03

Utilizando menus

Transcrição

No último vídeo introduzimos no botão "+" a função de ir até o formulário e no botão "Salvar" a de voltar para a lista. Também conseguimos apertar o botão de voltar da tela do celular e fazer com que ele retorne ao *Launcher*.

Agora, vamos alterar o design do botão "Salvar" para um desenho mais bacana. Para que ele combine com o padrão *google* do botão de adicionar.

Vamos introduzir o botão de salvar junto a nossa barra de "Formulário". Desse modo, se o formulário for muito grande e a barra tiver que rolar, o botão ficará em cima dela.

Para fazer essas modificações, acessamos a aba `FormularioActivity.java` .

A barra, entretanto, é controlada pelo próprio *Android*. Quando extendemos a `activity` com o `AppCompatActivity` estamos dizendo que queremos uma aplicação que tenha uma *action bar*. Então, precisamos comunicar ao *Android* que queremos inserir um botão na barra.

Podemos fazer isso introduzindo um método na `activity` . Para isso, damos alguns "Enters" após o último método e escrevemos `onCreateOptionsMenu` . Dando um enter ele automaticamente preenche o *override* desse método.

Esse método cria a barra do `Formulário` e define quais itens do menu constarão nela. Os botões que queremos inserir fazem parte do menu da aplicação. A diferença é que algumas opções serão mostradas dentro da barra e outras como menu.

Ficaremos com:

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

    Button botaoSaLvar = (Button) findViewById(R.id.formulario_salvar);
    botaoSalvar.setOnCIickListener(new View.0nCIickListener() {
        @Override
            public void onClick(View v) {
                Toast.makeText(FormularioActivity.this. "Aluno salvo!", Toast.LENGTH_SHORT)
                finish();
            }
    });
}


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        return super.onCreateOptionsMenu(menu);

    }
```

Inicialmente, não temos nada dentro do método e para criar um menu vamos na pasta "res". Clicando duas vezes nela vemos que já existem dois menus criados: "menu_formulario.xml" e o "menu_lista_alunos.xml". Eles foram criados automaticamente pelo *Android Studio*, pois, toda vez que criamos uma `activity` criam-se esse menus.

Para ver isso, basta ir em menu, "File > New File > Blanky Activity". Quando abrir a janela pedindo os dados da nova *Activity* repare que existe um campo referente ao menu, o *Menu Resources Name*. Esse campo diz respeito ao menu que será criado automaticamente junto com a `activity` .

Se abrirmos o "menu_formulario.xml" veremos que já parece um menu mais ou menos pronto. Nele há uma *tag* `menu` que caracteriza um menu. Temos o seguinte código na tela:

```
<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:context="br.com.alura.agenda.FormularioActivity">
    <item android:id="@+id/action_settings"
    android:title="Settings"
    android:orderInCategory="100" app:showAsAction="never" />

</menu>
```

Se você não encontrar a pasta `menu` , basta criar o arquivo `menu_formulario.xml` com o conteúdo abaixo

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <item
        android:id="@+id/menu_formulario_ok"
        android:title="Ok"
        android:icon="@drawable/ic_confirmar"
        app:showAsAction="always"/>
</menu>
```

Dentro do menu temos um item, o `Settings` . Esse item não será necessário e vamos simplesmente apagá-lo.

Vamos começar a descrever os itens que queremos no menu. Damos um Enter e inserimos a *tag* `item` , introduzindo também um `id` que chamaremos de `menu_formulario_ok` . Faremos isso pois iremos manipulá-lo mais adiante. Vamos inserir o `android:id="@+id/menu_formulario_ok"` .

