Réaliser une Interface Utilisateur Native

Temps : **2h**

Dificulté : **

Le but de ce TD de se familiariser avec les éléments graphiques fournis par l'API Android. La bibliothèque d'éléments graphiques Android se compose de **Layouts** (conteneur d'éléments), de **Widgets** (éléments graphiques ou interactifs) et de **Menus** pour structurer une application (Action Bar, Navigation Drawer, etc.). Les exercices suivants proposent de mettre en pratique les éléments d'interface graphique suivant :

le **ConstraintLayout**

un conteneur d'élément, tel que le défilement

la gestion du mode portrait ou paysage

un menu Android

Note : Chaque élément graphique sera mis en œuvre dans une nouvelle *Activity* qui devra être déclarée dans le fichier AndroidManifest.xml.

A. Agencement de l'Interface Utilisateur

Le *Layout* [16], soit agencement en français, est un conteneur d'éléments graphiques. Cela permet d'avoir des modèles, *templates*, pour placer les éléments graphiques dans l'interface utilisateur.

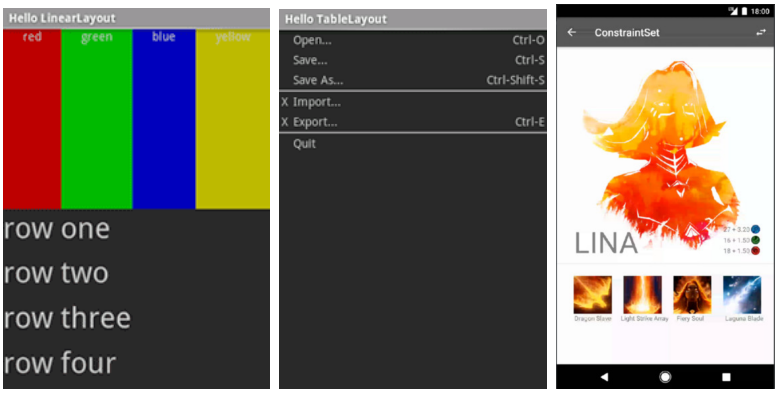

Figure 1 – Principaux types d'agencement Android

1] Principaux agencements

Les *layouts*, agencement, Android sont utilisés pour agencer des éléments graphiques dans un écran, les principaux sont : **LinearLayout** [17], **TableLayout** [41], **GridLayout**, **RelativeLayout** [34], et **ConstraintLayout** [7](cf. Figure 1).

Le **ConstraintLayout** est l'évolution du **RelativeLayout**[48], c'est un **ViewGroup** qui affiche des éléments graphiques à des positions relatives. D'une part, il permet d'afficher un élément par rapport à un autre élément (à gauche ou en dessous d'un élément). D'autre part, il permet d'afficher un élément par rapport à la zone d'affichage (aligné à gauche, en bas , au centre, etc.). Ce type d'agencement est intéressant car il permet d'utiliser un seul **ViewGroup** là où il y en aurait besoin de plusieurs.
Concernant le **ConstraintLayout**, il y a deux types d'attributs. D'une part, il y a l'attribut du genre app :layout_constraintTop_toTopOf prenant pour valeur *parent* :

```
app:layout_constraintTop_toTopOf="parent"
```

Il place notre élément dans son parent. Dans la même veine, il y a :

- app :layout_constraintStart_toStartOf
- app :layout_constraintEnd_toEndOf.

D'autre part, il y a l'attribut du genre app :layout_constraintTop_toBottomOf prenant pour valeur l'identifiant d'un autre élément :

```
app:layout_constraintTop_toBottomOf="@id/itemText"
```

Il place notre élément en dessous de l'autre élément. Dans la même veine, il y a :

- app :layout_constraintBottom_toTopOf
- app :layout_constraintStart_toEndOf
- app :layout_constraintEnd_toStartOf
- app :layout_constraintLeft_toRightOf
- app :layout_constraintRight_toLeftOf

De manière générale, avec cet agencement, chaque élément a besoin d'un attribut de placement vertical et un attribut de placement horizontal.

1. Ouvrez un nouveau projet.

2. Dans le fichier XML, res/layout/activity_main, vérifiez l'élément racine, *Root element*, il devrait être un **ConstraintLayout** lors de la création.
3. Ajoutez-y, une image et deux textes.
5. Vérifiez dans votre Activity principale, MainAtivity.java, le fichier XML de défaut activity_main.xml.
6. Modifiez le fichier de vue afin d'obtenir le résultat de la figure 2 (une image avec à côté deux textes, cela dans un **ConstraintLayout**, ce dernier est placé dans une *CardView*).

