

The background features a complex network of thin grey lines and dots, forming a web-like structure. Scattered throughout are various triangles of different sizes and orientations, some with solid black dots at their vertices. The overall aesthetic is modern and technical.

# **PROGRAMA DEV VENTURE**

---

Desenvolvimento Android

**FRAGMENTS**  
O que são fragments

**01**

**NAVIGATION**  
Navigation Component

**02**

# Aula 06 & 07

## AGENDA

**TEMAS & ESTILOS**  
Estilizando Lance Dados

**03**



# 01

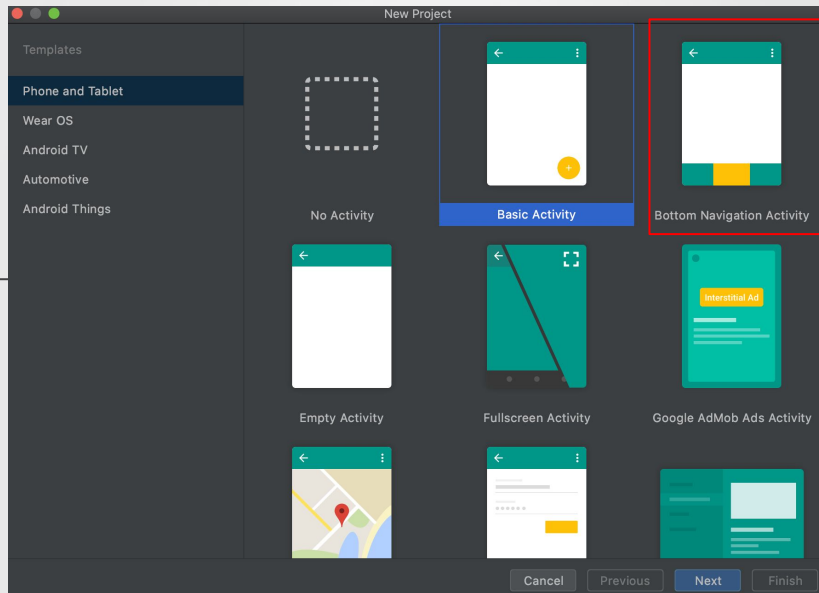
## FRAGMENTS

---



# dojo | FRAGMENTS

**Crie um novo projeto:** Teste fragments a partir de uma BottomNavigation Activity





# *dojo* | FRAGMENTS

---

- Quantas telas este app possui?
  - Quantas activities?
  - Em quais arquivos xml os botões next e previous foram definidos?
  - Com quais classes os arquivos xml dos botões foram associados?
  - Onde está definida a ação do FAB?
- 





# FRAGMENT

---

Tanto a **Activity** como o **Fragment** são componentes que fornecem uma interface gráfica(UI), para permitir que o usuário interaja com a aplicação.

A diferença fundamental é que um **Fragment necessita de uma Activity para ser apresentado.**

---





# FRAGMENT

---

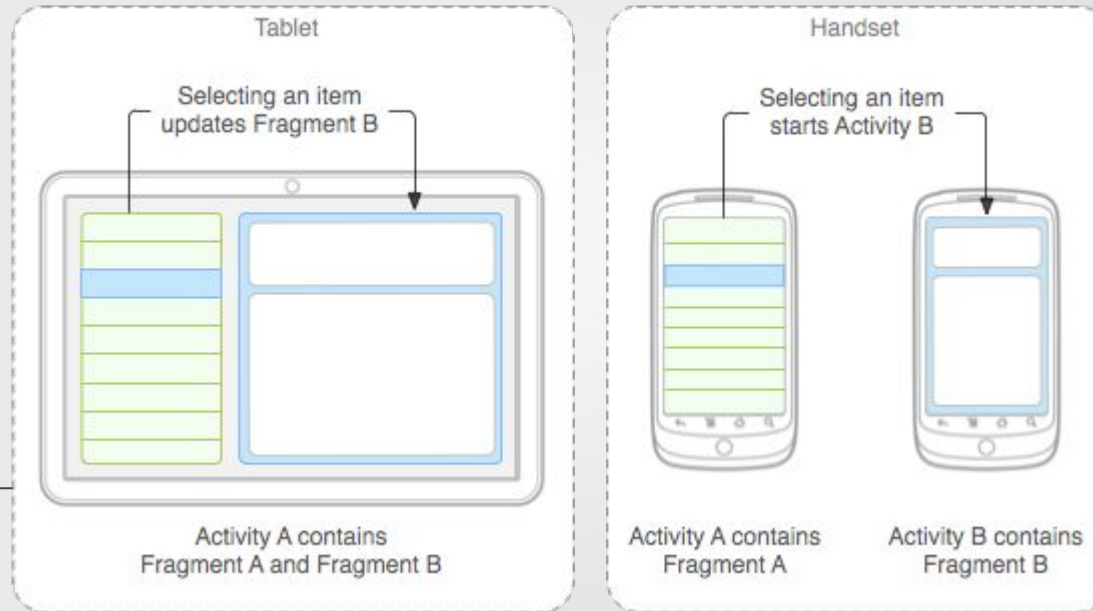
O Android introduziu os fragmentos no Android 3.0 (API de nível 11) **principalmente para suportar mais projetos de IU flexíveis e dinâmicos em telas grandes, como em tablets.** Como a tela de um tablet é muito maior que a de um celular, há mais espaço para combinar e alternar componentes da IU.

