

# Android com Kotlin

Capítulo 1: Iniciando com Android e Kotlin

<https://github.com/rafael-vieira-coelho/android>

## Android Programming with Kotlin for Beginners

Build Android apps starting from zero programming experience  
with the new Kotlin programming language



# Capítulo 1: Iniciando com Android e Kotlin

- Aprender como Kotlin e Android trabalham juntos
- Como configurar a IDE Android Studio
- Compreender o que é um Android app
- Como fazer o deploy de um app em um emulador Android
- Como rodar um app em um celular

# Por que não usar Java com Android?

- Android existe desde 2008, sendo grátis e aberta.
- O Software Development Kit (SDK) do Android é implementado em boa parte em Java.
- Independente se o seu app foi implementado em Java ou Kotlin, gera-se um código DEX (Dalvik Executable) para rodar no celular.
- Diferentemente de Java, Kotlin é uma linguagem sucinta e de curva de aprendizado mais acentuada. E ela visa fazer mais com menos código.
- Como exemplos de apps implementados em Kotlin: Kindle, Twitter e Netflix.

# O que são os recursos de Android?

- Além do código-fonte em linguagem Kotlin, podemos ter arquivos auxiliares em nossos projetos Android.
- Podem ser imagens, arquivos de som, layout, etc.
- Para organizar estes arquivos podemos separá-los pacotes (packages).
- Os pacotes que são disponibilizados pela API Android: “<https://developer.android.com/reference/packages>

# Como Android e Kotlin trabalham juntos?

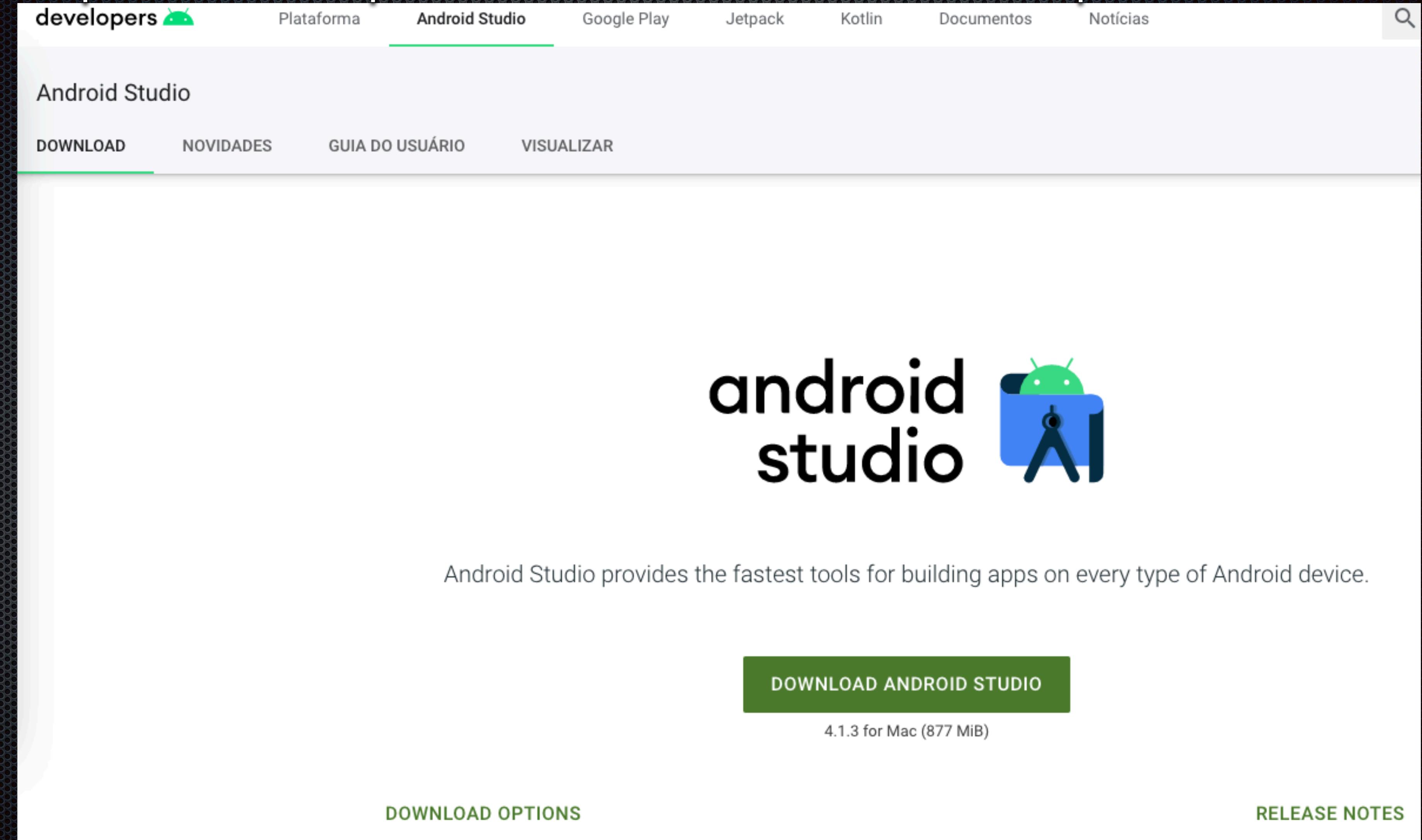
- Android roda em uma versão simplificada do Sistema Operacional Linux.
- Precisamos aprender a usar as classes da API do Android.
- O código DEX (juntamente com outros recursos) são armazenados em um arquivo APK (Android Application Package).
- Iremos programar na IDE Android Studio.

# **INSTALANDO IDE ANDROID STUDIO**

# Como Configurar o Android Studio?

<https://developer.android.com/studio?hl=pt>

- Inicialmente, precisamos baixar o software no site.
- Ele detectará automaticamente o seu SO.



The screenshot shows the 'Android Studio' section of the developer.android.com website. At the top, there's a navigation bar with links for 'developers' (with a green Android icon), 'Plataforma', 'Android Studio' (which is underlined in green), 'Google Play', 'Jetpack', 'Kotlin', 'Documentos', and 'Notícias'. A search icon is also present. Below the navigation, the word 'Android Studio' is displayed in bold. Underneath it, there are four tabs: 'DOWNLOAD' (underlined in green), 'NOVIDADES', 'GUIA DO USUÁRIO', and 'VISUALIZAR'. The main content area features the 'android studio' logo, which consists of the words 'android' and 'studio' in lowercase black font next to a blue icon of a robot head with a wrench. Below the logo, a subtitle reads 'Android Studio provides the fastest tools for building apps on every type of Android device.' At the bottom of the page, there's a large green button labeled 'DOWNLOAD ANDROID STUDIO' and a link below it stating '4.1.3 for Mac (877 MiB)'. Other options like 'RELEASE NOTES' and 'DOWNLOAD OPTIONS' are also visible at the bottom.

- Aceitar os termos e condições

Download Android Studio X

Before downloading, you must agree to the following terms and conditions.

## Terms and Conditions

This is the Android Software Development Kit License Agreement

### 1. Introduction

1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.

1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: <http://source.android.com/>, as updated from time to time.

1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which

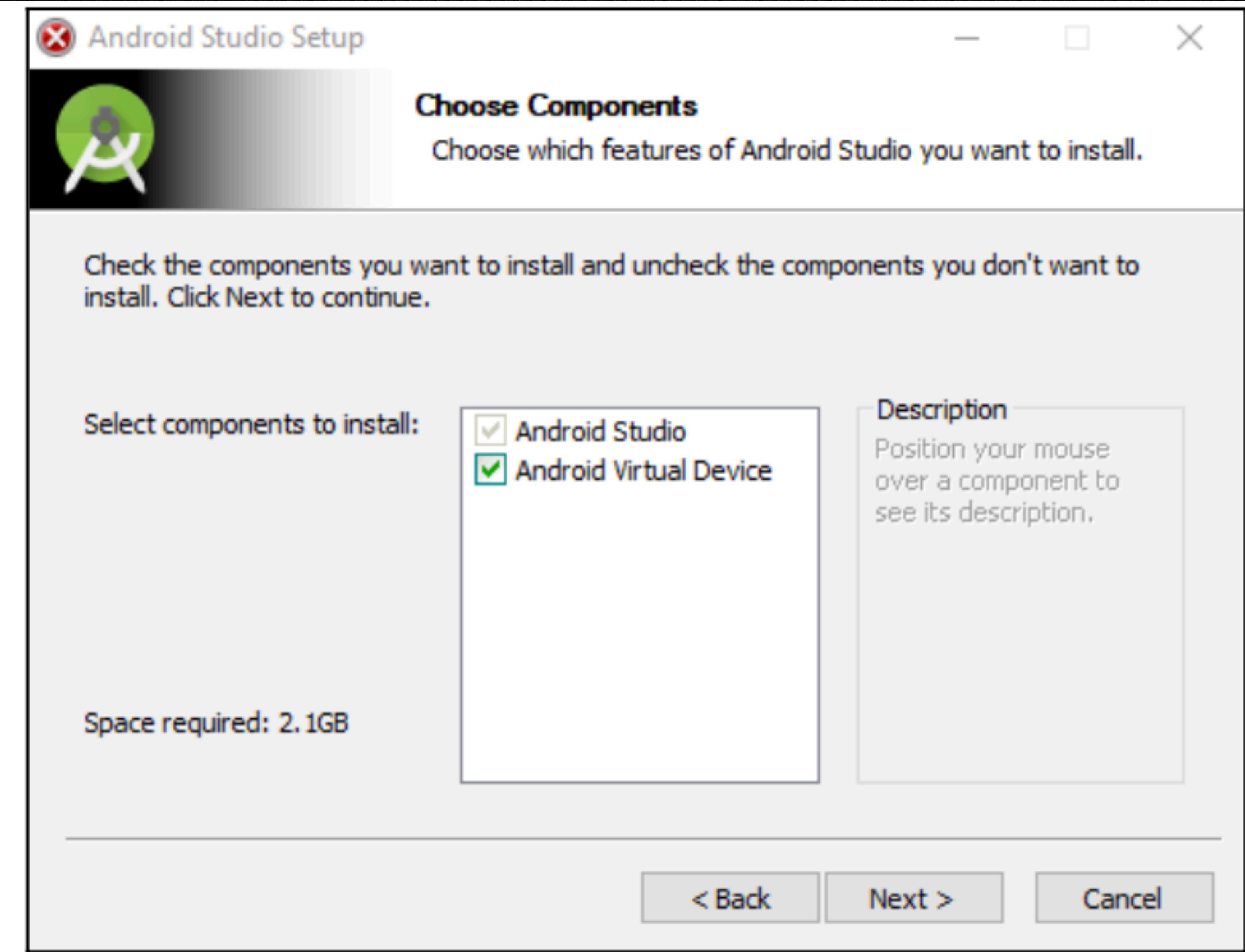
I have read and agree with the above terms and conditions

