

Programmation Mobile

Anatomie d'une Application Android

2024 - 2025
Dr. Abdelkader Ouared



INTRODUCTION À ANDROID: PRÉSENTATION



Android OS vs Android SDK

- **Android OS** est ce que les utilisateurs finaux interagissent avec sur leurs appareils
- **Android SDK** est utilisé par les développeurs pour créer des applications qui fonctionnent sur ce système d'exploitation

Introduction à Android: Présentation

- **Qu'est ce qu'Android OS**

- 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, Microphone,...).
 - 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 les caméras numérique
- 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.

ARCHITECTURE

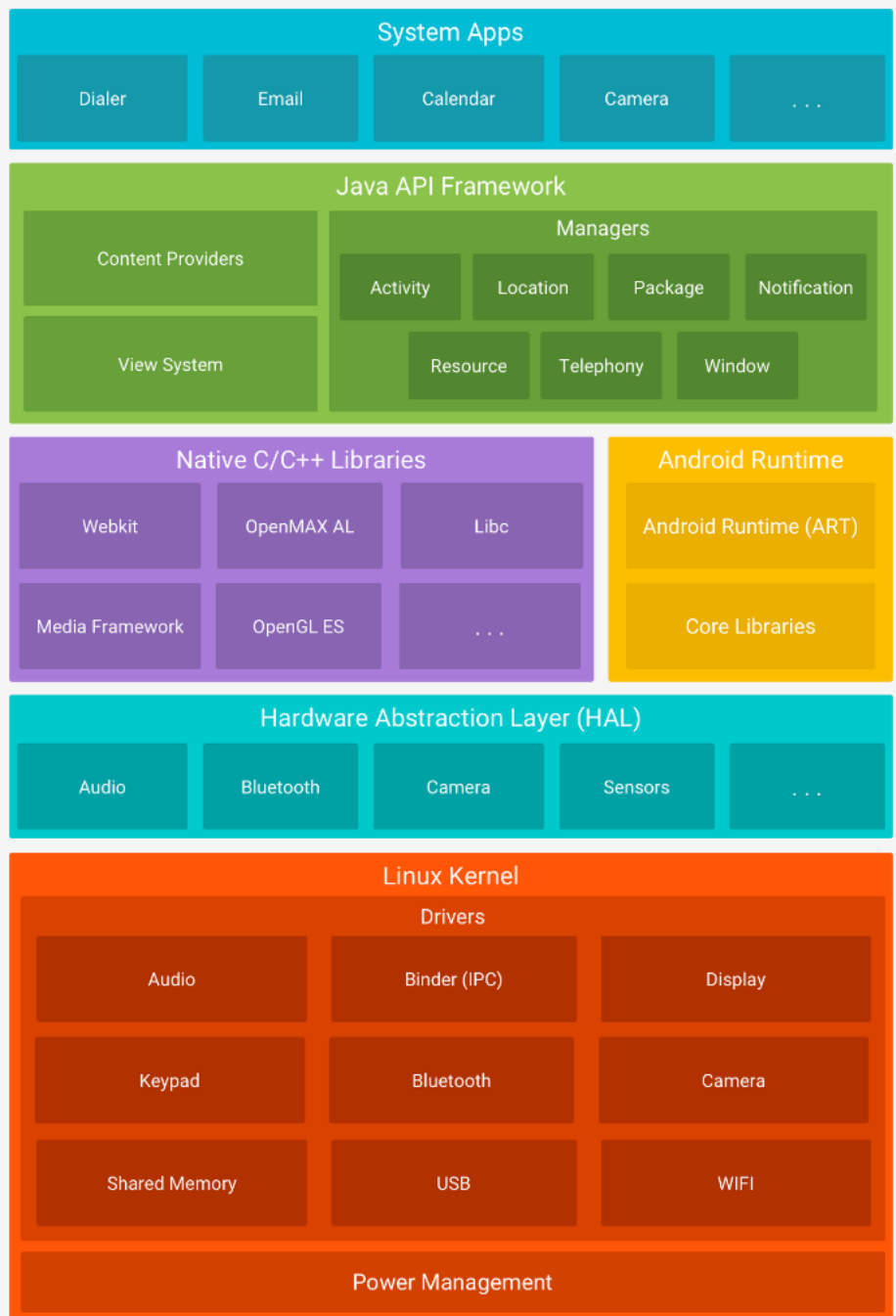


Architecture des applications Android

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

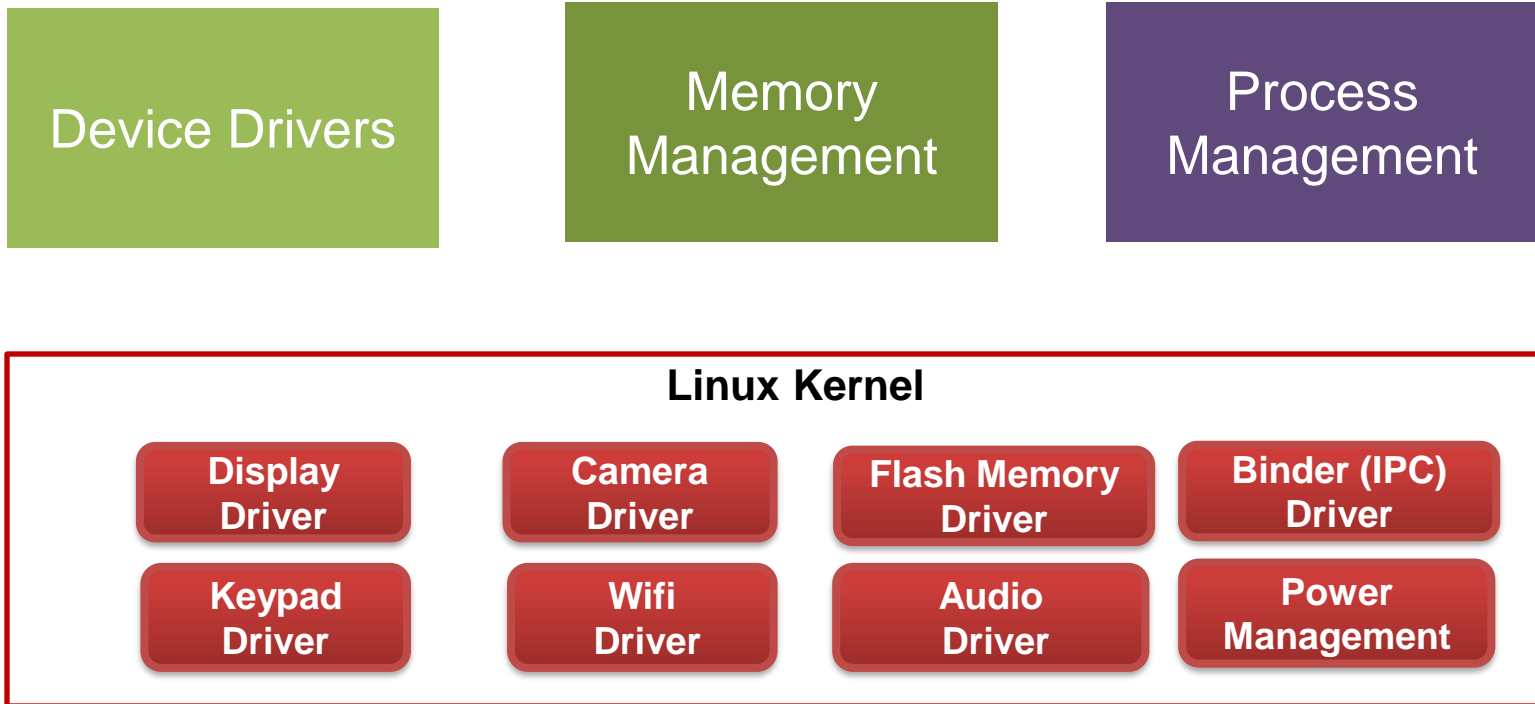
Peut être modifié, si nécessaire

Pas de redevances ni de frais de licence



Architecture des applications

Android: **Linux Kernel**



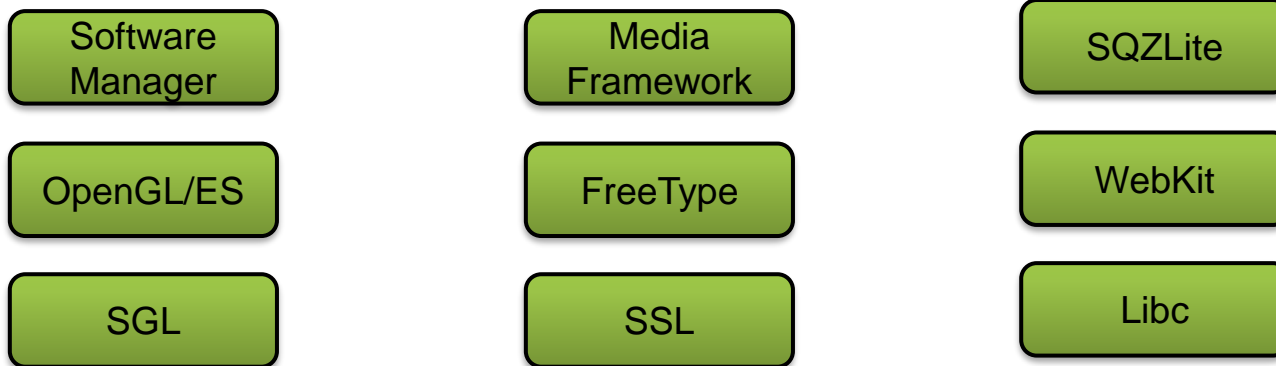
Binder (IPC) Driver: Inter-Process Communication

Architecture des applications

Android: **Libraries**



Libraries



Architecture des applications

Android: **Application Manager**

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

Application Manager

