Google Android

Uma abordagem prática e didática

Rafael Guimarães Sakurai

Google Android

Uma abordagem prática e didática

Rafael Guimarães Sakurai

Esse livro está à venda em http://leanpub.com/google-android

Essa versão foi publicada em 2018-03-02

Conteúdo

1. Olá Android!

Vamos criar uma aplicativo inicial, no estilo **Olá Mundo!** para Android. O objetivo é apresentar como criar um projeto Android e utilizar uma `Activity` para controlar a tela e os componentes `TextView`, `ImageView` e `Toast`.

O aplicativo terá um texto `Olá Android!!!` e a figura do logo do Android, também será adicionado uma ação na figura que ao clicar nela aparece uma mensagem `Olá!`. No final, o aplicativo ficará conforme apresentado na `Figura 1.1`.

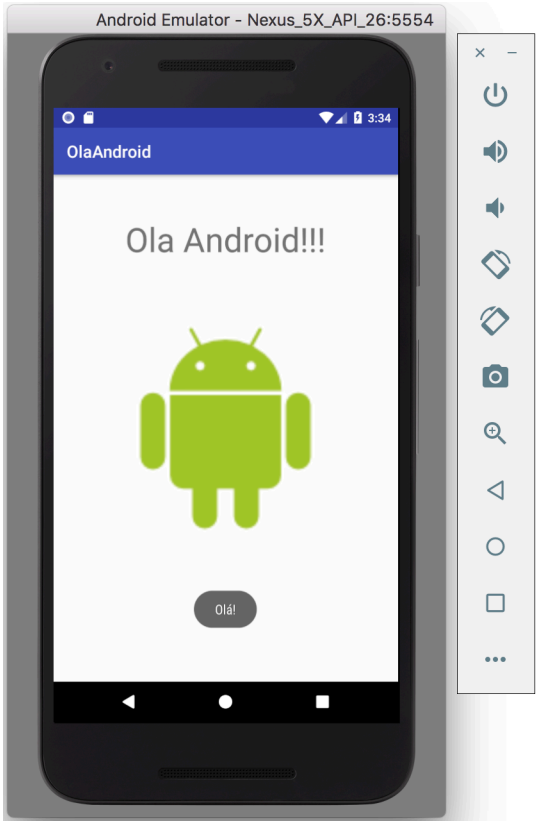

Figura 1.1 - Aplicativo Olá Android!

Para desenvolvimento dos aplicativos Android utilizaremos a IDE Android Studio[1].

[1] http://developer.android.com/sdk/index.html

1.1 Criando um novo projeto no Android Studio

Para criar um projeto, selecione a opção **Start a new Android Studio project (Iniciar um novo projeto Android Studio)** na tela de bem vindo ao Android Studio, como apresentado na `Figura 1.1.1`.

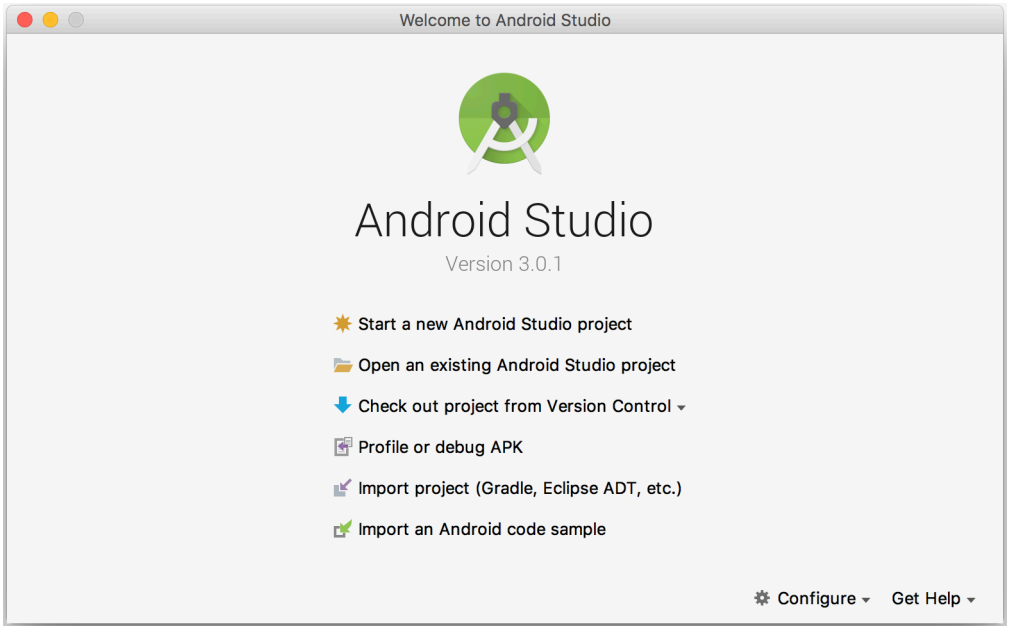

Figura 1.1.1 - Android Studio.

Na tela **Create Android Project**, apresentada na `Figura 1.1.2`, configuramos as informações básicas do projeto, como:

- **Application name (Nome do aplicativo)**: OlaAndroid;
- **Company Domain (Domínio da empresa)**: seudominio.com.br, exemplo: googleandroid.leanpub.com ou se preferir livro.capitulo1;
- **Project location (Localização do projeto)**: Caminho do projeto no seu computador;

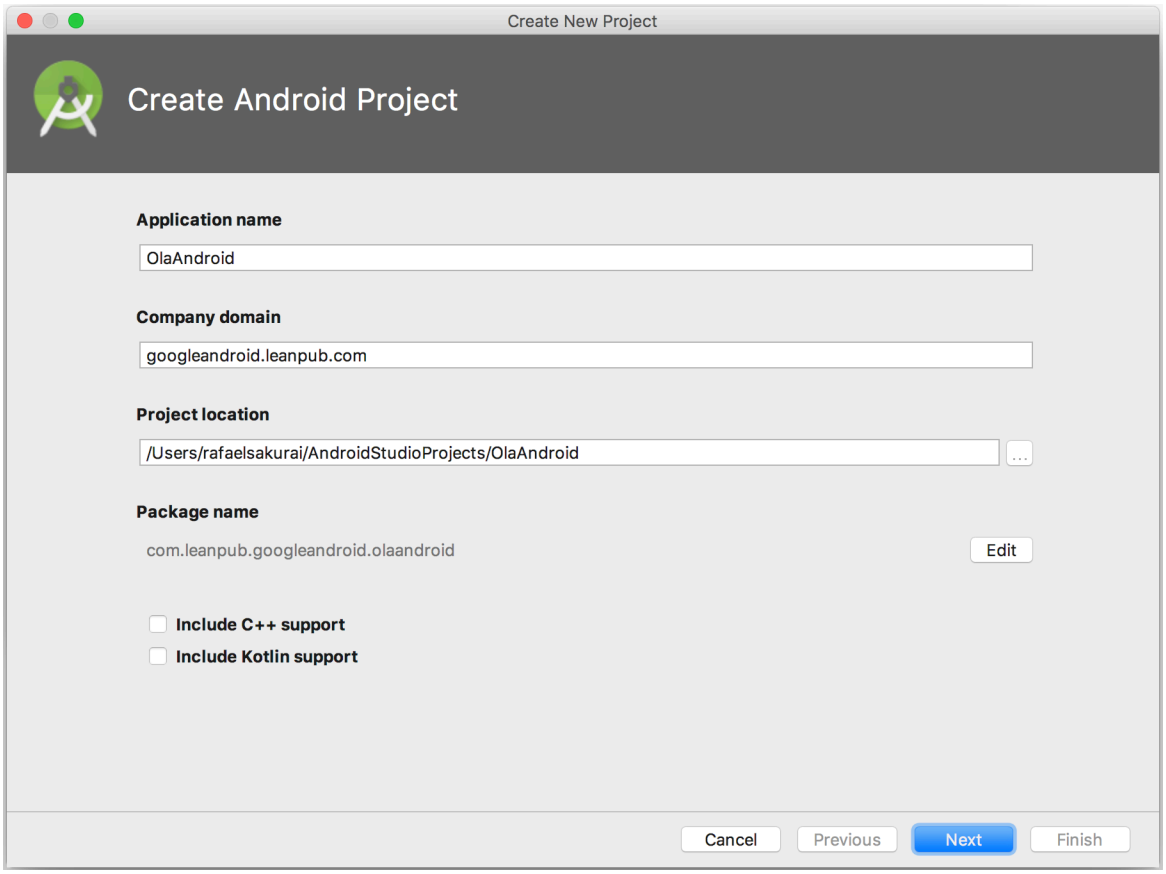

Figura 1.1.2 - Criando um novo projeto Android.

Continuando, a tela **Target Android Devices (Dispositivos alvo do Android)**, apresentada na `Figura 1.1.3`, permite escolher qual será o foco principal do aplicativo, que pode ser Celular e Tablet; Wear (como o relógio); TV; Android Auto (uso em automóveis); e para Android Things.

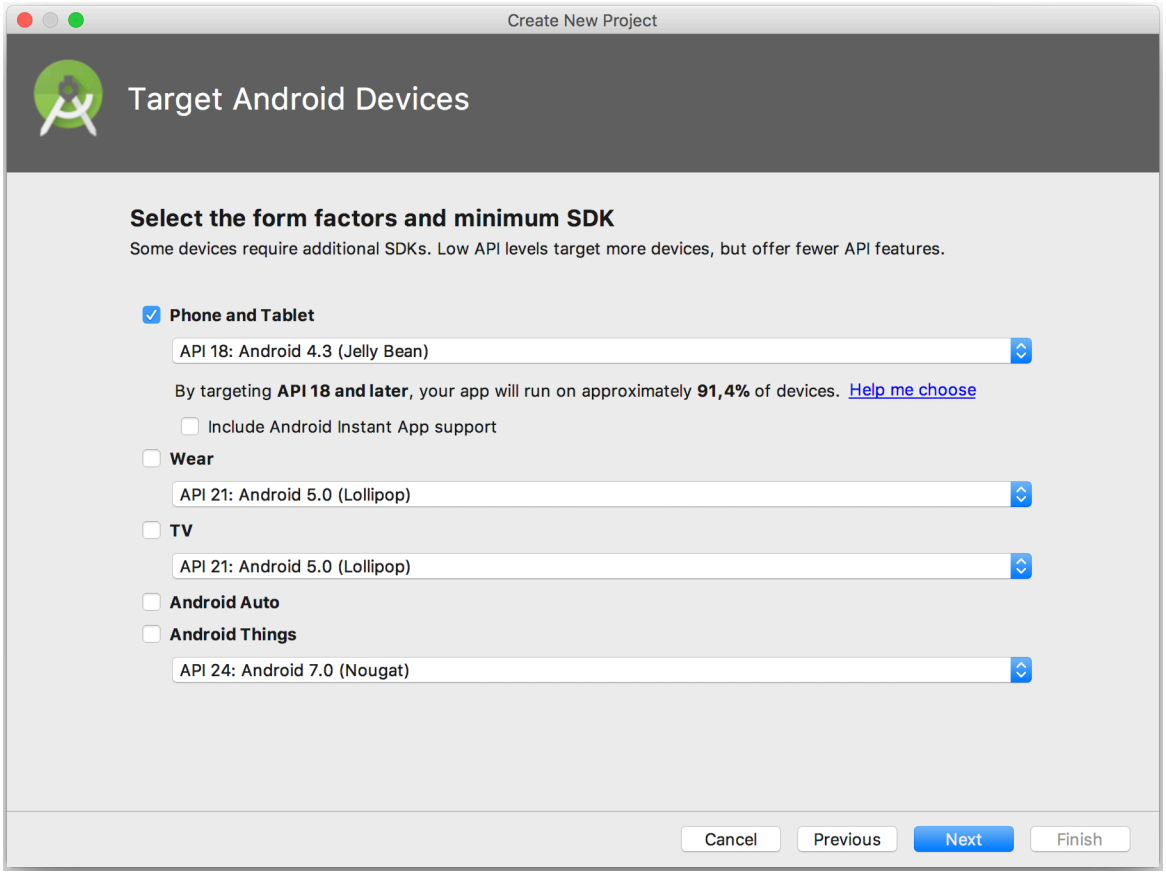

Figura 1.1.3 - Escolhendo o dispositivo e versão mínima do Android.

