

Mobile Application Development

Produced
by

David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
<http://www.wit.ie>

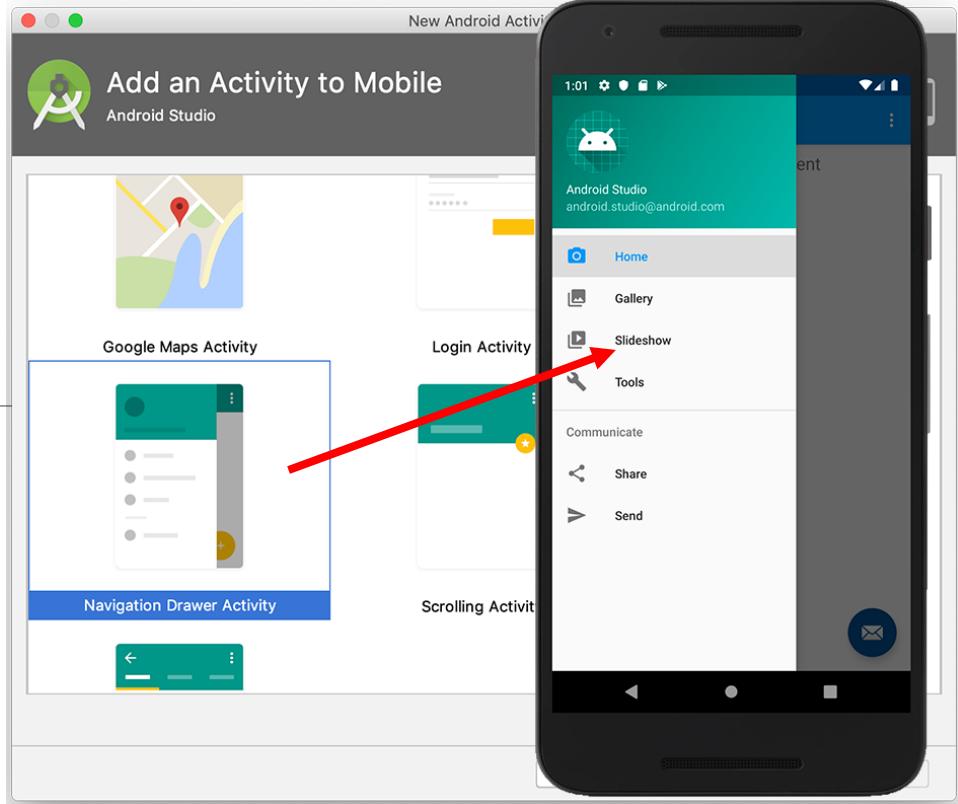


Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



DonationX

Walkthrough





Agenda

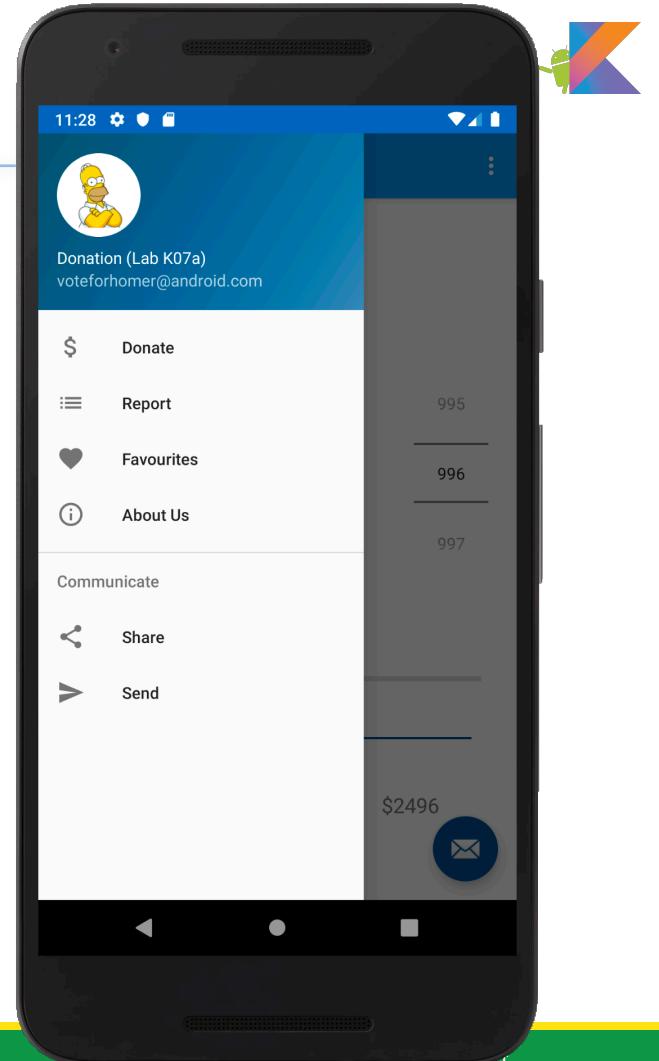
- ❑ Navigation Component usage in DonationX
- ❑ Initial Setup Look
- ❑ Code Extracts from DonationX

Case Study

❑ An Android App to keep track of donations made to ‘Homers Presidential Campaign ’.

❑ App Features

- Accept donation via number picker or typed amount
- Keep a running total of donations
- Display report on donation amounts and types
- Display running total on progress bar

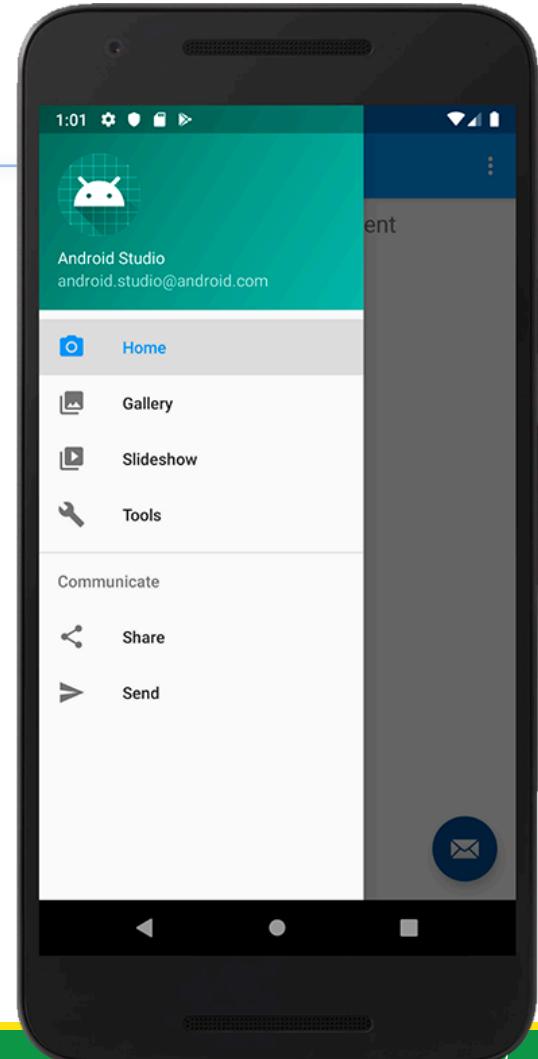
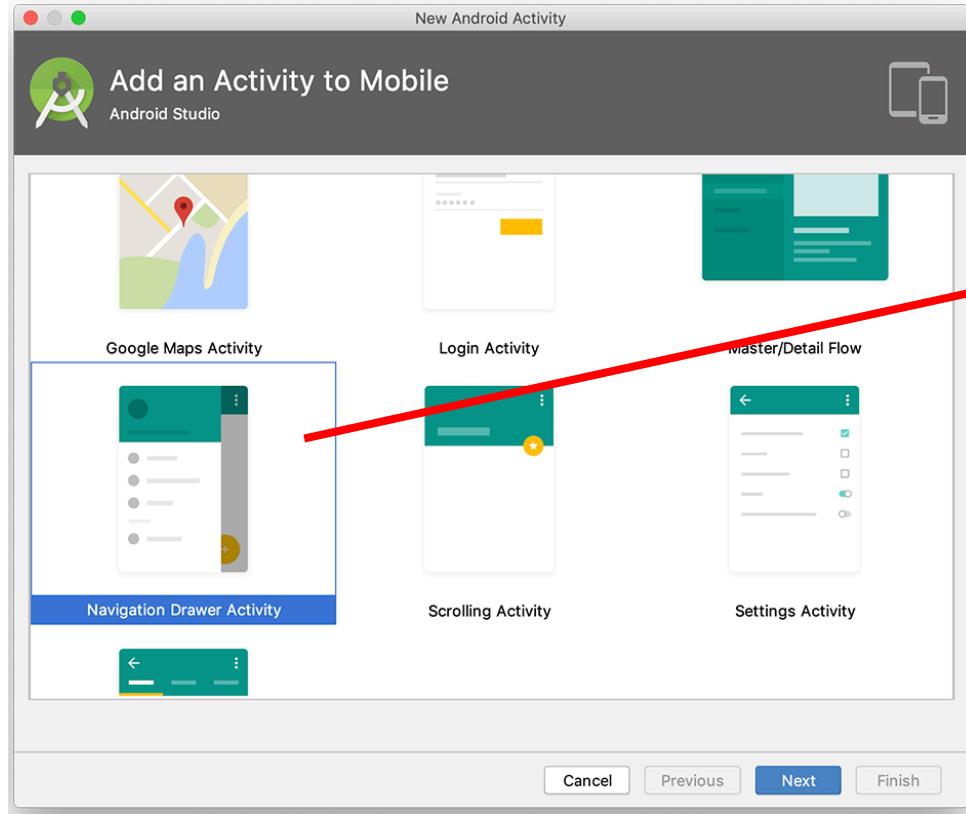


