

МОБИЛЬНЫЕ И WEB-ПРИЛОЖЕНИЯ

Android studio

Преподаватели:

Казакова Анастасия Семеновна

Ассистент кафедры информатики

ОС Android

Android - операционная система для мобильных устройств: смартфонов, планшетных компьютеров, КПК. В настоящее время именно Android является самой широко используемой операционной системой для мобильных устройств.

Платформа Android объединяет операционную систему, построенную на основе ядра ОС Linux, промежуточное программное обеспечение и встроенные мобильные приложения. Разработка и развитие мобильной платформы Android выполняется в рамках проекта AOSP (Android Open Source Project) под управлением OHA (Open Handset Alliance), руководит всем процессом поисковый гигант Google.

Android поддерживает фоновое выполнение задач; предоставляет богатую библиотеку элементов пользовательского интерфейса; поддерживает 2D и 3D графику, используя OpenGL стандарт; поддерживает доступ к файловой системе и встроенной базе данных SQLite.

С точки зрения архитектуры, система Android представляет собой полный программный стек, в котором можно выделить следующие уровни:

- Базовый уровень (Linux Kernel) - уровень абстракции между аппаратным уровнем и программным стеком;
- Набор библиотек и среда исполнения (Libraries & Android Runtime) обеспечивает важнейший базовый функционал для приложений, содержит виртуальную машину Dalvik и базовые библиотеки Java необходимые для запуска Android приложений;
- Уровень каркаса приложений (Application Framework) обеспечивает разработчикам доступ к API, предоставляемым компонентами системы уровня библиотек;
- Уровень приложений (Applications) - набор предустановленных базовых приложений.

Среда исполнения

Среда исполнения включает в себя библиотеки ядра, обеспечивающие большую часть низкоуровневой функциональности, доступной библиотекам ядра языка Java, и виртуальную машину Dalvik, позволяющую запускать приложения. Каждое приложение запускается в своем экземпляре виртуальной машины, тем самым обеспечивается независимость работающих приложений от ОС и друг от друга. Поэтому код с ошибками или вредоносное ПО не смогут испортить Android и устройство на его базе, когда сработают.

Для исполнения на виртуальной машине Dalvik Java-классы компилируются в исполняемые файлы с расширением .dex с помощью инструмента dx, входящего в состав Android SDK. DEX (Dalvik EXecutable) - формат исполняемых файлов для виртуальной машины Dalvik, оптимизированный для использования минимального объема памяти. При использовании IDE Eclipse и плагина ADT (Android Development Tools) компиляция классов Java в формат .dex происходит автоматически.

Каркас приложений

На еще более высоком уровне располагается каркас приложений (Application Framework), архитектура которого позволяет любому приложению использовать уже реализованные возможности других приложений, к которым разрешен доступ. В состав каркаса входят следующие компоненты:

- богатый и расширяемый набор представлений (Views), который может быть использован для создания визуальных компонентов приложений, например, списков, текстовых полей, таблиц, кнопок или даже встроенного web-браузера;
- контент-провайдеры (Content Providers), управляющие данными, которые одни приложения открывают для других, чтобы те могли их использовать для своей работы;
- менеджер ресурсов (Resource Manager), обеспечивающий доступ к ресурсам без функциональности (не несущим кода), например, к строковым данным, графике, файлам и другим;
- менеджер оповещений (Notification Manager), позволяющий приложениям отображать собственные уведомления для пользователя в строке состояния;
- менеджер действий (Activity Manager), управляющий жизненными циклами приложений, предоставляющий систему навигации по действиям;

Уровень приложений

И, наконец, самый высокий, самый близкий к пользователю уровень приложений. Именно на этом уровне пользователь взаимодействует со своим устройством, управляемым ОС Android. Здесь представлен набор базовых приложений, который предустановлен на ОС Android. Например, браузер, почтовый клиент, программа для отправки SMS, карты, календарь, менеджер контактов и др.

Список интегрированных приложений может меняться в зависимости от модели устройства и версии Android.

К этому уровню также относятся все пользовательские приложения.

Разработчик обычно взаимодействует с двумя верхними уровнями архитектуры Android для создания новых приложений.

Библиотеки, система исполнения и ядро Linux скрыты за каркасом приложений.

Повторное использование компонентов других приложений приводит к идее задач в Android.

Приложение может использовать компоненты другого Android приложения для решения задачи, например, если разрабатываемое приложение предполагает использование фотографий, оно может вызвать приложение, управляющее фотографиями и зарегистрированное в системе Android, выбрать с его помощью фотографию и работать с ней.

Для установки приложения на устройствах с ОС Android создается файл с расширением *.apk (Android package), который содержит исполняемые файлы, а также вспомогательные компоненты, например, файлы с данными и файлы ресурсов. После установки на устройство каждое приложение "живет" в своем собственном изолированном экземпляре виртуальной машины Dalvik.

Обзор сред программирования

Прежде чем начать разрабатывать приложения под Android, рассмотрим существующие инструменты, подходящие для этих целей. Можно выделить необходимые инструменты, без которых разработка мобильных приложений под Android просто невозможна.

С другой стороны, существует большое количество вспомогательных систем, в какой-то мере упрощающих процесс разработки.

К обязательным инструментам относится Android SDK - набор средств программирования, который содержит инструменты, необходимые для создания, компиляции и сборки мобильного приложения.

- **SDK Manager** - инструмент, позволяющий загрузить компоненты Android SDK. Показывает пакеты Android SDK и их статус: установлен (Installed), не установлен (Not Installed), доступны обновления (Update available).
- **Android Emulator (emulator)** - виртуальное мобильное устройство, которое создается и работает на компьютере разработчика, используется для разработки и тестирования мобильных приложений без привлечения реальных устройств.
- **AVD Manager** - предоставляет графический интерфейс для создания виртуальных Android устройств (AVDs), предусмотренных Android Emulator, и управления ими. (В ЛР№1 подробно рассматривается создание и использование виртуального устройства).
- **Android Debug Bridge (adb)** - гибкий инструмент, позволяющий управлять состоянием эмулятора или реального Android устройства, подключенного к компьютеру. Также может использоваться для установки Android приложения (.apk файл) на реальное устройство.

В современных условиях разработка ПО в большинстве случаев ведется с использованием **интегрированных сред разработки (IDE)**.

IDE имеют несомненные достоинства: процесс компиляции, сборки и запуска приложения обычно автоматизирован, в связи с чем для начинающего разработчика создать свое первое приложение труда не составляет. Но чтобы заниматься разработкой всерьез, необходимо потратить силы и время на изучение возможностей самой среды.

Рассмотрим IDE, пригодные для разработки под Android¹⁾.

