

Développement Mobile : Introduction au Développement d'Applications

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-10-14

Table des matières

1	État du marché mobile en 2025	1
2	Projet Fil Rouge	1
2.1	Rendu et évaluation	2
3	Plateformes mobiles principales	2
4	Approches de développement mobile	2
4.1	Applications natives	2
4.2	Applications multiplateformes	3
5	Langages de programmation	3
5.1	Android	3
5.2	iOS	3
5.3	Multiplateforme	4
6	Outils de développement	4
6.1	Android Studio	4
6.2	Xcode	4
6.3	Visual Studio Code	4
6.4	Outils complémentaires	5
7	Points Majeurs du Développement Mobile	5
8	UI Native	5
8.1	UI Multiplateforme	5
8.2	UI Hybride	6
9	Expérience Utilisateur (UX)	6
10	Gestion des Données	6
10.1	Communication Backend	6
11	Applications Natives Android	7
11.1	Principes Fondamentaux	7
11.2	Caractéristiques Clés	7
11.3	Outils & Environnement	7
12	Applications Flutter	8

12.1	Principes Fondamentaux	8
12.2	Caractéristiques Clés	8
12.3	Outils & Écosystème	8
13	Les Applications avec React Native	8
14	Les Applications avec Ionic/Angular	9
15	Les Applications avec Kotlin Multiplateforme	9
16	Pour ce cours	9

1 État du marché mobile en 2025

- Parts de marché mondiales:
 - Android: ~72%
 - iOS: ~27%
 - Autres: ~1%
- Plus de 7 milliards d'utilisateurs de smartphones
- Le marché des applications représente plus de 200 milliards de dollars

2 Projet Fil Rouge

- Thème : « Proposer une application originale de Choix Social qui utilise les atouts d'un usage mobile. »

« L'application vise à faciliter la prise de décisions collectives en permettant aux utilisateurs de voter électroniquement sur diverses propositions, questions ou problèmes. Elle repose sur des principes de la théorie du

choix social, où les préférences et les opinions de chaque individu contribuent à la décision finale.»

- Travail en équipe sur une application mobile complète :
 - Développement en parallèle de la même applications avec chacune des solutions.
 - Utilisation d'un serveur REST ou d'une base de données NoSQL.

2.1 Rendu et évaluation

- Rendu : Une application mobile implantée avec plusieurs solutions techniques prototype. Un document qui propose une comparaison des approches illustrées par le projet, dresse un bilan et justifie le choix d'une architecture pour un éventuel produit industriel.
- Evaluation double :
 - Créativité : Capacité et méthodologie suivie par l'équipe pour s'approprier les spécificités du développement mobile pour proposer une application originale dans le thème demandé.
 - Ingénierie: Capacité de l'équipe à apprendre/utiliser et comparer les différentes solutions techniques.

3 Plateformes mobiles principales

- Android:
 - Système d'exploitation développé par Google.
 - Utilise Java et Kotlin comme langages de programmation principaux.
 - Présence sur une grande variété de dispositifs.
- iOS:
 - Système d'exploitation développé par Apple.

- Utilise Swift et Objective-C comme langages de programmation principaux.
- Exclusif aux dispositifs Apple tels que l'iPhone et l'iPad.

- Web:

- Application Web dans un conteneur Natif.

4 Approches de développement mobile

4.1 Applications natives

💡 Avantages

- 🏆 **Performance**
 - Optimisation maximale
 - Animations fluides
 - Temps de réponse minimal
- 🛠️ **Intégration**
 - Accès complet aux APIs
 - Support natif du système
 - Fonctionnalités avancées
- 👤 **Expérience**
 - UX optimale
 - Design système natif
 - Accessibilité native

⚠️ Inconvénients

- 💰 **Ressources**
 - Coûts élevés
 - Équipes spécialisées
 - Double développement
- ⌚ **Temps**
 - Cycle long
 - Maintenance multiple
 - Formation spécifique
- 📦 **Contraintes**
 - Dépendance plateforme
 - Compétences distinctes
 - Distribution contrôlée

