

Développement des Applications mobiles



Architecture du Cours & Sources

❖ Cours

1. Introduction
 - a. Généralités
 - b. Application Mobile
2. Android Studio
 - a. Architecture
 - b. Environnement de développement
 - c. Structure d'un projet Android
 - d. Composants d'une application Android
 - e. Cycle de vie d'une activité

3.Création d'IHM

4.Element d'Interaction

- a. Inter process communication IPC (Intent, Service)
- b. Persistance des données
- c. Connectivité réseau, géolocalisation, Bluetooth

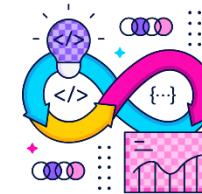
❖ TP

❖ Project DEV

❖ Développement Mobile



Cours magistraux



Expériences en DevOps



<https://developer.android.com/docs>

1 Introduction

1. Introduction

a. Généralités

- ❖ L'industrie des communications sans fil a connu une croissance exponentielle

Nombre total d'abonnés au téléphone cellulaire augmente de plus en plus.

- ❖ Au monde (en 67,1 % de la population mondiale utilise un mobile)
- ❖ Au Maroc ANRT en premier trimestre 2021) : ~ 49,47 millions d'abonnés en 1 er trimestre de 2021

L'émergence rapide d'Internet

- ❖ Les gens sont devenus de plus en plus dépendants des informations disponibles sur le Net
- ❖ Au Monde : 62,5 % de la population mondiale utilise Internet,
- ❖ Au Maroc : 33,86 millions d'abonnés Internet 3eme trimestre de 2021

1. Introduction

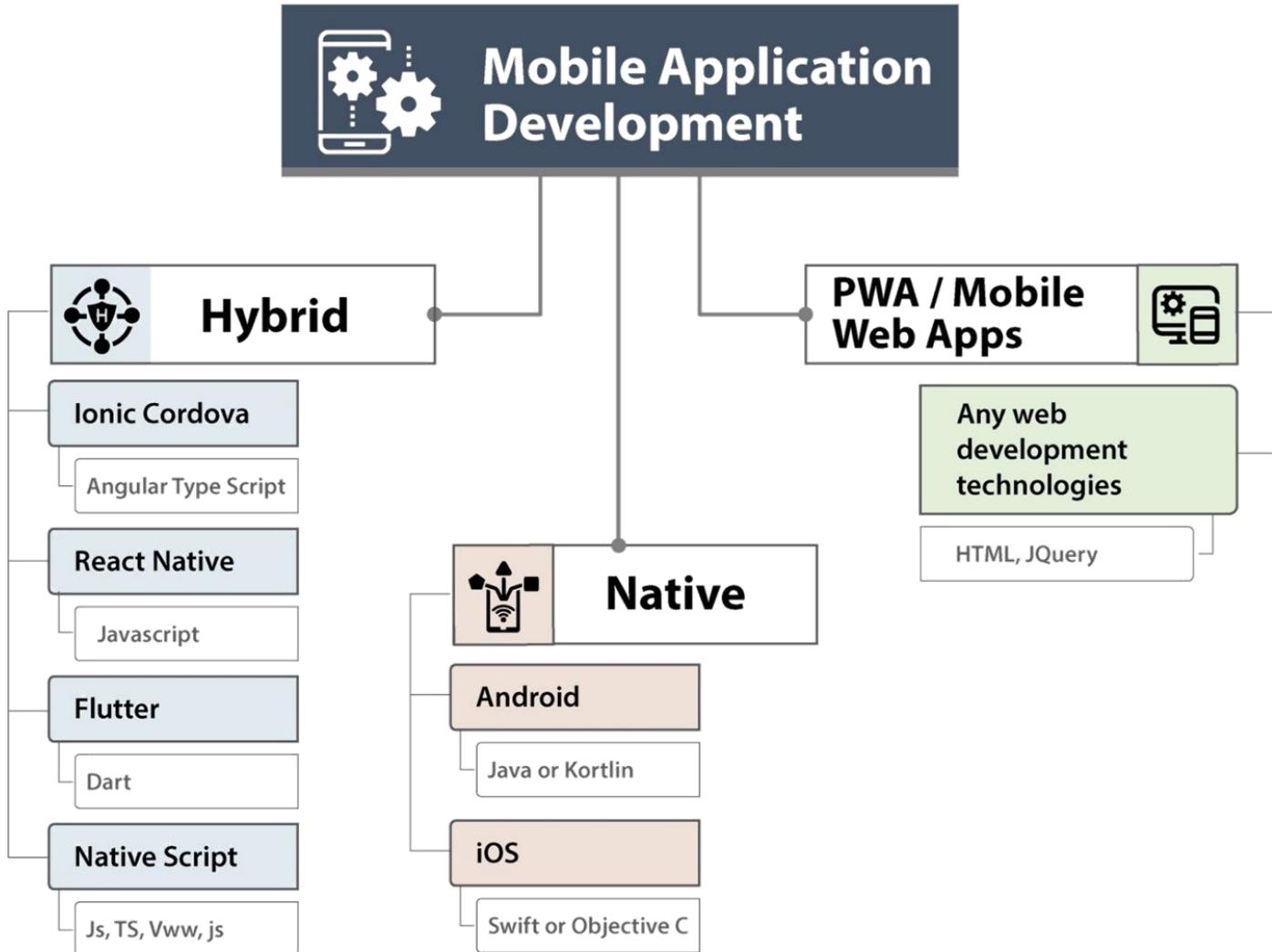
a. Généralités

- ❖ **Les applications locales** qui peuvent s'exécuter sur des terminaux sans couverture réseau(jeux autonomes, calculatrices...)
- ❖ **Les applications fonctionnant en réseau peer to peer** qui sont des applications réseau terminal à terminal sans serveur (chat en direct, jeux en réseau...).
- ❖ **Les applications client serveur (client léger)** dans lesquelles le terminal côté client a une logique applicative limitée (ex: applications qui mettent en forme du contenu fourni par une application serveur)
- ❖ **Les applications client serveur (client lourd)** c'est l'ensemble des applications dont leur exécution demande une très grande ressource au niveau du client.

1. Introduction

a. Généralités

Catégories



Source: <https://xebia.com/blog/mobile-app-development-choices/>

1. Introduction

a. Généralités

Systèmes d'exploitation pour mobile

- ❖ 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 mélange de système d'exploitation informatique avec quelques fonctionnalités supplémentaires. En outre, ils sont relativement légers et simples.
- ❖ 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.

1. Introduction

a. Généralités

Systèmes d'exploitation pour mobile

Gestion de la mémoire

- ❖ Quel que soit le programme exécuté, il doit être présent dans la mémoire principale. Par conséquent, il peut y avoir plus d'un programme présent à la fois ce qui rend nécessaire de gérer la mémoire. Le système d'exploitation alloue et désalloue la mémoire pendant le multitraitemet.

Gestion du processeur

- ❖ Lorsque plus d'un processus s'exécute, le système d'exploitation décide comment et quand un processus utilisera le processeur. Le système d'exploitation Alloue et désalloue le processeur aux processus.

Gestion des dispositifs

- ❖ Les processus peuvent nécessiter des dispositifs pour leur fonctionnement tels que les capteurs et les périphériques d'entrées sorties. Cette gestion est effectuée par l'OS. Le système d'exploitation alloue et désalloue des dispositifs à différents processus.

1. Introduction

a. Généralités

Systèmes d'exploitation pour mobile

Gestion des fichiers

- ❖ Les fichiers d'un système sont stockés dans différents répertoires. Le système d'exploitation : - Gère des registres de l'état et de l'emplacement des fichiers. - Alloue et désalloue des ressources.

Sécurité

- ❖ Le système d'exploitation assure la sécurité du système et des programmes grâce à l'authentification d'une part, et la gestion des permissions de l'utilisation des ressources d'une autre part.

1. Introduction

a. Généralités

Systèmes d'exploitation pour mobile

OS pour Mobile

Android

Le système d'exploitation Android est le plus utilisé parmi tous les systèmes d'exploitation mobiles qui existent. Il est gratuit et open source basé sur le noyau Linux développé par Google.

Apple iOS

C'est l'un des OS les plus populaires après Android. Il est conçu pour fonctionner sur les appareils Apple tels que les iPhones, les tablettes iPad, etc. De plus, comme dans le cas du système Android, IOS permet le téléchargement de ses applications via son AppStore.

1. Introduction

a. Généralités

Systèmes d'exploitation pour mobile

OS pour Mobile

Système d'exploitation BlackBerry

Le développeur de ce système d'exploitation est Research In Motion (RIM). Il a été spécialement conçu pour les appareils BlackBerry.

Harmony OS

C'est un OS conçu par la société chinoise Huawei. Il est spécialement conçu pour être utilisé dans les appareils IoT en incluant les téléphones portables et tous ce qui est mobile.

Système d'exploitation Windows Mobile

Le développeur de ce système d'exploitation est Microsoft. Il est essentiellement conçu pour les Pocket PC et les smartphones fabriqués par la société Microsoft

1. Introduction

b. Application Mobile

- ❖ Application mobile (app mobile) est un programme informatique ou une application logicielle conçue pour s'exécuter sur un appareil mobile tel qu'un téléphone, une tablette, téléviseur, une montre, etc.
- ❖ Applications mobiles contrastent souvent avec les applications de bureau conçues pour s'exécuter sur des ordinateurs de bureau et les applications Web qui s'exécutent dans les navigateurs Web mobiles plutôt que directement sur l'appareil mobile.
- ❖ Il ne suffit pas de fournir un produit ou un service - les concepteurs doivent vraiment comprendre les utilisateurs finaux pour offrir et améliorer continuellement l'expérience utilisateur.

1. Introduction

b. Application Mobile

- ❖ Les applications mobiles sont généralement classées en trois types : natives, hybrides, et Web.
- ❖ **Applications natives** sont conçues spécifiquement pour un système d'exploitation mobile, généralement iOS ou Android.
- ❖ **Applications Web (Progressive Web App: PWA)** sont écrites en HTML5 et CSS et s'exécutent généralement via un navigateur.
- ❖ **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.

1. Introduction

b. Application Mobile

Comparaison des Applications Mobiles

	Web app	Application native	Application hybride
Compatibilité sur les plateformes	(+) Compatible sur tous les terminaux	(-) Compatible sur 1 seule plateforme	(+) Compatible sur tous les terminaux (nécessite une configuration particulière pour chacune)
Expérience utilisateur	(-) Limitée (même ergonomie et graphisme pour toutes les plateformes)	(+) Sur mesure	(-) Limitée (même ergonomie et graphisme pour toutes les plateformes)
Performances	(-) Réduites (dépend de la qualité du réseau et du navigateur utilisé)	(+) Optimales	(-) Réduites (dépend de la qualité du réseau)
Hors connexion	(-) Non disponible	(+) Disponible	(+) Disponible
Distribution et mise à jour	Via Internet (+) Mises à jour transparentes (-) Fidélisation plus compliquée qui influe sur le trafic de l'application	Sur un store (-) Nécessite une re-publication et validation à chaque mise à jour (un peu long pour iOS)	Sur un store (+) Mises à jour transparentes pour la partie web
Monétisation	(-) Impossible de vendre le téléchargement de l'application (+) Publicité	(+) L'application peut être payante sur le store (mais un pourcentage est prélevé)	(+) L'application peut être payante sur le store (mais un pourcentage est prélevé)
		(+) Achats in-app	

1. Introduction

b. Application Mobile

En tant que développeur, il faut trouver des moyens créatifs pour combiner toutes les fonctionnalités suivantes dans un seul projet:

- **Hautes performances** : utile pour garder les utilisateurs connectés et augmenter la base de données.
- **Rentabilité** : développer une application prend du temps, et le temps coûte cher. Un projet bien fait aura toujours à l'esprit le coût de production et la portée du marché potentiel.
- **Évolutivité** : le monde des applications mobiles est en constante évolution. De nouvelles mises à jour des systèmes d'exploitation sortent chaque jour ce qui nécessite que les applications à développer doivent être évolutives.

1. Introduction

b. Application Mobile

Une application mobile est composée principalement de deux parties:

❖ **Interface de l'utilisateur :**

Visible : les composantes visibles de l'interface de l'utilisateur 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.

Invisible : 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.

1. Introduction

b. Application Mobile

Une application mobile est composée principalement de deux parties:

- ❖ **Comportement - Evènements :**

Évènement: 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.

1. Introduction

b. Application Mobile

❖ Frameworks:

Flutter : créé par Google en 2013, son langage de programmation DART a été inventé par Google en 2011. Flutter est présenté comme étant le successeur de JavaScript en mieux. Depuis sa 1ere version en 2018, Flutter connaît une croissance d'utilisation de 279 % [4]. Comme exemple d'applications conçues avec Flutter, on peut citer Alibaba.

Avantages:

- **Simplicité** : Flutter est facile à apprendre car il est basé sur Dart qui ressemble un peu à Javascript. Il produit des applications compatibles avec les deux plateformes Android et IOS à partir d'une base de code unique.
- **Outils et fonctions avancés** : Flutter permet de bénéficier de nombreux outils de développement comme la fonction de recharge à chaud intégrée qui permet de voir instantanément les modifications apportées à l'application sans avoir à la recompiler.
- **Utilisable pour autres plateformes non mobiles**: en mai 2019, Google a annoncé une nouvelle prise en charge de Flutter pour les applications de bureau, intégrées, mobiles et Web à partir de la même base de code.

1. Introduction

b. Application Mobile

❖ Frameworks:

Flutter : créé par Google en 2013, son langage de programmation DART a été inventé par Google en 2011. Flutter est présenté comme étant le successeur de JavaScript en mieux. Depuis sa 1ere version en 2018, Flutter connaît une croissance d'utilisation de 279 % [4]. Comme exemple d'applications conçues avec Flutter, on peut citer Alibaba.

Inconvénients:

- **Des mise à jours fréquentes** : cela rend plus difficile le suivi de toutes les mises à jour du framework.
- **Dart** : ce langage est encore très nouveau. Il peut donc être difficile de trouver un personnel suffisamment qualifiées pour créer une application avec.
- **Les applications Flutter sont volumineuses** : en raison de leurs widgets intégrés, les applications Flutter sont plus volumineuses que leurs homologues natives. Étant donné que les utilisateurs ont une mémoire limitée sur leurs appareils, il faut toujours en tenir compte de cette contrainte.

1. Introduction

b. Application Mobile

❖ Frameworks:

React Native : a été créé par Facebook (2015). Il est basé sur React qui est une librairie JavaScript, langage habituellement utilisé pour le Web. Facebook, Instagram, Skype ou Airbnb ont été réalisés en partie ou entièrement grâce à cette technologie.

Avantages:

- **Apprentissage facile** : comme il est basé sur JavaScript ou React, l'utilisation de ce framework est simple. Il n'y a rien à apprendre de nouveau à part gérer les composants personnalisés offerts par cet outil.
- **Idéal pour la productivité** : React Native est livré avec des fonctionnalités utiles telles que le rechargement à chaud et les outils de développement Chrome pour le débogage des applications.
- **Performances** : les performances des applications créées à l'aide de React Native sont proches de celles d'un code Natif.
- **Une communauté importante** : plus de 1500 contributeurs soutiennent cet outil, et de grandes entreprises technologiques l'utilisent comme Airbnb, Uber et Tesla.

1. Introduction

b. Application Mobile

❖ Frameworks:

React Native : a été créé par Facebook (2015). Il est basé sur React qui est une librairie JavaScript, langage habituellement utilisé pour le Web. Facebook, Instagram, Skype ou Airbnb ont été réalisés en partie ou entièrement grâce à cette technologie.

Inconvénients:

- **Peu de composants tiers** : ce projet étant encore nouveau, il n'y a pas autant de composants tiers que parfois, on est obligé de mettre en œuvre une solution personnalisée à partir de zéro.

1. Introduction

b. Application Mobile

❖ Frameworks:

Ionic: Ionic est un framework gratuit créé en 2013 qui utilise les technologies web comme HTML, CSS ou JavaScript [5]. Il s'appuie sur AngularJS, Cordova et Node JS. IONIC est utilisé pour développer des applications mobiles et aussi des applications web comme le cas de Flutter.

Avantages:

- **Prise en charge de différentes technologies :** à partir d'Ionic 4, il est possible d'utiliser n'importe quel framework frontend pour créer des applications mobiles.
- **Facile :** il se base sur HTML, CSS et JavaScript, donc l'utilisation de Ionic ne nécessite pas beaucoup d'effort pour une personne qui a été déjà formé au développement Web.

Inconvénients

- **Ne convient pas aux applications volumineuses :** Malheureusement, pour les applications volumineuses, l'utilisation de la composante WebView qui est très utile pour Ionic peut entraîner un manque de performances important.
- **Peut manquer de prise en charge des plugins natifs :** certains plugins natifs ne sont pas stables et peuvent entrer en conflit les uns avec les autres. De plus, ils peuvent être complètement absents dans la liste des plugins d'une distribution.

1. Introduction

b. Application Mobile

❖ Plateformes:

Android

Java/Kotlin
Eclipse
Android Studio



iOS

Objective C
Swift
X-Code



Windows Mobile

C# et C++
Visual Studio



2 Android Studio

2. Android Studio

a. Architecture

Android

est un système d'exploitation Open Source pour smartphones, PDA et terminaux mobiles. 2003 : naissance de la startup Android Inc qui s'intéressait au développement mobile (fondée par Andy Rubin).

En juillet 2005, Google a racheté la petite startup qui développait des applications pour téléphones mobiles « Android, Inc » En 2007, le 5 novembre, l'OHA (l'Open Handset Alliance) a été officiellement annoncée rassemblant plus que 34 acteurs:

- d'opérateur mobile (Vodafone, Teleponica, Telecom Italia, China Mobile...)
- de fabricants de téléphone mobiles (Asus, HTC, LG, Motorola...)
- de fabricants de semi conducteur (Intel, Nvidia, ARM...)
- d'éditeurs logiciels (Ebay, Google, PacketVideo...)
- de distributeurs (Aplix corporation, Borqs, TAT...)

2. Android Studio

a. Architecture

Android

OHA a comme objectif de développer des standards open source pour appareil mobile.

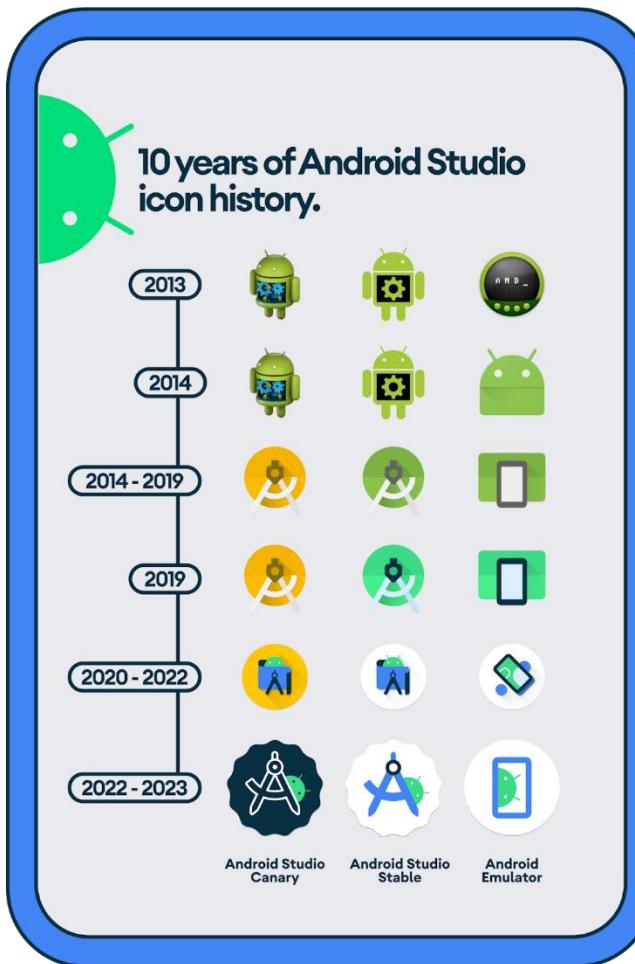
<http://www.openhandsetalliance.com>

- En 2007 OHA a annoncé le développement d'Android, la première plate forme ouverte et complète pour terminaux mobiles (téléphone, PDA, etc).
- Cette plate forme est composée d'un système d'exploitation, de librairies "middleware", et d'un ensemble d'applications : un client mail, un navigateur, un calendrier, ...

2. Android Studio

a. Architecture

Versions



Source: <https://android-developers.googleblog.com/2023/05/redesigning-android-studio-logo.html>

2. Android Studio

a. Architecture

1. Noyau Linux :

1. Le fondement du système d'exploitation Android est le noyau Linux. Il fournit des services système de base tels que la gestion de la mémoire, la gestion des processus, les pilotes de périphériques et la sécurité.

2. Couche d'abstraction matérielle (HAL) :

1. La HAL repose sur le noyau Linux et fournit une interface normalisée pour les pilotes matériels. Cette couche permet à la plateforme Android d'être compatible avec une large gamme de composants matériels sans nécessiter de modifications aux couches supérieures.

3. Bibliothèques natives :

1. Ce sont des bibliothèques écrites en C ou C++ qui fournissent des fonctionnalités de base au système Android. Exemples :
 1. libc : Bibliothèque C standard.
 2. Surface Manager : Gère le sous-système d'affichage.
 3. Cadre multimédia : Gère les tâches multimédias telles que la lecture audio et vidéo.
 4. OpenGL ES : Bibliothèque de rendu graphique pour les graphiques 2D et 3D.



<https://developer.android.com/guide/platform>

2. Android Studio

a. Architecture

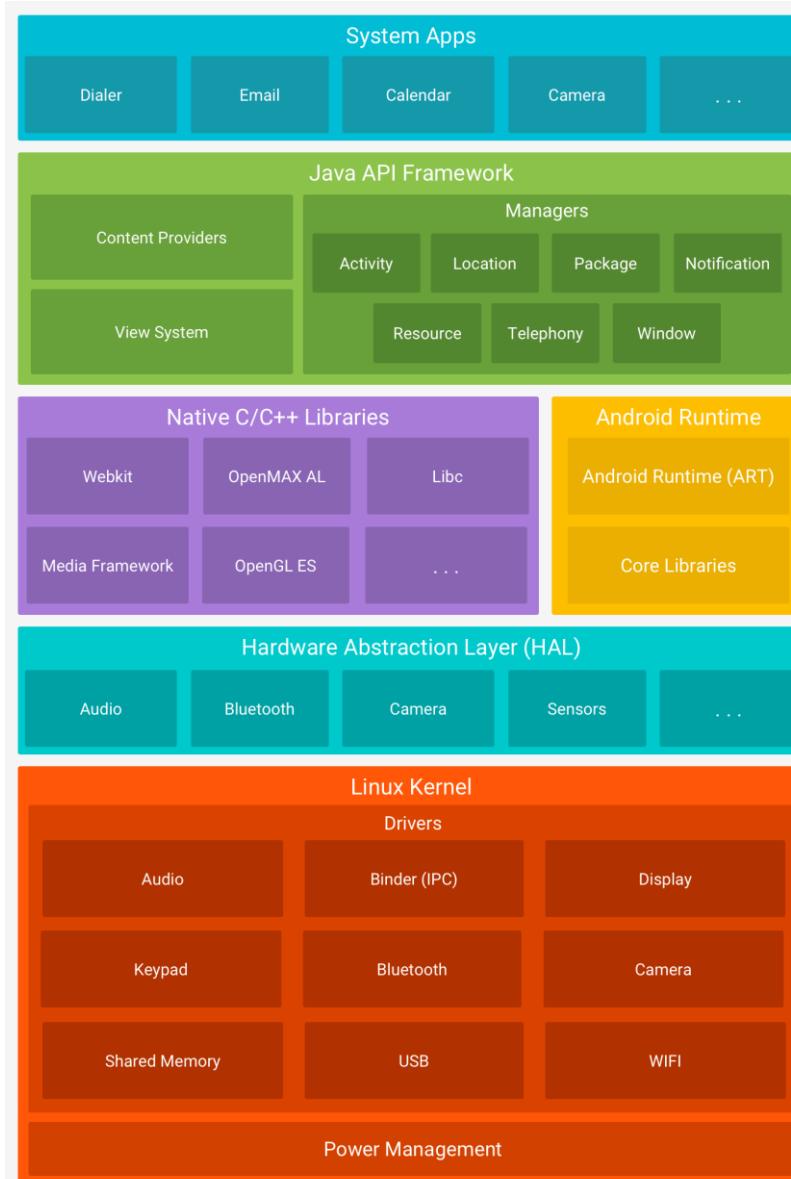
4. Runtime Android :

Cette couche inclut la machine virtuelle Dalvik ou ART (Android Runtime), chargée d'exécuter et de gérer les applications Android. Android est passé de Dalvik à ART pour améliorer les performances et l'efficacité.

5. Cadre d'application :

La couche du Cadre d'application fournit des services de haut niveau aux applications grâce à un ensemble de classes Java. Les composants clés incluent :

1. Activity Manager : Gère le cycle de vie des applications et fournit une pile de navigation commune.
2. Package Manager : Gère l'installation, la mise à niveau et la désinstallation des applications.
3. Content Providers : Permettent le partage de données entre les applications.
4. Système de vue : Fournit un ensemble de composants d'interface utilisateur utilisés pour construire l'interface utilisateur de l'application.
5. Gestionnaire de téléphonie, Gestionnaire de localisation, etc. : Fournissent l'accès à diverses fonctionnalités de l'appareil.



<https://developer.android.com/guide/platform>

2. Android Studio

a. Architecture

6. Applications système :

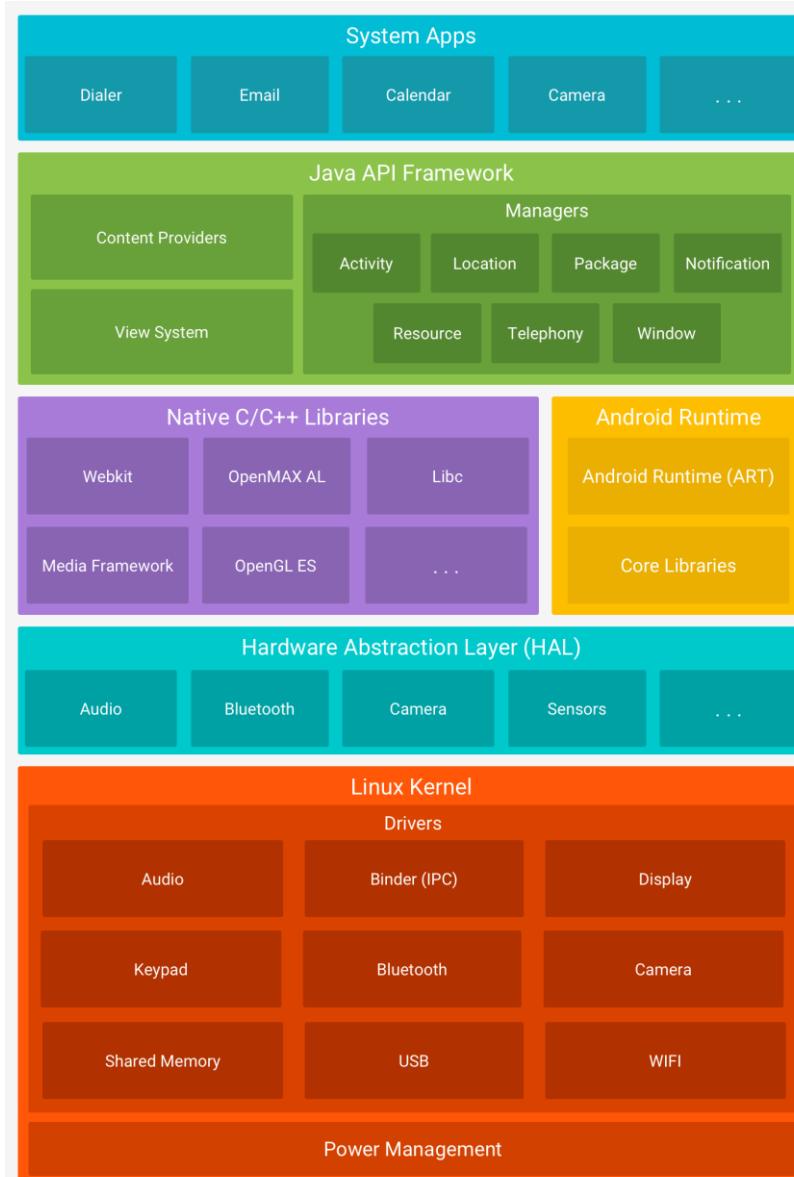
1. Ce sont des applications préinstallées qui accompagnent le système d'exploitation Android. Des exemples incluent l'application Téléphone, Contacts, Navigateur, et d'autres.

7. Interface utilisateur (UI) :

1. Cette couche représente l'interface utilisateur graphique de l'appareil, comprenant l'écran d'accueil, les widgets et divers éléments d'interface utilisateur. Les développeurs peuvent personnaliser l'interface utilisateur pour créer des expériences utilisateur uniques.

8. Applications :

1. En haut de la pile se trouvent les applications installées par l'utilisateur. Celles-ci peuvent être téléchargées depuis le Google Play Store ou d'autres sources. Les applications Android sont écrites en Java (ou Kotlin) et s'exécutent sur le Runtime Android.



<https://developer.android.com/guide/platform>

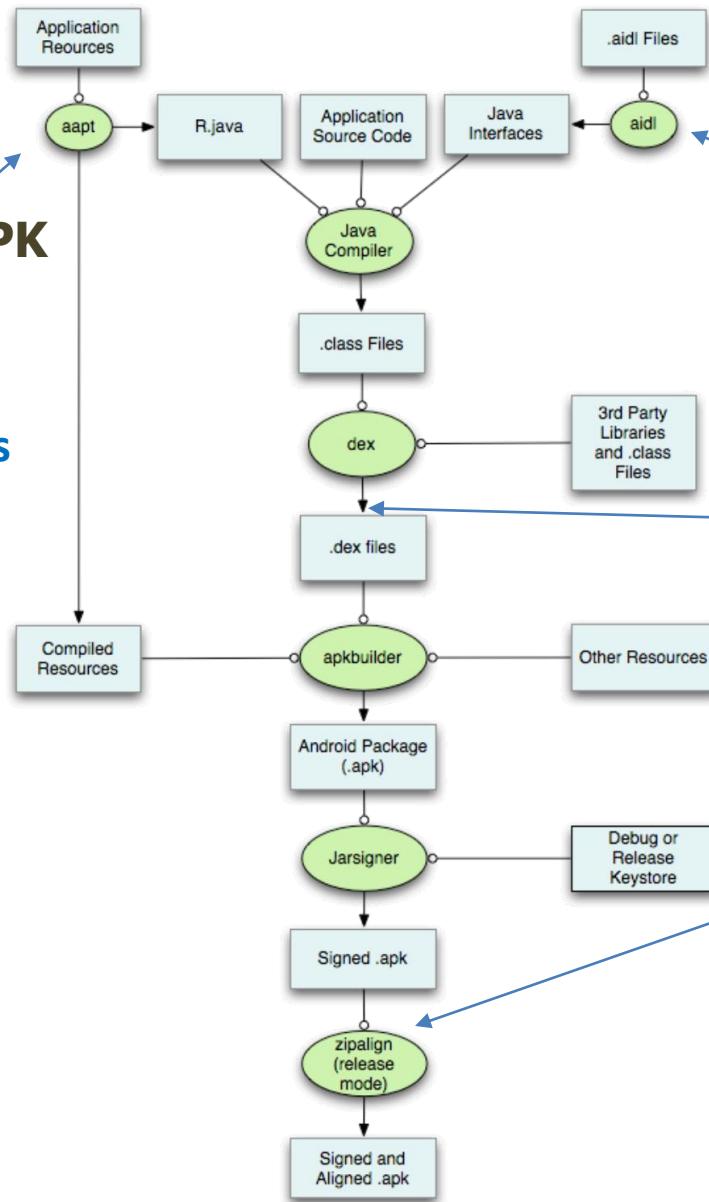
2. Android Studio

a. Architecture

Processus de génération de l'APK

Android Asset Packaging Tools

compile les fichiers de ressources :
AndroidManifest.xml les fichiers XML
pour les activités



Android Interface Definition Language

- Permet l'exposition de service
- Communication entre applications via les services

DalvikBytecode

outil d'alignement d'archives qui optimise la façon dont un package d'application Android (APK) est emballé

2. Android Studio

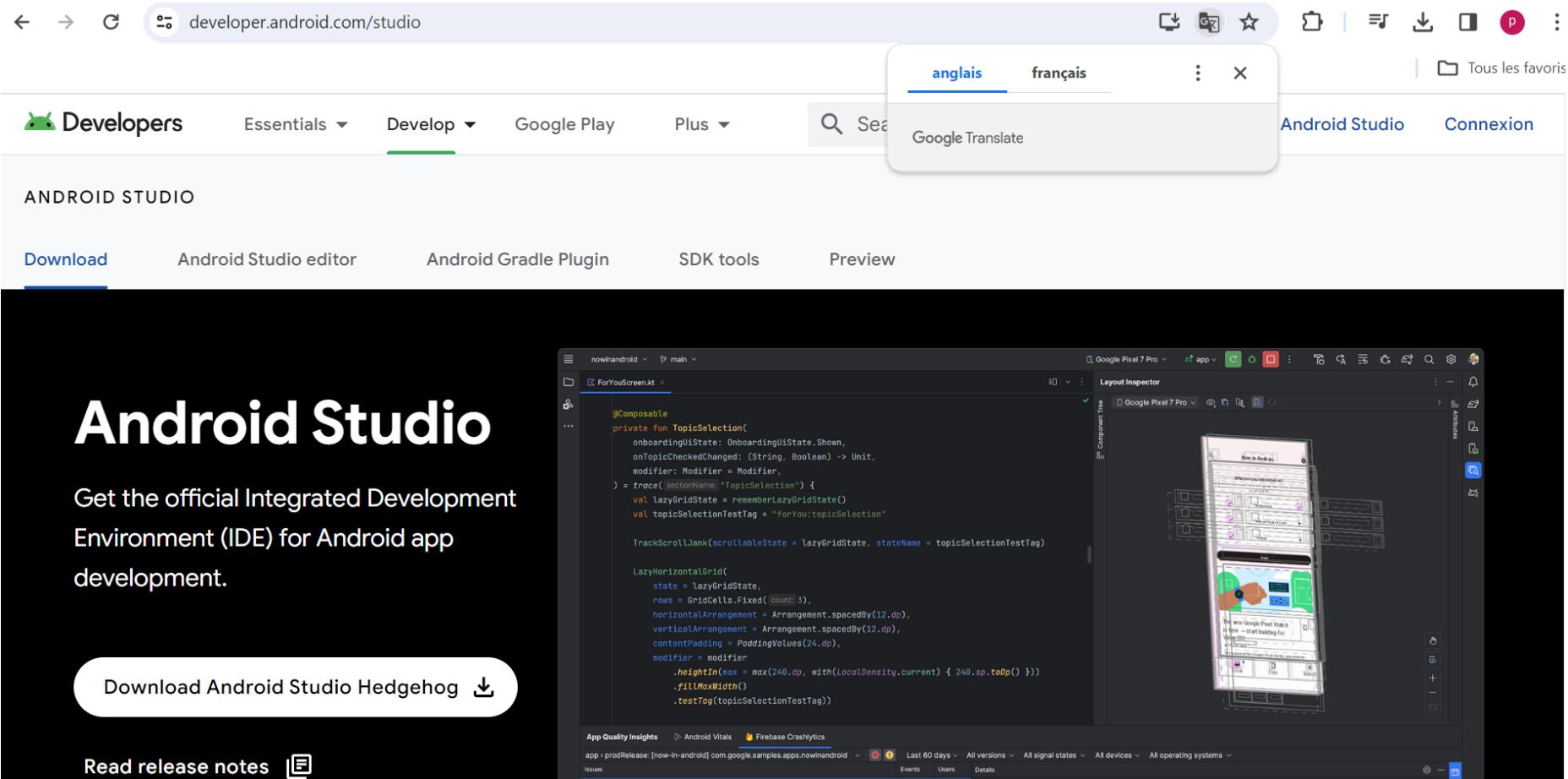
a. Architecture

Les avantages de la plateforme Android et de ses APIs

- Pas de License à obtenir,
- Développer des applications « location based » en utilisant le GPS ;
- Utiliser des cartes géographiques avec Google Maps ;
- Recevoir et émettre des SMS, envoyer et recevoir des données sur le réseau mobile ;
- Enregistrer et lire image, son et vidéo ;
- Utiliser compas, accéléromètre, gyroscope ;
- Des IPC, inter process communication, avec les Notifications et Intents pour une relation « event driven » avec l'utilisateur et l'échange de messages entre applications ;
- Des outils de stockage de données partagés (SQLite en version Sandbox) ;
- Content Provider pour partager l'accès à ses données ;
- Un navigateur que l'on peut inclure basé sur WebKit ;
- Des services et des applications qui peuvent tourner en tâche de fond : qui peuvent réagir au cours d'une action, à votre position dans la ville, à l'heure qu'il est, suivant l'identité de l'appelant ..

