

Vamos usar a material toolbar porque ela tem mais recursos



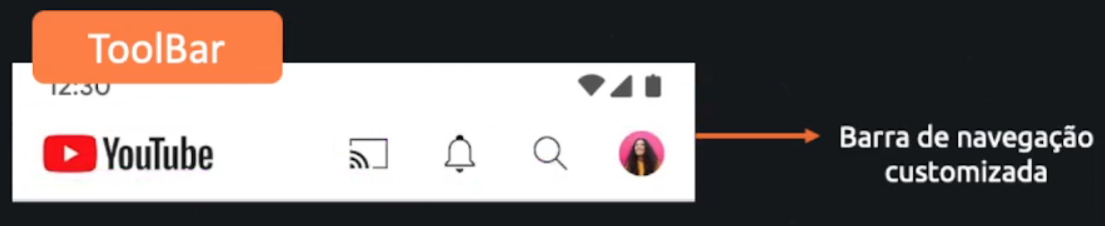
# Criando a toolbar

lembrar que ela é um método que você deve fazer igual a actionBar

Vamos adicionar ela na nossa tela, colocar um ID para ela e passar os dados e depois chamar ela no método setSupportActionBar(binding.tbPrincipal)

class MainActivity : AppCompatActivity() {  
  
  
 private val binding by *lazy* **{** ActivityMainBinding.inflate(*layoutInflater*) **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *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  
 **}** inicializarToolBar()  
  
 }  
  
 private fun inicializarToolBar() {  
 */\*\* Esse metodo serve para configurar nossa barra para diferentes versoes  
 \*  
 \* A baixo vamos configurar os dados como nome, subtitulo e apos vamos inicialiazar a toolbar  
 \*  
 \*/* binding.tbPrincipal.setTitleTextColor(ContextCompat.getColor(this, R.color.*white*)) /\* Configuração da cor \*/  
 binding.tbPrincipal.*title* = "Youtube"  
 binding.tbPrincipal.*subtitle* = "Videos a todo momento"  
  
  
 */\*\* Nesse metodo vamos passar nossa toolbar e vamos configurar ela como se fosse uma ActionBar  
 \*  
 \*/* setSupportActionBar(binding.tbPrincipal)  
 }  
  
  
}

# Customizando uma ToolBar



# Configurando um tema para o 3:

Abrir o themas.xml

Aqui estamos criando um stylo, que herda(parente) e após isso adiciona uma cor

<style name="actionOverflowButton" parent="Widget.AppCompat.ActionButton.Overflow">  
 <item name="tint">@color/white</item>  
</style>

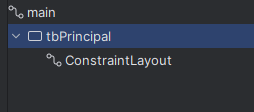
Agora sim podemos chamar nosso novo Stylo que fizemos

<resources xmlns:tools="http://schemas.android.com/tools">  
 <!-- Base application theme. -->  
 <style name="Base.Theme.Toolbar" parent="Theme.Material3.DayNight.NoActionBar">  
 <!-- Customize your light theme here. -->  
 <!-- <item name="colorPrimary">@color/my\_light\_primary</item> -->  
   
 <item name="actionOverflowButtonStyle">@style/actionOverflowButton</item>

Outra maneira de fazer é utilizando a ToolBar como actionBar, ai vamos chamar o método e passar ela, como pode ver no código marcado, e ai podemos fazer todo passo a passo da aula anterior, que é customizando uma actionBar

# Customizando a ToolBar

Primeiro vamos colocar um constraint layout dentro da nossa toolbar, para poder personalizar ele.

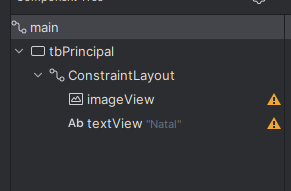


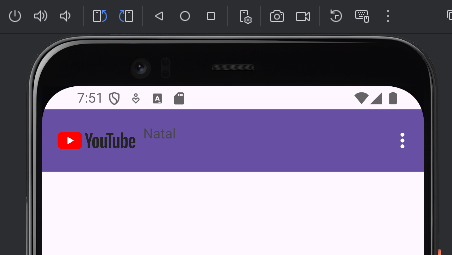
Atençao: Devemos ir em code e fazer manualmente ficando dessa forma

<com.google.android.material.appbar.MaterialToolbar  
 android:id="@+id/tbPrincipal"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:background="?attr/colorPrimary"  
 android:elevation="4dp"  
 android:minHeight="?attr/actionBarSize"  
 android:theme="?attr/actionBarTheme"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" >  
  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
  
  
  
</com.google.android.material.appbar.MaterialToolbar>

Observa que o constraint layout esta dentro da nossa toolbar

Repara como ficou





Aqui poderíamos adicionar mais botões, e fazer infinitas coisas, ate um linearlayout também.

