

Environnements Logiciels pour l'Informatique Mobile

Android : API Graphique

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Button
        android:id="@+id/un"
        android:layout_width="match_parent"
        android:layout_height="wrap content"
```

Présentation

- Polytech'Nice-Sophia 2012 (IAM)
- 5 ans chez Sopra-Steria
 - Développement
 - Architecture
 - Projets innovants
 - Formation
- gregory.marro@soprasteria.com



Les widgets/Views

- De nombreux widgets dans le SDK, mais possibilité de les surcharger/créer
- Destinés à afficher/interagir avec l'utilisateur
- La vue est réalisée dans le XML, les interactions dans le Java :
 - `Button myButton = (Button) findViewById(R.id.my_button);`
 - `myButton.setOnClickListener(new View.OnClickListener() {
 @Override
 public boolean onClick(View v) {
 Toast.makeText(context, "click", Toast.LENGTH_SHORT).show();
 return false;
 }
});`

TextView

- Affichage de texte non modifiable par l'utilisateur

- Peut être formaté

- Documentation :

<https://developer.android.com/reference/android/widget/TextView.html>

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/textView"
    android:textSize="8sp" />
```

Button

- Hérite de TextView, supporte donc les mêmes attributs
- Cliquable
- Documentation :
<https://developer.android.com/reference/android/widget/Button.html>

```
<Button  
    android:id="@+id/un"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="Bouton 1" />
```

CheckBox

```
<CheckBox  
    android:text="CheckBox"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/checkBox" />
```

- Hérite de Button, supporte donc les mêmes attributs
- Deux états possibles : cochée ou pas
- Documentation :
<https://developer.android.com/reference/android/widget/CheckBox.html>
- Voir aussi : android.widget.Switch

RadioButton/RadioGroup

- Hérite de Button, supporte donc les mêmes attributs
- Les RadioButton doivent être regroupés dans un RadioGroup (qui est un layout) pour être utilisés correctement
- Documentation : <https://developer.android.com/reference/android/widget/RadioButton.html>

```
<RadioGroup
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="horizontal" >
  <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="radio1"
    android:checked="true" />
  <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="radio2" />
  <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="radio3" />
</RadioGroup>
```

EditText

- Champ texte rempli par l'utilisateur
- Noter la différence entre android:hint (qui ne remplit pas le champ) par rapport à android:text
- AutoCompleteTextView permet de gérer l'autocomplétion
- Documentation :
<https://developer.android.com/reference/android/widget/EditText.html>

```
<EditText  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="@string/editTextHint"  
    android:inputType="textMultiLine"  
    android:lines="2" />
```


ProgressBar

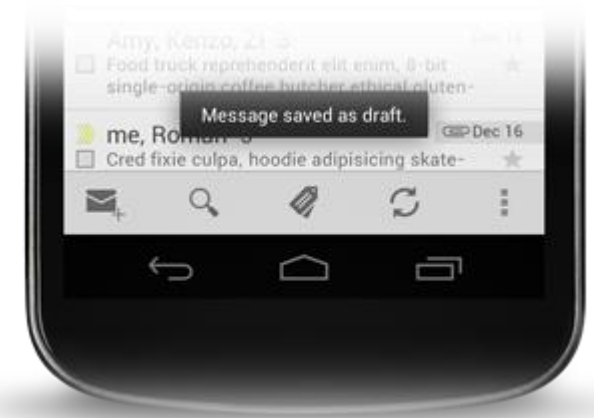
- Déterminée ou indéterminée
- Utile pour les Threads séparés

- Documentation :

<https://developer.android.com/reference/android/widget/ProgressBar.html>

```
<ProgressBar  
    style="?android:attr/progressBarStyleHorizontal"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:id="@+id/progressBar2"  
    android:max="100"  
    android:progress="10"  
    android:visibility="visible" />
```