Note : Dans le contexte d'application pour smartphone, il est préconisé d'utiliser des images au format *.png* (format reconnu pour sa légèreté).

Exercice : Définissez une taille générique pour la taille de l'image via le fichier dimens.xml.

```
<dimen name="size_120">120dp</dimen>
```

Mise en pratique de la Grille

L'exercice proposé consiste à mettre en pratique la grille soit le composant **GridLayout** à ne pas confondre avec la **GridView** [14].

1. Créez un fichier dans res/layout/, nommez-le activity_grid.xml et insérez ce qui suit :

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2"
    android:rowCount="1"
    tools:context=".MainActivity">

    <View
        android:layout_width="match_parent"
        android:layout_height="@dimen/common_width"
        android:background="@color/colorAccent"
        android:layout_column="0"
        android:layout_row="0"
        android:layout_columnSpan="2"
        />
</GridLayout>
```

2. Créez une nouvelle classe GridActivity.java qui hérite de **android.app.AppCompatActivity**
3. Liez la vue à l'activité dans la méthode onCreate() :

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_grid)
    }
```

4. Ajoutez l'activité ainsi créée au fichier manifest afin qu'elle soit le point d'entré de votre application (à la place de MainActivity).

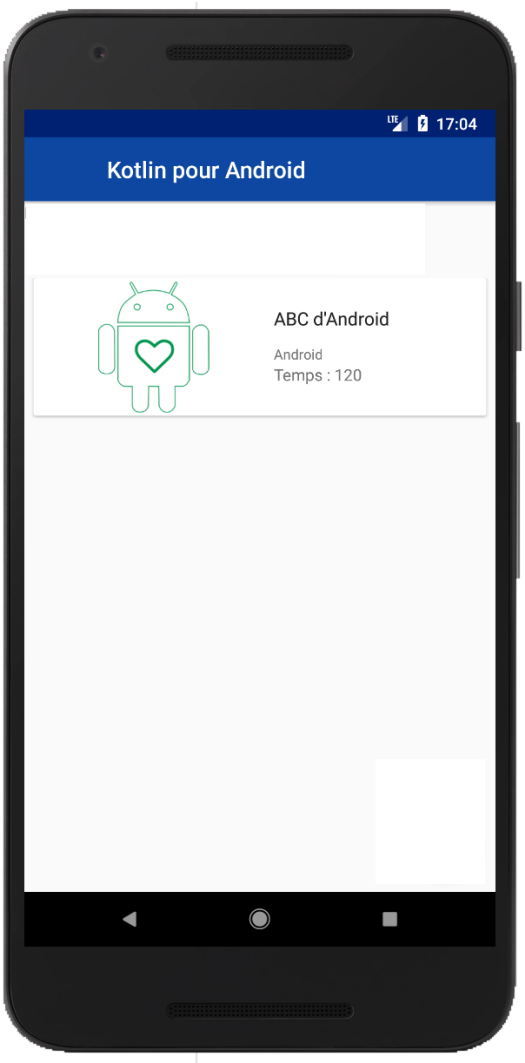

FIGURE 2 – Exemple d'agencement (ConstraintLayout dans un Card)

Exercice : Modifiez afin d'afficher une grille à deux colonnes et quatre lignes (cf. Figure 3).

