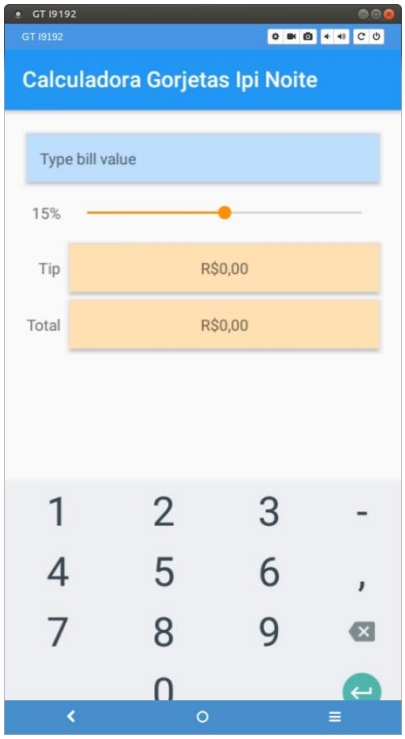
1. Introdução

Este material baseia-se no Capítulo 3 do livro [1]. Iremos construir uma aplicação que obtém do usuário um valor de consumo e calcula dois valores: o valor de gorjeta com base em um percentual que varia de 0% a 30% e o valor total da conta. Veja a Figura 1.

Figura 1



Passo 1. Vamos começar criando a aplicação com os seguintes valores.

Application Name: Calculadora de Gorjetas

Company Domain: br.com.bossini.

Mantenha os nomes da Activity e do arquivo de layout.

Template: Empty Activity

Minimum SDK: API 21: Android Lollipop.

Passo 2. Apague o TextView que contém o texto Hello World.

Passo 3. O gerenciador de layouts dessa aplicação será um GridLayout. Abra o arquivo activity_main.xml em modo texto e troque o RelativeLayout padrão (ou qualquer outro que o Android Studio tenha usado) para GridLayout.

Passo 4. Ele terá 2 colunas e 4 linhas. O número de colunas deve ser configurado utilizando a propriedade columnCount. Não é necessário configurar rowCount; ele será configurado automaticamente conforme adicionamos componentes visuais ao gerenciador.

Passo 5. Por padrão, não há margens separando os componentes visuais em um gerenciador de layout. O conjunto de recomendações conhecido como Material Design (muito importante, pesquise sobre isso!) sugere pelo menos 8dp entre os componentes. Para tal, basta marcar a checkbox useDefaultMargins.

Passo 6. Agora vamos construir a interface gráfica (GUI) arrastando e soltando componentes à árvore de componentes. A primeira linha contém um TextView e um EditText que na verdade vão se sobrepor. Logo a razão disso ficará mais clara. Arraste e solte um Number EditText e altere seu id para amountEditText, seu layout:column para 0 e seu layout:columnSpan para 2.

Passo 7. Arraste e solte um Medium Text TextView da paleta sobre o EditText na árvore de componentes. Troque seu id para amountTextView, seu layout:row para 0, seu layout:column para 0 e seu layout:columnSpan para 2. Nota: Caso não encontre um Medium Text na paleta, basta adicionar um comum. Caso julgue necessário, adicione o atributo da Listagem 1 à tag que o representa.

Listagem 1

```
android:textAppearance="@style/TextAppearance.AppCompat.Medium"/>
```

Passo 8. A segunda linha conterá um TextView que serve de rótulo para uma SeekBar, às vezes chamada de slider. Trata-se de um componente visual que o usuário poder utilizar para escolher um número inteiro variando em 0 e 30, que representa o percentual de gorjeta. Arraste e solte um Medium TextView sobre o amountTextView na árvore de componentes. Seu id deve ser percentTextView.

Passo 9. Arraste e solte uma SeekBar sobre o percentTextView na árvore de componentes cujo id deve ser percentSeekBar.

Passo 10. A terceira linha conterá dois TextView. O primeiro serve de rótulo para o segundo, indicando o texto “gorjeta” enquanto o outro exibe o valor de gorjeta calculado. Seus ids devem ser tipLabelTextView e tipTextView, respectivamente.

Passo 11. A quarta linha conterá também dois TextView. O primeiro serve de rótulo para o segundo, indicando o texto “total” enquanto o outro exibe o valor total calculado, ou seja, o total

de consumo mais o valor da gorjeta. Seus ids são totalLabelTextView e totalTextView, respectivamente.

