Antigamente, o desenvolvimento para Android no Brasil era difundido utilizando a plataforma **Eclipse** com o Android Development Kit (ADK), fornecido pelo Google, que lançou a plataforma Android Studio. Este é baseado no IntelliJ IDEA, que é uma IDE que também oferece suporte ao Android, mas possui um custo elevado.

Saiba mais: **IntelliJ IDEA**

A IDE Android Studio possui algumas vantagens como, por exemplo, o gerenciador de dependências Gradle (vide seção **Links**), também baseado no IntelliJ, muito utilizado fora do Brasil. Este é um dos grandes trunfos do editor da plataforma, pois oferece mais opções ao desenvolvedor na hora de compilar, já que o Eclipse utilizava o jeito clássico de compilação.

Saiba mais: **Android Studio**

O Android Studio pode ser instalado nos sistemas operacionais Windows, OSX e Linux e é recomendado pelo próprio Google que o hardware possua, no mínimo, 4 GB de memória e 1GB de espaço livre em disco, mas recomendamos que se tenha mais memória, pois foi observado que o Android Studio ainda fica um pouco lento. É necessário ter o Java instalado na máquina através do **JDK (Java Development Kit)** e não a JRE, como normalmente é instalado, pois para desenvolver em Android é necessário que todas as classes de desenvolvimento do **Java** estejam presentes na máquina.

Saiba mais: **Curso de Android SDK**

Instalação do Android Studio

Para instalar o Android Studio bastar entrar em seu site oficial (seção **Links**) e fazer o download da última versão disponível. Escolha a sua plataforma, como é mostrado na **Figura 1**.

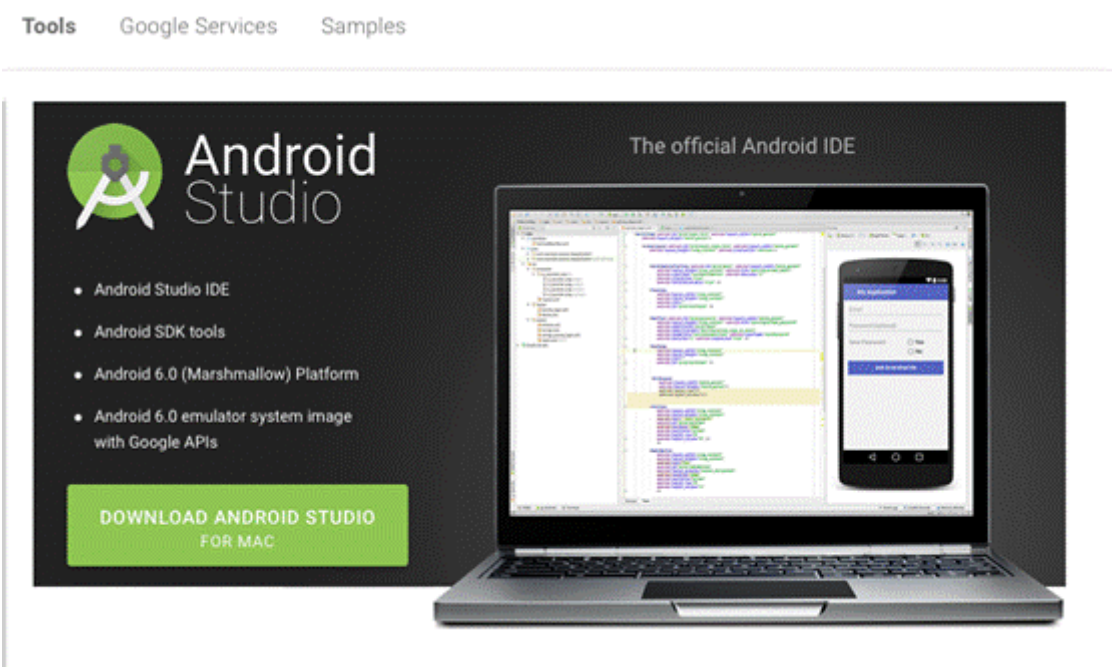

Figura 1. Download do Android Studio

Após a instalação é necessário instalar o SDK do Android, pois sem este não é possível desenvolver para a plataforma. Para isso clique em “Configure”, como é mostrado na **Figura 2**.