**DOWNLOAD ANDROID STUDIO FOR WINDOWS**

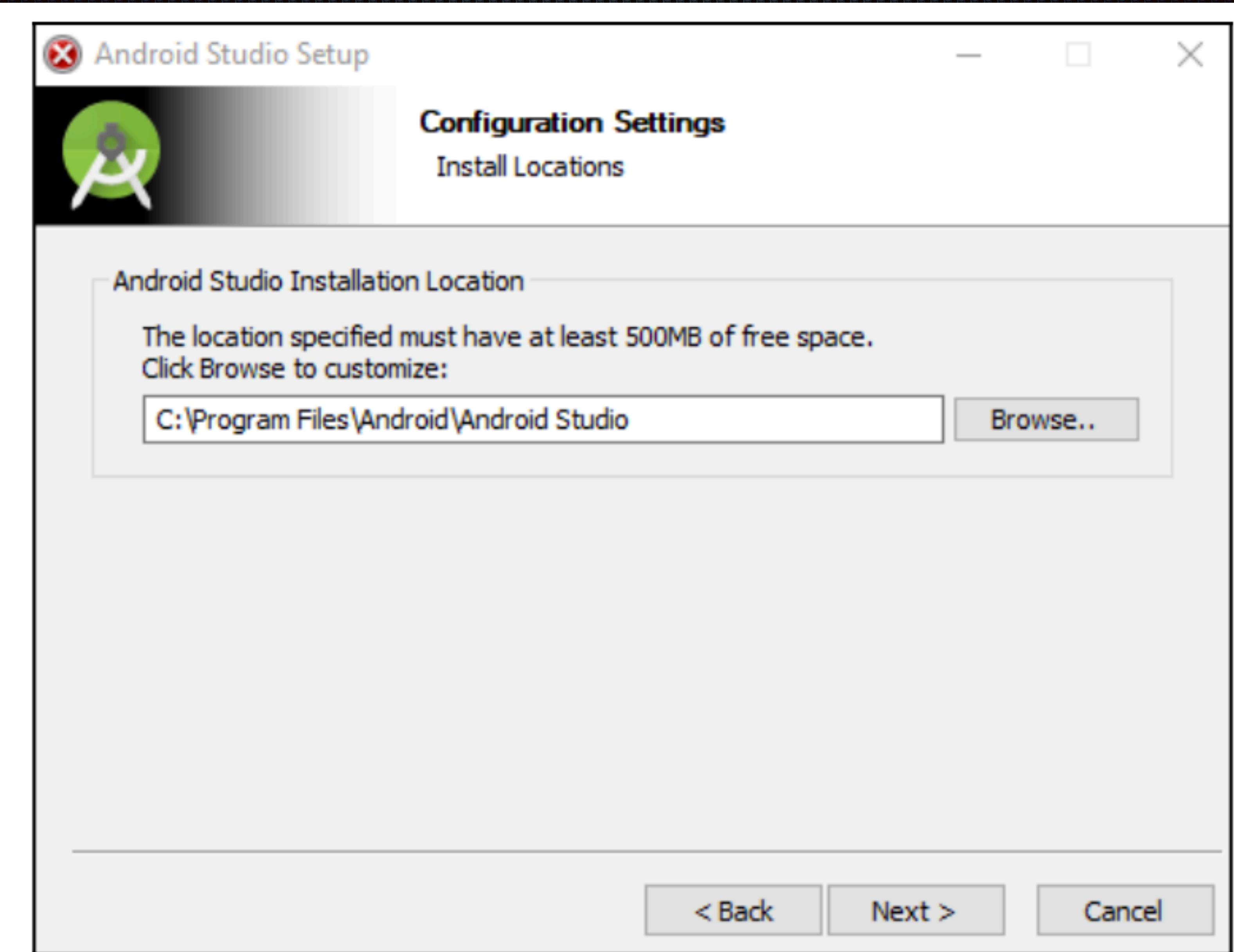
- Iniciar a instalação (next).



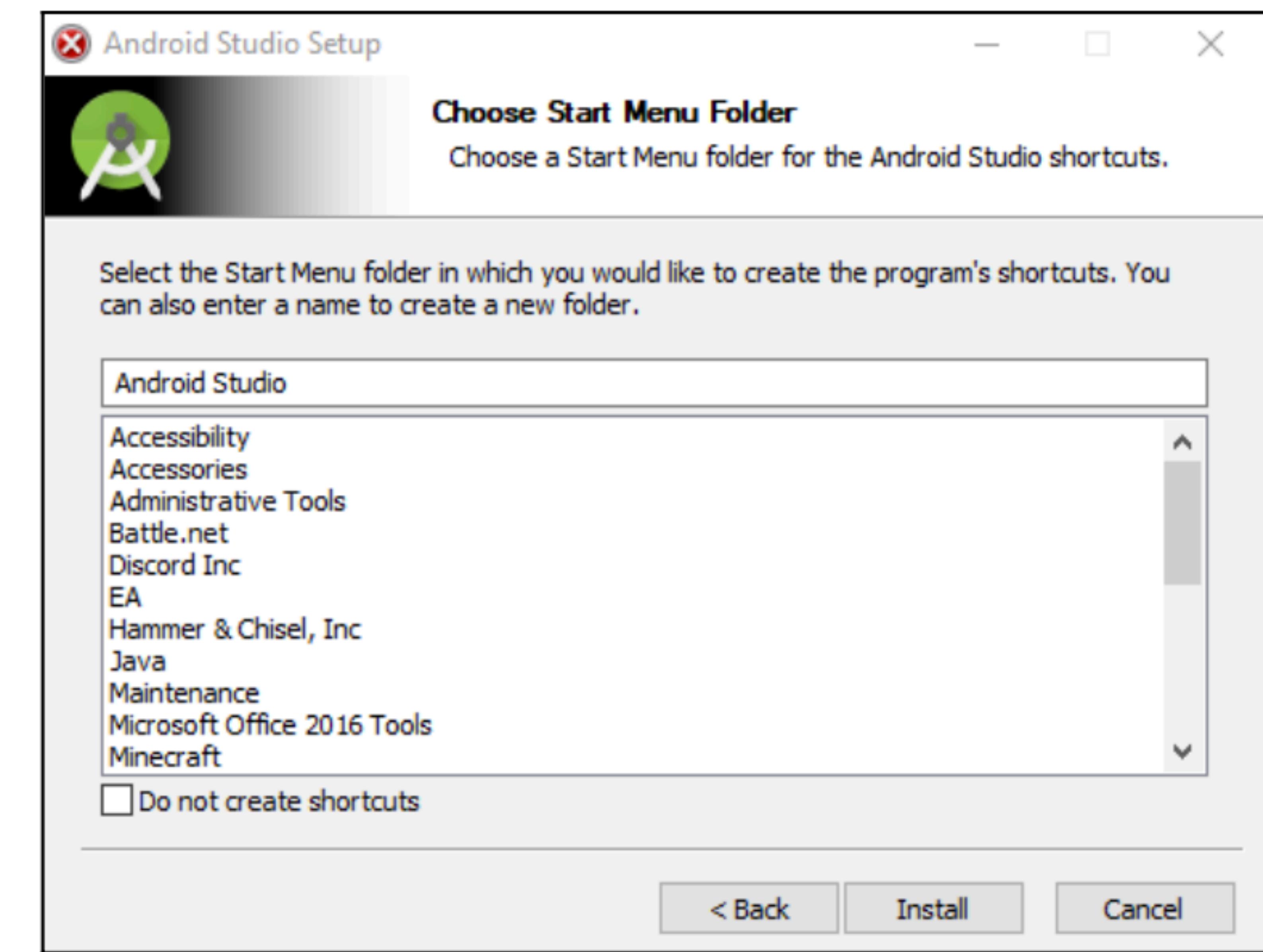
- Selecionar os componentes que devem ser instalados.



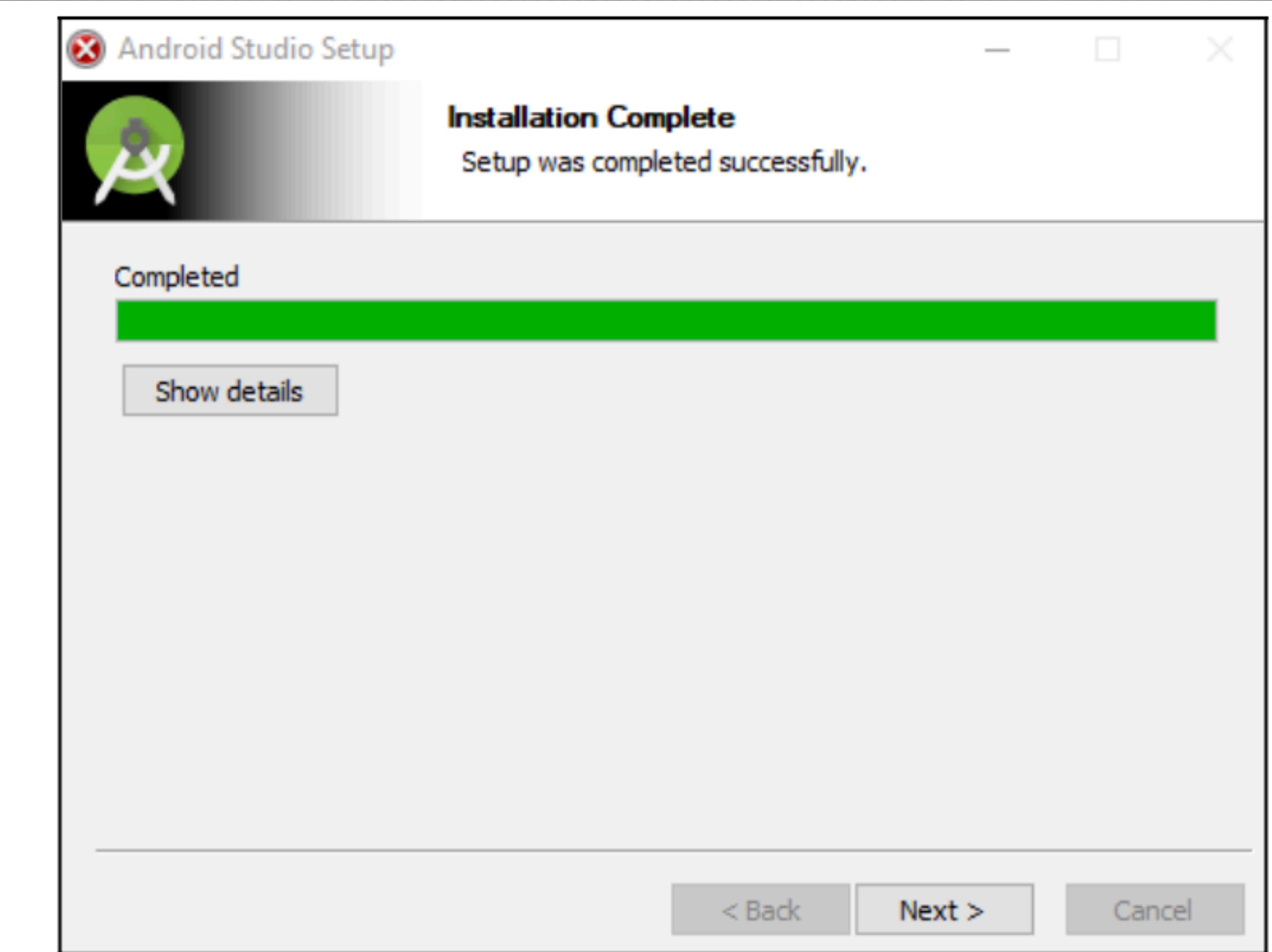
- Definir a pasta destino da instalação.



