

Лабораторная работа № 2

«Создание первого Android-приложения.

Использование различных классов макетов»

Продолжительность выполнения лабораторной работы: 2 ак. часа.

Целью лабораторной работы является знакомство со средой разработки Android Studio, создание первого Android-приложения, получение навыков создания различных классов компоновок пользовательского интерфейса.

Теоретическая часть

Создание Android-приложения

Для создания Android-приложения выполните следующие действия.

1. Запустите Android Studio. В приветственном окне выберите пункт New Project (Новый проект) или выберите команду меню File → New → New Project (Файл → Новый → Новый проект).
2. В следующем окне (рис. 1) необходимо выбрать шаблон активности, который будет создан по умолчанию. В лабораторных работах будет использоваться шаблон Empty Activity. **В Android Studio 2023 выбирайте пункт Empty Views Activity.**

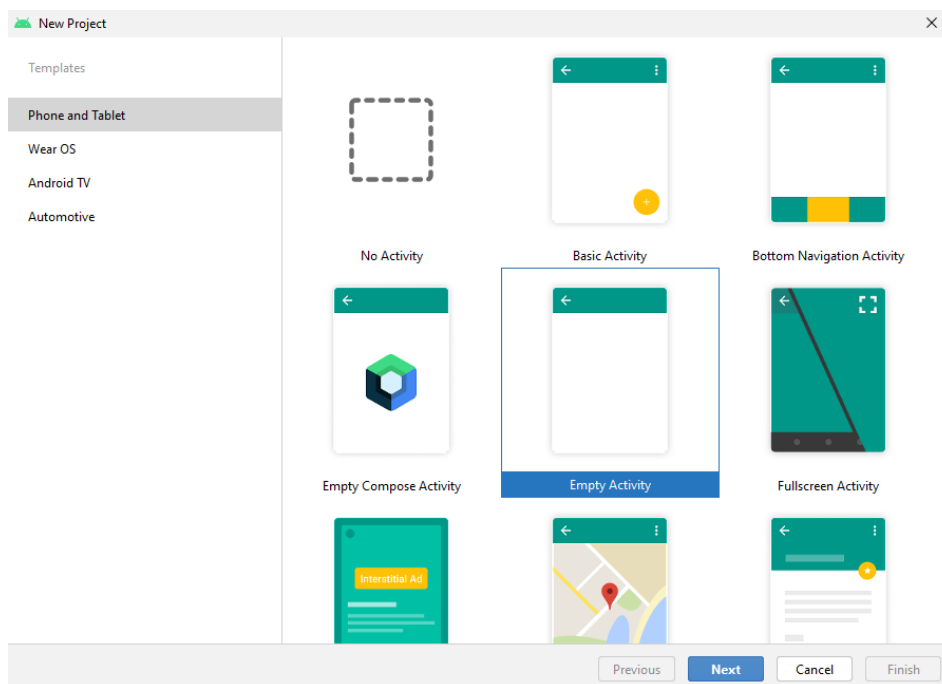
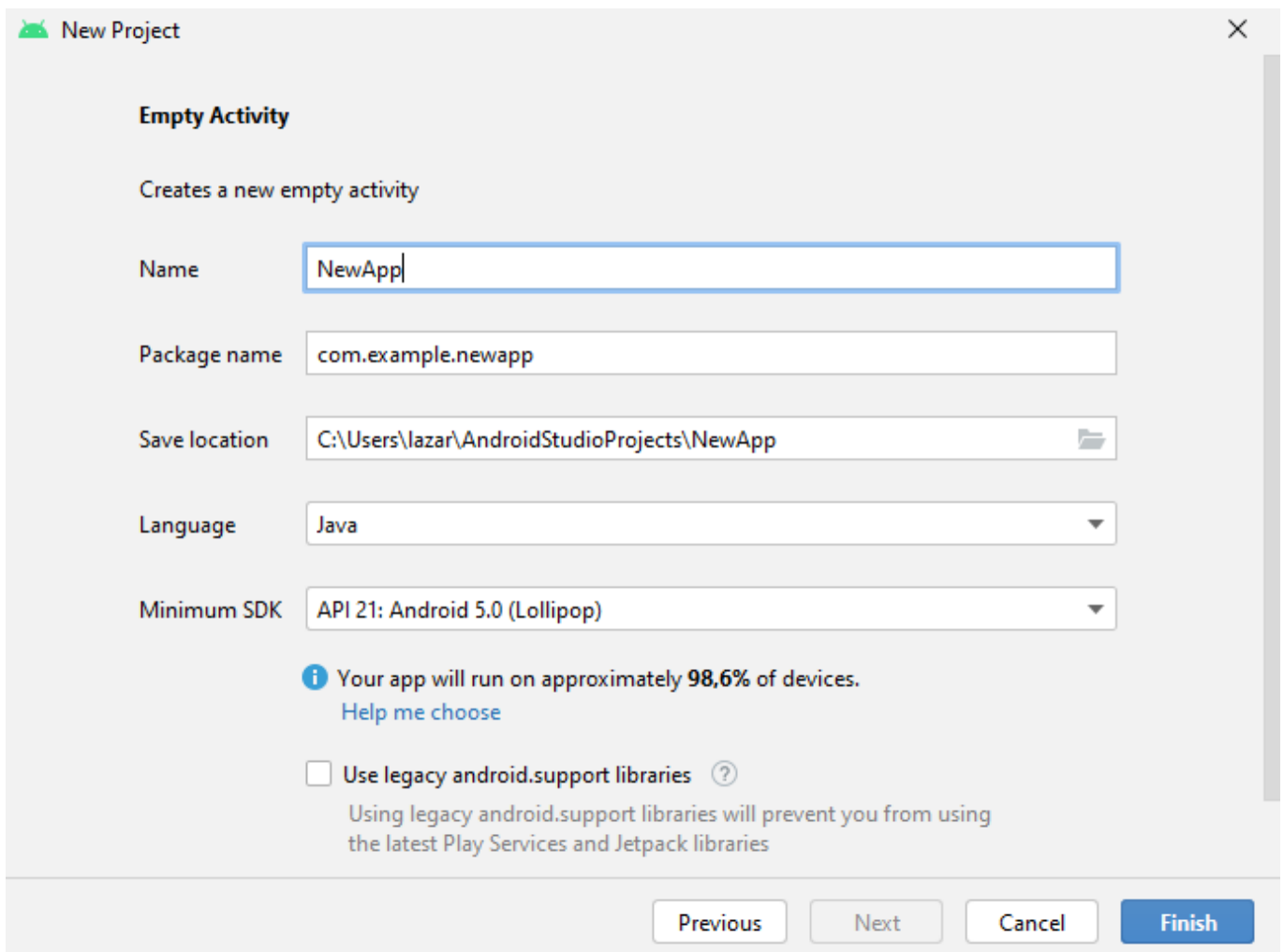


