

PROGRAMMATION MOBILE



ECOLE MAROCAINE DES
SCIENCES DE L'INGENIEUR
Membre de
HONORIS UNITED UNIVERSITIES

Pr. FATIMA ZAHRA FAGROUD

INTRODUCTION AUX APPLICATIONS MOBILES



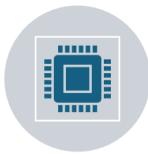
Introduction

- ❖ De nos jours, les appareils mobiles sont équipés d'une multitude d'applications pouvant accomplir une variété de tâches différentes.
- ❖ Le développement d'une application mobile efficace est devenu actuellement un enjeu important pour les entreprises afin d'étaler leurs services ou leurs produits, et d'établir une liaison directe avec les clients, et aussi pour le grand public afin de communiquer et collaborer avec les membres de la famille, les amis et collègues de travail.

Introduction

- ❖ Le développement mobile consiste à créer des applications logicielles pour des appareils mobiles tels que les smartphones et les tablettes.
- ❖ Ces applications peuvent être conçues pour des systèmes d'exploitation spécifiques (comme iOS ou Android) ou être multiplateformes, fonctionnant sur plusieurs systèmes.
- ❖ Le développement mobile est un domaine en pleine expansion, porté par l'utilisation massive des smartphones et l'innovation technologique.

Application Mobile : Définition



UNE APPLICATION MOBILE EST UN LOGICIEL DÉVELOPPÉ SPÉCIFIQUEMENT POUR ÊTRE UTILISÉ SUR UN APPAREIL MOBILE, TEL QU'UN SMARTPHONE OU UNE TABLETTE.



CES APPLICATIONS SONT CONÇUES POUR FOURNIR DES Fonctionnalités Spécifiques AUX UTILISATEURS, TELLES QUE LA COMMUNICATION, LE DIVERTISSEMENT, LA PRODUCTIVITÉ, OU TOUT AUTRE SERVICE ADAPTÉ À UN USAGE MOBILE.



ELLES SONT GÉNÉRALEMENT TÉLÉCHARGÉES ET INSTALLEES À PARTIR D'UNE BOUTIQUE D'APPLICATIONS, COMME L'APP STORE POUR IOS OU GOOGLE PLAY STORE POUR ANDROID.

Application Mobile : Définition

- Une application mobile s'exécute sur un support matériel **mobile**:
 - Ressources limitées
 - Batterie (énergie), interface graphique, CPU, périphériques d'IO, ...
 - Périphériques très **divers**
 - De très élémentaire au très évolué
 - **Utilisation** ubiquitaire
 - Ubiquité géographique
 - Ubiquité des utilisateurs
 - Utilisation basée sur l'interaction avec l'utilisateur via une interface mobile

Application Mobile : Evolution

- ❖ **Années 1990** : Apparition des premiers téléphones mobiles avec des applications basiques (jeux Snake sur Nokia, calculatrices, calendriers).
- ❖ **Années 2000** : Introduction des smartphones (BlackBerry, Palm) avec des applications plus avancées.
- ❖ **2007** : Lancement de l'iPhone et de l'App Store, marquant le début de l'économie des applications mobiles.
- ❖ **2008** : Google lance Android et son Play Store, rendant le marché plus concurrentiel.
- ❖ **2010s** : Expansion des applications mobiles avec les réseaux sociaux, le cloud et les objets connectés.
- ❖ **Aujourd'hui** : Dominance des applications intelligentes exploitant l'IA, la réalité augmentée et la 5G.

Application Mobile : Domaine d'application

- ❖ Les domaines d'application d'une application mobile sont très vaste.
- ❖ Exemple :
 - ❖ la géolocalisation
 - ❖ Itinéraire
 - ❖ Payement mobile
 - ❖ Les jeux
 - ❖ Les réseaux sociaux

Application Mobile : Importance

- ❖ **Accès** : Elles permettent aux utilisateurs d'accéder à des services et des informations en tout lieu et à tout moment.
- ❖ **Productivité** : Les applications de gestion de tâches, de messagerie et de collaboration ont transformé la manière dont nous travaillons.
- ❖ **Divertissement** : Jeux, streaming vidéo et musique rendent les temps libres plus agréables.
- ❖ **Commerce** : Le shopping en ligne via des applications mobiles a révolutionné le secteur du commerce de détail.
- ❖ **Santé** : Les applications de suivi de la santé et du bien-être contribuent à améliorer la qualité de vie.

Application Mobile : Système d'exploitation

- ❖ Un système d'exploitation (OS) est un programme qui agit comme une interface entre le matériel du système et l'utilisateur. De plus, il gère toutes les interactions entre le logiciel et le matériel.
- ❖ Un système d'exploitation mobile permet à l'utilisateur d'exécuter de différents logiciels d'application sur le mobile, les tablettes, les montres intelligentes, etc.
- ❖ Un type de système d'exploitation spécialement conçu pour les mobiles.
- ❖ Ils sont un mélange de système d'exploitation informatique avec quelques fonctionnalités supplémentaires. En outre, ils sont relativement légers et simples.

Application Mobile : Système d'exploitation



Les principaux systèmes d'exploitation pour les applications mobiles sont :



- Android : Développé par Google, utilisé sur une grande variété d'appareils mobiles.



- iOS : Développé par Apple, utilisé exclusivement sur les appareils iPhone et iPad.



- Windows Phone : Développé par Microsoft, utilisé principalement sur les appareils Windows Phone.



- BlackBerry OS : Développé par BlackBerry, utilisé sur les appareils BlackBerry.



- HarmonyOS : Développé par Huawei, visant à être utilisé sur divers appareils intelligents.

Application Mobile : Types

Les applications mobiles sont généralement classées en trois types : natives, hybrides, et Web.

- ❖ Les applications natives sont conçues spécifiquement pour un système d'exploitation mobile, généralement iOS ou Android.
- ❖ Les applications Web sont écrites en HTML5 et CSS et s'exécutent généralement via un navigateur.
- ❖ Les applications hybrides sont conçues à l'aide de technologies Web telles que JavaScript, CSS et HTML 5 et fonctionnent comme des applications Web mais dans un conteneur natif.

Application Mobile : Types

Applications natives

- ❖ Les applications natives sont conçues spécifiquement pour le système d'exploitation (OS) d'un appareil mobile.
- ❖ Ces applications ne fonctionnent que sur les appareils pour lesquels elles sont conçues.
- ❖ L'objectif principal de la création de telles applications est d'assurer les meilleures performances pour un système d'exploitation mobile spécifique.
- ❖ Technologies utilisées:
 - Les applications natives sont codées à l'aide de divers langages de programmation. Voici quelques exemples : Java, Kotlin, Python, Swift, Objective-C, C++ et React.

Application Mobile : Types

Les avantages des applications natives sont :

- ❖ elles communiquent facilement avec l'appareil sur lequel elles s'exécutent.
- ❖ Le rendu de la page, c'est-à-dire le rendu de l'interface utilisateur, se poursuit sans aucun problème de suspension ou de blocage.
- ❖ Le flux de mise en page est impeccable et facile à utiliser.
- ❖ Ces applications sont plus conviviales, plus faciles à utiliser et peuvent facilement tolérer une charge d'activité rapide des utilisateurs.
- ❖ Ces applications peuvent réagir à la même vitesse avec ce que l'utilisateur les utilise.

Application Mobile : Types

Applications Web

- ❖ Les applications Web se comportent de la même manière que les applications natives, mais sont accessibles via un navigateur Web sur votre appareil mobile.
- ❖ Ce ne sont pas des applications autonomes dans le sens où vous devez télécharger et installer du code sur votre appareil.
- ❖ Ce sont en fait des sites Web réactifs qui adaptent leur interface utilisateur à l'appareil sur lequel l'utilisateur se trouve.
- ❖ Un type d'application Web est l'application Web progressive (PWA), qui est essentiellement une application native exécutée dans un navigateur.

Application Mobile : Types

- ❖ L'avantage de ces applications est : le gain de temps et d'argent réalisé grâce à leur développement unique et leur déploiement multiplateformes.
- ❖ Cette approche n'offre pas d'accès aux fonctionnalités "natives" des systèmes et nécessite une connexion Internet.
- ❖ Technologies utilisées:
 - Les applications Web sont conçues à l'aide de HTML5, CSS, JavaScript, Ruby et des langages de programmation similaires utilisés pour le travail Web.

Application Mobile : Types

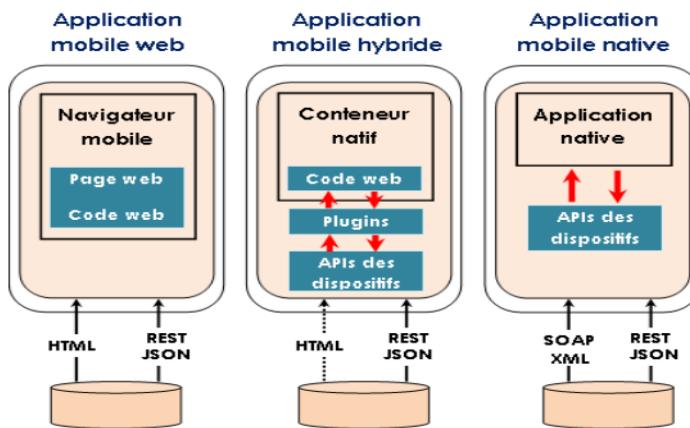
Applications hybrides

- ❖ Le terme hybride signifie « mélange ». Ainsi, une application hybride fait référence à des applications natives et basées sur le Web.
- ❖ Les applications hybrides sont une intégration d'applications natives et Web. Ces applications utilisent à la fois des technologies Web et des API natives.
- ❖ Ces applications peuvent utiliser la plupart des fonctionnalités de l'appareil, telles que les applications natives.
- ❖ les applications hybrides s'exécutent dans un conteneur natif et exploitent le moteur de navigateur de l'appareil pour afficher le code HTML et traiter le code JavaScript localement.

Application Mobile : Types

- ❖ L'avantage de ces applications c'est qu'elles nous permettent d'utiliser un seul et même outil pour le développement et les langages issus du développement Web pour tous les mobiles (IOS, Android et Windows Phone).
- ❖ Elles peuvent être le bon choix en fonction des besoins et des contraintes de ressources.
- ❖ Technologies utilisées :
 - Les applications hybrides utilisent un mélange de technologies Web et d'API natives. Ils sont développés en utilisant : Ionic, Objective C, Swift, HTML5 et autres.
 - Les applications développées à l'aide d'Apache Cordova, Xamarin, React Native, Sencha Touch entrent dans cette catégorie d'applications hybride.

Application Mobile : Types



Application Mobile : Types

Critères	Applications Web	Applications Hybrides	Applications Natives
Compatibilité	Multi-plateforme	Multi-plateforme	Spécifique à un OS (Android/iOS)
Développement	Rapide et moins coûteux	Plus rapide que natif	Plus long et coûteux
Performances	Limitée	Inférieure aux applications natives	Optimales
Fonctionnalités	Réduites (accès limité aux fonctionnalités matérielles)	Moyenne (via des plugins)	Complètes (accès total aux fonctionnalités matérielles)
Expérience utilisateur	Moyenne	Moyenne à bonne	Excellent
Maintenance	Facile	Facile	Nécessite deux bases de code
Connexion Internet	Nécessaire	Peut fonctionner hors ligne	Peut fonctionner hors ligne
Mises à jour	Automatiques via le web	Via les magasins d'applications	Via les magasins d'applications
Sécurité	Moyenne	Moyenne	Meilleure

Application Mobile : Types

Applications fonctionnant sans internet	Appelées applications indépendantes, ce sont des applications qui fonctionnent sans avoir besoin de connexion internet ou téléphonique dont on cite liste de contacts, calculatrice et autres.
Applications exigeant connexion	Contrairement aux applications indépendantes, ces applications doivent avoir accès à l'internet pour fonctionner
Applications connectées	C'est une application qui nécessite une connexion internet pour un bon fonctionnement.

Application mobile: Composantes

Interface de l'utilisateur :

- ❖ **les composantes visibles** : sont représentées par les boutons, champs de texte, Images, Liste déroulante, etc. elles sont utilisées comme un moyen d'interaction avec l'utilisateur pour lui permettre d'entrer des informations ou lui afficher un contenu.
- ❖ **les composantes invisibles** : sont des composantes qui permettent une interaction avec les utilisateurs sans qu'elles soit visible en permettant d'utiliser les ressources du mobile comme l'envoi et la réception des SMS, la géolocalisation, textToSpeech, accéléromètre, etc.

Comportement :

- ❖ **Évènements** : ce sont les évènements auxquels peut s'associer une action comme le clique d'un bouton, l'activation de camera, la réception d'un appel, etc.
- ❖ **Méthodes** : ce sont les actions et les comportements qu'on veut associer à des évènements. Elles sont paramétrables, réutilisables et personnalisables.

Application mobile: Écosystème

Langages de Programmation :

- ❖ **Java** : Utilisé principalement pour le développement d'applications Android. C'est un langage orienté objet avec une syntaxe robuste et une vaste bibliothèque de classes.
- ❖ **Kotlin** : Un langage moderne qui est également utilisé pour le développement Android. Kotlin est plus concis et moins sujet aux erreurs que Java, et il est maintenant officiellement pris en charge par Google pour le développement Android.
- ❖ **Swift** : Le langage de programmation développé par Apple pour créer des applications iOS, macOS, watchOS et tvOS. Swift est conçu pour être rapide, sûr et interactif.
- ❖ **JavaScript** : Utilisé avec des frameworks comme React Native et Ionic pour le développement d'applications mobiles hybrides. JavaScript permet de créer des interfaces utilisateur dynamiques et interactives.

Application mobile: Écosystème

Outils de Développement :

- ❖ **Android Studio** : L'environnement de développement intégré (IDE) officiel pour le développement d'applications Android, basé sur IntelliJ IDEA. Il fournit des outils puissants pour la création, le test et la débogage des applications.
- ❖ **Xcode** : L'IDE officiel pour le développement d'applications iOS et macOS, fourni par Apple. Il inclut un simulateur, des outils de débogage avancés et une intégration complète avec les services Apple.
- ❖ **Frameworks Hybrides :**
 - **React Native** : Permet de créer des applications mobiles en utilisant JavaScript et React. Les applications peuvent fonctionner sur iOS et Android avec une base de code unique.
 - **Flutter** : Un framework open-source de Google pour le développement d'applications nativement compilées pour mobile, web, et desktop à partir d'une seule base de code.
 - **Ionic** : Un framework open-source pour le développement d'applications mobiles hybrides en utilisant des technologies web telles que HTML, CSS et JavaScript.

Application mobile: Écosystème

Plateformes de Distribution :

- ❖ **App Store** : La boutique d'applications d'Apple pour les appareils iOS. Elle impose des standards de qualité stricts et des critères de sécurité rigoureux pour les applications.
- ❖ **Google Play** : La boutique d'applications de Google pour les appareils Android. Elle offre un large public et des options variées de monétisation pour les développeurs.

Application mobile: Enjeux et Défis

Expérience Utilisateur (UX) et Design

- ❖ **Ergonomie** : Assurer une interface intuitive et facile à utiliser. Les utilisateurs doivent pouvoir naviguer et interagir avec l'application sans effort.
- ❖ **Esthétique** : Un design attrayant et cohérent augmente l'engagement et la satisfaction des utilisateurs.
- ❖ **Réactivité** : L'application doit être réactive et offrir une expérience fluide, sans latence perceptible.
- ❖ **Accessibilité** : Prendre en compte les besoins des personnes handicapées en intégrant des fonctionnalités telles que le lecteur d'écran, le contraste élevé et les options de texte agrandi.

Application mobile: Enjeux et Défis

Performance et Optimisation

- ❖ **Vitesse de Chargement :** L'application doit se charger rapidement pour éviter de frustrer les utilisateurs.
- ❖ **Utilisation de la Mémoire :** Optimiser l'utilisation de la mémoire pour éviter les ralentissements et les crashes.
- ❖ **Consommation d'Énergie :** Minimiser l'impact sur la batterie pour que l'application puisse être utilisée pendant de longues périodes.
- ❖ **Gestion des Ressources :** Gérer efficacement les ressources telles que les images et les animations pour assurer des performances optimales.

Application mobile: Enjeux et Défis

Sécurité et Protection des Données

- ❖ **Chiffrement des Données :** Utiliser le chiffrement pour protéger les données sensibles pendant la transmission et le stockage.
- ❖ **Authentification et Autorisation :** Mettre en place des mécanismes robustes pour garantir que seuls les utilisateurs autorisés peuvent accéder à certaines fonctionnalités ou données.
- ❖ **Protection Contre les Attaques :** Protéger l'application contre les vulnérabilités courantes telles que les injections SQL, les attaques XSS et les fuites de données.
- ❖ **Politique de Confidentialité :** Informer les utilisateurs sur la manière dont leurs données sont collectées, utilisées et protégées.

Application mobile: Enjeux et Défis

Compatibilité et Fragmentation des Appareils

- ❖ **Diversité des Dispositifs :** Assurer que l'application fonctionne bien sur une variété de dispositifs avec des tailles d'écran, des résolutions et des capacités matérielles différentes.
- ❖ **Versions des Systèmes d'Exploitation :** Maintenir la compatibilité avec plusieurs versions d'iOS et d'Android, car tous les utilisateurs ne mettent pas à jour leurs appareils régulièrement.
- ❖ **Tests Multiplateformes :** Effectuer des tests sur différents dispositifs et simulateurs pour identifier et corriger les problèmes spécifiques à certains appareils ou configurations.
- ❖ **Adaptabilité :** Concevoir des interfaces adaptatives qui s'ajustent automatiquement aux différentes tailles d'écran et orientations.

Quiz

Qu'est-ce qu'une application mobile ?

- Un programme conçu pour fonctionner sur des appareils mobiles comme les smartphones et les tablettes.
- Un logiciel exclusivement utilisé sur les ordinateurs.
- Un système d'exploitation pour les téléphones portables.
- Une fonctionnalité matérielle des smartphones.

Quel est le système d'exploitation mobile développé par Apple ?

- Android
- Windows Phone
- iOS
- Tizen

Quiz

Quel est l'avantage principal des applications natives ?

- Elles fonctionnent sur tous les systèmes d'exploitation sans modification.
- Elles offrent des performances optimales et un accès complet aux fonctionnalités du matériel.
- Elles sont moins chères à développer.
- Elles ne nécessitent pas de téléchargement.

Quel langage de programmation est principalement utilisé pour développer des applications Android ?

- Swift
- JavaScript
- C#
- Kotlin

Quiz

Quel est le rôle d'un système d'exploitation mobile ?

- Gérer uniquement les applications tierces.
- Servir d'interface entre l'utilisateur et le matériel, tout en gérant les ressources.
- Remplacer les applications natives.
- Limiter l'accès aux fonctionnalités du téléphone.

Quiz

Quel est un exemple de composante invisible dans une application mobile ?

- Un bouton
- Un champ de texte
- La géolocalisation
- Une image

Quel est l'outil de développement officiel pour les applications Android ?

- Android Studio
- Xcode
- Visual Studio
- Eclipse

Quiz

La sécurité des données n'est pas un enjeu important dans le développement d'applications mobiles.

- Vrai
- Faux

Android Studio est l'outil de développement officiel pour les applications iOS.

- Vrai
- Faux

Les événements dans une application mobile incluent des actions comme le clic sur un bouton ou la réception d'un SMS.

- Vrai
- Faux



2024-2025

PROGRAMMATION MOBILE(FAGROUD)

35

INTRODUCTION À ANDROID

Qu'est-ce qu'Android OS ?

Android OS est un système d'exploitation mobile développé par Google.

Il est basé sur le noyau Linux et est principalement utilisé dans les smartphones et les tablettes.

Android OS est connu pour sa flexibilité, sa personnalisation et son large éventail d'applications disponibles sur Google Play Store.