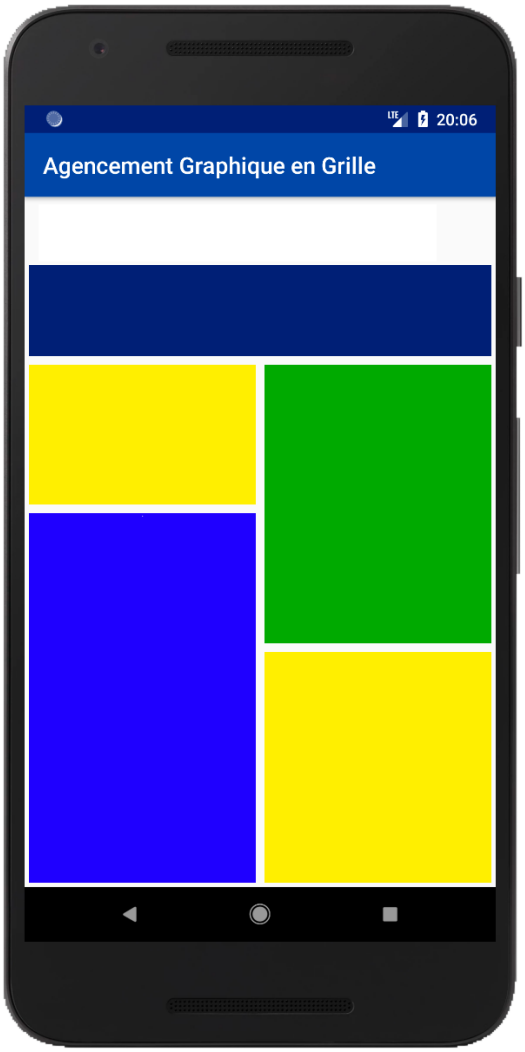

FIGURE 3 – Exemple d'agencement (GridLayout avec 2 colonnes et 4 lignes)

2] Agencement de type conteneur

Certains **Layouts** sont utilisées pour agencer une liste d'éléments, images ou textes, plus précisément ils sont utilisés pour présenter une liste de données (objet ou image). Citons-en quelques uns : **GridView** [14], **ListView** [18], **Gallery** [12] (cf. Figure 4), **MapView** [19]. Cette dernière affiche une carte du monde interactive sur laquelle ils est possible de placer des images à des coordonnées géographiques.
Par ailleurs, la **WebView** [47] permet d'afficher une page web (cf. Figure 4).

Mise en pratique du défilement

Exercice : Dans une vue XML, placez un texte très long et utilisez le composant graphique *ScrollView* afin de l'afficher à l'écran.

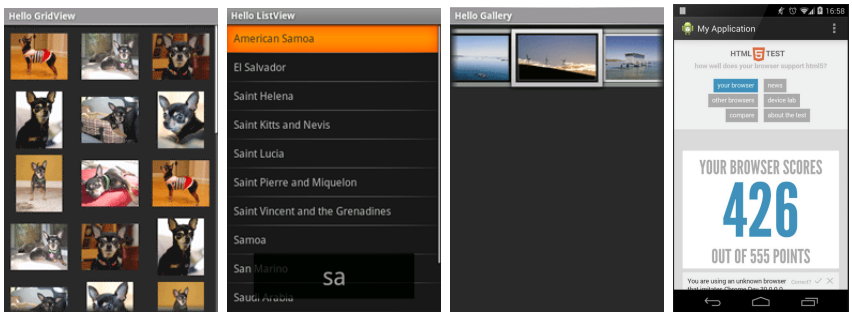

FIGURE 4 – Principales vues ou views

B. Utilisation des Ressources à bon escient

Mise en pratique du mode paysage

Exercice : Créez une vue spécifique pour le mode paysage (cf. Figure 5). En effet, on souhaite, en mode portrait, afficher une image suivi d'un texte (soit un agencement linéaire vertical), tandis qu'en mode paysage on souhaite afficher une image à côté d'un texte (soit un agencement linéaire horizontal). Il s'agit d'ajouter un dossier de ressource spécifique au mode paysage.

Mise en pratique de l'internationalisation

Exercice : Ajoutez le support de la langue française dans votre projet.

Mise en pratique d'un style

Exercice : Ajoutez un style dans le fichier *styles.xml* et utilisez le sur des éléments d'une vue.

Mise en pratique des tableaux de chaîne de caractères

Exercice : Ajoutez un tableau de chaîne de caractères (*string-array*) et accédez aux valeurs depuis le code.

Mise en pratique de l'interpolation de chaîne de caractères

Exercice : Modifiez un texte depuis le code en mettant en pratique l'interpolation de chaîne de caractères *Kotlin*.

FIGURE 5 – Exemple de gestion du mode Portrait/Paysage

C. Éventail d'éléments graphiques

Les composants graphiques les plus courants sont le **DatePicker**, le **TimePicker** [26], le **Spinner** [38], l'**AutoComplete** [2], la **SeekBar** [37], le **Progress** [28], les **Dialogs** [9]... Il y a aussi les éléments de formulaires (cf. Figure 6) : **Button**, **RadioButton** [33], **CheckBox**, **ToggleButton** [40], **EditText** [10], **RatingBar**.
Ceci dit, il existe une multitude de composants graphiques Android consultables à partir du site Material Design de Google.