2. Android Studio

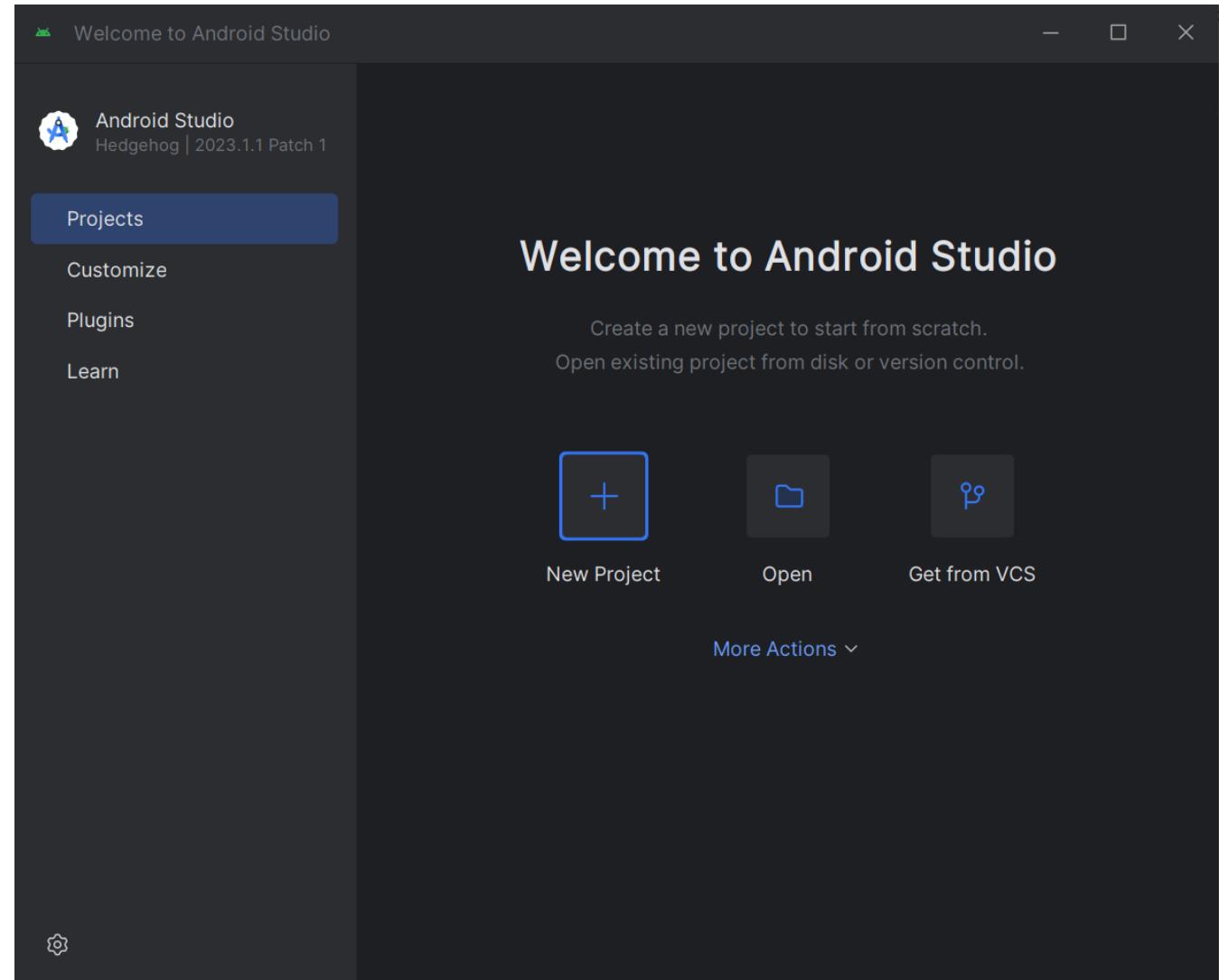
b. Environnement de développement



2. Android Studio

b. Environnement de développement

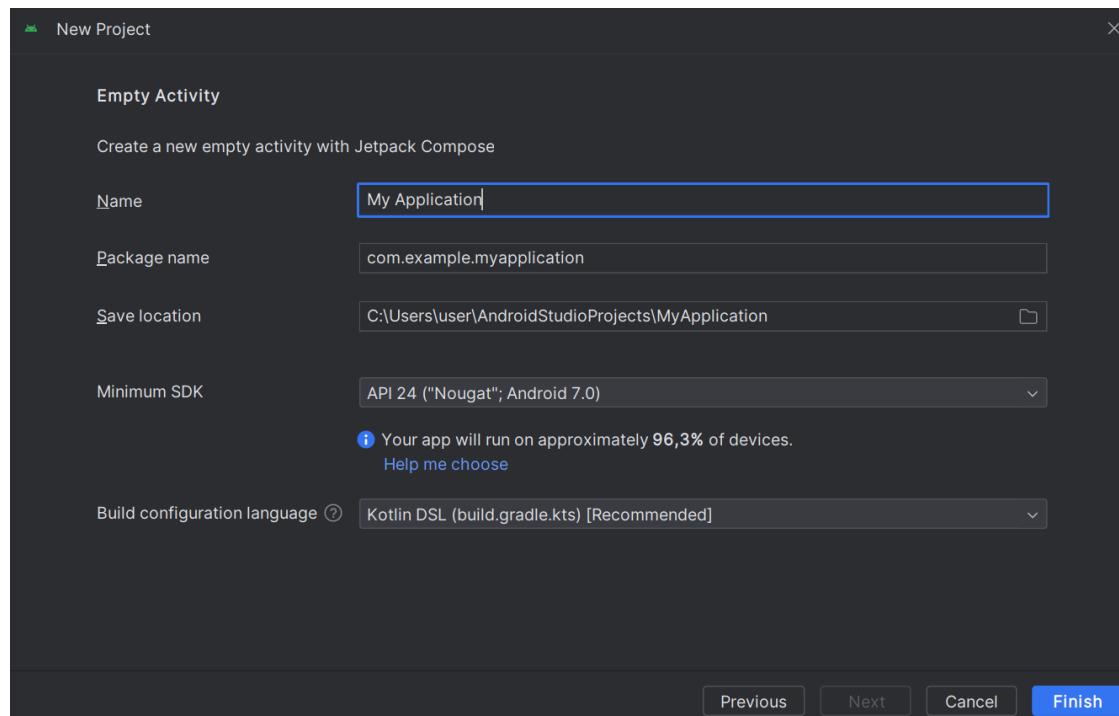
Lancer l'environnement



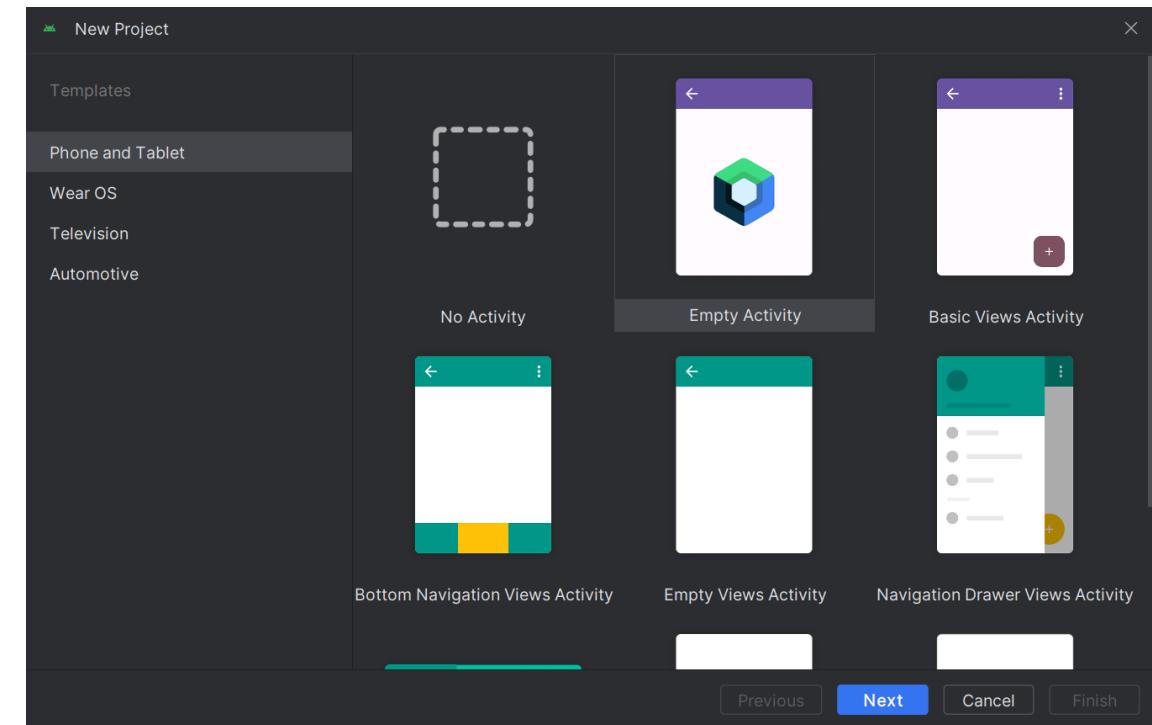
2. Android Studio

b. Environnement de développement

Nommer le projet & choisir la version



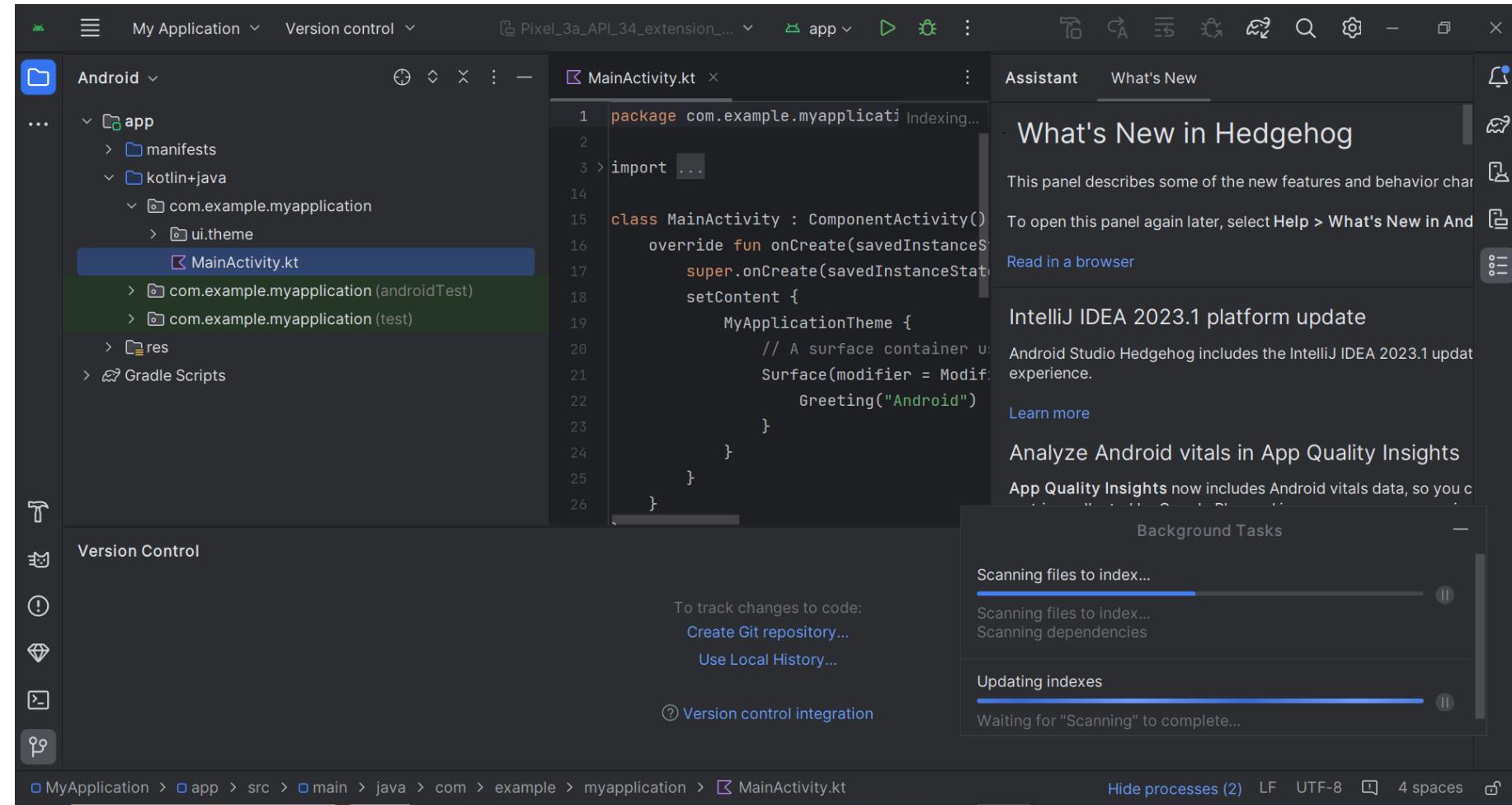
Choisir le type projet



2. Android Studio

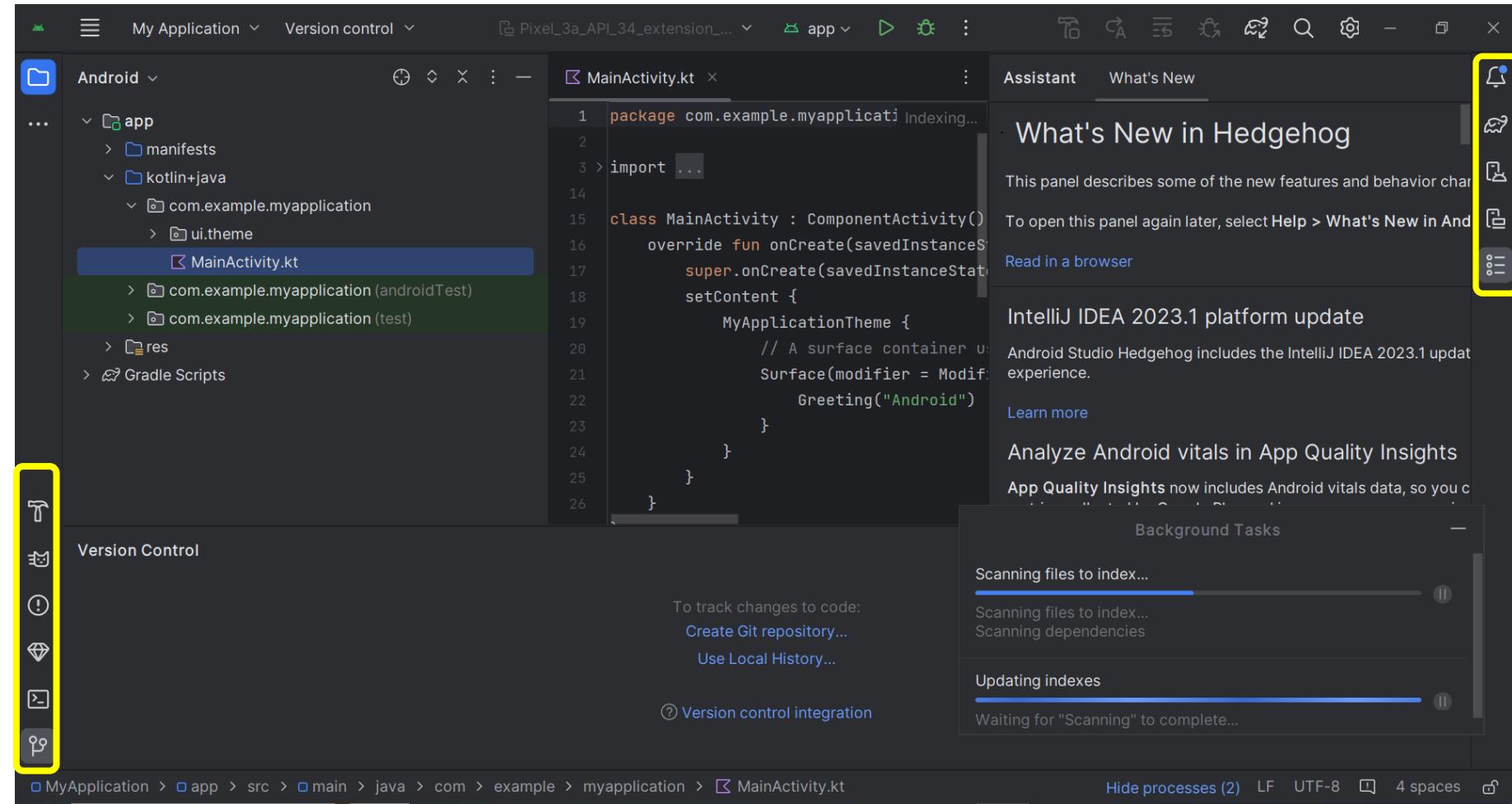
b. Environnement de développement

IDE



2. Android Studio

b. Environnement de développement



IDE

Android Virtual Device Manager

Android SDK Manager

Profile

Logs

Terminal

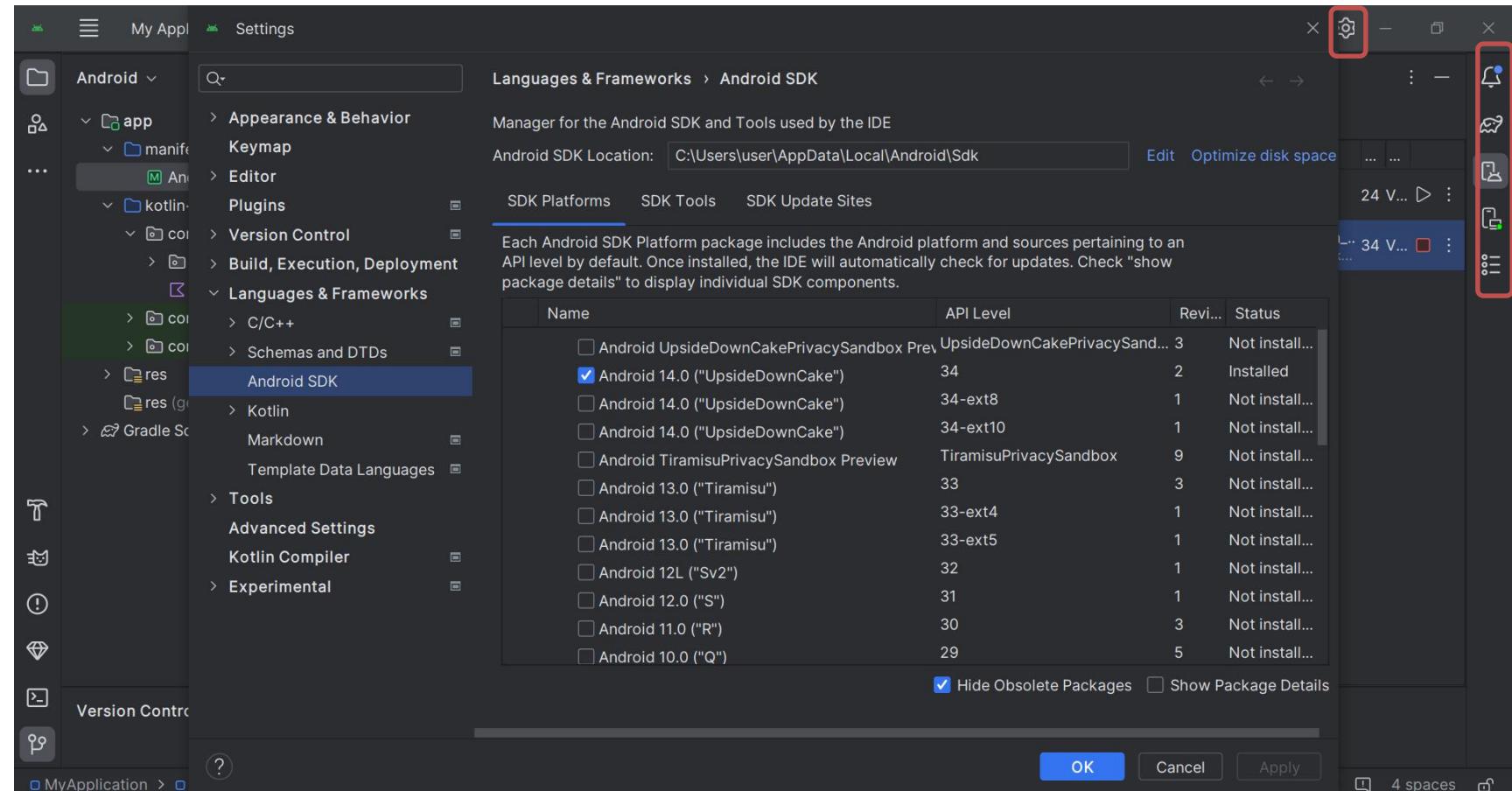
2. Android Studio

b. Environnement de développement

IDE

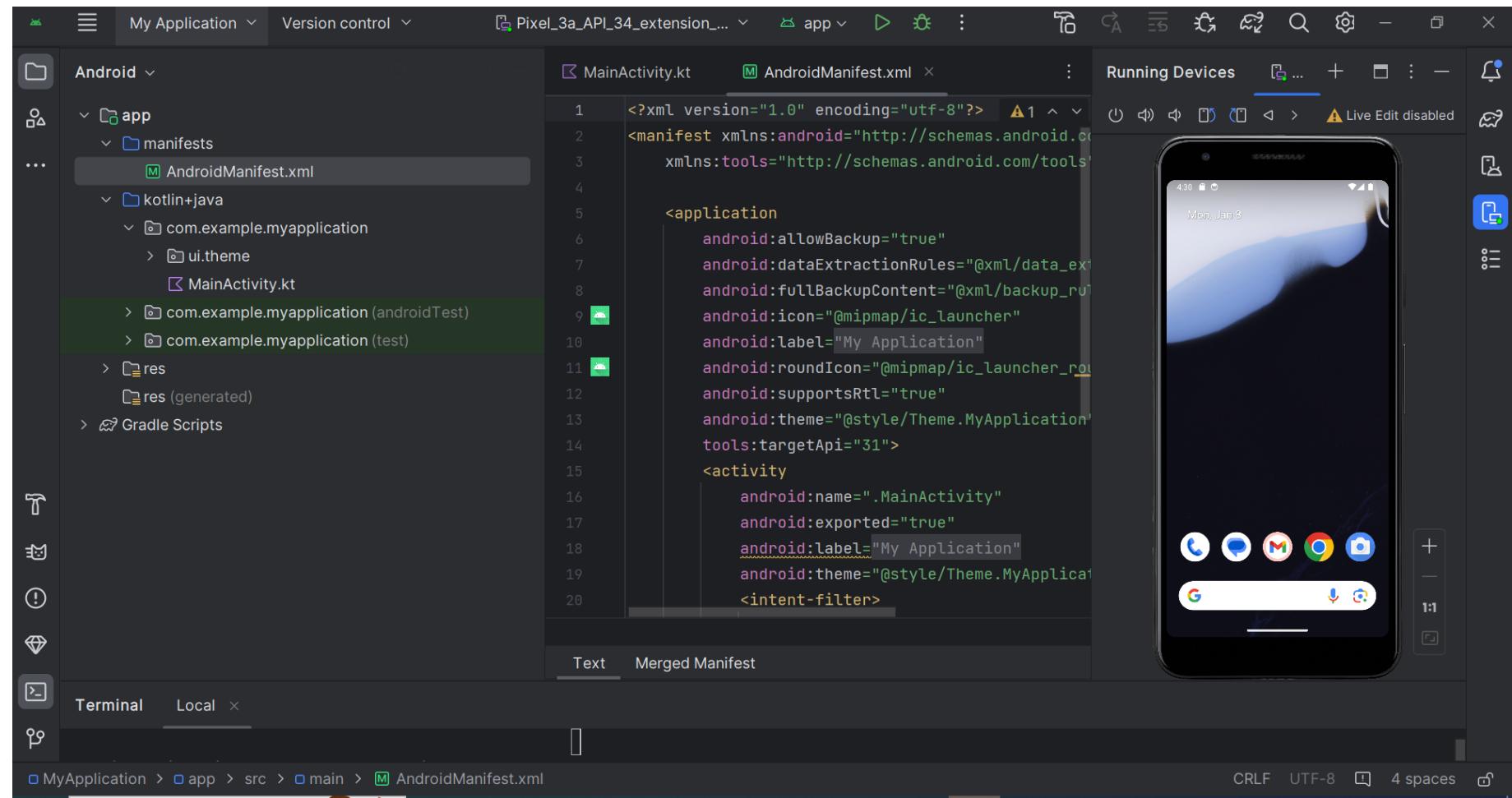
Android SDK Manager

Sélectionnez les API dont vous avez besoin



2. Android Studio

b. Environnement de développement



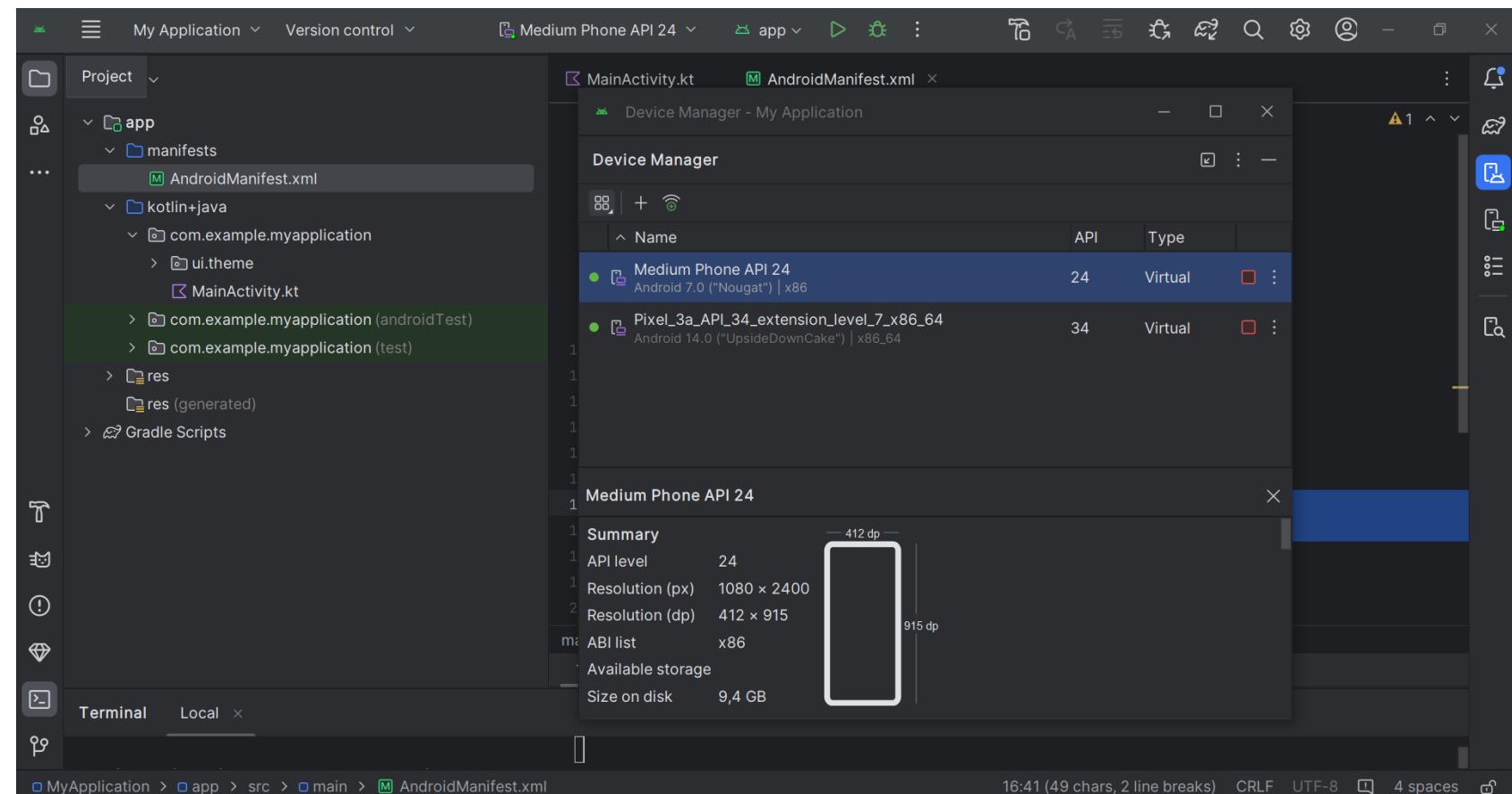
IDE

Android Virtual Device Manager

2. Android Studio

b. Environnement de développement

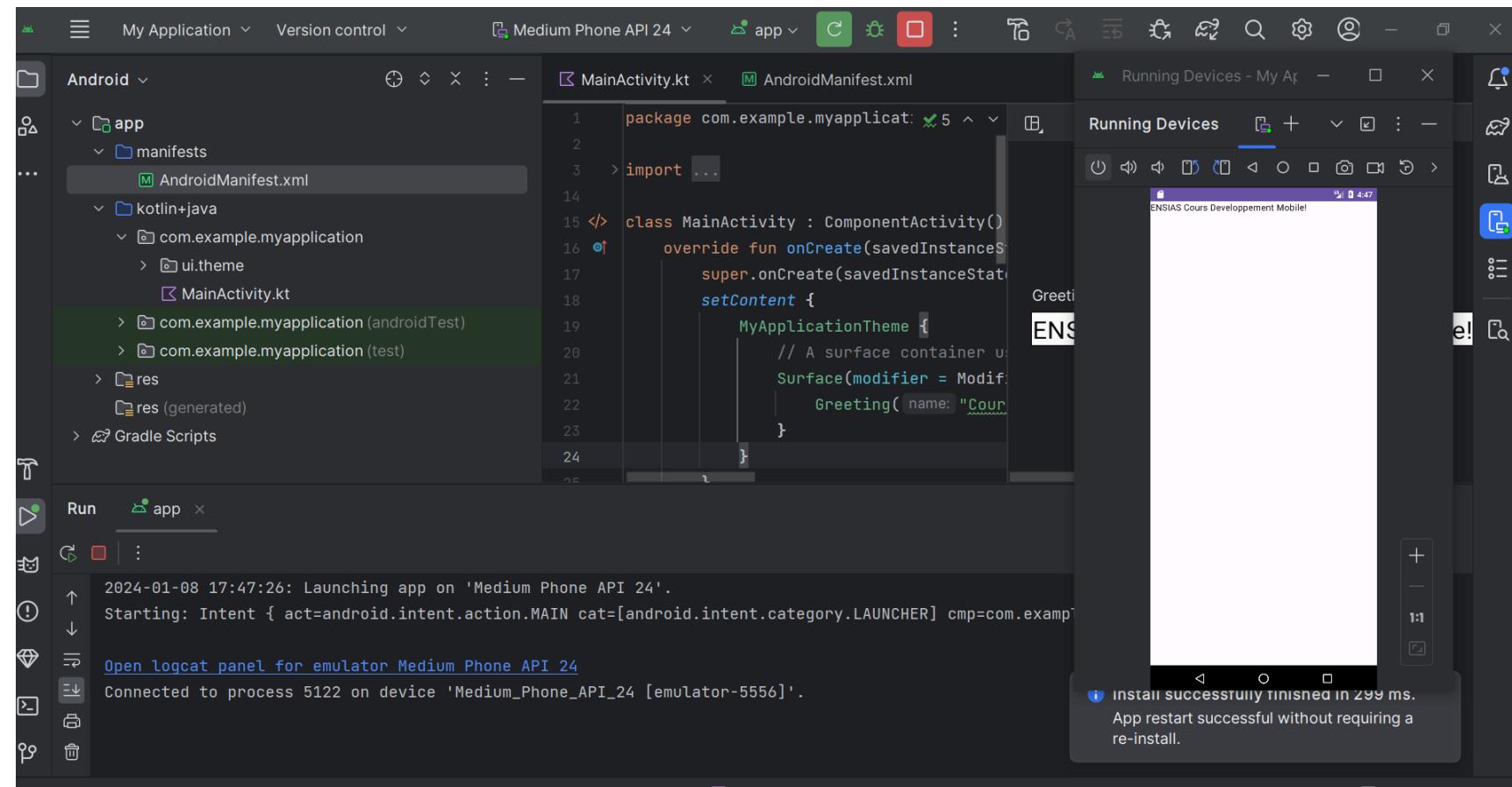
IDE
Android Virtual Device Manager



2. Android Studio

b. Environnement de développement

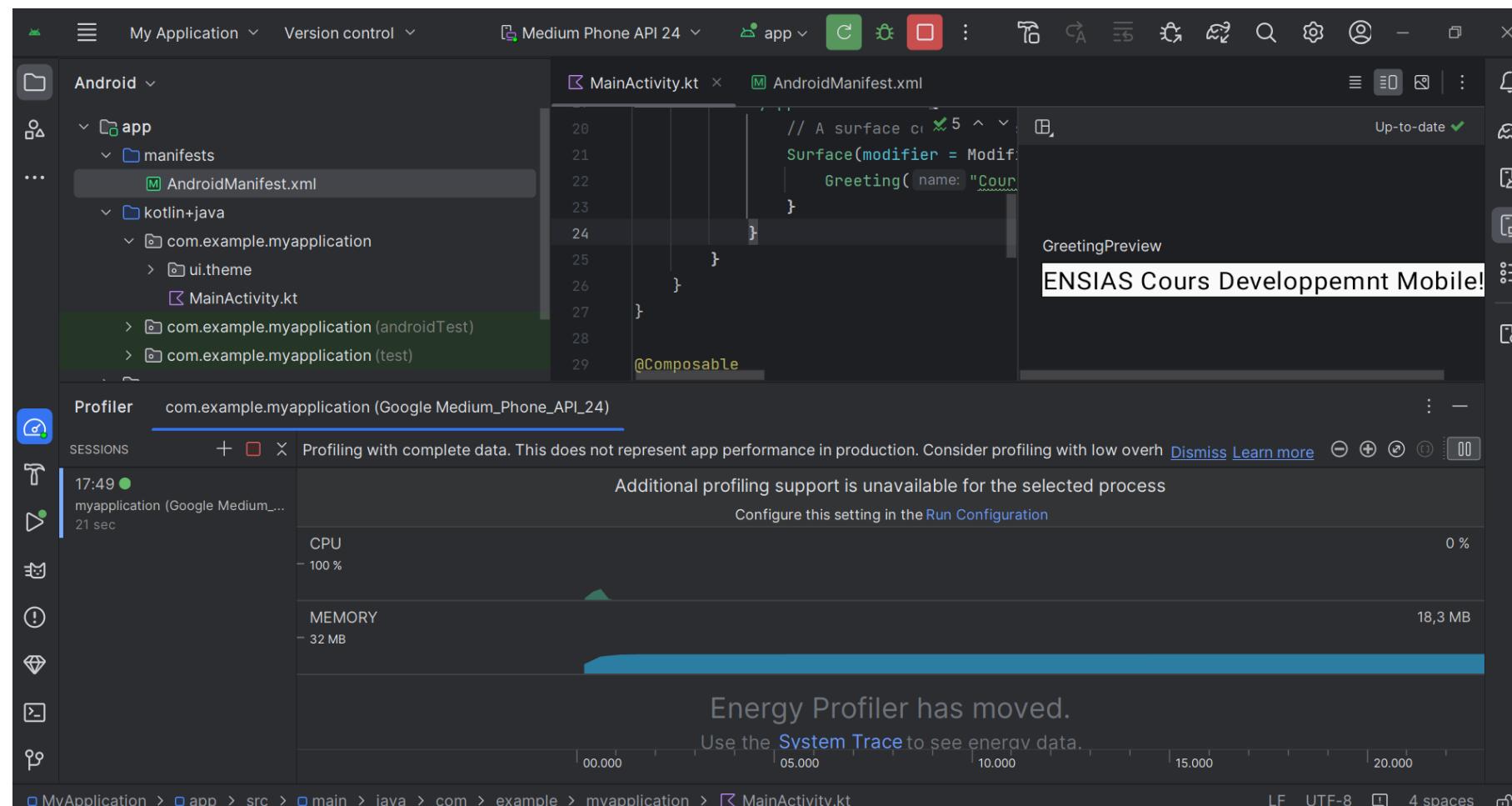
IDE
Android Virtual Device Manager



2. Android Studio

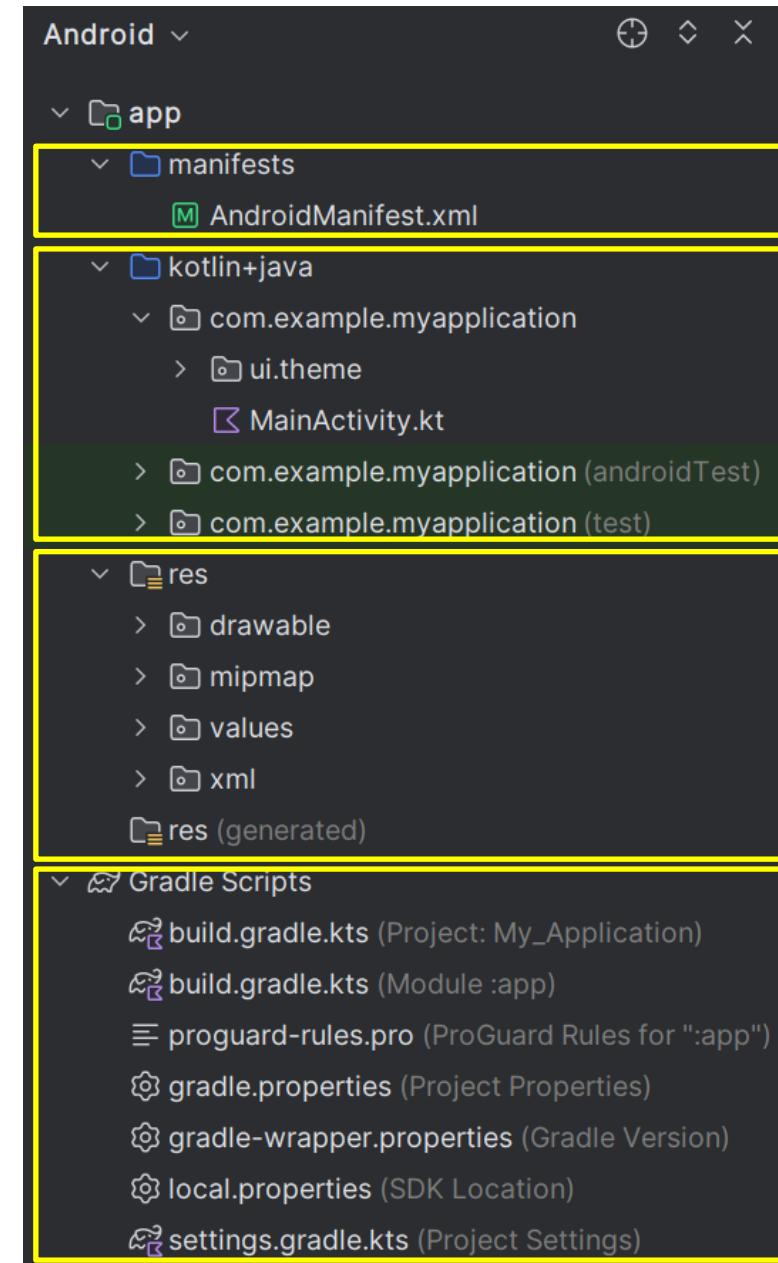
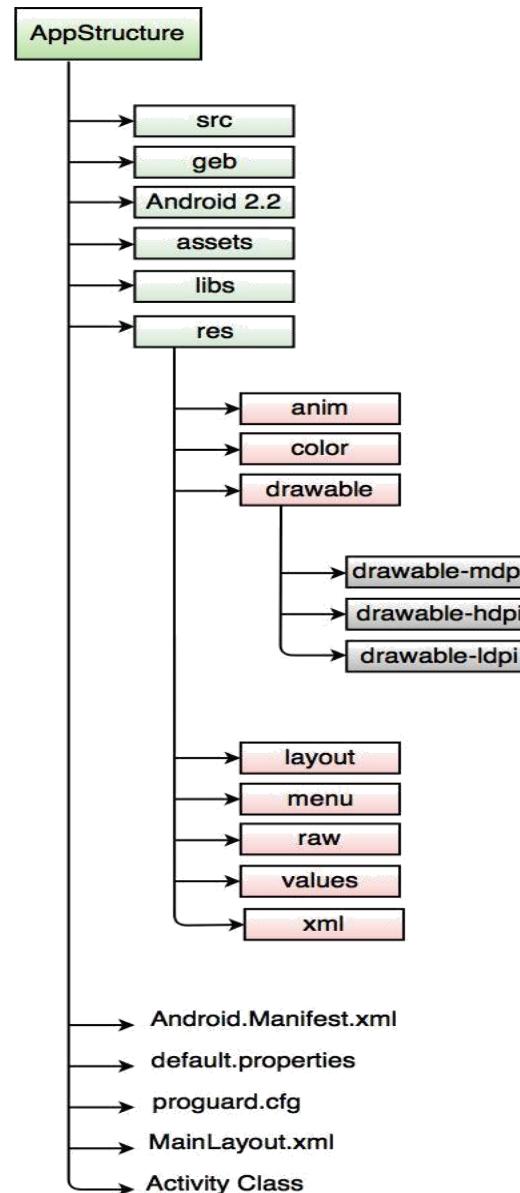
b. Environnement de développement

Aider à comprendre comment votre application utilise les ressources du processeur, de la mémoire, du réseau et de la batterie



2. Android

c. Structure du Projet Android



Manifests

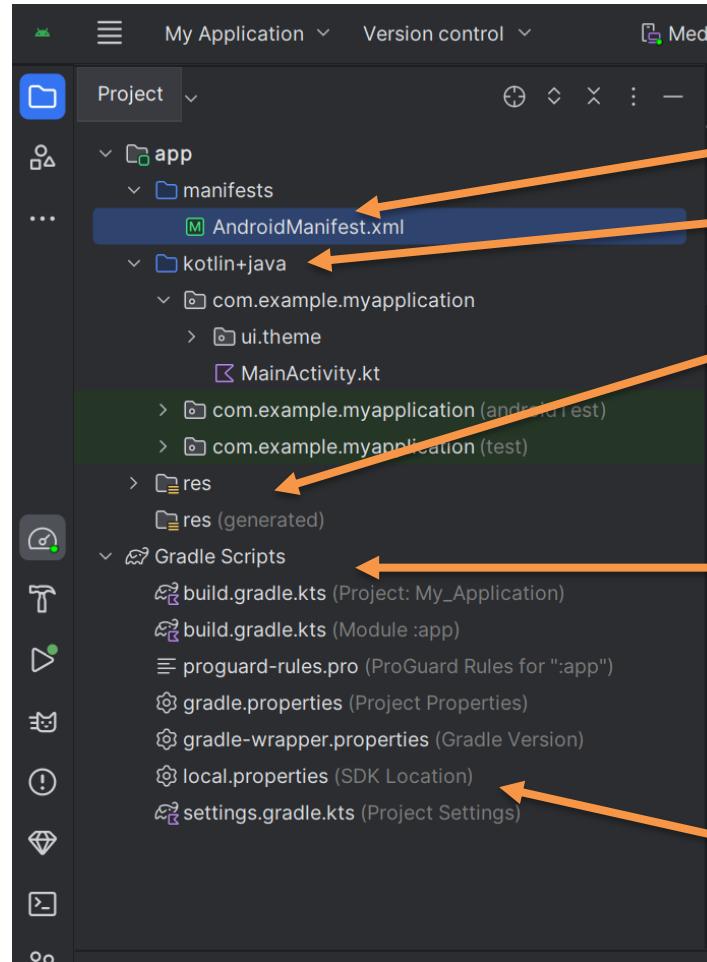
Fichiers Java rangés

Ressources:
Fichiers XML et images de l'interface

Gradle: outil de compilation de projet

2. Android Studio

c. Structure du Projet Android



Le fichier Manifest de l'application: contient un fichier de configuration de l'application

répertoire des sources

répertoire des ressources (images, descriptions d'interfaces, valeurs)

règles de compilation et génération (configuration et dépendances): la version du SDK utilisée pour la compilation, la version minimale du SDK, l'identifiant de l'application

chemin d'accès au SDK (software development kit)

2. Android Studio

c. Structure du Projet Android

Type de ressources	Répertoire associé	Description
Valeurs Simples	res/values	Fichiers XML dont le nom reflète le type de ressources contenus: - Array.xml - String.xml - ...
Drawables	res/drawable	- drawable-hdpi: images en haute définition - drawable-ldpi: images en basse définition - drawable-mdpi: images en moyenne définition
Layouts	res/layout	Mise en page d'écran(ou de partie d'écran) appelé gabarits
Animations	res/anim	Fichiers xml convertis en objets animation
Ressources XML	res/xml	fichiers XML utilisés directement par l'application (usage de la méthode resources.getXML)
Ressources brut	res/raw	Les autres types de ressources : sons, vidéos, ...
Menus	res/menu	Définit les propriétés des entrées pour un menu

Les types des ressources

2. Android

c. Structure du Projet Android

Les types des ressources

Classe R

• aapt(android asset packaging tool) génère, lors de la compilation de l'application , la class R qui contient les identifiants de toutes les ressources se trouvant dans les sous répertoires du « res ».

• Pour chaque type de ressource, il existe une sous classe de R (ex R.drawable) et pour chaque ressource de ce type, il existe un entier statique (ex R.drawable.icon). Cet entier est l'ID de ressource que vous pouvez utiliser pour récupérer votre ressource.

String.XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, TestActivity!</string>
    <string name="app_name">Test_Application</string>
    <string name="test">test</string>
</resources>
```

2. Android

c. Structure du Projet Android

Les types des ressources

Classe R

• aapt(android asset packaging tool) génère, lors de la compilation de l'application , la class R qui contient les identifiants de toutes les ressources se trouvant dans les sous répertoires du « res ».

• Pour chaque type de ressource, il existe une sous classe de R (ex R.drawable) et pour chaque ressource de ce type, il existe un entier statique (ex R.drawable.icon). Cet entier est l'ID de ressource que vous pouvez utiliser pour récupérer votre ressource.

R.XML

```
public final class R {
    public static final class attr {
    }
    public static final class color {
        public static final int alert_color=0x7f050000;
        public static final int nice_color=0x7f050001;
        public static final int white=0x7f050002;
    }
    public static final class dimen {
        public static final int dim_hello=0x7f060000;
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int monButton=0x7f080003;
        public static final int monEditText=0x7f080001;
        public static final int monImage=0x7f080002;
        public static final int monText2=0x7f080000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f070001;
        public static final int hello=0x7f070000;
        public static final int test=0x7f070002;
    }
    public static final class xml {
        public static final int countries=0x7f040000;
    }
}
```

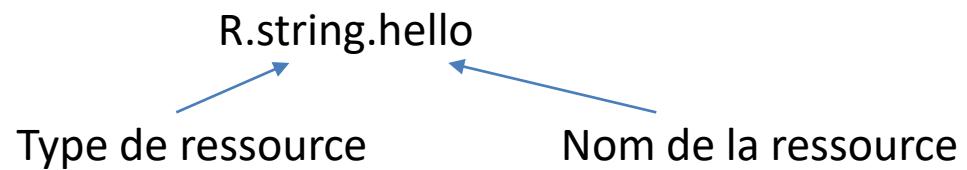
2. Android Studio

c. Structure du Projet Android

Les types des ressources

Deux façons pour accéder aux ressources:

Code: L'utilisation d'un nombre entier statique de la classe R, ex:



XML: Utilisation de ID défini dans la class R ex: @string/hello

2. Android Studio

c. Structure du Projet Android

Les types des ressources

AndroidManifest.xml

- Chaque application android nécessite un fichier de configuration AndroidManifest.xml placé dans la racine du projet;

Ce fichier décrit :

- le point d'entrée de l'application (quel code doit être exécuté au démarrage de l'application) ;
- quels composants constituent ce programme ;
- les permissions nécessaires à l'exécution du programme (accès à Internet, accès à l'appareil photo...).

2. Android Studio

c. Structure du Projet Android

Structure de fichier

Espace de nom android



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="fr.domaine.application">

</manifest>
```

2. Android Studio

c. Structure du Projet Android

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mabik.tp2_1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="TP2_ListView"
        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>
        <activity android:name=".ListActivity"></activity>
    </application>

</manifest>
```

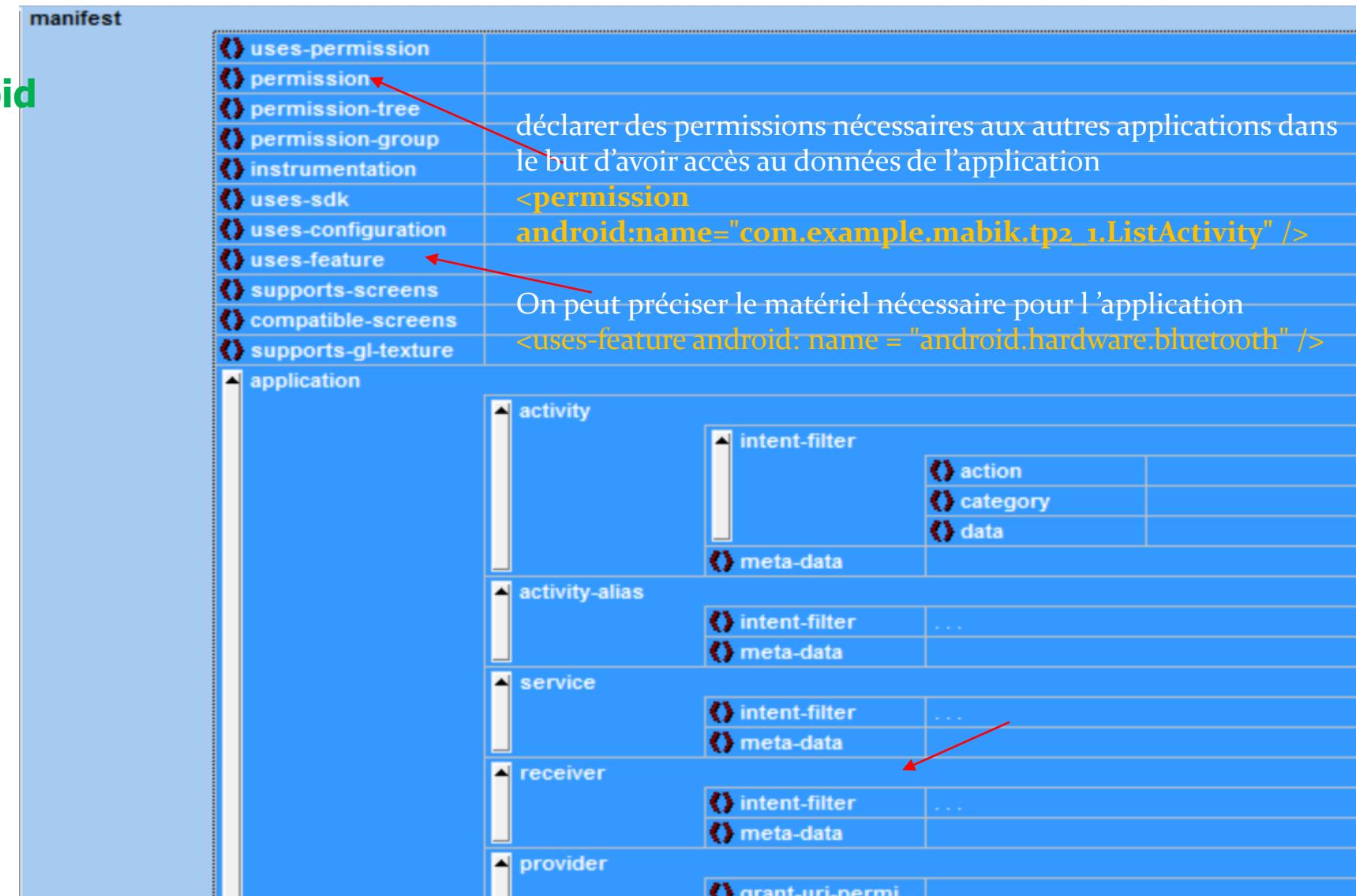
The diagram shows the structure of the `AndroidManifest.xml` file. A callout points to the `allowBackup` attribute with the text: "Synchronisation est activé dans le cloud (backup et restore)". Another callout points to the `supportsRtl` attribute with the text: "support right-to-left (RTL) layouts".

2. Android Studio

c. Structure du Projet Android

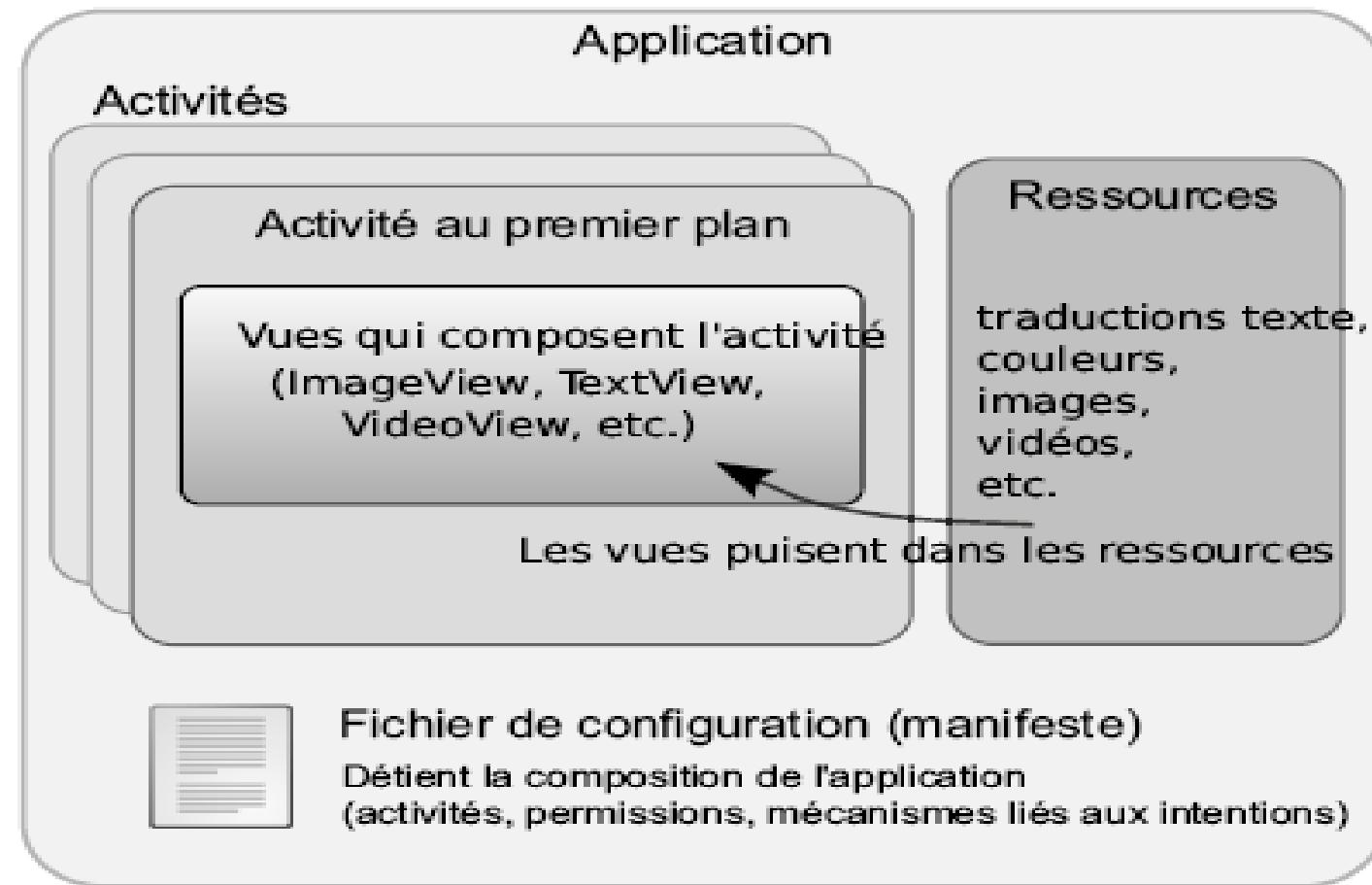
Les permissions nécessaires à l'application (lors de l'installation):
(connexion réseau, localisation...)

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



2. Android Studio

d. Composants d'une application Android



2. Android Studio

d. Composants d'une application Android

Une application Android consiste en un assemblage de composants graphiques liés par un fichier de configuration. Les concepts fondamentaux sont :

- Les vues:** éléments de l'interface graphique (que l'utilisateur voit et sur lesquels il pourra agir) qui héritent de la classe View. Ils accèdent aux textes et aux images à partir des fichiers ressources de l'application.