Activity Manager

Windows Manager

Content Providers

View System

Package
Manager

Telephony Manager

Resource Manager

Location Manager

Notification
Manager

Architecture des applications

Android: **Android Runtime**

❑ Dalvik VM

- Dalvik est une machine virtuelle qui exécute les applications Android.
- Légère et optimisée (en termes de mémoire et de consommation d'énergie.)
- **Caractéristiques :**
 - **Compilation Just-In-Time (JIT)** : le code est compilé lors de l'exécution.
 - **Format DEX** : Les applications Android sont compilées en un format de bytecode spécifique appelé DEX (Dalvik Executable) qui est optimisé pour fonctionner dans l'environnement Dalvik.
 - **Gestion de la mémoire** : Dalvik comprend un ***garbage collector*** qui gère la mémoire de manière efficace pour minimiser la consommation de ressources.

❑ Core Libraries (libcore)

- un ensemble de bibliothèques Java incluses dans le runtime d'Android
ex. java.lang, java.util, java.io

Android Runtime

Core Libraries

Virtual Machine

Architecture des applications

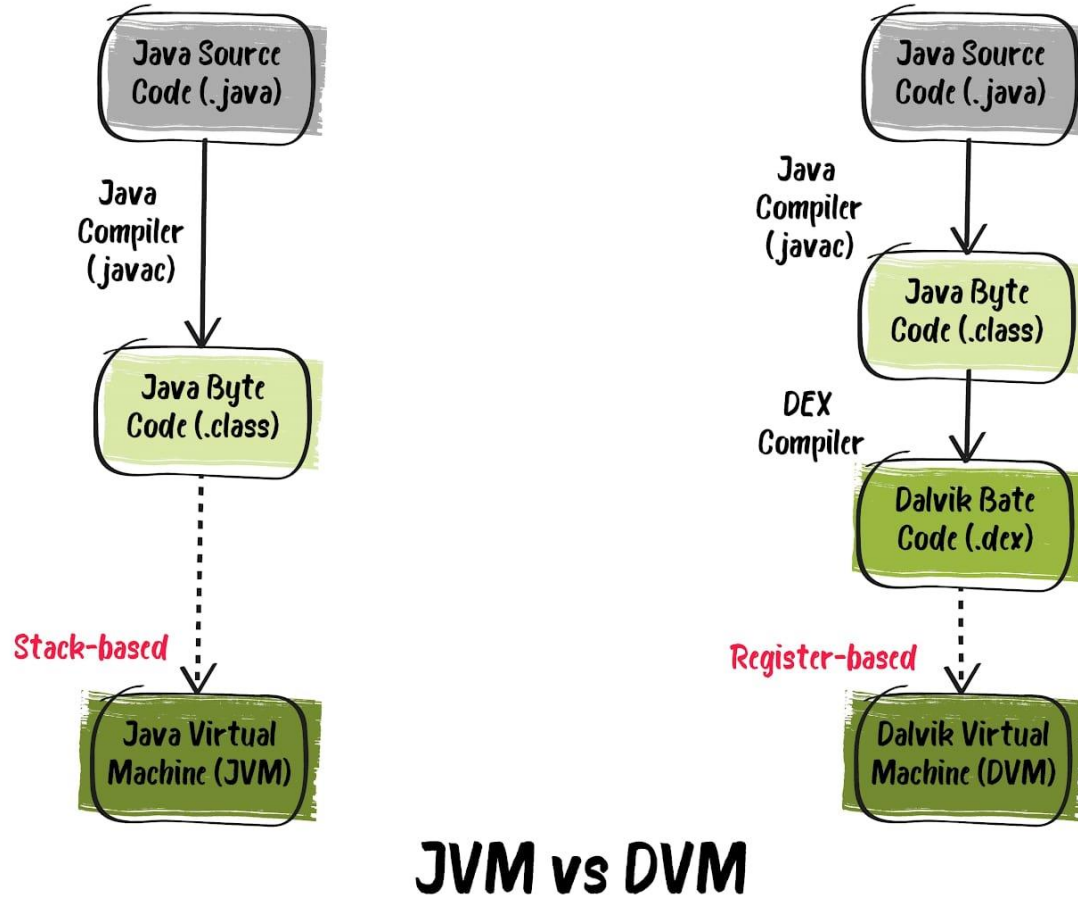
Android: **Android Runtime**

Dalvik virtual machine: JVM optimisé pour les plateformes mobiles

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

Architecture des applications Android:

