Mobilité: Programmation Android

Erick STATTNER

Maître de Conférences en Informatique

Université des Antilles

erick.stattner@univ-antilles.fr

www.erickstattner.com

Description de l'enseignement

Objectifs pédagogiques:

- Se familiariser à la Programmation d'applications pour mobile
- Maitriser les principes autour des applications Android
- Concevoir des applications graphiques sous Android
- Mettre en place la persistance des données

Organisation:

- 30h
- 1 CC + 1 CT

Sommaire

Chapitre III. Les interfaces

1. Les Layouts
2. Les Widgets
3. Exemples d'interfaces
4. Les Logs

III. Les interfaces

Layout

Les layouts

- Permettent de mettre en place les interfaces des applications
- Conteneurs qui *"facilitent"* l'organisation et la disposition des éléments
- Tous les **layout** héritent de la classe **ViewGroup**
- Tous les **layouts** sont paramétrables avec un ensemble d'attributs
- Deux manières de procéder
  - Création statique
    - S'effectue en XML
  - Création dynamique
    - S'effectue en JAVA
- On peut combiner les deux pour avoir des interfaces riches

III. Les interfaces

Layout

FrameLayout

- Layout le plus simple
- Représente un espace à remplir avec l'objet de votre choix
- Par défaut, l'objet se positionne en haut à gauche
  - Modifier avec attribut **gravity**
- Ne peut afficher qu'un seul objet
- Si plusieurs objets sont ajoutés, le ***FrameLayout*** les empile, le dernier élément ajouté est l'élément visible.

III. Les interfaces

Layout

LinearLayout

- Permet d'aligner les éléments linéairement
- Qqs attributs possibles
  - **android:orientation**
    Type alignement (horizontale ou verticale)
  - Gravité des éléments
    - **android:layout_gravity**
      positionnement de l'élément dans son conteneur (left, right, …)
    - **android:gravity**
      Positionnement du contenu de l'élément (left, right, …)
  - **android:layout_weight**
    Poids des éléments, qui correspond à un rapport d'échelle.
    Permet à des éléments d'occuper plus au moins d'espace.

III. Les interfaces

Layout

TableLayout

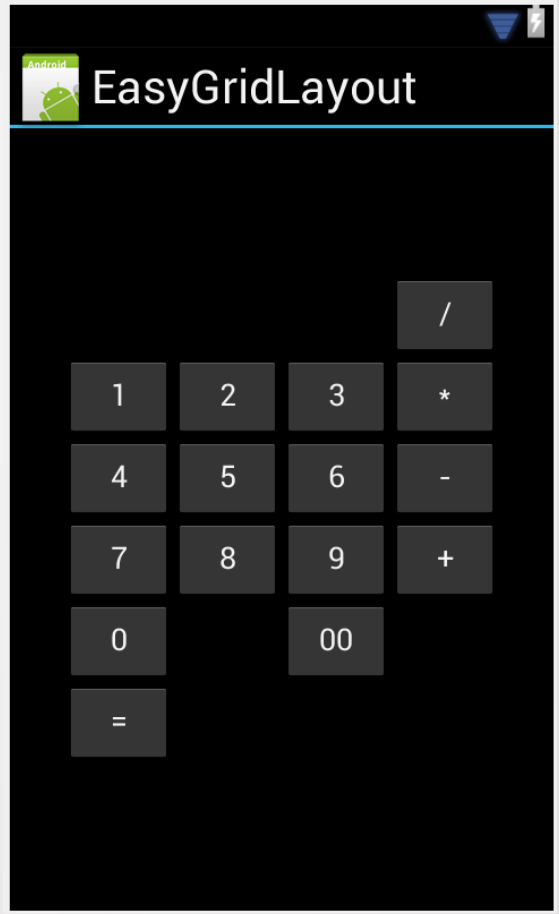
- Positionne les éléments selon une grille
- Une cellule peut rester vide
- Cellules n'ont pas nécessairement la même largeur
- Utiliser **TableRow** pour créer des lignes
- Qqs attributs possibles
  - **android:padding**
    Pour spécifier l'espacement interne des éléments
  - **android:marging**
    Pour spécifier l'espacement externe des éléments

