Université Paris Cité **Programmation des composants mobiles**
M1 informatique **Année 2022-2023**

TP n° 12

ViewPager2 et fragments

Application à programmer

Le but est d'améliorer l'application développée dans les TP7 et TP8.

Dans cette nouvelle incarnation de Géographie l'application est composée d'une activité qui affiche un de deux fragments, soit `AjouterPaysFragment` soit `AfficherPaysFragment`.

Le changement de fragment affiché se fait soit en choisissant un tab correspondant soit par le mouvement swap sur l'écran comme dons le cours 12.

1 Squelette de deux fragments

Dans `build.gradle` dans la section `dependencies` on ajoute

```
def fragment_version  = "1.5.4"
implementation "androidx.fragment:fragment-ktx:$fragment_version"

implementation "androidx.viewpager2:viewpager2:1.0.0"
```

et toutes les dependances nécessaires pour `room` et `LiveData`

Dans AndroidManifest changer le thème de l'application :

```
android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
```

On commence par écrire le squelette de deux fragments.

Créer le fragment `AjouterPaysFragments` juste avec

```
class AjouterPaysFragment : Fragment(R.layout.fragment_ajouter_pays) {

    companion object {
        @JvmStatic
        fun newInstance() =
            AjouterPaysFragment()
    }

}
```

Créer de façon similaire le fragment `AfficherPaysFragments` en utilisant bien sûr le layout correspondant.

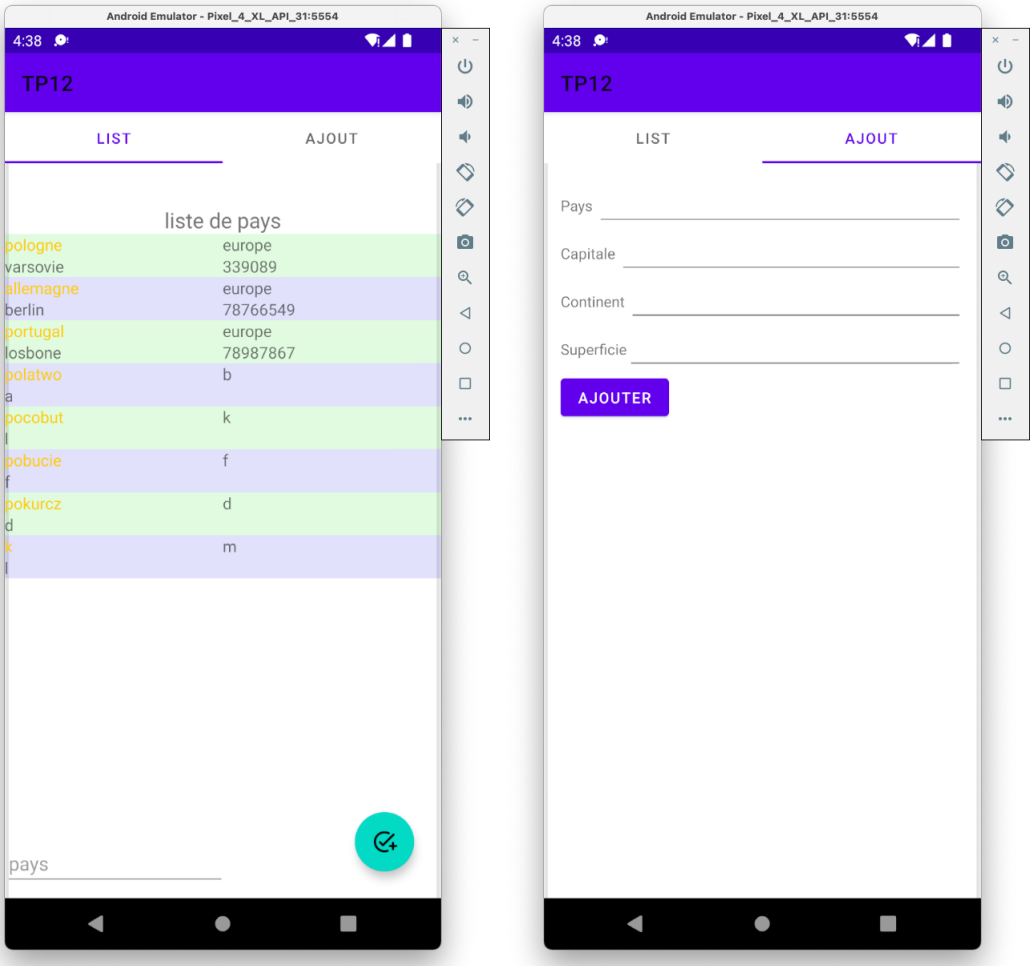
2 Activité

Pour écrire l'activité on procède comme dans le cours 12, la seule chose qui change se sont les noms et le nombre de fragments et les chaînes de caractères affichés dans le `TabLayout`.