Saiba mais: **Curso de Android - Criando uma loja virtual**

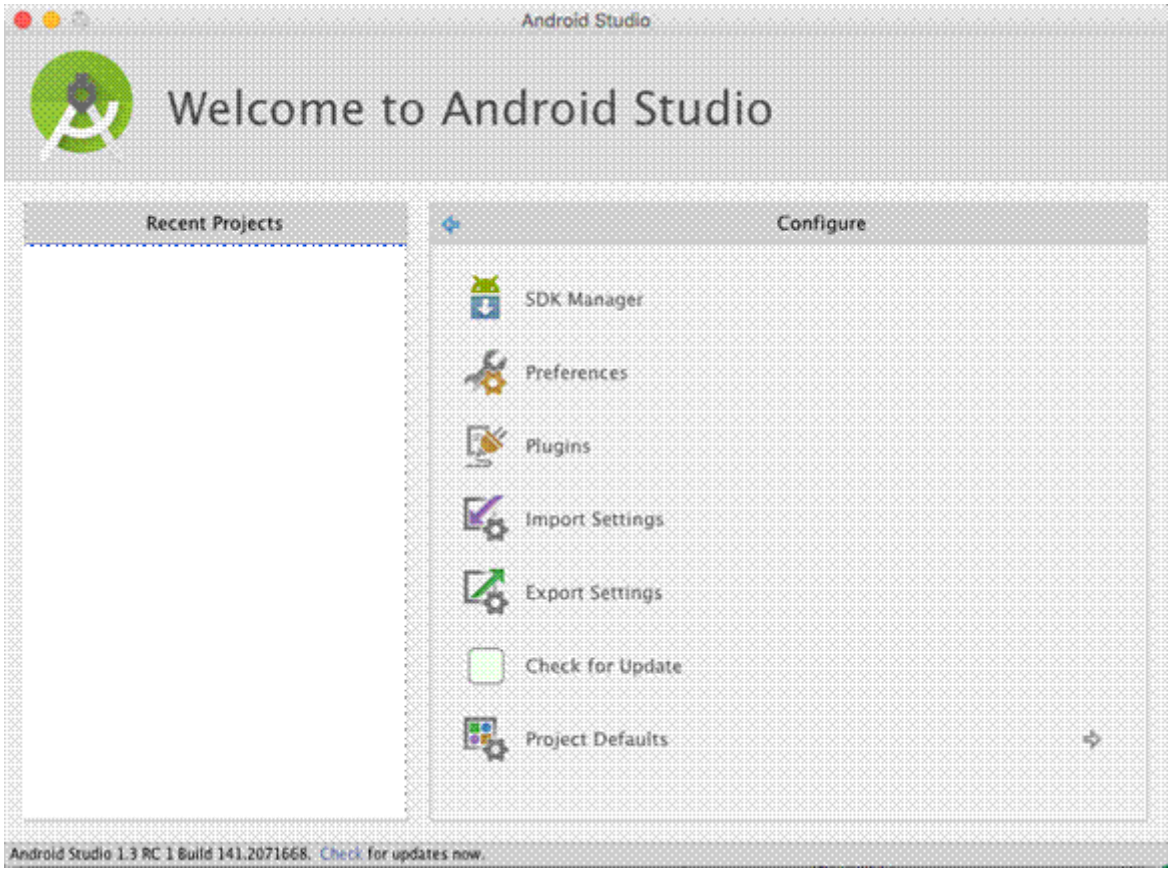

Figura 2. Telas de Boas Vindas

Após entrar na próxima tela, clique em “SDK Manager” e veja que o gerenciador de SDKs do Android será aberto. Marque as opções apresentada na **Figura 3**.

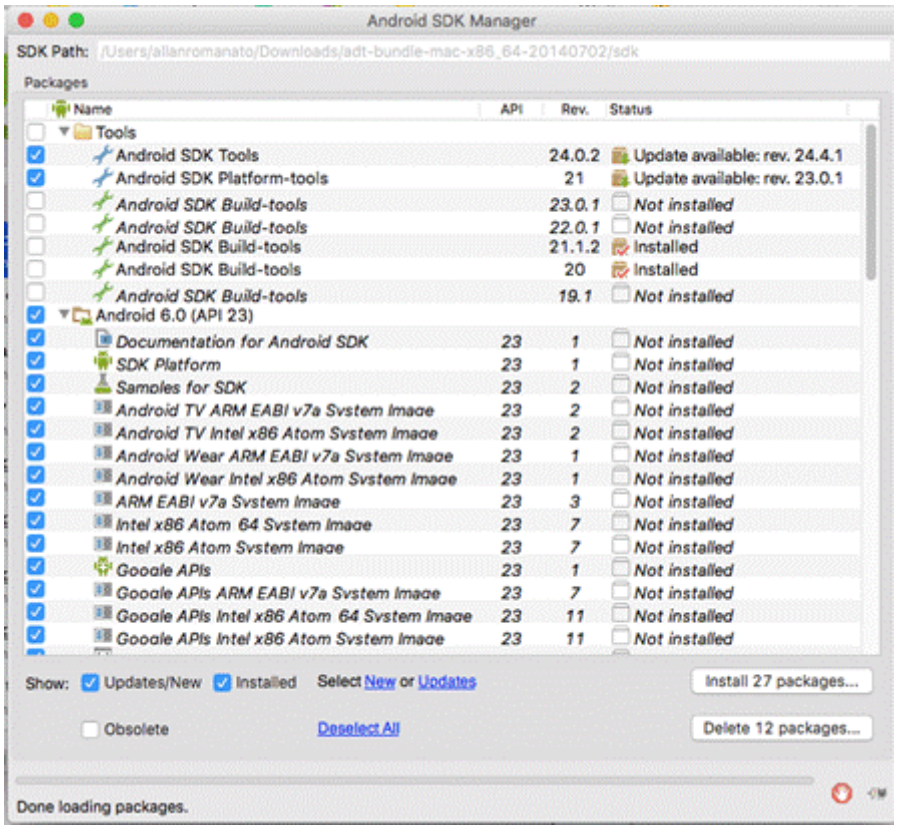

Figura 3. SDK Manager

Lembre-se que para iniciar qualquer desenvolvimento no Android é necessário ter instalado:

- Android SDK Tools, onde estão contidas as ferramentas para o SDK e o emulador;**Android SDK Build-Tools**, onde estão contidas as ferramentas utilizadas para a compilação de seu código. Dependendo da versão que tiver feito o download, ele será utilizado para compilar o seu código.
- Android SDK Platform-tools, onde estão contidas todas as ferramentas do Android.

Os outros itens podem ser instalados para se ter mais componentes, como uma documentação, emuladores de Android TV, Android Wear entre outros elementos que podem ser utilizados pelo desenvolvedor durante o projeto. Com tudo instalado pode-se dar início a criação do primeiro projeto utilizando o Android Studio.

Primeira Aplicação com o Android Studio

Para iniciar um novo projeto deve-se clicar em “Start a new Android Studio Project”. Observe na **Figura 4** que colocamos o nome da aplicação, o domínio e onde salvar para criar o pacote principal.

