



## Atelier 8

### Pages Contact & Ajout\_modif\_contact

Matière : ATELIER FRAMEWORK CROSS-PLATFORM

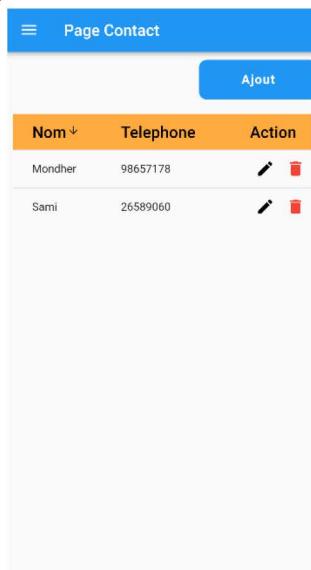
Enseignants : M. Hadjji & S. Hadhri

DSI3

L'objectif de cet atelier est de manipuler une base de données SQLite et d'exécuter des requêtes SQL qui représentent les différentes opérations CRUD. Il s'agit de créer une base de données "voyage" possédant la table suivante :

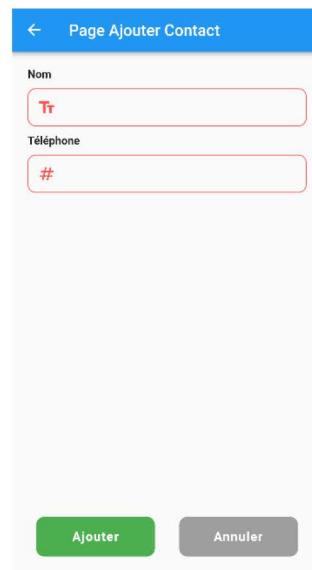
**contact (id, nom, tel)**

Les captures des pages créées sont les suivantes :



Nom	Telephone	Action
Mondher	98657178	 
Sami	26589060	 

Page Contact



← Page Ajouter Contact

Nom  
Tr

Téléphone  
#

Ajouter Annuler

Page ajout\_modif\_contact

1. Ajouter les dépendances de SQFLite (un plugin SQLITE) au fichier « *pubspec.yaml* » :  
<https://pub.dev/packages/sqlite>

#### Fichier pubspec.yaml

```
...  
dependencies:  
  sqflite:  
  path:  
  intl:  
  http:  
  shared_preferences:  
  flutter:  
    sdk: flutter  
...  
...
```

Puis lancer la commande : `flutter pub get`

2. Créer un dossier model sous le dossier lib y ajouter un fichier « *contact.model.dart* » qui représente le model de ce projet et définir une classe « *contact* » contenant :
  - Les attributs id, nom et tel
  - Un constructeur
  - Une méthode *fromJson()* qui convertit un objet Map en un contact
  - Une méthode *toJson()* qui convertit un contact en un objet Map

### Fichier contact.model.dart

```
class Contact {  
    static String table = "contact";  
  
    int? id;  
    String? nom;  
    int? tel;  
  
    Contact({this.id, this.nom, this.tel});  
  
    static Contact fromJson(Map<String, dynamic> json) {  
        return Contact(  
            id: json['id'], nom: json['nom'].toString(), tel: json['tel']);  
    }  
  
    Map<String, dynamic> toJson() {  
        return {  
            'id': id,  
            'nom': nom,  
            'tel': tel,  
        };  
    }  
}
```

3. Créer un dossier « utils » sous « lib » et y ajouter un fichier « contact.database.dart ». Ce fichier contiendra une classe « ContactDatabase » dans laquelle sont implémentées les différentes méthodes pour la création de la base de données et les opérations CRUD.

