

Университет ИТМО
Факультет программной инженерии и компьютерной техники

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
По дисциплине: “Разработка мобильных приложений”

Выполнил:
Студент группы Р33101
Макаров Глеб Вадимович

Преподаватель:
Ключев Аркадий Олегович

Санкт-Петербург
2023 г.

Оглавление	
Глава 1. Android Studio	3
Глава 2. Интерфейс.....	5
Глава 3. Код.....	14
Глава 4. APK.....	18
Глава 5. GitHub.....	23
Вывод.....	25
Список источников	26

Глава 1. Android Studio

Android Studio - это полностью интегрированная среда разработки (IDE), разработанная компанией Google для создания приложений Android. Она предоставляет полный набор инструментов и ресурсов для проектирования, разработки, тестирования и отладки приложений Android. Android Studio призвана упростить процесс разработки приложений и помочь разработчикам создавать высококачественные, эффективные и удобные приложения.

Android Studio используется для создания нативных приложений Android с использованием языков программирования Java, Kotlin и C++. Языки Java и Kotlin являются наиболее распространенными языками для разработки приложений Android, а C++ используется для разработки критически важного кода и нативных библиотек. Android Studio поддерживает широкий спектр версий Android SDK, от Android 2.2 (Froyo) до последней версии Android.

Android Studio была впервые выпущена в 2013 году, заменив Eclipse в качестве официальной IDE для разработки Android. Она построена на базе IntelliJ IDEA Community Edition, популярной Java IDE. С момента выпуска Android Studio стала стандартной IDE для разработки приложений для Android, и она регулярно обновляется новыми функциями и улучшениями.

Среди известных приложений для Android, которые были разработаны с помощью Android Studio, - WhatsApp, Instagram и Uber. Эти приложения демонстрируют универсальность и мощь Android Studio, а также ее способность справляться со сложными проектами по разработке приложений.

Android Studio поставляется с широким набором функций и инструментов, помогающих разработчикам создавать приложения для Android. Некоторые из наиболее заметных функций включают:

Визуальный редактор макетов: Мощный WYSIWYG-редактор (What You See Is What You Get) для проектирования макетов приложений с помощью перетаскиваемых компонентов пользовательского интерфейса.

Редактор кода: Полнофункциональный редактор кода с подсветкой ошибок и другими полезными функциями.

Система сборки Gradle: Гибкая система сборки, позволяющая разработчикам настраивать процесс сборки и управлять зависимостями.

Эмулятор: Встроенный эмулятор, позволяющий разработчикам тестировать свои приложения на различных устройствах и версиях Android.

Android Profiler: Мощный инструмент для анализа производительности приложений и выявления узких мест.

Android Virtual Device Manager: Инструмент для создания и управления виртуальными устройствами для тестирования приложений.

Android Asset Studio: Набор инструментов для создания иконок приложений, иконок пусковой установки и других графических элементов.

В целом, Android Studio - это незаменимый инструмент для разработчиков приложений для Android. Ее мощные функции и интуитивно понятный интерфейс делают ее популярным выбором для разработки приложений Android всех типов и размеров.

Отдельным пунктом стоит отметить наличие тёмной темы интерфейса.

После создания проекта нужно выбрать шаблон для моего проекта. Во-первых, нужно выбрать, под какую систему мы создаем проект: телефоны/планшеты, Wear OS, то есть под смарт-часы, Android TV, то есть телевизоры, и Automotive, то есть машины с соответствующим ПО.

Выбрав первый пункт, передо мной предстала тонна шаблонов. Я выбрал Empty Views Activity.

Далее я выбрал название проекта, выбрал Kotlin в качестве языка программирования и в качестве "Minimum SDK" выбрал Android 7.0. Такой проект запустится на 94.4% устройств, как мне сообщил Android Studio.

Глава 2. Интерфейс

После запуска проекта программа подгружает различные библиотеки и синхронизирует их, это занимает некоторое количество времени.

Так выглядит интерфейс при первом запуске:

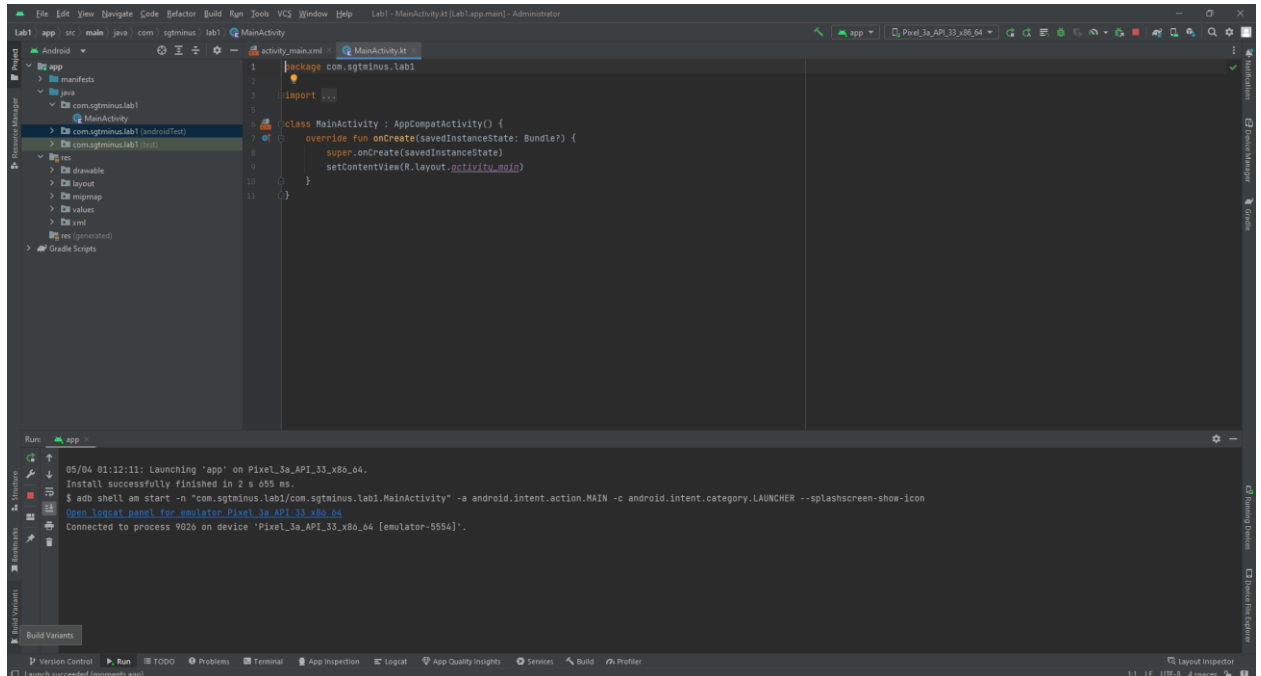


Рисунок 1

Несмотря на огромное количество кнопок, разобраться во всём этом не составит труда, всё по большей части структурировано.

Сразу же нажав кнопку запуска, я узнал, в данном шаблоне по умолчанию программа заключается в отображении надписи «Hello World!» на убивающем мои глаза белом фоне, то есть, по сути, в рамках данной лабораторной работы мне не нужно писать код и вносить какие-то изменения в интерфейс.

Далее я опишу некоторые пункты из интерфейса, которые, на мой взгляд, покрывают базовые потребности пользователя.

Итак, огромную часть экрана занимает текстовый редактор, в котором мы пишем код, задающий функционал нашей программы.

Далее я заметил визуальный редактор, открыв activity_main.xml. В нём мы оформляем дизайн приложения. Работа идет с файлами xml независимо от языка программирования, на котором ведется разработка функционала.

Здесь же можно указать абсолютно любой дизайн, добавлять различные объекты вроде кнопок или надписей, изображений, видео и т.д., расставлять

их, ну и, соответственно, дизайн будет полностью совпадать с тем, что вы видели в графическом редакторе.

Чтобы добавить новый объект, нужно выбрать его во вкладке Palette и перетащить на экран.

