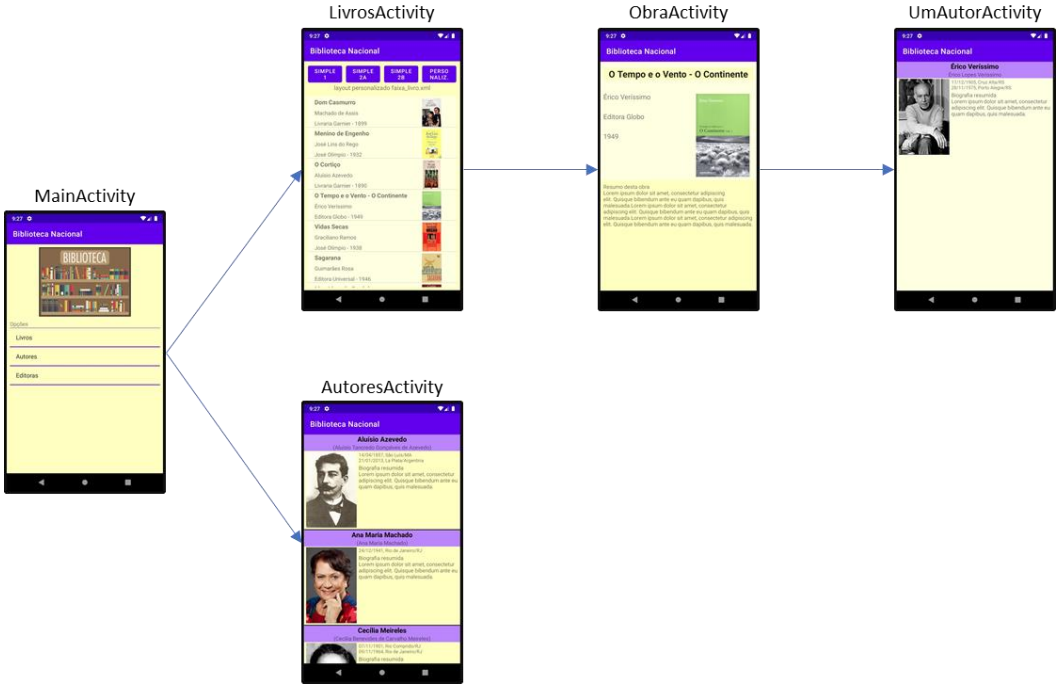
Ficha do Aplicativo

App08 – Biblioteca Nacional
Resumo Este é um aplicativo multitelas organizado de modo a ter uma tela inicial de entrada, uma tela intermediária de opções e uma tela final de detalhes. Trata-se de uma forma de organização muito usada pelos aplicativos em geral.
Objetivos de Aprendizagem 1. Praticar o uso de Intents para navegar entre as telas 2. Trabalhar novamente com o componente ImageView e com “drawables” 3. Utilizar Listeners e sua instanciação usando objetos anônimos embutidos 4. Conhecer o componente ListView 5. Usar Adapters para preencher o ListView
Dinâmica do Aplicativo Este aplicativo conterá e exibirá um cadastro de Livros e Autores. Um terceiro item – Editoras – estará no menu de entrada, porém não implementado. Na tela de entrada haverá um menu com as três opções citadas que darão acesso à tela intermediária exibindo a lista de itens cadastrados. A partir dessa tela intermediária chega-se à tela de detalhe que exibirá todas as informações do item clicado pelo usuário.

Lista de Activities do Aplicativo

Nome	Layout
MainActivity.java	activity_main.xml
LivrosActivity.java	activity_livros.xml
ObraActivity.java	activity_obra.xml
UmAutorActivity.java	activity_um_autor.xml
AutoresActivity.java	activity_autores.xml



Resources

strings.xml

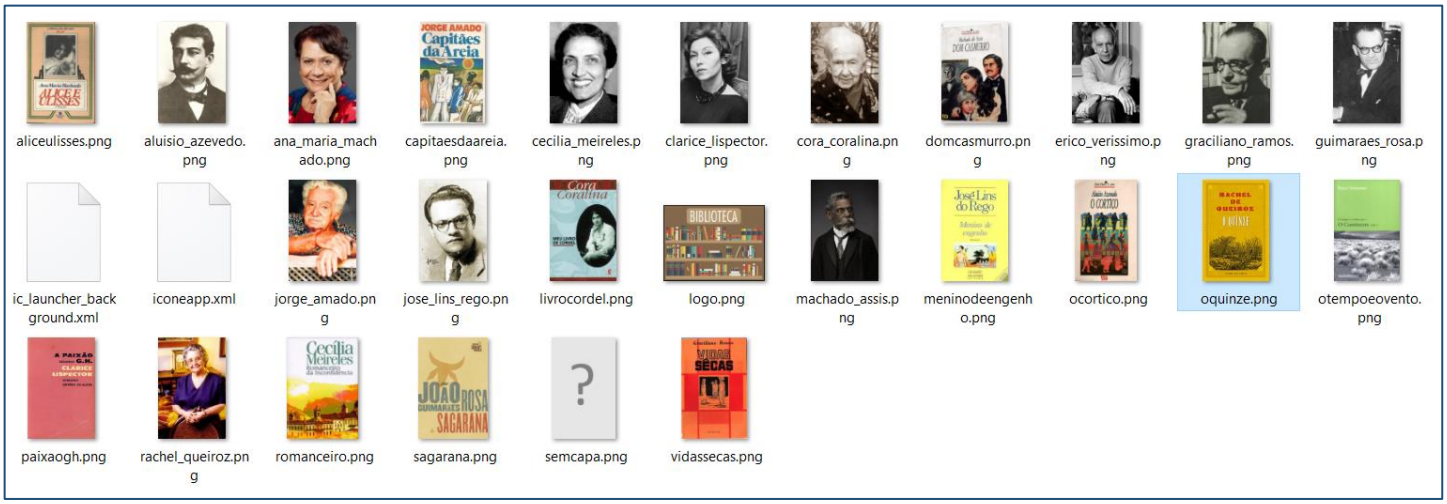
```
<resources>
    <string name="app_name">Biblioteca Nacional</string>
    <string name="main_opc">Opções</string>
    <string name="strlogo">Logotipo do aplicativo</string>
    <string-array name="opcoes">
        <item>Livros</item>
        <item>Autores</item>
        <item>Editoras</item>
    </string-array>
</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="corFundo">#FFFFFFC0</color>
    <color name="corFundoLight">#FFFFFFE1</color>
    <color name="corDivisor">#FFA996CA</color>
</resources>
```

Drawables deste Aplicativo

Este aplicativo contém um grande número de imagens que são fotos de autores e capas de livros, um logotipo e um ícone no formato .XML, conforme mostrado a seguir. Na minha página vocês podem fazer o download desses recursos.