Android Runtime



COMPOSANTES D'UNE APPLICATION ANDROID



Composante de l'application

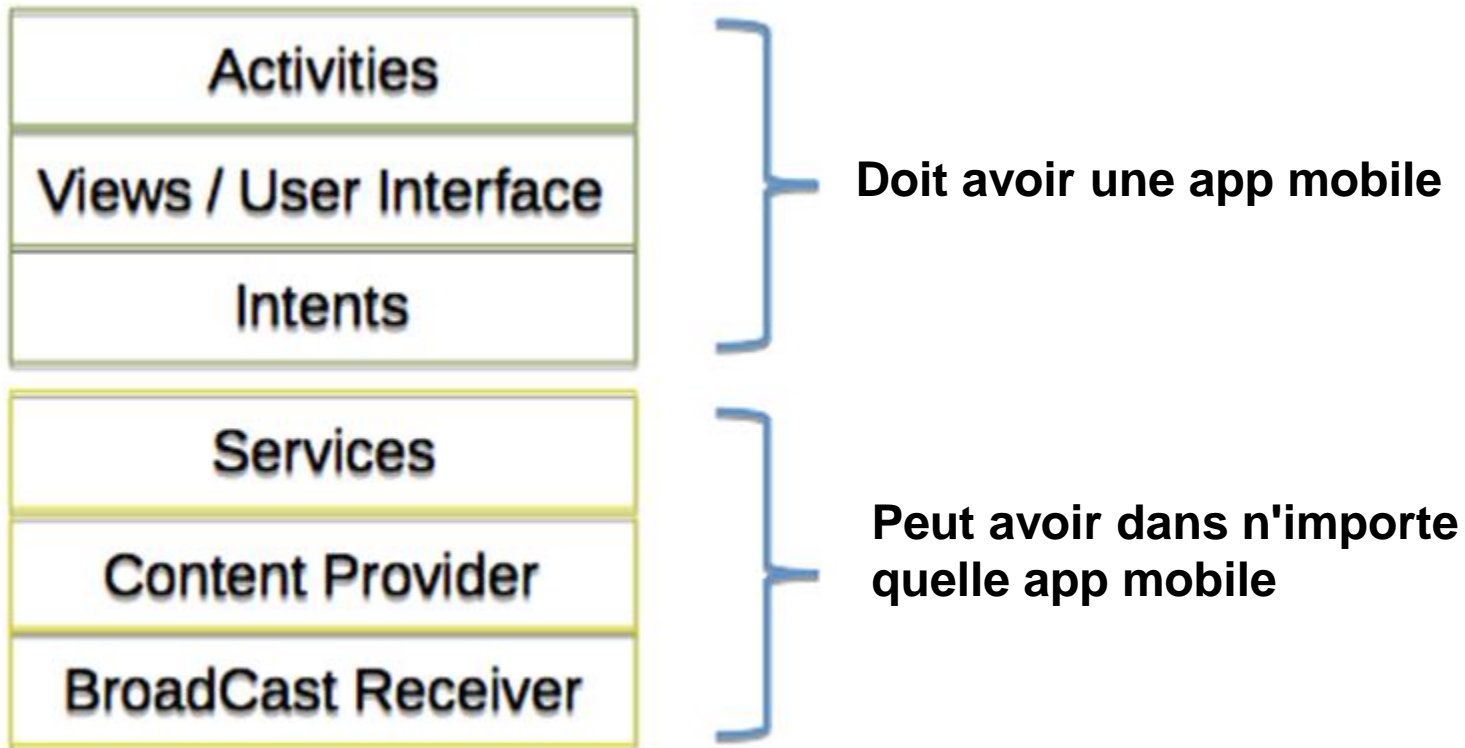
Composants de base

- Activities
- Services
- Broadcast Receivers
- Content Providers

Composants supplémentaires

- Fragements
- Views
- Layouts
- Resources
- Manifest

Composante de l'application

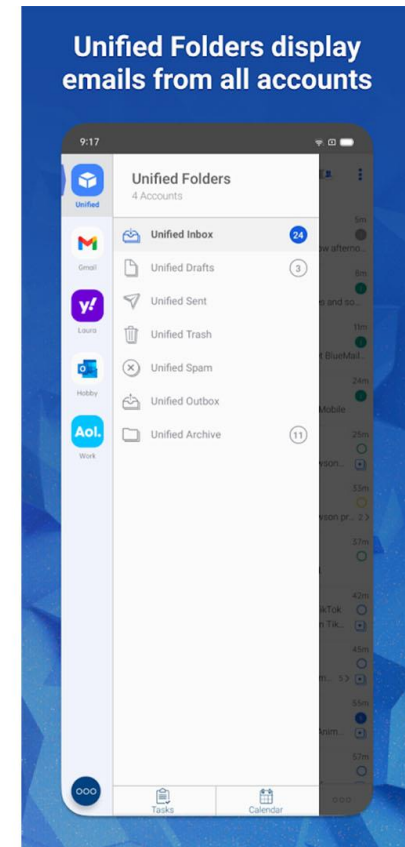
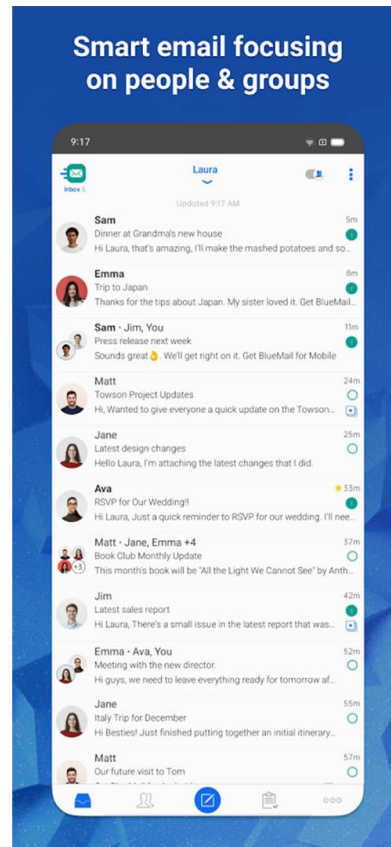
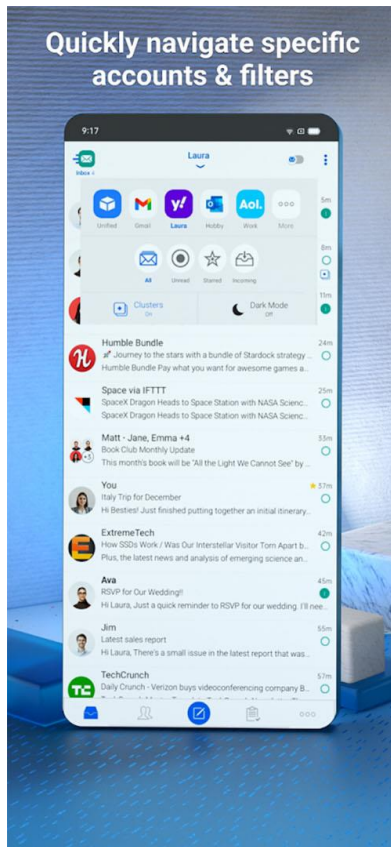


Activity

- 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

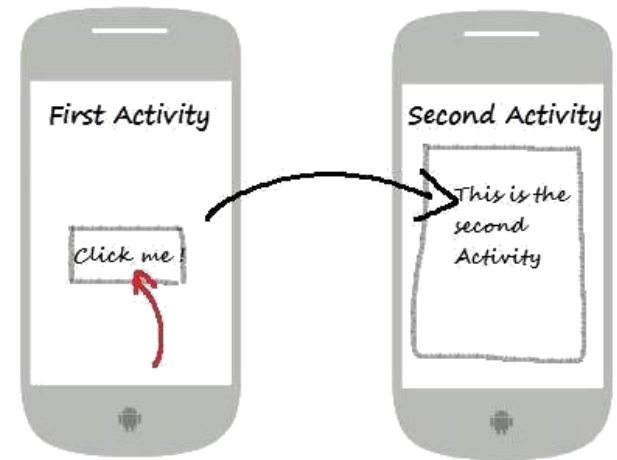
```
public class MainActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Your code for initializing the activity goes here  
    }  
}
```

