[https://youtu.be/6zp-AKBaMT8](https://youtu.be/6zp-AKBaMT8" \t "_blank)

<https://youtu.be/yihpEWmKoOw>

**Тема: Навигация в приложении. Начало работы с Navigation Architecture Component**

**Время выполнения 2 часа**

**Теоретический материал**

Navigation Architecture Component упрощает осуществление навигации, а также помогает визуализировать navigation flow вашего приложения. Библиотека предоставляет ряд преимуществ, в том числе:

* Автоматическая обработка транзакций фрагментов
* Корректная обработка кнопок «В**верх»** и «**Назад»** по умолчанию
* Поведение по умолчанию для анимации и переходов
* Deep linking как first class operation
* Реализация шаблонов навигации пользовательского интерфейса (таких как navigation drawer и bottom navigation**)** с небольшой дополнительной работой
* Безопасность типов при передаче информации во время навигации
* Инструменты Android Studio для визуализации и редактирования navigation flow приложения

**Обзор компонента навигации**

Компонент навигации состоит из трех ключевых частей:

* Navigation graph: ресурс XML, который содержит всю связанную с навигацией информацию в одном централизованном месте. Это включает в себя все отдельные области содержимого в вашем приложении, называемые destinations (пункты назначения), а также возможные пути, которые пользователь может пройти через ваше приложение.
* NavHost: Пустой контейнер, который отображает пункты назначения из вашего графика навигации. Компонент Navigation содержит реализацию NavHost по умолчанию — [NavHostFragment](https://developer.android.com/reference/androidx/navigation/fragment/NavHostFragment.html" \t "_blank), которая отображает фрагменты — места назначения.
* NavController: Объект, который управляет навигацией приложения в NavHost. NavController управляет перемещениями контента мест назначения в NavHost, в процессе перемещения пользователей по приложению.

 Мы используем объект NavController, сообщая ему путь в ресурсе Navigation Graph. Затем объекту NavController будет показан соответствующий пункт назначения в NavHostFragment.

Давайте посмотрим, как это выглядит на практике, начиная с нового ресурса Navigation Graph.

**Destinations**

Компонент навигации представляет концепцию Destinations —  пункта назначения . Пункт назначения — это любое место, в котором вы можете перемещаться в приложении, обычно это фрагмент или активити. Они поддерживаются «из коробки», но вы также можете создавать свои собственные типы назначения, если это необходимо.

**Navigation Graph**

Navigation Graph представляет собой новый тип ресурса , который определяет все возможные пути, доступные пользователю в приложении. Он показывает визуально все пункты назначения, которые могут быть достигнуты из данного пункта назначения. Редактор навигации Android Studio отображает Navigation Graph наглядно.

**Ход работы:**

**Создаем новый проект в Android Studio** с использованием шаблона Empty views Activity.

**Добавляем в проект библиотеку Navigation Architecture Component**

Чтобы добавить библиотеку Navigation Architecture Component в наш проект, откройте файл сборки build.gradle модуля App. Добавьте в секцию dependencies{}  такие зависимости:

***dependencies* {  
  
 *implementation* ("androidx.navigation:navigation-fragment-ktx:2.3.0-beta01")  
 *implementation* ("androidx.navigation:navigation-ui-ktx:2.3.0-beta01")  
  
 *//*** *Dynamic Feature Module Support  
 implementation* ("androidx.navigation:navigation-dynamic-features-fragment:2.3.0-beta01")  
  
 *// Testing Navigation  
 androidTestImplementation* ("androidx.navigation:navigation-testing:2.3.0-beta01")

После всех изменений синхронизируйте проект с Gradle нажав на кнопку Sync.

**Navigation graph** Давайте его создадим.

Кликните правой кнопкой мыши на папке res вашего проекта и в контекстном меню выберите пункты New > Android Resource File. Откроется диалог New Resource File, где нужно указать имя файла, например, «nav\_graph». В выпадающем списке Resource type выберите Navigation. Нажмите кнопку ОК для закрытия диалога и создания файла навигационного графа.

