

**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
RIO GRANDE DO NORTE
Campus Natal - Central

Programando Intenções

Prof. Fellipe Aleixo (fellipe.Aleixo@ifrn.edu.br)

Conteúdo

- Desenvolvimento de uma aplicação para registrar contatos (nome e telefones) e realizar chamadas
- **Intent** (intenções)
 - Aplicações com mais de uma View
 - Passagem de parâmetros entre Views
 - Utilização de aplicações nativas
- Componentes de Interface
 - **Menu, ScrollView**
- Serviços do Android (**System-Level Services**)
 - `Layout_Inflater`

Intents

- As **Intents** representam ações que uma aplicação Android deseja executar
 - São objetos da classe **android.content.Intent**
 - Representam mensagens (broadcast) para o sistema operacional

Intents

- As **Intents** podem ser utilizadas para:
 - Abrir novas Views em uma aplicação
 - Abrir Views de outras aplicações e
 - Acessar aplicações nativas
 - Iniciar processamentos em segundo plano (**Broadcast Receivers** e **Services**)

Intents – Métodos de Lançamento

- Métodos da classe **Activity** usados com Intents:
- **startActivity (Intent intent [, Bundle options])**
 - Inicia uma nova Activity, passando o controle da aplicação para a nova atividade
 - O parâmetro **Bundle** é opcional e usado para passar parâmetros entre as atividades

Intents – Métodos de Lançamento

- `startActivityForResult (Intent intent, int requestCode [, Bundle options])`
 - Inicia uma nova `Activity` da qual é esperado algum retorno
 - O parâmetro `requestCode` é usado para identificar a atividade que retorna o resultado

Intents – Métodos de Retorno

- `setResult (int resultCode[, Intent data])`
 - Usado para informar o valor de retorno da atividade para a Activity chamadora
 - Valores padrões: `RESULT_CANCELED` ou `RESULT_OK`
 - O parâmetro `data` é opcional e usado para retornar dados para a Activity chamadora

Intents – Métodos de Retorno

- `finish ()`
 - Usado para encerrar (fechar) uma `Activity`
 - Dispara o método `onActivityResult` na `Activity` chamadora

Intents - Construtores

- Intent (String action, Uri uri)
 - Cria uma intenção para uma determinada ação (*action*), normalmente aplicações nativas
 - O parâmetro URI define as informações necessárias à execução da ação
 - **ACTION_CALL**: Realiza uma chamada telefônica
 - **ACTION_VIEW**: Abre uma aplicação para a visualizar a uri fornecida

Intents - Construtores

- Intent (Context packageContext, Class cls)
 - Cria uma intenção para uma classe específica dentro de uma aplicação
 - Utilizada para abrir novas atividades dentro de uma aplicação

Menu - Recursos

- Definindo menus de uma aplicação Android:
 - Utilizando um arquivo na Pasta “res\menu”

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
  <item
    android:id="@+id/action_settings"
    android:orderInCategory="100"
    android:showAsAction="never"
    android:title="@string/action_settings" />
  <item
    android:id="@+id/action_close"
    android:orderInCategory="101"
    android:showAsAction="never"
    android:title="@string/fechar" />
</menu>
```

Menu - onCreateOptionsMenu

- Utilizando o método Add da classe Menu:
 - Add(int groupId, int itemId, int order, int titleRes)
 - Add(int groupId, int itemId, int order, CharSequence title)

