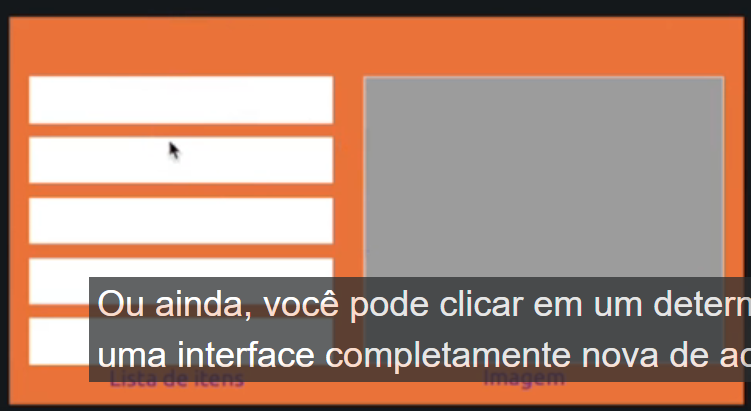
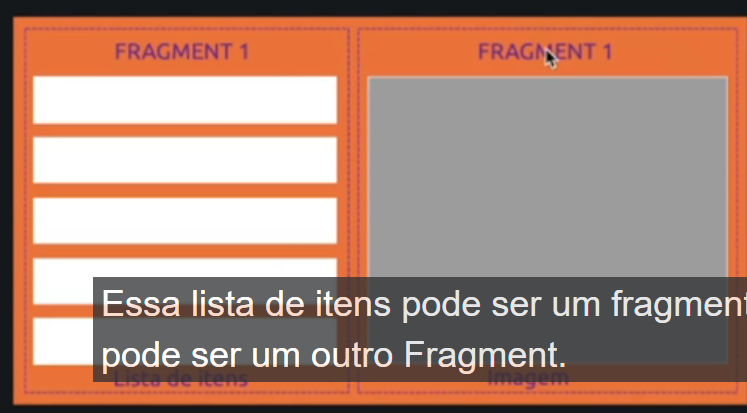
# Fragment o que é?

* Representa uma parte reutilizável da interface de usuário do seu App.
* Uma fragmente define e gerencia o próprio layout, tem o próprio clico de vida e pode processar eventos de entrada.
* Fragments não existem por conta própria, eles precisam ser hospedados uma atividade ou por outro fragmento.



Imagina que clicamos no primeiro quadrado brando e carregamos uma imagem no outro lado.

Ou ainda você pode clicar em um menu e carregar uma interface totalmente nova.



Nessa imagem temos 2 fragments

# Exemplos de utilização de fragments

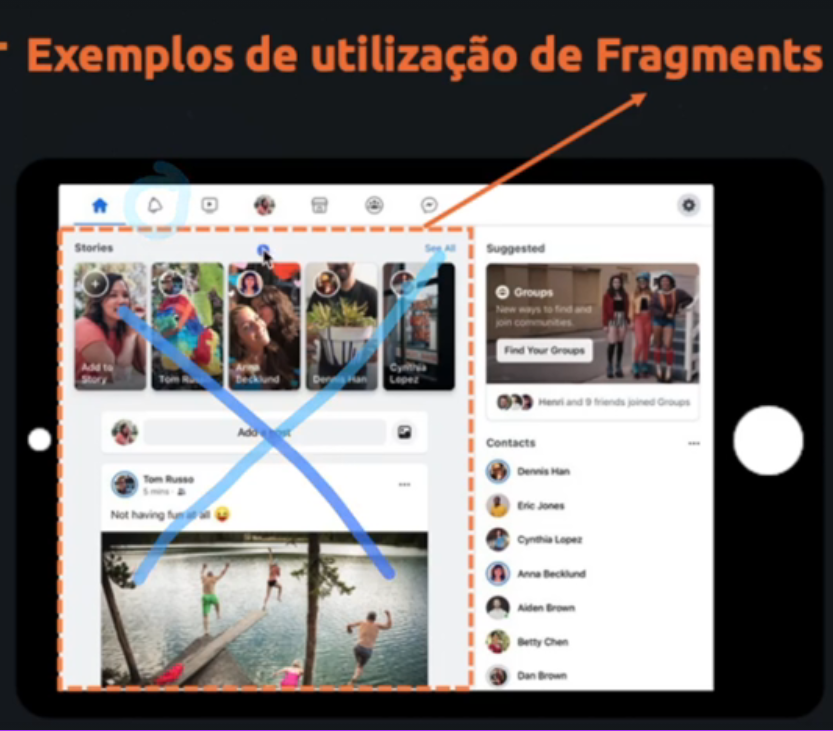
Segue o exemplo do whatsapp, temos as abas e em baixo dependendo do item que você clicar na aba, ele vai ter o fragment em baixo



Se clicar em conversa, abre o fragment da conversa, se clicar em status abrimos outro fragment

Moral da história, o fragment é ótimo para você não precisar fazer uma nova activity toda vez que você clicar em algo, e sim abrir um novo fragment.