Nav Drawer in DonationX

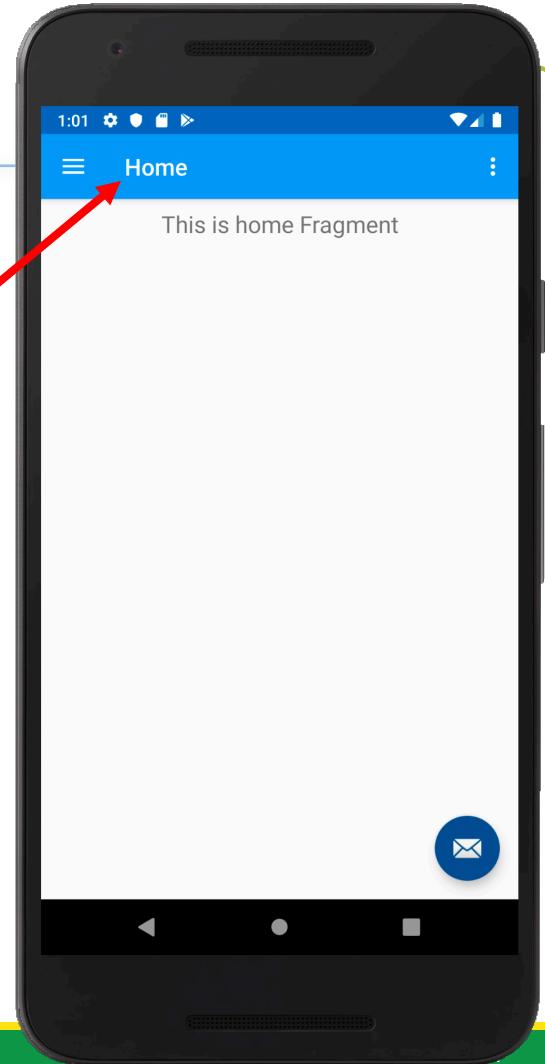
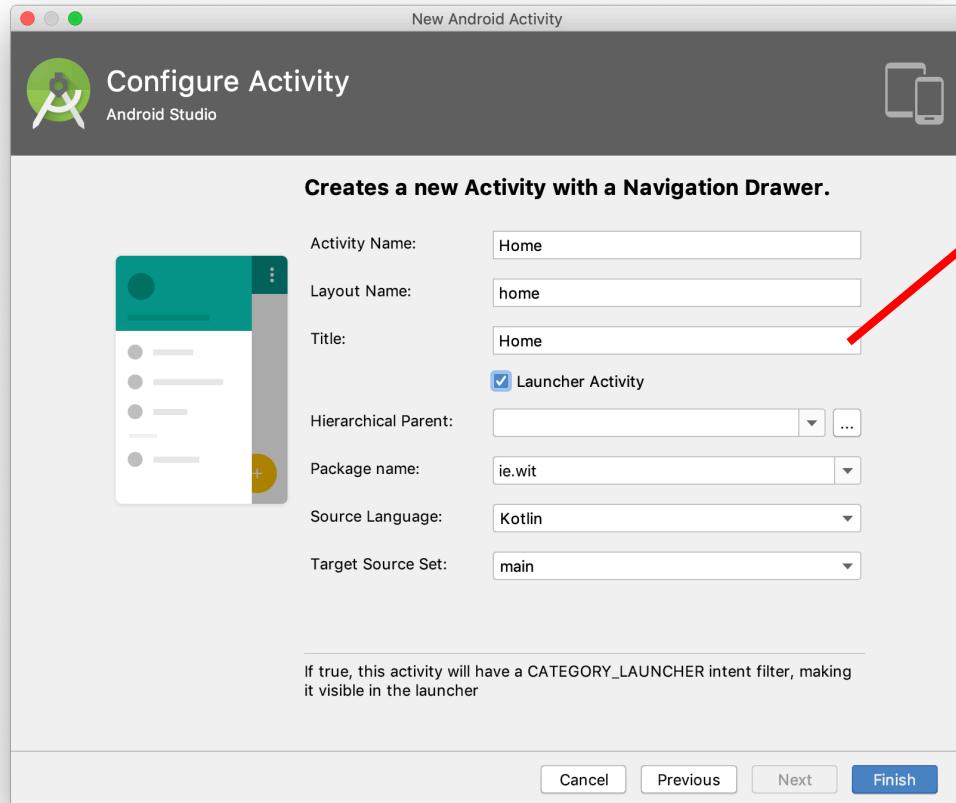
Auto Setup (via Navigation Component)



Initial Setup



Initial Setup



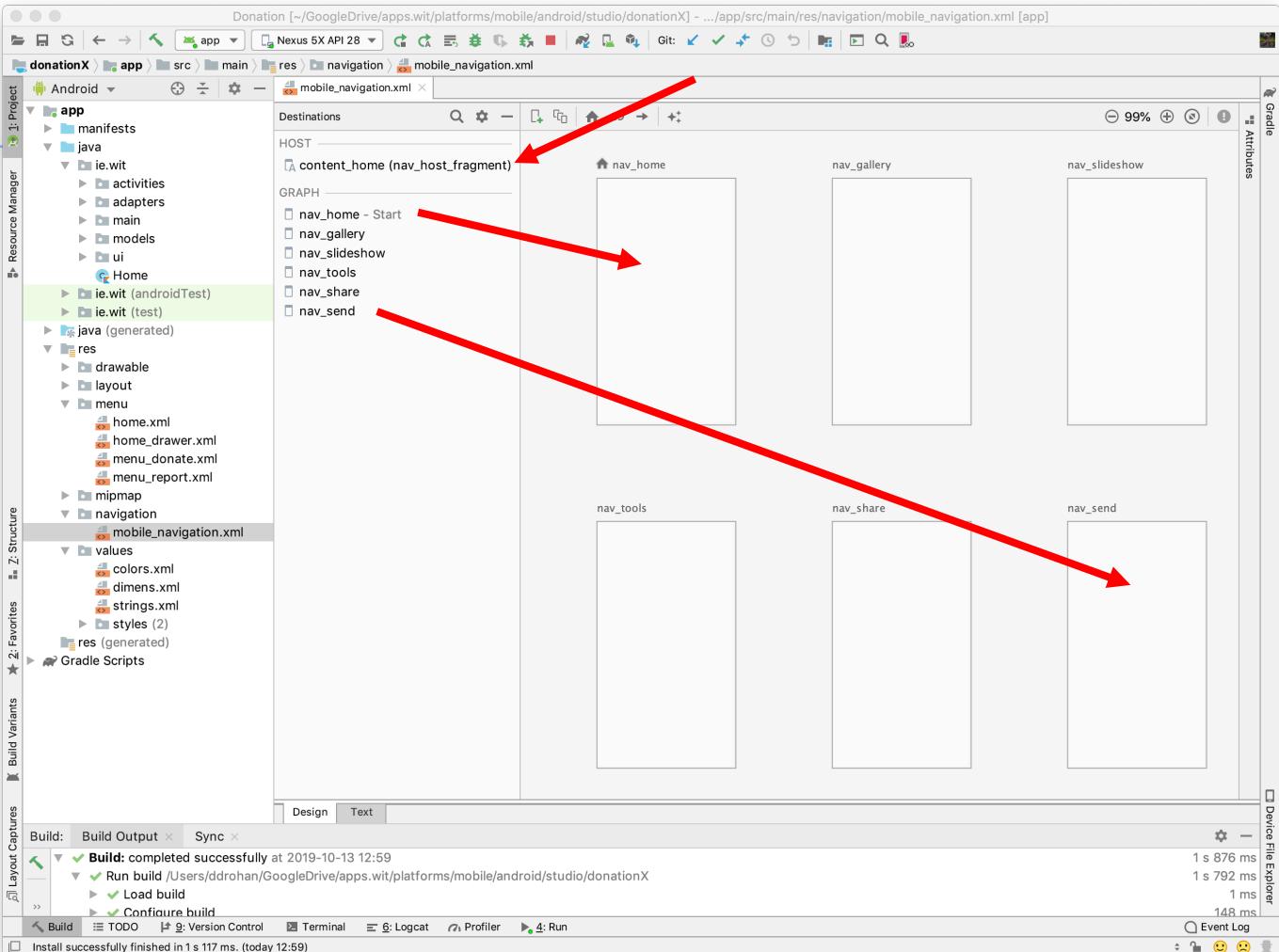


What Do You Get?

