

Kotlin

for Android App Development

DESENVOLVIMENTO MOBILE

ANDROID – OTIMIZANDO FINDVIEW

```
//lateinit = "Confia que vou inicializar antes de usar"
lateinit var dado_imagem: ImageView

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    var botão: Button = findViewById(R.id.roll_button)

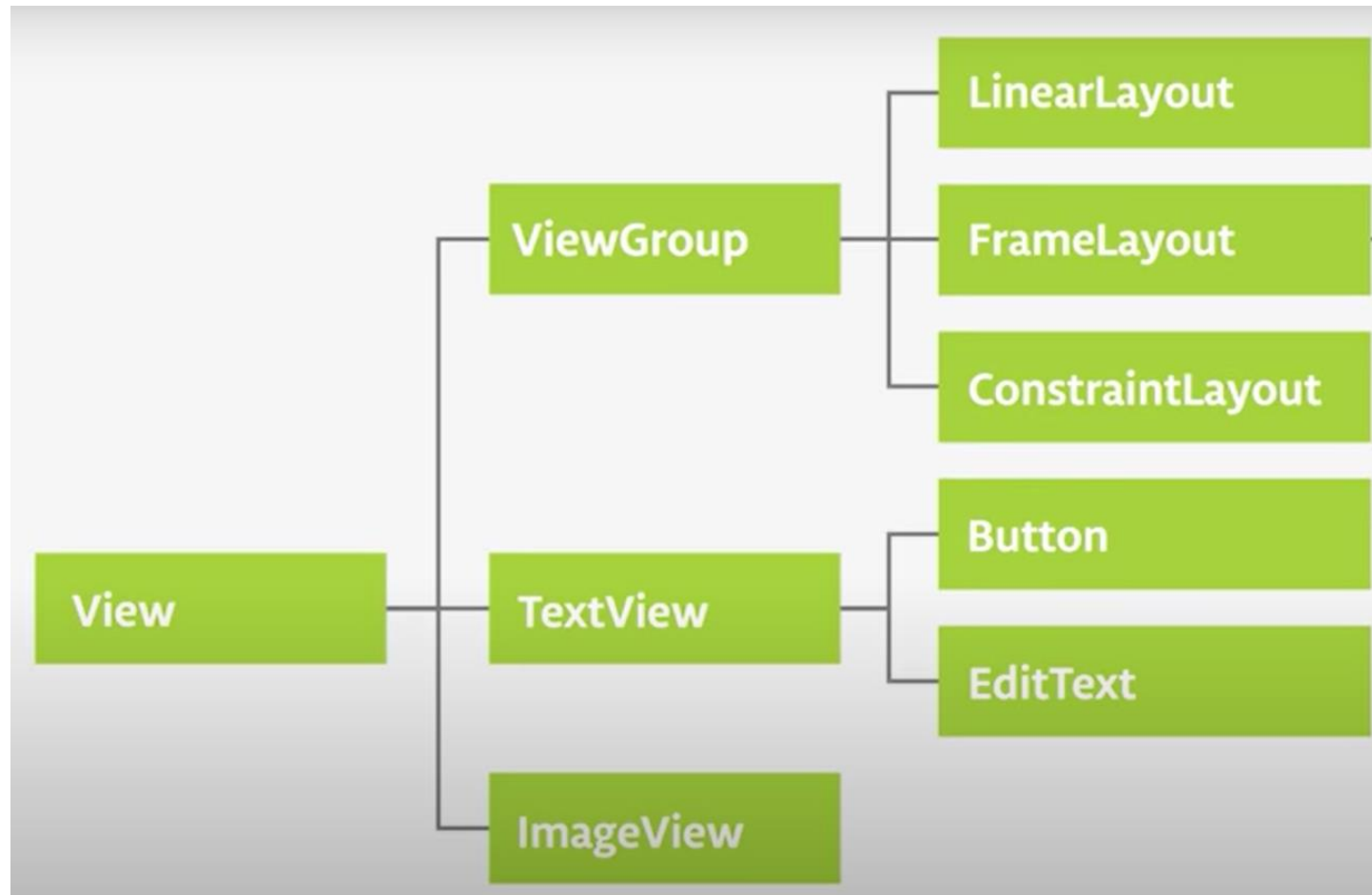
    dado_imagem = findViewById(R.id.dado_imagem)
}
```

ANDROID – COMPONENTES BÁSICOS

View : Um objeto gráfico visível, exemplos como TextView, Button, ImageView, CheckBox, RadioButton, EditText, ProgressBar...

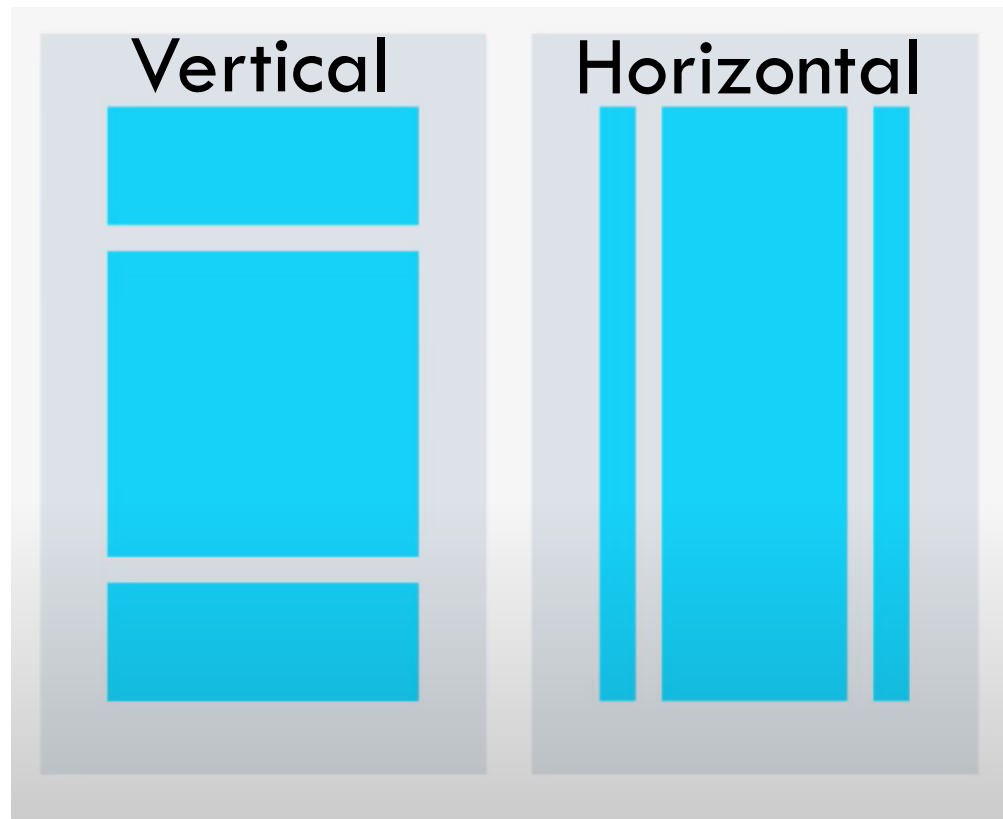
ANDROID – VIEWGROUP

Viewgroup : Uma View que contém Views.



ANDROID – LINEAR LAYOUT

Linear Layout : Viewgroup que sequencia elementos



ANDROID – SCROLLVIEW

Scroll View: Viewgroup que trabalha com rolagem



ANDROID – SCROLLVIEW

The screenshot displays the Android Studio interface with three main panels:

- Component Tree:** Shows a vertical layout containing a `TextView` with the text `"@string/nome"`, an `ImageView` with the resource `estrela_amarela`, and a `ScrollView` containing a `TextView` with the text `"@string/biografia_t..."`.
- Text Preview:** Displays a paragraph of text from a biographical string resource: "a Terra logo após a concepção de Peter, a fim de continuar sua guerra intergaláctica, deixando Meredith na Terra para protegê-la. Nove meses mais tarde, durante um fenômeno astronômico raro, quando muitos dos planetas estavam alinhados, Peter nasceu e pouco tempo depois sua mãe Meredith, por razões que ela não conseguia explicar, teve o desejo de levá-lo para fora e apresentá-lo para as estrelas. Durante dez anos, Quill foi criado por Meredith, a qual permaneceu uma mãe solteira, mas teve a ajuda de sua melhor amiga Lisa Chang e a filha dela Monica. Peter logo começou a desenvolver um grande interesse em programas de ficção científica, e após uma visita a um parque de diversões onde ele ficou impressionado com as maravilhas do espaço, ele decidiu que queria ser um astronauta quando crescesse.[1] No entanto, ele mais tarde acabou ficando um pouco desapontado após descobrir que os alienígenas no passeio não eram reais. Um dia, Peter retornou para casa, ferido por tentar impedir
- Declared Attributes:** Lists the attributes for the selected `TextView` in the `ScrollView`.