Outro exemplo



# Como criar um Fragment

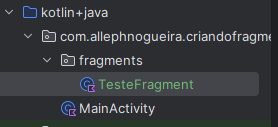
Vamos criar a estrutura de fragment

* Usando XML
* Programaticamente

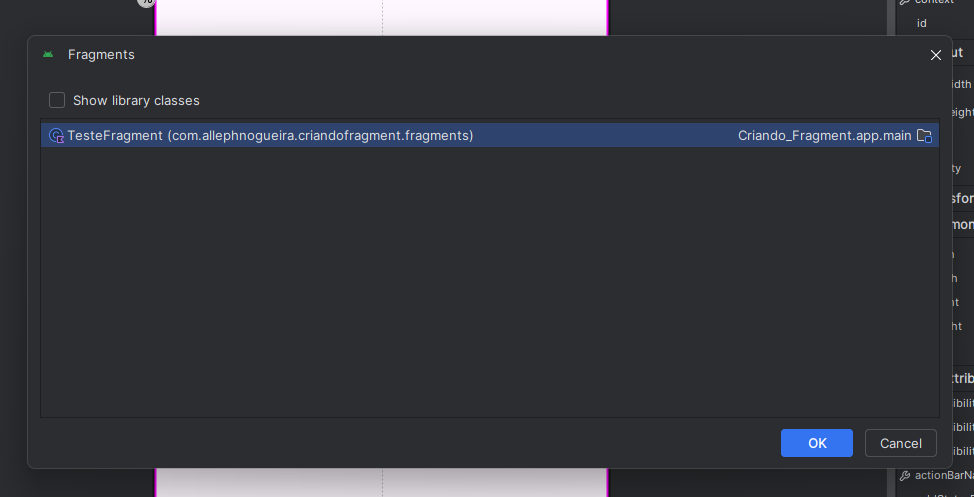
Para criar um fragment, fazemos a mesma coisa quando queremos criar um activity

Criamos um pacote chamado fragment.

Clicamos em New – e vamos na opção fragment, la vai ter varias opções para você usar.

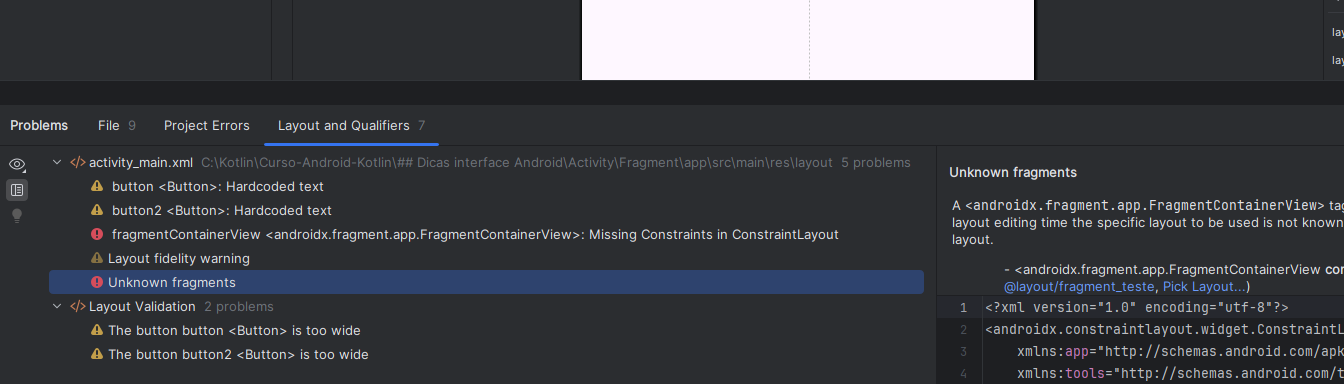


Quando clicamos em fragment e arrastamos para a tela, ele pergunta qual fragment você quer usar



Nesse caso temos apenas 1.

Quando você adicionar vai do esse erro, esse erro diz que é necessário ter um fragment teste, ate mesmo para você visualizar, então é só você clicar ali em teste



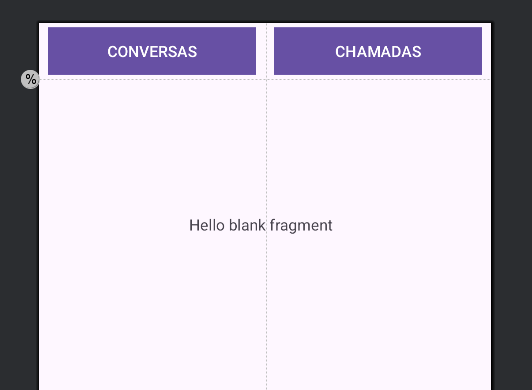


Agora para gerar a visualização e gerado o Tools

<androidx.fragment.app.FragmentContainerView  
 android:id="@+id/fragmentContainerView"  
 android:name="com.allephnogueira.criandofragment.fragments.TesteFragment"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 tools:layout\_editor\_absoluteX="137dp"  
 tools:layout\_editor\_absoluteY="174dp"  
 tools:layout="@layout/fragment\_teste" />

Oque é o tools? São coisas que quando você compilar o projeto ele não vai levar em consideração.