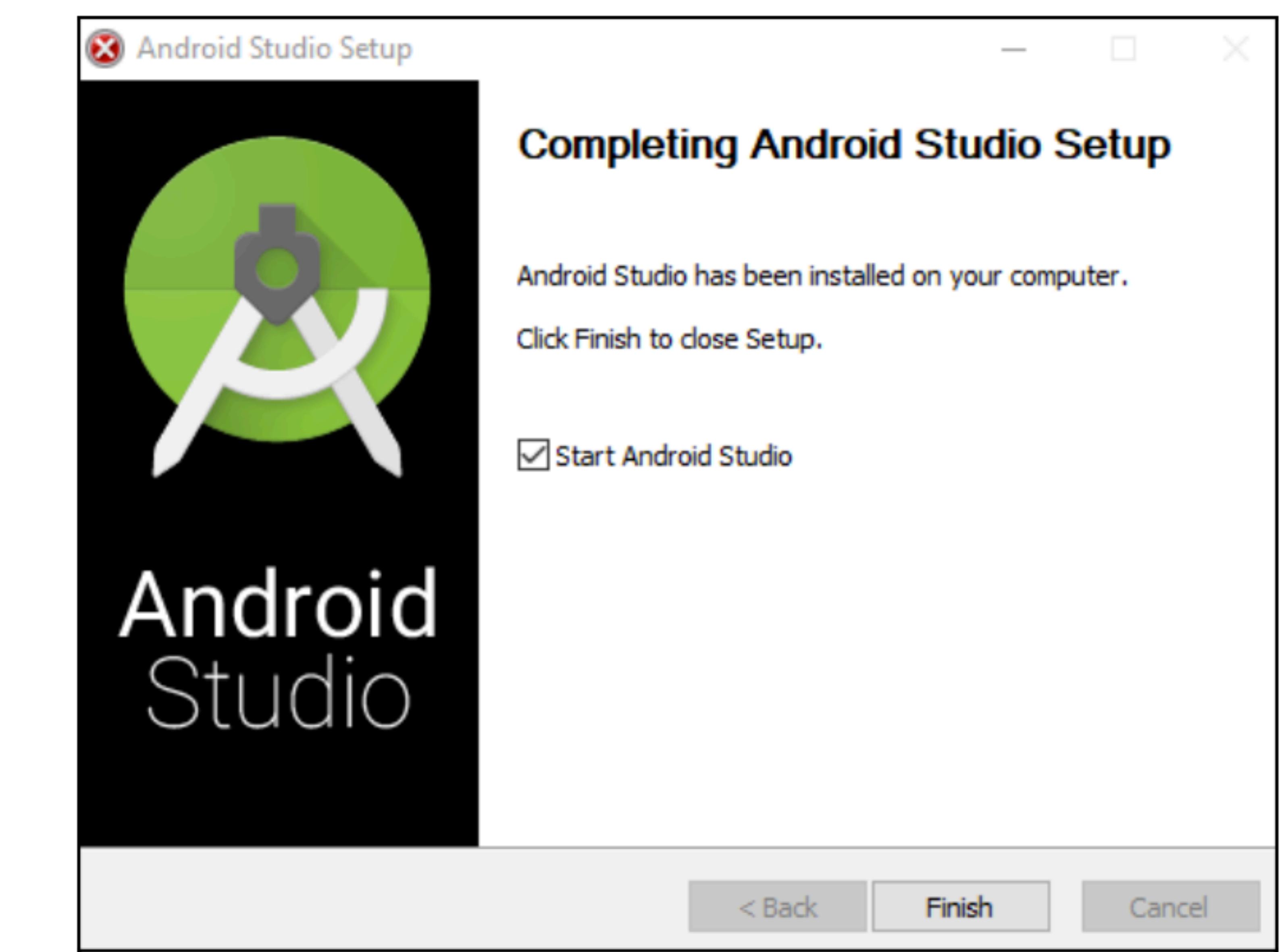
- Definir qual pasta do menu iniciar será colocado seu atalho.



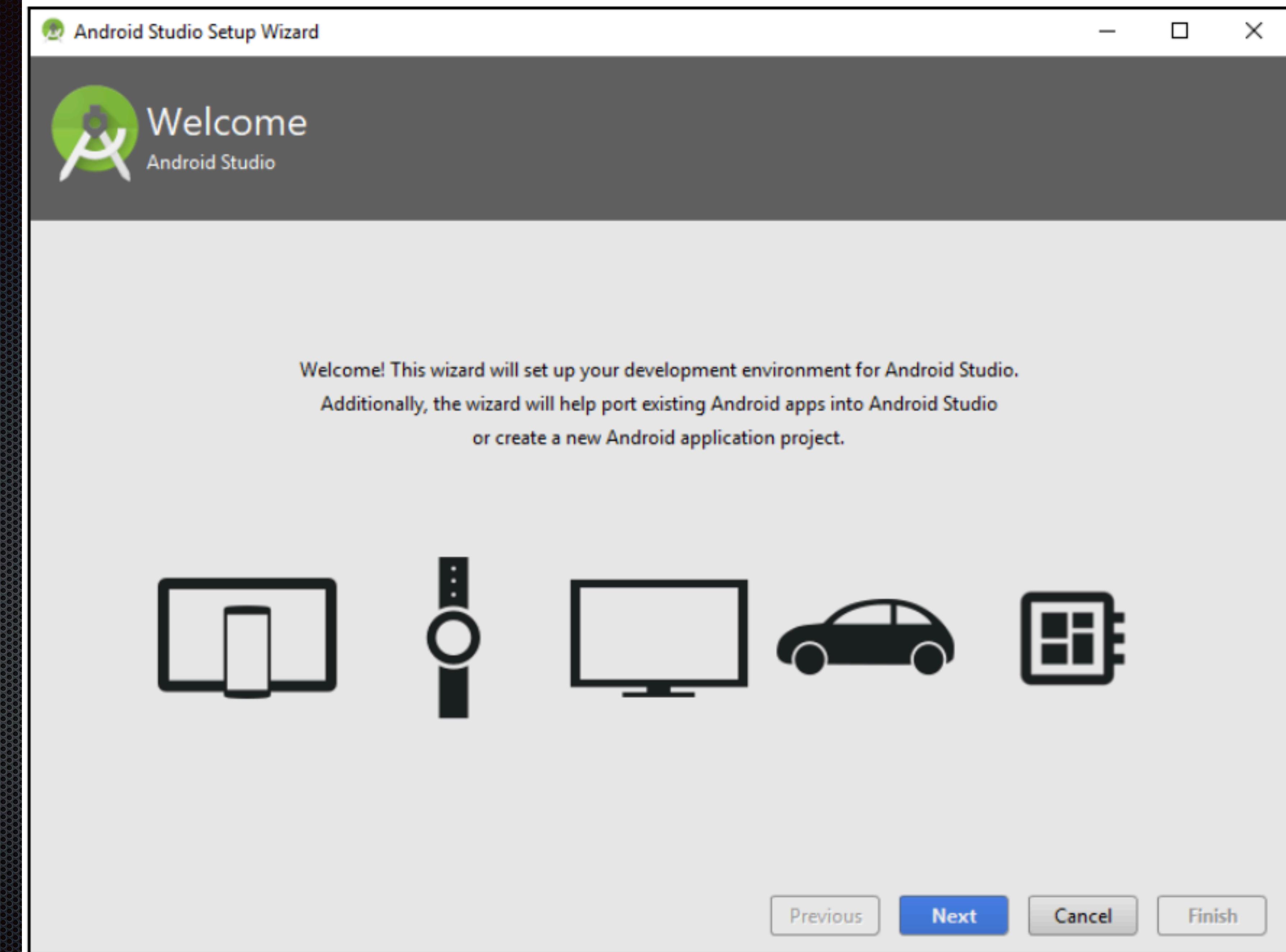
- Esperar a finalização da instalação e clicar em next.



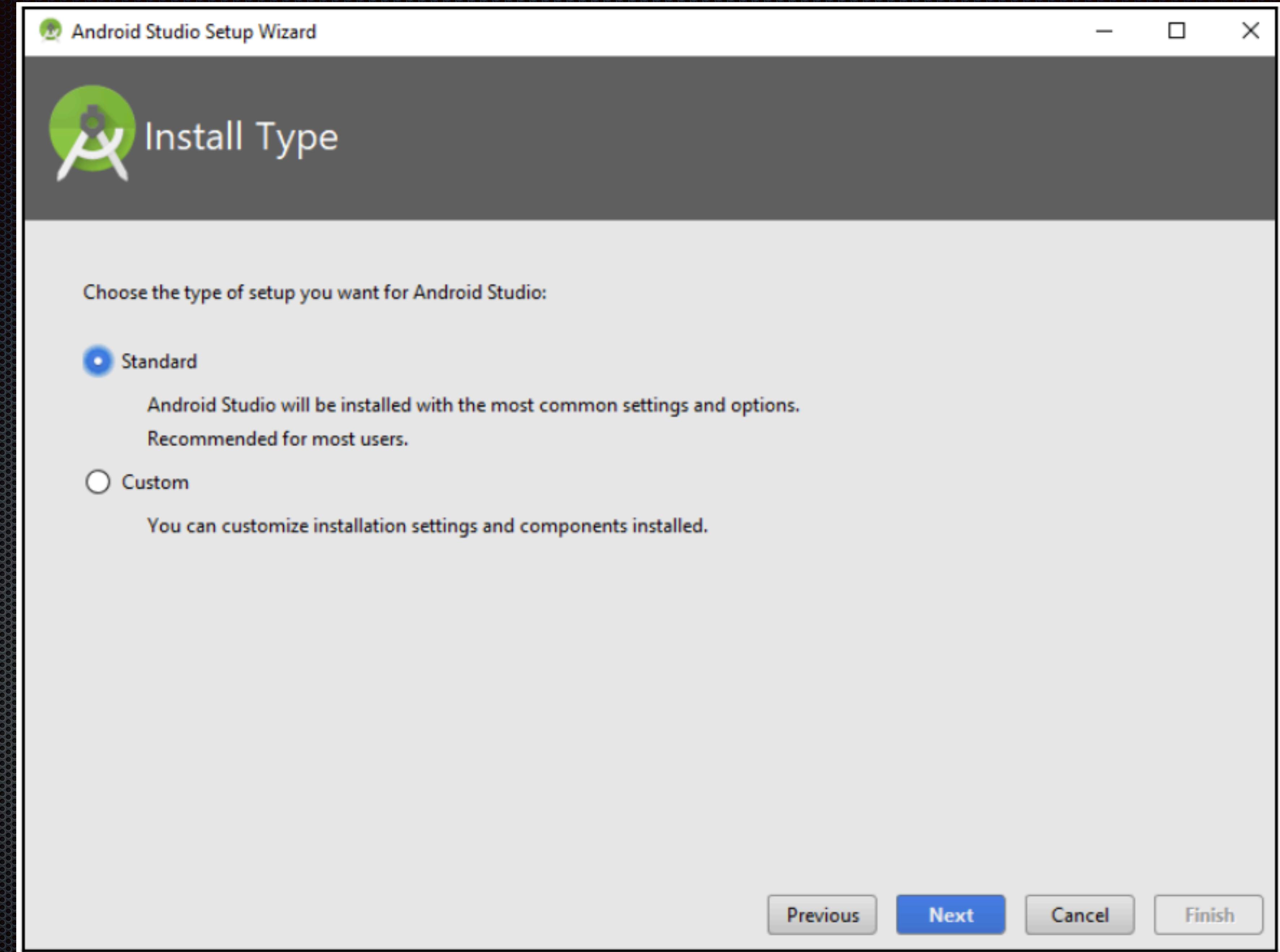
- Finalizar a instalação.
- Mantenha marcada a caixa “Start Android Studio”.



