heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Widgets Android
Série d'*Exercices 1*

Fabien Dutoit

SYM – Systèmes mobiles

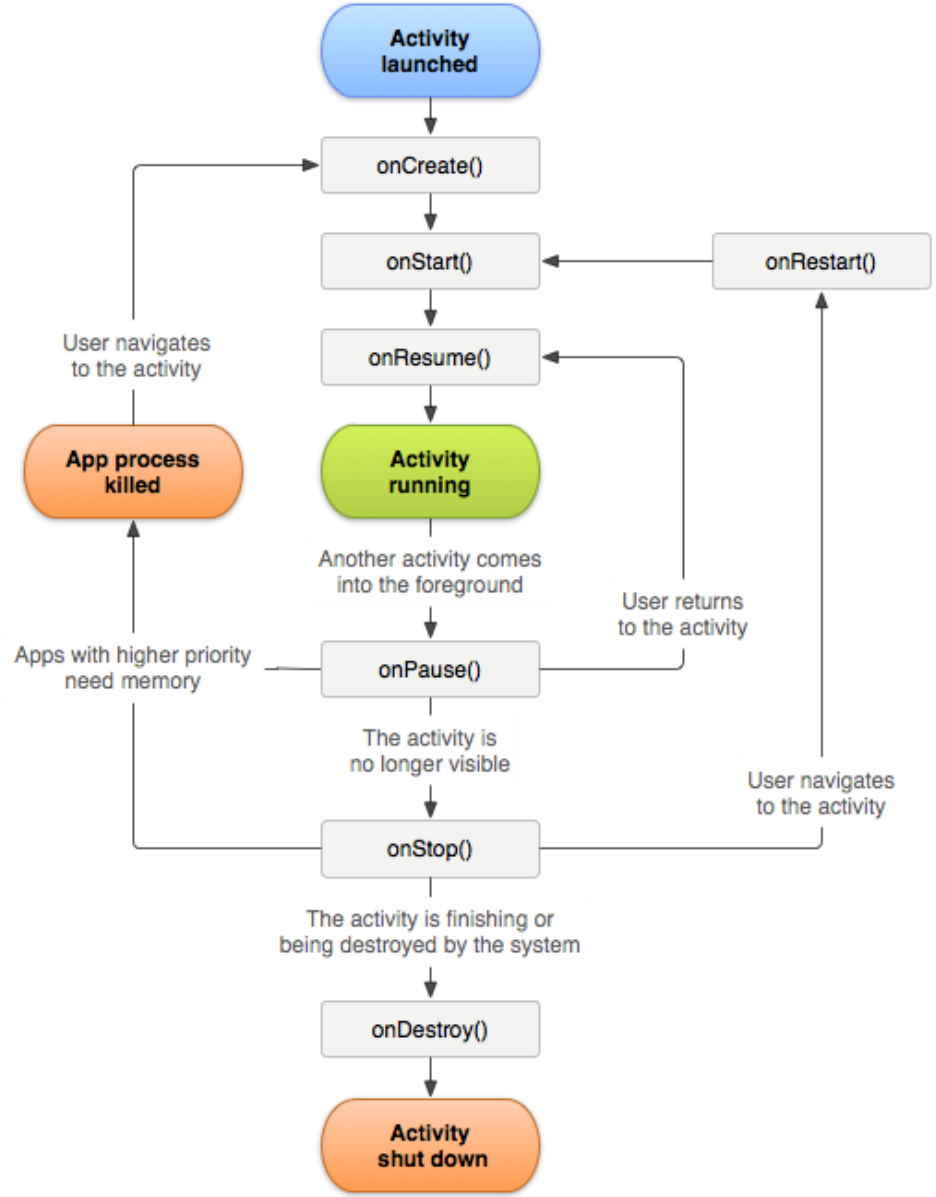
Explications

- *Ce chapitre va mettre en avant certaines particularités de la programmation Android ainsi que des exemples d'utilisation pour certains widgets et fonctionnalités*
- *Une application «bac à sable» vous est fournie, elle contiendra quelques exercices à réaliser durant ce chapitre*