Layouts não associados a activities

Neste aplicativo serão usados dois layouts que não estão diretamente associados a uma activity, nos moldes como vimos até agora. Esses layouts serão usados na personalização das faixas que serão exibidas através do componente ListView. Aqui vamos apenas listar esses dois layouts e no final deste documento ambos estão detalhados. Procure a seção Personalização de Adaptadores – Parte 2. Um desses layouts também será reutilizado em outro layout – cláusula <inlcude> - e isso está descrito na seção sobre o layout de UmAutorActivity.

faixa_livro.xml	Usado para personalizar o ListView contido em LivrosActivity
faixa_autor.xml	Usado para personalizar o ListView contido em AutoresActivity e também incluído no layout de UmAutorActivity

Arquivos Java Puro (não associados a activities)

Este aplicativo contém quatro arquivos Java não associados a Activity. Eles estão descritos nesta tabela e serão comentados adiante neste documento.

Autor.java	Define a classe que contém os dados dos Autores que estarão presentes no aplicativo. Mostrada logo abaixo
AutoresAdapter.java	Classe onde é implementado o adaptador que fará a ligação entre a classe Autor e o ListView Será mostrada mais adiante.
Livro.java	Define a classe que contém os dados dos Livros que estarão presentes no aplicativo. Mostrada logo abaixo
LivrosAdapter.java	Classe onde é implementado o adaptador que fará a ligação entre a classe Livro e o ListView Será mostrada mais adiante.

Classe Autor (arquivo Autor.java)

```
package com.ilp506.bibliotecanacional;

public class Autor {
    private String nome;
    private String nomeRegistro;
    private String dataNasc;
    private String localNasc;
    private String dataMorte;
    private String localMorte;
    private String foto;

    public Autor(String nome, String nomeRegistro, String dataNasc, String localNasc, String dataMorte, String
localMorte, String foto) {
        this.nome = nome;
        this.nomeRegistro = nomeRegistro;
        this.dataNasc = dataNasc;
        this.localNasc = localNasc;
        this.dataMorte = dataMorte;
        this.localMorte = localMorte;
        this.foto = foto;
    }

    public String getNome() {
        return nome;
    }

    public String getNomeRegistro() {
        return nomeRegistro;
    }

    public String getDataNasc() {
        return dataNasc;
    }

    public String getDataMorte() {
        return dataMorte;
    }

    public String getLocalNasc() {
        return localNasc;
    }

    public String getLocalMorte() {
        return localMorte;
    }

    public String getVida() {
        StringBuilder s = new StringBuilder();
        s.append(dataNasc);
        s.append(", ");
        s.append(localNasc);
        if (!dataMorte.equals("")) {
            s.append("\n");
            s.append(dataMorte);
            s.append(", ");
            s.append(localMorte);
        }
        return String.valueOf(s);
```

```
    }

    public String getFoto() {
        return foto;
    }

    public String toString() {
        return nome;
    }

    public static final Autor[] autores = {
            new Autor("Aluísio Azevedo", "Aluísio Tancredo Gonçalves de Azevedo", "14/04/1857", "São Luís/MA", "21/01/2013", "La
Plata/Argentina", "aluisio_azevedo"),
            new Autor("Ana Maria Machado", "Ana Maria Machado", "24/12/1941", "Rio de Janeiro/RJ", "", "", "ana_maria_machado"),
            new Autor("Cecília Meireles", "Cecília Benevides de Carvalho Meireles", "07/11/1901", "Rio Comprido/RJ", "09/11/1964", "Rio
de Janeiro/RJ", "cecilia_meireles"),
            new Autor("Clarice Lispector", "Chaya Pinkhasovna Lispector", "Chechelnyk/Ucrânia", "10/12/1920", "09/12/1977", "Rio de
Janeiro/RJ", "clarice_lispector"),
            new Autor("Cora Coralina", "Anna Lins dos Guimarães Peixoto Bretas", "20/08/1889", "Cidade de Goiás/GO", "10/04/1985",
"Goiânia/GO", "cora_coralina"),
            new Autor("Érico Veríssimo", "Érico Lopes Veríssimo", "17/12/1905", "Cruz Alta/RS", "28/11/1975", "Porto Alegre/RS",
"erico_verissimo"),
            new Autor("Graciliano Ramos", "Graciliano Ramos de Oliveira", "27/10/1892", "Quebrangulo/AL", "20/03/1953", "Rio de
Janeiro/RJ", "graciliano_ramos"),
            new Autor("Guimarães Rosa", "João Guimarães Rosa", "27/06/1908", "Cordisburgo/MG", "19/11/1967", "Rio de Janeiro/RJ",
"guimaraes_rosa"),
            new Autor("Jorge Amado", "Jorge Leal Amado de Faria", "10/08/1912", "Itabuna/BA", "06/08/2001", "Salvador/BA",
"jorge_amado"),
            new Autor("José Lins do Rego", "José Lins do Rego Cavalcanti", "03/06/1901", "Pilar/PB", "12/09/1957", "Rio de Janeiro/RJ",
"jose_lins_rego"),
            new Autor("Machado de Assis", "Joaquim Maria Machado de Assis", "21/06/1839", "Rio de Janeiro/RJ", "29/09/1908", "Rio de
Janeiro/RJ", "machado_assis"),
            new Autor("Rachel de Queiroz", "Rachel de Queiroz", "17/11/1910", "Fortaleza/CE", "04/11/2003", "Rio de Janeiro/RJ",
"rachel_queiroz")
    };
}
```

Classe Livro (arquivo Livro.java)

```
public class Livro {
    private String nome;
    private int autor;
    private String editora;
    private int anoPub;
    private int imgResId;

    private Livro(String nome, int autor, String editora, int anoPub, int imgResId) {
        this.nome = nome;
        this.autor = autor;
        this.editora = editora;
        this.anoPub = anoPub;
        this.imgResId = imgResId;
    }

    public String toString() { return nome; }

    public String getNome() { return nome; }

    public int getAutor() { return autor; }

    public String getEditora() { return editora; }

    public int getAnoPub() { return anoPub; }

    public int getImgResId() { return imgResId;}

    public static final Livro[] livros = {
            new Livro("Dom Casmurro", 10, "Livraria Garnier", 1899, R.drawable.domcasmurro),
            new Livro("Menino de Engenho", 9, "José Olímpio", 1932, R.drawable.meninodeengenho),
            new Livro("O Cortiço", 0, "Livraria Garnier", 1890, R.drawable.ocortico),
            new Livro("O Tempo e o Vento - O Continente", 5, "Editora Globo", 1949, R.drawable.otempoeovento),
            new Livro("Vidas Secas", 6, "José Olímpio", 1938, R.drawable.vidassecas),
            new Livro("Sagarana", 7, "Editora Universal", 1946, R.drawable.sagarana),
            new Livro("Meu Livro de Cordel", 4, "Global", 1976, R.drawable.livrocordel),
            new Livro("Alice e Ulisses", 1, "Francisco Alves", 1984, R.drawable.aliceulisses),
            new Livro("Romanceiro da Inconfidência", 2, "Livros d'Portugal", 1953, R.drawable.romanceiro),
            new Livro("A Paixão Segundo G.H.", 3, "Rocco", 1964, R.drawable.paixaogh),
            new Livro("O Quinze", 11, "José Olímpio", 1930, R.drawable.oquinze),
            new Livro("Capitães da Areia", 8, "José Olímpio", 1933, R.drawable.capitaesdaareia)
    };
}
```

