



Université de Nouakchott
Faculté des Sciences et Techniques

Développement mobile en Flutter

Dr Ahmed Mohameden
amed.mohameden@gmail.com

À propos du cours

Enseignant : Ahmed Mohameden, docteur en informatique et coordinateur de licence informatique à la FST de l'UNA.

Ce cours de 30 heures concerne le développement mobile en Flutter. Il couvre 53 compétences techniques réparties en 8 chapitres et comprend 35 exemples illustratifs ainsi que 22 activités pratiques dont le code source est disponible sur Github :

<https://github.com/dr-ahmed/flutter>

Prérequis :

- POO ;
- API REST ;
- SGBD relationnel et embarqué ;
- Bases de Python pour le backend Django.

Chapitres

1. Initiation à Dart ;
2. Environnement de travail ;
3. Principaux widgets ;
4. Publication et clonage sur Github ;
5. State management ;
6. Persistance des données via SQLite ;
7. Back-end : Django et Firebase ;
8. Publication sur le Store ;
9. Clean architecture et test-driven development.

Chapitre 1 : initiation à Dart

```
@override
Stream<SeltzerHttpResponse> handle(
  SeltzerHttpRequest request,
  Object data,
) {
  return new HttpClient()
    .openUrl(request.method, Uri.parse(request.url))
    .then((r) async {
      request.headers.forEach(r.headers.add);
      final response = await r.close();
      final headers = <String, String>{};
      response.headers.forEach((name, value) {
        headers[name] = value.join(' ');
      });

      if (WebSocketTransformer.isUpgradeRequest(request)) {
        var socket = await WebSocketTransformer.upgrade(request);
        var channel = new IOWebSocketChannel(socket);
        _socketController.add(new ChannelWebSocket(channel));
      } else {
        _requestController
          .add(new DefaultServerSeltzerHttpRequest.fromHttpRequest(request));
      }

      return new SeltzerHttpResponse.fromBytes(response, headers: headers);
    }).asStream();
}
```

Acquis techniques

À la fin de ce chapitre, vous serez capable de maîtriser les éléments suivants:

- Concepts de base
 - Types primitifs
 - Conditions
 - Boucles
 - Null safety
 - Inférence de type
- Classes
- Structures de données
- Génériques
- Fonctions anonymes
- Expressions Lambda

Rappel historique

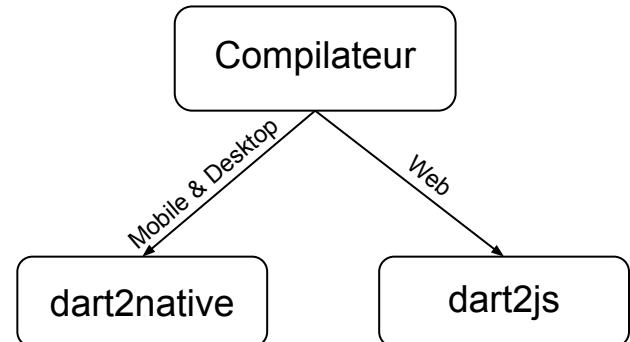
- Août 2010 : Google est accusé d'avoir enfreint le droit de copyright d'Oracle sur Java
- Fin 2011 : Google lance la première version des langages Dart et Kotlin
- Cela a permis à Google plus tard de se débarrasser de la dépendance d'Oracle
- Un scénario similaire s'est déjà passé entre Sun Microsystems et Microsoft
- 2019 : Google considère Kotlin le langage préféré pour Android au lieu de Java
- 2021 : la Cour Suprême des États-Unis confirme l'usage loyal de Java par Google
- Ce jour, la présence de Java dans le mobile chez Google est presque nulle :
 - Natif : Kotlin
 - Multiplateforme : Dart

Dart ?

- Destiné initialement aux applications web
- Langage open source, créé en 2011 par Google
- Inspiré de Java avec le support de :

- Mixins
- Null safety
- Inférence de type
- Génériques réifiés
- Fonctions Lambda

- Normalisé par le comité [TC529](#) de l'Ecma
- Dart en ligne : <https://dart.dev/#try-dart>
- Hello World : `void main() => print('Hello, World!');`



Types primitifs

```
void main() {  
    int age = 23;  
    double overallAverage = 15.74;  
    bool isSingle = true;  
    String firstName = 'Sid\'Ahmed', lastName = "Ebnou";  
    print('''Je m'appelle $firstName $lastName,  
    j'ai $age ans,  
    j'ai une moyenne générale de $overallAverage,  
    et je suis ${isSingle ? 'célibataire' : 'marié'}.'''');  
}
```

List

```
void main() {  
    List<int> list = List.empty(growable: true);  
    List list = [1,2,3,4,5,6];  
    list.add(7);  
    list.removeAt(4);  
    print(list[4]);  
    list.removeRange(2,5);  
    list.removeLast();  
    list.removeWhere((int num) => num % 2 == 0);  
}
```

Set

```
void main() {  
    Set set = {1,2,3,4,5,6};  
    set.add(7);  
    set.add(1);  
    print(set);  
  
    bool isRemoved = set.remove(4);  
    print(isRemoved);  
    print(set);  
  
    Set newSet = set.map((e) => e % 2 != 0 ? 0 : e).toSet();  
    print(newSet);  
    print(set.intersection(newSet));  
}
```

Map

```
void main() {  
    Map map = {'first_name': 'Sid Ahmed', 'age': '23'};  
    map['age'] = 24;  
    print(map['age']);  
    map.addAll({'last_name': 'Ebnou', 'overall_average': '15.74'});  
    map.remove('overall_average');  
    print(map);  
  
    map.removeWhere((key, value) => value == 24);  
    map.removeWhere((key, value) => key.startsWith('f'));  
  
    for(String s in map.keys)  
        print(map[s]);  
  
    map.forEach((key, value) => print('key = $key, value = $value'));  
}
```

Structures conditionnelles

```
enum Field{MPI, PC, BG}
switch (field) {
    case Field.MPI:
        print("Mathématique-Physique-Informatique");
        break;
    case Field.PC:
        print("Physique-Chimie");
        break;
    case Field.BG:
        print("Biologie-Géologie");
        break;
    default: print("Filière inconnue!");
}

if (overallAverage > 15)
    print("Vous êtes autorisé à s'inscrire en Doctorat");
else
    print("Vous n'êtes pas autorisé à s'inscrire en Doctorat");
```

Boucles

for :

```
for(int i = 0; i < list.length; i++)
    print("${list[i]} correspond à l'élément numéro $i");
```

while :

```
var iterator = list.iterator;
while(iterator.moveNext())
    print(iterator.current);
```

forEach :

```
list.forEach((int num) => print(num));
```

Opérateurs de Null Safety (1/3)

Conditional subscript access :

```
List firstList = [1,2,3];
print(firstList[2]);
List? secondList;
print(secondList[2]); // will crash
print(secondList?[2]); // null
```

Conditional member access :

```
String firstName = 'Ahmed';
print(firstName);
String? lastName = null;
print(lastName.length); // will crash
print(lastName?.length); // null
```

Opérateurs de Null Safety (2/3)

Null assertion operator :

```
String firstName = 'Sid Ahmed';
print(firstName);
String? lastName;
print(lastName); // null
print(lastName!); // will crash
```

Null-aware operator :

```
print(0 ?? 1); // 0
print(1 ?? null); // 1
print(null ?? null); // null
print(null ?? null ?? 2); // 2
```

Opérateurs de Null Safety (3/3)

Null-coalescing assignment operator :

```
int? value;  
print(value); // null  
value ??= 5;  
print(value); // null -> 5  
value ??= 6;  
print(value); // no change
```

Null-aware spread operator :

```
List<int> firstList = [1, 2, 3];  
List<String>? secondList;  
print(['Dart', ...?secondList]); // [Dart]  
print([0, ...?secondList, ...firstList]); // [0, 1, 2, 3]  
print(['Flutter!', ...secondList]); // will crash
```

Inférence de type

- Lorsque `var` est défini sans valeur initiale, il est équivalent à `dynamic`
- L'utilisation de `dynamic` peut conduire à des erreurs inattendues!

Mot-clé	Quand ?	Mutabilité		Vérification statique du type
		Valeur	Type	
<code>final</code>	Constant	—	—	+
<code>var</code>	Type initial inconnu	+	—	+
<code>dynamic</code>	Type inconnu	+	+	—
<code>Object</code>	Polymorphisme	+	—	+

Classes

```
class Student {  
    int _age;  
    String name;  
    bool _isSingle;  
  
    Student(this._age, this.name, this._isSingle);  
  
    String get maritalStatus => _isSingle ? 'célibataire' : 'marié';  
  
    void set studentAge(int age) {  
        if(age > 15) this._age = age;  
        else print('âge invalide!');  
    }  
  
    void info() => print('Age : $_age, nom : $name, statut marital : $maritalStatus');  
}  
  
Student(23, 'Khaled', true).info();
```

Paramètres optionnels et nommés

```
class Student {  
    int? age;  
    String? name;  
  
    Student({this.age, this.name});  
  
    void info() => print('age : $age, name : $name');  
}  
  
void main(){  
    Student().info();  
    Student(age:24).info();  
    Student(name:'Khaled').info();  
    Student(age:24,name:'Khaled').info();  
}
```

Héritage

```
class Person {  
    int age;  
    String name;  
  
    Person(this.age, this.name);  
  
    String info() => 'Age : $age, nom : $name';  
}  
  
class Student extends Person{  
    int departement;  
  
    Student(super.age, super.name, this.departement);  
  
    @override  
    String info() => '${super.info()}, departement : $departement';  
}
```

Mixin

- Originaire de LISP
- Appelés Traits en Ruby et Scala
- Une classe munie d'attributs et de méthodes
- Mais sans constructeur et sans héritage
- Permet la réutilisation de code entre classes
- Vu en quelque sorte comme héritage multiple
- Déclaré à travers le mot-clé `mixin` et utilisé via `with`
- On permet d'indiquer que le mixin ne peut être utilisé qu'avec une classe
- Plus d'informations : [Gilad Bracha speech](#).

Exemple (1/2)

```
class Student {  
    int age;  
    String name;  
  
    Student(this.age, this.name);  
  
    String info() => 'Age : $age, nom : $name';  
}  
  
class Supervision { // mixin class  
    int articlesNumber = 0;  
  
    String articles() => 'articles : $articlesNumber';  
}
```

Exemple (2/2)

```
class PhDCandidate extends Student with Supervision {  
    int articlesNumber;  
  
    PhDCandidate(int age, String name, int articlesNumber):  
        this.articlesNumber = articlesNumber,  
        super(age, name);  
  
    @override  
    String info() => '${super.info()}, ${super.articles()}';  
}  
  
void main() {  
    print(PhDCandidate(27, 'Moustapha', 4).info());  
}
```

Génériques

- Concept similaire aux [templates](#) en C++
- Aide au type-safety en vérifiant le type de la classe parente
- Par défaut, les collections sont hétérogènes :

```
List list = ['Khaled', 5, 0.13, true];
```

- On peut aussi déclarer une collection homogène :

```
List<String> list = ['Khaled', 'Mohamed', 'Ahmed'];
```

- Réutilisation du code via le polymorphisme :

```
List<Person> list = [Person(23, 'Khaled'),  
                     Student(25, 'Mohamed', 9)];  
  
for(Person p in list)  
    print(p.info());
```

Classe générique

```
class Stack<T> {
    List<T> _stack = [];

    T pop() => _stack.removeLast();
    void push(T item) => _stack.add(item);

    void printStack() => print(_stack);
}

void main() {
    Stack<int> stk = Stack();
    stk.push(4);
    stk.push(1);
    stk.pop();
    stk.printStack();
}
```

Fonctions anonymes (1/2)

Fonction sans nom!

Exemples en Dart :

```
(a, b) {  
    return a + b;  
};
```

```
List<String> names = ['Ebnou', 'Khaled', 'Mohamed', 'Ahmed'];  
(String item) {  
    for( item in names)  
        print('${names.indexOf(item)}: $item');  
};
```

Fonctions anonymes (2/2)

```
names.forEach(  
    (String item) {  
        print('${names.indexOf(item)}: $item');  
    }  
) ;
```

Exemple en Flutter :

```
ElevatedButton(  
    onPressed: () {  
        Navigator.pushNamed(context, '/home');  
    }  
    title: 'Accueil',  
)
```

Fonctions Lambda

- Lambda est une notation pour représenter les courtes fonctions
- Les fonctions Lambda sont également appelées Arrow Functions
- La syntaxe Lambda ne permet de renvoyer qu'une seule expression, pas de bloc
- Recommandé pour l'exécution des instructions courts n'ayant pas besoin de référence
- Exemple : int addNum(int a, int b) => a + b;
- Functions Lambda anonymes :

```
names.forEach((item) => print('${names.indexOf(item)}: $item'));  
print([1, 2, 3].where((x) => x.isOdd).map((x) => x - 3));  
File('log.txt')  
    .openRead()  
    .map(utf8.decode)  
    .transform(new LineSplitter())  
    .forEach((l) => print('line: $l'));
```

Lambda ≠ Anonyme

Lambda		Non Lambda	
Anonyme	Non anonyme	Anonyme	Non anonyme
(a, b) => a + b;	int sum(int a, int b) => a + b;	(a, b) { return a + b; }	int sum(int a, int b){ return a + b; }

Activité 1

Répondez par Vrai ou Faux en justifiant.

Expression	Vrai ou Faux
Le type char existe en Dart	
Les fonctions sont des objets	
Dart supporte l'héritage multiple	
Dart supporte les classes imbriquées	
Dart supporte les fonctions imbriquées	
Dart fournit des getters/setters implicites	
Le super constructeur peut être appelé en première position	

Activité 2

Répondez par Vrai ou Faux en justifiant.

Expression	Vrai ou Faux
Une mixin ne doit pas hériter d'une classe	
Une mixin ne peut pas avoir un constructeur	
Les noms des objets statiques commençant par _	
Dart n'a pas de mots-clés public, private et protected	
Une variable enum peut être déclarée dans une classe	
Tout objet doit être soit initialisé, soit déclaré en tant que nullable	
L'opérateur ! permet de ne pas lancer une exception en cas de nul	

Activité 3

Donnez la signification des mots-clés suivants :

- get
- set
- with
- show
- hide
- await
- async
- super
- mixin
- export
- dynamic
- factory
- required

Activité 4

Compléter le tableau ci-dessous:

Opérateur	Signification
? []	
.	
? .	
?	
!	
??	
? ?=	
... ?	

Activité 5

Soit le constructeur suivant :

```
Student(int age, double overallAverage, String name) :  
    this.age = age,  
    this.overallAverage = overallAverage,  
    this.name = name;
```

Proposez, en une seule ligne, un constructeur équivalent.

