

TD2 – Les interfaces sous Android

Sommaire :

Introduction.....

- 1. Découverte des ressources, layouts et vues.

Réalisation.....

- 2. Tutoriel sur les groupements de vues
 - 2.1. *Mise en page avec des LinearLayout*
 - 2.2 *Mise en page avec un TableLayout*
 - 2.3 *Mise en page avec un RelativeLayout*
 - 2.4. *Mise en page avec un ConstraintLayout*
- 3. Exercices
 - 3.1 *Saisie d'un livre : livre.xml*
 - 3.2 *Saisie de repas : repas.xml*
- 4. Travail à rendre

Conclusion.....

Introduction :

Après avoir réalisé une simple interface avec un message « Hello World » (TD1), on va maintenant réaliser une interface avec des boutons. On va partir d'un projet minimal, type « Empty Activity », nommez-le TD2.

1.Découverte des ressources, layouts et vues

Android contient plusieurs dizaines de composants d'interface :

- des vues : TextView, Button, EditText. . . lire controls.html
- des groupes : LinearLayout, RelativeLayout, TableLayout. . . lire declaring-layout.html.

Cette semaine, on ne s'intéresse qu'à l'apparence, à la mise en page d'interfaces et non pas à leur activité, ce qui sera l'objet du cours et tp de la semaine prochaine, donc vous pouvez ignorer tout ce qui est Responding to Events.

2. Tutoriel sur les groupements de vues

On va s'intéresser aux vues qui permettent d'arranger d'autres vues.

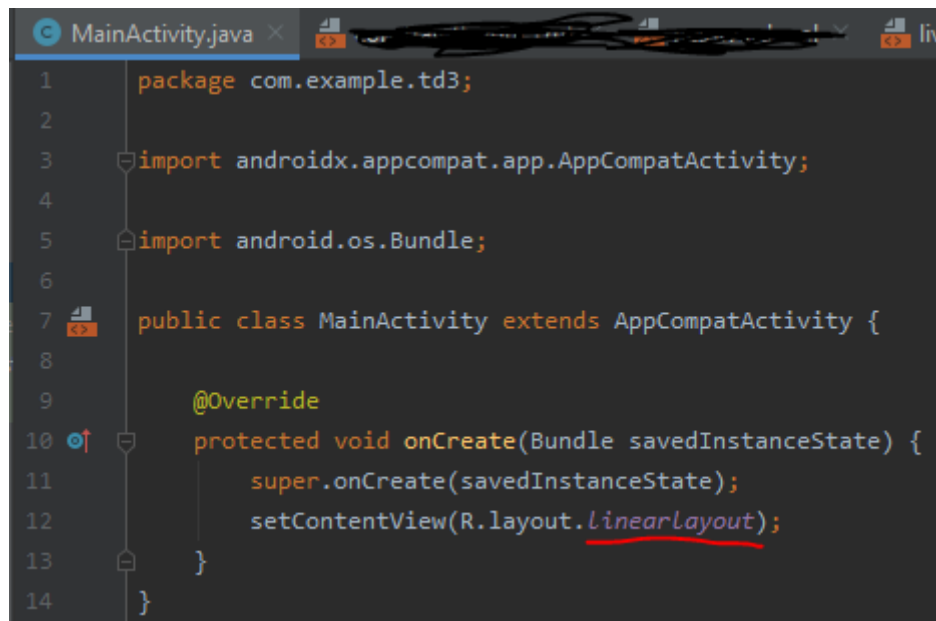
Elles dérivent de la classe ViewGroup.

Pour simplifier, elles ont des vues enfants et décident de leur placement, taille et position, en fonction de propriétés présentes sur les enfants.

Voici les quatre à connaître :

- *LinearLayout pour former des lignes ou des colonnes,*
 - *TableLayout pour disposer en tableau,*
 - *RelativeLayout pour mises en page statiques plus complexes,*
 - *ConstraintLayout pour des dispositions dynamiques complexes*
-

Avant toute chose, il faut avoir créer la classe MainActivity.java (TD1) afin de tester les interfaces que l'on va créer :

A screenshot of the Android Studio IDE showing the MainActivity.java file. The code is as follows:

```
1 package com.example.td3;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.LinearLayout);
13    }
14 }
```

The text "LinearLayout" on line 12 is underlined in red. The file explorer on the right shows a folder named "livre".

