# Annexe 8 – StateListDrawable, Selector, Shape

Bien que les composants Material réagissent à des actions avec des effets visuels intégrés, on peut vouloir assigner nos propres effets visuels à des composants.

## Avantages des StateListDrawables ( Selectors ) :

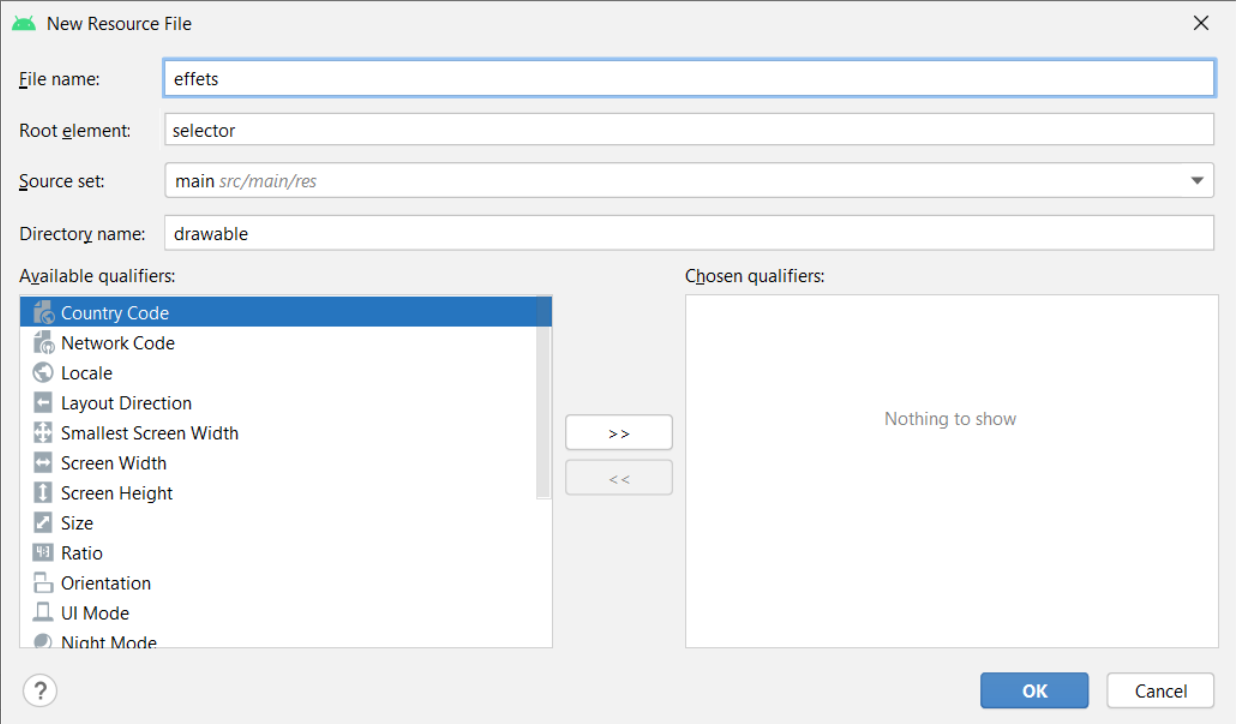
* Personnalisation de l'apparence d'un composant en fonction de son état (pressé, activé, désactivé, etc.) sans écrire de code Java.
* Centralisation du style dans des fichiers XML, facilitant la réutilisabilité et la maintenance des ressources graphiques.

Exercice :

Ajoutez un bouton sur le visuel d’une activité et modifiez le code dans le fichier de positionnement xml de Button. Vu qu’il subit le thème Material, on va lui retirer le contenu de la propriété backagroundTint de base en faisant backgroundTint= « @null »

Assignons un StateListDrawable à ce bouton puisqu'il a perdu ce qui venait du thème Material.

En vous plaçant sur res / drawable , créer un nouveau Drawable Resource File :



Assurez-vous que selector est l'élément-racine, donnez un nom à votre fichier ( effets )

Un fichier effets.xml est produit. Ajoutez le système de balises suivant comme point de départ à l'intérieur de la balise <selector>:

<**item android:state\_pressed="true"** >  
 <**shape android:shape="rectangle"** >  
 <**solid android:color="#ff0000"**/>  
 <**corners android:radius="12dip"** />  
 <**stroke android:width="12px" android:color="@color/teal\_200"** />  
 </**shape**>  
</**item**>  
  
<**item** >  
 <**shape android:shape="rectangle"** >  
 <**corners android:radius="12dp"** />  
 <**stroke android:width="12dp" android:color="#ff0000"** />  
 <**solid android:color="#ff0000"**/>  
 </**shape**>  
</**item**>

De cette façon, le bouton aura une bordure de couleur différente si on appuie dessus.

