DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Ubiratan Roberte Cardoso Passos

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

Componentes de tela do usuário

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar os principais componentes de tela do usuário.
- Explicar o uso dos principais componentes de tela do usuário.
- Demonstrar o uso dos principais componentes de tela do usuário.

Introdução

Neste capítulo, você conhecerá os principais componentes e recursos disponibilizados pela plataforma Android para trabalhar com telas de usuário e compreenderá o uso desses componentes.

Principais componentes de tela do usuário

A plataforma Android oferece diversos componentes visuais sofisticados, principalmente quando temos em conta como se desenvolviam os aplicativos em aparelhos celulares antigos, sendo possível encontrar inúmeros componentes que promovem efeitos impressionantes, permitindo a sua utilização e personalização. Nesse sentido, a plataforma Android tem se tornado padrão de referência em quantidade e qualidade de componentes visuais.

Os componentes visuais no Android, conforme mencionado em capítulo anterior, são definidos a partir de *tags* no arquivo XML de leiaute e têm correspondência para que possamos utilizá-los no código (*activity*).

Antes de aprofundarmos sobre os componentes de interface, é importante entender o conceito de *view* no Android: de acordo com Cordeiro (2016, documento *on-line*), em resumo, a *view* é basicamente um retângulo na tela que exibe algum conteúdo, podendo constituir uma imagem, um texto, um botão ou qualquer outro elemento que o aplicativo seja capaz de mostrar, e formando, com todas as demais *views*, o leiaute da interface. Tudo o que se

vê ou com que interagimos no aplicativo é chamado de interface de usuário, ou UI (*user interface*).

Há diferentes tipos de *views* no Android, mas inicialmente enfatizaremos três (CORDEIRO, 2016, documento *on-line*):

1. `TextView`: exibe algum texto;
2. `ImageView`: exibe uma imagem;
3. `Button`: exibe um botão.

Como falamos, cada *view* é um retângulo na tela, mas onde se localizam todos esses retângulos? Tecnicamente, eles estão invisíveis, mas, na Figura 1, foram destacados para que possamos observar os limites de cada um deles.

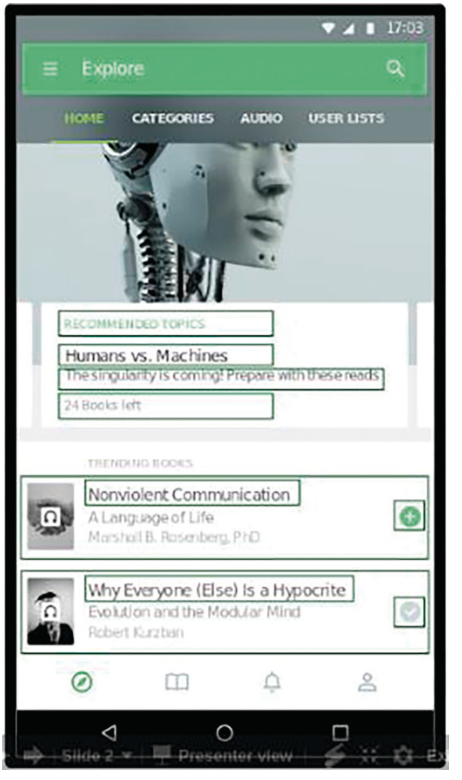

Figura 1. Imagem com *views* destacadas.
Fonte: Cordeiro (2016, documento *on-line*).

Cada aplicativo, nesse sentido, pode ser dividido em *views* individuais, ou seja, as *views* constituem os componentes básicos para construir o leiaute do seu aplicativo (CORDEIRO, 2016, documento *on-line*).

Tipos de componentes de tela mais utilizados

Pelo fato de as *views* que serão utilizadas em um leiaute representarem um ponto fundamental do desenvolvimento Android, analisaremos brevemente os componentes mais comuns empregados na maioria dos leiautes dos aplicativos (CORDEIRO, 2016, documento *on-line*).

