8. Menus d'options

Implanter un menu d'options (*overflow menu*), un `FloatingActionButton`, un `Snackbar` et un `Toast`.

Android Studio Flamingo Essentials, Kotlin Edition: chapitre 52

8.1. Introduction

1. Expliquez brièvement ce qu'est un menu d'options dans une application *Android*

2. Exécutez l'application afin d'avoir accès à l'émulateur et illustrer ce qu'est un menu d'options

   a. Dans l'émulateur, retournez à la page d'accueil de l'appareil, et sélectionnez l'application de messagerie

   b. Touchez le menu d'options dans la barre d'entête de l'application et expliquez les différentes options d'affichage d'items de menu

3. Soulignez que le gabarit de projet « Empty Views Activity » n'inclue pas par défaut la barre d'application; il faut donc activer cette option dans tout nouveau projet basé sur ce gabarit

8.2. Introduction

4. Créez un projet *Android Studio* avec le gabarit « Empty Views Activity », nommé **`Menus`**

   a. Ne pas oublier d'activer la liaison de vues

      i. Dans *app » Gradle Scripts » build.gradle.kts (Module:apps)* ajoutez :

```
buildFeatures {
    viewBinding = true
}
```

ii. Modifiez `MainActivity.kt` :

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        ...
    }
```

et accédez aux widgets de l'activité via **`binding`**.

5. Dans le nouveau projet, éditez les deux fichiers du répertoire `app » res » values » themes`, soient `themes.xml` et `themes.xml (night)`, afin d'en supprimer l'option `NoActionBar` :

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Base.Theme.Menus" parent="Theme.Material3.DayNight.NoActionBar">
        <!-- Customize your light theme here. -->
        <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
    </style>

    <style name="Theme.Menus" parent="Base.Theme.Menus" />
</resources>
```

6. Expliquez que le gabarit de projet « Empty Views Activity » n'intègre pas un menu d'options à l'activité générée

   a. Il faut donc le faire soi-même, ou utiliser un autre gabarit

7. Pour ajouter au projet un fichier XML définissant un menu d'options, il faut au préalable avoir un répertoire à cet effet dans le projet

   a. Si le répertoire `app » res » menu` n'existe pas, créez-le en cliquant le bouton droit de la souris sur le répertoire `app » res` dans le panneau Project, puis sélectionnez la commande New » Android resource directory

   b. Au champ Resource type, sélectionnez la valeur « menu »

   c. Cliquez « OK » pour créer le nouveau répertoire

8. Ajoutez au projet un fichier XML définissant un menu d'options

   a. Cliquez le bouton droit de la souris sur le répertoire `app » res » menu` dans le panneau Project, puis sélectionnez la commande New » Menu resource file

   b. Nommez le nouveau fichier **`menu_main.xml`**

9. Définissez trois items de menu dans le nouveau fichier

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/menu_rouge"
        android:title="Rouge"
        app:showAsAction="never" />
    <item
        android:id="@+id/menu_vert"
        android:title="Vert"
        app:showAsAction="never" />
    <item
        android:id="@+id/menu_belu"
        android:title="Bleu"
        app:showAsAction="never" />
</menu>
```

10. Dans le code *Java* de l'activité, surchargez la fonction `onCreateOptionMenu()` afin que le menu soit affiché dans la barre supérieure de l'activité

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    ...

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        var inflater = getMenuInflater()
        inflater.inflate(R.menu.menu_main, menu)

        return super.onCreateOptionsMenu(menu)
    }
```

11. Expliquez qu'il faut maintenant surcharger la fonction `onOptionsItemSelected()` qui est responsable de gérer la sélection des items du menu d'options

    a. Au préalable, attribuez l'identificateur `mainLayout` au `ConstraintLayout` de l'activité

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    when (item.itemId) {
        R.id.menu_rouge -> binding.mainLayout.setBackgroundColor(Color.RED)
        R.id.menu_vert  -> binding.mainLayout.setBackgroundColor(Color.GREEN)
        R.id.menu_bleu  -> binding.mainLayout.setBackgroundColor(Color.BLUE)
    }

    return super.onOptionsItemSelected(item)
}
```

    b. Exécutez l'application pour démontrer son fonctionnement

12. Montrez maintenant qu'un item de menu peut être accompagné d'une boîte à cocher :

```
<item
    android:id="@+id/menu_rouge"
    android:title="Rouge"
    app:showAsAction="never"
    android:checkable="true" />
<item
    android:id="@+id/menu_vert"
    android:title="Vert"
    app:showAsAction="never"
    android:checkable="true" />
<item
    android:id="@+id/menu_bleu"
    android:title="Bleu"
    app:showAsAction="never"
    android:checkable="true" />
```

    a. Exécutez à nouveau l'application pour démontrer que les items sont maintenant accompagnés d'une boîte cochable

13. Soulignez cependant que l'item de menu sélectionné n'est pas automatiquement coché

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    item.isChecked = true

    ...
```

    a. Exécutez à nouveau l'application pour démontrer que l'item sélectionné est maintenant coché

14. Soulignez qu'un item de menu ne se décoche pas automatiquement lorsqu'un autre item est sélectionné, et qu'il existe deux façons d'y remédier

    a. On doit le faire soi-même

```
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private var menu: Menu? = null

    ...

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        var inflater = getMenuInflater()
        inflater.inflate(R.menu.menu_main, menu)

        this.menu = menu

        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        menu?.findItem(R.id.menu_rouge)?.isChecked = false;
        menu?.findItem(R.id.menu_vert)?.isChecked = false;
        menu?.findItem(R.id.menu_bleu)?.isChecked = false;

        item.isChecked = true

        ...
```

