02

Tela de Cadastro

Vamos criar uma nova tela para que possamos cadastrar novos alunos.

Para atingir esse objetivo precisaremos criar uma nova `Activity`. Vamos chamá-la de `FormularioActivity` e vinculá-la a um novo layout que chamaremos de `formulario.xml`.

Podemos realizar essa tarefa sem muita dificuldade com um atalho do ADT ou criando uma Activity. Como o template de Activity do Android variou nas últimas versões do ADT, vamos optar por criar uma Activity do zero.

Criando nova `Activity` do zero

Primeiro vamos criar o layout do nosso formulário. Crie um novo layout e chame-o de **formulario.xml**, agora precisamos fazer com que ele atinja o seguinte resultado:

Para isso iremos criar vários componentes orientados verticalmente. Por esse motivo a tag raiz de nosso layout deverá ser um `LinearLayout` com a orientação vertical:

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <!-- outros elementos em breve aqui! -->

</LinearLayout>
```

Dentro do `LinearLayout` o primeiro elemento que desejamos colocar é uma imagem para que mais tarde possamos ter a foto do aluno cadastrado.

Para isso vamos criar um `ImageView`. Ele deve estar centralizado horizontalmente na tela portanto vamos configurar o atributo `layout_gravity` de nosso `ImageView`:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
            android:src="@drawable/ic_launcher" />

        <!-- outros elementos em breve aqui! -->

    </LinearLayout>
```

Devemos criar varias entradas de dados, ou seja `EditText` s e cada um deles deve ter um texto indicativo de qual informação desejamos que seja preenchida. Ou seja para cada `EditText` vamos criar um `TextView` logo acima.

O último elemento da tela será um `Button` com o texto `Gravar` . Então nosso **layout** ficará dessa maneira:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:src="@drawable/ic_launcher" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nome:" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <!-- outros campos -->

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Gravar" />

</LinearLayout>
```

Logo acima do botão gravar precisaremos pedir que o usuário escolha a nota to aluno. No lugar de pedir para que ele digite em um `EditText` vamos fazer a seleção em um `RatingBar` com um valor máximo de `5` .

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <!-- outras tags criadas previamente -->

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Nota:" />

    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:numStars="5" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Gravar" />

</LinearLayout>
```

O problema que podemos enfrentar é a variedade de dispositivos Android disponíveis no mercado. Para os dispositivos de tela pequena não será possível visualizar todos os componentes que colocamos no layout e nossa tela não foi configurada para permitir **scroll**.

Vamos melhorar a compatibilidade de nossa *app* fazendo com que o `LinearLayout` fique dentro de uma `ScrollView` que será a nova tag raiz do layout **formulario.xml**:

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical" >

        <!-- outras tags criadas previamente -->

    </LinearLayout>
</ScrollView>
```

Repare que a tag raiz sempre deve conter o namespace (xmlns) do Android, para definir quais tags podemos usar!

Para que seja possível buscar esses elementos no código devemos criar **id**s para cada um dos campos de entrada de dados (os `EditText` s e o `RatingBar`).

Agora que já temos o layout, precisamos de uma Activity para vinculá-lo. Crie uma nova classe chamada `FormularioActivity` e a faça herdar de `Activity` conforme segue:

```
public class FormularioActivity extends Activity {

}
```

Sobrescreva o método `onCreate` e com o método `setContentView` vincule nosso layout a esta Activity.

```
public class FormularioActivity extends Activity {

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

Vamos informar o Android de que temos uma nova tela na nossa aplicação. Faremos isso adicionando o seguinte código dentro do **AndroidManifest.xml** dentro da tag `<Application ...> ... </Application>` .

```
<Activity android:name="br.com.caelum.cadastro.FormularioActivity" />
```

Criando Menus

Agora que possuímos duas telas, precisamos encontrar alguma forma de fazer o usuário navegar entre elas.

