|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №1-12**

по дисциплине «Разработка мобильных приложений на языке Котлин»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-11-22 | Гришин А.В. |
| **Проверил:** | Преподаватель Степанов. П.В, |

2024 г.

**Оглавление**

[Практическая работа №1 3](#_Toc185104299)

[**Ссылка на Github** 3](#_Toc185104300)

[Практическая работа №2 5](#_Toc185104301)

[**Ссылка на Github** 6](#_Toc185104302)

[Практическая работа №3 10](#_Toc185104303)

[**Ссылка на Github** 10](#_Toc185104304)

[Практическая работа №4 15](#_Toc185104305)

[**Ссылка на Github** 15](#_Toc185104306)

[Практическая работа №5-6 23](#_Toc185104307)

[**Ссылка на Github** 23](#_Toc185104308)

[Практическая работа №7-8 32](#_Toc185104309)

[**Ссылка на Github** 32](#_Toc185104310)

[Практическая работа №9 39](#_Toc185104311)

[**Ссылка на Github** 39](#_Toc185104312)

[Практическая работа №10 42](#_Toc185104313)

[**Ссылка на Github** 42](#_Toc185104314)

[Практическая работа №11 50](#_Toc185104315)

[**Ссылка на Github** 50](#_Toc185104316)

[Практическая работа №12 59](#_Toc185104317)

[**Ссылка на Github** 59](#_Toc185104318)

**Практическая работа №1**

**Задание:**

Используя знания о переменных, циклах, условных операторах, классах, методах и конструкторах в Kotlin создать приложение, которое помогает пользователю отслеживать его личные расходы. Программа должна отвечать следующим требованиям:

1. Наличие класса, содержащего информацию о расходах (сумма расхода, категория, дата). Класс должен содержать метод, выводящий информацию о конкретном расходе.
2. Наличие класса, содержащего информацию о списке всех расходов. Класс должен содержать метод добавления нового расхода в список, метод вывода всех расходов, а также метод подсчета суммы всех расходов по каждой категории.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac1>

*Листинг 1 – Expense*

|  |
| --- |
| import java.time.LocalDate  class Expense {  val expenseAmount: Double  val category: String  val date: LocalDate  constructor(expenseAmount: Double, category: String, date: LocalDate) {  this.expenseAmount = expenseAmount  this.category = category  this.date = date  }  fun displayInfo() {  println("Сумма расхода: $expenseAmount\nКатегория: $category\nДата: $date")  }  } |

*Листинг 2 – ExpensesList*

|  |
| --- |
| class ExpensesList {  private val expensesList: MutableList<Expense> = mutableListOf()  fun addExpense(expense: Expense) {  expensesList.add(expense)  println("Расход добавлен: ${expense.category} - ${expense.expenseAmount}")  println("---")  }  fun printExpenses() {  if (expensesList.isEmpty()) {  println("Список пуст")  println("---")  } else {  println("Список расходов:")  expensesList.forEach { it.displayInfo() }  println("---")  }  }  private fun getTotalByCategory(): Map<String, Double> {  return expensesList.groupBy { it.category }  .mapValues { entry -> entry.value.sumOf { it.expenseAmount } }  }  fun printTotalByCategory() {  val totals = getTotalByCategory()  if (totals.isEmpty()) {  println("Расходы отсутствуют.")  println("---")  } else {  println("Сумма расходов по категориям:")  totals.forEach { (category, total) ->  println("Категория: $category, Сумма: $total")  }  println("---")  }  }  } |

*Листинг 3 – Main*

|  |
| --- |
| import java.time.LocalDate  fun main() {  val expensesList = ExpensesList()  expensesList.addExpense(Expense(  208.0, "Фастфуд", LocalDate.of(2024, 7, 31)  ))  expensesList.addExpense(Expense(  60.0, "Проезд в транспорте", LocalDate.of(2023, 8, 31)  ))  expensesList.addExpense(Expense(  1325.99, "Проезд в транспорте", LocalDate.of(2022, 8, 31)  ))  expensesList.addExpense(Expense(  327.0, "Фастфуд", LocalDate.of(2023, 7, 31)  ))  expensesList.printExpenses()  expensesList.printTotalByCategory()  } |

**Практическая работа №2**

**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.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac2>

*Листинг 4 – MainActivity*

|  |
| --- |
| package com.rarmash.prac2.ui  import android.os.Bundle  import androidx.appcompat.app.AppCompatActivity  import androidx.fragment.app.FragmentManager  import androidx.fragment.app.FragmentTransaction  import com.rarmash.prac2.R  class MainActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  if (savedInstanceState == null) {  val fragment1: FirstFragment = FirstFragment()  val fragmentManager: FragmentManager = supportFragmentManager  val fragmentTransaction: FragmentTransaction = fragmentManager.beginTransaction()  fragmentTransaction.replace(R.id.fragment\_container, fragment1)  fragmentTransaction.commit()  }  }  } |

*Листинг 5 – FirstFragment*

|  |
| --- |
| package com.rarmash.prac2.ui  import android.os.Bundle  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import androidx.navigation.fragment.findNavController  import com.rarmash.prac2.R  class FirstFragment : Fragment() {  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_1, container, false)  val buttonNext: Button = view.findViewById(R.id.btn\_next)  // Ручное управление  buttonNext.setOnClickListener {  val fragment2 = SecondFragment()  parentFragmentManager.beginTransaction()  .replace(R.id.fragment\_container, fragment2)  .addToBackStack(null)  .commit()  }  // Навигация с помощью API  // buttonNext.setOnClickListener {  // findNavController().navigate(R.id.action\_fragment1\_to\_fragment2)  // }  return view  }  } |

*Листинг 6 – SecondFragment*

|  |
| --- |
| package com.rarmash.prac2.ui  import android.os.Bundle  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import androidx.navigation.fragment.findNavController  import com.rarmash.prac2.R  class SecondFragment : Fragment() {  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_2, container, false)  // Ручное управление  val buttonNext: Button = view.findViewById(R.id.btn\_next)  val buttonBack: Button = view.findViewById(R.id.btn\_back)  buttonNext.setOnClickListener {  val fragment3 = ThirdFragment()  parentFragmentManager.beginTransaction()  .replace(R.id.fragment\_container, fragment3)  .addToBackStack(null)  .commit()  }  buttonBack.setOnClickListener {  parentFragmentManager.popBackStack()  }  // API  // buttonNext.setOnClickListener {  // findNavController().navigate(R.id.action\_fragment2\_to\_fragment3)  // }  //  // buttonBack.setOnClickListener {  // findNavController().popBackStack()  // }  return view  }  } |

*Листинг 7 – ThirdFragment*

|  |
| --- |
| package com.rarmash.prac2.ui  import android.os.Bundle  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import androidx.navigation.fragment.findNavController  import com.rarmash.prac2.R  class ThirdFragment : Fragment() {  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_3, container, false)  // Ручное управление  val buttonBack: Button = view.findViewById(R.id.btn\_back)  buttonBack.setOnClickListener {  parentFragmentManager.popBackStack()  }  // API  // buttonBack.setOnClickListener {  // findNavController().popBackStack()  // }  return view  }  } |

**Практическая работа №3**

**Задание:**

Используя Android Architecture Components преобразовать структуру проекта из практической работы №2.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac3>

*Листинг 8 – MainActivity*

|  |
| --- |
| package com.rarmash.prac3.ui  import android.os.Bundle  import androidx.appcompat.app.AppCompatActivity  import androidx.fragment.app.FragmentManager  import androidx.fragment.app.FragmentTransaction  import com.rarmash.prac3.R  class MainActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  // if (savedInstanceState == null) {  // val fragment1: FirstFragment = FirstFragment()  // val fragmentManager: FragmentManager = supportFragmentManager  // val fragmentTransaction: FragmentTransaction = fragmentManager.beginTransaction()  // fragmentTransaction.replace(R.id.fragment\_container, fragment1)  // fragmentTransaction.commit()  // }  }  } |