Toast



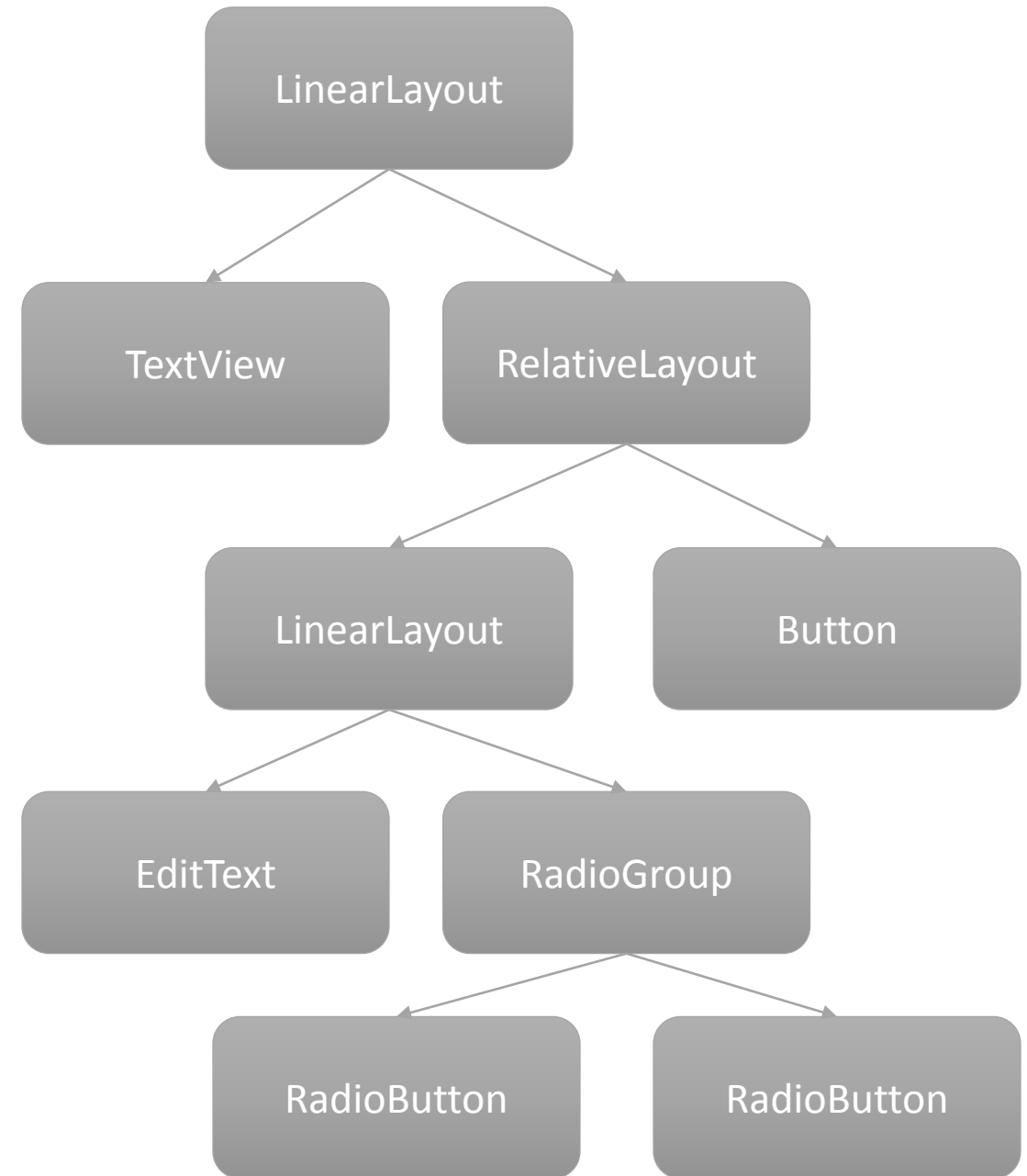
- Permet d'afficher un message pendant un temps prédéfini
- Ne pas oublier l'appel à la méthode show()
- Documentation :
<https://developer.android.com/guide/topics/ui/notifiers/toasts.html>

Le cas des listes

- Fait appel à des Adapter qui vont construire, pour chaque élément de la liste, une vue :
 - ArrayAdapter : pour les listes d'éléments simples
 - SimpleAdapter : pour les objets plus complexes
 - CursorAdapter : pour les requêtes en base de données
- Chacune de ces vues sera ensuite liée par un AdapterView qui permet la création de la liste, la gestion du scroll etc. On peut citer :
 - ListView (liste des contacts)
 - GridView (galerie d'images)
 - Spinner

L'utilisation des Layouts

- Permet de disposer les Layouts/Widgets dans la vue
- Plusieurs manières de réaliser une même vue
- Toujours penser aux différentes tailles d'écran