Ex : boutons, champs de saisie, case à cocher, ...

2. Android Studio

d. Composants d'une application Android

- **Activités:** Une Activité représente la fenêtre qui sera affichée à l'utilisateur.
 - C'est un ensemble de vues composant une interface logique.
 - Elle permet également de gérer des fonctionnalités telles que l'appui sur une vue, l'affichage de messages, etc.
 - Ce concept repose essentiellement sur l'interaction de l'utilisateur avec l'écran.

2. Android Studio

d. Composants d'une application Android

Les principaux composants d'une application Android sont:

- Activités ;
- Services ;
- Fournisseurs de contenu ;
- Gadgets ;
- Objets Intent ;
- Récepteurs d'Intents ;
- Notifications.

2. Android Studio

d. Composants d'une application Android

Composants applicatifs:

Activité, service, fournisseurs de contenu et gadget

Éléments d'interaction

Intents, récepteurs, notifications

Ces éléments permettent l'interaction entre les différents composants du système, entre les applications installées sur l'appareil ou avec l'utilisateur

2. Android Studio

d. Composants d'une application Android

L'activité représente le bloc de base d'une application. Elle correspond à la partie présentation de l'application et fonctionne par le biais de vues qui affichent des interfaces graphiques et répondent aux actions utilisateur.

Une activité est composée de deux volets

- ❑ Logique de l'activité et la gestion du cycle de vie de l'activité qui sont implémentés en Java (ou kotlin) dans une classe héritant de Activity;
- ❑ Interface utilisateur pourra être définie soit dans le code de l'activité soit de façon plus générale dans un fichier XML placé dans les ressources de l'application

Package associé: android.app.Activity

2. Android Studio

d. Composants d'une application Android

- ❑ Une application Android peut avoir plusieurs activités l'une des activités est désignée comme activité de démarrage.
- ❑ Une activité utilise la méthode setContentView pour s'associer avec une interface graphique
- ❑ Les activités sont indépendantes les unes des autres mais elles peuvent collaborer pour échanger des données et des actions
- ❑ Le passage d'une activité à une autre est réalisé via un Intent.

2. Android Studio

d. Composants d'une application Android

Éléments d'interaction : Intents, récepteurs, notifications

Les éléments suivants permettent l'interaction entre les différents composants du système, entre les applications installées sur l'appareil ou avec l'utilisateur.

Éléments d'interaction : L'objet Intent

- L'intent permet de diffuser des messages en demandant la réalisation d'une action.
- Les intents permettent aux applications de fournir ou demander des services ou des données (car l'accès aux autres applications et au système est restreint par le modèle de sécurité Android)
- La transmission se fait à travers tout le système et peut cibler précisément une activité ou un service.

`android.content.Intent`

2. Android Studio

d. Composants d'une application Android

Récepteur d'Intents (Récepteur d'Intents (Broadcast Receiver))

- Permet à une application d'être à l'écoute des autres afin de répondre aux objets Intent qui lui sont destinés et qui sont envoyés par d'autres composants applicatifs.

`android.content.BroadcastReceiver`

Éléments d'interaction: Notification

- Une notification signale une information à l'utilisateur sans interrompre ses actions en cours.

`android.app.Notification`

2. Android Studio

d. Composants d'une application Android

Composant applicatif: fournisseur de contenu

- ❑ Permet de gérer et de partager des informations. Un même fournisseur permet d'accéder à des données au sein d'une application et entre applications.
- ❑ Exemple : le Carnet d'adresses . Les accès aux données d'une application différente de la nôtre se font à l'aide des fournisseurs de contenu (content providers).

`android.content.ContentProvider`

Composant applicatif: gadget

- ❑ est un composant graphique qui s'installe sur le bureau Android.
Ex : Le calendrier qui affiche de l'information ou le lecteur audio qui permet de contrôler la lecture de fichiers

`android.appwidget.*`

2. Android Studio

e. Cycle de vie d'une application

Gestion des processus

- ❑ Pour une application Android, il n'y a qu'une activité « active » à la fois, c-à-d. en « avant-plan »
- ❑ Lorsque la quantité de mémoire libre du système est trop basse, le système d'exploitation est libre de terminer une activité qui est en arrière-plan
- ❑ Une application Android, lorsqu'elle n'est plus en avant-plan, doit être capable de maintenir son état (retrouver les valeurs et affichage précédents lorsqu'il remet l'application en avant-plan).
- ❑ C'est à l'application de gérer son état, ses données, et ses ressources afin d'être prête à être interrompue ou bien terminée à tout moment.
- ❑ Pour répondre à ses événements, des méthodes de réponse, callback, sont définies : `onCreate()`, `onResume()`, `onPause()`, `onDestroy()`

2. Android Studio

e. Cycle de vie d'une application

Description des différents callback

`onCreate()` : est appelé quand l'application démarre ou redémarre. Initialiser les données statiques, établir des liens vers les données et les ressources, positionner l'interface avec `setContentView()`.

`onResume()` : appelé quand une activité passe en avant-plan.

Reprendre le contrôle des ressources exclusives. Continuer les lectures audio et vidéo ou les animations.

`onRestart()`: appelé quand on veut reprendre une activité après son arrêté par `onPause`. Toujours suivi par `onStart ()`

`onPause()` : appelé quand l'activité quitte l'avant plan. Sauvegarder les données non encore sauvegardées, libérer l'accès aux ressources exclusives, stopper la lecture audio, vidéo et les animations.

`OnStop:` appelée lorsque l'activité n'est plus visible pour l'utilisateur. Suivie soit `onRestart ()` si cette activité revient à interagir avec l'utilisateur, ou `OnDestroy ()` si cette activité doit être détruite.

`onDestroy()` : appelé quand l'application est fermée. Nettoyer les données statiques de l'activité, libérer toutes les ressources obtenues.

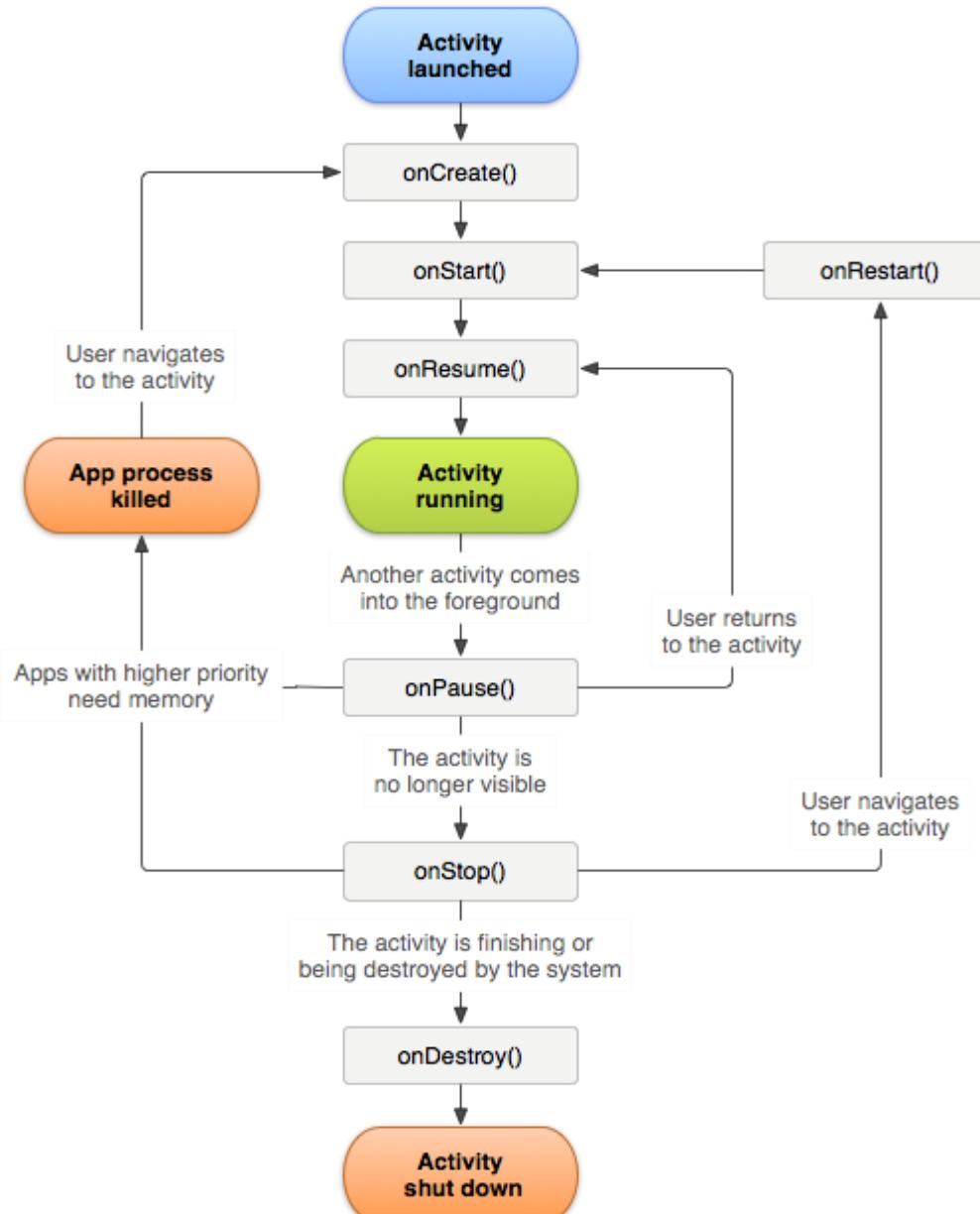
2. Android Studio

e. Cycle de vie d'une application

Description des différents callback

Une activité est composée de deux volets :

- ❑ la logique de l'activité et la gestion du cycle de vie de l'activité qui sont implémentés en Java (ou kotlin) dans une classe héritant de Activity;
- ❑ l'interface utilisateur pourra être définie soit dans le code de l'activité soit de façon plus générale dans un fichier XML placé dans les ressources de l'application.



2 Android Studio

e. Cycle de vie d'une application

Description des différents callback

```
package com.example.mabik.tp2_1;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
public class Test extends AppCompatActivity {

    /**
     * Appelée lorsque l'activité est créée.
     * Permet de restaurer l'état de l'interface
     * utilisateur grâce au paramètre savedInstanceState.
     */
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Placez votre code ici
    }

    /**
     * Appelée lorsque que l'activité a fini son cycle de vie.
     * C'est ici que nous placerons notre code de libération de
     * mémoire, fermeture de fichiers et autres opérations
     * de "nettoyage".
     */
    @Override
    public void onDestroy(){
        // Placez votre code ici
        super.onDestroy();
    }

