Programmation Mobile

Anatomie d'une Application Android

2025 - 2026

Dr. Abdelkader Ouared

INTRODUCTION À ANDROID: PRÉSENTATION

Introduction à Android: Présentation

- **Qu'est ce qu'Android**
  - Une pile de solutions logicielles orientées tablettes et smartphone, proposée par Google, incluant:
    - Un OS (Operating System) basé sur le noyau Linux.
    - Un ensemble de drivers pour piloter les composants matériels (appareils photos, WIFI, accéléromètres,..).
    - Un ensemble de libraires accessibles via le langage de programmation Java/Kotlin.
    - Un ensemble d'applications basées ces libraires.

Introduction à Android: Présentation

- La start-up avait initialement pour objectif de développer des systèmes d'exploitation pour numérique caméras
- Le nom « Android » vient du nom de l'entreprise qui a initialement crée le système d'exploitation, avant son achat par Google
- Entièrement **personnalisable** sur le noyau Linux

Android Inc. a été fondée en **octobre 2003** par Andy Rubin, Nick Sears, Chris White et Rich Miner

Introduction à Android: Présentation

- **Android SDK (Software Development Kit)**
  - Un ensemble d'outils de développement qui permet aux développeurs de créer des apps Android.
  - Il comprend divers composants (par ex. **les bibliothèques** de code, les **émulateurs**, les **outils de débogage**, les **bibliothèques tierces**, les API **Android**).
  - L'Android SDK peut être utilisé avec n'importe quel environnement de développement, pas seulement Android Studio.

- **Android Studio :**
  - Environnement de développement officiellement recommandé par Google pour le développement d'applications Android. C'est un IDE (Integrated Development Environment) basé sur IntelliJ IDEA et spécialement conçu pour Android.

Android Studio, Android SDK, et Android OS

Élément	Type	Rôle principal
Android OS	Système d’exploitation	Fait tourner ton téléphone et exécute les apps
Android SDK	Ensemble d’outils pour développeurs	Permet de créer et compiler des applications Android
Android Studio	Environnement de développement (IDE)	Logiciel utilisé pour écrire du code et utiliser le SDK facilement

ARCHITECTURE

Pourquoi étudier l'architecture des applications Android ?

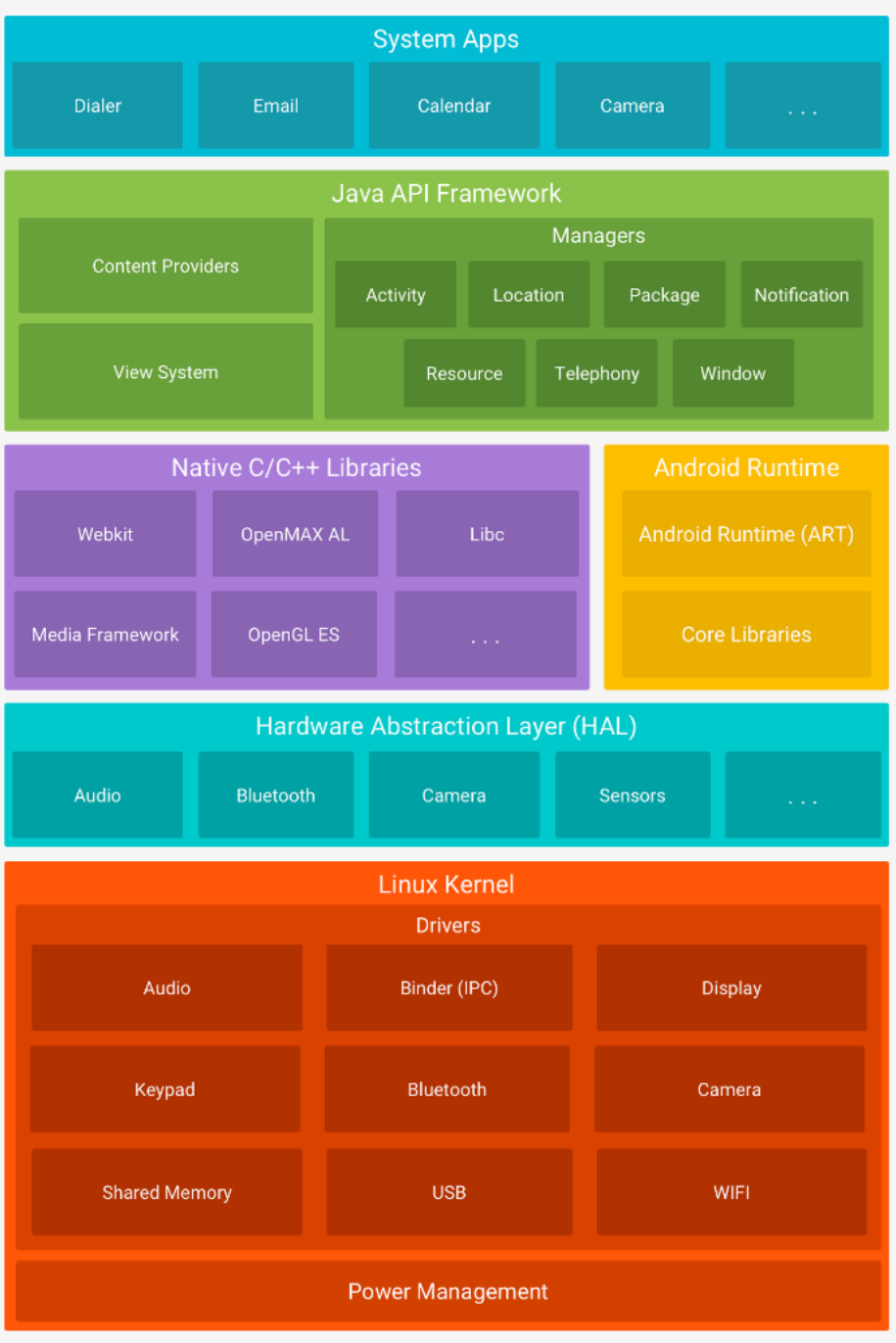
1. Comprendre la structure interne d'une application
2. Concevoir des applications robustes et performantes
    - Ex. d'Optimisation : Éviter que vos Services ou Activity tuent la batterie ou la mémoire
3. Comprendre l'intégration avec le système Android
4. Développer efficacement et déboguer plus facilement
5. Préparer à l'évolution du développement mobile

Architecture des applications Android

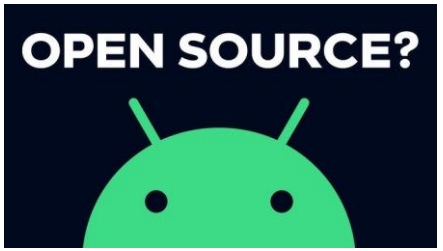
Google avait publié le code Android en open source sous licence Apache

Peut être modifié, si nécessaire

Pas de redevances ni de frais de licence



Android et l'Open Source



❑ **Google a publié Android en open source** dès 2008, sous la **licence Apache 2.0**.

→ Tout développeur peut **consulter, comprendre et améliorer** le code du système Android.

❑ **Apache License 2.0** :

- Autorise la modification, la distribution et l'usage libre du code ;
- Protège les droits des contributeurs ;
- Encourage l'innovation ouverte.

❑ **Android Open Source Project (AOSP)** :
Le projet officiel qui gère le code source d'Android :
https://source.android.com

❑ **Contribuer à Android**, c'est :

- Participer à l'un des plus grands projets open source au monde ;
- Apprendre les meilleures pratiques industrielles ;
- Rejoindre une communauté mondiale de développeurs.

