|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 №8**

по дисциплине «Разработка мобильных приложений»

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

Москва 2024 г.

СОДЕРЖАНИЕ

[ПРАКТИЧЕСКОЕ ЗАДАНИЕ 3](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915647)

[Код практического задания 4](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915648)

[Тестирование практического задания 5](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915649)

[Вывод 7](file:///D:\Desktop\Kotlin%20отчёты\Пр4\Отчёт_4_РМП_НекрасовГА.docx#_Toc184915650)

# ПРАКТИЧЕСКОЕ ЗАДАНИЕ

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 UI-тестов, а также модульных тестов (см. Листинги 1-2)

Листинг 1. 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)  } } |

Листинг 2. MainActivityTest

|  |
| --- |
| 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(2000)  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())  } } |

# Тестирование практического задания

Выполним проверку работы нашего кода (см. Рисунки 1-2)

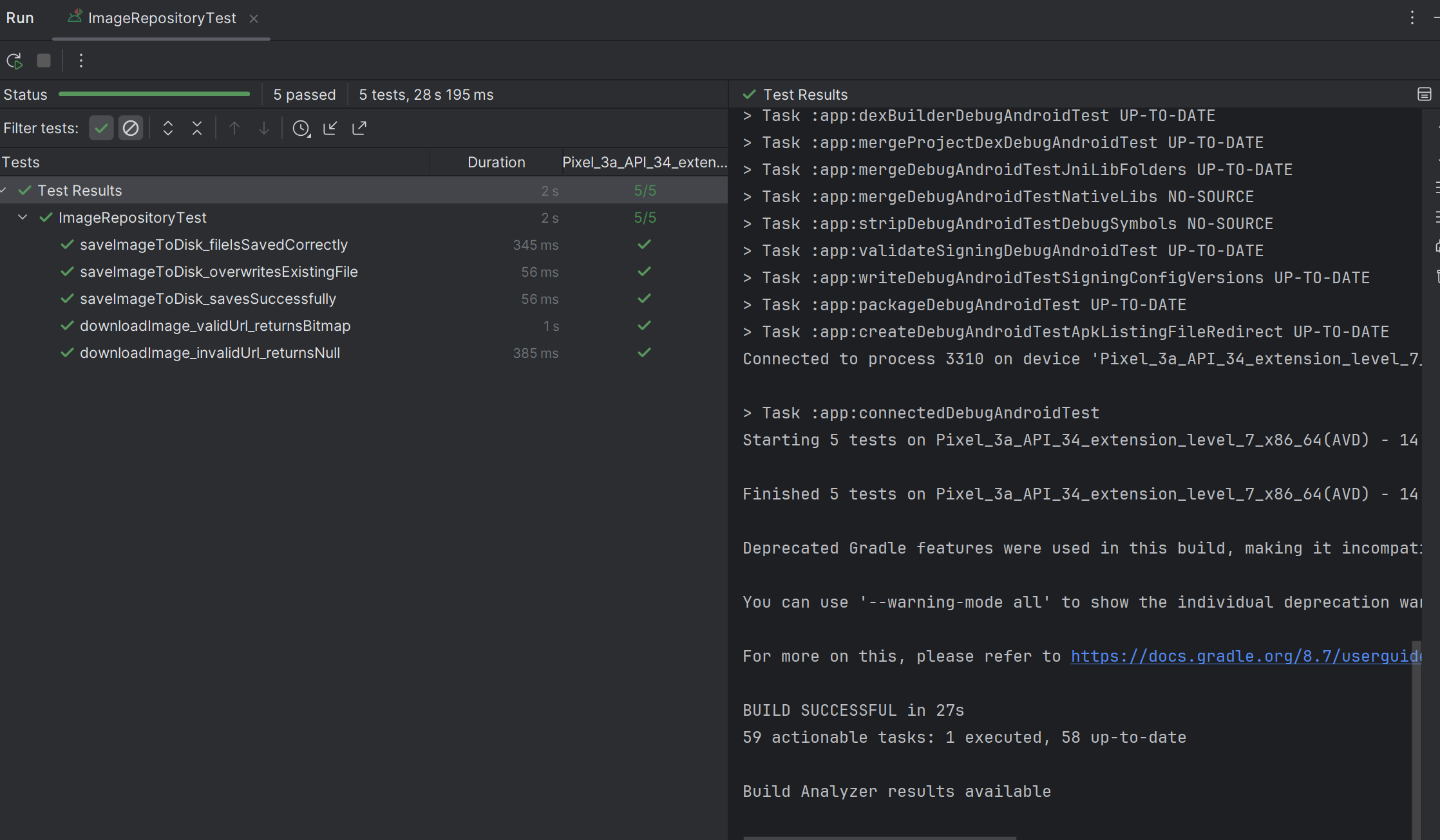


Рисунок 1. Запуск модульных тестов

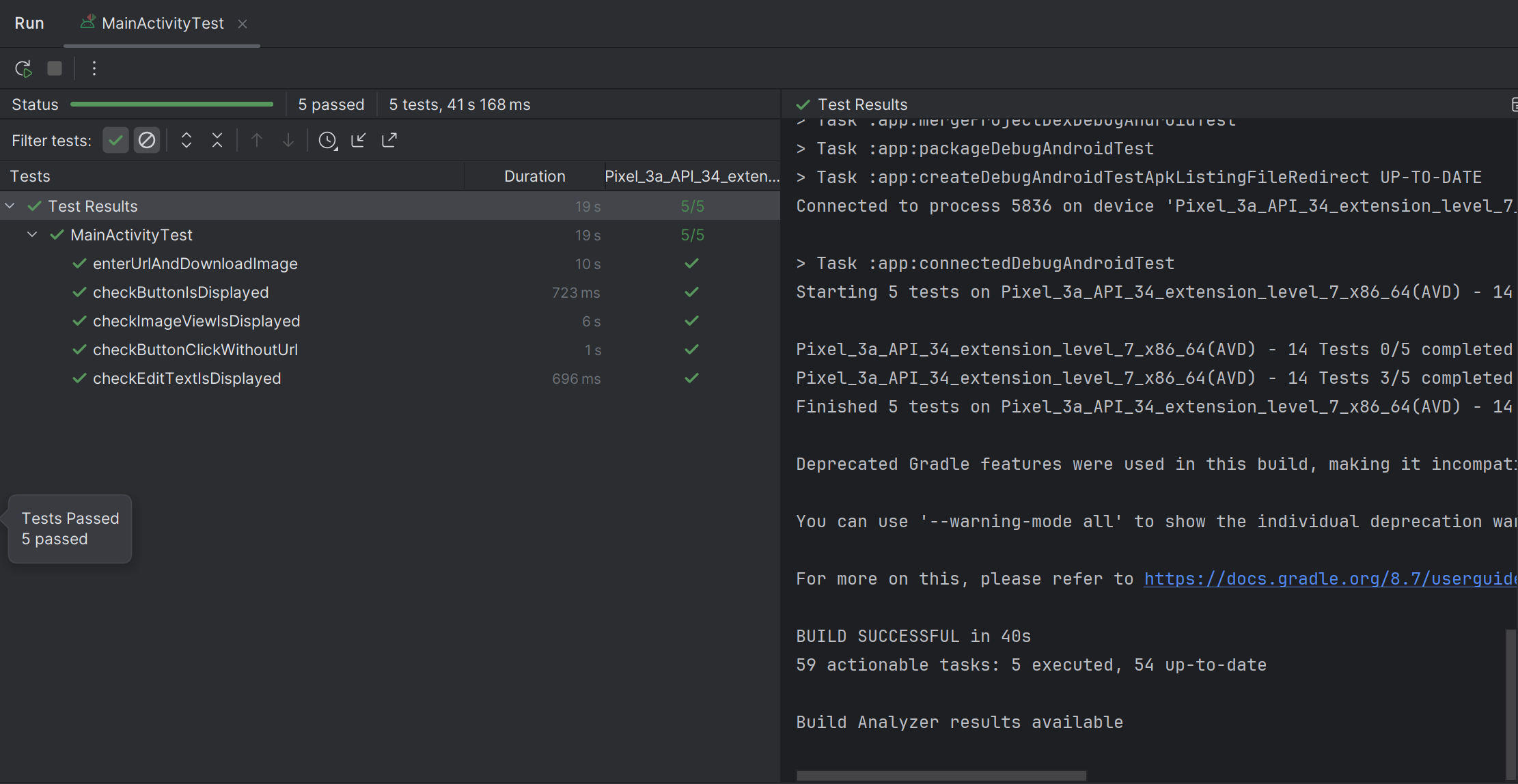


Рисунок 2. Запуск UI тестов

# Вывод

В процессе выполнения работы мы научились писать модульные а также UI тесты