```
public boolean onCreateOptionsMenu(Menu menu) {  
    menu.add(Menu.NONE, Menu.First, Menu.NONE, R.string.novoContato);  
    menu.add(Menu.NONE, Menu.First+1, Menu.NONE, R.string.sobre);  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}
```

Menu - onOptionsItemSelected

- O método `onOptionsItemSelected` da Activity é chamado ao selecionar um item de menu
- É necessário testar qual o item selecionado

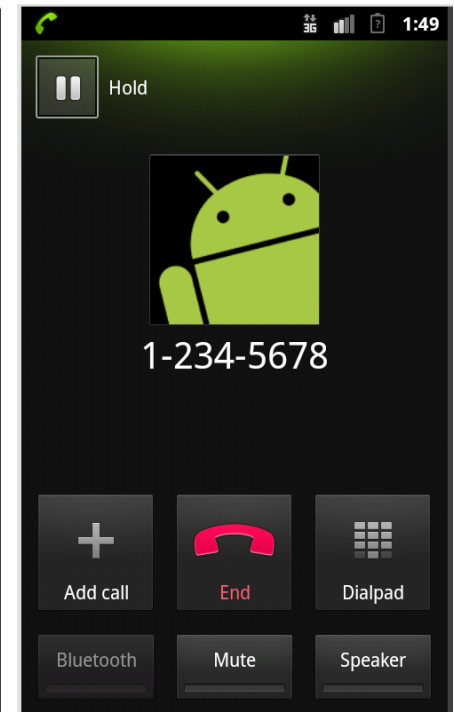
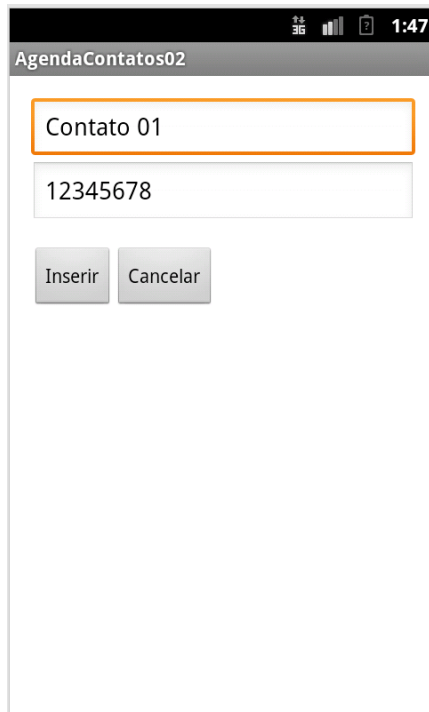
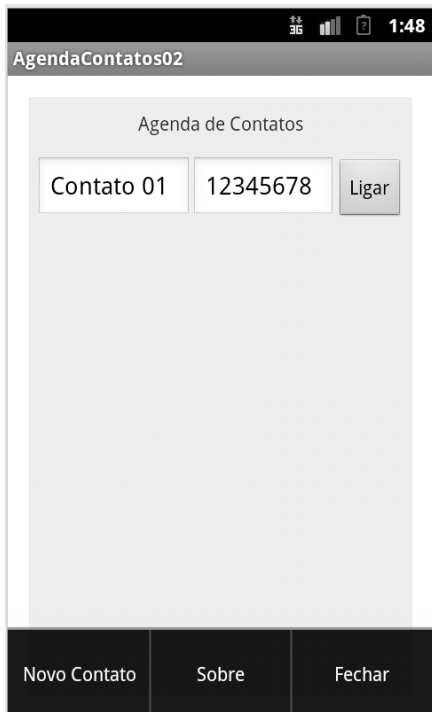
```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch(item.getItemId()) {  
        case Menu.First :  
            Intent novo_contato = new Intent(this, NovoContatoActivity.class);  
            startActivityForResult(novo_contato, 0);  
            break;  
    }  
}
```

Serviços

- O método `getSystemService` da classe `Activity` pode ser usado para acessar um serviço do sistema
 - Permite acessar serviços do dispositivo como: Controle de Energia, Localização, Teclado, Wi-Fi, ...
 - O `LAYOUT_INFLATER_SERVICE` pode ser utilizado para inflar recursos em uma aplicação
 - O método `inflate` da classe `LayoutInflater` permite instanciar componentes, definidos um arquivo XML, em tempo de execução
 - `LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);`
 - `View row = inflater.inflate(R.layout.tablerow_novo_contato, null);`

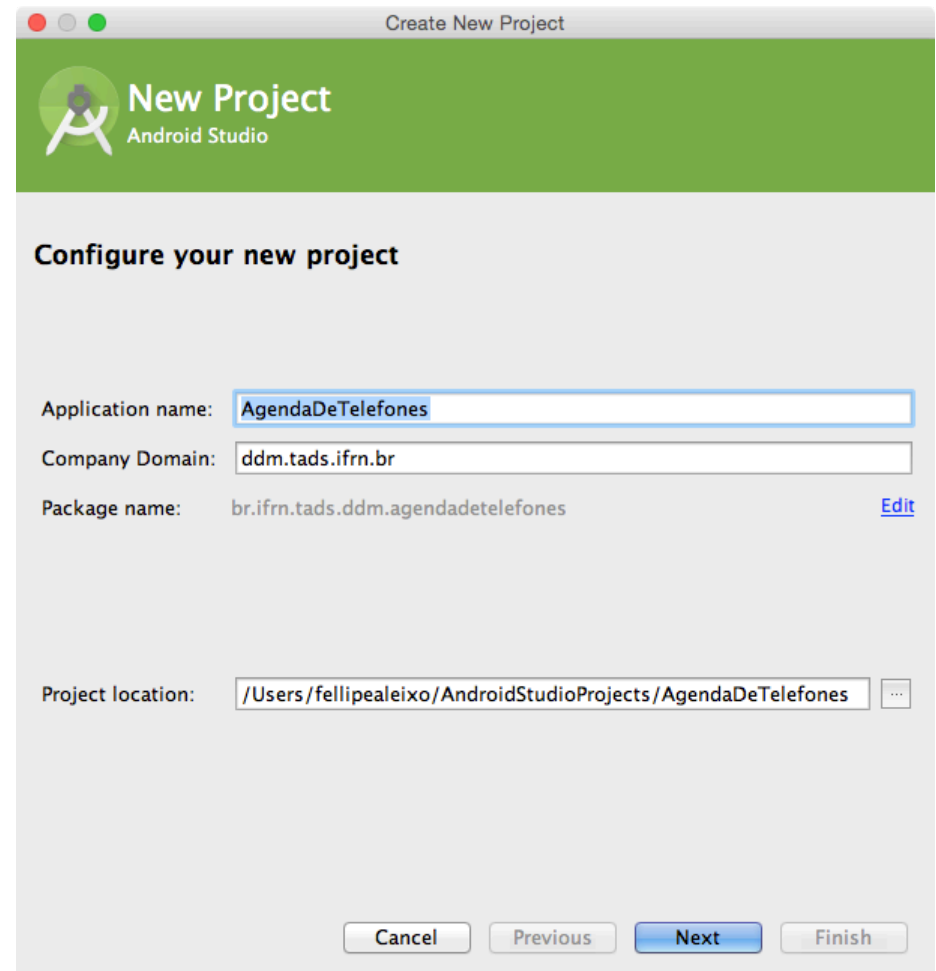
Exemplo – Agenda de Contatos

- Agenda de Contatos
 - Três atividades
 - Utilização da “ação” de chamada telefônica



Criação do Projeto

- No Android Studio, siga os passos do exemplo anterior
- A interface padrão e demais arquivos do projeto são criados



strings.xml

- No arquivo strings.xml, defina as strings a serem utilizadas na aplicação:

```
<resources>
  <string name="app_name">Agenda de Telefones</string>
  <string name="action_settings">Configurações</string>
  <string name="action_close">Fechar</string>
  <string name="titulo">Agenda de Telefones</string>
  <string name="subtitulo">IFRN 2015.1</string>
  <string name="novoContato">Novo Contato</string>
  <string name="nome">Digite o seu nome aqui</string>
  <string name="telefone">Digite o seu telefone aqui</string>
  <string name="inserir">Inserir</string>
  <string name="cancelar">Cancelar</string>
  <string name="ligar">Ligar</string>
  <string name="sobre">Sobre</string>
</resources>
```

activity_main: LinearLayout

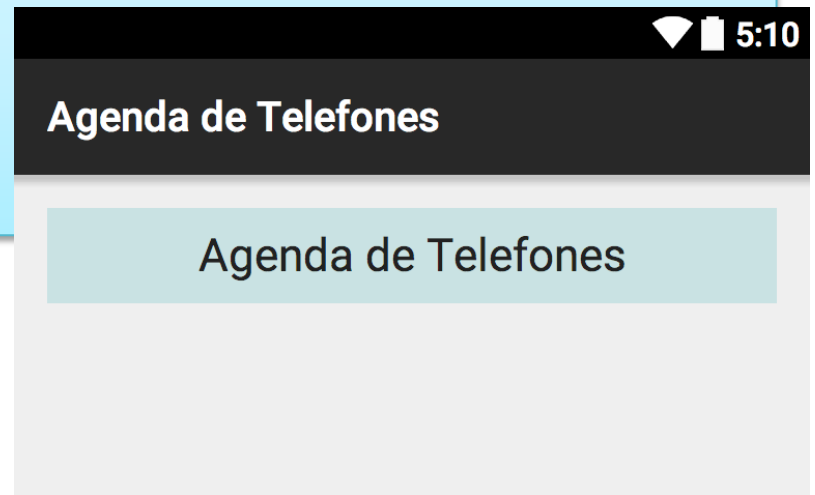
- Substitua o **RelativeLayout** por **LinearLayout**, com a propriedade **orientation** igual a vertical

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

activity_main: TextView

- Adicione um TextView (*Large*) no layout para mostrar o “cabeçalho” da lista de telefones

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:background="@color/highlighted_text_material_dark"
    android:text="@string/titulo"
    android:id="@+id/textView"
    android:paddingTop="10dp"
    android:paddingBottom="10dp"
    android:gravity="center"/>
```



activity_main: ScrollView

- Adicione um **ScrollView** e um **TableLayout** para servir de container para os contatos



activity_main: ScrollView



```
<TextView .../>  
  
<ScrollView  
    android:layout_width="match_parent"  
    android:layout_height="0dp"  
    android:layout_weight="1"  
    android:background="#ffffff"  
    android:id="@+id/scrollView" >  
    <TableLayout  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:padding="5dp"  
        android:stretchColumns="0,1">  
    </TableLayout>  
</ScrollView>
```

activity_novo_contato


- Crie uma nova *Activity* na aplicação, selecionando **File | New | Activity**
 - Escolha “*Blank Activity*”
- Defina os nomes
 - Classe: **NovoContatoActivity**
 - Layout: **activity_novo_contato**
- Essa *Activity* utilizará o **LinearLayout**

activity_novo_contato

New Android Activity

 Customize the Activity 

Creates a new blank activity with an action bar.


Blank Activity

Activity Name:

Layout Name:

Title:

Menu Resource Name:

Launcher Activity

Hierarchical Parent: ...

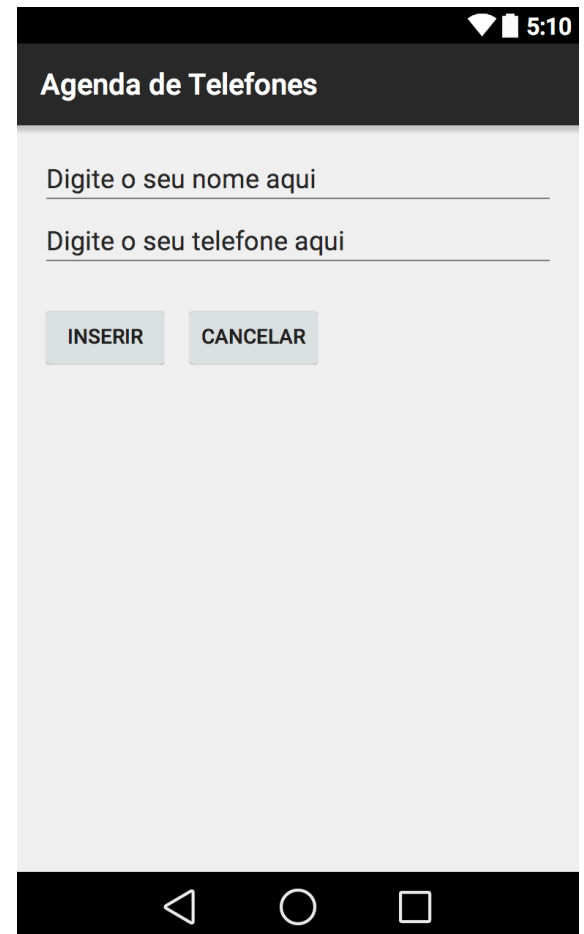
Package name:

The name of the activity class to create

activity_novo_contato

- Defina o layout para a inserção de contatos:

```
<LinearLayout ... android:orientation="vertical">  
  
    <EditText ...    android:hint="@string/nome"  
                    android:id="@+id/editText1" />  
    <EditText ...    android:hint="@string/telefone"  
                    android:id="@+id/editText2" />  
  
    <RelativeLayout ... >  
        <Button...    android:text="@string/inserir"  
                    android:id="@+id/button1" />  
        <Button...    android:text="@string/cancelar"  
                    android:id="@+id/button2" />  
    </RelativeLayout>  
</LinearLayout>
```



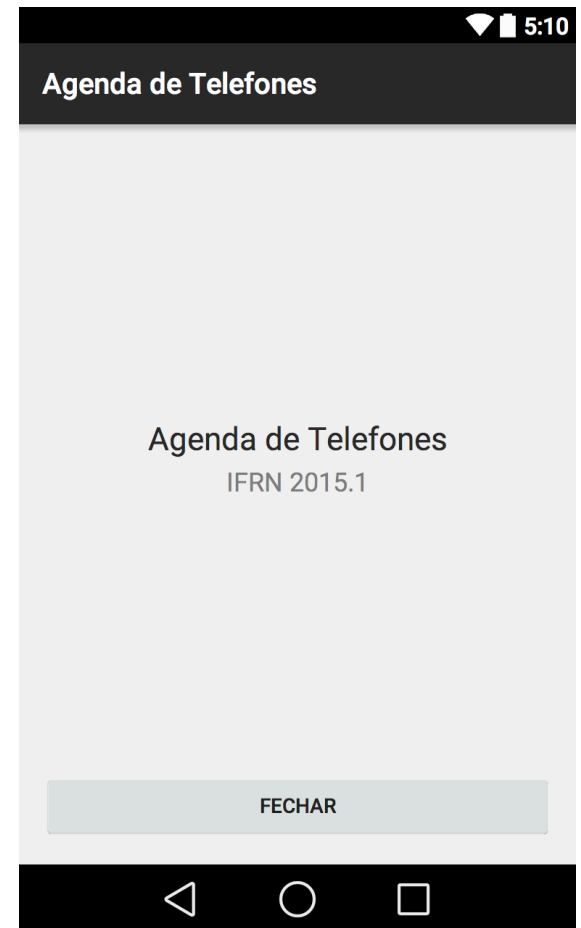
activity_sobre

- Insira mais uma *Activity* na aplicação
- Defina os nomes
 - Da classe: `SobreActivity`
 - Do arquivo: `activity_sobre`
 - Do título: `Sobre a Aplicação`
- Altere essa *Activity* para utilizar o `LinearLayout`

activity_sobre

- Defina a interface da View sobre:

```
<TextView
    android:layout_width="match_parent"
    android:gravity="bottom|center_horizontal"
    android:layout_weight="1"
    android:layout_marginBottom="6dp" ... />
<TextView
    android:layout_width="match_parent"
    android:gravity="top|center_horizontal"
    android:layout_weight="1" ... />
<Button
    android:layout_width="match_parent" ... />
```





tablerow_novo_contato

- Insira mais um layout na aplicação
 - File | New | XML | Layout XML File
 - Com o nome: `tbrow_novo_contato`
- Selecione **TableRow** como **Root Element**

tablerow_novo_contato

New Android Activity

 Customize the Activity 

Creates a new XML layout file.

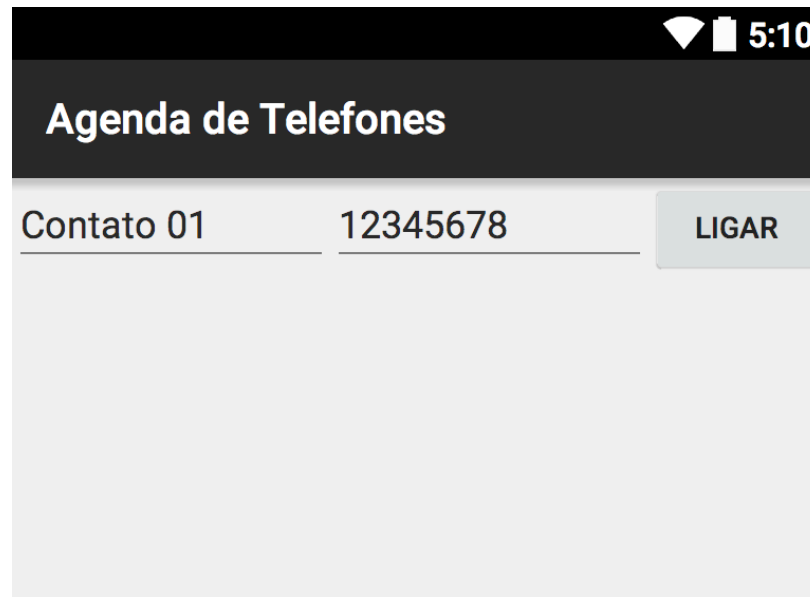
Layout File Name:

Root Tag:

Name of the layout XML file.

tablerow_novo_contato

- Defina a visão da seguinte forma:



tablerow_novo_contato

```
<TableRow ... android:layout_width="match_parent"
               android:layout_height="match_parent">
  <EditText ...
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="5" />
  <EditText ...
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="5" />
  <Button ...
    style="?android:attr/buttonStyleSmall"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
</TableRow>
```

AndroidManifest.xml

- Inserir a permissão para “efetuar ligações”

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ...>
  <uses-permission android:name="android.permission.CALL_PHONE"/>
  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher" ...>
    <activity
      android:name=".MainActivity"
      android:label="@string/app_name" >
      <intent-filter> <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name=".NovoContatoActivity"
      android:label="@string/title_activity_novo_contato" ></activity>
    <activity
      android:name=".SobreActivity"
      android:label="@string/title_activity_sobre" ></activity>
  </application>
</manifest>
```

MainActivity – Passo 01

- Declare constantes para serem utilizadas pelos itens do menu
- Declare a referência para o `TableLayout` da View `activity_main`
- O método `onCreate`, recupera a referência do `TableLayout` utilizado para inserir os componentes `tbrow_novo_contato`

MainActivity – Passo 01

```
private final int ITEM_NOVOCONTATO = Menu.FIRST;
private final int ITEM_SOBRE = Menu.FIRST + 1;
private TableLayout tableLayout;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tableLayout = (TableLayout) findViewById(R.id.tablaLayout1);
}
...
```

MainActivity – Passo 2

- O método `onCreateOptionsMenu` adiciona os itens de menu “Novo Contato” e “Sobre”, com os Ids `ITEM_NOVOCONTATO` e `ITEM_SOBRE`

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add(Menu.NONE, ITEM_NOVOCONTATO, Menu.NONE, R.string.novoContato);
    menu.add(Menu.NONE, ITEM_SOBRE, Menu.NONE, R.string.sobre);
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

MainActivity – Passo 3

- O método `onOptionsItemSelected` é chamado quando um item de menu é selecionado
- O ID do item é testado, se foi selecionado

MainActivity – Passo 3

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case ITEM_NOVOCONTATO :
            Intent novo_contato = new Intent(this, NovoContatoActivity.class);
            startActivityForResult(novo_contato, 0); break;
        case ITEM_SOBRE :
            Intent sobre = new Intent(this, SobreActivity.class);
            startActivity(sobre); break;
        case R.id.action_close :
            finish(); break;
    }
    return true;
}
```

MainActivity – Passo 4

- Item de menu: Novo Contato
 - Quando o item “Novo Contato” é selecionado, uma nova `Intent` é instanciada. No construtor a classe `NovoContatoActivity` é passada como parâmetro
 - O método `startActivityForResult` abre a nova atividade e fica aguardando um valor de retorno
 - O valor 0 é passado como `requestCode` para identificar a atividade que está sendo chamada
 - `Intent novo_contato = new Intent(this, NovoContatoActivity.class);`
 - `startActivityForResult(novo_contato, 0); break;`

MainActivity – Passo 5

- Item de menu: Sobre
 - Quando o Sobre é selecionado, a nova `Intent` recebe a classe `SobreActivity` como parâmetro.
 - O método `startActivity` abre a atividade e, neste caso, nenhum valor de retorno é esperado
 - `Intent sobre = new Intent(this, SobreActivity.class);`
 - `startActivity(sobre); break;`
- Item de menu: Fechar
 - O método `finish` encerra a aplicação
 - `finish(); break;`

MainActivity – Passo 6

- O método `onActivityResult` é chamado quando a atividade novo contato encerra
- Três valores são retornados:
 - `requestCode`: é o identificador da atividade chamada
 - `resultCode`: é um código de resultado, normalmente `OK` ou `CANCELED`
 - `data`: é um objeto com dados do retorno (nome e fone do contato)

MainActivity – Passo 6

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 0) {
        if (resultCode == RESULT_OK) {
            String nome = data.getCharSequenceExtra("nome").toString();
            String fone = data.getCharSequenceExtra("fone").toString();
            inserirContato(nome, fone);
        }
    }
}
```


MainActivity – Passo 7

- O método `inserirContato` usa o `LayoutInflater_Service` para inflar o layout `tablerow_novo_contato`
- As `EditTexts` no layout recebem nome e fone do contato
- O objeto `buttonListener` é setado como `listener` do botão Ligar

MainActivity – Passo 7

```
private void inserirContato(String aNome, String aFone) {  
    LayoutInflater inflater =  
        (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);  
    View row = inflater.inflate(R.layout.tbrow_novo_contato, null);  
  