    /**
     * Appelée lorsque l'activité démarre.
     * Permet d'initialiser les contrôles.
     */
    @Override
    public void onStart(){
        super.onStart();
        // Placez votre code ici
    }

    /**
     * Appelée lorsque l'activité passe en arrière plan.
     * Libérez les écouteurs, arrêtez les threads, votre activité
     * peut disparaître de la mémoire.
     */
    @Override
    public void onStop(){
        // Placez votre code ici
        super.onStop();
    }

    /**
     * Appelée lorsque l'activité sort de son état de veille.
     */
    @Override
    public void onRestart(){
        super.onRestart();
        // Placez votre code ici
    }

    /**
     * Appelée lorsque que l'activité est suspendue.
     * Stoppez les actions qui consomment des ressources.
     * L'activité va passer en arrière-plan.
     */
    @Override
    public void onPause(){
        // Placez votre code ici
        super.onPause();
    }

    /**
     * Appelée après le démarrage ou une pause.
     * Relancez les opérations arrêtées (threads).
     * Mettez à jour votre application et vérifiez vos écouteurs.
     */
    @Override
    public void onResume(){
        super.onResume();
        // Placez votre code ici
    }

    /**
     * Appelée lorsque l'activité termine son cycle visible.
     * Sauvez les données importantes.
     */
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState) {
        // Placez votre code ici
        // sans quoi l'activité aura perdu son état
        // lors de son réveil
        super.onSaveInstanceState(savedInstanceState);
    }
}
```

2. Android Studio

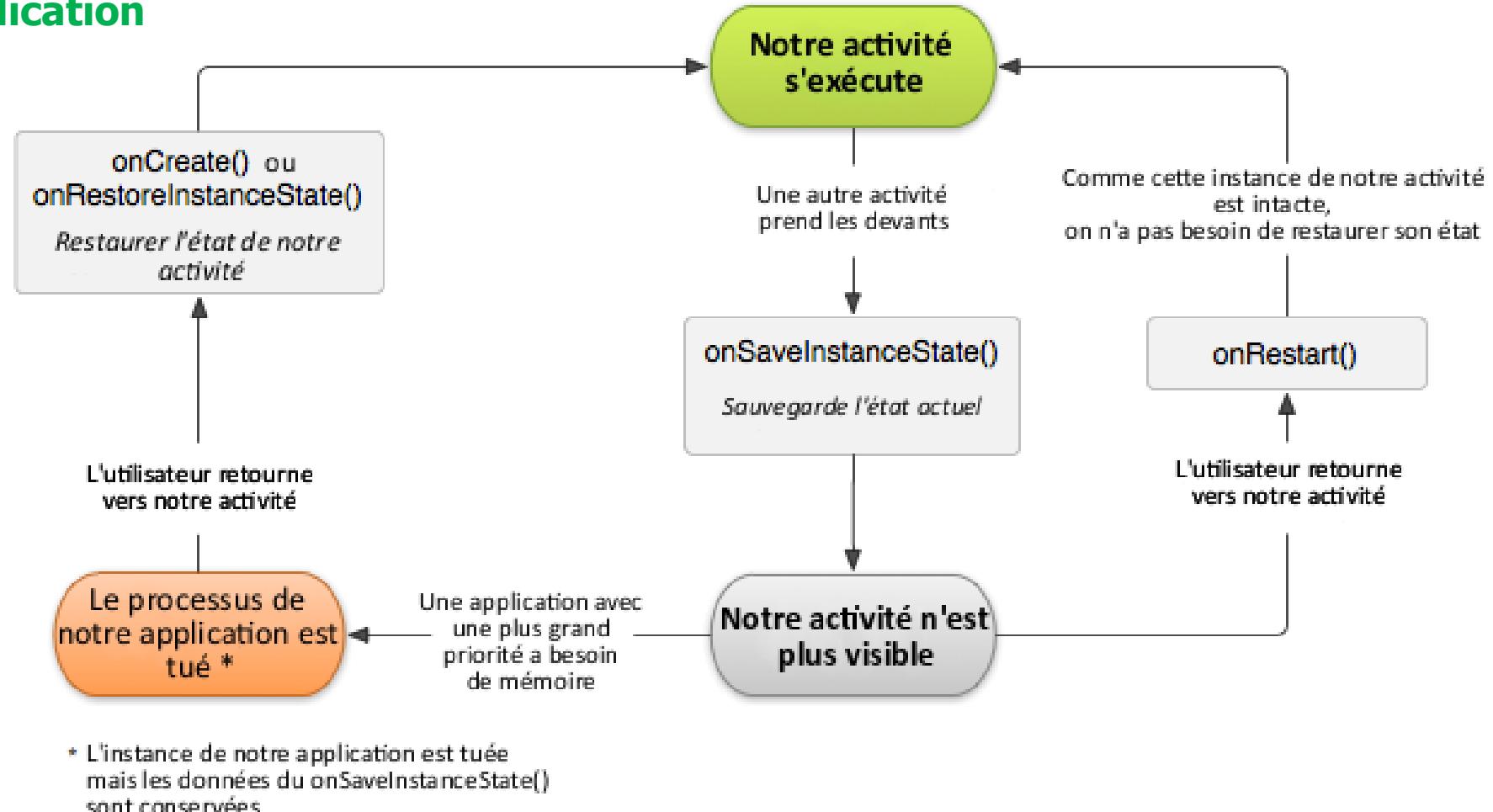
Description des différents callback

```
/**  
 * Appelée après onCreate.  
 * Les données sont rechargées et l'interface utilisateur.  
 * est restaurée dans le bon état.  
 */  
@Override  
public void onRestoreInstanceState(Bundle savedInstanceState) {  
    super.onRestoreInstanceState(savedInstanceState);  
    //Placez votre code ici  
}  
}
```

2. Android

e. Cycle de vie d'une application

Description des différents callback



Source: <https://formations.imt-atlantique.fr/introductionAndroid/tp-gestionEtatActivite.html>

3 Crédit d'IHM

3. Création d'IHM

Pour créer une IHM nous avons besoin:

- Une définition de l'interface utilisateur (layout) dans un fichier XML
- Une définition de la logique utilisateur càd le comportement de l'interface dans une classe d'activité



Séparation entre la présentation et la logique fonctionnelle
de votre application

3. Création d'IHM

Pour créer une IHM nous avons besoin:

- On construit l'IHM avec un fichier XML rangé dans res\layout
- Ce fichier est repéré dans le code Java par `R.layout.nomlayout`
- On positionne l'IHM de l'activité à l'aide de `setContentView(R.layout.nomlayout)` dans l'activité
- On récupère un composant graphique particulier de l'IHM par `findViewById(R.id.idDuComposant)`

3. Création d'IHM

Layout

- Les layouts sont de type ViewGroup (conteneurs de composants graphiques)
 - gèrent le placement des composants
- Attributs XML obligatoires
 - largeur et hauteur (layout_width et layout_height)
 - `fill_parent`: remplit l'espace du conteneur parent,
 - `wrap_content`: se dimensionne en fonction du contenu du composant
 - les valeurs absolues: dip, px, mm, in, ...

3. Création d'IHM

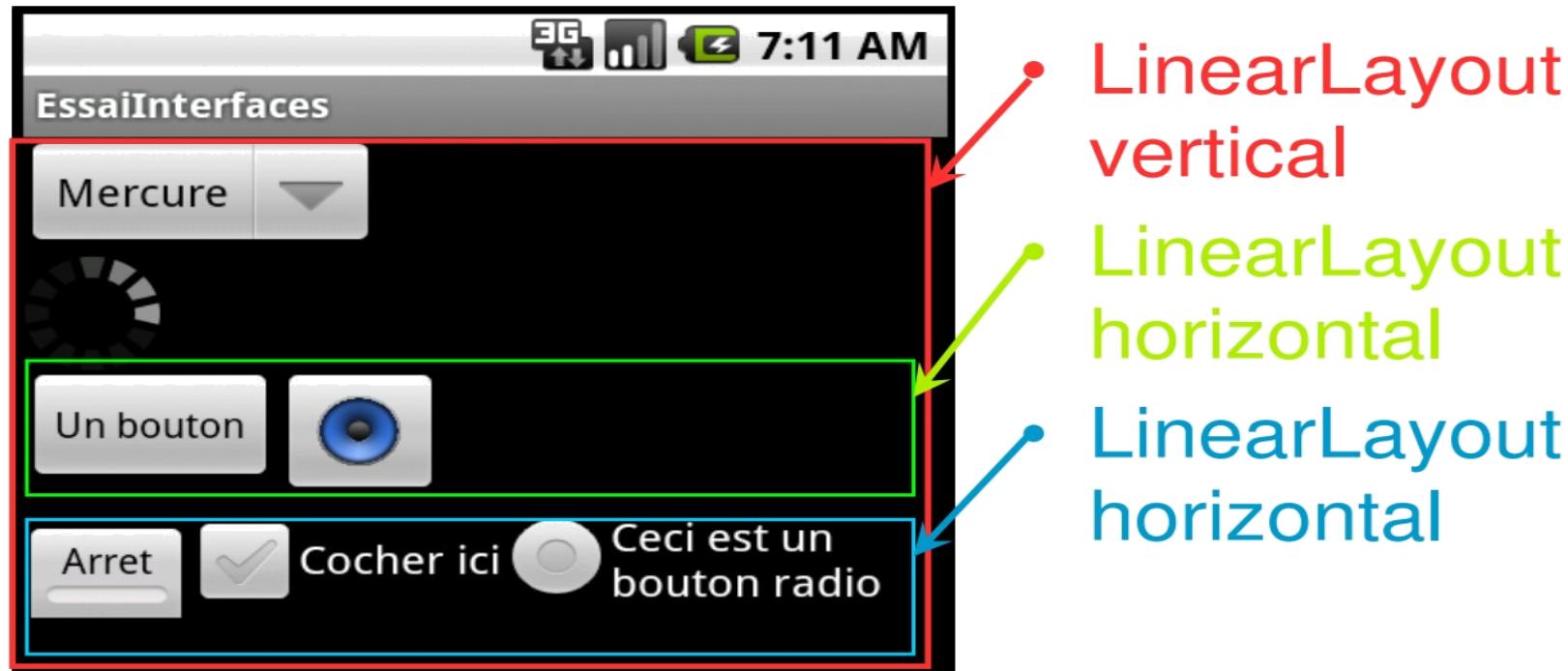
Layout

- Les layouts sont de type ViewGroup (conteneurs de composants graphiques)
 - gèrent le placement des composants
- Attributs XML obligatoires
 - largeur et hauteur (layout_width et layout_height)
 - `fill_parent`: remplit l'espace du conteneur parent,
 - `wrap_content`: se dimensionne en fonction du contenu du composant
 - les valeurs absolues: dip, px, mm, in, ...

3. Création d'IHM

LinearLayout

Permet d'aligner les éléments de gauche à droite ou de haut en bas



`android:orientation= "horizontal" ou vertical`

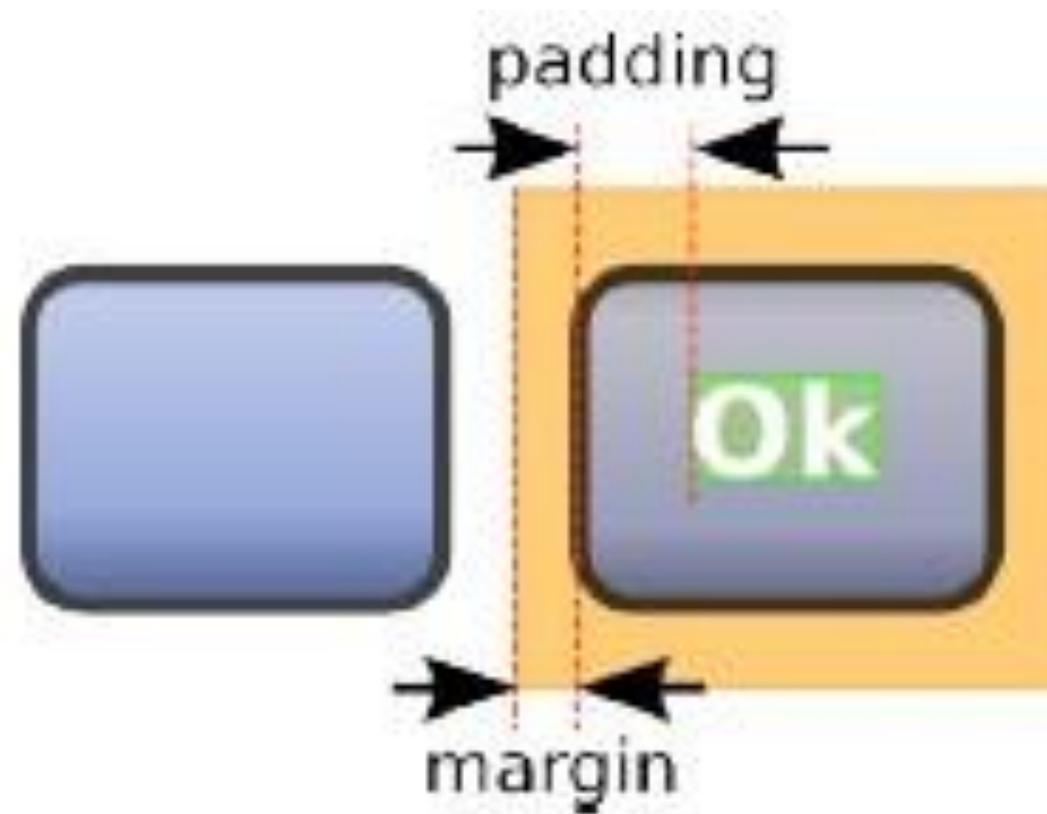
3. Création d'IHM

LinearLayout : gestion des vues

- **Gravité des éléments** : les éléments sont alignés de gauche à droite et de haut en bas : left, center_horizontal, top, bottom, right
 - android:layout_gravity="center_horizontal"
- **Poids des éléments** : partager l'espace libre entre les vues (0 par défaut). On peut utiliser les % (somme \leq 100)
 - Ex : android:layout_weight=1" pour la 1ere vue et android:layout_weight=2 » pour la 2eme vue —> la 2eme vue prendra deux fois plus d'espace libre que la première
- **Espacement**
 - **Interne** : précise l'espace situé entre le contour de la vue et son contenu réel (padding)
 - Ex: android:paddingTop="10dp"
 - **Externe** : précise l'espace externe par rapport au bort de la vue (margin)->
android:layout_marginTop="10dp"

3. Création d'IHM

LinearLayout : gestion des vues



3. Création d'IHM

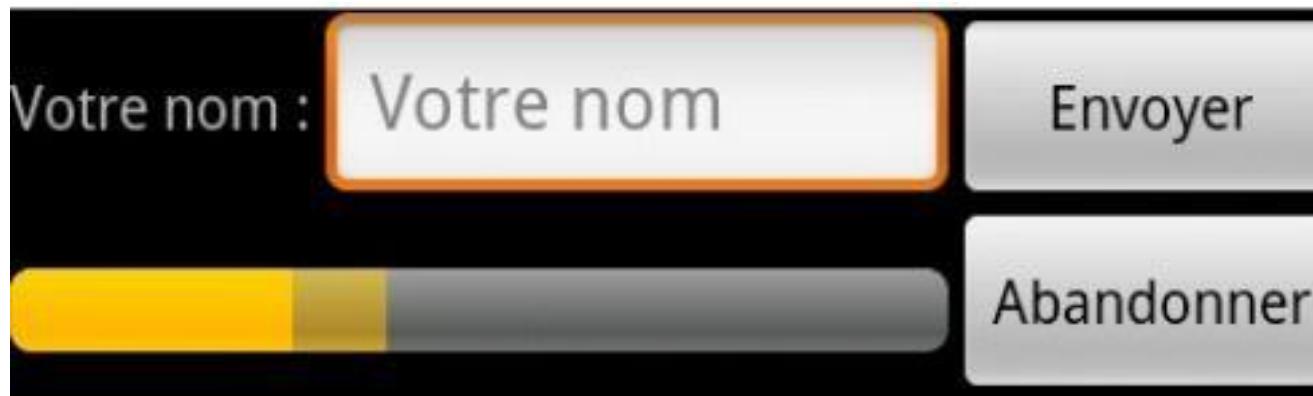
FrameLayout

- Ne contient qu'un seul élément (si on en met plusieurs ils se superposent)
 - Ex: cadre dans lequel on veut charger des images

3. Création d'IHM

TableLayout

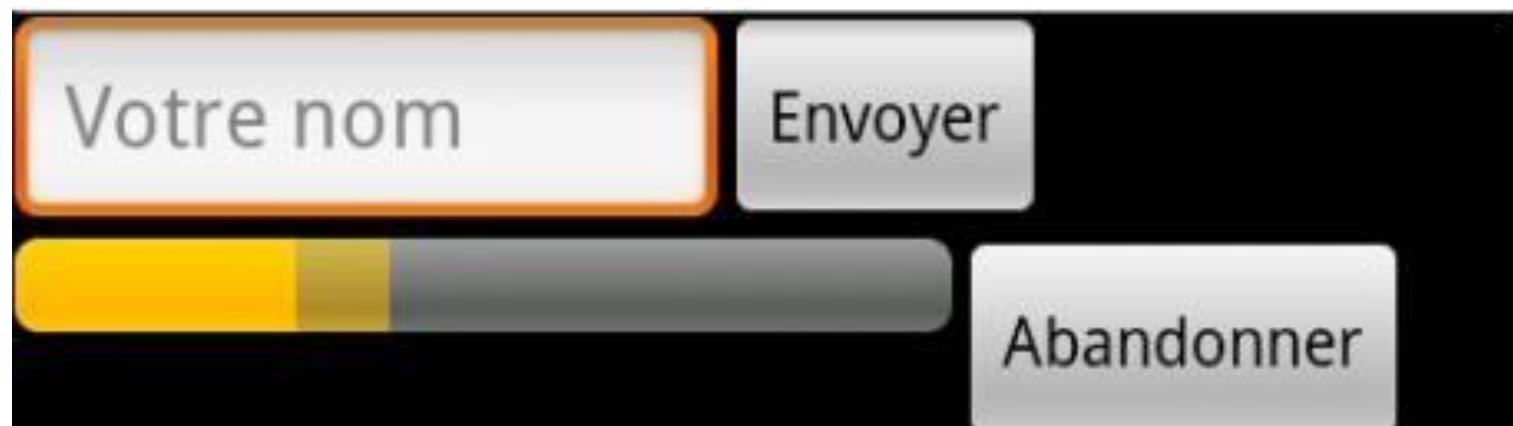
permet de positionner vos vues en lignes et colonnes à l'instar d'un tableau



3. Création d'IHM

RelativeLayout

Permet de placer des éléments les uns relativement aux autres



3. Création d'IHM

RelativeLayout

- 2 cas de figure:
 - Situer une vue par rapport à un RelativeLayout
 - Placer les éléments les uns par rapport aux autres

3. Création d'IHM

RelativeLayout:

Exemple situer une vue par rapport à un RelativeLayout (alignement des bords)

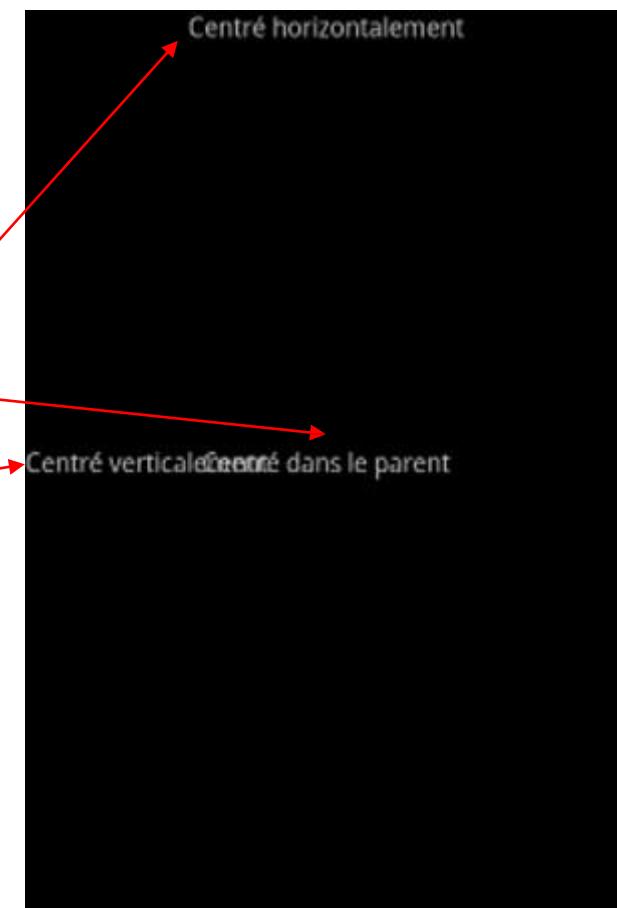
situer une vue par rapport à un RelativeLayout	
android:layout_alignParentBottom="true"	
android:layout_alignParentTop="true"	coller le plafond d'une vue au plafond du RelativeLayout
android:layout_alignParentLeft="true"	coller le bord gauche d'une vue avec le bord gauche du RelativeLayout
android:layout_alignParentRight="true"	coller le bord droit d'une vue avec le bord droit du RelativeLayout

3. Création d'IHM

RelativeLayout

RelativeLayout (suite): ex situer une vue par rapport à un RelativeLayout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent" >  
  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Centré dans le parent"  
        android:layout_centerInParent="true" />  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Centré verticalement"  
        android:layout_centerVertical="true" />  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="Centré horizontalement"  
        android:layout_centerHorizontal="true" />  
  
</RelativeLayout>
```



3. Création d'IHM

RelativeLayout



placer les éléments les uns par rapport aux autres

positionner deux bords opposés de deux vues différentes ensemble		coller deux bords similaires ensemble	
android:layout_below	aligner le plafond d'une vue sous le plancher d'une autre	android:layout_align Bottom	aligner le plancher de la vue avec le plancher d'une autre
android:layout_above	aligner le plancher d'une vue sur le plafond d'une autre	android:layout_align Top	aligner le plafond de la vue avec le plafond d'une autre
android:layout_toRight Of	aligner le bord gauche d'une vue au bord droit d'une autre	android:layout_align Left	aligner le bord gauche d'une vue avec le bord gauche d'une autre
android:layout_toLeftOf	aligner le bord droit d'une vue au bord gauche d'une autre	android:layout_align Right	aligner le bord droit de la vue avec le bord droit d'une autre

3. Création d'IHM

RelativeLayout:

placer les éléments les uns par rapport aux autres

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     android:layout_width="fill_parent"
3     android:layout_height="fill_parent" >
4     <TextView
5         android:id="@+id/premier"
6         android:layout_width="wrap_content"
7         android:layout_height="wrap_content"
8         android:text="[I] En haut à gauche par défaut" />
9     <TextView
10        android:id="@+id/deuxieme"
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="[II] En dessous de (I)"
14        android:layout_below="@+id/premier" />
15     <TextView
16        android:id="@+id/troisieme"
17        android:layout_width="wrap_content"
18        android:layout_height="wrap_content"
19        android:text="[III] En dessous et à droite de (I)"
20        android:layout_below="@+id/premier"
21        android:layout_toRightOf="@+id/premier" />
22     <TextView
23        android:id="@+id/quatrieme"
24        android:layout_width="wrap_content"
25        android:layout_height="wrap_content"
26        android:text="[IV] Au dessus de (V), bord gauche aligné avec le bord gauche de (II)"
27        android:layout_above="@+id/cinquieme"
28        android:layout_alignLeft="@+id/deuxieme" />
29     <TextView
30        android:id="@+id/cinquieme"
31        android:layout_width="wrap_content"
32        android:layout_height="wrap_content"
33        android:text="[V] En bas à gauche"
34        android:layout_alignParentBottom="true"
35        android:layout_alignParentRight="true" />
36   </RelativeLayout>
```



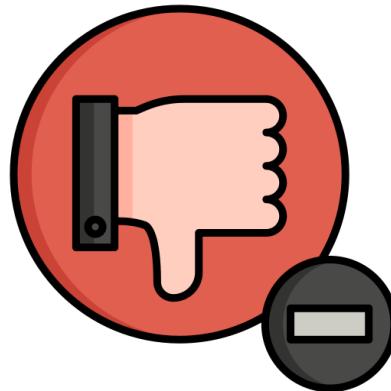
3. Création d'IHM

Imbrication de layouts

- Lors du développement de vos interfaces vous serez mener à imbriquer des layouts : mettre des linearlayout dans un relativelayout...
 - EX: dans un relativelayout centrer un ensemble de vues horizontalement
 - android:layout_centerInParent="true" pour chaque vue
 - -> des vues superposés
 - Besoin d'y mettre un linearlayout : mettre vos views dans un linearlayout et le centrer par rapport au relativelayout

3. Création d'IHM

Imbrication de layouts



Ralentissement lors du
chargement des vues
imbriquées

3. Création d'IHM

ConstraintLayout

- Apparu en Mai 2016
- Il hérite des contraintes de placement du **RelativeLayout** mais avec des règles de positionnement beaucoup plus puissantes.
- Il apporte des nouveautés avec la possibilité de lier des *Widgets*, *Text*, ... entre eux, d'appliquer un « poids » à la manière du *LinearLayout* ou encore d'attribuer un identifiant unique à plusieurs éléments pour effectuer des actions groupées.

<https://developer.android.com/training/constraint-layout/index.html>

<http://tutos-android-france.com/constraintlayout-partie-1/>

3. Création d'IHM

ConstraintLayout

layout_constraint(X)_to(Y)of

- X le côté du vue où l'on place la contrainte
- Y le côté du vue de destination

RelativeLayout
layout_alignLeft
layout_alignStart
layout_alignRight
layout_alignEnd
layout_alignTop
layout_alignBottom
layout_alignBaseline
layout_alignParentLeft
layout_alignParentStart
layout_alignParentRight
layout_alignParentEnd
layout_alignParentTop
layout_alignParentBottom
layout_toLeftOf
layout_toStartOf
layout_toRightOf
layout_toEndOf
layout_above
layout_below

vers -> ConstraintLayout
-> layout_constraintLeft_toLeftOf
-> layout_constraintStart_toStartOf
-> layout_constraintRight_toRightOf
-> layout_constraintEnd_toEndOf
-> layout_constraintTop_toTopOf
-> layout_constraintBottom_toBottomOf
-> layout_constraintBaseline_toBaselineOf
-> layout_constraintLeft_toLeftOf
-> layout_constraintStart_toStartOf
-> layout_constraintRight_toRightOf
-> layout_constraintEnd_toEndOf
-> layout_constraintTop_toTopOf
-> layout_constraintBottom_toBottomOf
-> layout_constraintRight_toLeftOf
-> layout_constraintEnd_toStartOf
-> layout_constraintLeft_toRightOf
-> layout_constraintStart_toEndOf
-> layout_constraintBottom_toTopOf
-> layout_constraintTop_toBottomOf

3. Création d'IHM

ConstraintLayout

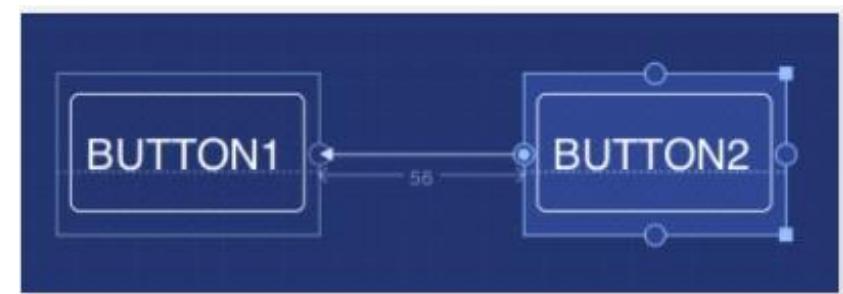
- **Les contraintes de positionnement :**

`layout_constraint(X)_to(Y) of`

X le côté du vue où l'on place la contrainte

Y le côté du vue de destination

Ex: `layout_constraintLeft_toRightOf`



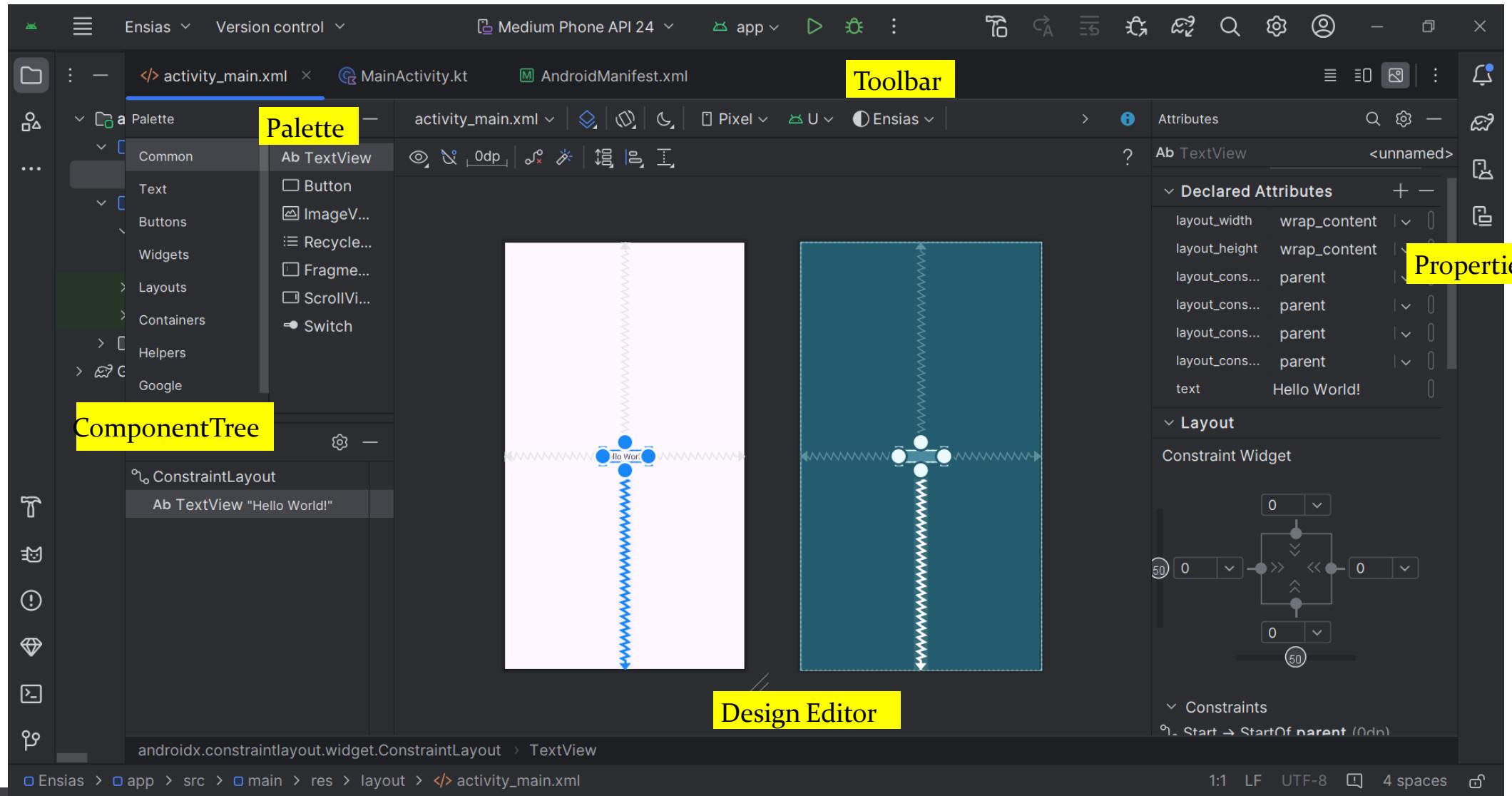
3. Création d'IHM

ConstraintLayout

- Pour définir la position d'une vue ConstraintLayout, vous devez ajouter au moins une contrainte horizontale et une contrainte verticale pour la vue.

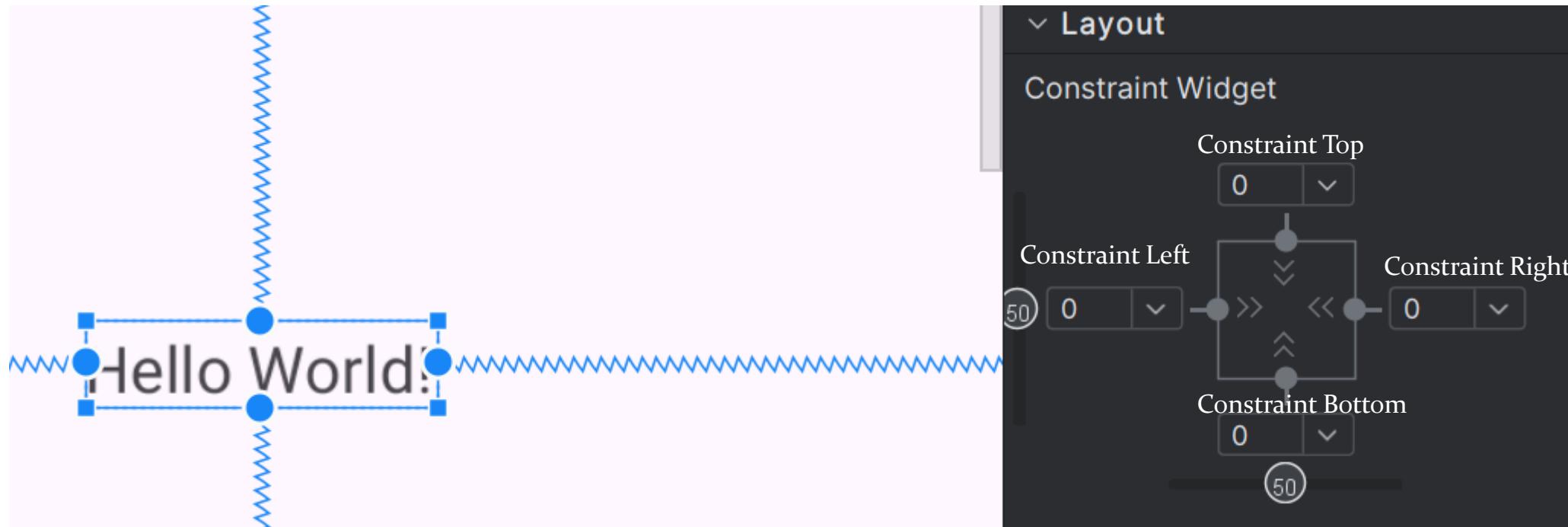
3. Création d'IHM

L'EDITEUR ANDROID STUDIO DESIGN (Layout)



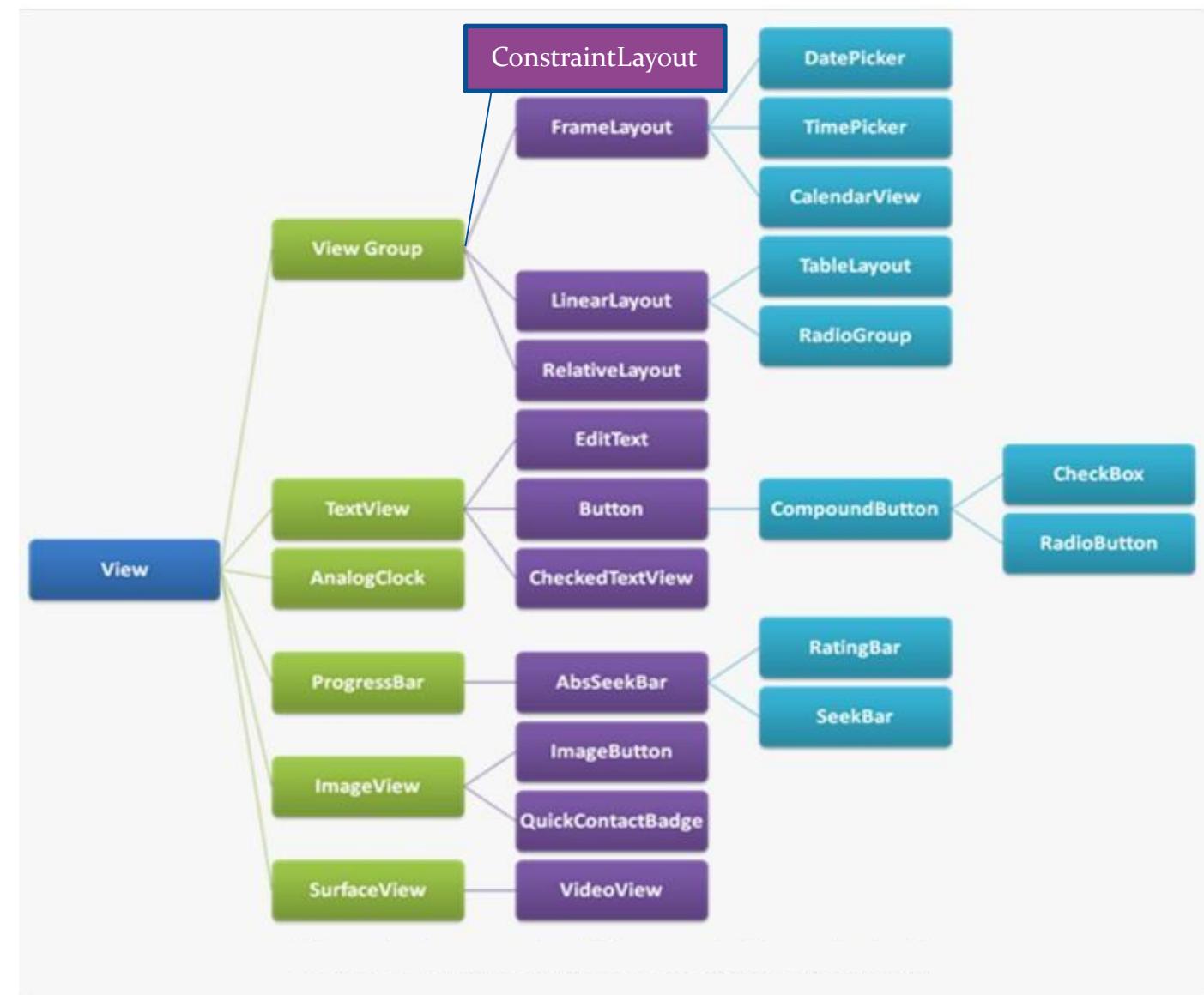
3. Création d'IHM

L'EDITEUR ANDROID STUDIO DESIGN (Layout)



3. Création d'IHM

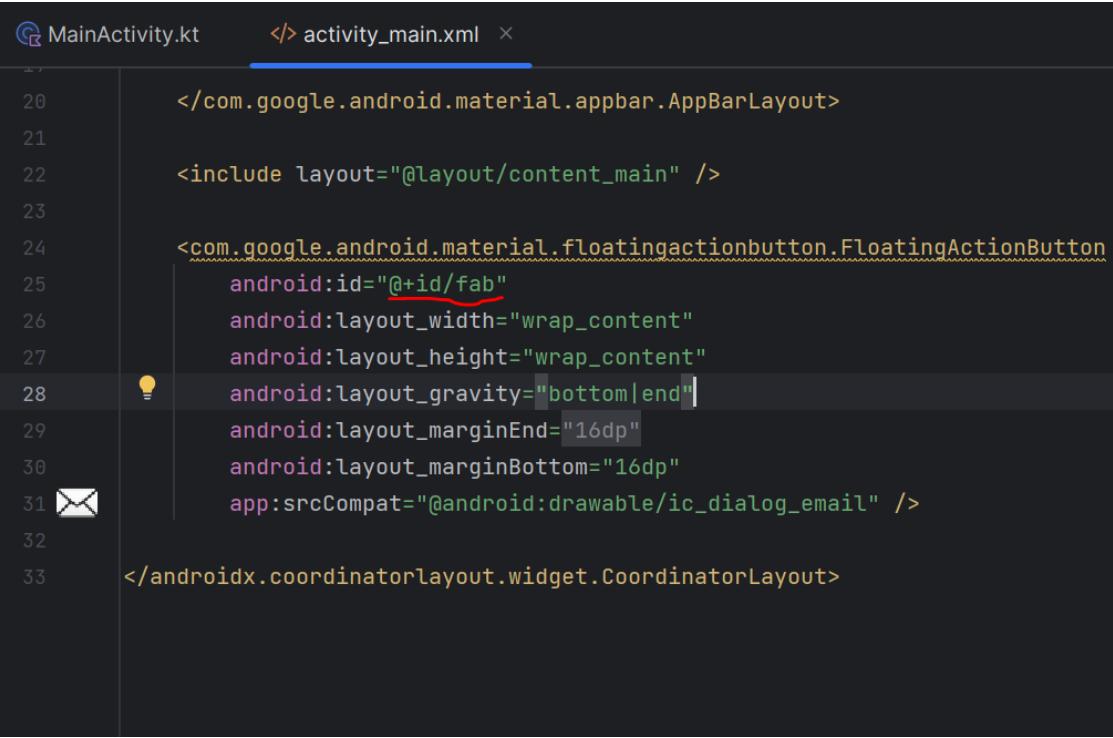
View: Eléments de base de l'IHM



3. Création d'IHM

@+id & @id

- Dans le fichier xml on peut utiliser @+id et @id
 - @+id indique qu'il faut créer une nouvelle entrée dans R.java
 - @id repère simplement l'identificateur id et il n'y a pas de création dans R.java

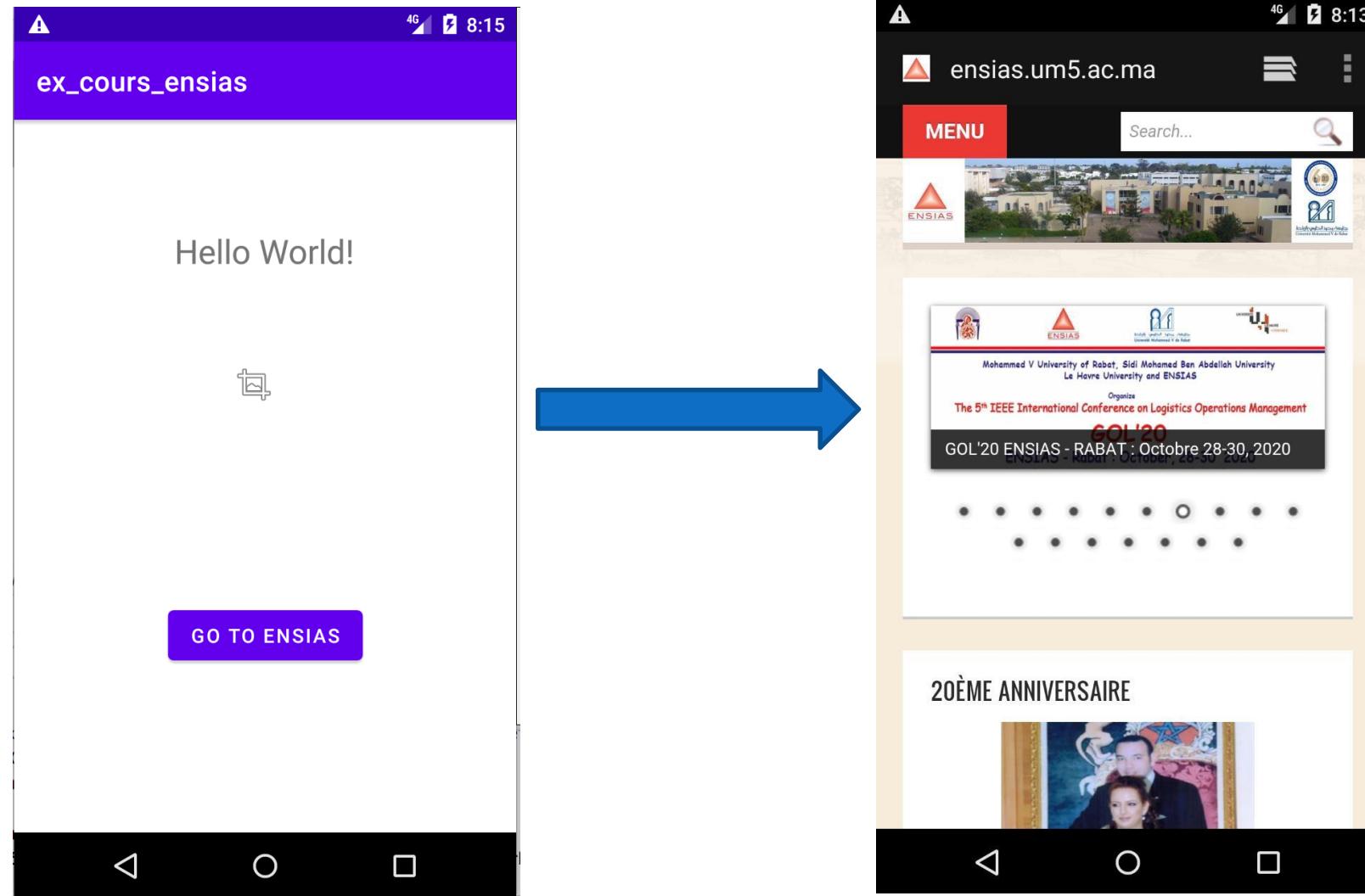


