# Développement d'applications mobiles

## Introduction à la conception d’une application mobile

### Les systèmes d’exploitation mobiles

Le SE est une plate-forme à bas niveau *q*ui contrôle toutes les fonctionnalités d’un appareil mobile; Il contrôle toutes les opérations de base du téléphone mobile comme option d’écran tactile, cellulaires, Bluetooth, Wifi, appareil photo, lecteur de musique…

### Les différents systèmes d’exploitation mobiles dominants sur le marché :

**Android OS :**

****

Android est un OS gratuit et complètement ouvert (sous licence open source).

C’est une variante de Linux qui fonctionne sur plusieurs appareils tels que les smartphones, tablettes, montres, téléviseurs et voitures.

Il a été développé par Google.

**L’App Store officiel des applications Android est le Google Play Store initialement Android Market.**

**iOS : Iphone OS**

 Il se trouve non seulement sur les différentes générations de iPhone mais également sur d’autres produits de Apple iPad et iPod touch.

Il est dérivé de Mac OS X dont il partage les fondations : kernel, les services Unix.

La société Apple ne délivre pas de licence du système d'exploitation pour le matériel tiers

**L’App Store officiel des applications iOS est : AppStore**

**Windows Phone:**

 C’est un système d'exploitation mobile développé par Microsoft pour succéder à Windows Mobile, C’est une évolution de Windows Pocket PC, ancêtre de Windows CE.

Windows Phone est lancé le 21 octobre 2010 en Europe, parmi ses versions :

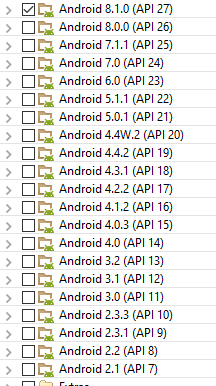
Windows Phone 7

Windows Phone 7.8

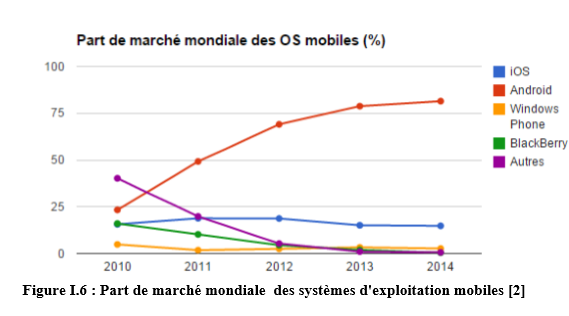
Windows Phone 8 « Apollo ».

L’App Store officiel des applications Windows phone est **Windows store**.

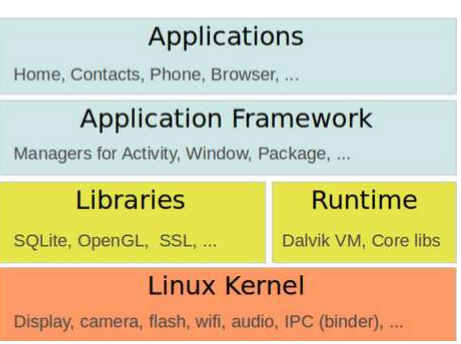
### Les versions de l’Android OS :



### Comparaison entres les systèmes d’exploitation



### Architecture d’un système d’exploitation Android :



1. **Linux Kernel:**

Android s’appuie sur le noyau Linux 2.6 pour les services système de base tels que la sécurité, la gestion de la mémoire et des processus, le réseau et la gestion des drivers. Le noyau sert de couche d’abstraction entre le matériel et le reste de la pile logicielle.

1. **Librairies:**

Des bibliothèques logicielles telles que: WebKit/Blink, OpenGL ES,  SQLite  ou FreeType ; ces bibliothèques permettent d'exécuter des programmes prévus pour la plate-forme Java ;

1. **Runtime :**

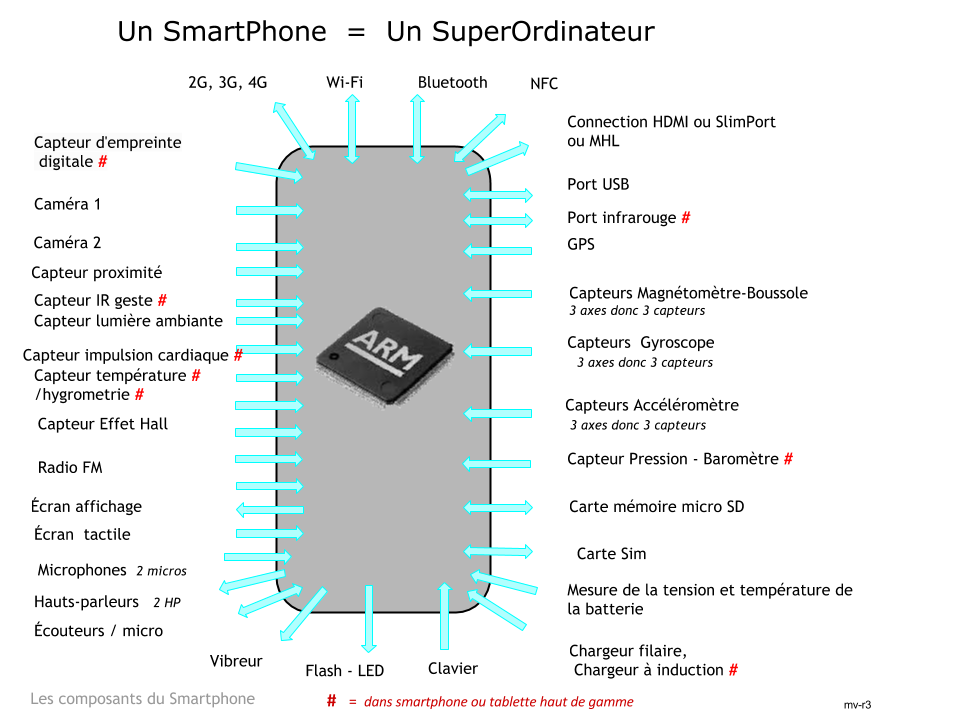
C’est une instance de la machine virtuelle Java, dans laquelle l’application Android s’exécute

1. **Le Framework d’application :**

C’est un ensemble de services et systèmes sur lesquels se reposent les applications Android pour leurs fonctionnements:

* + «Views»
  + «Content Providers»
  + «Ressource Manager»
  + «Activity Manager »

### Les organes d'un téléphone interagissant avec le microprocesseur.



### Les parties sensibles du téléphone :



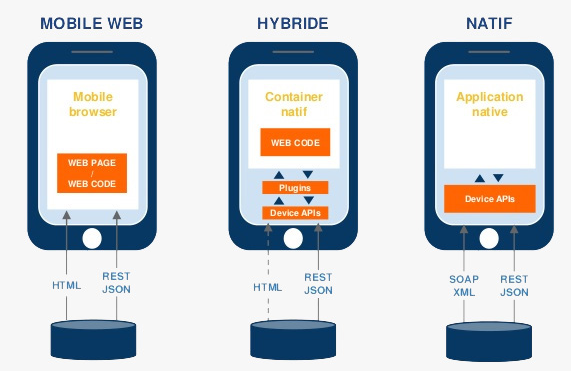
## Les applications mobiles :

### Définition

Une application mobile est un logiciel applicatif développé pour être installé sur un appareil électronique mobile, comme un Smartphone, une tablette ou un baladeur numérique.

Une application mobile peut être soit installée directement sur l’appareil dès sa fabrication en usine, soit téléchargée depuis un magasin d’applications dit «application store» telle que Play Store, l'App Store ou encore le Windows Phone Store ou télécharger à partir d’un serveur à distant.

### Les types d’application :



#### Applications natives (= = Windows Application en VS) :

Les applications natives sont des logiciels conçus spécifiquement pour une plate-forme mobile, en utilisant le SDK propre à celle-ci. Les applications ainsi crées sont ensuite téléchargeables depuis une plateforme dédiée au système, généralement un magasin d’application type App Store d’Apple ou google play de Android.

Les langages utilisés sont:

* + iOS : ObjectiveC+ Xcode
  + Android : Java + ~~Eclipse /~~ Android Studio/ C#Visual Studio
  + Windows : C# + Visual Studio(FrameWork xamarin.Forms)
  + Python : Python.

Exemple : Application de Gestion de location de voitures.



#### Applications Hybrides :



Il s’agit d’une application mobile qui fusionne entre les caractéristiques de web application (développement en HTML5 CSS3 et JavaScript) et celles de l’application native. De cette manière, l’application mobile sera accessible sur toutes les plateformes d’application.

Ce type d’application mobile minimise les charges et la durée de son développement même si cela sera au détriment de perfectionnement et de la qualité qui caractérise l’application native.

#### Web Application

Se sont les applications web basées sur les langages (web HTML5 CSS3 et JavaScript….) qui sont lancée par des navigateurs.

### Déploiement d’une application.



### Création d’une application Mobile Native :

#### Préparation de l’environnement :

1. **Installer Le JDK (Java développement Kit)**
2. **Télécharger et installer Le Genymotion avec Virtual Box (compatibilités).**
3. **Télécharger et Installer Android Studio.**

#### Les Emulateurs.

L'Émulateur Genymotion pour Android est une application de bureau qui émule un appareil Android (machine virtuelle). Il fournit un environnement virtualisé dans lequel vous pouvez déboguer et tester des applications Android sans appareil physique.

Exemple 1:

1. Démarrer votre Genymotion.
2. Installer une version d’Android de votre Emulateur.
3. Démarrer l’API de votre choix.

**Remarque :**

**Pour bien exécuter une application Mobile, il faut que sa version doit antérieur à celle de l’émulateur (Prise en charge).**

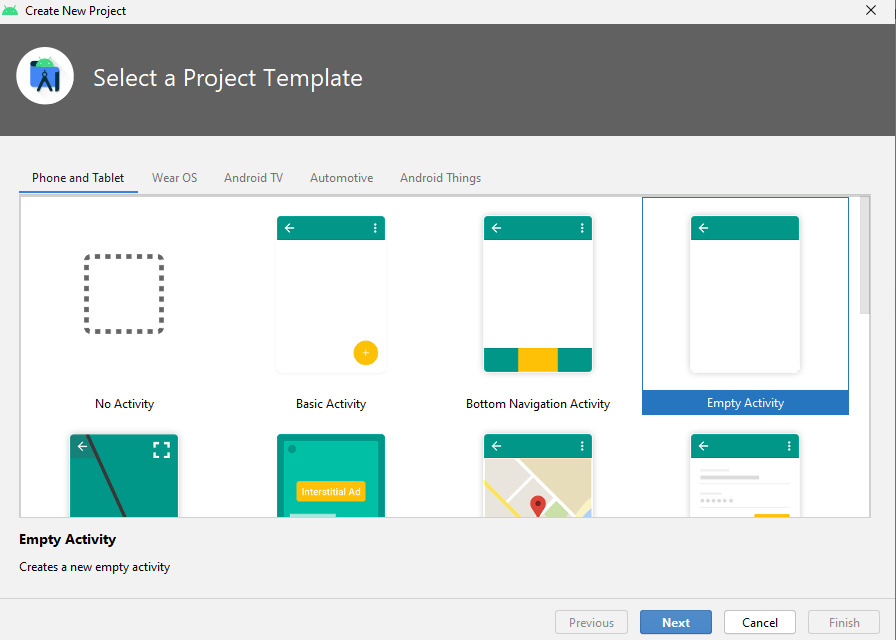
#### Utilisation du devise ou l’appareil physique :

1. Télécharger le pilote Mobile de Votre device.
2. Brancher votre device et installer son pilote.
3. Activer le mode dévéloppeur de Votre device.
4. Activer le débogage USB de votre téléphone.

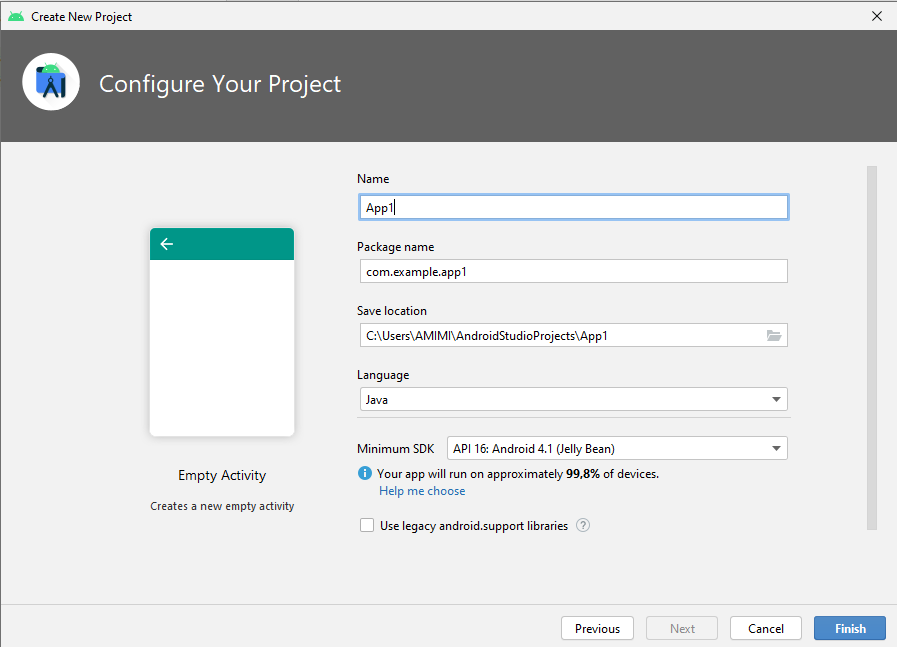
**Votre appareil physique sera chargée dans la liste des emulateurs de android studio.**

#### Création d’une application native.

* + - 1. Démarrer votre Emulateur.
      2. Démarrer Votre android studio et choisir un projet Vide.

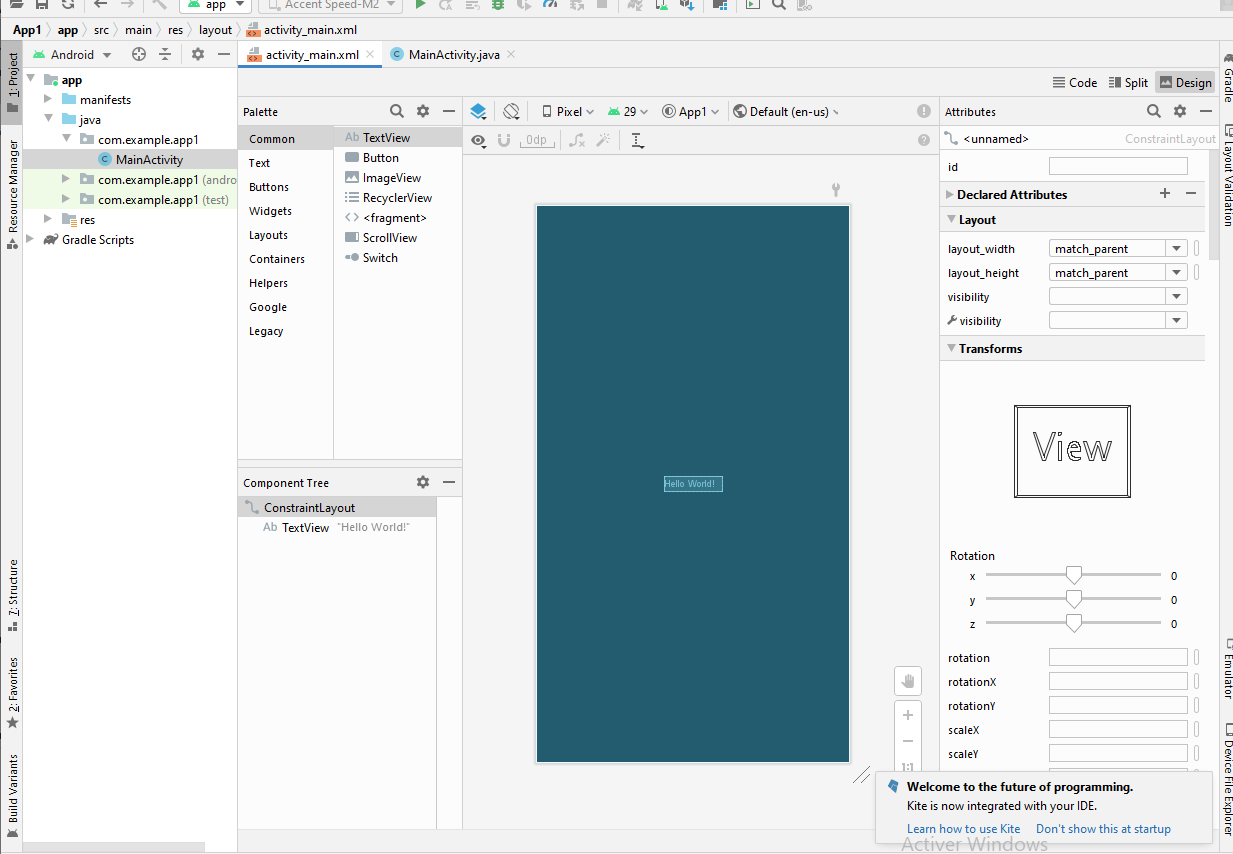


Cliquer sur next et Dans la boite ouverte saisir ou sélectionner les informations convenables.



* Le nom du projet .
* Le nom du package (unique) utilisé par google Play.
* L’emplacement d’enregistrement du projet.
* Le langage utilisé (ici le JAVA).
* La version de l’API de l’application (Attention).

Après le click de fin on aura l’interface suivante :



#### La structure d’un projet sous android studio.

Le volet « project » contient la structure d’une application Native ; elle constituée par :

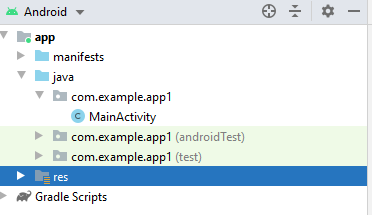
* **Android manifest :**

C’est un fichier xml qui est réservé pour définir les options de votre application Android comme le code de version ou le nom d'affichage de votre application, la liste des permissions requises par votre application, etc.

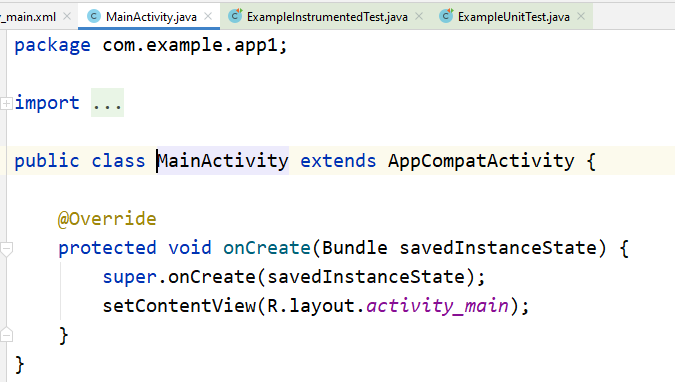
***<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.myapplication">  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.MyApplication">  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>**

Ce fichier xml définit :

* Les permissions,
* L’icône de l’application,
* Le nom de l’application,
* Les composants (Activité, service, BroadcastReceiver, ContenProvider) de votre application,……
* **Le répertoire Java:**



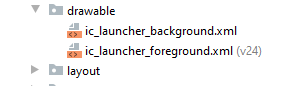
Ce dossier contient les activités crées par le développeur ; se sont tous les fichiers de code source de votre application.



* **Le répertoire res :**

**C’est le dossier des ressources de l’application ;**

* + Les fichiers drawable : Ce répertoire contient les images



* + **mipmap**:Ce répertoire contient plusieurs icons utilisées dans l’application tel que l’icon de l’application et résolution différentes.
  + **Layout :** Ce dossier contient les fichiers xml, représentants les interfaces graphique de l’application.
  + **Values : contient des fichiers sources des valeurs utilisées par l’application tels que :**
  + **colors.xml**: Ce fichier contient les couleurs utilisées dans votre application.
  + **strings.xml**: Ce fichier contient les chaines de caractères utilisées dans votre application
  + **themes.xml**: Ce fichier contient tous les themes de l’application

**(On va revenir sur ces fichiers).**

#### Le fichier activity\_main.xml.

Le fichier layout/activity\_main.xml est un fichier XML qui contient les balises représentant les composants d’une interface graphique d’une application ; il permet de modifier directement les propriétés ou les attributs des composants de l’interface ; ce fichier est lié à un fichier (\*.java) appelé activité (Main\_activity.java) qui contient le code suivant :

**package com.example.app1;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.view.View;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 }  
  
  
 public void Affiche(View view) {  
 }  
}**

Ce fichier contient :

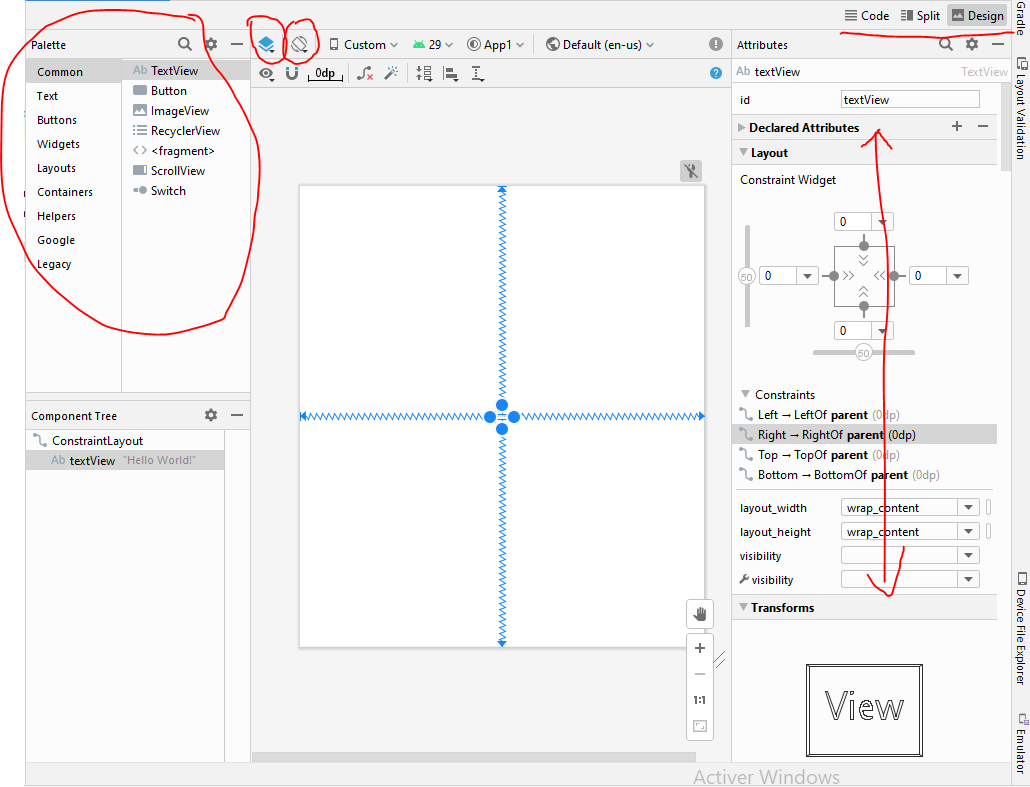
* Le nom du package : **com.example.app1 utiliser dans le Google Play.**
* Les bibliothèques implémentées dans le fichier pour utiliser les fonctions et les attributs en relation.
* La classe dérivée **MainActivity** de la classe **AppCompatActivity ; cette classe contient un classe dérivée onCreate(Bundle savedInstanceState)qui permet de charger l’interface en utilisant la méthode**

**setContentView(R.layout.*activity\_main*);**

Contenant le nom du fichier xml à charger (dans notre cas activity\_main.xml).

