Desenvolvimento para Dispositivos Móveis

Aula 06

Navegação entre telas, Ciclo de vida, Activities, Fragments e Controllers

Prof. Msc. Francisco de Assis de Souza Rodrigues

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Objetivos

- Compreender o ciclo de vida das Activities e Fragments
- Implementar navegação entre telas no Android
- Diferenciar Activities e Fragments
- Usar Controllers (ViewModel e Navigation Component) para melhorar a organização do código

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Visão Geral

- Introdução à Navegação no Android
- Ciclo de Vida no Android
- Activities e Fragments
- Navegação entre Telas



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

A navegação em aplicativos Android refere-se ao **fluxo de interação do usuário ao mover-se entre diferentes telas** (**Activities**) ou componentes (**Fragments**) dentro de um app.

Um bom **sistema de navegação** melhora a experiência do usuário e a organização do código.

https://developer.android.com/guide/navigation?hl=pt-br

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

📌 Conceito de Navegação

Navegação é o processo de transição entre diferentes partes de um aplicativo, como:

- **Troca de telas (Activities e Fragments)**
- **Abertura de novas janelas ou diálogos**
- **Movimentação entre seções de um app, como abas ou menus**

O Android oferece diferentes formas de navegação para garantir que o usuário consiga acessar funcionalidades com fluidez e eficiência.

Introdução à Navegação no Android

📌 Tipos de Navegação

Navegação Explícita

A navegação explícita ocorre quando uma tela específica do app é chamada diretamente, utilizando um Intent para abrir outra Activity.

📌 **Exemplo:** Abrindo uma nova tela (Activity) dentro do app:

```
val intent = Intent(this, SegundaActivity::class.java)
startActivity(intent)
```

💡 **Uso comum:** Quando o app precisa navegar para uma tela interna, como "Configurações" ou "Perfil".

Introdução à Navegação no Android

Navegação Implícita

A navegação implícita ocorre quando o app solicita uma ação sem especificar uma tela específica, deixando o Android decidir qual aplicativo pode processar a requisição.

📌 **Exemplo:** Abrindo um link no navegador:

```
val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com"))
startActivity(intent)
```

💡 **Uso comum:**

- Abrir um link no navegador
- Enviar um e-mail
- Fazer uma ligação telefônica

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

Bottom Navigation (Navegação Inferior)

O **Bottom Navigation** é um menu fixo na parte inferior do app que permite alternar entre diferentes seções do app sem perder o estado da navegação.

📌 **Exemplo de Bottom Navigation com 3 itens:**

- 🏠 **Home**
- 🔍 **Pesquisa**
- ⚙️ **Configurações**

💡 **Uso comum:** Aplicativos com navegação em abas, como **Instagram, YouTube e Twitter**.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

Bottom Navigation (Navegação Inferior)

Exemplo de implementação no activity_main.xml:

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottom_navigation"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom"
    app:menu="@menu/navigation_menu"/>
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

Bottom Navigation (Navegação Inferior)

Exemplo de implementação no MainActivity.kt

```
bottom_navigation.setOnNavigationItemSelectedListener { item ->
    when (item.itemId) {
        R.id.menu_home -> {
            // Ação para a tela Home
            true
        }
        R.id.menu_search -> {
            // Ação para a tela de Pesquisa
            true
        }
        R.id.menu_settings -> {
            // Ação para a tela de Configurações
            true
        }
        else -> false
    }
}
```

Introdução à Navegação no Android

Drawer Navigation (Menu Lateral)

O **Drawer Navigation** é um menu lateral deslizante que contém opções de navegação para diferentes telas do aplicativo.

📌 **Exemplo de apps que usam Drawer Navigation:**

- **Gmail** (Menu lateral para acessar caixas de entrada)
- **Google Drive** (Alternância entre contas e pastas)

💡 **Uso comum:** Quando o app possui muitas seções e precisa de um menu compacto que o usuário possa abrir conforme necessário.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

Drawer Navigation (Menu Lateral)

Exemplo de drawer_layout.xml:

```
<androidx.drawerlayout.widget.DrawerLayout
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navigation_view"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        app:menu="@menu/navigation_menu"/>

</androidx.drawerlayout.widget.DrawerLayout>
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Introdução à Navegação no Android

Drawer Navigation (Menu Lateral)

Exemplo de MainActivity.kt

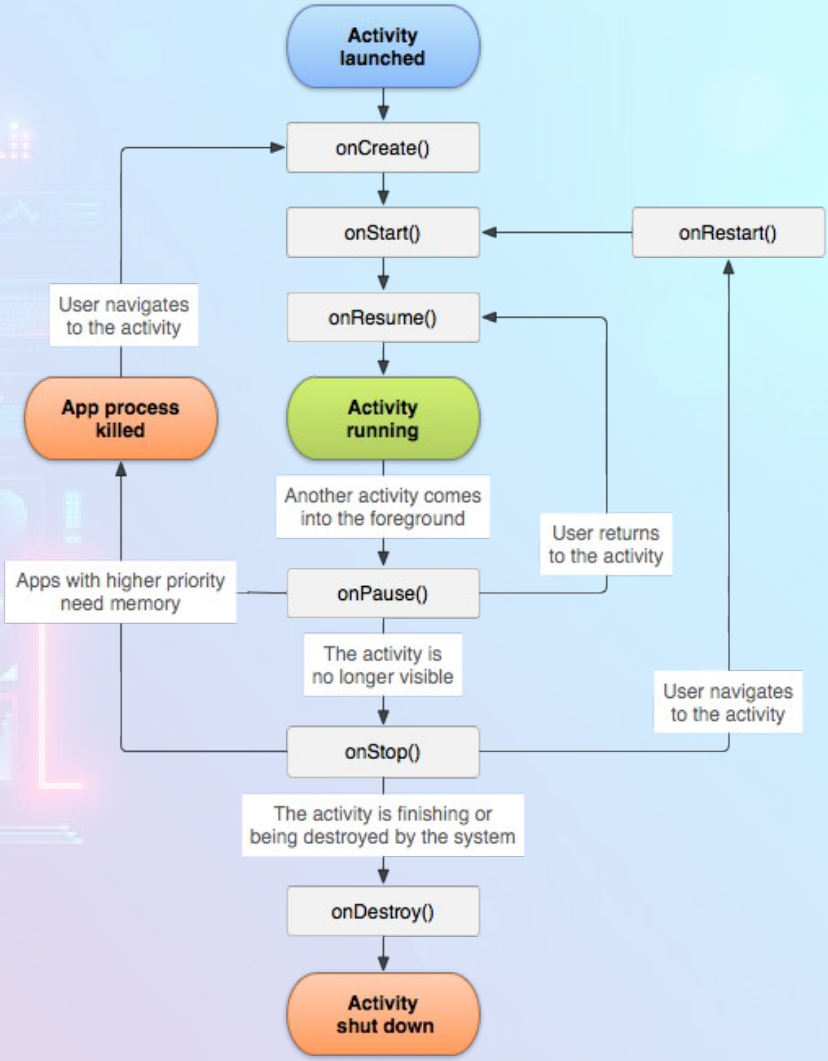
```kotlin
val drawerLayout: DrawerLayout = findViewById(R.id.drawer_layout)
val navigationView: NavigationView = findViewById(R.id.navigation_view)

