Applications mobiles

Kamal Mehaoued

Université de Béjaia

Troisième année licence informatique

2017/2018

Table des matières

Introduction

- Les interfaces prennent une place de plus en plus importante dans le choix d'une application par l'utilisateur
- Le succès d'une application auprès d'un utilisateur dépend du succès de son interface graphique
- Il ne peut y avoir de bonne application sans une bonne interface graphique

Les Vues

le composant graphique élémentaire sous android est la vue classe View. Créer une vue revient donc à créer un objet graphique.

- Une vue constitue le composant graphique élémentaire dans la plate-forme android
- Tous les composants graphiques (boutons, cas à cocher ...) héritent de la classe View
- Créer une vue revient à créer un objet graphique
- Plusieurs vues peuvent être regroupées dans une seule structures arborescente ViewGroup
- Cette structure peut à son tour regrouper d'autres éléments ViewGroup

Les gabarits

- Un gabarit, Layout, est une extension de la classe ViewGroup.
- C'est un conteneur qui aide à positionner les objets au sein d'une interface.
- Les gabarits (layout) peuvent s'imbriquer les uns dans les autres.
- Un gabarit parent contient un gabarit enfant
- un gabarit enfant est contenu dans un gabarit parent

Les gabarits

Selon le gabarit utilisé, les composants qui s'y trouvent peuvent être disposés différemment.

LinearLayout

Permet d'aligner de gauche à droite ou du haut vers le bas, les éléments qui s'y trouveront. Selon la valeur de la propriété orientation, nous pouvons indiquer au gabarit dans quel sens afficher ses composants enfants.

- Si la valeur de orientation est horizontal, l'affichage se fait de gauche à droite
- Si la valeur de orientation est vertical, l'affichage se fait du haut vers le bas

Les gabarits

RelativeLayout

Les composants s'y trouvant seront disposés les uns par rapport aux autres. Le premier composant(enfant) sert de référence aux autres

FrameLayout

Chaque enfant est positionné dans le coin supérieur gauche de l'écran et affiché au dessus des enfants précédents en les cachant partiellement ou entièrement. Ce gabarit est utilisé pour l'affichage d'un élément.

TableLayout

Permet de positionner les composants(enfants) d'un gabarit en ligne et colonnes comme dans un tableau

Exemple d'un gabarit linéaire en XML

```
<!– Mon premier gabarit –>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent"
        >
</LinearLayout>
```

Les gabarits

Les gabarits possèdent des propriétés communes telles que :

- layout_width : permet de spécifier le comportement de remplissage en largeur. Sa taille peut être précisée explicitement
- layout_height : permet de spécifier le comportement de remplissage en hauteur . Sa taille peut être précisée explicitement.
- fill_parent : spécifie que le gabarit doit prendre toute la place disponible sur la largeur et la hauteur du gabarit père. Si le gabarit père est l'écran lui même alors le gabarit enfant va occuper tout l'écran.
- wrap_content : Dans ce cas le gabarit prend la place qui lui est nécessaire.

Création d'interface utilisateur

La création d'une interface se fait par la création de deux éléments

- Définition de l'interface graphique utilisateur de façon déclarative dans un fichier XML
- Définition de la logique de fonctionnement de l'interface dans une classe d'activité
- Cette séparation fait de sorte qu'un spécialiste en graphique n'interfère pas avec le code du développeur

Définition d'interface

Une bonne pratique est de définir l'interface graphique dans la déclaration xml. Sachant qu'il est possible de la définir dans le code du développeur.
Exemple :

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent"
<TextView
        android :layout_width="fill_parent"
        android :layout_height="wrap_content"
        android :id="@+id/monText"
        />
</LinearLayout>
```

Définition d'interface

L'équivalent dans le code du développeur :

```
import android.app.Activity;
import android.os.Bundle;
public class Main extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }
}
```

Récupérer et utiliser un élément de l'interface

```
import android.app.Activity ;
import android.os.Bundle ;
import android.widget.TextView ;
public class Main extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState) ;
                setContentView(R.layout.main) ;
                TextView monTexte =
(TextView)findViewById(R.id.monText) ;
                monTexte.setText("Bonjour tout le monde !") ;
        }
}
```

L'équivalent dans le code du développeur sans la définition xml :

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
public class Main extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                TextView monTextView = new TextView(this);
                setContentView(monTextView);
                monTextView.setText("Bonjour tout le monde!");
        }
}
```