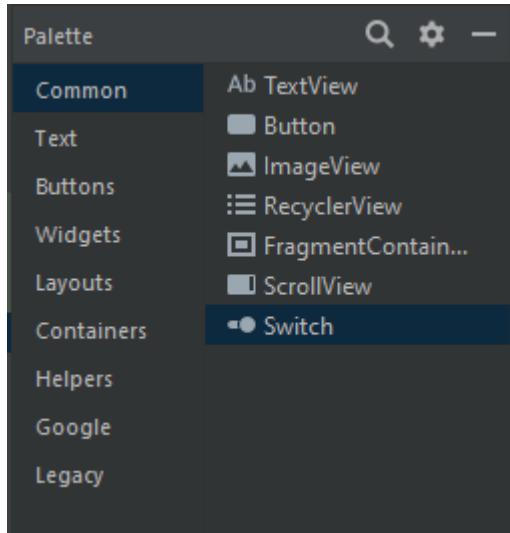


Рисунок 2

Характеристики объекта можно указать в панели справа

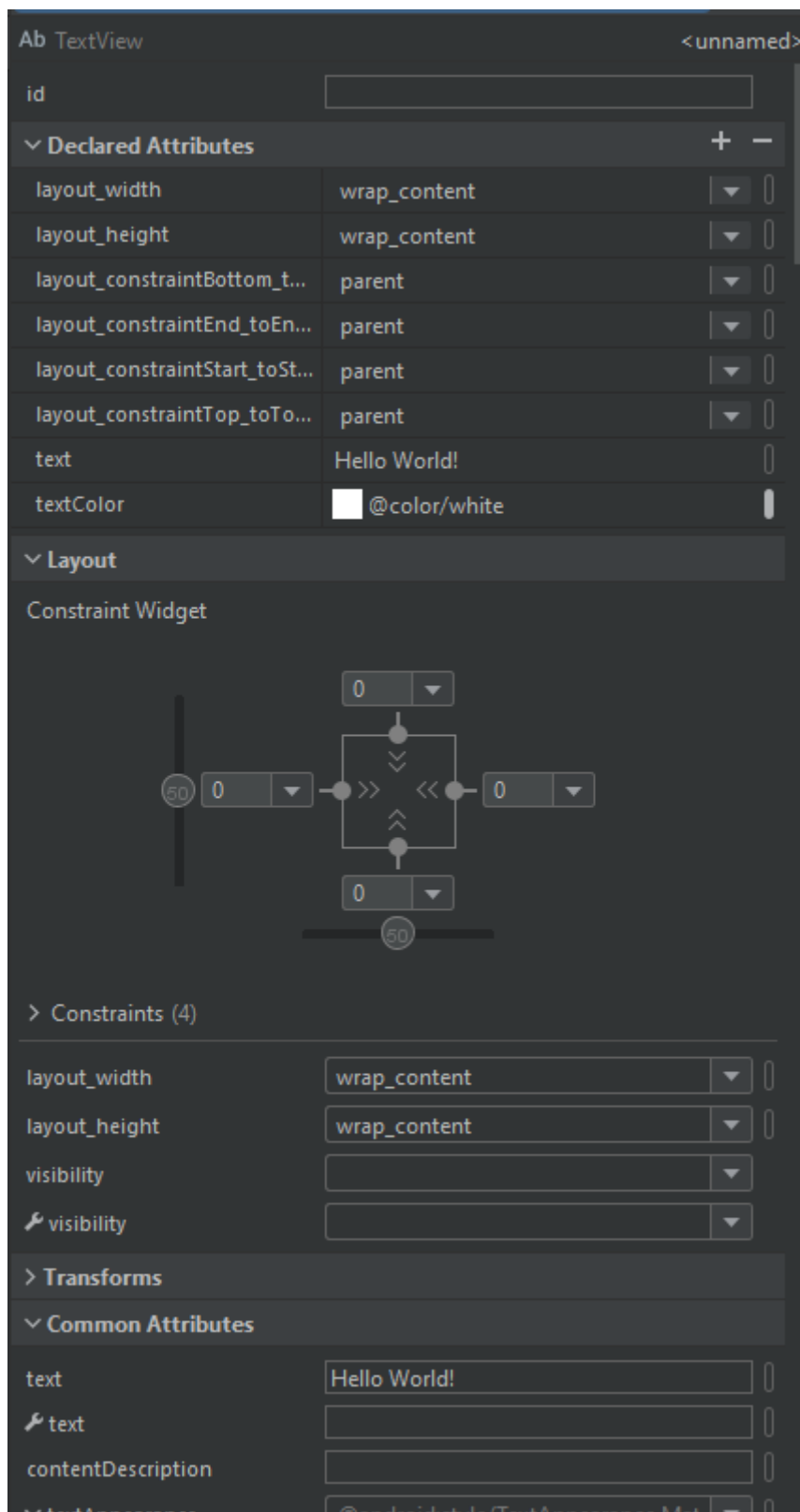


Рисунок 3

Воспользовавшись моментом, я сразу же поменял цвет фона на черный, а цвет текста на белый. Сделал я это посредством изменения параметра `textColor` у компонента `TextView`, а у компонента `ConstraintLayout` я изменил параметр `background`. Замечу, что при первой смене цвета, когда у обоих

компонентов цвета стали визуально похожими, Android Studio оповестил меня об этом.

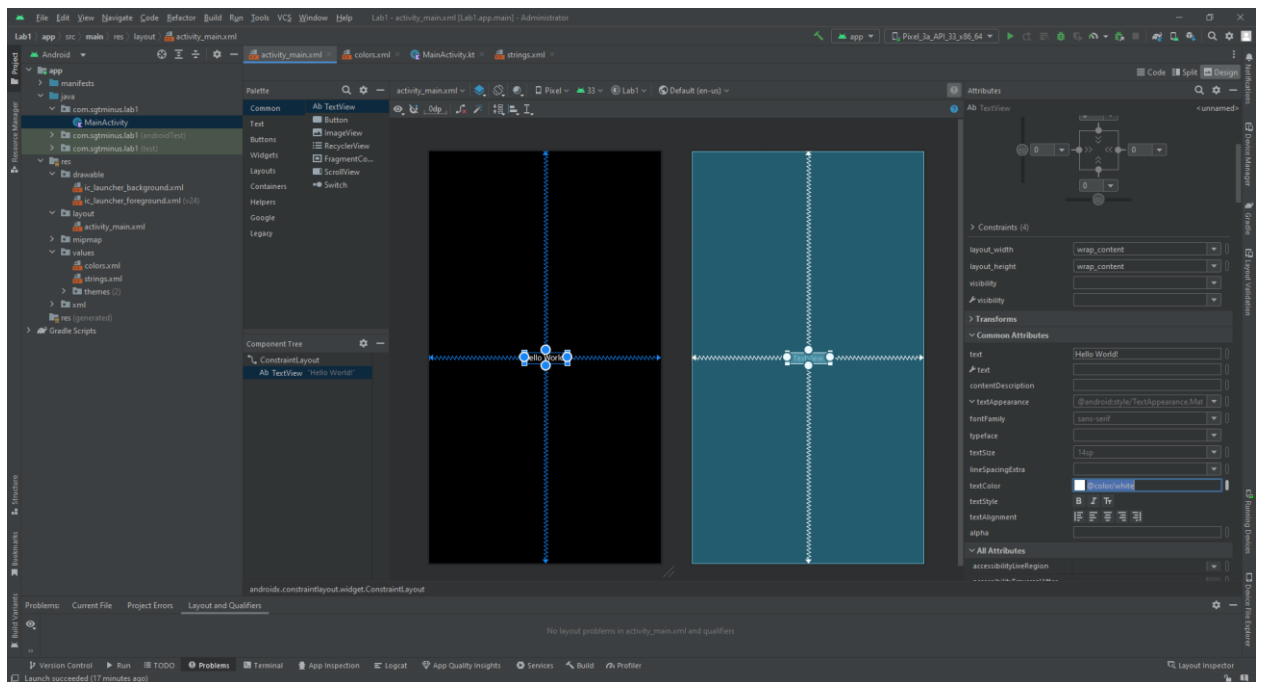


Рисунок 4

Я добавил кнопку, и Android Studio сразу же начал выдавать ошибку, связанную с тем, что для объекта необходимо указать отступ. Это можно сделать с помощью стрелочек. Таким образом, мы прижимаем объект к углу экрана, а после этого можем перемещать наш объект и за ним будет зафиксирован отступ по вертикали и горизонтали. Также стрелочки можно проводить к другим объектам и фиксировать положения нашего объекта относительно другого объекта.

