Navigation Architecture Component

Daniel Hartwich 🏞

Agenda

- Principles of Navigation
- Implement navigation using architecture component + Demo
- Migrating existing projects
- Implement support for new destinations
- Conditional navigation
- Global actions / common destinations

Principles of navigation

Principles of navigation

"The app should have a fixed starting destination"

Principles of navigation

"A stack is used to represent the "navigation state" of an app"

Principles of navigation

"The Up button never exits your app"

Principles of navigation

"Up and Back are equivalent within your app's task"

Principles of navigation

"Deep linking to a destination or navigating to the same destination should yield the same stack"

Implement navigation using architecture component

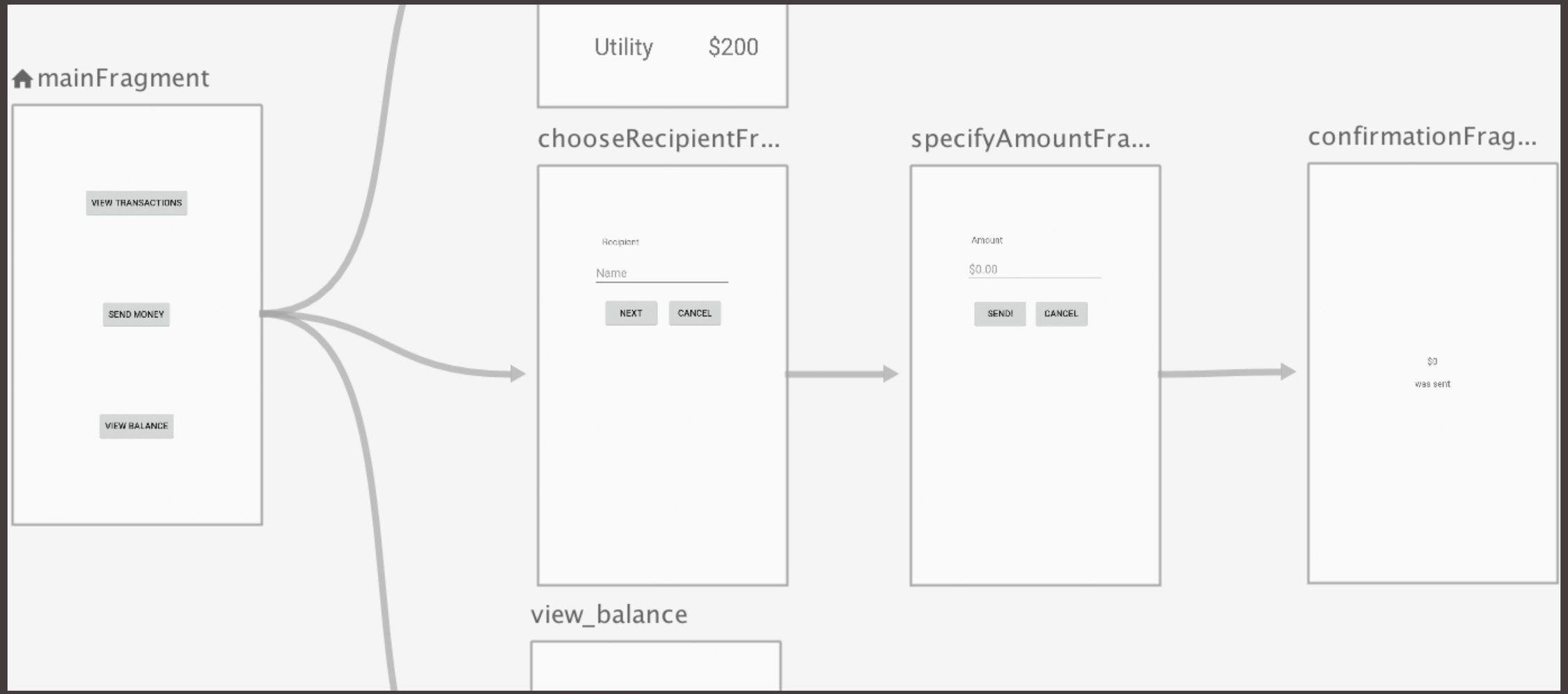
Implement navigation using architecture component

- Simplify navigation between ***destinations*** of your app
- Set of ***destinations*** compose app's ***navigation graph***
- A ***destination*** is any place you can navigate to in your app

Implement navigation using architecture component

- Destinations are usually ***fragments***
  - Activities
  - different navigation graphs / subgraphs
  - Custom destination types
- Connections between different destinations are called **actions**

Example Navigation Graph



Demo

Set up navigation component in Android Studio

Android Studio

https://developer.android.com/topic/libraries/architecture/navigation/navigation-implementing