Pour appliquer le StateListDrawable au bouton, on change le background du bouton pour

Cliquez ou appuyez ici pour entrer du texte.

## Les états possibles ( dépendant de la composante ) :

    android:state\_pressed=["true" | "false"]  
        android:state\_focused=["true" | "false"]  
        android:state\_hovered=["true" | "false"]  
        android:state\_selected=["true" | "false"]  
        android:state\_checkable=["true" | "false"]  
        android:state\_checked=["true" | "false"]  
        android:state\_enabled=["true" | "false"]  
        android:state\_activated=["true" | "false"]  
        android:state\_window\_focused=["true" | "false"]

\*\*\* important de terminer par la version au repos ( sans événement )

## Les types de "shape" possibles :

|  |  |
| --- | --- |
| "rectangle" | Rectangle qui remplit l'affichage parent. Il s'agit de la forme par défaut. |
| "oval" | Forme ovale qui correspond aux dimensions de l'affichage parent. |
| "line" | Ligne horizontale qui occupe toute la largeur de l'affichage parent.  Cette forme nécessite que l'élément <stroke> définisse la largeur de la ligne. |
| "ring" | Forme circulaire. Demande d'autres attributs tels android:innerRadius ou  android:thickness |

## Les propriétés des formes :

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

<shape

xmlns:android="http://schemas.android.com/apk/res/android"

android:shape=["rectangle" | "oval" | "line" | "ring"] >

<corners

android:radius="integer"

android:topLeftRadius="integer"

android:topRightRadius="integer"

android:bottomLeftRadius="integer"

android:bottomRightRadius="integer" />

<gradient

android:angle="integer"

android:centerX="float"

android:centerY="float"

android:centerColor="integer"

android:endColor="color"

android:gradientRadius="integer"

android:startColor="color"

android:type=["linear" | "radial" | "sweep"]

android:useLevel=["true" | "false"] />

<padding

android:left="integer"

android:top="integer"

android:right="integer"

android:bottom="integer" />

<size

android:width="integer"

android:height="integer" />

<solid

android:color="color" />

<stroke

android:width="integer"

android:color="color"

android:dashWidth="integer"

android:dashGap="integer" />

</shape>

<https://developer.android.com/guide/topics/resources/drawable-resource?hl=fr#Shape>