III. Les interfaces

Layout

GridLayout

- Permet d'organiser les éléments en ligne et en colonne
- Etaler une case sur plusieurs lignes ou colonnes
- Possibilité de laisser des cases vides avec balise **Space**
- Qqs attributs possibles
  - **android:columnCount**
    Pour spécifier le nombre de colonnes du layout
  - **android:layout_columnSpan**
    Pour étendre un élément sur plusieurs colonnes



III. Les interfaces

Layout

RelativeLayout

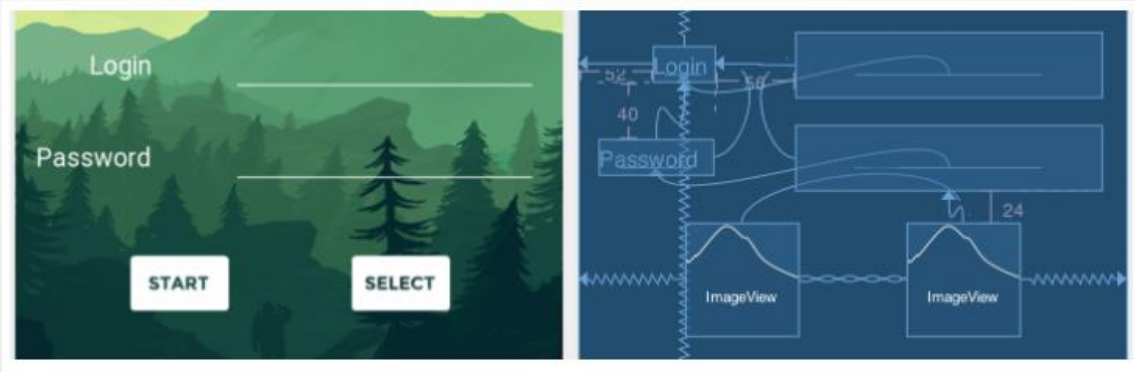
- Permet de placer les éléments en fonction d'un autre
- Soit en fonction
  - De son conteur
  - D'un autre élément
- Qqs attributs possibles
  - **android:layout_alignParent[Top/Bottom/Left/Right]**
    aligner l'élément avec le haut/bas/gauche/droite du conteneur
  - **android:layout_[above/below]**
    Place l'élément au-dessus/en dessous de l'élément référencé
  - **android:layout_to[left/right]Of**
    place un élément à gauche ou à droite de l'élément référencé
  - **android:layout_align[Top/Bottom/Left/Right]**
    aligner des éléments à gauche, en bas, à droite ou à gauche

III. Les interfaces

Layout

ConstraintLayout

- Présenté en mai 2016
- Rassemble les avantages du **LinearLayout** et **RelativeLayout**
  - Elément placé en fonction des autres
  - Appliquer un poids
- Optimisé et Performant pour des interfaces complexes
- Tutorial: http://tutos-android-france.com/constraintlayout-partie-1/



III. Les interfaces

Widgets

Pour aller plus loin

- De plus en plus, interfaces structurées via HTML
- Possibilité de construire l'interface via HTML, CSS et JavaScript
- En Utilisant
  - **TextView** (html simple uniquement)
  - **WebView**
- Attention aux performances !
- Possibilité de solutions hybrides
  - Une partie en natif
  - Une partie via HTML , CSS et JavaScript