**Os fragments serviam então como "mini activities"**

---

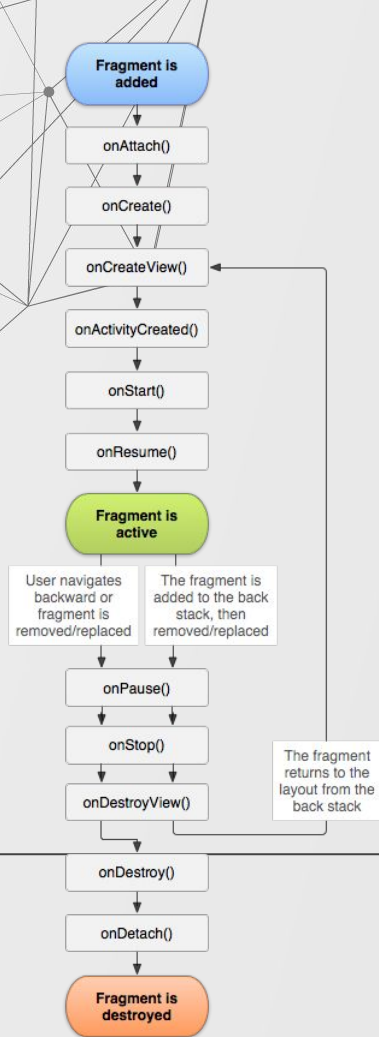


# FRAGMENT





# FRAGMENT



O ciclo de vida de um fragment está diretamente conectado ao ciclo de vida da sua activity "mãe", com algumas particularidades.



# VIEW BINDING

---

A vinculação de visualizações é ativada em um módulo por base de módulo.  
Para ativar a vinculação de visualizações em um módulo, adicione o elemento `viewBinding` ao arquivo `build.gradle` dele,

---



02

NAVIGATION

---





# NAVIGATION

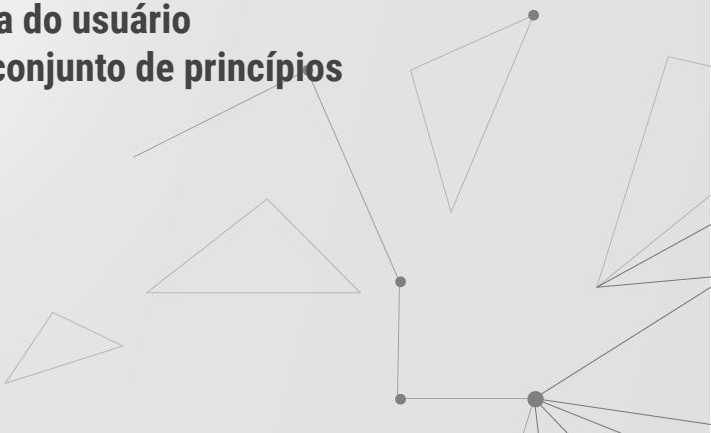
---

A navegação se refere às interações que permitem aos usuários navegar, entrar e sair de diferentes partes do conteúdo no aplicativo.

O **componente de navegação do Android Jetpack** ajuda você a implementar a navegação.

