



ANDROID APRENDIZ

Um guia para iniciantes

Crie seu primeiro aplicativo Android

Fillipe Cordeiro

Android Aprendiz

Um guia para iniciantes
Crie seu primeiro aplicativo Android

Sumário

#1 Passo: Introdução a Plataforma Android	9
#2 Passo: Estrutura da Plataforma Android	11
#3 Passo: Meu Primeiro Projeto Android	13
#4 Passo: Entendendo o Projeto Android	22
#5 Passo: Interface do Projeto Android	25
#6 Passo: Programação do Projeto Android	32

Sobre o Autor

Fillipe Cordeiro é engenheiro e possui mais de 8 anos de experiência em desenvolvimento de softwares. Dentre suas experiências com desenvolvimento de aplicativos, estão tecnologias como Java, Android e Python. Formado em Engenharia da Computação, dedica-se a levar conhecimento de alto nível a profissionais e aprendizes que, como ele, acreditam que o Mobile e a Internet das Coisas não estão no futuro e sim no presente.

Sobre o Livro

O objetivo deste livro é apresentar você ao mundo do Android de uma forma fácil e rápida, sem muita enrolação.

Hoje em dia, cada vez mais pessoas utilizam o celular para acessar a Internet, o volume é tanto que o acesso à web, por dispositivos móveis, já está quase superando o acesso dos computadores tradicionais. Por essa razão, muitas empresas, na área mobile, faturam milhões anualmente desenvolvendo aplicativos próprios e para terceiros.

Com a altíssima demanda por profissionais qualificados e a grande falta destes, começam a aparecer grandes oportunidades para quem quer seguir uma carreira como desenvolvedor Android, seja em grandes empresas ou autônomos.

O livro Android Aprendiz mostra exatamente o marco inicial do caminho a ser percorrido, para que você adentre o mundo da plataforma e, num breve futuro, alcance esse grau de profissionalismo. É um conteúdo de qualidade, focado na criação de sua primeira aplicação Android e está disponível gratuitamente.

Dúvidas

É muito comum aparecerem dúvidas durante o aprendizado de uma nova plataforma. Neste momento, não será diferente.

Quando comecei a estudar sobre o Android, tive muitas duvidas sobre muitas coisas e conceitos. Às vezes, era difícil achar respostas claras para o que eu buscava.

Por isso, para que você possa se desenvolver o mais rápido possível, estou disponibilizando meu e-mail para entrar em contato, quando suas dúvidas surgirem. Eu responderei a todos os e-mails; então, não deixe de me contatar, caso esteja com alguma dificuldade.

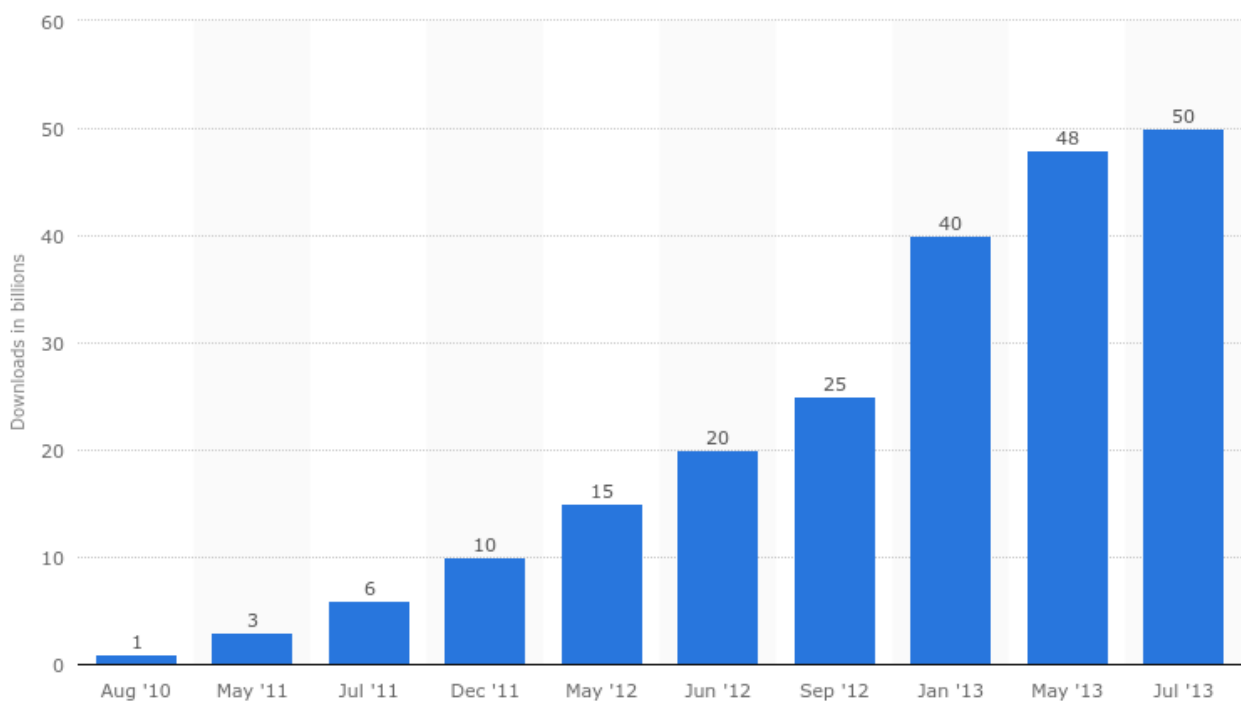
fillipe@androidpro.com.br



Introdução a Plataforma Android

É muito comum surgirem muitas dúvidas sobre uma plataforma muito antes de você começar a estudá-la; por isso este material tem como missão passar o máximo de informações possíveis para dar todo o conhecimento que você precisa saber sobre a plataforma do Google.

Hoje, o Android é a plataforma mobile mais utilizada. Vejamos alguns dados estatísticos para entender o alcance que essa plataforma tem pelo mundo.



Esse quadro apresenta os dados do site Statista, até julho de 2013, onde podemos ver que foram feitos mais de **50 bilhões** de downloads de aplicativos pelo mundo todo.

Outro dado muito interessante é a quantidade de aparelhos em que o Android está presente, **85%** dos aparelhos em mais de **190** países. E esses números não param de crescer, pois segundo o Google, a cada dia, mais de um milhão de usuários novos chegam ao Android para consumir jogos e aplicativos.



Claro que, com todas essas pessoas utilizando a plataforma, gera muita demanda para a criação de aplicativos e jogos de diversas áreas. Muitas empresas estão incorporando, em seu Negócio, aplicativos mobile e cresce o número de empresas especializadas em desenvolvimento de aplicativos para terceiros.

Toda essa demanda de desenvolvimento de aplicativos acaba influenciando o mercado de trabalho de desenvolvimento de aplicativos e jogos. Hoje, o mercado mobile está **muito aquecido e muito exigente** para com o desenvolvedor mobile.