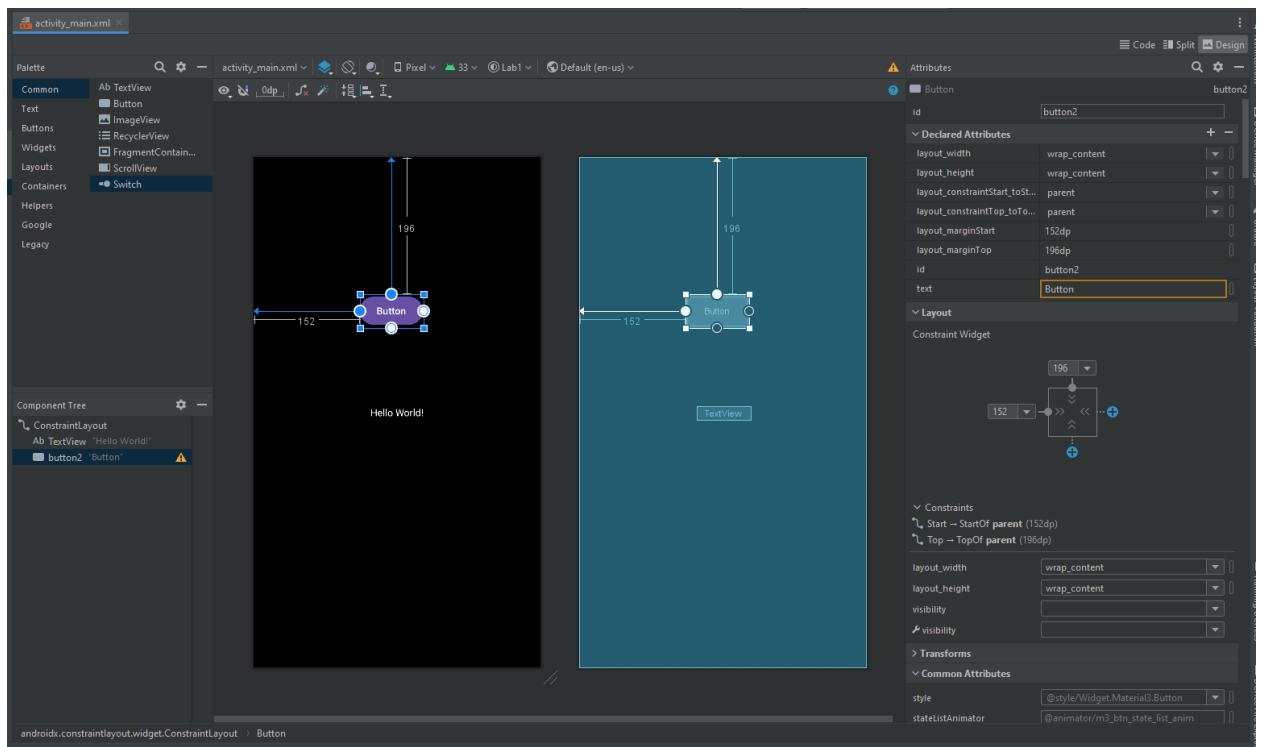


Рисунок 5

Также если я уменьшу данную кнопку, программа будет выдавать ошибку «Touch target size is too small».

Также программа ругается за то, что текст на кнопке ни на что не ссылается.

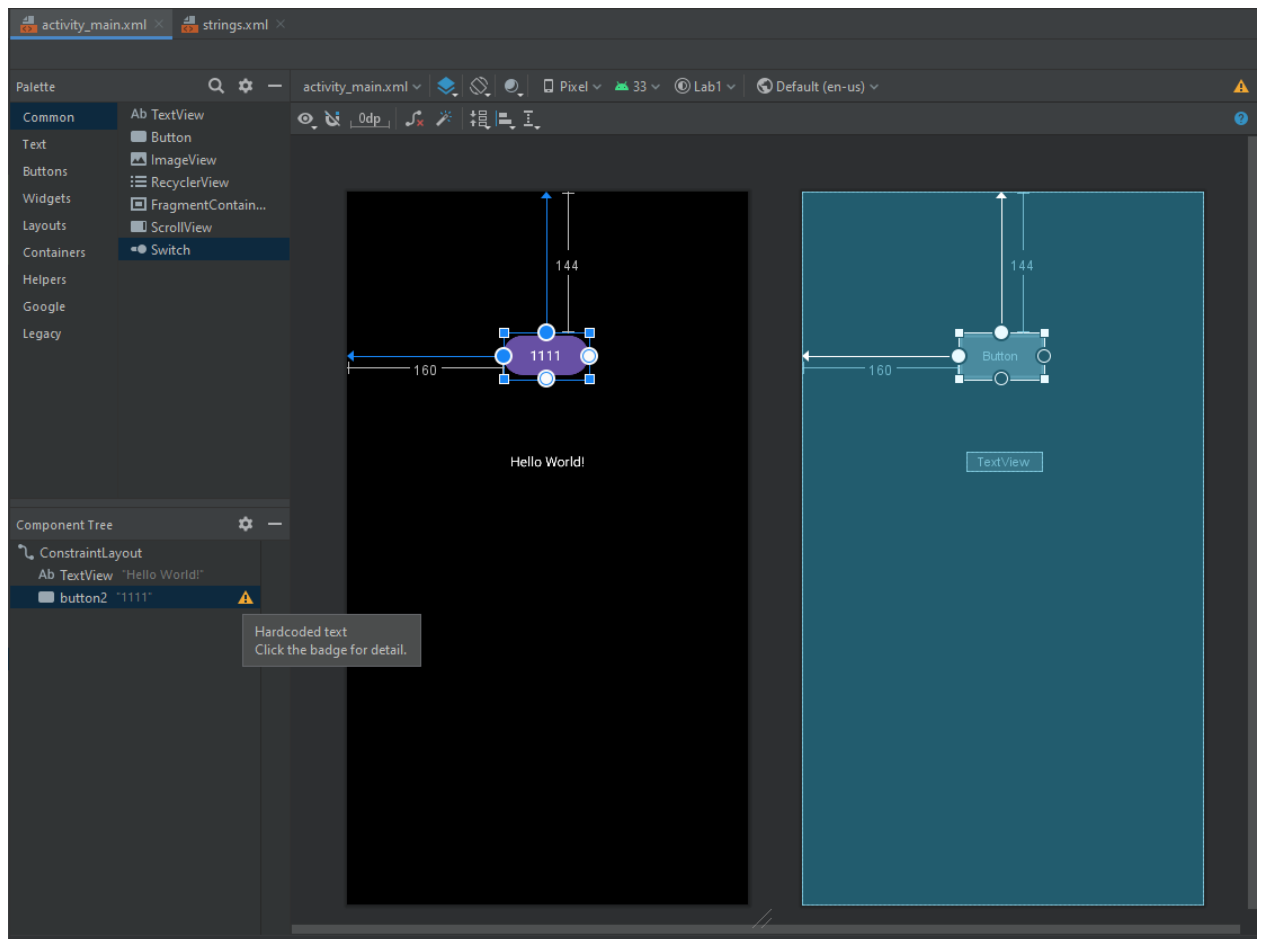


Рисунок 6

Так что я зашёл в strings.xml и добавил соответствующую строку, после этого я внёс в атрибут text моей кнопки своего рода ссылку на строку из strings.xml, итогом, надпись изменилась, а программа перестала ругать меня.