Activité recréée

- *Dans certains cas, une activité peut être détruite automatiquement par le système:*
  - *Si le système a besoin de mémoire et que l'activité n'est plus visible (en arrière plan)*
  - *Lors d'un changement de configuration:*
    - *Rotation de l'écran*
    - *Changement de langue*
    - *…*
- *Il appartient au développeur de sauvegarder l'état de l'activité afin de pouvoir le restaurer ultérieurement*

Activité recréée



Activité recréée

- *Lorsqu'une activité est détruite de manière temporaire, le système fait appel à la méthode*
  `onSaveInstanceState(Bundle outState)`
  On sauvegarde dans le bundle les valeurs nécessaires pour recréer l'activité dans le même état

- *Ce bundle est redonné à la nouvelle instance de l'activité via les méthodes*
  `onCreate(Bundle savedState)` *et*
  `onRestoreInstanceState(Bundle savedState)`

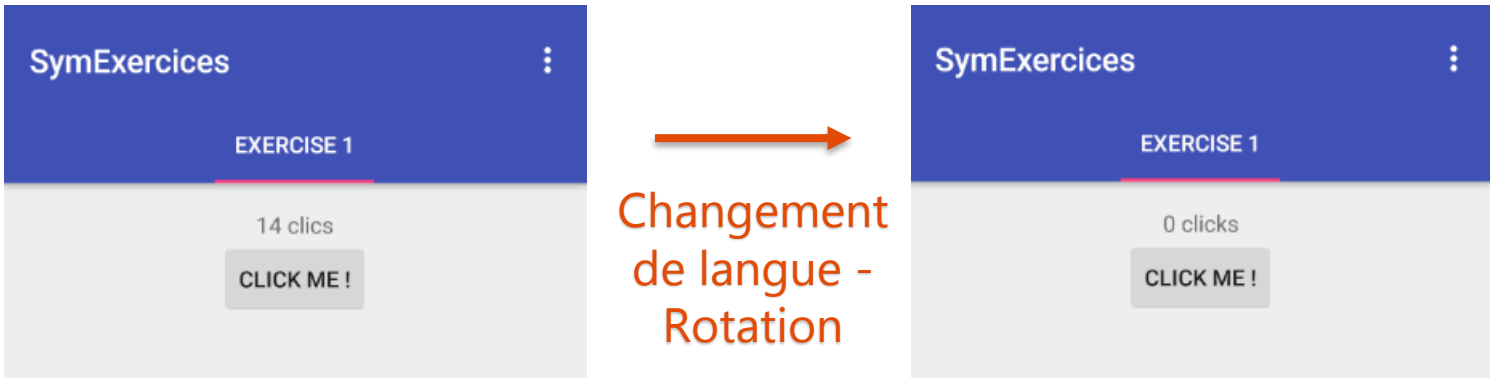
Activité recréée

- *Les fragments contenus dans une activité recréée sont logiquement aussi recréés*
- `onSaveInstanceState(Bundle outState)` *existe aussi pour les fragments*
- *Par analogie, le bundle est passé aux méthodes:*

```
onCreate(Bundle savedState)
onCreateView(…, Bundle savedState)
onViewStateRestored(Bundle savedState)
```

Exercice 1

- *Compteur avec un bouton:*



- *Modifier le fragment* `FragmentExercise1` *afin que la valeur du compteur persiste au changement de configuration*

Exercice 1 – Au passage

- *L'affichage de la valeur du compteur utilise une ressource plurals d'Android:*

```
<plurals name="click_counter">
    <item quantity="one">One click</item>
    <item quantity="other">%1$d clicks</item>
</plurals>
```