Рисунок 1. Окно New Project.

3. Укажите имя приложения (Name). Это имя будут видеть пользователи. Введите имя пакета (Package name), удовлетворяющее требованиям пространства имен стандартных пакетов Java. Укажите путь размещения папки проекта (Save location). Выберите язык разработки (Language). Выберите минимальную версию платформы для приложения (Minimum SDK). Чем выше минимальная SDK (API), тем на меньшем проценте устройств будет работать ваше приложение. Для завершения создания проекта нажмите кнопку Finish (рис. 2).



New Project

Empty Activity

Creates a new empty activity

Name:

Package name:

Save location:

Language:

Minimum SDK:

i Your app will run on approximately **98,6%** of devices.
[Help me choose](#)

☐ Use legacy android.support libraries **?**
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

Рисунок 2. Окно New Project (Empty Activity).

Управление виртуальными устройствами Android

Чтобы запустить приложение в эмуляторе Android, сначала вы должны сконфигурировать Android Virtual Device (AVD). Профиль AVD описывает тип устройства, которое вы хотите имитировать в эмуляторе, включая версию поддерживаемой платформы. Вы можете указать различные размеры экранов и ориентации, а также имеет ли эмулятор SD-карту и, если она есть, ее емкость и т.д. Ниже приведена пошаговая инструкция для создания типичного AVD.

1. Нажмите на кнопку Device Manager (рис. 4) или откройте соответствующую вкладку.

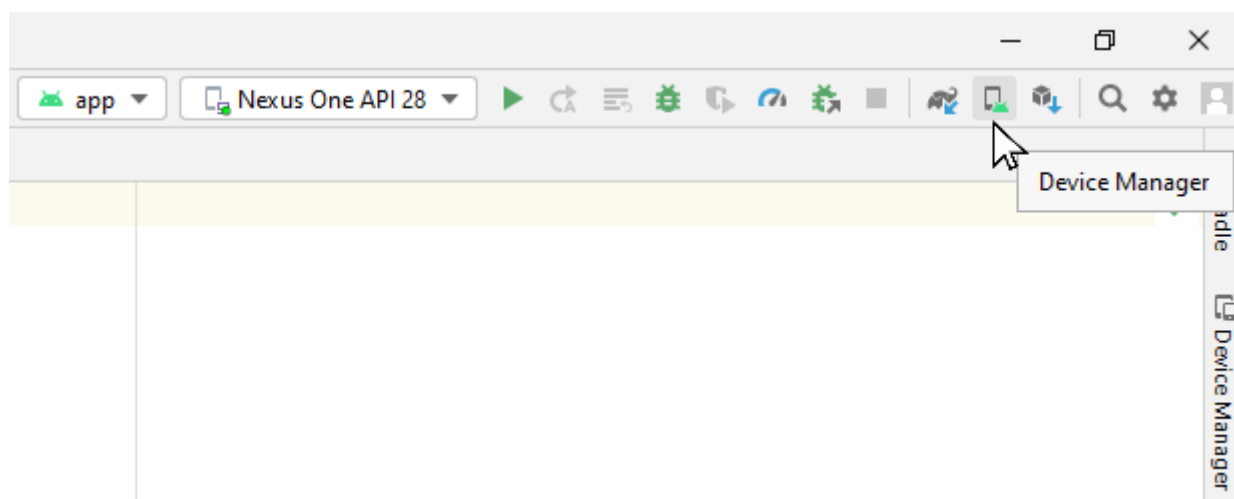


Рисунок 3. Кнопка Device Manager.

2. На появившейся панели (рис. 4) нажмите на кнопку Create device (Создать устройство).

