GHENT UNIVERSITY

Lab 4: Navigation with Fragments

Design and Development of Mobile Applications
E. De Coninck, S. Leroux, P. Simoens
2019-2020

In the previous session we made an application that shows a detail page for a single traffic notification from the traffic around Ghent. In this sesssion we will focus on managing the navigation from a simple fragment to this detail page. First we will need to refactor to fragments. The layout for this session is shown in Figure 1.

You will learn the following concepts:

- Using the Navigation Architecture Component `https://developer.android.com/topic/libraries/architecture/navigation`
- Create single Activity applications with fragments
- Provide multiple layout files for landscape and portrait
- Databinding with fragments

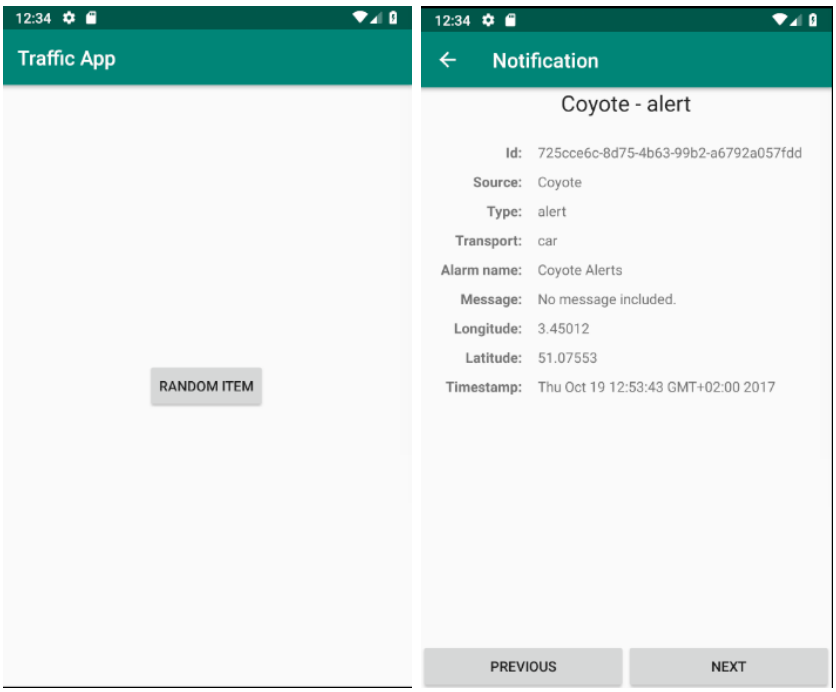

Figure 1: Example layout of the Traffic Notifications Application with fragments.

1 Using Navigation Architecture Component

In this session, we will setup our appication with Navigation Architecture Component to manage the navigation from one screen to the other, selecting a random item from the notification list.

1.1 Project set-up

- The start code project can be downloaded from Ufora. This start code is the solution of the last session.
- Add the dependencies for Navigation components as explained on this link: `https://developer.android.com/topic/libraries/architecture/adding-components#navigation`.

1.2 Provide a Navigation Graph

We will update the application according to the master detail flow within the MVVM structure. Therefor we need to refactor our code to use `Fragments` and provide a navigation graph resource file. The navigation graph requires a `NavHostFragment` inside the `Activity` and multiple `Fragments` to navigate between.

- Create a new resource file with the **Navigation** resource type.
- Open the resource file with the Navigation Editor. If it states that this is not available try syncing your project (File –> Sync with Gradle files)

1.2.1 Master Detail Flow

- Create two blank `Fragments` from the Navigation editor (do not include interface callbacks and factory methods).
  - MainFragment: a single button to select a random item. Use data binding to handle click events. The `Fragment` is responsible to handle the navigation with the Navigation Controller.
  - DetailFragment: same layout as in the **activity_main.xml** layout file.
  - Update `Fragments` to support databinding and set the correct variables (`https://developer.android.com/topic/libraries/data-binding/expressions#binding_data`). The repository is set when the `ViewModel` is used for the first time (`MainFragment`).
  - Connect the `Fragments` in the Navigation editor.
- Update `MainActivity` resource file to provide a `NavHostFragment` (can be found in the design editor) inside a `FrameLayout`. The Navigation Controller will use this navigation host to manage navigation.
- Bind the click event from the `MainFragment`'s button to show and navigate to a random traffic notification.
  - In the `ViewModel` provide a method to get a random item and a method to set a item. Make sure that the next() and previous() methods still function as designed.
  - get a random item from the view model and set it as the selected item when the button is pressed.
  - Use `Navigation.findNavController(...)` to navigate using the defined connection.

1.2.2 Master Detail in landscape

- Create a new layout resource file with the same name as the **activity_main.xml** file.
  - Add a orientation qualifier so the layout file will only be used in landscape.

- Add the Main and Detail Fragment to this layout in a horizontal `LinearLayout`. The left fragment should only be 1/3 of the screen.

- Handle the navigation in the click event of the button only when there is a navigation controller available.

1.2.3 Up navigation

- Setup `MainActivity` to handle up navigation corrrectly. In most cases up navigation's behaviour is the same as the back button. Look at `NavigationUI` to setup the action bar with the Navigation controller.