Laboratoire 4

Menus et barre d'outils

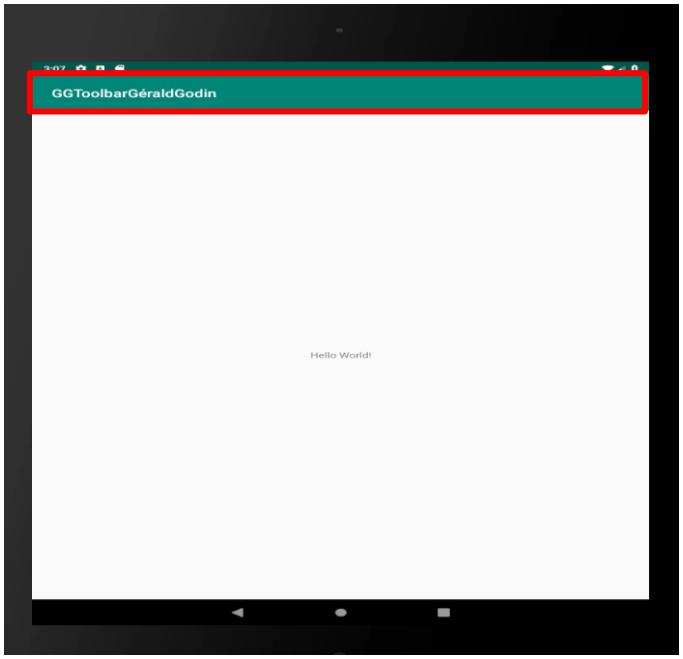
Objectifs d'apprentissage

- Créer et utiliser un menu d'activité
- Créer et utiliser un menu contextuel
- Ajouter une option de menu à la barre d'outils

1. Menu d'activité

Le menu d'activité propose un choix dépendant de l'activité affichée. Il peut contenir des icônes mais pas de cases à cocher ni de boutons radios.

- Créer un nouveau projet avec comme nom d'application **GGToolbarNomPrenom.**
- Comme minimum SDK, choisissez l'API 21 : Android 5.0 (Lollipop)
- Choisissez l'option Empty Activity
- Lancez votre application. Vous devez voir un écran comme ci-dessous ***avec une barre de titre*** portant par défaut le nom de votre application qui est dans ce cas "GGToolbarNomPrenom".



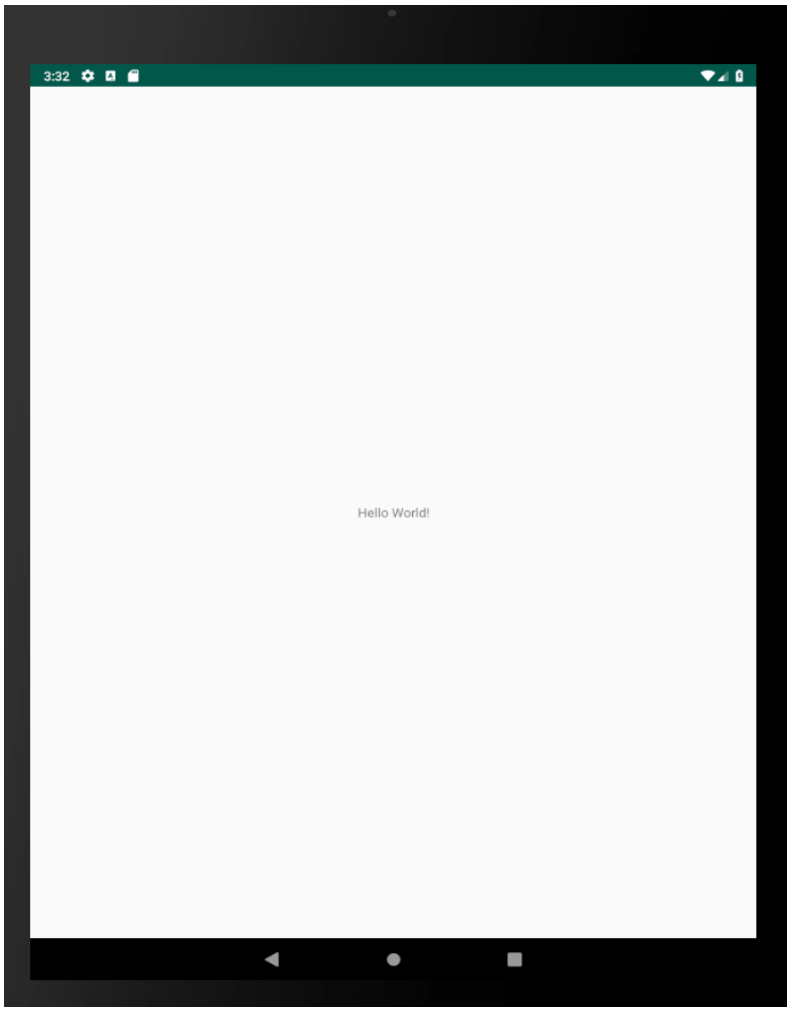
- Ouvrez le fichier styles.xml se trouvant dans le répertoire res/values.
- Dans la première ligne de ce fichier, vous avez un "DarkActionBar" qui vous affiche la barre d'action en question comme vous pouvez l'observez à la ligne suivante.

```
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
```