Detalhamento das Activities

Layout: activity_main.xml

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

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="match_parent"
        android:layout_height="182dp"
        android:layout_marginTop="8dp"
        android:contentDescription="@string/strlogo"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="4dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="4dp"
        android:text="@string/main_opc"
        android:textSize="14sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView2" />

    <ListView
        android:id="@+id/lvOpcoes"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginStart="4dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="4dp"
        android:divider="@color/purple_500"
        android:dividerHeight="2dp"
        android:entries="@array/opcoes"
        android:scrollbars="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <android.view.View
        android:layout_width="0dp"
        android:layout_height="2dp"
        android:layout_marginStart="4dp"
        android:layout_marginEnd="4dp"
        android:background="@color/purple_500"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/lvOpcoes" />

</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 {

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

        setLvOpcoesOnItemClick();
    }

    private void setLvOpcoesOnItemClick() {
        ListView lvOpcoes = findViewById(R.id.lvOpcoes);
        lvOpcoes.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {
                if (position == 0) {
                    Intent intent = new Intent(MainActivity.this, LivrosActivity.class);
                    startActivity(intent);
                }
                else if (position == 1) {
                    Intent intent = new Intent(MainActivity.this, AutoresActivity.class);
                    startActivity(intent);
                }
                else if (position == 2) {
                    exibeMsg("Opção Editoras - implementação pendente");
                }
            }
        });
    }

    private void exibeMsg(String msg) {
        Toast.makeText(this, msg, Toast.LENGTH_SHORT).show();
    }
}
```

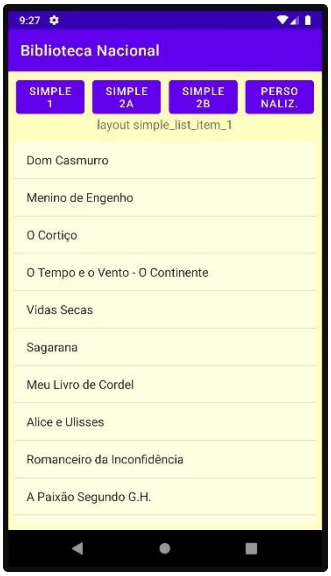
Layout: activity_livros.xml

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/corFundo"
    tools:context=".LivrosActivity">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/gdlH1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.01" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/gdlH2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.14" />

    <Button
        android:id="@+id/btnS1"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:onClick="btnsOnClick"
        android:text="simple1"
        app:layout_constraintEnd_toStartOf="@+id/btnS2a"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/gdlH1" />

    <Button
        android:id="@+id/btnS2a"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:onClick="btnsOnClick"
        android:text="simple2a"
        app:layout_constraintEnd_toStartOf="@+id/btnS2b"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnS1"
        app:layout_constraintTop_toTopOf="@+id/gdlH1" />

    <Button
        android:id="@+id/btnS2b"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:onClick="btnsOnClick"
        android:text="simple2b"
        app:layout_constraintEnd_toStartOf="@+id/btnP"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnS2a"
        app:layout_constraintTop_toTopOf="@+id/gdlH1" />

    <Button
        android:id="@+id/btnP"
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:onClick="btnsOnClick"
        android:text="Personaliz."
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnS2b"
        app:layout_constraintTop_toTopOf="@+id/gdlH1" />

    <TextView
        android:id="@+id/txtOpcLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:text="Escolha o Adaptador tocando nos botões acima"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/gdlH2"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
```

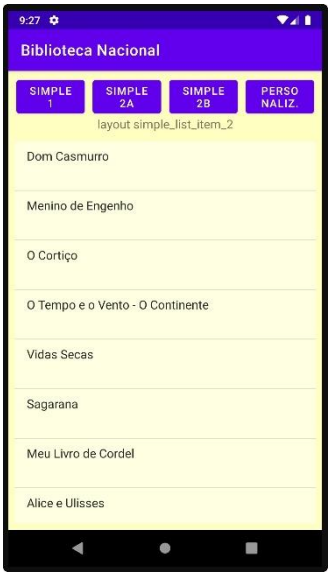
```
    <ListView
        android:id="@+id/lvLivros"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_margin="8dp"
        android:background="@color/corFundoLight"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/gdlH2">

    </ListView>

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

Neste layout o componente ListView lvLivros poderá ser preenchido por 4 diferentes adaptadores. Os layouts das 4 opções são mostrados abaixo

simple_list_item_1



simple_list_item_2



simple_list_item_2 personalizado

totalmente personalizado faixa_livro.xml

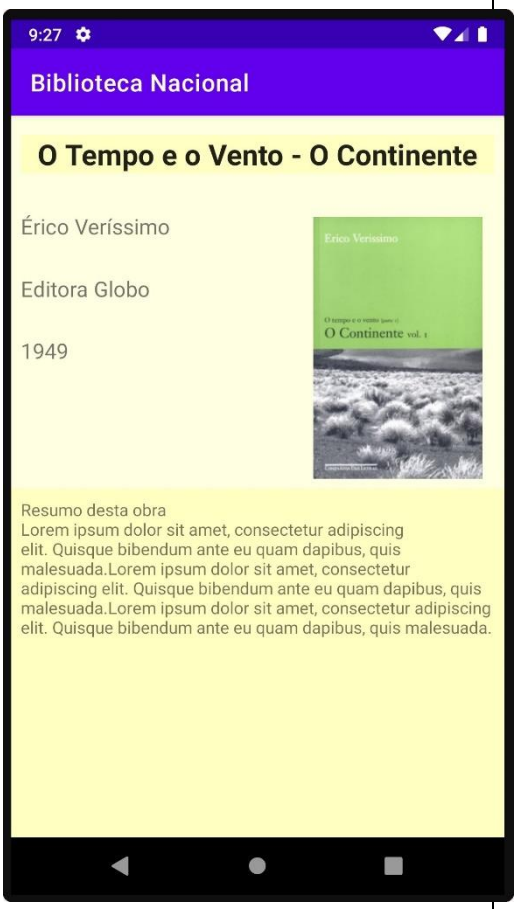
Código: LivrosActivity.java

```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 LivrosActivity extends AppCompatActivity {

    private ListView lvLivros;

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

        lvLivros = findViewById(R.id.lvLivros);