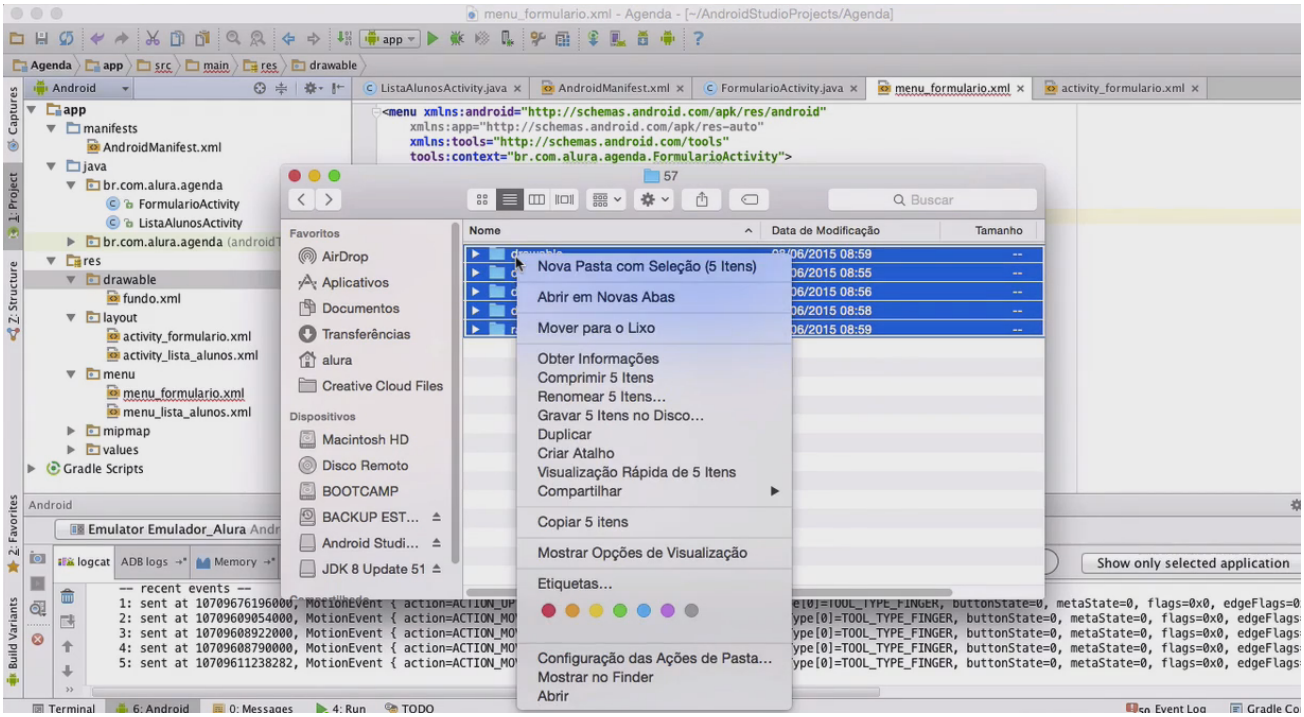
Também iremos introduzir um texto, para indicar o que significa essa opção, vamos colocar que o título dele será "ok". Damos um enter e acrescentamos `android:title="Ok"` .

Outra coisa que acrescentaremos é um ícone. Usaremos, para tanto, o `icon` , assim, o botão aparecerá em forma de desenho na barra. Para fazer isso utilizaremos, na próxima linha, `icon` e também um recurso desenhável, o `drawable` , ficando `android:icon="@drawable"` . Teremos:

```
<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:context="br.com.alura.agenda.FormularioActivity">
    <item android:id="@+id/menu_formulario_ok"
        android:title="Ok"
```

