RecyclerView é onde vamos colocar imagens dinâmicas, imagina o app da Netflix, eles tem vários filmes, você acha que eles fazem uma por uma ImageView? Não eles criam apenas 1 e ela vai sendo multiplicada.

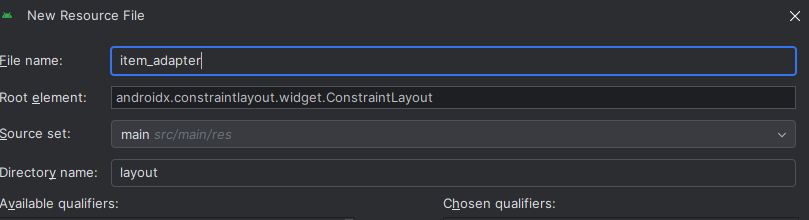
# Adapter

É uma classe onde vamos passar a referencia dos nossos itens para cada imagem do RecycleView

Ex: imagemSenhorDosAneis vai passar para o RecyclerView 1

HarryPotter vai passar para o RecycleView 2

# 1 Criando o Layout

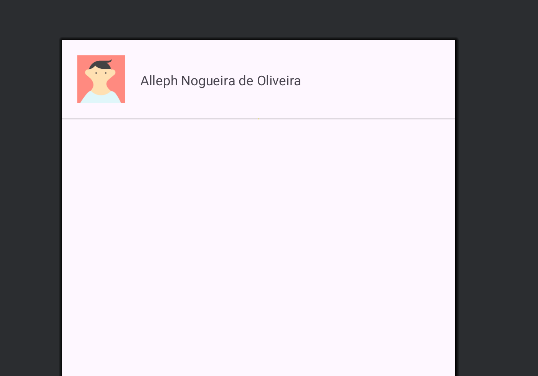


Vamos na pasta Res – Layout e vamos criar esse item, onde vamos fazer nosso Layout

Importante e sempre usar a palavra filme\_adapter... exemplo

# 2 Montando o layout (item\_adapter)

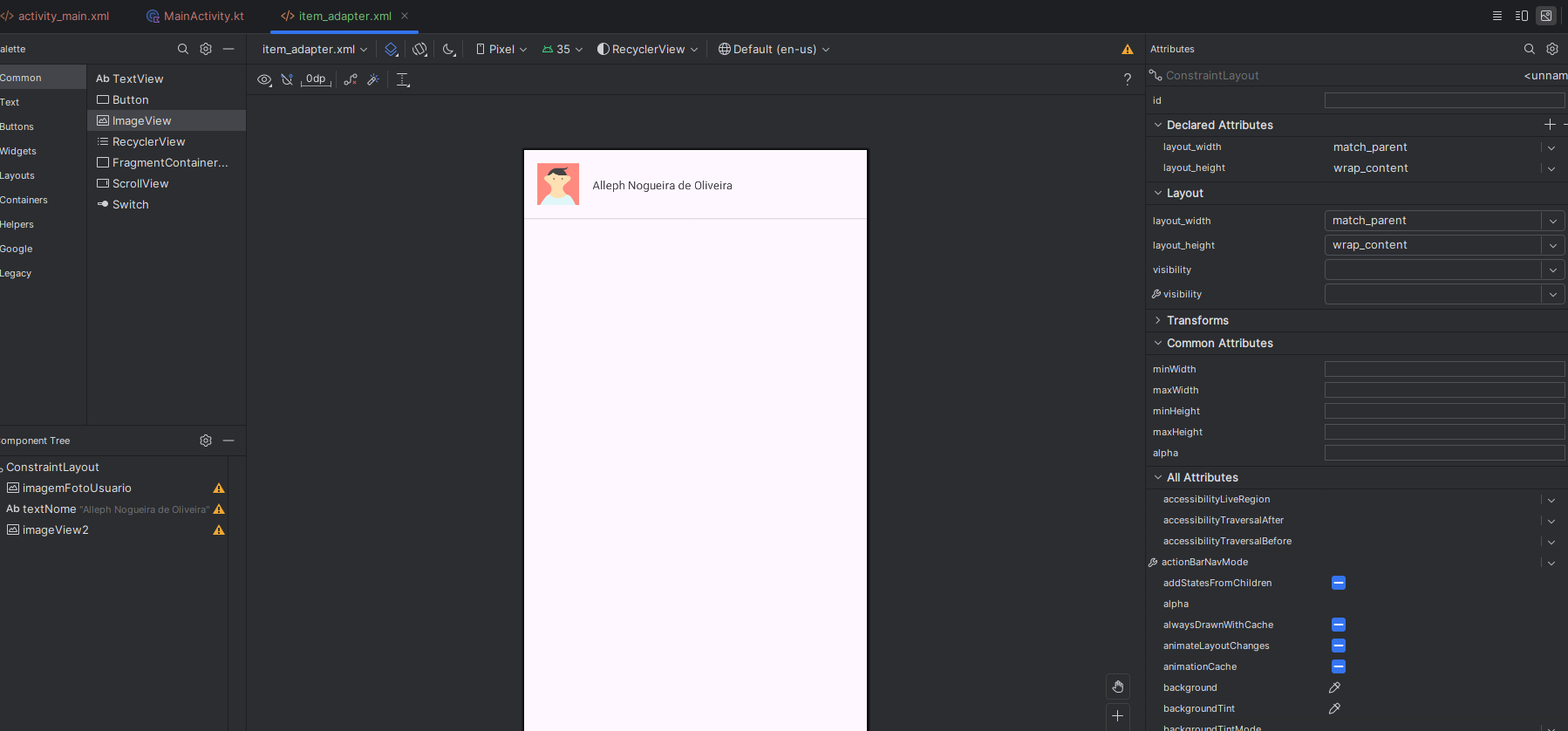
<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"> <!-- ATENÇÃO: Aqui tem que ser assim para ele ocupar apenas o espaçamento necessario -->  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="50dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 tools:srcCompat="@tools:sample/avatars" />  
  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginTop="16dp"  
 android:layout\_marginEnd="16dp"  
 android:text="Alleph Nogueira de Oliveira"  
 app:layout\_constraintBottom\_toBottomOf="@+id/imageView"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/imageView"  
 app:layout\_constraintTop\_toTopOf="@+id/imageView" />  
  