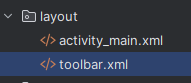
Atenção, colocamos tudo dentro do nosso ConstraintLayout.

# Separando a toolbar para usar em outros arquivos.

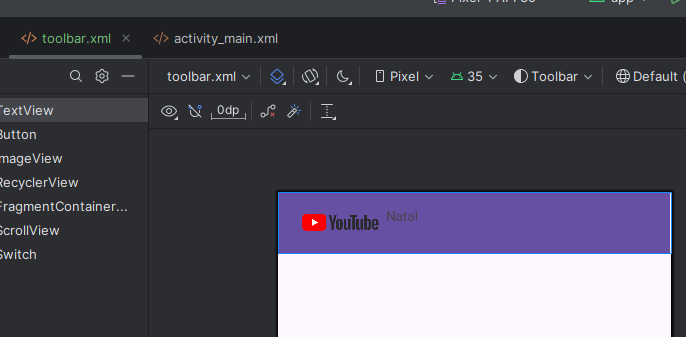
Imagina que temos uma toolbar e queremos utilizar em 2 activity, então vamos separar ela e vamos utilizar sempre que quiser.

1 vamos em layout e vamos criar um novo layout

Podemos chamar ele de toolbar

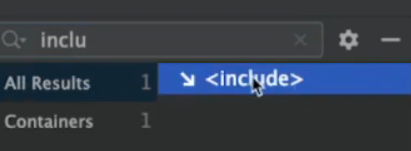


Repara que isolamos, criamos um toolbar para poder utilizar em outros locais

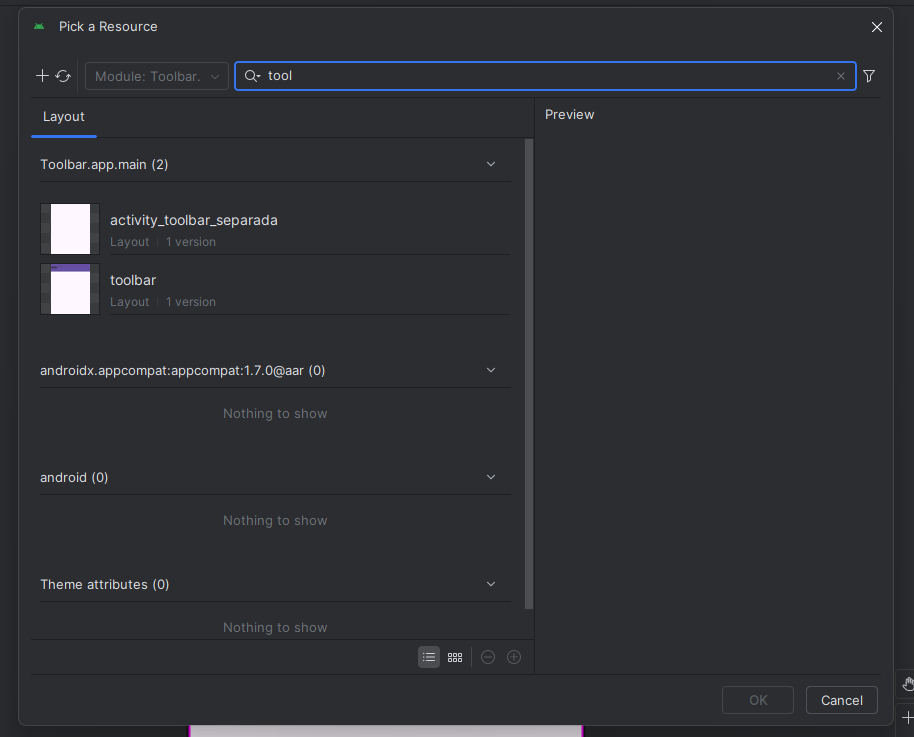


2 Vamos criar uma nova activity(activity que vamos importar a toolbar separada)

3 Vamos no XML da nova activity e vamos la nos pallets e vamos procurar por include



Vai abrir essa tela, aqui podemos usar o recurso que quiser, no nosso caso vamos utilizar nosso toolbar.xml



E ai precisamos também da um id para ela, para você poder chamar em qualquer local.