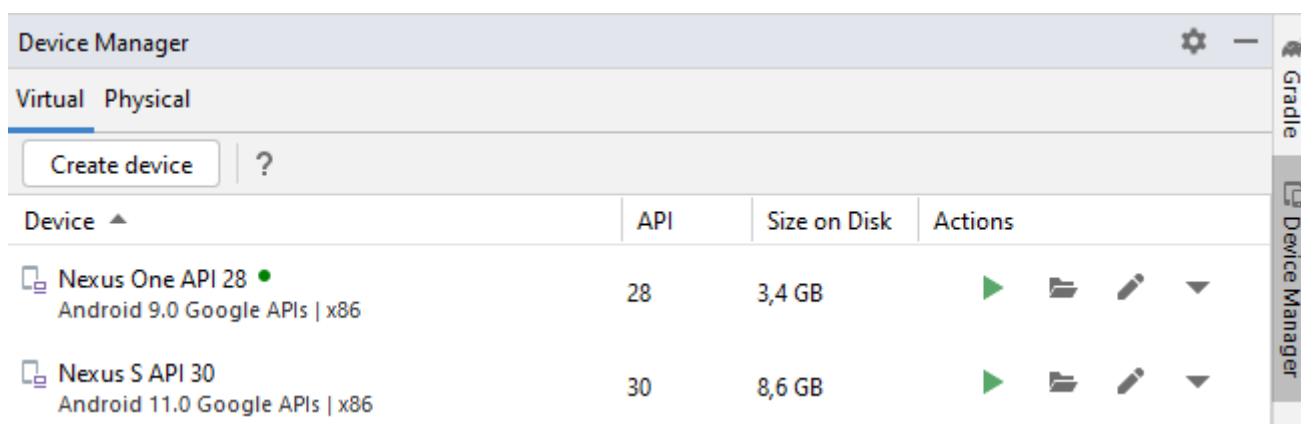


Рисунок 4. Панель Device Manager

3. Выберите тип устройства и размеры экрана (рис. 5). Чем больше будет размер экрана, тем медленнее будет загружаться виртуальное устройство.

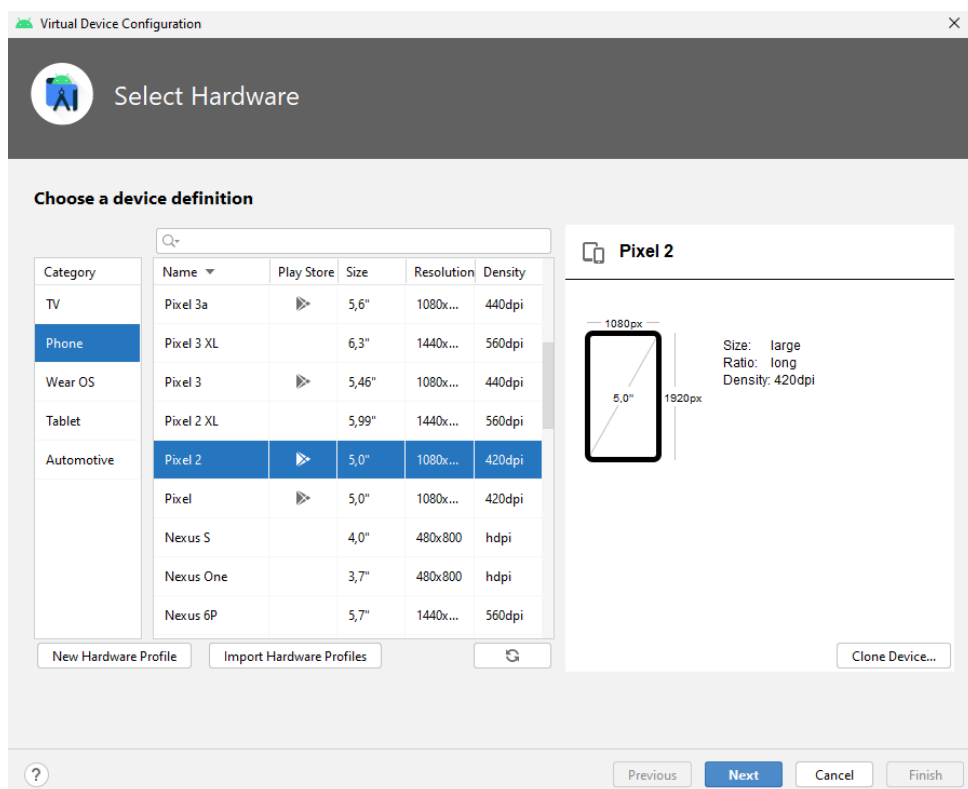


Рисунок 5. Выбор устройства.

- Выберите версию ОС Android, которая будет установлена на виртуальное устройство (рис. 6). Если нужная версия не загружена, её можно скачать, кликнув по ссылке Download рядом с её именем.

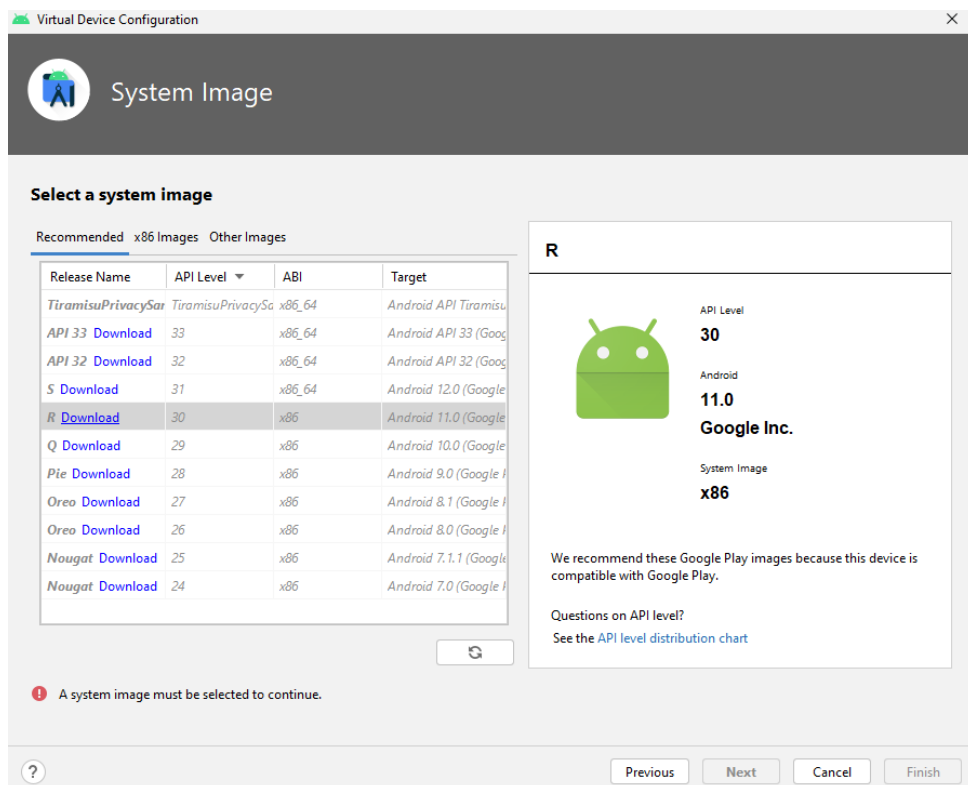


Рисунок 6. Выбор версии.