III. Les interfaces

Widgets

Les widgets

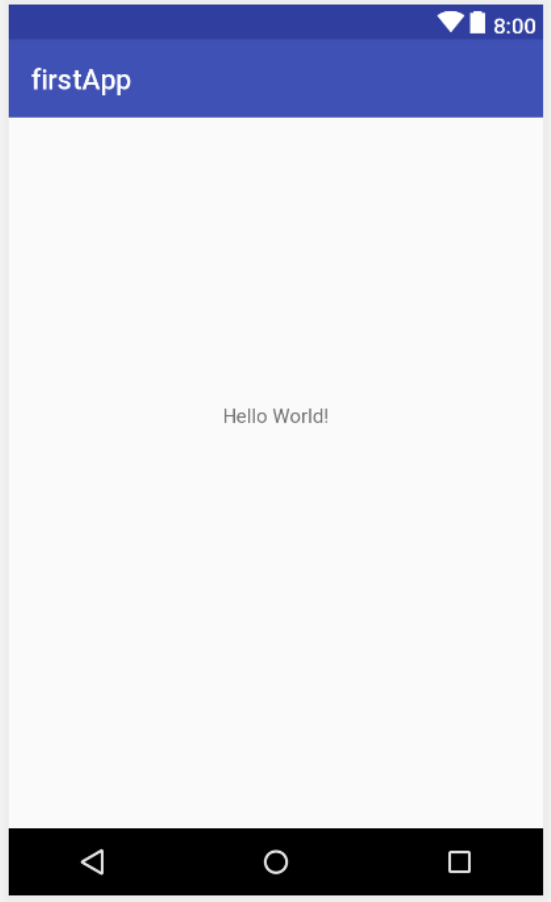
- Les composants (***widgets***) héritent de la classe **view**
- Tous les composants de l'interface sont repérables via un identifiant
- Déclaré sous la forme
  android:id="@+id/nom_identifiant"
  - android:id nom de l'attribut
  - @+id: indique la déclaration d'un nouvel identifiant
  - @nom_identifiant: nom donné à l'élément
- L'élément devient accessible
  - Dans un fichier JAVA
    R.id.nom_identifiant
  - Dans un fichier XML
    @id/nom_identifiant

III. Les interfaces

Widgets

TextView

- Permet d'afficher un texte sur l'interface
- Qqs attributs possibles
  - **android:text**
    Permet de préciser la chaine de caractères à afficher
    Ex.
    android:text="Hello World !";
  - En général le texte sera spécifier à l'aide d'une ressource
    Ex.
    android:text="@string/idDeMonTexte;
  - D'autres attributs possibl pour la police, la couleur, etc.

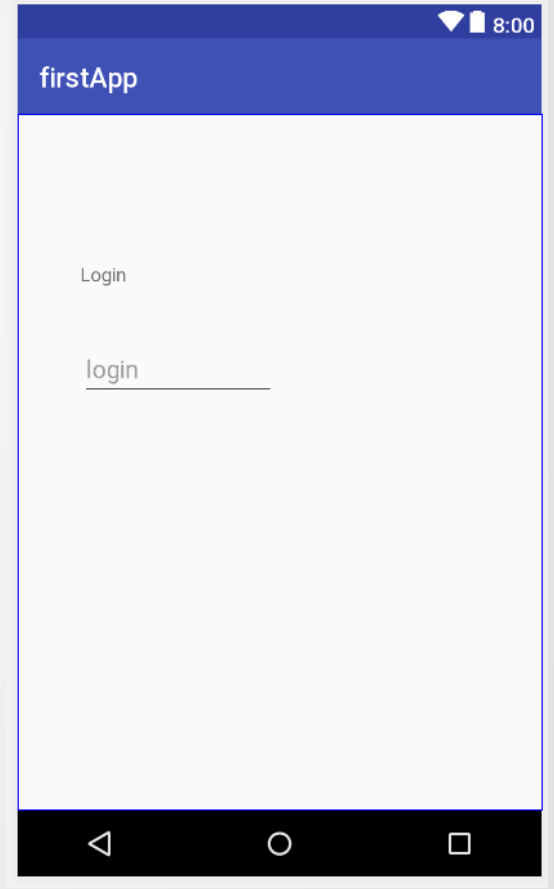


III. Les interfaces

Widgets

EditText

- Permet de mettre en place des champs de saisi
- Qqs attributs possibles
  - **android:hint**
    Permet de indiquer le type de texte attendu
    ex.
    android:hint="login"
  - **android:inputType**
    Permet de préciser le type de clavier à utiliser.
    (text, textPassword, number, …)
    Plusieurs valeurs possibles, séparée par |
    Ex.
    android:inputType="textPassword"