        // configura o listener OnItemClick de lvLivros
        setLvLivrosOnItemClick();
    }

    public void btnsOnClick(View view) {
        // Nesta activity são usados 4 diferentes adaptadores e podemos alternar usando os botões no topo do layout
        TextView txtOpcLayout = findViewById(R.id.txtOpcLayout);
        if (view.getId() == R.id.btnS1) {
            setAdaptadorSimples1();
            txtOpcLayout.setText("layout simple_list_item_1");
        }
        else if (view.getId() == R.id.btnS2a) {
            setAdaptadorSimples2_Basico();
            txtOpcLayout.setText("layout simple_list_item_2");
        }
        else if (view.getId() == R.id.btnS2b) {
            setAdaptadorSimples2_Personalizado();
            txtOpcLayout.setText("layout simple_list_item_2 com getView");
        }
        else if (view.getId() == R.id.btnP) {
            setAdaptadorPersonalizado();
            txtOpcLayout.setText("layout personalizado faixa_livro.xml");
        }
    }

    private void setAdaptadorSimples1() {
        ArrayAdapter<Livro> livroArrayAdapter;
        livroArrayAdapter = new ArrayAdapter<Livro>(
                this,
                android.R.layout.simple_list_item_1,
                Livro.livros
        );
        lvLivros.setAdapter(livroArrayAdapter);
    }

    private void setAdaptadorSimples2_Basico() {
        ArrayAdapter<Livro> livroArrayAdapter;
        livroArrayAdapter = new ArrayAdapter<Livro>(
                this,
                android.R.layout.simple_list_item_2,
                android.R.id.text1,
                Livro.livros);
        lvLivros.setAdapter(livroArrayAdapter);
    }

    private void setAdaptadorSimples2_Personalizado() {
        ArrayAdapter<Livro> livroArrayAdapter;
        livroArrayAdapter = new ArrayAdapter<Livro>(
                this,
                android.R.layout.simple_list_item_2,
                android.R.id.text1,
                Livro.livros)
        {
            @Override
            public View getView(int position, View convertView, ViewGroup parent) {
                View view = super.getView(position, convertView, parent);
                TextView text1 = view.findViewById(android.R.id.text1);
                TextView text2 = view.findViewById(android.R.id.text2);

                text1.setText(Livro.livros[position].getNome());
                int nAutor = Livro.livros[position].getAutor();
                text2.setText(Autor.autores[nAutor].getNome() + " - " + Livro.livros[position].getEditora());
                return view;
```

```
            }
        };
        lvLivros.setAdapter(livroArrayAdapter);
    }

    private void setAdaptadorPersonalizado() {
        LivrosAdapter livrosAdapter = new LivrosAdapter(Arrays.asList(Livro.livros) , this);
        lvLivros.setAdapter(livrosAdapter);
    }

    private void setLvLivrosOnItemClick() {
        lvLivros.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int position, long id) {
                Intent intent = new Intent(LivrosActivity.this, ObraActivity.class);
                intent.putExtra(ObraActivity.EXTRA_NUMLIVRO, position);
                startActivity(intent);
            }
        });
    }

}
```

Layout: activity_obra.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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/corFundoLight"
    tools:context=".ObraActivity">

    <ImageView
        android:id="@+id/imgCapa"
        android:layout_width="145dp"
        android:layout_height="219dp"
        android:layout_marginStart="248dp"
        android:layout_marginTop="36dp"
        android:layout_marginEnd="16dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNome"
        app:srcCompat="@drawable/domcasmurro" />

    <TextView
        android:id="@+id/txtNome"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="8dp"
        android:background="@color/corFundo"
        android:gravity="center_horizontal"
        android:text="Nome Obra"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/txtAutor"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="32dp"
        android:text="Autor"
        android:textSize="18sp"
        app:layout_constraintEnd_toStartOf="@+id/imgCapa"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNome" />

    <TextView
        android:id="@+id/txtEditora"
```



```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="84dp"
        android:text="Editora"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNome" />

    <TextView
        android:id="@+id/txtAnoPub"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="136dp"
        android:text="AnoPub"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNome" />

    <TextView
        android:id="@+id/txtResumoObra"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginTop="8dp"
        android:background="@color/corFundo"
        android:padding="8dp"
        android:text="TextView"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imgCapa" />

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

Código: ObraActivity.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 ObraActivity extends AppCompatActivity {
    public static final String EXTRA_NUMLIVRO = "numlivro";

    private int numAutor;

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

        int numLivro = (Integer)getIntent().getExtras().get(EXTRA_NUMLIVRO);
        Livro obra = Livro.livros[numLivro];

        ImageView imgCapa = findViewById(R.id.imgCapa);
        imgCapa.setImageResource(obra.getImgResId());
        imgCapa.setContentDescription(obra.getNome());
        TextView txt;
        txt = findViewById(R.id.txtNome);
        txt.setText(obra.getNome());
        numAutor = obra.getAutor();
        txt = findViewById(R.id.txtAutor);
        txt.setText(Autor.autores[numAutor].getNome());
        txt = findViewById(R.id.txtEditora);
        txt.setText(obra.getEditora());
        txt = findViewById(R.id.txtAnoPub);
        txt.setText(Integer.toString(obra.getAnoPub()));

        Random rnd = new Random();
        int qtde = rnd.nextInt(7) + 1;
        StringBuilder s = new StringBuilder();
        s.append("Resumo desta obra\n");
        for (int cont = 0; cont < qtde; cont++)
            s.append("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque bibendum ante eu quam dapibus, quis malesuada.");
        txt = findViewById(R.id.txtResumoObra);
        txt.setText(String.valueOf(s));

        setTxtAutorOnClick();
    }
```

```
    private void setTxtAutorOnClick() {
        TextView txtAutor = findViewById(R.id.txtAutor);
        txtAutor.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(ObraActivity.this, UmAutorActivity.class);
                intent.putExtra(UmAutorActivity.EXTRA_NUMAUTOR, numAutor);
                startActivity(intent);
            }
        });

    }
}
```

Layout: activity_autores.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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/corFundo"
    tools:context=".AutoresActivity">

    <ListView
        android:id="@+id/lvAutores"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:divider="@color/black"
        android:dividerHeight="2dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Código: AutoresActivity.java

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


public class AutoresActivity extends AppCompatActivity {

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

        ListView lvAutores = findViewById(R.id.lvAutores);
        AutoresAdapter autoresAdapter = new AutoresAdapter(Arrays.asList(Autor.autores), this);
        lvAutores.setAdapter(autoresAdapter);
    }
}
```

Código: AutoresAdapter.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 AutoresAdapter extends BaseAdapter {

    private final List<Autor> autores;
    private final Activity atividade;

    public AutoresAdapter(List<Autor> autores, Activity atividade) {
        this.autores = autores;
        this.atividade = atividade;
```

```
    }

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

    @Override
    public Object getItem(int i) {
        return autores.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }


    @SuppressLint("ClickableViewAccessibility")
    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        if (view == null) {
            LayoutInflater inflater;
            inflater = atividade.getLayoutInflater();
            view = inflater.inflate(R.layout.faixa_autor, viewGroup, false);
        }
        Autor autor = autores.get(i);
        TextView txt;
        txt = view.findViewById(R.id.txtNome);
        txt.setText(autor.getNome());
        txt = view.findViewById(R.id.txtNomeRegistro);
        txt.setText(String.format("(%s)", autor.getNomeRegistro()));
        txt = view.findViewById(R.id.txtVida);
        txt.setText(autor.getVida());

        txt = view.findViewById(R.id.txtBio);
        StringBuilder s = new StringBuilder();
        s.append("Biografia resumida\n");
        Random rnd = new Random();
        int qtde = rnd.nextInt(6) + 1;
        int cont;
        for (cont = 0; cont < qtde; cont++)
            s.append("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque bibendum ante eu quam dapibus, quis malesuada.");
        txt.setText(s);

        Resources resources = atividade.getResources();
        int resourceid = resources.getIdentifier(autor.getFoto(), "drawable", atividade.getPackageName());
        ImageView img = view.findViewById(R.id.imgFoto);
        img.setImageResource(resourceid);

        return view;
    }
}
```

