Ficha do Aplicativo

App 03 – Aplicativo Brincando com Inteiros
Resumo Neste App03 poderemos gerar e exibir um número inteiro, verificar se ele é par ou ímpar, se é primo ou não e apresentar os divisores caso seja não primo não primo. Neste aplicativo vamos trabalhar com componentes de layout e views. Layouts são contêineres e usaremos o ConstraintLayout de modo programático. No caso dos componentes usaremos TextView, EditText e Button, já vistos nos aplicativos anteriores e acrescentaremos o RadioButton.
Objetivos de Aprendizagem 1. Trabalhar a criação de layouts que podem ser manipulados em tempo de execução do app. 2. Usar componentes do tipo Container na programação, no caso será usado o ConstraintLayout 3. Usar componentes do tipo View, no caso serão usados TextView e Button, já vistos no App01 da aula anterior e acrescentaremos o EditText e o RadioButton. 4. Possibilitar a rolagem de texto em um TextView. 5. Usar novamente a classe Toast para exibição de mensagens. 6. Personalizar a exibição de um Toast 7. Conhecer os conceitos iniciais sobre ciclo de vida de uma Activity.
Dinâmica do Aplicativo Este aplicativo conterá 2 RadioButtons com o objetivo de permitir a escolha do método de entrada para um número inteiro, que poderá ser gerado aleatório ou digitado. Em seguida, o botão “Calcular Propriedades” executará as verificações: **Paridade**: verificação se um número é par ou ímpar e coloca o resultado em um TextView **Primo**: verificação se o número é primo ou não e coloca o resultado em um TextView **Divisores**: determinação de todos os divisores do número e exibe o resultado em um TextView (um valor por linha) O botão “Limpar Propriedades” limpará todos os componentes.

Lista de Activities do Aplicativo

Nome	Layout
MainActivity.java	layout\activity_main.xml

Resources

strings.xml

```
<resources>
    <string name="app_name">Brincando com Inteiros</string>
    <string name="rotulo">Escolha a opção de obtenção do número</string>
    <string name="gerar">Gerar</string>
    <string name="digitar">Digitar</string>
    <string name="gerarnum">Gerar Número</string>
    <string name="calcular">Calcular Propriedades</string>
    <string name="paridade">Paridade</string>
    <string name="primo">Primo</string>
    <string name="divisores">Divisores</string>
    <string name="limpar">Limpar Propriedades</string>
</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="corPrimaria">#9603AE</color>
    <color name="corFundo">#EFDCC8</color>
    <color name="corFundoNumero">#FFFCF8</color>
    <color name="corTexto">#800090</color>
</resources>
```

Layout de faixa: pasta layout\digitar_numero.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/obtemNumero"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/corFundoNumero"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/btnGerarNum"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginStart="8dp"
        android:layout_weight="2"
        android:gravity="center_vertical"
        android:text="Digite o número" />

    <EditText
        android:id="@+id/edtNumeroDigitado"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:gravity="center_horizontal"
        android:layout_weight="3"
        android:ems="10"
        android:textColor="@color/corTexto"
        android:inputType="number" />

</LinearLayout>
```

Layout de faixa: pasta layout\gerar_numero.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/obtemNumero"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/corFundoNumero"
    android:orientation="horizontal">

    <Button
        android:id="@+id/btnGerarNum"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginStart="8dp"
        android:layout_weight="2"
```

```
        android:text="@string/gerarnum"
        app:cornerRadius="10dp" />

    <TextView
        android:id="@+id/txtNumeroGerado"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_weight="3"
        android:paddingTop="12dp"
        android:paddingBottom="12dp"
        android:ems="10"
        android:gravity="center_horizontal"
        android:textSize="18sp"
        android:textColor="@color/corTexto"
        android:inputType="number" />

</LinearLayout>
```

Detalhamento das Activities

Layout na posição retrato: pasta 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/layoutNumeros"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/corFundo"
    tools:context=".MainActivity">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/gdl"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.3" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:gravity="center_vertical"
        android:text="@string/rotulo"
        android:textColor="@color/corTexto"
        app:layout_constraintBottom_toBottomOf="@+id/radgOpcoes"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/radgOpcoes" />

    <RadioGroup
        android:id="@+id/radgOpcoes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView"
        app:layout_constraintTop_toTopOf="parent">

