

Curso Desenvolvendo Aplicativos para Android



ANDROID



Este material é exclusivo do canal (youtube): canalfessorbruno e do site www.cfbcursos.com.br
não pode ser distribuído ou comercializado por outra fonte.



Apostila: versão 2.0 - Básico

20/04/2016

www.youtube.com/canalfessorbruno

www.cfbcursos.com.br

canalfessorbruno@gmail.com

www.facebook.com/canalfessorbruno

twitter: @fessorBruno



Sumário

Introdução.....	5
Instalando	5
Instalando o Android Studio	10
Criando o primeiro projeto	14
Android Virtual Device Manager.....	17
Rodando nossa aplicação.....	20
O ambiente do Android Studio	22
Introdução sobre programação básica	23
Variáveis.....	24
Operadores / Operações com variáveis.....	26
Modificadores	28
IF – Comando de decisão	29
IF - ELSE	29
IF – ELSE – IF.....	30
Operador ternário / IF ternário.....	31
Switch.....	31
Switch com faixas de valores	32
Comando de Loop FOR	32
Comando de loop while	32
Do while	33
Métodos definidos pelo programador.....	34
Métodos com parâmetros de entrada.....	34
Métodos com retorno.....	34
Criando novas classes	35
Construindo a Interface com o usuário	36
A estrutura de um aplicativo no Android Studio	42
Adicionando comando ao botão.....	45
Segundo aplicativo – Calculadora básica	50
Terceiro projeto – LocaFest – Aplicativo para locar itens para festas	59
A classe principal AppCompatActivity.....	63
Criando e Abrindo novas telas - XML.....	67
Definindo o menu de opções.....	74
Agenda básica com banco de dados	78
Trabalhando com várias Activities	95
Intent.....	101



Passando parâmetros em chamadas de Activities usando Bundle.....	101
Passando parâmetros usando o próprio Intent.....	103
Operações com Intent.....	104
Acessando página da Internet	104
Abrindo o discador com um número de contato da agenda	106
Abrir um determinado contato e ver todas suas informações.....	108
Obtendo o nome de um contato da agenda.....	109
Criando um classe para caixas de mensagem.....	110
Aplicação em segundo plano	113
Notificações	124
Gerando o APK (não assinado) e instalando no Telefone.....	116
Criando uma assinatura para seus aplicativos.....	117
Publicar o aplicativo na Google Play	122



Introdução

O sistema operacional Android se difundiu muito nos últimos anos e cresce mais a cada dia, hoje não está mais limitado a telefones celulares, mas também é usado em tablets, automóveis, TVs e relógios.

Este curso tem o objetivo de introduzir você ao mundo do desenvolvimento de aplicativos para Android, esta apostila é um material básico/introdutório, esta apostila não se aprofunda em lógica de programação e nem em programação Java, somente o necessário para desenvolver o exemplos e mostrar à você como desenvolver um aplicativo para Android usando o Androis Studio.

Instalando

Os passos que iremos seguir neste processo são:

- a) Instalar o JDK
- b) Configurar o path do JDK no Windows
- c) Instalar o Android Studio

Então, vamos ao primeiro passo que é instalar o JDK, para baixar use o link a seguir.

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

The screenshot shows the Oracle Java SE Downloads page. At the top, there's a navigation bar with links for Sign In/Register, Help, Country, Communities, Products, Solutions, Downloads, and Support. Below that is a breadcrumb trail: Oracle Technology Network > Java > Java SE > Downloads. On the left, there's a sidebar with links for Java SE, Java EE, Java ME, Java SE Support, Java SE Advanced & Suite, Java Embedded, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine. The main content area has tabs for Overview, Downloads (which is selected), Documentation, and Community. Below the tabs, it says "Java SE Downloads" and shows a "DOWNLOAD" button with the Java logo. Underneath, it says "Java Platform (JDK) 8u73 / 8u74". At the bottom, there's a section for "Java Platform, Standard Edition" with a link to "Java SE 8u73 / 8u74".

Clique no botão azul “download” a na tela a seguir, selecione a opção “Accept License Agreement” e clique no link de download correspondente a seu sistema operacional.

OBS: Obviamente que você irá baixar a versão mais recente, quando inicie esta apostila a versão mais recente era “8u74”, quando terminei a versão mais recente era “8u77”.

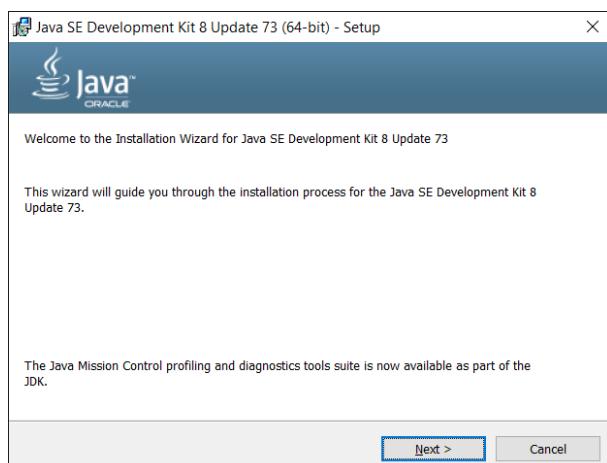


The screenshot shows the Oracle Java SE Development Kit 8 Downloads page. On the left, there's a sidebar with links like Java SE, Java EE, Java ME, etc. The main content area has tabs for Overview, Downloads (which is selected), Documentation, Community, Technologies, and Training. Below the tabs, there's a section titled "Java SE Development Kit 8 Downloads". It includes a brief description of what the JDK is, a "See also" section with links to developer newsletters, Java Developer Day events, and Java Magazines. At the bottom of this section, there's a note about accepting the Oracle Binary Code License Agreement. A yellow arrow points to the "Accept License Agreement" radio button. Below it is a table showing download links for various platforms.

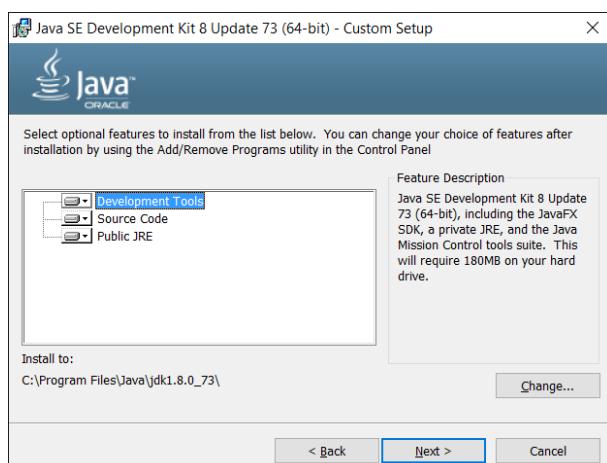
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.73 MB	[link]
Linux ARM 64 Hard Float ABI	74.68 MB	[link]
Linux i386	149.82 MB	[link]
Linux x86	174.91 MB	[link]
Linux x64	152.73 MB	[link]
Linux x64	172.91 MB	[link]
Mac OS X x64	227.25 MB	[link]
Solaris SPARC 64-bit (SVR4 package)	139.01 MB	[link]
Solaris SPARC 64-bit	99.08 MB	[link]
Solaris x64 (SVR4 package)	140.36 MB	[link]
Solaris x64	96.78 MB	[link]
Windows x86	181.5 MB	[link]
Windows x64	188.84 MB	[link]

Após baixar o Java SE Development Kit (JDK) instale-o.

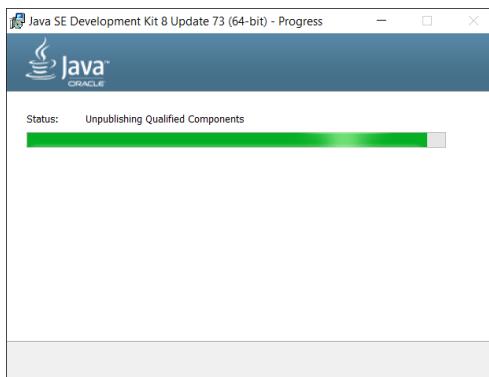
Vamos ao processo de instalação do JDK, na primeira tela não há nada o que fazer a não se clicar no botão “Next >”.



Na segunda tela podemos selecionar os recursos que serão instalados e o local da instalação, não vamos mudar nada, clique novamente em “Next >”.



Agora vamos aguardar e conferir o progresso da instalação.



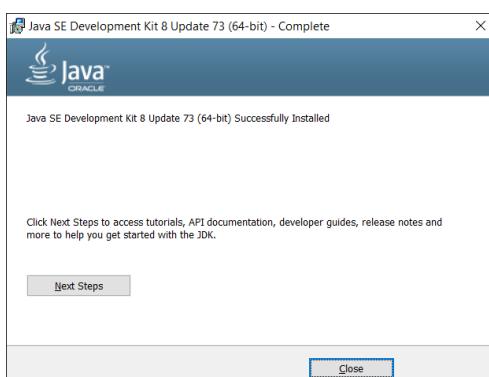
Quando terminar será mostrada uma janela para que possamos alterar a pasta de destino de instalação do Java, não iremos mudar, vamos deixar a pasta padrão, então clique no botão “Próximo >”.



Novamente vamos aguardar e conferir o progresso da instalação do Java.



Ao terminar basta clicar no botão “Close”.

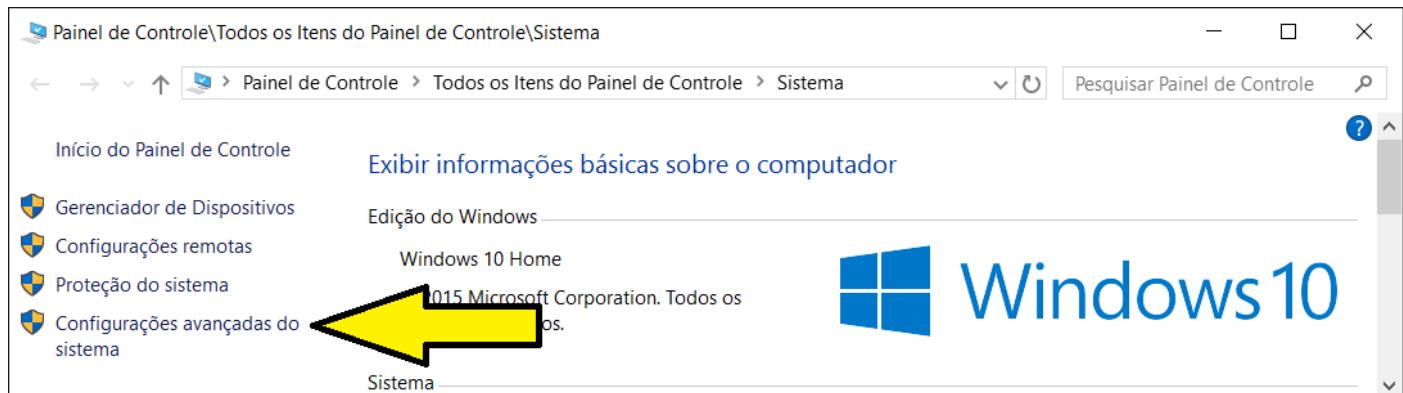


Configurando variáveis de ambiente.

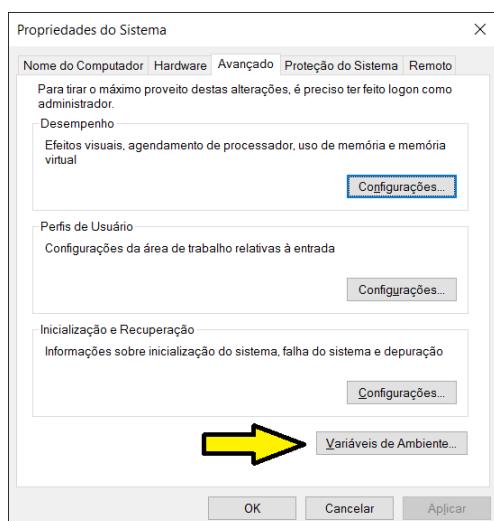


Neste capítulo vou mostrar como configurar o Windows para que o Android Studio possa encontrar o JDK.

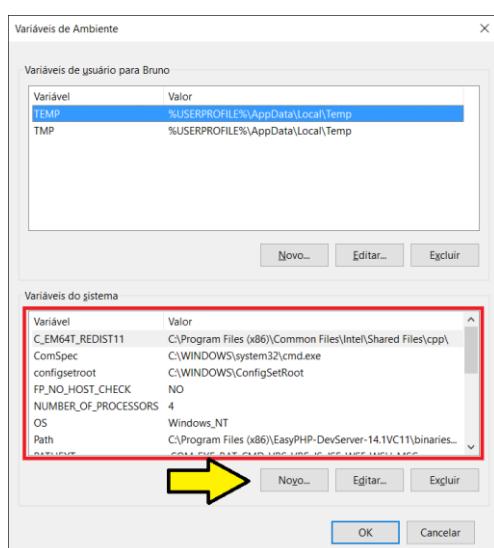
Na janela “Sistema” encontrada no “Painel de Controle”, clique no item “Configurações avançadas do sistema”.



Na janela que se abrirá, vá na aba “Avançado” e clique em “Variáveis de Ambiente...”

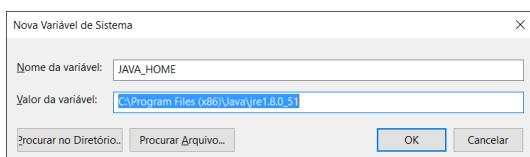


Na janela que se abrirá, procure pela variável “JAVA_HOME” na área destacada em vermelho, caso não encontre clique no botão “Novo”.

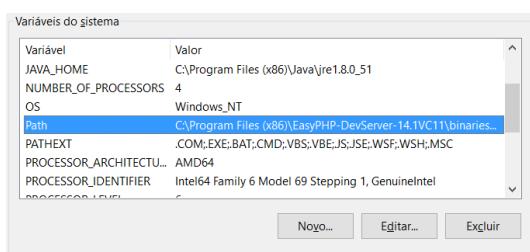




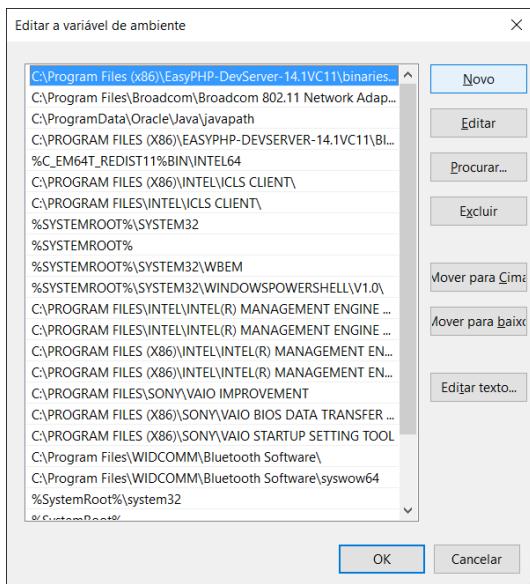
Digite o nome da variável “JAVA_HOME” sem aspas e o caminho onde instalou o Java Anteriormente, então clique em OK.



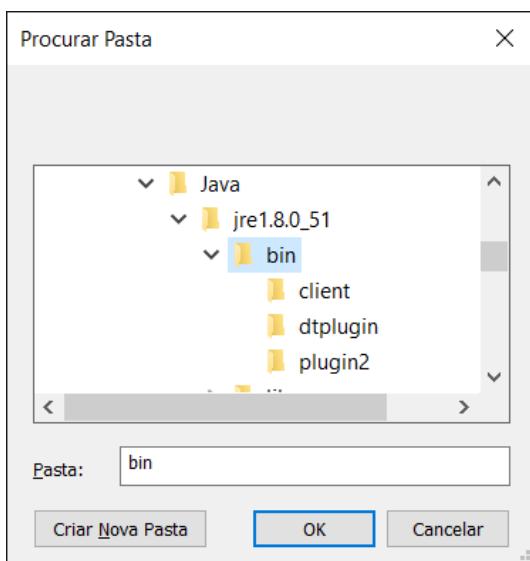
Ainda na área de variáveis do sistema encontre a variável “Path” e clique no botão “Editar...”

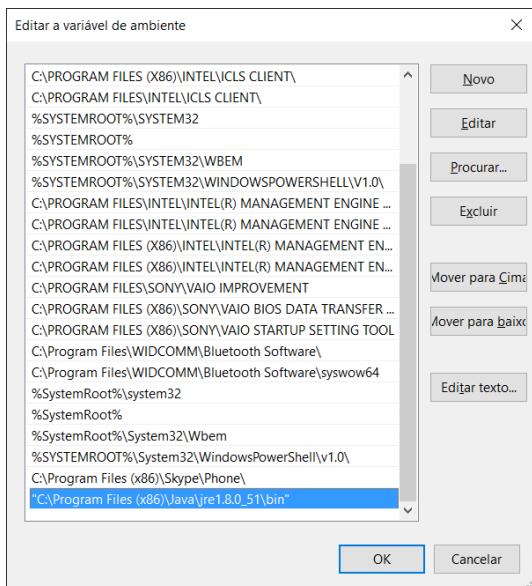


Clique no botão “Novo” e no botão “procurar” para adicionarmos o path do Java.

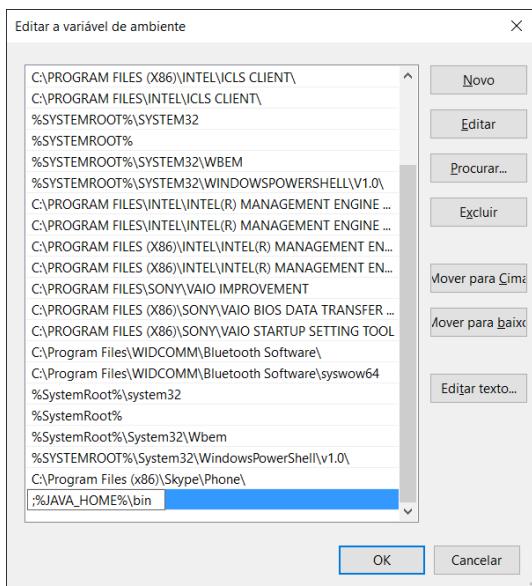


Encontre o local onde instalou o JAVA e selecione a pasta “bin”, clicando em OK.





Veja que foi adicionado o path para o Java, clique em OK



Clique em OK nas janelas anteriores.

Para testar, abra o prompt de comando e digite o comando “java –version” sem aspas, será mostrada a versão do Java instalado em seu sistema.

```
Prompt de Comando
Microsoft Windows [versão 10.0.10586]
(c) 2015 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Bruno>java -version
java version "1.8.0_73"
Java(TM) SE Runtime Environment (build 1.8.0_73-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.73-b02, mixed mode)

C:\Users\Bruno>
```

Instalando o Android Studio

Agora precisamos baixar e instalar o Android Studio use o link a seguir.



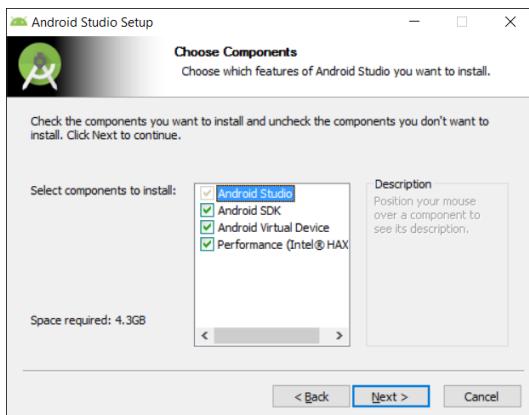
<http://developer.android.com/sdk/index.html>

Baixe e vamos iniciar a instalação.

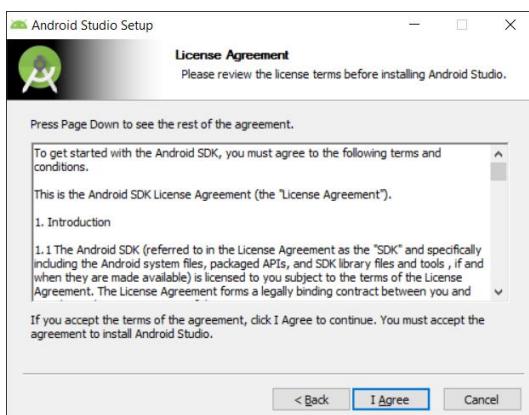
A primeira tela da instalação basta clicar no botão “Next >”.



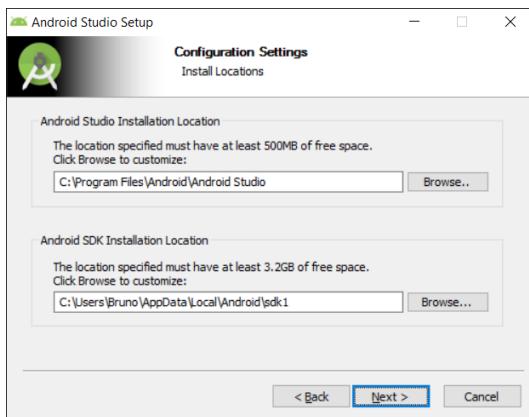
Na próxima tela podemos escolher os componentes que iremos instalar, selecione todos e clique no botão “Next >”.



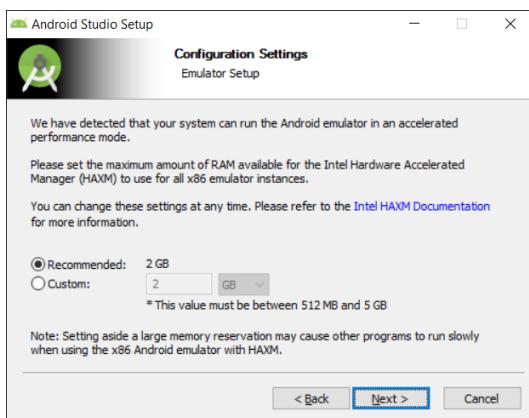
Aceite os termos de licença clicando no botão “I Agree”.



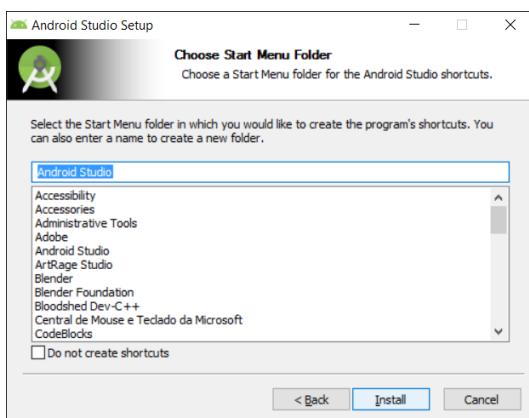
Nesta próxima tela podemos indicar os locais de instalação para o Android Studio e o Android SDK, não iremos mudar os locais padrões, então clique em “Next >”.



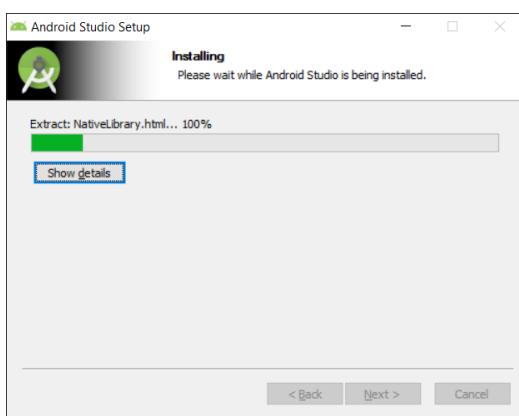
Na próxima tela clique em “Next >” novamente, use a opção “Recommended” e clique em “Next >”.

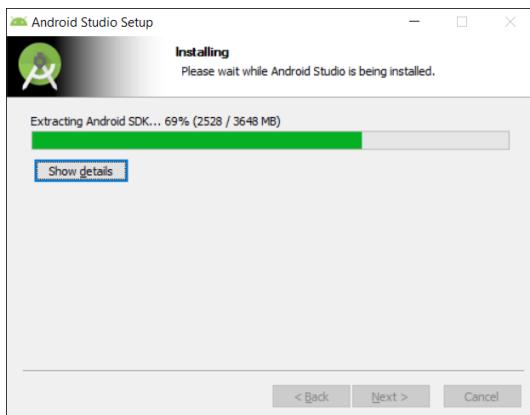


Escolha o nome da pasta no Menu do Windows ou deixe a padrão “Android Studio”, clique em “Install”.

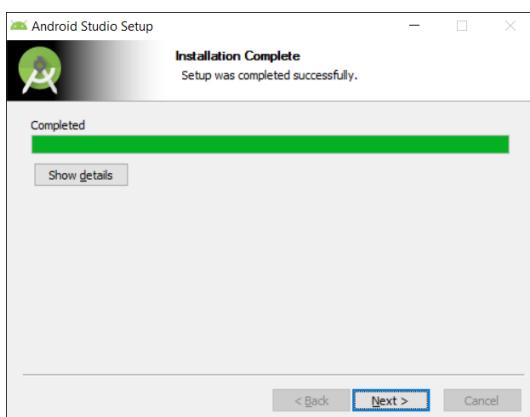


Agora aguarde a instalação.





Ao final do progresso clique em “Next >”.



A última tela clique no botão “Finish” e o Android Studio será aberto.

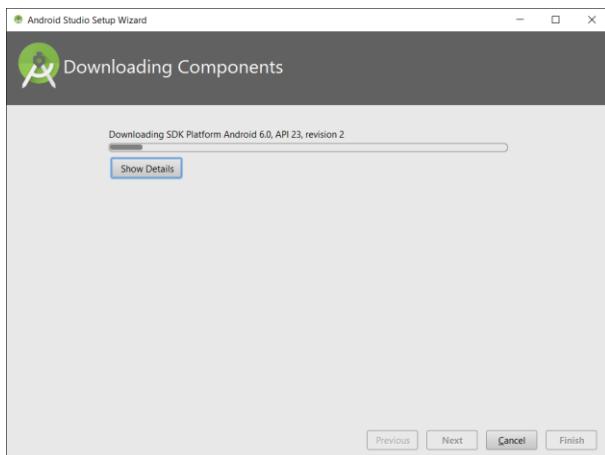


Aguarde o carregamento do Android Studio.

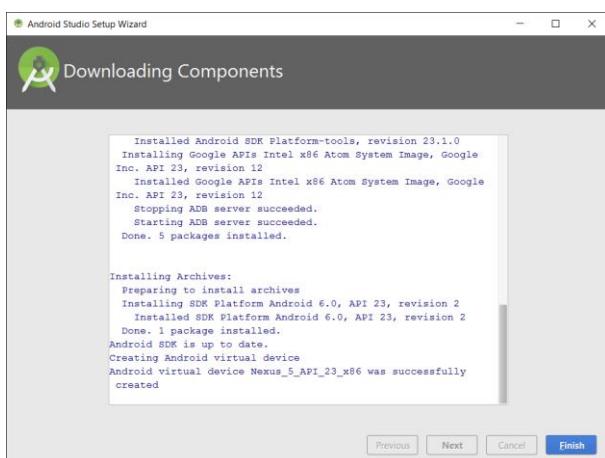




Antes de iniciar o programa será rodado o “Setup Wizard” que efetuará alguns downloads e irá analisar o computador em busca do JDK.



Ao terminar este processo clique no botão “Finish”.

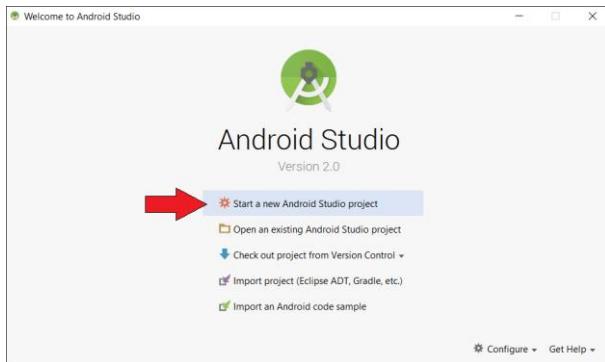


Então será aberta a primeira tela do Android Studio 2.0.

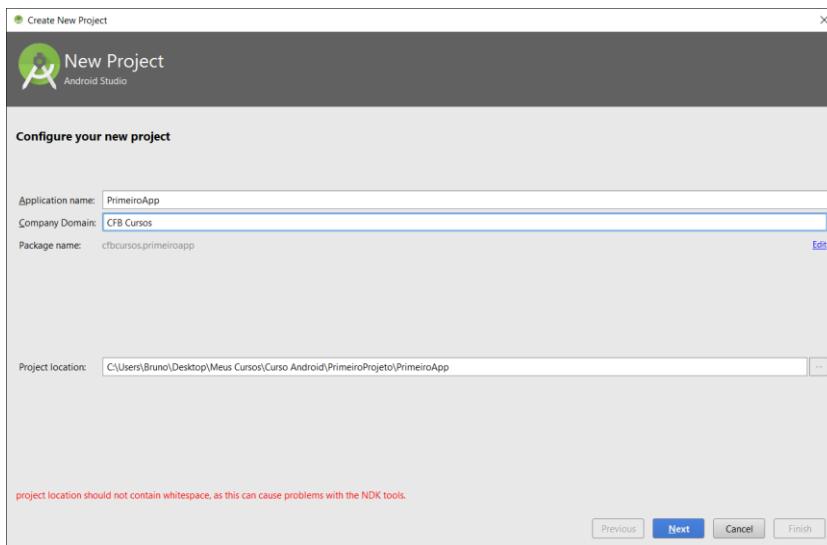


Criando o primeiro projeto

Para iniciar um novo projeto clique na opção “Start a new Android Studio Project”.



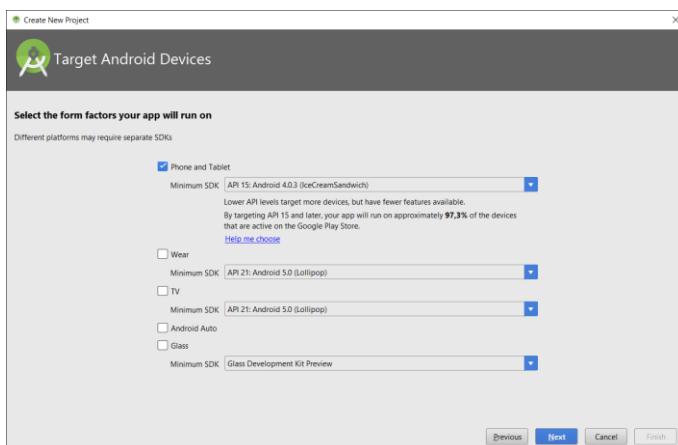
Na próxima tela, digite o nome do seu aplicativo, irei usar “PrimeiroApp” e o local onde será armazenado o projeto, após preencher estes campos clique no botão “Next”.



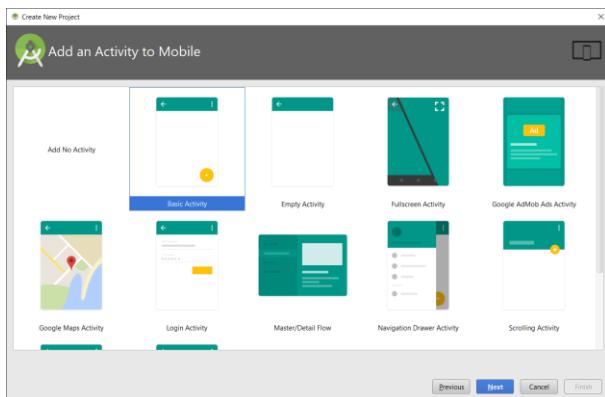
Na próxima tela selecione a versão mínima do Android que deseja trabalhar, lembre-se que se você usar a última versão do Android, somente dispositivos com esta versão rodarão seu aplicativo.

Note que além de celulares e tablets temos várias opções e dispositivos.

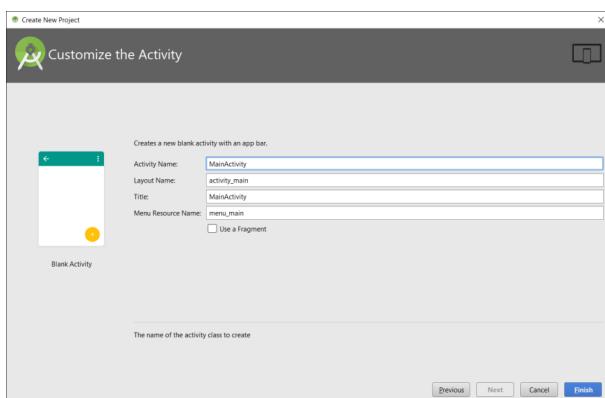
Após selecionar clique no botão “Next”.



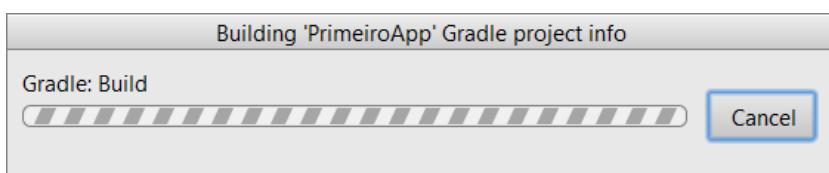
Na próxima janela vamos selecionar o layout inicial da nossa aplicação, selecione “Basic Activity” e clique em “Next”.



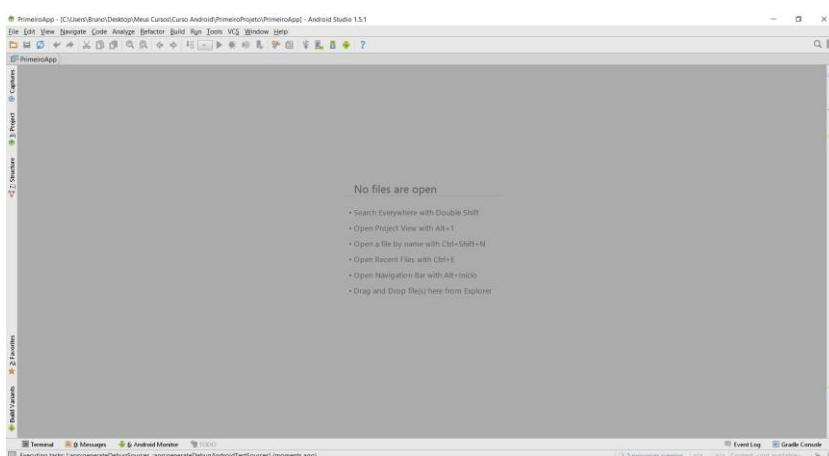
A próxima janela mostra alguns nomes, não iremos alterar, iremos aceitar a sugestão do Android Studio, então clique em “Finish” para criar o esqueleto básico da nossa primeira aplicação.



Aguarde a criação do projeto.



Quando estiver terminado será aberta a janela do principal do Android Studio.

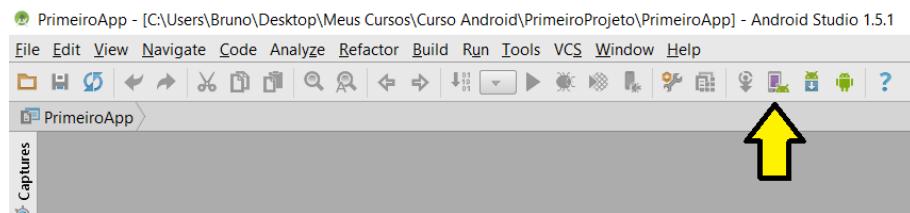


Nosso próximo passo é configurar o dispositivo virtual para os testes.

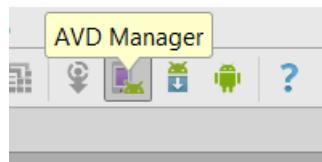


Android Virtual Device Manager

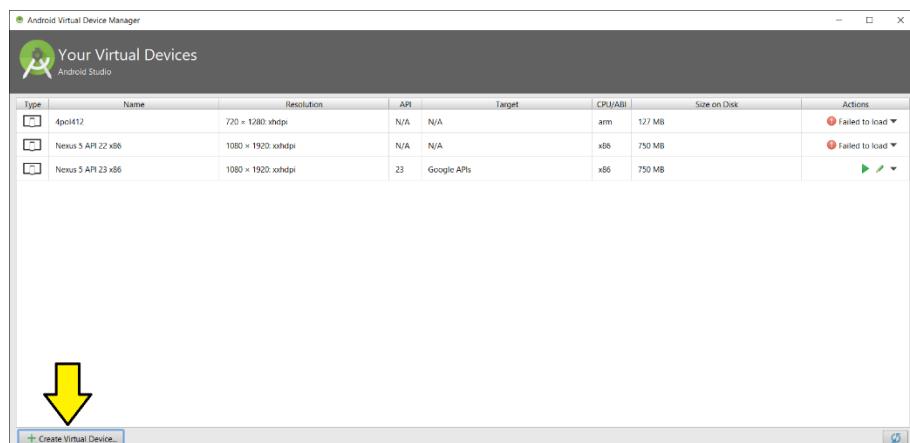
Clique no botão “AVD Manager” mostrado nas ilustrações a seguir.



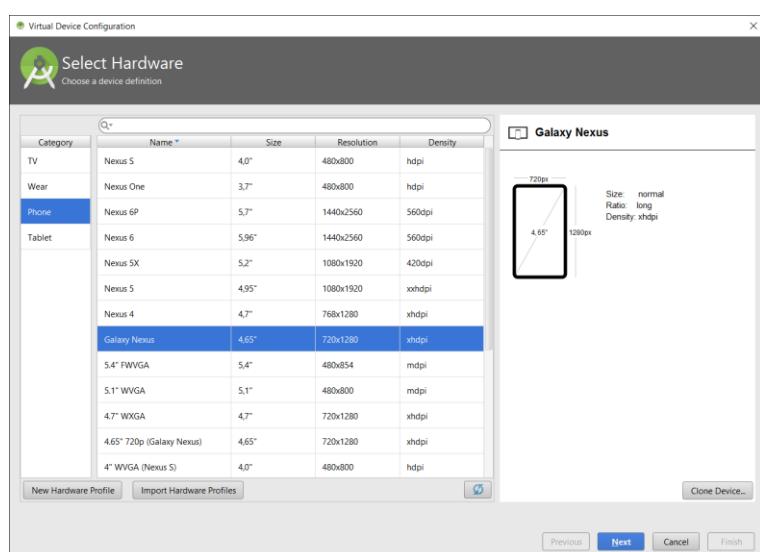
App] - Android Studio 1.5.1



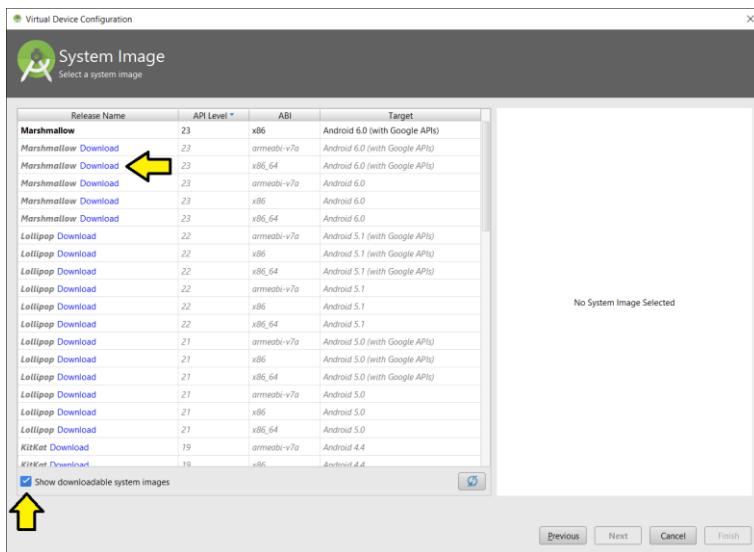
Na janela para selecionar o dispositivo clique no botão “+ Create Virtual Device”



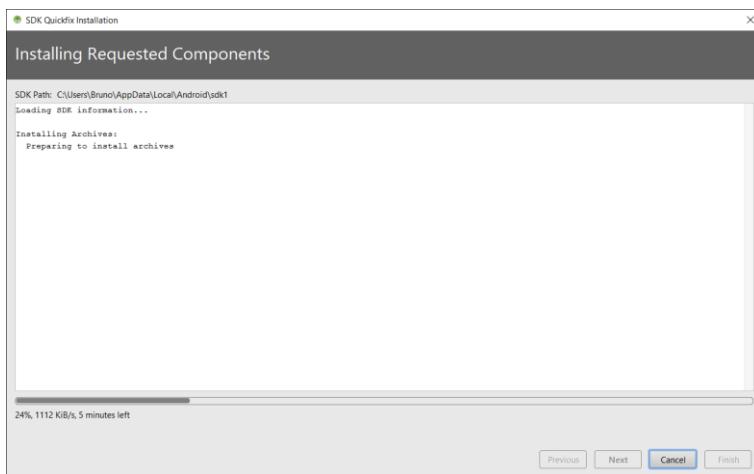
Selecione a opção “Galaxy Nexus” e clique em “Next”.



Marque a caixa “Show downloadable system images” e baixe a opção “x86_64”.



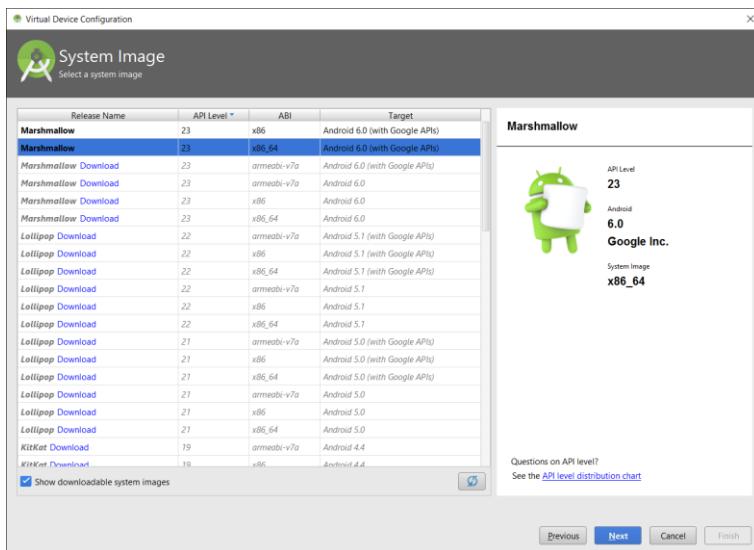
Aguarde o download.



