Projetos de Aplicativos Móveis

Prof. Fernando Antonio Fernandes Anselmo (Versão 1.2)

1 Introdução ao Android

O sistema operacional **Android**, é líder de mercado neste segmento, utiliza Java como a linguagem de programação para seus aplicativos, e traz um conjunto rico de classes e ferramentas para o desenvolvimento de programas que façam uso dos principais recursos existentes nestes aparelhos. Além disso tem como base o Kernel (núcleo) do Sistema Operacional Linux.

Nesta veremos o escopo dos projeto e entenderemos os conceitos básicos na criação dos aplicativos, Um aplicativo para Android é um software desenvolvido para ser executado em dispositivos móveis Android, como smartphones, tablets e smartwatches. Geralmente, um aplicativo é composto por vários componentes, inclui:

- **Activity**: representa a tela do aplicativo que o usuário pode interagir.
- **Layout**: define a aparência visual das *activities* e como os elementos da interface do usuário (botões, campos de texto, imagens, etc.) são organizados e exibidos.
- **Intent**: permite a comunicação entre diferentes componentes do aplicativo ou entre aplicativos.
- **Service**: executa tarefas em segundo plano, como reprodução de música ou sincronização de dados.
- **Broadcast receiver**: permite que o aplicativo receba eventos do sistema ou de outros aplicativos.
- **Content provider**: permite que o aplicativo acesse e compartilhe dados com outros aplicativos.

Além desses componentes, um aplicativo Android geralmente é desenvolvido usando a linguagem de programação Java (ou Kotlin) e usa a API do Android para interagir com o sistema operacional e os recursos do dispositivo. O aplicativo é empacotado em um arquivo de pacote Android (APK) e distribuído por meio da Google Play Store ou outras lojas de aplicativos.

Os aplicativos Android podem ter diferentes finalidades e funcionalidades, desde jogos, aplicativos de mídia social, aplicativos de produtividade até aplicativos de negócios e muito mais. A interface do usuário pode ser personalizada para atender às necessidades do aplicativo e pode incluir gráficos e animações avançadas para fornecer uma experiência de usuário agradável e envolvente.

2 Hello World com Menu

Para demonstração de como funciona a aplicação padrão gerada pelo Android, vamos modificar esta e adicionar um simples menu, para tanto uma aplicação deve ser gerada através da "*Basic Activity*".

Objetivo: Compreender o projeto básico criado.

Para o nosso primeiro projeto vamos realizar um passo a passo completo, para os próximos projetos essa fase deve ser repetida, após iniciar a IDE Android Studio pressionar o botão **New Project** (na tela Welcome to Android Studio).

Se ao iniciar sua IDE cair direto na tela de projeto, selecione a partir do Menu Principal, as opções "File ▷ New ▷ New Project...". Podemos marcar, sempre que a IDE for inicializada ir para a tela de novo projeto, para tanto, em "System Settings ▷ Appearance and Behavior ▷ System Settings", desmarque a opção "Reopen projects on startup".

Nas templates do projeto usar a "Atividade Vazia"conforme a seguinte imagem:

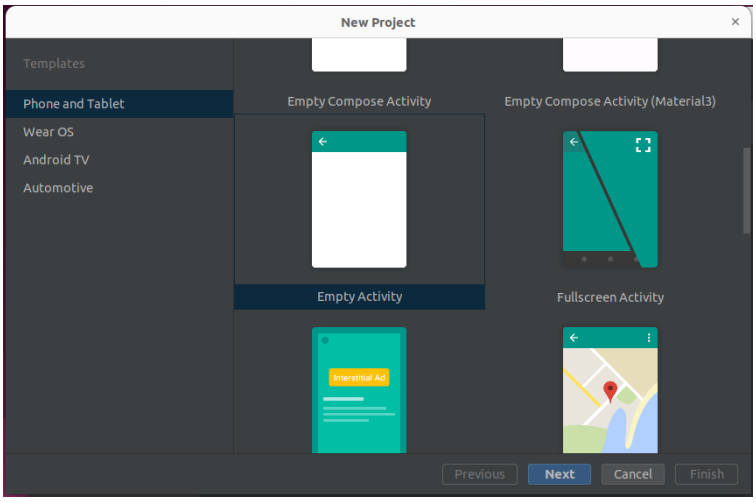

Figura 1: *Android Studio*

Em seguida marque as opções do projeto:

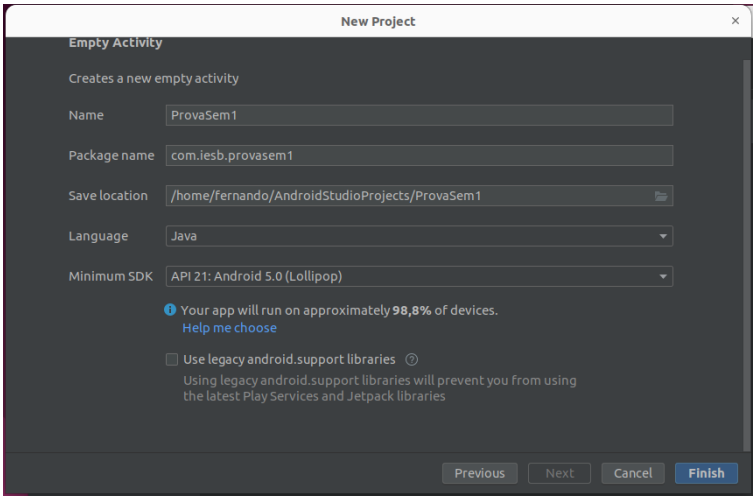

Figura 2: *Android Studio*

Definir os recursos de cor para o XML de valor **color.xml**:

```
<color name="purple_500">#FF6200EE</color>
<color name="purple_700">#FF3700B3</color>
```

Verificar se o arquivo de **themes.xml** está correto, e herda da classe que apresenta a barra de menu:

```
<style name="Base.Theme.GraphBasic"
 parent="Theme.MaterialComponents.DayNight.DarkActionBar">
<item name="colorPrimary">@color/purple_500</item>
<item name="colorPrimaryVariant">@color/purple_700</item>
...
</style>
```

Criar em **res** uma subpasta chamada **menu** e nesta um arquivo chamado main.xml,com o seguinte

conteúdo:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto">
  <item
    android:id="@+id/acao1"
    android:title="@string/opc1" />
  <item
    android:id="@+id/acao2"
    android:title="@string/opc2" />
</menu>
```

Dica: Usar **ALT + Enter** para transferir a String. E o resultado do arquivo **strings.xml** (na pasta values), será este:

```
<string name="opc1">Opção 1</string>
<string name="opc2">Opção 2</string>
```

Adicionar 2 métodos na **MainActivity.java**:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.main, menu);
   return true;
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
  Context context = getApplicationContext();
  int duration = Toast.LENGTH_SHORT;
  Toast toast = null;
  switch (item.getItemId()) {
    case R.id.acao1:
      toast = Toast.makeText(context, getString(R.string.opc1), duration);
      toast.show();
      return true;
    case R.id.acao2:
      toast = Toast.makeText(context, getString(R.string.opc2), duration);
      toast.show();
      return true;
    default:
      return super.onOptionsItemSelected(item);
  }
}
```

3 Layout e Activity

Layout é um componente fundamental do desenvolvimento de aplicativos Android que representa a aparência e organização visual de uma tela, ou seja, a interface do usuário. O *layout* é responsável por definir a posição, tamanho e estilo dos elementos visuais da tela, como botões, campos de texto, imagens e outros *widgets*.

O *layout* é definido em um arquivo XML (Extensible Markup Language) separado, que é carregado pelo sistema Android quando a *activity* (falaremos a seguir das *Activities*) correspondente é exibida. O arquivo XML contém uma hierarquia de elementos de layout, que descrevem como os widgets e outros elementos visuais da tela devem ser organizados e posicionados.

Existem vários tipos de *layouts* disponíveis no Android, cada um adequado para diferentes tipos de aplicativos e interfaces do usuário. Alguns exemplos incluem:

- **LinearLayout**: organiza os elementos visuais em uma única linha ou coluna.
- **RelativeLayout**: organiza os elementos visuais em relação uns aos outros, permitindo a criação de layouts mais complexos e dinâmicos.
- **FrameLayout**: organiza os elementos visuais em camadas, com cada elemento empilhado em cima do outro.
- **GridLayout**: organiza os elementos visuais em uma grade de linhas e colunas.

Além disso, os *layouts* do Android também suportam a capacidade de dimensionar e posicionar dinamicamente os elementos visuais da tela em resposta a diferentes resoluções de tela e orientações de dispositivo.

Objetivo: Como utilizar os componentes básicos do Android e o *LinearLayout* para a criação da nossa atividade.