- `getQuantityString(R.plurals.click_counter, value, value)`

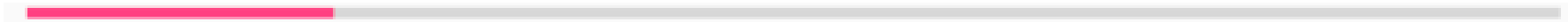
ressource id

quantité

valeur utilisée pour remplacer le %1$d

ProgressBar

- *Utilisé pour indiquer à l'utilisateur la progression d'une opération pas immédiate. Il existe deux fonctionnements différents:*
  - *Déterminé: la progression de l'opération est rendue visuellement*



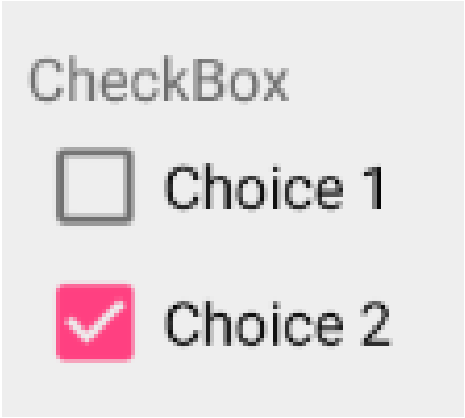
  - *Indéterminé: une animation de chargement est affichée*

SeekBar

- *Extension d'une ProgressBar, permettant une interaction de l'utilisateur. Utilisé par exemple pour permettre l'entrée d'une valeur numérique*

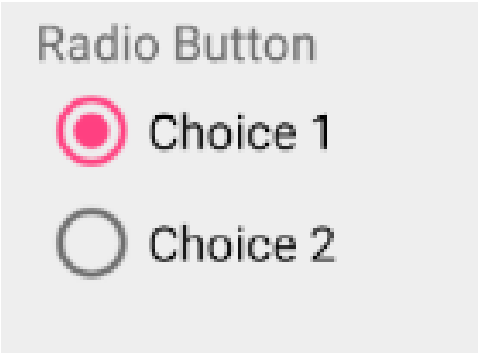
CheckBox

- *Case à cocher permettant la saisie d'un booléen*
- *En groupant plusieurs Checkboxs, on permet la saisie de 0…N réponses à une question à choix multiples*



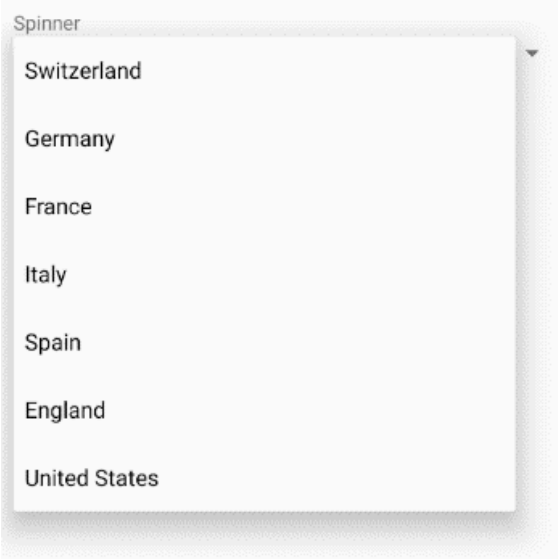
RadioBox

- *Similaire à une Checkbox*
- *Case à cocher permettant l'entrée d'un booléen*
- *En groupant plusieurs RadioBoxes, on permet une réponse unique à une question à choix multiples. Nécessite l'utilisation d'un RadioGroup*



Spinner

- *Un spinner est équivalent à un RadioGroup, mais les différents choix sont présentés dans une liste déroulante*



- *Les différents choix peuvent provenir d'une ressource de type string-array (contenu statique) ou d'un Adapter (contenu dynamique)*

```
<string-array name="country_arrays">
    <item>Switzerland</item>
    <item>Germany</item>
    <item>France</item>
    <item>Italy</item>
    <item>Spain</item>
    <item>England</item>
    <item>United States</item>
</string-array>
```