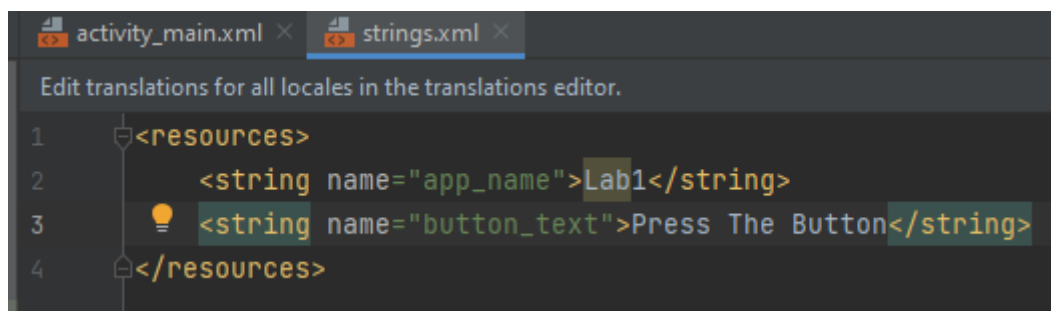


Рисунок 7

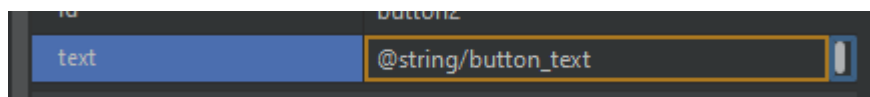


Рисунок 8

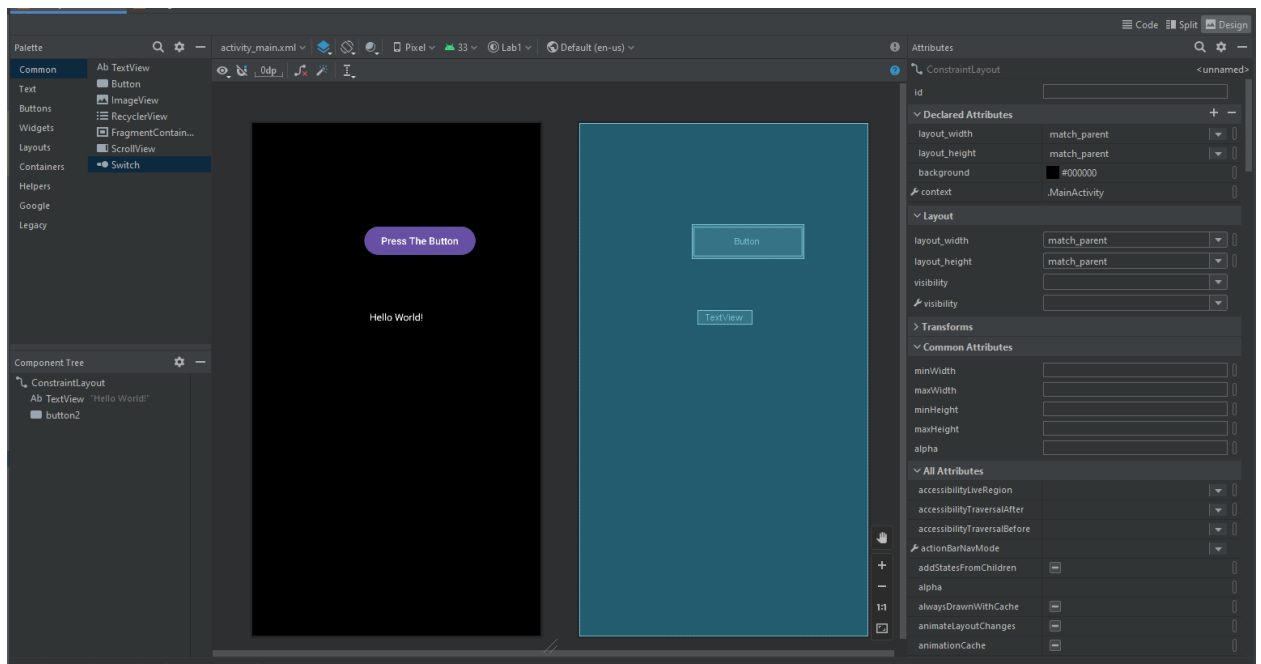


Рисунок 9

Теперь перейдем ко вкладкам

Во вкладке Project можно найти все файлы, все библиотеки, которые принадлежат нашему файлу.

В данный момент у меня есть папка под названием app, содержащая в себе три другие папки: manifest, java, res.

В manifest находится файл, описывающий глобальные характеристики всего проекта: название проекта, основная иконка, тема, activity (страницы, на данный момент всего одна – та самая MainActivity).

В java находятся 3 папки. Первая содержит классы, отвечающие за функционал, в текущий момент это MainActivity.kt. Сюда же мы будем добавлять новые файлы, если будем создавать новые страницы, новые фрагменты и т.д. Две оставшиеся папки содержат в себе классы для тестирования нашего проекта: androidTest и test, обе импортируют JUnit, стандартную библиотеку для проведения модульного тестирования.

В папке res содержатся ресурсы, которые есть внутри вашего проекта: изображения, текстовые файлы, видео/аудиофайлы, иконки и т.д. Содержит 4 папки: drawable, layout, mipmap, values.

- Drawable содержит файлы с расширением xml, которые мы можем редактировать с помощью визуального редактора, по факту тут находятся иконки.

- В layout находятся xml файлы, описывающие дизайн определенных activity, в нашем случае тут только тот самый файл activity_main.xml.
- В mipmap содержатся иконки, причем для устройств с различными плотностями пикселей. То есть, если у вас XXHDPI, то все остальные такие же файлы, но неподходящие под устройство, будут удалены.
- В values находятся основные характеристики, такие как основные цвета, основные надписи, основные темы приложения. Там же есть папка themes, содержащая два файла xml, в которых вы можете настроить стандартную и тёмную темы своего приложения

Помимо вкладки Project, слева также имеется Resource Manager, позволяющий находить ресурсы проекта.

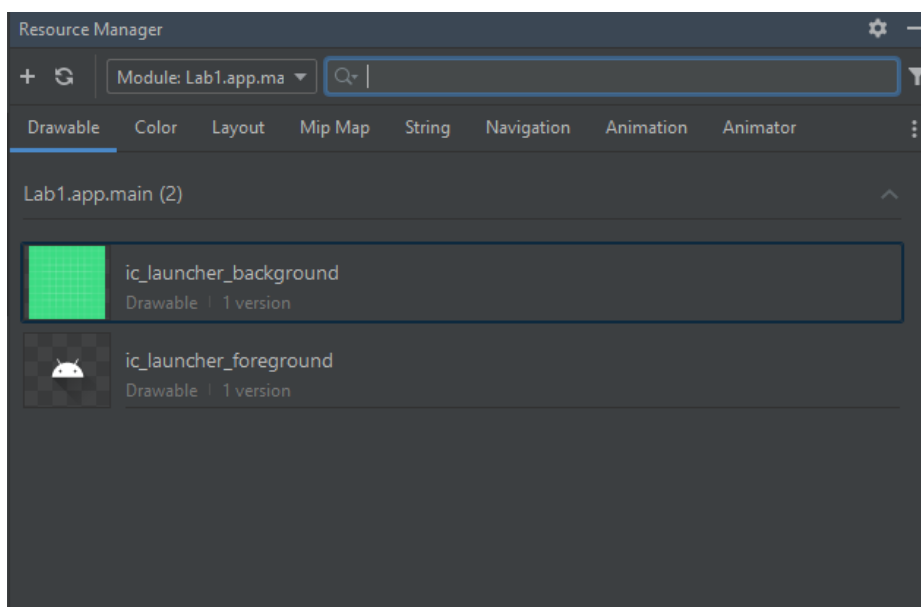


Рисунок 10

Structure описывает структуру всего проекта.

Есть вкладка Bookmarks, позволяющая быстро перейти к файлам, на которых вы оставили закладки.

Build Variants – различные варианты построения проекта, например, для тестирования.

Внизу большинство вкладок стандартные и интуитивно понятные, здесь же находится Profiler, о котором я говорил в начале отчёта.

Справа есть вкладка Running Devices – эмулятор. Также мы можем подключить реальный телефон для проверки работы приложения.