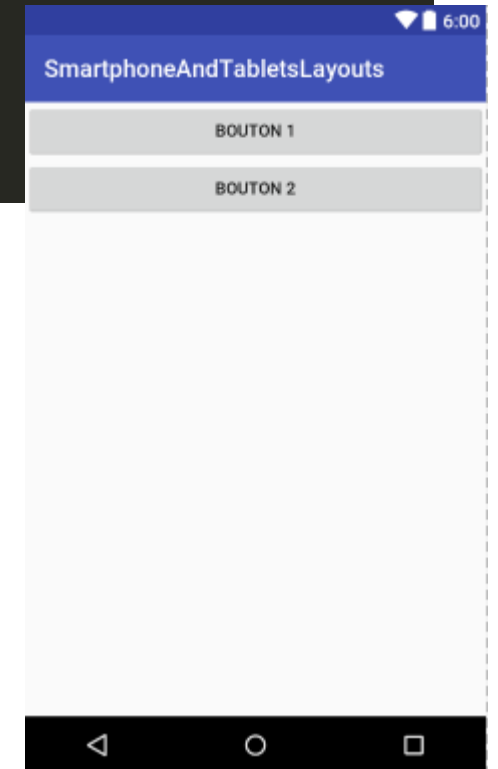


LinearLayout

- Horizontal ou vertical
- Pas de chevauchement possible
- Penser au ScrollView
- Attention, si un élément fait toute la hauteur, il cache les autres

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <Button
        android:id="@+id/un"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Bouton 1" />

    <Button
        android:id="@+id/deux"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Bouton 2" />
</LinearLayout>
```



RelativeLayout

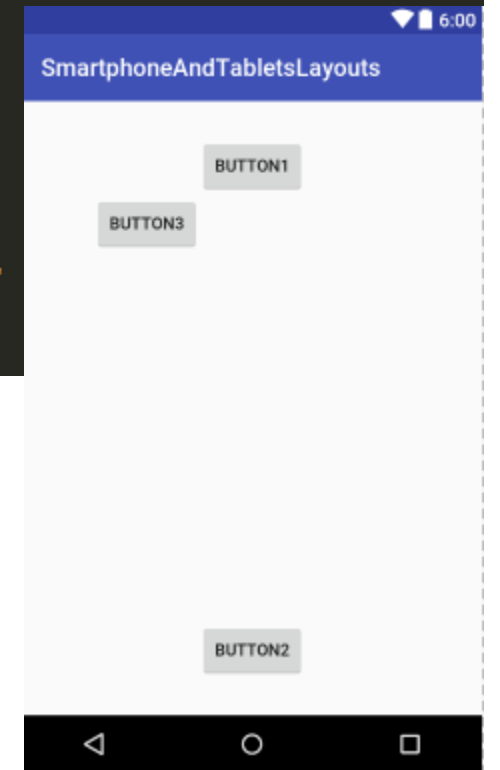
- Relations avec ses parents et pairs
- Nécessite l'utilisation des ids :
 - `android:id= "@+id/idElem"`
- Très puissant et performant
- Peut être sensible aux modifications, plus compliqué à maîtriser

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <Button
        android:text="Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:id="@+id/button1" />

    <Button
        android:text="Button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignEnd="@id/button1"
        android:layout_marginBottom="30dp"
        android:id="@+id/button2" />

    <Button
        android:text="Button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/button1"
        android:layout_toStartOf="@id/button1"
        android:id="@+id/button3" />
</RelativeLayout>
```



TableLayout

- Layout sous un format de tableau
- Proche du « table » HTML
- Fusion de cellules avec layout_span