### Fichier contact.database.dart

```
import 'dart:async';  
import 'package:path/path.dart';  
import 'package:sqflite/sqflite.dart';  
import '../model/contact.model.dart';  
  
class ContactDatabase {  
    static Database? _database;  
  
    initDB() async {  
        if (_database == null) {  
            String databasePath = await getDatabasesPath();  
            String _path = join(databasePath, 'voyage.db');  
            _database = await openDatabase(_path, version: 1, onCreate: onCreate);  
        }  
    }  
  
    onCreate(Database db, int version) async {  
        String sql =  
            'CREATE TABLE contact (id INTEGER PRIMARY KEY AUTOINCREMENT, nom  
STRING, tel STRING)';  
        await db.execute(sql);  
    }  
  
    Future<List<Map<String, dynamic>>> recuperer() async {  
        await initDB();  
        return _database!.query(Contact.table);  
    }  
  
    Future<int> inserer(Contact contact) async {  
        await initDB();  
        return _database!.insert(Contact.table, contact.toJson());  
    }  
}
```

```

Future<int> modifier(Contact contact) async {
    await initDB();
    return _database!.update(Contact.table, contact.toJson(),
        where: 'id = ?', whereArgs: [contact.id]);
}

Future<int> supprimer(Contact contact) async {
    await initDB();
    return _database!
        .delete(Contact.table, where: 'id = ?', whereArgs: [contact.id]);
}
}

```

4. Créer un dossier « services » sous « lib » et y ajouter un fichier « contact.service.dart ». Ce fichier contiendra une classe « ContactService » dans laquelle sont implémentées les différents services pour les différentes opérations CRUD.

#### Fichier contact.service.dart

```

import '../model/contact.model.dart';
import '../utils/contact.database.dart';

class ContactService {
    ContactDatabase contactDatabase = ContactDatabase();

    Future<List<Contact>> listeContacts() async {
        var contacts = await contactDatabase.recuperer();
        return contacts.map((item) => Contact.fromJson(item)).toList();
    }

    Future<bool> ajouterContact(Contact c) async {
        int res = await contactDatabase.inserer(c);
        return res > 0 ? true : false;
    }

    Future<bool> modifierContact(Contact c) async {
        int res = await contactDatabase.modifier(c);
        return res > 0 ? true : false;
    }

    Future<bool> supprimerContact(Contact c) async {
        int res = await contactDatabase.supprimer(c);
        return res > 0 ? true : false;
    }
}

```

5. Créer une nouvelle page « lib/pages/ajout\_modif\_contact.page.dart »

#### Fichier ajout\_modif\_contact.page.dart

```

import 'package:flutter/material.dart';

class AjoutModifContactPage extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text('Page Ajouter/Modifier Contact'),
            ),
            body: Center(
                child: Text(
                    'Page Ajouter/Modifier Contact', style: TextStyle(fontSize: 22),
                )
            )
        );
    }
}

```

```

        ),
    ),
);
}

```

6. A partir de cette question, l'interface graphique sera développée et améliorée à chaque étape.

Ajouter les dépendances du package SnippetCoderUtils au fichier « *pubspec.yaml* » :

[https://pub.dev/packages/snippet\\_coder\\_utils](https://pub.dev/packages/snippet_coder_utils)

**Remarque :**

SnippetCoderUtils est un package utilitaire construit avec Flutter SDK pour rendre le développement Flutter plus facile. Il offre plusieurs classes comme :

- FormHelper pour créer une interface utilisateur personnalisée TextBox, DropDown,...
- DataTable avec Generic Collection, Sorting, Edit, Delete&Custom Action Button

7. Ajouter dans la page Contact un bouton « Ajout » pour ouvrir la page « ajout\_modif\_contact.page »

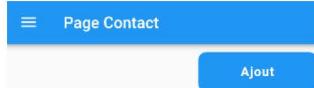
Fichier contact.page.dart

```

...
import 'package:snippet_coder_utils/FormHelper.dart';
import 'ajout_modif_contact.page.dart';

...
body: Center(
    child: Column(
        children: [
            SizedBox(
                height: 10,
            ),
            Align(
                alignment: Alignment.centerRight,
                child: FormHelper.submitButton(
                    "Ajout",
                    () {
                        Navigator.push(
                            context,
                            MaterialPageRoute(
                                builder: (context) => AjoutModifContactPage(),
                            ),
                        );
                    },
                    borderRadius: 10,
                    btnColor: Colors.blue,
                    borderColor: Colors.blue,
                ),
            ),
        ],
    ),
...

```



8. Convertir la page « ajout\_modif\_contact.page » en StatefulWidget et y ajouter un formulaire permettant d'ajouter un contact dans la base de données.