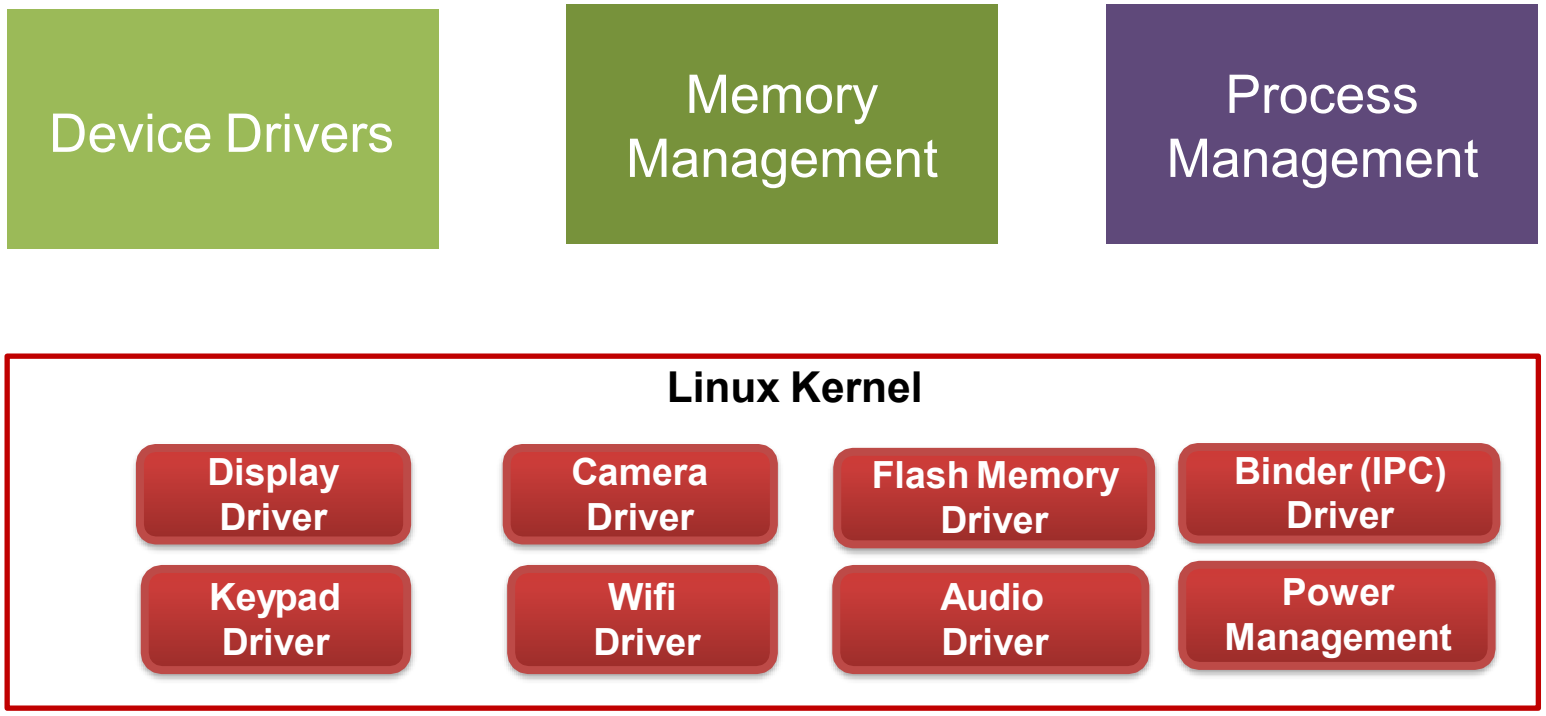
❑ **Code source sur GitHub (miroirs et projets liés)** :

- https://github.com/aosp-mirror
- https://github.com/android

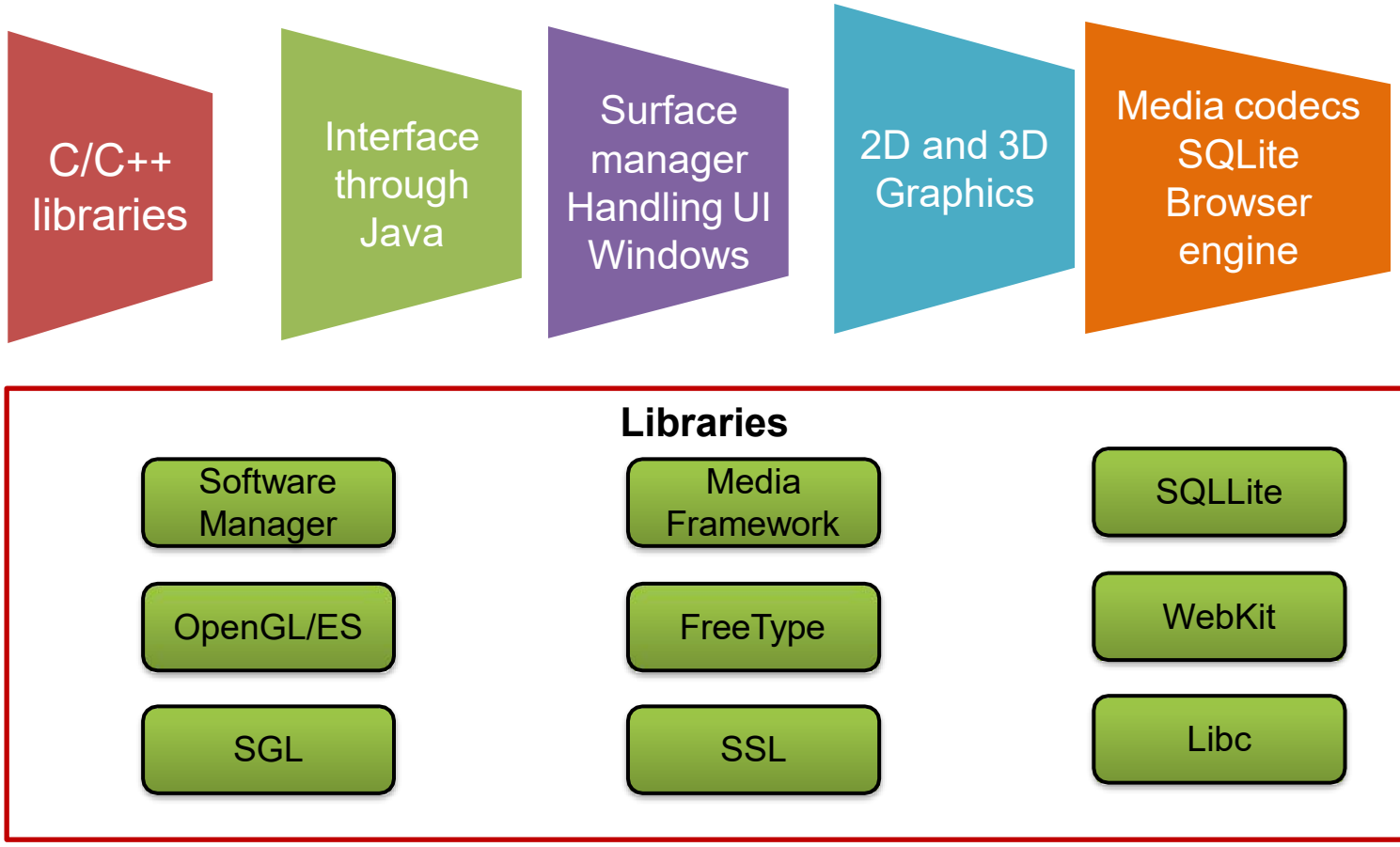
❑ **Contributions célèbres :**

- **JetBrains & Google** → développement du langage **Kotlin** ;
- **AndroidX** → librairies open source maintenues par Google ;
- Des milliers de **pull requests** d'étudiants, chercheurs, et développeurs du monde entier.

Architecture des applications Android: Linux Kernel

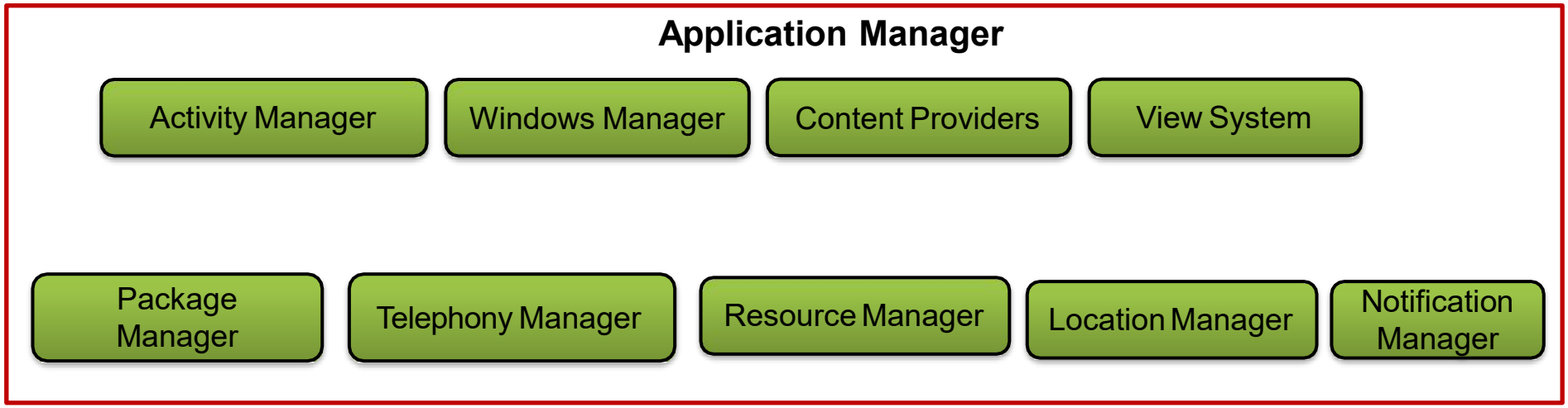


Architecture des applications Android: Libraries



Architecture des applications Android: Application Manager

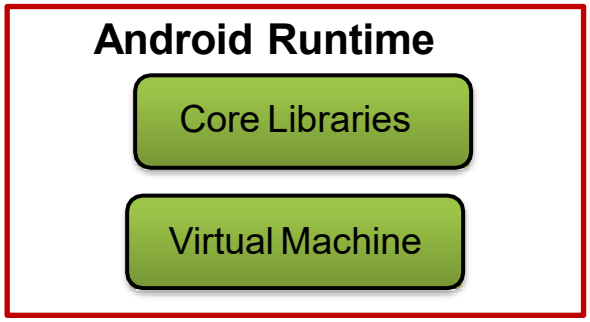
- **Interface API**
- **Gestionnaire d'activités** : gère le cycle de vie des applications



Architecture des applications Android: Android Runtime

- **Dalvik VM**
  - Fichiers compilés Dex (Dalvik Executable) spécifiques à l'environnement d'exécution Android
  - Compact et efficace par rapport aux fichiers de classe en termes de mémoire et de puissance de la batterie.