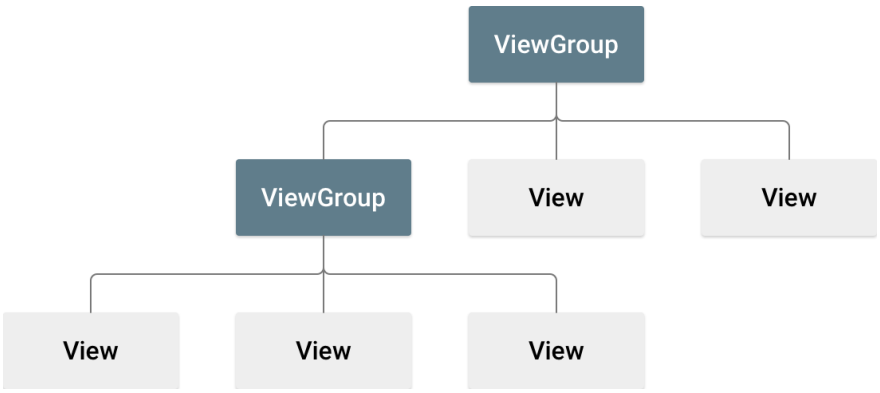

Figura 3: *ViewGroup e View*

3.1 Projeto Mostra Frase

Modificar a activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  <Space
```

```
    android:layout_width="match_parent"
    android:layout_height="50dp" />
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Frase para mostrar!"
    android:layout_gravity="center_horizontal" />
  <Space
    android:layout_width="match_parent"
    android:layout_height="100dp" />
  <Button
    android:id="@+id/botao"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Repetir"
    android:layout_gravity="center_horizontal" />
  <Space
    android:layout_width="match_parent"
    android:layout_height="100dp" />
  <EditText
    android:id="@+id/texto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Digite aqui sua frase" />
</LinearLayout>
```

match_parent - fornece apenas o espaço necessário para o objeto // wrap_content - aloca todo o espaço, seja da largura ou altura.

Modificar o método onCreate na **MainActivity.java**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  final EditText texto = (EditText)findViewById(R.id.texto);
  final Context ctx = getApplicationContext();

  findViewById(R.id.botao).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      Toast torrada = Toast.makeText(ctx, texto.getText(), Toast.LENGTH_LONG);
      torrada.show();
    }
  });
}
```

3.2 Projeto Alterna Tela

Uma *activity* é um componente fundamental do desenvolvimento de aplicativos Android. Pois representa a tela com uma interface do usuário, onde este pode interagir com o aplicativo. As *activities* são usadas para exibir informações, coletar entradas do usuário e iniciar outras, como a abertura de outras telas ou a realização de outras tarefas.

Cada *activity* é geralmente implementada como uma classe Java, que herda da classe base *Activity* do framework Android. A classe define a aparência e o comportamento da *activity*, como a exibição de elementos da interface, processamento de eventos e a interação com outros componentes do sistema.

Uma aplicação Android pode ter várias *activities*, representa uma tela ou fluxo de interação específico do usuário. Por exemplo, uma aplicação de notícias pode ter uma *activity* para exibir uma lista de artigos, outra *activity* para exibir o conteúdo de um artigo individual e outra *activity* para permitir que o usuário faça uma pesquisa.

As *activities* são gerenciadas pelo sistema Android e são empilhadas em uma pilha de atividades, onde a atividade mais recente é exibida na parte superior da pilha. Quando uma *activity* é iniciada, será colocada na parte superior da pilha e, quando o usuário pressiona o botão "voltar", a activity mais recente é removida da pilha e a *activity* anterior é exibida.

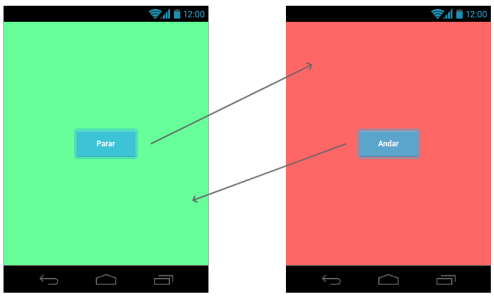

Figura 4: *Projeto Troca Tela*

Objetivo: Demonstrar como funciona a alternância de *activities* no Android.

Adicionar no colors.xml:

```
<color name="vermelho">#F00</color>
<color name="verde">#0F0</color>
```

Modificar a activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:background="@color/verde">
  <Button
    android:id="@+id/botaoParar"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Parar"
    android:layout_gravity="center_horizontal" />
</LinearLayout>
```

Criar um layout chamado activity_secundary.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="center"
  android:background="@color/vermelho">
  <Button
    android:id="@+id/botaoAndar"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Andar"
    android:layout_gravity="center_horizontal" />
</LinearLayout>
```

Na MainActivity.java, modifique os seguintes métodos:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  findViewById(R.id.botaoParar).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      chamarParar();
    }
  });
}
private void chamarParar() {
  startActivity(new Intent(this, SecundaryActivity.class));
}
```

Criar a SecundaryActivity.java e modificar os seguintes métodos:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_secundary);
  findViewById(R.id.botaoAndar).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      chamarAndar();
    }
  });
}
private void chamarAndar() {
  finish();
}
```

Adicionar no AndroidManifest.xml (app ▷ manifests):

```
    ...
```

```
    </intent-filter>
  </activity>
  <activity android:name=".SecundaryActivity"></activity>
</application>
```

3.3 Projeto Alterna Tela 2

Objetivo: Demonstrar um modelo mais dinâmico de programação sem alto custo de *Activities*.

Adicionar a propriedade ao botaoParar na activity_main.xml:

```
  android:onClick="chamarParar"
```

Adicionar a propriedade ao botaoAndar na activity_secundary.xml:

```
  android:onClick="chamarAndar"
```

Eliminar o método setOnClickListener da MainActivity.java e da SecundaryActivity.java. E mudar a assinatura dos métodos chamarParar() e chamarAndar() nos respectivos programas para:

```
public void chamarParar(View view) { ... }

public void chamarAndar(View view) { ... }
```

4 Ciclo de Vida e Mais Cores

Demonstrar o uso do Ciclo de Vida e a utilização das cores.

4.1 Projeto Ciclo de Vida

O ciclo de vida para uma Activity é composto por diversos estados que indicam o seu estado atual, desde o momento em que é criada até finalizada. O ciclo de vida é importante porque afeta a forma como a Activity se comporta em relação aos outros componentes do sistema, como outras Activities, Services e Broadcast Receivers.

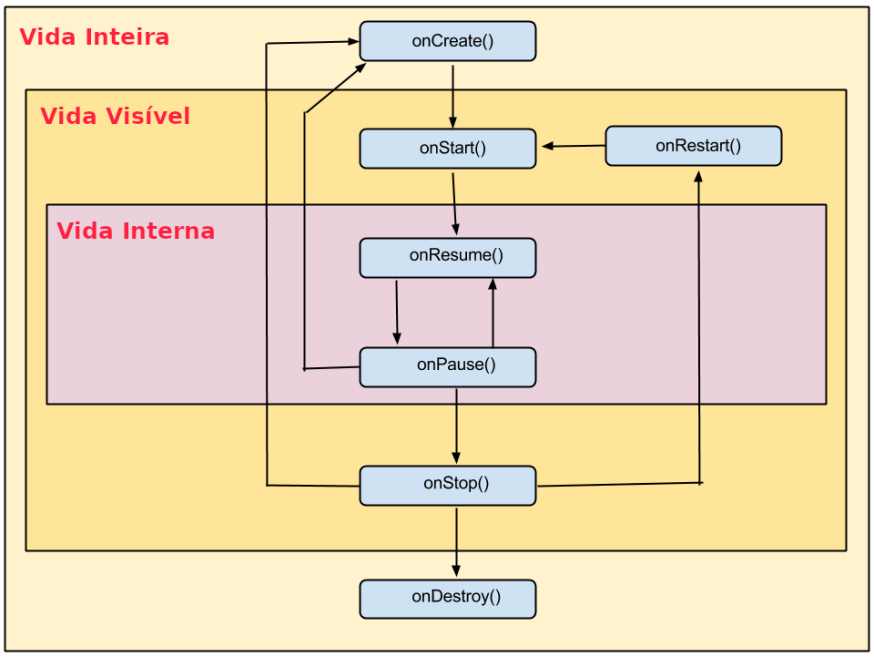

Figura 5: *Esquema do Ciclo de Vida*

O ciclo de vida de uma Activity é composto por sete estados principais:

- **Created (Criada)**: Este é o primeiro estado de uma Activity. A Activity é criada e o método onCreate() é chamado. Neste estado, a Activity ainda não é visível para o usuário.
- **Started (Iniciada)**: Quando a Activity é iniciada, o método onStart() é chamado. Neste estado, a Activity já foi criada e está visível para o usuário.
- **Resumed (Retomada)**: Quando a Activity está em primeiro plano e o usuário está interagindo com ela, o método onResume() é chamado. Neste estado, a Activity está em pleno funcionamento e é responsável por fornecer uma experiência interativa para o usuário.
- **Paused (Pausada)**: Quando outra Activity é iniciada ou quando a Activity atual perde o foco, o método onPause() é chamado. Neste estado, a Activity ainda está visível, mas não está mais em primeiro plano.
- **Stopped (Interrompida)**: Quando a Activity é interrompida, o método onStop() é chamado. Neste estado, a Activity não está mais visível para o usuário e pode ser removida da memória pelo sistema operacional, caso seja necessário liberar recursos.
- **Restarted (Reiniciada)**: Quando a Activity é reiniciada, o método onRestart() é chamado. Neste estado, a Activity está sendo preparada para ser iniciada novamente.
- **Destroyed (Destruída)**: Quando a Activity é finalizada, o método onDestroy() é chamado. Neste estado, a Activity é completamente removida da memória pelo sistema operacional.