Créer une interface sans définition XML avec un gabarit

```
import android.app.Activity ;
import android.os.Bundle ;
import android.widget.LinearLayout ;
import android.widget.TextView ;
public class Main extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState) ;
                LinearLayout monLinearLayout = new LinearLayout(this) ;
                monLinearLayout.setOrientation(LinearLayout.VERTICAL) ;
                TextView monTextView1 = new TextView(this) ;
                TextView monTextView2 = new TextView(this) ;
                monLinearLayout.addView(monTextView1) ;
                monLinearLayout.addView(monTextView2) ;
                setContentView(monLinearLayout) ;
                monTextView1.setText("Bonjour tout le monde !") ;
                monTextView2.setText("Ceci est mon 2eme texte") ;
```

Les composants graphiques

Le fichier .java de notre activité est dorénavant celui-ci :
Avec main.xml, le fcihier ou est déclarée notre interface graphique.

```
import android.app.Activity ;
import android.os.Bundle ;
import android.widget.LinearLayout ;
import android.widget.TextView ;
public class MyActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState) ;
                setContentView(R.layout.main) ;
        }
}
```

zone de texte

TextView est une vue basique, une simple zone de texte :
Considérons l'exemple suivant avec quatre TextView positionnés verticalement.

Figure – exemple de zone de texte

zone de teste suite

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <TextView
                android :text="Plain"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
        <TextView
                android :text="Serif"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"
                android :typeface="serif" />
```

zone de texte suite

```
<TextView
        android :text="Bold"
        android :layout_width="wrap_content"
        android :layout_height="wrap_content"
        android :textStyle="bold" />
<TextView
        android :text="Italic"
        android :layout_width="wrap_content"
        android :layout_height="wrap_content"
        android :textStyle="italic"
/>
</LinearLayout>
```

zone de texte suite

en exécutant le code de l'activité précédente, nous obtenons :

Figure – exemple de zone de texte

zone d'édition

L'EditText est une extension du TextView.Cette vue est une zone d'édition. Considérons l'exemple suivant :

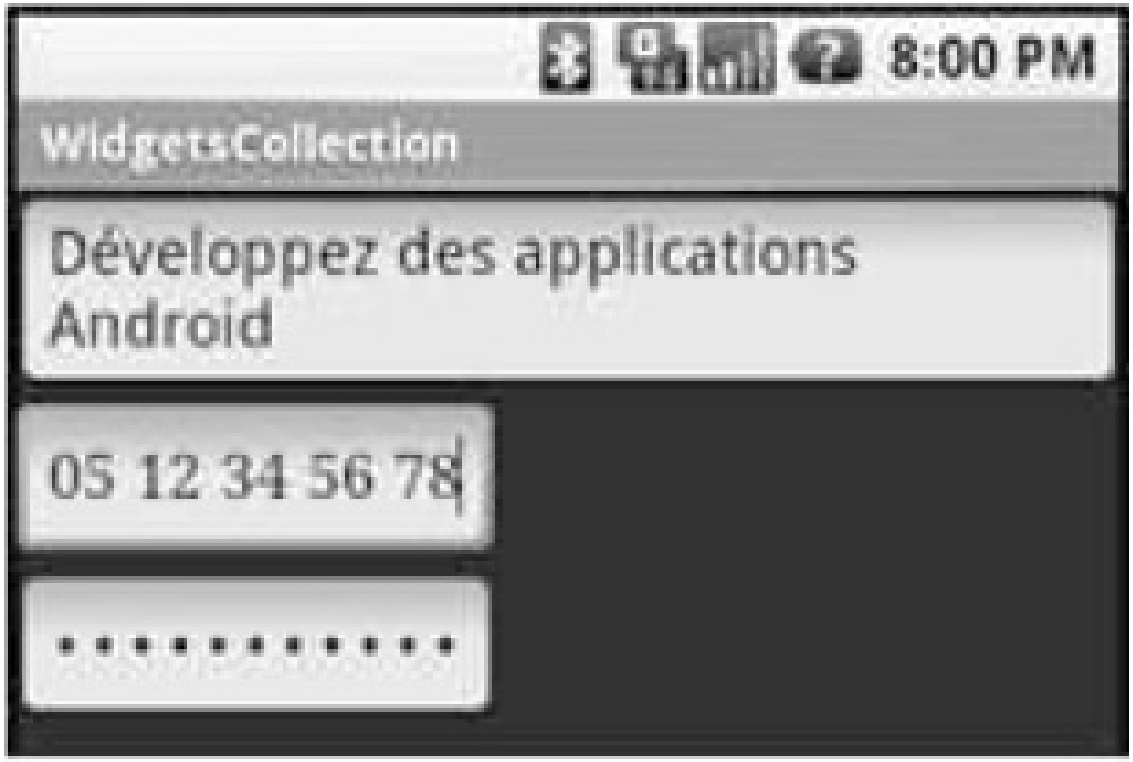

Figure – exemple de zone d'édition

zone d'édition suite

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <EditText
                android :id="@+id/title"
                android :text="android"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
```

zone d'édition suite

```
<EditText
        android :id="@+id/phoneNumber"
        android :text="05 12 34 56 78"
        android :layout_width="wrap_content"
        android :layout_height="wrap_content"
        android :typeface="serif"
        android :phoneNumber="true" />
<EditText
        android :id="@+id/password"
        android :text="monPassword"
        android :layout_width="wrap_content"
        android :layout_height="wrap_content"
        android :password="true" />
</LinearLayout>
```

zone d'édition suite

Toujours en exécutant le code de l'activité, nous aurons :

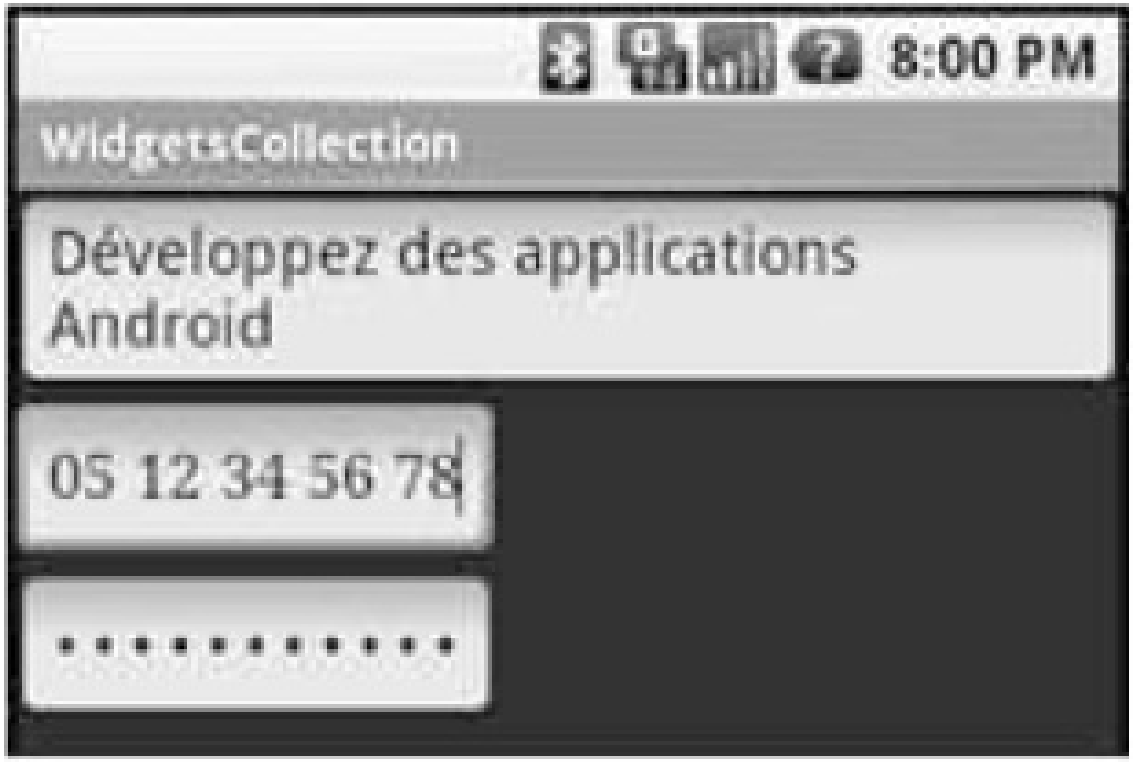

Figure – exemple de zone d'édition

Case à cocher

La classe CheckBox est une case à cocher.
Considérons l'exemple suivant des cases à cocher :