Observação:

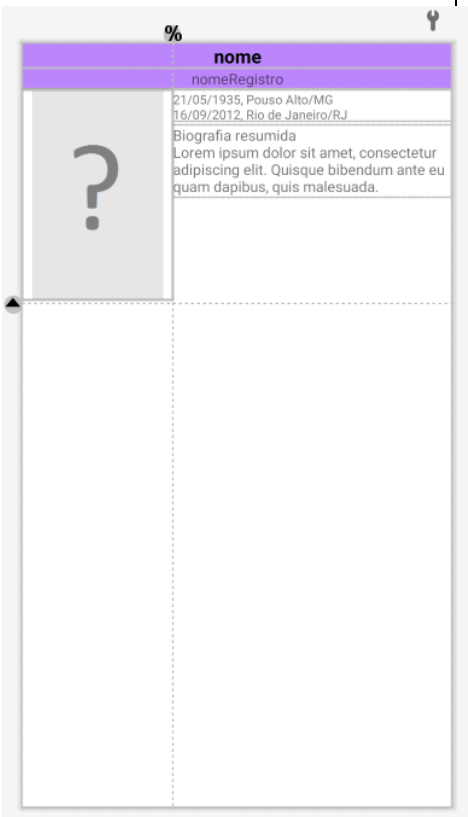
Repare no código acima, dentro do método getView(), é usado o layout faixa_autor. Este layout é usado para preencher o ListView contido em activity_autores.xml

Nas notas técnicas a seguir é descrito, passo a passo, como esses elementos se conectam (e também vimos isto em aula). Essa descrição está no final deste documento, na seção denominada Personalização de Adaptadores – Parte 2. Nessa seção o exemplo mostrado é feito com os Livros. Aqui nesta parte eu disponibilizei os elementos para a exibição dos Autores.

Layout: faixa_autor.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:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/txtBio"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
```

```
        android:text="Biografia resumida\nLorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque bibendum ante eu quam dapibus, quis malesuada."
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="@+id/guideline2"
        app:layout_constraintTop_toBottomOf="@+id/txtVida" />

    <TextView
        android:id="@+id/txtNome"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/purple_200"
        android:gravity="center"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="nome" />

    <TextView
        android:id="@+id/txtNomeRegistro"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:background="@color/purple_200"
        android:gravity="center"
        android:textSize="14sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNome"
        tools:text="nomeRegistro" />

    <ImageView
        android:id="@+id/imgFoto"
        android:layout_width="0dp"
        android:layout_height="200dp"
        android:layout_marginTop="2dp"
        app:layout_constraintEnd_toStartOf="@+id/guideline2"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txtNomeRegistro"
        app:srcCompat="@drawable/semcapa" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.35" />

    <TextView
        android:id="@+id/txtVida"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:textSize="12sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/guideline2"
        app:layout_constraintTop_toBottomOf="@+id/txtNomeRegistro"
        tools:text="21/05/1935, Pouso Alto/MG\n16/09/2012, Rio de Janeiro/RJ" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_begin="250dp" />


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



Agora veja o arquivo XML do layout a seguir. Veja com atenção a tag <include />. Com ela, é possível reaproveitar um layout já existente, inserindo-o e fazendo com que seja parte integrante de um novo layout. Deste modo, em activiy_um_autor.xml reutilizamos o layout faixa_autor.xml

Layout: activity_um_autor.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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/corFundoLight"
    tools:context=".UmAutorActivity">

    <include
        layout="@layout/faixa_autor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />

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

Código: UmAutorActivity.java

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


public class UmAutorActivity extends AppCompatActivity {
    public static final String EXTRA_NUMAUTOR = "numautor";

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

        int numAutor = (Integer)getIntent().getExtras().get(EXTRA_NUMAUTOR);
        Autor autor = Autor.autores[numAutor];

        TextView txt;
        txt = findViewById(R.id.txtNome);
        txt.setText(autor.getNome());
        txt = findViewById(R.id.txtNomeRegistro);
        txt.setText(autor.getNomeRegistro());
        txt = findViewById(R.id.txtVida);
        txt.setText(autor.getVida());