Android IDE - среда разработки под Android, основанная на Eclipse. Предоставляет интегрированные инструменты для разработки, сборки и отладки мобильных приложений. В данном курсе Android IDE выбрана в качестве основной среды разработки. Возможности этой среды более подробно рассмотрены в первой лабораторной работе. Также там даны рекомендации по установке и настройке среды, созданию и запуску первого приложения как на эмуляторе, так и на реальном устройстве.

Android Studio - среда разработки под Android, основанная на IntelliJ IDEA. Подобно Android IDE, она предоставляет интегрированные инструменты для разработки и отладки. Дополнительно ко всем возможностям, ожидаемым от IntelliJ, в Android Studio реализованы:

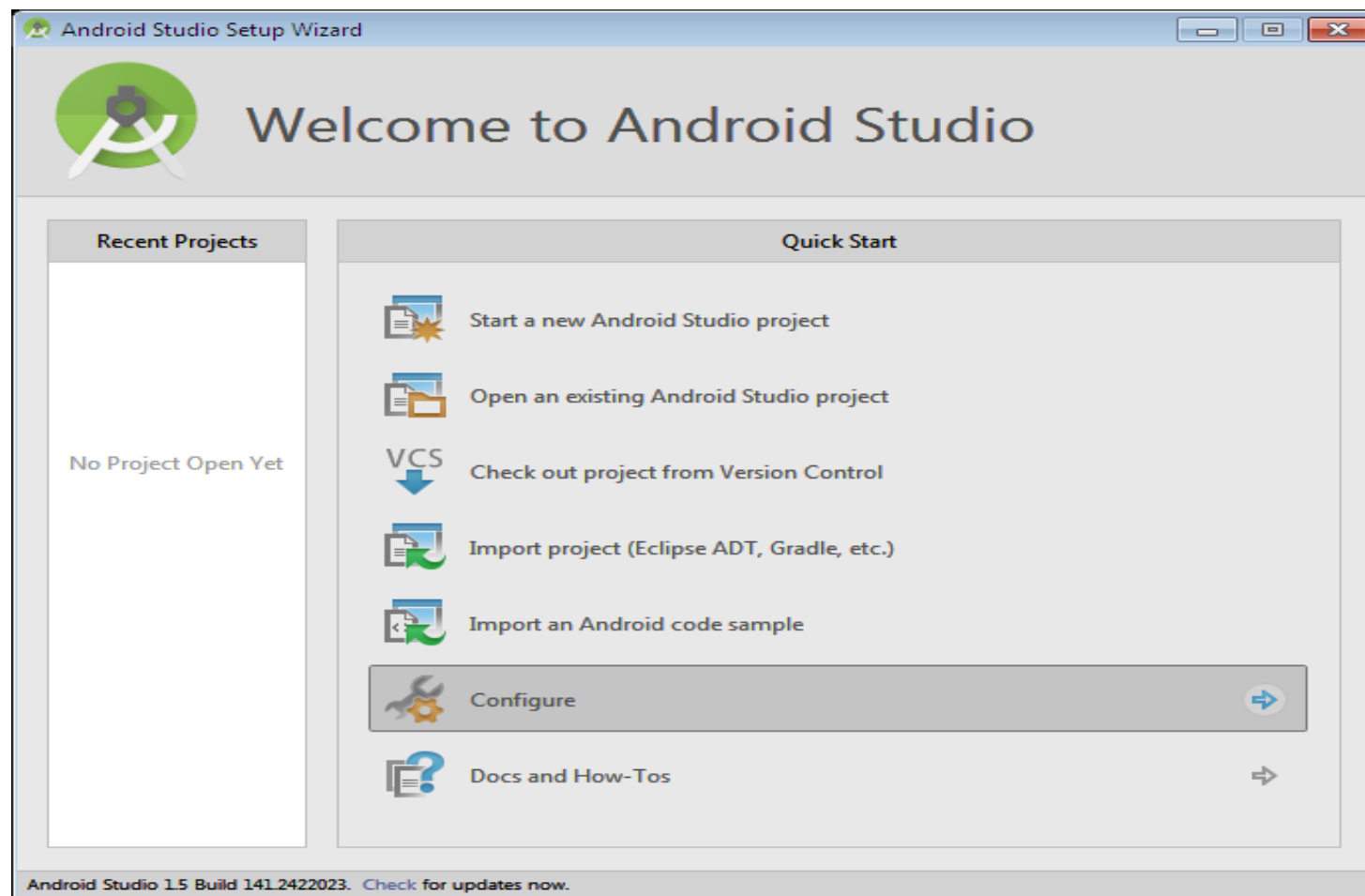
- поддержка сборки приложения, основанной на Gradle;
- специфичный для Android рефакторинг и быстрое исправление дефектов;
- lint инструменты для поиска проблем с производительностью, с юзабилити, с совместимостью версий и других;
- возможности ProGuard (утилита для сокращения, оптимизации и обфускации кода) и подписи приложений;
- основанные на шаблонах мастера для создания общих Android конструкций и компонентов;
- WYSIWYG редактор, работающий на многих размерах экранов и разрешений, окно предварительного просмотра, показывающее запущенное приложение сразу на нескольких устройствах и в реальном времени;
- встроенная поддержка облачной платформы Google.

Эмуляция. Стандартный эмулятор Android

Эмуляция (англ. emulation) в вычислительной технике - комплекс программных, аппаратных средств или их сочетание, предназначенное для копирования (или эмулирования) функций одной вычислительной системы (гостя) на другой, отличной от первой, вычислительной системе (хосте) таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы (гостя). Целью является максимально точное воспроизведение поведения.

Эмулятор - виртуальное мобильное устройство, которое запускается на компьютере. При помощи эмулятора можно разрабатывать и тестировать приложения без использования реальных устройств.

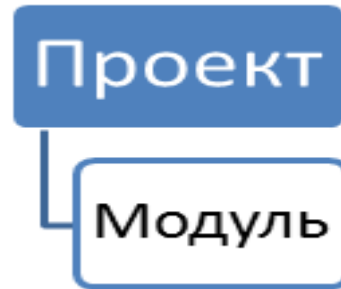
ОСНОВНОЕ ОКНО



Создание приложения

Чтобы создать приложение, нам нужно в Android Studio создать проект. При создании проекта, в нем создается модуль. В этом модуле мы проектируем формы приложения и пишем код. При запуске этого модуля мы получаем готовое приложение. Поэтому модуль по сути и является приложением. А проект - контейнер для модуля.

Таким образом, в самом простом случае структура проекта такова:



Есть проект, и в нем есть модуль. При запуске проекта запускается модуль и мы получаем Android-приложение, которое создано в этом модуле.

