



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе №1

по дисциплине «Проектирование и разработка мобильных приложений»

Выполнил:

Студент группы ИКБО-20-23

Кузнецов Лев Андреевич

Проверил:

старший преподаватель кафедры
МОСИТ

Шешуков Л. С.

МОСКВА 2025 г.

СОДЕРЖАНИЕ

1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ.....	3
1.1 Установка Android Studio	3
1.2 Создание проекта	5
1.3 Структура проекта.....	9
1.4 Запуск проекта.....	10
1.5 Создание графического интерфейса	17
1.6 Вёрстка в Android. Язык XML	19
1.7 Ресурсы в Android	22
1.8 Создание интерфейса пользователя при помощи ресурсов.....	25
1.9 Компоненты разметки	26
1.9.1 TextView	27
1.9.2 EditText.....	27
1.9.3 Button	28
1.9.4 ImageView	28
2 Практическая часть	30
2.1 Решение задачи.....	30
2.1.1 TextView в задаче	31
2.1.2 EditText в задаче	32
2.1.3 Button в задаче	32
2.1.4 ImageView в задаче	33
ЗАКЛЮЧЕНИЕ	34

1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ

1.1 Установка Andriod Studio

Множество сред разработки поддерживают создание приложений для Android, но наиболее предпочтительной считается Android Studio. Она разработана специально для Android, предлагает интегрированные инструменты и оптимизированный рабочий процесс, что делает ее наиболее предпочтительным выбором для разработчиков. Для начала работы с Android Studio скачаем установщик с официального сайта (Рисунок 1).

Android Studio downloads

Download the latest version of Android Studio. For more information, see the [Android Studio release notes](#).

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-2024.2.2.14-windows.exe Recommended	1.2 GB	0130f969c432643d3d94f0f0d3a3218b18bf7dc970e0db7cd31d2ee2033fd49a
Windows (64-bit)	android-studio-2024.2.2.14-windows.zip No .exe installer	1.2 GB	9a3649890c71abe8f3c67c18c42742c32b450ed439c1fbfabe5d55c1876cf33
Mac (64-bit)	android-studio-2024.2.2.14-mac.dmg	1.3 GB	de246ff95d635a4df642e3f7e64457d384c72f1916a2a318a630d8a43219b9c3
Mac (64-bit, ARM)	android-studio-2024.2.2.14-mac_arm.dmg	1.3 GB	e58381031e32e9d2792a9b62561837b00d23fd48f662cc3cd242821ae6844d45
Linux (64-bit)	android-studio-2024.2.2.14-linux.tar.gz	1.3 GB	73db7c190c366b36087f8bf288a4a89676c64d47aad9ad8c528010de8bb2344
ChromeOS	android-studio-2024.2.2.14-cros.deb	1.0 GB	1e0c096f3a1d694fe69c0111edfcc1a796673dfc5741d307af72251e19fee27f

Рисунок 1 - Официальный сайт Andriod Studio

Среда Android Studio требует для своей работы много системных ресурсов. Как следствие, для комфортной работы программиста рекомендуется достаточно мощный компьютер (так, например, рекомендуемый объем оперативной памяти - 8 ГБ). Для разработки на Android в дополнение к Android Studio нам потребуется также установить Android Software Development Kit (SDK). Этот набор инструментов является обязательным для создания приложений, поскольку включает в себя библиотеки API, эмуляторы устройств, инструменты

для отладки и многое другое. А также AVD, который является эмулятором виртуального устройства для Android Studio.

При установке Android Studio будет предложено установить базовый пакет и в него будут включены все необходимые инструменты (Рисунки 2-4).

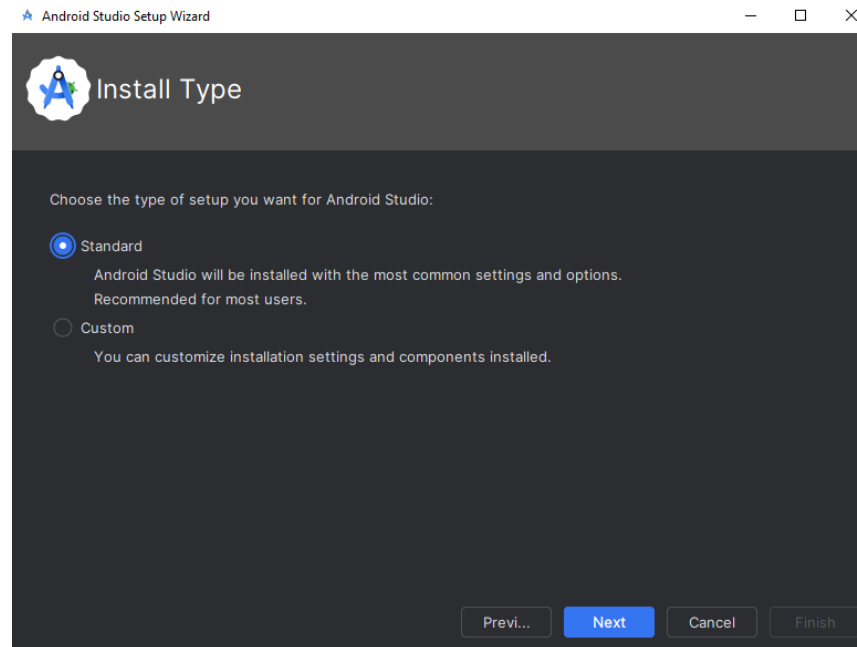


Рисунок 2 - Процесс установки Android Studio

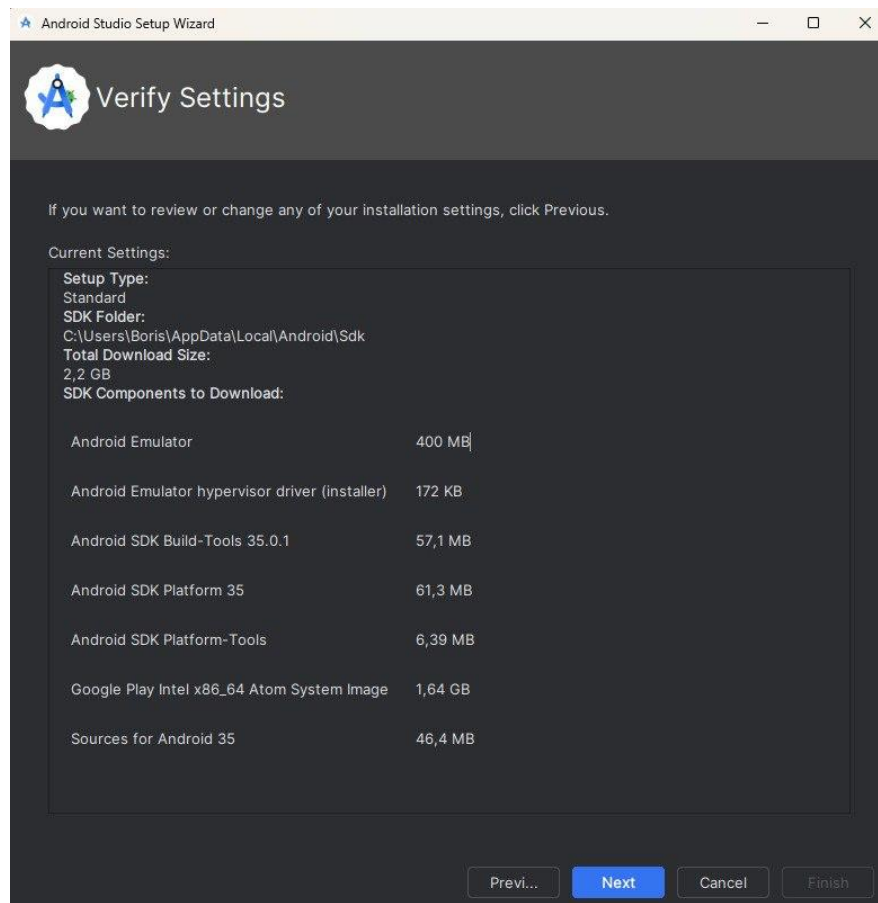


Рисунок 3 - Часть 1 установки инструментов для Android Studio

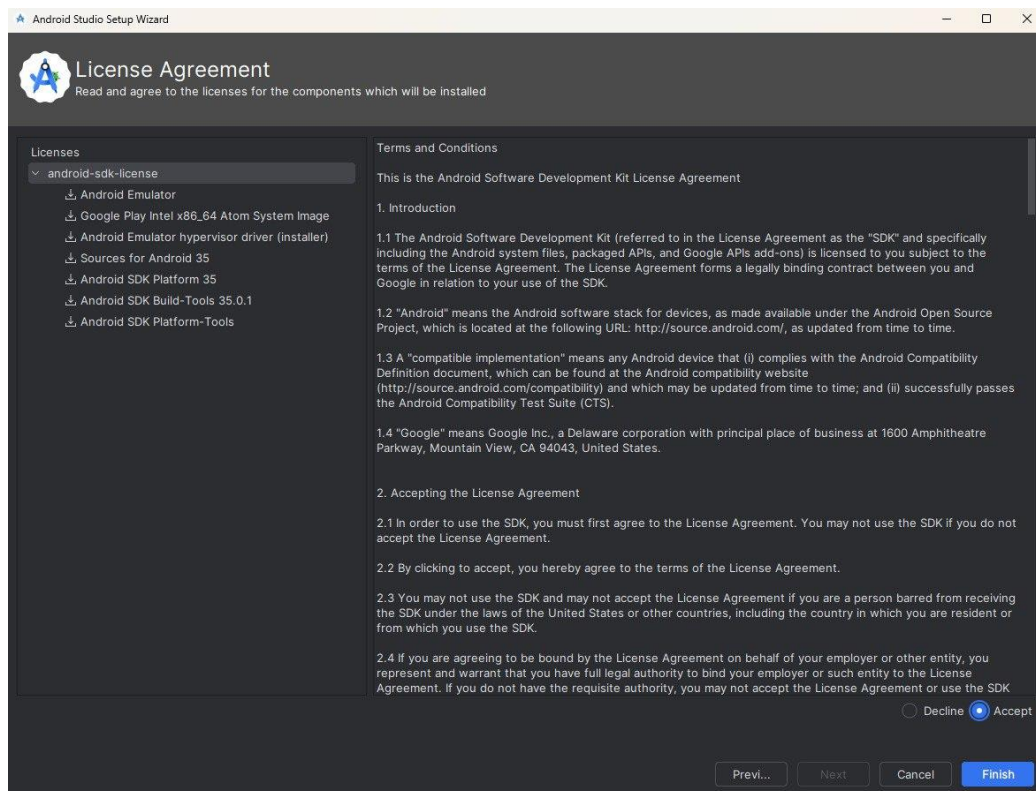


Рисунок 4 - Часть 2 установки инструментов для Android Studio

1.2 Создание проекта

После установки и запуска Android Studio создаём новый проект. Для этого на начальном экране выбираем кнопку "new project" (Рисунок 5).