#### Fichier ajout\_modif\_contact.page.dart

```
...
import 'package:snippet_coder_utils/FormHelper.dart';
import '../model/contact.model.dart';
import '../services/contact.service.dart';
...
class _AjoutModifContactPageState extends State<AjoutModifContactPage> {
    GlobalKey<FormState> globalKey = GlobalKey<FormState>();
    Contact contact=Contact();
    ContactService contactService=ContactService();
...
    body: Form(
        key: globalKey,
        child: _formUI(),
    ),
    bottomNavigationBar: SizedBox(
        height: 100,
        child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceEvenly,
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
                FormHelper.submitButton(
                    "Ajouter",
                    () {
                        if (validateAndSave()) {
                            contactService.ajouterContact(contact!).then((value) {
                                Navigator.pop(context);
                            });
                        }
                    },
                    borderRadius: 10,
                    btnColor: Colors.green,
                    borderColor: Colors.green,
                ),
                FormHelper.submitButton(
                    "Annuler",
                    () {
                        Navigator.pop(context);
                    },
                    borderRadius: 10,
```

```

        btnColor: Colors.grey,
        borderColor: Colors.grey,
    ),
),
),
),
);
}

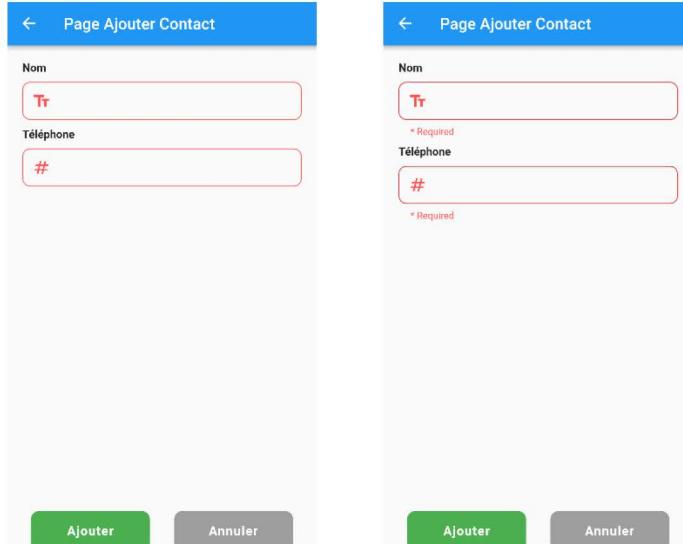
formUI() {
return SingleChildScrollView(
child: Padding(
padding: const EdgeInsets.all(10),
child: Column(
children: [
FormHelper.inputFieldWidgetWithLabel(
context,
"nom",
"Nom",
"",
(onValidate) {
if (onValidate.isEmpty) {
return "* Required";
}
return null;
},
(onSaved) {
contact!.nom = onSaved.toString().trim();
},
initialValue: "",
showPrefixIcon: true,
prefixIcon: const Icon(Icons.text_fields),
borderRadius: 10,
contentPadding: 15,
fontSize: 14,
labelFontSize: 14,
paddingLeft: 0,
paddingRight: 0,
prefixIconPaddingLeft: 10,
),
FormHelper.inputFieldWidgetWithLabel(
context, "telephone", "Téléphone", "", (onValidate) {
if (onValidate.isEmpty) {
return "* Required";
}
return null;
},
(onSaved) {
contact!.tel = int.parse(onSaved.toString().trim());
},
initialValue: "",
showPrefixIcon: true,
prefixIcon: const Icon(Icons.numbers),
borderRadius: 10,
contentPadding: 15,
fontSize: 14,
labelFontSize: 14,
paddingLeft: 0,
paddingRight: 0,
prefixIconPaddingLeft: 10,
isNumeric: true),
),
),
),
);
}

```

```

    }
    bool validateAndSave() {
        final form = globalKey.currentState;
        if (form!.validate()) {
            form.save();
            return true;
        }
        return false;
    }
}

```



9. Convertir la page Contact en StatefulWidget puis y afficher un DataTable contenant la liste des contacts

#### Fichier contacts.page.dart

```

...
import 'package:snippet_coder_utils/list_helper.dart';
import '../model/contact.model.dart';
import '../services/contact.service.dart';
...
class _ContactPageState extends State<ContactPage> {
    ContactService contactService = ContactService();
    @override
    Widget build(BuildContext context) {
...
        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) => AjoutModifContactPage(
                    ),
            ),
        ).then((value) {
            setState(() {});
        });
    },
    borderRadius: 10,
    btnColor: Colors.blue,
    borderColor: Colors.blue,
),
),
SizedBox(
    height: 10,
),

