

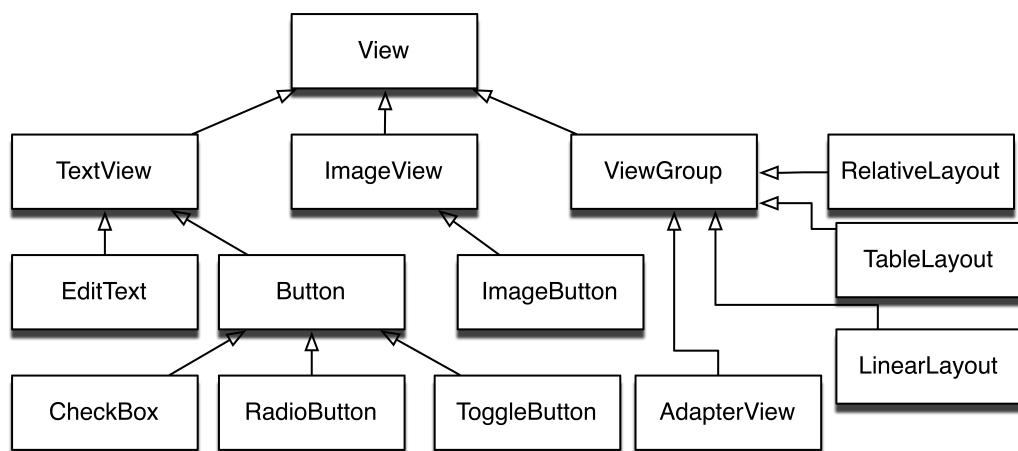
# ANDROID

VIEWS e WIDGETS



## View

- A classe `android.view.View` é a raiz da hierarquia de componentes visuais



# View

- Atributos da **View**: todos os widgets possuem estes atributos
  - **android:id** : id a ser usado para recuperar o widget, "@+id/id\_do\_elemento"
  - **android:background** : uma imagem ou uma cor para pintar o fundo do elemento
  - **android:clickable** : (true/false) se o elemento reage a eventos de clique
  - **android:focusable** : (true/false) se o elemento pode receber o foco
  - **android:longClickable** : (true/false) se o elemento reage a eventos de clique longo
  - **android:onClick** : nome do método a ser invocado quando o elemento é clicado

Prof. Razer A N R Montaño

2017

3

# Influência dos Layouts

- Para os widgets, atributos iniciando com **layout** dizem ao **ViewGroup** (layout) para acertar os valores
  - **android:layout\_width**: largura do elemento. Valores:
    - **match\_parent**: tamanho deve ser o mesmo do elemento pai
    - **wrap\_content**: tamanho suficiente para mostrar o conteúdo
    - Valor: px, dp, sp, in, mm
  - **android:layout\_height**: altura do elemento. Valores como os da largura
  - **android:layout\_below**: usando um RelativeLayout, indica que este elemento deve ficar abaixo de outro

Prof. Razer A N R Montaño

2017

4

# Medidas

- Medidas

- **px** : pixels, corresponde a pixels da tela (não recomendado)
- **dp** : pixels independentes de densidade, unidade abstrata baseada na densidade física da tela.
  - Ex, se a resolução da tela é 160 dpi (*dots per inch*), 1 dp = 1 pixel no total de 160.
  - Em uma resolução maior, a quantidade de pixels para desenhar 1 dp é adequado para a densidade da tela. O mesmo para resolução menor
  - $\text{px} = \text{dp} * (\text{dpi} / 160)$
  - A razão entre dp e pixel muda conforme a densidade da tela.
  - SEMPRE USAR PARA MEDIDAS
- **sp** : pixels independentes de escala, como o **dp** mas baseado no tamanho da fonte do usuário. Melhor para setar tamanhos de fontes, pois se ajusta à densidade da tela e à escolha do usuário. SEMPRE USAR PARA FONTES
- **pt** : pontos, 1/72 de uma polegada, baseado no tamanho físico da tela
- **in** : polegadas, baseado no tamanho físico da tela, 1 in = 2,54 cm
- **mm** : milímetros, baseado no tamanho físico da tela