        <RadioButton
            android:id="@+id/radGerar"
            android:layout_width="match_parent"
            android:layout_height="40dp"
            android:text="@string/gerar"
            android:textColor="@color/corTexto" />

        <RadioButton
            android:id="@+id/radDigitar"
            android:layout_width="wrap_content"
            android:layout_height="40dp"
            android:text="@string/digitar"
            android:textColor="@color/corTexto" />
    </RadioGroup>

    <!--
    <include
```

```
        layout="@layout/gerar_numero"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@id/radgOpcoes"
        />
        -->

    <Button
        android:id="@+id/btnCalcPropr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:text="@string/calcular"
        app:cornerRadius="10dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/radgOpcoes" />

    <Button
        android:id="@+id/btnLimpaPropr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:text="@string/limpar"
        app:cornerRadius="10dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="2dp"
        android:background="@color/corFundoNumero"
        android:paddingStart="4dp"
        android:paddingEnd="8dp"
        android:text="@string/paridade"
        android:textColor="@color/corTexto"
        android:textSize="20dp"
        app:layout_constraintEnd_toStartOf="@+id/gdl"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnCalcPropr" />

    <TextView
        android:id="@+id/txtParidade"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:background="@color/corFundoNumero"
        android:paddingStart="8dp"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/gdl"
        app:layout_constraintTop_toTopOf="@+id/textView2" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="2dp"
        android:background="@color/corFundoNumero"
        android:paddingStart="4dp"
        android:paddingEnd="8dp"
        android:text="@string/primo"
        android:textColor="@color/corTexto"
        android:textSize="20dp"
        app:layout_constraintEnd_toStartOf="@+id/gdl"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />

    <TextView
```

```
        android:id="@+id/txtPrimo"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:background="@color/corFundoNumero"
        android:paddingStart="8dp"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/gdl"
        app:layout_constraintTop_toTopOf="@+id/textView4" />

    <TextView
        android:id="@+id/textView6"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="2dp"
        android:background="@color/corFundoNumero"
        android:paddingStart="4dp"
        android:paddingEnd="8dp"
        android:text="@string/divisores"
        android:textColor="@color/corTexto"
        android:textSize="20dp"
        app:layout_constraintBottom_toBottomOf="@+id/txtDivisores"
        app:layout_constraintEnd_toStartOf="@+id/gdl"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/txtDivisores" />

    <TextView
        android:id="@+id/txtDivisores"
        android:layout width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="16dp"
        android:layout_marginBottom="24dp"
        android:background="@color/corFundoNumero"
        android:paddingStart="8dp"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/btnLimpaPropr"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout constraintStart toStartOf="@+id/gdl"
        app:layout_constraintTop_toBottomOf="@+id/textView4" />


</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 static final int OPC_GERAR = 1;
    private static final int OPC_DIGITAR = 2;

    private ConstraintLayout layoutNumeros;
    private View faixaNumero;
    private RadioButton radGerar;
    private RadioButton radDigitar;
    private TextView txtNumero;
    private Button btnCalcPropr;
    private Button btnLimpaPropr;
    private Random gerador;
    private TextView txtParidade;
    private TextView txtPrimo;
    private TextView txtDivisores;
    private int opcEscolhida;
    private int numero;

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

        // Código caso queira alterar a cor da barra de ações - lembrar de no arquivo de temas remover o item
NoActionBar (se não fizer isso o código abaixo dá erro)
        ActionBar actionBar = getSupportActionBar();
        ColorDrawable colorDrawable = new ColorDrawable(Color.parseColor("#9603AE"));
        actionBar.setBackgroundDrawable(colorDrawable);
        actionBar.setTitle(Html.fromHtml("<font color='#FFFFFF'>Brincando com Inteiros</font>", 0));

        opcEscolhida = 0;
        layoutNumeros = findViewById(R.id.layoutNumeros);
        setRadOpcoes();

        gerador = new Random();
        setBtnCalcPropr();
        setBtnLimpaPropr();

        txtParidade = findViewById(R.id.txtParidade);
        txtPrimo = findViewById(R.id.txtPrimo);
        txtDivisores = findViewById(R.id.txtDivisores);
        txtDivisores.setMovementMethod(new ScrollingMovementMethod());
    }

    //region Código de configuração do topo da Activity - RadioButtons e faixa dinâmica criada conforme a
opção
    private void setRadOpcoes() {
        radGerar = findViewById(R.id.radGerar);
        radDigitar = findViewById(R.id.radDigitar);