Nesse exemplo, escolha a opção **Phone and Tablet**, também defina qual será a API mínima para poder executar o aplicativo, como a **API 18: Android 4.3 (Jelly Bean)**.

Clicando em **Next**, vamos continuar a criação do projeto. Na tela **Add an Activity to Mobile (Adicionar uma Activity ao aplicativo móvel)**, apresentada na `Figura 1.1.4`, podemos informar se queremos criar uma **Activity** inicial para o projeto, de modo geral uma Activity é usada para realizar a ações de uma ou mais telas do aplicativo.

> Neste capítulo abordaremos apenas o básico para criar e iniciar um projeto, nos demais capítulos apresentamos outros modelos de Activities.

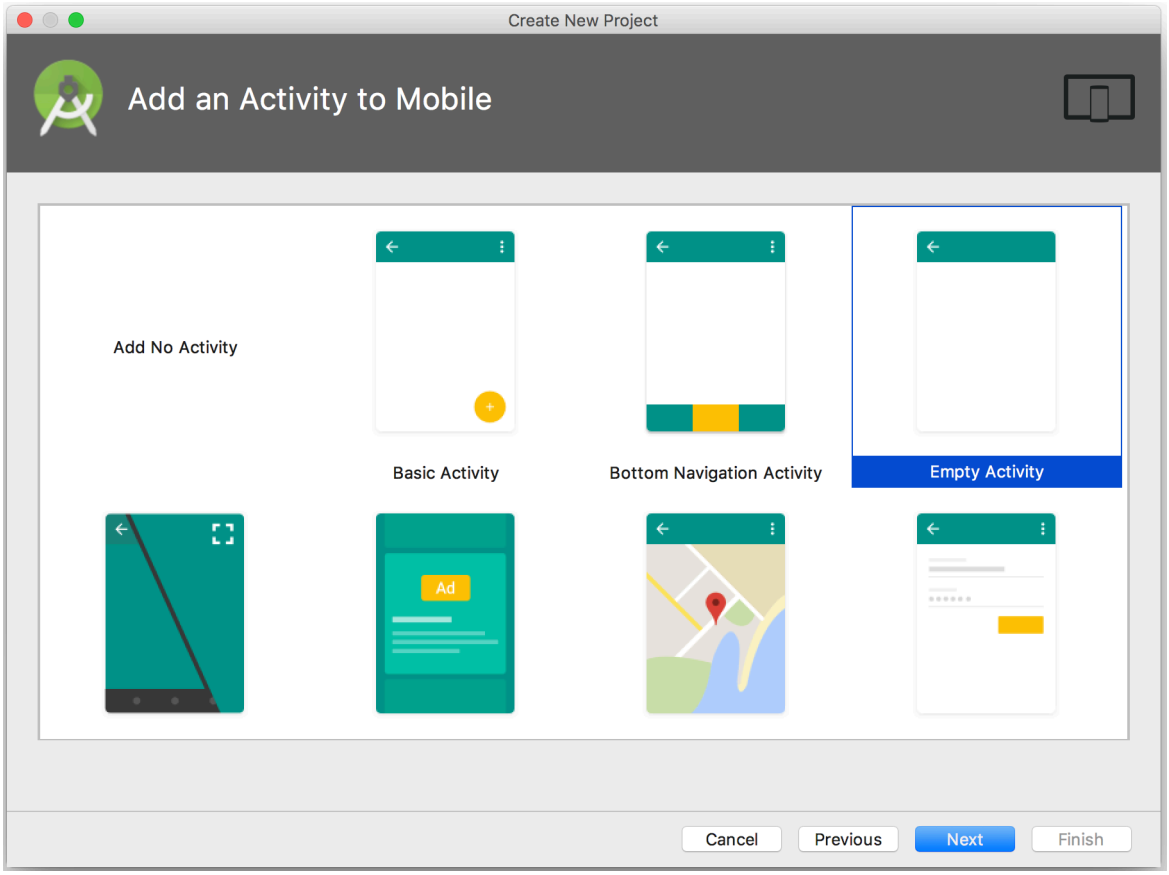

Figura 1.1.4 - Adicionando uma Activity ao projeto.

Marque a opção **Empty Activity (Activity em branco)** e clique em **Next** para prosseguir. Essa opção cria uma Activity e tela com um texto de exemplo, algo bem simples que a partir dele é possível continuar a implementação de diversos aplicativos.

Na tela **Configure Activity (Configurar Activity)**, apresentada na `Figura 1.1.5`, temos a opção de customizar os dados da Activity inicial que será criada no projeto. As informações que temos são:

- **Activity Name (Nome da Activity)**: MainActivity
- **Layout Name (Nome do layout)**: activity_main

A **Activity** representa o nome da classe que será criada para controlar as ações e atributos do layout da tela; e o **Layout** é um arquivo XML que possui a estrutura e componentes da tela.

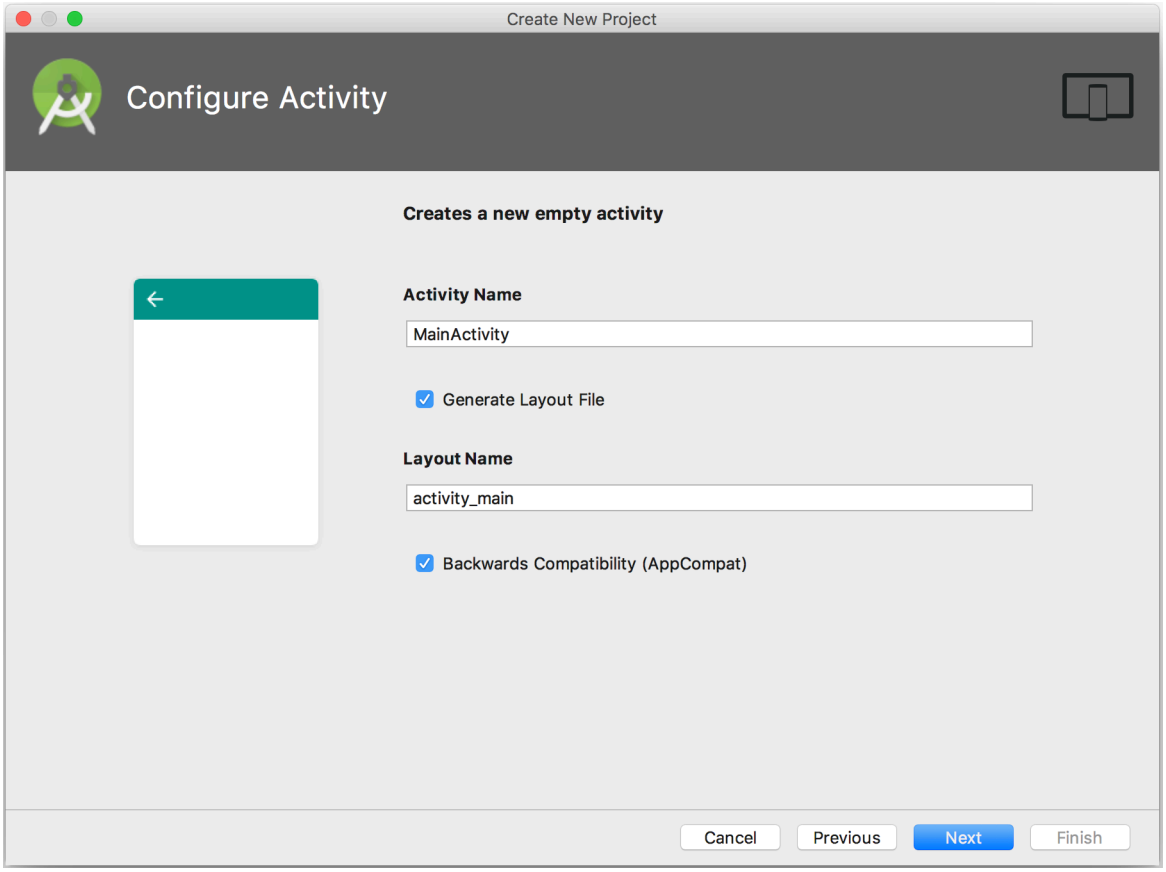

Figura 1.1.5 - Configurando a Activity inicial do projeto.

Na primeira vez que um projeto Android é criado, pode ser necessário fazer alguns downloads, como o apresentado na `Figura 1.1.6`.

