Ficha do Aplicativo

App 01 – Aplicativo Boas Vindas
Resumo Neste aplicativo o usuário clicará em um botão e uma mensagem será apresentada na tela.
Objetivos de Aprendizagem 1. Primeiro aplicativo a ser desenvolvido. Execução dos passos iniciais para criação do aplicativo buscando compreender os parâmetros envolvidos em sua criação. 2. Compreender o que são e como se relacionam: • Layout, que é o desenho da tela do aplicativo • Código Java, que garante a funcionalidade do aplicativo 3. Conhecer os elementos iniciais de configuração de Layout. 4. Usar os primeiros componentes do tipo View, no caso: TextView e Button. 5. Implementar e testar o código necessário ao funcionamento do clique no botão. 6. Apresentação da estrutura de um projeto Android, evidenciando seus principais elementos: • Layout (desenho da tela, ou interface visual) • Activity (código java) • Resources • Classe R 7. Conhecer e usar a classe Toast para exibição de mensagens.
Dinâmica do Aplicativo Neste aplicativo haverá dois botões (button) e duas caixas de visualização de texto (TextView). Veja o layout da tela na próxima página. Uma caixa de mensagem – componente TextView – estará centralizada na tela terá seu texto inicial ("...") alterado para uma mensagem de boas-vindas aleatória (dentre 5 possibilidades) quanto for clicado o botão posicionado na parte superior do layout. O botão posicionado na base da tela limpará a caixa de mensagem, retornando seu conteúdo para "..." e exibirá um Toast, mensagem temporária. A TextView posicionada na parte superior é apenas um rótulo que contém o texto “Sejam todos bem vindos” e é inerte do ponto de vista da interação com o usuário.

Lista de Activities do Aplicativo

Nome	Layout
MainActivity.java	activity_main.xml

Resources

strings.xml

```
<resources>
    <string name="app_name">Aplicativo de Boas Vindas</string>
    <string name="texto_rotulo">Olá pessoal</string>
    <string name="txt_btnmsg">Clique para obter mensagem</string>
    <string name="txt_btnlimpa">Limpar Mensagem</string>
</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="corFundo">#F2B9FB</color>
    <color name="corFundoMsg">#FADBFF</color>
    <color name="corTextoMsg">#0C0C9A</color>
</resources>
```

Detalhamento das Activities

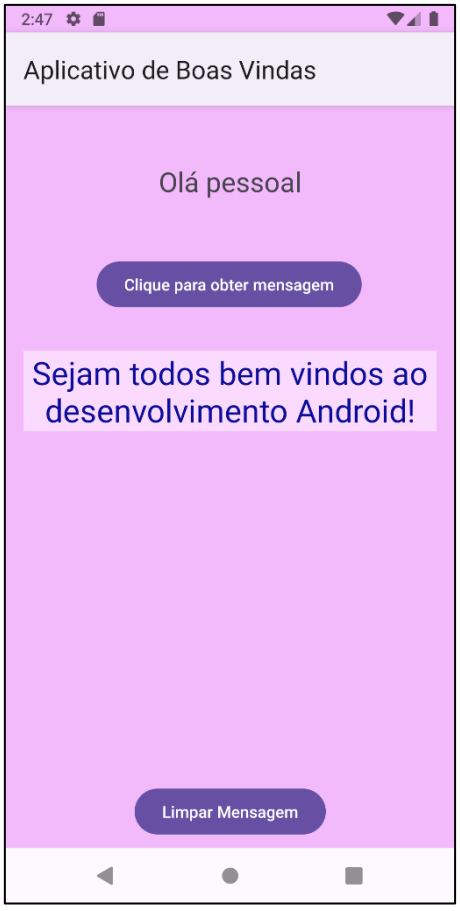
Layout: activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/corFundo"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/texto_rotulo"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@+id/btnExibeMsg"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnExibeMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="btnExibeMsgOnClick"
        android:text="@string/txt_btnmsg"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.496"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.22" />

    <TextView
        android:id="@+id/txtMensagem"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:background="@color/corFundoMsg"
        android:gravity="center"
        android:text="..."
        android:textColor="@color/corTextoMsg"
        android:textSize="28sp"
        app:layout_constraintBottom_toTopOf="@+id/btnLimpaMsg"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnExibeMsg"
        app:layout_constraintVertical_bias="0.10" />
```