Prof. Razer A N R Montaño

2017

5

# Compatibilidade entre Telas

- Dispositivos são separados em categorias de densidade

Densidade	Densidade da Tela	Tamanho Físico	Tamanho do Pixel	Proporção de Recursos (ex. em px)
ldpi	120 dpi	0.5 x 0.5 in	0.5 in x 120 dpi = 60 x 60 px	0.75 (75 x 75)
mdpi	160 dpi	0.5 x 0.5 in	0.5 in x 160 dpi = 80 x 80 px	1 (base) (100 x 100)
hdpi	240 dpi	0.5 x 0.5 in	0.5 in x 240 dpi = 120 x 120 px	1.5 (150 x 150)
xhdpi	320 dpi	0.5 x 0.5 in	0.5 in x 320 dpi = 160 x 160 px	2.0 (200 x 200)
xxhdpi	480 dpi	0.5 x 0.5 in	0.5 in x 480 dpi = 240 x 240 px	3.0 (300 x 300)

Prof. Razer A N R Montaño

2017

6

# Widgets

- **TextView**: mostra um texto
- **Button**: botão de ação
- **EditText**: edita um texto
- **RadioButton**: botão de rádio, mutuamente exclusivo
- **CheckBox**: caixa de seleção
- **ToggleButton**: botão de dois estados
- **Switch**: botão de dois estados deslizante
- **ImageButton**: botão com imagem
- **ImageView**: mostra uma imagem
- **ProgressBar**: barra de progresso de tarefa
- **SeekBar**: barra de progresso selecionável
- **DatePicker**: obtém uma data
- **Spinner**: seleção de itens em uma lista suspensa (combobox)
- **DatePicker**: obtém uma data

Prof. Razer A N R Montaño

2017



8

# TextView

Prof. Razer A N R Montaño

2017

# TextView

- Elemento usado para mostrar um Texto ou uma Imagem associada
- Link: <http://developer.android.com/reference/android/widget/TextView.html>
- Atributos
  - Todos os atributos de `View`
  - `android:text`: texto a ser apresentado, pode ser uma string ou "@string/nome\_da\_string"
  - `android:textColor`: cor do texto a ser apresentada, "#FF00FF" ou "@color/cor\_do\_texto"
  - `android:textSize`: tamanho do texto, recomendado unidade sp ou "@dimens/tamanho"
  - `android:textStyle`: estilo do texto a ser apresentado, valores:
    - "normal", "bold", "italic", "bold | italic"

# TextView

- Classe TextView
- Métodos
  - `getText()`: obtém o texto do elemento
  - `setText()`: seta o texto do elemento

# TextView

- Exemplo no Layout

```
<TextView  
    android:id="@+id/texto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/usuario" />
```

- Exemplo na Activity

```
TextView texto = (TextView) findViewById(R.id.texto);  
String str = texto.getText().toString();  
texto.setText("Olá mundo");
```

12

# EditText

# EditText

- Elemento usado para mostrar um campo para edição de texto
- Filho de **TextView**
- É um **TextView** configurado para ser editável
- Link: <http://developer.android.com/reference/android/widget/EditText.html>
- Atributos
  - **android:maxLength**: tamanho máximo do texto a ser digitado
  - **android:singleLine**: "true"/"false", se digita em uma linha correndo para o lado (true) ou pulando a linha (false)
  - **android:inputType**: tipo do texto a ser entrado
    - **number**
    - **datetime**
    - **text**
    - **textPassword**

# EditText

- Classe **EditText**
- Métodos
  - **getText()**: obtém o texto do elemento
  - **setText()**: seta o texto do elemento

# EditText

- Exemplo no Layout

```
<EditText  
    android:id="@+id/nome"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content" />
```