    EditText editNome = (EditText) row.findViewById(R.id.editText3);  
    EditText editFone = (EditText) row.findViewById(R.id.editText4);  
    Button button = (Button) row.findViewById(R.id.button3);  
  
    editNome.setText(aNome);  
    editFone.setText(aFone);  
  
    button.setOnClickListener(buttonListener);  
    tableLayout.addView(row);  
}
```

MainActivity – Passo 8

- Finalmente, o objeto `buttonListener` realiza a chamada para o fone do contato, utilizando uma `Intent` com a ação `ACTION_CALL`
- A classe `MainActivity` está concluída

MainActivity – Passo 8

```
private OnClickListener buttonListener = new OnClickListener() {
    @Override
    public void onClick(View v) {
        TableRow row = (TableRow) v.getParent();

        EditText editFone = (EditText) row.findViewById(R.id.row_editFone);
        String s = "tel:" + editFone.getText().toString();

        Uri uri = Uri.parse(s);
        Intent fone = new Intent(Intent.ACTION_CALL, uri);

        startActivity(fone);
    }
};
```

NovoContatoActivity – Passo 1

- A classe **NovoContatoActivity** é usada para definição do nome e fone de um contato

```
private Button btnOk;  
private Button btnCancel;  
private EditText editNome;  
private EditText editFone;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_novo_contato);  
    btnOk = (Button) findViewById(R.id.nc_btnOk);  
    btnCancel = (Button) findViewById(R.id.nc_btnCancel);  
    editNome = (EditText) findViewById(R.id.nc_editNome);  
    editFone = (EditText) findViewById(R.id.nc_editFone);  
}
```

NovoContatoActivity – Passo 2

- O método `buttonClick` retorna `OK` com os dados do contato ou `CANCELED`

