
HE IG VD
IICT
Institut des
Technologies de
l'information et de
la communication

HEIG VD
IICT
Institut des
Technologies de
l'information et de
la communication

Développement d'applications *Android*

Corrigé – Laboratoire n°1

Prise en main d'Android Studio, les ressources et les layouts

Elliot Ganty

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

Manipulation 4 : Le *LinearLayout* (portrait)

Les ressources à créer

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="mycolor1">#ABC</color>
   <color name="mycolor2">#CBA</color>
</resources>
```

res/values/dimensions.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <dimen name="finger_size">48dp</dimen>
</resources>
```

Manipulation 4 : Le *LinearLayout* (portrait)

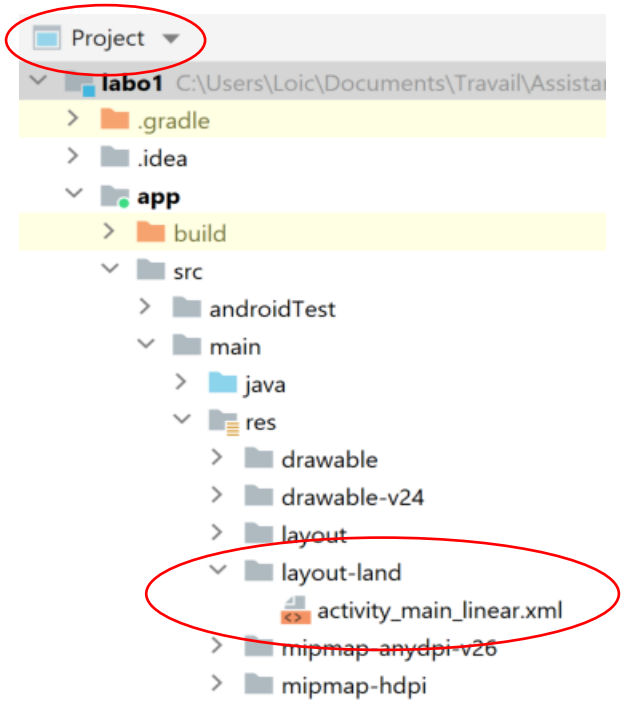
res/layout/activity_main_linear.xml

```
<?xml version="1.0" encoding="utf-8"?>

<ScrollView
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"
   tools:context=".MainActivity">

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

      <View
         android:layout_width="match_parent"
         android:layout_height="@dimen/finger_size"
         android:background="@color/mycolor1" />
      <View
         android:layout_width="match_parent"
         android:layout_height="@dimen/finger_size"
         android:background="@color/mycolor2" />

   </LinearLayout>

</ScrollView>
```

- Le ***LinearLayout*** a une orientation verticale. Il s'adapte en hauteur par rapport au contenu et aura la même largeur que son parent
- Les *View* auront comme **hauteur** celle définie par la variable «finger_size» du fichier de ressource «dimensions.xml». Pareil pour la couleur avec les variables «mycolor1-2» définie dans «colors.xml»
- **Effet de bord**: si les vues à afficher représentent une surface supérieure à celle disponible à l'écran: impossible de scroller pour toutes les voir !
  - Une ***ScrollView*** doit être utilisée afin de pouvoir naviguer dans tous les éléments de la liste
- **Problème en mode paysage** : Les lignes restent horizontales
  - Il va falloir créer une **variation de layout** en mode paysage afin d'afficher les éléments de manière horizontale

Manipulation 4 : Le *LinearLayout* (paysage)

