Ficha do Aplicativo

App03 – Aplicativo Trabalhando com Imagens
Resumo O nome já diz, neste App04 vamos trabalhar com imagens, ou seja, usaremos o componente ImageView e veremos como exibir, trocar e ajustar a imagem dentro da área de visualização. Também aproveitaremos para ver o uso de três novos componentes: ImageButton, Spinner e Switch.
Objetivos de Aprendizagem 1. Entender o conceito de drawable – que é bem mais amplo do que se pode imaginar a princípio 2. Configurar e usar ImageButton 3. Configurar e usar ImageView 4. Configurar e usar Switch 5. Configurar e usar Spinner 6. Trabalhar a criação de layout que não aceita mudança de orientação
Dinâmica do Aplicativo Este aplicativo conterá um elemento central ImageView que exibirá uma imagem. Essa imagem poderá ser trocada através do acionamento dos botões e do switch que estarão posicionados no topo da tela. Através do uso das opções que ficarão disponíveis no Spinner será possível testar as várias opções de escala e posicionamento da imagem, dentro da área útil do componente ImageView.

Lista de Activities do Aplicativo

Nome	Layout
MainActivity.java	layout\activity_main.xml

Resources

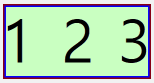
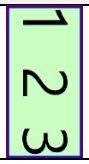
strings.xml

```
<resources>
    <string name="app_name">Trabalhando com Imagens</string>
    <string name="imgdescr_peq">Imagem pequena</string>
    <string name="imgdescr_med">Imagem média</string>
    <string name="imgdescr_gde">Imagem grande</string>
    <string name="imgdescr_principal">Imagem Exemplo</string>
    <string name="lblconjimg">Usar o conjunto vertical de imagens</string>
    <string name="lblswitch">Exibir espaço ocupado pelo ImageView</string>
    <string name="lblopcoes">Opções de Visualização</string>
    <string name="btntroca">Troca Opção de Visualização</string>
    <string-array name="opcoesImagem">
        <item>centerInside</item>
        <item>center</item>
        <item>centerCrop</item>
        <item>fitStart</item>
        <item>fitCenter</item>
        <item>fitEnd</item>
        <item>fitXY</item>
        <item>matrix</item>
    </string-array>
</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="corFundoLayout">#E3F4F3</color>
    <color name="corFundoFerramentas">#73D2F8</color>
</resources>
```

Drawables deste Aplicativo

Nome do arquivo	Dimensões	Tipo	Imagem
btn_tamanhoimagem_12.xml	12 x 12	XML – Vector	
btn_tamanhoimagem_24.xml	24 x 24	XML – Vector	
btn_tamanhoimagem_36.xml	36 x 36	XML – Vector	
android120x120.png	120 x 120	Imagem PNG	
android600x600.png	600 x 600	Imagem PNG	
img300x150.png	300 x 150	Imagem PNG	
img600x300.png	600 x 300	Imagem PNG	
img1200x600.png	1200 x 600	Imagem PNG	
imgv150x300.png	150 x 300	Imagem PNG	
imgv300x600.png	300 x 600	Imagem PNG	
imgv600x1200.png	600 x 1200	Imagem PNG	

Para saber mais sobre Gráficos Vetoriais acesse o link: https://www.devmedia.com.br/entendendo-e-usando-o-svg/19773

Detalhamento da Activity

Layout na posição retrato: 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/corFundoLayout"
    tools:context=".MainActivity">

    <View
        android:id="@+id/dividerBtns"
        android:layout_width="0dp"
        android:layout_height="58dp"
        android:layout_marginStart="4dp"
        android:layout_marginTop="4dp"
        android:layout_marginEnd="4dp"
        android:background="@color/corFundoFerramentas"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ImageButton
        android:id="@+id/btnTamPeq"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:backgroundTint="#E7E8EA"
        android:contentDescription="@string/imgdescr_peq"
        android:tint="#0915E8"
        app:layout_constraintBottom_toBottomOf="@+id/dividerBtns"
        app:layout_constraintEnd_toStartOf="@+id/btnTamMed"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/dividerBtns"
        app:srcCompat="@drawable/btn_tamanhoimagem_12" />