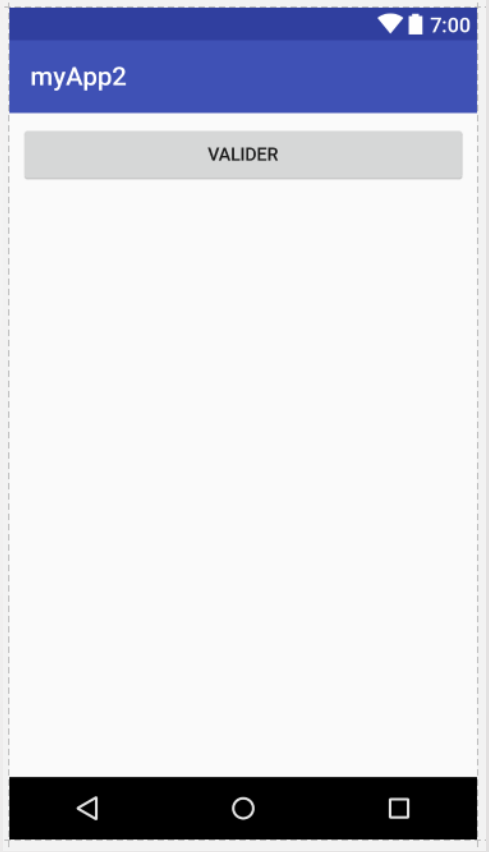


III. Les interfaces

Widgets

Button

- Permet de mettre en place des boutons sur l'interface
- Qqs attributs possibles
  - **android:text**
    pour indiquer le texte à afficher sur le bouton
    ex.
    android:text="valider"

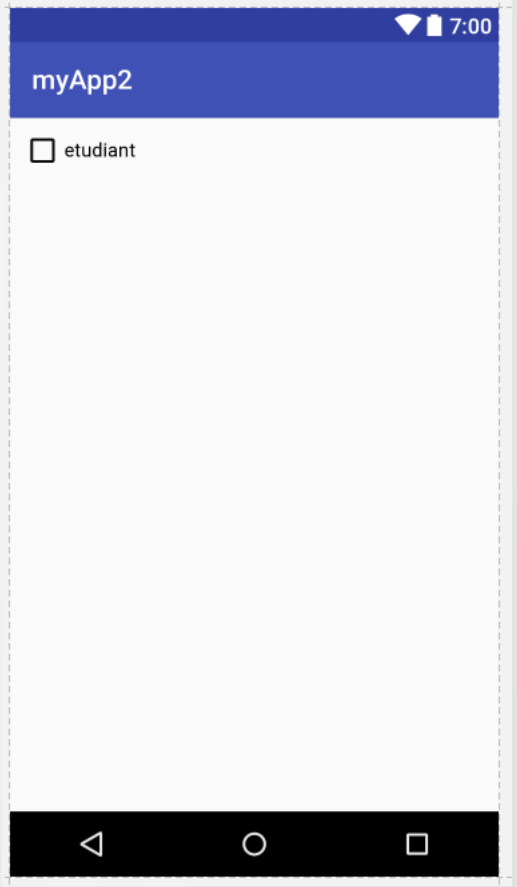


III. Les interfaces

Widgets

Checkbox

- Permet de créer des boites à cocher sur l'interface
- Qqs attributs possibles
  - **android:text**
    Pour spécifier le texte associé
  - **andoid:checked**
    Pour préciser si la case est cochée par défaut ou pas
    ex.
    andoid:checked="false"