- **Core Libraries (libcore)**
  - un ensemble de bibliothèques Java incluses dans le runtime d'Android
    ex. java.lang, java.util, java.io

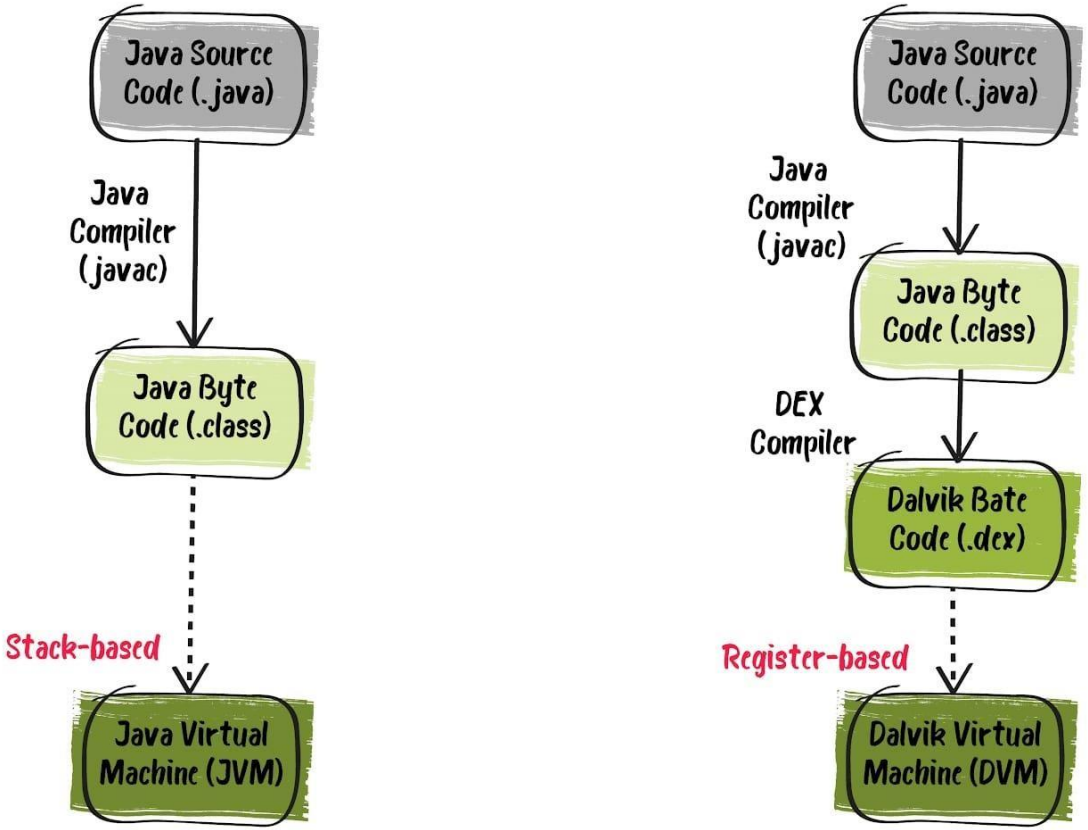


Architecture des applications Android: Android Runtime

Dalvik virtual machine: JVM optimisé pour les platformes mobiles

- **Dalvik n'est pas compatible Java SE**
  - Machine à processeurs (et non à pile, comme une JVM traditionelle)
- **.java => .class => .dex**
  - .dex = fichier en langage machine natif pour la VM Dalvik
  - .dex produit par le post-compilateur dx.
- **Dalvik est en interconnection directe avec le noyau Linux**

Architecture des applications Android:
Android Runtime

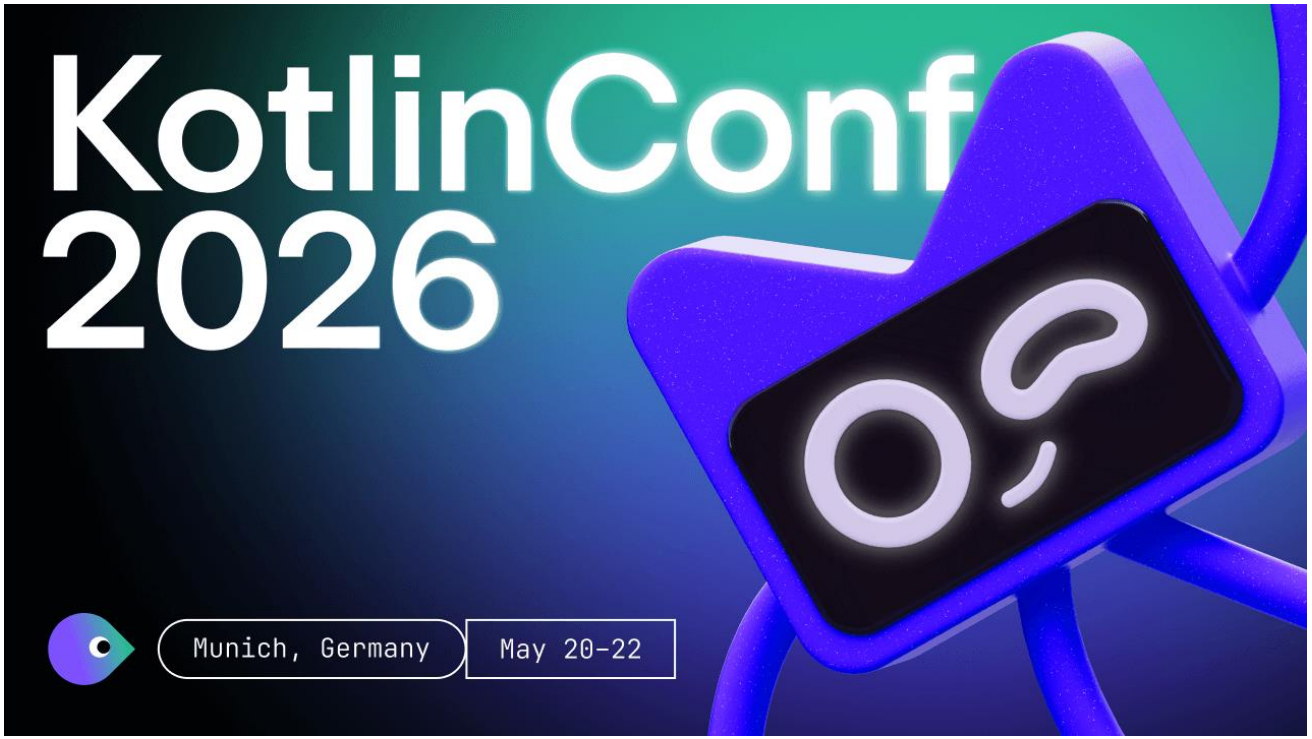


Question?

Quel langage parmi **Java**, **Python** et **Kotlin** est le plus **éco-responsable** pour développer une application mobile ?