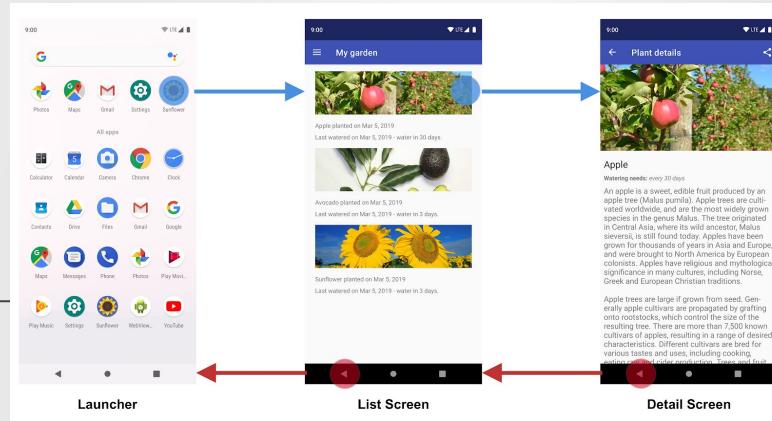
**Esse componente também garante uma experiência do usuário consistente e previsível por meio da adesão a um conjunto de princípios estabelecido.**

---



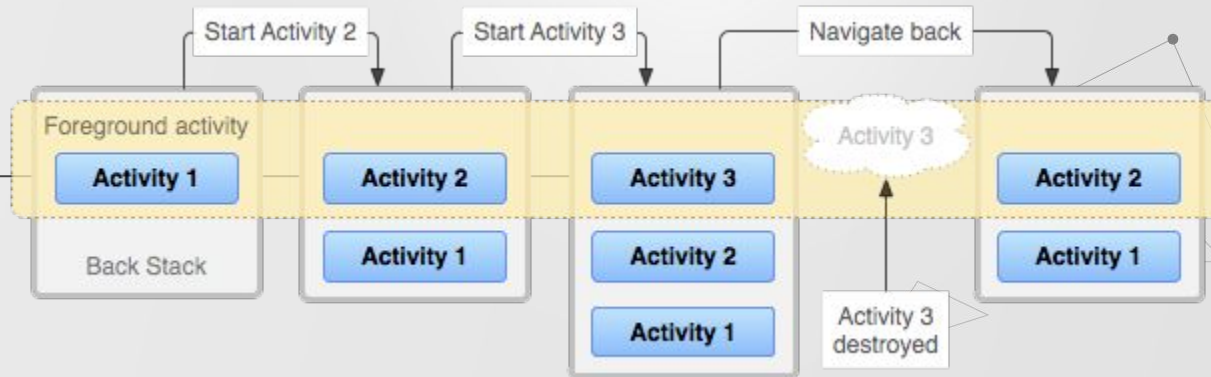
# DESTINO INICIAL FIXO

Todo app tem um destino inicial fixo. Essa é a **primeira tela** que o usuário vê ao abrir o app da tela de início. Esse destino também é a **última tela** que o usuário vê quando retorna à tela de início após pressionar o botão "Voltar".



# PILHA DE DESTINOS

Quando o app for iniciado pela primeira vez, uma nova *task* será criada para o usuário e o app exibirá o destino inicial. Esse se tornará o destino base do que é conhecido como *pilha de retorno (back stack)* e é a base para o estado de navegação do app. O topo da pilha é a tela atual, e os destinos anteriores na pilha representam o histórico de onde você esteve. A pilha de retorno sempre tem o destino inicial do app na parte inferior da pilha.





# BOTÕES VOLTAR

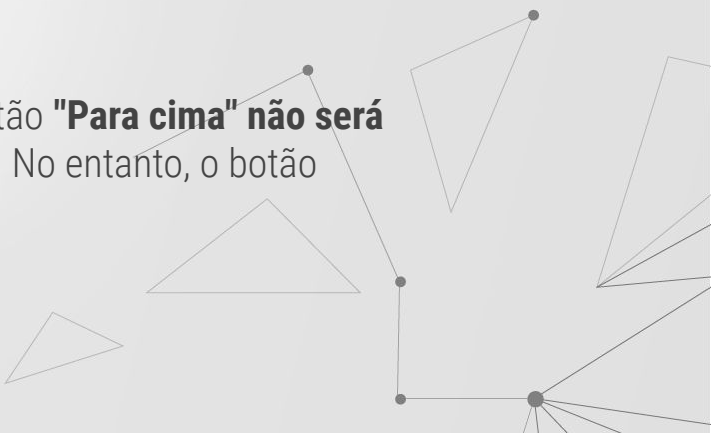
---

O botão "Voltar" é exibido na barra de navegação do **sistema** na parte **inferior da tela**. É usado para navegar em ordem **cronológica inversa** pelo histórico de telas com as quais o usuário trabalhou recentemente.