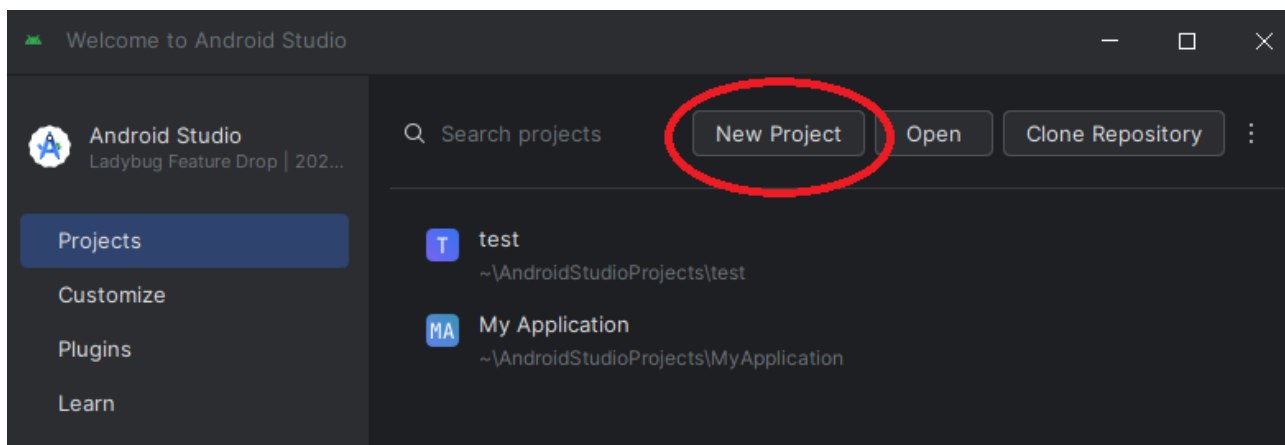


Рисунок 5 - Создание нового проекта

Android Studio предлагает различные шаблоны для нашего нового проекта. Для простого приложения выбираем тип шаблона "Phone and Tablet" и шаблон "Empty View Activity", который представляет собой пустой шаблон проекта с одним экраном (Рисунок 6).

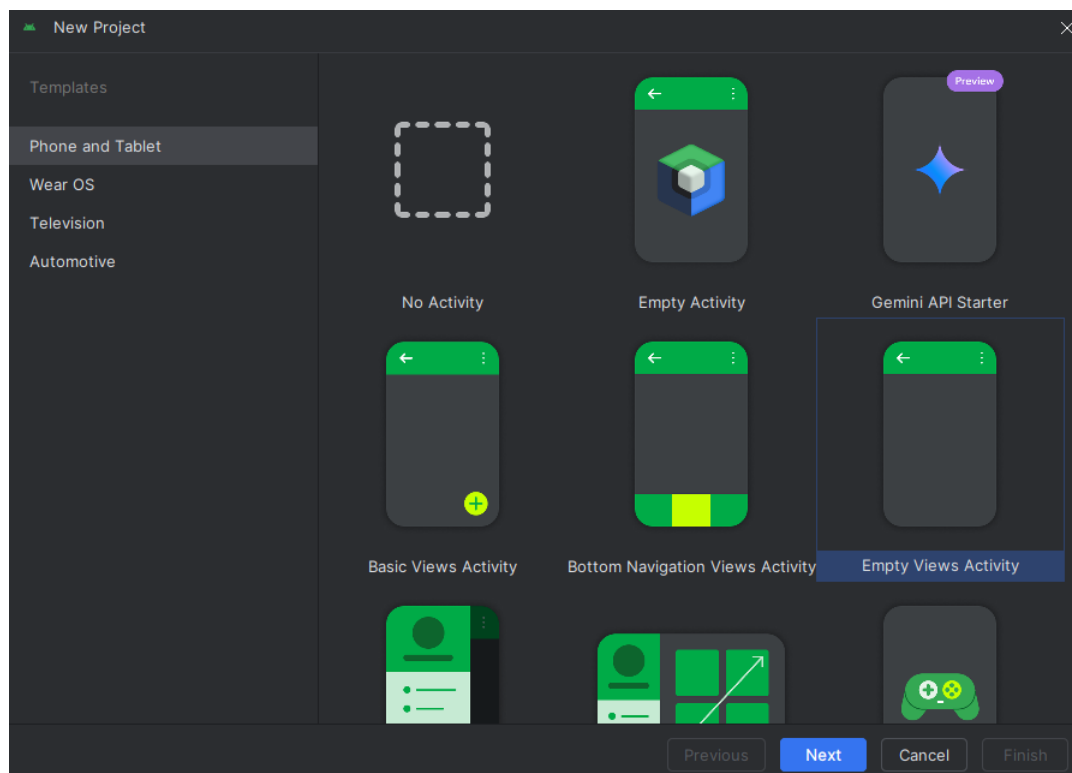


Рисунок 6 - Выбор шаблона проекта

На данной вкладке всего предлагается 10 шаблонов:

- No Activity – это шаблон в Android Studio, который не создает активностей (Activities) и не добавляет никаких компонентов пользовательского интерфейса по умолчанию. Предназначен для разработки библиотек фоновых сервисов, а также серверных приложений, где UI не нужен;
- Empty Activity – это базовый шаблон для создания приложения с одной пустой активностью. Он включает только минимальный код для активности, которая отображает простое представление. Подходит для проектов, где вы хотите начать с нуля и настроить все компоненты и логику вручную;
- Gemini API Starter – шаблон предназначенный для разработки приложений, использующих Gemini API (например, для работы с облачными сервисами, такими как Google Cloud). Этот шаблон может включать базовую настройку для интеграции с API и авторизации. Полезен для начинающих работать с облачными сервисами и API, такими как Google Cloud или другие облачные платформы;
- Basic Views Activity – это шаблон, создающий проект с основной активностью, которая использует простые элементы пользовательского

интерфейса (кнопки, текстовые поля, изображения и т.д.). Хорошо подходит для приложений, которые требуют базовой верстки с использованием стандартных элементов UI. Это идеальный старт для простых приложений, не требующих сложной навигации;

- Bottom Navigation Views Activity – этот шаблон создает приложение с нижней навигацией (Bottom Navigation), которая позволяет пользователю переключаться между несколькими фрагментами с помощью значков внизу экрана. Применяется для приложений с несколькими разделами, где удобнее использовать навигацию внизу экрана (например, приложения с вкладками);
- Empty Views Activity – шаблон создает пустую активность с поддержкой отображения представлений, но без предустановленных компонентов или логики. Это шаблон для приложений, которые будут работать с кастомными представлениями. Подходит для случаев, когда необходимо отображать кастомные компоненты или представления, например, при работе с динамическими UI;
- Navigation Drawer Views Activity – этот шаблон создает приложение с навигационным ящиком (Navigation Drawer) – популярной UI-методикой, которая позволяет пользователю переходить по разделам приложения через боковое меню. Идеален для приложений с большим количеством разделов или настроек, где необходимо организовать простую и удобную навигацию через боковое меню;
- Response Views Activity – этот шаблон создаёт активность, предназначенную для отображения данных, полученных от сервера или API, в виде ответа, например, в виде списка, таблицы или графика. Подходит для приложений, которые получают и обрабатывают данные, например, новостные агрегаторы, социальные сети или приложения для работы с внешними сервисами;
- Game Activity (C++) – шаблон, который используется для разработки игр, используя C++. Он включает в себя базовую активность с настройками для работы с игровыми движками, такими как Unreal Engine или другие движки, которые поддерживают C++. Подходит для создания игр, которые требуют

высокой производительности или специфичных возможностей, которые предоставляет C++;

- Native C++ - этот шаблон позволяет разработать приложение на чистом C++ с использованием Android NDK (Native Development Kit). Такой проект предоставляет возможность использовать нативный код, чтобы повысить производительность. Идеален для приложений, которые требуют нативной производительности, например, для работы с графикой, звуком, или при разработке игр;

Далее необходимо заполнить настройки проекта (Рисунок 7).

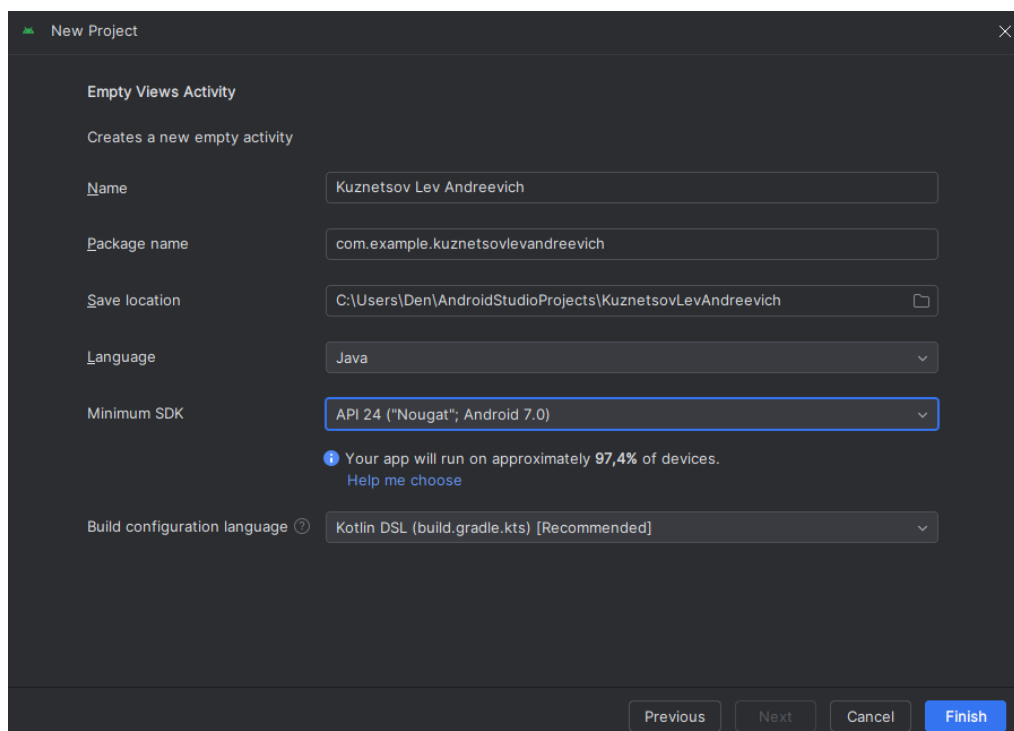


Рисунок 7 - Заполнение настроек проекта

Опишем, что означает каждый из пунктов Рисунка 5:

- name: отвечает за название проекта;
- package name: название пакета. Уникальный идентификатор нашего приложения;
- save location: место на компьютере, где будет сохранён проект;
- language: язык программирования, на котором будет написан проект;
- minimum SDK: минимальная версия Android, которую будет поддерживать наше приложение;

- build configuration language: язык, который будет применяться для определения конфигурации построения проекта.

После настройки нажимаем на кнопку “Finish” и ждем загрузки проекта.

1.3 Структура проекта

Теперь рассмотрим структуру проекта (Рисунок 6).