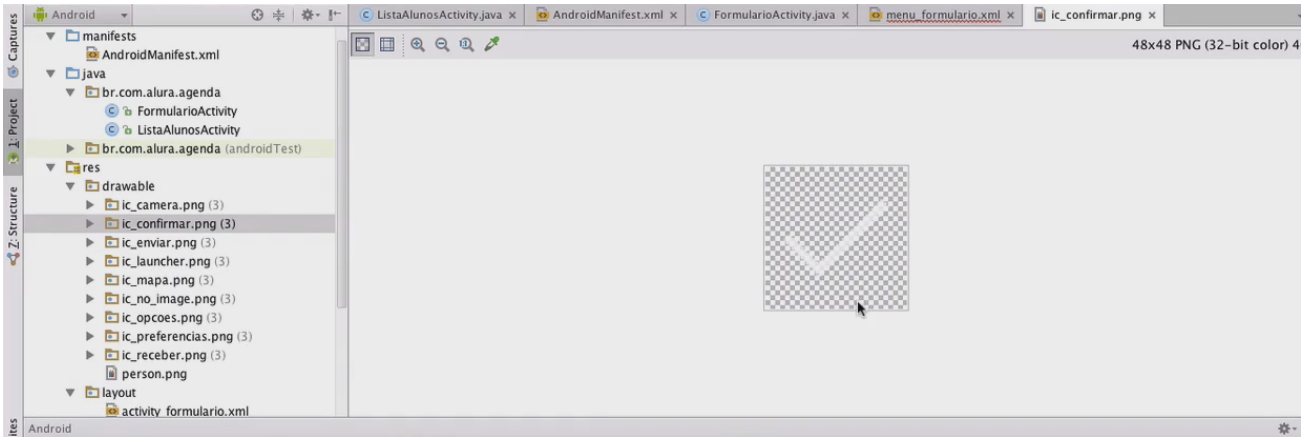
```
        android:icon="@drawable" />

</menu>
```

Para completar o `icon` temos que utilizar um recurso, por enquanto, se observamos a pasta "drawable", temos apenas o arquivo `fundo.xml` . Portanto, vamos no arquivo e copiamos todas as pastas pois nelas estão contidas todas as imagens que utilizaremos ao longo do curso.



Selecionamos com o mouse todas as pastas, clicamos com o botão direito e "Copiar 5 itens" e levamos para a pasta do nosso projeto. Fazemos o seguinte caminho, "AndroidStudioProjects > Agenda > app > src > main > res ". Chegando no "res" é só pedir para colar os itens, clicando com o botão direito e selecionando "Colar 5 itens". Se abrir uma janela perguntando se desejamos substituir é só escolher o "Substituir". Agora temos novos recursos na aplicação.

Assim, voltando para o *Android Studio* se abrirmos a pasta "res", veremos que ela está cheia de novos itens. O que utilizaremos é o item de confirmar, clicando duas vezes ele irá aparecer:



O nome desse item é `ic_confirmar.png` .

Vamos retornar na aba `menu_formulario.xml` para completar o `icon` com o nome do item que vamos adicionar. Ficaremos com `android:icon="@drawable/ic_confirmar"` . Lembre-se de fechar utilizando `/>` . Teremos:

```
<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:context="br.com.alura.agenda.FormularioActivity">

    <item android:id="@+id/menu_formulario_ok"
        android:title="Ok"
        android:icon="@drawable/ic_confirmar"/>

</menu>
```

Agora que criamos o `.xml` precisamos utilizá-lo em nosso código. Se voltamos no `FormularioActivity.java` vemos o método que o *Android* criou para chamar a barrinha:

```
@Override
 public boolean onCreateOptionsMenu(Menu menu) {
       return super.onCreateOptionsMenu(menu);

}
```

Nesse método vamos pedir que o *Android* pegue o `menu_formulario.xml` e transforme em `View`. Isto é, transformar em coisas com as quais o usuário possa interagir.

Para pegar um `.xml` qualquer em um conjunto de `Views` usaremos um conceito do *Android* chamado 'inflater'

Para isso, ficamos no `FormularioActivity.java` e no método menu, damos um enter e na próxima linha, acrescentamos um conceito especialista em fazer menus, que é o `getMenuInflater`.

Esse método `getMenuInflater` está vindo da `AppCompatActivity` e serve para devolver uma instância de um menu inflater. Então, acrescentamos antes do que já escrevemos, `MenuInflater inflater` e um "=" a `getMenuInflater`. Teremos, `MenuInflater inflater = getMenuInflater`. Se ficar em vermelho lembre de importar dando "Alt+Enter".

Devemos lembrar que o `inflater` deve inflar algo, no caso, ele é especialista em inflar uma `.xml`, então, é preciso informar qual `xml` ele deve inflar.