```
MainActivity.kt           activity_main.xml ×

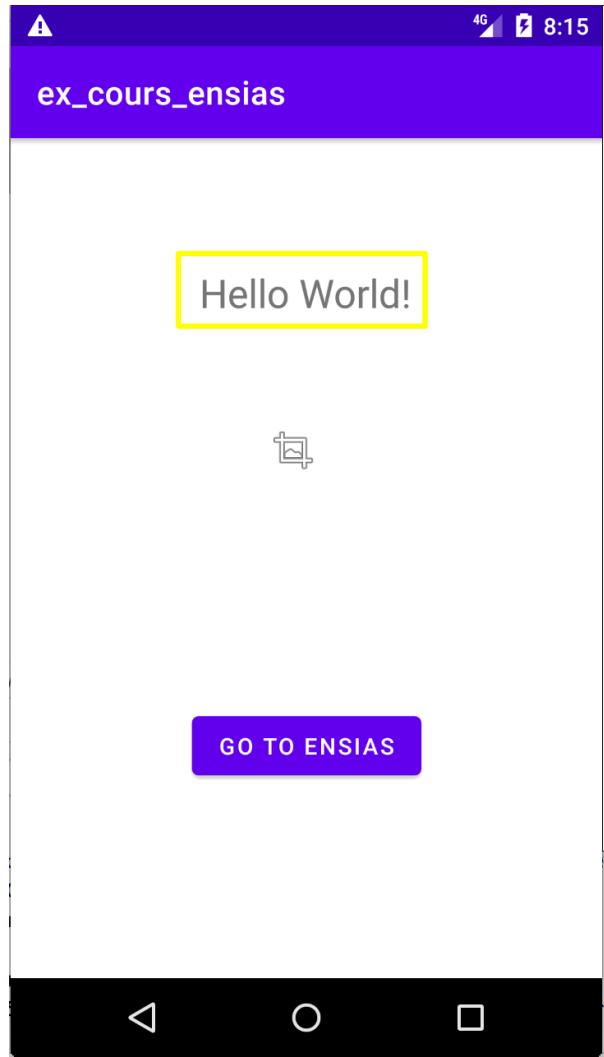
20   </com.google.android.material.appbar.AppBarLayout>
21
22   <include layout="@layout/content_main" />
23
24   <com.google.android.material.floatingactionbutton.FloatingActionButton
25       android:id="@+id/fab"
26       android:layout_width="wrap_content"
27       android:layout_height="wrap_content"
28       android:layout_gravity="bottom|end"
29       android:layout_marginEnd="16dp"
30       android:layout_marginBottom="16dp"
31       app:srcCompat="@android:drawable/ic_dialog_email" />
32
33   </androidx.coordinatorlayout.widget.CoordinatorLayout>
```

3. Création d'IHM

Exemple



3. Création d'IHM



```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.162" />
    <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="156dp"
        android:layout_marginLeft="156dp"
        android:layout_marginTop="96dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/textView"
        app:srcCompat="@android:drawable/ic_menu_crop"
        tools:srcCompat="@tools:sample/avatars" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="140dp"
        android:text="Go to ensias"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />
```

3. Création d'IHM

Styles et Thèmes

- Un style est un ensemble de propriétés précisant l'aspect de composants graphiques (View)
 - Police de caractères et tailles pour les textes ,
 - Couleurs, images. . .
 - Géométrie par défaut des vues : taille, espacement, remplissage. . .
- Il est défini dans un fichier XML séparé de l'IHM(res/values/styles.xml). Cette notion est similaire au CSS
- Les styles personnalisés doivent étendre les styles existants du framework ou de bibliothèque du support d'android -> assurer la compatibilité avec les styles de l'IHM (disponible pour chaque version android)
 - <https://android.googlesource.com/platform/frameworks/base/+/refs/heads/master/core/res/res/values/styles.xml>

3. Création d'IHM

Styles et Thèmes

- Un thème est un style appliqué à une activité ou application.
 - Une activité ayant un thème associé transmet les indications du thème aux View contenus dans l'activité
- Un style appliqué à un Viewgroup ne se propage pas pour les composants contenus dans le ViewGroup
- Un thème appliqué à une activité est propagé à tous les composants de l'activité
- Les indications de thème sont écrites dans le `AndroidManifest.xml` à l'aide l'attribut `android:theme` de la balise `application`

3. Création d'IHM

Styles et Thèmes

Style

inherits text appearance styles from the support library

The screenshot shows an Android XML resource file. At the top, there is a style definition:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="GreenText" parent="TextAppearance.AppCompat">
        <item name="android:textColor">#00FF00</item>
    </style>
</resources>
```

A red arrow points from the "parent" attribute of the style definition down to the "style" attribute of a `TextView` element below:

```
<TextView
    style="@style/GreenText"
    ... />
```

Theme -> toute l'application

```
<manifest ... >
    <application android:theme="@style/Theme.AppCompat" ... >
        </application>
    </manifest>
```

Theme -> une activité

```
<manifest ... >
    <application ... >
        <activity android:theme="@style/Theme.AppCompat.Light" ... >
            </activity>
        </application>
    </manifest>
```

<https://developer.android.com/guide/topics/ui/look-and-feel/themes>

3. Création d'IHM

Création d'IHM à partir du code

- Instanciation du layout
 - `LinearLayout layout = new LinearLayout(this)`
- Instantiation des vues
 - `TextView TV = new TextView(this)`
- Ajout de la vue au layout
 - `Layout.addView(TV)`
- Appliquer le layout sur l'activité
 - `setContentView(layout)`
- ...

3. Création d'IHM

Recommandations

Pour assurer l'usage de votre application par un large public vous avez besoin d'assurer le bon fonctionnement de votre application par les différents Android devices.

D'où le besoin d'assurer le support de:

- Différentes langues
- Différents écrans
- Différentes versions

3. Création d'IHM

Recommandations

Quantificateurs

Format d'écriture

res/ressource-quantificateur1-quantificateur2-quantificateurN

Exemples

res/drawable-hdpi
res/drawable-xhdpi
res/layout-land
res/layout-sw600dp-land

3. Création d'IHM

Recommandations: Support de différentes langues

- Internationalisation

Le système de ressources permet de gérer très facilement l'internationalisation d'une application. Il suffit de créer des répertoires ex **values-XX** où **XX** est le code de la langue que l'on souhaite planter.

Ex:

```
Projet/  
    res/  
        values/ strings.xml  
        values-en/ strings.xml  values-  
        fr/  
            strings.xml
```

Android chargera le fichier de ressources approprié en fonction de la langue du système.

3. Création d'IHM

Recommandations: Support de Différents écrans

- Le système assure une mise à l'échelle et un redimensionnement pour que l'application fonctionne sur différents écrans
- **Mais** pour maximiser la qualité de l'expérience utilisateur pour tous les devices
 - L'utilisateur ressent que l'application est faite pour son device

3. Création d'IHM

Recommandations: Support de Différents écrans

- Le système assure une mise à l'échelle et un redimensionnement pour que l'application fonctionne sur différents écrans
- **Mais** pour maximiser la qualité de l'expérience utilisateur pour tous les devices
 - L'utilisateur ressent que l'application est faite pour son device

Optimiser votre application pour différentes tailles et densités d'écran

3. Création d'IHM

Recommandations: Support de Différents écrans

- Le système assure une mise à l'échelle et un redimensionnement pour que l'application fonctionne sur différents écrans
- **Mais** pour maximiser la qualité de l'expérience utilisateur pour tous les devices
 - L'utilisateur ressent que l'application est faite pour son device

Optimiser votre application pour différentes tailles et densités d'écran

3. Création d'IHM

Recommandations: Support de Différents écrans

- **Taille de l'écran** Taille physique réelle, mesurée en diagonale de l'écran
 - Android propose plusieurs breakpoints pour gérer l'affichage quelque soit la taille d'écran (petit, grand, haut resolution, tablette...)
- **Densité de l'écran** La quantité de pixels dans une ligne de 2,54 cm (=1 pouce). Généralement appelé dpi (dots per inch: points par pouce)
- **Orientation:** L'orientation de l'écran du point de vue de l'utilisateur (paysage ou portrait)
- **Résolution** Le nombre total de pixels physiques sur un écran (n'est pas pris en compte pour le support de différents écran)

3. Création d'IHM

Recommandations: Support de Différents écrans

- px : correspond à 1 pixel de l'écran
- Millimètre(mm): Basé sur la taille physique de l'écran
- Pouce (in) : unité de longueur qui correspond à 2,54 cm basé sur la taille physique de l'écran.
- Point(pt): 1/72 d'un pouce

Mais pour exprimer les dimensions des layouts ou la position de manière indépendante de la taille physique on a **dp Density-independent pixel**(Densité de pixels indépendante)

3. Création d'IHM

Recommandations: Support de Différents écrans

- Density-independent pixel (dp) :
 - dp = à un pixel physique sur un écran ayant une densité de 160 dpi, qui est la densité de base supposée par le système pour un écran de densité "moyenne".
 - Si la densité réelle du device est différente de 160, le système gère de manière transparente toute mise à l'échelle des unités dp.
 - La conversion des dp en pixel de l'écran est : $px = dp * (dpi / 160)$

Ex : un écran de 240 dpi → 1 dp équivaut à 1,5 pixels

Vous devez toujours utiliser les unités dp lors de la définition de vos layouts

<https://developer.android.com/training/multiscreen/screendensities>

3. Création d'IHM

Recommandations: Support de Différents écrans

- Scale independent Pixel (sp) : (Echelle indépendant)
utilisée pour les tailles de polices. Cette taille est basée sur les « **dp** », mais intègre le choix de l'utilisateur quant à la taille du texte.
 $1 \text{ dp} = 1 \text{ sp}$ mais si l'utilisateur choisit une taille de texte différente, on aura:
 $1 \text{ dp} = f^* 1 \text{ sp}$ (f étant le facteur d'agrandissement du texte).

Cette unité est particulièrement importante, en particulier pour
utilisateurs nécessitant des aides d'accessibilité

3. Création d'IHM

Recommandations: Support de Différents écrans

- Scale independent Pixel (sp) : (Echelle indépendant)
utilisée pour les tailles de polices. Cette taille est basée sur les « **dp** », mais intègre le choix de l'utilisateur quant à la taille du texte.
 $1 \text{ dp} = 1 \text{ sp}$ mais si l'utilisateur choisit une taille de texte différente, on aura:
 $1 \text{ dp} = f^* 1 \text{ sp}$ (f étant le facteur d'agrandissement du texte).

Cette unité est particulièrement importante, en particulier pour
utilisateurs nécessitant des aides d'accessibilité

3. Création d'IHM

Recommandations: Support de Différents écrans

Android a divisé les écrans des appareils Android en groupes de taille et de densité :

- Quatre types de taille d'écran:
 - *small* (~ 426dp x 320dp)
 - *normal* (~ 470dp x 320dp)
 - *large* (~ 640dp x 480dp)
 - *xlarge* (~ 960dp x 720dp)
- Pour supporter les tablettes, on utilise plutôt les dossiers suivants (depuis Android 3.2):
 - layout: pour les Smartphones
 - layout-sw600dp: pour les tablettes 7 pouces
 - layout-sw700dp: pour les tablettes 10 pouces
 - -....

3. Création d'IHM

Recommandations: Support de Différents écrans

- Six types de densité :
 - ldpi : low density (~120dpi)
 - mdpi : medium density (~160dpi)
 - hdpi : high density (~240dpi)
 - xhdpi : extra-high (~320dpi)
 - xxhdpi : extra-extra-high (~480dpi)
 - xxxhdpi ; extra-extra-extra-high: ~640dpi

3. Création d'IHM

Recommandations: Support de Différents écrans

Exemple d'application qui supporte différents écran

taille de l'écran:

```
res/layout/my_layout.xml           // layout for normal screen size ("default")
res/layout-large/my_layout.xml     // layout for large screen size
res/layout-xlarge/my_layout.xml   // layout for extra-large screen size
res/layout-xlarge-land/my_layout.xml // layout for extra-large in landscape orientation
```

```
res/drawable-mdpi/graphic.png    // bitmap for medium-density
res/drawable-hdpi/graphic.png    // bitmap for high-density
res/drawable-xhdpi/graphic.png   // bitmap for extra-high-density
res/drawable-xxhdpi/graphic.png  // bitmap for extra-extra-high-density
```

taille + orientation de l'écran:

```
res/layout-land/my_layout.xml      //layout for normal screen whith landscape orientation
res/layout-large-land/my_layout.xml //layout for largescreen with landscape orientation
```

....

3. Création d'IHM

Recommandations: Support de Différents écrans

Densité des images:

res/drawable

res/drawable-mdpi

res/drawable-hdpi

...

Outil de génération d'images avec plusieurs densité : <https://github.com/redwarp/9-Patch-Resizer>

La densité peut être aussi utilisé pour d'autres types de ressources :
values (ex: string.xml , dimen.xml), mipmap (pour les icon lunched)

3. Création d'IHM

Graphique Vectoriel multi-densités

- Android studio offre l'outil Vector Asset Studio qui permet d'ajouter des icônes et d'importer des fichier SVG (Scalable Vector Graphic) sous forme de ressources vectorielles.
- Ce type d'image s'adapte à tout type d'écran (pas besoin de créer plusieurs images)
 - Ce qui réduit la taille de l'APK

3. Création d'IHM

Graphique Vectoriel multi-densités

- Pour Android 4.4 et les versions antérieures, Android ne prennent pas en charge l'image vectorielle. Pour les raisons de compatibilité, Vector Asset Studio offre deux propositions:
 - générer lors de la compilation les images raster du vecteur (les images vecteurs et rasters sont embarqués ensemble dans l'APK).
 - L'usage de `VectorDrawableCompat` qui est une classe de la bibliothèque de support et qui permet de prendre en charge de `VectorDrawable`
 - Il faut modifier le `build.gradle` avant de lancer le Vector Asset Studio

3. Création d'IHM

Graphique Vectoriel multi-densités

Exemple d'application qui supporte différents écran

- On veut avoir un titre selon la densité du layout (mdpi ...)

On définit pour chaque ressource (ex res/value-mdpi)
un fichier dimen.xml

ex: dimen.xml

```
<resources>
    <dimen name="size_title">18sp</dimen>
    ...
</resources>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Exemple"
    android:textSize="@dimen/size_title"
    ...
/>
```

3. Création d'IHM

Support de Différentes versions

- build.gradle décrit les versions d'Android prises en charge
 - minSdkVersion et targetSdkVersion

```
    android {  
        compileSdkVersion 27  
        defaultConfig {  
            applicationId "com.example.mac.ttt"  
            minSdkVersion 15  
            targetSdkVersion 27  
            versionCode 1  
            versionName "1.0"  
            testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
        }  
    }
```

3. Création d'IHM

Evénements

- Les événements permettent de gérer les actions utilisateurs sur les vues:
 - Pour gérer les évènements sur les vues, il suffit d'ajouter un écouteur
 - Si un listener est défini pour un élément graphique et un évènement précis, la plate-forme android appellera la méthode associée dès que l'événement sera produit sur le composant.
 - On a trois manières de déclarer les listeners

3. Création d'IHM

Evénements

- Surcharge du Listener au niveau du View

```
public class Main extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        // Nous cherchons le bouton dans notre interface
        ((Button)findViewById(R.id.monBouton))
        // Nous paramétrons un écouteur sur l'événement 'click' de ce bouton
        .setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                // Nous affichons un message à l'utilisateur
                Toast.makeText(Main.this, "Bouton cliqué !", Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

3. Création d'IHM

Evénements

- Dans le fichier XML (valable juste pour le click), ajouter au vue concernée l'attribut

android:onClick="myClick"

dans le code définir la fonction

```
public void myClick(View v) {  
    .....  
}
```

- Implémenter l'interface **implements View.OnClickListener** au niveau de l'activité

```
public void myClick(View v) {  
    if v.getId()==R.id.monBouton { gérer l'action }  
}
```

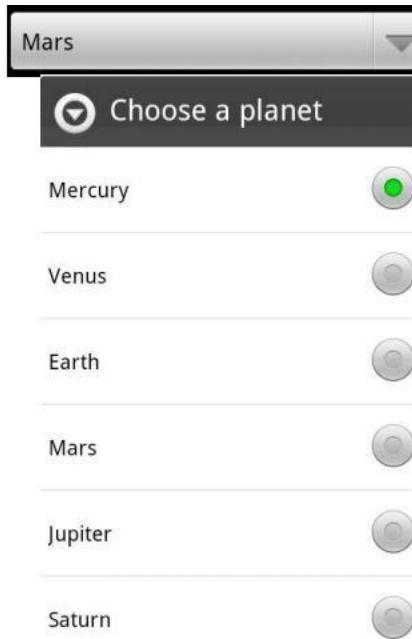
ex: **monBouton.setOnClickListener(this)**

3. Création d'IHM

Composants IHM Avancée

- Certains composants graphiques permettent d'afficher beaucoup d'items par des mécanismes de défilement:

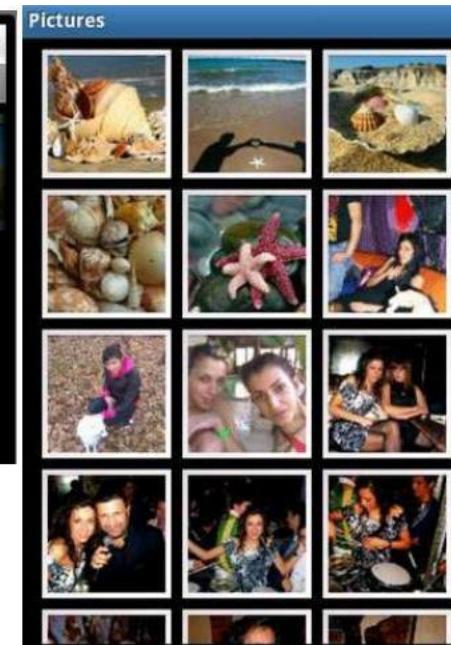
Spinner



Gallery



GridView



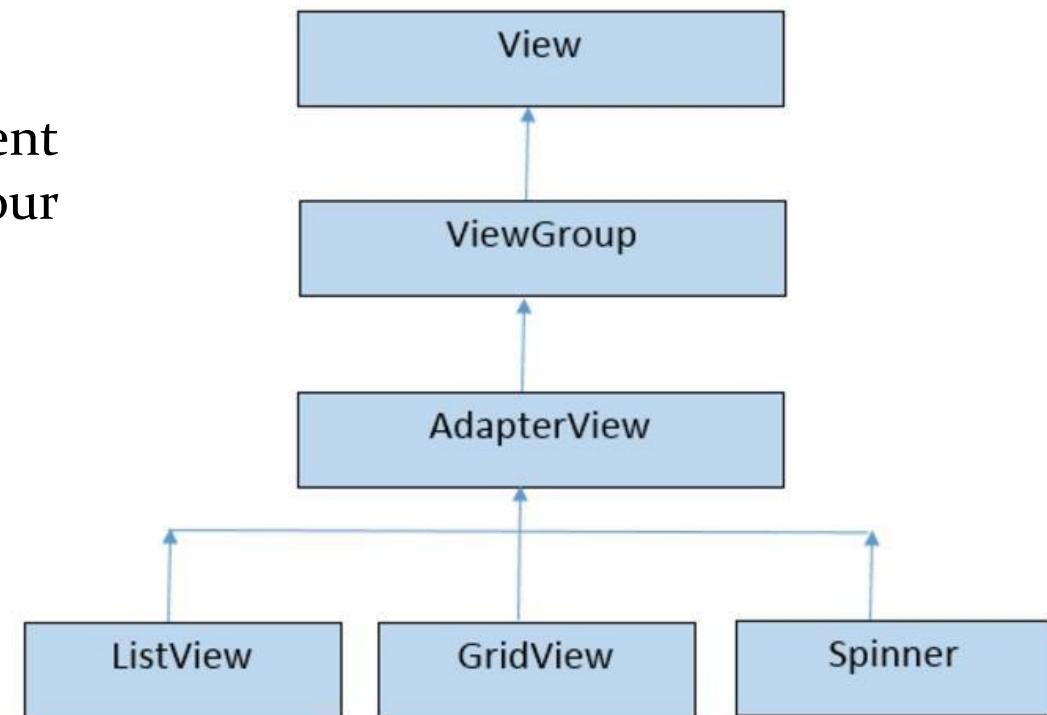
ListView/RecyclerView



3. Création d'IHM

AdapterView

AdapterView est une vue complexe qui contient plusieurs vues, et qui nécessite un adaptateur pour construire ses vues filles



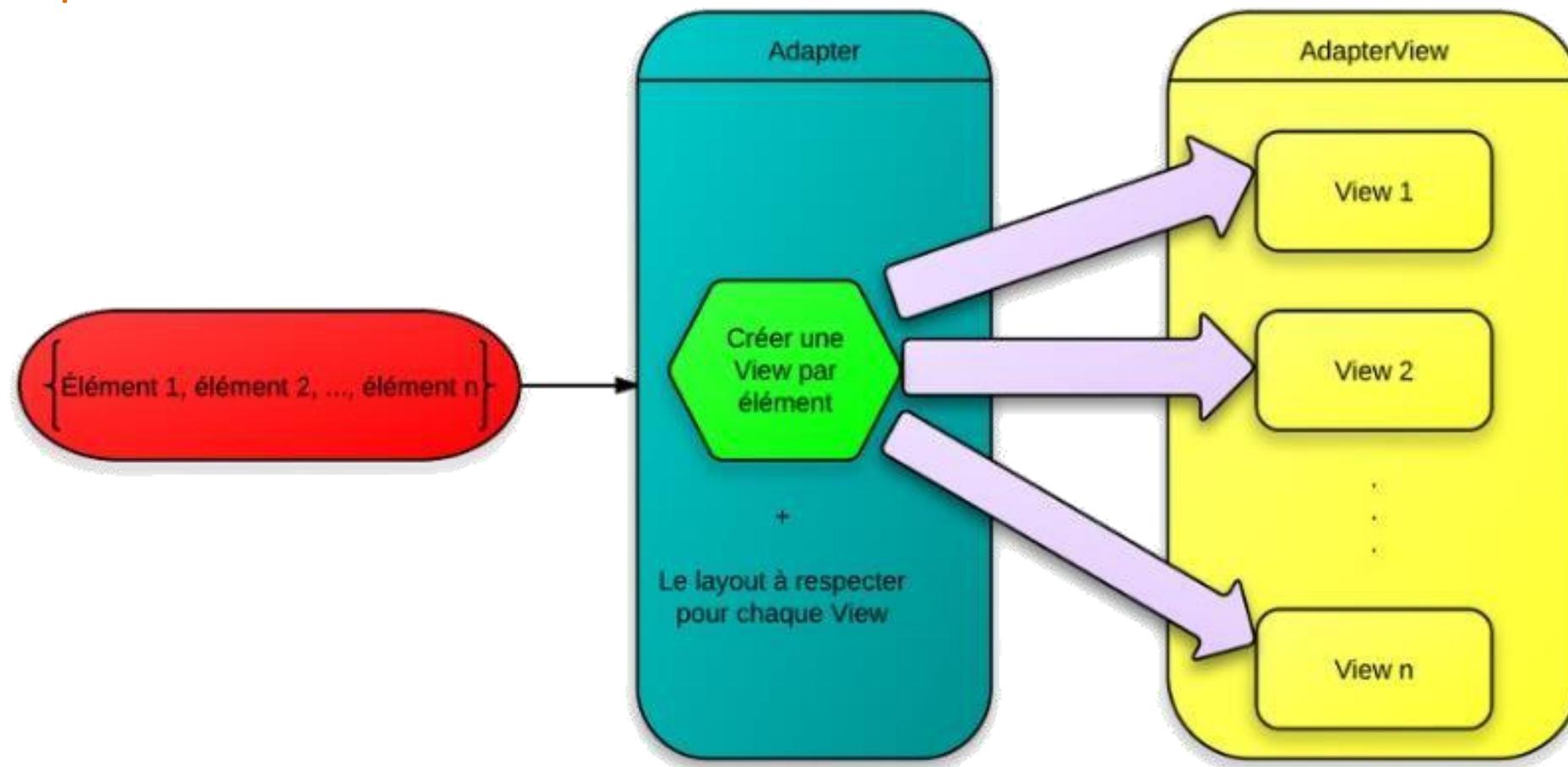
3. Création d'IHM

Adaptateur

- Les adaptateurs servent à gérer les données qui seront affichées dans les vues qui composent l'adapterView
- Ils réalisent la liaison entre les sources de données (tableau de string, BD, fournisseur de contenu...) et les contrôles de l'interface utilisateur (via notre AdapterView)
 - Pour ajouter des éléments à un AdapterView listView, Spinner...) on utilise des adaptateurs.

3. Création d'IHM

List & Adapter



3. Création d'IHM

List & Adapter

→ Android propose des adapters génériques que le développeur peut utiliser pour afficher des informations dans un élément de groupe

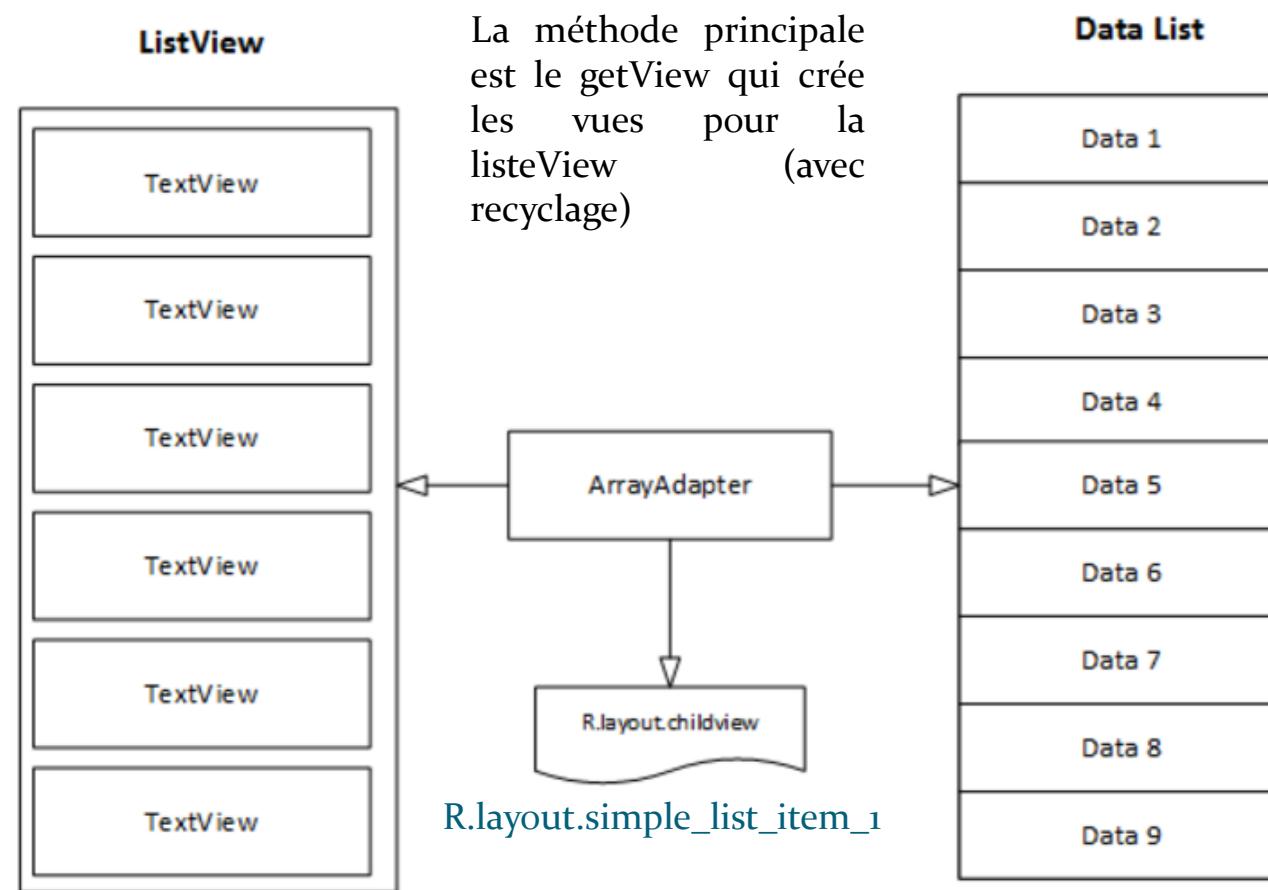
- BaseAdapter pour créer des adaptateurs personnalisés et c'est la base des autres adapter :
- ArrayAdapter pour tous type de tableaux
- Cursor Adapter pour traiter les données de type cursor

→...

→ Le développeur peut définir son propre adapter pour personnaliser l'affichage

3. Création d'IHM

EX: Simple ListView



3. Création d'IHM

Adaptateur personnalisé

Si le contenu de la liste est plus complexe qu'un textView , il faut définir votre propre adaptateur en respectant:

- 1- Définir un layout qui représente une vue
- 2-Créer une classe qui étend un adaptateur générique

Redéfinir la méthode `getView(int position, View convertView, ViewGroup parent)` qui est appelée à chaque fois qu'un item est affiché à l'écran. Permet de retourner pour chaque élément de l'AdapterView un objet view

Layout auquel
rattaché la vue

Permet le recyclage des vues (quand le nombre d'item > la taille de l'affichage de l'écran)
Si Null : on doit construire une nouvelle vue
Sinon : contient la valeur de la vue qui vient de disparaître.on peut modifier cette vue pour l'élément demandé

3. Création d'IHM

Adaptateur personnalisé

- Utiliser la classe statique interne `ViewHolder` qui permet de charger les données dans la vue pour l'afficher dans la `ListView`. C'est un objet attaché à chaque ligne de la `listView`

Ex: On suppose que notre layout contient des lignes de trois view : la source de données contient des éléments composés de trois données