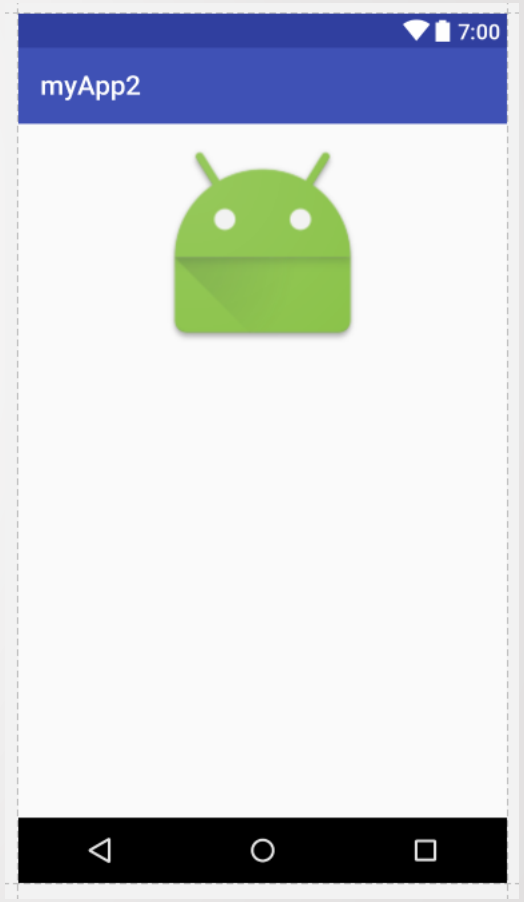


III. Les interfaces

Widgets

ImageView

- Pour utiliser des images dans l'interface
- Qqs attributs possibles
  - **android:src**
    Pour indiquer l'image à utiliser
  - **androind:contentDescription**
    Pour spécifier une petite description de l'image



III. Les interfaces

Widgets

De nombreux autres éléments

- RadioButton
- ProgressBar
- Switch
- SeekBar
- RatingBar
- etc.

