



Desenvolvimento para dispositivos móveis

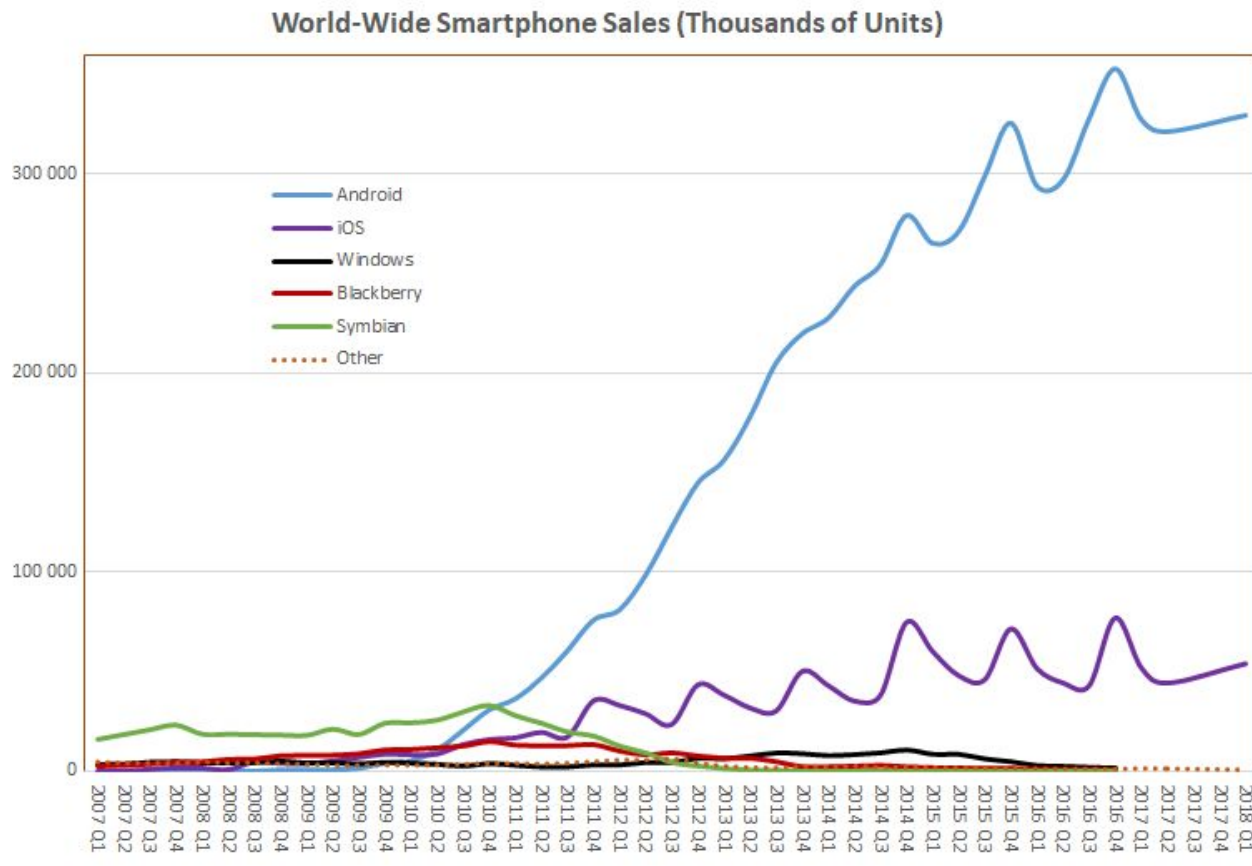
Desenvolvimento para Android – Conceitos Fundamentais

Professor Msc. Fabio Pereira da Silva
E-mail: fabio.pereira@faculdadeimpacta.com.br

Android

- » Sistema operacional da Open Handset Alliance
(<http://www.openhandsetalliance.com>)
 - – Liderado pelo Google
 - – Motorola, LG, Samsung, Sony, HTC, Toshiba, Telefonica
 - – Mais de 47 membros, entre provedores, fabricantes de hardware e fabricantes de software
- » Baseado em Linux
 - – Mesmo kernel de SO como Ubuntu, Debian e outros
 - – Android é responsável por gerenciar memória e processos

Por que desenvolver para mobile?



Fonte:
https://upload.wikimedia.org/wikipedia/commons/8/83/World_Wide_Smartphone_Sales.png
 By Efa2 - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=35496127>

Android Studio

- Android Studio é o ambiente de desenvolvimento de aplicativos para Android do Google
 - Utiliza linguagem Java ou Kotlin
 - Contém a IDE e o Android SDK
- Download
 - <https://developer.android.com/studio/index.html>

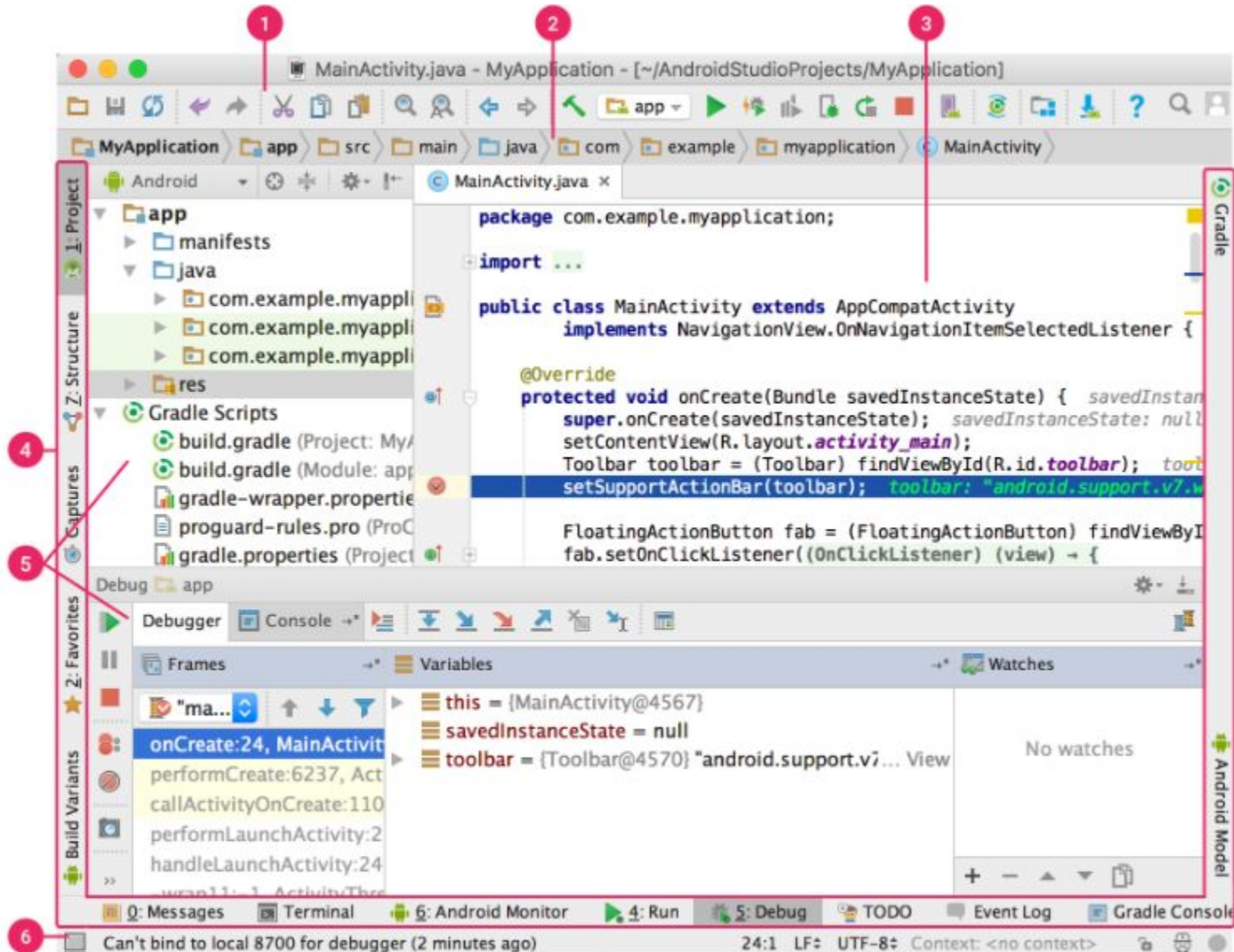
Android Studio

- Versão e Configuração utilizada em aula:
 - Android Studio: 3.5
 - Kotlin: 1.3.41
 - API Android: 28
 - Linguagem: Kotlin

Android Studio

Recurso	Descrição
1 – Barra de ferramentas	Permite realizar diversas ações, inclusive executar apps e inicializar ferramentas do Android.
2- Barra de navegação	Ajuda a navegar pelo projeto e a abrir arquivos para edição. Ela oferece uma visualização mais compacta da estrutura visível na janela Project .
3- Janela do Editor	É onde você cria e modifica o código. Dependendo do tipo de arquivo atual, o editor pode mudar. Por exemplo, ao visualizar um arquivo de layout, o editor abre o Layout Editor.
4- Barra de janela de ferramentas	Fica fora da janela do ambiente de desenvolvimento integrado e contém os botões que permitem expandir ou recolher as janelas de cada ferramenta.
5- Janela de ferramentas	permitem acessar tarefas específicas, como gerenciamento de projetos, pesquisa e controle de versões, entre outras. As janelas podem ser expandidas e recolhidas.
6 – Barra de status	Exibe o status do projeto e do próprio ambiente de desenvolvimento integrado, bem como todos os avisos ou mensagens.

Android Studio

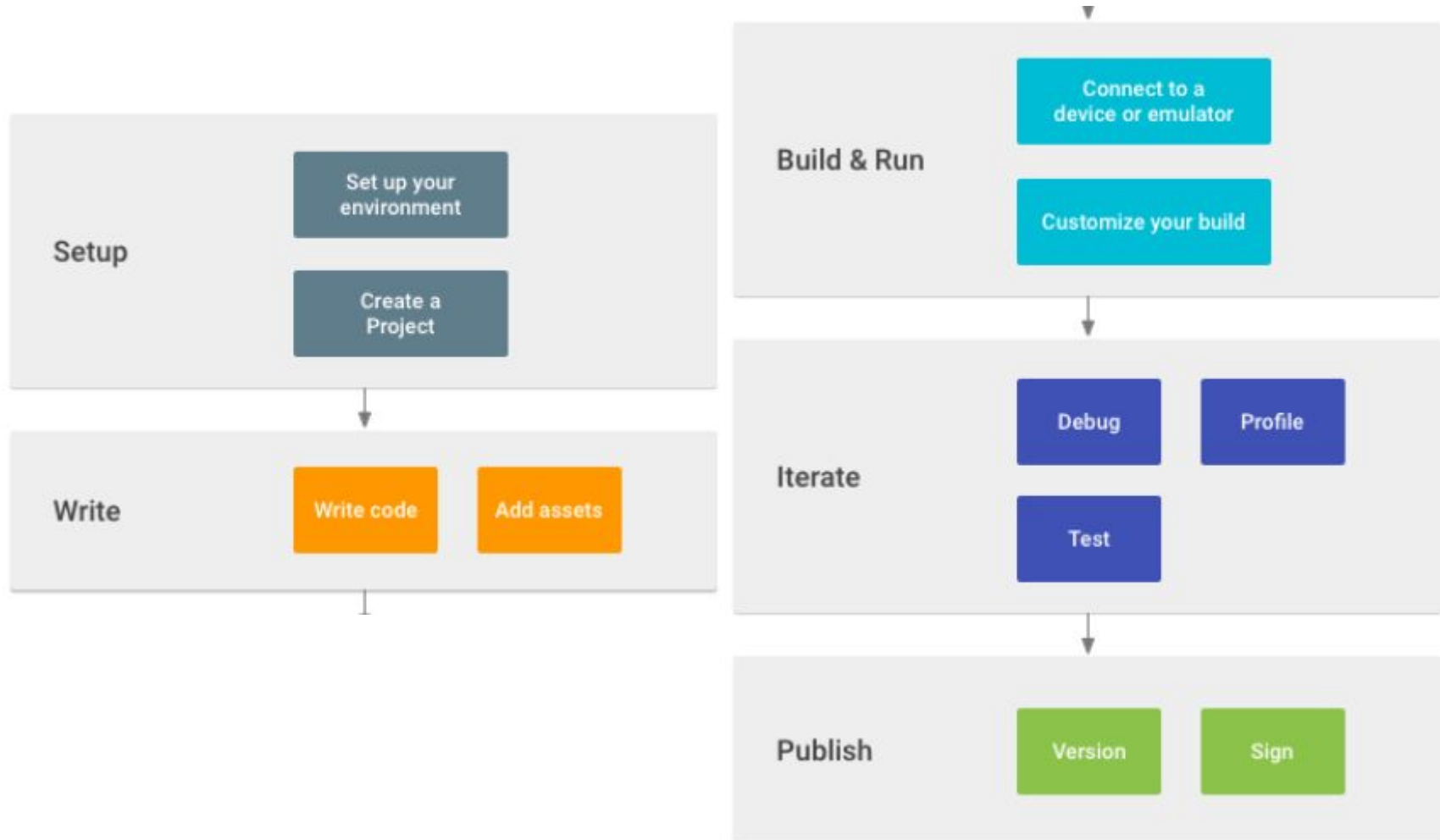


Componentes de uma aplicação Android

- Desenvolver aplicações para Android significa compor uma série de componentes para que o objetivo final da aplicação seja atingido.