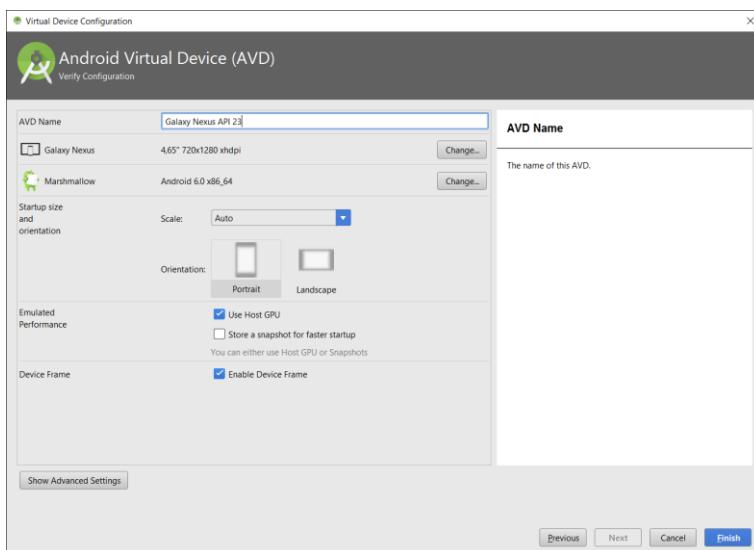
Ao terminar clique em "Finish".



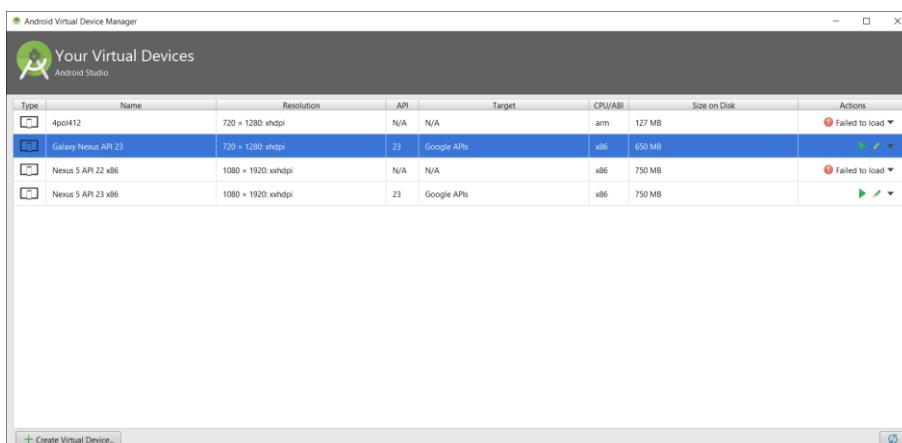
Selecione a opção que acabamos de baixar e clique em "Next".



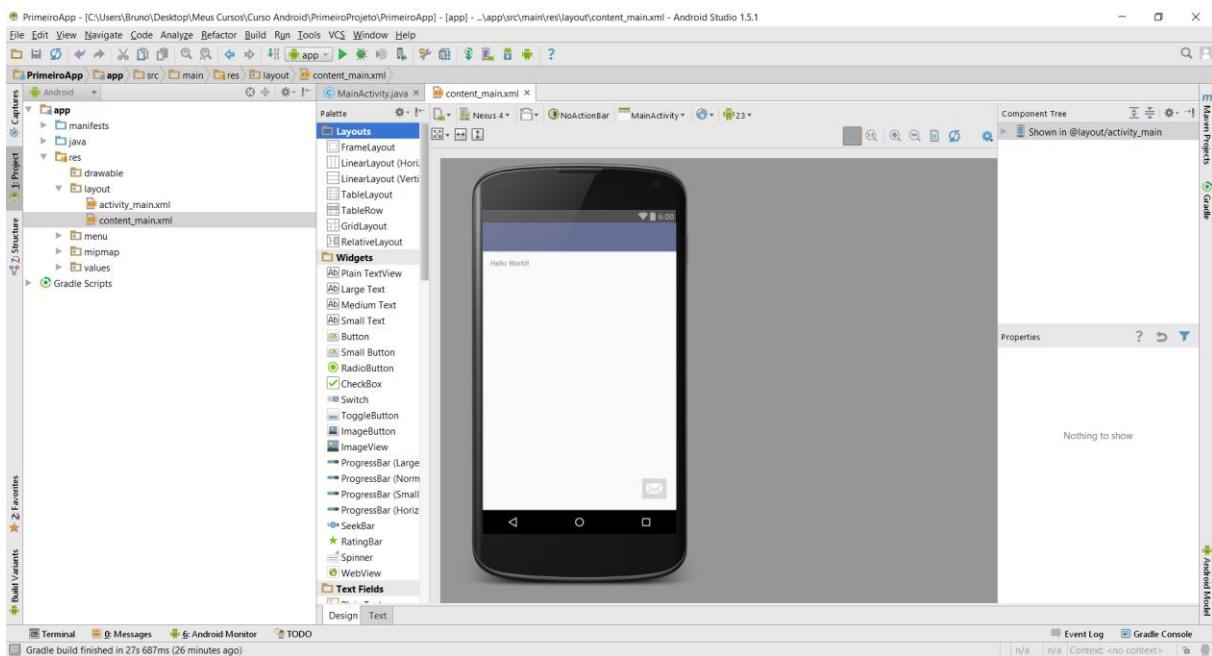
Em seguida podemos configurar algumas características do dispositivo, clique em “Finish” para terminar a adição deste dispositivo.



Caso apresente alguma erro, basta editar e selecionar a opção “x86” que já tinha disponível ao invés de “x86_64”.

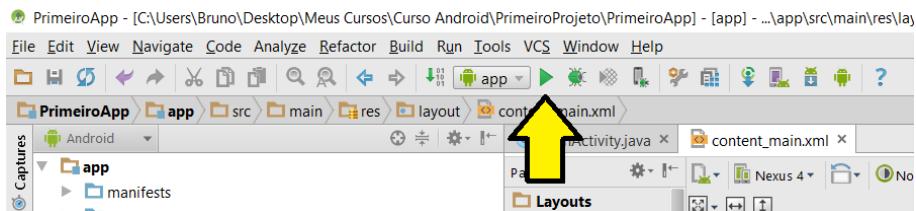


OK, basta fechar este janela, nosso dispositivo já foi adicionado e estamos prontos para criar nossa aplicação.

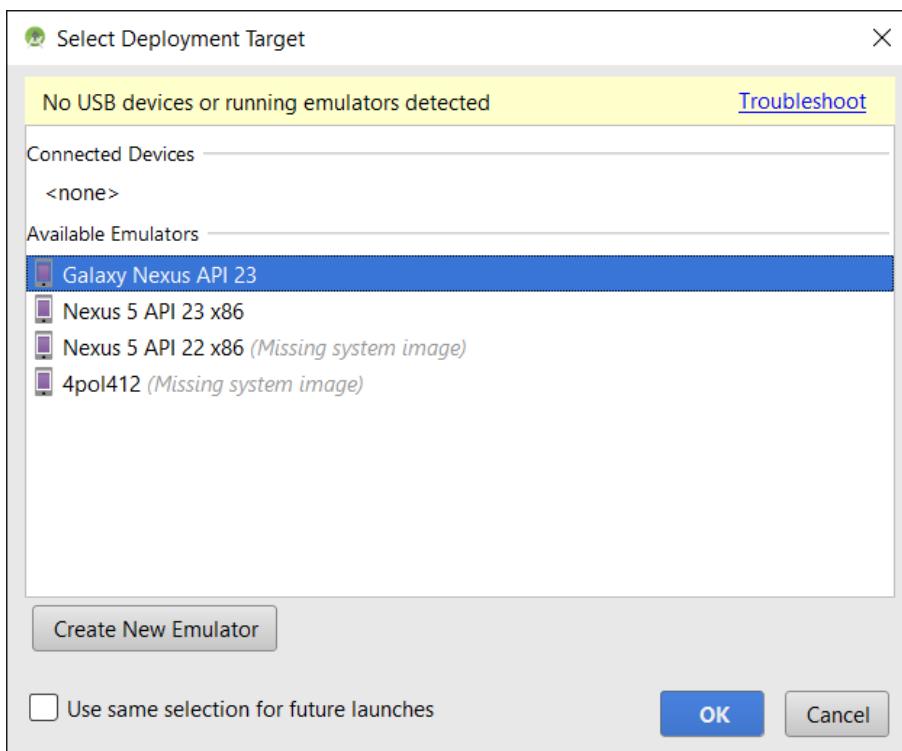


Rodando nossa aplicação

Para rodar a aplicação atual basta clicar no botão “Run App” ou pressionar “shift+F10”.



Na janela seguinte selecione o dispositivo que adicionamos “Galaxy Nexus” e clique em OK.



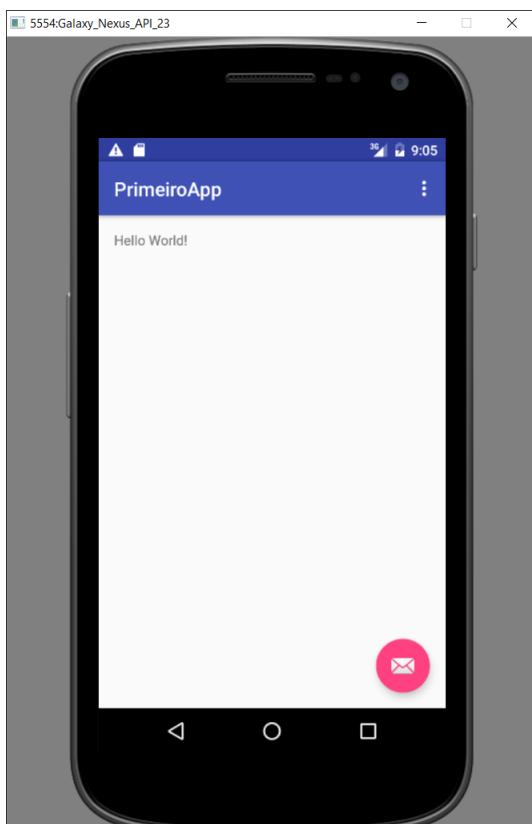


O processo é um pouco demorado, até abrir a janela com o dispositivo, tenha paciência.

O S.O. Android será carregado no dispositivo, como se estivesse ligando o telefone.



Até que o S.O. seja carregado e o aplicativo inicializado.



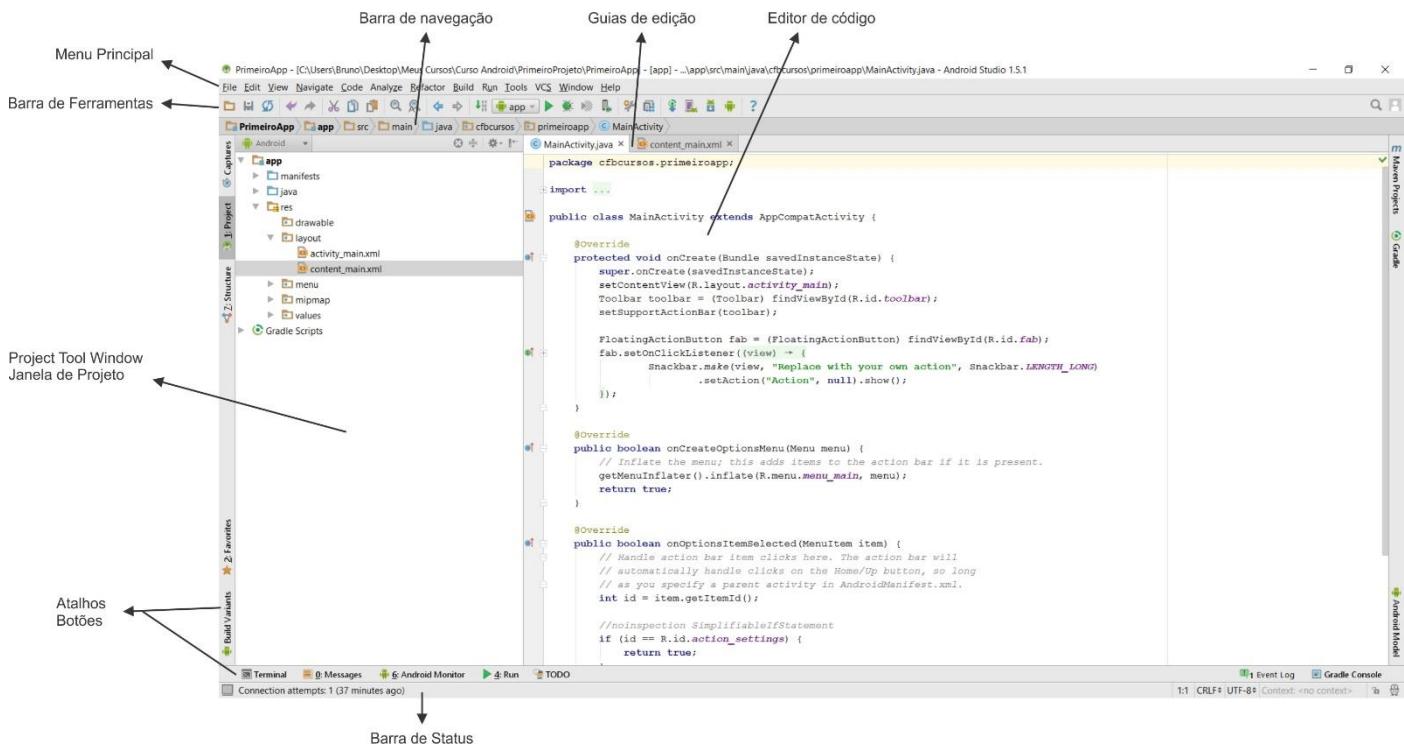


Você pode navegar pelos botões normalmente como se fosse no celular, se fechar o aplicativo é só navegar no celular e abri-lo novamente.



O ambiente do Android Studio

A primeira vista o ambiente do Android Studio parece ser bem complicado, mas aos poucos você vai se acostumando com os itens disponíveis na tela, a ilustração a seguir mostra os elementos básicos na tela do Android Studio.





Introdução sobre programação básica

Neste capítulo vamos entender um pouco sobre programação no Android Studio, a linguagem de programação usada pelo Android Studio é o Java, inclusive no início da apostila até precisamos instalar o Java SE.

Não vou aprofundar muito nas explicações de cada rotina, pois em nosso canal no Youtube (www.youtube.com.br/canalfessorbruno) temos disponível o curso de C++, para alunos que não tem familiaridade com lógica de programação, recomendo assistir estas aulas para entender como funciona a lógica dos comandos.

Portanto, vamos dedicar este capítulo à relembrar as rotinas básicas de programação, porém utilizando a linguagem Java no Android.

Quando iniciamos uma nova aplicação no Android Studio o arquivo “MainActivity.java” possui o código a seguir como padrão.

```
1 package cfbcursos.basicoprogramacao;
2
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
22         fab.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
26                     .setAction("Action", null).show();
27             }
28         });
29     }
30
31     @Override
32     public boolean onCreateOptionsMenu(Menu menu) {
33         // Inflate the menu; this adds items to the action bar if it is present.
34         getMenuInflater().inflate(R.menu.menu_main, menu);
35         return true;
36     }
37
38     @Override
39     public boolean onOptionsItemSelected(MenuItem item) {
40         // Handle action bar item clicks here. The action bar will
41         // automatically handle clicks on the Home/Up button, so long
42         // as you specify a parent activity in AndroidManifest.xml.
43         int id = item.getItemId();
44
45         //noinspection SimplifiableIfStatement
46         if (id == R.id.action_settings) {
47             return true;
48         }
49
50         return super.onOptionsItemSelected(item);
51     }
52 }
53 }
```

Na linha 1 temos o comando que determina onde nosso projeto está sendo armazenado.



Nas linhas 3 a 10, temos os imports das bibliotecas principais para a maioria dos projetos no Android.

Na linha 12 iniciamos o conteúdo desta classe, nesta linha temos a declaração desta Activity, veja que é uma classe pública, pode ser acessada por outra classe, e estende da classe “AppCompatActivity”, todas as aplicações que formos criar a classe principal irá herdar “AppCompatActivity”, por padrão.

Na linha 15 temos a declaração do método “onCreate”, mas note que na linha 14 temos um comando “anotação” @Override, o que é isto?

Vemos @Override também nas linhas, 23, 31 e 38. Quando existe a anotação @Override antes do método significa que não estamos criando um novo método e sim sobrescrevendo um método existente, existem muitos métodos já prontos para que possamos utilizar de forma a simplesmente adicionar novas funcionalidades a este método.

É o caso do “onCreate” na linha 15, não estamos declarando um método novo e sim adicionando novas funcionalidades a ele.

O método “onCreate” já possui alguns comandos básicos como definir qual é o arquivo de layout (tela) que será mostrada inicialmente, na linha 17 definimos o layout (activity_main) como o primeiro a ser carregado.

Setamos uma barra de ferramentas inicial nas linhas 18 e 19.

Nas linhas 31 a 52 temos as declarações do menu de opções.

Variáveis

Variáveis são espaços reservados na memória RAM do dispositivo para armazenamento temporário de dados, para declarar uma variável em nosso aplicativo precisamos definir o tipo de dados e nome, também podemos definir o modificador, mas isto irei falar mais adiante.

A tabela a seguir mostra alguns tipos básicos.

Tipo	Descrição
boolean	Tipo lógico, verdadeiro/true/1 ou falso/false/0
char	Tipo caractere, dados alfanuméricos.
String	Tipo texto.
byte	Inteiro de 8 bits. Valores entre -2 ⁷ =-128 e 2 ⁷ -1=127.
short	Inteiro de 16 bits. Valores entre -2 ¹⁵ =-32.768 e 2 ¹⁵ -1=32.767
int	Inteiro de 32 bits. Valores entre -2 ³¹ =-2.147.483.648 e 2 ³¹ -1=2.147.483.647.
long	Inteiro de 64 bits. Valores entre -2 ⁶³ e 2 ⁶³ -1.
float	Números reais (ponto flutuante) de 32 bits. Valores entre 1.40239846e-46 e 3.40282347e+38
double	Números reais (ponto flutuante) de 64 bits. Valores entre 4.94065645841246544e-324 e 1.7976931348623157e+308

Vamos declarar algumas variáveis globais, estas variáveis podem ser acessadas de qualquer lugar da nossa Activity.



```
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     String nome;
15     int ano;
16     float valor;
17     boolean vivo;
18     char continuar;
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_main);
24         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
25         setSupportActionBar(toolbar);
26
27         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
28         fab.setOnClickListener(new View.OnClickListener() {
29             @Override
30             public void onClick(View view) {
31                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
32                     .setAction("Action", null).show();
33             }
34         });
35     }
}
```

Vamos declarar variáveis locais dentro do método “onCreate”, neste caso estas variáveis serão de “propriedade” do método “onCreate”.

Veja pelo código de exemplo que usamos os mesmos nomes de variáveis, isso não tem problema, pois o escopo das variáveis é diferente, o que não pode acontecer é variáveis com mesmo nome no mesmo escopo.

```
12 public class MainActivity extends AppCompatActivity {
13
14     String nome;
15     int ano;
16     float valor;
17     boolean vivo;
18     char continuar;
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22
23         String nome;
24         int ano;
25         float valor;
26         boolean vivo;
27         char continuar;
28
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.activity_main);
31         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
32         setSupportActionBar(toolbar);
33
34         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
35         fab.setOnClickListener(new View.OnClickListener() {
36             @Override
37             public void onClick(View view) {
38                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
39                     .setAction("Action", null).show();
40             }
41         });
42     }
}
```



Vamos alterar as declarações das linhas 23 a 27.

```
11
12 public class MainActivity extends AppCompatActivity {
13
14     String nome;
15     int ano;
16     double valor;
17     boolean vivo;
18     char continuar;
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22
23         nome="Bruno";
24         ano=2016;
25         valor=10.500;
26         vivo=true;
27         continuar='s';
28
29         super.onCreate(savedInstanceState);
30         setContentView(R.layout.activity_main);
31         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
32         setSupportActionBar(toolbar);
33     }
34 }
```

Note que nas variáveis dentro do método “onCreate” retiramos o tipo e atribuímos valores a elas, neste caso, estas variáveis dentro do método “onCreate” são as mesmas declaradas fora, assim, simplesmente atribuímos valores às variáveis globais declaradas anteriormente.

Operadores / Operações com variáveis

Podemos realizar várias operações com variáveis usando os diversos operadores disponíveis, vamos ver os principais operadores que podemos usar.

Operadores matemáticos.

No programa a seguir mostro o uso de operações com os operadores matemáticos básicos.

```
11
12 public class MainActivity extends AppCompatActivity {
13
14     int num1,num2,res;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         num1=10;
20         num2=2;
21
22         res=num1+num2; //Soma
23         res=num1-num2; //Subtração
24         res=num1*num2; //Multiplicação
25         res=num1/num2; //Divisão
26         res=num1%num2; //MOD = Resto da divisão
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
34         fab.setOnClickListener(new View.OnClickListener() {
35             @Override
```

OBS: O valor da variável res vai sendo substituído ao longo do programa, primeiro recebe o valor 12, depois é substituído pelo valor 8, depois é substituído pelo valor 20, depois por 5 e finalmente pelo valor 0, então, o valor final da variável res é 0 (zero).

Operadores de atribuição, incremento e decremento.



São os operadores que adicionam valores às variáveis, adicionam ou subtraem valeres nas variáveis.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int num1,res;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         num1=10;
20
21         num1++; //igual a num1=num1+1
22         num1--; //igual a num1=num1-1
23         num1+=10; //igual a num1=num1+10
24         num1-=10; //igual a num1=num1-10
25         num1*=2; //igual a num1=num1*2;
26         res=num1;
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
34         fab.setOnClickListener(new View.OnClickListener() {
```

No programa anterior iniciamos atribuindo o valor 10 à variável num1.

Na linha 21 incrementamos o valor de num1 em 1, passando a valer 11, na linha 22 decrementamos o valor em 1, voltando a valer 10.

Na linha 23 incrementamos em 10 passando a valer 20 e na linha 24 decrementamos em 10 voltando a valer 10.

Na linha 25 multiplicamos o valor por 2, passando a valer 20 e na linha 26 atribuímos o valor de num1 na variável res, então, o valor de res no final é 20.

Operadores de comparação

Existem os operadores de comparação, na maioria das vezes são usados em instruções de decisão, como IF por exemplo.

Tipo	Descrição
>	Maior, verifica de um elemento é maior que outro, se o primeiro elemento do teste for maior que o segundo a comparação retorna true/verdadeiro e não for retorna false/falso.
<	Menor, verifica de um elemento é menor que outro, se o primeiro elemento do teste for menor que o segundo a comparação retorna true/verdadeiro e não for retorna false/falso.
>=	Maior ou igual, se o primeiro elemento for maior ou igual ao segundo a comparação retorna true/verdadeiro.
<=	Menor ou igual, se o primeiro elemento for menor ou igual ao segundo a comparação retorna true/verdadeiro.
==	Igualdade, retorna true/verdadeiro se os dois elementos do teste forem iguais.
!=	Diferença, retorna true/verdadeiro se os dois elementos do teste forem diferentes.

Confira alguns exemplos no código.



```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     boolean res;
15     int nota1, nota2;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19
20         nota1=35;
21         nota2=45;
22         res=nota1>nota2; //False
23         res=nota1<nota2; //True
24         res=nota1==nota2; //False
25         res=nota1!=nota2; //true
26
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_main);
29         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
30         setSupportActionBar(toolbar);
31
32         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
33         fab.setOnClickListener(new View.OnClickListener() {
34             @Override
35             public void onClick(View view) {
```

Modificadores

Os modificadores definem como, por onde e se a variável, classe ou método serão acessados/modificados.

public

O modificador public pode ser acessado de qualquer lugar do nosso aplicativo.

private

O modificador private não podem ser acessado fora de seu escopo, por outra classe por exemplo. Não se aplica às classes, somente a seus métodos e atributos (variáveis), estas variáveis e métodos também não podem ser visualizados por classes herdadas.

protected

O modificador protected torna o membro acessível às classes do mesmo pacote ou através de herança, seus membros herdados não são acessíveis a outras classes fora do pacote em que foram declarados.

final

Quando é aplicado na classe, não permite estende-la, nos métodos impede que o mesmo seja sobreescrito (overriding) na subclasse, e nos valores de variáveis não pode ser alterado depois que já tenha sido atribuído um valor.

abstract

Esse modificador não é aplicado nas variáveis, apenas nas classes. Uma classe abstrata não pode ser instanciada, ou seja, não pode ser chamada pelos seus construtores. Se houver alguma declaração de um método como abstract (abstrato), a classe também deve ser marcada como abstract.

static

É usado para a criação de uma variável que poderá ser acessada por todas as instâncias de objetos desta classe como uma variável comum, ou seja, a variável criada será a mesma em todas as instâncias e quando seu conteúdo é



modificado numa das instâncias, a modificação ocorre em todas as demais. E nas declarações de métodos ajudam no acesso direto à classe, portanto não é necessário instanciar um objeto para acessar o método.

Nas variáveis estáticas isso acontece porque todas as instâncias da mesma classe compartilham a mesma cópia das variáveis estáticas, sendo inicializadas quando a classe é carregada (instanciada).

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     public int num1;
15     private int num2;
16     public static int num3;
17     protected int num4;
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21
22         num1=10;
23         num2=10;
24         num3=10;
25         num4=10;
26
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_main);
29         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
30         setSupportActionBar(toolbar);
```

IF – Comando de decisão

O comando IF é usado para tomada de decisão, de acordo com um teste lógico que é uma comparação de valores de variáveis, o comando pode executar seu bloco de comandos ou não, veja um exemplo simples.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,result;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         nota1=35;
20         nota2=45;
21         result=nota1+nota2;
22
23         if(result >= 70){
24             //Comandos executados caso o valor
25             //da variável result seja maior ou igual a 70
26         }
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
34         fab.setOnClickListener(new View.OnClickListener() {
```

No código acima é realizado um teste condicional IF que verifica o valor da variável “result”, se seu valor for maior ou igual a 70 os comandos inseridos dentro do IF serão executados, se forem menores que 70 não entre dentro do IF, a execução do programa simplesmente irá pular o bloco de comandos IF e prosseguirá normalmente.

IF - ELSE

Um complemento ao comando IF é a instrução ELSE, que é um “se não” ou “caso contrário”, vamos ao código de exemplo que é somente uma variação do código anterior.



```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,result;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         nota1=35;
20         nota2=45;
21         result=nota1+nota2;
22
23         if(result >= 70){
24             //Comandos executados caso o valor
25             //da variável result seja maior ou igual a 70
26         }else{
27             //Comandos executados caso o valor
28             //da variável result seja menor que 70
29         }
30
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_main);
33         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
34         setSupportActionBar(toolbar);
35
36         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
37         fab.setOnClickListener(new View.OnClickListener() {
```

Neste código de exemplo se a nota for maior ou igual a 70 serão executados os comandos do bloco IF, caso contrário, se for menor que 70 serão executados os comandos do bloco ELSE.

IF – ELSE – IF

Podemos inserir quantas alternativas forem necessárias ao nosso teste, basta incrementar o ELSE com um novo teste IF.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,result;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         nota1=35;
20         nota2=45;
21         result=nota1+nota2;
22
23         if(result >= 70){
24             //Comandos executados caso o valor
25             //da variável result seja maior ou igual a 70
26         }else if(result >= 50){
27             //Comandos executados caso o valor
28             //da variável result seja maior ou igual a 50
29             //e menor que 70
30         }else {
31             //Comandos executados caso o valor
32             //da variável result seja menor que 50
33         }
34
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_main);
37         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
38         setSupportActionBar(toolbar);
```



Operador ternário / IF ternário

É uma forma mais compacta de realizar um teste IF, é mais indicado quando você tem um teste simples que precisa atribuir um valor à uma variável dependendo do valor do teste.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1, nota2, soma;
15     String resultado;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19
20         nota1=35;
21         nota2=45;
22
23         soma=nota1+nota2;
24
25         resultado=(soma >= 70 ? "Aprovado" : "Reprovado");
26
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_main);
29         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
30         setSupportActionBar(toolbar);
31
32         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
33         fab.setOnClickListener(new View.OnClickListener() {
```

O mesmo procedimento com o comando IF seria:

```
if(soma >= 70){
    resultado="Aprovado";
} else{
    resultado="Reprovado";
}
```

Switch

Outro comando para tomada de decisões é o switch, seu funcionamento é bem simples, testamos uma expressão e de acordo com o resultado desta expressão executamos um determinado bloco de comandos.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         nota=3;
20
21         switch(nota){
22             case 1:
23                 //Nota ruim
24                 break;
25             case 2:
26                 //Nota regular
27                 break;
28             case 3:
29                 //Nota Boa
30                 break;
31             default:
32                 //Nota inválida
33                 break;
34         }
35
36         super.onCreate(savedInstanceState);
37         setContentView(R.layout.activity_main);
38         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
39         setSupportActionBar(toolbar);
40
41         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
```



Switch com faixas de valores

Podemos usar o switch para testar pequenas faixas de valores, no próximo código verificados se a nota for de 0 a 3 é considerada ruim, de 4 a 7 a nota é considerada regular, de 8 a 9 é considerada boa.

```
12 public class MainActivity extends AppCompatActivity {
13
14     int nota;
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18
19         nota=7;
20
21         switch(nota){
22             case 0:
23             case 1:
24             case 2:
25             case 3:
26                 //Nota ruim
27                 break;
28             case 4:
29             case 5:
30             case 6:
31             case 7:
32                 //Nota regular
33                 break;
34             case 8:
35             case 9:
36             case 10:
37                 //Nota Boa
38                 break;
39             default:
40                 //Nota inválida
41                 break;
42         }
43
44         super.onCreate(savedInstanceState);
45         setContentView(R.layout.activity_main);
46         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
```

Comando de Loop FOR

Sempre que for necessário repetir um bloco de comandos podemos utilizar uma estrutura de loop, a primeira que iremos aprender é o loop FOR.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16
17         int i,inicia,max;
18
19         inicia=0;
20         max=10;
21
22         for(i=inicia; i<max; i++){
23             //rotina de comandos a ser repetida
24         }
25
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_main);
28         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
29         setSupportActionBar(toolbar);
```

Comando de loop while

A tradução de while é “enquanto”, então, é verificada a expressão dentro dos parênteses e enquanto essa expressão for verdadeira os comandos serão executados.



```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16
17         int i,inicia,max;
18
19         inicia=0;
20         max=10;
21
22         i=inicia;
23         while(i<max){
24             //rotina de comandos a ser repetida
25             i++; //Não podemos de esquecer o incremento dentro do while
26         }
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
```

Do while

Se diferencia do comando while, por executar os comandos pelo menos uma vez, mesmo que a condição da expressão já tenha sido satisfeita, isso porque, primeiro executa os comandos e ao final testa o while.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16
17         int i,inicia,max;
18
19         inicia=0;
20         max=10;
21
22         i=inicia;
23         do{
24             //rotina de comandos a ser repetida
25             i++; //Não podemos de esquecer o incremento dentro do while
26         }while(i<max);
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
34         fab.setOnClickListener(new View.OnClickListener() {
```



Métodos definidos pelo programador

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,soma;
15
16     void somar(){
17         soma=nota1+nota2;
18     }
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22
23         nota1=35;
24         nota2=45;
25
26         somar();
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30
31 }
```

Métodos com parâmetros de entrada

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,soma;
15
16     void somar(int n1, int n2){
17         soma=n1+n2;
18     }
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22
23         nota1=35;
24         nota2=45;
25
26         somar(nota1,nota2);
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33 }
```

Métodos com retorno

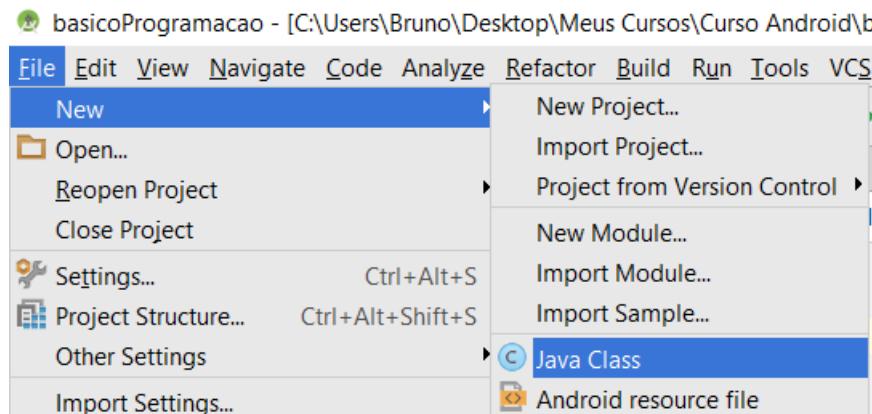
```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,soma;
15
16     int somar(int n1, int n2){
17         return n1+n2;
18     }
19
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22
23         nota1=35;
24         nota2=45;
25
26         soma = somar(nota1,nota2);
27
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
31         setSupportActionBar(toolbar);
32
33 }
```



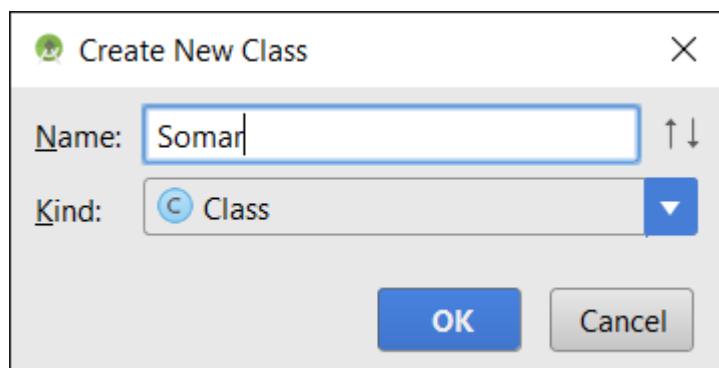
Criando novas classes

Iremos criar uma nova classe bem simples, com um método para somar dois valores e retornar o resultado desta soma.

Para criar uma nova classe, clique no menu FILE – NEW – Java Class.



Digite o nome da classe, em nosso exemplo será “Somar”, clique em OK.



A nova classe será criada, crie o novo método “soma2” de acordo com a ilustração a seguir.

```
package cfbcursos.basicopogramacao;

/**
 * Created by Bruno on 30/03/2016.
 */
public class Somar {

    public int soma2(int n1, int n2) {
        return n1+n2;
    }
}
```

The screenshot shows the Android Studio code editor with three tabs at the top: 'MainActivity.java', 'Somar.java', and 'content_main.xml'. The 'Somar.java' tab is active. The code editor displays the following Java code:

```
package cfbcursos.basicopogramacao;

/**
 * Created by Bruno on 30/03/2016.
 */
public class Somar {

    public int soma2(int n1, int n2) {
        return n1+n2;
    }
}
```

The code is syntax-highlighted, with 'cfbcursos.basicopogramacao' in blue, package, class, and method names in black, and comments in green. The 'soma2' method is highlighted with a red rectangular box.

Veja o método irá receber dois parâmetros do tipo “int” e irá retornar a soma destes dois “int”.

Agora vamos usar nossa classe soma, iremos declarar um objeto desta classe com nome “s”.



```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     int nota1,nota2,resultado;
15
16     Somar s=new Somar();
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20
21         nota1=35;
22         nota2=45;
23
24         resultado=s.soma2(nota1,nota2);
25
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_main);
28         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
29         setSupportActionBar(toolbar);
```

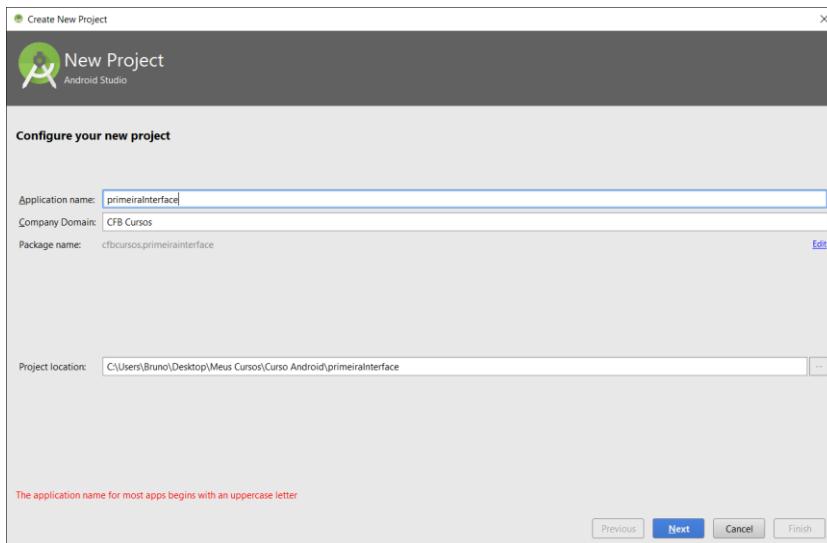
Note que na linha 24 simplesmente chamamos o método “soma2” o objeto “s”.

Construindo a Interface com o usuário

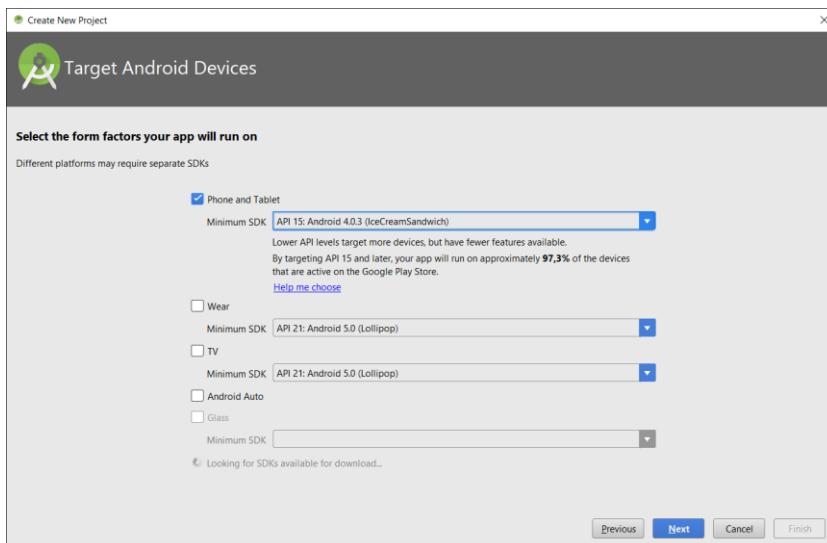
Vamos iniciar um novo projeto no Android Studio.

Clique no menu FILE – NEW – NEW PROJECT

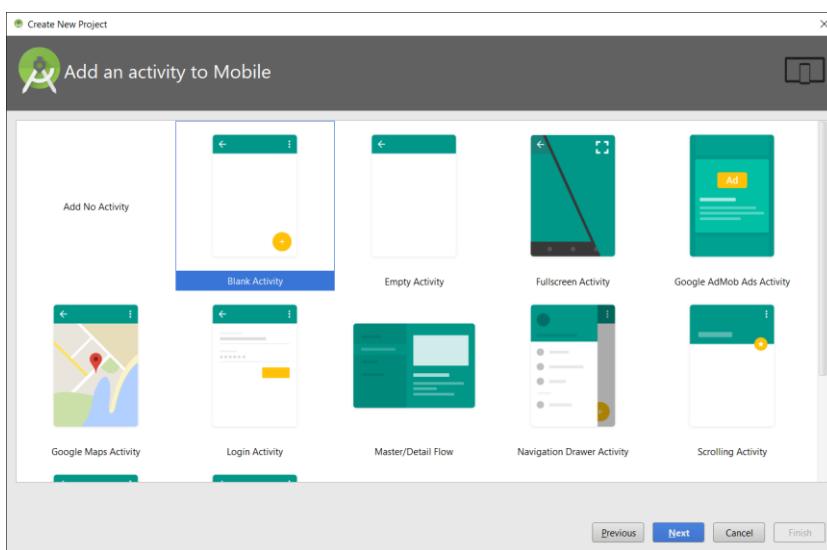
Configure o nome do novo projeto para “primeiralInterface”, indique o local de armazenamento do projeto e clique no botão “Next”.



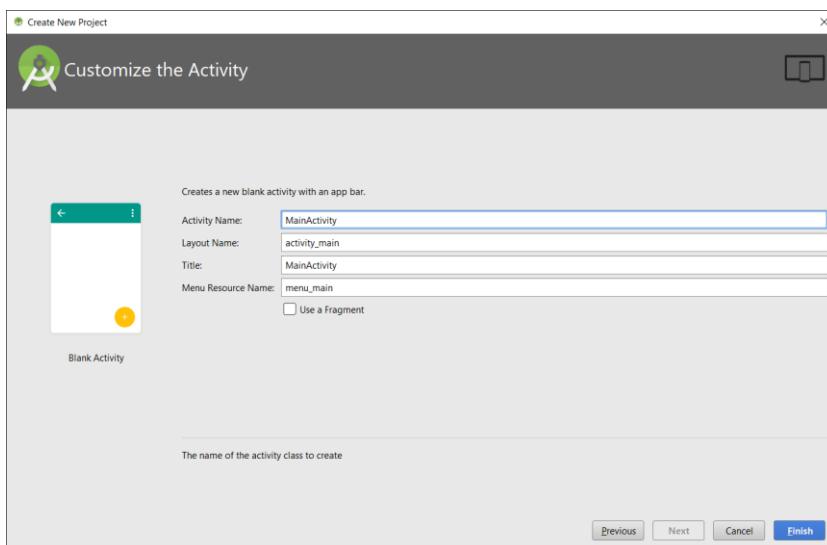
Configure o projeto para telefones e tablets e clique em “Next”.