Mise en pratique des boutons

Exercice : Créez une classe qui étend **AppCompatActivity** et implémente la méthode *onCreate(..)*.
Initialisez la vue xml avec des éléments **Button**(déclaration du **Button** dans un fichier XML).
Modifiez le style afin qu'il présente différentes ergonomies (cf. Figure 7).

Mise en pratique de boîte de dialogue

Exercice : Ouvrez une boîte de dialogue lorsque l'utilisateur clique sur un bouton [49].

Mise en pratique du navigateur web

Exercice : Ouvrez le navigateur web lorsque l'utilisateur clique sur un bouton [49].

Mise en pratique du champs d'édition

Exercice : Utilisez l'interface **TextWatcher** avec l'élément graphique **EditText** afin d'afficher un **Toast** lorsque l'utilisateur entre du texte.

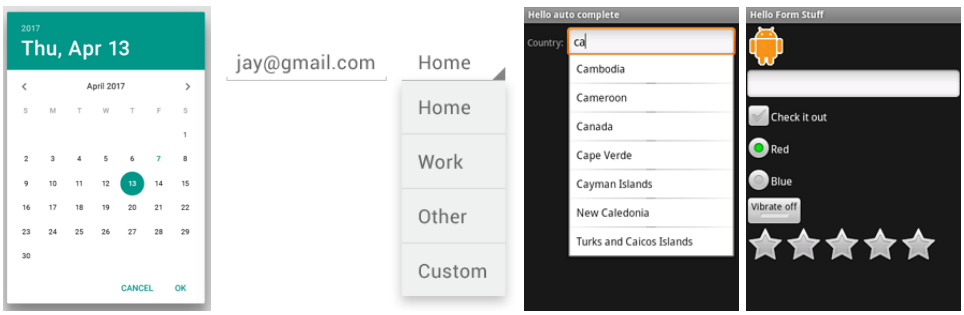

FIGURE 6 – Éléments de formulaire

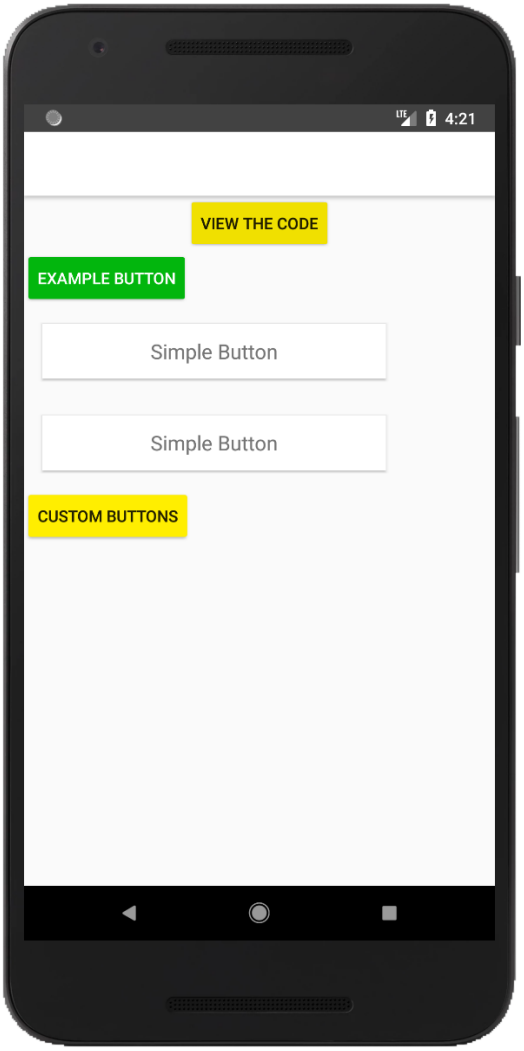

FIGURE 7 – Exemple de différents styles graphiques sur des boutons

D. Élément de Navigation

Il existe différents types de menu Android, certains sont apparus au cours des évolutions des divers versions d'Android [22].
Le **OptionsMenu** est un menu dédié à un écran, il apparaît lorsque la touche menu du téléphone est pressée. Ce menu doit être différent pour chaque écran d'une application. Toute fois, cette association avec la *touche menu* n'est utilisée que dans la version 2.3, il a été placé dans l'**ActionBar** [3].
L'**ActionBar** est arrivée avec la version 3.0, HoneyComb, (cf. Figure **??** [1]), elle contient l'**OptionsMenu**. Cela dit, elle a été remplacé par la **Toolbar** [45] avec la version 5, Marshmallow.
Le **NavigationDrawer** est, de manière générale, un menu commun à l'ensemble des écrans de l'application (cf. Figure **??** [23]).
Le **ContextMenu** est un menu relatif à un élément d'un écran, il peut apparaître lorsque l'utilisateur laisse son doigt appuyé longtemps sur cet élément (cf. Figure **??**).
La structure de ces menus a la même hiérarchie qu'une View (structure d'arbre). Pour les créer il suffit de réécrire la méthode onCreateOptionsMenu() ou bien onCreateContextMenu() d'une Activity. Ensuite, il est possible de déclarer les éléments du menu dans un fichier XML ou bien avec du code java. Nous opterons pour la première méthode, la moins fastidieuse.

Mise en pratique du Menu option pour ToolBar

Exercice :

1. Reprenez la classe MainActivity.
2. Créer un dossier menu dans le dossier res/
3. Créer un fichier main.xml dans le dossier menu :

```
<menu 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"
    tools:context="com.example.heartrate.MainActivity" >
    <item
        android:id="@+id/action_settings"
        android:icon="@drawable/ic_settings"
        android:title="@string/action_settings"
        app:showAsAction="ifRoom"/>
</menu>
```

4. Créez les ressources utilisées dans strings.xml et dans drawable, vérifiez le contexte (le nom du package).