Componente	Descrição
Activities	É representado por uma tela na aplicação. Possui interface de usuário, composta por Views, Componentes Gráficos e Eventos.
Services	Código sem interfaces de usuário que executam em background. Possuem ciclo de vida próprio.
Broadcast Receivers	Trata a reação de evento externo, sendo um mecanismo de alerta.
Content Providers	Permite compartilhar dados entre aplicativos. Armazenar e recuperar dados em um repositório.

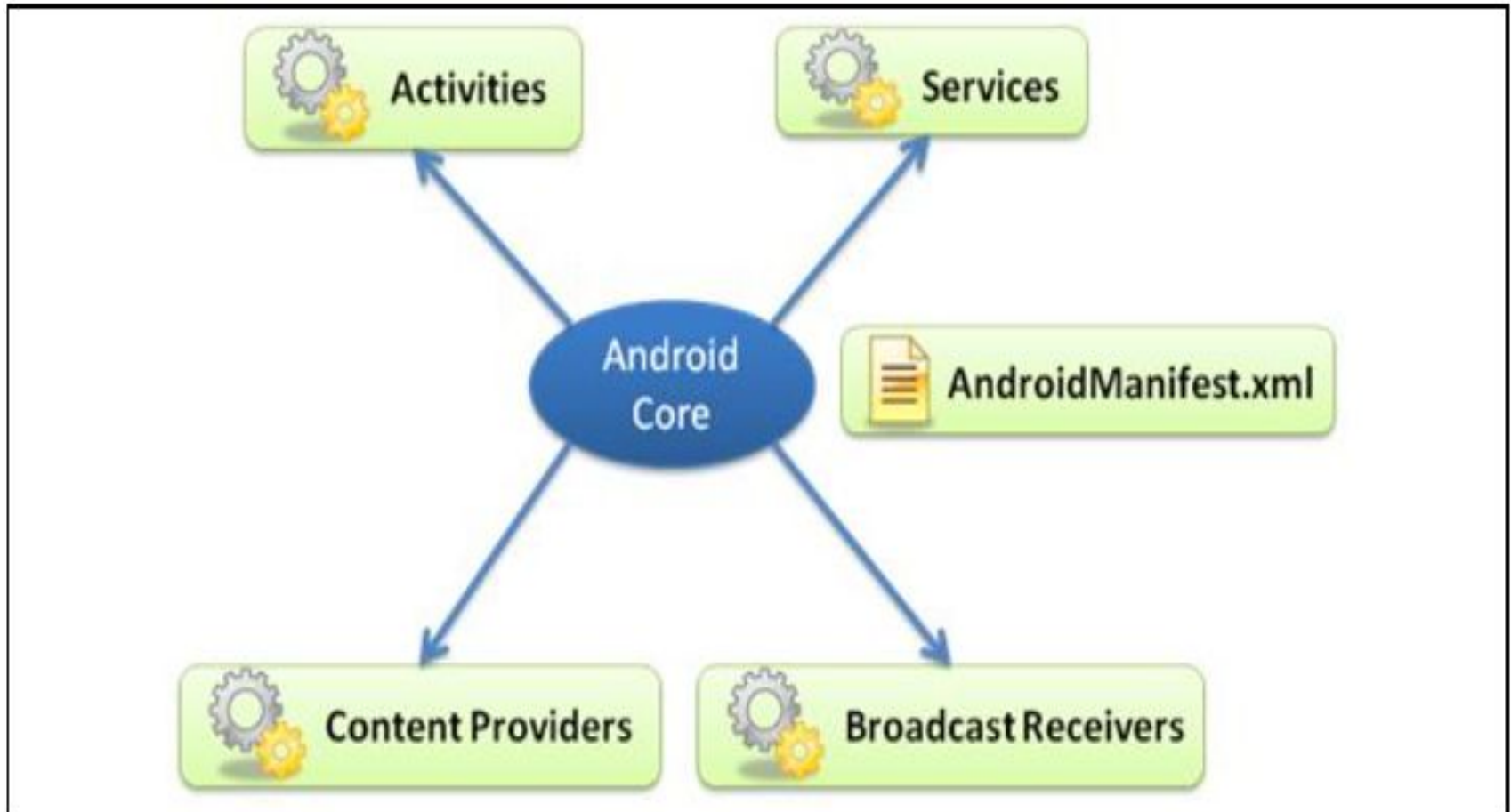
Fluxo de trabalho



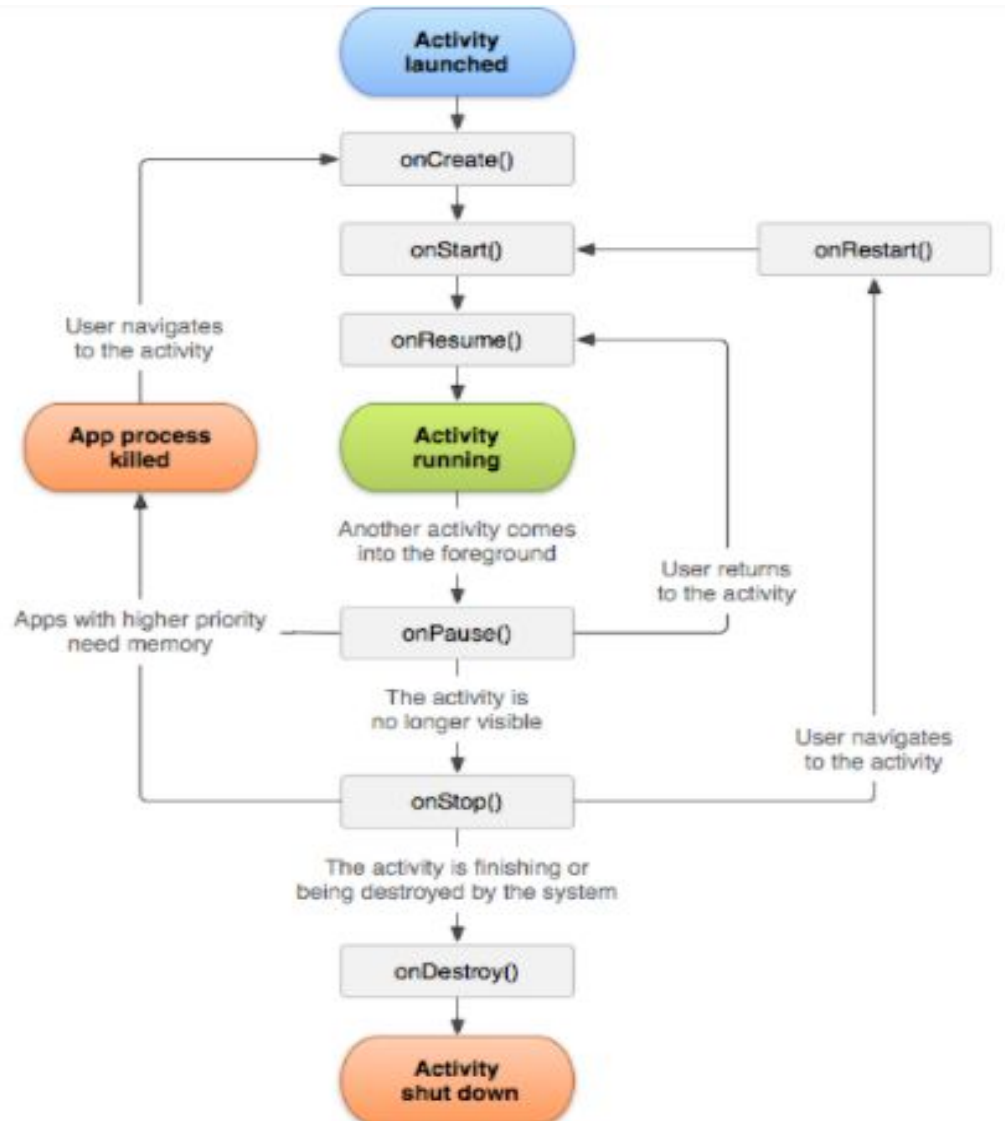
Fluxo de trabalho

Etapa	Descrição
Configurar seu espaço de trabalho	Instalação do Android Studio e criação do projeto
Programar o app	O Android Studio inclui uma variedade de ferramentas e inteligência para ajudar você a trabalhar mais rapidamente, programar códigos de qualidade, projetar uma IU e criar recursos para diferentes tipos de dispositivo.
Depurar, criar perfil e testar	Esta é a fase iterativa, em que você continua a programar seu app, mas com foco na eliminação de bugs e otimização do desempenho.
Publicar	Quando estiver tudo pronto para liberar seu app para os usuários, considere alguns outros aspectos, como o controle de versões do app e a assinatura dele com uma chave.

Componentes de uma aplicação Android



Ciclo de vida de uma atividade

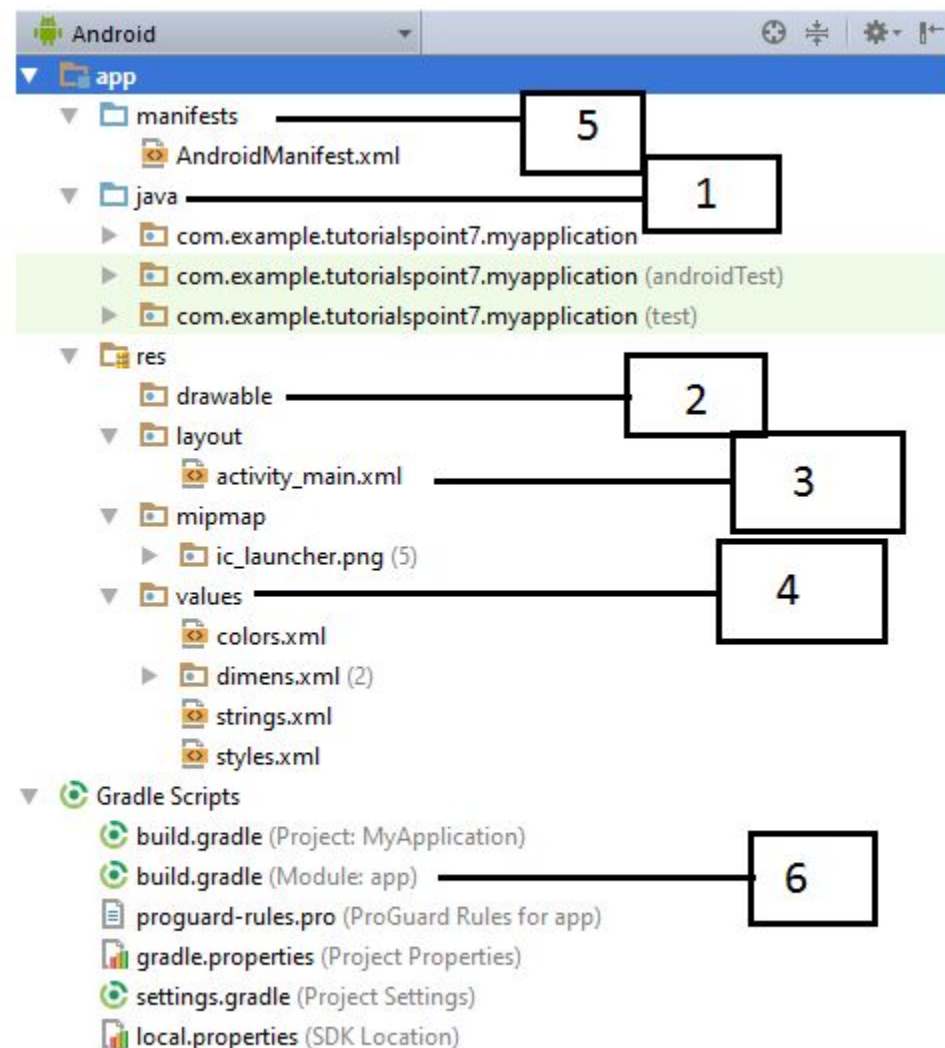


Arquitetura do Android

- » A arquitetura do sistema operacional Android é dividida em camadas.
- » Cada parte é responsável por gerenciar seus respectivos processos.