Digitamos a classe `R`, um `.` (ponto final) e como queremos inflar um menu, vamos busca-lo entre as opções apresentadas. Escolhemos "menu", digitamos `.` (ponto final) de novo e vamos escolher o "menu_formulario", que é justamente o que desejamos inflar. Teremos: `inflater(R.menu.menu_formulario)`.

Precisamos dizer, ainda, onde ele vai colocar essas informações, no caso, no menu da *action bar* que está vazio. Ele é o "menu" da *action bar* que está vazio.

Completaremos com um `.` e `menu`, para indicar que queremos que todos os itens estejam nele. Completando, ficaremos com, `inflater(R.menu.menu_formulario, menu)`.

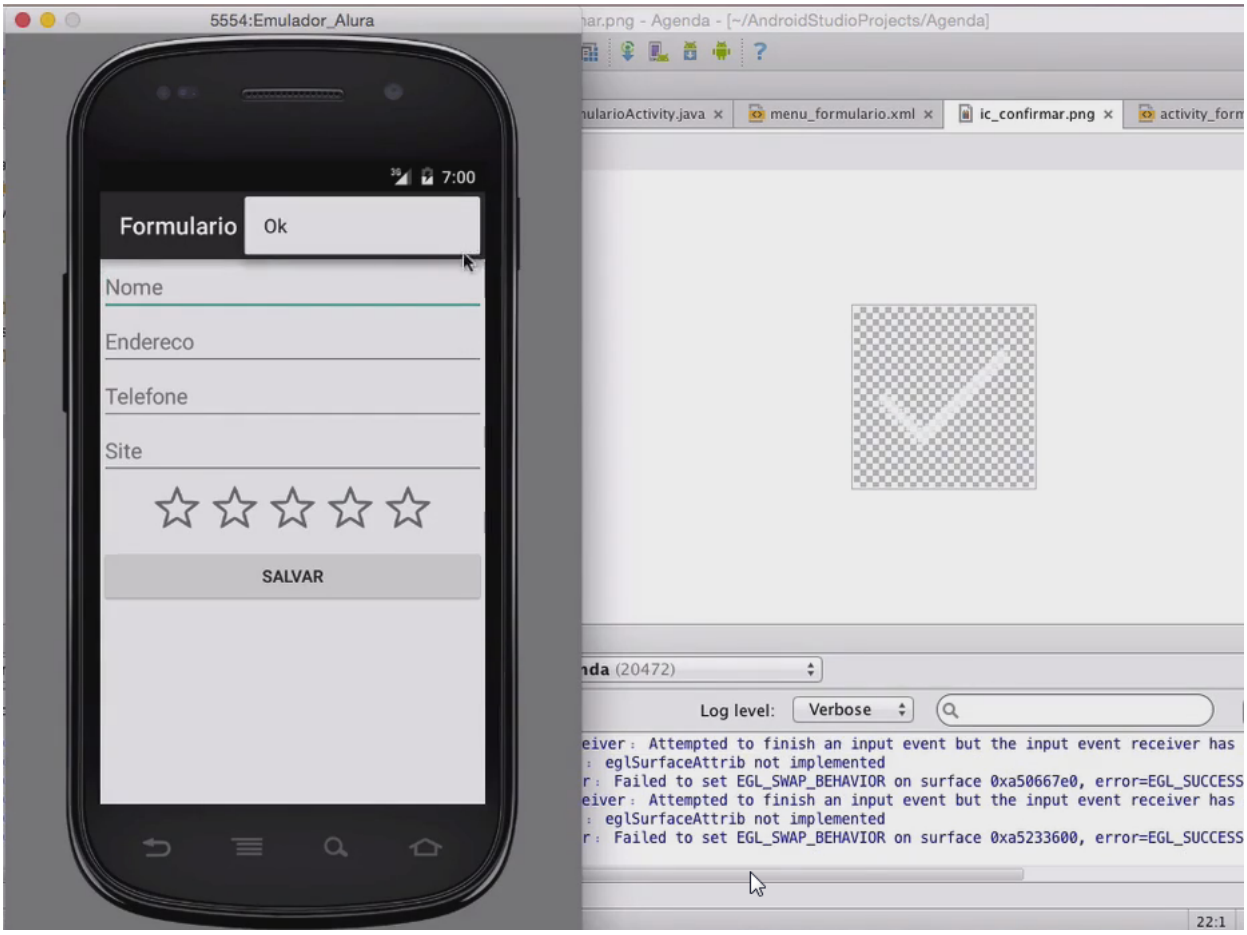
Não podemos esquecer de chamar o método `inflate`. Ele será adicionado depois do `inflater`, portanto, damos um `.` digitamos `inflate`.