```
    <Button
        android:id="@+id/btnLimpaMsg"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:onClick="btnLimpaMsgOnClick"
        android:text="@string/txt_btnlimpa"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Código: MainActivity.java

```
/* Observação:
   O nome do package está omitido, porque ao fazer o seu projeto esse nome será outro.
   Os imports também estão omitidos. O motivo para isso é que a cada nova versão do Android Studio pode
   haver mudanças nos nomes dos caminhos da biblioteca. */


public class MainActivity extends AppCompatActivity {

    private TextView txtMensagem;
    private Button btnExibeMsg;
    private Button btnLimpaMsg;
    private Random gerador;
    private int nMsg = -1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        txtMensagem = findViewById(R.id.txtMensagem);
        btnExibeMsg = findViewById(R.id.btnExibeMsg);
        btnLimpaMsg = findViewById(R.id.btnLimpaMsg);
        gerador = new Random();

        setBtnsOnClick();
    }

    private void setBtnsOnClick() {
        btnExibeMsg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                trataBtnExibeMsgOnClick();
            }
        });

        btnLimpaMsg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                trataBtnLimpaMsgOnClick();
            }
        });
    }

    private void trataBtnExibeMsgOnClick() {
        int sorteio = gerador.nextInt(6);
        while (sorteio == nMsg)
            sorteio = gerador.nextInt(6);
        nMsg = sorteio;

        if (nMsg == 0)
            txtMensagem.setText("Sejam todos bem vindos ao desenvolvimento Android!");
        else if (nMsg == 1)
            txtMensagem.setText("Desenvolver para Android é legal.\nVamos nos divertir!");
        else if (nMsg == 2)
            txtMensagem.setText("Vamos começar. Temos muito a aprender.");
        else if (nMsg == 3)
            txtMensagem.setText("Que as nossas aulas sejam boas e divertidas.");
        else if (nMsg == 4)
            txtMensagem.setText("Se chover, não esqueça o guarda-chuva.");
```

```
        else if (nMsg == 5)
            txtMensagem.setText("Se esfriar, não esqueça o agasalho.");
    }

    private void trataBtnLimpaMsgOnClick() {
        txtMensagem.setText("...");
        Toast.makeText(this, "Campo mensagem foi limpo", Toast.LENGTH_SHORT).show();
    }
}
```

Notas Técnicas

Iniciando o aplicativo

Na tela de entrada do Android Studio selecione a opção “Start a new Android Studio Project” e siga as telas do passo a passo iniciando o aplicativo como “Empty Activity”.

No campo Name da segunda tela coloque o nome “Boas Vindas”.

No campo Package Name vou usar sempre um nome de pacote iniciando com “com.ilp506.___”, onde no lugar de ‘___‘ haverá o nome do aplicativo. Sugiro que padronizem os nomes de pacotes dos seus aplicativos como algo do tipo “com.seunome.nomedoaplicativo”.

No campo minimum API level você deve escolher qual será a menor versão do sistema operacional Android capaz de rodar o seu aplicativo. Nas nossas aulas vamos usar API 22 (Android 5.1 Lollipop), por dois motivos principais:

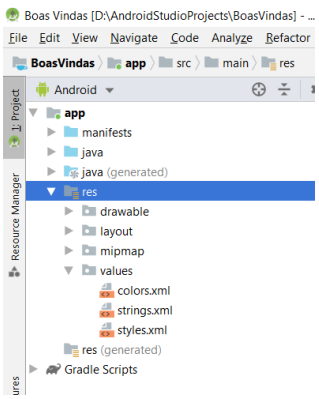
- Ao adotá-la seu aplicativo vai abranger significativa porcentagem (cerca de 80%, em julho/2019) de dispositivos móveis usados em todo o mundo;
- É a partir dessa versão que está disponível o Material Design, importante padrão visual desenvolvido pelo Google e amplamente usado nos dias atuais;

Para saber mais sobre a API 22 – Android 5.1 Lollipop acesse:
https://developer.android.com/about/versions/lollipop?hl=pt-br

Para saber mais sobre o Material Design acesse: https://material.io/design/?hl=pt-br

Resources

Resources (recursos) são importantes elementos presentes em um projeto de aplicativo Android. Os principais tipos de resources são layouts, strings, cores, imagens, menus, embora existam mais alguns tipos. Em geral ficam armazenados em arquivos são armazenados em arquivos e podem ser inseridos/editados pelo programador.



Para ter acesso a um resource, no painel esquerdo da tela do Android Studio navegue pelo caminho **app -> res -> ...** conforme mostrado na imagem ao lado.

Abra os arquivos colors.xml e strings.xml analise o conteúdo dos mesmos. O primeiro contém nomes associados a cores fornecidas em valores Hexadecimal. O segundo contém textos também associados a nomes.

Ao usar resources desse tipo você torna seu aplicativo passível de ser customizado no futuro, sem a necessidade de recompilação do mesmo. Isso é útil quando se deseja criar skins e suporte a outros idiomas.

Agora compare os resources desse seu projeto recém iniciado com o conteúdo apresentado na página 2 deste documento e faça as alterações nos arquivos do seu projeto para que contenham os mesmos elementos/valores. Isso será muito usado e com o tempo vocês vão ganhar prática com essas edições, bem como descobrir outros elementos que podem ser inseridos.

Importância do par Layout + Código

Em um projeto Android Studio o par formado por Layout + Código é uma Activity. Em palavras simples, uma Activity é uma tela do aplicativo que é constituída por dois elementos:

Layout: é o desenho da tela e é armazenado em um arquivo XML. Na verdade, um layout é resource. É no layout que definimos a aparência de uma tela do aplicativo.

Código: o código é escrito em uma linguagem de programação, notadamente Java ou Kotlin, e define o comportamento do aplicativo. É através do uso da linguagem de programação que o desenvolvedor vai implementar as regras de negócio especificadas para o aplicativo. Neste curso usaremos a linguagem Java, por isso nossos códigos serão salvos em arquivos com extensão ".java".

