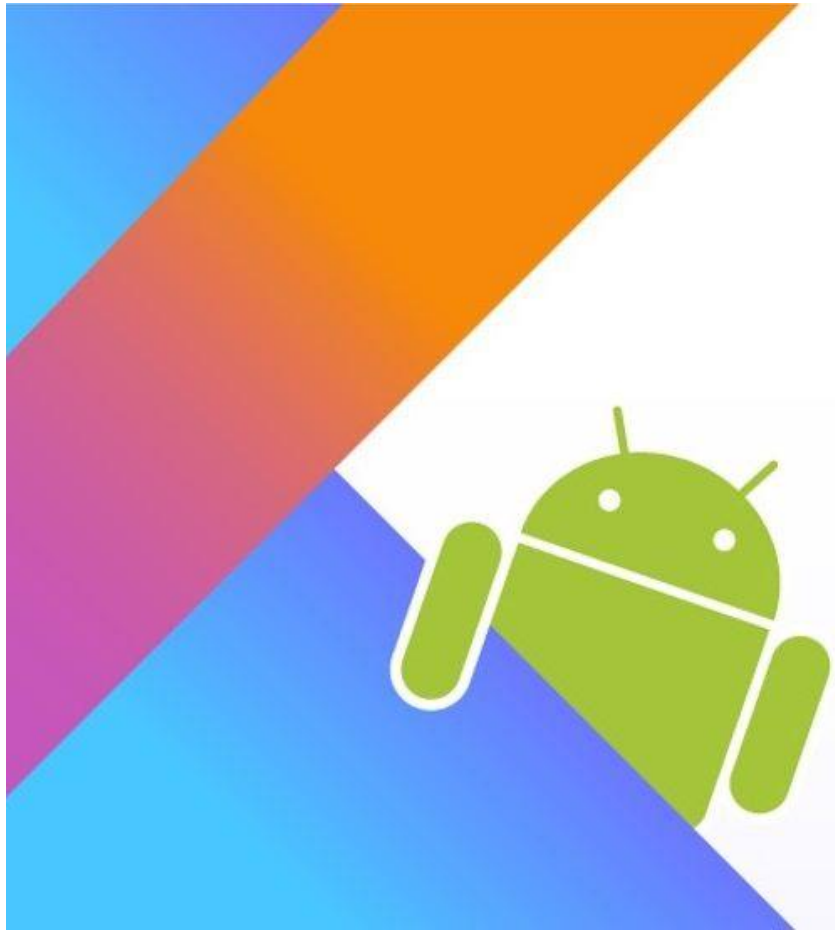
UNI GUAIRACÁ
CENTRO UNIVERSITÁRIO



Kotlin
for Android App Development

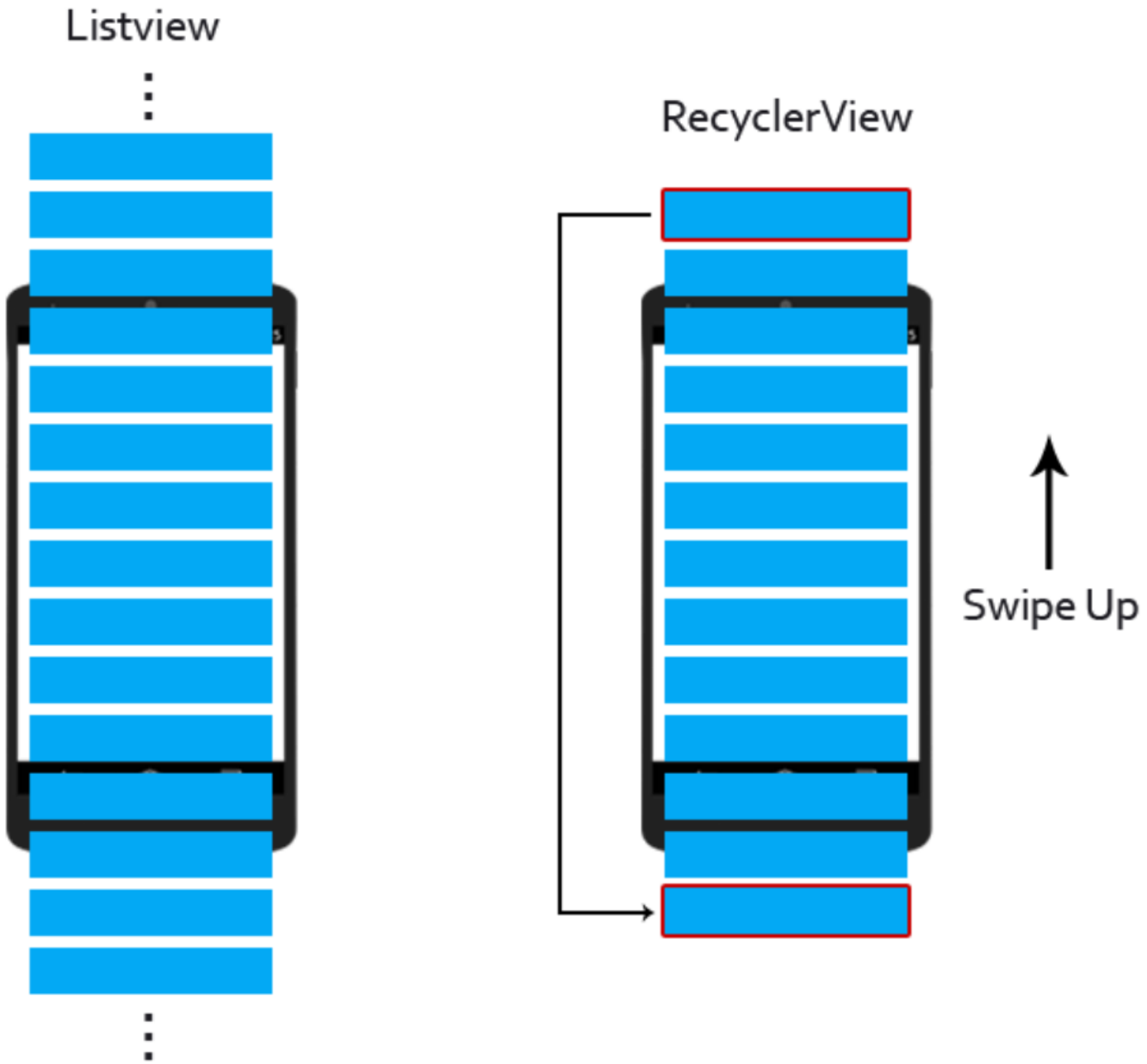
DESENVOLVIMENTO MOBILE

ANDROID – RECYCLER VIEW

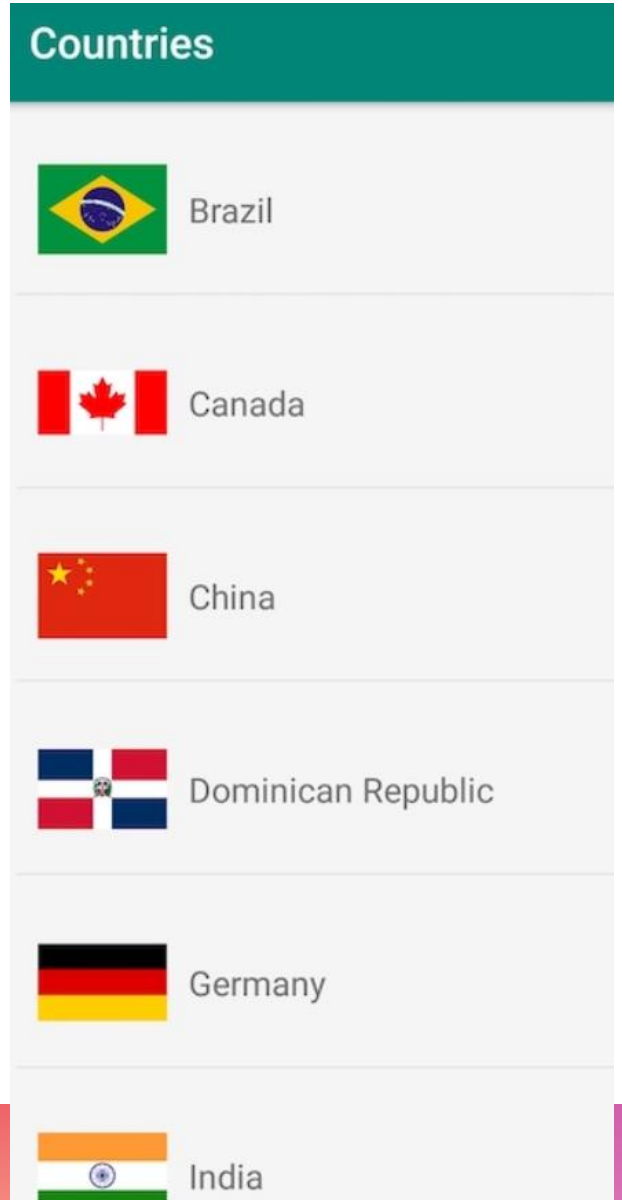
Uma View que representa uma LISTA de elementos.

Nome vêm da ideia de Reaproveitar Objetos Gráficos

ANDRO



ANDROID – RECYCLER VIEW



ANDROID – RECYCLER VIEW

A Lista é feita pelo Trio de Componentes:

- Lista/Recycler View: Representa / Desenha a Lista.

- Item/View Holder : Representa / Desenha um elemento da lista