III. Les interfaces

Widgets

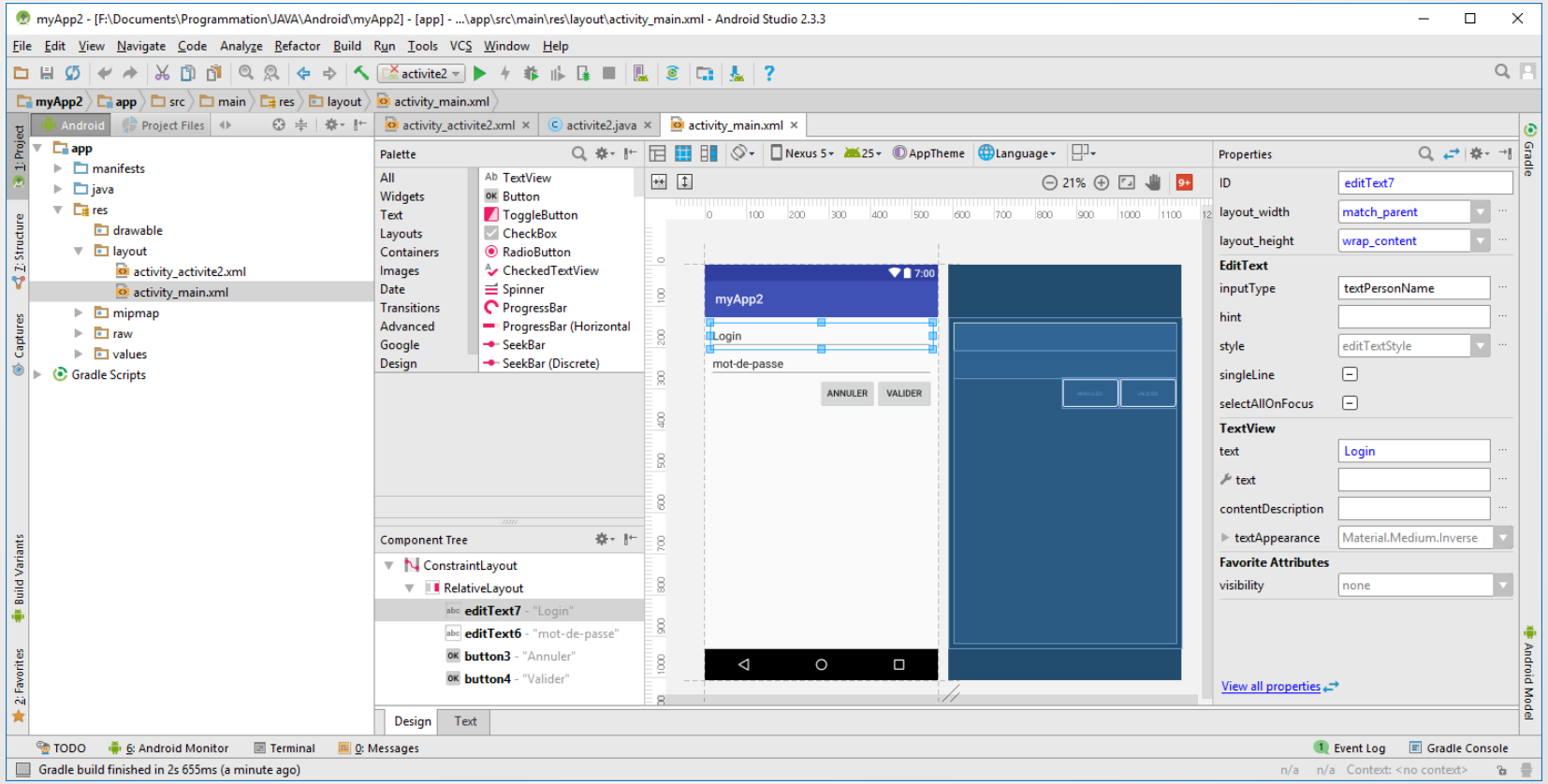
Taille des éléments

- Chaque élément de l'interface a une taille (largeur et hauteur)
- Attributs **layout:width** et **layout:height**
- 3 choix possibles
  - **match_parent**
    taille égale à celle de l'élément parent (celle du conteneur)
    Ex. Un bouton aura la même taille que son conteneur
  - **wrap_content**
