



Interfaces Graphiques

sous android



I. Introduction

Les interfaces graphiques prennent une place de plus en plus importante dans le choix des applications par les utilisateurs, les interfaces d'applications Android sont organisées en vues et gabarits, avec néanmoins quelques spécificités. Une interface n'est pas une image statique mais un ensemble de composants graphiques, qui peuvent être des boutons, du texte, mais aussi des groupements d'autres composants graphiques, pour lesquels nous pouvons définir des attributs communs (taille, couleur, positionnement, etc.). [1]

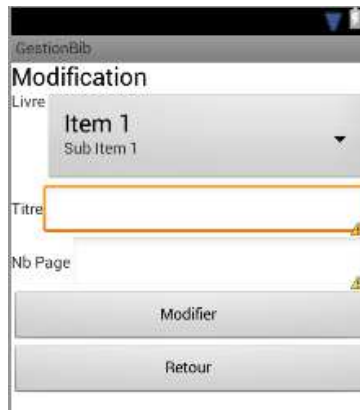


Figure 1 : Exemple d'Interface

L'interface peut se faire de deux façons :

- Par description de l'interface dans des fichiers XML
- Par programme

Les fichiers XML qui décrivent une interface sont placés dans le répertoire **res/layout**. Ils sont référencés par **R.layout.nom_du_fichierXML**.

Les activités peuvent utiliser la méthode **setContentView(R.layout.nom_du_fichierXML)** pour mettre en place l'interface décrite par un tel fichier. [2]

II. Structure d'un projet android

La structure d'un projet android contient :

- AndroidManifest.xml : dans lequel se fait la déclaration des permissions, des activités qui composent l'application et de l'activité de démarrage.
- Un dossier res qui contient les différentes ressources de l'application :
 - o deux dossiers drawable et mipmap pour les images,
 - o un dossier layout pour les interfaces des activités,
 - o un dossier menu pour les menus,
 - o un dossier values qui contient strings.xml pour les constantes de type texte, styles.xml pour les style et dims.xml pour les dimensions.
- Un dossier java qui contient les package des classes java de l'application.

III. La classe Activity

Dans l'univers Android, les activités (Activity en anglais) font partie des objets les plus utilisés. Chaque écran que voit et manipule l'utilisateur est, en effet, implémenté par une classe qui hérite de la classe Activity. [6]

À quelques exceptions près (principalement les services), une application comporte au minimum une classe héritant de Activity et peut, bien sûr, en comporter plusieurs. Une et une seule activité est lancée au démarrage d'une application. Chaque activité, pour être lancée, doit impérativement être déclarée dans le fichier Manifest.xml, dans une balise <activity>, balise enfant de la balise <application>. [6]

En fonction de l'état de l'activité, le développeur a la possibilité de gérer celle-ci à l'aide des méthodes héritées de la classe Activity. [6]

Parmi ces méthodes, on trouve:

- onCreate(): lorsque l'activité est créée.
- onStart(): lorsque l'activité est démarrée (visible par l'utilisateur).
- onResume(): lorsque l'activité redémarre après une pause.
- onPause(): lorsque l'activité est en pause (non visible par l'utilisateur).
- onStop(): lorsque l'activité s'arrête.
- onDestroy(): lorsque l'activité est détruite.

