**Khmeissi Imene**

**3ème année DSI TI Promo 8 EAD**

### 

### ****TP Android n°3 : Construction de l'interface****

**Prérequis :** Aucun (mais pour la suite, il faut avoir des connaissances en Java et programmation objet).

**But :** Concevoir l'interface graphique d'une application Android en utilisant des **layouts**, des **widgets**, et des notions de **positionnement**.

#### 1. **Création du fichier de ressources Layout**

Nous avons créé un fichier de ressources XML nommé activity\_main.xml qui va servir à définir l'interface de l'application. Ce fichier contient la disposition des éléments graphiques que l'utilisateur interagira avec sur l'écran, tels que les boutons, les champs de texte et les images. Ce fichier est ensuite lié à l'activité principale de l'application (MainActivity.java) pour afficher l'interface à l'utilisateur.

#### 2. **Utilisation des layouts**

L'interface est structurée dans un **RelativeLayout** qui contient plusieurs **LinearLayout**. Ces layouts servent à organiser les éléments de manière verticale et horizontale.

* Exemple de RelativeLayout comme conteneur principal :

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

* Exemple d’un **LinearLayout** vertical pour les champs de saisie (poids, taille, âge) :

<LinearLayout

android:layout\_width="match\_parent"

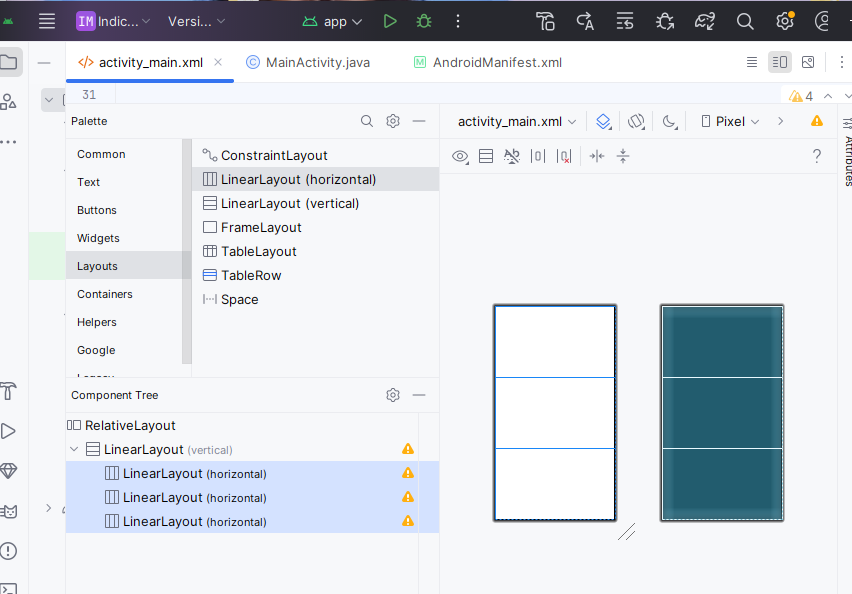
android:layout\_height="wrap\_content"

android:orientation="horizontal">

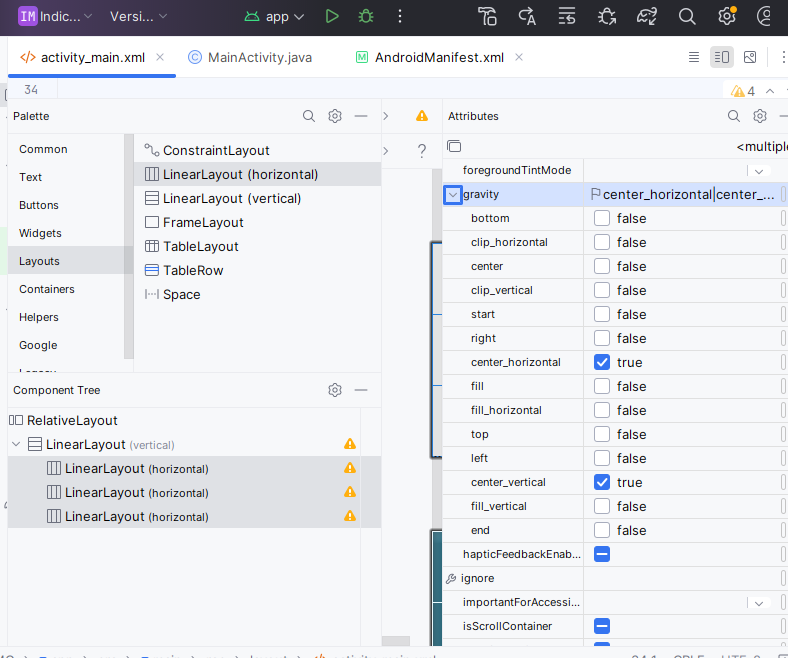
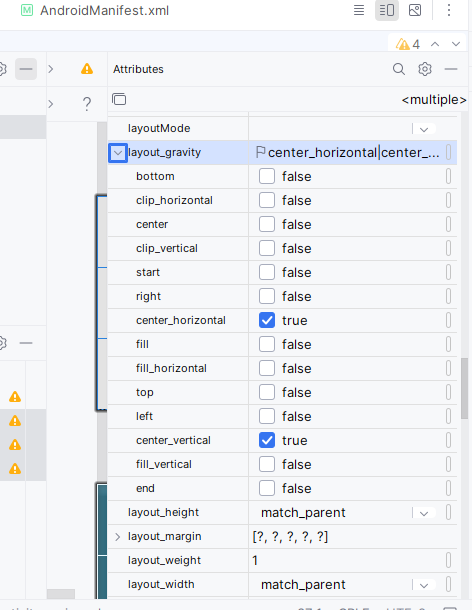
<TextView android:text="Poids" />

<EditText android:inputType="number" />

</LinearLayout>

****

* On clique sur « Distribute weights evenly » 🡺 Permet une distribution de taille égale des 3 layouts.