O botão "**Para cima**" é exibido na **barra de apps** na parte superior da tela. Na tarefa do app, os botões "**Para cima**" e "**Voltar**" **se comportam de maneira idêntica**.

Se um usuário estiver no destino inicial do app, o botão "**Para cima**" **não será exibido**, porque ele **nunca é usado para sair do app**. No entanto, o botão "Voltar" é exibido e causa a saída do app.

---



# NAVIGATION COMPONENT

---







# CONCEITOS-CHAVE

---

## NAVGRAPH

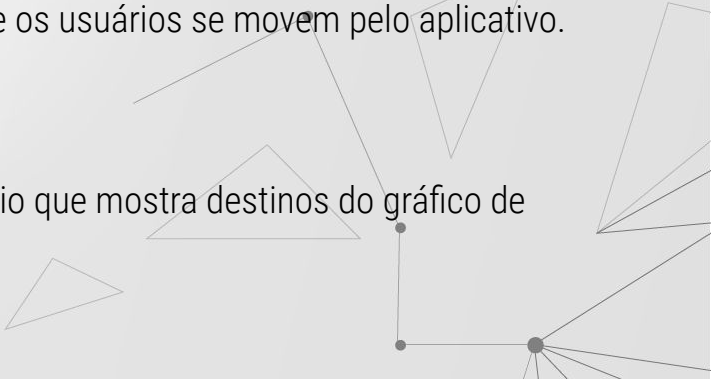
é um recurso XML que contém todas as informações relacionadas à navegação. Todas as áreas de conteúdo individual no aplicativo, chamadas destinos, e todos os caminhos que podem ser percorridos

