Desenvolvimento para Dispositivos Móveis

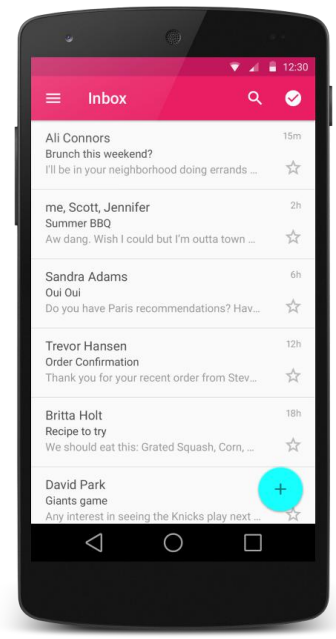
Prof. Dr. Alan Souza

alan.souza@unama.br

2020

9. RecyclerView - Listas

- RecyclerView é um widget que permite a criação de listas de dados;
- Ele é a evolução do ListView;
- A principal diferença entre o RecyclerView e a ListView é em relação à performance: o RecyclerView é superior;
- Por outro lado, em relação à implementação (programação), o ListView é mais simples que o RecyclerView.



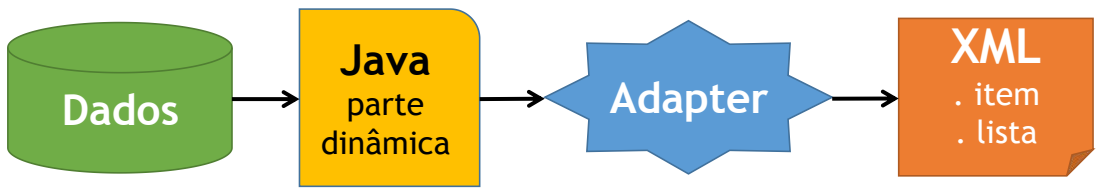
9. RecyclerView - Listas

• Passos para utilização do RecyclerView:

1. Adicionar dependência no arquivo build.gradle (Módulo app);
2. Criar um layout “solto” do item da lista;
3. Adicionar RecyclerView ao layout (XML, parte estática);
4. Criar um Adapter para a lista;
5. Iniciar a parte dinâmica (Java do layout);

9. RecyclerView - Listas

• Esquema de funcionamento do RecyclerView:

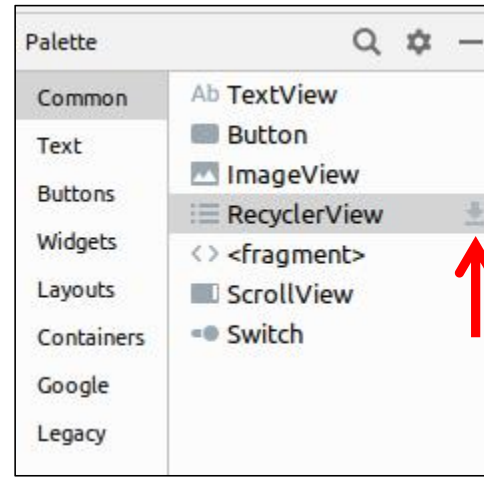


9. RecyclerView - Listas

- Passos para utilização do Recycler View:

1. **Adicionar dependência no arquivo build.gradle (Módulo app):**

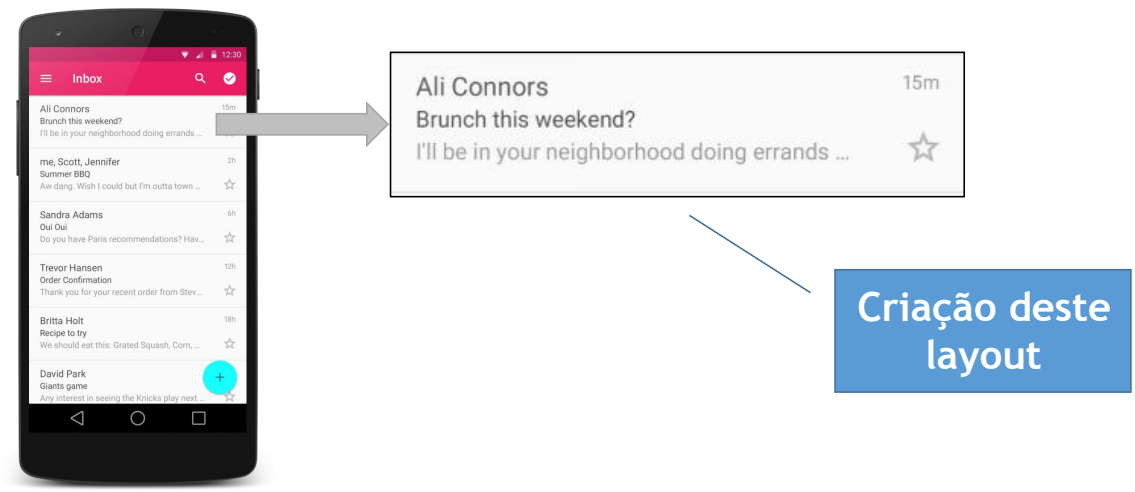
```
dependencies {
   implementation 'androidx.recyclerview
         :recyclerview:1.0.0'  (LINHA ÚNICA)
}
```



9. RecyclerView - Listas

- Passos para utilização do RecyclerView:

2. **Criar um layout "solto" do item da lista:**



9. RecyclerView - Listas

• Passos para utilização do RecyclerView:

2. Criar um layout "solto" do item da lista:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:orientation="vertical">
    <TextView
        android:id="@+id/txt_item_lista"
        ... />
</LinearLayout>
```

9. RecyclerView - Listas

• Passos para utilização do RecyclerView:

3. Adicionar RecyclerView ao layout (XML, parte estática,*drag and drop/a*rrastar e soltar):

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/lista_rv"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

9. RecyclerView - Listas

4. Criar um Adapter para a lista (1/5):

```
public class MeuAdapter extends
                RecyclerView.Adapter<MeuAdapter.ViewHolder> {
  private String[ ] mDataset;
  // Contrutor do Adapter:
  public MeuAdapter(String[ ] dados) {
    this.mDataset = dados;
  }
  // continua...
```

9. RecyclerView - Listas

4. Criar um Adapter para a lista (2/5):

```
public class MeuAdapter extends
        RecyclerView.Adapter<MeuAdapter.ViewHolder> {
  // códigos anteriores...
  // Concede uma referência para os componentes de cada item da lista
  public static class ViewHolder extends RecyclerView.ViewHolder {
    public TextView textView;

    public ViewHolder(View item) {
      super(item);
      textView = item.findViewById(R.id.txt_item_lista);
    }
  } // fim da classe estática
```

9. RecyclerView - Listas

4. Criar um Adapter para a lista (3/5):

```
public class MeuAdapter extends
        RecyclerView.Adapter<MeuAdapter.ViewHolder> {
  // códigos anteriores...
  // Infla o tipo de layout do item da lista
  @Override
   public MeuAdapter.ViewHolder onCreateViewHolder(
                        ViewGroup parent, int viewType) {
      View view = LayoutInflater.from(parent.getContext()).
            inflate(R.layout.item_lista, parent, false);
      ViewHolder viewHolder = new ViewHolder(view);
      return viewHolder;
   }
```

9. RecyclerView - Listas

4. Criar um Adapter para a lista (4/5):

```
public class MeuAdapter extends
        RecyclerView.Adapter<MeuAdapter.ViewHolder> {
  // códigos anteriores...
  // Muda o conteúdo dos itens
  @Override
   public void onBindViewHolder(@NonNull ViewHolder holder,
                        int position) {
      holder.textView.setText(mDataset[position]);
   }
```

9. RecyclerView - Listas

4. Criar um Adapter para a lista (5/5):