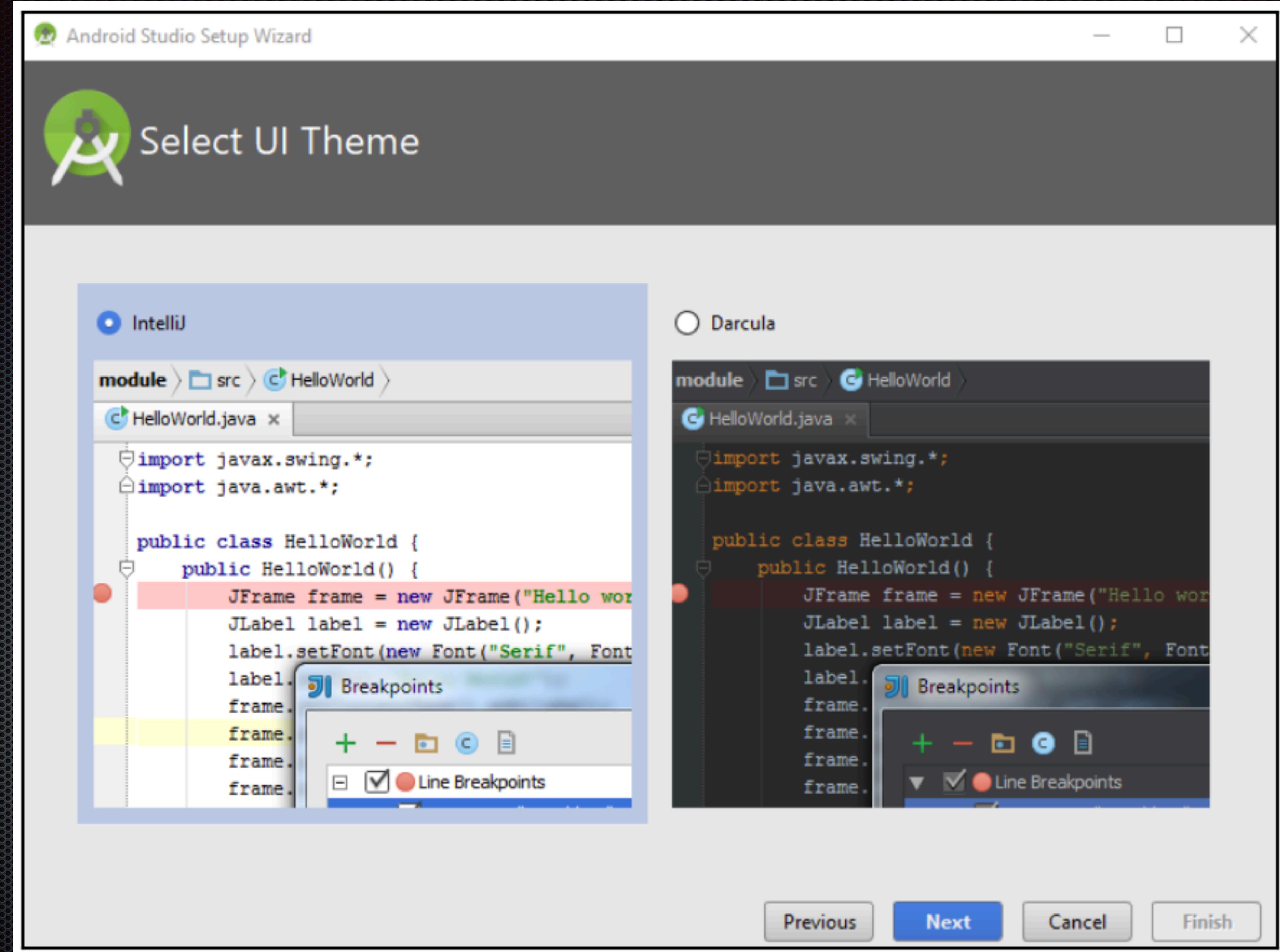
- Esta é a janela de boas vindas que será executada somente na primeira vez.



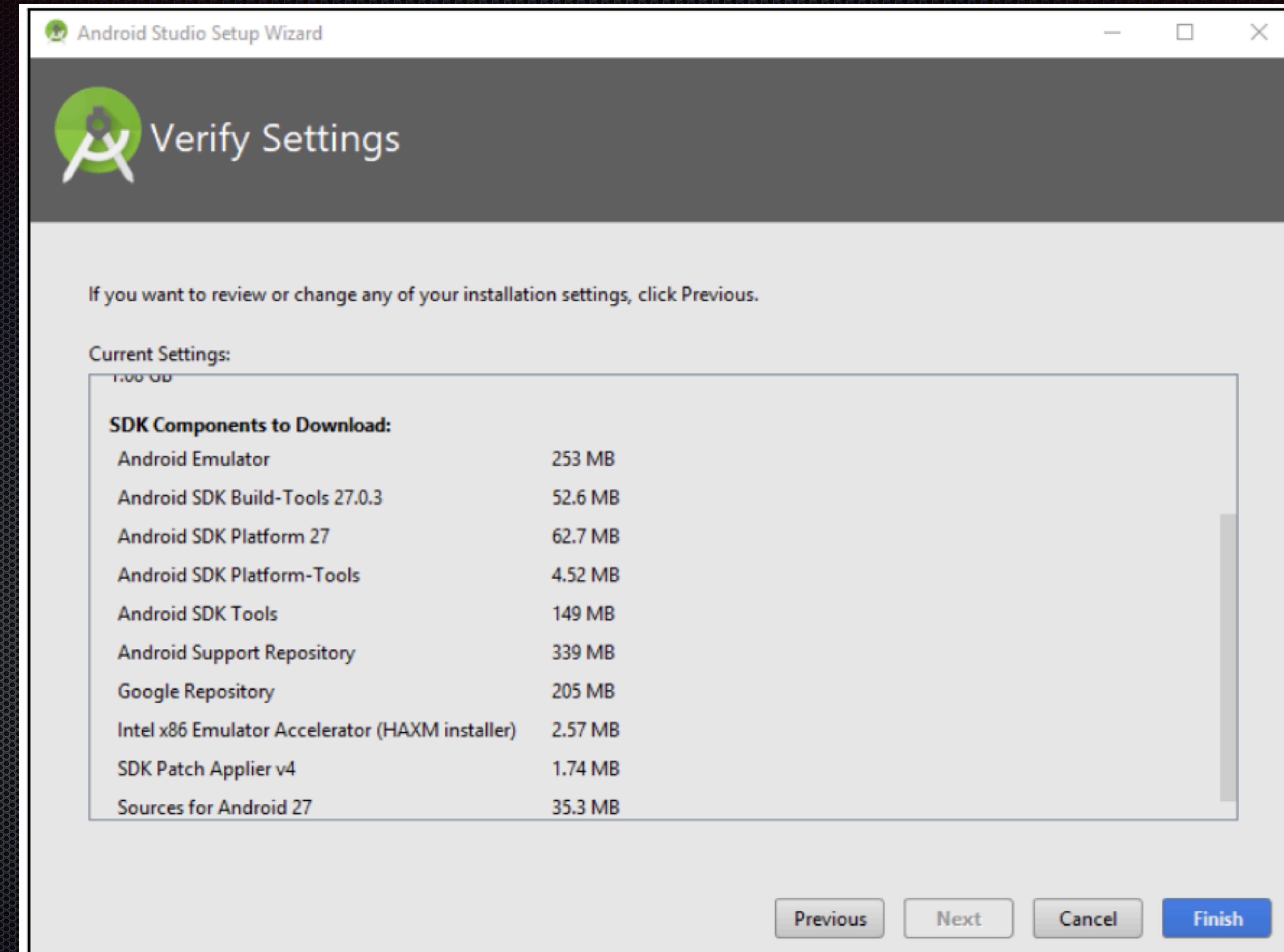
- Escolhemos o tipo de instalação padrão e clicamos em next.



- Escolhemos o tema da IDE (escuro ou claro).

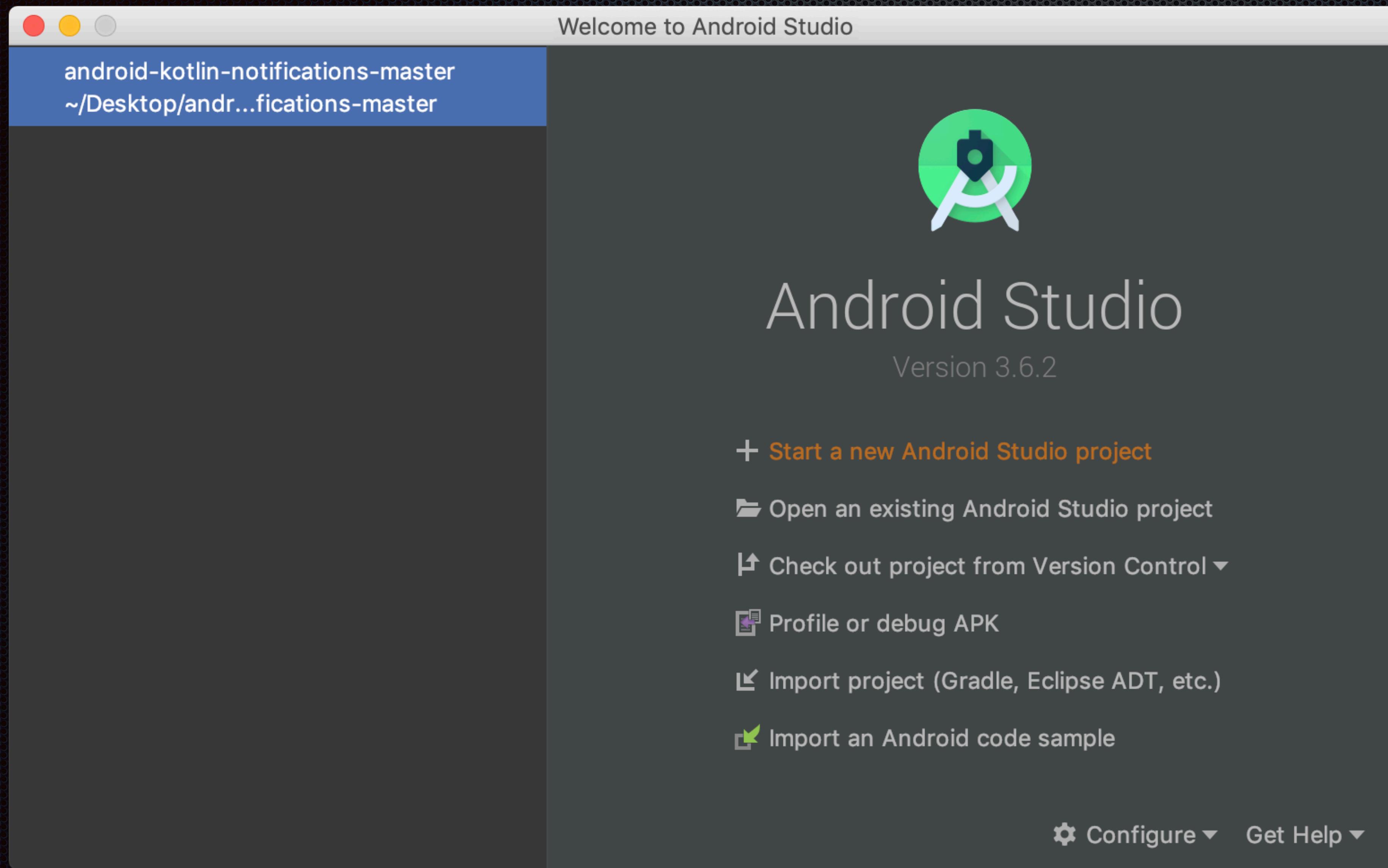


- É realizada uma verificação da instalação de seus componentes.

