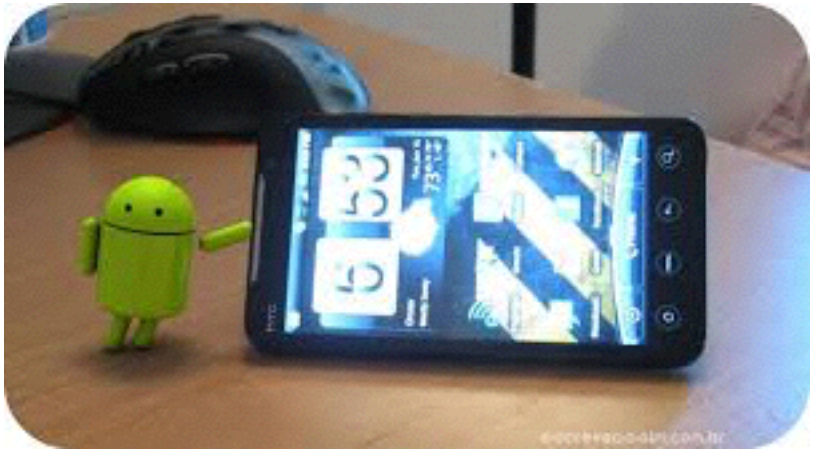


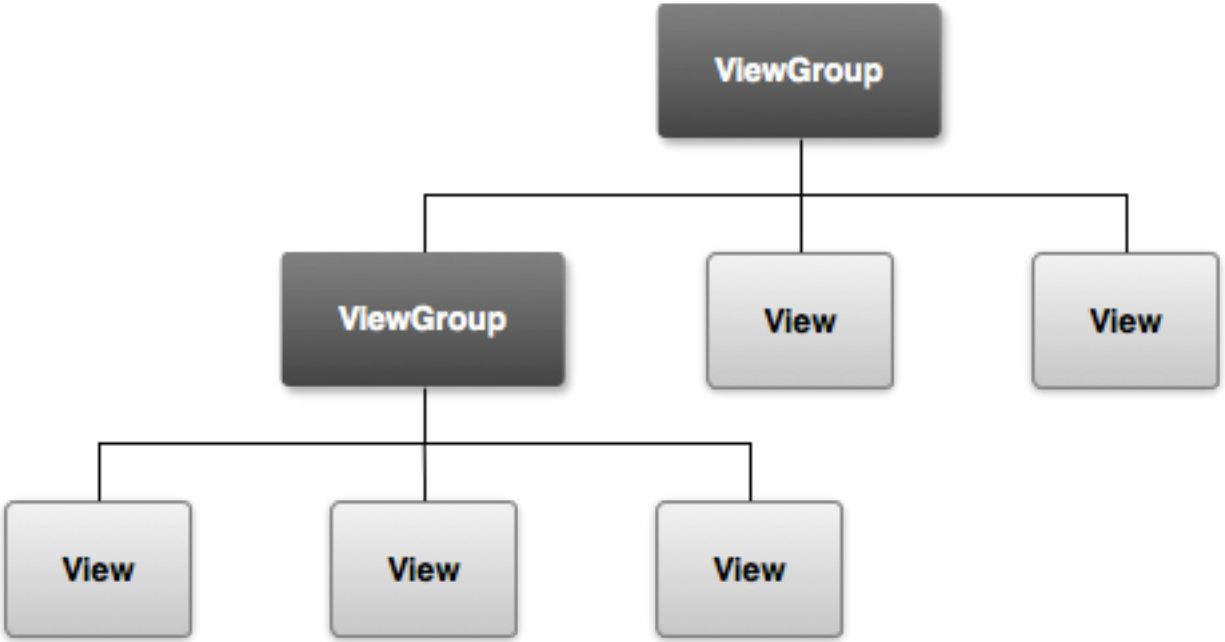
Interface do Usuário

Disciplina: Linguagem de Programação III
Professor: Rafael Brazão

USJ

Visão geral da Interface do Usuário (IU)

- Todos os elementos da interface do usuário em um aplicativo para Android são criados usando objetos **View** e **ViewGroup**.
- Uma View é um objeto que desenha algo na tela com o qual o usuário pode interagir. Um ViewGroup é um objeto que contém outros objetos View (e ViewGroup) para definir o layout da interface



USJ

Visão geral da Interface do Usuário (IU)

- Para declarar o layout, é possível instanciar objetos View no código e começar a criar uma árvore. Mas a forma mais fácil e efetiva de definir o layout é com um arquivo XML.
- O nome de um elemento XML para uma view é respectivo à classe do Android que ele representa. Portanto, um elemento <TextView> cria um widget TextView na IU e um elemento <LinearLayout> cria um grupo de view de LinearLayout

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

USJ

Componentes de entrada de dados

Tipo de controle	Descrição	Classes relacionadas
Botão	Um botão de pressão que pode ser pressionado ou clicado pelo usuário para realizar uma ação.	`Button`
Campo de texto	Um campo de texto editável. É possível usar o widget `AutoCompleteTextView` para criar um widget de entrada de texto que forneça sugestões para preenchimento automático	`EditText`, `AutoCompleteTextView`
Caixa de seleção	Uma chave liga/desliga que pode ser alternada pelo usuário. Use caixas de seleção ao apresentar aos usuários um grupo de opções selecionáveis que não sejam mutualmente exclusivas.	`CheckBox`
Botão de opção	Similar às caixas de seleção, exceto que somente uma opção pode ser selecionada no grupo.	`RadioGroup` `RadioButton`
Botão de alternância	Um botão liga/desliga com um indicador de luz.	`ToggleButton`
Controle giratório	Uma lista suspensa que permite que os usuários selecionem um valor de um conjunto.	`Spinner`
Seletores	Uma caixa de diálogo para que os usuários selecionem um valor para um conjunto usando botões para cima/para baixo ou via gesto de deslizar	`DatePicker`, `TimePicker`