Durante o ciclo de vida em uma Activity, é possível que esta passe por diferentes estados, dependendo da interação do usuário com o aplicativo ou de outros fatores externos, como a disponibilidade de recursos do sistema. Para lidar com esses diferentes estados, é importante que os desenvolvedores implementem as diferentes funções de callback do ciclo de vida da Activity, para garantir que a aplicação se comporte de maneira adequada em todas as situações.

Criamos um novo projeto e vamos modificar a activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:gravity="center">
  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:id="@+id/chamar"
    android:onClick="chamarAcao" />
</LinearLayout>
```

Modificar a classe **MainActivity.java**

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toast.makeText(this, "1. OnCreate", Toast.LENGTH_SHORT).show();
  }

  public void chamarAcao(View view) {
    startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com")));
  }

  @Override
  protected void onStart() {
    super.onStart();
    Toast.makeText(this, "2. OnStart", Toast.LENGTH_SHORT).show();
  }

  @Override
  protected void onResume() {
    super.onResume();
    Toast.makeText(this, "3. OnResume", Toast.LENGTH_SHORT).show();
  }
```

```
33
34    @Override
35    protected void onRestart() {
36      super.onRestart();
37      Toast.makeText(this, "4. OnRestart", Toast.LENGTH_SHORT).show();
38    }
39
40    @Override
41    protected void onPause() {
42      super.onPause();
43      Toast.makeText(this, "5. OnPause", Toast.LENGTH_SHORT).show();
44    }
45
46    @Override
47    protected void onStop() {
48      super.onStop();
49      Toast.makeText(this, "6. OnStop", Toast.LENGTH_SHORT).show();
50    }
51
52    @Override
53    protected void onDestroy() {
54      super.onDestroy();
55      Toast.makeText(this, "7. OnDestroy", Toast.LENGTH_SHORT).show();
56    }
57  }
```

Adicionar no AndroidManifest.xml (app ▷ manifests):

```
1  ...
2  <uses-permission android:name="android.permission.INTERNET" />
3  <application
4  ...
```

4.2 Projeto Arco Íris

O Android permite as seguintes unidades de medida:

- **px** (pixel): corresponde a um pixel de janela (basicamente 1 pixel possui a 0,26458 cm)
- **in** (polegadas): com base no tamanho físico da janela (basicamente 1 polegada possui 2,54 cm).
- **mm** (milímetros): com base no tamanho físico da janela.
- **pt** (pontos): corresponde a 1/72 de uma polegada e é com base no tamanho físico da janela.

Essas medidas existem por mera questão de compatibilidade, profissionalmente usamos:

- **dp** ou **dip** (*density-independent pixel*): com base na densidade física da tela. É relativa a uma tela de 160 dpi, ou seja, 1 dp corresponde a 1 pixel em uma tela de 160 dpi, qualquer outra será automaticamente ajustado.
- **sp** (*scale-independent pixel*): funciona de modo similar ao dp, porém é recomendado para especificação do tamanho de fontes.

As **Unidades de Cores** no Android é definida no padrão RGB, ou seja, quantidades de R (Red - Vermelho), G (Green - Verde) e B (Blue - Azul). E existem as seguintes opções:

- **#RGB** trata-se da matriz simplificada de cores variando de 0 a 15 em números hexadecimais (ou seja, de '0' a 'F').
- **#ARGB** adicionamos mais um valor para a intensidade ou o Canal Alfa que fornece transparência a cor.
- **#RRGGBB** ou **#AARRGGBB** no padrão mais afinado variando de 0 a 255 ('00' a 'FF' no padrão hexadecimal)

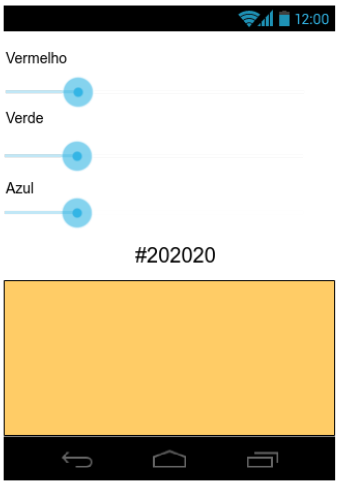

Figura 6: *Projeto Troca Tela*

Objetivo: Demonstrar como funciona o *widtget SeekBar* e como criar um aplicativo prático para entender o funcionamento das cores, no Android.

Modificar a activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Vermelho" />
  <SeekBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/corVermelho"
    android:max="255" />
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Verde" />
  <SeekBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/corVerde"
    android:max="255" />
  <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
    android:text="Azul" />
  <SeekBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/corAzul"
    android:max="255" />
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/corSelecionada"
    android:textSize="20sp"
    android:text="#000000"
    android:layout_gravity="center_horizontal"/>
  <TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#000000"
    android:id="@+id/barraCor" />
</LinearLayout>
```

Modificar a classe **MainActivity.java**

```
import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Color;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
  SeekBar corVermelho;
  SeekBar corVerde;
  SeekBar corAzul;
  private TextView corSelecionada;
  private TextView barraCor;
  private String [] hexCor = {"00","00", "00"};

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    corVermelho = findViewById(R.id.corVermelho);
    corVermelho.setOnSeekBarChangeListener(new EventoSeek((byte)0));
    corVerde = findViewById(R.id.corVerde);
    corVerde.setOnSeekBarChangeListener(new EventoSeek((byte)1));
    corAzul = findViewById(R.id.corAzul);
    corAzul.setOnSeekBarChangeListener(new EventoSeek((byte)2));
    corSelecionada = findViewById(R.id.corSelecionada);
    barraCor = findViewById(R.id.barraCor);
  }

  private class EventoSeek implements  SeekBar.OnSeekBarChangeListener {
    private byte cor;

    EventoSeek(byte cor) {
      this.cor = cor;
    }
    @Override
```

```
36      public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
37        acertaCor(i, cor);
38      }
39      @Override
40      public void onStartTrackingTouch(SeekBar seekBar) { }
41      @Override
42      public void onStopTrackingTouch(SeekBar seekBar) {}
43    }
44
45    private void acertaCor(int p, byte cor) {
46      String c = Integer.toHexString(p);
47      hexCor[cor] = (c.length() == 2?"":"0") + c;
48      mudarCor();
49    }
50
51    private void mudarCor() {
52      String corFinal = "#" + hexCor[0] + hexCor[1] + hexCor[2];
53      corSelecionada.setText(corFinal);
54      barraCor.setBackgroundColor(Color.parseColor(corFinal));
55    }
56  }
```

5 Múltiplos Layouts

Uma das melhores coisas que aconteceram com o Android foi a possibilidade de se combinar múltiplos layouts em um único layout. Isso possibilita a construção de aplicativos das mais diversas formas.

5.1 Projeto Mega Sena

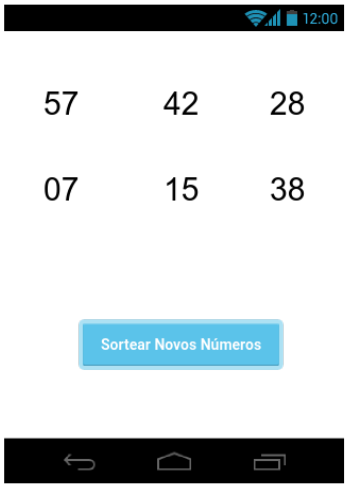

Figura 7: *Projeto Mega Sena*

Alterar o layout **activity_main**:

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity" >
  <GridView
    android:id="@+id/gridView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/button1"
    android:layout_marginTop="10dp"
    android:numColumns="3"
    android:stretchMode="columnWidth" />
  <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:onClick="onClick"
    android:text="Sortear Novos Números" />
</RelativeLayout>
```

Criar outro Layout chamado **elemento**, que será injetado no elemento GridView do Layout anterior:

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
```

```
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical" >
  <TextView
    android:id="@+id/pos"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:textSize="30sp"
    android:textColor="#F00"
    android:textStyle="bold" />
  <TextView
    android:id="@+id/numero"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:textSize="30sp"
    android:textColor="#F00"
    android:textStyle="bold" />
</LinearLayout>
```

E modificar a classe **MainActivity** para a seguinte codificação:

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import java.util.Arrays;
import java.util.Random;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

  private Random random;
  private GridView grid;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    random = new Random();
    grid = findViewById(R.id.gridView1);
    atualizarNumeros();
  }

  private void atualizarNumeros() {
    Integer[] numeros = new Integer[6];
    for (int i = 0; i < 6; i++) {
      numeros[i] = random.nextInt(60) + 1;
      for (int j = 0; j < i; j++) {
        if (numeros[i].equals(numeros[j])) {
          i--;
          break;
        }
      }
    }
```

```
    Arrays.sort(numeros);
    ArrayAdapter<Integer> adaptador = new ArrayAdapter<>(this,
    R.layout.elemento, R.id.numero, numeros);
    grid.setAdapter(adaptador);
  }

  public void onClick(View v) {
    atualizarNumeros();
    Toast.makeText(this, "Boa sorte!", Toast.LENGTH_SHORT).show();
  }
}
```

5.2 Projeto Turismo em Brasília

Para este projeto é necessário localizar algumas imagens da cidade e colocá-las na pasta **drawable**, como por exemplo:

Figura 8: *Catedral*

Figura 9: *Congresso*

Figura 10: *Museu*

Atenção os nomes das imagens devem ser todos em minúsculas, sem caracteres especiais e separado apenas por "_".