    taille égale à celle de son contenu
    Ex. La taille du bouton défini par son étiquette
  - **Une valeur en dp**
    La taille sont précisées en dp (*density-independent pixels*) pour que les éléments conservent le même ratio quel que soit la taille de l'écran.

III. Les interfaces

Exemples d'interfaces

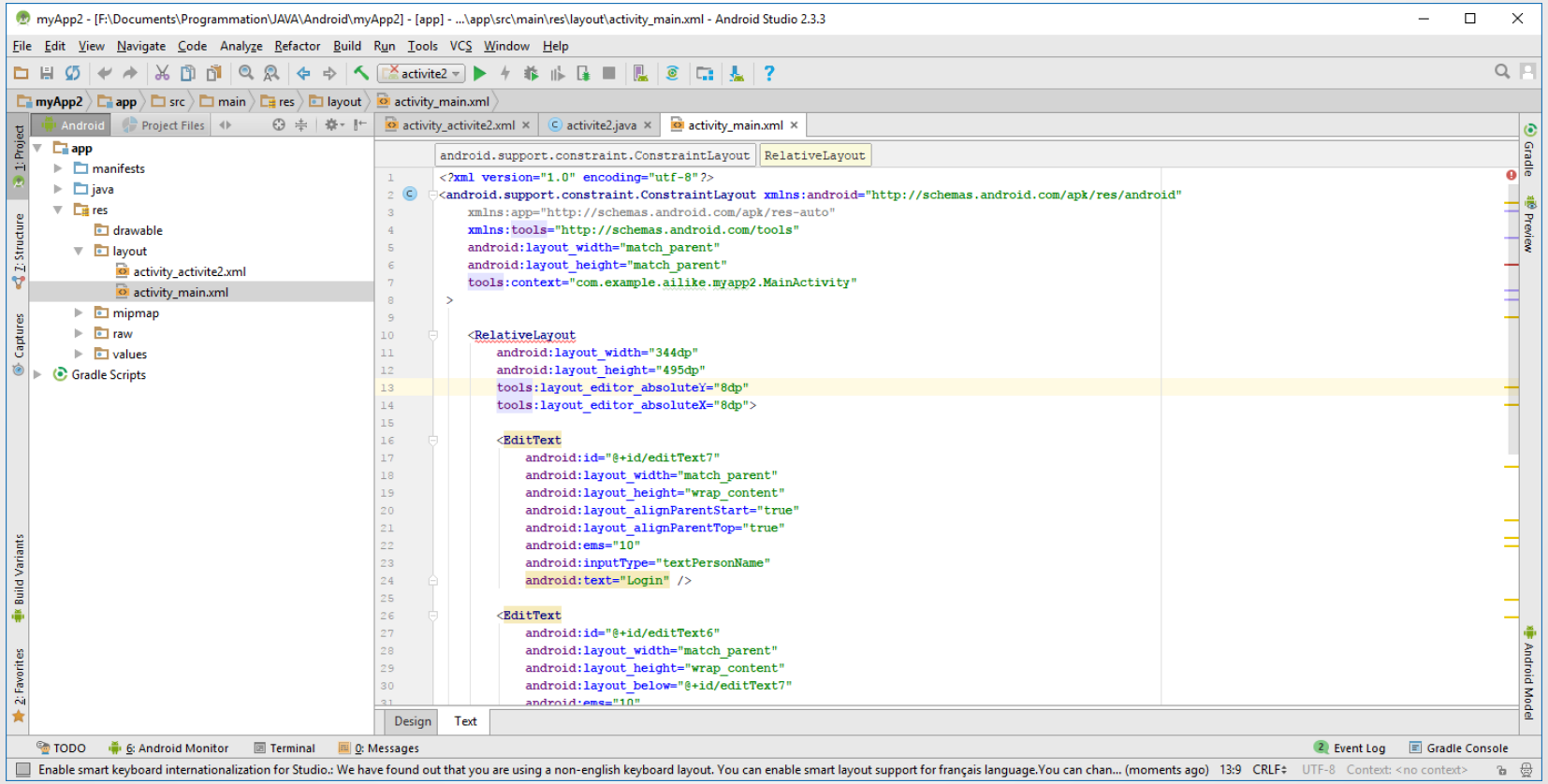
Exemples d'interfaces: Graphique vs Code



