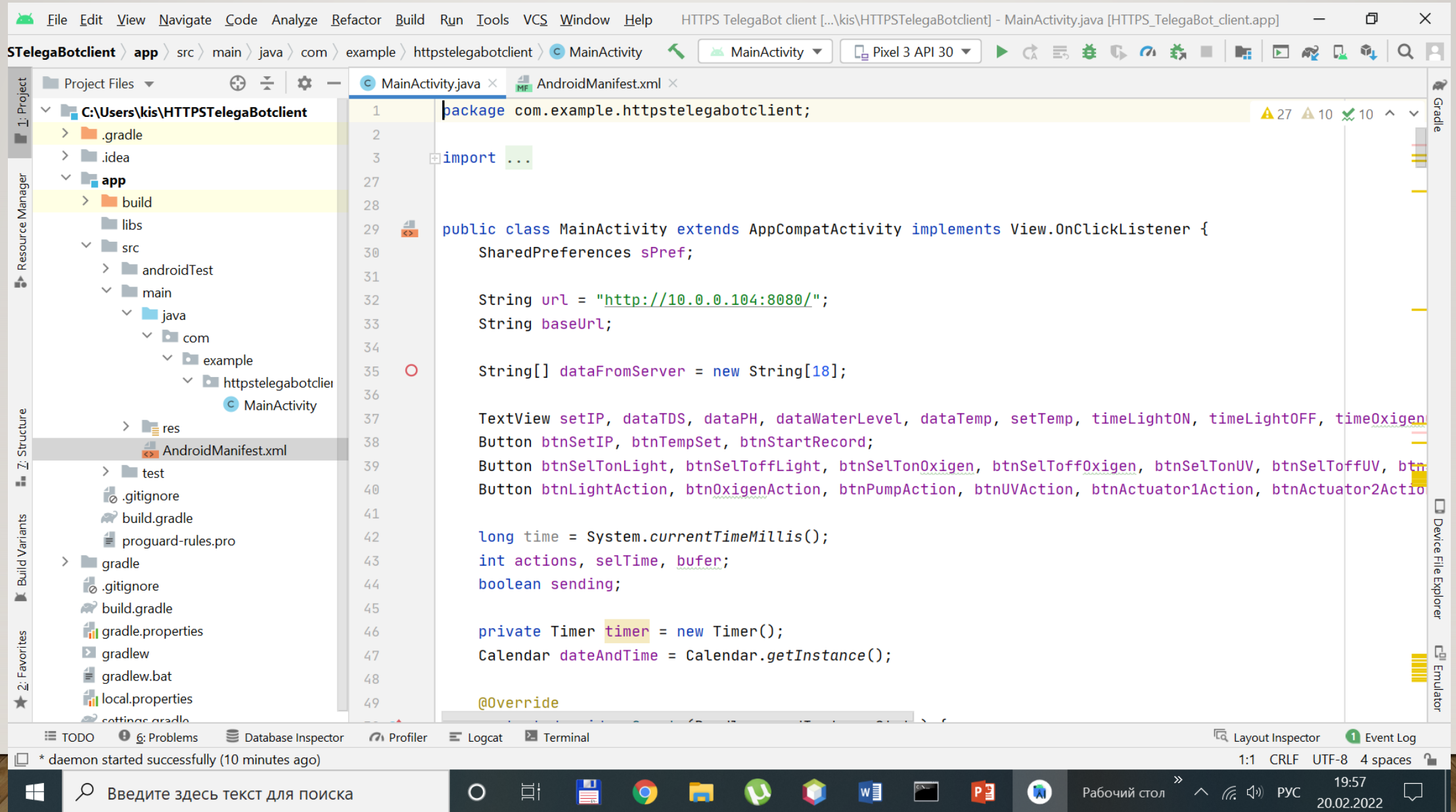


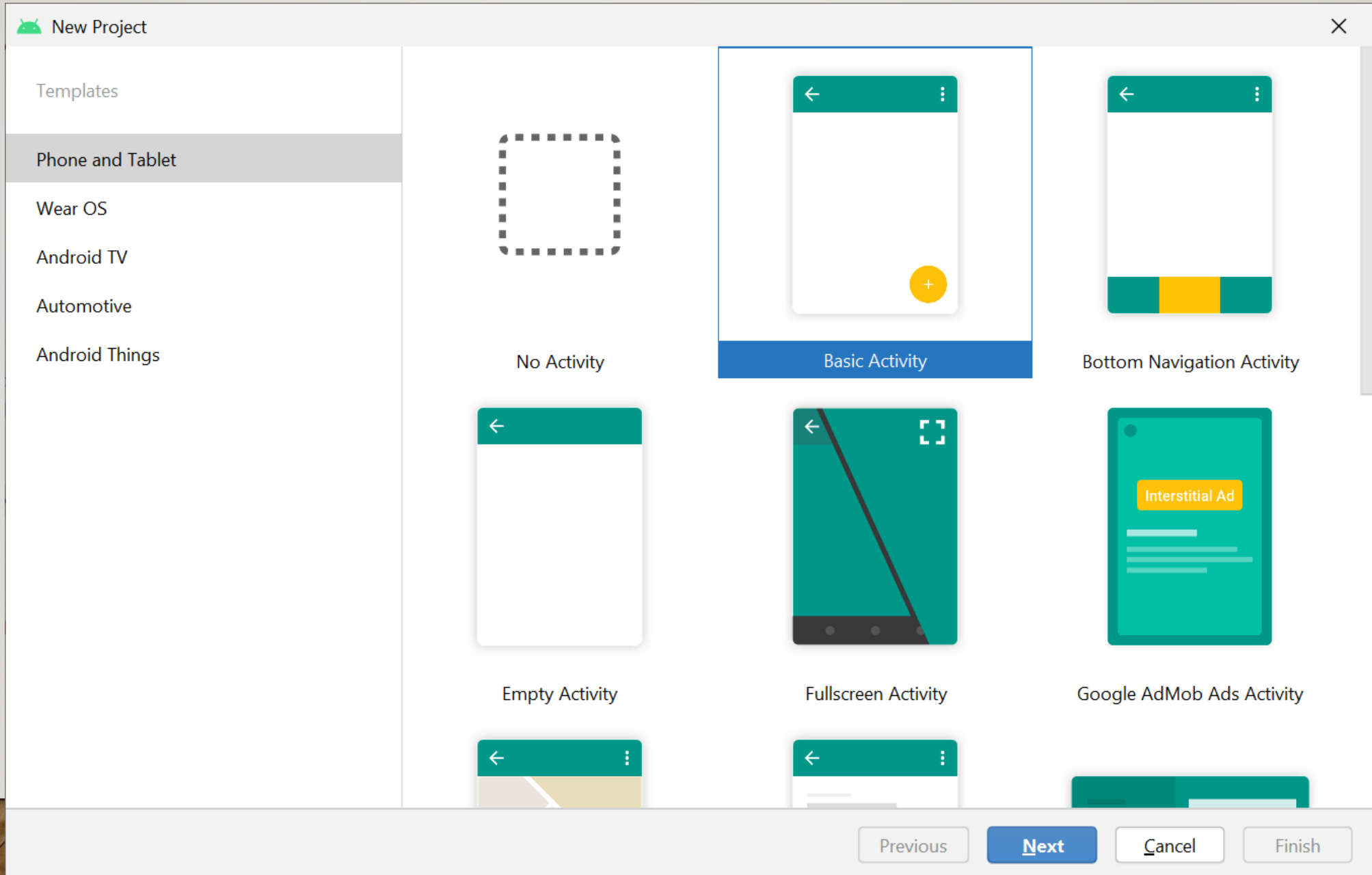
РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

ANDROID STUDIO

ANDROID STUDIO



НОВЫЙ ПРОЕКТ



НОВЫЙ ПРОЕКТ



New Project



Basic Activity

Creates a new basic activity with the Navigation component

Name

My Application

Package name

com.example.myapplication

Save location

C:\Users\kis\AndroidStudioProjects\MyApplication3



Language


Java




Minimum SDK

API 30: Android 11.0 (R)



 Your app will run on approximately **24,3%** 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

