MVVM et LiveData

Le design pattern MVVM

Le desgin pattern MVVM, pour Model-View-ViewModel est un Pattern pour séparer plus distinctement les responsabilités dans le développement d'une application avec une interface graphique.

- Model : Logique du domaine, services, objets du modèle de donné. Dans notre projet, la classe `Todo`
- View : Création des éléments de l'interface graphique. Dans notre projet, les Fragments, aussi bien le template AXML que leur class de CodeBehind.
- ViewModel : Orgnisation du lien entre View et Model, expose des méthodes pour gérer les actions de l'utilisateurs ainsi que des données dynamiques auxquelles la View pourra se lier. Dans notre projet, des classes ViewModel que nous allons créer.

Les principaux intérêt du Pattern MVVM sont :

- La séparation des responsabilités
- Tester de façon unitaire la logique renfermée par ViewModel, comme il ne dépend pas de la View

MVVM et Android JetPack

Le framework Android fournit des outils afin d'implémenter facilement le pattern MVVM. Tout d'abord, une classe Abstraite `ViewModel` est fourni que nos ViewModels pourront étendre pour bénéficier de certaines fonctionnalités.

Ensuite, nous allons pour tirer parti des LiveData. Les LiveData sont des valeurs observables, c'est à dire qu'on peut s'abonner à leur changement. On peut créer des LiveData pour tout classe en utilisant un paramètre de type :

```
MutableLiveData<String> name = new MutableLiveDate<>();
```

Une LiveData possède une méthode `setValue` qui permet de mettre à jours la valeur, et une méthode `observe` qui prend en paramètre une lambda qui va permettre de réagir à la nouvelle valeur. Exemple :

```
name.observe(this, textView::setText);
```

Premier ViewModel

Nous allons donc procéder au refactoring de notre application en utilisant le Design Pattern MVVM. Créez un package `viewmodel` et dedans une nouvelles classe, `TodoListViewModel` qui étend la classe `ViewModel` :

```
public class TodoListViewModel extends AndroidViewModel{

}
```

Commençons par définir des propriété de données avec des LiveData :

```
private MutableLiveData<List<Todo>> todos = new MutableLiveData<>();
```

Ajoutons ensuite une méthode `init` qui va charger quelques Todos :

```
public void init() {
  todos.setValue(Arrays.asList(
    new Todo(UUID.randomUUID().toString(), "Test Todo Title 1", "Test Todo Description 1"),
    new Todo(UUID.randomUUID().toString(), "Test Todo Title 2", "Test Todo Description 2")
  ));
}
```

Nous pouvons aussi ajouter un gestionnaire d'événement qui ajoute un Todo :

```
public void addTodo(){
  List<Todo> todoList = todos.getValue();
  todoList.add(new Todo(UUID.randomUUID().toString(), "New Test Todo Title", "New Test Todo Description"));
  todos.setValue(todoList);
}
```

Nous devons maintenant lier notre ViewModel dans notre View. Commencez par supprimez l'initialisation de donnée, dont la resposabilité incombe au ViewModel. Créez également un `FloatingActionButton` et récupérez sa référence avec ButterKnife.

Ensuite créez un nouvel attributs de type `TodoListViewModel` dans votre classe, dans la méthode `onViewCreated`, récupérez le :

```
viewModel = ViewModelProviders.of(this).get(TodoListViewModel.class);
```