 <ImageView  
 android:id="@+id/imageView2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:layout\_marginTop="16dp"  
 android:background="?actionBarDivider"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView"  
 tools:srcCompat="@tools:sample/avatars" />  
</androidx.constraintlayout.widget.ConstraintLayout>



# 3 Detalhes

Agora como vamos trabalhar de forma dinâmica, eles precisam ter IDS

Todos os itens que vao ser modificados precisam ter IDS



# 4 Criando a classe adpter

Vamos criar uma classe em Kotlin

Lembrar que o ideal e você da o nome para ele, exemplo Netflix

AdapterFilmes

Ele vai ser responsável em adaptar a tela com imagens dinâmicas, para os filmes.

Primeiro vamos dizer que aquela classe e do tipo Adapter

package com.allephnogueira.recyclerview  
  
import androidx.recyclerview.widget.RecyclerView  
  
class Adapter () :RecyclerView.Adapter <> {  
}

package com.allephnogueira.recyclerview  
  
import android.view.View  
import android.view.ViewGroup  
import androidx.recyclerview.widget.RecyclerView  
  
  
// Vamos criar uma classe do tipo RecycleView.Adapter  
// Vamos passar < O nome da classe que criamos em baixo >  
class Adapter () : RecyclerView.Adapter<Adapter.MyViewHolder>() {  
  
 // Aqui vamos trabalhar com oque? Nesse exemplo vamos trabalhar com Strings, mas poderiamos trabalhar com objetos também, como os filmes que chegam..  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {  
 // Aqui é responsavel pro criar o Layout de cada linha que nos temos.  
 *TODO*("Not yet implemented")  
 }  
  
 override fun getItemCount(): Int {  
 // Aqui é onde vamos passar o total de itens que ele tem que criar, exemplo vamos dizer que queremos criar 10 filmes  
  
 *TODO*("Not yet implemented")  
 }  
  
 override fun onBindViewHolder(holder: MyViewHolder, position: Int) {  
 // Aqui é o metodo que exibimos as informações (Nome do filme, foto do filme...)  
 // Aqui que vamos passar nossas informações para la.  
 *TODO*("Not yet implemented")  
 }  
  