В этом случае: один проект = одно Android-приложение (один модуль).

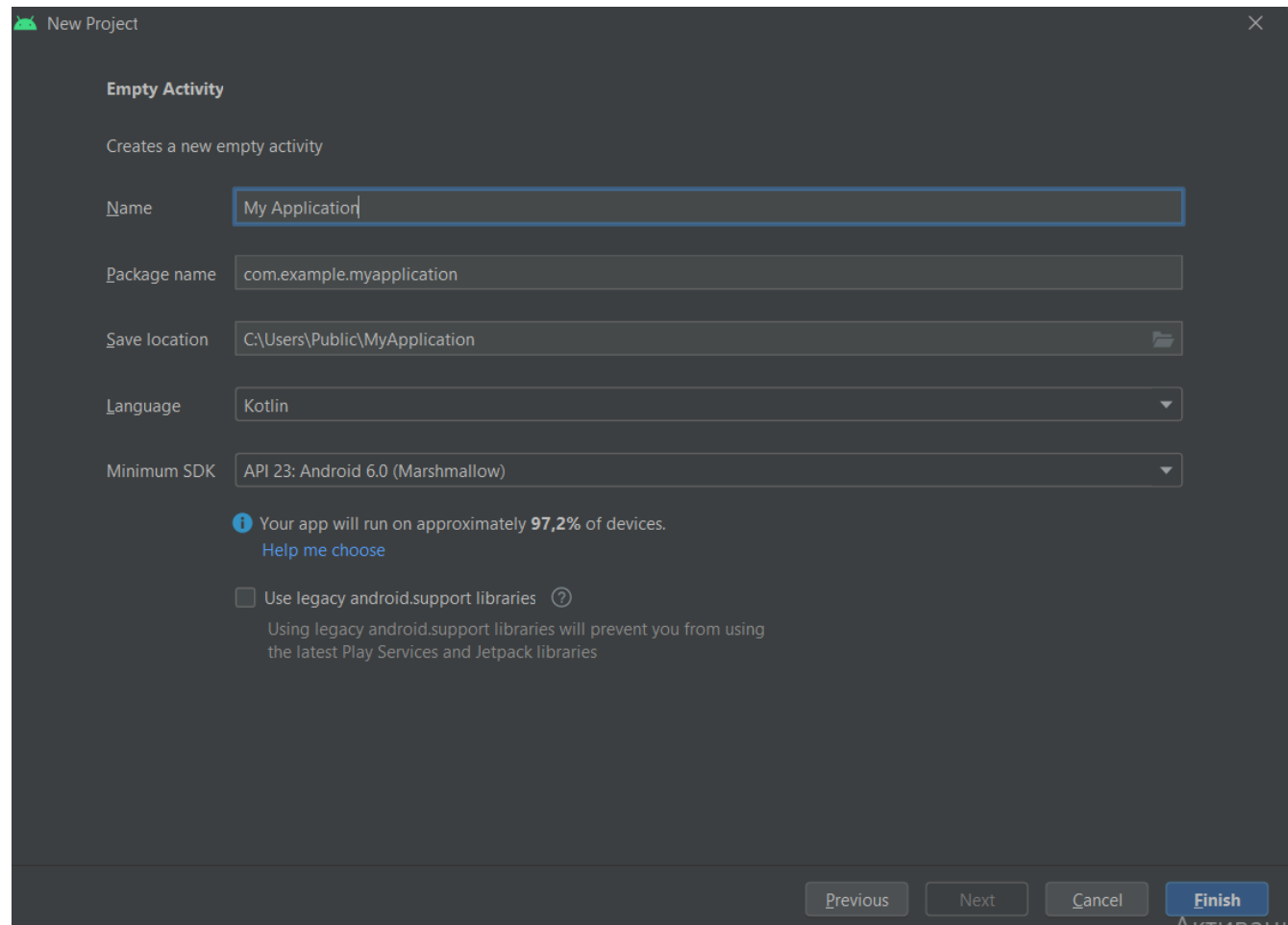
Но в одном проекте может быть несколько модулей.



Создание проекта

Выбираем пункт основного окна

Start a new Android Studio Project



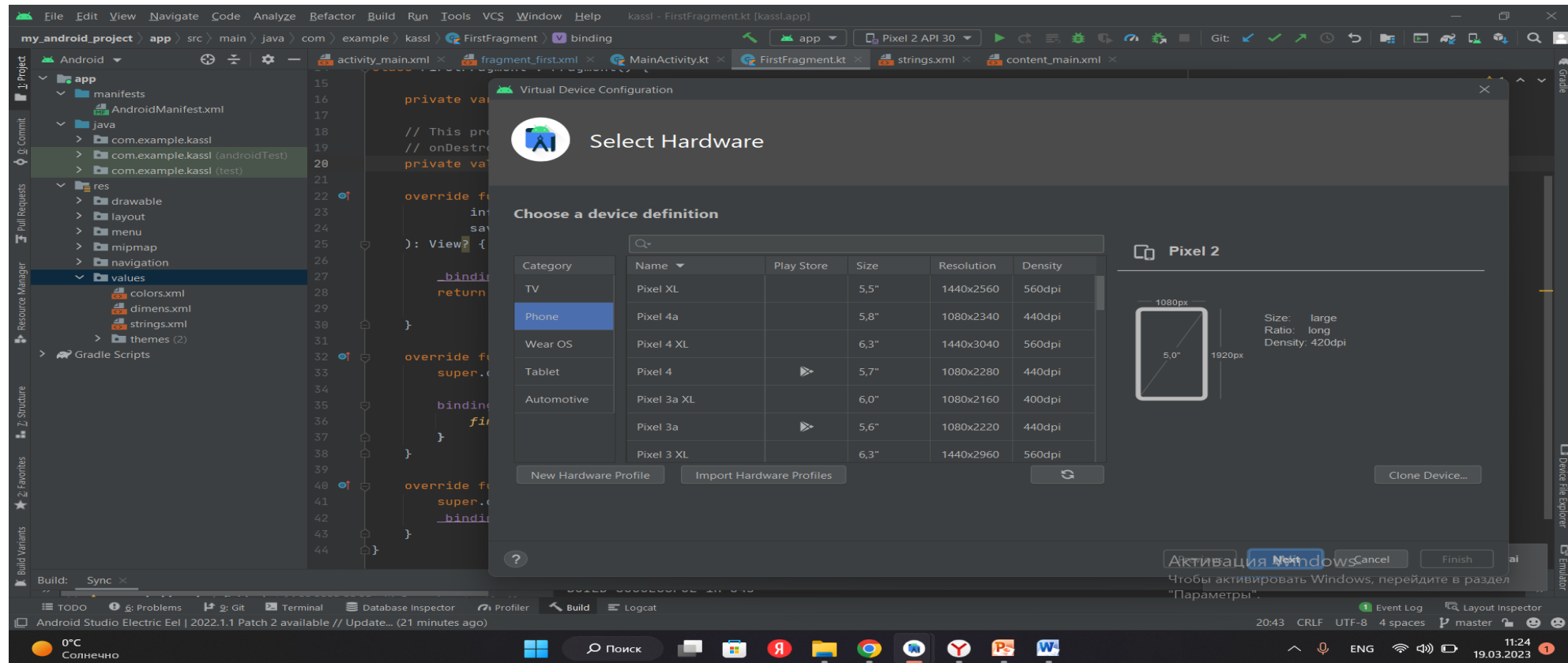
The screenshot shows the 'New Project' dialog in Android Studio. The dialog is titled 'New Project' and has a close button (X) in the top right corner. It contains the following fields and options:

- Empty Activity**: A section header.
- Creates a new empty activity**: A descriptive text.
- Name**: A text field containing 'My Application'.
- Package name**: A text field containing 'com.example.myapplication'.
- Save location**: A text field containing 'C:\Users\Public\MyApplication'.
- Language**: A dropdown menu set to 'Kotlin'.
- Minimum SDK**: A dropdown menu set to 'API 23: Android 6.0 (Marshmallow)'.
- Information**: A blue information icon followed by the text 'Your app will run on approximately 97,2% of devices.' and a link 'Help me choose'.
- Legacy libraries**: A checkbox labeled 'Use legacy android.support libraries' with a question mark icon. Below it, a note states: 'Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries'.
- Navigation**: At the bottom right, there are four buttons: 'Previous' (disabled), 'Next' (disabled), 'Cancel' (disabled), and 'Finish' (active).

Запуск проекта на эмуляторе устройства

После создания проекта нужно создать эмулятор устройства. Это можно сделать, нажав на кнопку на панели инструментов, device manager.

Откроется Android Virtual Device Manager. Пока в нем нет ни одного виртуального устройства. Нужно добавить, выбрать версию и скачать нужную.



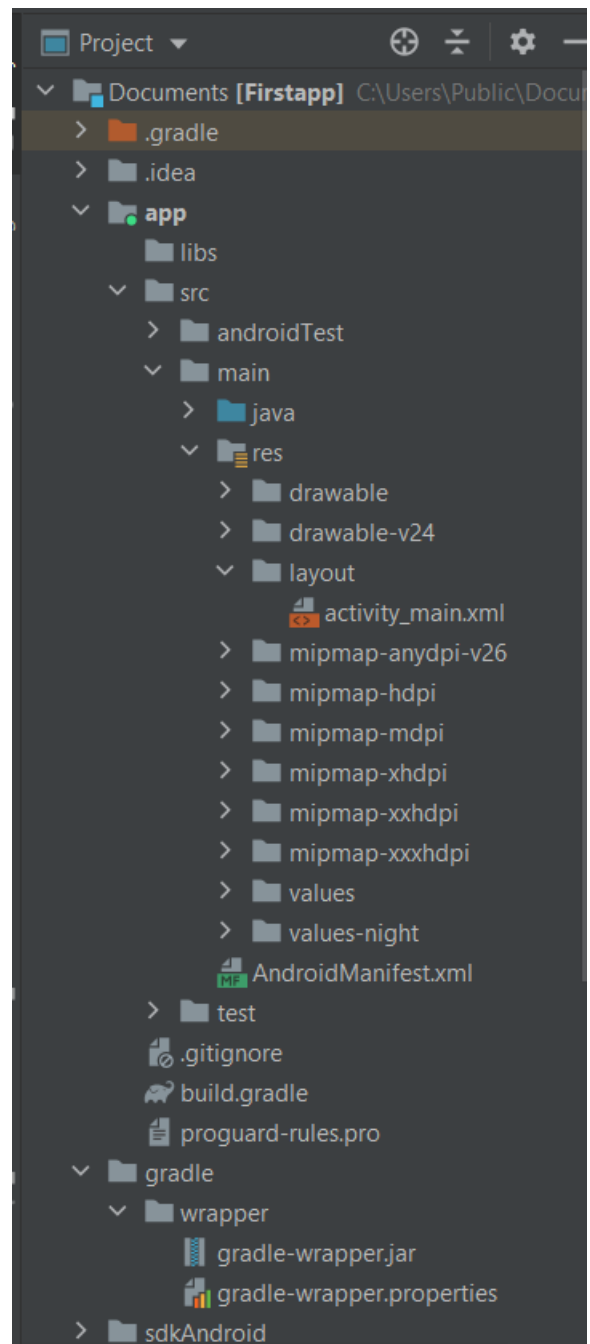
В итоге эмулятор должен заработать и мы попадаем на экран нашего приложения, которое мы как-то назвали



Настройка активности

Большинство приложений на Android имеют свой экран (форму, окно), которое называется активностью или деятельностью (Activity).

Структура проекта



Структура модуля

Можно раскрыть этот модуль и посмотреть его содержимое.