    <ImageButton
        android:id="@+id/btnTamMed"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:backgroundTint="#E7E8EA"
        android:contentDescription="@string/imgdescr_med"
        android:tint="#0915E8"
        app:layout_constraintBottom_toBottomOf="@+id/dividerBtns"
        app:layout_constraintEnd_toStartOf="@+id/btnTamGde"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnTamPeq"
        app:layout_constraintTop_toTopOf="@+id/dividerBtns"
        app:srcCompat="@drawable/btn_tamanhoimagem_24" />

    <ImageButton
        android:id="@+id/btnTamGde"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:backgroundTint="#E7E8EA"
        android:contentDescription="@string/imgdescr_gde"
        android:tint="#0915E8"
        app:layout_constraintBottom_toBottomOf="@+id/dividerBtns"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnTamMed"
        app:layout_constraintTop_toTopOf="@+id/dividerBtns"
        app:srcCompat="@drawable/btn_tamanhoimagem_36" />

    <Switch
        android:id="@+id/swtImgVerticais"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:checked="false"
        android:showText="false"
        android:switchMinWidth="80dp"
        android:text="@string/lblconjimg"
        app:layout_constraintEnd_toEndOf="@+id/dividerImg"
        app:layout_constraintStart_toStartOf="@+id/dividerImg"
        app:layout_constraintTop_toBottomOf="@+id/dividerBtns"
        tools:ignore="UseSwitchCompatOrMaterialXml" />
```

```
    <View
        android:id="@+id/dividerImg"
        android:layout_width="350dp"
        android:layout_height="350dp"
        android:layout_marginTop="64dp"
        android:background="?android:attr/listDivider"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/dividerBtns" />

    <ImageView
        android:id="@+id/imgExemplo"
        android:layout_width="350dp"
        android:layout_height="350dp"
        android:contentDescription="@string/imgdescr_principal"
        android:scaleType="centerInside"
        app:layout_constraintBottom_toBottomOf="@+id/dividerImg"
        app:layout_constraintEnd_toEndOf="@+id/dividerImg"
        app:layout_constraintStart_toStartOf="@+id/dividerImg"
        app:layout_constraintTop_toTopOf="@+id/dividerImg"
        app:srcCompat="@drawable/android600x600" />

    <Switch
        android:id="@+id/swtEspOcupado"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:checked="true"
        android:showText="false"
        android:switchMinWidth="80dp"
        android:text="@string/lblswitch"
        app:layout constraintEnd toEndOf="@+id/dividerImg"
        app:layout_constraintStart_toStartOf="@+id/dividerImg"
        app:layout_constraintTop_toBottomOf="@+id/dividerImg"
        tools:ignore="UseSwitchCompatOrMaterialXml" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="1dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="291dp"
        android:text="@string/lblopcoes"
        app:layout_constraintEnd_toEndOf="@+id/dividerImg"
        app:layout_constraintHorizontal_bias="0.01"
        app:layout_constraintStart_toStartOf="@+id/dividerImg"
        app:layout_constraintTop_toBottomOf="@+id/swtEspOcupado" />

    <Spinner
        android:id="@+id/spnOpcoes"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:background="#FFFFFF"
        android:entries="@array/opcoesImagem"
        android:paddingTop="16dp"
        android:paddingBottom="16dp"
        app:layout_constraintEnd_toEndOf="@+id/dividerImg"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/dividerImg"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

</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 ImageView imgExemplo;
    private Switch swtImgVerticais;
    private Switch swtEspOcupado;
    private Spinner spnOpcoes;
```

```
    @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;
        });

        imgExemplo = findViewById(R.id.imgExemplo);
        configBtnsTam();
        configSwtImgVerticais();
        configSwtEspOcupado();
        configSpnOpcoes();
    }


    private void configBtnsTam() {
        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                trataBtnTamOnClick(view);
            }
        };

