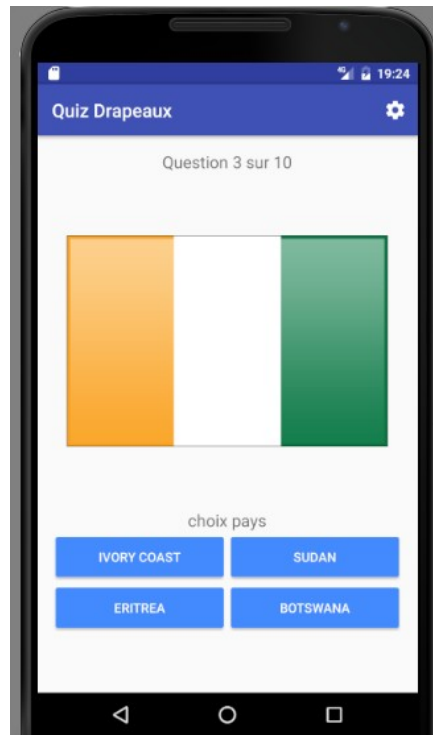


Atelier 3 : Quiz drapeaux

Le but de cet atelier est de réaliser une application qui permet de créer un jeu de quiz contenant 10 question à choix multiple concernant les drapeaux de différents pays. l'interface de l'application aura l'allure suivante :



1. Création du projet

- a) Créer un nouveau projet utilisant le template *Basic Activity* pour avoir une barre de menu.
- b) Cocher la case **use fragment** pour utiliser les fragments. Laisser les valeurs par défaut de *Activity Name* , *Layout Name* , *Title* et *Menu Resource Name*
- c) L'IDE crée entre autre les fichiers suivants :
 1. une classe MainActivity
 2. une sous classe MainActivityFragment de la classe Fragment
 3. menu_main.xml où on définit les options de du menu.
 4. activity_main.xml : contient un CoordinatorLayout qui définit un template contenant un toolbar tout en assurant la compatibilité avec les anciennes versions. Ce Layout permet de rajouter des dépendances entre plusieurs vues adjacentes. En fonction du comportement qu'on leur affecte, elles peuvent par exemple se déplacer ensemble. Par défaut ce template contient un FloatingActionButton (bouton rond en rose) qu'on supprimera (pas besoin pour cette application)
 5. content_main.xml : définit la partie de l'interface graphique du MainActivity qui apparaît sous la barre d'application et au-dessus de la barre de système. Lorsque vous choisissez l'option Fragment, ce fichier ne contient qu'un <fragment> qui affiche l'interface graphique du MainActivityFragment défini dans fragment_main.xml. Si vous ne choisissez pas l'option Fragment, ce fichier définit un RelativeLayout contenant un TextView.
 6. fragment_main.xml
- d) Ajouter une icône au projet
- e) Sélectionner Nexus 6 comme périphérique virtuel

2. Ajout des images des drapeaux au projet

- Créer un répertoire nommé assets en cliquant par le bouton droit sur app New > Folder > Assets Folder
- Copier tout les répertoires contenu dans le répertoire images (fourni) dans assets

3. string.xml

Dans les précédents ateliers nous avons appris à ajouter une String resource en utilisant le Resources dialog. Dans cet atelier on préparera à l'avance les chaînes dont on aura besoin

- ouvrir string.xml
- aller à open editor (en haut à droite) pour ouvrir le Translations Editor .
- Entrer nombre_de_choix dans key et nombre de choix dans Default Value, puis valider
- répéter ces tâches pour les ressources suivantes :

