|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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 «Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-21-23 | Муравьев А. О. |
| **Проверил:**  Старший преподаватель кафедры МОСИТ | Шешуков Л.С. |

Москва 2025 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc191678436)

[1 ТЕОРИТИЧЕСКАЯ ЧАСТЬ 4](#_Toc191678437)

[1.1 Установка Android Studio 4](#_Toc191678438)

[1.2 Создание проекта 5](#_Toc191678439)

[1.3 Структура проекта 11](#_Toc191678440)

[1.4 Запуск проекта 13](#_Toc191678441)

[1.5 Создание графического интерфейса 22](#_Toc191678442)

[1.6 Верстка в Android. Язык XML 25](#_Toc191678443)

[1.7 Ресурсы в Android 28](#_Toc191678444)

[1.8 Создание интерфейса пользователя при помощи ресурсов 30](#_Toc191678445)

[1.9 Компоненты разметки 31](#_Toc191678446)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 33](#_Toc191678447)

[2.1 Создание проекта с ФИО студента в названии 33](#_Toc191678449)

[2.2 Разметка 35](#_Toc191678450)

[2.1.1 Текст 35](#_Toc191678451)

[2.1.2 Кнопка 39](#_Toc191678452)

[2.1.3 Картинка 41](#_Toc191678453)

[2.1.4 Поле ввода 43](#_Toc191678454)

[ЗАКЛЮЧЕНИЕ 46](#_Toc191678455)

ВВЕДЕНИЕ

Цель данной работы — ознакомиться с основами проектирования и разработки мобильных приложений под Android, а также научиться использовать файлы разметки (XML) и базовые компоненты пользовательского интерфейса: текстовые поля, кнопки, поля ввода и изображения.

В ходе выполнения задания будет создан проект с несколькими файлами разметки и различными элементами интерфейса, что позволит отработать навыки применения стандартных Android-компонентов и их расположения в макете.

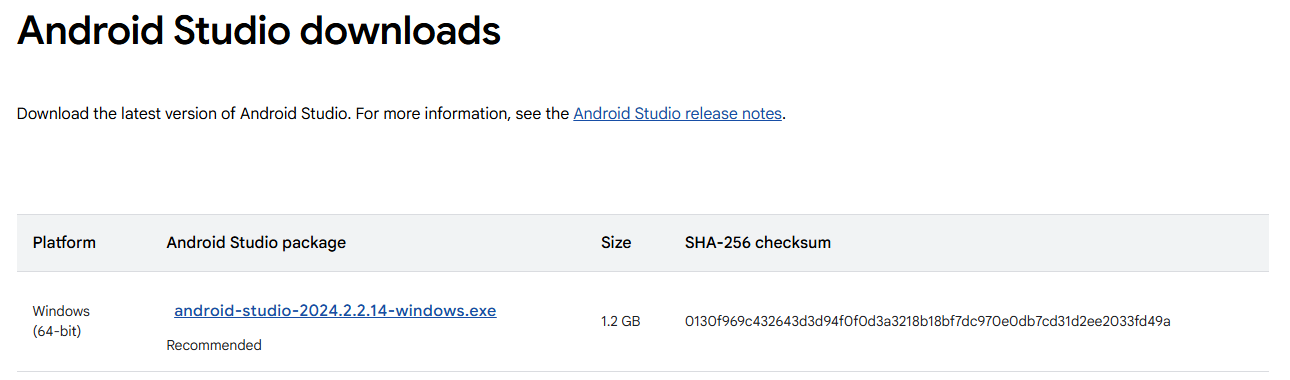
Создать проект с ФИО студента в названии.

Реализовать несколько файлов разметки и создать там базовые компоненты: Текст, Кнопка, Поле ввода, Картинка. (При желании можно расширить перечень собственными компонентами).

# ТЕОРИТИЧЕСКАЯ ЧАСТЬ

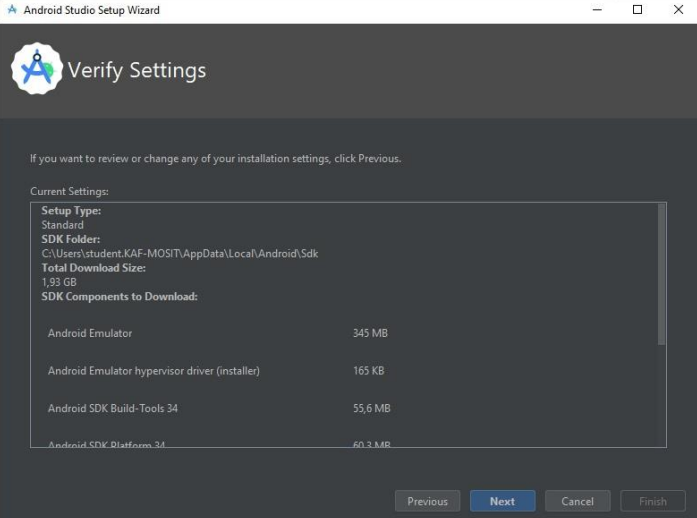
## Установка Android Studio

Существует множество сред разработки, позволяющих создавать Android-приложения, однако Android Studio считается наиболее оптимальным вариантом. Эта среда специально разработана для Android, предоставляет интегрированные инструменты и оптимизированный рабочий процесс, что делает её фаворитом среди разработчиков. Чтобы начать работу, достаточно скачать установочный файл с официального сайта. Нужная строчка в таблице установочных файлов показана на рисунке 1.

  
Рисунок 1 – Ссылка для скачивания установщика Android Studio

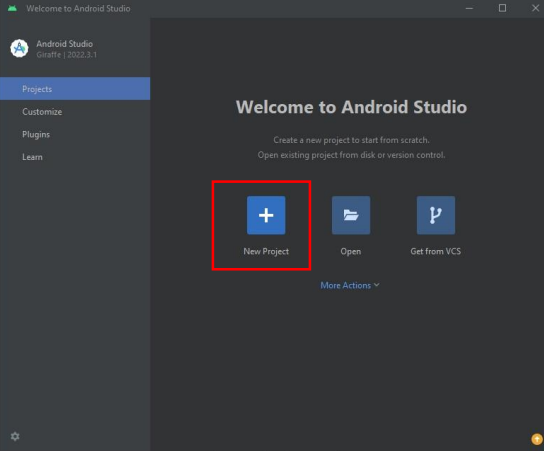
Для разработки на Android, помимо Android Studio, необходимо установить Android Software Development Kit (SDK). Этот набор инструментов обязателен для создания приложений, поскольку включает библиотеки API, эмуляторы устройств, инструменты отладки и многое другое.

При первом запуске Android Studio, если Android SDK не установлен, среда предложит установить недостающие компоненты, автоматически определив необходимые дополнения. К основному Android SDK могут относиться дополнительные пакеты API, инструменты для работы с базами данных, утилиты для профилирования производительности и другие ресурсы, улучшающие процесс разработки, что показано на рисунке 2.

  
Рисунок 2 – Установка Android Studio, настройка устанавлиаемых компонентов

## Создание проекта

После запуска Android Studio необходимо создать новый проект, выбрав на стартовом экране кнопку "Create New Project", который выделен на рисунке 3.

  
Рисунок 3 - Интерфейс управления проектами в Android Studio, выделена кнопка для создания проекта

В Android Studio (AS) есть несколько шаблонов для создания новых проектов или активностей. Каждый шаблон имеет определённые особенности, настройки и используется в зависимости от требований нашего приложения. Вот основные шаблоны:

**Empty Views Activity** (Пустая активность)

* **Описание**: Это минимальный шаблон, который создаёт лишь основную активность с базовой настройкой и пустым экраном. Это идеальный выбор, если мы хотим начать проект с нуля и контролировать все настройки и компоненты.
* **Настройки**:
  + **Name** (Имя): Имя ашей активности (например, MainActivity).
  + **Layout name** (Имя разметки): Имя XML-файла разметки для этой активности.
  + **Language** (Язык): Выбор между Java и Kotlin.
  + Use androidx. artifacts\* (Использовать библиотеки androidx): Выбираем, если хотим использовать библиотеки AndroidX.
* **Почему выбираем**: Этот шаблон является наилучшим выбором, когда мы хотим полную гибкость в создании приложения. Он предоставляет чистое и простое пространство для дальнейшего добавления всех необходимых компонентов, не ограничивая вас стандартными элементами.

2) **Basic Activity** (Базовая активность)

* **Описание**: Этот шаблон создаёт активность с базовым пользовательским интерфейсом, включающим в себя кнопку и текстовое поле, а также механизм обработки нажатий.
* **Настройки**: Аналогичные настройкам для **Empty Activity**, но с добавлением элементов UI, таких как кнопки.
* **Когда использовать**: Если нам нужно создать приложение с простым пользовательским интерфейсом, где сразу можно работать с кнопками и действиями пользователя.

3) **Navigation Drawer Activity** (Активность с панелью навигации)

* **Описание**: Этот шаблон создаёт приложение с боковой панелью навигации (Navigation Drawer), которая позволяет переключаться между экранами.
* **Когда использовать**: когда наше приложение должно поддерживать многозадачность и иметь несколько экранов, доступных через боковое меню.