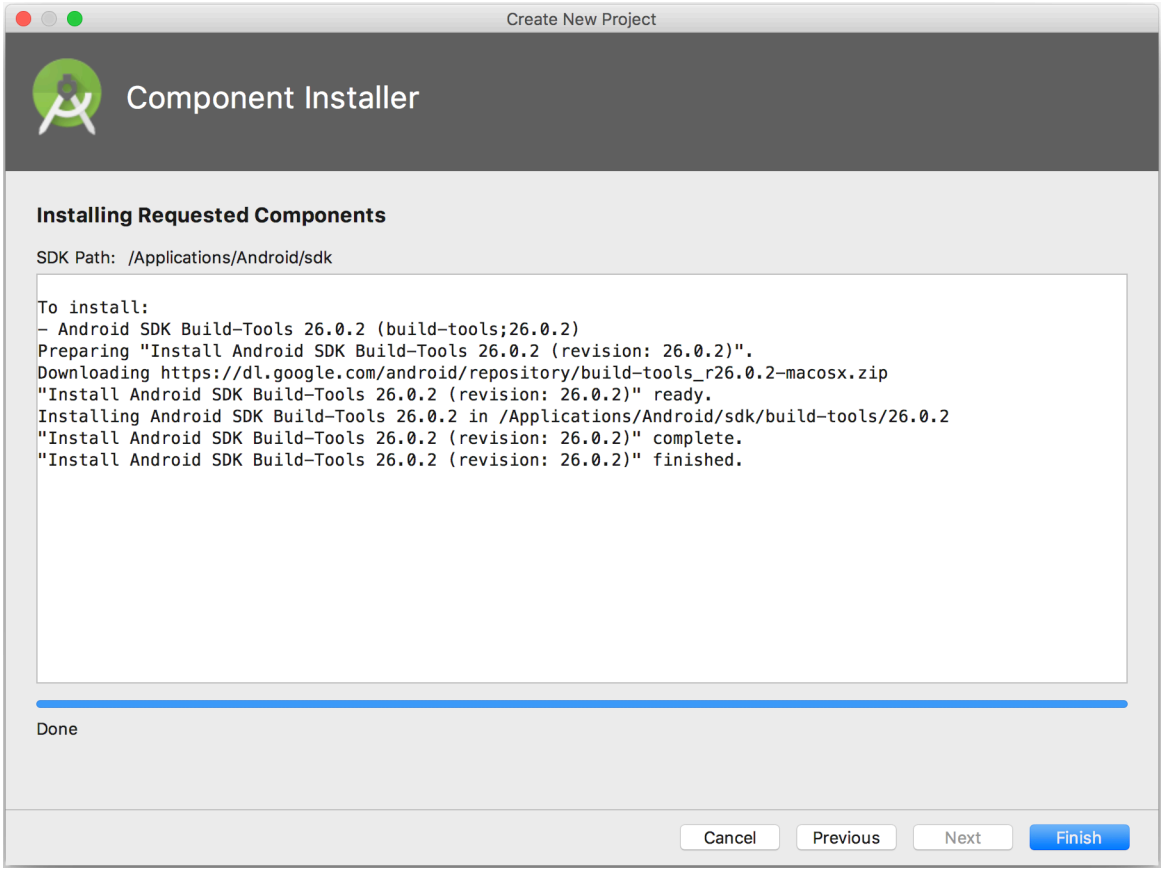

Figura 1.1.6 - Atualizações para o projeto Android.

Se após criar o projeto, aparecer uma mensagem de erro, como o da `Figura 1.1.7`, normalmente é porque tem mais alguma atualização necessária para compilar o projeto.

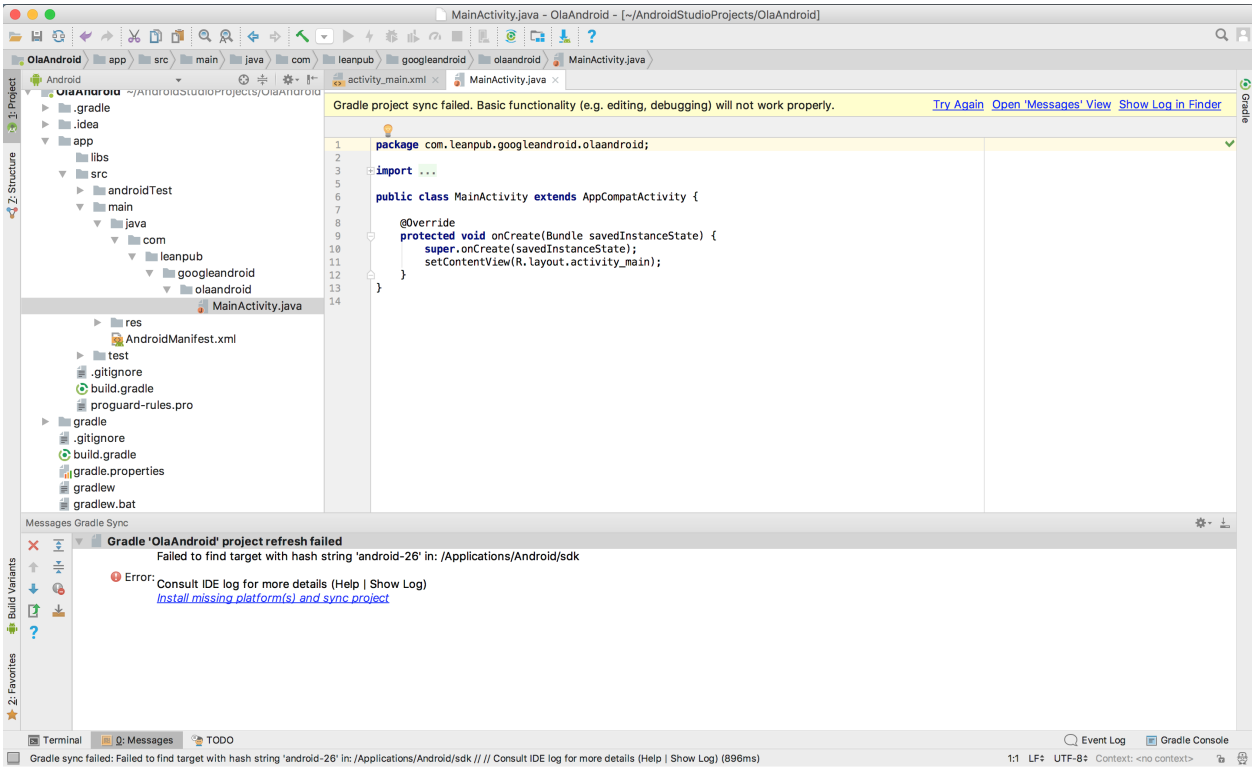

Figura 1.1.7 - Atualizações necessárias para compilar o projeto Android.

Então clique no link que automaticamente inicia as atualizações necessárias, como mostrado na `Figura 1.1.8`.

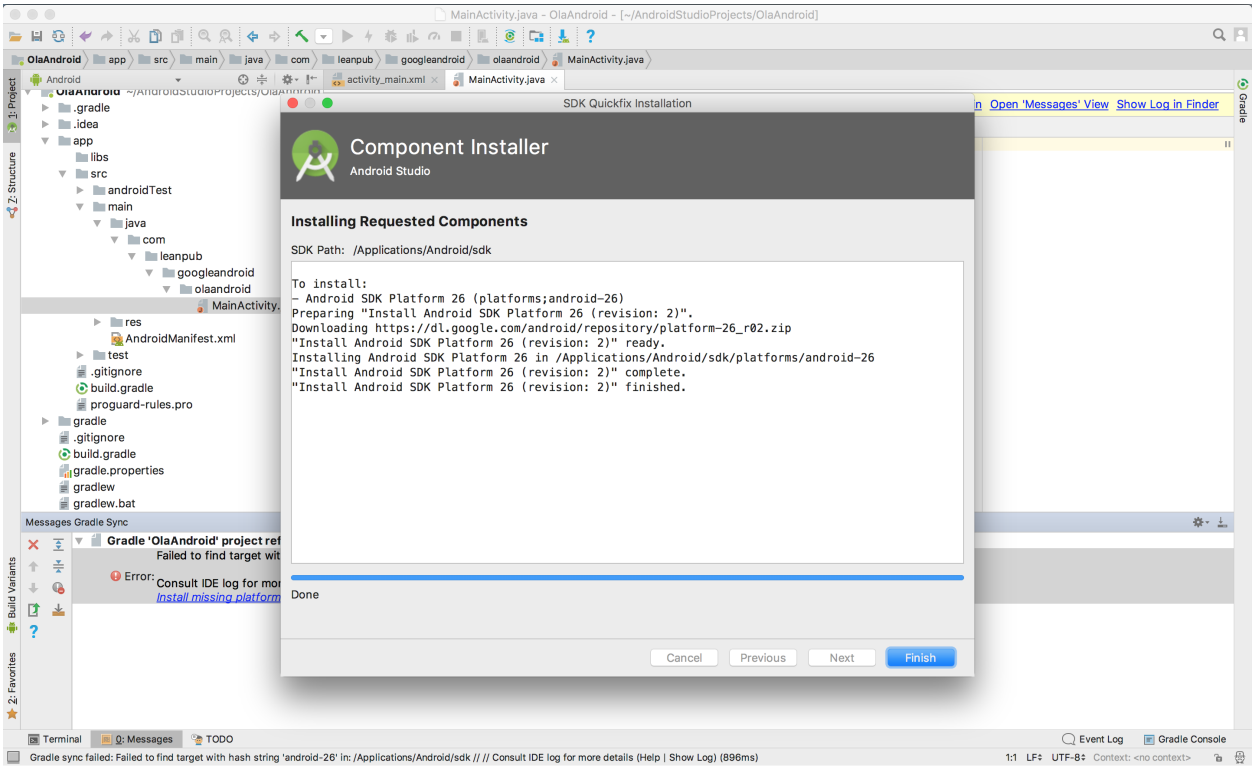

Figura 1.1.8 - Atualizações necessárias para compilar o projeto Android.

Após a criação do projeto e demais atualizações, o Android Studio abre uma visualização completa da estrutura do projeto, como mostrado na `Figura 1.1.9`.

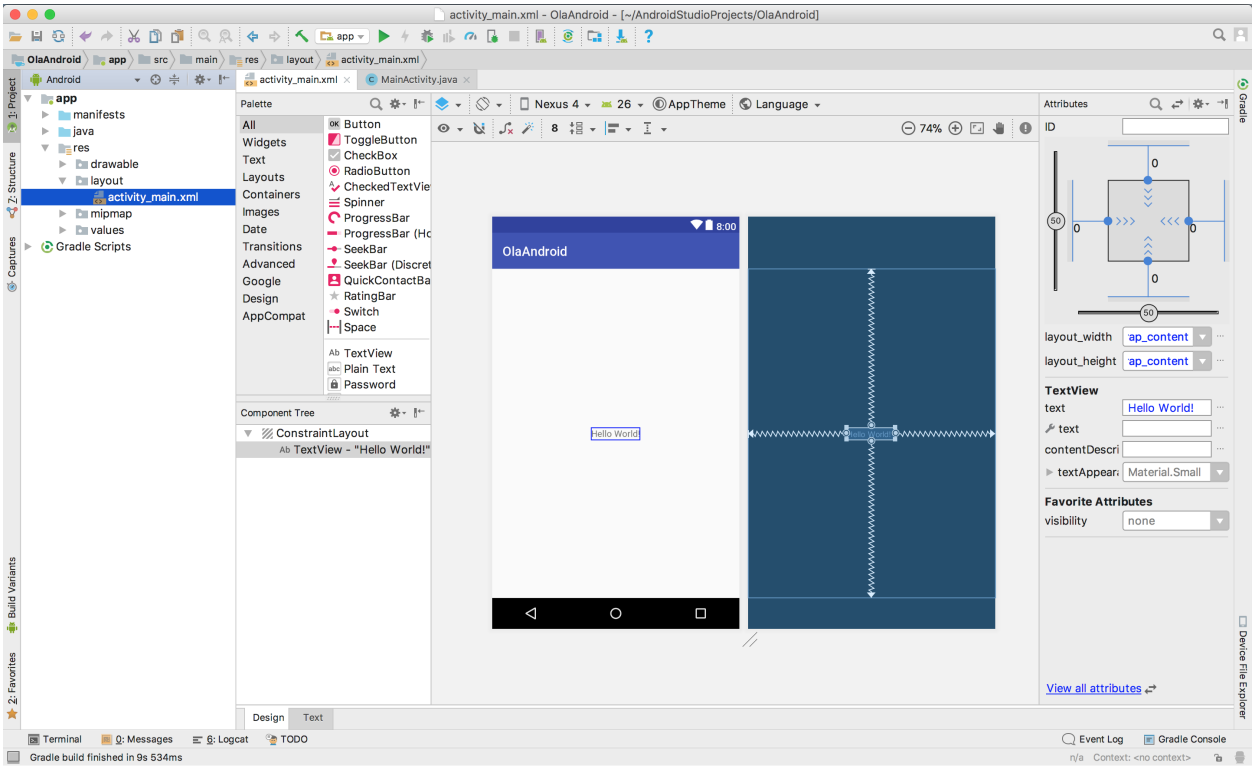

Figura 1.1.9 - Projeto aberto no Android Studio.