KotlinConf is the official Kotlin conference by JetBrains

https://kotlinconf.com/



COMPOSANTES D'UNE APPLICATION ANDROID

Composante de l’application

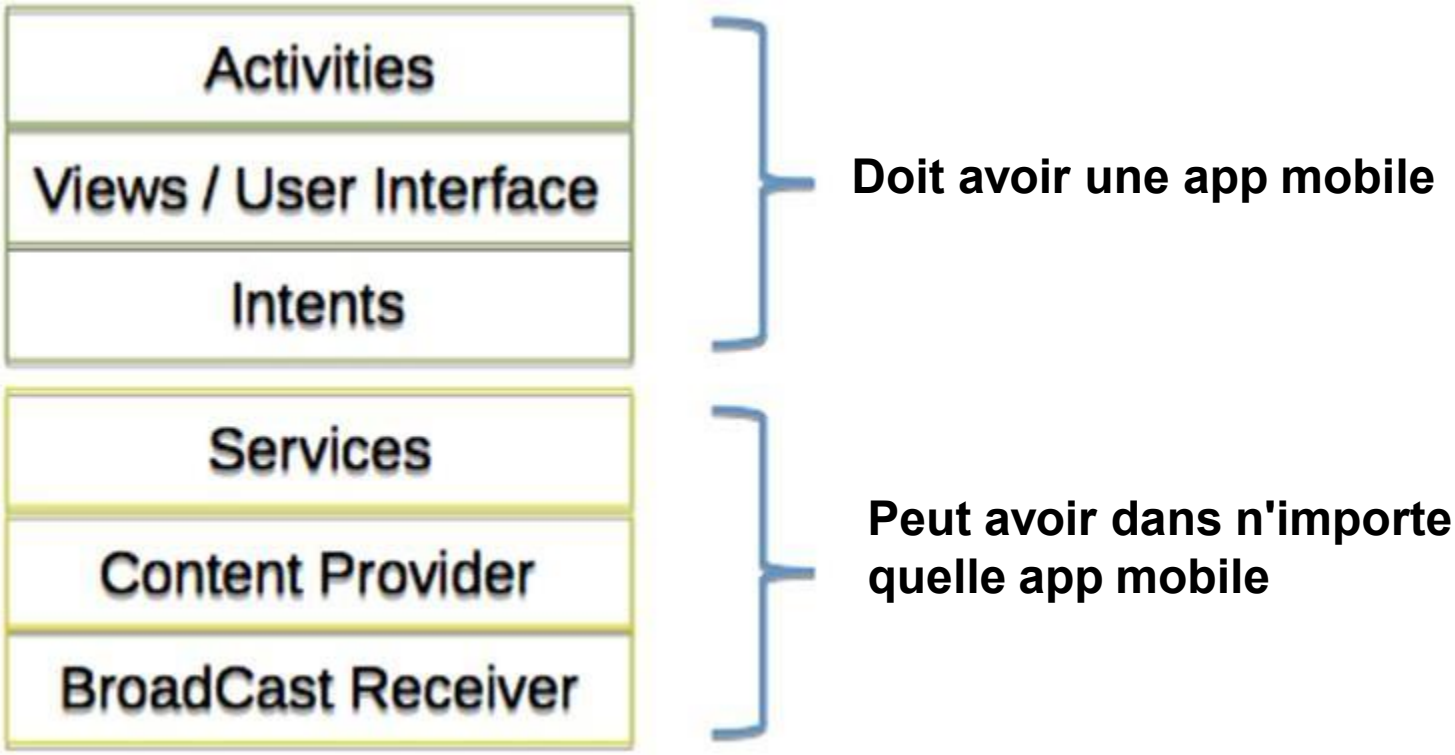
Composants de base

- Activities
- Services
- Broadcast Receivers
- Content Providers

Composants supplémentaires

- Fragements
- Views
- Layouts
- Resources
- Manifest

Composante de l’application

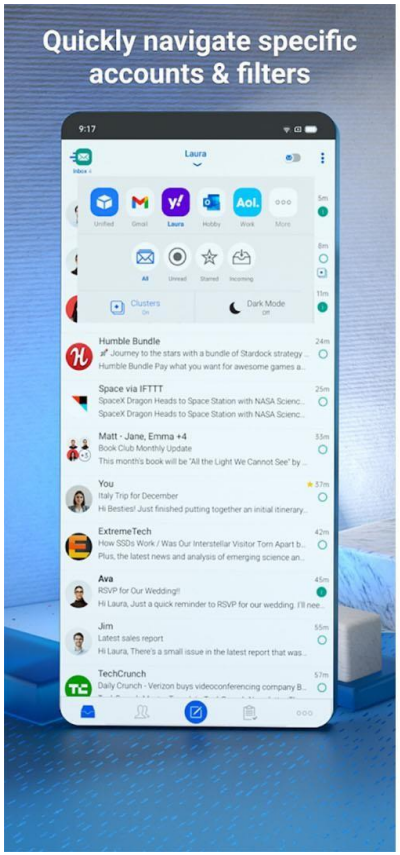


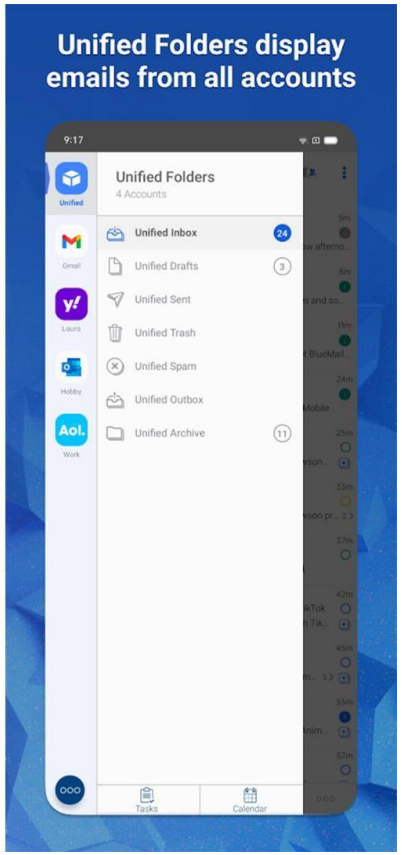
Activity

- Représenter un seul écran avec une interface utilisateur
- Contient une ou plusieurs vues (View: Composants de l'interface utilisateur)
- Gérer les interactions des utilisateurs
- Chaque activité est complètement isolée les unes des autres
- Une application différente peut invoquer l'activité d'une autre application, si cela est autorisé

Activity

- **Exemple : Application Email**





Activity

- Implementation

java

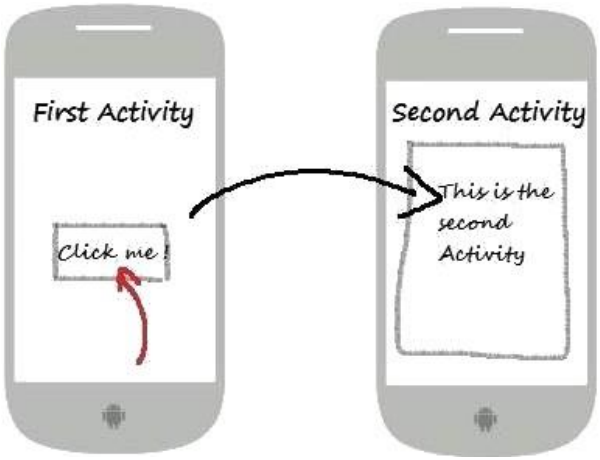
```java
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Your code for initializing the activity goes here
    }
}
```

kotlin

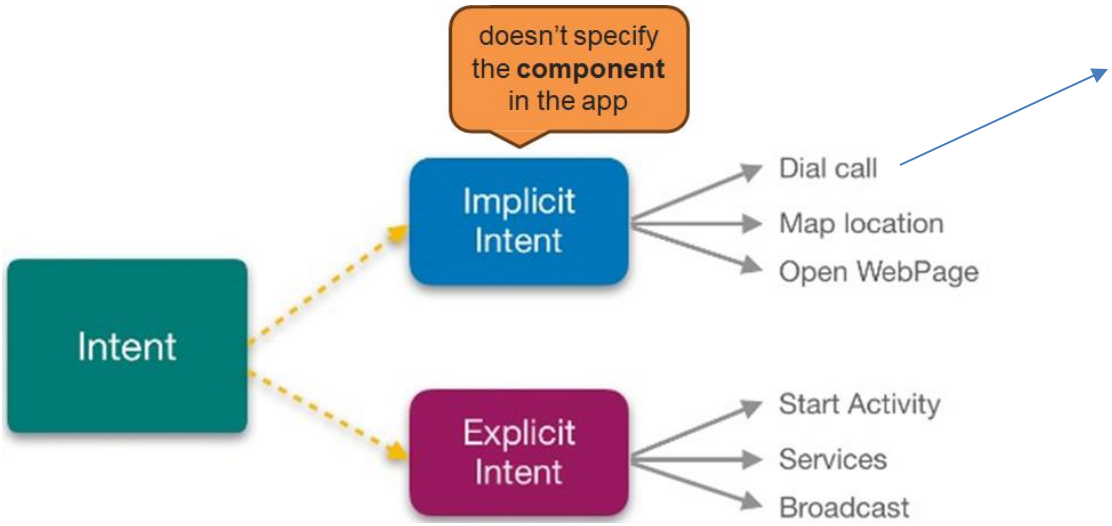
```kotlin
class MainActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Your code for initializing the activity goes here
    }
}
```

Intent

- **Intent** (Intention) sont des messages asynchrones qui permettent aux composants d'application de demander des fonctionnalités à partir d'autres composants **Android**.
- **Intent** est le moyen de transmission entre les composants (activities, content providers, broadcast receivers, services, etc., etc.)
- **intent** est un objet de la classe **android.content.Intent**.



Intent



Quand vous voulez **ouvrir le composeur téléphonique** avec un numéro, vous ne savez pas **quelle application téléphone** l'utilisateur utilise **(Samsung Phone, Google Phone, Huawei Dialer, etc.).**

Vous dites simplement à Android :
« Je veux *composer un numéro*, trouve l'app qui sait le faire »

>>> une **action générique** → Android cherche l'application enregistrée pour gérer l'action **ACTION_DIAL**.