(errors in strings.xml & dimens.xml for a start ☺)

☐ Navigation Graph

- Navigation destinations & actions
(none yet)



☐ Navigation Graph

- Navigation destinations & actions (none yet)
- Fragment Setup

```
<?xml version="1.0" encoding="utf-8"?>
<navigation 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:id="@+id/mobile_navigation"
    app:startDestination="@+id/nav_home">

    <fragment
        android:id="@+id/nav_home"
        android:name="ie.wit.ui.home.HomeFragment"
        android:label="Home"
        tools:layout="@layout/fragment_home" />

    <fragment
        android:id="@+id/nav_gallery"
        android:name="ie.wit.ui.gallery.GalleryFragment"
        android:label="Gallery"
        tools:layout="@layout/fragment_gallery" />

    <fragment
        android:id="@+id/nav_slideshow"
        android:name="ie.wit.ui.slideshow.SlideshowFragment"
        android:label="Slideshow"
        tools:layout="@layout/fragment_slideshow" />

    <fragment
        android:id="@+id/nav_tools"
        android:name="ie.wit.ui.tools.ToolsFragment"
        android:label="Tools"
        tools:layout="@layout/fragment_tools" />

    <fragment>
```

The screenshot shows the Android Studio interface with the project navigation graph open. The left sidebar displays the project structure, with the 'navigation' folder under 'res' highlighted. The main editor area shows the XML code for the navigation graph. Five fragment definitions are listed, each with its ID, name, label, and layout. Three red arrows point from the text labels 'nav_home', 'nav_gallery', and 'nav_slideshow' to their respective XML elements. The bottom status bar indicates a successful build.

Various Classes

■ Home

The screenshot shows the Android Studio interface with the project 'donationX' open. The left sidebar displays the project structure under the 'app' module, specifically the 'ie.wit' package which contains various fragments and view models. A red box highlights this package structure. A red arrow points from the bottom right towards the 'appBarConfiguration' code in the 'Home.kt' file. The code itself is as follows:

```
package ie.wit

import ...

class Home : AppCompatActivity() {

    private lateinit var appBarConfiguration: AppBarConfiguration

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.home)
        val toolbar: Toolbar = findViewById(R.id.toolbar)
        setSupportActionBar(toolbar)

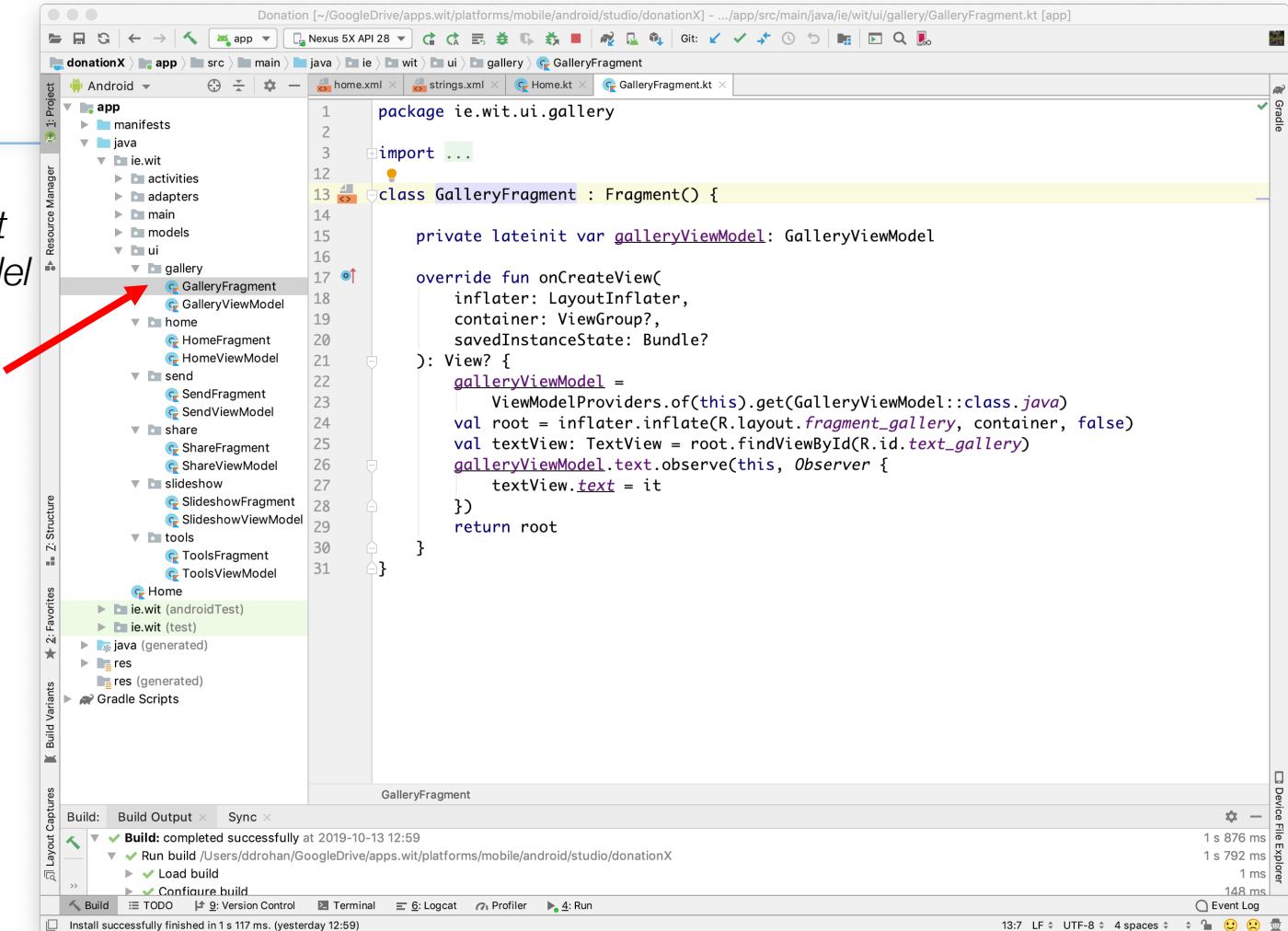
        val fab: FloatingActionButton = findViewById(R.id.fab)
        fab.setOnClickListener { view ->
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show()
        }
        val drawerLayout: DrawerLayout = findViewById(R.id.drawer_layout)
        val navView: NavigationView = findViewById(R.id.nav_view)
        val navController = findNavController(R.id.nav_host_fragment)
        // Passing each menu ID as a set of IDs because each
        // menu should be considered as top level destinations.
        appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.nav_home, R.id.nav_gallery, R.id.nav_slideshow,
                R.id.nav_tools, R.id.nav_share, R.id.nav_send
            ), drawerLayout
        )
        setupActionBarWithNavController(navController, appBarConfiguration)
        navView.setupWithNavController(navController)
    }
}
```

The bottom status bar indicates a successful build: 'Build completed successfully at 2019-10-13 12:59'. The bottom right corner shows the page number '11'.

Various Classes

■ Gallery

GalleryFragment
GalleryViewModel



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure under the "app" module. A red arrow points from the "GalleryFragment" file in the "ui/gallery" package towards the code editor.
- Code Editor:** The main window displays the `GalleryFragment.kt` file. The code defines a Fragment named `GalleryFragment` that uses `GalleryViewModel` via `lateinit`. It overrides `onCreateView` to inflate a layout and observe a `text` observer on the `textView`.
- Build Output:** The bottom-left pane shows the build log, indicating a successful build completed at 2019-10-13 12:59.
- Event Log:** The bottom-right pane shows the event log with several entries related to the build process.

```
package ie.wit.ui.gallery

import ...

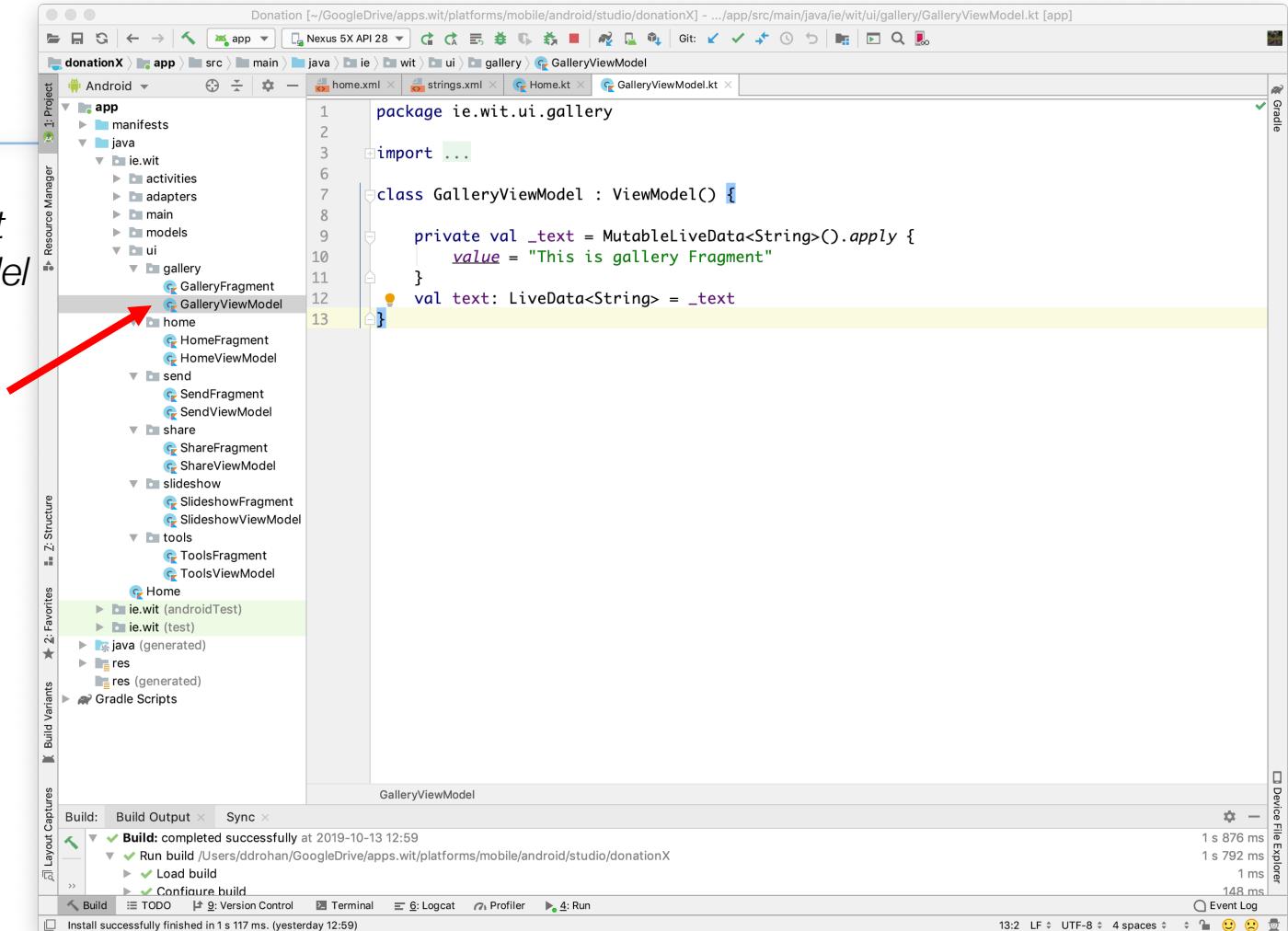
class GalleryFragment : Fragment() {

    private lateinit var galleryViewModel: GalleryViewModel

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        galleryViewModel =
            ViewModelProviders.of(this).get(GalleryViewModel::class.java)
        val root = inflater.inflate(R.layout.fragment_gallery, container, false)
        val textView: TextView = root.findViewById(R.id.text_gallery)
        galleryViewModel.text.observe(this, Observer {
            textView.text = it
        })
        return root
    }
}
```

Various Classes

- **Gallery**
GalleryFragment
GalleryViewModel
- Ditto for the rest

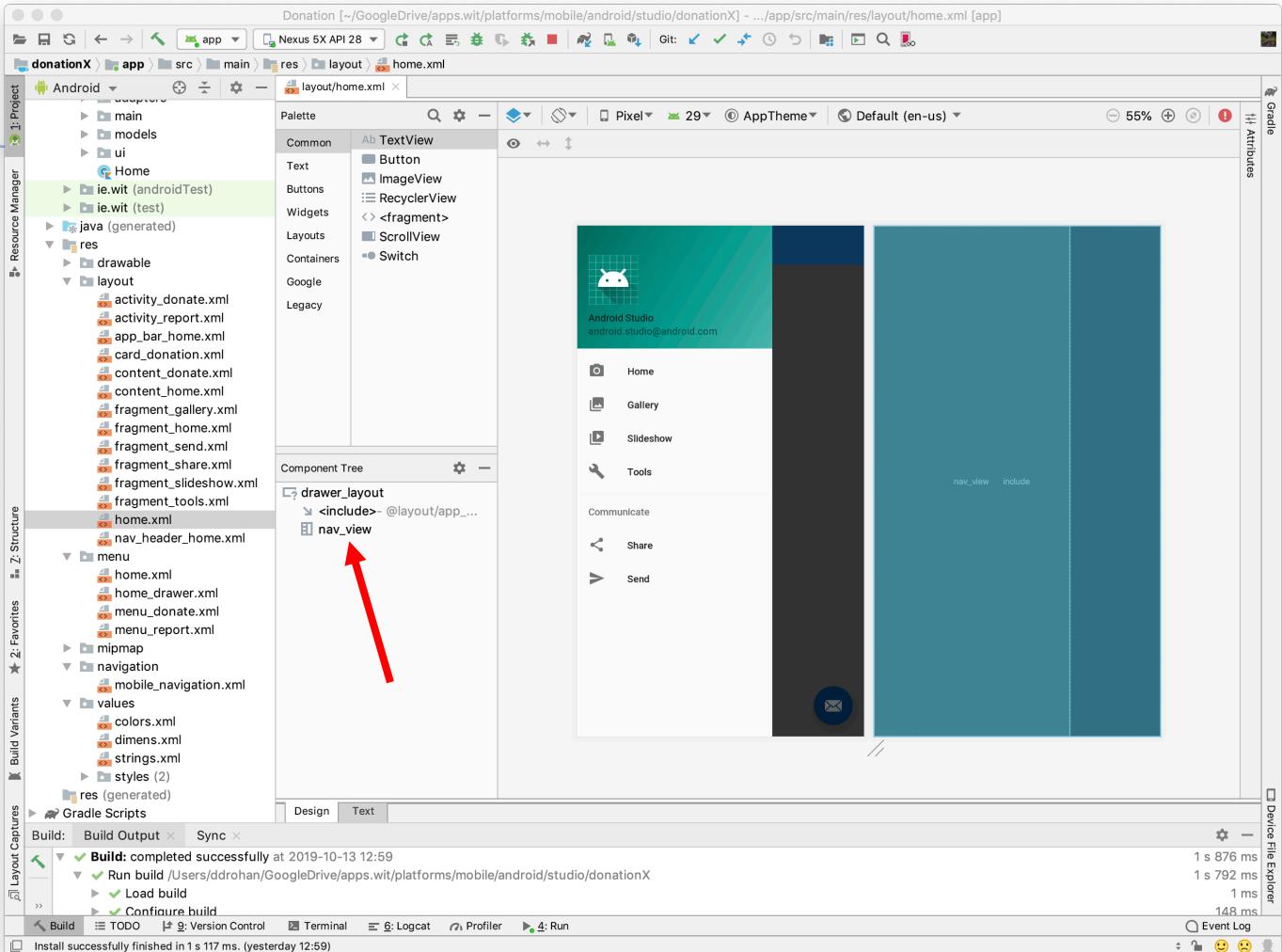


The screenshot shows the Android Studio interface with the following details:

- Project Tree:** The "app" module is selected. The "src/main/java/ie/wit/ui/gallery" package contains the `GalleryFragment` and `GalleryViewModel` files. A red arrow points from the text "Ditto for the rest" to the `GalleryViewModel` entry in the tree.
- Code Editor:** The `GalleryViewModel.kt` file is open. It defines a `GalleryViewModel` class that extends `ViewModel`. It contains a private `MutableLiveData<String>` named `_text` with a value of "This is gallery Fragment". A `LiveData<String>` named `text` is also defined, which observes changes in `_text`.
- Build Output:** The build was completed successfully at 2019-10-13 12:59. The log shows the build process: Run build, Load build, Configure build.
- Bottom Bar:** Shows tabs for Build, TODO, Version Control, Terminal, Logcat, Profiler, and Run. A message indicates the install was successful.
- Right Panel:** Device File Explorer and Event Log are visible.

Various Layouts

■ home.xml



Various Layouts

■ home.xml

The screenshot shows the Android Studio interface with the project navigation bar at the top. The main area displays the XML code for `home.xml`. Several red arrows point to specific parts of the code, highlighting the structure of the DrawerLayout and its components.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout 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:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <include
        layout="@layout/app_bar_home"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

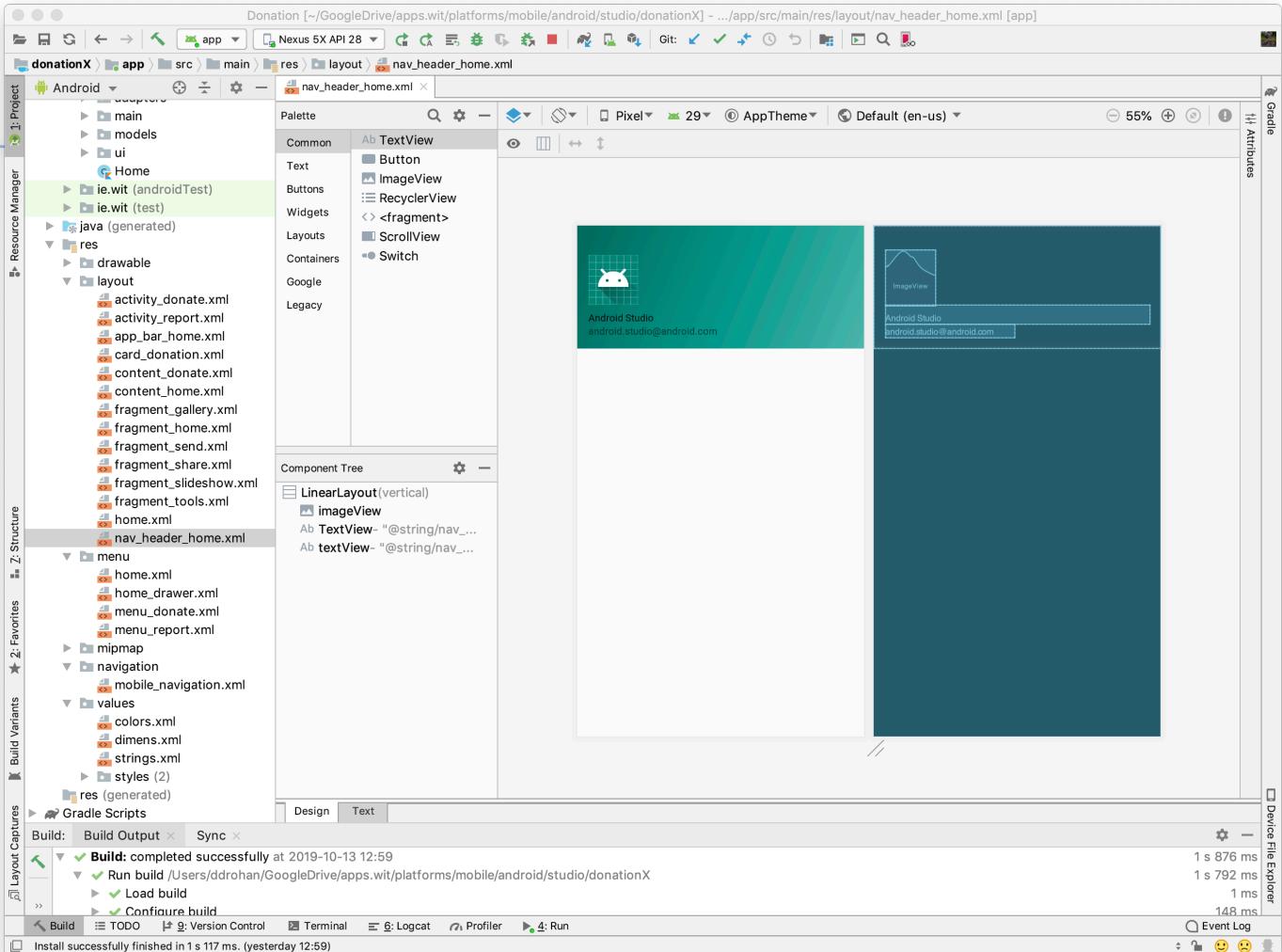
    <com.google.android.material.navigation.NavigationView
        android:id="@+id/nav_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:fitsSystemWindows="true"
        app:headerLayout="@layout/nav_header_home"
        app:menu="@menu/home_drawer" />