Selecione “Black Activity” e clique em “Next”.



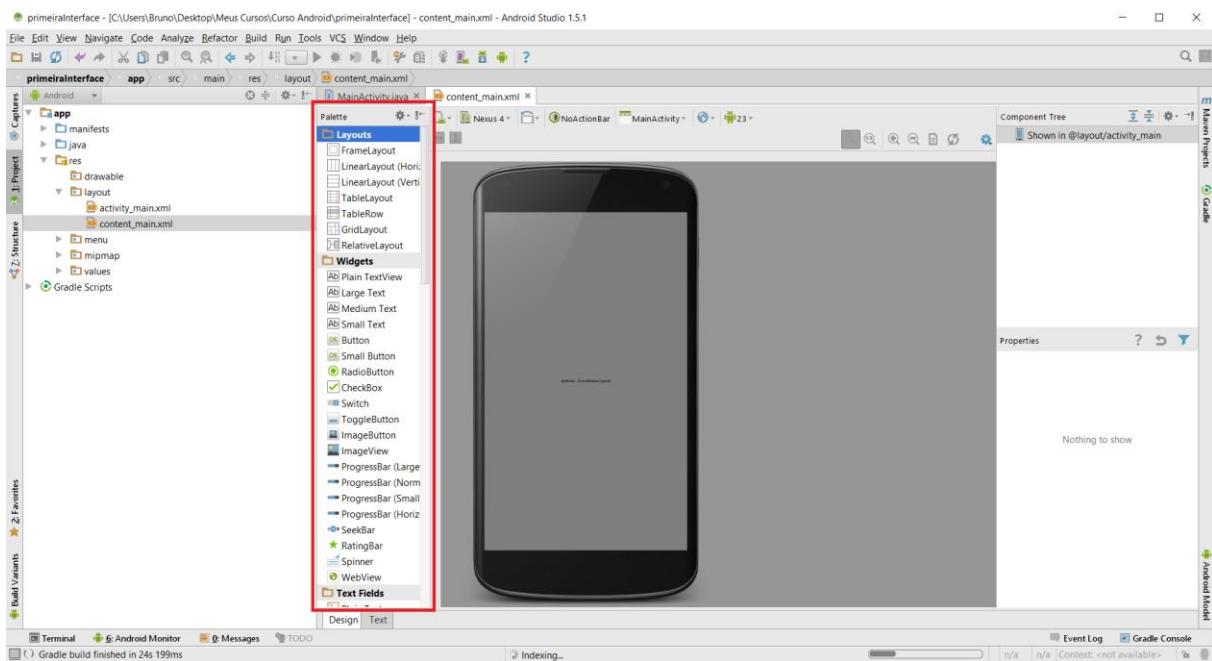
E por fim clique em “Finish”.



Aguarde a preparação do novo projeto.



Aqui está a tela pronta para o trabalho, note que destaquei com um retângulo vermelho a paleta de componentes para interface.

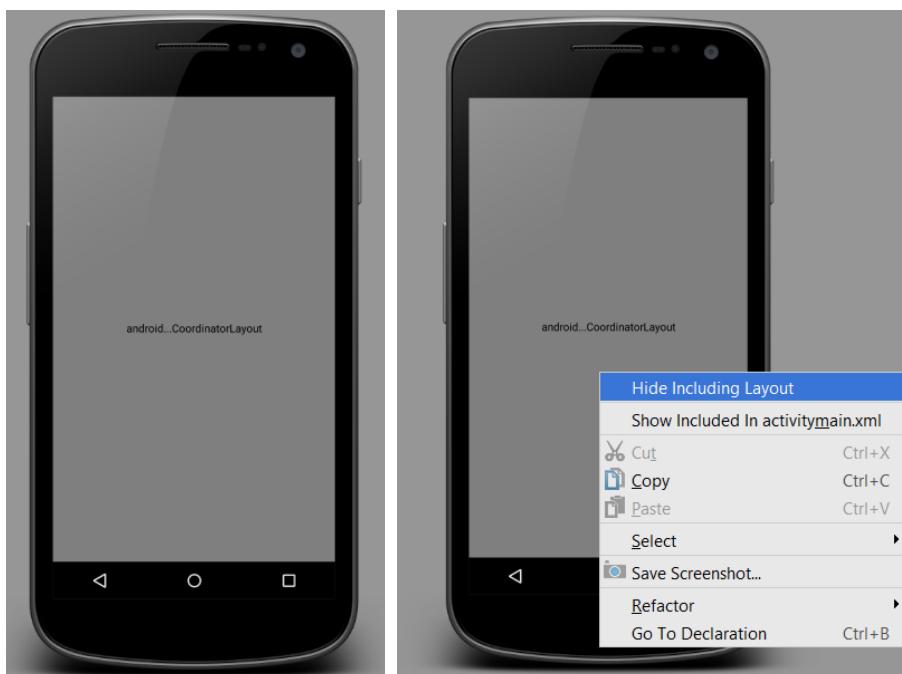


Nesta paleta estão todos os componentes que poderão ser usados para construir nossa interface.

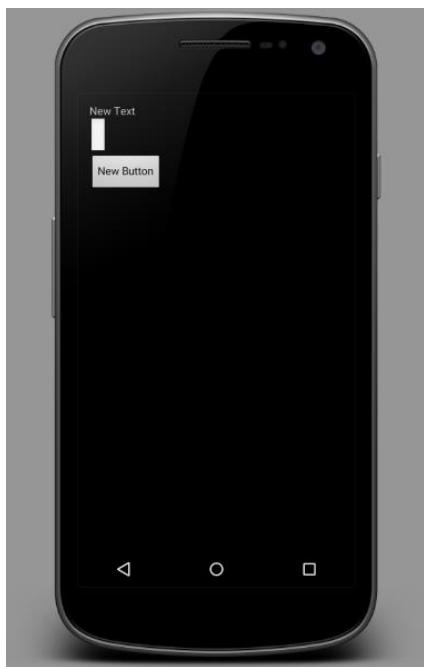
Podemos construir a interface da nossa aplicação clicando e arrastando para a tela do celular ao lado, ou editando diretamente o arquivo XML.

Primeiramente vamos ver como usar o modo gráfico, clicando e arrastando.

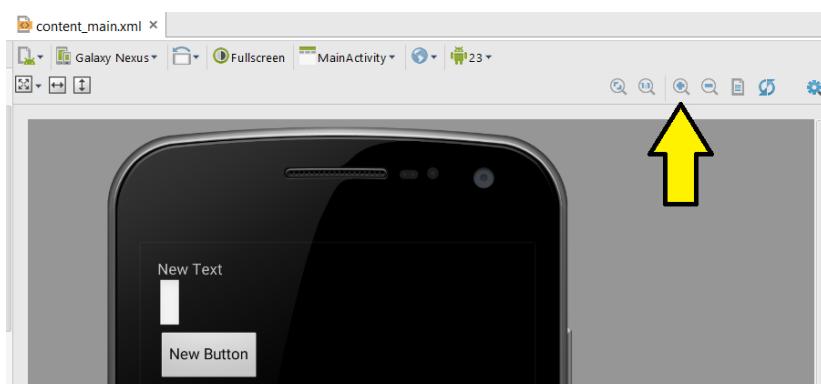
Se o modelo do celular estiver sendo mostrado com a tela cinza, como na ilustração a seguir, clique com o botão direito do mouse e selecione “Hide Including Layout”.



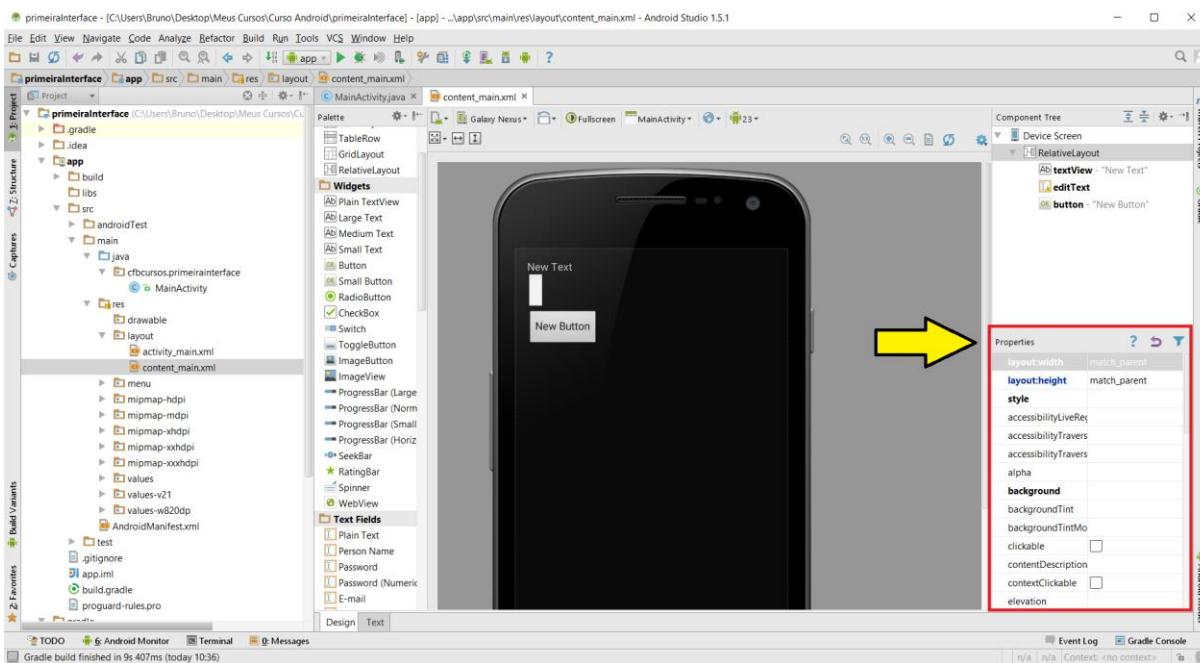
Agora vamos arrastar os componentes Plain TextView, Plain Text e Button.



Podemos usar o botão (Zoom In +) para aumentar o zoom.



Para configurar os elementos inseridos em nosso aplicativo, usamos o painel “properties/Propriedades” em destaque na ilustração a seguir.



Neste painel podemos formatar todas as propriedades dos elementos, como ID, tamanho, cor de fundo, etc.

Vamos configurar os elementos de acordo com as referências a seguir.

Plain TextView

Propriedade	Valor
text	Digite seu nome:
textSize	20dp

Plain Text

Propriedade	Valor
width	300dp

Button

Propriedade	Valor
background	#1179ce
height	40dp
textColor	#ffffff
width	150dp

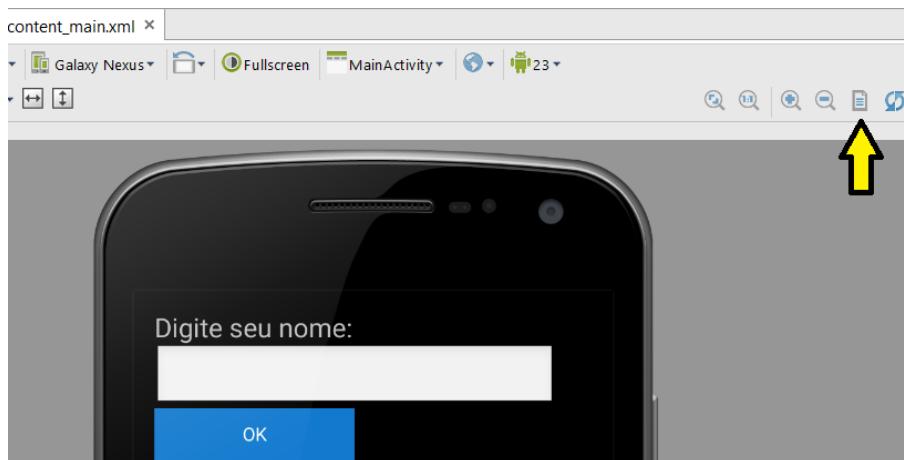
Confira a seguir como ficam os componentes depois das alterações.



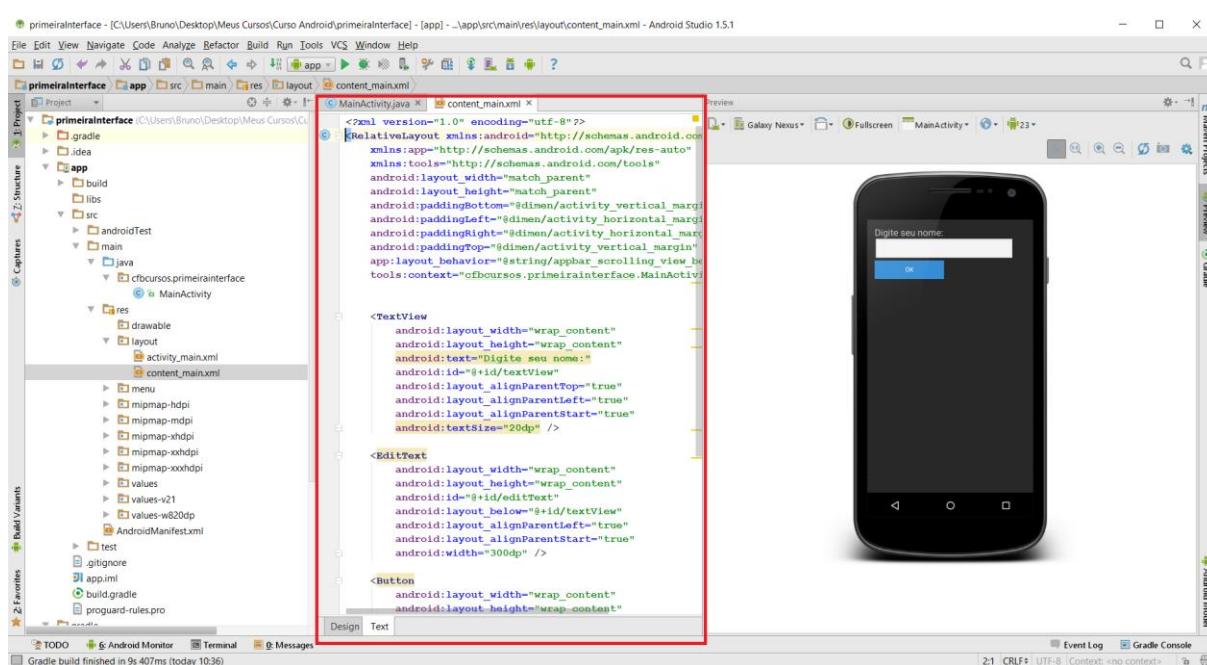
Sempre que adicionamos e configuramos um componente em nosso aplicativo o arquivo content_main.xml é alterado recebendo o código deste novo componente.

Podemos abrir o arquivo content_main.xml para editar ou inserir os componentes manualmente, vamos abrir o arquivo para conferir os códigos referentes aos componentes que adicionamos.

Clique no botão “Jump to Source” apontado na ilustração a seguir.



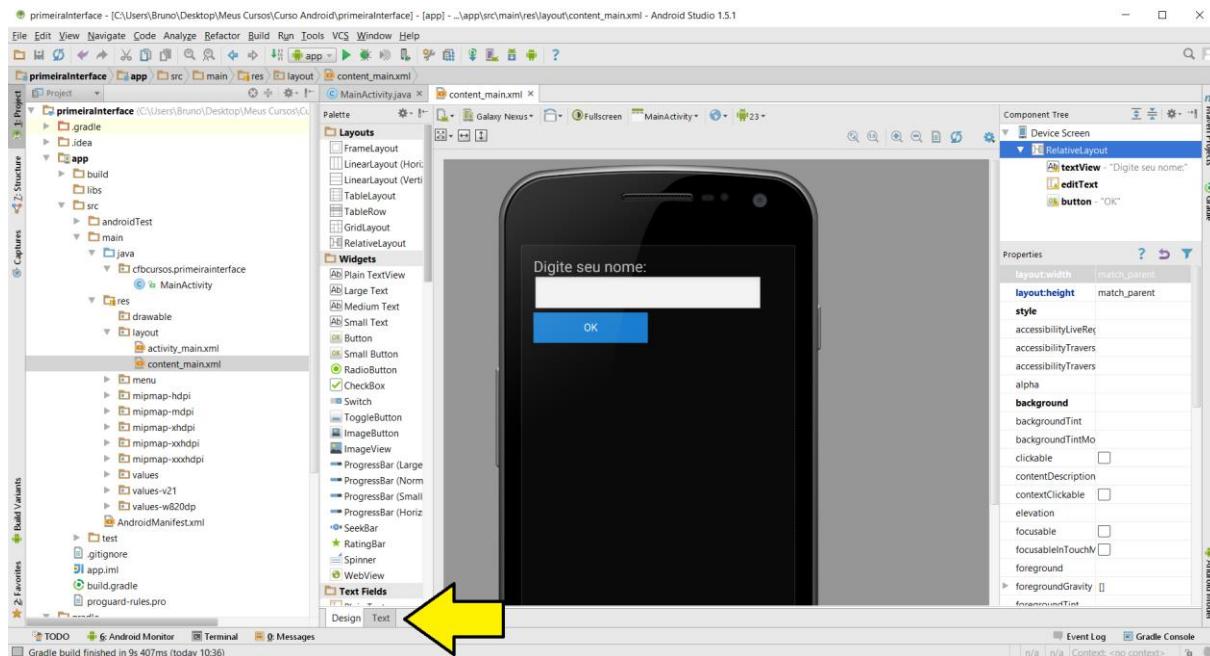
O código XML será exibido conforme a ilustração a seguir, observe os comandos referentes aos componentes.



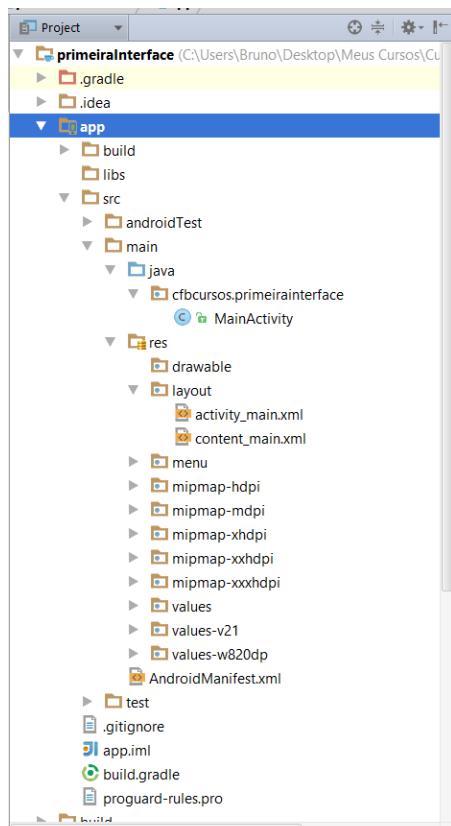


Para voltar na exibição da paleta dos componentes, basta aplicar um clique duplo da imagem do celular.

Outra maneira de alternar entre a exibição do código e dos componentes é usar as abas mostradas na ilustração a seguir.



A estrutura de um aplicativo no Android Studio



Nosso aplicativo fica organizado em uma estrutura de árvore no Android Studio, e devemos aprender alguns elementos importantes desta estrutura.

Dentro da pasta “app” encontramos toda a estrutura da nossa aplicação, vejamos as que nos interessa por enquanto.

src → Esta é a pasta “source” onde ficam todos os arquivos da aplicação.

res – values → Armazena alguns valores estáticos referentes aos elementos do nosso projeto, por exemplo o nome da aplicação no arquivo “strings.xml”

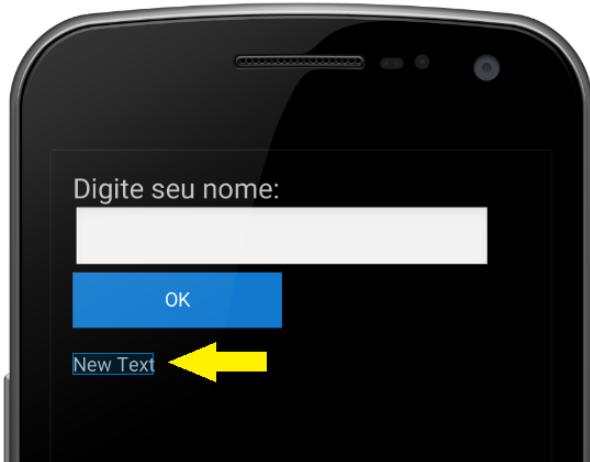
```
<resources>
    <string name="app_name">primeiraInterface</string>
    <string name="action_settings">Settings</string>
</resources>
```

O arquivo strings.xml podemos modificar, por exemplo, vamos inserir uma nova variável chamada “cfb”.

```
<resources>
    <string name="app_name">primeiraInterface</string>
    <string name="cfb">Canal Fessor Bruno</string>
    <string name="action_settings">Settings</string>
</resources>
```

res – layout → No arquivo “content_main.xml” estão todas as configurações dos componentes adicionados em nosso aplicativo.

Vamos adicionar mais um elemento “Plain TextView” em nosso aplicativo.



Agora no arquivo “content_main.xml” podemos observar o código referente a este componente.

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="New Text"  
    android:id="@+id/textView2"  
    android:layout_below="@+id/button"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:layout_marginTop="18dp" />  
  
</RelativeLayout>
```

Vamos alterar a propriedade “text” para usar o valor da variável “cfb” que adicionamos no arquivo “strings.xml”, observe a alteração no arquivo “content_main.xml” e o resultado na ilustração do celular.

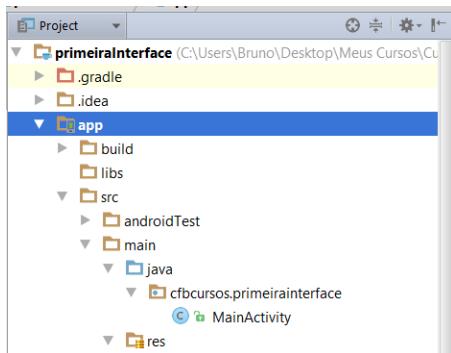
```
    android:text="@string/cfb"
```

```
    android:id="@+id/editText"  
    android:layout_below="@+id/textView"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:width="300dp" />  
  
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="OK"  
    android:id="@+id/button"  
    android:layout_below="@+id/editText"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:background="#1179ce"  
    android:height="40dp"  
    android:textColor="#ffffffff"  
    android:width="150dp" />  
  
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/cfb"  
    android:id="@+id/textView2"  
    android:layout_below="@+id/button"  
    android:layout_alignParentLeft="true"  
    android:layout_alignParentStart="true"  
    android:layout_marginTop="18dp" />
```

res-drawable → Armazena as imagens que serão usadas em nosso aplicativo.



O arquivo “MainActivity.java” contém o código principal da nossa aplicação, tudo que nosso aplicativo irá fazer estará codificado aqui, você pode ver pela ilustração a seguir que se encontra na pasta app -> src -> main -> java -> cfbcursos.primeirainterface.



Aplicando um clique duplo neste arquivo podemos ver todo o código fonte “Java” deste arquivo.

Vamos a algumas informações básicas importantes.

1) Toda aplicação será um derivado da classe “AppCompatActivity”, por isso a classe “MainActivity” estende da classe “AppCompatActivity”.

```
public class MainActivity extends AppCompatActivity {
```

2) R.java, esta classe NÃO DEVE SER MODIFICADA manualmente, esta classe é usada para atualizar o que for feito em nosso projeto, o caminho do arquivo R.java em nosso aplicação é app -> build -> generated -> source -> r -> debug -> cfbcursos.primeirainterface.

No topo do arquivo R.java encontramos esta mensagem.

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */
```

“Esta classe foi gerado automaticamente pela ferramenta aapt a partir dos dados de recursos que encontrou. Não deve ser modificado com a mão.”

Neste arquivo R.java vamos encontrar as definições básicas dos elementos que iremos usar em nosso programa.

Observe que o arquivo “AppCompatActivity.java” encontramos vários usos da classe R.java como parâmetro em funções, a ilustração a seguir mostra alguns em destaque.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener(OnClickListener) (view) &gt;
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
});
```



R.layout -> Faz referência à classe layout dentro da classe R que guarda informações do layout principal da aplicação.

R.id -> Faz referência à classe id dentro da classe R que dispõe todos os ids dos elementos usados em nossa aplicação.

No arquivo “AndroidManifest.xml” temos definições importantes sobre nosso aplicativo, informações como ícone, sobre o arquivo arquivo principal “MainActivity.java”, etc.

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

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

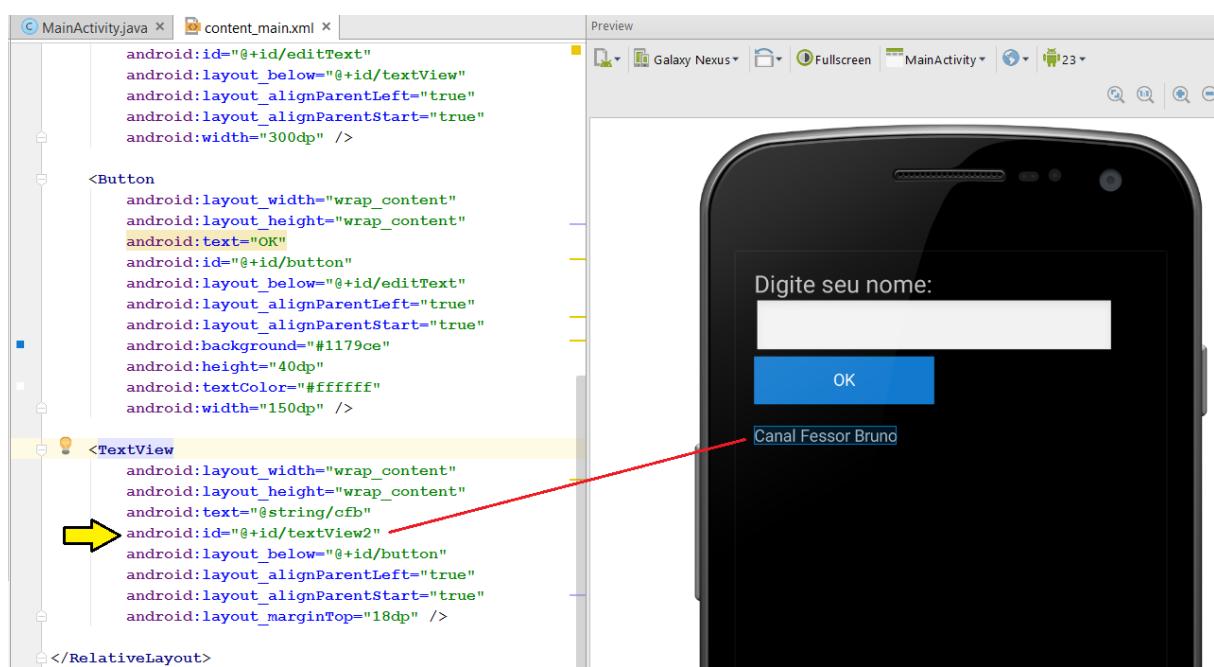
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Adicionando comando ao botão

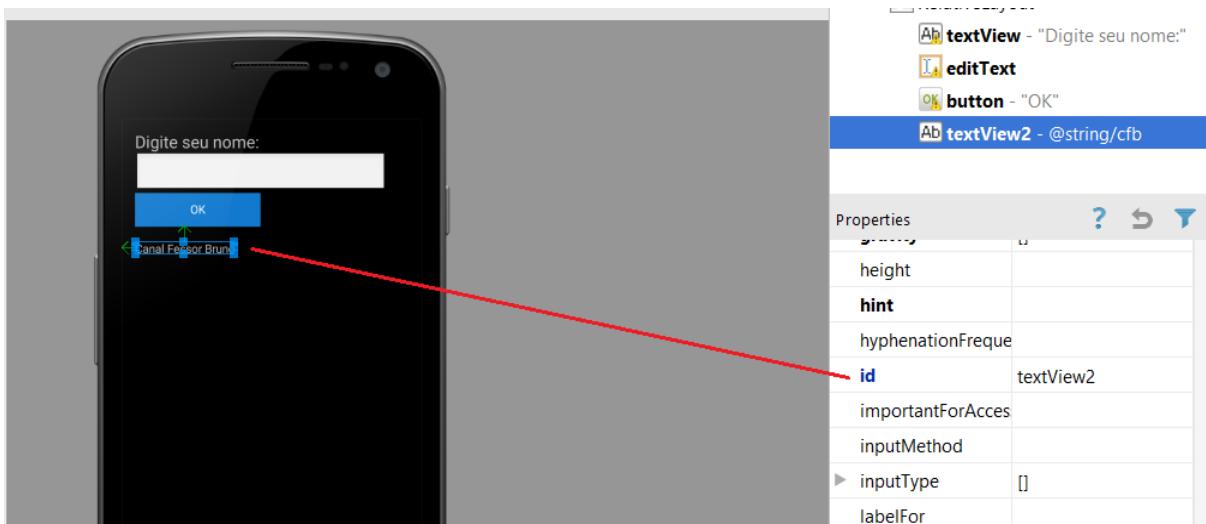
Em nossa aplicação nós adicionamos um botão, vamos adicionar um comando a este botão que pegará o texto inserido na caixa de texto acima do botão e irá adicionar este texto no componente de texto abaixo do botão.

Antes, observe que o ID do componente TextView que iremos alterar o “text” é “textView2”.





Podemos observar também pelo painel de propriedades.



Vamos ao código.

No arquivo “MainActivity.java” adicione o código destacado com a caixa vermelha a seguir.

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });

        //Programação do clique do botão
        Button b1=(Button)findViewById(R.id.button);

        b1.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v){
                TextView txtv2=(TextView)findViewById(R.id.textView2);
                EditText et=(EditText)findViewById(R.id.editText);
                txtv2.setText(et.getText());
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
    }
}
```

Vou explicar de forma simples estes comandos.

Primeiramente declaramos um objeto do tipo “Button” com nome “b1” e adicionamos neste objeto o botão do nosso layout, usamos a função “findViewById” para localizar o componente pelo seu ID, como não alteramos o ID do nosso botão, está usando o ID padrão “button”.



```
Button b1=(Button) findViewById(R.id.button);
```

O segundo passo foi definir o listener para o evento de clique.

```
b1.setOnClickListener(new View.OnClickListener() {
```

Dentro deste listener criamos um método de clique chamado “onClick” onde vamos inserir os comandos do clique.

```
public void onClick(View v) {
```

Em seguida inserimos os comandos que serão executados quando o botão for clicado.

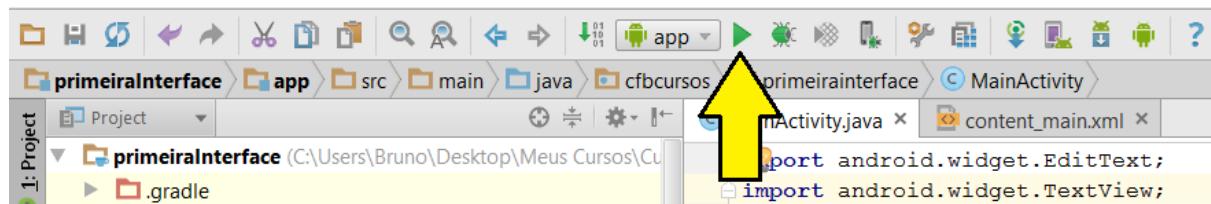
```
TextView txtv2=(TextView) findViewById(R.id.textView2);
EditText et=(EditText) findViewById(R.id.editText);
txtv2.setText(et.getText());
```

Criamos dois objetos um do tipo TextView com nome txtv2 e um do tipo EditText com nome et, adicionamos os respectivos elementos nestes objetos.

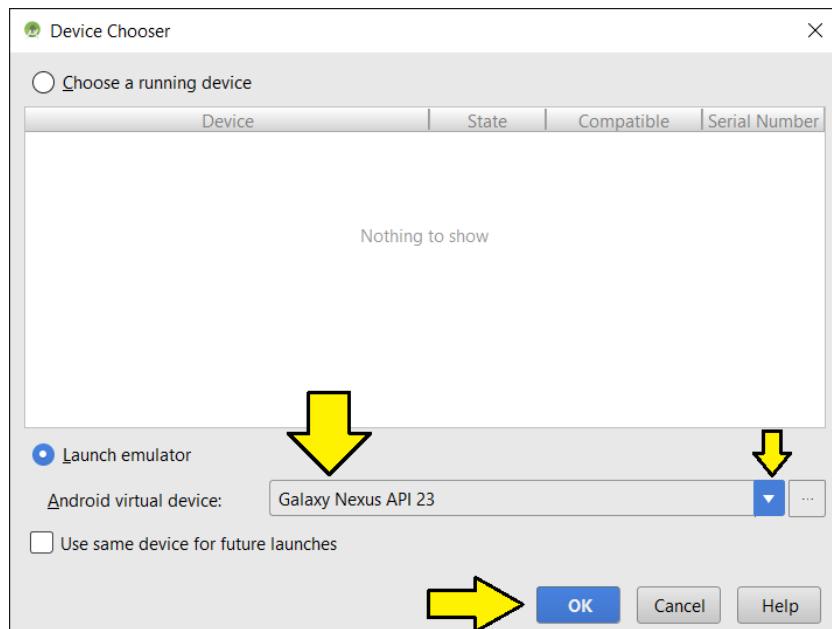
O último comando chama a função “setText” do objeto “txtv2” para definir seu texto e como parâmetro desta função recebe o retorno da função “getText” do objeto et, que retorna o texto do elemento.

Então: “getText” retorna o valor da propriedade “text” do componente e “setText” define o valor da propriedade “text” do componente.

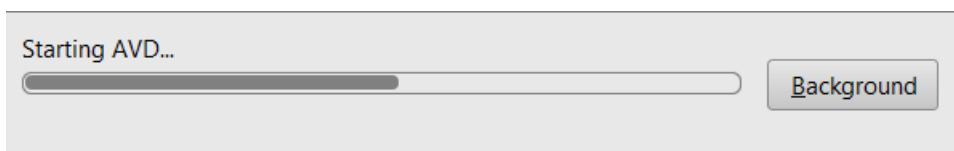
É hora de salvar tudo e testar nossa aplicação, clique no botão “Run app” na barra de ferramentas.



Selecione o dispositivo que definimos anteriormente “Galaxy Nexus” e clique em OK.

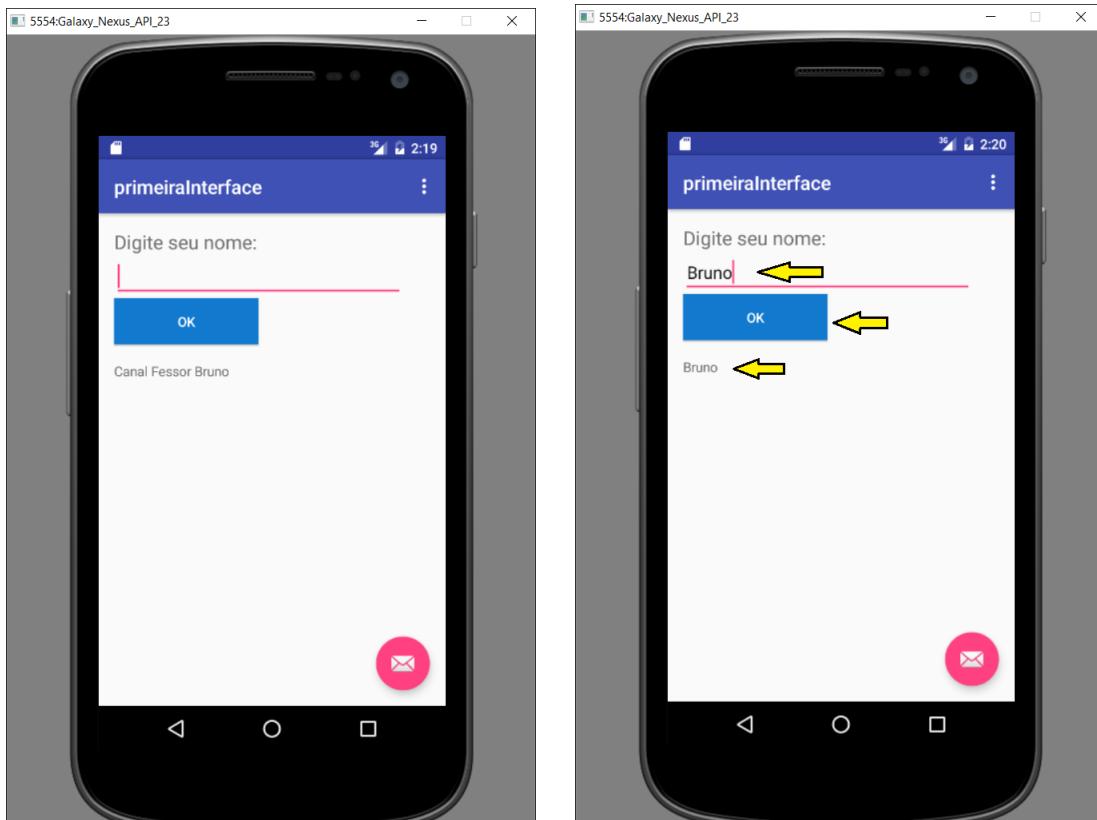


Agora vamos aguardar até que o emulador seja aberto e tenha nossa aplicação carregada, vai demorar...



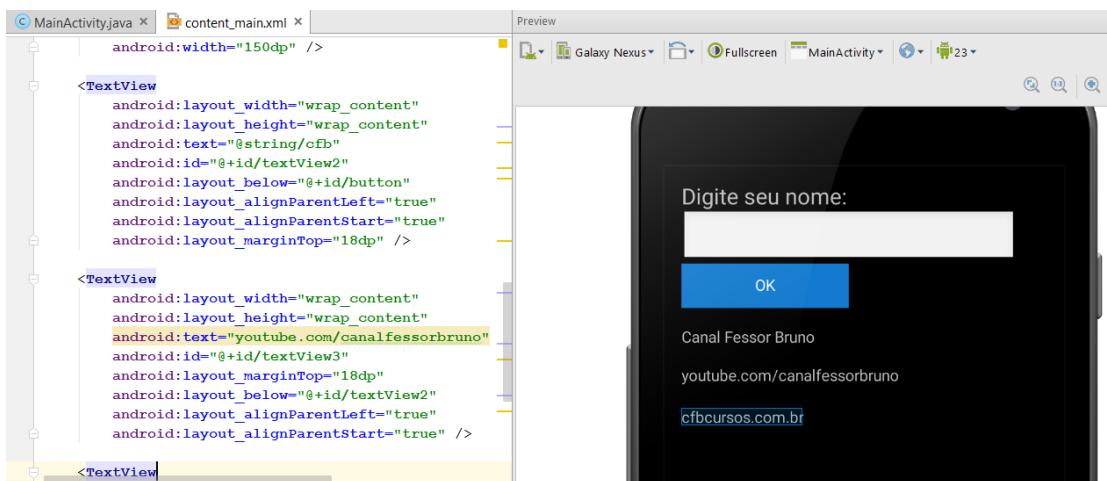
Caso abra o emulador e não abra a aplicação automaticamente, basta você navegar pelo Android no simulador do telefone e abrir manualmente a aplicação.

Digite seu nome na caixa de texto e clique no botão.



NÃO FECHE O EMULADOR, vamos realizar uma pequena alteração em nosso aplicativo, mas sem fechar o emulador do Android.

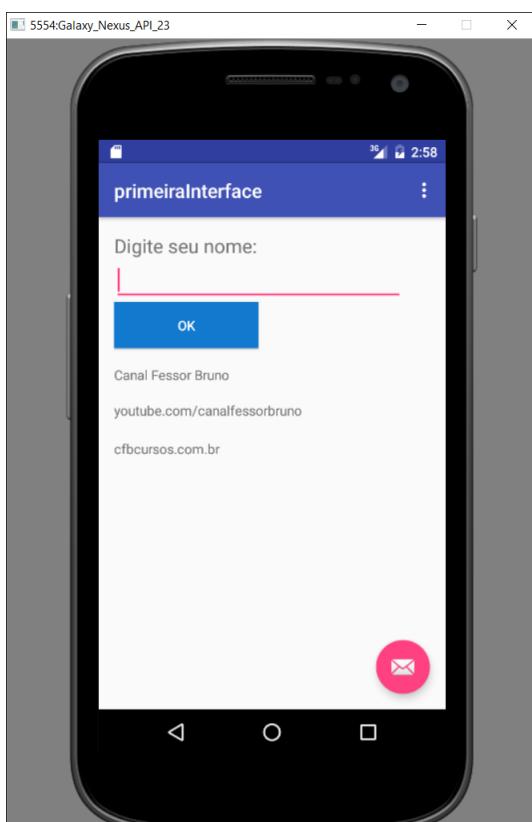
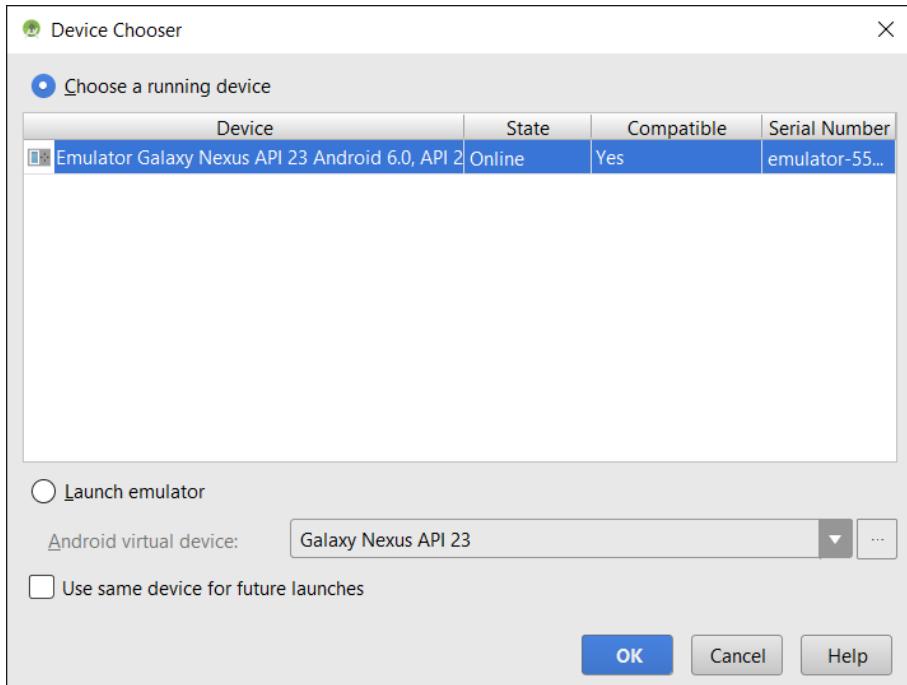
Vamos adicionar mais dois componentes “Plain TextView” conforme a ilustração a seguir.