Previous

Next

Cancel

Finish

ПРОЕКТ

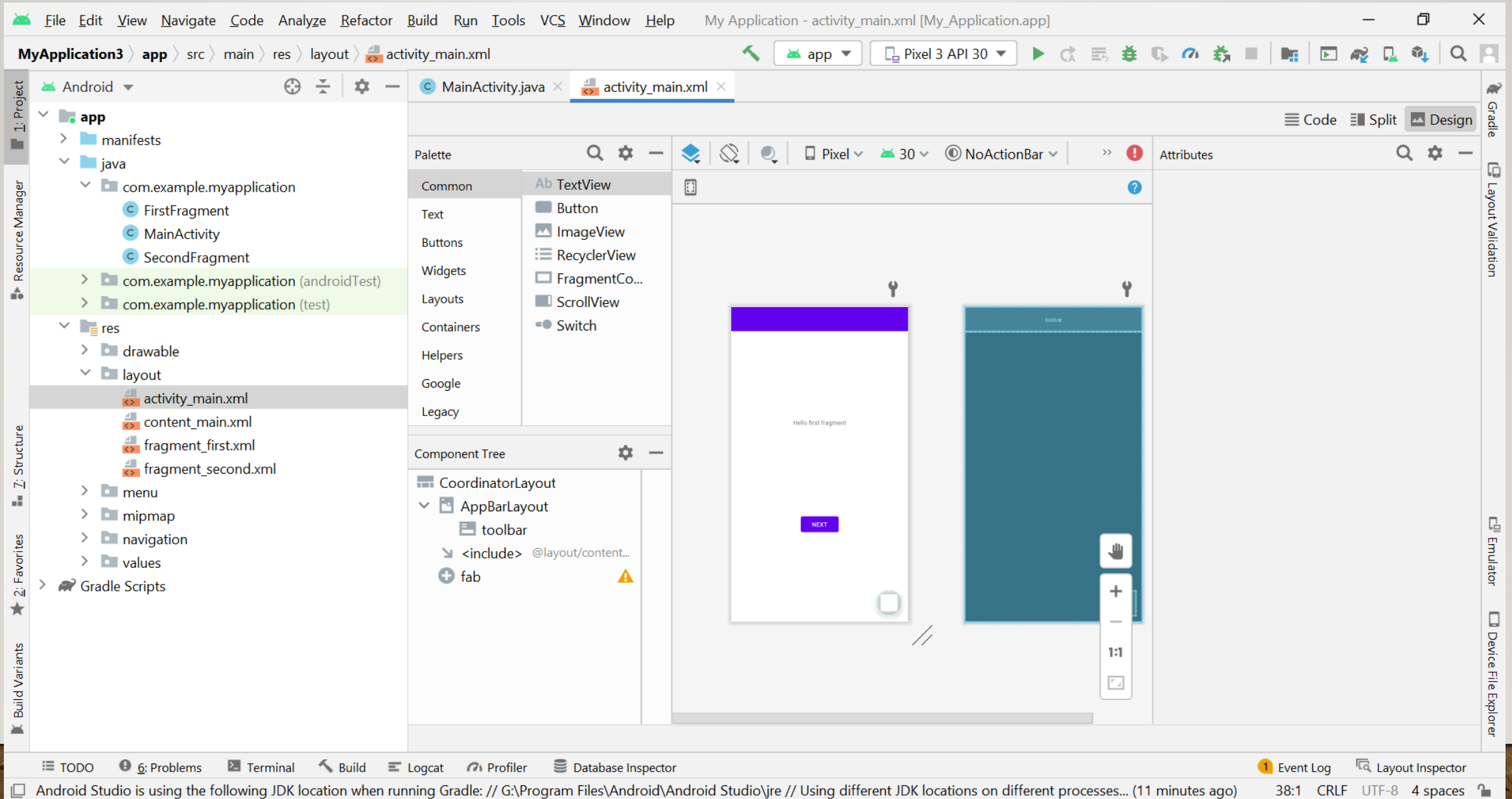
MyApplication3 > app > src > main > java > com > example > myapplication > MainActivity

Run 'app' Shift+F10

```
1 package com.example.myapplication;
2
3 import ...
4
20
21 public class MainActivity extends AppCompatActivity {
22
23     private AppBarConfiguration appBarConfiguration;
24     private ActivityMainBinding binding;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29
30         binding = ActivityMainBinding.inflate(getLayoutInflater());
31         setContentView(binding.getRoot());
32
33         setSupportActionBar(binding.toolbar);
34
35         NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_
36         appBarConfiguration = new AppBarConfiguration.Builder(navController.getGraph()).build(
37         NavigationUI.setupActionBarWithNavController( activity: this, navController, appBarConfigur
38
39         binding.fab.setOnClickListener(new View.OnClickListener() {
40             @Override
41             public void onClick(View view) {
42                 Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
```

Run selected configuration

1:1 CRLF UTF-8 4 spaces



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help My Application - activity_main.xml [My_Application.app]