        ImageButton btnTam;
        btnTam = findViewById(R.id.btnTamPeq);
        btnTam.setOnClickListener(listener);
        btnTam = findViewById(R.id.btnTamMed);
        btnTam.setOnClickListener(listener);
        btnTam = findViewById(R.id.btnTamGde);
        btnTam.setOnClickListener(listener);
    }

    private void trataBtnTamOnClick(View view) {
        ImageView imgExemplo = findViewById(R.id.imgExemplo);
        Switch swtImgVerticais = findViewById(R.id.swtImgVerticais);
        if (view.getId() == R.id.btnTamPeq)
            if (swtImgVerticais.isChecked())
                imgExemplo.setImageResource(R.drawable.imgv150x300);
            else
                imgExemplo.setImageResource(R.drawable.img300x150);
        if (view.getId() == R.id.btnTamMed)
            if (swtImgVerticais.isChecked())
                imgExemplo.setImageResource(R.drawable.imgv300x600);
            else
                imgExemplo.setImageResource(R.drawable.img600x300);
        if (view.getId() == R.id.btnTamGde)
            if (swtImgVerticais.isChecked())
                imgExemplo.setImageResource(R.drawable.imgv600x1200);
            else
                imgExemplo.setImageResource(R.drawable.img1200x600);
    }

    private void configSwtImgVerticais() {
        swtImgVerticais = findViewById(R.id.swtImgVerticais);
        swtImgVerticais.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                trataSwtImgVerticais();
            }
        });
    }

    public void trataSwtImgVerticais() {
        ImageView imgExemplo = findViewById(R.id.imgExemplo);
        Random gerador = new Random();
        int opc = gerador.nextInt(2);
        if (opc == 0)
            imgExemplo.setImageResource(R.drawable.android120x120);
        else
            imgExemplo.setImageResource(R.drawable.android600x600);
    }

    private void configSwtEspOcupado() {
        swtEspOcupado = findViewById(R.id.swtEspOcupado);
        swtEspOcupado.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                trataSwtImgEspOcupado();
            }
        });
    }

    private void trataSwtImgEspOcupado() {
        View dividerImg = findViewById(R.id.dividerImg);
        Switch swtEspOcupado = findViewById(R.id.swtEspOcupado);
        if (swtEspOcupado.isChecked())
            dividerImg.setVisibility(View.VISIBLE);
        else
            dividerImg.setVisibility(View.INVISIBLE);
    }

    private void configSpnOpcoes() {
        spnOpcoes = findViewById(R.id.spnOpcoes);
        spnOpcoes.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                trataSpnOpcoesOnItemClick(position);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {
            }
        });
    }