Activité 6

Soit la classe `Bicycle`, définie dans [la documentation Java sur Oracle](#).

```
public class Bicycle {  
    private int cadence;  
    private int gear;  
    private int speed;  
  
    public Bicycle(int startCadence, int startSpeed, int startGear) {  
        gear = startGear;  
        cadence = startCadence;  
        speed = startSpeed;  
    }  
    ...  
}
```

Proposez une classe `Dart` similaire, en 23 lignes au lieu de 40.

Activité 7

Qu'affiche ce code ?

```
String fun(int l) => 'B${'o' * l}m!';  
  
void main() {  
    final values = new List<int>.generate(5, (i) => i + 1);  
    values.map(fun).forEach(print);  
}
```

Proposer une version simplifiée via for avec des valeurs implicites du tableau.

Activité 8

Qu'affiche ce code ?

```
void main() {  
    List<String> names = ['Khaled', 'Mohamed', 'Sid\'Ahmed', 'Ebnou'];  
    names.where((name) => name.contains('med')).forEach(print);  
}
```

Proposer une version équivalente en utilisant while et les expressions régulières.

Activité 9 (1/4)

Expliquer le code suivant :

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(primarySwatch: Colors.blue),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

Activité 9 (2/4)

```
class MyHomePage extends StatefulWidget {  
    final String title;  
  
    const MyHomePage({  
        Key? key,  
        required this.title,  
    }) : super(key: key);  
  
    @override  
    State<MyHomePage> createState() => _MyHomePageState();  
}
```

Activité 9 (3/4)

```
class _MyHomePageState extends State<MyHomePage> {
    int _counter = 0;

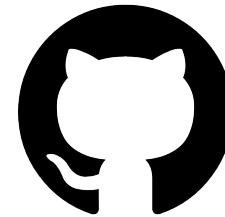
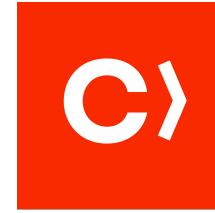
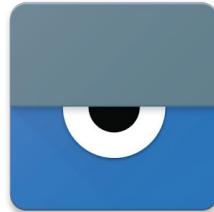
    void _incrementCounter() {
        setState(() {
            _counter++;
        });
    }

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text(widget.title),
            ),
        );
    }
}
```

Activité 9 (4/4)

```
body: Center(  
    child: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
            const Text('You have pushed the button this many times:'),  
            Text(  
                '$_counter',  
                style: Theme.of(context).textTheme.headline4,  
            ))],  
        floatingActionButton: FloatingActionButton(  
            onPressed: _incrementCounter,  
            tooltip: 'Increment',  
            child: const Icon(Icons.add),  
        )),  
    }},
```

Chapitre 2 : environnement de travail



Acquis techniques

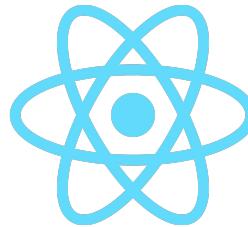
À la fin de ce chapitre, vous serez capable de :

- Comprendre les points forts et faibles des différents frameworks mobiles ;
- Installer et configurer l'environnement de travail ;
- Créer un nouveau projet Flutter sur VSC via deux méthodes ;
- Saisir les rôles des principaux éléments de l'arborescence du projet ;
- Lancer une application Flutter sur un téléphone Android ou iPhone ;
- Exécuter une application Flutter sur un émulateur Android ou iPhone ;
- Mettre en miroir une application Flutter sur Android et iPhone ;
- Apporter quelques rectifications au code Hello World ;
- Exporter l'apk/ipa pour une utilisation éventuelle ;
- Visualiser le code Java équivalent au code Dart en décompilant l'APK.

Pourquoi le cross-platform ?

Technologie \ Critère	Natif	Cross-platform	RWD	PWA
Multiplateforme	—	+	+	+
Coût	—	+	+	+
Installation	—	—	+	+
Mise à jour	—	—	+	+
Performance	+	—	—	—
Hors ligne	+	+	—	+
Notifications	+	+	—	+
Écran d'accueil	+	+	—	+
Plein écran	+	+	—	+

Pourquoi Flutter ?



Critère Framework	Création	Langage	Performance	Widgets	Architecture
Flutter	Google, 2017	Dart	Proche du natif	Exhaustif	Skia
React native	Facebook, 2015	JS	Assez robuste	Minimal	Bridge
Ionic	Drifty Co., 2013	HTML, CSS & JS	Assez robuste	Standard	Cordova

Premiers pas

- Téléchargez et décompressez le SDK Flutter ;
- Ajoutez /flutter/bin aux variables d'environnement ;
- flutter --version
- flutter create test ;
- cd test ;
- flutter run ;
- flutter doctor ;
- Installez Visual Studio Code ainsi que l'extension Flutter ;
- Android : Android Studio, Android SDK et AVD ;
- iOS : Xcode et CocoaPods.

Arborescence du projet

Élément	Description
dart_tool	Chemin et version des packages (package_config), données en cache (flutter_build), etc.
idea	Bibliothèques et modules de l'application.
android	Version Android de l'application.
build	Fichiers APK et IPA, fichiers de police typographique, etc.
ios	Version iOS de l'application.
lib	Code source de l'application.
test	Classes de tests unitaire, tests d'intégration et test des widgets.
web	Version web de l'application.
pubspec.yaml	Dépendances requises, contraintes sur la SDK Flutter, etc.

Nouveau projet

→ Ligne de commande :

- flutter create test ;
- cd test ;
- flutter run -d chrome ;

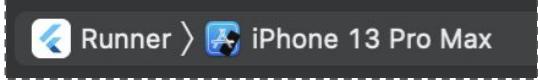
→ Interface graphique :

- Command + Shift + P ;
- New Project ;
- Application ;
- Choisir un dossier ;
- Saisir le nom de l'application.

Lancement d'application sur Android

- Paramètres
 - > À propos du téléphone
 - > Informations sur le logiciel
 - > Numéro de version
 - > Cliquer dessus 7 fois
 - > Options de développement
 - > Activer le débogage USB
- Connecter le téléphone à l'ordinateur via un câble USB ;
- Sélectionner le téléphone au niveau du menu des émulateurs dans VSC ;
- Lancer l'application.

Lancement d'application sur iPhone

- Connecter votre iPhone au Mac via USB ;
- Cliquer sur “Faire confiance à cet ordinateur” ;
- open ios/Runner.xcworkspace ;
- Sélectionner votre iPhone de cette liste 
- Cliquer sur Runner, premier élément à gauche ;
- Onglet General, rectifier ‘example’ au niveau du nom du package ;
- Onglet Signing & Capabilities, Add an account ;
- Vérifier que votre iPhone est sélectionnée et n'est pas verrouillée ;
- Cliquer sur le bouton Start en haut à gauche ;
- Saisir le Mac user password puis cliquer sur ‘Always allow’.
- Lancer désormais l'application à partir de VSC.

Versions iOS supportées

Finder

- > Applications
- > Xcode
 - > Clic droit
 - > Show Package Contents
 - > Contents
 - > Developer
 - > Platforms
 - > iPhoneOS.platform
 - > DeviceSupport

Plus d'informations : <https://developer.apple.com/support/xcode/>

Création d'un émulateur Android et iPhone

- Android
 - > Android Studio, More actions
 - > AVD Manager
 - > Create Virtual Device
 - > Choisir le device puis Next
 - > Finish
- iOS : Command + espace, Simulator
- Sélectionner l'émulateur au niveau du menu des émulateurs dans VSC ;
- Lancer l'application.

Spécification d'émulateur

```
→ flutter emulators  
→ flutter devices  
→ flutter emulators --launch nexus_5x_api_29  
→ flutter devices  
→ flutter run -d android  
→ flutter emulators --launch apple_ios_simulator  
→ flutter run -d iphone  
→ code .
```

Mise en miroir sur Android et iPhone

- Installer Vysor : <https://www.vysor.io/download/>
- Connecter le téléphone à l'ordinateur via un câble USB ;
- Android : Activer le débogage USB;
- iPhone : connecter également le téléphone au Mac via Bluetooth ;
- Autoriser l'accès ;
- Cliquer sur le bouton Start pour commencer la mise en miroir.

- QuickTime Player
 - > File
 - > New Movie Recording
 - > Sélectionner votre iPhone de la liste déroulante à côté de l'icône d'enregistrement.

Exportation d'APK et d'IPA

- Android
 - flutter build apk
 - Chemin : build/app/outputs/flutter-apk/app-debug.apk
- iPhone
 - flutter build ios
 - Chemin : build/iOS/iphoneos/Runner.app
 - Placer Runner.app dans un nouveau dossier Payload
 - Compresser Payload au format zip
 - Changer l'extension de .zip à .ipa.
- Envoyez l'APK ou l'IPA par email ou partagez-le sur le Cloud.

Décompilation de l'APK

- Installer JADX : <https://github.com/skylot/jadx>
- git clone https://github.com/skylot/jadx.git
- cd jadx
- ./gradlew dist
- Command + O, sélectionner l'APK

```
public class FlutterActivity extends Activity implements FlutterActivityAndFragmentDelegate.Host, LifecycleOwner {
    public static final int FLUTTER_VIEW_ID = ViewUtils.generateViewId(61938);
    private static final String TAG = "FlutterActivity";
    protected FlutterActivityAndFragmentDelegate delegate;
    private LifecycleRegistry lifecycle = new LifecycleRegistry(this);

    public static Intent createDefaultIntent(Context launchContext) {
        return withNewEngine().build(launchContext);
    }

    public static NewEngineIntentBuilder withNewEngine() {
        return new NewEngineIntentBuilder(FlutterActivity.class);
    }
}
```

Activité 10

Reliez les éléments de la première colonne à leurs correspondants de la seconde.

Élément	Description
Gradle	Gestionnaire de dépendances
SDK	Moteur de production
Émulateur	Ensemble d'outils facilitant le développement de logiciels
CocoaPods	Application de mise en miroir
Vysor	Logiciel permettant de simuler l'environnement d'un OS mobile

Activité 11

Répondre par Vrai ou Faux en justifiant.

- En exécutant la commande `flutter doctor`, on voit apparaître Android Studio parmi les éléments nécessaires à fournir ;
- L'installation d'Android Studio est nécessaire pour lancer une app Flutter ;
- À travers Android Studio, on peut développer uniquement des apps Android ;
- On peut lancer une application Flutter sur FireFox ;
- Le débogage peut se faire sur n'importe quel navigateur à travers DevTools ;
- À travers la mise en miroir, on peut lancer une application Flutter via VSC ;
- L'installation de l'extension Flutter n'installe pas forcément l'extension Dart.

Activité 12

Soient les tâches suivantes, donnez les commandes correspondantes.

Créer un projet Flutter

Lancer l'application Flutter

Ouvrir VSC

Lister les émulateurs

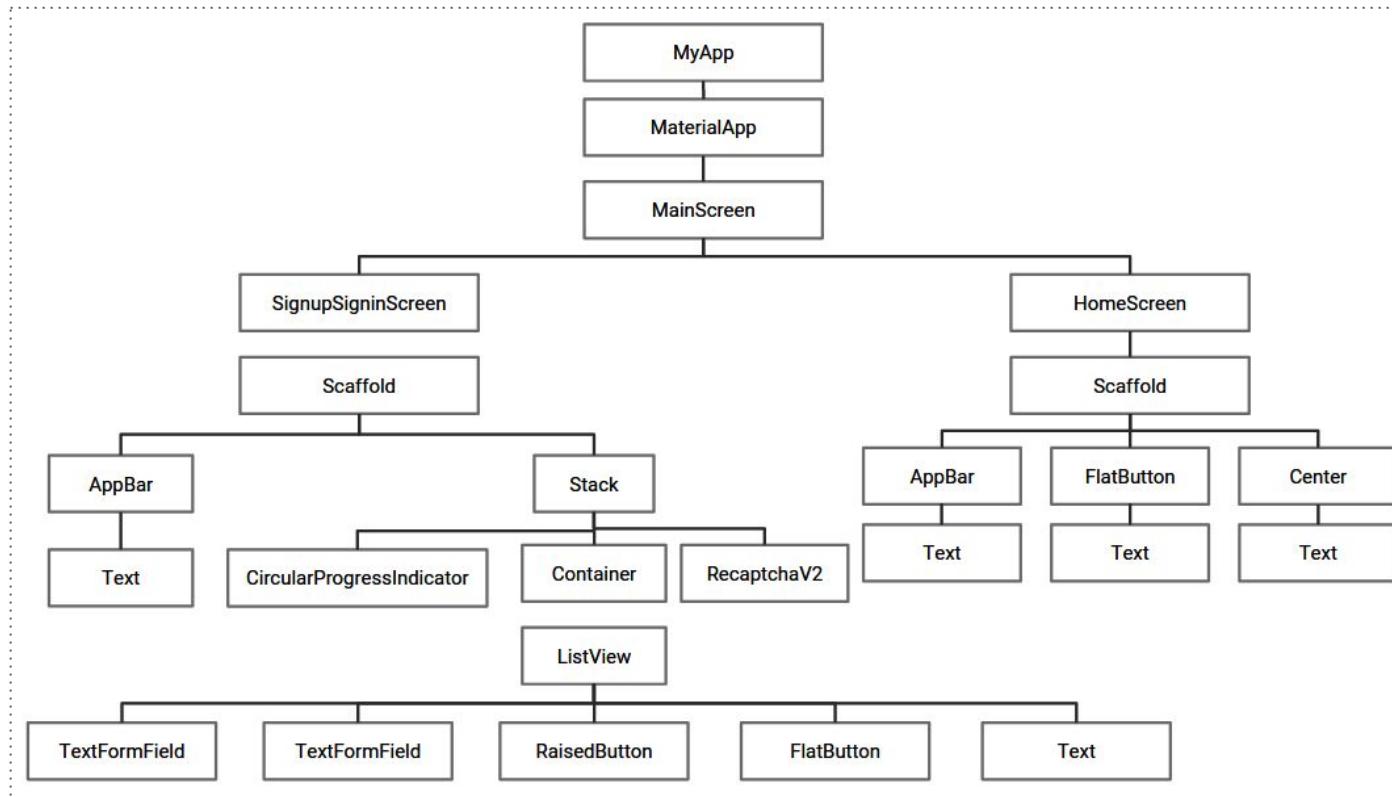
Lister les appareils connectés

Lancer l'application sur l'émulateur
Android

Faire un Hot reload

Quitter l'application

Chapitre 3 : principaux widgets



Acquis techniques

À la fin de ce chapitre, vous serez capable de :

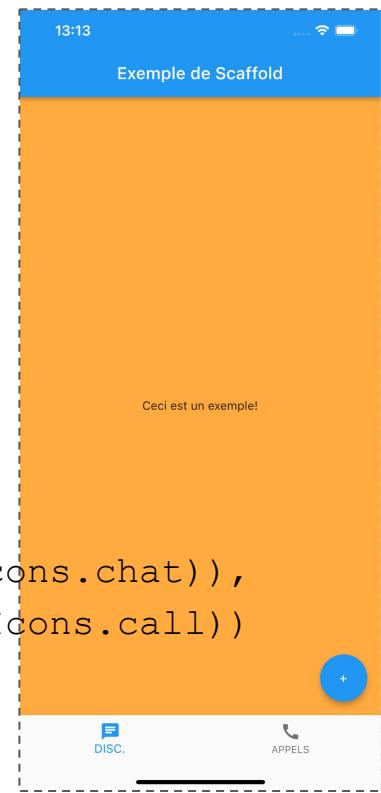
- Comprendre ce qu'est un Widget en Flutter ;
- Distinguer les rôles des principaux widgets ;
- Pratiquer les principales propriétés de ces widgets ;
- Savoir positionner différents éléments à travers Row, Column et Stack.