III. Les interfaces

Exemples d'interfaces

Exemples d'interfaces: Graphique vs Code



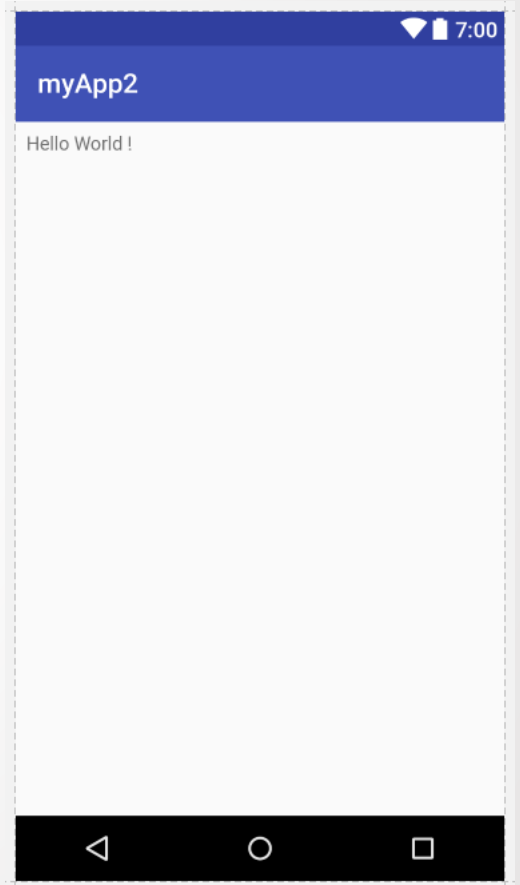
III. Les interfaces

Exemples d'interfaces

Exemples d'interfaces

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Hello World !" />
</LinearLayout>
```



III. Les interfaces

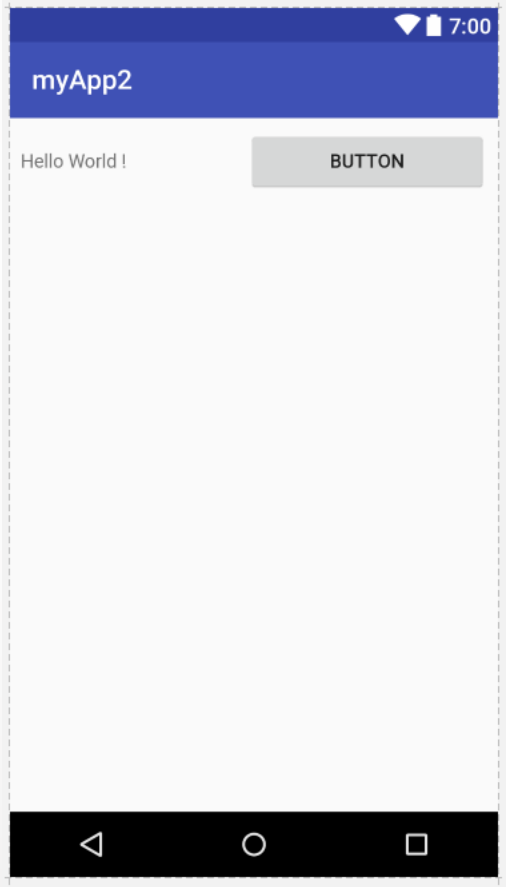
Exemples d'interfaces

Exemples d'interfaces

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Hello World !" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />
</LinearLayout>
```



III. Les interfaces

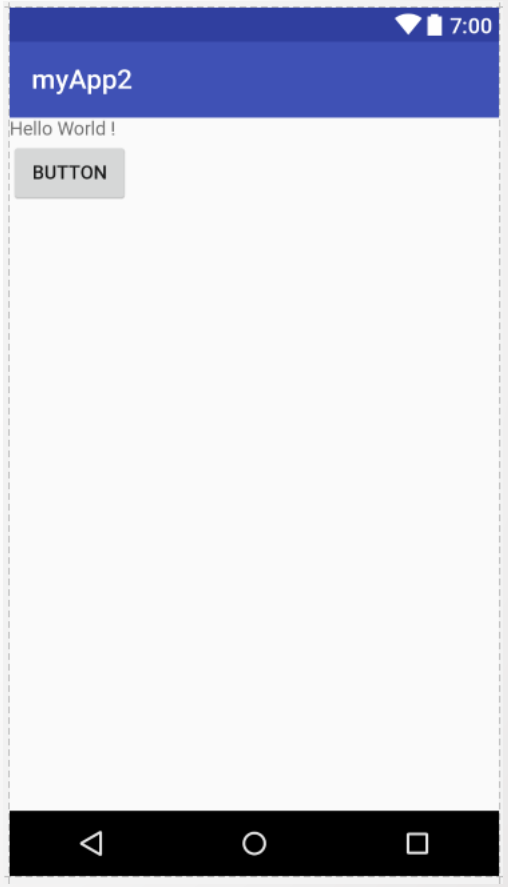
Exemples d'interfaces

Exemples d'interfaces

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

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Hello World !" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button" />
</LinearLayout>
```



III. Les interfaces

Les logs

Gestion des logs

- Indispensable dans le développement sur Android
- Permet de visualiser les messages générés par les applications
- A l'aide de l'outil **Logcat**
  - Voir les messages
  - Filtrer les messages selon tag, application, type de messages, etc.
  - Recherche un message
  - Sauvegarder les logs

III. Les interfaces

Les logs

Gestion des logs

- 6 types de logs
- Pour chaque type, une méthode dédiée qui prend en paramètre un **TAG** et le **MESSAGE**.
  - d (Debug) Afficher un message de debug
    Log.d("MON ACTIVITE", "Le message associé")
  - e (Erreur): Afficher un message d'erreur
    Log.e("MON ACTIVITE", "Le message associé")
  - i (Information): Afficher un message d'information
    Log.i("MON ACTIVITE", "Le message associé")
  - v (verbose): Afficher un message "verbose"
    Log.v("MON ACTIVITE", "Le message associé")
  - w (Warning): Afficher avertissement
    log.w("MON ACTIVITE", "Le message associé")
  - wtf (What a Terrible Failure)
    log.wtf("MON ACTIVITE", "Le message associé")