|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 практическим работам №1-12**

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

[ПРАКТИЧЕСКАЯ РАБОТА №1 4](#_Toc185525687)

[Код практического задания 5](#_Toc185525688)

[Тестирование практического задания 7](#_Toc185525689)

[Вывод 8](#_Toc185525690)

[ПРАКТИЧЕСКАЯ РАБОТА №2 9](#_Toc185525691)

[Описание ручного подхода и Navigation API 10](#_Toc185525692)

[Код практического задания 11](#_Toc185525693)

[Тестирование практического задания 15](#_Toc185525694)

[Вывод 19](#_Toc185525695)

[ПРАКТИЧЕСКАЯ РАБОТА №3 20](#_Toc185525696)

[Код практического задания 21](#_Toc185525697)

[Тестирование практического задания 27](#_Toc185525698)

[Вывод 28](#_Toc185525699)

[ПРАКТИЧЕСКАЯ РАБОТА №4 29](#_Toc185525700)

[Код практического задания 30](#_Toc185525701)

[Тестирование практического задания 34](#_Toc185525702)

[Вывод 37](#_Toc185525703)

[ПРАКТИЧЕСКАЯ РАБОТА №5-6 38](#_Toc185525704)

[Код практического задания 39](#_Toc185525705)

[Тестирование практического задания 46](#_Toc185525706)

[Вывод 48](#_Toc185525707)

[ПРАКТИЧЕСКАЯ РАБОТА №7-8 49](#_Toc185525708)

[Код практического задания 50](#_Toc185525709)

[Тестирование практического задания 54](#_Toc185525710)

[Вывод 57](#_Toc185525711)

[ПРАКТИЧЕСКАЯ РАБОТА №9 58](#_Toc185525712)

[Код практического задания 59](#_Toc185525713)

[Тестирование практического задания 61](#_Toc185525714)

[Вывод 61](#_Toc185525715)

[ПРАКТИЧЕСКАЯ РАБОТА №10-12 62](#_Toc185525716)

[Код практического задания 63](#_Toc185525717)

[Тестирование практического задания 70](#_Toc185525718)

[Вывод 73](#_Toc185525719)

[ЗАКЛЮЧЕНИЕ 73](#_Toc185525720)

# ПРАКТИЧЕСКАЯ РАБОТА №1

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

## Код практического задания

В ходе выполнения работы, мы написали код на языке Kotlin (см. Листинг 1)

Листинг 1. MainActivity.kt

|  |
| --- |
| package com.example.practi4eskaya\_1  import java.time.LocalDate import java.time.format.DateTimeParseException import java.util.Scanner  // Класс для хранения информации о конкретном расходе data class Expense(  val amount: Double,  val category: String,  val date: LocalDate ) {  // Метод для вывода информации о расходе  fun displayExpense() {  println("Сумма: $amount, Категория: $category, Дата: $date")  } }  // Класс для работы со списком всех расходов class ExpenseTracker {  private val expenses = mutableListOf<Expense>()   // Метод для добавления нового расхода в список  fun addExpense(amount: Double, category: String, date: LocalDate) {  val expense = Expense(amount, category, date)  expenses.add(expense)  println("Добавлен новый расход: $amount, категория: $category, дата: $date")  }   // Метод для вывода всех расходов  fun displayAllExpenses() {  if (expenses.isEmpty()) {  println("Расходы отсутствуют.")  } else {  println("Все расходы:")  expenses.forEach { it.displayExpense() }  }  }   // Метод для подсчета суммы всех расходов по каждой категории  fun calculateExpensesByCategory() {  if (expenses.isEmpty()) {  println("Расходы отсутствуют.")  return  }  val categoryTotals = expenses.groupBy { it.category }  .mapValues { entry -> entry.value.sumOf { it.amount } }   println("Сумма расходов по категориям:")  categoryTotals.forEach { (category, total) ->  println("Категория: $category, Сумма: $total")  }  } }  // Пример с динамическим меню fun main() {  val expenseTracker = ExpenseTracker()  val scanner = Scanner(System.`in`)   while (true) {  println(  """  Выберите действие:  1. Добавить новый расход  2. Показать все расходы  3. Показать сумму расходов по категориям  4. Выйти  """.trimIndent()  )  print("Введите номер действия: ")  when (scanner.nextLine().trim()) {  "1" -> {  try {  print("Введите сумму расхода: ")  val amount = scanner.nextLine().toDouble()   print("Введите категорию расхода: ")  val category = scanner.nextLine()   print("Введите дату расхода (в формате ГГГГ-ММ-ДД): ")  val date = LocalDate.parse(scanner.nextLine())   expenseTracker.addExpense(amount, category, date)  } catch (e: NumberFormatException) {  println("Ошибка: сумма должна быть числом.")  } catch (e: DateTimeParseException) {  println("Ошибка: дата должна быть в формате ГГГГ-ММ-ДД.")  } catch (e: Exception) {  println("Произошла ошибка: ${e.message}")  }  }   "2" -> {  expenseTracker.displayAllExpenses()  }   "3" -> {  expenseTracker.calculateExpensesByCategory()  }   "4" -> {  println("Выход из программы. До свидания!")  break  }   else -> {  println("Ошибка: выберите корректное действие.")  }  }  println()  } } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 1-4)

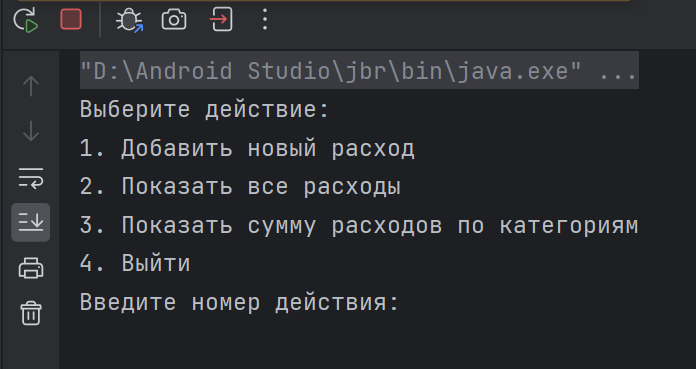


Рисунок 1. Пользовательское меню

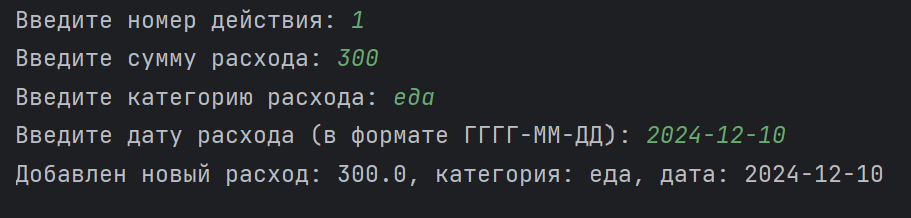


Рисунок 2. Добавление нового расхода

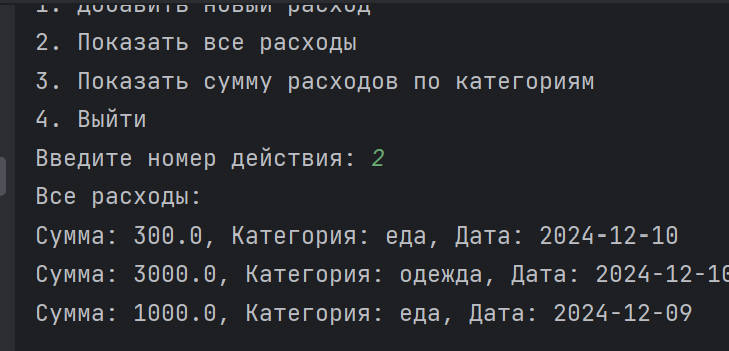


Рисунок 3. Вывод всех расходов

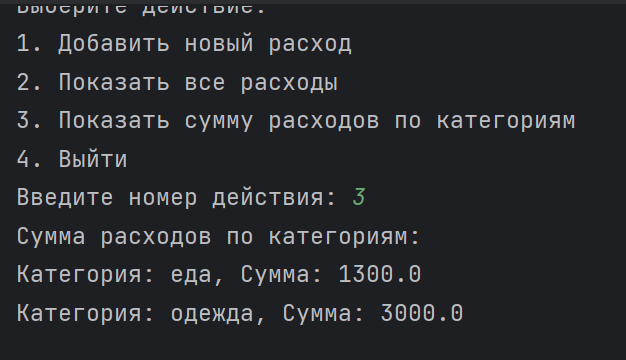


Рисунок 4. Вывод сумм расходов по категориям

## Вывод

В процессе выполнения работы мы создали приложение, которое помогает пользователю отслеживать его личные расходы

# ПРАКТИЧЕСКАЯ РАБОТА №2

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 2-5)

Листинг 2. 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 3. 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 4. 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 5. 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 5-10)

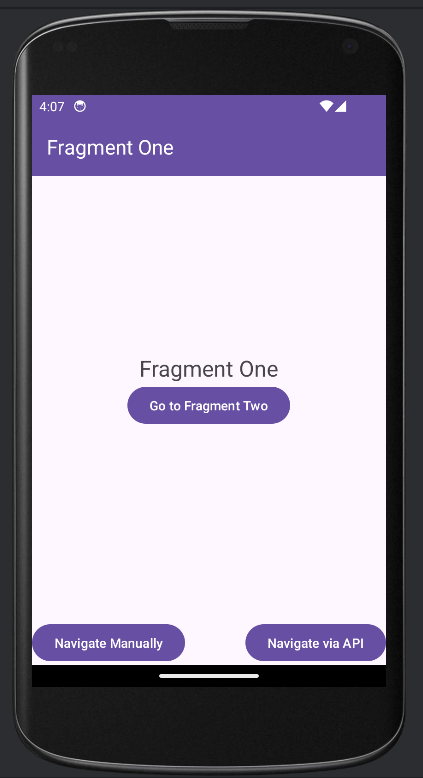


Рисунок 5. Запуск приложения



Рисунок 6. Кнопки управления фрагментами (вручную/через API)

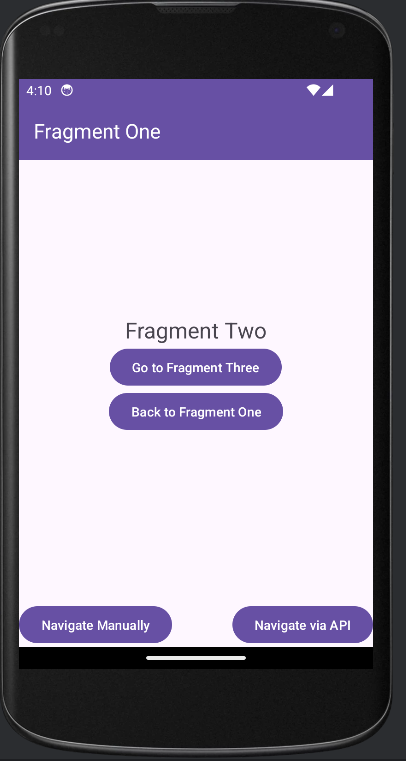


Рисунок 7. Проверка выполнения перехода между фрагментами вручную (режим Navigate Manually)

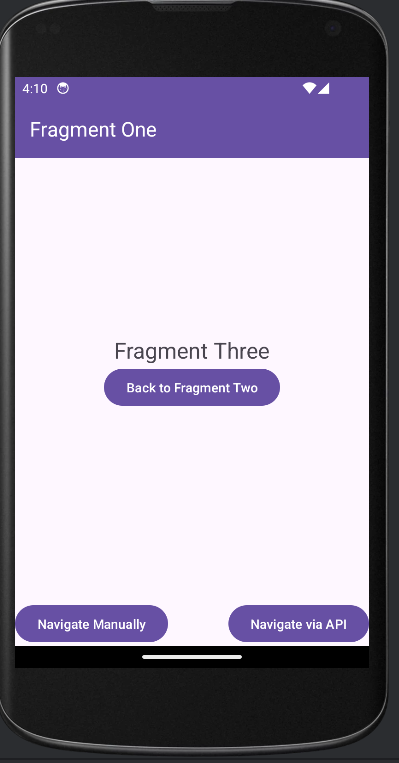


Рисунок 8. Проверка выполнения перехода между фрагментами вручную (режим Navigate Manually)

