Desafio Android - RecyclerView

Lucas Sales

Sobre o RecyclerView

O RecyclerView é uma evolução da ListView, presente desde a primeira versão do Android para se fazer listas de conteúdo.
Suas principais vantagens são:

- **Performance na atualização de itens**: a ListView está ligada ao Adapter, que possui uma lista de objetos. Se inserirmos um objeto nessa lista, temos que invocar o método notifyDatasetChanged() que fará com que toda a lista seja refeita/redesenhada. Com a RecyclerView podemos atualizar só um item da lista (inserindo/atualizando/excluindo) ou um intervalo específico.
- **Layouts diferenciados para cada situação**: com a RecyclerView podemos configurar gerenciadores de layouts, indicando por exemplo, que a lista terá uma única coluna quando o aparelho estiver em portrait e duas quando estiver em landscape. Ou ainda dizer que o primeiro item da lista será diferente dos demais.
- **Animações e gestos**: Com a RecyclerView, à medida que os itens vão sendo adicionados ou removidos, uma animação é realizada dando um feedback visual para o usuário do que aconteceu. A utilização de gestos também ficou bastante simples. Ações como o swipe sobre um item da lista é algo trivial de ser feito.
- **Scroll em ambos os sentidos**: a RecyclerView permite o scroll em na horizontal e na vertical, o que não era possível nativamente na ListView.
- **Baixa curva de aprendizagem**: o conceito utilizado pela RecyclerView é muito parecido com o que temos na ListView. Então quem já a conhece não terá muitos problemas.

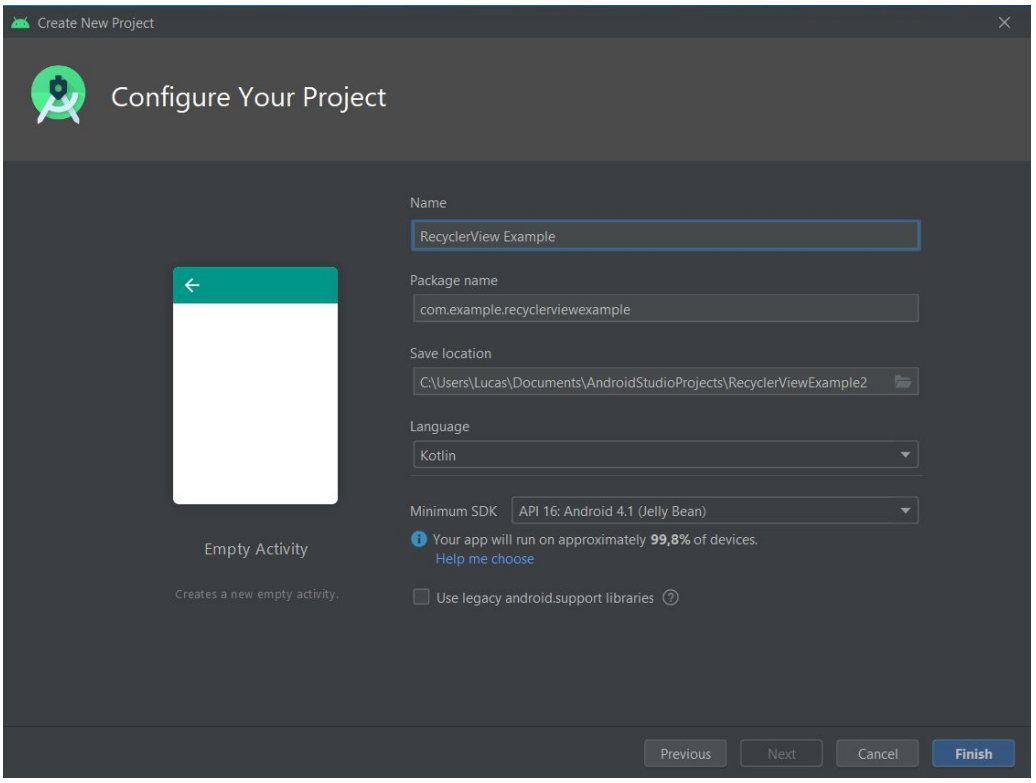
Componentes no RecyclerView

RecyclerView é apenas o ViewGroup onde irá conter a lista, para utilizá-lo é necessário alguns componentes a mais. Segue uma breve descrição abaixo:

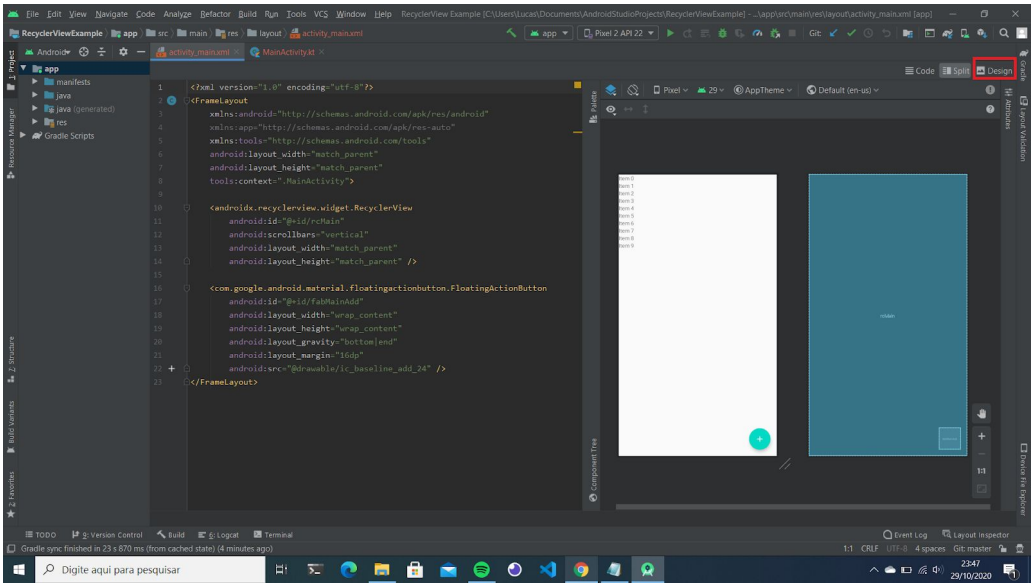
- **RecyclerView**: Componente visual que ficará na Activity e irá posicionar a lista na tela do usuário.
- **LayoutManager**: Nele é definido qual é a disposição dos itens da lista (se será uma lista vertical ou horizontal, por exemplo).
- **Adapter**: Classe responsável por associar a lista de conteúdo a view correspondente. Onde cada objeto da lista será um item na lista. É no Adapter onde se define se um item será exibido ou não.
- **ViewHolder**: É a referência para a view que é a parte visual de cada item da lista, que será replicada para todos os elementos.

Aplicativo Exemplo

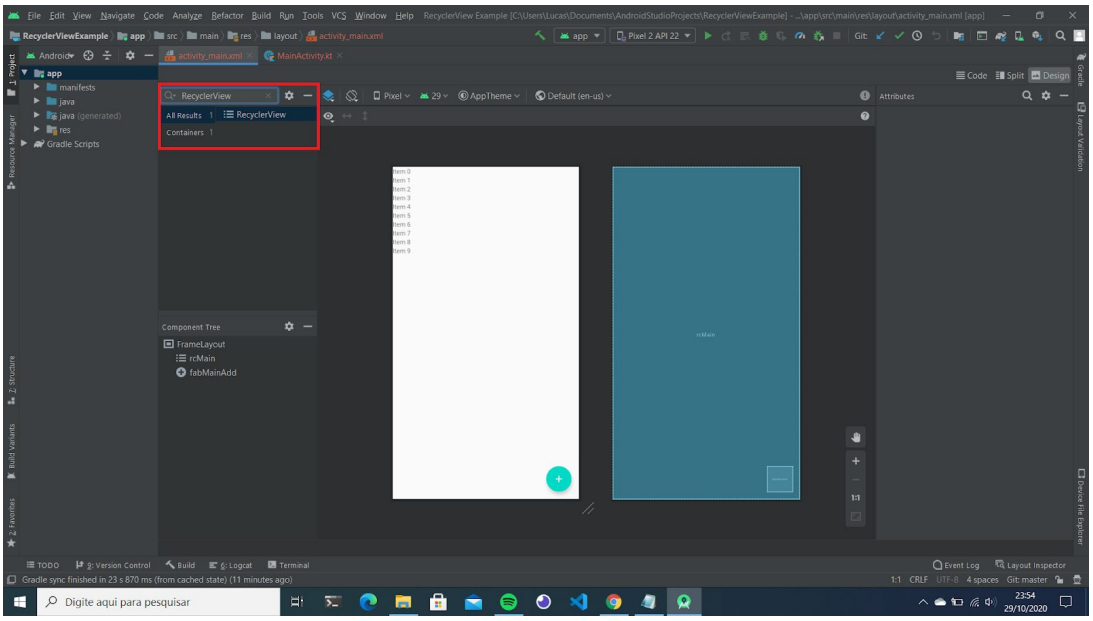
O aplicativo é uma demonstração simples do uso do RecyclerView, nele é possível adicionar nomes em uma lista e exibi-la. Primeiramente se deve criar o projeto no Android Studio como na imagem a seguir:



Com o projeto aberto, selecione o arquivo xml responsável por gerar a tela inicial do aplicativo e clique na aba “Design” no canto superior direito como na imagem abaixo:



Em seguida, adicione o componente RecyclerView a tela, procure por ele digitando na lupa que fica acima, como na imagem a seguir:



Arraste e solte ele na tela ao lado que representa a view, aparecerá uma caixa de diálogo perguntando se você quer adicionar o componente ao projeto, aceite e espere o download do componente terminar. Faça o mesmo processo para o componente FloatingActionButton que iremos utilizar para levar ao formulário que adiciona novos itens à lista, arraste e solte o componente na tela e na janela que abrir procure pelo ícone “ic_baseline_add_24” selecione-o e clique em ok . Volte para a aba “Split” e edite o xml para ficar como na imagem abaixo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    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"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rcMain"
        android:scrollbars="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabMainAdd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        android:src="@drawable/ic_baseline_add_24" />
</FrameLayout>
```

Nessa view há apenas o RecyclerView e o FloatingActionButton responsável por levar ao formulário de criação de novos itens da lista.
É preciso fazer agora o layout do ViewHolder, que será o layout usado em cada uma das linhas. Abra a pasta do projeto no lado esquerdo da tela e procure pela pasta layout em app>res>layout, clique com o botão direito do mouse nela e crie um novo arquivo xml, New>Layout Resource File. Nomeie como list_layout e clique em ok, edite-o para que fique como na imagem a seguir:

```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="100dp"
    android:orientation="horizontal">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvListaNome"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Nome"
            android:textSize="30dp" />

    </LinearLayout>

</LinearLayout>
```

Em seguida, crie uma nova classe que representará a classe de domínio de negócio, a lista na tela principal será uma lista de objetos dessa classe, para esse exemplo criaremos uma classe qualquer apenas para exemplificar. Na pasta do projeto no lado esquerdo procure por app>java>com.example.recyclerviewexample, clique com o botão direito do mouse nela e New>Kotlin File/Class, dê o nome de Example e abaixo escolha a opção Class. Edite o arquivo para ficar como na imagem abaixo:

```kotlin
class Example(var nome: String): Serializable {
    override fun toString(): String {
        return "Nome: ${nome}"
    }
}
```

Faça os mesmos passos executados na criação da classe de exemplo para criar o Recycler Adapter que deve ficar dessa forma:

```
class RecyclerAdapter(var lista: ArrayList<Example>) : RecyclerView.Adapter<RecyclerAdapter.ViewHolder>() {
    class ViewHolder(view: View): RecyclerView.ViewHolder(view) {
        val tvNome = view.findViewById<TextView>(R.id.tvListaNome)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.list_layout, parent,  attachToRoot: false)
        return ViewHolder(view)
    }

    override fun getItemCount(): Int {
        return lista.size
    }

    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val example = lista[position]
        holder.tvNome.text = example.nome
    }
}
```

Agora devemos criar uma nova Activity que será a do formulário responsável pela criação de novos objetos e que passará esses objetos para a Activity principal. No canto superior esquerdo clique em File>New>Activity>Empty Activity, dê o nome de FormActivity e clique em finish, o xml e a activity do formulário deverão ficar como nas imagens abaixo, respectivamente:

```
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".FormActivity">
    <EditText
        android:id="@+id/etAddNome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Nome"
        android:gravity="center"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
        <Button
            android:id="@+id/btnAddCancelar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Cancelar"/>
        <Button
            android:id="@+id/btnAddSalvar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Salvar"/>
    </LinearLayout>
</LinearLayout>
```

Essa view tem um EditText que será onde o usuário digitará o nome que ele deseja adicionar à lista e dois botões para cancelar e voltar para a tela inicial ou adicionar um novo item na lista.

```
class FormActivity : AppCompatActivity() {
    private lateinit var etNome: EditText
    private lateinit var btnCancelar: Button
    private lateinit var btnSalvar: Button

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_form)

        this.etNome = findViewById(R.id.etAddNome)
        this.btnCancelar = findViewById(R.id.btnAddCancelar)
        this.btnSalvar = findViewById(R.id.btnAddSalvar)
        this.btnCancelar.setOnClickListener { cancelar(it) }
        this.btnSalvar.setOnClickListener { salvar(it) }
    }

    private fun cancelar(view: View) {
        finish()
    }

    private fun salvar(view: View) {
        val nome = this.etNome.text.toString()
        val example  = Example(nome)
        val it = Intent()
        it.putExtra( name: "EXAMPLE", example)
        setResult(Activity.RESULT_OK, it)
        finish()
    }
}
```

Aqui é possível ver os métodos responsáveis por adicionar ou cancelar.
E por fim, o MainActivity deverá ficar como na imagem a seguir:

```
class MainActivity : AppCompatActivity() {
    private lateinit var rcMain: RecyclerView
    private lateinit var fabAdd: FloatingActionButton
    private lateinit var lista: ArrayList<Example>
    val FORM = 1

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        this.lista = ArrayList()
        this.rcMain = findViewById(R.id.rcMain)
        this.fabAdd = findViewById(R.id.fabMainAdd)
        rcMain.layoutManager = LinearLayoutManager( context: this, RecyclerView.VERTICAL,  reverseLayout: false)
        rcMain.adapter = RecyclerAdapter(lista)
        this.fabAdd.setOnClickListener { add(it) }
    }

    private fun add(view: View) {
        val it = Intent( packageContext: this@MainActivity, FormActivity::class.java)
        startActivityForResult(it, FORM)
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)
        if(resultCode == Activity.RESULT_OK && requestCode == FORM) {
            val example = data?.getSerializableExtra( name: "EXAMPLE") as Example
            this.lista.add(example)
            (this.rcMain.adapter as RecyclerAdapter).notifyDataSetChanged()
        }
    }
}
```

O aplicativo ficará como na imagem abaixo:

Seguindo esses passos é possível criar um aplicativo simples usando o RecyclerView.