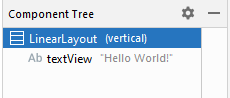
Dans le fichier **MainActivity.java on ajoute le code source de l’application.**

#### Environnement d’android Studio :



#### Structure d’un fichier activity\_main.xml (layout):

Changer la racine de ce fichier en utilisant linearLayout (vertical).

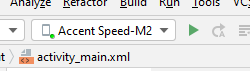


Le code xml pour cette layout est :

***<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World!" />  
</LinearLayout>**

#### Lancement d’une application :

Sur la barre d’outils ; sélectionner l’émulateur soit sur genymotion ou votre device qui cliquer sur la flèche verte :



### Les objets de bases d’une application mobile :

#### Définition de Layout :

Une layout est l’interface graphique représentée par un fichier xml qui est constitué par une Racine et l’empilement des balises représentant les widgets.

#### ConstraintLayout :

C’est un conteneur dans lequel on peut placer des composants (widgets) les un par rapports aux autres ( à droite, à gauche, au-dessus , au-dessous …..).

Parmi les propriétés de cette layout :

* **Layout\_width :** c’est la largeur du conteneur qui aura les valeurs :

**Match\_parent** : la largeur maximale de son parent.

**Wrap\_content** : Ajustement selon son contenu.

Ou une valeur comme (72dp)

* **Layout\_height :** c’est la largeur du conteneur qui aura les valeurs :

**Match\_parent** : la largeur maximale de son parent.

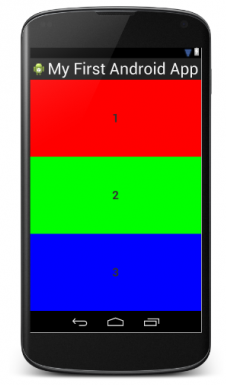
**Wrap\_content** : Ajustement selon son contenu.

Ou une valeur comme (250dp)

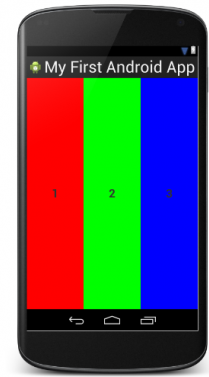
#### Les linearLayouts :

C’est un conteneur qui permet de disposer les composants widgets en sens vertical (android:orientation="vertical") ou en sens horizontal (android:orientation="horizontal").

**Exemple 1 :** android:orientation="vertical")



**Exemple 2 :(** android:orientation="horizontal")



#### Les widgets :

* + **TextView ou label :**

Parmi les propriétés de ce widget :

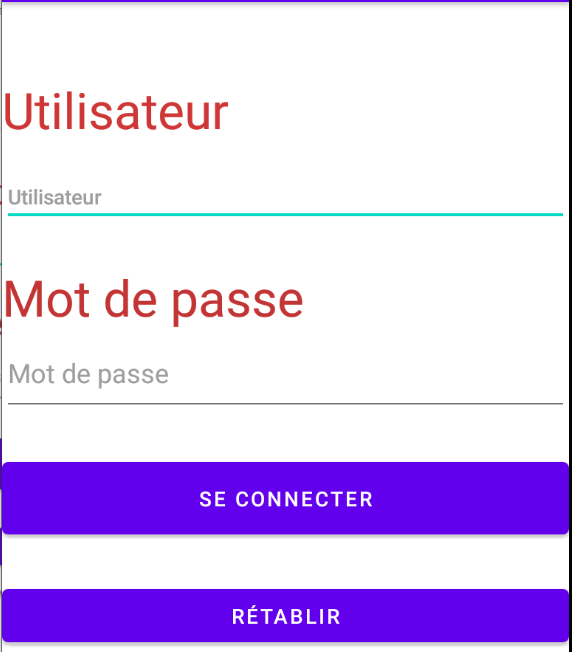
* android:id="@+id/textView"
* android:layout\_width="match\_parent"
* android:layout\_height="72dp"
* android:text="Hello World!"

1. **EditText ou Zone de texte :**

Parmi les propriétés de ce widget :

* android:id="@+id/editTextTextPersonName"
* android:layout\_width="match\_parent"
* android:layout\_height="wrap\_content"
* android:layout\_marginTop="20dp"
* android:ems="10"
* android:hint="Utilisateur"
* android:inputType="textPersonName"
* android:textAppearance="@style/TextAppearance.AppCompat.Body2" />

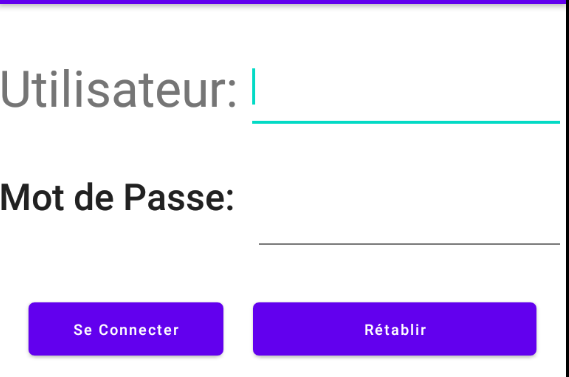
**Exemple 1: Utilisation de LinearLayout verticale**



Le code xml est le suivant :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="42dp"  
 android:layout\_marginTop="50dp"  
 android:text="Utilisateur"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1"  
 android:textColor="#CF3737" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20dp"  
 android:ems="10"  
 android:hint="Utilisateur"  
 android:inputType="textPersonName"  
 android:textAppearance="@style/TextAppearance.AppCompat.Body2" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="25dp"  
 android:text="Mot de passe"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1"  
 android:textColor="#C33535" />  
  
 <EditText  
 android:id="@+id/editTextTextPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="58dp"  
 android:ems="10"  
 android:hint="Mot de passe"  
 android:inputType="textPassword" />  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="61dp"  
 android:layout\_marginTop="25dp"  
 android:text="Se connecter"  
 android:textAllCaps="false" />  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="25dp"  
 android:text="Rétablir"  
 android:textAllCaps="false" />  
</LinearLayout>

**Exemple 1: Utilisation de LinearLayout horizontale**



Le code xml est le suivant :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:layout\_marginTop="25dp"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="63dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:gravity="center\_vertical"  
 android:text="Utilisateur:"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="textPersonName" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="58dp"  
 android:layout\_marginTop="25dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="170dp"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="center\_vertical"  
 android:layout\_weight="1"  
 android:autoText="false"  
 android:freezesText="false"  
 android:text="Mot de Passe:"  
 android:textAllCaps="false"  
 android:textAppearance="@style/TextAppearance.AppCompat.Body2"  
 android:textSize="25sp" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="textPersonName" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="25dp"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="83dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_weight="1"  
 android:text="Se Connecter"  
 android:textAllCaps="false"  
 android:textSize="10sp" />  
  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="143dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginRight="20dp"  
 android:layout\_weight="1"  
 android:text="Rétablir"  
 android:textAllCaps="false"  
 android:textSize="10sp" />  
 </LinearLayout>  
</LinearLayout>

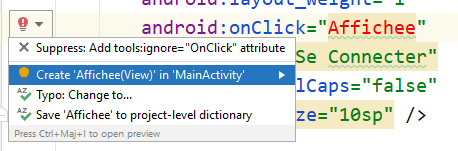
1. **Button :**

Parmi les propriétés de ce widget :

* android:id="@+id/button4"
* android:layout\_width="143dp"
  + - android:layout\_height="wrap\_content"
    - android:layout\_marginStart="20dp"
    - android:layout\_marginLeft="20dp"
    - android:layout\_marginEnd="20dp"
    - android:layout\_marginRight="20dp"
    - android:layout\_weight="1"
    - android:text="Rétablir"
    - android:textAllCaps="false"
    - android:textSize="10sp"

### Associer un événement à un bouton :

Ajouter la ligne : android:onClick="Affichee" à la balise Button et générer la méthode Affiche (à gauche de la ligne ci-dessus) cliquer sur create « Affichee()….. :



La fonction Affichee() s’ajoute au fichier MainActivity.java

public void Affiche(View view) {  
}

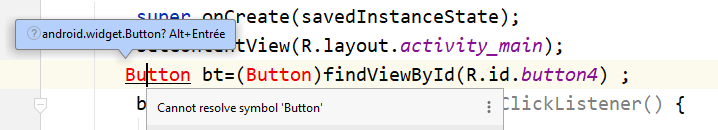
Ou bien dans la fenêtre des propriétés rechercher l’évènement onclick et taper votre fonction Affichee ; et ensuite générer la fonction comme ci-dessus.

Sans ajouter un évènement au fichier xml, on peut créer Aussi un écouteur d’événement dans le fichier MainActivity.java:

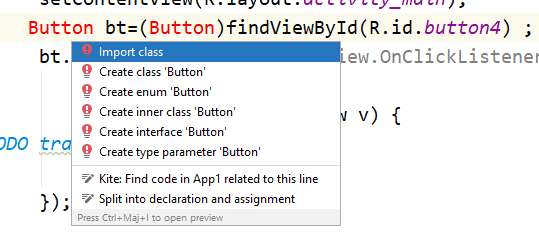
Ajouter la ligne suivante dans la méthode EnCreate().

**Button bt=(Button)findViewById(R.id.*button4*) ;**

Les composants généraux de l’interface ne sont pas pris en charge par android studio comme les autres systèmes ; pour cela il faut ajouter la bibliothèque correspondante.



En cliquant sur alt+entree on affiche le menu suivant :



Cliquer sur import class ; la ligne suivant s’ajoute à votre MainActivity.java.

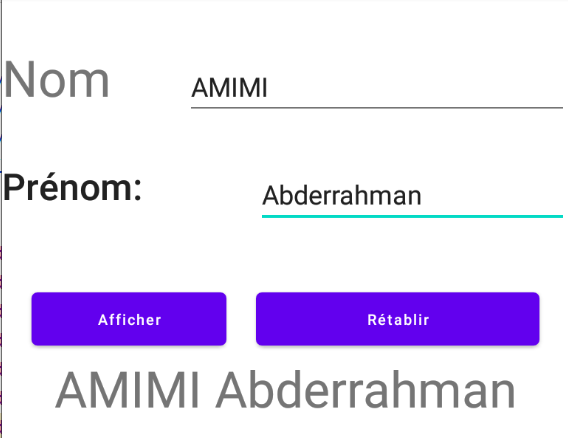
import android.widget.Button;

En fin on écrit le code suivant :

**Button bt=(Button)findViewById(R.id.*button4*) ;  
 bt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *// Ecrire votre code ici* }  
 });**

**Exemple 1:**

1. Concevoir une interface qui permet de lire votre nom et prénom et afficher votre nom complet.
2. Ecrire le code correspondant.
3. Ajouter le bouton rétablir pour rétablir le contenu de l’interface.
4. Réponse :



Le code XML est le suivant :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:layout\_marginTop="25dp"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="63dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:gravity="center\_vertical"  
 android:text="Nom"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1" />  
  
 <EditText  
 android:id="@+id/t1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="text" />  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="58dp"  
 android:layout\_marginTop="25dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="170dp"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="center\_vertical"  
 android:layout\_weight="1"  
 android:text="Prénom:"  
 android:textAllCaps="false"  
 android:textAppearance="@style/TextAppearance.AppCompat.Body2"  
 android:textSize="25sp" />  
  
 <EditText  
 android:id="@+id/t2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="text" />  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="25dp"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/bt"  
 android:layout\_width="83dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_weight="1"  
 android:text="Afficher"  
 android:textAllCaps="false"  
 android:textSize="10sp" />  
  
 <Button  
 android:id="@+id/rst"  
 android:layout\_width="143dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginRight="20dp"  
 android:layout\_weight="1"  
 android:text="Rétablir"  
 android:textAllCaps="false"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="63dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:text="Résultat"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1" />  
 </LinearLayout>  
</LinearLayout>

**Le code de mainActivity.java est :**

TextView t1=(TextView)findViewById(R.id.*t1*);  
 TextView t2=(TextView)findViewById(R.id.*t2*);  
 TextView t3=(TextView)findViewById(R.id.*textView4*);  
 Button bt=(Button)findViewById(R.id.*bt*) ;  
Button rst=(Button)findViewById(R.id.*rst*) ;  
  
 bt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 String x= t1.getText().toString();;  
 String y= t2.getText().toString();  
 String z=x+" "+y;  
 t3.setText(z);  
  
 }  
 });

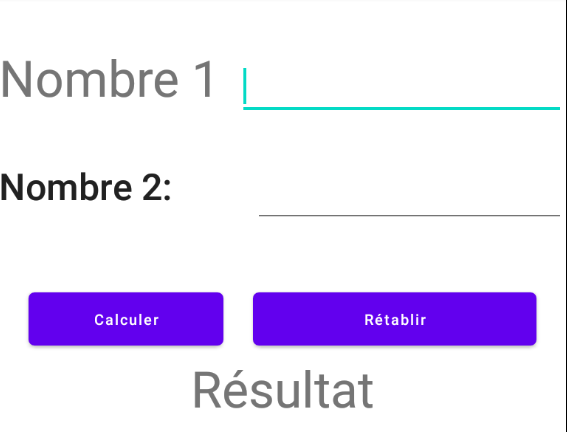
Pour verifier si un texte est vide :

String x=ed1.getText().toString();  
if(x.isEmpty())  
 Toast.*makeText*(getApplicationContext(),"Vide", Toast.*LENGTH\_SHORT*).show();

**Exemple 2:**

1. Concevoir une interface qui permet de calculer la somme de deux nombres entiers.
2. Ecrire le code correspondant.
3. Ajouter le bouton rétablir pour rétablir le contenu de l’interface.

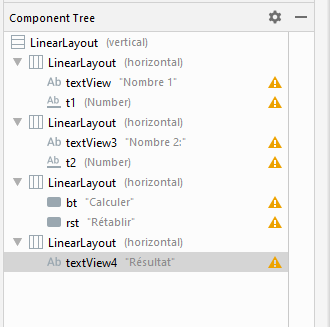
Réponse :



Le code xml est :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:layout\_marginTop="25dp"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="63dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:gravity="center\_vertical"  
 android:text="Nombre 1"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1" />  
  
 <EditText  
 android:id="@+id/t1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="number" />  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="58dp"  
 android:layout\_marginTop="25dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="170dp"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="center\_vertical"  
 android:layout\_weight="1"  
 android:autoText="false"  
 android:freezesText="false"  
 android:text="Nombre 2:"  
 android:textAllCaps="false"  
 android:textAppearance="@style/TextAppearance.AppCompat.Body2"  
 android:textSize="25sp" />  
  
 <EditText  
 android:id="@+id/t2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="number" />  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="25dp"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/bt"  
 android:layout\_width="83dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_weight="1"  
 android:text="Calculer"  
 android:textAllCaps="false"  
 android:textSize="10sp" />  
  
 <Button  
 android:id="@+id/rst"  
 android:layout\_width="143dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginEnd="20dp"  
 android:layout\_marginRight="20dp"  
 android:layout\_weight="1"  
 android:text="Rétablir"  
 android:textAllCaps="false"  
 android:textSize="10sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="63dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:gravity="center"  
 android:text="Résultat"  
 android:textAppearance="@style/TextAppearance.AppCompat.Display1" />  
 </LinearLayout>  
</LinearLayout>

La structure du document est :



On passe au code de MainActivity.java.

public void onClick(View v) {  
 String x= t1.getText().toString();  
 String y= t2.getText().toString();  
 int z=Integer.*parseInt*(x)+Integer.*parseInt*(y);  
 t3.setText(Integer.*toString*(z));  
  
}

ou bien

bt.setOnClickListener(this);

public void onClick(View v) {  
 t3.setText("autre méthode");

}

Ou

String x= ed1.getText().toString();  
String y= ed2.getText().toString();  
**double** d=Double.*parseDouble*(x)+Double.*parseDouble*(y);  
**int** z=Integer.*parseInt*(x)+Integer.*parseInt*(y);  
ed3.setText(String.*format*(**"%.2f"**, d));

Pour rétablir l’interface on utilise :

**rst.setOnClickListener(this::afficher);**

private void afficher(View view) {  
 t3.setText("");  
 t1.setText("");  
 t2.setText("");  
}

**Exercice d’application : TP1 exercice 1 et 2.**

[**https://github.com/amimiabderrahman/XAMARIN-TPS/blob/master/TP1.pdf**](https://github.com/amimiabderrahman/XAMARIN-TPS/blob/master/TP1.pdf)

1. **RadioButton et les CheckBox :**

Parmi les propriétés de ce widget:

<RadioGroup  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" >  
  
 <RadioButton  
 android:id="@+id/radioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="Facile" />

……….

</RadioGroup>

Exemple :



Le code XML est le suivant :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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">  
 <RadioGroup  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" >  
 <RadioButton  
 android:id="@+id/radioButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="Facile" />  
  
 <RadioButton  
 android:id="@+id/radioButton2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Moyen" />  
  
 <RadioButton  
 android:id="@+id/radioButton3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Difficile" />  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="25dp"  
 android:text="Afficher"  
 android:textAllCaps="false" />  
 </RadioGroup>  
</LinearLayout>

Pour accéder au composants radioButtons :

bt1=(RadioButton)findViewById(R.id.*radioButton*);  
bt2=(RadioButton)findViewById(R.id.*radioButton*);  
bt3=(RadioButton)findViewById(R.id.*radioButton*);  
rg=(RadioGroup)findViewById(R.id.*rg*) ;  
t1=(TextView)findViewById(R.id.*textView*);  
b1=(Button)findViewById(R.id.*button*);  
b1.setOnClickListener(this::Afficher);

private void Afficher(View view) {  
*//recupérér le text du radio button coché* RadioButton rb = (RadioButton)findViewById(rg.getCheckedRadioButtonId());  
 t1.setText(rb.getText());  
 }

ou bien

private void Afficher(View view) {  
*//recupérér le text du radio button coché* if(bt1.isChecked()) {  
 t1.setText("1");  
 }  
 if(bt2.isChecked()) {  
 t1.setText("2");  
 }  
 if(bt3.isChecked()) {  
 t1.setText("3");  
 }  
 }

on peut aussi utiliser switch :

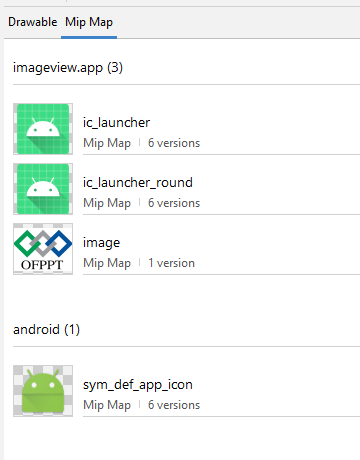
private void Afficher(View view) {  
 switch(rg.getCheckedRadioButtonId())  
 {  
 case R.id.*radioButton*:  
 {  
 t1.setText("facile");  
 break;  
 }  
 case R.id.*radioButton2*:  
 {  
 t1.setText("Moyen");  
 break;  
 }  
 case R.id.*radioButton3*:  
 {  
 t1.setText("Difficile");  
 break;  
 }  
 }

TP2 - Ex1 : <https://github.com/amimiabderrahman/XAMARIN-TPS/blob/master/TP2.pdf>

**cc**.setVisibility(View.***VISIBLE***);

1. **ImageView : Pour insérer une image.**

En premier lieu glisser votre image vers le dossier mipmap ou drawable ; puis glisser le widget ImageView vers votre layout et choisir l’image soit dans l’onglet drawable ou mipmap.



Puis cliquer sur OK.

Parmi les attributs de ce composant :

android:id="@+id/imageView2"  
android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"  
**app:srcCompat="@mipmap/image"**

Le code pour charger une image est :

b1=(Button)findViewById(R.id.*button*);  
 img=(ImageView)findViewById(R.id.*imageView2*);  
 b1.setOnClickListener(this::inserer);  
}  
  
private void inserer(View view) {  
 this.img.**setImageResource**(R.drawable.*ic\_launcher\_background*);  
}

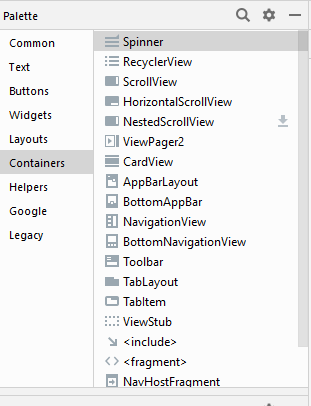
1. **ImageButton : Pour insérer un button avec une image.**

Parmi les attributs de ce composant :

android:id="@+id/simpleImageButtonYouTube"  
android:layout\_width="wrap\_content"  
android:layout\_height="wrap\_content"  
android:layout\_below="@+id/simpleImageButtonHome"  
android:layout\_centerHorizontal="true"  
android:layout\_marginTop="20dp"  
android:background="#005"  
android:padding="20dp"  
app:srcCompat="@mipmap/image"

* **spinner ou le combobox:**

A partir de la palette Containers on glisse le spinner vers notre layout.