Вкратце пройдемся по интересующим нас элементам

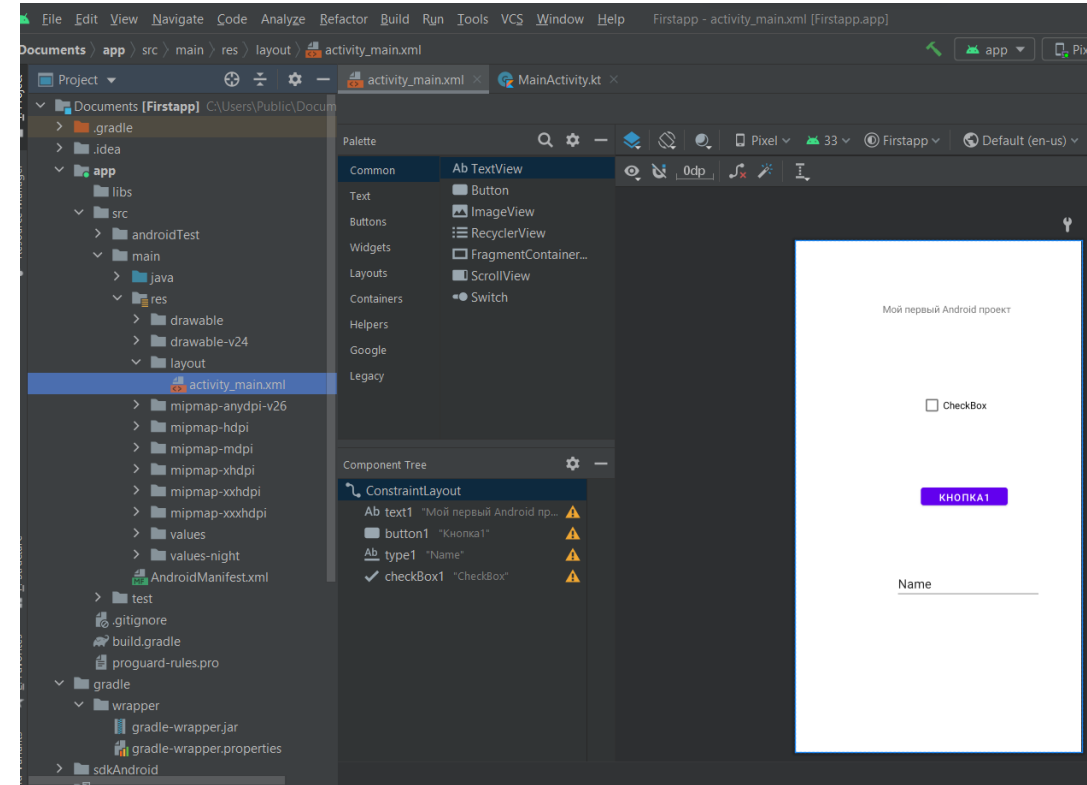
Файл **AndroidManifest.xml** – манифест или конфигурационный файл приложения

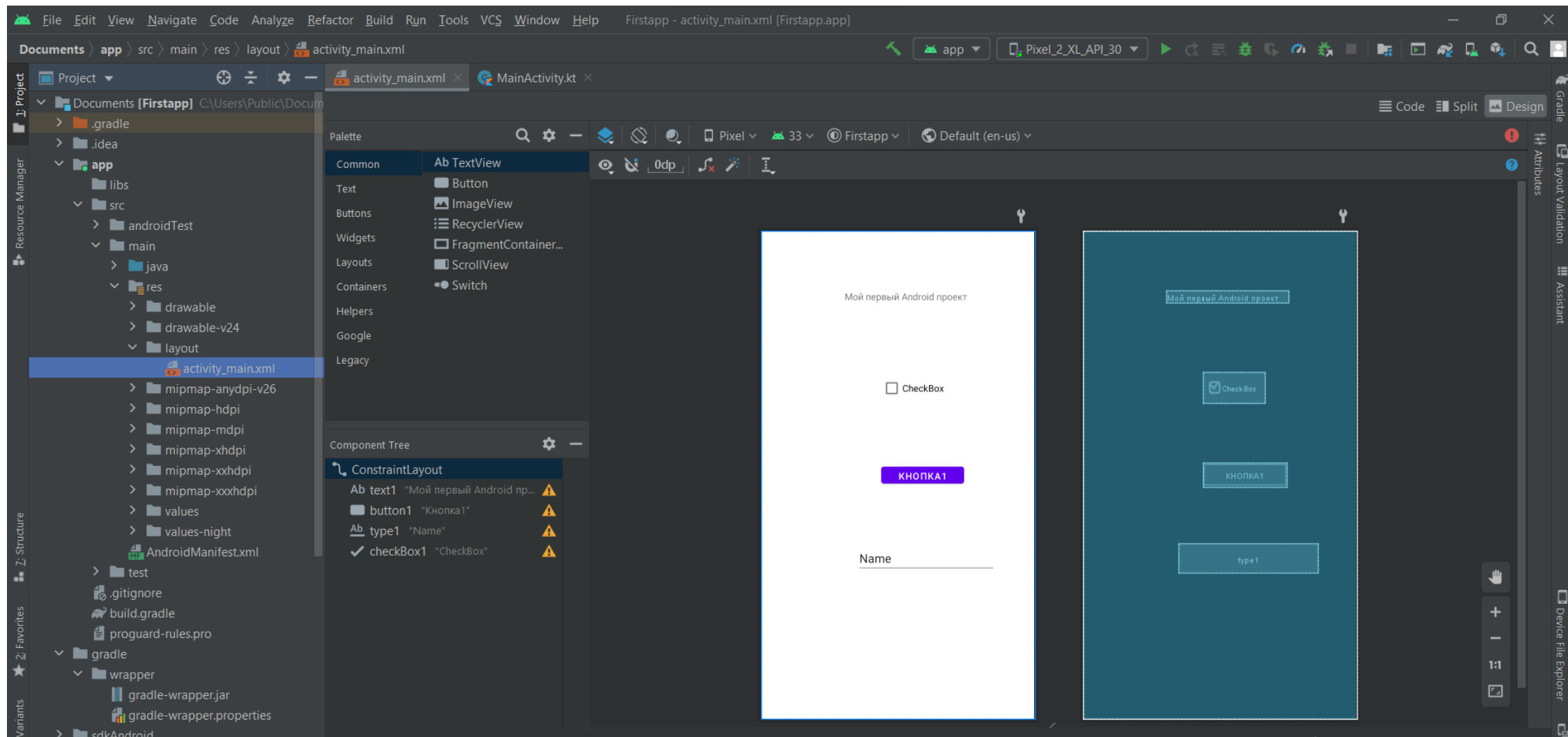
В папке **java** и ее подпапках будет весь написанный нами, код приложения

Папка **res** используется для файлов-ресурсов различного типа.

XML-файл. Графический редактор

В папке res в подпапке layout находится xml-файл, который является оболочкой нашей активности. Именно этот файл будет виден на экране устройства.