Passo 12. Agora criaremos os pares chave/valor necessários para cada texto que a aplicação exibe. Lembre-se que eles devem todos ficar nos arquivos de recurso strings.xml.Para começar, vamos criar o texto que serve de dica para o usuário entender onde ele deve digitar o valor total de consumo. Na árvore de componentes, clique no amountTextView e selecione sua propriedade hint. O resource name deve ser "enter_amount" e o resource value deve ser "Digite o valor consumido".

Passo 13. Repita os passos para adicionar os pares chave/valor para a propriedade text de cada view da Tabela 1.

Tabela 1

View	Resource name	Resource value
percentTextView	tip_percentage	15%
tipLabelTextView	tip	Tip
totalLabelTextView	total	Total

Passo 14. Agora vamos alinhar à direita os elementos da coluna da esquerda. Segure CTRL (ou CMD no Mac) e clique nos componentes percentTextView, tipLabelTextView e totalLabelTextView. Expanda a propriedade layout:gravity e marque a checkbox right.

Passo 15. O campo em que o usuário digita o valor deve aceitar somente números e vamos restringi-los a no máximo 6 dígitos. Para isso, selecione o componente amountEditText e configure sua propriedade digits com o valor 0123456789 e maxLength com 6.

Passo 16. As propriedades do componente amountTextView serão as seguintes.

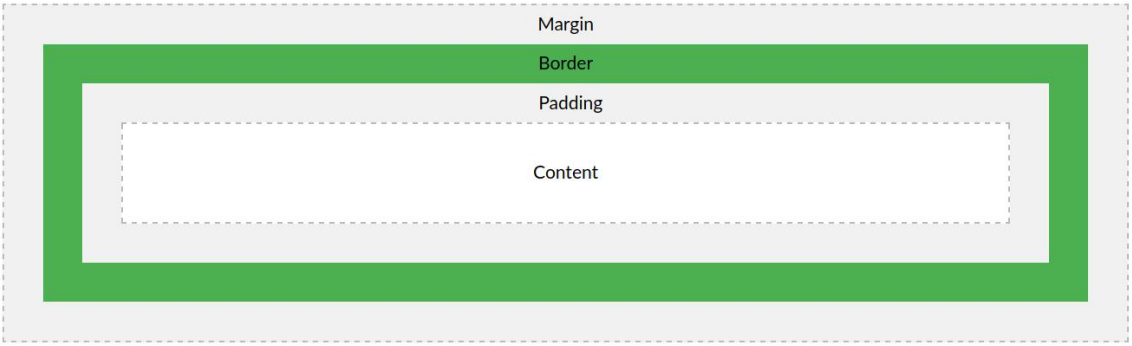
16.1. text deve ficar vazia, ou seja, apague o texto Medium Text.

16.2. expanda layout:gravity e configure fill para horizontal, indicando que esse campo deve ocupar todo o espaço horizontal restante na linha em que está.

16.3. background deve ser configurada para um par chave/valor com chave igual a "amount_background" e valor igual a #BBDEFB, que é uma cor azul escolhida do Material Design (pesquise sobre isso!!!!).

16.4. Adicione padding em volta do TextView, que especifica espaço extra em torno do seu conteúdo. Expanda padding e escolha all. Crie um novo par chave/valor com chave igual a textview_padding e valor 12dp. Esse é um recurso que será usado novamente em breve, para outros TextView. **Obs**: Veja a Figura 2 para relembrar do que se trata o padding, margin etc.

Figura 2



16.5. Agora vamos adicionar sombra ao componente. Sua propriedade elevation deve ser configurada para um par chave/valor com chave igual a elevation e valor igual a 4dp.

Passo 17. O componente percentTextView está alinhado um pouco mais alto do que a seekBar. Vamos centralizá-lo verticalmente. Para isso, expanda layout:gravity e configure o valor center para vertical.

Passo 18. A percentSeekBar será configurada com as seguintes propriedades.

18.1. Por padrão, seu intervalo é de 0 a 100 e seu progresso atual é indicado pelo propriedade progress. Queremos que o intervalo seja de 0 a 30 e que o valor padrão seja 15. Assim, configure max para 30 e progress para 15.

18.2. Expanda layout:gravity e configure fill como horizontal para que ela ocupe toda sua célula horizontalmente.

18.3. Configure layout:height com um novo par chave/valor cuja chave é seekbar_height e valor 40dp.

Passo 19. As configurações dos componentes tipTextView e totalTextView são as seguintes.

19.1. Apague Medium Text da propriedade text de ambas.

19.2. Expanda layout:gravity e configure fill para horizontal.

19.3. Configure background como um novo recurso de chave result_background e valor #FFE0B2, que representa um laranja claro do Material Design (nem precisa falar, né?).