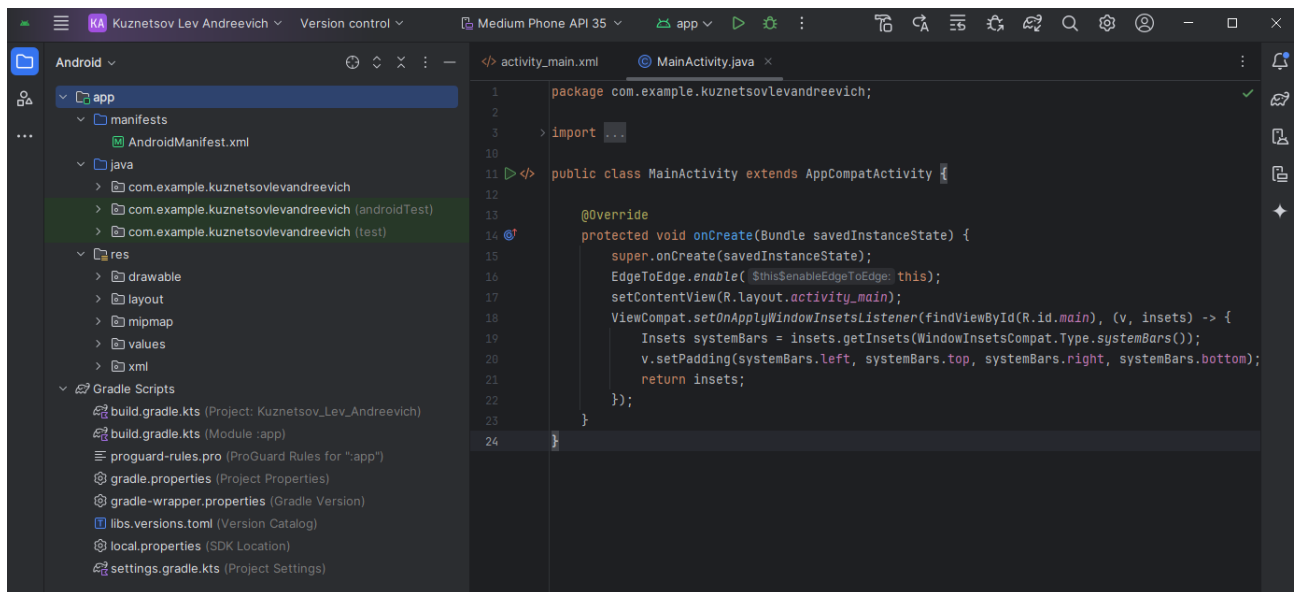


Рисунок 8 - Структура проекта на Android Studio

Модуль "app" является основным компонентом проекта Android и содержит файлы, необходимые для сборки проекта. В нём хранятся следующие элементы:

- manifests: содержит файл AndroidManifest.xml где описаны основные характеристики приложения — компоненты (активности, службы и т.д.), требуемые разрешения (на отправку уведомлений, интернет соединение и др.) и настройки приложения (название, тема, иконка и т.д.).
- java: содержит три пакета, отвечающих за код приложения, код для инструментальных тестов, выполняющихся на Android устройствах и код для модульных тестов, выполняющихся на вашем компьютере.

- `res`: содержит все ресурсы, не связанные с кодом, такие как XML-макеты, UI элементы (кнопки, текстовые поля), строки, изображения (drawables), стили и темы.

Модуль "Gradle Scripts" управляет процессом сборки вашего проекта. В нём хранятся следующие элементы:

- `build.gradle.kts` (Project: «Название вашего проекта»): Этот файл на уровне проекта содержит конфигурацию, применяемую ко всем модулям в проекте, а также включает ссылки на плагины Gradle, используемые проектом.
- `build.gradle.kts` (Module :app): Этот файл на уровне модуля содержит конфигурацию сборки, специфичную для данного модуля. Здесь определяются настройки конкретного модуля, такие как версия SDK, версии зависимостей, конфигурации сборки (например, отладка и выпуск) и другие настройки Android, специфичные для модуля.
- `settings.gradle.kts`: Содержит ссылки на модули, включенные в ваш проект. Каждый модуль, который должен быть собран как часть вашего проекта, должен быть заявлен здесь.

1.4 Запуск проекта

Созданный проект можно запустить как на реальном, так и на виртуальном устройстве. Рассмотрим оба варианта.

Для запуска проекта на физическом устройстве необходимо активировать режим разработчика и разрешить отладку по USB.

Чтобы это сделать нам нужно перейти в «Настройки», далее выбрать пункт «Мой телефон» и 7 раз нажать на пункт «Номер сборки», после чего появится всплывающее уведомление о том, что режим разработчика включен (Рисунок 9).

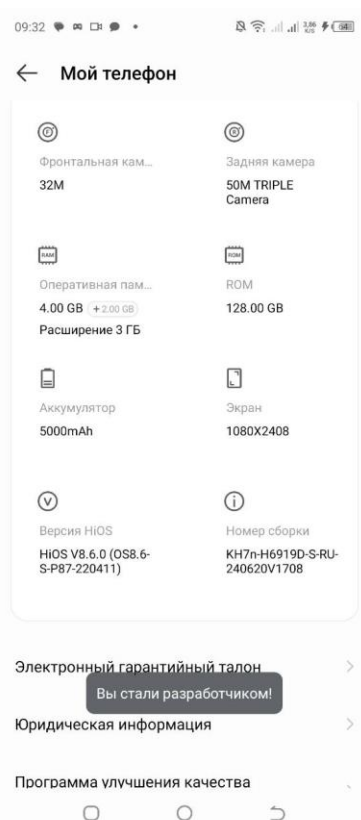


Рисунок 9 - Переход в режим разработчика на мобильном устройстве

Теперь, для включения отладки по USB вернёмся в «Настройки» и в разделе «Система» выберем пункт «Для разработчиков», где необходимо включить пункт «Отладка по USB» (Рисунок 10).

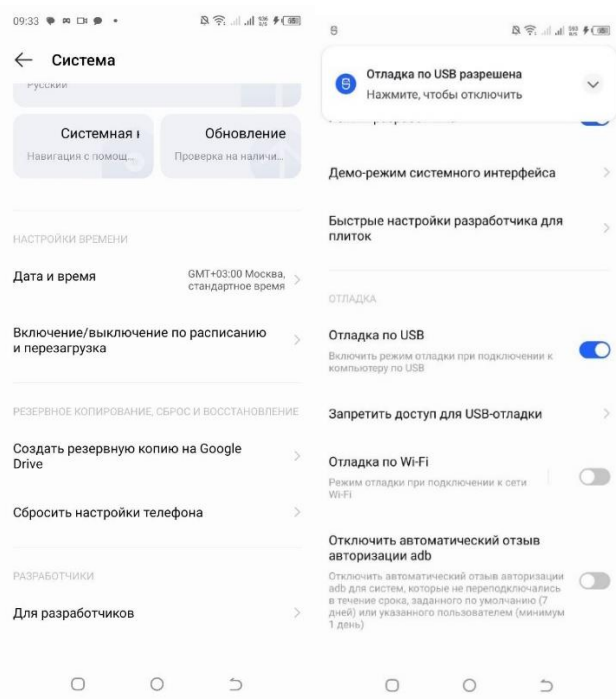


Рисунок 10 - Разрешение отладки по USB через настройки

Далее подключаем устройство к компьютеру с открытым Android Studio. Программа должна автоматически обнаружить устройство и отобразить его в разделе "Available devices" (Рисунок 11). Запускаем проект, нажав зеленую кнопку справа.

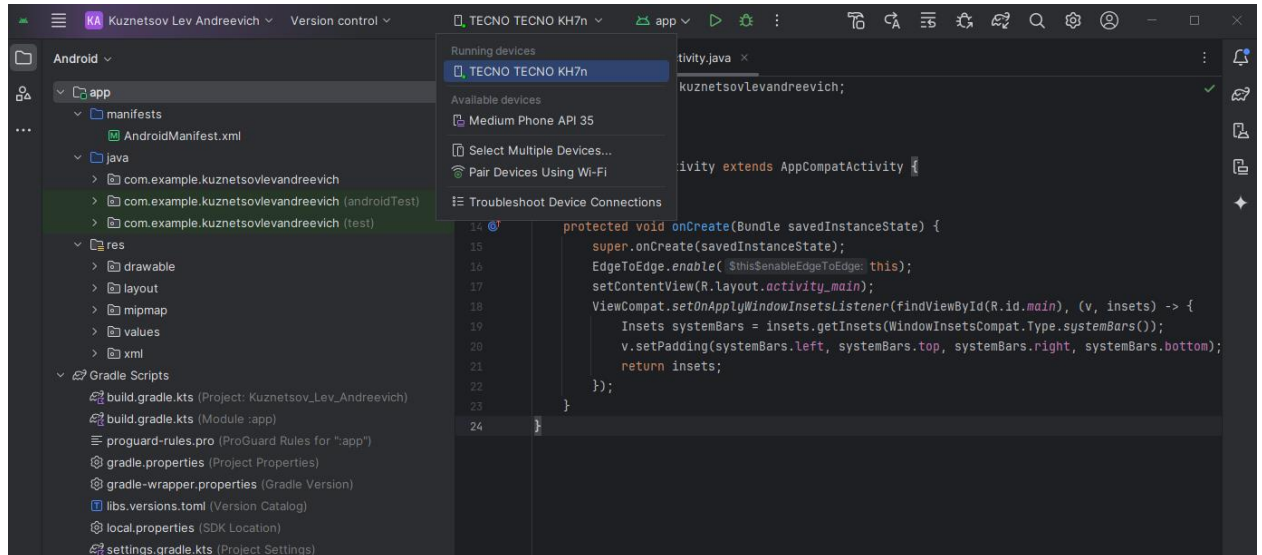


Рисунок 11 - Запуск приложения на мобильном устройстве через Android Studio

После сборки приложения мы сможем увидеть запущенное приложение на экране нашего телефона (Рисунок 12).

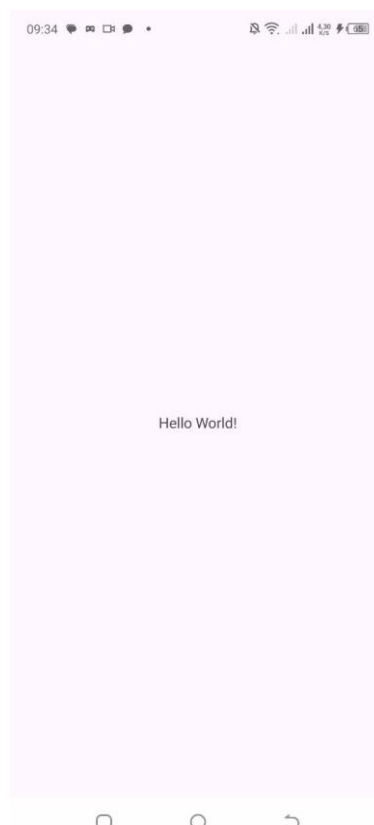


Рисунок 12 - Приложение, запущенное на мобильном устройстве

Для запуска на виртуальном устройстве необходимо создать его в Android Studio. Чтобы это сделать перейдём в пункт "Tools" и выберем раздел "Device Manager". Затем, в открывшемся боковом окне нажмём «+» для создания нового устройства (Рисунок 13).

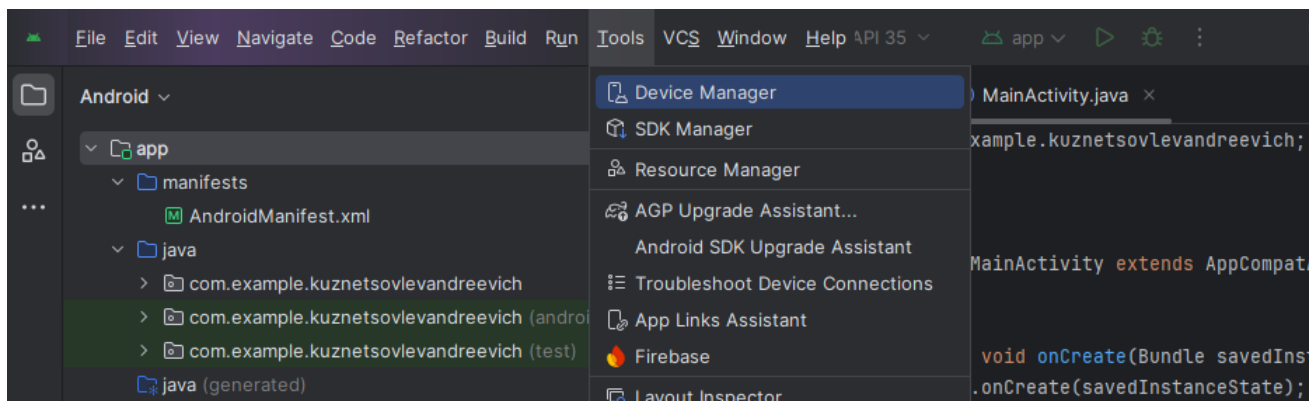


Рисунок 13 - Создание виртуального устройства

Теперь выберем параметры виртуального устройства. Сначала настроим тип устройства и его размеры (Рисунок 14).