Dans le `app/src/main/res` créer les sous-répertoires `drawable-hdpi`, `drawable-mdpi`, `drawable-xhdpi`, `drawable-xxhdpi` et copier dedans le fichier `round_add_task_black_24dp.png`. Le fichier contient l'icône que doit être affiché dans Floating Action Button (FAB).

L'implementation de `FragmentStateAdapter` peut être récopiée dans modification à partir de transparents de cours 12.

Vérifier si vous pouvez changer le fragment affiché dans l'activité par le mouvement swap ou par un clic sur un tab dans `TabLayout`.



3 Floating Action Button

Quand le fragment `AfficherPaysFragment` s'affiche le clic sur le FAB doit ramener sur l'écran le fragment `AjouterPaysFragment`. Pour obtenir cet effet il suffit que la propriété `currentItem` de `ViewPager2` prenne la valeur égale à l'indice du fragment dans le `TabLayout`.

Faire en sorte que le FAB s'affiche uniquement quand le fragment `AfficherPaysFragment` est visible sur l'écran.

Indications. `fab.hide()` et `fab.show()` cache et montre le fab sur l'écran.

La documentation de `ViewPager2` `https://developer.android.com/reference/androidx/viewpager2/wid` indique que la méthode `registerOnPageChangeCallback` permet d'installer un callback qui est appelé par android quand on change le fragment affiché. Regardez les méthodes dans ce callback.

4 Base de donnée, ViewModel, RecycklerView.Adapter

Le Dao, le ViewModel, la base de données et `RecyclerView.Adapter` peuvent être copiés à partir de TP08.

Pour connecter le ViewModel à l'activité il suffit d'ajouter dans l'activité `val model : MyViewModel by viewl` où à la place de `MyViewModel` vous allez mettre le nom de votre View modèle.

`item_layout.xml` est le fichier layout pour un item de `RecyclerView` utilisé dans `AfficherPaysFragment`

5 Fragment AjouterPaysFragment

Les deux fragments utilisent le ViewModel de l'activité ; c'est le moyen de partager les données et les préserver quand on tourne l'appareil. Pour obtenir l'accès à ce ViewModel dans le fragment il suffit :

```
private val model: MyViewModel by activityViewModels()
```

Le code de l'activité qui a affiché la liste de pays dans le TP08 est à copier dans `AjouterPaysFragment`. Mais pas n'importe comment.

Le view binding est à créer dans la méthode `onViewCreated` avec

```
binding = FragmentAjouterPaysBinding.bind( view )
```

où `view` est un paramètre de `onViewCreated`.

Toujours dans `onViewCreated` on récupère les valeurs stockées le bundle de sauvegarde (si vous l'utilisez).

Les opérations sur l'interface graphique, comme par exemple l'installation de `onClickListener` sur le bouton, s'effectuent aussi dans `onViewCreated`.

L'installation des observeurs sur les objets LiveData c'est plutôt dans la méthode `onStart()` de fragment.

Pour obtenir le contexte ce n'est plus `this` (le fragment n'est pas un contexte). On peut essayer la propriété `context` du fragment ou bien la méthode `requireContext()`.

6 Fragment AfficherPaysFragment

Les mêmes consignes s'appliquent au fragment `AfficherPaysFragement`.

7 Tester l'application

Tester si votre application marche correctement.

En particulier effectuer le test suivant.

Ajouter deux pays dont le préfixes sont différents, par exemple Pologne et Allemagne.

Aller vers `AfficherPaysFragment`. Taper le préfixe `Po`. Revenir vers `AjouterPaysFragment`. Ajouter un nouveau pays avec le préfixe `Po`, par exemple `Portugal`.

Revenir vers le fragment `AfficherPaysFragment`. Est-ce que la liste affiche bien tous les pays avec le préfixe `Po`, y compris le Portugal ?

8 Si le temps permet ...

Ajouter un et un bouton `Supprimer` dans le layout de `AfficherPaysFragment`. L'utilisateur sélectionne des pays et le clic sur le bouton provoque la suppression de ces pays de la BD. Dans le `SnackBar` afficher le nombre de pays effectivement supprimés.