Principaux widgets

- Scaffold
- FloatingActionButton
- BottomNavigationBar
- Text
- ElevatedButton et Toast
- TextField
- Container
- Image
- Row
- Column
- ListView
- Stack

Scaffold

```
Scaffold(  
    backgroundColor: Colors.orangeAccent,  
    appBar: AppBar(title: Text('Exemple de Scaffold')),  
    body: Center(child: Text('Ceci est un exemple!')),  
    floatingActionButton: FloatingActionButton(  
        child: Text('\u2795'),  
        onPressed: () => print('J\'ai reçu un clic!'))  
,  
    bottomNavigationBar: BottomNavigationBar(items: [  
        BottomNavigationBarItem(label: 'DISC.', icon: Icon(Icons.chat)),  
        BottomNavigationBarItem(label: 'APPELS', icon: Icon(Icons.call))  
    ])  
)
```



Text

```
Center(  
    child: Text('السلام عليكم و رحمة الله',  
        textAlign: TextAlign.justify,  
        textDirection: TextDirection rtl,  
        style: TextStyle(  
            fontSize: 35,  
            color: Colors.blueAccent,  
            fontWeight: FontWeight.bold,  
            backgroundColor: Colors.orangeAccent,  
            fontFamily: 'Courier',  
            decoration: TextDecoration.combine([  
                TextDecoration.underline,  
                TextDecoration.overline  
            ]),  
            decorationColor: Colors.black,  
            decorationStyle: TextDecorationStyle.dashed)  
    )  
)
```



ElevatedButton & Toast

```
flutter pub add fluttertoast & brew install cocoapods
```

```
ElevatedButton(  
    child: Text('Ajouter'),  
    style: ElevatedButton.styleFrom(  
        minimumSize: Size(280, 80),  
        textStyle: TextStyle(fontSize: 28),  
        backgroundColor: Colors.orange,  
        foregroundColor: Colors.black),  
    onPressed: () => Fluttertoast.showToast(  
        msg: "Ajout avec succès",  
        backgroundColor: Colors.blueAccent,  
        textColor: Colors.black,  
        fontSize: 18,  
        toastLength: Toast.LENGTH_SHORT,  
        gravity: ToastGravity.BOTTOM)  
)
```



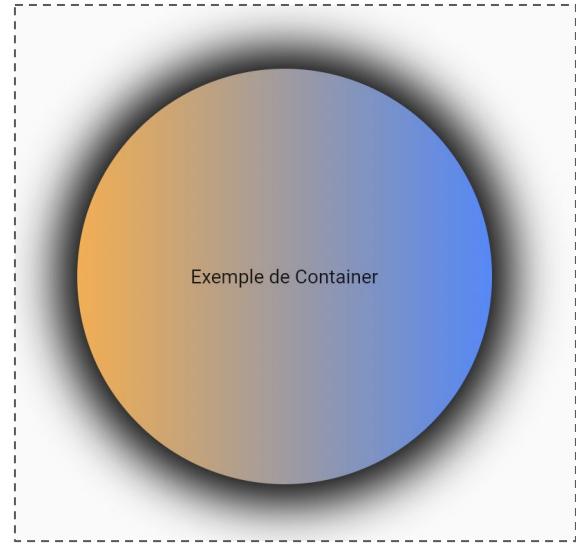
TextField

```
Padding(  
  padding: EdgeInsets.all(15),  
  child: Center(  
    child: TextField(  
      maxLength: 4,  
      keyboardType: TextInputType.number,  
      obscureText: true,  
      decoration: InputDecoration(  
        labelText: 'PIN',  
        hintText: 'Code PIN',  
        filled: true,  
        fillColor: Colors.orangeAccent,  
        prefixIcon: Icon(Icons.login),  
        border: OutlineInputBorder(borderSide: BorderSide(color: Colors.black)),  
        focusedBorder: OutlineInputBorder(borderSide: BorderSide(color: Colors.black))),  
    ),  
)  
)
```



Container

```
Center(  
    child: Container(  
        width: 300,  
        height: 300,  
        alignment: Alignment.center,  
        child: Text('Exemple de Container'),  
        decoration: BoxDecoration(  
            shape: BoxShape.circle,  
            gradient: LinearGradient(  
                begin: Alignment.centerLeft,  
                end: Alignment.centerRight,  
                colors: [Colors.orangeAccent, Colors.blueAccent]),  
            boxShadow: [BoxShadow(color: Colors.black, blurRadius: 30, spreadRadius: 15)]  
        )  
    )  
)
```



Image

→ Local :

- mkdir assets ;
- mv ../image.png assets/image.png ;
- pubspec.yaml :
assets:
 - assets/
- Image.asset('assets/image.jpg') .

→ En ligne :

```
Image.network('https://picsum.photos/250')
```



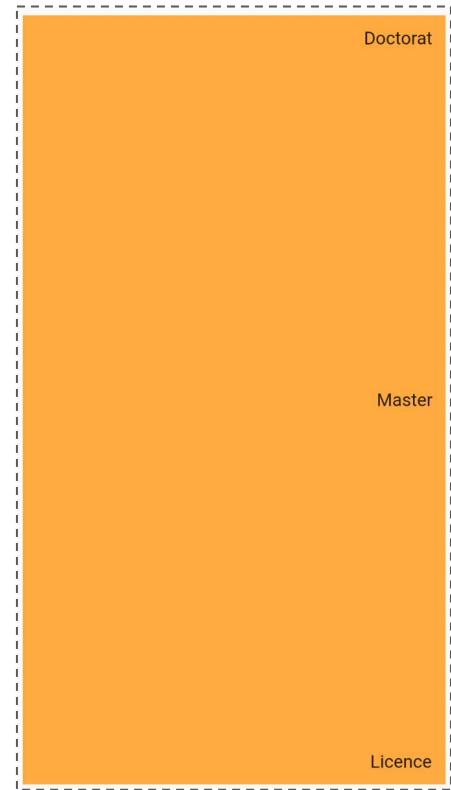
Row

```
Container(  
    height: 200,  
    color: Colors.orangeAccent,  
    padding: EdgeInsets.all(10),  
    child: Row(  
        mainAxisAlignment: MainAxisAlignment.max,  
        textDirection: TextDirection rtl,  
        verticalDirection: VerticalDirection.up,  
        crossAxisAlignment: CrossAxisAlignment.start,  
        mainAxisSize: MainAxisSize.spaceAround,  
        children: [  
            Text('Doctorat'),  
            Text('Master'),  
            Text('Licence')  
        ]  
    )  
)
```



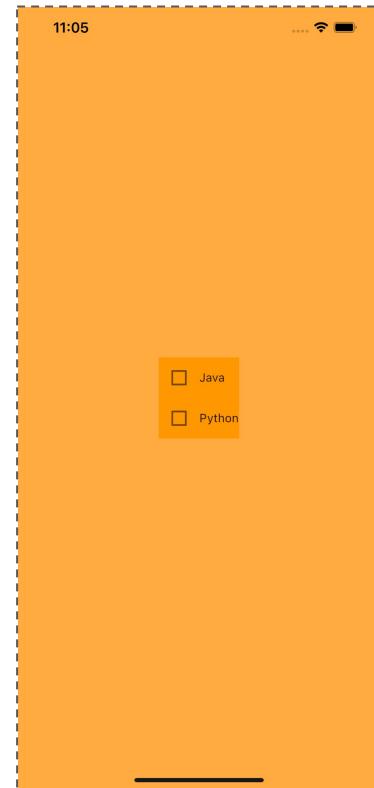
Column

```
Container(  
    width: 400,  
    color: Colors.orangeAccent,  
    padding: EdgeInsets.all(10),  
    child: Column(  
        mainAxisSize: MainAxisSize.max,  
        crossAxisAlignment: CrossAxisAlignment.end,  
        mainAxisAlignment: MainAxisAlignment.spaceBetween,  
        children: [  
            Text('Doctorat'),  
            Text('Master'),  
            Text('Licence')  
        ]  
    )  
)
```



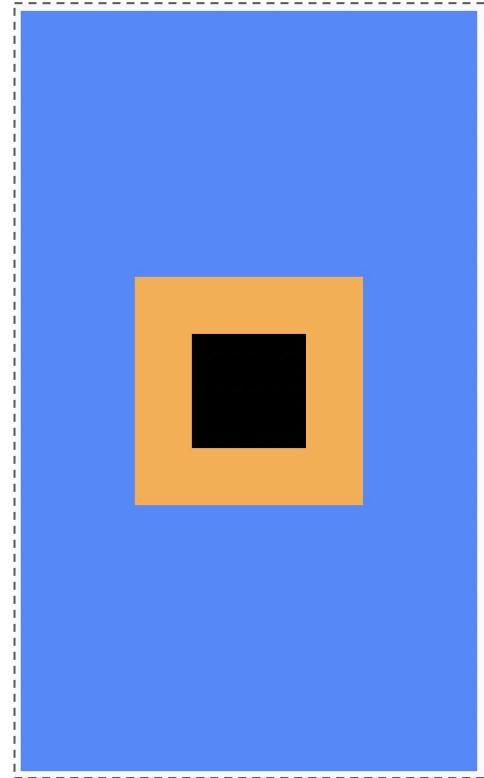
Row et Column

```
Column(  
    mainAxisSize: MainAxisSize.min,  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
        Row(mainAxisSize: MainAxisSize.min, children: [  
            Text('Java'),  
            Checkbox(value: false, onChanged: (bool? value) {})  
        ]),  
        Row(mainAxisSize: MainAxisSize.min, children: [  
            Text('Python'),  
            Checkbox(value: false, onChanged: (bool? value) {})  
        ])  
    ]  
)
```



Stack

```
Container(  
    width: 400,  
    height: 700,  
    color: Colors.blueAccent,  
    child: Stack(  
        alignment: Alignment.center,  
        children: [  
            Container(  
                width: 200,  
                height: 200,  
                color: Colors.orangeAccent,  
            ),  
            Container(  
                width: 100,  
                height: 100,  
                color: Colors.black,  
            )  
        ]  
    )  
)
```



ListView

```
ListView(  
  children: [  
    ListTile(title: Text('Licence')),  
    ListTile(title: Text('Master')),  
    ListTile(title: Text('Doctorat')),  
    ElevatedButton(  
      child: Text('Quitter'),  
      onPressed: () {}  
    )  
  ]  
)
```



Activité 13

Réalisez ces interfaces via Row et Column.

The image displays two side-by-side screenshots of mobile application interfaces, likely from an iPhone, separated by a vertical dashed line.

Screenshot 1 (Left): User Profile Edit Screen

- Top: Circular user profile icon of a man in a suit.
- Bottom: Three circular buttons: "Ajouter" (unselected), "Modifier" (selected), and "Supprimer" (unselected).
- Form fields:
 - Nom: Text input field.
 - Prénom: Text input field.
 - Adresse: Text input field.
- Checklist:
 - Baccalauréat: checked
 - BTS: unselected
 - Licence: checked
 - Master: unselected
 - Doctorat: unselected
- Bottom right: A large blue button labeled "CONFIRMER" with the word "Résultat" below it.

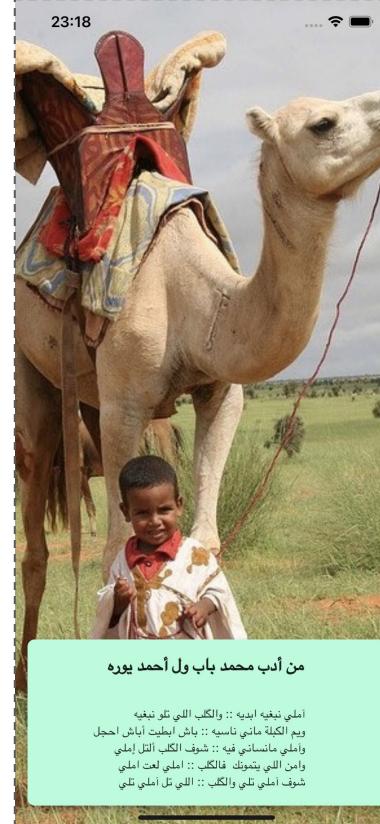
Screenshot 2 (Right): Application Usage Statistics

- Top: Circular user profile icon of a woman with red hair.
- Section title: "Utilisateur de l'application"
- Table:

Élément	Nombre
Rendez-vous	1
Suivi	2
Notification	7
Rendez-vous	0
Urgence	0
Suivi	0
- Bottom right: "Flutter CLINIQUE" logo with "Envoyer" and "Annuler" buttons.

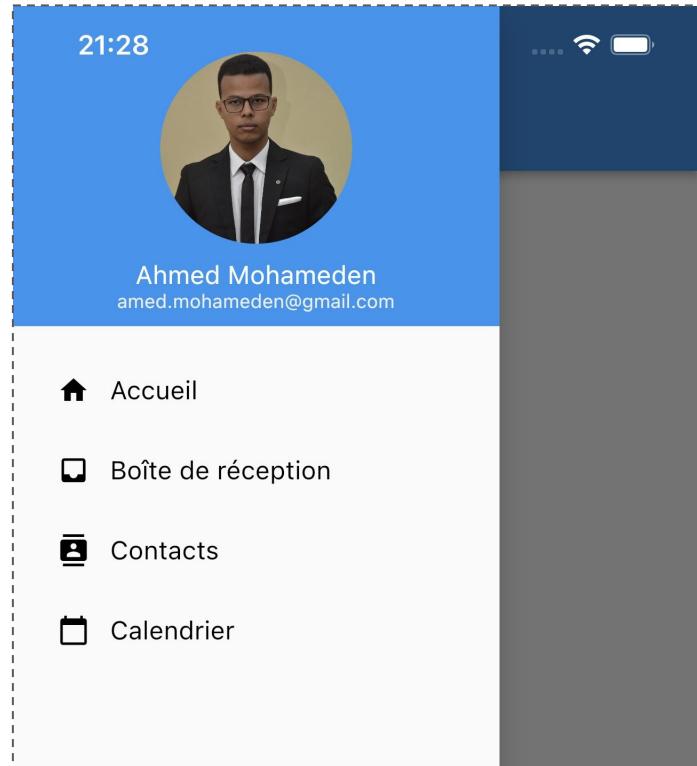
Activité 14

Réalisez cette interface via Stack.