```
static class ViewHolder
{
    public TextView label;
    public TextView desc;
    public ImageView img;
}
```

3. Création d'IHM

1. convertView est Null (la vue n'est pas dans le cache de listView):

- on initialise le convertView, en appelant la méthode inflate pour *charger notre layout xml dans un objet View*

`convertView = inflater.inflate(R.layout.item, null);`

- crée une instance (holder) de la classe ViewHolder qui permettra d'initialiser le contenu d'un item de la liste une première fois et *associe les composants de notre layout dans le holder*

`holder = new ViewHolder()`

`Ex : holder.label = (TextView)convertView.findViewById(R.id.label)`

- Sauvegarder (en mémoire) cette valeur dans un tag attaché à l'objet convertView
`convertView.setTag(holder)`

2. convertView n'est pas Null

- *réutilisation un holder déjà existant*

`holder = (ViewHolder) convertView.getTag();`

3. On rempli le holder (dans les deux cas)

`holder.label.setText(Source.get(position).getlabel());`

3. Création d'IHM

Dans class ArrayEtablissementAdapter:

```
@Override
// getView retournera la vue de l'item pour l'affichage.
public View getView(int position, View convertView, ViewGroup parent) {
    ViewHolder holder;
    // Réutilisation des vues
    if(convertView == null)
    {
        holder = new ViewHolder();
        //charger le fichier xml item
        convertView = inflater.inflate(R.layout.item, null);
        // Associer les views du layout item au holder pour le retouner à la listView
        holder.label = (TextView)convertView.findViewById(R.id.label);
        holder.desc = (TextView)convertView.findViewById(R.id.desc);
        holder.img= (ImageView) convertView.findViewById(R.id.img);
        //sauvgarde la ref du holder en memoire pour la réutilisation par la suite
        convertView.setTag(holder);
    }
    else {
        //réutilisation du holder déjà existant
        holder = (ViewHolder) convertView.getTag();
    }

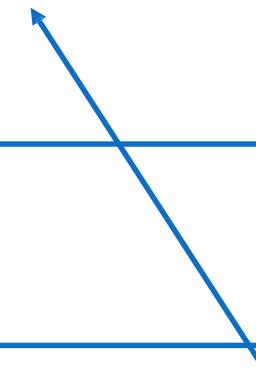
    //stocker les données dans une vue via settag
    holder.label.setText(Etablissements.get(position).getlabel());
    holder.desc.setText(Etablissements.get(position).getname());
    holder.img.setImageResource(Etablissements.get(position).getimage());
    return convertView;
}

static class ViewHolder
{
    public TextView label ;
    public TextView desc;
    public ImageView img;
}
```

3. Création d'IHM

Au niveau de la classe qui va afficher la ListView:

```
//liaison entre la source des données (Etablissements) et les controles du layout item i&gt;% afficher
adapter = new ArrayEtablissementAdapter(this, R.layout.item, Etablissements);
//L'association entre la ListView et l'Adapter
l = (ListView) findViewById(R.id.listview);
/// Binding resources Array to ListAdapter
l.setAdapter(adapter);
```



```
<ListView
    android:id="@+id/listview"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    tools:ignore="MissingConstraints" />
```

3. Création d'IHM

RecyclerView

- En 2014 (android Lollipop) a introduit le concept de RecycleView ayant les mêmes objectifs que Listview.. du système android
- C'est une bibliothèque qui n'est pas dans le noyau mais assure une compatibilité avec les anciennes version d'android
- Ceci permet à google de faire des mises à jour à cette bibliothèque tout en assurant la compatibilité avec toutes les versions d'android.
- Plus riche que ListView mais un peu plus compliqué
 - Les fonctionnalités de base sont assez similaires que ListView

3. Création d'IHM

RecyclerView

- LayoutManager est le responsable de l'organisation des vues (verticale , horizontale ou une grille) et du défilement ainsi que le recyclage.
 - RecyclerView support 3 types de LayoutManager : LinearLayoutManager, StaggeredLayoutManager et GridLayoutManager
- Recycleview nous force à utiliser les bonnes manières pour implémenter la liste
 - Ex: force l'usage du Viewholder (RecyclerView.ViewHolder)
- Possibilité de mettre des animations
 - RecyclerView.ItemAnimator

3. Création d'IHM

RecyclerView

- Pour créer une liste avec RecycleView on a besoin d':
 - ajouter le recyclerView au layout d'affichage
 - ajouter un layout pour chaque vue à afficher (ex ligne de la liste)
 - Ajouter un adapter qui fait la liaison entre la vue de la liste et la donnée du data list. Cette classe étend Recyclerview.Adapter avec un ViewHolder:
 - onCreateViewHolder qui inflate le layout et le passe au view holder
 - onBindViewHolder qui associe les données aux vues à l'aide du view holder

3. Création d'IHM

RecyclerView

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {  
  
    private final ArrayList Etablissements ;  
  
    public MyAdapter( ArrayList Etablissements) {  
        this.Etablissements = Etablissements ;  
    }  
    @Override  
    // retourne le nb total de cellule que contiendra la liste  
  
    public int getItemCount() {  
        return Etablissements.size();  
    }  
  
    public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
        LayoutInflator inflater = LayoutInflator.from(parent.getContext());  
        View view = inflater.inflate(R.layout.item, parent, false);  
        return new MyViewHolder(view);    }  
  
    @Override  
    // Inflate le layout et l'associe au view holder  
    public void onBindViewHolder(MyViewHolder holder, int position) {  
        Etablissement Etab= (Etablissement) Etablissements.get(position);  
        holder.name.setText(Etab.getname());  
        holder.label.setText(Etab.getLabel());  
        holder.img.setImageResource(Etab.getImage());    }  
  
    public class MyViewHolder extends RecyclerView.ViewHolder {  
        private final TextView label; private  
        final TextView name; private final  
        ImageView img; private Etablissement  
        currentEtab;  
  
        public MyViewHolder(final View itemView) {  
            super(itemView);  
            label = itemView.findViewById(R.id.label);  
            name = itemView.findViewById(R.id.name);  
            img= itemView.findViewById(R.id.img);  
            // quand on click sur l'établissement  
            itemView.setOnClickListener(new View.OnClickListener() {  
                @Override  
                public void onClick(View view) {  
                    // code pour traiter le clic  
                }  
            });  
        }  
    }  
}
```

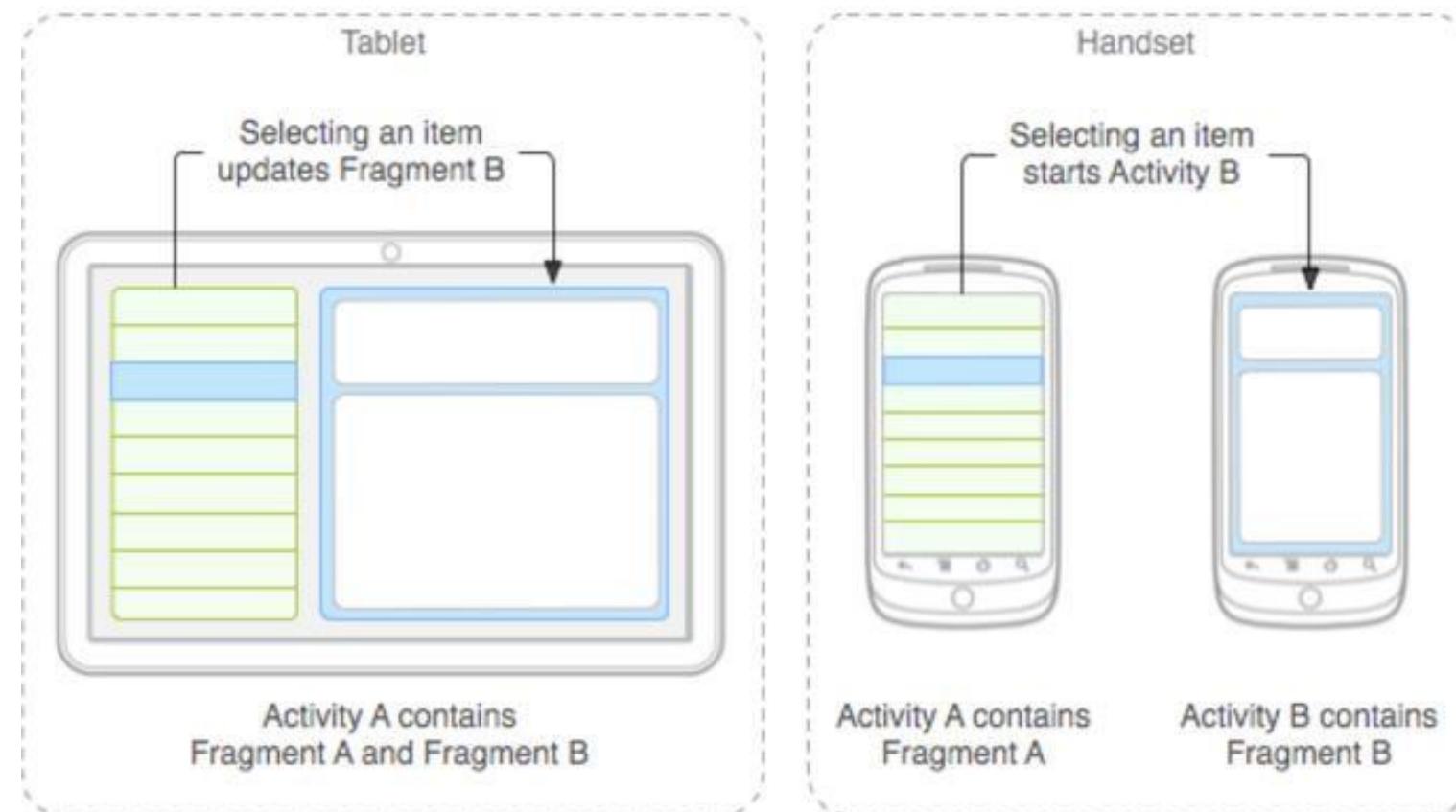
3. Création d'IHM

Fragement

- La taille des écrans devient de plus en plus grande (tablette, télévisions connectées...)
- une application android est destinée à être exécutée sur ces différents devices.
 - Besoin d'exploiter l'espace efficacement sur chacune de ces plates-formes.
- Google a introduit le fragment à partir des versions 3.X et a développé aussi une bibliothèque de compatibilité les version inférieurs de 3.X
 - `android.support.v4.app.fragment`

3. Création d'IHM

Fragement

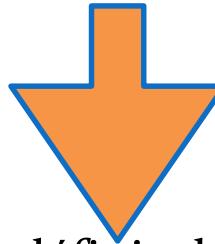


Duplicer les codes et créer deux interfaces (pour téléphone et pour mobile)

3. Création d'IHM

Fragement

- Un Fragment est un composant autonome avec sa propre interface utilisateur (UI) et son propre cycle de vie qui peut être réutilisé dans différentes parties de l'interface utilisateur d'une application.
- Fragment peut également être utilisé sans interface utilisateur, afin de conserver les valeurs malgré les modifications de configuration,



- Les fragments permettent de définir des morceaux d'interfaces, un ensemble de vues, qu'il est possible de contrôler et de réutiliser sur plusieurs écrans.
 - Plusieurs fragments peuvent être combinés dans une seule activité pour construire une interface à plusieurs volets (multi-pane)
 - Un fragment peut être réutilisé dans plusieurs activités

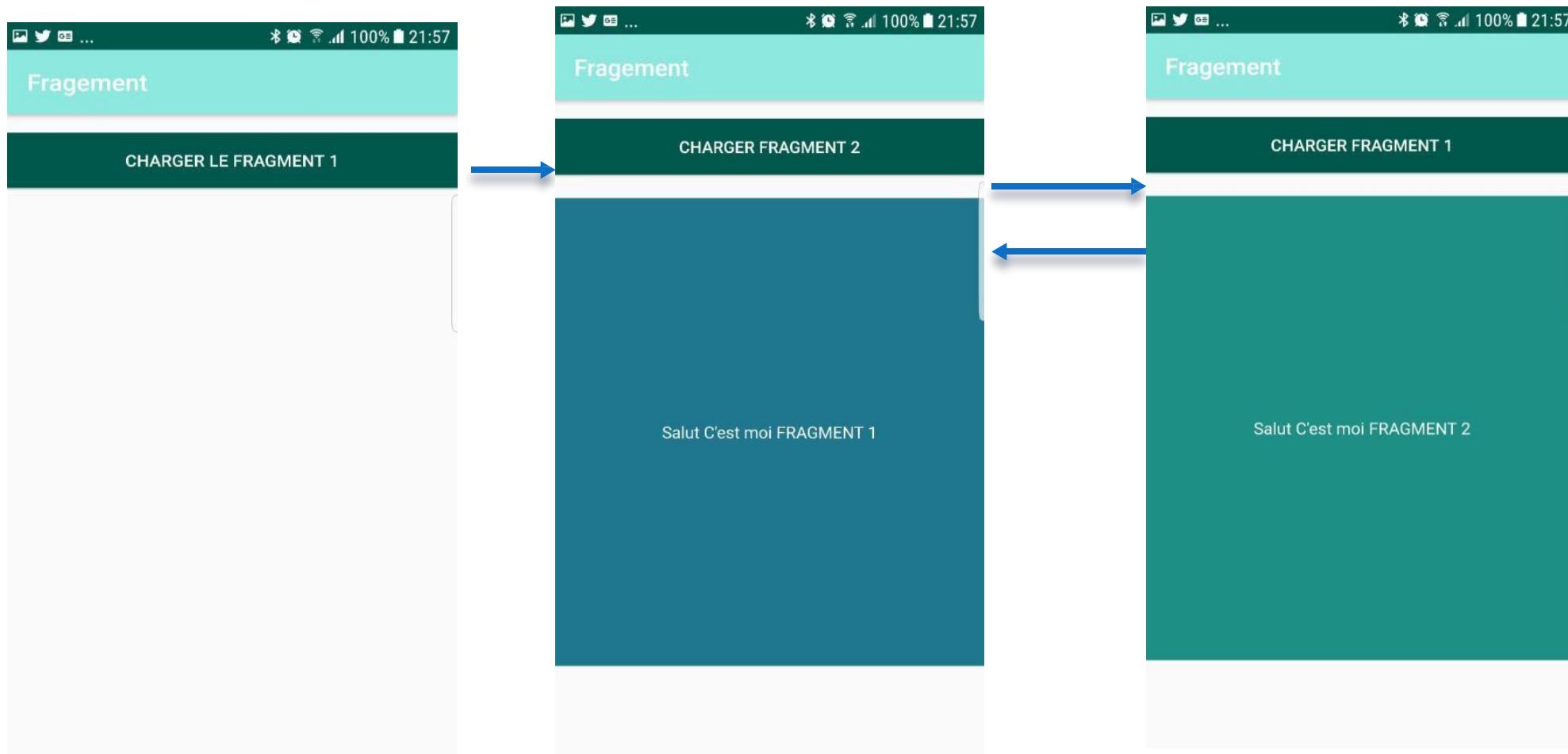
3. Création d'IHM

Fragement

- Un fragment
 - a son propre cycle de vie, mais directement affecté par celui de son hôte
 - reçoit ses propres entrées
 - peut être ajouté ou enlevé dynamiquement pendant que l'activité s'exécute
- Peut être ajouté à l'activité
 - Dans le fichier Layout XML par <fragment>
 - Dans le code de l'application en l'ajoutant à un conteneur (ViewGroup) existant via le FragmentManager

3. Création d'IHM

Fragement



3. Création d'IHM

Fragement: exemple

- Fragment : créer les 2 fragments + layouts associés

```
public class Fragment_two extends Fragment {  
  
    public Fragment_two() {  
        // Required empty public constructor  
    }  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                            Bundle savedInstanceState) {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.fragment_two_layout, container, false);  
    }  
}
```

- Dans l'activité principale on intègre les fragments
 - Cette activité est associée à son layout qui contient un conteneur qui va intégrer le fragment

3. Création d'IHM

Fragement: exemple

```
public class MainActivity extends AppCompatActivity {
    Button bt;
    Boolean etat = false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bt= (Button) findViewById(R.id.bn) ;
        bt.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentManager fragmentManager = getSupportFragmentManager();
                FragmentTransaction fragmentTransaction= fragmentManager.beginTransaction();
                if(!etat) {
                    Fragmentone fr1= new Fragmentone();
                    fragmentTransaction.add(R.id.fragment_container, fr1).commit();
                    bt.setText("charger Fragment 1");
                    etat = true;
                } else {
                    Fragment_two f2= new Fragment_two();
                    fragmentTransaction.add(R.id.fragment_container, f2);
                    fragmentTransaction.addToBackStack(null);
                    fragmentTransaction.commit();
                    bt.setText("charger Fragment 2");
                    etat = false;
                }
            }
        });
    }
}
```

4. Elément d'Interaction

4. Elément d'Interaction

Intents, récepteurs, notifications

- Éléments d'interaction
 - Intents, récepteurs, notifications
 - Ces éléments permettent l'interaction entre les différents composants du système, entre les applications installées sur l'appareil ou avec l'utilisateur

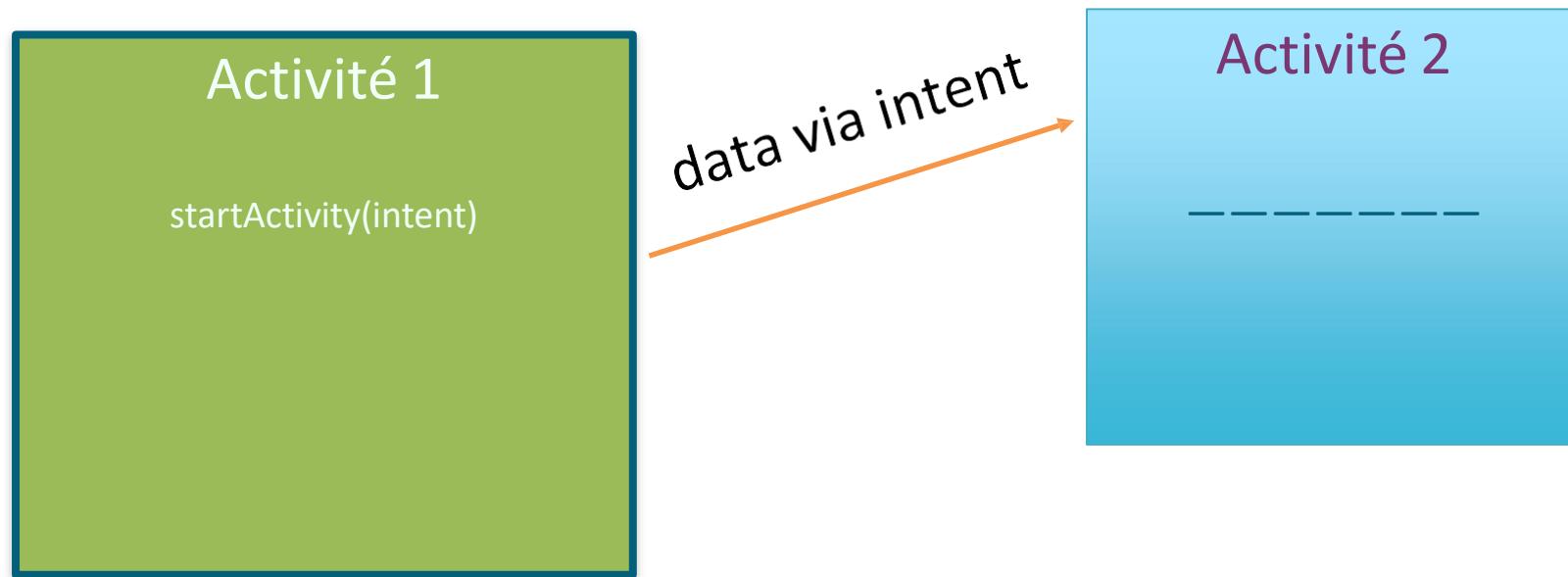
4. Elément d'Interaction

Communication entre applications classe Intent

- **Les Intents** sont des messages asynchrones qui permettent aux composants d'une application de demander des fonctionnalités à d'autres composants Android.
- **L'intent** permet d'invoquer les activités :
 - **startActivity(intent)** pour lancer une activité
 - **startActivityForResult (intent, request_code)** (si on veut recevoir un résultat de l'activité invoquée)
 - On doit définir dans l'activité appelante « **onActivityResult** » qui est appelée lorsque l'activité appelé prend fin(retour).
 - **data.getExtras().getString(key)** : récupérer les données
 - On doit définir dans l'activité fille, la méthode **setResult(Resultcode, data)**

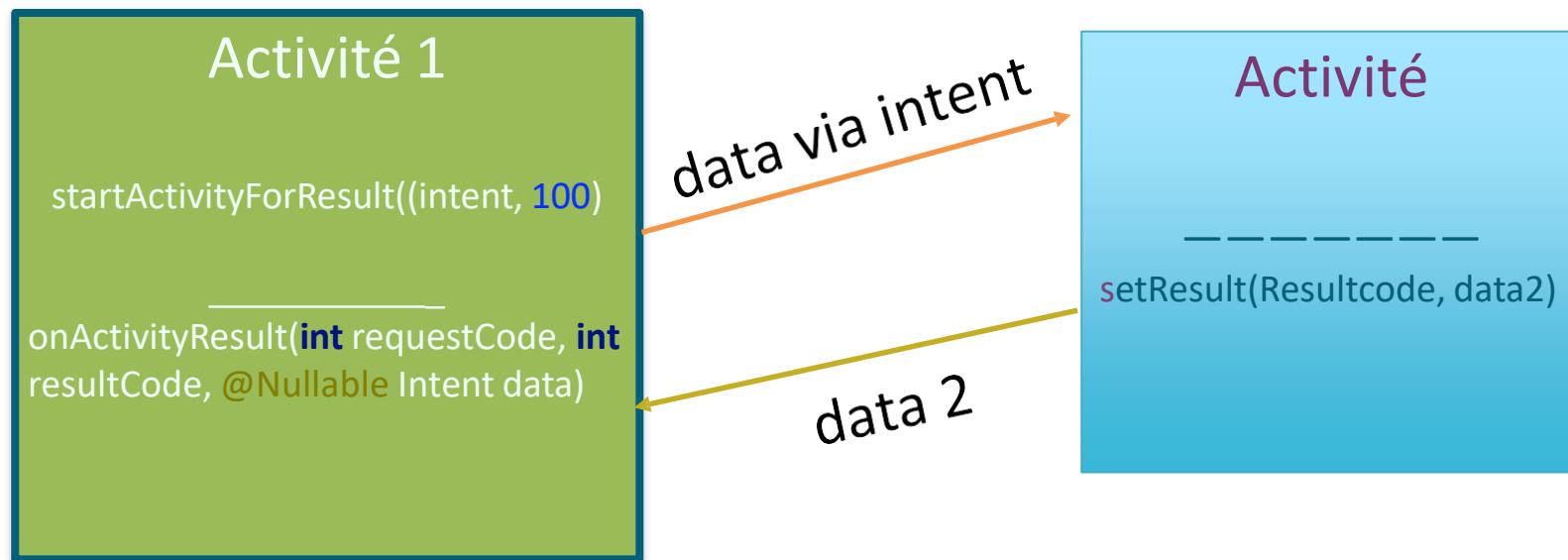
4. Elément d'Interaction

Communication entre deux activités (cas 1)



4. Elément d'Interaction

Old Communication entre deux activités (cas 2)

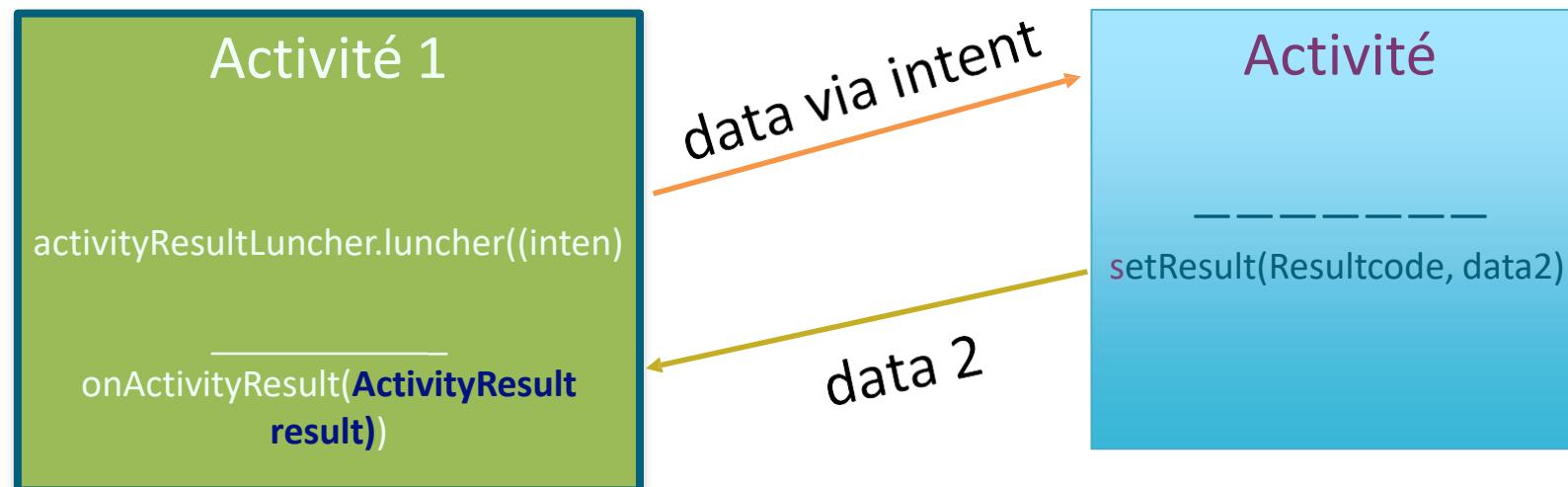


4. Elément d'Interaction

New Communication entre deux activités (cas 2)

En 2020, Google a introduit l'API Activity Result qui permet l'échange entre activités.