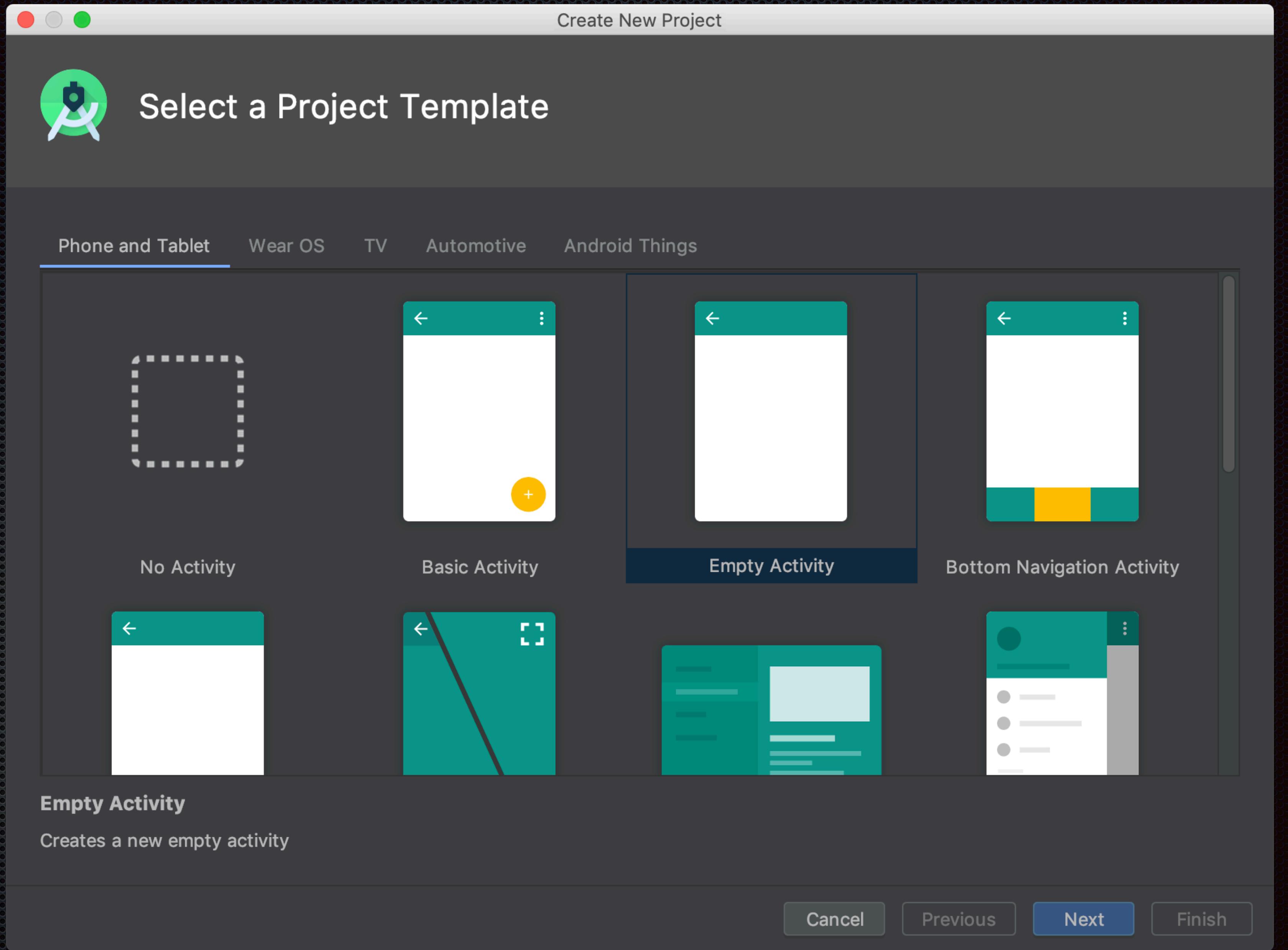


# CRIANDO UM NOVO PROJETO

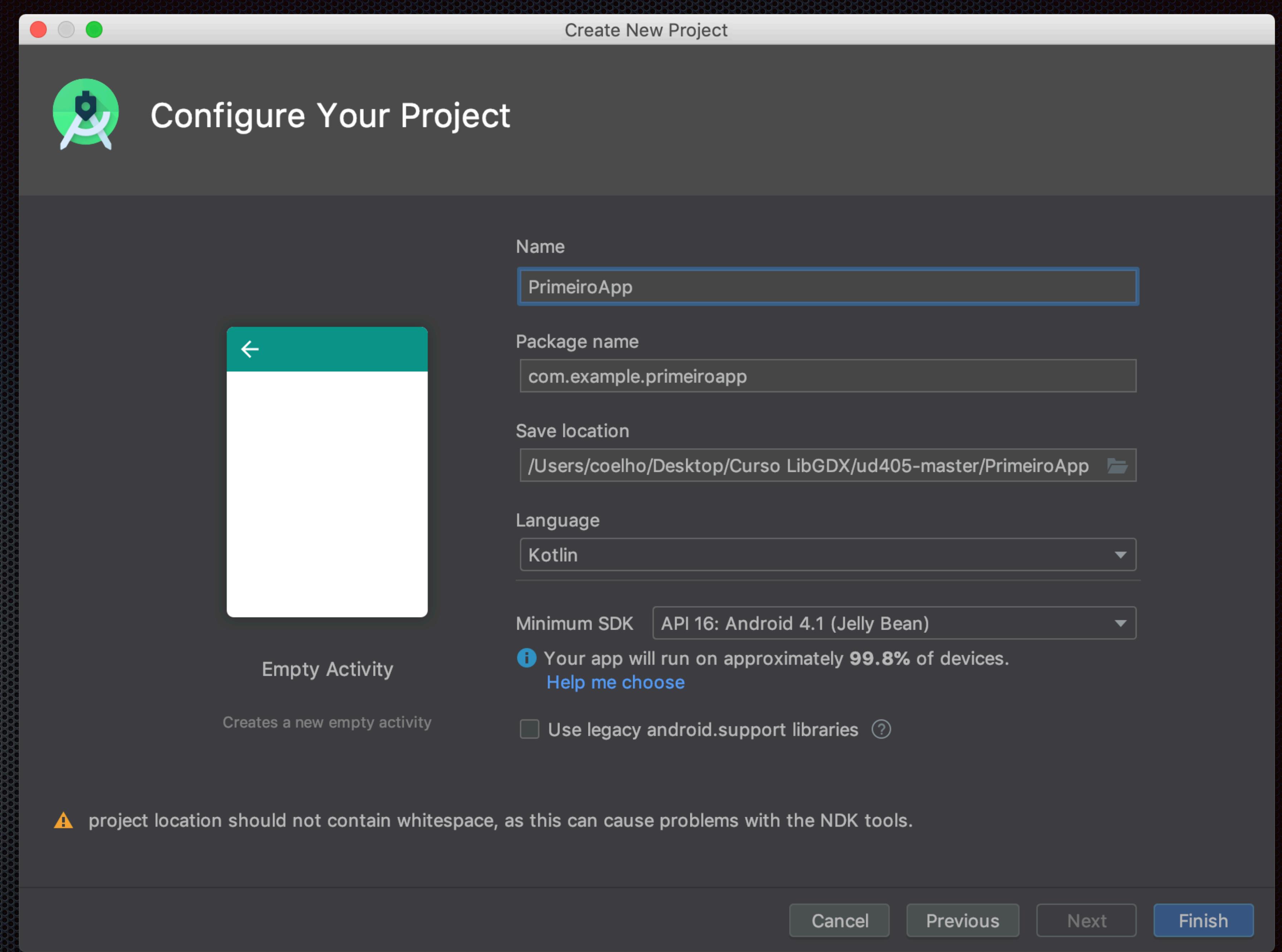
- É mostrada a tela inicial do Android Studio.



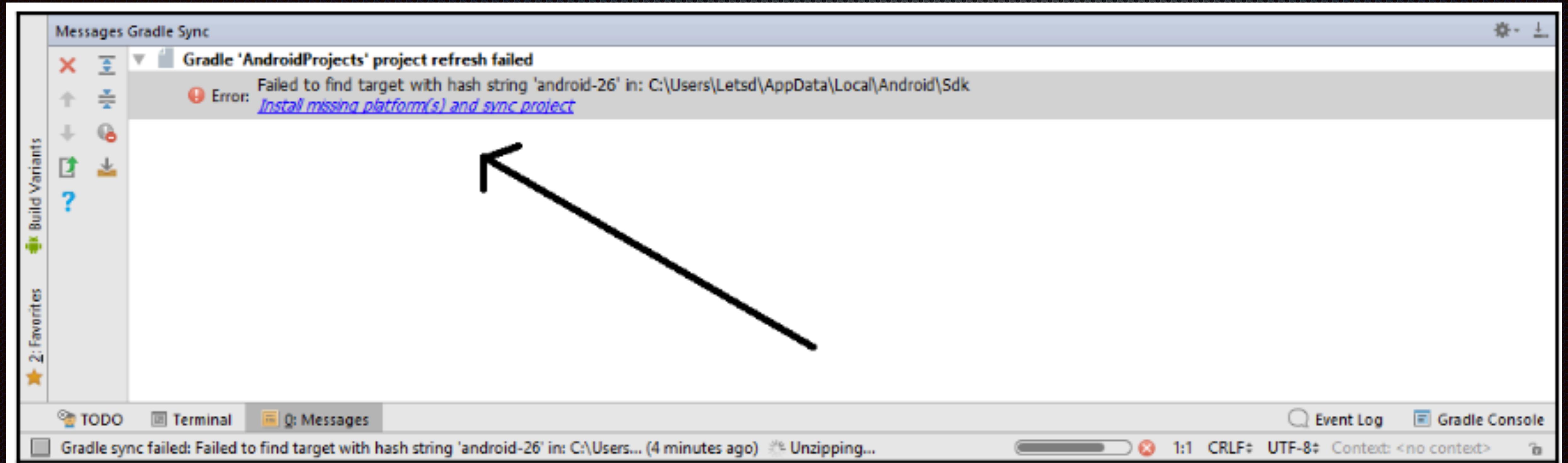
- Quando criamos um projeto novo (primeira opção da tela anterior), devemos escolher qual tipo de projeto é inicialmente.



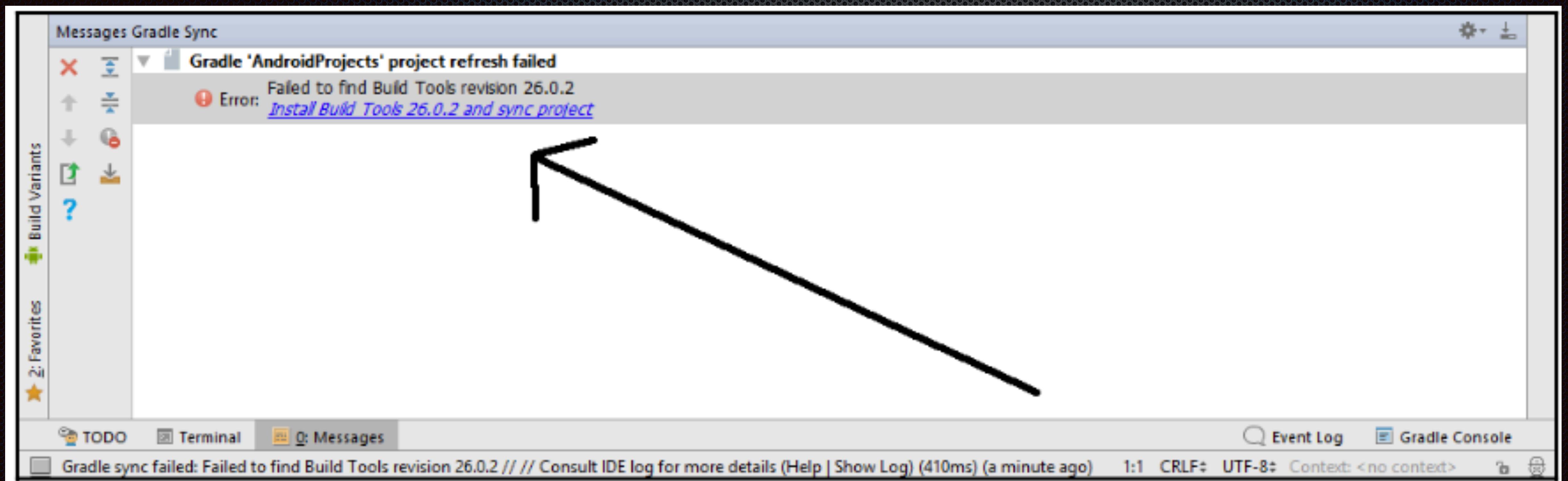
- Precisamos configurar o projeto.