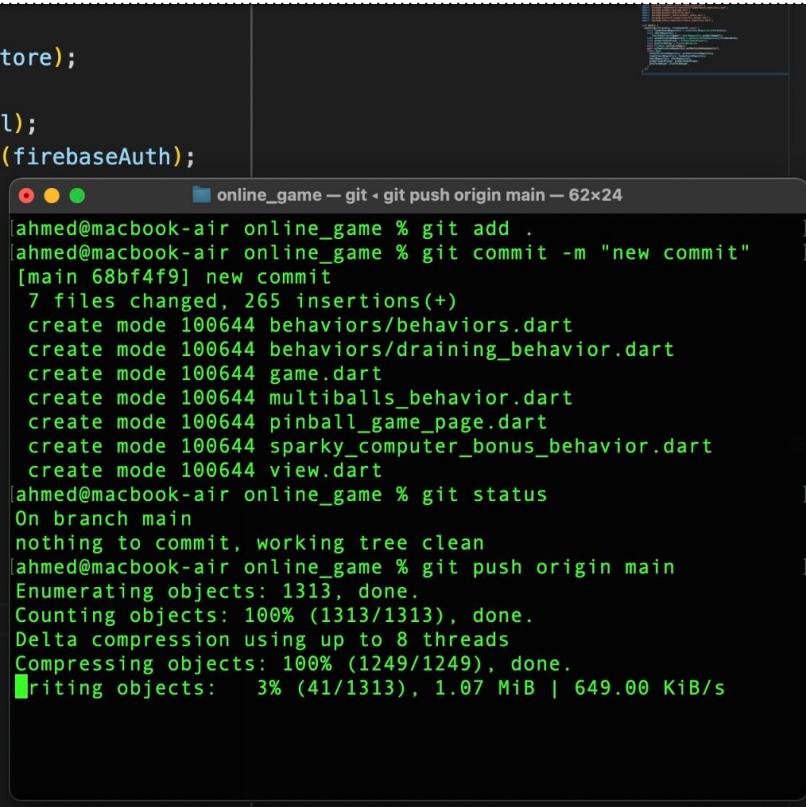
Activité 15

Réalisez une interface similaire en utilisant Drawer.



Chapitre 4 : publication et clonage sur Github

```
bootstrap(firestore, firebaseAuth) async {
    final leaderboardRepository = LeaderboardRepository(firestore);
    const shareRepository =
        ShareRepository(appUrl: ShareRepository.pinballGameUrl);
    final authenticationRepository = AuthenticationRepository(firebaseAuth);
    final pinballAudioPlayer = PinballAudioPlayer();
    final platformHelper = PlatformHelper();
    await Firebase.initializeApp();
    await authenticationRepository.authenticateAnonymously();
    return App(
        authenticationRepository: authenticationRepository,
        leaderboardRepository: leaderboardRepository,
        shareRepository: shareRepository,
        pinballAudioPlayer: pinballAudioPlayer,
        platformHelper: platformHelper,
    );
}
}
```



The terminal window shows the following command and its output:

```
online_game — git - git push origin main — 62x24
ahmed@macbook-air online_game % git add .
ahmed@macbook-air online_game % git commit -m "new commit"
[main 68bf4f9] new commit
 7 files changed, 265 insertions(+)
  create mode 100644 behaviors/behaviors.dart
  create mode 100644 behaviors/draining_behavior.dart
  create mode 100644 game.dart
  create mode 100644 multiballs_behavior.dart
  create mode 100644 pinball_game_page.dart
  create mode 100644 sparky_computer_bonus_behavior.dart
  create mode 100644 view.dart
ahmed@macbook-air online_game % git status
On branch main
nothing to commit, working tree clean
ahmed@macbook-air online_game % git push origin main
Enumerating objects: 1313, done.
Counting objects: 100% (1313/1313), done.
Delta compression using up to 8 threads
Compressing objects: 100% (1249/1249), done.
Writing objects: 3% (41/1313), 1.07 MiB | 649.00 KiB/s
```

Acquis techniques

À la fin de ce chapitre, vous serez capable de :

- Achever la configuration de Git ;
- Comprendre comment fonctionnent Git et Github ;
- Créer un dépôt sur Github ;
- Publier un projet sur Github ;
- Soumettre les modifications d'un projet ;
- Synchroniser les changements d'un projet ;
- Associer un token Github à un plugin ;
- Supprimer un projet de Github ;
- Convertir la portée d'un projet de public à privé et vice-versa.

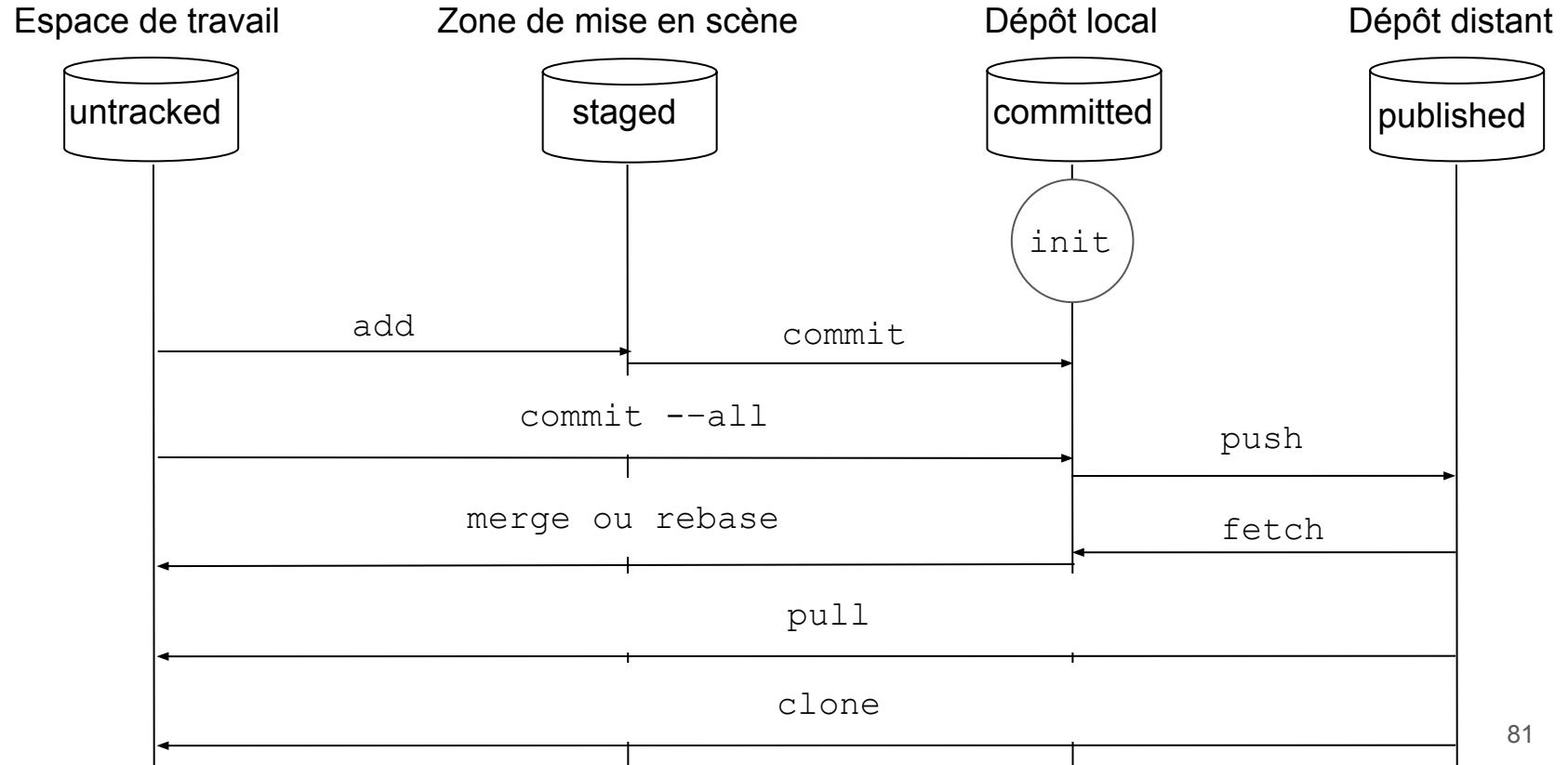
Configuration de Git

- Installer git : <https://git-scm.com/downloads>
- git --version
- git config user.name
- git config --global user.name 'dr-ahmed'
- git config user.email
- git config --global user.email 'amed.mohameden@gmail.com'
- git branch
- git config --global init.defaultBranch main

Principales commandes

Commande	Description
init	Créer un dépôt vide ou réinitialiser un existant.
add	Ajouter des fichiers à la zone de mise en scène.
commit	Enregistrer les modifications dans le dépôt local.
commit --all	Committer les rectifications des fichiers mis en scène.
push	Apporter les changements du dépôt local au dépôt distant.
fetch	Télécharger les changements du dépôt distant.
merge	Fusionner les modifications d'une branche dans la branche cible.
rebase	Déplacer les modifications vers une nouvelle branche.
pull	Rectifier le dépôt local sur la base des changements du dépôt distant
clone	Cloner un dépôt distant dans un nouveau répertoire.

Flux de travail



Publication, mise à jour et clonage sur Github

- Créer un compte Github : <https://github.com/signup>
- Créer un dépôt nommé flutter_demo
- Créer un nouveau projet Flutter
- git init
- git add .
- git commit -m "premier commit"
- git remote
add origin https://github.com/dr-ahmed/flutter_demo.git
- git push -u origin main

Publication, mise à jour et clonage sur Github

- Génération de token :
 - Settings
 - > Developer settings
 - > Personal access tokens
 - > Generate new token.
- Accès : repo, read:org et gist.
- Mise à jour : git pull --rebase origin
- Clonage : git clone
https://github.com/dr-ahmed/flutter_demo.git

Activité 16

Reliez les éléments de la première colonne à leurs correspondants de la seconde.

Élément	Description
init	Ajouter un fichier à la zone de mise en scène
add	Télécharger les modifications du dépôt distant
commit	Apporter les modifications du dépôt distant
push	Initialiser le répertoire Git
fetch	Transférer le fichier vers le dépôt local
pull	Publier le fichier sur le dépôt distant

Activité 17

Répondre par Vrai ou Faux en justifiant.

- pull = fetch + rebase;
- La zone de mise en scène comprend les fichiers comités ;
- La commande reset désigne l'inverse de pull ;
- La commande init transfère les fichiers du projet dans un dossier .git ;
- commit --all = add + commit;
- La commande push désigne l'inverse de fetch ;
- pull et clone désignent la même chose ;
- rebase fusionne les modifications dans une seule branche ;
- merge crée une nouvelle branche à partir des récentes modifications.

Activité 18

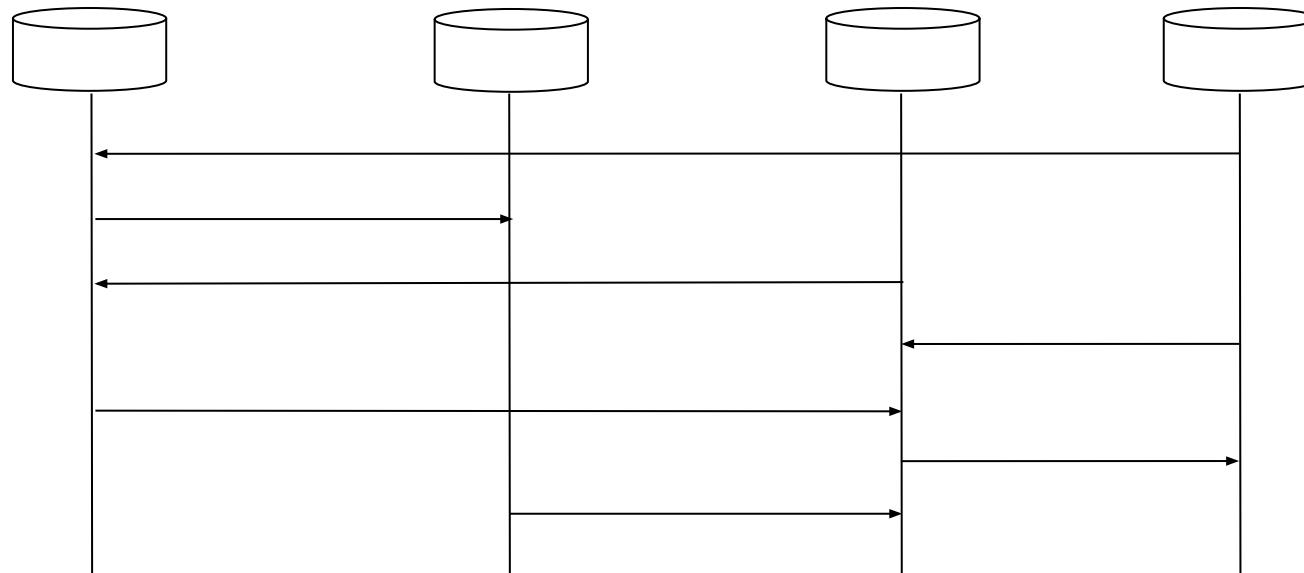
Donnez les commandes correspondantes :

- Réinitialiser le dépôt local ;
- Ajouter un fichier à la zone de mise en scène ;
- Vérifier l'état de la branche et de la zone de mise en scène ;
- Transférer le fichier vers le dépôt local ;
- Vérifier de nouveau l'état de la branche et de la zone de mise en scène ;
- Créer une nouvelle branche ;
- Lister l'ensemble des branches ;
- Publier les mises à jour de la branche main
- Cloner le projet dans un nouveau répertoire.