kotlin

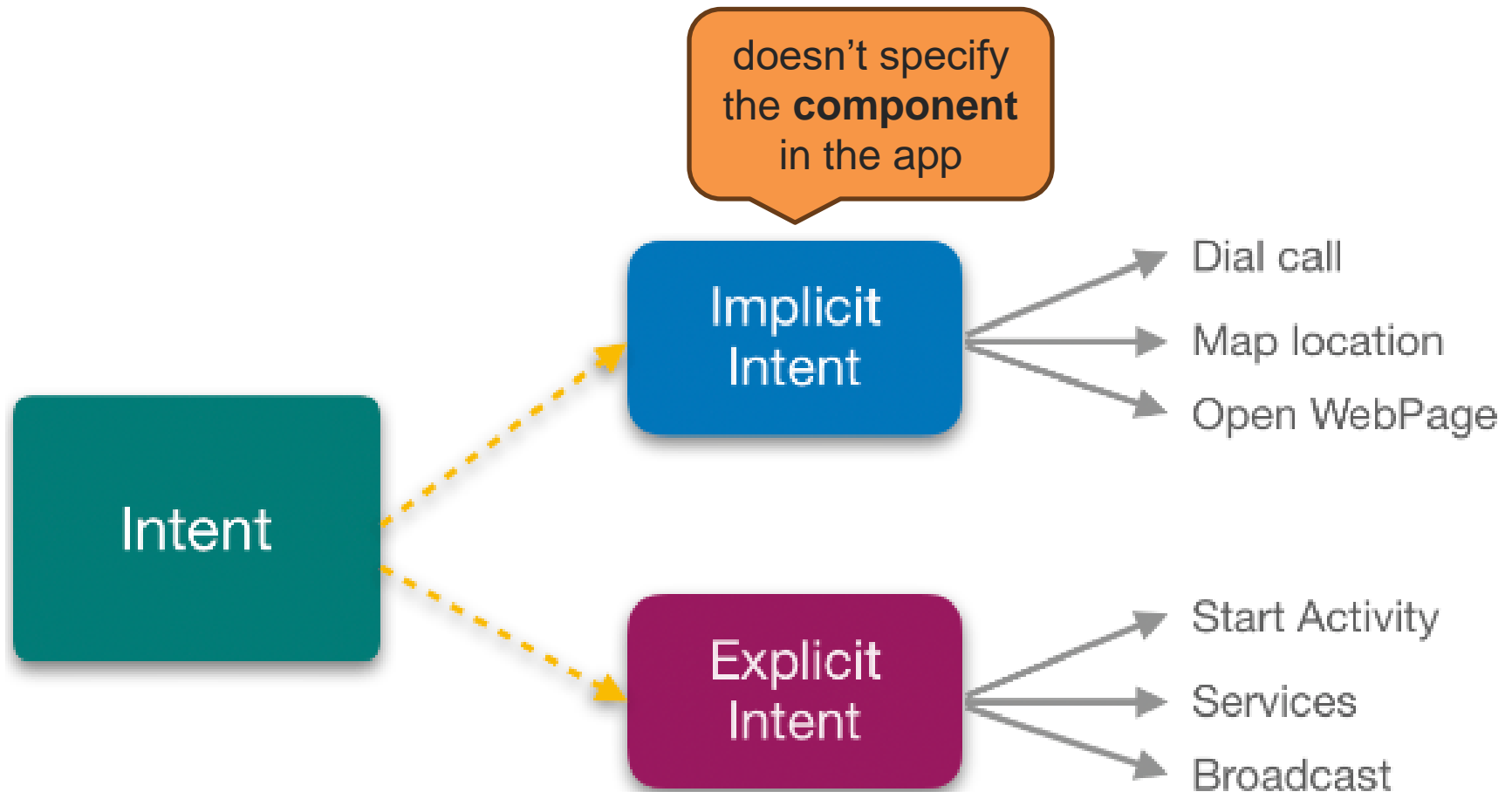
```
class MainActivity : Activity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        // Your code for initializing the activity goes here  
    }  
}
```

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 (activités, content providers, broadcast receivers, services, etc., etc.)
- **intent** est un objet de la classe **android.content.Intent**.



Intent



Intent

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

```
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