Camadas	Descrição
Camada de aplicações	Onde encontram-se todos os aplicativos que são executados sobre o sistema operacional.
Camada de bibliotecas	Possui as bibliotecas C e C++ que são utilizadas pelo sistema.
Camada de Runtime	Camada onde é instanciada a máquina virtual Dalvik para gerenciamento do desempenho.
Camada de kernel do Linux	O núcleo do sistema operacional Android é derivado do Kernel 2.6 do Linux herdando várias características da plataforma.

Estrutura de uma aplicação Android



Item	Descrição
1	Contém os arquivos fontes do projeto.
2	Objetos projetos para a tela.
3	Layouts da aplicação.
4	Outros arquivos XML, com uma coleção de recursos para a aplicação.
5	Descreve as características fundamentais do aplicativo e define cada um de seus componentes.
6	Gerado automaticamente, contém os arquivos de configuração e build.

Informações úteis

- Comando para link do projeto e resolução do problema de caracteres ASCII no Windows.
- Realizando uma cópia de todos os arquivos do diretório com caracteres especiais para um outro diretório.
- Todas as alterações do projeto em C:\Android-SDK serão refletidas no segundo diretório.
- `mklink /D "C:\android-sdk" "C:\Fabio\Disciplinas Impacta\Disciplinas ministradas em 2021\2º semestre\Aula 6\"`
- Para executá-lo abra o prompt de comando como administrador.

APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window
Manager

Content
Providers

View
System

Package Manager

Telephony
Manager

Resource
Manager

Location
Manager

Notification
Manager

LIBRARIES

Surface Manager

Media
Framework

SQLite

OpenGL | ES

FreeType

WebKit

SSL

SSL

libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual
Machine

LINUX KERNEL

Display
Driver

Camera Driver

Flash Memory
Driver

Binder (IPC)
Driver

Keypad Driver

WiFi Driver

Audio
Drivers

Power
Management

Pasta manifest

- Contém apenas um arquivo:
AndroidManifest.xml
- AndroidManifest.xml é o arquivo principal do projeto que centraliza as configurações do aplicativo
 - Indica a tela e a Activity inicial do projeto
 - Indicar quais as Activities existentes
 - Nome e imagem do aplicativo
 - Nome do pacote do projeto

Pasta manifest

- Observações
 - Todas as Activities do projeto devem ser declaradas no AndroidManifest.xml
 - Apenas uma Activity deve ser configurada como principal (MAIN) e com o ícone do aplicativo (LAUCHER)
 - As outras Activities podem ter outros tipos de filtros, mas inicialmente elas devem ser declaradas apenas com o nome

```
<activity android:name=".NomeActivity" />
```

Pasta java

- Na pasta java estarão todos os módulos Kotlin ou Java do seu projeto
- Pode criar outros pacotes para organizar os arquivos, mas tente manter todos dentro do pacote principal (configurado na criação do app e declarado no `AndroidManifest.xml`)
- Quando o projeto é criado uma Activity foi criada
 - O padrão é `MainActivity`, mas pode ter qualquer nome

Pasta java - MainActivity

- O importante é que esta classe, e qualquer outra Activity do projeto, seja filha da classe `android.app.Activity`
 - No exemplo, está herdando `AppCompatActivity`
 - `AppCompatActivity` é filha de `android.app.Activity`
- MainActivity está configurada para ser a Activity inicial e ser iniciada quando clicar no ícone do aplicativo
 - Lembre-se de da configuração em `AndroidManifest.xml`

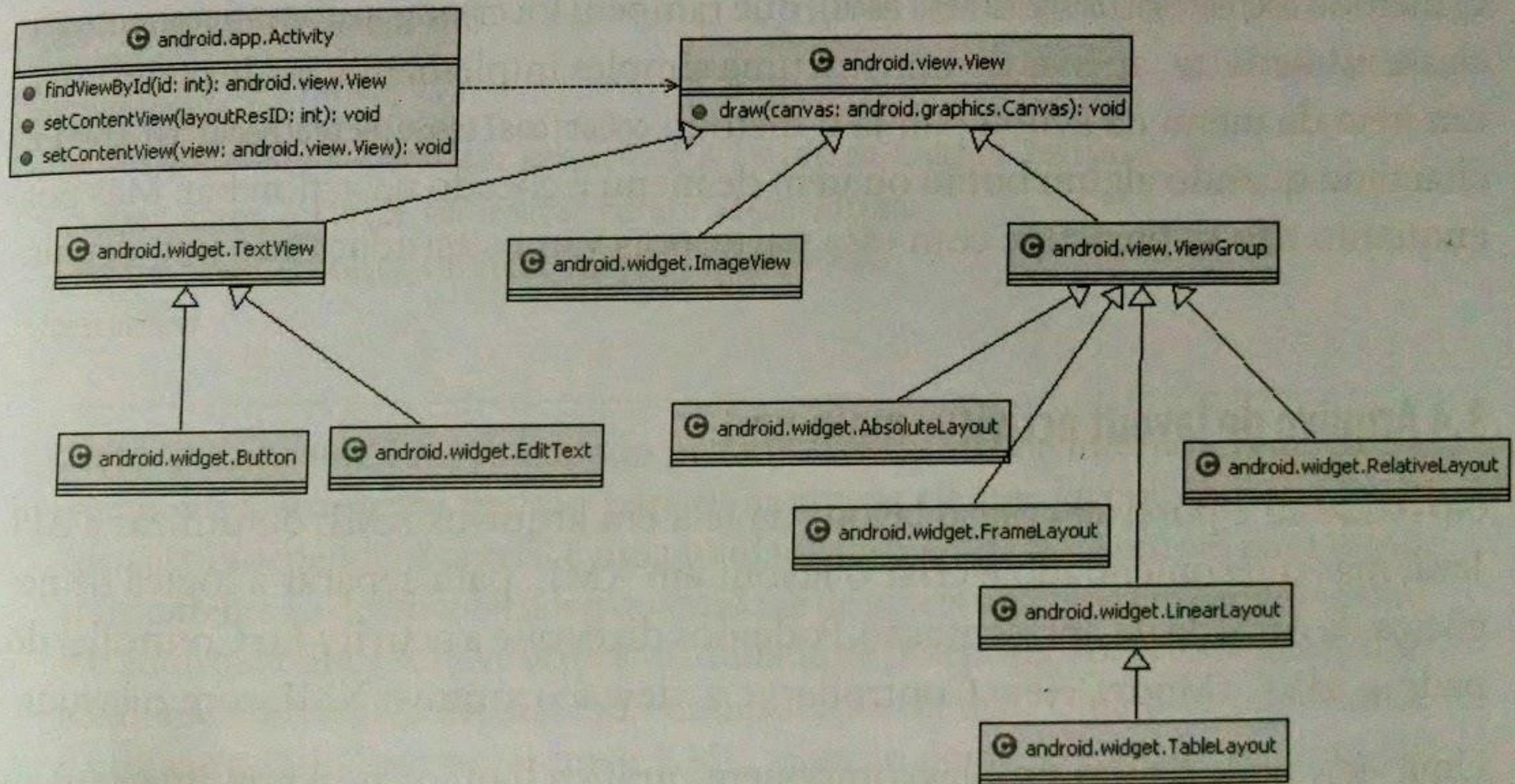
Pasta java - MainActivity

- Uma Activity representa uma tela do aplicativo
 - Controla o estado
 - Controla os eventos de tela
 - Cada tela do seu aplicativo terá uma Activity relacionada
- Para cada Activity criada, é obrigatório implementar o método onCreate(bundle)
 - Este método é chamado automaticamente quando a tela é criada

Pasta java - MainActivity

- Apesar de representar a tela, uma Activity não sabe “desenhar” os elementos da tela
 - Precisa de uma classe do tipo `android.view.View`
 - Uma View é responsável por desenhar os elementos da tela
 - Textos, campos, botões, imagens, layouts...
 - Cada elemento da tela é uma subclasse de `android.view.View`

Pasta java - MainActivity



Pasta java - MainActivity

- Para adicionar um elemento na tela relacionada a Activity, utilize o método setContentView
 - Método já implementado na classe mãe Activity
 - **Responsável por fazer a ligação entre a Activity e a View que desenha a interface gráfica**
 - Deve ser chamado sempre no método onCreate da Activity
- O setContentView pode receber como argumento
 - Uma instância de View: para construir telas diretamente pelo código Kotlin ou Java (mais dinâmico)

Pasta java - MainActivity

- No exemplo, o método `setContentView` recebe como argumento `R.layout.activity_main`
 - Constante inteira
 - Representa um recurso do aplicativo
 - Todos os recursos estarão dentro da pastas do projeto
- Neste caso, `R.layout.activity_main` representa arquivo de layout `activity_main.xml`

Pasta res

- A pasta res contém os recursos do aplicativo
- Recurso é qualquer arquivo utilizado no aplicativo que não seja o arquivo de configuração (AndroidManifest.xml) e classes kotlin ou java
 - Imagens – subpasta drawable
 - Layouts de telas – subpasta layout
 - Strings/textos – subpasta values, arquivo string.xml
 - Estilos – subpasta values, arquivo styles.xml
 - ...
- Todos os recursos são associados a uma constante inteira presente

Layouts

- Layouts: Determinam o sistema de posicionamento dos elementos da interface
- Layouts comuns
- Linear Layout
- Relative Layout
- Layouts Dinâmicos
- List View
- Grid View

Layouts

Linear Layout



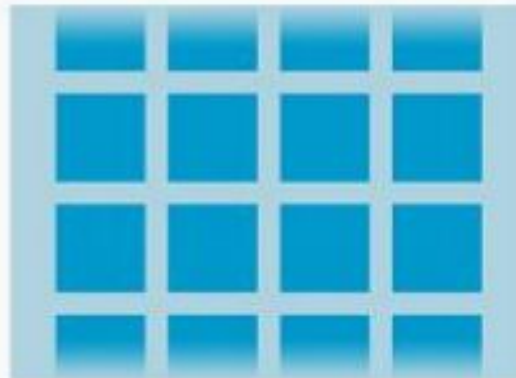
Relative Layout



List View



Grid View



Utilizando os recursos da classe R

- Qualquer recurso definido na classe R pode ser utilizado, tanto em outros recursos como nas classes Kotlin / Java.
- Dentro de arquivos XML, o padrão de uso dos recursos é sempre o mesmo