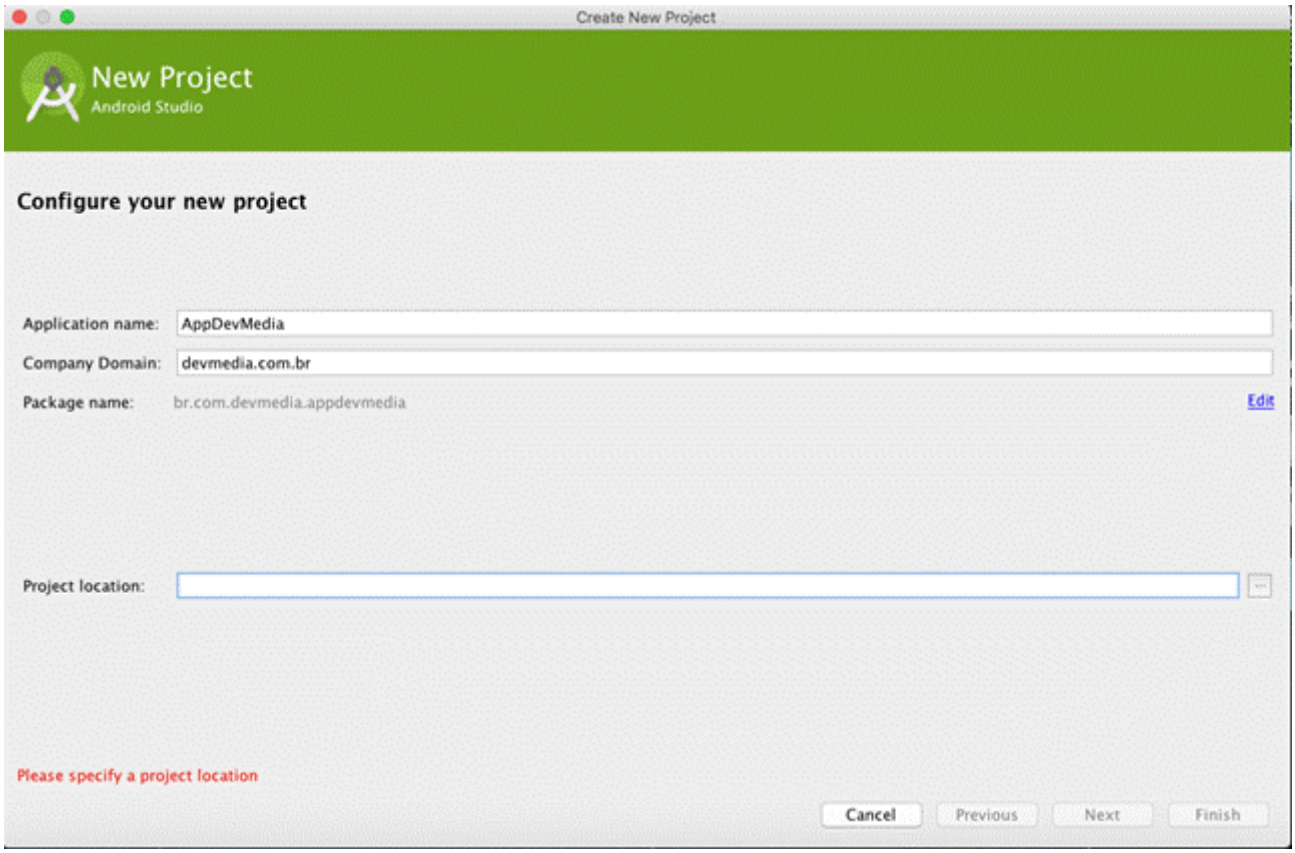

Figura 4. Configuração do Projeto I

Após concluído, clique no botão “Next” para ir para a próxima tela (**Figura 5**) onde faremos a configuração dos SDKs que precisaremos para simular, ou seja, em qual dispositivo (Smartphones/Tablets, Android TV, Android Wear ou Android Glass) executaremos a aplicação.

Para o nosso exemplo selecionaremos apenas smartphones e Tablets.

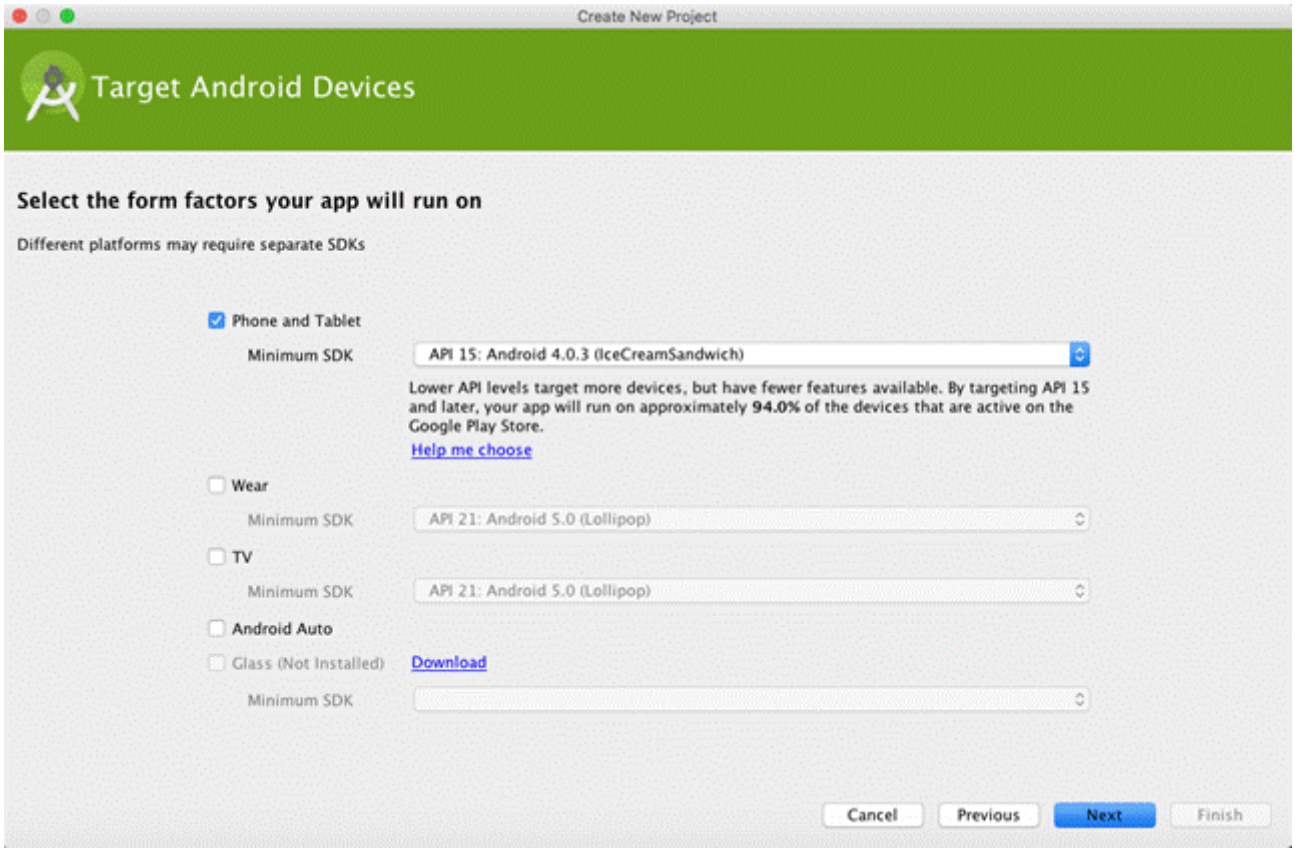

Figura 5. SDK Mínimo

Algo bem interessante que se nota nessa tela é que ao selecionar um SDK é mostrada a quantidade aproximada de dispositivos que poderão utilizar a aplicação a ser criada, de acordo com dispositivos ativos na Google Play Store.