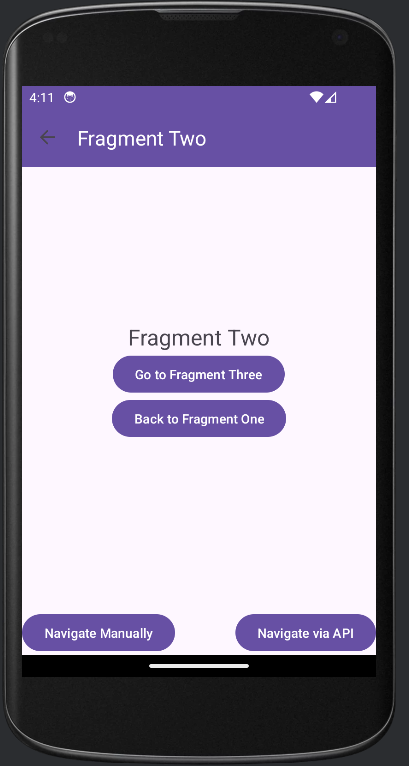


Рисунок 9. Проверка выполнения перехода между фрагментами через API (режим Navigate via API)

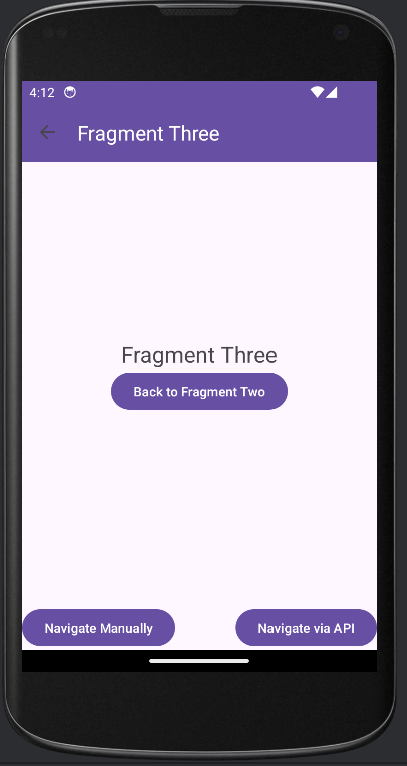


Рисунок 10. Проверка выполнения перехода между фрагментами через API (режим Navigate via API)

## Вывод

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

# ПРАКТИЧЕСКАЯ РАБОТА №3

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

## Код практического задания

В ходе выполнения работы, мы написали код для реализации Android Architecture Components в фрагментах (см. Листинги 6-16)

Листинг 6. UserDao.kt

|  |
| --- |
| package com.example.pract3.dao  import androidx.room.Dao import androidx.room.Insert import androidx.room.Query import com.example.pract3.models.User  @Dao interface UserDao {  @Insert  suspend fun insert(user: User)   @Query("SELECT \* FROM users WHERE id = :id")  suspend fun getUser(id: Int): User } |

Листинг 7. User.kt

|  |
| --- |
| package com.example.pract3.models  import androidx.room.Entity import androidx.room.PrimaryKey  @Entity(tableName = "users") data class User(  @PrimaryKey(autoGenerate = true) val id: Int,  val name: String,  val dob: String,  val email: String ) |

Листинг 8. UserRepository.kt

|  |
| --- |
| package com.example.pract3.repository  import androidx.lifecycle.LiveData import androidx.lifecycle.MutableLiveData import androidx.lifecycle.liveData import com.example.pract3.dao.UserDao import com.example.pract3.models.User  class UserRepository(private val userDao: UserDao) {   val userLiveData = MutableLiveData<User>()   suspend fun insert(user: User) {  userDao.insert(user)  }   fun getUser(id: Int): LiveData<User> {  return liveData {  val user = userDao.getUser(id)  emit(user)  }  } } |

Листинг 9. LoginViewModel.kt

|  |
| --- |
| package com.example.pract3.vm  import android.app.Application import androidx.lifecycle.AndroidViewModel import androidx.lifecycle.LiveData import com.example.pract3.AppDatabase import com.example.pract3.models.User import com.example.pract3.repository.UserRepository  class LoginViewModel(application: Application) : AndroidViewModel(application) {   private val repository: UserRepository  val userLiveData: LiveData<User>   init {  val userDao = AppDatabase.getDatabase(application).userDao()  repository = UserRepository(userDao)  userLiveData = repository.userLiveData  }   fun login(email: String, password: String) {  // Логика авторизации  } } |

Листинг 10. ProfileViewModel.kt

|  |
| --- |
| package com.example.pract3.vm  import android.app.Application import androidx.lifecycle.AndroidViewModel import androidx.lifecycle.LiveData import com.example.pract3.AppDatabase import com.example.pract3.models.User import com.example.pract3.repository.UserRepository  class ProfileViewModel(application: Application) : AndroidViewModel(application) {   private val repository: UserRepository  val userLiveData: LiveData<User>   init {  val userDao = AppDatabase.getDatabase(application).userDao()  repository = UserRepository(userDao)  userLiveData = repository.getUser(1)  } } |

Листинг 11. SettingsViewModel.kt

|  |
| --- |
| package com.example.pract3.vm  import android.app.Application import androidx.lifecycle.AndroidViewModel import androidx.lifecycle.LiveData import androidx.lifecycle.MutableLiveData  class SettingsViewModel(application: Application) : AndroidViewModel(application) {   private val \_notificationsEnabled = MutableLiveData<Boolean>()  val notificationsEnabled: LiveData<Boolean> get() = \_notificationsEnabled   fun setNotificationsEnabled(enabled: Boolean) {  \_notificationsEnabled.value = enabled  } } |

Листинг 12. AppDatabase.kt

|  |
| --- |
| package com.example.pract3  import android.content.Context import androidx.room.Database import androidx.room.Room import androidx.room.RoomDatabase import com.example.pract3.dao.UserDao import com.example.pract3.models.User  @Database(entities = [User::class], version = 1) abstract class AppDatabase : RoomDatabase() {  abstract fun userDao(): UserDao   companion object {  @Volatile  private var INSTANCE: AppDatabase? = null   fun getDatabase(context: Context): AppDatabase {  return INSTANCE ?: synchronized(this) {  val instance = Room.databaseBuilder(  context.applicationContext,  AppDatabase::class.java,  "app\_database"  ).build()  INSTANCE = instance  instance  }  }  } } |

Листинг 13. LoginFragment.kt

|  |
| --- |
| package com.example.pract3  import android.os.Bundle import androidx.fragment.app.Fragment import android.view.View import android.widget.Button import androidx.lifecycle.ViewModelProvider import androidx.navigation.fragment.findNavController import com.example.pract3.vm.LoginViewModel   class LoginFragment : Fragment(R.layout.fragment\_login) {   private lateinit var viewModel: LoginViewModel   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   viewModel = ViewModelProvider(this).get(LoginViewModel::class.java)   val loginButton = view.findViewById<Button>(R.id.loginButton)  val registerButton = view.findViewById<Button>(R.id.registerButton)   loginButton.setOnClickListener {  findNavController().navigate(R.id.action\_loginFragment\_to\_profileFragment)  // Логика авторизации  }   registerButton.setOnClickListener {  // Логика регистрации  }  } } |

Листинг 14. MainActivity.kt

|  |
| --- |
| package com.example.pract3  import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import androidx.navigation.fragment.NavHostFragment import androidx.navigation.ui.NavigationUI import com.example.pract3.R import com.google.android.material.bottomnavigation.BottomNavigationView  class MainActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)   // Находим NavHostFragment  val navHostFragment = supportFragmentManager.findFragmentById(R.id.nav\_host\_fragment) as NavHostFragment  // Получаем NavController из NavHostFragment  val navController = navHostFragment.navController   // Связываем BottomNavigationView с NavController  val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottomNavigationView)  NavigationUI.setupWithNavController(bottomNavigationView, navController)   // Управляем логикой навигации через NavigationUI  bottomNavigationView.setOnItemSelectedListener { item ->  when (item.itemId) {  R.id.nav\_login -> {  navController.navigate(R.id.loginFragment)  true  }  R.id.nav\_profile -> {  navController.navigate(R.id.profileFragment)  true  }  R.id.nav\_settings -> {  navController.navigate(R.id.settingsFragment)  true  }  else -> false  }  }  } } |

Листинг 15. ProfileFragment.kt

|  |
| --- |
| package com.example.pract3  import android.os.Bundle import androidx.fragment.app.Fragment import android.view.View import android.widget.TextView import androidx.lifecycle.ViewModelProvider import com.example.pract3.vm.ProfileViewModel  class ProfileFragment : Fragment(R.layout.fragment\_profile) {   private lateinit var viewModel: ProfileViewModel   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   viewModel = ViewModelProvider(this).get(ProfileViewModel::class.java)   val nameTextView = view.findViewById<TextView>(R.id.nameTextView)  val emailTextView = view.findViewById<TextView>(R.id.emailTextView)  val dobTextView = view.findViewById<TextView>(R.id.birthDateTextView)   // Убедитесь, что данные пользователя не `null` перед отображением  viewModel.userLiveData.observe(viewLifecycleOwner) { user ->  user?.let {  nameTextView.text = it.name  emailTextView.text = it.email  dobTextView.text = it.dob  } ?: run {  // Данные пользователя еще не загружены  nameTextView.text = "Loading..."  emailTextView.text = "Loading..."  dobTextView.text = "Loading..."  }  }  } } |

Листинг 16. SettingsFragment.kt

|  |
| --- |
| package com.example.pract3  import android.os.Bundle import androidx.fragment.app.Fragment import android.view.View import android.widget.CheckBox import androidx.lifecycle.ViewModelProvider import com.example.pract3.vm.SettingsViewModel  class SettingsFragment : Fragment(R.layout.fragment\_settings) {   private lateinit var viewModel: SettingsViewModel   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   viewModel = ViewModelProvider(this).get(SettingsViewModel::class.java)   val checkBox = view.findViewById<CheckBox>(R.id.notificationsCheckBox)   viewModel.run {  notificationsEnabled.observe(viewLifecycleOwner) { enabled ->  checkBox.isChecked = enabled  }  }   checkBox.setOnCheckedChangeListener { \_, isChecked ->  viewModel.setNotificationsEnabled(isChecked)  }  } } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 11-13)