В результате Android Studio создаст в папке res папку navigation, и в ней файл nav\_graph.xml.

**Navigation Editor**

После создания навигационного графа он откроется в редакторе навигации. Как и файлы разметки его можно редактировать двумя способами – либо в визуальном редакторе, либо в XML. Окно редактора навигации похоже на окно редактора макетов – слева список пунктов назначения, в центре визуальное представление, справа панель атрибутов выбранного компонента. Кнопками можно переключаться между представлениями – дизайн, текст или комбинированное представление.

Переключитесь на текстовое представление и посмотрите его описание:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?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"              android:id="@+id/nav\_graph">    </navigation> |

Сейчас навигационный граф не связан ни с одним файлом разметки, содержащим компонент NavHostFragment. Давайте это исправим.

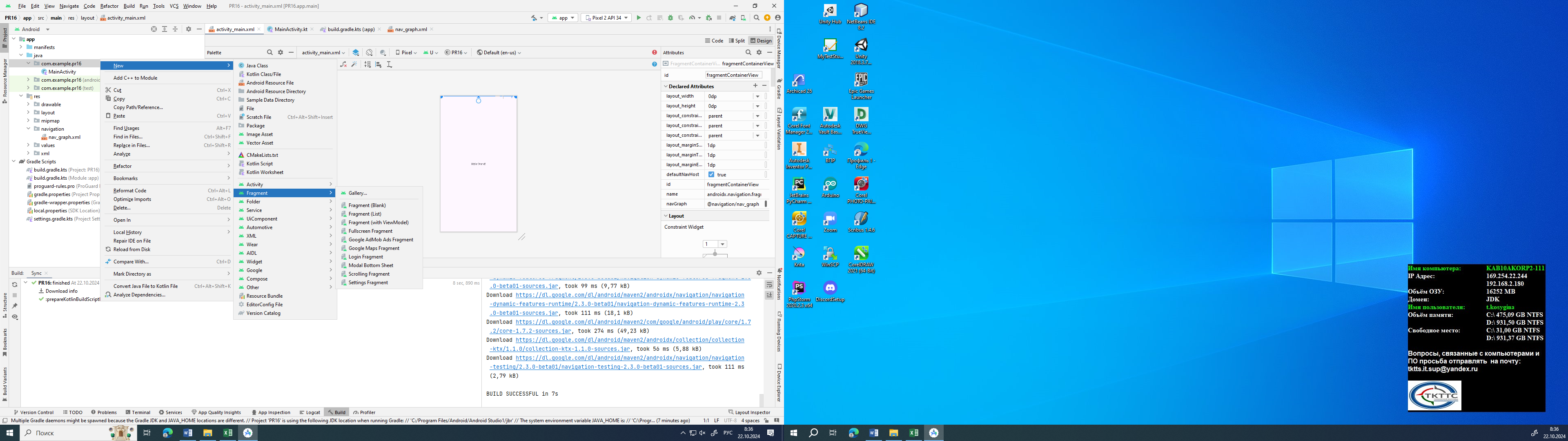
**NavHostFragment**

Для каждого навигационного графа должен быть определен хост. Например, хостом может служить активити, к которому привязаны фрагменты, представляющие пункты назначения для навигации. Причем для каждого такого активити должен быть определен свой навигационный граф.

Чтобы добавить хост, откройте файл разметки activity\_main.xml в редакторе макетов. Переключитесь на вкладку дизайна. В палитре компонентов слева найдите в группе Containers компонент NavHostFragment и перетащите его в макет активити. При этом откроется окно выбора навигационного графа, где показан тот, который мы недавно создали – выберите его. Выполните привязку компонента к краям экрана, нажав и потянув поочередно каждый из маркеров к своему краю экрана. Затем переключитесь в режим текста и укажите размеры для компонента NavHostFragment – wrap\_content. Компонент займет весь экран.

**Добавление пунктов назначения**

Давайте добавим новый фрагмент в приложение, щёлкнув правой клавишей мыши по главному пакету приложения и выбрав пункты контекстного меню: New> Fragment> Fragment (Blank).