</androidx.drawerlayout.widget.DrawerLayout>
```

The code defines a `DrawerLayout` with an `include` tag for the `app_bar_home` layout. It also contains a `NavigationView` component with attributes like `headerLayout` and `menu`.

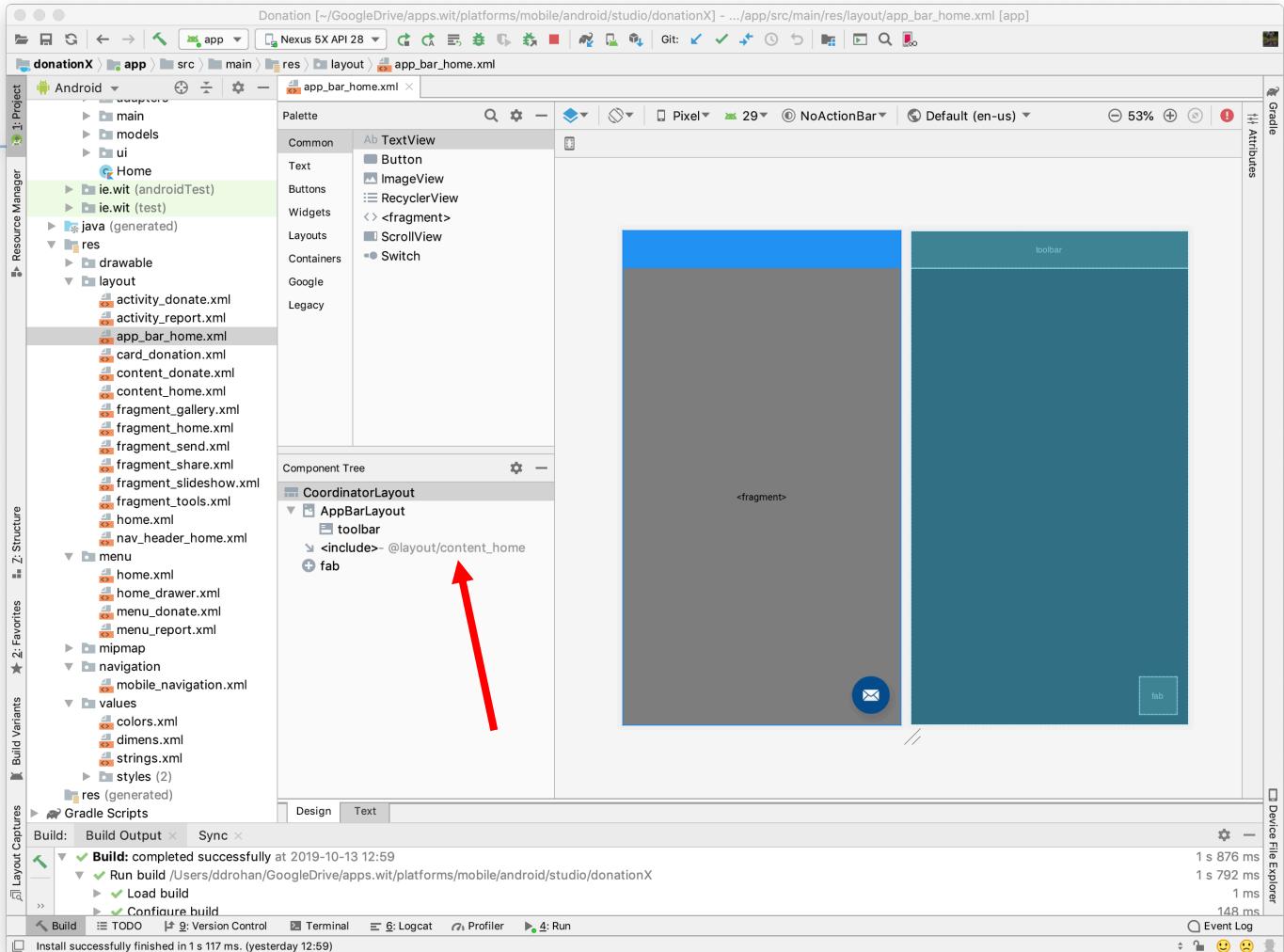
❑ Various Layouts

- nav_header_home.xml



❑ Various Layouts

- app_bar_home.xml



❑ Various Layouts

- **app_bar_home.xml**

```
1: Project 2: Resource Manager 3: 4: 5: 6: 7: 8: 9: 10: 11: 12: 13: 14: 15: 16: 17: 18: 19: 20: 21: 22: 23: 24: 25: 26: 27: 28: 29: 30: 31: 32: 33:
```

```
    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=".Home">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_home" />

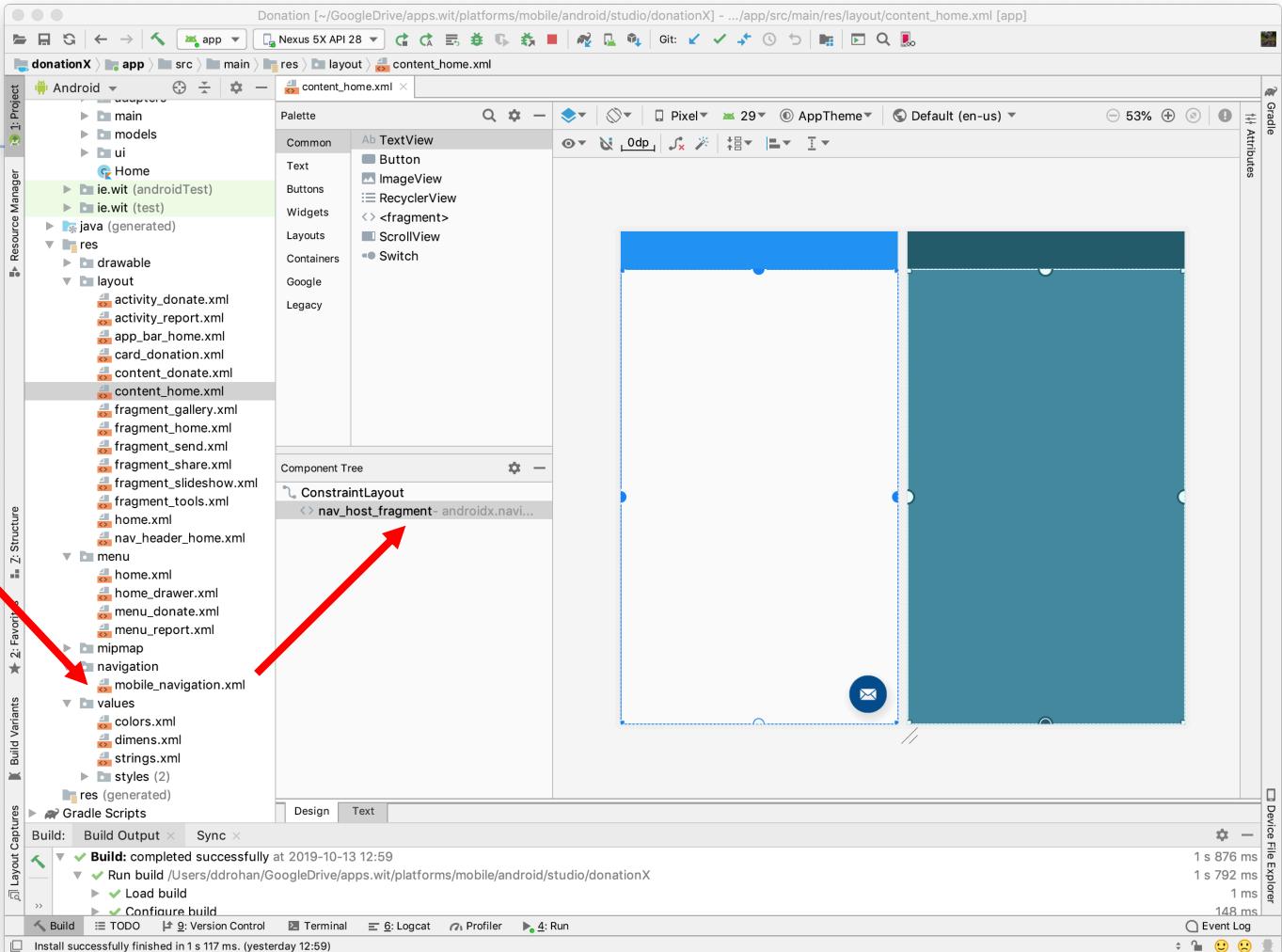
    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        app:srcCompat="@android:drawable/ic_dialog_email" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

The screenshot shows the Android Studio interface with the project structure on the left and the XML code for `app_bar_home.xml` on the right. The code defines an `AppBarLayout` containing a `Toolbar`, followed by an `<include>` tag pointing to `content_home`. A red arrow highlights this `<include>` tag. The bottom status bar indicates a successful build.

