


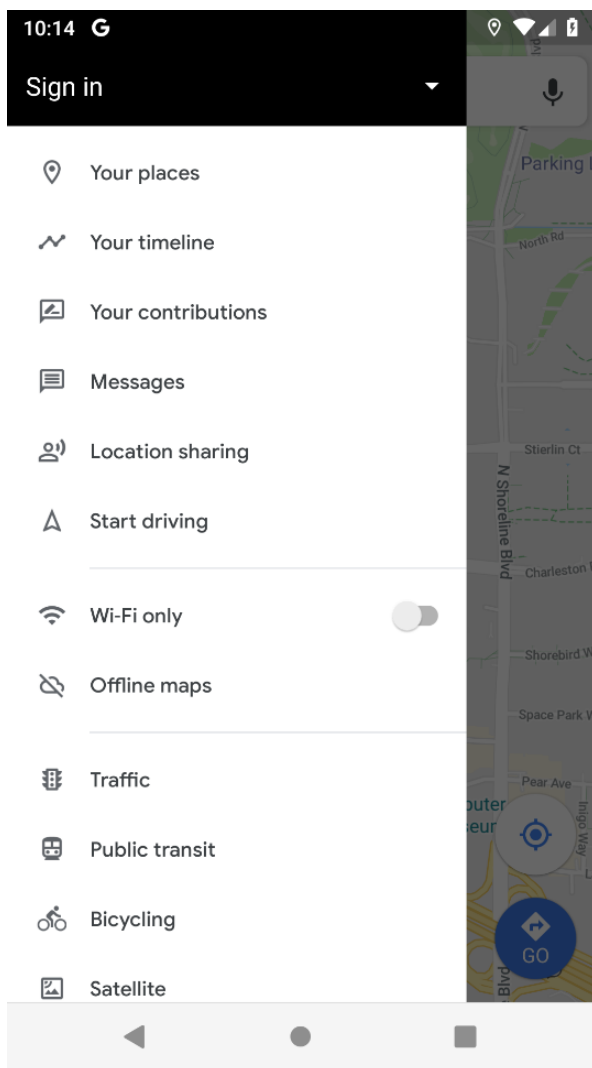
10. Zadanie: dodaj szufladę nawigacji (navigation drawer)

Szuflada nawigacji to panel, który wysuwa się z krawędzi ekranu. Szuflada zazwyczaj zawiera nagłówek i menu.

W urządzeniach wielkości telefonu szuflada nawigacji jest ukryta, gdy nie jest używana. Dwa rodzaje działań użytkownika mogą powodować pojawienie się szuflady nawigacji:

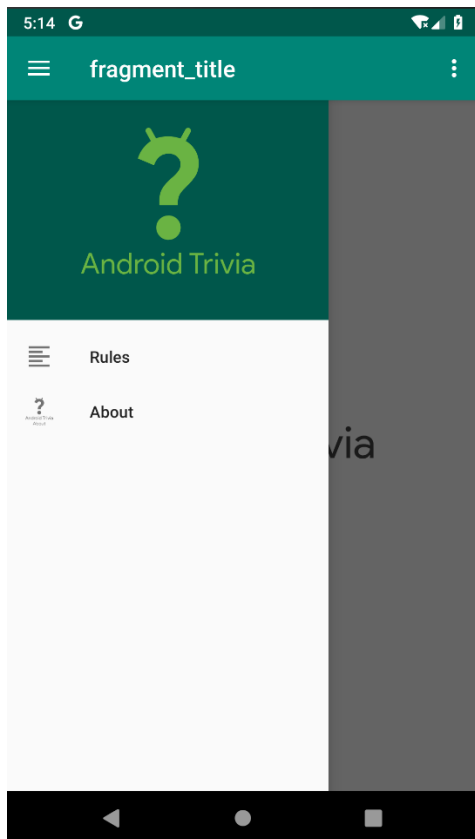
- Szuflada pojawia się, gdy użytkownik przesuwą się od początkowej krawędzi ekranu w kierunku końcowej krawędzi ekranu. W aplikacji szuflada nawigacji pojawi się, gdy użytkownik przesunie palcem od lewej do prawej.
- Szuflada pojawia się, gdy użytkownik znajduje się w początkowym miejscu aplikacji i stuknie ikonę szuflady na pasku aplikacji. (Ikona szuflady jest czasami nazywana przyciskiem szuflady nawigacji lub ikoną hamburgera). *nav drawer button* or *hamburger icon* )