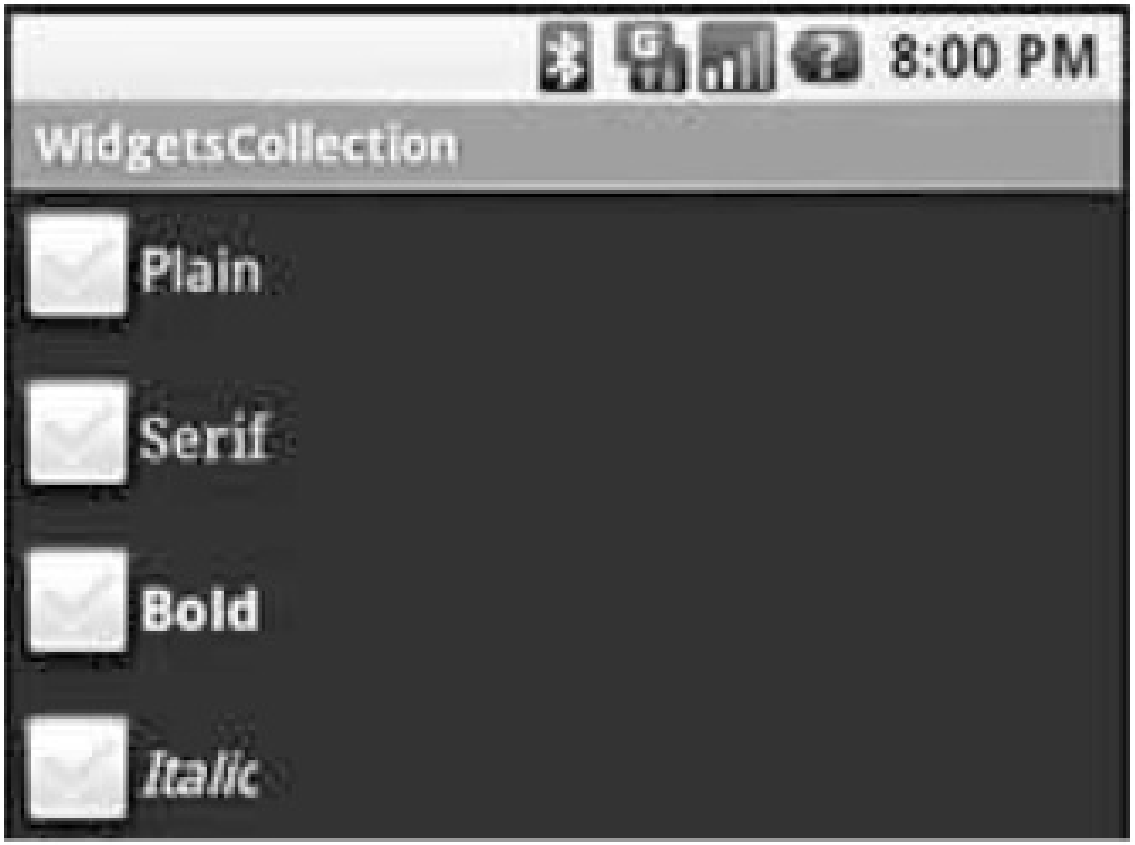

Figure – exemple de case à cocher

Case à cocher suite

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <CheckBox
                android :id="@+id/plain_cb"
                android :text="Plain"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
        <CheckBox
                android :id="@+id/serif_cb"
                android :text="Serif"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"
```

Case à cocher suite