4.2 Applications multiplateformes

💡 Avantages

- **☒ Développement**
 - Code unique
 - Équipe unique
 - Déploiement unifié
- **💰 Économie**
 - Coûts réduits
 - Maintenance simplifiée
 - Time-to-market rapide
- **🔄 Flexibilité**
 - Mise à jour simultanée
 - Adaptation facile
 - Réutilisation du code

⚠️ Inconvénients

- **↘ Performance**
 - Légèrement réduites
 - Overhead framework
 - Taille plus grande
- **🔧 Limitations**
 - APIs restreintes
 - UX compromise
 - Graphismes limités
- **📦 Dépendances**
 - Framework externe
 - Mises à jour forcées
 - Support variable

5 Langages de programmation

5.1 Android

i Java

- Language historique pour Android (depuis 2008)
- Orienté objet, mature et stable
- Grande bibliothèque standard
- Importante communauté de développeurs
- Support complet dans Android Studio

i Kotlin

- Language moderne adopté officiellement par Google (2019)
- 100% interopérable avec Java
- Syntaxe concise et sûre
- Fonctionnalités modernes: null-safety, coroutines
- Recommandé pour les nouveaux projets Android

5.2 iOS

i Swift

- Langage moderne créé par Apple (2014)
- Remplace progressivement Objective-C
- Syntaxe moderne et sécurisée
- Performances optimisées
- Support natif des dernières APIs iOS

i Objective-C

- Langage historique d'iOS
- Encore présent dans de nombreuses applications
- En cours de remplacement par Swift
- Maintenance des applications existantes

i ☒ Fonctionnalités clés

- Layout Editor visuel
- Profilers de performances
- Debugger avancé
- Gestion des SDK
- Intégration Git

5.3 Multiplateforme

i Web Technologies

- **HTML5/CSS3/JavaScript**
 - Standards du web moderne
 - Frameworks populaires: React, Angular, Vue
 - PWA (Progressive Web Apps)

i Langages spécifiques

- **Dart** (Flutter)
 - Optimisé pour les UI
 - Compilation native
 - Hot Reload
- **TypeScript** (Ionic/Angular)
 - Typage statique
 - Orienté objet
 - Excellent support IDE

6.2 Xcode

i ☒ Caractéristiques principales

- IDE officiel Apple pour iOS/macOS
- Disponible uniquement sur macOS
- Support complet Swift et Objective-C
- Simulateur iOS intégré

i ☒ Fonctionnalités clés

- Interface Builder
- Instruments pour profilage
- TestFlight intégré
- Asset Catalog
- SwiftUI Preview

6.3 Visual Studio Code

i ☒ Caractéristiques principales

- IDE léger et extensible
- Multi-plateforme (Windows, macOS, Linux)
- Excellent pour le développement web
- Grande collection d'extensions

6 Outils de développement

6.1 Android Studio

i ☒ Caractéristiques principales

- IDE officiel pour Android développé par Google
- Basé sur IntelliJ IDEA
- Support complet de Java et Kotlin
- Émulateur Android intégré

Extensions recommandées

- **React Native Tools**
 - Débogage et IntelliSense
- **Flutter**
 - Support Dart et Flutter
- **Angular Essentials**
 - Suite complète pour Angular

6.4 Outils complémentaires

Tests et débogage

- **Firebase Test Lab**
 - Tests sur vrais appareils
- **Chrome ou Firefox DevTools**
 - Débogage web et PWA

Utilitaires

- **Postman**
 - Tests d'API REST
- **GitHub/GitLab**
 - Gestion de version
- **Figma/Sketch**
 - Design d'interface

7 Points Majeurs du Développement Mobile

Points Clés à Maîtriser Pour toute solution de développement mobile, quatre aspects fondamentaux doivent être étudiés et maîtrisés :