```
public class MeuAdapter extends
        RecyclerView.Adapter<MeuAdapter.ViewHolder> {
  // códigos anteriores...
  // Retorna o tamanho do conjunto de dados (layout manager precisa)
  @Override
  public int getItemCount( ) {
    return mDataset.length;
  }

} // fim do MeuAdapter
```

9. RecyclerView - Listas

5. Iniciar a parte dinâmica (Java do layout) (1/3):

```
public class MainActivity extends AppCompatActivity {
  // declaração do Adapter e do RecyclerView:
  private MeuAdapter mListadapter;
  private RecyclerView mRecyclerView;

  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // integração entre XML e JAVA:
    mRecyclerView  = findViewById(R.id.lista_rv);
    // continua...
```

9. RecyclerView - Listas

5. Iniciar a parte dinâmica (Java do layout) (2/3):

```
public class MainActivity extends AppCompatActivity {
   protected void onCreate(Bundle savedInstanceState) {
      // códigos anteriores...
      LinearLayoutManager m = new LinearLayoutManager(this);
      m.setOrientation(LinearLayoutManager.VERTICAL);
      mRecyclerView.setLayoutManager(m);
      mRecyclerView.setHasFixedSize(true);
```

9. RecyclerView - Listas

5. Iniciar a parte dinâmica (Java do layout) (3/3):

```
public class MainActivity extends AppCompatActivity {
   protected void onCreate(Bundle savedInstanceState) {
      // códigos anteriores...
      // criando dados fictícios e passando para o construtor do Adapter
      String cores[ ] = {"Azul", "Branco", "Verde", "Vermelho", "Rosa",
                  "Laranja", "Lilás", "Amarelo", "Preto", "Magenta", "Ciano"};
      mListadapter = new MeuAdapter( cores );
      // atribuindo o Adapter no RecyclerView:
      mRecyclerView.setAdapter(mListadapter);
   } // fim do onCreate
```

9. RecyclerView - Listas

Resultado:

Desenvolvimento para Dispositivos Móveis

Prof. Dr. Alan Souza

alan.souza@unama.br

2020

9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

```
public class CorBD extends SQLiteOpenHelper {
  // Inner class para definir o conteúdo da tabela:
  public static class TabCor implements BaseColumns{
    public static final String TABELA = "tab_cor";
    public static final String COL_NOME = "nome";
  }
  // String de criação da tabela:
  private static final String SQL_TABELA =
    "CREATE TABLE IF NOT EXISTS " + TabCor.TABELA + " (" +
    "_id INTEGER PRIMARY KEY , " +
    TabCor.COL_NOME + " TEXT )";
}
```

9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

```
public class CorBD extends SQLiteOpenHelper {
  // Continuação...
  public static final int BD_VERSAO = 1;
  public static final String BD_NOME = "BancoDados.db";

  public CorBD(Context context) {
    super(context, BD_NOME, null, BD_VERSAO);
  }
  public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_TABELA);
  }
}
```

9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

```
public class CorBD extends SQLiteOpenHelper {
    // Continuação...
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {

    }
}
```

9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

```
public class MainActivity extends AppCompatActivity {
   protected void onCreate(Bundle savedInstanceState) {
      CorBD corBD = new CorBD( this );
      // Pega o BD no modo de leitura:
      SQLiteDatabase bd = corBD.getReadableDatabase();