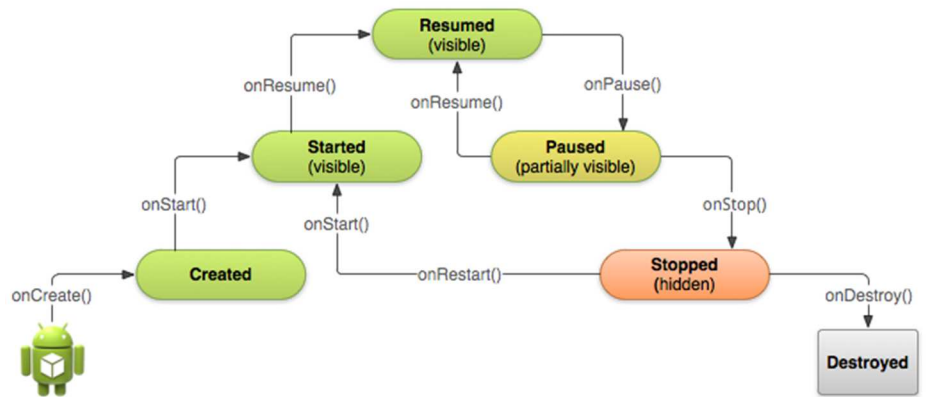


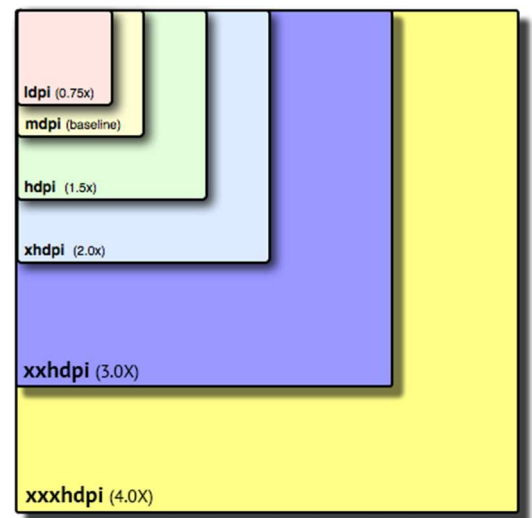
Figure 2 : cycle de vie d'une activité [2]

IV. Propriétés et classes de base des interfaces

IV.1. Les unités

Lorsque l'on indique des unités dans les fichiers XML elles peuvent l'être de plusieurs façons :

- en pixels (faire suivre la valeur de **px**)
- en millimètres (faire suivre la valeur de **mm**)
- en pouces (faire suivre la valeur de **in**) avec $1\text{px} \approx 2,54\text{ cm}$
- en points = $1/72$ pouce (faire suivre la valeur de **pt**)
- en pixels à densité indépendante valent 1 pixel pour un écran de 160 dpi (dots per inch) (faire suivre la valeur de **dp**)
- relativement à la taille de la fonte de caractères (faire suivre la valeur de **sp**)



IV.2. Les couleurs

Elles sont définies par un code hexadécimal indiquant la transparence et les composantes (RVB) sous la forme "#AARRVBBB" où : AA est la transparence (FF opaque, 00 totalement transparent). Si le code de couleur ne comporte que 3 composantes AA est considéré comme FF c'est-à-dire que #44FA8B est équivalent à #FF44FA8B. RR VV et BB sont, respectivement, les composantes rouge, verte et bleue

IV.3. La classe View

Les classes View et ViewGroup ne sont pas utilisées directement mais constituent les classes mères des autres.

Elles sont décrites ici pour éviter de répéter leurs propriétés dans les autres classes d'interface.

View est la classe dont héritent toutes les classes utilisées pour réaliser des interfaces. Ses propriétés et ses méthodes se retrouvent donc dans tous les éléments d'interface.

IV.4. La classe ViewGroup

un type de View contenant d'autres View, gérées par un même gestionnaire de mise en page : positionnement des éléments les uns par rapport aux autres (grille, liste verticale, etc.). [8]

IV.5. Hiérarchie des sous classes de View

La figure suivante présente la hiérarchie des sous classes de View.