Poniższy zrzut ekranu pokazuje otwartą szufladę nawigacji.



Szuflada nawigacji jest częścią biblioteki [Material Components for Android](#) lub Material library. Korzystasz z biblioteki materiałów, aby wdrażać wzorce, które są częścią wytycznych Google dotyczących projektowania Material Design guidelines.

W aplikacji AndroidTrivia szuflada nawigacji będzie zawierać dwa elementy menu. Pierwszy element wskazuje na istniejący fragment "about", a drugi element wskazuje nowy fragment "rules".



Krok 1: Dodaj bibliotekę materiałów do swojego projektu (Material library)

1. W pliku kompilacji Gradle na poziomie aplikacji dodaj zależność dla biblioteki materiałów Material library:

```
dependencies {  
    ...  
    implementation  
    "com.google.android.material:material:$supportlibVersion"  
    ...  
}
```

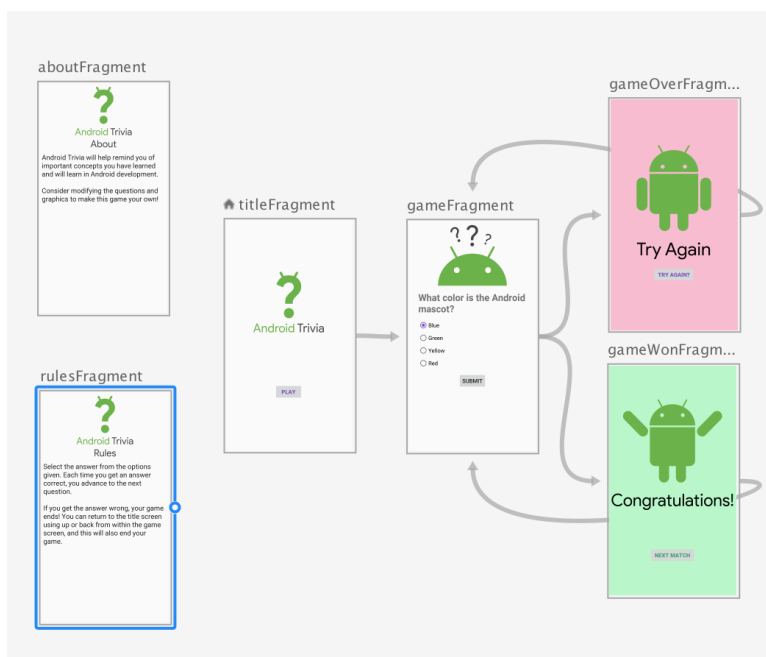
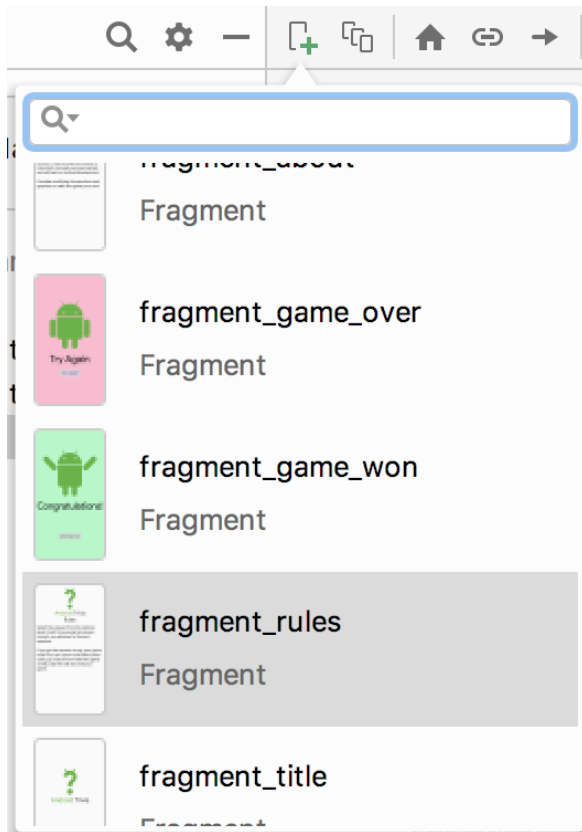
2. Sync your project.