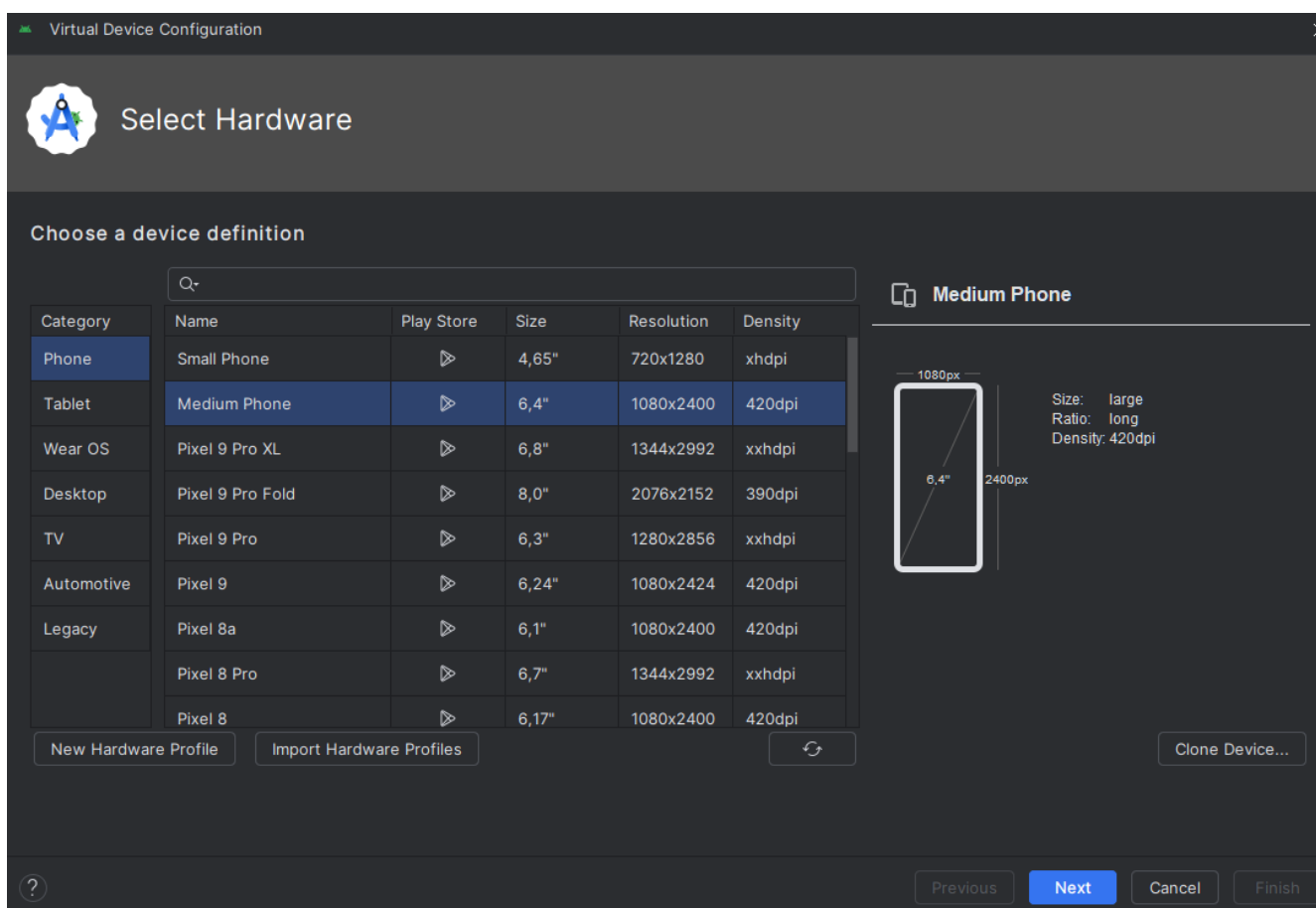


Рисунок 14 - Выбор параметров виртуального устройства

На выбор предоставляется всего 7 различных категорий определения устройства:

- Phone - это наиболее распространённая категория устройств, которая включает в себя все стандартные смартфоны с операционной системой Android. Подходит для большинства приложений, ориентированных на мобильных пользователей;
- Tablet - это устройства, которые являются большими по размеру и предназначены для планшетов. Применяется для тестирования приложений, предназначенных для планшетов с большими экранами, где важен интерфейс и мультизадачность;
- Wear OS - это устройства с операционной системой Android, предназначенные для носимых устройств, таких как смарт-часы. Подходит для тестирования приложений, созданных для носимой электроники, таких как фитнес-трекеры или умные часы;
- Desktop - это устройства, которые имитируют работу Android на ПК или настольных устройствах. Используется для тестирования приложений в среде, где Android работает на десктопах или ноутбуках;
- TV - Это устройства с операционной системой Android TV, предназначенные для использования на телевизорах. Применяется для разработки и тестирования приложений, которые предназначены для телевизоров и имеют интерфейс, адаптированный для больших экранов и управления с пульта;
- Automotive - это устройства, использующие Android для автомобильных информационно-развлекательных систем. Используется для создания и тестирования приложений, которые должны интегрироваться с автомобильными мультимедийными системами и навигацией;
- Legacy - это устройства с устаревшими версиями Android или устаревшими характеристиками, которые больше не поддерживаются в современных приложениях. Этот тип устройства используется для тестирования совместимости с очень старыми версиями Android (например, Android 4.4 KitKat и ниже);

В данном варианте мы выбрали определение устройства как Medium Phone, который предопределяет размер, разрешение и плотность пикселей.

Далее выбираем версию ОС Android, которая будет установлена на устройство, лучше выбирать самую последнюю. Если бы мы создавали устройство впервые, то перед выбором ОС ее нужно бы скачать, для этого надо будет нажать на кнопку загрузки, рядом с версией ОС (Рисунок 15).

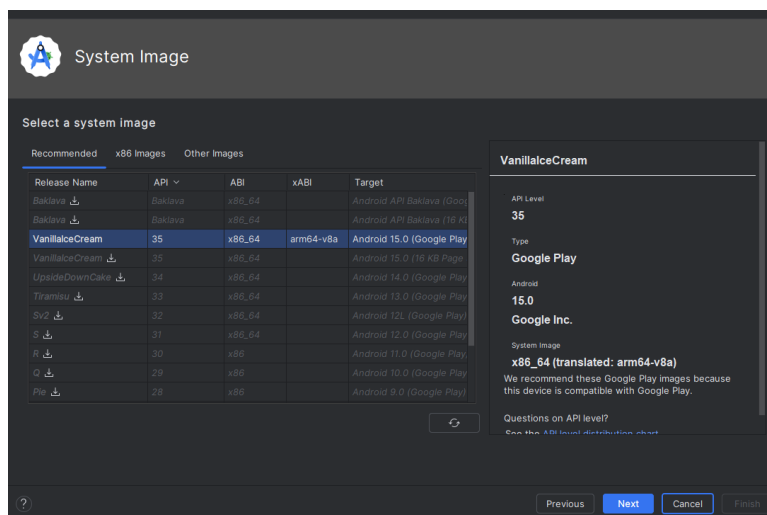


Рисунок 15 - Выбор операционной системы виртуального устройства

Все ОС, перечисленные на Рисунке 15, относятся к смартфонам и ориентированы на использование специфических интерфейсов и взаимодействий (использование стилуса, комбинации кнопок на смартфоне и т.д.).

Выбрав устройство и версию Android, мы сможем посмотреть выбранные характеристики, выбрать тип ориентации и создать устройство (Рисунок 16).

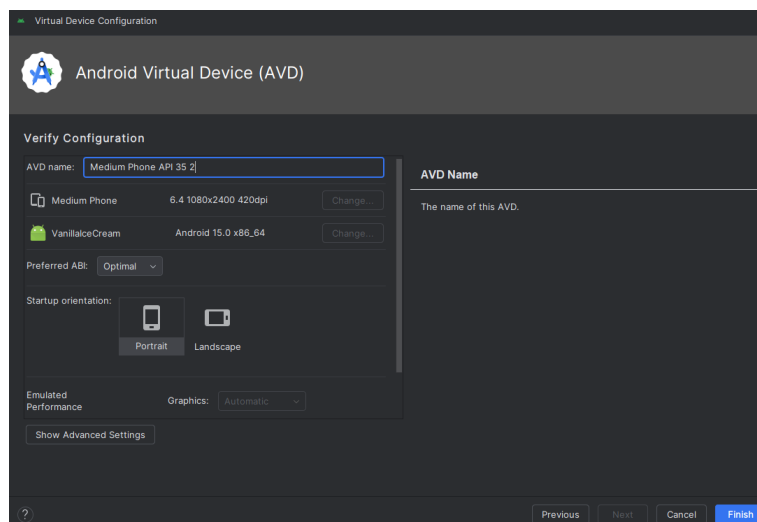


Рисунок 16 - Завершение создания виртуального устройства

Когда устройство создано, мы можем выбрать его в качестве основного в разделе "Available devices" и запустить приложение на нем (Рисунок 17).

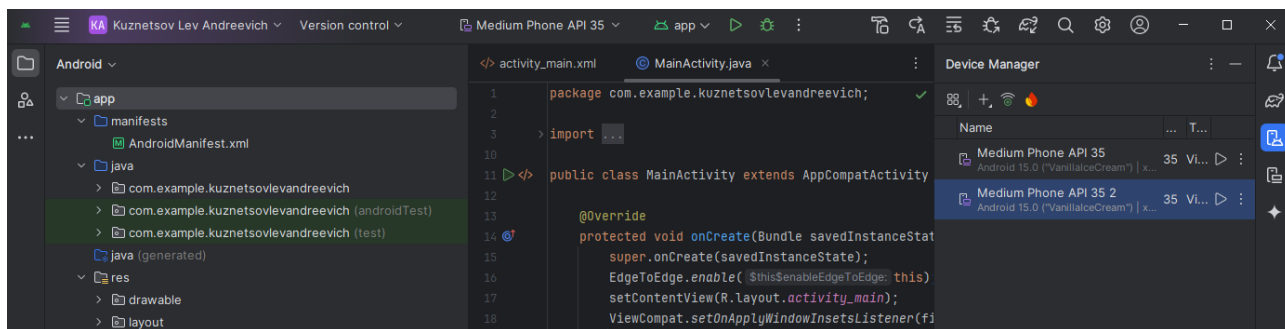


Рисунок 17 - Запуск приложения на созданном виртуальном устройстве

Запущенное устройство будет доступно в пункте "Running Devices", который автоматически откроется при окончании сборки проекта. Функционал приложения будет полностью идентичен с физическим устройством (Рисунок 18). И на этом завершается раздел с запуском проекта на виртуальном и физическом устройствах, в ходе которого мы испытали собственное приложение на физическом устройстве, а также создали виртуальное устройство для последующей работы с ним. Также на Рисунке 18 проект запущен в режиме "Run", а жучок рядом представляет собой режим отладки.

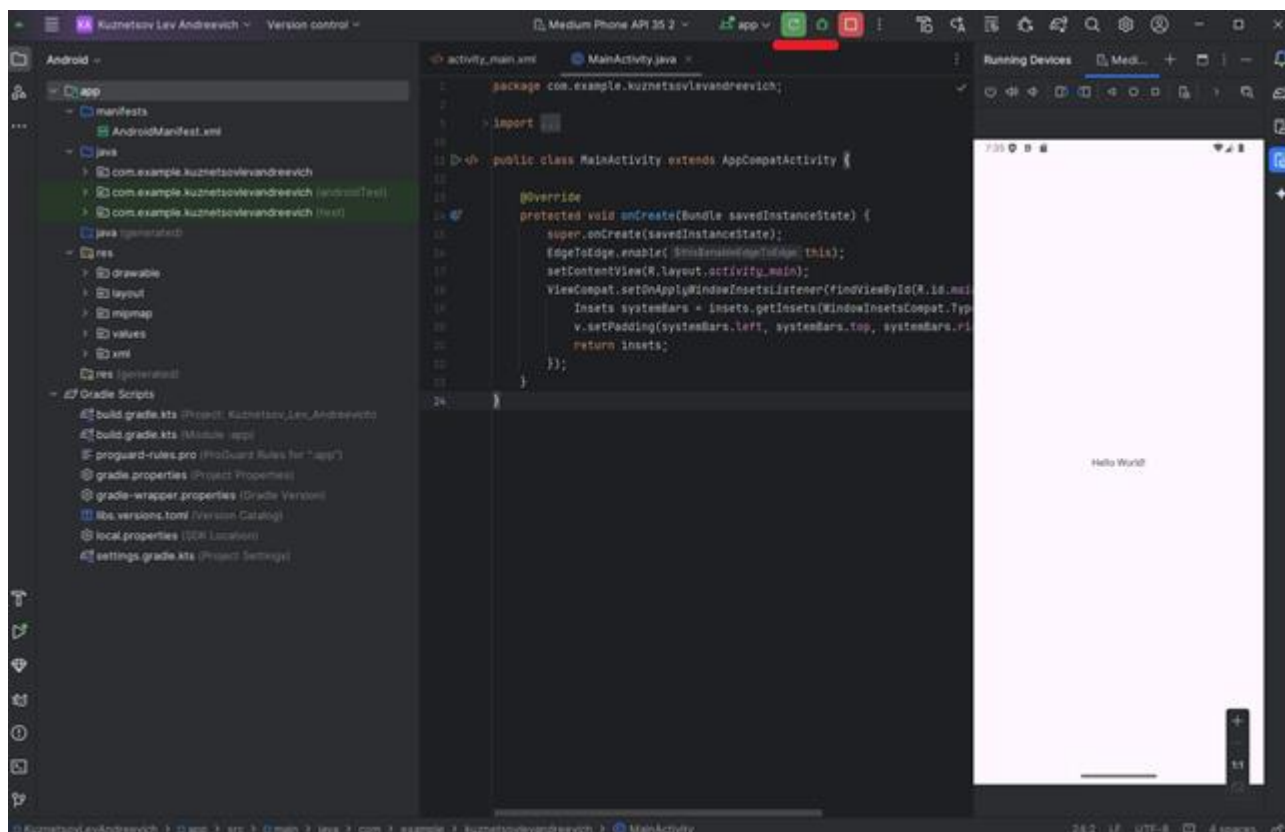
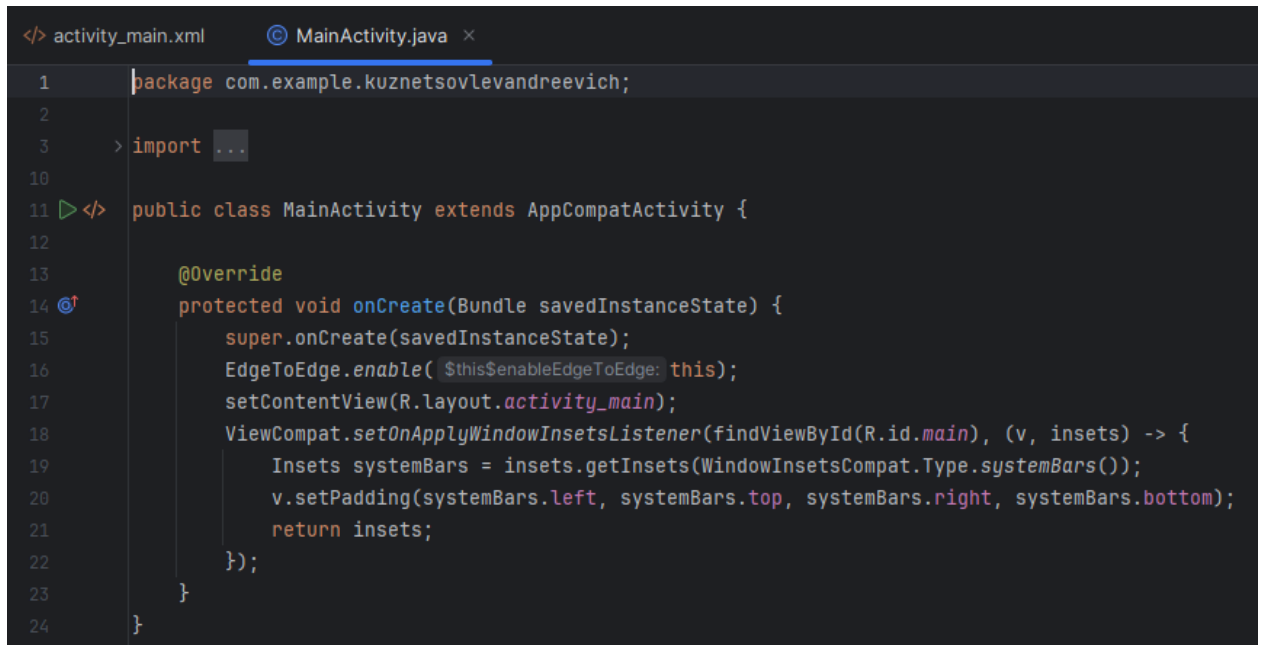


Рисунок 18 - Тестирование приложения на виртуальном устройстве

1.5 создание графического интерфейса

Выполнение приложения Android по умолчанию начинается с класса MainActivity, который по умолчанию открыт в Android Studio (Рисунок 17).



```
</> activity_main.xml  MainActivity.java x
1 package com.example.kuznetsovlevandreevich;
2
3 > import ...
10
11 </> public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         EdgeToEdge.enable(this);
17         setContentView(R.layout.activity_main);
18         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
19             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
20             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
21             return insets;
22         });
23     }
24 }
```

Рисунок 19 - Класс MainActivity

Каждый отдельный экран или страница в приложении описывается таким понятием как activity. В литературе могут использоваться различные термины: экран, страница, активность. Если запустить приложение, то на экране мы, по сути, увидим определенную activity. По умолчанию MainActivity содержит только один метод onCreate() - один из важнейших методов жизненного цикла активности в Android. Он вызывается при создании активности, то есть когда она впервые запускается системой. Этот метод является основным местом для инициализации всех компонентов активности, таких как настройка интерфейса, связывание с ресурсами и другие начальные операции. А ресурсы разметки графического интерфейса передаются в метод setContentView() - это ключевой метод в Android, который используется для установки разметки (UI) активности. Он связывает файл XML, содержащий описание пользовательского интерфейса, с конкретной активностью. После того как мы вызовем setContentView(), Android отобразит UI, который мы описали в этом XML-файле, в том числе все элементы

интерфейса, такие как кнопки, текстовые поля, изображения и другие компоненты.

Android Studio позволяет работать с визуальным интерфейсом как в режиме кода, так и в графическом режиме (Рисунок 20). Так, по умолчанию файл открыт в графическом режиме, и мы наглядно можем увидеть, как у нас примерно будет выглядеть экран приложения. И даже набросать с панели инструментов какие-нибудь элементы управления, например, кнопки или текстовые поля, при этом мы сможем изменить их свойства как через код, так и через пользовательский интерфейс, предлагаемый Android Studio.

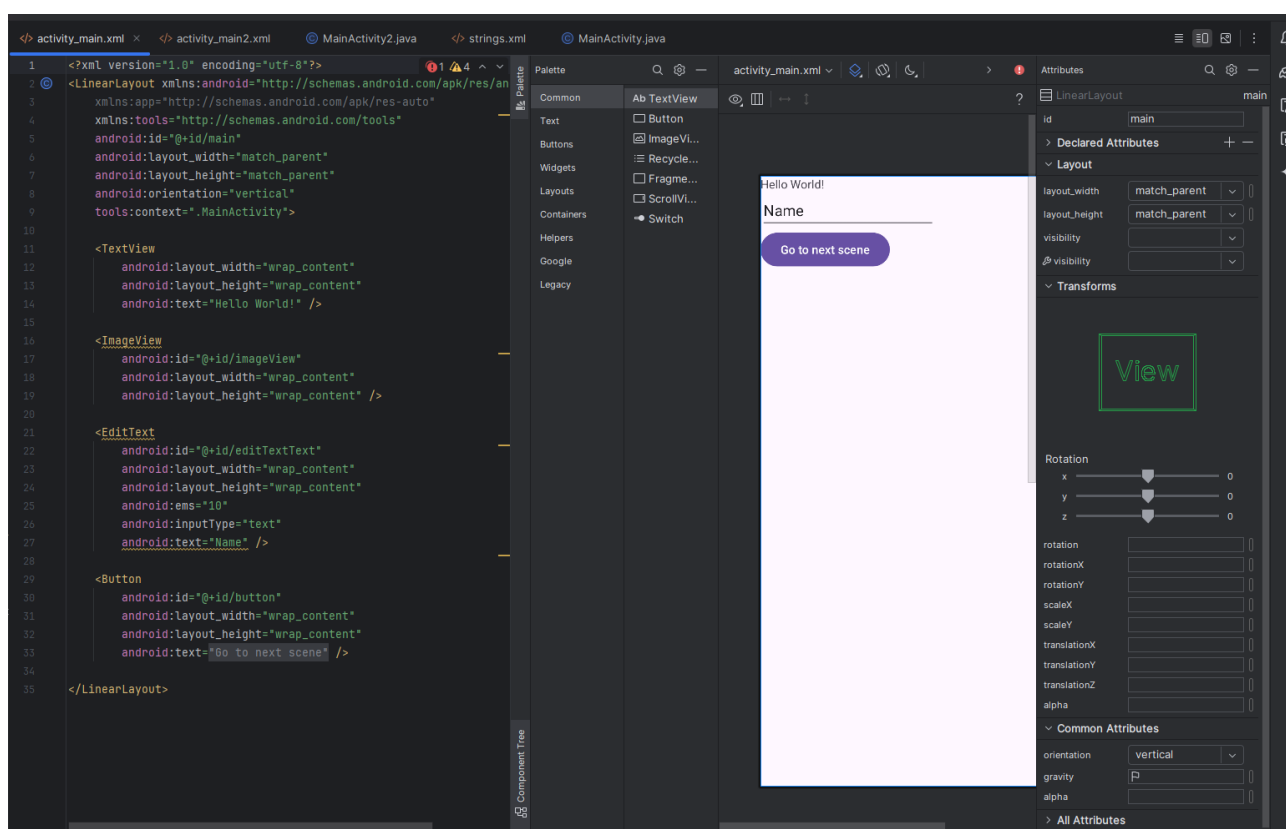


Рисунок 20 - Графический и кодовый вид приложения

Но также мы можем работать с файлом в режиме кода, поскольку `activity_main.xml` — это обычный текстовый файл с разметкой `xml`. Для переключения к коду нажмем на кнопку `Code` над графическим представлением. (Дополнительно с помощью кнопки `Split` можно переключиться на комбинированное представление код + графический дизайнер) (Рисунок 21).