Já na próxima tela selecionamos um dos templates de Activity que estão disponíveis. O template selecionado para o nosso primeiro exemplo será o Blank Activity, como mostra a **Figura 6**.

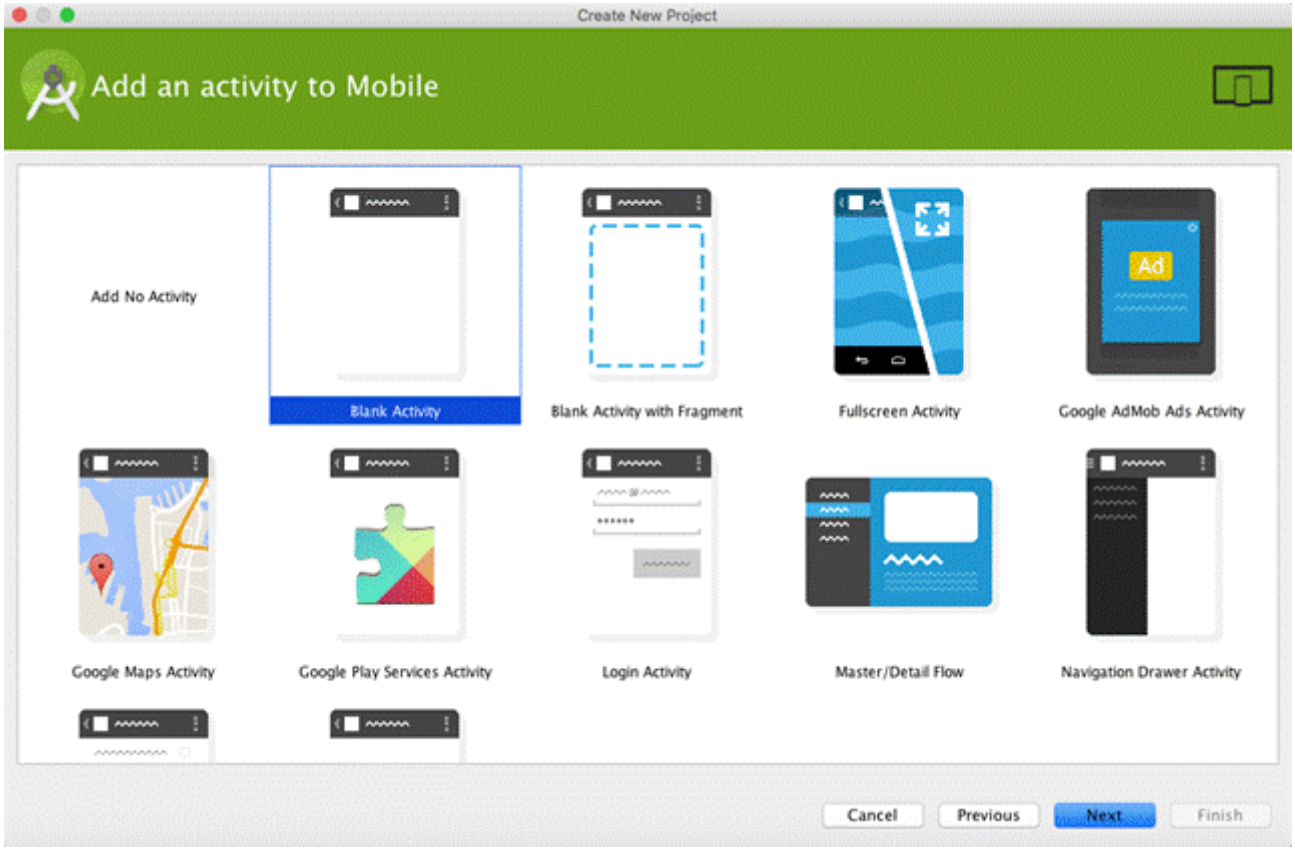

Figura 6. Seleção de Template

Saiba mais: **Linear, Table e Relative Layouts com Android Studio**

A próxima tela, vista na **Figura 7**, é onde será definido o nome da Activity e de seu template. Vale lembrar que o template é um arquivo XML onde os elementos de tela são definidos e a Activity é o código Java que acessa esses elementos do Template.