- Modifier l'attribut parent pour obtenir ce qui suit :

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

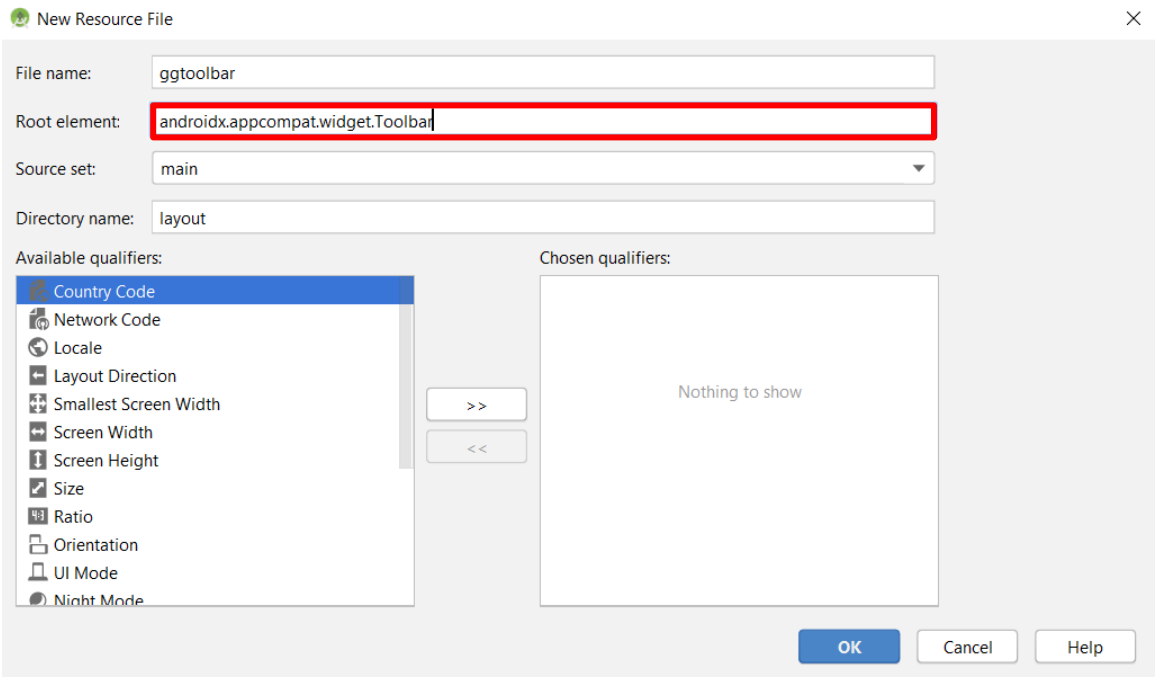
- Lancez votre application à nouveau et remarquez bien que la barre disparait comme à l'écran ci-dessous. Vous pouvez le remarquer aussi dans votre fenêtre d'activité.



- Ouvrez le fichier AndroidManifest et remarquez bien qu'on demande à l'application d'utiliser le style que vous venez de modifier à travers l'attribut android:theme="@style/AppTheme". Lignes ci-dessous.

```
<application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
```

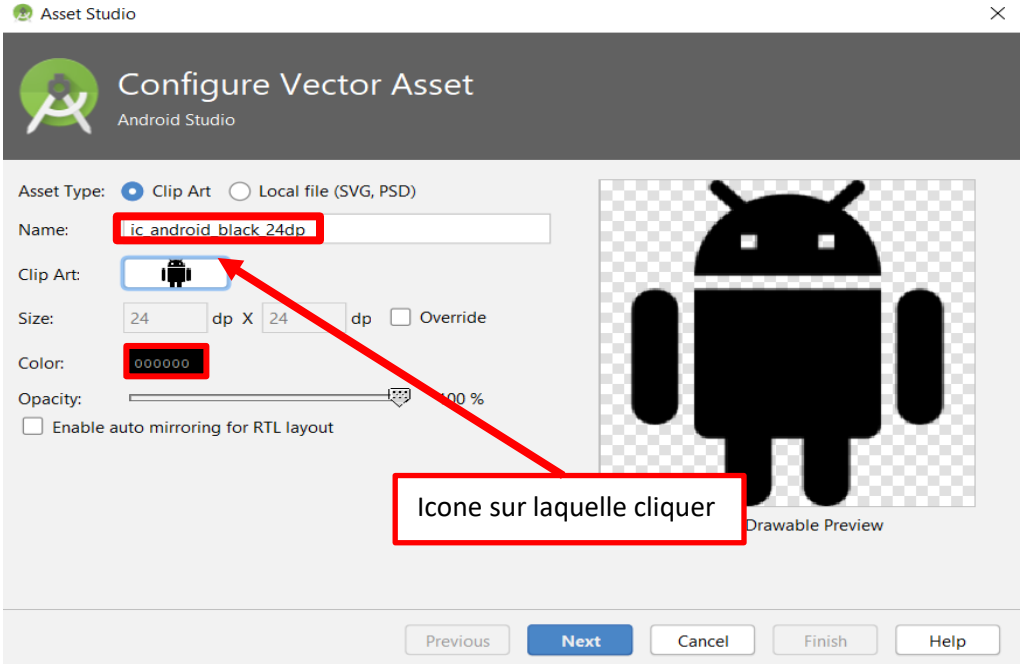
- Ouvrez le fichier de Layout correspondant à votre activité et modifiez le *textView* qui est affiché par défaut. Modifiez la largeur à match_constraint. Modifiez le texte pour que ça affiche *au milieu* "**Menu et Toolbar**". N'oubliez pas d'utiliser le fichier strings.xml.
- Créez un fichier ggtoolbar.xml de layout (Layout Resource File) et modifiez les valeurs comme affichées ci-dessous. Remarquez bien que le "root element" est androidx.appcompat.widget.Toolbar.



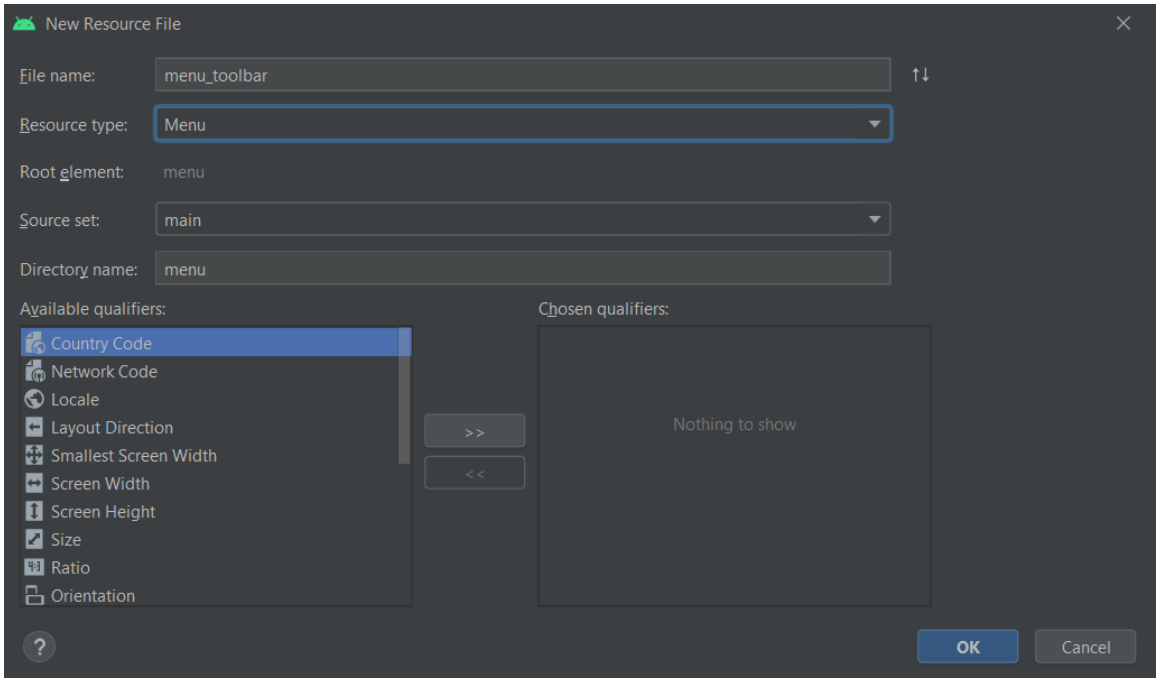
- Jetez un coup d'œil sur la documentation du site d'Android pour comprendre les barres d'outils (https://developer.android.com/reference/android/support/v7/widget/Toolbar.html)

- Dans le répertoire drawable, ajoutez une icône à utiliser dans le menu. Faites un clic droit sur le répertoire drawable, cliquez sur **New->Vector Asset**. Cliquez sur l'icône de clipart pour choisir une image comme indiqué ci-dessous. Vous pouvez voir les images disponibles à l'adresse suivante : (https://design.google.com/icons/#ic_settings).

Choisissez l'image de "settings". Modifiez la couleur du clipart en bleu ainsi que le nom du fichier xml en **ic_baseline_settings_24**. Cliquez sur **Next** et **Finish** pour sauvegarder le fichier.



Faites un clic droit sur le dossier res -> New -> Android Resource Files et remplissez la fenêtre comme ci-dessous et cliquez sur **OK**.



- Allez dans le fichier *menu_toolbar*, ajouter les items de menu qui sont donnés dans le tableau ci-dessous. Vous pouvez utiliser la palette des composants pour

ajouter les items à votre menu. Les différents items à ajouter vous sont donnés dans le tableau ci-dessous. Utilisez la même approche du glisser-déposer des fichiers de layout. L'élément racine étant un élément de type Menu. Les valeurs de titre doivent provenir du fichier strings.xml.

Item	id	title	icone	showAsAction
Item 1	menu_toolbar_settings	Paramètres	ic_settings_gray_24dp	ifRoom
Item 2	menu_toolbar_profile	Profil	ic_person_gray_24dp	ifRoom
Item3	menu_toolbar_share	Partager	@android:drawable/ic_menu_share	always
Item4	menu_toolbar_add	Ajouter	@android:drawable/ic_menu_add	always|collapseActionView
Item 5	menu_toolbar_help	Aide	@android:drawable/ic_menu_help	collapseActionView

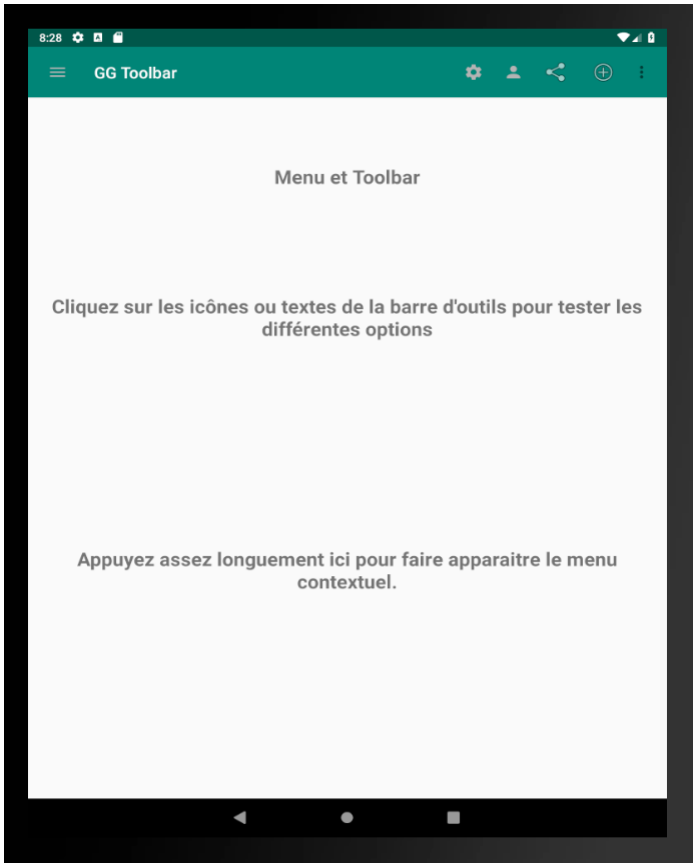
- Pour le premier item, vous aurez inséré les informations suivantes par exemple:
  - id: menu_toolbar_settings
  - title: Paramètres (Créez votre string dans le fichier strings.xml)
  - icon: @drawable/ic_baseline_settings24 (Fichier créé antérieurement)
  - showAsAction: ifRoom

Ajouter la chaine menu_toolbar_settings dans le fichier strings.xml avec pour valeur "Settings"

Ce qui correspond en XML au résultat suivant:

```
<item
    android:id="@+id/menu_toolbar_settings"
    android:icon="@drawable/ic_baseline_settings_24"
    android:title="@string/menu_toolbar_settings"
    app:showAsAction="ifRoom" />
```

- Dans le fichier activitymain.xml, dans la palette des composants, glissez-déposer le widget **include** sur votre interface. Choisissez le fichier de layout que vous avez créé et qui contient votre toolbar **(@layout/ggtoolbar**). Modifiez la propriété layout_width à math_constraint et ajoutez les contraintes (horizontales et verticales) pour que le widget occupe la largeur de l'écran, et soit situé en haut au-dessus du textview ***Menu et Toolbar***. Donnez un id **ggtoolbar** à la balise **include**. Il ne doit y avoir aucune marge ou padding pour le toolbar.
- Ajoutez trois textView à votre interface pour avoir un écran comme ci-dessous.



- Dans le fichier ***MainActivity***, récupérez le toolbar en utilisant la fonction **findViewById** dans une variable ggtoolbar.
- Pour utiliser le toolbar, on doit le préciser dans notre activité. Dans la fonction **onCreate()**, ajoutez la ligne suivante pour ce faire:

```
setSupportActionBar(ggtoolbar as Toolbar?)
```

Le toolbar que vous devez importer est le suivant:

```
import androidx.appcompat.widget.Toolbar;
```

- Ajoutez les méthodes suivantes à la classe **MainActivity** :

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
        menuInflater.inflate(R.menu.menu_toolbar, menu)
        return true
    }
```

- La méthode onOptionsItemSelected sera appelée au clic d'un item.

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return super.onOptionsItemSelected(item)
}
```

- Comprendre le code de la méthode ***onCreateOptionsMenu(Menu menu)*** de *MainActivity* qui permet de rattacher le menu à l'activité.
- Tester votre application. Le menu est présent en haut à droite.

- Ajoute*r la ligne suivante au fichier strings.xml :*
  <string name="menu_selectionne">Item de menu %s sélectionné</string>
  Remarquez le placeholder %s qui est utilisé dans ce cas. Cette information peut être passée en paramètre comme dans la fonction getString ci-dessous.
- En procédant comme précédemment, ajouter un item de menu *menu_toolbar_option1* qui doit être visible seulement quand on clique sur les trois points verticaux à droite. Cette option doit contenir 2 sous-menus option1.1 et option1.2. Vous devez donc ajouter deux sous-menus (option1.1 et option1.2) au menu_toolbar_option1. Pour ce faire, ajoutez les items de sous-menus à partir de la palette comme vous aviez fait pour les items de menu principaux (glisser-déposer). Le résultat en XML sera semblable à ce qui suit:

```
<item android:title="@string/option1" android:id="@+id/menu_toolbar_option1"
app:showAsAction="never">
    <menu>
        <item android:title="@string/option1_1"
android:id="@+id/menu_toolbar_option1_1"/>
        <item android:title="@string/option1_2"
android:id="@+id/menu_toolbar_option1_2"/>
    </menu>
</item>
```

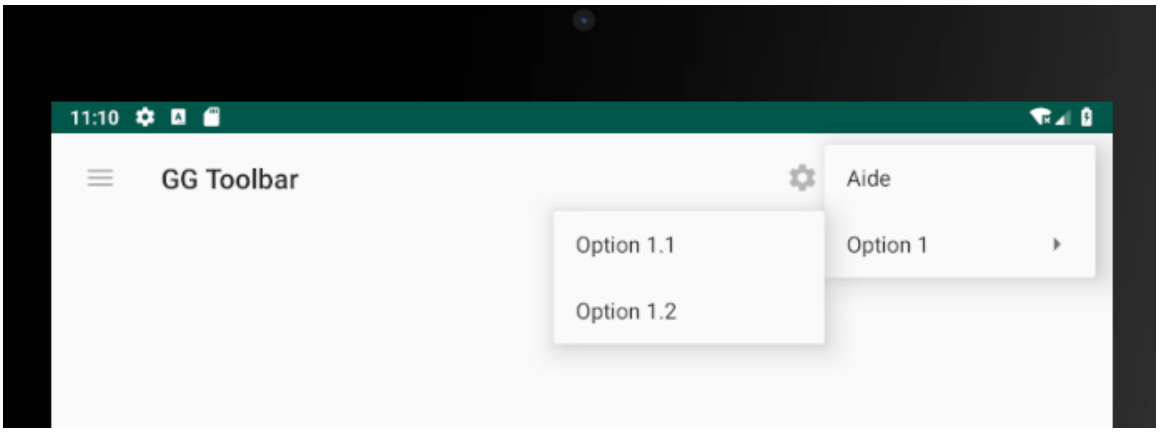
- Pour afficher un toast quand on clique sur les items de menu, modifiez le code de la fonction onOptionsItemSelected comme suit

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    Toast.makeText(this,
        getString(R.string.menu_selectionne, item.title),
        Toast.LENGTH_SHORT
    ).show()

    when (item.itemId) {
        R.id.menu_toolbar_settings -> return true
        R.id.menu_toolbar_option1 -> return true
    }

    return super.onOptionsItemSelected(item)
}
```

- Lancez l'application. Appuyez sur l'item de menu *"Paramètres"*. Vérifier que le message "Item de m*enu Paramètres sélectionné" est* affiché dans le Toast. Tester aussi avec les autres items de menu. Vous devez avoir un message similaire.

- Modifiez le menu pour que l'item de menu "Paramètres" s'affiche quand on clique de préférence sur les trois points verticaux à droite.

- Tester votre application
- Vous devez avoir un écran similaire à celui-là quand vous cliquez sur les trois points verticaux -> Option 1.



2. Menu Contextuel

Le menu contextuel est affiché en appuyant longuement sur un widget de votre interface. Si un tel menu est déclaré par la vue, il apparaît dans une fenêtre en premier plan.

- Compléter la vue de *activity_main.xml* comme suit :

- Donner un identifiant **mainConstraintLayout** au ConstraintLayout de base de *activity_main.xml*.
- Ajouter au sous-dossier *menu*, un fichier *menu_contextuel.xml (Clic droit sur le dossier menu->New->Menu resource file)*. Saisissez le nom du fichier(menu_contextuel) et cliquez sur OK.

  Notre menu contextuel contient un menu *option1* ainsi que deux groupes contenants chacun deux items de menu.
- Copier et bien comprendre le code XML en gras suivant dans le fichier *menu_contextuel.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
  <item android:id="@+id/menu_contextuel_option1"
        android:title="@string/menu_contextuel_option1">
  </item>

    <group android:id="@+id/group1"
```

```
            android:checkableBehavior="single">
            <item  android:id="@+id/menu_contextuel_option2"
                    android:checked="true"
                    android:title="@string/menu_contextuel_option2">
             </item>
              <item android:id="@+id/menu_contextuel_option3"
                    android:title="@string/menu_contextuel_option3">
            </item>
    </group>
    <group android:id="@+id/group2"
           android:checkableBehavior="all">
          <item android:id="@+id/menu_contextuel_option4"
                android:checked="true" android:title="@string/menu_contextuel_option4">
          </item>
            <item android:id="@+id/menu_contextuel_option5"
                    android:title="@string/menu_contextuel_option5"></item>
    </group>
</menu>
```

- Définir toutes les variables chaines de caractères non définis dans le fichier strings.xml. Les chaines *menu_contextuel_option<n>* auront pour valeur *"Option <n>". Par exemple la chaine menu_contextuel_option1 aura pour valeur "Option 1".*
- Pour pouvoir utiliser ce menu contextuel, ajouter le code suivant à *MainActivity.kt.*

```
override fun onCreateContextMenu(menu: ContextMenu, v: View, info:
ContextMenu.ContextMenuInfo?) {
    when (v.id) {
        R.id.mainConstraintLayout ->
menuInflater.inflate(R.menu.menu_contextuel, menu)
        else -> super.onCreateContextMenu(menu, v, info)
    }
}

override fun onContextItemSelected(item: MenuItem): Boolean {
    Toast.makeText(
        this,
        getString(R.string.menu_selectionne, item.title),
Toast.LENGTH_SHORT
    ).show()
    when (item.itemId) {
        R.id.menu_contextuel_option2, R.id.menu_contextuel_option3,
R.id.menu_contextuel_option4, R.id.menu_contextuel_option5 -> {
            item.isChecked = !item.isChecked
            return true
        }
    }
    return super.onContextItemSelected(item)
}
```

// Ajouter ce code dans la méthode onCreate()

```
val view = findViewById<View>(R.id.mainConstraintLayout);
//récupérer l'identifiant du layout principal
registerForContextMenu(view);  // spécifier qu'il prend en charge un menu contextuel
```

- Importez les packages nécessaires.
- Tester votre application. Voici le résultat de l'exécution après avoir lancé l'application et faire un clic droit sur l'écran.


8:54
GGToolbar
Menu et toolbar
Cliquez sur les icones ou textes de la barre d'outils pour tester les différentes options
Appuyez longtemps ici pour
menu contextuel.
Option 1
Option 2
Option 3
Option 4
Option 5

3. Icône de l'application, deuxième activité

- Modifiez le fichier contenant la barre d'outils pour ajouter un logo (Utilisez "Image Asset" ou ajoutez vos propres images) à l'activité.
- Vous devez afficher un message à l'utilisateur quand il clique sur le logo de l'application *"Vous avez cliqué sur l'icône de la barre d'outils"* et lui afficher une deuxième activité que vous allez créer tout de suite. Ajoutez le code nécessaire à votre fichier MainActivity pour mettre un listener sur votre logo pour ce faire.
- Créez une deuxième activité **Main2Activity** avec un bouton retour sur la première activité et un textView affichant "Vous êtes à la deuxième activité. Cliquez sur le bouton Retour pour aller à la première activité". Après avoir créé la deuxième activité, précisez la première activité comme son activité parente dans le fichier AndroidManifest en ajoutant l'attribut "parentActivityName" à la deuxième activité.

```
<activity android:name=".Main2Activity"
android:parentActivityName=".MainActivity"></activity>
```

  Quand l'utilisateur clique sur le logo de la première activité, la deuxième activité doit être affichée et quand il clique sur le bouton **Retour** de la deuxième activité, il doit retourner à la première activité. Programmez le bouton **Retour** pour ce faire (Images ci-dessous).
- Tester votre application.