- Exemplo na Activity

```
EditText nome = (EditText) findViewById(R.id.nome);  
String str = nome.getText().toString();  
nome.setText("Razer");
```

16

# Button

# Button

- Elemento usado para mostrar um botão clicável
- Link: <http://developer.android.com/reference/android/widget/Button.html>
- Atributos
  - `android:text`
  - `android:onClick`: método da Activity que será invocado quando o botão for clicado

Prof. Razer A N R Montaño

2017

17

# Button

- Para tratar o clique pode-se usar
    1. Atributo `android:onClick`
    2. Fazer uma classe implementar `OnClickListener` e adicionar ao botão
  - 1) Atributo `android:onClick`  
`<Button`  
 `android:id="@+id/nome"`  
 `android:layout_width="match_parent"`  
 `android:layout_height="wrap_content"`  
 `android:onClick="calcular"/>`
  - Na Activity deve-se ter o método calcular recebendo uma View como parâmetro
- ```
public void calcular(View view) {
    // faz o que tem que fazer
}
```

Prof. Razer A N R Montaño

2017

18

# Button

- 2) Fazer uma classe implementar `OnClickListener` e adicionar ao botão

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

▪ Na Activity, onCreate(), deve-se adicionar um listener ao botão

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button b = (Button) findViewById(R.id.botao);
    b.setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                calcular(v);
            }
        });
}
```

Prof. Razer A N R Montaño

2017

19

# Button

- 2) Fazer uma classe implementar `OnClickListener` e adicionar ao botão
- Pode-se criar uma classe, ou fazer a própria Activity implementar `OnClickListener`
  - Deve-se implementar o método `onClick(View v)` na classe, que será invocado no evento de clique

```
public class MainActivity extends AppCompatActivity
    implements View.OnClickListener {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        Button b = (Button) findViewById(R.id.botao);
        b.setOnClickListener(this);
    }

    public void onClick(View v) {
        // Faz o que tem que fazer
    }
}
```

Prof. Razer A N R Montaño

2017

20



## Exercícios

1. Fazer um formulário com Nome, E-mail e Endereço, com dois botões, um para salvar e outro para limpar o formulário.
  - O botão **limpar** apaga o conteúdo dos campos (seta com valores "")
  - O botão **salvar** mostra em um TextView abaixo o nome, e-mail e endereço entrados
2. Fazer o exercício de Calculadora que está no Moodle

22

## RadioButton

# RadioButton

- Elemento usado para mostrar um botão de rádio
- Link: <http://developer.android.com/reference/android/widget/RadioButton.html>
- Deve ser colocado dentro de um **RadioGroup** para se ter o comportamento mutuamente exclusivo
  - Link: <https://developer.android.com/reference/android/widget/RadioGroup.html>
- Atributos
  - **android:text**
  - **android:checked**: se o botão de rádio inicia marcado

Prof. Razer A N R Montaño

2017

23

# RadioButton

- Exemplo no Layout
- ```
<RadioGroup
    android:id="@+id/radio_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <RadioButton
        android:id="@+id/plano"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/plano"
        android:checked="true" />
    <RadioButton
        android:id="@+id/sus"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="@string/sus" />
</RadioGroup>
```

Prof. Razer A N R Montaño

2017

24

# RadioButton

- Classe RadioButton
- Métodos
  - `isChecked()`: (booleano) obtém se o botão está marcado ou não
- Exemplo na Activity

```
RadioButton sus = (RadioButton) findViewById(R.id.sus);

if (sus.isChecked()) {
    // faz o que tem que fazer
}
```

# RadioButton

- Pode-se obter o `RadioButton` selecionado através do `RadioGroup`
  - Classe `RadioGroup`
- Métodos
  - `getCheckedRadioButtonId()`: retorna o id de recurso do `RadioButton` selecionado
- Exemplo na Activity

```
RadioGroup grupo = (RadioGroup) findViewById(R.id.radio_group);
int idSelecionado = grupo.getCheckedRadioButtonId();
RadioButton sel = (RadioButton) findViewById(idSelecionado);