MyApplication3 > app > src > main > res > layout > activity_main.xml

Android > MainActivity.java > activity_main.xml

Code Split Design

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   tools:context=".MainActivity">
8
9   <com.google.android.material.appbar.AppBarLayout
10     android:layout_width="match_parent"
11     android:layout_height="wrap_content"
12     android:theme="@style/Theme.MyApplication.AppBarOverlay">
13
14     <androidx.appcompat.widget.Toolbar
15       android:id="@+id/toolbar"
16       android:layout_width="match_parent"
17       android:layout_height="?attr/actionBarSize"
18       android:background="?attr/colorPrimary"
19       app:popupTheme="@style/Theme.MyApplication.PopupOverlay" />
20
21   </com.google.android.material.appbar.AppBarLayout>
22
23   <include layout="@layout/content_main" />
24
```

androidx.coordinatorlayout.widget.CoordinatorLayout

TODO Problems Terminal Build Logcat Profiler Database Inspector

Android Studio is using the following JDK location when running Gradle: // G:\Program Files\Android\Android Studio\jre // Using different JDK locations on different processes mi... (today 20:09) 5:34 CRLF UTF-8 4 spaces

xml activity_hello_world.xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_hello_world"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.java2blog.helloworldapp.HelloWorldActivity">

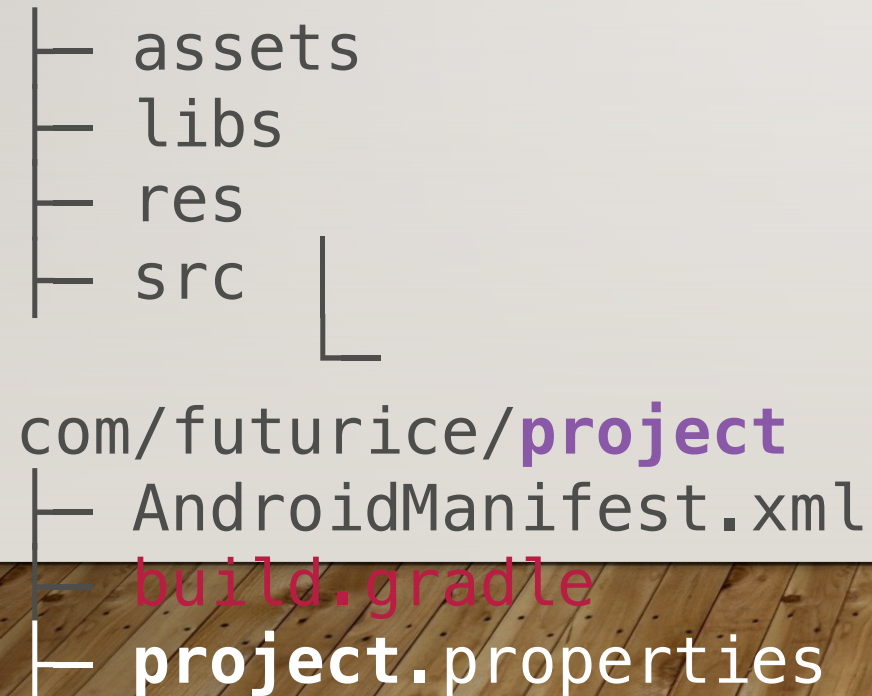
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```


Структура проекта

Есть два распространённых варианта: старая *Ant & Eclipse ADT* структура проекта — либо новая *Gradle & Android Studio*. Лучше выбрать второй вариант. Если ваш проект использует старую структуру, рекомендуем её портировать.

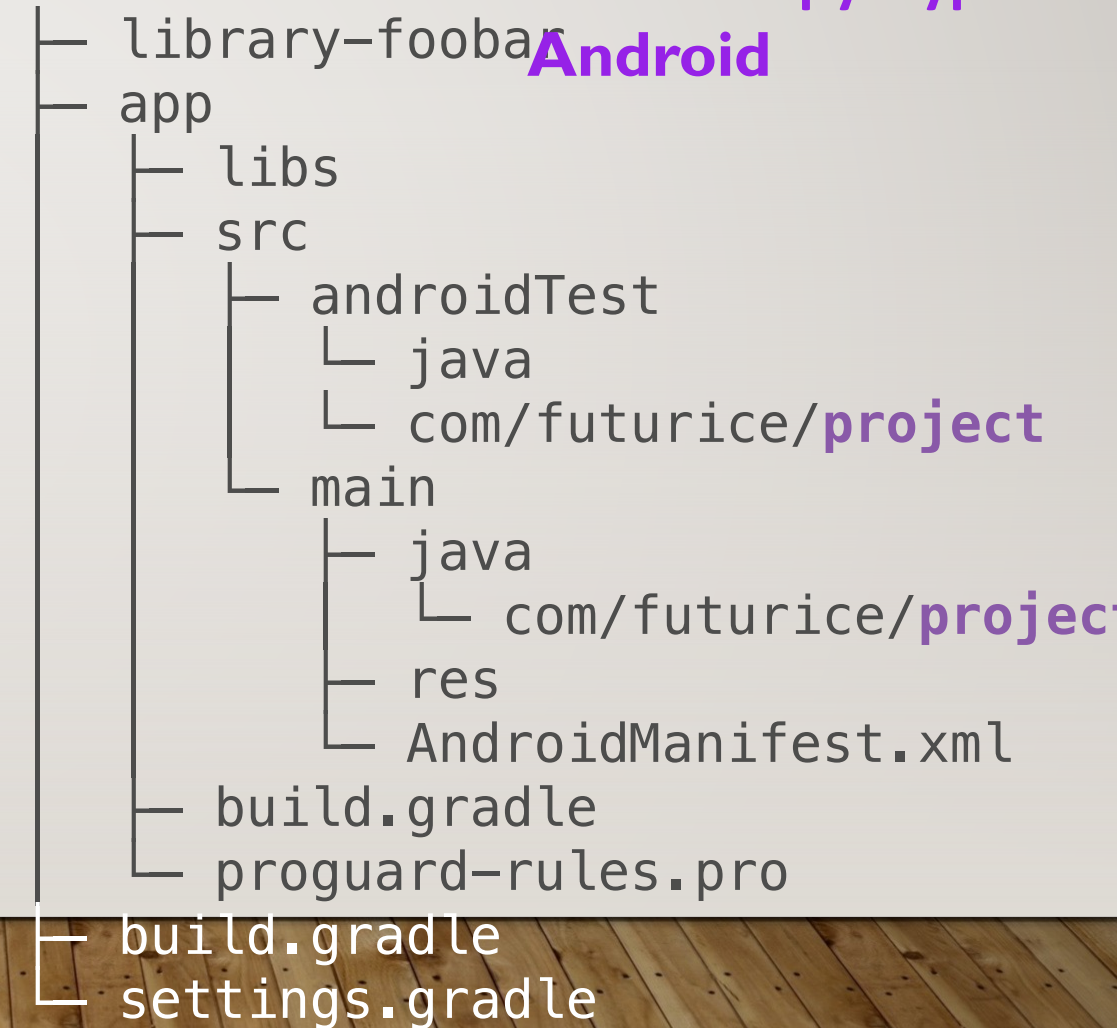
Старая структура Eclipse

old-structure



new-structure

Новая структура Android

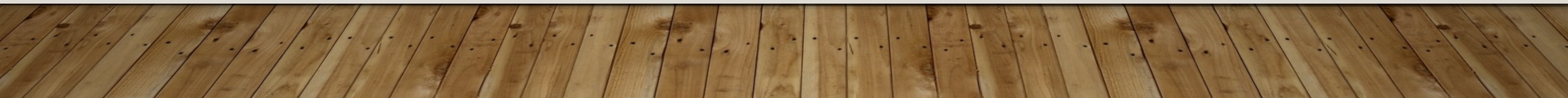


Android. Структура проекта

Android проект представлен 2 корневыми папками: *app* и *Gradle Scripts*

Папка *app* включает 3 подпапки:

1. Папка *manifests* содержит файлы конфигураций или файлы манифеста приложения
2. В папке *java* находится исходный код приложения.
3. Папка *res* содержит файлы используемых в *Android* приложения ресурсов (картинки, стили, размерности для различных устройств и т.д.)



Проект **Android** может состоять из различных модулей.

По умолчанию, когда мы создаем проект, создается один модуль - **app**,

Состоящий из трёх подпапок:

- **manifests**: хранит файл манифеста **AndroidManifest.xml**, который описывает конфигурацию приложения и определяет каждый из компонентов данного приложения.
- **java**: хранит файлы кода на языке java, которые структурированы по отдельным пакетам.

Так, в папке ***com.example.helloapp***

(название которого было указано на этапе создания проекта)

имеется по умолчанию файл **MainActivity.java**

с кодом на языке Java, который представляет класс **MainActivity**,

запускаемый по умолчанию при старте приложения

java

Папка *java* содержит исходный код приложения. Классы могут быть расположены в различных пакетах, но обязательно внутри папки *java*.



res

В папке **res** расположены все используемые приложением ресурсы. Внутри папки **res** эти все ресурсы распределены по своим папкам:

- Папка ***drawable*** содержит файлы с изображениями, которые будут использоваться в приложении.
- Папка ***layout*** располагает xml файлами, которые используются для построения пользовательского интерфейса ***Android*** приложения. По умолчанию здесь есть файл ***activity_main.xml***
- В папке ***menu*** находятся xml файлы, используемые только для создания меню.
- В ***ipmap*** папке хранят только значки приложения. Любые другие ***drawable*** элементы должны быть размещены в своей папке.
- ***values*** хранит те ***xml*** файлы, в которых определяются простые значения типа строк, массивов, целых чисел, размерностей, цветов и стилей

- папка **values** хранит различные xml-файлы, содержащие коллекции ресурсов - различных данных, которые применяются в приложении. По умолчанию здесь есть два файла и одна папка:
 - файл **colors.xml** хранит описание цветов, используемых в приложении
 - файл **strings.xml** содержит строковые ресурсы, используемые в приложении
 - папки **themes** хранит две темы приложения - для светлую (дневную) и темную (ночную)



AndroidManifest.xml файл

Файл *AndroidManifest.xml* - один из самых важных в *Android* проекте. В нем содержится информация о пакетах приложениях, компонентах типа *Activity*, *Service* (и др.).

Файл *AndroidManifest.xml* выполняет следующие задачи:

- Предоставляет разрешения приложению на использование или доступ к другим компонентам системы.
- Определяет как будут запускаться, например, *Activity* (какие фильтры использовать).



В файле манифеста только два элемента: **<manifest>** и **<application>** являются обязательными и при этом встречаются ровно по одному разу. Остальные элементы могут встречаться несколько раз или не появляться совсем, в этом случае *манифест* определяет пустое *приложение*.
Следующий листинг демонстрирует общую структуру файла манифеста.