После создания макет разметки этого фрагмента будет доступен для выбора в редакторе навигации. Кликните по кнопке с изображением экрана с зеленым плюсиком в правом нижнем углу, и в окне добавления пункта назначения вы увидите список макетов разметки. Выберите только что добавленный фрагмент, и он добавится на центральную панель. Обратите внимание – он обозначен иконкой в виде домика – это значит, что это домашний, стартовый экран для этого навигационного графа.

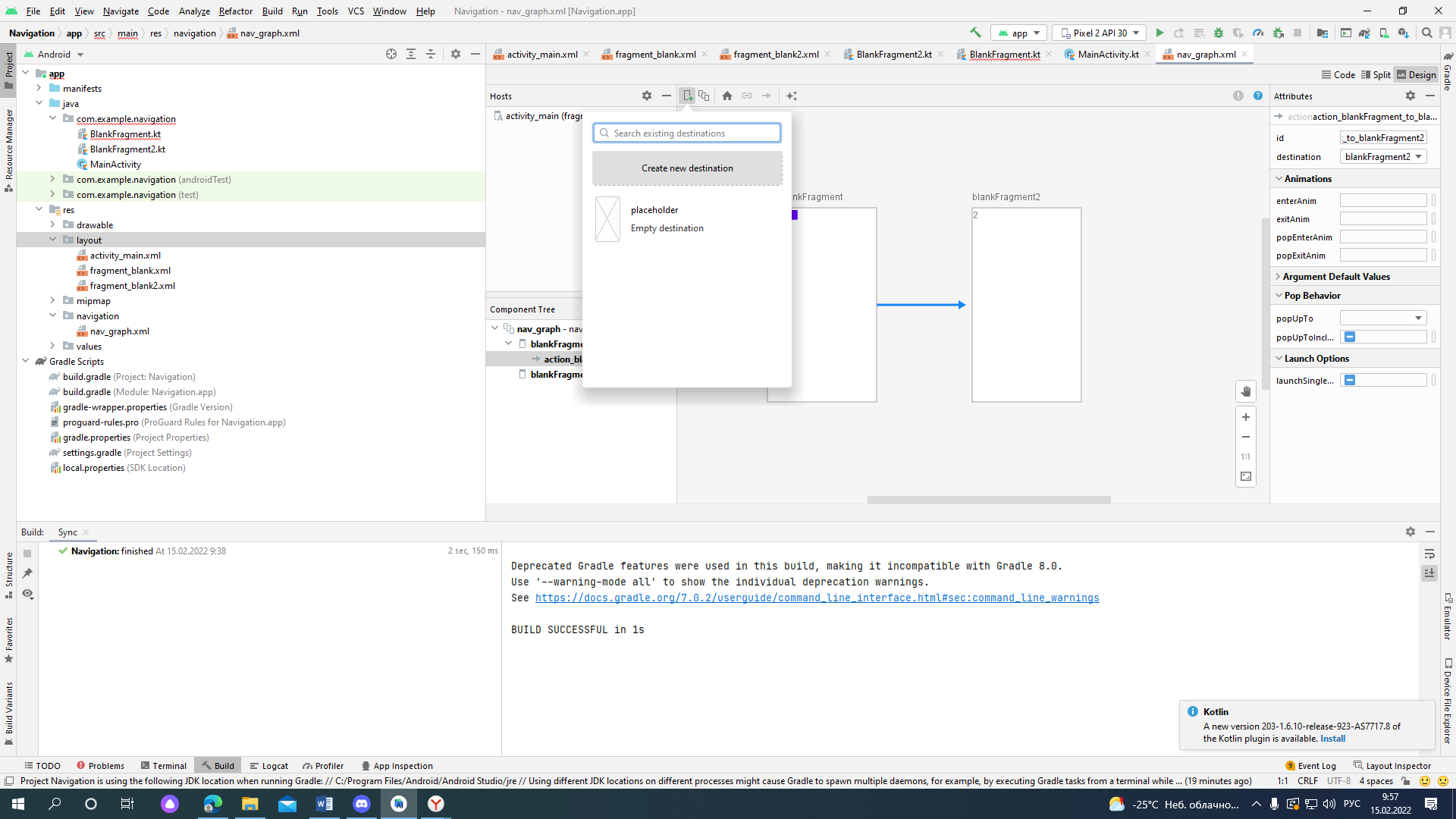
Вы можете добавить существующий фрагмент или активити в качестве пункта назначения, или создать его прямо из редактора навигации. Для создания нового пункта назначения нажмите кнопку Create new destination. Будет создан и добавлен на центральную панель новый фрагмент.