String x = sel.getText().toString();
```



# CheckBox

Prof. Razer A N R Montaño

2017

## CheckBox

- Elemento usado para mostrar uma caixa de marcação
- Link: <https://developer.android.com/reference/android/widget/CheckBox.html>
- Atributos
  - **android:text**
  - **android:checked**: se a caixa inicia marcada

Prof. Razer A N R Montaño

2017



# CheckBox

- Exemplo no Layout

```
<CheckBox
    android:id="@+id/fumante"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="Fumante" />
```

# CheckBox

- Classe CheckBox
- Métodos
  - `isChecked()` : (booleano) obtém se o botão está marcado ou não
- Exemplo na Activity

```
CheckBox fumante = (CheckBox) findViewById(R.id.fumante);

if (fumante.isChecked()) {
    // faz o que tem que fazer
}
```



## Exercícios

1. No mesmo formulário do exercício anterior (Nome, E-mail e Endereço), adicionar botões de rádio para o Sexo (masculino, feminino, indefinido) e uma caixa de seleção (checkbox) para indicar se é fumante ou não. Deve-se ter dois botões, um para salvar e outro para limpar o formulário.
  - O botão **limpar** apaga o conteúdo dos campos (seta com valores "")
  - O botão **salvar** mostra em um TextView abaixo o nome, e-mail, endereço, sexo e se é ou não fumante.
2. Fazer o exercício **Conversão de Temperaturas** que está no Moodle.

Prof. Razer A N R Montaño

2017

31

32

## Switch

Prof. Razer A N R Montaño

2017

# Switch

- Elemento usado para mostrar um botão de dois estados
- Link: <https://developer.android.com/reference/android/widget/Switch.html>
- Atributos
  - **android:textOn**: texto a ser mostrado quando o botão está "ligado"
  - **android:textOff**: texto a ser mostrado quando o botão está "desligado"
  - **android:showText**: "true"/"false", se deve mostrar os textos de ligado/desligado
  - **android:checked**: se o botão inicia "ligado"

# Switch

- Exemplo no Layout

```
<Switch  
    android:id="@+id/ligado"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:textOn="Ligado"  
    android:textOff="Desligado"  
    android:checked="true"  
    android:showText="true" />
```

# Switch

- Classe Switch
- Métodos
  - `isChecked()`: (booleano) obtém se o botão está ligado
- Exemplo na Activity

```
Switch ligado = (Switch)findViewById(R.id.ligado);  
if (ligado.isChecked()) {  
    // Faz o que tem que fazer  
}
```

36

# ImageButton

# ImageButton

- Elemento usado para mostrar um botão com uma imagem, ao invés de texto
- Link: <https://developer.android.com/reference/android/widget/ImageButton.html>
- Atributos
  - **android:src**: a imagem a ser colocada no botão, "@drawable/imagem"
  - **android:onClick**: método a ser invocado no clique
- Para se ter uma imagem disponível
  - Copia-se a imagem para o diretório drawable

`<projetos>/MeuProjeto/app/src/main/res/drawable/botao.jpg`

- No atributo **android:src** usa-se

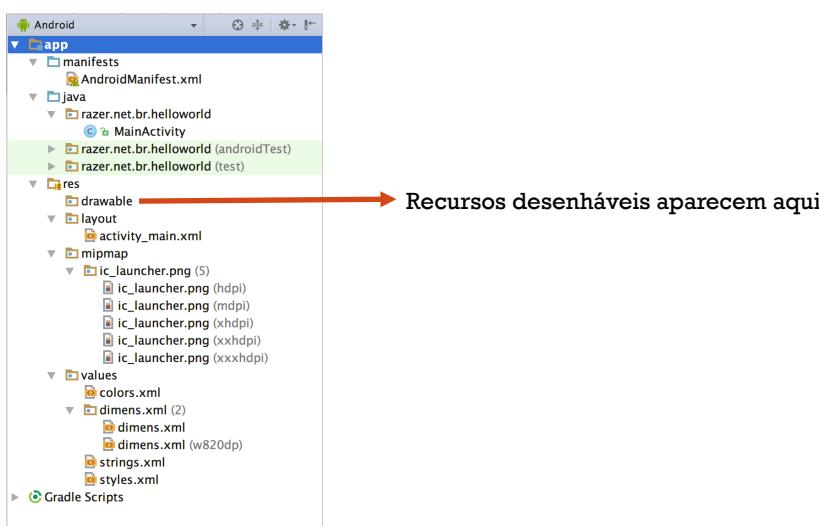
"@drawable/botao"

Prof. Razer A N R Montaño

2017

37

# Drawable



Prof. Razer A N R Montaño

2017

38

# Drawable

- Link: <https://developer.android.com/guide/topics/resources/drawable-resource.html?hl=pt-br>
- Compatibilidade com várias telas:  
[https://developer.android.com/guide/practices/screens\\_support.html?hl=pt-br](https://developer.android.com/guide/practices/screens_support.html?hl=pt-br)
- **Arquivo BITMAP:**
  - .png (preferencial), .jpg (aceitável), .gif (não recomendado)
- **Nine-Patch:**
  - É um arquivo PNG onde pode-se definir áreas esticáveis (atributo marcado com `wrap_content`)
  - Cria-se no próprio Android Studio
  - Extensão .9.png

Prof. Razer A N R Montaño

2017

39

# ImageButton

- Exemplo no Layout