```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest>
```

```
    <uses-permission />
```

```
    <permission />
```

```
    <permission-tree />
```

```
    <permission-group />
```

```
    <instrumentation />
```

```
    <uses-sdk />
```

```
        <uses-configuration />
```

```
    <uses-feature />
```

```
    <support-screens />
```

```
    <compatible-screens />
```

```
    <supports-gl-texture />
```


<application>

<activity>

<intent-filter>

<action />

<category />

<data />

</intent-filter>

<meta-data />

</activity>

<activity-alias>

<intent-filter> ... </intent-filter>

<meta-data />

</activity-alias>

<service>

<intent-filter> ... </intent-filter>

<meta-data />

</service>

<receiver>

<intent-filter> ... </intent-filter>

<meta-data />

</receiver>

<provider>

<grant-uri-permission />

<meta-data />

<path-permission />

</provider>

<uses-library />

</application>

</manifest>

МАНИФЕСТ

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.java2blog.helloworldapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".HelloWorldActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

Gradle

Скрипты *Gradle* используются для автоматизации сборки проекта.

Этот процесс сборки осуществляется с использованием системы *Gradle* — инструментария для автоматической сборки с помощью набора конфигурационных файлов.

Gradle скрипты написаны на языке *groove*



Gradle

Gradle — система автоматической сборки, построенная на принципах Apache Ant и Apache Maven, но предоставляющая DSL на языках Groovy и Kotlin вместо традиционной XML-образной формы представления конфигурации проекта.

В отличие от Apache Maven, основанного на концепции жизненного цикла проекта, и Apache Ant, в котором порядок выполнения задач (targets) определяется отношениями зависимости (depends-on), Gradle использует направленный ациклический граф для определения порядка выполнения задач.

Gradle был разработан для расширяемых многопроектных сборок, и поддерживает инкрементальные сборки, определяя, какие компоненты дерева сборки не изменились и какие задачи, зависящие от этих частей, не требуют перезапуска.

