|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №2**

по дисциплине «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  студент группыИКБО-28-22 | Некрасов Г.А. |
| **Проверил:**  к.э.н доцент | Степанов П.В. |

Москва 2024 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКОЕ ЗАДАНИЕ 3](#_Toc184750546)

[Описание ручного подхода и Navigation API 4](#_Toc184750547)

[Код практического задания 5](#_Toc184750548)

[Тестирование практического задания 9](#_Toc184750549)

[Вывод 13](#_Toc184750550)

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1. Реализовать приложение, состоящее из трех фрагментов. Фрагменты должны иметь разное наполнение, а также минимальный функционал для возможности их идентификации по внешнему виду.

2. Реализовать навигацию между созданными фрагментами ручным управлением транзакцией и с использованием Navigation API.

3. Обе реализации навигации должны иметь возможность возвращений к предыдущему фрагменту.

# Описание ручного подхода и Navigation API

**Ручной подход** — это когда вы напрямую используете FragmentManager для управления переходами между фрагментами. Например, вы явно вызываете метод replaceFragment() при нажатии на кнопку. Это требует ручного управления транзакциями (например, добавления в стек возврата).

Эти переходы *не зависят от графа навигации*, и вы сами определяете, как обрабатывать стек возврата.

**Navigation API** — это когда вы используете NavController для выполнения навигации, основанной на заранее заданном графе навигации (nav\_graph.xml). В этом случае переходы можно инициировать как с помощью кнопок, так и через элементы, связанные с NavController, например:

Кнопки: Используя navController.navigate().

ActionBar/Toolbar: При использовании встроенной поддержки Navigation API, стрелка "Назад" и заголовки автоматически работают на основе текущего состояния графа.

*Граф навигации управляет тем, как происходят переходы*, в каком порядке, и какие элементы стека возврата используются.

# Код практического задания

В ходе выполнения работы, мы написали фрагменты, а также реализовали переходы между фрагментами на языке Kotlin (см. Листинги 1-4)

Листинг 1. FragmentOne.kt

|  |
| --- |
| package com.example.practi4eskaya\_2  import android.os.Bundle import android.view.View import android.widget.Button import androidx.fragment.app.Fragment import androidx.navigation.fragment.findNavController  class FragmentOne : Fragment(R.layout.fragment\_one) {   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   val buttonNext = view.findViewById<Button>(R.id.buttonNext)  buttonNext.setOnClickListener {  val mainActivity = activity as MainActivity   if (mainActivity.useNavigationApi) {  findNavController().navigate(R.id.action\_fragmentOne\_to\_fragmentTwo)  } else {  parentFragmentManager.beginTransaction()  .replace(R.id.fragmentContainer, FragmentTwo())  .addToBackStack(null)  .commit()  }  }  } } |

Листинг 2. FragmentTwo.kt

|  |
| --- |
| package com.example.practi4eskaya\_2  import android.os.Bundle import android.view.View import android.widget.Button import androidx.fragment.app.Fragment import androidx.navigation.fragment.findNavController  class FragmentTwo : Fragment(R.layout.fragment\_two) {   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   val buttonNext = view.findViewById<Button>(R.id.buttonNext)  val buttonBack = view.findViewById<Button>(R.id.buttonBack)   // Переход к FragmentThree  buttonNext.setOnClickListener {  val mainActivity = activity as MainActivity   if (mainActivity.useNavigationApi) {  findNavController().navigate(R.id.action\_fragmentTwo\_to\_fragmentThree)  } else {  parentFragmentManager.beginTransaction()  .replace(R.id.fragmentContainer, FragmentThree())  .addToBackStack(null)  .commit()  }  }   // Возврат к FragmentOne  buttonBack.setOnClickListener {  if (parentFragmentManager.backStackEntryCount > 0) {  parentFragmentManager.popBackStack()  } else if ((activity as MainActivity).useNavigationApi) {  findNavController().popBackStack()  }  }  } } |

Листинг 3. FragmentThree.kt

|  |
| --- |
| package com.example.practi4eskaya\_2  import android.os.Bundle import android.view.View import android.widget.Button import androidx.fragment.app.Fragment import androidx.navigation.fragment.findNavController  class FragmentThree : Fragment(R.layout.fragment\_three) {   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   val buttonBack = view.findViewById<Button>(R.id.buttonBack)   // Возврат к FragmentTwo  buttonBack.setOnClickListener {  if (parentFragmentManager.backStackEntryCount > 0) {  parentFragmentManager.popBackStack()  } else if ((activity as MainActivity).useNavigationApi) {  findNavController().popBackStack()  }  }  } } |