Salve as alterações e clique no botão “Run app” novamente, veja que como o emulador já está aberto o processo será bem mais rápido.

Ao clicar novamente no botão “Run app” será mostrada a janela para selecionar o dispositivo, neste caso como nosso emulador já está aberto com o “Galaxy Nexus” iremos selecioná-lo e aplicação será atualizada no emulador.



Viu como o processo é mais rápido, então, não precisamos fechar o emulador todas vezes.

SEM FECHAR O EMULADOR.



Vamos adicionar estas informações nas variáveis globais de string, isto irá facilitar sempre que for preciso utilizar estas informações.

Lembra-se do arquivo “strings.xml” que está na pasta “values”? Abra este arquivo e vamos criar as variáveis.

Adicione as variáveis destacadas na ilustração a seguir.

```
<resources>
    <string name="app_name">primeiraInterface</string>
    <string name="cfb">Canal Fessor Bruno</string>
    <string name="sitecfb">cfbcursos.com.br</string>
    <string name="canalcfb">youtube.com/canalfessorbruno</string>
    <string name="action_settings">Settings</string>
</resources>
```

Salve estas alterações e vamos utilizá-las em nosso layout no arquivo “content_main.xml”.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/canalcfb" <-- Yellow arrow here
    android:id="@+id/textView3"
    android:layout_marginTop="18dp"
    android:layout_below="@+id/textView2"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/sitecfb" <-- Yellow arrow here
    android:id="@+id/textView4"
    android:layout_marginTop="21dp"
    android:layout_below="@+id/textView3"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />

</RelativeLayout>
```

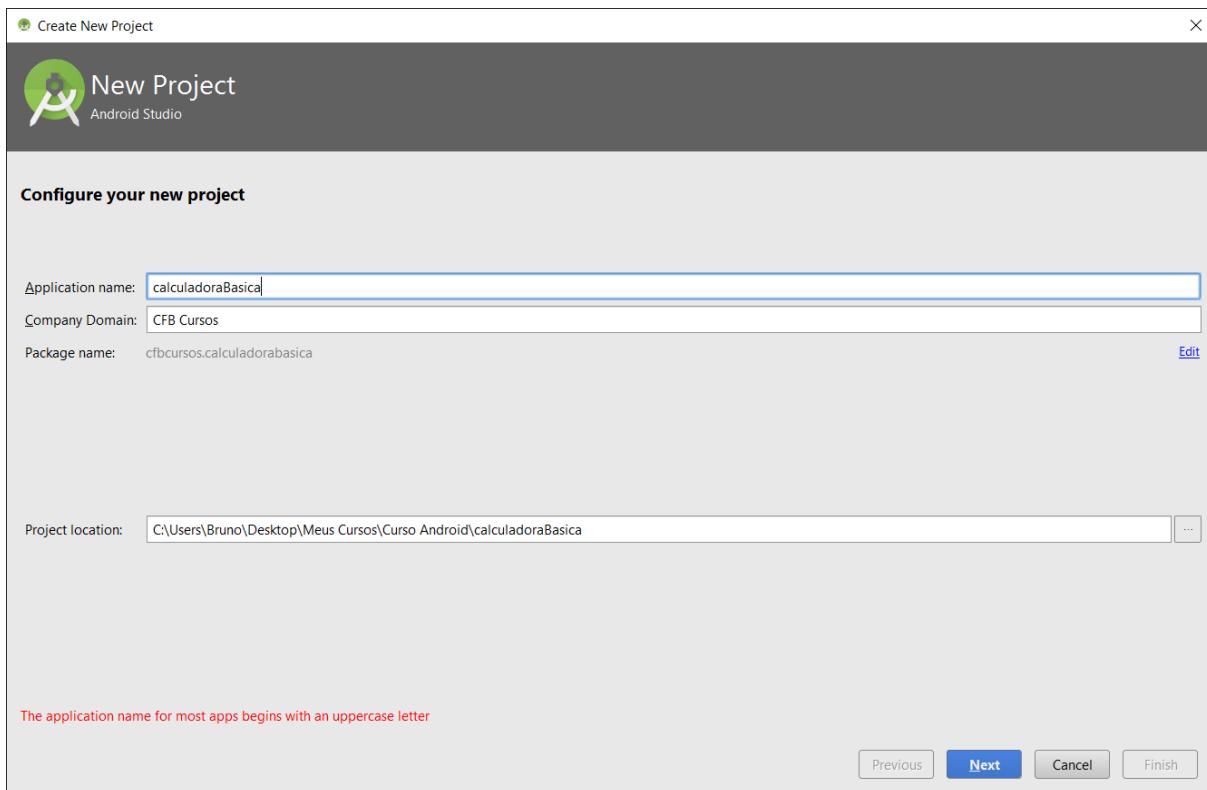
Salte todas as alterações e rode novamente a aplicação, mas não veremos alterações no resultado final, pois os textos são os mesmos.

Vamos finalizar aqui nossa primeira aplicação, salve todas as alterações e clique no menu FILE – CLOSE PROJECT.

Segundo aplicativo – Calculadora básica

Vamos iniciar um projeto um pouco mais complexo que o anterior, vamos criar uma calculadora básica, desta maneira vamos poder aprender e praticar um pouco mais sobre as técnicas de programação no Android.

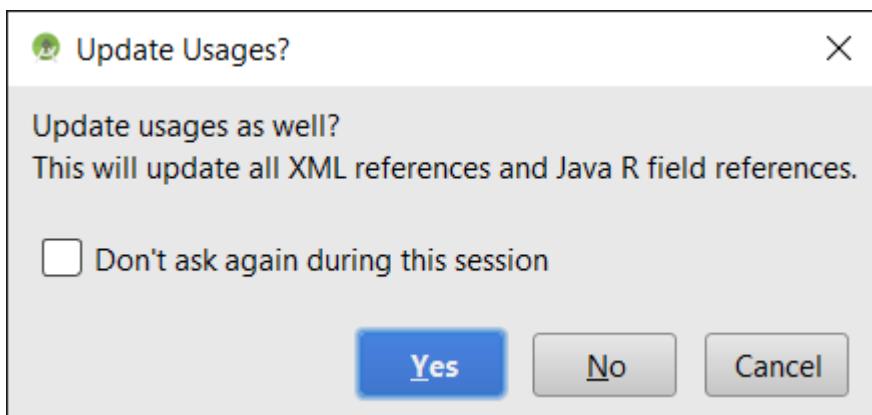
Nosso novo projeto terá o nome de “calculadoraBasica”, conforme a ilustração a seguir.



Inicie o novo projeto com uma “Black Activity” conforme já aprendemos.

Vamos preparar a interface conforme os elementos a seguir, insira um abaixo do outro e aplique as configurações mostradas nas tabelas para cada componente.

Sempre que você alterar a propriedade “id” a caixa de mensagem será exibida, informando que o arquivo XML será atualizado, marque a caixa de mensagem “Don’t ask again during this session” e clique em Yes, isso fará com que a caixa não seja mais mostrada.



Plain TextView

Propriedade	Valor
id	txtv1
text	Calculadora Básica - CFB

Plain TextView

Propriedade	Valor
-------------	-------



id	txtv2
text	Digite o Primeiro Número

Plain Text

Propriedade	Valor
id	et_num1
width	200dp
numeric	Marque: integer, signed e decimal

Plain TextView

Propriedade	Valor
id	txtv3
text	Digite o Segundo Número

Plain Text

Propriedade	Valor
id	et_num2
width	200dp
numeric	Marque: integer, signed e decimal

Button

Propriedade	Valor
id	bt_somar
text	Somar
width	200dp

Button

Propriedade	Valor
id	bt_subtrair
text	Subtrair
width	200dp

Button

Propriedade	Valor
id	bt_dividir
text	Dividir
width	200dp

Button

Propriedade	Valor
id	bt_multiplicar
text	Multiplicar



width	200dp
-------	-------

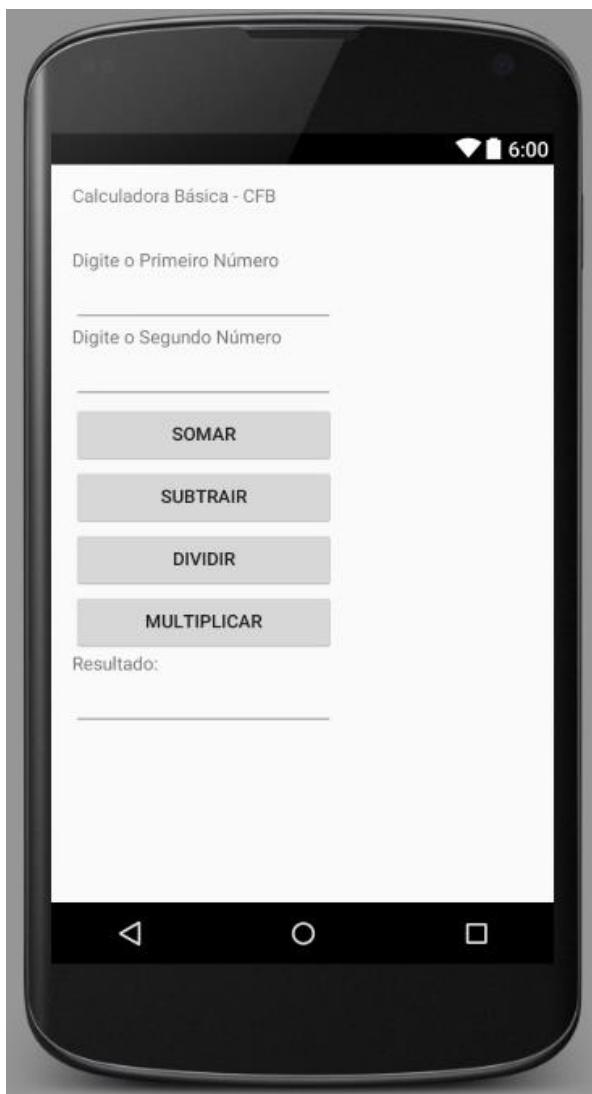
Plain TextView

Propriedade	Valor
id	Txtv4
text	Resultado:

Plain Text

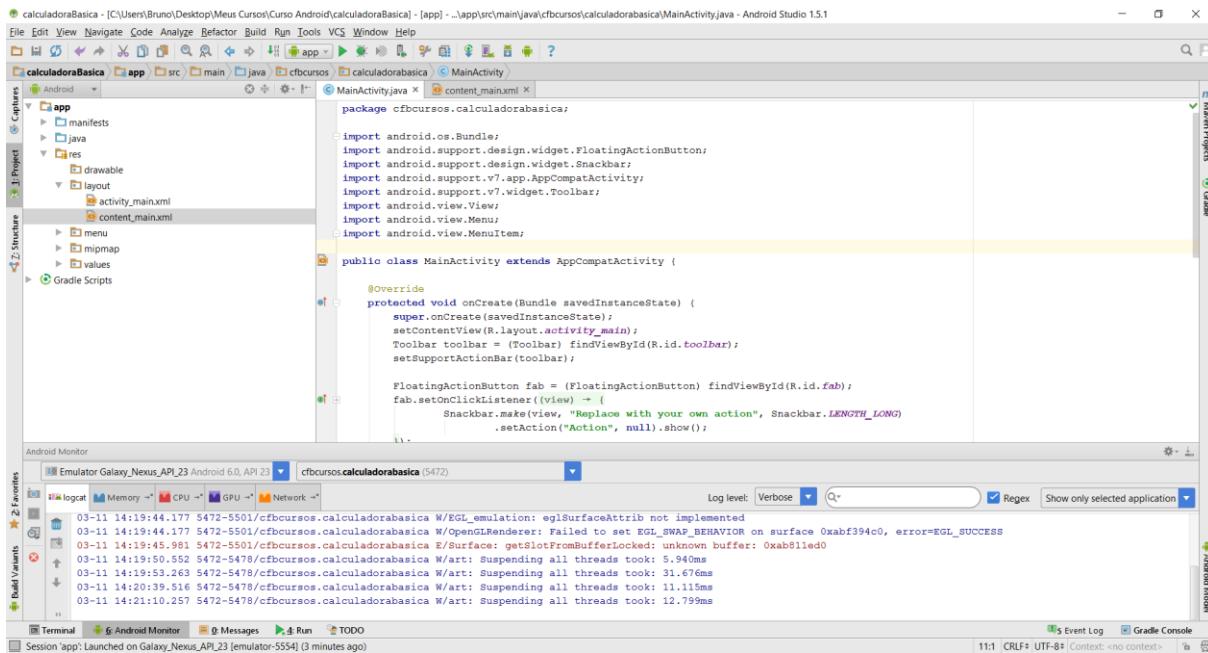
Propriedade	Valor
id	et_resultado
width	200dp
numeric	Marque: integer, signed e decimal

Confira a seguir o resultado do nosso layout.



Vamos à programação.

Abra o arquivo “MainActivity.java”.



O primeiro passo será importar a biblioteca “widget” para que possamos manipular com facilidade os componentes do layout.

```
import android.widget.*;
```

Veja na ilustração como deve ficar o código.

```
import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.*; //Para manipular os componentes
```

Com a biblioteca importada vamos criar as variáveis globais necessárias.

Primeiramente vamos declarar as variáveis relacionadas aos componentes.

```
//Variáveis para manipular os componentes
EditText vetnum1,vetnum2,vetres;
Button vbtsomar,vbtsubtrair,vbtdividir,vbtmultiplicar;
```

Em seguida vamos declarar as variáveis para realizar as operações de cálculo.

```
//Variáveis
double num1,num2,res;
```

Veja a ilustração da declaração das variáveis.



```
>MainActivity.java x content_main.xml x
package cfbcurros.calculadorabasica;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.*; //Para manipular os componentes

public class MainActivity extends AppCompatActivity {

    //Variáveis para manipular os componentes
    EditText vetnum1,vetnum2,vetres;
    Button vbtsomar,vbtsubtrair,vbtdividir,vbtmultiplicar;

    //Variáveis
    double num1,num2,res;

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

O próximo passo é associar os componentes do nosso layout com as variáveis criadas.

```
//Associando os componentes às variáveis criadas
vetnum1 = (EditText) findViewById(R.id.et_num1);
vetnum2 = (EditText) findViewById(R.id.et_num2);
vetres = (EditText) findViewById(R.id.et_resultado);
vbtsomar = (Button) findViewById(R.id.bt_somar);
vbtsubtrair = (Button) findViewById(R.id.bt_subtrair);
vbtdividir = (Button) findViewById(R.id.bt_dividir);
vbtmultiplicar = (Button) findViewById(R.id.bt_multiplicar);
```

Vamos à ilustração do código.

```
//Variáveis para manipular os componentes
EditText vetnum1,vetnum2,vetres;
Button vbtsomar,vbtsubtrair,vbtdividir,vbtmultiplicar;

//Variáveis
double num1,num2,res;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
    fab.setOnClickListener((view) ->
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();
    );

    //Associando os componentes às variáveis
    vetnum1 = (EditText) findViewById(R.id.et_num1);
    vetnum2 = (EditText) findViewById(R.id.et_num2);
    vetres = (EditText) findViewById(R.id.et_resultado);
    vbtsomar = (Button) findViewById(R.id.bt_somar);
    vbtsubtrair = (Button) findViewById(R.id.bt_subtrair);
    vbtdividir = (Button) findViewById(R.id.bt_dividir);
    vbtmultiplicar = (Button) findViewById(R.id.bt_multiplicar);
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
```



O último passo da programação é adicionar os eventos de clique em cada um dos botões logo após a associação das variáveis.

Já vimos como funciona o procedimento para adicionar o evento de clique no botão anteriormente, então vamos focar nos procedimentos dos cálculos destacados na cor vermelho.

```
//Adicionando a programação dos botões no evento de clique
    vbtSomar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1+num2;
            vetres.setText(String.valueOf(res));
        }
    });

    vbtSubtrair.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1-num2;
            vetres.setText(String.valueOf(res));
        }
    });

    vbtDividir.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1/num2;
            vetres.setText(String.valueOf(res));
        }
    });

    vbtMultiplicar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1*num2;
            vetres.setText(String.valueOf(res));
        }
    });
});
```

Vou detalhar usando o procedimento de soma.

Primeiramente inserimos nas variáveis num1 e num2 os valores digitados nos campos de texto EditText.

```
num1 = Double.parseDouble(vetnum1.getText().toString());
num2 = Double.parseDouble(vetnum2.getText().toString());
```

Como criamos as variáveis num1, num2 e res do tipo double, precisamos converter usando o método `Double.parseDouble`.

Como parâmetro do `parseDouble` pegamos o texto do elemento EditText (vetnum1 e vetnum2) com o método `getText()`, convertendo em string obviamente.

Calculamos, somando num1 e num2 e adicionando o resultado na variável res.

```
res = num1+num2;
```

Finalmente definimos o texto do componente EditText (vetres) com o valor da variável res convertido para string.

```
vetres.setText(String.valueOf(res));
```



Veja como fica o código completo.

```
package cfbcursos.calculadorabasica;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.*; //Para manipular os componentes

public class MainActivity extends AppCompatActivity {

    //Variáveis para manipular os componentes
    EditText vetnum1,vetnum2,vetres;
    Button vbtosumar,vbtsubtrair,vbtdividir,vbtmultiplicar;

    //Variáveis
    double num1,num2,res;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Snackbar.make(v, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    //Associando os componentes às variáveis
    vetnum1 = (EditText) findViewById(R.id.et_num1);
    vetnum2 = (EditText) findViewById(R.id.et_num2);
    vetres = (EditText) findViewById(R.id.et_resultado);
    vbtosumar = (Button) findViewById(R.id.bt_somar);
    vbtsubtrair = (Button) findViewById(R.id.bt_subtrair);
    vbtdividir = (Button) findViewById(R.id.bt_dividir);
    vbtmultiplicar = (Button) findViewById(R.id.bt_multiplicar);

    //Adicionando a programação dos botões no evento de clique
    vbtosumar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1+num2;
            vetres.setText(String.valueOf(res));
        }
    });

    vbtsubtrair.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1-num2;
            vetres.setText(String.valueOf(res));
        }
    });

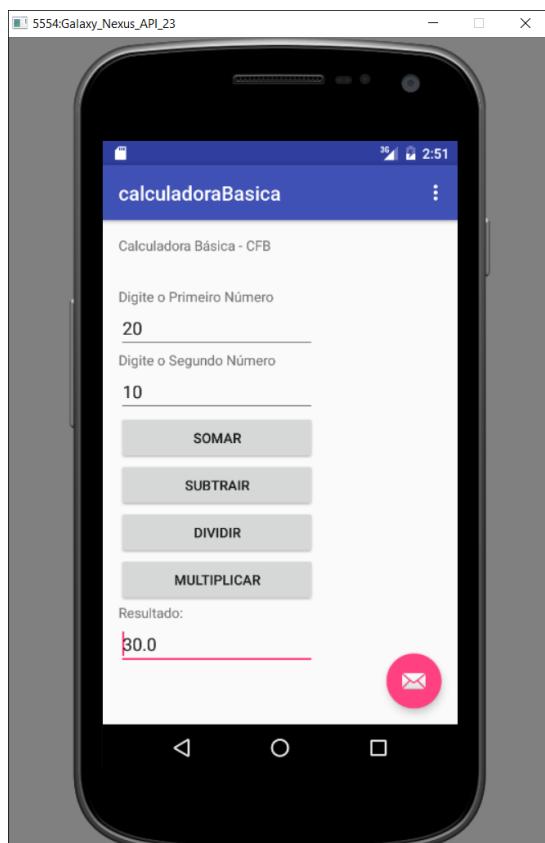
    vbtdividir.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            num1 = Double.parseDouble(vetnum1.getText().toString());
            num2 = Double.parseDouble(vetnum2.getText().toString());
            res = num1/num2;
            vetres.setText(String.valueOf(res));
        }
    });
}
```



```
});  
  
    vbtmultiplicar.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            num1 = Double.parseDouble(vetnum1.getText().toString());  
            num2 = Double.parseDouble(vetnum2.getText().toString());  
            res = num1*num2;  
            vetres.setText(String.valueOf(res));  
        }  
    });  
}  
  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    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);  
}  
}
```

Ótimo, tudo pronto, agora basta salvar as modificações e rodar nossa aplicação para testar.

A ilustração a seguir mostra o resultado após clicar no botão SOMAR.





Terceiro projeto – LocaFest – Aplicativo para locar itens para festas

Em nosso próximo aplicativo, vamos praticar um pouco de programação com os componentes checkBox, vamos criar um aplicativo onde o usuário irá selecionar os produtos que deseja locar e informar a quantidade que desejar de cada item.

Ao final irá clicar em calcular e obter o valor total da locação para os itens selecionados.

Vamos aos elementos que iremos adicionar para este aplicativo.

Plain TextView

Propriedade	Valor
text	Selecione os ítems e a quantidade que deseja

CheckBox

Propriedade	Valor
id	cb1
text	Taças: R\$0,25

CheckBox

Propriedade	Valor
id	cb2
text	Taças: R\$0,20

CheckBox

Propriedade	Valor
id	cb3
text	Taças: R\$0,15

CheckBox

Propriedade	Valor
id	cb4
text	Taças: R\$0,15

EditText

Propriedade	Valor
id	tb1
text	0
width	100dp
numeric	integer

EditText

Propriedade	Valor



id	tb2
text	0
width	100dp
numeric	integer

EditText

Propriedade	Valor
id	tb3
text	0
width	100dp
numeric	integer

EditText

Propriedade	Valor
id	tb4
text	0
width	100dp
numeric	integer

Button

Propriedade	Valor
id	btCalcular
text	Calcular

EditText

Propriedade	Valor
id	etRes
width	100dp
numeric	Decimal

Confira a seguir o visual do nosso aplicativo.





Pronto para programação?

Primeiramente vamos importar as bibliotecas “widget” e “AlertDialog”.

```
import android.widget.*;
import android.app.AlertDialog;
```

Em seguida precisamos criar as variáveis para associar os componentes.

```
CheckBox vcb1,vcb2,vcb3,vcb4;
EditText vet1,vet2,vet3,vet4,vetRes;
Button vBtCalc;
Double valorTotal;
```

Precisamos fazer as associações dos componentes às variáveis que criamos.

```
//Associações
vcb1=(CheckBox) findViewById(R.id.cb1);
vcb2=(CheckBox) findViewById(R.id.cb2);
vcb3=(CheckBox) findViewById(R.id.cb3);
vcb4=(CheckBox) findViewById(R.id.cb4);
vet1=(EditText) findViewById(R.id.et1);
vet2=(EditText) findViewById(R.id.et2);
vet3=(EditText) findViewById(R.id.et3);
vet4=(EditText) findViewById(R.id.et4);
vetRes=(EditText) findViewById(R.id.etRes);
vBtCalc=(Button) findViewById(R.id.btCalcular);
```

Por último vamos inserir a rotina do clique no botão de calcular.

A primeira tarefa neste código é zerar o valor da variável valorTotal.

Em seguida iremos verificar se o checkBox está selecionado, usando o comando IF, se estiver vamos multiplicar o valor da quantidade pelo valor unitário do item e somar ao valor total da locação.

Depois de vericiar todos os checkBox vamos adiconal o valor total da locação no EditText dos resultados.

Por último vamos mostrar a caixa de mensagem informando que o valor total foi calculado.

```
//Rotinas de clique nos checkbox
vBtCalc.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valorTotal=0.0;
        if(vcb1.isChecked()){
            valorTotal += 0.25 * Double.parseDouble(vet1.getText().toString());
        }
        if(vcb2.isChecked()){
            valorTotal += 0.20 * Double.parseDouble(vet2.getText().toString());
        }
        if(vcb3.isChecked()){
            valorTotal += 0.15 * Double.parseDouble(vet3.getText().toString());
        }
        if(vcb4.isChecked()){
            valorTotal += 0.15 * Double.parseDouble(vet4.getText().toString());
        }
        vetRes.setText(String.valueOf(valorTotal));

        AlertDialog.Builder cxMsg = new AlertDialog.Builder(MainActivity.this);

        cxMsg.setMessage("Valor total calculado\nClique em OK para fechar");

        cxMsg.setNeutralButton("OK", null);

        cxMsg.show();
    }
});
```



Veja o código completo.

```
package cfbcurso.locafest;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.*;
import android.app.AlertDialog;

public class MainActivity extends AppCompatActivity {

    CheckBox vcb1,vcb2,vcb3,vcb4;
    EditText vet1,vet2,vet3,vet4,vetRes;
    Button vBtCalc;
    Double valorTotal;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    //Associações
    vcb1=(CheckBox) findViewById(R.id.cb1);
    vcb2=(CheckBox) findViewById(R.id.cb2);
    vcb3=(CheckBox) findViewById(R.id.cb3);
    vcb4=(CheckBox) findViewById(R.id.cb4);
    vet1=(EditText) findViewById(R.id.et1);
    vet2=(EditText) findViewById(R.id.et2);
    vet3=(EditText) findViewById(R.id.et3);
    vet4=(EditText) findViewById(R.id.et4);
    vetRes=(EditText) findViewById(R.id.etRes);
    vBtCalc=(Button) findViewById(R.id.btCalcular);

    //Rotinas de clique nos checkbox
    vBtCalc.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            valorTotal=0.0;
            if(vcb1.isChecked()){
                valorTotal += 0.25 * Double.parseDouble(vet1.getText().toString());
            }
            if(vcb2.isChecked()){
                valorTotal += 0.20 * Double.parseDouble(vet2.getText().toString());
            }
            if(vcb3.isChecked()){
                valorTotal += 0.15 * Double.parseDouble(vet3.getText().toString());
            }
            if(vcb4.isChecked()){
                valorTotal += 0.15 * Double.parseDouble(vet4.getText().toString());
            }
            vetRes.setText(String.valueOf(valorTotal));
        }
    });

    AlertDialog.Builder cxMsg = new AlertDialog.Builder(MainActivity.this);
    cxMsg.setMessage("Valor total calculado\nClique em OK para fechar");
    cxMsg.setNeutralButton("OK",null);
    cxMsg.show();
});

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        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

```

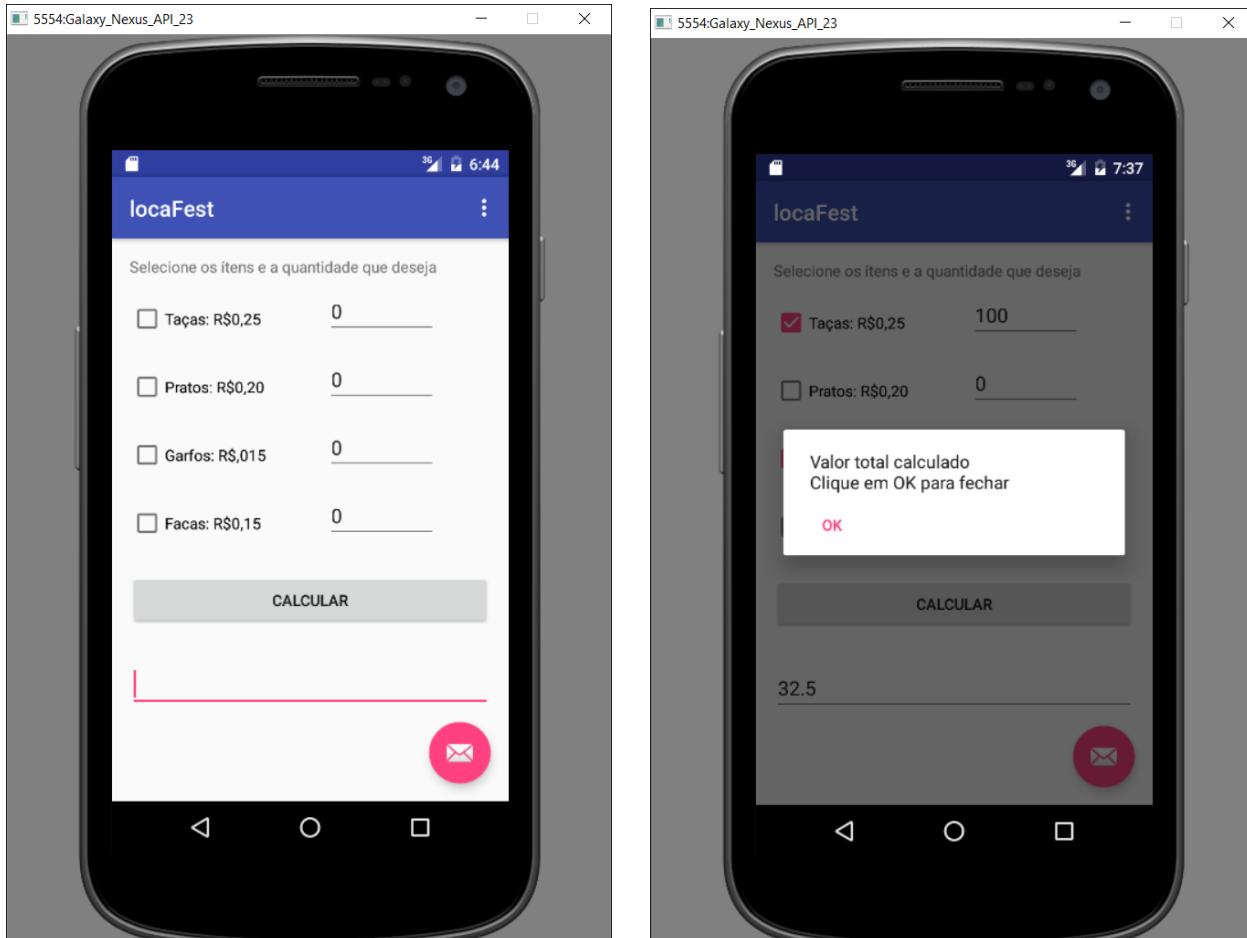


```
// automatically handle clicks on the Home/Up button, so long
// as you specify a parent activity in AndroidManifest.xml.
int id = item.getItemId();

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

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

Salve e rode a aplicação do emulador para testar o programa.



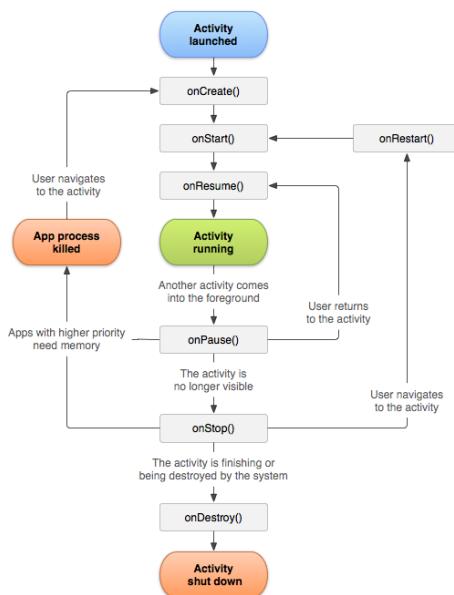
A classe principal AppCompatActivity

A classe “AppCompatActivity” é a classe base das nossas aplicações para Android, todos os aplicativos que formos construir no Android Studio irão estender desta classe. Esta classe vai prover todos os recursos necessários para construir nossos aplicativos no Android.

```
public class MainActivity extends AppCompatActivity {
```



Veja na ilustração a seguir o ciclo de funcionamento de uma Activity no Android.



Quando o aplicativo é carregado (Activity launched) são chamados os métodos “onCreate” e em seguida “onStart” onde a aplicação é iniciada e como não é iniciada em pause existe o método “onResume”, então aplicação será rodada efetivamente.

Enquanto a aplicação estiver sendo rodada, se uma outra Activity de prioridade maior for executada, nosso aplicativo entra em pause “onPause”, se uma ligação for recebida por exemplo o aplicativo entra em pausa.

Note que na pause o SO pode direcionar para três comportamentos “onResume” para voltar normalmente ao aplicativo de onde parou, pode encerrar o aplicativo “App process Killed” ou pode parar o aplicativo “onStop” e reiniciar novamente ao voltar “onRestart”.

Quanto o aplicativo for finalizado será chamado o método “onDestroy” e a activity irá ser encerrada “Activity shut down”.

Estes métodos são automaticamente definidos, porém, nós podemos personalizá-los.

Vamos criar uma nova aplicação com nome “vidaActivity”, após todo procedimento de criação, vamos observar o código básico.

```
package cfbcursos.vidaactivity;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        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);
    }
}
```

Note que o método “onCreate” já é inserido por padrão, o que iremos fazer é declarar os outros métodos que mencionamos anteriormente.

No código a seguir, destaquei em vermelho todos os comandos que precisamos adicionar, precisamos importar uma biblioteca para trabalhar com a mensagem de saída na janela de log.

```
package cfbcursos.vidaactivity;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.util.Log;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
        Log.v("CFB", "App Iniciado");
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        Log.v("CFB", "App Reiniciado");
    }

    @Override
    protected void onResume() {
        super.onResume();
        Log.v("CFB", "App Retornado");
    }

    @Override
    protected void onPause() {
        super.onPause();
        Log.v("CFB", "App Pausado");
    }

    @Override
    protected void onStop() {
        super.onStop();
        Log.v("CFB", "App Parado");
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.v("CFB", "App Destruido");
    }

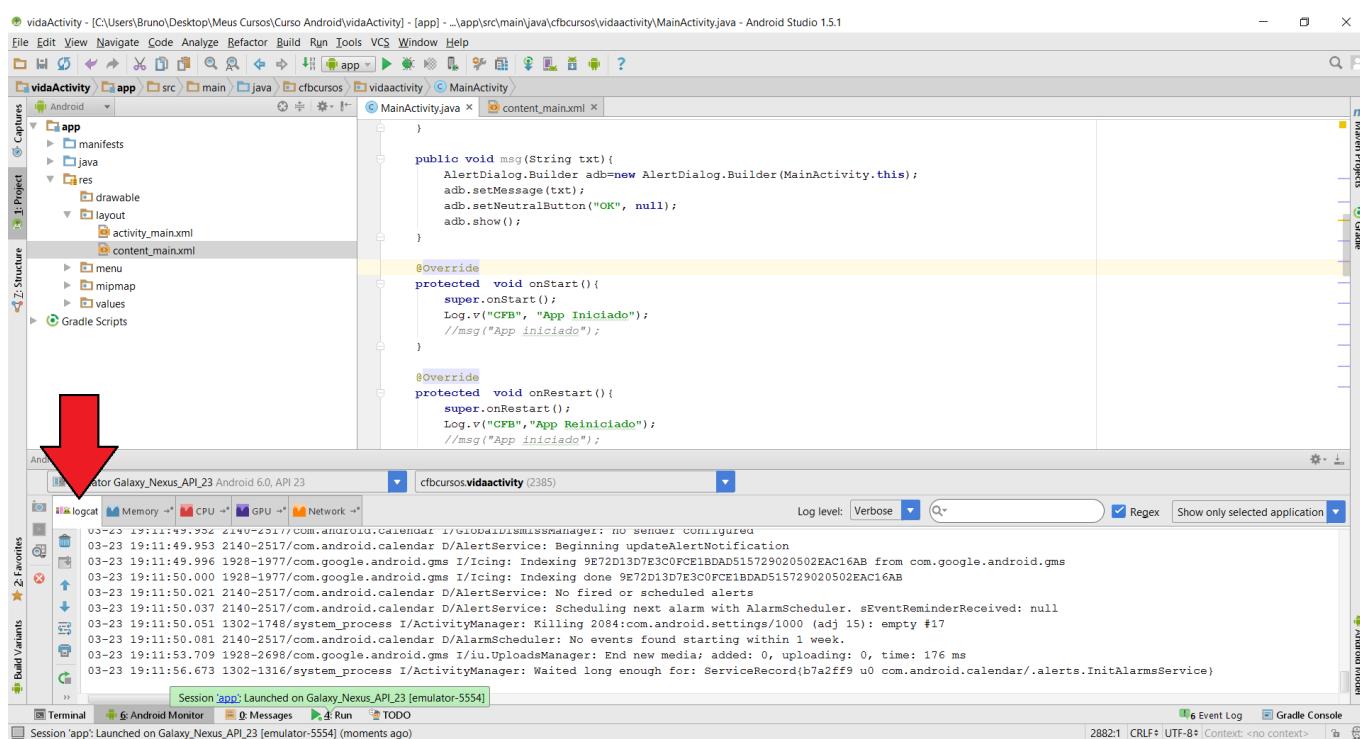
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}
```



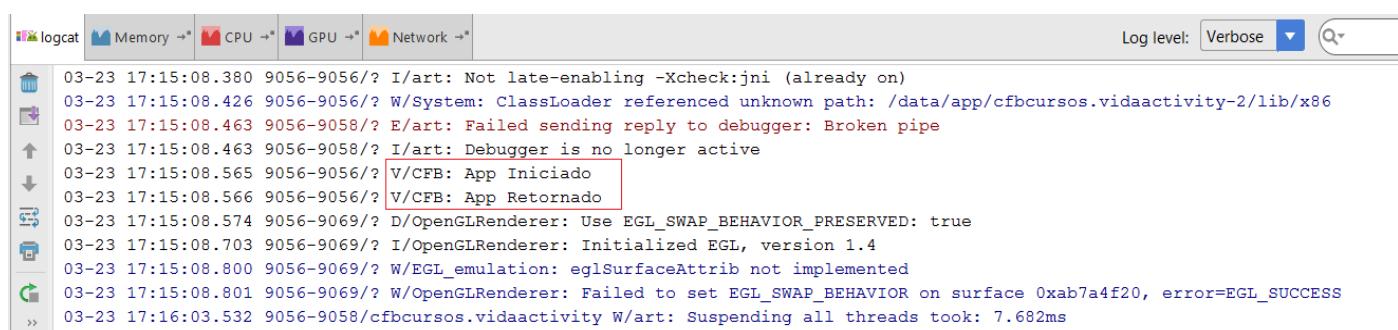
```
@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    // Handle action bar item clicks here. The action bar will  
    // automatically handle clicks on the Home/Up button, so long  
    // as you specify a parent activity in AndroidManifest.xml.  
    int id = item.getItemId();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

O comando “Log.v()” gera uma mensagem de texto na janela de log, vamos salvar e rodar nossa aplicação para verificar a mensagem.

Quando rodamos a aplicação a janela “Logcat” mostra todas as mensagens de log, inclusive as que definimos nos eventos.



Veja que no log temos as mensagens dos métodos “onStart” e “onResume”.



Quando pausamos o aplicativo veja as mensagens.



```
03-23 17:16:03.532 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 7.682ms
03-23 17:16:24.091 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 13.023ms
03-23 17:16:40.134 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 5.742ms
03-23 17:16:50.647 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 8.031ms
03-23 17:17:18.593 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 7.157ms
03-23 17:17:32.056 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 6.047ms
03-23 17:17:54.264 9056-9056/cfbcursos.vidaactivity V/CFB: App Pausado
03-23 17:17:55.053 9056-9069/cfbcursos.vidaactivity E/Surface: getSlotFromBufferLocked: unknown buffer: 0xab799f50
03-23 17:17:55.397 9056-9056/cfbcursos.vidaactivity V/CFB: App Parado
```

Agora iremos retomar o aplicativo.

```
03-23 17:17:55.053 9056-9069/cfbcursos.vidaactivity E/Surface: getSlotFromBufferLocked: unknown buffer: 0xab799f50
03-23 17:17:55.397 9056-9056/cfbcursos.vidaactivity V/CFB: App Parado
03-23 17:18:03.836 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 6.544ms
03-23 17:18:53.749 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 68.832ms
03-23 17:18:56.212 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 6.886ms
03-23 17:18:59.944 9056-9056/cfbcursos.vidaactivity V/CFB: App Reiniciado
03-23 17:18:59.948 9056-9056/cfbcursos.vidaactivity V/CFB: App Iniciado
03-23 17:18:59.950 9056-9056/cfbcursos.vidaactivity V/CFB: App Retornado
03-23 17:19:00.008 9056-9069/cfbcursos.vidaactivity W/EGL_emulation: eglSurfaceAttrib not implemented
03-23 17:19:00.008 9056-9069/cfbcursos.vidaactivity W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR on surface 0x
```

E por último vamos fechar.