`@tipoRecurso/nome_recurso`

- Para imagens e arquivos de layout, o nome do recurso é o nome do arquivo sem a extensão; o tipo do recurso é o nome da pasta (layout, drawable)
- Para os outros, o nome do recurso está dentro do arquivo correspondente (strings.xml, styles.xml). O nome do arquivo representa o tipo de recurso

Utilizando os recursos da classe R

- Exemplos

`@layout/activity_main`

- Tipo de recurso: layout
- Nome do recurso: activity_main
- Arquivo de layout com nome activity_main.xml dentro da pasta res/layout

`@string/mensagem`

- Tipo de recurso: string
- Nome do recurso: mensagem
- Texto com o identificador mensagem dentro do arquivo strings.xml na pasta res/

Utilizando os recursos da classe R

- Nas classes Kotlin/Java, também existe um padrão de uso:

```
R.tipoRecurso.nome_recurso
```

- Para imagens e arquivos de layout, o nome do recurso é o nome do arquivo sem a extensão; o tipo do recurso é o nome da pasta (layout, drawable)
- Para os outros, o nome do recurso está dentro do arquivo correspondente (strings.xml, styles.xml). O nome do arquivo representa o tipo de recurso

Utilizando os recursos da classe R

- Exemplos

`R.layout.activity_main`

- Tipo de recurso: layout
- Nome do recurso: activity_main
- Arquivo de layout com nome activity_main.xml dentro da pasta res/layout

`R.string.mensagem`

- Tipo de recurso: string
- Nome do recurso: mensagem
- Texto com o identificador mensagem dentro do arquivo strings.xml na pasta res/

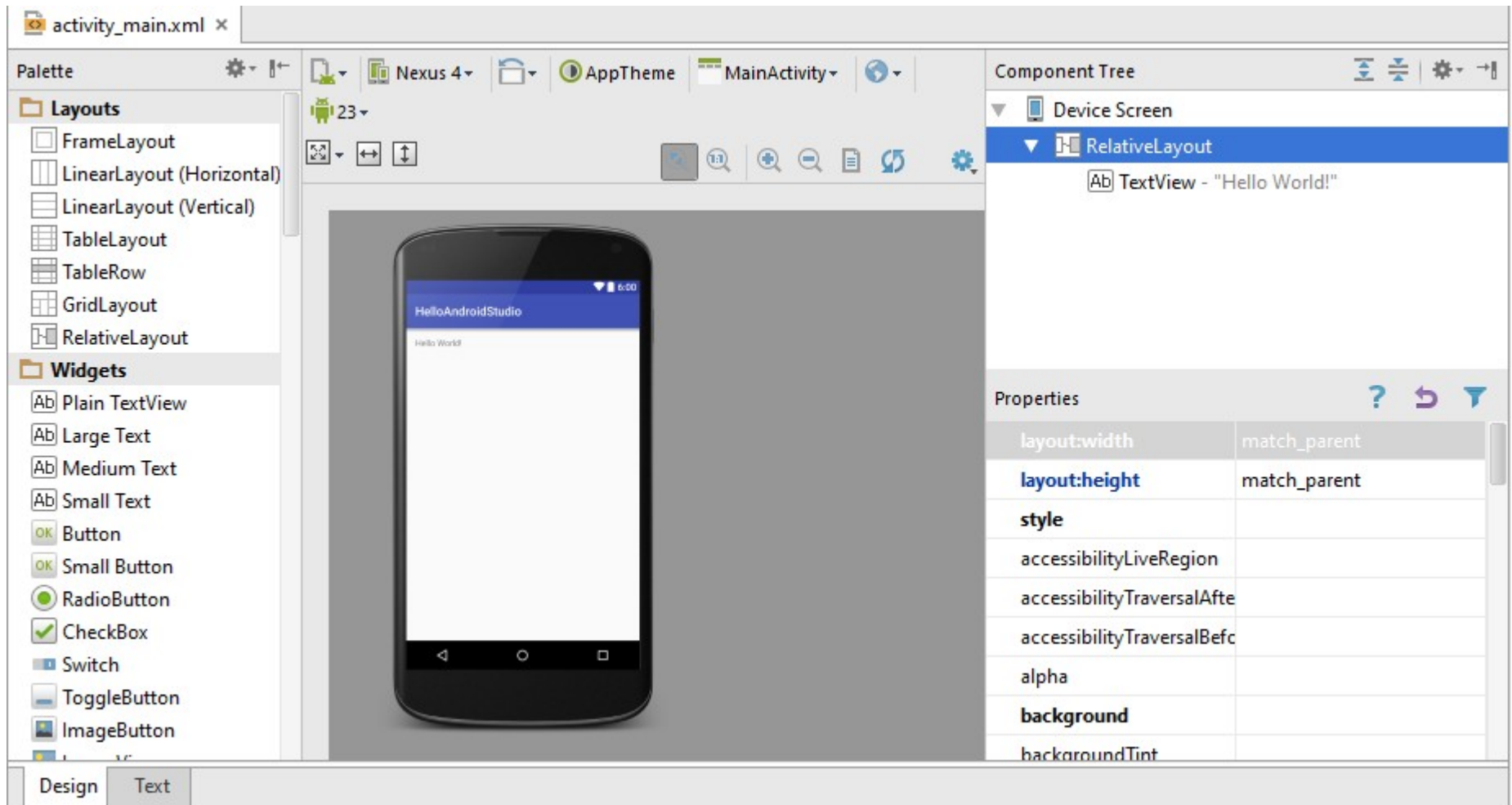
Pasta res - layout activity_main.xml

- Layouts de tela podem ser criados utilizando arquivos XML ou utilizar Kotlin/Java
 - Recomendado: XML
 - Separar lógica de negócio da apresentação
 - Activity: “Controller”; Arquivo XML de layout: “View”
- Todos os layouts devem ficar em res/layout
- Quando o projeto foi criado o arquivo activity_main.xml foi criado, para comportar a tela inicial do aplicativo

Pasta res - layout activity_main.xml

- Todos os elementos de tela são filhos de `android.view.View`
- No arquivo XML de layout, os marcadores representam a subclasse de `View`, e os atributos dos marcadores são os atributos da subclasse
- A construção da tela pode ser feita escrevendo os marcadores diretamente do código fonte ou utilizando o editor visual

Pasta res - layout activity_main.xml



Pasta res - layout activity_main.xml

activity_main.xml x

RelativeLayout

```

<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="br.com.fernandosousa.helloandroidstudio.MainActivity">


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

```

Design
Text

Preview

Nexus 4
AppTheme



Pasta res - layout activity_main.xml

- O arquivo de layout será utilizado para definir a View representada pela Activity
- O acesso na Activity é feito por uma constante inteira, como argumento do método `setContentView()`
 - `R.layout.activity_main`
- Para acessar qualquer arquivo de layout dentro de uma classe kotlin/java, utiliza-se o padrão:
 - `R.layout.nome_arquivo`

Pasta res - string.xml

- **strings.xml é um arquivo de recursos com as mensagens e textos do aplicativo**
 - Organizar os textos em um arquivo
 - Suporte a internacionalização (vários idiomas)
- Quando o aplicativo for criado, este arquivo contém o nome do aplicativo

<string

name="app_name">LMSApp</string>

- Cada <string> é um recurso, definido pelo atributo name
 - O valor está entre a abertura e o fechamento do marcador

Pasta res - string.xml

- Estes recursos podem ser utilizados em qualquer arquivo xml do projeto
 - AndroidManifest
 - Outros recursos, como layout
- Para acessar qualquer recurso de string de um arquivo XML (como layout), basta utilizar o seguinte padrão
 - @string/nome_recurso
 - Por exemplo: @string/app_name

Recursos de imagens

- As imagens utilizadas no aplicativo devem ficar res/drawable
 - No projeto do Android Studio, o ícone do aplicativo fica dentro de res/mipmap
 - Para acessar os recursos de drawable
`@drawable/nome_arquivo`
 - Para acessar os recursos de mipmap
`@mipmap/nome_arquivo`

Recursos de imagens

- Um id é um recurso assim como imagens, estilos, strings e layout, porém criados dentro dinamicamente dentro de outros recursos
 - Por isso utiliza-se @+id, indicando que é um recurso novo
- Os ids estão também na classe R, e podem ser utilizados nos códigos Kotlin e Java

Tratamento de eventos

- Os eventos de tela são vinculados tratados dentro da Activity
- A vinculação de um evento é feito utilizando o método `setOnClickListener`.
- Este método pode receber como argumento uma função anônima, que será responsável por tratar o evento
 - Ou seja, será executada quando acontecer o evento

Tratamento de eventos

- Por exemplo, mostrar uma mensagem Toast após clicar no botão
 - Repare na sintaxe lambda do Kotlin

```
botaoLogin.setOnClickListener {  
    Toast.makeText(this, "Clicou no botão", Toast.LENGTH_LONG).show()  
}
```

