Exercices : Environnement Mobile

Filière Informatique de Développement d'Applications (IDA) Date : 9 juillet

2025

Table des matières

1 Introduction

Ce document propose une série d'exercices pratiques pour consolider vos connaissances sur le module **Environnement Mobile**. Les exercices couvrent les plateformes mobiles (Android, iOS), les outils de développement, et la création d'applications avec Flutter. Chaque exercice est conçu pour être réalisable par des étudiants débutants ou intermédiaires, avec des objectifs clairs et des instructions détaillées.

2 Exercices Pratiques

2.1 Exercice 1 : Analyse des Plateformes Mobiles

Objectif : Comprendre les différences entre les plateformes Android et iOS, ainsi que les frameworks multiplateformes.

Instructions :

1. Rédigez un tableau comparatif (sur papier ou dans un document texte) des caractéristiques suivantes pour Android, iOS, et Flutter :
   - Système d'exploitation
   - Langages de programmation principaux
   - Environnement de développement (IDE)
   - Avantages
   - Inconvénients
2. Répondez aux questions suivantes :
   (a) Pourquoi Android souffre-t-il de fragmentation ? Donnez un exemple concret.
   (b) Pourquoi le développement iOS nécessite-t-il un appareil Apple ?
   (c) Dans quel cas utiliseriez-vous Flutter plutôt qu'un développement natif ?

Livrable : Un tableau comparatif et des réponses écrites aux questions.

—

2.2 Exercice 2 : Installation et Configuration de l'Environnement Flutter

Objectif : Configurer un environnement de développement pour Flutter et créer un projet de base.

Instructions :

1. Téléchargez et installez **Flutter SDK** depuis `https://flutter.dev`.
2. Installez un IDE (Android Studio ou Visual Studio Code) et configurez les plugins Flutter et Dart.
3. Créez un nouveau projet Flutter avec la commande : `flutter create exo2_application`.
4. Lancez l'émulateur Android ou iOS (ou connectez un appareil physique).
5. Exécutez le projet par défaut avec `flutter run` et vérifiez qu'il s'affiche correctement.
6. Prenez une capture d'écran de l'application par défaut fonctionnant sur l'émulateur ou l'appareil.

Livrable : Une capture d'écran de l'application par défaut et un court texte confirmant que l'environnement est configuré.

—

2.3 Exercice 3 : Création d'une Application Flutter avec Interaction

Objectif : Développer une application Flutter simple avec une interaction utilisateur.
Instructions :

1. Créez un nouveau projet Flutter nommé `exo3_compteur`.
2. Modifiez le fichier `lib/main.dart` pour créer une application avec :
   — Une barre d'application (AppBar) avec le titre "Compteur".
   — Un texte au centre affichant un nombre (initialisé à 0).
   — Deux boutons : un pour augmenter le compteur (+1) et un pour diminuer le compteur (-1).
3. Utilisez un `StatefulWidget` pour gérer l'état du compteur.
4. Voici un squelette de code pour vous aider :

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: CounterPage(),
    );
  }
}

class CounterPage extends StatefulWidget {
  @override
  _CounterPageState createState() => _CounterPageState();
}

class _CounterPageState extends State<CounterPage> {
  int _counter = 0;

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

  void _decrementCounter() {
    setState(() {
      // À compléter
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Compteur'),
      ),
      body: Center(
        child: Column(
```

```
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              'Compteur : $_counter',
              style: TextStyle(fontSize: 24),
            ),
            SizedBox(height: 20),
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                ElevatedButton(
                  onPressed: _incrementCounter,
                  child: Text('+1'),
                ),
                SizedBox(width: 20),
                ElevatedButton(
                  onPressed: _decrementCounter,
                  child: Text('-1'),
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }
}
```

5. Complétez la fonction `_decrementCounter` pour diminuer le compteur.
6. Testez l'application sur un émulateur ou un appareil.

Livrable : Le fichier `main.dart` complété et une capture d'écran de l'application en fonctionnement.

—

2.4 Exercice 4 : Exploration du Cycle de Vie Android

Objectif : Comprendre le cycle de vie d'une application Android.
Instructions :

1. Créez une nouvelle application Android dans Android Studio (utilisez Kotlin ou Java).
2. Dans l'activité principale (`MainActivity`), ajoutez des messages de journalisation (`Log.d`) pour chaque étape du cycle de vie (`onCreate`, `onStart`, `onResume`, `onPause`, `onStop`, `onDestroy`).
3. Exemple en Kotlin :

```
import android.os.Bundle
import android.util.Log
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    private val TAG = "MainActivity"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        Log.d(TAG, "onCreate called")
    }
```

```
    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart called")
    }

    // Ajouter les autres méthodes
}
```

4. Complétez le code pour les autres étapes du cycle de vie.
5. Lancez l'application et effectuez les actions suivantes :
   — Ouvrir l'application.
   — Mettre l'application en arrière-plan (appuyer sur le bouton Home).
   — Revenir à l'application.
   — Fermer l'application.
6. Observez les messages dans la console Logcat d'Android Studio.

Livrable : Le fichier `MainActivity.kt` complété et un court rapport listant les messages observés dans Logcat pour chaque action.

—

2.5 Exercice 5 : Conception d'une Interface Utilisateur

Objectif : Créer une interface utilisateur respectant les guidelines de design mobile.
Instructions :

1. Choisissez une plateforme (Android avec XML ou Flutter).
2. Créez une interface pour une application fictive de "Liste de Tâches" avec :
   — Une barre d'application avec un titre.
   — Une liste de tâches (utilisez une liste statique pour simplifier).
   — Un bouton flottant pour ajouter une tâche (non fonctionnel).
3. Respectez les guidelines de design :
   — Pour Android : Utilisez Material Design.
   — Pour Flutter : Utilisez des widgets Material.
4. Exemple de code Flutter pour la liste :

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

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: TodoListPage(),
    );
  }
}

class TodoListPage extends StatelessWidget {
  final List<String> tasks = [
    'Faire les courses',
```

```
    'Réviser pour l'examen',
    'Appeler un ami'
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Liste de Tâches'),
      ),
      body: ListView.builder(
        itemCount: tasks.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text(tasks[index]),
          );
        },
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {},
        child: Icon(Icons.add),
      ),
    );
  }
}
```

5. Testez l'interface sur différentes tailles d'écran (utilisez un émulateur).

Livrable : Le code de l'interface et des captures d'écran sur au moins deux tailles d'écran différentes.

—

3 Conclusion

Ces exercices vous permettent de mettre en pratique les concepts du module Environnement Mobile. En les réalisant, vous développerez des compétences en configuration d'environnement, programmation Flutter, compréhension du cycle de vie des applications, et conception d'interfaces utilisateur. Assurez-vous de tester vos applications sur des émulateurs ou des appareils réels pour valider leur fonctionnement.