Key	Default Value
nombre_de_choix_description	Afficher 2, 4, 6 ou 8 choix
monde_regions	Régions
monde_regions_description	Régions à inclure
choix_pays	Choisir pays
resultats	%1\$d faux, %2\$.02f%% correctes
reponse_incorecte	Incorrecte!
message_region_default	Une région doit être sélectionner. La région par défaut est Africa
restarting_quiz	Quiz redémarre avec votre nouvelle configuration
question	Question %1\$d sur %2\$d
reset_quiz	Reset Quiz
image_description	Image du drapeau courant
default_region	Africa

la notation de 1\$ indique que la première valeur à insérer dans la chaîne devrait remplacer le spécificateur de format %1\$d. De même, 2\$ indique que la deuxième valeur à insérer dans la chaîne devrait remplacer le spécificateur de format % 2.02f\$. Le d au premier spécificateur de format formate un nombre entier et f dans le second format un nombre à virgule flottante. Dans les versions localisées de strings.xml, le spécificateurs de format %1\$d et %2\$.02f sont nécessaires pour traduire correctement la ressource String.

4. arrays.xml

Techniquement, toutes les ressources de votre application dans le dossier res/values peuvent être définies dans le même fichier. Cependant, pour faciliter la gestion des différents types de ressources, des fichiers séparés sont généralement utilisés pour chacun. Par exemple, les ressources de type tableau sont dans arrays.xml, couleurs dans Colors.xml, String dans strings.xml et les valeurs numériques dans values.xml. Cette application utilise trois tableau de chaîne qui sont définis dans arrays.xml:

- regions_list spécifie les noms des régions du monde avec leurs mots séparés par des underscores ces valeurs sont utilisées pour charger les noms de fichiers d'image à partir des dossiers appropriés et que les valeurs sélectionnées pour les régions dans le SettingsActivityFragment. (Africa , Asia , Europe , North_America , Oceania , South_America)
- regions_list_configuration spécifie les noms des régions du monde avec leur mots séparés par des espaces ces valeurs sont utilisées dans le SettingsActivityFragment pour

afficher les noms de régions dans des cases à cocher à l'utilisateur. (Africa , Asia , Europe , North America , Oceania , South America)

- choix_list : liste des valeurs (2,4,6,8) des choix possibles.

Pour créer arrays.xml , procédez comme suit:

- a) Dans le dossier res du projet, cliquez droit sur le dossier values, puis sélectionnez New > Values resource file pour afficher la boîte de dialogue New Resource File. Parce que vous avez cliqué sur le dossier des valeurs, la boîte de dialogue est préconfiguré pour ajouter un fichier de values ressources.
- b) Spécifiez arrays.xml dans le champ Name, puis cliquez sur OK pour créer le fichier.
- c) Android Studio ne fournit pas un éditeur de ressources de chaîne pour les tableaux de chaîne, de sorte que vous aurez besoin de modifier le XML pour créer les ressources. Chaque tableau est décrit par :

```
<string-array name=" nom de ressourcevaleur du premier élément </item>
<item> valeur du second élément </item>
...
</string-array>
```

5. colors.xml

Cette application affiche les réponses correctes en vert et les réponses incorrectes en rouge. Comme pour toute autre ressource, les ressources de couleur doivent être définies en XML de sorte que vous pouvez facilement changer les couleurs sans modifier le code source Java.

En générale, les couleurs sont définies dans le fichier Colors.xml, qui est créé pour vous par la plupart des modèles d'Android Studio, sinon, vous devez créer le fichier.

Ici, vous allez ajouter des ressources de couleur pour les réponses correctes et incorrectes et de modifier la couleur d'accent de l'application. Pour ce faire, vous modifiez le XML directement.

Ouvrir Colors.xml de ajouter les lignes :

```
<color name="reponse_correcte">#00CC00</color>
<color name="reponse_incorrecte">#FF0000</color>
```

De plus, modifiez la valeur nommée colorAccent à # 448AFF (une nuance de bleu)

6. button_text_color.xml


Pour définir une coloration différente des boutons dans les états enable et disable on déclare les états sous forme d'items dans un selector, pour cela :

- a) faire un clic droit sur res et choisir New > Andorid resource file
- b) spécifier button_text_color.xml comme File name
- c) sélectionner Color dans Resource type
- d) valider et ajouter le code suivant dans le selector

```
<item
    android:color="@android:color/primary_text_dark"
    android:state_enabled="true"/>
<item
    android:color="@android:color/darker_gray"
    android:state_enabled="false"/>
```