5. Введите имя виртуального устройства в поле AVD Name.

Пункт Startup orientation позволяет задать ориентацию экрана по умолчанию.

Пункт Emulated Performance позволяет задать, как будет рендериться графика.

Пункт Device Frame позволяет включить отображение рамки, имитирующей реальное мобильное устройство.

С помощью кнопки Show Advanced Settings можно включить отображение пунктов настройки других параметров (рис. 7).

Пункты Camera и Network позволяют задать настройки эмуляции камер (фронтальной и задней) и сетевого соединения соответственно.

В пункте Memory and Storage можно настроить параметры эмуляции памяти: RAM – объём оперативной памяти, VM Heap — размер кучи¹ виртуальной машины. Выберите размер внутренней памяти устройства в пункте Internal Storage и емкость SD-карты. Образ этой SD-карты будет занимать пространство на жестком диске вашего компьютера, поэтому не задавайте слишком большой объем, желательно не больше 1024 Мбайт.

Пункт Keyboard позволяет включить использование клавиатуры компьютера для ввода данных в виртуальном устройстве. Если галочка не отмечена, будет эмулироваться виртуальная клавиатура.

Нажмите на кнопку Finish. Создание AVD может занять некоторое время.

¹ Куча— структура данных, с помощью которой реализована динамически распределяемая память приложения. Куча использует память, выделяемую динамически или запрошенную статически у операционной системы. Эта память используется для размещения объектов, динамически созданных программой.

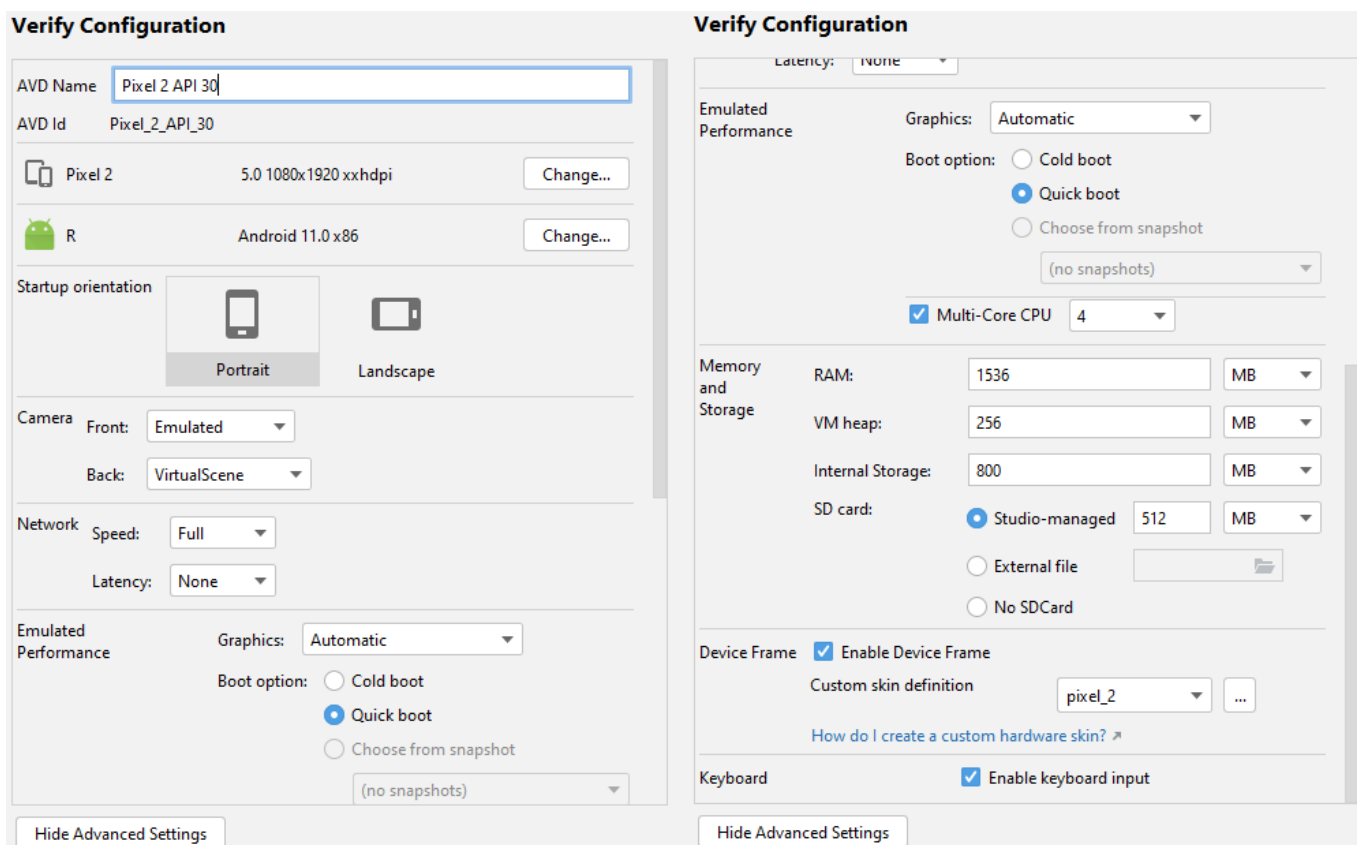


Рисунок 7. Настройки виртуального устройства.