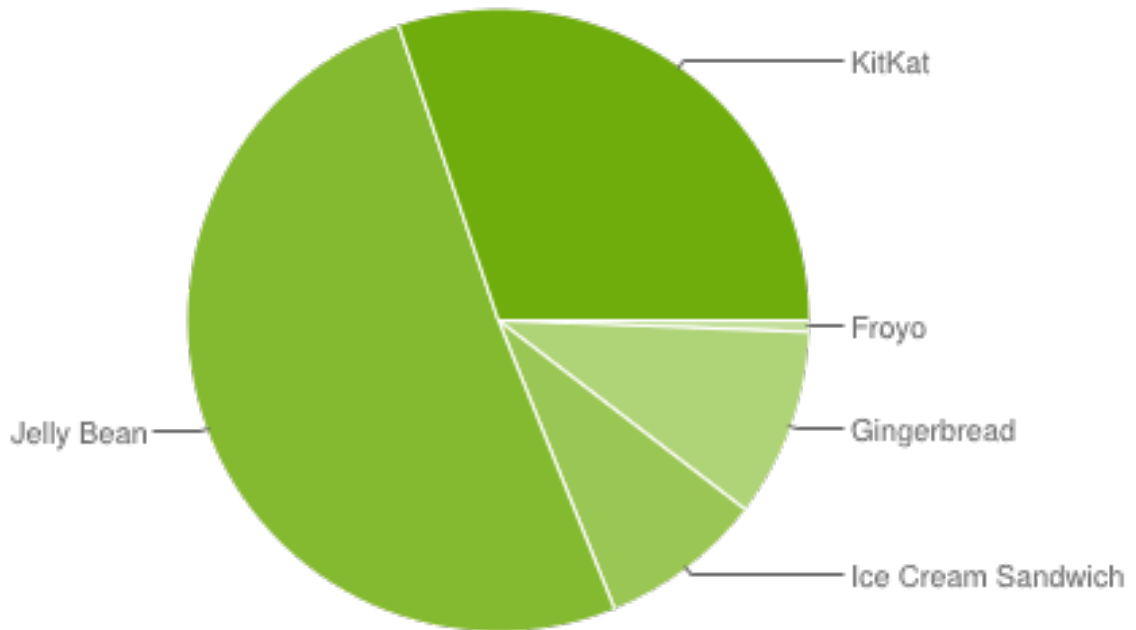
Muitos desenvolvedores usam seu tempo livre para desenvolver aplicativos próprios e publicar na Google Play, para lucrar com a venda ou publicidade. Existem também um mercado muito grande de trabalhos de *freelancer*, principalmente, vindos de agências que têm uma demanda muito grande de projetos.

Um bom desenvolvedor Android tem de saber dar **qualidade** ao aplicativo que está desenvolvendo e isso inclui boa qualidade de código, alta produtividade no desenvolvimento e conhecimento em integrações de sistemas.

Existem diversas **ferramentas** e **frameworks** que ajudam o desenvolvedor com ganho de **produtividade** e a **minimizar bugs** em seus aplicativos, melhorando, assim, a qualidade do software.

Estrutura da Plataforma Android

Desde 2007, o Android vem sofrendo atualizações e a cada versão que é lançada a plataforma fica mais poderosa em termos de processamento e integração com outros aparelhos eletrônicos. Veja, na tabela abaixo, a lista das versões da plataforma.



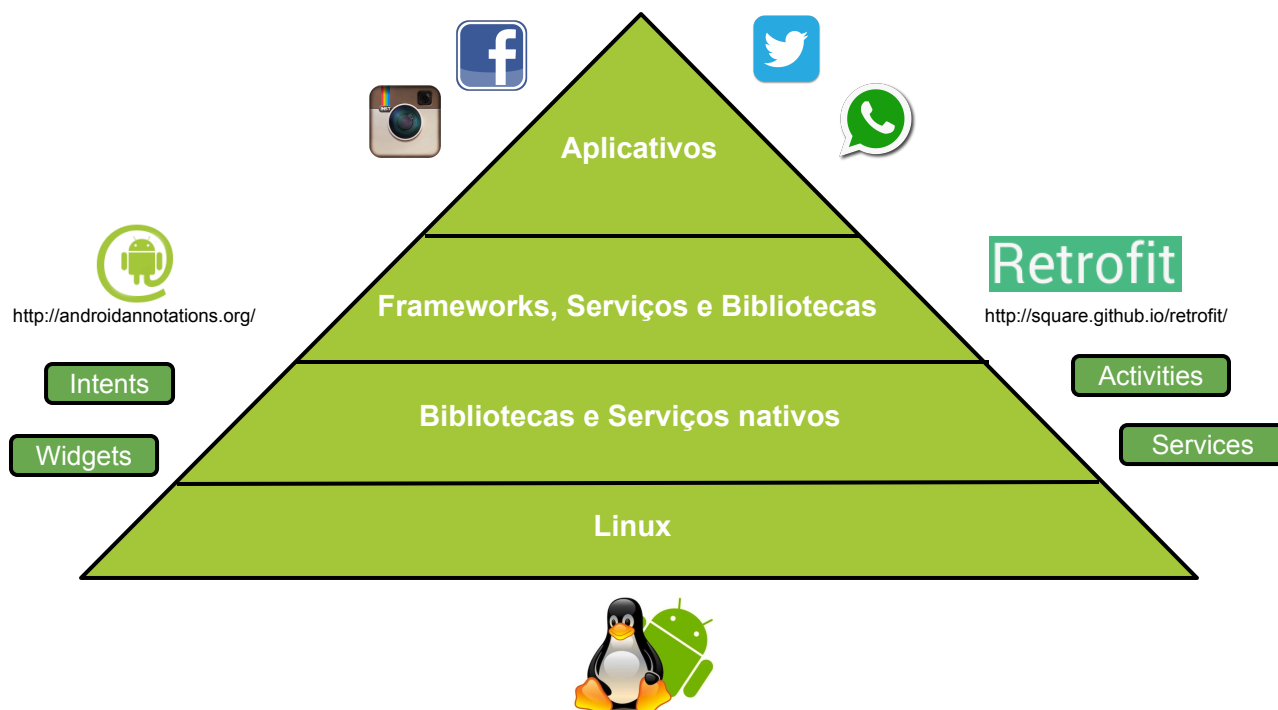
(Fonte: <https://developer.android.com/about/dashboards/index.html>)

Podemos ver que a versão Jelly Bean, hoje, é a mais utilizada pelos usuários.

Recentemente, o Google lançou a versão 5.0 da plataforma chamada Lollipop; como ainda é muito recente, não tem dados estatísticos oficiais no site oficial do Android. Essa nova versão vem com muitas novidades e houve mudanças significativas na plataforma, é uma aposta muito grande do Google e vem com conceitos que influenciam diretamente no desenvolvimento de aplicativos e jogos para o Android.

Nos próximos parágrafos vamos entender melhor sobre a arquitetura da plataforma.

Todo usuário de Android sabe as funções básicas da plataforma, como fazer uma chamada, enviar uma mensagem de texto, mudar as configurações do sistema, instalar ou desinstalar aplicativos etc. Bem! Todos os usuários do Android sabem isso, mas não é o suficiente para ser um desenvolvedor Android. Então, vou explicar as camadas que compõem a plataforma Android para que você inicie o caminho como desenvolvedor Android.



A figura acima mostra todas as camadas da plataforma e podemos ver que a base de todo Android é desenvolvida baseada em **Linux**:

- Aplicativos: São os aplicativos e jogos desenvolvidos em Java.
- Frameworks, serviços bibliotecas, geralmente, também escritos em Java e servem para facilitar o desenvolvimento de aplicativos e jogos.
- Bibliotecas e serviços nativos são recursos que já vêm com o Android, para serem utilizados pelo desenvolvedor.
- Linux é a base de tudo e inclui todos os drivers de hardware e redes (Bluetooth, câmeras, USB, GPS's etc), sistemas de arquivos e processamento.

Os aplicativos e jogos são desenvolvidos, utilizando a linguagem de programação Java e, hoje pela internet, existe muito material sobre essa linguagem desde o básico até o mais avançado.

Meu primeiro Projeto Android

Neste capítulo, vou ensiná-lo a criar seu primeiro **Projeto Android**. Para isso, é preciso ter o ambiente de desenvolvimento configurado já com a ferramenta **Eclipse ADT** e o **Android SDK**.

Caso ainda não tenha seu ambiente configurado, dê uma olhada no Post escrito por mim [Android: Configurando o Ambiente de Desenvolvimento](#) no Blog [Produção de Jogos](#).