1.2 Executando um aplicativo Android

Para executar um aplicativo Android podemos utilizar um hardware físico como celular, tablet, etc, ou um emulador que simula o comportamento do aplicativo no dispositivo.

Na barra de tarefa do Android Studio (`Figura 1.2.1`), temos a opção de executar o aplicativo clicando botão verde de Play ou usando o atalho **CRTL + R**.

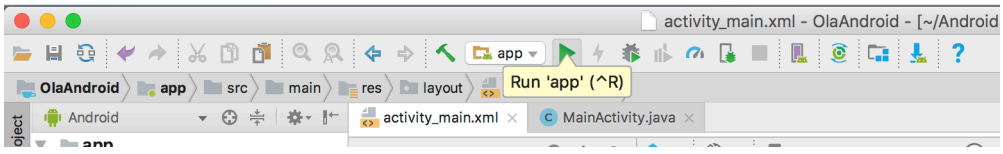

Figura 1.2.1 - Executar projeto Android.

A tela **Select Deployment Target (Selecione o alvo da publicação)**, apresentada na `Figura 1.2.2`, será aberta para escolher se deve executar o aplicativo no dispositivo (conectado via USB) ou no emulador.

> Se for utilizar seu smartphone ou tablet para testar os exemplos, ative no dispositivo a **Depuração de USB**, está opção está disponível em **Configuração → Opções do**

desenvolvedor. Isso é algo que inicialmente vem escondido, mas na Internet você encontra facilmente como ativar as **Opções do desenvolvedor** de acordo com o seu dispositivo.

Na figura a seguir, meu smartphone está conectado no computador e não tenho nenhum emuladores criado.

Se for utilizar seu dispositivo, selecione ele e click em **OK** para publicar este aplicativo diretamente no seu dispositivo. Para utilizar os emuladores do Android, clique no botão **Create New Virtual Device (Criar novo dispositivo virtual)**.

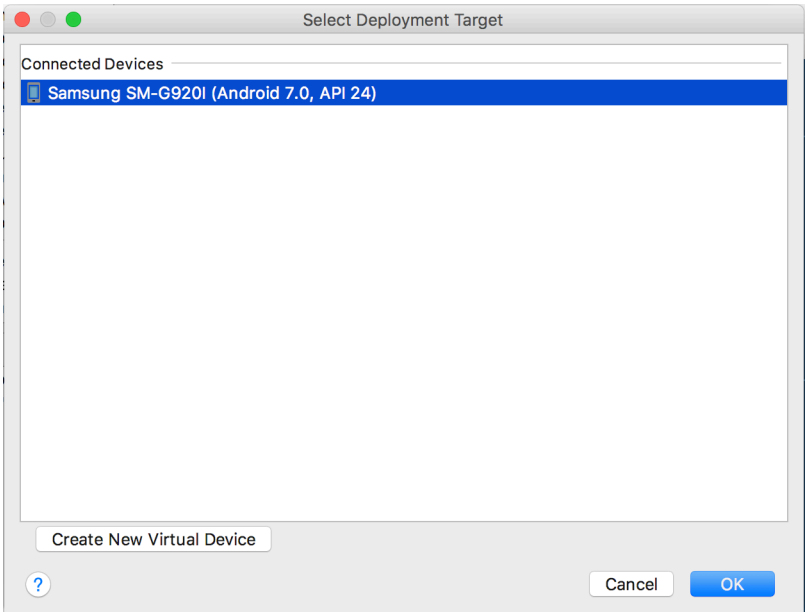

Figura 1.2.2 - Escolhendo um dispositivo virtual - emulador.

Ma tela **Virtual Device Configuration (Configuração virtual de dispositivos)**, apresentada na `Figura 1.2.3`, há as categorias do emulador como TV, Phone, Wear e Tablet, e para cada categoria temos a opção de usar as configurações de um hardware existente, como: Nexus 5, Nexus One, etc, ou um formato de acordo com o tamanho da tela 5.4", 4", etc.

A escolha da configuração do emulador deixa definir o tipo de hardware (tamanho, resolução e densidade) que será o foco do aplicativo (mais adiante no livro será apresentado o que significa cada uma dessas características e como criar um aplicativo que funcione de modo agradável em várias resoluções). Por agora você pode criar um emulador com base no hardware do **Nexus 5X** ou criar um emulador que simule o comportamento do seu smartphone.

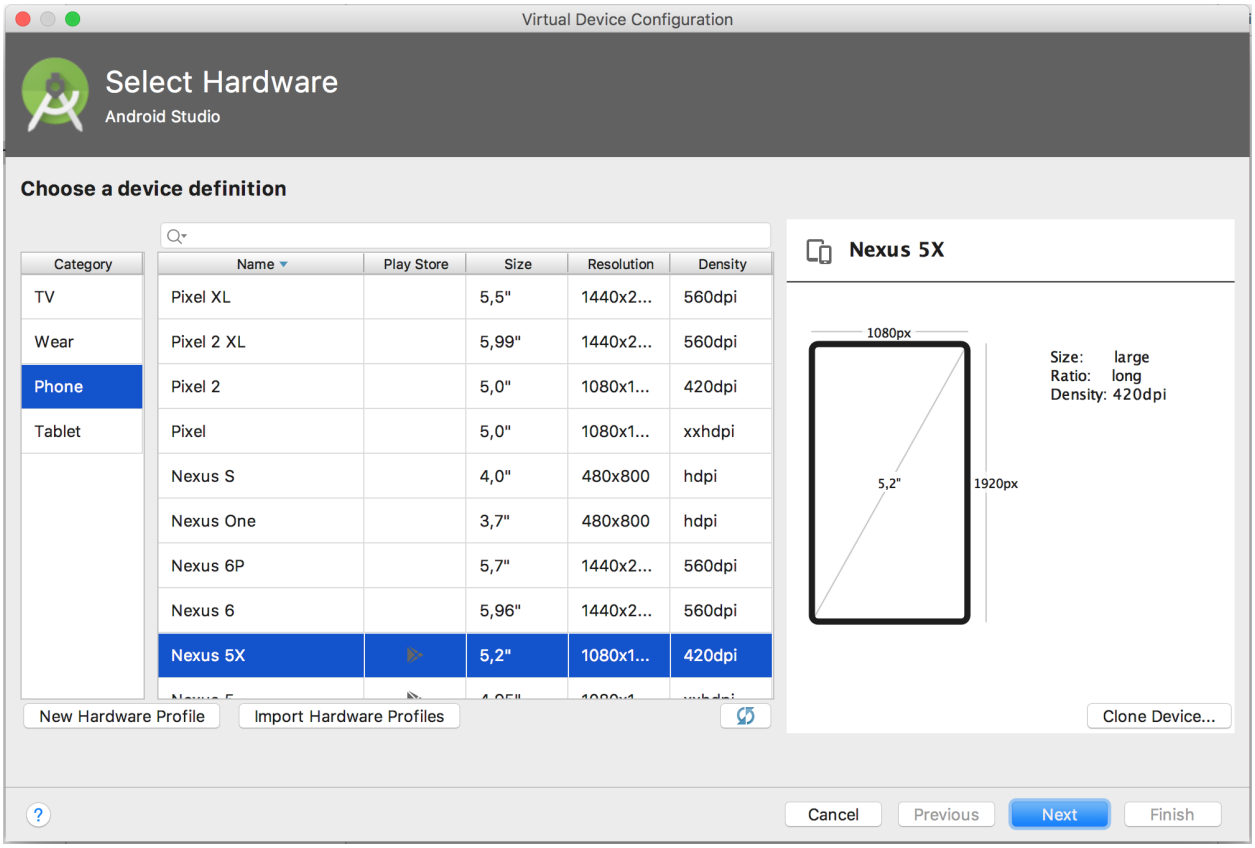

Figura 1.2.3 - Escolhendo o hardware do novo dispositivo virtual.

No segundo passo precisamos informar qual a **System Image (Imagem do Sistema)**, apresentado na `Figura 1.2.4`, será utilizado no emulador, nesse caso a versão do Android.