7. menu_main.xml

Pour ajouter une icône de configuration au menu on procède comme suit :

- a) Aller à File > New > Vector Asset puis cliquer sur icon, cela fourni une liste d'icônes recommandées par Google (<https://www.google.com/design/icons/>)
- b) choisir l'icône qui convient,  par exemple, et valider

- c) par défaut la couleur de l'icône est noir et vous pouvez la changer dans le fichier XML correspondant
- d) Ouvrir menu_main.xml et indiquer le chemin de l'icône choisie dans la propriété android:icon

```
android:icon="@drawable/ic_settings_black_24dp"
```

- e) pour forcer l'affichage du menu dans le toolbar changer la propriété app:showAsAction à always

8. Création d'une animation de vibration du drapeau en cas de réponse incorrecte

Pour créer une animation :

- a) faire un clic droit sur res et choisir New > Android resource file
- b) File name : vibration_incorrect.xml
- c) Resource type : Animation
- d) valider et modifier le contenu comme suit

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/decelerate_interpolator" >
    <translate android:duration="100" android:fromXDelta="0"
        android:toXDelta="-5%p" />
    <translate android:duration="100" android:fromXDelta="-5%p"
        android:toXDelta="5%p" android:startOffset="100" />
    <translate android:duration="100" android:fromXDelta="5%p"
        android:toXDelta="-5%p" android:startOffset="200" />
</set>
```

d'autres actions peuvent être définies et combinées : alpha (transparence), scale (redimensionnement), rotate (rotation).

Consulter <https://developer.android.com/guide/topics/resources/animation-resource.html> pour plus de détails.

9. preferences.xml pour la configuration de l'application

On crée un fichier preferences.xml que le SettingsActivityFragment utilise pour afficher les préférences de l'application.

- a) faire un clic droit sur res et choisir New > Android resource file
- b) File name : preferences.xml
- c) Resource type : XML.
- d) Valider et éditer le fichier
- e) Ajouter deux types de préférences au fichier ListPreference et MultiSelectListPreference avec les propriétés suivantes :

1. ListPreference

Propriété	Valeur	description
entree	@array/choix_list	Tableau de chaînes qui sera afficher dans la liste d'options
entreeValeurs	@array/choix_list	Un tableau de valeurs associées aux options de la propriété entrées. La valeur d'entrée sélectionnée sera enregistrée dans les SharedPreferences
key	pref_nombreChoix	Le nom de la préférence stockée dans les SharedPreferences
titre	@string/nombre_de_choix	Le titre de la préférence affichée dans l'interface
sommaire	@string/nombre_de_choix_description	Une description sommaire de la préférence qui est affiché en dessous de son titre.

persistent	true	Que la préférence devrait persister après l'application se termine, si vrai, la classe PreferenceFragment persiste immédiatement la valeur de préférence à chaque fois qu'il change.
valeurDefaut	4	Nombre de choix par défaut

2. MultiSelectListPreference

Propriété	Valeur	description
entree	@array/ regions_list_configurati on	Tableau de chaînes qui sera afficher dans la liste d'options
entreeValeurs	@array/regions_list	Un tableau de valeurs associées aux options de la propriété entrées. La valeur d'entrée sélectionnée sera enregistrée dans les SharedPreferences
key	pref_regionsAInclure	Le nom de la préférence stockée dans les SharedPreferences
titre	@string/ monde_regions	Le titre de la préférence affichée dans l'interface
sommaire	@string/monde_regions _description	Une description sommaire de la préférence qui est affiché en dessous de son titre.
persistent	true	La préférence persiste après la terminaison de l'application
valeurDefaut	@array/regions_list	Tableau des régions par défaut

10. Ajout des classes ConfigActivity et ConfigActivityFragment

- Faire un clic droit sur le répertoire app et New > Activity > Basic Activity
- Activity Name : ConfigActivity
- Title : Configuration
- cocher Use a Fragment
- Hierarchical Parent : MainActivity
- valider
- ouvrir activity_config.xml et supprimer le FloatingActionButton

11. activity_main.xml

Le fichier content_main.xml défini par l'IDE contient un <fragment> comme layout principal. Lors de l'exécution, l'interface graphique du MainActivityFragment remplira la partie de l'écran occupée par ce <fragment>.

Modifier l'id de ce fragment à quizFragment.

12. fragment_main.xml Layout

on s'intéresse maintenant au design de l'application qui aura cette allure :

numeroQuestionTextView

drapeauxImageView

choixPaysTextView

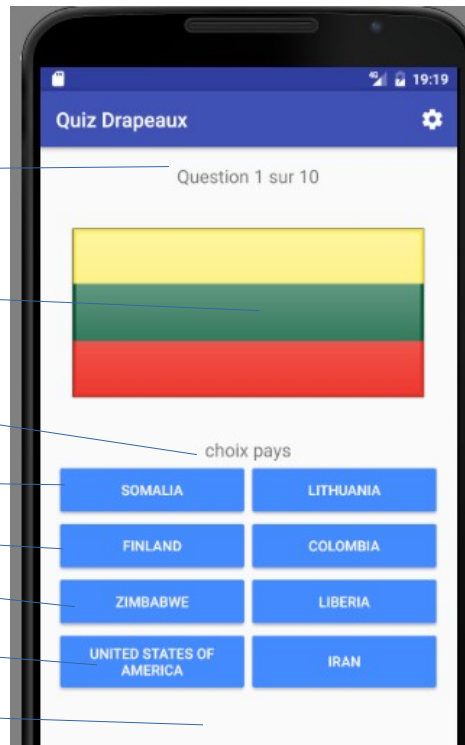
row1LinearLayout

row2LinearLayout

row3LinearLayout

row4LinearLayout

reponseTextView



- Changer le RelativeLayout en LinearLayout dans fragment_main.xml avec orientation verticale, la largeur et hauteur à match_parent et l'id à quizLinearLayout
- Ajouter un TextView d'id numeroQuestionTextView dans le quizLinearLayout en le centrant horizontalement, le layout:margin : @dimen/spacing que vous définissez à 8dp uniquement pour bottom margin, la propriété text à @string/question déjà définie
- Ajouter un ImageView dans le quizLinearLayout avec l'id : drapeauxImageView et avec les propriétés suivantes :
 - layout:width : match_parent
 - layout:height : 0dp
 - layout:gravity center : both
 - layout:margin bottom : @dimen/spacing
 - layout:margin left et right : @dimen/activity_horizontal_margin
 - layout:weight : 1
 - adjustViewBounds : true
 - contentDescription : @string/image_description
 - scaleType : fitCenter
- Ajouter un textView d'id choixPaysTextView avec les propriétés suivantes :
 - layout:gravity center : horizontal
 - text : @string/choix_pays
- Ajouter les 8 boutons au LinearLayout
 - Ajouter un LinearLayout (Horizontal) au quizLinearLayout avec id=row1LinearLayout et layout:height à wrap_content .
 - Ajouter un Button au row1LinearLayout
 - répéter l'étape 2 pour l'autre bouton de la première ligne
 - répéter les étapes 1 à 3 pour les 3 autres LinearLayout en indiquant leurs id respectifs
 - sélectionner les 8 buttons et changer les propriétés :
 - layout:width à 0dp et layout:weight à 1
 - faite en sorte que les boutons partagent équitablement l'espace horizontal
 - layout:height à match_parent

4. style à `@android:style/Widget.Material.Button.Colored` et définir la couleur du texte et des boutons dans les cas enable et disable
- f) Ajouter le textView reponseTextView avec comme propriétés :
1. layout:gravity : cocher bottom et center to horizontal
 2. gravity : center_horizontal
 3. textSize : `@dimen/taillre_reponse` créer cette dimension à 36dp
 4. textStyle : bold

13. Implémenter les classes nécessaires au déroulement du quiz