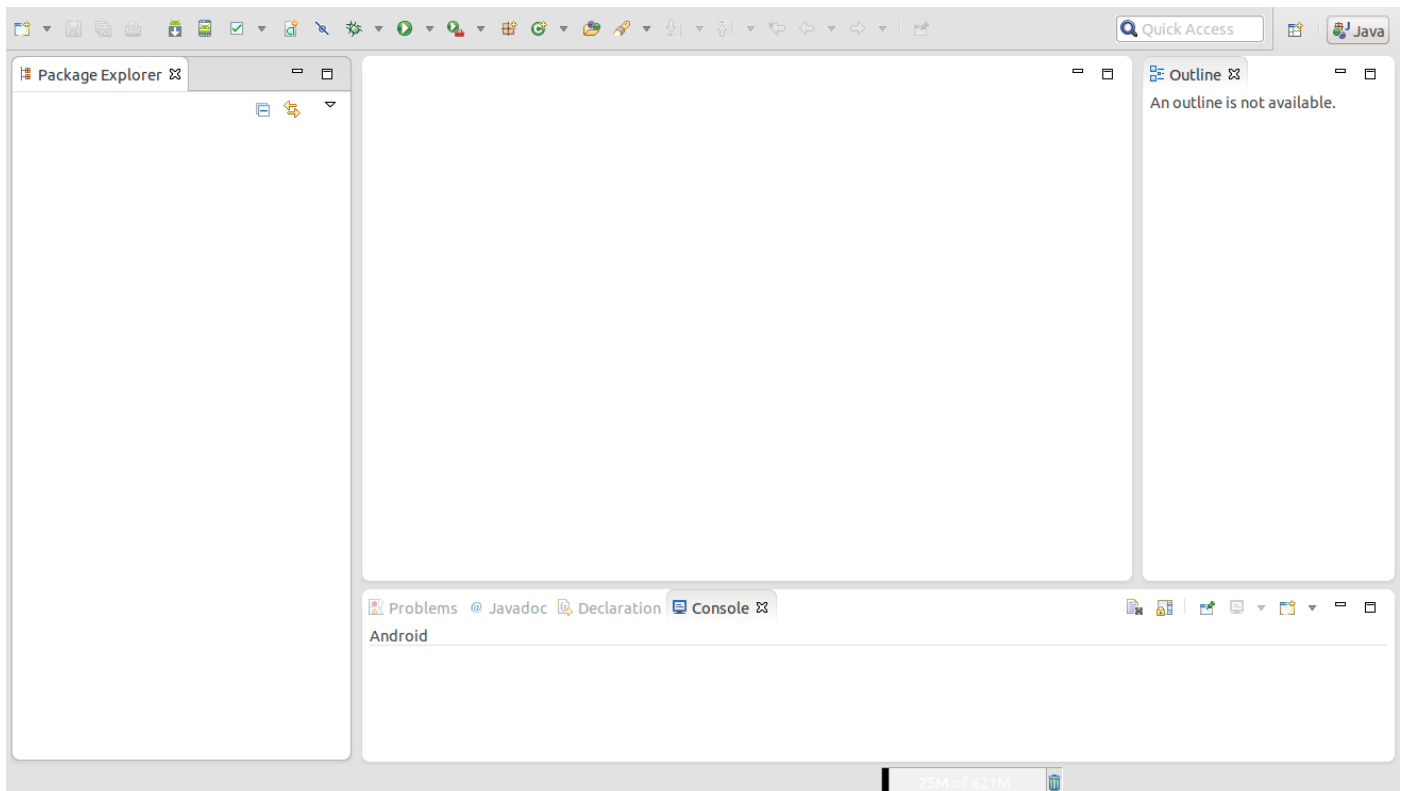
Se você já configurou seu ambiente, é hora de colocar a mão na massa.



Já que esta é sua primeira vez, criando um aplicativo Android, inicie a criação do seu projeto pelo caminho mais simples com a ajuda da ferramenta **Eclipse ADT**.

Execute o **Eclipse ADT** que está localizado dentro da pasta do **Android SDK**, conforme visto na configuração do ambiente de desenvolvimento.

Ele deve se parecer com a imagem abaixo:

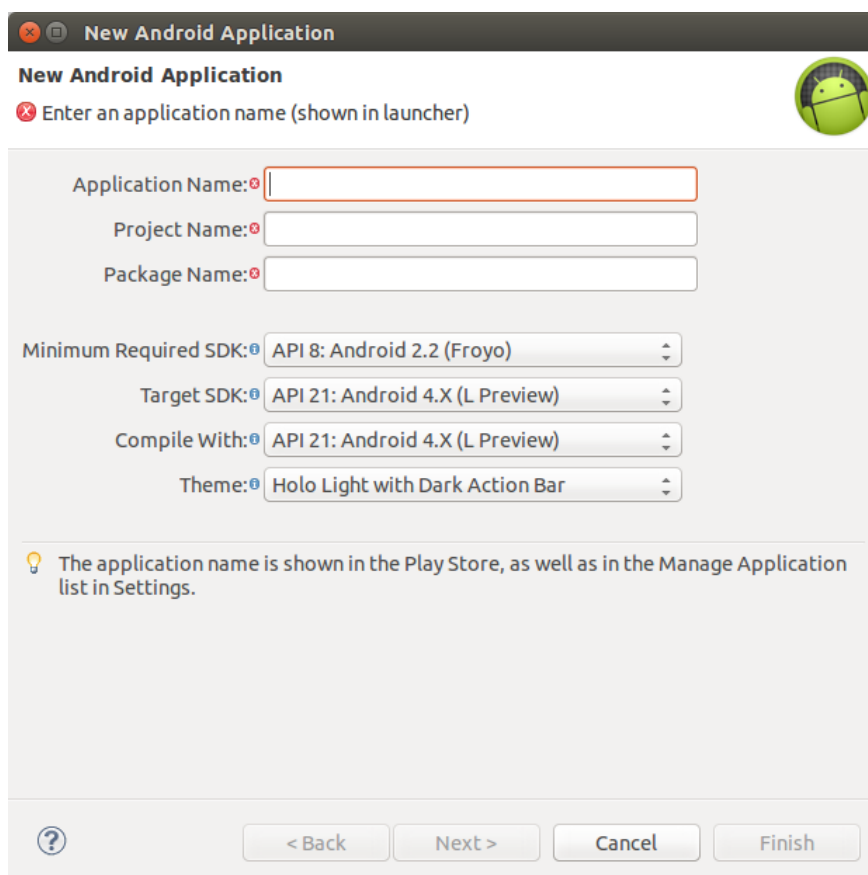


À primeira vista, o **Eclipse ADT** pode parecer um pouco confuso, com tantos botões, funcionalidades e telas, mas não se preocupe com isso agora. A princípio, ignore todos os botões e funcionalidades que ainda não aprendeu que, no decorrer do livro, vou explicar alguns deles.

Agora crie seu projeto Android, acessando o menu da seguinte forma:

- **Arquivo > Novo > Projeto de Aplicativo Android**
- Ou em inglês: **File > New > Android Application Project**

Deve aparecer uma tela como a figura a seguir:



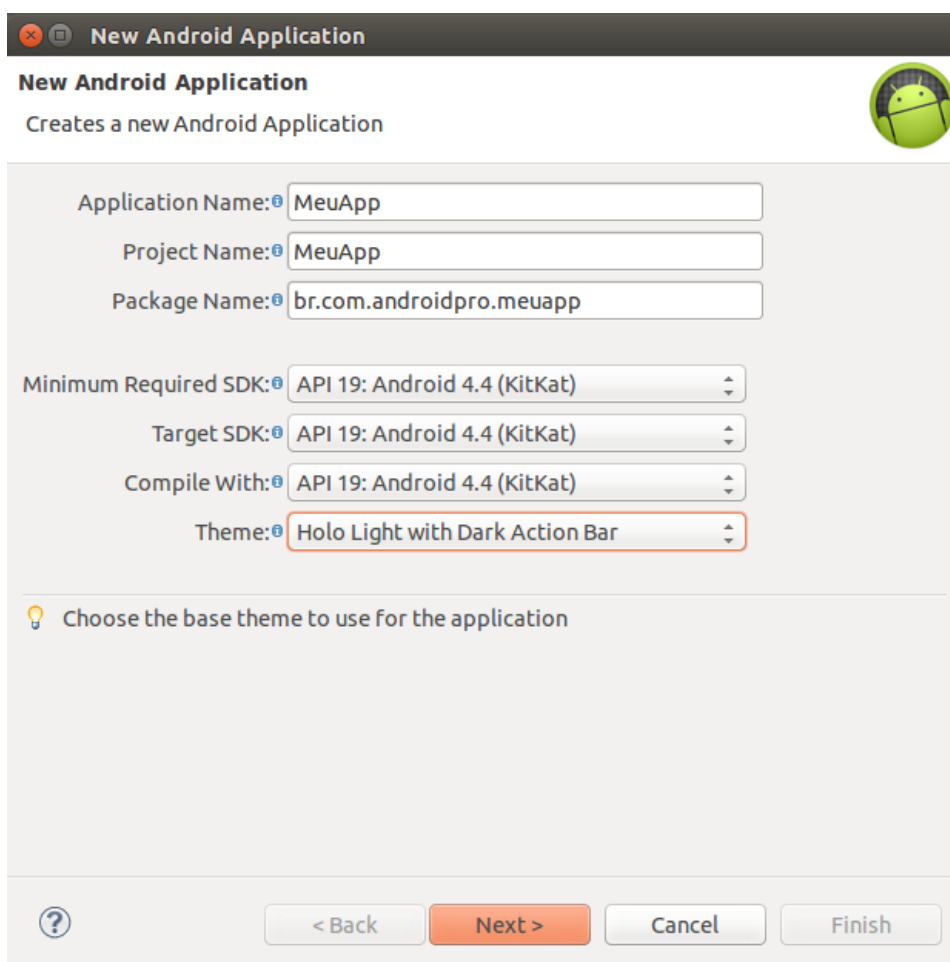
The screenshot shows the 'New Android Application' dialog box. At the top, it says 'New Android Application' with a red 'X' icon and a green Android robot icon. Below this, it says 'Enter an application name (shown in launcher)'. The dialog contains several input fields and dropdown menus: 'Application Name' (with a red border), 'Project Name', 'Package Name', 'Minimum Required SDK' (set to 'API 8: Android 2.2 (Froyo)'), 'Target SDK' (set to 'API 21: Android 4.X (L Preview)'), 'Compile With' (set to 'API 21: Android 4.X (L Preview)'), and 'Theme' (set to 'Holo Light with Dark Action Bar'). At the bottom, there is a lightbulb icon with a note: 'The application name is shown in the Play Store, as well as in the Manage Application list in Settings.' Below the note are four buttons: '?', '< Back', 'Next >', 'Cancel', and 'Finish'.

Explicarei os elementos dessa primeira tela antes de você começar as configurações:

- **Application Name:** Nome do aplicativo que será configurado automaticamente. É o nome que aparece no ícone, quando você instala um aplicativo no seu celular.
- **Project Name:** É o nome do projeto dentro do **Eclipse ADT**. Não, necessariamente, precisa ser o mesmo que o **Application Name**.
- **Package Name:** É o nome do pacote onde ficarão seus códigos Android. No desenvolvimento **Java**, é uma boa prática utilizar desta forma, considerando que estamos no Brasil, **br.com.suaaplicacao**.

- **Minimum Required SDK:** É a versão mínima do Android, suportada pelo seu aplicativo.
- **Target SDK:** É a versão máxima do Android, suportada pelo seu aplicativo.
- **Compile With:** É a versão do Android que vamos utilizar para compilar o projeto, ou seja, onde estão as bibliotecas nativas da plataforma para gerar o “executável” do aplicativo. Essa versão costuma ser a mesma configurada em **Minimum Required SDK**.
- **Theme:** É um tema padrão que o **Eclipse ADT** vai configurar no seu projeto automaticamente, com alguns componentes.

Agora, **preencha** os dados da tela conforme a figura abaixo:



New Android Application
Creates a new Android Application

Application Name:

Project Name:

Package Name:

Minimum Required SDK:

Target SDK:

Compile With:

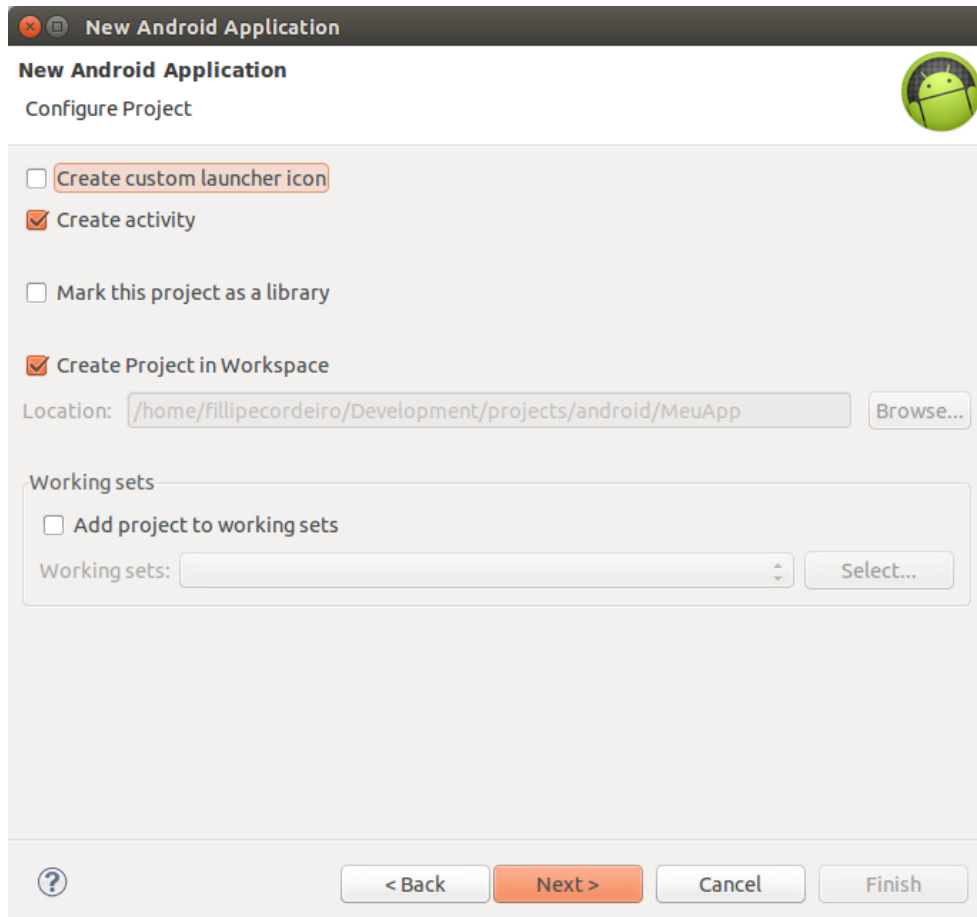
Theme:

Choose the base theme to use for the application

< Back **Next >** Cancel Finish

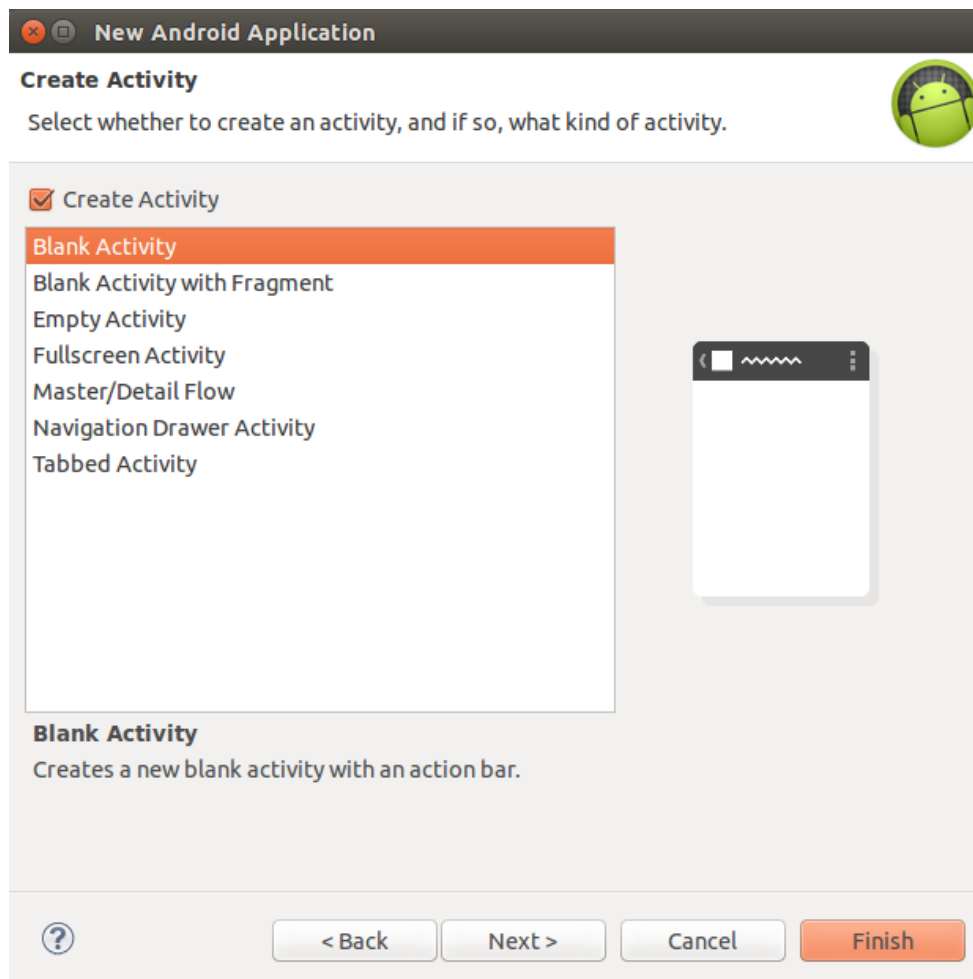
Clique no botão **Next >** para ir para o próximo passo da configuração.