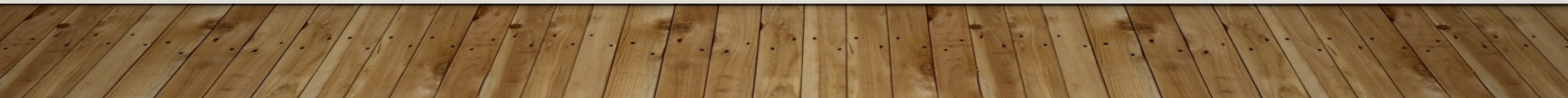
idea

Папка *.idea* на картинке не видна, но если выбрать закладку *Project Files*, то она появится.


Среда разработки *Eclipse* использует файл *project.properties* для настройки метаданных проекта.




В *Android Studio* этим занимается папка *.idea*.


Это означает, что метаданные конкретного проекта хранятся в *Android Studio*.



ОТЛАДКА






 Android Virtual Device Manager








Your Virtual Devices

Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 3 API 30		1080 × 2160: 440dpi	30	Android 11.0 (Goog...	x86	9,6 GB	  

 Create Virtual Device...



Основные задачи разработчика мобильных приложений:

- создание ТЗ (технического задания) на разработку мобильного приложения;
- обсуждение с заказчиком этапов и хода работы проекта;
- построение архитектуры приложения;
- непосредственно программирование;
- работа с дизайнерами;
- поддержка мобильных приложений;
- работа с тестировщиками над отладкой и тестированием приложений;
- помощь в создании инструкций по работе с готовым приложением;
- оформление документации;
- размещение приложений в AppStore и Google Play Market, Amazon Appstore, Opera Mobile Store и других магазинах мобильных приложений.



РАЗРАБОТКА

Архитектура приложения

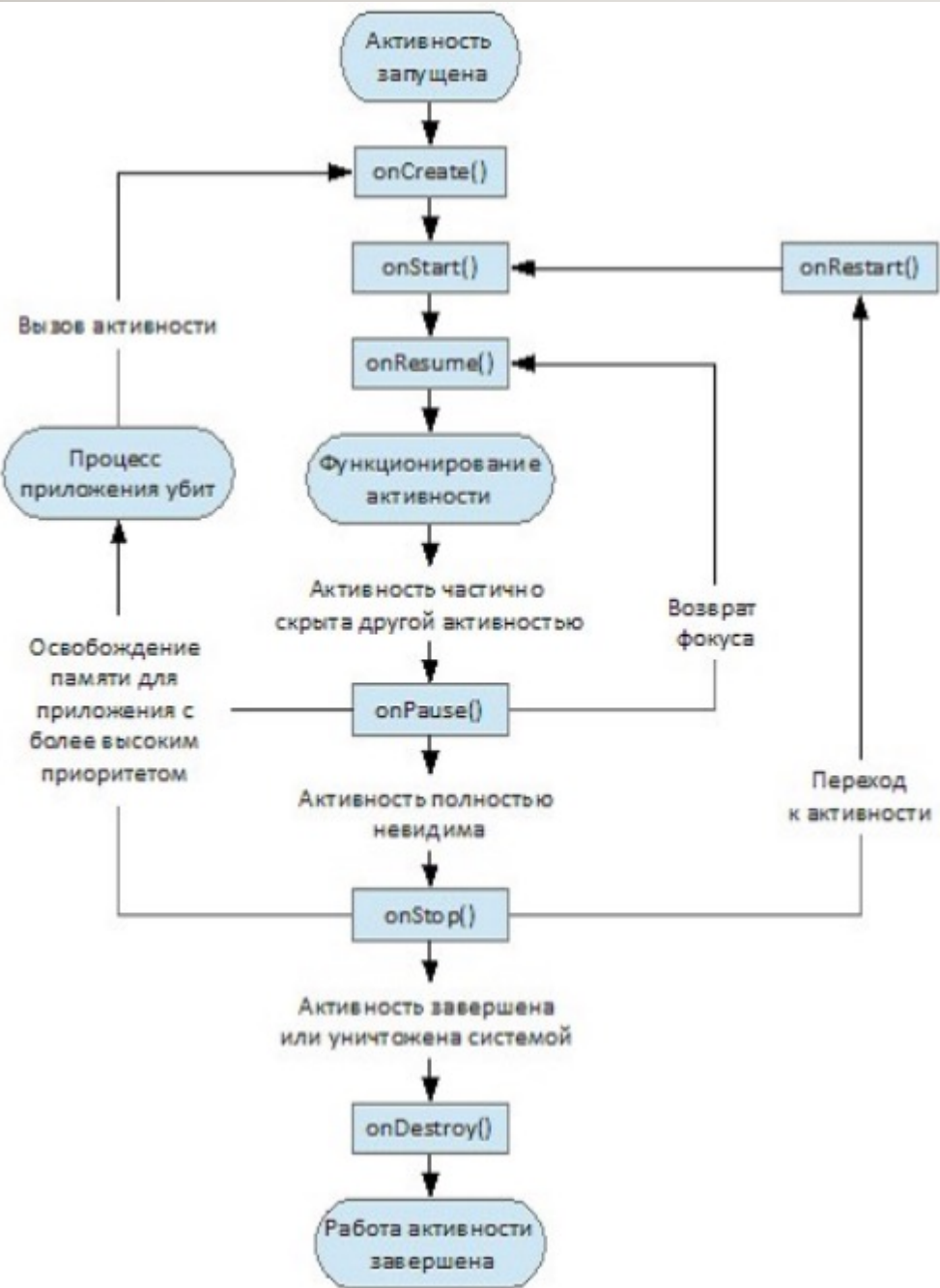
Архитектура *Android* приложений основана на идее многократного использования компонентов, которые являются основными строительными блоками. Каждый компонент является отдельной сущностью и помогает определить общее поведение приложения.

Можно выделить **четыре** различных **типа компонентов**, каждый тип служит для достижения определенной цели и имеет свой особый жизненный цикл, который определяет способы создания и разрушения соответствующего компонента.

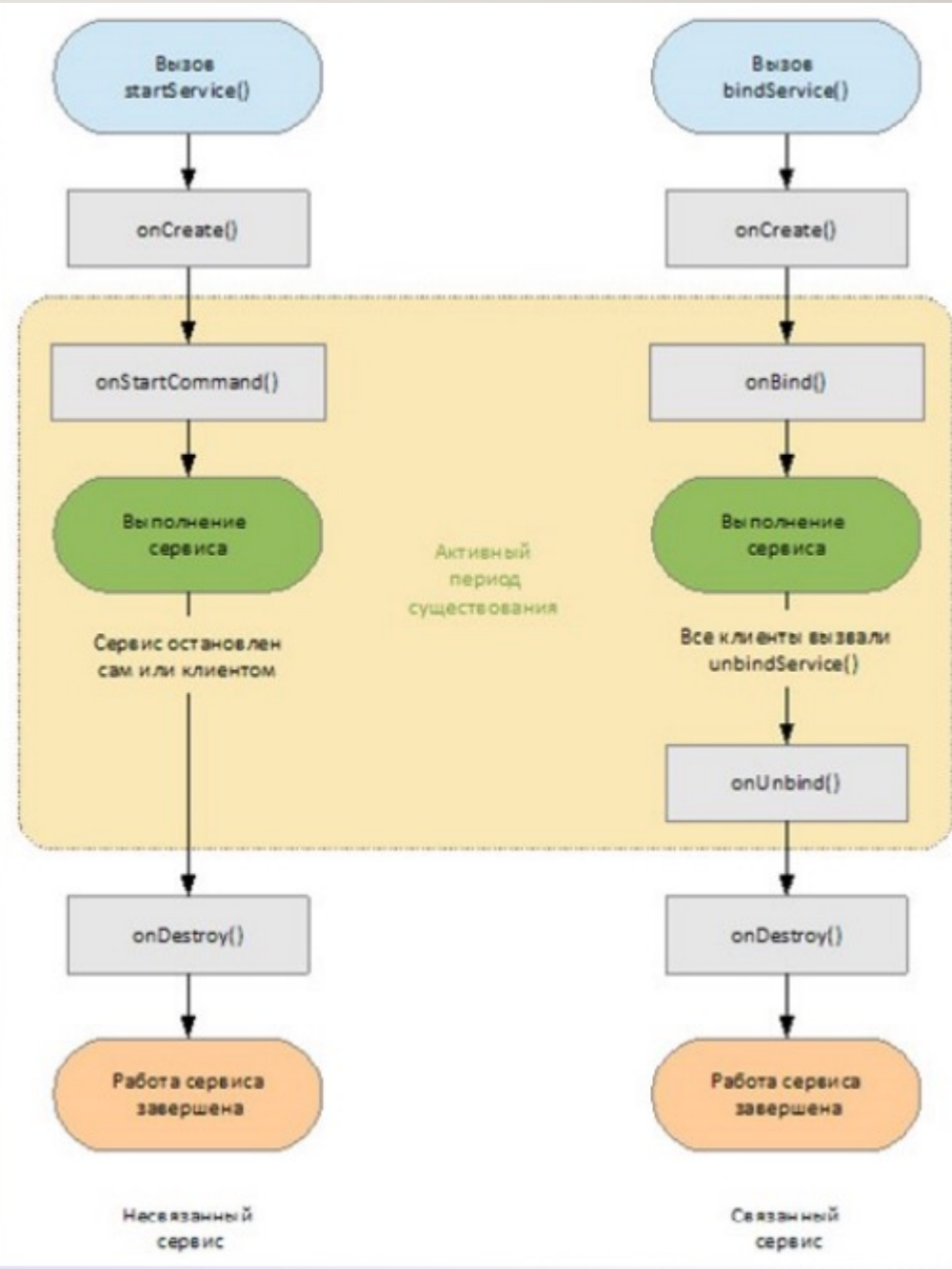


РАЗРАБОТКА

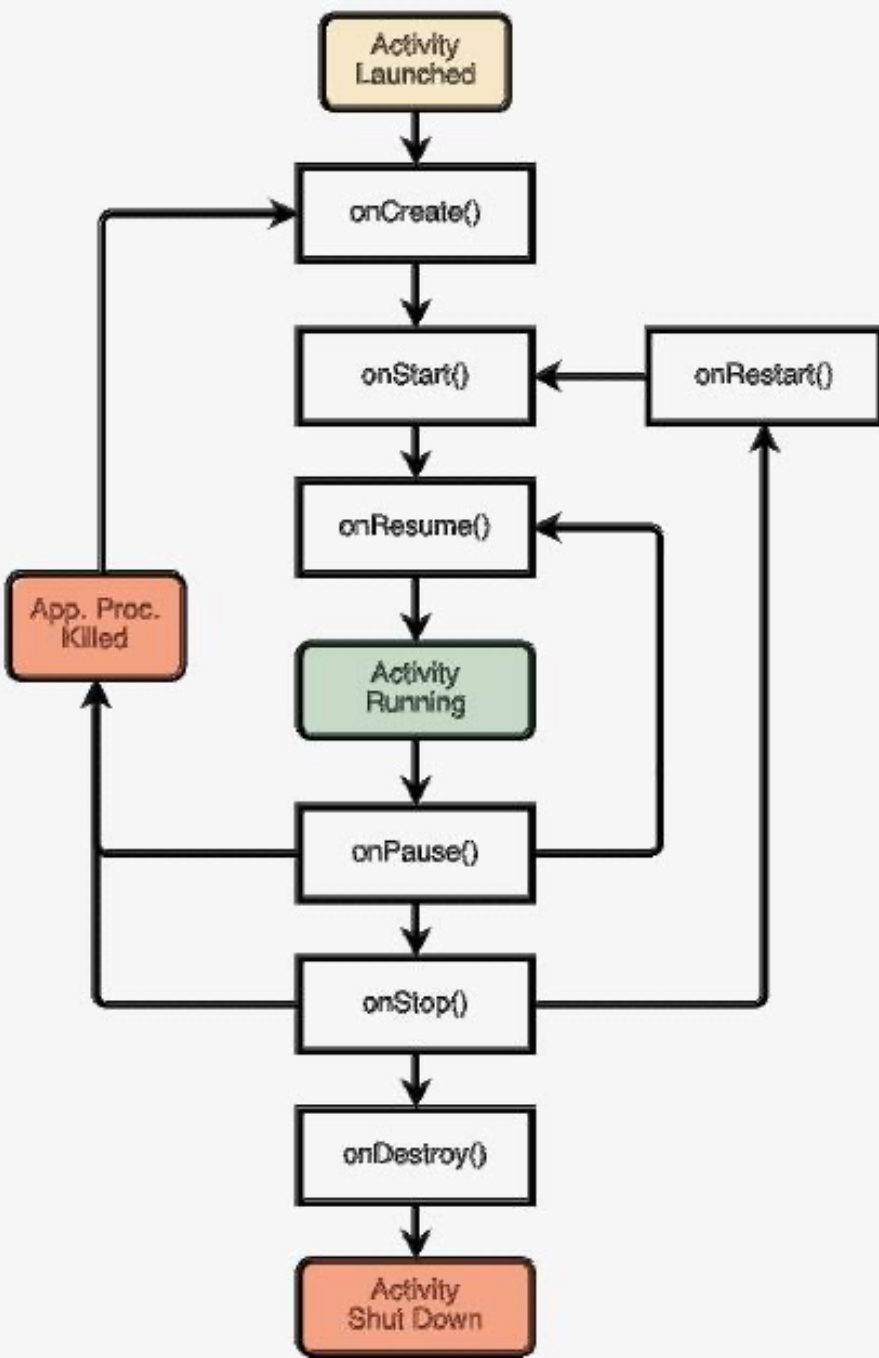