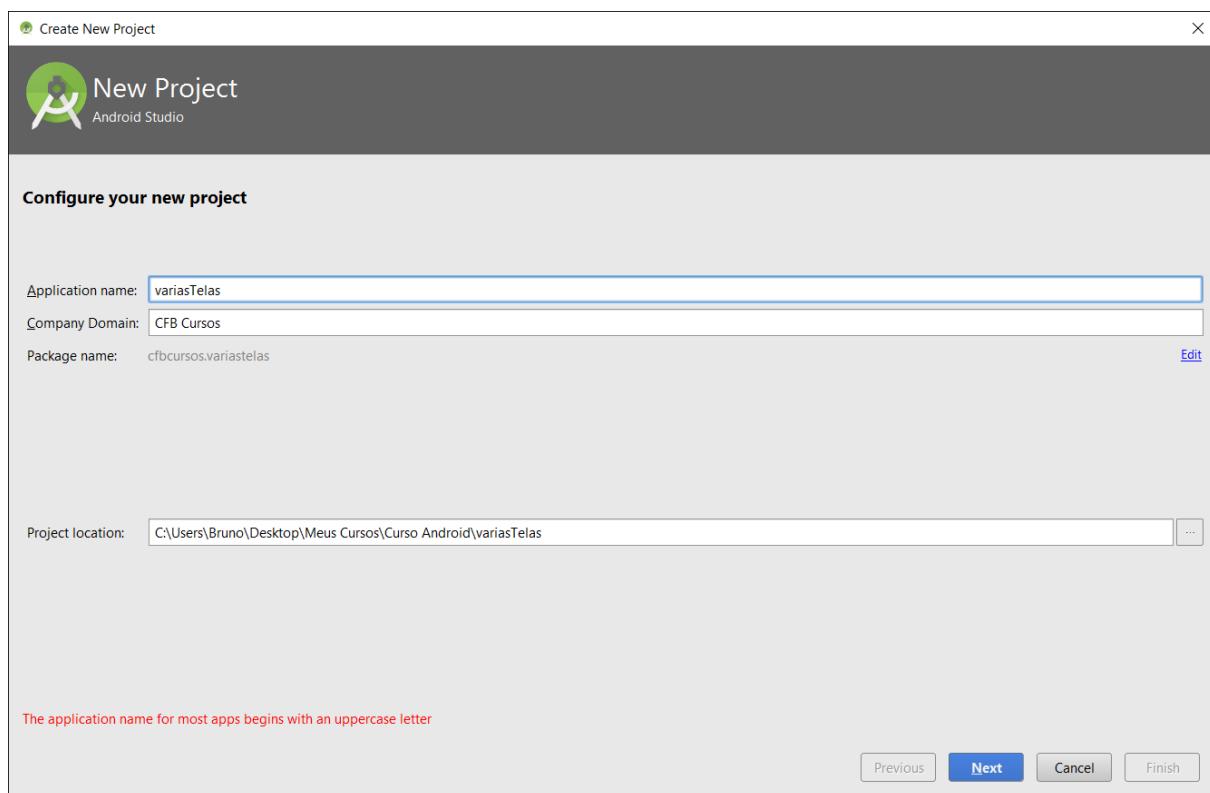
```
03-23 17:20:10.512 9056-9069/cfbcursos.vidaactivity W/EGL_emulation: eglSurfaceAttrib not implemented
03-23 17:20:10.512 9056-9069/cfbcursos.vidaactivity W/OpenGLRenderer: Failed to set EGL_SWAP_BEHAVIOR
03-23 17:20:10.547 9056-9062/cfbcursos.vidaactivity W/art: Suspending all threads took: 5.483ms
03-23 17:20:12.351 9056-9069/cfbcursos.vidaactivity E/Surface: getSlotFromBufferLocked: unknown buffer
03-23 17:20:14.886 9056-9056/cfbcursos.vidaactivity V/CFB: App Pausado
03-23 17:20:14.909 9056-9056/cfbcursos.vidaactivity V/CFB: App Parado
03-23 17:20:14.931 9056-9069/cfbcursos.vidaactivity E/Surface: getSlotFromBufferLocked: unknown buffer
03-23 17:20:15.670 9056-9062/cfbcursos.vidaactivity W/art: Suspending all threads took: 13.036ms
03-23 17:20:17.071 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 29.360ms
03-23 17:20:23.540 9056-9058/cfbcursos.vidaactivity W/art: Suspending all threads took: 7.152ms
```

Declarar este métodos em uma aplicação não é obrigatório, somente precisamos declará-los se for necessário executar alguma tarefa específica em algum deles.

Criando e Abrindo novas telas - XML

Neste capítulo vamos aprender como chamar uma nova tela em nosso aplicativo, vamos desenvolver um aplicativo com mais de uma tela e aprender a navegar entre estas telas.

Inicie um novo projeto no Android com nome “variasTelas” da mesma forma que fizemos com os anterios.



Neste projeto iremos ter três telas, uma principal e duas auxiliares e vamos criar botões para navegar entre as telas auxiliares e voltar para a tela principal.

Na tela principal construa o layout conforme os componentes a seguir.

Large Text (TextView)

Propriedade	Valor
text	Tela 1 - CFB

Button

Propriedade	Valor
id	bt2
text	Tela 2

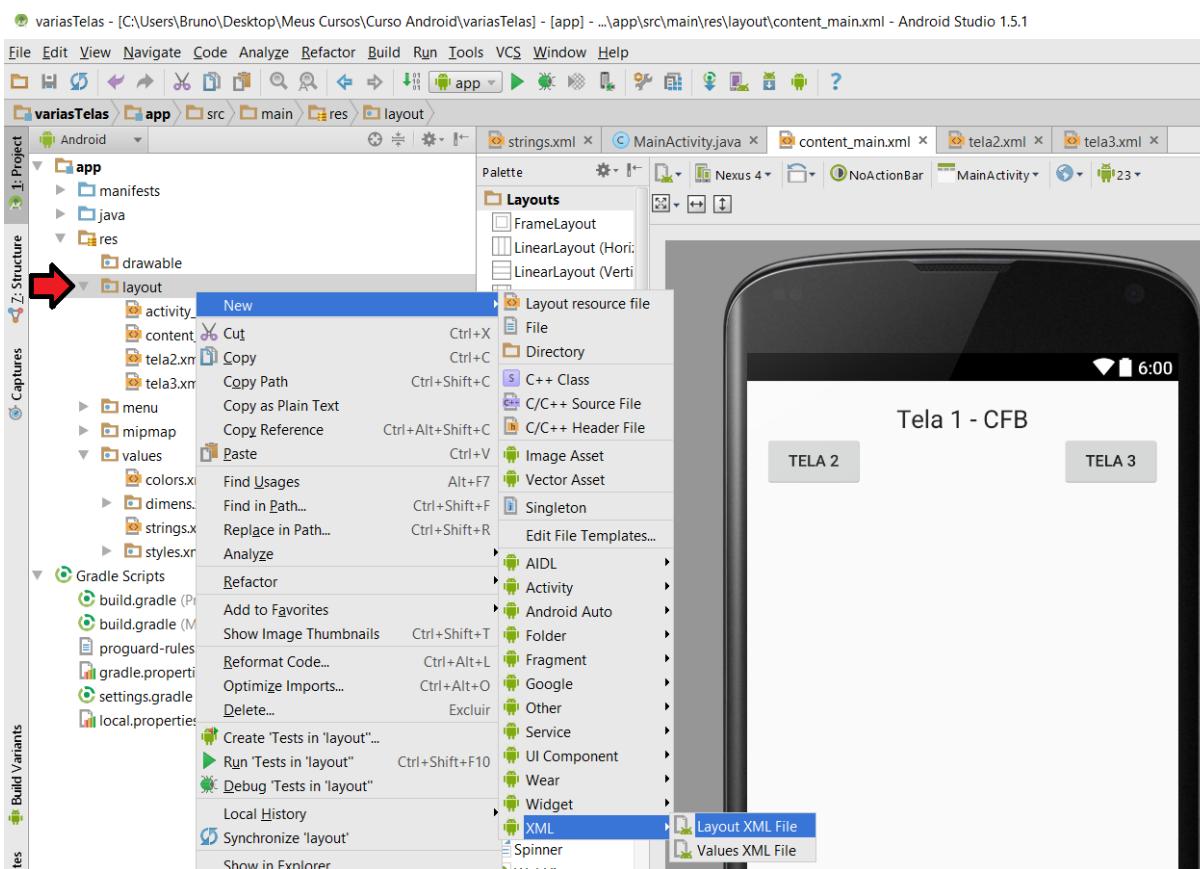
Button

Propriedade	Valor
id	bt3
text	Tela 3

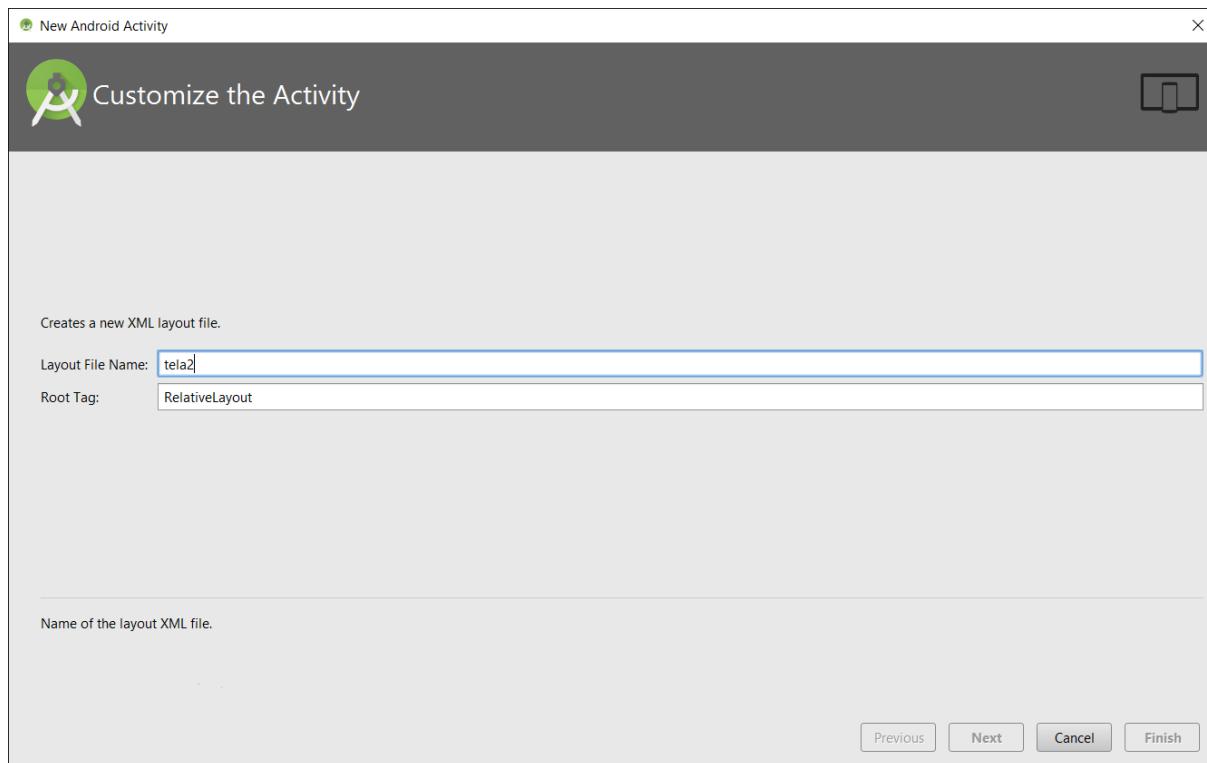


Iremos programar os botões para navegar nas telas que iremos criar, mas antes de programá-los vamos criar as telas.

Na janela de projeto do lado esquerdo clique com o botão direito do mouse na pasta “layout”, selecione New -> XML -> e clique em “Layout XML File” conforme a ilustração a seguir.

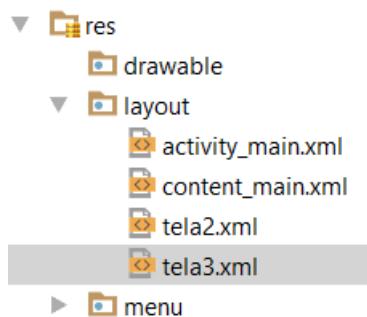


A janela a seguir será mostrada, configure com nome “tela2” e Root Tag com o valor “RelativeLayout” conforme a ilustração a seguir e clique em “Finish”.



Repita o procedimento para criar a terceira tela com nome “tela3” e Root Tag com o valor “RelativeLayout” conforme ilustração a seguir e clique em “Finish”.

Note que na tela de projeto agora temos os arquivos XML referentes à tela principal “content_main.xml” e as outras duas telas “tela2.xml” e “tela3.xml”.



Configurações dos elementos da tela 2.

Plain TextView (TextView)

Propriedade	Valor
text	2
textSize	50dp

Button

Propriedade	Valor
id	btv2
text	voltar



Configurações dos elementos da tela 3.

Plain TextView (TextView)

Propriedade	Valor
text	3
textSize	50dp

Button

Propriedade	Valor
id	btv3
text	voltar



Com as telas prontas podemos partir para programação.

Como não existem grandes novidades neste código, exceto p método “setContentView()” vou disponibilizar todo o código de uma só vez, a explicação está nos comentários do próprio código, então, preste atenção nos comentários, as partes inseridas estarão destacadas em vermelho.

```
package cfbcursos.variastelas;  
  
import android.net.Uri;  
import android.os.Bundle;  
import android.support.design.widget.FloatingActionButton;  
import android.support.design.widget.Snackbar;
```



```
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.*; //Import da biblioteca para manipular os componentes diretamente pelo código

import com.google.android.gms.appindexing.Action;
import com.google.android.gms.appindexing.AppIndex;
import com.google.android.gms.common.api.GoogleApiClient;

public class MainActivity extends AppCompatActivity {

    /**
     * ATTENTION: This was auto-generated to implement the App Indexing API.
     * See https://g.co/AppIndexing/AndroidStudio for more information.
     */
    private GoogleApiClient client;

    //Declaração dos botões da primeira tela
    Button btT2, btT3;

    //Método para abrir a tela 2
    public void abreTela2() {
        //Método que chama a tela 2
        setContentView(R.layout.tela2);
        //Declaração do botão para voltar à tela principal
        Button vbtv2 = (Button) findViewById(R.id.btv2);
        //Definição do evento de clique do botão voltar para tela principal
        vbtv2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Chama o método para voltar à tela principal
                voltaPrincipal();
            }
        });
    }

    //Método para abrir a tela 3
    public void abreTela3() {
        //Método que chama a tela 3
        setContentView(R.layout.tela3);
        //Declaração do botão para voltar à tela principal
        Button vbtv3 = (Button) findViewById(R.id.btv3);
        //Definição do evento de clique do botão voltar para tela principal
        vbtv3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Chama o método para voltar à tela principal
                voltaPrincipal();
            }
        });
    }

    //Método para chamar a tela principal
    public void voltaPrincipal() {
        //Método que chama a tela 3
        setContentView(R.layout.content_main);
        //Declaração dos botões para voltar abrir as telas 2 e 3
        Button btT2 = (Button) findViewById(R.id.bt2);
        Button btT3 = (Button) findViewById(R.id.bt3);

        //Definição dos eventos de clique dos botões para as telas 2 e 3
        btT2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v){
                abreTela2();
            }
        });
        btT3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                abreTela3();
            }
        });
    }

    //Aqui começa a execução principal do programa
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

```



```
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
        });

//Declaração dos botões para voltar abrir as telas 2 e 3
btT2 = (Button) findViewById(R.id.bt2);
btT3 = (Button) findViewById(R.id.bt3);

//Definição dos eventos de clique dos botões para as telas 2 e 3
btT2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v){
        abreTela2();
        Button vbtv2 = (Button) findViewById(R.id.btv2);
        vbtv2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v){
                voltaPrincipal();
            }
        });
    }
});
btT3.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v){
        abreTela3();
        Button vbtv3 = (Button) findViewById(R.id.btv3);
        vbtv3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v){
                voltaPrincipal();
            }
        });
    }
});

// ATTENTION: This was auto-generated to implement the App Indexing API.
// See https://g.co/AppIndexing/AndroidStudio for more information.
client = new GoogleApiClient.Builder(this).addApi(AppIndex.API).build();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    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);
}

@Override
public void onStart() {
    super.onStart();

    // ATTENTION: This was auto-generated to implement the App Indexing API.
    // See https://g.co/AppIndexing/AndroidStudio for more information.
    client.connect();
    Action viewAction = Action.newAction(
        Action.TYPE_VIEW, // TODO: choose an action type.
        "Main Page", // TODO: Define a title for the content shown.
        // TODO: If you have web page content that matches this app activity's content,
        // make sure this auto-generated web page URL is correct.
        // Otherwise, set the URL to null.
        Uri.parse("http://host/path"),
        // TODO: Make sure this auto-generated app deep link URI is correct.
        Uri.parse("android-app://cfbcursos.variastelas/http/host/path")
    );
    AppIndex.AppIndexApi.start(client, viewAction);
}

@Override
public void onStop() {
    super.onStop();

    // ATTENTION: This was auto-generated to implement the App Indexing API.
```



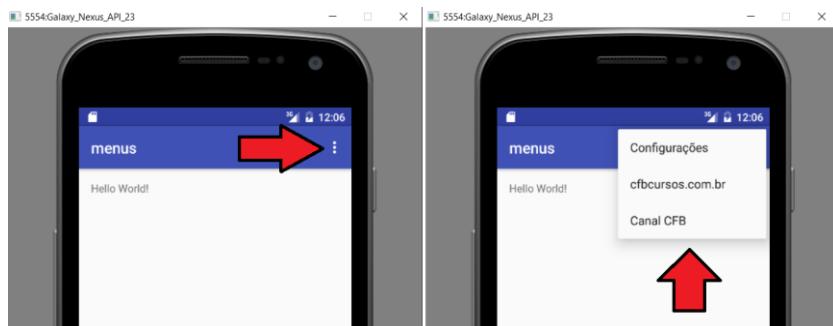
```
// See https://g.co/AppIndexing/AndroidStudio for more information.
Action viewAction = Action.newAction(
    Action.TYPE_VIEW, // TODO: choose an action type.
    "Main Page", // TODO: Define a title for the content shown.
    // TODO: If you have web page content that matches this app activity's content,
    // make sure this auto-generated web page URL is correct.
    // Otherwise, set the URL to null.
    Uri.parse("http://host/path"),
    // TODO: Make sure this auto-generated app deep link URI is correct.
    Uri.parse("android-app://cfbcursos.variastelas/http/host/path")
);
AppIndex.AppIndexApi.end(client, viewAction);
client.disconnect();
}
```

Basicamente o método setContentView() exibe o layout formado pelo arquivo XML, então, o comando setContentView(R.layout.tela2); mostra a tela formada pelo arquivo tela2.xml.

Salve todas as alterações e rode a aplicação para testar os botões.

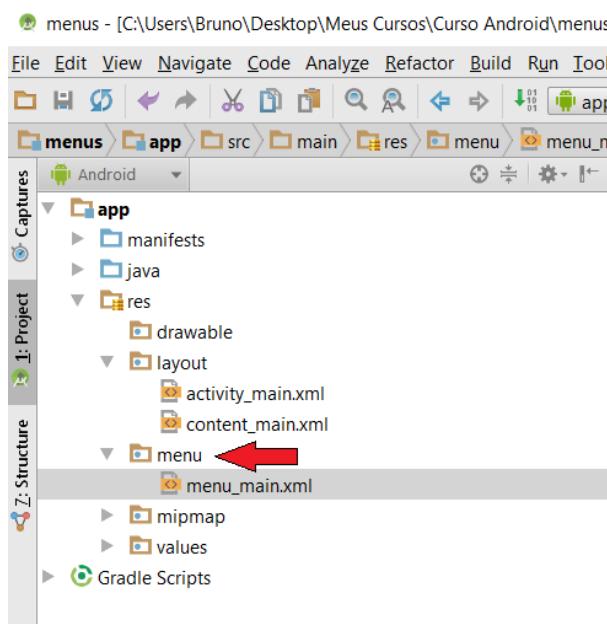
Definindo o menu de opções

Neste capítulo vamos aprender a definir, exibir e inserir itens no menu de opções da aplicação, é uma tarefa bem simples, veja o exemplo do menu que iremos criar.



Vamos criar um novo projeto com nome “menus”.

Quanto a criação do projeto estiver concluída vamos observar que já existe uma pasta reservada para o menu, inclusive já com um arquivo xml referente ao menu em nosso projeto.





Os elementos que irão compor nosso menu deverão ser inseridos neste arquivo xml “menu_main.xml”.

Aplique um clique duplo no arquivo “menu_main.xml” para abri-lo, veja que no código básico já existem um item no menu com id “action_settings” e título “settings”.

```
<menu 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"
    tools:context="cfbcursos.menus.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="settings"
        app:showAsAction="never" />
</menu>
```

Como vimos na ilustração do início deste capítulo, precisamos de três itens em nosso menu (Configurações, cfbcursos.com.br e Canal CFB).

O primeiro item já existe basta muda o id e title, os outros dois precisamos adicionar, então altere este código conforme a ilustração a seguir.

```
<menu 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"
    tools:context="cfbcursos.menus.MainActivity">
    <item
        android:id="@+id/configuracoes" ←
        android:orderInCategory="100"
        android:title="Configurações" ←
        app:showAsAction="never" />
    <item
        android:id="@+id/site"
        android:orderInCategory="100"
        android:title="cfbcursos.com.br"
        app:showAsAction="never" />
    <item
        android:id="@+id/canal"
        android:orderInCategory="100"
        android:title="Canal CFB"
        app:showAsAction="never" />
</menu>
```

Agora que os itens do menu estão definidos precisamos adicionar alguma funcionalidade a estes itens abra o código principal “MainActivity.java”.



Em nosso código principal já temos definidas as funções que adicionam o menu e verificam qual item foi clicado, o que torna nosso trabalho bem mais simples.

No final do código iremos encontrar os métodos “onCreateOptionsMenu” para exibir o menu de opções e “onOptionsItemSelected” que verifica qual o item que foi clicado.

O método “onCreateOptionsMenu”

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

Neste código chamamos o método chamado “inflate” que abre um determinado menu, em nosso caso o menu “menu_main.xml”

O método “onOptionsItemSelected” precisa ser alterado para se adaptar ao nosso menu, então vamos ao trabalho.

A seguir veja o método original.

```
@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);  
}
```

Agora vamos às alterações na estrutura IF destacadas em vermelho.

```
@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.configuracoes) {  
        return true;  
    }else if (id == R.id.site) {  
        return true;  
    }else if (id == R.id.canal) {  
        return true;  
    }  
  
    return super.onOptionsItemSelected(item);  
}
```

Iremos adicionar uma tarefa simples em cada um dos menus que será simplesmente mostrar uma mensagem, então vamos criar um método para mostrar uma mensagem, podemos criar no final de tudo como mostra a ilustração a seguir.



```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    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.configuracoes) {  
        return true;  
    }else if (id == R.id.site) {  
        return true;  
    }else if (id == R.id.canal) {  
        return true;  
    }  
  
    return super.onOptionsItemSelected(item);  
}  
  
public void msg(String txt){  
    AlertDialog.Builder al = new AlertDialog.Builder(MainActivity.this);  
    al.setMessage(txt);  
    al.setNeutralButton("OK", null);  
    al.show();  
}
```

Agora vamos chamar este método nos menus, cada um com sua mensagem.

```
@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.configuracoes) {  
        msg("Menu Configurações clicado"); ← Red arrow  
        return true;  
    }else if (id == R.id.site) {  
        msg("Menu Site clicado"); ← Red arrow  
        return true;  
    }else if (id == R.id.canal) {  
        msg("Menu Canal clicado"); ← Red arrow  
        return true;  
    }  
  
    return super.onOptionsItemSelected(item);  
}
```

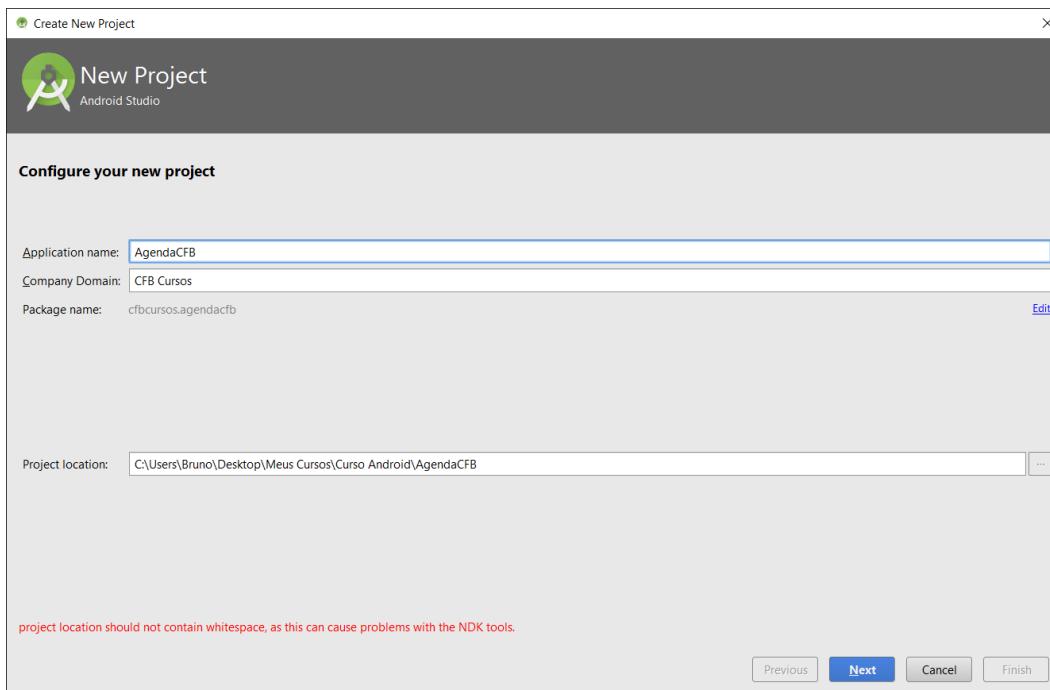


Tudo pronto, salve todas as alterações e rode a aplicação, clique nos menus para testar e veja que cada menu abre um caixa de mensagem com um texto diferente.

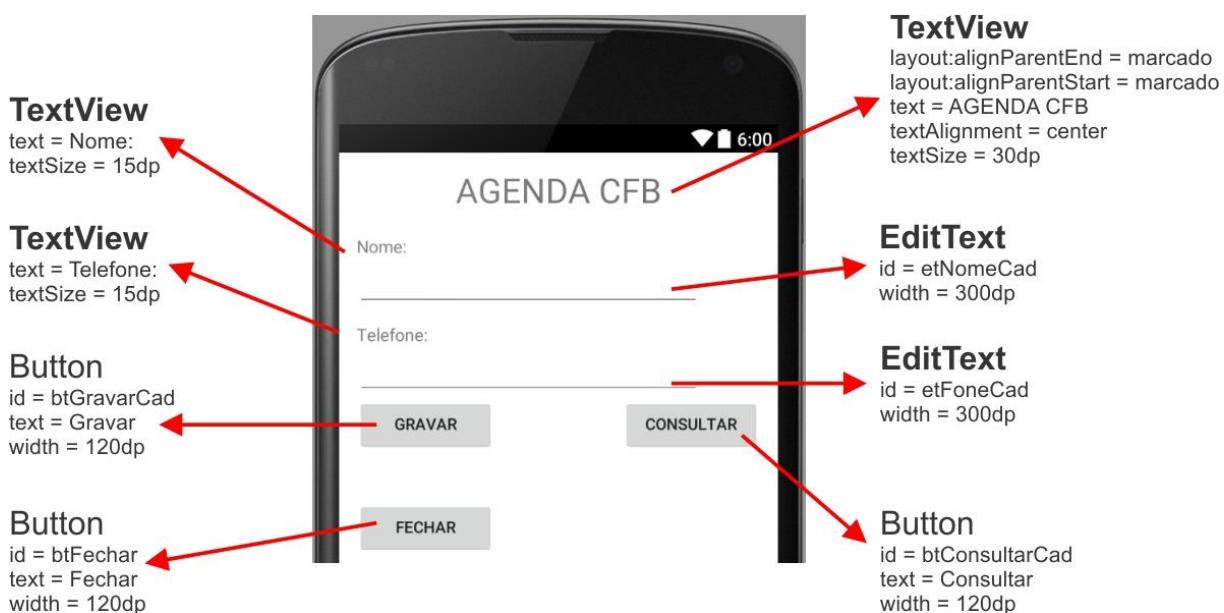
Agenda básica com banco de dados

Neste novo projeto iremos avançar um passo grande, iremos aprender como criar um banco de dados e a inserir dados e consultar os registros neste banco.

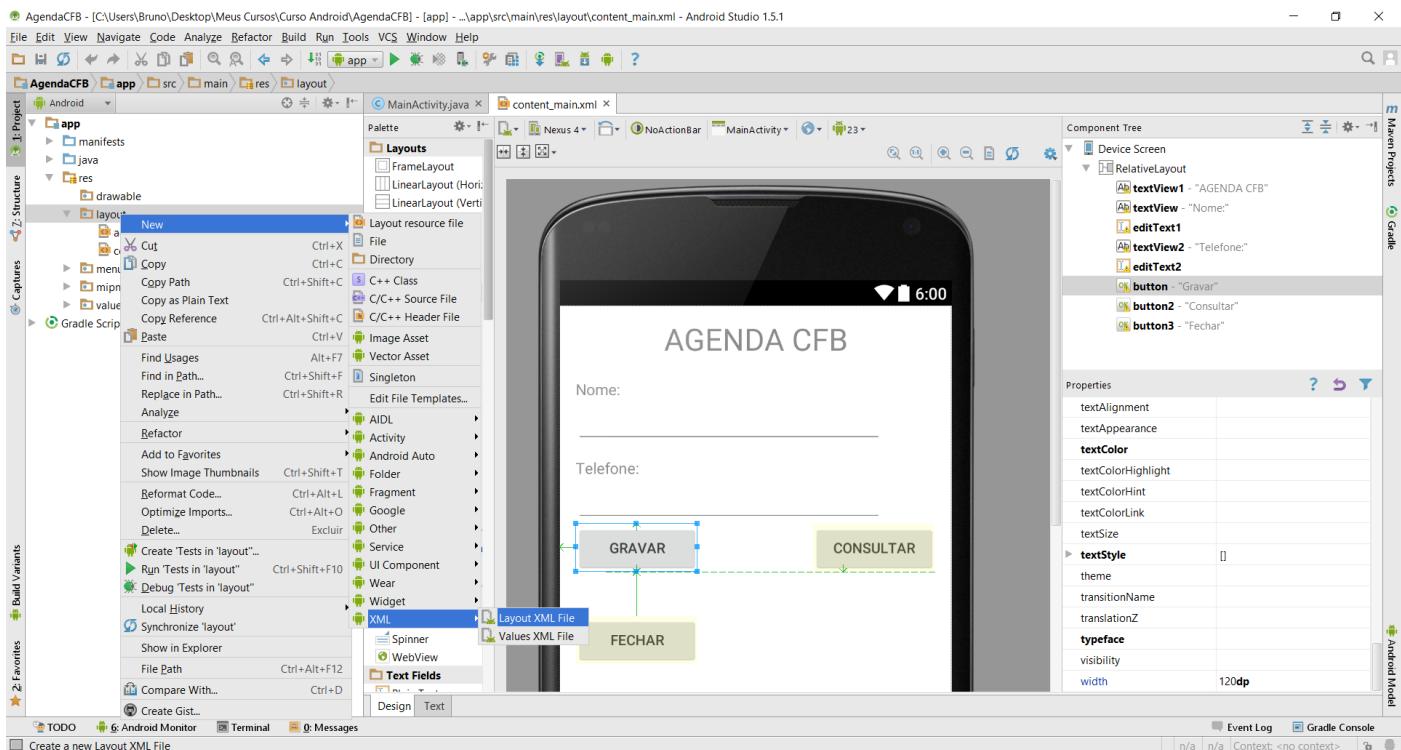
Vamos iniciar um novo projeto no Android Studio com nome “agendaCFB”.



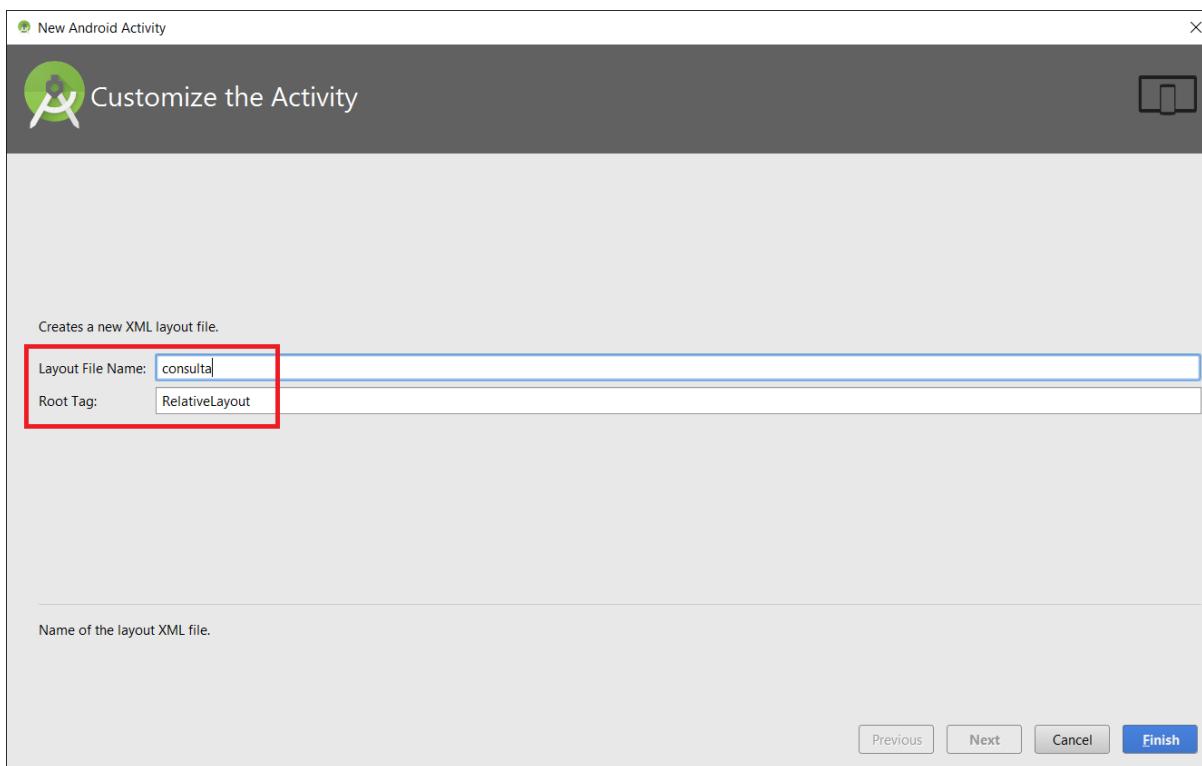
Adicione os elementos para a tela principal conforme a ilustração a seguir.



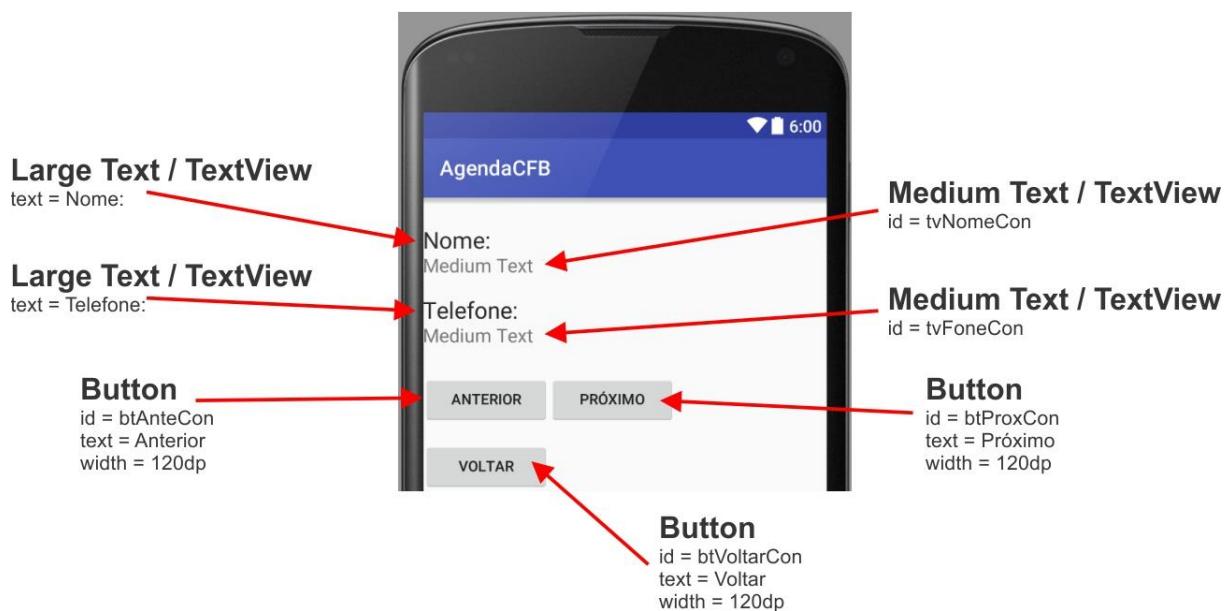
Agora vamos criar a tela de consulta, conforme a ilustração a seguir, clique com o botão direito do mouse sobre a pasta “layout” e selecione new -> XML -> Layout XML File.



Para o nome do arquivo digite “consulta” e em “Root Tag:” digite “RelativeLayout”.

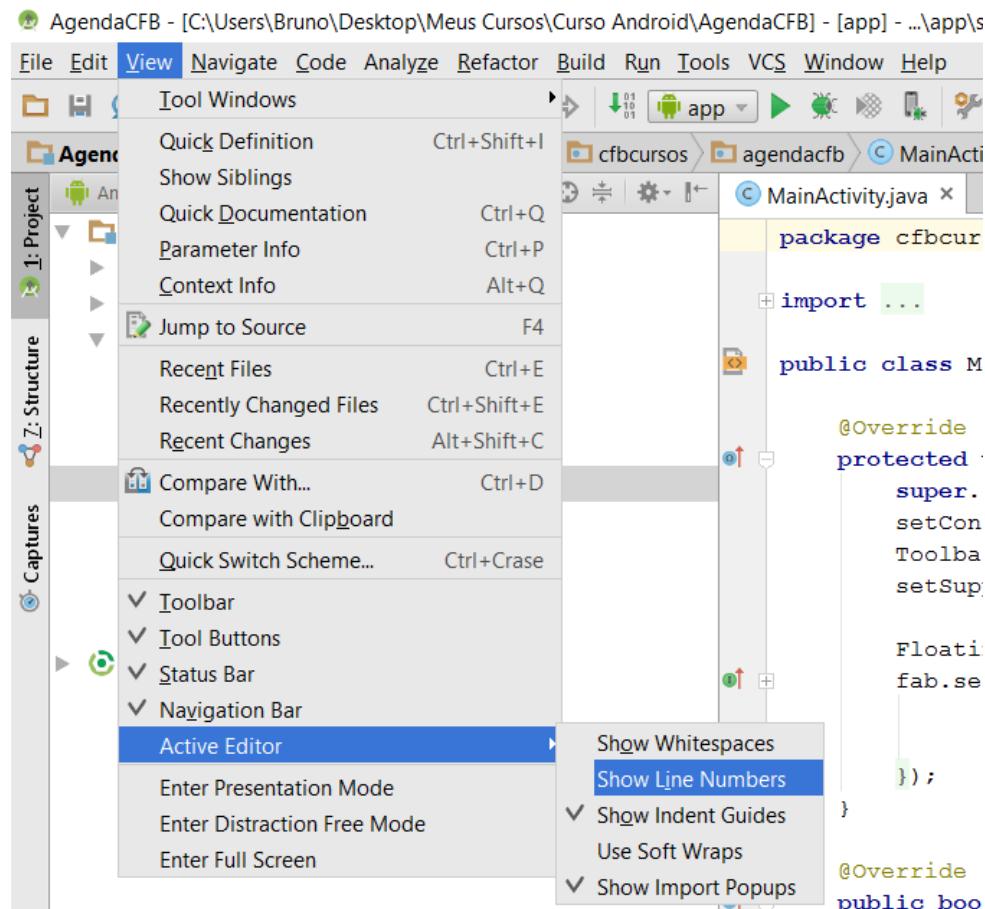


Clique em Finish para criar o layout básico, insira e configure os elementos conforme a ilustração a seguir.



Com as tela prontas é hora de programar, mas antes, vamos exibir o número das linhas na janela de código para facilitar nossa vida.

Selecione a janela do código principal “MainActivity.java”, em seguida clique no menu View -> Active Editor -> Show Line Numbers, veja a ilustração a seguir.





Isso fará com que as linhas de código sejam numeradas.

Vamos ao código, a seguir você confere o código básico existente.

```
package cfbcursos.agendacfb;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        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);
    }
}
```

A primeira tarefa vai ser importar as bibliotecas necessárias ao projeto.

```
>MainActivity.java x content_main.xml x consulta.xml x
1 package cfbcursos.agendacfb;
2
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11 import android.database.sqlite.SQLiteDatabase; //Banco de dados
12 import android.database.Cursor; //Navegar entre os dados no banco de dados
13 import android.widget.*; //Manipular os elementos diretamente via código
14
```

Iremos trabalhar com o banco de dados SQLite, o android já tem todo suporte necessário para trabalhar com este banco de dados.