Вкладка Gradle – сборщик, позволяющий добавить различные новые библиотеки и расширения, отслеживать уже добавленные библиотеки.

Сверху у нас кнопки запуска приложения, выбора устройств, отладки и всё прочее, что присутствует в уже знакомой IntelliJ Idea.

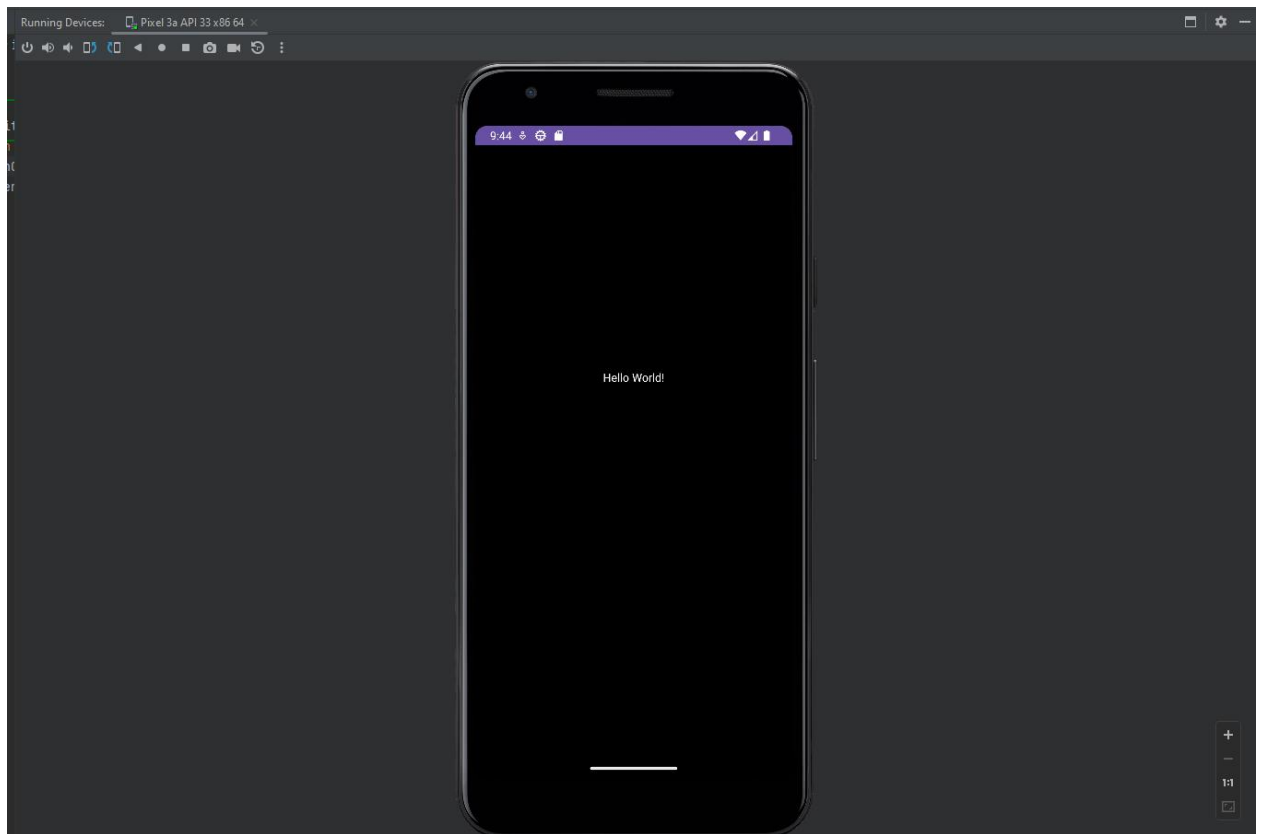


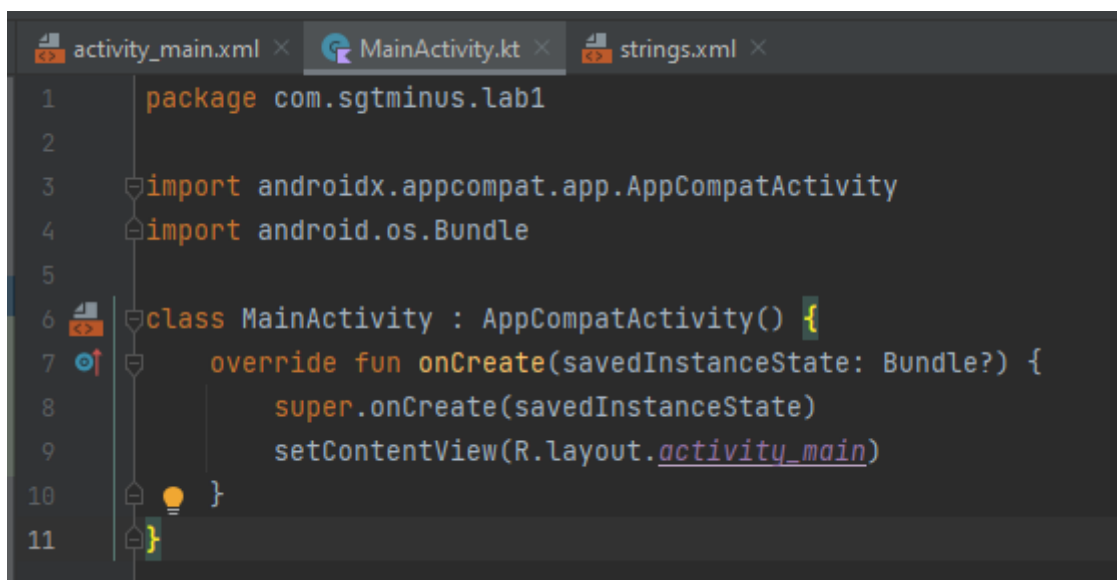
Рисунок 11

Глава 3. Код

Давайте перейдем к небольшой части, к коду.

Первым делом поговорим об Activity. Activity это просто страница в приложении, их может быть много и у каждой страницы будут свои определенные состояния.

Зайдём в наш MainActivity.kt

A screenshot of an IDE showing the MainActivity.kt file. The code defines a class MainActivity that inherits from AppCompatActivity. It overrides the onCreate method, which calls super.onCreate and setContentView. The package is com.sgtminus.lab1.

```
1 package com.sgtminus.lab1
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }
```

Рисунок 12

Здесь описан сам класс, наследующий AppCompatActivity. AppCompatActivity это базовый класс для страниц. Раньше актуальным классом был ActionBarActivity, но с развитием Android этот класс стал deprecated. Важно отметить, что класс AppCompatActivity обладает прямой и обратной совместимостями. AppCompatActivity позволяет подключиться к внутренней логике AppCompatActivity, включающей жизненный цикл, разного рода игру с цветами и всё прочее, что хотелось бы использовать в приложении.