4) **Bottom Navigation Activity** (Активность с нижней панелью навигации)

* **Описание**: Этот шаблон создаёт приложение с нижней панелью навигации, где каждый элемент панели связан с экраном или фрагментом.
* **Когда использовать**: Для приложений, которые должны иметь простую навигацию между экранами, с удобным доступом из нижней части экрана.

5) **Full Screen Activity** (Активность на весь экран)

* **Описание**: Шаблон для создания полноэкранных приложений. Используется, когда нужно скрыть статус-бар и сделать приложение полностью экранным.
* **Когда использовать**: Если наше приложение требует использования всего пространства экрана, например, для игр или мультимедийных приложений.

6) **Master/Detail Flow** (Шаблон мастер/детали)

* **Описание**: Этот шаблон создаёт структуру, состоящую из двух частей: основной и детали. Он используется, когда у вас есть список элементов, а при выборе элемента отображаются подробности.
* **Когда использовать**: Для приложений, которые работают с большим количеством данных, таких как каталоги или новости.

7) **Google Maps Activity** (Активность с картой Google)

* **Описание**: Создаёт приложение с интеграцией Google Maps. В этом шаблоне уже настроен объект карты и добавлена базовая функциональность.
* **Когда использовать**: Если наше приложение использует карты и геолокацию, и нам нужно начать с готовой интеграции карт.

8) **Fullscreen Activity with a Video** (Полноэкранная активность с видео)

* **Описание**: Шаблон, который создаёт активность для воспроизведения видео в полноэкранном режиме.
* **Когда использовать**: Если наше приложение требует воспроизведения видео в полноэкранном формате.

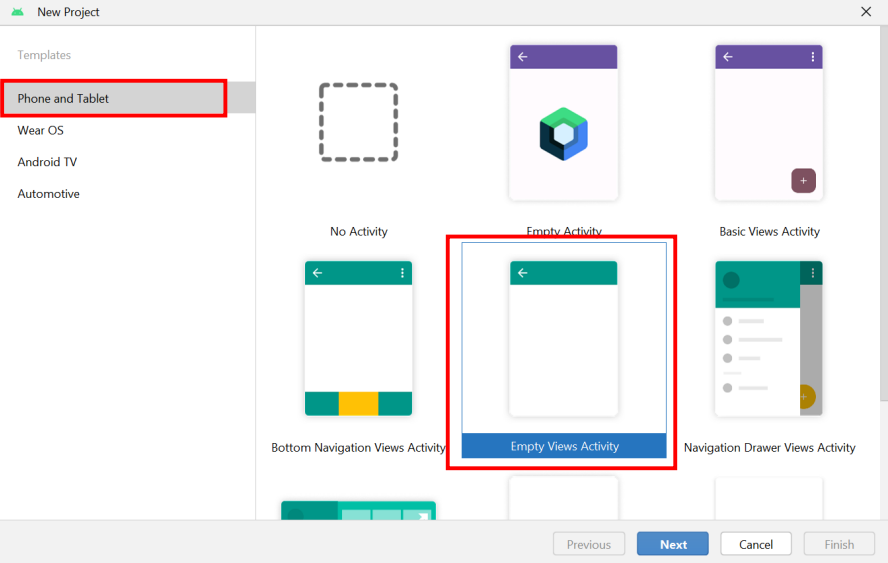
Почему выбираем **Empty Views Activity**?

**Empty Views Activity** — это универсальный и гибкий шаблон, который позволяет разработчику полностью контролировать процесс создания приложения. Мы начинаем с чистого листа и можем добавить любые компоненты и функциональность, которые нам необходимы. Это особенно полезно для опытных разработчиков, которые уже знают, какие элементы интерфейса им нужны, или если проект имеет уникальные требования.

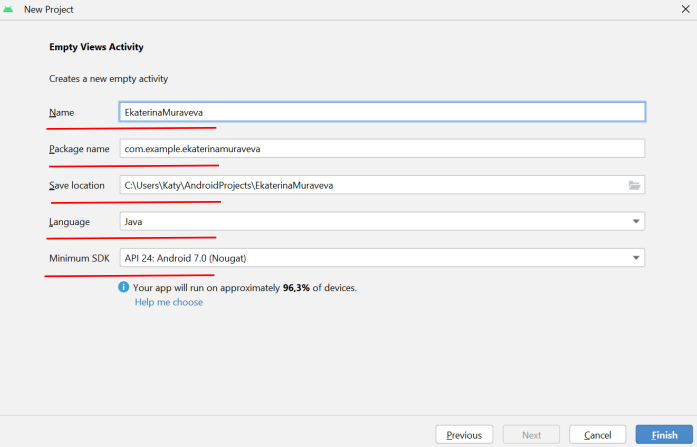
Основные причины выбрать **Empty Views Activity**:

* **Гибкость**: Мы получаем полностью пустое приложение, с минимальной настройкой.
* **Контроль**: Мы контролируем, какие компоненты и библиотеки добавить в проект.
* **Меньше лишнего кода**: В отличие от других шаблонов, в котором уже могут быть добавлены лишние элементы, Empty Views Activity не содержит ничего лишнего.

Нужный элемент выделен на рисунке 4.

  
Рисунок 4 – Интерфейс выбора шаблонов проектов для Android Studio, выделены нужные элементы

Далее необходимо заполнить настройки проекта. Нужные настройки выделены на рисунке 5.

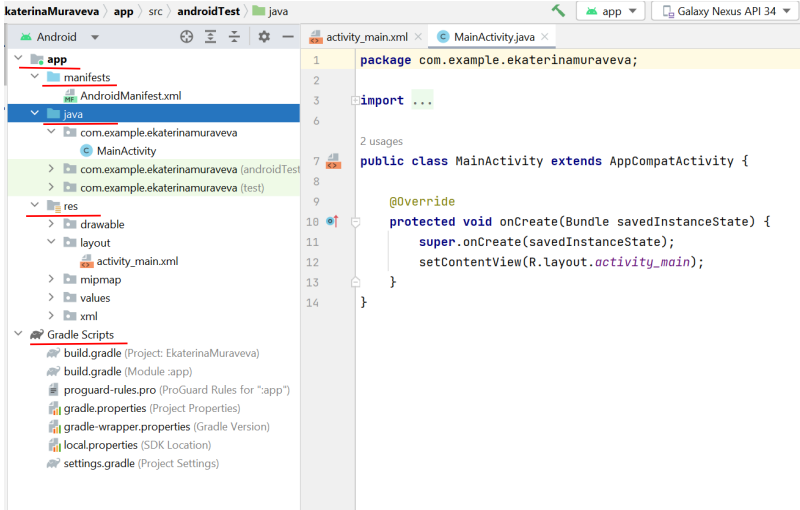
  
Рисунок 5 – Интерфейс настройки создаваемого проекта, выделены параметры для настройки

* Name: отвечает за название проекта;
* Package name: название пакета. Уникальный идентификатор вашего приложения, обычно следует формату com.example.myapp;
* Save location: место на компьютере, где будет сохранен проект;
* Language: язык программирования, на котором будет написан проект;
* Minimum SDK: минимальная версия Android, которую будет поддерживать ваше приложение.

После настройки нужно нажать на кнопку "Finish" и дождаться загрузки проекта.

## Структура проекта

На рисунке 6 продемонстрирована структура пустого проекта.

  
Рисунок 6 – Структура проекта, выделенные ключевые элементы

Модуль «app» представляет собой главный компонент Android-проекта, в котором находятся файлы, необходимые для сборки приложения. В его структуре присутствуют следующие разделы:

1. manifests: Содержит файл AndroidManifest.xml, где задаются основные характеристики приложения — его компоненты (активности, службы и т.д.), необходимые разрешения (для отправки уведомлений, подключения к интернету и т.д.) и настройки (название, тема, иконка и прочее).
2. java: Содержит три пакета, предназначенные для кода самого приложения, для инструментальных тестов, выполняемых на Android-устройствах, и для модульных тестов, запускаемых на компьютере.
3. res: Хранит все ресурсы, не связанные непосредственно с кодом, такие как XML-макеты, элементы пользовательского интерфейса (кнопки, текстовые поля), строковые ресурсы, изображения (drawables), а также стили и темы.