Com os imports feitos, vamos criar um método para definir as funcionalidades da tela principal, vamos chamar este método de “telaPrincipal”.

```
11 import android.database.sqlite.SQLiteDatabase; //Banco de dados
12 import android.database.Cursor; //Navegar entre os dados no banco de dados
13 import android.widget.*; //Manipular os elementos diretamente via código
14
15 public class MainActivity extends AppCompatActivity {
16
17     public void telaPrincipal(){
18
19     }
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_main);
25         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
26         setSupportActionBar(toolbar);
```

Neste método vamos configurar os elementos que serão manipulados nesta tela e os botões, associando com as respectivas variáveis.

```
14
15     public class MainActivity extends AppCompatActivity {
16
17         public void telaPrincipal(){
18             EditText etNome=(EditText)findViewById(R.id.etNomeCad);
19             EditText etFone=(EditText)findViewById(R.id.etFoneCad);
20             Button btGrava=(Button)findViewById(R.id.btGravarCad);
21             Button btConsulta=(Button)findViewById(R.id.btConsultarCad);
22             Button btFechar=(Button)findViewById(R.id.btFechar);
23         }
24
25         @Override
26         protected void onCreate(Bundle savedInstanceState) {
27             super.onCreate(savedInstanceState);
```

Já podemos adicionar também os listeners dos botões.



```
13 import android.widget.*; //Manipular os elementos diretamente via código
14
15 public class MainActivity extends AppCompatActivity {
16
17     public void telaPrincipal(){
18         EditText etNome=(EditText)findViewById(R.id.etNomeCad);
19         EditText etFone=(EditText)findViewById(R.id.etFoneCad);
20         Button btGrava=(Button)findViewById(R.id.btGravarCad);
21         Button btConsulta=(Button)findViewById(R.id.btConsultarCad);
22         Button btFechar=(Button)findViewById(R.id.btFechar);
23
24         btGrava.setOnClickListener(new View.OnClickListener() {
25             @Override
26             public void onClick(View v) {
27                 //Rotina para gravação
28             }
29         });
29
30         btConsulta.setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View v) {
33                 //Rotina para consulta
34             }
35         });
36
37         btFechar.setOnClickListener(new View.OnClickListener() {
38             @Override
39             public void onClick(View v) {
40                 //Rotina para fechar o programa
41             }
42         });
43
44     }
45
46     @Override
47     protected void onCreate(Bundle savedInstanceState) {
48         super.onCreate(savedInstanceState);
```

Já podemos também criar o método que irá configurar os elementos da tela de consulta.

```
38         btFechar.setOnClickListener(new View.OnClickListener() {
39             @Override
40             public void onClick(View v) {
41                 //Rotina para fechar o programa
42             }
43         });
44
45
46     public void telaConsulta(){
47         TextView tvNome=(TextView)findViewById(R.id.tvNomeCon);
48         TextView tvFone=(TextView)findViewById(R.id.tvFoneCon);
49         Button btAnterior=(Button)findViewById(R.id.btAnteCon);
50         Button btProximo=(Button)findViewById(R.id.btProxCon);
51         Button btVoltar=(Button)findViewById(R.id.btVoltarCon);
52     }
53
54     @Override
55     protected void onCreate(Bundle savedInstanceState) {
56         super.onCreate(savedInstanceState);
57         setContentView(R.layout.activity_main);
58         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
59         setSupportActionBar(toolbar);
```



Assim como fizemos anteriormente, vamos adicionar os listeneres destes botões da tela de consulta.

```
46     public void telaConsulta() {
47         TextView tvNome=(TextView) findViewById(R.id.tvNomeCon);
48         TextView tvFone=(TextView) findViewById(R.id.tvFoneCon);
49         Button btAnterior=(Button) findViewById(R.id.btAnteCon);
50         Button btProximo=(Button) findViewById(R.id.btProxCon);
51         Button btVoltar=(Button) findViewById(R.id.btVoltarCon);
52
53         btAnterior.setOnClickListener(new View.OnClickListener() {
54             @Override
55             public void onClick(View v) {
56                 //Registro anterior
57             }
58         });
59
60         btProximo.setOnClickListener(new View.OnClickListener() {
61             @Override
62             public void onClick(View v) {
63                 //Próximo registro
64             }
65         });
66
67         btVoltar.setOnClickListener(new View.OnClickListener() {
68             @Override
69             public void onClick(View v) {
70                 //Voltar para tela principal
71             }
72         });
73     }
74
75     @Override
76     protected void onCreate(Bundle savedInstanceState) {
77         super.onCreate(savedInstanceState);
```

Vamos precisar de um método para facilitar a emissão de mensagens, então vamos criar um método com a rotina de caixas de mensagem.

```
68         btVoltar.setOnClickListener(new View.OnClickListener() {
69             @Override
70             public void onClick(View v) {
71                 //Voltar para tela principal
72             }
73         });
74
75
76         public void msg(String txt){
77             AlertDialog.Builder adb=new AlertDialog.Builder(MainActivity.this);
78             adb.setMessage(txt);
79             adb.setNeutralButton("OK",null);
80             adb.show();
81         }
82
83     @Override
84     protected void onCreate(Bundle savedInstanceState) {
85         super.onCreate(savedInstanceState);
86         setContentView(R.layout.activity_main);
87         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
88         setSupportActionBar(toolbar);
```

Vou detalhar este método.

Veja que inserimos um parâmetro de entrada do tipo “String” com nome “txt”, dentro do método inserimos a rotina básica para criação da caixa de mensagem “AlertDialog”.



No parâmetro do método “setMessage” veja que estamos passando o conteúdo do parâmetro “txt”, então, quando for chamar o método “msg” precisamos digitar a mensagem que desejamos exibir, como no exemplo a seguir.

```
msg("Mensagem a ser exibida");
```

Ótimo, agora vamos adicionar o método que irá carregar efetivamente as telas “setContentView” nos métodos “telaPrincipal” e “telaConsulta”.

```
16  public class MainActivity extends AppCompatActivity {  
17  
18      public void telaPrincipal(){  
19          setContentView(R.layout.content_main);  
20          EditText etNome=(EditText) findViewById(R.id.etNomeCad);  
21          EditText etFone=(EditText) findViewById(R.id.etFoneCad);  
22          Button btGrava=(Button) findViewById(R.id.btGravarCad);  
23          Button btConsulta=(Button) findViewById(R.id.btConsultarCad);  
24          Button btFechar=(Button) findViewById(R.id.btFechar);  
25  
26          btGrava.setOnClickListener(new View.OnClickListener() {
```

Agora para o método “telaConsulta”.

```
48      public void telaConsulta(){  
49          setContentView(R.layout.consulta);  
50          TextView tvNome=(TextView) findViewById(R.id.tvNomeCon);  
51          TextView tvFone=(TextView) findViewById(R.id.tvFoneCon);  
52          Button btAnterior=(Button) findViewById(R.id.btAnteCon);  
53          Button btProximo=(Button) findViewById(R.id.btProxCon);  
54          Button btVoltar=(Button) findViewById(R.id.btVoltarCon);  
55  
56          btAnterior.setOnClickListener(new View.OnClickListener() {  
57              @Override  
58              public void onClick(View v) {
```

Ótimo, estamos indo muito bem até agora, sem grandes novidades.

É hora de adicionar a chamada dos métodos “telaPrincipal” e “telaConsulta”.

O método “telaPrincipal” será chamado em dois momentos, primeiro no evento de clique do botão “btVoltar” presente na tela de consulta e depois no método principal “onCreate”.

Primeiro vamos adicionar no botão “btVoltar”.



```
48     public void telaConsulta() {
49         setContentView(R.layout.consulta);
50         TextView tvNome=(TextView) findViewById(R.id.tvNomeCon);
51         TextView tvFone=(TextView) findViewById(R.id.tvFoneCon);
52         Button btAnterior=(Button) findViewById(R.id.btAnteCon);
53         Button btProximo=(Button) findViewById(R.id.btProxCon);
54         Button btVoltar=(Button) findViewById(R.id.btVoltarCon);
55
56         btAnterior.setOnClickListener(new View.OnClickListener() {
57             @Override
58             public void onClick(View v) {
59                 //Registro anterior
60             }
61         });
62
63         btProximo.setOnClickListener(new View.OnClickListener() {
64             @Override
65             public void onClick(View v) {
66                 //Próximo registro
67             }
68         });
69
70         btVoltar.setOnClickListener(new View.OnClickListener() {
71             @Override
72             public void onClick(View v) {
73                 //Voltar para tela principal
74                 telaPrincipal();
75             }
76         });
77     }
```

Agora no “onCreate”.

```
86     @Override
87     protected void onCreate(Bundle savedInstanceState) {
88         super.onCreate(savedInstanceState);
89         setContentView(R.layout.activity_main);
90         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
91         setSupportActionBar(toolbar);
92
93         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
94         fab.setOnClickListener(new View.OnClickListener() {
95             @Override
96             public void onClick(View view) {
97                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
98                     .setAction("Action", null).show();
99             }
100        });
101
102        telaPrincipal();
103    }
```

Agora vamos adicionar a chamada do método “telaConsulta” no evento de clique do botão “btConsulta” da tela principal.

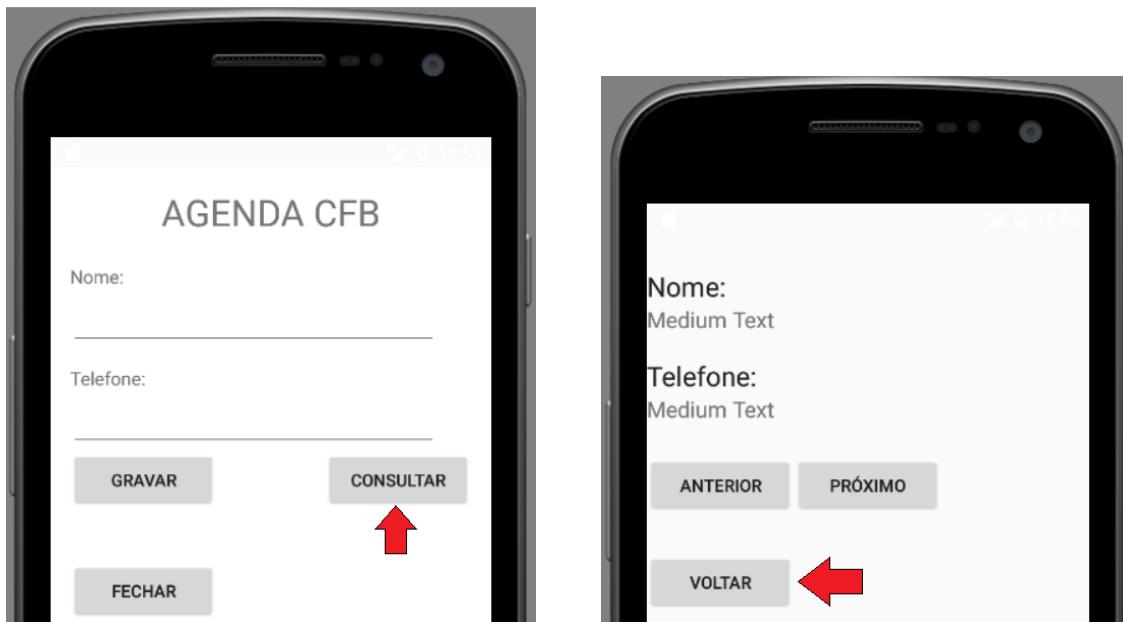


```
18     public void telaPrincipal() {
19         setContentView(R.layout.content_main);
20         EditText etNome=(EditText)findViewById(R.id.etNomeCad);
21         EditText etFone=(EditText)findViewById(R.id.etFoneCad);
22         Button btGrava=(Button)findViewById(R.id.btGravarCad);
23         Button btConsulta=(Button)findViewById(R.id.btConsultarCad);
24         Button btFechar=(Button)findViewById(R.id.btFechar);
25
26         btGrava.setOnClickListener(new View.OnClickListener() {
27             @Override
28             public void onClick(View v) {
29                 //Rotina para gravação
30             }
31         });
32
33         btConsulta.setOnClickListener(new View.OnClickListener() {
34             @Override
35             public void onClick(View v) {
36                 //Rotina para consulta
37                 telaConsulta();
38             }
39         });

```

É hora de testar para verificar se está tudo funcionando, salve as alterações e clique no botão “Run ‘app’” para rodar o aplicativo no emulador e verificar se está tudo OK.

Ao abrir clique nos botões “Consultar” e “Voltar” várias vezes para verificar se a navegação entre as telas está funcionando perfeitamente.



Com tudo funcionando até agora vamos adicionar a rotina para fechar nosso aplicativo.

Vamos ao evento de clique do botão “btFechar” presente na tela principal e adicionar a linha de comando para “matar” o processo referente à nossa aplicação, fechado assim nosso programa.



```
33     btConsulta.setOnClickListener(new View.OnClickListener() {
34         @Override
35         public void onClick(View v) {
36             //Rotina para consulta
37             telaConsulta();
38         }
39     });
40
41     btFechar.setOnClickListener(new View.OnClickListener() {
42         @Override
43         public void onClick(View v) {
44             //Rotina para fechar o programa
45             android.os.Process.killProcess(android.os.Process.myPid());
46         }
47     });
48
49
50     public void telaConsulta() {
```

Agora que toda rotina bálica está finalizada podemos partir para a criação e manipulação do banco de dados.

Vamos criar um método para abrir ou criar o banco de dados caso ele não exista, a primeira coisa a ser feita é declarar o banco de dados, as declarações serão feitas no topo do código, logo após a declaração da classe principal “MainActivity”.

```
10    import android.view.Menu;
11    import android.view.MenuItem;
12    import android.database.sqlite.SQLiteDatabase; //Banco de dados
13    import android.database.Cursor; //Navegar entre os dados no banco de dados
14    import android.widget.*; //Manipular os elementos diretamente via código
15
16    public class MainActivity extends AppCompatActivity {
17
18        SQLiteDatabase db = null;
19
20        public void telaPrincipal(){
21            setContentView(R.layout.content_main);
```

Agora podemos criar o método para abrir ou criar o bando de dados.

```
14    import android.widget.*; //Manipular os elementos diretamente via código
15
16    public class MainActivity extends AppCompatActivity {
17
18        SQLiteDatabase db = null;
19
20        public void abreCriadb() {
21            try{
22                db=openOrCreateDatabase("bancoCFB", MODE_PRIVATE, null);
23            }catch (Exception erro){
24                msg("Erro ao criar ou abrir banco");
25            }
26            try{
27                db.execSQL("CREATE TABLE IF NOT EXISTS contatos (id INTEGER PRIMARY KEY, nome TEXT, fone TEXT);");
28            }catch (Exception erro){
29                msg("Erro ao criar tabela");
30            }finally {
31                msg("Tabela contatos criada com sucesso.");
32            }
33        }
34
35        public void telaPrincipal(){
```

Note que inserimos o método “openOrCreateDatabase” em um bloco “try”, isso para que, caso ocorra algum erro, possamos tratar este erro e receber uma mensagem específica informando “Erro ao criar ou abrir banco”.



O mesmo procedimento de tratamento de erro foi adotado na criação da tabela “contatos”.

Note que criamos a tabela contatos com três campos, id, nome e fone.

Os blocos “try” não são obrigatórios, porém, nos possibilita um bom tratamento em caso de erro.

Precisamos também de um método para fechar o banco de dados, é uma boa prática fechar o banco de dados sempre que não estiver em uso.

```
27     db.execSQL("CREATE TABLE IF NOT EXISTS contatos (id
28         }catch (Exception erro){
29             msg("Erro ao criar tabela");
30         }finally {
31             msg("Tabela contatos criada com sucesso.");
32         }
33     }
34
35     public void fechadb() {
36         db.close();
37     }
38
39     public void telaPrincipal() {
40         setContentView(R.layout.content_main);
41         EditText etNome=(EditText) findViewById(R.id.etNomeCad);
42         EditText etFone=(EditText) findViewById(R.id.etFoneCad);
43         Button btGravar=(Button) findViewById(R.id.btGravarCad);
```

Já criamos os métodos para abrir e fechar nosso banco de dados, o próximo passo é criar um método para inserir os dados digitados nome e telefone na tabela contatos.

```
35     public void fechadb() {
36         db.close();
37     }
38
39     public void insereRegistro() {
40         try{
41             abreCriadb();
42             EditText etNome=(EditText) findViewById(R.id.etNomeCad);
43             EditText etFone=(EditText) findViewById(R.id.etFoneCad);
44             db.execSQL("INSERT INTO contatos (nome, fone) values ('" + etNome.getText().toString() + "','" + etFone.getText().toString() + "')");
45             msg("Registro inserido");
46             etNome.setText(null);
47             etFone.setText(null);
48             etNome.requestFocus();
49             fechadb();
50         }catch (Exception erro){
51             msg("Erro ao inserir registro");
52         }
53     }
54
55     public void telaPrincipal() {
56         setContentView(R.layout.content_main);
57         EditText etNome=(EditText) findViewById(R.id.etNomeCad);
```

No método “insereRegistro” temos as seguintes tarefas, linha a linha.

41 = Chama o método para abrir ou criar o banco de dados e a tabela contatos.

42 e 43 = Associação dos elementos EditText às variáveis etNome e etFone.

44 = O método execSQL executa um comando SQL passado como parâmetro, no nosso caso o comando para inserir os dados digitados nos EditText nos campos da nossa tabela contatos.

45 = Exibimos uma caixa de mensagem informando que o registro foi inserido.

46 e 47 = Apagamos os textos digitados nos campos EditText.

48 = Inserimos o cursor de inserção no campo nome.

49 = Após todos os procedimentos fechamos o banco de dados.

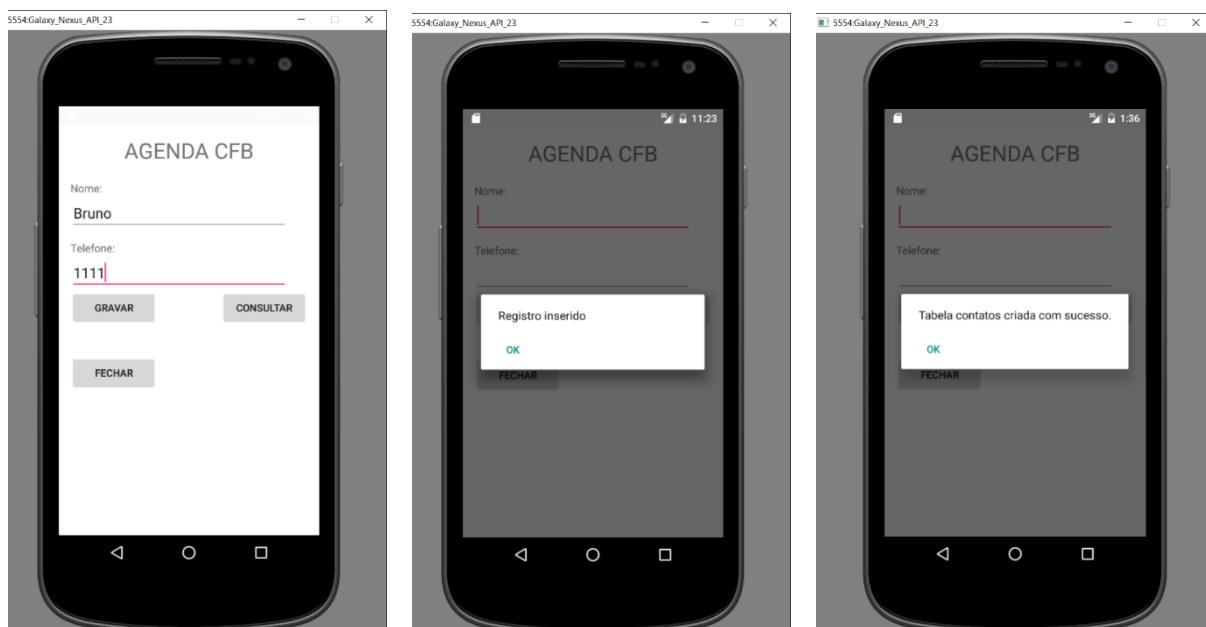
Ótimo, agora precisamos inserir a chamada do método “insereRegistro” no evento de clique do botão gravar da tela principal.



```
55     public void telaPrincipal(){
56         setContentView(R.layout.content_main);
57         EditText etNome=(EditText)findViewById(R.id.etNomeCad);
58         EditText etFone=(EditText)findViewById(R.id.etFoneCad);
59         Button btGrava=(Button)findViewById(R.id.btGravarCad);
60         Button btConsulta=(Button)findViewById(R.id.btConsultarCad);
61         Button btFechar=(Button)findViewById(R.id.btFechar);
62
63         btGrava.setOnClickListener(new View.OnClickListener() {
64             @Override
65             public void onClick(View v) {
66                 //Rotina para gravação
67                 insereRegistro();
68             }
69         });
70
71         btConsulta.setOnClickListener(new View.OnClickListener() {
72             @Override
73             public void onClick(View v) {
74                 //Rotina para consulta
```

É hora de testar e verificar as rotinas de criação e inserção de registros, Salve todas as alterações e vamos rodar nossa aplicação no emulador.

Digite um nome e um telefone e clique no botão Gravar, veja que deve ser mostrada a mensagem “Registro inserido”, após esta mensagem será mostrada outra mensagem “Tabela contatos criada com sucesso”.



Esta última mensagem é resultado do “finally” do bloco “try” do método “abreCiadb”, para não mostrar mais esta mensagem vamos remover este “finally”, a ilustração a seguir mostra o código que será removido “riscado em vermelho”.



```
20     public void abreCriadb() {
21         try{
22             db=openOrCreateDatabase("bancoCFB", MODE_PRIVATE, null);
23         }catch (Exception erro){
24             msg("Erro ao criar ou abrir banco");
25         }
26         try{
27             db.execSQL("CREATE TABLE IF NOT EXISTS contatos (id INTEGER PRIMARY KEY, nome TEXT, fone TEXT);");
28         }catch (Exception erro){
29             msg("Erro ao criar tabela");
30         }finally{
31             msg("Tabela contatos criada com sucesso.");
32         }
33     }
```

A seguir o código com o “finally” removido.

```
20     public void abreCriadb() {
21         try{
22             db=openOrCreateDatabase("bancoCFB", MODE_PRIVATE, null);
23         }catch (Exception erro){
24             msg("Erro ao criar ou abrir banco");
25         }
26         try{
27             db.execSQL("CREATE TABLE IF NOT EXISTS contatos (id INTEGER PRIMARY KEY, nome TEXT, fone TEXT);");
28         }catch (Exception erro){
29             msg("Erro ao criar tabela");
30         }
31     }
```

Insira mais alguns registros em nosso banco de dados e vamos para as rotinas de consulta, iremos criar um método para mostrar os dados da consulta nos elementos “TextView” da tela de consulta, mas antes precisamos declarar o cursor para nos movimentar entre os registros.

```
12     import android.database.sqlite.SQLiteDatabase; //Banco de dados
13     import android.database.Cursor; //Navegar entre os dados no banco de dados
14     import android.widget.*; //Manipular os elementos diretamente via código
15
16     public class MainActivity extends AppCompatActivity {
17
18         SQLiteDatabase db = null;
19         Cursor cursor;
20
21         public void abreCriadb() {
22             try{
23                 db=openOrCreateDatabase("bancoCFB", MODE_PRIVATE, null);
24             }catch (Exception erro){
```

Agora vamos à construção do método para buscar os dados na tabela.



```
54     public void mostrarDados() {
55         TextView etNome=(TextView)findViewById(R.id.tvNomeCon);
56         TextView etFone=(TextView)findViewById(R.id.tvFoneCon);
57
58         etNome.setText(cursor.getString(cursor.getColumnIndex("nome")));
59         etFone.setText(cursor.getString(cursor.getColumnIndex("fone")));
60     }
61
62     public boolean buscarDados() {
63         try{
64             abreCriadb();
65             cursor = db.query("contatos",
66                 new String[]{"nome", "fone"},
67                 null,//Selection / Where
68                 null,//Selection args / Parâmetros do where
69                 null,//Group by
70                 null,//having
71                 null,//"ORDER BY nome",
72                 null//Limite dos registros retornado
73             );
74             if(cursor.getCount()!= 0){
75                 cursor.moveToFirst();
76                 mostrarDados();
77                 return true;
78             }else {
79                 msg("Nenhum registro");
80                 return false;
81             }
82         }catch (Exception erro) {
83             msg("Erro ao buscar registros");
84             return false;
85         }
86     }
87
88     public void telaPrincipal(){}
```

Vamos aos detalhes deste método, linha a linha.

64 = Chama o método para abrir o banco de dados.

65 a 73 = Query para realizar a consulta e retornar para o cursor.

74 = SE contagem de registros for diferente de zero

75 = Posiciona no primeiro registro

76 = Chama o método para mostrar os registros nos TextView

Vamos adicionar a chamada do método “buscarDados” dentro do método “telaConsulta”.



```
121 public void telaConsulta(){
122     setContentView(R.layout.consulta);
123     TextView tvNome=(TextView) findViewById(R.id.tvNomeCon);
124     TextView tvFone=(TextView) findViewById(R.id.tvFoneCon);
125     Button btAnterior=(Button) findViewById(R.id.btAnteCon);
126     Button btProximo=(Button) findViewById(R.id.btProxCon);
127     Button btVoltar=(Button) findViewById(R.id.btVoltarCon);
128
129     buscarDados();
130
131     btAnterior.setOnClickListener(new View.OnClickListener() {
132         @Override
133         public void onClick(View v) {
134             //Registro anterior
135         }
136     });
137
138     btProximo.setOnClickListener(new View.OnClickListener() {
139         @Override
140         public void onClick(View v) {
141             //Próximo registro
142         }
143     });

```

Salve as alterações e vamos rodar nosso programa para verificar se tudo está funcionando normalmente, clique no botão “Consultar” e veja que o primeiro registro já está sendo mostrado.



Precisamos implementar agora os métodos para navegar entre os registros, vamos criar os métodos “proxReg” e “antReg” para navegar entre os registros.



```
145
146
147 btVoltar.setOnClickListener(new View.OnClickListener() {
148     @Override
149     public void onClick(View v) {
150         //Volta para tela principal
151         telaPrincipal();
152     }
153 });
154
155 public void proxReg() {
156     try {
157         cursor.moveToNext();
158         mostrarDados();
159     }catch (Exception erro){
160         if(cursor.isAfterLast()){
161             msg("Não há mais registros");
162         }else {
163             msg("Erro ao navegar");
164         }
165     }
166 }
167 public void msg(String txt){
168     AlertDialog.Builder adb=new AlertDialog.Builder(MainActivity.this);
169     adb.setMessage(txt);
170 }
```

Veja que simplesmente usamos o método “moveToNext” para mover para o próximo registro, chamamos o método “mostrarDados” para atualizar os campos e o restante é tratamento de erro.

Vamos ao método para o registro anterior.

```
154
155     public void proxReg() {
156         try {
157             cursor.moveToNext();
158             mostrarDados();
159         }catch (Exception erro){
160             if(cursor.isAfterLast()){
161                 msg("Não há mais registros");
162             }else {
163                 msg("Erro ao navegar");
164             }
165         }
166
167     public void antReg() {
168         try{
169             cursor.moveToPrevious();
170             mostrarDados();
171         }catch (Exception erro){
172             if(cursor.isBeforeFirst()){
173                 msg("Não há registros anteriores");
174             }else {
175                 msg("Erro ao navegar");
176             }
177         }
178     }
179
180     public void msg(String txt){
181         AlertDialog.Builder adb=new AlertDialog.Builder(MainActivity.this);
182         adb.setMessage(txt);
183         adb.setNeutralButton("OK", null);
184         adb.show();
185     }
186 }
```

A diferença é que para o registro anterior chamamos o método “moveToPrevious”.

Com os métodos definidos, precisamos adicioná-los aos devidos botões “Anterior” e “Próximo” de carona vamos adicionar o método para fechar o banco de dados no botão voltar.



```
121     public void telaConsulta() {
122         setContentView(R.layout.consulta);
123         TextView tvNome=(TextView)findViewById(R.id.tvNomeCon);
124         TextView tvFone=(TextView)findViewById(R.id.tvFoneCon);
125         Button btAnterior=(Button)findViewById(R.id.btAnteCon);
126         Button btProximo=(Button)findViewById(R.id.btProxCon);
127         Button btVoltar=(Button)findViewById(R.id.btVoltarCon);
128
129         buscarDados();
130
131         btAnterior.setOnClickListener(new View.OnClickListener() {
132             @Override
133             public void onClick(View v) {
134                 //Registro anterior
135                 antReg();
136             }
137         });
138
139         btProximo.setOnClickListener(new View.OnClickListener() {
140             @Override
141             public void onClick(View v) {
142                 //Próximo registro
143                 proxReg();
144             }
145         });
146
147         btVoltar.setOnClickListener(new View.OnClickListener() {
148             @Override
149             public void onClick(View v) {
150                 //Voltar para tela principal
151                 fechadb();
152                 telaPrincipal();
153             }
154         });
155     }
```

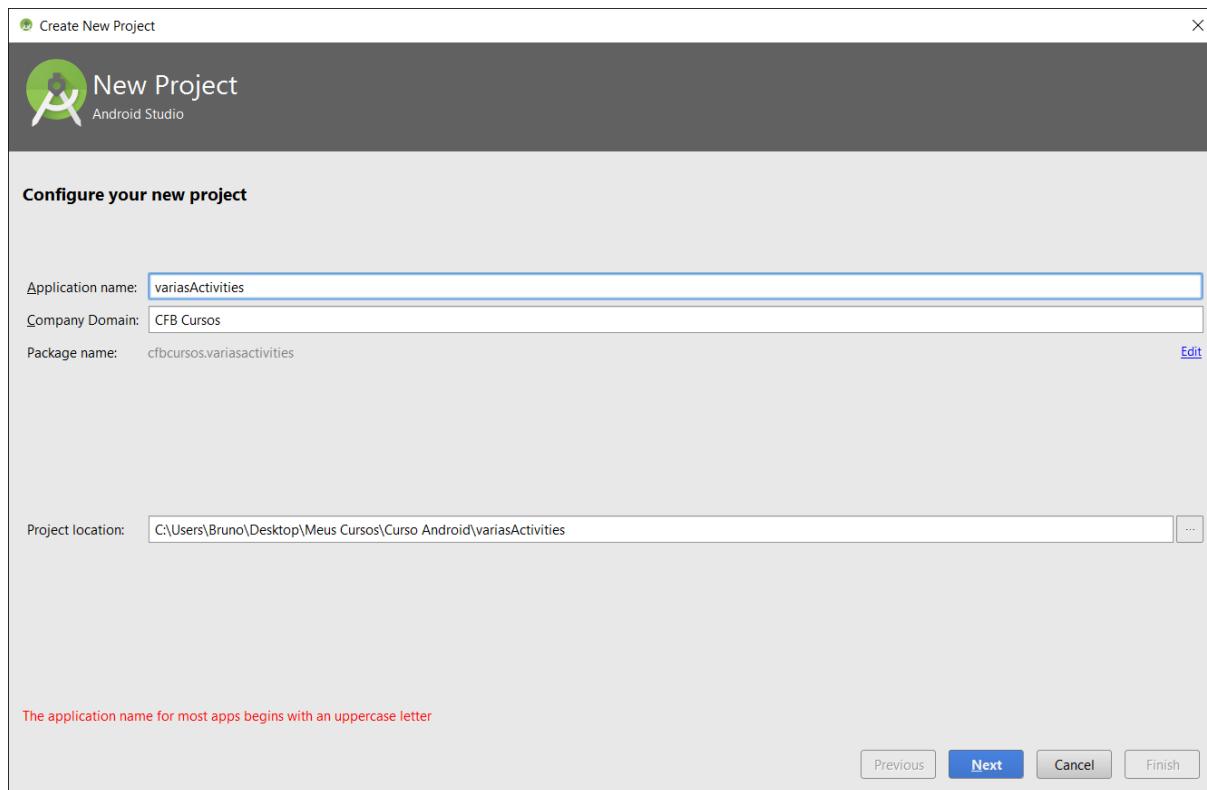
Salve as alterações e rode o aplicativo, na tela de consulta clique nos botões “Anterior” e “Próximo” para navegar entre os registros.

Trabalhando com várias Activities

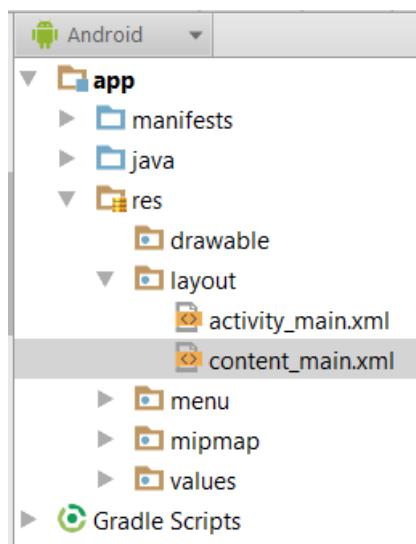
Anteriormente criamos as telas usando somente os arquivos XML, não está errado, mas acredite é um procedimento mais trabalhoso, isso porque todo o código do aplicativo, de todas as telas, ficam todos misturados no mesmo arquivo “MainActivity.java”.

Para que tenhamos mais praticidade e trabalhar de uma forma mais profissional, vamos criar uma Activity para cada tela, não se assuste com isso, pois é uma tarefa bem simples.

Vamos iniciar um novo projeto com o nome “variasActivities”.

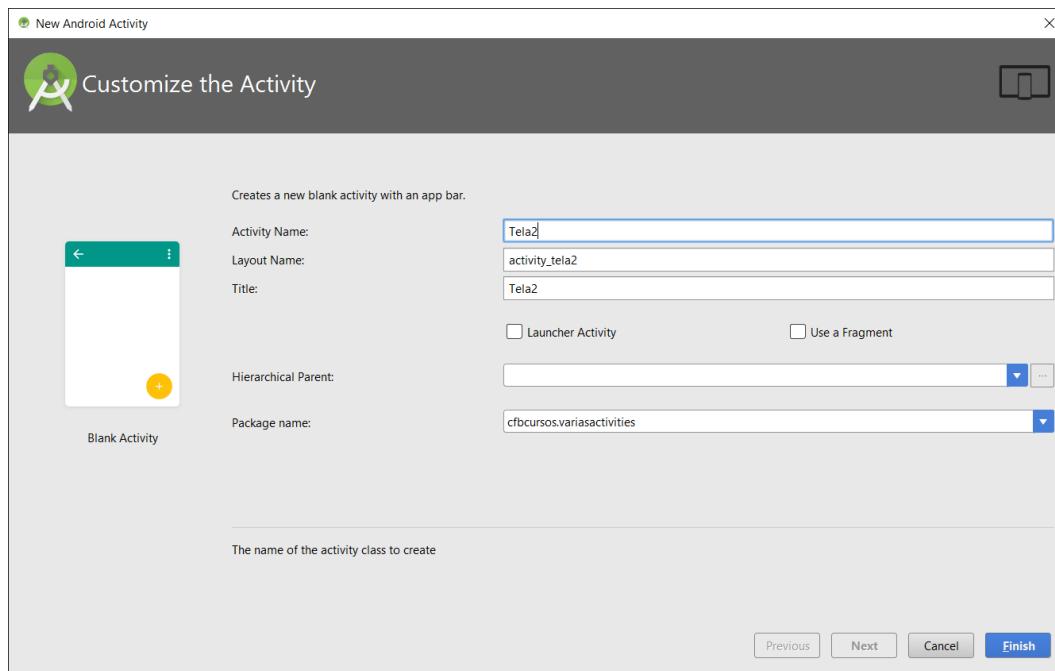


Ao iniciar o novo projeto notamos a presença da tela principal “content_main.xml”

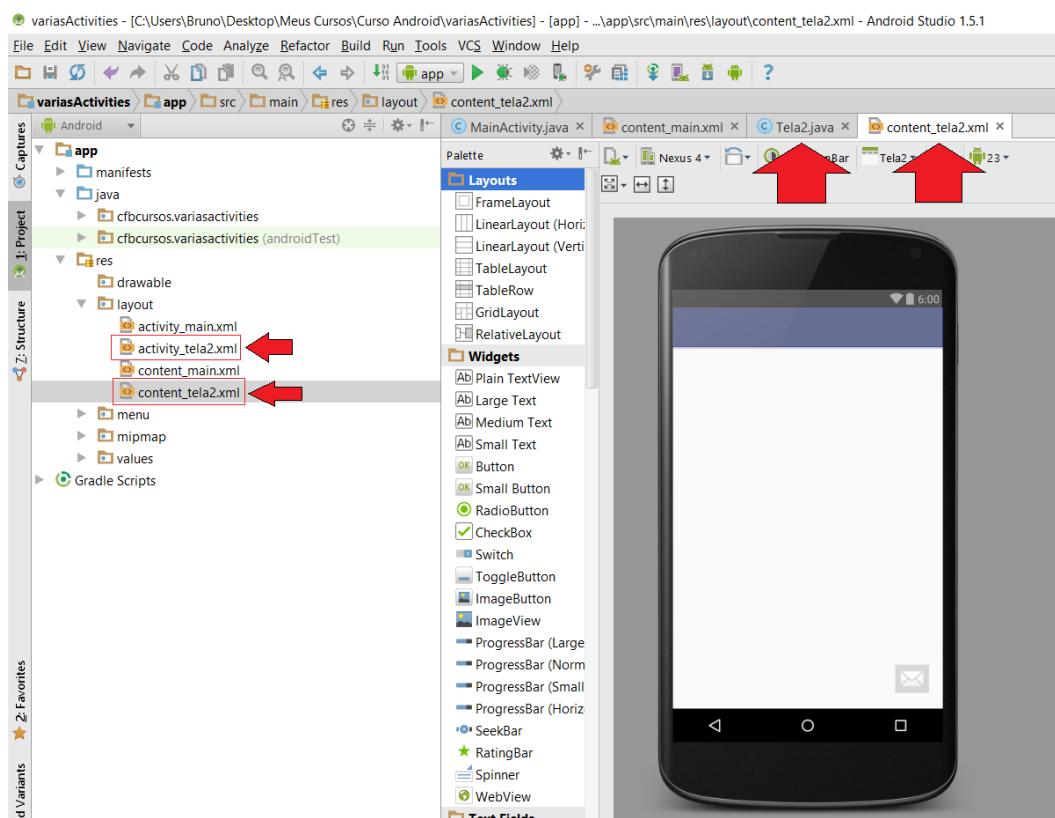


Localize a pasta Java, clique com o botão direito do mouse e selecione New -> Activity -> Basic Activity, como na ilustração a seguir.

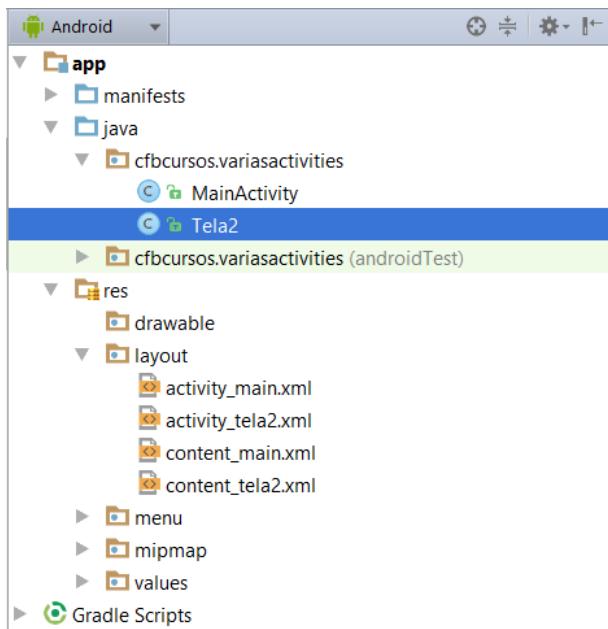
Configure o nome desta nova Activity para “Tela2” e clique em “Finish”.



Ao clicar no botão “Finish” será criada uma nova Activity com um novo arquivo XML associado a ele automaticamente, veja a ilustração a seguir apontando para os novos elementos criados.



Clicando na pasta “java” abrindo as Activities podemos ver as duas Activities “MainActivity” e “Tela2” que acabamos de criar.



Vamos configurar a primeira tela “content_main” como na ilustração a seguir.



TextView

```
layout:alignParentEnd = Marcado  
layout:alignParentStart = Marcado  
text = CFB – Tela 1  
textAlignment = center  
textSize = 30dp
```

Button

```
layout:alignParentEnd = Marcado  
layout:alignParentStart = Marcado  
id = btTela2  
text = Abrir Tela 2
```

Layout

```
background = #7299bb
```

Agora vamos configurar “Content_tela2.xml”.

**TextView**

```
layout:alignParentEnd = Marcado  
layout:alignParentStart = Marcado  
text = CFB – Tela 2  
textAlignment = center  
textSize = 30dp
```

Button

```
layout:alignParentEnd = Marcado  
layout:alignParentStart = Marcado  
id = btTela1  
text = Voltar para tela 1
```

Layout

```
background = #ded9a7
```

Com as telas configuradas agora podemos inserir a programação necessária nos botões para chamar as telas, já sabemos este procedimento.

No arquivo “MainActivity.java” adicione os comandos em destaque na ilustração a seguir.



```
3 import android.os.Bundle;
4 import android.support.design.widget.FloatingActionButton;
5 import android.support.design.widget.Snackbar;
6 import android.support.v7.app.AppCompatActivity;
7 import android.support.v7.widget.Toolbar;
8 import android.view.View;
9 import android.view.Menu;
10 import android.view.MenuItem;
11 import android.widget.*;
12 import android.content.Intent;
13
14 public class MainActivity extends AppCompatActivity {
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
21         setSupportActionBar(toolbar);
22
23         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
24         fab.setOnClickListener(view -> {
25             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
26                 .setAction("Action", null).show();
27         });
28
29
30         Button btTela2=(Button) findViewById(R.id.btTela2);
31         btTela2.setOnClickListener(new View.OnClickListener() {
32             @Override
33             public void onClick(View v) {
34                 startActivity(new Intent(MainActivity.this,Tela2.class));
35             }
36         });
37
38     }
39 }
```

No evento de clique chamamos o método “startActivity” que abre uma nova activity, basta passar como parâmetro quem está chamando no caso “MainActivity” e que está sendo chamado “Tela2.class”.

O mesmo procedimento vamos fazer na Activity Tela2.



```
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.support.design.widget.FloatingActionButton;
6 import android.support.design.widget.Snackbar;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9 import android.view.View;
10 import android.widget.*;
11
12 public class Tela2 extends AppCompatActivity {
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_tela2);
18         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
19         setSupportActionBar(toolbar);
20
21         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
22         fab.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
26                     .setAction("Action", null).show();
27             }
28         });
29
30         Button btTela1=(Button)findViewById(R.id.btTela1);
31         btTela1.setOnClickListener(new View.OnClickListener(){
32             @Override
33             public void onClick(View v){
34                 startActivity(new Intent(Tela2.this, MainActivity.class));
35                 //Parâmetros do método Intent(Origem (onde está), destino (Quem será chamado))
36             }
37         });
38 }
```

A única diferença são os parâmetros do método “startActivity”.

Salve todas as alterações e rode o programa, clique nos botões para navegar entre as telas para testar a aplicação.

Intent

Na aplicação anterior vimos o método “startActivity” que necessita como parâmetro um “Intent”, mas o que é um intent?