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

36

Composants de l'écosystème Android



L'écosystème Android comprend plusieurs composants clés :



- Système d'exploitation Android : Le noyau Linux et la couche logicielle développée par Google.



- Appareils Android : Smartphones, tablettes, téléviseurs, montres intelligentes, etc.



- Google Play Store : La principale plateforme de distribution d'applications pour Android.



- Développeurs Android : La communauté de développeurs qui créent des applications pour Android.

Android pour qui ?



Android est conçu pour une large gamme d'utilisateurs, notamment :



- Utilisateurs finaux : Les particuliers qui utilisent des smartphones et des tablettes pour diverses activités telles que la communication, le divertissement, et la productivité.



- Développeurs d'applications : Les professionnels de développement qui créent des applications pour la plateforme Android, profitant de sa large base d'utilisateurs et de son écosystème riche.



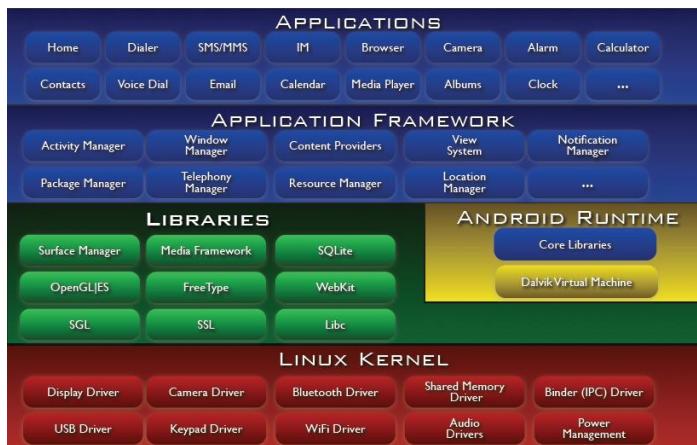
- Fabricants de dispositifs : Les entreprises qui produisent des appareils mobiles utilisant Android comme système d'exploitation pour leurs produits.

Versions

- Historique :
 - Crée en 2005 par la société Android
 - Rachat en 2007 par Google
 - 20 versions depuis la 1.0 (Apple Pie) en 2008 jusqu'à la 14.0 en 10/2023.
 - Une version = 1 API
 - Apparition de nouvelles fonctionnalités
 - Modification de fonctionnalités existantes
 - Disparition de certaines fonctionnalités

version	Nom de code	API
4.0	Ice Cream Sandwich	15
4.1	Jelly Bean	16
4.2	Jelly Bean	17
4.3	Jelly Bean	18
4.4	Kitkat	19
5.0	Lollipop	21
5.1	Lollipop	22
6.0	Marshmallow	23
7.0	Nougat	24
7.1	Nougat	25
8.0	Oreo	26
8.1	Oreo	27
9.0	Pie	28
10.0	Q (Quince Tart)	29
11.0	R (Red Velvet Cake)	30
12.0	S (Snow Cone)	31-32
13.0	T (Tiramisu)	33
14.0	U(UpSide Down Cake)	34
15.0	V(Vanilla Ice Cream)	35

Architecture d'Android



Architecture d'Android

1. Linux Kernel (Noyau Linux)

- Android repose sur un noyau Linux, qui assure la gestion des processus, de la mémoire, des pilotes matériels et de la sécurité.
- Il inclut des modules spécifiques à Android comme **Binder** (pour la communication entre processus) et **Power Management** (gestion de l'énergie).

2. Hardware Abstraction Layer (HAL)

- Fournit des interfaces pour interagir avec le matériel (caméra, capteurs, audio, Bluetooth...).

3. Android Runtime (ART) et Bibliothèques natives

- **ART (Android Runtime)** : Machine virtuelle qui exécute les applications Android en compilant le bytecode en code natif lors de l'installation (**AOT - Ahead Of Time**).
- **Bibliothèques natives** : Inclut des composants C/C++ tels que OpenGL (rendu graphique), SQLite (gestion de bases de données), WebKit (moteur de navigation).

Architecture d'Android

4. Framework d'application

- Fournit des API pour les développeurs d'applications.
- Contient des gestionnaires essentiels comme :
 - **Activity Manager** (gestion du cycle de vie des applications),
 - **Window Manager** (gestion des fenêtres et de l'affichage),
 - **Content Providers** (partage de données entre applications),
 - **Location Manager** (gestion des services de localisation),
 - **Package Manager** (gestion des applications installées).

5. Applications utilisateur

- Couches visibles pour l'utilisateur, incluant les applications préinstallées (Google Chrome, Messages, Play Store) et celles téléchargées depuis Google Play.

Architecture d'Android

Android est conçu par des développeurs pour des développeurs.

Android nécessite aucune certification pour devenir développeur.

Avec **Android**, on peut distribuer et monétiser des applications.

Puissant et intuitif, **Android** facilite le développement mobile.

Ouvert, un SDK simple et puissant. L'absence de coût de licence attire plus de développeur.

Facilité d'accès au matériel de bas niveau avec une série d'API.

Android fonctionne sur plusieurs marques de Smartphones.

Intégration de Google Maps et utilisation de services d'arrière plan.

Android et JAVA

Android ressemble à JAVA mais l'en est pas. Les applications sont écrites par JAVA mais ne sont pas interprétées par la machine virtuelle Java ME.

Les applications Java ne fonctionnent pas nativement sous **Android**.

Les applications **Android** sont exécutées par une machine virtuelle spécifique Dalvik (ART) et non par une JVM classique.

Android inclus une implémentation Open source spécifique de java basée non pas sur **OpenJdk**, mais sur le **projet Harmony** (implémentation java 5 de la fondation apache).

Google a adapté **Harmony** et supprimé certains packages non nécessaires en environnement mobile.

Les étapes du développement d'une application Android

1. Développement	2. Test	3. Publication
Android Studio	Émulateur ou Appareil mobile	Google Play

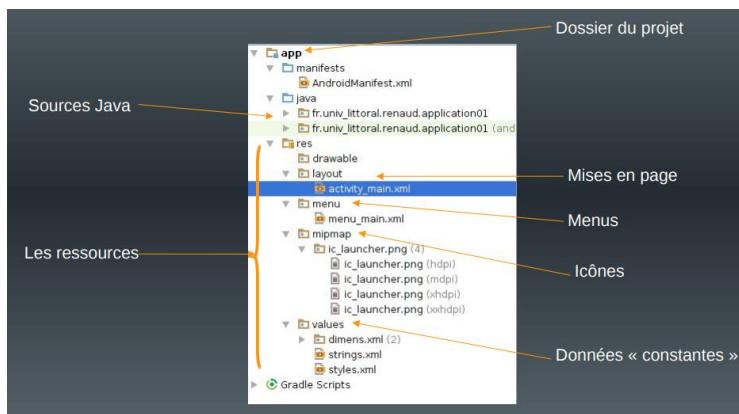
Android Studio

- ❖ C'est l'environnement de développement intégré (IDE) officiel pour le développement d'applications Android.
- ❖ Il inclut un émulateur Android, des outils de débogage et de test, ainsi qu'un éditeur de code avancé.
- ❖ Android Studio s'appuie sur IntelliJ IDEA et inclut le plug-in Android préinstallé, ainsi que quelques ajustements spécifiquement conçus pour la plate-forme Android.

Android Studio: Installation

1. Télécharger Android Studio depuis le site officiel de Google.
[\(https://developer.android.com/studio?hl=fr\)](https://developer.android.com/studio?hl=fr)
2. Installer l'IDE et le SDK Android.
3. Configurer un émulateur ou connecter un appareil physique.

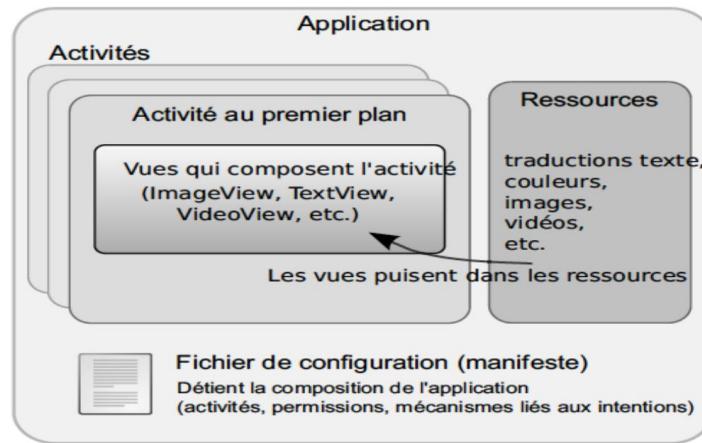
Structure d'un projet



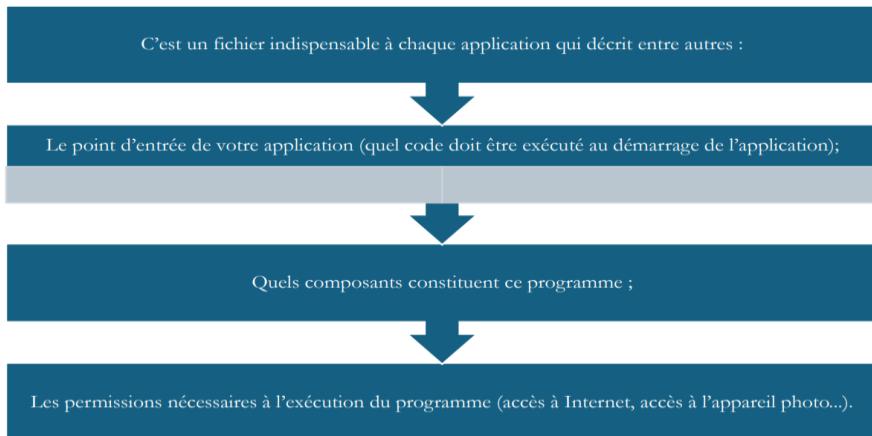
Structure d'un projet

- ❖ `AndroidManifest.xml` : définit des informations essentielles sur votre application, comme le nom du package, les composants (activités, services, etc.) et les autorisations.
- ❖ `java/` : contient les fichiers source
- ❖ `res/` :
 - ❖ contient les ressources (images, descriptions d'interfaces, valeurs) .
 - ❖ Organisé en sous dossiers par types (drawable, layout, mipmap, values, ...)
- ❖ `Gradle Scripts :`
 - ❖ Scripts de compilation
 - ❖ `build.gradle` : Fichier de configuration de Gradle, qui gère les dépendances et les paramètres de build de l'application

Application Android : Composition



Application Android : Fichier de configuration



Exemple : AndroidManifest.xml

```

<?xml version="1.0" encoding="utf-8" ?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.hp.myapplication">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="My Application"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>

</manifest>
  
```

Quiz

Dans quelle couche se trouvent les API Frameworks ?