- Quando o projeto terminar de ser criado, acontecerá um erro pois o sdk do Android ainda não foi baixado.



- É necessário clicar no link em azul para instalar.



# Android Virtual Device Manager

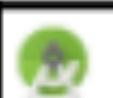
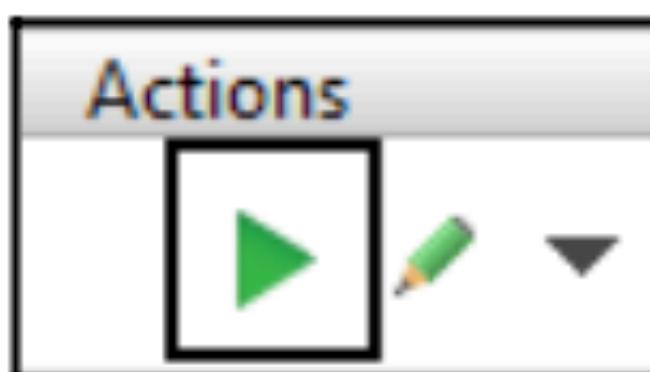


## Your Virtual Devices

Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
Pixel 2 XL API 28			1440 x 2880: 560dpi	28	Android 8.+ (Google APIs)	x86	10 GB	

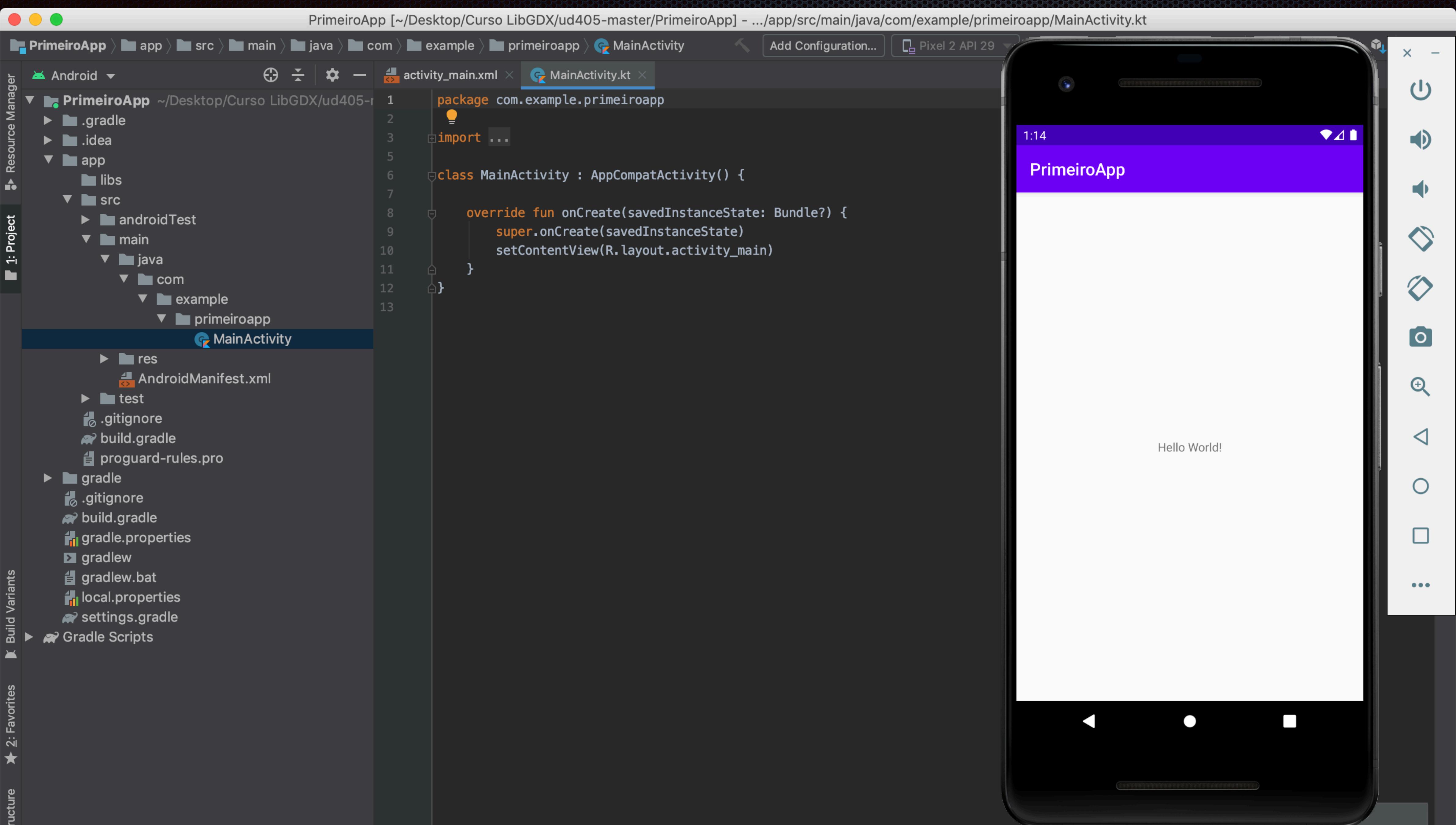
Create Virtual Device...



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help



# Este é o layout do nosso celular do emulador.



- Este é o código-fonte padrão em linguagem Kotlin que carrega a view.

```
1 package com.example.primeiroapp ✓  
2  
3 import androidx.appcompat.app.AppCompatActivity ← Importações necessárias  
4 import android.os.Bundle  
5  
6 class MainActivity : AppCompatActivity() {  
7  
8     override fun onCreate(savedInstanceState: Bundle?) {  
9         super.onCreate(savedInstanceState)  
10        setContentView(R.layout.activity_main)  
11    }  
12 }  
13 |
```

- Este é o código-fonte padrão em linguagem Kotlin que carrega a view.

```
1 package com.example.primeiroapp ✓  
2  
3 import androidx.appcompat.app.AppCompatActivity  
4 import android.os.Bundle  
5  
6 class MainActivity : AppCompatActivity() { ← Extendendo a Classe AppCompatActivity  
7  
8     override fun onCreate(savedInstanceState: Bundle?) {  
9         super.onCreate(savedInstanceState)  
10        setContentView(R.layout.activity_main)  
11    }  
12 }  
13 |
```

- Este é o código-fonte padrão em linguagem Kotlin que carrega a view.

```
1 package com.example.primeiroapp
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
13 |
```

Sobrescrevemos o método onCreate()  
Todo o que deve ser feito na criação do App

- Este é o código-fonte padrão em linguagem Kotlin que carrega a view.

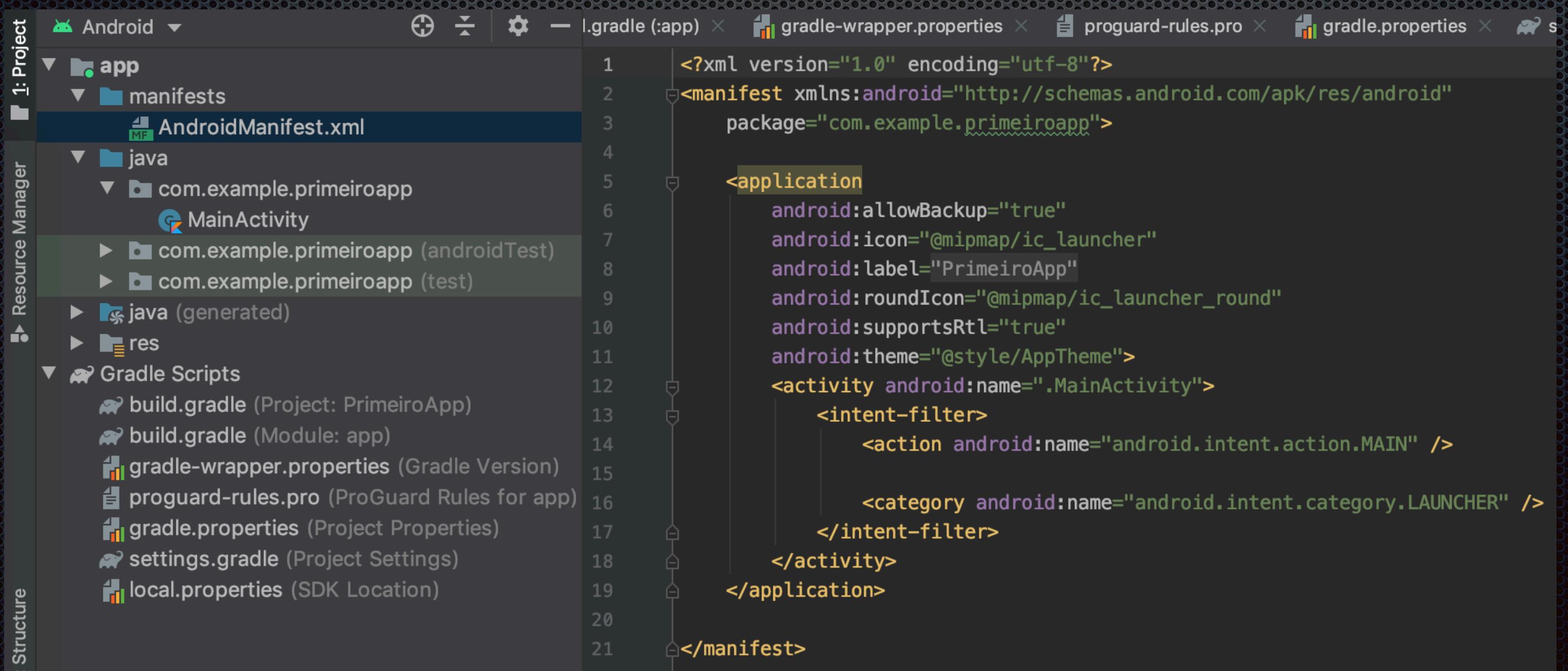
```
1 package com.example.primeiroapp
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
13 |
```



Carregamos o arquivo de layout (xml) com nossa UI (User Interface)

# Arquivo AndroidManifest.xml

- Este arquivo descreve algumas informações básicas sobre o nosso App.
- Caso nossa aplicação necessite de permissões especiais para rodar, deve estar declarado no manifesto.