Modificar a classe **activity_main** para a seguinte codificação:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">
 <GridView
  android:id="@+id/gridView1"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:numColumns="1"
  android:layout_marginTop="10dp"
  android:stretchMode="columnWidth" />
</LinearLayout>
```

Criar um novo layout chamado **cartao**:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
```

```
 android:layout_height="match_parent">
 <TextView
  android:id="@+id/idDescricao"
  android:textStyle="italic"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_marginLeft="5dp"
  android:layout_alignParentTop="true"
  android:text="Descrição Monumento"
  android:textSize="15sp" />
 <ImageView
  android:id="@+id/idImagem"
  android:layout_alignParentRight="true"
  android:layout_width="150dp"
  android:layout_height="100dp"
  android:src="@drawable/catedral"/>
 <TextView
  android:id="@+id/idHistoria"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:layout_alignParentTop="true"
  android:layout_marginLeft="5dp"
  android:layout_marginTop="110dp"
  android:layout_marginBottom="20dp"
  android:layout_marginRight="5dp"
  android:text="História Monumento"
  android:textSize="10sp" />
</RelativeLayout>
```

Criar uma nova classe chamada **Monumento**:

```
public class Monumento {
  private String imagem;
  private String descricao;
  private String historia;

  public Monumento(String imagem, String descricao, String historia) {
   this.setImagem(imagem);
   this.setDescricao(descricao);
   this.setHistoria(historia);
  }
  public String getImagem() {
   return imagem;
  }
  public void setImagem(String imagem) {
   this.imagem = imagem;
  }
  public String getDescricao() {
   return descricao;
  }
  public void setDescricao(String descricao) {
   this.descricao = descricao;
  }
  public String getHistoria() {
   return historia;
  }
  public void setHistoria(String historia) {
```

```
    this.historia = historia;
  }
}
```

Criar uma nova classe chamada **MonumentoAdapter**:

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import java.util.List;

public class MonumentoAdapter extends ArrayAdapter {
 private List<Monumento> itens;
 private Context ctx;

 public MonumentoAdapter(Context ctx, int resource, List<Monumento> objs) {
  super(ctx, resource, objs);
  this.ctx = ctx;
  this.itens = objs;
 }

 @Override
 public int getCount() {
  return super.getCount();
 }

 @NonNull
 @Override
 public View getView(int posicao, @Nullable View convertView, @NonNull ViewGroup parent)
    {
  View v = convertView;
  if (v == null) {
   LayoutInflater inflater = (LayoutInflater)
    getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
   v = inflater.inflate(R.layout.cartao, null);
  }
  ImageView iv = v.findViewById(R.id.idImagem);
  TextView tv1 = v.findViewById(R.id.idDescricao);
  TextView tv2 = v.findViewById(R.id.idHistoria);
  Monumento item = itens.get(posicao);
  int id = ctx.getResources().
  getIdentifier("drawable/" + item.getImagem(), null, ctx.getPackageName());
  iv.setImageResource(id);
  tv1.setText(item.getDescricao());
  tv2.setText(item.getHistoria());
  return v;
 }
}
```

E modificar a classe **MainActivity** para a seguinte codificação:

```
1  import androidx.appcompat.app.AppCompatActivity;
2  import android.os.Bundle;
3  import android.widget.GridView;
4  import java.util.ArrayList;
5  import java.util.List;
6
7  public class MainActivity extends AppCompatActivity {
8
9   @Override
10  protected void onCreate(Bundle savedInstanceState) {
11   super.onCreate(savedInstanceState);
12   setContentView(R.layout.activity_main);
13   GridView grid = findViewById(R.id.gridView1);
14   MonumentoAdapter adaptador = new MonumentoAdapter(this, R.layout.cartao, getLista());
15   grid.setAdapter(adaptador);
16  }
17
18  private List<Monumento> getLista() {
19   List<Monumento> lst = new ArrayList<>();
20   lst.add(new Monumento("congresso", "Congresso Nacional",
21   "O Congresso Nacional é o titular do Poder Legislativo Federal, e o exerce por meio da
       Camara dos Deputados e do Senado Federal, cabendo-lhe legislar sobre as matérias de
       competência da União bem como fiscalizar as entidades da administração direta e
       indireta, com o auxílio do Tribunal de Contas da União."));
22   lst.add(new Monumento("itamaraty", "Palácio do Itamaraty",
23   "Palácio do Itamaraty é a sede do Ministério das Relações Exteriores e foi concebido
       para apresentar o Brasil aos visitantes estrangeiros. Por isso, foi construído
       apenas com materiais nacionais e seus salões abrigam obras apenas de artistas
       nascidos ou naturalizados brasileiros. O projeto é de Oscar Niemeyer e o paisagismo é
        de autoria de Roberto Burle Marx."));
24   lst.add(new Monumento("museu", "Museu Nacional",
25   "O Museu Nacional é integrante do Conjunto Cultural da República. É um espaço que
       insere Brasília no circuito internacional das artes e mostra o que há de melhor na
       arte brasileira. O espaço é utilizado para exposições itinerantes de artistas
       renomados e temas importantes para a sociedade, palestras, mostra de filmes,
       seminários e eventos importantes."));
26   lst.add(new Monumento("ponte",
27   "Ponte Juscelino Kubitschek", "Mais conhecida como Ponte JK ou Terceira Ponte, liga o
       Lago Sul, Paranoá e São Sebastião a parte central de Brasília, através do Eixo
       Monumental, atravessando o Lago Paranoá. Inaugurada em 15 de dezembro de 2002, a
       estrutura da ponte tem um comprimento de travessia total de 1,2 Km, tendo largura de
       24 metros com duas pistas, cada uma com 3 faixas de rolamento, duas passarelas nas
       laterais para uso de ciclistas e pedestres com 1,5 metros de largura e comprimento
       total dos vãos de 720 metros."));
28   lst.add(new Monumento("catedral", "Catedral Brasilia", "A Catedral Metropolitana Nossa
       Senhora Aparecida, mais conhecida como Catedral de Brasília, é a catedral católica
       que serve a Brasília, a capital do Brasil, sendo a sede da Arquidiocese de Brasília.
       Fica ao sul da S1, no Eixo Monumental, na região da Esplanada dos Ministérios."));
29   return lst;
30  }
31 }
```

6 JSON

JSON (JavaScript Object Notation) é uma estrutura de dados leve e amplamente utilizada para intercâmbio de dados entre diferentes plataformas e linguagens de programação. A estrutura de dados JSON é baseada em um conjunto de pares chave-valor, onde as chaves são strings e os valores podem ser strings, números, booleanos, null, arrays ou objetos.

Um objeto JSON é delimitado por chaves e consiste em uma coleção de pares chave-valor separados por vírgulas. Um array JSON é delimitado por colchetes [] e consiste em uma coleção ordenada de valores separados por vírgulas. Cada valor dentro de um objeto ou array JSON pode ser um objeto ou array JSON aninhado, permitindo a criação de estruturas de dados complexas e hierárquicas.

A estrutura de dados JSON é independente da linguagem e, portanto, é fácil de entender e usar em diferentes ambientes de programação. É amplamente usado na comunicação entre o cliente e o servidor em aplicativos web, em serviços web, em aplicativos móveis e em muitas outras aplicações.

Uma das principais vantagens do JSON é sua simplicidade e facilidade de leitura e escrita, tanto para humanos quanto para máquinas. Além disso, o JSON pode ser facilmente convertido em diferentes formatos de dados, como XML, CSV ou YAML, tornando-o uma opção flexível para o intercâmbio de dados.

6.1 Projeto Meus Amigos

Criar uma pasta "assets"na raiz do projeto Android, junto com outras pastas como "java", "res"e "AndroidManifest.xml". Selecionar "New» "Directory". E nas opções "src/main/assets". Criar um arquivo com o nome **nomes.json**:

```
[
  {
    "name": "John",
    "age": 25,
    "gender": "masculino"
  },{
    "name": "Jane",
    "age": 30,
    "gender": "feminino"
  },{
    "name": "Bob",
    "age": 40,
    "gender": "masculino"
  }
]
```

Alterar o layout **activity_main**:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">