Caractéristique	Intent explicite	Intent implicite
Cible	Composant spécifique	Composant choisi par le système
Usage	Entre composants internes	Interaction interne/externe
Exemple	startActivity(DetailActivity)	ACTION_SEND, ACTION_VIEW
Contrôle	Direct	Indirect

1. Intent explicite

Un **Intent explicite** indique **directement quel composant** doit être démarré.

•On connaît **le nom exact de l'Activity, Service ou BroadcastReceiver** cible.

•Utilisé surtout **dans sa propre application**, quand tu veux appeler un composant précis.

```
Intent intent = new Intent(MainActivity.this, DetailActivity.class);
startActivity(intent);
```

2. Intent implicite

Un **Intent implicite** ne spécifie **pas le composant exact** ; il décrit plutôt **l'action que l'on veut réaliser**.

•Le système Android choisit le composant approprié capable de gérer cette action.

•Utilisé pour **interagir avec d'autres applications** ou services du système.

Exemple : Partager un texte via n'importe quelle application de messagerie ou mail :

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_TEXT, "Bonjour tout le monde !");
startActivity(Intent.createChooser(intent, "Partager via"));
```

Intent

java

```java
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class SecondActivity extends AppCompatActivity {

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

        // Récupérez l'Intent qui a lancé cette activité
        Intent intent = getIntent();

        // Récupérez les données supplémentaires de l'Intent
        String message = intent.getStringExtra("MESSAGE");

        // Affichez le message dans un TextView
        TextView textView = findViewById(R.id.textView);
        textView.setText(message);
    }
}
```

Services

- Est un composant qui gère l'application associée au traitement en arrière-plan
- Opérations de longue durée
- Aucune interface utilisateur
- Exécution sans blocage de l'interaction de l'utilisateur

- **Exemples:**
  - Service de Téléchargement
  - Service de Lecteur de Musique
  - Service de Mise à Jour en Temps
  - Service de Sauvegarde Automatique
  - Service de Mise à Jour des Données

Services

- **Implémentation**

java

```java
public class MyService extends Service {
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // Perform long-running operation here
        // When the task is complete, stop the service using stopSelf()
        stopSelf();
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Services

- **Implémentation**

kotlin

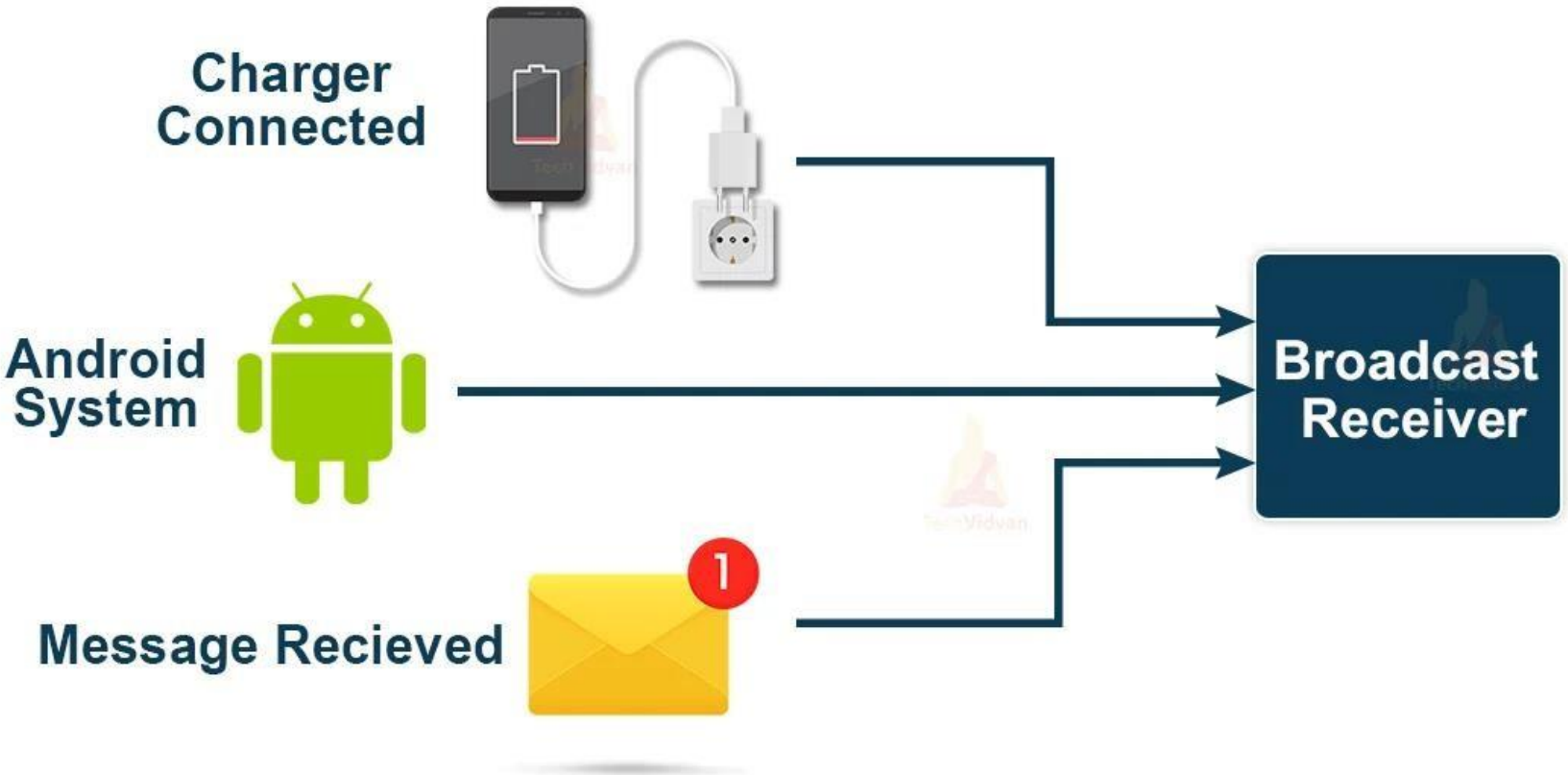
```kotlin
class MyService : Service() {

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int):
        // Perform long-running operation here
        // When the task is complete, stop the service using stopSelf()
        stopSelf()
        return START_STICKY
    }

    override fun onBind(intent: Intent?): IBinder? {
        return null
    }
}
```

BroadcastReceiver

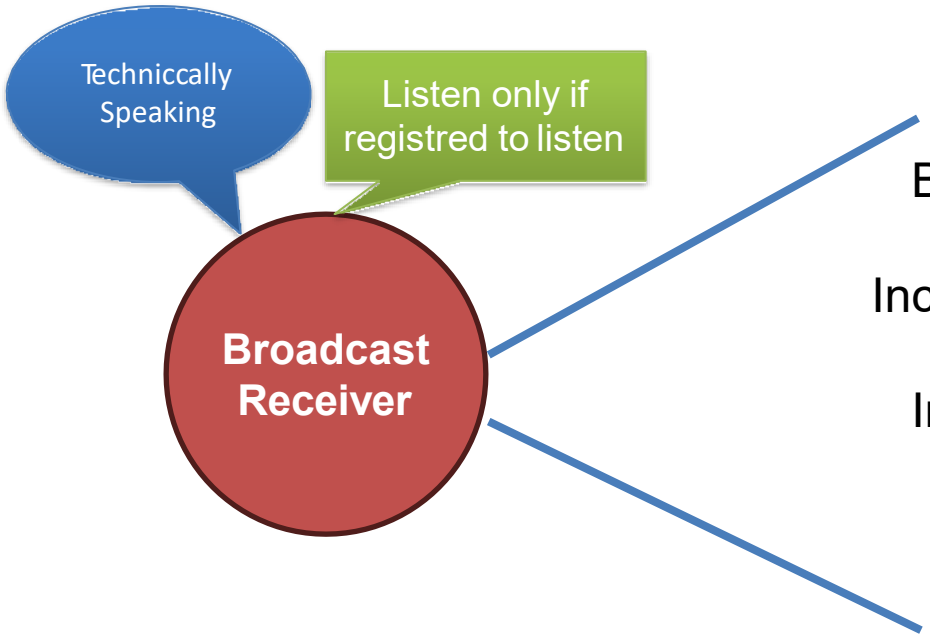
Besoin d'un « Listner »

- Recevoir & réagir aux broadcast intents
- Répondre au message diffusé
- Pas d'interface utilisateur mais peut démarrer une activité
- Répondre aux notifications ou aux changements de statut

BroadcastReceiver



BroadcastReceiver



Ce n'est pas une liste d'événements exhaustive.

Battery Low

Incomming Call

Incoming SMS

Wi-Fi availability

Bluetooth device connected

Charger (dis)connected

BroadcastReceiver: Implementation

A broadcasr receiver est implémenté en tant que sous-classe de la classe **BroadcastReceiver** et chaque message est diffusé en tant qu'objet Intent (**intent object**).