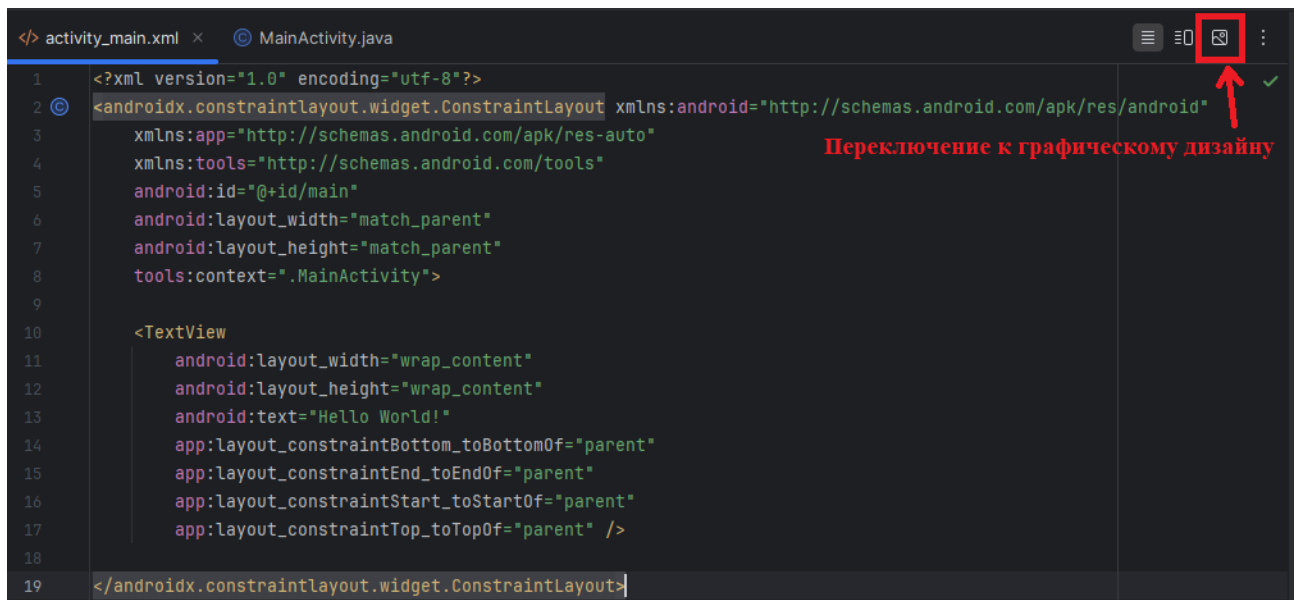


Рисунок 21 - Переключение к графическому дизайну

Большинство визуальных элементов, наследующихся от класса View, такие как кнопки, текстовые поля и другие, располагаются в пакете android.widget.

При определении визуального интерфейса есть три стратегии:

- создать элементы управления программно в коде java;
- объявить элементы интерфейса в XML;
- сочетание обоих способов - базовые элементы разметки определить в XML, а остальные добавлять во время выполнения.

1.6 Вёрстка в android. Язык xml

Как правило, для определения ресурсов, а в том числе и визуального интерфейса, в проектах под Android используются специальные файлы xml. Эти файлы являются ресурсами разметки и хранят определение визуального интерфейса в виде кода XML.

Объявление пользовательского интерфейса в файлах XML позволяет отделить интерфейс приложения от кода. Что означает, что мы можем изменять определение интерфейса без изменения кода java. Кроме того, объявление разметки в XML позволяет легче визуализировать структуру интерфейса и облегчает отладку.

XML — это язык свободного описания структур документов. То есть, если необходимо, чтобы в документе присутствовал какой-либо элемент, то мы для него определяем некоторый тег (маркер в тексте). Например, для описания элемента «текстовая строка» можно условиться использовать тег, где первая метка указывает начало описания элемента, а вторая (со знаком /) — конец описания. Между парой тегов помещается текстовое представление содержимого элемента. Для каждого элемента применяется своя пара тегов, при этом однотипные элементы описываются одинаковой парой тегов. Таким образом, для описания двух строк нужны две пары тегов (Рисунок 22).

```
<string name="app_name">Kuznetsov Lev Andreevich</string>  
<string name="sarsaparilla">Nothingness</string>
```

Рисунок 22 - Описание строк при помощи тегов

В открывающем теге можно поместить атрибуты описываемого элемента, такие как цвет, размер, начертание, выравнивание и т. п., то есть описать особенности формируемого элемента. Атрибут — это свойство описываемого элемента. При этом у однотипных элементов полный набор атрибутов будет совпадать, но в описании можно использовать не все свойства. Каждому имени атрибута присваивается значение, записанное в виде текстовой строки, то есть заключенное в двойные кавычки. Разделяются свойства пробелом либо переносом строки. Вернемся к рассматриваемому примеру (Рисунок 23).

```
<string name="app_name" color="red" align="center">Kuznetsov Lev Andreevich</string>
```

Рисунок 23 - Описание атрибутов элемента в тегах

Данная разметка (Рисунок 23) описывает текстовую строку, написанную красным шрифтом (начертание и размер установлены по умолчанию, поскольку эти свойства не указаны при описании) с выравниванием в центре страницы.

Каким бы свободным не был стиль XML-документа, все-таки существуют правила его формирования:

- в языке XML все теги парные. Это значит, что у каждого открывающего тега обязательно должен присутствовать закрывающий тег. Это правило позволяет описывать вложенные элементы, то есть помещать внутри одного элемента

другие. Если тело тега пусто, то два тега записываются в один, который завершается косой чертой;

- документ может содержать декларацию — строку заголовка, в которой указывается версия языка и используемая текстовая кодировка;
- имена тегов могут содержать буквы, цифры и специальные знаки, такие как знак подчеркивания (), но должны начинаться с буквы. Теги записываются с соблюдением регистра, поскольку XML регистрозависим;
- если возникает необходимость использования одинаковых имён элементов для разного типа структур документа, применяют понятие пространства имен. Чтобы различать такие элементы, необходимо задать соответствие — специальный уникальный идентификатор ресурса или URI с конкретным именем элемента. В качестве идентификатора чаще всего используется адрес своего (необязательно реально существующий) ресурса. Пространство имен определяется благодаря атрибуту `xmlns` в начальном теге элемента;
- в XML-тексте комментарии выделяются тегами.

Выше перечисленные правила формирования XML-документа можно заметить при ознакомлении с созданным проектом в Android Studio (Рисунок 24).

Элемент — это структурная единица XML-документа. Границы элементов маркируются одинаковыми начальным и конечным тегами.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Рисунок 24 - Код файла `activity_main.xml`

На Рисунке 24 представлен элемент `TextView`, обязательными атрибутами которого являются:

- `android:layout_width` - этот атрибут указывает ширину `TextView`. В данном случае используется значение `wrap_content`, что означает, что ширина будет автоматически подстраиваться под длину текста;
- `android:layout_height` - этот атрибут указывает высоту `TextView`. Значение `wrap_content` означает, что высота будет подстраиваться под высоту текста.
- `android:text` - это основной атрибут, который указывает текст, который должен отображаться в `TextView`. В примере это просто строка "Hello World!".

Внутри этой границы может быть текстовая строка значения элемента. Элемент может быть также представлен пустым тегом, то есть не включающим в себя другие элементы и/или символьные данные.

Помимо текстового значения элемент может включать другие элементы. Такие элементы называются дочерними (`child`) элементами. Дочерних элементов может быть несколько. Элемент, который окружает дочерний элемент, называется родительским (`parent`). По естественным причинам у дочернего элемента может быть только один родительский. Важно, чтобы любой дочерний элемент располагался целиком внутри родительского. То есть пары открывающих и закрывающих тегов всех дочерних элементов должны быть заключены (окружены) парой открывающего и закрывающего тегов родительского элемента. В случае нарушения этого правила любая программа не сможет прочесть ваш документ и выдаст сообщение об ошибочности. Автор документа, вкладывая одни элементы в другие, задает иерархическую структуру внутри документа.

1.7 Ресурсы в android

Создавая приложение для Android, помимо написания программ на языке Java необходимо также работать с ресурсами. В экосистеме Android принято отделять такие файлы, как изображения, музыка, анимации, стили, макеты окон,

строковые константы — в общем все части оформления GUI (Graphical User Interface — графический интерфейс пользователя) от программного кода. Большая часть ресурсов (за исключением мультимедийных) хранятся во внешних XML-файлах. При создании и развитии программного проекта внешние ресурсы легче поддерживать, обновлять и редактировать.

Как уже было показано, каждое приложение на Android содержит каталог для ресурсов `res/`. Доступ к информации в каталоге ресурсов из приложения осуществляется через класс `R`, который автоматически генерируется средой разработки.

В общем случае ресурсы представляют собой файл (например, изображение) или значение (например, заголовок программы), связанные с создаваемым приложением по имени ресурса. Удобство использования ресурсов заключается в том, что их можно заменять/изменять без изменения программного кода приложения или компиляции. Поскольку имена файлов для ресурсов фактически будут использованы как имена констант в `R`, то они должны удовлетворять правилам написания имен переменных в Java. Так как разработка ведется на различных ОС (Windows, Mac, Linux), то также есть еще ограничения. В итоге имена файлов должны состоять исключительно из букв в нижнем регистре, чисел и символов подчеркивания.

В Android используются два подхода к процессу создания ресурсов — первый подход заключается в том, что ресурсы задаются внутри файла и тогда его имя задается в месте его описания. Второй подход — ресурс задается в виде самого файла, и тогда имя файла уже и есть имя ресурса.

Для различных типов ресурсов, определенных в проекте, в каталоге `res` создаются подкаталоги. Поддерживаемые подкаталоги (Рисунок 25):

- `animator/`: xml-файлы, определяющие анимацию свойств;
- `anim/`: xml-файлы, определяющие tween-анимацию;
- `color/`: xml-файлы, определяющие список цветов;
- `drawable/`: Графические файлы (.png, .jpg, .gif);
- `dimensions/`: xml-файлы, определяющие размерности элементов;

- `drawable/`: Графические файлы, используемые для иконок приложения под различные разрешения экранов;
- `layout/`: xml-файлы, определяющие пользовательский интерфейс приложения;
- `menu/`: xml-файлы, определяющие меню приложения;
- `raw/`: различные файлы, которые сохраняются в исходном виде;
- `values/`: xml- Каталог `values/` является одной из ключевых частей проекта Android и предназначен для хранения XML-файлов с различными ресурсами, которые используются в приложении. Эти ресурсы позволяют организовать и централизовать данные, что делает код более гибким, удобным в сопровождении и адаптируемым для разных конфигураций устройств, локализаций и тем оформления. Каталог `values/` играет важную роль в структурировании данных внутри Android-приложения. Использование XML-файлов из этого каталога позволяет избежать «жесткого» кодирования строк, цветов, размеров и других параметров, что делает проект более гибким, удобным для локализации и адаптации к различным устройствам;
- `xml/`: произвольные xml-файлы;
- `font/`: файлы с определениями шрифтом и расширениями `.ttf`, `.otf` или `.ttc`, либо файлы XML, которые содержат элемент;

Чаще других используют следующие ресурсы: разметка (`layout`), строки (`string`), цвета (`color`) и графические рисунки (`bitmap`, `drawable`):

- `layout/` – для разметки интерфейса (`activity_main.xml`);
- `string/` – для хранения текстовых значений (`strings.xml`);
- `color/` – для хранения палитр цветов (`colors.xml`);
- `drawable/` – для хранения изображений и графических элементов;
- `bitmap/` – для отображения изображений, их обработки, изменения размеров, поворота и других манипуляций.