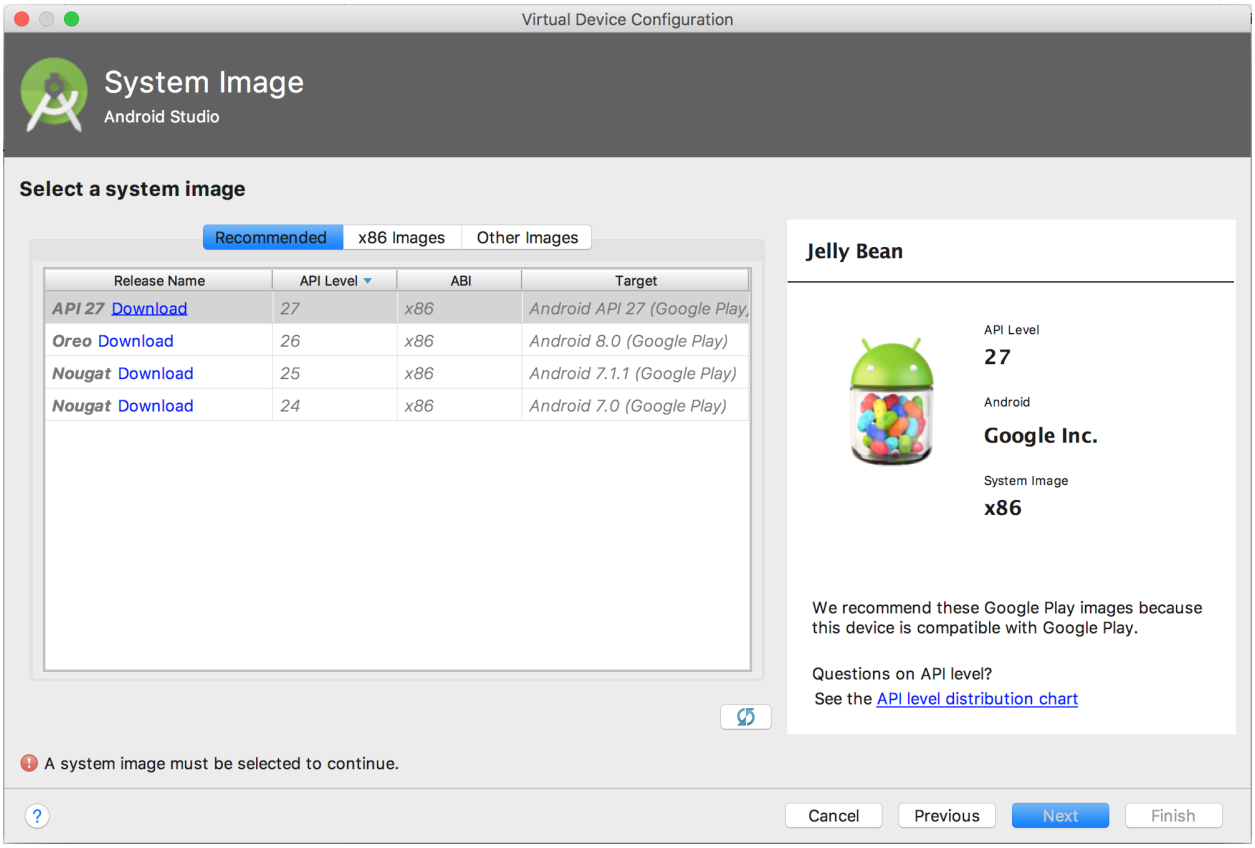

Figura 1.2.4 - Escolhendo a imagem (versão do Android) do dispositivo virtual.

Neste exemplo, não há nenhuma imagem do Android, por tanto aparece a opção para fazer o download. Nessa tela há três categorias de imagens: **Recommended (Recomendado)** são as versões mais recentes do Android, **x86 Images (Imagens x86)** são outras versões do Android para arquitetura de hardware **x86_64** e **Other images (Outras imagens)** são versões do Android para outros hardwares (normalmente antigos) como: armeabi, arm64 e mips, essas versões são mais lentas e podem ser utilizadas caso o seu computador não suporte as versões mais recentes com **x86** ou **x86_64**.

A instalação apresentada aqui será do **Oreo**, **API Level: 26**, **ABI: x86 e Target: Android 8.0 (Google Play)**, mas se preferir a instalação de outra versão é muito similar ao passo a passo apresentado a seguir.

> Sugiro não fazer download de todas as versões, cada uma tem em média 1GB de tamanho.

Quando for fazer o download de uma nova imagem do Android é apresentado a tela **License Agreement (Contrato de Licença)** (`Figura 1.2.5`), leia o contrato e se aceitar clique em **Accept (Aceitar)** e **Next (Próximo)** para continuar.

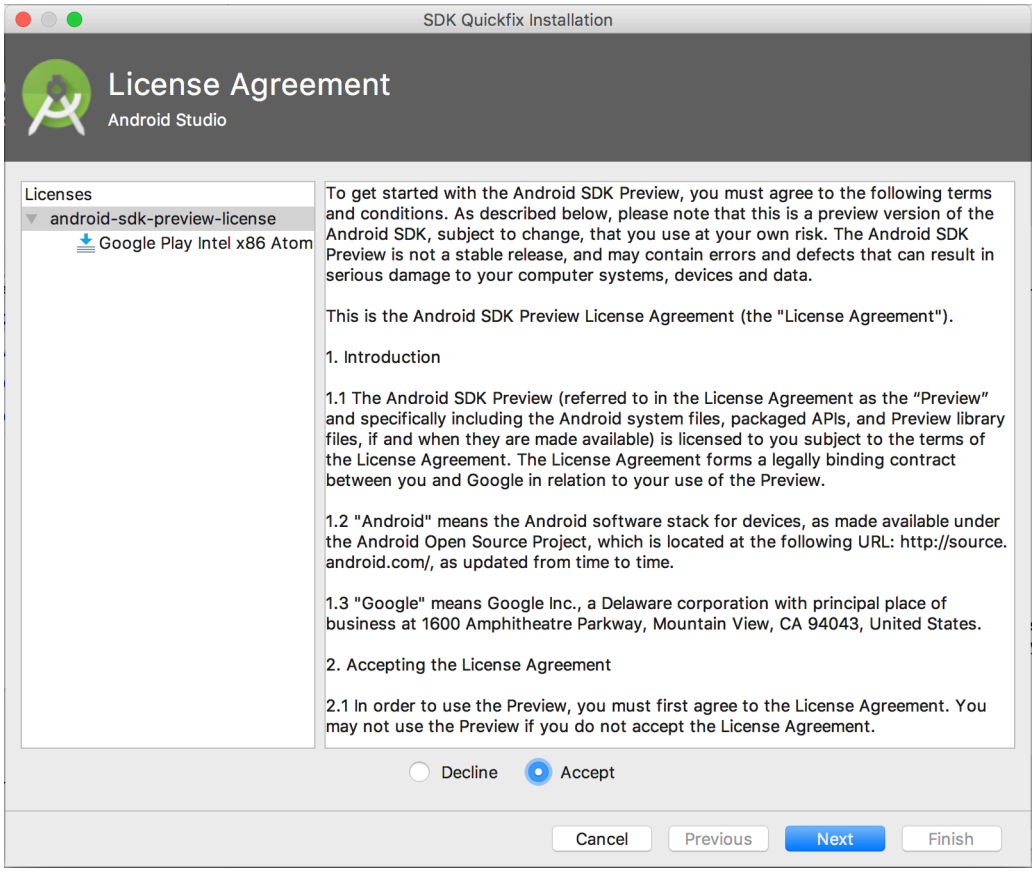

Figura 1.2.5 - Aceitando a licença de uso do Android.

A tela **Component Installer (Instalação dos componentes)** (`Figura 1.2.6`) apresenta os componentes instalados desta nova imagem do Android. Clique em **Finish (Finalizar)** para terminar a instalação desta versão.

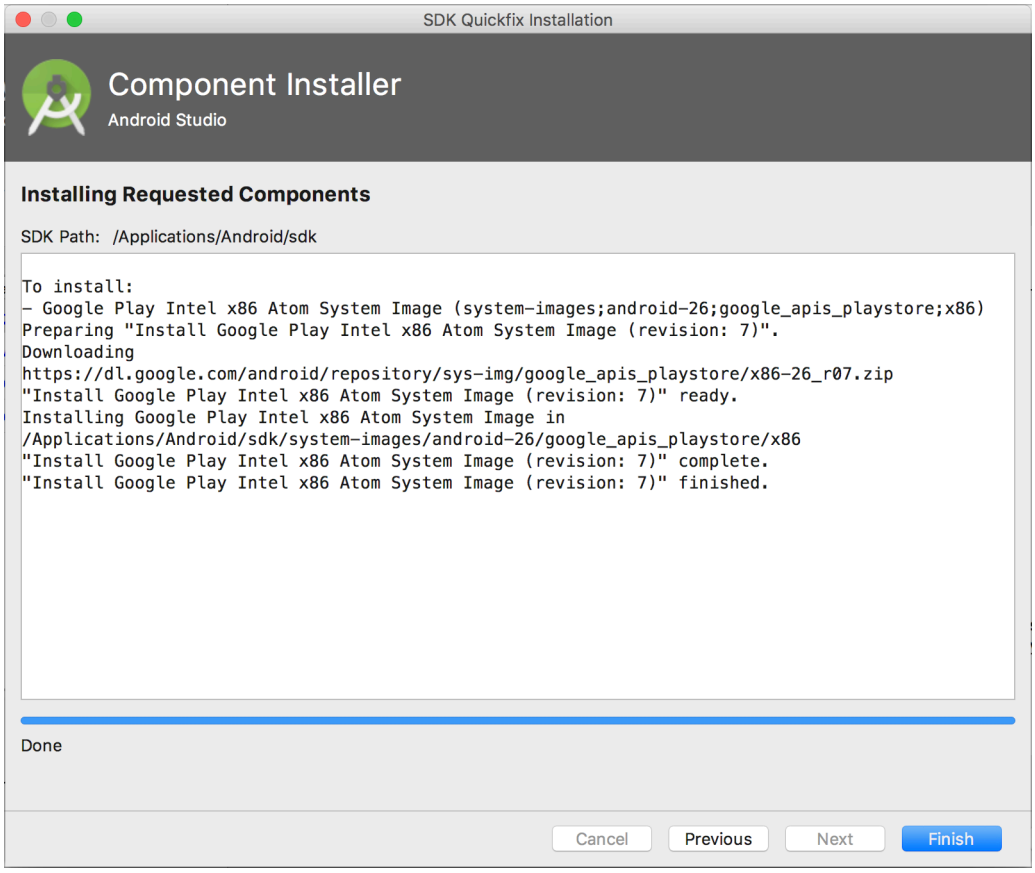

Figura 1.2.6 - Finalizando a instalação da imagem para o emulador.

Voltando para a tela **System Image (Imagem do Sistema)** (`Figura 1.2.7`), selecione a imagem que será utilizada nesse emulador e clica em **Next (Próximo)** para continuar.