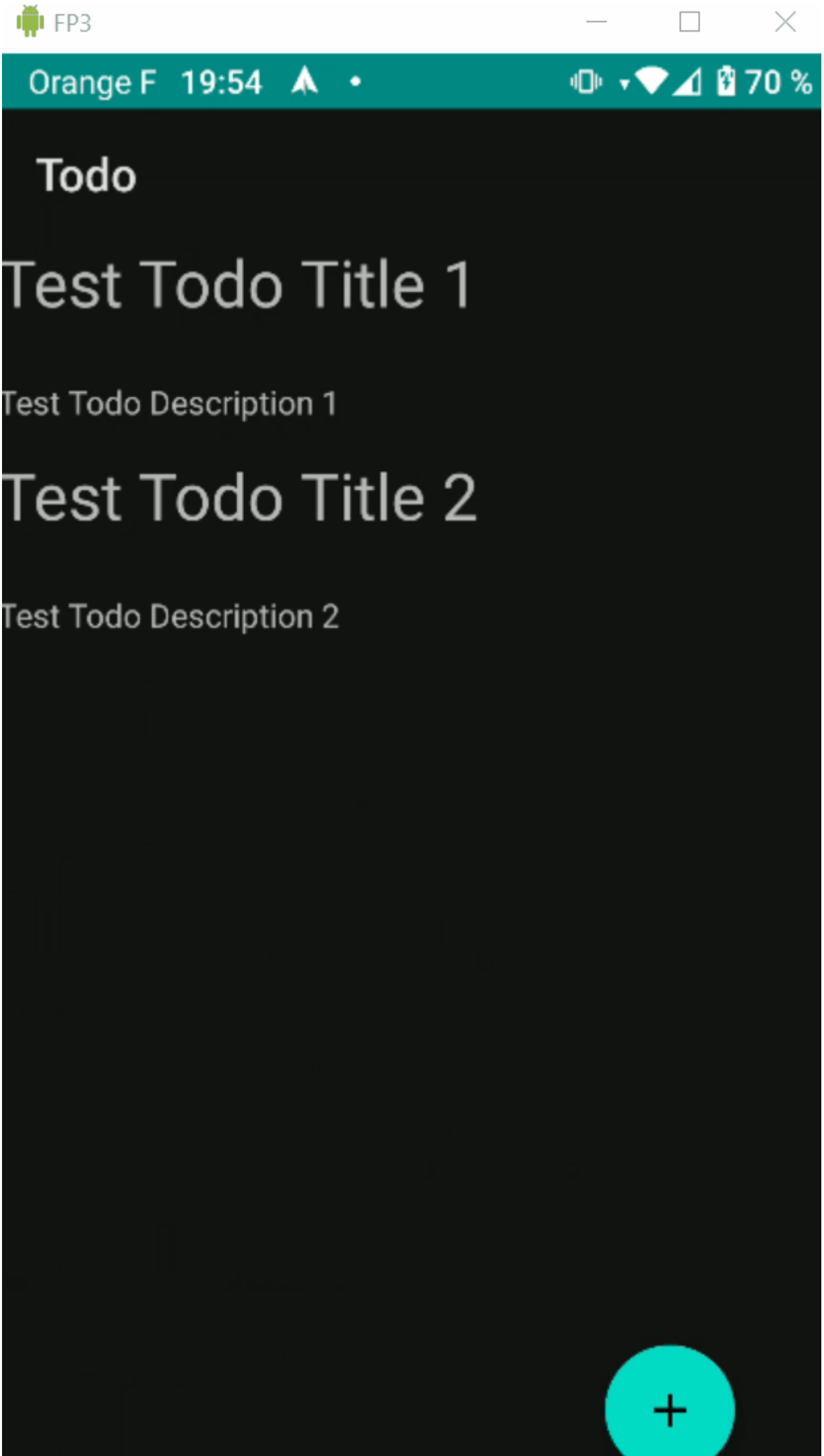
On peut maintenant appeler `init` pour créer les données de base, puis faire not liaisons de données et d'événements :

```
TodoListAdapter adapter = new TodoListAdapter(new ArrayList<>());

recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
recyclerView.setAdapter(adapter);

viewModel.getTodos().observe(this, adapter::setItems);
fab.setOnClickListener(v -> viewModel.addTodo());
```

On crée notre adapter avec une liste de vide et ensuite, on lie la LiveData de la TodoList au setters de l'adapteur afin de mettre à jours la valeur en l'adapteur à chaque fois que la valeur du LiveData est mise à jours.

Ensuite on ajoute tout simplement notre `addTodo` en gestionnaire d'événement `OnClick` du `FloatingActionButton`.

Voilà, notre application est refactorée avec le Pattern MVVM ! On peut observer le resultat suivant :


FP3
Orange F 19:54
70 %
Todo
Test Todo Title 1
Test Todo Description 1
Test Todo Title 2
Test Todo Description 2

Revision #7
Created 31 January 2021 14:43:40 by Arsène Lapostolet
Updated 21 May 2021 07:44:34 by Arsène Lapostolet