# Navegação

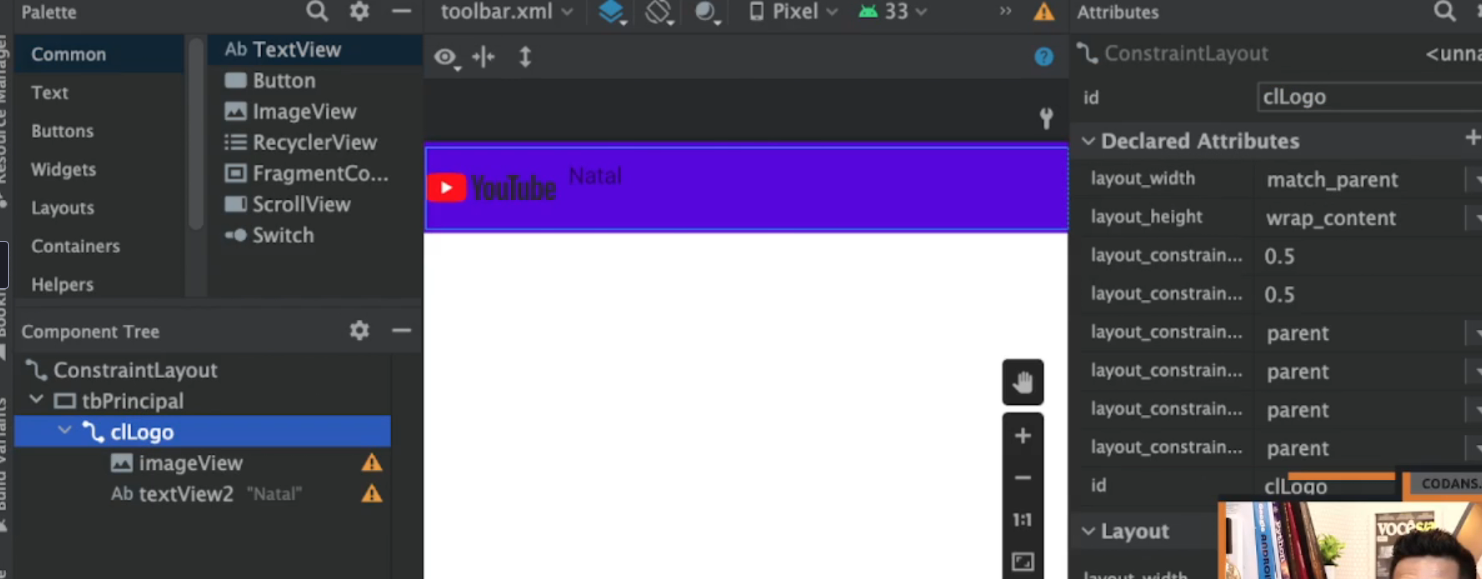
Imagina que estamos em uma tela, navegamos para a outra e na outra tela exibimos o botão de voltar.

1 configurar um botão para isso

Esse botão vai jogar a gente para uma nova activity

private fun inicializarToolBar() {  
  
 binding.btnToolbarSeparada.setOnClickListener **{** */\*\* Abrir uma nova activity quando for clicado \*/* val intent = Intent(this, ToolbarSeparada::class.*java*)  
 startActivity(intent)  
 **}**

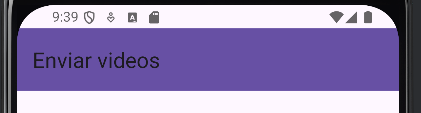
2 no nosso arquivo de Toolbar, adicionamos um id para o Constraint Layout da toolbar