Tratamento de eventos

- Segunda opção: delegar para um método
 - A vantagem é que o método pode ser reutilizado para botões que tem o mesmo tratamento de evento

```

override fun onCreate(savedInstanceState: Bundle?) {
    // Código do onCreate

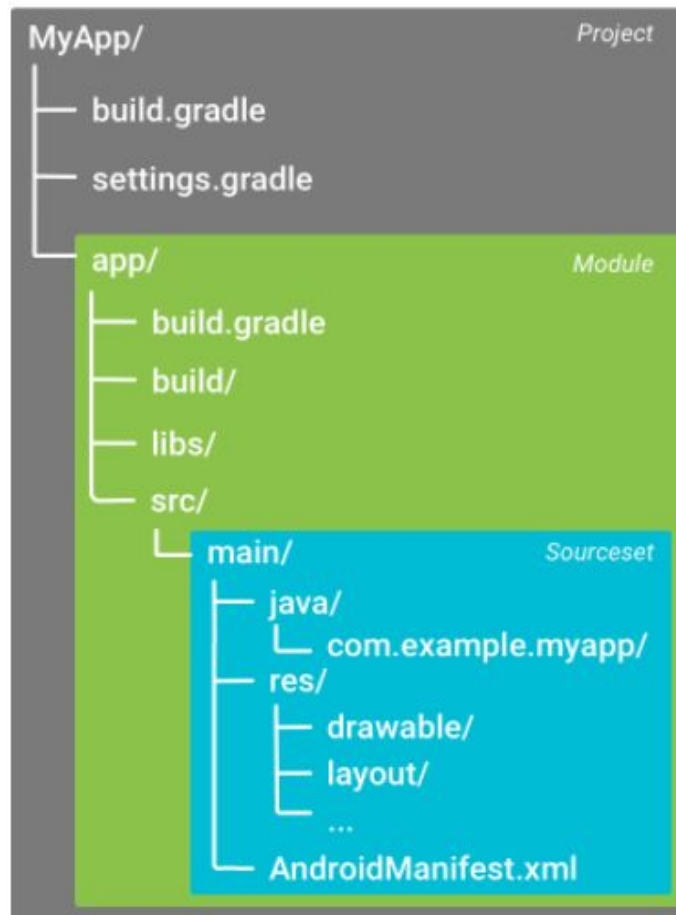
    // segunda forma: delegar para método
    botaoLogin.setOnClickListener {onClickLogin() }

}

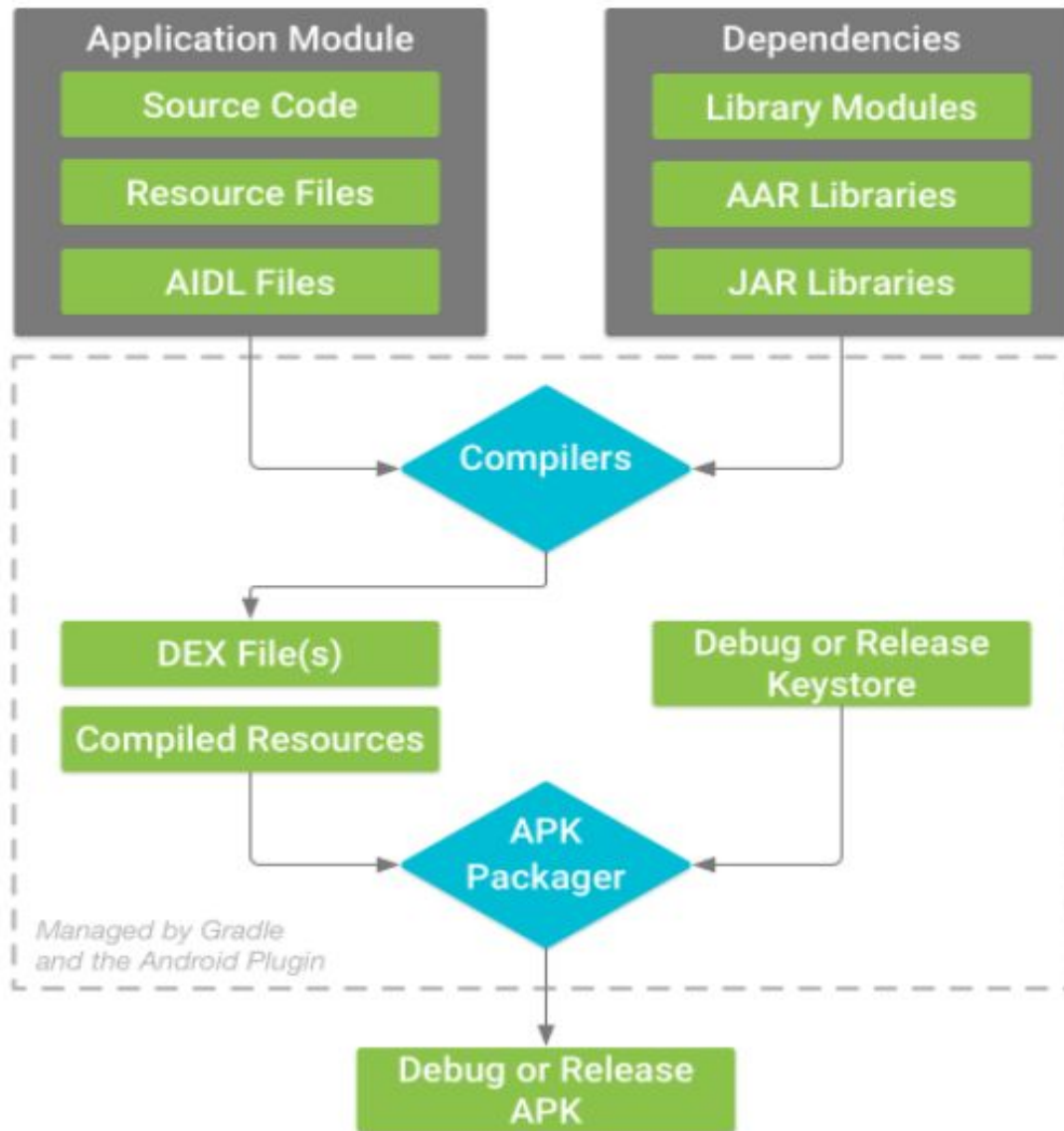
fun onClickLogin(){
    val valorUsuario = campo_usuario.text.toString()
    val valorSenha = campo_senha.text.toString()
    Toast.makeText(this, "$valorUsuario : $valorSenha",
    Toast.LENGTH_LONG).show()
}
  
```

Estrutura de uma aplicação Android

- Ao iniciar uma aplicação Android, é criada automaticamente a estrutura abaixo.



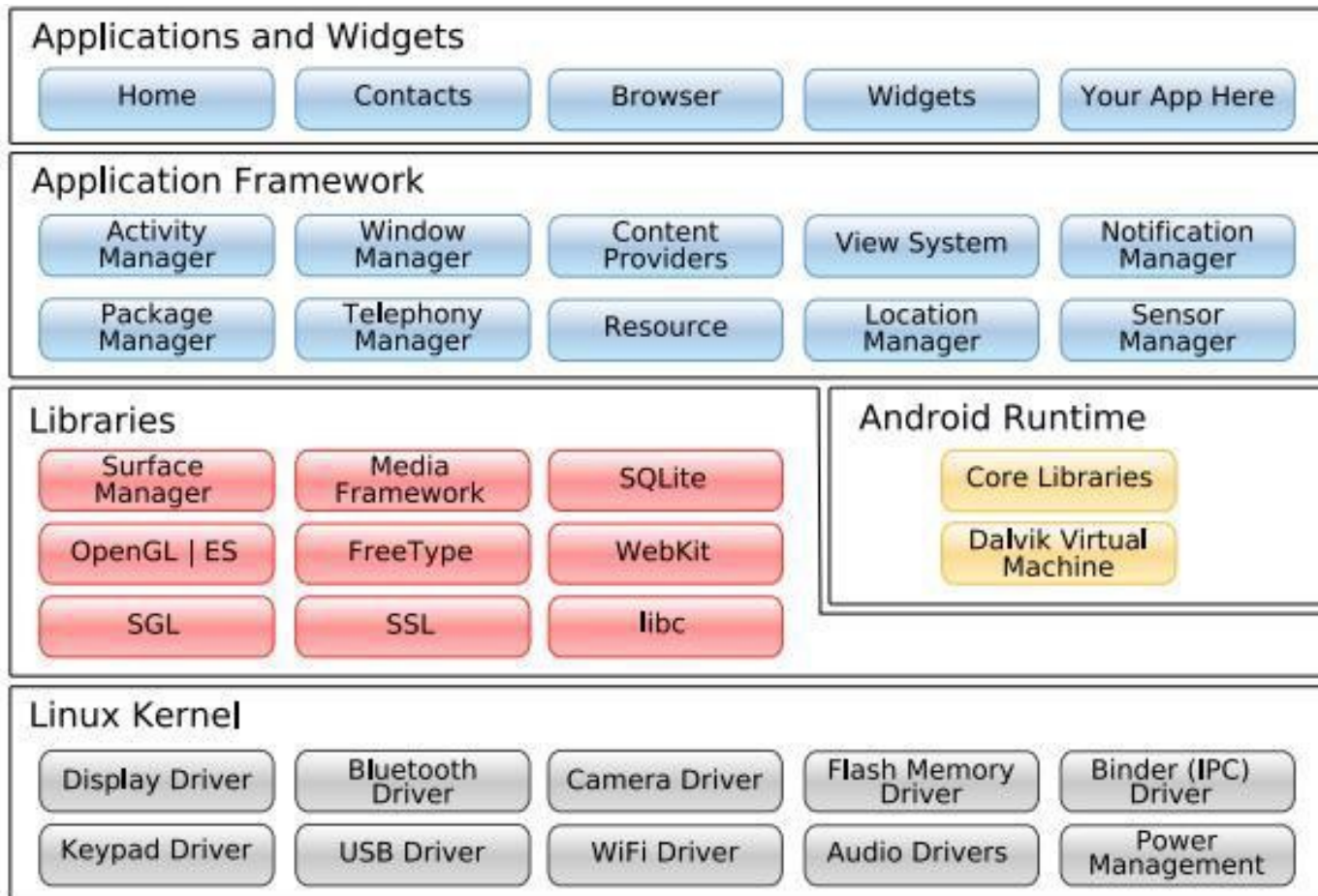
Processo de compilação



Android

- Arquitetura

a



<http://kebomix.wordpress.com/2010/08/17/android-system-architecture>

Módulo e versão

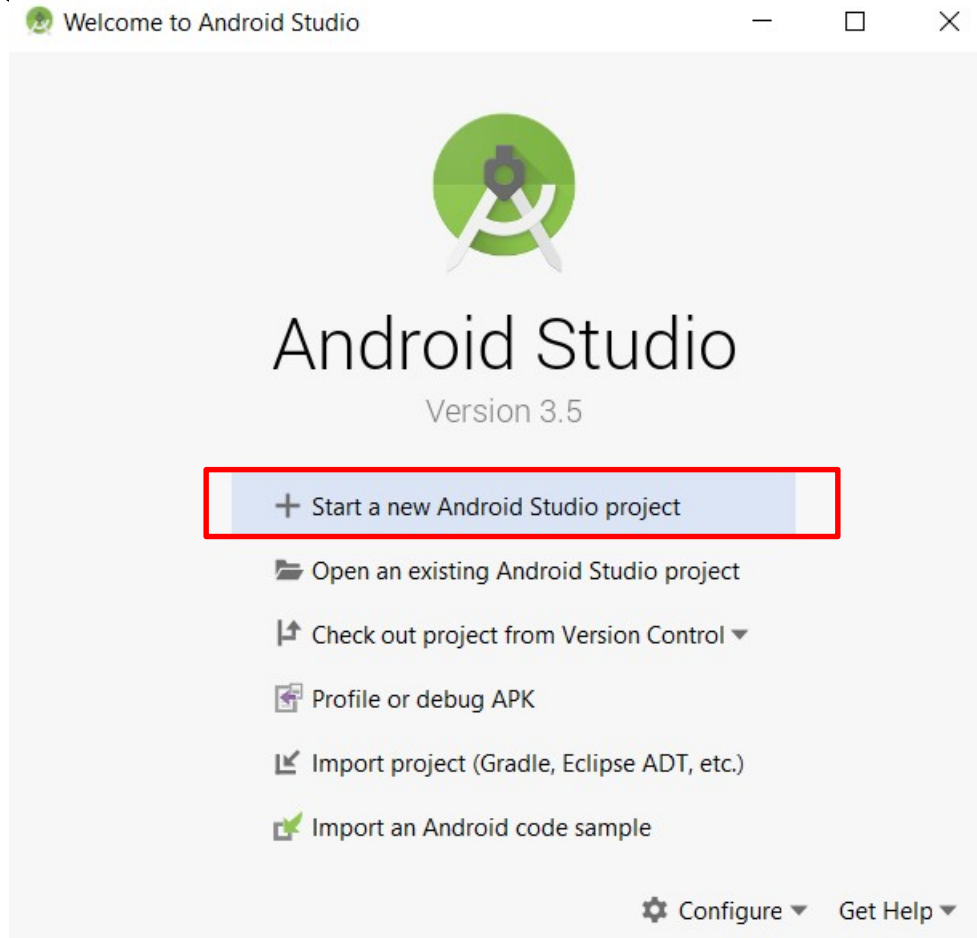
- Módulo: nome único da publicação da aplicação
- Versão: determina a partir de qual versão o projeto Android será executado

Android – Desenvolvimento (oficial)

- Linguagem Java ou Kotlin
- Qualquer computador com
 - JDK
 - Android SDK
 - Android Studio
- Totalmente customizável
- É possível substituir aplicativos nativos
 - Tela inicial, agenda, mensagens...