  
 // Agora vamos criar a class (class que vamos utilizar la em cima no tipo do Recycle  
 class MyViewHolder(itemView: View) : RecyclerView.ViewHolder (itemView) {}  
}

# 5 Passando os dados para o Adapter

Observa que os dados são passados dentro do construtor da classe

class Adapter (  
 // Aqui nossa variavel vai ser do tipo List e a lista vai ser com entrada de Strings.  
 private val myList: List<String>  
) : RecyclerView.Adapter<Adapter.MyViewHolder>() {

# 6 Passando a quantidade de itens que vamos renderizar com as fotos

override fun getItemCount(): Int {  
 // Aqui é onde vamos passar o total de itens que ele tem que criar, exemplo vamos dizer que queremos criar 10 filmes  
   
 return myList.size  
}

Aqui ele vai pegar a quantidade de itens da lista e vai criar imagens com essa quantidade

Vamos supor que esta chegando da api 300 filmes, ele vai calcular e vai criar esse tanto de filmes.

# 7 Agora vamos puxar os itens que queremos modificar

class MyViewHolder(itemView: View) : RecyclerView.ViewHolder (itemView) {  
 // Aqui vamos colocar os componentes que temos no nosso layout  
 // \*\*\* repara que estamos usando a classe itemView que tem os metodos de pegar os dados da view \*\*\*  
 val nomeUsuario : TextView = itemView.findViewById(R.id.*textNome*)  
}

# 8 Criando as linhas de código responsável por modificar os elementos na tela

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {  
 // Aqui é responsavel pro criar o Layout de cada linha que nos temos.  
 // Agora aqui vamos criar as linhas de codigo responsavel por modificar cada elemento da tela  
  
 // aqui no inflate temos que passar 3 informaçoes  
 // 1 o xml que criamos  
 // 2 View Group, que podemos passar o parent  
 // Terceiro podemos passar um false  
  
 val itemView = LayoutInflater.from(parent.*context*).inflate(R.layout.*item\_adapter*, parent, false)  
 return MyViewHolder(itemView)  
}

# 9 Passando os dados como referencia

override fun onBindViewHolder(holder: MyViewHolder, position: Int) {  
 // Aqui é o metodo que exibimos as informações (Nome do filme, foto do filme...)  
 // Aqui que vamos passar nossas informações para la.  
  
 val nome = myList[position]  
  
   
 holder.nomeUsuario.*text* = nome  
}

# 10 Tomada de decisão quando o usuário clicar em cima de cada item no RecycleView

Temos que ativar o ViewBinding

viewBinding **{** enable = true  
 **}  
}***dependencies* **{** *implementation*(*libs*.*androidx*.*core*.*ktx*)  
 *implementation*(*libs*.*androidx*.*appcompat*)  
 *implementation*(*libs*.*material*)  
 *implementation*(*libs*.*androidx*.*activity*)  
 *implementation*(*libs*.*androidx*.*constraintlayout*)  
 *testImplementation*(*libs*.*junit*)  
 *androidTestImplementation*(*libs*.*androidx*.*junit*)  
 *androidTestImplementation*(*libs*.*androidx*.*espresso*.*core*)  
**}**

# 11 Configuração do viewBinding

package com.allephnogueira.recyclerview  
  
import android.app.Activity  
import android.os.Bundle  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import com.allephnogueira.recyclerview.databinding.ActivityMainBinding  
  
class MainActivity : AppCompatActivity() {  
  
 //\*\*\* configuração inicial  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(*layoutInflater*) // \*\*\*\* configuração do binding  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*) //\*\*\*  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** }  
   
   
   