6. Ajoutez dans la classe MainActivity, l'initialisation du **OptionsMenu** avec le fichier main.xml :

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.main, menu);
  restoreActionBar();
  ActionBar actionBar = getSupportActionBar();
  actionBar.setDisplayShowTitleEnabled(true);
  actionBar.setTitle(mTitle);
  return super.onCreateOptionsMenu(menu);
}
```

7. Ajoutez les méthodes d'interaction sur le menu :

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // Handle presses on the action bar items
  switch (item.getItemId()) {
    case R.id.action_settings:
            //faites un toast !
      return true;
  default:
    return super.onOptionsItemSelected(item);
  }
}
```

Mise en pratique des Menus

Exercice : Créez une activité dont le modèle est *NavigationDrawer*. Personnalisez les menus dans l'objectif de construire une application mettant en valeur ce que vous apprenez dans ce module d'enseignement.

Références du cours Interface Utilisateur Native

A. Vue

1] View & Activity, Fragment

2] Configuration

3] Agencement

— Éditer une vue avec Android Studio [15]
— Déclarer des agencements [16]
— Agencement linéaire [17]
— Agencement relatif [34]
— Agencement en fenêtre [11]
— Agencement en tableau [41]
— Agencement en grille [13]

B. Ressource

1] Res

2] Drawable

3] Values

Utiliser les ressources pour rendre une application accessible [32]

C. Élément graphique natif

1] Éléments indispensables

— Zone d'affichage d'un texte [42]
— Zone d'édition d'un texte [10]
— Bouton [4]
— Case à cocher [6]
— Image

2] Élément de formulaire

- Dérouleur [38]
- Auto-complétion [2]
- Bouton radio [33]
- Bouton à bascule [44],[40]
- Curseur [37]
- Processeur d'étape[39]
- Sélecteur d'heure[27]
- Sélecteur de date[26]

3] Conteneur

- Gallerie [12]
- Carte [5]
- Liste [18]
- Grille [14]
- Défilant [36]
- Vue Web [47]
- Carte géographique [19]

4] Message utilisateur

- Notification [25]
- Fenêtre de dialogue [8] [9] [43]
- Barre d'avancement [29] [31] [30] [28]

D. Élément de navigation

- Barre d'application [1] [45]
- Menu d'action [22]
- Menu Hamburger [23][24]
- Onglet [46]

E. Material Design

- Concept [20]
- Ressources d'icônes Google [21]
- interface responsive [35]

Webographie

[1] Developer Android. App bar. https://developer.android.com/training/appbar/index.html.

[2] Developer Android. Auto-complete. https://developer.android.com/reference/android/widget/AutoCompleteTextView.html.

[3] Developer Android. Backward compatibility. http://developer.android.com/design/patterns/compatibility.html.

[4] Developer Android. Button. https://developer.android.com/guide/topics/ui/controls/button.html.

[5] Developer Android. Cards. https://developer.android.com/training/material/lists-cards.html.

[6] Developer Android. Checkbox. https://developer.android.com/guide/topics/ui/controls/checkbox.html.