Add the dependencies to the navigation component

```
implementation 'android.arch.navigation:navigation-fragment:1.0.0-alpha05'
implementation 'android.arch.navigation:navigation-ui:1.0.0-alpha05'
implementation 'android.arch.navigation:navigation-fragment-ktx:1.0.0-alpha05'
implementation 'android.arch.navigation:navigation-ui-ktx:1.0.0-alpha05'
```

Declare an activity as the Navigation Host

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/my_nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        app:navGraph="@navigation/mobile_navigation"
        app:defaultNavHost="true"
        />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Create blank destination

- In the "Design" Tab of the navigation resource. Click on 'Create blank destination'

Create links between destinations

- Create actions by dragging connections between two destinations

Use Actions to navigate between two destinations

- Create actions by dragging connections between two destinations

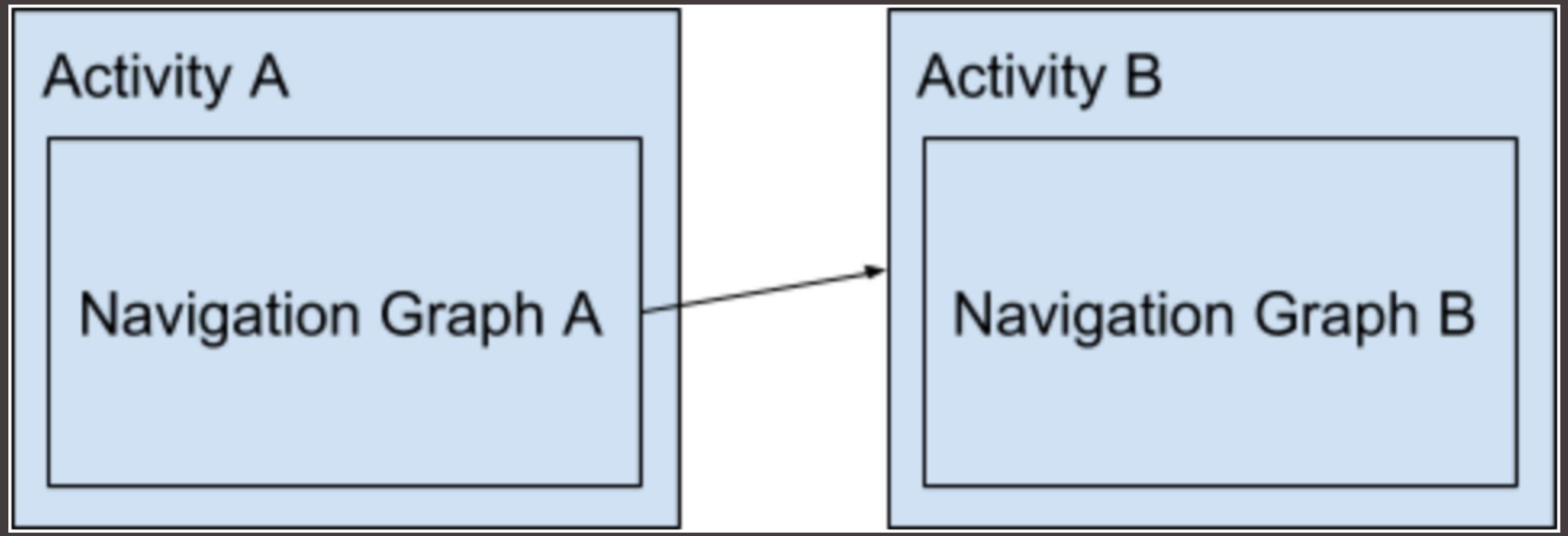
Call the actions to execute navigation

```
view.seeArticlesButton.setOnClickListener {
            view.findNavController().navigate(R.id.action_lastPurchasesFragment_to_articleDetailFragment)
}
```

or

```
view.seeArticlesButton.setOnClickListener(Navigation.createNavigateOnClickListener(R.id.overviewFragment))
```

Migrating existing projects



Migrating existing projects

- `NavController` + navigation graph are contained within a single activity
- When migrating, migrate one activity at a time
- To migrate create a navigation graph for all destinations inside an activity

Migrating existing projects

- Add activity destinations in the navigation graph
- Replace existing usages of `startActivity()`
- Ideally, migrate activities to fragments
- --> Single Activity Applications

