Programmation de composants mobiles
Examen le 3 janvier 2023, 18h00-19h45

Documents autorisés

Documents et matériel autorisés :

- tous les documents papier sauf livres,
- la documentation sur le site android :
  - `https://developer.android.com/docs` et
  - `https://developer.android.com/reference/androidx/packages`
- documentation kotlin `https://kotlinlang.org/docs/reference/`
- tous les documents sur le moodle du cours,
- vos propres programmes de TP et votre projet,
- vos appareils android pour tester les programmes.

Il est strictement **interdit** :

- de poser des questions sur les forums de développeurs,
- d'utiliser un chat ou mail,
- utiliser l'ordinateur, tablettes, smartphones à d'autres fins que le développement et le test de vos programmes.

Le sujet comporte 5 pages.

Consignes

L'examen consiste de plusieurs exercices qui permettent de développer un prototype d'une application Annuaire.
Vous devez déposer votre travail dans un dépôt sur moodle. Pour cela vous allez créer le fichier zip depuis AndroidStudio :

```
File -> Export -> Export to ZIP File ...
```

et le déposer sur le moodle du cours dans le dépôt dans la section Examen (il semble que parfois `export to Zip` n'est pas sous l'onglet `Export`, cela dépend de la version d'Android Studio et de système d'exploitation, mais dans tous les cas il faut chercher dans le menu `Fichier`).
Si moodle ne marche pas, en dernier ressort, vous pouvez envoyer le fichier zip par mail à `zielonka@irif.fr` avec le sujet `exam android`. Dans votre mail n'oubliez pas de donner votre nom, prénom, numéro d'étudiant (je vais ignorer les mails dont l'auteur(e) n'est pas identifiable).
Pendant l'examen vous devez développer une seule application mais le travail est divisé en plusieurs exercices.

Il est prudent et vivement conseillé de sauvegarder sur votre ordinateur le fichier zip après chaque exercice réussi. Si vous ne suivez pas cette consigne vous risquez de vous retrouver à la fin de l'examen avec un programme qui ne marche plus. Si vous avez toujours la sauvegarde de la dernière version qui marche vous éviterez ce problème.

Avant de commencer les exercices récupérez sur moodle (section Examen) les fichiers layout et les fichiers kt.
Vous pouvez, et parfois vous devez, modifier à votre guise les fichiers layout fourni. Vous pouvez aussi fabriquer votre propre layout, même si je n'y vois pas d'intérêt.

Préliminaires

Le but est d'implementer une application qui utilise une base de données (avec une seule table) avec l'affichage de contenu dans un `RecyclerView`.
Je rappelle que dans `build.gradle` (Module) dans la section `plugins` il faut ajouter

```
id 'kotlin-kapt'
```

Si vous utilisez `ViewBinding` (recommandé mais pas obligatoire) dans la section `android` vous devez ajouter

```
/* pour ViewBinding */
    buildFeatures {
        viewBinding true
    }
```

Et finalement dans la section `dependencies` vous ajouterez :

```
    implementation 'androidx.gridlayout:gridlayout:1.0.0'
    implementation 'androidx.recyclerview:recyclerview:1.2.1'
    implementation "androidx.cardview:cardview:1.0.0"

    def lifecycle_version = '2.6.0-alpha03'
    // ViewModel
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
    // LiveData
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"

    //Room
    def room_version = "2.4.3"
    implementation "androidx.room:room-runtime:$room_version"
    annotationProcessor "androidx.room:room-compiler:$room_version"

    // To use Kotlin annotation processing tool (kapt)
    kapt "androidx.room:room-compiler:$room_version"
```

1 Annuaire

Le but est d'écrire une application qui permet à terme faire office d'un annuaire d'adresses mail, envoyer et stocker les mails.
Mais pour l'examen on se contentera d'implémenter deux activités, l'activité `Main` gère l'annuaire et l'activité `SendMail` qui prépare un mail à envoyer.
Il est très recommandé d'associer un `ViewModel` (ou `AndroidViewModel`) à l'activité `Main`.

L'activité `SendMail` n'a pas besoin de `ViewModel`.
L'application utilise une base de données composée d'une seule table qui contient des informations relatives aux destinataires de mails.