        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (view.getId() == R.id.radGerar)
                    setOpcEscolhida(OPC_GERAR);
                else if (view.getId() == R.id.radDigitar)
                    setOpcEscolhida(OPC_DIGITAR);
            }
        };

        radGerar.setOnClickListener(listener);
        radDigitar.setOnClickListener(listener);
    }

    public void setOpcEscolhida(int opcEscolhida) {
        this.opcEscolhida = opcEscolhida;
        if (opcEscolhida == OPC_GERAR)
```

```
            criaViewGerar();
        else if (opcEscolhida == OPC_DIGITAR)
            criaViewDigitar();
    }

    private void removeFaixaNumero() {
        layoutNumeros.removeView(faixaNumero);
    }

    private void criaViewGerar() {
        removeFaixaNumero();
        criaFaixaNumero(R.layout.gerar_numero);
        txtNumero = faixaNumero.findViewById(R.id.txtNumeroGerado);
        criaBtnGerarOnClick();
    }

    private void criaViewDigitar() {
        removeFaixaNumero();
        criaFaixaNumero(R.layout.digitar_numero);
        txtNumero = faixaNumero.findViewById(R.id.edtNumeroDigitado);
    }

    private void criaFaixaNumero(@LayoutRes int resource) {
        faixaNumero = getLayoutInflater().inflate(resource, layoutNumeros, false);
        layoutNumeros.addView(faixaNumero);

        ViewGroup.MarginLayoutParams lp = (ViewGroup.MarginLayoutParams) faixaNumero.getLayoutParams();
        lp.topMargin = 8;
        faixaNumero.setLayoutParams(lp);

        ConstraintSet cs = new ConstraintSet();
        cs.clone(layoutNumeros);
        cs.connect(faixaNumero.getId(), ConstraintSet.TOP, R.id.radgOpcoes, ConstraintSet.BOTTOM);
        cs.applyTo(layoutNumeros);
    }

    private void criaBtnGerarOnClick() {
        Button btnGerarNum = faixaNumero.findViewById(R.id.btnGerarNum);
        btnGerarNum.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int n = gerador.nextInt(100000);
                txtNumero.setText(String.format("%d", n));
            }
        });
    }
//endregion Código de configuração do topo da Activity - RadioButtons e faixa dinâmica criada conforme a
opção


    //region Código de configuração dos botões de cálculo e limpeza de propriedades do número
    private void setBtnCalcPropr() {
        btnCalcPropr = findViewById(R.id.btnCalcPropr);
        btnCalcPropr.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                trataBtnCalcProprOnClick();
            }
        });
    }

    private void setBtnLimpaPropr() {
        btnLimpaPropr = findViewById(R.id.btnLimpaPropr);
        btnLimpaPropr.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                trataBtnLimpaPropr();
            }
        });
    }

    private void trataBtnCalcProprOnClick() {
        if (opcEscolhida > 0) {
            if (txtNumero != null && !txtNumero.getText().toString().equals(""))
                setNumero(Integer.parseInt(txtNumero.getText().toString()));
            else
                exibeToast("Gere ou digite um número", Toast.LENGTH_LONG);
        }
        else
            exibeToast("Escolha uma opção\nGerar/Digitar", Toast.LENGTH_LONG);
    }
```

```
    private void trataBtnLimpaPropr() {
        if (txtNumero != null) {
            txtNumero.setText("");
            txtParidade.setText("");
            txtPrimo.setText("");
            txtDivisores.setText("");
            numero = 0;
        }
    }
//endregion Código de configuração dos botões de cálculo e limpeza de propriedades do número


    //region Código de cálculo de propriedades do número
    private void setNumero(int numero) {
        this.numero = numero;
        calculaPropriedades();
    }

    private void calculaPropriedades() {
        // Paridade
        if (numero % 2 == 0)
            txtParidade.setText("Par");
        else
            txtParidade.setText("Ímpar");

        // Primo
        if (ePrimo(numero))
            txtPrimo.setText("sim, é Primo");
        else
            txtPrimo.setText("não é Primo");