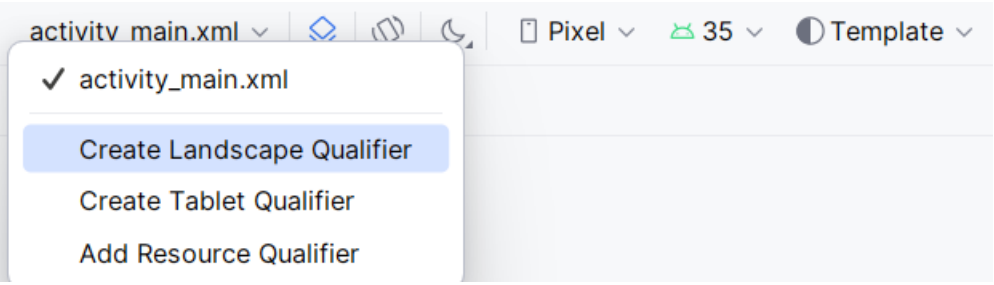
Création de la variante paysage (méthode 1)



- En activant la vue «**Project**» à la place de «Android», il est possible de voir l'arborescence complète des dossiers
- Il suffit de créer un dossier «**layout-land**» dans le répertoire «res» et d'y placer un fichier *layout* XML du même nom que la version portrait
- Ce *layout* va automatiquement être sélectionné lorsque le téléphone basculera en mode paysage

Manipulation 4 : Le *LinearLayout* (paysage)

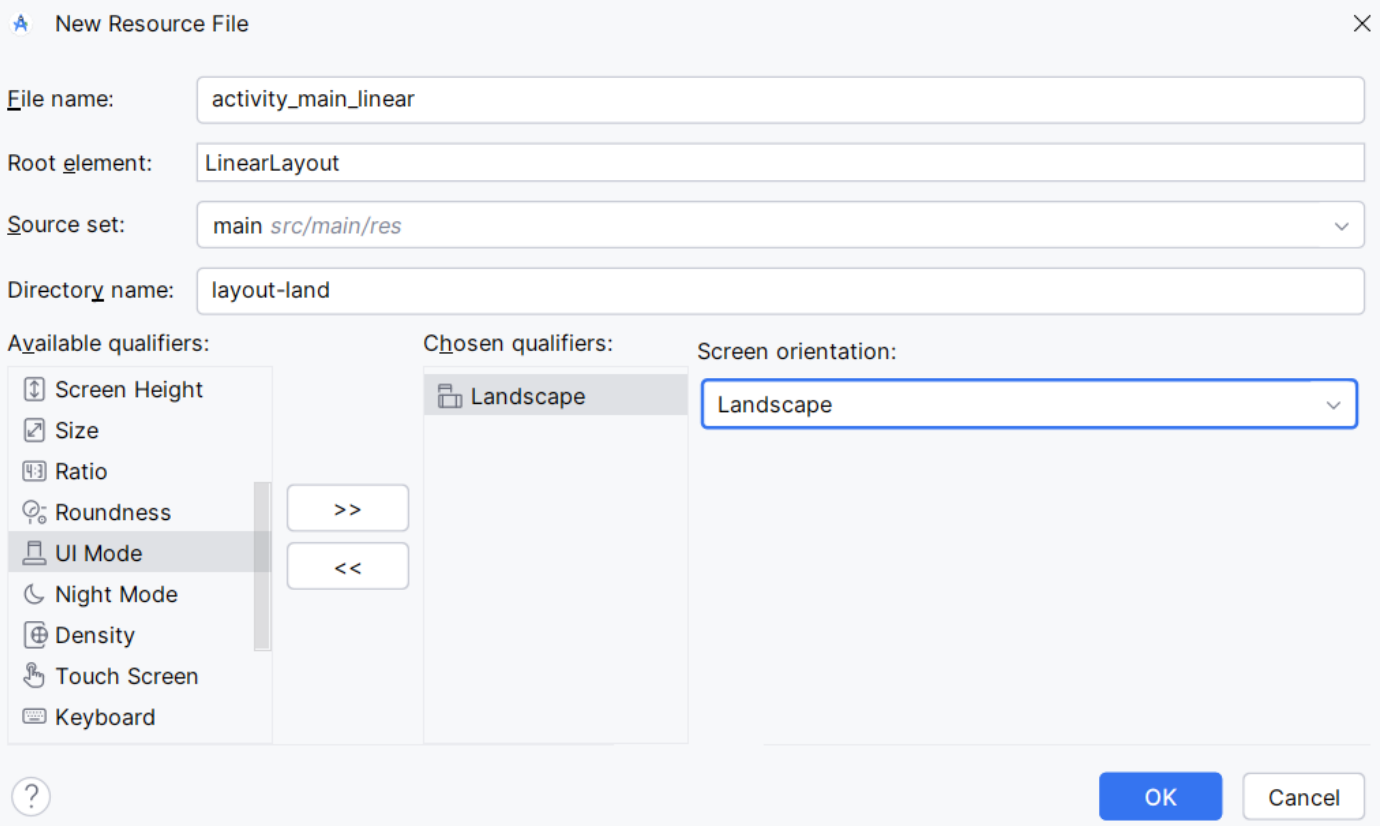
Création de la variante paysage (méthode 2)