* On modifie au niveau des attributs Gravity et Layout\_gravity les propriétés center-horizontal et center\_vertical en True 🡺 cela permet de centrer les objets verticalement et horizontalement à l’intérieur de layout

3. **Widgets**

Les widgets incluent des éléments interactifs et d’affichage comme les **RadioButton**, **EditText**, **Button**, et **ImageView**.

* Exemple d'un **RadioGroup** avec deux **RadioButton** pour choisir Homme ou Femme :

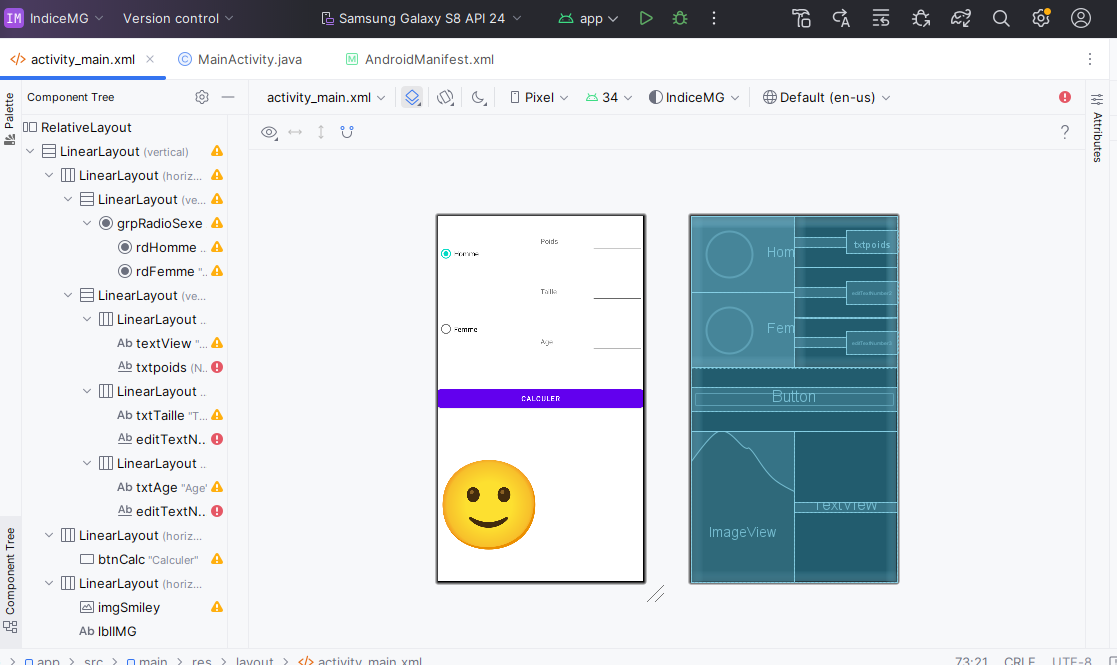
<RadioGroup android:id="@+id/grpRadioSexe">

<RadioButton android:id="@+id/rdHomme" android:text="Homme" />

<RadioButton android:id="@+id/rdFemme" android:text="Femme" />

</RadioGroup>

* On a modifie les textView par Poids, Taille et Age, le bouton central par Calculer et les boutons Radio par Homme « checked par défaut » et Femme.
* On insère une photo « Smiley » via la viewImage.
* Et on termine par donner des identiiants aux différents objets graphiques qu’on va yutiliser dans l’application



#### 4. **Positionnement et organisation des éléments**

L'utilisation de **android:layout\_weight** permet de gérer l'espace attribué aux éléments dans un LinearLayout.

* Exemple de **TextView** et **EditText** pour saisir le poids, avec gestion de la largeur en fonction du poids :

<EditText

android:id="@+id/txtPoids"

android:layout\_width="0dp"

android:layout\_weight="1"

android:inputType="number" />

#### 5. **Conclusion**

Dans cet atelier, nous avons appris à construire une interface en utilisant les layouts et widgets d'Android. Nous avons créé un fichier de ressources activity\_main.xml qui définit l'interface graphique.

La disposition des éléments se fait de manière flexible grâce à **LinearLayout** et **RelativeLayout**, permettant ainsi de positionner les widgets de manière fluide dans l'interface graphique.

Cela sert de base pour la création d'une interface simple où l'utilisateur peut interagir avec les champs de saisie, les boutons et visualiser les résultats. Ce fichier activity\_main.xml est essentiel pour configurer l'apparence de l'interface de l'application.