Также описан сам метод onCreate. Помимо onCreate, очевидно, существуют и другие методы, которые вызываются в определенный момент жизни самой страницы. Здесь же имеет смысл показать и описать жизненный цикл страницы:

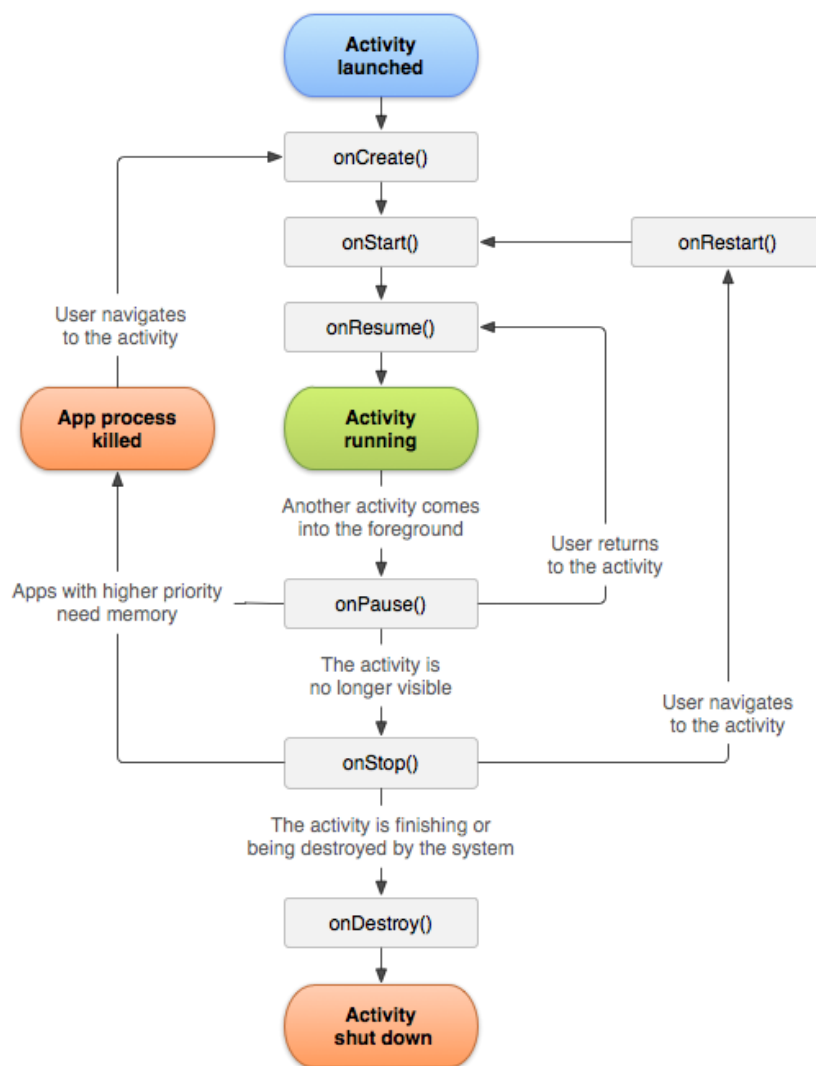


Рисунок 13

При создании страницы, как я уже сказал, запускается метод `onCreate()`. Сразу же после него запускается `onStart()`, всё ещё предшествующий показу страницы пользователю, но срабатывает после того, как полностью выполнится `onCreate()`. `onResume()` – уже процесс отображения страницы пользователю. Далее страница будет запущена, происходит процесс работы пользователя с приложением. Если пользователь, например, свернет приложение или же развернет Status Bar, то страница встанет на паузу и вызовется `onPause()`. Соответственно, `onPause()` будет срабатывать каждый раз при таких ситуациях. Когда пользователь вернется обратно, то снова сработает `onResume()`. Если окно становится невидимым для пользователя, то сработает `onStop()`. Если пользователь вернется обратно, то также сработает `onRestart()` и далее по схеме. Если же пользователь окончательно закрыл приложение, то после `onStop()` запустится `onDestroy()`.

Обратим внимание на класс Bundle в методе onCreate().

Когда страница создается заново после того, как она было ранее уничтожена, вы можете восстановить сохраненное состояние экземпляра из этого самого Bundle, который система передает вашему действию. Оба метода обратного вызова onCreate() и onRestoreInstanceState() получают один и тот же пакет, содержащий информацию о состоянии экземпляра. Поскольку метод onCreate() вызывается независимо от того, создает ли система новый экземпляр вашей страницы или воссоздает предыдущий, вам необходимо проверить, равен ли Bundle null, прежде чем пытаться его прочитать. Если оно равно null, то система создает новый экземпляр Activity вместо восстановления предыдущего, который был уничтожен.

Метод setContentView() размещает пользовательский интерфейс на экран. В параметре можно передать либо идентификатор ресурса, как у нас это сделано, либо объект View.

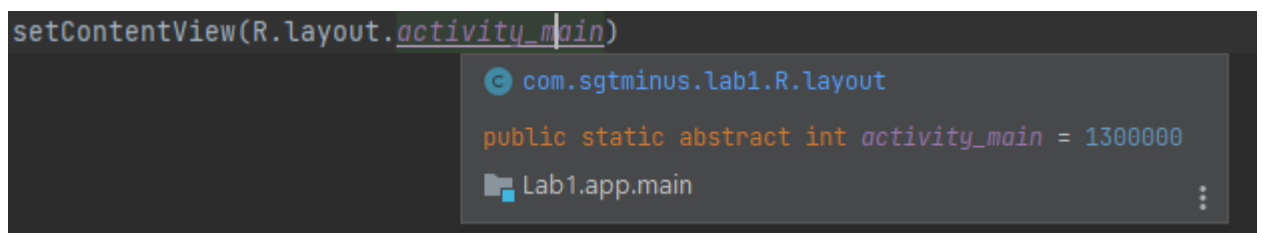


Рисунок 14



Рисунок 15

Что касается View, этот класс представляет собой базовый строительный блок для компонентов пользовательского интерфейса. Вид занимает прямоугольную область на экране и отвечает за рисование и обработку событий. View - это базовый класс для виджетов, которые используются для создания интерактивных компонентов пользовательского интерфейса

(кнопок, текстовых полей и т.д.). Подкласс ViewGroup является базовым классом для макетов, которые представляют собой невидимые контейнеры, содержащие другие View (или другие ViewGroup) и определяющие их свойства макета.

Глава 4. APK

Что ж, теперь я создам APK файл. Как это сделать, я покажу на скриншоте

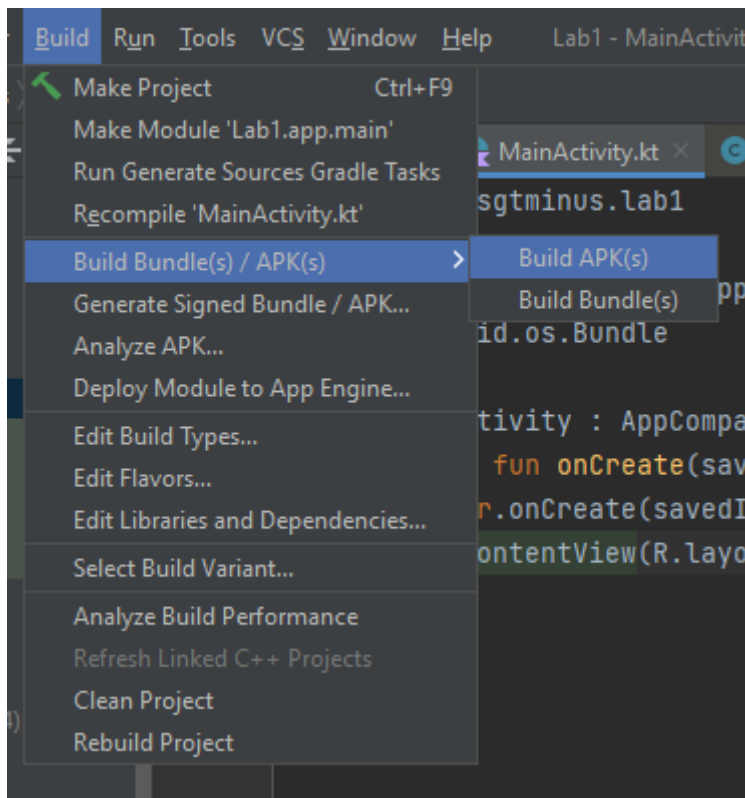


Рисунок 16

Вот нам предоставили оповещение об успешном создании арк файла, нажимаем locate и нас переносит в папку.

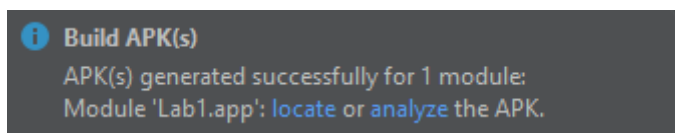


Рисунок 17

В моём случае меня перенесло в Lab1\app\build\outputs\apk\debug. Навязывается вопрос, почему именно в папку debug?