    private void trataSpnOpcoesOnItemClick(int position) {
        if (spnOpcoes.getSelectedItemPosition() == 0)
            imgExemplo.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
        else if (spnOpcoes.getSelectedItemPosition() == 1)
            imgExemplo.setScaleType(ImageView.ScaleType.CENTER);
        else if (spnOpcoes.getSelectedItemPosition() == 2)
            imgExemplo.setScaleType(ImageView.ScaleType.CENTER_CROP);
        else if (spnOpcoes.getSelectedItemPosition() == 3)
            imgExemplo.setScaleType(ImageView.ScaleType.FIT_START);
        else if (spnOpcoes.getSelectedItemPosition() == 4)
            imgExemplo.setScaleType(ImageView.ScaleType.FIT CENTER);
        else if (spnOpcoes.getSelectedItemPosition() == 5)
            imgExemplo.setScaleType(ImageView.ScaleType.FIT_END);
        else if (spnOpcoes.getSelectedItemPosition() == 6)
            imgExemplo.setScaleType(ImageView.ScaleType.FIT_XY);
        else if (spnOpcoes.getSelectedItemPosition() == 7) {
            // é necessário configurar um objeto da classe Matrix que fará a transformação da imagem.
            //Matrix m;
            //m = ...
            //imgExemplo.setImageMatrix(m);
            //imgExemplo.setScaleType(ImageView.ScaleType.MATRIX);
        }
    }
}
```

Notas Técnicas

Drawables

No ambiente Android a palavra "drawable" se refere a todo e qualquer elemento que possa ser desenhado em uma interface gráfica. Na página do Android Developers (link a seguir) encontramos a seguinte definição: "Drawable é uma abstração genérica para a expressão 'algo que pode ser desenhado' ".

Ao ler sobre drawables e provavelmente pensar em um termo em português como *"desenháveis"* você pode estar pensando em imagens PNG ou JPEG e não estará errado, mas também não estará completo. Drawables vão além de imagens. Pense em recursos visuais, tais como bordas, sombras, perfis geométricos, gradientes de cor, entre outros e você estará mais perto do que o conceito Drawable representa. Imagens PNG ou JPEG são apenas parte desse universo.

Bem, então quais são os drawables existentes? Segue a lista, diretamente extraída da nossa fonte primária de consulta:

https://developer.android.com/reference/android/graphics/drawable/Drawable

Bitmap: Este é o Drawable mais simples, bastante comum na web, uma imagem PNG ou JPEG. Importante ressaltar BMPs não são suportados em Android e não se recomenda o uso de outros formatos de imagem, mesmo que sejam suportados (como GIF, por exemplo);

9-Patch: Este é um conceito emprestado do artesanato, pense em uma colcha de retalhos, ou patchwork. Trata-se de uma extensão do formato PNG que suporta redimensionamento sem perder a qualidade da imagem. Estes arquivos tem a extensão 9.png e você pode esticá-lo ou encolhê-lo mantendo a qualidade.

Nestes links você encontrará mais informações e exemplos

https://developer.android.com/studio/write/draw9patch?hl=pt-br

https://developer.android.com/guide/topics/graphics/drawables?hl=pt-br#nine-patch

Vector: Este é um drawable bem diferente do que você imagina. Seu arquivo não contém uma imagem, mas sim texto. Trata-se de um arquivo XML que define como desenhar uma imagem como um conjunto de pontos, linhas e curvas junto com suas informações de cor associadas. Este tipo de drawable pode ser dimensionado à vontade sem perda de qualidade de exibição. Neste App usamos Vector nos três pequeno botões no topo da aplicação.

Shape: Também é um arquivo XML que contém comandos de desenho, mas é bem mais simples que um Vector. Também permite um bom redimensionamento, mas não é tão flexível como o Vector.

Layers: É um drawable composto, que desenha vários drawables subjacentes uns sobre os outros. Ele é um container para um conjunto de outros drawables superpostos para compor um efeito.

States: É um drawable composto que contém um conjunto com diversos drawables que serão usados como um leque de opções. Com base em seu estado, um de seus elementos é selecionado e exibido, se o estado for alterado troca-se a exibição.

Levels: Um drawable composto que seleciona um de um conjunto de drawables com base em seu nível.

Scale: Um drawable composto com um único drawable filho, cujo tamanho geral é modificado com base no nível atual.

Os últimos quatro tipos da lista acima fariam parte de um curso mais avançado de desenvolvimento Android. Não teremos aplicativos para exemplificá-los, pois precisamos investir tempo em outros assuntos do universo Android. Então, ficará por sua conta descobrir exatamente o que são e como usá-los.

Uma informação importante é que qualquer que seja o tipo de drawable que você irá usar, cada um será um arquivo (nos formatos PNG, JPEG ou XML) que deverá estar salvo dentro da pasta **...\app\src\main\res\drawable**. Além disso, o nome dos arquivos devem conter apenas letras minúsculas, algarismos e underline. Outros caracteres como letras maiúsculas, espaço em branco e caracteres especiais vão gerar erro na sua aplicação.

O Componente Switch

Você deve ter visto acima que usamos um componente Switch no nosso aplicativo. Este componente é um botão deslizante muito utilizado em situações em que necessitamos configurar algum parâmetro como ligado ou desligado. Ao usuário do aplicativo basta arrastar o botão deslizante de um lado para outro e poderá controlar o estado ligado/desligado.