Parmi les attributs de ce composant :

android:id="@+id/spinner"  
android:layout\_width="match\_parent"  
android:layout\_height="wrap\_content"

Android Spinner a 2 modes (modes) avec des interfaces complètement différents:

android:spinnerMode="dropdown"

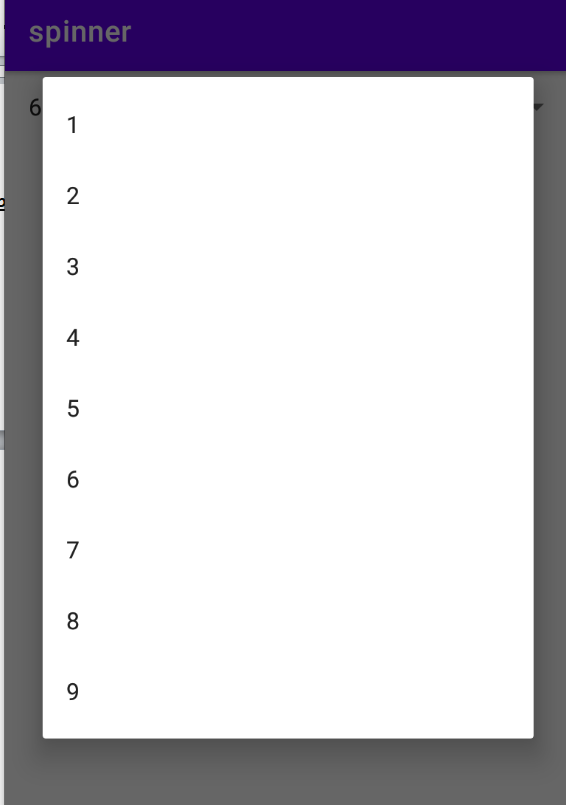
android:spinnerMode="dialog"

* **Remplir le spinner :**

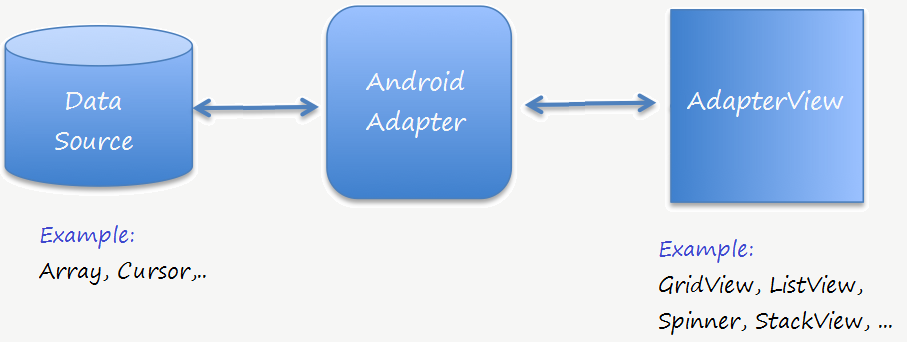
**Exemple 1 : A partir d’un tableau :**

sp=(Spinner)findViewById(R.id.*spinner*);  
Integer[]t=new Integer[]{1,2,3,4,5,6,7,8,9};  
 ArrayAdapter<Integer> ad  
 = new ArrayAdapter<Integer>(this, android.R.layout.*simple\_list\_item\_1* , t);  
 sp.setAdapter(ad);

Le Résultat est :



***Un adapdateur est une classe qui permet de passer des données d’une base de données (Tableau, liste, arraylist….) vers une interface contenant un spinner.***



ArrayAdapter() est un constructeur qui permet de remplir le spinner à partir d’un tableau et choisir le type de Liste pour le spinner.

**Exemple 2 : A partir d’un tableau du fichier strings.xml:**

Les propriétés du spinner sont :

<Spinner  
 android:id="@+id/spinner"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 />

Créant un tableau dans le fichier strings.xml

Ouvrir le fichier strings.xml.

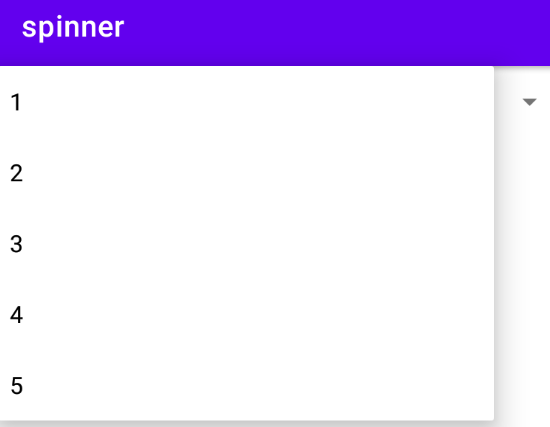
<string-array name="liste">  
 <item>1</item>  
 <item>2</item>  
 <item>3</item>  
 <item>4</item>

<item>5</item>  
</string-array>

Le code pour remplir le spinner à partir du tableau de strins.xml est :

sp=(Spinner)findViewById(R.id.*spinner*);  
ArrayAdapter<CharSequence> adapter = ArrayAdapter.*createFromResource*(this,  
 R.array.*liste*, android.R.layout.*simple\_spinner\_dropdown\_item*);  
sp.setAdapter(adapter);

Le résultat est :



Pour choisir un élément de spinner ; il faut implémenter des méthodes

@Override  
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
  
}  
  
@Override  
public void onNothingSelected(AdapterView<?> parent) {  
  
}

Pour le faire modifier la ligne de la classe MainActivity en ajoutant

public class MainActivity extends AppCompatActivity implements  
 AdapterView.OnItemSelectedListener{……}

En suite cliquer sur implement method dans le menu contextuel de cette partie soulignée en rouge.

Ajouter la ligne suivante à la méthode onCreate () pour que le spinner ecoute la selection de l’élément.

sp.setOnItemSelectedListener(this);

En suite modifier la méthode :

@Override  
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {  
String x=parent.getItemAtPosition(position).toString();  
 t1.setText(x);}

**Exemple 3 : A partir de ArrayList :**

sp=(Spinner)(findViewById(R.id.*spinner*));  
liste=new ArrayList();  
for(int i=1;i<=10;i++) {  
  
 liste.add(Integer.*toString*(i));  
}  
ArrayAdapter ad=new ArrayAdapter(this, android.R.layout.*simple\_spinner\_dropdown\_item*,liste);  
sp.setAdapter(ad);

**Exemple 4 : A partir d’une List :**

Ajouter la bibliothèque : import java.util.List;

sp=(Spinner)(findViewById(R.id.*spinner*));  
liste=new ArrayList();  
for(int i=1;i<=10;i++) {  
  
 liste.add(Integer.*toString*(i));  
}  
ArrayAdapter ad=new ArrayAdapter(this, android.R.layout.*simple\_spinner\_dropdown\_item*,liste);  
sp.setAdapter(ad);

Remarque : Pour récupérér la valeur Spinner on peut utiliser sp. getSelectedItem() ;

**tv1**.setText(**sp**.getSelectedItem().toString());

* **ListView:**

C’est un composant Widget qui permet d’afficher des données empilées dans une colone.

Parmi ses attributs :

<ListView  
 android:id="@+id/listView1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />

* **Remplir une ListView**
* **Parmi les propriétés de ListView :**

android:choiceMode="singleChoice"

ou

android:choiceMode="MultipleChoice"

* **A partir d’un tableau :**

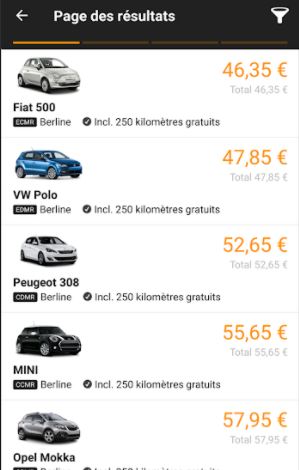
liste1=(ListView)(findViewById(R.id.*listView1*));  
Integer[]t=new Integer[]{1,2,3,4,5,6,7,8,9};  
ArrayAdapter ad=new ArrayAdapter(this, android.R.layout.*simple\_list\_item\_1*,t);  
liste1.setAdapter(ad);

* **A partir de ArrayList:**

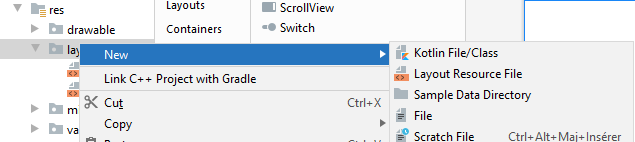
liste1=(ListView)(findViewById(R.id.*listView1*));  
liste=new ArrayList();  
for(int i=1;i<=10;i++) {  
  
 liste.add(Integer.*toString*(i));  
}  
ArrayAdapter ad=new ArrayAdapter(this, android.R.layout.*simple\_list\_item\_1*,liste);  
liste1.setAdapter(ad);

* **Utilisation de la classe BaseAdapter :**

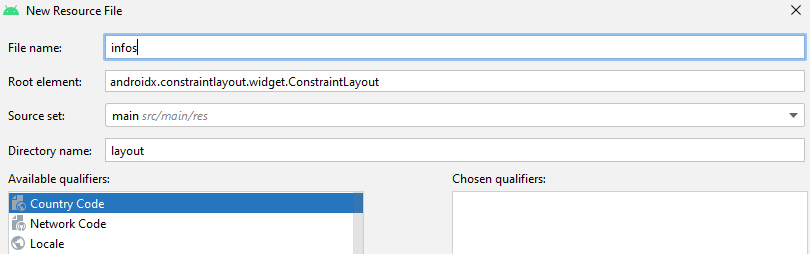
**Exemple:**

****

1. Ajouter une Layout( dans le volet à gauche du projet) dans le dossier : res/layout activer le menu contextuel et cliquer sur Layout Ressources File.(infos.xml)

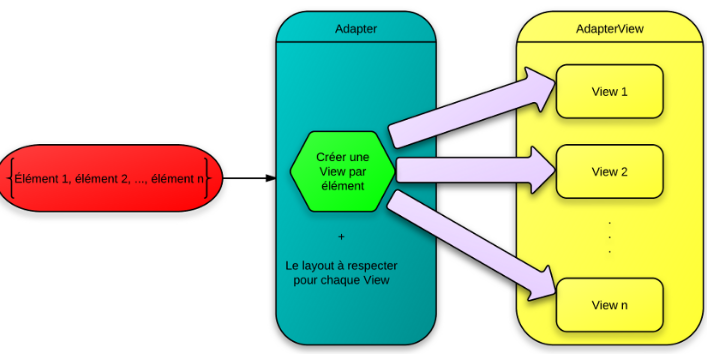


1. La fenêtre suivante s’ouvre :



Taper le nom de la nouvelle layout (Il doit en minuscule)/OK.

Le code xml de cette layout est :





*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:layout\_width="181dp"  
 android:layout\_height="99dp" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="107dp"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="49dp"  
 android:text="TextView" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="47dp"  
 android:text="TextView" />  
 </LinearLayout>  
</LinearLayout>

1. Créant une classe Voiture (intitule, prix , image) et l’ajouter dans le dossier java/com.exemple/Votreprojet du projet.

public class Stagiaire {  
public int image;  
public String nom, prenom;  
public Stagiaire(int image, String nom, String prenom)  
{  
 this.image = image;  
 this.nom=nom;  
 this.prenom=prenom;  
}

1. Ajouter les images dans Drawable ou mipmap :

Sélectionner les images de votre dossier et les glisser vers Drawable ou mipmap  de votre projet.

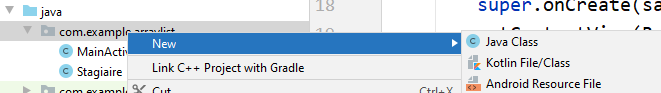
1. Créer une liste d’objet Voiture et la remplir par des exemples d’enregistrements.

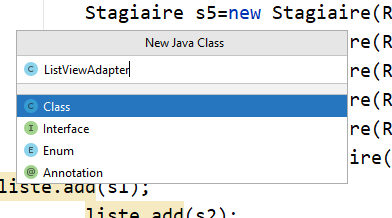
ls=(ListView)(findViewById(R.id.*listView1*));  
 ArrayList liste=new ArrayList();  
 Stagiaire s1=new Stagiaire(R.drawable.*a1*,"a1","b");  
 Stagiaire s2=new Stagiaire(R.drawable.*a1*,"a2","b");  
 Stagiaire s3=new Stagiaire(R.drawable.*a1*,"a3","b");  
 Stagiaire s4=new Stagiaire(R.drawable.*a1*,"a4","b");  
 Stagiaire s5=new Stagiaire(R.drawable.*a1*,"a5","b");  
 Stagiaire s6=new Stagiaire(R.drawable.*a1*,"a6","b");  
 Stagiaire s7=new Stagiaire(R.drawable.*a1*,"a7","b");  
 Stagiaire s8=new Stagiaire(R.drawable.*a1*,"a8","b");  
 Stagiaire s9=new Stagiaire(R.drawable.*a1*,"a9","b");  
 Stagiaire s10=new Stagiaire(R.drawable.*a1*,"10a","b");  
liste.add(s1);  
 liste.add(s2);  
 liste.add(s3);  
 liste.add(s4);  
 liste.add(s5);  
 liste.add(s6);  
 liste.add(s7);  
 liste.add(s8);  
 liste.add(s9);  
 liste.add(s10);

1. Créer la classe ListViewAdapter :

On crée une liste ListViewAdapter qui est dérivée de la classe BaseAdapter et qui permet de transférer une liste de données vers ListView.

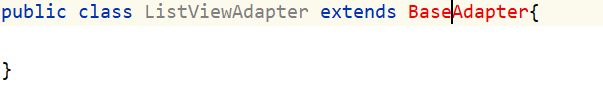
* Dans le dossier java/com.exemples/Votreprojet ajouter une classe par menu contextuel et la nommer sous : ListViewAdapter





Puis valider par Entrer.

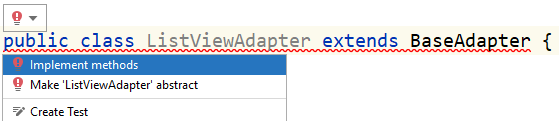
* Hériter cette classe de la classe BaseAdapter ;



Puis Appuyer sur Alt+Entrer pour charger la bibliothique :

import android.widget.BaseAdapter;

Ensuite cliquer sur implement methods :



Le résultat sera :

public class ListViewAdapter extends BaseAdapter {  
 @Override  
 public int getCount() {  
 return 0;  
 }  
 @Override  
 public Object getItem(int position) {  
 return null;  
 }  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 return null;  
 }

Compléter la classe ListViewAdapter par :

package com.example.arraylist;  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.constraintlayout.solver.LinearSystem;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ListViewAdapter extends BaseAdapter {  
 private List<Stagiaire> listData;  
 private LayoutInflater layoutInflater;  
 private Context context;  
  
 public ListViewAdapter(Context aContext, List<Stagiaire> listData) {  
 this.context = aContext;  
 this.listData = listData;  
 layoutInflater = LayoutInflater.*from*(aContext);  
 }  
 @Override  
 public int getCount() {  
 return listData.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return listData.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 layoutInflater=(LayoutInflater)context.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  
 convertView = layoutInflater.inflate(R.layout.*infos*, parent, false);  
 Stagiaire s = (Stagiaire) this.listData.get(position);  
 ImageView img=(ImageView) convertView.findViewById(R.id.*image*);  
 TextView t1=(TextView)convertView.findViewById(R.id.*textView*);  
 TextView t2=(TextView)convertView.findViewById(R.id.*textView2*);  
 img.setImageResource(s.image);  
 t1.setText(s.nom);  
 t2.setText(s.prenom);  
 return convertView; }}

ListViewAdapter ad=**new** ListViewAdapter(**this**,liste);  
**ls**.setAdapter(ad);

Récupérer les données d’un item sélectionné :

Dans la méthode OnCreate() Ajouter le code de l’évenement sur LS :

ls.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
  
 @Override  
 public void onItemClick(AdapterView<?> a, View v, int position, long id) {  
 Object o = ls.getItemAtPosition(position);  
 Stagiaire s = (Stagiaire) o;  
 Toast.*makeText*(MainActivity.this, "Selected :" + " " + s.nom, Toast.*LENGTH\_LONG*).show();  
 }  
});

* **A partir des tableaux:**

**ListviewAdapter :**

package com.example.arraylist;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.constraintlayout.solver.LinearSystem;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ListViewAdapter extends BaseAdapter {  
 *// Ajouter les attributs en relation avec la layout Gabarit* Context context;  
 int[]images;  
 String[]noms;  
 String[]prenoms;  
 private LayoutInflater layoutInflater;  
 *// Ajouter le constructeur* ListViewAdapter(Context context, int[]images,String[]noms,String[]prenoms)  
 {  
 super();  
  
 this.context=context;  
 this.images=images;  
 this.noms=noms;  
 this.prenoms=prenoms;  
 }  
 @Override  
 public int getCount() {  
 return noms.length;  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return noms[position];  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return position;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
 layoutInflater=(LayoutInflater)context.getSystemService(Context.*LAYOUT\_INFLATER\_SERVICE*);  
 convertView = layoutInflater.inflate(R.layout.*infos*, parent, false);  
 ImageView img=(ImageView) convertView.findViewById(R.id.*image*);  
 TextView t1=(TextView)convertView.findViewById(R.id.*textView*);  
 TextView t2=(TextView)convertView.findViewById(R.id.*textView2*);  
 img.setImageResource(images[position]);  
 t1.setText(noms[position]);  
 t2.setText(prenoms[position]);  
 return convertView;  
 }  
}

**MAinActivity :**

ls=(ListView)(findViewById(R.id.*listView1*));  
 int[]images={R.drawable.*a1*,R.drawable.*a2*,R.drawable.*a3*,R.drawable.*a4*,R.drawable.*a5*,R.drawable.*a6*,R.drawable.*a7*,R.drawable.*a8*,R.drawable.*a9*,R.drawable.*a10*};  
 String[]noms={"a","a","a","a","a","a","a","a","a","a"};  
 String[]prenoms={"p","p","p","p","p","p","p","p","p"};  
ls.setAdapter(new ListViewAdapter(this, images,noms,prenoms));  
  
Toast.*makeText*(MainActivity.this, "Selected :" + " " , Toast.*LENGTH\_LONG*).show();  
 ls.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
  
 @Override  
 public void onItemClick(AdapterView<?> a, View v, int position, long id) {  
 Object o = ls.getItemAtPosition(position);  
 Toast.*makeText*(MainActivity.this, "Selected :" + " " , Toast.*LENGTH\_LONG*).show();  
 }  
 });  
 }

**La méthode Infate() est une méthode de la classe LayouInflater qui permet de créer une variable sous forme de View.**

* **CalendarView ou** **DatePicker: Insertion de calendrier.**

Parmi les attributs de ce widget :

<CalendarView  
 android:id="@+id/calendarView3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />

On peut la configurer à partir de la fenêtre des propriétés:

android:maxDate="31/12/2023"  
android:**minDate**="01/01/2019"

android:firstDayOfWeek="2"

Le code pour récupérer la valeur du calendrier est :

cv=(CalendarView)findViewById(R.id.*calendarView3*);  
tv=(TextView)findViewById(R.id.*textView*);  
cv.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {  
 @Override  
 public void onSelectedDayChange(CalendarView view, int year, int month, int dayOfMonth) {  
 *// display the selected date by using a toast*  
 }  
});

* **Le scrollView Vertical:**

C’est un widget qui permet de défiler son contenu qui dépasse la hauteur du device.

Parmi les propriétés de ce widget :

<ScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">

…..

</ScrollView>

Dans ce widget scollView , on ajoute une linearLayout Verticale qui contient des boutons comme suit :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
  
 <ScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical" >  
  
 <Button  
 android:id="@+id/button30"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button40"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button44"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button45"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button46"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button31"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button32"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button43"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button42"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button41"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button39"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button38"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button33"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button34"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button37"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button35"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button36"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
 </LinearLayout>  
 </ScrollView>  
</LinearLayout>

* **Le scrollView Horizontal:**

**Elle permet de défiler les widgets disposés hrizontalement.**

**Exemple :**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
 <HorizontalScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button47"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button48"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button49"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button50"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button51"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button52"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Button" />  
 </LinearLayout>  
 </HorizontalScrollView>  
</LinearLayout>

* **Le Switch:**

**Parmi les propriétés de ce widget sont :**

<Switch

android:id="@+id/simpleSwitch"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

Le code pour vérifier ci elle active :

Boolean switchState = simpleSwitch.isChecked();

* **AnalogClock :**

<AnalogClock  
 android:id="@+id/simpleAnalogClock"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content" />

* **DigitalClock :**

<DigitalClock

android:id="@+id/simpleDigitalClock"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:background="#0f0"

android:padding="30dp"/>

* **TimePicker:**

<**TimePicker  
 android:id="@+id/simpleTimePicker"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:timePickerMode="spinner"** />

Les modes de TimePicker :

**android:timePickerMode="spinner"**

**android:timePickerMode="clock"**

time = (TextView) findViewById(R.id.time);

simpleTimePicker = (TimePicker) findViewById(R.id.simpleTimePicker);

simpleTimePicker.setIs24HourView(false); // used to display AM/PM mode

// perform set on time changed listener event

simpleTimePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener() {

@Override

public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {

// display a toast with changed values of time picker

Toast.makeText(getApplicationContext(), hourOfDay + " " + minute, Toast.LENGTH\_SHORT).show();

time.setText("Time is :: " + hourOfDay + " : " + minute); // set the current time in text view

}

});

**Les Boites de dialogue**

* + - * 1. **La Boite Toast ou Boite Transitoire :**

C’est un message transitoire; il ne nécessite aucune intervention de la part de l'utilisateur et ne prend même pas le focus.