Также вместо пункта назначения можно добавить временный заполнитель —  placeholder. В редакторе навигации с заполнителями можно взаимодействовать так же, как и с обычными пунктами назначения, например, определять действия – actions. В текстовом представлении заполнитель обозначается тегами <fragment></fragment>, как и обычные фрагменты.

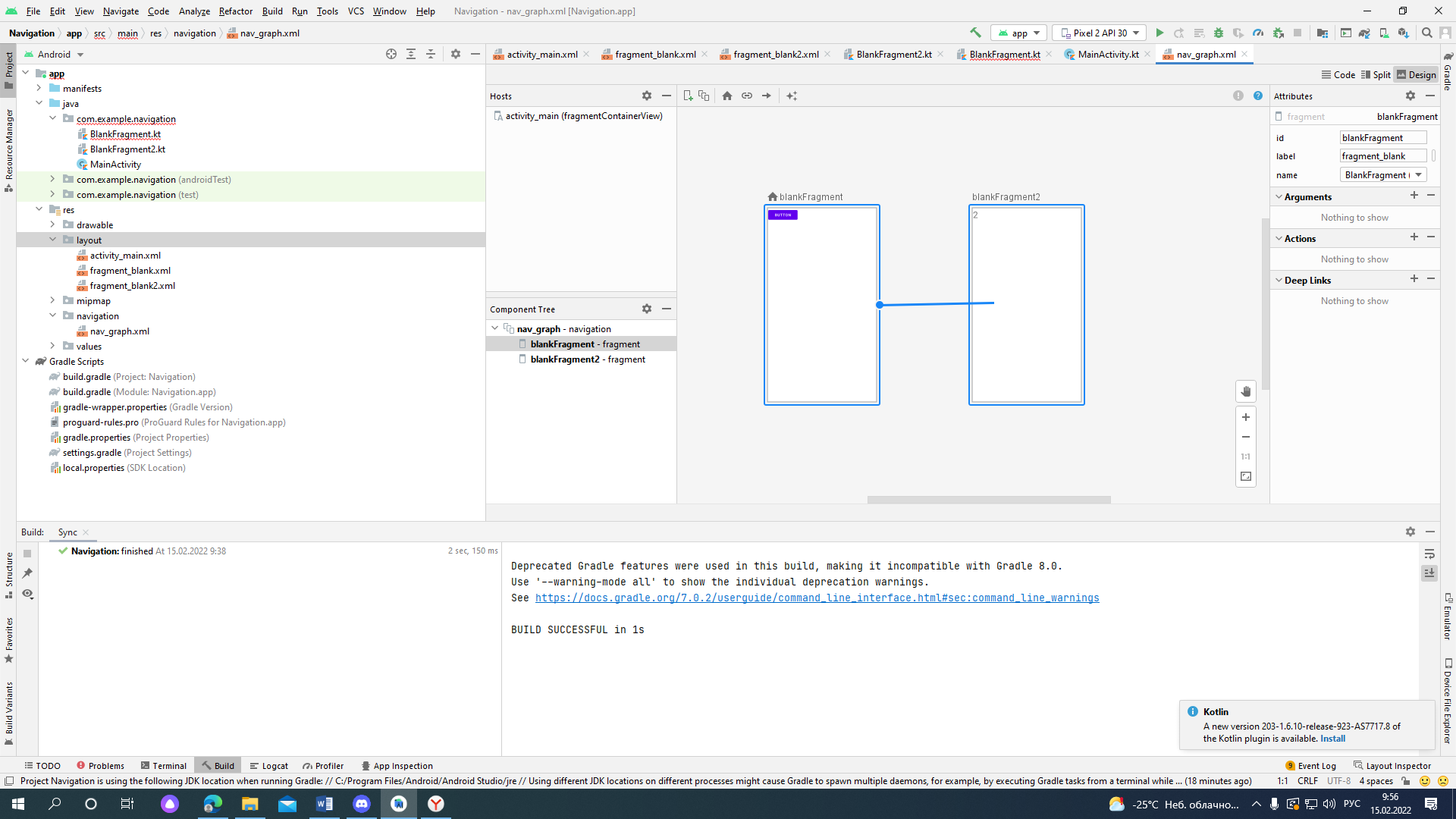
Перед тестированием приложения нужно заменить все заполнители настоящими пунктами назначения, иначе попытка перехода в приложении на placeholder завершится ошибкой.

