ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de HONORIS UNITED UNIVERSITIES

Développement Mobile

Développement d'applications pour Android

Par : I. AITOUHANNI,S.E. BENTALBA, M. BELATAR

Mohammed BELATAR

- Lauréat EMSI Rabat 2004
- Master + Doctorat en Informatique spécialisé en "Dispositifs Interactifs Mobiles" à l'Université de Bretagne-Sud
- 18+ années d'expérience à l'enseignement: l'IUT+IUP de Vannes et Lorient, ENSIBS, Académie militaire de Saint-Cyr Coëtquidan, IQA et EMSI
- Ingénieur R&D chez Orange Labs (France Telecom) à Lannion 2009-2010
- Fondateur de TECHNOLOGY & TELECOM 2012-aujourd'hui

Chapitre 1 : Environnement de Développement Mobile sous Android

Objectifs du chapitre

- Comprendre l'écosystème Android
- Installer et configurer un environnement de développement Android
- Connaître les différents types d'appareils Android
- Découvrir les approches de développement mobile : Native, Hybride, Web

Qu'est-ce qu'Android ?

- Système d'exploitation mobile développé par Google
- Open source, basé sur un noyau Linux
- Langages utilisés : Java, Kotlin, (possible C++)
- Android = écosystème complet (OS + SDK + Google Play)

Historique rapide

1. **2008** — Première version publique
2. **2011** — Android 4.0 Ice Cream Sandwich
3. **2014** — Android 5.0 Lollipop (Material Design)
4. **2017** — Kotlin devient langage officiel
5. **2019** — Android 10 (fin des noms de desserts)
6. **2021** — Android 12 (Material You)
7. **2023** — Android 14
8. **2024** — Android 15
9. **2025** — Android 16 en développement

L'écosystème Android

Android OS

Android SDK

Google Play Services

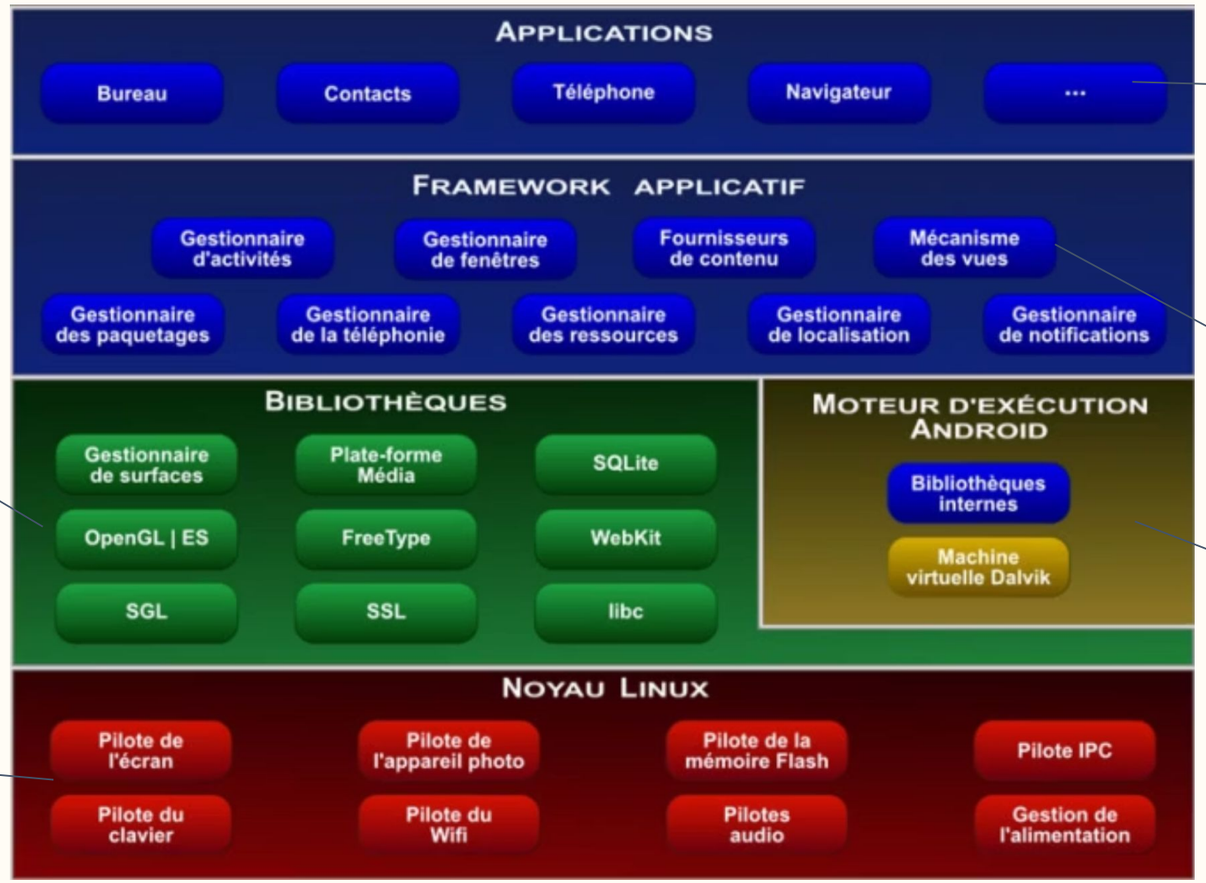
Jetpack (bibliothèques support)