Toast.*makeText*(getApplicationContext(), "Message", Toast.*LENGTH\_LONG*).show();

Ou

bt=(Button)findViewById(R.id.*button53*);  
bt.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v) {  
 Toast toast = Toast.*makeText*(MainActivity.this, "Je suis un Message!", Toast.*LENGTH\_SHORT*);  
 toast.setGravity(Gravity.*TOP* | Gravity.*LEFT*, 20, 30);  
 toast.show();  
 }  
});

* **duration = Toast.LENGTH\_LONG**, signifie que **Toast**s'affichera pendant une longue période, à savoir 3,5 secondes.
* **duration = Toast.LENGTH\_SHORT,** signifie que **Toast**s'affichera pendant une courte période, à savoir 2 secondes.
  + - * 1. **La Boite Toast Personnalisée :**
* **Ajouter une nouvelle layout :**
* **Ajouter des widgets à votre layout :**

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/toast\_layout\_root"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#E13B3B"  
 android:orientation="vertical">  
  
 <ImageView  
 android:id="@+id/imageView4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 tools:srcCompat="@tools:sample/avatars" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="62dp"  
 android:text="Bonjour" />  
  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
  
</LinearLayout>

* **Ajouer le code suivant à Votre bouton dans Main\_activity :**

Context context = getApplicationContext();  
LayoutInflater inflater = getLayoutInflater();  
 View v0 = inflater.inflate(R.layout.*toast1*, null );  
 ImageView img=v0.findViewById(R.id.*imageView4*);  
img.setImageResource(R.drawable.*ic\_launcher\_foreground*);  
 Toast toast = new Toast(context);  
 toast.setView(v0 );  
 toast.setDuration(Toast.*LENGTH\_LONG*);  
 toast.show();

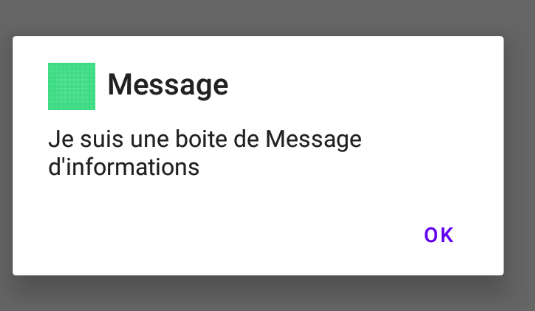
* + - * 1. **La Boite de Message AlertDialog :**

AlertDialog.Builder ad=new AlertDialog.Builder(MainActivity.this);  
ad.setTitle("Message");  
ad.setMessage("Je suis une boite de Message d'informations");  
ad.setIcon(R.drawable.*ic\_launcher\_background*);

ad.setCancelable(false);  
 ad.setNegativeButton("OK",  
 new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(getApplicationContext(), "Click sur le boutton OK", Toast.*LENGTH\_SHORT*).show();  
 };  
});

ad.show();

Le résultat est :



* + - * 1. **La Boite de Confirmation à deux boutons AlertDialog :**

AlertDialog.Builder ad=new AlertDialog.Builder(MainActivity.this);  
 ad.setTitle("Message");  
 ad.setMessage("Voulez vous quitter l'application?");  
 ad.setIcon(R.drawable.*ic\_launcher\_background*);  
ad.setPositiveButton("Non", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(getApplicationContext(), "Click sur le boutton OK", Toast.*LENGTH\_SHORT*).show();  
 };  
 });  
 ad.setCancelable(false);  
 ad.setNegativeButton("Oui",  
 new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(getApplicationContext(), "Click sur le boutton Oui", Toast.*LENGTH\_SHORT*).show();  
 };  
 });

ad.create();  
ad.show();

Le résultat est :



* + - * 1. **La Boite de Confirmation à trois buttons AlertDialog :**

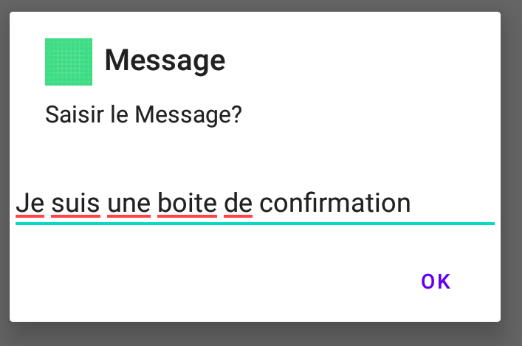
Ajouter le code suivant au code ci-dessus :

ad.setNeutralButton("Annuler",  
 new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(getApplicationContext(), "Click sur le boutton Annuler", Toast.*LENGTH\_SHORT*).show();  
 };  
});  
 ad.create();  
 ad.show();

* + - * 1. **La Boite de Saisi AlertDialog :**

AlertDialog.Builder ad=new AlertDialog.Builder(MainActivity.this);  
 ad.setTitle("Message");  
 ad.setMessage("Saisir le Message?");  
 ad.setIcon(R.drawable.*ic\_launcher\_background*);  
 **EditText ed1=new EditText(MainActivity.this);  
 ad.setView(ed1);**   
ad.setPositiveButton("Ok", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int which) {  
 Toast.*makeText*(getApplicationContext(), "Votre message est :"+ed1.getText(), Toast.*LENGTH\_SHORT*).show();  
 };  
 });  
 ad.create();  
 ad.show();

**Le résultat est :**



* + - * 1. **La Boite personnalisée de AlertDialog :**
* Créer une layout de votre choix :

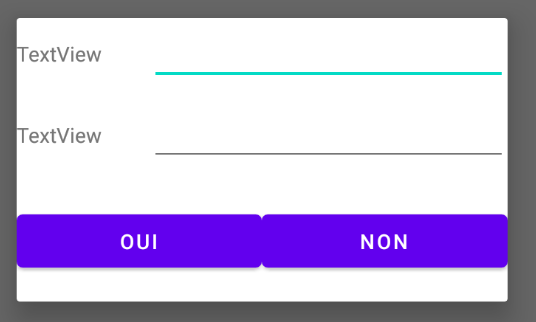
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="55dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="29dp"  
 android:layout\_weight="1"  
 android:text="TextView" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="textPersonName"  
 android:text="Name" />  
</LinearLayout>  
  
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="72dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:id="@+id/textView3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="TextView" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:ems="10"  
 android:inputType="textPersonName"  
 android:text="Name" />  
</LinearLayout>  
  
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="65dp"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Oui" />  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Non" />  
</LinearLayout>

* Ajouter le code suivant à Main\_actvity.java :

Button bt=(Button)findViewById(R.id.*button*);  
 bt.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v) {  
 LayoutInflater inflater = getLayoutInflater();  
 View v0 = inflater.inflate(R.layout.*message*, null );  
 EditText ed1=v0.findViewById(R.id.*editTextTextPersonName*);  
 EditText ed2=v0.findViewById(R.id.*editTextTextPersonName2*);  
Button b1=v0.findViewById(R.id.*button2*);  
 Button b2=v0.findViewById(R.id.*button3*);  
 b1.setOnClickListener(new View.OnClickListener(){  
  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(getApplicationContext(), "Bouton 1", Toast.*LENGTH\_LONG*).show();  
 }  
 });  
 AlertDialog.Builder ad=new AlertDialog.Builder(MainActivity.this);  
 ad.setTitle("Message");  
 ad.setMessage("Saisir le Message?");  
 ad.setIcon(R.drawable.*ic\_launcher\_background*);  
 ad.setView(v0);  
 ad.create();  
 ad.show();  
  
 }  
 });

AlertDialog.Builder(MainActivity.this);  
 ad.setTitle("Message");  
 ad.setMessage("Saisir le Message?");  
 ad.setIcon(R.drawable.*ic\_launcher\_background*);  
 ad.setView(v0);  
 ad.create();  
 ad.show();

**Le Rsultat est :**



Les projets Multi-vues(layouts)

* + - * 1. Définition :

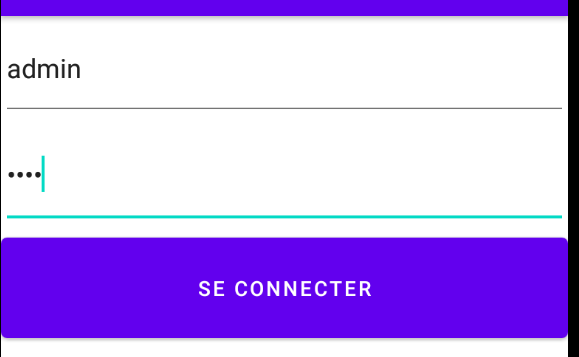
Les intents sont des messages (intentions ou requetes) qui assurent la communication entres les vues ou layouts d’une application Mobile.

Exemple :

* Définir un **Intent** qui devrait être utilisée pour lancer une **Activity**.
* Envoyer des données entre les Vues à travers les intents.
* Démarrer un service
* ……
  + - * 1. Les types d’intents:
* **Intents Explicites :** Ces intents sont généralement utilisés dans une application Mobile ; ils spécifient explicitement le nom du composant cible pour gérer l'intention.

**Exemple 1** : Passer des données d’une activité à une autre.

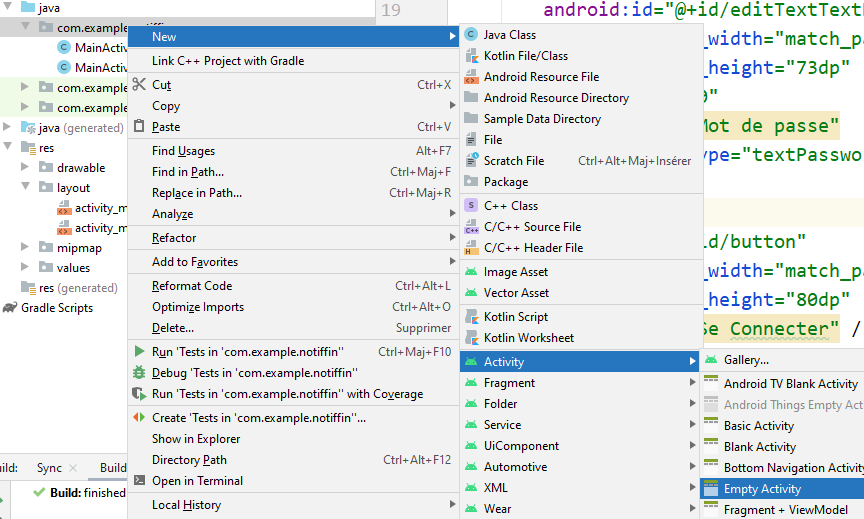
* Créer un nouveau projet sous android studio :



Utliser le code suivant :

<EditText  
 android:id="@+id/editTextTextPersonName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="71dp"  
 android:ems="10"  
 android:hint="Utilisateur"  
 android:inputType="textPersonName" />  
  
<EditText  
 android:id="@+id/editTextTextPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="73dp"  
 android:ems="10"  
 android:hint="Mot de passe"  
 android:inputType="textPassword" />  
  
<Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="80dp"  
 android:text="Se Connecter" />

* Ajouter une nouvelle activité :



* Taper le nom de la nouvelle activité (remarquer l’ajout du fichier xml).
* Remarquer l’ajout d’une ligne dans androimanifet :

<activity android:name=".MainActivity3"></activity>

* Dans MainActivity Ecrire le code suivant :

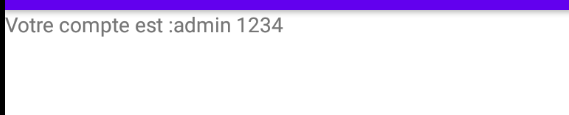
Button bt=(Button)findViewById(R.id.*button*);  
ed1=(EditText)findViewById(R.id.*editTextTextPersonName*);  
ed2=(EditText)findViewById(R.id.*editTextTextPassword*);  
bt.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v) {  
 Intent intent=new Intent(MainActivity.this,MainActivity2.class);  
 intent.putExtra("login",ed1.getText().toString());

intent.putExtra("pwd",ed2.getText().toString());  
 startActivity(intent);  
  
 }  
});

**Ce code permet d’ouvrir une nouvelle activité et transmettre les donénes de la variable msg1 vers la deuxieme activité.**

* **Dans** MainActivity2 Ecrire le code suivant pour l’afficher dans cette activité.

TextView tv1=(TextView)findViewById(R.id.*textView*);  
Intent t=getIntent();  
String login=t.getStringExtra("login");  
String pwd=t.getStringExtra("pwd");  
tv1.setText("Votre compte est :"+ login+" "+pwd);



**Exemple 2** : Passer des données d’une activité à une autre avec Retour.

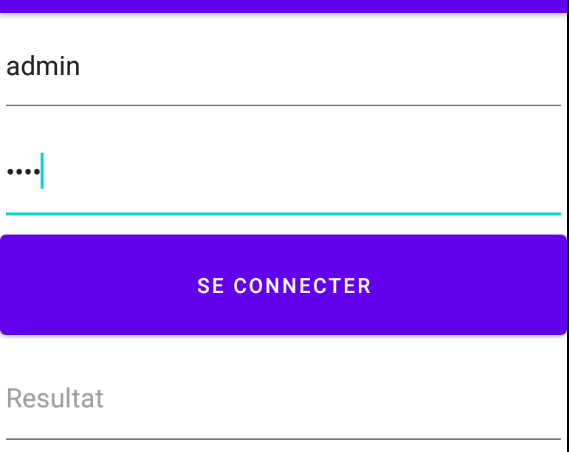
* Dans ActivityMain ajouter le code xml pour récupérer le résultat retournée:

<EditText  
 android:id="@+id/res"  
 android:layout\_width="match\_parent"  
 android:layout\_height="73dp"  
 android:ems="10"  
 android:hint="Resultat"  
 android:inputType="textPersonName" />

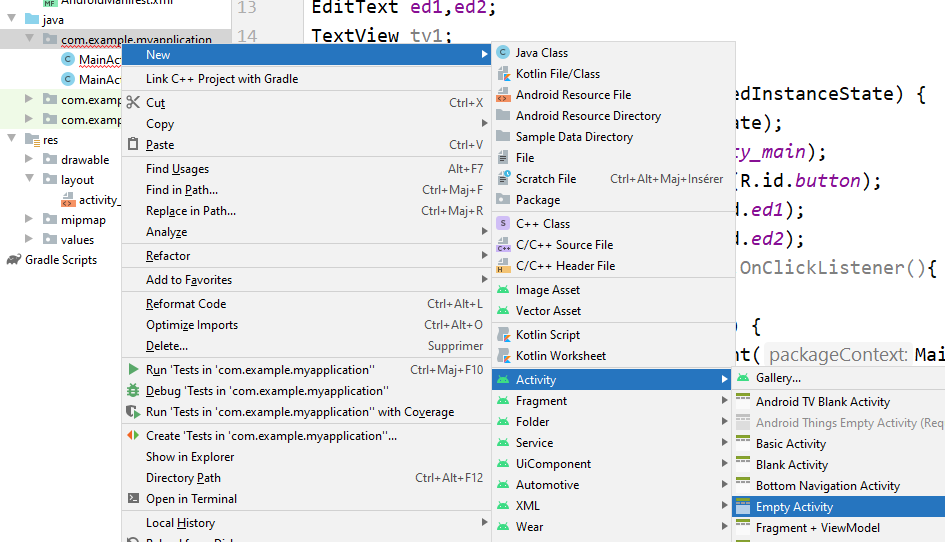
* Dans MainActivity Ecrire le code suivant :

Button bt=(Button)findViewById(R.id.*button*);  
ed1=(EditText)findViewById(R.id.*editTextTextPersonName*);  
ed2=(EditText)findViewById(R.id.*editTextTextPassword*);  
bt.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v) {  
 Intent intent=new Intent(MainActivity.this,MainActivity2.class);  
 intent.putExtra("login",ed1.getText().toString());  
 intent.putExtra("pwd",ed2.getText().toString());  
 startActivityForResult(intent,0);  
 }  
});

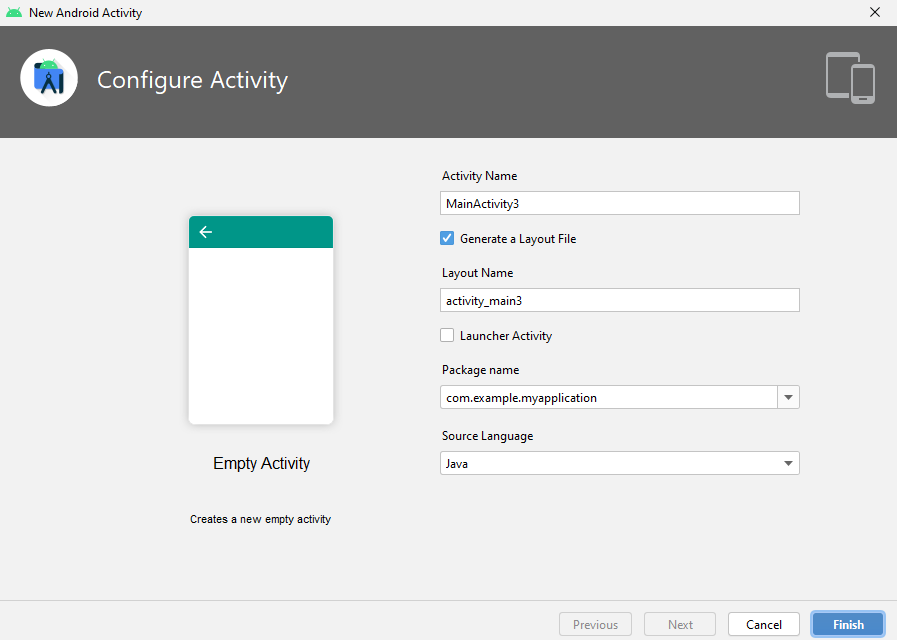
La méthode startActivityForResult() permet de démarrer une activité en fonction du résultat.



**Ajouter une Activité avec sa layout correspeondante :**

****

On aura la boite suivante :



Après on remarque l’ajout d’une activité dans le dossier des activités java/com.exemple.MyApplication et une layout dans le dossier layout.

Dans Mainactivity.java ajouter la méthode protégée onActivityResult()

@Override  
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);

}

Cette méthode est caractérisée par le code resultCode qui indique si on aura une valeur retournée et data contient les données retournées:

@Override  
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
 super.onActivityResult(requestCode, resultCode, data);  
 if(resultCode==*RESULT\_OK*)  
 tv1.setText(data.getStringExtra("login"));  
  
}

* Dans Mainactivity2 ajouter le code suivant la méthode OnCreate() directement pour l’executer directement après le chargement de l’acitivité.

Intent t=getIntent();  
 String login=t.getStringExtra("login");  
 String pwd=t.getStringExtra("pwd");  
  
 Intent inta=new Intent(this,MainActivity.class);  
inta.putExtra("login",login+" "+pwd);  
setResult(*RESULT\_OK*,inta);  
this.finish();  
  
 }

* **Les intents Implicites :**

Se sont des intents qui permettent d’assurer une communication entre des vues de deux ou plusieurs Applications.

Exemple1 : Afficher une page web

Intent alpha = new Intent (Intent.*ACTION\_VIEW*,  
 Uri.*parse*("http://www.google.com"));  
startActivity(alpha);

Le constructeur de Intent est paramétré par :

Action : c’est l’action à réaliser.

URI : Les données sur les quelles, on réalise l’action sous forme de URI (Uniform Ressources Indentifier).

Exemple2-1 : Envoyer un SMS.

Uri sms = Uri.*parse*("sms:0606060606");  
Intent alpha = new Intent (Intent.*ACTION\_SENDTO*,sms);  
startActivity(alpha);

Exemple2-2 : Envoyer un SMS à plusieurs personnes.

Uri sms = Uri.*parse*("sms:0606060606,0606060607?body=Salut les les amis");  
Intent alpha = new Intent (Intent.*ACTION\_SENDTO*,sms);  
startActivity(alpha);

Exemple2 : Démarre le numérateur téléphonique:

Uri tel = Uri.*parse*("tel:9510300000");  
Intent alpha = new Intent(Intent.*ACTION\_VIEW*, tel);  
startActivity(alpha);

Liste des actions standard que les intentions peuvent utiliser pour les activités de lancement (généralement par le biais startActivity (intention) :

• ACTION\_MAIN

• ACTION\_VIEW

• ACTION\_ATTACH\_DATA

• ACTION\_EDIT

• ACTION\_PICK ACTION\_CHOOSER

• ACTION\_GET\_CONTENT

• ACTION DIAL ACTION\_ANSWER

• ACTION\_INSERT

• ACTION\_DELETE

• ACTION\_RUN

• ACTION\_SYNC

• ACTION\_PICK\_ACTIVITY

• ACTION\_SEARCH

• ACTION WEB SEARCH

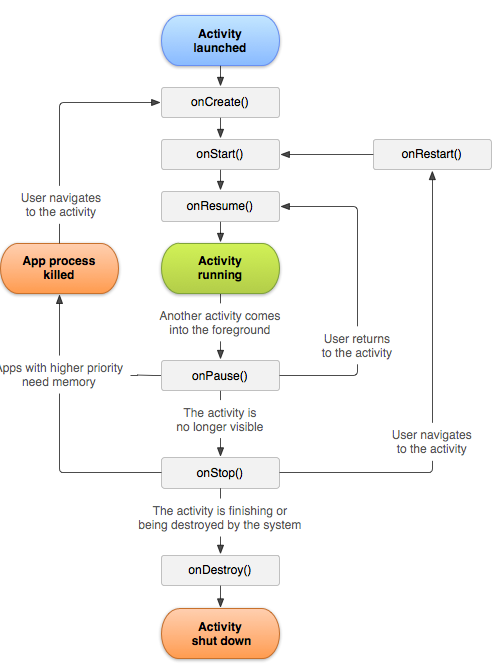
• ACTION\_ACTION\_CALL ACTION\_SEND ACTION\_SENDTO

• ACTION\_WEB\_ACTION\_FACTORY\_TEST

# Le cycle de vie d'une activité

## Définition

Chaque activité a une duréee de vie aucours de l’éxecution d’une application mobile ; ce cycle de vie est représenté par plusieurs méthodes qui permettent au developpeur de gérer ou implémenter certaines fonctionnalités.



## La méthode oncreate() :

Cette méthode est appelée lors de la création de l’activité avec sa vue(Layout). Elle appellée une seule fois dans la vie d’une activité.

Dans cette méthode, on peut déclarer les objets utilsés dans l’application et charger des fonctionnalités au debut de la création de l’activité.

@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);  
}

## La méthode onStart() :

Cette méthode est appellée lorsque l’activité est sur le point de devenir visible(avant son affichage) à l’utilisateur.

@Override  
protected void onStart() {  
 super.onStart();  
   
}

## La méthode onResume() :

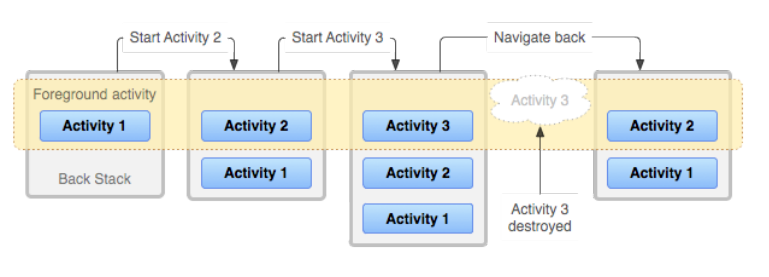
Cette méthode est appellée chaque fois que l’acivité devient visible à l’utilisateur (En premier plan) et devient stable. Elle peut être appelée aussi lorsqu’elle sort de pause.

@Override  
protected void onPause() {  
 super.onPause();  
}

## La méthode onPause() :

L'activité est suspendue, c'est à dire qu'une autre activité vient au premier plan, recouvrant en partie l'activité de notre application. Il faut stopper les traitements qui consomment beaucoup de ressources.

Cette méthode est appellée chaque fois que l’activité devient en arrière plan. Dans ce cas en parle de la Pile « LIFO » (Last In First Out).



@Override  
protected void onPause() {  
 super.onPause();  
 Toast.*makeText*(getApplicationContext(),"OnPause", Toast.*LENGTH\_SHORT*).show();  
  
}

## La méthode OnStop() :

Cette méthode est appellée lorsque l’activité est sur le point de lancer la destruction de l’acitivité (Avant la destruction de l’activité) ; ou mise en arrière plan.

@Override  
protected void onStop() {  
 super.onStop();  
 Toast.*makeText*(getApplicationContext(),"Onstop", Toast.*LENGTH\_SHORT*).show();  
  
}

## La méthode OnDestroy() :

C’est la fin du cycle de vie de l’activité, tous les codes en relation seront déchargés de la mémoire, et toutes les activités seront aussi détruites.

Par exemple, ce peut être après avoir appelée la méthode **finish()**, ou si le système décide d'arrêter l'activité pour libérer de la mémoire.