Krok 2: Upewnij się, że fragmenty docelowe mają identyfikatory

Szuflada nawigacji będzie zawierać dwa elementy menu, z których każdy reprezentuje fragment, do którego można uzyskać dostęp z szuflady nawigacji. Oba miejsca docelowe muszą mieć identyfikator na grafie nawigacyjnym.

AboutFragment ma już identyfikator na wykresie nawigacyjnym, ale RulesFragment nie ma, więc dodaj go teraz:

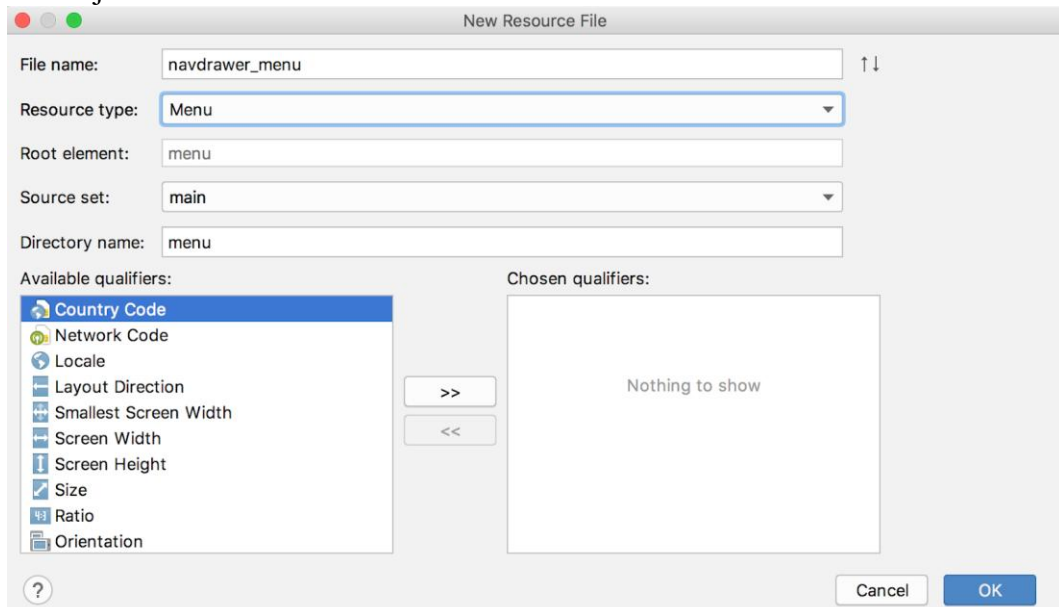
1. Otwórz plik `fragment_rules.xml` aby zobaczyć, jak to wygląda. Kliknij kartę **Design** aby wyświetlić podgląd w edytorze projektu.
2. Otwórz plik `navigation.xml` w Edytorze nawigacji. Kliknij przycisk **New Destination** i dodaj *rules fragment*. Ustaw jego **ID** na `rulesFragment`.



Krok 3: Utwórz drawer menu i drawer layout

Aby utworzyć szufladę nawigacji, należy utworzyć menu nawigacji. Musisz także umieścić swoje widoki wewnątrz `DrawerLayout`.