❑ Various Layouts

- `content_home.xml`
- ‘links up’ with navigation graph



❑ Various Layouts

- `content_home.xml`
- ‘links up’ with navigation graph

The screenshot shows the Android Studio interface with the project `donationX` open. The `content_home.xml` file is selected in the layout editor. A red arrow points from the text "links up" in the list above to the `app:navGraph` attribute in the XML code. Another red arrow points from the text "navigation graph" in the list above to the `mobile_navigation.xml` file listed in the Project Structure tab.

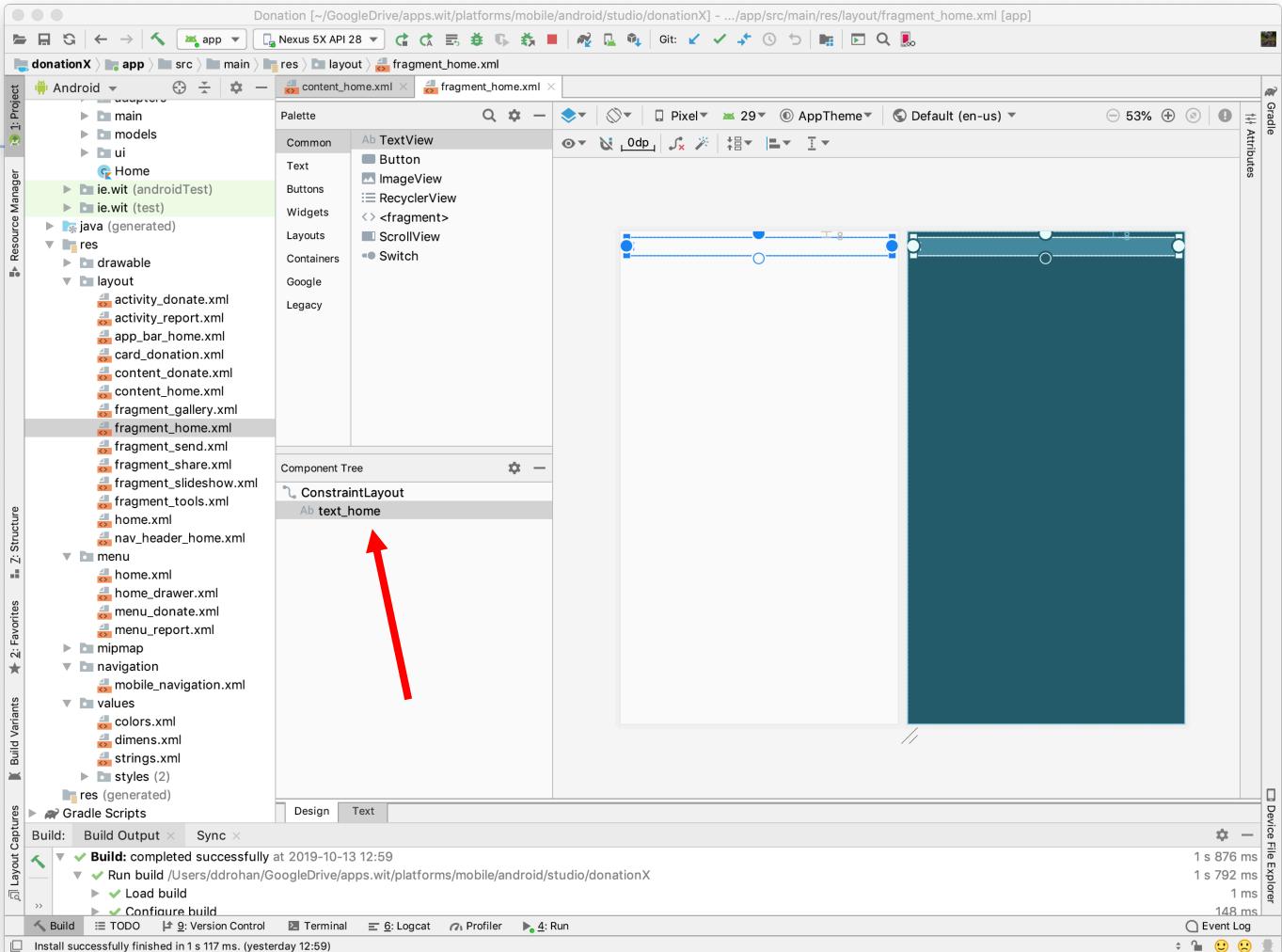
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="com.google.android.material.appbar.AppBarLayout$ScrollingViewBehavior"
    tools:showIn="@layout/app_bar_home">

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

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

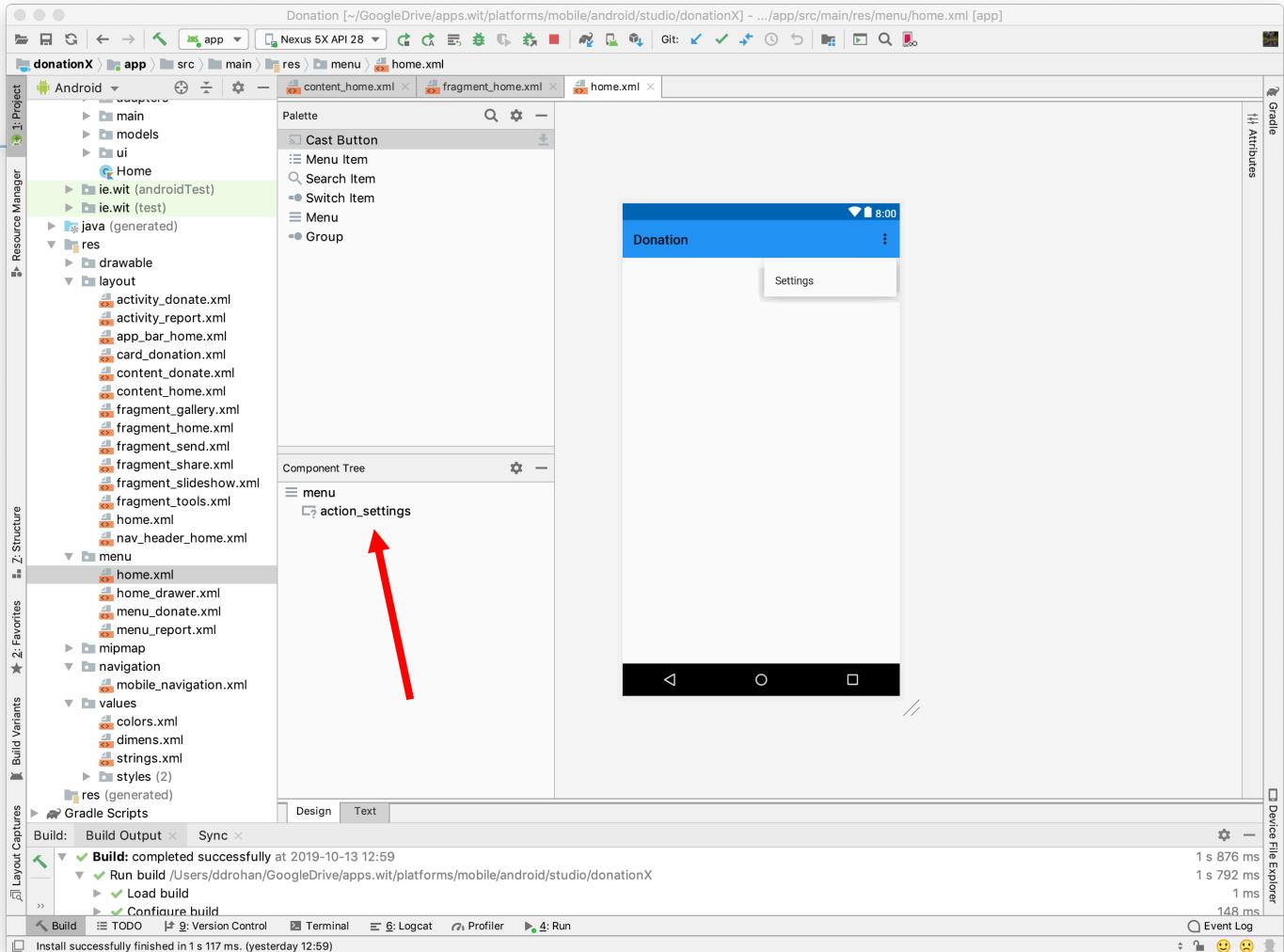
❑ Various Layouts

- `fragment_home.xml`
- Ditto for *gallery*, *slideshow*, *tools*, *share*, *send*