- Ouvrir le fichier XML du *layout* en mode portrait
- Sélectionner le mode de vue «Split» ou «Design»
- Il est ensuite possible de créer une variation de *layout* en mode paysage

Manipulation 4 : Le *LinearLayout* (paysage)

Création de la variante paysage (méthode 3)



- En retournant dans la vue «**Android**» à la place de «Project», il suffit de sélectionner via un clic droit sur le layout ciblé : New / Layout Resource File
- Puis de choisir l'option «**Orientation**» dans la liste
- Finalement lui donner le même nom que le layout ciblé
- On crée ainsi une variation automatiquement sélectionnée lorsque l'orientation du téléphone change.

- Il est possible de définir d'autres variations selon d'autres critères en choisissant un élément différent comme le mode sombre (night mode).

Manipulation 4 : Le *LinearLayout* (paysage)

res/layout-land/activity_main_linear.xml

```
<?xml version="1.0" encoding="utf-8"?>

<HorizontalScrollView
   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"
   tools:context=".MainActivity">

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

      <View
         android:layout_width="@dimen/finger_size"
         android:layout_height="match_parent"
         android:background="@color/mycolor1" />
      <View
         android:layout_width="@dimen/finger_size"
         android:layout_height="match_parent"
         android:background="@color/mycolor2" />

</LinearLayout>

</HorizontalScrollView>
```

- Il faut utiliser une *HorizontalScrollView* afin de pouvoir naviguer dans tous les éléments de la liste dans le sens horizontal.
- Cette fois le *LinearLayout* prendra la même hauteur que le parent et sa largeur sera adaptée au contenu
- Chaque *View* aura comme **largeur** celle définie par la clé «finger_size» du fichier de ressource «dimensions.xml»

Manipulation 5 : Le *RelativeLayout*

Les ressources à créer

res/values/strings.xml

```
<resources>
   <string name="main_login_title">Log-in</string>
   <string name="main_login_username">Username</string>
   <string name="main_login_password">Password</string>
   <string name="main_login_connect">Connect</string>
</resources>
```

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name=“purple">#CE93D8</color>
   <color name=“teal">#80CBC4</color>
</resources>
```

res/values/dimensions.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <dimen name="box_width">350dp</dimen>
   <dimen name="box_title_width">100dp</dimen>
</resources>
```

Manipulation 5 : Le *RelativeLayout*

res/layout/activity_main_relative.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
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"
   tools:context=".MainActivity">
   <RelativeLayout
      android:layout_width="@dimen/box_width"
      android:layout_height="wrap_content"
      android:layout_centerInParent="true">