- Application
- Runtime d'Android
- Bibliothèques
- Noyau Linux

Quel composant de la structure de projet Android est utilisé pour gérer les dépendances ?

- AndroidManifest.xml
- build.gradle
- MainActivity.java
- res/values/strings.xml

Quiz

Qu'est-ce qu'un fichier "res/layout/activity_main.xml" ?

- Fichier de configuration
- Fichier de code source
- Fichier de layout (mise en page)
- Fichier de ressources de chaîne de caractères

Quel fichier contient les autorisations requises par l'application ?

- AndroidManifest.xml
- build.gradle
- MainActivity.java
- res/values/strings.xml

Quiz

Quelle est la fonction de l'Activity dans Android ?

- Gérer les services en arrière-plan
- Gérer les bases de données locales
- Gérer les transactions réseau
- Gérer l'interface utilisateur



TP1

Création d'un Nouveau Projet

1. Ouvrez Android Studio et cliquez sur "Start a new Android Studio project".
2. Sélectionnez "Empty Views Activity" et cliquez sur "Next".
3. Remplissez les champs suivants :
 - Name : Nom de votre application (ex. : MonPremierApp).
 - Package name : Nom de votre package (ex. : com.example.monpremierapp).
 - Save location : Emplacement où vous souhaitez enregistrer votre projet.
 - Language : Java.
 - Minimum API level : Sélectionnez l'API minimum que vous souhaitez supporter.
4. Cliquez sur "Finish".

Exécution de l'Application

1. Cliquez sur l'icône de l'émulateur dans la barre d'outils pour lancer un émulateur Android.
2. Sélectionnez un appareil virtuel existant ou créez-en un nouveau.
3. Cliquez sur le bouton "Run" (icône verte en forme de flèche) pour compiler et exécuter l'application sur l'émulateur.

Modification du Contenu de l'Activité

- Ouvrez le fichier **activity_main.xml**.
- Remplacez le contenu actuel par le code suivant pour afficher un message personnalisé :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bonjour, bienvenue dans mon application !"
        android:textSize="24sp"
        android:textColor="@android:color/black"/>
</LinearLayout>
```

Modification du Contenu de l'Activité

- Ouvrez le fichier **MainActivity.java**.
- Ajoutez le code suivant pour initialiser et afficher le message dans votre activité :

```
// Java
package com.example.monpremierapp;

import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView textView = findViewById(R.id.textView);
        textView.setText("Bonjour, bienvenue dans mon application !");
    }
}
```

Exécution de l'Application

1. Cliquez sur l'icône de l'émulateur dans la barre d'outils pour lancer un émulateur Android.
2. Sélectionnez un appareil virtuel existant ou créez-en un nouveau.
3. Cliquez sur le bouton "Run" (icône verte en forme de flèche) pour compiler et exécuter l'application sur l'émulateur.



LES ACTIVITÉS

Définition

Contrairement à une application classique, une application Android est découpée en écran

Le système Android contrôle les activités, pas le développeur

Une activité est une classe qui hérite de Activity ou d'une classe dérivée de Activity

Chaque écran est géré par une activité (Activity)

Aucune méthode main dans un programme Android

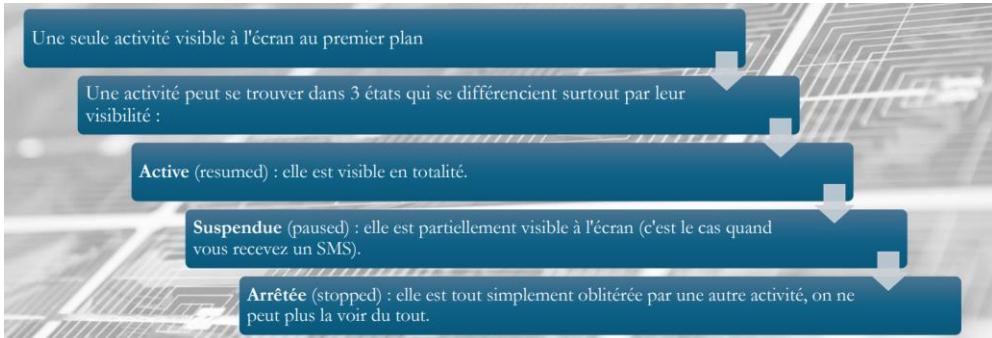
Le code d'une activité réagit à des évènements du système Android

Définition

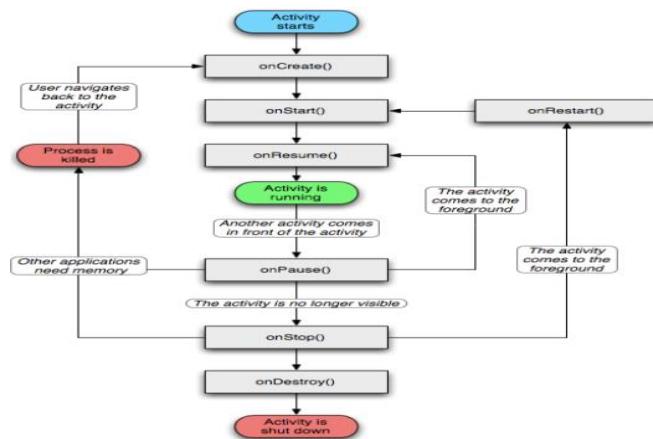
Activity = Java or Kotlin Class which return a graphical user interface called Layout

Activity without layout is called Service

Définition

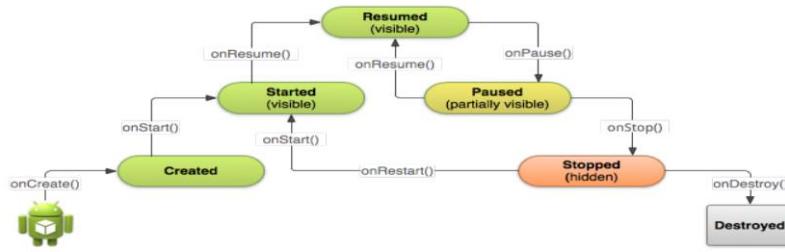


Cycle de vie



Cycle de vie

Le cycle de vie d'une activité (*activity life cycle*) correspond à l'ensemble des états qu'une activité peut avoir pendant toute sa durée de vie, de sa création à sa destruction. Lorsqu'un utilisateur navigue dans une application, les activités passent d'un état à un autre au cours de leur cycle de vie.



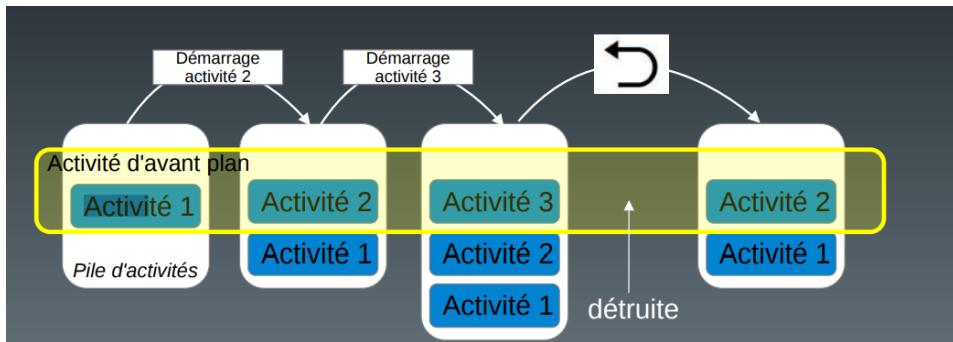
Cycle de vie

Activity.onXXX()

Les méthodes appelées lors des changements d'état :

- **onCreate(Bundle savedInstanceState)** : appelée à la création. Le paramètre permet de récupérer un état sauvegardé lors de l'arrêt de l'activité (si on a fait une sauvegarde)
- **onStart()** : appelée quand l'activité démarre
- **onRestart()** : appelée quand l'activité redémarre
- **onResume()** : appelée quand l'activité vient en premier plan
- **onPause()** : appelée quand l'activité n'est plus en premier plan
- **onStop()** : appelée quand l'activité n'est plus visible
- **onDestroy()** : appelée quand l'activité se termine
- On redéfinie les méthodes que l'on veut et on oublie pas d'appeler en premier **super.onXXX()**

Empilement des activités



Développement

- Une classe java par activité ;
- Les ressources associées (layout, menu, etc.) ;
- La classe hérite de la classe AppCompatActivity ;
- Génération d'un code minimum par défaut sous Android Studio.

