAlert(
          context, ListEtudiants(0, scolList.codClass, '', '', ''), true),
    );
  },
  child: Icon(Icons.add),
  backgroundColor: Colors.pink,
```

5- In the DbHelper class of the dbhelper.dart file, add a deleteItem method, as follows

```
Future<int> deleteStudent(ListEtudiants student) async {
  int result =
      await db.delete("etudiants", where: "id = ?", whereArgs:
[student.id]);
  return result;
}
```

6- Add a Dismissible widget to the students screen,.

```
import 'package:flutter/material.dart';
import 'package:useSQLLite/ui/list_student_dialog.dart';
```

```
import '../models/list_etudiants.dart';
import '../models/scol_list.dart';
import '../util/dbuse.dart';

class StudentsScreen extends StatefulWidget {
  final ScolList scolList;
  StudentsScreen(this.scolList);
  @override
  _StudentsScreenState createState() =>
_StudentsScreenState(this.scolList);
}

class _StudentsScreenState extends State<StudentsScreen> {
  final ScolList scolList;
  _StudentsScreenState(this.scolList);
  dbuse helper;
  List<ListEtudiants> students;

  @override
  Widget build(BuildContext context) {
    helper = dbuse();
    ListStudentDialog dialog = new ListStudentDialog();
    showData(this.scolList.codClass);
    return Scaffold(
      appBar: AppBar(
        title: Text(scolList.nomClass),
      ),
      body: ListView.builder(
          itemCount: (students != null) ? students.length : 0,
          itemBuilder: (BuildContext context, int index) {
            return Dismissible(
              key: Key(students[index].nom),
              onDismissed: (direction) {
                String strName = students[index].nom;
                helper.deleteStudent(students[index]);
                setState(() {
                  students.removeAt(index);
                });
                Scaffold.of(context)
                    .showSnackBar(SnackBar(content: Text("$strName
deleted")));
              },
              child: ListTile(
                title: Text(students[index].nom),
                subtitle: Text(
                    'Prenom: ${students[index].prenom} - Date Nais:
${students[index].datNais}'),
                onTap: () {},
                trailing: IconButton(
                  icon: Icon(Icons.edit),
                  onPressed: () {
                    showDialog(
                        context: context,
                        builder: (BuildContext context) =>
                            dialog.buildAlert(context, students[index],
false));
                  },
                ),
              ),
            );
          }),
```

```
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          showDialog(
            context: context,
            builder: (BuildContext context) => dialog.buildAlert(
                context, ListEtudiants(0, scolList.codClass, '', '', ''),
true),
          );
        },
        child: Icon(Icons.add),
        backgroundColor: Colors.pink,
      ),
    );
  }

  Future showData(int idList) async {
    await helper.openDb();
    students = await helper.getEtudiants(idList);
    setState(() {
      students = students;
    });
  }
}
```

Note: if a problem occurred with Sqflite and the api version, then delete the folder of the api in sdk\platform. And then rerun the app. Automatically Flutter install the appropriate api.