   </RelativeLayout>
</RelativeLayout>
```

Widgets relatifs au formulaire

- Le second *RelativeLayout* sera un bloc centré horizontalement et verticalement par rapport à son parent

- A l'intérieur de celui-ci, tous les éléments devront être placés en utilisant les paramètres relatifs au parent ou aux éléments adjacents:
  - **(layout_alignParentStart)**
  - **Layout_below**
  - **…**

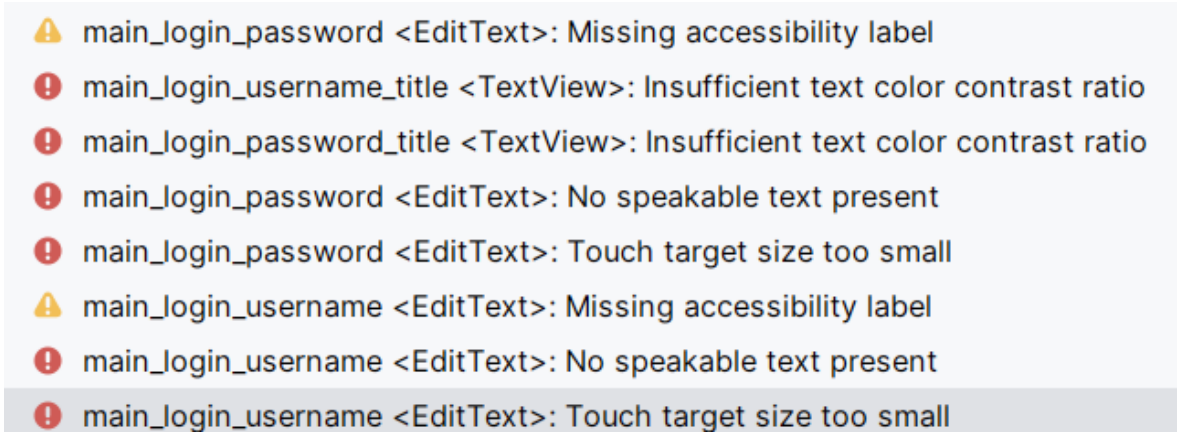
Manipulation 5 : Le *RelativeLayout*

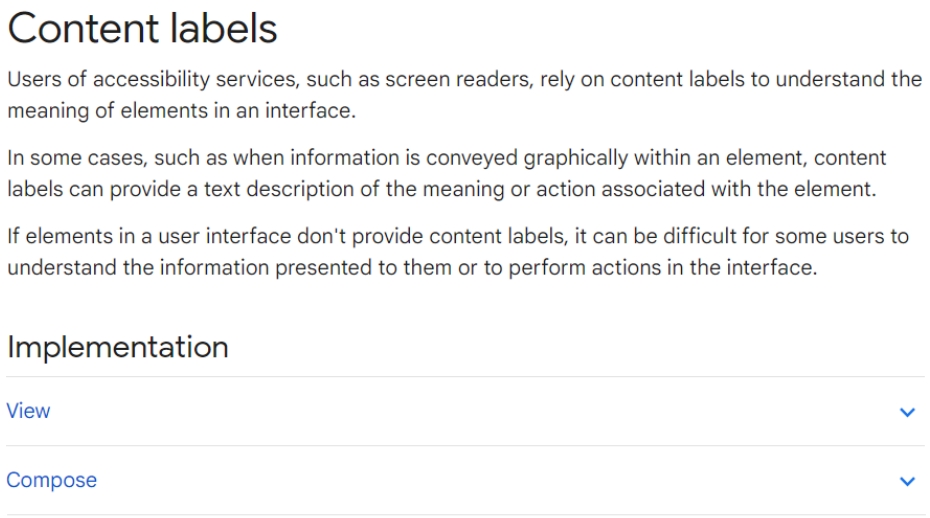
```
<TextView
   android:id="@+id/main_login_title"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_alignParentStart="true"
   android:layout_alignParentTop="true"
   android:layout_alignParentEnd="true"
   android:background="@color/teal_200"
   android:text="@string/main_login_title"
   android:textAppearance="@style/TextAppearance.AppCompat.Medium" />