Implement support for new destinations

Implement support for new destinations

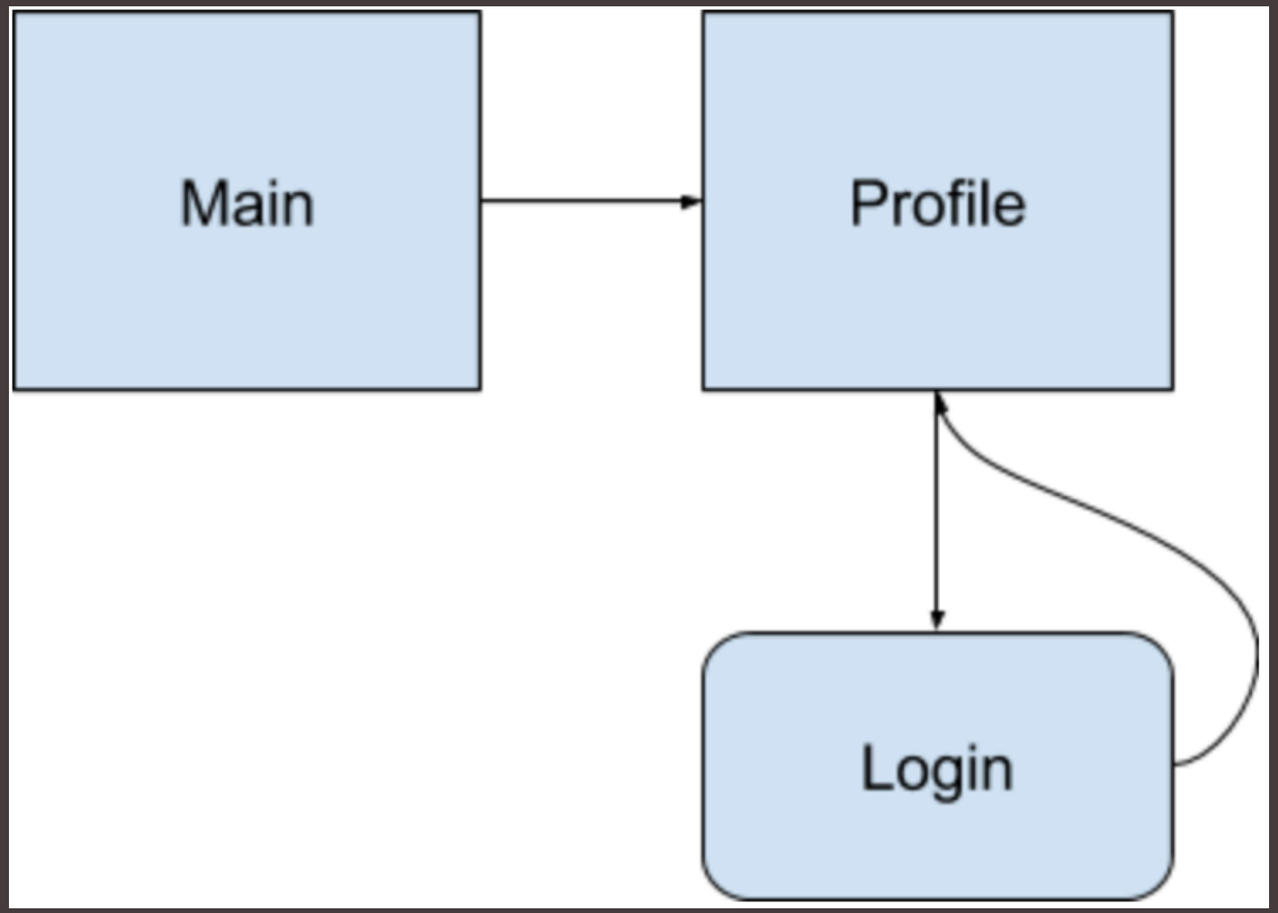
- `NavControllers` rely on one or more `Navigator` objects to perform navigation.
- By default only Activities and Fragments are supported
- Subclass `Navigator` to implement navigation to your custom destination
- Use `getNavigatorProvider()` to add your custom Navigator using `addNavigator()`

Implement support for new destinations

```
val customNavigator = CustomNavigator()
navController.navigatorProvider += customNavigator
```

- Most `Navigator` classes have a nested destination subclass
- It's used to specify additional attributes unique to the custom destination

Conditional navigation



Conditional navigation

- Destinations that can only be reached under certain conditions should still be an own destination
- e.g. Profile screen behind a Log In mechanic
- Login is presented after the user tries to access 'Profile' without log in
- After login is completed we call `popBackStack()` to dismiss the login screen and show the profile again.

Global actions / common destinations

- An action that is accessible from multiple screens
- Example: A 'cancel' button in a multi step process that should go back to the first screen
- Right Click a destination and add a global action.
- This action is now accessible from everywhere and can be called the same way as other actions.

Navigating to a global action

```
viewTransactionButton.setOnClickListener { view ->
    view.findNavController().navigate(R.id.action_global_mainFragment)
}
```

Additional notes

- Unit Tests not possible, without further abstraction
- Compatible with known architecture patterns
- It's possible to navigate between two modules through `<include>` in the navigation graph

Additional notes

- Support for navigation drawer and overflow menu possible through matching IDs (fragment + menu.xml)

Set up the navigation view with the NavController:

```
val navigationView = findViewById<NavigationView>(R.id.nav_view)
navigationView.setupWithNavController(navController)
```

Sources

- Official documentation
- Medium article about multi module navigation
- dhartwich1991/navigation-component
- googlesamples/android-sunflower
- NavController documentation

Questions?

Thank you! 🙏 💚