```
<ImageButton
    android:id="@+id/botao_imagem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/botao"
    android:onClick="clickBotaoImagen"/>
```

- Exemplo na Activity

```
public void clickBotaoImagen(View view) {
    // Faz o que tem que fazer
}
```

Prof. Razer A N R Montaño

2017

40



## Exercícios

1. No mesmo formulário do exercício anterior (Nome, E-mail, Endereço, Sexo e Fumante), trocar os dois botões por botões de imagem (ImageButton).

Prof. Razer A N R Montaño

2017

41



## ImageView

Prof. Razer A N R Montaño

2017

# ImageView

- Elemento usado para mostrar uma imagem
  - Link: <https://developer.android.com/reference/android/widget/ImageView.html>
  - Atributos
    - **android:src**: a imagem a ser colocada no botão, "@drawable/imagem"
  - Para se ter uma imagem disponível
    - Copia-se a imagem para o diretório **drawable**
- <projetos>/MeuProjeto/app/src/main/res/drawable/botao.jpg
- No atributo **android:src** usa-se
- "@drawable/botao"

Prof. Razer A N R Montaño

2017

43

# ImageView

- Exemplo no Layout

```
<ImageView
    android:id="@+id/imagem"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/botao" />
```

Prof. Razer A N R Montaño

2017

44



# ProgressBar

Prof. Razer A N R Montaño

2017

## ProgressBar

- Elemento usado para mostrar um indicador visual de progresso de alguma operação
- Link: <https://developer.android.com/reference/android/widget/ProgressBar.html>
- Atributos
  - `android:max`: o valor máximo da barra
  - `android:progress`: o valor atual da barra
  - `style`: o estilo da barra. Por default é uma roda girando. Os estilos são:
    - `Widget.ProgressBar`: roda média girando
    - `Widget.ProgressBar.Horizontal`: barra horizontal
    - `Widget.ProgressBar.Small`: roda pequena girando
    - `Widget.ProgressBar.Large`: roda grande girando
    - `Widget.ProgressBar.Inverse`: roda média girando com cores invertidas
    - `Widget.ProgressBar.Small.Inverse`: roda pequena girando com cores invertidas
    - `Widget.ProgressBar.Large.Inverse`: roda grande girando com cores invertidas

Prof. Razer A N R Montaño

2017



# ProgressBar

- Exemplo no Layout

```
<ProgressBar  
    android:id="@+id/progresso"  
    style="@android:style/Widget.ProgressBar.Horizontal"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:max="100"  
    android:progress="45" />
```

Prof. Razer A N R Montaño

2017

47

# ProgressBar

- Classe **ProgressBar**
- Métodos
  - **setProgress (int)**: seta o progresso por um determinado valor
  - **incrementProgressBy (int)**: aumenta o progresso por um determinado valor
- Exemplo na Activity

```
ProgressBar p = (ProgressBar) findViewById(R.id.progresso);  
p.setVisibility(View.VISIBLE);
```

Prof. Razer A N R Montaño

2017

48



# SeekBar

Prof. Razer A N R Montaño

2017

## SeekBar

- Uma extensão de **ProgressBar** onde o usuário pode mover o indicador de progresso
- Link: <https://developer.android.com/reference/android/widget/SeekBar.html>
- Atributos
  - **android:max**: o valor máximo da barra
  - **android:progress**: o valor atual da barra

Prof. Razer A N R Montaño

2017



# SeekBar

- Exemplo no Layout

```
<SeekBar
    android:id="@+id/seek"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="100"
    android:progress="45" />
```

# SeekBar

- Classe SeekBar
- Métodos
  - **getProgress ()**: obtém o valor da barra de progresso
- Exemplo na Activity

```
SeekBar p = (SeekBar) findViewById(R.id.seek);
int valor = p.getProgress();
```



# Spinner

Prof. Razer A N R Montaño

2017

# Spinner

- Mostra uma lista de elementos para selecionar, mostrando somente o selecionado
- Link: <https://developer.android.com/reference/android/widget/Spinner.html>
- Atributos
  - **android:entries**: a lista de elementos, estática, configurada dentro de **strings.xml**

Prof. Razer A N R Montaño

2017



# Spinner

- Exemplo no Layout

```
<Spinner
    android:id="@+id/modo"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:entries="@array/modo" />
```

- Dentro de **strings.xml**

```
<string-array name="modo">
    <item>Dinheiro</item>
    <item>Cartão Débito</item>
    <item>Cartão Crédito</item>
</string-array>
```

Prof. Razer A N R Montaño

2017

55

# Spinner

- Classe Spinner
- Métodos
  - **getSelectedItem()**: obtém o item selecionado
  - **setAdapter()**: para setar uma lista de elementos
- Para carregar uma lista programaticamente:
  - Criar um **ArrayAdapter**
  - Setar ao **Spinner**
- Exemplo na Activity

```
Spinner spinner = (Spinner) findViewById(R.id.modo);

ArrayList<String> modos = new ArrayList<>();
modos.add("Dinheiro");
modos.add("Cartão Débito");
modos.add("Cartão Crédito");

ArrayAdapter<String> adapter = new ArrayAdapter<String>(
    this,
    android.R.layout.simple_spinner_dropdown_item,
    modos);

spinner.setAdapter(adapter);
```

Prof. Razer A N R Montaño

2017

56

# Spinner

- Para obter o elemento selecionado

```
Spinner spinner = (Spinner) findViewById(R.id.modo);  
String str = spinner.getSelectedItem().toString();
```



## Exercícios

1. Fazer o exercício Churrasco que está no Moodle.



# DatePicker

Prof. Razer A N R Montaño

2017

## DatePicker

- Mostra um componente para seleção de uma data
- Link: <https://developer.android.com/reference/android/widget/DatePicker.html>
- Atributos
  - **android: datePickerMode**: o modo de entrada da data
    - **spinner**: seleciona a data como spinners de dia, mês e ano
    - **calendar**: seleciona a data como um calendário

Prof. Razer A N R Montaño

2017



# DatePicker

- Exemplo no Layout

```
<DatePicker
    android:id="@+id/data"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android: datePickerMode="spinner" />
```

Prof. Razer A N R Montaño

2017

61

# DatePicker

- Classe DatePicker
- Métodos
  - `init(ano, mês, dia, changed listener)`: setar a data inicial do elemento
- Exemplo na Activity, setar a data inicial como a data atual