```
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  
    }  
}
```

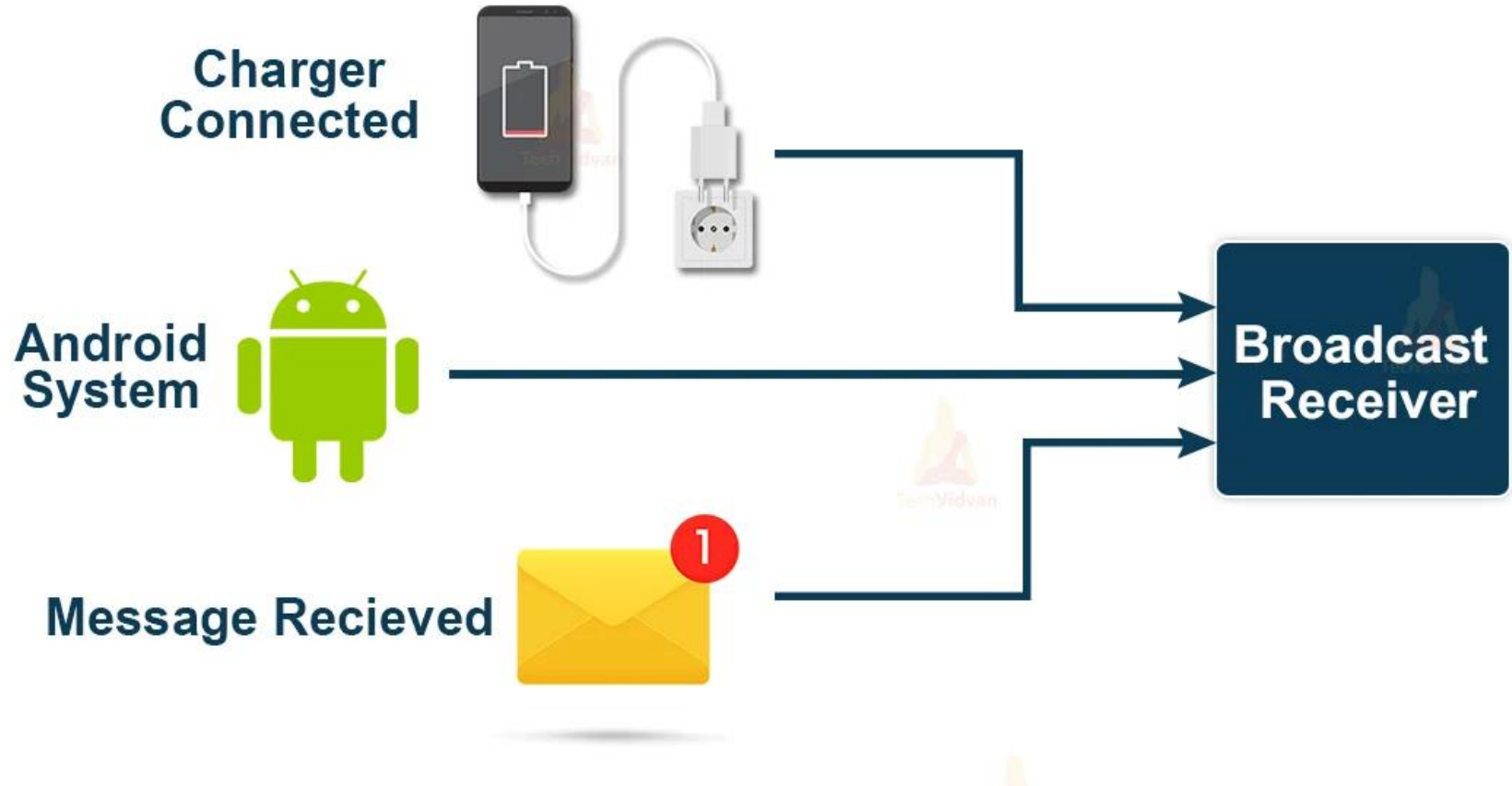
BroadcastReceiver



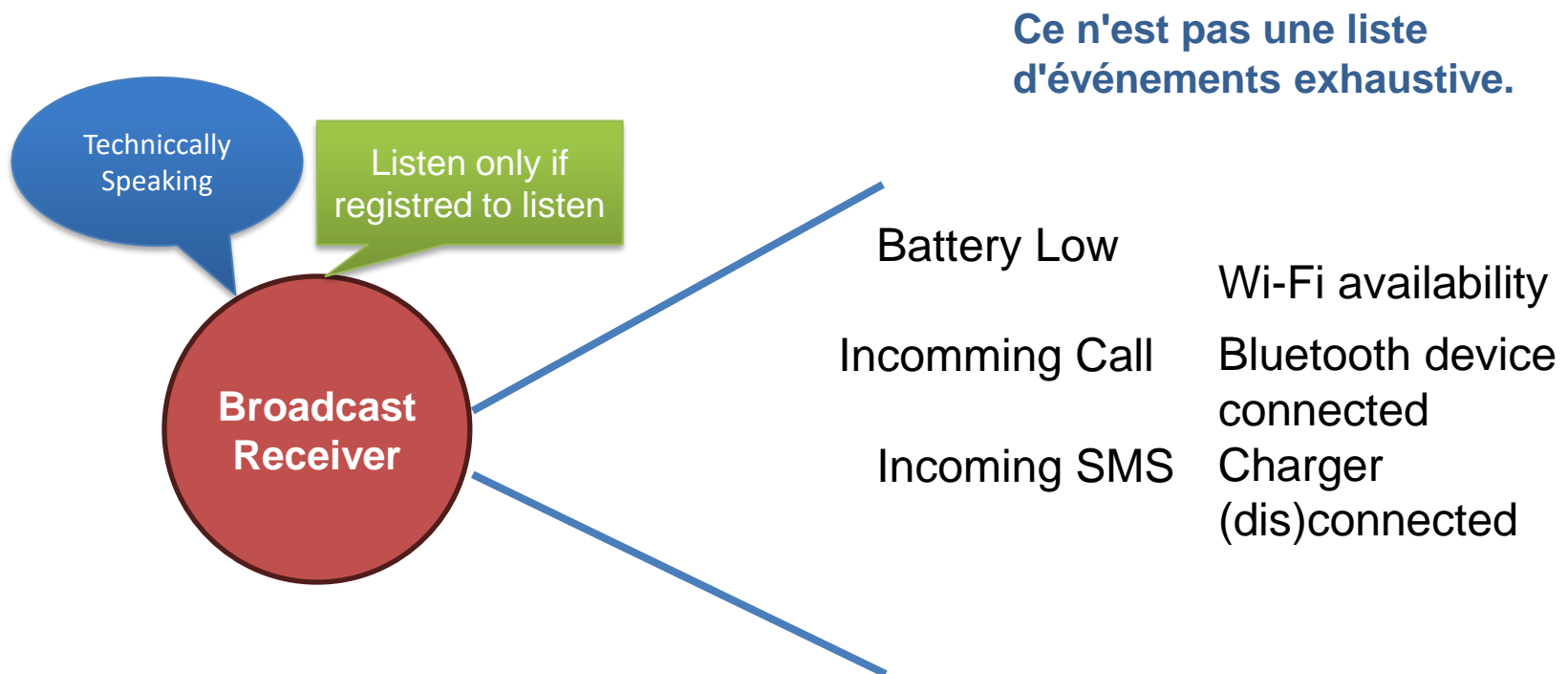
Besoin d'un « Listener »

- 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



BroadcastReceiver: Implémentation

A broadcast 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