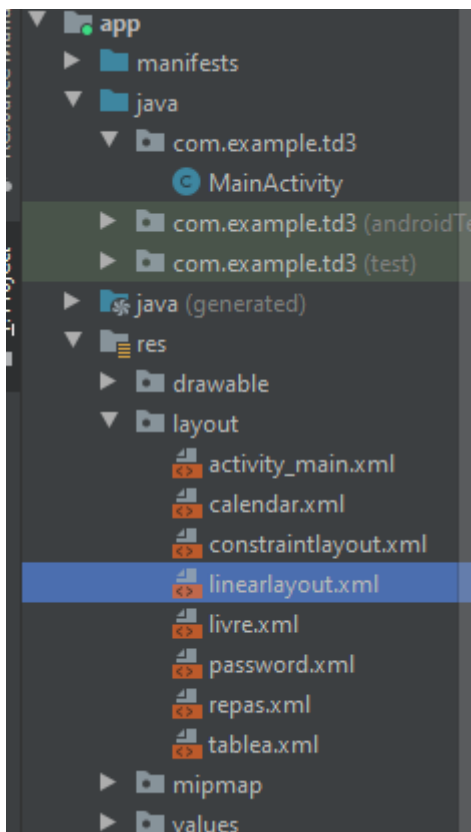
Dans cette classe, on choisit quel fenêtre ([fenetre].xml) ouvrir au lancement en changeant le texte souligné en rouge au dessus
(par exemple si on veut la fenêtre intitulé livre.xml on écrit :

```
setContentView(R.layout.livre);
```

2.1. Mise en page avec des *LinearLayout*

*On crée une nouvelle fenêtre xml appelé *LinearLayout**

Chemin : app/res/layout



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

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

        <Button
            android:text="@string/ok"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <Button
            android:text="@string/annuler"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

        <Button
            android:text="@string/annuler_tout"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>

    </LinearLayout>

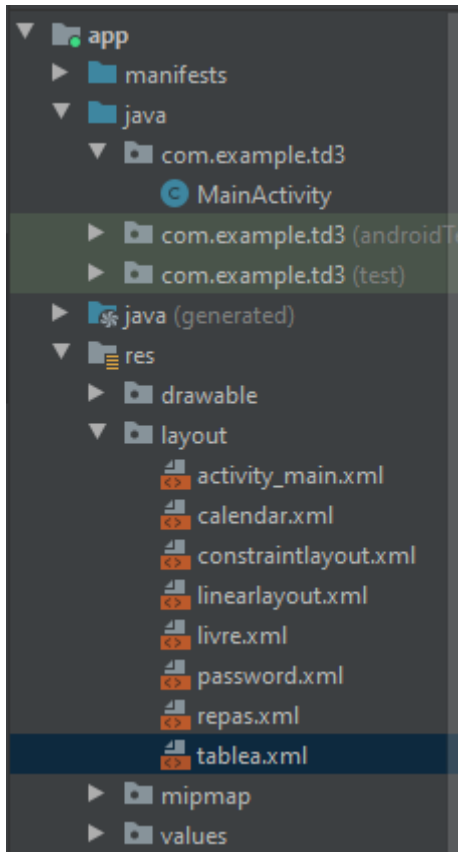
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ce qui donne :