19.4. Configure gravity como center, o que afeta o texto que elas exibem (e não sua posição no layout).

19.5. Expanda padding, clique nos três pontos de all e selecione o recurso textview_padding que você criou anteriormente.

19.6. Adicione sombra usando o elevation que você também já havia criado.

Passo 20. Toda aplicação tem um tema que define seu look-and-feel, especificado no arquivo AndroidManifest.xml. Iremos fazer algumas alterações no tema de nossa aplicação. Abra o arquivo styles.xml e clique em Open editor no canto superior direito e configure os seguintes itens.

20.1. Clique na cor de amostra de colorPrimary. Escolha a amostra Material Blue 500 (passe o mouse por cima das amostras para ver seu nome).

20.2. Siga os mesmos passos para colorPrimaryDark e escolha a única amostra disponível, Material Blue 700.

20.3. Finalmente, troque colorAccent para Orange accent 400.

Passamos agora à lógica da aplicação.

Passo 21. A Listagem 2 mostra as variáveis de instância de nossa Activity.

Listagem 2

```
private static final NumberFormat currencyFormat =
    NumberFormat.getCurrencyInstance();
private static final NumberFormat percentFormat =
    NumberFormat.getPercentInstance();
private double billAmount = 0.0;
private double percent = 0.15;
private TextView amountTextView;
private TextView percentTextView;
private TextView tipTextView;
private TextView totalTextView;
```

Passo 22. A Listagem 3 mostra a sobrescrita do método onCreate, herdado da classe Activity.

Listagem 3

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  amountTextView = (TextView) findViewById(R.id.amountTextView);
  percentTextView = (TextView) findViewById(R.id.percentTextView);
  tipTextView = (TextView) findViewById(R.id.tipTextView);
  totalTextView = (TextView) findViewById(R.id.totalTextView);
  tipTextView.setText(currencyFormat.format(0));
  totalTextView.setText(currencyFormat.format(0));
  EditText amountEditText =
      (EditText) findViewById(R.id.amountEditText);
  amountEditText.addTextChangedListener(amountEditTextWatcher);
  SeekBar percentSeekBar =
      (SeekBar) findViewById(R.id.percentSeekBar);
  percentSeekBar.setOnSeekBarChangeListener(seekBarListener);
}
```

Passo 23. A aplicação possui uma regra de negócio muito simples, implementada por um único método criado na própria classe Activity, exibido pela Listagem 4.

Listagem 4

```
private void calculate (){
  percentTextView.setText(percentFormat.format(percent));
  double tip = billAmount * percent;
  double total = billAmount + tip;
  tipTextView.setText (currencyFormat.format(tip));
  totalTextView.setText(currencyFormat.format(total));
}
```

Passo 24. A Listagem 5 mostra a implementação do objeto responsável por responder aos eventos envolvendo a percentSeekBar.

Listagem 5

```
private final SeekBar.OnSeekBarChangeListener seekBarListener =          new
SeekBar.OnSeekBarChangeListener() {

      @Override
      public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        percent = progress / 100.0;
        calculate();
      }
      @Override
      public void onStartTrackingTouch(SeekBar seekBar) {
      }
      @Override
      public void onStopTrackingTouch(SeekBar seekBar) {
      }
    };
```

Passo 25. Finalmente, a Listagem 6 mostra a implementação do objeto responsável por lidar com os eventos ocorridos no EditText.

Listagem 6

```
private final TextWatcher amountEditTextWatcher = new TextWatcher(){
  @Override
  public void beforeTextChanged(CharSequence s, int start, int count, int after) {
  }
  @Override
  public void onTextChanged(CharSequence s, int start, int before, int count) {
    try{
      billAmount = Double.parseDouble(s.toString()) / 100.0;
      amountTextView.setText((currencyFormat.format(billAmount)));
    }
    catch (NumberFormatException e){
      amountTextView.setText("");
      billAmount = 0.0;
    }
    calculate();
  }
  @Override
  public void afterTextChanged(Editable s) {
  }
};
```

Passo 26. No arquivo AndroidManifest.xml, adicione as configurações da Listagem 6, que devem ser propriedades da MainActivity.

Listagem 6

```
android:screenOrientation="portrait"
android:windowSoftInputMode="stateAlwaysVisible">
```

Bibliografia

[1] Deitel, P., Deitel, H., Wald, A. Android 6 for Programmers – An App-Driven Approach. 3rd edition. Pearson, 2016.