Basicamente um intent é uma intenção, uma descrição abstrata de uma operação a ser executada. Ele pode ser usado com “startActivity” para carregar uma Activity, com “broadcastIntent” para enviá-lo para quaisquer componentes BroadcastReceiver e “StartService(Intent)” ou “bindService(Intent, ServiceConnection, int)” para se comunicar com um serviço em segundo plano.

O uso mais significativo é no carregamento de Activities.

Outra maneira de chamar novas telas/Activities é usando o código a seguir com Intent desmenbrado.

```
Intent itTela2 = new Intent(MainActivity.this,Tela2.class);
startActivity(itTela2);
```

Passando parâmetros em chamadas de Activities usando Bundle

Podemos chamar uma Activity e passar algum valor como parâmetro para ela, basta usar o “Bundle”, um “Bundle” permite passar dados através de chaves para uma Activity chamada, basicamente é um pacote para passagem de parâmetros no estilo (key,value) / (chave,valor).

Vamos alterar nosso aplicativo de forma que iremos passar três parâmetros para o Tela2.



```
32     Button btTela2=(Button)findViewById(R.id.btTela2);
33     btTela2.setOnClickListener((v) -> {
34         Bundle parametros=new Bundle();
35         parametros.putString("canal","CFB"); // (chave, valor)
36         parametros.putString("nome","Bruno");
37         parametros.putInt("ano",2016);
38         startActivity(new Intent(MainActivity.this, Tela2.class).putExtras(parametros));
39         //Parâmetros do método Intent(Origem (onde está), destino(Quem será chamado))
40     });
41 }
42 }
43 }
44 }
```

Passamos três parâmetros para a Tela2.

Parâmetro (Key/Chave)	Valor / Value	Tipo
canal	CFB	String
nome	Bruno	String
ano	2016	Integer

Agora vamos recuperar estes parâmetro na Tela2.

```
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_tela2);
19         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
20         setSupportActionBar(toolbar);
21
22         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
23         fab.setOnClickListener((view) -> {
24             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
25                 .setAction("Action", null).show();
26         });
27
28
29
30         String vcanal,vnome;
31         Integer vano;
32
33         Intent dadosRecebidos = getIntent();
34         if(dadosRecebidos != null){
35             Bundle parametrosRecebidos = dadosRecebidos.getExtras();
36             vcanal = parametrosRecebidos.getString("canal");
37             vnome = parametrosRecebidos.getString("nome");
38             vano = parametrosRecebidos.getInt("ano");
39         }
40
41         Button btTela1=(Button)findViewById(R.id.btTela1);
42         btTela1.setOnClickListener((v) -> {
43             startActivity(new Intent(Tela2.this, MainActivity.class));
44             //Parâmetros do método Intent(Origem (onde está), destino(Quem será chamado))
45         });
46
47     });
48 }
49 }
50 }
```

Note que criamos as variáveis, duas Strings e um Integer, para guardar os valores passados, depois declaramos um Intent, verificamos se ele não está vazio, pode ser que não foi passado nenhum parâmetro, dentro do IF recuperamos os valores passados e guardamos nas variáveis.

Vamos mostrar em uma caixa de mensagem os valores passados, altere o código conforme a ilustração a seguir.



```
31     String vcanal,vnome;
32     Integer vano;
33
34     Intent dadosRecebidos = getIntent();
35     if(dadosRecebidos != null){
36         Bundle parametrosRecebidos = dadosRecebidos.getExtras();
37         vcanal = parametrosRecebidos.getString("canal");
38         vnome = parametrosRecebidos.getString("nome");
39         vano = parametrosRecebidos.getInt("ano");
40
41         AlertDialog.Builder adb = new AlertDialog.Builder(Tela2.this);
42         adb.setMessage("Canal: "+vcanal+"\nNome: "+vnome+"\nAno: "+vano.toString());
43         adb.setNeutralButton("OK",null);
44         adb.show();
45     }
46
47     Button btTela1=(Button)findViewById(R.id.btTela1);
48     btTela1.setOnClickListener(v) -> {
49         startActivity(new Intent(Tela2.this, MainActivity.class));
50         //Parâmetros do método Intent(Origem (onde está), destino (Quem será chamado))
51     });
52
53 }
```

Veja agora que ao chamar a Tela2, será mostrada uma caixa de mensagem com os valores dos parâmetros passados.



Passando parâmetros usando o próprio Intent

Outra maneira simples de passar parâmetros em chamadas de Activities é usar o próprio Intent, veja o procedimento para enviar.

```
32     Button btTela2=(Button)findViewById(R.id.btTela2);
33     btTela2.setOnClickListener(new View.OnClickListener() {
34         @Override
35         public void onClick(View v) {
36             Intent itTela2 = new Intent(MainActivity.this,Tela2.class);
37
38             itTela2.putExtra("canal","CFB");
39             itTela2.putExtra("nome","Bruno");
40             itTela2.putExtra("ano",2016);
41
42             startActivity(itTela2);
43         }
44     });
45 }
```

E a seguir, o procedimento para receber.



```
30
31     String vcanal,vnome;
32     Integer vano;
33
34     Intent dadosRecebidos = getIntent();
35     if(dadosRecebidos != null){
36         vcanal = dadosRecebidos.getStringExtra("canal");
37         vnome = dadosRecebidos.getStringExtra("nome");
38         vano = dadosRecebidos.getIntExtra("ano",0);
39
40
41         AlertDialog.Builder adb = new AlertDialog.Builder(Tela2.this);
42         adb.setMessage("Canal: "+vcanal+"\nNome: "+vnome+"\nAno: "+vano.toString());
43         adb.setNeutralButton("OK", null);
44         adb.show();
45     }
46
```

OBS: Em nossos exemplo passamos valores “fixos”, inseridos diretamente pelo código, observe também que podemos passar valores digitados em caixa de texto “EditText”, usando o seguinte esquema.

```
final EditText etNome=(EditText)findViewById(R.id.etNome);
Intent itTela2 = new Intent(MainActivity.this,Tela2.class);

itTela2.putExtra("nome",etNome.getText().toString());
```

E para inserir os valores passados como parâmetro para um TextView usamos o seguinte procedimento.

```
final TextView tvNome=(EditText)findViewById(R.id.tvNome);
Intent itTela2 = new Intent(MainActivity.this,Tela2.class);

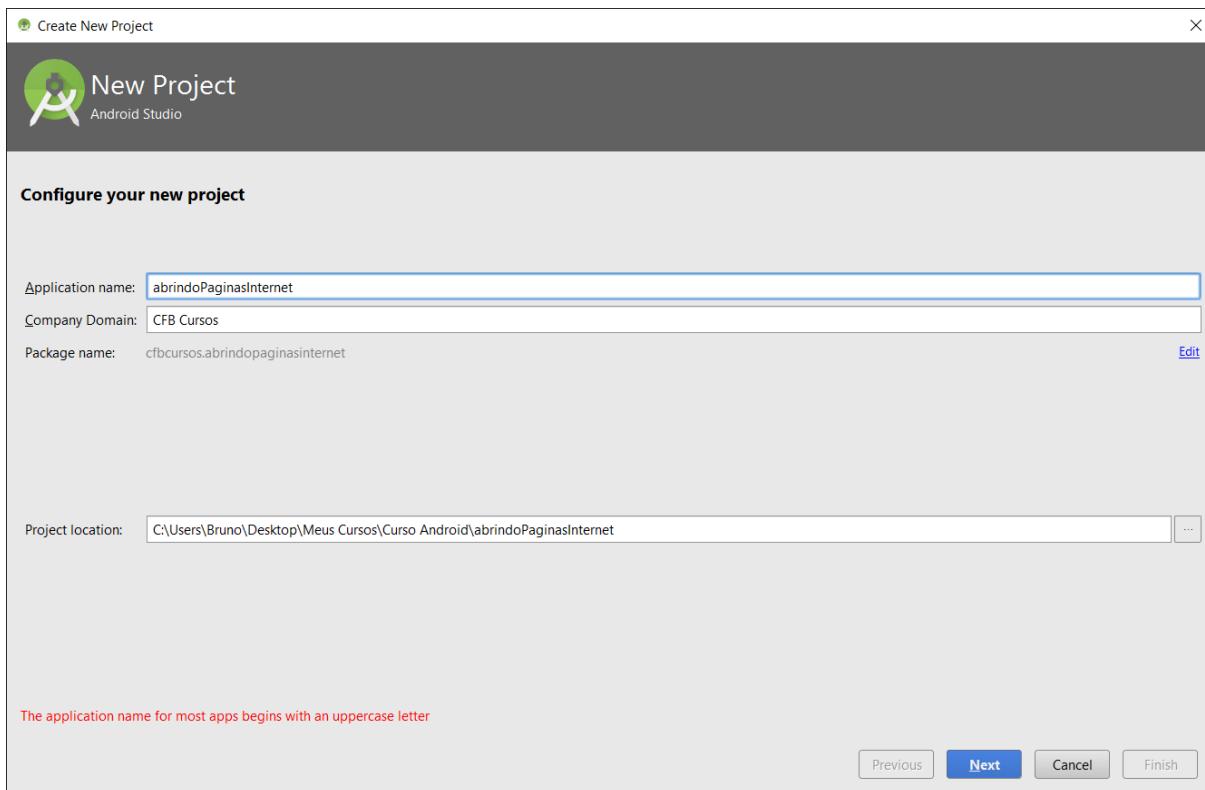
tvNome.setText("Nome: " + itTela2.getStringExtra("nome"));
```

Operações com Intent

Existem várias tarefas que podemos realizar com um Intent, por exemplo, abrir uma página web, abrir o discador, manipular os contatos, etc, neste capítulo vamos ver algumas das operações que podemos realizar com Intent.

Acessando página da Internet

Vamos aprender como usar Intent para acessar páginas de internet, é um procedimento muito simples, inicie um novo projeto com nome “abrindoPaginasInternet”.



Ao iniciar o projeto abra o arquivo “MainActivity.java” e vamos ao código.

```
9 import android.view.Menu;
10 import android.view.MenuItem;
11 import android.content.Intent;
12 import android.net.Uri;
13
14 public class MainActivity extends AppCompatActivity {
15
16     public void abrirWeb(){
17         Uri uri = Uri.parse("http://cfbcursos.com.br");
18         Intent intent = new Intent(Intent.ACTION_VIEW, uri);
19         startActivity(intent);
20     }
21
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_main);
26         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
27         setSupportActionBar(toolbar);
28
29         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
30         fab.setOnClickListener(view) -> {
31             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
32                 .setAction("Action", null).show();
33         };
34
35         abrirWeb();
36     }
37
38 }
39 }
```

Primeiramente fizemos o import das bibliotecas necessárias.

Das linhas 16 a 20 construímos o método para abrir o browser e a página de Internet.

Criamos um objeto do tipo URI (Uniform Resource Identifier) com a URL do site que será aberto, depois simplesmente passamos para o Intent.



O último comando foi o método `startActivity` para chamar a Intent com a URL especificada.

Na linha 38 simplesmente chamamos o método criado.



Abrindo o discador com um número de contato da agenda

Para chamar o discador basta usar a ação `ACTION_DIAL` de uma Intent, não é difícil, na verdade é extremamente simples, da mesma forma que a aplicação anterior iremos criar um método para ligar para um determinado número na agenda de contatos.

Inicie uma nova aplicação no Android com nome “ligandoParaContato”.

Vamos ao código.

```
10 import android.view.Menu;
11 import android.view.MenuItem;
12 import android.content.Intent;
13
14 public class MainActivity extends AppCompatActivity {
15
16     public void ligacao() {
17
18         Uri uriLiga = Uri.parse("tel:1111");
19         Intent itLiga = new Intent(Intent.ACTION_DIAL, uriLiga);
20         startActivity(itLiga);
21     }
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
28         setSupportActionBar(toolbar);
29
30         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
31         fab.setOnClickListener(new View.OnClickListener() {
32             @Override
33             public void onClick(View view) {
34                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
35                     .setAction("Action", null).show();
36             }
37         });
38
39         ligacao();
40     }
41 }
```



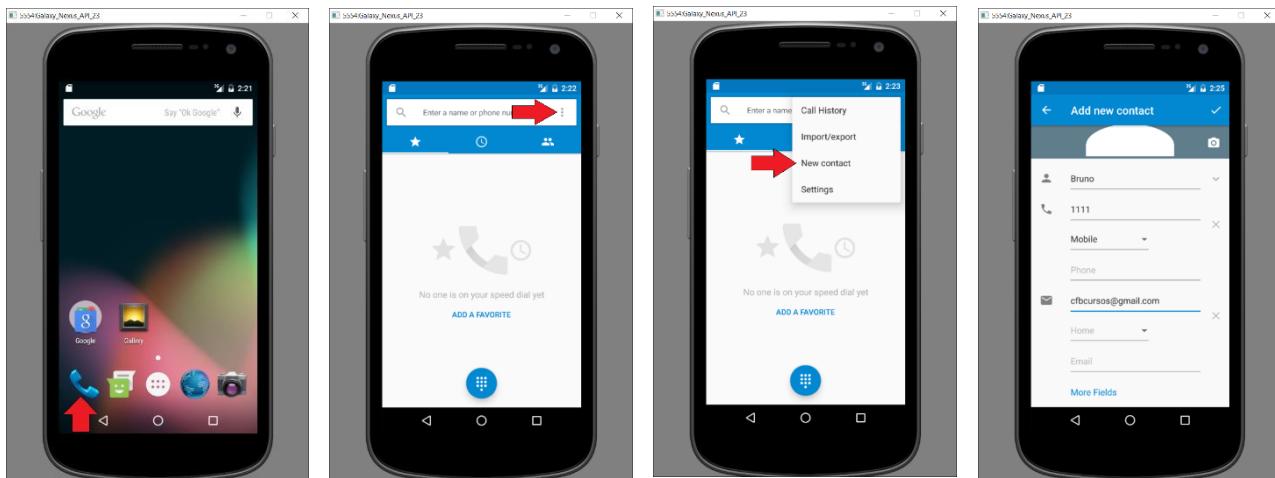
Note que importamos a biblioteca Intent na linha 12.

Das linhas 16 a 21 criamos o método que chama a ação “ACTION_DIAL” especificamente na linha 19.

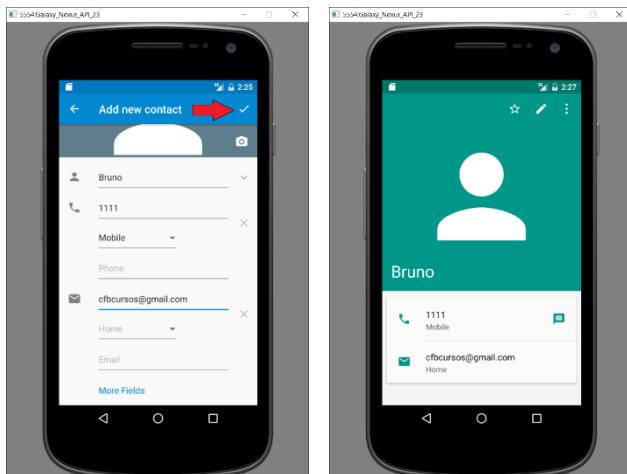
Na linha 39 simplesmente chamamos o método “ligacao()” para ser executado.

O número de telefone “1111” não será encontrado, então, minimize o aplicativo e vamos adicionar o número na agenda.

Depois clique no ícone do telefone azul.

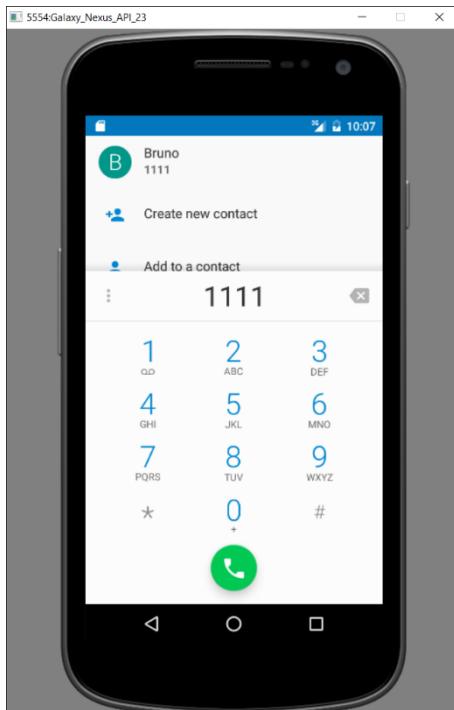


Após digitar as informações do contato conforme a ilustração clique em concluir.



Pronto, agora já temos um contato adicionado em nossa agenda.

Roda novamente o programa e veja que o discador será aberto já com o contato selecionado e o discador aberto com o número digitado.

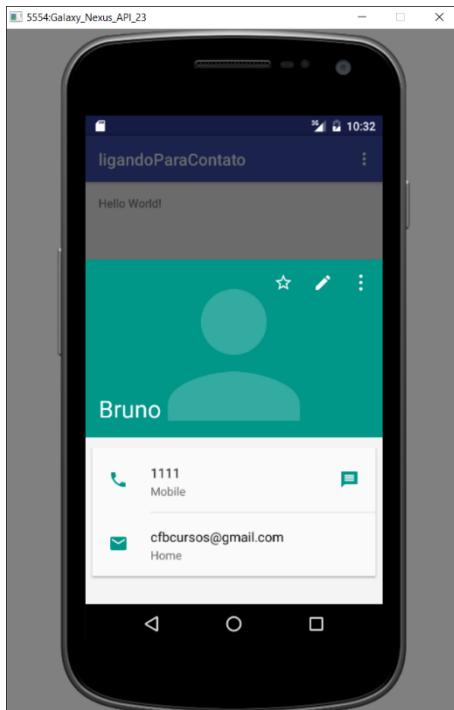


Abrir um determinado contato e ver todas suas informações

Para abrir as informações de um determinado contato da agenda do celular, devemos usar a ação “ACTION_VIEW”, vamos ao código.

```
10 import android.view.Menu;
11 import android.view.MenuItem;
12 import android.content.Intent;
13
14 public class MainActivity extends AppCompatActivity {
15
16     public void lerContato(){
17         Uri uriContato = Uri.parse("content://com.android.contacts/contacts/1");
18         Intent itContato = new Intent(Intent.ACTION_VIEW,uriContato);
19         startActivity(itContato);
20     }
21
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_main);
26         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
27         setSupportActionBar(toolbar);
28
29         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
30         fab.setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View view) {
33                 Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
34                     .setAction("Action", null).show();
35             }
36         });
37
38         lerContato();
39
40     }
}
```

Na linha 17 temos o número 1 no final da string, este é o número do contato na agenda, significa que queremos abrir as informações do primeiro contato na agenda.



Obtendo o nome de um contato da agenda

Para obter o nome da agenda temos um procedimento um pouco mais elaborado, vamos precisar de um “Cursor”, já sabemos sobre este elemento.

Veja o código a seguir.

```
10 import android.support.v7.widget.Toolbar;
11 import android.view.View;
12 import android.view.Menu;
13 import android.view.MenuItem;
14 import android.content.Intent;
15 import android.database.Cursor;
16
17 public class MainActivity extends AppCompatActivity {
18
19     public void msg(String txt){
20         AlertDialog.Builder adb = new AlertDialog.Builder(MainActivity.this);
21         adb.setMessage(txt);
22         adb.setNeutralButton("OK", null);
23         adb.show();
24     }
25
26     public void obterInfoContato(){
27         int infoCont=0;
28         Uri uriInfoContato = Uri.parse("content://com.android.contacts/contacts/");
29         Intent itInfoContato = new Intent(Intent.ACTION_PICK,uriInfoContato);
30         startActivityForResult(itInfoContato, infoCont);
31     }
32
33     protected void onActivityResult(int cod, int res, Intent it){
34         Uri vres = it.getData();
35         Cursor cr = getContentResolver().query(vres, null, null, null, null);
36         cr.moveToFirst();
37         int inome=cr.getColumnIndexOrThrow(ContactsContract.Contacts.DISPLAY_NAME);
38         String nome=cr.getString(inome);
39         msg("Nome: "+nome);
40     }
41
42     @Override
43     protected void onCreate(Bundle savedInstanceState) {
44         super.onCreate(savedInstanceState);
```



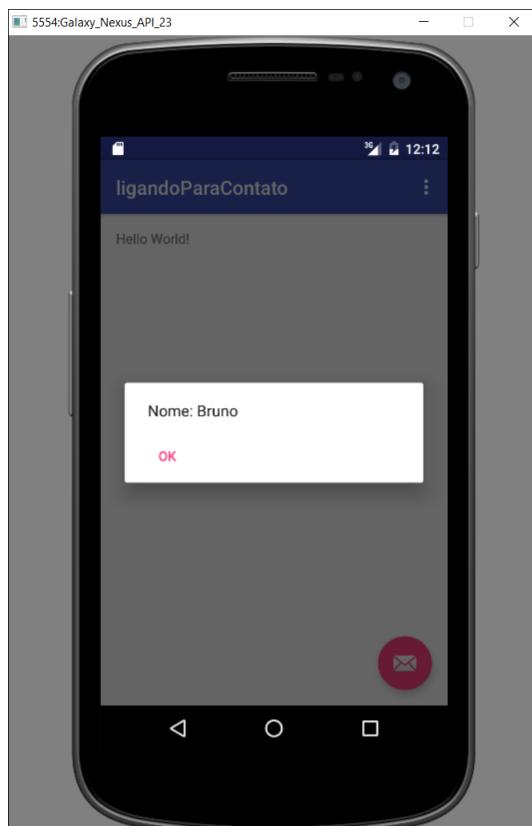
Note que usamos um método já existente “onActivityResult”.

Neste método montamos o procedimento que obtém o nome do contato “DISPLAY_NAME”.

Em seguida chamamos o método.

```
43 protected void onCreate(Bundle savedInstanceState) {  
44     super.onCreate(savedInstanceState);  
45     setContentView(R.layout.activity_main);  
46     Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
47     setSupportActionBar(toolbar);  
48  
49     FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
50     fab.setOnClickListener(new View.OnClickListener() {  
51         @Override  
52         public void onClick(View view) {  
53             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)  
54                 .setAction("Action", null).show();  
55         }  
56     });  
57  
58     obterInfoContato();  
59  
60 }
```

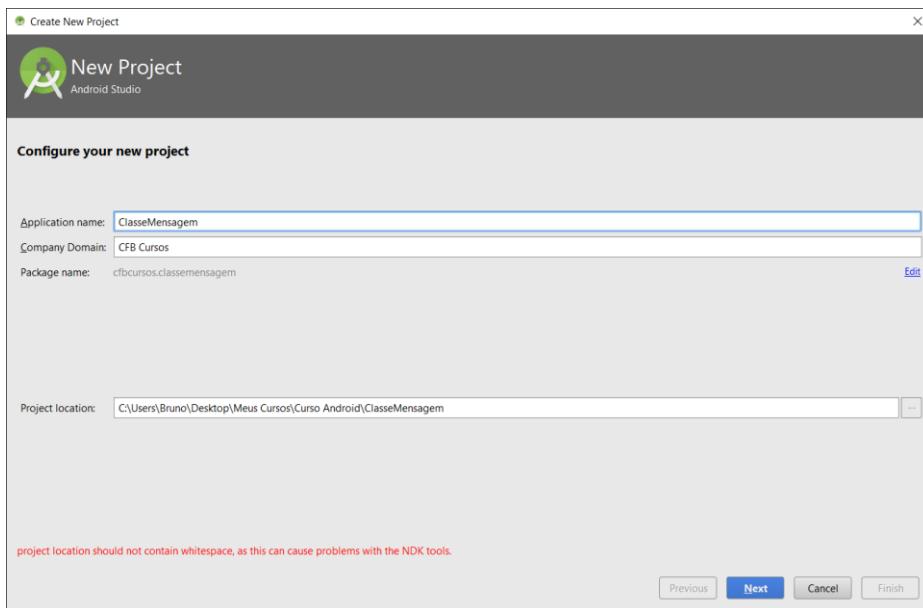
Ao rodar o programa, clique em um dos contatos da agenda e veja que será mostrada uma caixa de mensagem com o nome do contato.



Criando um classe para caixas de mensagem

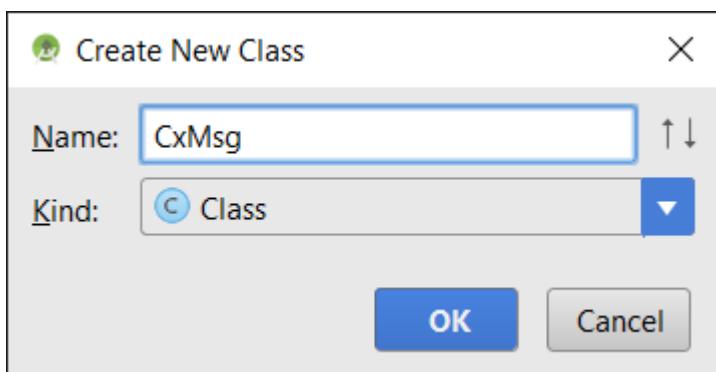
Iremos aprender como definir toda a rotina para caixas de mensagem padrão em uma classe, isso irá facilitar nossa vida, pois iremos definir a classe uma única vez em nosso projeto e usar quantas vezes for necessário.

Vamos criar um novo projeto com nome “ClasseMensagem”.



Já sabemos como criar uma nova classe, então clique na aba do código principal “MainActivity.java” e clique em FILE – NEW – Java Class.

Nossa classe irá se chamar “CxMsg”.



Na classe “CxMsg” digite o código para criar a caixa de mensagem como na ilustração a seguir.

```
package cfbcursos.classemensagem;

import android.app.Activity;
import android.app.AlertDialog;

/**
 * Created by Bruno on 30/03/2016.
 */
public class CxMsg {

    public void mostra(String txt, Activity act) {
        AlertDialog.Builder adb = new AlertDialog.Builder(act);
        adb.setMessage(txt);
        adb.setNeutralButton("OK", null);
        adb.show();
    }
}
```



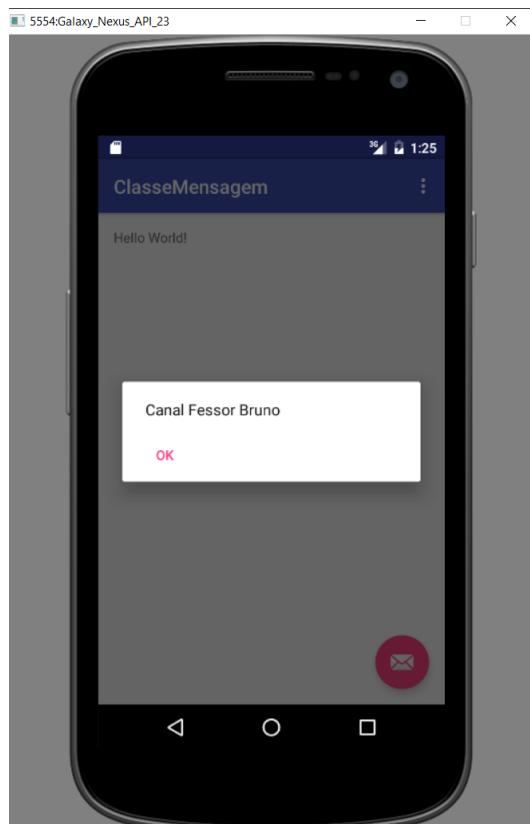
Pronto, quando precisar, basta declarar um objeto do tipo “CxMsg” uma única vez no arquivo e sempre que for mostrar a caixa de mensagem basta chamar o método “mostra” passando a mensagem e a Activity como parâmetros.

Voltando ao código “MainActivity.java”, vamos declarar um objeto com nome “msg” do tipo “CxMsg” na linha 14.

Na linha 32 chamamos o método mostra.

```
10 import android.view.MenuItem;
11
12 public class MainActivity extends AppCompatActivity {
13
14     CxMsg msg=new CxMsg();
15
16     @Override
17     protected void onCreate(Bundle savedInstanceState) {
18         super.onCreate(savedInstanceState);
19         setContentView(R.layout.activity_main);
20         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
21         setSupportActionBar(toolbar);
22
23         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
24         fab.setOnClickListener((view) -> {
25             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
26                 .setAction("Action", null).show();
27         });
28
29
30         msg.mostra("Canal Fessor Bruno", MainActivity.this);
31     }
32
33     @Override
34     public boolean onCreateOptionsMenu(Menu menu) {
35
36     }
```

Veja o resultado.



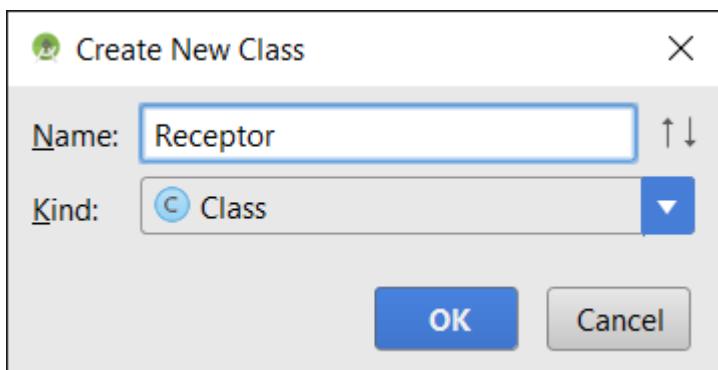


Aplicação em segundo plano

Vamos iniciar um novo projeto no Android Studio com nome “SegundoPlano”.

Para criar aplicações que executem tarefas em segundo plano “sem que seja perceptível ao usuário” precisamos trabalhar com “BroadcastReceiver”.

Nossa primeira tarefa será criar a classe com a rotina de programação que nosso aplicativo irá executar em segundo plano, então clique no menu FILE - NEW – Java Class, nossa nova classe terá o nome “Receptor”.



A nova classe será criada, vamos realizar alguns imports de bibliotecas e inserir o comportamento deste “BroadcastReceiver” de acordo com a ilustração a seguir.

```
1 package cfbcursos.segundoplano;
2
3 import android.content.BroadcastReceiver;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.widget.Toast;
7
8 /**
9 * Created by Bruno on 30/03/2016.
10 */
11 public class Receptor extends BroadcastReceiver {
12
13     @Override
14     public void onReceive(Context context, Intent intent) {
15         Toast.makeText(context, "Canal Fessor Bruno", Toast.LENGTH_LONG).show();
16     }
17 }
18
19 }
```

Nossa aplicação de segundo plano irá mostrar uma mensagem “Toast” para que possamos ver que aplicação será executada, então, na linha 15 iremos declarar um objeto do tipo Toast com a mensagem “Canal Fessor Bruno” e iremos configurar o tempo de exibição como longo “LENGTH_LONG”.

Com o receiver pronto, agora precisamos registrar e chamar este receiver em nosso programa.

No arquivo “MainActivity.java”, vamos realizar os imports necessários, na linha 32 iremos registrar este receiver e na linha 33 iremos executar este receiver, embora a execução seja em segundo plano, iremos notar que foi executada pela exibição da mensagem “Canal Fessor Bruno”.



```
1 package cfbcursos.segundoplano;
2
3     import android.content.IntentFilter;
4     import android.os.Bundle;
5     import android.support.design.widget.FloatingActionButton;
6     import android.support.design.widget.Snackbar;
7     import android.support.v7.app.AppCompatActivity;
8     import android.support.v7.widget.Toolbar;
9     import android.view.View;
10    import android.view.Menu;
11    import android.view.MenuItem;
12    import android.content.Intent;
13
14    public class MainActivity extends AppCompatActivity {
15
16        @Override
17        protected void onCreate(Bundle savedInstanceState) {
18            super.onCreate(savedInstanceState);
19            setContentView(R.layout.activity_main);
20            Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
21            setSupportActionBar(toolbar);
22
23            FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
24            fab.setOnClickListener(new View.OnClickListener() {
25                @Override
26                public void onClick(View view) {
27                    Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
28                        .setAction("Action", null).show();
29                }
30            });
31
32            registerReceiver(new Receptor(), new IntentFilter("executar_receptor"));
33            sendBroadcast(new Intent("executar_receptor"));
34        }
35
36        @Override
```

Veja a execução do programa, exibindo a mensagem Toast.



OBS: Desta maneira que fizemos nosso conteúdo de segundo plano só irá executar caso a aplicação esteja rodando.



Outra maneira de criar um “BroadcastReceiver” para rodar em segundo plano é inserindo no “AndroidManifest.xml”, neste caso, nossa aplicação de segundo plano poderá ser executada mesmo que a classe não tenha sido chamada por nossa aplicação e mesmo que a aplicação nem esteja sendo executada.

Abra o arquivo “AndroidManifest.xml” e insira o receiver conforme a ilustração a seguir.

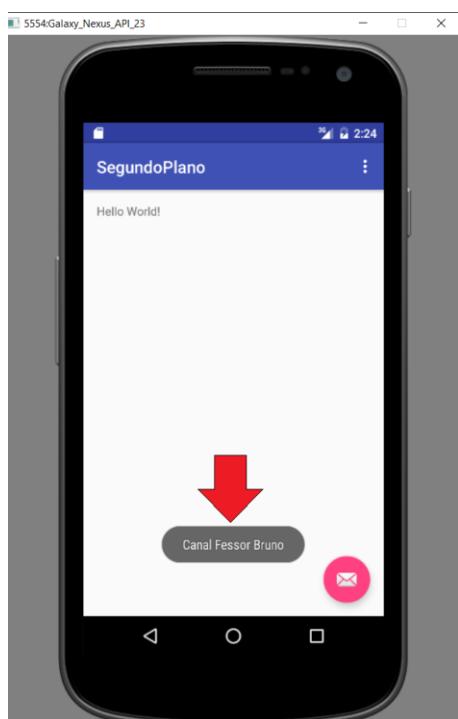
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cfbcursos.segundoplano">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="SegundoPlano"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="SegundoPlano"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".Receptor">
            <intent-filter>
                <action android:name="exec_receptor"/>
                <category android:name="android.intent.category.DEFAULT"/>
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

Voltando em nosso programa principal “MainActivity.java”, vamos comentar o código anterior e inserir o novo código na linha 34 de acordo com a ilustração a seguir.

```
30
31
32     //registerReceiver(new Receptor(), new IntentFilter("executar_receptor"));
33     //sendBroadcast(new Intent("executar_receptor"));
34
35     sendBroadcast(new Intent("exec_receptor"));
36
37
38     @Override
```

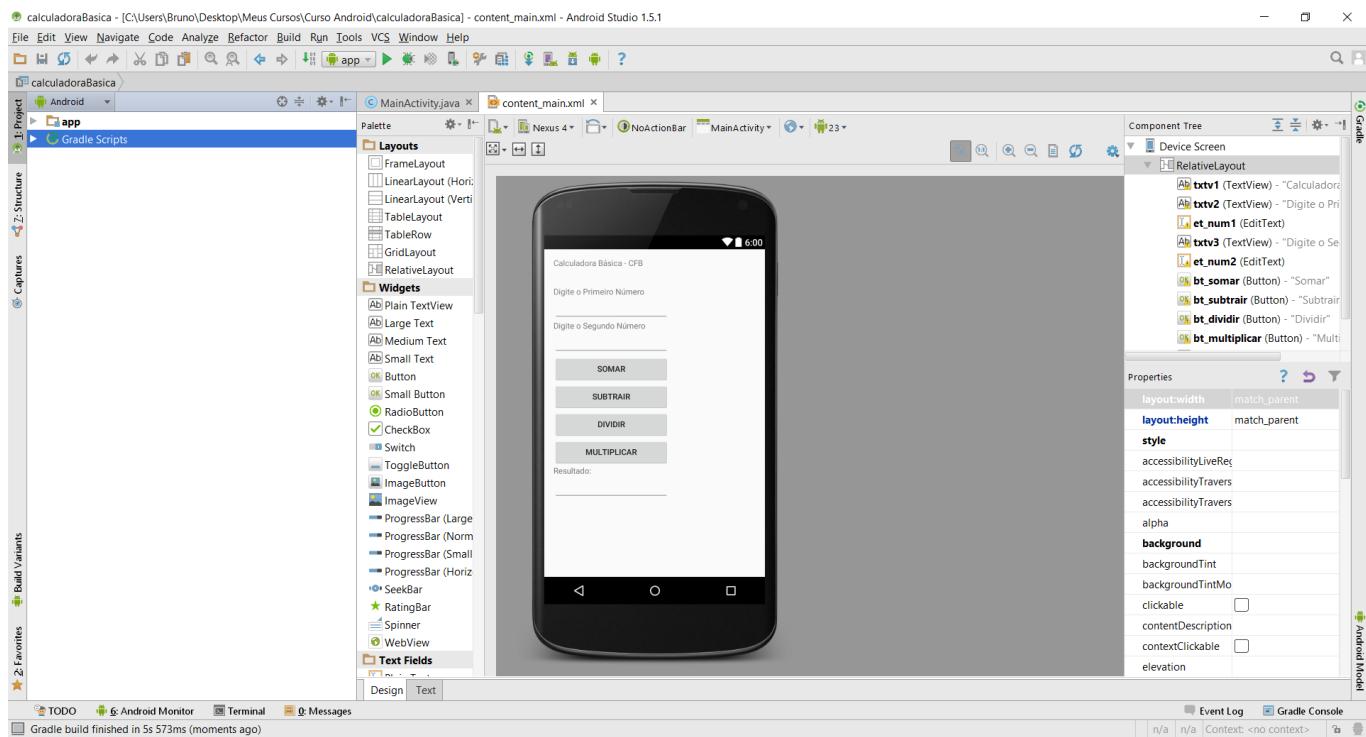
Salve todas as alterações e rode a aplicação, veja que em nosso caso, o resultado final será o mesmo.



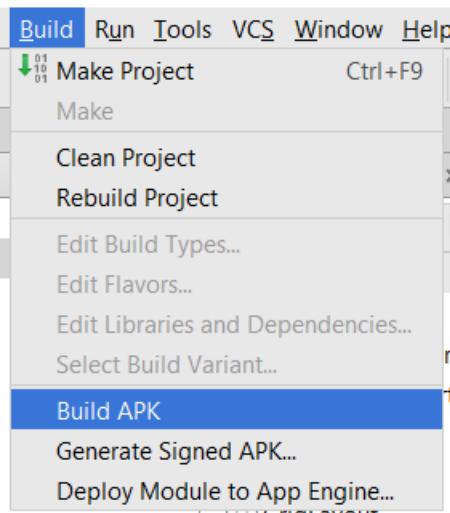


Gerando o APK (não assinado) e instalando no Telefone

Vamos abrir nosso aplicativo da calculadora básica.



Gerar o instalador “APK” é bem simples, basta clicar no menu BUILD – BUILD APK.



Quando o processo terminar será mostrada essa pequena caixa de mensagem informando que o arquivo APK foi gerado com sucesso, clique no link “Show in Explorer” para que seja aberta a pasta onde o arquivo foi gerado.



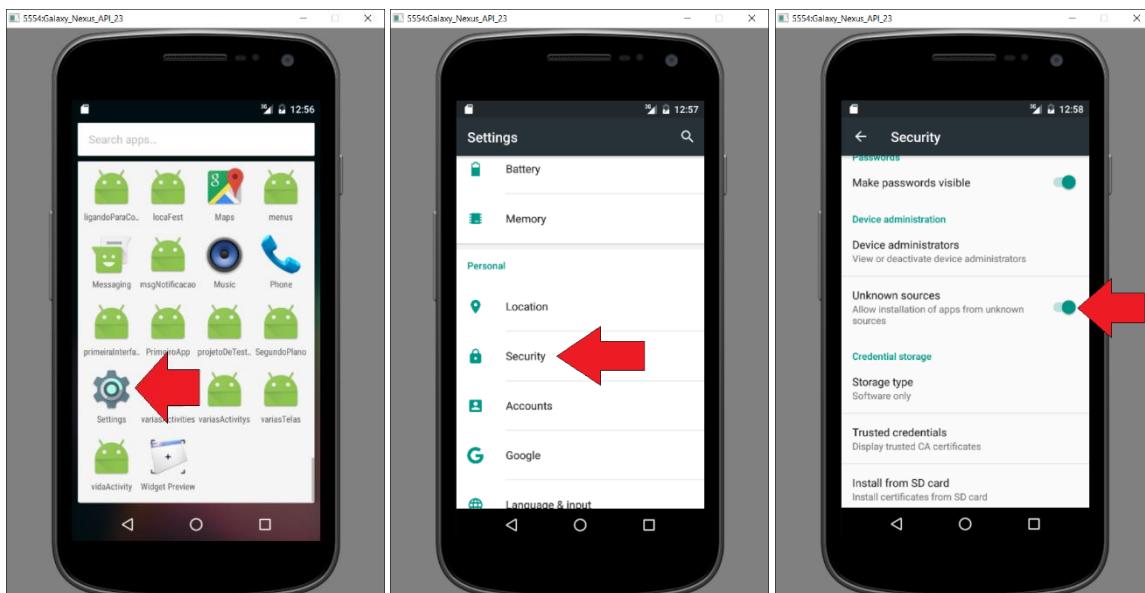
Geralmente o APK é gerado na pasta APK dentro do projeto, no seguinte caminho (calculadoraBasica\app\build\outputs\apk).



Agora, você precisa copiar o arquivo “app-debug.apk” para alguma pasta em seu celular para que possa realizar a instalação.

Como geramos um APK sem assinatura, precisamos configurar nosso celular para aceitar a instalação do nosso aplicativo.

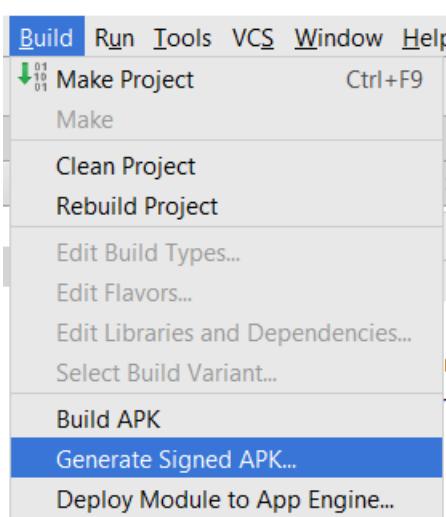
Clique em AJUSTES (Settings) – SEGURANÇA (Security) – FONTES DESCONHECIDAS (Unknown sources).