   } // fim do onCreate
```

9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

```
public class MainActivity extends AppCompatActivity {
  protected void onCreate(Bundle savedInstanceState) {
    Cursor cursor = bd.query(
      CorBD.TabCor.TABELA, // Nome da tabela
      null, // Um vetor com o nome das colunas de retorno
      null, // Colunas para fazer filtro (where)
      null, // Valores de cada coluna do filtro do where
      null, // Colunas para agrupar (group by)
      null, // Colunas para having
      CorBD.TabCor.COL_NOME // Coluna para ordenação dos dados
      );
  } // fim do onCreate
```

9. RecyclerView - Listas

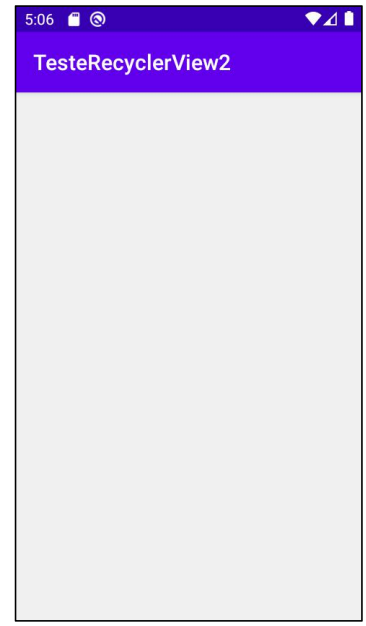
6. Criando e usando um Banco de Dados:

```
public class MainActivity extends AppCompatActivity {
  protected void onCreate(Bundle savedInstanceState) {
    List listaCores = new ArrayList<>();
    while(cursor.moveToNext()) {
      String cor = cursor.getString(
      cursor.getColumnIndexOrThrow(CorBD.TabCor.COL_NOME)
      );
      listaCores.add(cor);
    }
    cursor.close();
  } // fim do onCreate
```

9. RecyclerView - Listas

Resultado:

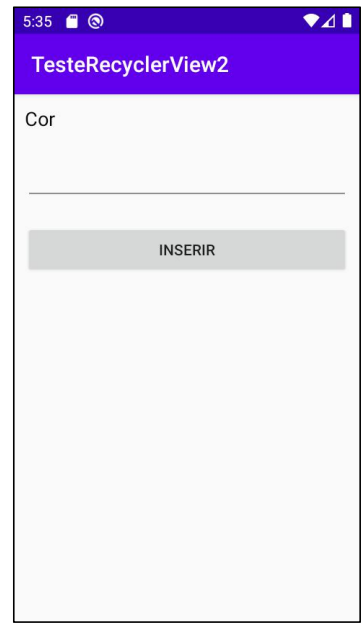
- Tela vazia, porque não tem nenhum dado inserido no banco de dados.
- Vamos criar a tela de inserir cor!



9. RecyclerView - Listas

Resultado:

- Tela para inserir cor:
- Bem simples!



9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

```
public class InserirCorActivity extends AppCompatActivity {
  public void inserirCor(View v) {
    CorBD corBD = new CorBD( getContext() );
    // Pega o BD no modo de escrita:
    SQLiteDatabase bd = corBD.getWritableDatabase();
    // Criar valor para ser armazenado no BD:
    ContentValues valor = new ContentValues();
    valor.put(CorBD.TabCor.COL_NOME, <COR DIGITADA>);
    // Inserir uma nova linha, retornando a chave-primária:
    long linhaID = bd.insert(CorBD.TabCor.TABELA, null, valor);
  } // fim do onCreate
```

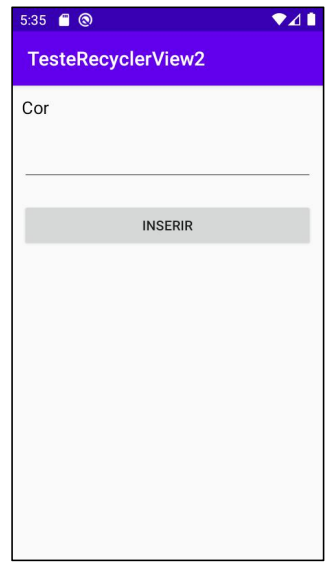
9. RecyclerView - Listas

6. Criando e usando um Banco de Dados:

- Na classe InserirCorActivity colocar try-catch no método de inserção;
- Fazer verificação se foi informado valor para inserir;
- Chamar o método inserirCor no clique do botão;
- Consertar o bug de recarregar a lista quando voltar para tela principal.

9. RecyclerView - Listas

Resultado final esperado:



9. RecyclerView - Listas

Exercício

1. Vamos reproduzir o código no Android Studio, realizando o mínimo de consulta possível ao material.