```
public void buttonClick(View v) {  
    if (v == btnOk) {  
        Intent ret = new Intent();  
        String nome = editNome.getText().toString();  
        String fone = editFone.getText().toString();  
        ret.putExtra("nome", nome);  
        ret.putExtra("fone", fone);  
        setResult(RESULT_OK, ret);  
        finish(); }  
    if (v == btnCancel) {  
        setResult(RESULT_CANCELED);  
        finish(); }  
}
```

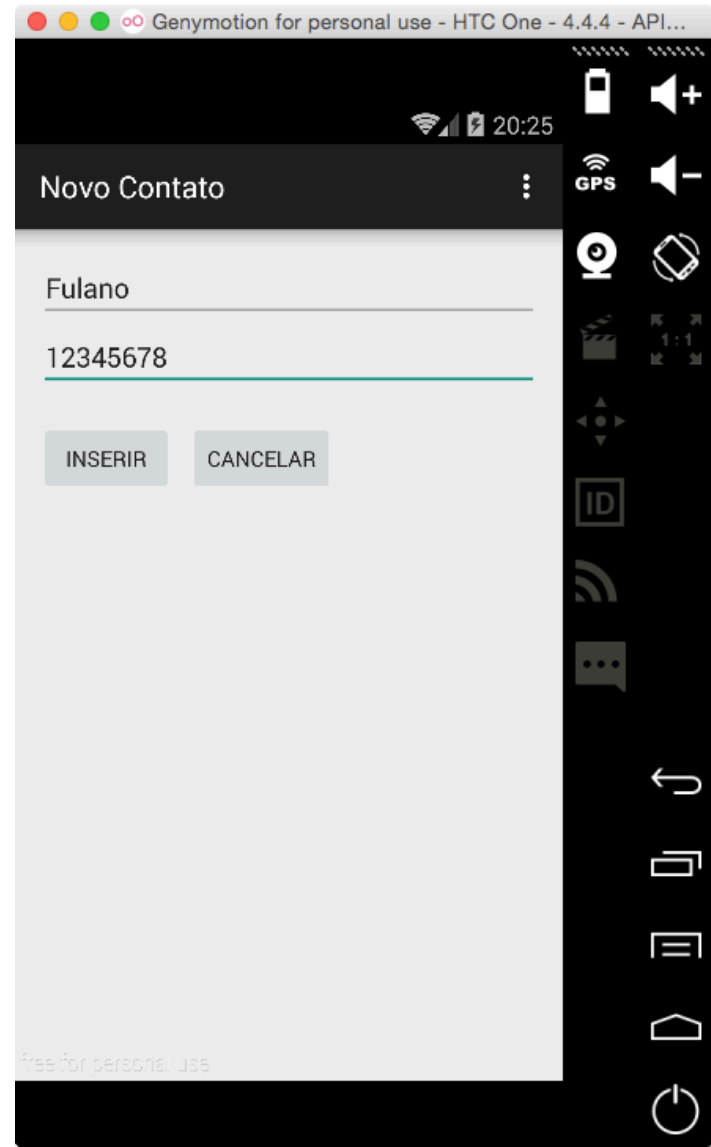
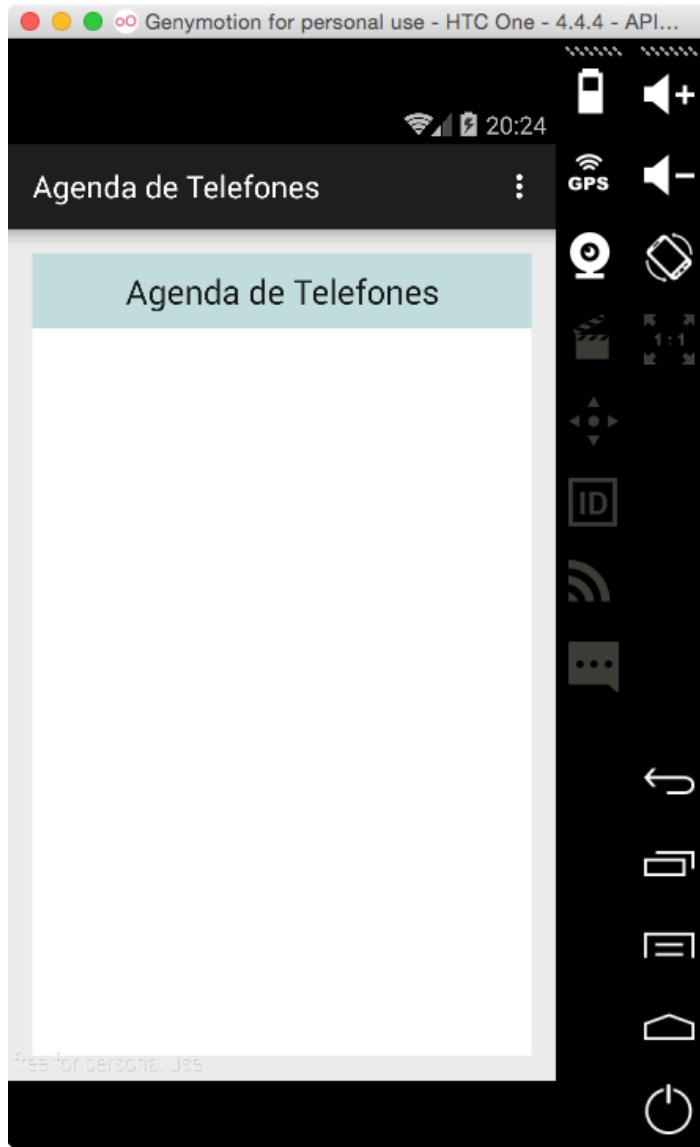
SobreActivity

- A classe **SobreActivity** define apenas fecha a atividade, após o clique do botão Fechar

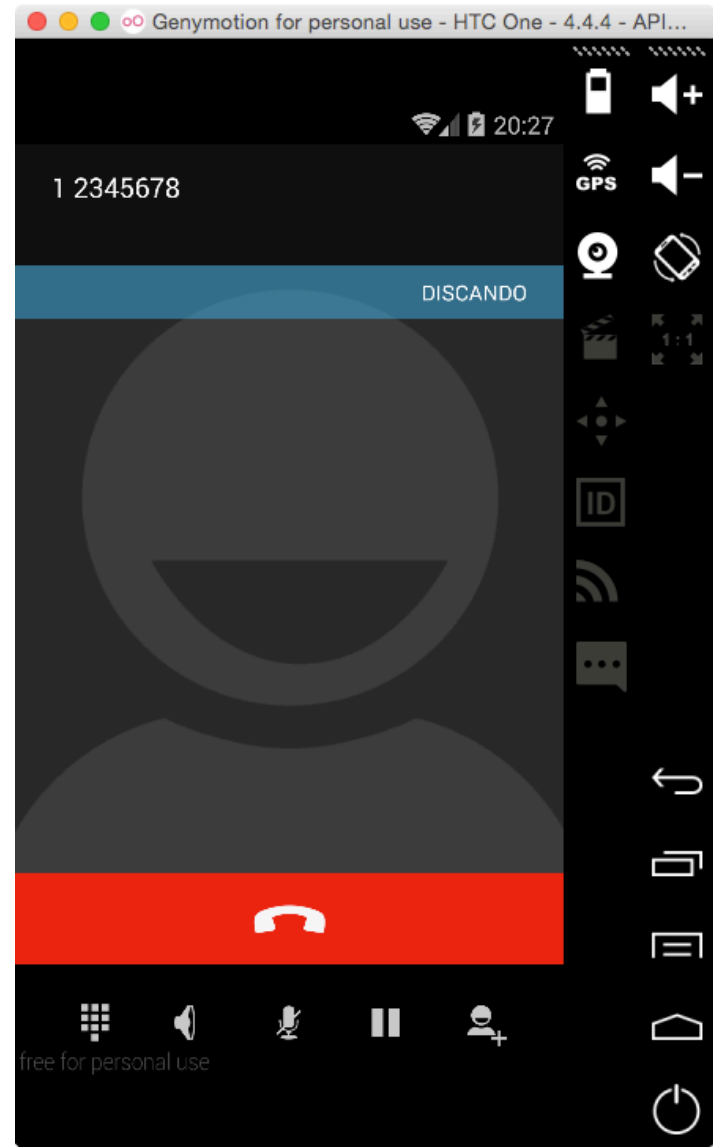
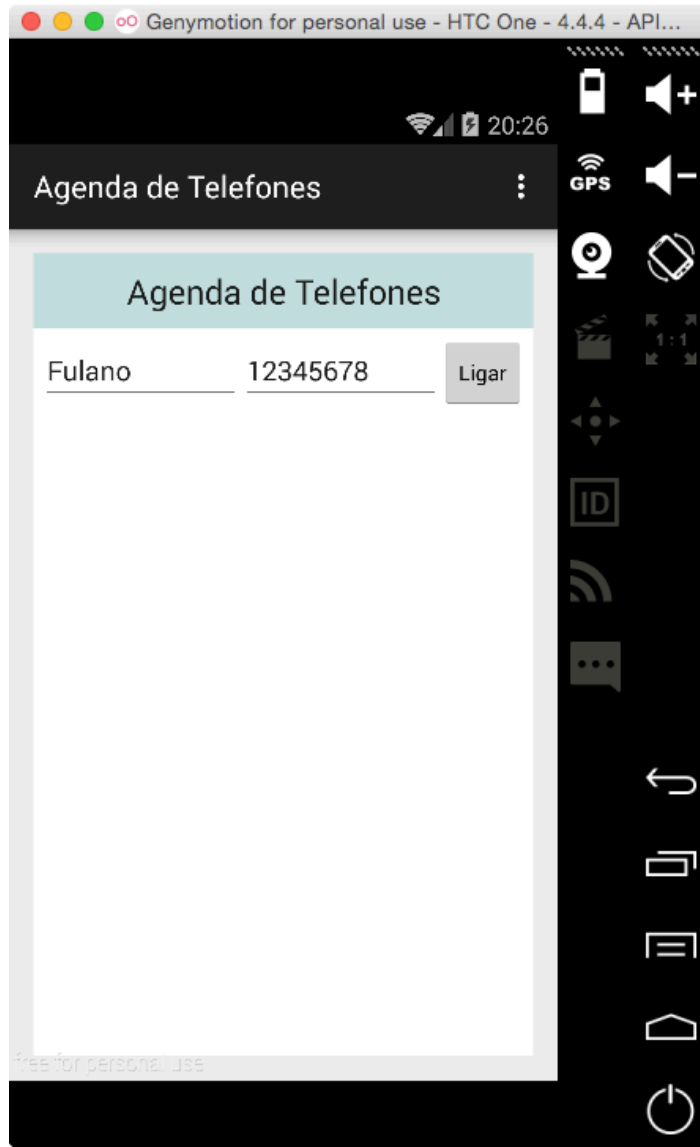
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_sobre);
}

public void btnFecharClick(View v)
{
    finish();
}
```