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

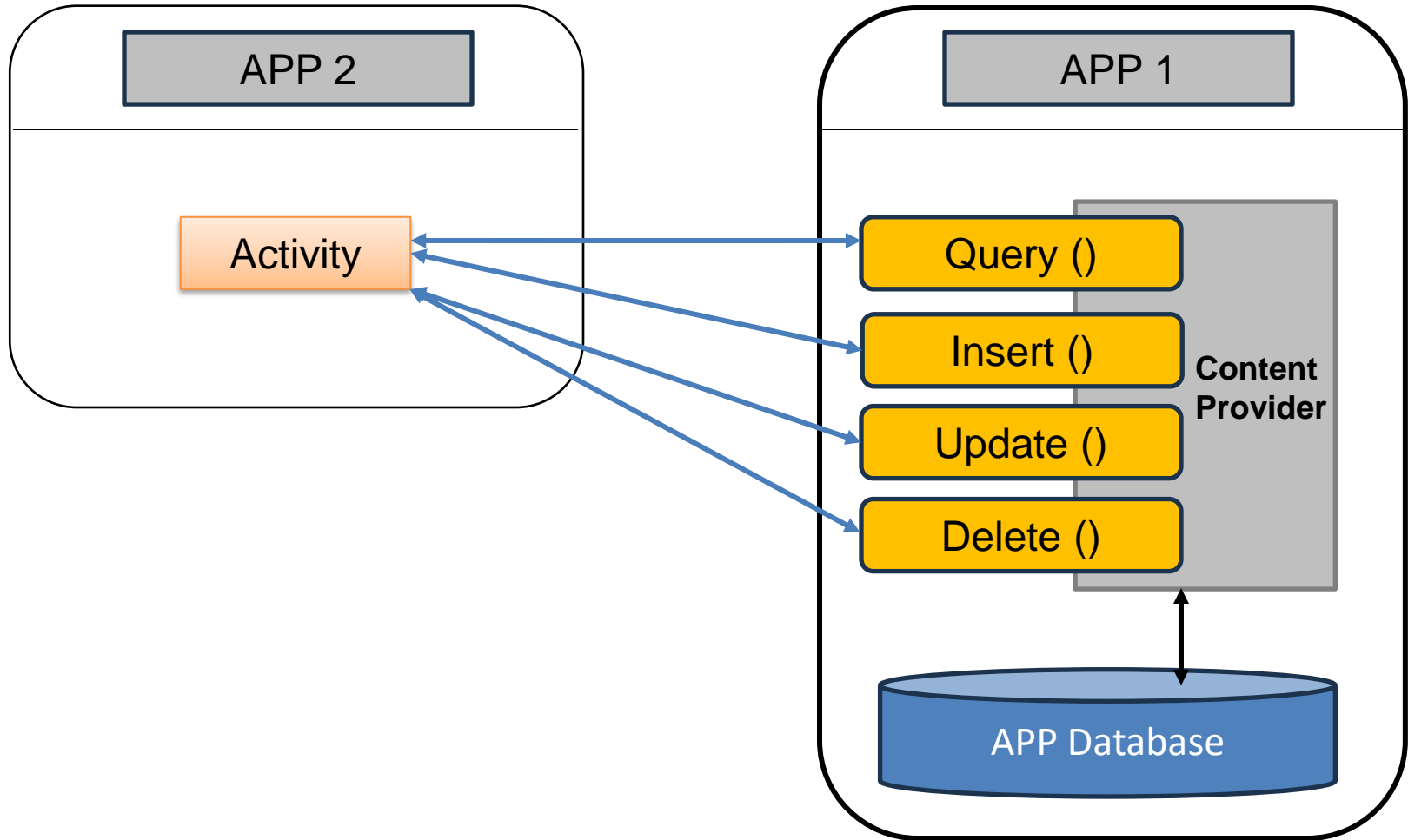
kotlin

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

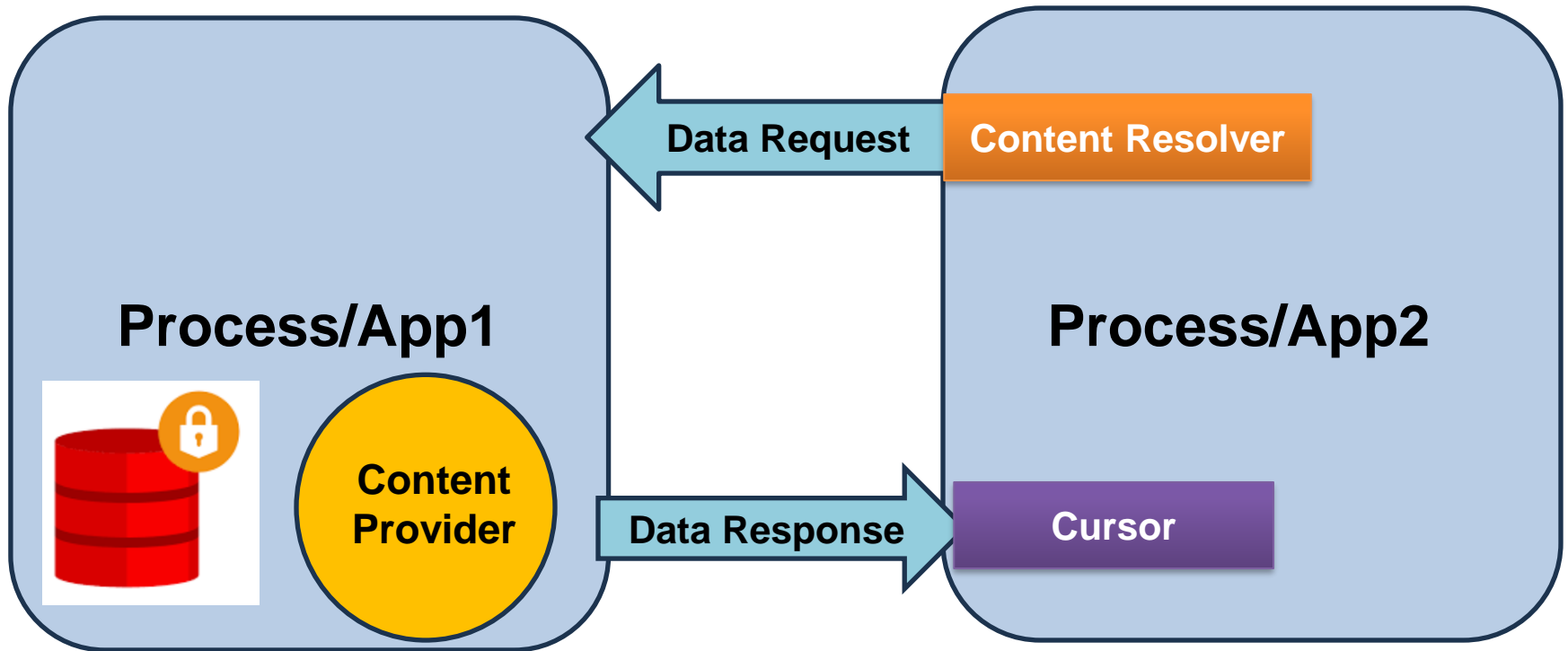
ContentProvider

- 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



Android OS

ContentProvider

- Implémentation

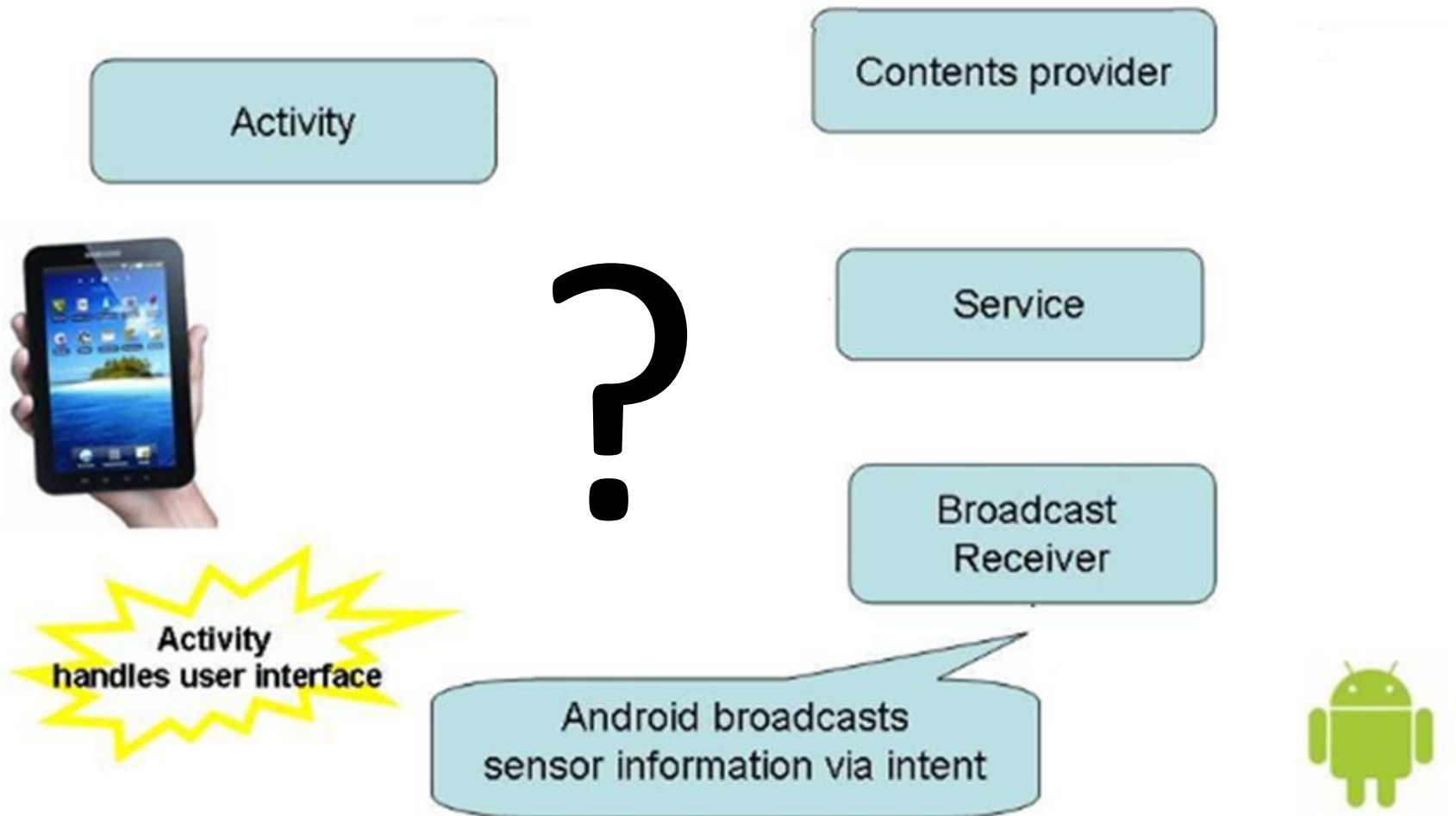
java