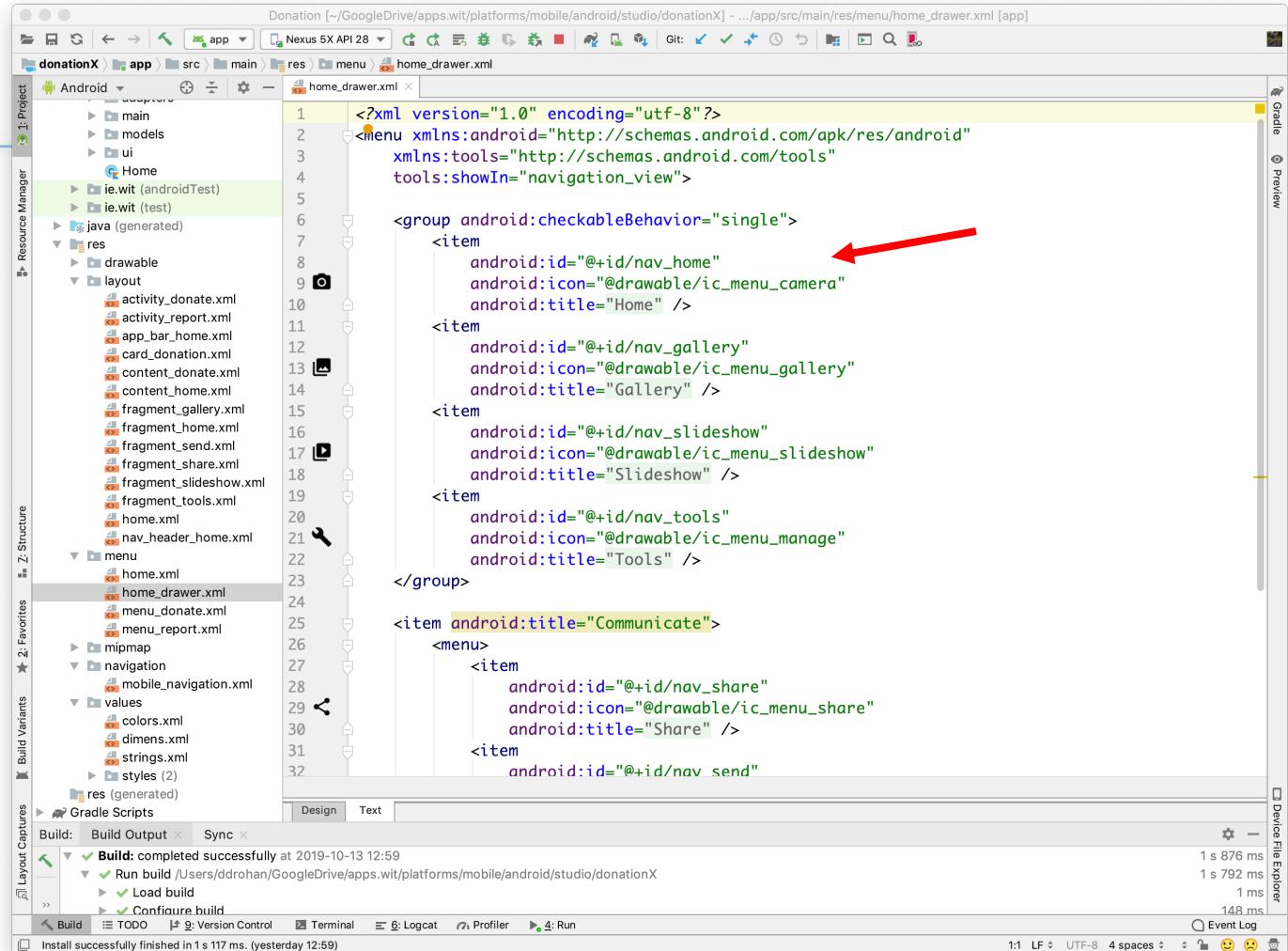
Various Menus

■ home.xml



Various Menus

■ home_drawer.xml



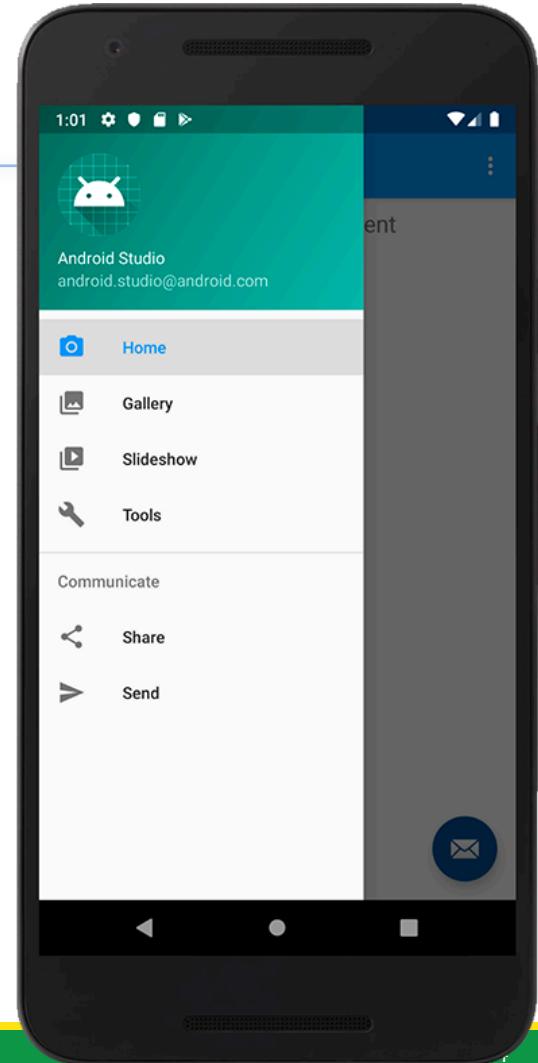
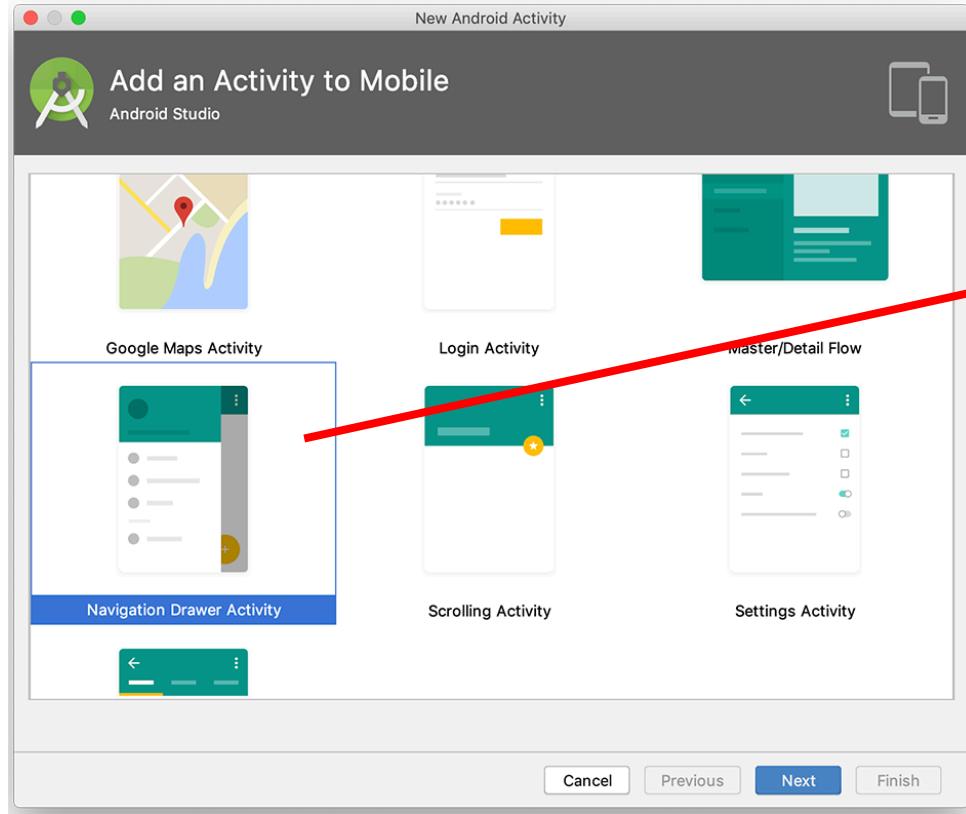
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">

<group android:checkableBehavior="single">
    <item
        android:id="@+id/nav_home"
        android:icon="@drawable/ic_menu_camera"
        android:title="Home" />
    <item
        android:id="@+id/nav_gallery"
        android:icon="@drawable/ic_menu_gallery"
        android:title="Gallery" />
    <item
        android:id="@+id/nav_slideshow"
        android:icon="@drawable/ic_menu_slideshow"
        android:title="Slideshow" />
    <item
        android:id="@+id/nav_tools"
        android:icon="@drawable/ic_menu_manage"
        android:title="Tools" />
</group>