Appareils Android (smartphones, tablettes, wearables)

Architecture Android (schéma)

Ensemble des bibliothèques fournissant un accès direct aux ressources du système.

Noyau Linux : une couche d'abstraction entre le matériel et le reste de la pile logicielle pour accéder aux périphériques.



englobe toutes les applications natives d'un smartphone: Camera, Calculatrice, agenda.....

Ensemble des Framework qui permet de développer l'application

une machine virtuelle Java adaptée : Machine Virtuelle Dalvik

Android Studio

- IDE officiel recommandé par Google
- Basé sur IntelliJ IDEA

Outils inclus :

- Editeur de code et UI
- Gestionnaire d'émulateurs (AVD)
- Debugger, Profiler
- Intégration Git/Gradle

Prérequis Système pour Android Studio

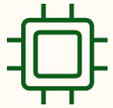
Pour une expérience de développement optimale avec Android Studio, assurez-vous que votre système répond aux spécifications suivantes :

Systèmes d'exploitation

Windows (64-bit, 8/10/11), macOS (10.14 ou supérieur), ou Linux (64-bit compatible GNOME/KDE).

Mémoire Vive (RAM)

Minimum 8 Go, avec 16 Go recommandés pour des performances fluides lors de la compilation et l'émulation.



Processeur

Intel Core i5 ou AMD Ryzen 5 (ou équivalent) avec support de la virtualisation (VT-x ou AMD-V).

Espace Disque

8 Go d'espace libre minimum. Un SSD est fortement recommandé pour accélérer les temps de chargement et de compilation.

Java Development Kit

JDK version 11 ou ultérieure est indispensable pour la compilation de vos projets Android.

Téléchargement & Installation

01

Installer Android Studio

depuis developer.android.com

02

Installer SDK Platform Tools, Build Tools

API 24, 35, 36

03

Créer un nouveau projet

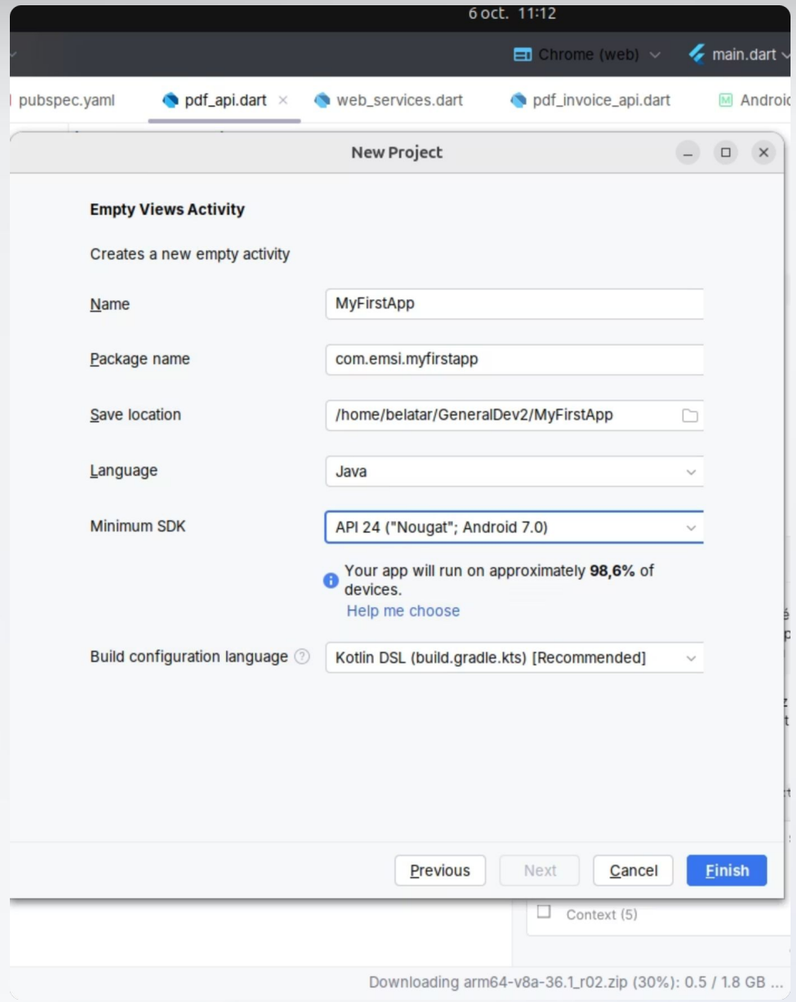
avec un "Empty Views Activity"