## Exercice 2 – lien avec le tp

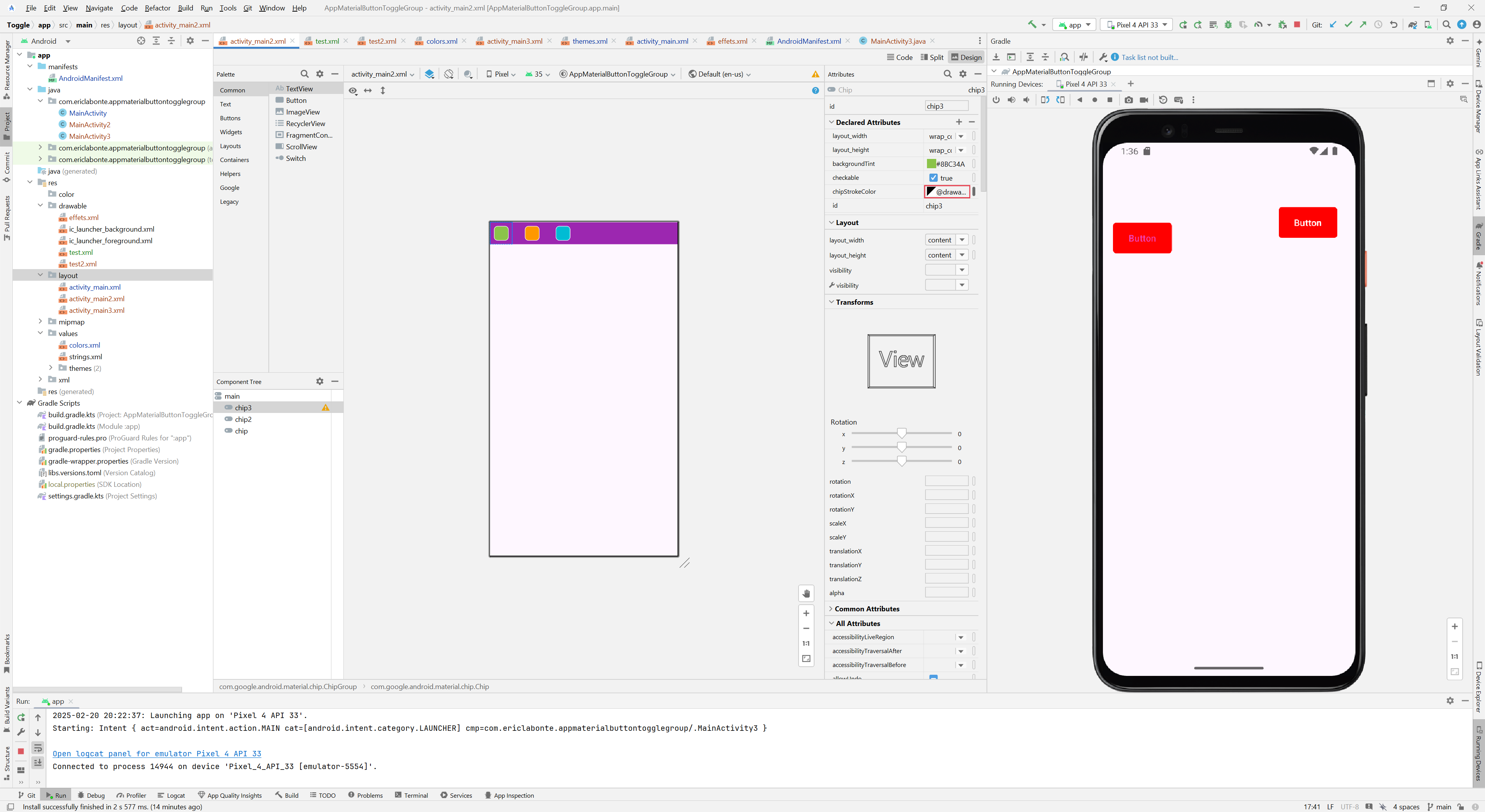
Question : Je veux indiquer par un contour différent la couleur sélectionnée

Plusieurs options s’offrent à nous en minimisant le rajout de code dans notre fichier java

Je vous propose l’utilisation du composant Chip à la place de Button :

* Un Chip est un composant pouvant être sélectionné ( checked ) ou non
* Un Chip peut être à l’intérieur d’un ChipGroup, maintenant ainsi le fait d’avoir un seul item sélectionné à la fois

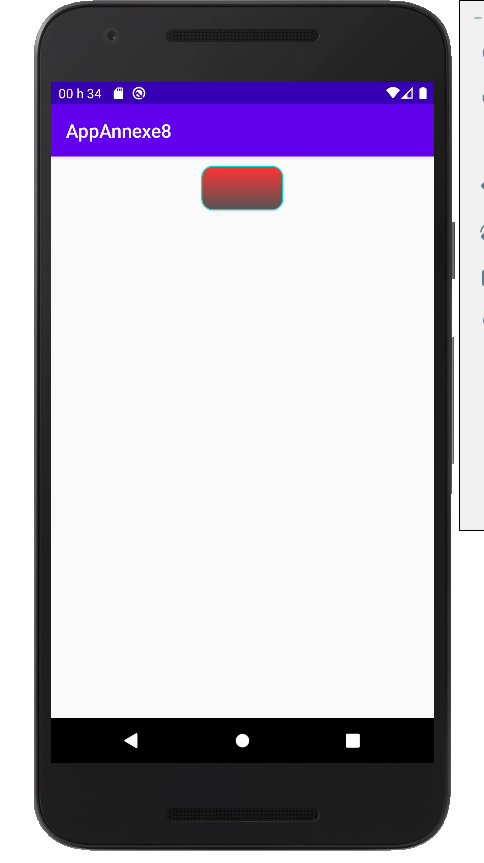
1. Dans une nouvelle activité, modifiez le layout de base par un com.google.android.material.chip.ChipGroup
2. Glissez-y trois objets Chip avec une couleur background\_tint différente ( on est ds le thème Material )

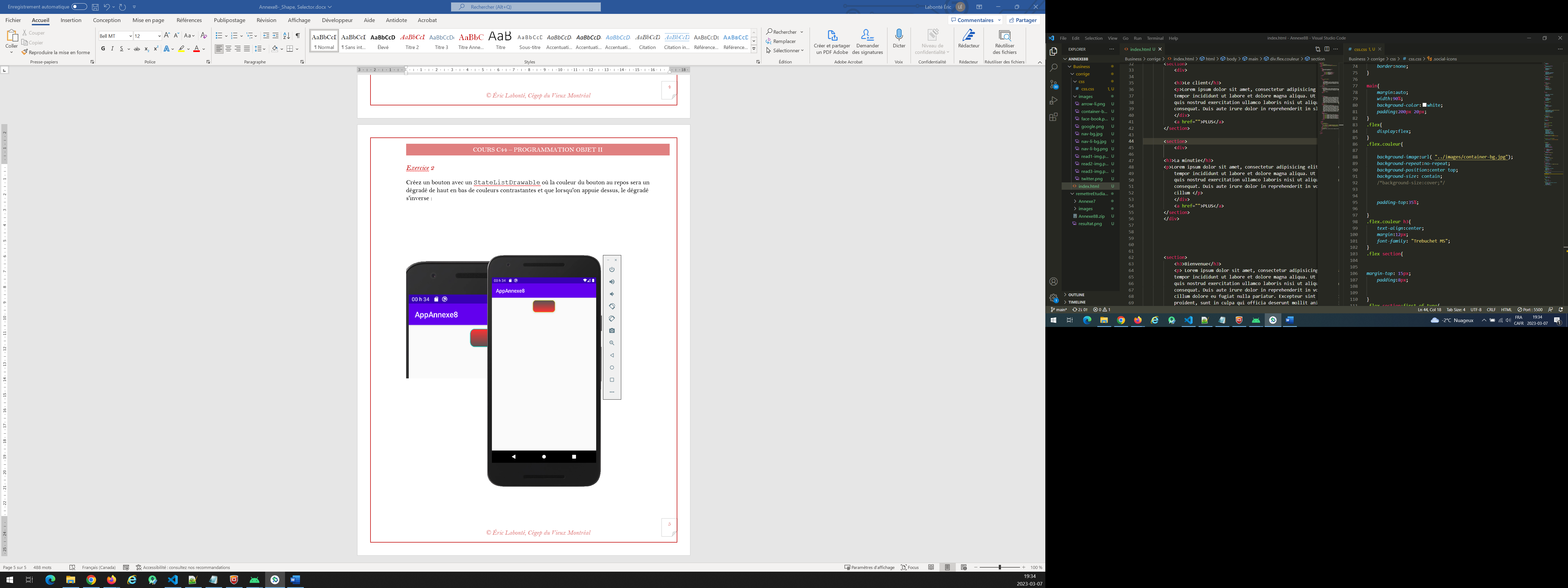


1. On doit ajouter les propriétés suivantes sur le groupe :
   1. app:selectionRequired="true"
   2. app:singleSelection="true"
2. On doit ajouter la propriété suivante à chacun des Chips
   1. android:checkable="true"
3. On va définir un StateListDrawable pour faire en sorte que le couleur soit modifiée si l’élément est « checked »
4. On applique le StateListDrawable à la propriété app :chipStrokeColor
5. C’est pour le graphisme de l’affaire…pour détecter quel chip est sélectionné ( « checked ») , on peut les ajouter à un écouteur qui met en œuvre l’interface CompoundButton.OnCheckedChangeListener

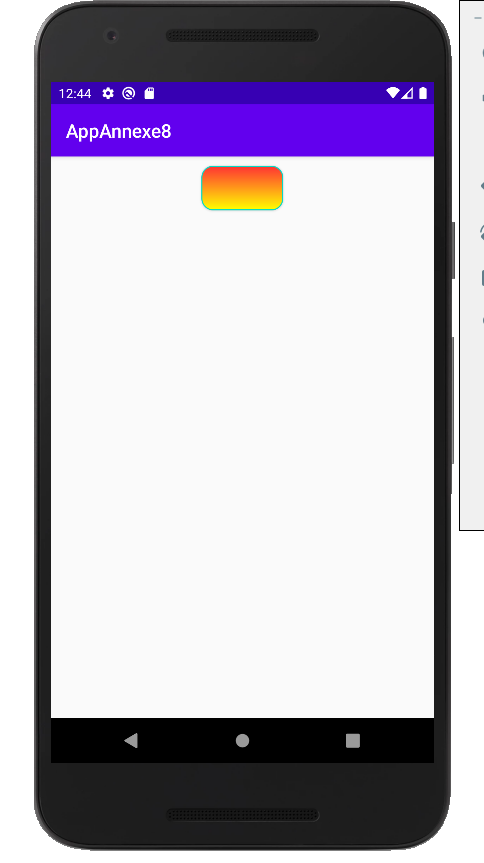
## Exercice3

Créez un bouton avec un StateListDrawable où la couleur du bouton au repos sera un dégradé de haut en bas de couleurs contrastantes et que lorsqu'on appuie dessus, le dégradé s'inverse :