Активности (Activities). Активность - это видимая часть приложения (экран, окно, форма), отвечает за отображение графического интерфейса пользователя. При этом приложение может иметь несколько активностей, например, в приложении, предназначенном для работы с электронной почтой, одна активность может использоваться для отображения списка новых писем, другая активность - для написания, и еще одна - для чтения писем. Несмотря на то, что для пользователя приложение представляется единым целым, все активности приложения не зависят друг от друга. В связи с этим любая из этих активностей может быть запущена из другого приложения, имеющего доступ к активностям данного приложения. Например, приложение камеры может запустить активность, создающую новые письма, чтобы отправить только что сделанную фотографию адресату, указанному пользователем.



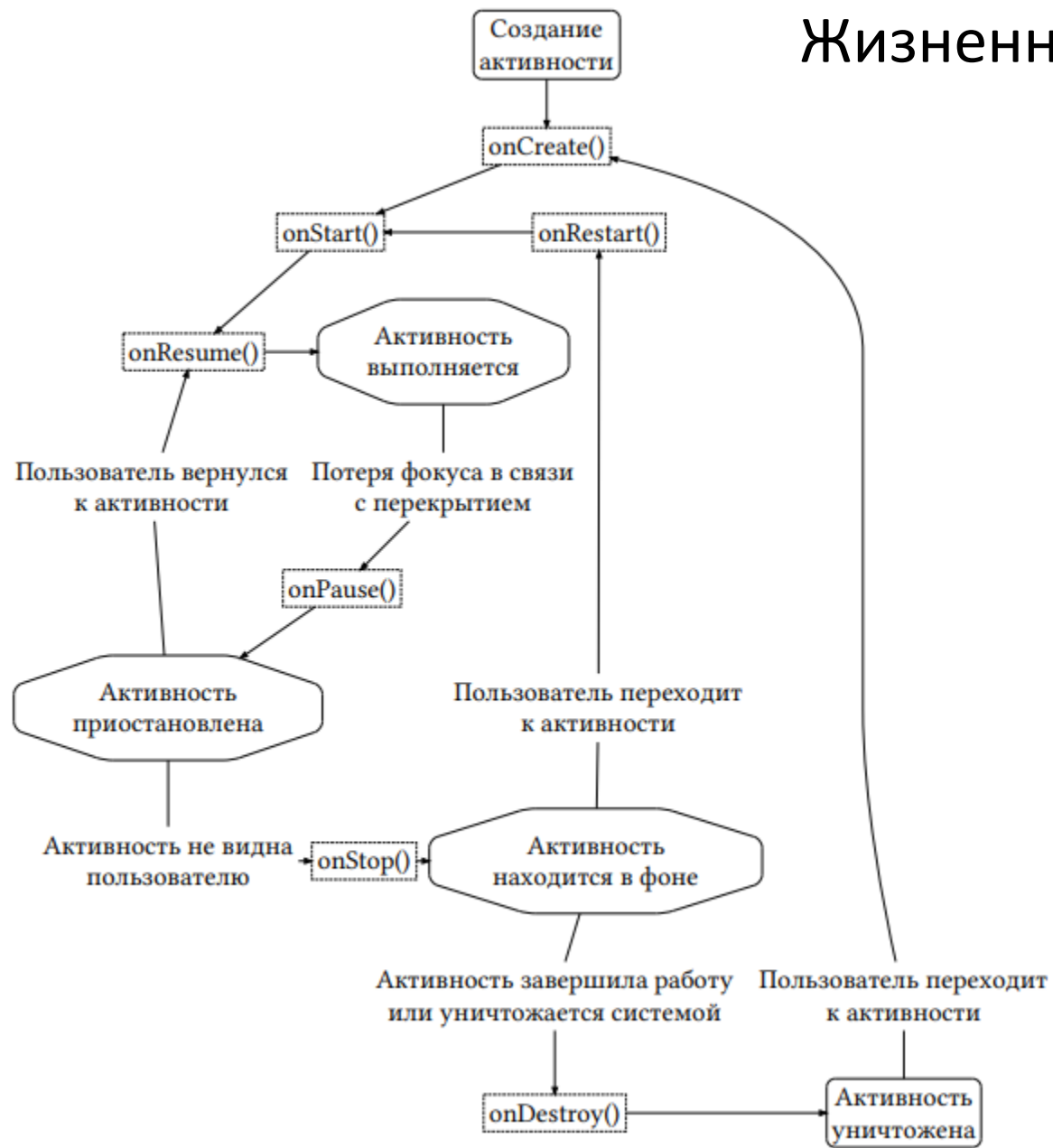
Жизненный цикл активности



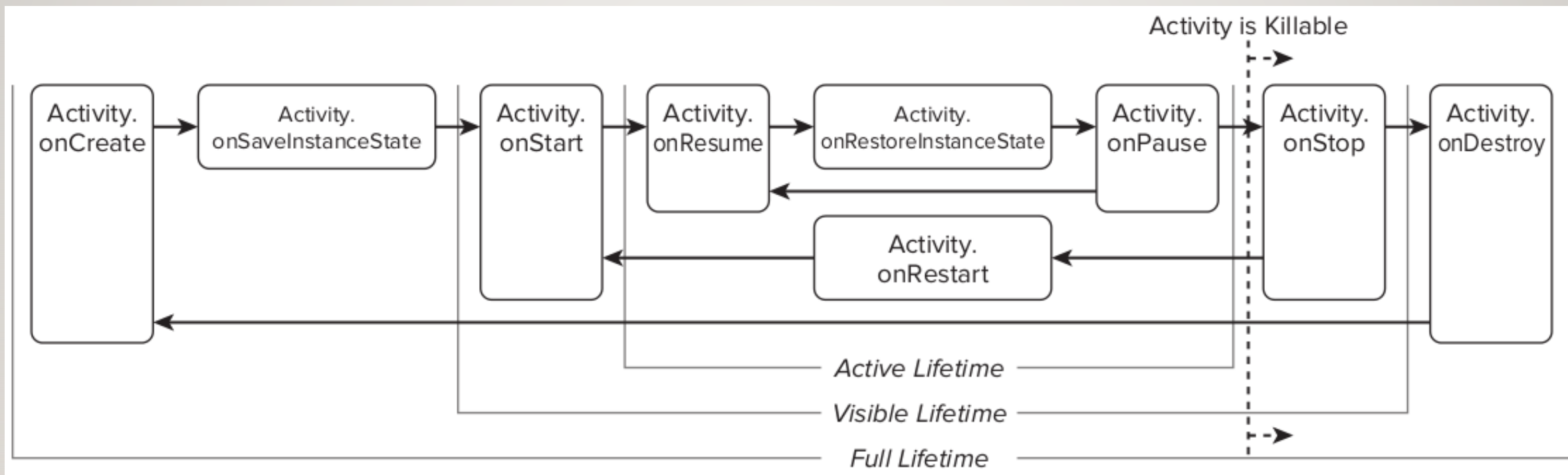
Жизненный цикл сервиса



Жизненный цикл активностей в Android

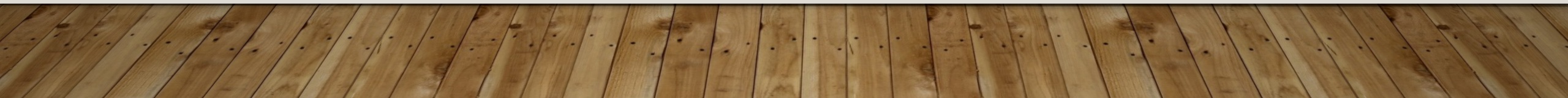


ЖИЗНЕННЫЙ ЦИКЛ ACTIVITY



РАЗРАБОТКА

Сервисы (Services). Сервис - компонент, который работает в фоновом режиме, выполняет длительные по времени операции или работу для удаленных процессов. Сервис не предоставляет пользовательского интерфейса. Например, сервис может проигрывать музыку в фоновом режиме, пока пользователь использует другое приложение, может загружать данные из сети, не блокируя взаимодействие пользователя с активностью. Сервис может быть запущен другим компонентом и после этого работать самостоятельно, а может остаться связанным с этим компонентом и взаимодействовать с ним.



РАЗРАБОТКА

Контент-провайдеры (Content providers). Контент-провайдер управляет распределенным множеством данных приложения. Данные могут храниться в файловой системе, в базе данных SQLite, в сети, в любом другом доступном для приложения месте. Контент-провайдер позволяет другим приложениям при наличии у них соответствующих прав делать запросы или даже менять данные. Например, в системе Android есть контент-провайдер, который управляет информацией о контактах пользователя. В связи с этим, любое приложение с соответствующими правами может сделать запрос на чтение и запись информации какого-либо контакта. Контент-провайдер может быть также полезен для чтения и записи приватных данных приложения, не предназначенных для доступа извне.