<TextView
   android:id="@+id/main_login_username_title"
   android:layout_width="@dimen/box_title_width"
   android:layout_height="wrap_content"
   android:layout_alignParentStart="true"
   android:layout_alignBaseline="@id/main_login_username"
   android:background="@color/purple_200"
   android:text="@string/main_login_username" />

<EditText
   android:id="@+id/main_login_username"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_below="@+id/main_login_title"
   android:layout_alignParentEnd="true"
   android:layout_toEndOf="@id/main_login_username_title" />
```

```
<Button
   android:id="@+id/main_login_button"
   android:layout_width="150dp"
   android:layout_height="wrap_content"
   android:layout_below="@+id/main_login_username_title"
   android:layout_centerInParent="true"
   android:layout_alignParentEnd="true"
   android:text="@string/main_login_connect"/>
```

- L'*EditText* prend plus de place que le *TextView*, il est donc préférable d'utiliser le paramètre ***layout_alignBaseline*** sur le *TextView*. Cela aura pour effet d'aligner les deux lignes de texte
- Le second *RelativeLayout* sera un bloc centré horizontalement et verticalement par rapport à son parent
- A l'intérieur de celui-ci, tous les éléments devront être placés en utilisant les paramètres relatifs au parent **(layout_alignParentXX)**

Manipulation 5 : Le *RelativeLayout*





- Ces erreurs / warnings au niveau du layout concernent des bonnes pratiques et également comment améliorer «l'accessibilité» de l'interface, donc le fait qu'elle soit utilisable par le plus grand nombre de personnes, y compris celles présentant certaines difficultés de lecture comme les personnes malvoyantes.

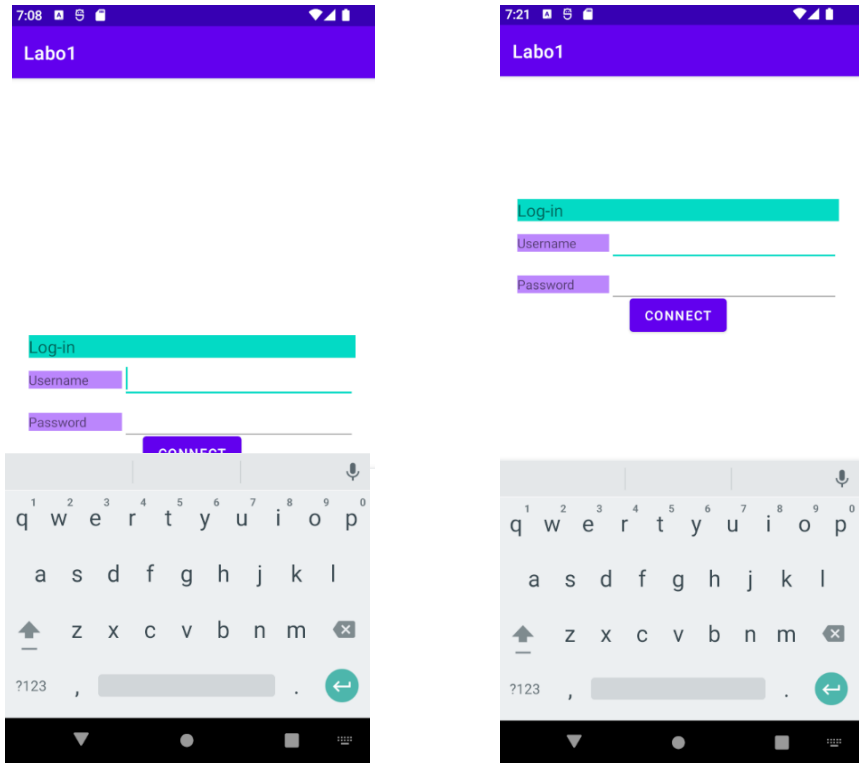
  il est important d'y être attentif !

- Ne pas hésiter à visiter les liens mentionnés pour comprendre comment les corriger

- Exemple ici : Les champs du formulaire devraient avoir l'attribut «contentDescription» indiquant un descriptif lu par les lecteurs d'écran

Manipulation 5 : Le *RelativeLayout*

Positionnement du clavier virtuel

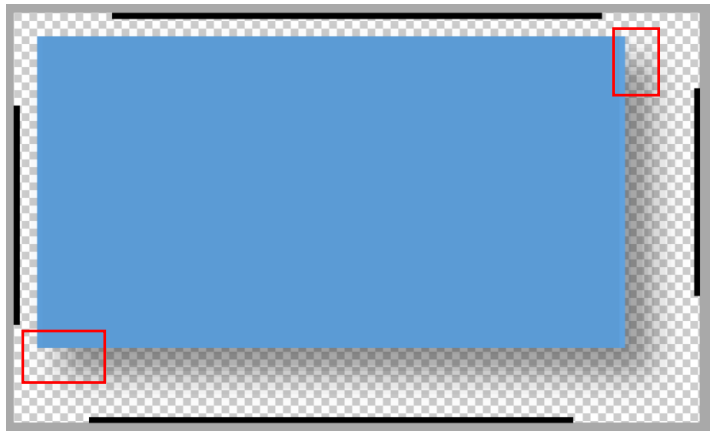


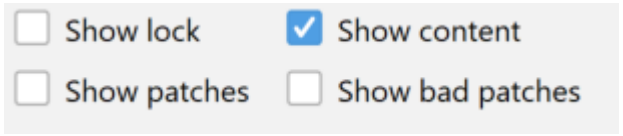
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/andr
oid"
   package="ch.heigvd.iict.and.labo1">
   <application … >
      <activity
         android:name=".MainActivity"
         android:windowSoftInputMode="adjustResize“
        … >
      </activity>
   </application>
</manifest>
```