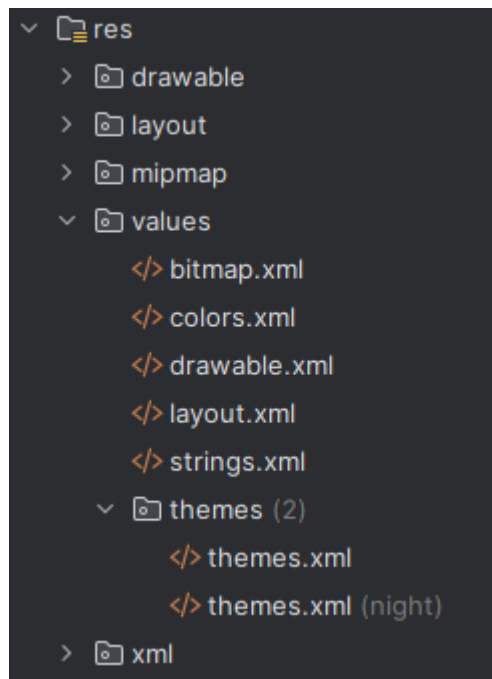


Рисунок 25 - Подкаталоги в каталоге res

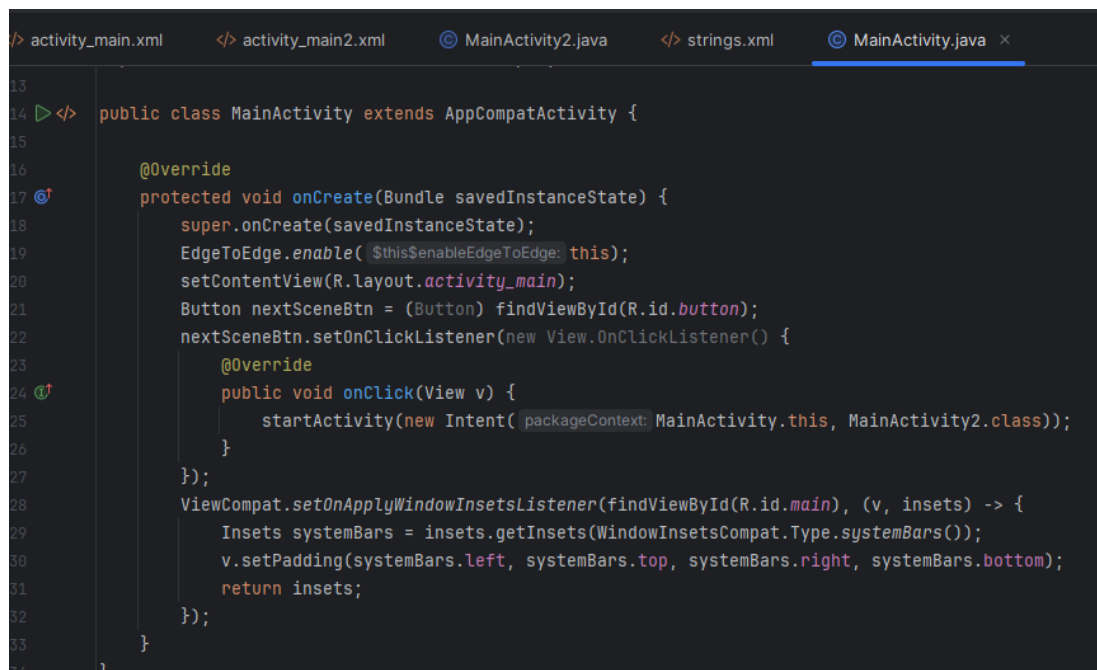
1.8 Создание интерфейса пользователя при помощи ресурсов

Графический интерфейс создается с помощью представлений (View - это базовый строительный блок для всех UI-компонентов в приложении) и групп представлений (ViewGroup - это специальный класс, который является контейнером для других View объектов (или даже других ViewGroup)). Эти элементы размещаются на активности, их описания помещаются в файл манифеста, а действия с объектами прописываются программно в файле кода MainActivity.java (Рисунок 26) в виде методов классов, наследуемых от классов View и ViewGroup или атрибутивно в файле разметки layout/activity_main.xml. У файла разметки также имеется графический вид Graphical layout — системная имитация мобильного устройства.

В файле определяются все графические элементы и их атрибуты, которые составляют интерфейс. При создании разметки в XML следует соблюдать некоторые правила: каждый файл разметки должен содержать один корневой элемент, который должен представлять объект View или ViewGroup.

По умолчанию при создании проекта с пустой activity уже есть один файл ресурсов разметки activity_main.xml. В нем корневым элементом является

элемент `ConstraintLayout`, который содержит элемент `TextView`. Как правило, корневой элемент содержит определение используемых пространств имен XML.

The image shows a code editor window for MainActivity.java. The code defines a MainActivity class that extends AppCompatActivity. It overrides the onCreate method, which calls super.onCreate, enables edge-to-edge, sets the content view to R.layout.activity_main, finds a button with R.id.button, and sets an onClick listener. The listener calls startActivity with an intent to MainActivity2.class. It also sets a window insets listener for the main view to handle system bars padding.

```
13
14 <> public class MainActivity extends AppCompatActivity {
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         EdgeToEdge.enable(this);
20         setContentView(R.layout.activity_main);
21         Button nextSceneBtn = (Button) findViewById(R.id.button);
22         nextSceneBtn.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View v) {
25                 startActivity(new Intent(MainActivity.this, MainActivity2.class));
26             }
27         });
28         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
29             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
30             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
31             return insets;
32         });
33     }
34 }
```

Рисунок 26 - Код файла MainActivity.java

1.9 компоненты разметки

Компоненты разметки имеют конкретный внешний вид и конкретные задачи. В SDK Android находятся множество различных компонентов, однако будет рассмотрен основной перечень из них. Компоненты разметки можно выбрать в редакторе вручную (Рисунок 27).

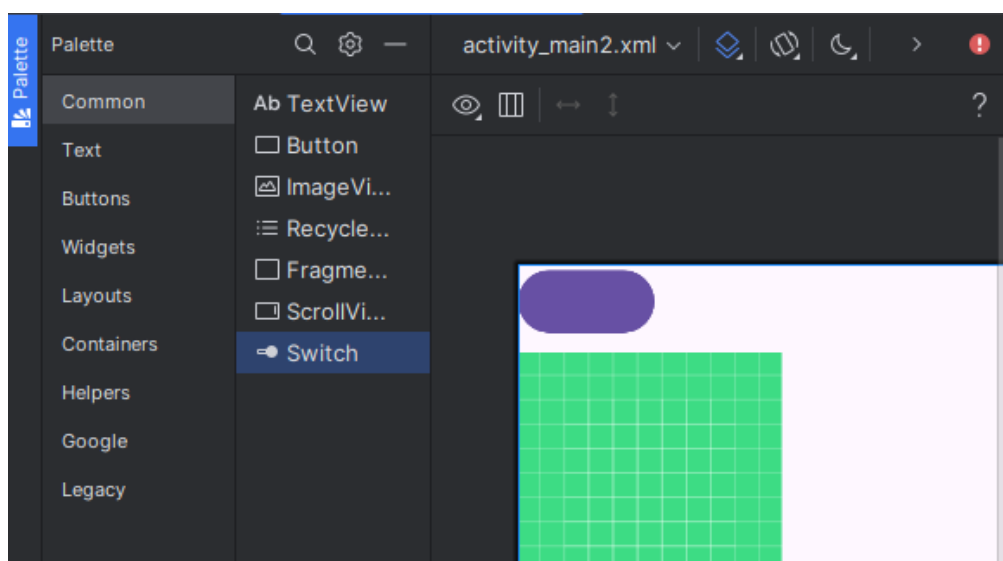


Рисунок 27 - Каталог элементов в редакторе Android Studio

1.9.1 TextView

TextView – компонент, предназначенный для простого вывода текста на экран. Он просто отображает текст без возможности его редактирования (Рисунок 28).

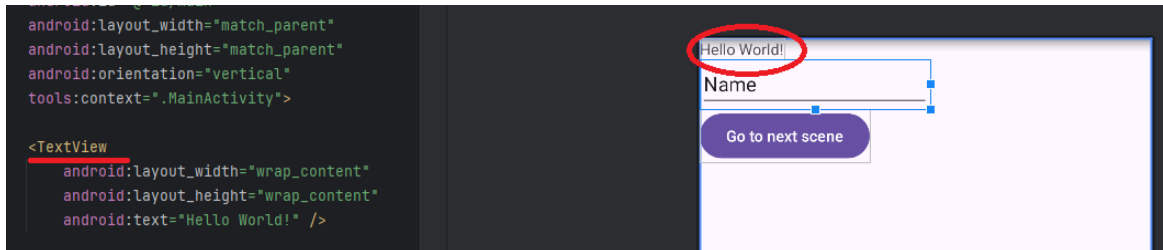


Рисунок 28 - Графическое и кодовое изображения элемента TextView

Опишем атрибуты данного элемента:

- `android:layout_width="wrap_content"` – задаёт ширину элемента. В данном случае ширина будет соответствовать длине текста, который мы указали в атрибуте `android:text`;
- `android:layout_height="wrap_content"` – задаёт высоту элемента. В данном случае высота будет равна высоте текста;
- `android:text="Hello World!"` - задает текст, который будет отображаться внутри TextView. В данном случае, текст — это строка "Hello World!".

1.9.2 EditText

EditText – является подклассом класса TextView. Он также представляет текстовое поле, но теперь уже с возможностью ввода и редактирования текста. Таким образом, в EditText мы можем использовать все те же возможности, что и в TextView (Рисунок 29).



Рисунок 29 - Графическое и кодовое изображения элемента EditText

Опишем необходимые атрибуты EditText:

- `android:layout_width="130dp"` – задаёт ширину элемента. В данном случае этот атрибут задает фиксированную ширину EditText равную 130dp (density-independent pixels), независимо от содержимого или других атрибутов;
- `android:layout_height="wrap_content"` – задаёт высоту элемента. В данном случае высота EditText будет автоматически подстраиваться под высоту содержимого, т.е. она будет такой, чтобы вместить весь текст, который в нем вводится. Для однострочного текста высота будет минимальной, но если текст займет несколько строк, высота увеличится.

1.9.3 Button

Button – один из часто используемых компонентов. Ключевой особенностью кнопок является возможность взаимодействия с пользователем через нажатия (Рисунок 30).

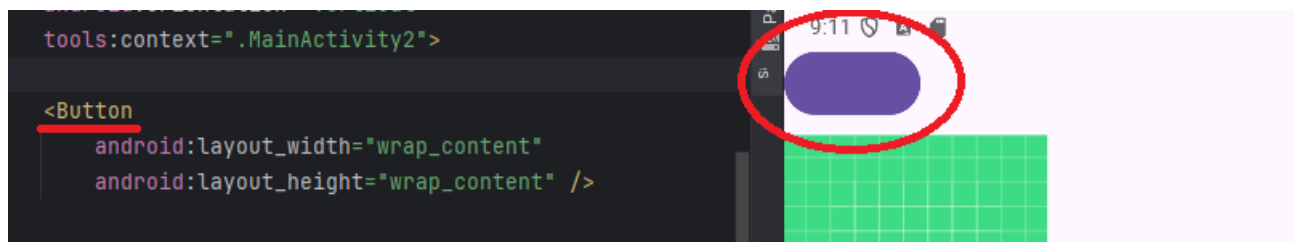


Рисунок 30 - Графическое и кодовое изображения элемента Button

Опишем атрибуты элемента Button:

- `android:layout_width="wrap_content"` - задаёт ширину элемента. В данном случае ширина будет равна контейнеру, в котором находится Button;
- `android:layout_height="wrap_content"` - задаёт высоту элемента. В данном случае высота будет равна контейнеру, в котором находится Button.