Quando essa ação ocorre, o programa poderá usar o método `isChecked()` para identificar o estado do componente e tomar a ação que se fizer necessária. É o que foi feito no método `swtEspOcupadoOnClick()` do nosso aplicativo, onde tornamos visível ou invisível o a View `dividerImg` que foi posicionada atrás da imagem com o objetivo de demarcar a área ocupada por ela.

Saiba mais em: https://developer.android.com/reference/kotlin/android/widget/Switch

O componente Spinner

O Spinner é um container. Com ele podemos apresentar ao usuário uma lista de Strings e permitir a escolha de uma das opções. Em linhas gerais é um componente que permite de forma simples e intuitiva escolher um item dentro de um conjunto. Ao usuário, o spinner tem a aparência de um controle giratório que permite a rolagem de seus itens e a seleção de um.

Neste aplicativo vamos usar o Spinner para oferecer as opções de visualização da imagem. Estas opções são descritas na seção seguinte. Os itens carregados no Spinner e que serão as opções oferecidas ao usuário são configurados no atributo "entries" e neste aplicativo esse atributo será carregado a partir de um array de strings definido no arquivo strings.xml

No caso do Spinner não é possível criar um método para associar à propriedade onClick. Esta propriedade até está disponível no painel de atributos mas se você associar algum método a ela o componente ficará cinza, indicando erro e ao tentar rodar o aplicativo ele será cancelado.

Para poder usar a escolha feita no Spinner, é preciso implementar o listener AdapterView.OnItemSelectedListener() que contém dois métodos call-back internos: onItemSelected() no qual implementamos a função que ajusta a exibição da imagem à opção selecionada; onNothingSelected() no qual não implementamos nada, pois neste app não temos uso para essa opção.

Opções de Visualização de uma Imagem

Agora vamos tratar do atributo ScaleType do componente ImageView. É através deste atributo que configuramos a forma como a imagem será exibida. Existem opções listadas a seguir e a página de consulta primária que contém informações sobre estas opções pode ser acessada pelo link https://developer.android.com/reference/android/widget/ImageView.ScaleType

centerInside Redimensiona a imagem sem alterar sua razão de aspecto e de modo que a largura e a altura sejam menores ou iguais que as correspondentes medidas internas do componente. Após o redimensionamento a imagem é centralizada dentro da view. Várias situações podem ocorrer, dependendo dos fatores: imagem quadrada ou retangular, tamanho das medidas maiores, iguais ou menores que o tamanho do componente.

center Centraliza a imagem na view sem fazer redimensionamento.

centerCrop Redimensiona a imagem sem alterar sua razão de aspecto e de modo que a largura e a altura sejam iguais ou maiores que as correspondentes medidas internas do componente. Após o redimensionamento a imagem é centralizada dentro da view. Para ter efeito visível pelo menos uma das medidas da imagem precisa ser maior que a correspondente medida da view.

fitStart Redimensiona a imagem sem alterar sua razão de aspecto e de modo que a maior medida ocupe toda a medida interna correspondente. Ao menos um eixo (X ou Y) vai ficar do mesmo tamanho interno disponível. Após o redimensionamento alinha o eixo de menor medida da imagem ao início (start) da view – que será o topo se a menor medida for a altura ou será o lado esquerdo se a menor medida for a largura.

fitCenter Redimensiona a imagem sem alterar sua razão de aspecto e de modo que a maior medida ocupe toda a medida interna correspondente. Ao menos um eixo (X ou Y) vai ficar do mesmo tamanho interno disponível. Após o redimensionamento alinha o eixo de menor medida da imagem ao centro da view.

fitEnd Redimensiona a imagem sem alterar sua razão de aspecto e de modo que a maior medida ocupe toda a medida interna correspondente. Ao menos um eixo (X ou Y) vai ficar do mesmo tamanho interno disponível. Após o redimensionamento alinha o eixo de menor medida da imagem ao fim (end) da view – que será a base se a menor medida for a altura ou será o lado direito se a menor medida for a largura.

fitXY Ajusta a imagem dentro da view ocupando toda a sua área. Esta opção não mantém a razão de aspecto e pode distorcer a imagem.

matrix Altera os parâmetros de visualização da imagem com base em uma matriz de transformação. Com esta opção você poderá fazer qualquer coisa com a imagem como translação, rotação (em torno de qualquer eixo X, Y ou Z – é isso mesmo Z), aumento e redução de escala, cisalhamento (skewing). Trata-se de recurso muito avançado que envolve conceitos matemáticos de um dos tópicos da geometria conhecido como "transformações conforme". Não vamos tratar disso aqui, mas se você quiser ir adiante leia este artigo a respeito:

https://medium.com/a-problem-like-maria/understanding-android-matrix-transformations-25e028f56dc7

Mas como a própria autora do artigo sugere, tome um ou dois cafés antes e respire fundo. Vai ter uma matemática geométrica para compreender antes de usar.

Razão de Aspecto é a relação entre altura e largura da imagem. Se essa relação é alterada a imagem sofre uma distorção. Não alterá-la, mantém as proporções e não gera distorção.

Acima quando menciono "medida interna da view" me refiro ao tamanho da view (largura/altura) menos o respectivo padding.