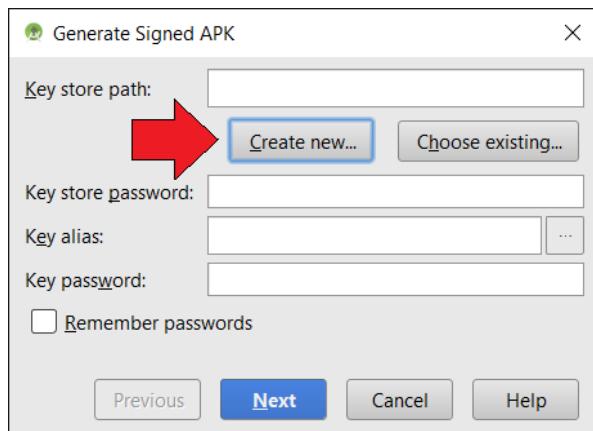
Marque esta opção para que seu Android possa aceitar instalações de APKs não assinados, depois basta navegar até a pasta que você copiou o APK e iniciar a instalação do aplicativo.

Criando uma assinatura para seus aplicativos

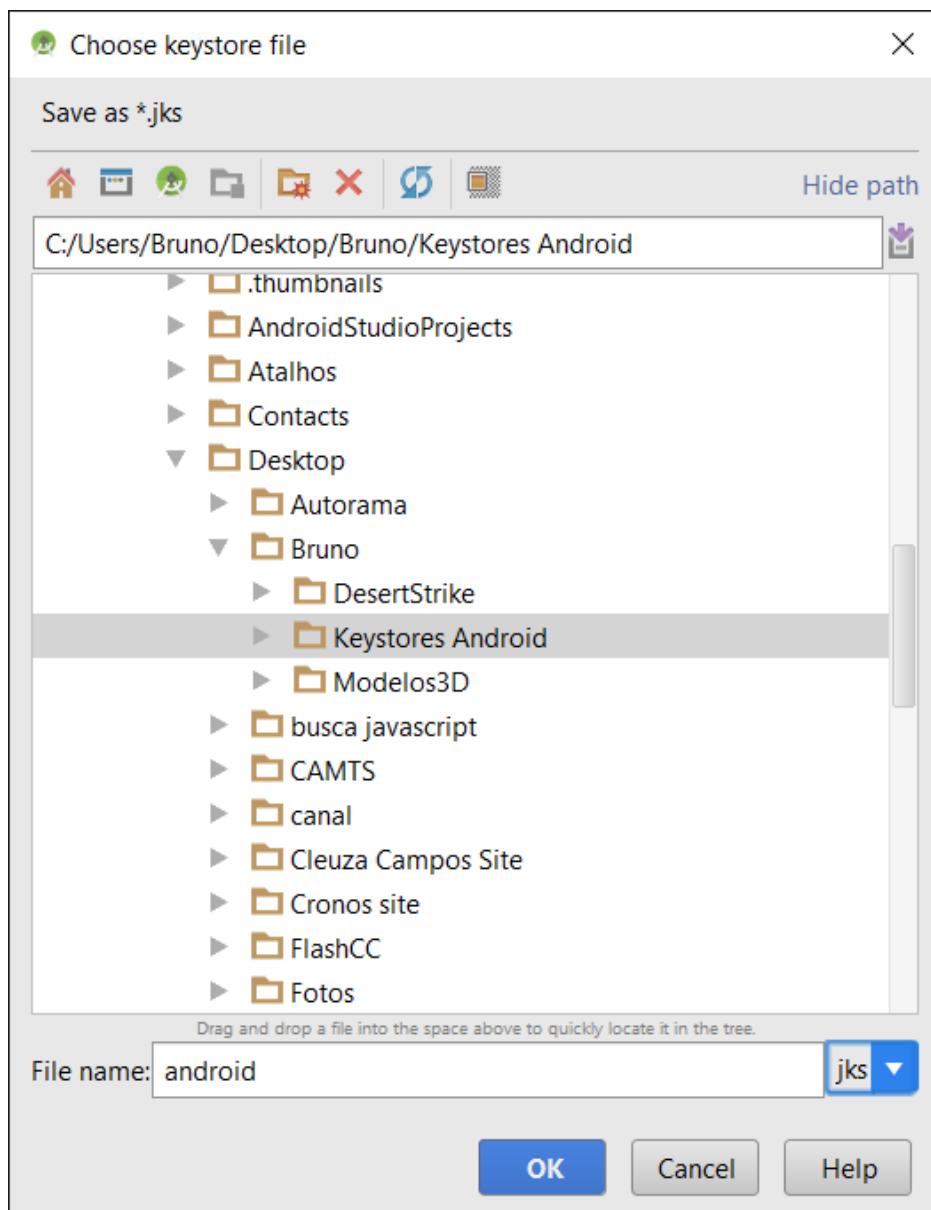
Clique no menu BUILD – GENERATE SIGNER APK.



Em seguida clique no botão “Create new...”.



Em seguida indique a pasta onde a Keystore irá ser criada e nome da sua Keystore.



Clicando em OK, preencha a tela com as informações da sua Keystore conforme modelo a seguir.



New Key Store

Key store path: \Bruno\Desktop\bruno\Keystores Android\android.jks ...

Password: Confirm:

Key

Alias: AndoidKeystore

Password: Confirm:

Validity (years): 25

Certificate

First and Last Name: Bruno P. Campos

Organizational Unit: CFB - Curso Android

Organization: Canal Fessor Bruno

City or Locality: Belo Horizonte

State or Province: Minas Gerais

Country Code (XX): 55

OK Cancel

Clique em OK, preencha as caixas de senha e Alias e clique em Next.

Generate Signed APK

Key store path: ctop\bruno\Keystores Android\android.jks

Create new... Choose existing...

Key store password:

Key alias: AndoidKeystore

Key password:

Remember passwords

Previous Next Cancel Help

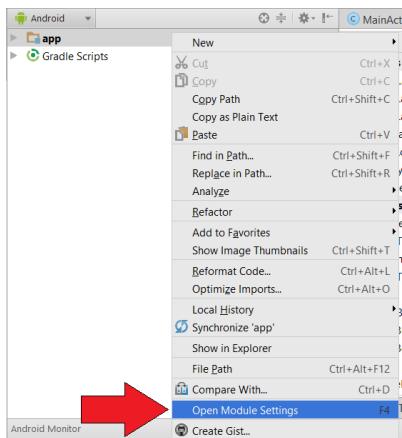
Na próxima janela clique em FINISH e aguarde o processo.



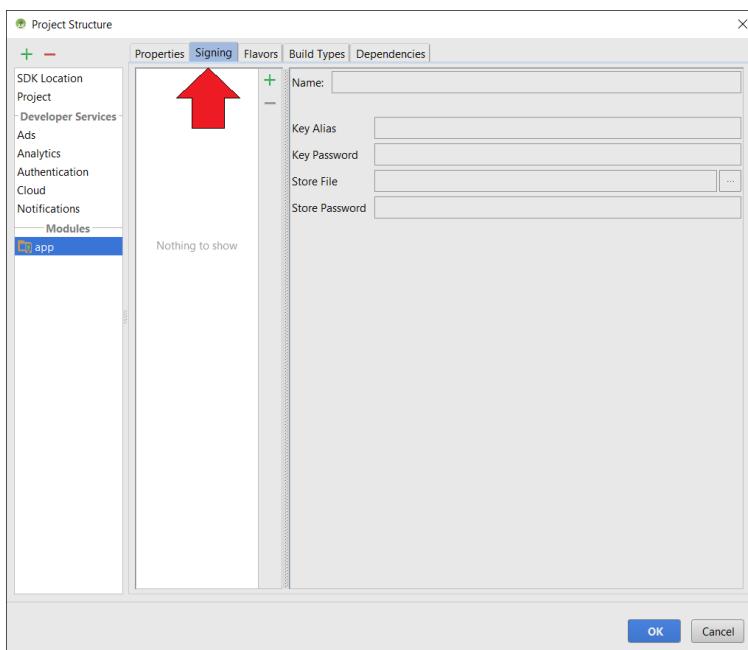
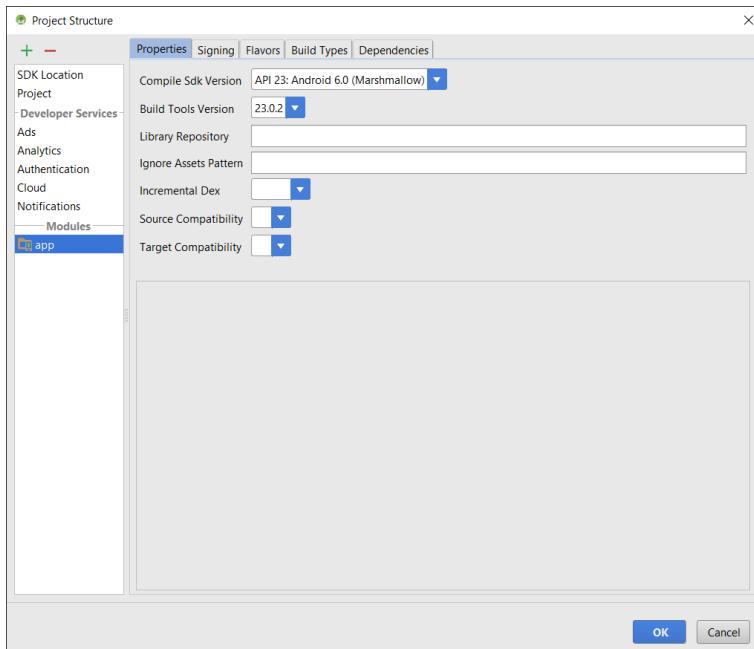
Após o término podemos assinar nossos aplicativos de forma fácil.

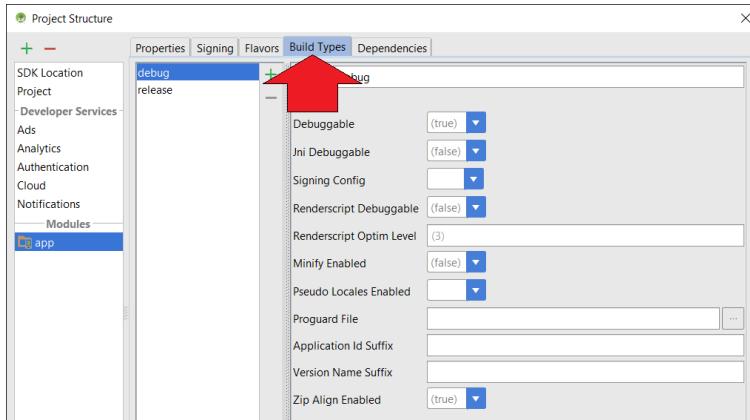
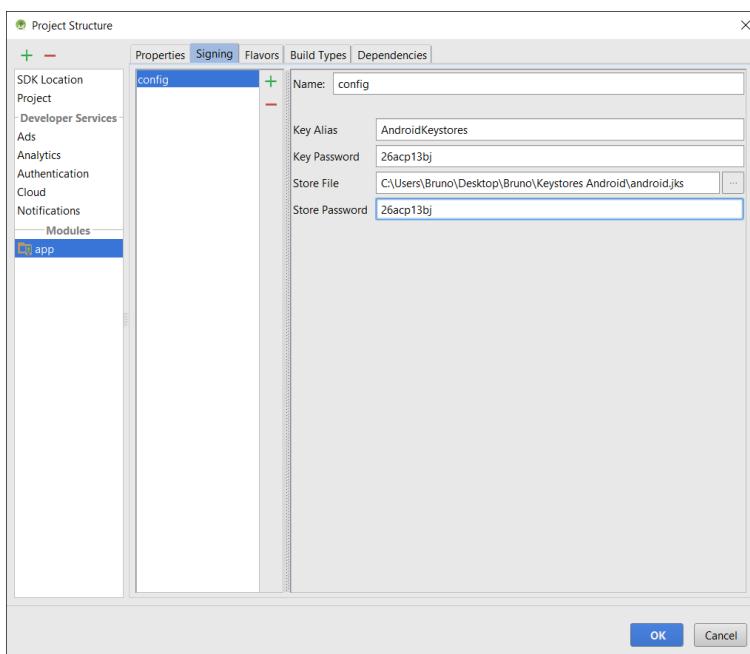
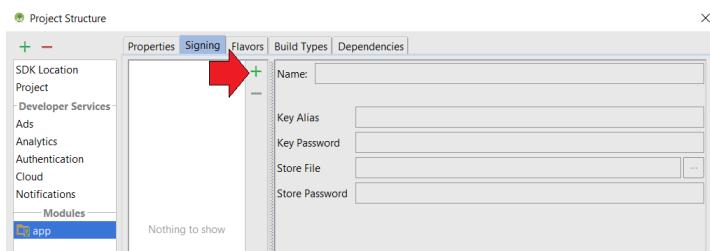


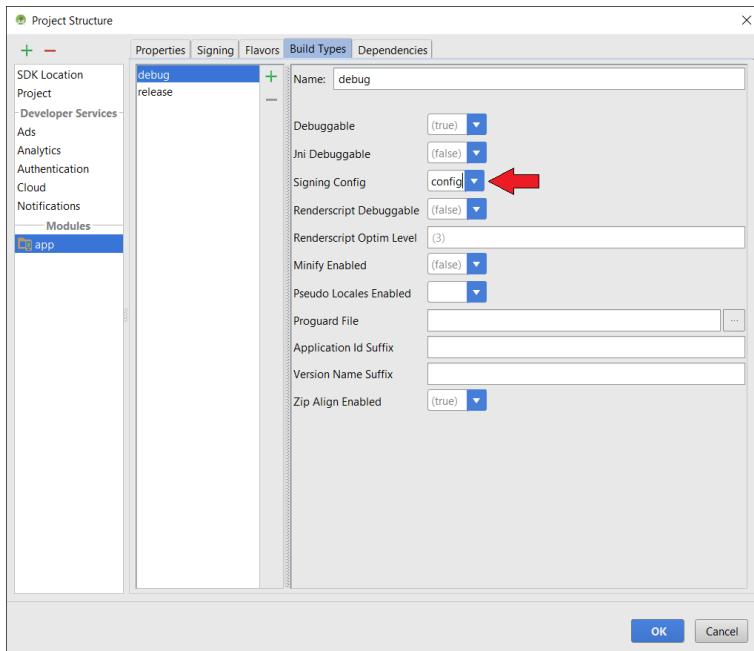
Clique com o botão direito do mouse sobre o App e selecione “Open Module Settings”.



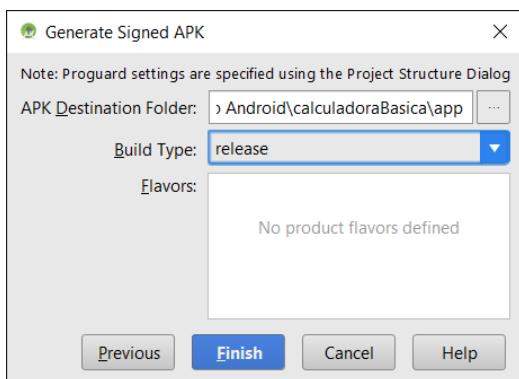
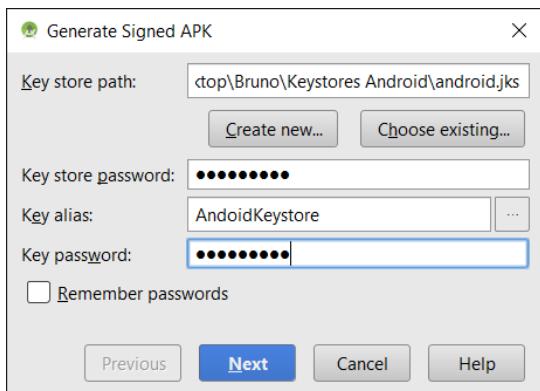
Siga as ilustrações a seguir.







Clique em OK.



Clicando em FINISH será gerado o arquivo “app-release-unaligned.apk”, pronto, agora temos uma instalação de aplicativo com assinatura.

Publicar o aplicativo na Google Play

Para publicar seus aplicativos na Google Play você precisa concluir o processo de assinatura anterior.

Acesse o endereço a seguir.

play.google.com/apps/publish



Faça login com sua conta do Google, leia os termos do “Contrato de distribuição do desenvolvedor do Google Play”, selecione abaixo indicando que concorda com o contrato.

Google Play Developer Console

Fazer login com a Conta do Google Aceitar o Contrato do Desenvolvedor Pagar taxa de registro Fornecer os detalhes da conta

VOCÊ ESTÁ CONECTADO COMO...

Bruno Campos canalfessorbruno@gmail.com Esta é a Conta do Google que será associada ao seu Developer Console. Para usar uma conta diferente, escolha uma das opções abaixo. Caso represente uma organização, convém registrar uma nova Conta do Google em vez de usar uma conta pessoal.

Fazer login com uma conta diferente Criar uma nova Conta do Google

ANTES DE CONTINUAR...

Leia e concorde com o [Contrato de distribuição do desenvolvedor do Google Play](#). Consulte os países de distribuição onde é possível distribuir e vender apps. Tenha seu cartão de crédito em mãos para pagar a taxa de registro de US\$ 25 na próxima etapa.

Concordo e estou disposto a associar meu registro de conta com o Contrato de distribuição do desenvolvedor do Google Play.

[Continuar pagamento](#)

RECURSOS ÚTEIS DO ANDROID PRECISA DE AJUDA?

Android Developers Central de Ajuda
Android Design Entre em contato com o suporte
Android.com

© 2016 Google - [Termos de Serviço do Google Play](#) - [Política de Privacidade](#) -

Você terá que pagar uma taxa de 25 dólares, informar os dados para pagamento e prosseguir para a finalização do cadastro.

Conclua sua compra

Google Play Developer Registration Fee US\$25,00 Quantidade: 1

Adic novo cartão de crédito

Número do cartão # VISA MasterCard AMEX DISCOVER JCB O número do cartão é obrigatório

MM / AA CVC

Nome do titular do cartão Bruno Campos

Endereço de faturamento

Ao continuar, você cria uma conta do Google Payments e concorda com [Termos de Serviço - Comprador \(BR\)](#) e [Aviso de privacidade](#).

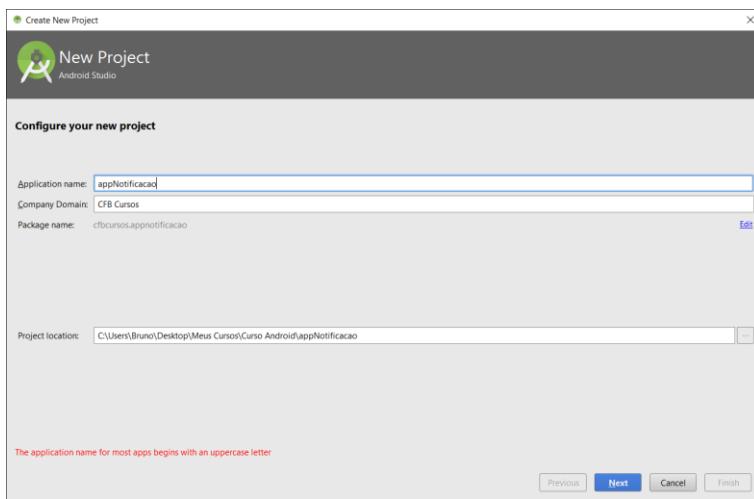
[PAY](#)

Depois você irá fornecer os dados da conta e enfim estará apto a publicar seus aplicativos.

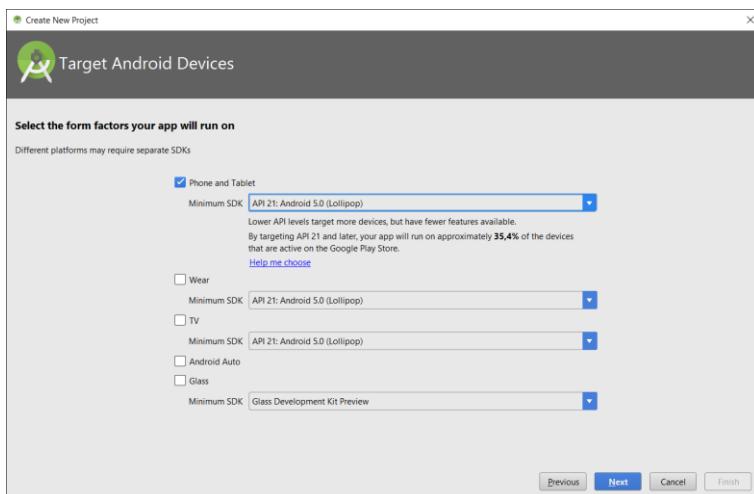


Notificações

Vamos iniciar um novo projeto com nome “appNotificacao”.



Clicando em “Next” e selecionar “API 21 Android 5.0 (Lollipop)”, desta maneira este aplicativo terá dificuldades de rodar em dispositivos que tiverem Android inferiores.

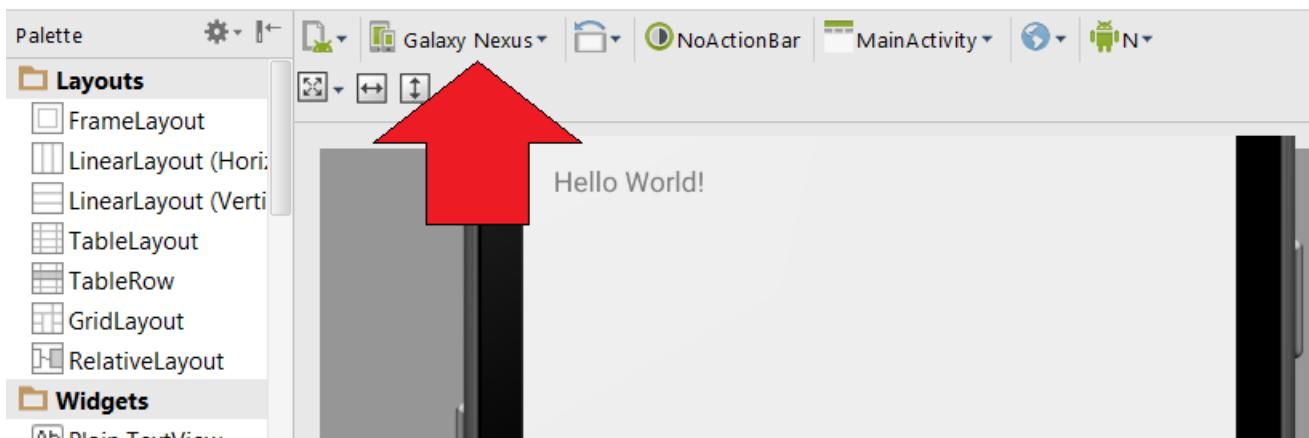


Clique em “Next” e selecione “Basic Activity”, clique em “Next” e em “Finish” para gerar nosso projeto.

Aguarde até que o projeto seja gerado.

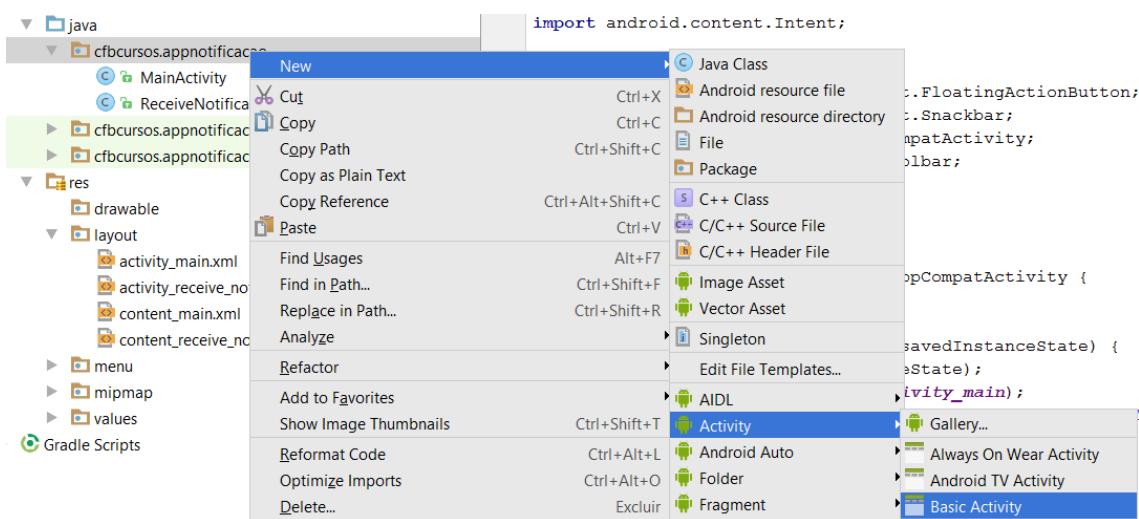
Nosso projeto será aberto.

Selecione como “virtual device” o “Galaxy Nexus” que criamos no início do curso.

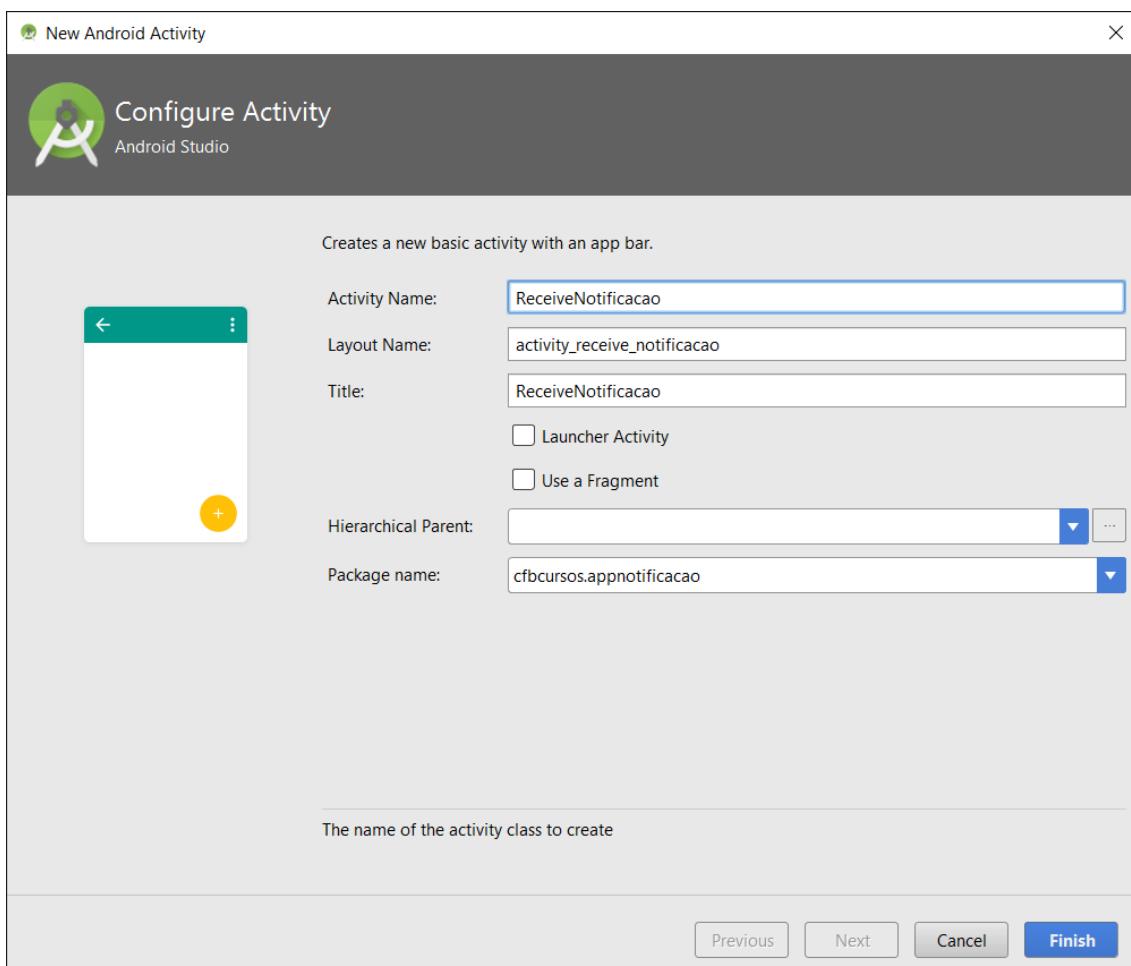


Para criarmos uma sistema de notificação precisaremos criar uma nova activity, para que seja chamada quando a notificação for clicada, basicamente com os comandos que serão pertinentes a tal notificação.

Já sabemos como criar uma nova activity, dentro da pasta “java” clique com o botão direito do mouse em “cfbcursos.appnotificacao”, selecione “New - Activity” e clique em “Basic Activity”.



Digite o nome da nossa activity “ReceiveNotificacao” e clique em “Finish”.



Agora vamos à programação principal no arquivo “MainActivity.java”.

Vamos declarar o método com o código que irá gerar a notificação.

Os imports destacados em vermelho serão feitos automaticamente, se forem feitos, você deverá fazê-los manualmente.

```
3 import android.app.Notification;
4 import android.app.NotificationManager;
5 import android.app.PendingIntent;
6 import android.content.Intent;
7 import android.widget.Button;

8
9 import android.os.Bundle;
10 import android.support.design.widget.FloatingActionButton;
11 import android.support.design.widget.Snackbar;
12 import android.support.v7.app.AppCompatActivity;
13 import android.support.v7.widget.Toolbar;
14 import android.view.View;
15 import android.view.Menu;
16 import android.view.MenuItem;
```

Agora vamos criar o método `criaNotificacao()`.



```
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_main);
25         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
26         setSupportActionBar(toolbar);
27
28         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
29         fab.setOnClickListener((view) -> {
30             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
31                 .setAction("Action", null).show();
32         });
33     }
34
35 }
36
37 public void criaNotificacao() {
38     Intent it=new Intent(this,ReceiveNotificacao.class);
39     PendingIntent pi=PendingIntent.getActivity(this,0,it,0); //getActivity é um método static por isso não precisa do new
40     Notification nt=new Notification.Builder(this)
41         .setContentTitle("CFB").
42         .setContentText("Canal Fessor Bruno - Curso de Android").
43         .setSmallIcon(R.drawable.notification_template_icon_bg).
44         .setContentIntent(pi).build();
45     NotificationManager ntm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
46     nt.flags |= Notification.FLAG_AUTO_CANCEL;
47     ntm.notify(0,nt);
48 }
49
50     @Override
51     public boolean onCreateOptionsMenu(Menu menu) {
52         // Inflate the menu; this adds items to the action bar if it is present.
53         getMenuInflater().inflate(R.menu.menu_main, menu);
54         return true;
55     }
56 }
```

Vou explicar o procedimento.

O primeiro passo é criar um intent para chamar nossa activity recém criada “ReceiveNotificacao” quando o usuário clicar na notificação.

```
Intent it=new Intent(this,ReceiveNotificacao.class);
```

O segundo passo é criar um “PendingIntent” para que seja possível para nossa aplicação passar a notificação ao sistema, note que passamos a intente “it” criada na linha anterior.

Outro detalhe importante é que getActivity é um método static por isso não precisa do new.

```
PendingIntent pi=PendingIntent.getActivity(this,0,it,0);
```

Em seguida criamos a notificação em si em nosso caso criamos com o nome “nt”.

```
Notification nt=new Notification.Builder(this).
```

Os próximos comandos são as configurações de título, texto e ícone da notificação.

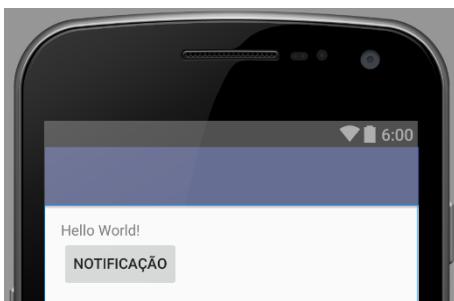
```
setContentTitle("CFB").
setContentText("Canal Fessor Bruno - Curso de Android").
setSmallIcon(R.drawable.notification_template_icon_bg).
```

Após configurar precisamos construir e efetivamente chamar a notificação.

```
setContentIntent(pi).build();
NotificationManager ntm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
nt.flags |= Notification.FLAG_AUTO_CANCEL;
nt.notify(0,nt);
```

Pronto, agora vamos adicionar um botão para que a notificação seja chamada ao clicar neste botão.

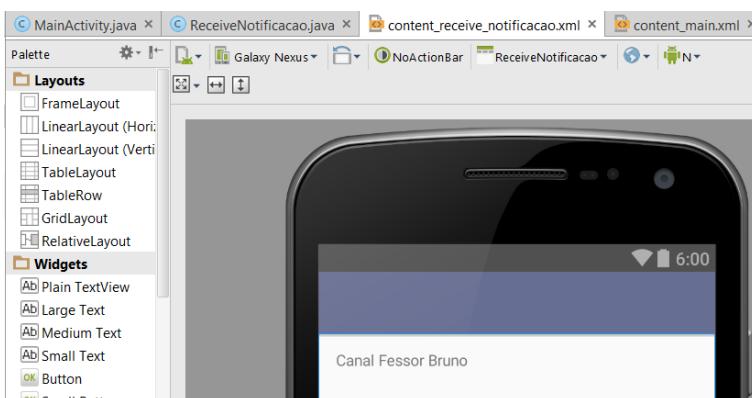
Adicione um elemento “Button” em nossa aplicação, configure com “text = Notificação” e “id = btNotificacao”.



Agora vamos simplesmente programar o botão para chamar o nosso método “criarNotificacao()”.

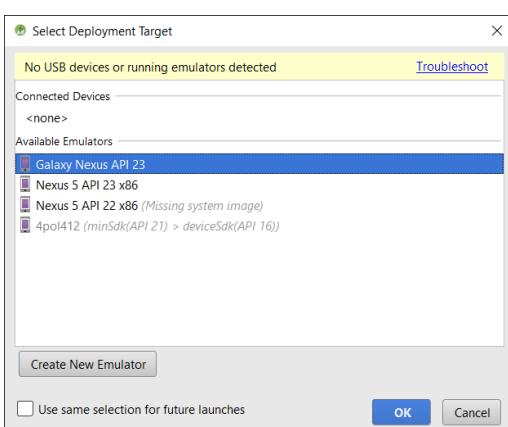
```
27 FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
28 fab.setOnClickListener(view) -> {
29     Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
30         .setAction("Action", null).show();
31 }
32
33
34
35
36 Button btNot=(Button)findViewById(R.id.btNotificacao);
37 btNot.setOnClickListener(new View.OnClickListener() {
38     @Override
39     public void onClick(View v) {
40         criaNotificacao();
41     }
42 }
43
44
45
46 public void criaNotificacao(){
47     Intent it=new Intent(this,ReceiveNotificacao.class);
```

Agora como última tarefa vamos adicionar um “Plain TextView” com o texto “Canal Fessor Bruno”.



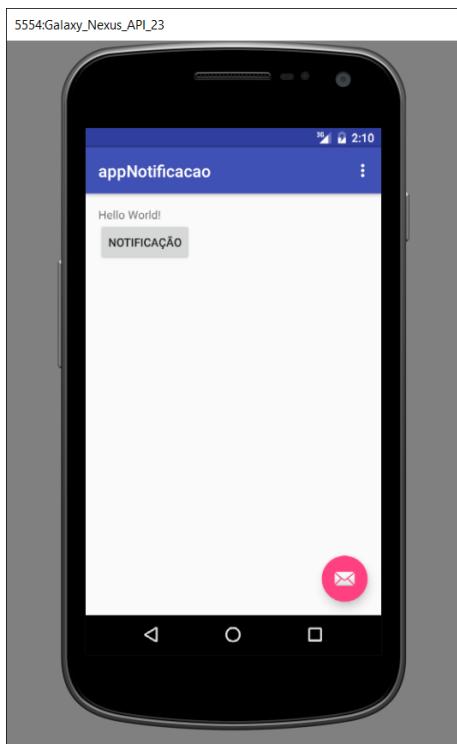
Tudo pronto, agora podemos rodar nossa aplicação para realizar o teste.

Clique no botão “Run App”, selecione “Galaxy Nexus API 23” e clique em OK.

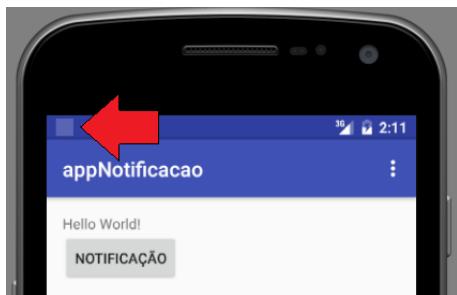




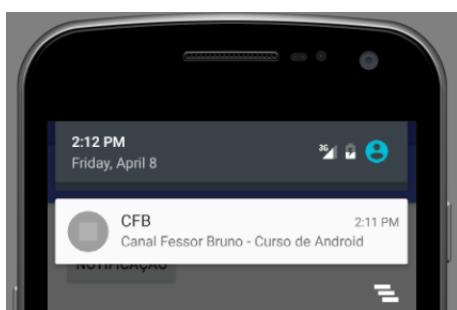
Com o programa rodando clique no botão “Notificação”.



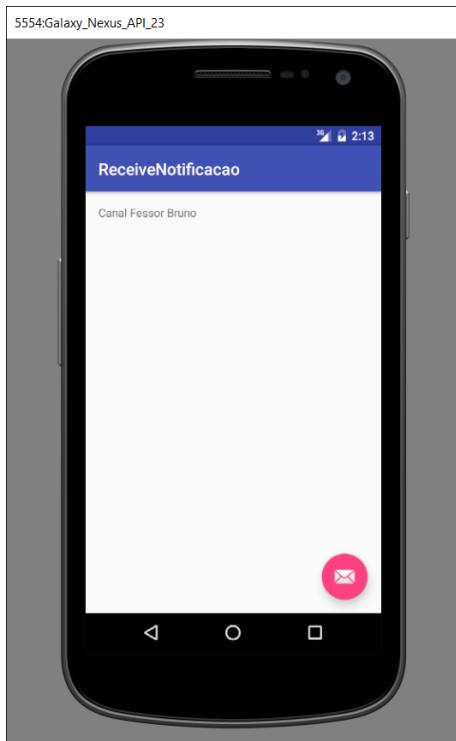
Será gerada a notificação.



Arraste para baixo para ver a notificação.

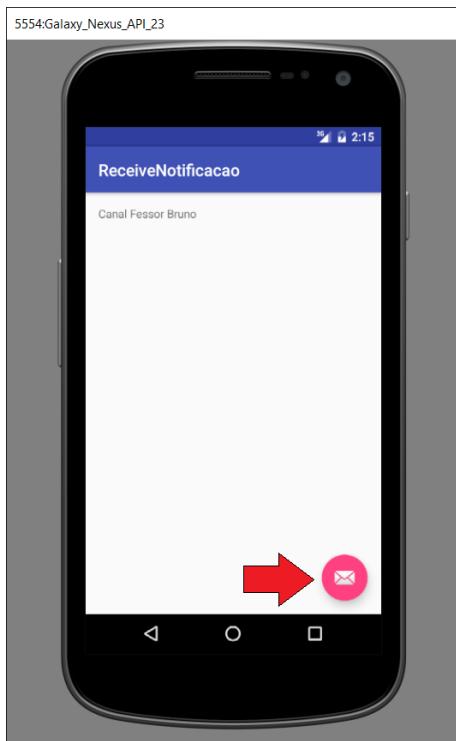


Clique na notificação e veja que será aberta a activity que criamos.

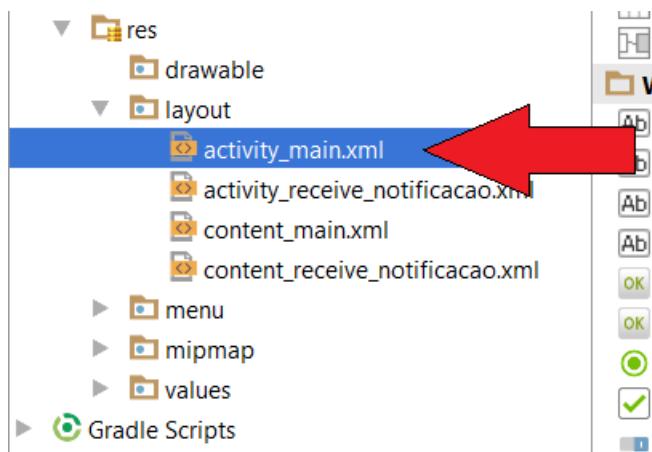


Removendo o ícone de mensagens padrão

Vamos aproveitar a aplicação anterior e remover o ícone do envelope que aparece na parte inferior direita da tela.



Abra o arquivo “activity_main.xml”.



Neste arquivo, remova o código destacado em vermelho na ilustração a seguir.

```
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_main" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:src="@android:drawable/ic_dialog_email" />

</android.support.design.widget.CoordinatorLayout>
```

Agora no arquivo “MainAcitivity” remova o código destacado em vermelho na ilustração a seguir.



```
18 public class MainActivity extends AppCompatActivity {  
19  
20     @Override  
21     protected void onCreate(Bundle savedInstanceState) {  
22         super.onCreate(savedInstanceState);  
23         setContentView(R.layout.activity_main);  
24         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);  
25         setSupportActionBar(toolbar);  
26  
27         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);  
28         fab.setOnClickListener(view -> {  
29             Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)  
30                 .setAction("Action", null).show();  
31         });  
32  
33     }  
34  
35     Button btNot=(Button)findViewById(R.id.btNotificacao);  
36     btNot.setOnClickListener(new View.OnClickListener() {  
37         @Override  
38         public void onClick(View v) {  
39             criaNotificacao();  
40         }  
41     });  
42 }  
43  
44 }
```

Salve todas as alterações e rode a aplicação, note que não existe mais o ícone nesta janela.