@Override  
protected void onDestroy() {  
 super.onDestroy();  
 Toast.*makeText*(getApplicationContext(),"Onstop", Toast.*LENGTH\_SHORT*).show();  
  
}

## La méthode OnRestart() :

Cette méthode est appellée lorsque l’activité est sur le point de démarrage(OnStart) après un Stop (onStop);

@Override  
protected void onRestart() {  
 super.onRestart();  
 Toast.*makeText*(getApplicationContext(),"Onstop", Toast.*LENGTH\_SHORT*).show();  
  
}

## Description des états :

Une Activité passe par des états dans sa durée de vie :

* État « start » : quand l’activité n’est pas encore référencée en mémoire, elle est dans un état « start ».
* État « Resume » / « Running » : une activité qui a le focus, est dans un état « Running » à un instant « t ». Cette activité se trouve au sommet de la pile.
* État « Pause » : une activité qui n’a pas le focus (donc elle n’interagit pas avec l’utilisateur), mais reste visible à l’écran, est dans un état « Pause ». Elle maintient son état et reste attachée au gestionnaire de fenêtres. Cette activité peut quand même être détruite par le système si ce dernier se trouve en manque de mémoire par exemple.
* État « Stopped » : une activité qui n’est pas visible à l’écran, mais existe quand même en mémoire, est dans un état « Stopped ».
* État « Destroyed » : quand l’activité n’est plus en mémoire, elle est détruite. Ce phénomène arrive, car le gestionnaire d’activités décide qu’une telle activité n’est plus utilisée.

# Les Ressources d’une Application Android

## Introduction :

Les Ressources d’une Application Android sont des informations statiques stockées en dehors du code java. Le dossier values est le dossier qui contient des sous dossiers et des fichiers xml, réservé pour le stockage des valeurs utilisées par l’application mobile : Les valeurs de types chaine, couleur, styles, tableaux ; dimensions…..

## Le fichier Strings.xml:

Le fichier Strings.xml est réservé pour le stckages des valeurs de type chaine ou de type tableau qui sont utilisés dans les activtés ou les vues d’une Application mobile.

<resources>  
 <string name="app\_name">My Application</string>  
 <string name="x">Je suis la valeur de x</string>  
 <string name="cal">Calculer</string>  
 <string name="rst">Rétablir</string>  
<string-array name="liste">  
 <item>1a</item>  
 <item>2a</item>  
 <item>3a</item>  
 <item>4a</item>  
 </string-array>

<integer-array name="nbre">  
 <item>1</item>  
 <item>2</item>  
 <item>3</item>  
 <item>4</item>  
</integer-array>

</resources>

Les valeurs de types chaine seront appellées par : "@string/cal"

Le tableau est appéllé par : R.array.liste ;

Exemple :

Pour le bouton Calculer pour changer son attribut text : on utilise l’attribut :

<Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text=""@string/cal" />

Pour le tableau, on peut utiliser le code java suivant pour récupérer le contenu du tableau de chaine:

public void onClick(View v) {  
 Resources res = getResources();  
 String[] t = res.getStringArray(R.array.*liste*);  
 Toast.*makeText*(MainActivity.this ," "+t[0],Toast.*LENGTH\_SHORT*).show();  
}

Ou bien pour le tableau des entiers :

Resources res = getResources();  
int[]t=res.getIntArray(R.array.*nbre*);  
Toast.*makeText*(MainActivity.this ," "+t[0],Toast.*LENGTH\_SHORT*).show();

Le fichier strings.xml peut contenir aussi la balise :

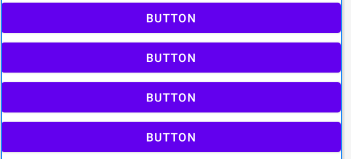
<plurals name="numberOfSongsAvailable">  
 <item quantity="one">One song found.</item>  
 <item quantity="few">%d songs found.</item>  
 <item quantity="other">%d songs found.</item>  
</plurals>

le code Java est :

private int count = 0;

public void onClick(View v) {  
 count++;  
 Resources res = getResources();  
 String songsFound = res.getQuantityString(R.plurals.*numberOfSongsAvailable*, count, count);  
 Toast.*makeText*(MainActivity.this, songsFound,Toast.*LENGTH\_SHORT*).show();  
}

## Exemple :



Modifier l’attribut Text des Boutons en utlisant les valeurs de type chaine du fichier strings.xml

<string name="rst">Rétablir</string>  
<string name="bt1">Somme</string>  
<string name="bt2">Produit</string>  
<string name="bt3">Soustraction</string>  
<string name="bt4">Division</string>

Dans Activity\_Main on modifi l’attributs android :Text comme suit :

<Button  
 android:id="@+id/button1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/bt1" />  
<Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/bt2" />  
<Button  
 android:id="@+id/button3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/bt3" />  
<Button  
 android:id="@+id/button4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="@string/bt4" />

## Le fichier color.xml:

C’est le dossier qui est réservé pour la déclration des valeurs de type couleur ;

<resources>  
 <color name="purple\_200">#FFBB86FC</color>  
 <color name="purple\_500">#FF6200EE</color>  
 <color name="purple\_700">#FF3700B3</color>  
 <color name="teal\_200">#FF03DAC5</color>  
 <color name="teal\_700">#FF018786</color>  
 <color name="black">#FF000000</color>  
 <color name="white">#FFFFFFFF</color>  
</resources>

Chaque balise color est caractérisée par un attribut name qui identifi la couleur et une valeur qui une valeur Hexadécimale d’une couleur.

On appele une couleur de colors.xml par :

android:textColor="@color/black"

Le système offre des couleurs prédéfinies et qui peuvent être appelée par :

android:textColor="@android:color/background\_dark"

ou

android:textColor="@color/design\_default\_color\_error"

## Le fichier themes.xml:

Ce fichier contient les styles qui peuvent être appliqué un composant d’une Layout ; elle est constitué par des propriétées item des couleurs et des paramêtres. Exemple :

<style name="Theme.MyApplication" parent="Theme.MaterialComponents.DayNight.DarkActionBar">  
 *<!-- Primary brand color. -->* <item name="colorPrimary">@color/purple\_500</item>  
 <item name="colorPrimaryVariant">@color/purple\_700</item>  
 <item name="colorOnPrimary">@color/white</item>  
 *<!-- Secondary brand color. -->* <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorSecondaryVariant">@color/teal\_700</item>  
 <item name="colorOnSecondary">@color/black</item>  
 *<!-- Status bar color. -->* <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>  
 *<!-- Customize your theme here. -->*</style>

Application modifier les valeurs d’une couleur et voir son effet sur les objets de l’interface.

En plus , on peut appliquer l’heritage d’une style à partir d’un autre en utilisant l’attribut parent qui a la valeur d’un style déjà défini.

Le système contient aussi comme les coleurs des styles prédéfini , pour les voir cliquer sur le bouton à droite de la propriété style :



Une liste de style sera afficher ; on peut les appliquer une application ou un composant de l’interface.

## Le fichier dimens.xml:

C’est un fichier xml qui est réservé pour le stockage des valeurs de type dimension ces valeurs sont utiliser pour les attribuer au diemension des composants :

Exemple :

<resources>  
 <dimen name="Bt" >60sp</dimen>  
 <dimen name="lar">70dp</dimen>  
</resources>

Pour appeler cette valeur on utilise :

android:textSize="@dimen/Bt"

android:layout\_width="@dimen/lar"

**Remarque :**

* **dp** : Density independent Pixel (Densité de pixels indépendant) - Unité abstraite qui est basés sur la densité physique de l'écran.Cette unité est égale à 160 DPI (Points par pouce) par écran. Cette dimension sera utilisée pour la mise en page des éléments.
* **sp**: Scale independent Pixel (Echelle de pixels indépendant) - Utilisé pour les tailles de polices. On pourrait comparer cette unité aux em du développement web. La police peut être plus ou moins grosse suivant les préférences utilisateurs
* **pt** : Point - 72 points par pouces. basé sur la taille physique de l'écran.
* **px** : Pixels - Corresponds aux pixels réels de l'écran. Cette unité de mesure n'est pas recommandée car le rendu sur les différents types d'écran peut être différent. Le nombre de pixels par pouce peut varier suivant les appareils.

# Les notifications

## Définition :

Une notification est une boite de dialogue envoyé par un service de l’application.

À partir d’API 11, le moteur de création des notifications est la classe Notification builder ; cette classe contient un constructeur qui permet de créer ce moteur de notification.

Une notification channel est une catégorie où sont regroupées des notifications de même nature. Pour une application de médias, il pourrait y avoir une catégorie pour les articles, et une autre pour les vidéos.

Ce regroupement en catégories permet à l’utilisateur d’appliquer des préférences différentes pour chaque channel. Par exemple, l’utilisateur peut n’être intéressé que par les articles et désactiver complètement le channel des vidéos. Il a également la possibilité de choisir l’importance de chaque catégorie pour décider comment être notifié (pop-up, vibration, sonnerie, etc.).

## Crétion de notification :

Pour créer une notification ; on suit les étapes suivantes :

**Exemple 1** :

* **Déclarer le channel de la notification :**

private static final String *CHANNEL\_ID* = "1";

* **Créer Un Moteur de Notifications qui est caractérisé par :**

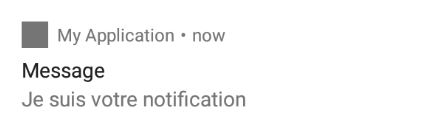
Le Titre de la Notification

Le Message de la notification.

L’icone de la notification

Le son de la notification.

Le Vibreur de la notification



* **Créer un moteur de notification** :

long[] vibrate = {0, 100, 200, 300};  
 Uri alarmSound = RingtoneManager.*getDefaultUri*(Notification.*DEFAULT\_SOUND*);  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(MainActivity.this,*CHANNEL\_ID*);  
 builder.setSmallIcon(R.drawable.*ic\_launcher\_background*);  
 builder.setContentTitle("Message");  
 builder .setContentText("Je suis votre notification");  
*/\* builder .setStyle(new NotificationCompat.BigTextStyle()  
 .bigText("Je suis une notification !"));\*/* builder .setPriority(*PRIORITY\_HIGH*);  
 builder .setSound(alarmSound);  
 builder .setVibrate(vibrate);

Ajouter la perimssion à AndroidManifest :

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

* **Créer un Gestionnaire de Notifications :**

NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(MainActivity.this);

* **Créer une Notification :**

notificationManager.notify(*notificationId*, builder.build());

Chaque Notification est caractérisée par un identifiant *notificationId ; ces notifications ont le meme contenu.*

*Exemple : Pour créer deux notifications de même contenu ; il suffit de les identifier par des notificationID différents.*

private static final int *notificationId* = 1;  
private static final int *notificationId1* = 2;

**Le code Globale est dans MainActivity :**

Button bt;  
 public static final String *CHANNEL\_1\_ID* = "channel1";  
 NotificationManagerCompat notificationManager;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 bt=(Button)findViewById(R.id.*button*);  
 notificationManager = NotificationManagerCompat.*from*(this);  
 bt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 long[] vibrate = {0, 100, 200, 300};  
 Uri alarmSound = RingtoneManager.*getDefaultUri*(Notification.*DEFAULT\_SOUND*);  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 NotificationChannel channel1 = new NotificationChannel(  
 *CHANNEL\_1\_ID*,  
 "Channel 1",  
 NotificationManager.*IMPORTANCE\_HIGH* );  
 channel1.setDescription("This is Channel 1");  
  
 String title = "1";  
 String message = "1";  
 Notification notification = new NotificationCompat.Builder(MainActivity.this, *CHANNEL\_1\_ID*)  
 .setSmallIcon(R.drawable.*ic\_launcher\_background*)  
 .setContentTitle(title)  
 .setContentText(message)  
 .setPriority(NotificationCompat.*PRIORITY\_HIGH*)  
 .setCategory(NotificationCompat.*CATEGORY\_MESSAGE*)  
 .setSound(alarmSound)  
 .setVibrate(vibrate)  
 .build();  
 notificationManager.notify(1, notification);  
  
 }  
  
 }  
 });

**Exemple 2** : Utilisation des méthodes :

* Créer la méthode createNotificationChannel()

private void createNotificationChannel() {  
 *// Create the NotificationChannel, but only on API 26+ because  
 // the NotificationChannel class is new and not in the support library* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 CharSequence name = "1";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel(*CHANNEL\_ID*, name,  
 importance);  
 channel.setDescription("description");  
 *// Register the channel with the system; you can't change the importance  
 // or other notification behaviors after this* NotificationManager notificationManager =  
 getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
}

* Créer la méthode SendNotification()

private void SendNotification() {  
 long[] vibrate = {0, 100, 200, 300};  
 Uri alarmSound = RingtoneManager.*getDefaultUri*(Notification.*DEFAULT\_SOUND*);  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(this,  
 *CHANNEL\_ID*).setSmallIcon(R.drawable.*ic\_launcher\_background*)  
 .setContentTitle("Ma notification")  
 .setContentText("")  
 .setStyle(new NotificationCompat.BigTextStyle()  
 .bigText("C'est une notification pour le test"))  
 .setPriority(*PRIORITY\_HIGH*)  
 .setSound(alarmSound)  
 .setVibrate(vibrate);  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(this);  
*// notificationId is a unique int for each notification that you must define* notificationManager.notify(*notificationId*, builder.build());  
 }

* Le code de Main\_activity :

private static final String *CHANNEL\_ID* = "1";  
private static final int *notificationId* = 1;

* Dans La méthode Oncreate() ajouter la méthode : createNotificationChannel();
* Dans la méthode du button on ajoute la méthode : SendNotification();

public void onClick(View v) {  
SendNotification();

}) ;

Exemple 3 :

* Créer une classe notifapp.java dérivée de android.app.Application dont le contenu est :

import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.os.Build;  
  
public class notifapp extends android.app.Application {  
 public static final String *CHANNEL\_1\_ID* = "channel1";  
 public static final String *CHANNEL\_2\_ID* = "channel2";  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 createNotificationChannels();  
 }  
  
 private void createNotificationChannels() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 NotificationChannel channel1 = new NotificationChannel(  
 *CHANNEL\_1\_ID*,  
 "Channel 1",  
 NotificationManager.*IMPORTANCE\_HIGH* );  
 channel1.setDescription("This is Channel 1");  
 NotificationChannel channel2 = new NotificationChannel(  
 *CHANNEL\_2\_ID*,  
 "Channel 2",  
 NotificationManager.*IMPORTANCE\_LOW* );  
 channel2.setDescription("This is Channel 2");  
 NotificationManager manager = getSystemService(NotificationManager.class);  
 manager.createNotificationChannel(channel1);  
 manager.createNotificationChannel(channel2);  
 }  
 }  
}

* Ajouter la ligne : android:name=".notifapp" à AndroidManifest dans la balise Application.
* Dans MainActivity on ajoute le code :

**Ajouter les Bibliothèques suivantes :**

import static com.example.notifcations.notifapp.*CHANNEL\_1\_ID*;  
import static com.example.notifcations.notifapp.*CHANNEL\_2\_ID*;

**Dans la mathode Oncreate() ajouter le code :**

notificationManager = NotificationManagerCompat.*from*(this);

et les méthodes :

public void sendOnChannel1() {  
 String title = "1";  
 String message = "1";  
 Notification notification = new NotificationCompat.Builder(this, *CHANNEL\_1\_ID*)  
 .setSmallIcon(R.drawable.*ic\_launcher\_background*)  
 .setContentTitle(title)  
 .setContentText(message)  
 .setPriority(NotificationCompat.*PRIORITY\_HIGH*)  
 .setCategory(NotificationCompat.*CATEGORY\_MESSAGE*)  
 .build();  
 notificationManager.notify(1, notification);  
}  
public void sendOnChannel2() {  
 String title = "2";  
 String message = "2";  
 Notification notification = new NotificationCompat.Builder(this, *CHANNEL\_2\_ID*)  
 .setSmallIcon(R.drawable.*ic\_launcher\_background*)  
 .setContentTitle(title)  
 .setContentText(message)  
 .setPriority(NotificationCompat.*PRIORITY\_LOW*)  
 .build();  
 notificationManager.notify(2, notification);  
}

Dans l’écouteur du bouton ajouter les méthodes :

public void onClick(View v) {  
 sendOnChannel1();  
 sendOnChannel2();  
}

Pour plus de détails voir les sites : <https://codinginflow.com/tutorials/android/notifications-notification-channels/part-1-notification-channels>

<https://blog.engineering.publicissapient.fr/2017/05/19/android-o-notification-channels/>

Pour supprimer une notification ou toutes les notifications on utilise :

bt1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 notificationManager.cancel(2);  
 }  
});  
bt2.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 notificationManager.cancelAll();  
 }  
});

Pour personnaliser le son d’une notification :

Ajouter le dossier raw au dossier res

Ajouter votre son (mp3) à ce dossier raw.

Ajouter la ligne comme suit à votre MainActivity :

Uri Sound=Uri.parse("android.resource://"+getPackageName()+"/raw/your\_sound");

Exemple 4 : Ouvir une activité à partir d’une Notification.

D’abort ajouter une activité à votre projet MainActivity2.java et son layout correcpondant AcitivityMain2.xml.

Dans MainActivity Ajouter le code suivant :

private static final String *CHANNEL\_ID* = "1";

public void onClick(View v) {  
 NotificationManager manager=(NotificationManager)  
 getApplicationContext().getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 Intent intent1=new Intent(MainActivity.this,MainActivity2.class);  
 intent1.putExtra("yes",true);  
 intent1.addFlags((Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*));  
 PendingIntent pi=PendingIntent.*getActivity*(MainActivity.this,0,intent1,PendingIntent.*FLAG\_ONE\_SHOT*);  
  
 Intent intent2=new Intent(MainActivity.this,MainActivity2.class);  
 intent2.putExtra("no",true);  
 intent2.addFlags((Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*|Intent.*FLAG\_ACTIVITY\_NEW\_TASK*));  
 PendingIntent pi2=PendingIntent.*getActivity*(MainActivity.this,1,intent2,PendingIntent.*FLAG\_ONE\_SHOT*);  
  
 Uri uri= RingtoneManager.*getDefaultUri*((RingtoneManager.*TYPE\_NOTIFICATION*));  
 NotificationCompat.Builder builder=new NotificationCompat.Builder(MainActivity.this,*CHANNEL\_ID*);  
 builder.setContentTitle("Request");  
 builder.setContentText("MmMM");  
 builder.setSmallIcon(R.drawable.*ic\_launcher\_background*);  
 builder.setSound(uri);  
 builder.setAutoCancel(true);  
 builder.setPriority(NotificationCompat.*PRIORITY\_HIGH*);  
 builder.addAction(R.drawable.*ic\_launcher\_background*,"yes",pi);  
 builder.addAction(R.drawable.*ic\_launcher\_background*,"no",pi2);  
 manager.notify(1,builder.build());  
 }  
});

Ajouter le code dans MainActivity2 :

tv=(TextView)findViewById(R.id.*textView*);  
  
NotificationManager manager=(NotificationManager)getApplicationContext()  
 .getSystemService(*NOTIFICATION\_SERVICE*);  
manager.cancelAll();  
  
if(getIntent().hasExtra("yes"))  
 tv.setText("Yes");  
else if(getIntent().hasExtra("no"))  
 tv.setText("not ok");

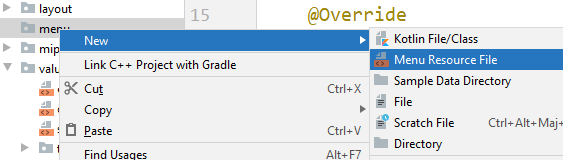
La layout ActivityMain.xml contient une TextView.

https://www.youtube.com/watch?v=GbqRwwT3Rg4&ab\_channel=AndroidCoding

# Les Menus sous android studio.

## Création d’un popuMenu :

* Ajouter un dossier menu à votre dossier res
* Ajouter un fichier menu.xml à votre dossier menu ;



Le contenu du fichier menu.xml est :

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 >  
 <item android:id="@+id/menu1"  
 android:title="Nouveau"  
 />  
 <item android:id="@+id/menu2"  
 android:title="Ouvrir" />  
 <item android:id="@+id/menu3"  
 android:title="Fermer" />  
</menu>

Le code pour afficher le menu en cliquant sur un bouton :

public void onClick(View v) {  
 PopupMenu popup = new PopupMenu(MainActivity.this, bt);  
 popup.inflate(R.menu.*menu*);  
 Menu menu = popup.getMenu();  
 popup.show();  
 }

Ajouter l’écouteur de popup à la méthode onClick() :

popup.show();  
popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener(){  
 @Override  
 public boolean onMenuItemClick(MenuItem item) {  
 switch(item.getItemId())  
 {  
 case R.id.*menu1*:  
 {  
 Toast.*makeText*(MainActivity.this,item.getTitle(),Toast.*LENGTH\_SHORT*).show();  
 break;  
 }  
 case R.id.*menu2*:  
 {  
 Toast.*makeText*(MainActivity.this,item.getTitle(),Toast.*LENGTH\_SHORT*).show();  
 break;  
 }  
 case R.id.*menu3*:  
 {  
 Toast.*makeText*(MainActivity.this,item.getTitle(),Toast.*LENGTH\_SHORT*).show();  
 break;  
 }  
 }  
 return false;  
 }  
});

En utilisant seuelement java on peut créer un popuMenu comme suit :

public void createPopupMenu(Context activity, View anchorView ) {

    // Create a PopupMenu and anchor it on a View.

    PopupMenu popupMenu = new PopupMenu(activity, anchorView);

    Menu menu = popupMenu.getMenu();

    // groupId, itemId, order, title

    MenuItem menuItemUpload = menu.add(1, 1, 1, "Upload");

    MenuItem menuItemBookmark = menu.add(2, 2, 2, "Bookmark");

    menuItemUpload.setIcon(R.drawable.icon\_upload);

    menuItemBookmark.setIcon(R.drawable.icon\_bookmark);

    // groupId, itemId, order, title

    SubMenu subMenuShare=  menu.addSubMenu(3, 3, 3, "Share");

    subMenuShare.setIcon(R.drawable.icon\_share);

    subMenuShare.add(4, 31, 1, "Google" );

    subMenuShare.add(5, 32, 2, "Instagram");

}

## Création d’un menu Text au niveau de actionBar :

En premier lieu ; on créer un fichier menu.xml dans le dossier menu :

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
 <item  
 android:title="Open"  
 />  
 <item  
 android:title="Share">  
 <menu>  
 <item android:title="Facebook" />  
 <item android:title="Instagram" />  
 </menu>  
 </item>  
  
</menu>

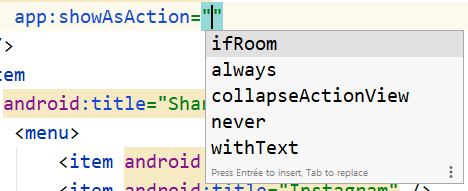
Dans MainActivty on ajoute la méthode abstraite:

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
 return super.onCreateOptionsMenu(menu);  
}

Qui contient le code :

@SuppressLint("RestrictedApi")  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
 MenuInflater inflater = getMenuInflater();  
 inflater.inflate(R.menu.*menu*, menu);  
 return super.onCreateOptionsMenu(menu);  
   
  
}

Pour afficher un menu dans l’actionBar, il suffit d’ajouter l’attribut : app:showAsAction="" qui aura les valeurs :



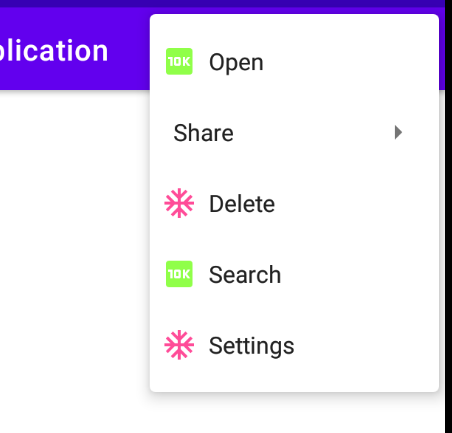
* ifRoom : affiche le menu si l’espace est disponible.
* always : le menu est toujour affiché en priorité.
* never : ne s’affiche jamais dans l’actionBar, et reste toujour dans le menu principal.