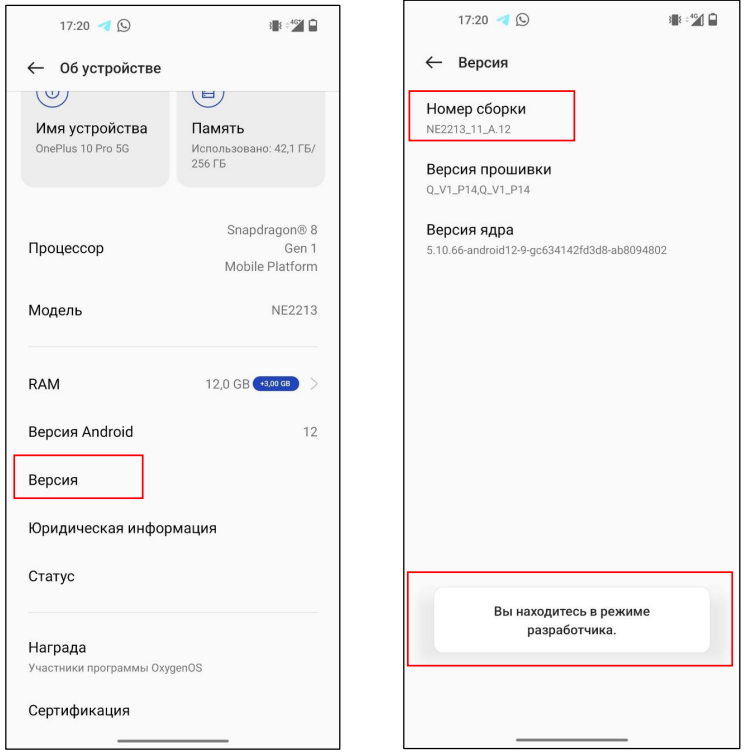
Модуль «Gradle Scripts» отвечает за управление процессом сборки проекта. Он включает следующие файлы:

1. build.gradle (Project: «Название вашего проекта»): Файл на уровне проекта, содержащий общую конфигурацию для всех модулей и ссылки на используемые плагины Gradle.
2. build.gradle (Module: app): Файл на уровне модуля, в котором задаются специфические настройки сборки, такие как версия SDK, версии зависимостей, а также конфигурации сборки (например, для отладки или выпуска).
3. settings.gradle: Файл, в котором перечислены все модули, входящие в проект. Каждый модуль, предназначенный для сборки, должен быть указан здесь.

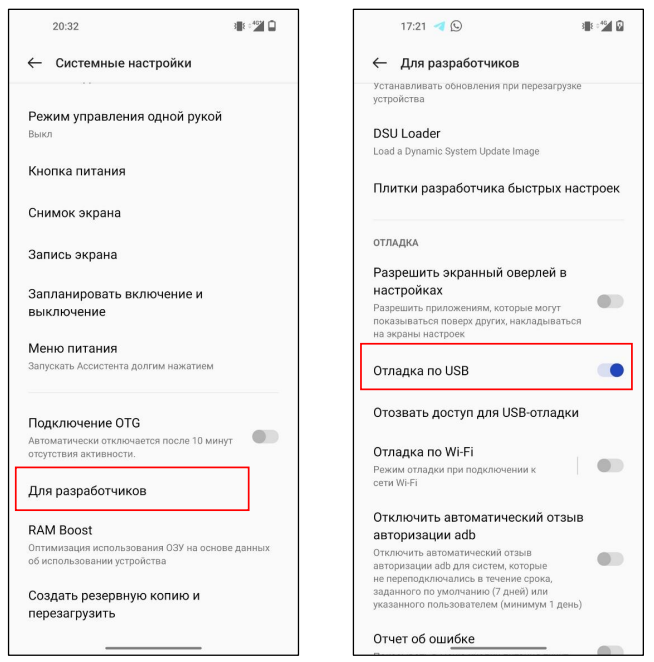
## Запуск проекта

Созданный проект можно запускать как на реальном, так и на виртуальном устройстве. При использовании физического устройства необходимо включить режим разработчика и разрешить отладку по USB. Примечание: приведенная ниже последовательность действий относится к OnePlus 10 Pro; для других моделей шаги и названия настроек могут отличаться.

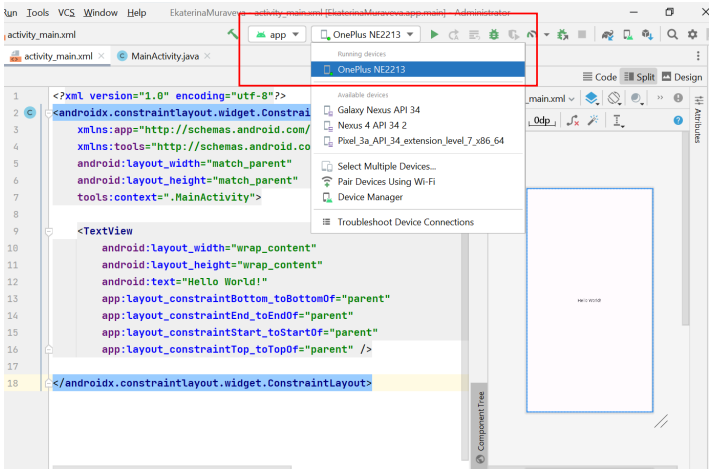
Для активации режима разработчика следует перейти в «Настройки», затем выбрать раздел «Об устройстве», а в подразделе «Версия» выполнить семикратное нажатие на пункт «Номер версии» (или «Номер сборки» в некоторых версиях), после чего появится уведомление об активации режима разработчика. Действия продемонстрированы на рисунке 7.

  
Рисунок 7 – Настройка телефона, выделены интересующие элементы

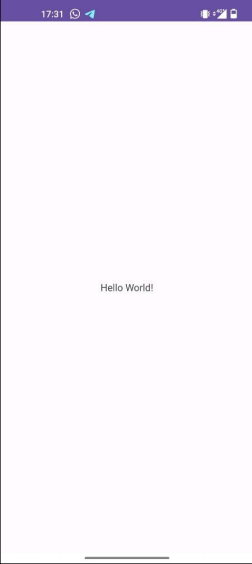
Теперь, для включения отладки по USB вернитесь в «Настройки» и в разделе «Дополнительные настройки» выберите пункт «Возможности разработчика», где необходимо включить пункт «Отладка по USB». Нужные элементы показаны на рисунке 8.

  
Рисунок 8 – Настройка телефона, указан путь к найстроке отладки по USB

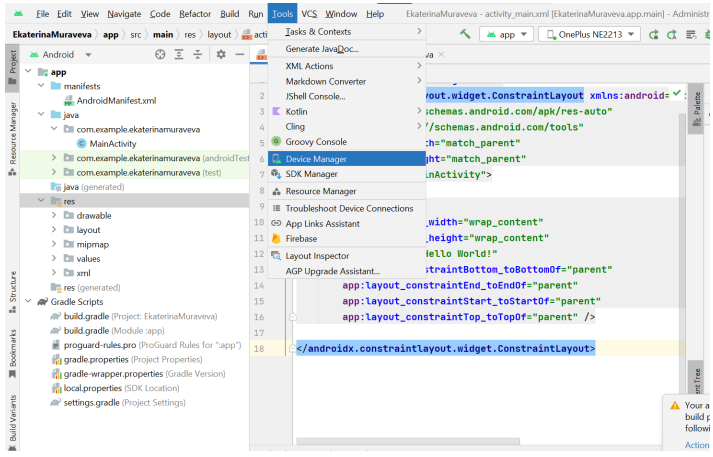
Далее подключаем устройство к компьютеру с открытым Android Studio. Программа должна автоматически обнаружить устройство и отобразить его в разделе "Available devices". Запускаем проект, нажав зеленую кнопку справа. Нужная часть интерфейса выделена на рисунке 9.

  
Рисунок 9 – Запуск проекта на телефоне, выделены элементы настройки запуска проекта

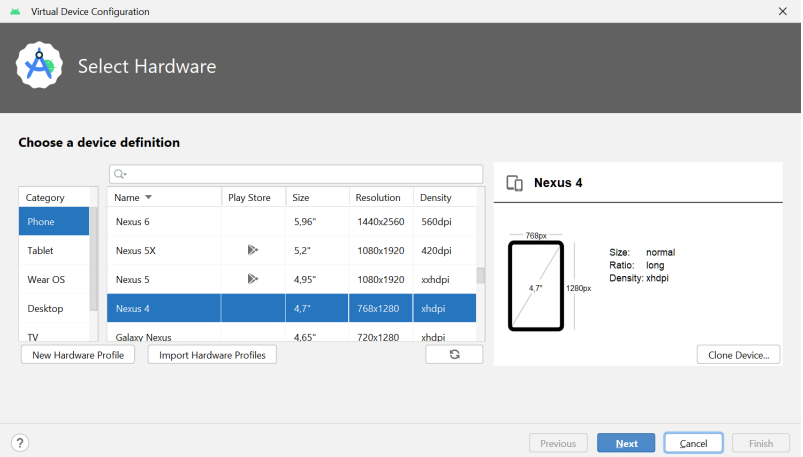
После сборки приложения вы сможете увидеть запущенное приложение на экране вашего телефона. Пример показан на рисунке 10.

  
Рисунок 10 – Запущенное пустое приложение

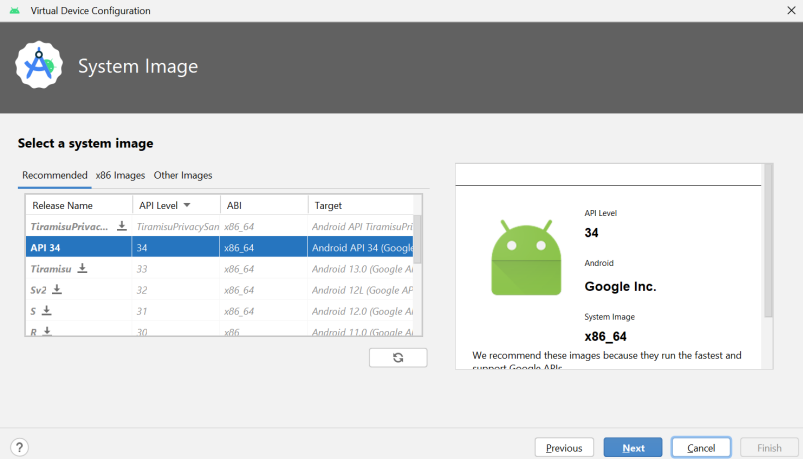
Запуск на виртуальном устройстве осуществляется посредством создания эмулятора в Android Studio. Для этого следует перейти в раздел «Tools», выбрать «Device Manager» и в открывшемся окне нажать «+» для создания нового устройства. Нужная часть интерфейса показана на рисунке 11.