04

Choisir un nom de package, langage (Java), niveau API (MinSDKVersion 24)

Création d'un projet (pas à pas)

- New Project → Empty Views Activity
- Nom : MyFirstApp
- Langage : Java
- API minimum : API 24 (Android 7.0)
- Nom du fichier principal : MainActivity.java



Structure d’un projet Android

Comprendre l'organisation des fichiers et des dossiers est essentiel pour naviguer et développer efficacement au sein d'un projet Android.

Voici les composants clés :

app/src/main/java

Ce répertoire contient tout le code source Java (ou Kotlin) de votre application, y compris vos activités, services et classes personnalisées.

res/layout

Ce dossier contient les fichiers XML qui définissent la mise en page de l'interface utilisateur (UI) de chaque écran ou composant de votre application.

res/values

Ce répertoire regroupe les ressources non visuelles telles que les chaînes de caractères (strings.xml), les styles (styles.xml), les couleurs (colors.xml), et les dimensions (dimens.xml), facilitant la localisation et la thématisation.

AndroidManifest.xml

Ce fichier manifeste est le cœur de votre application. Il déclare toutes ses composantes, ses permissions, et ses métadonnées au système Android.

MainActivity.java : Le point d'entrée de votre application

Le fichier `MainActivity.java` est le cœur de votre première activité Android. Il gère le cycle de vie de l'écran principal de votre application et définit le contenu qui y sera affiché. Le code ci-dessous montre la structure de base d'une activité.

```
public class MainActivity extends AppCompatActivity {
    @Override  protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

La méthode `onCreate()` est appelée lors de la création de l'activité. C'est ici que l'interface utilisateur est définie à l'aide de `setContentView(R.layout.activity_main)`, qui charge la mise en page (layout) définie dans le fichier XML `activity_main.xml`.

Appareils Android : Diversité et Écosystème

L'écosystème Android ne se limite pas aux smartphones. Il s'étend à une multitude d'appareils, chacun offrant des fonctionnalités et des expériences utilisateur uniques, s'adaptant à différents besoins et contextes d'utilisation.

Smartphones

Le dispositif le plus répandu, offrant un accès complet aux applications, services et fonctionnalités de communication pour la vie quotidienne.

Tablettes

Avec des écrans plus grands, les tablettes Android sont idéales pour la productivité, la consommation de médias et les jeux, avec une interface optimisée.

Montres Connectées (Wear OS)

Ces compagnons de poignet intelligents gèrent les notifications, le suivi de la santé et permettent un contrôle rapide des fonctionnalités du téléphone.

Android TV

Intégré aux téléviseurs et aux boîtiers de streaming, Android TV offre une interface utilisateur pour accéder aux applications de divertissement, films et jeux.

Appareils Connectés (IoT)

Android et ses dérivés alimentent une gamme croissante d'objets connectés, des systèmes de domotique aux écrans intelligents, étendant son influence au-delà des appareils mobiles traditionnels.

Particularités du Développement Mobile

Le développement d'applications Android implique de gérer la grande diversité des appareils et des tailles d'écran, ce qui présente des défis uniques.

Design Responsif

Adapter l'interface utilisateur pour qu'elle s'affiche correctement sur une multitude de tailles et résolutions d'écran est essentiel.

Layouts Spécifiques

Utiliser des ressources alternatives (ex: `layout-sw600dp`) pour optimiser l'expérience sur les tablettes et grands écrans.

Fragmentation

La diversité des versions d'Android et des spécifications matérielles nécessite des tests rigoureux sur divers appareils.

Android Virtual Device (AVD)

L'Android Virtual Device (AVD) est un outil indispensable pour simuler un environnement Android complet directement sur votre machine de développement.