6. Выберите виртуальное или реальное Android-устройство, подключенное к компьютеру, для запуска приложения. Далее для запуска приложения нажмите кнопку Run 'app' (рис. 8).

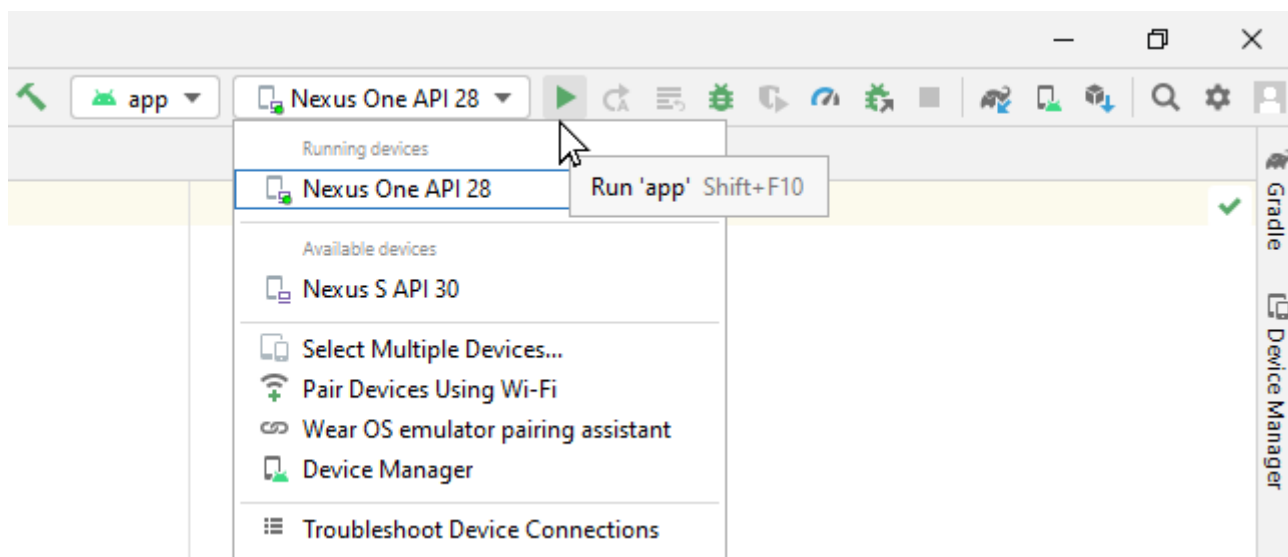


Рисунок 8. Запуск приложения.

Компоновка пользовательского интерфейса

Для описания архитектуры расположения элементов пользовательского интерфейса конкретного активити используются компоновки. **Компоновка (макет, разметка)** может быть объявлена двумя способами: в специальном xml-файле или в программном коде активити путем создания объектов классов View и ViewGroup. Эти два способа можно совмещать, например, объявить элементы интерфейса в xml-файле и управлять их свойствами в программном коде.

Объявление элементов интерфейса в xml-файле позволяет отделить представление (англ. view) приложения от кода, управляющего поведением приложения (англ. controller), в соответствии с парадигмой MVC (англ. Model-View-Controller, Модель-Представление-Контроллер). Отделение представления от контроллера позволяет изменять интерфейс активити без внесения исправлений в исходный код и его повторной компиляции. Также есть возможность создавать разные xml-файлы компоновок для разных размеров и ориентаций экрана, для разных языков и т.д. Помимо этого, использование xml-компоновок упрощает визуализацию структуры интерфейса пользователя и упрощает его отладку.

Для всех элементов пользовательского интерфейса (например, кнопок, текстовых полей, переключателей) базовым классом является **класс View** (представление). Для коллекции объектов View базовым является **класс ViewGroup** (группа представлений). Объект ViewGroup является контейнером, который может содержать объекты View и ViewGroup. Класс ViewGroup также является базовым для подклассов компоновок, например, линейной компоновки (англ. LinearLayout).

В xml-файле компоновки должен быть равно **один корневой элемент**, который должен быть потомком класса View или ViewGroup. Чаще всего в качестве корневого элемента используется один из стандартных классов компоновок: LinearLayout (линейная компоновка), RelativeLayout (относительная компоновка), ConstraintLayout (компоновка на основе ограничений), WebView (представление веб-страницы), AdapterView (адаптивное, заполняемое во время выполнения представление).

XML-файл компоновки приложения хранится в каталоге res/layout/ проекта Android (рис. 9). Для назначения компоновки активити необходимо в методе

обратного вызова `onCreate ()` для данного активити вызвать метод `setContentView ()` и передать ему в качестве параметра ссылку на ресурс компоновки в следующем виде: `R.layout.layout_name`, где `layout_name` – имя файла компоновки без расширения `.xml` (листинг 1).

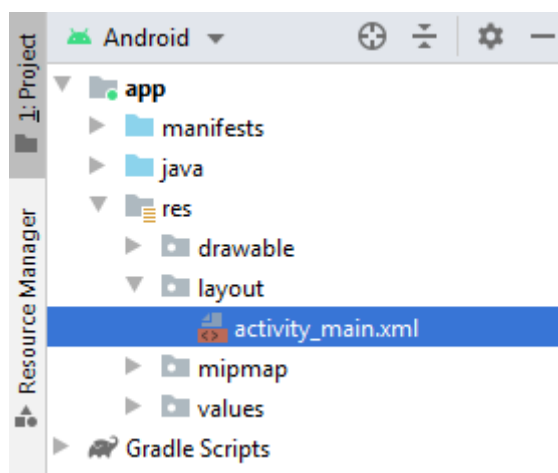


Рисунок 9. Расположение XML-файла компоновки.