        Resources resources = getResources();
        int resourceid = resources.getIdentifier(autor.getFoto(), "drawable", getPackageName());
        ImageView img = findViewById(R.id.imgFoto);
        img.setImageResource(resourceid);
    }
}
```

Para saber mais sobre reutilização de layouts:

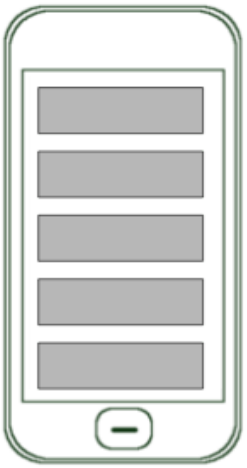
https://developer.android.com/training/improving-layouts/reusing-layouts

Notas Técnicas

Componente ListView

Na plataforma Android o componente ListView foi um dos mais importantes elementos e um dos mais usados. No entanto, atualmente, ele é um legado e é mantido no Android Studio para manter a compatibilidade com aplicativos já desenvolvidos. O componente RecyclerView é seu substituto e tem a vantagem de utilizar com mais eficiência a memória e o processador do dispositivo, bem como é dotado de recursos de animação, rolagem e interação mais modernos e interessantes, do ponto de vista do usuário final.

Apesar disto, neste aplicativo utilizamos o ListView, para que o mesmo seja conhecido pelos alunos uma vez que ainda está presente em muitos aplicativos já prontos e que são mantidos dessa forma pois o custo da alteração seria alto.



Para aplicativos novos convém usar o RecyclerView. Vamos usá-lo em um aplicativo mais adiante.

Este componente exibe uma coleção de Views e é automaticamente dotado de rolagem vertical. Cada View dentro do ListView é uma faixa horizontal, posicionada imediatamente abaixo da View anterior na lista. A imagem à direita ilustra isso. O programador não precisa fazer nada de especial, em termos de código, para que isso ocorra, bastando fornecer dados que preencham cada faixa.

ListView (e RecyclerView) podem ser usados para mostrar uma quantidade bem grande de dados. Porém, devemos ter cuidado com o consumo de memória do dispositivo.

Processador e memória dos dispositivos têm limitações e precisam ser usados de modo eficiente.

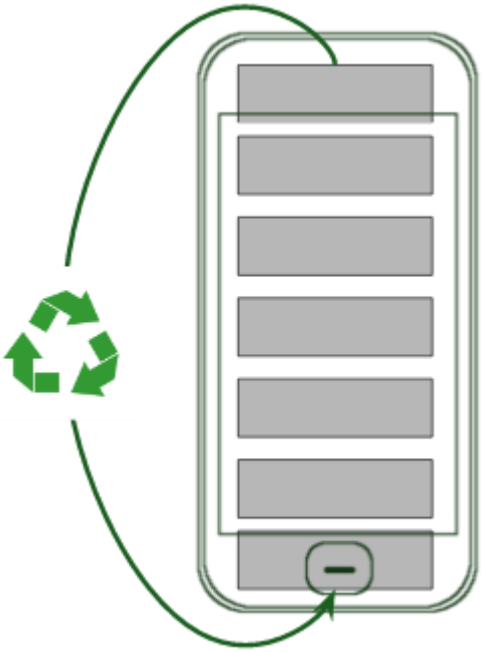
Estes recursos do dispositivo móvel são usados para manipular todos os tipos de elementos com os quais o usuário interage, como músicas, mensagens de texto, fotos e muito mais. Estarão compartilhados entre o sistema operacional Android, todos os aplicativos que o usuário queira, incluindo o nosso aplicativo também.

Agora, imagine um aplicativo de contatos que contém 1.000 nomes, números de telefone e endereços de e-mail armazenados. Se usarmos um LinearLayout na nossa aplicação em conjunto com três TextViews para cada contato, rapidamente vamos consumir toda a memória RAM disponível, bem como degradar a capacidade de processamento deixando o aparelho muito lento e rapidamente consumindo toda a energia da bateria. É inviável.

Por outro lado, em um aplicativo de contatos os usuários só veem alguns contatos na tela por vez. Portanto, não há necessidade de criar milhares de TextViews e ocupar memória de forma desnecessária.

A solução para esse problema é fazer uma reutilização de Views na quantidade exata para o espaço de exibição que se tem na tela. Esse é a principal estratégia dos componentes ListView e RecyclerView, sendo que este último faz as coisas com um grau mais sofisticado de otimização, incluindo efeitos de animação bem suaves enquanto a tela é rolada.

A estratégia de Reciclagem de Views ajuda a exibir listas longas dentro dos aplicativos reutilizando as Views que não estão visíveis na tela naquele momento.



Quando falamos em Views, tenha em mente todo o layout para uma única faixa da lista. O layout para uma única faixa pode ser uma simples caixa de texto, ou algo bem mais complexo composto por várias Views dentro de ViewGroups.

Nós precisamos criar apenas as Views suficientes dos itens que serão mostrados e irão preencher a tela do usuário. Isso significa que se rolarmos a lista para cima, não precisamos mais ver as Views antigas e se uma View não está mais visível na tela, podemos reutilizá-la alterando apenas os dados exibidos através dela, como textos e imagens.

Essa abordagem faz com que não precisemos perder tempo de processador criando a View do zero novamente, bem como consome apenas a memória necessária para as Views que são visíveis e reutilizáveis.

Todas as Views que não estão sendo utilizadas mais, são colocadas em uma “pilha de sucata” para serem recicladas e reutilizadas mais tarde.

Na prática, quando o usuário rola a tela há dados que saem da área visível e outros que entram nessa área. Supondo que a rolagem esteja acontecendo de baixo para cima, uma faixa "sai da tela" em cima, é ocultada, preenchida com novos dados, reposicionada na parte inferior da lista e volta a "entrar na tela" tornando-se visível.

Isso tudo acontece muito rápido e para o usuário, ao rolar a tela, parece que os itens da lista estavam lá o tempo todo.

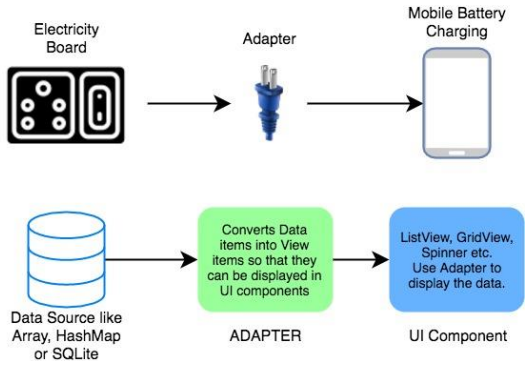
Para saber mais:

https://developer.android.com/reference/android/widget/ListView

https://guides.codepath.com/android/Using-an-ArrayAdapter-with-ListView

Adaptadores

Componentes ListView e RecyclerView não funcionam sem a presença de um Adaptador, ou Adapter, que é como vamos chamar esses elementos. Pode-se afirmar que um Adaptador é um elo de ligação entre uma fonte de dados e o ListView. A fonte de dados pode ser um array ou um banco de dados.



Fonte: Studytonight/adapterview

O Adaptador é responsável por gerenciar e vincular os dados às Views que compõem o layout de cada faixado ListView. O Adaptador é o responsável por criar a aparência visual para cada item do conjunto de dados e exibi-lo em uma faixa.

Um exemplo de layout muito utilizado em uma lista, tem uma imagem no lado esquerdo e duas linhas de texto no meio, conforme a imagem a seguir.

O Adaptador infla o layout de cada faixa contida no ListView e atribui os dados para as Views individuais. O Android fornece algumas implementações de Adaptador prontas para serem usadas e nada impede o programador de criar sua própria implementação. Fizemos exatamente isso neste aplicativo, quando implementamos a 3ª e 4ª opções para a atividade LivrosActivity.

As classes de Adaptadores mais usadas são: BaseAdapter, ArrayAdapter, SimpleCursorAdapter e CursorAdapter.

Todo Adaptador é uma classe Java, sendo que BaseAdapter é a classe raiz de todas as demais.

ArrayAdapter é especializada em preencher o ListView a partir de arrays de objetos.

SimpleCursorAdapter e CursorAdapter são especializadas em preencher o ListView a partir de dados vindos do banco de dados do Android, o SQLite (que veremos no nosso próximo aplicativo).

Para saber mais:

https://developer.android.com/topic/libraries/data-binding/binding-adapters

https://developer.android.com/reference/android/widget/Adapter

O que é inflar um layout?

Trata-se de um termo muito comum no ambiente Android. Já sabemos que um layout é um arquivo XML.

A questão é: como fazemos para transformar esse XML em algo que possa ser utilizável no código Java?

É necessário criar objetos Java que tenham a aparência e as funcionalidades especificadas no XML. "Inflar o layout" é essa conversão de especificações XML em objetos Java funcionais. Quando um layout é inflado isso significa que um arquivo XML foi lido e interpretado pelo programa e todos os objetos Java necessários foram instanciados e construídos em memória, ficando disponíveis para terem seus métodos executados pelo aplicativo.

Parece muita coisa e parece complicado de fazer. E é mesmo.

Mas a boa notícia é que essas coisas complicadas não são feitas por nós programadores de aplicativos. Quem faz isso é o pessoal que desenvolve a API do Android. Nós temos apenas que saber usar a classe LayoutInflater disponível nas APIs do Android desde a versão 1.

Na 4ª versão do layout em lvLivros deste aplicativo é exatamente o que fazemos: usamos a classe LayoutInflater para criar a nossa view. Depois é só preencher usando findViewById e os métodos setText(), setImageResource(), etc...

Dê uma olhada no código do Adaptador personalizado na classe LivroAdapter.java (mais adiante) e você compreenderá como é simples inflar e preencher uma View contida em um ListView.

Personalização de Adaptadores – Parte 1

Como foi apresentado em aula, neste aplicativo usamos 4 diferentes Adaptadores para fazer a conexão entre a origem de dados, o vetor de livros, e o componente visual, o ListView lvLivros.

Os dois primeiros exigem apenas que façamos uso do construtor da classe ArrayAdapter<> passando os parâmetros requeridos. Nestes dois casos usamos dois layouts – ***simple_list_item1*** e ***simple_list_item2*** – prontos do Android, informamos a origem dos dados e, no caso do layout com dois componentes TextView informamos qual deles deverá ser preenchido quando o adaptador inflar a View.

No terceiro caso usamos o layout ***simple_list_item2*** e fizemos uma personalização usando uma classe anônima aninhada na chamada do construtor da classe ArrayAdapter<>. Esse não é um conceito muito óbvio para quem está iniciando em programação Java, então existe a necessidade de explicação específica. Vamos lá.

E neste aplicativo estamos trabalhando com os adaptadores do ramo ArrayAdapter que tem a função de preencher um ListView com os dados provenientes de um array. e para personalizá-lo o programador deve criar uma classe herdeira, sobrescrevendo o método getView(). É dentro desse método que se escreve o código que irá preencher o layout da forma desejada, ou seja, personalizada.

Veja o código do método `setAdaptadorSimples2_Personalizado()` copiado a seguir.