При разборе интерфейса я упоминал вкладку Build Variants, через которую мы можем строить разные варианты проекта, поэтому меня и перенесло в папку debug

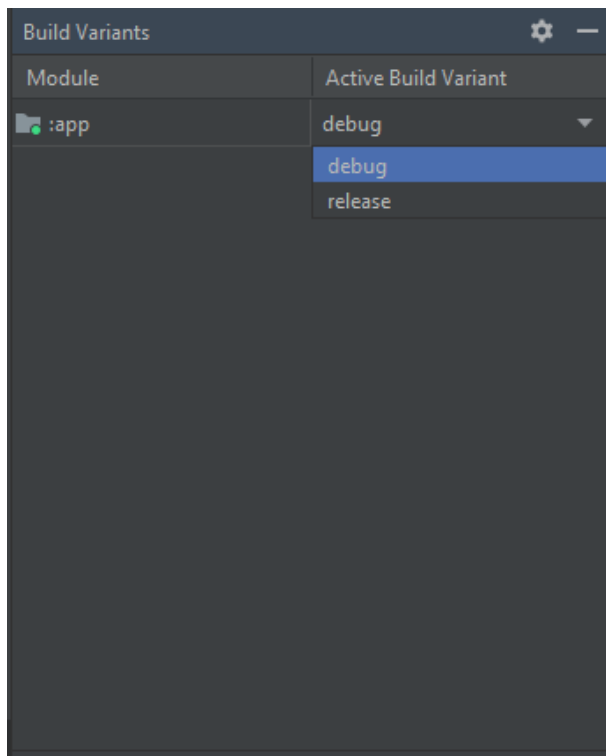


Рисунок 18

Меняем debug на release, заново жмем Build APK, всё отлично срабатывает.

Далее я задался вопросом: почему apk debug- версии весит больше?

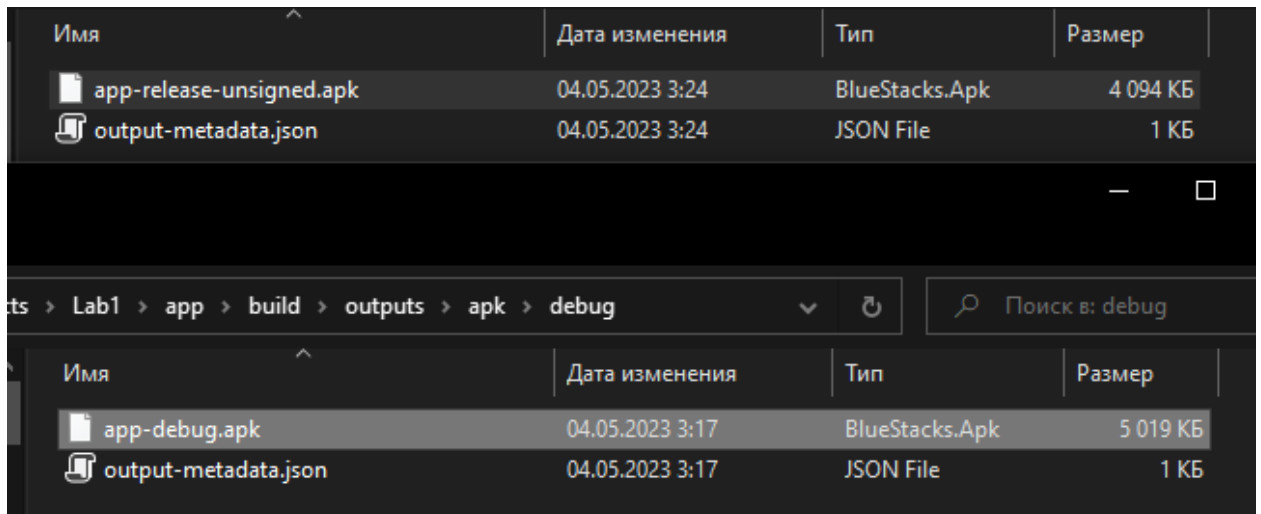


Рисунок 19

Во вкладке Build у нас есть кнопка Analyze APK, а также у нас есть кнопка для сравнения APK файлов, я ей воспользовался, результаты ниже на скриншоте:

app-debug.apk (old) vs app-release-unsigned.apk (new)			
File	Old Size	New Si...	Diff Size
▼ app-debug.apk	4,9 MB	4 MB	-925 KB
> assets/	0 B	1,8 KB	1,8 KB
kotlin-tooling-metadata.json	0 B	626 B	626 B
> res/	490,9 KB	490,9 KB	0 B
> kotlin/	28,3 KB	28,3 KB	0 B
> META-INF/	406 B	406 B	0 B
DebugProbesKt.bin	1,7 KB	1,7 KB	0 B
AndroidManifest.xml	3,6 KB	3,6 KB	-52 B
classes3.dex	1,1 KB	0 B	-1,1 KB
resources.arsc	802,9 KB	784,4 KB	-18,5 KB
classes2.dex	452,4 KB	0 B	-452,4 KB
classes.dex	8,7 MB	7,7 MB	-1 MB
<input type="checkbox"/> Show File-By-File patch size (may take a long time)			
OK		Cancel	

Рисунок 20

Я предполагаю, что удаляются классы, связанные с тестированием за отсутствием необходимости их наличия в релизной версии.

Поиски ответов на данный вопрос натолкнули меня на статью об оптимизации сборок Android приложений.

Итак, снизу показана схема сборки от компиляции исходного кода до установки на устройство:

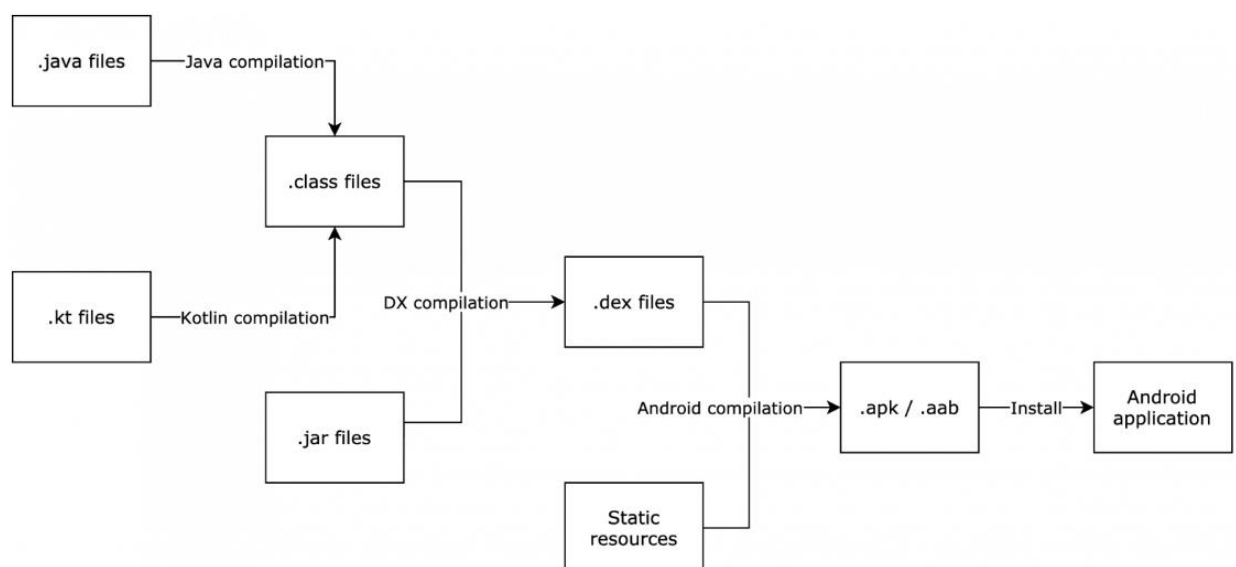


Рисунок 21

Оптимизация сборки в общем случае включает в себя несколько пунктов.

Первым пунктом является shrinking – «сокращение». Есть code shrinking, есть resource shrinking.

Code shrinking. Код можно разделить на достижимый и недостижимый. Строится дерево, корнем которого являются специальные входные точки, названные seeds. Они подаются на вход, который изначально является достижимым. Код каждого seed просматривается на предмет использования кода из других файлов. Весь используемый код помечается как достижимый и добавляется в очередь на проверку. Для всех достижимых файлов проводится то же самое, что и для seeds. В итоге недостижимый код удаляется. Сами seeds могут формироваться на основе правил библиотек, правил, сгенерированных Android Gradle Plugin, а также пользовательских правил.