*Листинг 9 – FirstFragment*

|  |
| --- |
| package com.rarmash.prac3.ui  import android.os.Bundle  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import androidx.lifecycle.Observer  import androidx.lifecycle.ViewModelProvider  import androidx.navigation.fragment.findNavController  import com.rarmash.prac3.R  import com.rarmash.prac3.databinding.Fragment1Binding  import com.rarmash.prac3.viewmodel.FirstFragmentViewModel  class FirstFragment : Fragment() {  private lateinit var viewModel: FirstFragmentViewModel  private lateinit var binding: Fragment1Binding  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  binding = Fragment1Binding.inflate(inflater, container, false)  viewModel = ViewModelProvider(this).get(FirstFragmentViewModel::class.java)  binding.imageView.setOnClickListener {  viewModel.changeImage()  }  viewModel.\_currentImage.observe(viewLifecycleOwner,  Observer { data ->  binding.imageView.setImageResource(data)  })  // Ручное управление  // buttonNext.setOnClickListener {  // val fragment2 = Fragment2()  // parentFragmentManager.beginTransaction()  // .replace(R.id.fragment\_container, fragment2)  // .addToBackStack(null)  // .commit()  // }  // Навигация с помощью API  binding.btnNext.setOnClickListener {  findNavController().navigate(R.id.action\_fragment1\_to\_fragment2)  }  return binding.root  }  } |

*Листинг 10 – SecondFragment*

|  |
| --- |
| package com.rarmash.prac3.ui  import android.os.Bundle  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import androidx.navigation.fragment.findNavController  import com.rarmash.prac3.R  class SecondFragment : Fragment() {  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_2, container, false)  // Ручное управление  val buttonNext: Button = view.findViewById(R.id.btn\_next)  val buttonBack: Button = view.findViewById(R.id.btn\_back)  // buttonNext.setOnClickListener {  // val fragment3 = ThirdFragment()  // parentFragmentManager.beginTransaction()  // .replace(R.id.fragment\_container, fragment3)  // .addToBackStack(null)  // .commit()  // }  // buttonBack.setOnClickListener {  // parentFragmentManager.popBackStack()  // }  // API  buttonNext.setOnClickListener {  findNavController().navigate(R.id.action\_fragment2\_to\_fragment3)  }  buttonBack.setOnClickListener {  findNavController().popBackStack()  }  return view  }  } |

*Листинг 11 – ThirdFragment*

|  |
| --- |
| package com.rarmash.prac3.ui  import android.os.Bundle  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.Button  import androidx.navigation.fragment.findNavController  import com.rarmash.prac3.R  class ThirdFragment : Fragment() {  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val view = inflater.inflate(R.layout.fragment\_3, container, false)  // Ручное управление  val buttonBack: Button = view.findViewById(R.id.btn\_back)  // buttonBack.setOnClickListener {  // parentFragmentManager.popBackStack()  // }  // API  buttonBack.setOnClickListener {  findNavController().popBackStack()  }  return view  }  } |

*Листинг 12 – FirstFragmentViewModel*

|  |
| --- |
| package com.rarmash.prac3.viewmodel  import androidx.lifecycle.MutableLiveData  import androidx.lifecycle.ViewModel  import com.rarmash.prac3.R  class FirstFragmentViewModel: ViewModel() {  val \_currentImage = MutableLiveData<Int>(R.drawable.marcus\_fenix)  private var imageChanged = false  fun changeImage() {  if (!imageChanged) {  \_currentImage.value = R.drawable.forza  imageChanged = true  } else {  \_currentImage.value = R.drawable.marcus\_fenix  imageChanged = false  }  }  } |

**Практическая работа №4**

**Задание:**

Реализовать приложение, имеющее два экрана: «Камера» и «Список»:

- на экране «Камера» необходимо реализовать функционал просмотра камеры на экране, а также кнопку фотографирования, которая при нажатии будет сохранять в файл "date", находящийся в папке "photos" время и дату сделанной фотографии;

- на экране «Список» реализовать с помощью RecyclerView список, отображающий данные с файла "date" в хронологическом порядке.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac4>

*Листинг 13 – CameraFragment*

|  |
| --- |
| package com.rarmash.prac4  import android.content.pm.PackageManager  import android.os.Bundle  import android.os.Environment  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.Manifest  import androidx.activity.result.contract.ActivityResultContracts  import androidx.camera.core.CameraSelector  import androidx.camera.core.Preview  import androidx.camera.lifecycle.ProcessCameraProvider  import androidx.camera.view.PreviewView  import androidx.core.content.ContextCompat  import androidx.navigation.fragment.findNavController  import com.rarmash.prac4.databinding.FragmentCameraBinding  import java.io.File  import java.io.FileOutputStream  import java.text.SimpleDateFormat  import java.util.Date  import java.util.Locale  import java.util.concurrent.ExecutorService  import java.util.concurrent.Executors  class CameraFragment : Fragment() {  private lateinit var binding: FragmentCameraBinding  private lateinit var cameraExecutor: ExecutorService // Камера в отдельном потоке  private lateinit var viewFinder: PreviewView // Изображение  private val requestPermissionLauncher =  registerForActivityResult(ActivityResultContracts.RequestMultiplePermissions()) { permissions ->  if (permissions.all { it.value }) {  startCamera()  } else {  requireActivity().finish()  }  }  companion object {  private val REQUIRED\_PERMISSIONS = arrayOf(Manifest.permission.CAMERA)  }  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  binding = FragmentCameraBinding.inflate(inflater, container, false)  binding.btnPht.setOnClickListener {  findNavController().navigate(R.id.action\_fragmentCamera\_to\_fragmentPhotos)  }  cameraExecutor = Executors.newSingleThreadExecutor()  viewFinder = binding.previewView  if (allPermissionsGranted()) {  startCamera()  } else {  requestPermissionLauncher.launch(REQUIRED\_PERMISSIONS)  }  binding.btnTake.setOnClickListener {  saveDateTimeToFile()  }  return binding.root  }  private fun startCamera() {  val cameraProviderFuture = ProcessCameraProvider.getInstance(requireContext()) // Подвязка камеры к ЖЦ активности  cameraProviderFuture.addListener({  val cameraProvider: ProcessCameraProvider = cameraProviderFuture.get()  val preview = Preview.Builder().build().also {  it.setSurfaceProvider(viewFinder.surfaceProvider)  }  // Выбор камеры  val cameraSelector = CameraSelector.DEFAULT\_FRONT\_CAMERA  try {  cameraProvider.unbindAll()  cameraProvider.bindToLifecycle(this, cameraSelector, preview)  } catch (exc: Exception) {  exc.printStackTrace()  }  }, ContextCompat.getMainExecutor(requireContext()))  }  private fun allPermissionsGranted() = REQUIRED\_PERMISSIONS.all { permission ->  ContextCompat.checkSelfPermission(  requireContext(),  permission  ) == PackageManager.PERMISSION\_GRANTED  }  private fun saveDateTimeToFile() {  val dateFormat = SimpleDateFormat("dd.MM.yyyy HH:mm:ss", Locale.getDefault())  val currentDateTime = dateFormat.format(Date())  val photosDir =  File(requireContext().getExternalFilesDir(Environment.DIRECTORY\_PICTURES), "photos")  if (!photosDir.exists()) {  photosDir.mkdirs()  }  val dateFile = File(photosDir, "photos.txt")  try {  FileOutputStream(dateFile, true).use { outputStream ->  outputStream.write("$currentDateTime\n".toByteArray())  }  } catch (e: Exception) {  e.printStackTrace()  }  }  override fun onDestroy() {  super.onDestroy()  cameraExecutor.shutdown()  }  } |