```

```
<ListView
 android:id="@+id/listView"
 android:layout_width="match_parent"
 android:layout_height="match_parent" />
</LinearLayout>
```

E modificar a classe **MainActivity** para a seguinte codificação:

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

  private ListView listView;
  // ArrayList para carregar os dados das Pessoas
  private ArrayList<Pessoa> pessoas = new ArrayList<>();
  // Arquivo JSON da pasta "assets"
  private String json;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    listView = findViewById(R.id.listView);
    lerArquivoJSon();
    converterJSonArrayList();
    // Criar um ArrayAdapter com o ArrayList de pessoas e o ListView
    ArrayAdapter<Pessoa> adapter = new ArrayAdapter<>(this,
    android.R.layout.simple_list_item_1, pessoas);
    listView.setAdapter(adapter);
  }

  private void lerArquivoJSon() {
    try {
      InputStream inputStream = getAssets().open("nomes.json");
      int size = inputStream.available();
      byte[] buffer = new byte[size];
      inputStream.read(buffer);
      inputStream.close();
      json = new String(buffer, "UTF-8");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }

  private converterJSonArrayList() {
    // Converter o JSON em um ArrayList de objetos Pessoa
    try {
      JSONArray jsonArray = new JSONArray(json);
```

```
49      for (int i = 0; i < jsonArray.length(); i++) {
50        JSONObject jsonObject = jsonArray.getJSONObject(i);
51        String name = jsonObject.getString("name");
52        int age = jsonObject.getInt("age");
53        String gender = jsonObject.getString("gender");
54        pessoas.add(new Pessoa(name, age, gender));
55      }
56    } catch (JSONException e) {
57      e.printStackTrace();
58    }
59  }
60
61  private class Pessoa {
62    private String name;
63    private int age;
64    private String gender;
65
66    public Pessoa(String name, int age, String gender) {
67      this.name = name;
68      this.age = age;
69      this.gender = gender;
70    }
71
72    @Override
73    public String toString() {
74      return name + ", " + age + ", " + gender;
75    }
76  }
77 }
```

6.2 Contatos

Criar uma pasta "assets"na raiz do projeto Android, junto com outras pastas como "java", "res"e "AndroidManifest.xml". Selecionar "New» "Directory". E nas opções "src/main/assets". Criar um arquivo com o nome **users_list.json**:

```
1  {
2    "users": [
3      {
4        "id": "1087",
5        "name": "Abhishek Saini",
6        "email": "saini.abhishek@gmail.com",
7        "gender": "male",
8        "contact": {
9          "mobile": "+91 0000000000",
10         "home": "00 000000",
11         "office": "00 000000"
12       }
13     },{
14       "id": "1088",
15       "name": "Gourav",
16       "email": "gourav9188@gmail.com",
17       "gender": "male",
18       "contact": {
19         "mobile": "+91 0000000000",
20         "home": "00 000000",
```

```
        "office": "00 000000"
      }
    },{
      "id": "1089",
      "name": "Akshay",
      "email": "akshay@gmail.com",
      "gender": "male",
      "contact": {
        "mobile": "+91 0000000000",
        "home": "00 000000",
        "office": "00 000000"
      }
    },{
      "id": "1090",
      "name": "Fernando",
      "email": "fernndo@gmail.com",
      "gender": "male",
      "contact": {
        "mobile": "+55 5556566423",
        "home": "00 000000",
        "office": "00 000000"
      }
    }
  ]
}
```

Alterar o layout **activity_main**:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context="com.example.lerjson.MainActivity">

 <androidx.recyclerview.widget.RecyclerView
  android:id="@+id/recyclerView"
  android:layout_width="match_parent"
  android:layout_height="match_parent" />
</RelativeLayout>
```

Criar um novo layout chamado "rowlayout.xml", com o seguinte conteúdo:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.cardview.widget.CardView
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:card_view="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/card_view"
 android:layout_width="match_parent"
 android:layout_margin="5dp"
 android:layout_height="wrap_content">

 <LinearLayout
```

```
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical"
  android:padding="10dp">
  <!--
   items para uma linha do RecyclerView
  -->
  <TextView
   android:id="@+id/name"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Name"
   android:textColor="#000"
   android:textSize="20sp" />
  <TextView
   android:id="@+id/email"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="email@email.com"
   android:textColor="#000"
   android:textSize="15sp" />
  <TextView
   android:id="@+id/mobileNo"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="e9999999999"
   android:textColor="#000"
   android:textSize="15sp" />
 </LinearLayout>
</androidx.cardview.widget.CardView>
```

Criar uma classe **CustomAdapter** com a seguinte codificação:

```
import android.annotation.SuppressLint;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.ArrayList;

public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.MyViewHolder> {

  ArrayList<String> personNames;
  ArrayList<String> emailIds;
  ArrayList<String> mobileNumbers;
  Context context;

  public CustomAdapter(Context context, ArrayList<String> personNames, ArrayList<String>
    emailIds, ArrayList<String> mobileNumbers) {
    this.context = context;
    this.personNames = personNames;
    this.emailIds = emailIds;
    this.mobileNumbers = mobileNumbers;
```

```
  }

  @Override
  public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    // Inflar o item Layout
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.rowlayout,
    parent, false);
    MyViewHolder vh = new MyViewHolder(v); // passar o view para View Holder
    return vh;
  }

  @Override
  public void onBindViewHolder(@NonNull MyViewHolder holder,
    @SuppressLint("RecyclerView") int position) {
    // Colocar os dados no item
    holder.name.setText(personNames.get(position));
    holder.email.setText(emailIds.get(position));
    holder.mobileNo.setText(mobileNumbers.get(position));
    // Implementar o setOnClickListener para o item view.
    holder.itemView.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Toast.makeText(context, personNames.get(position), Toast.LENGTH_SHORT).show();
      }
    });
  }

  @Override
  public int getItemCount() {
    return personNames.size();
  }

  public class MyViewHolder extends RecyclerView.ViewHolder {
    TextView name, email, mobileNo; // iniciar os item view's

    public MyViewHolder(View itemView) {
      super(itemView);
      // Obter as referências
      name = (TextView) itemView.findViewById(R.id.name);
      email = (TextView) itemView.findViewById(R.id.email);
      mobileNo = (TextView) itemView.findViewById(R.id.mobileNo);
    }
  }
}
```

E modificar a classe **MainActivity** para a seguinte codificação:

```
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
```

```
11
12 public class MainActivity extends AppCompatActivity {
13
14   // ArrayList para person names, email Id's e mobile numbers
15   ArrayList<String> personNames = new ArrayList<>();
16   ArrayList<String> emailIds = new ArrayList<>();
17   ArrayList<String> mobileNumbers = new ArrayList<>();
18
19   @Override
20   protected void onCreate(Bundle savedInstanceState) {
21     super.onCreate(savedInstanceState);
22     setContentView(R.layout.activity_main);
23     RecyclerView recyclerView = (RecyclerView) findViewById(R.id.recyclerView);
24     LinearLayoutManager linearLayoutManager = new
       LinearLayoutManager(getApplicationContext());
25     recyclerView.setLayoutManager(linearLayoutManager);
26
27     carregarArrays();
28     CustomAdapter customAdapter = new CustomAdapter(this, personNames, emailIds,
       mobileNumbers);
29     recyclerView.setAdapter(customAdapter); // set the Adapter to RecyclerView
30   }
31
32   private void carregarArrays() {
33     try {
34       JSONObject obj = new JSONObject(loadJSONFromAsset());
35       JSONArray userArray = obj.getJSONArray("users");
36       for (int i = 0; i < userArray.length(); i++) {
37         JSONObject userDetail = userArray.getJSONObject(i);
38         personNames.add(userDetail.getString("name"));
39         emailIds.add(userDetail.getString("email"));
40         JSONObject contact = userDetail.getJSONObject("contact");
41         mobileNumbers.add(contact.getString("mobile"));
42       }
43     } catch (JSONException e) {
44       e.printStackTrace();
45     }
46   }
47
48   public String loadJSONFromAsset() {
49     String json = null;
50     try {
51       InputStream is = getAssets().open("users_list.json");
52       int size = is.available();
53       byte[] buffer = new byte[size];
54       is.read(buffer);
55       is.close();
56       json = new String(buffer, "UTF-8");
57     } catch (IOException ex) {
58       ex.printStackTrace();
59       return null;
60     }
61     return json;
62   }
63 }
```

7 JSON e Web

7.1 Projeto Pokemon

Na pasta Gradle Scripts, modificar o arquivo **build.gradle.kts (Module :app)** com a adição dos pacotes:

```
dependencies {
 implementation("com.squareup.picasso:picasso:2.8")
 implementation("com.squareup.okhttp3:okhttp:4.9.0")
 ...
}
```

ATENÇÃO: Ao terminar a modificação (e salvar) uma mensagem aparece para sincronização dos pacotes. Pressionar *Sync Now* e aguarde seu término.

No arquivo **AndroidManifest.xml** obter a permissão para Internet:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools">
 <uses-permission android:name="android.permission.INTERNET" />
 <application
  ...
</manifest>
```