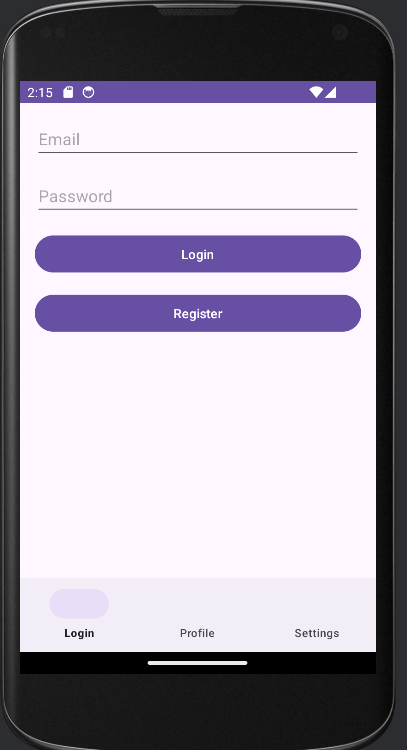


Рисунок 11. Фрагмент Login

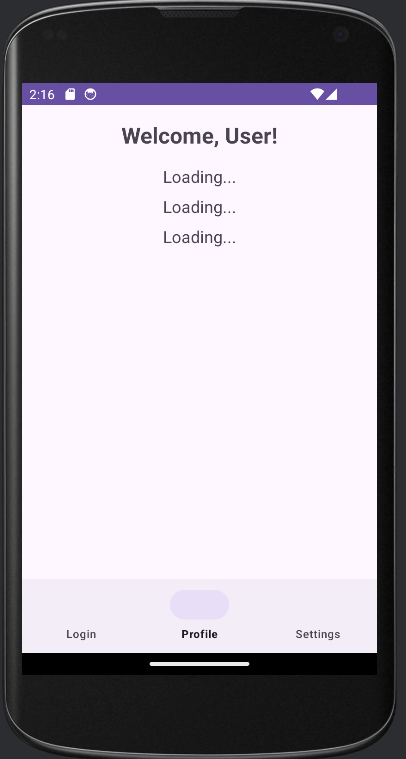


Рисунок 12. Фрагмент Profile

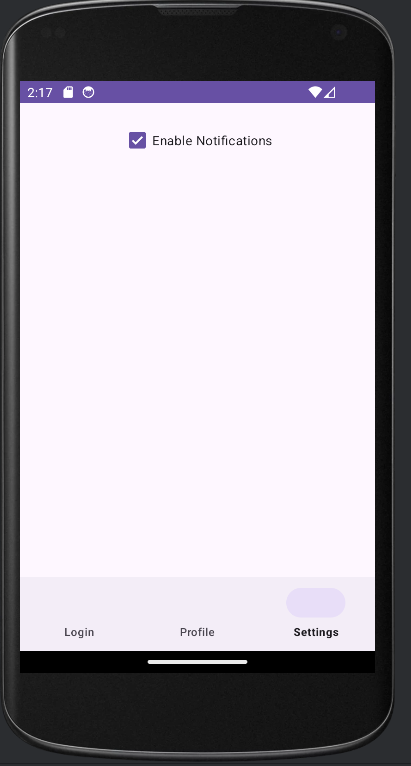


Рисунок 13. Фрагмент Settings

## Вывод

В процессе выполнения работы мы узнали, как работает структура Android Architecture Components, а также её реализовали

# ПРАКТИЧЕСКАЯ РАБОТА №4

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.

## Код практического задания

В ходе выполнения работы, мы написали код для камеры, списка, а также MainActivity для выбора списка/камеры (см. Листинги 17-19)

Листинг 17. CameraActivity.kt

|  |
| --- |
| package com.example.practi4eskaya\_4  import android.Manifest import android.content.Intent import android.content.pm.PackageManager import android.os.Bundle import android.util.Log import android.widget.Button import androidx.activity.result.contract.ActivityResultContracts import androidx.appcompat.app.AppCompatActivity import androidx.core.app.ActivityCompat import androidx.core.content.ContextCompat import androidx.recyclerview.widget.LinearLayoutManager import androidx.recyclerview.widget.RecyclerView import java.io.File import java.text.SimpleDateFormat import java.util.\* import androidx.camera.core.Camera import androidx.camera.core.CameraSelector import androidx.camera.core.Preview import androidx.camera.lifecycle.ProcessCameraProvider import androidx.camera.view.PreviewView import java.util.concurrent.ExecutorService import java.util.concurrent.Executors   // CameraActivity: Handles camera preview and photo capturing class CameraActivity : AppCompatActivity() {  private val permissions = arrayOf(Manifest.permission.CAMERA, Manifest.permission.WRITE\_EXTERNAL\_STORAGE)  private lateinit var previewView: PreviewView   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_camera)   previewView = findViewById(R.id.preview\_view)   if (!hasPermissions()) {  ActivityCompat.requestPermissions(this, permissions, 1)  } else {  startCamera()  }   val buttonCapture = findViewById<Button>(R.id.button\_capture)  buttonCapture.setOnClickListener {  savePhotoTimestamp()  }  }   private fun hasPermissions(): Boolean {  return permissions.all { perm ->  ContextCompat.checkSelfPermission(this, perm) == PackageManager.PERMISSION\_GRANTED  }  }   private fun startCamera() {  val cameraProviderFuture = ProcessCameraProvider.getInstance(this)   cameraProviderFuture.addListener({  val cameraProvider = cameraProviderFuture.get()   val preview = Preview.Builder().build().also {  it.setSurfaceProvider(previewView.surfaceProvider)  }   val cameraSelector = CameraSelector.DEFAULT\_BACK\_CAMERA   try {  cameraProvider.unbindAll()  cameraProvider.bindToLifecycle(this, cameraSelector, preview)  } catch (exc: Exception) {  Log.e("CameraActivity", "Use case binding failed", exc)  }   }, ContextCompat.getMainExecutor(this))  }   private fun savePhotoTimestamp() {  val photosDir = File(getExternalFilesDir(null), "photos")  if (!photosDir.exists()) {  val created = photosDir.mkdirs()  Log.d("CameraActivity", "Photos directory created: $created")  }   val file = File(photosDir, "date")  if (!file.exists()) {  val created = file.createNewFile()  Log.d("CameraActivity", "Date file created: $created")  }   val currentTime = System.currentTimeMillis() + (3 \* 60 \* 60 \* 1000) // Добавляем 3 часа  val timestamp = SimpleDateFormat("yyyy-MM-dd HH:mm:ss", Locale.getDefault()).format(Date(currentTime))  file.appendText("$timestamp\n")  Log.d("CameraActivity", "Saved timestamp: $timestamp at ${file.absolutePath}")  } } |

Листинг 18. ListActivity.kt

|  |
| --- |
| package com.example.practi4eskaya\_4  import android.Manifest import android.content.Intent import android.content.pm.PackageManager import android.os.Bundle import android.util.Log import android.widget.Button import androidx.activity.result.contract.ActivityResultContracts import androidx.appcompat.app.AppCompatActivity import androidx.core.app.ActivityCompat import androidx.core.content.ContextCompat import androidx.recyclerview.widget.LinearLayoutManager import androidx.recyclerview.widget.RecyclerView import java.io.File import java.text.SimpleDateFormat import java.util.\*  // ListActivity: Displays timestamps in a RecyclerView class ListActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_list)   val recyclerView = findViewById<RecyclerView>(R.id.recycler\_view)  recyclerView.layoutManager = LinearLayoutManager(this)   val photosDir = File(getExternalFilesDir(null), "photos")  val file = File(photosDir, "date")   val timestamps = if (file.exists()) {  file.readLines().sortedDescending()  } else {  emptyList()  }   recyclerView.adapter = TimestampsAdapter(timestamps)  } }  // TimestampsAdapter: Binds timestamps to RecyclerView class TimestampsAdapter(private val timestamps: List<String>) : RecyclerView.Adapter<TimestampsAdapter.ViewHolder>() {   class ViewHolder(val view: android.widget.TextView) : RecyclerView.ViewHolder(view)   override fun onCreateViewHolder(parent: android.view.ViewGroup, viewType: Int): ViewHolder {  val textView = android.widget.TextView(parent.context)  textView.setPadding(16, 16, 16, 16)  return ViewHolder(textView)  }   override fun onBindViewHolder(holder: ViewHolder, position: Int) {  holder.view.text = timestamps[position]  }   override fun getItemCount(): Int = timestamps.size } |

Листинг 19. MainActivity.kt

|  |
| --- |
| package com.example.practi4eskaya\_4  import android.Manifest import android.content.Intent import android.content.pm.PackageManager import android.os.Bundle import android.util.Log import android.widget.Button import androidx.activity.result.contract.ActivityResultContracts import androidx.appcompat.app.AppCompatActivity import androidx.core.app.ActivityCompat import androidx.core.content.ContextCompat import androidx.recyclerview.widget.LinearLayoutManager import androidx.recyclerview.widget.RecyclerView import java.io.File import java.text.SimpleDateFormat import java.util.\*  // MainActivity: Handles navigation between Camera and List screens class MainActivity : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)   findViewById<Button>(R.id.button\_camera).setOnClickListener {  startActivity(Intent(this, CameraActivity::class.java))  }   findViewById<Button>(R.id.button\_list).setOnClickListener {  startActivity(Intent(this, ListActivity::class.java))  }  } } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 14-20)