Ficaremos com:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_formulario, menu);

return super.onCreateOptionsMenu(menu);


}
```

Vamos ver o que acontece se fizermos só isso:

A barra não está mais vazia! Ela está com esses três pontinhos no canto direito. Se clicarmos em cima das três bolinhas o que aparece é um "ok" escrito. Nosso ícone do menu veio parar aqui e de maneira textual.



Mas, nós tínhamos pedido para ele colocar o item de *check mark*. Lembra?!

Para mostrar o item do menu, não em forma de texto, mas em forma de ícone vamos voltar na aba 'menu_formulario.xml'. Vamos usar a propriedade `showAsAction` e quando digitarmos ele vai dar algumas opções de escolha, vamos usar o `always` ,pois queremos que o ícone seja sempre mostrado na *action bar*. Acrescentaremos, `android:showAsAction="always"` .

O `showAsAction` precisa vir de outro *name space*. Aqui temos dois *name space*, um 'android' e um 'app'. Usaremos o `app` . Apagamos o `android` e digitamos `app` no lugar e `:` (dois pontos). Teremos `app:showAsAction="always"` . Ficaremos com:

```
<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:context="br.com.alura.agenda.FormularioActivity">

    <item android:id="@+id/menu_formulario_ok"
        android:title="Ok"
        android:icon="@drawable/ic_confirmar"
        app:showAsaction="always"/>

</menu>
```

Com isso estamos dizendo que esse item do menu deve ser mostrado com uma das ações da *action bar*.

Vamos rodar para ver como fica.

Agora temos o botão de confirmar, que inclusive já tem uma animação quando clicamos nele!

Falta colocar o comportamento do botão e excluir o botão de "Salvar" que está embaixo do formulário.

Vamos na aba `activity_formulario.xml` para remover o botão antigo. Selecionamos com o mouse o seguinte trecho do código e depois apagamos:

```
<Button android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Salvar"
    android:id="@+id/formulario_salvar"/>
```

Pronto! Se voltarmos agora no `FormularioActivity.java` não encontraremos mais o `R.id.formulario_salvar` .

Só falta definir o comportamento do clique no menu. Poderíamos definir um `Listener` , mas existe uma forma mais elegante de fazer isso no *Android*. Vamos sobrescrever um método que já existe, o `onOptionsItemSelected` , e acrescentaremos ele abaixo do seguinte código:

```
//...

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {

            MenuInflater inflater = getMenuInflater;
               inflater.inflate(R.menu.menuformulario, menu);

        return super.onCreateOptionsMenu(menu);


        }
```

Ao digitar o método `onOptionsItemSelected` o *Android* sinaliza as opções. Escolhemos o método que queremos e é só dar um "Enter". Pronto! Está feito o *override* do método. Teremos:

```
//...

        @Override
        public boolean onCreateOptionsMenu(Menu menu) {

            MenuInflater inflater = getMenuInflater;
            inflater.inflate(R.menu.menuformulario, menu);

        return super.OnCreateOptionMenu(menu);

        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            return super.onOptionsItemSelected(item);

        }
```

Repare que recebemos nesse método o item que foi selecionado, o `MenuItem item` . Assim, toda vez que clicarmos no item no menu o *Android* vai evocar esse método na sua `activity` e vai informar qual objeto do menu foi clicado.

Agora, vamos pegar o comportamento do botão salvar, o `Toast` , e selecionar com o mouse até o `finish` .