        // Divisores
        String s = produzDivisores(numero);
        txtDivisores.setText(s);
    }

    private boolean ePrimo (int pN) {
        int i;
        int r = 1;
        double raiz;
        if (pN == 2)
            return true;
        else if (pN % 2 == 0)
            return false;
        else {
            raiz = Math.sqrt(pN);
            i = 3;
            while (i <= raiz && r != 0) {
                r = pN % i;
                i += 2;
            }
            return r != 0;
        }
    }

    private String produzDivisores(int valor) {
        int i;
        double fim;
        String s = "";
        fim = valor / 2;
        i = 2;
        while (i <= fim) {
            if (valor % i == 0)
                s = s + i + "\n";
            i++;
        }
        return s;
    }
//endregion Código de cálculo de propriedades do número


//region Toast personalizado
    private void exibeToast(String txt, int duration) {
        LayoutInflater inflater = getLayoutInflater();
        View toastLayout = inflater.inflate(R.layout.toast, findViewById(R.id.raiz_layout_toast));
        ImageView imagem_toast = toastLayout.findViewById(R.id.imagem_toast);
        imagem_toast.setImageResource(R.drawable.ic_alerta);
        TextView texto_toast = toastLayout.findViewById(R.id.texto_toast);
        texto_toast.setText(txt);
```

```
        Toast toast = new Toast(getApplicationContext());
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);
        toast.setDuration(duration);
        toast.setView(toastLayout);
        toast.show();
    }
//endregion Toast personalizado

}
```

Notas Técnicas

Layout Responsivo

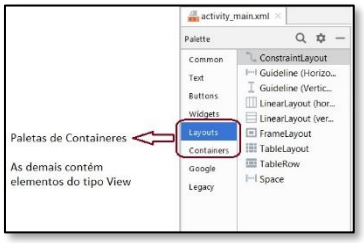
Existem dispositivos físicos com uma gama muito variados em termos de formatos e tamanhos. Deste modo um aplicativo para esses dispositivos precisa ter seu layout visual muito flexível. Um layout que seja bem visualizado, ou seja, tenha uma boa aparência, nos diversos dispositivos existentes é dito "responsivo".

Desta forma, o desenvolvedor não pode criar seus layouts com proporções, posições de componentes e dimensões rígidas, pressupondo um formato e tamanho específico de aparelho. Um bom layout precisa ter a capacidade de responder adequadamente e de forma eficiente a variações de tamanho, densidade e orientação da tela. Daí vem o nome "Layout Responsivo".

Até por questões comerciais, convém ao desenvolvedor que seu aplicativo possa ser executado na mais ampla gama possível de aparelhos em uso pelo seu público-alvo.

Algumas técnicas e cuidados devem ser empregadas para atingir este objetivo:

- Usar o componente ConstraintLayout como elemento básico de layout da tela (interface do usuário);
- Na posição e dimensões de componentes usar a unidade Density Pixel (DP) e nunca usar pixels físicos;
- Nos tamanhos de fonte (texto) usar a unidade Scale Pixel (SP), que de forma automática faz a combinação de Density Pixels com o tamanho de fonte da preferência do usuário. Este último pode ser configurado no pelo dono do aparelho no menu de configurações do Android;
- Usar dimensões parametrizadas (wrap_content, match_constraint, match_parent) capazes de se adaptar a diferentes tamanhos físicos de tela, redimensionando o layout;
- Combinar de modo conveniente os componentes dos tipos Container e View.



Containers são componentes capazes de agrupar em seu espaço interno outros componentes, dos dois tipos. Os componentes desse tipo ficam nas paletas Layouts e Containers.

Views são componentes com os quais o usuário interage diretamente. Os componentes desse tipo ficam nas outras paletas.

Para saber mais consulte o link: https://developer.android.com/training/multiscreen/screensizes?hl=pt-br

Notificação Toast

Conforme já foi visto antes, 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. A figura ao lado mostra como um Toast padrão é exibido. Para exibi-la deve-se usar a linha de código exibida a seguir.


```
Toast.makeText(this, "Aqui vai a mensagem.", Toast.LENGTH_SHORT).show();
```

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

Notificação Toast Personalizada

O Toast padrão já é suficiente para exibição de mensagens rápidas. Porém, caso o desenvolvedor queira acrescentar um aspecto visual diferenciado isso é possível. Para isso é preciso criar um arquivo XML com o layout desejado. Em seguida, no código deve-se inflar esse layout e associá-lo ao Toast antes de fazer a exibição. A imagem ao lado mostra o resultado da execução do código abaixo.

```
private void exibeToast(String txt, int duration) {
    LayoutInflater inflater = getLayoutInflater();
    View toastLayout = inflater.inflate(R.layout.toast,findViewById(R.id.raiz_layout_toast));
    ImageView imagem_toast = toastLayout.findViewById(R.id.imagem_toast);
    imagem_toast.setImageResource(R.drawable.ic_alerta);
    TextView texto_toast = toastLayout.findViewById(R.id.texto_toast);
    texto_toast.setText(txt);

    Toast toast = new Toast(getApplicationContext());
    toast.setGravity(Gravity.CENTER_VERTICAL, 0, 100);
    toast.setDuration(duration);
    toast.setView(toastLayout);
    toast.show();
}
```

Neste código o layout contido no arquivo toast.xml é lido e para cada um de seus elementos é criado um objeto Java que pode ser manipulado pelo programador. Quem permite essa operação é a classe LayoutInflater com a qual é feita a operação conhecida como “inflar a view”. Essa palavra “inflar” é comumente usada e representa exatamente a ação de criação de objetos a partir do arquivo XML.

Ciclo de Vida de uma Atividade

O Ciclo de Vida de uma Atividade é um conceito que deve ser bem conhecido pelos desenvolvedores de aplicativos para a plataforma Android. É preciso conhecer a maneira como as atividades são iniciadas e organizadas, a forma como as atividades interagem com o sistema operacional e umas com as outras. E é disso que vamos tratar agora.

A seguir fazemos uma descrição completa do Ciclo de Vida, embora neste aplicativo não usaremos todos os métodos envolvidos.

Uma atividade (Activity) está associada à uma tela (Layout) onde é desenhada a interface do usuário. Em geral, os aplicativos possuem múltiplas atividades com suas respectivas telas.

Normalmente, escolhe-se uma atividade para ser a principal (MainActivity), que será a primeira tela a aparecer quando o usuário inicia o aplicativo. A partir daí, podem-se iniciar outras atividades para executar ações específicas dentro do aplicativo. Por exemplo, a atividade principal pode mostrar uma lista geral de produtos e, a partir dessa lista, ao tocar em um produto, dispara-se uma segunda atividade para visualizar em tela cheia os detalhes do produto que não caberiam na lista geral. Desta forma existe no Android um mecanismo que permite que uma Atividade inicie outra.

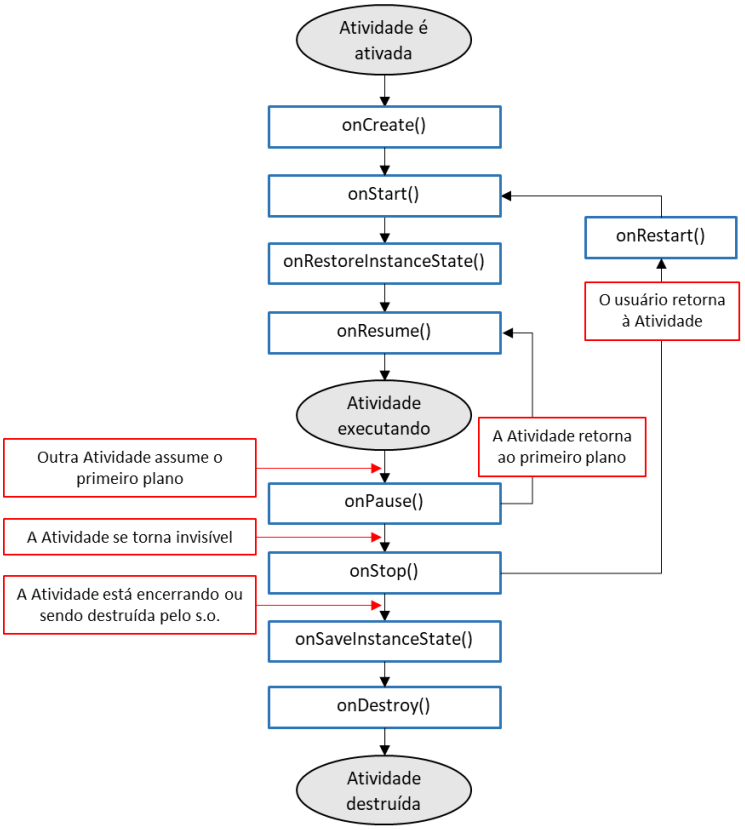
Um Aplicativo é, portanto, composto por uma coleção de atividades, que são planejadas para compor um conjunto coeso e coerente. No entanto, o grau de dependência entre elas é muito baixo, cada atividade pode ser iniciada a qualquer momento (vamos aprender a fazer isso mais adiante) e nada impede que esse início ocorra a partir de uma atividade de outro aplicativo, feito por terceiros.

Isso mesmo, uma atividade do seu aplicativo pode ser iniciada pelo aplicativo feito pelo seu colega e vice-versa. Isso traz inúmeros benefícios, permitindo, por exemplo, que o seu aplicativo inicie um outro para envio de e-mails, abra a galeria, acione a câmera ou quaisquer outras atividades de aplicativos instalados no dispositivo e que aceitem interação externa. Além disso, para tudo isso acontecer não é obrigatório que o seu aplicativo inicie a MainActivity do outro aplicativo. Ele pode iniciar a atividade que for necessária.

Diferentemente dos paradigmas de programação nos quais os aplicativos são iniciados com o método main () ou similar, o sistema Android inicia o código em uma instância de uma atividade – que pertence a um aplicativo, mas que pode ser iniciada diretamente. Para manter a coerência e garantir o correto funcionamento das aplicações, o programador deve conhecer os métodos (funções inseridas nas classes) que são invocados a cada etapa da execução da atividade. Tais métodos podem ou não ser implementados e cada um corresponde a um estágio específico do seu ciclo de vida.

Essa é uma estratégia muito própria do Android e confere grande flexibilidade ao iniciar, rodar e encerrar cada atividade. À medida que o usuário navega pelo aplicativo cada Atividade muda de estado dentro de um conjunto de possibilidades que compõem o que é conhecido como Ciclo de Vida da Atividade.

A figura ao lado mostra os estados em que uma Atividade pode estar (elipses) e os métodos (caixas de borda azul) que podem ser implementados para tratar as transições de um estado para outro. A causa das transições são destacadas nas caixas de borda vermelha.



Seis desses métodos – onCreate(), onStart(), onResume(), onPause(), onStop() e onDestroy() – dizem respeito ao ciclo de vida da atividade propriamente dito. Os outros dois – onSaveInstanceState() e onRestoreInstanceState() – são usados, respectivamente, para salvar e restaurar o estado da atividade.

Descrição dos métodos do ciclo de vida da Atividade

onCreate(Bundle) – é executado quando a Activity é criada. Pode ser usado para instanciar quaisquer objetos a serem usados na Activity, para inicializar variáveis locais e para criar Listeners e associá-los a Views. Esse método fornece um Bundle que permite recuperar um estado salvo anteriormente.

onStart() – é executado quando a atividade está se tornando visível. Lembrando que estar visível não é o mesmo que ter o foco. Uma segunda atividade que ocupe apenas parte da tela ou possua transparência pode ter o foco, ao mesmo tempo que permite que a outra esteja visível.

onResume() – é executado quando a atividade recebe o foco.

onPause() – é executado quando a atividade deixa de estar no primeiro plano, ou seja, perde o foco. Isto ocorre porque outra atividade está iniciando. A atividade seguinte não inicia de fato até que este método termine. Por isso o código do mesmo deve sempre ser rápido.

onStop() – é executado quando a atividade deixa de estar visível. Ocorre em duas situações: ou porque outra atividade a está cobrindo ou porque a atividade está sendo destruída. É sucedido por onRestart() se a atividade se tornar visível outra vez, ou por onDestroy() se a atividade vai ser destruída.

onRestart() – é executado quando a atividade está se tornando novamente visível depois de ter se tornado invisível.

onDestroy() – é executado quando a atividade está sendo destruída, seja porque houve reconfiguração da mesma (o usuário virou o aparelho, p.ex) ou porque o aplicativo foi fechado

onRestoreInstanceState(Bundle) – é executado imediatamente após o método onStart() quando a atividade está sendo reinicializada a partir de um estado que tenha sido previamente salvo usando onSaveInstanceState(). A maior

onSaveInstanceState(Bundle) – é executado imediatamente antes de onDestroy(). Este método oferece um meio de salvar o estado da atividade através de um Bundle que pode ser recuperado se a atividade for reiniciada.

Para saber mais consulte estes links oficiais do Android

Básico sobre Activities: https://developer.android.com/guide/components/activities/intro-activities

Sobre o ciclo de vida de uma Activity: https://developer.android.com/guide/components/activities/activity-lifecycle