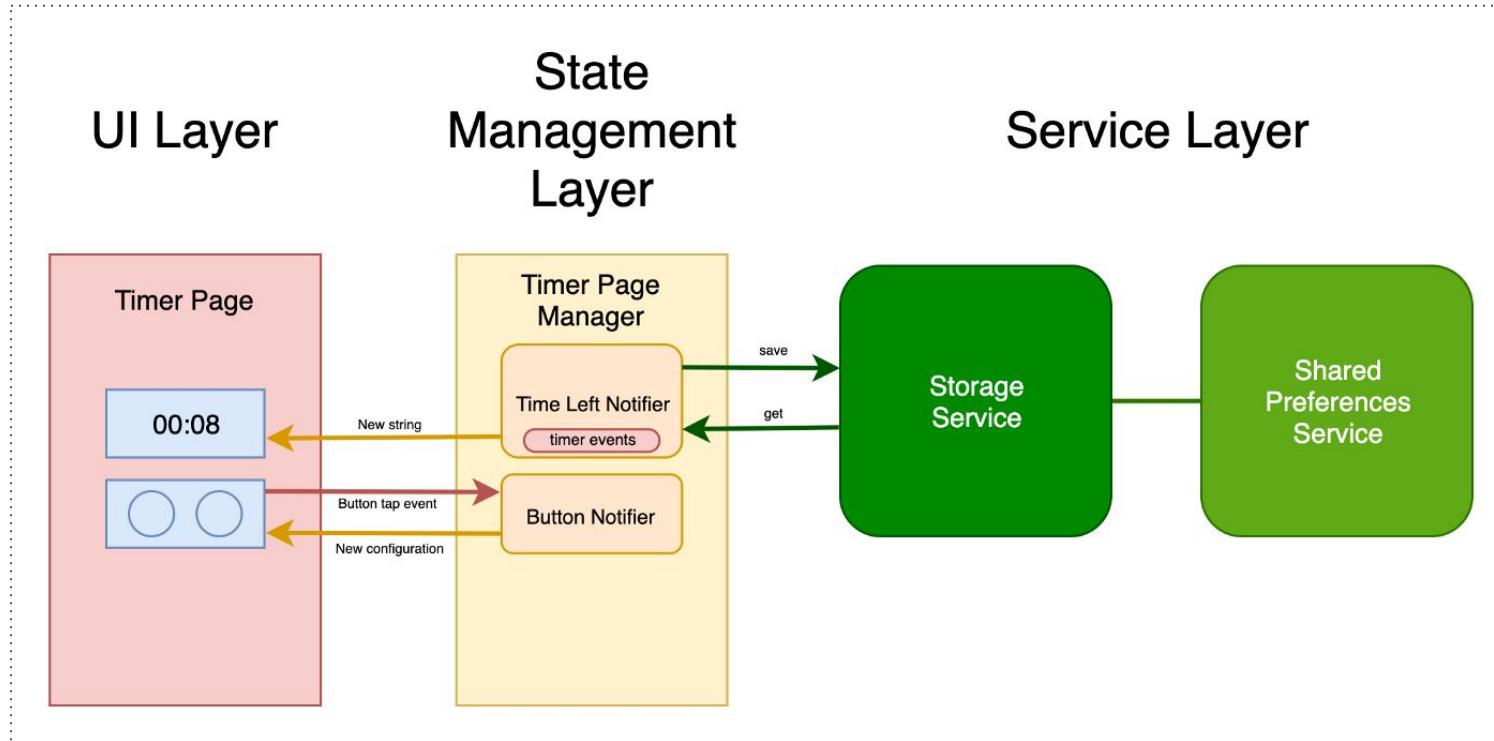
Activité 19

Compléter le schéma ci-dessous par les commandes suivantes :

add, commit, commit --all, push, fetch, merge, rebase, pull ou clone.



Chapitre 5 : state management



Acquis techniques

À la fin de ce chapitre, vous serez capable de :

- Définir ce qu'est un State ;
- Saisir la nécessité de State management ;
- Distinguer les différentes approches de gestion de state ;
- Comprendre :
 - ◆ Comment fonctionnent BLoC et Provider ;
 - ◆ Le mécanisme de Navigation 1.0 et 2.0.

State management ?

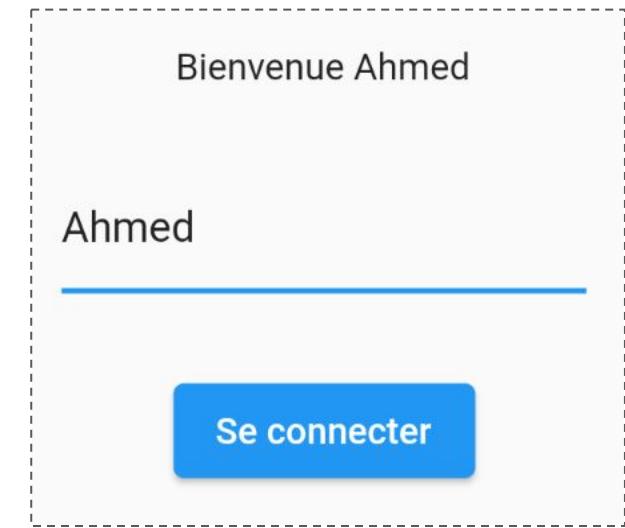
- State : état des données dans un widget à un moment donné ;
- State management : gestion automatique de state, notamment ;
 - Rafraîchissement automatique des widgets ;
 - Passage des données entre widgets.
- Ce n'est pas spécifique à Flutter ;
- Inhérent à tous les frameworks de front-end ;
 - Android : Jetpack Compose ;
 - iOS : SwiftUI.
- StatelessWidget : widgets statiques, immutable state ;
- StatefulWidget : widgets dynamiques, mutable state.

StatefulWidget

```
String name = '';
final myController = TextEditingController();

void notify() {
    setState(() {
        name = 'Bienvenue ${myController.text}';
    });
}

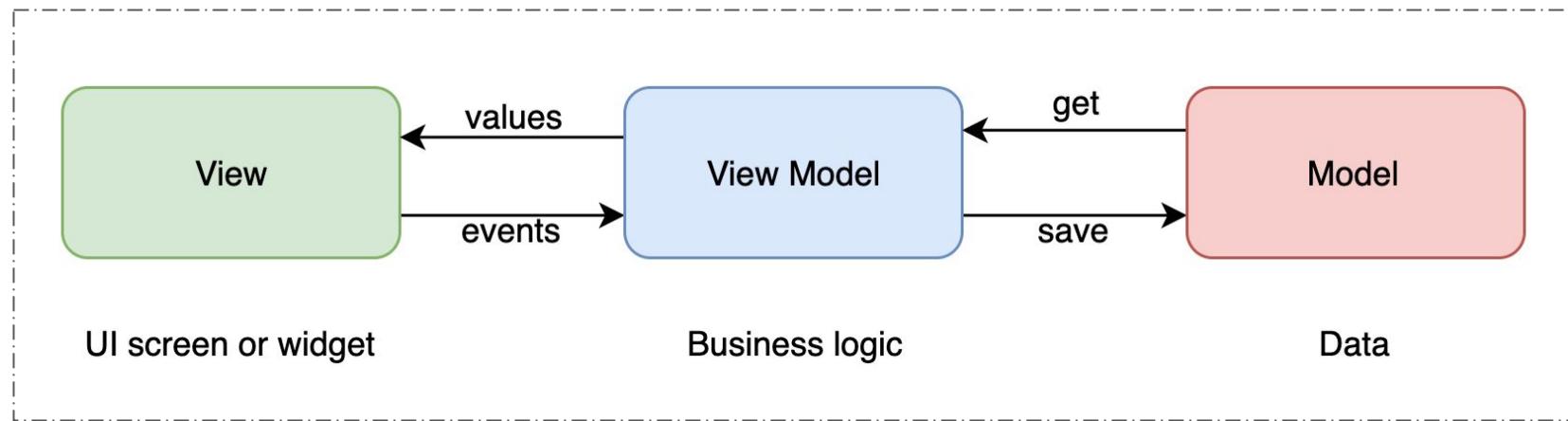
Column(
    mainAxisAlignment: MainAxisAlignment.spaceAround,
    children: [
        Text('$name'),
        TextField(
            controller: myController,
            decoration: InputDecoration(hintText: 'Nom')),
        ElevatedButton(
            onPressed: () => notify(),
            child: Text('Se connecter'))
    ]
)
```



Approches de State management

- StatelessWidget :
 - ◆ Toute l'arborescence des widgets est reconstruite à chaque fois que l'état d'un seul widget change ;
 - ◆ Il ne permet pas de séparer la logique métier des composants de l'interface graphique ;
- Solutions ?
 - ◆ Architecture : séparer le métier de la présentation via MVVM ;
 - ◆ Principaux patterns : BLoC et Provider ;
 - ◆ Autres approches : GetX, Redux, Binder, RxDart, Riverpod, etc.

Architecture MVVM



BLoC

- Business Logic Components
- Design pattern créé par Google en 2018
- Concepts de base :
 - ◆ Stream : flux continu ou succession de données ;
 - ◆ Event : élément d'un stream ;
 - ◆ Cubit < Bloc : types spéciaux de stream.
- Installation :
 - ◆ flutter pub add bloc
 - ◆ flutter pub add flutter_bloc
 - ◆ flutter pub get

BLoC

Principales classes :

Classe	Description
BlocObserver	Observation de tous les changements d'état dans l'application.
BlocProvider	Reconstruction de widget lors du changement d'état du Bloc.
BlocBuilder	Création et fourniture du Bloc aux éléments enfants.
BlocListener	Réponse aux changements d'état dans le bloc. Appelé une seule fois.
BlocConsumer	Inclusion d'un builder et d'un listener pour réagir aux changements d'état.

Exemple (1/3)

```
class NameObserver extends BlocObserver {  
    @override  
    void onChange(BlocBase<dynamic> bloc, Change<dynamic> change) {  
        super.onChange(bloc, change);  
    }  
  
    class NameCubit extends Cubit<String> {  
        NameCubit() : super('');  
        void changeName(String newName) => emit('Bienvenue $newName');  
    }  
}
```

Exemple (2/3)

```
class NamePage extends StatelessWidget {
    const NamePage({super.key});

    @override
    Widget build(BuildContext context) {
        return BlocProvider(
            create: (_) => NameCubit(),
            child: NameView()
        );
    }
}
```

Exemple (3/3)

```
class NameView extends StatelessWidget {
    final myController = TextEditingController();

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Column(
                mainAxisAlignment: MainAxisAlignment.spaceAround,
                children: [
                    BlocBuilder<NameCubit, String>(builder: (context, state) {
                        return Text('$state');
                    }),
                    TextField(controller: myController,
                        decoration: InputDecoration(hintText: 'Nom')),
                    ElevatedButton(child: Text('Se connecter'),
                        onPressed: () => context.read<NameCubit>().changeName(myController.text))])
    );
}
```

Provider

- Amélioration de BLoC ;
- Package introduit en 2020 ;
- Concepts de base :
 - ◆ Event : données subissant un changement d'état ;
 - ◆ Pas de notions de Bloc ou de Cubit.
- Installation :
 - ◆ flutter pub add provider
 - ◆ flutter pub get

Provider

Principales classes :

Classe	Description
ChangeNotifier	Notification de modification des données de la classe qui l'hérite.
ChangeNotifierProvider	Un widget parent contenant la référence d'un ChangeNotifier.
Consumer	Reconstruction de widget lors de notification par ChangeNotifier.

Exemple (1/2)

```
runApp(ChangeNotifierProvider(  
    create: (_ ) => Person(),  
    child: MyApp(),  
)  
)  
  
class Person extends ChangeNotifier {  
    String name = '';  
  
    void changeName(String newName) {  
        name = 'Bienvenue $newName';  
        notifyListeners();  
    }  
}
```

Exemple (2/2)

```
Column(  
    mainAxisAlignment: MainAxisAlignment.spaceAround,  
    children: [  
        Text('${Provider.of<Person>(context).name}'),  
        TextField(  
            controller: myController,  
            decoration: InputDecoration(hintText: 'Nom')  
        ),  
        ElevatedButton(  
            child: Text('Se connecter'),  
            onPressed: () => Provider.of<Person>(context, listen: false)  
                .changeName(myController.text))  
    ]  
)
```

Navigation

Version	Techniques	Portée	Récente ?
1.0	Push/pop & routes nommées, compatibles entre elles	Petites applications	Non mais supportée
2.0	Déclaration de routes & mise à jour de state	Applications d'entreprise	Oui

Navigation 1.0

→ Push & pop :

```
Navigator.push(  
    context,  
    MaterialPageRoute(builder: (context) => LogoutScreen())  
)  
Navigator.pop(context)
```

→ Routes nommées :

```
initialRoute: '/',  
routes: {  
    '/login': (context) => LoginScreen(),  
    '/logout': (context) => LogoutScreen()  
}  
Navigator.pushNamed(context, '/login ou logout')
```

Navigation 2.0

```
int routeID = 0;

onScreenChange(newRouteID) {
    setState(() {
        routeID = newRouteID;
    });
}

Navigator(
    pages: [
        if (routeID == 0 || routeID == 1) MaterialPage(child: LoginScreen(onScreenChange)),
        if (routeID == 2) MaterialPage(child: LogoutScreen(onScreenChange))
    ],
    onPopPage: (
        (route, result) {
            return route.didPop(result);
    }
)
)
```

Activité 20

Corrigez l'erreur du code ci-dessous :

```
bool isChecked = false;

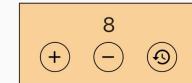
void notify(bool? value) {
    setState(() {
        isChecked = value!;
    });
}

Column(
    mainAxisAlignment: MainAxisAlignment.min,
    crossAxisAlignment:
        CrossAxisAlignmentAlignment.start,
    children: [
        ...
    ]
)
```

```
Row(mainAxisSize: MainAxisSize.min, children: [
    Checkbox(
        value: isChecked,
        onChanged: (bool? value) {
            notify(value);
        }),
    Text('Java')
])
),
Row(mainAxisSize: MainAxisSize.min, children: [
    Checkbox(
        value: isChecked,
        onChanged: (bool? value) {
            notify(value);
       }),
    Text('Python')
])
)
```

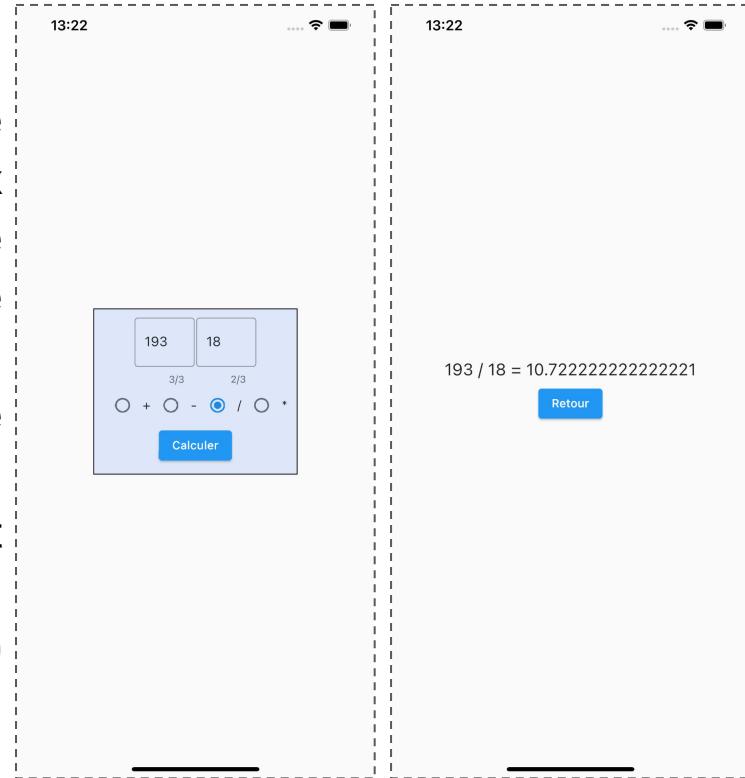
Activité 21

Réalisez une interface similaire permettant d'incrémenter, de décrémenter ou de réinitialiser la valeur d'un Text. Le state management doit être fait par trois approches : setState, BLoC et Provider.