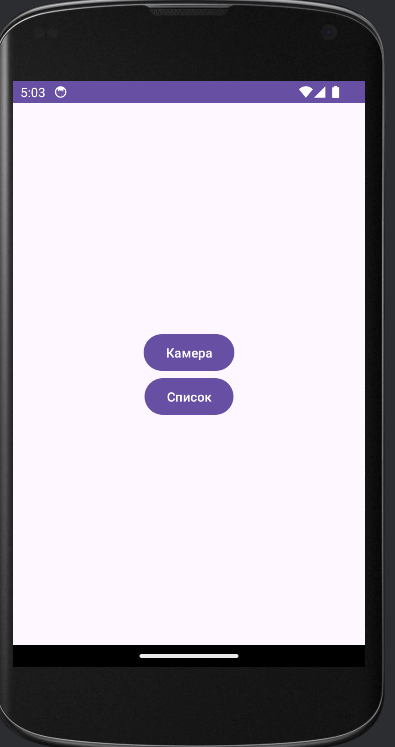


Рисунок 14. Запуск приложения

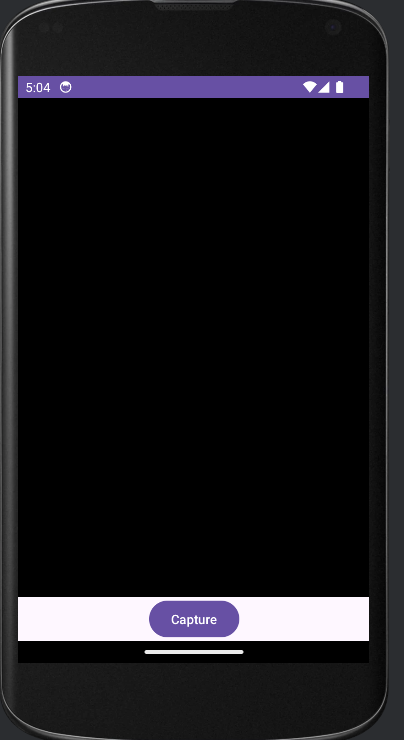


Рисунок 15. Активность с камерой

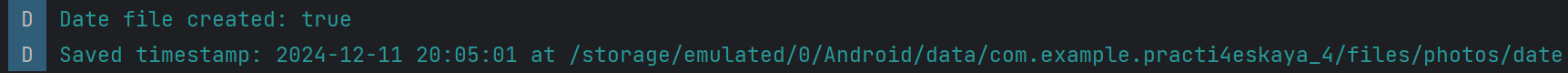


Рисунок 16. Сохранение данных о фотографии при нажатии кнопки Capture

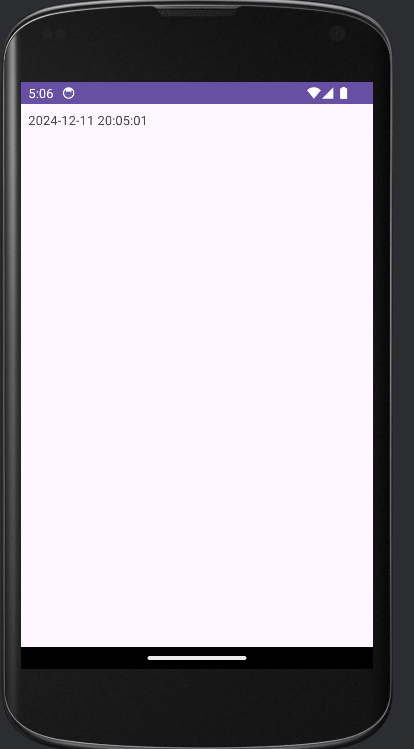


Рисунок 17. Проверка работы списка

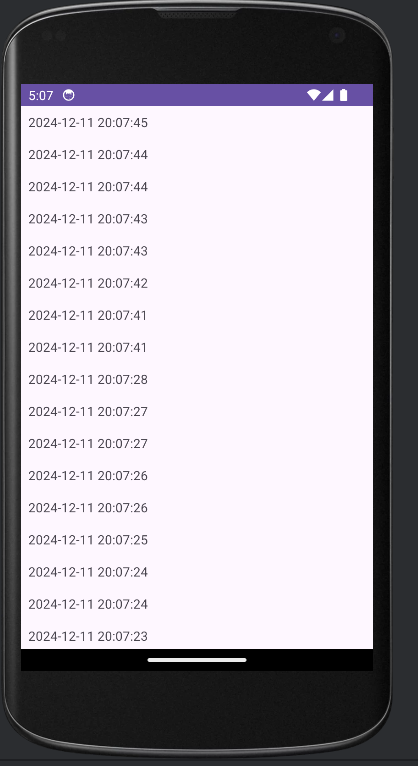


Рисунок 18. Проверка выполнения работы RecyclerView

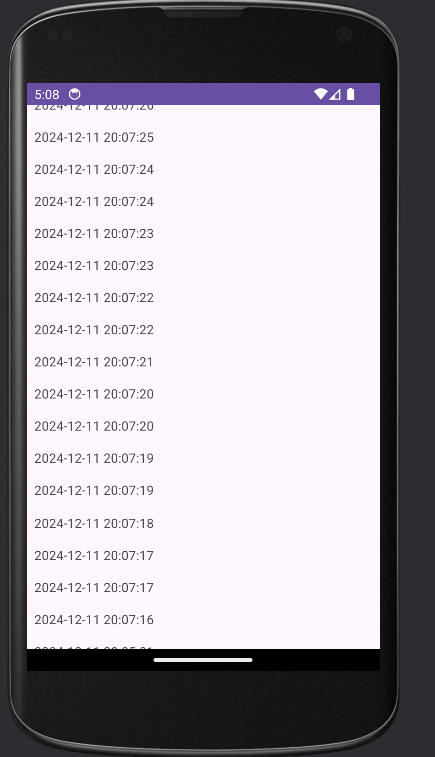


Рисунок 19. Проверка выполнения работы RecyclerView

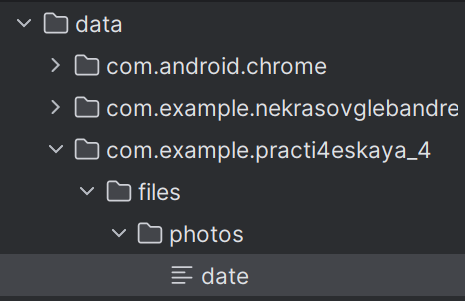


Рисунок 20. Проверка успешного создания файла для списка

## Вывод

В процессе выполнения работы мы создали приложение, которое позволяет пользователю использовать камеру с возможностью сохранения данных в списке

# ПРАКТИЧЕСКАЯ РАБОТА №5-6

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.

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

## Код практического задания

В ходе выполнения работы, мы написали приложение с функциями получения и отображения данных с внешнего API, а также реализовали реализовать Dependency Injection (см. Листинги 20-29)

Листинг 20. App.kt

|  |
| --- |
| package com.example.pr\_5  import android.app.Application import dagger.hilt.android.HiltAndroidApp  @HiltAndroidApp class App:Application() {  } |

Листинг 21. AppDatabase.kt

|  |
| --- |
| package com.example.pr\_5  import androidx.room.Database import androidx.room.Room import androidx.room.RoomDatabase import android.content.Context import androidx.room.TypeConverters  //База данных @Database(entities = [Product::class], version = 1, exportSchema = false) @TypeConverters(Converters::class) // Здесь мы регистрируем конвертеры abstract class AppDatabase: RoomDatabase() {  abstract fun productDao(): ProductDao   companion object {  @Volatile  private var INSTANCE: AppDatabase? = null   fun getDatabase(context: Context): AppDatabase {  return INSTANCE ?: synchronized(this) {  val instance = Room.databaseBuilder(  context.applicationContext,  AppDatabase::class.java,  "app\_database"  ).build()  INSTANCE = instance  instance  }  }  } } |

Листинг 22. Converters.kt

|  |
| --- |
| package com.example.pr\_5  import androidx.room.TypeConverter import com.google.gson.Gson import com.google.gson.reflect.TypeToken  //сложные конветроты class Converters {  private val gson = Gson()   //преобразование сложных типов, которые не поддерживаются напрямую  @TypeConverter  //этот метод преобразует список строк List<String> в одну строку, разделяя элементы запятыми  fun fromStringList(value: List<String>?): String? {  return value?.joinToString(",")  }   @TypeConverter  //обратно в список  fun toStringList(value: String?): List<String>? {  return value?.split(",") ?: emptyList()  }   @TypeConverter  //габариты  fun fromDimensions(dimensions: Dimensions?): String? {  return dimensions?.let { "${it.width},${it.height},${it.depth}" }  }   @TypeConverter  fun toDimensions(value: String?): Dimensions? {  return value?.split(",")?.let {  if (it.size == 3) {  Dimensions(it[0].toFloat(), it[1].toFloat(), it[2].toFloat())  } else {  null  }  }  }   @TypeConverter  //отзыв  fun fromReviewList(reviews: List<Review>?): String? {  return gson.toJson(reviews)  }   @TypeConverter  fun toReviewList(value: String?): List<Review>? {  val listType = object : TypeToken<List<Review>>() {}.type  return gson.fromJson(value, listType)  }   @TypeConverter  fun fromMeta(meta: Meta?): String? {  return gson.toJson(meta)  }   @TypeConverter  fun toMeta(value: String?): Meta? {  return gson.fromJson(value, Meta::class.java)  } } |

Листинг 23. MainActivity.kt

|  |
| --- |
| package com.example.pr\_5  import android.content.Intent import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.widget.Button import android.widget.TextView import com.example.pr\_5.ProductApi import dagger.hilt.android.AndroidEntryPoint import jakarta.inject.Inject import kotlinx.coroutines.CoroutineScope import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.launch import retrofit2.Retrofit import retrofit2.converter.gson.GsonConverterFactory  @AndroidEntryPoint class MainActivity : AppCompatActivity() {  private lateinit var db: AppDatabase   @Inject  lateinit var retrofit: Retrofit   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_main)  val tv = findViewById<TextView>(R.id.tv)  val b = findViewById<Button>(R.id.button)  val buttonList = findViewById<Button>(R.id.buttonList)   //Инициализация локальной базы данных  db = AppDatabase.getDatabase(this)    val productApi = retrofit.create(ProductApi::class.java)   b.setOnClickListener {  //Используется для запуска фоновой задачи в потоке ввода-вывода (IO), чтобы не блокировать основной поток.  CoroutineScope(Dispatchers.IO).launch {  //Выполняет запрос к API для получения информации о продукте с id = 2  val product = productApi.getProductById(2)   // Сохранение в базу данных  val productEntity = Product(  id = product.id,  title = product.title,  description = product.description,  price = product.price,  discountPercentage = product.discountPercentage,  rating = product.rating,  stock = product.stock,  brand = product.brand,  category = product.category,  thumbnail = product.thumbnail,  availabilityStatus = product.availabilityStatus,  dimensions = product.dimensions,  images = product.images,  meta = product.meta,  minimumOrderQuantity = product.minimumOrderQuantity,  returnPolicy = product.returnPolicy,  reviews = product.reviews,  shippingInformation = product.shippingInformation,  sku = product.sku,  tags = product.tags,  warrantyInformation = product.warrantyInformation,  weight = product.weight)  //Сохранения объекта в локальную бд  db.productDao().insertProduct(productEntity)    //Обновление пользовательского интерфейса  runOnUiThread {  tv.text = product.title  }  }  }  buttonList.setOnClickListener {  startActivity(Intent(this, ProductListActivity::class.java))  }   } } |

Листинг 24. Module.kt

|  |
| --- |
| package com.example.pr\_5  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) //тоже имеет доступ из любого места в коде object Module {   @Provides  @Singleton //доступ, ко всем классам  fun provideRetrofit(): Retrofit = Retrofit.Builder()  .baseUrl("https://dummyjson.com")  .addConverterFactory(GsonConverterFactory.create())  .build() } |

Листинг 25. Product.kt

|  |
| --- |
| package com.example.pr\_5  import androidx.room.Entity import androidx.room.PrimaryKey import jakarta.inject.Inject  //поля, продуктов @Entity(tableName = "products") data class Product (  @PrimaryKey val id: Int,  val title: String,  val description: String,  val price: Float,  val discountPercentage: Float,  val rating: Float,  val stock: Int,  val brand: String,  val category: String,  val thumbnail: String,  val images: List<String>,  val tags: List<String>,  val sku: String,  val weight: Float,  val dimensions: Dimensions,  val warrantyInformation: String,  val shippingInformation: String,  val availabilityStatus: String,  val reviews: List<Review>,  val returnPolicy: String,  val minimumOrderQuantity: Int,  val meta: Meta )  data class Dimensions(  val width: Float,  val height: Float,  val depth: Float )  data class Review(  val rating: Int,  val comment: String,  val date: String,  val reviewerName: String,  val reviewerEmail: String )  data class Meta(  val createdAt: String,  val updatedAt: String,  val barcode: String,  val qrCode: String ) |

Листинг 26. ProductAdapter.kt

|  |
| --- |
| package com.example.pr\_5  import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import android.widget.TextView import androidx.recyclerview.widget.RecyclerView  //Адаптер для продукта class ProductAdapter(private val products: List<Product>) :  RecyclerView.Adapter<ProductAdapter.ProductViewHolder>() {   class ProductViewHolder(view: View) : RecyclerView.ViewHolder(view) {  val title: TextView = view.findViewById(R.id.productTitle)  val description: TextView = view.findViewById(R.id.productDescription)  val price: TextView = view.findViewById(R.id.productPrice)  }   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ProductViewHolder {  val view = LayoutInflater.from(parent.context)  .inflate(R.layout.product\_item, parent, false)  return ProductViewHolder(view)  }   override fun onBindViewHolder(holder: ProductViewHolder, position: Int) {  val product = products[position]  holder.title.text = product.title  holder.description.text = product.description  holder.price.text = "Price: $${product.price}"  }   override fun getItemCount() = products.size } |

Листинг 27. ProductApi.kt

|  |
| --- |
| package com.example.pr\_5  import retrofit2.http.GET import retrofit2.http.Path  interface ProductApi{  //HTTP-аннотация Retrofit, которая указывает, что метод будет делать GET-запрос на сервер  @GET("products/{id}")  //объявление асинхронного метода  suspend fun getProductById(@Path("id") id: Int): Product } |

Листинг 28. ProductDao.kt

|  |
| --- |
| package com.example.pr\_5  import androidx.room.Dao import androidx.room.Insert import androidx.room.OnConflictStrategy import androidx.room.Query // @Dao interface ProductDao {  @Insert(onConflict = OnConflictStrategy.REPLACE)  suspend fun insertProduct(product: Product)   @Query("SELECT \* FROM products")  suspend fun getAllProducts(): List<Product> } |

Листинг 29. ProductListActivity.kt

|  |
| --- |
| package com.example.pr\_5  import android.os.Bundle import android.util.Log import androidx.appcompat.app.AppCompatActivity import androidx.recyclerview.widget.LinearLayoutManager import androidx.recyclerview.widget.RecyclerView import dagger.hilt.android.AndroidEntryPoint import jakarta.inject.Inject import kotlinx.coroutines.CoroutineScope import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.launch  class ProductListActivity : AppCompatActivity() {   private lateinit var db: AppDatabase   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.activity\_product\_list)   db = AppDatabase.getDatabase(this)   val recyclerView = findViewById<RecyclerView>(R.id.recyclerView)  recyclerView.layoutManager = LinearLayoutManager(this)   CoroutineScope(Dispatchers.IO).launch {  val products = db.productDao().getAllProducts()  runOnUiThread {  recyclerView.adapter = ProductAdapter(products)  }  }  } } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 21-23)