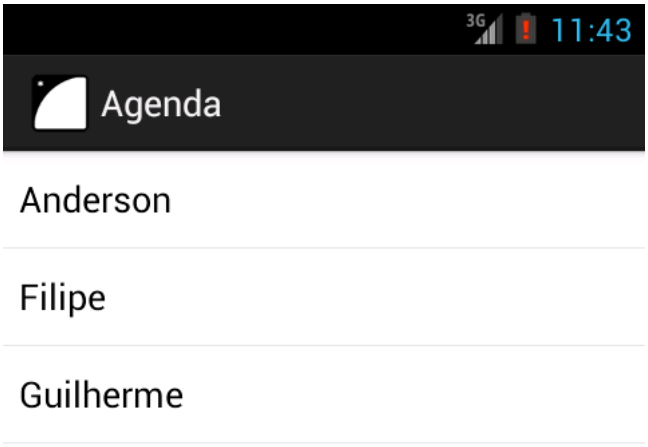
Na maioria das vezes precisamos de diversas telas numa mesma aplicação e de menus de acesso a funções específicas.

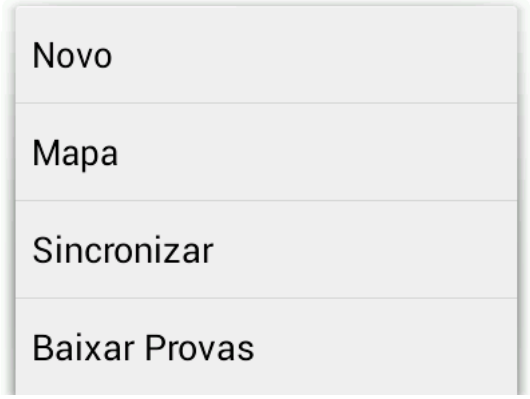
Felizmente, toda `Activity` do Android já vem preparada para lidar com menus. Para criá-lo, basta sobrescrever o método `onCreateOptionsMenu(Menu menu)` da `Activity` . Faremos isso na nossa `ListaAlunosActivity` .

Quando criamos um item no menu de uma aplicação, quem se encarrega de encaixá-lo na posição correta da tela é o próprio sistema, você não precisa se preocupar. Para isso, dentro do `onCreateOptionsMenu(Menu menu)` , faríamos:

```
menu.add("Novo");
menu.add("Mapa");
menu.add("Sincronizar");
menu.add("Baixar Provas");
menu.add("Preferências");
```

Para que o menu de opções apareça precisamos apenas clicar no botão `Menu` de nosso aparelho:





Nas versões mais atuais do Android, o menu é exibido na ActionBar do aplicativo.

Como comportamento padrão, o sistema colocará como nome do botão o parâmetro passado no método `add`, mas, se preferir, pode-se trocar por um ícone.

Basta chamar no item de menu recém-criado, que é devolvido pelo método `add`, o método `setIcon`, passando um `R.drawable.nomeDaImagem`.

```
MenuItem mapa = menu.add("Mapa");
mapa.setIcon(R.drawable.mapa);
```

O método `onCreateOptionsMenu(Menu menu)` deve retornar `true` (ou o resultado do `super`) para indicar que o menu deve aparecer na tela.

Apesar da possibilidade da criação programática do menu, como ele é um elemento de tela damos a preferência de criar seus itens em um `xml`.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/menu_novo"
        android:icon="@drawable/ic_novo"
        android:title="Novo"/>
    <item
        android:id="@+id/menu_mapa"
        android:icon="@drawable/ic_mapa"
        android:title="Mapa"/>
    <item
        android:id="@+id/menu_enviar_alunos"
        android:icon="@drawable/ic_enviar"
        android:title="Sincronizar"/>
    <item
        android:id="@+id/menu_receber_provas"
        android:icon="@drawable/ic_receber"
        android:title="Baixar Provas"/>
    <item
```