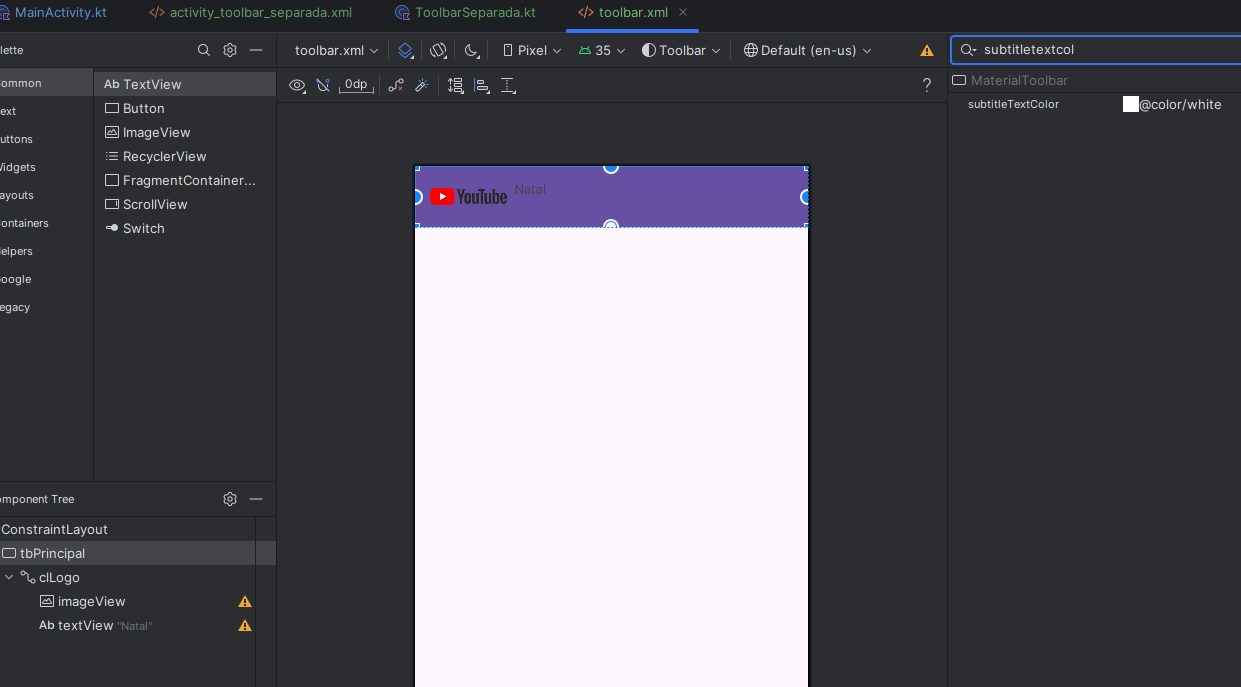


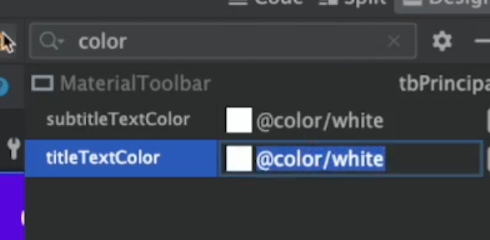
3 dentro da nossa nova activity configuramos tudo novamente, a inicialização da toolbar e também os dados que vai ser carregadas nelas.

inicializarToolBar()  
  
 }  
  
 private fun inicializarToolBar() {  
  
 binding.includeToolbar.clLogo.*visibility* = View.*GONE* // Nao quero que apareça o toolbar que definimos no XML  
  
  
 binding.includeToolbar.tbPrincipal.*title* = "Enviar videos"  
  
  
  
 setSupportActionBar(binding.includeToolbar.tbPrincipal) // Exibir a nossa toolbar como actionBar  
  
 }  
}

4 Mudando as cores da toolbar





Agora fomos direto no nosso arquivo XML e mudamos a cor por la. 

4 dizendo que toda vez que o usuario clicar no botao voltar ele vai retornar para a tela PAI

<activity  
 android:name=".ToolbarSeparada"  
 android:parentActivityName=".MainActivity"  
 android:exported="false" />  
<activity

Vai ficar dessa forma, toda vez que clicar no botão voltar ele vai retornar para o PAI(parente)

Resumindo o android vai entender que quando você pressionar o botão de voltar ele vai voltar para a tela que definimos ai em cima.

5 precisamos configurar o botão de voltar

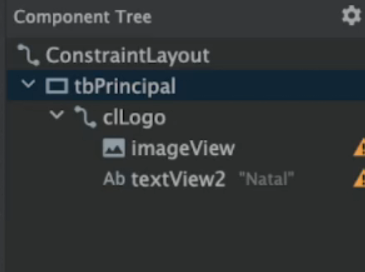
private fun inicializarToolBar() {  
  
 binding.includeToolbar.clLogo.*visibility* = View.*GONE* // Nao quero que apareça o toolbar que definimos no XML  
  
  
 binding.includeToolbar.tbPrincipal.*title* = "Enviar videos"  
  
  
  
 setSupportActionBar(binding.includeToolbar.tbPrincipal) // Exibir a nossa toolbar como actionBar  
  
  
 *supportActionBar*?.setDisplayHomeAsUpEnabled(true) // habilitar o botao de voltar para a activity inferior  
  
}



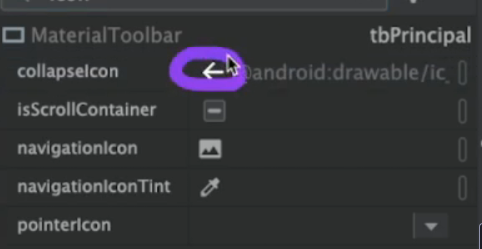
Resumindo ativamos o botão de voltar, e no androidmanifest adicionamos para onde queremos que ele volte por padrão.

6 configurando a cor do botão de voltar

Agora para configurar algo, vamos sempre na toolbar.xml que é nossa toolbar separada e la podemos configurar tudo que vai carregar igual em todas as paginas



Icone



Em baixo temos a cor do ícone