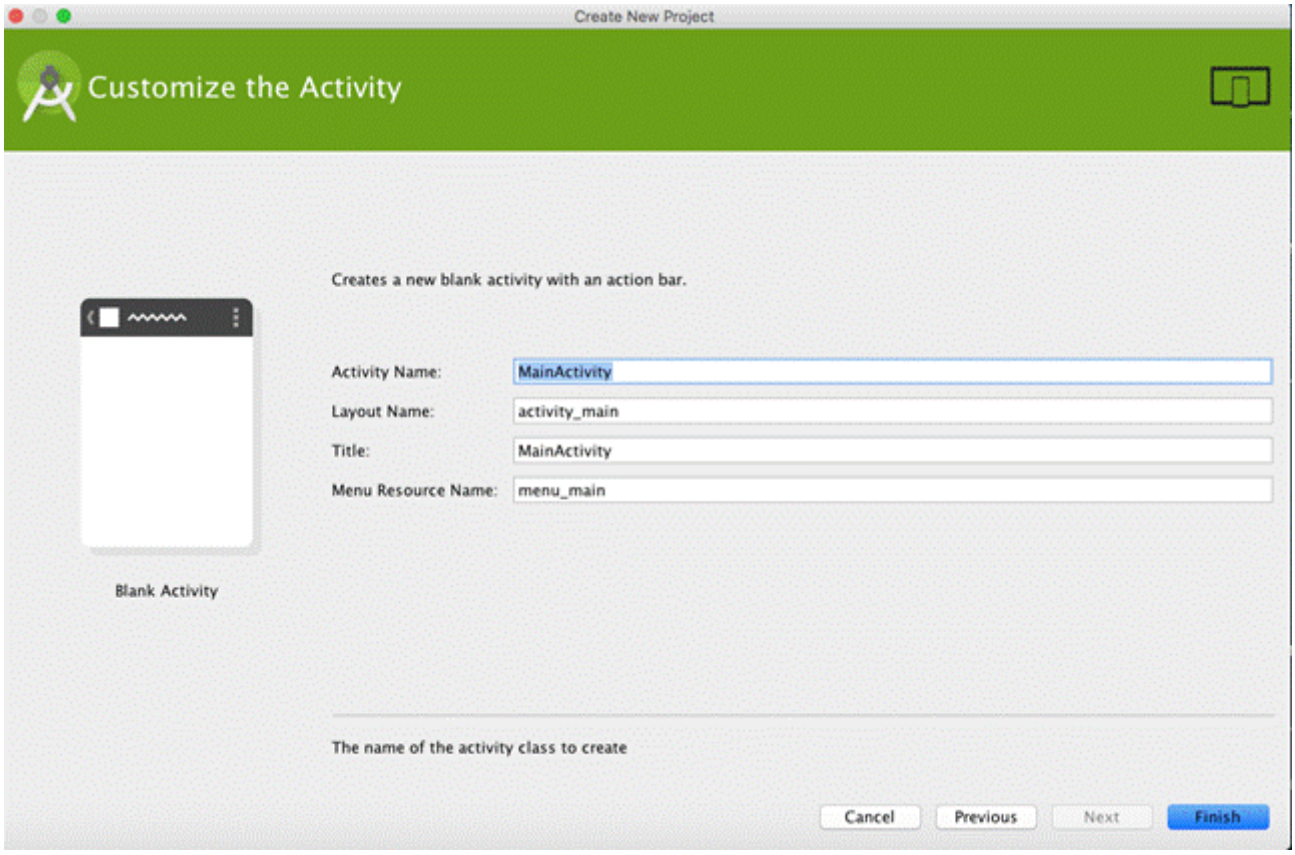

Figura 7. Definindo os Nomes

Após clicar em "Finish" o projeto é criado e a **Figura 8** mostra a tela principal de desenvolvimento do Android Studio com o template XML e os elementos de tela renderizados. Ao alterar o código essa tela é alterada automaticamente e, se algum elemento é alterado, este é renderizado no código sem maiores problemas.

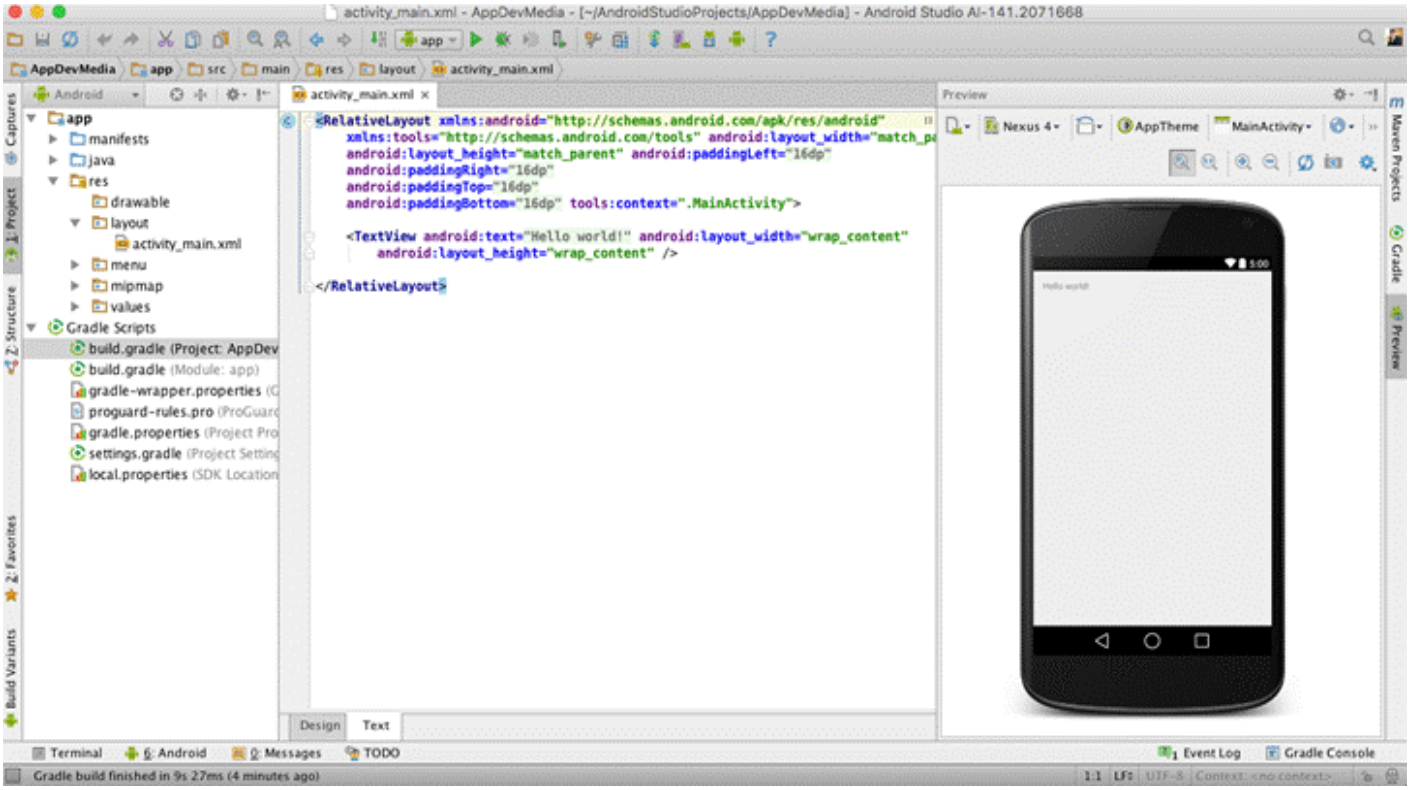

Figura 8. Tela Android Studio

O código XML gerado automaticamente quando se cria um projeto utilizando o template Blank pode ser visto na **Listagem 1**.

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"
  tools:context=".MainActivity">

  <TextView android:text="@string/hello_world" android:layout_width="wrap
    android:layout_height="wrap_content" />

</RelativeLayout>
```

Listagem 1. Código XML padrão

Também é gerado um código Java padrão que faz a referência para o XML, podendo exibir na tela do dispositivo, lembrando que a Activity sempre irá controlar a tela de um aplicativo Android.

A **Listagem 2** mostra o código do Java gerado por padrão quando se cria qualquer projeto utilizando template Blank pelo Android Studio.

```
package br.com.devmedia.appdevmedia;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is p
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
```

```
    }
}
```

Listagem 2. Código Java padrão

Também é criado o arquivo Manifest.xml , que é um arquivo mandatório para qualquer aplicação Android, é nele que existem informações sobre Activities, permissões e outras informações relevantes para a aplicação que esta sendo escrita. A **Listagem 3** mostra o arquivo de manifesto gerado por padrão.

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

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

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

</manifest>
```

Listagem 3. Manifest.xml gerado por padrão

Rodando a aplicação no Android Studio

Para rodar uma aplicação é necessário criar um dispositivo virtual, o qual irá emular o sistema operacional do Android, onde a aplicação desenvolvida será testada e validada antes de outros tipos de testes, como em dispositivos reais. Para criar um dispositivo virtual, clique no botão destacado na **Figura 9**.

Figura 9. Botão para criar Emulador

A tela que será exibida após o clique desse botão, mostra a listagem de todas os Emuladores já criados, na primeira execução a listagem estará vazia pois o primeiro emulador deverá ser criado agora. Ao clicar no botão "Create Virtual Device" como mostrado na **Figura 10**, irá direcionar para a tela de criação de emuladores.

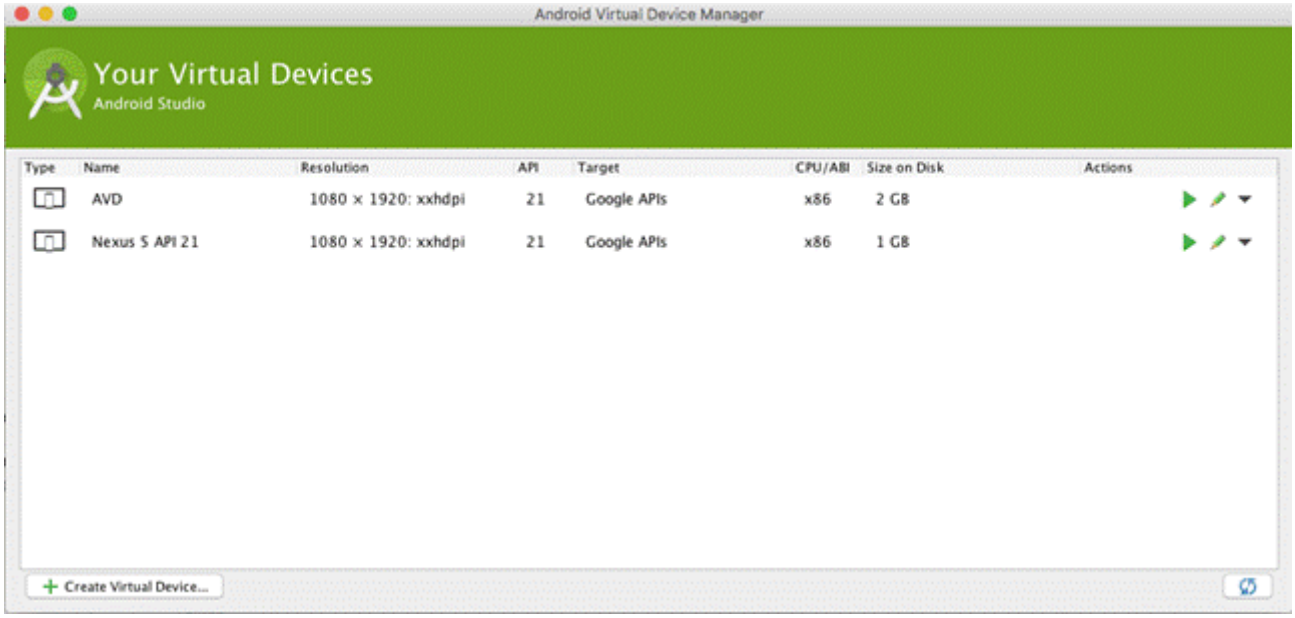

Figura 10. Listagem de Emuladores

Para criar um emulador, a primeira copa que deve se feita é escolher que tipo de dispositivo ira ser emulado, nesse caso, sera um telefone do tipo Nexus 5. A **Figura 11** mostra como é a layout da tela.

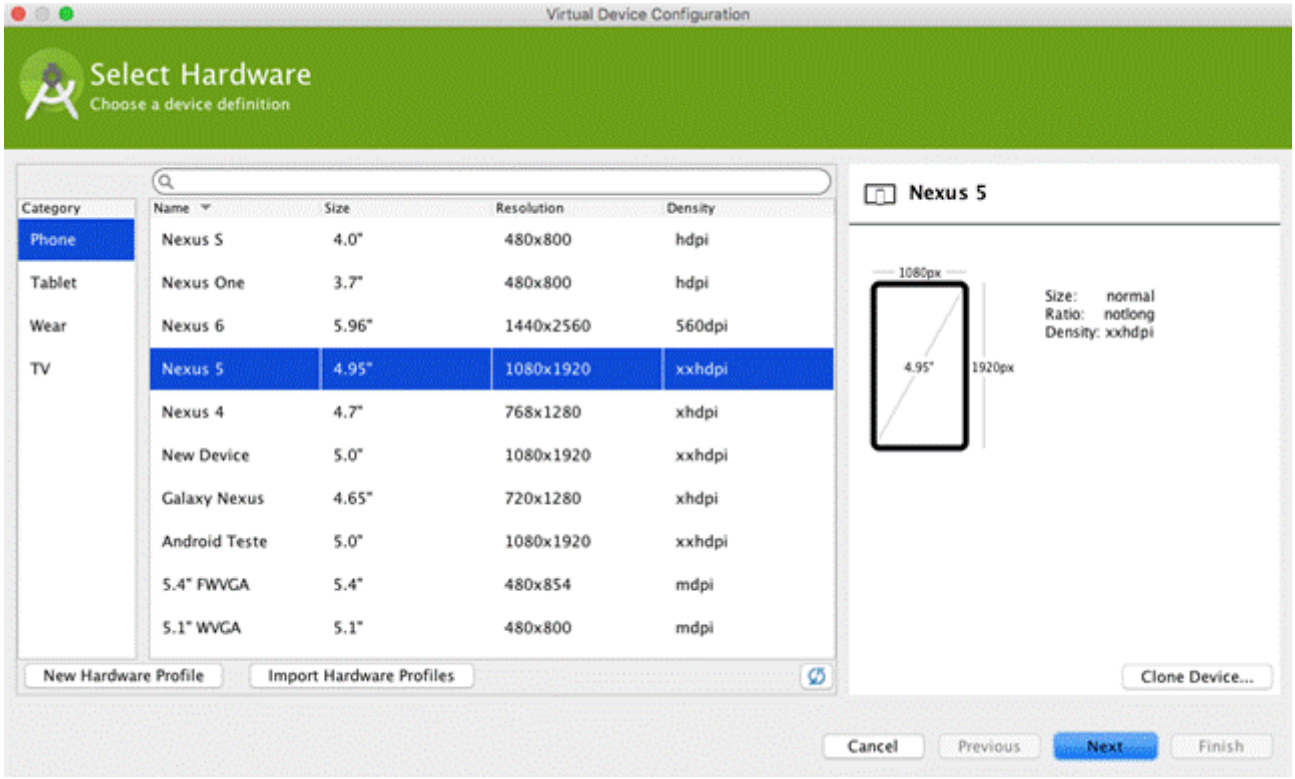

Figura 11. Seleção de Hardware

Após selecionar o Hardware, será necessário criar a imagem a ser utilizada, ou seja, qual a API, que o desenvolvedor deseja criar a aplicação como mostrado na **Figura 12**.

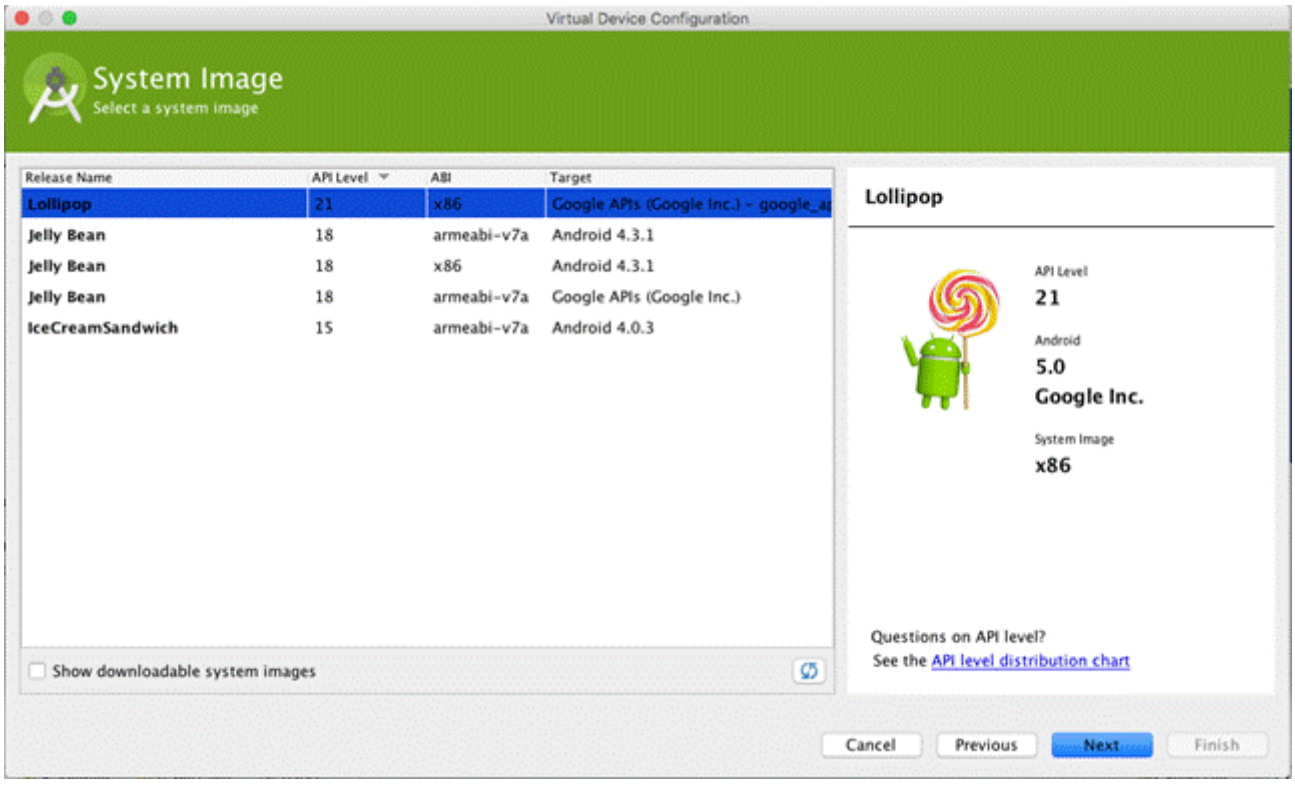

Figura 12. Seleção de imagemp

Com a imagem propriamente selecionada, o Android Virtual Device será criado, aqui deve-se colocar informações como nome, orientação da tela, se a câmera será habilitada ou não, ou se irá emular cartão SD e o tipo de performance será utilizada. A **Figura 13** mostra a tela descrita.

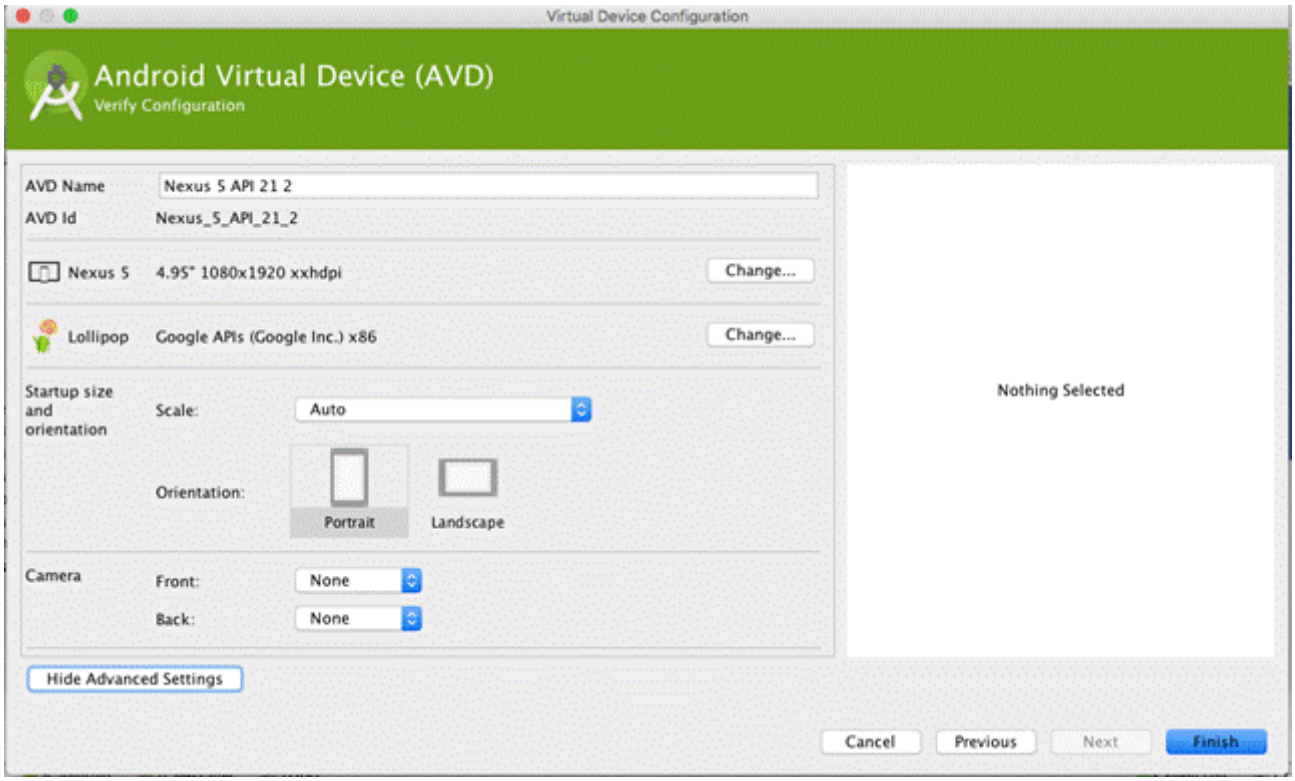

Figura 13. Criação da AVD

Testando a aplicação

Com isso tudo feito, o emulador pode ser iniciado para teste, é importante que se navegue pelo sistema emulado para que se tenha certeza que tudo esta funcionando corretamente. A **Figura 14** mostra o sistema emulado.

Figura 14. Sistema emulado

Agora a pergunta que a maioria dos desenvolvedores que estão iniciando no Android perguntam: Como que meu aplicativo será instalado no emulador? A resposta é bem simples, o Android Studio se encarrega de executar comandos necessários para instalar o arquivo APK dentro do sistema emulado pelo Virtual Device. Aprenda a desenvolver aplicativos com **React Native: do Hello World ao CRUD**.

Para que isso seja concluído basta clicar no botão destacado na **Figura 15**. Primeiramente o Android Studio irá verificar o código para ver se existe algum erro de sintaxe que impeça a compilação, caso nenhum e erro seja encontrado, o sistema irá gerar o APK, e instala-lo dentro do sistema previamente emulado.

Figura 15. Botão para rodar aplicação

Agora como demonstração do Android Studio, um pequeno aplicativo será desenvolvido para fins didáticos e consolidar os conhecimentos abordados acima. O que será desenvolvido não tem segredo nenhum, conterá uma tela, nessa tela, essa tela possuirá um Button e um TextView, ao clicar nesse botão, uma mensagem será exibida no TextView, portanto deve-se alterar o arquivo XML activity_mail.xml, é algo bem simples, mas ao clicar no botão mostrado na **Figura 15**, todo o processo de instalação da aplicação dentro do dispositivo virtual será iniciado.
A **Listagem 4** mostra o código XML do template que foi modificado para atender os critérios descritos anteriormente.

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">


  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Me Clique"
    android:id="@+id/button"
    android:layout_centerVertical="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Large Text"
    android:id="@+id/textView2"
    android:layout_alignParentTop="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

</RelativeLayout>
```

Listagem 4. Código XML modificado

Com o arquivo XML alterado a compilação ja pode ser feita, e o layout modificado será exibido na tela, porem ele não terá funcionalidade alguma, pois que foi modificado, foi apenas a estrutura do template, o código que irá controlar e adicionar as funcionalidades não esta contido no XML e sim na classe Java que extende a uma super classe do tipo Activity, vale lembrar que

toda Activity no Android é responsável por controlar uma interface gráfica, ou seja, será atrelado a um arquivo XML de layout. O código java responsável por fazer essa junção e dar a funcionalidade propriamente dita para o XML é bem simples, e pode ser visto na **Listagem 5**.

```
package br.com.devmedia.appdevmedia;

import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends ActionBarActivity {

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

        Button botao = (Button)findViewById(R.id.button);
        final TextView texto = (TextView)findViewById(R.id.textView2);

        botao.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                texto.setText("O botão foi clicado!!!!");
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is p
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
```

```
            // Handle action bar item clicks here. The action bar will
            // automatically handle clicks on the Home/Up button, so long
            // as you specify a parent activity in AndroidManifest.xml.
            int id = item.getItemId();

            //noinspection SimplifiableIfStatement
            if (id == R.id.action_settings) {
                return true;
            }

            return super.onOptionsItemSelected(item);
        }
    }
```

Listagem 5. Código Java modificado

Observem o código contido na **Listagem 6**.

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  Button botao = (Button)findViewById(R.id.button);
  final TextView texto = (TextView)findViewById(R.id.textView2);

  botao.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
  texto.setText("O botão foi clicado!!!!");
    }
  });
}
```

Listagem 6. Fragmento de código Java

Notem que um objeto do tipo Button é declarado mas não é iniciado utilizando o new e sim utilizando o método findViewById , esse método é o responsável por ligar o elemento de tela que

esta no XML com o objeto declarado no código Java, vale lembrar também que o findViewById devolve um objeto do tipo View portanto é necessário que se faça o cast para o tipo de dado que foi declarado no inicio. O mesmo ocorre para o TextView, o findViewById é chamado, o cast é feito e o elemento XML é colocado nos parâmetros do método, nesse caso o TextView é declarado como final por ser utilizado dentro de um método interno, que nada mais é que o método `onClick`, um listner para que os eventos do botão sejam capturados e exibidos na tela.

A **Figura 16** mostra o resultado do código descrito anterior.

Figura 16. Resultado Final

Nesse artigo pode-se aprender como criar e executar uma aplicação simples em Android utilizado a poderosa plataforma Android Studio, apenas funcionalidades básicas foram abordadas, mas com isso já é possível o desenvolvedor criar seus códigos e testa-los no ambiente virtual emulando o sistema Android.