1. **Interface Utilisateur (UI)**
 - Composants et layouts
 - Adaptation aux écrans
 - Navigation et flux
 - Thèmes et styles
2. **Expérience Utilisateur (UX)**
 - Performance et réactivité
 - Ergonomie mobile

- Accessibilité
- Modes hors-ligne

3. Gestion des Données

- Stockage local
- État de l'application
- Cache et persistance
- Synchronisation

4. Communication Backend

- APIs REST
- Temps réel
- Sécurité
- Gestion réseau

8 UI Native

- **Android (XML/Jetpack Compose)**
 - Composants Material Design
 - Layouts Android natifs
 - Animations fluides natives
 - Accès direct aux widgets système
- **iOS (Storyboards/SwiftUI)**
 - Composants Human Interface
 - Auto Layout
 - Animations natives
 - Intégration système parfaite

8.1 UI Multiplateforme

- **Flutter**
 - Design unique et cohérent
 - Widgets personnalisables
 - Rendu graphique propre
 - Performance proche du natif
- **React Native**
 - Composants natifs
 - Style façon CSS
 - Hot Reload
 - Large écosystème
- **Ionic**
 - Standards web (HTML/CSS)
 - Composants pré-conçus
 - Adaptation automatique
 - PWA ready

8.2 UI Hybride

- **Web embarqué**
 - Interface web classique
 - WebView native
 - Plugins natifs
 - Déploiement facile
- **PWA**
 - Site web responsive
 - Installation possible
 - Mode hors-ligne
 - APIs web modernes

9 Expérience Utilisateur (UX)

- Facilité d'utilisation: Interface intuitive et conviviale.
- Performances: Réactivité et fluidité de l'application.
- Accessibilité: Assurer l'accessibilité pour tous les utilisateurs.

10 Gestion des Données

Stockage Local

- **Fichiers**
 - Stockage simple et direct
 - Cache et ressources
 - Documents utilisateur
 - Préférences d'application
- **Base de données**
 - Données structurées
 - Requêtes complexes
 - Relations entre données
 - Indexation et recherche

Stockage Distant

- **Solutions Cloud**
 - Services managés (Firebase, AWS)
 - APIs personnalisées (REST)
 - Temps réel (WebSocket)
 - Base de données cloud
- **Synchronisation**
 - Mode hors-ligne
 - Résolution de conflits
 - Mise à jour en arrière-plan
 - Optimisation réseau

10.1 Communication Backend

APIs & Services

- **REST**
 - Architecture client-serveur
 - Opérations CRUD standards
 - Format JSON/XML
 - Stateless
- **GraphQL**
 - Requêtes flexibles
 - Données optimisées
 - Schema typing
 - Single endpoint

Temps Réel

- **Solutions**
 - WebSockets
 - Server-Sent Events
 - Firebase Realtime
 - SignalR
- **Use Cases**
 - Chat/Messaging
 - Notifications Push
 - Live Updates
 - Real-time sync

Sécurité

- **Authentification**

- OAuth 2.0
- JWT tokens
- API keys
- Social auth

- **Protection**

- HTTPS/SSL
- Rate limiting
- Data encryption
- Access control

11.2 Caractéristiques Clés

Avantages Techniques

- **Performance**

- Exécution native
- Optimisation GPU
- Gestion mémoire efficace
- Temps de démarrage rapide

- **Intégration**

- APIs système complètes
- Hardware access
- Background services
- System notifications

11 Applications Natives Android

11.1 Principes Fondamentaux

Définition

- **Application Native Android**

- Développée spécifiquement pour Android
- Utilise Java ou Kotlin (recommandé)
- Suit les guidelines Material Design
- Accès natif aux APIs système

- **Architecture**

- Components Android (Activities, Services)
- Cycle de vie des composants
- Gestion des ressources
- Support multi-densité