*Листинг 14 – MainActivity*

|  |
| --- |
| package com.rarmash.prac4  import android.os.Bundle  import androidx.appcompat.widget.Toolbar  import androidx.appcompat.app.AppCompatActivity  import androidx.navigation.NavController  import androidx.navigation.fragment.NavHostFragment  import androidx.navigation.ui.setupActionBarWithNavController  class MainActivity : AppCompatActivity() {  private lateinit var navController: NavController  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  val toolbar: Toolbar = findViewById(R.id.toolbar)  setSupportActionBar(toolbar)  val navHostFragment = supportFragmentManager  .findFragmentById(R.id.nav\_host\_fragment) as NavHostFragment  navController = navHostFragment.navController  setupActionBarWithNavController(navController)  }  override fun onSupportNavigateUp(): Boolean {  return navController.navigateUp() || super.onSupportNavigateUp()  }  } |

*Листинг 15 – MyAdapter*

|  |
| --- |
| package com.rarmash.prac4  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.TextView  import androidx.recyclerview.widget.RecyclerView  class MyAdapter(private val data: List<String>) :  RecyclerView.Adapter<MyAdapter.ViewHolder>() {  class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {  val textView: TextView =  view.findViewById(R.id.itemTextView)  }  override fun onCreateViewHolder(  parent: ViewGroup, viewType:  Int  ): ViewHolder {  val view =  LayoutInflater.from(parent.context).inflate(  R.layout.item\_layout,  parent, false  )  return ViewHolder(view)  }  override fun onBindViewHolder(  holder: ViewHolder, position: Int  ) {  val item = data[position]  holder.textView.text = item  }  override fun getItemCount() = data.size  } |

*Листинг 16 – PhotosFragment*

|  |
| --- |
| package com.rarmash.prac4  import android.os.Bundle  import android.os.Environment  import androidx.fragment.app.Fragment  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import androidx.recyclerview.widget.LinearLayoutManager  import com.rarmash.prac4.databinding.FragmentPhotosBinding  import java.io.File  import java.io.FileInputStream  import java.io.InputStreamReader  class PhotosFragment : Fragment() {  private lateinit var binding: FragmentPhotosBinding  override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View {  binding = FragmentPhotosBinding.inflate(inflater, container, false)  val data = readDateTimeFromFile()  val adapter = MyAdapter(data)  binding.recyclerView.adapter = adapter  binding.recyclerView.layoutManager = LinearLayoutManager(requireContext())  return binding.root  }  private fun readDateTimeFromFile(): List<String> {  val photosDir =  File(requireContext().getExternalFilesDir(Environment.DIRECTORY\_PICTURES), "photos")  val dateFile = File(photosDir, "photos.txt")  val data = mutableListOf<String>()  try {  FileInputStream(dateFile).use { inputStream ->  InputStreamReader(inputStream).use { reader ->  reader.forEachLine { line ->  data.add(line)  }  }  }  } catch (e: Exception) {  e.printStackTrace()  }  return data  }  } |

**Практическая работа №5-6**

**Задание 5 работы:**

Разработать приложение с функциями получения и отображения данных с внешнего API.

1. Получение данных реализовать при помощи библиотеки Retrofit.
2. Полученные данные сохранять в локальную базу данных.
3. На отдельном экране реализовать отображение данных, сохраненных в базу данных.

**Задание 6 работы:**

При помощи библиотеки Dagger, Hilt или Koin реализовать Dependency Injection в приложении из практической работы №5.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac5-6>

*Листинг 17 – App*

|  |
| --- |
| package com.rarmash.prac5  import android.app.Application  import dagger.hilt.android.HiltAndroidApp  @HiltAndroidApp  class App : Application() {  } |

*Листинг 18 – Cart*

|  |
| --- |
| package com.rarmash.prac5  import androidx.room.Entity  import androidx.room.PrimaryKey  @Entity  data class Cart(  @PrimaryKey val id: Int,  val products: List<String>,  val total: Double,  val discountedTotal: Double,  val userId: Int,  val totalProducts: Int,  val totalQuantity: Int  ) |

*Листинг 19 – CartAdapter*

|  |
| --- |
| package com.rarmash.prac5  import android.annotation.SuppressLint  import android.view.LayoutInflater  import android.view.View  import android.view.ViewGroup  import android.widget.TextView  import androidx.recyclerview.widget.RecyclerView  class CartAdapter(private var carts: List<Cart>) : RecyclerView.Adapter<CartAdapter.CartViewHolder>() {  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): CartViewHolder {  val view = LayoutInflater.from(parent.context).inflate(R.layout.item\_carts, parent, false)  return CartViewHolder(view)  }  override fun onBindViewHolder(holder: CartViewHolder, position: Int) {  val cart = carts[position]  holder.bind(cart)  }  override fun getItemCount(): Int {  return carts.size  }  @SuppressLint("NotifyDataSetChanged")  fun updateCarts(newCarts: List<Cart>) {  this.carts = newCarts  notifyDataSetChanged()  }  class CartViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  private val cartTotalDiscounted: TextView = itemView.findViewById(R.id.cartTotalDiscounted)  private val cartUserId: TextView = itemView.findViewById(R.id.cartUserId)  fun bind(cart: Cart) {  cartTotalDiscounted.text = cart.discountedTotal.toString()  cartUserId.text = cart.userId.toString()  }  }  } |

*Листинг 20 – CartApi*

|  |
| --- |
| package com.rarmash.prac5  import retrofit2.http.GET  import retrofit2.http.Query  interface CartApi {  @GET("carts")  suspend fun getCarts(  @Query("limit") limit: Int,  ): CartResponse  } |

*Листинг 21 – CartDao*

|  |
| --- |
| package com.rarmash.prac5  import androidx.lifecycle.LiveData  import androidx.room.Dao  import androidx.room.Insert  import androidx.room.OnConflictStrategy  import androidx.room.Query  @Dao  interface CartDao {  @Insert(onConflict = OnConflictStrategy.REPLACE)  suspend fun insertCarts(carts: List<Cart>)  @Query("SELECT \* FROM cart")  fun getAllCarts(): LiveData<List<Cart>>  } |

*Листинг 22 – CartRepository*

|  |
| --- |
| package com.rarmash.prac5  import androidx.lifecycle.LiveData  import javax.inject.Inject  class CartRepository @Inject constructor(private val cartApi: CartApi, private val cartDao: CartDao) {  suspend fun fetchAndSaveCarts() {  val carts = cartApi.getCarts(3).carts  cartDao.insertCarts(carts)  }  fun getAllCarts(): LiveData<List<Cart>> {  return cartDao.getAllCarts()  }  } |

*Листинг 23 – CartResponse*

|  |
| --- |
| package com.rarmash.prac5  data class CartResponse(  val carts: List<Cart>,  val total: Int,  val skip: Int,  val limit: Int  ) |

*Листинг 24 – CartViewModel*

|  |
| --- |
| package com.rarmash.prac5  import androidx.lifecycle.LiveData  import androidx.lifecycle.ViewModel  import androidx.lifecycle.viewModelScope  import dagger.hilt.android.lifecycle.HiltViewModel  import kotlinx.coroutines.launch  import javax.inject.Inject  @HiltViewModel  class CartViewModel @Inject constructor(private val repository: CartRepository) : ViewModel() {  val carts: LiveData<List<Cart>> = repository.getAllCarts()  fun fetchCarts() {  viewModelScope.launch {  repository.fetchAndSaveCarts()  }  }  } |

*Листинг 25 – CartViewModelFactory*

|  |
| --- |
| package com.rarmash.prac5  import androidx.lifecycle.ViewModel  import androidx.lifecycle.ViewModelProvider  class CartViewModelFactory(private val repository: CartRepository) : ViewModelProvider.Factory {  @Suppress("UNCHECKED\_CAST")  override fun <T : ViewModel> create(modelClass: Class<T>): T {  if (modelClass.isAssignableFrom(CartViewModel::class.java)) {  return CartViewModel(repository) as T  }  throw IllegalArgumentException("Unknown ViewModel class")  }  } |