Simulation fidèle : Réplique le comportement et les fonctionnalités d'un appareil Android réel (téléphone, tablette, montre, TV, etc.).

Tests sans matériel : Permet de tester vos applications sur diverses configurations matérielles et versions d'Android sans avoir besoin d'acheter de multiples appareils physiques.

Création facile : Disponible et configurable directement depuis Android Studio via le menu `Tools → Device Manager`.

Débogage sur Smartphone Android

Le débogage sur un smartphone Android physique offre une expérience plus réaliste et souvent plus performante que l'utilisation d'un émulateur. Voici les étapes et avantages clés pour mettre en place le débogage direct sur votre appareil.

Performance Optimale

Utiliser un appareil physique permet des tests plus rapides et plus fluides, offrant une fidélité d'expérience utilisateur inégalée par rapport aux émulateurs qui nécessitent une machine performante.

Activer les Options Développeur

Accédez aux **Paramètres** de votre téléphone, puis **À propos du téléphone**, et tapez sept fois sur le **Numéro de build** pour débloquer ces options cachées.

Activer le Débogage USB

Une fois les options développeur activées, vous trouverez l'option **Débogage USB** à activer. Cela permet à Android Studio de communiquer avec votre appareil via un câble USB.

Débogage via Wi-Fi

Pour une flexibilité accrue, configurez le débogage sans fil. Cela vous permet d'exécuter et de déboguer votre application sans connexion physique, après une configuration initiale via USB.

Lancement de l’application

Après avoir configuré votre projet et écrit votre code, le lancement de l'application est l'étape suivante pour voir votre création prendre vie.

Démarrer l'exécution

Utilisez le bouton **"Run"** (icône ▶) dans la barre d'outils d'Android Studio.

Sélection de la cible

Choisissez si vous voulez exécuter l'appli sur un **AVD** (émulateur) ou un **smartphone USB** connecté.

Surveillance Logcat

Ouvrez la fenêtre **"Logcat"** pour observer les messages de débogage et les erreurs de l'application.

Votre "Hello World!"

Félicitations ! Votre application se lance et affiche le célèbre message **"Hello World!"**.

Approches de Développement Mobile

Le choix de la technologie est une décision stratégique cruciale lors de la conception d'une application mobile. Il existe plusieurs approches principales, chacune offrant un équilibre distinct entre performance, coût de développement et portée multi-plateforme.

Développement Natif

Utilise les langages et outils spécifiques à la plateforme (Java/Kotlin avec l'Android SDK pour Android). Cette approche garantit des performances optimales, une intégration complète avec les fonctionnalités du système d'exploitation et la meilleure expérience utilisateur.

Développement Hybride

Des frameworks comme Ionic ou Cordova permettent de développer des applications avec des technologies web (HTML, CSS, JavaScript) encapsulées dans un conteneur natif (WebView). Ceci réduit les coûts et le temps pour cibler plusieurs plateformes, souvent au détriment de la performance pure.

Web Mobile (PWA)

Les Progressive Web Apps sont des sites web qui imitent le comportement des applications natives. Elles sont accessibles via un navigateur, peuvent être "installées" sur l'écran d'accueil, et offrent des fonctionnalités comme l'accès hors ligne, sans nécessiter de téléchargement depuis un store.

Applications Natives via Frameworks

Une application native n'est pas forcément codée directement avec les SDKs spécifiques à Android. Des frameworks multi-plateformes permettent de créer des applications performantes et à l'apparence native à partir d'une seule base de code pour iOS et Android.

Flutter

Développé par Google, ce framework utilise le langage **Dart** pour construire des interfaces utilisateur riches et expressives. Il permet de compiler le code directement en code machine natif pour des performances élevées.

React Native