Resource shrinking. Примерно такие же манипуляции происходят и с ресурсами. В итоге недостижимые ресурсы удаляются, достижимые – сохраняются.

Вторым пунктом является оптимизация. Оптимизация проводит манипуляции с байт-кодом с целью оптимизации с программной точки зрения.

Третьим пунктом является obfuscate – «запутывание». Обфускация переименовывает названия из исходного кода на неосмысленные. Исходные имена находятся в отдельном словаре обфускации, всем именам из словаря даются новые названия, на их основе меняется код приложения.

Четвертым пунктом является preverify – «предварительная проверка». Здесь осуществляется проверка кода на соответствие требованиям среды эксплуатации.

Таким образом, засчет этих этапов мы оптимизируем расход памяти, скорость работы и уровень безопасности.

Существует несколько оптимизаторов сборки приложений. Google предоставляет собственную технологию – R8. Помимо этого, существует

также и ProGuard.

Optimization	ProGuard	R8
Remove unused code	+	+
Outline common code into new methods	-	+
Propagate constant arguments	+	-
...

Рисунок 22

Глава 5. GitHub

Теперь я загружу свой проект на GitHub, основанный на системе контроля версий Git.

Для начала в Settings/Version Control нужно авторизоваться с помощью токена.

Далее после нажатия на Share Project on GitHub в VCS, нужно заполнить форму и выбрать файлы, которые нужно загрузить. В конце появится оповещение об успешной загрузке на GitHub.

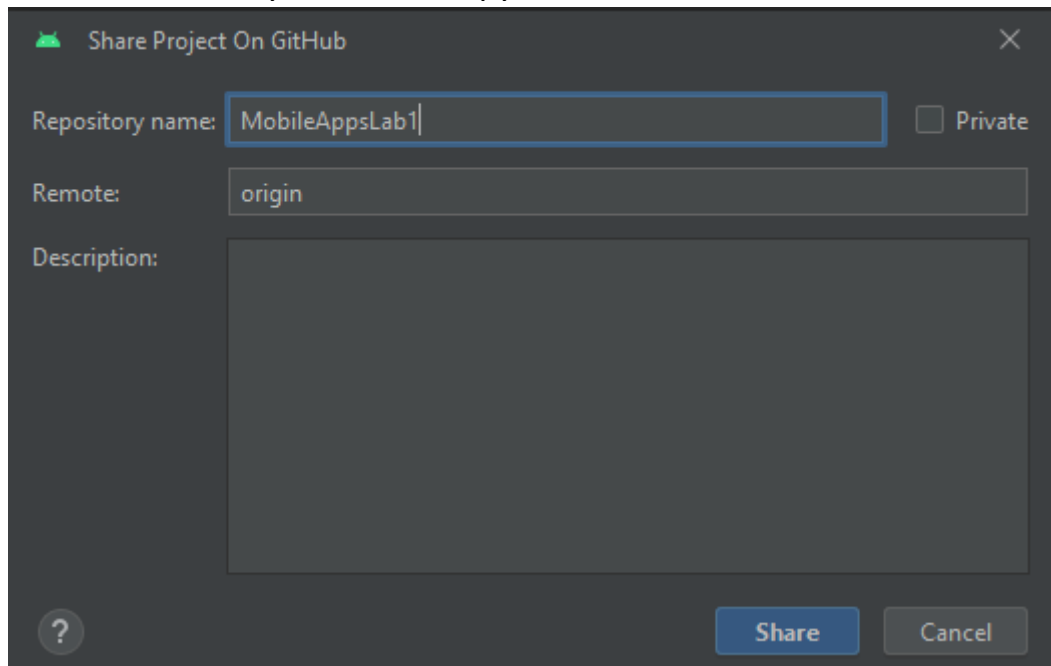


Рисунок 23

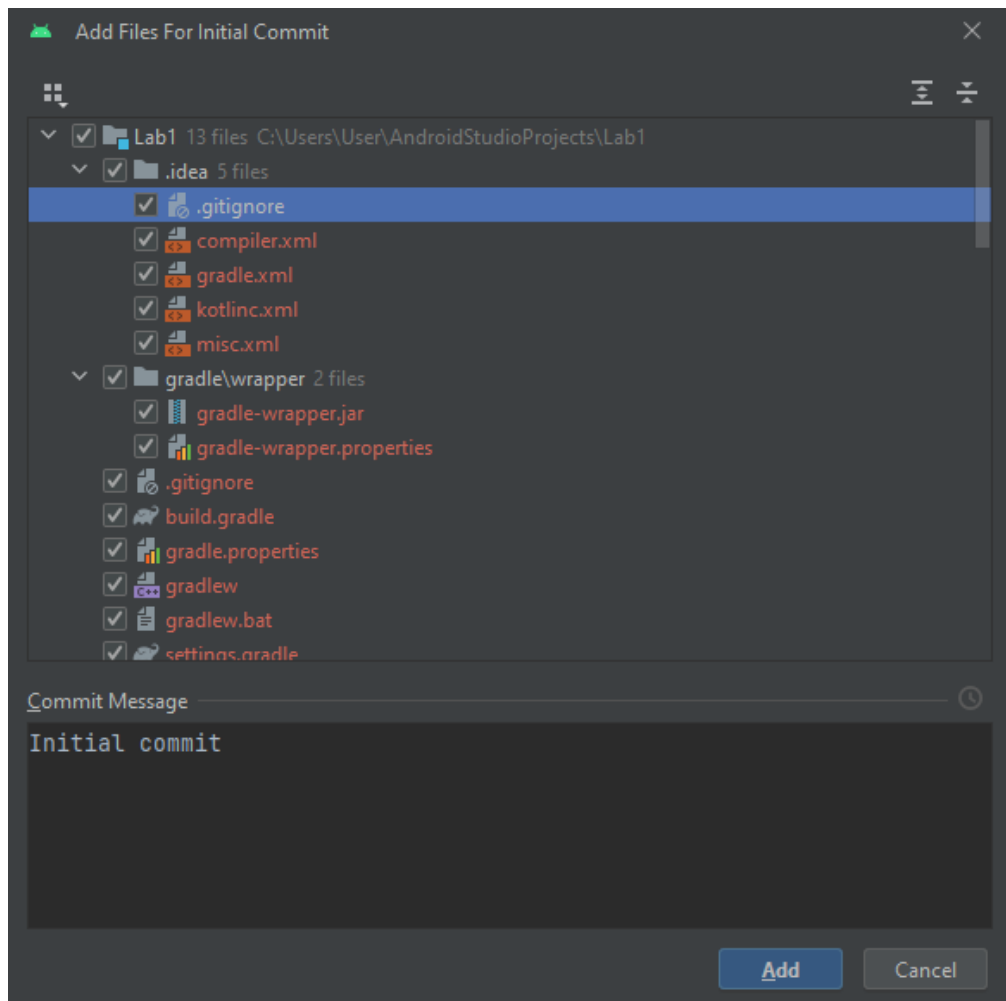


Рисунок 24

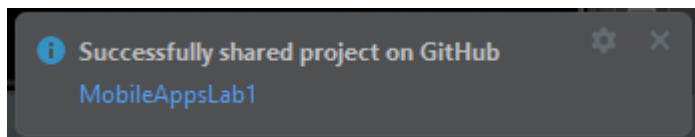


Рисунок 25

<https://github.com/SgtMinus/MobileAppsLab1>

Вывод

В рамках выполнения данной лабораторной работы я познакомился с Android Studio, с языком программирования Kotlin, а также некоторыми деталями их реализации, после постижения которых я с уверенностью могу сказать: «Hello World»

Список источников:

1. <https://habr.com/ru/articles/533578/>
2. <https://developer.android.com>
3. <https://stackoverflow.com/questions/60529016/why-is-the-release-apk-smaller-than-th-debug-apk>