15. Montrez l'éditeur de menus intégré à *Android Studio*

    a. Sélectionnez le fichier `menu_main.xml` du panneau Project (mode *Android*), et affichez l'éditeur Design

    b. Expliquez que l'éditeur facilite la construction de menus d'options

    c. L'éditeur Design devrait afficher graphiquement le menu. S'il ne le fait pas, fermez le projet puis ouvrez-le à nouveau dans *Android Studio*

    d. Expliquez brièvement la signification des valeurs possibles de l'attribut `showAsAction` d'un item de menu

16. On va maintenant utiliser l'éditeur de menus de *Android Studio* pour gérer les coches d'items (afin qu'ils soient exclusifs)

    a. Restaurez le code à ce qu'il était avant l'étape 14

    ```
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        item.isChecked = true

        ...
    ```

    b. Via l'éditeur de menu en mode Design, intégrez les items de menu dans `Group` dont l'attribut `ckeckableBehavior` est mis à **`Single`**

    c. Effacez l'attribut `checkable` (via l'éditeur Code) de chaque `MenuItem`, ce qui remet cette attribut à sa valeur par défaut

    d. Exécutez l'application pour démontrer que ça fonctionne, et soulignez le nouveau style de coche (un cercle plutôt qu'un carré) des `MenuItem`

8.3. Material Design

17. Décrivez brièvement ce qu'est *Material Desing* de Google

    a. C'est un ensemble de règles décrivant le *look and feel* d'une application *Android*

    b. Impose des normes sur l'apparence des widgets et des activités

    c. Objectif : que toutes les applications *Android* aient un look semblable, permettant ainsi à l'utilisateur de se sentir confortable avec l'appareil

8.4. Qu'est-ce qu'bouton flottant (`FloatingActionButton`) ?

18. En anglais : *floating action button* (*fab*)

19. Bouton animé semblant « flotter » au-dessus de l'activité

20. Généralement utilisé pour donner accès à une fonctionnalité importante ou prioritaire de l'activité

21. Règles de *Material Design*

    a. Doit être de forme prédéfinie (selon la version du OS)

    b. Deux formats : par défaut (56dp x 56dp) ou miniature (40dp x 40dp)

    c. Doit contenir une image de 24dp x 24dp (pas de texte)

    d. Maximum d'un seul bouton flottant dans une activité

22. Ajouter un *fab* dans le layout de l'application via l'éditeur de code (c.à.d. en XML) :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    ...

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="end|bottom"
        android:layout_margin="16dp"
        android:src="@android:drawable/ic_input_add"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

23. Puis ajouter un *listener* au *fab* :

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    binding.fab.setOnClickListener { view ->
        binding.textView.text = "Fab touché"
    }
}
```

8.5. Qu'est-ce qu'un `Snackbar` ?

24. Barre affichable au bas de l'écran

25. Permet d'afficher un descriptif court et un bouton d'action

26. Disparaît par lui-même après quelques secondes, sa durée d'affichage étant configurable

27. L'exemple suivant configure le `fab` afin d'afficher un `Snackbar` :

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)

    binding.fab.setOnClickListener { view ->
        val snackbar = Snackbar.make(view, "Voici le Snackbar",
                                     Snackbar.LENGTH_LONG)
            .setAnchorView(R.id.fab);

        snackbar.show()
    }
}
```

    a. Expliquez que l'attribut *`LENGTH_LONG`* (≈ *4 secondes*) peut être remplacé par *`LENGTH_SHORT`* (≈ *2 secondes*) pour que le `Toast` demeure affiché plus longtemps

28. Un `Snackbar` dispose d'un bouton d'action optionnel auquel on peut attribuer un *listener* afin d'effectuer une tâche lorsque touché. L'exemple suivant ferme le `Snackbar` :

```
binding.fab.setOnClickListener { view ->
    val snackbar = Snackbar.make(view, "Voici le Snackbar", Snackbar.LENGTH_LONG)
        .setAnchorView(R.id.fab);

    snackbar.setAction("Fermer", View.OnClickListener { view ->
        binding.textView.text = "Snackbar supprimé"
        snackbar.dismiss();
    })

    snackbar.show()
}
```

29. Soulignez qu'il n'est pas rare que l'action associée à un Snackbar soit d'annuler une opération ayant été effectuée (c.à.d. *UNDO*)

8.6. Qu'est-ce qu'un `Toast` ?

30. Petite fenêtre de type *pop-up* permettant d'afficher temporairement une information succincte

31. La place occupée est ajustée à la taille du message, laissant ainsi l'activité en cours visible et interactive

32. Le `Toast` disparaît automatiquement après un certain délai

33. Le `Toast` est une alternative simplifiée au `Snackbar`

34. Modifiez l'activité afin d'afficher un `Toast` plutôt qu'un `Snackbar` :

```
binding.fab.setOnClickListener { view ->
    Toast.makeText(applicationContext,"Ceci est un message Toast",
                   Toast.LENGTH_SHORT).show()
}
```

    a. Expliquez que l'attribut *LENGTH_SHORT* peut être remplacé par *LENGTH_LONG*, comme pour le `Snackbar`

35. Contrairement au `Snackbar`, un `Toast` ne dispose pas d'une bouton d'action optionnel

Exercice 8.1	Indiquez aux étudiants qu'ils peuvent récupérer l'énoncé de l'exercice (en format PDF) sur le portail éducatif du collège. Ils devraient pouvoir solutionner l'exercice sans aide en se référant au chapitre 31 du livre de référence