```
        @Override
        public void onClick(View v) {
            Toast.makeText(FormularioActivity.this. "Aluno salvo!", Toast.LENGTH_SHORT).shou
            finish();
        }
```

Podemos recortar o 'Toast', através do "Comand+X" ou "Ctrl+X" e colar no Override do `onOptionsItemSelected` com "Command+V" ou "Ctrl+V".

Esse comportamento que acabamos de inserir será extendido para os demais itens do menu. Assim, se tivéssemos 50 itens no menu, todos eles seriam englobados nesse comportamento. Como não queremos que isso ocorra, precisamos fazer uma distinção. Para isso, podemos fazer um `switch`, que acrescentaremos na linha de baixo de `public void`. Digitaremos `switch` e entre parênteses colocaremos `item`, um `.` (ponto final) e acrescentaremos o `Id` do item. Teremos, `switch (item.getItemId())`.

Dependendo do `Id` que tivermos teremos casos distintos. Na linha de baixo do `switch` teremos `case R.id.menu_formulario_ok`.

Caso o `id` seja o `menu_formulario_ok` e como queremos ter o mesmo comportamento inserido anteriormente, selecionamos o `Toast` até o `finish`, copiamos ele e o deslocamos para baixo do *case* que acabamos de introduzir.

Lembre-se de colocar no final o `break`, seguido de um `;`. Caso contrário, ele ficará executando todos os cases que existem depois dele. Se não quisermos esse comportamento, acrescentamos o `break` que "tranca" o `switch`.

```
@Override
        public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
                case R.id.menu_formulario_ok:
                    Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT)
                    finish();
                    break;

            }
        return super.onOptionsItemSelected(item);

        }
```

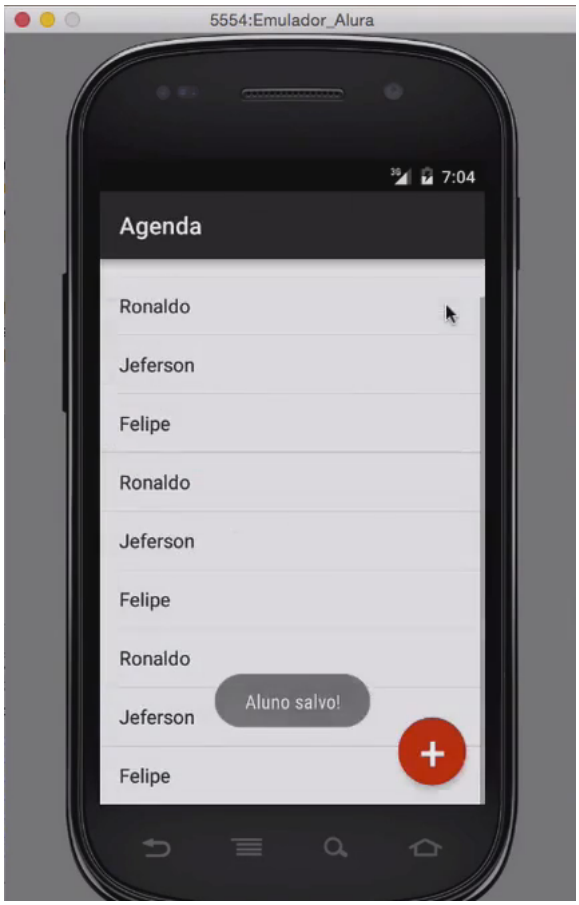
Vamos deletar também o botão de salvar, pois ele não existe mais na nossa tela. Deletamos o seguinte:

```
        Button botaoSalvar = (Button)  findViewById(R.id.formulario_salvar);
        botaoSalvar.setOnCIickListener(new View.0nCIickListener() {
                                @Override
                                 public void onClick(View v) {

                }
        });
```

Agora, quando clicarmos no botão ele vai chamar o método `onOptionsItemSelected` passando o item que foi selecionado. Se esse item for o `menu_formulario_ok` que é o item que definimos no `menu_formulario.xml`, ele mostrará o

`Toast` e terminará a `activity` como queremos.

Vamos salvar e rodar o emulador?



Ao adicionar um novo aluno, preencheremos o formulário e dando um "Ok", ele nos avisa que o aluno foi salvo e retorna para a lista!