Activité 22

Réalisez une application permettant de faire les quatre opérations arithmétiques de deux nombres réels. Les valeurs doivent être fournies sur un écran puis le calcul et le résultat sont effectués sur un autre écran. Outre la gestion de division sur zéro, lorsque l'une des valeurs n'est pas renseignée, l'application doit ne pas procéder au calcul et afficher un Toast d'avertissement. La navigation doit être faite en deux versions : 1.0 et 2.0.



Chapitre 6 : persistance des données via SQLite

```
DatabaseHelper._privateConstructor();
static final DatabaseHelper instance = DatabaseHelper._privateConstructor();

static Database? _database;
Future<Database> get database async => _database ??= await _initDatabase();

Future<Database> _initDatabase() async {
    Directory documentsDirectory = await getApplicationDocumentsDirectory();
    String path = join(documentsDirectory.path, 'students.db');
    return await openDatabase(
        path,
        version: 1,
        onCreate: _onCreate,
    );
}

Future<List<Student>> getStudents() async {
    Database db = await instance.database;
    var students = await db.query('students', orderBy: 'name');
    List<Student> studentList = students.isNotEmpty
        ? students.map((c) => Student.fromMap(c)).toList()
        : [];
    return studentList;
}
```

Acquis techniques

À la fin de ce chapitre, vous serez capable de :

- Installer les packages nécessaires pour SQLite ;
- Comprendre le principe du pattern Data Access Object (DAO) ;
- Créer l'entité modèle qui représente la classe DAO ;
- Réaliser les méthodes correspondant aux opérations CRUD ;
- Communiquer avec la base de données à travers une ListView.

SQLite ?

- SGBDR embarqué, open-source et multiplateforme
- Écrit en C en 2000 par Dr Richard Hipp
- Accessible par le langage SQL
- Conforme aux propriétés ACID
- Léger et très répandu dans les systèmes embarqués
- Plus d'informations : <https://sqlite.org>

- flutter create sqlite_demo
- flutter pub add sqflite
- flutter pub add path_provider

Classe DAO

```
class Student {  
    final int? id;  
    final String name;  
  
    Student({this.id, required this.name});  
  
    factory Student.fromMap(Map<String, dynamic> json) => new Student(  
        id: json['id'],  
        name: json['name'],  
    );  
  
    Map<String, dynamic> toMap() {  
        return {  
            'id': id,  
            'name': name,  
        };  
    }  
}
```

Couche de persistance

```
DatabaseHelper._privateConstructor();
static final DatabaseHelper instance = DatabaseHelper._privateConstructor();

static Database? _database;
Future<Database> get database async => _database ??= await _initDatabase();

Future<Database> _initDatabase() async {
    Directory documentsDirectory = await getApplicationDocumentsDirectory();
    String path = join(documentsDirectory.path, 'students.db');
    return await openDatabase(path, version: 1, onCreate: _onCreate);
}

Future _onCreate(Database db, int version) async {
    await db.execute('''CREATE TABLE students(id INTEGER PRIMARY KEY, name TEXT)''' );
}
```

Create, Read

```
Future<int> create(Student student) async {
    Database db = await instance.database;
    return await db.insert('students', student.toMap());
}

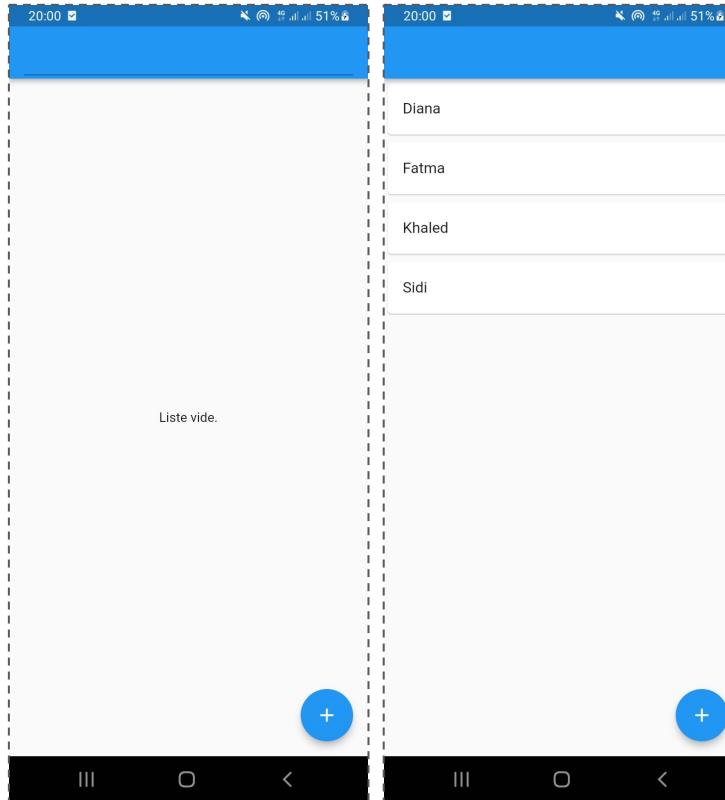
Future<List<Student>> read() async {
    Database db = await instance.database;
    var students = await db.query(
        'students',
        orderBy: 'name',
    );
    List<Student> studentList = students.isNotEmpty
        ? students.map((c) => Student.fromMap(c)).toList()
        : [];
    return studentList;
}
```

Update, Delete

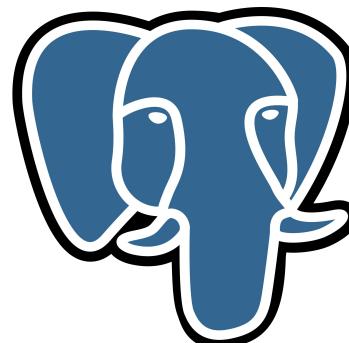
```
Future<int> update(Student student) async {
    Database db = await instance.database;
    return await db.update(
        'students',
        student.toMap(),
        where: "id = ?",
        whereArgs: [student.id],
    );
}

Future<int> delete(int id) async {
    Database db = await instance.database;
    return await db.delete(
        'students',
        where: 'id = ?',
        whereArgs: [id],
    );
}
```

Illustration



Chapitre 7 : back-end, Django et Firebase



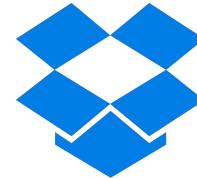
Acquis techniques

À la fin de ce chapitre, vous serez capable de :

- Créer un environnement virtuel Python ;
- Installer et configurer Django ;
- Comprendre le principe de Model-View-Template ;
- Saisir les rôles des éléments de l'arborescence d'un projet Django ;
- Installer et configurer SQLite et PostgreSQL ;
- Réaliser, à travers Django REST, une API munie de CRUD ;
- Tester GET, POST, PUT et DELETE via Postman ;
- Créer une application Flutter qui s'interface avec Django REST ;
- Créer et configurer un projet Firebase pour l'authentification SMS ;
- Créer une application Flutter qui s'interface avec Firebase Authentication.

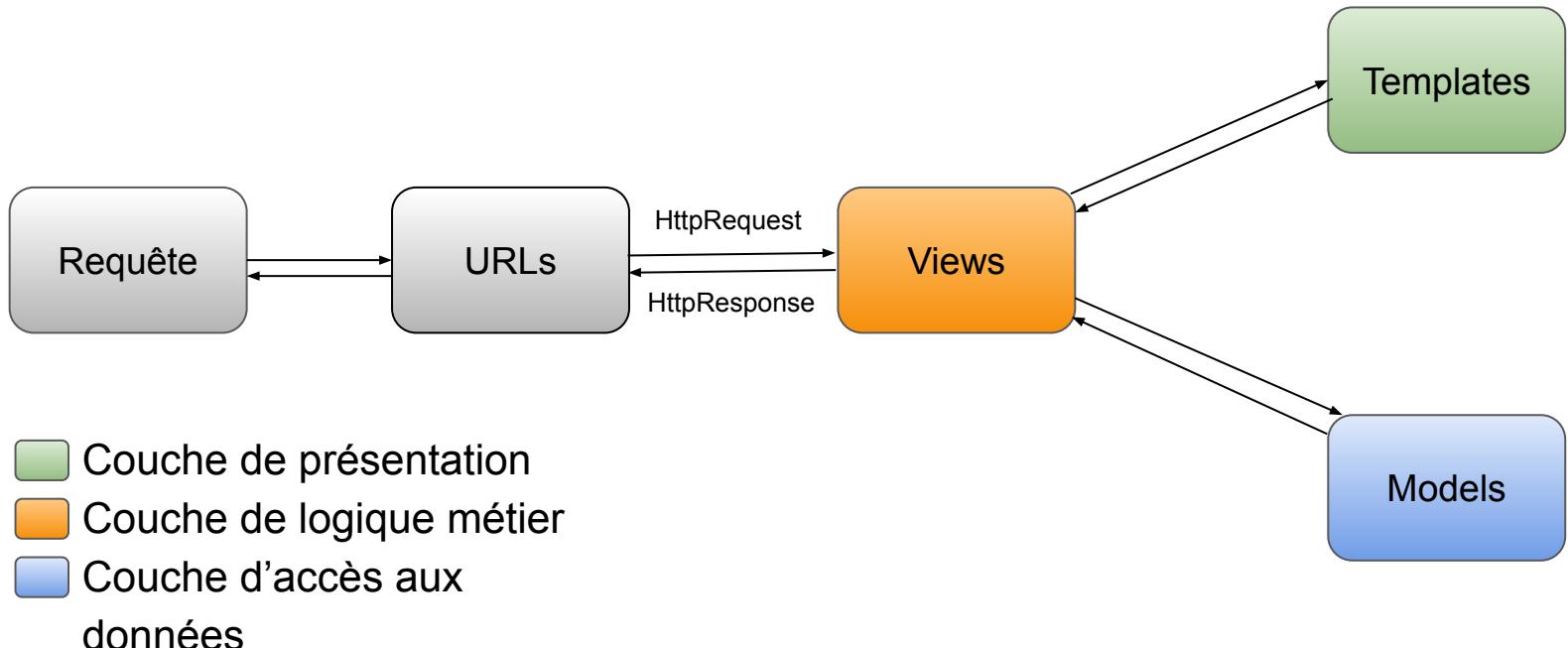
Django ?

- Framework open-source, créé en 2005, écrit en Python
- Utilisé principalement dans la partie backend malgré le DTL*
- De type “Batteries included”: autosuffisant, muni du nécessaire
- Design pattern : MVT
- Références :

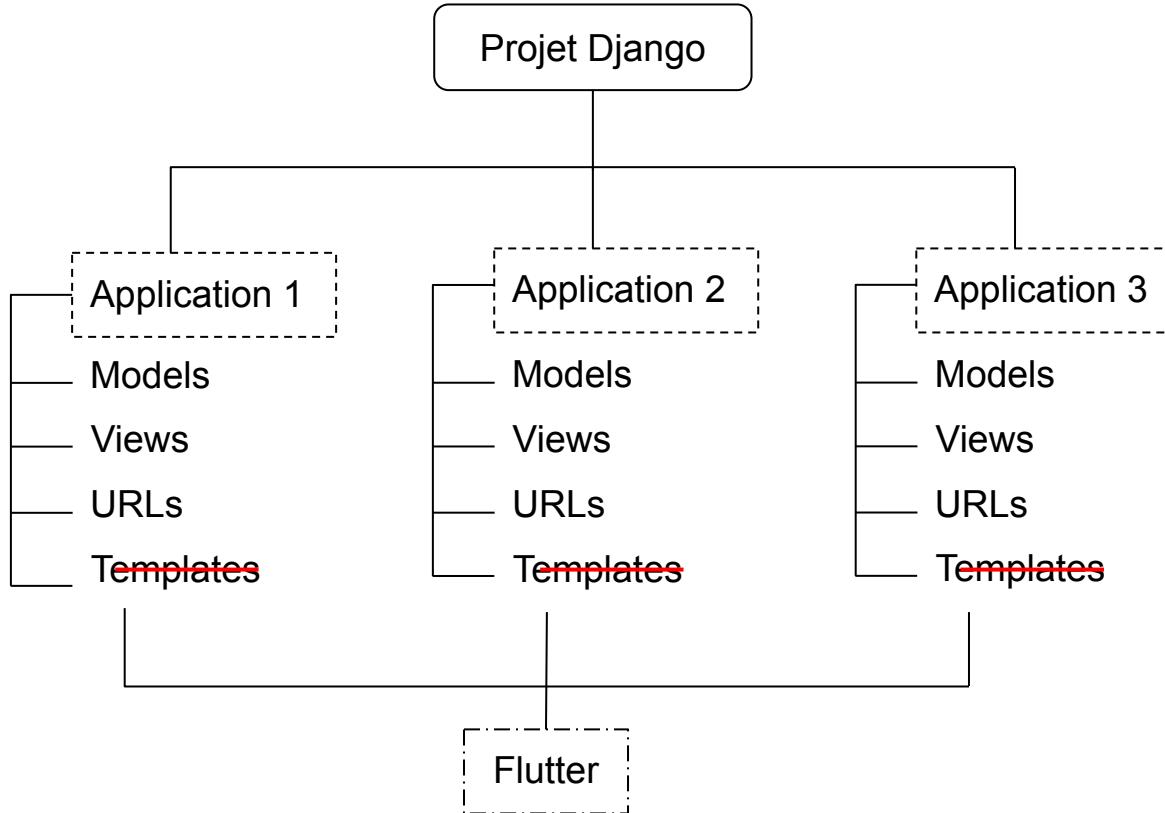


* Django Template Language

Model-View-Template



Structure de projet



Premiers pas

→ **Python** : brew install python, python3 --version

→ **Environnement virtuel** :

```
pip3 install pipenv
```

```
cd Documents
```

```
python3 -m venv django_env
```

```
source django_env/bin/activate
```

→ **Django** :

```
pip3 install django
```

```
cd Desktop
```

```
django-admin startproject first_project
```

```
cd first_project
```

```
python3 manage.py runserver 9000
```

→ Aller à <http://localhost:9000>

Intégration de l'environnement virtuel à VSC

- cd first_project
- code .
- Command + Shift + P
- Python interpreter
- Select interpreter
- Enter interpreter path : /Users/ahmed/Documents/django_env/bin/python
- python3 manage.py runserver 9000
- Aller à <http://localhost:9000>
- Installer l'extension Python

Exemple

urls.py :

```
from django.urls import path
from django.http import HttpResponse

def hello(request):
    return HttpResponse('Hello World!')

urlpatterns = [
    path('', hello)
]
```

```
python manage.py runserver
```

Exemple d'application

- Créer une nouvelle application :

```
python manage.py startapp first_app
```

- Ajouter le nom de l'application dans INSTALLED_APPS

- Première view :

```
from django.http import HttpResponse
def first_view(request):
    return HttpResponse('Hello World!')
```

- first_project/urls.py :

```
from django.urls import include
urlpatterns = [
    path('first_app/', include('first_app.urls'))
]
```