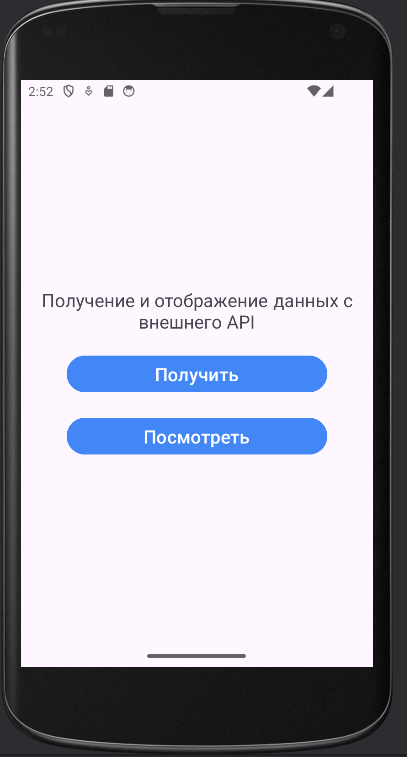


Рисунок 21. Запуск приложения

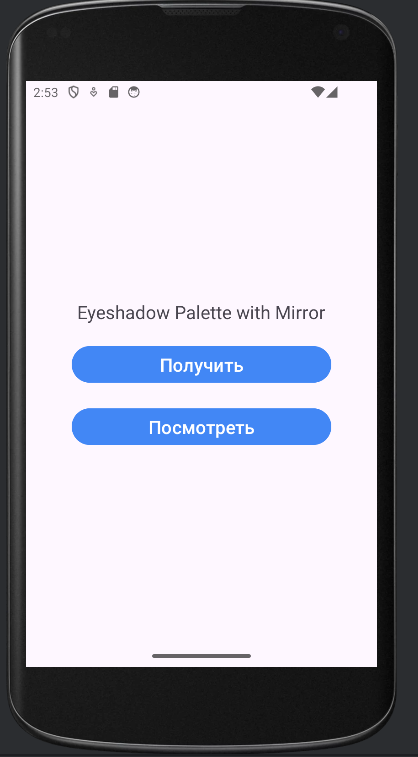


Рисунок 22. Получаем данные с внешнего API

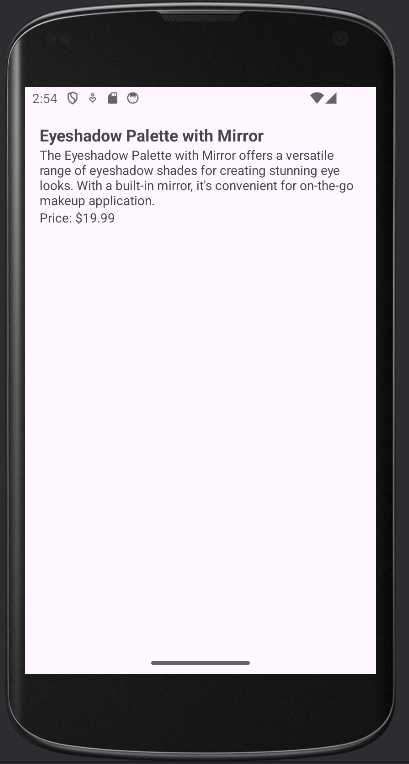


Рисунок 23. Просмотр данных источника API

## Вывод

В процессе выполнения работы мы создали приложение, которое позволяет пользователю получать и просматривать данные с внешнего API

# ПРАКТИЧЕСКАЯ РАБОТА №7-8

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:
4. В потоке Network необходимо выполнить загрузку изображения.
5. В потоке Disk сохранить загруженное изображение во внутренней памяти устройства.

Реализовать в приложении из практической работы №7:

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

## Код практического задания

В ходе выполнения работы, мы написали код для загрузки изображения из сети на экран, а также его сохранение во внутреннее хранилище устройства, а также написали модульные и пользовательские тесты (см. Листинги 30-33)

Листинг 30. MainActivity.kt

|  |
| --- |
| package com.example.pr78  import android.os.Bundle import android.widget.Button import android.widget.EditText import android.widget.ImageView import androidx.activity.enableEdgeToEdge import androidx.appcompat.app.AppCompatActivity import androidx.lifecycle.lifecycleScope import com.example.pr78.R import com.pr78.pr78.ImageRepository import com.squareup.picasso.Picasso import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.launch import kotlinx.coroutines.withContext  class MainActivity : AppCompatActivity() {   private lateinit var urlEditText: EditText  private lateinit var downloadButton: Button  private lateinit var imageView: ImageView  private val imageRepository = ImageRepository()   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  enableEdgeToEdge()  setContentView(R.layout.activity\_main)   urlEditText = findViewById(R.id.urlEditText)  downloadButton = findViewById(R.id.downloadButton)  imageView = findViewById(R.id.imageView)   downloadButton.setOnClickListener {  val imageUrl = urlEditText.text.toString()  if (imageUrl.isNotEmpty()) {  downloadImage(imageUrl)  }  }  }   private fun downloadImage(imageUrl: String) {  // Запуск двух потоков: загрузка и сохранение  lifecycleScope.launch(Dispatchers.IO) {  val imageBitmap = imageRepository.downloadImage(imageUrl)  if (imageBitmap != null) {  imageRepository.saveImageToDisk(applicationContext, imageBitmap)   withContext(Dispatchers.Main) {  // Отображение изображения на ImageView  Picasso.get().load(imageUrl).into(imageView)  }  }  }  } } |

Листинг 31. ImageRepository.kt

|  |
| --- |
| package com.pr78.pr78  import android.content.Context import android.graphics.Bitmap import android.graphics.BitmapFactory import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.withContext import java.io.File import java.io.FileOutputStream import java.net.URL  class ImageRepository {   suspend fun downloadImage(imageUrl: String): Bitmap? {  return withContext(Dispatchers.IO) {  try {  val url = URL(imageUrl)  val connection = url.openConnection()  connection.connect()  val inputStream = connection.getInputStream()  return@withContext BitmapFactory.decodeStream(inputStream)  } catch (e: Exception) {  e.printStackTrace()  null  }  }  }   suspend fun saveImageToDisk(context: Context, bitmap: Bitmap) {  withContext(Dispatchers.IO) {  val filename = "downloaded\_image.png"  val file = File(context.filesDir, filename)  FileOutputStream(file).use { out ->  bitmap.compress(Bitmap.CompressFormat.PNG, 100, out)  }  }  } } |

Листинг 32. MainActivityTest.kt

|  |
| --- |
| package com.pr78.pr78  import androidx.test.espresso.Espresso.onView import androidx.test.espresso.action.ViewActions.click import androidx.test.espresso.action.ViewActions.typeText import androidx.test.espresso.assertion.ViewAssertions.matches import androidx.test.espresso.matcher.ViewMatchers.hasErrorText import androidx.test.espresso.matcher.ViewMatchers.isDisplayed import androidx.test.espresso.matcher.ViewMatchers.withId import androidx.test.ext.junit.rules.ActivityScenarioRule import com.example.pr78.MainActivity import com.example.pr78.R import org.junit.Rule import org.junit.Test  class MainActivityTest {   @get:Rule  var activityRule = ActivityScenarioRule(MainActivity::class.java)   @Test  fun checkEditTextIsDisplayed() {  onView(withId(R.id.urlEditText)).check(matches(isDisplayed()))  }   @Test  fun checkButtonIsDisplayed() {  onView(withId(R.id.downloadButton)).check(matches(isDisplayed()))  }   @Test  fun checkImageViewIsDisplayed() {  onView(withId(R.id.urlEditText)).perform(typeText("https://i.pinimg.com/736x/a2/2a/a6/a22aa63c501c8b010b4865e21b49b950.jpg"))  onView(withId(R.id.downloadButton)).perform(click())  Thread.sleep(2000)  onView(withId(R.id.imageView)).check(matches(isDisplayed()))  }   @Test  fun enterUrlAndDownloadImage() {  onView(withId(R.id.urlEditText)).perform(typeText("https://avatars.mds.yandex.net/i?id=4161e28d6e5490fc1c58446a737d43f37831f139b1247fde-10601011-images-thumbs&n=13"))  onView(withId(R.id.downloadButton)).perform(click())  Thread.sleep(5000)  onView(withId(R.id.imageView)).check(matches(isDisplayed()))  }   @Test  fun checkButtonClickWithoutUrl() {  onView(withId(R.id.urlEditText)).perform(typeText(""))  onView(withId(R.id.downloadButton)).perform(click())  } } |

Листинг 33. ImageRepositoryTest.kt

|  |
| --- |
| package com.pr78.pr78  import android.content.Context import android.graphics.Bitmap import android.graphics.BitmapFactory import androidx.test.core.app.ApplicationProvider import junit.framework.TestCase.assertEquals import junit.framework.TestCase.assertNotNull import junit.framework.TestCase.assertNull import junit.framework.TestCase.assertTrue import kotlinx.coroutines.runBlocking import org.junit.Test import java.io.File  class ImageRepositoryTest {   private val imageRepository = ImageRepository()  private val context = ApplicationProvider.getApplicationContext<Context>()   @Test  fun downloadImage\_validUrl\_returnsBitmap() = runBlocking {  val imageUrl = "https://avatars.mds.yandex.net/i?id=4161e28d6e5490fc1c58446a737d43f37831f139b1247fde-10601011-images-thumbs&n=13"  val bitmap = imageRepository.downloadImage(imageUrl)  assertNotNull(bitmap)  }   @Test  fun downloadImage\_invalidUrl\_returnsNull() = runBlocking {  val imageUrl = "https://invalid-url.com/image.png"  val bitmap = imageRepository.downloadImage(imageUrl)  assertNull(bitmap)  }   @Test  fun saveImageToDisk\_savesSuccessfully() = runBlocking {  val bitmap = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB\_8888)  imageRepository.saveImageToDisk(context, bitmap)  val file = File(context.filesDir, "downloaded\_image.png")  assertTrue(file.exists())  }   @Test  fun saveImageToDisk\_fileIsSavedCorrectly() = runBlocking {  val bitmap = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB\_8888)  imageRepository.saveImageToDisk(context, bitmap)  val file = File(context.filesDir, "downloaded\_image.png")  val savedBitmap = BitmapFactory.decodeFile(file.path)  assertNotNull(savedBitmap)  }   @Test  fun saveImageToDisk\_overwritesExistingFile() = runBlocking {  val bitmap1 = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB\_8888)  val bitmap2 = Bitmap.createBitmap(200, 200, Bitmap.Config.ARGB\_8888)  imageRepository.saveImageToDisk(context, bitmap1)  imageRepository.saveImageToDisk(context, bitmap2)  val file = File(context.filesDir, "downloaded\_image.png")  val savedBitmap = BitmapFactory.decodeFile(file.path)  assertEquals(200, savedBitmap.width)  assertEquals(200, savedBitmap.height)  } } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 24-28)