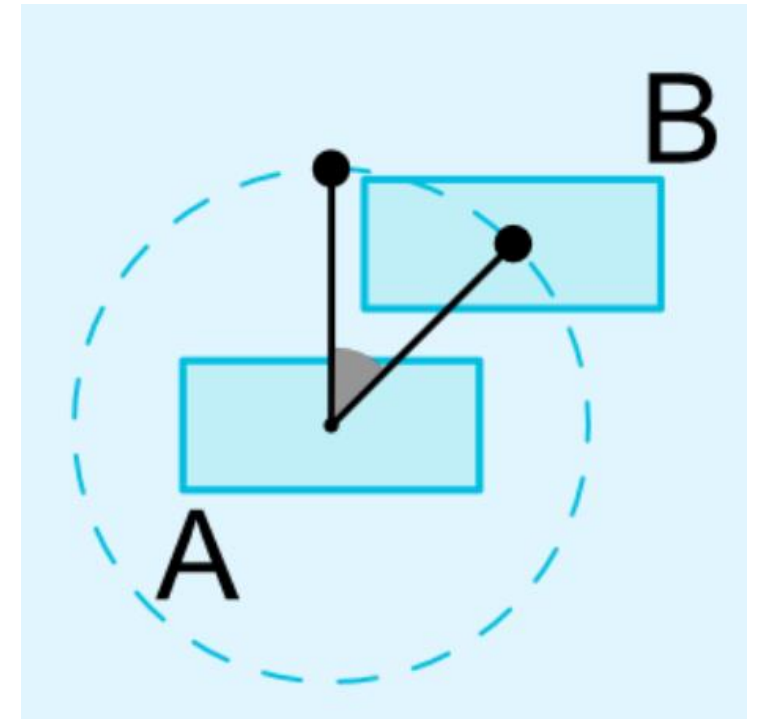
Declared Attributes		
<code>layout_width</code>	<code>match_parent</code>	
<code>layout_height</code>	<code>wrap_content</code>	
<code>layout_marginLeft</code>	<code>8sp</code>	
<code>layout_marginRi...</code>	<code>8sp</code>	
<code>id</code>	<code>bio</code>	
<code>lineSpacingMulti...</code>	<code>1.2</code>	
<code>paddingBottom</code>	<code>32sp</code>	
<code>text</code>	<code>@string/biografia_textc</code>	
<code>textAlignment</code>	<code>textStart</code>	
<code>textSize</code>	<code>18sp</code>	
Layout		
<code>layout_width</code>	<code>match_parent</code>	
<code>layout_height</code>	<code>wrap_content</code>	

```
fun inicializarViewVars() {  
    edit = findViewById(R.id.apelido_edit)  
    apelido = findViewById(R.id.apelido_text)  
    ok_button = findViewById(R.id.ok_button)  
    ok_button.setOnClickListener {atualizarApelido(it)}  
}  
  
fun atualizarApelido(view: View) {  
    if (edit.text.toString().trim().isEmpty()) return;  
  
    apelido.text = edit.text  
  
    edit.visibility = View.GONE  
    ok_button.visibility = View.GONE  
  
    apelido.visibility = View.VISIBLE  
  
    (getSystemService(Context.INPUT_METHOD_SERVICE)  
        as InputMethodManager)  
        .hideSoftInputFromWindow(view.windowToken, flags: 0)  
}
```

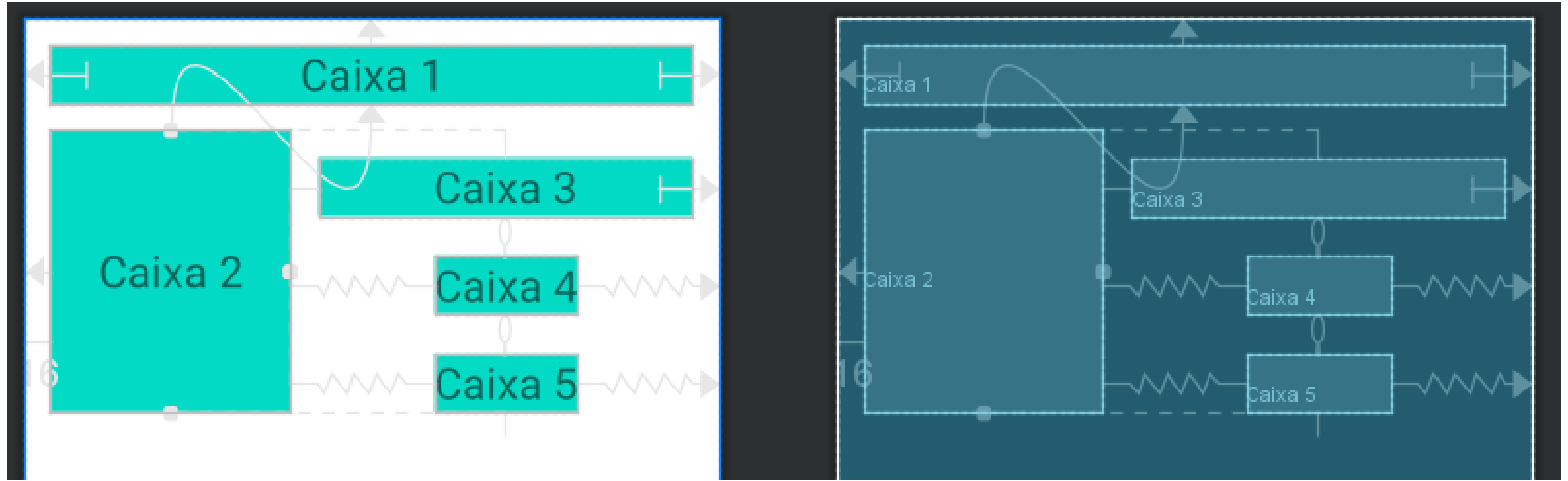

ANDROID – CONSTRAINT LAYOUT

Constraint Layout: Viewgroup que trabalha com regras (**constraints**) de posicionamento **relativo**.

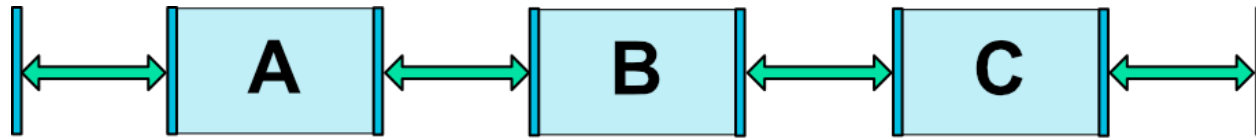
- Coloque A perto da borda esq.
- Coloque B a 45 graus de A



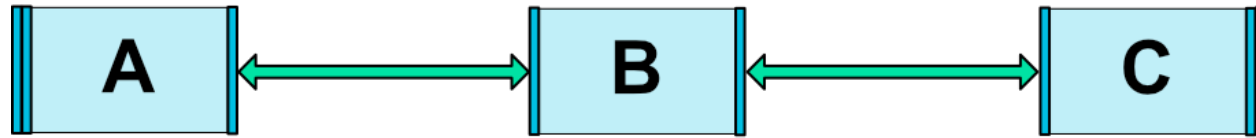
ANDROID – CONSTRAINT LAYOUT



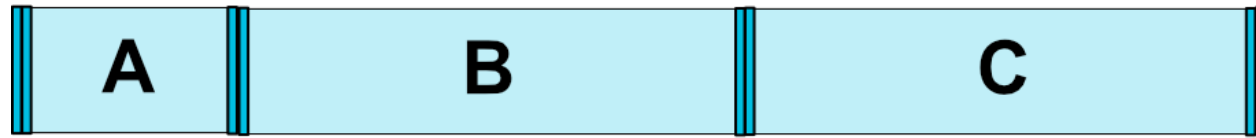
ANDROID – CONSTRAINT CHAIN



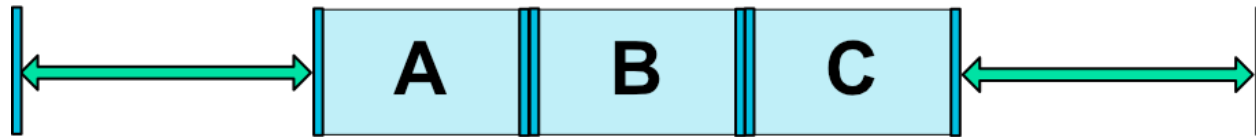
Spread Chain



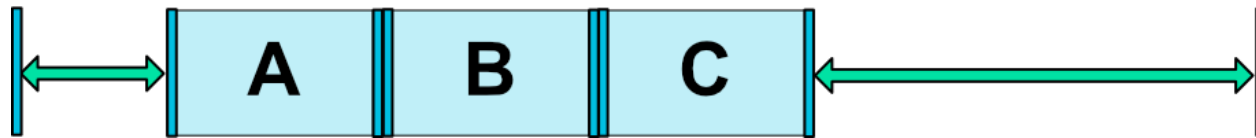
Spread Inside Chain



Weighted Chain



Packed Chain



Packed Chain with Bias

ANDROID – EXERCÍCIO

Faça as caixas começarem em branco, e fiquem colorida quando se clica nela. Dica:

```
// Boxes using Color class colors for background
R.id.box_one_text -> view.setBackgroundColor(Color.DKGRAY)
R.id.box_two_text -> view.setBackgroundColor(Color.GRAY)

// Boxes using Android color resources for background
R.id.box_three_text -> view.setBackgroundResource(android.R.color.holo_green_light)
R.id.box_four_text -> view.setBackgroundResource(android.R.color.holo_green_dark)
R.id.box_five_text -> view.setBackgroundResource(android.R.color.holo_green_light)

else -> view.setBackgroundColor(Color.LTGRAY)
```

ANDROID – EXERCÍCIO

```
fun colorirCaixa(caixa: View) {  
    // Dependendo de qual caixa pinta de uma cor diferente  
    when (caixa.id) {  
  
        //por cores definidas na classe android Color  
        R.id.Caixa1 -> caixa.setBackgroundColor(Color.DKGRAY)  
        R.id.Caixa2 -> caixa.setBackgroundColor(Color.GRAY)  
  
        //pode-se usar também cores no pacote de recursos básico  
        R.id.Caixa3 -> caixa.setBackgroundResource(android.R.color  
■ .holo_green_light)  
        R.id.Caixa4 -> caixa.setBackgroundResource(android.R.color  
■ .holo_green_dark)  
        R.id.Caixa5 -> caixa.setBackgroundResource(android.R.color  
■ .holo_blue_light)  
  
        else -> caixa.setBackgroundColor(Color.YELLOW)  
    }  
}
```

ANDROID – VIEW BINDING

No build.gradle de módulo

```
android {  
    buildFeatures {  
        viewBinding true  
    }  
}
```