```
        android:id="@+id/menu_preferencias"
        android:icon="@drawable/ic_preferencias"
        android:title="Preferências"/>
</menu>
```

Repare inclusive que já estamos escolhendo arquivos dentro da pasta `drawable` para ícones de nossos itens de menu. Faça download das imagens usadas para os menus clicando aqui (https://s3.amazonaws.com/caelum-online-public/FJ-57/imagens.zip).

Agora precisamos fazer o `Android` criar elementos na tela - `Views` - para cada item do menu descrito no xml.

O Android precisa ter a capacidade de ler os `xmls` que criamos e construir objetos do tipo `View` a partir das tags. Essa é a especialidade dos **Inflaters**.

No caso específico de um `xml de menu` podemos utilizar um `Inflater` especializado em interpretar esse tipo de arquivo, trata-se do **MenuInflater**.

No método `onCreateOptionsMenu` podemos fazer uma chamada a `getMenuInflater()`, um método da classe `Activity` que nos dá acesso a um `MenuInflater`.

Repare que o método `onCreateOptionsMenu` já nos fornece um objeto do tipo **Menu**, basta utilizarmos o inflater para ler nosso xml e criar a partir deles os itens que serão colocados no menu. Para isso podemos utilizar o método mais importante de um `Inflater`: o `inflate`.

```
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.meu_arquivo_de_menu, menu);
}
```

A abordagem de utilizar botões físicos no aparelho foi um diferencial competitivo dos aparelhos `Android` em relação ao `iPhone`. Enquanto o produto da `Apple` baseava toda sua usabilidade em `Gestures` (inclusive o botão voltar, que ocupa espaço na tela) os aparelhos Android podiam combinar cliques na tela com botões no aparelho, possibilitando a fabricação de aparelhos com telas bem menores e utilizando telas com sensibilidade inferior, se necessário.

Hoje a qualidade dos aparelhos Android evoluiu e o mercado consumidor optou por utilizar principalmente aparelhos com telas maiores, fazendo com que a usabilidade nos aparelhos Android fosse alterada. Hoje a tendência é minimizar a utilização de botões físicos no aparelho (boa parte dos devices mais novos já não possuem botões como o `Search` e a tendência é minimizar ainda mais).

Para alguns dispositivos mais antigos, quando uma tela possui um `OptionsMenu` não existe nenhum indicativo visual na tela dessa presença, fazendo com que a usabilidade da app seja prejudicada.

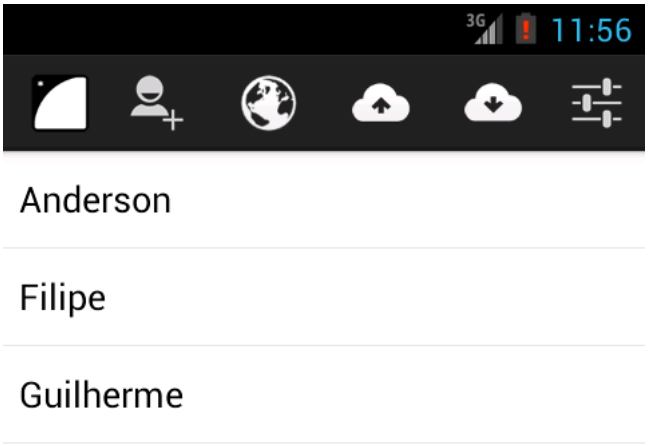
Em nossa aplicação vamos seguir a tendência do Android e colocar o Menu em um elemento visual na tela através do uso de um componente que surgiu na versão 3.0 do Android: a `Action Bar`.

Para configurarmos os itens de menu para aparecerem na `Action Bar` simplesmente configuramos o atributo `showAsAction` da tag `item`:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:id="@+id/menu_novo"
        android:showAsAction="always"
        android:icon="@drawable/ic_novo"
        android:title="Novo"/>

    <item
        android:id="@+id/menu_mapa"
        android:showAsAction="always"
        android:icon="@drawable/ic_mapa"
        android:title="Mapa"/>

    <!-- outros itens -->
</menu>
```