*Листинг 26 – CartsDatabase*

|  |
| --- |
| package com.rarmash.prac5  import androidx.room.Database  import androidx.room.RoomDatabase  import androidx.room.TypeConverters  @Database(entities = [Cart::class], version = 1)  @TypeConverters(Converters::class)  abstract class CartsDatabase : RoomDatabase() {  abstract fun cartDao(): CartDao  } |

*Листинг 27 – Converters*

|  |
| --- |
| package com.rarmash.prac5  import androidx.room.TypeConverter  import com.google.gson.Gson  import com.google.gson.reflect.TypeToken  import java.lang.reflect.Type  class Converters {  @TypeConverter  fun fromStringList(value: List<String>?): String {  val gson = Gson()  val type: Type = object : TypeToken<List<String>>() {}.type  return gson.toJson(value, type)  }  @TypeConverter  fun toStringList(value: String): List<String>? {  val gson = Gson()  val type: Type = object : TypeToken<List<String>>() {}.type  return gson.fromJson(value, type)  }  } |

*Листинг 28 – MainActivity*

|  |
| --- |
| package com.rarmash.prac5  import android.os.Bundle  import androidx.activity.viewModels  import androidx.appcompat.app.AppCompatActivity  import androidx.recyclerview.widget.LinearLayoutManager  import androidx.recyclerview.widget.RecyclerView  import dagger.hilt.android.AndroidEntryPoint  @AndroidEntryPoint  class MainActivity : AppCompatActivity() {  private lateinit var recyclerView: RecyclerView  private lateinit var adapter: CartAdapter  private val cartViewModel: CartViewModel by viewModels()  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  recyclerView = findViewById(R.id.recyclerView)  recyclerView.layoutManager = LinearLayoutManager(this)  adapter = CartAdapter(emptyList())  recyclerView.adapter = adapter  cartViewModel.carts.observe(this) { carts ->  adapter.updateCarts(carts)  }  cartViewModel.fetchCarts()  }  } |

*Листинг 29 – RetrofitModule*

|  |
| --- |
| package com.rarmash.prac5  import dagger.Module  import dagger.Provides  import dagger.hilt.InstallIn  import dagger.hilt.components.SingletonComponent  import retrofit2.Retrofit  import retrofit2.converter.gson.GsonConverterFactory  import javax.inject.Singleton  @Module  @InstallIn(SingletonComponent::class)  object RetrofitModule {  @Provides  @Singleton  fun providesRetrofit(): Retrofit {  return Retrofit.Builder()  .baseUrl("https://dummyjson.com/")  .addConverterFactory(GsonConverterFactory.create())  .build()  }  @Provides  @Singleton  fun providesCartApi(retrofit:Retrofit): CartApi {  return retrofit.create(CartApi::class.java)  }  } |

*Листинг 30 – RoomModule*

|  |
| --- |
| package com.rarmash.prac5  import android.app.Application  import androidx.room.Room  import dagger.Module  import dagger.Provides  import dagger.hilt.InstallIn  import dagger.hilt.components.SingletonComponent  import javax.inject.Singleton  @Module  @InstallIn(SingletonComponent::class)  object RoomModule {  @Provides  @Singleton  fun providesCartsDb(app: Application): CartsDatabase{  return Room.databaseBuilder(  app,  CartsDatabase::class.java, "cart-database"  ).build()  }  @Provides  fun provideCartDao(database: CartsDatabase): CartDao {  return database.cartDao()  }  } |

**Практическая работа №7-8**

**Задание 7 работы:**

Разработать приложение, соответствующее следующим условиям:

1. Должна быть предусмотрена возможность ввода ссылки.
2. Загрузка изображения должна осуществляться при нажатии на кнопку.
3. Нажатие на кнопку должно запускать 2 потока:
   1. В потоке Network необходимо выполнить загрузку изображения.
   2. В потоке Disk сохранить загруженное изображение во внутренней памяти устройства.

**Задание 8 работы:**

Реализовать в приложении из практической работы №7:

1. 5 модульных тестов, демонстрирующих работу функционала приложения.
2. 5 тестов элементов пользовательского интерфейса.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac7-8>

*Листинг 31 – MainActivity*

|  |
| --- |
| package com.rarmash.prac7  import android.content.Context  import android.graphics.Bitmap  import android.os.Bundle  import android.os.Environment  import android.widget.Button  import android.widget.EditText  import android.widget.ImageView  import android.widget.Toast  import androidx.appcompat.app.AppCompatActivity  import com.rarmash.prac7.NetworkUtilities  import kotlinx.coroutines.CoroutineScope  import kotlinx.coroutines.Dispatchers  import kotlinx.coroutines.launch  import kotlinx.coroutines.withContext  import java.io.File  import java.io.FileOutputStream  class MainActivity : AppCompatActivity() {  val network = NetworkUtilities()  lateinit var editTextURL: EditText  lateinit var buttonDownload: Button  lateinit var imageView: ImageView  public override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  editTextURL = findViewById(R.id.editTextUrl)  buttonDownload = findViewById(R.id.buttonDownload)  imageView = findViewById(R.id.imageView)  buttonDownload.setOnClickListener {  val imageURL = editTextURL.text.toString()  downloadAndSaveImage(imageURL, this)  }  }  private fun downloadAndSaveImage(imageUrl: String, context: Context) {  CoroutineScope(Dispatchers.Main).launch {  val bitmapDeferred = network.downloadImage(imageUrl)  val bitmap = bitmapDeferred.await()  if (bitmap != null) {  imageView.setImageBitmap(bitmap)  saveImageToDisk(bitmap, context)  } else {  Toast.makeText(context, "Ошибка загрузки изображения", Toast.LENGTH\_SHORT).show()  }  }  }  fun saveImageToDisk(bitmap: Bitmap?, context: Context) {  CoroutineScope(Dispatchers.IO).launch {  try {  val file = File(  context.getExternalFilesDir(Environment.DIRECTORY\_PICTURES),  "downloaded\_image.jpg"  )  FileOutputStream(file).use { outputStream ->  if (bitmap != null) {  bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)  }  outputStream.flush()  }  withContext(Dispatchers.Main) {  Toast.makeText(  context,  "Изображение сохранено: ${file.path}",  Toast.LENGTH\_SHORT  ).show()  }  } catch (e: Exception) {  e.printStackTrace()  withContext(Dispatchers.Main) {  Toast.makeText(context, "Ошибка сохранения изображения", Toast.LENGTH\_SHORT)  .show()  }  }  }  }  } |

*Листинг 32 – NetworkUtilities*

|  |
| --- |
| package com.rarmash.prac7  import android.graphics.Bitmap  import android.graphics.BitmapFactory  import kotlinx.coroutines.CoroutineScope  import kotlinx.coroutines.Deferred  import kotlinx.coroutines.Dispatchers  import kotlinx.coroutines.async  import java.net.URL  class NetworkUtilities {  fun downloadImage(imageUrl: String): Deferred<Bitmap?> {  return CoroutineScope(Dispatchers.IO).async {  try {  val url = URL(imageUrl)  val connection = url.openConnection()  connection.doInput = true  connection.connect()  val input = connection.getInputStream()  BitmapFactory.decodeStream(input)  } catch (e: Exception) {  e.printStackTrace()  null  }  }  }  } |