  
Рисунок 11 – Путь к активации Device Manager

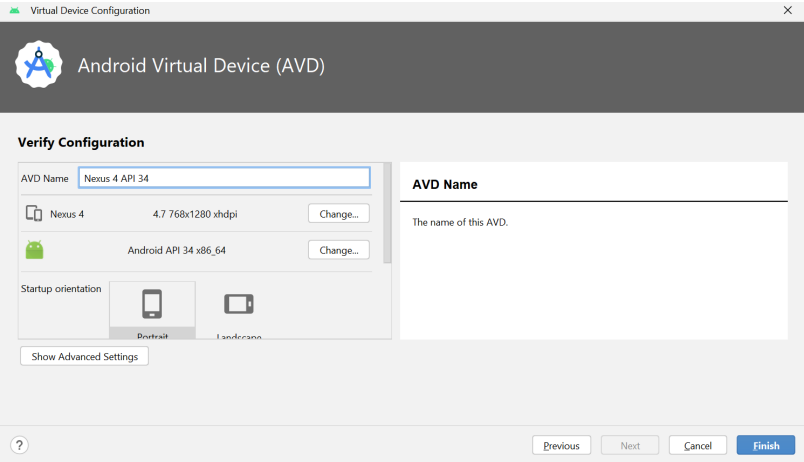
На следующем этапе требуется настроить параметры виртуального устройства: сначала определите тип устройства, затем задайте его размеры, как показано на рисунке 12.

  
Рисунок 12 – Выбор устройства и его размеров для виртуального телефона

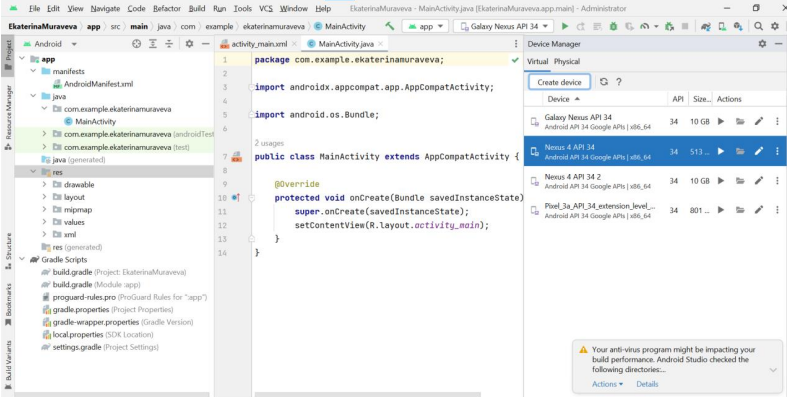
Следующим шагом выбирается версия ОС Android для установки на виртуальное устройство, предпочтительно последняя доступная. Если устройство создается впервые и ОС еще не загружена, необходимо нажать на кнопку загрузки рядом с нужной версией, как продемонстрировано на рисунке 13.

  
Рисунок 13 – Выбор версии ОС для виртуального устройства

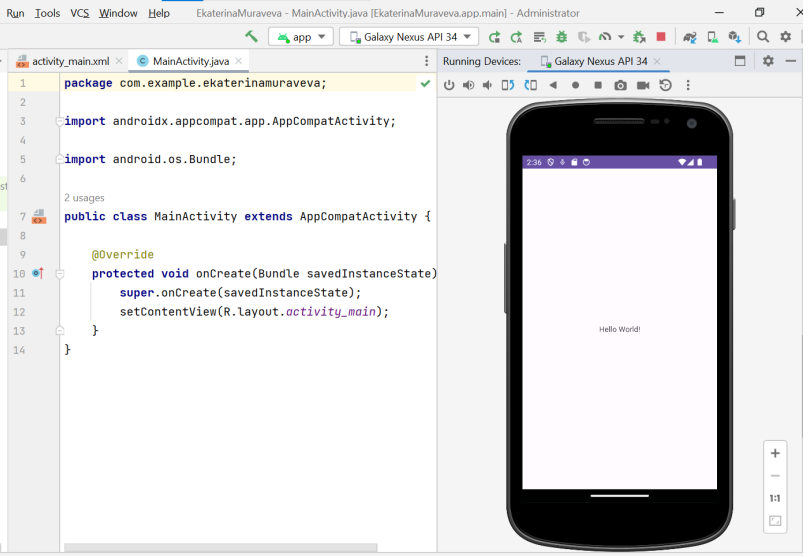
После выбора устройства и версии Android можно просмотреть итоговые параметры, установить тип ориентации экрана и создать виртуальное устройство. Интерфейс показан на рисунке 14.

  
Рисунок 14 – Настройка характеристик и ориентации виртуального устройства

После создания устройства его можно выбрать в разделе «Available devices» и запустить на нем приложение. Нужная часть интерфейса выделена на рисунке 15.

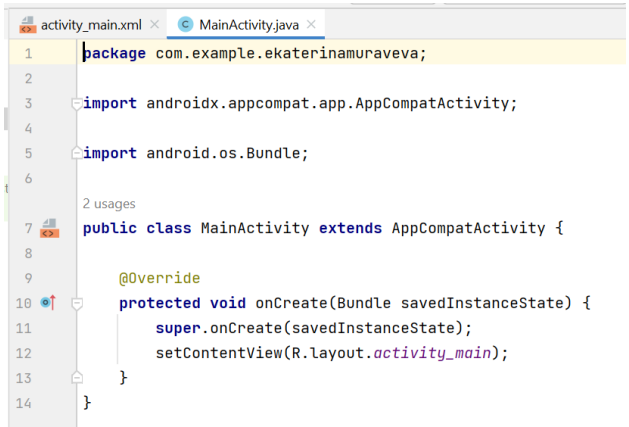
  
Рисунок 15 – Демонстрация настроенного устройства в менеджере устройств

Запущенное устройство будет доступно в пункте "Running Devices", который автоматически откроется при окончании сборки проекта. Функционал приложения будет полностью идентичен с физическим устройством. Запуск виртуального мобильного устройства показан на рисунке 16.

  
Рисунок 16 – Демонстрация работы приложения на виртуальном устройстве

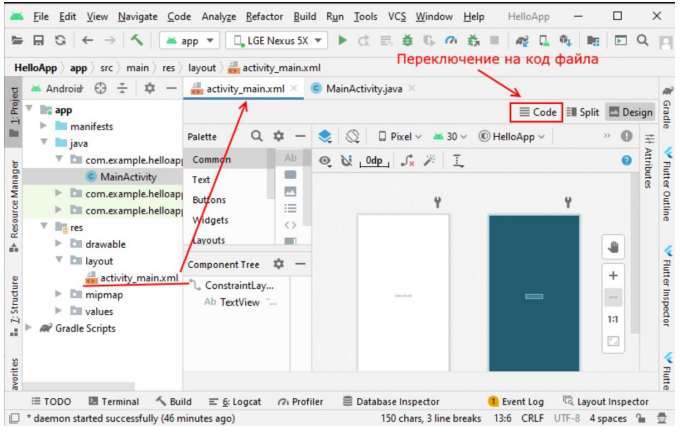
## Создание графического интерфейса

Запуск Android-приложения по умолчанию начинается с класса MainActivity, который автоматически открыт в Android Studio, код которого отображён на рисунке 17.

  
Рисунок 17 – Код класса MainActivity, управляющего поведением главной страницы

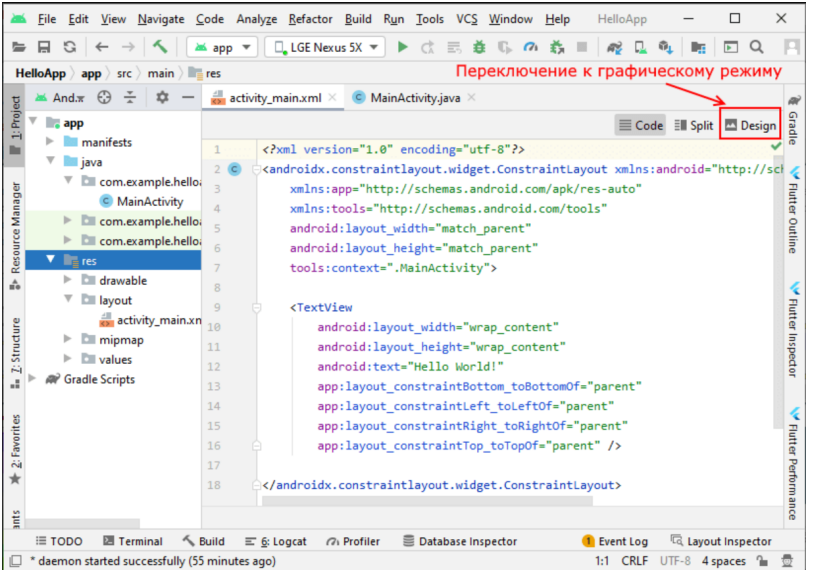
Каждый экран или страница в Android-приложении представляют собой отдельную activity, что может обозначаться как экран, страница или активность. При запуске приложения отображается соответствующая activity.