РАЗРАБОТКА

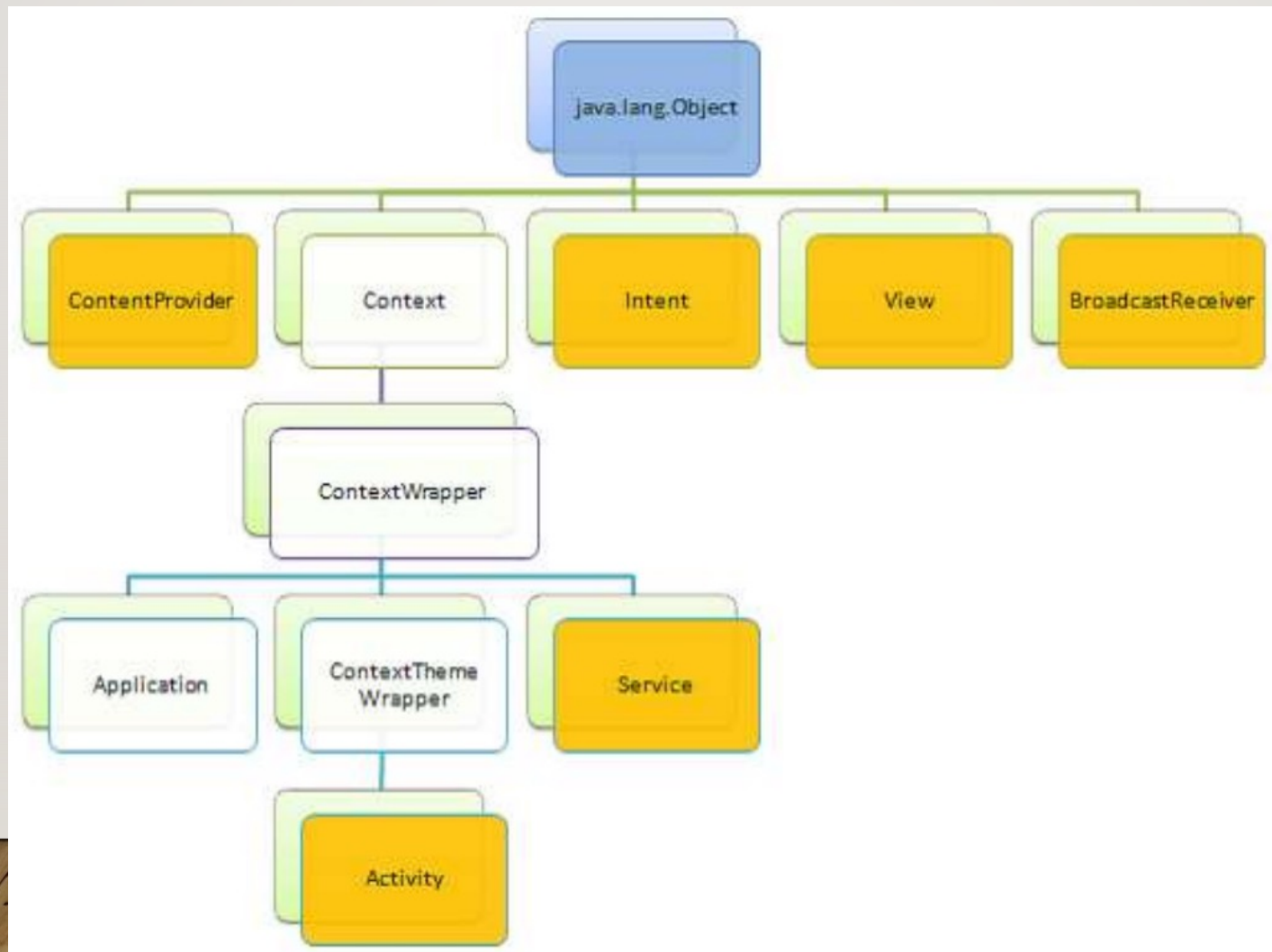
Приемники широковещательных сообщений (Broadcast Receivers).

Приемник - компонент, который реагирует на широковещательные извещения. Большинство таких извещений порождаются системой, например, извещение о том, что экран отключился или низкий заряд батареи. Приложения также могут инициировать широковещание, например, разослать другим приложениям сообщение о том, что некоторые данные загружены и доступны для использования. Хотя приемники не отображают пользовательского интерфейса, они могут создавать уведомление на панели состояний, чтобы предупредить пользователя о появлении сообщения. Такой приемник служит проводником к другим компонентам и предназначен для выполнения небольшого объема работ, например, он может запустить соответствующий событию сервис.



РАЗРАБОТКА

Иерархия классов Android SDK



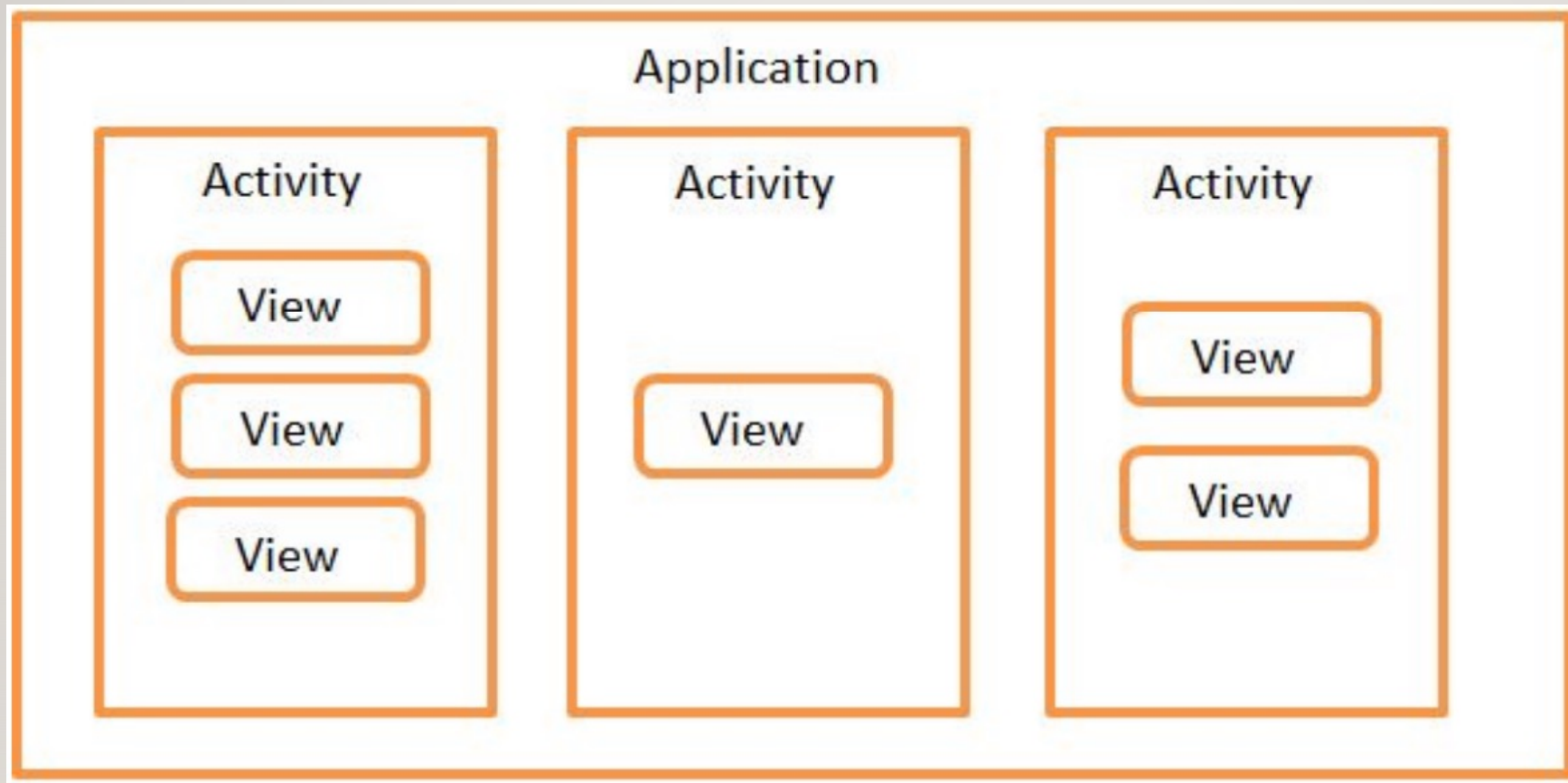
РАЗРАБОТКА

Виды приложений:

- Приложения ***переднего плана*** выполняют свои функции только, когда видимы на экране, в противном же случае их выполнение приостанавливается. Такими приложениями являются, например, игры, текстовые редакторы, видеопроигрыватель
- ***Фоновые приложения*** после настройки не предполагают взаимодействия с пользователем, большую часть времени находятся и работают в скрытом состоянии. Примерами таких приложений могут служить, службы экранирования звонков, SMS автоответчики.
- ***Смешанные приложения*** большую часть времени работают в фоновом режиме, однако допускают взаимодействие с пользователем и после настройки.
- ***Виджеты*** - небольшие приложения, отображаемые в виде графического объекта на рабочем столе. Примерами могут служить, приложения для отображения динамической информации, такой как заряд батареи, прогноз погоды, дата и время.



Элементы экрана и их свойства



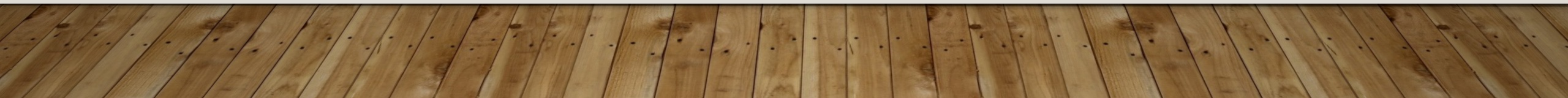
Если проводить аналогию с **Windows**, то приложение состоит из окон, называемых **Activity**

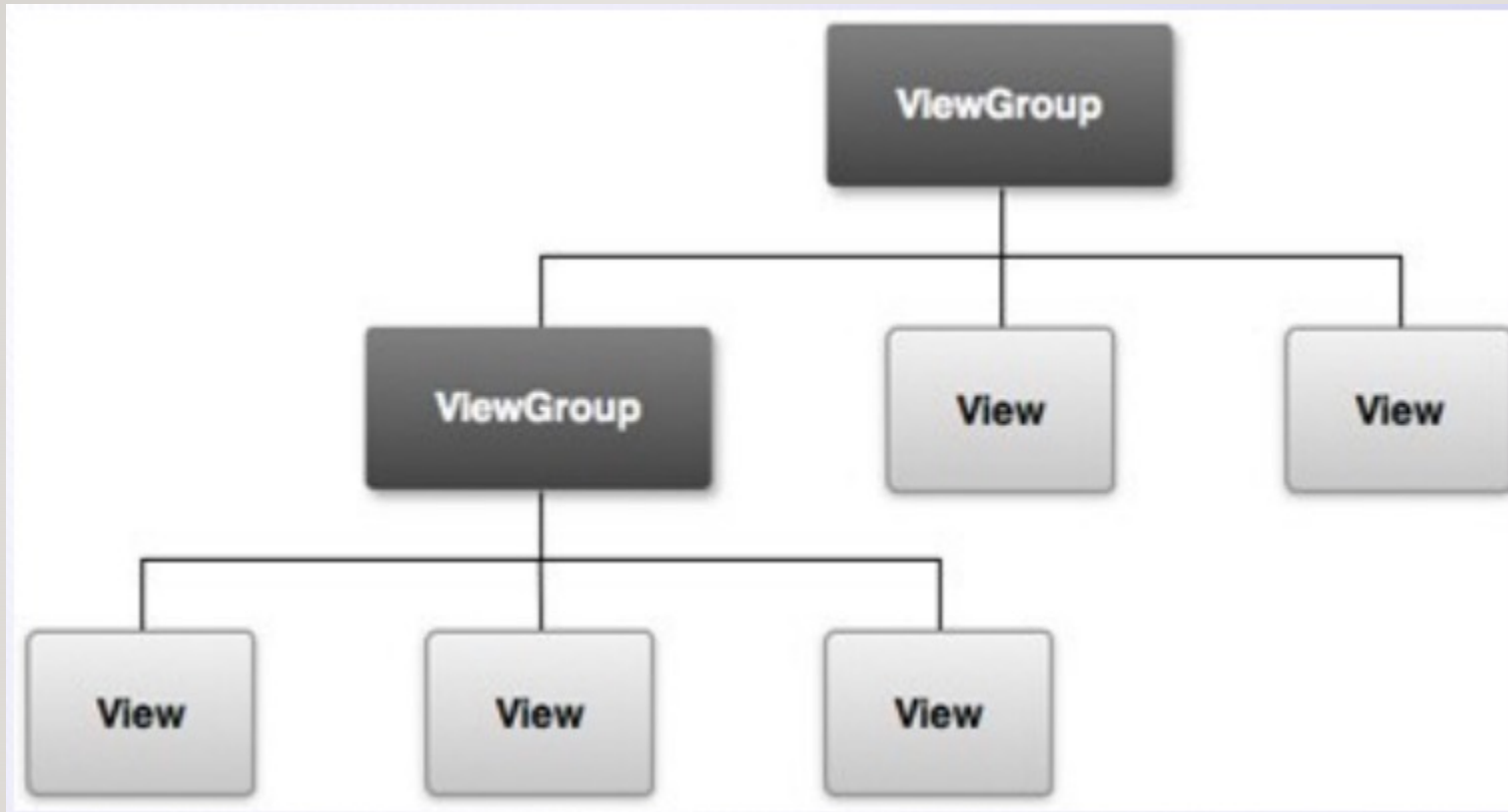