1.9.4 ImageView

ImageView – является базовым элементом-контейнером для использования графики. Можно загружать изображения из разных источников, например, из ресурсов программы, контент-провайдеров (Рисунок 31).



Рисунок 31 - Графическое и кодовое изображения элемента ImageView

Опишем атрибуты данного элемента:

- android:layout_width="170dp" - задаёт ширину элемента. В данном случае ширина будет равна 170dp (density-independent pixels);
- android:layout_height="188dp" - задаёт высоту элемента. В данном случае высота будет равна 188dp (density-independent pixels);
- app:srcCompat="@drawable/ic_launcher_background" - используется в Android для установки изображения. В данном случае "@drawable/ic_launcher_background" указывает на ресурс изображения.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Решение задачи

Поставив перед собой задачу, перейдём к её решению. Для этого расположим различные базовые компоненты, а именно TextView, ImageView, Button и Plain Text в файле разметки activity_main.xml (Рисунки 32-33).

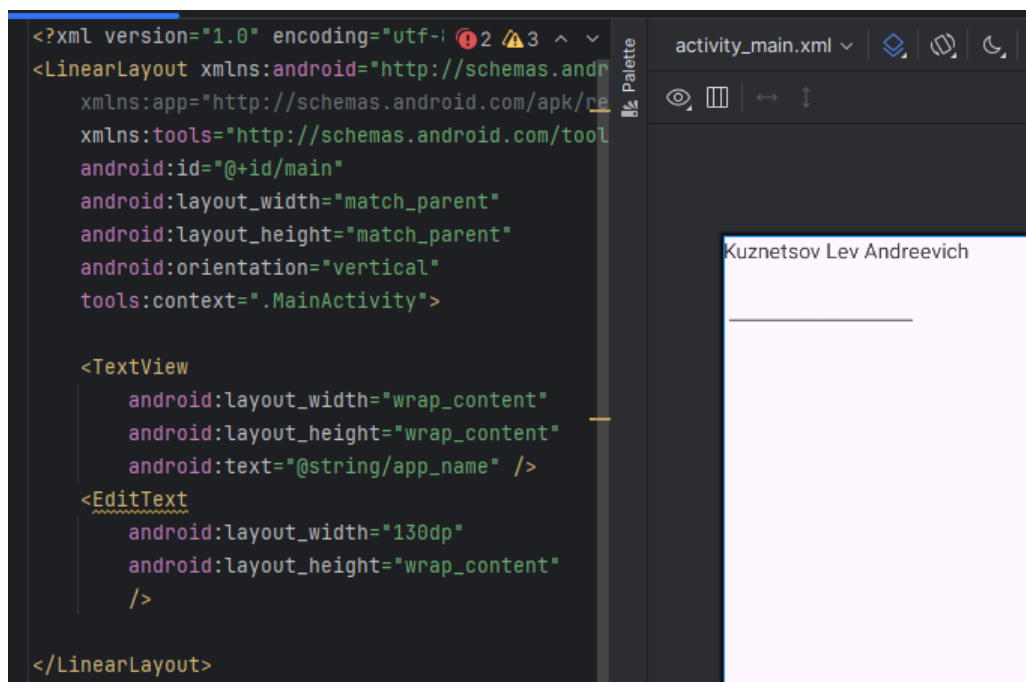


Рисунок 32 - Часть 1 отображения базовых элементов в визуальном редакторе

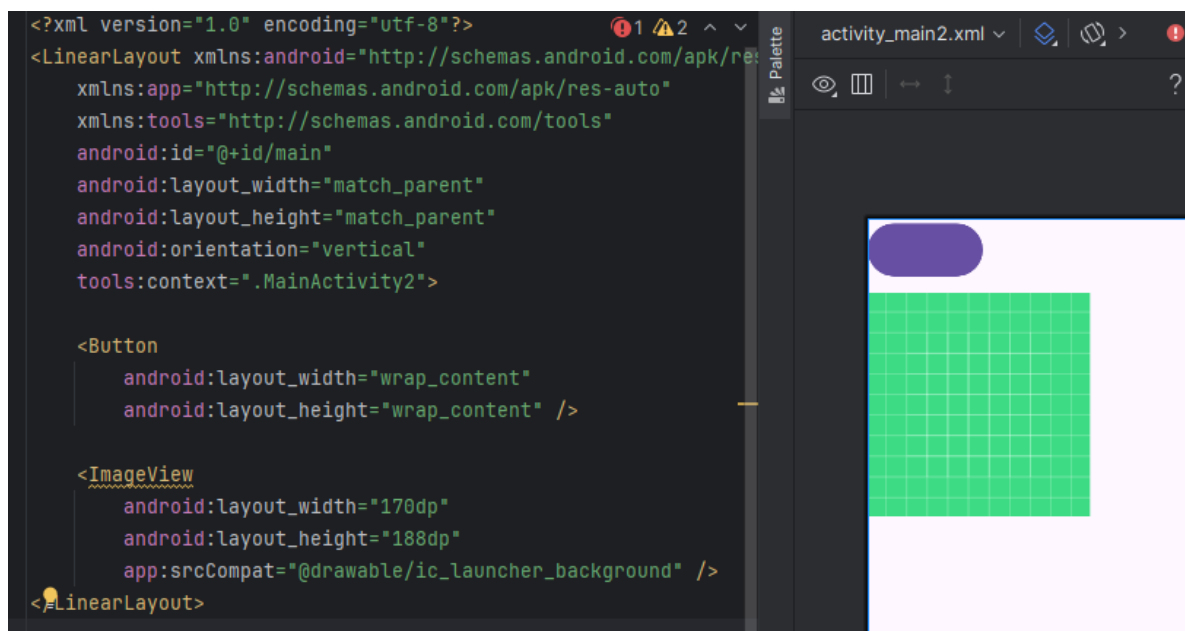


Рисунок 33 - Часть 2 отображения базовых элементов в визуальном редакторе

Как видно из Рисунков 32 и 33 мы взяли из редактора 4 элемента разметки и расположили их в LinearLayout на нашем виртуальном устройстве.

2.1.1 TextView в задаче

Первым элементом на рассмотрение является TextView, который имеет схожие значения атрибутов, перечисленных выше.

Из Рисунка 32 видно, что TextView (Рисунок 34) имеет следующие атрибуты:

- `layout_width="wrap_content"` – задаёт ширину элемента. В данном случае ширина будет соответствовать длине текста, который мы указали в атрибуте `android:text`;
- `android:layout_height="wrap_content"` – задаёт высоту элемента. В данном случае высота будет равна высоте текста;
- `android:text="@string/app_name!"` - задает текст, который будет отображаться внутри TextView. В данном случае, наш текст ссылается на ресурс string (Рисунок 35).



Рисунок 34 - Отображение TextView в нашем приложении



Рисунок 35 - Код ресурса strings.xml

2.1.2 EditText в задаче

Также из Рисунка 32 видно, что EditText (Рисунок 36) имеет следующие атрибуты:

- `layout_width="130dp"` – задаёт ширину элемента. В данном случае этот атрибут задает фиксированную ширину EditText равную 130dp (density-independent pixels), независимо от содержимого или других атрибутов;
- `android:layout_height="wrap_content"` – задаёт высоту элемента. В данном случае высота EditText будет автоматически подстраиваться под высоту содержимого, т.е. она будет такой, чтобы вместить весь текст, который в нем вводится. Для однострочного текста высота будет минимальной, но если текст займет несколько строк, высота увеличится.

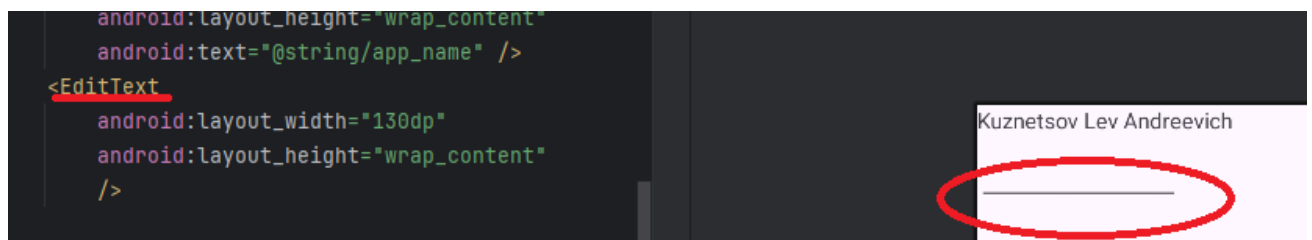


Рисунок 36 - Отображение EditText в нашем приложении

2.1.3 Button в задаче

Далее переходим к рассмотрению Рисунка 33 и всех элементов разметки, расположенных на нём.

Из Рисунка 33 видно, что Button (Рисунок 37) имеет следующие атрибуты:

- `android:layout_width="wrap_content"` - задаёт ширину элемента. В данном случае ширина будет равна контейнеру, в котором находится Button;
- `android:layout_height="wrap_content"` - задаёт высоту элемента. В данном случае высота будет равна контейнеру, в котором находится Button.

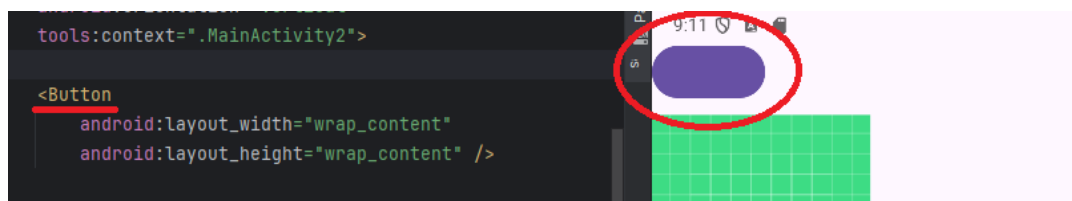


Рисунок 37 - Отображение Button в нашем приложении

2.1.4 ImageView в задаче

Из Рисунка 33 видно, что ImageView(Рисунок 38) имеет следующие атрибуты:

- `android:layout_width="170dp"` - задаёт ширину элемента. В данном случае ширина будет равна 170dp (density-independent pixels);
- `android:layout_height="188dp"` - задаёт высоту элемента. В данном случае высота будет равна 188dp (density-independent pixels);
- `app:srcCompat="@drawable/ic_launcher_background"` - используется в Android для установки изображения. В данном случае `"@drawable/ic_launcher_background"` указывает на ресурс изображения.



Рисунок 38 - Отображение ImageView в нашем приложении

Как итог, приложение работает исправно и все базовый элементы выполняют свой функционал.

ЗАКЛЮЧЕНИЕ

В ходе освоения мы познакомились с официальной интегрированной средой разработки Android Studio, установив её на персональный компьютер, после чего настроили. Создали свой первый проект с базовыми элементами и несколькими файлами разметки, проверив корректность работы при помощи запуска приложения на виртуальном устройстве.