```
private void setAdaptadorSimples2_Personalizado() {
    ArrayAdapter<Livro> livroArrayAdapter;
    livroArrayAdapter = new ArrayAdapter<Livro>(
            this,
            android.R.layout.simple_list_item_2,
            android.R.id.text1,
            Livro.livros)
    {
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            View view = super.getView(position, convertView, parent);
            TextView text1 = view.findViewById(android.R.id.text1);
            TextView text2 = view.findViewById(android.R.id.text2);

            text2.setText(Livro.livros[position].getNome() + " - " + Livro.livros[position].getEditora());
            text1.setText(Livro.livros[position].getAutor());
            return view;
        }
    };
    lvLivros.setAdapter(livroArrayAdapter);
}
```

Construtor ArrayAdapter<>

Nele usamos o construtor da classe e logo a seguir abrimos um bloco com chaves {}. É a parte destacada com o retângulo e nela fazemos a sobrescrita do método getView(). O fato de sobrescrever um método significa que estamos criando uma nova classe herdeira, pois é assim que o interpretador Java trata essa situação. Como não demos um nome a essa nova classe decorre que trata-se de uma classe anônima.

Para saber mais sobre classes aninhadas e anônimas leia este artigo:

https://www.devmedia.com.br/classes-anonimas-e-aninhadas-em-java/31167

Personalização de Adaptadores – Parte 2

Na parte 1 falamos de personalização envolvendo uma classe anônima definida no momento da construção do ArrayAdapter<>.

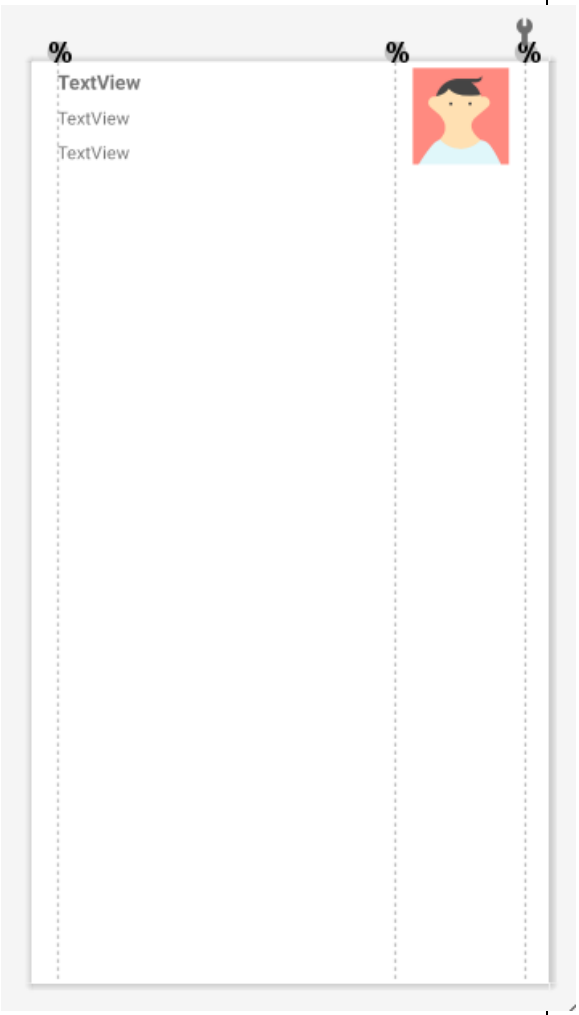
Agora vamos tratar de uma personalização mais completa, envolvendo o uso de layout criado para esse fim e a criação de uma classe totalmente voltada para a personalização. Essa personalização envolve três passos:

1. Crie o layout desejado. Neste aplicativo criamos o layout faixa_livros.xml;
2. Crie uma classe estendendo a classe BaseAdapter e implemente seu construtor e métodos abstratos. Será preciso implementar os métodos abstratos getCount(), getItem(), getItemId(), getView(). Neste aplicativo criamos a classe LivrosAdapter;
3. Use o LivrosAdapter na atividade LivrosActivity para preencher o ListView.

Foi o que fizemos e reproduzimos os dois primeiros elementos a seguir. O terceiro elemento está no código do método setAdaptadorPersonalizado() da classe LivrosActivity, na página 7 acima.

Layout: faixa_livro.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/gdl1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.05" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/gdl2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.7" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/gdl3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.95" />

    <TextView
        android:id="@+id/txtNome"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:text="TextView"
        android:textSize="16sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toStartOf="@+id/gdl2"
        app:layout_constraintStart_toStartOf="@+id/gdl1"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/txtAutor"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="TextView"
        app:layout_constraintEnd_toStartOf="@+id/gdl2"
        app:layout_constraintStart_toStartOf="@+id/gdl1"
        app:layout_constraintTop_toBottomOf="@+id/txtNome" />

    <TextView
        android:id="@+id/txtEditoraAno"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="TextView"
        app:layout_constraintEnd_toStartOf="@+id/gdl2"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/gdl1"
        app:layout_constraintTop_toBottomOf="@+id/txtAutor" />

    <ImageView
        android:id="@+id/imgCapa"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="4dp"
        android:scaleType="fitCenter"
        app:layout_constraintBottom_toBottomOf="@+id/txtEditoraAno"
        app:layout_constraintEnd_toStartOf="@+id/gdl3"
        app:layout_constraintStart_toStartOf="@+id/gdl2"
        app:layout_constraintTop_toTopOf="@+id/txtNome"
        tools:srcCompat="@tools:sample/avatars" />

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



Código: LivroAdapter.java