Содержимое **Activity** формируется из различных компонентов, называемых **View**.

Самые распространенные **View** - это кнопка, поле ввода, чекбокс и т.д.

Элементы экрана и их свойства

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 ...)





Иерархия компонентов, определяющая компоновку интерфейса пользователя

Элементы экрана и их свойства

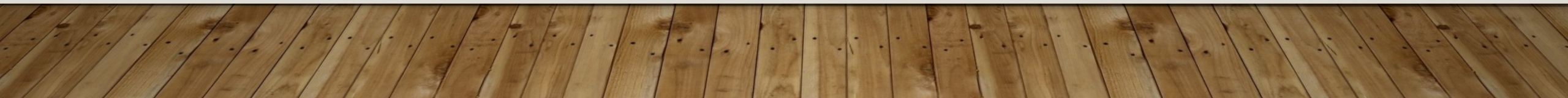
Виды ***Layouts***. Ключевые отличия и свойства.

LinearLayout – отображает ***View***-элементы в виде одной строки (если он ***Horizontal***) или одного столбца (если он ***Vertical***).

TableLayout – отображает элементы в виде таблицы, по строкам и столбцам.

RelativeLayout – для каждого элемента настраивается его положение относительно других элементов.

AbsoluteLayout – для каждого элемента указывается явная позиция на экране в системе координат (x,y)



Элементы экрана и их свойства

Layout параметры для **View**-элементов.

Layout width и **Layout height**

Используются следующие единицы измерения (ЕИ):

dp или **dip** - Density-independent Pixels. Абстрактная ЕИ, позволяющая приложениям выглядеть одинаково на различных экранах и разрешениях.

sp - Scale-independent Pixels. То же, что и **dp**, только используется для размеров шрифта в View элементах

pt - 1/72 дюйма, определяется по физическому размеру экрана. Эта ЕИ из типографии.

px — пиксел, не рекомендуется использовать т.к. на разных экранах приложение будет выглядеть по-разному.

mm — миллиметр, определяется по физическому размеру экрана

in — дюйм, определяется по физическому размеру экрана

Кол-во пикселей в одном дюйме называется **dpi (dot per inch)**.

Элементы экрана и их свойства

КОНСТАНТЫ

match_parent (fill_parent) – означает, что элемент займет всю доступную ему в родительском элементе ширину/ высоту.

wrap_content – ширина/высота элемента будет определяться его содержимым

weight – вес. для нескольких элементов свободное пространство распределяется между элементами пропорционально их ***weight***-значениям.

Layout gravity

layout_gravity аналогичен выравниванию из Word или Excel

Отступ слева и сверху. ***margin left = 10 dp, margin top = 20 dp***