- registerForActivityResult renvoie **ActivityResultLuncher** pour démarrer une autre activité.
- Elle a comme paramètres **ActivityResultContract** qui définit le type de sortie du résultat et **ActivityResultCallback** qui est une interface avec la méthode **onActivityResult**



4. Elément d'Interaction

Intent

- Un Intent sert à demander la réalisation d'une action par un autre composant de la plate-forme Android (activity, service, content provider, broadcast receiver)
 - Il est aussi utilisé pour véhiculer toutes les informations nécessaires à la réalisation de l'action demandée.
- Une application peut définir les composants cibles soit:
 - directement dans l'Intent → **Intent explicite**
 - en demandant au système Android de rechercher le composant adéquat en se basant sur les données de l'Intent **Intent implicite**

4. Elément d'Interaction

Intent: Création

- Crédit d'un Intent :
 - new Intent(Context c, Class<?> cls) pour appel explicite
 - pour un appel inter-application, on peut utiliser

```
Intent intent = new Intent()  
intent.setClassName(String package, String className)
```

- new Intent(String action, Uri uri) pour appel implicite

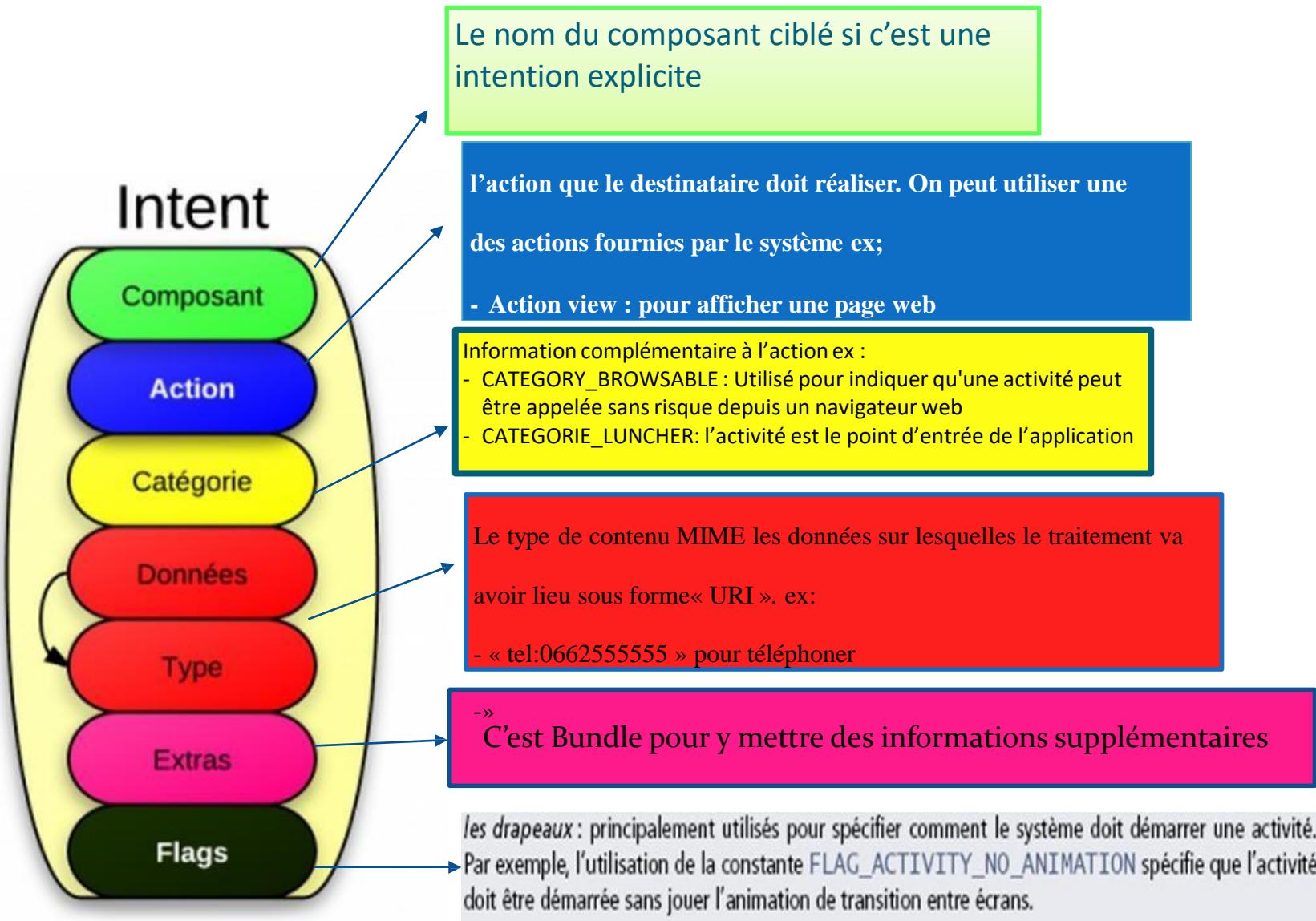
4. Elément d'Interaction

Intent

- Les informations véhiculées par l'intent sont de deux types:
 - Informations à destination de l'application qui va recevoir cet intent(action à effectuer et les données avec lesquelles agir);
 - Information nécessaires au système pour son traitement(catégorie du composant cible du message et instructions d'exécution de l'action)

4. Elément d'Interaction

Objet Intent



4. Elément d'Interaction

Injection des données dans un intent

- Les *Intents* permettent de transporter des informations (Extra) à destination de l'activité cible
 - `putExtra` : insertion des extra
 - `getExtra` : récupérer les extrats d'un intent

4. Elément d'Interaction

Intent: Naviguer entre écrans au sein d'une application

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    connecter= (Button) this.findViewById(R.id.button);  
    connecter.setOnClickListener(new View.OnClickListener() {  
        public void onClick(View v) { myClick(v);  
        }  
    );  
  
}  
  
private void myClick(View v) { iv  
    Intent intent=new Intent(this, Test.class) ;  
    //charger donnée  
    intent.putExtra("madonnee","Hi");  
    //Démarrer l'activité  
    this.startActivity(intent);
```

4. Elément d'Interaction

Intent: Naviguer entre écrans au sein d'une application(suite)

- Démarrer une activité

```
public void onClick(View v) {  
    if(v == monBouton) {  
        Intent monIntent = new Intent(this,MonAutreActivite.class);  
        startActivity(monIntent);  
    }  
}
```

4. Elément d'Interaction

ActivityResultLuncher

```
activityResultLauncher= registerForActivityResult(new ActivityResultContracts.StartActivityForResult(),
    new ActivityResultCallback<ActivityResult>() {
        @Override
        public void onActivityResult(ActivityResult result) {

            if (result.getResultCode()== 2000 && result.getData()!=null ){
                edit.setText(""+ result.getData().getExtras().getString("Val_Retournée"));
            } } );}

private void myClick(View v) {
    Intent intent = new Intent (this, MainActivity.class);
    intent.putExtra("madonne", "Hi");
    //this.startActivity(intent);
    if (intent.resolveActivity(getApplicationContext())!=null){
        activityResultLauncher.launch(intent);
    }

    // Donnée à envoyer par la sous activité (retour d'information)
    Intent data = new Intent();
    data.putExtra("Val_Retournée", "Bonjour");
    // Activité ayant le code 2000 retourne l'info
    this.setResult(2000,data);

}
```

4. Elément d'Interaction

Communication entre application

- Intent (filter) : délégué au système le choix de l'application (usage des Intent-Filters)
- Broadcast Receivers
 - sendBroadcast(intent) qui envoie un intent à tous les composants Broadcast Receivers intéressés
- Services

pour communiquer avec un service en arrière plan (local ou distant).

4. Elément d'Interaction

Communication entre application

Déléguer au système le choix de l'application

- Vous pouvez envoyer une intention et demander au système de choisir le composant le plus approprié pour exécuter l'action transmise.

Ex: Composer un numéro de téléphone

```
Uri uri = Uri.parse("tel:0612345678");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent)
```

4. Elément d'Interaction

Communication entre application

Déléguer au système le choix de l'application

- Vous pouvez envoyer une intention et demander au système de choisir le composant le plus approprié pour exécuter l'action transmise.

Ex: Composer un numéro de téléphone

```
Uri uri = Uri.parse("tel:0612345678");
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent)
```

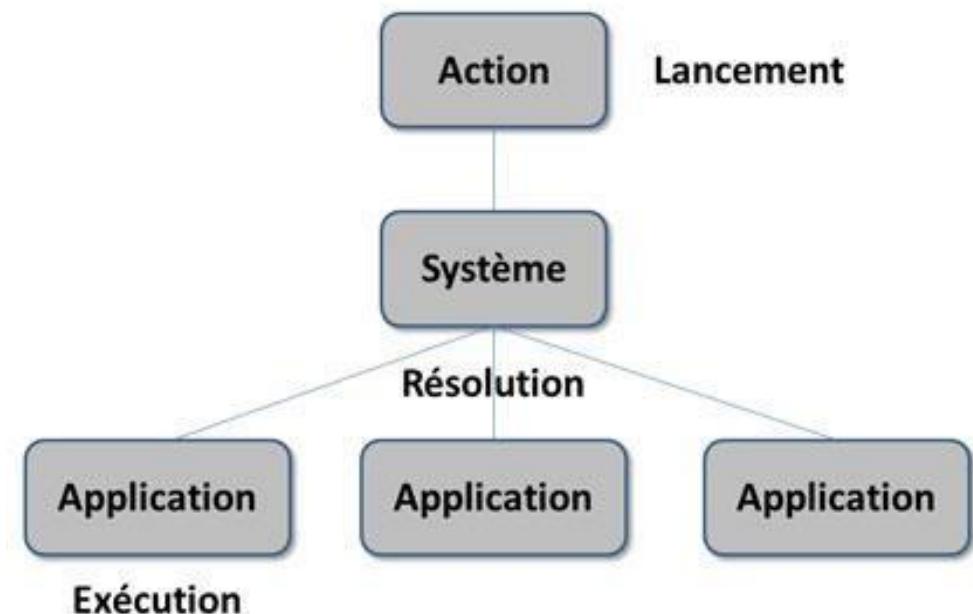
4. Elément d'Interaction

Communication entre application

Solliciter d'autres applications

Pour décider du composant le plus approprié, le système se base sur les informations que vous spécifiez dans votre objet Intent : action, données, catégorie, etc.

Si plusieurs composants sont enregistrés pour une même action..., un choix est présenté à l'utilisateur



```
if (sendIntent.resolveActivity(getApplicationContext()) != null) {  
    startActivity(sendIntent);  
}
```

Si aucune application n'est disponible et que la vérification n'est pas effectuée alors l'application plante

4. Elément d'Interaction

Communication entre application

Si aucune application n'est disponible

```
public void openWebPage(String url) {  
    Uri webpage = Uri.parse(url);  
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);  
    if (intent.resolveActivity(getApplicationContext()) != null) {  
        startActivityForResult(intent);  
    }  
}
```

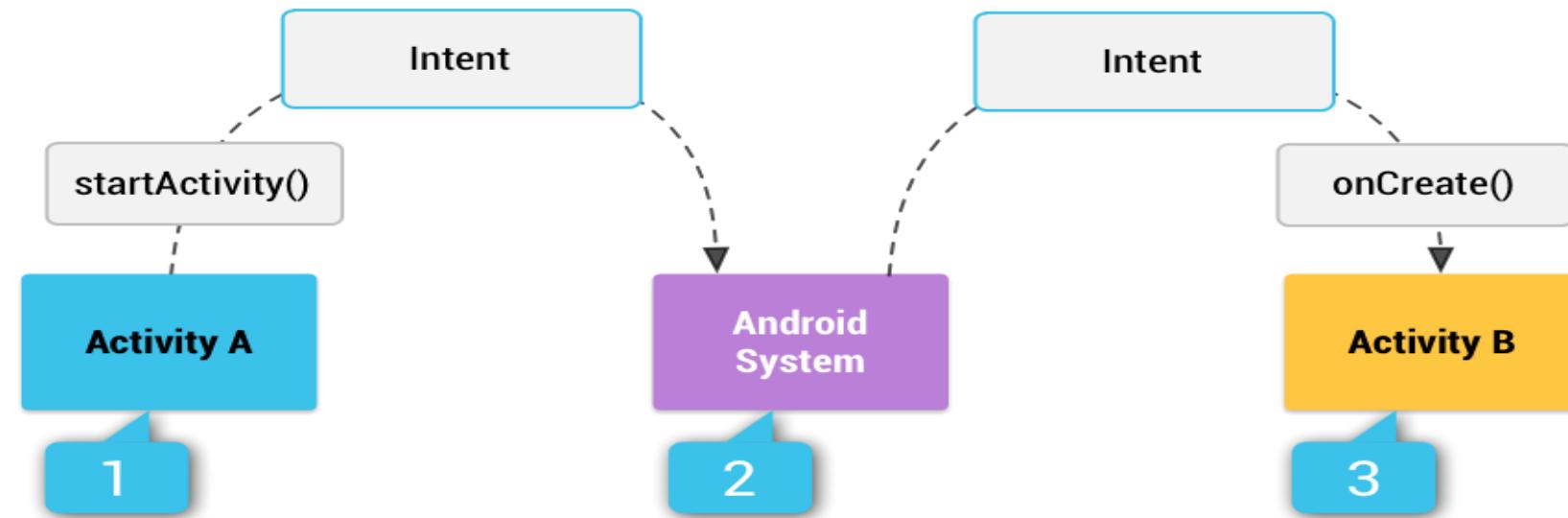


Renvoie le composant d'activité à utiliser pour gérer l'intention

4. Elément d'Interaction

Communication entre application

EX: Intent Implicit



- Activité A crée un intent avec une description de l'action et passe à startActivity ().
- Le système Android recherche toutes les applications pour un filtre intention qui correspond à l'intention. Lorsqu'une correspondance est trouvée,
 - le système démarre l'activité de correspondance (activité B) en invoquant onCreate ()

4. Elément d'Interaction

Filtres d'intents

- Android permet aux applications de spécifier quelles sont les actions qu'elles gèrent : le système peut ainsi choisir le composant le mieux adapté au traitement d'une action (véhiculée dans un objet Intent).

4. Elément d'Interaction

Intents et Intent-Filters

- L'utilisation d'une activité (ou d'un service) se fait par une requête auprès du système android.
- chaque activité (ou service) déclare les fonctionnalités(actions) pour lesquelles elle est disponible (intent filter).

Lorsqu'une application désire utiliser une fonctionnalité(action), elle déclare son intention de le faire (intent), le système android détermine et démarre l'activité/service correspondant

4. Elément d'Interaction

Intent: Les actions natives

Action	Définition
ACTION_ANSWER	Prendre en charge un appel entrant.
ACTION_CALL	Appeler un numéro de téléphone. Cette action lance une activité affichant l'interface pour composer un numéro puis appelle le numéro contenu dans l'URI spécifiée en paramètre.
ACTION_DELETE	Démarrer une activité permettant de supprimer une donnée identifiée par l'URI spécifiée en paramètre.
ACTION_DIAL	Afficher l'interface de composition des numéros. Celle-ci peut être pré-remplie par les données contenues dans l'URI spécifiée en paramètre.
ACTION_EDIT	Éditer une donnée.
ACTION_SEARCH	Démarrer une activité de recherche. L'expression de recherche de la pourra être spécifier dans la valeur du paramètre SearchManager.QUERY envoyé en extra de l'action.
ACTION_SEND	Envoyer des données texte ou binaire par courriel ou SMS. Les paramètres dépendront du type d'envoi.
ACTION_SENDTO	Lancer une activité capable d'envoyer un message au contact défini par l'URI spécifiée en paramètre.
ACTION_VIEW	Démarrer une action permettant de visualiser l'élément identifié par l'URI spécifiée en paramètre. C'est l'action la plus commune. Par défaut les adresses commençant par <i>http:</i> lanceront un navigateur web, celles commençant par <i>tel:</i> lanceront l'interface de composition de numéro et celles débutant par <i>geo:</i> lanceront Google Map.
ACTION_WEB_SEARCH	Effectuer une recherche sur Internet avec l'URI spécifiée en paramètre comme requête.

4. Elément d'Interaction

Intent-Filters

- Dans un filtre on met:
 - L'action: identifiant unique (chaine de caractère)

```
<activity>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <action android:name="android.intent.action.SENDTO" />
    </intent-filter>
</activity>
```

- Catégorie: premier niveau de filtrage de l'action. Elle définit le contexte de l'action (n'est pas obligatoire dans un intent)

4. Elément d'Interaction

Intent-Filters

EX :

```
<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>
```

- *Parmi ces activités, l'une est plus importante que les autres : elle apparait dans le intent-filter avec la catégorie «launcher».*

4. Elément d'Interaction

Intent-Filters

Catégorie	Description
CATEGORY_DEFAULT	Indique qu'il faut effectuer le traitement par défaut sur les données correspondantes. Concrètement, on l'utilise pour déclarer qu'on accepte que ce composant soit utilisé par des intents implicites.
CATEGORY_BROWSABLE	Utilisé pour indiquer qu'une activité peut être appelée sans risque depuis un navigateur web. Ainsi, si un utilisateur clique sur un lien dans votre application, vous promettez que rien de dangereux ne se passera à la suite de l'activation de cet intent.
CATEGORY_TAB	Utilisé pour les activités qu'on retrouve dans des onglets.
CATEGORY_ALTERNATIVE	Permet de définir une activité comme un traitement alternatif dans le visionnage d'éléments. C'est par exemple intéressant dans les menus, si vous souhaitez proposer à votre utilisateur de regarder telles données de la manière proposée par votre application ou d'une manière que propose une autre application.
CATEGORY_SELECTED_ALTERNATIVE	Comme ci-dessus, mais pour des éléments qui ont été sélectionnés, pas seulement pour les voir.
CATEGORY_LAUNCHER	Indique que c'est ce composant qui doit s'afficher dans le lanceur d'applications.
CATEGORY_HOME	Permet d'indiquer que c'est cette activité qui doit se trouver sur l'écran d'accueil d'Android.
CATEGORY_PREFERENCE	Utilisé pour identifier les PreferenceActivity (dont nous parlerons au chapitre suivant).

4. Elément d'Interaction

Intent-Filters

- Données: Il est possible de préciser plusieurs informations sur les données que cette activité peut traiter. Ex préciser le type MIME avec android:mimeType, préciser le nom de domaine via l'attribut android:host.

EX:

```
<data android:mimeType="text/plain" android:host="ensias.ma" />
```

4. Elément d'Interaction

Permissions liées aux actions

- Certaines actions ont besoin des privilèges spéciaux →
Ajouter des permissions dans le AndroidManifest
 - Ex Pour l'action ACTION_CALL on ajoute android.permission.CALL_PHONE

```
<manifest....>
<uses-permission
    android:name="android.permission.CALL_PHONE" />

</manifest>
```

4. Elément d'Interaction

Broadcast Intent

- L'objectif de l'intent est de demander à un autre composant de traiter son action
- L'objectif du Broadcast Intent est de prévenir les autres applications qu'un évènement est survenu (pour les intéressés)
 - C'est des intents anonymes et diffusés à tout le système
 - Très utilisé au niveau système pour transmettre des informations: appel entrant, réseau WI-FI connecté, état batterie...
- Usage d'un système de filtrage pour déterminer qui peut recevoir l'intent

4. Elément d'Interaction

Broadcast Intent/ Broadcast Receiver

- Etapes de diffusions et réception d'intent

Etape 1 : diffuser l'intent

- `sendBroadcast (intent)` : envoie l'intent au système

```
Public void doBroadcast(){  
    Intent intent = new Intent("com.exemple.MON_INTENT");  
    Intent.putExtra("extra", "hello")  
    sendBroadcast(intent)  
}
```

4. Elément d'Interaction

Broadcast Intent/ Broadcast Receiver

Etape 2: recevoir et traiter l'intent diffusé

- Ecouter le flux de message via la méthode BroadcastReceiver
 - Ajoutez une classe qui dérive de BroadcastReceiver (pas IHM associé) et implémenter le onReceive
 - À chaque réception de message il y a appel de la méthode onReceive par le système (l'application n'a pas besoin d'être en exécution)

```
public final class MyBroadcastReceiver extends BroadcastReceiver {  
    public static final String VIEW =  
        "eyrolles.intent.action.VIEW";  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        ... // Insérer le code de traitement de l'Intent ici.  
    }  
}
```

4. Elément d'Interaction

Broadcast Intent/ Broadcast Receiver

- **Enregistrez le récepteur de diffusion:**

Ajouter dans le manifeste le composant receiver portant le nom de la classe qui dérive de BroadcastReceiver

```
<manifest ...>
  <application ...>
    ...
      <receiver android:name="MyBroadcastReceiver">
        <intent-filter>
          <action android:name="com.exemple.MON_INTENT" />
        </intent-filter>
      </receiver>
    ...
  </application>
</manifest>
```



Le récepteur écoutera toujours l'événement même si l'application est détruite

4. Elément d'Interaction

Broadcast Intent/ Broadcast Receiver

Sécurité

- N'importe quelle application peut envoyer des broadcast intents à votre receiver, ce qui est une faiblesse au niveau sécurité.

Vous pouvez faire en sorte que votre receiver déclaré dans le Manifest ne soit accessible qu'à l'intérieur de votre application en lui ajoutant l'attribut `android:exported="false"`.

4. Elément d'Interaction

Broadcast Intent/ Broadcast Receiver

- On peut aussi ajouter une permission au niveau de sendBroadcast ainsi seuls les receivers qui déclarent cette permission pourront recevoir ces broadcast intents

```
private String COUCOU_BROADCAST =  
"sdz.chapitreTrois.permission.COUCOU_BROADCAST";  
  
...  
  
sendBroadcast(i, COUCOU_BROADCAST);
```

- Puis dans le Manifest, il suffit de rajouter :

```
<uses-permission  
    android:name="sdz.chapitreTrois.permission.COUCOU_BROADCAST"/>
```

4. Elément d'Interaction

Composant applicatif: Service

- Un service est un composant d'application qui peut exécuter des opérations en arrière-plan et ne fournit pas d'interface utilisateur. android.app.Service
 - Un autre composant d'application peut démarrer un service et il continue à s'exécuter en arrière-plan même si l'utilisateur passe à une autre application.
- un composant peut se lier à un service pour interagir avec lui et même effectuer la communication interprocessus (IPC)
 - Ex un service peut gérer des transactions réseau, lire de la musique, effectuer des E/S de fichiers ou interagir avec un fournisseur de contenu, tous cela en arrière-plan.

4. Elément d'Interaction

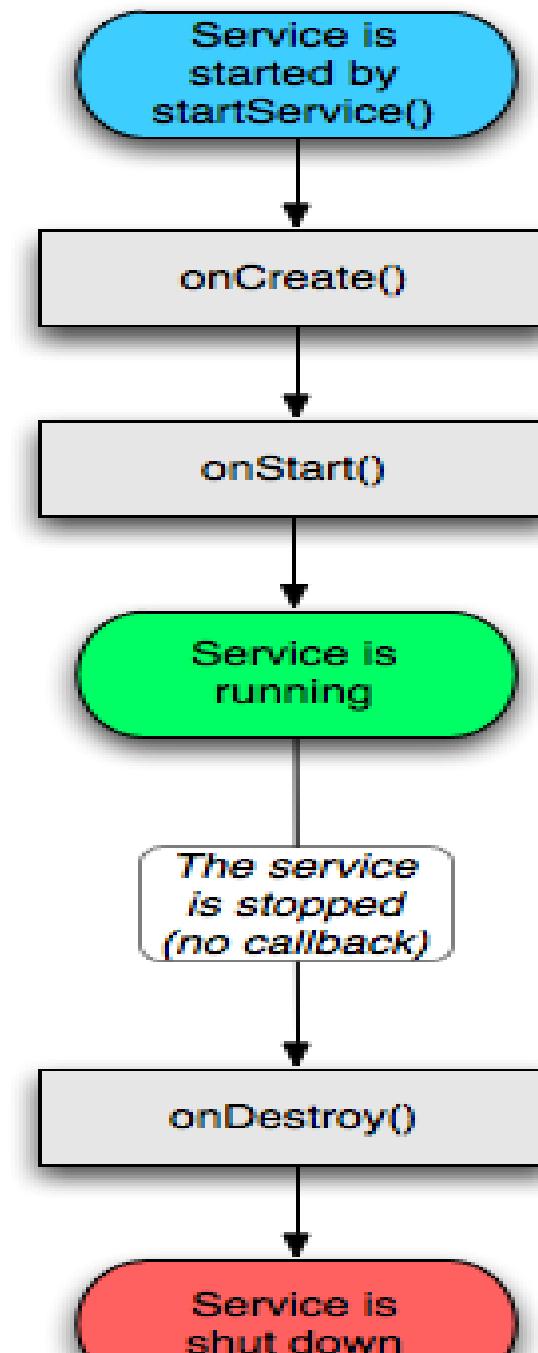
Composant applicatif: Service

- Deux types de services
 - **LocalService** : Services qui s'exécutent dans **le même processus** que votre application.
 - **RemoteService** : Ils s'exécutent dans **des processus indépendants** de votre application.

4. Elément d'Interaction

Cycle de vie du service

Similaire au cycle de vie de l'activité, il peut être contrôlé par d'autres applications (activités, services ...)



4. Elément d'Interaction

Persistante des données

- Android offre plusieurs méthodes pour stocker les données d'une application. La solution choisie va dépendre des besoins :
 - données privées ou publiques,
 - un petit ensemble de données à préserver
 - ou un large ensemble à préserver localement ou via le réseau.

4. Elément d'Interaction

Persistante des données

- Quatre techniques sont à la disposition du développeur pour faire persister des données :
 - La persistante des données de l'activité
 - Les préférences partagées
 - Stockage des données dans des fichiers
 - Stockage des données dans une base de donnée

4. Elément d'Interaction

Persistante des données de l'activité

l'enregistrement du parcours de l'utilisateur dans l'application (écrans rencontrés avec leur contexte) ou *persistence des activités* (*un écran étant lié à une activité*).

- les méthodes du cycle de vie de l'activité facilitent l'enregistrement et la gestion de l'état de l'interface tout au long du cycle de vie de l'activité (1 objet Bundle qui permet de restaurer les View qui possèdent un id)
- La donnée n'est pas persistante et n'est disponible que si l'application est utilisée

4. Elément d'Interaction

Persistante des données de l'activité

Android peut arrêter l'activité et la redémarrer quand il y a:

- Rotation de l'écran.
- Changement de langue.
- L'application est en arrière-plan et le système a besoin de ressources.
- Le click sur le « retour » (« back »).

4. Elément d'Interaction

Persistance des données de l'activité

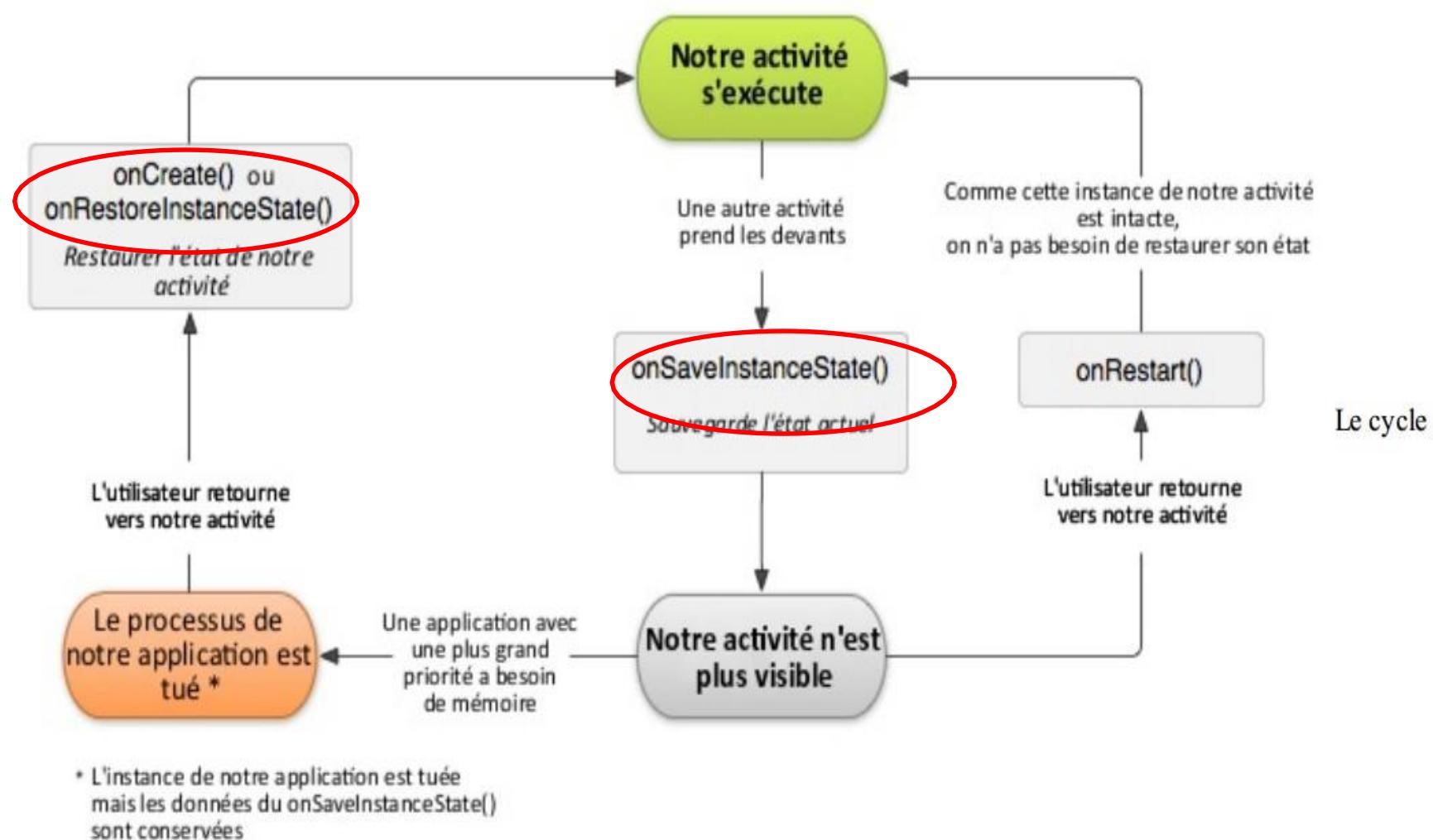
Ex: `@Override`

```
protected void onSaveInstanceState(Bundle savedInstanceState) {  
    super.onSaveInstanceState(savedInstanceState);  
    Toast.makeText(this, "État de l'activité sauvegardé",  
        Toast.LENGTH_SHORT).show();  
}
```

On peut redéfinir la méthode
`onRestoreInstanceState(Bundle)`, qui est appelée
après `onStart()` et qui recevra le même objet

```
/**  
 * Appelée après onCreate.  
 * Les données sont rechargées et l'interface utilisateur.  
 * est restaurée dans le bon état.  
 */  
@Override  
public void onRestoreInstanceState(Bundle savedInstanceState) {  
    super.onRestoreInstanceState(savedInstanceState);  
    //Placez votre code ici  
}
```

4. Elément d'Interaction



4. Elément d'Interaction

Préférences partagées

- enregistrer et retrouver d'autres informations propres à l'application(les paramètres de l'utilisateur, la configuration de l'application, etc) ex: conserver la couleur préférée de l'utilisateur
- Ce mécanisme d'enregistrement, *permet la persistance de propriétés sous la forme d'un ensemble de paires clé/valeur*
- Les préférences partagées peuvent stocker des données de types boolean, int, long, float et String dans un fichier

Les données ne sont pas cryptées

4. Elément d'Interaction

Préférences partagées

- La classe **SharedPreferences** permet de gérer des paires de clé/valeurs associées à une activité. On récupère un tel objet par l'appel à **getPreferences**
- Pour ajouter ou modifier des couples dans un SharedPreferences, il faut utiliser un objet de type SharedPreferences.Editor

4. Elément d'Interaction

Préférences partagées

Si vous avez besoin de plusieurs fichiers que vous identifierez par leurs noms, alors utilisez `getSharedPreferences`

Mode: pour que l'accès ne soit que par l'application qui l'a créé

```
Prefs = getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = prefs.edit();
editor.putString("nomUtilisateur", PrefValeur);
editor.apply();
```

Usage d'un fichier xml standard par activité

```
prefs = getPreferences(MODE_PRIVATE);
PrefValeur = prefs.getString("nomUtilisateur", "Introuvable");
if(PrefValeur!=null) {
//utiliser la valeur
}
```

4. Elément d'Interaction

Modifications des préférences et événements

- La classe SharedPreferences propose un mécanisme d'événements pour vous permettre de maîtriser les changements de données effectués sur les préférences partagées par l'utilisateur
 - registerOnSharedPreferenceChangeListener permet de spécifier une méthode de rappel/écouteur qui sera appelée à chaque modification de vos préférences.
 - Les préférences sont stockées dans des fichiers du système de fichiers interne d'Android

4. Elément d'Interaction

Modifications des préférences et événements

```
SharedPreferences prefs = getSharedPreferences(MES_PREFERENCES,  
        Context.MODE_PRIVATE);  
String nom = prefs.getString("nomUtilisateur", null);  
if (nom != null)  
    setNomUtilisateur(nom);  
  
// Enregistrement des écouteurs de changement de valeurs.  
prefs  
    .registerOnSharedPreferenceChangeListener(  
        new OnSharedPreferenceChangeListener() {  
  
            @Override  
            public void onSharedPreferenceChanged(  
                SharedPreferences sharedPreferences, String key) {  
                // Mettez votre traitement ici.  
            }  
        };
```

On a spécifié le fichier de préférence

4. Elément d'Interaction

Ecran de préférences

- Vous pouvez créer des écrans de préférences
 - L'activité PreferenceActivity permet de construire l'interface de votre menu de préférences
 - Vous pouvez aussi utiliser une simple activité qui appelle un fragment qui étendent PreferenceFragment (pour version >14)
 - Recommandée pour les apps développé sous android 3.0 ou plus

4. Elément d'Interaction

Ecran de préférences

- Les étapes nécessaires pour implémenter une activité de préférences (via l'utilisation d'un fichier XML) sont les suivantes.
 - Créer un fichier XML avec la structure des paramètres.
 - La racine de ce fichier doit être PreferenceScreen.
 - Mettre le fichier .xml dans res/xml
 - Créer une classe qui étends PreferenceFragment
 - Charger la structure XML des paramètres.
 - Créer une activité qui étend PreferenceActivity
 - Récupère le fragment
 - Récupérer les valeurs de chaque paramètre à l'aide de getDefaultSharedPreferences.

4. Elément d'Interaction

Ecran de préférences



4. Elément d'Interaction

Exe

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

    <PreferenceCategory android:title="Utilisateur">

        <EditTextPreference
            android:key="login"
            android:title="Nom d'utilisateur"
            android:summary="Entrez votre nom d'utilisateur">
        </EditTextPreference>

        <EditTextPreference
            android:title="Mot de passe"
            android:key="password"
            android:summary="Entrez votre mot de passe"
            android:password="true">
        </EditTextPreference>

    </PreferenceCategory>

    <PreferenceCategory android:title="Notifications">

        <RingtonePreference>
```

4. Elément d'Interaction

Exemple

- Créez un fichier XML qui présente le PreferenceScreen(ex prefs.xml)
- Créez une classe qui étend PreferenceFragment

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    addPreferencesFromResource(R.xml.prefs);  
}
```

- Créez une activité qui étend PreferenceActivity

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    getFragmentManager().beginTransaction().replace(android.R.id.content,  
        new SettingFragment()).commit();  
}
```

Nom du fragment

4. Elément d'Interaction

Stockage dans des fichiers (internes)

- En plus des API standards de manipulation de l'espace de noms `java.io` (`FileInputStream`, `FileOutputStream`), Android propose deux méthodes, `openFileOutput` et `openFileInput`, pour simplifier la manipulation des fichiers depuis un contexte local.

4. Elément d'Interaction

Stockage dans des fichiers

Ouverture d'un fichier en écriture :

```
private static final String NOM_FICHIER = "MonFichier.dat";
...
try {
    // Crée un flux de sortie vers un fichier local.
    FileOutputStream ous = openFileOutput(NOM_FICHIER, MODE_PRIVATE);
    ...
} catch (FileNotFoundException e) {
    // Traitement des exceptions ...
}
```

Ouverture d'un fichier en lecture :

```
private static final String NOM_FICHIER = "MonFichier.dat";
...
try {
    // Crée un flux d'entrée depuis un fichier local.
    FileInputStream ins = openFileInput(NOM_FICHIER);
    ...
} catch (FileNotFoundException e) {
    // Traitement des exceptions ...
}
```

- Utilisez write() pour écrire dans le fichier

- Fermez le flux de sortie par close().

Ex ous.write

(TEXT.getbyt())

- Utilisez read() pour lire les bytes à partir du fichier

- Fermez le flux d'entrer par close().

4. Elément d'Interaction

Stockage dans des fichiers Externes

- Effectuer des opérations de lecture et écriture à partir d'un média externe (carte mémoire SD ..) pour effectuer des opérations I/O
- Ajouter la permission dans le manifest

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

- il faudra vérifier que l'unité externe est bien disponible(Montée)

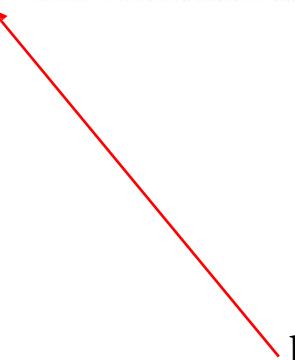
```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```

4. Elément d'Interaction

Stockage dans des fichiers Externes

- Il faut tester le mode qui a été autorisé

```
/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```



l'écriture a été autorisée

4. Elément d'Interaction

Stockage dans des fichiers Externes

- Préciser le chemin à joindre au nom du fichier,

```
File root = Environment.getExternalStorageDirectory();
File directory = new File (root.getAbsolutePath() + "/Data");
directory.mkdirs();
File myFile = new File (directory, "textfile.txt");
```

Le fichier «textfile.txt» va être disponible dans le répertoire «/sdcard/Data»

Meme methodes d'ecriture et de lecture utilisés pour les fichiers internes

4. Elément d'Interaction

Stockage dans une base de données SQLite

- Android fournit un support de bases de données relationnelles au travers de SQLite, une base de données très utilisée (vu sa légèreté 2 MO) dans le domaine des appareils mobiles, lecteurs mp3, navigateurs, etc.
- A la différence d'autres bases de données, SQLite s'exécute sans nécessiter de serveur
 - l'exécution des requêtes sur la base de données s'effectue dans le même processus que l'application

Chemin BD : DATA/data/APP_NAME/databases/Filename.DATABASE



On peut gérer une petite quantité de données

4. Elément d'Interaction

SQLite

- SQLiteDatabase est la classe de base pour travailler avec une base de données SQLite sous Android
 - fournit des méthodes pour ouvrir, effectuer des requêtes, mettre à jour et fermer la base de données



La méthode SQLiteOpenHelper est recommandée pour créer et gérer une BD SQLite

4. Elément d'Interaction

SQLite: SQLiteOpenHelper

- Permet de créer et de manipuler la base de données de manière simple
- Étapes à suivre:
 - Créer une classe qui hérite de SQLiteOpenHelper (qui permet la gestion de la création et des versions de la bd)
 - Créer la base de données et les tables nécessaires
 - Implémenter les méthodes suivantes
 - Le constructeur
 - onCreate: contient les opérations réalisées à la création de la base de données
 - onUpgrade: opérations réalisées quand la base fait un upgrade

4. Elément d'Interaction

SQLite: SQLiteOpenHelper

- Pour que votre SQLiteOpenHelper reste lisible
 - Créer une classe par table. Cette classe définit les méthodes de gestion de la table (insertion, suppression...)