2.2.Mise en page avec un *TableLayout*

On crée une nouvelle fenêtre xml appelé tablea.xml



```
<TableLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:shrinkColumns="0">

    <TableRow
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

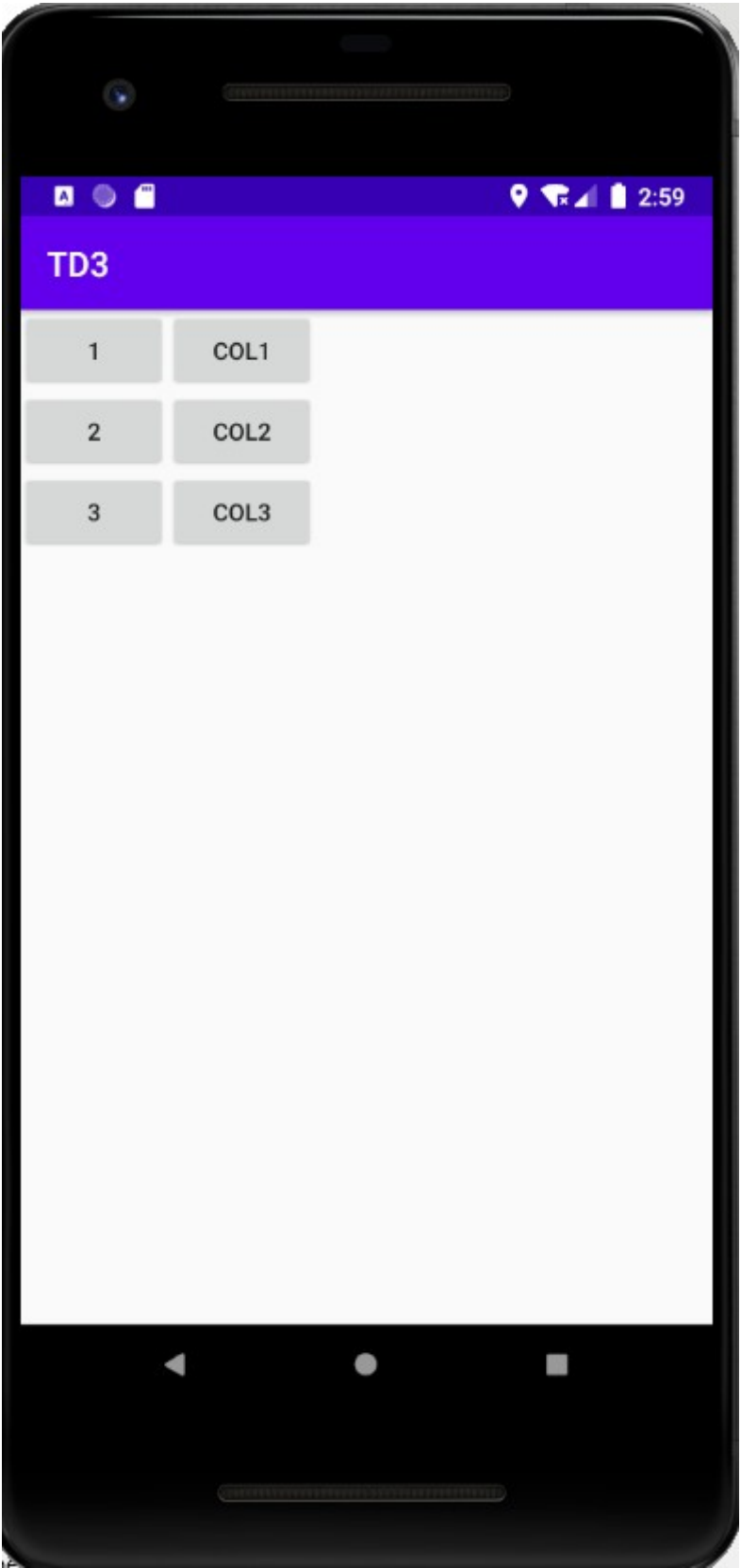
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="1" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="col2" />

    </TableRow>
```

Ici, le `<TableLayout>` , permet d'initialiser une disposition prédéfini sous forme de Layout, le `<TableRow>` permet de créer une ligne de tableau tandis que les `<Button>` sont simplement les composant de cette ligne de tableau avec leurs propriétés (tailles, texte etc).

On repète le code du `<TableRow>` 3 fois de manière à avoir un tableau avec 3 lignes et 2 colonnes, ce qui donne :



2.3.Mise en page avec un RelativeLayout

On crée une nouvelle fenêtre xml appelé relative.xml

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_centerVertical="true"
        android:text="BTN1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/button1"
        android:layout_centerVertical="true"
        android:layout_above="@id/button1"
        android:text="BTN2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:layout_toLeftOf="@id/button1"
        android:text="BTN3" />

    <Button
        android:id="@+id/button4"
        android:layout_width="176dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/button3"
        android:layout_alignRight="@id/button1"
        android:layout_centerVertical="true"
        android:layout_marginTop="0dp"
        android:text="BTN4" />

    <Button
        android:id="@+id/button5"
        android:layout_width="wrap_content"
        android:layout_height="96dp"
        android:layout_below="@id/button2"
        android:layout_centerVertical="true"
        android:layout_toRightOf="@id/button1"
        android:text="BTN5" />