Créé par Facebook, ce framework s'appuie sur **JavaScript** (ou TypeScript) et la librairie React pour développer des applications mobiles. Il utilise des composants UI natifs pour offrir une expérience utilisateur authentique.

Bénéfices

Ces approches offrent un développement plus rapide, une maintenance simplifiée et une cohérence visuelle sur différentes plateformes, optimisant ainsi les ressources.

Comparaison des Approches de Développement Mobile

Le tableau ci-dessous récapitule les principales caractéristiques et compromis de chaque approche, vous aidant à choisir la solution la plus adaptée à vos besoins spécifiques.

Critère	Native	Hybride	Web Mobile
Performance	⭐⭐⭐⭐⭐	⭐⭐⭐	⭐⭐
Accès API Android	Complet	Limité	Très limité
Accès hors ligne	Oui	Oui	Partiel
Mises à jour	Moins facile	Facile	Immédiates

Bonnes pratiques pour débuter

Pour un développement Android efficace et la création d'applications robustes, il est essentiel d'adopter de bonnes habitudes dès le début. Voici quelques pratiques clés :

Tester sur diverses résolutions

Les appareils Android existent en une multitude de tailles d'écran et de densités de pixels. Assurez-vous que votre application s'affiche correctement et est utilisable sur différentes résolutions pour garantir une expérience utilisateur cohérente.

Maîtriser le Débogueur et Logcat

Le débogueur d'Android Studio et la fenêtre Logcat sont vos meilleurs amis pour identifier et résoudre les problèmes. Apprenez à les utiliser efficacement pour suivre le flux de votre application et diagnostiquer les erreurs rapidement.

Commenter et Structurer le Code

Un code bien commenté et organisé est plus facile à comprendre, à maintenir et à faire évoluer, que ce soit pour vous ou pour d'autres développeurs. Suivez les conventions de codage et structurez vos fichiers de manière logique.

Nommer clairement les Composants UI

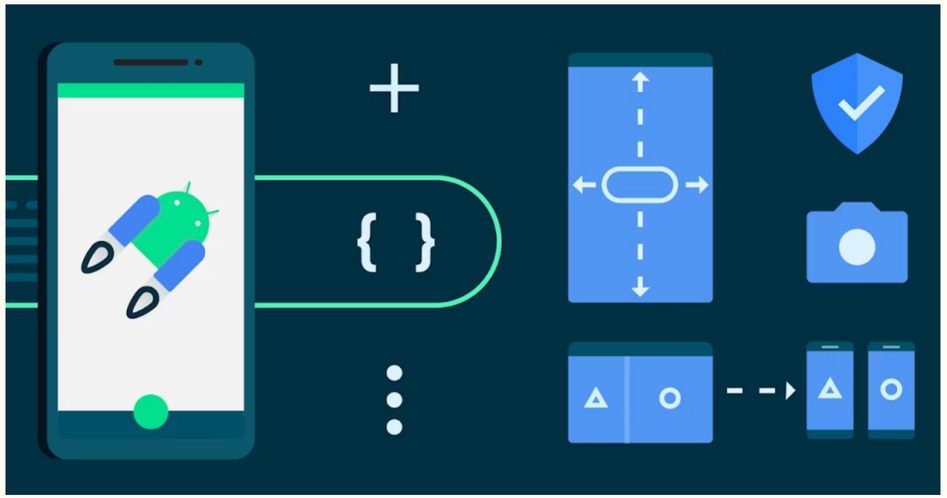
Utilisez des noms descriptifs et cohérents pour les identifiants (IDs) de vos vues (boutons, champs de texte, etc.) et autres ressources UI. Cela améliore grandement la lisibilité du code XML et Java/Kotlin de votre interface.

Android Jetpack (Vue d'ensemble)

Android Jetpack est une suite de bibliothèques développées par Google pour accélérer le développement d'applications Android. Il aide à réduire le code passe-partout et à construire des applications robustes et performantes.

Simplification du Développement

- Gère les tâches complexes liées au cycle de vie des composants, à la navigation et à la persistance des données.
- Encourage les bonnes pratiques de conception et d'architecture.
- Assure une meilleure compatibilité entre les versions d'Android.



Composants Clés

LiveData et ViewModel : Gestion du cycle de vie et des données réactives.

Navigation : Facilite les transitions entre écrans.

Room : Couche d'abstraction pour la base de données SQLite.

WorkManager : Gestion des tâches en arrière-plan fiables.

Ressources utiles

Pour approfondir vos connaissances et résoudre les défis rencontrés durant le développement Android, une multitude de ressources sont à votre disposition. Exploitez ces outils pour continuer à apprendre et à maîtriser l'écosystème Android.

Documentation Officielle

Le site developer.android.com est la référence incontournable. Il contient des guides détaillés, des références d'API, des exemples de code et les dernières actualités du développement Android.

Codelabs Interactifs

Les Codelabs Android proposent des tutoriels pratiques et guidés pour apprendre de nouvelles fonctionnalités ou technologies, souvent avec des exercices pas à pas et des exemples concrets.

Communautés et Forums

Des plateformes comme **Stack Overflow** sont essentielles pour trouver des réponses à vos questions techniques. **GitHub** regorge d'exemples de projets open source à explorer et dont vous pouvez vous inspirer.

Build Your First Android App (Project-Centered Course)

Un module Coursera de Centrale Supélec:

https://www.coursera.org/learn/android-app