По умолчанию MainActivity включает лишь метод onCreate(), в котором формируется весь интерфейс приложения посредством передачи ресурса разметки в метод setContentView(), определяющего визуальное представление MainActivity. Визуальный редактор представлен на рисунке 18.

  
Рисунок 18 – Визуальный редактор Android Studio, выделены элементы для перехода в режим кода

Android Studio позволяет работать с визуальным интерфейсом в двух режимах: графическом и текстовом. По умолчанию файл открывается в графическом режиме, что дает возможность наглядно увидеть примерный вид экрана приложения и добавлять элементы управления, такие как кнопки и текстовые поля, с панели инструментов. activity\_main.xml представляет собой обычный текстовый файл с XML-разметкой, который можно редактировать в режиме кода.

Для перехода к коду достаточно нажать кнопку «Code» над графическим представлением. Также доступен режим «Split», позволяющий одновременно видеть код и графический дизайнер, показанный на рисунке 19.

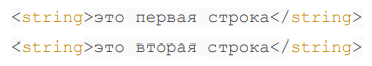
  
Рисунок 19 – Кодовый редактор XML, выделен элемент для переключения к графическому режиму

Большинство визуальных компонентов, унаследованных от класса View – такие как кнопки, текстовые поля и другие — располагаются в пакете android.widget. При создании визуального интерфейса можно использовать одну из трёх стратегий: программное создание элементов управления в коде на Java; объявление элементов в XML-разметке; либо их комбинирование – базовую разметку определить в XML, а остальные элементы добавлять во время выполнения.

## Верстка в Android. Язык XML

Обычно для определения ресурсов, включая визуальный интерфейс, используются специальные файлы XML. Они служат ресурсами разметки и содержат описание интерфейса в виде XML-кода. Такой подход позволяет отделить определение интерфейса от кода Java, что дает возможность изменять внешний вид без модификации программной логики. Кроме того, разметка в XML облегчает визуализацию структуры интерфейса и упрощает отладку.

XML является языком для описания структур документов, где для каждого требуемого элемента определяется свой тег. Например, для описания текстовой строки может применяться тег <string>: открывающий тег указывает начало элемента, а закрывающий (с символом /) — его окончание. Между тегами размещается текстовое содержимое, а для каждого элемента одного типа требуется своя пара тегов. Таким образом, для описания двух строк понадобятся две отдельные пары тегов. Пример показан на рисунке 20.

  
Рисунок 20 – Теги string для установки заготовленных строк

В открывающем теге можно задать атрибуты, определяющие свойства элемента, такие как цвет, размер, начертание, выравнивание и прочее.

Атрибут представляет собой характеристику элемента, и для элементов одного типа набор атрибутов, как правило, совпадает, хотя можно использовать не все из них.

Каждому атрибуту присваивается значение, записанное в виде текстовой строки, заключённой в двойные кавычки, а свойства разделяются пробелами или переносами строки. Вернемся к рассматриваемому примеру на рисунке 21.

  
Рисунок 21 – Заготовленная строка с установлеными атрибутами цвета и выравниванием по центру

Разметка описывает текстовую строку, отображаемую красным цветом (при этом шрифт, его начертание и размер остаются по умолчанию, поскольку данные параметры не заданы) и выровненную по центру страницы.

Несмотря на относительную гибкость стиля XML-документов, при его формировании необходимо соблюдать определённые правила:

* В XML все теги являются парными, то есть каждому открывающему тегу соответствует закрывающий. Это обеспечивает возможность вложенности элементов, позволяющей помещать один элемент внутрь другого. Если содержимое тега отсутствует, открывающий и закрывающий тег объединяются в один с завершающей косой чертой;
* Документ может начинаться с декларации, представляющей собой строку-заголовок, где указываются версия языка и используемая кодировка;
* Имена тегов допускают использование букв, цифр и некоторых специальных символов, таких как знак подчеркивания (\_), однако начинаться они должны с буквы. XML учитывает регистр символов, поэтому имена тегов чувствительны к нему;
* При необходимости использования идентичных имён для элементов различных структур применяется концепция пространства имен. Для различения таких элементов вводится специальный уникальный идентификатор ресурса или URI, который ассоциируется с конкретным именем элемента. Обычно в качестве идентификатора используется адрес ресурса, который может быть условным. Пространство имен определяется посредством атрибута xmlns в начальном теге элемента;
* Комментарии в XML-тексте выделяются специальными тегами, начинающимися с «<!» и заканчивающимися «>».

Элемент представляет собой структурную единицу XML-документа, границы которой определяются парой одинаковых начального и конечного тегов. Внутри элемента может находиться текстовое значение, либо элемент может быть представлен пустым тегом без вложенных данных или других элементов.

Помимо текстового содержимого, элемент может включать в себя дочерние элементы, которых может быть несколько. Элемент, содержащий дочерние элементы, называют родительским, а каждый дочерний элемент имеет только одного родителя. Важно, чтобы каждый дочерний элемент полностью находился внутри тегов родительского элемента, иначе программа не сможет корректно прочитать документ и выдаст сообщение об ошибке. Таким образом, вложение элементов создает иерархическую структуру документа.

## Ресурсы в Android

При разработке Android-приложения, помимо программирования на Java, необходимо работать с ресурсами. В Android принято разделять файлы, такие как изображения, музыка, анимации, стили, макеты и строковые константы — то есть все элементы оформления графического интерфейса — от исходного кода. Большинство ресурсов (за исключением мультимедийных) хранится во внешних XML-файлах, что упрощает их поддержку, обновление и редактирование в процессе разработки.

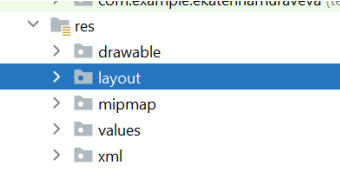
Каждое приложение имеет каталог ресурсов res/, доступ к содержимому которого осуществляется через автоматически генерируемый класс R. Ресурсом может быть как файл (например, изображение), так и значение (например, заголовок приложения), связываемое с приложением по имени ресурса. Это позволяет изменять или заменять ресурсы без необходимости модификации кода или пересборки проекта. Поскольку имена файлов ресурсов используются как имена констант в классе R, они должны соответствовать правилам именования переменных в Java и ограничениям различных операционных систем, то есть состоять только из строчных букв, цифр и символов подчеркивания.

Существует два подхода к созданию ресурсов в Android:  
— Ресурс задается внутри XML-файла, где его имя определяется непосредственно при описании.  
— Ресурс представлен как отдельный файл, и его имя совпадает с именем ресурса.

Для различных типов ресурсов в каталоге res/ создаются соответствующие подкаталоги:

* animator/ — XML-файлы, определяющие анимацию свойств.
* anim/ — XML-файлы, описывающие tween-анимацию.
* color/ — XML-файлы со списками цветов.
* drawable/ — графические файлы (например, .png, .jpg, .gif).
* dimensions/ — XML-файлы, задающие размерности элементов.
* mipmap/ — графические файлы для иконок приложения, адаптированные под различные разрешения экранов.
* layout/ — XML-файлы, определяющие пользовательский интерфейс.
* menu/ — XML-файлы, описывающие меню приложения.
* raw/ — файлы, сохраняемые в исходном виде.
* values/ — XML-файлы, содержащие различные значения, используемые в приложении (например, строки).
* xml/ — произвольные XML-файлы.
* font/ — файлы шрифтов с расширениями .ttf, .otf или .ttc, либо XML-файлы с определением .

Наиболее часто используемыми ресурсами являются разметка (layout), строки (string), цвета (color) и графические изображения (bitmap, drawable). Структура файлов показана на рисунке 22.

  
Рисунок 22 – Папки ресурсов проекта

## Создание интерфейса пользователя при помощи ресурсов

Графический интерфейс формируется с помощью представлений (View) и групп представлений (ViewGroup). Эти элементы размещаются на активности, их описания регистрируются в манифесте, а действия с объектами определяются программно в файле MainActivity.java через методы классов, наследующихся от View и ViewGroup, либо задаются атрибутами в файле разметки layout/activity\_main.xml. Файл разметки также имеет графическое представление (Graphical layout) — системную имитацию мобильного устройства.

В этом файле описаны все графические элементы и их атрибуты, составляющие интерфейс. При создании XML-разметки необходимо соблюдать правило: каждый файл должен содержать один корневой элемент, представляющий объект View или ViewGroup. По умолчанию в проекте с пустой активностью присутствует ресурс разметки activity\_main.xml, где корневым элементом является ConstraintLayout, содержащий элемент TextView, а также определены используемые пространства имен XML.