```

```

        _fetchData(),
    ],
),
);
}

fetchData() {
return FutureBuilder<List<Contact>>(
future: contactService.listeContacts(),
builder: (BuildContext context, AsyncSnapshot<List<Contact>> contacts) {
if (contacts.hasData) return _buildDataTable(contacts.data!);
return Center(child: CircularProgressIndicator());
},
);
}

buildDataTable(List<Contact> listContacts) {
return Padding(
padding: const EdgeInsets.all(8.0),
child: ListUtils.buildDataTable(
context,
["Nom", "Telephone", "Action"],
["nom", "tel", ""],
false,
0,
listContacts,
(Contact c) {
// Modifier contact
},
(Contact c) {
// Supprimer contact
},
headingRowColor: Colors.orangeAccent,
isScrollable: true,
columnTextFontSize: 20,
columnTextBold: false,
columnSpacing: 50,
onSort: (columnIndex, columnName, asc) {},
),
);
}
}
}

```

Nom	Telephone	Action
Mondher	98657178	
Sami	26589060	

10. Apporter les modifications nécessaires aux pages « Contact » et « ajout\_modif\_contact » afin de pouvoir modifier un contact

#### Fichier ajout\_modif\_contact.page.dart

```

...
class AjoutModifContactPage extends StatefulWidget {
  AjoutModifContactPage({this.contact, this.modifMode = false});

  final Contact? contact;
  final bool modifMode;

  ...
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.modifMode ? 'Page Modifier Contact' : 'Page Ajouter
Contact'),
      ),
      body: Form(
        key: globalKey,
        child: _formUI(),
      ),
      bottomNavigationBar: SizedBox(
        height: 100,
        child: Row(
          ...
          children: [
            FormHelper.submitButton(
              widget.modifMode ? "Modifier" : "Ajouter",
              () {
                if (validateAndSave()) {
                  if (widget.modifMode)
                    contactService.modifierContact(contact!).then((value) {
                      Navigator.pop(context);
                    });
                  else
                    contactService.ajouterContact(contact!).then((value) {
                      Navigator.pop(context);
                    });
                }
              }
            )
          ],
        ),
      ),
    );
  }

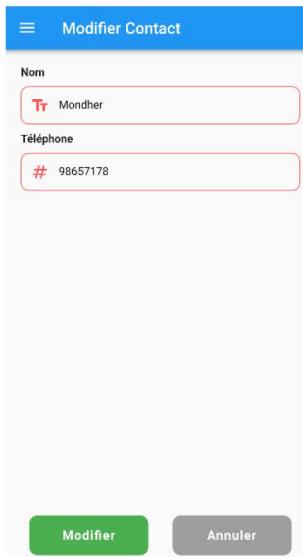
  @override
  void initState() {
    super.initState();
    if (widget.modifMode) contact = widget.contact!;
  }

  _formUI() {
    ...
    initialValue: widget.modifMode ? contact!.nom! : "",
    ...
    initialValue: widget.modifMode ? contact!.tel.toString() : "",
    ...
  }
}

```

### Fichier contact.page.dart

```
...
_buildDataTable(List<Contact> listContacts) {
...
    (Contact c) {
        // Modifier contact
        Navigator.push(
            context,
            MaterialPageRoute(
                builder: (context) => AjoutModifContactPage(
                    modifMode: true,
                    contact: c,
                ),
            ),
        ).then((value) {
            setState(() {});
        });
    },
...
}
...
```



11. Apporter les modifications nécessaires à la page Contact afin de pouvoir supprimer un contact

### Fichier contact.page.dart

```
...
_buildDataTable(List<Contact> listContacts) {
...
    (Contact c) {
        // Supprimer contact
        return showDialog(
            context: context,
            builder: (BuildContext context) {
                return AlertDialog(
                    title: const Text("Supprimer Contact"),
                    content: const Text(
                        "Etes vous sur de vouloir supprimer ce contact?"),
                    actions: [
                        Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            crossAxisAlignment: CrossAxisAlignment.center,
                            children: [
...
```