BroadcastReceiver: Implémentation

java

```java
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Votre code à exécuter lorsque le broadcast est reçu
    }
}
```

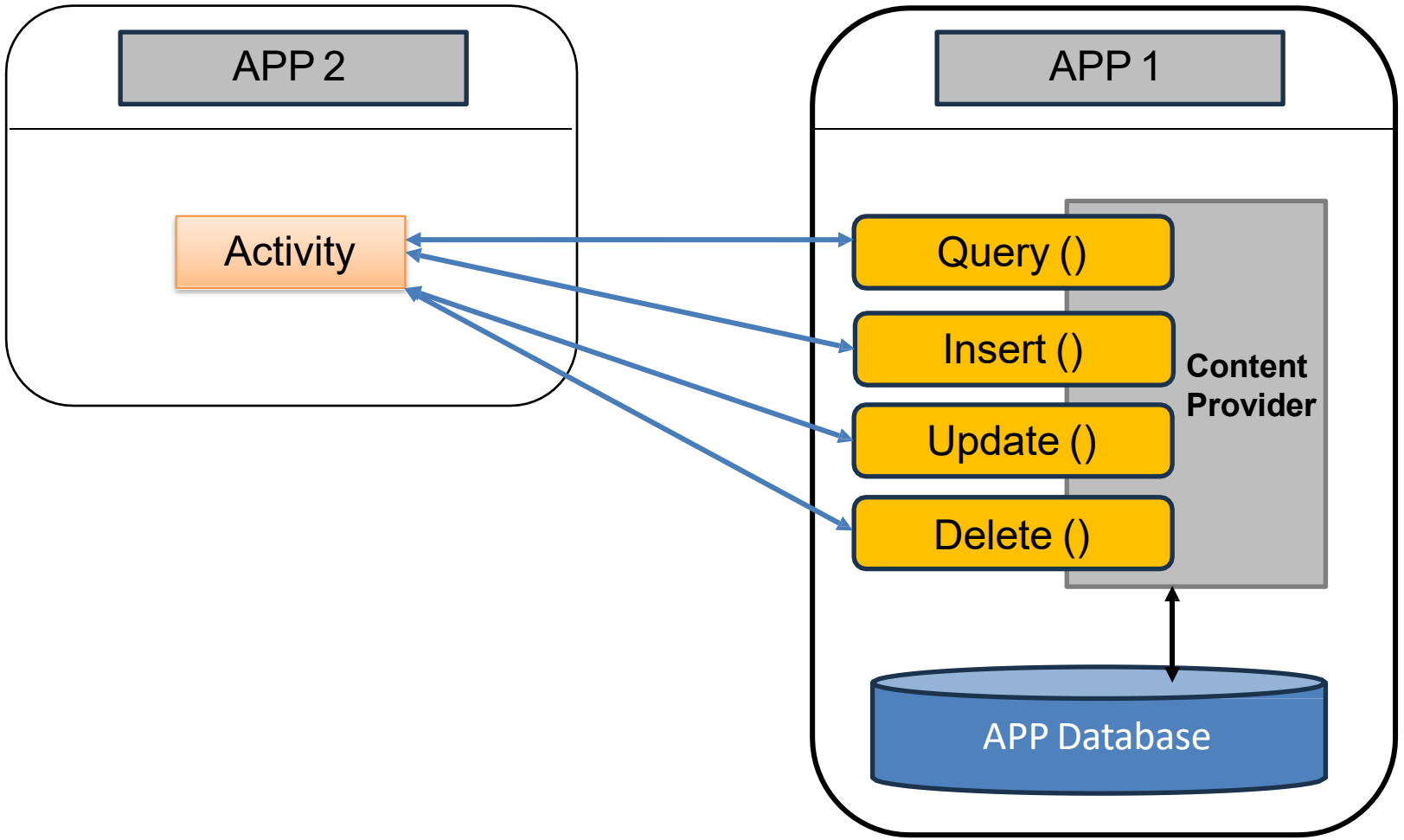
kotlin

```kotlin
class MyReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        // Votre code à exécuter lorsque le broadcast est reçu
    }
}
```

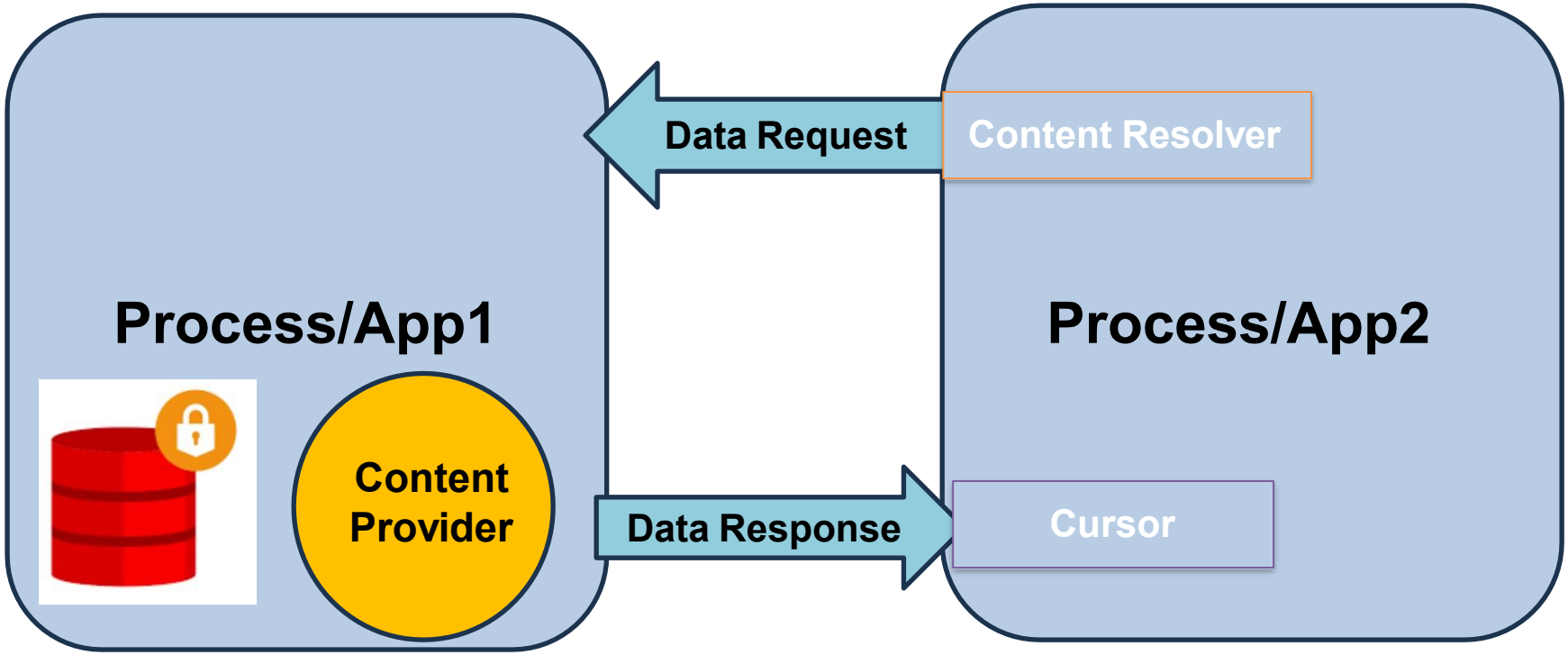
ContentProvider

- Gérer de partager des données de **manière sécurisée** entre différentes applications.
  - Permettant un accès sécurisé aux données via des autorisations
  - Garantissant l'intégrité des données.
- Une passerelle entre l'application et un stockage de données centralisé(par ex. BD SQLite, un fichier texte, un fichier JSON).
- Ils offrent un **mécanisme standardisé** pour effectuer des opérations CRUD (Create, Read, Update, Delete) sur les données,

ContentProvider



ContentProvider



ContentProvider

- **Implémentation**

java

```java
public class MyContentProvider extends ContentProvider {
    @Override
    public boolean onCreate() {
        // Votre code à exécuter lors de la création du ContentProvider
        return true;
    }

    // Reste de l'implémentation des méthodes de ContentProvider
}
```

kotlin

```kotlin
class MyContentProvider : ContentProvider() {
    override fun onCreate(): Boolean {
        // Votre code à exécuter lors de la création du ContentProvider
        return true
    }

    // Reste de l'implémentation des méthodes de ContentProvider
}
```

Android Activity Lifecycle

Activity

Contents provider

?

