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

TP n° 8

Base de données Room

Application à programmer

Le but est d'améliorer l'application développée dans le TP précédent.

1 Activité de recherche

L'activité de recherche a un nouveau layout. Rassurez-vous cela n'a pas d'incidence sur le code que vous vez écrit en TP 7, le nouveau layour conserve tous les éléments de layour précédent et ajoute un `Buton` et deux `RadioButton`.

1.1 Sélection d'un élément de la liste affichée dans le `RecyclerView`

Notez que la racine de `item_layout.xml` est un `CardView`, l'élément auquel on peut attacher un `OnClickListener`.

Il faut faire en sorte que

(1) quand l'utilisateur touche sur un item de la liste affichée dans le `RecyclerView` et le background de cet élément est de couleur `colorPaire` ou `colorImpaire` alors le item prendra une nouvelle couleur background :

```
val colorSelected = Color.argb(0.5f, 0.2f, 0.2f, 0.2f)
```

(2) Si un item de la liste est déjà de couleur `colorSelected` et l'utilisateur touche cet item la couleur background prend soit la valeur `colorPaire` soir `colorImpaire` en fonction de la parité de item sur la liste.

(3) à chaque moment on ne peut avoir qu'au plus un item sélectionné sur la liste. Si l'utilisateur touche un item non-sélectionné et à ce moment il y a déjà un autre item sélectionné ce dernier doit être automatiquement dé-sélectionné.

(4) Bien sûr l'item sélectionné doit être préservé quand on tourne l'appareil.

Spoiler. Le pays sélectionné doit être stocké dans `RechercheViewModel` sinon on ne pourra pas assurer (4). Bien sur l'attribut correspondant pourra aussi prendre la valeur `null` quand aucun pays n'est sélectionné.

Mais le `RecyclerView.Adapter` aura besoin ce pays pour afficher correctement les couleurs. Donc il faut passer la référence de `RechercheViewModel` vers `RecyclerView.Adapter`, le mieux à la construction d'adapter.

Pour implementer la condition (3) (au plus un élément sélectionné) il est utile (nécessaire ?) que le holder dans le `RecyclerView.Adapter` contient une référence vers le pays affiché pas son View. Cela donne la définition suivante :

```
class VH(val binding: ItemLayoutBinding) :
      RecyclerView.ViewHolder(binding.root) {
       var pays: Pays? = null
}
```

1.2 Modifier la fonction `loadPartialName` de Dao

Dans le TP 7 nous avons suggéré que la fonction `loadPartialName` retourne `List<Pays>`. Pour implémenter les opération UPDATE et DELETE il serait plus commode que cette fonction retourne `LiveData<List<Pays>>`.

Les transparents à la fin de cours sur Room (Cours 6) expliquent entre les deux cas et les ajustement dans le code à faire.

1.3 Opération DELETE

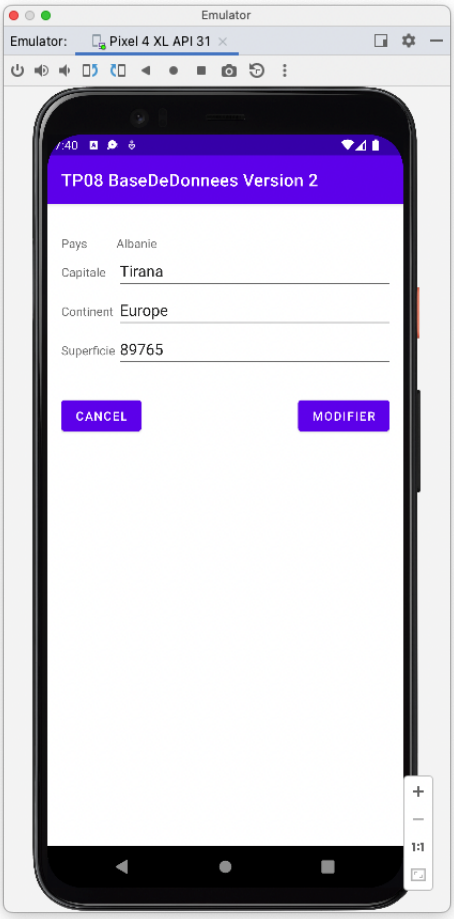
Dans l'activité de recherche, quand le `RadioButton supprimer` est sélectionner et l'utilisateur clique sur le `Button action` alors l'élément sélectionné sur la liste est supprimé de la table de BD et de la liste.

Mais avant la suppression que l'utilisateur confirme la suppression par un dialogue.

S'il n'y a aucun élément sélectionné et l'utilisateur clique sur le `Button action` alors il faut juste affiché un message indiquant que aucun pays n'est sélectionné.

2 Opération UPDATE

Pour implémenter UPDATE il nous faut une nouvelle activité `ModifierActivity`, fichier layout `activity_modifier.xml`



Heureusement, pas besoin de `ViewModel` cette fois.

L'utilisateur clique sur le bouton `action` dans `RechercheActivity` et le `RadioButton modifier` est sélectionné. L'activité `RechercheActivity` envoie vers `ModifierActivity` les quatre attributs du pays sélectionnée sur la liste.

`ModifierActivity` affiche ces quatre attributs dans des `EditText` (le nom de pays n'est pas modifiable et sera affiche dans un `TextView`). L'utilisateur fait des modification de valeurs d'attributs du pays et sur un clique sur le bouton `MODIFIER` renvoie en réponse les nouvelle valeurs [1] vers `RechercheActivity`. A la réception le `RechercheActivity` effectue l'opération UPDATE sur la table. Bien sûr cela sera visible sur le `RecyclerView` mais si vous avez fait tout comme il faut ce sera automatique.

3 Colorier le nom du pays

Dans la recherche quand on cherche par exemple les pays dont le nom commence par `Al` nous voulons que le préfix `Al` s'affiche dans une couleur différente que celle du reste du nom de pays.

Indication. Pour un texte dans deux couleurs :

```
val t : String =
"<font color=#cc0029>prefix</font> <font color=#ffcc00>suffix</font>"

pays.text = HtmlCompat.fromHtml(t, HtmlCompat.FROM_HTML_MODE_LEGACY)
```

1. qui peuvent être les mêmes que les anciennes