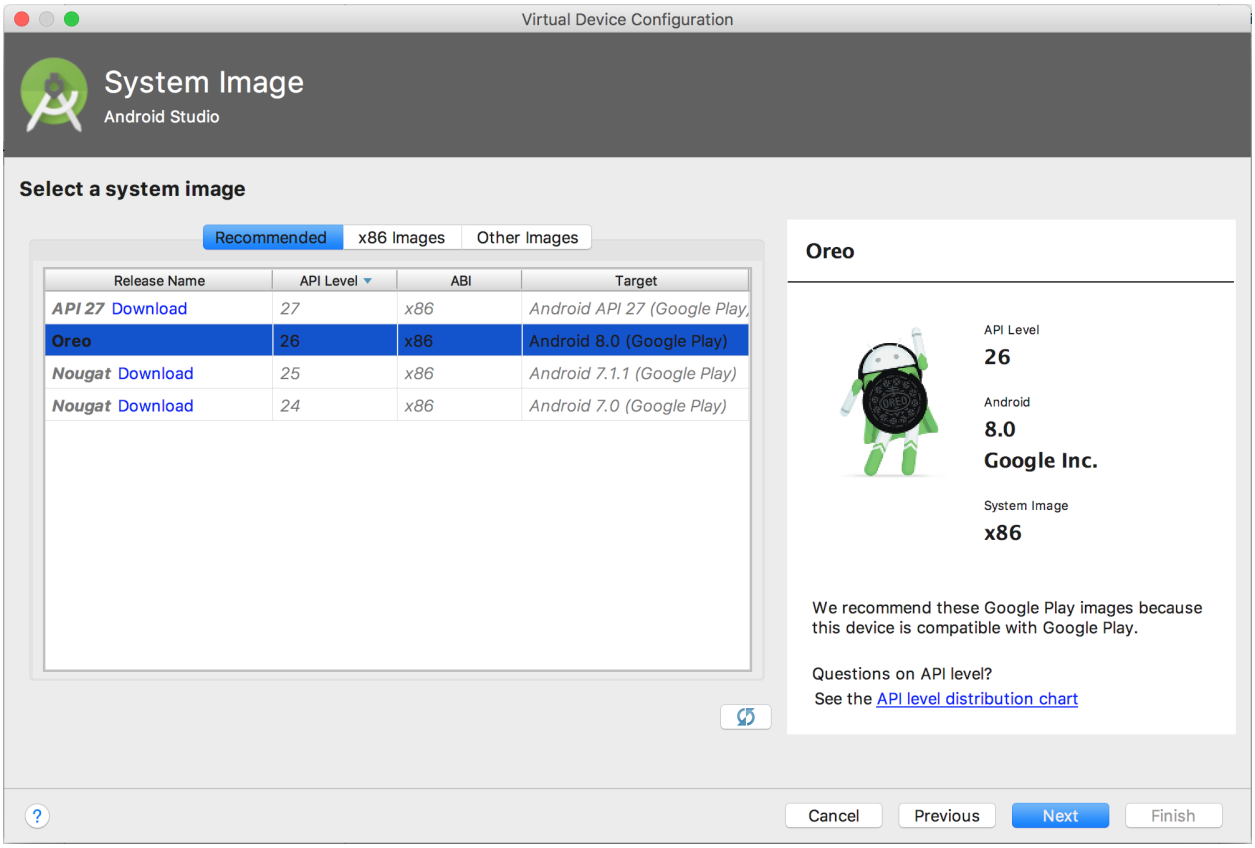

Figura 1.2.7 - Escolhendo a imagem do emulador.

E, por fim, revisamos as configurações do emulador, como apresentado na `Figura 1.2.8`:

- O nome do AVD;
- Dispositivo que será simulado;
- Versão do Android;
- Escala e orientação.

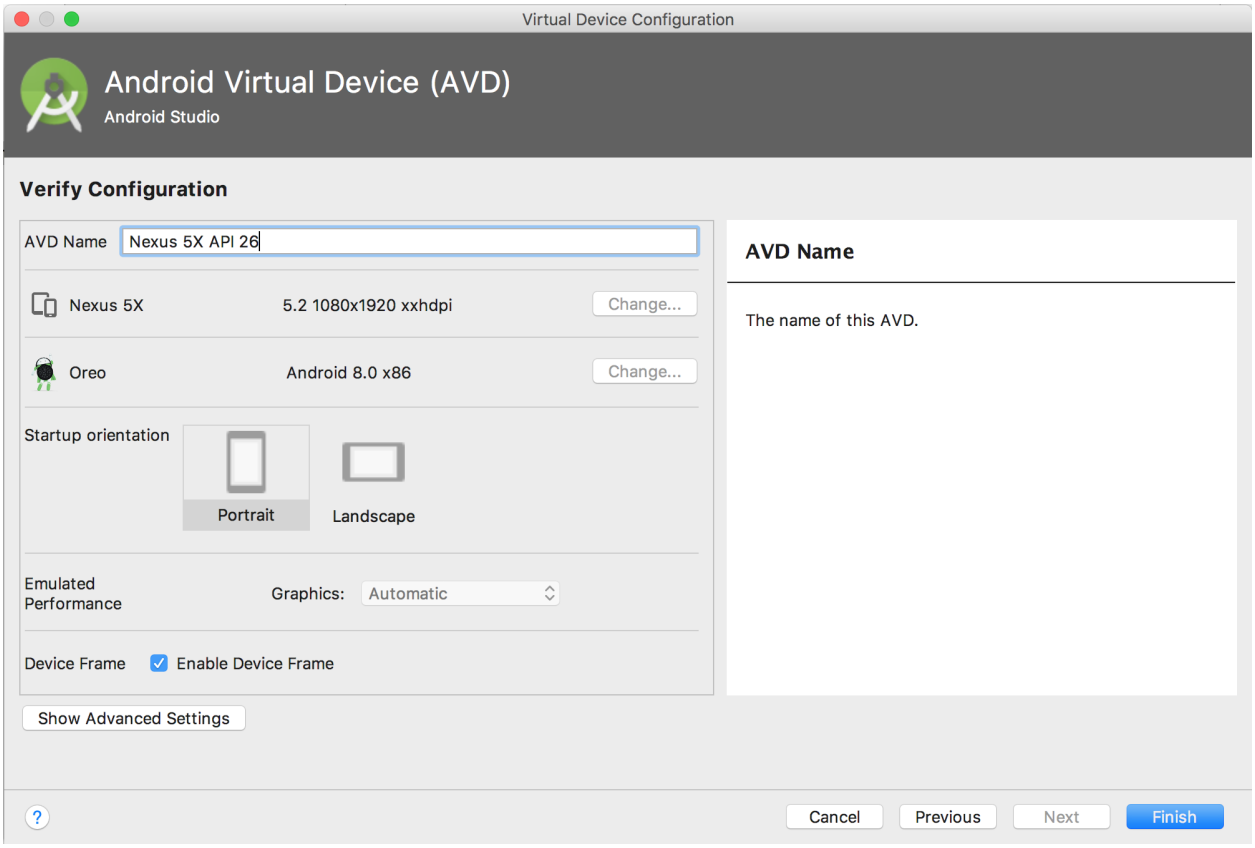

Figura 1.2.8 - Revisando as configurações do dispositivo virtual.

Clique em **Finish (Finalizar)** para terminar a criação do emulador e pode fechar a tela do Android Virtual Device Manager.

Voltando para a tela **Select Deployment Target (Selecione o alvo da publicação)** (`Figura 1.2.9`), escolha o emulador que será iniciado e clique em **OK**.

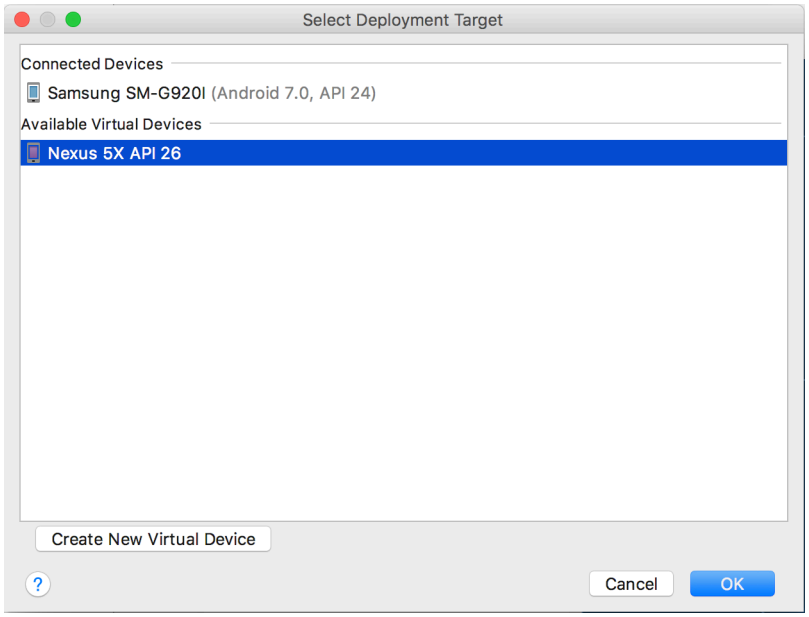

Figura 1.2.9 - Iniciando um emulador.

O emulador será iniciado, desbloqueie a tela e o aplicativo **OlaAndroid** será publicado e automaticamente iniciado, como apresentado na `Figura 1.2.10`.

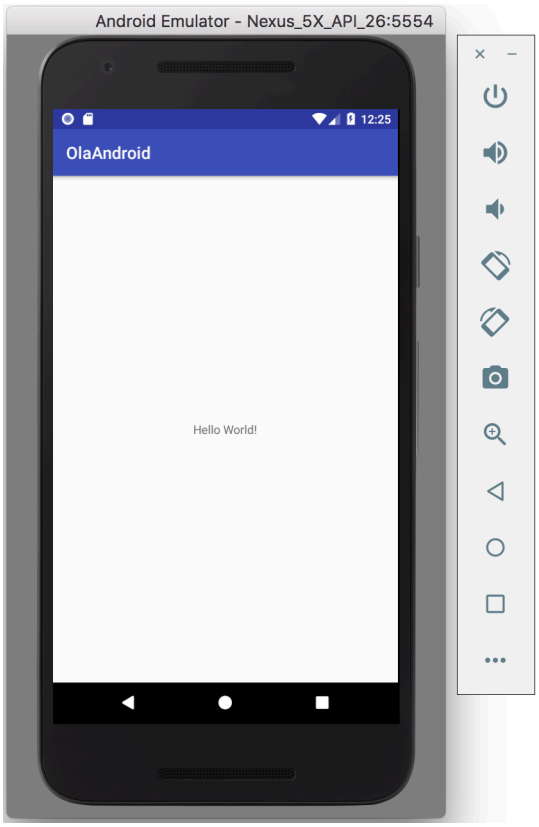

Figura 1.2.10 - Aplicativo iniciado no emulador.

Dica: O emulador demora um pouco para iniciar, então enquanto estiver desenvolvendo mantenha o emulador ligado.

1.3 Personalizando o aplicativo OlaAndroid

Voltando ao Android Studio, vamos alterar o aplicativo para ficar um pouco mais interessante.

Na estrutura do projeto temos alguns pacotes que são de uso frequente como:

- **app → manifests** - O arquivo **AndroidManifest.xml** contém a configuração do projeto;
- **app → java** - Estrutura de pacotes com códigos fonte e testes do aplicativo;
- **app → res** - Pacote de recursos;
- **app → res → drawable** – Pacote com as imagens de acordo com a resolução;
- **app → res → layout** – Arquivos no formato .xml com a estrutura do layout das telas;
- **app → res → values** – Pasta **dimens.xml** com configurações de dimensões e tamanho de fontes que podem ser reutilizados em todo projeto; o arquivo **strings.xml** com os textos strings; e o arquivo **styles.xml** com o estilo de telas do aplicativo.