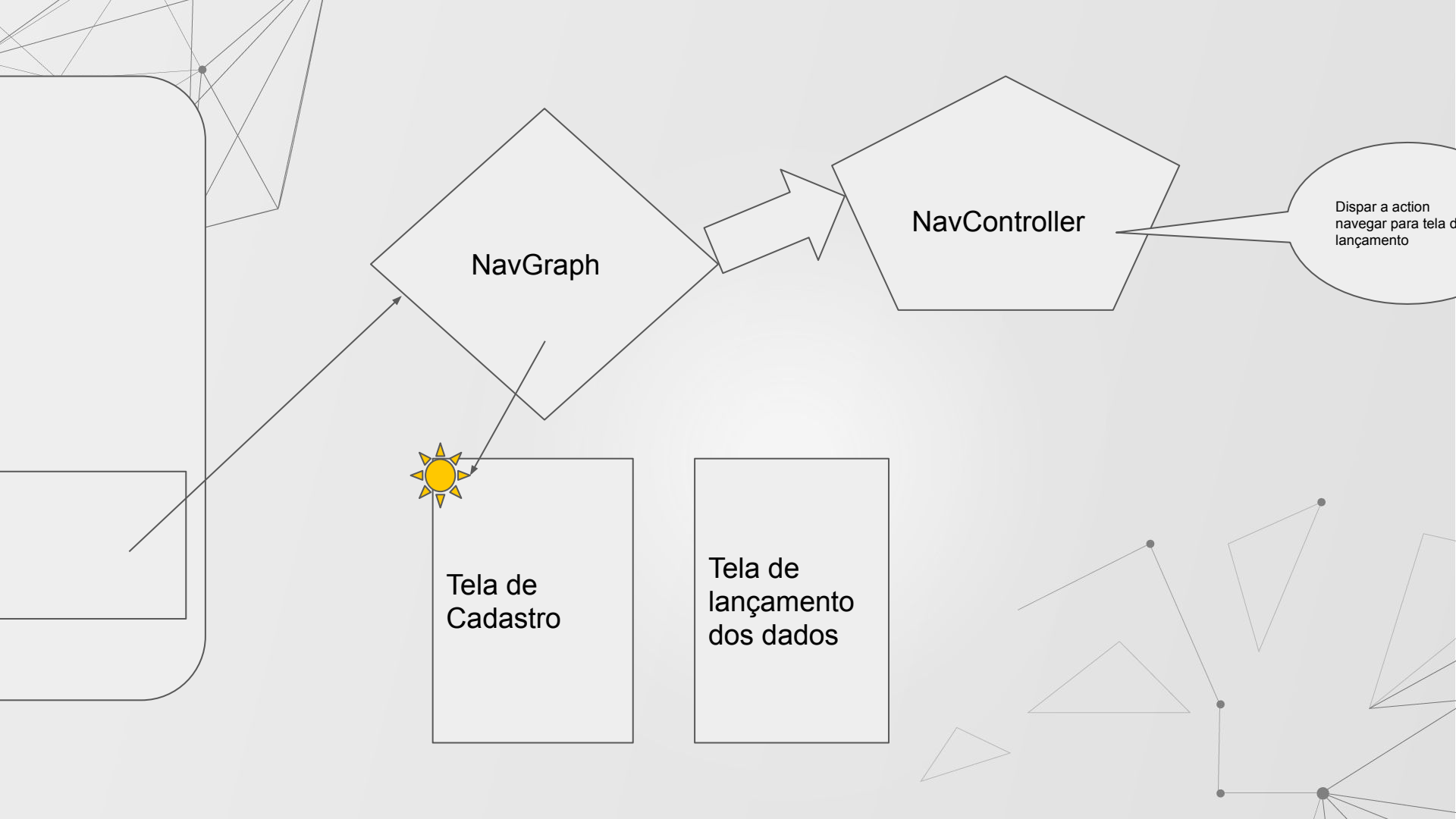
## NAVCONTROLLER

é um objeto que gerencia a navegação em um NavHost. O NavController organiza a troca do conteúdo de destino no NavHost conforme os usuários se movem pelo aplicativo.

## NAVHOST

é um contêiner vazio que mostra destinos do gráfico de navegação.







# *dojo* | LANCE DADOS

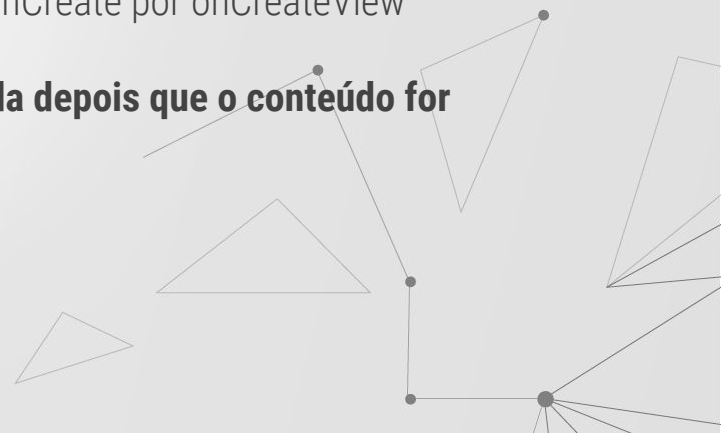
---

Vamos refatorar nossa app Lance Dados para usar o **Navigation Component**

1. Crie um novo Fragment chamado **CadastroFragment**
2. Transfira o layout activity\_cadastro para o fragment\_cadastro
3. Transfira o código da Classe CadastroActivity para a CadastroFragment
  - a. Atenção na substituição do método onCreate por onCreateView

**Obs: a classe CadastroActivity pode ser removida depois que o conteúdo for transferido**

---





# *dojo* | LANCE DADOS FRAGMENT

---

1. Crie um novo Fragment chamado **LanceDadosFragment**
2. Transfira o layout activity\_main para o fragment\_lance\_dados
3. Transfira o código da Classe MainActivity para a LanceDadosFragment
  - a. Atenção na substituição do método onCreate por onCreateView
  - b. Por hora, comente a linha :

```
|  
//      throwDiceLabel.text = "Será hoje seu dia de sorte, ${.getStringExtra("playerName")}?"  
|      }  
|
```