Рисунок 24. Запуск приложения

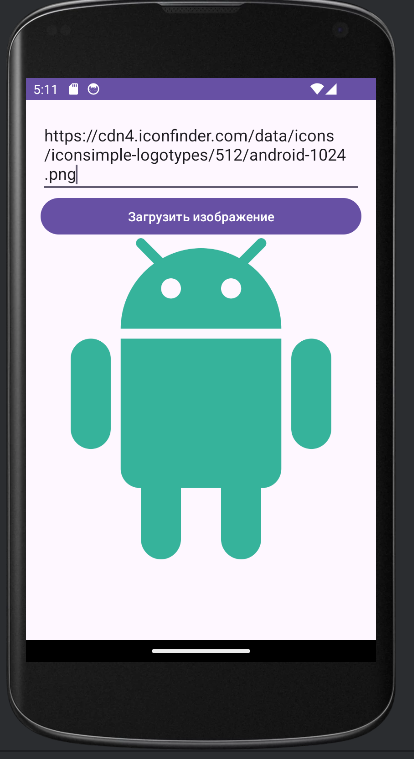


Рисунок 25. Загрузка изображения из интернета

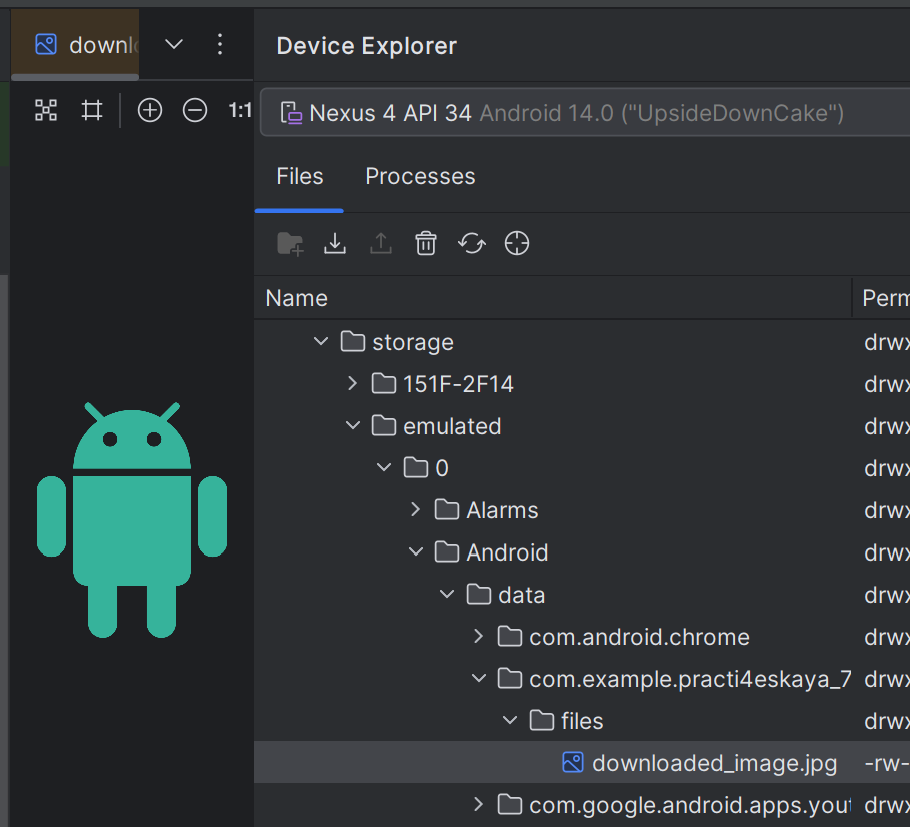


Рисунок 26. Проверка внутреннего хранилища устройства

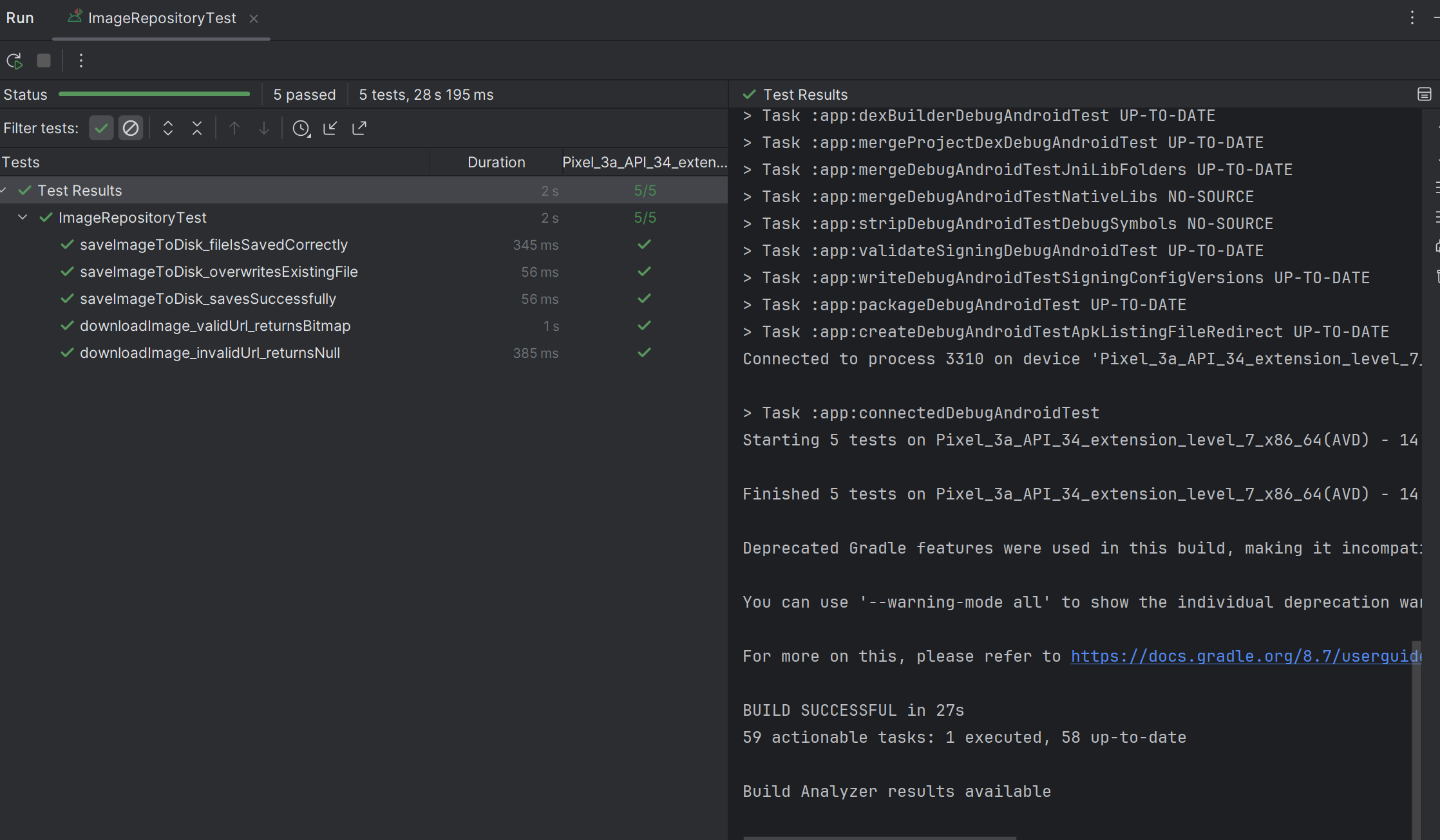


Рисунок 27. Запуск модульных тестов

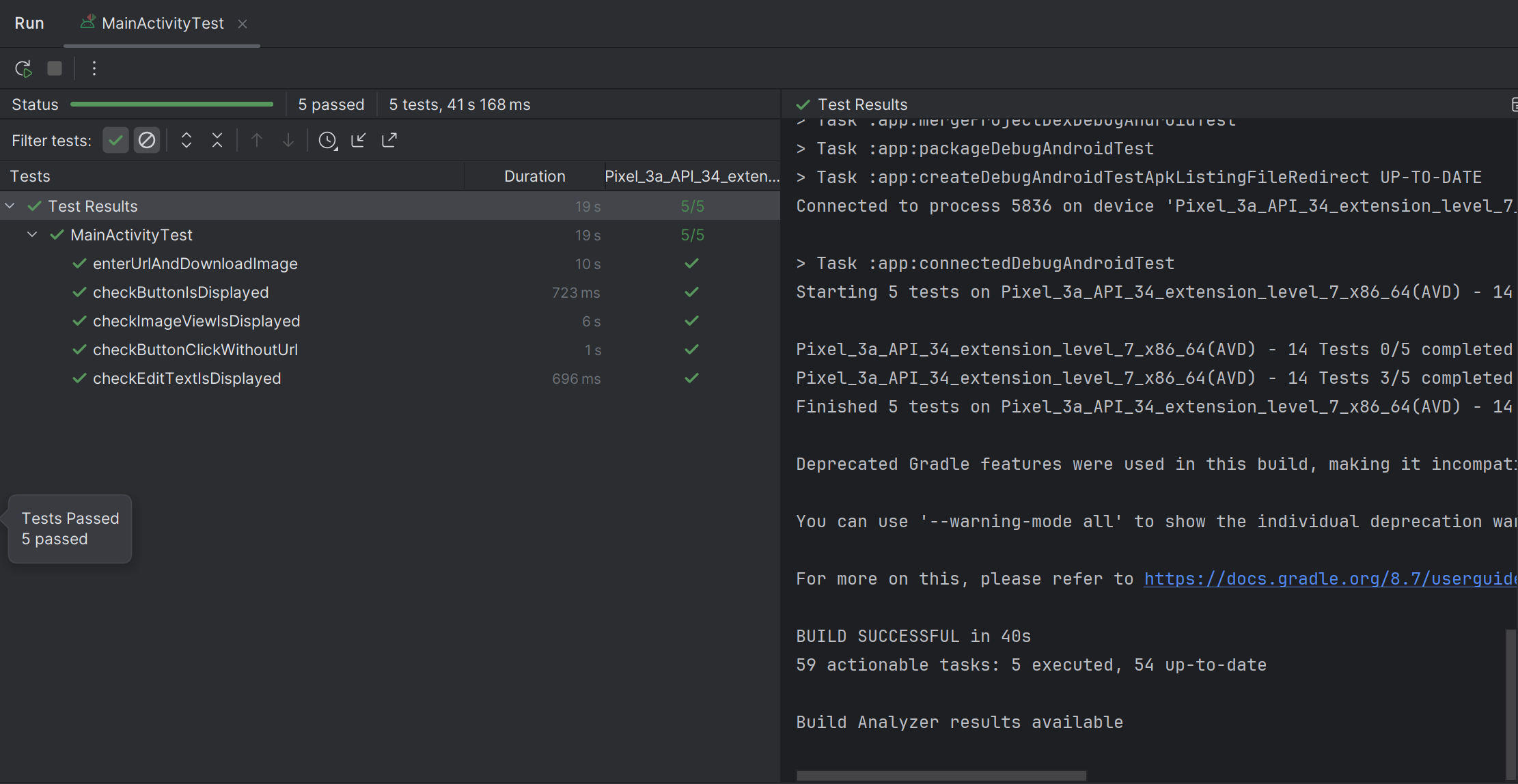


Рисунок 28. Запуск UI тестов

## Вывод

В ходе выполнения работы, мы написали код для загрузки изображения из сети на экран, а также его сохранение во внутреннее хранилище устройства, а также написали модульные и пользовательские тесты

# ПРАКТИЧЕСКАЯ РАБОТА №9

Реализовать приложение, которое выводит ФИО и группу студента, также добавить свой шрифт, цвет и т.д. используя Compose

## Код практического задания

В ходе выполнения работы, мы написали оформление для нашего приложения, а также вывели ФИО и группу (см. Листинг 34)