*Листинг 33 – MainActivityModuleTest*

|  |
| --- |
| package com.rarmash.prac7  import org.junit.runners.JUnit4  import junit.framework.TestCase.assertNotNull  import org.junit.Test  import org.junit.runner.RunWith  @RunWith(JUnit4::class)  class MainActivityModuleTest {  @Test  fun testImageIsLoaded() {  val network = NetworkUtilities()  val imageUrl = "https://i.imgur.com/VSaDVhp.jpeg"  val bitmap = network.downloadImage(imageUrl)  assertNotNull(bitmap)  }  } |

*Листинг 34 – MainActivityUITest*

|  |
| --- |
| package com.rarmash.prac7  import android.os.Handler  import android.os.Looper  import androidx.test.espresso.Espresso.onView  import androidx.test.espresso.action.ViewActions.click  import androidx.test.espresso.action.ViewActions.closeSoftKeyboard  import androidx.test.espresso.action.ViewActions.typeText  import androidx.test.espresso.assertion.ViewAssertions.matches  import androidx.test.espresso.matcher.RootMatchers  import androidx.test.espresso.matcher.ViewMatchers.isDisplayed  import androidx.test.espresso.matcher.ViewMatchers.withId  import androidx.test.espresso.matcher.ViewMatchers.withText  import androidx.test.ext.junit.rules.ActivityScenarioRule  import androidx.test.ext.junit.runners.AndroidJUnit4  import com.rarmash.prac7.MainActivity  import org.hamcrest.Matchers.not  import org.junit.Rule  import org.junit.Test  import org.junit.runner.RunWith  @RunWith(AndroidJUnit4::class)  class MainActivityUITest {  @get:Rule  val activityRule = ActivityScenarioRule(MainActivity::class.java)  @Test  fun checkUiElementsVisibility() {  onView(withId(R.id.editTextUrl)).check(matches(isDisplayed()))  onView(withId(R.id.buttonDownload)).check(matches(isDisplayed()))  }  @Test  fun checkEditTextInitialState() {  onView(withId(R.id.editTextUrl)).check(matches(withText("")))  }  @Test  fun checkButtonText() {  onView(withId(R.id.buttonDownload)).check(matches(withText("Загрузить изображение")))  }  @Test  fun checkButtonClick() {  onView(withId(R.id.editTextUrl)).perform(  typeText("https://example.com/invalid.jpg"),  closeSoftKeyboard()  )  onView(withId(R.id.buttonDownload)).perform(click())  Handler(Looper.getMainLooper()).postDelayed({  activityRule.scenario.onActivity { activity ->  onView(withText("Ошибка загрузки изображения"))  .inRoot(RootMatchers.withDecorView(not(activity.window.decorView)))  .check(matches(isDisplayed()))  }  }, 1000)  }  @Test  fun checkImageViewDisplayAfterDownload() {  onView(withId(R.id.editTextUrl)).perform(  typeText("https://i.imgur.com/VSaDVhp.jpeg"),  closeSoftKeyboard()  )  onView(withId(R.id.buttonDownload)).perform(click())  Handler(Looper.getMainLooper()).postDelayed({  onView(withId(R.id.imageView)).check(matches(isDisplayed()))  }, 3000)  }  } |

**Практическая работа №9**

**Задание:**

1. Реализовать интерфейс приложения, состоящий из элемента Text, на котором должно быть указано ФИО и номер группы студента.
2. Реализовать собственный набор стилей к этому приложению: тему, шрифты, размеры и т.д.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac9>

*Листинг 35 – MainActivity*

|  |
| --- |
| package com.rarmash.prac9  import android.os.Bundle  import androidx.activity.ComponentActivity  import androidx.activity.compose.setContent  import androidx.activity.enableEdgeToEdge  import androidx.compose.foundation.layout.fillMaxSize  import androidx.compose.foundation.layout.padding  import androidx.compose.material3.Scaffold  import androidx.compose.material3.Text  import androidx.compose.runtime.Composable  import androidx.compose.ui.Modifier  import com.rarmash.prac9.ui.theme.Prac9Theme  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  enableEdgeToEdge()  setContent {  Prac9Theme {  Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->  Greeting(  name = "Гришин А.В. ИКБО-11-22",  modifier = Modifier.padding(innerPadding)  )  }  }  }  }  }  @Composable  fun Greeting(name: String, modifier: Modifier = Modifier) {  Text(  text = "$name",  modifier = modifier  )  } |

*Листинг 36 – Color*

|  |
| --- |
| package com.rarmash.prac9.ui.theme  import androidx.compose.ui.graphics.Color  val PBlue = Color(0xFF6600FF) |

*Листинг 37 – Theme*

|  |
| --- |
| package com.rarmash.prac9.ui.theme  import androidx.compose.material3.MaterialTheme  import androidx.compose.material3.lightColorScheme  import androidx.compose.runtime.Composable  import androidx.compose.ui.graphics.Color  private val ColorScheme = lightColorScheme(  primary = Color.Blue,  secondary = Color.Black,  tertiary = Color.Cyan,  background = PBlue,  surface = PBlue,  onPrimary = Color.White,  onSecondary = Color.White,  onTertiary = Color.White  )  @Composable  fun Prac9Theme(  content: @Composable () -> Unit  ) {  val colorScheme = ColorScheme  MaterialTheme(  colorScheme = colorScheme,  typography = Typography,  content = content  )  } |

*Листинг 38 – Type*

|  |
| --- |
| package com.rarmash.prac9.ui.theme  import androidx.compose.material3.Typography  import androidx.compose.ui.text.TextStyle  import androidx.compose.ui.text.font.FontFamily  import androidx.compose.ui.text.font.FontWeight  import androidx.compose.ui.unit.sp  val Typography = Typography(  bodyLarge = TextStyle(  fontFamily = FontFamily.SansSerif,  fontWeight = FontWeight.Bold,  fontSize = 15.sp,  lineHeight = 23.sp,  letterSpacing = 0.6.sp  )  ) |

**Практическая работа №10**

**Задание:**

Реализовать приложение из практической работы №7 средствами Jetpack Compose, сохранив изначальное расположение элементов разметки, применив:

1. Изученные контейнеры компоновки (Column, Row, Box, Card).
2. Динамический список (LazyColumn или LazyRow на выбор).