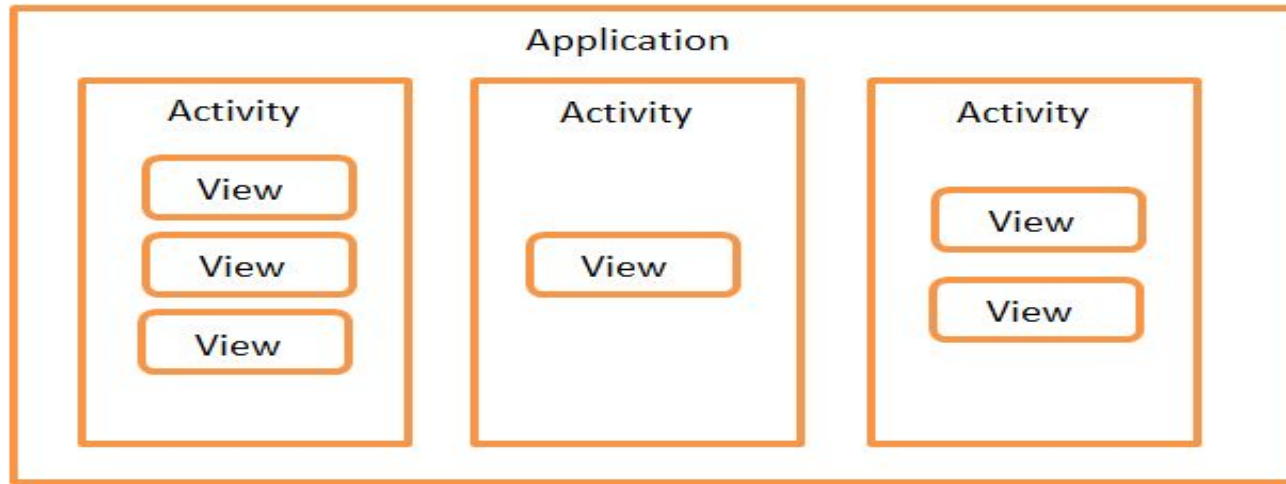


Элементы экрана и их свойства

Если проводить аналогию с Windows, то приложение состоит из окон, называемых Activity. В конкретный момент времени обычно отображается одно Activity и занимает весь экран, а приложение переключается между ними. В качестве примера можно рассмотреть почтовое приложение. В нем одно Activity – список писем, другое – просмотр письма, третье – настройки ящика. При работе вы перемещаетесь по ним.

Содержимое Activity формируется из различных компонентов, называемых View. Самые распространенные View - это текст, кнопка, поле ввода, CheckBox и т.д.

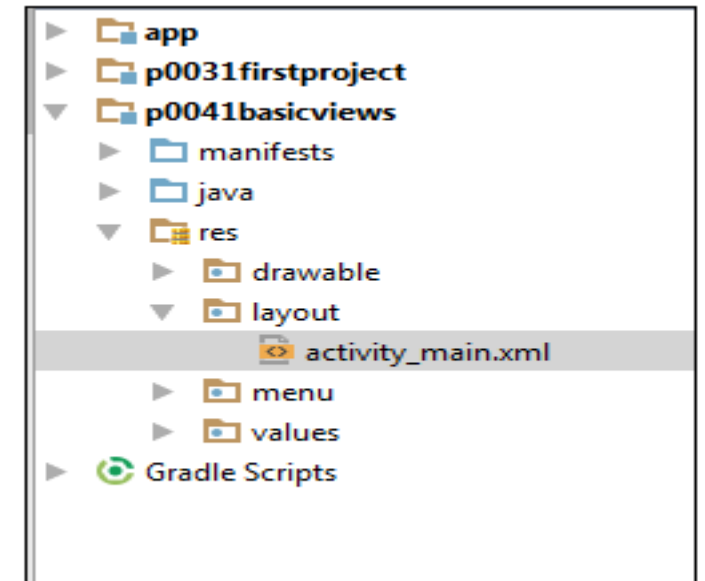
Примерно это можно изобразить так:



Необходимо заметить, что View обычно размещаются в ViewGroup. Самый распространенный пример ViewGroup – это Layout. Layout бывает различных типов и отвечает за то, как будут расположены его дочерние View на экране (таблицей, строкой, столбцом ...)

В нашем модуле нам интересен файл: `res > layout > activity_main.xml`

Это layout-файл. В нем мы определяем набор и расположение элементов View, которые хотим видеть на экране. При запуске приложения Activity читает этот файл и отображает нам то, что мы добавили. Скорее всего, он у вас уже открыт на редактирование, но на всякий случай давайте еще раз откроем его двойным кликом и посмотрим, какой набор View он содержит по умолчанию.,



Слева видим список View, разделенный на группы. Здесь отображены все View-элементы, которые вы можете использовать в своих приложениях.

Обратим внимание на белый экран. Мы видим, что на экране сейчас присутствует элемент с текстом Hello world! Чтобы узнать, что это за View нажмите на этот текст.

Справа во вкладке Component Tree вы видите все элементы, которые описаны в этом layout-файле.

Теперь изменим надписи на компонентах нашего экрана.

Во вкладке Component Tree жмем на textView. Теперь нам нужна вкладка Properties. Она отображает свойства выделенного в Component Tree или на экране View элемента.

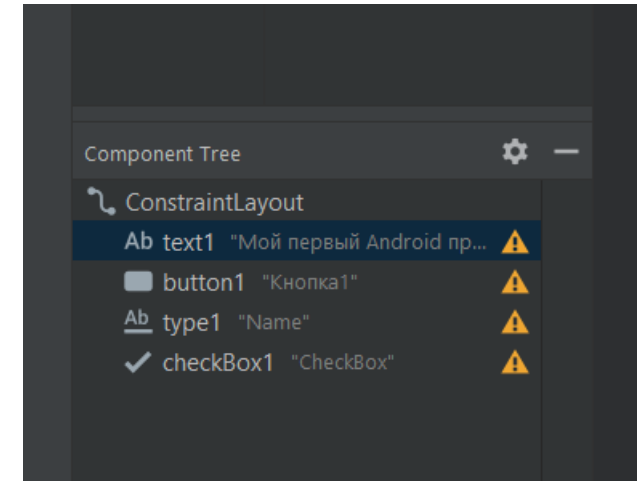
Располагается она обычно сразу под Component Tree.

Найдем во вкладке Properties свойство text. Сейчас там стоит ссылка на текстовую константу. Напишем сюда свой текст: «Мой первый Android проект»

Видим, что выделенный нами элемент – это TextView. Это элемент, который умеет отображать текст. Обратите внимание, что он вложен в элемент ConstraintLayout – это один из видов ViewGroup.

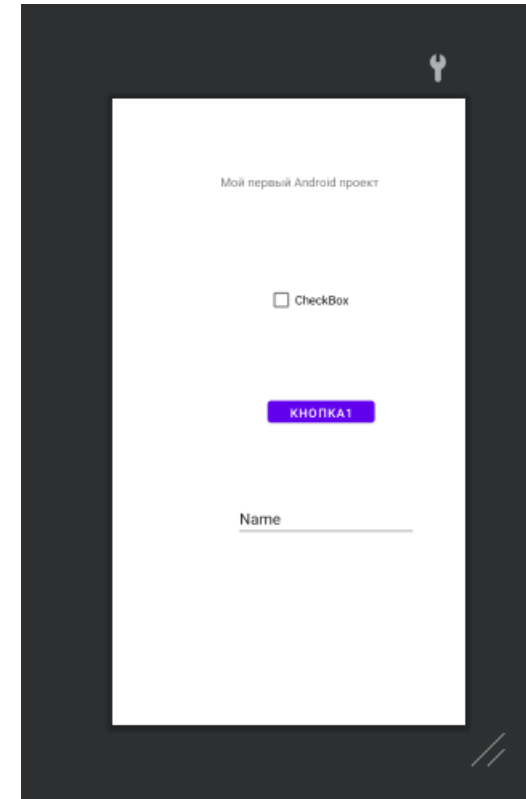
Добавим еще элементов на экран, пусть это будут Button и CheckBox. Для этого просто найдите в списке слева и перетащите на экран вашего будущего приложения. Также можно перетащить их на ConstraintLayout во вкладке Component Tree, результат будет почти тот же. Кроме Button и CheckBox, добавим еще на экран Plain Text из группы Text Fields.