Листинг 34. MainActivity.kt

|  |
| --- |
| package com.example.practi4eskaya\_9  import android.os.Bundle import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.layout.\* import androidx.compose.foundation.layout.Box import androidx.compose.material3.\* import androidx.compose.runtime.Composable import androidx.compose.ui.Alignment import androidx.compose.ui.Modifier import androidx.compose.ui.graphics.Color import androidx.compose.ui.res.painterResource import androidx.compose.ui.text.TextStyle import androidx.compose.ui.text.font.Font import androidx.compose.ui.text.font.FontFamily import androidx.compose.ui.text.font.FontWeight import androidx.compose.ui.text.style.TextAlign import androidx.compose.ui.tooling.preview.Preview import androidx.compose.ui.unit.sp import androidx.compose.ui.unit.dp  // Кастомный шрифт val CustomFontFamily = FontFamily(  Font(R.font.roboto\_regular, FontWeight.Normal),  Font(R.font.roboto\_regular, FontWeight.Bold) )  // Цветовая схема val CustomColorScheme = lightColorScheme(  primary = Color(0xFF6200EA),  onPrimary = Color.White,  secondary = Color(0xFF03DAC5),  background = Color(0xFFF3F4F6),  onBackground = Color.Black )  // Типографика с кастомным шрифтом val CustomTypography = Typography(  bodyLarge = TextStyle(  fontSize = 20.sp,  fontFamily = CustomFontFamily,  fontWeight = FontWeight.Bold,  textAlign = TextAlign.Center,  color = Color.Black  ) )  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContent {  MaterialTheme(  colorScheme = CustomColorScheme,  typography = CustomTypography  ) {  StudentInfoScreen()  }  }  } }  @Composable fun StudentInfoScreen() {  Box(  modifier = Modifier  .fillMaxSize()  ) {  // Фон изображения  Image(  painter = painterResource(id = R.drawable.background\_image), // Укажите свой фон в ресурсах  contentDescription = null,  modifier = Modifier.fillMaxSize()  )  // Центрирование текста  Text(  text = "ФИО: Некрасов Глеб Андреевич\nНомер группы: ИКБО-28-22",  style = MaterialTheme.typography.bodyLarge,  modifier = Modifier.align(Alignment.Center)  )  } }  @Preview(showBackground = true) @Composable fun PreviewStudentInfoScreen() {  MaterialTheme(  colorScheme = CustomColorScheme,  typography = CustomTypography  ) {  StudentInfoScreen()  } } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунок 29)



Рисунок 29. Запуск приложения

## Вывод

В процессе выполнения работы мы научились оформлять наше приложение

# ПРАКТИЧЕСКАЯ РАБОТА №10-12

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).

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.

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

## Код практического задания

В ходе выполнения работы, мы написали приложение по выполнению всех пунктов практических №10-12 (см. Листинги 35-36)

Листинг 35. ImageRepository.kt

|  |
| --- |
| package com.example.practi4eskaya1011\_test  import android.content.Context import android.graphics.Bitmap import android.graphics.BitmapFactory import android.util.Log import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.withContext import java.io.File import java.io.FileOutputStream import java.io.InputStream import java.net.HttpURLConnection import java.net.URL  class ImageRepository(private val context: Context) {   suspend fun downloadImage(imageUrl: String): Bitmap? {  return withContext(Dispatchers.IO) {  try {  val url = URL(imageUrl)  val connection: HttpURLConnection = url.openConnection() as HttpURLConnection  connection.doInput = true  connection.connect()  val inputStream: InputStream = connection.inputStream  BitmapFactory.decodeStream(inputStream)  } catch (e: Exception) {  Log.e("ImageRepository", "Error downloading image", e)  null  }  }  }   suspend fun saveImageToInternalStorage(bitmap: Bitmap, filename: String): Boolean {  return withContext(Dispatchers.IO) {  try {  val file = File(context.filesDir, "$filename.jpg")  val fos = FileOutputStream(file)  bitmap.compress(Bitmap.CompressFormat.JPEG, 100, fos)  fos.flush()  fos.close()  true  } catch (e: Exception) {  Log.e("ImageRepository", "Error saving image", e)  false  }  }  } } |

Листинг 36. MainActivity.kt

|  |
| --- |
| package com.example.practi4eskaya1011\_test  import android.annotation.SuppressLint import android.os.Bundle import android.widget.Toast import androidx.activity.ComponentActivity import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.border import androidx.compose.foundation.layout.Arrangement import androidx.compose.foundation.layout.Box 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.lazy.LazyColumn import androidx.compose.foundation.text.BasicTextField import androidx.compose.material3.Button import androidx.compose.material3.ExperimentalMaterial3Api import androidx.compose.material3.MaterialTheme import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.material3.TopAppBar 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.TextFieldValue import androidx.compose.ui.unit.dp import kotlinx.coroutines.launch import androidx.compose.ui.res.painterResource import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.border import androidx.compose.foundation.layout.\* import androidx.compose.foundation.lazy.LazyColumn import androidx.compose.material3.Button import androidx.compose.material3.Scaffold import androidx.compose.material3.Text import androidx.compose.material3.TopAppBar import androidx.compose.runtime.\* import androidx.compose.ui.graphics.asImageBitmap import androidx.compose.ui.platform.LocalContext import androidx.compose.ui.unit.dp import kotlinx.coroutines.Dispatchers import kotlinx.coroutines.launch import kotlinx.coroutines.withContext import androidx.activity.compose.setContent import androidx.compose.foundation.Image import androidx.compose.foundation.border import androidx.compose.foundation.layout.\* import androidx.compose.foundation.lazy.LazyColumn import androidx.compose.material.icons.Icons import androidx.compose.material.icons.filled.Home import androidx.compose.material.icons.filled.List import androidx.compose.material.icons.filled.Menu import androidx.compose.material3.\* import androidx.compose.runtime.\* import androidx.compose.ui.graphics.asImageBitmap import androidx.compose.ui.platform.LocalContext import androidx.compose.ui.unit.dp import androidx.work.OneTimeWorkRequestBuilder import androidx.work.WorkManager import androidx.work.Worker import androidx.work.WorkerParameters import kotlinx.coroutines.launch import java.util.concurrent.TimeUnit  class MainActivity : ComponentActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContent {  MyApp()  }  } }  @SuppressLint("UnusedMaterial3ScaffoldPaddingParameter") @OptIn(ExperimentalMaterial3Api::class) @Composable fun MyApp() {  val context = LocalContext.current  val imageRepository = remember { ImageRepository(context) }  var imageUrl by remember { mutableStateOf(TextFieldValue("")) }  val coroutineScope = rememberCoroutineScope()   // Список изображений  var imageList by remember { mutableStateOf<List<android.graphics.Bitmap>>(emptyList()) }   // Состояние текущего выбранного экрана  var selectedScreen by remember { mutableStateOf("Home") }   // Состояние для открытия/закрытия Drawer  val drawerState = rememberDrawerState(initialValue = DrawerValue.Closed)  val coroutineScopeDrawer = rememberCoroutineScope()   // Основной Scaffold  ModalNavigationDrawer(  drawerState = drawerState,  drawerContent = {  ModalDrawerSheet {  Text(  text = "Navigation",  modifier = Modifier.padding(16.dp),  style = MaterialTheme.typography.titleMedium  )  Spacer(modifier = Modifier.height(8.dp))  NavigationDrawerItem(  label = { Text("Home") },  selected = selectedScreen == "Home",  onClick = {  selectedScreen = "Home"  coroutineScopeDrawer.launch { drawerState.close() }  }  )  NavigationDrawerItem(  label = { Text("Gallery") },  selected = selectedScreen == "Gallery",  onClick = {  selectedScreen = "Gallery"  coroutineScopeDrawer.launch { drawerState.close() }  }  )  }  }  ) {  Scaffold(  topBar = {  TopAppBar(  title = { Text("Dynamic Image List") },  navigationIcon = {  IconButton(onClick = { coroutineScopeDrawer.launch { drawerState.open() } }) {  Icon(Icons.Default.Menu, contentDescription = "Menu")  }  }  )  },  bottomBar = {  BottomAppBar {  NavigationBar {  NavigationBarItem(  icon = { Icon(Icons.Default.Home, contentDescription = "Home") },  label = { Text("Home") },  selected = selectedScreen == "Home",  onClick = { selectedScreen = "Home" }  )  NavigationBarItem(  icon = { Icon(Icons.Default.List, contentDescription = "Home") },  label = { Text("Gallery") },  selected = selectedScreen == "Gallery",  onClick = { selectedScreen = "Gallery" }  )  }  }  },  content = {  when (selectedScreen) {  "Home" -> HomeScreen(  imageUrl = imageUrl,  onImageUrlChange = { imageUrl = it },  imageList = imageList,  onAddImage = { bitmap ->  imageList = imageList + bitmap  imageUrl = TextFieldValue("") // Очищаем поле ввода  },  imageRepository = imageRepository  )  "Gallery" -> GalleryScreen(imageList = imageList)  }  }  )  }   // Добавление WorkManager  WorkManager.getInstance(context).enqueue(  OneTimeWorkRequestBuilder<ImageTaskWorker>()  .setInitialDelay(15, TimeUnit.MINUTES) // Выполнение задачи через 15 минут после запуска  .build()  ) }  @Composable fun HomeScreen(  imageUrl: TextFieldValue,  onImageUrlChange: (TextFieldValue) -> Unit,  imageList: List<android.graphics.Bitmap>,  onAddImage: (android.graphics.Bitmap) -> Unit,  imageRepository: ImageRepository ) {  val context = LocalContext.current  val coroutineScope = rememberCoroutineScope()   Column(  modifier = Modifier  .fillMaxSize()  .padding(16.dp),  verticalArrangement = Arrangement.spacedBy(16.dp)  ) {  BasicTextField(  value = imageUrl,  onValueChange = { onImageUrlChange(it) },  modifier = Modifier  .fillMaxWidth()  .padding(8.dp)  .height(50.dp)  .border(1.dp, MaterialTheme.colorScheme.primary)  .padding(8.dp),  decorationBox = { innerTextField ->  Box(Modifier.fillMaxSize()) {  if (imageUrl.text.isEmpty()) {  Text(  "Enter image URL",  style = MaterialTheme.typography.bodySmall,  color = MaterialTheme.colorScheme.onSurfaceVariant  )  }  innerTextField()  }  }  )   Button(  onClick = {  if (imageUrl.text.isNotEmpty()) {  coroutineScope.launch {  val loadedImage = imageRepository.downloadImage(imageUrl.text)  if (loadedImage != null) {  onAddImage(loadedImage)  Toast.makeText(context, "Image added to list", Toast.LENGTH\_SHORT).show()  } else {  Toast.makeText(context, "Error loading image", Toast.LENGTH\_SHORT).show()  }  }  }  },  modifier = Modifier  .fillMaxWidth()  .height(50.dp)  ) {  Text(text = "Add Image to List")  }  } }  @Composable fun GalleryScreen(imageList: List<android.graphics.Bitmap>) {  LazyColumn(  modifier = Modifier.fillMaxSize(),  verticalArrangement = Arrangement.spacedBy(16.dp)  ) {  items(imageList.size) { index ->  val bitmap = imageList[index]  Box(  modifier = Modifier  .fillMaxWidth()  .height(300.dp),  contentAlignment = Alignment.Center  ) {  Image(  bitmap = bitmap.asImageBitmap(),  contentDescription = null,  modifier = Modifier.fillMaxSize()  )  }  }  } }  class ImageTaskWorker(appContext: android.content.Context, workerParams: WorkerParameters) : Worker(appContext, workerParams) {  override fun doWork(): Result {  // Задача, которую будет выполнять WorkManager  return Result.success()  } }  //@Preview(showBackground = true) @Composable fun DefaultPreview() {  MyApp() } |

## Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 30-34)