Ces valeurs sont reservées pour les menus texte.

## Création d’un menu avec Icones au niveau de l’actionBar :

Soit le menu menu.xml du dossier menu qui contient :

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
 <item  
 android:icon="@drawable/ic\_baseline\_10k\_24"  
 android:title="Open"  
 app:showAsAction="collapseActionView" />  
 <item  
 android:icon="@drawable/abc\_vector\_test"  
 android:title="Share">  
 <menu>  
 <item android:title="Facebook" />  
 <item android:title="Instagram" />  
 </menu>  
 </item>  
 <item  
 android:icon="@drawable/ic\_baseline\_ac\_unit\_24"  
 android:title="Delete" />  
  
 <item  
 android:id="@+id/app\_bar\_search"  
 android:icon="@drawable/ic\_launcher\_background"  
 android:title="Search"  
 />  
  
 <item  
 android:icon="@drawable/ic\_launcher\_background"  
 android:title="Settings" />  
  
</menu>



Pour les autres valeurs de showAsAction :

* collapseActionView :
* withText : On peut utiliser  **withText** ainsi que **always** ou **ifRoom** ; pour forcer le texte de l'article à apparaître à côté de l'icône lorsqu'il est affiché dans la barre d'action.

app:showAsAction="always|withText"

Remarque :

Le menu ActionBar s’affiche selon l’ordre dans le menu du fichier menu.xml ; pour changer leur ordre on utilise l’attribut :

android:orderInCategory="i"

où i est l’ordre de l’élement dans le menu.

Dans MainActivity ajouter le code suivant , pour gerer les événements sur les menus :

@Override  
public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 switch(item.getItemId())  
 {  
 case R.id.*menu1*:  
 {  
 break;  
 }  
 case R.id.*menu2*:  
 {  
 break;  
 }  
 case R.id.*menu3*:  
 {  
 break;  
 }  
 }  
 return super.onOptionsItemSelected(item);  
  
}

## Création d’un menu Contextuel :

Créer le fichier menu.xml dans le dossier menu :

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android">  
 <item  
 android:id="@+id/Ajout"  
 android:title="ajouter"  
 />  
 <item  
 android:id="@+id/modif"  
 android:title="Modifier"  
 />  
 <item  
 android:id="@+id/Supp"  
 android:title="Supprimer"  
 />  
</menu>

Dans MainActivity ajouter la méthode Abstraite et saisir son code :

@Override  
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo) {  
 super.onCreateContextMenu(menu, v, menuInfo);  
 MenuInflater menuInflater=getMenuInflater();  
 menuInflater.inflate(R.menu.*menu*,menu);  
  
}

Et la méthode pour activer les actions sur les menus :

@Override  
public boolean onContextItemSelected(@NonNull MenuItem item) {  
 switch (item.getItemId())  
 {  
 case R.id.*Ajout*:  
 {  
 break;  
 }  
 case R.id.*modif*:  
 {  
 break;  
 }  
 case R.id.*Supp*:  
 {  
 break;  
 }  
 }

On ajoute la méthode : registerForContextMenu(View view) (laisser vide)

@Override  
public void registerForContextMenu(View view) {  
 super.registerForContextMenu(view);  
  
}

En fin ; limiter la vue qui sera en relation avec le mennu contextuel ;

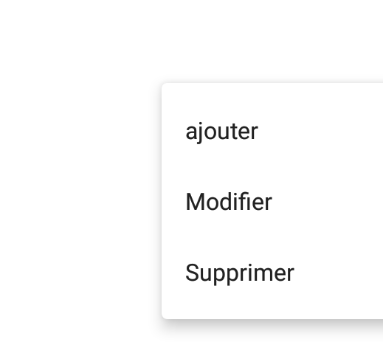
Exemple :

Menu contextule en cliquant sur une layout :

cc=(LinearLayout)findViewById(R.id.*cc*);  
registerForContextMenu(cc);

ce code sera ajouté à la méthode onCreate() de l’activité en question.

Le resultat est : (Click long ou touche longue) pour afficher le menu contextuel.



Fin du chapitre

**Les fragments sous android studio**

## Définition :

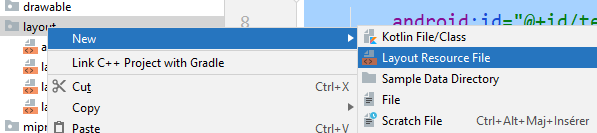
Un fragment est une sous activité qui peut être intégrée dans une activité; autrement Il permet de découpe un écran en plusieurs parties, plusieurs fragments. Cette sous activité est est héritée de la classe Fragment.



Chaque fragment a une durée de vie comme une activité.

## 2) Création de fragment :

* Ajouter une vue layout2:



Dont le contenu est :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="128dp"  
 android:background="#40F149"  
 android:text="Fragment1"  
 android:textSize="50sp" />  
</LinearLayout>

* Ajouter une vue layout3 qui contient:

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:scrollbarSize="50sp">  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="224dp"  
 android:background="#CF2E2E"  
 android:text="Fragment2"  
 android:textSize="50sp" />  
</LinearLayout>

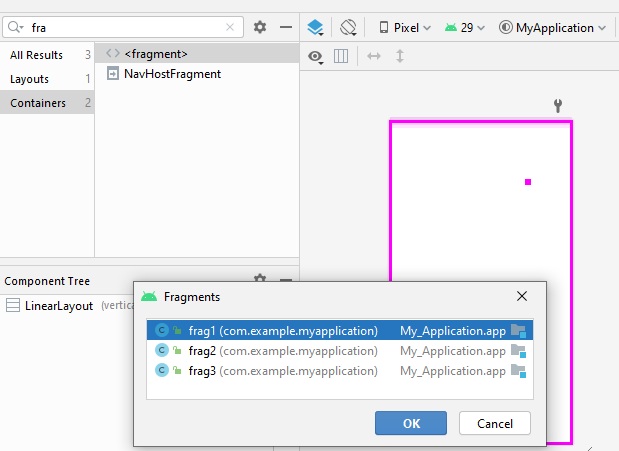
* Ajouter une classe frag1.java qui sera hérité de la classe Fragment et lui ajouter la méthode abstraite onCreateView().

public class frag1 extends Fragment {  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.*layout2*, container, false);  
 return view;  
 }  
  
  
}

* Ajouter une classe frag2.java qui sera hérité de la classe Fragment et lui ajouter la méthode abstraite onCreateView().

public class frag1 extends Fragment {  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.*layout3*, container, false);  
 return view;  
 }  
}

* Dans MainActivity ajouter les composants <fragment> de la pallette :



Et les lier aux fragments déjà crées (frag1, frag2).

Définir les attributs des fragments à partir de la fenêtre prorpiétés ou par code xml.

* Démarrer votre projet le résultat est comme ci-dessus les deux fragments sont intégrés dans MainActivity.

## 3) Utilisation de FrameLayout :

* Créer une layout layouttop qui contient un menu de deux boutons ,dont le contenu est :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
</LinearLayout>

* Créer une layout connexion qui contient :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:scrollbarSize="50sp">  
  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="42dp"  
 android:text="Utilisateur" />  
  
 <EditText  
 android:id="@+id/editTextTextPersonName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="47dp"  
 android:text="Mot de passe" />  
  
 <EditText  
 android:id="@+id/editTextTextPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="67dp"  
 android:ems="10"  
 android:inputType="textPassword" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Se connecter" />  
  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Quitter" />  
 </LinearLayout>  
</LinearLayout>

* Créer classe frag1 dérivé de la classe Fragment et Ajouter la méthode :

public class frag1 extends Fragment {  
Button bt1,bt2;  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
  
 View view = inflater.inflate(R.layout.layouttop, container, false);  
 bt1=(Button)view.findViewById(R.id.*button*);  
 bt2=(Button)view.findViewById(R.id.*button2*);  
 bt1.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 FragmentTransaction ft = getFragmentManager().beginTransaction();  
 ft.replace(R.id.*frame*, new frag2());  
 ft.commit();  
 }  
 });  
  
 return view;  
 }

* Créer classe frag2 dérivé de la classe Fragment et Ajouter la méthode :

public class frag2 extends Fragment {  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.*layout3*, container, false);  
 return view;  
 }

* Dans activity\_main.xml ajouter un fragment de la pallette et une Framelayout :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <fragment  
 android:id="@+id/fragment"  
 android:name="com.example.myapplication.frag1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="98dp" />  
  
 <FrameLayout  
 android:id="@+id/frame"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 </FrameLayout>  
</LinearLayout>

Pour les Boutons de connexion ; leur code sera dans frag2.java ; Voici un exemple :

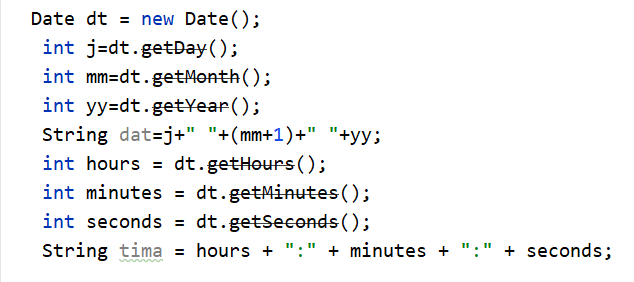
public class frag2 extends Fragment {  
 Button bt12;  
 @Nullable  
 @Override  
 public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {  
 View view = inflater.inflate(R.layout.*layout3*, container, false);  
 bt12=(Button)view.findViewById(R.id.*button3*);  
 bt12.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(getActivity(),"ok",Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
  
 return view;  
 }

Fin du chapitre

# Les classes Date et time

## Les classes Date et Time:

La classe Date est arrétée pour son utilisation sous android (deprecated api usage):



La classe Calender permet de récupérer la date et heure système :

* Pour récupérer la date système:

Calendar cc = Calendar.*getInstance*();  
int d=cc.get(Calendar.*DAY\_OF\_MONTH*);  
int m=cc.get(Calendar.*MONTH*);  
int y=cc.get(Calendar.*YEAR*);  
String datet=d+" "+(m+1)+" "+y;  
tv.setText(datet);

* Pour récupérer l’heure système:

Calendar c = Calendar.*getInstance*();  
int second = c.get(Calendar.*SECOND*);  
int minute = c.get(Calendar.*MINUTE*);  
int hour = c.get(Calendar.*HOUR*);  
String time = hour + ":" + minute + ":" + second;  
tv.setText(time);

* Pour récupérer une date Formatée :

SimpleDateFormat df = new SimpleDateFormat("dd-MMM-yyyy");  
 String date = df.format(c.getTime());  
tv.setText(date);

* Pour récupérer une heure Formatée :

SimpleDateFormat df = new SimpleDateFormat("HH:mm:ss");  
 String date = df.format(c.getTime());  
tv.setText(date);

## Différence de Dates :

**Exemple 1 :**

En utilisant la calsse Date :

public void onClick(View v) {  
 SimpleDateFormat Formatter = new SimpleDateFormat("dd/MM/yyyy");  
 try {  
 Date dateAvant = Formatter.parse("25/02/2012");  
 Date dateApres = Formatter.parse("31/03/2012");  
 long diff = dateApres.getTime() - dateAvant.getTime();  
 float res = (diff / (1000\*60\*60\*24));  
 tv1.setText("Nombre de jours entre les deux dates est: "+res);  
 }  
 catch(Exception e)  
 {  
 e.printStackTrace();  
 }  
  
}

## Comparaison entre deux dates  en utilisant compareTo():

SimpleDateFormat Formatter = new SimpleDateFormat("dd/MM/yyyy");  
 try {  
 Date dateAvant = Formatter.parse("25/02/2012");  
 Date dateApres = Formatter.parse("31/03/2012");  
 int x=dateApres.compareTo(dateAvant);  
 tv1.setText(""+x);  
 }  
 catch(Exception e)  
 {  
 tv1.setText("Erreur ");  
 }

La valeur retournée est -1 ou 1.

## Comparaison entre deux dates  en utilisant equals() ou ==:

**SimpleDateFormat Formatter = new SimpleDateFormat("dd/MM/yyyy");  
 try {  
 Date dateAvant = Formatter.parse("25/02/2012");  
 Date dateApres = Formatter.parse("31/03/2012");  
 boolean x=dateApres.equals(dateAvant);  
 tv1.setText(""+x);  
 }  
 catch(Exception e)  
 {  
 tv1.setText("Erreur ");  
 }**

La valeur retournée est : true ou false.

Oubien

Date dateAvant = Formatter.parse("25/06/2021");  
Date dateApres = Formatter.parse("31/03/2012");  
if(dateAvant==dateApres)  
tv1.setText("=");  
else  
 tv1.setText("!=");

## Comparaison entre deux dates  en utilisant before() ou after() :

SimpleDateFormat Formatter = new SimpleDateFormat(« dd/MM/yyyy ») ;  
 try {  
 Date dateAvant = Formatter.parse(« 25/02/2012 ») ;  
 Date dateApres = Formatter.parse(« 31/03/2012 ») ;  
 boolean x=dateApres.before(dateAvant) ;

*// boolean x=dateApres.after(dateAvant);*

tv1.setText(«  »+x) ;  
 }  
 catch(Exception e)  
 {  
 tv1.setText(« Erreur « ) ;  
 }

La valeur retournée est true ou false

**Remarque :**

On peut comparer les dates de deux calender sans recupérer leur date et les comparer.

Calendar call1=Calendar.*getInstance*();  
 Calendar call2=Calendar.*getInstance*();  
*// tv1.setText(call1.compareTo(call2));* boolean x=call1.before(call2);  
 boolean y=call1.after(call2);  
 tv1.setText(""+x);

On peut modifier la date d’un calender par :

SimpleDateFormat Formatter = new SimpleDateFormat("dd/MM/yyyy");  
try  
{  
 Date dateAvant = Formatter.parse("25/06/2021");  
 Calendar call1=Calendar.*getInstance*();  
 call1.setTime(dateAvant);  
 Calendar call2=Calendar.*getInstance*();  
 boolean y=call1.after(call2);  
 tv1.setText(""+y);  
}  
catch(Exception e)  
{  
 tv1.setText("Erreur");  
}

## Ajouter ou retrancher un nombre de jours,mois, ou Années à (ou d’) une date :

Pour les opérations sur les dates, Calendar dispose des constantes suivantes:

Calendar.DATE: ajouter/soustraire un jour  
Calendar.MONTH: ajouter/soustraire un mois  
Calendar.YEAR: ajouter/soustraire une année

Exemple 1:

Calendar cl=Calendar.*getInstance*();  
 cl.add(Calendar.*DAY\_OF\_MONTH*,1);  
cl.add(Calendar.*MONTH*,1);  
 cl.add(Calendar.*YEAR*,1);  
 Date dt = new Date();  
 dt=cl.getTime();  
 tv1.setText(""+dt);

Pour soudtraire un nombre de jours , mois ou année :

Calendar cl=Calendar.*getInstance*();  
 cl.add(Calendar.*DAY\_OF\_MONTH*,-1);  
cl.add(Calendar.*MONTH*,-1);  
 cl.add(Calendar.*YEAR*,-1);  
 Date dt = new Date();  
 dt=cl.getTime();  
 tv1.setText(""+dt);

## Le Timer :

Le timer est une classe qui permet d’executer une tache ou plusieurs taches régulieurement pendant un interval de temp ; il permet de programmer des tâches pour une exécution dans un *thread* d’arrière-plan.

La programmation du *timer* se fait par la méthode schedule(TimerTask task, long delay, long period) en lui indiquant en paramétres :

* task : la tâche à exécuter qui est un objet de type TimerTask
* delay : le délai en ms après lequel la tâche sera exécutée
* period : la période en ms de répétition de l’exécution de la tâche
* Création d’une tache :

La classe TimerTask permet de créer une tâche qui peut être planifiée pour une exécution unique.

Soit la méthode MaTask() qui permet d’executer la tache d’un compteur .

public void MaTask() {  
 myTimerTask = new TimerTask() {  
 @Override  
 public void run() {  
 count+=1;  
runOnUiThread(new Runnable() {  
 public void run() {  
  
 tv.setText(" "+count);  
 }  
 });  
 }  
 };  
 timer.schedule(myTimerTask, 1000, 1000);  
 }

Cette méthode contient la methode run() et et la méthode runOnUithread() qui vérifie le thread actuel et exécute le Runnableimmédiatement ; et exécuter du code dans un thread d'arrière-plan et vous devez mettre à jour vos vues à partir de ce thread d'arrière-plan.

Puis on fait appel au Timer (timer.schedule(myTimerTask, 1000, 1000);

Le bouton Démarrer contient le code dans mainActivity :

if(timer != null){  
 timer.cancel();  
 }  
  
 timer = new Timer();  
  
 MaTask();  
}

On n’oublie pas les déclarartions dans Main\_Activity.java.

TextView tv;  
 Button bt;  
 Timer timer;  
 TimerTask myTimerTask;  
int count=1;

Autre exemple : Insertion de la date synchronisée dans une application :

public class MainActivity extends AppCompatActivity {  
 TextView tv;  
 Button bt;  
 Timer timer;  
 TimerTask myTimerTask;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 tv = (TextView) findViewById(R.id.*textView*);  
 bt = (Button) findViewById(R.id.*button*);  
 bt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(timer != null) {  
 timer.cancel();  
 }  
  
 timer = new Timer();  
  
 MaTask();  
 }  
 });  
 }  
  
 public void MaTask() {  
 myTimerTask = new TimerTask() {  
 @Override  
 public void run() {  
  
 Calendar calendar = Calendar.*getInstance*();  
 SimpleDateFormat simpleDateFormat = new SimpleDateFormat("HH:mm:ss a");  
 final String strDate = simpleDateFormat.format(calendar.getTime());  
  
 runOnUiThread(new Runnable() {  
 public void run() {  
  
 tv.setText(strDate);  
 }  
 });  
 }  
 };  
 timer.schedule(myTimerTask, 1000, 1000);  
 }  
}

## ProgressBar :

### Définition :

Le ProgressBar est un weidget android qui permet de suivre la progression d’une action (Téléchargment, Uploading….) .

On a deux modes de ProgressBar :

### Spinning Wheel ProgressBar



### Horizontal ProgressBar



### Création d’un ProgressBar :

Dans la Pallete rechercher Progress et choisir l’un des modes :ProgressBar ou ProgressBar (Horizontal).

<ProgressBar  
 android:id="@+id/progressBar"  
 android:layout\_width="97dp"  
 android:layout\_height="wrap\_content"  
 android:indeterminate="true"  
 android:indeterminateTint="@color/cardview\_dark\_background"  
 android:indeterminateTintMode="src\_atop"  
  
 />  
  
<ProgressBar  
 android:id="@+id/progressBar2"  
 style="@android:style/Widget.ProgressBar.Horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="29dp"  
 android:max="100"  
 android:progress="40" />

### c)Les styles des ProgressBar :

Pour **Spinning Wheel ProgressBar** :

* style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Small"
* style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Small.Inverse"
* style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Large"
* style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Large.Inverse"
* style="@android:style/Widget.ProgressBar.Large"
* style="@android:style/Widget.ProgressBar.Large.Inverse"

Pour **Horizontal ProgressBar** :

* **style="?android:attr/progressBarStyleHorizontal"** (Default of Horizontal ProgressBar)
* **style="@android:style/Widget.DeviceDefault.Light.ProgressBar.Horizontal"**
* **style="@android:style/Widget.DeviceDefault.ProgressBar.Horizontal"**
* **style="@android:style/Widget.Holo.Light.ProgressBar.Horizontal"**
* **style="@android:style/Widget.Material.Light.ProgressBar.Horizontal"**
* **style="@android:style/Widget.Holo.ProgressBar.Horizontal"**
* **style="@android:style/Widget.Material.Light.ProgressBar.Horizontal"**
* **style="@android:style/Widget.ProgressBar.Horizontal"**

### Exemple: ProgressBar (Thread, Handler)

On peut relier le PorgressBar à un Timer Ou un thread :

Button bt;  
TextView tv1;  
private Handler handler = new Handler();  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 bt = (Button) findViewById(R.id.*button*);  
 ProgressBar progressbar = (ProgressBar) findViewById(R.id.*progressBar2*);  
  
 bt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 int MAX=100;  
  
 Thread thread = new Thread(new Runnable() {  
  
 @Override  
 public void run() {  
 handler.post(new Runnable() {  
 public void run() {  
 progressbar.setEnabled(false);  
 }  
 });  
 for( int i =0; i < MAX; i++) {  
 final int progress = i + 1;  
 *// Do something (Download, Upload, Update database,..)* SystemClock.*sleep*(1000); *// Sleep 20 milliseconds.  
  
 // Update interface.* handler.post(new Runnable() {  
 public void run() {  
 progressbar.setProgress(progress);  
 int percent = (progress \* 100) / MAX;  
  
 if(progress == MAX) {  
  
 progressbar.setEnabled(true);  
 }  
 }  
 });  
 }  
 }  
 });  
 thread.start();  
 }  
  
 });

Création de ProgressDialog :

private ProgressDialog pDialog;

**pDialog** = **new** ProgressDialog(MainActivity.**this**);  
 **pDialog**.setMessage(**"Please wait..."**);

**pDialog**.setTitle(**"Message"**);

**pDialog**.setCancelable(**false**);  
 **pDialog**.show();

# La classe Random

## Défintion :

C’est une Classe qui permet de généer des nombres aléatoires en utilisant ses méthodes ; Cette classe est plus puissante parce qu'elle offre une génération de plusieurs types pseudo-aléatoire: entier, float, double, byte, boolean

## Générer des nombres entiers :

La méthode **next(int n)** génère un nombre uniformément situé entre 0 et n-1 :

Random rnd=new Random();  
 int x= rnd.nextInt(11);  
tv1.setText(""+x);

L'exécution de ce code retourne un chiffre entre 0 et 10 choisi par le générateur.

### Générer des nombres entiers entre deux nombres :

  Exemple : afficher un nombre aléatoire entre x et y ;

int nbre=x+rnd.nextInt(y-x).

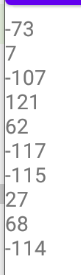
Exemple : x=10 et y=20.