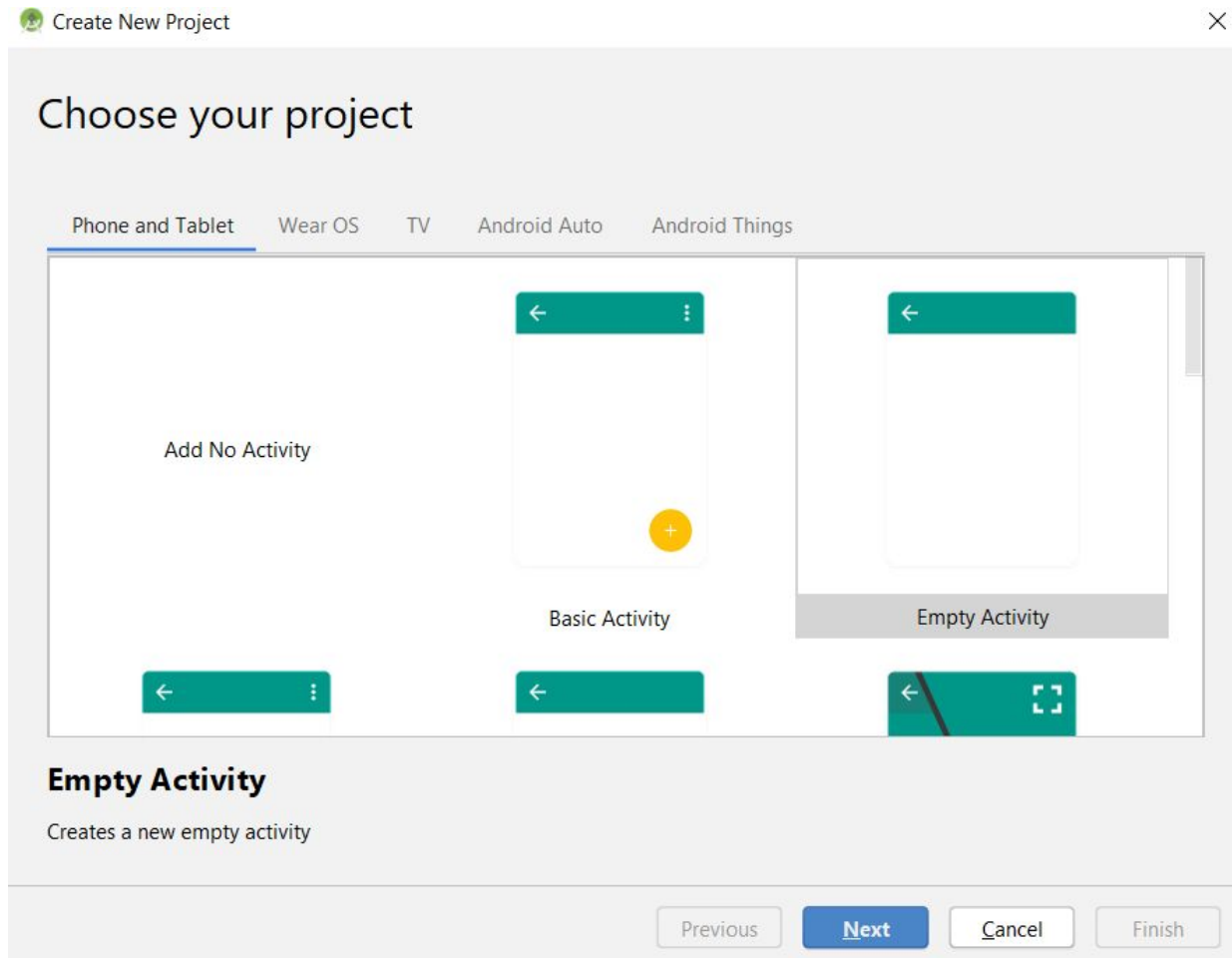
Criando Aplicativo

- Abra Android Studio, Selecione a opção “Start a new Android Studio project”



Criando Aplicativo

- Na próxima tela, escolha Empty Activity

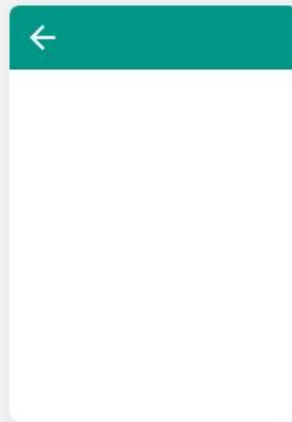


Criando Aplicativo

- Na próxima tela, defina:
 - Nome da aplicação (Name)
 - Nosso aplicativo de aula chamará LMSApp
 - Nome do pacote (Package Name)
 - O nome do pacote precisa ser único na Google Play, por isso o domínio da empresa, ao contrário, é uma boa alternativa
 - Local onde salvar o projeto (Save Location)
 - Linguagem (Language): Kotlin
 - Nível da API (Minimum API Level): 28
 - Quanto menor a versão, em mais dispositivos o aplicativo funcionará, mas menos recursos estarão disponíveis

- Clique “Finish”

Configure your project



Empty Activity

Creates a new empty activity

Name

LMSApp

Package name

br.com.fernandosousa.lmsapp

Save location

D:\LMSApp

Language

Kotlin

Minimum API level

API 28: Android 9.0 (Pie)

i Your app will run on < 1% of devices.

[Help me choose](#)

☐ This project will support instant apps

☒ Use androidx.* artifacts

Previous

Next


Cancel

Finish

Criando Aplicativo

Elementos do projeto	Descrição
Name	Contém o nome do projeto
Package Name	Por padrão, esse nome de pacote também se torna o ID do aplicativo, que pode ser modificado mais tarde.
Save Localization	Especifica em qual diretório será armazenado o projeto. Preferencialmente o mesmo não deve conter acentos.
Language	Especifica o idioma desejado para o Android Studio ao criar uma amostra codificada para o projeto
Minimum API level,	Define o nível mínimo da API com o qual o app será compatível. Quando você seleciona um nível de API anterior, seu app fica limitado a um número menor de APIs modernas do Android.

Criando Emulador

- O Emulador Android é um dispositivo virtual que simula a configuração real de um celular
 - É possível criar emuladores com diversas configurações diferentes, incluindo versão da API Android
- Clique no ícone do AVD Manager () para um novo emulador
- Na tela que abrir, clique em “Create virtual Device”

Criando Emulador

Android Virtual Device Manager



Your Virtual Devices

Android Studio



Virtual devices allow you to test your application without having to own the physical devices.


[+ Create Virtual Device...](#)

To prioritize which devices to test your application on, visit the [Android Dashboards](#), where you can get up-to-date information on which devices are active in the Android and Google Play ecosystem.

Criando Emulador

- A primeira tela mostra as opções de hardware
 - Existem opções pré-configuradas para diversos tipos de smartphones, tablets, wearables e TV
- Para a aula, vamos utilizar a opção 5.4” FWVGA, dentro da categoria “Phone”
- Clique “Next”

Criando Emulador



Select Hardware

Choose a device definition

Category

Phone

Tablet

Wear


TV

Name	Size	Resolution	Density
Nexus 5	4,95"	1080x1920	xxhdpi
Nexus 4	4,7"	768x1280	xhdpi
Galaxy Nexus	4,65"	720x1280	xhdpi
5.4" FWVGA	5,4"	480x854	mdpi
5.1" WVGA	5,1"	480x800	mdpi

New Hardware Profile

Import Hardware Profiles

5.4" FWVGA



Size: large
Ratio: long
Density: mdpi

Clone Device...

Previous

Next


Cancel

Finish

Criando Emulador

- Na próxima tela selecione a imagem do emulador, com a versão do Android
 - As opções mostradas serão aquelas já instaladas na máquina de desenvolvimento
 - Lembre-se sempre de escolher uma versão igual ou superior ao da API escolhida para o aplicativo
- Clique “Next”

Criando Emulador




System Image

Select a system image

Recommended x86 Images Other Images

Release Name	API Level	ABI	Target
Marshmallow	23	x86_64	Android 6.0 (v)
Marshmallow	23	x86	Android 6.0 (v)
Lollipop	22	x86_64	Android 5.1 (v)
Lollipop	22	x86	Android 5.1 (v)
KitKat	19	x86	Android 4.4 (v)

Marshmallow



API Level

23

Android

6.0

Google Inc.

System Image

x86

These images are recommended because they run the fastest and include support for Google APIs

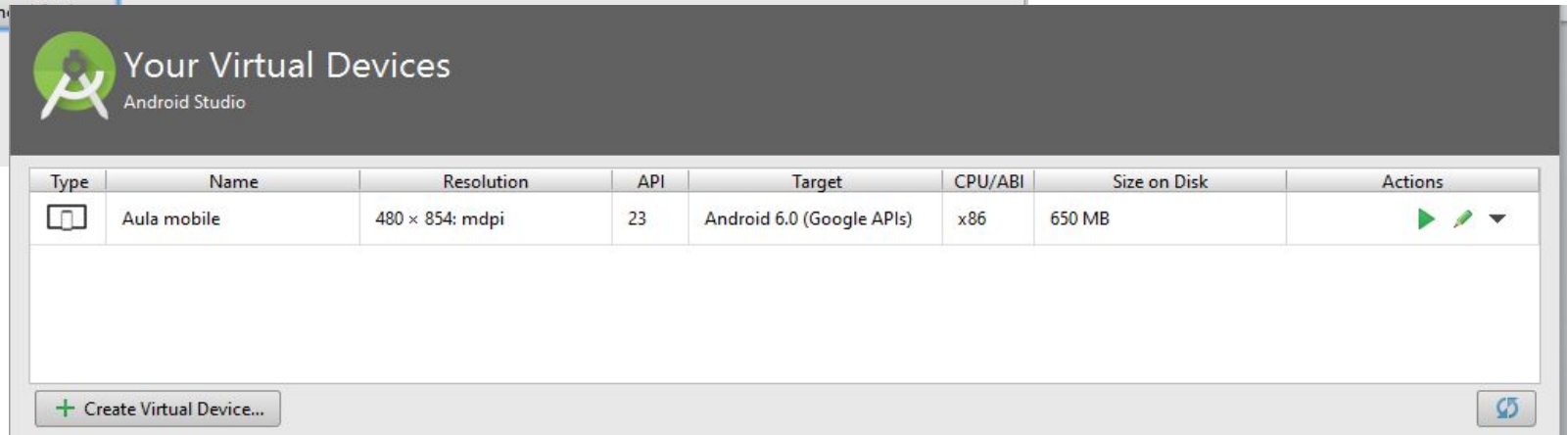
Questions on API level?
See the [API level distribution chart](#)

Previous Next Cancel Finish

Criando Emulador

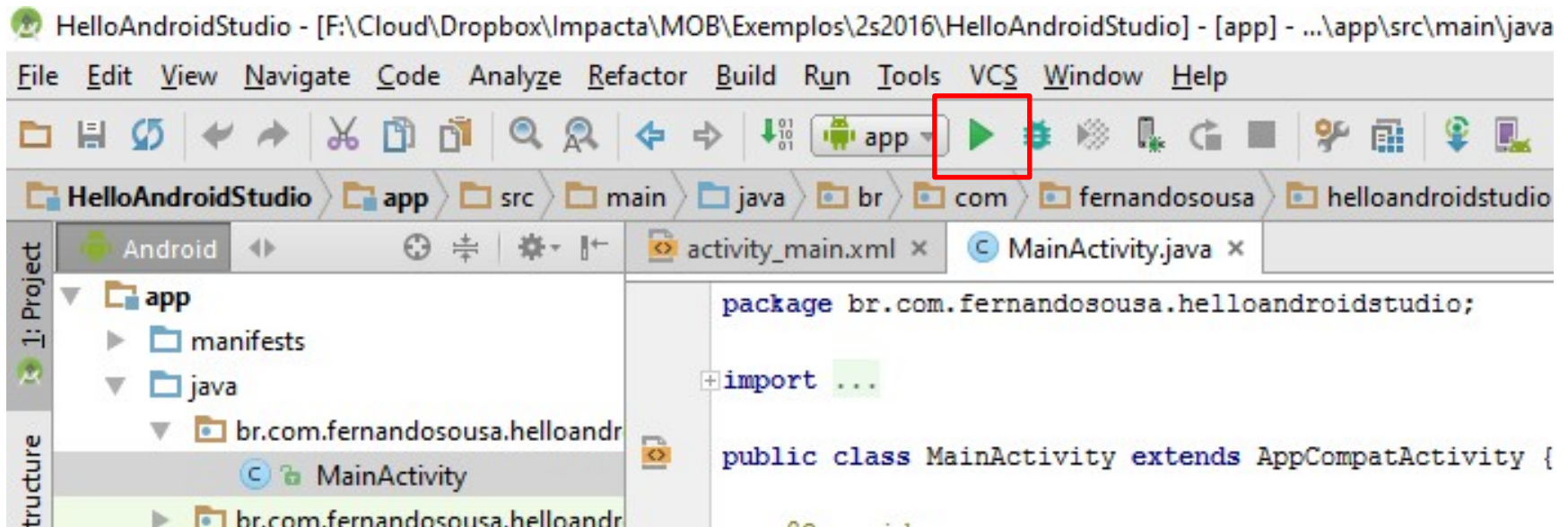
- Depois, basta verificar e alterar alguma configuração necessária
 - Para o hardware escolhido, temos que alterar o nome, que está inválido.
- Clique em Finish
- Você pode criar quantos emuladores quiser
 - Testar seu aplicativo em diferentes versões de API, configurações de hardware e tamanhos de tela