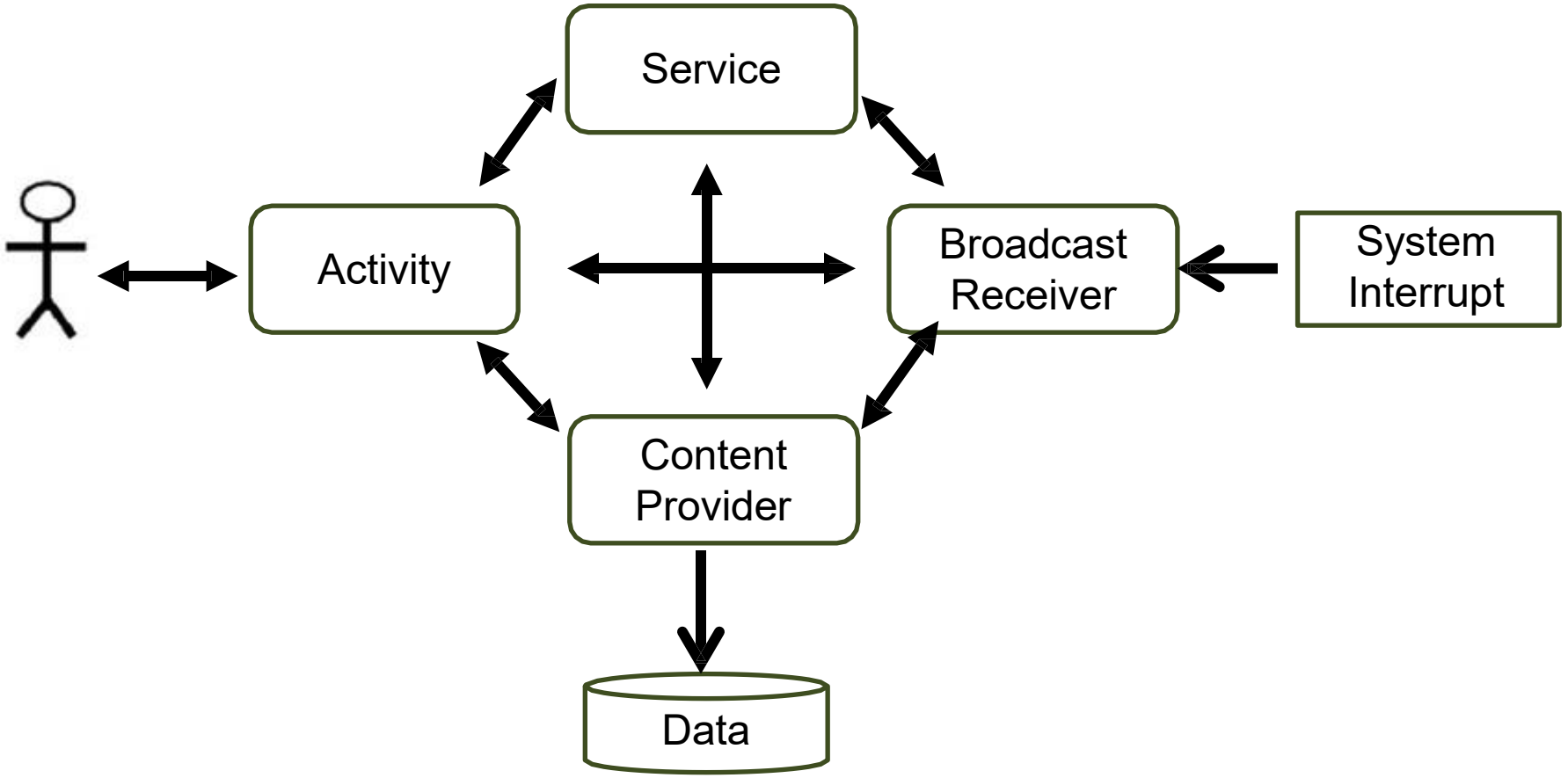
Service

Broadcast Receiver

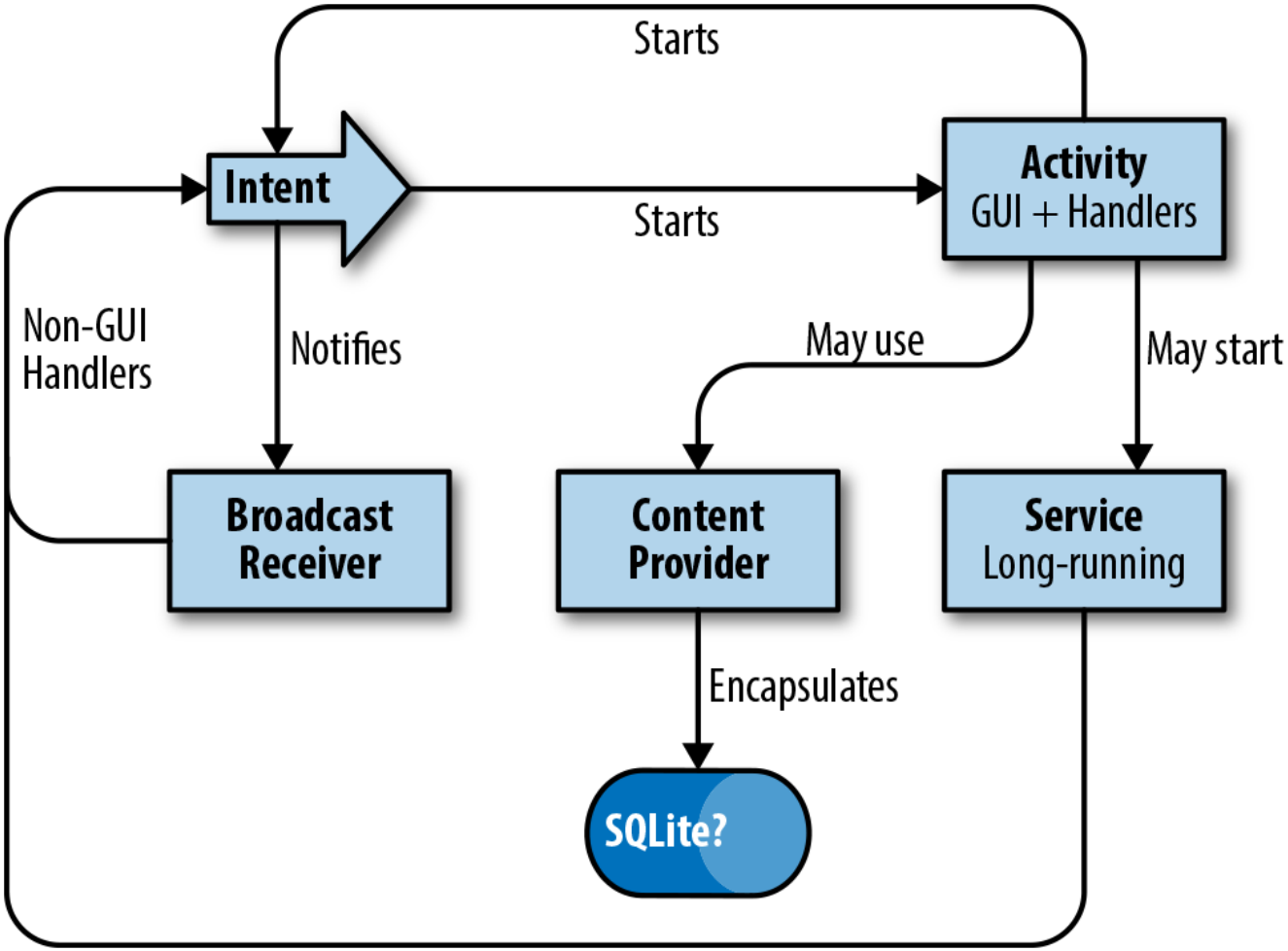
Activity handles user interface

Android broadcasts sensor information via intent

Relation entre les composants



Android Activity Lifecycle

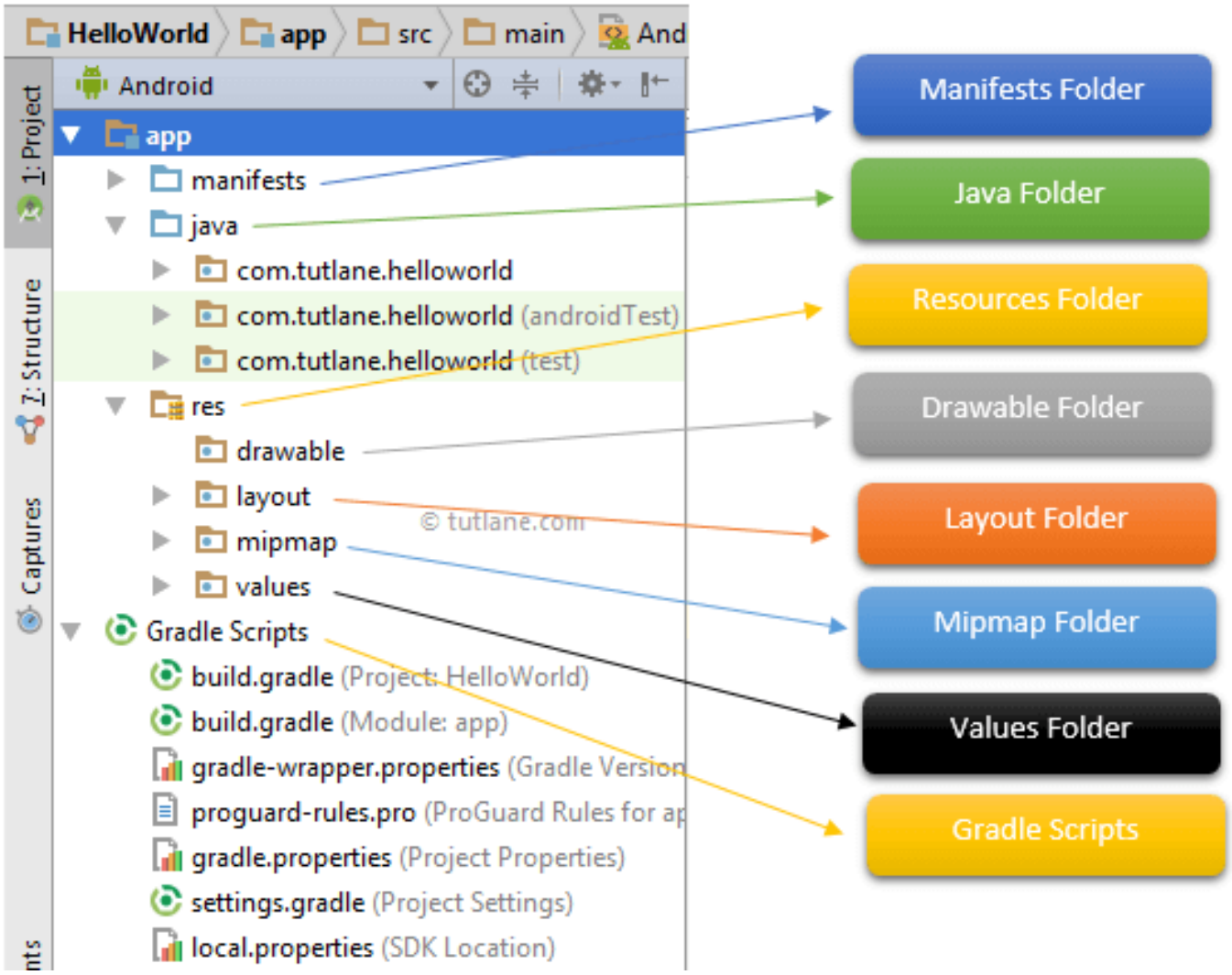


Composants supplémentaires

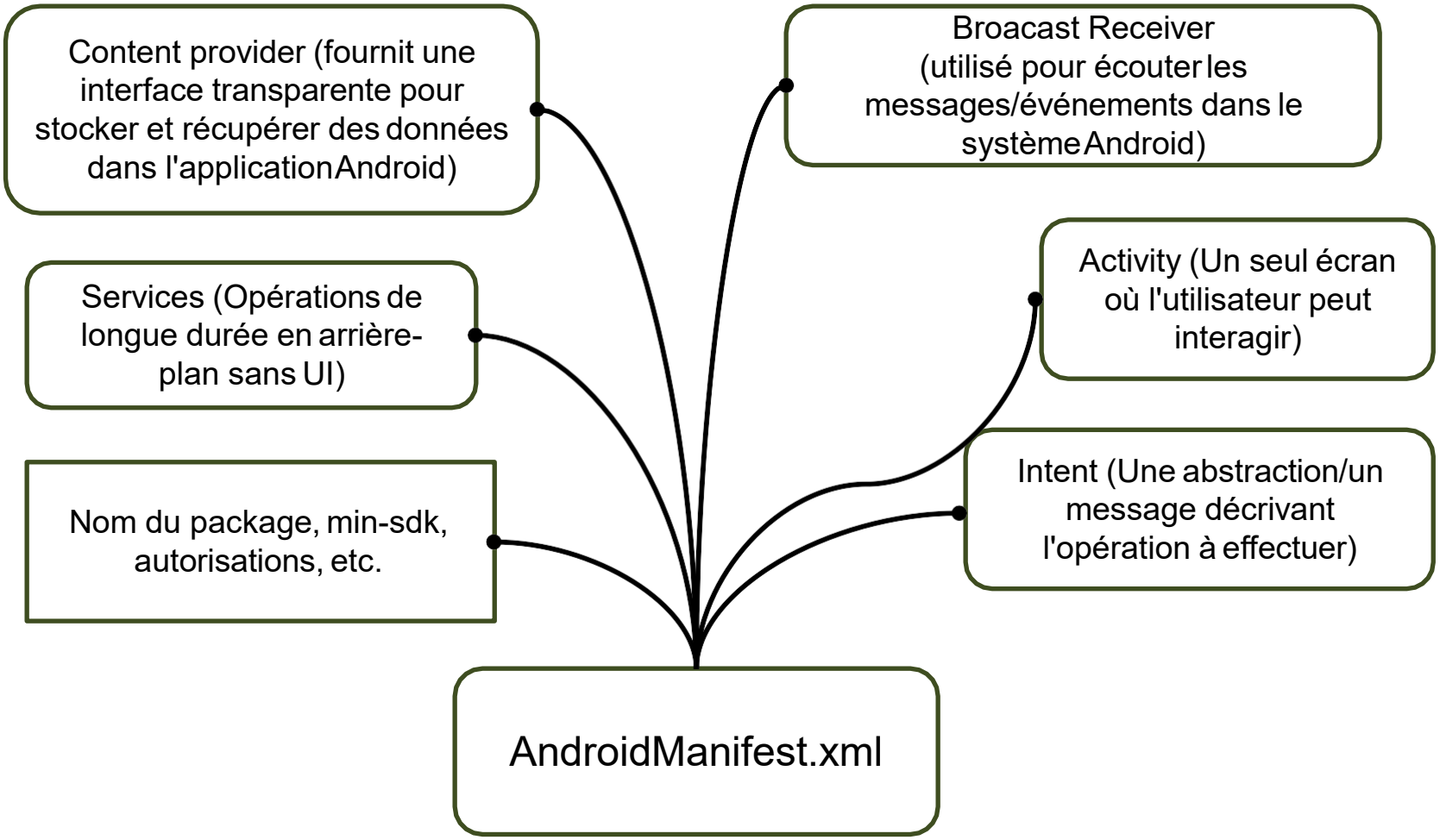
Composante	Description
Fragments	Représente une partie de l'interface utilisateur dans une activité.
Views	Éléments d'interface utilisateur noyés à l'écran, notamment des boutons, des formulaires de listes, etc.
Layouts	Hiérarchies de vues (views) qui contrôlent le format d'écran et l'apparence des vues (views)
Resources	Éléments externes, tels que des chaînes (String), des constantes et des images dessinables(drawable pictures)
Manifest	Fichier de configuration de l'application

Structure de projet d'une application Android

Structure de projet d'une application Android



Configuration de Android Manifest



Configuration de Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.example"
          android:versionCode="1"
          android:versionName="1.0">

    <uses-sdk android:minSdkVersion="15" />

    <application android:label="@string/app_name"
                 android:debuggable="true"
                 android:icon="@drawable/ic_la

        <activity android:name="MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <service>...</service>

        <receiver>...</receiver>

        <provider>...</provider>

    </application>
</manifest>
```

1. Manifest element
2. Uses-sdk element
3. Application element
4. Activity element
5. Service element
6. Receiver element
7. Provider element

Configuration de Android Manifest

Permissions: L'élément <uses-permission> est utilisé pour déclarer les autorisations dont l'application a besoin pour accéder à certaines fonctionnalités ou données sur l'appareil.