**Добавление переходов**

Давайте пока удалим заполнитель и оставим два фрагмента. Второй фрагмент создаем



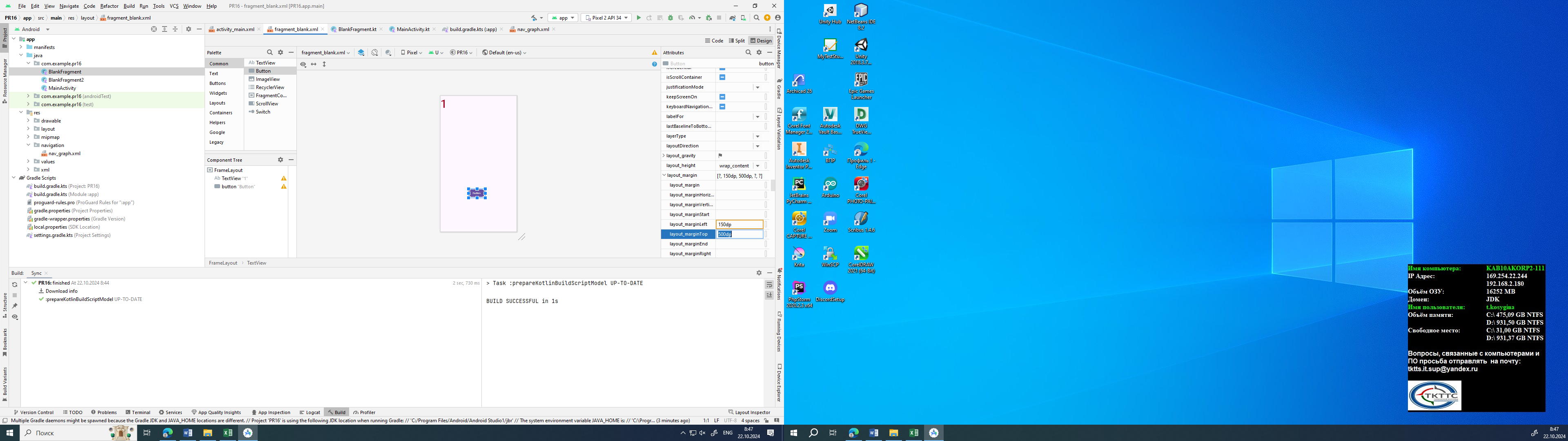
Добавим действие для перехода с одного экрана на другой.



Как вы уже поняли, для этого нужно нажать маркер одного фрагмента и протянуть до другого. Добавится стрелочка, обозначающая действие – переход с одного фрагмента на другой.

Если мы посмотрим на текстовое представление, то увидим, что у домашнего фрагмента появился тег <action></action>, внутри которого обозначен идентификатор действия и пункт назначения.

Теперь откройте макет домашнего фрагмента и вместо шаблонного текста в поле TextView добавьте цифру 1, а размер текста установите побольше. Также добавьте сюда кнопку. ЕЕ опустите вниз с помощью свойств



В макете второго фрагмента аналогично напишите цифру 2 поле TextView.