Na tela abaixo, apenas desmarque a opção **Create custom launcher icon**. Desmarcar essa opção significa optar por não utilizar, nesse momento, nenhum logo ou ícone customizado em seu aplicativo, desse modo, o ícone utilizado será o padrão do Android.



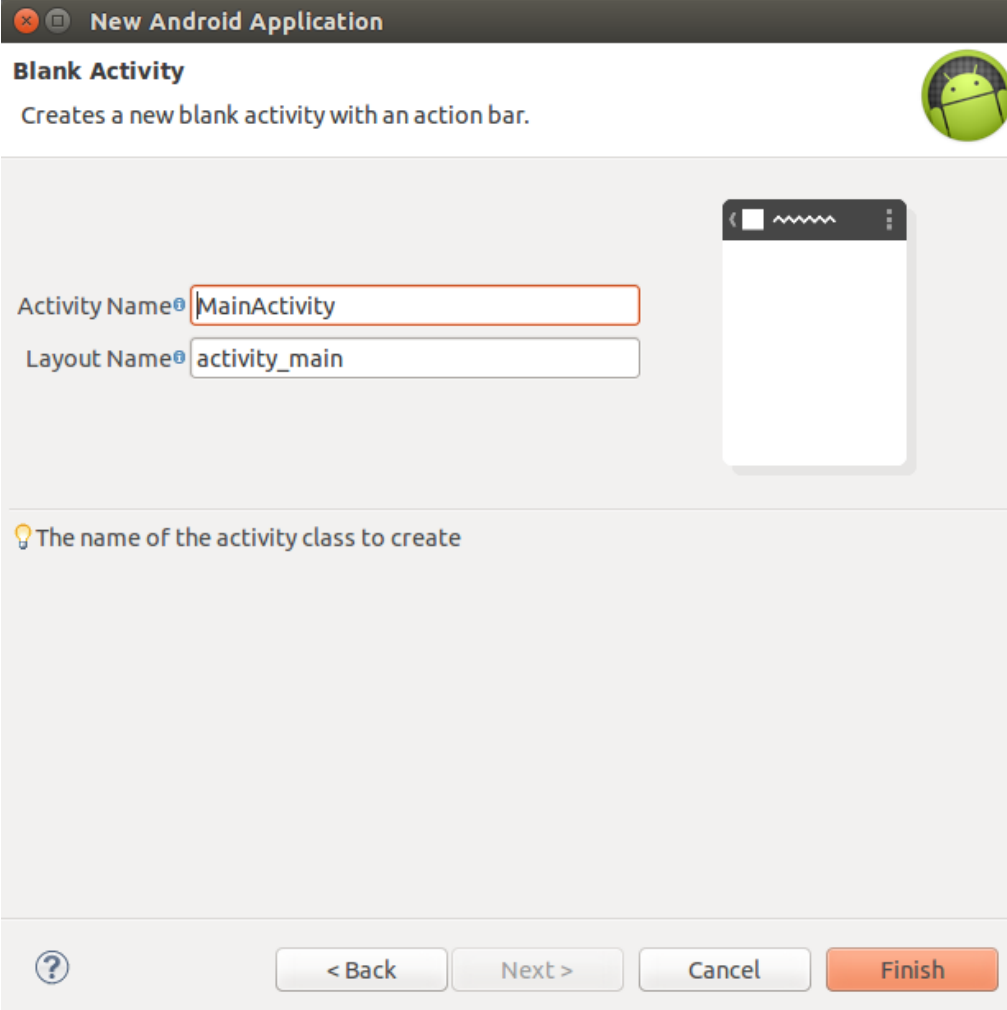
Clique no botão **Next >** para ir para o próximo passo da configuração.

Agora, escolha um layout inicial para seu projeto. Nesse caso, escolha a opção mais básica, chamada **Blank Activity**.



Clique no botão **Next >** para ir para o próximo passo.

Pronto, você está muito perto de criar seu primeiro projeto Android. Clique no botão **Finish** para terminar todo o processo.



New Android Application

Blank Activity
Creates a new blank activity with an action bar.