Листинг 1. MainActivity.java

```
package com.example.myapplication;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Линейная компоновка

При использовании линейной компоновки дочерние элементы, находящиеся внутри корневого элемента **LinearLayout**, располагаются либо в ряд по горизонтали, либо в столбец по вертикали, в зависимости от параметра `android:orientation`.

При формировании интерфейса пользователя компоновка может содержать ещё одну или несколько вложенных компоновок. Например, внутри линейной компоновки с вертикальной ориентацией могут быть вложены компоновки с горизонтальной ориентацией. Однако не рекомендуется использовать сложную иерархию компонентов, так как это замедляет отрисовку интерфейса.

Рассмотрим пример (листинг 2).

Листинг 2. Компоновка activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Button 1" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/button2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button 2" />
        <Button
            android:id="@+id/button3"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button 3" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:id="@+id/button4"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button 4" />
        <Button
            android:id="@+id/button5"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button 5" />
        <Button
            android:id="@+id/button6"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Button 6" />
    </LinearLayout>
</LinearLayout>
```

В данном примере внутри компоновки LinearLayout с вертикальной ориентацией располагается кнопка и два вложенных элемента LinearLayout с горизонтальной ориентацией, в которых в свою очередь помещены две и три кнопки соответственно (рис 10).

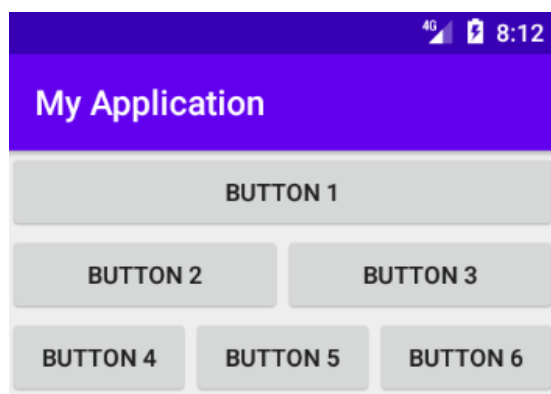


Рисунок 10. Линейная компоновка.

Рассмотрим атрибуты, использованные в данной компоновке (табл. 1).

Таблица 1. Атрибуты элементов в xml-компоновке.

Атрибут	Назначение атрибута	Пример значения атрибута	Описание значения атрибута
android:orientation	Задаёт расположение элементов внутри компоновки по горизонтали или по вертикали	vertical	Расположение элементов по вертикали
		horizontal	Расположение элементов по горизонтали
android:layout_width	Задаёт ширину элемента	match_parent	Ширина по размеру родительского контейнера
		wrap_content	Ширина по размеру содержимого
		100dp	Ширина в 100 плотностно-независимых пикселей
android:layout_height	Задаёт высоту элемента	match_parent	Высота по размеру родительского контейнера
		wrap_content	Высота по размеру содержимого
		100dp	Высота в 100 плотностно-независимых пикселей
android:layout_weight	Назначает индивидуальный вес для дочернего элемента	1	Вес равен единице
android:id	Задаёт уникальный идентификатор	@+id/new_id	Создание идентификатора new_id
android:text	Задаёт текст, отображаемый в элементе	Текст	Фиксированный текст (англ. hardcoded text)
		@string/name	Ссылка на строковый ресурс с именем name

Атрибут `android:layout_weight` назначает индивидуальный вес для дочернего элемента. Этот атрибут определяет "важность" объекта View и позволяет этому элементу расширяться, чтобы заполнить любое оставшееся пространство в родительском объекте View. Заданный по умолчанию вес является нулевым.

Например, если есть три кнопки, и двум из них объявлен вес со значением 1, в то время как третьей не дается никакого веса (0), третья кнопка без веса не будет расширяться и займет область, определяемую размером текста, отображаемого на ней. Другие две расширятся одинаково, чтобы заполнить остаток пространства, не занятого третьей кнопкой. Если третьей кнопке присвоить вес 2 (вместо 0), она будет объявлена как "более важная", чем две других, так что третья кнопка получит 50% общего пространства, в то время как первые две получают по 25% общего пространства.

В атрибутах **`android:layout_width`** и **`android:layout_height`** для задания ширины могут использоваться следующие единицы измерения:

- px (англ. pixels) — пиксели;
- dp (англ. density-independent pixels) — независимые от плотности пиксели;
- sp (англ. scale-independent pixels) — независимые от масштабирования пиксели;
- in (англ. inches) — дюймы;
- pt (англ. points) — 1/72 дюйма;
- mm (англ. millimeters) — миллиметры.

Относительная компоновка

При использовании относительной компоновки **`RelativeLayout`** расположение дочерних объектов задаётся относительно родительского объекта или относительно соседних дочерних элементов (по идентификатору элемента).

В **`RelativeLayout`** дочерние элементы расположены так, что, если первый элемент расположен по центру экрана, другие элементы, выровненные относительно первого элемента, будут выровнены относительно центра экрана. При таком расположении, при объявлении компоновки в XML-файле, элемент, на который будут ссылаться для

позиционирования другие объекты, должен быть объявлен раньше, чем элементы, которые обращаются к нему по его идентификатору.

Для создания **идентификатора** используется атрибут **android:id**. В качестве значения атрибута должно быть указано сочетание символов `@+id/` и уникальное имя идентификатора, например, атрибут `android:id="@+id/button1"` создаёт идентификатор элемента с именем `button1`.

Если в программном коде мы не работаем с некоторыми элементами пользовательского интерфейса, создавать идентификаторы для них необязательно, однако определение идентификаторов для объектов важно при создании `RelativeLayout`. В компоновке `RelativeLayout` расположение элемента `View` (представления) может определяться относительно другого элемента (таблица 2), на который ссылаются через его уникальный идентификатор:

`android:layout_toLeftOf="@id/TextView1"`

Обратите внимание, что атрибуты, которые обращаются к идентификаторам относительных элементов (например, `layout_toLeftOf`), используют синтаксис относительного ресурса `@id/id`.