- Adaptador: Transforma DADOS em ELEMENTOS de lista

ANDROID – RECYCLER VIEW

RecyclerView

Dados

```
mutableListOf(
    País( nome: "Brazil",  bandeira: "R.drawable.Brasil"),
    País( nome: "Canada",  bandeira: "R.drawable.FlagCanada"),
    País( nome: "China",  bandeira: "R.drawable.Mandarim"),
```

Adaptador

View Item

ANDROID – RECYCLER VIEW

A Lista é feita pelo Trio de Componentes:

- Lista/Recycler View: Representa / Desenha a Lista.

- Item/View Holder : Representa / Desenha um elemento da lista

- Adaptador: Transforma DADOS em ELEMENTOS de lista

ANDROID – DADOS

```
package com.example.listaa

data class Afazer(var titulo: String, var descricao: String) {

    companion object {
        var afazeres: MutableList<Afazer> = mutableListOf(
            Afazer( titulo: "Ver Top Gun 2",  descricao: "Filme de Avião"),
            Afazer( titulo: "Escrever TCC",  descricao: "Quero diploma"),
            Afazer( titulo: "Ler \'A Ladra do Demônio\'",  descricao: "O último dos magos 2")
        )

        fun obter(): MutableList<Afazer> {
            return afazeres
        }
    }
}
```

ANDROID – RECYCLER VIEW

A Lista é feita pelo Trio de Componentes:

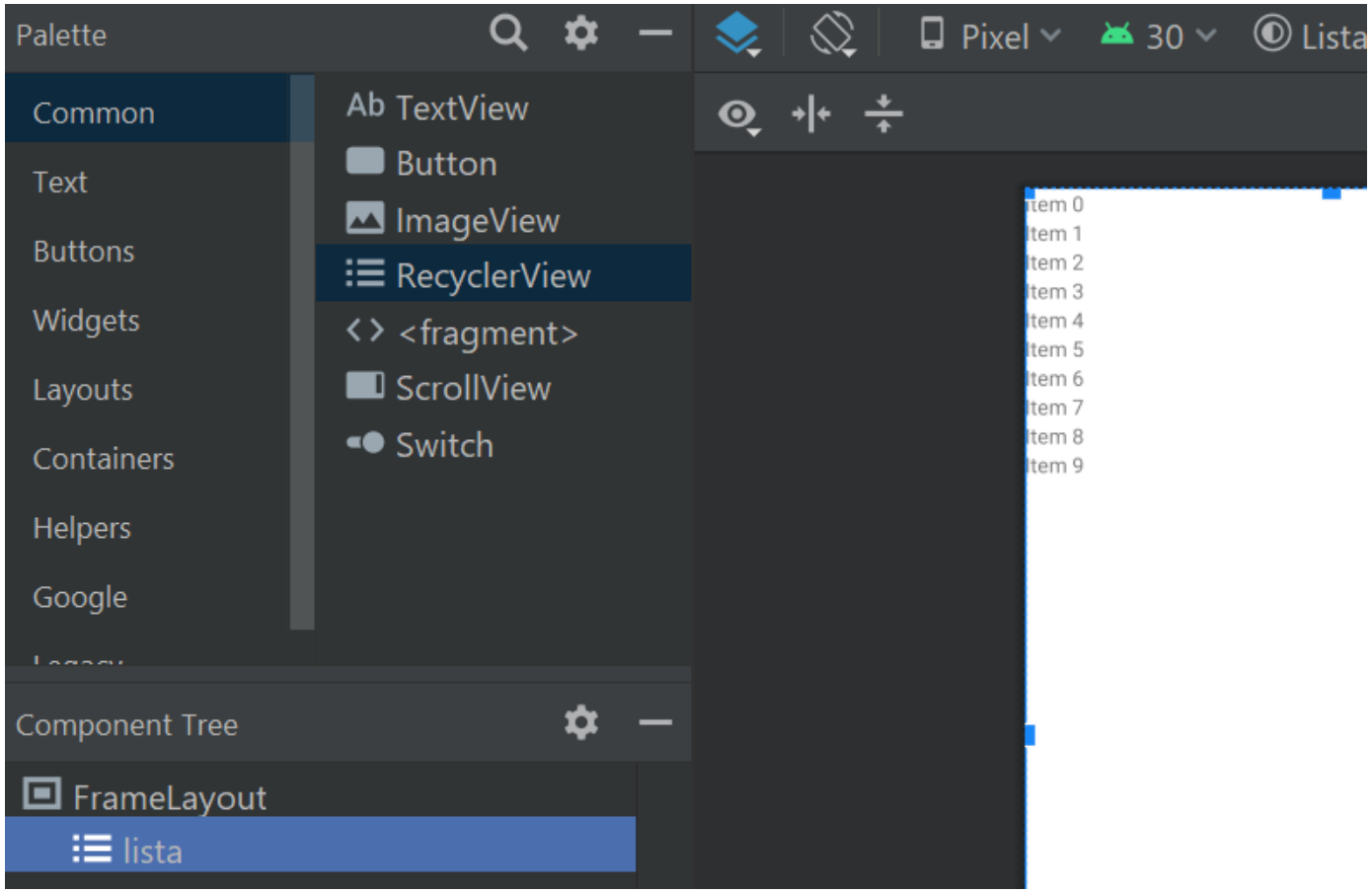
- Lista/Recycler View: Representa / Desenha a Lista.