1.1 Les fichiers fournis

Pour faciliter le travail on vous fourni plusieurs fichiers :

- `activity_main.xml` - le fichier layout de l'activité `Main`,
- `activity_send_mail.xml` - le fichier layout de l'activité `Main`,
- `item_layout.xml` - le fichier layout pour un item de `RecuclerView` utilisé dans l'activité `Main`,
- `BDMails.kt` - le fichier contenant
  - la définition de l'unique table `Destinataire` de la base de données et
  - la définition de la base de données elle-même,
- `MonDao.kt` – le fichier contenant le `Dao`,
- `SendMail.kt` qui contient une fonction `sendMail` qui prépare un `Intent` pour activer l'envoi de mail.

Les captures d'écran en fin du sujet présentent les activités de l'application.

1.2 Activité `Main`

L'activité `Main` est composée d'un `RecyclerView`, de plusieurs `TextViews` et `EditTexts` et, en bas trois boutons : `Afficher Annotation`, `Ajouter Destinataire`, `Envoyer Mail`.
Le `RecyclerView` affichera les items de la table `Destinataire` de la BD, plus exactement on affichera pour chaque destinataire son nom, prénom, l'adresse mail et, en utilisant un `CheckBox` on indique la présence absence de l'annotation (`CheckBox` « checké » si l'annotation de destinataire est non null et différente de `String` vide `""`).

Activité `Main`

Exercice 1 :
Initialement il faudra alimenter la table `Destinataire`. Dans l'activité `Main` l'utilisateur remplit les champs (les `EditTexts`) en spécifiant le nom, prénom et l'adresse mail un destinataire. On peut éventuellement ajouter une annotation (c'est juste un texte quelconque). Les champs nom, prénom et mail sont obligatoires, l'annotation est facultative.
L'utilisateur fait in clic sur le bouton `Ajouter Destinataire` et le nouveau destinataire est ajouté dans la table de BD et il doit s'afficher immédiatement dans le `RecyclerView`. Pour l'instant il suffit afficher dans le `RecyclerView` juste le nom, prénom et le mail, on ne s'occupe pas de `CheckBox`. Implémentez le comportement décrit.
Strictement parlant on peut faire une implémentation qui n'utilise pas de `AndoridViewModel` mais le `AndroidViewModel` sera utile dans la suite, donc il est conseillé de l'utiliser de le début.

Exercice 2 :
Et maintenant il faut faire en sorte que le `CheckBox` dans chaque item de `RecyclerView` soit dans l'état « checké » (propriété `isChecked` avec la valeur `true`) si et seulement si le item (`Destinataire` correspondant) possède une annotation (c'est-à-dire l'annotation est non null et différente de `String` vide `""`).

Exercice 3 :

Ajoutez quelques destinataires, certains avec une annotation, d'autres sans.
Faire en sorte que les items sur les positions paires/implaires de `RecyclerView` s'affichent avec les couleurs de background respectivement

```
ColorUtils.setAlphaComponent(Color.CYAN, 50)
ColorUtils.setAlphaComponent(Color.GRAY, 50)
```

Exercice 4 :

Quand l'utilisateur fait un clic sur un item de `RecyclerView` cet item doit être sélectionné : son background prend la couleur

```
ColorUtils.setAlphaComponent(Color.RED, 120)
```

Quand l'utilisateur effectue un clic sur un item sélectionné de `RecyclerView` cet item devient de-sélectionné et sa couleur background revient à une de couleurs paire/impaire de l'exercice précédent.
Implémentez ce comportement.

Rappel 1. Soit `holder` une référence vers le `RecyclerView.Holder`.
`holder.absoluteAdapterPosition` donne la position de view correspondant dans le `RecyclerView` (les positions à partir de 0).

Rappel 2. Les items du `RecyclerView` possèdent à la racine un `CardView` et il est possible d'installer un `OnClickListener` sur un `CardView`. Il ne faut pas installer de listener sur le `CheckBox`, les `CheckBox`es affichent l'état fixe qui dépend de la présence ou absence d'une annotation. L'utilisateur n'interagit pas avec les `CheckBox`es.

Activité `SendMail`

Exercice 5 :

L'utilisateur sélectionne un ou plusieurs items de `RecyclerView` et effectue un clic sur le bouton `ENVOYER MAIL`. Cela fait démarrer l'activité `SendMail`. L'activité `SendMail` doit recevoir les adresses mails de tous les destinataires sélectionnés. L'activité `SendMail` affiche tous ces adresses mail dans un `TextView` (voir l'image en fin de sujet), les adresses sont séparées par des points-virgules.
L'utilisateur remplit les champs sujet et message et fait un clic sur le bouton `ENVOYER`.
Cela doit provoquer l'activation de l'application de l'envoi de mail. Il faut préparer un `Intent` (la fonction `sendMail() fourni sur moodle dans le fichier  \code{SendMail.kt` s'en chargera pour vous) et on démarre la nouvelle activité de façon habituelle.

Indication. Il est possible de passer un `StringArray` comme un extra d'un `Intent`.

Vous pouvez tester l'envoi de mail sur vos appareils à condition que vous avez au moins une application configurée pour l'envoi de mail. Sur les simulateurs par défaut il y a déjà des applications de mails mais elles ne sont pas configurées. Pas la peine de les configurer, vous allez implementer l'activité `SendMail` mais vous mettrez en commentaire l'envoi de l'Intent.

Retour au développement de l'activité `Main`

Exercice 6 :

Les items sélectionnés dans le `RecyclerView` se perdent quand l'utilisateur tourne l'appareil. Si vous travaillez avec un simulateur il semble que la possibilité de faire tourner l'appareil est désactivée par défaut. Donc pour observer le problème il faut aller dans `Settings` de l'appareil, choisir `Display` et activer `Auto-rotate screen`.
Faire en sorte que quand on tourne l'appareil les sélections dans `RecyclerView` soient préservées.

Exercice 7 :

Implementer le comportement suivant :

- s'il y a un seul item sélectionné dans le `RecyclerView` et
- cet item possède une annotation (ce qui correspond au `CheckBox` « checké »)

et quand l'utilisateur clique sur le bouton `AFFICHER ANNOTATION` l'annotation correspondante du destinataire s'affiche sur l'écran dans un dialogue, voir l'image à la fin du sujet.

Exercice 8 : Plus difficile

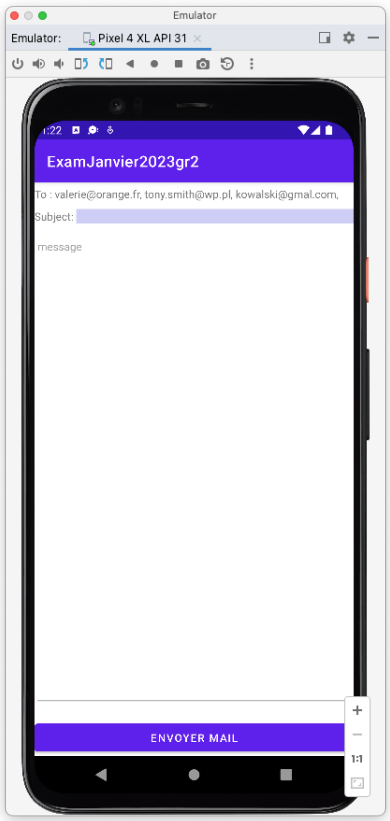
Faire en sorte que le bouton `ENVOYER MAIL` dans l'activité `MainActivity` soit désactivé (la propriété `isEnabled` `false`) quand aucun destinataire n'est sélectionné dans le `RecyclerView` .

Indication. Si on englobe la liste des items sélectionnés (ou la liste de positions de ces items) dans un `MutableLiveData` on peut installer un observeur sur cet objet. Cet observeur peut vérifier si la liste est vide ou non pour désactiver/activer le bouton. Le problème est que l'ajout/suppression d'un élément dans la liste qui est dans `MutableLiveData` ne déclenche pas l'observeur donc après chaque modification de la liste qui est à l'intérieur de `MutableLiveData` il faut exécuter

```
selected.setValue( selected.value )
```

où `selected` est l'objet `MutableLiveData` pour signaler la modification de données.

(a) L'activité MainActivity avec 4 items dans le RecyclerView.



(b) L'activité SendMail.

(c) L'activité MainActivuty avec l'annotation affichée dans un dialogue.