Faire un autre fichier selector utilisant la locale "espagnol" pour que le bouton ait les couleurs du drapeau de l'Espagne quand la langue de l'émulateur est l'espagnol :



*Le téléphone utilise le fichier correspondant à la langue du téléphone*

**Exercice 1 : Bouton avec StateListDrawable**

1. Ajoutez un <Button> à votre layout :

<Button

android:id="@+id/btn\_effets"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Mon Bouton"

android:backgroundTint="@null" <!-- on retire le tint Material -->

/>

1. Créez un nouveau fichier res/drawable/effets.xml (root <selector>).
2. Complétez effets.xml :

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<!-- État pressé -->

<item android:state\_pressed="true">

<shape android:shape="rectangle">

<solid android:color="#FF0000"/>

<corners android:radius="12dp"/>

<stroke

android:width="2dp"

android:color="@color/teal\_200"/>

</shape>

</item>

<!-- État par défaut -->

<item>

<shape android:shape="rectangle">

<solid android:color="#FFFFFF"/>

<corners android:radius="12dp"/>

<stroke

android:width="2dp"

android:color="#FF0000"/>

</shape>

</item>

</selector>

1. Appliquez-le au bouton :

<Button

...

android:background="@drawable/effets"

/>

**Exercice 2 : ChipGroup et contours dynamiques**

**Objectif**

Indiquer visuellement quel Chip est sélectionné en modifiant la couleur du contour (stroke).

1. Dans res/color/, créez chip\_stroke\_selector.xml :

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<!-- État checked -->

<item android:state\_checked="true" android:color="@color/blue"/>

<!-- État par défaut -->

<item android:color="@color/gray"/>

</selector>

1. Dans votre layout XML :

<com.google.android.material.chip.ChipGroup

android:id="@+id/chip\_group"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:singleSelection="true"

app:selectionRequired="true">

<com.google.android.material.chip.Chip

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checkable="true"

app:chipBackgroundTint="@color/primary"

app:chipStrokeColor="@color/chip\_stroke\_selector"

app:chipStrokeWidth="2dp"

android:text="Option 1"/>

<com.google.android.material.chip.Chip

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checkable="true"

app:chipBackgroundTint="@color/primary"

app:chipStrokeColor="@color/chip\_stroke\_selector"

app:chipStrokeWidth="2dp"

android:text="Option 2"/>

</com.google.android.material.chip.ChipGroup>

Vous pouvez récupérer le Chip sélectionné en Java/Kotlin via setOnCheckedChangeListener sur le ChipGroup.

**Exercice 3 : Dégradés inversés et localisation**

**3.1 Bouton à dégradé inversé**

Créez res/drawable/button\_gradient\_selector.xml :

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<!-- Pressé : dégradé inversé -->

<item android:state\_pressed="true">

<shape android:shape="rectangle">

<gradient

android:angle="270"

android:startColor="#FF5722"

android:endColor="#FFC107"/>

<corners android:radius="8dp"/>

</shape>

</item>

<!-- Repos : dégradé normal -->

<item>

<shape android:shape="rectangle">

<gradient

android:angle="90"

android:startColor="#FF5722"

android:endColor="#FFC107"/>

<corners android:radius="8dp"/>

</shape>

</item>

</selector>

Appliquez-le ainsi :

<Button

...

android:background="@drawable/button\_gradient\_selector"

/>

**3.2 Version espagnole (locale es)**

Créez le même fichier sous res/drawable-es/button\_gradient\_selector.xml :

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

<selector xmlns:android="http://schemas.android.com/apk/res/android">

<item android:state\_pressed="true">

<shape android:shape="rectangle">

<gradient

android:angle="270"

android:startColor="#AA151B" <!-- rouge Espagne -->

android:centerColor="#F1BF00" <!-- jaune Espagne -->

android:endColor="#AA151B"/>

<corners android:radius="8dp"/>

</shape>

</item>

<item>

<shape android:shape="rectangle">

<gradient

android:angle="90"

android:startColor="#AA151B"

android:centerColor="#F1BF00"

android:endColor="#AA151B"/>

<corners android:radius="8dp"/>

</shape>

</item>

</selector>

Android choisira automatiquement cette version lorsque la langue de l'appareil est configurée en espagnol.