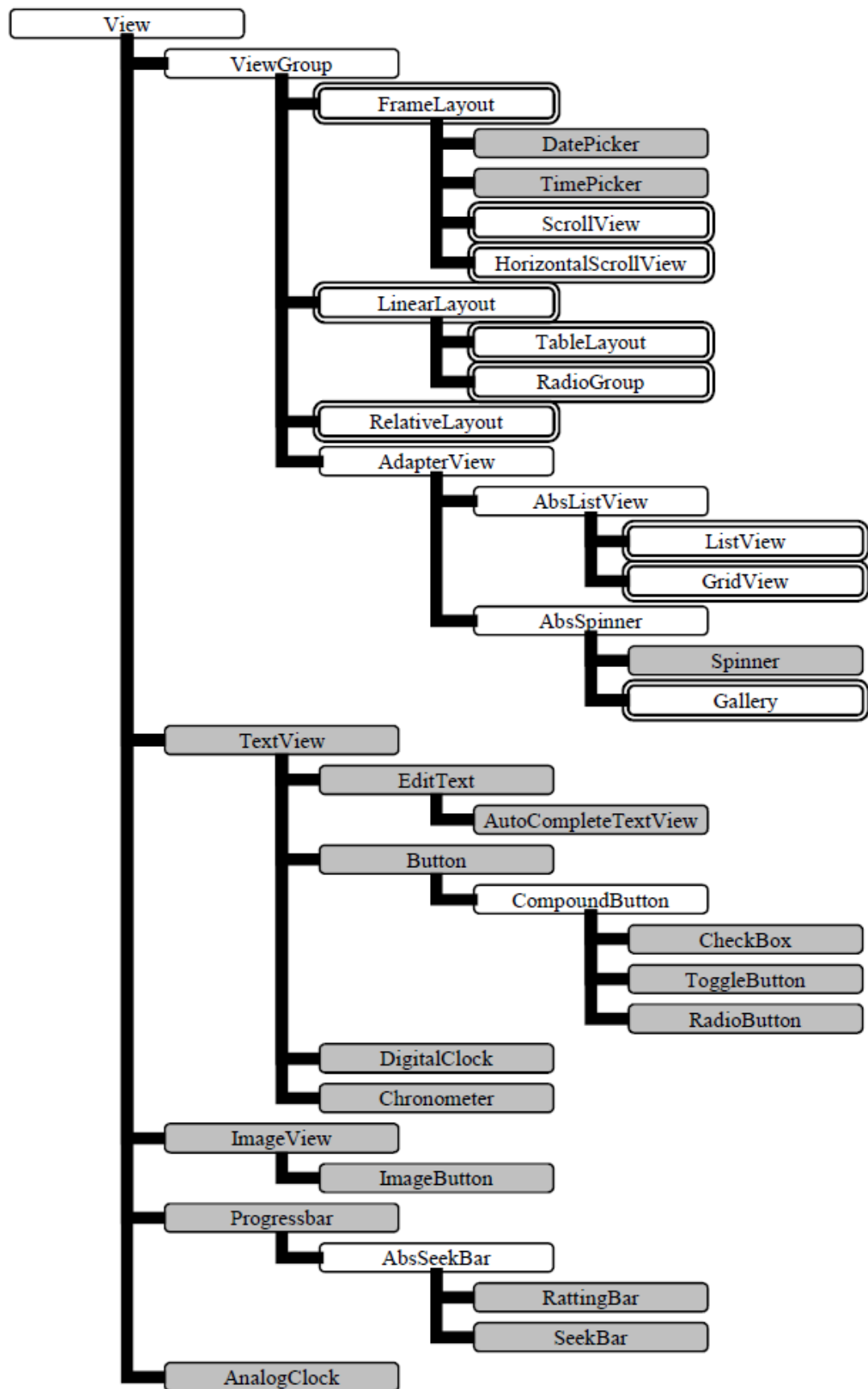






Figure 3 Hiérarchie des classes pour les interfaces [2]

V. Les gabarits ou les conteneurs (les layouts)

Les conteneurs sont utilisés pour placer des éléments d'interface ou d'autres conteneurs. Ils héritent tous de `ViewGroup`. Leurs propriétés sont donc au moins celles décrites ci-dessus pour `View` et `ViewGroup` auxquelles viennent s'ajouter des propriétés spécifiques décrites dans cette partie. [2]










Des `ViewGroup` particuliers sont prédéfinis: ce sont des gabarits (layout) qui proposent une prédisposition des objets graphiques:

Image du Layout	Intitulé du Layout	Usage
	<code>FrameLayout</code>	Placement en haut à gauche. Si l'on place plusieurs éléments ils se superposent, généralement <code>FrameLayout</code> est utilisé pour ne placer qu'un seul élément.
	<code>LinearLayout</code>	Place les éléments les uns à côté des autres horizontalement ou verticalement.
	<code>TableLayout</code>	Place les éléments en lignes et colonnes (Matricielle)
	<code>RelativeLayout</code>	Place des éléments les uns relativement aux autres

VI. Eléments graphiques

Un gabarit peut contenir des éléments graphiques simples, ou des groupes ou même d'autres gabarits.

VI.1. Eléments simples

Image de l'élément graphique	Intitulé	Usage
	<code>TextView</code>	Affiche de texte a priori non éditable
	<code>EditText</code>	Affiche de texte éditable
	<code>Button</code>	Affiche un bouton de commande
	<code>ToggleButton</code>	Affiche un bouton à 2 états allumé ou éteint
	<code>CheckBox</code>	Affiche une case à cocher
	<code>RadioButton</code>	Affiche un bouton radio
	<code>DatePicker</code>	Affiche un calendrier
	<code>SeekBar</code>	Affiche une barre horizontale dotée d'un curseur permettant de modifier la valeur.
	<code>ImageView</code>	Affiche une zone dans laquelle s'affiche une image.

VI.2. Eléments de type liste

VI.2.1. Spinner

Propose une liste de choix. Le choix actuellement sélectionné est affiché, la flèche permet de faire apparaître les autres possibilités sous la forme d'un `RadioGroup`.