В Component Tree они появятся под названиями button1, checkBox1 и Text1, type1. данные названия, это ID, заданные пользователем



После этих манипуляций ваш экран будет выглядеть примерно так:

Вся эта конфигурация экрана сохранится в файле `activity_main.xml`.



Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:text="HelloWorld!"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="24dp"
        android:gravity="center"
        android:textSize="25sp"/>
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="24dp"
        android:text="Click Me"/>
</LinearLayout>
```



Каждая строка задает параметры элемента на экране: надпись, размеры, величину букв и т. п. Можно поиграться с параметрами, добиваясь желаемого эффекта.

Activity_main.xml

Изменим фон для экрана приложения. Сейчас у нас экран белого цвета.

Возвращаемся в файл разметки **activity_main.xml**. Справа найдите вкладку **Attributes**, в которой отображаются свойства для выбранного элемента. А слева есть вкладка **Component Tree**, который отображает структуру компонентов на экране.

Необходимо выделить какой-нибудь компонент, чтобы на вкладке свойств увидеть все доступные свойства компонента.

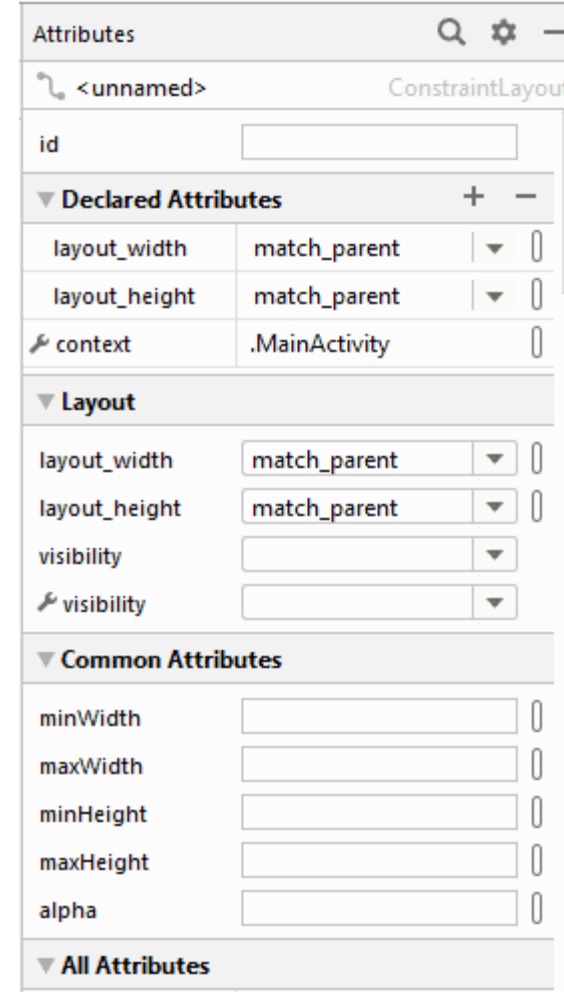
Сейчас у нас есть окно активности, графическая кнопка **ImageButton** и текстовая метка **TextView** с надписью **Hello World!**.

Так как мы собираемся работать с фоном экрана приложения, то щёлкните на **ConstraintLayout**. В панели свойств отобразятся самые употребительные свойства выбранного компонента. К ним относятся идентификатор, ширина и высота.

Находим свойство **background**. Щёлкните рядом с этим словом во второй колонке, где нужно прописывать значения.

Появится текстовое поле, в которое можно ввести значение вручную. Рядом есть маленькая кнопка, которая запустит диалоговое окно для создания ресурса.

Переходим на вкладку **Color** и выбираем цвет. Вы можете выбрать цвет, определённый в приложении (секция **app**) или в системе (секция **android**).



Если переключиться в текстовый режим, то увидим, что у элемента **ConstraintLayout** добавилась строка: `android:background="@color/colorAccent"`. Мы связали фон экран с именем цветового ресурса. Всегда используйте ресурсы.

Activity_main.xml

Добавим image на кнопку.

Находим подходящее изображение и копируем его в папку **res/drawable**.

Простое перетаскивание из проводника в папку студии не сработает. Поэтому лучше скопировать картинку в буфер, затем щёлкнуть правой кнопкой мыши на папке **drawable** в студии, выбрать команду "Вставить". Сначала появится первое диалоговое окно для выбора папки, выбираем папку **drawable**.

Выделяем элемент **ImageButton** на форме и в панели свойств выбираем свойство **srcCompat**, щёлкаем на кнопку и выбираем в диалоговом окне ресурс в категории **Drawable** - вы там должны увидеть ресурс *pinkhellokitty* (имя добавленного ранее файла).

Если переключиться в текстовый режим, то увидим, что у элемента **ConstraintLayout** добавилась строка:
`android:background="@color/colorAccent"`
Мы связали фон экран с именем цветового ресурса.
Всегда используйте ресурсы.

Чтобы обратиться к элементу экрана из кода, нам нужен его ID. Он прописывается либо в Properties, либо в layout-файлах, как вам удобнее. Для ID существует четкий формат - @+id/name, где + означает, что это новый ресурс и он должен добавиться в R.java класс, если он там еще не существует

MainActivity.kt

Далее откроем класс MainActivity и добавим для обработки нажатие и после-нажатия запускается 2й экран.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    val textView = findViewById<TextView>(R.id.textView)  
    val button = findViewById<Button>(R.id.button)  
    button.setOnClickListener {  
        textView.text = "You clicked button"  
    }  
}
```

Для того, чтобы обратиться к элементу View из кода, надо вызвать метод `findViewById`. Он по ID возвращает View.

Давайте напишем вызов этого метода. Напомню, что пока мы пишем наш код в методе `onCreate`. Это метод, который вызывается при создании Activity.