Bénéfices Utilisateur

- **Expérience Native**

- UI/UX Android standard
- Animations fluides
- Gestes système
- Multi-fenêtrage

- **Distribution**

- Google Play Store
- In-app purchases
- App bundles
- Play Console analytics

11.3 Outils & Environnement

Développement

- **Android Studio**

- IDE officiel
- Layout Editor
- Performance profilers
- Debugging avancé

- **SDKs & Tools**

- Android SDK
- Build tools
- Emulateurs
- Testing frameworks

12 Applications Flutter

12.1 Principes Fondamentaux

Framework Flutter

- **Définition**
 - Framework UI open-source by Google
 - Développement multiplateforme
 - Langage Dart moderne et typé
 - Interface déclarative avec widgets
- **Architecture**
 - Widget Tree
 - State Management
 - Hot Reload
 - Custom rendering engine

12.2 Caractéristiques Clés

⚡ Avantages Techniques

- **Performance**
 - Compilation native (ARM/x64)
 - Rendering engine dédié
 - 60/120 FPS animations
 - Custom widgets optimisés
- **Développement**
 - Hot Reload instantané
 - Code unique multiplateforme
 - Widgets personnalisables
 - Tests automatisés intégrés

☑ Bénéfices Projet

- **Productivité**
 - Time-to-market rapide
 - Équipe unique
 - Maintenance simplifiée
 - Large écosystème packages
- **Déploiement**
 - iOS App Store
 - Google Play Store
 - Web (PWA)
 - Desktop (Beta)

12.3 Outils & Écosystème

☑ Développement

- **Flutter SDK**
 - CLI puissante
 - DevTools complets
 - Profilers intégrés
 - Documentation riche
- **IDE Support**
 - Android Studio/IntelliJ
 - VS Code
 - DartPad (web)
 - Extensions dédiées

13 Les Applications avec React Native

- **Qu'est-ce que React Native ?**
 - Framework open-source de Facebook pour le développement d'applications multiplateformes.
 - Utilise JavaScript et React.
- **Caractéristiques de React Native**
 - Développement rapide.
 - Partage de code entre les plateformes.
 - Utilisation de composants React pour une interface utilisateur déclarative.

- Compilation en code natif pour plusieurs plateformes.

14 Les Applications avec Ionic/Angular

- **Qu'est-ce que Ionic ?**
 - Framework open-source pour le développement d'applications mobiles multiplateformes.
 - Utilise des technologies web standard : HTML, CSS, JavaScript/TypeScript.
- **Caractéristiques d'Ionic**
 - Développement rapide : Réutilisation de compétences web existantes.
 - Composants UI : Bibliothèque de composants prêts à l'emploi pour une interface utilisateur moderne.
 - Performance : Applications réactives et performantes grâce à l'utilisation de technologies modernes comme Angular ou React.

15 Les Applications avec Kotlin Multiplateforme

- **Qu'est-ce que Kotlin Multiplateforme ?**
 - Approche de développement multiplateforme de JetBrains utilisant le langage Kotlin.
 - Partage de code entre différentes plateformes tout en conservant des performances élevées.
- **Caractéristiques de Kotlin Multiplateforme**
 - Partage de logique métier et de données entre les plateformes.
 - Utilisation de bibliothèques spécifiques à chaque plateforme pour l'interface utilisateur.
 - Compilation en code natif pour chaque plateforme cible.

16 Pour ce cours

- Nous allons tous utiliser Kotlin pour le développement Android.
- Nous allons commencer par apprendre les bases de Kotlin.
- Nous allons explorer les principes fondamentaux du développement Android.
- Nous allons créer des applications natives Android avec Kotlin.
- Puis au choix pour chaque étudiant :
 - Explorer les applications multiplateformes avec Kotlin.
 - Explorer les applications multiplateformes avec Ionic/Angular ou React Native.