```
<CheckBox
        android :id="@+id/bold_cb"
        android :text="Bold"
        android :layout_width="wrap_content"
        android :layout_height="wrap_content"
        android :textStyle="bold" />
<CheckBox
        android :id="@+id/italic_cb"
        android :text="Italic"
        android :layout_width="wrap_content"
        android :layout_height="wrap_content"
        android :textStyle="italic" />
</LinearLayout>
```

Case à cocher suite

Toujours en exécutant le code de l'activité, nous aurons :

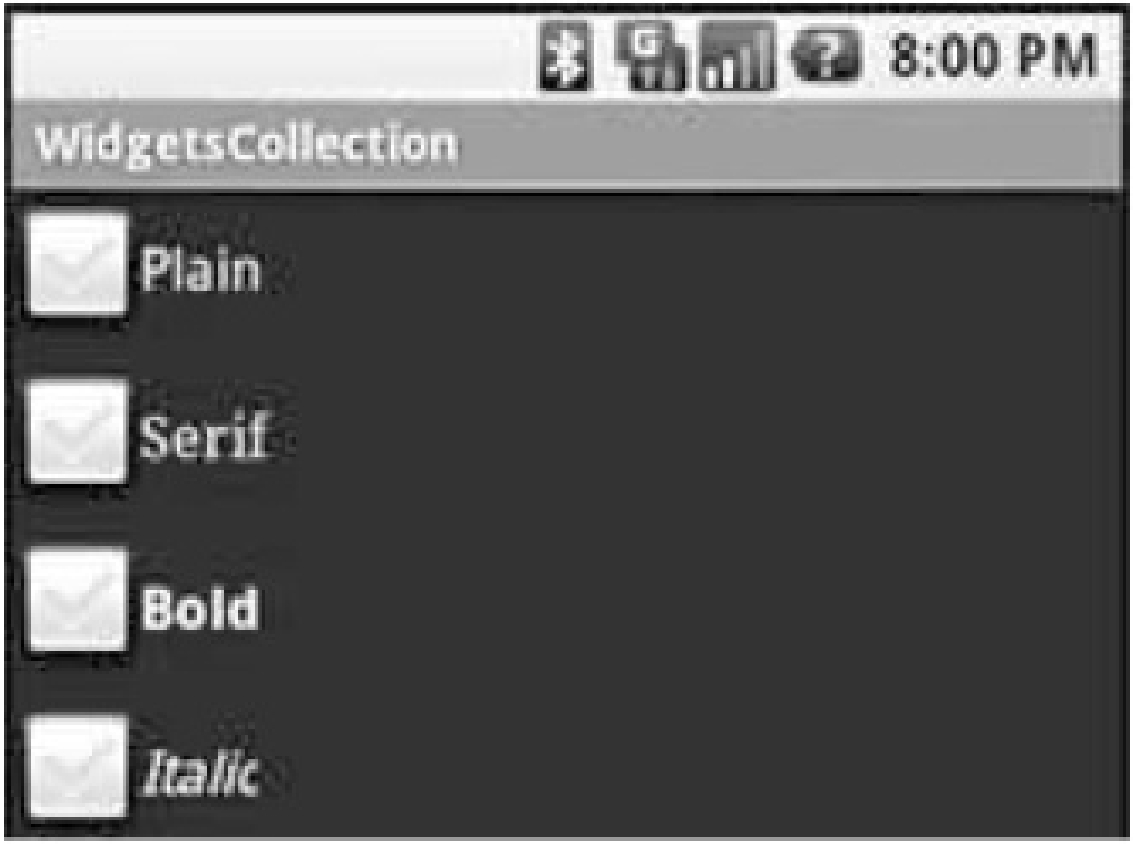

Figure – exemple de cases à cocher

Bouton

La classe Button permet de créer un bouton qui pourra être actionné par l'utilisateur afin de déclencher un événement.
Considérons l'exemple suivant :

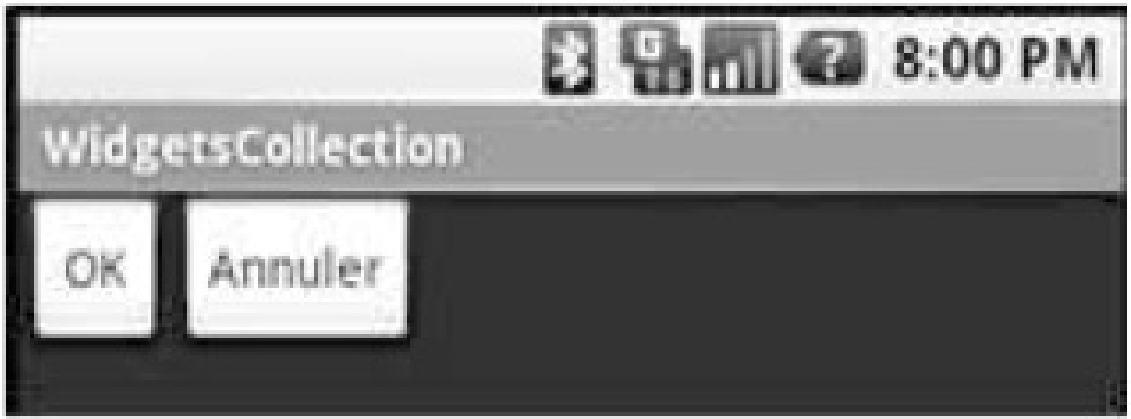

Figure – exemple de boutton

Bouton suite

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="horizontal"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <Button
                android :id="@+id/okButton"
                android :text="OK"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
        <Button
                android :id="@+id/CancelButton"
                android :text="Annuler"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
```