Таблица 2. Атрибуты, использующиеся для определения расположения элемента в `RelativeLayout`.

Атрибут	Описание
<code>android:layout_above</code>	Располагает нижний край данного представления над заданным по <code>id</code> представлением
<code>android:layout_alignBaseline</code>	Помещает базовую линию представления на базовую линию заданного представления
<code>android:layout_alignBottom</code>	Делает нижний край представления совпадающим с нижним краем заданного представления
<code>android:layout_alignEnd</code>	Делает конечный край представления совпадающим с конечным краем заданного представления
<code>android:layout_alignLeft</code>	Делает левый край представления совпадающим с левым краем заданного представления
<code>android:layout_alignParentBottom</code>	Если <code>true</code> , то нижний край представления совпадает с нижним краем родителя
<code>android:layout_alignParentEnd</code>	Если <code>true</code> , то конечный край представления совпадает с конечным краем родителя

Продолжение таблицы 2.

android:layout_alignParentLeft	Если true, то левый край представления совпадает с левым краем родителя
android:layout_alignParentRight	Если true, то правый край представления совпадает с правым краем родителя
android:layout_alignParentStart	Если true, то начальный край представления совпадает с начальным краем родителя
android:layout_alignParentTop	Если true, то верхний край представления совпадает с верхним краем родителя
android:layout_alignRight	Делает правый край представления совпадающим с правым краем заданного представления
android:layout_alignStart	Делает начальный край представления совпадающим с начальным краем заданного представления
android:layout_alignTop	Делает верхний край представления совпадающим с верхним краем заданного представления
android:layout_alignWithParentIfMissing	Если установлено значение true, родительский элемент будет использоваться в качестве привязки, когда привязка не может быть найдена для <code>layout_toLeftOf</code> , <code>layout_toRightOf</code> и т.д.
android:layout_below	Располагает верхний край представления под заданным представлением
android:layout_centerHorizontal	Если true, центрирует дочерний элемент по горизонтали внутри родителя
android:layout_centerInParent	Если true, центрирует дочерний элемент по горизонтали и вертикали внутри родителя
android:layout_centerVertical	Если true, центрирует дочерний элемент по вертикали внутри родителя
android:layout_toEndOf	Помещает начальный край представления в конец заданного представления
android:layout_toLeftOf	Позиционирует правый край представления слева от заданного представления
android:layout_toRightOf	Позиционирует левый край представления справа от заданного представления
android:layout_toStartOf	Позиционирует конечный край представления в начало заданного представления

Следует уточнить, что понимается под начальным и конечным краями представления. Для большинства языков начальным краем является левый, а конечным краем — правый. При направлении письма справа налево (англ. RTL — Right-to-left), например, в арабском языке, иврите, начальным краем является правый, а конечным краем — левый.

При использовании `RelativeLayout` расположение элементов зависит от разрешения и ориентации экрана мобильного устройства. Если вы будете использовать `RelativeLayout` в собственных приложениях, всегда проверяйте внешний вид окна для различных разрешений и ориентации экрана, поскольку возможно наложение элементов друг на друга.

Макет на основе ограничений

Макет или компоновка на основе ограничений (`ConstraintLayout`) стал доступен для устройств начиная с уровня Android API 9. Он позволяет создавать большие и сложные компоновки с плоской иерархией представлений (без вложенных групп представлений). **`ConstraintLayout`** похож на `RelativeLayout` в том, что для всех вложенных элементов задаётся их размещение относительно друг друга или родительского контейнера, но он более гибкий и имеет лучшую производительность, чем `RelativeLayout`, и его проще использовать с **визуальным редактором макетов** в Android Studio. С его помощью можно создать компоновку целиком путем перетаскивания объектов в режиме Design вместо редактирования xml-файла компоновки (рис. 11).