Теперь в классе первого фрагмента переопределите метод жизненного цикла onViewCreated() и внутри него напишите код слушателя нажатия кнопки:

|  |  |
| --- | --- |
|  | override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)  val button = requireView().findViewById<View>(R.id.*button*)  button.setOnClickListener **{** view: View **->** view.*findNavController*().navigate(R.id.*action\_blankFragment\_to\_blankFragment2*)  **}** |

 Здесь мы находим контроллер навигации и вызываем его метод navigate, которому передаем идентификатор действия (action), добавленного в редакторе навигации между фрагментами.

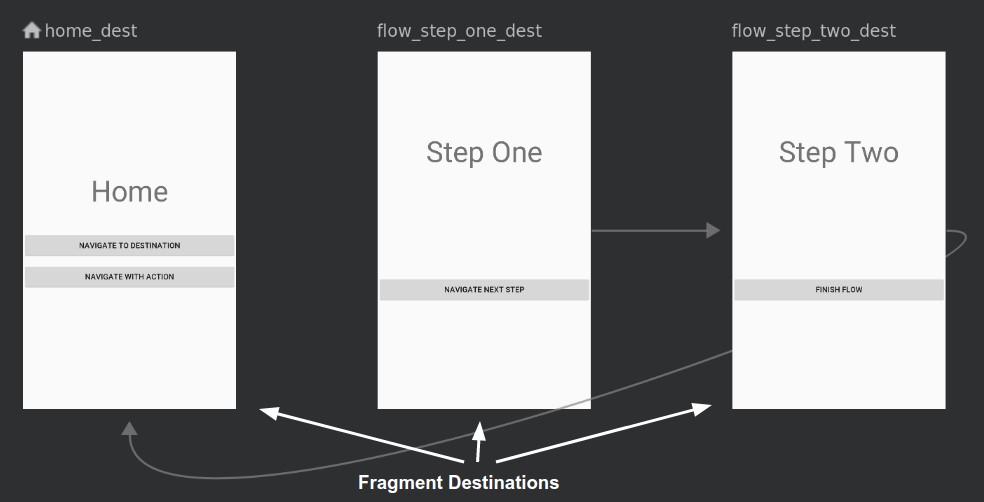
**Проверка переходов в приложении**

Теперь запустите приложение в эмуляторе или смартфоне. Обратите внимание – при открытии отображается домашний фрагмент, хотя мы его не прописывали в макете активити. Мы просто определили макет активити как хост навигационного графа, и теперь при старте этого активити библиотека навигации отображает нам фрагмент, указанный как домашний в этом графе.

Нажмите на кнопку, и выполнится действие – переход от первого фрагмента ко второму. Нажатие системной кнопки «Назад» вернет вас на первый экран. Это простейший пример навигации, а на следующем уроке мы разнообразим его и добавим более сложную навигацию в приложение.

**Часть 2**

1. Откройте res/navigation/ nav\_graph.xml
2. Перейдите в режим «Дизайн»:
3. сделайте изменения



Navigation Graph показывает доступные пункты назначения. Стрелки между пунктами назначения называются *actions* (*действия)*. Больше мы поговорим о них позже.

1. Нажмите на пункт назначения, чтобы увидеть его атрибуты.
2. Нажмите на любое действие, представленное стрелкой, чтобы увидеть его атрибуты.

**Анатомия навигационного XML-файла**

Все изменения, которые вы делаете в графическом редакторе навигации, изменяют базовый XML-файл, подобно тому, как редактор макетов изменяет XML-макет.

Перейдите на вкладку « **Текст** »:

Вы увидите такой XML-код:



|  |  |
| --- | --- |
|  | <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"       app:startDestination="@+id/home\_dest">        <!-- ...tags for fragments and activities here -->   </navigation> |

Примечание:

* <navigation> является корневым узлом каждого навигационного графа.
* <navigation>содержит один или несколько пунктов назначения, представленных элементами <activity>или <fragment>.
* app:startDestination является атрибутом, который указывает место назначения, которое запускается по умолчанию, когда пользователь впервые открывает приложение.