A. Propriétés XML

- `android:gravity="g"` (où `g` peut prendre les valeurs : `top`, `bottom`, `left`, `right`, `center_vertical`, `fill_vertical`, `center_horizontal`, `fill_horizontal`, `center`, `fill`) définit comment se placent les éléments de choix.
- `android:prompt="texte"` définit le titre de la fenêtre qui s'ouvre lorsque l'on fait un choix
- `android:entries="@array/maliste"` définit le contenu de la liste à partir d'une ressource de type tableau de chaînes de caractères mise dans un fichier xml placé dans `res/values/` sous la forme :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="maliste">
        <item>Mercure</item>
        <item>Venus</item>
        <item>Terre</item>
        <item>Mars</item>
    </string-array>
</resources>
```

B. Méthodes de la classe `android.widget.Spinner`

- Construction
 - `Spinner(Context)` le paramètre est généralement l'activité elle-même
- Contenu
 - `setAdapter(ArrayAdapter)` Permet de remplir la liste de choix. La façon de procéder est décrite dans `ListView`.
- Sélection
 - `getCount()` renvoie le nombre de choix
 - `setSelection(int)` sélectionne un élément de la liste par son rang (à partir de 0)
 - `getSelectedItem()` renvoie l'objet sélectionné
 - `getSelectedItemPosition()` renvoie l'indice de l'objet sélectionné

VI.2.2. ListView

`ListView` place les éléments en liste avec un ascenseur vertical si nécessaire. `ListView` est normalement utilisé pour afficher des éléments textuels éventuellement accompagnés d'une case à cocher lorsqu'il s'agit d'une liste à choix multiples. Il est toutefois possible d'y afficher des éléments plus complexes en utilisant un gestionnaire de contenu.

A. Propriétés XML

- Contenu de type texte

`android:entries="@array/maliste"` définit le contenu de la liste à partir d'une ressource de type tableau de chaînes de caractères définie dans un fichier xml placé dans **res/values/** sous la forme suivante :

```
<string-array name="maliste">
<item>premier élément</item>
<item>deuxième élément</item>
...
<item>dernier élément</item>
</string-array>
```

- Séparateurs
 - `android:divider="couleur"` définit la couleur des séparateurs ou
 - `android:divider="@drawable/monimage"` pour utiliser une image. L'image est placée dans `res/drawable/` et s'appelle `monimage.x` (où `x` est `png`, `jpg`, ou `gif`).
 - `android:dividerHeight="unité"` définit la hauteur des séparateurs (si c'est une image elle sera déformée).
- Type de choix
 - `android:choiceMode="c"` (où `c` peut prendre les valeurs : `none`, `singlechoice`, `multipleChoice`) pour indiquer le mode de choix dans la liste (aucun, un seul, plusieurs).

B. Méthodes de la classe `android.widget.ListView`

- Construction
 - `ListView(Context)` le paramètre est généralement l'activité elle-même
- Contenu

Le contenu d'une `ListView` peut être défini de façon statique par la propriété `android:entries` dans le fichier `xml`. Lorsque l'on souhaite avoir un contenu dynamique on utilise un `ArrayAdapter` (collection) que l'on remplit (méthodes `add` ou `insert` du `ArrayAdapter`) et que l'on associe à la `ListView` par la méthode `setAdapter(ArrayAdapter)`.

La classe `ArrayAdapter` possède les méthodes suivantes :

- Construction

`ArrayAdapter(Context, type)` le premier paramètre est généralement l'activité elle-même, le second paramètre peut être une valeur prédéfinie :

- `android.R.layout.simple_list_item_1` pour une liste à choix unique ou
- `android.R.layout.simple_list_item_multiple_choice` pour une liste à choix multiple (dans ce cas une case à cocher apparaît à côté de chaque élément de la liste). Le second paramètre peut également être l'identificateur d'un widget `TextView` personnalisé défini dans un fichier `xml` placé dans **res/layout** et désigné par `R.layout.nom_du_fichier_xml`

- Eléments

- `add(Object)` pour ajouter à la fin
- `insert(Object, int)` pour insérer un élément au rang donné en 2^{ème} paramètre
- `clear()` pour enlever tous les éléments
- `remove(Object)` pour enlever un élément
- `getCount()` renvoie le nombre d'éléments
- `getItem(int)` renvoie l'élément (`Object`) dont le rang est donné en 2^{ème} paramètre
- `getPosition(Object)` renvoie le rang de l'objet désigné en paramètre

VII. Les boîtes de dialogue

VII.1. Toast

//Création d'un Toast

- `Toast.makeText(Context, String, int)` cette méthode renvoie l'objet de classe `Toast` créé. Le premier paramètre est généralement l'activité elle-même, le deuxième paramètre est le message à afficher, le dernier paramètre indique la durée d'affichage les seules valeurs possibles sont : `Toast.LENGTH_SHORT` (2 secondes) ou `Toast.LENGTH_LONG` (5 secondes).

//Positionnement d'un Toast

- `setGravity(int, int, int)` cette méthode doit être appelée avant l'affichage par `show` pour indiquer où s'affichera le message. Le premier paramètre sert à placer le message par rapport à l'écran. Il peut prendre l'une des valeurs définies dans la classe `Gravity` soit : `Gravity`. (`TOP`, `BOTTOM`, `LEFT`, `RIGHT`, `CENTER_VERTICAL`, `FILL_VERTICAL`, `CENTER_HORIZONTAL`, `FILL_HORIZONTAL`, `CENTER`, `FILL`). Les deux paramètres suivants indiquent le décalage horizontal et vertical (en pixels).

//Affichage d'un Toast

- `show()` cette méthode affiche le message pour la durée définie lors de sa création.

VII.2. AlertDialog

Le rôle `AlertDialog` est utilisé pour notifier à l'utilisateur des informations urgentes qui requièrent son attention immédiate. Comme le nom l'indique, `AlertDialog` est un type de boîte de dialogue.

//Création d'un Builder