```

...
public class Bidon extends AppCompatActivity {
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.Bidon);
        ...
        // initialisation des variables internes
    }
}
...
}
```



TP 2

Énoncé

Objectif :

Créer une application simple en Android qui explore les différents états du cycle de vie d'une activité (onCreate, onStart, onResume, onPause, onStop, onDestroy).

Étapes du TP :

1. Créer un nouveau projet Android
2. Modifier le fichier layout
- 3.Modifier le code de MainActivity
4. Exécuter l'application

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cycle de vie d'Android\nBonjour depuis MainActivity !"
        android:textSize="20sp"
        android:textColor="@android:color/holo_red_dark" />
</LinearLayout>
```

MainActivity.java

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

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("CycleDeVie", "onCreate appelé");
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.d("CycleDeVie", "onStart appelé");
    }
}
```

MainActivity.java

```

@Override
protected void onResume() {
    super.onResume();
    Log.d( tag: "CycleDeVie", msg: "onResume appelé");
}

@Override
protected void onPause() {
    super.onPause();
    Log.d( tag: "CycleDeVie", msg: "onPause appelé");
}

@Override
protected void onStop() {
    super.onStop();
    Log.d( tag: "CycleDeVie", msg: "onStop appelé");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d( tag: "CycleDeVie", msg: "onDestroy appelé");
}
}

```

Modification du label et du thème dans le fichier AndroidManifest.xml

Vous devez effectuer les modifications suivantes :

1. Remplacez le label par un nouveau label, par exemple : "MyFirstApp".
2. Changez le thème actuel par un thème personnalisé, tel que :
 - @style/Theme.AppCompat.Light.
 - @style/Theme.AppCompat.DayNight
 - @style/Theme.AppCompat.DayNight.Dialog
3. Vérifiez que le nouveau label "MyCustomApp" apparaît dans le lanceur d'applications.
4. Observez les changements du thème



LES RESSOURCES

PROGRAMMATION MOBILE(FAGROUD)

77

Architecture d'Android

Le répertoire res/ contient toutes les ressources qui seront mises dans le fichier application (apk). Il est constitué de sous répertoires :

drawable (images)	layout (description en XML des interfaces)	values (définitions en XML de constantes (avec i18n) : string, color, dim, style...)	anim (description en XML d'animations)
menus (description en XML de menus pour l'application)	xml (fichiers XML utilisés directement par l'application)	raw (tous les autres types de ressources : sons, vidéos, ...)	On peut ajouter d'autres sous répertoires

Exemple : internationalization (i18n)

- Valeur par défaut:

dans res/values/string.xml

```
<resources>
    <string name="helloworld">Hello World</string>
</resources>
```

- Valeur pour le français:

dans res/values-fr/string.xml

```
<resources>
    <string name="helloworld">Bonjour monde</string>
</resources>
```

La classe R

C'est une classe générée par Android Studio qui permet à l'application d'accéder aux ressources

Elle contient des classes internes dont les noms correspondent aux types de ressources (id, drawable, layout ...)

Elle est constituée à partir des fichiers placés dans les sous répertoires du répertoire **res/**

Une propriété est créée pour :

- Chaque image placée dans drawable
- Chaque identificateur défini dans des fichiers XML (objets d'interface, constantes)
- Chaque fichier placé dans les répertoires xml , raw ...

Référence aux ressources

- Une ressource dans un fichier XML d'une autre ressource:
@*[paquetage:]*type/nomDeLaRessource
par exemple @string/helloworld
- Ressources **système** en utilisant le paquetage android
(par ex. @android:color/blue)
- Dans un fichier source Java, par une constante (int) générée dans le fichier R.java, par ex. **R.string.helloworld**
- Utilisation de Resources context.getResources()
(Activity hérite de Context) pour obtenir un getter de ressources
- Quelques exemples :
R.id.loginButton, getString(R.string.helloworld),
getColor(R.color.my_nice_color), getLayout(R.layout.activity_layout),
getDimension(R.dimen.world_width),
getDrawable(R.drawable.card_picture),
openRawResource(R.raw.bindata) (retourne un InputStream)

LES VUES



Vues

- ❖ Une "vue" (ou "View" en anglais) est l'élément de base de l'interface utilisateur
- ❖ Permettent de définir les composants visibles et interactifs d'une application, tels que les boutons, les champs de texte, les images, etc.
- ❖ Chaque vue correspond à un composant individuel dans l'interface utilisateur.

Vues : de base

TextView :

- ❖ Affiche du texte à l'écran.
- ❖ Peut être stylisée avec différentes polices, couleurs et tailles de texte.

```
<TextView  
    android:id="@+id/textView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Bonjour le monde!" />
```

Vues : de base

Button :

- ❖ Représente un bouton cliquable.
- ❖ Peut déclencher des actions lorsqu'il est cliqué.

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cliquer" />
```

Vues : de base

ImageView :

- ❖ Affiche une image.
- ❖ Peut charger des images à partir de ressources locales ou d'URL.

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/ic_launcher_foreground" />
```

Vues : de base

EditText :

- ❖ Champ de saisie de texte.
- ❖ Permet à l'utilisateur d'entrer du texte.

```
<EditText  
    android:id="@+id/editText"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Entrer du texte" />
```

Vues : de base

Checkbox :

- ❖ Une case à cocher permet de sélectionner ou désélectionner une option.

```
<CheckBox  
    android:id="@+id/checkbox"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Option" />
```

Vues : de base

RadioButton:

- ❖ Permet à l'utilisateur de sélectionner une seule option parmi un groupe d'options.
- ❖ Il est souvent utilisé en combinaison avec un **RadioGroup**, qui regroupe plusieurs boutons radio pour garantir qu'une seule option peut être sélectionnée à la fois.

```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:id="@+id/radio_option1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Option 1" />

    <RadioButton
        android:id="@+id/radio_option2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Option 2" />

</RadioGroup>
```

Vues : de base

ProgressBar:

- ❖ Utilisé pour indiquer visuellement la progression d'une tâche ou d'un processus
- ❖ Elle peut être affichée de manière **indéterminée** (chargement en cours sans durée spécifique) ou **déterminée** (avec une progression mesurable).

```
<ProgressBar
    android:id="@+id/horizontal_progress_bar"
    style="@android:style/Widget.ProgressBar.Horizontal"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="100"
    android:progress="30" />
```

```
<ProgressBar
    android:id="@+id/progress_bar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:indeterminate="true"
    style="?android:attr/progressBarStyleLarge" />
```

Vues : de base

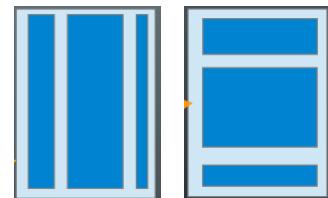
Autres:

- ❖ **CheckedTextView** : chaîne cochable (implante Checkable)
- ❖ **SeekBar**: barre de réglage
- ❖ **SearchView** : champ de recherche avec proposition de suggestions
- ❖ **ImageButton**: bouton avec image
- ❖ **VideoView**: affichage contrôlable de vidéo

Vues : Groupes

LinearLayout :

- ❖ Dispose les vues enfants en une seule ligne verticale ou horizontale.



```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <!-- Vues enfants ici -->
</LinearLayout>
```

Vues : Groupes

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="First TextView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="A Button" />
</LinearLayout>
```

Vues : Groupes

RelativeLayout :

- ❖ Dispose les vues enfants en relation les unes avec les autres ou avec le parent.

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!-- Vues enfants ici -->
</RelativeLayout>
```

Vues : Groupes

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView"
        android:text="I am a TextView" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I'm Below"
        android:layout_below="@+id/textView" />
</RelativeLayout>
```

Vues : Groupes

ConstraintLayout :

- ❖ Dispose les vues enfants avec des contraintes relatives les unes aux autres et au parent.
- ❖ Plus flexible et performant que RelativeLayout.

```
<ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!-- Vues enfants ici -->
</ConstraintLayout>
```

Vues : Groupes

```
<ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello ConstraintLayout"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I'm Aligned"
        app:layout_constraintTop_toBottomOf="@+id/textView"
        app:layout_constraintStart_toStartOf="parent" />
</ConstraintLayout>
```

Vues : Groupes

TableLayout :

- ❖ Organise les vues en lignes et colonnes, comme une table.

```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
    <TableRow>
        <TextView android:text="Colonne 1" />
        <TextView android:text="Colonne 2" />
    </TableRow>
</TableLayout>
```

Vues : Groupes

GridLayout :

- ❖ disposition matricielle avec N colonnes et un nombre infini de lignes

```
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:rowCount="2"
    android:columnCount="2">
    <TextView android:text="Élément 1" />
    <TextView android:text="Élément 2" />
    <TextView android:text="Élément 3" />
    <TextView android:text="Élément 4" />
</GridLayout>
```

Vues : Groupes

FrameLayout :

- ❖ Disposition en haut à gauche en empilant les éléments

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:src="@drawable/image" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Texte superposé"
        android:layout_gravity="center" />
</FrameLayout>
```

TP 3



Énoncé 1

Objectif :

Créer une application Android qui affiche un message de bienvenue, un bouton, une image, et un champ de saisie de texte.

Étapes du TP :

1. Créer un nouveau projet Android
2. Modifier le fichier layout
- 3.Modifier le code de MainActivity
4. Exécuter l'application

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bienvenue dans l'application!"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"
        android:layout_below="@+id/textView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp" />
```

activity_main.xml

```
<EditText
    android:id="@+id/editText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Entrez du texte ici"
    android:layout_below="@+id/imageView"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp" />

<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cliquer"
    android:layout_below="@+id/editText"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp" />
```

MainActivity.java

```

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

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

        final TextView textView = findViewById(R.id.textView);
        final EditText editText = findViewById(R.id.editText);
        Button button = findViewById(R.id.button);

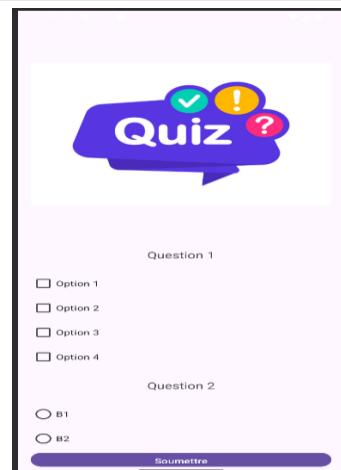
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String inputText = editText.getText().toString();
                textView.setText(inputText);
            }
        });
    }
}

```

Énoncé 2

Objectif :

- Créer une activité pour un quiz



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/l"/>
    <TextView
        android:id="@+id/questionTextView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Question 1"
        android:textSize="18sp"
        android:gravity="center"
        android:layout_marginBottom="20dp" />
    <CheckBox
        android:id="@+id/option1CheckBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 1" />
    <CheckBox
        android:id="@+id/option2CheckBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 2" />
    <CheckBox
        android:id="@+id/option3CheckBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 3" />
    <CheckBox
        android:id="@+id/option4CheckBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Option 4" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Question 2"
        android:textSize="18sp"
        android:gravity="center"
        android:layout_marginBottom="20dp" />
```

activity_main.xml