}

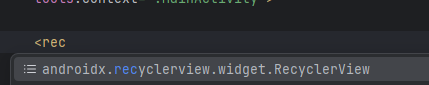
# 12 Metodo para chamar o recycle

class MainActivity : AppCompatActivity() {  
  
 //\*\*\* configuração inicial  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(*layoutInflater*) // \*\*\*\* configuração do binding  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*) //\*\*\*  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** }  
  
 private fun initRecycleView() {  
 // layoutManager é a forma da gente exibir o recycleView  
 // Vamos colocar para ele exibir um a baixo do outro.  
 // Também temos o grid que podemos dividir os itens em coluna  
 binding.recyclerView.*layoutManager* = LinearLayoutManager(this)  
 // Isso aqui é para ele gerar uma performance melhor no carregamento.  
 binding.recyclerView.setHasFixedSize(true)  
 // Aqui ele espera receber uma lista de String que foi oque passamos la.  
 binding.recyclerView.*adapter* = Adapter(getList())  
 }  
  
  
 private fun getList() = *listOf* ("Alleph", "Fernanda", "Crixus", "Amora", "Anastacia", "Calopsita Pai", "Calopsita Mae", "Zeca", "Bethoven")  
// Criamos esse método com essa lista apenas para fazer de exemplo  
}

# 13 Configurando o recycleView no layout

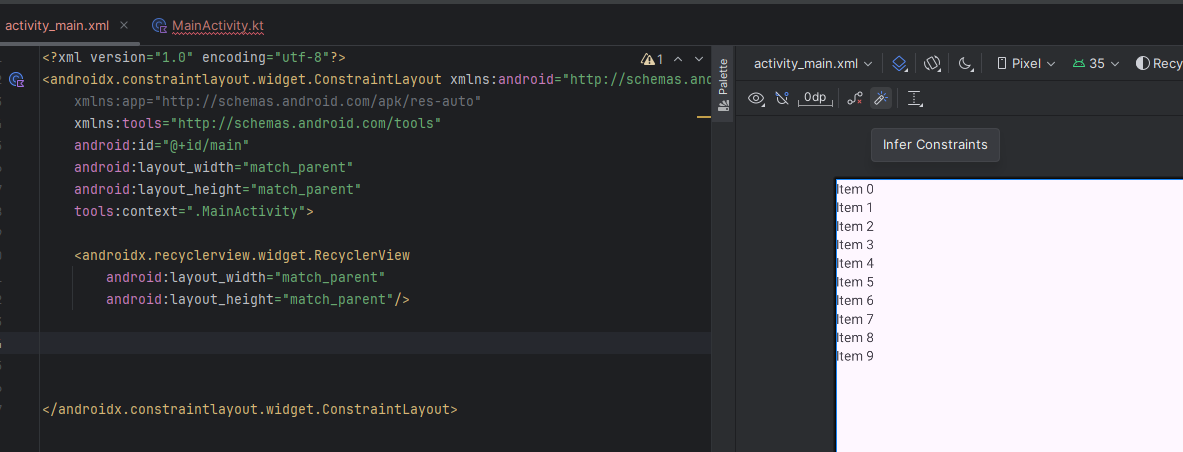
Vamos la no layout da pagina (O principal)

\*\*\*\* Atenção no pacote que vai ser usado



Para ele ocupar toda a tela vamos colocar em

<androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recyclerView"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />



Essa opção infer Constr... serve para ele fazer as regras de layout automaticamente.

# 14 – Chamando os dados no onCreate para ele criar.

Pode reparar que chamamos o método que criamos com a lista.

class MainActivity : AppCompatActivity() {  
  
 //\*\*\* configuração inicial  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(*layoutInflater*) // \*\*\*\* configuração do binding  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*) //\*\*\*  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** initRecycleView()  
 }  
  
 private fun initRecycleView() {  
 // layoutManager é a forma da gente exibir o recycleView  
 // Vamos colocar para ele exibir um a baixo do outro.  
 // Também temos o grid que podemos dividir os itens em coluna  
 binding.recyclerView.*layoutManager* = LinearLayoutManager(this)  
 // Isso aqui é para ele gerar uma performance melhor no carregamento.  
 binding.recyclerView.setHasFixedSize(true)  
 // Aqui ele espera receber uma lista de String que foi oque passamos la.  
 binding.recyclerView.*adapter* = Adapter(getList())  
 }  
  
  
 private fun getList() = *listOf* ("Alleph", "Fernanda", "Crixus", "Amora", "Anastacia", "Calopsita Pai", "Calopsita Mae", "Zeca", "Bethoven")  
  