## Компоненты разметки

Компоненты разметки имеют определенный внешний вид и выполняют конкретные задачи. В Android SDK представлено множество различных компонентов, но рассмотрим основные из них.

**TextView** — это компонент, предназначенный для вывода текста на экран. Он используется для отображения информации, которую не нужно изменять пользователю. Этот элемент поддерживает как статический, так и динамически изменяемый текст, позволяет настраивать шрифт, цвет, размер и выравнивание текста. Однако TextView не предоставляет функционала для редактирования текста пользователем. Основные обязательные настройки компонента включают layout\_width и layout\_height, которые определяют ширину и высоту компонента, например, в значениях match\_parent или wrap\_content.

**EditText** — это подкласс TextView, который предоставляет возможность ввода и редактирования текста пользователем. Этот компонент используется для ввода данных, таких как имя, пароль, электронная почта или другие формы ввода. Он поддерживает различные типы ввода, такие как текст, числа или пароль, и может скрывать символы при вводе пароля. EditText также поддерживает отображение подсказки (hint) для того, чтобы пользователь знал, что нужно ввести. Обязательные настройки компонента включают layout\_width и layout\_height, которые задают ширину и высоту элемента, а также могут быть настроены как match\_parent или wrap\_content.

Button — это компонент, позволяющий пользователю инициировать действия в приложении при нажатии. Он используется для выполнения различных операций, таких как переход между экранами, отправка данных или активация функций. Button предоставляет возможность настраивать внешний вид, включая текст, шрифт, цвет фона и иконки. Обработчик нажатия можно задать как программно, так и через XML-атрибуты. Компонент поддерживает различные анимации и эффекты для улучшения взаимодействия. Обязательные настройки включают layout\_width, layout\_height, onClick и text.

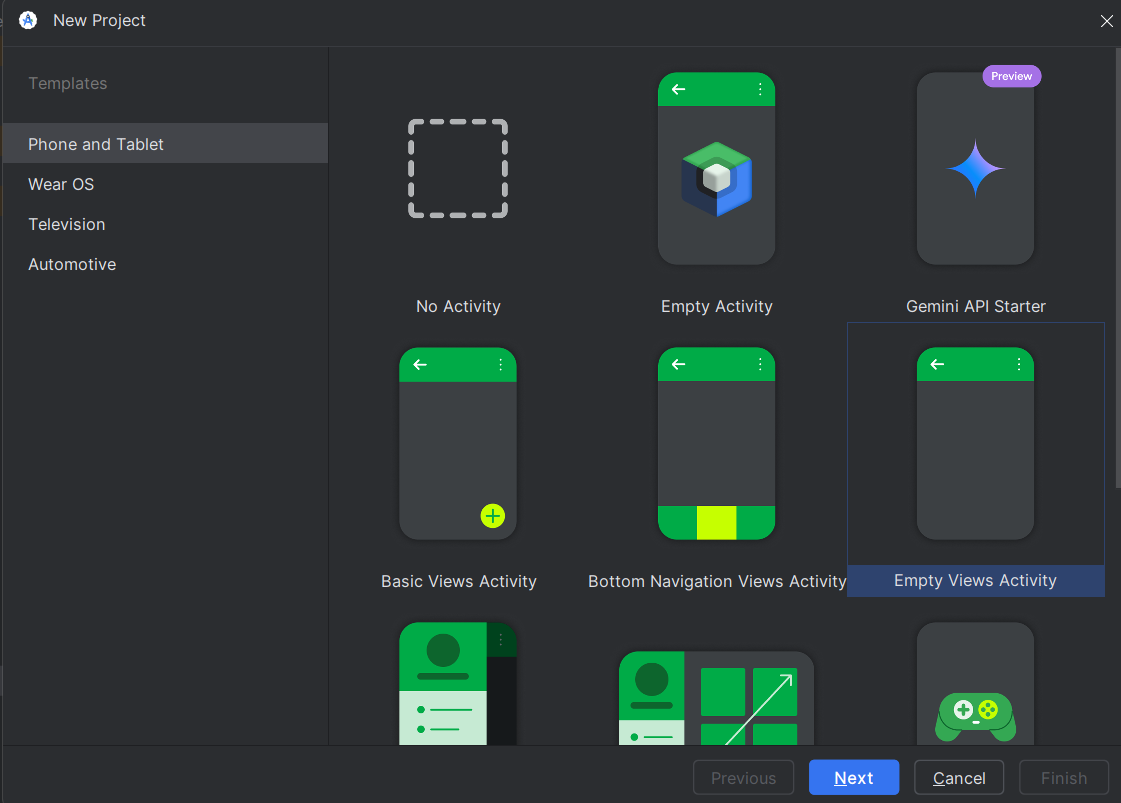
ImageView — это компонент для отображения изображений в приложении. Он используется для работы с графическими ресурсами, такими как фотографии, иконки или изображения, загруженные из интернета. ImageView поддерживает различные способы загрузки изображений, включая локальные ресурсы, URL-адреса и контент-провайдеры. Компонент также предлагает настройку масштабирования изображений с помощью атрибута android:scaleType, который позволяет выбирать режимы центрирования, обрезки или растягивания. Для загрузки изображений из интернета часто используется с библиотеками, такими как Glide или Picasso, которые добавляют функционал кэширования и улучшение производительности. Обязательные настройки включают layout\_width, layout\_height и src (для изображения).

# ПРАКТИЧЕСКАЯ ЧАСТЬ

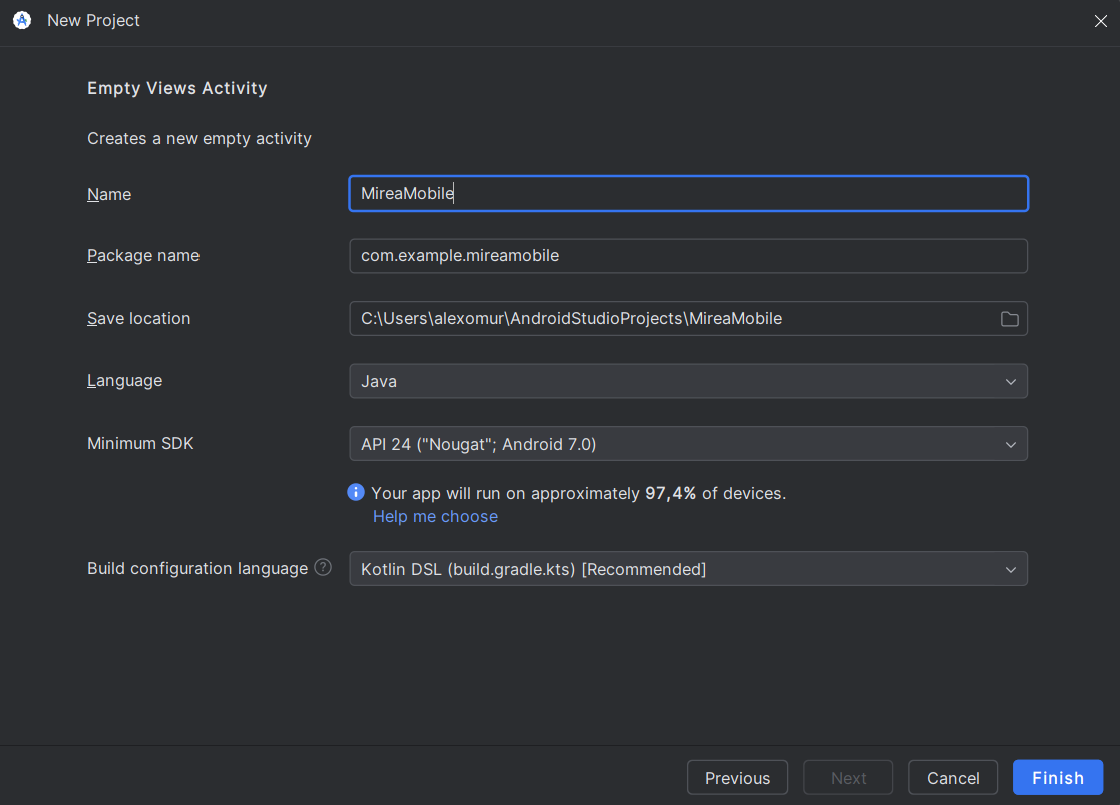


## Создание проекта с ФИО студента в названии

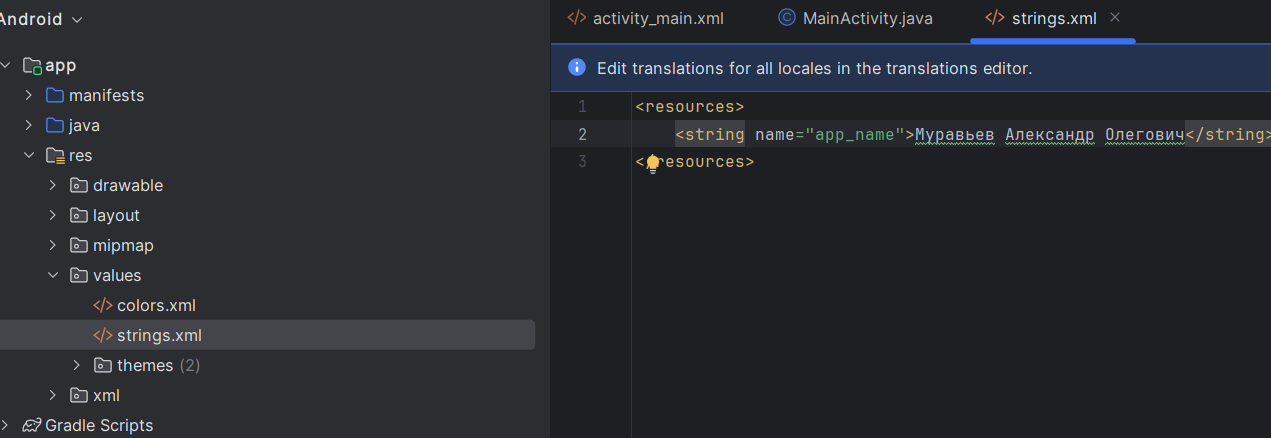
При создании нового проекта выбран шаблон «Empty Views Activity», что показано на рисунке 23.