<item android:title="Communicate">
    <menu>
        <item
            android:id="@+id/nav_share"
            android:icon="@drawable/ic_menu_share"
            android:title="Share" />
        <item
            android:id="@+id/nav_send" />
    </menu>
</item>

```

Initial Setup





Navigation Drawer Overview

❑ Android Studio (and the Navigation Library) does a lot of the heavy lifting for you, but generally the following steps are necessary to add a Navigation Drawer to your app

- *Create drawer layout*
- *Bind to navigation drawer layout*
- *Handle navigation drawer click and*
- *Update content based on user selection*

❑ Let's compare....



Overview - *Create Drawer Layout*

- ❑ Done (you just need to change it to suit your app, or maybe not? ☺)



Overview – *Bind to the Drawer Layout etc.*

- ❑ Bind to the Drawer and Navigation View to allow you to handle the user navigation and switching content based on user selection...
- ❑ Done



Overview – *Bind to the Drawer Layout etc.*

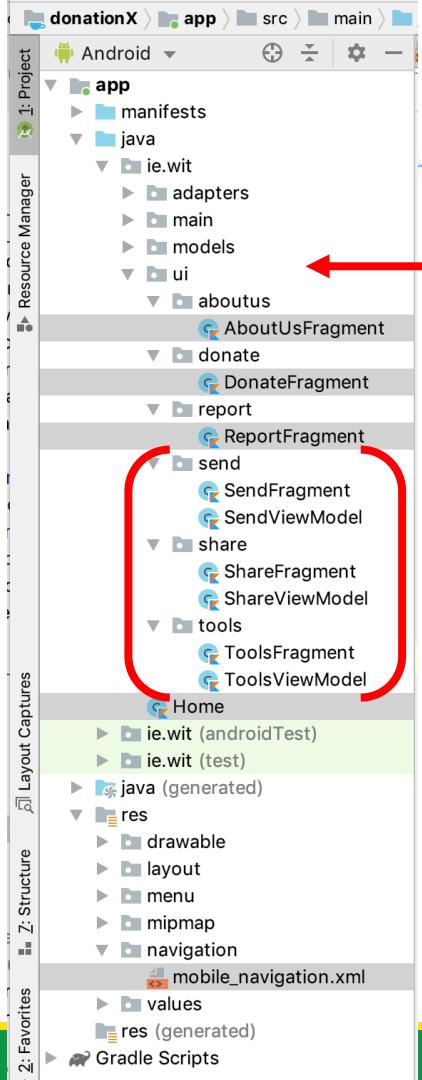
- ❑ Display some kind of initial landing page once the app starts – **Done**
 - (after simply replacing the ‘Home’ fragment and layout with our own ‘Donate’ logic and layout)



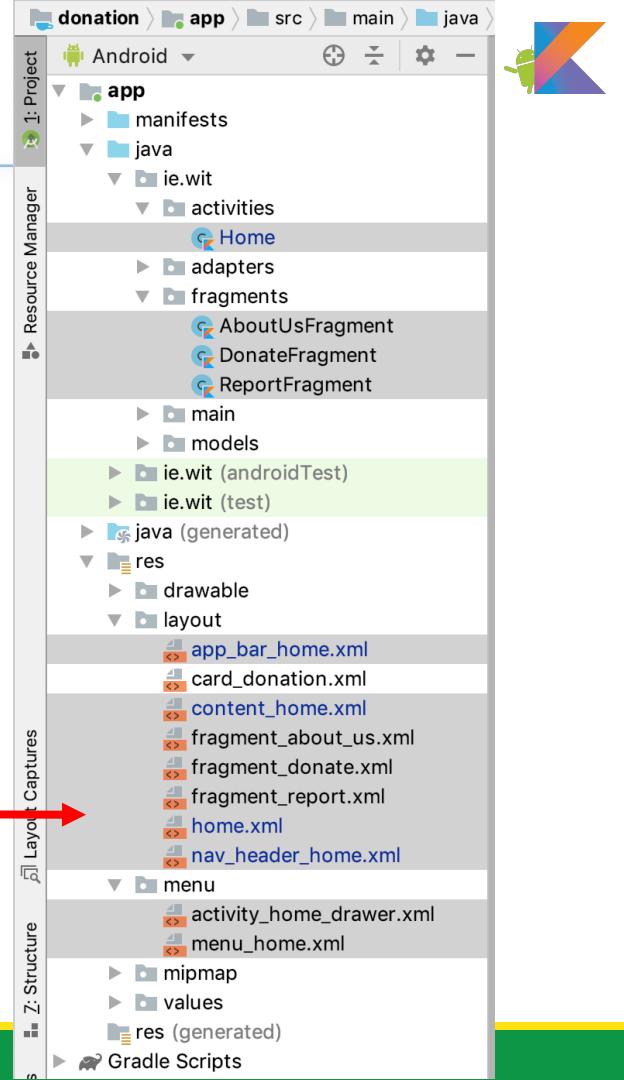
Overview – *Handle Drawer Click & Update Content*

Done

Project Structure



DonationX

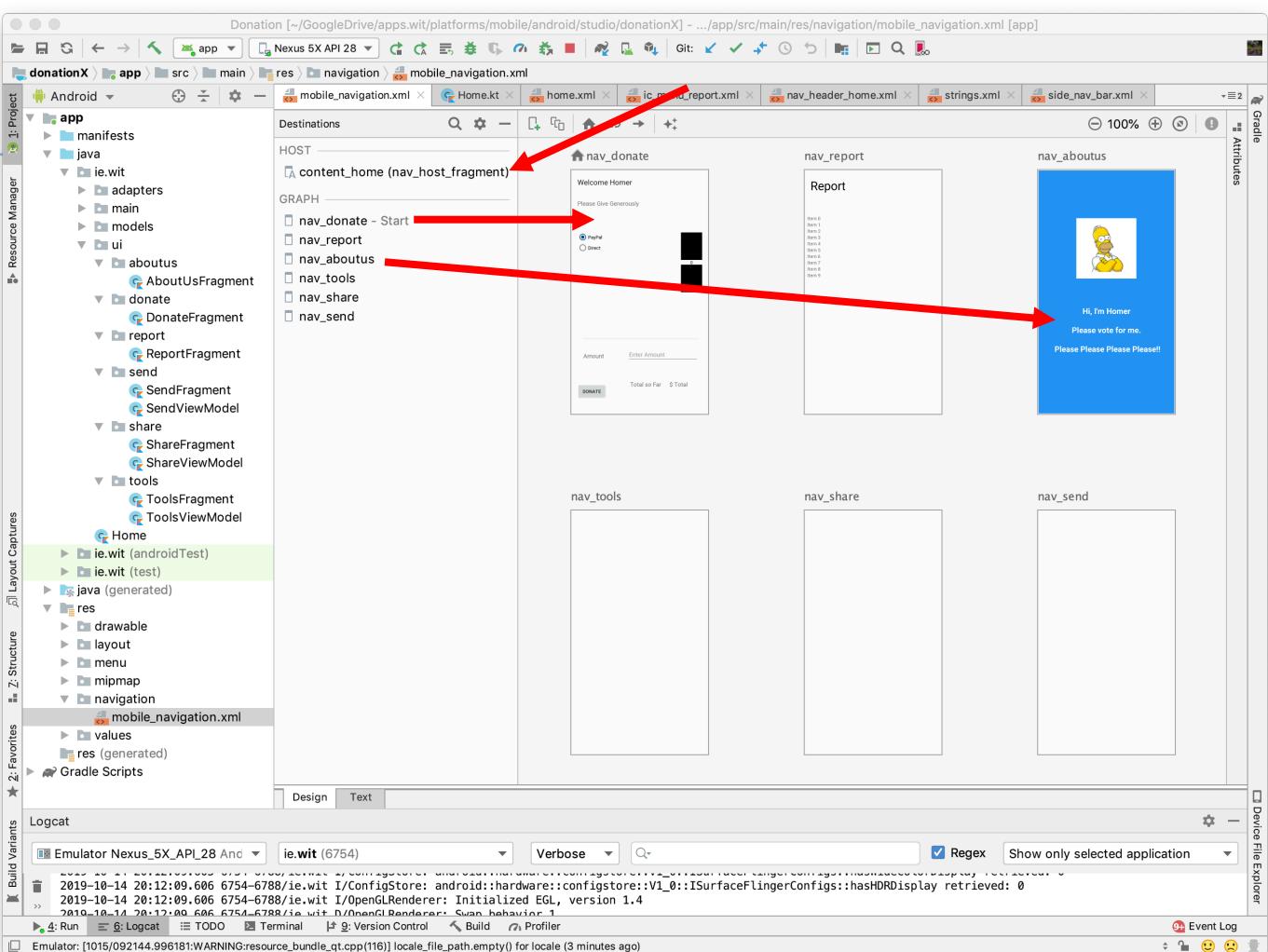


Donation

Navigation in DonationX

Navigation Graph

- Navigation destinations & actions





Source Code Solution
available
at end of
Lab K07a



References

Sources:

- <https://blog.mindorks.com/android-navigation-drawer-in-kotlin>
- <https://medium.com/@umang.burman.micro/navigation-drawer-with-navigation-component-4f032bfdeae6>
- <https://developer.android.com/guide/navigation>
- <https://medium.com/@myric.september/navigation-in-android-navigation-architecture-component-a1f103a6cc52>
- <https://material.io/components/navigation-drawer/#>

Thanks.