```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B1" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="B2" />
</RadioGroup>

<Button
    android:id="@+id/submitButton"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Soumettre" />
</LinearLayout>
```

MainActivity.java

```
package com.example.tp31;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.CheckBox;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

MainActivity.java

```
// Récupérer les éléments de l'interface utilisateur
TextView questionTextView = findViewById(R.id.questionTextView);
CheckBox option1CheckBox = findViewById(R.id.option1CheckBox);
CheckBox option2CheckBox = findViewById(R.id.option2CheckBox);
CheckBox option3CheckBox = findViewById(R.id.option3CheckBox);
CheckBox option4CheckBox = findViewById(R.id.option4CheckBox);
Button submitButton = findViewById(R.id.submitButton);

// Exemple de question et réponses
questionTextView.setText("Quels sont les langages de programmation ?");
option1CheckBox.setText("Java");
option2CheckBox.setText("HTML");
option3CheckBox.setText("C++");
option4CheckBox.setText("CSS");
```



GESTION DES ÉVÉNEMENTS

PROGRAMMATION MOBILE(FAGROUD)

111

Listeners

OnClickListener :

- ❖ Il est utilisé pour gérer les événements de clic sur des composants de l'interface comme les boutons.

```
Button monBouton = findViewById(R.id.submitButton);

monBouton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Code exécuté lorsque le bouton est cliqué
        Toast.makeText(getApplicationContext(), text: "Bouton cliqué!", Toast.LENGTH_SHORT).show();
    }
});
```

Listeners

OnLongClickListener

- ❖ Utilisé pour détecter un appui prolongé sur un composant.

```
monBouton.setOnLongClickListener(new View.OnLongClickListener() {
    no usages
    @Override
    public boolean onLongClick(View v) {
        // Code pour un clic long
        Toast.makeText(getApplicationContext(), text: "Clic long détecté!", Toast.LENGTH_SHORT).show();
        return true; // Renvoie true si l'événement est consommé
    }
});
```

Listeners

OnTouchListener

- ❖ Utilisé pour interagir avec des événements tactiles, comme les glissements ou les gestes.

```
TextView monTexte = findViewById(R.id.questionTextView);
monTexte.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            Toast.makeText(getApplicationContext(), "Élément touché!", Toast.LENGTH_SHORT).show();
        }
        return false; // false signifie que l'événement peut continuer à être traité par d'autres listeners
    }
});
```

Listeners

Autres:

- ❖ **OnKeyListener** : Pour capturer les événements liés aux touches du clavier, comme la touche "Entrée".
- ❖ **OnFocusChangeListener** : Pour réagir lorsque le focus d'un élément change.
- ❖ **Custom Listener** : Vous pouvez créer vos propres listeners pour gérer des événements spécifiques.

Gestion des événements de saisie utilisateur

❖ **addTextChangedListener** est une méthode utilisée dans le développement Android pour surveiller les modifications du texte dans un champ de type EditText.

❖ Elle utilise l'interface TextWatcher pour gérer les changements.

```
EditText editText = findViewById(R.id.myedittexte);

editText.addTextChangedListener(new TextWatcher() {
    @Usage
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        // Appelé avant que le texte ne soit modifié
    }

    @NoUsage
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        // Appelé lorsque le texte est en cours de modification
    }

    @Override
    public void afterTextChanged(Editable s) {
        // Appelé après que le texte ait été modifié
        Log.d("TextWatcher", msg: "Texte modifié : " + s.toString());
    }
});
```

Gestes et GestureDetector

Le GestureDetector est une classe qui permet de reconnaître des gestes courants effectués par l'utilisateur, comme :

- ❖ Tap (Appui simple)
- ❖ Double tap (Double appui)
- ❖ Scroll (Glissement ou défilement)
- ❖ Fling (Mouvement rapide)

Gestes et GestureDetector

- ❖ OnGestureListener est une interface dans Android utilisée pour détecter et gérer les gestes effectués par l'utilisateur sur l'écran tactile.
- ❖ Elle fournit des méthodes permettant de réagir à une variété de gestes, notamment les pressions, les glissements, et les appuis longs.
- ❖ fonctions principales :
 - **onDown(MotionEvent event)** : Appelée lorsqu'un appui initial est détecté (un simple contact sur l'écran).
 - **onShowPress(MotionEvent event)** : Appelée lorsqu'un appui est détecté et maintenu pendant un court instant, mais avant qu'il ne devienne un appui long.
 - **onSingleTapUp(MotionEvent event)** : Détecte un appui simple suivi du relâchement du doigt.
 - **onScroll(MotionEvent e1, MotionEvent e2, float distanceX, float distanceY)** : Détecte un mouvement glissé (scroll) entre deux points de contact.
 - **onLongPress(MotionEvent event)** : Appelée lorsqu'un utilisateur maintient son doigt appuyé sur l'écran sans le relâcher pendant un certain temps
 - **onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)** : Détecte un mouvement rapide et brusque effectué par l'utilisateur (glissement avec une forte vitesse).

Gestes et GestureDetector

```

public class MainActivity extends AppCompatActivity implements GestureDetector.OnGestureListener {

    2 usages
    private GestureDetector gestureDetector;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        gestureDetector = new GestureDetector(context: this, listener: this);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        return gestureDetector.onTouchEvent(event);
    }

    @Override
    public boolean onDown(MotionEvent event) {
        Log.d(msg: "Gestures", msg: "Appui simple détecté");
        return true;
    }

    @Override
    public boolean onSingleTapUp(@NonNull MotionEvent e) {
        return false;
    }

    @Override
    public boolean onScroll(@Nullable MotionEvent e1, @NonNull MotionEvent e2, float distanceX, float distanceY) {
        return false;
    }

    @Override
    public void onLongPress(@NonNull MotionEvent e) {
    }

    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
        Log.d(msg: "Gestures", msg: "Mouvement rapide détecté");
        return true;
    }
}

```

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

119



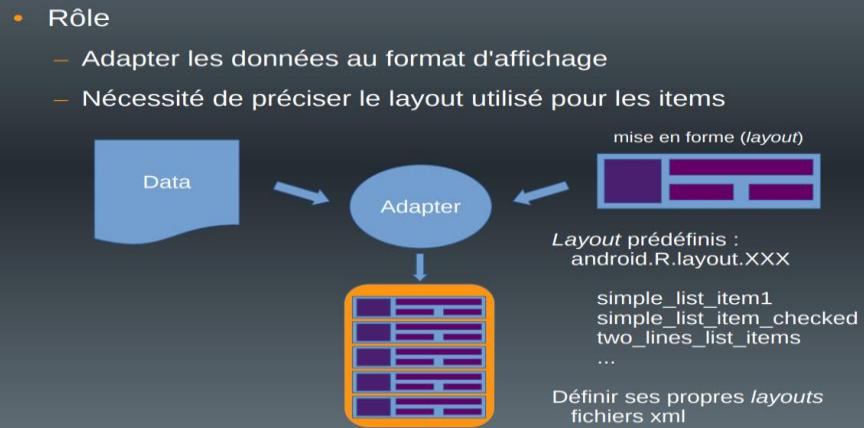
2024-2025

PROGRAMMATION MOBILE(FAGROUD)

120

LES ADAPTATEURS

Définition



Définition

Pourquoi les adaptateurs sont utiles :

- ❖ **Séparation des données et des vues** : Permet de manipuler les données sans affecter directement les composants de l'interface utilisateur.
- ❖ **Réutilisation efficace des ressources** : Les vues sont recyclées (notamment avec RecyclerView) pour améliorer les performances.
- ❖ **Personnalisation flexible** : Les adaptateurs personnalisés permettent de gérer des mises en page complexes et des interactions spécifiques.

Adaptateurs principaux

ArrayAdapter : Utilisé pour connecter des tableaux ou des listes simples à une vue comme une ListView.

```
// Récupération de la ListView
ListView listView = findViewById(R.id.listView);

// Tableau de données
String[] fruits = {"Pomme", "Banane", "Orange", "Mangue", "Ananas", "Fraise"};

// Création de l'ArrayAdapter
ArrayAdapter<String> adapter = new ArrayAdapter<>(context: this, android.R.layout.simple_list_item_1, fruits);

// Association de l'ArrayAdapter à la ListView
listView.setAdapter(adapter);
```

Adaptateurs principaux

Le **SimpleAdapter** est utilisé pour connecter des collections de données plus complexes (comme une liste de HashMap) aux vues.

```
// Récupération de la ListView
ListView listView = findViewById(R.id.listView);

// Liste de données (chaque élément est une HashMap)
List<Map<String, Object>> dataList = new ArrayList<>();

// Tableau des noms des fruits
String[] fruits = {"Pomme", "Banane", "Orange", "Mangue", "Ananas"};
int[] images = {R.drawable.pomme, R.drawable.banane, R.drawable.orange, R.drawable.mangue, R.drawable.ananas};

// Remplir la liste avec les données
for (int i = 0; i < fruits.length; i++) {
    Map<String, Object> item = new HashMap<>();
    item.put("name", fruits[i]); // Clé "name" pour le texte
    item.put("image", images[i]); // Clé "image" pour l'icône
    dataList.add(item);
}

// Définition des clés et des vues correspondantes
String[] from = {"image", "name"}; // Clés utilisées dans le HashMap
int[] to = {R.id.imageView, R.id.textView}; // ID des vues dans item_layout.xml

// Créeation de l'adaptateur SimpleAdapter
SimpleAdapter adapter = new SimpleAdapter(this, dataList, R.layout.item_layout, from, to);

// Associer l'adaptateur à la ListView
listView.setAdapter(adapter);
```

Adaptateurs principaux

Le **CursorAdapter** est une classe utilisée dans Android pour relier des données provenant d'une base de données (via un Cursor) à une vue comme une ListView ou une GridView

```
Cursor cursor = database.rawQuery("SELECT * FROM produits", null);
2 usages
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
    android.R.layout.simple_list_item_1, cursor,
    new String[]{"nom_produit"}, new int[]{android.R.id.text1}, 0);
listView.setAdapter(adapter);
```

Adaptateurs

Custom Adapter (adaptateur personnalisé) :

- Créé lorsqu'un adaptateur par défaut (comme ArrayAdapter) ne répond pas aux besoins spécifiques

RecyclerView.Adapter :

- Spécifiquement conçu pour le composant RecyclerView. Il offre une gestion plus performante, surtout pour de grandes quantités de données.
- Nécessite la création d'une classe ViewHolder et la surcharge de méthodes comme onCreateViewHolder, onBindViewHolder et getItemCount.