**Obs: remova todo código transferido da MainActivity**

---



# *dojo* | MANIFEST

---

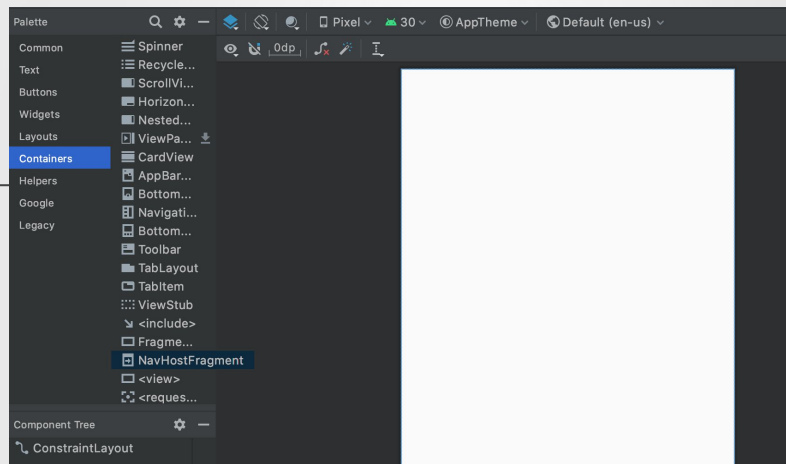
1. Atualize o Manifest.xml e defina a MainActivity como a Activity inicial (action MAIN, category LAUNCHER)
- 



# dojo | MAIN ACTIVITY

A MainActivity será agora responsável por "segurar" os fragments

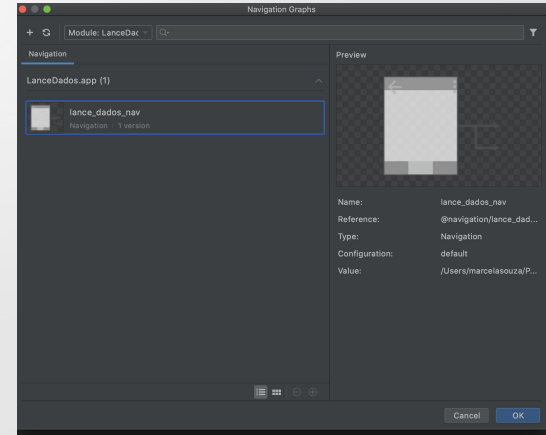
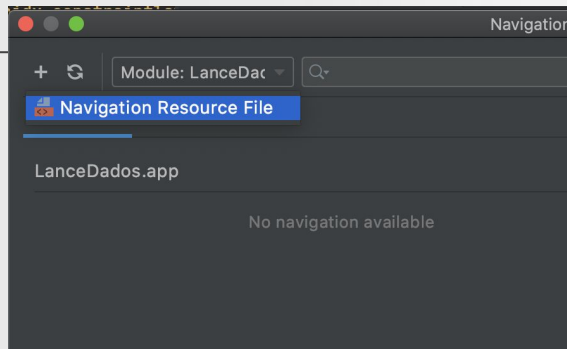
1. Adicione um NavHost no layout activity\_main.xml



# dojo | NAVHOST

A MainActivity será agora responsável por "segurar" os fragments

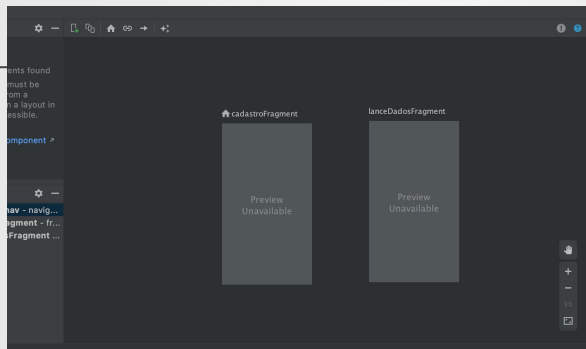
1. Crie um NavGraph lance\_dados\_nav que descreverá a navegação entre os fragments da app
2. Associe o NavGraph ao NavHost



# dojo | NAVGRAPH

O NAVGRAH está vazio. Vamos preenchê-lo com os destinos existentes na nossa app e as ações que conectam os destinos. **Em outras palavras: vamos definir o grafo de navegação da app.**

1. Adicione os fragments Cadastro e LanceDados como destinos (repare que o CadastroFragment tem um ícone de home indicando que ele é o fragment inicial)







## *dojo* / DESTINO

---

Clique em um destino para selecioná-lo e observe os seguintes atributos no painel Attributes:

**Type** indica se o destino é implementado como um fragmento, uma activity ou outra classe personalizada no código-fonte.