Criando Emulador



Executando Aplicativo

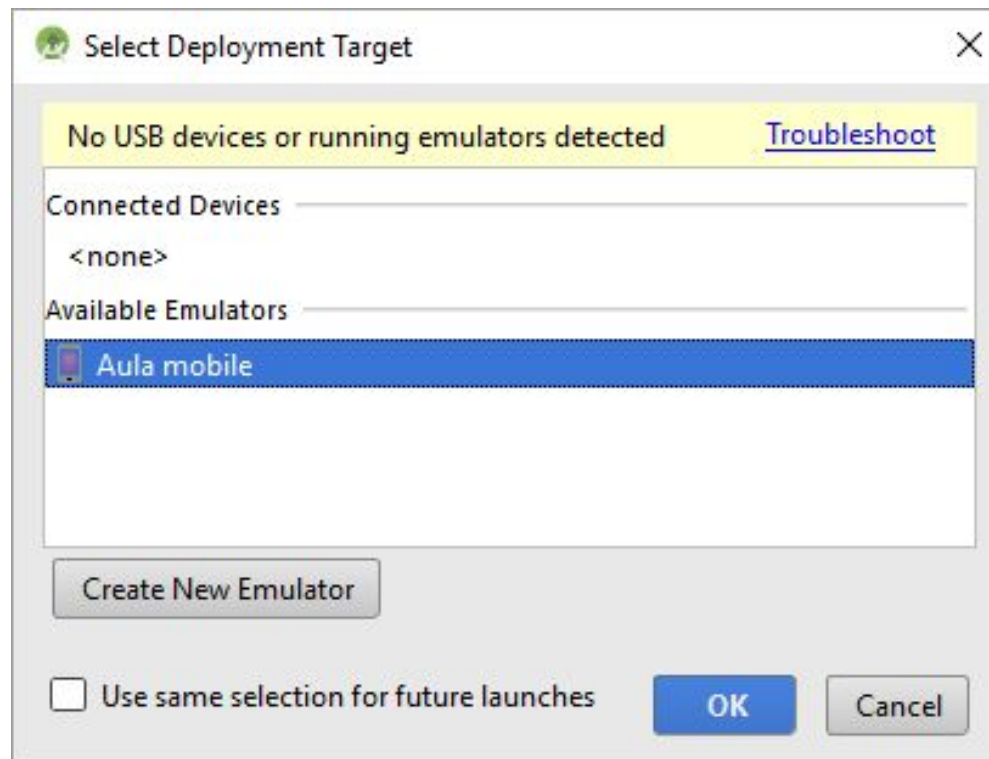
- Volte a tela principal da IDE e execute seu aplicativo clicando no “Play” da barra superior



- Escolha o emulador, e clique “OK”

Executando Aplicativo

- Ao apertar o play você deve selecionar qual emulador utilizará. O emulador abrirá com seu aplicativo executando



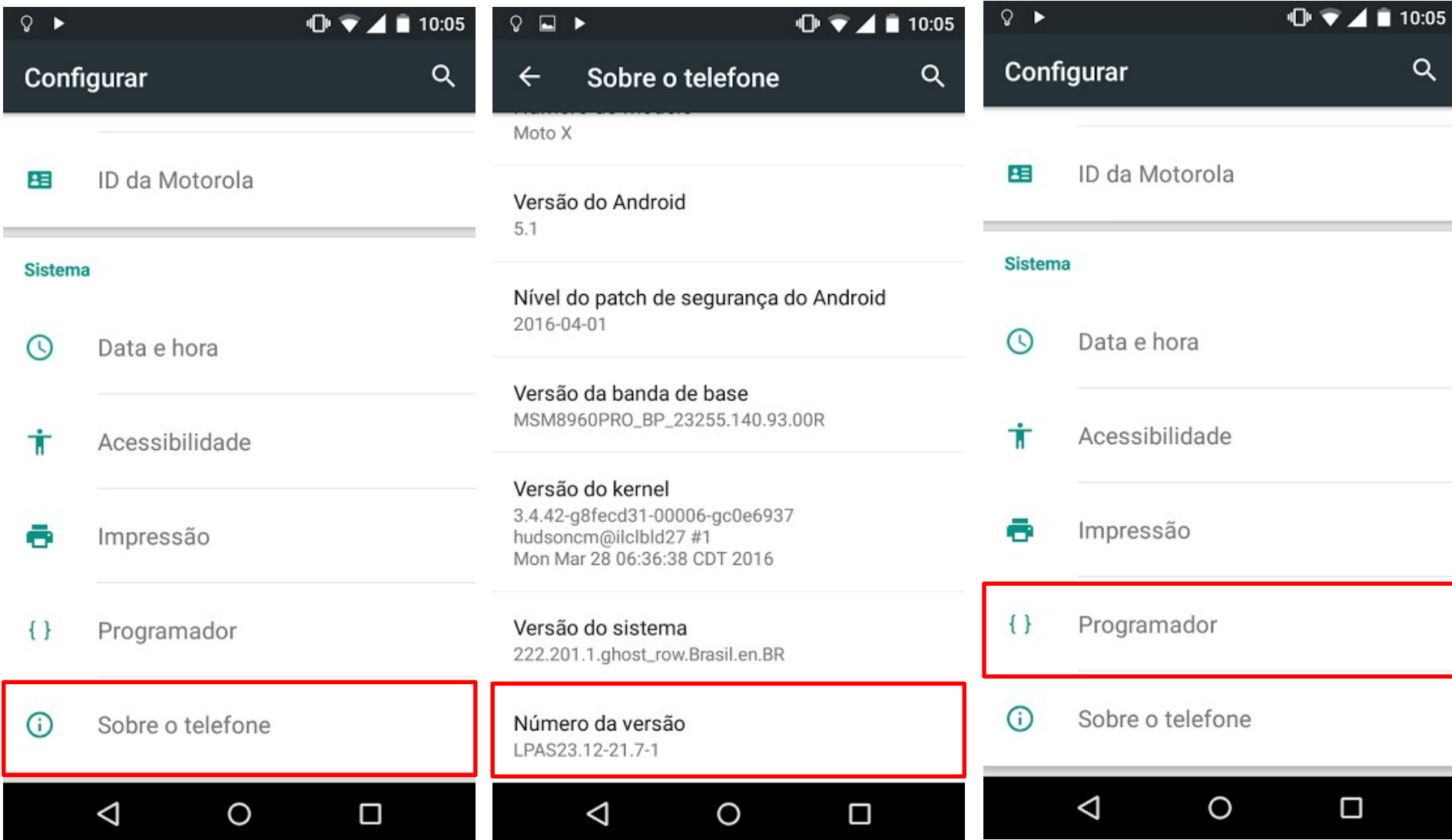
Executando Aplicativo

- Você também pode executar o aplicativo diretamente do seu smartphone Android
- A configuração depende do fabricante do seu smartphone

Executando Aplicativo no Smartphone

- Para conseguir executar diretamente no telefone, primeiro precisam habilitar seu telefone para desenvolvimento. Na versão Lollipop faça o seguinte:
 - Abra as configuração do telefone
 - Vá até a opção “Sobre o telefone” (último item)
 - Clique 7 vezes no item “Número da versão”
 - Volte às configurações, e haverá uma nova opção, “Programador” (penúltimo item)
 - Abra esta opção

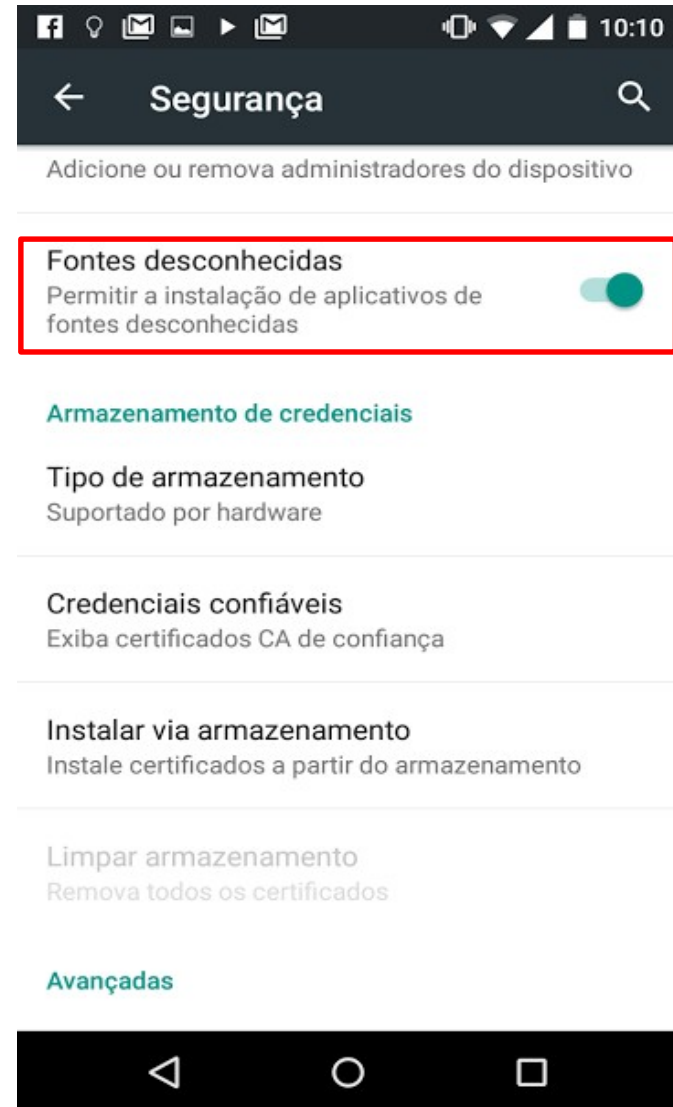
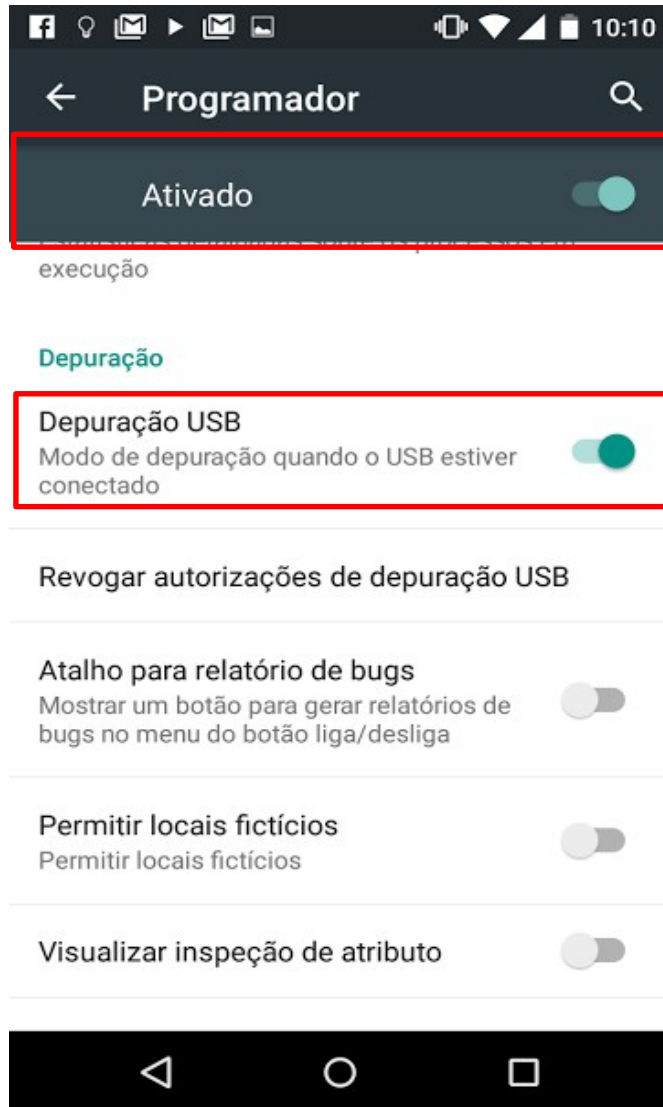
Executando Aplicativo no Smartphone