ANDRO

```
lateinit var caixas: List<TextView>
lateinit var binding: ActivityMainBinding
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    binding = ActivityMainBinding.inflate(layoutInflater)
    val view = binding.root
    setContentView(view)
    //setContentView(R.layout.activity_main)

    caixas = listOf(
        binding.Caixa1,
        binding.Caixa2,
        binding.Caixa3,
        binding.Caixa4,
        binding.Caixa5,
    )
    for (caixa in caixas)
        caixa.setOnClickListener{colorirCaixa(it)}

    binding.layouti.setOnClickListener{colorirCaixa(it)}
```

ANDROID STUDIO – ATALHOS ÚTEIS

Aceitar sugestões de mudança: ALT + ENTER

Reformatar código: CTRL + ALT + L

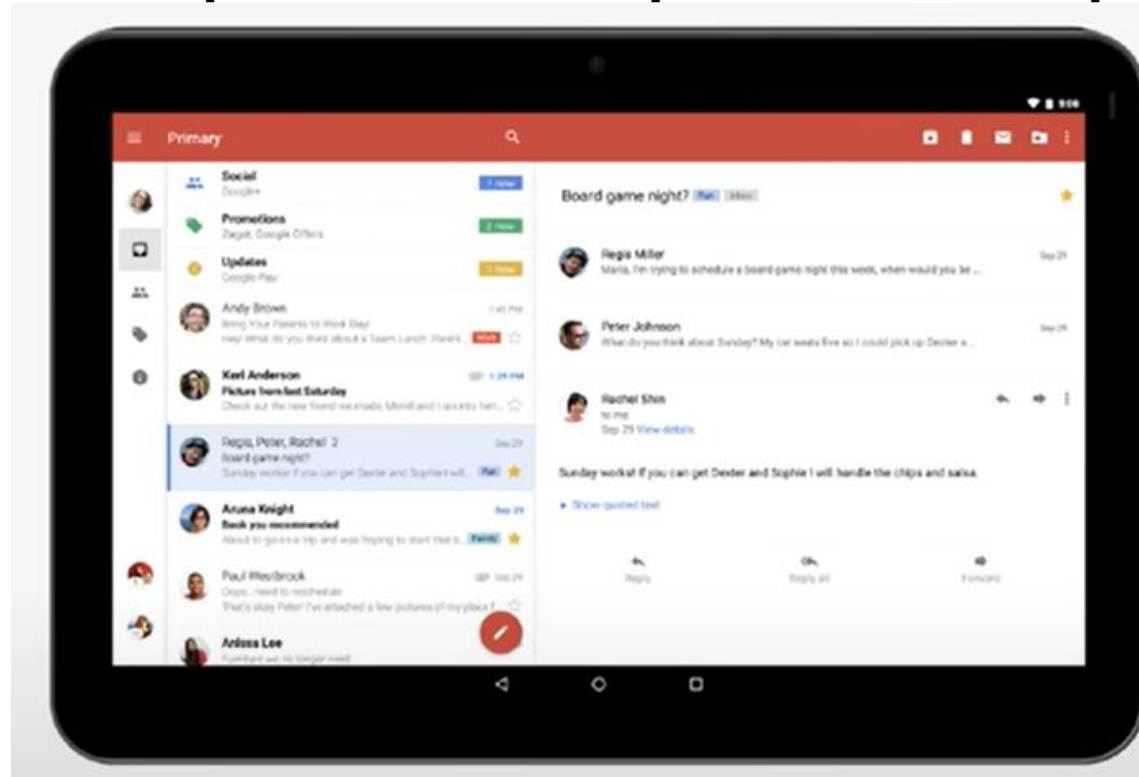
Otimizar Imports: CTRL + ALT + O

Abrir sugestões de code-complete: CTRL + SPACE

Comentar seleção ou linha: CTRL + /

ANDROID – FRAGMENTS

Fragments: Uma Activity, só que ao invés de representar a tela toda pode representar apenas uma parte da tela.

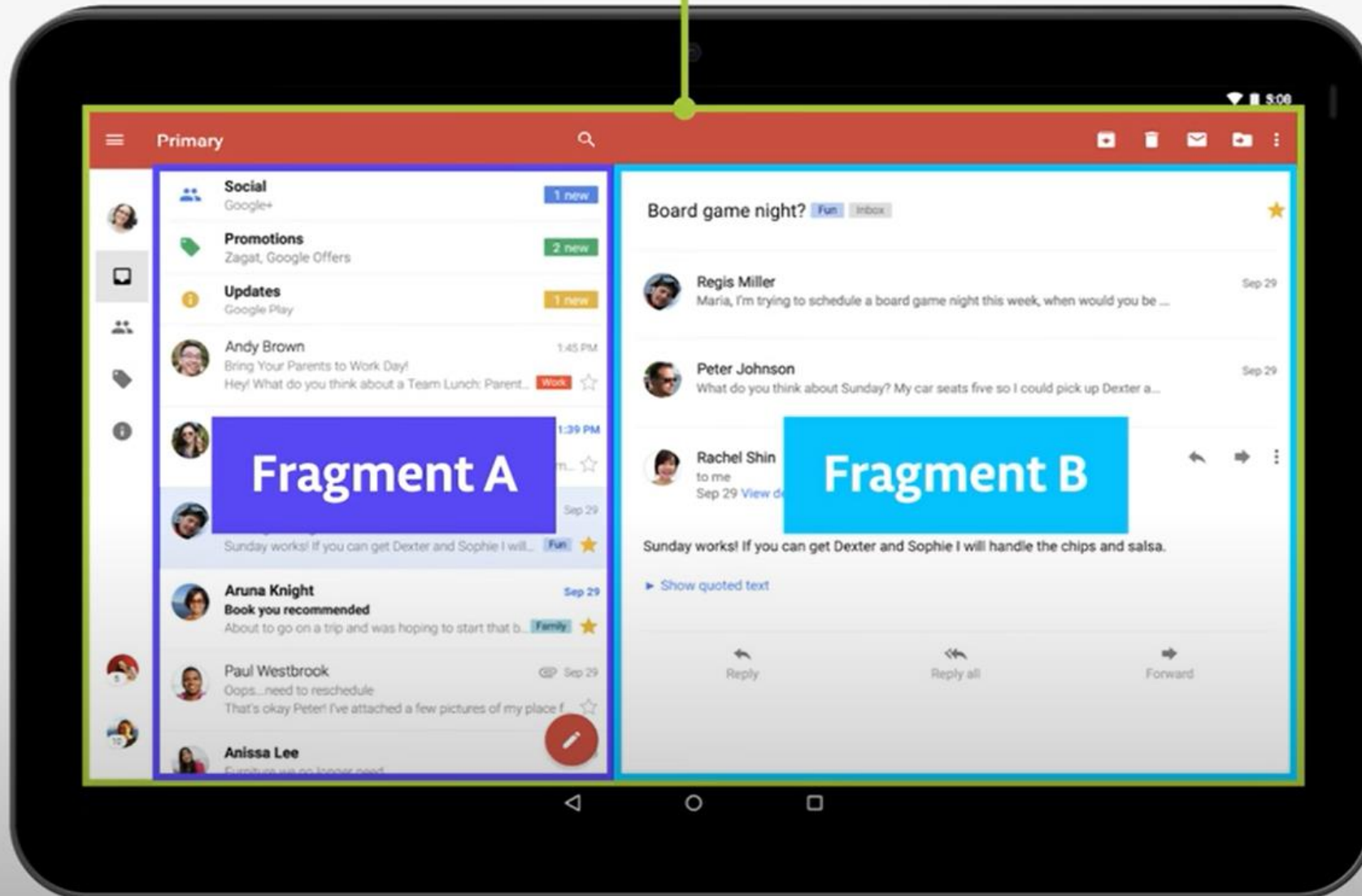


ANDR

Fragment

tela to

Activity



entar a
ela.

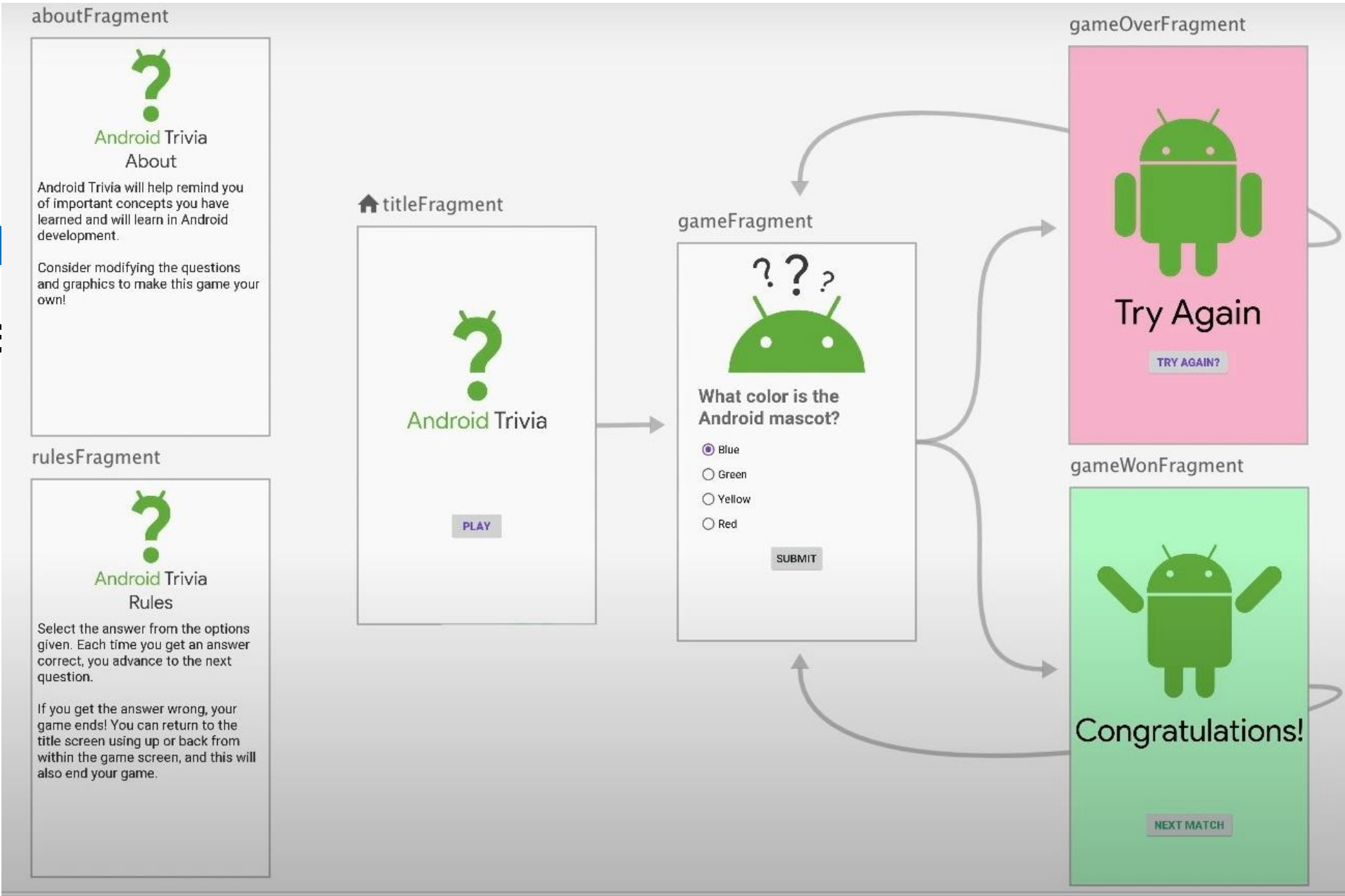
ANDROID – FRAGMENTS

Fragments: Também oferece recursos de troca de tela e navegação entre telas.

A N D

Fragment

have

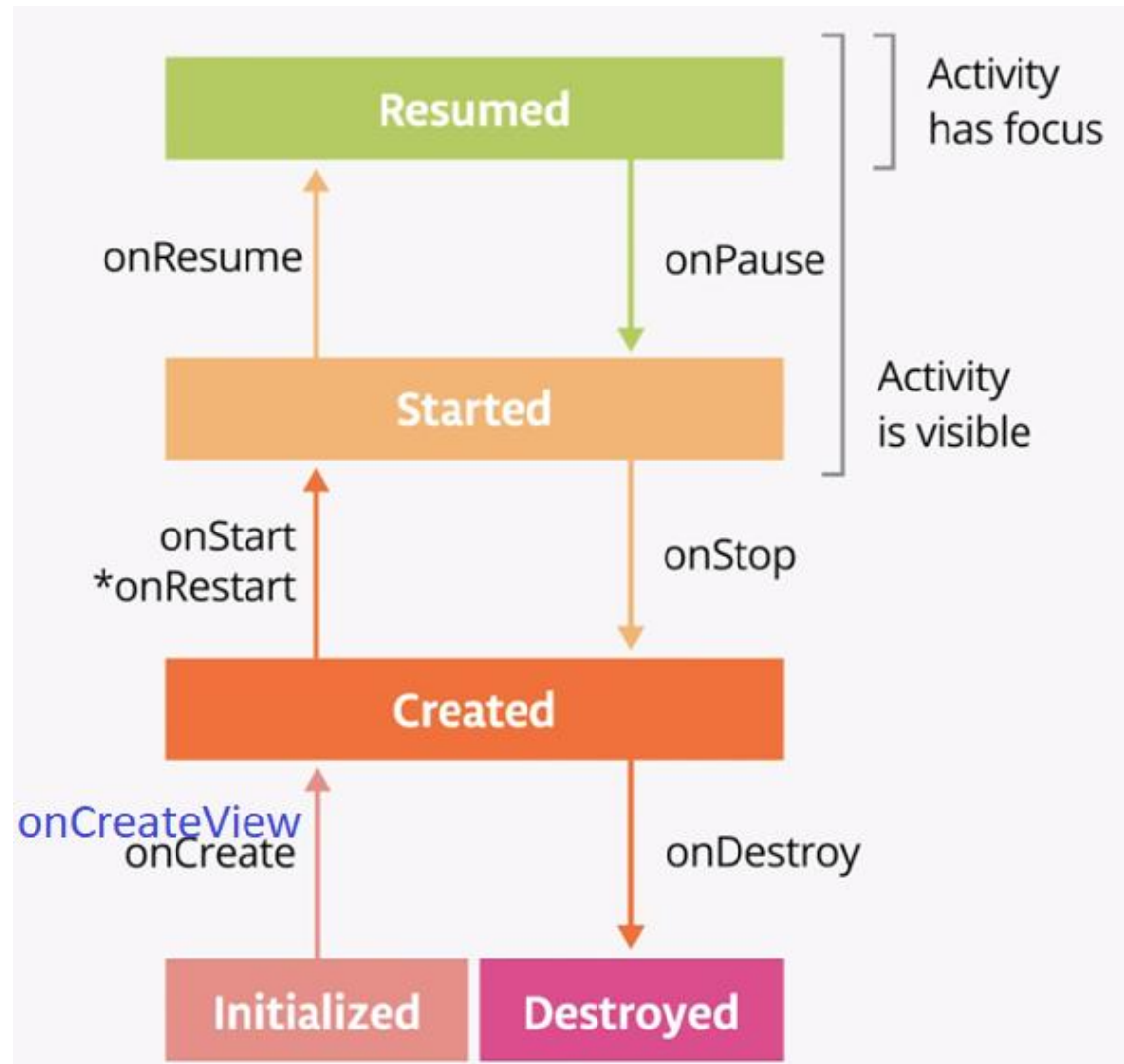


the

ANDROID – FRAGMENTS

Fragments: Usando Fragments a Activity representa seu aplicativo (o ponto de partida) e Fragmentos as suas telas.

ANDROID – CICLO DE VIDA



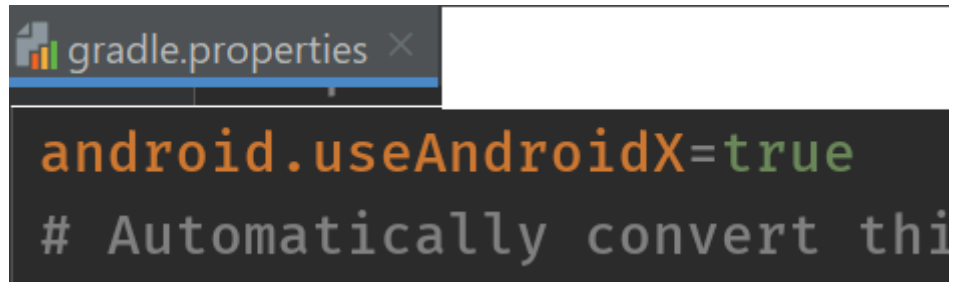
ANDROID – FRAGMENTOS

No build.gradle de módulo

```
dependencies {  
  
    //..  
  
    def fragment_version = "1.2.5"  
  
    // Java language implementation  
    implementation "androidx.fragment:fragment:$fragment_version"  
    // Kotlin  
    implementation "androidx.fragment:fragment-ktx:$fragment_version"  
}
```

ANDROID – FRAGMENTOS

(Também verifique o `gradle.properties`)

A screenshot of a code editor window titled 'gradle.properties'. The window shows two lines of code: 'android.useAndroidX=true' and a comment '# Automatically convert thi'. The text is color-coded: 'android' is orange, 'useAndroidX' is green, and 'true' is blue. The comment is in a lighter gray font.

```
gradle.properties ×  
android.useAndroidX=true  
# Automatically convert thi
```


ANDROID – FRAGMENTOS

Vamos criar um Fragmento novo...

ANDROID – FRAGMENTOS

Carregando o fragmento direto pelo layout

A N

Co

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.fragment.app.FragmentContainerView
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:tools="http://schemas.android.com/tools"
```

```
android:id="@+id/fragment"
```

```
android:name="com.example.fragmentoss.StartFragment"
```

```
tools:context=".MainActivity"
```

```
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
```

```
/>
```

ANDROID – FRAGMENTOS

Carregando o fragmento no código da Activity

ANDROID – FRAGMENTOS

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        supportFragmentManager.commit { this: FragmentTransaction  
            setReorderingAllowed(true)  
            add<StartFragment>(R.id.fragmentContainer)  
        }  
    }  
}
```

ANDROID – FRAGMENTOS

É recomendado verificar se o fragmento já foi criado

ANDROID – FRAGMENTOS

```
class MainActivity : AppCompatActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        if (savedInstanceState == null) {  
            supportFragmentManager.commit { this: FragmentTransaction  
                setReorderingAllowed(true)  
                add<StartFragment>(R.id.fragmentContainer)  
            }  
        }  
    }  
}
```

ANDROID – NAVEGAÇÃO MODERNA

A Navegação é feita pelo Trio de Componentes:

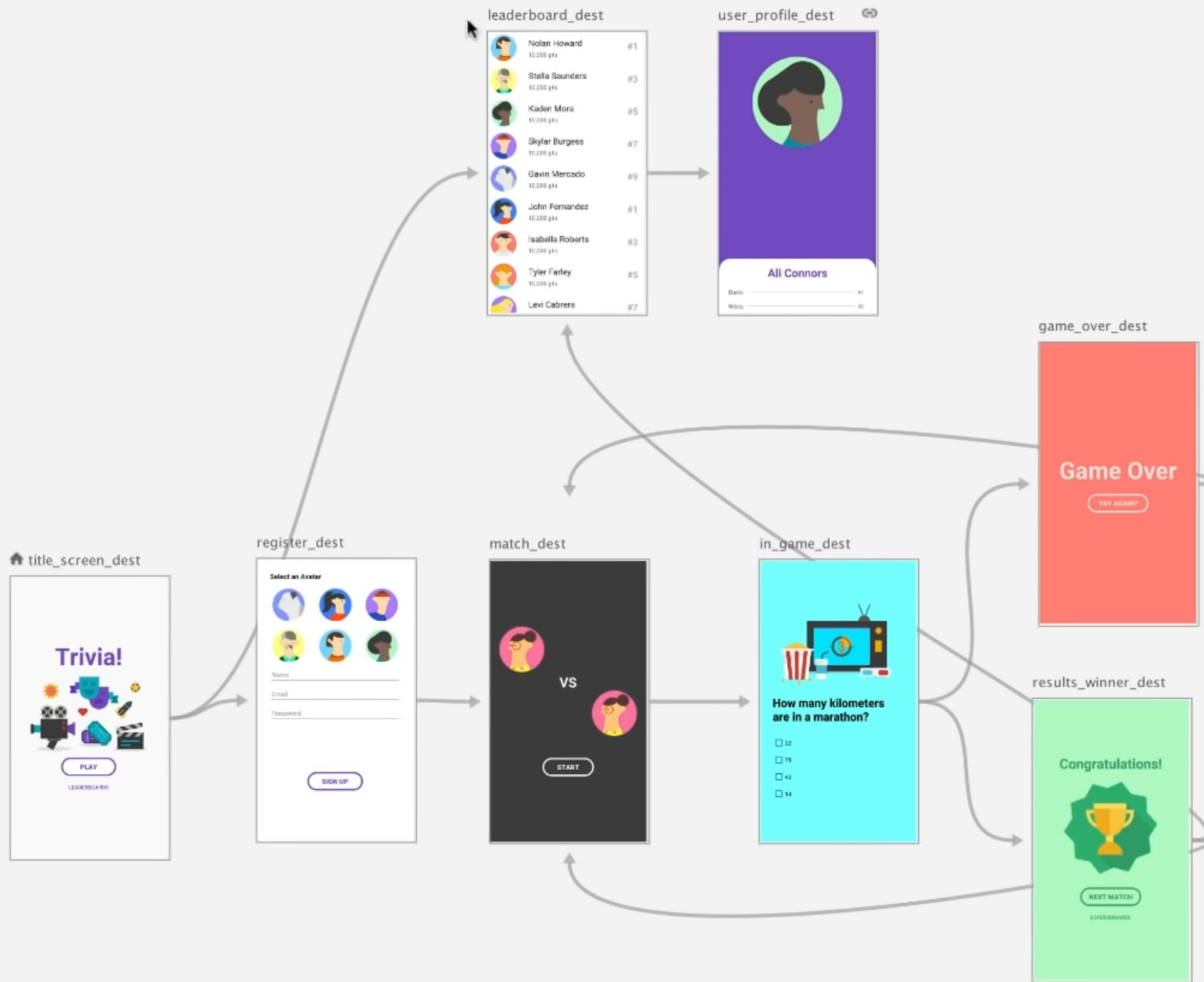
- Navigation Graph: Um XML que demonstra os caminhos do aplicativo.
- NavHostFragment: Um Fragmento-Layout que pode ser preenchido com uma tela
- NavController: Um Objeto que oferece comandos de troca de tela e navegação

ANDROID – NAVEGAÇÃO MODERNA

Navigation Graph: As telas são destinos, e cada caminho de uma tela para outra é chamado de ação

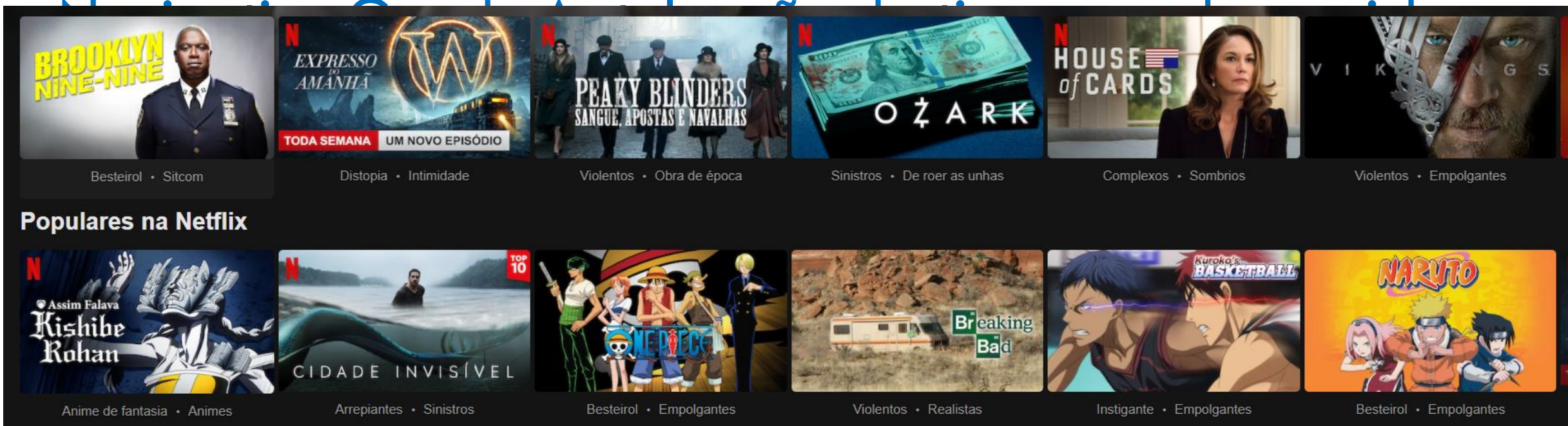
ANDRO

Naviga de umc



inho

ANDROID – NAVEGAÇÃO MODERNA



A

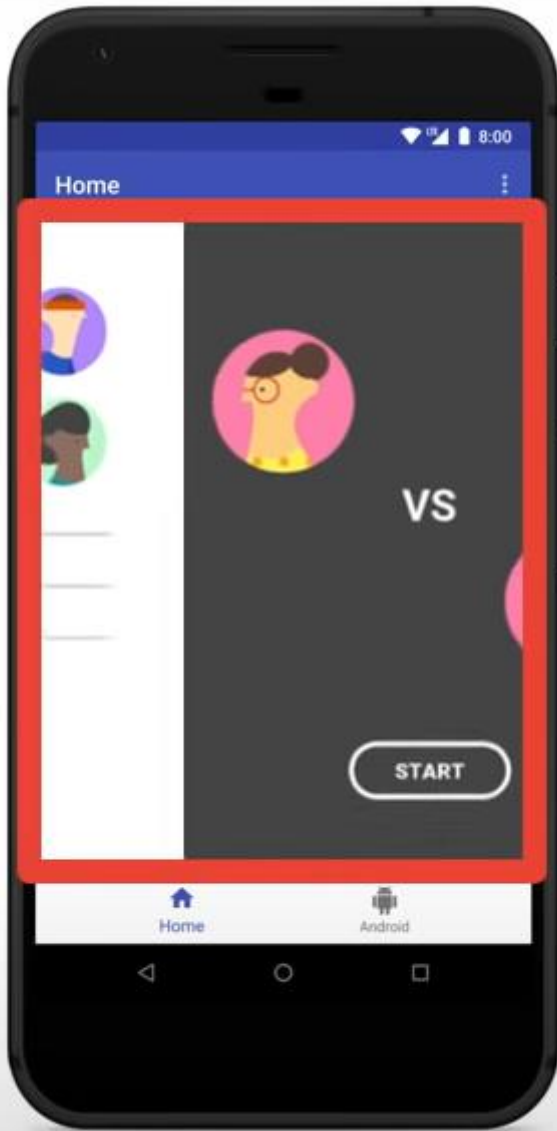
U

c



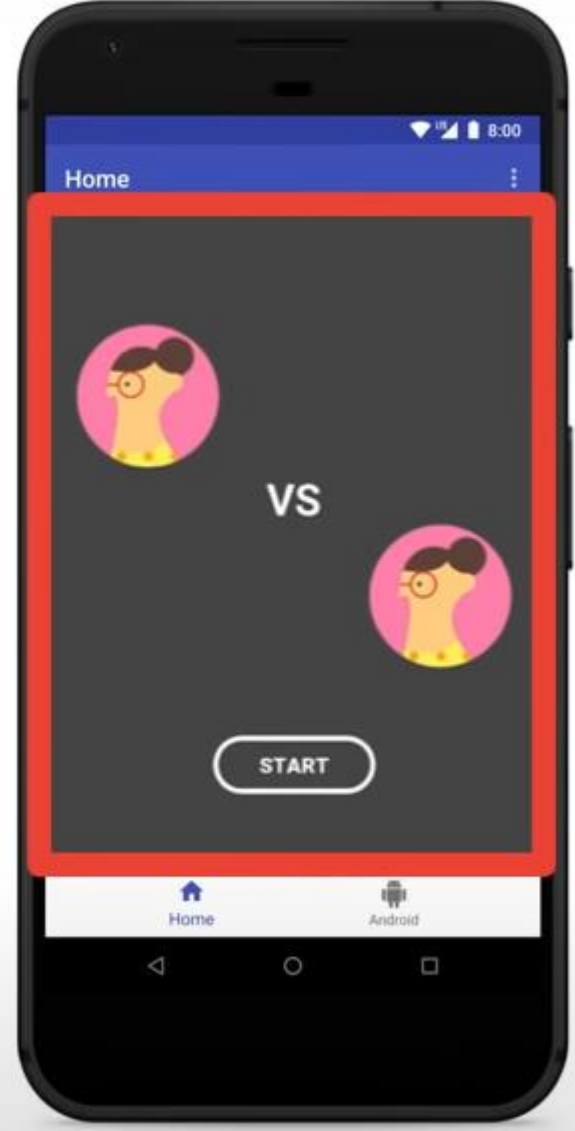
A

U



U

U



om

ANDROID – NAVEGAÇÃO MODERNA

NavHostFragment: Uma tela “vazia” que é preenchida com outras telas



ANDROID – NAVEGAÇÃO MODERNA

NavController: possui os métodos de troca de tela e navegação



```
findNavController().navigate(<Destination or Action id>)
```

ANDROID – NAVEGAÇÃO MODERNA

No build.gradle de PROJETO

buildscript { to view and edit your project configuration

Open (Ctrl+Alt+Shift+S) Hide notification

```
jcenter()
}
dependencies {
    classpath "com.android.tools.build:gradle:4.1.2"
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

    def nav_version = "2.3.3"
    classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
    // NOTE: Do not place your application dependencies here; they belong
    // in the individual module build.gradle files
}
```

buildscript {

ANDROID – NAVEGAÇÃO MODERNA

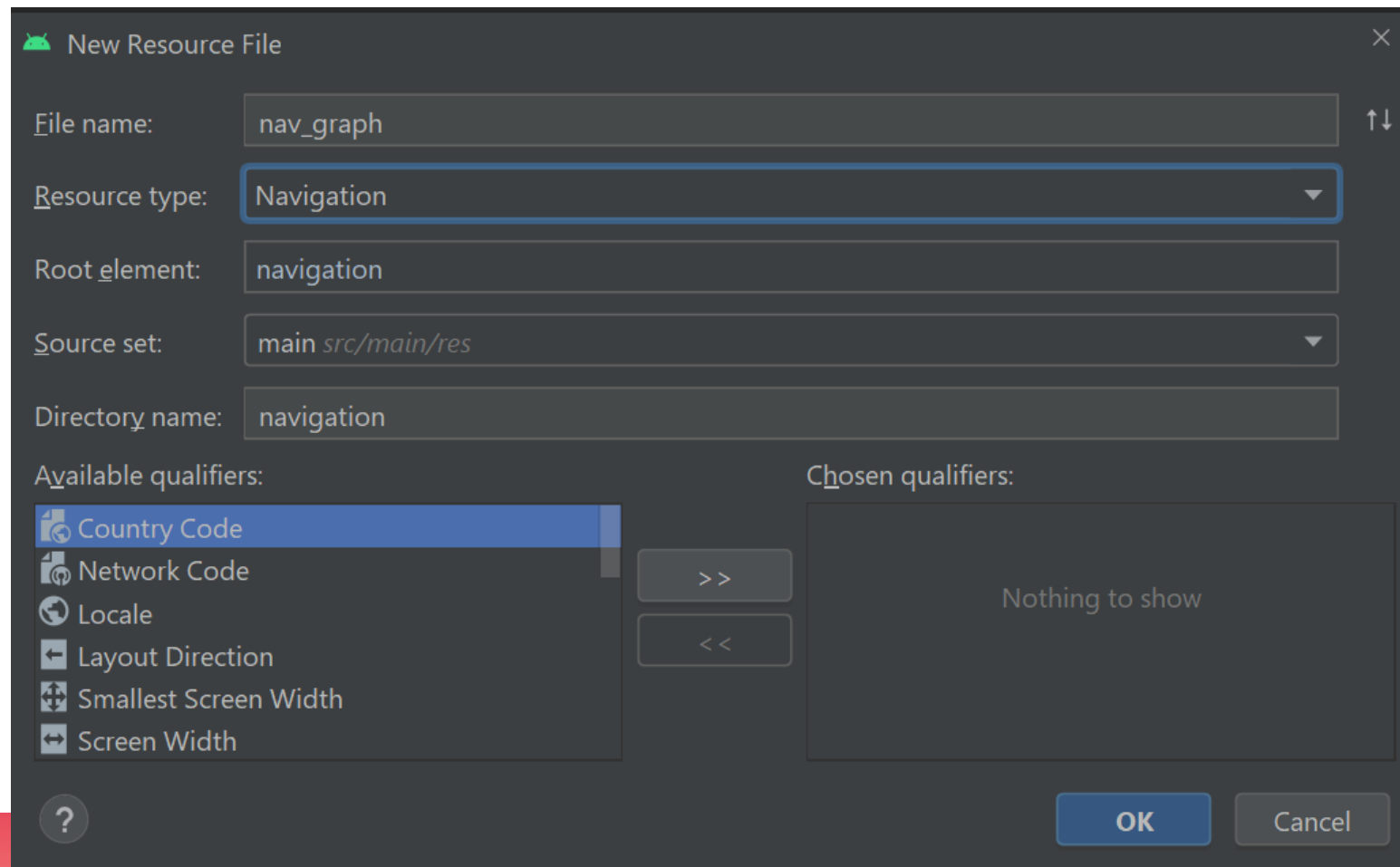
No build.gradle de módulo


```
def nav_version = "2.3.3"
// Java language implementation
implementation "androidx.navigation:navigation-fragment:$nav_version"
implementation "androidx.navigation:navigation-ui:$nav_version"
// Kotlin
implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
// Feature module Support
implementation "androidx.navigation:navigation-dynamic-features-fragment
:$nav_version"
// Testing Navigation
androidTestImplementation "androidx.navigation:navigation-testing:$nav_version"
// Jetpack Compose Integration
implementation "androidx.navigation:navigation-compose:1.0.0-alpha07"
```

```
apply plugin: "androidx.navigation.safeargs.kotlin"
```

ANDROID – NAVEGAÇÃO MODERNA

1. Vamos criar o NavGraph



ANDROID – NAVEGAÇÃO MODERNA

1. Vamos criar o NavGraph

1. Do Editor adicione o `startFragment` como destino
2. E em seguida, na tela de novo destino use a opção de criar novo fragmento para criarmos outra tela
3. Interligue as duas telas

ANDROID

RNA

1. Vamos

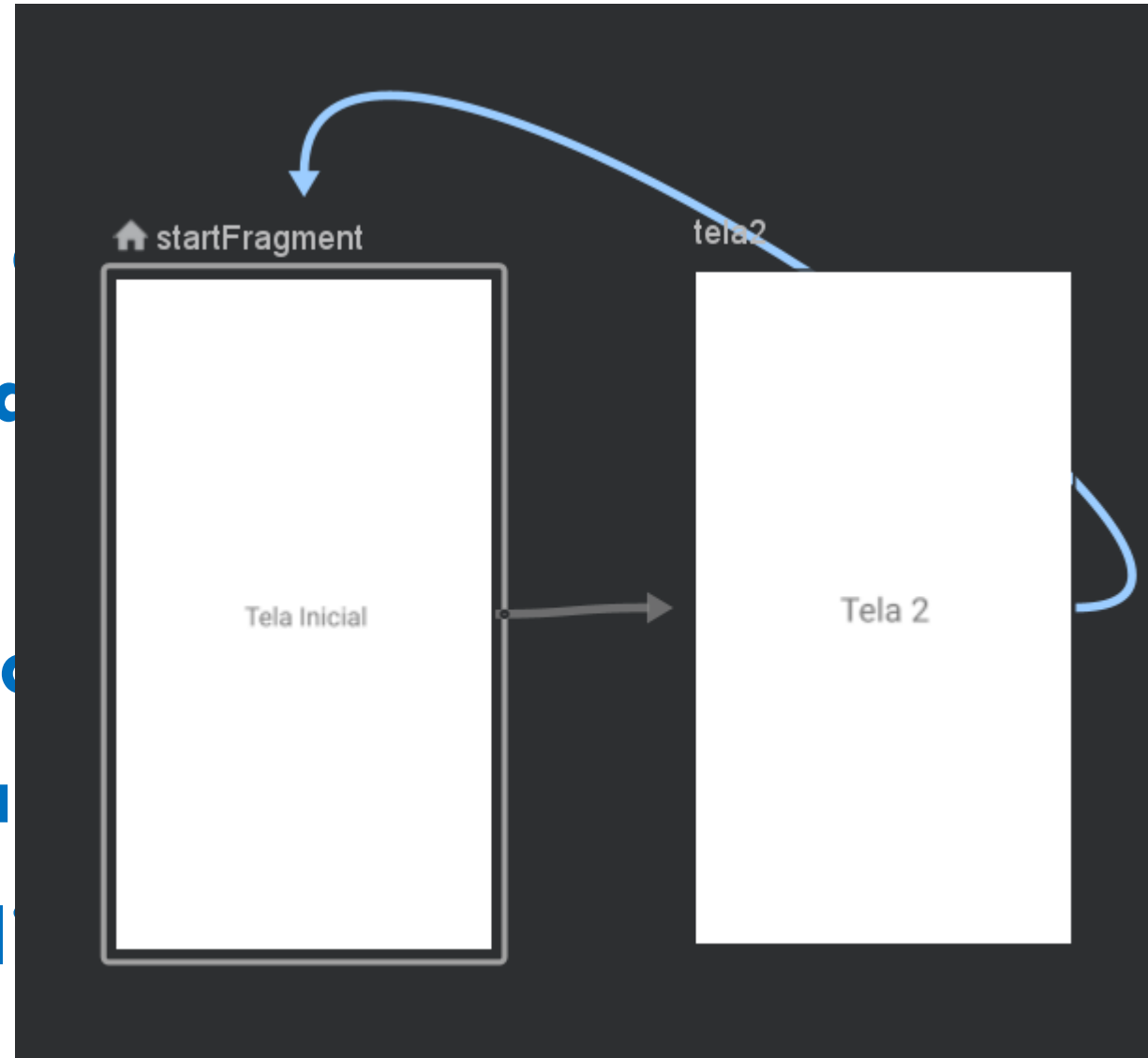
1. Do Ec

2. E em

opção

outra

3. Interl



como destino

no use a

criarmos

ANDROID – NAVEGAÇÃO MODERNA

2. O NavHost será o Layout carregado pela Main Activity

A N D

2. 0

tivity

```
<androidx.fragment.app.FragmentContainerView
```

```
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:id="@+id/navHost"
```

```
tools:context=".MainActivity"
```

```
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
```

```
android:name="androidx.navigation.fragment.NavHostFragment"
```

```
app:defaultNavHost="true"
```

```
app:navGraph="@navigation/nav_graph" />
```

ANDROID – NAVEGAÇÃO MODERNA

3. Agora podemos usar o NavController

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    super.onCreateView(view, savedInstanceState)  
  
    val botao: Button = view.findViewById(R.id.botao)  
  
    botao.setOnClickListener { it: View!  
        findNavController().navigate(R.id.action_startFragment_to_tela2)  
    }  
}
```

ANDROID – NAVEGAÇÃO MODERNA

3. O plugin de safeArgs permite usar ações

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    super.onCreateView(view, savedInstanceState)  
  
    val botao: Button = view.findViewById(R.id.botao)  
  
    botao.setOnClickListener { it: View!  
        val ação = StartFragmentDirections.actionStartFragmentToTela2()  
        findNavController().navigate(ação)  
    }  
}
```


ANDROID – VIEW BINDING EM FRAGMENTOS

```
class StartFragment : Fragment() {  
  
    private var _binding: FragmentStartBinding? = null  
    private val binding get() = _binding!!  
  
    override fun onCreateView(  
        inflater: LayoutInflater, container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        // Inflate the layout for this fragment  
        _binding = FragmentStartBinding.inflate(inflater, container,  
attachToParent: false)  
  
        return binding.root  
    }  
}
```

ANDROID – VIEW BINDING EM FRAGMENTOS

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    val ação = StartFragmentDirections.actionStartFragmentToTela2()  
    findNavController().navigate(ação)  
}  
  
binding.botaoTelaInicial2  
    .setOnClickListener { it: View!  
        val ação = StartFragmentDirections.actionStartFragmentToTela32()  
        findNavController().navigate(ação)  
    }  
}
```

ANDROID – VIEW BINDING EM FRAGMENTOS

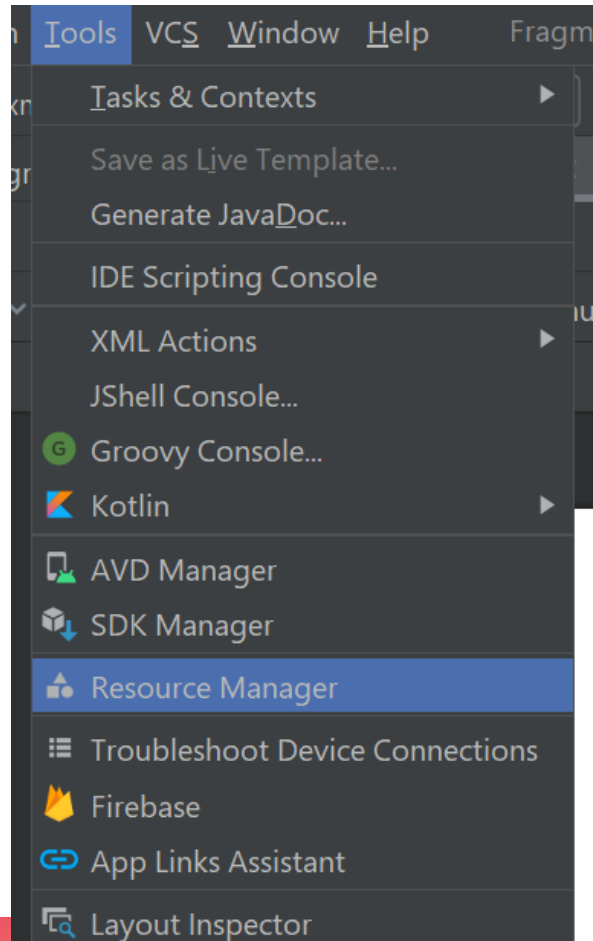
```
override fun onDestroyView() {  
    super.onDestroyView()  
    _binding = null  
}
```

ANDROID – VIEW BINDING EM FRAGMENTOS

```
override fun onDestroyView() {  
    super.onDestroyView()  
    _binding = null  
}
```

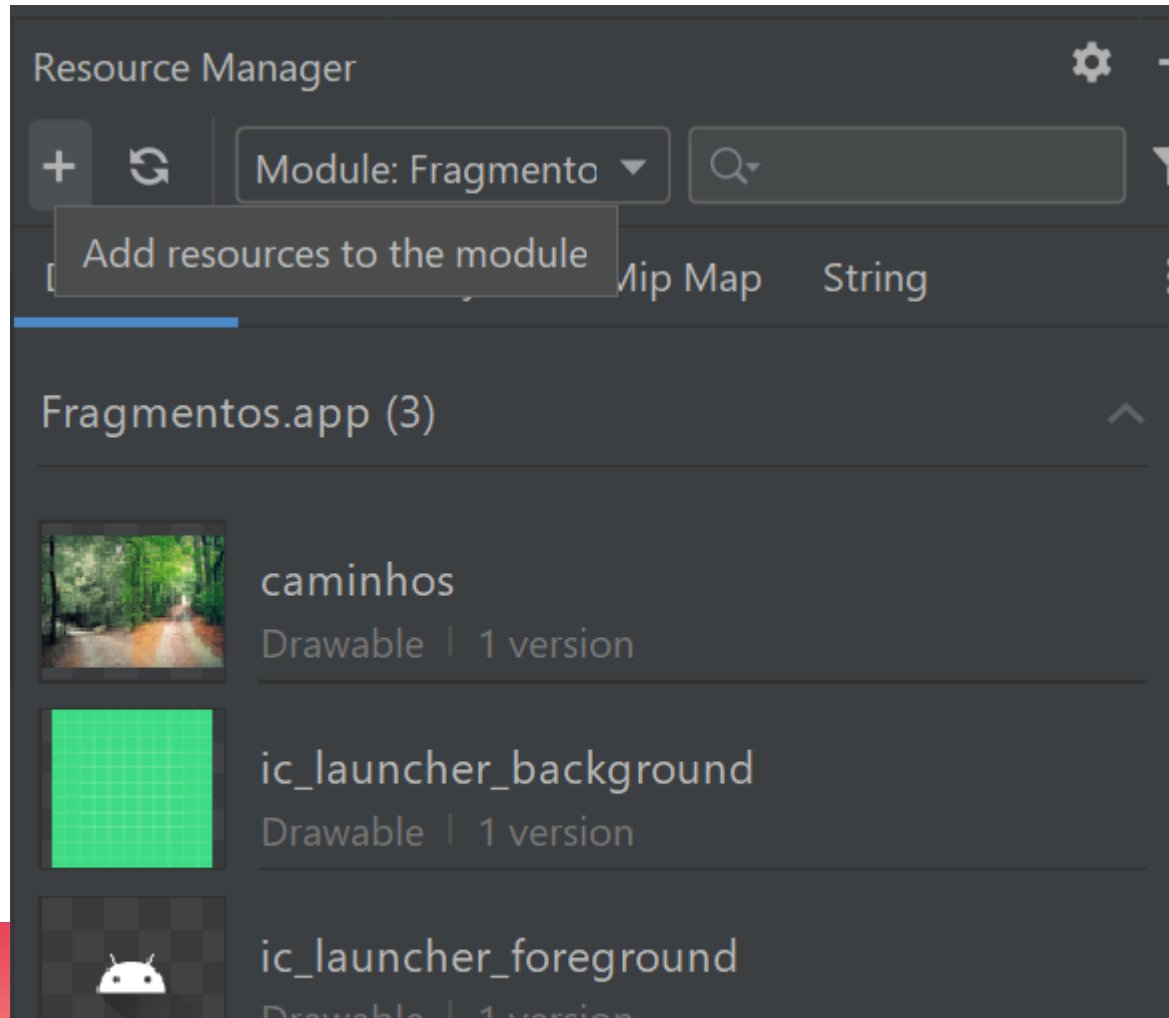
ANDROID – ADICIONANDO IMAGEM

1. Tools -> Resource Manager



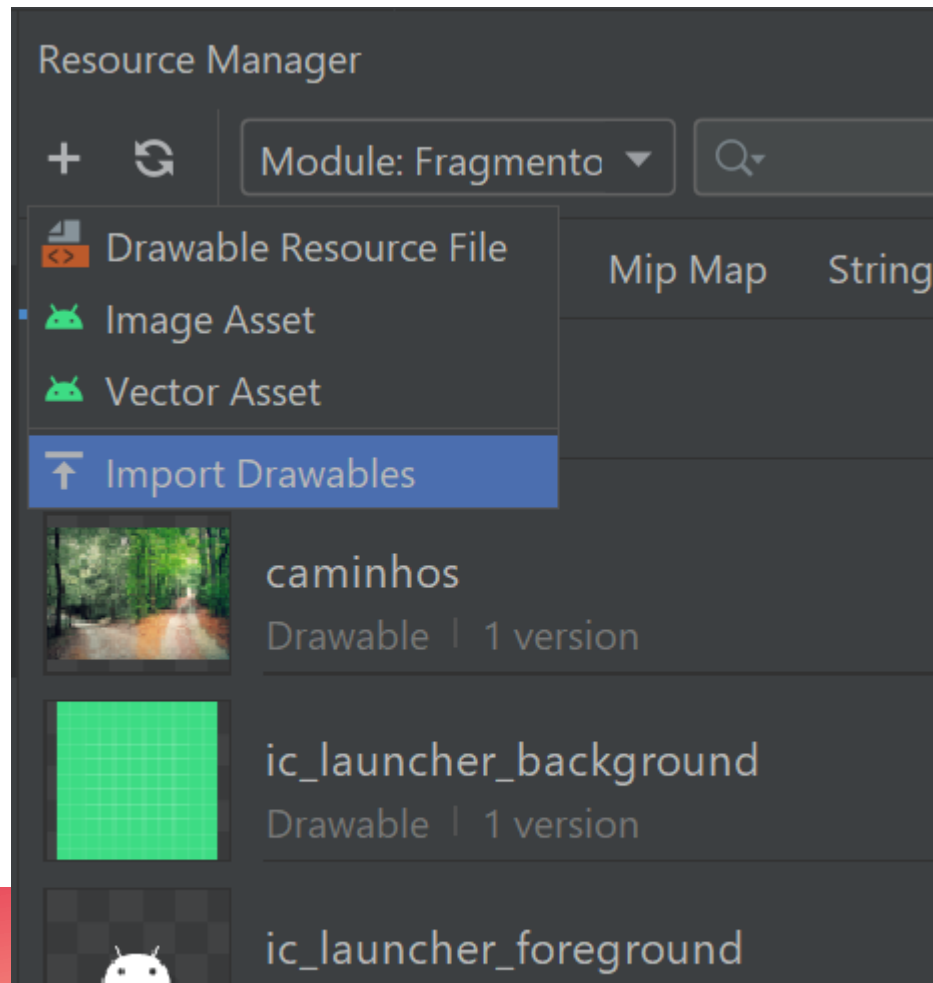
ANDROID – ADICIONANDO IMAGEM

2. Clica no +



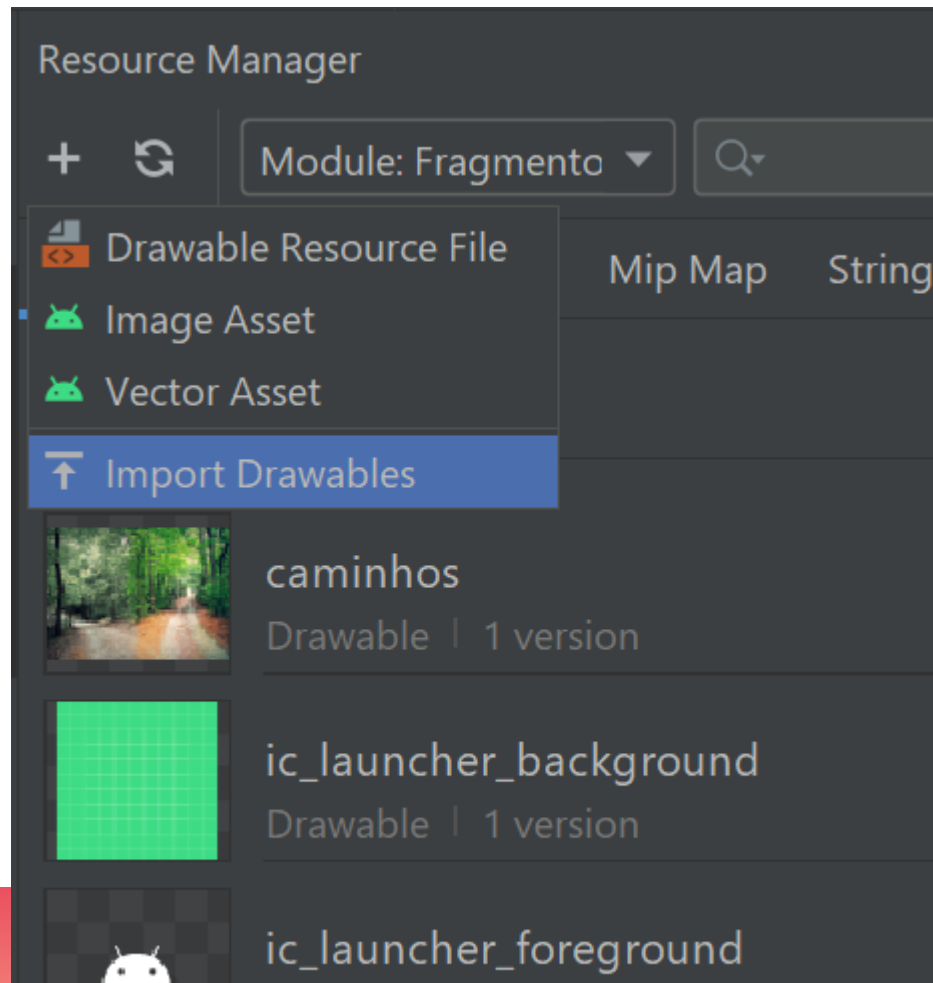
ANDROID – ADICIONANDO IMAGEM

3. “Import Drawables”



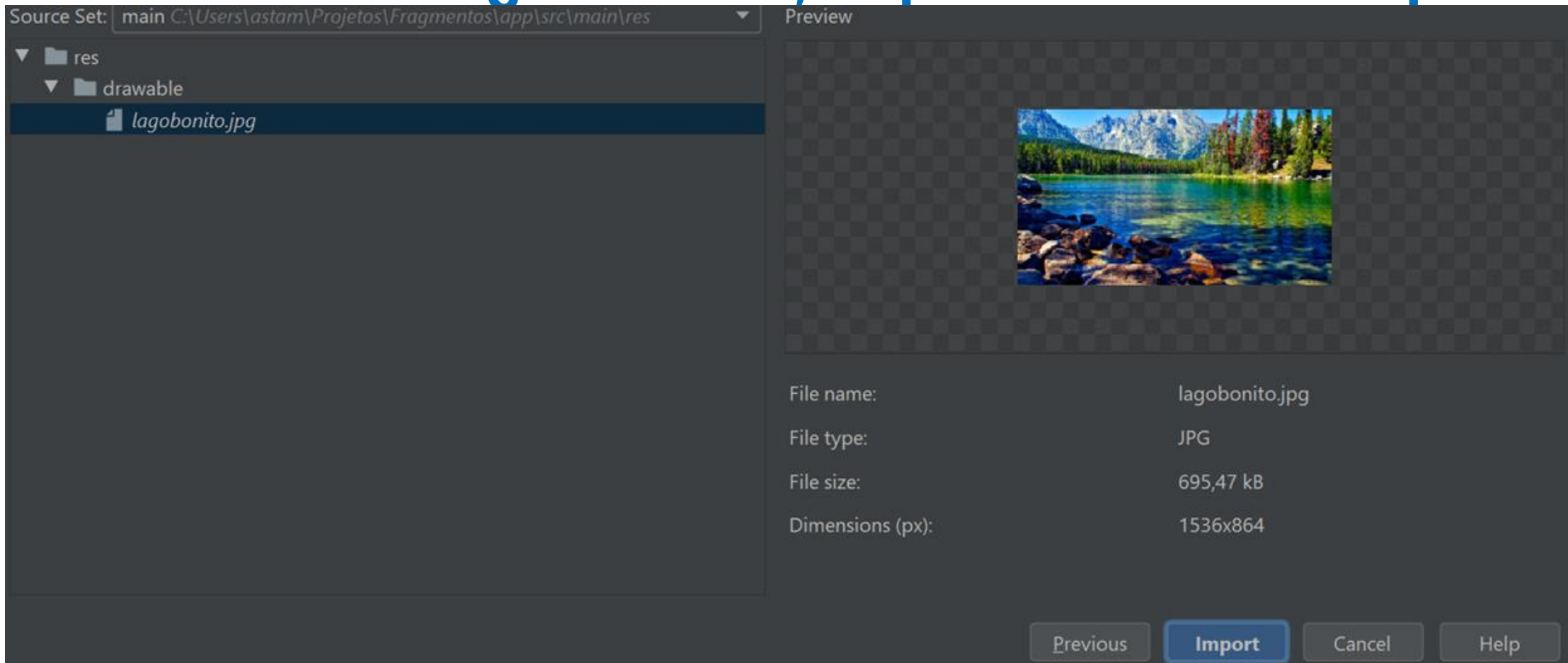
ANDROID – ADICIONANDO IMAGEM

3. “Import Drawables”



ANDROID – ADICIONANDO IMAGEM

4. Escolhe a imagem no PC, depois “Next” e “Import”



ANDROID – ADICIONANDO IMAGEM

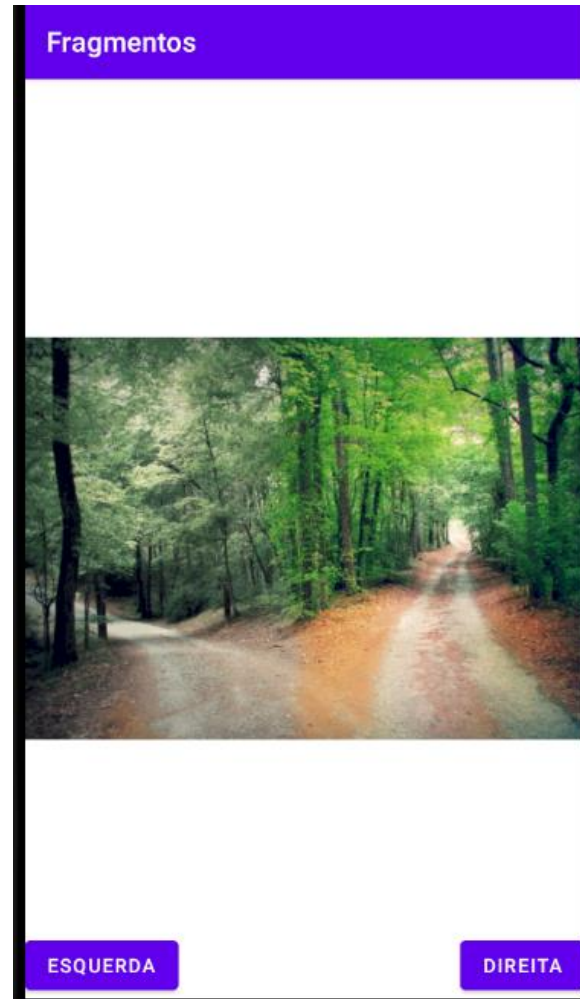
5. No Layout desejado arraste ImageView para janela de elementos

ANDROID – ADICIONANDO IMAGEM

6. Em código de layout a imagem é carregada com esse `android:src`

```
<ImageView
    android:id="@+id/imageView2"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:src="@drawable/lagobonito"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

ANDROID – EXERCÍCIO DE NAVEGAÇÃO



ANDROID – RECYCLER VIEW

ANDRO