  
Рисунок 23 – Создание нового проекта, отмечен нужный пустой шаблон

Проект будет называться MireaMobile (название приложения установим позже), Package Name оставляем по умолчанию, проект сохраняем в папке для проектов AndroidStudio, используем язык Java, а версию SDK – 24, язык билда выбираем рекомендованный. Настройки показаны на рисунке 24.

  
Рисунок 24 – Создание нового проекта, настройка названия, версии SDK и других настроек

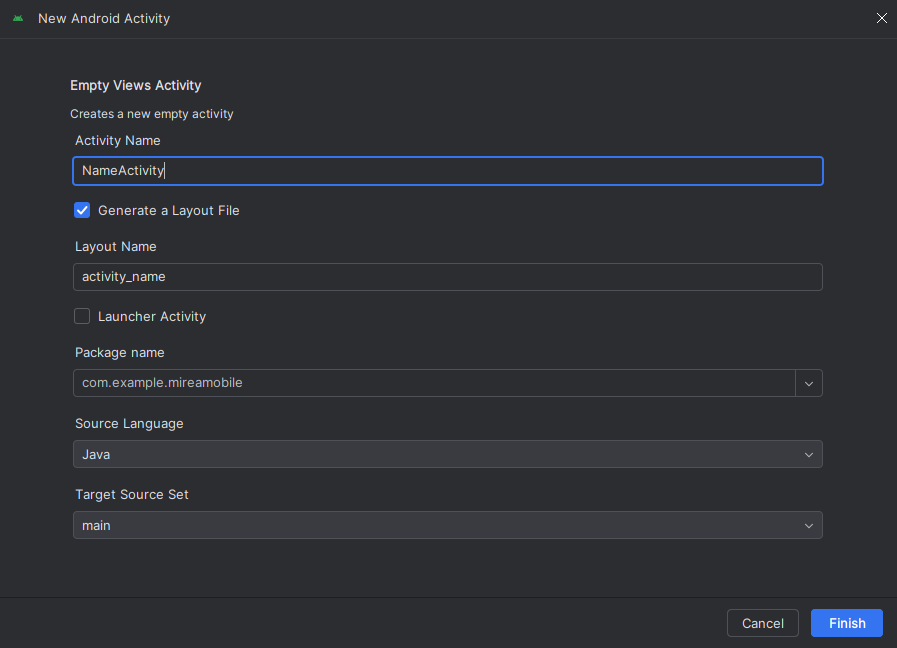
Чтобы изменить название приложения, нужно изменить файл strings.xml. Устанавливаем значение строки «app\_name» на «Муравьев Александр Олегович», как показано на рисунке 25.

  
Рисунок 25 – Настройка строк приложения, название приложения установлено на ФИО студента

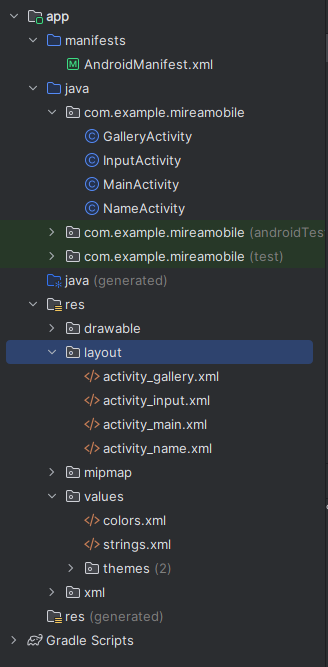
## Разметка

### Текст

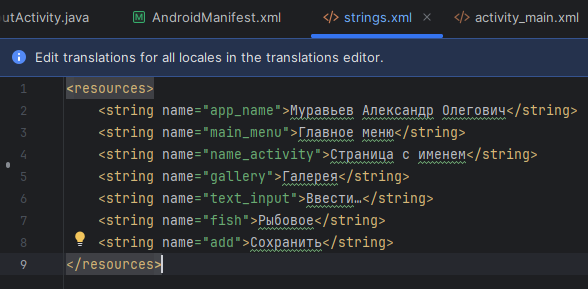
Планируется реализация четырёх страниц: главная страница, страница с именем студента, галерея и страница с вводом. Главная страница создана по умолчанию (MainActivity). Нужно добавить страницу NameActivity с именем студента. Настройки новой страницы показаны на рисунке 26.

  
Рисунок 26 – Создание страницы NameActivity

Таким же образом создаются страницы GalleryActivity (галерея) и InputActivity (страница с вводом). Должна выйти архитектура проекта, как на рисунке 27 (за исключением файлов с меткой «generated»).

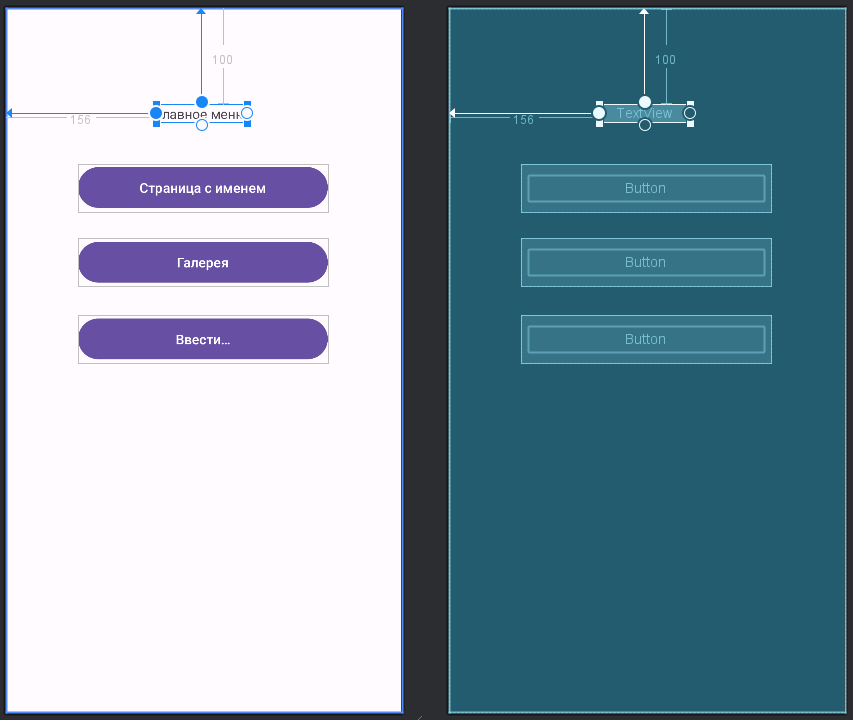
  
Рисунок 27 – Полученная архитектура проекта

Нужно заготовить строки для проекта через настройку файла strings.xml. Переименовать проект, назвав его именем студента, а также установить несколько тегов, которые понадобятся позже. Результат показан на рисунке 28.

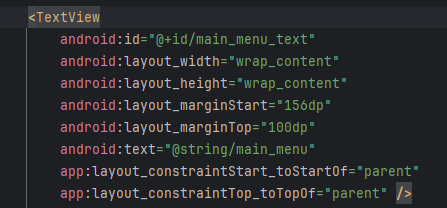
  
Рисунок 28 – Содержимое файла strings.xml

На главной странице (activity\_main.xml) размечен текст «Главное меню».

Чтобы текст мог корректно отображаться на устройствах, компонент TextView прикрепляется к рамкам. На рисунке 29 видно, как расположен компонент текста.