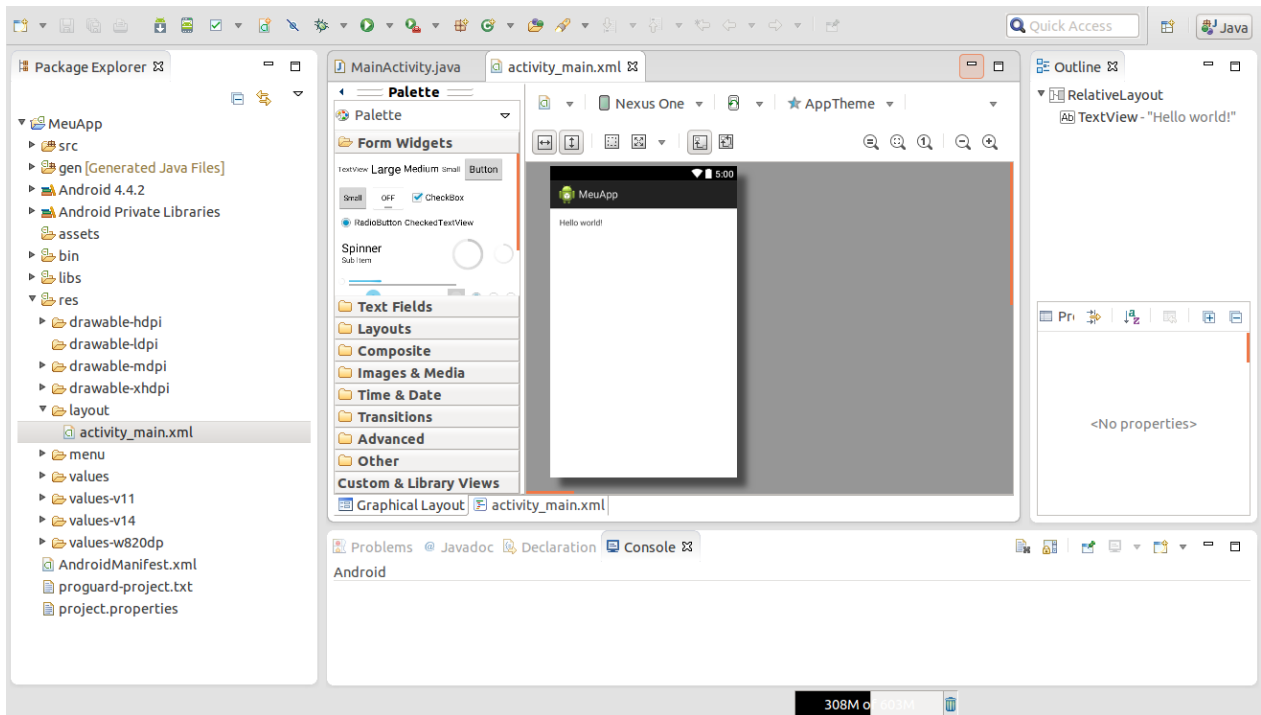
Activity Name@ MainActivity

Layout Name@ activity_main

⚡ The name of the activity class to create

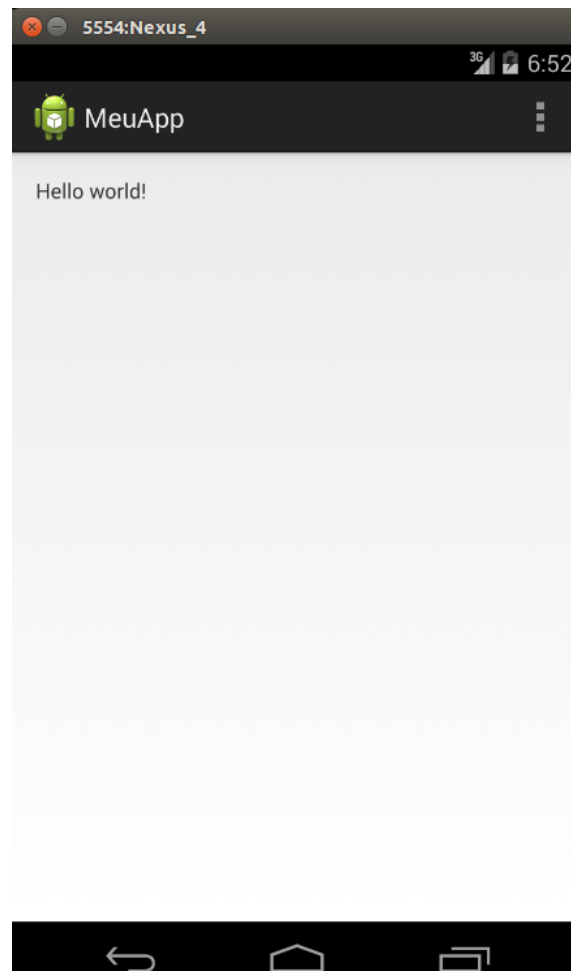
? < Back Next > Cancel Finish

Se tudo ocorreu bem, sua tela estará parecida com o exemplo a seguir.



Agora, você vai executar o projeto, usando o emulador que ensinei a criar no Post [Android: Configurando o Ambiente de Desenvolvimento](#) no Blog [Produção de Jogos](#). (no capítulo Configurando o Ambiente de Desenvolvimento)

Clique com o **botão direito do mouse**, em cima do seu projeto **MeuApp**, e siga para o menu **Run As > Android Application**.



Parabéns!

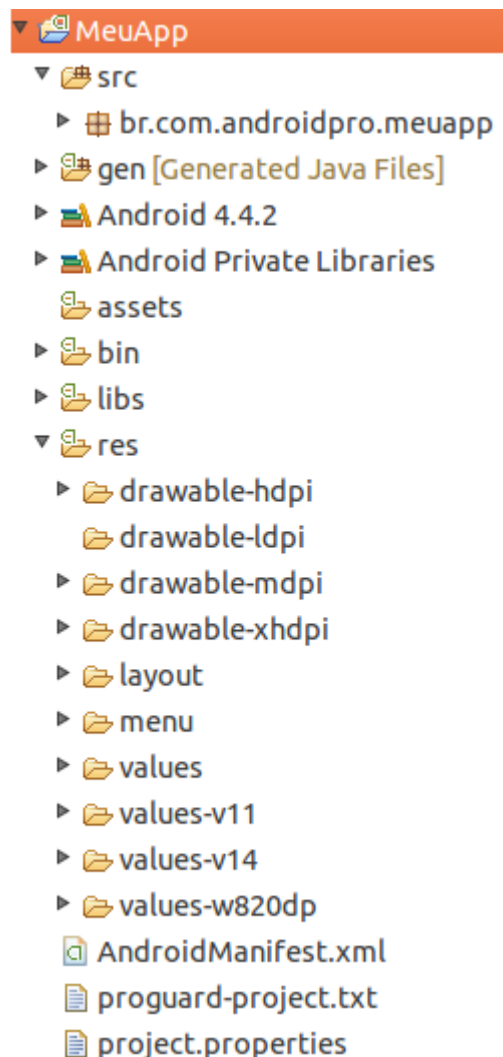
Você criou seu primeiro projeto Android :D



Entendendo o Projeto Android

Agora que você já criou seu primeiro projeto Android, vai entender do que é feito o projeto, seus arquivos e pastas. Nesse primeiro momento, não vou explicar tudo o que compõe o projeto, mas apenas o necessário para você entender por agora.

Não se preocupe em tentar decorar tudo o que cada pasta ou arquivo faz, pois no decorrer do seu aprendizado, isso será algo comum para você.



Primeiro, você entenderá o arquivo mais importante do projeto Android: o **AndroidManifest.xml**.

O **AndroidManifest.xml** é responsável por definir as características do projeto como permissões, versão, logo, nome e seus componentes. Ao longo dos estudos, você aprenderá várias configurações desse arquivo.

Veja um exemplo do arquivo:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.androidpro.meuapp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="19" />

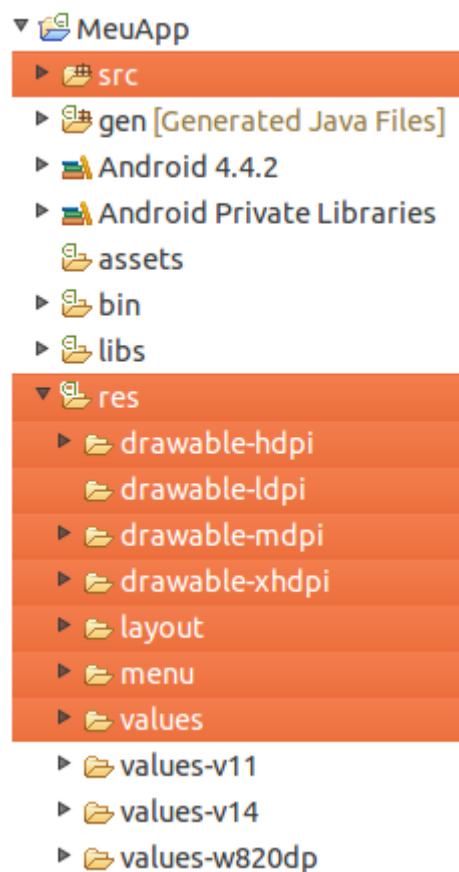
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>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