4. Elément d'Interaction

SQLite

```
public class MaBaseSQLite extends SQLiteOpenHelper {

    private static final String TABLE_LOGIN = "table_login";
    private static final String COL_ID = "ID";
    private static final String COL_login = "Login";
    private static final String COL_pass = "Pass";

    private static final String CREATE_BDD = "CREATE TABLE " + TABLE_LOGIN + " (" +
        COL_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " + COL_login + " TEXT NOT NULL, " +
        COL_pass + " TEXT NOT NULL);";

    public MaBaseSQLite(Context context, String name, CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //on crée la table à partir de la requête écrite dans la variable CREATE_BDD
        db.execSQL(CREATE_BDD);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

        db.execSQL("DROP TABLE " + TABLE_LOGIN + ";");
        onCreate(db);
    }
}
```

4. Elément d'Interaction

SQLite

- Pour réaliser des écritures ou lectures, on utilise les méthodes **getWritableDatabase()** et **getReadableDatabase()** qui renvoient une instance de **SQLiteDatabase**. Sur cet objet, une requête peut être exécutée au travers de la méthode **query()**
 - Ex : `bd = maBaseSQLite.getWritableDatabase();`
- Les curseurs sont des objets qui contiennent les résultats d'une recherche dans une base de données(via query).

4. Elément d'Interaction

SQLite

```
package com.example.mabik.tp3;

import ...

public class LoginBDD {

    private static final int VERSION_BDD = 1;
    private static final String NOM_BDD = "login.db";
    private static final String TABLE_LOGIN = "table_login";
    private static final String COL_ID = "ID";
    private static final int NUM_COL_ID = 0;
    private static final String COL_LOGIN = "login";
    private static final int NUM_COL_LOGIN = 1;
    private static final String COL_PASS = "Pass";
    private static final int NUM_COL_PASS = 2;

    private SQLiteDatabase bdd;
    private MaBaseSQLite maBaseSQLite;

    public LoginBDD(Context context){
        //création de la BDD et la table
        maBaseSQLite = new MaBaseSQLite(context, NOM_BDD, null, VERSION_BDD);
    }

    public void open(){
        //ouvrir la BDD en écriture
        bdd = maBaseSQLite.getWritableDatabase();
    }

    public void close(){
        //fermeture de l'accès à la BDD
        bdd.close();
    }

    public SQLiteDatabase getBDD() { return bdd; }
}
```

4. Elément d'Interaction

SQLite

Pour insérer un élément dans une table de la base:

- Utiliser l'objet ContentValues, qui va contenir, pour chaque enregistrement, les valeurs des différentes colonnes.

ex :

```
ContentValues values = new ContentValues();
values.put(COL_LOGIN, login.getLogin());
values.put(COL_PASS, login.getPass());
```

- l'insérer dans la base en utilisant un objet (db) de type SQLiteDatabase

```
db.insert(TABLE_LOGIN, null, values);
```

- ❖ SQLite ne permet pas d'insérer une ligne complètement vide sans donner au moins le nom d'une colonne (values est Null).
- ❖ le paramètre nullColumnHack fournit le nom de la colonne Nullable pour insérer explicitement un NULL dans le cas où vos valeurs sont vides.

4. Elément d'Interaction

SQLite

```
public Login getLoginWithlogin(String log, String Pass) {  
    String[] selectionArgs = {log, Pass};  
    String whereClause = COL_LOGIN + "=? " + " and " + COL_PASS + "=? ";  
    //Récupérer dans un Cursor les valeurs correspondantes à un login)  
    Cursor c = bdd.query(TABLE_LOGIN, new String[] {COL_ID, COL_LOGIN, COL_PASS}, whereClause , selectionArgs, groupBy: null, having: null, orderBy: null);  
    return cursorToLogin(c);  
}
```

Projections: Colonnes à afficher

groupBy, having, orderBy, limit

- L'objet de type **Cursor** permet de traiter la réponse (en lecture ou écriture), par exemple:

- **getCount()**: nombre de lignes de la réponse
- **moveToFirst()**: déplace le curseur de réponse à la première ligne
- **moveToNext()**: avance à la ligne suivante
- **getColumnName(int)**: donne le nom de la colonne désignée par l'index
- ...

[https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#query\(boolean, java.lang.String, java.lang.String\[\], java.lang.String, java.lang.String\[\], java.lang.String, java.lang.String, java.lang.String, java.lang.String\)](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#query(boolean, java.lang.String, java.lang.String[], java.lang.String, java.lang.String[], java.lang.String, java.lang.String, java.lang.String, java.lang.String))

4. Elément d'Interaction

ORM : Room

- Les requêtes SQLite sont écrites dans des variables statiques
 - Aucune vérification des requêtes au niveau de la compilation.
- Besoin de plusieurs code pour avoir un objet Java
 - D'où l'apparition des ORM (Object relation Mapping)
 - Gestion de la BD relationnelle SQLite avec des objet Java.
- Room est un ORM développé par google

4. Elément d'Interaction

Room: Etapes

- Définir les entités
 - Les classes avec l'annotation `@Entity` pour les définir comme des tables de votre BD : `@Entity(tableName = "xxxx")`
 - Chaque class doit avoir une `@PrimaryKey`
- Définir le Design Pattern DAO (data access object) pour manipuler les entités par la réalisation des actions CRUD (ajouter, Lire, mettre à jour et supprimer)
 - Pour chaque entité on crée son interface Dao qui permet de manipuler les données de cette classe « Table »
- Définir une classe qui permet de **lier** toutes les classes/interfaces créées et **configurer** la base de données
 - C'est une classe abstraite qui étends RoomDatabase avec l'annotation `@Database` associée à deux infos : les entités (tables) et la version de BD
 - Contient des abstract méthodes qui Retournent les DAO class
- Le Room.databaseBuilder crée pour la première fois un nouveau objet RoomDatabase et crée aussi le fichier de la base Sqlite. Après la création il retourne juste la référence de la BD

4. Elément d'Interaction

Exemple

```
@Entity (tableName = "users")
public class User {
    @PrimaryKey
    @NonNull
    public String login;

    @ColumnInfo(name = "user_pass")
    public String pass;

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPass() {
        return pass;
    }

    public void setPass(String pass) {
        this.pass = pass;
    }
}

@Dao
public interface Usr_dao {

    @Insert(onConflict= OnConflictStrategy.IGNORE)
    public void adduser(User user);

    @Query("select * from users where login= :login")
    public List<User> getUser(String login);

    @Query("select * from users")
    public List<User> getUsers();

    @Delete
    public void deletuser(User user);

    @Update
    public void updatuser(User user);
}

@Database(entities = {User.class }, version = 1)
public abstract class MyDatabase extends RoomDatabase {

    public abstract Usr_dao mydao();

}
```

4. Elément d'Interaction

- Après dans votre activité :

```
static MyDatabase mydatabase;  
  
mydatabase =  
Room.databaseBuilder(getApplicationContext(),MyDatabase.class,  
"user_bd").allowMainThreadQueries().build();
```

- `allowMainThreadQueries()`: permettre à Room de fonctionner dans le thread UI

4. Elément d'Interaction

Partager des données

- L'accès à une base de données n'est possible que depuis l'application à partir de laquelle elle a été créée.
- Pour exposer les données d'une application à d'autres applications (ex des photos prises par votre application, Carnet d'adresses...)
 - **fournisseur de contenu**, sous la forme d'une interface générique permettant d'exposer les données d'une application en lecture et/ou en écriture
 - `ContentResolver()` permet d'accéder au fournisseur de contenu
 - Usage de Curseur pour récupérer les données
`Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)`

4. Elément d'Interaction

Thread

- Quand une application ne réagisse pas rapidement avec les actions utilisateurs, ce dernier détecte le ralentissement s'il dure plus de 100 ms, le gestionnaire d'activité « ActivityManager » peut tuer le service au-delà de 5 seconde d'attente le système (Android affiche une boîte de dialogue **L'application ne répond pas**).
 - Grand travail à faire en arrière-plan
 - Service
 - Thread

<https://openclassrooms.com/fr/courses/2023346-creez-des-applications-pour-android/2027457-le-travail-en-arriere-plan>

Déetecter et diagnostiquer les problèmes:

https://developer.android.com/topic/performance/vitals/anr.html#detect_and_diagnose_problems

4. Elément d'Interaction

Thread

- Quand une activité est lancée, le système crée un thread principal dans lequel s'exécutera l'application appelée aussi Thread UI
- Il faut respecter deux règles dès qu'on manipule des threads :
 - 1. Ne jamais bloquer le thread UI
 - 2. Ne pas manipuler les vues en dehors du thread UI

4. Elément d'Interaction

Thread

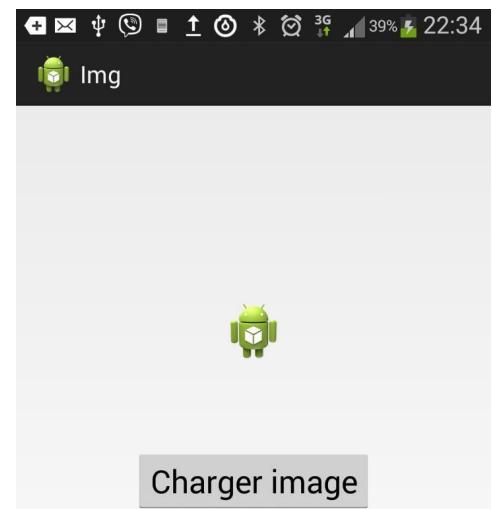
- évitez certaines opérations dans le thread UI, en particulier :
 - Accès à un réseau
 - Certaines opérations dans la base de données (comme les sélections multiples).
 - Les accès fichiers, qui sont des opérations plutôt lentes.
 - les accès matériels
 -

4. Elément d'Interaction

Thread

```
mImageView = (ImageView) findViewById(R.id.imageView);
mbutton = (Button) findViewById(R.id.button);
mbutton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    Log.e("Pb_myThread", e.toString());
                }
                mBitmap = BitmapFactory.decodeResource(getResources(),
                        R.drawable.myimage);
                mImageView.setImageBitmap(mBitmap);

                start(
                });
            });
        });
    }
});
```



4. Elément d'Interaction

Thread

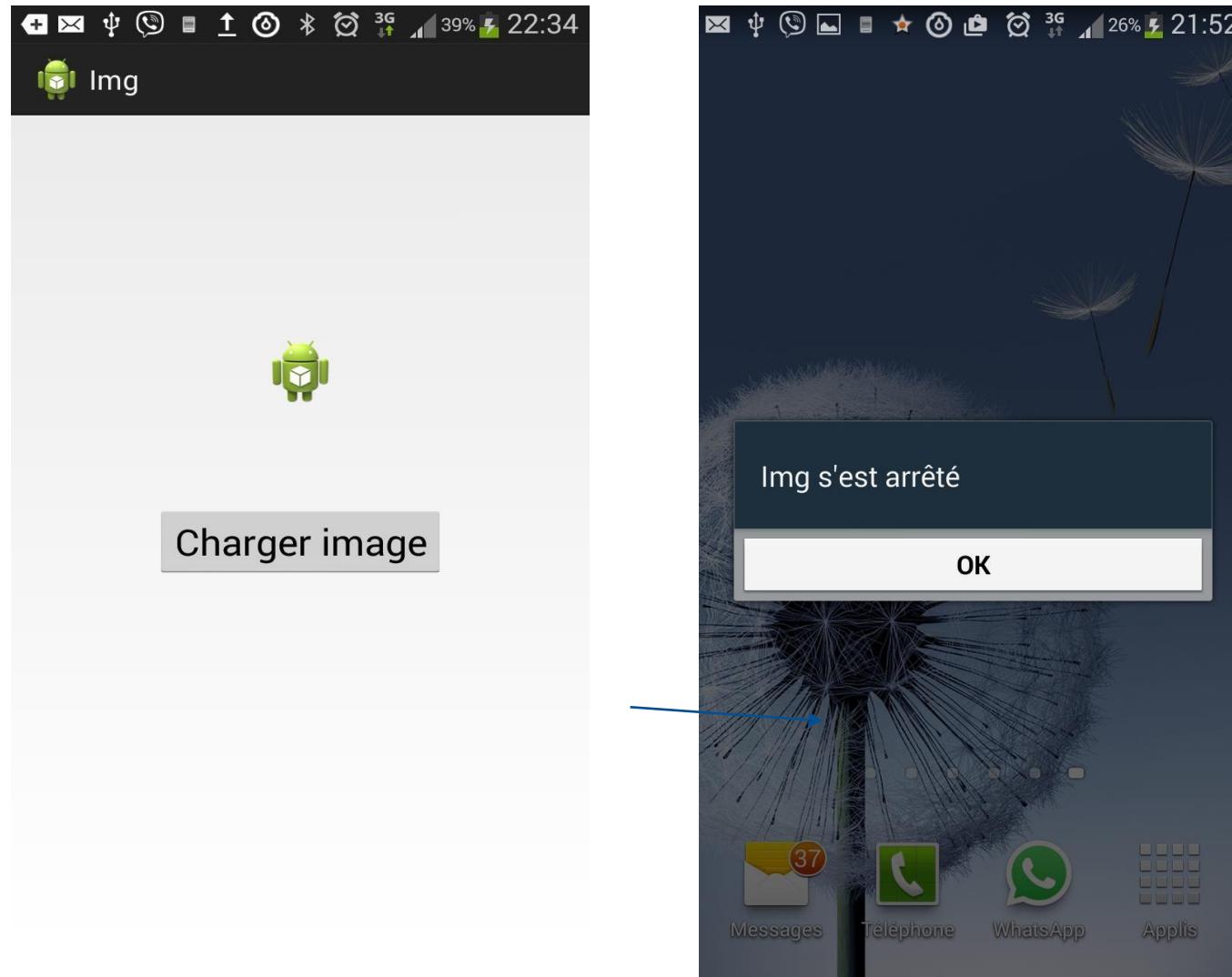
- L'opération coûteuse (chargement de l'image) s'exécute dans un autre thread.
- Pb: la vue est aussi manipulée dans ce thread (#du thread UI)



Non respect de la règle 2

4. Elément d'Interaction

Thread



`android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.`

4. Elément d'Interaction

Thread

- Pour remédier au problème, Android offre plusieurs moyens pour accéder au thread UI depuis d'autres threads

4. Elément d'Interaction

Thread

```
runOnUiThread(new Runnable() {
    @Override
    public void run() {
        mImageView.setImageBitmap(mBitmap);
    }
}); |
```

spécifie qu'une action doit s'exécuter dans le thread UI.
Si le thread actuel est le thread UI, alors l'action est exécutée immédiatement.
Sinon, l'action est ajoutée à la pile des évènements du thread

```
mImageView.post(new Runnable() {
    @Override
    public void run() {
        mImageView.setImageBitmap(mBitmap);
    }
}); |
```

Via le view envoyer des Runnable à l'UI thread (pile des messages) grâce à la méthode **post**.
postDelay(runnable,time) : Ajout dans la pile après qu'une certaine durée time s'est écoulée

```
Handler handler = new Handler(Looper.getMainLooper());
handler.post(new Runnable() {
    public void run() {
        // UI code goes here
        mImageView.setImageBitmap(mBitmap);
    }
}); |
```

- Création d'un thread
- Le handler possède des mécanismes d'ajout des messges (ou Runnable) à la file de message
- Le thread UI possède lui aussi un handler
- Chaque handler créé communiquera par défaut avec le handler de l'UI

4. Elément d'Interaction

Thread

- La class `AsyncTask` : Elle permet de nous abstraire des "threads" et des Runnable à déposer dans l'UI Thread. Elle permet d'exécuter des instructions en arrière-plan et de se synchroniser à nouveau avec le thread principal.
- Idéal pour les opérations courtes d'arrière-plan qui doivent mettre à jour l'interface utilisateur (moins de 5ms)
- Si votre application doit gérer les changements de configuration de l'activité (orientation) utilisez `AsyncTaskLoader` qui remplit la même fonction que `AsyncTask`
 - `AsyncTask` n'est pas détruit si l'activité qui l'a créé est détruite (fuite de mémoire)

4. Elément d'Interaction

Thread

AsyncTask est moins performant que le handler qui permet la gestion et la planification de plusieurs messages (ex si on a plusieurs tâches qui se répètent optez pour le handller...).

4. Elément d'Interaction

Thread

- Un défi de l'utilisation des thread est de prendre en compte le cycle de vie de l'application. Le système Android peut tuer l'**activité** ou déclencher un changement de configuration qui la redémarrera.
- Mais le thread n'est pas tué et il pointe toujours vers l'espace mémoire du handler et de l'activité qui ne sont pas détruits (même si l'activité est détruite). Le GarbageCollector ne peut pas donc collecter ses ressources qui sont pour lui utilisés. Lors de création d'une autre activité, un autre thread est lancé...
- En général quand il y a un problème de fuite de mémoire vous aurez : `java.lang.OutOfMemoryError` indiquant que l'espace mémoire alloué à votre application n'est plus suffisant pour son exécution

4. Elément d'Interaction

Thread

Utilisez ex : <https://developer.android.com/studio/debug/stacktraces>. Ou
<https://firebase.google.com/products/crashlytics> . (<https://firebase.google.com/docs/crashlytics?hl=fr>
<https://firebase.google.com/docs/crashlytics/get-started-new-sdk?platform=android&authuser=2>

Ou <https://developer.android.com/studio/profile/memory-profiler>

- Ex solution
 - si le cycle de vie de l'activité doit être lié à celui du Thread utilisez deux booléens « synchronized » appelés AomicBoolean en plus de ces deux éléments de la méthode
 - On peut ajouter en plus des ces 2 méthodes la méthode onRetainNonConfigurationInstance
 - Si la fin de l'activité ne termine pas le Thread, il est recommandé d'utiliser un service entre l'activité et le Thread.

4. Elément d'Interaction

Thread et Fuites de Mémoire(Memory Leaks)

<https://blog.smartnsoft.com/les-fuites-de-m%C3%A9moire-dans-une-application-android-e4223a6c0b93>

<https://openclassrooms.com/fr/courses/4568576-recuperez-et-affichez-des-donnees-distantes/4893741-trouvez-facilement-les-fuites-de-memoire>

<https://feanorin.developpez.com/tutoriels/android/controler-fuite-memoire/>

4. Elément d'Interaction

Service vs Thread

- Un service est un composant qui peut s'exécuter en arrière-plan, même lorsque l'utilisateur n'interagit pas avec votre application. **Il s'exécute en général dans le Thread UI.** donc si vous voulez mettre à jour l'interface utilisateur utilisez les Thread dans le service.
- Un service a un cycle de vie. Il peut être contrôlé par d'autres applications activités, services...
- Le service est plus prioritaire que le thread
- Un service peut faire communiquer des applications distantes en déclarant une interface qui peut être invoquée à distance.

4. Elément d'Interaction

Service vs Thread

- Si l'utilisateur interagit avec votre application, et s'il y a une tache à effectuer en dehors de UI principal, utilisez un thread.
- Un service n'est pas une alternative aux Threads d'arrière-plan, mais il fournit une autre manière pour exécuter vos Threads.
 - Un service peut être invoqué par une activité, un service, a cycle de vie...
 - si la fin de l'activité ne termine pas la Thread, il est recommandé d'utiliser un service entre l'activité et la Thread.

4. Elément d'Interaction

fonctionnalités Android

Connectivité Réseau

- On peut surveiller la connexion et contrôler l'accès au réseau
 - ConnectivityManager : informations sur sa disponibilité de manière générale
 - NetworkInfo: informations sur l'une des interfaces réseau (le réseau mobile 3G,4G..ou le WiFi)

Ajouter dans le Manifest la permission:

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

Pour réagir au changement de l'état du réseau ajoutez un receiver :

```
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
```

4. Elément d'Interaction

Accès Internet

- Ajouter dans le fichier « AndroidManifest.xml » les autorisations nécessaires pour permettre l'accès à l'internet, comme suit :

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

Les étapes :

- Créer l'URL: URL url = new URL(str_url)
- Ouvrir la connexion : conn = HttpURLConnection(url.openConnection())
- Définir le requête (POST, GET,...): conn.setRequestMethod("GET");
- Si c'est GET ouvrir le flux d'entrée et lire les données de réponse du serveur
InputStream in = new BufferedInputStream(connection.getInputStream())
BufferedReader reader = new BufferedReader(new InputStreamReader(in));
String line, str="";
while ((line = reader.readLine()) != null) {
 str= str + line + "\n" ;
}
• Si c'est POST, ouvrir le flux de sortie et y écrire les données en utilisant le OutputStream, getOutputStream
- Après avoir utilisé HttpURLConnection, n'oubliez pas de fermer votre connexion:
conn.disconnect();

4. Elément d'Interaction

Géolocalisation google maps et GPS

- Les services de géolocalisation d'Android sont divisés en deux grandes parties
 - les API qui gèrent les plans (dans l'espace de noms com.google.android.maps)
-> c'est la plus utilisée
 - gestion automatique des fournisseurs de localisation, les déplacements des utilisateurs et la précision de la localisation. Elle gère également la planification des mises à jour d'emplacement en fonction des paramètres de consommation d'énergie fournis
 - les API qui gèrent la localisation à proprement parler (l'espace de noms android.location)
 - Pour déterminer la position courante on distingue deux types de fournisseurs :
 - le fournisseur basé sur la technologie GPS de type LocationManager.GPS_PROVIDER
 - le fournisseur qui se repère grâce aux antennes des opérateurs mobiles et aux points d'accès Wi-Fi, de type LocationManager.NETWORK_PROVIDER

4. Elément d'Interaction

Géolocalisation google maps et GPS

- Les API Google
 - L'utilisation des cartes Google Maps et de certaines fonctionnalités comme le géo(dé)codage (déterminer des coordonnées en latitude et longitude à partir d'une adresse ou l'inverse)
 - Il faut obtenir une clé pour utiliser Google Maps (cf TP)

4. Elément d'Interaction

Capteurs

- La plupart des appareils fonctionnant sous Android ont des capteurs intégrés qui mesurent le mouvement, l'orientation et diverses conditions environnementales.
- Ces capteurs sont capables de fournir des données brutes avec une grande précision et sont utiles si vous souhaitez surveiller le mouvement ou le positionnement d'un appareil en trois dimensions, ou si vous souhaitez surveiller les changements dans l'environnement ambiant à proximité d'un appareil.

4. Elément d'Interaction

Capteurs

- Les capteurs de mouvements : en mesurant les forces d'accélération et de rotation sur les trois axes, ces capteurs sont capables de déterminer dans quelle direction se dirige l'appareil. On y trouve l'accéléromètre, les capteurs de gravité, les gyroscopes et les capteurs de vecteurs de rotation.
- Les capteurs de position : Ils déterminent la position de l'appareil. On trouve ainsi les capteurs d'orientation et le magnétomètre.
- Les capteurs environnementaux : ce sont trois capteurs (baromètre, photomètre et thermomètre) qui mesurent la pression de l'air ambiant, l'éclairage et l'humidité

<https://developer.android.com/guide/topics/sensors/>

4. Elément d'Interaction

Capteurs

Nom du capteur	Valeur système	Type	Description	Utilisation typique
Accéléromètre	TYPE_ACCELEROMETER	Matériel	Mesure la force d'accélération appliquée au terminal sur les trois axes (x, y et z), donc la force de gravitation (m/s^2).	Détecter les mouvements.
Tous les capteurs	TYPE_ALL	Matériel et logiciel	Représente tous les capteurs qui existent.	
Gyroscope	TYPE_GYROSCOPE	Matériel	<i>Mesure le taux de rotation sur chacun des trois axes en radian par seconde (rad/s).</i>	<i>Détecter l'orientation de l'appareil.</i>
Photomètre	TYPE_LIGHT	Matériel	Mesure le niveau de lumière ambiante en lux (lx).	Détecter la luminosité pour adapter celle de l'écran de l'appareil.
Magnétomètre	TYPE_MAGNETIC_FIELD	Matériel	Mesure le champ géomagnétique sur les trois axes en microtesla (μT).	Créer un compas.

4. Elément d'Interaction

Capteurs

Nom du capteur	Valeur système	Type	Description	Utilisation typique
Orientation	TYPE_ORIENTATION	Logiciel	Mesure le degré de rotation que l'appareil effectue sur les trois axes.	Déterminer la position de l'appareil.
Baromètre	TYPE_PRESSURE	Matériel	<i>Mesure la pression ambiante en hectopascal (hPa) ou millibar (mbar).</i>	<i>Surveiller les changements de pression de l'air ambiant.</i>
Capteur de proximité	TYPE_PROXIMITY	Matériel	Mesure la proximité d'un objet en centimètres (cm).	Déetecter si l'utilisateur porte le téléphone à son oreille pendant un appel.
Thermomètre	TYPE_TEMPERATURE	Matériel	Mesure la température de l'appareil en degrés Celsius (°C).	Surveiller la température.

4. Elément d'Interaction

Gérer le Bluetooth

- BluetoothAdapter : permet de récupérer les appareils voisins et de communiquer avec eux.
- BluetoothDevice: représente un appareil Bluetooth distant, et est utilisé pour récupérer les informations sur l'appareil et créer une connexion
- BluetoothSocket et BluetoothServerSocket : les points de connexion permettant aux appareils d'échanger des données entre eux.



connexion point à point & connexion à plusieurs appareils simultanément

À ne pas oublier : `<uses-permission android:name="android.permission.BLUETOOTH" />`

4. Elément d'Interaction

Dessiner en 2D

2 façons pour le faire :

- Dessiner dans une vue → on peut le faire pour des dessins simples avec une mise à jour peu fréquentes
 - Associer un drawable à une vue image qui va l'afficher
- Dessiner dans un canvas (toile): pour les dessin complexe et mis à jour fréquente

4. Elément d'Interaction

DRAWABLE

- Ce qu'on peut dessiner comme :
 - Bitmap
 - Color
 - Shape
 - ...
- Exemple :
 - BitmapDrawable : matrice de pixels
 - ShapeDrawable : cercle, rectangle..
 - ColorDrawable: une couleur unie

4. Elément d'Interaction

DRAWABLE

Pour dessiner sur un canvas on a besoin de 4 composants de base:

- une image BitMap qui est une matrice de pixels dans laquelle on veut dessiner (surface sur laquelle on dessine)
- 1er méthode

```
Bitmap createBitmap(int width, int height, Bitmap.Config config)
```

- Config.RGB_565 : chaque pixel ne puisse être qu'une couleur
- Config.ARGB_8888 : chaque pixel puisse être soit une couleur, soit transparent
- Config.ALPHA_8: pixels transparents

4. Elément d'Interaction

DRAWABLE

- 2ème méthode

```
decodeResource(Resources ressources, int id) : à partir d'un fichier d'image
```

```
BitmapFactory.decodeStream(InputStream) : à partir d'un stream
```

4. Elément d'Interaction

DRAWABLE

- 3ème méthode

Via XML

```
image = ( ImageView ) findViewById ( R . Id . ex_image )
```

- Un objet Canvas qui appelle les méthodes de dessin pour mettre à jour la matrice BitMap
 - Canvas c = new Canvas(b); (b : le BitMap que vous avez créé)
- Une fonction pour dessiner (ligne, texte...) la forme à représenter :
 - DrawText(), DrawPoints(), DrawColor()..
- Un objet de peinture (pour régler les couleurs et styles des opérations que vous effectuez)
 - Paint p = new Paint();
 - p.setStyle(Paint.Style.FILL), p.setColor(myDrawColor)...

4. Elément d'Interaction

Affichage

2 façons pour accéder à Canvas :

- Par View

On crée une classe qui étend View

- création d'une vue qui reçoit l'objet canvas lors de l'appel de la méthode OnDraw.
 - Les dessins seront à effectuer dans la méthode onDraw (Canvas canvas) puis on définit le bitmap : canvas.drawBitmap la forme...
 - Le onDraw est appelé automatiquement mais si on veut le faire on utilise la méthode invalidate()

Le rafraîchissement de l'écran ne se produit pas immédiatement. Il ne se produit qu'après la prochaine arrivée du signal VSYNC généré par le noyau et qui se produit toutes les 16,6 ms (~ 60 images par seconde)

- pb de mise à jour dans le Thread Principal → Non recommandée pour les jeux

- Par SurfaceView

- permet de dessiner dans un thread différent du thread UI
- prise en charge par OpenGL pour les opérations graphiques compliquées

4. Elément d'Interaction

SurfaceView

- SurfaceView .
 - on ne peut pas accéder à cette surface directement on doit utiliser une couche abstraite représentée par la classe SurfaceHolder.
 - SurfaceHolder getHolder()
 - Dans le constructeur : on récupère la ref du holder ex mSurfaceHolder = **getHolder();**
 - On peut gérer le cycle de vie de la surface en implementant la classe de surfaceriew par l'interface SurfaceHolder.Callback et on l'associant au holder via la methode addCallback
 - Methodes abstraites à définir : surfaceCreated, surfaceDestroyed, void surfaceChanged

Etapes :

- Création de la class qui va manipuler le dessin et qui étend SurfaceView et qui s'exécute dans un thread
 - Ajoutez deux méthodes pour réagir aux événements pause et le resume
Quand l'activité est en pause, la surface sur laquelle se fait le dessin sera supprimée. Elle sera recréée une fois l'activité reprend.
 - Dans le constructeur instance votre Holder
 - Dans le run de cette classe Implémenter la zone de dessin (canevas)
 - Vérifiez le disponibilité de la surface : if (mySurfaceHolder.getSurface().isValid())
 - Définir le canevas, et le bloquer pour que l'on puisse dessiner dessus Canvas
c = mySurfaceHolder.lockCanvas();
 - une fois le dessin terminé, vous appelez unlockCanvasAndPost (Canvas canvas) pour débloquer le canevas.
 - **mySurfaceHolder.unlockCanvasAndPost(c);**

4. Elément d'Interaction

Exemple Dessin graphique

<https://google-developer-training.github.io/android-developer-advanced-course-concepts/unit-5-advanced-graphics-and-views/lesson-11-canvas/11-2-c-the-surfaceview-class/11-2-c-the-surfaceview-class.html>

<https://andrologiciels.wordpress.com/astuces-android/graphisme/surfaceviewutilisation/>

<https://openclassrooms.com/fr/courses/4428411-developpez-des-applications-android-connectees/4529356-tp-de-mise-en-pratique-developpez-un-jeu-de-morpion>

4. Elément d'Interaction

Media/Animation

- <https://google-developer-training.github.io/android-developer-advanced-course-practicals/android-developer-advanced-course-practicals.pdf>

4. Elément d'Interaction

Accessibilité

- Selon une étude faite par la banque mondiale plus de 15% de la population mondiale souffre d'un handicap.
- Les personnes en situation de handicap dépendent des applications et services accessibles pour communiquer, apprendre, travailler...

4. Elément d'Interaction

Accessibilité

- Lorsque vous développez des applications en ayant à l'esprit l'accessibilité, vous améliorez l'expérience utilisateur, en particulier pour les utilisateurs en situation de handicap (physique et/ou cognitif).

4. Elément d'Interaction

Accessibilité

- Étiqueter les éléments d'interface utilisateur par des labels significatifs
 - Si la vue est statique on donne la description dans le fichier XML (ex pour une image on ajoute l'attribut android:contentDescription). Si c'est dynamique on l'ajoute dans le code (ex: setContentDescription() pour une image)
 - Assurez vous que chaque label est unique -> TalkBack
- Proposez grandes cibles tactiles
 - **Utile pour les personnes ayant une** déficience motrice ou visuelle, les personnes âgés..

4. Elément d'Interaction

Accessibilité

- Fournir un contraste de couleur adéquat
 - En augmentant les taux de contraste entre les couleurs de premier plan et d'arrière-plan de votre application, vous permettez aux utilisateurs d'utiliser facilement votre application
- Utilisez des indices autres que la couleur
 - Mettre en plus de la couleur du texte, des formes.. Pour aider les personnes avec déficience visuelle
- Rendre les médias accessibles
 - Ajouter des contrôles de pause d'arrêt, de volume, et sous titre...

4. Elément d'Interaction

Accessibilité

- Pour plus d'info :
 - <https://developer.android.com/guide/topics/ui/accessibility/>
 - + ressources sur la plate-forme

4. Elément d'Interaction

Accessibilité

Pour vos développements, gardez à l'esprit que les périphériques mobiles ont souvent :

- Une puissance processeur plus faible
- Une RAM limitée
- Des capacités de stockage permanent limitées
- De petits écrans avec de faibles résolutions
- Des coûts élevés de transfert de données
- Des taux de transfert plus lents avec une latence élevée
- Des connexions réseau moins fiables
- Des batteries à autonomie limitée

4. Elément d'Interaction

Accessibilité

- Tester votre application
 - <https://developer.android.com/training/testing/index.html>
- Publier une application
 - <https://developer.android.com/studio/publish/index.html>
 - <https://openclassrooms.com/courses/creez-des-applications-pour-android/publier-et-rentabiliser-une-application>

RESSOURCES

<https://google-developer-training.github.io/android-developer-fundamentals-course-concepts/en/>

-<https://developer.android.com/courses/fundamentals-training/overview-v2>

- <https://corp.beapp.fr/expertises/questions-frequentes/difference-entre-web-app-application-native-hybride>
- <https://developer.android.com/training/testing/index.html>

-Livre : Programmation Android. Damien Guignard, Julien Chable, Emmanuel Robel

<https://openclassrooms.com/courses/creez-des-applications-pour-android/>

<https://codelabs.developers.google.com/>

<http://grail.cba.csuohio.edu/~matos/notes/cis-493/lecture-notes/slides/Android- Chapter12->

[Files%20and%20Preferences.pdf](#)

<https://developer.android.com/reference/android/view/SurfaceHolder>