}

# 15 Evento de click no nosso RecycleView

Vamos entrar no nosso Adapter e vamos dizer qual tipo de dado que queremos retornar

class Adapter (  
 // Vamos retornar o nome do Usuario que vai vir do tipo String  
 val retornoNomeUsuario : (String) -> Unit,

override fun onBindViewHolder(holder: MyViewHolder, position: Int) {  
 // Aqui é o metodo que exibimos as informações (Nome do filme, foto do filme...)  
 // Aqui que vamos passar nossas informações para la.  
  
 val nome = myList[position]  
 holder.textName.*text* = nome  
  
 // Aqui é onde vamos tratar os eventos de click  
 // Vamos pegar o holder  
 // O nome do componente que queremos alterar  
 // Aqui vamos passar o nome que colocamos la como construtor da classe.  
 // E vamos passar o nosso nome mesmo, no caso se ele clicar em alleph, vamos passar o alleph pra ele.  
 // Aqui ele so vai habilitar o evento de click se o usuario clicar em cima do componente de textView no caso (textName)  
 /\* holder.textName.setOnClickListener{retornoNomeUsuario(nome)} \*/  
   
 // Podemos fazer para habilitar o evento quando o usuario clicar em cima de qualquer local do componente  
 holder.itemView.setOnClickListener**{**retornoNomeUsuario(nome)**}**}

# 16 Atenção vai gerar um erro no MainActivity

É porque temos que respeitar a ordem, colocamos o evento de click antes da lista, e nos parâmetros la a lista esta passando primeiro, sendo que o evento de click teria que vir primeiro, então vamos só alterar a ordem.

class Adapter (  
 private val myList: List<String>, // Aqui nossa variavel vai ser do tipo List e a lista vai ser com entrada de Strings.  
 val retornoNomeUsuario : (String) -> Unit // Vamos retornar o nome do Usuario que vai vir do tipo String

# 17 Passando o parâmetro para a função

private fun initRecycleView() {  
 // layoutManager é a forma da gente exibir o recycleView  
 // Vamos colocar para ele exibir um a baixo do outro.  
 // Também temos o grid que podemos dividir os itens em coluna  
 binding.recyclerView.*layoutManager* = LinearLayoutManager(this)  
 // Isso aqui é para ele gerar uma performance melhor no carregamento.  
 binding.recyclerView.setHasFixedSize(true)  
 // Aqui ele espera receber uma lista de String que foi oque passamos la.  
 binding.recyclerView.*adapter* = Adapter(getList()) **{** nome **->** // Isso aqui a gente chama de CallBack é um ouvinte.  
 Toast.makeText(this, "$nome", Toast.*LENGTH\_SHORT*).show() // Aqui vamos passar a string que recebemos   
 **}**}

Toast e para gerar uma mensagem na tela, no caso ai a mensagem que vai vir e o próprio nome do usuário.

Mas poderíamos também passar o nome do filme para exibir detalhes dele na próxima pagina.

Poderíamos passar a informação para outra acitivity

# BONUS ERRO ANDROID 8 PARA O 11

Atenção se der o erro pedindo para você mudar o projeto para java 11 voce pode ir em, file – Project Structure - SDK Location – Gradle settings – ai muda para o JDK do android 11

# CODIGO COMPLETO ADAPTER CLASS

package com.allephnogueira.recyclerview  
  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
  
  
// Vamos criar uma classe do tipo RecycleView.Adapter  
// Vamos passar < O nome da classe que criamos em baixo >  
class Adapter (  
 private val myList: List<String>, // Aqui nossa variavel vai ser do tipo List e a lista vai ser com entrada de Strings.  
 val retornoNomeUsuario : (String) -> Unit // Vamos retornar o nome do Usuario que vai vir do tipo String  
  
  
) : RecyclerView.Adapter<Adapter.MyViewHolder>() {  
  
  
 // Aqui vamos trabalhar com oque? Nesse exemplo vamos trabalhar com Strings, mas poderiamos trabalhar com objetos também, como os filmes que chegam..  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {  
 // Aqui é responsavel pro criar o Layout de cada linha que nos temos.  
 // Agora aqui vamos criar as linhas de codigo responsavel por modificar cada elemento da tela  
  