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

</manifest>
```

Outras partes que compõem seu projeto Android são:

- **src/**: É a pasta onde ficam as fontes do seu projeto, ou seja, as classes **Java** que você desenvolve.
- **res/**: Contém vários recursos do projeto como imagens, layout, xmls de configuração como veremos a seguir.
 - **drawable/**: existem várias pastas drawable que contêm as imagens utilizadas no projeto. Cada uma das pastas contém uma versão de uma determinada imagem, separadas por definição de tela.
 - **layout/**: nesta pasta, ficam os arquivos responsáveis pelo design das telas do seu projeto.
 - **menu/**: nesta pasta, ficam os arquivos xml referente aos menus do seu projeto.
 - **values/**: contém outras configurações em xml para parametrização do projeto como cores, mensagens dentre outras.

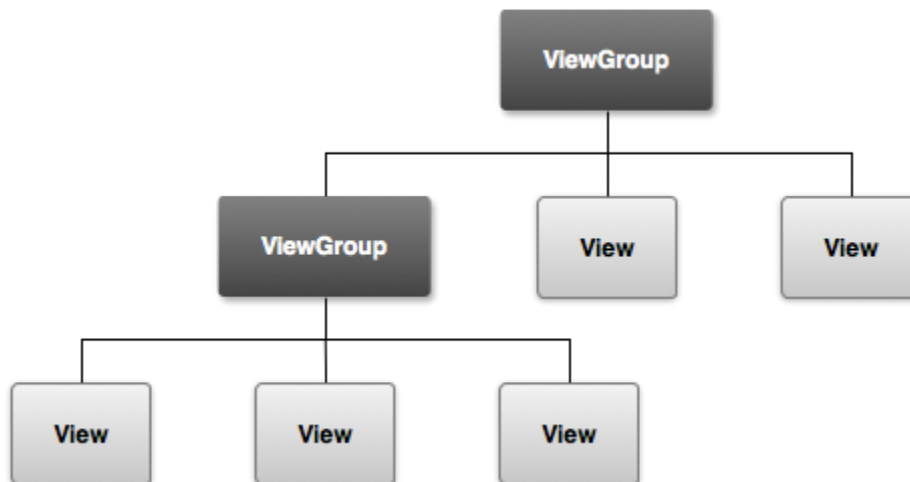


Interface do Projeto Android

Neste capítulo, você estudará um pouco a interface gráfica do Android, como ela funciona e sua estrutura.

Basicamente, todos os componentes visuais do Android são baseados em **View** e **ViewGroup**. As **Views** são componentes como botões, campos de texto, combo-box dentre outros. Os **ViewGroups** são componentes de organização de layout como listas e tabelas, ou seja, agrupamento de **Views**.

A hierarquia dos componentes pode ser representada desta forma:



Para entender melhor sobre o View e ViewGroup, você vai usar seu projeto como exemplo, abra o arquivo **activity_main.xml**.

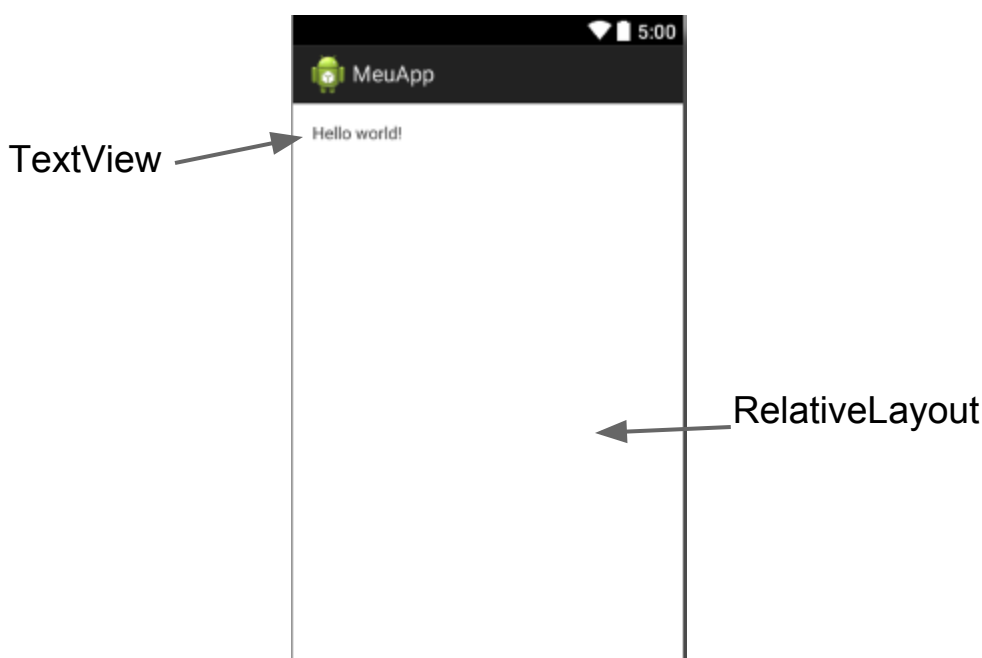
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.com.androidpro.meuapp.MainActivity" >

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

</RelativeLayout>
```

Nesse exemplo, há dois componentes que compõem a interface da tela principal do projeto:

- **RelativeLayout**: é um **ViewGroup** que posiciona os componentes filhos em relação uns aos outros ou em relação ao próprio **RelativeLayout**.
- **TextView**: é uma **View** que basicamente serve para mostrar um texto para o usuário.



Agora, coloque a mão na massa, adicionando dois componentes novos em sua tela principal. Remova o **TextView** e adicione um **EditText** e um **Button**. O arquivo vai ficar dessa forma:

```
<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="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.com.androidpro.meuapp.MainActivity" >

    <!-- Campo texto para entrada de dados pelo usuário -->
    <EditText
        android:id="@+id/editTextNome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="Digite seu nome"/>

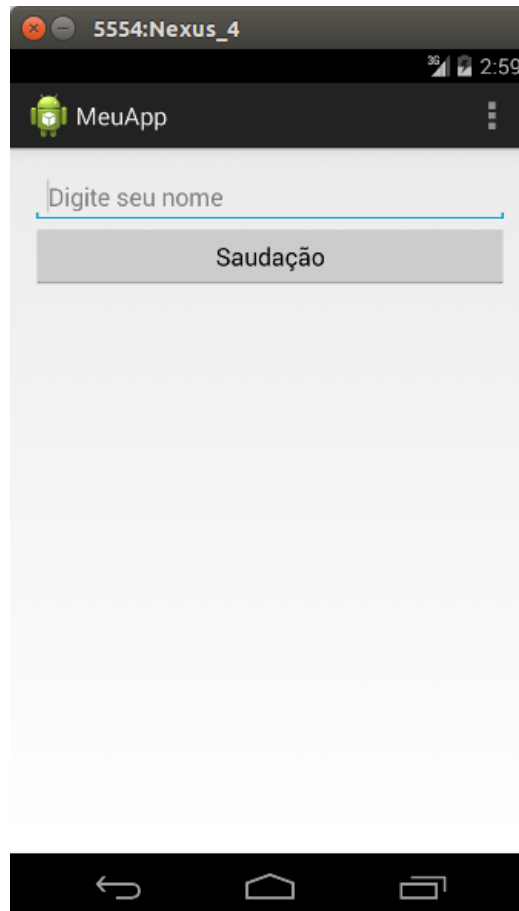
    <!-- Botão de ação -->
    <Button
        android:id="@+id/buttonAcao"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/editTextNome"
        android:text="Saudação" />

</RelativeLayout>
```

Entenda alguns atributos importantes das **Views**:

- **android:layout_width/android:layout_height**: definem o tamanho do componente e são obrigatórios para todas as **Views**.
- **android:id**: é um identificador único que faz referência a **View**. Sempre que for definir um id para um componente, ele seguirá essa forma **@+id/NOME_DO_ID**, onde o **@+** indica que estamos criando um id para aquela **View**.
- **android:hint**: é um valor padrão que é mostrado, quando o campo texto está vazio.
- **android:layout_below**: posiciona o componente abaixo de outra **View**.

Execute o projeto no emulador, para ver como ficou seu aplicativo. Lembrando que, para executar seu projeto, clique com o botão direito do mouse em cima do projeto e vá em **Run As > Android Application**.



Bom, você já deu uma “cara” diferente para seu projeto, usando novos componentes. Por mais simples que seja, já é um aplicativo que pode interagir com o usuário.

Voltando um pouco para o layout principal **activity_main.xml**, você verá que o **Eclipse** está sublinhando de amarelo duas linhas do seu arquivo e mostrando a seguinte mensagem.

“[I18N] Hardcoded string “Digite seu nome”, should use @string resource”

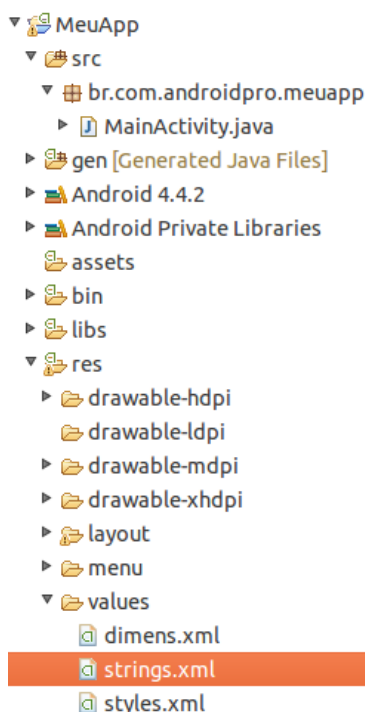
“[I18N] Hardcoded string “Saudação”, should use @string resource”

Mas o que isso quer dizer?