Exemple d'application

→ `first_app/urls.py` :

```
from django.urls import path
from . import views
urlpatterns = [
    path('test', views.first_view)
]
```

→ Accéder à http://localhost:8000/first_app/test

→ `cd first_app/`

→ `mkdir templates && nano templates/test.html`
`<p style="color:blue;font-size:50px">Hello world!</p>`

→ `views.py` :

```
def first_view(request):
    return render(request, 'test.html')
```

Arborescence du projet

Élement	Description
migrations	Informations sur les tables de la base de données
admin.py	Gestion de l'interface d'administrateur
app.py	Fichier de configuration
models.py	Gestion des données provenant de la base de données
tests.py	Tests unitaires
view.py	Gestion de requêtes http
settings.py	Fichier principal de configuration
urls.py	Fichier de configuration des liens
manage.py	Utilitaire de ligne de commande de Django

Django REST

- Créé en 2014 et maintenu par [Encode OSS Ltd](#)
- Un toolkit basé sur Django, dédié à la création des APIs
- Majoritairement connu pour les APIs REST
- Mais aussi valable pour les APIs SOAP à travers le toolkit Spyne
- Possibilité de génération de documentation d'API
- Déploiement facile du back-end sur [Heroku](#)
- Installation :
 - pip3 install djangorestframework
 - git clone <https://github.com/encode/django-rest-framework>
- 'rest_framework' -> INSTALLED_APPS
- Plus d'informations : <https://www.django-rest-framework.org/>

Exemple

```
→ mkdir first_api && cd first_api
→ touch views.py urls.py
→ views.py :
    from rest_framework.response import Response
    from rest_framework.decorators import api_view

    @api_view(['GET'])
    def getData(request):
        student = {'id': 'C04537', 'enrolled': '2010-03-16'}
        return Response(student)
→ first_api/urls.py : urlpatterns = [path('', views.getData)]
→ first_project/urls.py : urlpatterns = [path('', include('first_api.urls'))]
```

Modèle

- python manage.py startapp first_app_api
- 'first_app_api' -> INSTALLED_APPS

- models.py :

```
from django.db import models
```

```
class Student(models.Model):  
    studentID = models.CharField(max_length=6)  
    enrolled = models.DateTimeField(auto_now=True)
```

Serialisation

```
→ cd first_app_api && touch serializers.py

→ serializers.py:
from rest_framework import serializers
from first_app_api.models import Student

class StudentSerializer(serializers.ModelSerializer):
    class Meta:
        model = Student
        fields = '__all__'
```

Create, Read

views.py:

```
from first_app_api.models import Student
from .serializers import StudentSerializer
from rest_framework.response import Response
from rest_framework.decorators import api_view

@api_view(['POST'])
def addStudent(request):
    serializer = StudentSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
    return Response(serializer.data)

@api_view(['GET'])
def getStudents(request):
    students = Student.objects.all()
    serializer = StudentSerializer(students, many=True)
    return Response(serializer.data)
```

Update, Delete

```
@api_view(['PUT'])
def updateStudent(request, pk):
    student = Student.objects.get(id=pk)
    serializer = StudentSerializer(instance=student, data=request.data)
    if serializer.is_valid():
        serializer.save()
    return Response(serializer.data)

@api_view(['DELETE'])
def deleteStudent(request, pk):
    student = Student.objects.get(id=pk)
    student.delete()
    return Response('Suppression réussie!')
```

URLs

→ **first_app_api/urls.py:**

```
from django.urls import path
from . import views

urlpatterns = [
    path('create/', views.addStudent),
    path('read', views.getStudents),
    path('update/<str:pk>/', views.updateStudent),
    path('delete/<str:pk>/', views.deleteStudent)
]
```

→ **first_project/urls.py :**

```
path('', include('first_app_api.urls'))
```

Manipulation de SQLite

- python manage.py shell
- >>>from first_app_api.models import Student
 Student.objects.create(studentID="C38124")
 Student.objects.create(studentID="C16193")
 Student.objects.create(studentID="C44001")
 students=Student.objects.all()
 print(students)
 exit()
- sqlite3 first_project/db.sqlite3
 >.tables
 >select * from first_app_api_student;
 >.quit
- <https://sqlitebrowser.org/dl/>

Installation de PostgreSQL

```
→ brew install postgresql
→ brew services start postgresql
→ brew services stop postgresql
→ pg_ctl -D /usr/local/var/postgres status
→ sudo -u postgres pg_ctl -D /Library/PostgreSQL/14/data status

→ export PATH="$PATH:/Library/PostgreSQL/14/bin"

→ postgres --version
→ psql postgres
→ \q

→ brew install --cask pgadmin4
```

Configuration de PostgreSQL

```
→ pip3 install psycopg2
→ Ouvrir pgAdmin
→ Créer une base de données nommée first_db
→ 'default': {
    'ENGINE': 'django.db.backends.postgresql',
    'NAME': 'first_db',
    'USER': 'postgres',
    'PASSWORD': 'root',
    'HOST': 'localhost',
    'PORT': '5432'
}
```

Déploiement

- python manage.py makemigrations
- python manage.py migrate
- python manage.py runserver
- <http://localhost:8000/read/>

Get Students

Get Students

GET /

HTTP 200 OK

Allow: OPTIONS, GET

Content-Type: application/json

Vary: Accept

```
[  
    {  
        "id": 1,  
        "studentID": "C01003",  
        "enrolled": "2022-09-25T15:26:50.707380Z"  
    },  
    {  
        "id": 2,  
        "studentID": "C15824",  
        "enrolled": "2022-09-25T15:27:35.712419Z"  
    }  
]
```

Viewsets et Router

→ `viewsets.py` :

```
from rest_framework import viewsets
from . import models, serializers

class StudentViewSet(viewsets.ModelViewSet):
    queryset = models.Student.objects.all()
    serializer_class = serializers.StudentSerializer
```

→ `router.py` :

```
from rest_framework import routers
from second_app_api.viewsets import StudentViewSet

router = routers.DefaultRouter()
router.register('student', StudentViewSet)
```

→ `first_project/urls.py` :

```
from second_app_api.router import router
urlpatterns = [path('', include(router.urls))]
```

Postman

- Application de test des APIs, créée en 2012
- Développée en JavaScript via Electron
- Propriétaire mais disponible en version gratuite
- Possibilité de partage de l'espace de travail
- Plus d'informations : <https://www.postman.com>
- Créer un compte Postman via Google
- Soumettez des exemples de requêtes suivantes :
 - GET
 - POST
 - PUT
 - DELETE

Connecter Flutter au backend Django (1/3)

```
→ flutter create student_frontend
→ flutter pub add http
→ <uses-permission android:name="android.permission.INTERNET"/>

→ class Student {
    final String? studentID;

    Student({this.studentID});

    factory Student.fromJson(Map<String, dynamic> json) {
        return Student(studentID: json['studentID']);
    }

    Map<String, dynamic> toJson() => {'studentID': studentID};
}
```

Connecter Flutter au backend Django (2/3)

```
Future<List<Student>> getStudentList() async {
    final response = await http
        .get(Uri(path: "http://10.0.2.2:8000/"));

    final items = json.decode(response.body)
        .cast<Map<String, dynamic>>();
    List<Student> students = items.map<Student>((json) {
        return Student.fromJson(json);
    }).toList();

    return students;
}
```

Connecter Flutter au backend Django (3/3)

```
FutureBuilder<List<Student>>(
    future: students,
    builder: (BuildContext context, AsyncSnapshot snapshot) {
        return ListView.builder(
            itemCount: snapshot.data.length,
            itemBuilder: (BuildContext context, int index) {
                var data = snapshot.data[index];
                return Card(
                    child: ListTile(
                        leading: Icon(Icons.person),
                        title: Text(data.studentID),
                    ),
                );
            },
        );
    },
)
```

Firebase ?

- Plateforme de type Backend as a Service (BaaS)
- Créeé en 2011 et racheté par Google en 2014
- Principaux services :
 - Hosting
 - Firestore
 - Analytics
 - Authentification
 - Realtime Database
 - Cloud Messaging
 - Machine Learning
- Plus d'informations : <https://firebase.google.com>
- Dans ce chapitre, on s'intéresse au service d'authentification par SMS.

Création et configuration du projet

- <https://firebase.google.com> -> Add project
- Désactiver “Google Analytics for this project” -> Create project -> Continue
- Cliquer sur le projet et choisir le type d'application Android
- flutter create firebase_auth_demo
- flutter pub add firebase_core
- flutter pub add firebase_auth
- Dans la clause android/defaultConfig du android/app/build.gradle, copier applicationId et coller-le dans le champ Android package name
- Cliquer sur Register app
- Télécharger le fichier google-services.json et copier-le dans android/app
- Copier les valeurs indiquées respectivement dans android/build.gradle et android/app/build.gradle
- Cliquer sur Continue to the console
- Cliquer sur l'application puis sur Settings

Génération des empreintes du certificat

- Générer les signatures SHA1 et SHA256 à travers Keytool pour authentifier votre application au niveau de Firebase :
 - Mac : keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
 - Windows : keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
- Dans la section SHA certificate fingerprints, ajouter les signatures générées
- Plus d'informations : <https://developers.google.com/android/guides/client-auth>

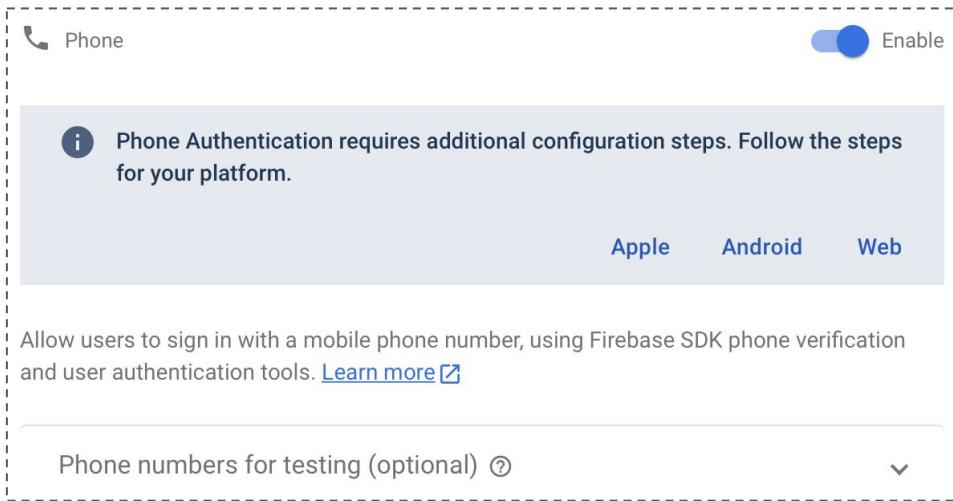
Génération des empreintes du certificat

```
ahmed@macbook-air ~ % keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
Alias name: androiddebugkey
Creation date: Jun 12, 2021
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=US, O=Android, CN=Android Debug
Issuer: C=US, O=Android, CN=Android Debug
Serial number: 1
Valid from: Sat Jun 12 07:50:10 GMT 2021 until: Mon Jun 05 07:50:10 GMT 2051
Certificate fingerprints:
    SHA1: D9:01:BA:61:04:19:5E:6E:6C:41:F8:D5:8C:24:34:4C:75:51:8F:C7
    SHA256: 79:A3:DE:2D:09:43:9A:5B:A6:BE:24:B1:04:00:79:09:61:9A:34:42:08:87:3E:41:0D:01:78:BA:27:47:E6:03
Signature algorithm name: SHA1withRSA (weak)
Subject Public Key Algorithm: 2048-bit RSA key
Version: 1

Warning:
The certificate uses the SHA1withRSA signature algorithm which is considered a security risk. This algorithm will be disabled in a future update.
ahmed@macbook-air ~ %
```

Mode d'authentification

- Aller à l'onglet Authentification dans la section Build
- Sélectionner Phone, activer le mode puis cliquer sur Save



- Aller à l'onglet Users

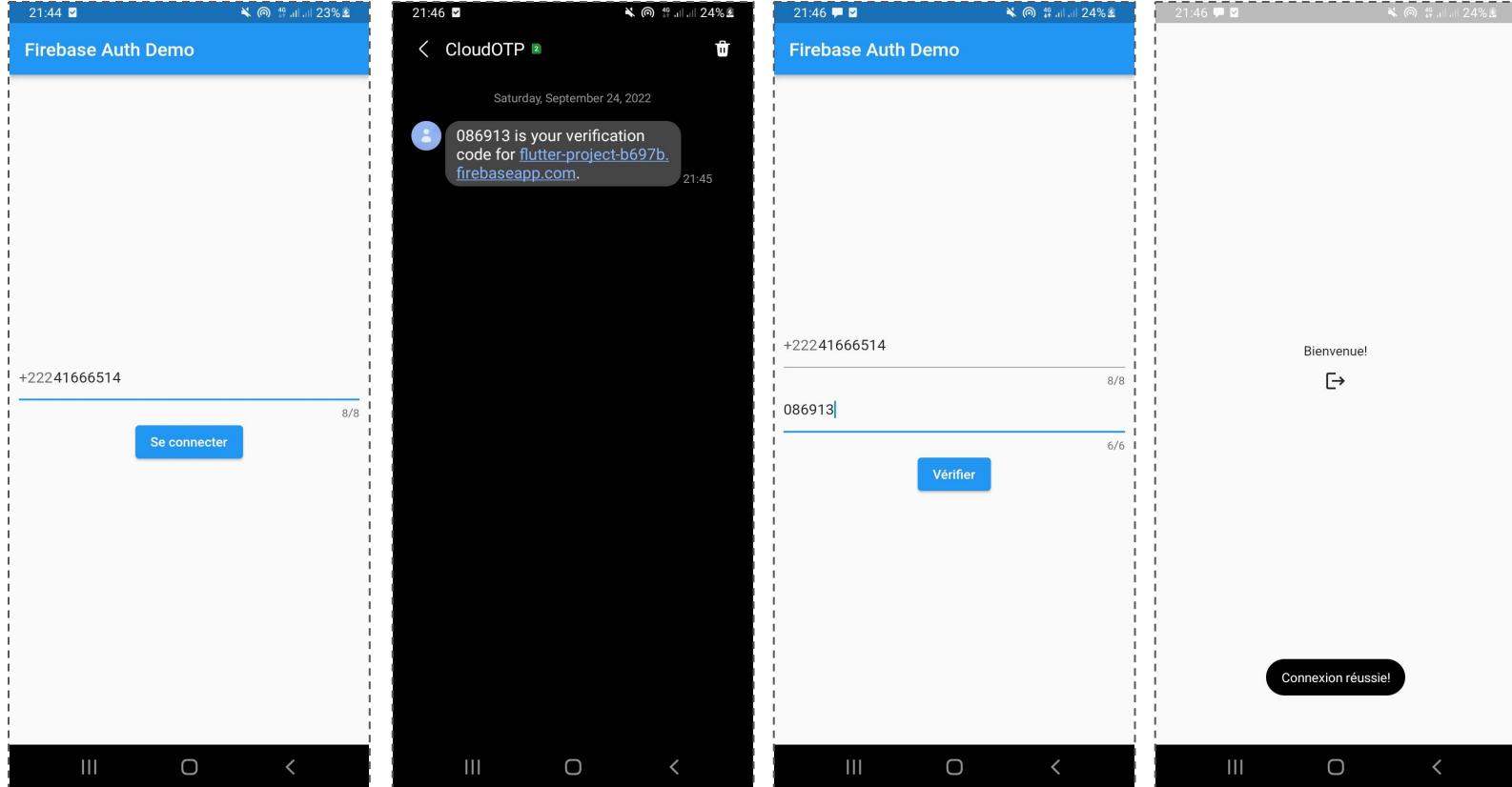
Envoi de SMS

```
await Firebase.initializeApp();
FirebaseAuth.instance.verifyPhoneNumber(phoneNumber,
    verificationCompleted: (PhoneAuthCredential credential) async {
        // Connexion réussie
        await auth.signInWithCredential(credential).then((value) {});
    },
    // Une erreur s'est produite
    // Quota de SMS dépassé, format du numéro de téléphone invalide, ...
    verificationFailed: (FirebaseAuthException e) {},
    // SMS envoyé
    codeSent: (String verificationId, int? resendToken) {},
    // Délai dépassé
    codeAutoRetrievalTimeout: (String verificationId) {},
);
}
```

Vérification du code OTP

```
PhoneAuthCredential credential = PhoneAuthProvider  
        .credential(verificationId, smsCode);  
  
await auth.signInWithCredential(credential).then(  
    (value) {  
        user = FirebaseAuth.instance.currentUser;  
    },  
) .whenComplete(  
    () {  
        if (user != null) // OTP correct  
        else // OTP incorrect  
    },  
) ;
```

Illustration



Historique des connexions

Revoir l'onglet Users, constat ?