```
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  
}
```

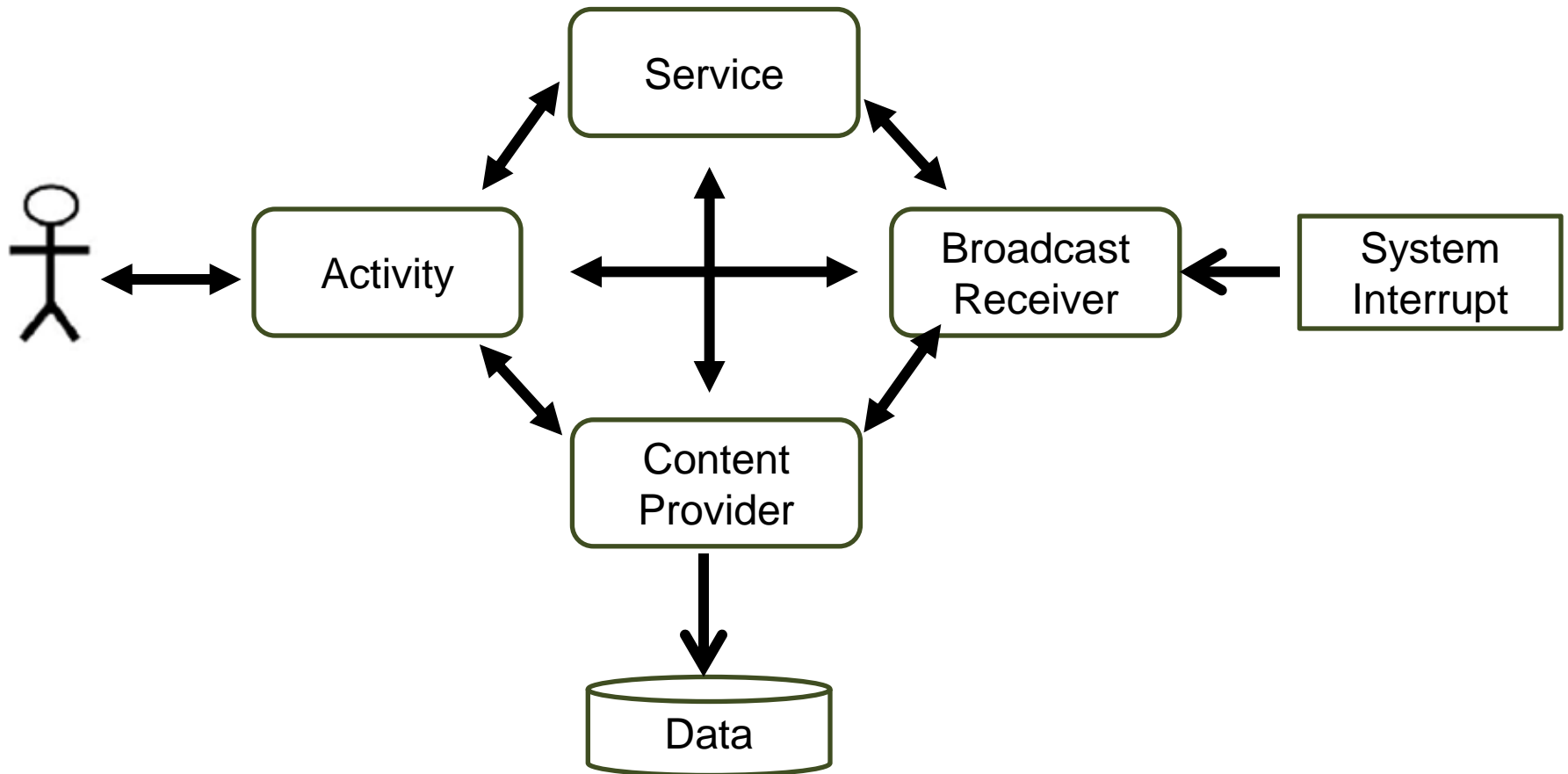
kotlin