Exercice 2

A. ***ProgressBar*** *- On veut l'incrémenter d'une unité à chaque clic sur le bouton et la remettre à zéro sur un clic long*

B. ***SeekBar*** *- On veut mettre à jour la TextView* `sbText` *à chaque changement de valeur de la SeekBar*

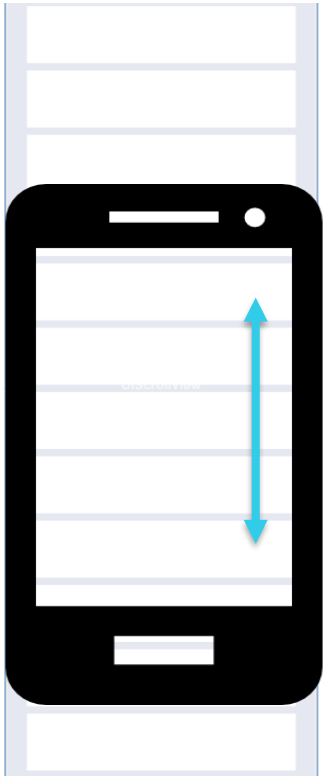
C. ***Checkbox*** *- Au clic sur le bouton Validate, on affichera un message, à l'aide de la fonction* `mListner.displaySnackBar()`*, indiquant si 0,* `1` *ou* `2` *choix ont été sélectionnés*

D. ***RadioBox*** *- on affichera un message à l'aide de la fonction* `mListner.displaySnackBar()` *lors d'un choix*

E. ***Spinner*** *- on affichera un message à l'aide de la fonction* `mListner.displaySnackBar()` *lors d'un choix – Vous noterez que l'événement est aussi lancé lors de l'initialisation, comment y remédier ?*

Exercice 2 – Au passage

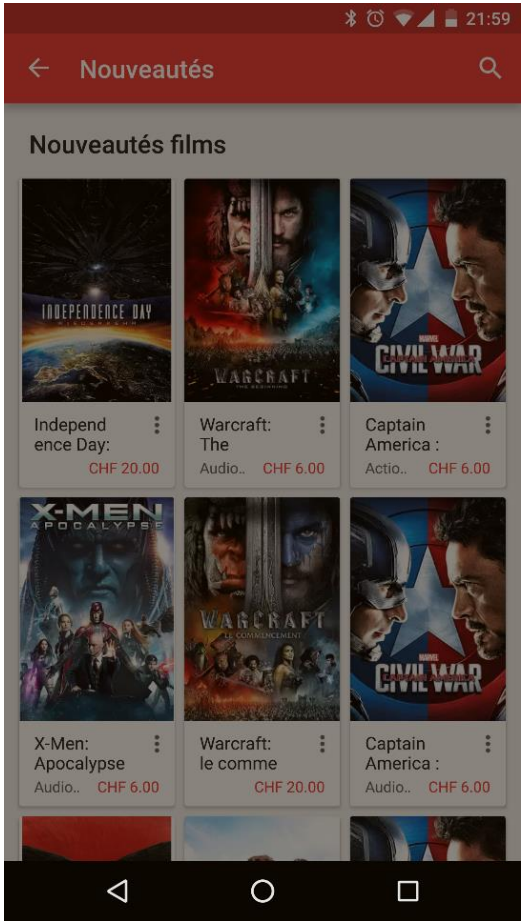
```
<RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <ScrollView
                android:layout_width="match_parent"
                android:layout_height="match_parent“ >

                <LinearLayout
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:orientation="vertical" >

                        <!-- contenu -->

                </LinearLayout>

        </ScrollView>

</RelativeLayout>
```