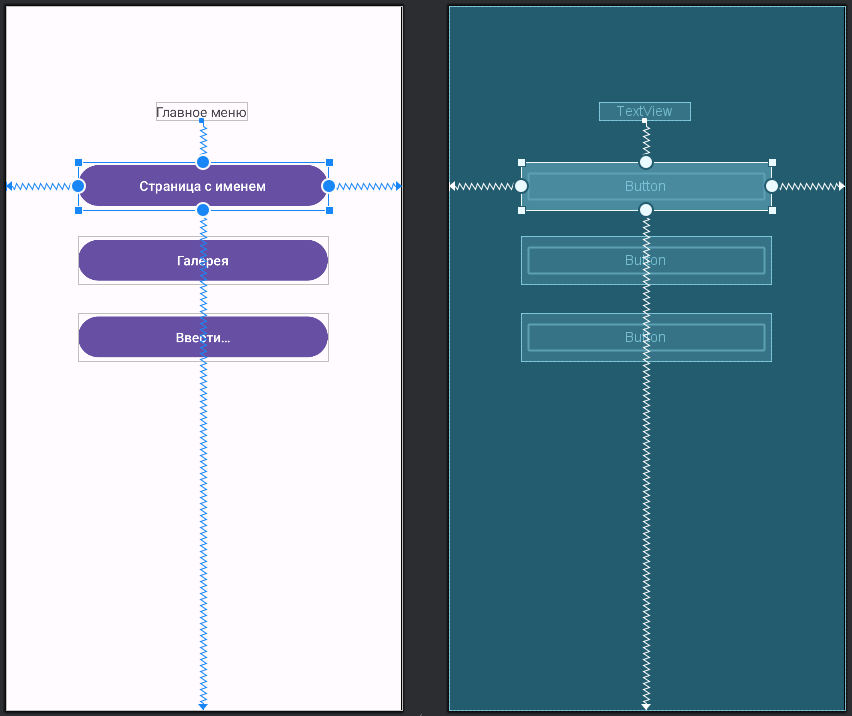
  
Рисунок 29 – Демонстрация крепления компонента TextView

Чтобы указать текст внутри элемента, была использована нотация «@string», которая позволяет хранить любые строки в одном файле, что было прописано ранее. Чтобы использовать нужную заготовленную строку, нужно в атрибуте android:text указать строку, построенную по схеме «@string/название\_строки». Текст «Главное меню» был сохранён под названием «main\_menu». То есть для отображения этого текста необходимо в поле text указать «@string/main\_menu». Кодовый формат компонента TextView показан на рисунке 30.

  
Рисунок 30 – XML-код разметки текста на странице activity\_main.xml

### Кнопка

Для реализации кнопки был добавлен компонент Button. Для корректного отображения кнопки на устройствах этот компонент также нуждается в прикреплении, что показано на рисунке 31.

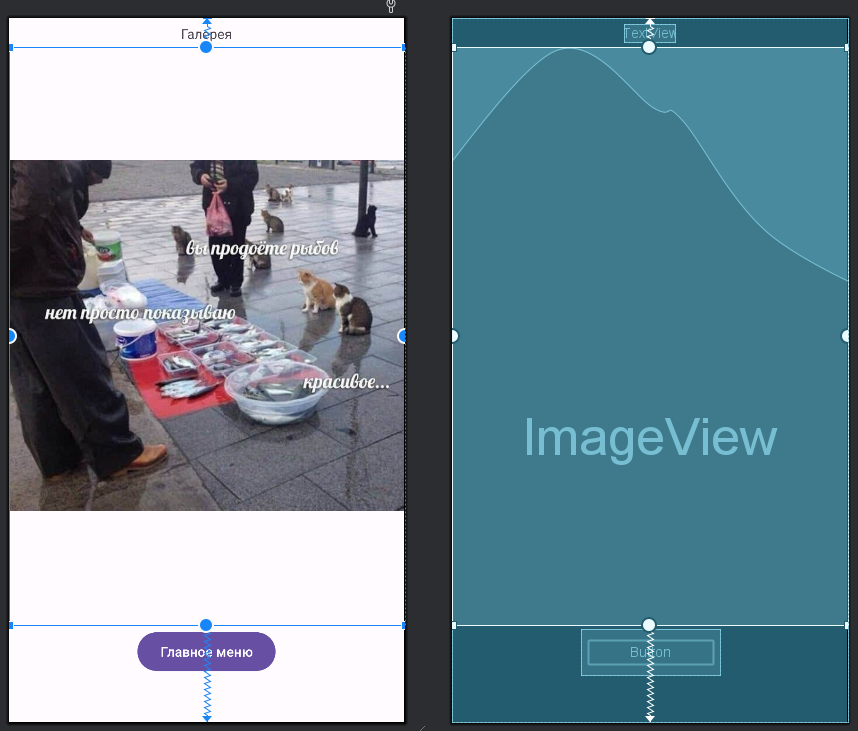
  
Рисунок 31 – Демонстрация крепления компонента Button в файле activity\_main.xml

Здесь также используется нотация «@string». Для отображения текста «Страница с именем» нужно ввести «@string/name\_activity». Кодовый формат файла показан на рисунке 32.

  
Рисунок 32 – XML-код разметки кнопки в файле activity\_main.xml

### Картинка

Для реализации компонента картинки создан отдельный файл activity\_gallery.xml. Как и другие элементы, ImageView требует привязки, в данном случае — к границам экрана, чтобы корректно отображаться. На рисунке 33 показано его расположение в макете.

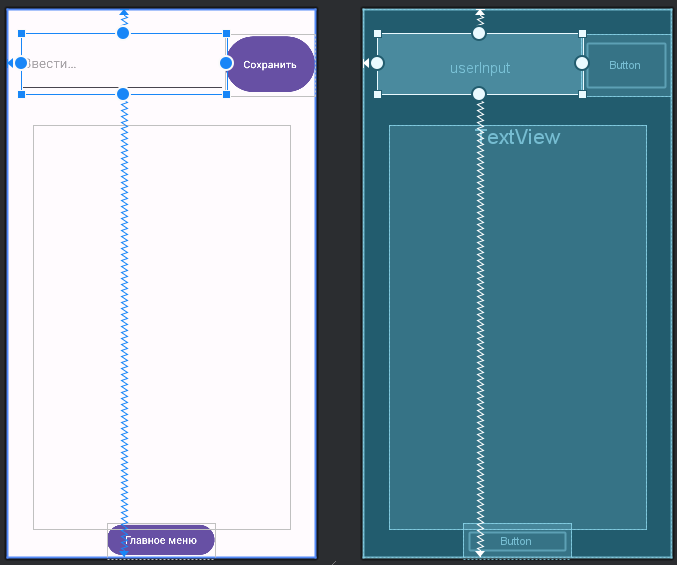
  
Рисунок 33 – Демонстрация крепления компонента ImageView в файле activity\_image.xml

Кодовый формат компонента картинки показан на рисунке 34.

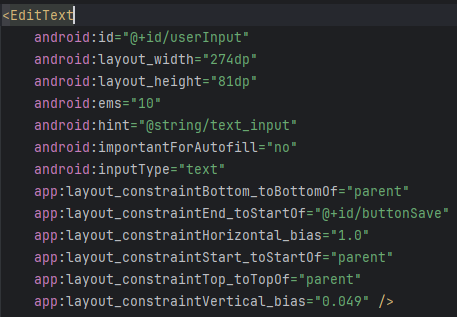
  
Рисунок 34 – XML-код разметки картинки в файле activity\_gallery.xml

### Поле ввода

Для того чтобы реализовать поле ввода, был создан файл макета под названием activity\_input.xml. Однако, чтобы кнопка корректно отображалась на различных устройствах, необходимо дополнительно прикрепить компонент TextInputEditText. Крепления показаны на рисунке 35.

  
Рисунок 35 – Демонстрация крепления компонента TextInputEditText в файле activity\_input.xml

Кодовый формат поля ввода показан на рисунке 36.

  
Рисунок 36 – XML-код разметки поля ввода

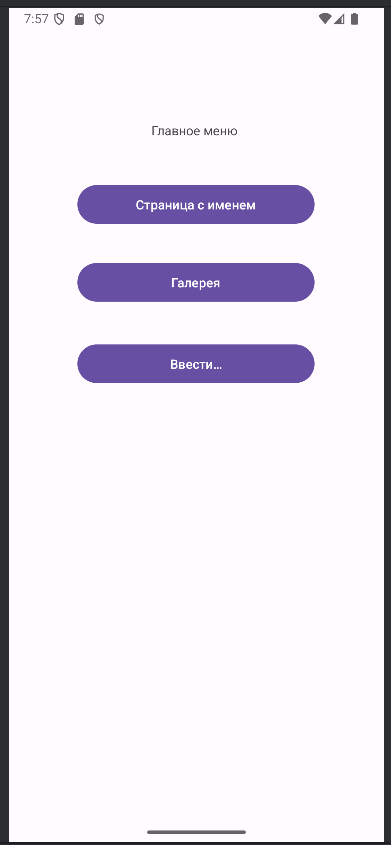
ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы были получены навыки создания мобильного проекта в среде разработки Android Studio, а также работы с файлами разметки XML. Были освоены основы проектирования пользовательского интерфейса, размещения и настраивания базовых компонентов, таких как текстовые поля (TextView), кнопки (Button), поля ввода (EditText) и изображения (ImageView).

Кроме того, была изучена работа с различными атрибутами UI-элементов и принципами их компоновки с использованием ConstraintLayout, что позволяет гибко управлять расположением элементов на экране.

Таким образом, в рамках данной работы были закреплены базовые знания по разработке интерфейсов в Android-приложениях.

В результате практической работы было создано приложение, продемонстрированное на рисунке 37.

  
Рисунок 37 – Запущенное на эмуляторе приложение