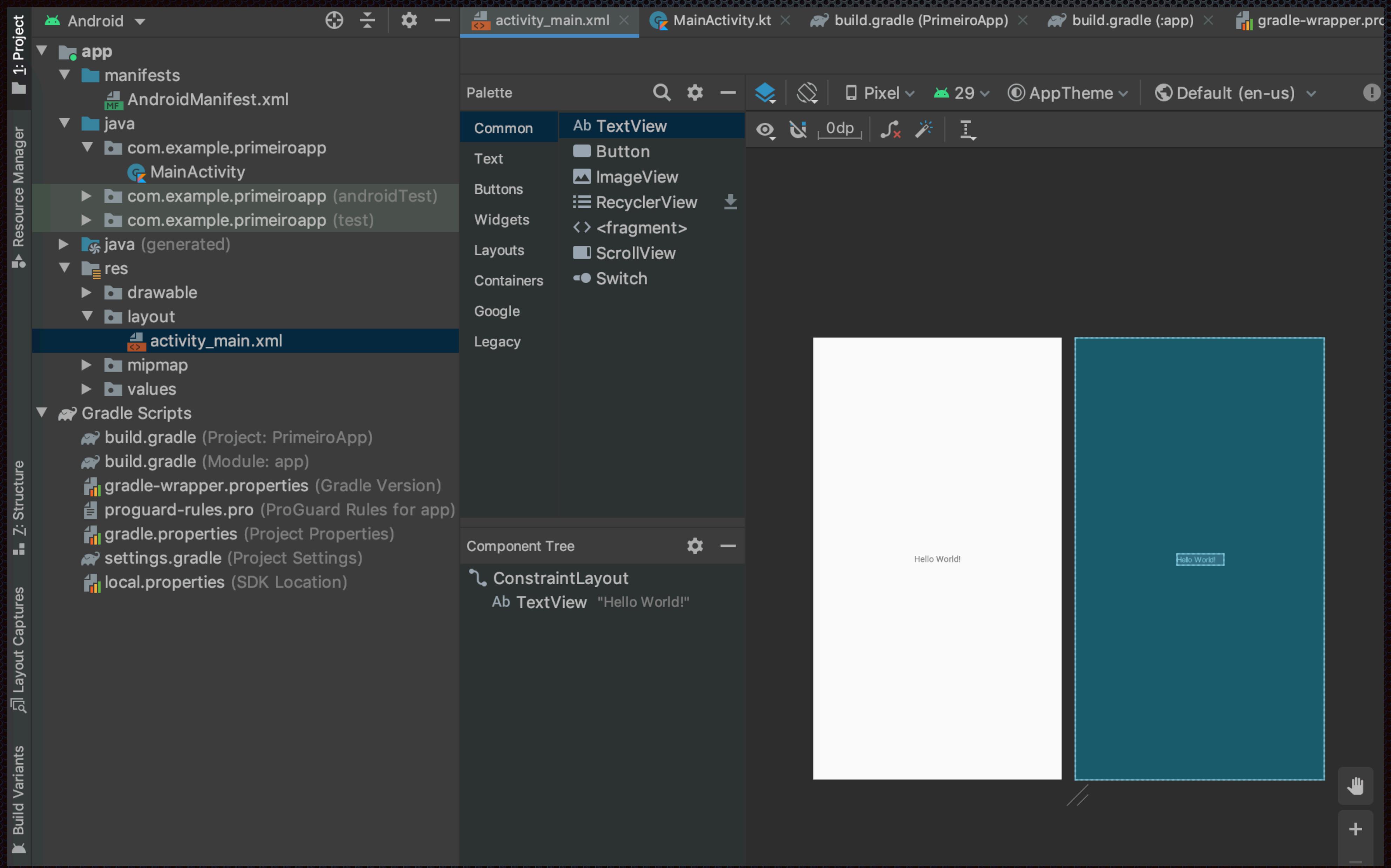
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.primeiroapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="PrimeiroApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3   package="com.example.primeiroapp">
4
5   <application
6     android:allowBackup="true"
7     android:icon="@mipmap/ic_launcher"
8     android:label="PrimeiroApp"
9     android:roundIcon="@mipmap/ic_launcher_round"
10    android:supportsRtl="true"
11    android:theme="@style/AppTheme">
12      <activity android:name=".MainActivity">
13        <intent-filter>
14          <action android:name="android.intent.action.MAIN" />
15
16          <category android:name="android.intent.category.LAUNCHER" />
17        </intent-filter>
18      </activity>
19    </application>
20
21 </manifest>
```

# AndroidManifest.xml

# Arquivo activity\_main.xml



# Podemos observar o texto XML

The screenshot shows the Android Studio interface with the following details:

- Top Bar:** Shows multiple open files: activity\_main.xml (selected), MainActivity.kt, build.gradle (PrimeiroApp), build.gradle (:app), gradle-wrapper.properties, proguard-rules.pro.
- Toolbar:** Includes icons for file operations, search, and navigation.
- activity\_main.xml Content:** XML code defining a ConstraintLayout with a TextView containing "Hello World!".

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```
- Preview Area:** Displays a ConstraintLayout with a single TextView centered in the middle, containing the text "Hello World!".
- Right Side Buttons:** A vertical stack of buttons for orientation, zoom, and other preview controls.
- Bottom Right Corner:** A small icon with a grid pattern.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 (c) <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/ap
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     <TextView
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Hello World!"
13        app:layout_constraintBottom_toBottomOf="parent"
14        app:layout_constraintLeft_toLeftOf="parent"
15        app:layout_constraintRight_toRightOf="parent"
16        app:layout_constraintTop_toTopOf="parent" />
17
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

# activity\_main.xml

Definindo a forma de disposição dos components (layout)



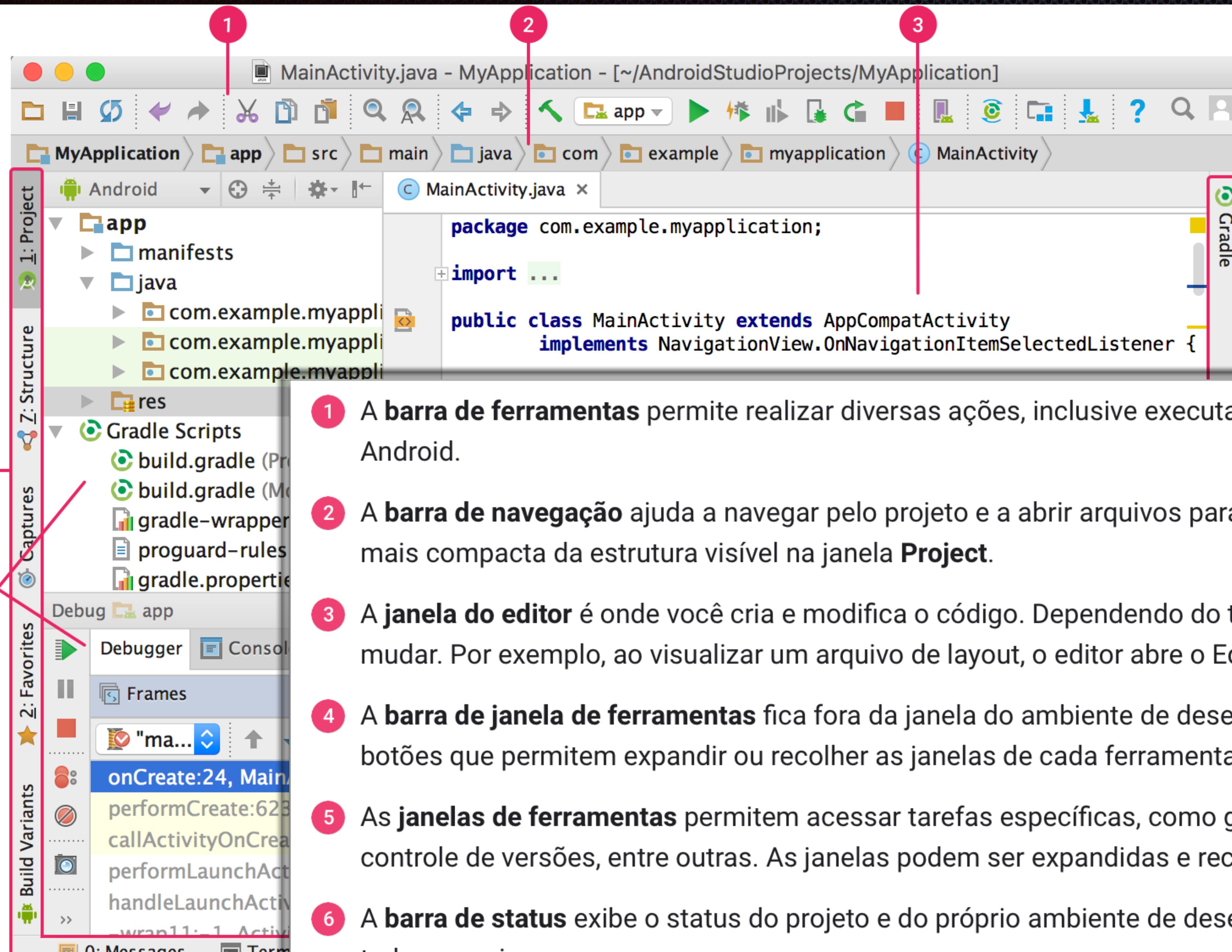
```
1 <?xml version="1.0" encoding="utf-8"?>           ✓
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/ap
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">← Definindo o arquivo xml que deve ser carregado para
8
9 <TextView
10    android:layout_width="wrap_content"
11    android:layout_height="wrap_content"
12    android:text="Hello World!"
13    app:layout_constraintBottom_toBottomOf="parent"
14    app:layout_constraintLeft_toLeftOf="parent"
15    app:layout_constraintRight_toRightOf="parent"
16    app:layout_constraintTop_toTopOf="parent" />
17
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