[7] Developer Android. Constraintlayout. https://developer.android.com/training/constraint-layout/.

[8] Developer Android. Dialog. http://developer.android.com/design/building-blocks/dialogs.html.

[9] Developer Android. Dialogs. https://developer.android.com/guide/topics/ui/dialogs.html.

[10] Developer Android. Edittext. https://developer.android.com/guide/topics/ui/controls/text.html.

[11] Developer Android. Framelayout. https://developer.android.com/reference/android/widget/FrameLayout.html.

[12] Developer Android. Gallery. https://developer.android.com/reference/android/widget/Gallery.html.

[13] Developer Android. Gridlayout. android-developers.blogspot.fr/2011/11/new-layout-widgets-space-and-gridlayout.html.

[14] Developer Android. Gridview. https://developer.android.com/guide/topics/ui/layout/gridview.html.

[15] Developer Android. Layout editor. https://developer.android.com/studio/write/layout-editor.html.

[16] Developer Android. Layouts. http://developer.android.com/guide/topics/ui/declaring-layout.html.

[17] Developer Android. Linearlayout. https://developer.android.com/guide/topics/ui/layout/linear.html.

[18] Developer Android. Listview. https://developer.android.com/guide/topics/ui/layout/listview.html.

[19] Developer Android. Map. https://developer.android.com/training/maps/index.html.

[20] Developer Android. Material design. https://developer.android.com/design/index.html.

[21] Developer Android. Material icons. https://design.google.com/icons/index.html.

[22] Developer Android. Menus. http://developer.android.com/guide/topics/ui/menus.html.

[23] Developer Android. Navigation drawer. https://developer.android.com/design/patterns/navigation-drawer.html.

[24] Developer Android. Navigationdrawer. https://developer.android.com/training/implementing-navigation/nav-drawer.html.

[25] Developer Android. Notifications. https://developer.android.com/guide/topics/ui/notifiers/notifications.html.

[26] Developer Android. Picker. https://developer.android.com/guide/topics/ui/controls/pickers.html.

[27] Developer Android. Pickers. http://developer.android.com/design/building-blocks/pickers.html.

[28] Developer Android. Progress and activity. http://developer.android.com/design/building-blocks/progress.html.

[29] Developer Android. Progress in notifications. https://developer.android.com/training/notify-user/display-progress.html.

[30] Developer Android. Progressbar. https://developer.android.com/reference/android/widget/ProgressBar.html.

[31] Developer Android. Progressdialog. https://developer.android.com/reference/android/app/ProgressDialog.html.

[32] Developer Android. Providing resources. https://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources.

[33] Developer Android. Radiobutton. https://developer.android.com/guide/topics/ui/controls/radiobutton.html.

[34] Developer Android. Relativelayout. https://developer.android.com/guide/topics/ui/layout/relative.html.

[35] Developer Android. Responsive ui. https://material.google.com/layout/responsive-ui.html.

[36] Developer Android. Scrollview. https://developer.android.com/reference/android/widget/ScrollView.html.

[37] Developer Android. Seek bars and slider. http://developer.android.com/design/building-blocks/seek-bars.html.

[38] Developer Android. Spinner. https://developer.android.com/guide/topics/ui/controls/spinner.html.

[39] Developer Android. Stepper. https://material.google.com/components/steppers.htmll.

[40] Developer Android. Switches. http://developer.android.com/design/building-blocks/switches.html.

[41] Developer Android. Tablelayout. https://developer.android.com/guide/topics/ui/layout/grid.html.

[42] Developer Android. Textview. https://developer.android.com/reference/android/widget/TextView.html.

[43] Developer Android. Toast. https://developer.android.com/guide/topics/ui/notifiers/toasts.html.

[44] Developer Android. Togglebutton. https://developer.android.com/guide/topics/ui/controls/togglebutton.html.

[45] Developer Android. Toolbar. https://developer.android.com/training/appbar/setting-up.html.

[46] Developer Android. Viewpager. https://developer.android.com/training/implementing-navigation/lateral.html.

[47] Developer Android. Webview. https://developer.android.com/guide/webapps/webview.html.

[48] Huyen Dao. Cool constraintlayout. https://speakerdeck.com/queencodemonkey/devfest-dc-2017-cool-constraintlayout.

[49] Macha. Utiliser la bibliothèque anko. https://www.chillcoding.com/blog/2017/10/09/utiliser-anko-kotlin-android/.