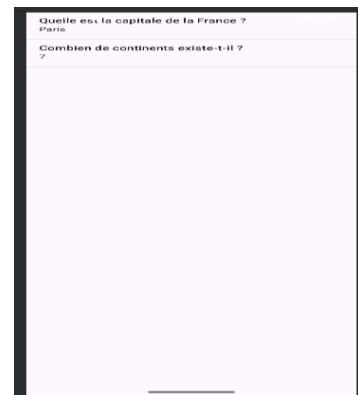


TP 4

Énoncé

Objectif :

- Créer une application Android qui affiche une liste de questions de quiz



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ArrayList<HashMap<String, String>> quizList = new ArrayList<>();
    HashMap<String, String> question1 = new HashMap<>();
    question1.put("question", "Quelle est la capitale de la France ?");
    question1.put("answer", "Paris");
    quizList.add(question1);

    HashMap<String, String> question2 = new HashMap<>();
    question2.put("question", "Combien de continents existe-t-il ?");
    question2.put("answer", "7");
    quizList.add(question2);

    ListView listView = findViewById(R.id.listView);

    SimpleAdapter adapter = new SimpleAdapter(
        context, this,
        quizList,
        android.R.layout.simple_list_item_2,
        new String[]{"question", "answer"},
        new int[]{android.R.id.text1, android.R.id.text2}
    );
    listView.setAdapter(adapter);
}
```



2024-2025

PROGRAMMATION MOBILE(FAGROUD)

131

DIALOGUE AVEC L'UTILISATEUR

Toast

- ❖ Permet d'afficher une courte notification à l'utilisateur sans interaction.
- ❖ S'efface automatiquement après quelques secondes.
- ❖ Idéale pour afficher des messages simples.

```
Toast.makeText(getApplicationContext(), text: "Message Toast affiché", Toast.LENGTH_SHORT).show();
```

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

132

Snackbar

- ❖ Le Snackbar est un message temporaire similaire au Toast mais avec la possibilité d'ajouter une action (bouton).
- ❖ Idéal pour offrir des retours d'information ou permettre des actions rapides.

```
Snackbar.make(findViewById(android.R.id.content), text: "Message avec Snackbar", Snackbar.LENGTH_SHORT)
    .setAction( text: "Répéter", v -> {
        // Logique lorsque l'utilisateur clique sur le bouton "Répéter"
        Toast.makeText(getApplicationContext(), text: "Action répétée", Toast.LENGTH_SHORT).show();
    })
    .show();
```

AlertDialog

- ❖ AlertDialog affiche une boîte de dialogue avec des options, comme des boutons pour interagir avec l'utilisateur.
- ❖ Idéal pour des alertes ou des choix simples.

```
AlertDialog.Builder builder = new AlertDialog.Builder( context: this );
builder.setTitle("Titre de l'alerte");
builder.setMessage("Voulez-vous continuer ?");
builder.setPositiveButton( text: "Oui", (dialog, which) -> {
    // Logique pour le bouton "Oui"
    Toast.makeText( context: this, text: "Appuyé sur Oui", Toast.LENGTH_SHORT).show();
});
builder.setNegativeButton( text: "Non", (dialog, which) -> {
    // Logique pour le bouton "Non"
    dialog.dismiss();
});
builder.show();
```

DialogFragment

- ❖ DialogFragment est une alternative avancée et modulaire à l'AlertDialog, qui est géré comme un Fragment.
- ❖ Idéal pour des dialogues personnalisés ou réutilisables.

CustomDialogFragment.java

```
public class CustomDialogFragment extends DialogFragment {
    ...
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        ...
        AlertDialog.Builder builder = new AlertDialog.Builder(requireActivity());
        builder.setTitle("Confirmation");
        builder.setMessage("Voulez-vous continuer ?");
        builder.setPositiveButton("Oui", (dialog, which) -> {
            ...
            Toast.makeText(getApplicationContext(), "Action confirmée", Toast.LENGTH_SHORT).show();
        });
        builder.setNegativeButton("Non", (dialog, which) -> {
            ...
            Toast.makeText(getApplicationContext(), "Action annulée", Toast.LENGTH_SHORT).show();
        });
        return builder.create();
    }
}
```

main_activity.java

```
Button buttonShowDialog = findViewById(R.id.button_show_dialog);
buttonShowDialog.setOnClickListener(v -> {
    ...
    // Afficher le DialogFragment
    CustomDialogFragment dialog = new CustomDialogFragment();
    dialog.show(getSupportFragmentManager(), "CustomDialog");
});
```

main_activity.xml

```
<Button
    android:id="@+id/button_show_dialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Afficher le DialogFragment"
    android:layout_gravity="center" />
```

Notifications système

- ❖ Les notifications sont un moyen de dialoguer avec l'utilisateur même si l'application est en arrière-plan.
- ❖ Elles servent à transmettre des informations importantes ou à engager l'utilisateur.

```
Tutoriel
private void showNotification() {
    String channelId = "CHANNEL_ID";
    String channelName = "Canal de notification";

    // Création du canal de notification (pour API 26+)
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel(
            channelId,
            channelName,
            NotificationManager.IMPORTANCE_DEFAULT
        );
        NotificationManager manager = getSystemService(NotificationManager.class);
        if (manager != null) {
            manager.createNotificationChannel(channel);
        }
    }

    // Création de la notification
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, channelId)
        .setSmallIcon(R.drawable.ic_notification)
        .setContentTitle("Titre de la notification")
        .setContentText("Voici le contenu de la notification.")
        .setPriority(NotificationCompat.PRIORITY_HIGH)
        .setAutoCancel(true);

    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);
    notificationManager.notify(1, builder.build());
}
```

Quand utiliser chaque outil ?

- ❖ **Toast** : Messages rapides et non-intrusifs (exemple : "Enregistré avec succès").
- ❖ **Snackbar** : Messages contextuels avec actions (exemple : "Message supprimé. Annuler ?").
- ❖ **AlertDialog** : Interaction nécessitant une réponse de l'utilisateur (exemple : "Voulez-vous quitter cette page ?").
- ❖ **DialogFragment** : Dialogues personnalisés pour des interactions complexes ou réutilisables.
- ❖ **Notifications système** : Communication avec l'utilisateur en arrière-plan ou hors application.



CHANGEMENTS D'ACTIVITÉS

Définition

❖ Les **intents** sont un outil clé dans Android pour naviguer entre activités ou communiquer entre composants.

❖ Exceptions liées aux intents

Dans le cas où l'objet intent n'est pas bien paramétré il peut causer des exceptions de type :

- ❖ android.content.ActivityNotFoundException: No Activity found to handle Intent
- ❖ Ceci veut dire qu'aucune activité ne peut prendre en considération cet objet.

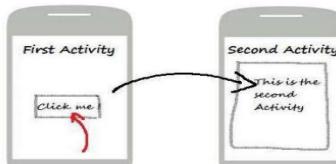
Intent explicites

❖ Ils sont utilisés pour passer d'une activité à une autre dans la même application.

❖ Utilisés pour spécifier précisément l'activité cible à lancer.

```
// Explicit Intent by specifying its class name
Intent i = new Intent(FirstActivity.this, SecondActivity.class);

// Starts TargetActivity
startActivity(i);
```





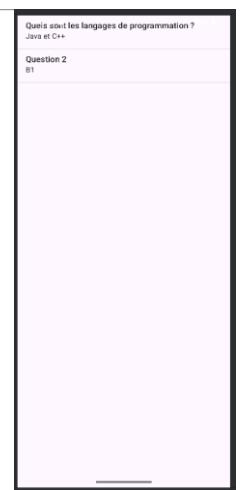
TP 5

Énoncé

Objectif:

Création d'une application avec plusieurs activités :

- Une activité principale contenant un Quiz
- Une deuxième activité qui affiche la solution du même Quiz.



Page2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
</LinearLayout>
```

Page2.java

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

        ArrayList<HashMap<String, String>> quizList = new ArrayList<>();
        HashMap<String, String> question1 = new HashMap<>();
        question1.put("question", "Quels sont les langages de programmation ?");
        question1.put("answer", "Java et C++");
        quizList.add(question1);

        HashMap<String, String> question2 = new HashMap<>();
        question2.put("question", "Question 2");
        question2.put("answer", "B1");
        quizList.add(question2);

        ListView listView = findViewById(R.id.listView);

        SimpleAdapter adapter = new SimpleAdapter(
            context, this,
            R.layout.simple_list_item_2,
            new String[]{"question", "answer"},
            new int[]{android.R.id.text1, android.R.id.text2}
        );
        listView.setAdapter(adapter);
    }
}
```

Androidmanifest.xml

```

<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="@tp31"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Tp31"
    tools:targetApi="31">
    <activity
        android:name=".MainActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".pagedecorrection" />
</application>

```

Mainactivity.java

```

Button monBouton = findViewById(R.id.submitButton);

monBouton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Code exécuté lorsque le bouton est cliqué
        //Toast.makeText(getApplicationContext(), "Bouton cliqué!", Toast.LENGTH_SHORT).show();
        Intent intent = new Intent( packageContext: MainActivity.this, pagedecorrection.class);
        startActivity(intent);
    }
});

```

Amélioration



Passage de données entre activités

Vous pouvez transmettre des données en utilisant un Intent avec des extras :

➤ Dans l'activité de départ :