```
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  
}
```

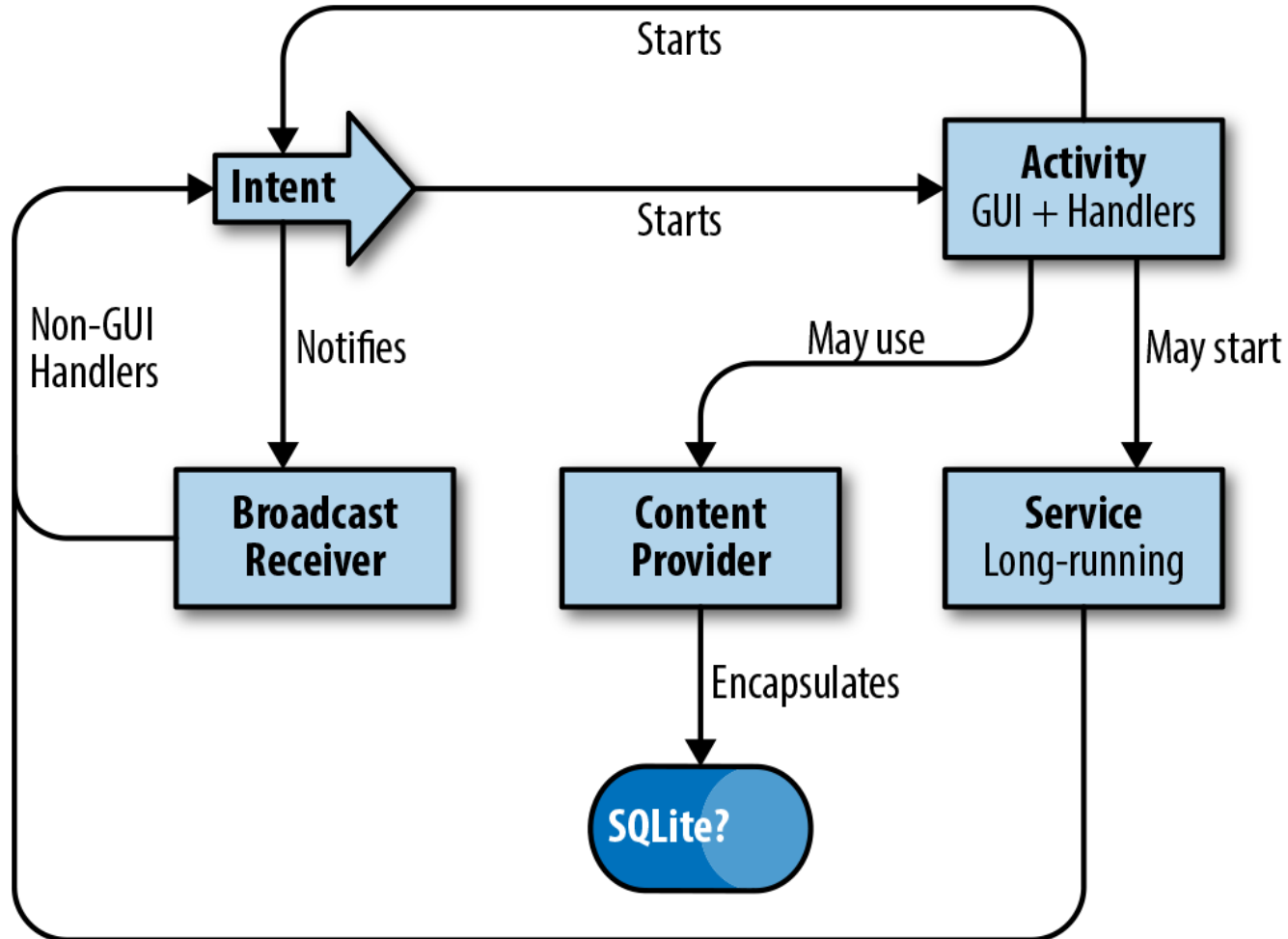
Android Activity Lifecycle



Relation entre les composants



Android Activity Lifecycle



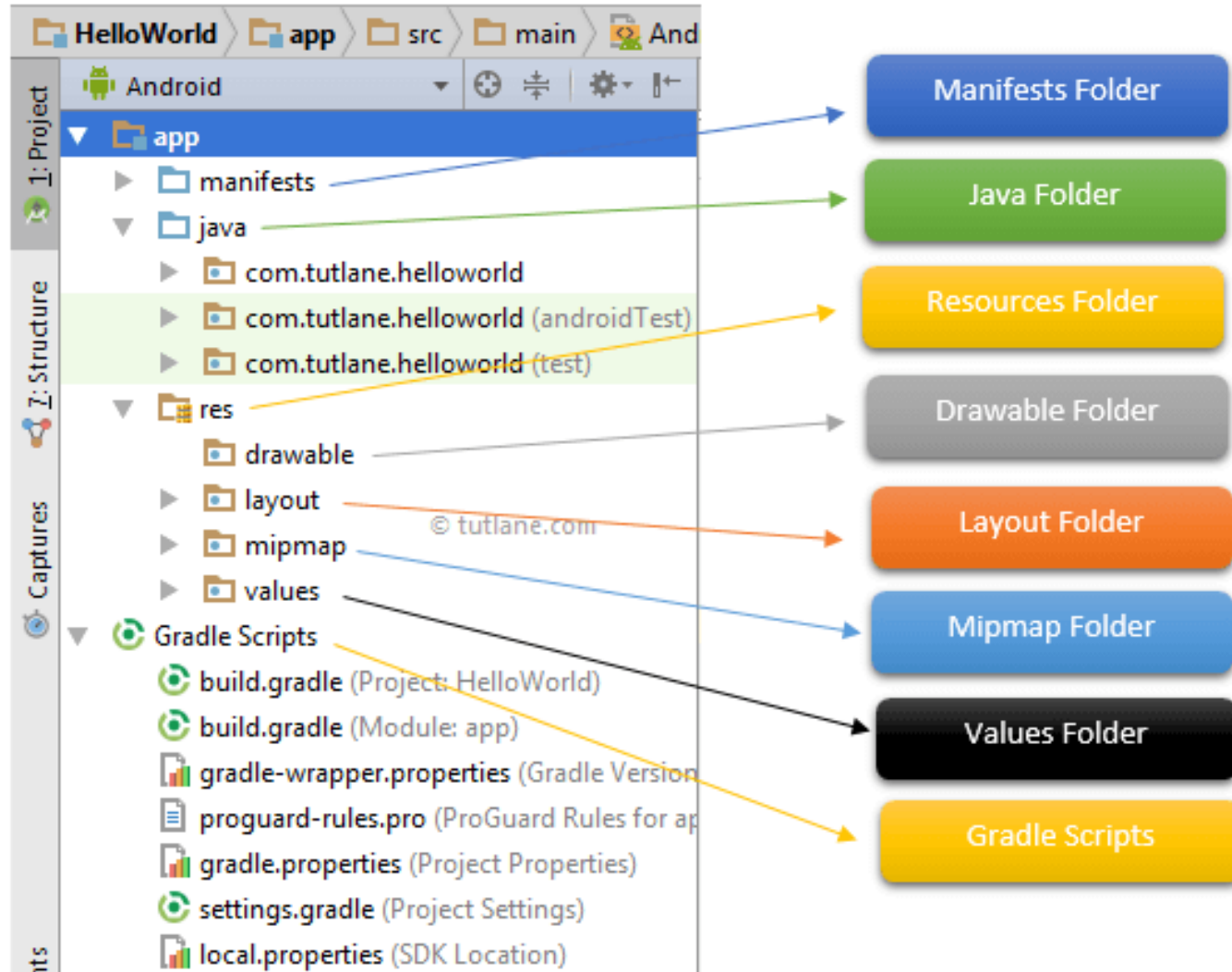
Composants supplémentaires

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	Hierarchies 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

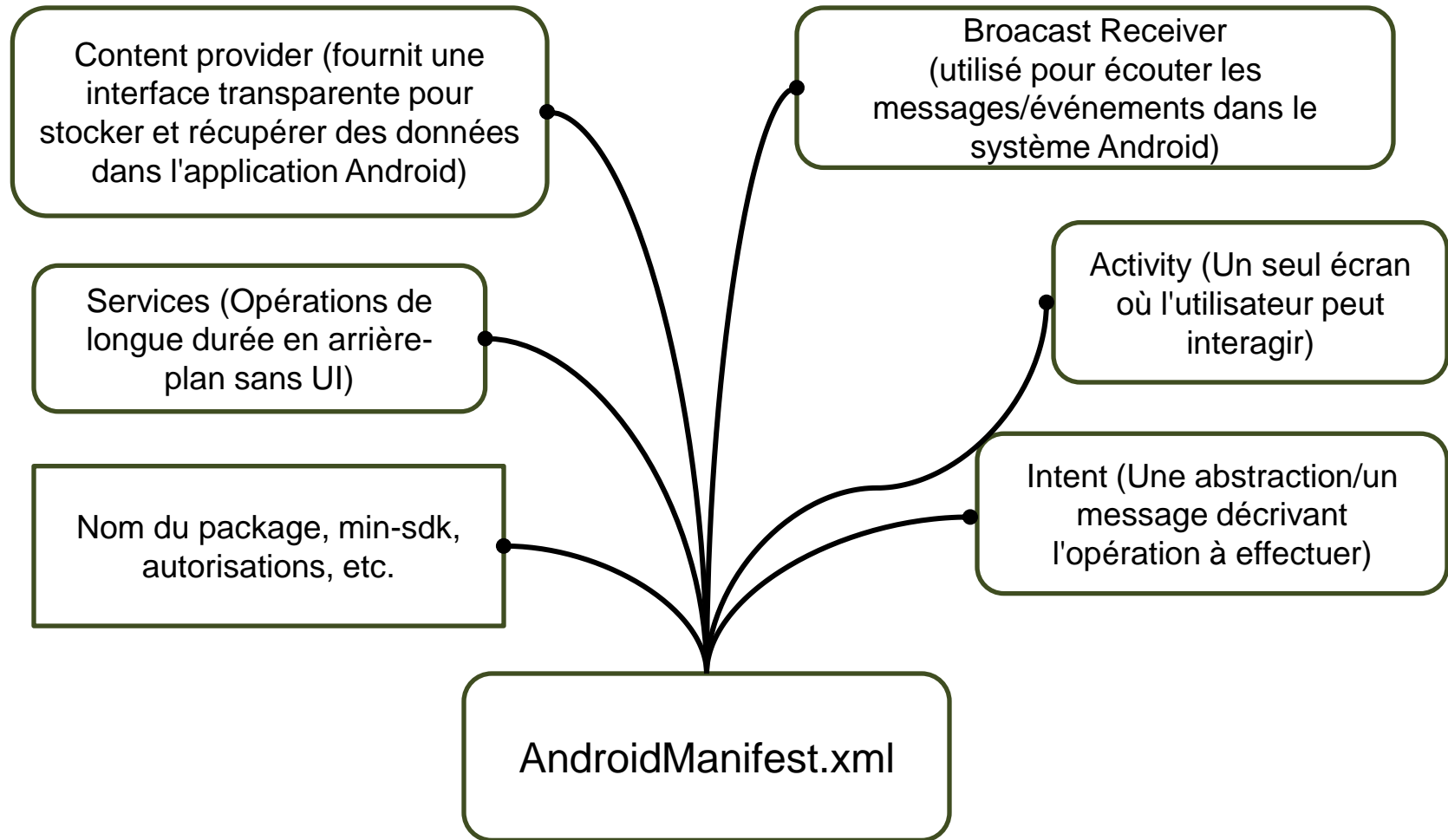
Structure de projet d'une application Android



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">
```

1 Manifest element

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

2 Uses-sdk element

```
<application android:label="@string/app_name"  
    android:debuggable="true"  
    android:icon="@drawable/ic_launcher">
```

3 Application element

```
<activity android:name="MainActivity"  
    android:label="@string/app_name">
```

4 Activity element

```
<intent-filter>
```

```
<action android:name="android.intent.action.MAIN"/>
```

```
<category android:name="android.intent.category.LAUNCHER"/>
```

```
</intent-filter>
```

```
</activity>
```

```
<service>...</service>
```

5 Service element

```
<receiver>...</receiver>
```

6 Receiver element

```
<provider>...</provider>
```

7 Provider element

```
</application>
```

```
</manifest>
```

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