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac10>

*Листинг 39 – MainActivity*

|  |
| --- |
| package com.rarmash.prac10  import com.rarmash.prac10.ui.theme.Prac10Theme  import android.graphics.Bitmap  import android.graphics.BitmapFactory  import android.os.Bundle  import android.os.Environment  import androidx.activity.ComponentActivity  import androidx.activity.compose.setContent  import androidx.compose.foundation.Image  import androidx.compose.foundation.layout.Column  import androidx.compose.foundation.layout.Spacer  import androidx.compose.foundation.layout.fillMaxSize  import androidx.compose.foundation.layout.fillMaxWidth  import androidx.compose.foundation.layout.height  import androidx.compose.foundation.layout.padding  import androidx.compose.material3.Button  import androidx.compose.material3.CircularProgressIndicator  import androidx.compose.material3.Surface  import androidx.compose.material3.Text  import androidx.compose.material3.TextField  import androidx.compose.runtime.Composable  import androidx.compose.runtime.MutableState  import androidx.compose.runtime.mutableStateOf  import androidx.compose.runtime.remember  import androidx.compose.ui.Alignment  import androidx.compose.ui.Modifier  import androidx.compose.ui.graphics.asImageBitmap  import androidx.compose.ui.unit.dp  import kotlinx.coroutines.CoroutineScope  import kotlinx.coroutines.Deferred  import kotlinx.coroutines.Dispatchers  import kotlinx.coroutines.async  import kotlinx.coroutines.launch  import kotlinx.coroutines.withContext  import java.io.File  import java.io.FileOutputStream  import java.net.URL  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContent {  Prac10Theme {  Surface(modifier = Modifier.fillMaxSize()) {  val imageUrlState = remember { mutableStateOf("") }  val bitmapState = remember { mutableStateOf<Bitmap?>(null) }  val isLoadingState = remember { mutableStateOf(false) }  val messageState = remember { mutableStateOf<String?>(null) }  ImageDownloaderScreen(  imageUrlState = imageUrlState,  bitmapState = bitmapState,  isLoadingState = isLoadingState,  messageState = messageState,  onDownloadClick = {  val imageUrl = imageUrlState.value  if (imageUrl.isNotBlank()) {  downloadAndSaveImage(  imageUrl = imageUrl,  context = this@MainActivity,  bitmapState = bitmapState,  isLoadingState = isLoadingState,  messageState = messageState  )  } else {  messageState.value = "Введите корректный URL!"  }  }  )  }  }  }  }  private fun downloadAndSaveImage(  imageUrl: String,  context: ComponentActivity,  bitmapState: MutableState<Bitmap?>,  isLoadingState: MutableState<Boolean>,  messageState: MutableState<String?>  ) {  CoroutineScope(Dispatchers.Main).launch {  isLoadingState.value = true  val bitmapDeferred = downloadImage(imageUrl)  val bitmap = bitmapDeferred.await()  isLoadingState.value = false  if (bitmap != null) {  bitmapState.value = bitmap  saveImageToDisk(bitmap, context, messageState)  } else {  messageState.value = "Ошибка загрузки изображения"  }  }  }  private fun downloadImage(imageUrl: String): Deferred<Bitmap?> {  return CoroutineScope(Dispatchers.IO).async {  try {  val url = URL(imageUrl)  val connection = url.openConnection()  connection.doInput = true  connection.connect()  val input = connection.getInputStream()  BitmapFactory.decodeStream(input)  } catch (e: Exception) {  e.printStackTrace()  null  }  }  }  private fun saveImageToDisk(  bitmap: Bitmap,  context: ComponentActivity,  messageState: MutableState<String?>  ) {  CoroutineScope(Dispatchers.IO).launch {  try {  val file = File(  context.getExternalFilesDir(Environment.DIRECTORY\_PICTURES),  "downloaded\_image.jpg"  )  FileOutputStream(file).use { outputStream ->  bitmap.compress(Bitmap.CompressFormat.JPEG, 100, outputStream)  outputStream.flush()  }  withContext(Dispatchers.Main) {  messageState.value = "Изображение сохранено: ${file.path}"  }  } catch (e: Exception) {  e.printStackTrace()  withContext(Dispatchers.Main) {  messageState.value = "Ошибка сохранения изображения"  }  }  }  }  }  @Composable  fun ImageDownloaderScreen(  imageUrlState: MutableState<String>,  bitmapState: MutableState<Bitmap?>,  isLoadingState: MutableState<Boolean>,  messageState: MutableState<String?>,  onDownloadClick: () -> Unit  ) {  Column(  modifier = Modifier.padding(16.dp),  horizontalAlignment = Alignment.CenterHorizontally  ) {  TextField(  value = imageUrlState.value,  onValueChange = { imageUrlState.value = it },  label = { Text("Введите URL изображения") },  modifier = Modifier.fillMaxWidth()  )  Spacer(modifier = Modifier.height(8.dp))  Button(  onClick = onDownloadClick,  modifier = Modifier.fillMaxWidth()  ) {  Text("Загрузить изображение")  }  Spacer(modifier = Modifier.height(16.dp))  if (isLoadingState.value) {  CircularProgressIndicator()  } else {  bitmapState.value?.let {  Image(  bitmap = it.asImageBitmap(),  contentDescription = "Загруженное изображение",  modifier = Modifier.fillMaxWidth()  )  }  }  Spacer(modifier = Modifier.height(8.dp))  messageState.value?.let {  Text(text = it)  }  }  } |

*Листинг 40 – Color*

|  |
| --- |
| package com.rarmash.prac10.ui.theme  import androidx.compose.ui.graphics.Color  val Purple80 = Color(0xFFD0BCFF)  val PurpleGrey80 = Color(0xFFCCC2DC)  val Pink80 = Color(0xFFEFB8C8)  val Purple40 = Color(0xFF6650a4)  val PurpleGrey40 = Color(0xFF625b71)  val Pink40 = Color(0xFF7D5260) |

*Листинг 41 – Theme*

|  |
| --- |
| package com.rarmash.prac10.ui.theme  import android.os.Build  import androidx.compose.foundation.isSystemInDarkTheme  import androidx.compose.material3.MaterialTheme  import androidx.compose.material3.darkColorScheme  import androidx.compose.material3.dynamicDarkColorScheme  import androidx.compose.material3.dynamicLightColorScheme  import androidx.compose.material3.lightColorScheme  import androidx.compose.runtime.Composable  import androidx.compose.ui.platform.LocalContext  private val DarkColorScheme = darkColorScheme(  primary = Purple80,  secondary = PurpleGrey80,  tertiary = Pink80  )  private val LightColorScheme = lightColorScheme(  primary = Purple40,  secondary = PurpleGrey40,  tertiary = Pink40  )  @Composable  fun Prac10Theme(  darkTheme: Boolean = isSystemInDarkTheme(),  dynamicColor: Boolean = true,  content: @Composable () -> Unit  ) {  val colorScheme = when {  dynamicColor && Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.S -> {  val context = LocalContext.current  if (darkTheme) dynamicDarkColorScheme(context) else dynamicLightColorScheme(context)  }  darkTheme -> DarkColorScheme  else -> LightColorScheme  }  MaterialTheme(  colorScheme = colorScheme,  typography = Typography,  content = content  )  } |

*Листинг 42 – Type*

|  |
| --- |
| package com.rarmash.prac10.ui.theme  import androidx.compose.material3.Typography  import androidx.compose.ui.text.TextStyle  import androidx.compose.ui.text.font.FontFamily  import androidx.compose.ui.text.font.FontWeight  import androidx.compose.ui.unit.sp  val Typography = Typography(  bodyLarge = TextStyle(  fontFamily = FontFamily.Default,  fontWeight = FontWeight.Normal,  fontSize = 16.sp,  lineHeight = 24.sp,  letterSpacing = 0.5.sp  )  ) |

**Практическая работа №11**

**Задание:**

В приложении из практической работы №10 реализовать контейнер Scaffold, который должен выступать центральным хостом для перемещения между созданными экранами и содержать в себе:

1. Название текущего экрана в верхней панели.
2. Наполнение основной области элементами текущего экрана.
3. Перенос каждого экрана в элементы BottomAppBar и Drawer.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac11>