# activity\_main.xml



```
1 <?xml version="1.0" encoding="utf-8"?>           ✓
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/ap
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     <TextView
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="Hello World!"←          Componente de texto
13        app:layout_constraintBottom_toBottomOf="parent"
14        app:layout_constraintLeft_toLeftOf="parent"
15        app:layout_constraintRight_toRightOf="parent"
16        app:layout_constraintTop_toTopOf="parent" />
17
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

# activity\_main.xml



- 1 A **barra de ferramentas** permite realizar diversas ações, inclusive executar apps e inicializar ferramentas do Android.
- 2 A **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 A **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 Editor de layout.
- 4 A **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 As **janelas 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 A **barra de status** exibe o status do projeto e do próprio ambiente de desenvolvimento integrado, bem como todos os avisos ou mensagens.

# Alguns Atalhos

Janela de ferramentas	Windows e Linux	Mac
Projeto	Alt + 1	Command + 1
Controle de versões	Alt + 9	Command + 9
Executar	Shift + F10	Control + R
Depurar	Shift + F9	Control + D
Logcat	Alt + 6	Command + 6
Voltar ao editor	Esc	Esc
Ocultar todas as janelas de ferramentas	Control + Shift + F12	Command + Shift + F12

# Como funciona um Projeto Gradle?

## Sistema de compilação Gradle

O Android Studio usa o Gradle como o sistema de compilação de base, com outros recursos específicos do Android disponibilizados pelo [Plug-in do Android para Gradle](#). Esse sistema de compilação é executado como uma ferramenta integrada no menu do Android Studio e de forma independente na linha de comando. Você pode usar os recursos do sistema de compilação para fazer o seguinte:

- Personalizar, configurar e ampliar o processo de programação.
- Criar diversos APKs para seu app com diferentes recursos usando o mesmo projeto e os mesmos módulos.
- Reutilizar código e recursos nos conjuntos de origem.

A flexibilidade do Gradle permite que você faça tudo isso sem modificar os arquivos de origem principais do seu app. Os arquivos de versão do Android Studio recebem o nome de `build.gradle`. Eles são arquivos de texto simples que usam a sintaxe do [Groovy](#) para configurar a criação com elementos fornecidos pelo Plug-in do Android para Gradle. Cada projeto tem um arquivo de compilação de nível superior para todo o projeto e arquivos de compilação de módulo separados para cada módulo. Quando você importa um projeto existente, o Android Studio gera automaticamente os arquivos de compilação versão.

Para saber mais sobre o sistema de compilação e como configurá-lo, consulte [Configurar sua compilação](#).

# Estrutura do Projeto Android

## A visualização de projeto Android

Para ver a estrutura de arquivos real do projeto, incluindo todos os arquivos ocultos na visualização do Android, selecione **Project** no menu suspenso na parte superior da janela Project.

Quando você seleciona a visualização **Project**, pode ver um número muito maior de arquivos e diretórios. Os mais importantes são:

**module-name/**

**build/**

Contém saídas de compilação.

**libs/**

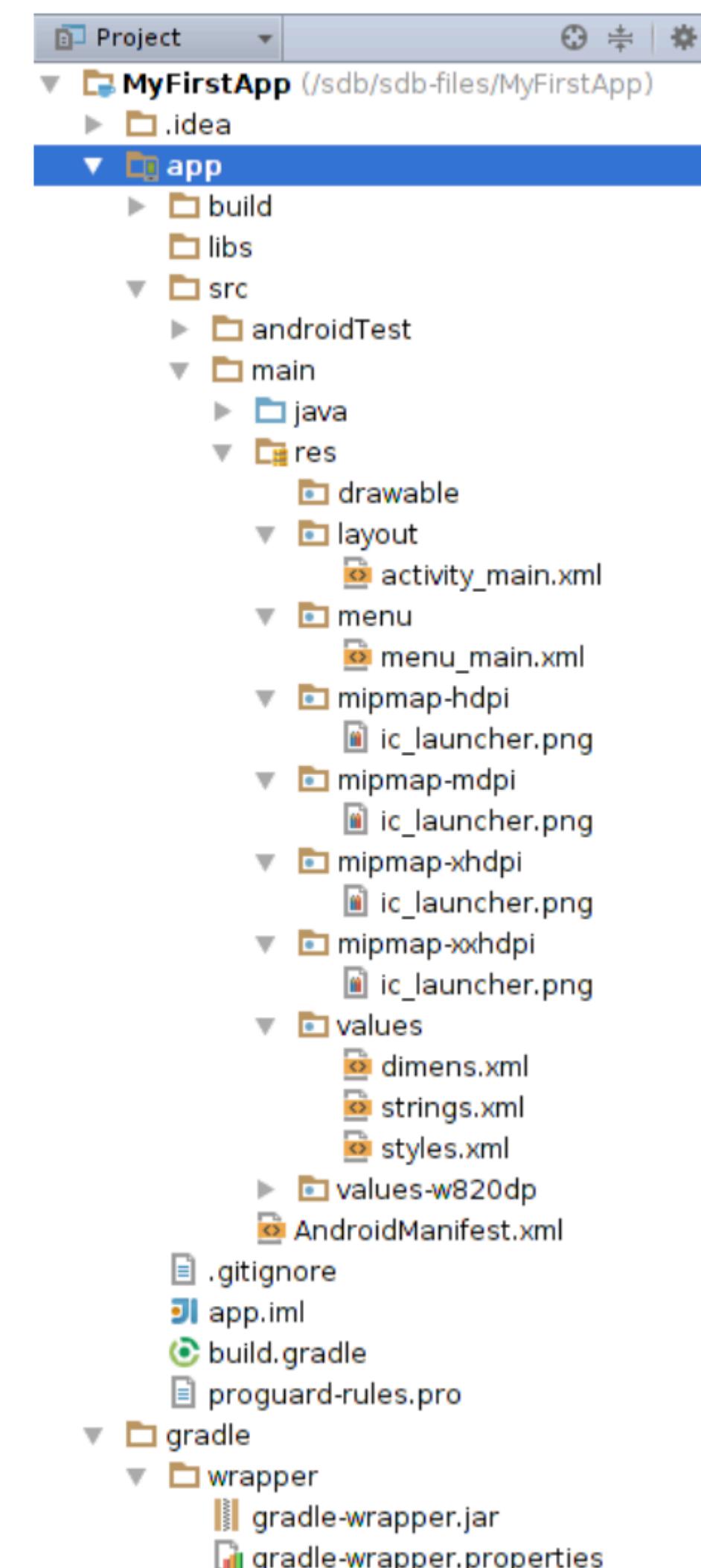
Contém bibliotecas privadas.

**src/**

Contém todos os arquivos de código e recursos do módulo nos seguintes subdiretórios:

**androidTest/**

Contém o código dos testes de instrumentação executados em um dispositivo Android. Para ver mais informações, consulte a [documentação do Android Test](#).



# Estrutura do Projeto Android

## main/

Contém os arquivos do conjunto de origem "main": o código Android e os recursos compartilhados por todas as variantes de compilação. Os arquivos para outras variantes de compilação ficam em diretórios irmãos, como `src/debug/` para o tipo de compilação de depuração.

## AndroidManifest.xml

Descreve a natureza do aplicativo e de cada um dos componentes dele. Para ver mais informações, consulte a documentação [AndroidManifest.xml](#).

## java/

Contém os códigos-fonte Java.

## jni/

Contém o código nativo que usa a Java Native Interface (JNI). Para ver mais informações, consulte a [documentação do Android NDK](#).

## gen/

Contém os arquivos Java gerados pelo Android Studio, como o arquivo `R.java` e as interfaces criadas de arquivos AIDL.

# Estrutura do Projeto Android

`res/`

Contém recursos de aplicativos, como arquivos drawable, arquivos de layout e strings de IU.  
Consulte [Recursos de aplicativo](#) para ver mais informações.

`assets/`

Contém o arquivo que precisa ser compilado em um arquivo `.apk` no estado em que está. Você pode navegar nesse diretório da mesma maneira que um sistema de arquivos típico usando URLs e ler arquivos como um fluxo de bytes usando [AssetManager](#) . Por exemplo, esse é um bom local para texturas e dados de jogos.

`test/`

Contém código para testes locais executados na JVM host.

**`build.gradle` (módulo)**

Define as configurações de compilação do módulo.

**`build.gradle` (projeto)**

Define a configuração de compilação que se aplica a todos os módulos. Esse arquivo faz parte do projeto, então precisa ser mantido no controle de revisões em conjunto com todo o código-fonte restante.

Para saber mais sobre outros arquivos de compilação, consulte [Configurar sua compilação](#).

# Rodando o App em um Celular Real

# Como rodar um App em um celular?

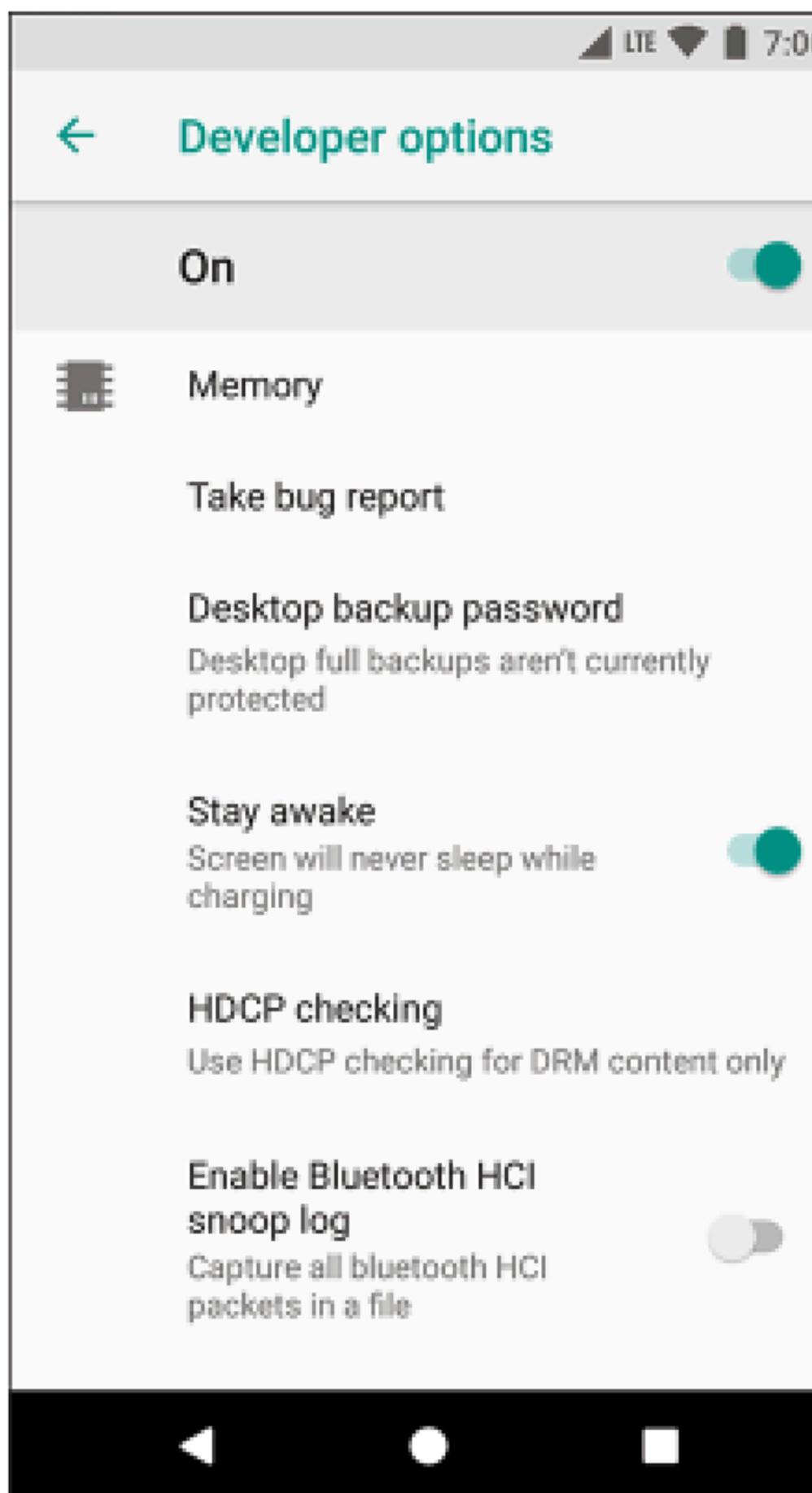
1. Deve-se acessar o site do fabricante do seu celular para verificar se não é necessário baixar e instalar drivers para o respectivo celular em sua máquina (principalmente se for um celular antigo e no Windows).
2. Antes de conectar o seu celular via USB na máquina, precisa acessar suas configurações e habilitar o modo desenvolvedor (Developer Options). Ele deve estar no modo DEBUG.
3. Conecte via USB o seu celular.
4. Instale o seu App no celular. Caso não tenha sucesso, reconstrua o projeto (rebuild)

For ADB to recognize your device as a target for deploying debuggable APKs, you must first enable USB debugging in the on-device developer options.

Depending on the version of Android you're using, proceed as follows:

- On Android 8.0 and higher, go to **Settings > System > About phone** and tap **Build number** seven times.
- On Android 4.2 through 7.1.2, go to **Settings > About phone** and tap **Build number** seven times.

Return to the main **Settings** menu to find **Developer options** at the bottom. In the **Developer options** menu, scroll down and enable **USB debugging**.



# LINKS

<https://www.oracle.com/java/technologies/javase-downloads.html>

<https://developer.android.com/studio>

<https://developer.android.com/studio/run/emulator.html>