Давайте посмотрим на место назначения фрагмента:



|  |  |
| --- | --- |
|  | <fragment      android:id="@+id/flow\_step\_one\_dest"     android:name="com.example.android.codelabs.navigation.FlowStepFragment"      tools:layout="@layout/flow\_step\_one\_fragment">      <argument          .../>        <action          android:id="@+id/next\_action"          app:destination="@+id/flow\_step\_two\_dest">      </action>  </fragment> |

Примечание:

* android:id определяет идентификатор для фрагмента, который вы можете использовать для ссылки на место назначения в другом месте этого XML и вашего кода.
* android:name объявляет полное имя класса фрагмента для создания экземпляра при переходе к этому месту назначения.
* tools:layout указывает, какой макет должен отображаться в графическом редакторе.

Некоторые теги <fragment> также содержат <action>, <argument>,и <deepLink>, все это мы рассмотрим позже.

Пример приложения уже содержит несколько пунктов назначения на графике. Давайте мы добавим новое назначение.

**Примечание**. Код для каждого шага в этой кодовой метке включен, закомментирован между операторами TODO в загруженном вами коде.

**NavController**

Наконец, когда пользователь делает что-то вроде нажатия кнопки, вам нужно запустить команду навигации. Нам поможет специальный класс **[NavController](http://d.android.com/reference/androidx/navigation/NavController" \t "_blank)**, который управляет фрагментами в NavHostFragment.



|  |  |
| --- | --- |
|  | // Command to navigate to flow\_step\_one\_dest  findNavController().navigate(R.id.flow\_step\_one\_dest) |

Обратите внимание, что вы передаете либо destination , либо action ID для навигации. Это идентификаторы, определенные в графе навигации XML. Это пример передачи destination ID.

NavController  мощный инструмент, потому что когда вы вызываете методы типа navigate() или popBackStack(), он переводит эти команды в соответствующие операции фреймворка в зависимости от типа пункта назначения, к которому вы переходите. Например, когда вы вызываете navigate() с места назначения активити, NavController вызывает startActivity() от вашего имени.

Есть несколько способов получить объект NavController, связанный с вашим NavHostFragment. В Kotlin рекомендуется использовать одну из следующих функций расширения в зависимости от того, вызываете ли вы команду навигации из фрагмента, активити или view-компонента:

* [Fragment.findNavController()](https://developer.android.com/reference/kotlin/androidx/navigation/fragment/package-summary#findnavcontroller)
* [View.findNavController()](https://developer.android.com/reference/kotlin/androidx/navigation/package-summary#%28android.view.View%29.findNavController%28%29)
* [Activity.findNavController(viewId: Int)](https://developer.android.com/reference/kotlin/androidx/navigation/package-summary#findnavcontroller)

Ваш NavController ассоциируется с NavHostFragment. Таким образом, какой бы метод вы ни использовали, вы должны быть уверены, что идентификатор фрагмента, активити или view либо собственно NavHostFragment  либо имеет NavHostFragment в качестве родителя. В противном случае вы получите IllegalStateException.

**Перейдите к месту назначения с помощью NavController**

Подключим кнопку **Navigate To Destination** , чтобы перейти к пункту назначения flow\_step\_one\_dest (который является пунктом назначения FlowStepFragment):

1. Открыть HomeFragment.kt
2. подключить navigate\_destination\_button в onViewCreated()

**HomeFragment.kt**



|  |  |
| --- | --- |
|  | val button = view.findViewById<Button>(R.id.navigate\_destination\_button)  button?.setOnClickListener {      findNavController().navigate(R.id.flow\_step\_one\_dest, null)  } |

1. Запустите приложение и нажмите кнопку «**Перейти к** месту **назначения»**. Обратите внимание, что кнопка перемещается к flow\_step\_one\_dest.

Этот метод создаст OnClickListener для перехода к заданному месту назначения с набором аргументов, которые будут переданы получателю.

Код слушателя кликов будет выглядеть так:



|  |  |
| --- | --- |
|  | val button = view.findViewById<Button>(R.id.navigate\_destination\_button)  button?.setOnClickListener(  Navigation.createNavigateOnClickListener(R.id.flow\_step\_one\_dest, null) |