*Листинг 43 – MainActivity*

|  |
| --- |
| package com.rarmash.prac11  import android.content.Context  import android.graphics.Bitmap  import android.graphics.BitmapFactory  import android.os.Bundle  import android.os.Environment  import android.widget.Toast  import androidx.activity.ComponentActivity  import androidx.activity.compose.setContent  import androidx.activity.enableEdgeToEdge  import androidx.compose.foundation.Image  import androidx.compose.foundation.layout.Arrangement  import androidx.compose.foundation.layout.Column  import androidx.compose.foundation.layout.Spacer  import androidx.compose.foundation.layout.fillMaxSize  import androidx.compose.foundation.layout.fillMaxWidth  import androidx.compose.foundation.layout.height  import androidx.compose.foundation.layout.padding  import androidx.compose.foundation.layout.size  import androidx.compose.foundation.text.KeyboardOptions  import androidx.compose.material.icons.Icons  import androidx.compose.material.icons.filled.Favorite  import androidx.compose.material.icons.filled.Menu  import androidx.compose.material3.BottomAppBar  import androidx.compose.material3.Button  import androidx.compose.material3.DrawerValue  import androidx.compose.material3.ExperimentalMaterial3Api  import androidx.compose.material3.Icon  import androidx.compose.material3.IconButton  import androidx.compose.material3.MaterialTheme  import androidx.compose.material3.ModalDrawerSheet  import androidx.compose.material3.ModalNavigationDrawer  import androidx.compose.material3.NavigationBarItem  import androidx.compose.material3.NavigationDrawerItem  import androidx.compose.material3.OutlinedTextField  import androidx.compose.material3.Scaffold  import androidx.compose.material3.Text  import androidx.compose.material3.TopAppBar  import androidx.compose.material3.rememberDrawerState  import androidx.compose.runtime.Composable  import androidx.compose.runtime.getValue  import androidx.compose.runtime.mutableStateOf  import androidx.compose.runtime.remember  import androidx.compose.runtime.rememberCoroutineScope  import androidx.compose.runtime.setValue  import androidx.compose.ui.Alignment  import androidx.compose.ui.Modifier  import androidx.compose.ui.graphics.asImageBitmap  import androidx.compose.ui.platform.LocalContext  import androidx.compose.ui.text.input.KeyboardType  import androidx.compose.ui.tooling.preview.Preview  import androidx.compose.ui.unit.dp  import com.rarmash.prac11.ui.theme.Prac11Theme  import kotlinx.coroutines.Dispatchers  import kotlinx.coroutines.launch  import kotlinx.coroutines.withContext  import okhttp3.OkHttpClient  import okhttp3.Request  import java.io.File  import java.io.FileOutputStream  import java.io.InputStream  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  enableEdgeToEdge()  setContent {  Prac11Theme {  MainScreen()  }  }  }  }  @Preview(showBackground = true)  @Composable  fun GreetingPreview() {  Prac11Theme {  ImageDownloadApp()  }  }  @OptIn(ExperimentalMaterial3Api::class)  @Composable  fun MainScreen() {  val drawerState = rememberDrawerState(DrawerValue.Closed)  val scope = rememberCoroutineScope()  var currentScreen by remember { mutableStateOf("ImageDownloader") }  val screens = listOf("ImageDownloader", "Info", "Settings")  ModalNavigationDrawer(  drawerState = drawerState,  drawerContent = {  ModalDrawerSheet {  Spacer(Modifier.height(16.dp))  screens.forEach { screen ->  NavigationDrawerItem(  label = { Text(screen) },  selected = currentScreen == screen,  onClick = {  currentScreen = screen  scope.launch { drawerState.close() }  },  icon = { Icon(Icons.Default.Favorite, contentDescription = null) }  )  }  }  },  content = {  Scaffold(  topBar = {  TopAppBar(  title = { Text(text = currentScreen) },  navigationIcon = {  IconButton(onClick = {  scope.launch {  if (drawerState.isClosed) {  drawerState.open()  }  }  }) {  Icon(Icons.Default.Menu, contentDescription = null)  }  }  )  },  bottomBar = {  BottomAppBar {  screens.forEach { screen ->  NavigationBarItem(  icon = { Icon(Icons.Default.Favorite, contentDescription = null) },  label = { Text(screen) },  selected = currentScreen == screen,  onClick = { currentScreen = screen }  )  }  }  },  content = { contentPadding ->  Column(  modifier = Modifier  .fillMaxSize()  .padding(contentPadding)  .padding(16.dp)  ) {  when (currentScreen) {  "ImageDownloader" -> ImageDownloadApp()  "Info" -> InfoScreen()  "Settings" -> SettingsScreen()  }  }  }  )  },  )  }  @Composable  fun ImageDownloadApp() {  var url by remember { mutableStateOf("") }  var bitmap by remember { mutableStateOf<Bitmap?>(null) }  var coroutineScope = rememberCoroutineScope()  val context = LocalContext.current  Column(  modifier = Modifier  .fillMaxSize()  .padding(16.dp)  ) {  OutlinedTextField(  value = url,  onValueChange = { url = it },  label = { Text("Enter Image URL") },  keyboardOptions = KeyboardOptions.Default.copy(keyboardType = KeyboardType.Uri),  modifier = Modifier.fillMaxWidth()  )  Spacer(modifier = Modifier.height(16.dp))  Button(  onClick = {  if (url.isNotEmpty()) {  coroutineScope.launch {  val downloadedBitmap = downloadImage(url, context)  bitmap = downloadedBitmap  }  } else {  Toast.makeText(context, "Please enter a valid URL", Toast.LENGTH\_SHORT).show()  }  },  modifier = Modifier.fillMaxWidth()  ) {  Text("Download Image")  }  Spacer(modifier = Modifier.height(16.dp))  bitmap?.let {  Image(  bitmap = it.asImageBitmap(),  contentDescription = null,  modifier = Modifier.size(400.dp)  )  }  }  }  @Composable  fun InfoScreen() {  Column(  modifier = Modifier.fillMaxSize(),  horizontalAlignment = Alignment.CenterHorizontally,  verticalArrangement = Arrangement.Center  ) {  Text("Information Screen", style = MaterialTheme.typography.titleLarge)  }  }  @Composable  fun SettingsScreen() {  Column(  modifier = Modifier  .fillMaxSize()  .padding(16.dp),  horizontalAlignment = Alignment.CenterHorizontally,  verticalArrangement = Arrangement.Center  ) {  Text("Settings Screen", style = MaterialTheme.typography.titleLarge)  }  }  suspend fun downloadImage(url: String, context: Context): Bitmap? {  return withContext(Dispatchers.IO) {  try {  val bitmap = downloadImageFromNetwork(url)  if (bitmap != null) {  saveImageToDisk(bitmap, context)  withContext(Dispatchers.Main) {  Toast.makeText(context, "Image downloaded successfully", Toast.LENGTH\_SHORT)  .show()  }  } else {  withContext(Dispatchers.Main) {  Toast.makeText(context, "Failed to download image", Toast.LENGTH\_SHORT).show()  }  }  bitmap  } catch (e: Exception) {  e.printStackTrace()  null  }  }  }  private fun downloadImageFromNetwork(url: String): Bitmap? {  return try {  val client = OkHttpClient()  val request = Request.Builder().url(url).build()  val response = client.newCall(request).execute()  if (response.isSuccessful) {  val inputStream: InputStream = response.body?.byteStream() ?: return null  BitmapFactory.decodeStream(inputStream)  } else {  null  }  } catch (e: Exception) {  e.printStackTrace()  null  }  }  private fun saveImageToDisk(bitmap: Bitmap, context: Context) {  val directory =  File(context.getExternalFilesDir(Environment.DIRECTORY\_PICTURES), "downloadedImages")  if (!directory.exists()) {  directory.mkdirs()  }  val file = File(directory, "downloaded\_image.png")  val outputStream = FileOutputStream(file)  bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream)  outputStream.flush()  outputStream.close()  } |

**Практическая работа №12**

**Задание:**

В приложении из практической работы №11 реализовать навигацию между экранами, а также добавить WorkManager, выполняющий любую задачу согласно выбранной тематики приложения.