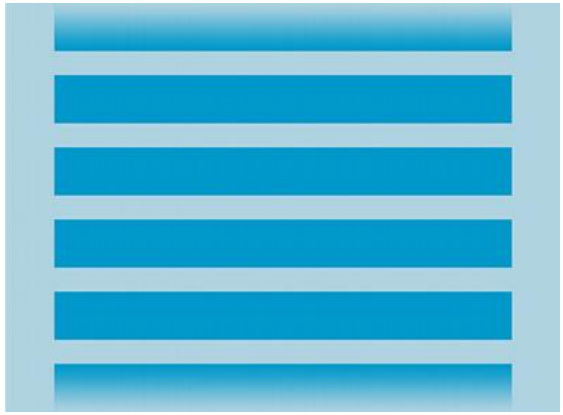
Exercice 2 – Au passage

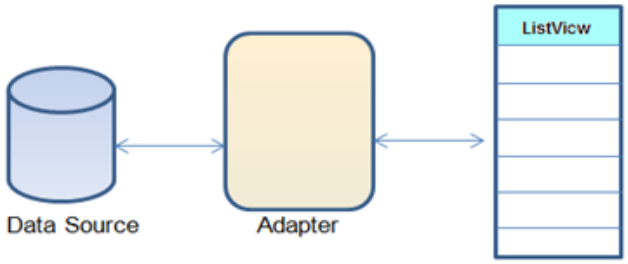
- *Attention à l'utilisation de ScrollView, l'utilisateur peut ne pas s'apercevoir qu'une partie du contenu est caché*
  - *Une solution est de faire en sorte d'avoir certains éléments visibles à moitié, pour inciter l'utilisateur à faire défiler la vue. Mais pas facile avec des tailles d'écrans variables*

- *Android propose l'événement clic long, qui peut aussi poser des problème d'usabilité. Sans indication, le risque est grand que l'utilisateur ne voit jamais cette fonction*



ListView





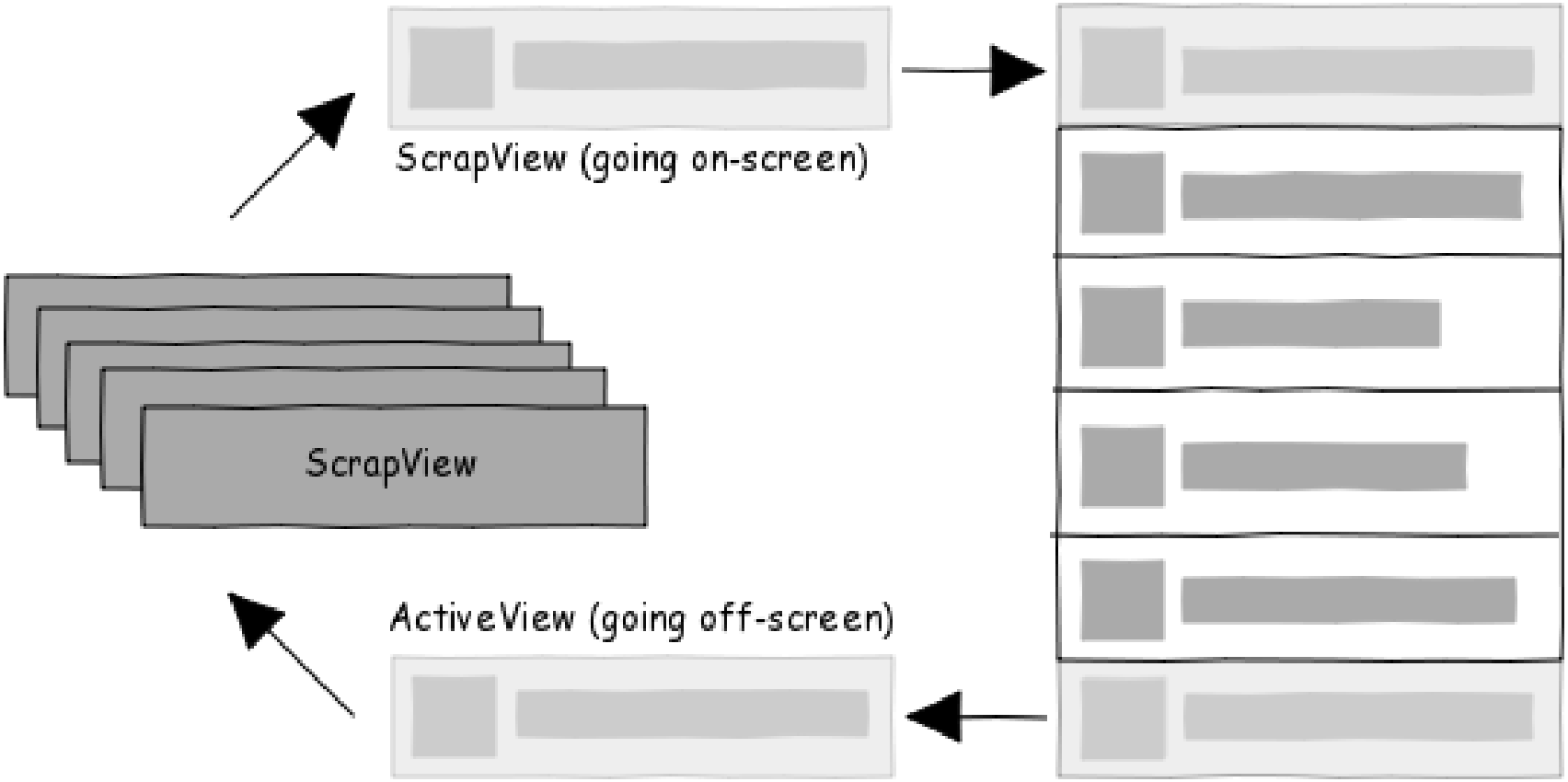
- *Une ListView permet l'affichage d'une liste défilante d'éléments*
- *Nécessite un Adapter faisant le lien avec les données, qui peuvent être sous différentes formes:*
  - *Tableau*
  - *Liste*
  - *Base de données*
- *L'utilisation d'une RecyclerView est généralement à privilégier à une ListView. Le fonctionnement étant similaire nous allons tout de même utiliser ici une ListView, plus simple à prendre en main.*