- Item/View Holder : Representa / Desenha um elemento da lista

- Adaptador: Transforma DADOS em ELEMENTOS de lista

ANDROID – RECYCLER VIEW

Lista/Recycler View: Representa / Desenha a Lista



ANDI

Lista

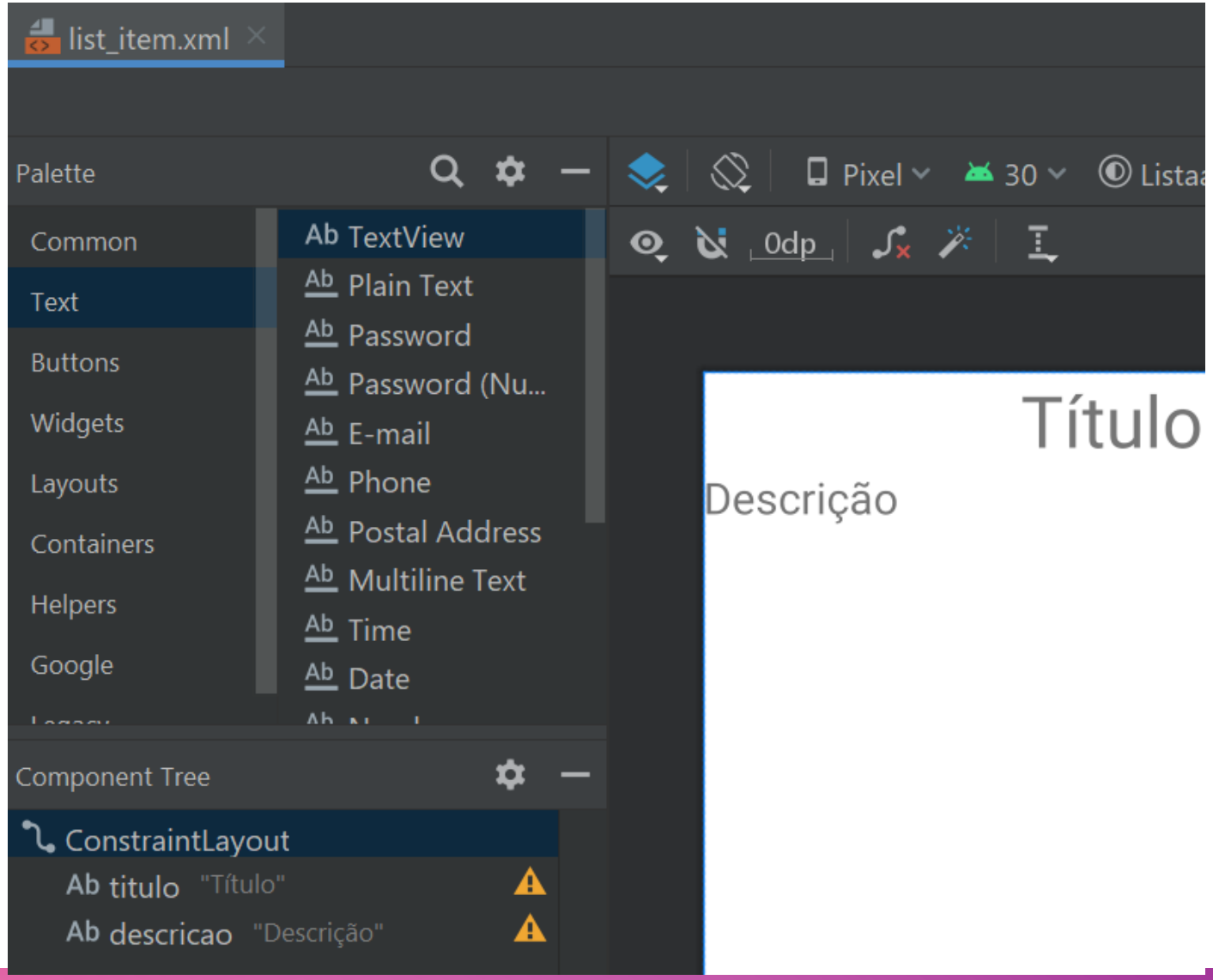
a_tela.xml | ListaTela.kt

```
private lateinit var linearLayoutManager: LinearLayoutManager

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_lista_tela, container,
ttachToRoot: false)
}

override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    linearLayoutManager = LinearLayoutManager(context)
    val recycler: RecyclerView = view.findViewById(R.id.lista)
    recycler.layoutManager = linearLayoutManager
```

ANDROID – RECYCLER VIEW

Layout de um item:

(mude o layout p/ Wrap content)