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TableRow>
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dip">
        <TextView
            android:id="@+id/textView1"
            android:text="Column 1"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:width="200dp"/>
        <Button
            android:id="@+id/button1"
            android:text="Column 2"
            android:width="180dp"/>
    </TableRow>
    <TableRow>
        android:id="@+id/tableRow2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="5dip">
        <EditText
            android:id="@+id/editText1"
            android:layout_span="2"
            android:text="Column 1 and 2 ....."
        </EditText>
    </TableRow>
</TableLayout>
```



FrameLayout

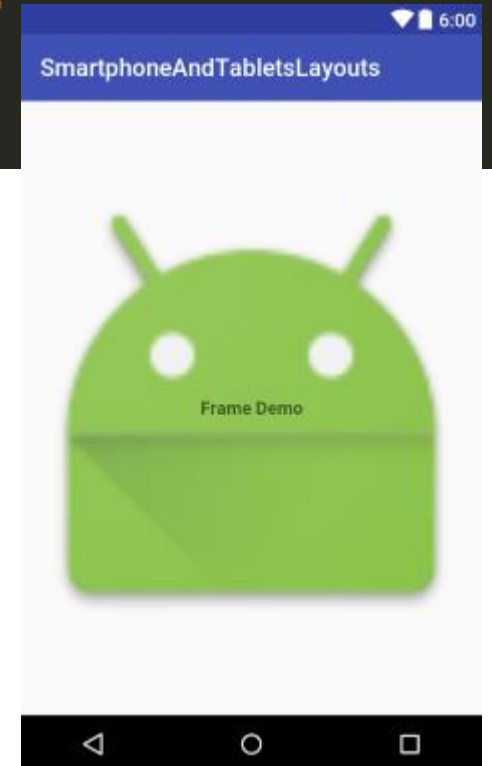
- Permet l'affichage d'une vue (exemple : visualisation des images)
- ou d'une superposition de vues (exemple : Google Maps)

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:src="@mipmap/ic_launcher"
        android:scaleType="fitCenter"
        android:layout_height="match_parent"
        android:layout_width="match_parent"/>

    <TextView
        android:text="Frame Demo"
        android:textSize="30px"
        android:textStyle="bold"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:gravity="center"/>

</FrameLayout>
```



ScrollView

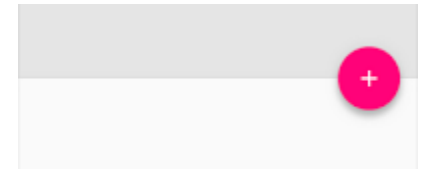
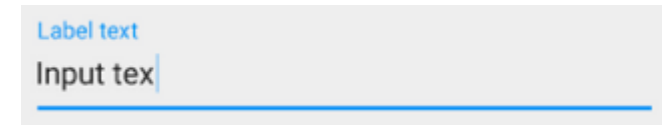
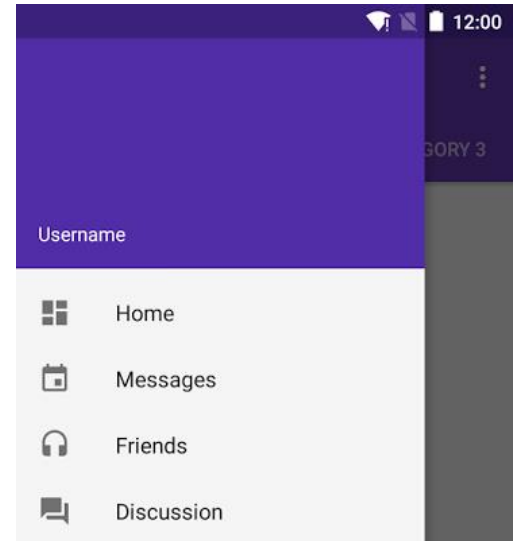
- Malgré son nom, c'est un Layout
- Permet de rendre un widget/layout scrollable
- Attention aux conflits si l'élément enfant gère déjà le scroll
- Il est fortement conseillé de n'avoir qu'un enfant pour l'utilisation de ce layout

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="wrap_content">
  <LinearLayout>
    <!-- contenu du layout -->
  </LinearLayout>
</ScrollView>
```

Smartphones & tablettes

Android

- Attention à la diversité des périphériques
- Composants `android.support.design.widget` :
 - `CoordinatorLayout`
 - `NavigationView`
 - `TextInputLayout`
 - `FloatingActionButton`
 - `SnackBar`
 - ...



TV

Android TV

- Pas de tactile sur les TV !
 - Dans le manifest

```
<manifest>
  <uses-feature android:name="android.hardware.touchscreen"
              android:required="false" />
  ...
</manifest>
```

- Utilisation du thème LeanBack