src: android.com, careerride.com

ListView

- *Adapter – API minimum*
  - **int getCount()**
    Indique le nombre d'élements dans la liste
  - **long getItemId(int position)**
    Retourne l'identifiant de l'élément à la position demandée
  - **Object getItem(int position)**
    Retourne l'élément à la position demandée
  - **View getView(int position, View rv, ViewGroup vg)**
    Retourne la vue représentant l'élément à la position demandée.
    Pratique le recyclage des vues !

ListView - Recyclage des vues



src: http://lucasr.org/

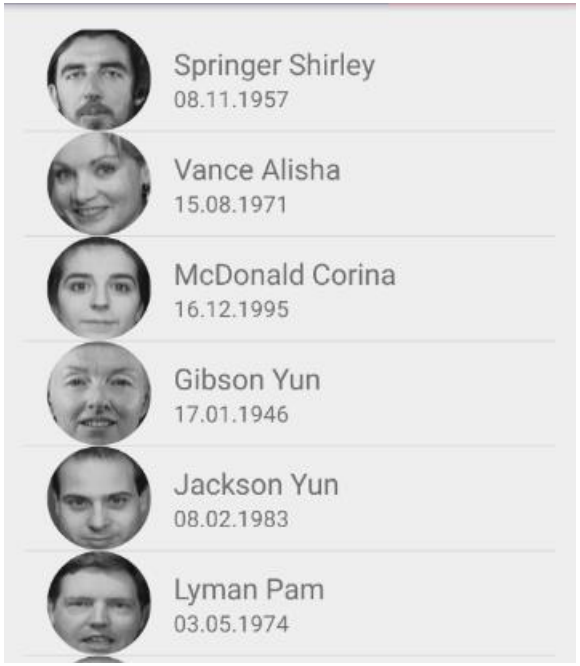
ListView

- *La méthode `notifyDataSetChanged()` du BaseAdapter permet d'informer la ListView que les données ont été modifiées*
  - *Provoque un rafraichissement de la vue*