Com as opções a seguir, já é possível trabalharmos com leiautes bonitos e usáveis (CORDEIRO, 2016, documento *on-line*):

- `TextView` — apresenta um texto formatado;
- `ImageView` — exibe um uma imagem;
- `Button` — efetua uma ação ao ser executado;
- `ImageButton` — exibe uma imagem com o comportamento de um botão;
- `EditText` — campo de texto editável para a entrada de dados;
- `ListView` — lista de itens que contém outras *views*.

A seguir, resumiremos cada um dos componentes apresentados e como podemos utilizá-los em nossos aplicativos.

Fique atento

Quando começamos a desenvolver a interface gráfica para um aplicativo, independentemente da linguagem de programação ou da plataforma de desenvolvimento, é obrigatório prevermos a interação entre os usuários e os componentes visuais, lembrando que estes podem ser botões, caixas de seleção, imagens, vídeos, etc. No *link* a seguir, você acessar o guia de *design* da Google para IU.

https://qrgo.page.link/x13fB

Componentes de tela mais utilizados

TextView

Tem o objetivo de exibir um texto na tela de um aplicativo Android, contendo uma lógica bem complexa para possibilitar a exibição de texto formatado, *hyperlinks*, números de telefone, e-mails, entre outros (CORDEIRO, 2016, documento *on-line*):

```
<TextView
    android:text="Olá Mundo!"
    android:background="@android:color/darker _ gray"
    android:layout _ width="150dp"
    android:layout _ height="75dp" />
```

ImageView

Com a função específica de exibir imagens em tela de aplicativos Android, pode ser usado para exibir recursos armazenados no aplicativo e imagens que realizam *downloads* da internet (CORDEIRO, 2016, documento *on-line*):

```
<ImageView
    android:src="@drawable/imagem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:scaleType="center"/>
```

Button

Um dos controles mais comuns disponíveis para aplicativos, funciona respondendo a cliques do usuário e trazendo um método em seu código para que seu aplicativo retorne o desejado quando o usuário pressiona o botão (CORDEIRO, 2016, documento *on-line*):

```
<Button
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:text="Meu Botão"
     android:onClick="clicar" />
```

ImageButton

`Button` com uma `ImageView`, junta ambas as características em um só componente:

```
<ImageButton
    android:id="@+id/imageButton"
    android:layout _ width="wrap _ content"
    android:layout _ height="wrap _ content"
    android:src="@drawable/imagem" />
```

EditText

Extensão do componente `TextView`, permite que os usuários editem ou entrem com um texto digitado pelo teclado (CORDEIRO, 2016, documento *on-line*):

```
<EditText
    android:id="@+id/email"
    android:layout _ width="wrap _ content"
    android:layout _ height="wrap _ content"
    android:hint="Preencha seu melhor email..."
    android:inputType="textEmailAddress" />
```

ListView

Trata-se de um componente empregado para exibir uma coleção de itens — selecionados individualmente, para exibição de mais detalhes ou realização de uma ação relacionada a esse item —, de modo linear em uma coluna única. Para adicionar um item a lista, utiliza-se a classe `ArrayAdapter` do Android, que tem a finalidade de adaptar itens em uma ListView (CORDEIRO, 2016, documento *on-line*).

É importante ressaltar que se trata de uma classe cuja implementação está pronta do Android, não sendo possível manipular esse *adapter*.

Nesse sentido, podemos implementar também o nosso próprio *adapter* (CORDEIRO, 2016, documento *on-line*):

```
<ListView xmlns:android="http://schemas.android.com/apk/res/
android"
    android:id="@+id/listview"
    android:layout _ width="match _ content"
    android:layout _ height="match _ content" />
```

O Android dispõe de um modelo de componentes bem vasto e sofisticado, para trabalharmos a *interface* com os usuários.

Trabalhar com componentes visuais no Android

Para o leiaute, é necessário instanciarmos objetos *view* no nosso código e começarmos a criar uma estrutura. A forma mais usual e efetiva de trabalharmos com leiaute se dá com arquivos XML, que oferecem uma estrutura legível muito parecida com a do HTML.

Saiba mais

São sete as formas de tratamento de evento com usuários da interface de programação do Android (CORDEIRO, 2016, documento *on-line*), conforme listado abaixo.