1. Utwórz menu dla szuflady. W okienku projektu kliknij prawym przyciskiem myszy folder **res** i wybierz **New Resource File**. Nazwij plik **navdrawer_menu**, v **Menu**, and i kliknij **OK**.



2. Otwórz **navdrawer_menu.xml** z folderu **res > menu**, a następnie kliknij kartę **Design**. Dodaj dwa elementy menu, przeciągając elementy menu z panelu **Palette** do panelu **Component Tree**.
3. W przypadku pierwszego elementu menu ustaw identyfikator **id** na **rulesFragment**. (Identyfikator pozycji menu powinien być taki sam jak identyfikator fragmentu.) Ustaw **title** na **@string/rules** oraz **icon** na **@drawable/rules**.



4. W przypadku drugiego elementu menu ustaw identyfikator na **aboutFragment**, **title** na **@string/about**, ikonę na **@drawable/about_android_trivia**.

id	aboutFragment	
title	@string/about	...
icon	@drawable/abou	...

Uwaga: Uwaga: Jeśli użyjesz tego samego identyfikatora dla pozycji menu, co dla fragmentu docelowego, nie musisz pisać żadnego kodu, aby zaimplementować `onClick` listener!

- Otwórz plik `activity_main.xml` layout file. Aby uzyskać wszystkie funkcje szuflady od reki, umieść swoje widoki w `DrawerLayout`. Zawień cały `<LinearLayout>` w `<DrawerLayout>`. (Innymi słowy, dodaj `DrawerLayout` jako widok główny.)

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <androidx.drawerlayout.widget.DrawerLayout
        android:id="@+id/drawerLayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

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

- Teraz dodaj szufladę, która jest widokiem nawigacji `NavigationView` i korzysta z właśnie zdefiniowanego menu `navdrawer_menu`. Dodaj następujący kod w `DrawerLayout`, , po elemencie `</LinearLayout>`:

```
<com.google.android.material.navigation.NavigationView
    android:id="@+id/navView"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    app:headerLayout="@layout/nav_header"
    app:menu="@menu/navdrawer_menu" />
```


Krok 4: Wyświetl szufladę nawigacji

Utworzono elementy menu dla szuflady nawigacji i układu szuflady nawigacji. Teraz musisz podłączyć szufladę nawigacyjną do kontrolera nawigacyjnego, aby po wybraniu elementów w szufladzie nawigacyjnej aplikacja nawigowała do odpowiedniego fragmentu.

- Otwórz plik `Mainactivity.kt`. W funkcji `onCreate()`, dodaj kod, który pozwala użytkownikowi wyświetlić szufladę nawigacji. Zrób to, wywołując `setupWithNavController()`. Dodaj następujący kod na dole `onCreate()`:

```
NavigationUI.setupWithNavController(binding.navView, navController)
```

2. Uruchom aplikację. Przesuń od lewej krawędzi, aby wyświetlić szufladę nawigacji, i upewnij się, że każdy element menu w szufladzie znajduje się we właściwym miejscu.

Chociaż szuflada nawigacji działa, musisz naprawić jeszcze jedną rzecz. Zazwyczaj aplikacje pozwalają również użytkownikom wyświetlać szufladę nawigacji, dotykając przycisku szuflady (trzy linie)  na pasku aplikacji na ekranie głównym. Twoja aplikacja nie wyświetla jeszcze przycisku szuflady na ekranie głównym.

Krok 5: Wyświetl szufladę nawigacji za pomocą przycisku szuflady

Ostatnim krokiem jest umożliwienie użytkownikowi dostępu do szuflady nawigacji za pomocą przycisku szuflady w lewym górnym rogu paska aplikacji.

1. W pliku `MainActivity.kt` dodaj zmienną `lateinit drawerLayout` reprezentującą drawer layout:

```
private lateinit var drawerLayout: DrawerLayout
```

Uwaga: Kotlin jest językiem “null safety language”. Jednym ze sposobów zapewniania bezpieczeństwa zerowego jest modyfikator `lateinit`, który pozwala opóźnić inicjalizację zmiennej bez ryzyka zwrócenia zerowego odniesienia (`null reference`).