```
<activity
  android:name="com.example.android.TvActivity"
  android:label="@string/app_name"
  android:theme="@style/Theme.Leanback">
```

- Pas d'utilisation d'ActionBar et ViewPager !
- Utilisation des GridView plutôt que les ListView

Wear

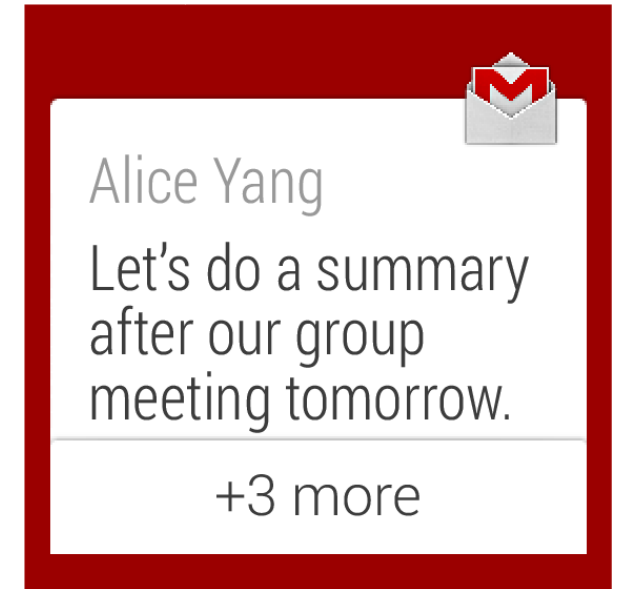
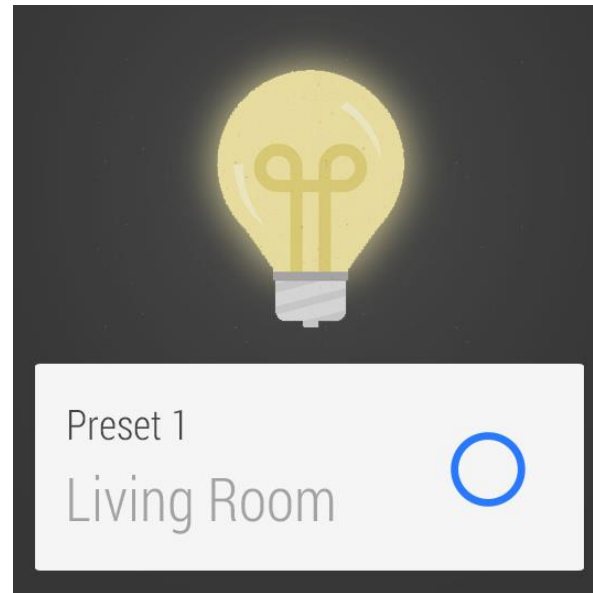
Android Wear

- Les wearables sont constamment connectés à un smartphone ou une tablette
- Il est nécessaire de créer plusieurs fichiers pour la même vue grâce au `WatchViewStub` :
 - Afficheur carré
 - Afficheur rond

Sauf si utilisation du `BoxInsetLayout` qui gère les deux affichages

Android Wear

- Utilisation des Cards



TD : Utilisation des listes

1. Création d'une liste à partir d'une liste d'éléments simples
2. Mettre en place une liste à 2 niveaux
3. Afficher une vue « complexe » (au moins 2 éléments par item)
4. Affichage en mode Grille

TD : Utilisation des listes

1. Création d'une liste à partir d'une liste d'éléments simples

➔ Utiliser un ArrayAdapter

```
// Définition de l'adapter  
// Premier Paramètre - Context  
// Second Paramètre - le Layout pour les Items de la Liste  
// Troisième Paramètre - l'ID du TextView du Layout des Items  
// Quatrième Paramètre - le Tableau de Données  
  
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
    android.R.layout.simple_list_item_1, android.R.id.text1, villes);  
  