A screenshot of a user management interface. At the top, there is a search bar with a magnifying glass icon and the placeholder text "Search by email address, phone number or user UID". To the right of the search bar are three buttons: "Add user", a circular refresh icon, and a vertical ellipsis icon. Below the search bar is a table header with columns: "Identifier", "Providers", "Created", "Signed in", and "User UID". The "Created" column has a downward arrow indicating it is sortable. The table contains one row of data. The "Identifier" column shows "+22241666514". The "Providers" column shows a telephone receiver icon. The "Created" column shows "25 Sept 2022". The "Signed in" column shows "25 Sept 2022". The "User UID" column shows "KGQ13czwUzVVCgxmAZs1Lmz2i...". At the bottom of the table, there are pagination controls: "Rows per page" set to "50" with a dropdown arrow, "1 – 1 of 1", and navigation arrows.

Identifier	Providers	Created	Signed in	User UID
+22241666514	📞	25 Sept 2022	25 Sept 2022	KGQ13czwUzVVCgxmAZs1Lmz2i...

Chapitre 8 : publication sur le Store

The screenshot shows the Google Play Console interface. On the left, there's a sidebar with various navigation options: All apps (selected), Inbox (8 notifications), Policy status, Users and permissions, Order management, Download reports, Account details, Developer page, Associated developer accounts, Activity log, and Setup. The main area is titled "All apps" and displays a message: "View all of the apps and games that you have access to in your developer account". A "Create app" button is located in the top right. Below this, there's a section for "Pinned apps" with a link to pin apps. The main content area shows "3 apps" listed in a table:

App	Installed audience	App status	Update status	Last updated	Actions
Application AMDRS com.infectdistrack	[progress bar]	[progress bar]		28 May 2021	Edit →
MathOBac maurimath.mathobac	[progress bar]	Production		29 Jun 2020	Edit →
Word Researcher com.wordresearcher	[progress bar]	Production		15 Apr 2020	Edit →

At the bottom, there are pagination controls: "Show rows 10 ▾ 1 - 3 of 3" and navigation arrows (< < > >).

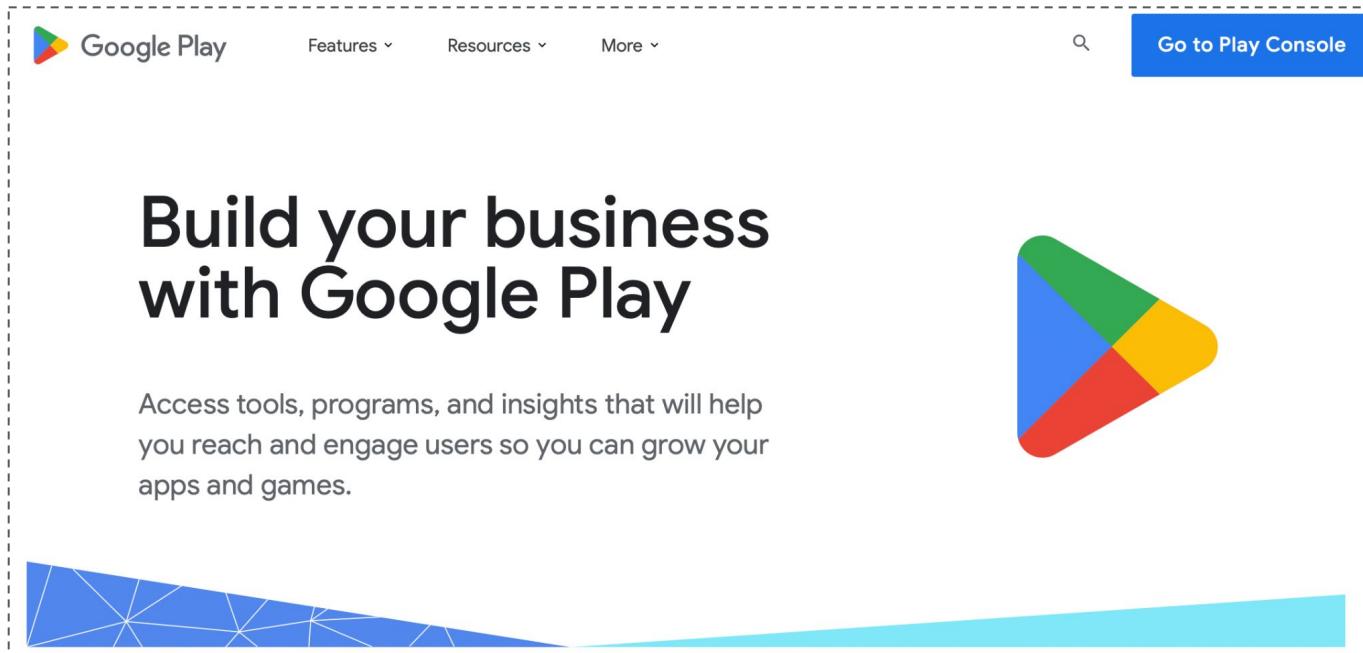
Acquis techniques

À la fin de ce chapitre, vous serez capable de :

- Créer un compte développeur au niveau de Google Play ;
- Comprendre les principaux éléments du tableau de bord ;
- Soumettre une application à Google Play Store ;
 - Signer l'application en mode release
 - Renseigner les formulaires
 - Ajouter les testeurs
 - Rectifier et envoyer le release
 - Ajouter l'icône et les captures d'écran
 - Demander à Google de faire une revue de l'application
- Faire des mises à jour de l'application.

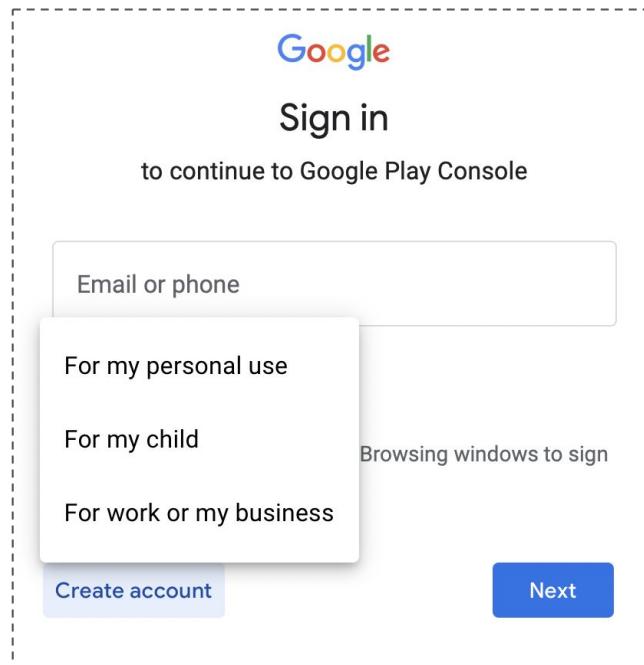
Création de compte développeur (1/5)

Aller à la Google Play Console : <https://play.google.com/console>



Création de compte développeur (2/5)

Accéder à la Play Console -> Créer un compte -> Usage personnel



Création de compte développeur (3/5)

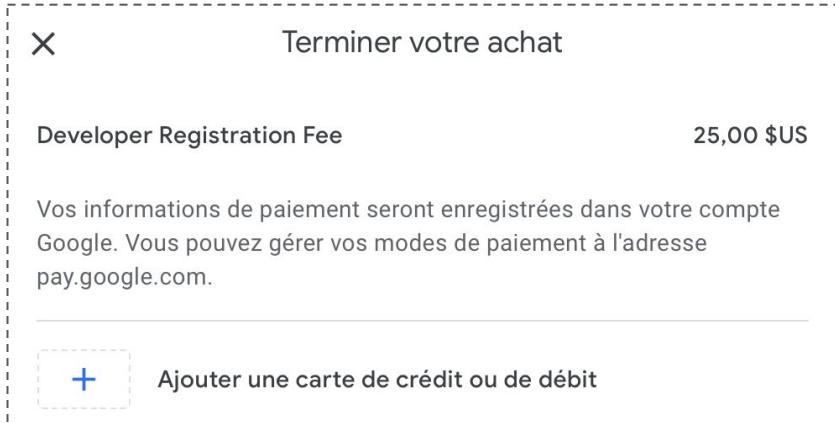
- Cliquez sur Se connecter à un compte existant
- Activez la validation en deux étapes si demandé
- Sélectionnez le type de compte : Pour vous
- Décrivez votre expérience en Android et Play Console dans la première zone de texte
- Mentionnez votre site web
- Remplissez le second formulaire : nom du compte de développeur, nom, e-mail, pays, numéro de téléphone et langue de communication
- Validez votre numéro de téléphone et votre e-mail par code OTP puis cliquer sur Suivant

Création de compte développeur (4/5)

- Sélectionner le nombre d'applications que vous prévoyez de publier sur Google Play dans les 12 prochains mois
- Préciser si vous comptez générer des revenus à partir des applications que vous publiez sur Google Play
- Sélectionner les catégories des applications prévues ;
- Confirmer que vous avez au moins 18 ans et que vous avez lu et accepté les conditions d'utilisation de la Google Play Console
- Cliquer sur Créer un compte et Payer

Création de compte développeur (5/5)

- Les frais de création de compte développeur sont à 25 dollars



- Cliquer sur Ajouter une carte de paiement
- Enregistrer la carte et procéder au paiement
- Vous serez redirigé au tableau de bord de Google Play Console

Création d'application

- Cliquer sur Create app

Create app

App details

App name This is how your app will appear on Google Play 0 / 30

Default language English (United Kingdom) – en-GB

App or game You can change this later in Store settings

App

Game

Free or paid You can edit this later on the paid app page

Free

- Saisir le nom de l'application et préciser son type
- Préciser si l'application est gratuite ou payante
- Accepter les conditions puis cliquer sur Create app

Ajout des testeurs

- Start testing now -> View tasks -> Select testers -> Create email list
- Saisir le nom de la liste ainsi que les emails des testeurs séparés par virgule
- Enter -> Save changes -> Create
- Fournir l'email de communication avec les testeurs

Testers

Up to 100 testers can join your internal tests. You can choose more than 100 testers, but only the first 100 to join will be successful.

Testers	<input checked="" type="checkbox"/> List name	Users	
	<input checked="" type="checkbox"/> Flutter testers	3	→

[Create email list](#)

Feedback URL or email address
amed.mohameden@gmail.com

Let testers know how to provide you with feedback

24 / 512

Envoi d'application pour test

- Start testing now -> View tasks -> Create a new release
- Signer l'application en mode release
- Générer l'application au format App Bundle :
`flutter clean`
`flutter build appbundle --release`
- Charger le fichier : build/app/outputs/bundle/release/app-release.aab
- Save -> Review release -> Start roll-out to Internal testing

Renseignement des formulaires

- Set up your app -> Set privacy policy -> Politique de confidentialité
- App access -> All or some functionality is restricted -> Add new instructions
- Ads -> No, my app does not contain ads -> Save
- Content ratings -> Start questionnaire -> renseigner le formulaire
- Target audience -> renseigner le formulaire
- News apps -> No -> Save
- COVID-19 -> My app is not a publicly available COVID-19 contact tracing
- Data safety -> renseigner le formulaire
- Select an app category and provide contact details -> renseigner le formulaire
- Set up your Store Listing -> description et graphics

Publication d'application

- Select countries and regions
- Create a new release
- Upload an app bundle or APK
- Save -> Review release
- Start roll out to Production

Principaux états (1/2)

État	Signification
Draft	L'application n'est pas encore soumise à Google pour la revue
Internal testing	L'application n'est disponible que pour les testeurs internes via un lien
Closed testing	L'application est visible sur Google Play, mais seuls les testeurs sélectionnés peuvent l'installer
Open testing	L'application est visible sur Google Play et tout utilisateur peut devenir testeur. Vous pouvez limiter le nombre de testeurs
In review	L'application est en cours de revue
Production	L'application est disponible pour les utilisateurs de Google Play dans les pays mentionnés

Principaux états (2/2)

État	Signification
Unpublished	La publication de l'application est annulée par le développeur. L'application et ses mises à jour demeurent disponibles pour les utilisateurs existants
Removed	L'application est supprimée du Store par Google à cause d'une violation du règlement de Google Play. Plusieurs suppressions peuvent entraîner une suspension. Solution : soumettre une mise à jour conforme au règlement de Google Play
Suspended	L'application est supprimée du Store par Google à cause d'un comportement malveillant de l'application. Plusieurs suspensions peuvent entraîner la désactivation du compte développeur. Solution : contester la suspension de l'application sur : https://support.google.com/googleplay/android-developer/contact/protectappeals

Mini-projet

Description du mini-projet

- Exigences de front-end :
 - Progress Indicator, Stepper, Time-Date Picker et Hero
 - Expandable List, Search Bar, Slidable et Motion Tab Bar.
- Exigences de back-end :
 - Opérations CRUD
 - Gestion de permissions
 - Authentification à base de token
 - Filtrage des données (expressions régulières)
 - Test d'API via Postman
- Sujets : merci de consulter la fiche des sujets proposés.