Листинг 4. MainActivity.kt

|  |
| --- |
| package com.example.practi4eskaya\_2  import android.os.Bundle import android.view.View import androidx.appcompat.app.AppCompatActivity import androidx.fragment.app.Fragment import androidx.navigation.NavController import androidx.navigation.fragment.NavHostFragment import androidx.navigation.ui.setupActionBarWithNavController import com.example.practi4eskaya\_2.databinding.ActivityMainBinding  class MainActivity : AppCompatActivity() {   private lateinit var binding: ActivityMainBinding  var useNavigationApi = true // Флаг для выбора типа навигации  private lateinit var navController: NavController   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  binding = ActivityMainBinding.inflate(layoutInflater)  setContentView(binding.root)   // Устанавливаем Toolbar как ActionBar  setSupportActionBar(binding.toolbar)   // Настраиваем NavController для Navigation API  val navHostFragment = supportFragmentManager.findFragmentById(R.id.navHostFragment) as NavHostFragment  navController = navHostFragment.navController   setupActionBarWithNavController(navController)   setupNavigation()  toggleNavigationMode(useNavigationApi)  }   private fun setupNavigation() {  binding.buttonNavigateManual.setOnClickListener {  toggleNavigationMode(false)  replaceFragment(FragmentOne()) // Ручная навигация  }   binding.buttonNavigateApi.setOnClickListener {  toggleNavigationMode(true)  navController.navigate(R.id.fragmentOne) // Навигация API  }  }   private fun toggleNavigationMode(enableApi: Boolean) {  useNavigationApi = enableApi  binding.navHostFragment.visibility = if (useNavigationApi) View.VISIBLE else View.GONE  binding.fragmentContainer.visibility = if (useNavigationApi) View.GONE else View.VISIBLE  }   private fun replaceFragment(fragment: Fragment) {  supportFragmentManager.beginTransaction()  .replace(R.id.fragmentContainer, fragment)  .addToBackStack(null) // Добавляем в стек для возврата  .commit()  }   override fun onSupportNavigateUp(): Boolean {  return if (useNavigationApi) {  navController.navigateUp() || super.onSupportNavigateUp()  } else {  supportFragmentManager.popBackStack() // Возвращает Boolean  true // Убедитесь, что метод возвращает true при ручной навигации  }  } } |

# Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 1-6)

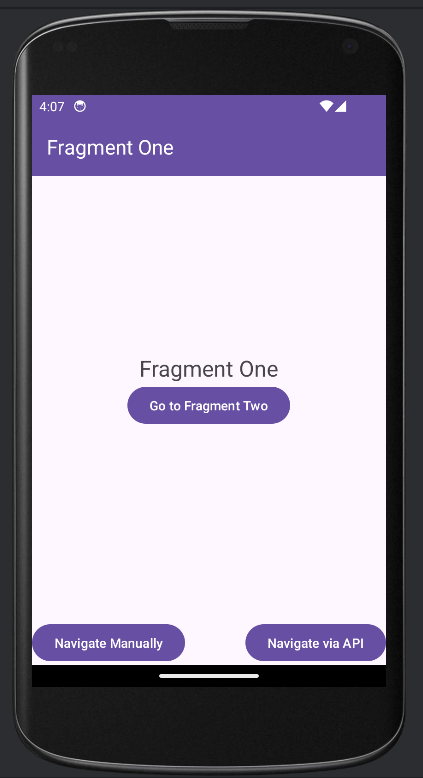


Рисунок 1. Запуск приложения



Рисунок 2. Кнопки управления фрагментами (вручную/через API)