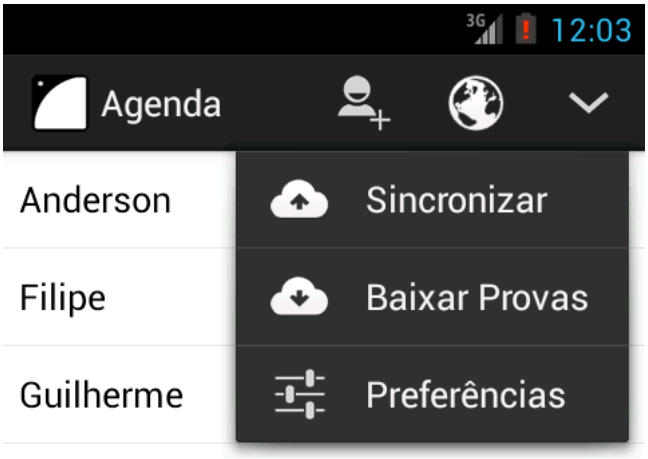


Para evitar que o nome de nossa aplicação desapareça nos aparelhos de tela pequena e para tornar a tela mais agradável, vamos deixar visíveis no menu principal apenas as opções mais úteis. Para isso, vamos criar um submenu de opções:

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

    <item
        android:id="@+id/menu_novo"
        android:icon="@drawable/ic_novo"
        android:showAsAction="always"
        android:title="Novo"/>
    <item
        android:id="@+id/menu_mapa"
        android:icon="@drawable/ic_mapa"
        android:showAsAction="always"
        android:title="Mapa"/>
    <item
        android:icon="@drawable/ic_opcoes"
        android:showAsAction="always"
        android:title="Mais opções">
        <menu>
            <item
                android:id="@+id/menu_enviar_alunos"
                android:icon="@drawable/ic_enviar"
                android:showAsAction="always"
                android:title="Sincronizar"/>
            <item
                android:id="@+id/menu_receber_provas"
                android:icon="@drawable/ic_receber"
                android:showAsAction="always"
                android:title="Baixar Provas"/>
            <item
                android:id="@+id/menu_preferencias"
                android:icon="@drawable/ic_preferencias"
                android:showAsAction="always"
                android:title="Preferências"/>
        </menu>
```

```
    </item>
</menu>
```



Navegando entre telas

Nem só de Toasts e menus viverá a nossa aplicação. Precisamos chamar outras Activities para trabalhar com outras telas do nosso sistema. Dessa forma, será possível navegar pelas telas da nossa aplicação usando os menus e botões.

Para darmos comportamento ao nosso menu podemos sobrescrever o método `onOptionsItemSelected(MenuItem item)` , este método é chamado pelo Android sempre que um menu de opções é clicado.

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

O parâmetro recebido é justamente o item de menu clicado, por ele podemos identificar qual menu foi clicado para decidirmos o que fazer:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.novo:
        //Abrir o formulário aqui
        break;

    default:
        break;
    }
```

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

Para abrir uma outra `Activity` é simples! Basta fazer com que o clique em determinado menu chame o método `startActivity(...)`, passando para ele um **Intent**, ou seja, uma intenção de ir para outra `Activity`.

O `Intent`, por sua vez, precisará receber o objeto de contexto atual (usualmente o `this`) e a classe que controla a tela para onde queremos ir.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.novo:
        Intent intent = new Intent(ContextoAtual.this, NovaActivity.class);
        startActivity(intent);

        break;

    default:
        break;
    }


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

Veremos em capítulos posteriores que podemos abrir outras funcionalidades através das `Intent`s, e não apenas outras activities. Aqui, pedimos **explicitamente** que a `NovaActivity` seja iniciada.