Откроем `MainActivity.kt` и после строки с вызовом метода `setContentView` напишем:

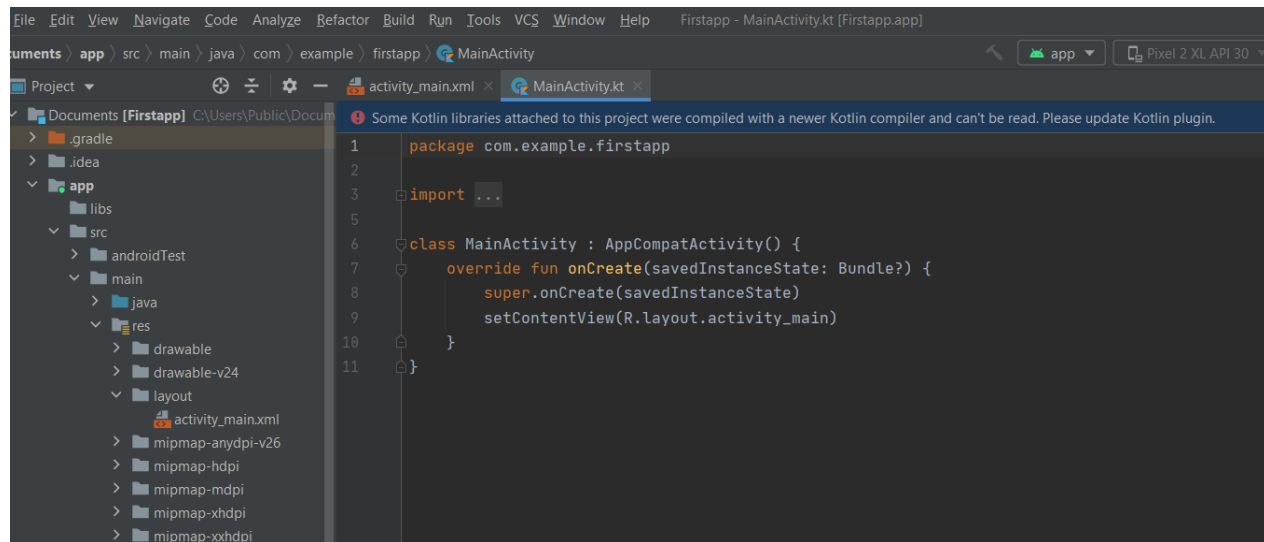
```
View myTextView = findViewById(R.id.myText);
```

Если View подчеркнуто красным, то скорее всего этот класс не добавлен в секцию `import`. Нажмите `CTRL+SHIFT+O` для автоматического обновления импорта.

Создание второго экрана

откроем класс MainActivity и добавим для обработки нажатие и после нажатие изменяем текст.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    val textView = findViewById<TextView>(R.id.textView)  
    val button = findViewById<Button>(R.id.button)  
    button.setOnClickListener {  
        textView.text = "You clicked button"  
    }  
}
```

Смотрим наш код.

Нас интересует метод onCreate – он вызывается, когда приложение создает и отображает Activity

Посмотрим код реализации onCreate.

Нас сейчас интересует строка:

`setContentView(R.layout.activity_main);`

Метод `setContentView(int)` устанавливает содержимое Activity из layout-файла. Но в качестве аргумента мы указываем не путь к layout-файлу (`res/layout/activity_main.xml`), а константу, которая является ID файла. Это константа генерируется автоматически в классе `R.java`. В этом классе будут храниться сгенерированные ID для всех ресурсов проекта (из папки `res/*`), чтобы мы могли к ним обращаться. *Имена этих ID-констант совпадают с именами файлов ресурсов (без расширений).*

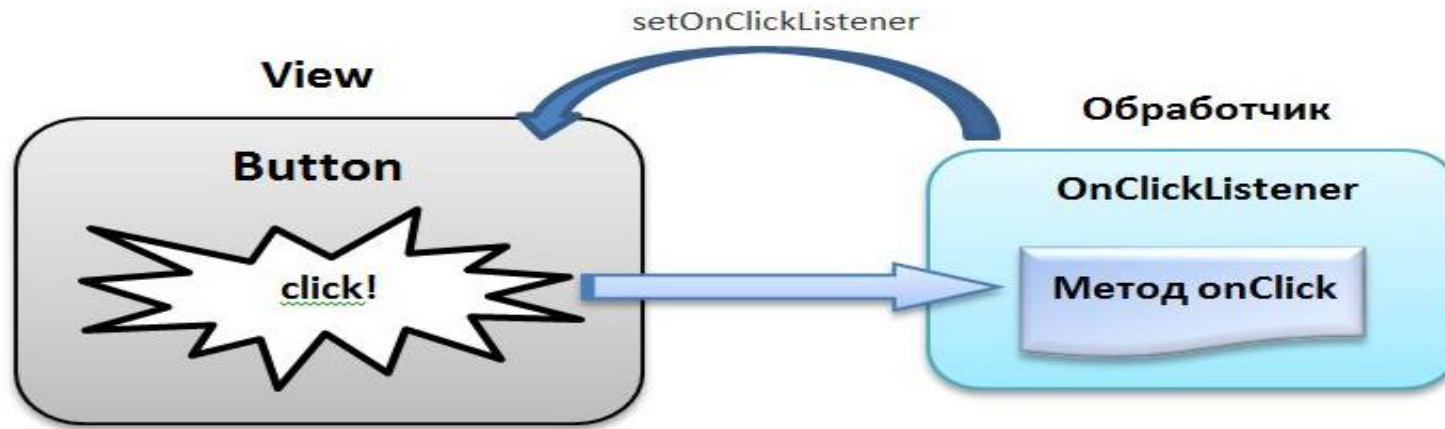
Нажатие на Cancel пока ни к чему не приводит, т.к. для нее мы обработчик не создали и не присвоили. Давайте сделаем это аналогично, как для кнопки ОК. Сначала мы создаем обработчик:

```
OnClickListener oclBtnCancel = new OnClickListener() {  
  
    @Override  
  
    public void onClick(View v) {  
  
        // Меняем текст в TextView (tvOut)  
  
        tvOut.setText("Нажата кнопка Cancel");  
  
    }  
  
};
```

Потом присваиваем его кнопке:

```
btnCancel.setOnClickListener(oclBtnCancel);
```

Еще раз повторим механизм обработки событий на примере перехода по тексту. Здесь нужен обработчик (его также называют слушателем - listener), который присваивается с помощью метода `setOnClickListener`. Когда на текст/кнопку нажимают, обработчик реагирует и выполняет код из метода `onClick`. Это можно изобразить так:



Соответственно для реализации необходимо выполнить следующие шаги:

- создаем обработчик
- заполняем метод `onClick`
- присваиваем обработчик кнопке

и система обработки событий готова.