</RelativeLayout>
```

La balise <RelativeLayout> permet de placer des objets tels que des boutons en les mettant à droite ou à gauche d'autres objets, c'est très pratique pour avoir une mise en page propre.

2.4.Mise en page avec un ConstraintLayout

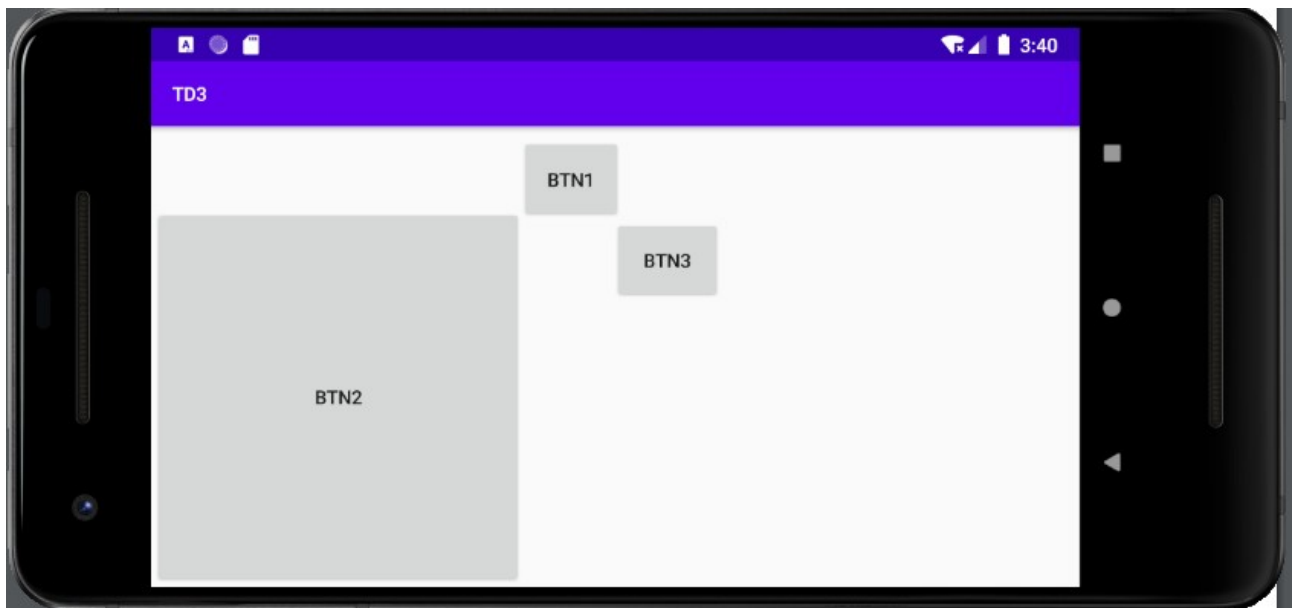
On crée une nouvelle fenêtre xml appelé constraint.xml

```
<Button
    android:id="@+id/button3"
    android:layout_width="80dp"
    android:layout_height="61dp"
    android:layout_marginStart="340dp"
    android:layout_marginTop="68dp"
    android:text="BTN3"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/button1"
    android:layout_width="75dp"
    android:layout_height="62dp"
    android:layout_marginStart="272dp"
    android:layout_marginTop="8dp"
    android:text="BTN1"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/button2"
    android:layout_width="271dp"
    android:layout_height="278dp"
    android:layout_marginStart="3dp"
    android:layout_marginTop="60dp"
    android:text="BTN2"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

La disposition ConstraintLayout permet de placer des objets pixels par pixels, pour avoir un positionnement parfait :



3. Exercices

3.1 Saisie d'un livre : livre.xml

On crée une nouvelle fenêtre xml appelé livre.xml

Créer un écran de saisie pour un livre : titre, auteur, année, isbn et notation.

La notation sera réalisée par un RatingBar. Cette vue est faite pour attribuer une note sous forme d'étoiles, mais elle est un peu déroutante à utiliser :

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

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

    <EditText
        android:layout_width="298dp"
        android:layout_height="wrap_content"
        android:layout_weight="0" />

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

    <EditText
        android:layout_width="298dp"
        android:layout_height="wrap_content"
        android:layout_weight="0" />

    <RatingBar
        android:id="@+id/ratingBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:stepSize="0.5" />

