Cours : Environnement Mobile Filière

Informatique de Développement d'Applications (IDA) Date : 9 juillet 2025

Table des matières

1 Introduction **2**
- 1.1 Objectifs d'apprentissage . . . 2

2 Les Plateformes Mobiles **2**
- 2.1 Android . . . 2
- 2.2 iOS . . . 2
- 2.3 Autres plateformes . . . 3

3 Environnements et Outils de Développement **3**
- 3.1 IDE (Environnements de Développement Intégrés) . . . 3
- 3.2 SDK (Software Development Kits) . . . 3
- 3.3 Émulateurs et Simulateurs . . . 3

4 Concepts de Base du Développement Mobile **3**
- 4.1 Architecture des Applications Mobiles . . . 3
- 4.2 Cycle de Vie d'une Application Mobile . . . 4
- 4.3 Interface Utilisateur (UI) . . . 4

5 Exemple Pratique : Création d'une Application Mobile Simple **4**
- 5.1 Exemple avec Flutter . . . 4
  - 5.1.1 Installation . . . 4
  - 5.1.2 Exemple de code : Application "Bonjour le monde" . . . 4
  - 5.1.3 Explication du code . . . 5
  - 5.1.4 Étapes pour exécuter . . . 5

6 Contraintes Spécifiques des Environnements Mobiles **5**

7 Ressources pour Approfondir **6**

8 Conclusion **6**

1 Introduction

Le module **Environnement Mobile** vise à familiariser les étudiants avec le développement d'applications pour les appareils mobiles. Ce cours couvre les bases des plateformes mobiles, des outils de développement, des paradigmes de programmation, et des bonnes pratiques pour créer des applications fonctionnelles et optimisées pour les smartphones et tablettes.

1.1 Objectifs d'apprentissage

- Comprendre les principales plateformes mobiles (Android, iOS).
- Découvrir les outils et environnements de développement (IDE, SDK).
- Apprendre les bases du développement d'applications mobiles.
- Explorer les concepts de conception d'interfaces utilisateur (UI) adaptées aux écrans mobiles.
- Comprendre les contraintes spécifiques des environnements mobiles (performance, batterie, connectivité).

2 Les Plateformes Mobiles

2.1 Android

- **Système d'exploitation** : Développé par Google, basé sur le noyau Linux.
- **Langage principal** : Java, Kotlin (recommandé depuis 2017).
- **Magasin d'applications** : Google Play Store.
- **Environnement de développement** : Android Studio, qui inclut le SDK Android.
- **Avantages** :
  - Grande variété d'appareils (Samsung, Xiaomi, etc.).
  - Open-source, offrant une grande flexibilité.
- **Inconvénients** :
  - Fragmentation (différentes versions d'Android sur les appareils).
  - Tests nécessaires sur plusieurs appareils.

2.2 iOS

- **Système d'exploitation** : Développé par Apple.
- **Langage principal** : Swift, Objective-C (moins utilisé aujourd'hui).
- **Magasin d'applications** : App Store.
- **Environnement de développement** : Xcode, disponible uniquement sur macOS.
- **Avantages** :
  - Écosystème homogène (moins de fragmentation).
  - Interface utilisateur fluide et optimisée.
- **Inconvénients** :
  - Nécessite un appareil Apple pour le développement.
  - Processus de validation strict pour l'App Store.

2.3 Autres plateformes

- **HarmonyOS** : Développé par Huawei, utilisé principalement en Chine.
- **Frameworks multiplateformes** :
  - **Flutter** : Utilise Dart, permet de développer pour Android et iOS à partir d'une seule base de code.
  - **React Native** : Basé sur JavaScript, populaire pour sa simplicité et sa compatibilité avec React.
  - **Ionic** : Utilise des technologies web (HTML, CSS, JavaScript) pour créer des applications hybrides.

3 Environnements et Outils de Développement

3.1 IDE (Environnements de Développement Intégrés)

- **Android Studio** :
  - Inclut un émulateur pour tester les applications.
  - Supporte Gradle pour la gestion des dépendances.
  - Interface visuelle pour concevoir des layouts.
- **Xcode** :
  - Fournit des outils pour concevoir des interfaces avec Interface Builder.
  - Intègre un simulateur iOS.
- **Visual Studio Code** :
  - Utilisé pour les frameworks multiplateformes comme Flutter ou React Native.
  - Léger, avec des extensions pour le développement mobile.

3.2 SDK (Software Development Kits)

- **Android SDK** : Fournit les bibliothèques et outils nécessaires pour développer sur Android.
- **iOS SDK** : Inclus dans Xcode, offre des API pour accéder aux fonctionnalités d'iOS (caméra, GPS, etc.).
- **Flutter SDK** : Contient les outils pour développer des applications multiplateformes.