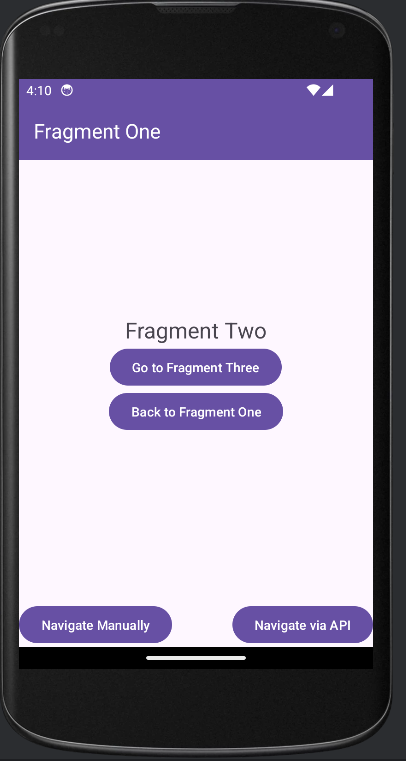


Рисунок 3. Проверка выполнения перехода между фрагментами вручную (режим Navigate Manually)

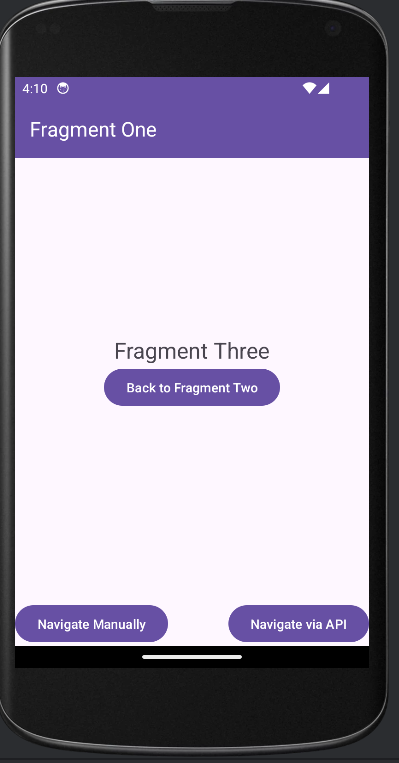


Рисунок 4. Проверка выполнения перехода между фрагментами вручную (режим Navigate Manually)

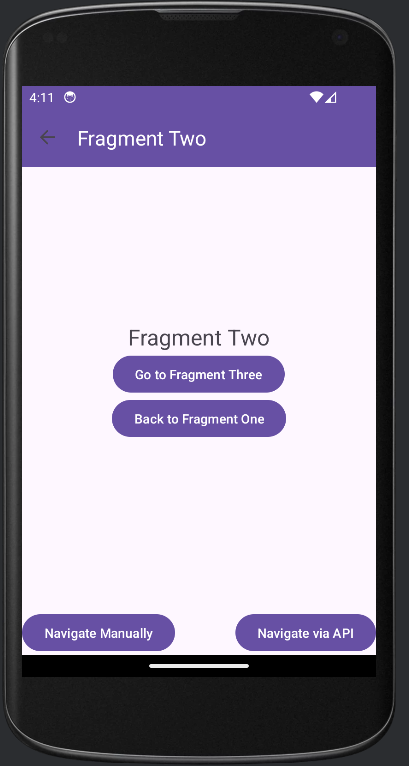


Рисунок 5. Проверка выполнения перехода между фрагментами через API (режим Navigate via API)

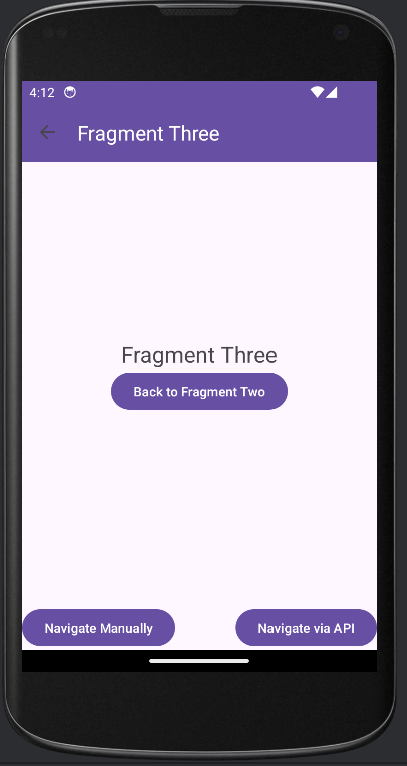


Рисунок 6. Проверка выполнения перехода между фрагментами через API (режим Navigate via API)

# Вывод

В процессе выполнения работы мы создали приложение, которое позволяет пользователю переходить между фрагментами вручную или через API