Agora ele vai aparecer o fragment como teste



Agora vamos para a parte do código, para fazer funcionar.

# Criando Fragment com CODIGO (Melhor opção)

Mas agora vamos criar do 0 um fragment.kt, não vamos utilizar o antigo.

Criamos a classe

package com.allephnogueira.criandofragment.fragments  
  
class ConversasFragment {  
}

Lembrar que vamos herda da classe Fragment

Vamos precisar também de um construtor vazio ()

Porque essa classe chama o método initLifecycle()

package com.allephnogueira.criandofragment.fragments  
  
import androidx.fragment.app.Fragment  
  
class ConversasFragment : Fragment() {  
}

Vamos utilizar o método onCreateView para criar nossa tela, mas não temos um layout.xml, então vamos criar ela também

Lembrar sempre de usar o nome com a nomenclatura

fragment\_conversas

package com.allephnogueira.criandofragment.fragments  
  
import android.os.Bundle  
import android.view.LayoutInflater  
import android.view.View  
import android.view.ViewGroup  
import androidx.fragment.app.Fragment  
import com.allephnogueira.criandofragment.R  
  
class ConversasFragment : Fragment() {  
  
  
 override fun onCreateView(  
 */\*\*  
 \* Inflate = inflar seu layout = construir seu layout  
 \*/* inflater: LayoutInflater,  
 container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 */\*\*  
 \* Esse metodo é onde vamos inflar nossa informação na tela  
 \* Inflar = jogar nossa informação na tela  
 \* Ele espera receber 3 parametros  
 \* Nosso layout  
 \*  
 \* Nosso view Group  
 \* Vamos passar o proprio container, porque ja recebemos ele ali (l-18)  
 \* Oque é esse container? é nosso componente o fragmentContainer  
 \*  
 \* o Ultimo é o attchToRoot = anexar o elemento raiz  
 \* false  
 \* Imagina que tem varios container(Fragmentos) na tela, podemos escolher de forma manual onde queremos colocar ele, mas nao queremos isso  
 \* Queremos deixar que ele coloque automatico, ate porque só temos 1 Container na tela.  
 \*  
 \* Se eu colocasse como TRUE  
 \* Seria eu mesmo que iria anexar de forma manual  
 \*/* return inflater.inflate(  
 R.layout.*fragment\_conversas*,  
 container,  
 false  
  
 )  
 }  
}

Agora vamos em activity\_main.xml, vamos acessar a área de code.

Aqui não vamos mais utilizar o testeFragment, porque criamos o nosso manual

android:name="com.allephnogueira.criandofragment.fragments.TesteFragment"

Agora vamos utilizar o fragment que criamos

<androidx.fragment.app.FragmentContainerView  
 android:id="@+id/fragmentContainerView2"  
 android:name="com.allephnogueira.criandofragment.fragments.ConversasFragment"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="@+id/guideline"  
 tools:layout="@layout/fragment\_conversas" />

Outra forma de fazer usando variável podendo alterar os dados como text, image..

package com.allephnogueira.criandofragment.fragments  
  
  
  
class ConversasFragment : Fragment() {  
  
  
 override fun onCreateView(  
 */\*\*  
 \* Inflate = inflar seu layout = construir seu layout  
 \*/* inflater: LayoutInflater,  
 container: ViewGroup?,  
 savedInstanceState: Bundle?  
 ): View? {  
 */\*\*  
 \* Esse metodo é onde vamos inflar nossa informação na tela  
 \* Inflar = jogar nossa informação na tela  
 \* Ele espera receber 3 parametros  
 \* Nosso layout  
 \*  
 \* Nosso view Group  
 \* Vamos passar o proprio container, porque ja recebemos ele ali (l-18)  
 \* Oque é esse container? é nosso componente o fragmentContainer  
 \*  
 \* o Ultimo é o attchToRoot = anexar o elemento raiz  
 \* false  
 \* Imagina que tem varios container(Fragmentos) na tela, podemos escolher de forma manual onde queremos colocar ele, mas nao queremos isso  
 \* Queremos deixar que ele coloque automatico, ate porque só temos 1 Container na tela.  
 \*  
 \* Se eu colocasse como TRUE  
 \* Seria eu mesmo que iria anexar de forma manual  
 \*/*// return inflater.inflate(  
// R.layout.fragment\_conversas,  
// container,  
// false  
//  
// )  
  
 */\*\*  
 \* Podemos fazer dessa forma também, porque assim vamos ter os itens que estão dentro do nosso fragment  
 \*/* val view = inflater.inflate(  
 R.layout.*fragment\_conversas*,  
 container,  
 false  
  
 )  
   
 view.findViewById<TextView>(R.id.*text\_Conversa*).*text* = "Alterando o texto"  
  
 return view  
  
 */\*\*  
 \* Agora imagina que queremos acessar o conversa  
 \*/* }  
  
}