Bouton suite

Toujours en exécutant le code de l'activité, nous aurons :

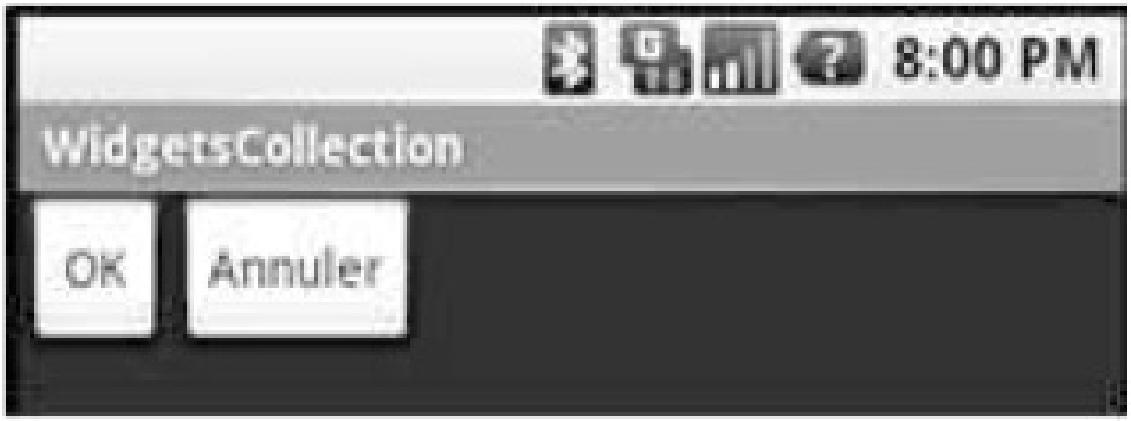

Figure – exemple de bouton

ToggleBouton

Un Toggle button est un bouton poussoir avec deux états vrai ou faux. A chaque état correspond un texte visuel. Considérons l'exemple suivant :

Figure – exemple de toggle boutton

Toggle button

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns :
        android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <ToggleButton
                android :id="@+id/toggle1"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"
                android :textOn="Allumé"
                android :textOff="Eteint" />
        <ToggleButton android :id="@+id/toggle2"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"
                android :textOn="On"
```

Date Picker

Elle offre une vue facilitant la saisie d'une date

Figure – Date Picker

Date Picker

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
        xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <DatePicker
                android :id="@+id/date"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"/>
</LinearLayout>
```

Time Picker

Il a le même focntionnement que le DatePicker mais il est destiné à la saisie de l'heure.

Figure – Time Picker

Time Picker

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <TimePicker
                android :id="@+id/time"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
</LinearLayout>
```

ImageView

L'ImageView est un vue dont la représentation est une image. La source de l'image peut provenir du répertoire layout ou être référencée par une URI. ImageView intègre des fonctionnalités de transformation.

Figure – Exemple d'Image View

ImageView

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <ImageView android :id="@+id/image"
                android :src="@drawable/logo"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"
                android :layout_gravity="center"/>
</LinearLayout>
```

ImageButton

ImageButton est une sous-classe d'ImageView laquelle en plus de pouvoir afficher une image, elle est est capable de jouer le rôle d'un bouton et être en interaction avec l'utilisateur.

Figure – Exemple de Bouton image

ImageButton

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <ImageButton
                android :id="@+id/image"
                android :src="@drawable/logo"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"
                android :layout_gravity="center"/>
</LinearLayout>
```

La classe AnalogClock est une simple horloge analogique permettant d'afficher l'heure actuelle.

Figure – Exemple d'horloge analogique

AnalogClock

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <AnalogClock
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
</LinearLayout>
```

La classe DigitalClock est une simple horloge numérique permettant d'afficher l'heure actuelle.

Figure – Exemple d'horloge numérique

DigitalClock

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <DigitalClock
                android :layout_width="wrap_content"
                android :layout_height="wrap_content" />
</LinearLayout>
```

RatingBar

Le RatingBar est un composant graphique permettant une notation de façon visuelle

Figure – Exemple d'un Rating Bar

RatingBar

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout
xmlns :android="http ://schemas.android.com/apk/res/android"
        android :orientation="vertical"
        android :layout_width="fill_parent"
        android :layout_height="fill_parent">
        <RatingBar
                android :id="@+id/gallery"
                android :layout_width="wrap_content"
                android :layout_height="wrap_content"/>
</LinearLayout>
```