É possível termos nos nossos projetos Android um arquivo Java não associado a uma tela. Isso ocorre quando modularizamos e software e criamos uma classe contendo funcionalidades que serão usadas por outras classes. Em breve haverá uma aplicação chamada "Cursos das Fatecs" onde isso será visto.

Também é possível que tenhamos um arquivo XML de layout não vinculado a uma tela, mas que será usado como parte da aplicação. Isso também será visto adiante.

Detalhes do Layout

A aparência de uma tela é definida em função dos componentes adicionados ao mesmo, bem como em função dos valores colocados para as diversas propriedades desses componentes. Analise atentamente o XML que começa na página 2. Todas as propriedades de componentes que foram alteradas constam desse XML.

Ao definir um layout atenção especial deve ser dada à propriedade "id" dos componentes com os quais haverá alguma iteração no código Java. Neste aplicativo três componentes tiveram seus ids ajustados: txtMensagem, btnExibeMsg, btnLimpaMsg. Procure esses identificadores na página 2. A propriedade id está na primeira do bloco de propriedades de cada componente no XML.

Faça o seu projeto conter os elementos mostrados e configure cada propriedade conforme o XML apresentado.

Detalhes do Código

Agora que seu aplicativo já tem um layout organizado, vamos dar funcionalidade ao mesmo. Está na hora de trabalhar com o código Java. O código completo deste aplicativo está em MainActivity.java apresentado na página 3. Analise esse código e observe que está definida a classe MainActivity que estende a classe AppCompatActiviy.

Ao criar um novo projeto no Android Studio haverá um método denominado onCreate nessa Classe. O conteúdo padrão desse método será objeto de estudo em uma lição futura.

Vamos dar vida ao botão btnExibeMsg. Para isso crie no seu aplicativo o código destacado abaixo:

```
public void btnExibeMsgOnClick(View view) {
   TextView txtMensagem = findViewById(R.id.txtMensagem);
   txtMensagem.setText("Sejam todos bem vindos ao nosso curso de\nAndroid Studio!");
}
```

Vamos começar por essa versão do código com uma única possibilidade de mensagem. Depois você implementa a versão que está na página 3, que contém cinco possíveis mensagens, das quais uma será exibida a depender de um número aleatório gerado.

Na linha `TextView txtMensagem = findViewById(R.id.txtMensagem);` estão acontecendo duas coisas: a) é declarado o objeto `txtMensagem` da classe `TextView` e b) o objeto é instanciado usando-se a função `findViewById`

A função `findViewById` é muito importante nos projetos Android pois ela permite que o código Java possa fazer referência aos componentes visuais do layout XML. Para estabelecer o vínculo do código com o layout é preciso fazer referência ao id do componente no layout. Isso é feito através do arquivo R. A referência ao identificador do componente é feito através do identificador `R.id.txtMensagem`.

O arquivo R reúne todos os elementos existentes no projeto e permite que o código Java tenha acesso aos mesmos. Este arquivo nunca deve ser editado pelo desenvolvedor. Sua criação e manutenção fica exclusivamente a cargo do Android Studio.

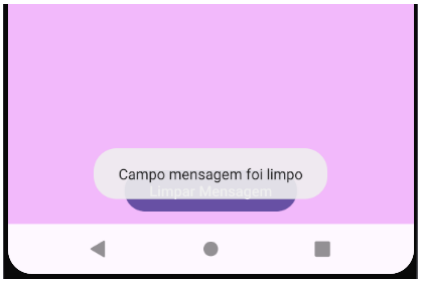
A segunda linha do método `btnExibeMsgOnClick` é uma chamada ao método `setText` do objeto `txtMensagem` passando o texto que se quer que apareça na tela do aplicativo.

O segundo método dessa classe `public void btnLimpaMsgOnClick(View view)` foi criado para voltar o conteúdo desse TextView aos pontos "..." originais.

Após a criação desses dois métodos no código Java deve-se voltar ao layout XML e configurar a propriedade onClick de cada botão (componente Button) para acionar o respectivo método.

Agora faça funcionar. Implemente esses dois métodos, configure a propriedade onClick de cada botão e execute o aplicativo. Se tudo foi feito correto, ao clicar nos botões você verá a mudança de mensagem.

Notificação Toast



Uma notificação Toast constitui um modo simples de exibir uma mensagem para o usuário, abrindo uma pequena janela pop-up. Esse pop-up só ocupa a quantidade de espaço necessária para a mensagem, e a atividade atual continua visível e interativa. Notificações Toast desaparecem automaticamente após um tempo limite. Existem dois tempos um mais curto que pode ser especificado pela constante LENGTH_SHORT e outro mais longo especificado através da constante LENGTH_LONG, ambas definidas dentro da classe Toast; A figura ao lado mostra como um Toast é exibido. Para exibi-la fazemos:

```
Toast.makeText(this, "Campo mensagem foi limpo", Toast.LENGTH_SHORT).show()
```

Mais sobre Toast neste link: https://developer.android.com/guide/topics/ui/notifiers/toasts?hl=pt-br