A ferramenta está avisando-o de que não está seguindo uma boa prática de desenvolvimento e está colocando as mensagens direto na sua **View**. Ele ainda indica para usar uma **@string resource** no lugar das mensagens.

Qualquer texto que for utilizado no seu projeto, sempre deve estar dentro de **Resources**. Os **Resources** permitem que você gerencie suas mensagens e outros dados em um único local e ajuda, também, a adicionar suporte de vários idiomas ao seu aplicativo.

Os **Resources** ficam na pasta **values** do seu projeto e, nesse caso, irá utilizar o **strings.xml**, para colocar seus textos.



Abra o arquivo **strings.xml**.

Pode-se ver que já existem algumas **strings** configuradas no arquivo.

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

    <string name="app_name">MeuApp</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

Vamos adicionar mais duas **strings** no arquivo, que deve ficar assim:

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

    <string name="app_name">MeuApp</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="hint_edit_mensagem">Digite seu nome</string>
    <string name="text_btn_acao">Saudação</string>

</resources>
```

Basicamente, você adicionou uma **tag string** com um conteúdo e deu um nome a ela.

`<string name="nome_x">Mensagem X</string>`

Lembrete: todas as mensagens utilizadas no seu projeto devem estar dentro desse arquivo e nunca diretamente no código ou nos arquivos de layout.

Legal! Você adicionou as **strings** dentro do arquivo, agora precisa ir para o arquivo **activity_main.xml** e trocar as mensagens pela referência delas.

O arquivo **activity_main.xml** vai ficar assim.

```
<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="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="br.com.androidpro.meuapp.MainActivity" >

    <!-- Campo texto para entrada de dados pelo usuário -->
    <EditText
        android:id="@+id/editTextNome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:hint="@string/hint_edit_mensagem"/>

    <!-- Botão de ação -->
    <Button
        android:id="@+id/buttonAcao"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/editTextNome"
        android:text="@string/text_btn_acao" />

</RelativeLayout>
```

Veja que trocou as mensagens pelas referências delas. Dessa forma, fica muito mais fácil gerenciar todas as mensagens do projeto.

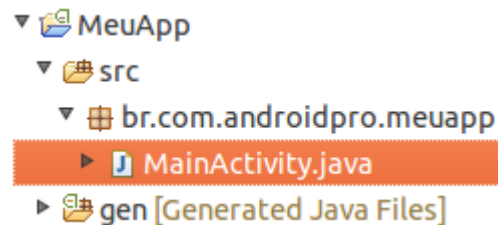
Programação do Projeto Android

Neste capítulo, você verá um pouco de programação e entenderá como interagir com a interface gráfica do projeto.

É preciso mostrar uma saudação para o nome que foi digitado no campo de texto; para isso, você precisa colocar uma ação no botão onde, quando o usuário clicar, sua ação de saudação aconteça, mostrando uma mensagem para o usuário junto com o que foi digitado no campo texto.



Abra o arquivo **MainActivity.java**.



Essa é uma **Activity**, um dos componentes mais importantes do Android. Ela é responsável por fazer a interação da interface gráfica com outras funcionalidades e bibliotecas.

```
public class MainActivity extends Activity {

    @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 present.
        getMenuInflater().inflate(R.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();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Todas as telas do seu projeto devem **estender** uma **Activity**, dessa forma você fala para o Android que aquela sua classe é uma **Activity** também.

Existe um método muito importante dentro de uma **Activity** que é o **onCreate**, ele é responsável por fazer a criação dos componentes na tela. Sempre que quiser trabalhar com um componente de tela usará o **onCreate**. Dentro do **onCreate**, há duas chamadas de métodos, o **super.onCreate(savedInstanceState)** que é o **onCreate** nativo da plataforma e o **setContentView(R.layout.activity_main)** que é onde relacionamos nosso layout com nossa **Activity**.

Sendo assim, você precisa resgatar os componentes pelos **id's**, para poder trabalhar com eles. A **Activity** do Android tem um método chamado **findViewById** onde pode pegar seu campo de texto e o botão.

Veja abaixo como fica seu código para fazer isso dentro do método **onCreate**:

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        // Resgatamos nossos componentes pelo id  
        final EditText editTextNome = (EditText) findViewById(R.id.editTextNome);  
        final Button btnAcao = (Button) findViewById(R.id.buttonAcao);  
    }  
  
    ...  
}
```

Agora que já tem seus componentes, pode começar a trabalhar com eles para atingir seu objetivo, mostrar uma mensagem de saudação para o nome digitado no campo de texto.

Agora, você vai adicionar uma ação em seu botão, utilizando o método **setOnClickListener**. Dentro do clique do botão, pega o nome dentro do campo de texto e mostra uma mensagem na tela do usuário, utilizando um **Toast**. O **Toast** é uma mensagem rápida que aparece para o usuário e desaparece depois de alguns segundos.

```
public class MainActivity extends Activity {

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

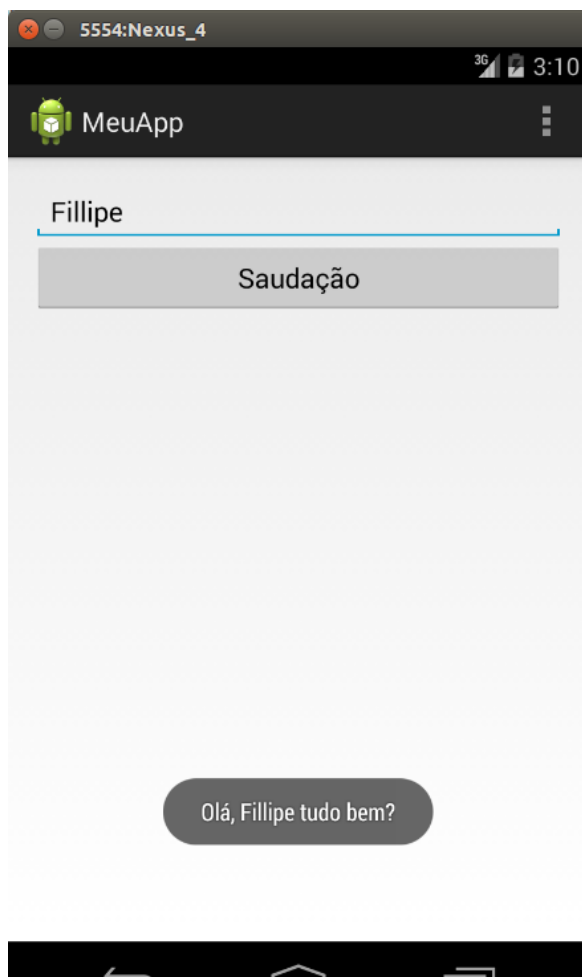
        // Resgatamos nossos componentes pelo id
        final EditText editTextNome = (EditText) findViewById(R.id.editTextNome);
        final Button btnAcao = (Button) findViewById(R.id.buttonAcao);

        // Adicionando uma ação no clique do botão
        btnAcao.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Pegamos o conteudo do campo de texto
                String nome = editTextNome.getText().toString();

                // Mostramos uma mensagem na tela do usuário
                Toast.makeText(MainActivity.this, "Olá, " + nome + " tudo
bem?", Toast.LENGTH_LONG).show();
            }
        });
    }

    ...
}
```

Execute seu projeto no emulador e veja o resultado.



Muito bem! Você cumpriu o objetivo, fez uma saudação personalizada para qualquer nome digitado no campo de texto.



Conclusão

Esse, foi apenas o início da sua jornada para se tornar um desenvolvedor de aplicativos para Android.

Porém, antes de partir, gostaria de deixar dois desafios para você como um bônus. Desafios são sempre bons para você começar a pensar na resolução de problemas dentro da plataforma Android.

Espero que nos vejamos em breve, com mais oportunidades, muito mais conteúdo e um maior grau de dificuldade.

Até já!

#1 Desafio: Boas práticas

Adicione a mensagem de saudação dentro de **strings.xml** e o utilize dentro do **Toast**.

Dica: para recuperar a mensagem na Activity utilize o método `getString(resId)`

#2 Desafio: Componentes da tela

Adicione mais um campo sobrenome abaixo do campo nome e o utilize na mensagem de saudação.