Executando Aplicativo no Smartphone

- Na opção programador:
 - Ative a opção
 - Ative a depuração USB
- Volte às configurações e selecione a opção segurança
 - Habilite a opção “Fontes Desconhecidas”

Executando Aplicativo no Smartphone

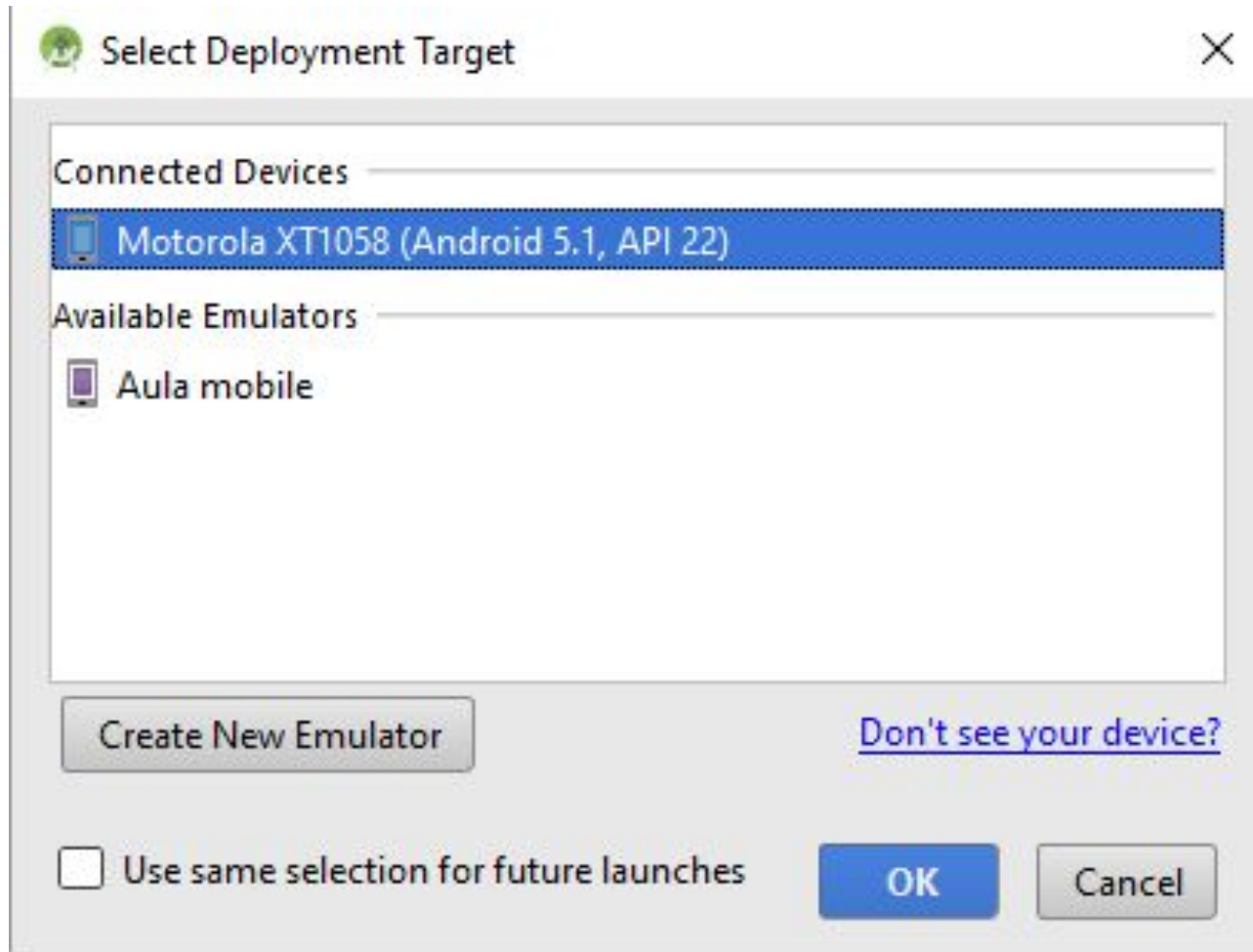


Executando Aplicativo no Smartphone

- Agora é preciso conectar o smartphone no computador via USB e ter o driver ADB interface, específico do fabricante, instalado
 - Para alguns fabricante, o driver é instalado automaticamente
 - Para outros é preciso instalar manualmente
 - [Consulte](https://developer.android.com/studio/run/oem-usb.html)
<https://developer.android.com/studio/run/oem-usb.html>
- Depois disso, o seu smartphone vai aparecer na lista de dispositivos quando executar o aplicativo pelo Android Studio

Basta selecionar o dispositivo e clicar em OK

Executando Aplicativo no Smartphone



Estrutura do Projeto

- Ao criar o projeto no Android Studio, existe uma região chamada “Project”, com a aba “Android” habilitada, do lado esquerdo da IDE
 - Esta aba mostra os arquivos essenciais do seu projeto
 - Aqueles que podem ser alterados dentro do seu projeto
- Existem basicamente 3 pastas dentro da pasta app, contendo o código fonte:
 - manifest
 - java
 - res

Diretórios

- App: Define tudo que está dentro da aplicação
- AndroidManifest.xml: é o arquivo principal de um projeto Android, é ele que define todas as configurações que serão trabalhadas na aplicação
- Java: Contém a lógica da aplicação
 - MainActivity: Primeira activity criada por padrão
 - Activity_main.xml: Lida com a parte visual da aplicação. Quando a activity é criada, ele fala que o conteúdo está presente

Diretórios

- RES: Contém os recursos da aplicação
- Drawable: Armazena recursos gráficos, uma imagem, sons, vídeos, todos os ícones ficam dentro da pasta
- Layouts: Contém as interfaces gráficas, as telas da aplicação
- Mipmap: Contém os ícones da aplicação

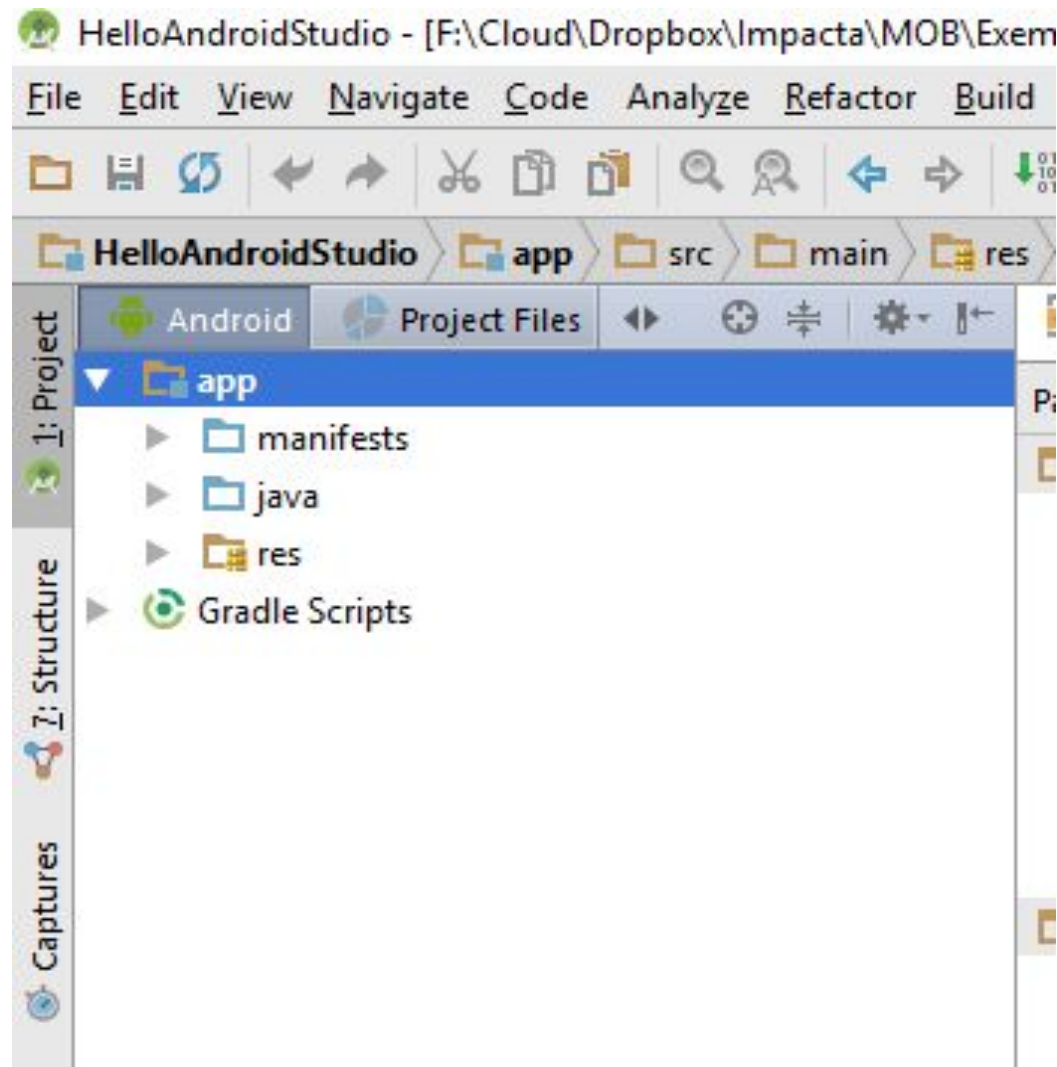
Diretórios

- Values: Contém os valores trabalhados para os atributos da aplicação. De maneira reutilizável
 - Colors.xml: armazena as cores trabalhadas pela a aplicação
 - Strings.xml: armazena textos
 - Styles.xml: armazena estilos da aplicação
- Gradle Scripts: Contém as dependências da aplicação. É um gerenciador de dependências
 - Build.gradle: Contém informações como dependências reutilizáveis, SDK mínimo e máximo e muitas outras informações

Componentes de Interface

Componente	Funcionalidade
Button	Botão para cliques do usuário
Text Field	Caixa de texto
CheckBoxes	Caixa de marcação (seleção múltipla)
RadioButton	Botão de rádio (seleção única)
TextView	Funciona com rótulos, agrupadores na interface

Executando Aplicativo no Smartphone



Exercícios

- 1. Repita o passo a passo e gere o seu primeiro Hello World em Android.
- 2. Adicione pelo menos uma funcionalidade na aplicação criada

Github

- Link com os códigos gerados nesta aula:
- <https://github.com/fabiodasilva500/Aulas-Mobile/tree/Aula-6-Overview-Android>

Referências

- LECHETA, R. R. Android Essencial com Kotlin. Edição: 1ª ed. Novatec, 2017.
- <https://docente.ifrn.edu.br/andrealmeida/disciplinas/2012.2/minicurso-introducao-ao-android/parte-01-conhecendo-o-sistema-e-primeiro-programa>
- <https://developer.android.com/studio>
- <https://developer.android.com/studio/workflow>
- <https://developer.android.com/studio/build>
- <https://developer.android.com/studio/projects/create-project>
- <https://developer.android.com/training/basics/firstapp?hl=pt-br>
- <https://statics-shoptime.b2w.io/sherlock/books/firstChapter/123530212.pdf>
- <https://www.sncticet.ufam.edu.br/2017/downloads/christianreis.pdf>