ANDROID – RECYCLER VIEW

View Holder:

```
class AfazerHolder(v: View, val elementos: MutableList<Afazer>):
RecyclerView.ViewHolder(v), View.OnClickListener {

    val titulo: TextView
    val descricao: TextView

    init {
        titulo = v.findViewById(R.id.titulo)
        descricao = v.findViewById(R.id.descricao)

        v.setOnClickListener(this)
    }

    override fun onClick(clicado: View) {

        val position = getAdapterPosition()
        val elemento = elementos[position]

        Log.d( tag: "RecyclerView",  msg: "${elemento.titulo}")
```

ANDROID – RECYCLER VIEW

Adapter:

```
class ListaAdapter(val elementos: MutableList<Afazer>): RecyclerView
 .Adapter<ListaAdapter.AfazerHolder>() {

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

        return AfazerHolder(view, elementos)
    }

    override fun onBindViewHolder(holder: AfazerHolder, position: Int) {
        holder.titulo.text = elementos[position].titulo
        holder.descricao.text = elementos[position].descricao
    }

    override fun getItemCount() = elementos.size

    class AfazerHolder(v: View, val elementos: MutableList<Afazer>):
```

ANDROID – RECYCLER VIEW

Conectando adaptador:

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    val recycler: RecyclerView = view.findViewById(R.id.lista)

    recycler.layoutManager = LinearLayoutManager(context)
    recycler.adapter = ListaAdapter(Afazer.obter())

    recycler.addItemDecoration(
        DividerItemDecoration(context, DividerItemDecoration.VERTICAL)
    )
}
```

ANDROID – RECYCLER VIEW

View Holder completo (com navegação para detalhes):

ANDRO

View Ho

```
class AFazerHolder(v: View, val dados: MutableList<Afazer>)
    : RecyclerView.ViewHolder(v) {

    val titulo: TextView
    val descricao: TextView
    val apagar: Button

    init {
        titulo = v.findViewById(R.id.titulo)
        descricao = v.findViewById(R.id.descricao)
        apagar = v.findViewById(R.id.apagar)

        titulo.setOnClickListener(::detalhes)
        descricao.setOnClickListener(::detalhes)
    }

    fun detalhes(v: View) {
        val dado = dados[adapterPosition]

        v.findNavController().navigate(R.id.lista_para_detalhes)
    }
```

ANDROID – RECYCLER VIEW

Adicionar e Remover Items da lista

```
fun apagar(position: Int) {
    dados.removeAt(position)
    notifyItemRemoved(position)
    notifyItemRangeChanged(position, itemCount: itemCount - position)
}

fun adicionar(titulo: String) {
    dados.add( index: 0, Afazer(titulo, descricao: "descrição..."))
    notifyItemInserted( position: 0)
    notifyItemRangeChanged( positionStart: 0, itemCount)
}
```

ANDROID – RECYCLER VIEW

Configurar remoção no onBindViewHolder do Adaptador

```
override fun onBindViewHolder(holder: AFazerHolder, position: Int) {
    holder.titulo.text = dados[position].titulo
    holder.descricao.text = dados[position].descricao

    holder.apagar.setOnClickListener {  it: View!
        remover(position)
    }
}
```

ANDROID – RECYCLER VIEW

Configurar adição no Fragmento da Tela da Lista:

```
val novoAfazer: TextView = view.findViewById(R.id.novoAfazer)
val botaoAdicionar: Button = view.findViewById(R.id.adicionar)
botaoAdicionar.setOnClickListener{ it: View!
    adaptador.adicionar(novoAfazer.text.toString())
    hideKeyboard()
}
```

ANDROID – RECYCLER VIEW

Infelizmente é chato esconder o teclado:

A

```
    fun hideKeyboard() {
        val inputManager: InputMethodManager = this.requireActivity()
            .getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager

        // check if no view has focus:
        val currentFocusedView = this.requireActivity().currentFocus
        if (currentFocusedView != null) {
            inputManager.hideSoftInputFromWindow(
                currentFocusedView.windowToken,
                InputMethodManager.HIDE_NOT_ALWAYS
            )
        }
    }
```

ANDROID – EXERCÍCIO

Fazer o Projeto da Lista de Afazeres (TODO List)