- *Cette méthode doit être appelée dès qu'une donnée a été ajoutée, modifiée ou supprimée. Ainsi que si l'ordre des éléments de la liste a été modifié (option de tri)*

Exercice 3

- *Nous vous fournissons un exemple d'utilisation d'une ListView avec un l'affichage de nombres*
- *Vous devez modifier l'Adapter ainsi que le layout utilisé pour un item afin d'afficher des personnes issues d'un répertoire en vous inspirant de la capture d'écran*
- *Une classe Directory vous est fournie, le constructeur vous permettant de générer des données aléatoires*
- *Les images sont déjà présentes dans les ressources*



Exercice 3 – Au passage

- *Dans l'exemple, on affiche les images dans un cercle*
- *Cette fonctionnalité n'est pas disponible par défaut*
- *Utilisation d'une librairie: https://github.com/hdodenhof/CircleImageView*

Gradle

```
dependencies {
    ...
    implementation 'de.hdodenhof:circleimageview:2.2.0'
}
```

Usage

```
<de.hdodenhof.circleimageview.CircleImageView
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/profile_image"
    android:layout_width="96dp"
    android:layout_height="96dp"
    android:src="@drawable/profile"
    app:civ_border_width="2dp"
    app:civ_border_color="#FF000000"/>
```

Persistance

- *Si l'activité (le fragment) utilisé pour l'exercice 3 est recréé, le contenu de la ListView est perdu car tout était stocké en mémoire*
- *Contrairement à l'exercice 1, il ne s'agit pas ici de l'état de l'activité mais de données que l'on souhaiterait aussi retrouver après un prochain redémarrage de l'application*
- *Il faut donc utiliser la persistance: fichier local ou base de données*