**Label** contém o nome do arquivo de layout XML do destino.

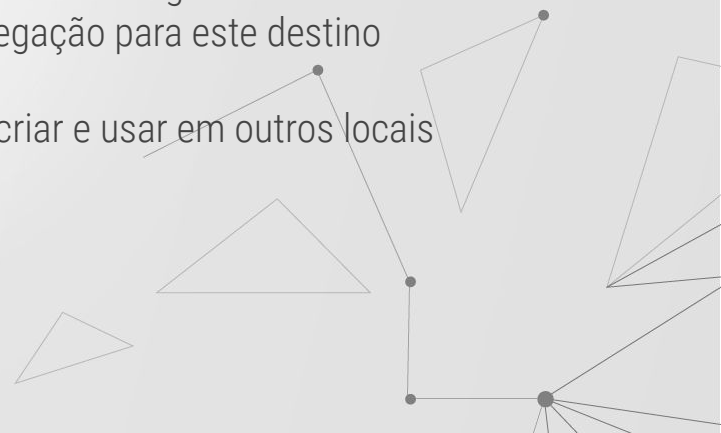
**ID** contém o ID do destino que é usado para referenciar o destino no código.

**Arguments** são informações que podem ser recebidas na navegação para este destino

**Actions** são ações relacionadas a este destino

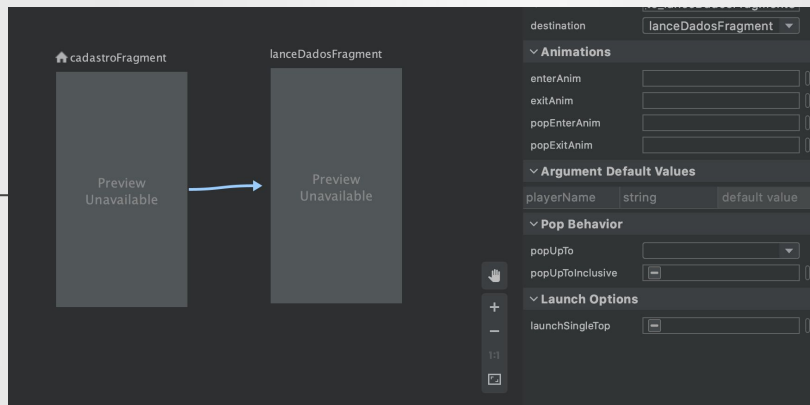
**Deep Links** são links diretos para este destino que você pode criar e usar em outros locais

---



# dojo | ACTIONS

1. Adicione uma ação que conecta o CadastroFragment ao LanceDadosFragment
2. Adicione um *argument* para o destino LanceDadosFragment chamado *playerName*



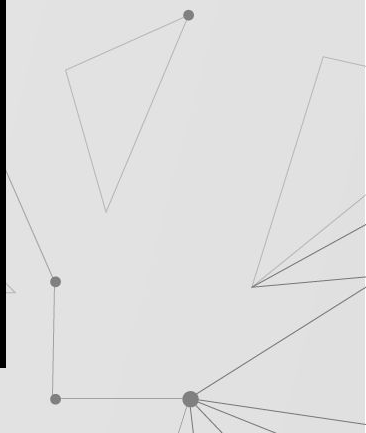


# dojo | ACTIONS

---

Neste ponto, nosso app compila, mas a navegação ainda não funciona perfeitamente. Nos falta refatorar a ação do botão "Regular" (e "Especial" caso você tenha concluído o desafio)

```
val playerName = inputPlayerName.text.toString()
view
    .findNavController()
    .navigate(
        R.id.action_cadastroFragment_to_lanceDadosFragment3,
        bundleOf("playerName" to playerName)
    )
```





# *dojo* | NAVARGS

---

Para finalizar, refatore a forma como pegamos a informação playerName da tela de cadastro para a tela de lançar dados:

```
throwDiceLabel.text = "Será hoje seu dia de sorte, ${arguments?.getString("playerName")}?"
```

---



03

TEMAS E ESTILOS

---





# TEMAS & ESTILOS

---

Os estilos e temas no Android permitem separar as informações de design do app daquelas relacionadas à estrutura e do comportamento da UI, semelhante às folhas de estilos em web design.

---





# PRATIQUE

---

Estilise sua app alterando:

- Fundo da tela
  - Cor da barra superior
  - Cor dos botões
- 





# DESAFIO

---

**Em duplas implemente o botão "Especial"**

---





Commite suas alterações :)

---



# Palavras chave da aula de hoje:

- Fragment
- Ciclo de vida
- Android Jetpack
- Navigation Component
- Intents
- NavGraph
- NavHost
- NavController
- Bundle

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**.

**Please keep this slide for attribution.**