- Clique.
- Clique longo.
- Menu de contexto.
- Evento de toque.
- Mudança de foco.
- Evento de tecla.
- Item selecionado.

Para que seja exibido, um nome de um elemento XML é relacionado à classe Android que representa, ou seja, um elemento `<TextView>` cria um *widget* `TextView` na UI e um elemento `<LinearLayout>` cria um grupo de exibições de `LinearLayout` (CORDEIRO, 2016, documento *on-line*).

Em um leiaute vertical simples, a exibição de texto e o botão se assemelham ao exemplo exibido a seguir.

Exemplo

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/androi
d"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >
  <TextView android:id="@+id/text"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="I am a TextView" />
  <Button android:id="@+id/button"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="I am a Button" />
</LinearLayout>
```

Fonte: Adaptada de Visão... (2019, documento *on-line*).

Componentes visuais como `EditText` e `Button` têm *listeners*, que nos permitem tratar eventos disparados por ações realizadas pelos usuários.

Dessa forma, no método relacionado ao evento, necessitamos sempre referenciar uma *view* como parâmetro, como no exemplo a seguir.

Exemplo

```
Button btn = (Button) findViewById(R.id.botaoMsg);
btn.setOnClickListener(new OnClickListener()
{
  @Override
  public void onClick(View view)
  {
  EditText edtMsg = (EditText) findViewById(R.id.edtMsg);

  String msg = edtMsg.getText().toString();

  if (!msg.trim().isEmpty()) Toast.makeText(getApplicationContext(), msg,
Toast.LENGTH_LONG).show();
  else Toast.makeText(getApplicationContext(), "Digite uma mensagem!",
Toast.LENGTH_SHORT).show();
  }
 });
```

Ainda que trabalhar com componentes visuais não seja difícil e não exija muito trabalho, já que muitos recursos são possíveis via leiaute, é importante prestarmos atenção nos seguintes pontos:

- atentar-se às transparências, principalmente quando uma camada de tela está sobreposta de outra parte de tela, já que, nesse caso, os componentes são lidos por trás da tela;
- pensar na usabilidade como parte da fase de concepção e desenvolvimento, pois, assim, será bem mais fácil melhorar a experiência do usuário.

Link

Acesse o *link* a seguir e veja como inserir componentes Android no arquivo XML.

https://qrgo.page.link/BxYzX

Fique atento

Pensando no problema frequente de aplicativos terem listas para apresentar seu conteúdo de forma eficiente, o que pode não agradar o usuário se mal implementados pelo desenvolvedor, o Android oferece um componente poderoso: o `RecyclerView` (ARAUJO, 2018, documento *on-line*).

Referências

ARAUJO, R. RecyclerView, o robusto componente de listas do Android. *Venturus – Plantando e colhendo insights: aqui estão nossas melhores ideias*, [*S. l.*], 8 fev. 2018. Disponível em: https://medium.com/venturus/recyclerview-o-robusto-componente-de-listas-do-android-aa50ef11538d. Acesso em: 25 jun. 2019.

CORDEIRO, F. Saiba como usar as Android Views da forma correta. *AndroidPro*, [*S. l.*], 18 jul. 2016. Disponível em: https://www.androidpro.com.br/blog/desenvolvimento-android/android-views-intro/. Acesso em: 25 jun. 2019.

VISÃO Geral da IU. *Android Developers*, [*S. l.*], 2019. Disponível em: https://developer.android.com/guide/topics/ui/overview.html?hl=pt-BR. Acesso em: 25 jun. 2019.

Leituras recomendadas

ANDROID Developers. [*S. l.*: *S. n.*], 2019. Disponível em: https://developer.android.com. Acesso em: 25 jun. 2019.

BURNETTE, E. *Hello, Android*: introducing Google's mobile development platform. 4. ed. Raleigh: The Pragmatic Programmers, 2015. 225 p.

DEITEL, P.; DEITEL, H.; WALD, A. *Android 6 para programadores*: uma abordagem baseada em aplicativos. 3. ed. Porto Alegre: Bookman, 2016. 618 p.

LEE, W. M. *Beginning Android tablet application development*. Indianapolis: Wrox, 2011. 288 p.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS