



SISTEMAS DE INFORMAÇÃO
DESENVOLVIMENTO ANDROID

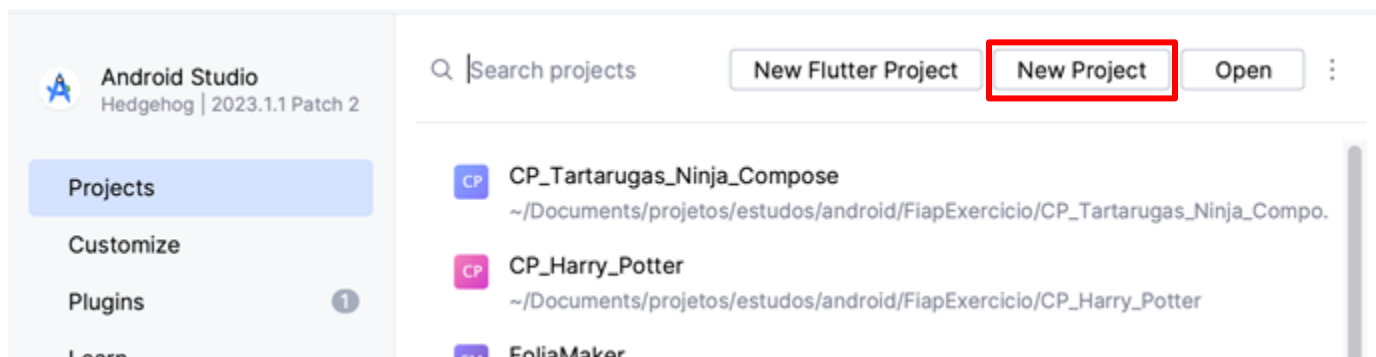


MOBILE DEVELOPMENT

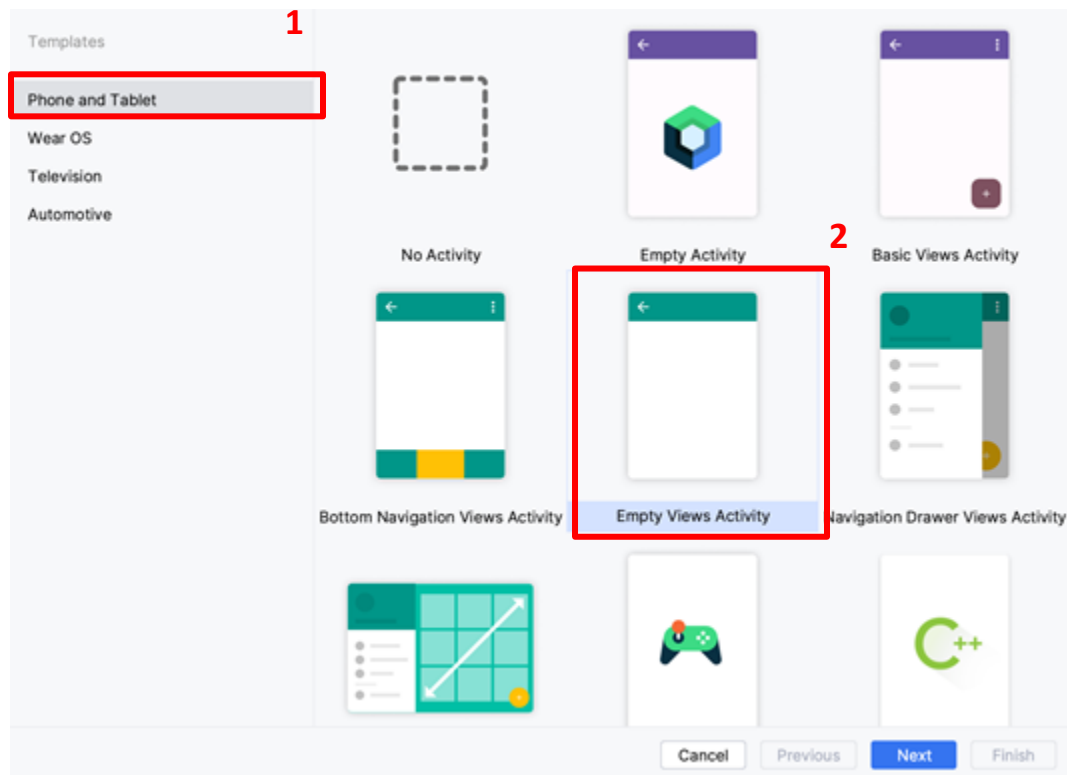


HORA DE COMEÇAR:
QUAL VAI SER O APP DE HOJE?

CRIANDO O PROJETO



CRIANDO O PROJETO



CRIANDO O PROJETO

Empty Views Activity

Creates a new empty activity


Name


Package name

Save location

Language

Minimum SDK

 Your app will run on approximately 81,2% of devices.
[Help me choose](#)

Build configuration language 

Gerador de Senha



GERADOR DE SENHA

Começar

Tamanho da senha: 8



Confirmar

Senha aqui

HORA DE COMEÇAR



build.gradle

Ativando o ViewBinding

Para fazer o bind iremos utilizar o **viewBinding**.

Abra o arquivo **build.gradle (app)** e adicione-o:

```
android {  
    namespace = "br.com.heiderlopes.geradordesenha"  
    compileSdk = 35  
  
    viewBinding {  
        enable = true  
    }  
}
```

CRIANDO A PRIMEIRA TELA

O que são Fragments?

Um **Fragment** é um componente independente do Android que pode ser usado por uma Activity. Eles encapsulam funcionalidades para que seja mais fácil reutilizar dentro de outras Activities e layouts.

Os **fragments** podem realizar muito mais que o agrupamento de interface. Eles permitem “modularizar” nossas activities, incluindo um ciclo de vida e os eventos que são utilizados em determinada interface.

Eles foram introduzidos pela **primeira vez na versão Honeycomb do Android** (primeira versão do Sistema Operacional a suportar tablets) para resolver um problema específico.

Por exemplo, se você tivesse um aplicativo com uma Activity de lista de itens, e quando clicasse em um item, abriria uma outra Activity com os detalhes. Esse tipo de fluxo de telas é chamado de Master/Detail, que vai da listagem de itens até o detalhe do item escolhido. Confira um exemplo na imagem abaixo:

O que são Fragments?



Casos de uso

Reutilizando componentes de visualização e lógica: os fragments permitem a reutilização de partes da sua tela, incluindo exibições e lógica de eventos várias vezes de maneiras diferentes em muitas actividades diferentes. Por exemplo, usando a mesma lista em diferentes fontes de dados em um aplicativo.

Suporte para Tablet: normalmente, em aplicativos, a versão para tablet de uma activity tem um layout diferente da versão para telefone, que é diferente da versão para TV. Os fragments permitem que actividades específicas do dispositivo reutilizem elementos compartilhados, ao mesmo tempo em que também apresentam diferenças.

Orientação de tela: normalmente a versão de retrato de uma activity tem um layout diferente da versão de paisagem. Fragments permitem que ambas as orientações reutilizem elementos compartilhados, ao mesmo tempo em que também possuem diferenças.

Como funcionam?

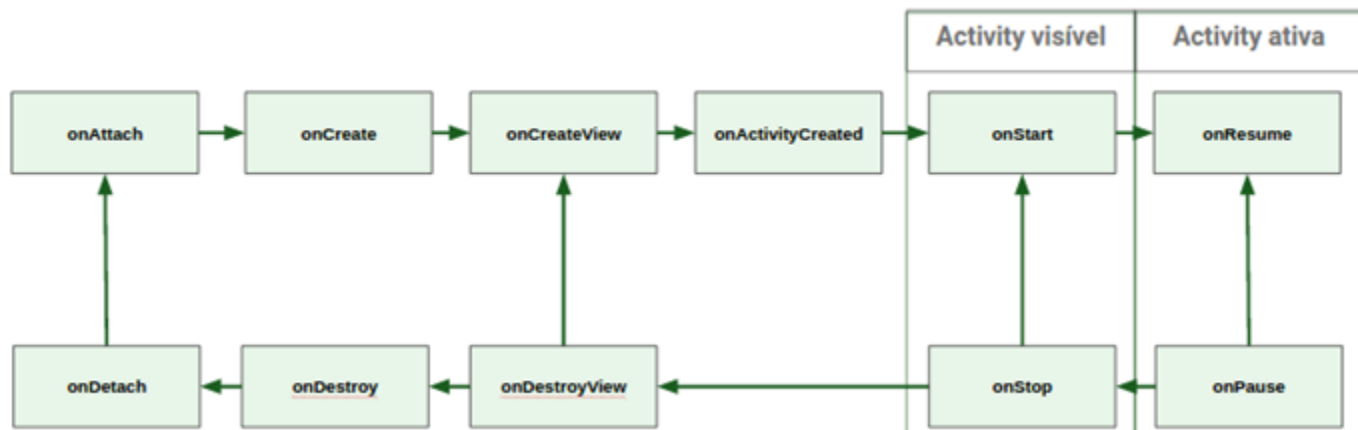
Os **Fragments** são executados no contexto de uma **Activity**, mas tem seu próprio ciclo de vida e normalmente sua própria interface de usuário.

Os eventos básicos do ciclo de vida de um **Fragment** são muito semelhantes às de uma **Activity** que está incorporando o mesmo.

E a medida que o ciclo de vida executa os eventos como **onStart**, **onResume**, **onPause** e **onStop**, esses mesmos eventos serão acionados dentro do próprio Fragment.

Assim, é possível mover a implementação feita nesses eventos da Activity para o Fragment sem muitos problemas, com algumas exceções.

Como funcionam?



Adicionando os arquivos do aplicativo

OnCreateView: é onde você constrói ou infla sua interface, faz conexão com alguma fonte de dados e retorna à Activity pai para poder integrá-lo em sua hierarquia de Views. Devemos utilizar esse método para construir a interface.

OnDestroyView: é correspondente ao onDestroy da Activity e é chamado imediatamente antes do Fragment ser destruído. Ele funciona independente da Activity pai. Aqui é devemos limpar quaisquer recursos especificamente relacionados à interface, como bitmaps na memória, cursores de dados, para garantir que não haja problemas de memória.

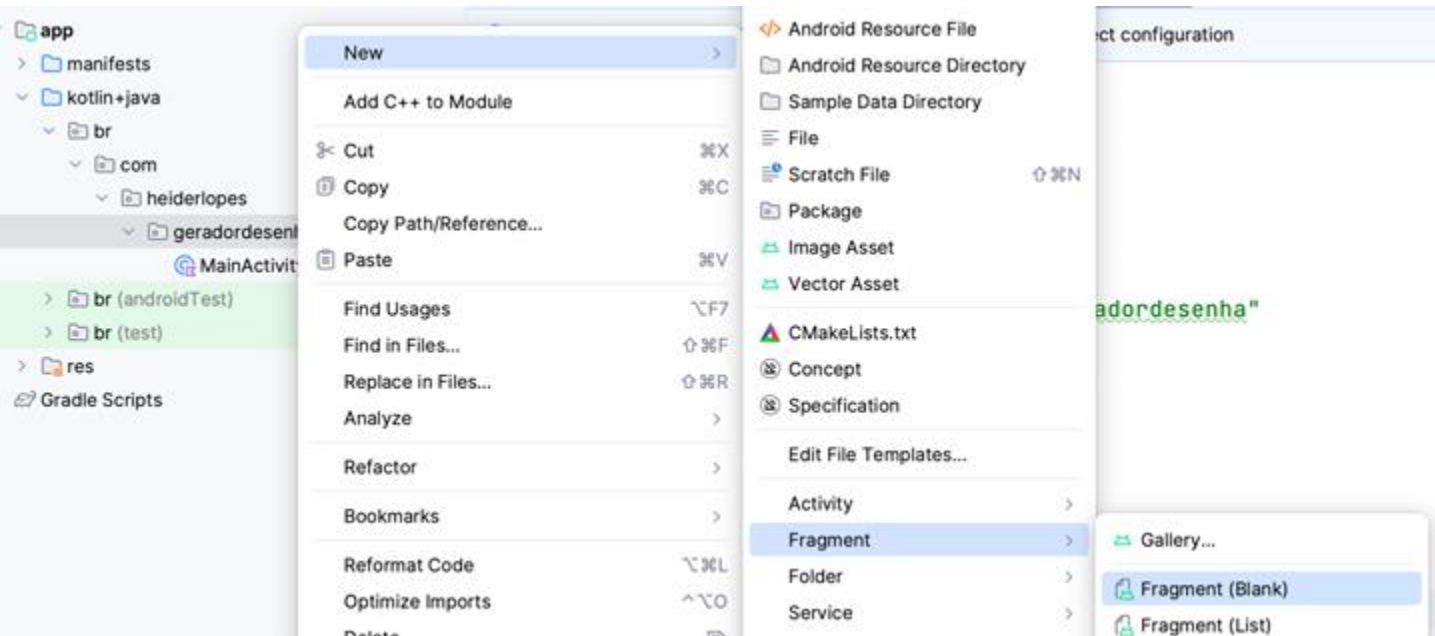
OnAttach: é onde podemos obter uma referência para a Activity pai.

OnDetach: é a última coisa que acontece no ciclo de vida, mesmo após o seu Fragment ser tecnicamente destruído.

OnActivityCreated: notifica nosso Fragment que a Activity pai completou seu ciclo no onCreate e é aqui que podemos interagir com segurança com a interface de usuário.

Criando o Fragment

Clique com o botão direito sobre o pacote principal → **New** → **Fragment** → **Fragment (Blank)**



Criando o Fragment

Defina o nome do **Fragment** e clique em **Finish**

Fragment (Blank)
Creates a blank fragment that is compatible back to API level 16

Fragment Name

Fragment Layout Name

Source Language

Cancel Previous Next **Finish**

LOTTIE

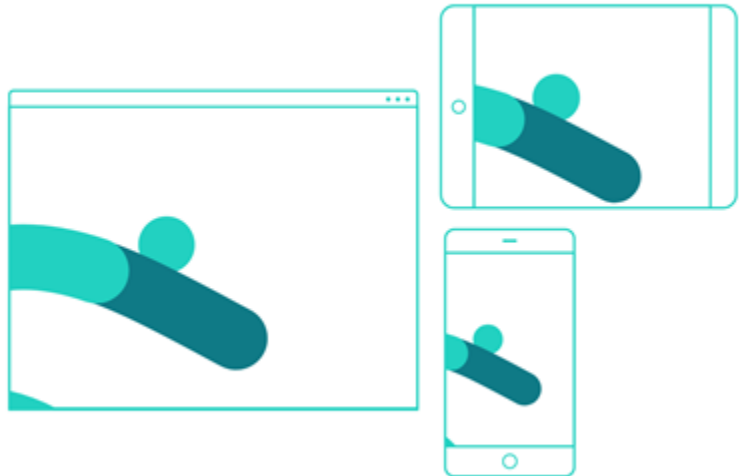
Lottie

Easily add high-quality animation to any native app.

Lottie is an iOS, Android, and React Native library that renders After Effects animations in real time, allowing apps to use animations as easily as they use static images.

[Get Started](#)

[Learn more >](#)



Saiba mais: <https://airbnb.design/lottie/>
<https://www.lottiefiles.com>

LOTTIE

Abra o arquivo **build.gradle (app)** e adicione a seguinte dependência:

```
implementation ("com.airbnb.android:lottie:6.4.0")
```

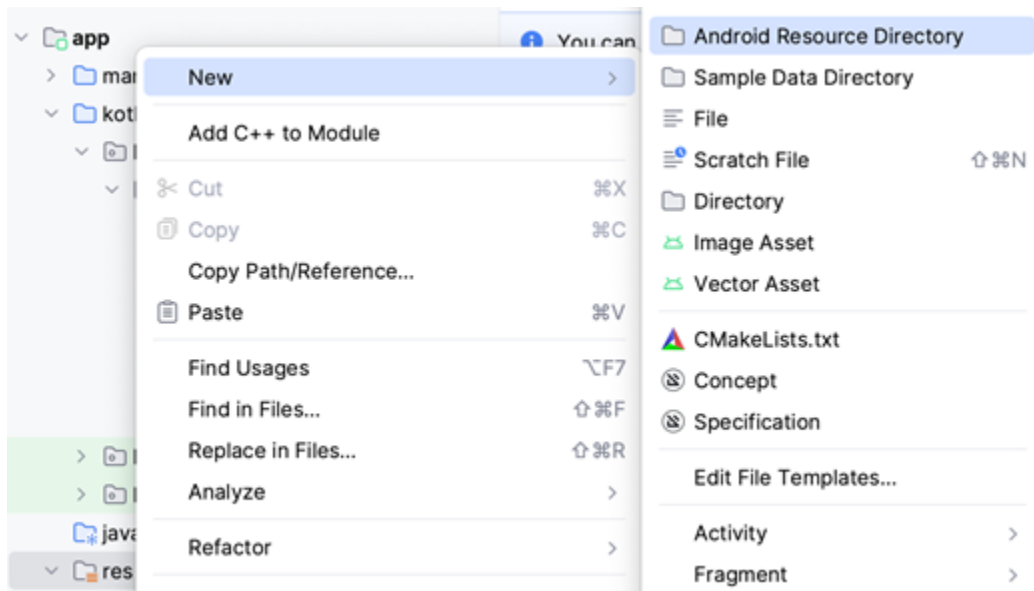
E clique em:

A button with a light blue background and a thin blue border, containing the text "Sync Now" in a blue sans-serif font.

Sync Now

LOTTIE

Crie um diretório dentro de **res** chamado **raw** (**botão direito sobre res** → **New** → **Android Resource Directory**)



LOTTIE

Crie um diretório dentro de **res** chamado **raw** (**botão direito sobre res** → **New** → **Android Resource Directory**)

The screenshot shows the 'New Android Resource Directory' dialog box. At the top, there are three input fields: 'Directory name:' with the text 'raw', 'Resource type:' with a dropdown menu showing 'raw' (highlighted with a blue border), and 'Source set:' with a dropdown menu showing 'main src/main/res'. Below these fields, there are two panels: 'Available qualifiers:' on the left and 'Chosen qualifiers:' on the right. The 'Available qualifiers:' panel contains a list of qualifiers with icons: Country Code, Network Code, Locale, Layout Direction, Smallest Screen Width, Screen Width, Screen Height, Size, Ratio, Orientation, and UI Mode. The 'Chosen qualifiers:' panel is empty and contains the text 'Nothing to show'. Between the two panels are two buttons: '>>' and '<<'. At the bottom left is a help icon (a question mark in a circle), and at the bottom right are 'Cancel' and 'OK' buttons.

Directory name:

Resource type:

Source set:

Available qualifiers:

- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation
- UI Mode

Chosen qualifiers:

Nothing to show

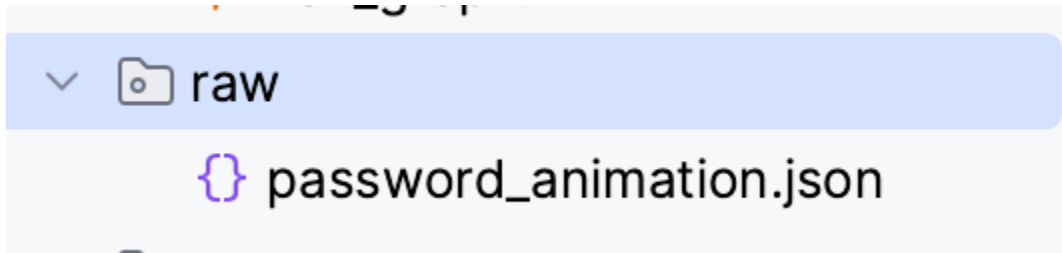
>> <<

? Cancel OK

LOTTIE

Baixe as animações desejada no lottie files (<https://lottiefiles.com/>) e adicione na pasta **raw** com o nome:

password_animation.json:



Criando o Fragment

Abra o arquivo **fragment_first.xml** e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:lottie="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="16dp">

    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/lottieAnimationView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_gravity="center"
        lottie:lottie_rawRes="@raw/password_animation"
        lottie:lottie_autoPlay="true"
        lottie:lottie_loop="true"/>
```


Criando o Fragment

Abra o arquivo **fragment_first.xml** e adicione o seguinte código:

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="GERADOR DE SENHA"
    android:textAlignment="center"
    android:textSize="20sp" />

<Button
    android:id="@+id/btIniciar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginVertical="16dp"
    android:text="Começar" />
</LinearLayout>
```

Criando o Fragment

Abra o arquivo **FirstFragment.kt** e adicione o código referente ao viewbinding das views:

```
class FirstFragment : Fragment() {  
  
    private lateinit var binding: FragmentFirstBinding  
  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View {  
        binding = FragmentFirstBinding.inflate(inflater, container,  
false)  
  
        binding.btIniciar.setOnClickListener {  
  
        }  
        return binding.root  
    }  
}
```

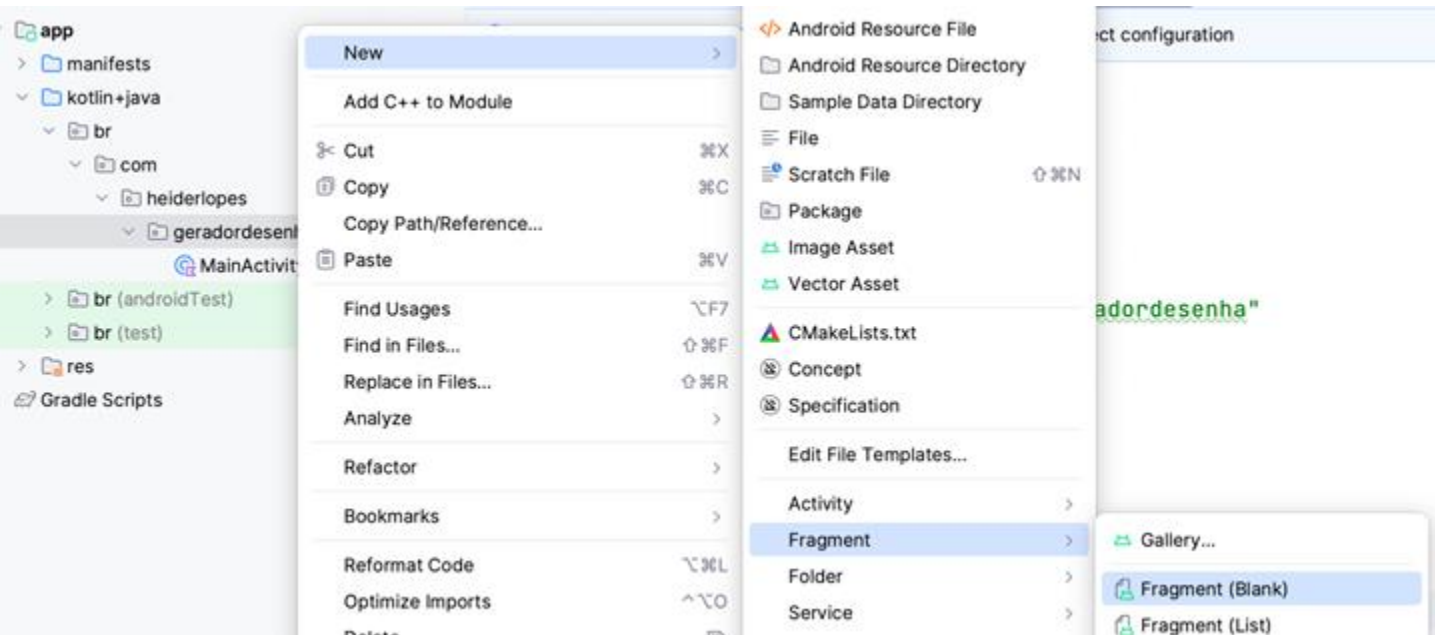


CRIANDO A SEGUNDA TELA



Criando o Fragment

Clique com o botão direito sobre o pacote principal → **New** → **Fragment** → **Fragment (Blank)**



Criando o Fragment

Defina o nome do **Fragment** e clique em **Finish**

Fragment (Blank)
Creates a blank fragment that is compatible back to API level 16

Fragment Name

Fragment Layout Name

Source Language

Criando o Fragment

Abra o arquivo **fragment_second.xml** e adicione o seguinte código:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/tvTamanho"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Tamanho da senha: 8" />
```

Criando o Fragment

Abra o arquivo **fragment_second.xml** e adicione o seguinte código:

```
<SeekBar
    android:layout_marginVertical="32dp"
    android:id="@+id/seekBar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:max="20"
    android:progress="8" />

<Button
    android:id="@+id/btnConfirmar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Confirmar" />
</LinearLayout>
```

Criando o Fragment

Abra o arquivo **SecondFragment.kt** e adicione o código referente ao viewbinding das views:

```
class SecondFragment : Fragment() {  
    private lateinit var binding: FragmentSecondBinding  
    private var tamanho = 8  
  
    override fun onCreateView(inflater: LayoutInflater, container:  
ViewGroup?, savedInstanceState: Bundle?): View {  
        binding = FragmentSecondBinding.inflate(inflater, container,  
false)
```


Criando o Fragment

Abra o arquivo **SecondFragment.kt** e adicione o código referente ao viewbinding das views:

```
binding.seekBar.setOnSeekBarChangeListener(object :
SeekBar.OnSeekBarChangeListener {
    override fun onProgressChanged(seekBar: SeekBar?, progress:
Int, fromUser: Boolean) {
        tamanho = progress.coerceAtLeast(4)
        binding.tvTamanho.text = "Tamanho da senha: $tamanho"
    }

    override fun onStartTrackingTouch(seekBar: SeekBar?) {}
    override fun onStopTrackingTouch(seekBar: SeekBar?) {}
})

binding.btnConfirmar.setOnClickListener {

}

return binding.root
}
}
```

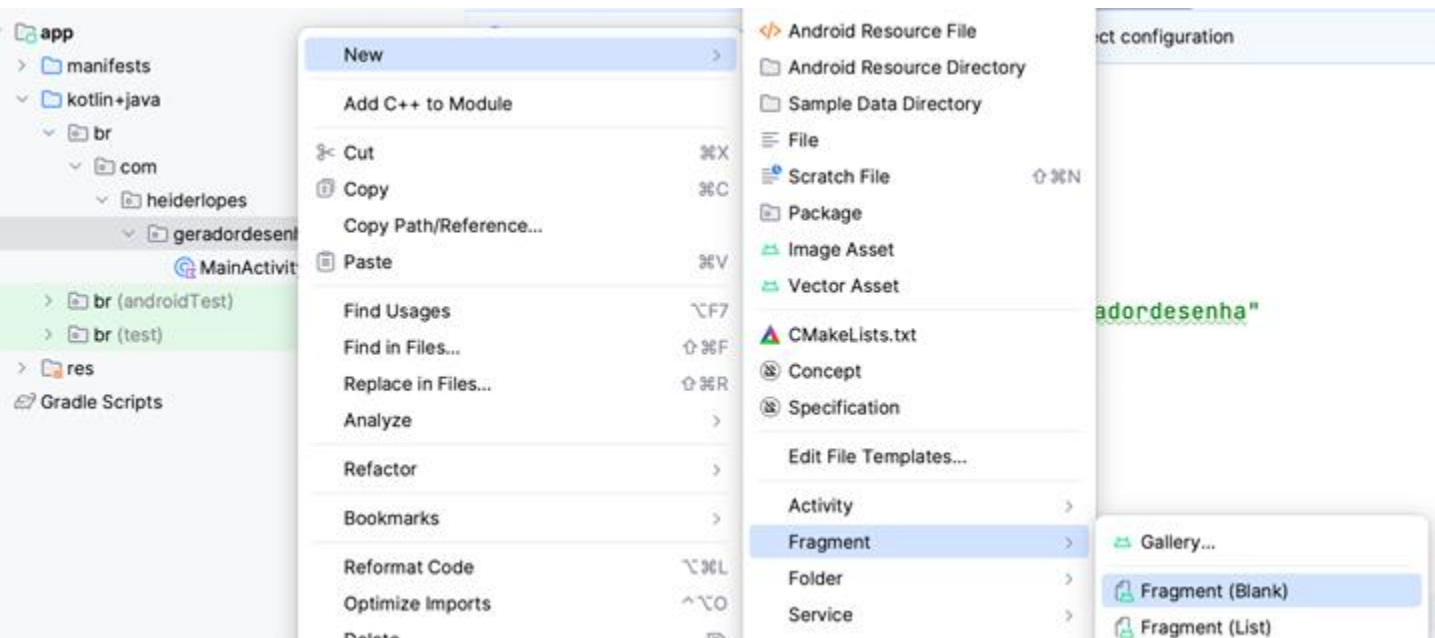


CRIANDO A TERCEIRA TELA



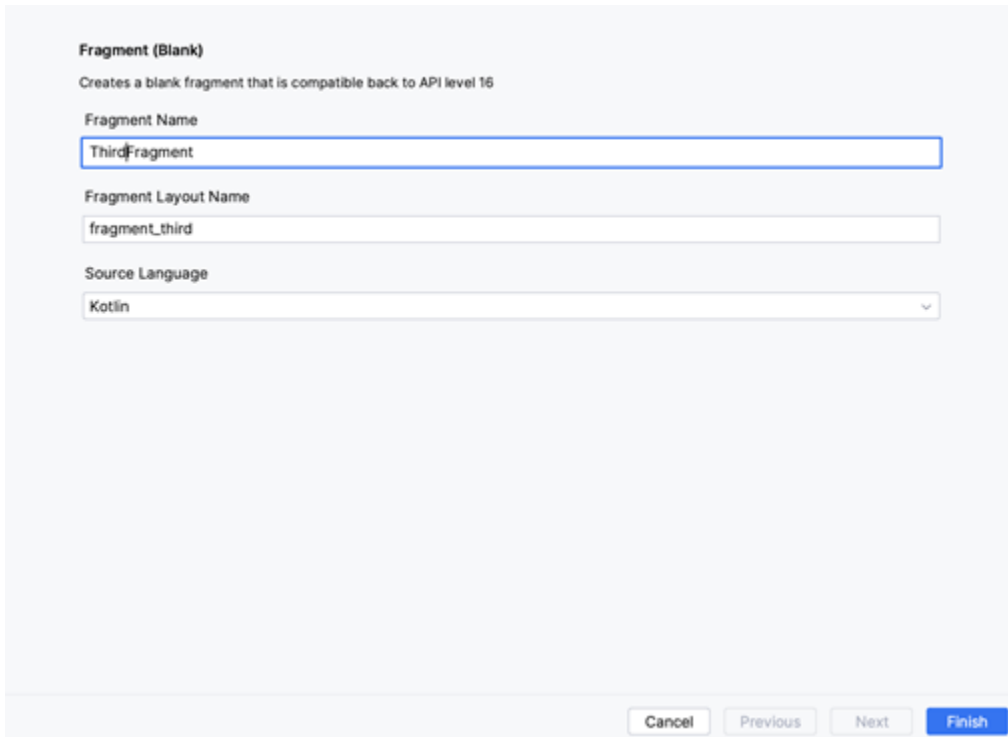
Criando o Fragment

Clique com o botão direito sobre o pacote principal → **New** → **Fragment** → **Fragment (Blank)**



Criando o Fragment

Defina o nome do **Fragment** e clique em **Finish**



Fragment (Blank)
Creates a blank fragment that is compatible back to API level 16

Fragment Name
ThirdFragment

Fragment Layout Name
fragment_third

Source Language
Kotlin

Cancel Previous Next **Finish**

Criando o Fragment

Abra o arquivo **fragment_third.xml** e adicione o seguinte código:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:textAlignment="center"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/tvSenha"
        android:text="Senha aqui"
        android:textSize="20sp" />

</LinearLayout>
```

Criando o Fragment

Abra o arquivo **ThirdFragment.kt** e adicione o código referente ao viewbinding das views:

```
class ThirdFragment : Fragment() {  
    private lateinit var binding: FragmentThirdBinding  
  
    override fun onCreateView(  
        inflater: LayoutInflater,  
        container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View {  
        binding = FragmentThirdBinding.inflate(inflater, container,  
false)  
  
        return binding.root  
    }  
}
```



CRIANDO A NAVEGAÇÃO



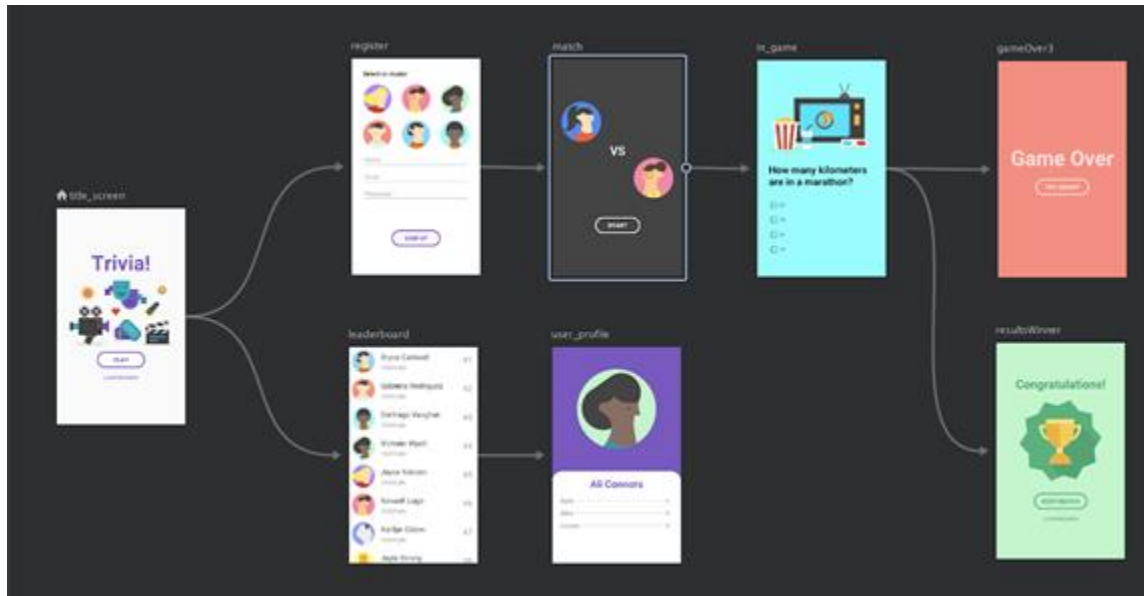
ANDROID JETPACK

É um conjunto de bibliotecas e ferramentas criado com o objetivo de simplificar e melhorar a qualidade do processo de desenvolvimento de aplicativos Android. Tais bibliotecas facilitam o desenvolvimento de apps com qualidade, previsibilidade e simplicidade.



NAVIGATION

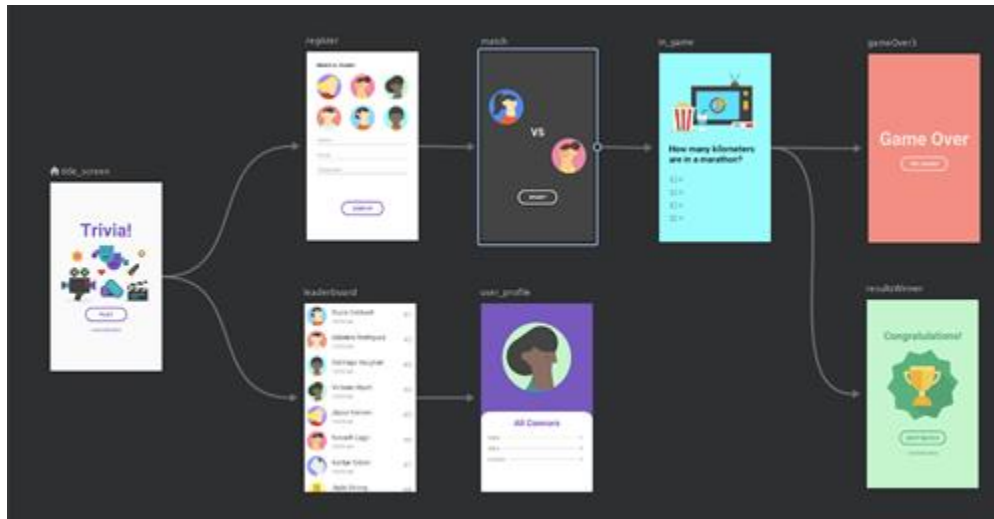
O principal objetivo de utilizar este novo componente é **reduzir a probabilidade de erro ao realizar alguma transação com fragments** e **melhorar a capacidade de testar a UI** de forma isolada.



NAVIGATION

Ao utilizar o **Navigation Component** você estará delegando ao componente os seguintes conceitos de navegação:

- Automação nas transações de fragments
- Implementação dos princípios de navegação



ADICIONANDO A BIBLIOTECA

ADICIONANDO O NAVIGATION

Abra o arquivo **build.gradle (app)** e adicione as linhas em negrito:

```
dependencies {  
  
    val nav_version = "2.8.9"  
  
    implementation ("androidx.navigation:navigation-fragment-  
ktx:$nav_version")  
    implementation ("androidx.navigation:navigation-ui-ktx:$nav_version")  
}
```

ADICIONANDO O NAVIGATION

Para realizar a navegação entre telas com passagem de parâmetros de forma segura através do componente de navegação é necessário configurar o plugin no **gradle**. Para isso, abra o arquivo **build.gradle (.)** e adicione o seguinte classpath:

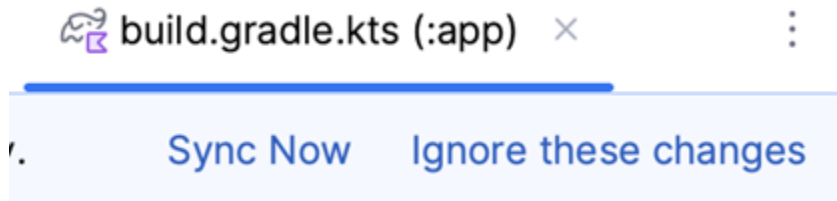
```
buildscript {  
    repositories {  
        google() // Google's Maven repository  
    }  
    dependencies {  
        classpath ("androidx.navigation:navigation-safe-args-gradle-  
plugin:2.8.9")  
    }  
}
```

ADICIONANDO O NAVIGATION

Abra o arquivo **build.gradle (app)** e adicione o seguinte plugin em negrito:

```
plugins {  
    id("com.android.application")  
    id("org.jetbrains.kotlin.android")  
    id("androidx.navigation.safeargs.kotlin")  
}
```

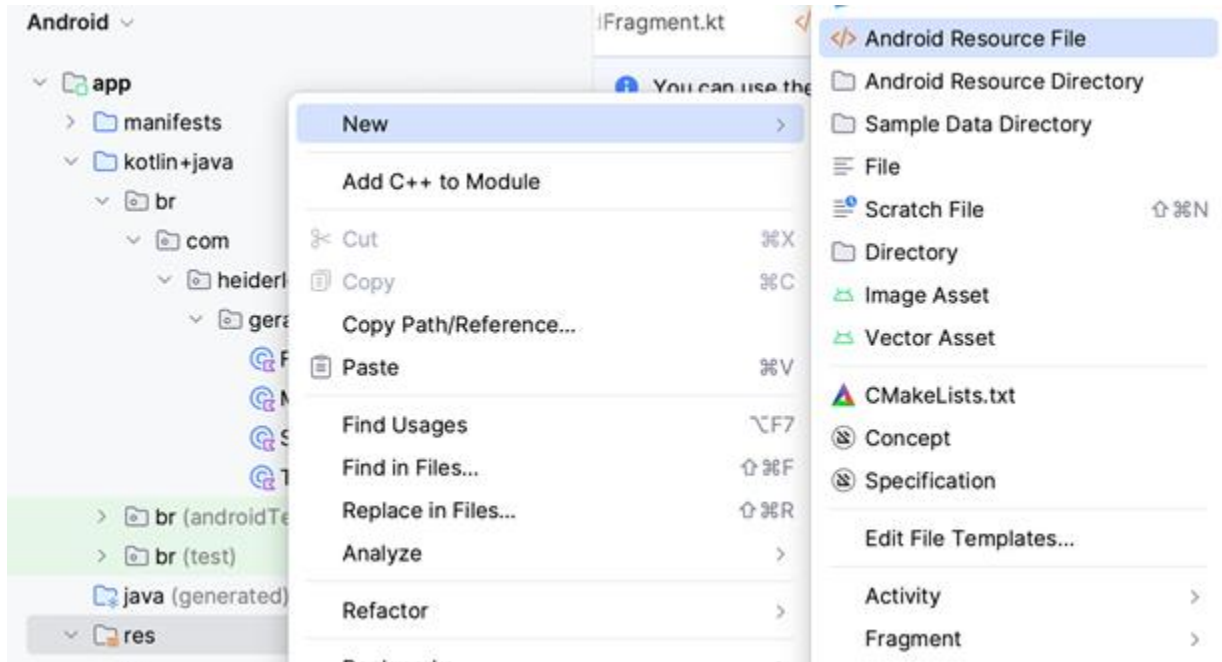
Em seguida, clique em **Sync Now**:



CRIANDO O ARQUIVO DE NAVEGAÇÃO

CRIANDO O ARQUIVO DE NAVEGAÇÃO

Clique com o botão direito em **res > New > Android Resource File**



CRIANDO O ARQUIVO DE NAVEGAÇÃO

Dê o nome do arquivo **nav_graph** e o **Resource type** como **Navigation**. Em seguida, clique **OK**

The screenshot shows the 'New Resource File' dialog in Android Studio. The 'File name' field is set to 'nav_graph'. The 'Resource type' dropdown is set to 'Navigation'. The 'Root element' is 'navigation'. The 'Source set' is 'main src/main/res'. The 'Directory name' is 'navigation'. The 'Available qualifiers' list on the left includes 'Country Code', 'Network Code', 'Locale', 'Layout Direction', 'Smallest Screen Width', 'Screen Width', 'Screen Height', and 'Size'. The 'Chosen qualifiers' list on the right is empty, displaying 'Nothing to show'. At the bottom, there is a help icon, a 'Cancel' button, and an 'OK' button.

File name:	nav_graph
Resource type:	Navigation
Root element:	navigation
Source set:	main src/main/res
Directory name:	navigation

Available qualifiers:

- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size

Chosen qualifiers:

Nothing to show

Buttons: ? Cancel OK



CONFIGURANDO A NAVEGAÇÃO NA ACTIVITY



CONFIGURANDO A NAVEGAÇÃO NA ACTIVITY

Abra o arquivo **activity_main.xml** e adicione o seguinte código:

```
<?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="match_parent"
    tools:context=".MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/nav_graph" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

CONFIGURANDO A NAVEGAÇÃO NA ACTIVITY

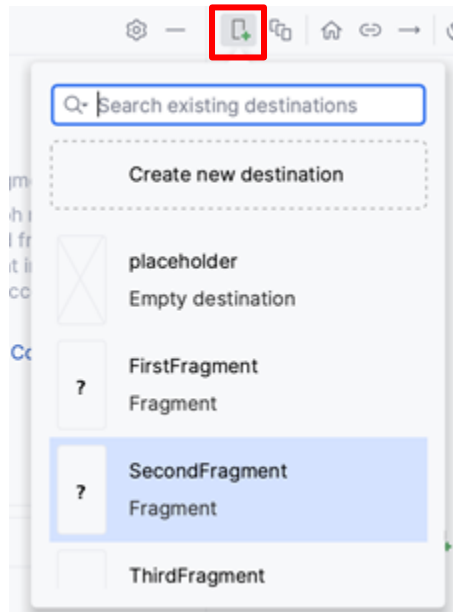
Abra o arquivo **MainActivity.xml** e adicione o seguinte código:

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        val binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
    }  
}
```

MONTANDO A NAVEGAÇÃO

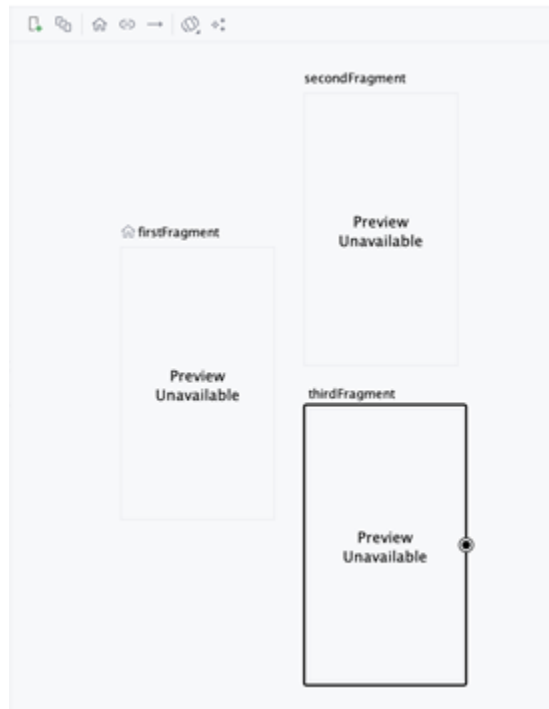
MONTANDO A NAVEGAÇÃO

O próximo passo é adicionar os fragments (telas) criados anteriormente no **nav_graph**. Com o arquivo **nav_graph.xml** aberto, clique sobre o **New Destination**.



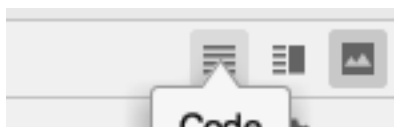
MONTANDO A NAVEGAÇÃO

Adicione nesta ordem os Fragments (**FirstFragment**, **SecondFragment**, **ThirdFragment**).



MONTANDO A NAVEGAÇÃO

Caso no graph não exiba o layout das suas telas, você pode entrar no modo **code** e adicionar a tag **tools:layout** no **fragment** e escolher o referente a aquela tela.



MONTANDO A NAVEGAÇÃO

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/nav_graph"
    xmlns:tools="http://schemas.android.com/tools"
    app:startDestination="@id/firstFragment">

    <fragment
        tools:layout="@layout/fragment_first"
        android:id="@+id/firstFragment"
        android:name="br.com.heiderlopes.geradordesenha.FirstFragment"
        android:label="FirstFragment" />

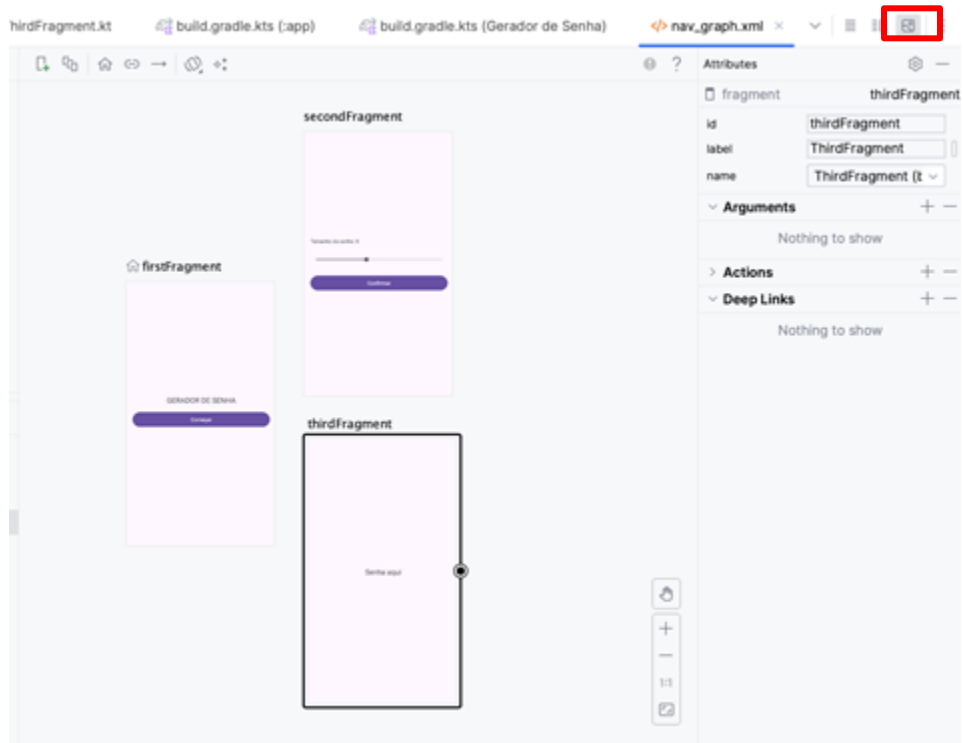
    <fragment
        tools:layout="@layout/fragment_second"
        android:id="@+id/secondFragment"
        android:name="br.com.heiderlopes.geradordesenha.SecondFragment"
        android:label="SecondFragment" />

    <fragment
        tools:layout="@layout/fragment_third"
        android:id="@+id/thirdFragment"
        android:name="br.com.heiderlopes.geradordesenha.ThirdFragment"
        android:label="ThirdFragment" />

</navigation>
```

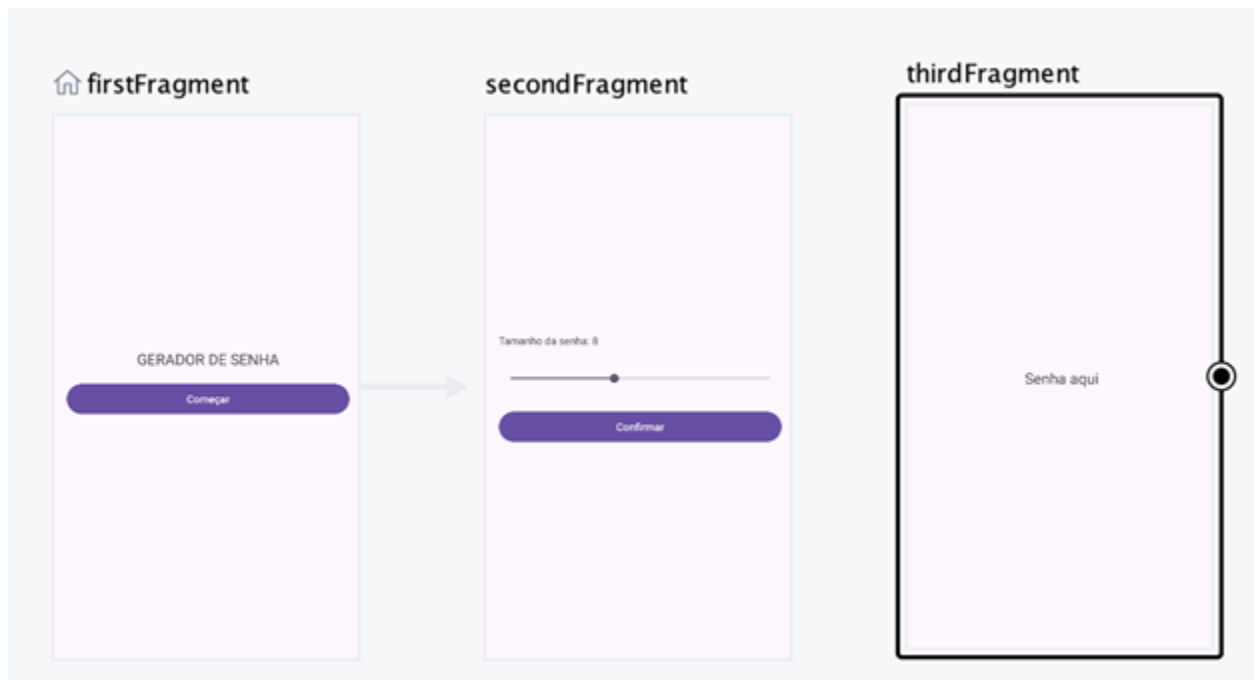
MONTANDO A NAVEGAÇÃO

Volte para o modo **Design**



MONTANDO A NAVEGAÇÃO

Selecione o **firstFragment** e no cursor  que aparece do lado direito clique segure e arraste até o **secondFragment**. Com isso, criamos uma navegação entre essas telas.



NAVEGANDO DA PRIMEIRA TELA PARA A SEGUNDA

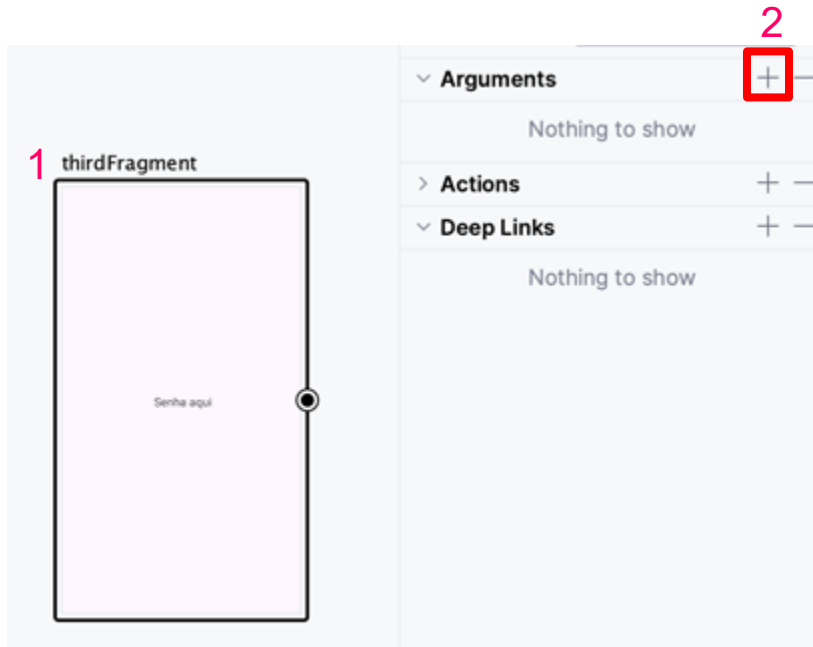
Abra o arquivo **FirstFragment.kt** e adicione  seguinte código em negrito ao clique do botão:

```
binding.btIniciar.setOnClickListener {  
    findNavController().navigate(R.id.action_firstFragment_to_secondFragment)  
}
```

PASSANDO PARÂMETROS COM JETPACK NAVIGATION

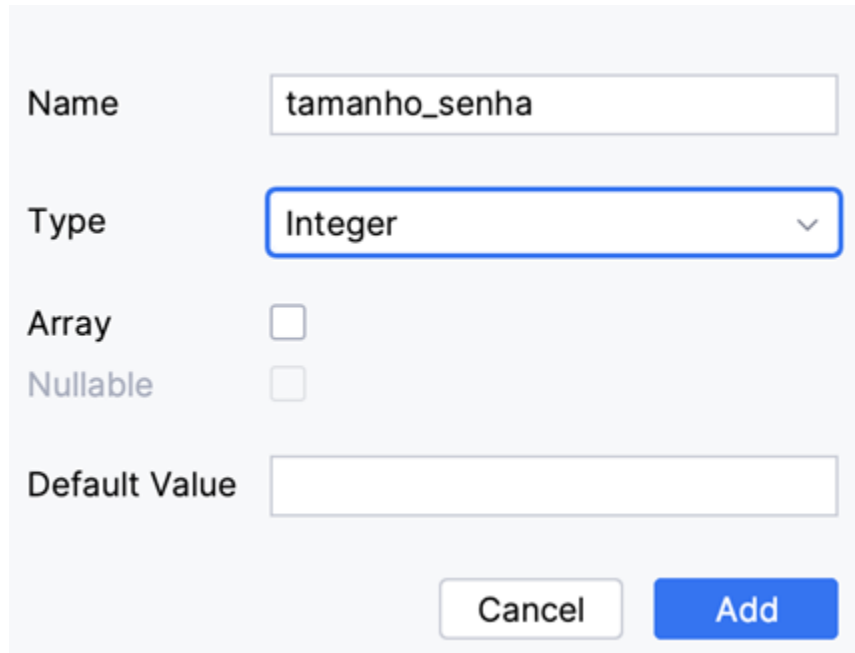
PASSANDO PARÂMETROS COM JETPACK NAVIGATION

Abra o arquivo **nav_graph.xml**, clique sobre **thirdFragment**, em seguida, no sinal **+** do **Arguments**



PASSANDO PARÂMETROS COM JETPACK NAVIGATION

Dê o **name** como **tamanho_senha** e o **type** como **Integer**



The image shows a configuration dialog for a navigation parameter. It has a light gray background and contains the following elements:

- Name:** A text input field containing the value "tamanho_senha".
- Type:** A dropdown menu with "Integer" selected and a blue border. A small downward arrow is visible on the right.
- Array:** A checkbox that is currently unchecked.
- Nullable:** A checkbox that is currently unchecked.
- Default Value:** An empty text input field.
- Buttons:** At the bottom right, there are two buttons: a "Cancel" button with a white background and a blue border, and an "Add" button with a solid blue background and white text.

ADICIONANDO O NAVIGATION

Faça o mesmo procedimento agora de **secondFragment** para **thirdFragment**



PASSANDO PARÂMETROS COM JETPACK NAVIGATION

Abra o arquivo **SecondFragment.kt** e adicione o parâmetro que será enviado ao clicar em **Confirmar**.

```
binding.btnConfirmar.setOnClickListener {  
    val action =  
        SecondFragmentDirections.actionSecondFragmentToThirdFragment(tamanho)  
    findNavController().navigate(action)  
}
```

PASSANDO PARÂMETROS COM JETPACK NAVIGATION

Abra o arquivo **ThirdFragment.kt** e recupere o valor utilizado para gerar a senha:

```
class ThirdFragment : Fragment() {
    private lateinit var binding: FragmentThirdBinding

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View {
        binding = FragmentThirdBinding.inflate(inflater, container, false)

        val length =
            ThirdFragmentArgs.fromBundle(requireArguments()).tamanhoSenha
        val senha = gerarSenha(length)
        binding.tvSenha.text = "Senha gerada:\n$senha"

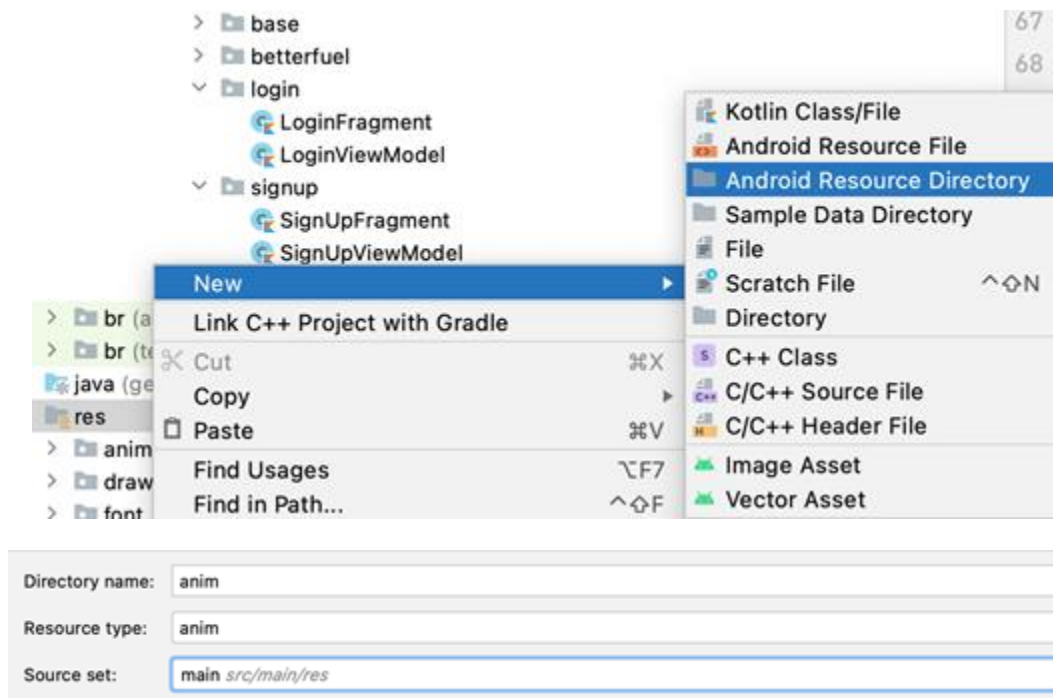
        return binding.root
    }

    private fun gerarSenha(tamanho: Int): String {
        val chars =
            "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#\$%&*"
        return (1..tamanho).map { chars.random() }.joinToString("")
    }
}
```

CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Crie uma pasta chamada **anim** dentro de **res**



CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Dentro da pasta **anim** crie um arquivo chamado **slide_in_left** e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="@android:integer/config_mediumAnimTime"
    android:fromXDelta="-100%p"
    android:toXDelta="0" />
```

CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Dentro da pasta **anim** crie um arquivo chamado **slide_in_right** e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="@android:integer/config_mediumAnimTime"
    android:fromXDelta="100%p"
    android:toXDelta="0" />
```

CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Dentro da pasta **anim** crie um arquivo chamado **slide_out_left** e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="@android:integer/config_mediumAnimTime"
    android:fromXDelta="0"
    android:toXDelta="-100%p" />
```

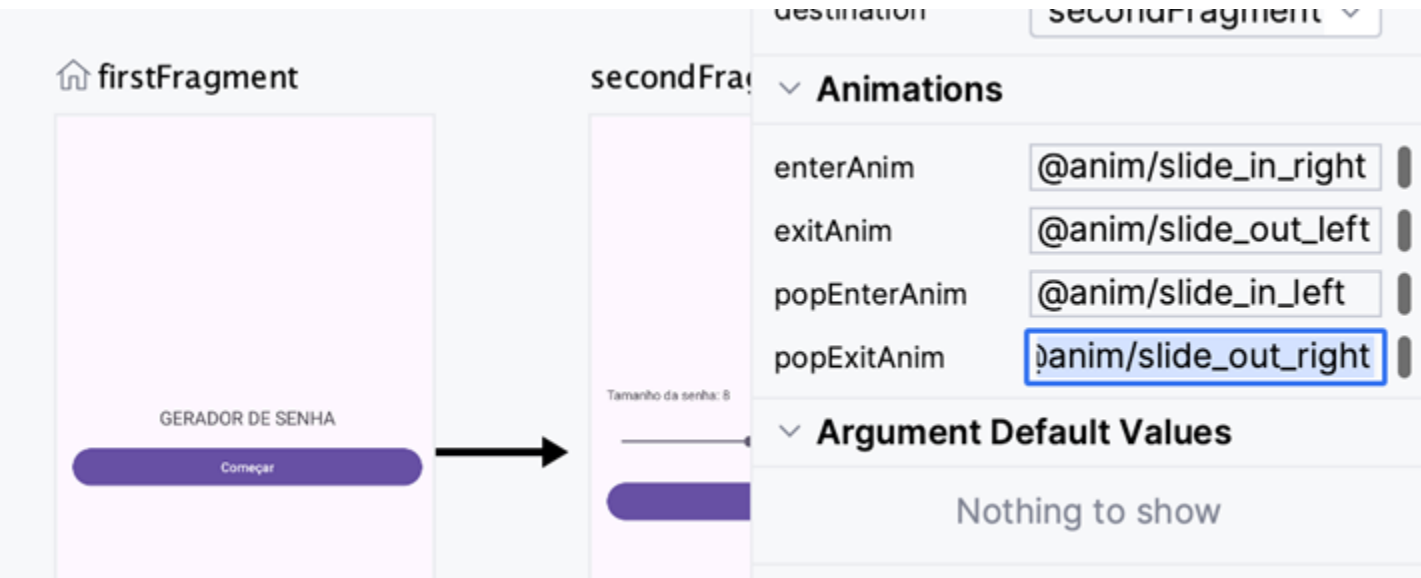
CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Dentro da pasta **anim** crie um arquivo chamado **slide_out_right** e adicione o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<translate
xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="@android:integer/config_mediumAnimTime"
    android:fromXDelta="0"
    android:toXDelta="100%p" />
```

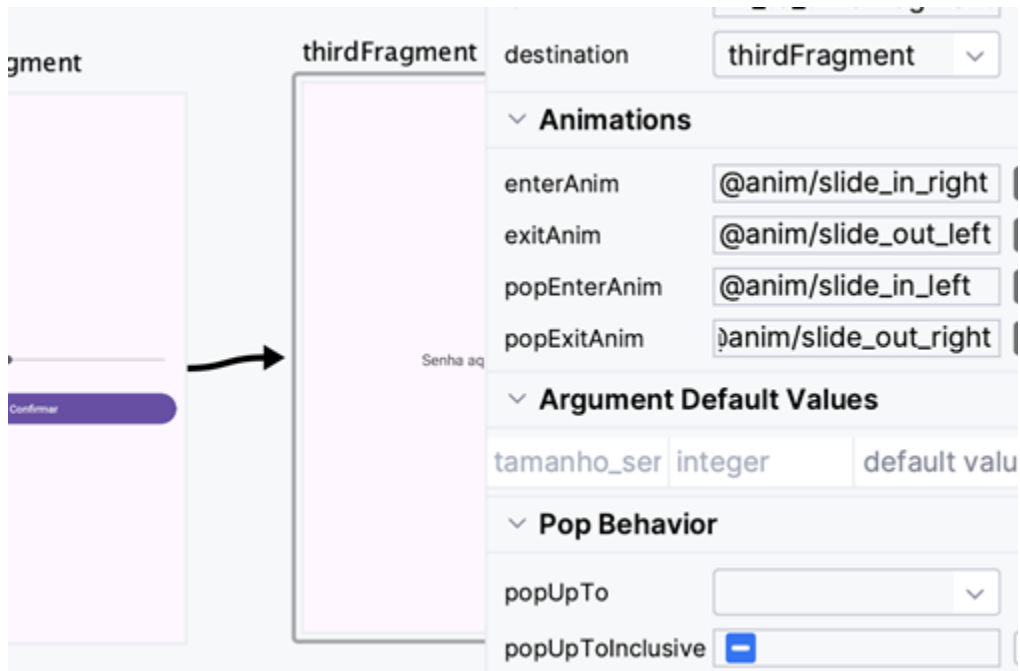

CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Aplique a animação na transição do fragment desejado. No caso, clique sobre a seta que sai do **FirstFragment** para o **SecondFragment**.



CRIANDO ANIMAÇÕES NAS TRANSIÇÕES DE TELAS

Aplique a animação na transição do fragment desejado. No caso, clique sobre a seta que sai do **SecondFragment** para o **ThirdFragment**.





MELHORANDO A NAVEGAÇÃO



MELHORANDO A EXPERIÊNCIA DA NAVEGAÇÃO

Clique sobre a action que vai da tela de **secondFragment** para o **thirdFragment** e configure o **popUpBehavior** conforme imagem abaixo. Com isso, removemos as telas necessárias do back stack

The diagram illustrates the navigation flow between three fragments:

- firstFragment**: Contains a button labeled "Começar".
- secondFragment**: Contains a button labeled "Confirmar".
- thirdFragment**: Contains the text "Senha aqui".

An arrow indicates the transition from **secondFragment** to **thirdFragment**. A settings panel on the right shows the configuration for this transition:

- enterAnim**: @anim/slide_in_right
- exitAnim**: @anim/slide_out_left
- popEnterAnim**: @anim/slide_in_left
- popExitAnim**: @anim/slide_out_right
- Argument Default Values**:
 - 1anho_senha: integer, default value
- Pop Behavior**:
 - popUpTo**: firstFragment (selected)
 - popUpToInclusive**: ☒
- Launch Options**:
 - launchSingleTop**: ☒

OBRIGADO



/heider.lopes



/in/heider-lopes-a06b2869/

FIAP

Copyright © 2019 | Professor (a) Heider Lopes

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.

FIAP