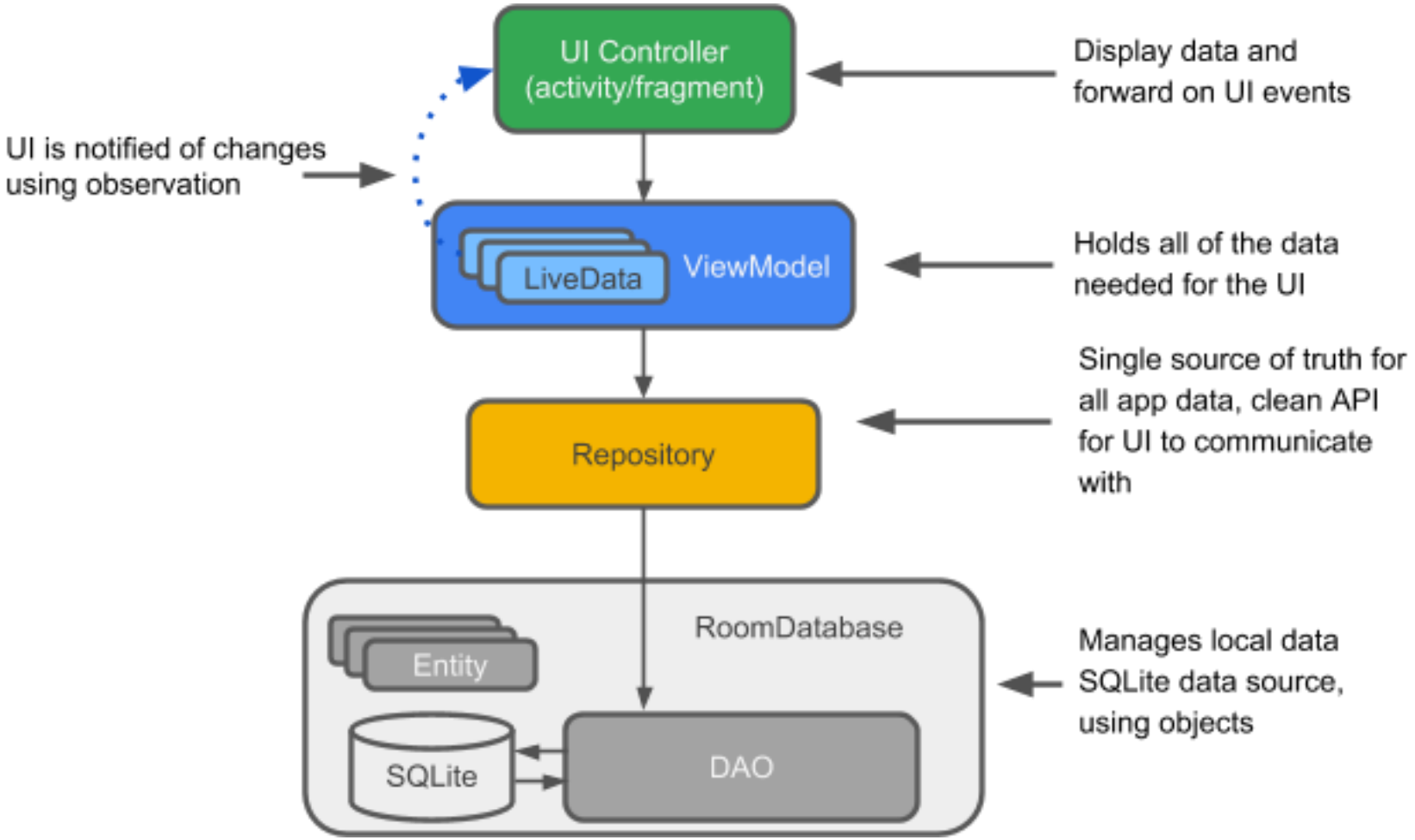
Persistance

- *L'utilisation d'un fichier local est possible, mais dans notre cas, cela nécessiterait de réaliser nous-même la sérialisation et la dé-sérialisation des données ainsi que de réécrire le fichier à chaque modification. Approche peu optimale…*
- *Utiliser une base de données locale (SQLite) permet d'accéder aléatoirement aux données à l'aide de requêtes SQL*
  - *Possibilité d'utiliser les clauses WHERE et ORDER BY*

Persistance - SQLite

- *Le SDK Android met à disposition une API permettant de créer et gérer une base de données locale SQLite - SQLiteOpenHelper https://developer.android.com/training/data-storage/sqlite*
- *Cela fonctionne très bien, mais peu aisé à prendre en main et nécessite d'écrire beaucoup de code, même pour une DB très simple*
- *Heureusement, depuis fin 2017 un ORM est mis à disposition: Android Room - https://developer.android.com/training/data-storage/room/ permettant de grandement simplifier le problème*
- *Utilisé en parallèle de la librairie LiveData, cela permet d'automatiser la propagation d'évènements*

Persistance – Android Room



Source: Google Codelabs

Exercice 3 - Avancé

- *Le but de cet exercice avancé est d'intégrer les librairies Room et LiveData à votre application et de les utiliser pour gérer la persistance du Directory et son affichage*

- *Vous pouvez suivre le codelabs suivant:*
  https://codelabs.developers.google.com/codelabs/android-persistence
  qui vous permettra une prise en main guidée de ces librairies

- *Une remarque: LiveData va retourner l'intégralité de la base de données dans une liste, à chaque changements. Ce n'est pas scalable !*
  Pour cela il existe la librairie Paging permettant de paginer les accès à la DB, malheureusement elle est uniquement compatible avec les RecyclerView. Un exemple:
  https://medium.com/@husayn.hakeem/android-by-example-googles-recent-android-paging-library-pokedex-d9ec1d4986e9