```
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("key", "value");
startActivity(intent);
```

➤ Dans l'activité cible, vous pouvez récupérer les données :

```
String value = getIntent().getStringExtra("key");
```



TP 6

Page2.java

```

EditText editText = findViewById(R.id.ed);
Button monBouton1 = findViewById(R.id.b1);

monBouton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Code exécuté lorsque le bouton est cliqué
        String data = editText.getText().toString();
        //Toast.makeText(getApplicationContext(), "Bouton cliqué!", Toast.LENGTH_SHORT).show();
        Intent intent1 = new Intent(getApplicationContext(), pagecorrection.this, feedback.class);
        intent1.putExtra("key_data", data);
        startActivity(intent1);
    }
});
```

Page3.xml + Page3.java

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="400dp"
        android:layout_height="wrap_content"
        android:id="@+id/TextView"
        tools:ignore="MissingConstraints"
        android:text=""
        android:layout_margin="70dp"
        android:background="@color/material_on_primary_emphasis_high_type"/>
</LinearLayout>
```

```
package com.example.tp31;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

public class feedback extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.feedback);
        Intent intent = getIntent();
        String receivedData = intent.getStringExtra("key_data");
        TextView textView = findViewById(R.id.TextView);
        textView.setText(receivedData);
    }
}
```

startActivityForResult()

- Utilisé lorsqu'une activité doit retourner un résultat à l'activité qui l'a appelée.
- Nécessite la méthode **onActivityResult()** pour traiter le résultat.
- Exemple :

```
Intent intent = new Intent(this, SecondActivity.class);
startActivityForResult(intent, REQUEST_CODE);
```

Pour recevoir le résultat :

```
@Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE && resultCode == RESULT_OK) {
        String result = data.getStringExtra("key");
    }
}
```



TP 7

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

153

Résultat attendue

The figure consists of three mobile phone screens connected by blue arrows pointing right, illustrating a user flow:

- Screen 1:** A white page with a purple header bar containing the text "Quiz1". Below the header, there is a box with the text: "Testez vos connaissances en programmation à travers ce quiz général couvrant les bases des langages, algorithmes, structures de données, et bonnes pratiques de codage. Plusieurs questions ont des réponses multiples pour confirmés !". At the bottom, there is a purple button labeled "Démarrer le Quiz" and the text "Votre score : 0".
- Screen 2:** A white page with a purple header bar containing the text "Quiz". Below the header, there is a question: "Quels sont les langages de programmation ?" followed by four options: "Java", "HTML", "C++", and "CSS", each with a small checkbox. Below the options, there is a section titled "Question 2" with two radio buttons labeled "B1" and "B2", and a "Soumettre" button at the bottom.
- Screen 3:** A white page with a purple header bar containing the text "Quiz1". Below the header, there is a box with the text: "Testez vos connaissances en programmation à travers ce quiz général couvrant les bases des langages, algorithmes, structures de données, et bonnes pratiques de codage. Plusieurs questions ont des réponses multiples pour confirmés !". At the bottom, there is a purple button labeled "Démarrer le Quiz", the text "Votre score : 2", and a purple button labeled "Voir la correction".

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

154

Pagehome.xml

```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="16dp"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp"
    android:layout_gravity="center">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="16dp">

        <TextView
            android:id="@+id/titleTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Quiz"
            android:textSize="18sp"
            android:textStyle="bold"
            android:gravity="center" />

        <TextView
            android:id="@+id/contentTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Testez vos connaissances en programmation à travers ce quiz général couvrant les bases des langages, algorithmes, structures de données et logique de programmation."/>
    

```

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <Button
        android:id="@+id/startQuizButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Démarrer le Quiz"/>

    <TextView
        android:id="@+id/scoreTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Votre score : 0"
        android:textSize="18sp"
        android:layout_marginLeft="40dp"
        android:layout_gravity="right"/>

```

`</LinearLayout>`

```
<Button
    android:id="@+id/correctionButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Voir la correction"
    android:visibility="gone"
    android:layout_marginTop="16dp"/>

```

`</LinearLayout>`
`</androidx.cardview.widget.CardView>`

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

155

Pagehome.java

```
public class homepage extends AppCompatActivity {

    2 usages
    private static final int QUIZ_REQUEST_CODE = 1;
    2 usages
    private TextView scoreTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.homepage);

        scoreTextView = findViewById(R.id.scoreTextView);
        Button startQuizButton = findViewById(R.id.startQuizButton);
        startQuizButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                startActivityForResult(intent, QUIZ_REQUEST_CODE);
            }
        });
    }
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    Button correctionButton = findViewById(R.id.correctionButton);
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == QUIZ_REQUEST_CODE && resultCode == RESULT_OK) {
        int score = data.getStringExtra("score", defaultValue: 0);
        scoreTextView.setText("Votre score : " + score);
    }
    // Affichage du bouton "Voir la correction"
    correctionButton.setVisibility(View.VISIBLE);
    correctionButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Code exécuté lorsque le bouton est cliqué
            // Toast.makeText(getApplicationContext(), "Bouton cliqué!", Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(getApplicationContext(), homepage.this, pageCorrection.class);
            startActivity(intent);
        }
    });
}
```

2024-2025

PROGRAMMATION MOBILE(FAGROUD)

156

MainActivity.java

```

    // Gestion du clic sur le bouton "Soumettre"
    submitButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            calculateScore();
            Intent resultIntent = new Intent();
            resultIntent.putExtra("name: score", score);
            setResult(RESULT_OK, resultIntent);
            finish();
        }
    });

    /**
     * Usage
     */
    private void calculateScore() {
        score = 0; // Réinitialisation du score

        // Vérification des bonnes réponses
        if (option1CheckBox.isChecked() && option1CheckBox.getText().toString().equals("Java")) {
            score++;
        }
        if (option3CheckBox.isChecked() && option3CheckBox.getText().toString().equals("C++")) {
            score++;
        }
        // Ajouter des vérifications similaires pour les autres cases si nécessaire
    }
}

```

Activity_main.xml

Modification 1

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">
        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/l1"/>
        <TextView
            android:id="@+id/questionTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Question 1"
            android:textSize="18sp"
            android:gravity="center"
            android:layout_marginBottom="20dp" />
        <CheckBox
            android:id="@+id/option1CheckBox"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Option 1" />
    


```

Modification 2

```

<!-- Bouton Soumettre -->
<Button
    android:id="@+id/submitButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Soumettre"
    android:layout_marginTop="16dp" />

</LinearLayout>
</ScrollView>

```

Pagedecorrection.java

```
Button monBouton2 = findViewById(R.id.b2);

monBouton2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        onBackPressed(); // Appelle le comportement du bouton retour
    }
});
```

Services

Les services en Android permettent d'exécuter des tâches en arrière-plan sans interaction directe avec l'utilisateur. Types de Services en Android:

Service standard (Service)

- S'exécute indépendamment jusqu'à ce qu'il soit arrêté.
- Exécution via startService().

Service au premier plan (Foreground Service)

- Affiche une notification persistante pendant son exécution.
- Exécution via startForeground().

Service lié (Bound Service)

- Permet à un composant (Activity, Fragment, etc.) de communiquer avec le service.
- Exécution via bindService().

TP 8



2024-2025

PROGRAMMATION MOBILE(FAGROUD)

161

TimerService.java

```
public class TimerService extends Service {
    ...
    private static final int TIMER_DURATION = 30000; // 30 seconds
    ...
    private CountDownTimer countDownTimer;
    ...
    private static final String CHANNEL_ID = "quiz_channel";
    ...
    @Override
    public void onCreate() {
        super.onCreate();
        createNotificationChannel();
        startTimer();
    }
    ...
    private void startTimer() {
        countDownTimer = new CountDownTimer(TIMER_DURATION, countDownInterval: 1000) {
            ...
            @Override
            public void onTick(long millisUntilFinished) {
                Log.d("Tag: TimerService", msg:"Temps restant : " + millisUntilFinished / 1000 + "s");
            }
            ...
            @Override
            public void onFinish() {
                sendNotification();
                stopSelf();
            }
        }.start();
    }
}
```

```
private void sendNotification() {
    NotificationCompat.Builder builder = new NotificationCompat.Builder(context, CHANNEL_ID)
        .setSmallIcon(R.drawable.icon)
        .setContentTitle("Temps écoulé !")
        .setContentText("Le temps du quiz est terminé")
        .setPriority(NotificationCompat.PRIORITY_HIGH)
        .setAutoCancel(true);

    NotificationManagerCompat manager = NotificationManagerCompat.from(context);
    if (ActivityCompat.checkSelfPermission(context, android.Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {
        // TODO: Consider calling
        // ActivityCompat#requestPermissions
        // here to request the missing permissions, and then overriding
        // public void onRequestPermissionsResult(int requestCode, String[] permissions,
        //                                            int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for ActivityCompat#requestPermissions for more details.
        return;
    }
    manager.notify(1, builder.build());
}
```

2024-2025
PROGRAMMATION MOBILE(FAGROUD)

162

TimerService.java

```

    ↑ usage
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "Quiz Channel";
        String description = "Notifications pour les quiz";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;

        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);

        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        if (notificationManager != null) {
            notificationManager.createNotificationChannel(channel);
        }
    }
}

no usages
@Override
public IBinder onBind(Intent intent) {
    return null;
}
}

```

Pagehome.java

```

scoreTextView = findViewById(R.id.scoreTextView);
Button startQuizButton = findViewById(R.id.startQuizButton);
startQuizButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intentS = new Intent(packageContext: homepage.this, TimerService.class);
        startService(intentS);
        Intent intent = new Intent(packageContext: homepage.this, MainActivity.class);
        startActivityForResult(intent, QUIZ_REQUEST_CODE);
    }
});
}

```

AndroidManifest.xml

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="Study APP"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.Tp31"
    tools:targetApi="31">
    <activity
        android:name=".toolbar"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".MainActivity" />
    <activity android:name=".homepage" />
    <activity android:name=".pagedecorrection" />
    <activity android:name=".feedback" />
    <service android:name=".TimerService"/>
</application>
```