```
public class LivrosAdapter extends BaseAdapter {

    private final List<Livro> livros;
    private final Activity atividade;

    public LivrosAdapter(List<Livro> livros, Activity atividade) {
        this.livros = livros;
        this.atividade = atividade;
    }

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

    @Override
    public Object getItem(int i) {
        return livros.get(i);
    }

    @Override
    public long getItemId(int i) {
        return i;
    }

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        if (view == null) {
            LayoutInflater inflater;
            inflater = atividade.getLayoutInflater();
            view = inflater.inflate(R.layout.faixa_livro, viewGroup, false);
        }
        Livro livro = livros.get(i);
        TextView txtTitulo = view.findViewById(R.id.txtNome);
        txtTitulo.setText(livro.getNome());
        int nAutor = livro.getAutor();
        TextView txtAutor = view.findViewById(R.id.txtAutor);
        txtAutor.setText(Autor.autores[nAutor].getNome());
        TextView txtEditoraAno = view.findViewById(R.id.txtEditoraAno);
        txtEditoraAno.setText(String.format("%s - %d", livro.getEditora(), livro.getAnoPub()));
        ImageView imgCapa = view.findViewById(R.id.imgCapa);
        imgCapa.setImageResource(livro.getImgResId());
        return view;
    }
}
```

Formas de uso dos drawables

Nas classes Autor e Livro foram usados drawables, porém de formas diferentes.

Na classe Autor são fotos dos autores e a forma de uso é através do nome do drawable. Essa classe contém uma propriedade String que será usada para guardar o nome do arquivo da imagem (sem a extensão .png – pq não é necessária). Essa propriedade será usada para carregar a foto do autor no componente ImageView. Essa forma de fazer vai exigir alguns passos a mais, conforme veremos adiante.

Na classe Livro são imagens das capas dos livros e a forma de uso é o número do drawable. Não comentamos isso ainda, mas todo drawable, quando inserido no projeto, recebe automaticamente um número inteiro. O programador não precisa conhecer esse número, mas fará referência a ele através da classe R, deste modo: R.drawable.*id_do_elemento*.

Nessa classe Livro há uma propriedade numérica inteira chamada imgResId que armazenará o *id_do_elemento*. Desse modo ele poderá ser usado diretamente na exibição da imagem no componente ImageView. Essa forma de fazer é mais direta.

Agora vamos ver os dois códigos usados para executar a carga da imagem no componente ImageView. Isso é feito com o método setImageResource(), e já o utilizamos no aplicativo "Trabalhando com Imagens". Esse método requer a passagem de um parâmetro inteiro que é o *id_do_elemento*.

Classe Autor	`Resources resources = atividade.getResources();`
	`int resourceid = resources.getIdentifier(autor.getFoto(), "drawable", atividade.getPackageName());`
	`ImageView img = view.findViewById(R.id.imgFoto);`
	`img.setImageResource(resourceid);`

Classe Livro	`ImageView imgCapa = view.findViewById(R.id.imgCapa);` `imgCapa.setImageResource(livro.getImgResId());`

Conforme pode ser visto acima, na classe Livro a carga da imagem é feita de forma imediata com setImageResource(). Na classe Autor, como não temos o *id_do_elemento*, mas sim o nome do drawable é preciso utilizar a classe Resources para recuperar o id_do_elemento cujo nome conhecemos. Isso é feito com o método getIdentifier() e ele recebe três parâmetros: o nome do drawable, o literal "drawable" indicando que estamos interessado em recuperar um elemento dessa natureza e o nome do pacote (package name) ao qual pertence o drawable desejado. Resumindo: temos o nome do drawable; usamos esse nome com o método getIdentifier() da classe Resources para obter seu id; com esse id carregamos a imagem.

Roteiro para criação deste aplicativo

Como vocês devem ter percebido, neste aplicativo demos um salto grande na quantidade de elementos necessários à sua criação. Além disso, o uso de Listeners e o uso de Adaptadores acrescenta um salto na complexidade. Por isso, é natural que quem está iniciando nesse aprendizado se sinta perdido.

Por esse motivo eu criei este roteiro. Este roteiro faz sentido depois que você já assistiu a aula. Na aula eu uso uma sequência diferente porque na aula estou interessado em ser mais didático.

Este roteiro serve como um planejamento de criação do aplicativo, voltado para quem já sabe o que deve ser feito.

1. Crie um projeto novo com Empty Activity
2. Baixe e descompacte o arquivo de resources que está na minha página e copie o conteúdo na pasta *drawable*
3. Edite os arquivos de resources strings.xml e colors.xml acrescentando os elementos usados
4. Crie a classe Livro
5. Crie a classe Autor
6. Crie o layout faixa_autor.xml
7. Crie o layout faixa_livro.xml
8. Crie a classe AutoresAdapter estendendo BaseAdapter
9. Crie a classe LivrosAdapter estendendo BaseAdapter
10. Crie o layout da atividade principal: activity_main.xml
11. Implemente o código Java da atividade principal: ActivityMain.java. Você poderá testar se o listener OnItemClick() do ListView está funcionando usando mensagens Toast. Com o listener funcionando, o próximo passo é usar Intent para abrir as duas atividades que ainda não foram criadas: LivrosActivity e AutoresActivity. Então, esses são os próximos passos
12. Crie uma nova atividade baseada no template Empty Activity → esta será LivrosActivity
13. Monte o layout: activity_livros.xml
14. Implemente o código Java da atividade LivrosActivity.java
15. Crie uma nova atividade baseada no template Empty Activity → esta será AutoresActivity
16. Monte o layout: activity_autores.xml
17. Implemente o código Java da atividade AutoresActivity.java
18. Teste o funcionamento e a navegação entre essas três atividades iniciais
19. Crie uma nova atividade baseada no template Empty Activity → esta será ObrasActivity
20. Monte o layout: activity_obras.xml
21. Implemente o código Java da atividade ObrasActivity.java
22. Crie uma nova atividade baseada no template Empty Activity → esta será UmAutorActivity
23. Monte o layout: activity_um_autor.xml
24. Implemente o código Java da atividade UmAutorActivity.java