 // aqui no inflate temos que passar 3 informaçoes  
 // 1 o xml que criamos  
 // 2 View Group, que podemos passar o parent  
 // Terceiro podemos passar um false  
  
 val itemView = LayoutInflater.from(parent.context).inflate(R.layout.item\_adapter, parent, false)  
 return MyViewHolder(itemView)  
 }  
  
 override fun getItemCount(): Int {  
 // Aqui é onde vamos passar o total de itens que ele tem que criar, exemplo vamos dizer que queremos criar 10 filmes  
  
 return myList.size  
 }  
  
 override fun onBindViewHolder(holder: MyViewHolder, position: Int) {  
 // Aqui é o metodo que exibimos as informações (Nome do filme, foto do filme...)  
 // Aqui que vamos passar nossas informações para la.  
  
 val nome = myList[position]  
 holder.textName.text = nome  
  
 // Aqui é onde vamos tratar os eventos de click  
 // Vamos pegar o holder  
 // O nome do componente que queremos alterar  
 // Aqui vamos passar o nome que colocamos la como construtor da classe.  
 // E vamos passar o nosso nome mesmo, no caso se ele clicar em alleph, vamos passar o alleph pra ele.  
 // Aqui ele so vai habilitar o evento de click se o usuario clicar em cima do componente de textView no caso (textName)  
 /\* holder.textName.setOnClickListener{retornoNomeUsuario(nome)} \*/  
  
 // Podemos fazer para habilitar o evento quando o usuario clicar em cima de qualquer local do componente  
 holder.itemView.setOnClickListener**{**retornoNomeUsuario(nome)**}** }  
  
  
 // Agora vamos criar a class (class que vamos utilizar la em cima no tipo do Recycle  
 class MyViewHolder(itemView: View) : RecyclerView.ViewHolder (itemView) {  
 // Aqui vamos colocar os componentes que temos no nosso layout  
 // \*\*\* repara que estamos usando a classe itemView que tem os metodos de pegar os dados da view \*\*\*  
 val textName : TextView = itemView.findViewById(R.id.textName)  
 // NOME DO COMPONENTE (textName)  
 }  
}

# CODIGO COMPLETO MAINACTIVITY

package com.allephnogueira.recyclerview  
  
import android.app.Activity  
import android.os.Bundle  
import android.widget.Toast  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.view.ViewCompat  
import androidx.core.view.WindowInsetsCompat  
import androidx.recyclerview.widget.LinearLayoutManager  
import com.allephnogueira.recyclerview.databinding.ActivityMainBinding  
  
class MainActivity : AppCompatActivity() {  
  
 //\*\*\* configuração inicial  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 binding = ActivityMainBinding.inflate(*layoutInflater*) // \*\*\*\* configuração do binding  
 *enableEdgeToEdge*()  
 setContentView(binding.*root*) //\*\*\*  
 ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.*main*)) **{** v, insets **->** val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)  
 insets  
 **}** initRecycleView()  
 }  
  
 private fun initRecycleView() {  
 // layoutManager é a forma da gente exibir o recycleView  
 // Vamos colocar para ele exibir um a baixo do outro.  
 // Também temos o grid que podemos dividir os itens em coluna  
 binding.recyclerView.*layoutManager* = LinearLayoutManager(this)  
 // Isso aqui é para ele gerar uma performance melhor no carregamento.  
 binding.recyclerView.setHasFixedSize(true)  
 // Aqui ele espera receber uma lista de String que foi oque passamos la.  
 binding.recyclerView.*adapter* = Adapter(getList()) **{** nome **->** // Isso aqui a gente chama de CallBack é um ouvinte.  
 Toast.makeText(this, "$nome", Toast.*LENGTH\_SHORT*).show() // Aqui vamos passar a string que recebemos  
 **}** }  
  
  
 private fun getList() = *listOf* ("Alleph", "Fernanda", "Crixus", "Amora", "Anastacia", "Calopsita Pai", "Calopsita Mae", "Zeca", "Bethoven"  
 ,"Alleph", "Fernanda", "Crixus", "Amora", "Anastacia", "Calopsita Pai", "Calopsita Mae", "Zeca", "Bethoven")  
  
}