```
DatePicker data = (DatePicker) findViewById(R.id.data);

Calendar c = Calendar.getInstance();
int ano = c.get(Calendar.YEAR);
int mes = c.get(Calendar.MONTH);
int dia = c.get(Calendar.DAY_OF_MONTH);

data.init (ano, mes, dia, null);
```

Prof. Razer A N R Montaño

2017

62

# DatePicker

- Para obter a data selecionada:
  - Métodos get
  - Listener
- Para obter a data selecionada através de métodos

```
DatePicker data = (DatePicker) findViewById(R.id.data);
int dia = data.getDayOfMonth();
int mes = data.getMonth() + 1;
int ano = data.getYear();
Calendar calendar = new GregorianCalendar(
    data.getYear(),
    data.getMonth(),
    data.getDate());
Date dt = calendar.getTime();
```

Prof. Razer A N R Montaño

2017

63

# DatePicker

- Para obter a data selecionada através de *listener*
  - Cria-se um *listener*: OnDateChangedListener
  - Adiciona-se no init()

```
DatePicker data = (DatePicker) findViewById(R.id.data);
...
data.init(ano, mes, dia,
    new DatePicker.OnDateChangedListener() {
        @Override
        public void onDateChanged(DatePicker view, int year,
            int monthOfYear, int dayOfMonth) {
            // faz o que tem que fazer
            Calendar calendar = new GregorianCalendar(year, monthOfYear, dayOfMonth);
            Date dt = calendar.getTime();

            System.out.println("Data = " + date);
        }
    });
}
```

Prof. Razer A N R Montaño

2017

64



## Exercícios

1. No mesmo formulário do exercício anterior (Nome, E-mail, Endereço, Sexo, Fumante), adicionar a data de nascimento, sendo obtida por um DatePicker. Deve-se ter dois botões, um para salvar e outro para limpar o formulário.
  - O botão **limpar** apaga o conteúdo dos campos (seta com valores "")
  - O botão **salvar** mostra em um TextView abaixo o nome, e-mail, endereço, sexo e se é ou não fumante e sua data de nascimento.

66

## Material Design

# Material Design

- Link: <https://developer.android.com/design/material/index.html?hl=pt-br>
- Guia de design visual para aplicações
- Novos componentes e funcionalidades a partir do Android 5.0 (API 21)
- Deve-se indicar o tema na aplicação
- Desenhar a aplicação seguindo as orientações
  - <https://material.io/guidelines/material-design/introduction.html?hl=pt-br#>
- Possui 3 variações
  - Escura (`Theme.Material`)
  - Clara (`Theme.Material.Light`)
  - Clara com ActionBar escura (`Theme.Material.DarkActionBar`)

Prof. Razer A N R Montaño

2017

67

# Material Design

- Para adicionar na aplicação
  - Dentro de `styles.xml` adicionar:

```
<resources>
    <style name="AppTheme" parent="android:Theme.Material.Light">
        <!-- theme customizations -->
    </style>
</resources>
```

Prof. Razer A N R Montaño

2017

68

# Material Design

- Pode-se personalizar a paleta de cores
- Dentro de `colors.xml`, tem-se as cores

```
<resources>
    <color name="colorPrimary">#ff0084</color>
    <color name="colorPrimaryDark">#ACACAC</color>
    <color name="colorAccent">#FF4000</color>
</resources>
```

- Aplica-se ao tema:

```
<resources>
    <!-- inherit from the material theme -->
    <style name="AppTheme" parent="android:Theme.Material">
        <!-- Main theme colors -->
        <!-- your app branding color for the app bar -->
        <item name="android:colorPrimary">@color/colorPrimary</item>
        <!-- darker variant for the status bar and contextual app bars -->
        <item name="android:colorPrimaryDark">@color/colorPrimaryDark</item>
        <!-- theme UI controls like checkboxes and text fields -->
        <item name="android:colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

Prof. Razer A N R Montaño

2017

69