```
AlertDialog.Builder b = new AlertDialog.Builder(AlertDialogActivity.this);
```

//Changer titre

```
b.setTitle("Save File...");
```

//Changer message

```
b.setMessage("Do you want to save this file?");
```

//Changer icône

```
b.setIcon(R.drawable.save);
```

//Bouton positif

```
b.setPositiveButton("YES", new DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int which) {  
        // User pressed YES button. Write Logic Here  
        Toast.makeText(getApplicationContext(), "You clicked on YES",  
            Toast.LENGTH_SHORT).show();  
    }  
});
```

//Bouton négatif

```
b.setNegativeButton("NO", new DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int which) {  
        // User pressed No button. Write Logic Here
```

```

        Toast.makeText(getApplicationContext(), "You clicked on NO",
        Toast.LENGTH_SHORT).show();
    }
});
//Bouton neutre
b.setNeutralButton("Cancel", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // User pressed Cancel button. Write Logic Here
        Toast.makeText(getApplicationContext(), "You clicked on Cancel",
        Toast.LENGTH_SHORT).show();
    }
});
//Creation et affichage de l'AlertDialog
AlertDialog alertDialog = b.create();
alertDialog.show();

```

Plan

I.	INTRODUCTION	1
II.	STRUCTURE D'UN PROJET ANDROID	1
III.	LA CLASSE ACTIVITY	1
IV.	PROPRIETES ET CLASSES DE BASE DES INTERFACES	2
IV.1.	LES UNITES	2
IV.2.	LES COULEURS	2
IV.3.	LA CLASSE VIEW	2
IV.4.	LA CLASSE VIEWGROUP	2
IV.5.	HIERARCHIE DES SOUS CLASSES DE VIEW	3
V.	LES GABARITS OU LES CONTENEURS (LES LAYOUTS)	4
VI.	ELEMENTS GRAPHIQUES	4
VI.1.	ELEMENTS SIMPLES	4
VI.2.	ELEMENTS DE TYPE LISTE	4
VI.2.1.	Spinner	4
VI.2.2.	ListView	5
VII.	LES BOITES DE DIALOGUE	6
VII.1.	TOAST	6
VII.2.	ALERTDIALOG	6

Références

- [1] Damien Guignard, Julien Chabre, Emmanuel Robles, Programmation Android de la conception au déploiement avec le SDK Google Android 2, Eyrolles, ISBN : 978-2-212-12587-0
- [2] Développement d'applications pour Android, M. Dalmau, IUT de Bayonne-Pays Basque
- [3] Jean-Francois Lalande, Développement sous Android, November 2012 - Version 2, Ensi de Bourges - Filière STI
- [4] https://developer.mozilla.org/fr/docs/Accessibilit%C3%A9/ARIA/Techniques_ARIA/Utiliser_le_role_alertdialog
- [5] <http://www.androidhive.info/2011/09/how-to-show-alert-dialog-in-android/>
- [6] <https://www.supinfo.com/articles/single/2844-activites-android>
- [7] <https://o7planning.org/fr/10423/tutoriel-android-ui-layouts>
- [8] http://www.lirmm.fr/~fmichel/old/ens/android/cours/android_GUI_Basic.pdf
- [9] <http://findnerd.com/list/view/What-should-a-designer-must-know-before-starting-design-for-an-android-app-/2969/>
- [10] <https://vinsol.com/blog/2014/11/20/tips-for-designers-from-a-developer/>