- Par défaut, il est possible que le clavier virtuel passe par-dessus les éléments graphiques lors de la saisie, ce qui peut être gênant dans certains cas
- La propriété *windowSoftInputMode* peut être ajoutée dans le fichier *Manifest.* Elle permet de gérer le comportement à adopter par l'activité quand le clavier virtuel apparaît
- Avec la valeur *adjustResize,* la fenêtre de l'activité va toujours essayer de se redimensionner pour laisser de la place au clavier virtuel

Manipulation 6 : Le *Nine-Patch*





- En ajoutant (via drag & drop par exemple) un fichier comportant l'extension «.9.png» dans le dossier «drawable» et en le sélectionnant, Android Studio nous ouvre un outil permettant de définir les zones de redimensionnement et de contenu
- Attention à ne pas prendre **les zones sans ombre** lors de la définition de la zone de redimensionnement
- La case à cocher «Show content» permet d'activer une prévisualisation de la zone de contenu dans des conteneurs de plusieurs tailles. Cela permet de rapidement voir si les zones définies et le redimensionnement sont corrects
- L'ajout du *Nine-Patch* en fond d'un élément se fait avec la propriété *background* et un lien vers le fichier

```
<RelativeLayout
   android:layout_width="@dimen/box_width"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true"
   android:background="@drawable/bg">
```

Manipulation 7 : Evénement utilisateur

Définir la méthode callback lors de l'appui sur le bouton

```
<Button
  android:id="@+id/main_login_button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_centerInParent="true"
  android:layout_below="@id/main_login_password_title"
  android:text="@string/main_login_connect"/>
```

1: Définir un identifiant pour le bouton dans le layout

Dans la méthode *onCreate* de l'activité :

```
val monButton = findViewById<Button>(R.id.main_login_button)

monButton.setOnClickListener {
   Toast.makeText(this, "Click", Toast.LENGTH_SHORT).show();
}
```

2: Récupérer l'instance du bouton

3: Enregistrer la méthode anonyme qui sera appelée lorsque l'événement se produit.
Ici on affiche un toast avec le message «Click»

HE VD
IG

HAUTE ÉCOLE
D'INGÉNIERIE
ET DE GESTION
DU CANTON
DE VAUD