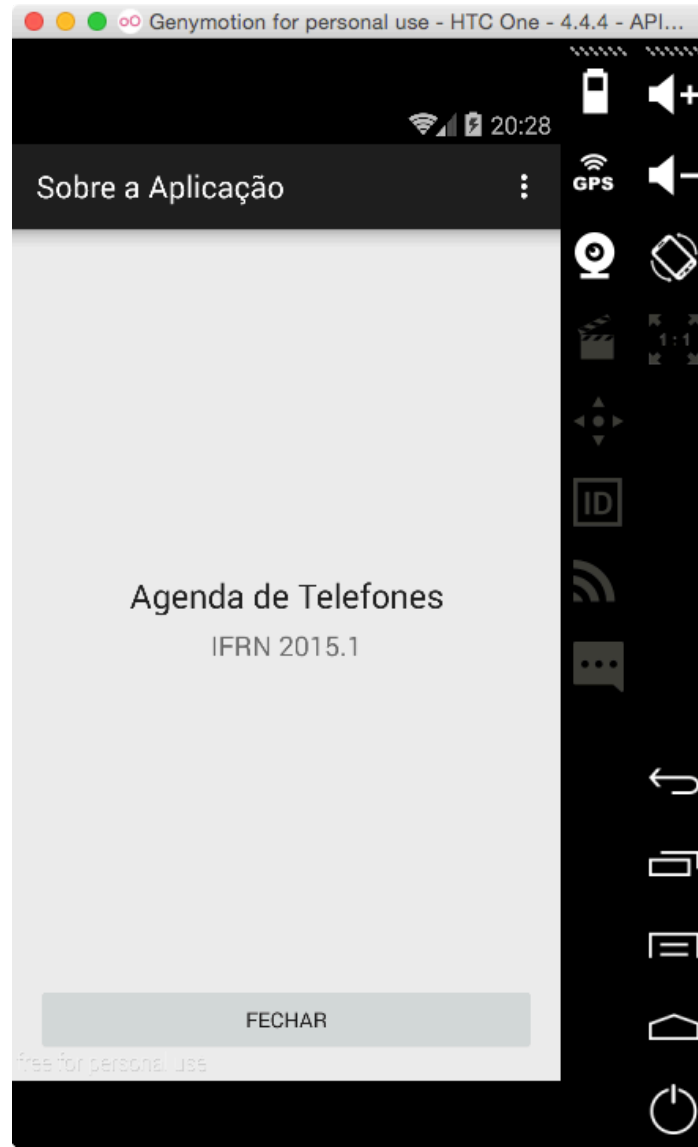
Executando a Aplicação



Executando a Aplicação



Executando a Aplicação



Referências

- Android para Programadores – Uma abordagem baseada em aplicativos. Paul Deitel ... [et al.]. Bookman, 2013
- Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK. Ricardo R. Lecheta. Novatec, 2013
- <http://developer.android.com/reference>