Modificar o Layout principal **activity_main** para receber os dados:

```
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <ListView
  android:id="@+id/listView"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content" />
</LinearLayout>
```

Criar o Layout **pokemon** para compor a linha do *ListView*:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal"
 android:padding="10dp">
 <ImageView
  android:id="@+id/image"
  android:layout_width="50dp"
  android:layout_height="50dp"
  android:layout_marginRight="10dp" />
 <TextView
  android:id="@+id/name"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textSize="16sp" />
```

```
</LinearLayout>
```

Criar a classe **Pokemon** para representar os dados:

```
public class Pokemon {
 private int id;
 private String nome;

 public Pokemon(int id, String nome) {
  this.id = id;
  this.nome = nome;
 }

 public String getNome() {
  return nome;
 }
 public int getId() {
  return id;
 }
}
```

Criar a classe **PokemonAdapter** para tranformar os dados para o Layout:

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import com.squareup.picasso.Picasso;
import java.util.List;

public class PokemonAdapter extends ArrayAdapter<Pokemon> {
 public PokemonAdapter(Context context, List<Pokemon> pokemons) {
  super(context, 0, pokemons);
 }

 @Override
 public View getView(int position, View convertView, ViewGroup parent) {
  Pokemon pokemon = getItem(position);

  if (convertView == null) {
   convertView = LayoutInflater.from(getContext()).inflate(R.layout.pokemon, parent,
    false);
  }

  ImageView imagem = convertView.findViewById(R.id.image);
  TextView nome = convertView.findViewById(R.id.name);

  nome.setText(pokemon.getNome());
  String id = ""+ pokemon.getId();
  while (id.length() < 3) {
   id = "0" + id;
  }

```

```
    Picasso.get().load("https://raw.githubusercontent.com/fanzeyi/pokemon.json/master/thumbnails/"
    + id + ".png").into(imagem);
  return convertView;
 }
}
```

E na classe principal:

```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import okhttp3.Callback;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import okhttp3.Call;
import org.json.JSONArray;
import org.json.JSONObject;
import org.json.JSONException;
import java.io.IOException;
import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

 private OkHttpClient client = new OkHttpClient();
 private ListView listView;
 private ArrayList<Pokemon> pokemons = new ArrayList<>();

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  listView = findViewById(R.id.listView);

  // Construindo a requisição
  Request request = new Request.Builder()
    .url("https://raw.githubusercontent.com/fanzeyi/pokemon.json/master/pokedex.json")
    .build();

  // Enviando a requisição e recebendo a resposta
  client.newCall(request).enqueue(new Callback() {
   @Override
   public void onFailure(Call call, IOException e) {
    // O que fazer em caso de falha
    e.printStackTrace();
   }

   @Override
   public void onResponse(Call call, Response response) throws IOException {
    // O que fazer em caso de resposta?
    if (response.isSuccessful()) {
     final String myResponse = response.body().string();

     MainActivity.this.runOnUiThread(new Runnable() {
      @Override
      public void run() {
```

```
50        try {
51         // Convertendo a resposta em um objeto JSON
52         // JSONObject json = new JSONObject(myResponse);
53         JSONArray pokes = new JSONArray(myResponse);
54         for (int i = 0; i < pokes.length(); i++) {
55          JSONObject poke = pokes.getJSONObject(i);
56          int id = poke.getInt("id");
57          JSONObject name = poke.getJSONObject("name");
58          pokemons.add(new Pokemon(id, name.getString("english")));
59         }
60         // Ler os dados do seu JSON, e atualizar a UI
61         ArrayAdapter<Pokemon> adapter = new PokemonAdapter(MainActivity.this, pokemons);
62         listView.setAdapter(adapter);
63        } catch (JSONException e) {
64         e.printStackTrace();
65        }
66       }
67      });
68     }
69    }
70   });
71  }
72 }
```

8 Parser de Outros Arquivos de Transporte

Parser é uma técnica utilizada pela maioria dos navegadores para acessar e manipular informações de arquivos que possuem formatos definidos. Além do JSON, temos outros formatos de arquivos populares destinado ao transporte de informações. É uma técnica é excelente para obter conteúdo dinâmico para nossas aplicações Android já que ambas notações possuem uma formatação bastante leve de troca de dados. Fácil de ler e escrever e para os desenvolvedores também é fácil de se implementar um programa de leitura.

8.1 Projeto Lista de Empregados com XML

Criar uma pasta "assets"na raiz do projeto Android, junto com outras pastas como "java", "res"e "AndroidManifest.xml". Selecionar "New≫ "Directory". E nas opções "src/main/assets". Criar um arquivo com o nome **funcionarios.xml**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<registro>
    <colaboradores>
        <setor>Matriz</setor>
        <empregado>
            <nome>João da Silva</nome>
            <salario>2800</salario>
            <cargo>Presidente</cargo>
        </empregado>
        <empregado>
            <nome>Maria da Silva</nome>
            <salario>4800</salario>
            <cargo>Sócio</cargo>
        </empregado>
        <empregado>
            <nome>Jose da Silva</nome>
            <salario>3200</salario>
            <cargo>Motorista</cargo>
        </empregado>
        <empregado>
            <nome>Marli da Silva</nome>
            <salario>7200</salario>
            <cargo>Cientista</cargo>
        </empregado>
    </colaboradores>
    <colaboradores>
        <setor>Filial</setor>
        <empregado>
            <nome>Marcelo da Silva</nome>
            <salario>3200</salario>
            <cargo>Vice-Presidente</cargo>
        </empregado>
    </colaboradores>
</registro>
```

Alterar o layout **activity_main**:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
```

```
 android:layout_height="match_parent"
 tools:context=".MainActivity">
 <TextView
  android:id="@+id/empregado"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Empregados da Empresa XYZ Silva" />
</RelativeLayout>
```

Modificar a classe **MainActivity.java**:

```
public class MainActivity extends AppCompatActivity {

  private final String TAG = "NOME";
  private TextView tv1;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tv1 = findViewById(R.id.empregado);
    carregarInformacoes();
  }

  private void carregarInformacoes() {
    try {
      InputStream is = getAssets().open("funcionarios.xml");
      DocumentBuilderFactory fabrica = DocumentBuilderFactory.newInstance();
      DocumentBuilder documento = fabrica.newDocumentBuilder();
      Document doc = documento.parse(is);
      Element elemento = doc.getDocumentElement();
      elemento.normalize();
      NodeList lista = doc.getElementsByTagName("empregado");
      for (int i = 0; i < lista.getLength(); i++) {
        Node node = lista.item(i);
        if (node.getNodeType() == Node.ELEMENT_NODE) {
          Element elemento2 = (Element) node;
          tv1.setText(tv1.getText()+"\n\nNome: " + getValue("nome", elemento2));
          tv1.setText(tv1.getText()+"\nSalario: " + getValue("salario", elemento2));
          tv1.setText(tv1.getText()+"\nCargo: " + getValue("cargo", elemento2));
        }
      }
    } catch (Exception e) {
      Log.e(TAG, "Erro: " + e.getMessage());
    }
  }

  private String getValue(String tag, Element el) {
    NodeList lista = el.getElementsByTagName(tag).item(0).getChildNodes();
    Node node = lista.item(0);
    return node.getNodeValue();
  }
}
```

8.2 Meus Contatos Tinder com CSV

Arquivos CSV (*Comma-Separated Values*) são um tipo de arquivo de texto que armazena dados em formato tabular, onde cada linha representa um registro e cada coluna representa um campo de informação. Os dados são separados por vírgulas ou outros caracteres delimitadores, o que permite a fácil leitura e gravação em programas de planilha eletrônica ou bancos de dados. Por serem arquivos de texto simples, os arquivos CSV podem ser facilmente manipulados por programas de processamento de texto ou linguagens de programação, tornando-os uma opção popular para a importação e exportação de dados entre diferentes aplicativos.

Alterar o layout **activity_main** com a codificação idêntica à do projeto anterior e adicionar uma permissão de acesso a Internet no arquivo **AndroidManifest.xml**.