USJ

Listeners de evento

- Um listener de evento é uma interface na classe View que contém um único método de retorno de chamada. Esses métodos serão chamados pela estrutura do Android, quando a View para a qual a listener estiver registrada for ativada pela interação do usuário com o item na IU.
- Inclusos nas interfaces listeners de evento estão os seguintes métodos de retorno de chamada:
  - onClick(): de View.OnClickListener
  - onLongClick(): de View.OnLongClickListener
  - onFocusChange(): de View.OnFocusChangeListener
  - onKey(): de View.OnKeyListener
  - onTouch(): de View.OnTouchListener
  - onCreateContextMenu(): de View.OnCreateContextMenuListener

USJ

Listeners de evento

Exemplo com o evento onClick():

```
// Create an anonymous implementation of OnClickListener
private OnClickListener botaoListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(botaoListener);
    ...
}
```

USJ

Toasts

- Um componente Toast apresenta uma mensagem em uma pequena caixa de pop-up.
- Ele ocupa somente o espaço necessário para exibir a mensagem, deixando a activity atual visível e interativa
- Exemplo:

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

- Como posicionar um Toast:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

USJ

Layouts

- O layout define a estrutura visual para uma interface do usuário, como a IU de uma atividade ou de um widget de aplicativo
- Cada arquivo de layout deve conter exatamente um elemento raiz, que deve ser um objeto View ou ViewGroup. Com o elemento raiz definido, é possível adicionar objetos ou widgets de layout extras como elementos filho para construir gradualmente uma hierarquia de View que define o layout
- Ao compilar o aplicativo, cada arquivo de layout XML é compilado em um recurso View. Deve-se carregar o recurso de layout do código do aplicativo na implementação de retorno de chamada Activity.onCreate(). Para isso, chame setContentView(), passando a referência para o recurso de layout na forma: R.layout.layout_file_name

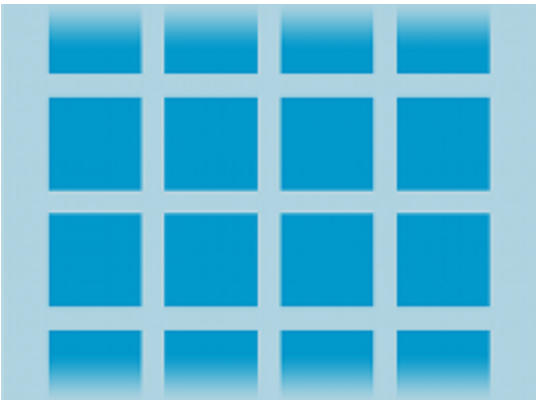
USJ

Layouts

- Alguns Layouts mais comuns:

LinearLayout

RelativeLayout



GridView

USJ

LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/to" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/subject" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:gravity="top"
        android:hint="@string/message" />
    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="@string/send" />
</LinearLayout>
```

USJ

RelativeLayout

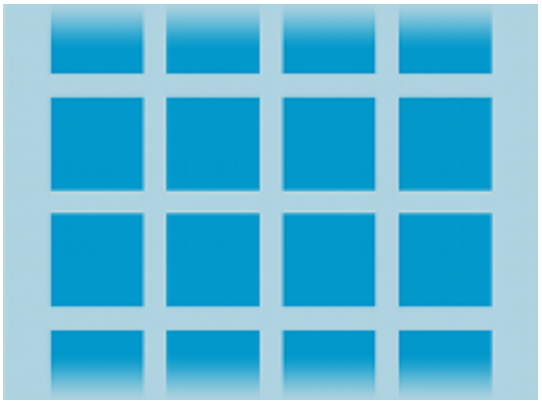
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```

USJ

Criação de Layouts com um adapter

- Quando o conteúdo do layout é dinâmico ou não predeterminado, é possível usar um layout que use uma subclasse de AdapterView para preencher o layout com uma view em tempo de execução
- Uma subclasse da classe AdapterView usa um Adapter para agrupar dados ao seu layout
- O Adapter se comporta como um intermediário entre a fonte dos dados e o layout do AdapterView — o Adapter recupera os dados (de uma fonte como uma matriz ou uma consulta de banco de dados) e converte cada entrada em uma vista que pode ser adicionada ao layout do AdapterView.

ListView



GridView

USJ

Preenchimento de uma AdapterView com dados

- É possível preencher um AdapterView como ListView ou GridView agrupando-se a instância do AdapterView a um Adapter, o que recupera dados de uma fonte externa e cria uma View que representa cada entrada de dados
- O Android oferece diversas subclasses de Adapter que são úteis para recuperar diferentes tipos de dados e criar vistas de um AdapterView. Os dois adaptadores mais comuns são:
  - **ArrayAdapter**
    - ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    - android.R.layout.simple_list_item_1, myStringArray);
    - ListView listView = (ListView) findViewById(R.id.listview);
    - listView.setAdapter(adapter);
  - **SimpleCursorAdapter**

USJ

Tratamento de eventos de clique em um AdapterView

```
// Create a message handling object as an anonymous class.
private OnItemClickListener mMessageClickedHandler =
  new OnItemClickListener() {
    public void onItemClick(AdapterView parent,
      View v, int position, long id) {
        // Do something in response to the click
    }
};

listView.setOnItemClickListener(mMessageClickedHandler);
```

USJ

Fim da aula

- Obrigado!

USJ