Random rnd=new Random();  
 int x= 10+rnd.nextInt(10);  
tv1.setText(""+x);

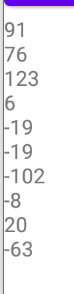
### Remplir un tableau par des nombres aléatoires :

La méthode nexBytes(bytes[]) permet de remplir un tableau de bytes aléatoires (entre -255 et 255):

String s="";  
Random rnd=new Random();  
byte[]t=new byte[10];  
 rnd.nextBytes(t);  
 for(int i=0;i<t.length;i++)  
 s=s+t[i]+"\n";  
 tv1.setText(s);



Remarque : le tableau peut avoir de nombres en répétition :



## Générer des Booléens :

La fonction nextBoolean() permet de retourner une valeur booleénne qui peut être true ou false :

Random rnd=new Random();  
boolean x=rnd.nextBoolean();  
 tv1.setText(""+x);

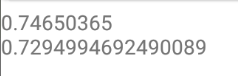
La valeur de x est : true ou false.

## Générer des nombres Float et double :

Les méthodes nextFloat() et nextDouble() permettent de retoruner un nombre entre 0.0 et 1.0 ; En java, le double est stocké sur 8 octets et le float sur 4 octets.

String s="";  
 Random rnd=new Random();  
float x=rnd.nextFloat();  
 double y=rnd.nextDouble();  
  
 tv1.setText(x+"\n"+y);

Exemple :



Pour générer un nombre réel entre deux nombres réels on utilise :

Random rnd=new Random();  
float h=rnd.nextFloat();  
Alors : 0< h <1.

0 <h(Max-min+1)< (Max-min+1)

min<h (Max-min+1)+min< (Max+1)

la valeur entre min et (max+1) est : h (max-min+1)+min.

float min=5;  
 float max=15;  
Random rnd=new Random();  
float h=rnd.nextFloat();  
float res=h\*(max-min+1)+min;  
*//le noùbre aléatoire sera entre 5 et 16.*tv1.setText(""+res);

## Utilisation de la méthode Math.random() :

La méthode random() est simple à utiliser, elle se trouve dans la classe [java.lang.Math](http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html) et retourne un double avec un signe positive supérieur à 0.0 et inférieur à 1.0. Le nombre généré est un double.

double x;  
x=Math.*random*();  
tv1.setText(""+x);

Pour obtenir un nombre aléatoire en 0 et un nombre y ; il suffit de multiplier x par y.

Exemple :

double x;  
x=2\*Math.*random*();  
tv1.setText(""+x);

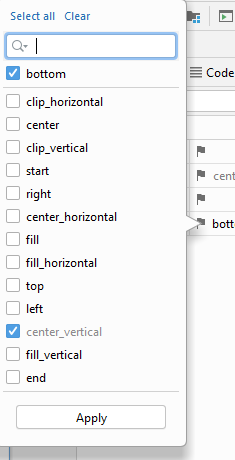
## FrameLayout :

FrameLayout est un conteneur qui peut contenir un ou plusieurs weidgets chevauchés Les vues sont placées les unes sur les autres dans l'ordre de leur ajout (les dernières vues ajoutées peuvent masquer les vues en fond).

**FrameLayout**a 9 zones de gravité (gravity) ; qui définissement la position d’un composants.



(Voir la fenêtre des attributs)



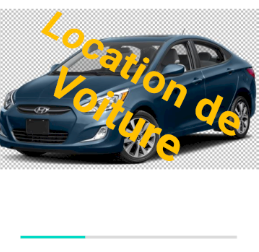
Pour changer la disposition en Z d’un composant, on utilise :

**android:translationZ="2dp"**

Ou par code :

**bt2**.bringToFront();

Exemple de FrameLayout :



# Autres layouts

## TableLayout :

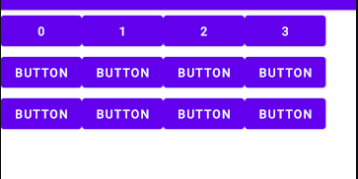
C’est un widget qui permet de définir un tableau dans lequel on ajoute des lignes (TableRow) qui peut contenir(Cellules) des vues ou des weidgets.

## Insertion d’un TableLayout :

Convertir Votre constraintLayout en TableLayout ; puis ajouter des TableRow à partir de la pallette.

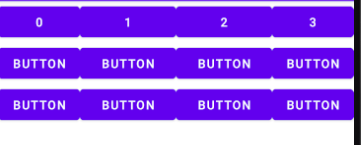
*<?*xml version="1.0" encoding="utf-8"*?>*<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".MainActivity" >  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 <TableRow  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
</TableLayout>

Dans**TableLayout**, les colonnes sont indexées 0, 1, 2 comme indiqué :



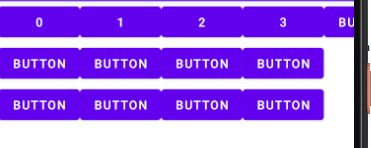
Parmis les attributs de TableLayout on a :

android:stretchColumns="1,2" : cet attribut permet de répartir l’espace vide sur la largment sur les cellules indiqué en valeur ; exemple : l’espace vide sera réparti sur les colonnes d’indices 1 et 2.

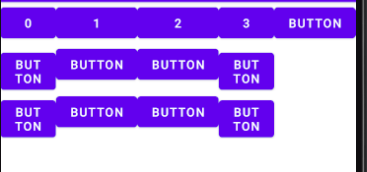


En utilisant android:stretchColumns="\*" ; cela indique que l’espace vide est réparti pour tous les colonnes.

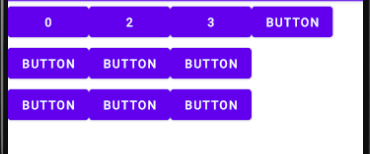
android:stretchColumns="0,3" : permet de passer une colonne en wrap\_content.



En ajoutant cet attribut on remarque que la colonne 4 s’affiche en totalité ;



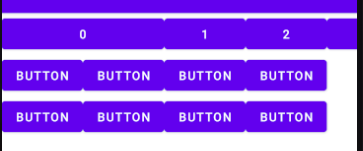
android:collapseColumns="1" : Permet de masquer une ou plusieurs colonnes .



Parmi les attributs des cases de TableRow on a :

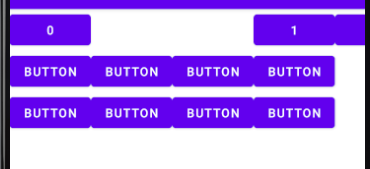
android:layout\_span="2" : Permet de fusionner deux colonnes ou plusieurs exmple:2.

<Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="0"  
 android:layout\_span="2"/>



android:layout\_column="3" : Permet de déplacer une cellule de sa position initiale evrs une nouvelle position. Exemple : le bouton d’indice 1 est déplacer vers l’indice 3.

<Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="1"  
 android:layout\_column="3"/>



<Space android:layout\_height="30dp"/>

Cette balise permet d’ajouter un espace entre les lignes :

</TableRow>  
<Space android:layout\_height="30dp"/>  
<TableRow

<View android:layout\_height="30dp"  
 android:background="@color/cardview\_dark\_background"/>

Cette balise permet permet ajouter une ligne.

</TableRow>  
 <View android:layout\_height="30dp"  
 android:background="@color/cardview\_dark\_background"/>  
<TableRow

# SnackBar :

C’est une barre d’ état qui s’affiche en bas de la vue ; elle n’existe pas dans les sdk , pour cela il faut l’implementer dans **Android SDK en ajouant la lige dans Gradle Script/build.gradle :**

dependencies **{** implementation 'com.google.android.material:material:1.0.0'  
 …………………………………….

}

Etape1:Concevoir l’activé activity\_main :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity"

android:id="@+id/base"

>  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
  
 android:text="Button" />  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
</LinearLayout>

Etape 2 : Le code Main\_Activité :

package com.example.snackbar;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.graphics.Color;  
import android.os.Build;  
import android.os.Bundle;  
import android.view.Gravity;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
  
import com.google.android.material.snackbar.Snackbar;  
  
public class MainActivity extends AppCompatActivity {  
 private View baseView;  
  
 private Button bt1;  
 private Button bt2;  
 private Button bt3;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 bt1=findViewById(R.id.*button1*);  
 bt2=findViewById(R.id.*button2*);  
 bt3=findViewById(R.id.*button3*);  
 baseView = this.findViewById(R.id.*constraintLayout*);  
 bt1.setOnClickListener(new View.OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
 showSnackbarDefault();  
 }  
  
  
 });  
 bt2.setOnClickListener(new View.OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
 showSnackbarActionCall();  
 }  
  
  
 });  
 bt3.setOnClickListener(new View.OnClickListener() {  
  
 @Override  
 public void onClick(View v) {  
 showSnackbarCustom();  
 }  
  
  
 });  
 }  
  
 private void showSnackbarCustom() {  
 Snackbar snackbar = Snackbar  
 .*make*(this.baseView, "Attention les données ne sont bien enregistrées!", Snackbar.*LENGTH\_LONG*)  
 .setAction("Ressayer", new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 }  
 });  
 snackbar.setActionTextColor(Color.*RED*);  
 View sbView = snackbar.getView();  
 TextView textView = sbView.findViewById(com.google.android.material.R.id.*snackbar\_text*);  
 textView.setTextColor(Color.*YELLOW*);  
 *// Align center.* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*){  
 textView.setTextAlignment(View.*TEXT\_ALIGNMENT\_CENTER*);  
 } else {  
 textView.setGravity(Gravity.*CENTER\_HORIZONTAL*);  
 }  
 *// Show Sneckbar* snackbar.show();  
 }  
  
 private void showSnackbarDefault() {  
 Snackbar snackbar = Snackbar  
 .*make*(this.baseView, "Message Envoyé!", Snackbar.*LENGTH\_LONG*);  
 snackbar.show();  
 }  
  
 private void showSnackbarActionCall() {  
 Snackbar snackbar = Snackbar  
 .*make*(this.baseView, "Message Enregistré", Snackbar.*LENGTH\_LONG*)  
 .setAction("Ok", new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 *// Show another Snackbar.* Snackbar snackbar1 = Snackbar.*make*(baseView, "Bien enregistré", Snackbar.*LENGTH\_SHORT*);  
 snackbar1.show();  
 }  
 });  
 snackbar.show();  
 }  
}

## NovigationBar :

On peut gréer une barre de navigation flottable qui permet d’afficher un menu vertical ; Pour cela il faut :

**Etape 1 :** Implementer les méthodes suivantes :

implementation 'com.google.android.material:material:1.2.0-alpha'  
implementation 'com.makeramen:roundedimageview:2.3.0'  
implementation 'androidx.navigation:navigation-fragment-ktx:2.2.0-rc03'  
implementation 'androidx.navigation:navigation-ui-ktx:2.2.0-rc03

(Syncroniser votre application)

**Etape 2:** Concevoir l’interface activity\_main.xml :

*<?*xml version="1.0" encoding="utf-8"*?>*<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 android:orientation="vertical">  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="?actionBarSize"  
 android:background="#FFFFFF"  
 android:backgroundTint="#4DBF7B"  
 android:orientation="vertical"  
 app:layout\_constraintTop\_toTopOf="parent"  
 >  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="40dp"  
 android:layout\_height="60dp"  
 app:srcCompat="@drawable/menu\_24" />  
  
  
 </LinearLayout>  
 </androidx.constraintlayout.widget.ConstraintLayout>  
 <com.google.android.material.navigation.NavigationView  
 android:id="@+id/nav"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start" />  
  
</androidx.drawerlayout.widget.DrawerLayout>

1. **Seekbar**

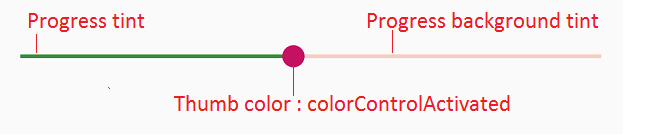
C’est curseur qui permet de choisir une valeur à la main entre deux valeurs min et max.



Le code Xml de seeckBar est :

<SeekBar  
 android:id="@+id/seekBar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="30dp"  
 android:layout\_marginTop="50dp"  
 android:layout\_marginBottom="50sp"  
 android:max="100"  
 android:progressTint="@color/red"  
 android:thumbTint="@color/red"  
 android:rotation="270"  
 android:background="#34A853"  
 />

Pour modifier la forme de seekBar :



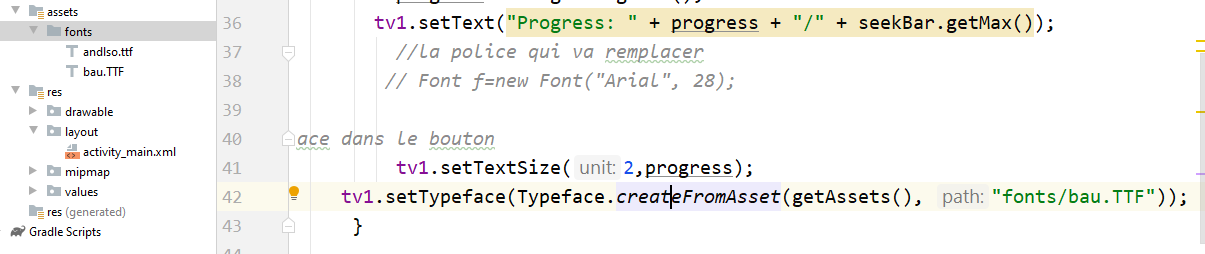
Le code dans Main\_Activity :

public class MainActivity extends AppCompatActivity {  
SeekBar sb;  
TextView tv1;  
Button bt;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 sb=findViewById(R.id.*seekBar*);  
 tv1=findViewById(R.id.*textView*);  
 bt=findViewById(R.id.*button*);  
 *//sb.setMax(100); par code* this.sb.getMax();  
 this.sb.setProgress(15);  
 int progress = 0;  
 this.tv1.setText("Progress: " + sb.getProgress() + "/" + sb.getMax());  
 *//ecouteur de seekBar* sb.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 progress = sb.getProgress();  
 tv1.setText("Progress: " + progress + "/" + seekBar.getMax());}  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
 }  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
 }  
 });  
 }

**Exemple**: changer la taille d’un texte avec SeekBar :

tv1.setTextSize(1,progress);

Pour changer les polices on utilise :



tv1.setTextSize(2,progress);  
tv1.setTypeface(Typeface.*createFromAsset*(getAssets(), "fonts/bau.TTF"));

tv1.setTypeface(null, Typeface.BOLD)

tv1.setTypeface(Typeface.SANS\_SERIF); //only font style tv1.setTypeface(null,Typeface.BOLD); //only text style(only bold) tv1.setTypeface(null,Typeface.BOLD\_ITALIC); //only text style(bold & italic) tv1.setTypeface(Typeface.SANS\_SERIF,Typeface.BOLD);

//font style & text style(only bold) tv1.setTypeface(Typeface.SANS\_SERIF,Typeface.BOLD\_ITALIC); //font style & text style(bold & italic)

## TabHost :

C’est un composant qui permet d’intégrer des onglets dans une application Mobile :

Etape1 :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity" >  
  
 <TabHost  
 android:id="@+id/tabhost"  
 android:layout\_width="match\_parent"  
 android:layout\_height="40dp"  
 >  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <TabWidget  
 android:id="@android:id/tabs"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="#35D8D8"  
 android:clipChildren="true"  
 android:foregroundTint="#FDD835"  
 android:layoutMode="opticalBounds" />  
  
 <FrameLayout  
 android:id="@android:id/tabcontent"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:id="@+id/tab1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/textv1"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Tabl1"  
 android:textSize="50sp" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/tab2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
 <TextView  
 android:id="@+id/textv2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Tabl2"/>  
 </LinearLayout>  
  
 <LinearLayout  
 android:id="@+id/tab3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
 <TextView  
 android:id="@+id/textv3"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Tabl3"/>  
  
 </LinearLayout>  
 </FrameLayout>  
 </LinearLayout>  
 </TabHost>  
</LinearLayout>

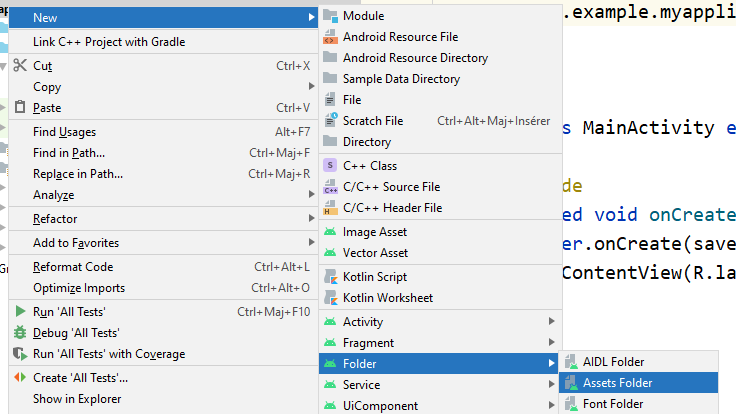
Le code de Main\_Activity :

import androidx.appcompat.app.AppCompatActivity;  
  
import android.annotation.SuppressLint;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.widget.TabHost;  
import android.widget.TabWidget;  
import android.widget.TextView;  
  
public class MainActivity extends AppCompatActivity {  
 TabHost tabHost;  
 TabWidget tw1;  
 @SuppressLint("ResourceType")  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
tw1=findViewById(android.R.id.*tabs*);  
 tabHost = (TabHost) findViewById(R.id.*tabhost*);  
 tabHost.setup();  
  
 *// tw1.setBackgroundColor(Color.YELLOW);*tabHost.setOnTabChangedListener(new TabHost.OnTabChangeListener() {  
 @Override  
 public void onTabChanged(String tabId) {  
 if(tabHost.getCurrentTab()==0)  
 {  
 initailiser();  
 tabHost.getTabWidget().getChildAt(tabHost.getCurrentTab()).setBackgroundColor(Color.*RED*); *//1st tab selected* }  
 if(tabHost.getCurrentTab()==1)  
 {  
 initailiser();  
 tabHost.getTabWidget().getChildAt(tabHost.getCurrentTab()).setBackgroundColor(Color.*RED*); *//1st tab selected* }  
  
 if(tabHost.getCurrentTab()==2)  
 {  
 initailiser();  
 tabHost.getTabWidget().getChildAt(tabHost.getCurrentTab()).setBackgroundColor(Color.*RED*); *//1st tab selected* }  
  
  
  
  
 }  
});  
  
  
*//tab1* TabHost.TabSpec spc=tabHost.newTabSpec("Tab One");  
 spc.setContent(R.id.*tab1*);  
 *//image aulieu de texte dans tab  
 //spc.setIndicator("",getResources().getDrawable(R.drawable.ic\_android\_black\_24dp));* spc.setIndicator("إعدادات");  
 tabHost.addTab(spc);  
 *//tab2* spc=tabHost.newTabSpec("Tab Two");  
 spc.setContent(R.id.*tab2*);  
 spc.setIndicator("Tab Two");  
 tabHost.addTab(spc);  
 *//tab3* spc=tabHost.newTabSpec("Tab Tree");  
 spc.setContent(R.id.*tab3*);  
 spc.setIndicator("Tab Tree");  
 tabHost.addTab(spc);  
 }  
 public void initailiser()  
 {  
 for(int i=0;i<tabHost.getTabWidget().getChildCount();i++)  
 {  
 tabHost.getTabWidget().getChildAt(i).setBackgroundColor(Color.*YELLOW*); *//unselected* }  
  
 }  
  
}

Accès aux Fichiers Json, xml et txt sous android

## Création d’un fichier Json :

* Créer un dossier Assets :



* Dans ce dossier Ajouter un fichier Liste.json dont le contenu est :

[  
 {"code":"1","prenom":"Tom", "nom":"Jackson"},  
 {"code":"2","prenom":"Linda", "nom":"Garner"},  
 {"code":"3","prenom":"Adam", "nom":"Cooper"}  
]

* Dans Main\_Activity on ajoute le code suivant :

TextView tv1;  
 JSONArray jsonArray;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 **tv1=(TextView)findViewById(R.id.*textView*);  
 String fileName="liste.json";  
 try {  
 String s="";  
 InputStream is = getResources().getAssets().open(fileName);  
 int size = is.available();  
 byte[] data = new byte[size];** is.read(data);  
 is.close();  
 String json = new String(data, "UTF-8");  
 jsonArray=new JSONArray(json);  
 if (jsonArray != null) {  
 for (int i = 0; i < jsonArray.length(); i++) {  
 s=s+jsonArray.getJSONObject(i).getString("code");  
  
 }  
 }  
tv1.setText(s);  
 }  
 catch (Exception e)  
 {  
 }

## Accès au Fichier XML:

* Créer le dossier Assets et dans ce dossier créer le fichier boutique.xml.

*<?*xml version= "1.0" encoding="UTF-8" standalone= "no"*?>*<boutique>  
 <telephone>  
 <marque>Samsung</marque>  
 <modele>Galaxy S3</modele>  
 </telephone>  
 <telephone>  
 <marque>Apple</marque>  
 <modele>iPhone 4</modele>  
 </telephone>  
 <telephone>  
 <marque>Nokia</marque>  
 <modele>Lumia 800</modele>  
 </telephone>  
</boutique>

* Dans activity\_main ajouter un TextViewet un bouton pour afficher les données du fichier xml.

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="TextView" />  
</LinearLayout>

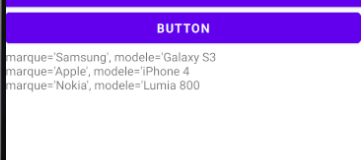
* Créer une classe telephon qui contient les attributs marque et modele :

public class telephon {  
 public String marque;  
 public String modele;  
  
 @Override  
 public String toString() {  
 return "marque='" + marque + '\'' +", modele='" + modele + "\n" ;  
 }  
}

* Dans Main\_Activity on ajoute le code suivant :

TextView tv1;  
Button bt;  
 ArrayList<telephon> liste;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 tv1 = (TextView) findViewById(R.id.*textView*);  
 bt=(Button)findViewById(R.id.*button*);  
 bt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 try {  
 XmlPullParserFactory xpf=XmlPullParserFactory.*newInstance*();  
 XmlPullParser parser=xpf.newPullParser();  
 InputStream is = getAssets().open("boutique.xml");  
 parser.setFeature(XmlPullParser.*FEATURE\_PROCESS\_NAMESPACES*,false);  
 parser.setInput(is,null);  
 processingparser(parser);  
  
 } catch (IOException | XmlPullParserException e) {  
 e.printStackTrace();  
 }  
 }  
 });  
 }  
 public void processingparser(XmlPullParser parser) throws XmlPullParserException, IOException {  
 liste=new ArrayList();  
 int eventype=parser.getEventType();  
 telephon tel=null;  
 while (eventype!=XmlPullParser.*END\_DOCUMENT*)  
 {  
 String tag =null;  
 switch (eventype)  
 {  
 case XmlPullParser.*START\_TAG*:  
 {  
 tag= parser.getName();  
 if("telephone".equals(tag))  
 {  
 tel=new telephon();  
 liste.add(tel);  
 }  
 else if(tel!=null)  
 {  
 if("marque".equals(tag))  
 {  
 tel.marque=parser.nextText();  
 }  
 else if("modele".equals(tag))  
 {  
 tel.modele=parser.nextText();  
 }  
 }  
 }  
 break;  
 }  
 eventype=parser.next();  
 }  
 afficher();  
 }  
 public void afficher()  
 {  
 String s="";  
 for(telephon tel:liste)  
 {  
 s=s+tel.toString();  
 }  
 tv1.setText(s);  
 }}

Le résultat d’execution est :



## Accès au Fichier txt :

* Ajouter les permissions en lecture et écriture à AndroidManifest :

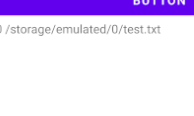
<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>

* Créer un fichier txt :

try {  
 FileOutputStream fOut = openFileOutput("test.txt", *MODE\_PRIVATE*);  
  
} catch (FileNotFoundException e) {  
 tv1.setText("n'existe pas");  
}

Le fichier test.txt est crée dans le chemin par défaut de l’application de l’emulateur ou du device.



Pour en récuprér utiliser le code :

Environment.*getExternalStorageDirectory*() + "/test.txt"

* Ecrire dans un fichier txt :

FileOutputStream fOut = null;  
try {  
 fOut = openFileOutput("test.txt", *MODE\_PRIVATE*);  
} catch (FileNotFoundException e) {  
 e.printStackTrace();  
}  
final String ch = new String("1000");  
OutputStreamWriter sr = new OutputStreamWriter(fOut);  
*// Write the string to the file*try {  
 sr.write(ch);  
 sr.close();  
} catch (IOException e) {  
 e.printStackTrace();  
}

* Lire le contenu d’un fichier txt :

*//la lecture du fichier test.txt*FileInputStream fIn = null;  
try {  
 fIn = openFileInput("test.txt");  
} catch (FileNotFoundException e) {  
 e.printStackTrace();  
}  
InputStreamReader isr = new InputStreamReader(fIn);  
char[] inputBuffer = new char[100];  
  
try {  
 isr.read(inputBuffer);  
} catch (IOException e) {  
 e.printStackTrace();  
}  
String readString = new String(inputBuffer);  
  
tv1.setText(readString+" "+ Environment.*getExternalStorageDirectory*() + "/test.txt");

## Accès au Fichier Json sur Internet :

* Concevoir l’interface suivante :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity" >  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Button" />  
  
 <TextView  
 android:id="@+id/textView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:text="TextView" />  
  
</LinearLayout>

* Ajouter la permission sur AndroidManifest ;

**<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />**

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

* Tester la connexion :

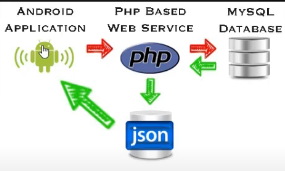
ConnectivityManager connManager =  
 (ConnectivityManager) getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
NetworkInfo networkInfo = connManager.getActiveNetworkInfo();  
if (networkInfo == null) {  
 tv1.setText("Non connecté");  
}  
 else  
 tv1.setText("Connecté");  
}

* Création de la classe feshData qui derivé de la classe AsyncTask ; elle contient les méthodes abstraites :

package com.example.myapplicationjson ;  
  
import android.app.ProgressDialog;  
import android.content.Context;  
import android.os.AsyncTask;  
  
import org.json.JSONArray;  
import org.json.JSONException;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.InputStreamReader;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.URL;  
  
public class feshData extends AsyncTask<Void, Void, Void> {  
String data="",dataParsed="",singleParsed="";  
 ProgressDialog pDialog;  
 @Override  
 protected Void doInBackground(Void... voids) {  
  
 try {  
 URL url = new URL("https://jsonplaceholder.typicode.com/posts");  
 HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();  
 InputStream inputStream = httpURLConnection.getInputStream();  
 BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));  
 String line = "";  
 while (line != null) {  
 line = bufferedReader.readLine();  
 data = data + line;  
 }  
  
  
 JSONArray JA = new JSONArray(data);  
 for (int i = 0; i < JA.length(); i++) {  
 JSONObject JO = (JSONObject) JA.get(i);  
 singleParsed = "userId:" + JO.get("userId") + "\n" +  
 "id:" + JO.get("id") + "\n";  
 dataParsed =dataParsed+singleParsed;  
 }  
  
 } catch (MalformedURLException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 } catch (JSONException jsonException) {  
 jsonException.printStackTrace();  
 }  
 return null;  
  
 }  
  
 @Override  
 protected void onPreExecute() {  
 super.onPreExecute();  
 }  
  
 @Override  
 protected void onPostExecute(Void aVoid) {  
 super.onPostExecute(aVoid);  
MainActivity.*tv*.setText(this.dataParsed);  
 }  
}

* Ajouter le code suivant à Main\_Activity :

package com.example.myapplicationjson;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
  
public class MainActivity extends AppCompatActivity {  
public static TextView *tv*;  
Button bt;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 *tv*=findViewById(R.id.*textView2*);  
 bt=findViewById(R.id.*button*);  
 bt.setOnClickListener(new View.OnClickListener()  
 {  
 @Override  
 public void onClick(View v) {  
feshData proc=new feshData();  
proc.execute();  
 }  
 });  
 }  
}



<https://api.npoint.io/b897b80db995f9cb6fb8>

<https://graph.facebook.com/me>

SQLITE Sous android studio

## Définition :

SQILTE est un Système de Gestion de Base de Données Relationnelle qui a la particularité de fonctionner sans serveur « StandAlone ». SQLite utilise donc la plus part des commandes SQL (CREATE, INSERT, UPDATE, DELETE, et SELECT).

## Création de l’interface de Gestion de Données d’une base de données.

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity" >  
  
 <TextView  
 android:id="@+id/textView4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Nom" />  
  
 <EditText  
 android:id="@+id/nom"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <TextView  
 android:id="@+id/textView5"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Prénom:" />  
  
 <EditText  
 android:id="@+id/prenom"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <TextView  
 android:id="@+id/textView7"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Age" />  
  
 <EditText  
 android:id="@+id/age"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:onClick="debut"  
 android:text="|&lt;" />  
  
 <Button  
 android:id="@+id/button5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:onClick="precedent"  
 android:text="&lt;" />  
  
 <Button  
 android:id="@+id/button6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:onClick="suivant"  
 android:text="&gt;" />  
  
 <Button  
 android:id="@+id/button7"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:onClick="fin"  
 android:text="&gt;|" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Ajouter" />  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Afficher" />  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Supprimer" />  
  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/textView10"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Résultat" />  
  
</LinearLayout>

## Méthode de création d’une base de données et Gestion des cette base de données :

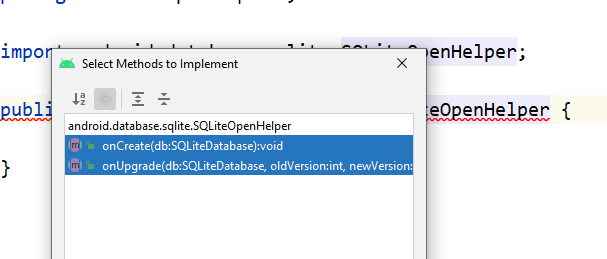
**Etape 0** : Création de l’interface Layout activity\_main :

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout 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"  
 android:orientation="vertical"  
 tools:context=".MainActivity" >  
  
 <Button  
 android:id="@+id/button8"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Se Connecter"  
 />  
  
 <TextView  
 android:id="@+id/textView4"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Nom" />  
  
 <EditText  
 android:id="@+id/nom"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <TextView  
 android:id="@+id/textView5"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Prénom:" />  
  
 <EditText  
 android:id="@+id/prenom"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <TextView  
 android:id="@+id/textView7"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Age" />  
  
 <EditText  
 android:id="@+id/age"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:ems="10"  
 android:inputType="textPersonName" />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
  
 android:text="|&lt;" />  
  
 <Button  
 android:id="@+id/button5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
  
 android:text="&lt;" />  
  
 <Button  
 android:id="@+id/button6"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
  
 android:text="&gt;" />  
  
 <Button  
 android:id="@+id/button7"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
  
 android:text="&gt;|" />  
  
 <Button  
 android:id="@+id/button9"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Pos" />  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Ajouter" />  
  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Afficher" />  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Supprimer" />  
 <Button  
 android:id="@+id/button15"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Modifier" />  
  
 </LinearLayout>  
  
 <TextView  
 android:id="@+id/textView10"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Résultat" />  
  
</LinearLayout>

**Etape 1 :** Créer une classe stagiaire (num, nom, prenom, age) dont les attributs sont Privés et générer un constructeur par défaut et un autre d’initialisation, et les accesseurs.

package com.example.myapplicationsqlite;  
  
public class Stagiaire {  
 int num, age;  
 String nom, prenom;  
  
 public Stagiaire(int num, String nom, String prenom,int age) {  
 this.num = num;  
 this.age = age;  
 this.nom = nom;  
 this.prenom = prenom;  
 }  
  
 public int getNum() {  
 return num;  
 }  
  
 public void setNum(int num) {  
 this.num = num;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public String getNom() {  
 return nom;  
 }  
  
 public void setNom(String nom) {  
 this.nom = nom;  
 }  
  
 public String getPrenom() {  
 return prenom;  
 }  
  
 public void setPrenom(String prenom) {  
 this.prenom = prenom;  
 }  
  
 @Override  
 public String toString() {  
 return  
 "num=" + num +  
 ", age=" + age +  
 ", nom='" + nom + '\'' +  
 ", prenom='" + prenom ;  
 }  
}

**Etape 2 :** Création d’un classe appelée **Sqlitehelper** dérivée de la classe **SQLiteOpenHelper** et implémentation des méthodes OnCreate() et Upgrade() et générer les constructeurs (2premiers seulement):



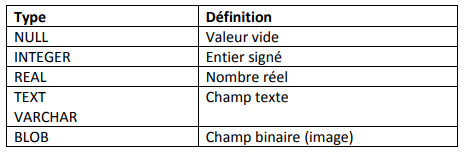
* ***onCreate()*** – Dans cette méthode on va créer des instructions des tables, lorsque la base de données est créée.
* ***onUpgrade()*** – Résérvée pour la mise à niveau de la base de données, comme la modification de la structure de la table, l'ajout de contraintes à la base de données, etc.

public class Stag\_helper extends SQLiteOpenHelper {  
 public Stag\_helper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {  
 super(context, name, factory, version);  
 }  
  
 public Stag\_helper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version, @Nullable DatabaseErrorHandler errorHandler) {  
 super(context, name, factory, version, errorHandler);  
 }  
  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 String req="create table stagiaire(num integer,nom text,prenom text,age integer );";  
 db.execSQL(req);  
  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("drop table if exists stagiaire");  
 onCreate(db);  
  
 }  
}

Dans Cette classe(Stag\_helper) on déclare les informations de la base de données :

private static final String *DATABASE\_NAME*="data";  
private static final int *DATABASE\_VERSION*=1;

on rappelle que la déclaration des champs d’une table se fait selon les types suivants:



Remarque :

Utiliser le système Db Browser(sqlite) pour téster votre table et vos requêtes.

On déclare : SQLiteDatabase bd; Et on crée une table Stagiaire dans la méthode Oncreate():

String req="create table stagiaire(num integer primary key autoincrement,nom text,prenom text,age integer );";  
db.execSQL(req);

Et dans la méthode onUpgrade Ecrire le code suivant :

db.execSQL("drop table if exists stagiaire");  
onCreate(db);

**Etape 3 :** Dans cette étape on va créer la base de données et on va vérifier s’elle existe :

Dans Main\_Activity , ajouter le code suivant dans l’écouteur d’un bouton Se connecter :

public void seconnecter(View view) {  
 bd = openOrCreateDatabase("data.db", Context.*MODE\_PRIVATE*,null);  
 if (bd== null)  
 Toast.*makeText*(getApplicationContext(), "La BD n'existe pas",Toast.*LENGTH\_LONG*).show();  
 else  
 Toast.*makeText*(getApplicationContext(), "La BD existe ",Toast.*LENGTH\_LONG*).show();  
  
  
}

Etape 4 : Créer la classe GestionDB.java ; cette classe permet la gestion des données de la base de données.

package com.example.myapplicationsqlite;  
  
import android.content.ContentValues;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.widget.Toast;  
  
import java.security.PublicKey;  
  
public class GestionDB {  
 Stag\_helper helper;  
 private SQLiteDatabase db;  
 private final String nom\_table="stagiaire";  
 String s="";  
 SQLiteDatabase bd;  
 int num=0;  
 public GestionDB(Stag\_helper helper) {  
 this.helper = helper;  
 db=helper.getWritableDatabase();  
 }  
 public long ajouterstag(Stagiaire s){  
 ContentValues values=new ContentValues();  
 values.put("num",s.getNum());  
 values.put("nom",s.getNom());  
 values.put("prenom",s.getPrenom());  
 values.put("age",s.getAge());  
  
 long idp= db.insert("stagiaire",null,values);  
return idp;  
 }  
 public String listerstag()  
 {  
 Cursor cursor=db.query(nom\_table,new String[]{"num","nom","prenom","age"},null,null,null,null,"num desc",null);*//rawQuery("select \* from produit",null);//* cursor.moveToFirst();  
 while (!cursor.isAfterLast()){  
 long num=cursor.getLong(0);  
 String nom=cursor.getString(1);  
 String prenom=cursor.getString(2);  
 int age=cursor.getInt(3);  
 s=s+" "+num+" "+nom+" "+prenom+age+"\n";  
 cursor.moveToNext();  
 }  
 return s;  
  
 }  
 public long supprimer(int code)  
 {  
 long id= db.delete("stagiaire"," num=?",new String[]{Integer.*toString*(code)});  
 return id;  
  
 }  
 public long modifier(int code,Stagiaire s)  
 {  
 ContentValues values=new ContentValues();  
 values.put("nom",s.getNom());  
 values.put("prenom",s.getPrenom());  
 values.put("age",s.getAge());  
 long idp= db.update("stagiaire",values,"num=?",new String[]{String.*valueOf*(code)});  
 return idp;  
 }  
  
}

Etape 5 : Codifiaction de Main\_Activity :

package com.example.myapplicationsqlite;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
public class MainActivity extends AppCompatActivity {  
 Button btcon,btaj,btmod,supp,btaff;  
 Button btfirst,btnext,btprev,btlast,pos;  
 EditText ed1, ed2,ed3;  
 Stag\_helper help;  
 TextView tv;  
 SQLiteDatabase bd;  
 Cursor c1;  
 private GestionDB data;  
 int num=0;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 tv=(TextView)findViewById(R.id.*textView10*) ;  
 btcon=(Button)findViewById(R.id.*button8*);  
 btaj=(Button)findViewById(R.id.*button2*);  
 supp=(Button)findViewById(R.id.*button4*);  
 btaff=(Button)findViewById(R.id.*button3*);  
 btfirst=(Button)findViewById(R.id.*button*);  
 btnext=(Button)findViewById(R.id.*button6*);  
 btprev=(Button)findViewById(R.id.*button5*);  
 btlast=(Button)findViewById(R.id.*button7*);  
 btmod=(Button)findViewById(R.id.*button15*);  
 pos=(Button)findViewById(R.id.*button9*);  
 ed1=(EditText)findViewById(R.id.*nom*);  
 ed2=(EditText)findViewById(R.id.*prenom*);  
 ed3=(EditText)findViewById(R.id.*age*);  
 bd = openOrCreateDatabase("data.db", Context.*MODE\_PRIVATE*,null);  
 help = new Stag\_helper(this, "data.db", null, 1);  
 *//Tester la connexion* btcon.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 if (bd== null)  
 Toast.*makeText*(getApplicationContext(), "La BD n'existe pas",Toast.*LENGTH\_LONG*).show();  
 else  
 Toast.*makeText*(getApplicationContext(), "La BD existe ",Toast.*LENGTH\_LONG*).show();  
 }  
 });  
 *//Ajouter* btaj.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 num=num+1;  
 String nom = ed1.getText().toString();  
 String prenom = ed2.getText().toString();  
 String age1 = ed3.getText().toString();  
 int age = Integer.*parseInt*(age1);  
 Stagiaire s=new Stagiaire(num,nom, prenom, age);  
 data= new GestionDB(help);  
 long idp=data.ajouterstag(s);  
 Toast.*makeText*(MainActivity.this," "+idp,Toast.*LENGTH\_LONG*).show();  
  
  
 }  
 });  
 *//Supprimer* supp.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 int code=3;  
 long idp=data.supprimer(code);  
 Toast.*makeText*(MainActivity.this," "+idp,Toast.*LENGTH\_LONG*).show();  
  
 }  
 });  
 *// Modifier* btmod.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 int numa=2;  
 String nom = ed1.getText().toString();  
 String prenom = ed2.getText().toString();  
 String age1 = ed3.getText().toString();  
 int age = Integer.*parseInt*(age1);  
 Stagiaire s=new Stagiaire(numa,nom, prenom, age);  
 long ipd=data.modifier(2,s);  
 String ss=data.listerstag();  
 tv.setText(ss);  
 }  
 });  
  
 *//Afficher* btaff.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 String s=data.listerstag();  
 tv.setText(s);  
 }  
 });  
*// first* btfirst.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 c1 = bd.rawQuery("SELECT \* FROM stagiaire", null);  
 if(c1.getCount()==0)  
 AfficheMessage();  
else  
 if(c1.moveToFirst())  
 {  
 ed1.setText(c1.getString(1));  
 ed2.setText(c1.getString(2));  
 ed3.setText(c1.getString(3));  
 }  
  
  
 }  
 });  
 btnext.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 if(c1.moveToNext())  
 {  
 ed1.setText(c1.getString(1));  
 ed2.setText(c1.getString(2));  
 ed3.setText(c1.getString(3));  
 }  
 else  
 Toast.*makeText*(MainActivity.this,"Fin de la liste",Toast.*LENGTH\_SHORT*).show();  
 }  
 });  
 btlast.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 if(c1.moveToLast())  
 {  
 ed1.setText(c1.getString(1));  
 ed2.setText(c1.getString(2));  
 ed3.setText(c1.getString(3));  
 }  
  
 }  
 });  
 btprev.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 data= new GestionDB(help);  
 if(c1.moveToPrevious())  
 {  
 ed1.setText(c1.getString(1));  
 ed2.setText(c1.getString(2));  
 ed3.setText(c1.getString(3));  
 }  
 else  
 Toast.*makeText*(MainActivity.this,"Début de la liste",Toast.*LENGTH\_SHORT*).show();  
  
 }  
 });  
 pos.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 int p=5;  
 c1.moveToPosition(p);  
 ed1.setText(c1.getString(1));  
 ed2.setText(c1.getString(2));  
 ed3.setText(c1.getString(3));  
 }  
 });  
  
 }  
 public void AfficheMessage()  
 {  
 AlertDialog.Builder builder=new AlertDialog.Builder(this);  
 builder.setCancelable(true);  
 builder.setTitle("Message");  
 builder.setMessage("La Requete est Vide");  
 builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
 });  
 builder.show();  
 }  
  
}

Pour récupérer le contenu du curseur on utilise les méthodes suivantes:

* moveToFirst () , se place au début du curseur
* moveToNext (), parcours en avant move
* ToPrevious() parcours en arrière
* moveToLast() se place à la fin du curseur

file:///C:/Users/AMIMI/Desktop/Nouveau%20dossier/SqliteExemple.pdf

<https://abhiandroid.com/database/sqlite>

btnAjouter.setOnClickListener(this);

btnSupprimer.setOnClickListener(this);

btnAffichertout.setOnClickListener(this);

btnSupprimer.setOnClickListener(this);

btnAfficher.setOnClickListener(this);

btnSuivant.setOnClickListener(this);

btnPrecdent.setOnClickListener(this);

btnPremier.setOnClickListener(this);

public void onClick(View view) {

if(view==btnAjouter)

{

}

}

<https://abhiandroid.com/programming/activity-life-cycle>

# Animation d’un Texte de TextView :

## Défilement Horizontal :

<TextView

android:id="@+id/mywidget"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentRight="true"

android:singleLine="true"

android:ellipsize="marquee"

android:fadingEdge="horizontal"

android:marqueeRepeatLimit="marquee\_forever"

android:scrollHorizontally="true"

android:textColor="#ff4500"

android:text="Simple application that shows how to use marquee, with a long text" />

Le code de Main\_Activity :

tv = (TextView) this.findViewById(R.id.textView);

tv.setSelected(true);

Pour mettre en forme un texe on utilise le code suivant :

tv1.setTypeface(Typeface.SANS\_SERIF);

//only font style

tv1.setTypeface(null,Typeface.BOLD);

//only text style(only bold) tv1.setTypeface(null,Typeface.BOLD\_ITALIC);

//only text style(bold & italic)

tv1.setTypeface(Typeface.SANS\_SERIF,Typeface.BOLD);

//font style & text style(only bold) tv1.setTypeface(Typeface.SANS\_SERIF,Typeface.BOLD\_ITALIC); //font style & text style(bold & italic)

tv1.setTextSize(15);

tv1.setTextSize(1,15);

tv1.setGravity(Gravity.*CENTER\_VERTICAL* | Gravity.*CENTER\_HORIZONTAL*);

tv1.setGravity(Gravity.*CENTER\_VERTICAL*);

tv1.setGravity(Gravity.*CENTER*);

tv1.setTextAlignment(View.*TEXT\_ALIGNMENT\_TEXT\_END*); //aligner à droite.

**Souligner un texte :**

t1=v.findViewById(R.id.*t1*);  
SpannableString content = new SpannableString("حجم الخط");  
content.setSpan(new UnderlineSpan(), 0, content.length(), 0);  
t1.setText(content);

**Justifier un texte de textView**

tv1.setTextSize(1,25);  
if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*) {  
 tv1.setJustificationMode(*JUSTIFICATION\_MODE\_INTER\_WORD*);  
}

ou sous Xml : android:justificationMode="inter\_word"

Changer la couleur :

tv1.setTextColor(Color.*BLUE*);

tv1.setBackgroundColor(Color.*BLUE*);

v.setBackgroundColor(Color.*parseColor*("#E56717"));

Recupérér la couleur d’un Text View

int color = ((ColorDrawable)tv1.getBackground()).getColor();  
int color1 = tv1.getCurrentTextColor();

Tester la couleur d’un Text View :

if(color1==Color.*BLUE*)  
{  
 tv1.setTextColor(Color.*BLACK*);  
}

Tester la taille d’un TextView :

float x=tv1.getTextSize();

Tester le Police d’un TextView :

**Typeface font = Typeface.createFromAsset(getActivity().getAssets(),** "fonts/customfont.ttf");

txt.[**setTypeface**](https://www.codota.com/code/java/methods/android.widget.TextView/setTypeface)(font);

Typeface tf = Typeface.*createFromAsset*(getAssets(),"fonts/mylai.ttf");  
if(tf==Typeface.*SANS\_SERIF*)  
{  
   
}

Tester le Style d’un TextView :

int bold = Typeface.*BOLD*;  
if(bold==Typeface.*BOLD\_ITALIC*)  
{  
   
}

Mise en page :

tv1.setPadding((int) 20, 0, (int) 30, 0);

## Défilement Vertical :