Criar uma pasta "assets"na raiz do projeto Android, junto com outras pastas como "java", "res"e "AndroidManifest.xml". Selecionar "New≫ "Directory". E nas opções "src/main/assets". Criar um arquivo com o nome **pessoas.csv**:

```
Nome,Idade,Cidade
Carla,45,São Paulo
Maria,38,Rio de Janeiro
Joana,49,Belo Horizonte
```

Alterar o layout **activity_main**:

```
<LinearLayout
 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:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 tools:context=".MainActivity">
 <TextView
  android:text="Dados do Tinder"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:textSize="20sp"
  android:padding="16dp"/>
 <ListView
  android:id="@+id/list_view"
  android:layout_width="match_parent"
  android:layout_height="match_parent"/>
</LinearLayout>
```

E a classe **MainActivity.java** com a seguinte codificação:

```
public class MainActivity extends AppCompatActivity {

  private ListView listView;
  private List<String[]> dataList = new ArrayList<>();

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

```
    listView = findViewById(R.id.list_view);
    readCsvFile();
    setupListView();
  }

  private void readCsvFile() {
    InputStream inputStream = null;
    BufferedReader reader = null;
    try {
      inputStream = getAssets().open("pessoas.csv");
      reader = new BufferedReader(new InputStreamReader(inputStream));
      String line;
      while ((line = reader.readLine()) != null) {
        String[] row = line.split(",");
        dataList.add(row);
      }
    } catch (IOException e) {
      Log.e("MainActivity", "Error reading CSV file: " + e);
    } finally {
      if (reader != null) {
        try {
          reader.close();
        } catch (IOException e) {
          Log.e("MainActivity", "Error closing reader: " + e);
        }
      }
    }
  }

  private void setupListView() {
    List<String> items = new ArrayList<>();
    for (String[] row : dataList) {
      String item = row[0] + ", " + row[1] + ", " + row[2];
      items.add(item);
    }
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
    android.R.layout.simple_list_item_1, items);
    listView.setAdapter(adapter);
  }
}
```

9 Jogos

Jogos são divertidos e algo que todo mundo deveria fazer para praticar sua lógica, existem vários modelos de jogos para celulares, mas aqueles considerados mais simples são os que fazem sucesso.

DESAFIO. Neste projeto foi usado uma simples STRING com uma única palavra escondida, teste seus conhecimentos trocando esta para uma lista de palavras de modo que cada vez que o jogo reinicia uma dessas palavras possa ser usada no jogo.

9.1 Projeto Forca

Criar uma pasta "assets"na raiz do projeto Android, junto com outras pastas como "java", "res"e "AndroidManifest.xml". Selecionar "New≫ "Directory". E nas opções "src/main/assets". Adicionar os arquivos imagem que correspondem ao movimento da forca, indo da imagem "Forca6.png" até "Forca0.png".

Seja criativo neste ponto, evite de pegar imagens prontas na Web, use um programa de geração de imagens (como o Gimp ou MS-Paint) para criar suas próprias imagens.

Adicionar ao arquivo **strings.xml**:

```
<resources>
 <string name="app_name">ForcaT1</string>
 <string name="letras_erradas">Letras Erradas:</string>
 <string name="imagem_da_forca">Imagem da Forca</string>
 <string name="digite_uma_letra">Digite uma letra</string>
 <string name="confirmar">Confirmar</string>
 <string name="tracos">_______</string>
</resources>
```

Alterar o layout **activity_main**:

```
<?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:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:textSize="20sp"
  android:text="@string/letras_erradas"
  android:id="@+id/letrasErradas" />
 <ImageView
  android:layout_gravity="center"
  android:layout_width="300dp"
  android:layout_height="500dp"
  android:id="@+id/minhaforca"
  android:contentDescription="@string/imagem_da_forca" />
 <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/tracos"
```

```
  android:textSize="25sp"
  android:letterSpacing="0.3"
  android:id="@+id/tracos"
  android:layout_gravity="end"
  android:layout_marginEnd="30dp" />
 <LinearLayout
  android:layout_marginTop="20dp"
  android:autofillHints="true"
  android:gravity="center_horizontal"
  android:orientation="horizontal"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <EditText
   android:id="@+id/letraDig"
   android:layout_width="140dp"
   android:layout_height="wrap_content"
   android:layout_marginEnd="20dp"
   android:hint="@string/digite_uma_letra"
   android:inputType="text"
   android:minHeight="48dp" />
  <Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="@string/confirmar"
   android:id="@+id/btConfirmar" />
 </LinearLayout>
</LinearLayout>
```

E modificar a classe **MainActivity.java**:

```
public class MainActivity extends AppCompatActivity {

 private ImageView imageForca;
 private EditText letraDig;
 private TextView letrasErradas;
 private TextView palavraEscond;

 private String palavra = "PALAVRA";
 private int posForca;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  imageForca = findViewById(R.id.minhaforca);
  letraDig = findViewById(R.id.letraDig);
  letrasErradas = findViewById(R.id.letrasErradas);
  palavraEscond = findViewById(R.id.tracos);
  // Limita letras a 1 caracter
  letraDig.setFilters(new InputFilter[]{
   new InputFilter.LengthFilter(1)});
  letraDig.addTextChangedListener(new TextWatcher() {
   @Override
   public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {
   }

   @Override
```

```
28    public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {
29    }
30
31    @Override
32    public void afterTextChanged(Editable editable) {
33     if (editable.length() == 1) {
34      // Fecha teclado
35      InputMethodManager imm = (InputMethodManager)
36      getSystemService(Context.INPUT_METHOD_SERVICE);
37      imm.hideSoftInputFromWindow(letraDig.getWindowToken(), 0);
38     }
39    }
40   });
41   findViewById(R.id.btConfirmar).
42   setOnClickListener(view ->
       verificarLetraDigitada(letraDig.getText().toString().toUpperCase()));
43   iniciar();
44  }
45
46  private void verificarLetraDigitada(String let) {
47   StringBuilder mnt = new StringBuilder("");
48   boolean acertei = false;
49   for (int i = 0; i < palavra.length(); i += 1) {
50    if (palavra.charAt(i) == let.charAt(0)) {
51     mnt.append(let);
52     acertei = true;
53    } else {
54     mnt.append(palavraEscond.getText().toString().charAt(i));
55    }
56   }
57   palavraEscond.setText(mnt.toString());
58   if (!acertei) {
59    letrasErradas.append(let);
60    posForca -= 1;
61    carregarImagem("Forca" + posForca + ".png");
62   } else {
63    boolean terminei = true;
64    for (int i = 0; i < palavraEscond.length(); i += 1) {
65     if (palavraEscond.getText().toString().charAt(i) == '_') {
66      terminei = false;
67      break;
68     }
69    }
70    if (terminei) {
71     fimDoJogo(true);
72    }
73   }
74   if (posForca == 0) {
75    fimDoJogo(false);
76   }
77  }
78
79  private void fimDoJogo(boolean tipo) {
80   AlertDialog.Builder construtorAlertas = new AlertDialog.Builder(this);
81   if (tipo) {
82    construtorAlertas.setTitle("Parabéns você GANHOU!");
83   } else {
84    construtorAlertas.setTitle("Você foi ENFORCADO!");
```

```
85   }
86   construtorAlertas.setMessage("Deseja continuar?");
87   construtorAlertas.setPositiveButton("Sim", new DialogInterface.OnClickListener() {
88    @Override
89    public void onClick(DialogInterface dialogInterface, int i) {
90     Toast.makeText(getApplicationContext(), "Vamos novamente então",
       Toast.LENGTH_SHORT).show();
91     iniciar();
92    }
93   });
94   construtorAlertas.setNegativeButton("Não", new DialogInterface.OnClickListener() {
95    @Override
96    public void onClick(DialogInterface dialogInterface, int i) {
97     Toast.makeText(getApplicationContext(), "Até uma próxima vez",
       Toast.LENGTH_SHORT).show();
98     finish();
99    }
100  });
101  // Criar e exibir o AlertDialog
102  AlertDialog alertDialog = construtorAlertas.create();
103  alertDialog.show();
104 }
105
106 private void iniciar() {
107  posForca = 6;
108  carregarImagem("Forca6.png");
109  montarPalavra();
110 }
111
112 private void montarPalavra() {
113  StringBuilder mnt = new StringBuilder("");
114  for (int i = 0; i < palavra.length(); i++) {
115   mnt.append("_");
116  }
117  palavraEscond.setText(mnt.toString());
118 }
119
120 private void carregarImagem(String image) {
121  try {
122   InputStream is = getAssets().open(image);
123   Bitmap bmp = BitmapFactory.decodeStream(is);
124   imageForca.setImageBitmap(bmp);
125  } catch (IOException e) {
126   Log.e("ERRO", e.getMessage());
127  }
128 }
129 }
```

10 Gráficos

A importância da apresentação visual de dados não pode ser subestimada e existem diversas formas de se fazê-lo. No entanto, uma das maneiras mais eficazes e atraentes é através do uso de gráficos. Especialmente no contexto do desenvolvimento de aplicativos para Android, onde a apresentação de informações em forma clara e intuitiva é fundamental para a experiência do usuário, os gráficos podem ser uma ferramenta valiosa para tornar os dados mais acessíveis e compreensíveis.

Além de fornecer informações importantes, os gráficos também podem ser uma forma de tornar a interface do usuário mais atraente e engajadora. Portanto, a utilização de gráficos é uma estratégia importante a ser considerada no design de aplicativos para Android..

10.1 Gráficos Básicos

Primeiro precisamos adicionar o repositório aonde se encontra a biblioteca, no arquivo settings.gradle, adicione aos repositórios:

```
1  maven { url 'https://jitpack.io' }
```

Para este primeiro exemplo usaremos a biblioteca MP Android Chart que pode ser obtida adicionando a seguinte dependência no arquivo build.gradle:

```
1  implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
```

Definir os recursos de cor para o XML de valor **color.xml**:

```
1 <color name="purple_200">#FFBB86FC</color>
2 <color name="purple_500">#FF6200EE</color>
3 <color name="purple_700">#FF3700B3</color>
4 <color name="purple_900">#FF25076A</color>
```

Verificar se o arquivo de **themes.xml** está correto, e herda da classe que apresenta a barra de menu:

```
1 <style name="Base.Theme.GraphBasic"
     parent="Theme.MaterialComponents.DayNight.DarkActionBar">
2   <item name="colorPrimary">@color/purple_500</item>
3   <item name="colorPrimaryVariant">@color/purple_700</item>
4   ...
5 </style>
```

Criar um menu básico de modo que possamos seguir para um segundo layout, para isso criar uma pasta em **res** chamando-a de menu e nesta um arquivo chamado **menugraph.xml** e adicinar o seguinte conteúdo:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu
3  xmlns:android="http://schemas.android.com/apk/res/android"
4  xmlns:app="http://schemas.android.com/apk/res-auto">
5  <item
6    android:id="@+id/maisGraph"
7    android:title="Mais Gráficos"
8    app:showAsAction="never" />
```

```
</menu>
```

Alterar o layout **activity_main** para obtermos um exemplo do gráfico de linhas e barras:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:rowCount="2"
 android:columnCount="1">
 <com.github.mikephil.charting.charts.LineChart
  android:id="@+id/line_chart"
  android:layout_rowWeight="1"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" />
 <com.github.mikephil.charting.charts.BarChart
  android:id="@+id/bar_chart"
  android:layout_rowWeight="2"
  android:layout_width="match_parent"
  android:layout_height="wrap_content" />
</GridLayout>
```

Criar mais uma atividade no modelo *Empty View Activities* com o layout chamada **Pagina2**. Modificar o layout **activity_pagina2** para mostrarmos mais exemplos de um gráfico de Pizza e outro modelo de linha:

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:rowCount="2"
 android:columnCount="1">
 <com.github.mikephil.charting.charts.PieChart
  android:id="@+id/pie_chart"
  android:layout_rowWeight="1"
  android:layout_width="match_parent"
  android:layout_height="300dp"/>
 <com.github.mikephil.charting.charts.LineChart
  android:id="@+id/line_chart"
  android:layout_rowWeight="1"
  android:layout_width="match_parent"
  android:layout_height="300dp" />
</GridLayout>
```

Na nossa atividade principal:

```
public class MainActivity extends AppCompatActivity {

  private LineChart lineChart;
  private BarChart barChart;

  @Override
```

```
7   protected void onCreate(Bundle savedInstanceState) {
8     super.onCreate(savedInstanceState);
9     setContentView(R.layout.activity_main);
10
11    // Gráfico 1
12    lineChart = findViewById(R.id.line_chart);
13    carregarLine();
14
15    // Gráfico 2
16    barChart = findViewById(R.id.bar_chart);
17    carregarBar();
18  }
19
20  @Override
21  public boolean onCreateOptionsMenu(Menu menu) {
22    getMenuInflater().inflate(R.menu.menugraph, menu);
23    return true;
24  }
25
26  @Override
27  public boolean onOptionsItemSelected(@NonNull MenuItem item) {
28    if (item.getItemId() == R.id.maisGraph) {
29      startActivity(new Intent(this, Pagina2.class));
30      return true;
31    }
32    return super.onOptionsItemSelected(item);
33  }
34
35  private void carregarLine() {
36    ArrayList<Entry> entries = new ArrayList<>();
37    entries.add(new Entry(0, 4));
38    entries.add(new Entry(1, 2));
39    entries.add(new Entry(2, 3));
40    entries.add(new Entry(3, 5));
41    entries.add(new Entry(4, 6));
42    LineDataSet lineDataSet =
43      new LineDataSet(entries, "Dados de Umidade");
44    lineDataSet.setColor(
45      ContextCompat.getColor(this, R.color.purple_200));
46    lineDataSet.setValueTextColor(
47      ContextCompat.getColor(this, R.color.purple_700));
48    LineData lineData = new LineData(lineDataSet);
49    lineChart.setData(lineData);
50    XAxis xAxis = lineChart.getXAxis();
51    xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
52    YAxis yAxisLeft = lineChart.getAxisLeft();
53    yAxisLeft.setAxisMinimum(0);
54    YAxis yAxisRight = lineChart.getAxisRight();
55    yAxisRight.setEnabled(false);
56    lineChart.invalidate();
57  }
58
59  private void carregarBar() {
60    ArrayList<BarEntry> entries = new ArrayList<>();
61    entries.add(new BarEntry(1, 50f));
62    entries.add(new BarEntry(2, 80f));
63    entries.add(new BarEntry(3, 60f));
64    entries.add(new BarEntry(4, 70f));
```

```
    entries.add(new BarEntry(5, 30f));

    BarDataSet dataSet = new BarDataSet(
      entries, "Dados da Temperatura");
    dataSet.setColor(
      ContextCompat.getColor(this, R.color.purple_200));

    BarData barData = new BarData(dataSet);
    barChart.setData(barData);

    barChart.setFitBars(true);
    barChart.getDescription().setEnabled(false);
    barChart.animateY(1000);
    barChart.setTouchEnabled(false);
  }
}
```

E modificar a atividade para a segunda parte denominada **Pagina2.java**:

```
public class Pagina2 extends AppCompatActivity {

  private PieChart pieChart;
  private LineChart mLineChart;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_pagina2);

    pieChart = findViewById(R.id.pie_chart);
    carregarPie();
    mLineChart = findViewById(R.id.line_chart);
    carregarLine();
  }

  private void carregarPie() {
    ArrayList<PieEntry> entries = new ArrayList<>();
    entries.add(new PieEntry(20f, "Maçã"));
    entries.add(new PieEntry(30f, "Banana"));
    entries.add(new PieEntry(40f, "Abacaxi"));
    entries.add(new PieEntry(10f, "Laranja"));

    // Definir a cor das fatias
    ArrayList<Integer> colors = new ArrayList<>();
    colors.add(ContextCompat.getColor(this, R.color.purple_200));
    colors.add(ContextCompat.getColor(this, R.color.purple_500));
    colors.add(ContextCompat.getColor(this, R.color.purple_700));
    colors.add(ContextCompat.getColor(this, R.color.purple_900));

    // Conjunto de dados do gráfico de pizza
    PieDataSet dataSet = new PieDataSet(entries, "Frutas");
    dataSet.setColors(colors);

    // Conjunto de dados ao gráfico
    PieData pieData = new PieData(dataSet);
    pieChart.setData(pieData);

```

```
    // Configurações adicionais do gráfico
    pieChart.getDescription().setEnabled(false);
    pieChart.setEntryLabelColor(Color.WHITE);
    pieChart.setEntryLabelTextSize(14f);
    pieChart.setUsePercentValues(true);
  }

  private void carregarLine() {
    // initialize chart
    mLineChart.setDragEnabled(true);
    mLineChart.setScaleEnabled(false);
    mLineChart.setDrawGridBackground(false);
    mLineChart.getDescription().setEnabled(false);

    setData();

    mLineChart.getLegend().setEnabled(false);
    mLineChart.getXAxis().setPosition(XAxis.XAxisPosition.BOTTOM);
    mLineChart.getXAxis().setTextColor(Color.WHITE);
    mLineChart.getAxisLeft().setTextColor(Color.WHITE);
    mLineChart.getAxisRight().setEnabled(false);

    mLineChart.animateX(1500);
  }

  private void setData() {
    ArrayList<Entry> values = new ArrayList<>();

    // add data to chart
    values.add(new Entry(0, 30));
    values.add(new Entry(1, 60));
    values.add(new Entry(2, 45));
    values.add(new Entry(3, 70));
    values.add(new Entry(4, 50));
    values.add(new Entry(5, 80));
    values.add(new Entry(6, 60));
    values.add(new Entry(7, 90));

    LineDataSet set1;
    if (mLineChart.getData() != null &&
    mLineChart.getData().getDataSetCount() > 0) {
      set1 = (LineDataSet) mLineChart.getData().getDataSetByIndex(0);
      set1.setValues(values);
      mLineChart.getData().notifyDataChanged();
      mLineChart.notifyDataSetChanged();
    } else {
      set1 = new LineDataSet(values, "Dados Linear da Temperatura");
      set1.setDrawIcons(false);
      set1.setColor(ContextCompat.getColor(this, R.color.purple_700));
      set1.setLineWidth(2f);
      set1.setDrawCircleHole(false);
      set1.setDrawCircles(false);
      set1.setFormLineWidth(1f);
      set1.setFormSize(15.f);
      set1.setDrawFilled(true);
      set1.setFillAlpha(50);
      set1.setFillColor(ContextCompat.getColor(this, R.color.purple_200));
      set1.setMode(LineDataSet.Mode.CUBIC_BEZIER);
```

```
97
98        ArrayList<ILineDataSet> dataSets = new ArrayList<>();
99        dataSets.add(set1);
100
101       LineData data = new LineData(dataSets);
102       mLineChart.setData(data);
103     }
104   }
105 }
```