3.3 Émulateurs et Simulateurs

- **Émulateur Android** : Permet de tester des applications sur différentes versions d'Android et tailles d'écran.
- **Simulateur iOS** : Simule un iPhone ou iPad pour tester les applications.
- **Avantages** : Pas besoin d'appareil physique.
- **Inconvénients** : Peut être lent sur des machines peu puissantes.

4 Concepts de Base du Développement Mobile

4.1 Architecture des Applications Mobiles

- **Applications natives** : Développées spécifiquement pour une plateforme (Java/Kotlin pour Android, Swift pour iOS).

— **Applications hybrides** : Utilisent des technologies web encapsulées dans une application native (ex. : Ionic, Cordova).

— **Applications multiplateformes** : Une seule base de code pour plusieurs plateformes (ex. : Flutter, React Native).

4.2 Cycle de Vie d'une Application Mobile

— **Android** :

  — `onCreate()` : Initialisation de l'application.

  — `onStart()` : L'application devient visible.

  — `onResume()` : L'application est active.

  — `onPause()`, `onStop()`, `onDestroy()` : Gèrent la mise en pause, l'arrêt ou la destruction.

— **iOS** :

  — `application(:didFinishLaunchingWithOptions:)` : *Lancement de l'application.*`viewDidLoad()` : *Chargement de l'interface.*

— `applicationWillResignActive(:)`, `applicationDidEnterBackground(:)`, *etc.*

4.3 Interface Utilisateur (UI)

— **Composants de base** :

  — Boutons, champs de texte, images, listes.

  — **Android** : Utilise XML pour définir les layouts (ou Jetpack Compose pour une approche moderne).

  — **iOS** : Utilise Storyboards ou SwiftUI.

— **Bonnes pratiques** :

  — Conserver une interface simple et intuitive.

  — Respecter les guidelines de design (Material Design pour Android, Human Interface Guidelines pour iOS).

  — Optimiser pour différentes tailles d'écran.

5 Exemple Pratique : Création d'une Application Mobile Simple

5.1 Exemple avec Flutter

Flutter est un choix populaire pour les débutants car il permet de créer des applications pour Android et iOS avec un seul code.

5.1.1 Installation

1. Téléchargez et installez **Flutter SDK** depuis https://flutter.dev.
2. Configurez un IDE comme **Visual Studio Code** ou **Android Studio**.
3. Ajoutez les plugins Flutter et Dart à votre IDE.

5.1.2 Exemple de code : Application "Bonjour le monde"

Voici un exemple simple d'une application Flutter affichant un texte et un bouton.

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Ma Première Application Mobile'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text(
                'Bonjour le monde !',
                style: TextStyle(fontSize: 24),
              ),
              SizedBox(height: 20),
              ElevatedButton(
                onPressed: () {},
                child: Text('Cliquez ici'),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

5.1.3 Explication du code

- `void main()` : Point d'entrée de l'application.
- `MyApp` : Widget principal qui définit la structure de l'application.
- `MaterialApp` : Fournit le thème et la structure de base (Material Design).
- `Scaffold` : Fournit une structure d'interface avec une barre d'application (AppBar) et un corps (body).
- `Text` et `ElevatedButton` : Widgets pour afficher du texte et un bouton interactif.

5.1.4 Étapes pour exécuter

1. Créez un nouveau projet Flutter avec la commande : `flutter create mon`*application.Remplacezleconten*
   dessus.
2. Lancez l'émulateur ou connectez un appareil mobile.
3. Exécutez l'application avec : `flutter run`.

6 Contraintes Spécifiques des Environnements Mobiles

- **Performance** : Optimisez le code pour réduire l'utilisation du CPU et de la mémoire.

— **Batterie** : Minimisez les tâches en arrière-plan et les appels réseau fréquents.
— **Connectivité** : Gérez les connexions instables (offline/online).
— **Taille d'écran** : Adaptez l'interface à différentes résolutions et densités de pixels.
— **Sécurité** : Protégez les données des utilisateurs (chiffrement, permissions).

7 Ressources pour Approfondir

— **Documentation officielle** :
  — Flutter Documentation
  — Android Developer
  — Apple Developer
— **Tutoriels** :
  — Cours en ligne sur Udemy, Coursera, ou YouTube.
  — Projets pratiques sur Nipissing (Ontario) sur GitHub.
— **Communautés** :
  — Forums comme Stack Overflow.
  — Groupes Discord ou Reddit pour les développeurs mobiles.

8 Conclusion

Le développement mobile est un domaine en constante évolution, avec des opportunités pour créer des applications innovantes. Ce cours fournit une base solide pour comprendre les plateformes, outils, et concepts clés. En pratiquant avec des projets simples comme l'exemple Flutter ci-dessus, vous développerez les compétences nécessaires pour créer des applications performantes et adaptées aux besoins des utilisateurs.