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:text="Valider" />
```

*On crée un texte avec
<TextView>*

*On crée une zone de texte
avec <EditText>*

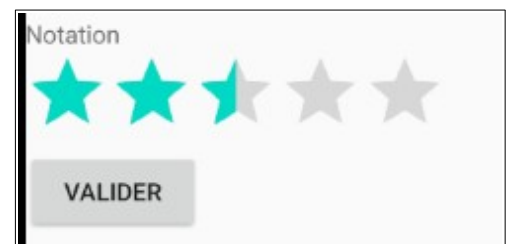
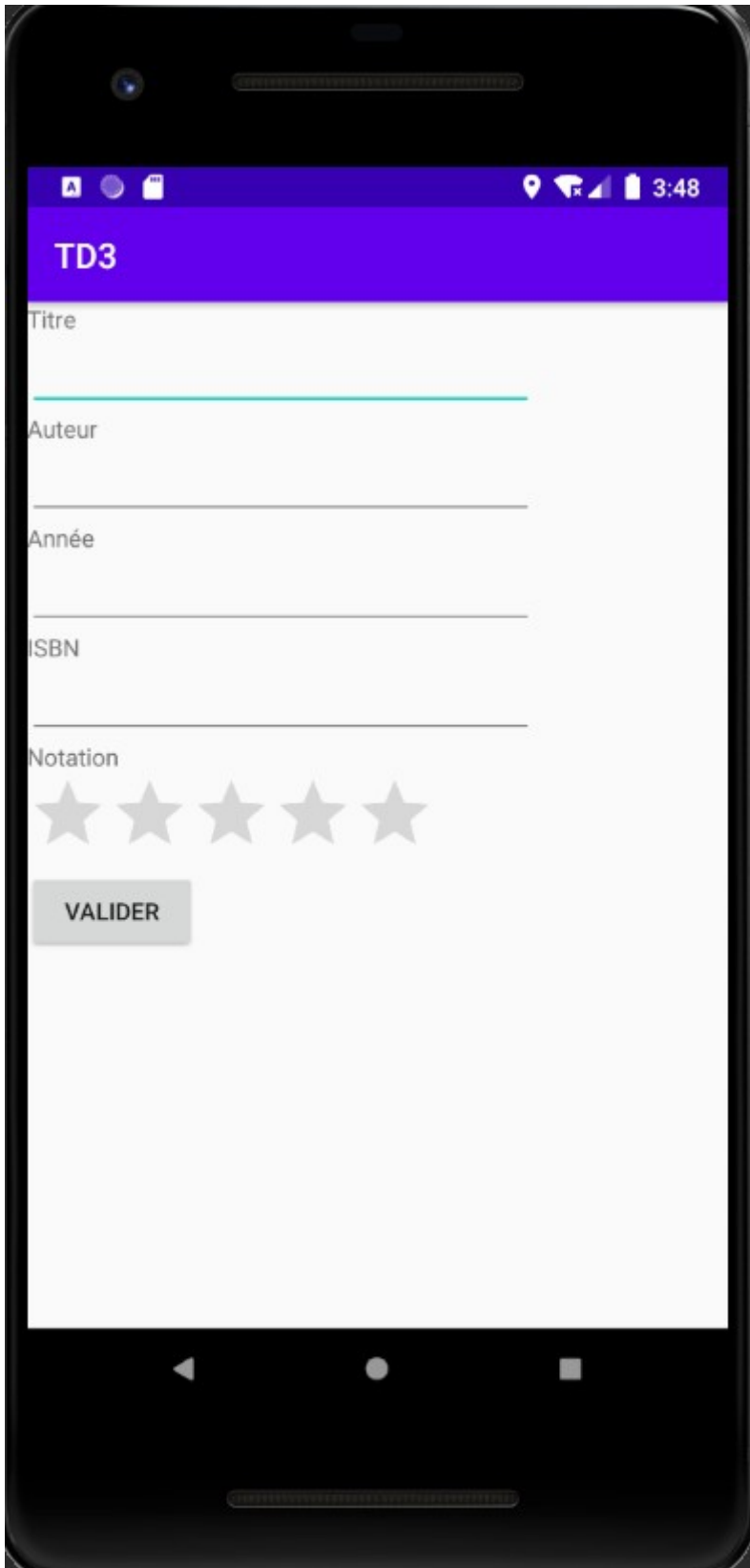
*On crée un texte avec
<TextView>*

*On crée une zone de texte
avec <EditText>*

*On crée une barre de notation
avec <RatingBar> et ses
propriétés*

*Et enfin un Bouton avec
<Button> avec comme texte
« Valider »*

Résultat de l'interface :



Voici des étoiles de notation comme demandé avec comme step 0.5 (c'est le pas de notation)

Conclusion :

Ce TD bien que un peu compliqué avec les soucis d'android studio à son installation est très intéressant, cela m'a permis de découvrir les différentes disposition (Linear/Relative/Constraint et Table) afin de réaliser une interface sous Android Studio.