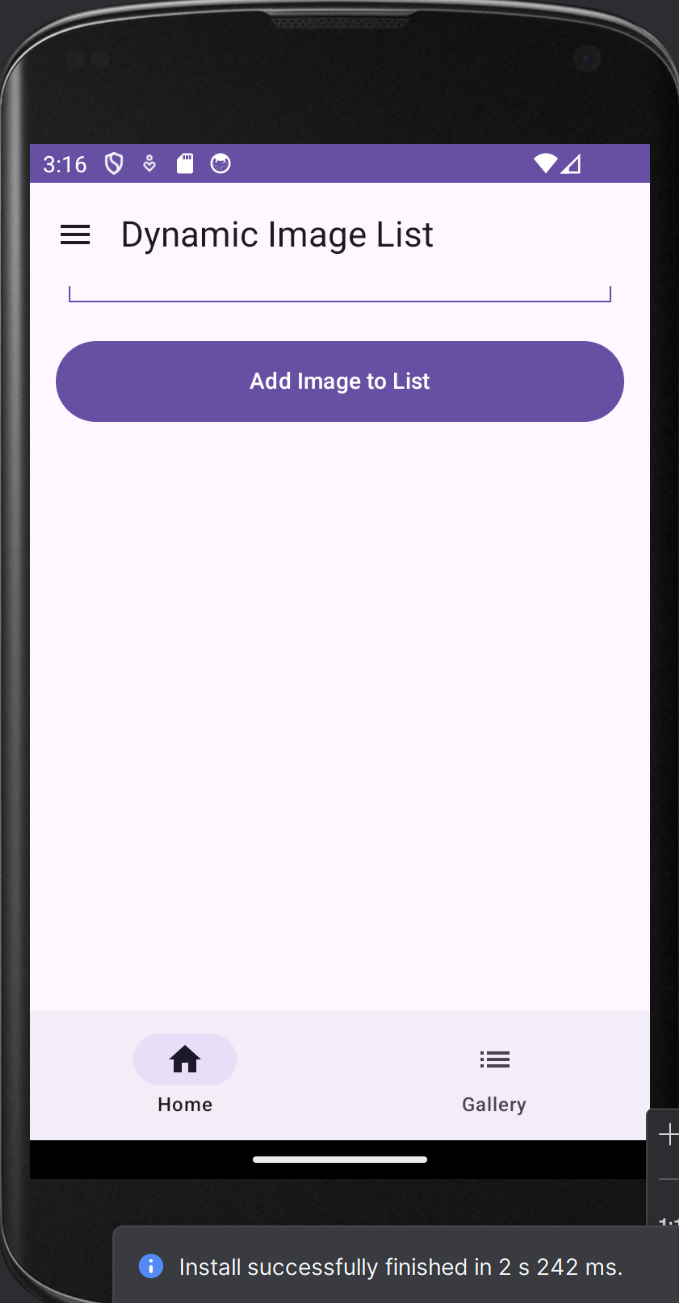


Рисунок 30. Запуск приложения

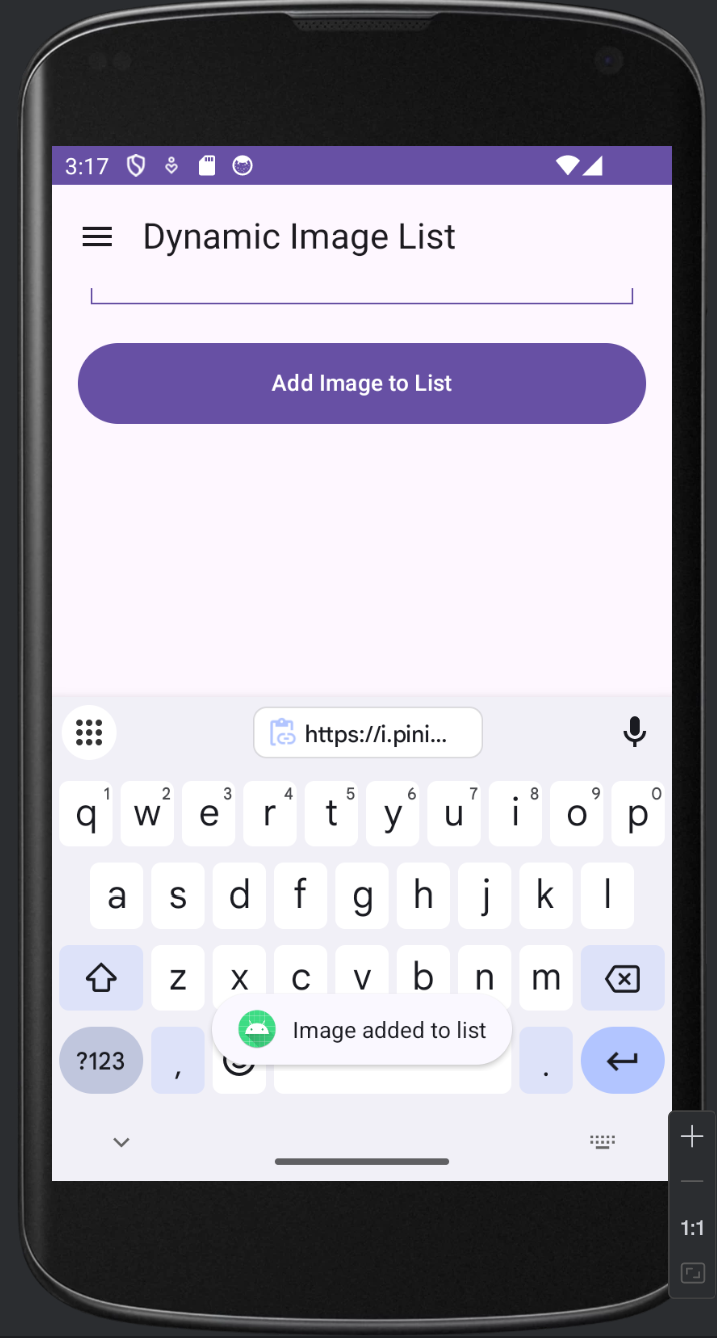


Рисунок 31. Добавили изображение в список

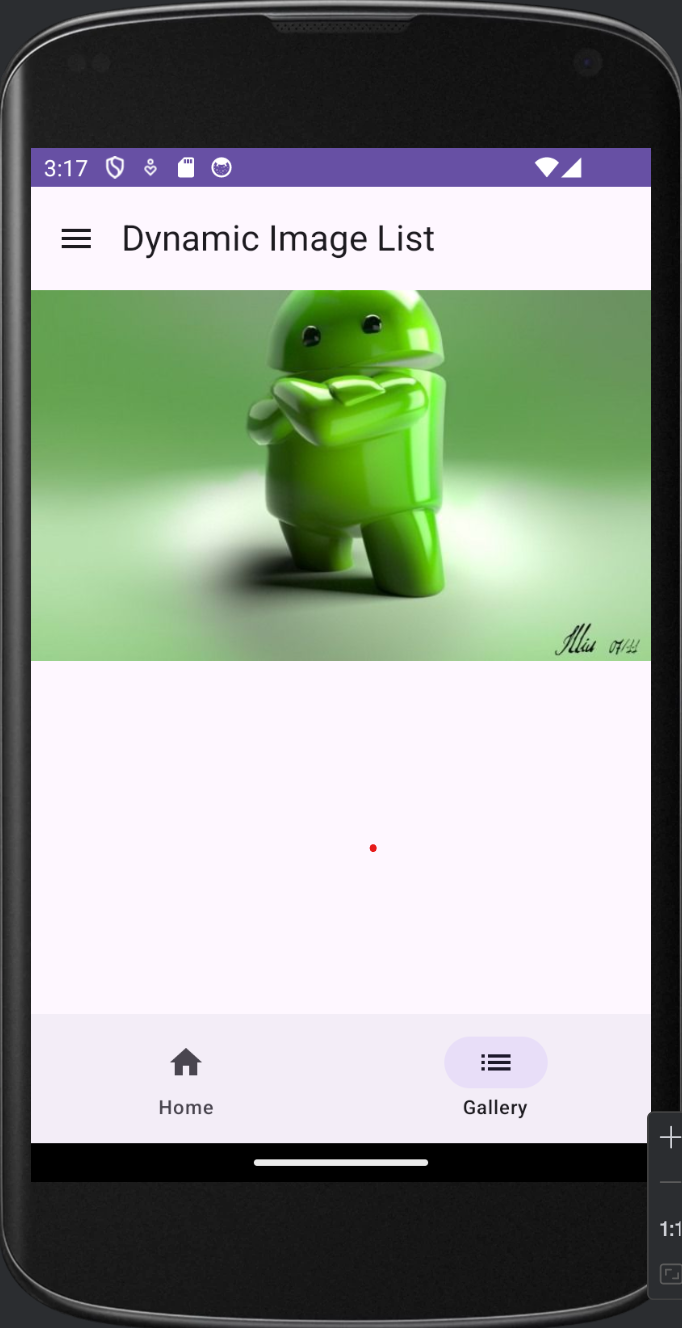


Рисунок 32. Проверка нашей галереи

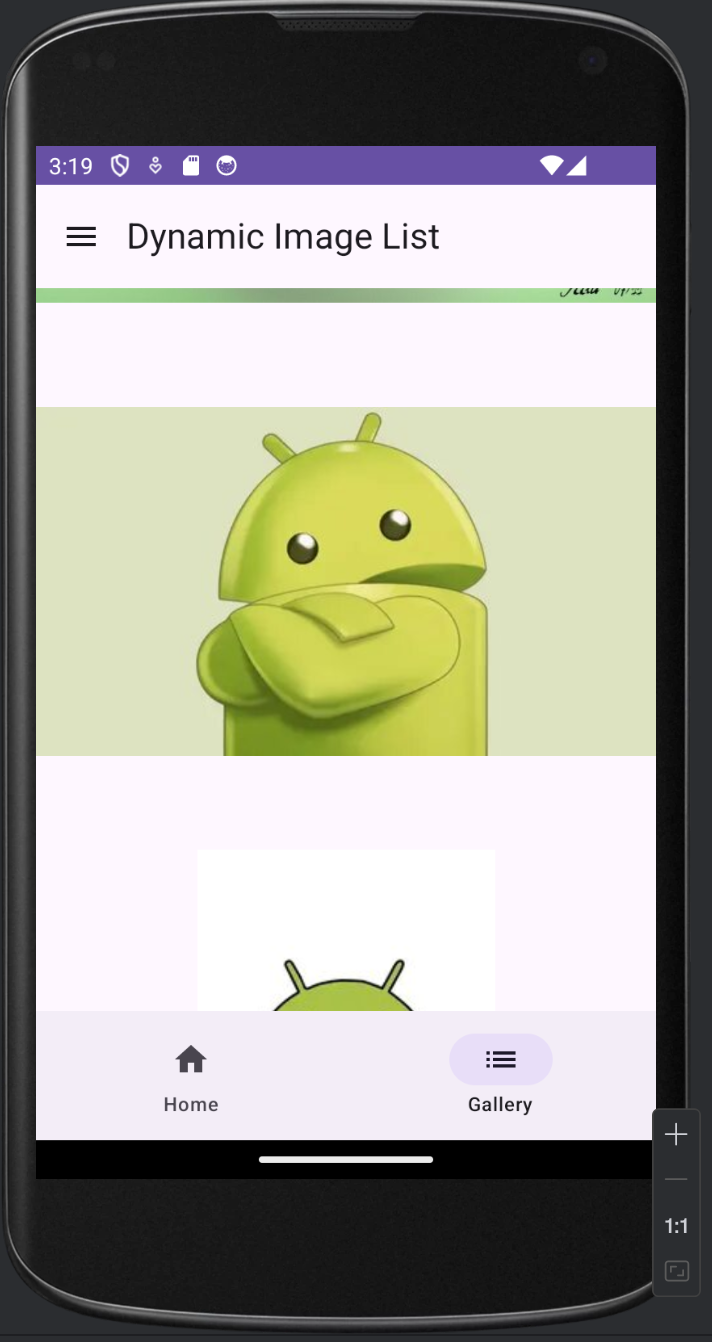


Рисунок 33. Проверка прокрутки галереи



Рисунок 34. Дополнительная навигация

## Вывод

В процессе выполнения работы мы научились работать с Compose, Scaffold, WorkManager

# ЗАКЛЮЧЕНИЕ

В ходе выполнения всех практик мы освоили значительные компоненты языка Kotlin