Par exemple, accéder à Internet, lire des contacts ou accéder à la caméra.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.CAMERA" />
```

Configuration de Android Manifest

Accès aux données d'application et d'utilisateur : L'élément <application> contient divers attributs et éléments enfants qui définissent la manière dont l'application interagit avec l'utilisateur et le système. Cela inclut la spécification de l'icône, du label, du thème, des activités, des services, des récepteurs de diffusion, des fournisseurs de contenu, etc.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <!-- Activities, services, broadcast receivers, content providers, etc.

</application>
```

Configuration de Android Manifest

Déclaration des composants: Les activités, services, récepteurs de diffusion et fournisseurs de contenu sont déclarés dans l'élément <application>.

Par exemple, un élément <activity> déclare un composant d'activité :

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Configuration de Android Manifest

Content Providers: Les fournisseurs de contenu, si votre application les utilise, sont déclarés dans le manifeste. Les fournisseurs de contenu permettent à votre application de partager des données avec d'autres applications.

Par exemple:

```
<provider
    android:name=".MyContentProvider"
    android:authorities="com.example.myapp.provider"
    android:exported="true">
</provider>
```

Configuration de Android Manifest

Pour ajouter la prise en charge des langues s'écrivant de droite à gauche (RTL:(RTL) languages such as Arabic in your Android app, you ne) telles que l'arabe dans votre application Android, vous devez effectuer certains ajustements dans votre fichier AndroidManifest.xml ainsi que dans vos fichiers de mise en page.

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <!-- Other application attributes -->
</application>
```

Thank you !

Questions ? abdelkader.ouared@univ-tiaret.dz

www. .univ-tiaret.dz