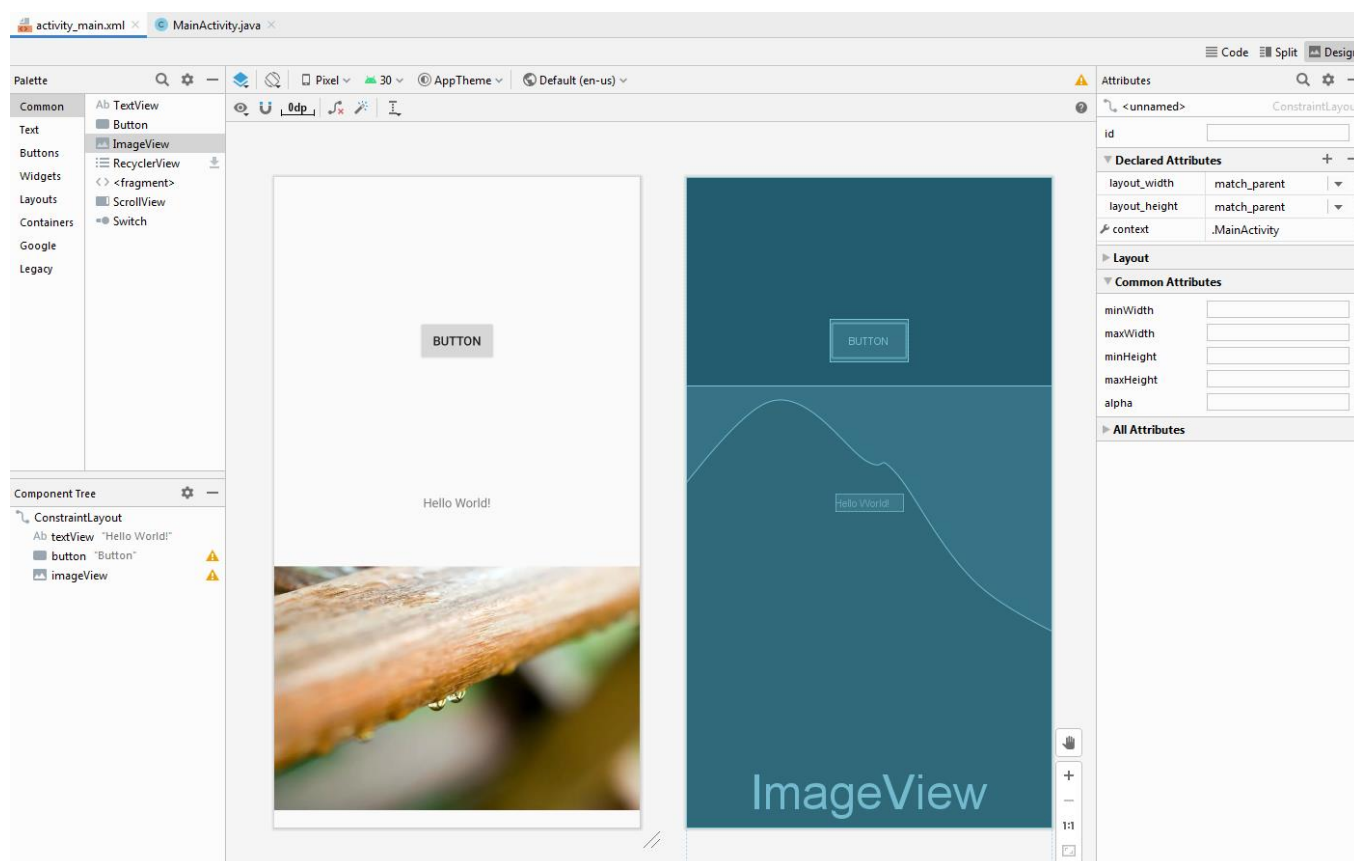


Рисунок 11. Визуальный редактор макетов в Android Studio.

Для переключения между режимом визуального редактирования компоновки и режимом редактирования xml-кода используются кнопки Code, Split и Design в верхнем правом углу редактора.

В таблице 3 приведено соответствие атрибутов элементов RelativeLayout и ConstraintLayout, задающих одинаковое положение элементов.

Таблица 3. Соответствие атрибутов элементов RelativeLayout и ConstraintLayout.

Атрибут RelativeLayout	Атрибут (атрибуты) ConstraintLayout
android:layout_above="@id/view"	app:layout_constraintBottom_toTopOf="@id/view"
android:layout_alignBaseline="@id/view"	app:layout_constraintBaseline_toBaselineOf="@id/view"
android:layout_alignBottom="@id/view"	app:layout_constraintBottom_toBottomOf="@id/view"
android:layout_alignEnd="@id/view"	app:layout_constraintEnd_toEndOf="@id/view"
android:layout_alignLeft="@id/view"	app:layout_constraintLeft_toLeftOf="@id/view"
android:layout_alignParentBottom="true"	app:layout_constraintBottom_toBottomOf="parent"
android:layout_alignParentEnd="true"	app:layout_constraintEnd_toEndOf="parent"
android:layout_alignParentLeft="true"	app:layout_constraintLeft_toLeftOf="parent"
android:layout_alignParentRight="true"	app:layout_constraintRight_toRightOf="parent"
android:layout_alignParentStart="true"	app:layout_constraintStart_toStartOf="parent"
android:layout_alignParentTop="true"	app:layout_constraintTop_toTopOf="parent"
android:layout_alignRight="@id/view"	app:layout_constraintRight_toRightOf="@id/view"
android:layout_alignStart="@id/view"	app:layout_constraintStart_toStartOf="@id/view"
android:layout_alignTop="@id/view"	android:layout_alignTop="@id/view"
android:layout_below="@id/view"	app:layout_constraintTop_toBottomOf="@id/view"
android:layout_centerHorizontal="true"	app:layout_constraintLeft_toLeftOf="parent" app:layout_constraintStart_toStartOf="parent" app:layout_constraintRight_toRightOf="parent" app:layout_constraintEnd_toEndOf="parent"
android:layout_centerInParent="true"	app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintLeft_toLeftOf="parent" app:layout_constraintStart_toStartOf="parent" app:layout_constraintRight_toRightOf="parent" app:layout_constraintEnd_toEndOf="parent" app:layout_constraintTop_toTopOf="parent"
android:layout_centerVertical="true"	app:layout_constraintBottom_toBottomOf="parent" app:layout_constraintTop_toTopOf="parent"
android:layout_toEndOf="@id/view"	app:layout_constraintStart_toEndOf="@id/view"
android:layout_toLeftOf="@id/view"	app:layout_constraintRight_toLeftOf="@id/view"
android:layout_toRightOf="@id/view"	app:layout_constraintLeft_toRightOf="@id/view"
android:layout_toStartOf="@id/view"	app:layout_constraintEnd_toStartOf="@id/view"

Задание

Создайте два Android-приложения с использованием двух различных типов компоновок, например, `LinearLayout` и `RelativeLayout` или `ConstraintLayout`. Разместите в этих компоновках по 7-10 различных представлений (например, `Button`, `TextView`, `ImageView`).

Контрольные вопросы:

1. Что такое компоновка?
2. Какие типы компоновок Вы знаете?
3. В чём особенности компоновки `LinearLayout`?
4. В чём особенности компоновки `ConstraintLayout`?
5. В чём особенности компоновки `RelativeLayout`?
6. За что отвечает атрибут `android:layout_height`?
7. Что означает значение ширины или высоты `match_parent`?
8. Что означает значение ширины или высоты `wrap_content`?
9. За что отвечает атрибут `android:gravity`?
10. За что отвечает атрибут `android:id`?
11. Что обозначает запись `@+id` в атрибуте `android:id`?
12. Какие единицы измерения используются на платформе Android для указания размеров представлений?