A `Figura 1.3.1` mostra a estrutura atual do projeto.

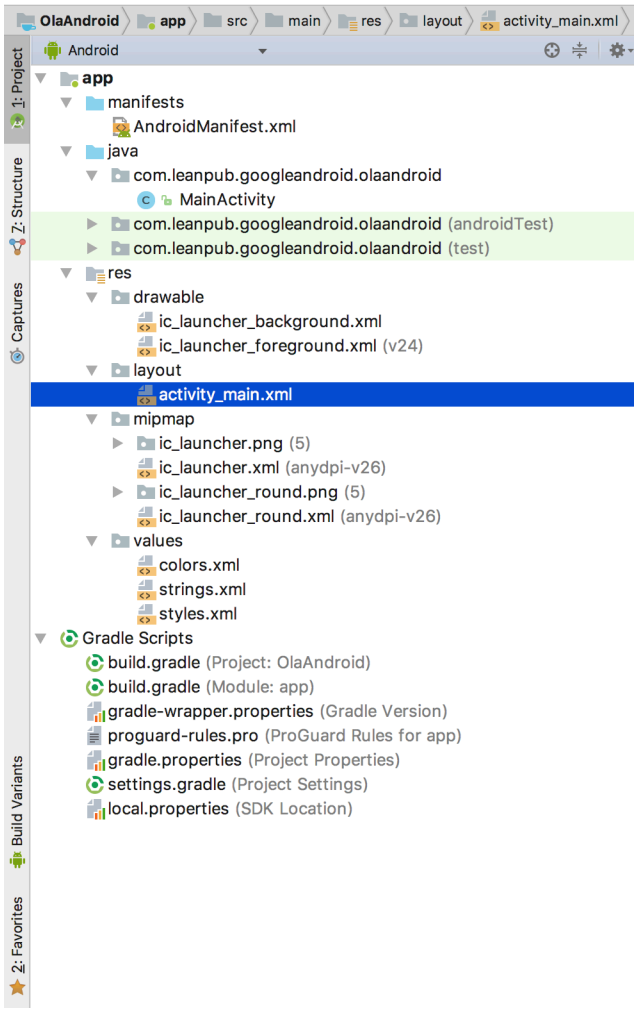

Figura 1.3.1 - Estrutura do projeto.

Para editar a tela do aplicativo, abra o arquivo **app → res → layout → activity_main.xml**.

A edição de uma tela pode ser feita de modo **Design** (`Figura 1.3.2`) adicionando os componentes por meio da paleta e configurando pelas propriedades.

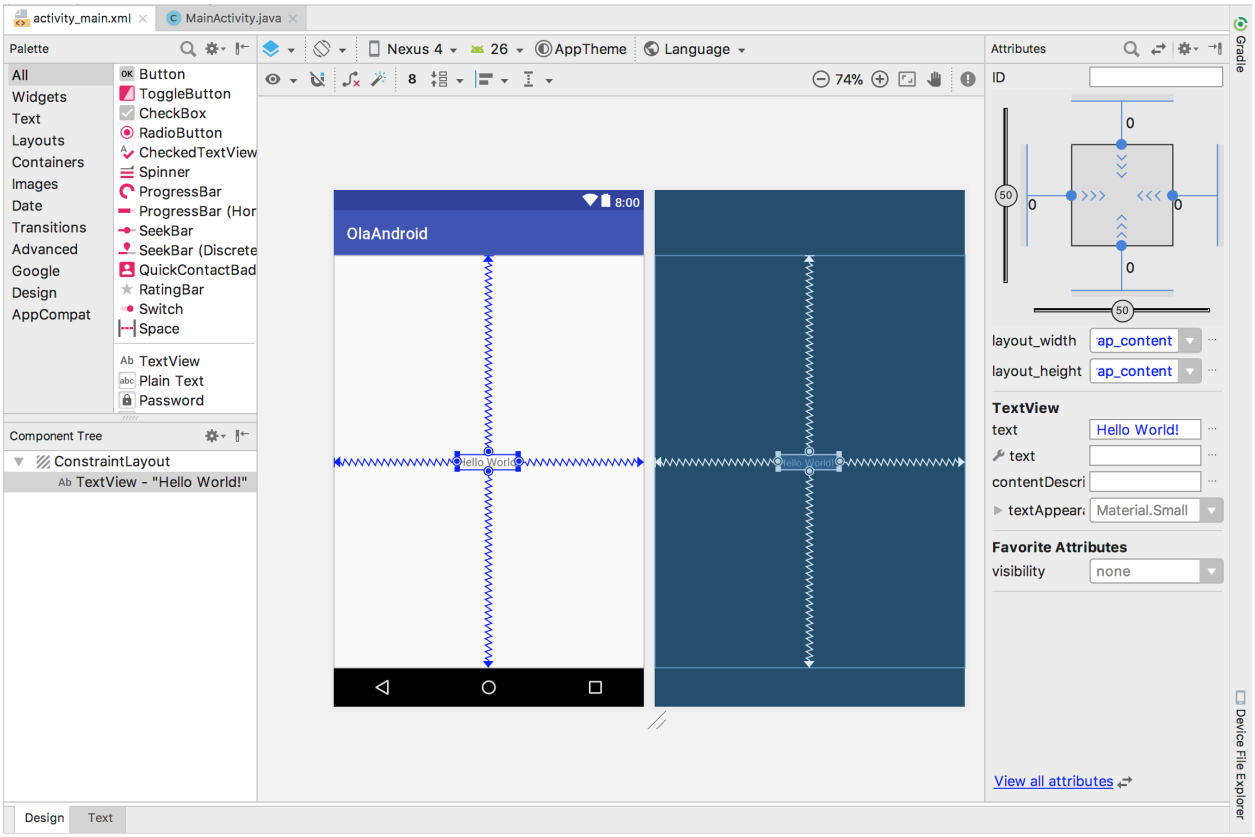

Figura 1.3.2 - Edição da tela no modo Design.

Também podemos alterar o código da tela editando o arquivo activity_main.xml no modo **Text** (`Figura 1.3.3`).

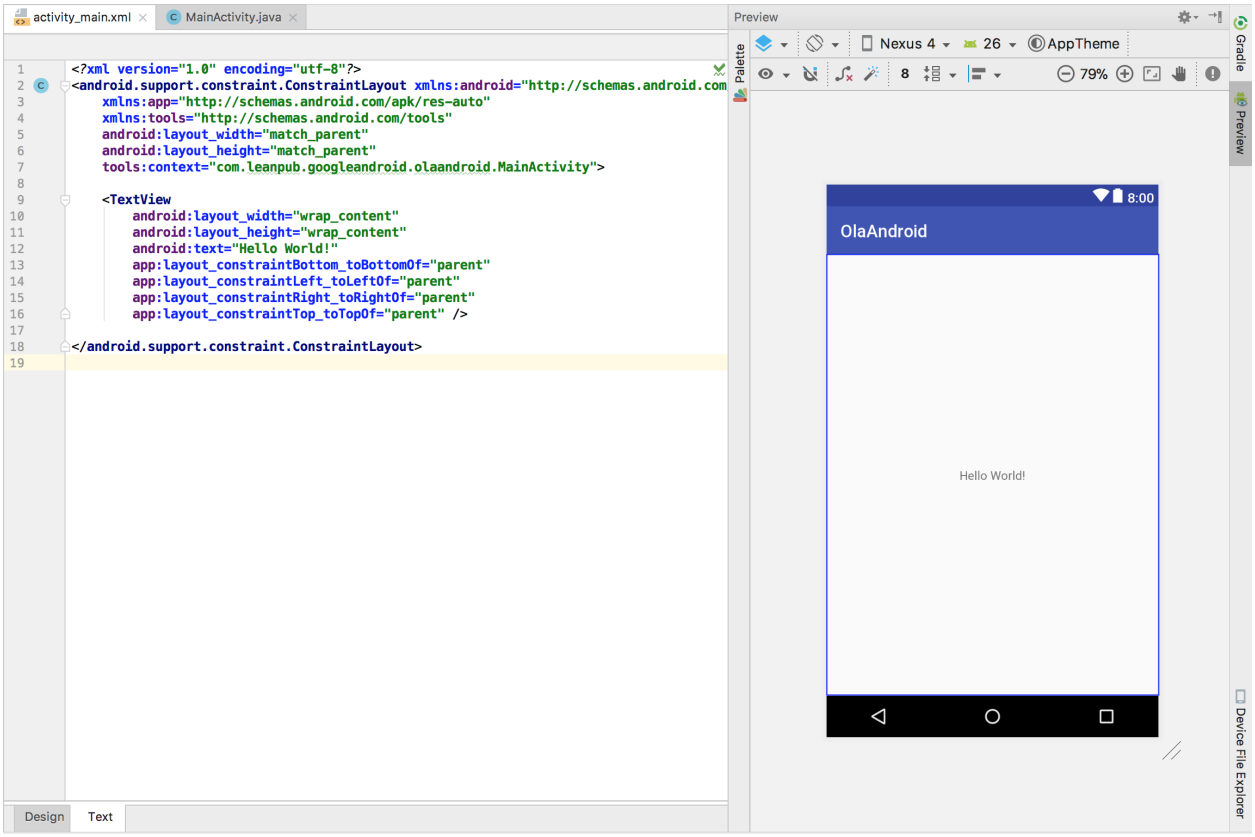

Figura 1.3.3 - Edição da tela no modo Text.

Note que em ambos modos, também temos mais opções para configurar o layout que é renderizado, podemos escolher o tipo de aparelho, se o aparelho está na vertical ou horizontal, o tema do aplicativo, a classe associada com essa tela, a localização para trabalhar com internacionalização de textos e a versão do Android utilizada quando for renderizar a tela do layout, como apresentado na `Figura 1.3.4`.

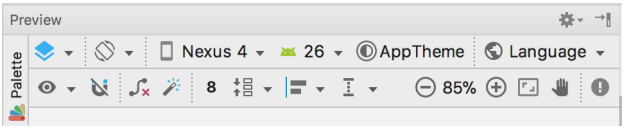

Figura 1.3.4 - Opções de preview do aplicativo.

No **activity_main.xml** altere apenas o código do **TextView** para:

```
<TextView
    android:id="@+id/texto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:gravity="center"
    android:text="@string/ola_android"
    android:textSize="40sp"
    app:layout_constraintTop_toTopOf="parent" />
```

Vai aparecer um erro na linha do **android:text**, mas não se preocupe que já corrigiremos.

O **TextView** representa um texto na tela, como um **label (rótulo)**, no qual pode ser definido o seu texto, como será o seu layout, tamanho do texto, cor do texto, entre outros.

Esse TextView está utilizando uma string (texto) chamada **ola_android**, para criar o conteúdo desse texto, na estrutura do projeto abra o arquivo ** app → res → values → strings.xml** e adicione mais uma linha para a string **ola_android** e conteúdo **Olá Android!!!**.

O arquivo **strings.xml** ficará com o seguinte conteúdo:

```
<resources>
    <string name="app_name">OlaAndroid</string>
    <string name="ola_android">Ola Android!!!</string>
</resources>
```

Agora vamos adicionar uma imagem nessa tela, para isso você pode utilizar uma imagem que exista no seu computador ou usar alguma imagem disponível na Internet, nesse exemplo estou usando um logo do robô Android disponível em: http://developer.android.com/images/brand/Android_Robot_100.png[2], esse logo pode ser usado, reproduzido e modificado gratuitamente.

Salve a figura **Android_Robot_100.png** ou copie e cole dentro da pasta **app → res → drawable**.

Outro detalhe importante é que o nome das imagens pode ter apenas os caracteres `[a-z0-9_]`, portanto após adicionar a imagem **Android_Robot_100.png** aparece a seguinte **mensagem de erro no Console: res/drawable-hdpi/Android_Robot_100.png: Invalid file name: must contain only [a-z0-9_.]**, porque no nome da figura tem caracteres maiúsculos.

Para resolver isso clique em cima da figura **Android_Robot_100.png** e escolha no menu **Refactor (Refatorar)** a opção **Rename (Renomear)** ou clique na figura e pressione SHIFT + F6. Altere o nome do arquivo para **android.png**.

No arquivo **activity_main.xml** após a declaração do TextView, adicione a declaração do **ImageView** que fará uso da imagem do android.png:

[2] http://developer.android.com/images/brand/Android_Robot_100.png

```
<ImageView
    android:id="@+id/imagem"
    android:src="@drawable/android"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:contentDescription="@string/ola_android"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Após adicionar a figura na pasta **drawable**, o Android Studio gera automaticamente um registro para **@drawable/android**.

A ideia é centralizar a figura para isso utilizamos as propriedades `app:layout_constraintBottom_toBottomOf="parent"` e `app:layout_constraintTop_toTopOf="parent"` que alinha a imagem em relação ao componente pai que nesse caso é o layout que controla toda a tela.

O Android também sugere que descrevamos a imagem que está sendo apresentado, isso pode ser feito com a propriedade `android:contentDescription= "@string/ola_android"` que está utilizando a string **ola_android**, esse texto será utilizado apenas para acessibilidade ele não é apresentado na tela.

A estrutura de código e aparência visual deve ser similar a `Figura 1.3.5`.

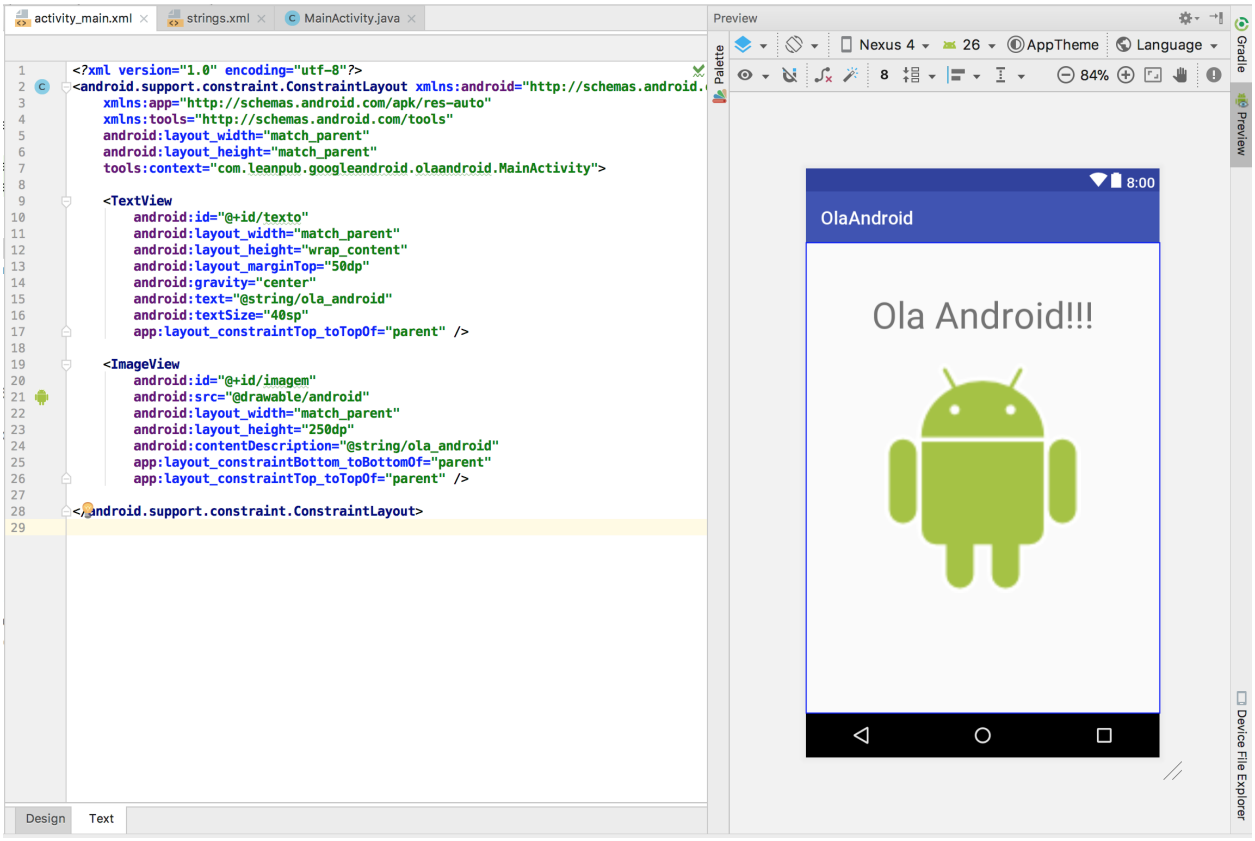

Figura 1.3.5 - Preview do projeto OlaAndroid.

Vamos adicionar uma mensagem `Olá!` quando a imagem do Android for clicada, para isso altere no arquivo **activity_main.xml** a código XML da declaração do ImageView adicionando as últimas duas linhas:

```
<ImageView
    android:id="@+id/imagem"
    android:src="@drawable/android"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:contentDescription="@string/ola_android"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:clickable="true"
    android:onClick="mostrarMensagem"/>
```

Foi adicionado a propriedade `android:clickable="true"` informando que essa imagem pode ser clicada e também a propriedade `android:onClick="mostrarMensagem"` informando que ao clicar na imagem será chamado o método **mostrarMensagem** que será declarado a seguir.

Abra a classe **MainActivity**, essa é a Activity responsável pela tela inicial do aplicativo, nela importe as classes:

```
import android.view.View;
import android.widget.Toast;
```

E adicione o método **mostrarMensagem**:

```
public void mostrarMensagem(View view) {
  Toast toast = Toast.makeText(this, "Olá!", Toast.LENGTH_SHORT);
  toast.show();
}
```

O **Toast** é uma mensagem que aparece e desaparece da tela e o `Toast.LENGTH_SHORT` especifica que é uma mensagem que fica por um tempo curto na tela.

O código completo da classe **MainActivity** ficará assim:

```
package com.leanput.googleandroid.olaandroid;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

    public void mostrarMensagem(View view) {
        Toast toast = Toast.makeText(this, "Olá!", Toast.LENGTH_SHORT);
        toast.show();
    }
}
```

E o código completo do **activity_main.xml** ficará assim:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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="com.leanpub.googleandroid.olaandroid.MainActivity">

    <TextView
        android:id="@+id/texto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:gravity="center"
        android:text="@string/ola_android"
        android:textSize="40sp"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageView
        android:id="@+id/imagem"
        android:src="@drawable/android"
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:contentDescription="@string/ola_android"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:clickable="true"
        android:onClick="mostrarMensagem"/>

</android.support.constraint.ConstraintLayout>
```

Execute novamente o aplicativo no emulador do Android e clique sobre a figura para visualizar a mensagem como mostrado na Figura 1.3.6:

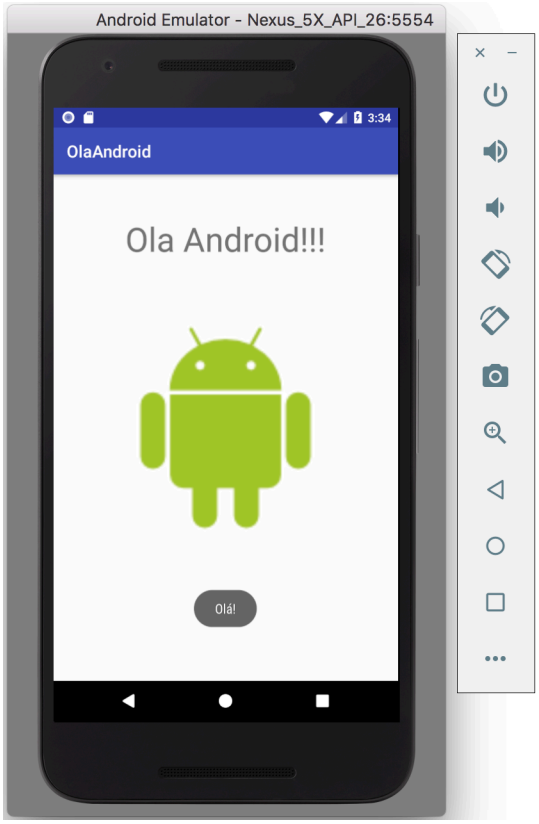

Figura 1.3.6 - Executando o projeto OlaAndroid no emulador.

1.4 Resumo

Neste capítulo foi apresentado como criar um novo projeto Android usando o Android Studio, como criar e instalar um aplicativo no emulador. Vimos alguns componentes básicos como a Activity, e na tela os componentes TextView e ImageView, para terminar mostramos como um método da Activity pode ser chamado ao executar uma ação de click na figura. Esse conceito de chamada de método é o mesmo para botões e outras ações como veremos mais adiante.

1.5 Exercícios

Exercício 1 - Crie um novo emulador para tablet, usando como base o hardware do Nexus 10 ou superior, e execute o aplicativo `OlaAndroid` nele.

Exercício 2 - Altere o exemplo OlaAndroid para colocar em TexView com o seu nome logo embaixo da figura do logo do Android.

Exercício 3 - Para alterar o título do aplicativo para Google Android devemos alterar em qual arquivo?

Exercício 4 - O que é o componente `Toast`? E o que significa definir ele com a propriedade `Toast.LENGTH_LONG`?

Exercício 5 - Para definir tamanho dos componentes usamos algumas propriedades do Android como `match_parent` e `wrap_content`, para que serve cada um deles?