// on assigne l'adapter à notre list  
listView.setAdapter(adapter);
```

TD : Utilisation des listes

2. Mettre en place une liste à 2 niveaux

→ Utiliser un ExpandableListView

Créer un xml pour l'affichage d'un groupe

Créer un xml pour l'affichage d'un enfant

TD : Utilisation des listes

3. Afficher une vue « complexe » (au moins 2 éléments par item)

➔ Utiliser un SimpleAdapter

Créer un xml pour l'affichage d'un élément

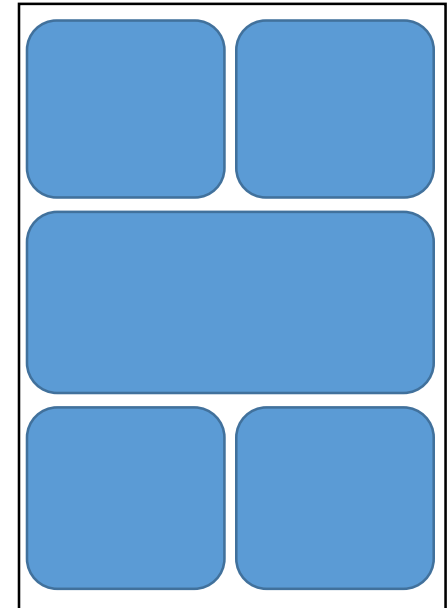
4. Affichage en mode Grille

➔ Utiliser un GridView

Créer un xml pour l'affichage d'un élément

TD : Utilisation des layouts

1. Réaliser un écran présentant 5 boutons dans le format suivant à l'aide :
 1. Uniquement des LinearLayout
 2. Uniquement des RelativeLayout
 3. Uniquement des TableLayout
2. Ajouter un ScrollView
3. Utilisation d'un CoordinatorLayout



TD : Utilisation des layouts

1. Réaliser un écran présentant 5 boutons dans le format suivant à l'aide :

1. Uniquement des LinearLayout



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight=".33">

    <Button
        android:id="@+id/un"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight=".5"
        android:text="Bouton 1" />

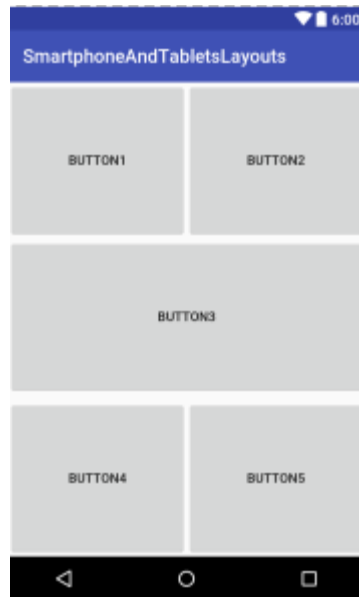
    <Button
        android:id="@+id/deux"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight=".5"
        android:text="Bouton 2" />

</LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
```

TD : Utilisation des layouts

1. Réaliser un écran présentant 5 boutons dans le format suivant à l'aide :
2. Uniquement des RelativeLayout



```
<android.support.percent.PercentRelativeLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <Button
        android:text="Button1"
        app:layout_heightPercent="33%"
        app:layout_widthPercent="50%"
        android:layout_alignParentTop="true"
        android:id="@+id/button1" />

    <Button
        android:text="Button2"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@id/button1"
        app:layout_heightPercent="33%"
        app:layout_widthPercent="50%"
        android:id="@+id/button2" />

    <Button
        android:text="Button3"
        app:layout_heightPercent="33%"
        app:layout_widthPercent="100%"
        android:layout_below="@id/button2"
        android:id="@+id/button3" />

    <Button
        android:text="Button4"
        app:layout_heightPercent="33%"
```

TD : Utilisation des layouts

1. Réaliser un écran présentant 5 boutons dans le format suivant à l'aide :
3. Uniquement des TableLayout



```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

```
<TableRow
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight=".33">

    <Button
        android:id="@+id/un"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight=".5"
        android:text="Bouton 1" />


    <Button
        android:id="@+id/deux"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight=".5"
        android:text="Bouton 2" />

</TableRow>
```

```
<TableRow
    android:layout_width="match_parent"
    android:layout_height="0dp"
    . . . . .
```


TD : Utilisation des layouts

2. Ajouter un ScrollView

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
  android:layout_height="match_parent">
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight=".33">
```

TD : Application de notes de frais

1. Lister les notes de frais dans une liste (avec la date et le montant)
2. Ajouter une note de frais en cliquant sur un FloatingActionButton, qui ouvre une activité permettant la création avec :
 1. Une catégorie de frais (restaurant, essence, hôtel, parking...)
 2. Un montant
 3. Une date
 4. Une photo de justificatif
3. Bonus : Supprimer une note de frais avec un swipe de droite à gauche