W takim przypadku, `drawerLayout` jest zadeklarowany za pomocą `lateinit` to aby uniknąć konieczności nadania mu wartości `null`. Zostanie zainicjowany `onCreate()`.

2. W metodzie `onCreate()` zainicjuj `drawerLayout`, po zainicjowaniu zmiennej `binding`.

```
val binding = DataBindingUtil.setContentView<ActivityMainBinding>(this,
    R.layout.activity_main)
```

```
drawerLayout = binding.drawerLayout
```

3. Dodaj `drawerLayout` jako trzeci parametr do metody `setupActionBarWithNavController()`:


```
NavigationUI.setupActionBarWithNavController(this, navController,
drawerLayout)
```

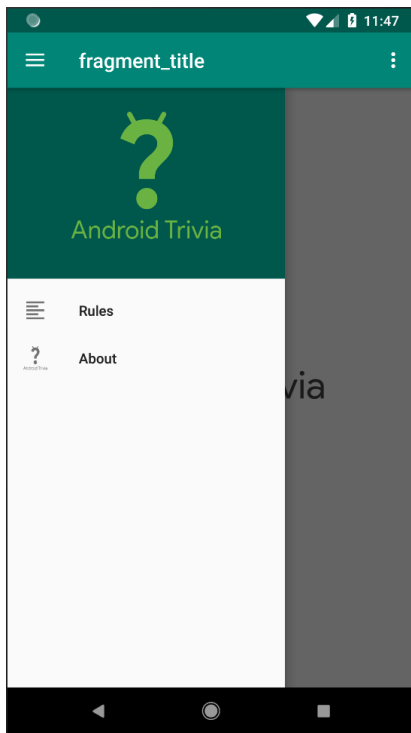
4. Edytuj metodę `onSupportNavigateUp()` aby zwracać `NavigationUI.navigateUp` zamiast zwracać `navController.navigateUp`. Przekaż kontroler nawigacyjny i układ szuflady do `navigateUp()`. Metoda będzie wyglądać następująco:

```
override fun onSupportNavigateUp(): Boolean {
    val navController = this.findNavController(R.id.myNavHostFragment)
    return NavigationUI.navigateUp(navController, drawerLayout)
}
```

5. Może być konieczne dodanie kolejnego importu do pliku, aby wszystkie odwołania zostały rozwiązane, na przykład:

```
import androidx.drawerlayout.widget.DrawerLayout
```

6. Uruchom aplikację. Przesuń od lewej krawędzi, aby wyświetlić szufladę nawigacji, i upewnij się, że każdy element menu w szufladzie znajduje się we właściwym miejscu.
7. Przejdź do ekranu głównego i dotknij przycisku szuflady nawigacji, aby upewnić się, że szuflada nawigacji się pojawi.  to make sure the navigation drawer appears. Upewnij się, że kliknięcie opcji **Rules** or **About** w szufladzie nawigacji przeniesie Cię we właściwe miejsce.



Gotowe!

Do swojej aplikacji dodano kilka różnych opcji nawigacji.
Użytkownik może teraz przechodzić przez aplikację, grając w grę.

Może wrócić do ekranu głównego w dowolnym momencie za pomocą przycisku W górę.
Może przejść do ekranu Informacje z menu Opcje lub z szuflady nawigacji.

Naciśnięcie przycisku Wstecz powoduje przejście z powrotem przez poprzednie ekrany w sposób, który ma sens dla aplikacji.

Użytkownik może otworzyć szufladę nawigacji, przeciągając palcem od lewej strony na dowolnym ekranie lub naciskając przycisk szuflady na pasku aplikacji na ekranie głównym.

Twoja aplikacja zawiera solidne, logiczne ścieżki nawigacji, które są intuicyjne w obsłudze dla użytkownika.