**Ссылка на Github**

<https://github.com/Rarmash/MIREA/tree/master/3%20%D0%BA%D1%83%D1%80%D1%81/5%20%D1%81%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%BC%D0%BE%D0%B1%D0%B8%D0%BB%D1%8C%D0%BD%D1%8B%D1%85%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%9A%D0%BE%D1%82%D0%BB%D0%B8%D0%BD/RMP_Prac12>

*Листинг 44 – MainActivity*

|  |
| --- |
| package com.rarmash.prac12  import android.content.Context  import android.graphics.Bitmap  import android.graphics.BitmapFactory  import android.os.Bundle  import android.os.Environment  import android.widget.Toast  import androidx.activity.ComponentActivity  import androidx.activity.compose.setContent  import androidx.activity.enableEdgeToEdge  import androidx.compose.foundation.Image  import androidx.compose.foundation.layout.Arrangement  import androidx.compose.foundation.layout.Column  import androidx.compose.foundation.layout.Spacer  import androidx.compose.foundation.layout.fillMaxSize  import androidx.compose.foundation.layout.fillMaxWidth  import androidx.compose.foundation.layout.height  import androidx.compose.foundation.layout.padding  import androidx.compose.foundation.layout.size  import androidx.compose.foundation.text.KeyboardOptions  import androidx.compose.material.icons.Icons  import androidx.compose.material.icons.filled.Favorite  import androidx.compose.material.icons.filled.Menu  import androidx.compose.material3.BottomAppBar  import androidx.compose.material3.Button  import androidx.compose.material3.DrawerValue  import androidx.compose.material3.ExperimentalMaterial3Api  import androidx.compose.material3.Icon  import androidx.compose.material3.IconButton  import androidx.compose.material3.MaterialTheme  import androidx.compose.material3.ModalDrawerSheet  import androidx.compose.material3.ModalNavigationDrawer  import androidx.compose.material3.NavigationBarItem  import androidx.compose.material3.NavigationDrawerItem  import androidx.compose.material3.OutlinedTextField  import androidx.compose.material3.Scaffold  import androidx.compose.material3.Text  import androidx.compose.material3.TopAppBar  import androidx.compose.material3.rememberDrawerState  import androidx.compose.runtime.Composable  import androidx.compose.runtime.getValue  import androidx.compose.runtime.mutableStateOf  import androidx.compose.runtime.remember  import androidx.compose.runtime.rememberCoroutineScope  import androidx.compose.runtime.setValue  import androidx.compose.ui.Alignment  import androidx.compose.ui.Modifier  import androidx.compose.ui.graphics.asImageBitmap  import androidx.compose.ui.platform.LocalContext  import androidx.compose.ui.platform.LocalLifecycleOwner  import androidx.compose.ui.text.input.KeyboardType  import androidx.compose.ui.tooling.preview.Preview  import androidx.compose.ui.unit.dp  import androidx.work.OneTimeWorkRequestBuilder  import androidx.work.WorkInfo  import androidx.work.WorkManager  import androidx.work.Worker  import androidx.work.WorkerParameters  import androidx.work.workDataOf  import com.rarmash.prac12.ui.theme.Prac12Theme  import kotlinx.coroutines.launch  import java.io.File  import java.io.FileOutputStream  import java.io.InputStream  import java.net.URL  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  enableEdgeToEdge()  setContent {  Prac12Theme {  MainScreen()  }  }  }  }  @Preview(showBackground = true)  @Composable  fun GreetingPreview() {  Prac12Theme {  ImageDownloadApp()  }  }  @OptIn(ExperimentalMaterial3Api::class)  @Composable  fun MainScreen() {  val drawerState = rememberDrawerState(DrawerValue.Closed)  val scope = rememberCoroutineScope()  var currentScreen by remember { mutableStateOf("ImageDownloader") }  val screens = listOf("ImageDownloader", "Info", "Settings")  ModalNavigationDrawer(  drawerState = drawerState,  drawerContent = {  ModalDrawerSheet {  Spacer(Modifier.height(16.dp))  screens.forEach { screen ->  NavigationDrawerItem(  label = { Text(screen) },  selected = currentScreen == screen,  onClick = {  currentScreen = screen  scope.launch { drawerState.close() }  },  icon = { Icon(Icons.Default.Favorite, contentDescription = null) }  )  }  }  },  content = {  Scaffold(  topBar = {  TopAppBar(  title = { Text(text = currentScreen) },  navigationIcon = {  IconButton(onClick = {  scope.launch {  if (drawerState.isClosed) {  drawerState.open()  }  }  }) {  Icon(Icons.Default.Menu, contentDescription = null)  }  }  )  },  bottomBar = {  BottomAppBar {  screens.forEach { screen ->  NavigationBarItem(  icon = { Icon(Icons.Default.Favorite, contentDescription = null) },  label = { Text(screen) },  selected = currentScreen == screen,  onClick = { currentScreen = screen }  )  }  }  },  content = { contentPadding ->  Column(  modifier = Modifier  .fillMaxSize()  .padding(contentPadding)  .padding(16.dp)  ) {  when (currentScreen) {  "ImageDownloader" -> ImageDownloadApp()  "Info" -> InfoScreen()  "Settings" -> SettingsScreen()  }  }  }  )  },  )  }  @Composable  fun ImageDownloadApp() {  var url by remember { mutableStateOf("") }  val context = LocalContext.current  var bitmap by remember { mutableStateOf<Bitmap?>(null) }  val lifecycleOwner = LocalLifecycleOwner.current  Column(  modifier = Modifier  .fillMaxSize()  .padding(16.dp)  ) {  OutlinedTextField(  value = url,  onValueChange = { url = it },  label = { Text("Enter Image URL") },  keyboardOptions = KeyboardOptions.Default.copy(keyboardType = KeyboardType.Uri),  modifier = Modifier.fillMaxWidth()  )  Spacer(modifier = Modifier.height(16.dp))  Button(  onClick = {  if (url.isNotEmpty()) {  val workManager = WorkManager.getInstance(context)  val inputData = workDataOf("imageUrl" to url)  val downloadWorkRequest = OneTimeWorkRequestBuilder<ImageDownloadWorker>()  .setInputData(inputData)  .build()  workManager.enqueue(downloadWorkRequest)  workManager.getWorkInfoByIdLiveData(downloadWorkRequest.id)  .observe(lifecycleOwner) { workInfo ->  if (workInfo != null && workInfo.state == WorkInfo.State.SUCCEEDED) {  val imagePath = workInfo.outputData.getString("imagePath")  if (imagePath != null) {  val imageBitmap = BitmapFactory.decodeFile(imagePath)  bitmap = imageBitmap  Toast.makeText(  context,  "Image downloaded to: $imagePath",  Toast.LENGTH\_SHORT  ).show()  }  }  }  }  },  modifier = Modifier.fillMaxWidth()  ) {  Text("Download Image")  }  Spacer(modifier = Modifier.height(16.dp))  bitmap?.let {  Image(  bitmap = it.asImageBitmap(),  contentDescription = null,  modifier = Modifier.size(400.dp)  )  }  }  }  class ImageDownloadWorker(context: Context, workerParams: WorkerParameters) :  Worker(context, workerParams) {  override fun doWork(): Result {  val imageUrl = inputData.getString("imageUrl") ?: return Result.failure()  return try {  val bitmap = downloadImage(imageUrl)  bitmap?.let {  val file = saveImageToDisk(it, applicationContext)  val outputData = workDataOf("imagePath" to file.absolutePath)  Result.success(outputData)  } ?: Result.failure()  } catch (e: Exception) {  Result.failure()  }  }  }  @Composable  fun InfoScreen() {  Column(  modifier = Modifier.fillMaxSize(),  horizontalAlignment = Alignment.CenterHorizontally,  verticalArrangement = Arrangement.Center  ) {  Text("Information Screen", style = MaterialTheme.typography.titleLarge)  }  }  @Composable  fun SettingsScreen() {  Column(  modifier = Modifier  .fillMaxSize()  .padding(16.dp),  horizontalAlignment = Alignment.CenterHorizontally,  verticalArrangement = Arrangement.Center  ) {  Text("Settings Screen", style = MaterialTheme.typography.titleLarge)  }  }  private fun downloadImage(url: String): Bitmap? {  return try {  val inputStream: InputStream = URL(url).openStream()  BitmapFactory.decodeStream(inputStream)  } catch (e: Exception) {  e.printStackTrace()  null  }  }  private fun saveImageToDisk(bitmap: Bitmap, context: Context): File {  val directory =  File(context.getExternalFilesDir(Environment.DIRECTORY\_PICTURES), "downloadedImages")  if (!directory.exists()) {  directory.mkdirs()  }  val file = File(directory, "downloaded\_image.png")  val outputStream = FileOutputStream(file)  bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream)  outputStream.flush()  outputStream.close()  return file  } |