navigationView.setNavigationItemSelectedListener { item ->
    when (item.itemId) {
        R.id.menu_profile -> {
            // Ação para abrir perfil
            true
        }
        R.id.menu_settings -> {
            // Ação para abrir configurações
            true
        }
        else -> false
    }
}
```

Ciclo de Vida no Android

O ciclo de vida no Android refere-se ao conjunto de estados pelos quais uma Activity ou um Fragment passa durante sua existência. Compreender esse ciclo é essencial para gerenciar corretamente os recursos e o comportamento do aplicativo, evitando consumo desnecessário de memória e garantindo uma experiência fluida ao usuário.

https://developer.android.com/guide/components/activities/activity-lifecycle?hl=pt-br



UniFAP

Ciclo de Vida no Android

📌 Métodos do Ciclo de Vida da Activity

A Activity no Android passa por diferentes estágios do seu ciclo de vida, definidos pelos seguintes métodos:

🔵 Estados e Métodos do Ciclo de Vida da Activity

Estado	Método	Descrição
Criado	`onCreate()`	A Activity está sendo criada. Inicialize componentes.
Iniciado	`onStart()`	A Activity se torna visível para o usuário.
Em Execução	`onResume()`	A Activity está em primeiro plano e interativa.
Pausado	`onPause()`	A Activity ainda está visível, mas não interativa (ex.: um pop-up na frente).
Parado	`onStop()`	A Activity não está mais visível para o usuário.
Destruído	`onDestroy()`	A Activity está sendo finalizada e removida da memória.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Métodos do Ciclo de Vida da Activity

📌 **Exemplo de implementação dos métodos no MainActivity.kt:**
💡 **Dica:** Você pode abrir o **Logcat** no Android Studio para ver os logs aparecendo à medida que a Activity muda de estado.

```kotlin
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        Log.d("CicloDeVida", "onCreate chamado")
    }

    override fun onStart() {
        super.onStart()
        Log.d("CicloDeVida", "onStart chamado")
    }

    override fun onResume() {
        super.onResume()
        Log.d("CicloDeVida", "onResume chamado")
    }

    override fun onPause() {
        super.onPause()
        Log.d("CicloDeVida", "onPause chamado")
    }

    override fun onStop() {
        super.onStop()
        Log.d("CicloDeVida", "onStop chamado")
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d("CicloDeVida", "onDestroy chamado")
    }
}
```

Ciclo de Vida no Android

📌 Comparação entre o Ciclo de Vida de Activity e Fragment

Os **Fragments** também possuem um ciclo de vida próprio, que é **fortemente dependente da Activity em que estão contidos**.

Principais diferenças:

Activity	Fragment
Vive independentemente.	Sempre precisa de uma Activity hospedeira.
Gerenciado pelo sistema Android.	Gerenciado pela Activity que o contém.
Possui um ciclo de vida completo.	Possui um ciclo de vida que depende da Activity.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

Ciclo de Vida de um Fragment

Os principais métodos do ciclo de vida de um Fragment são:

Método	Descrição
`onAttach()`	O Fragment está sendo anexado à Activity.
`onCreate()`	O Fragment está sendo criado (inicialização de variáveis).
`onCreateView()`	Criar e retornar a interface do Fragment.
`onViewCreated()`	Interface criada, é seguro acessar views.
`onStart()`	O Fragment se torna visível.
`onResume()`	O Fragment está interativo.
`onPause()`	O Fragment perde o foco.
`onStop()`	O Fragment não está mais visível.
`onDestroyView()`	A interface do Fragment será destruída.
`onDestroy()`	O Fragment será destruído.
`onDetach()`	O Fragment está sendo removido da Activity.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Exemplo de ciclo de vida de um Fragment em MyFragment.kt:

```kotlin
class MyFragment : Fragment(R.layout.fragment_my) {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d("CicloDeVidaFragment", "onCreate chamado")
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        Log.d("CicloDeVidaFragment", "onCreateView chamado")
        return inflater.inflate(R.layout.fragment_my, container, false)
    }
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Exemplo de ciclo de vida de um Fragment em MyFragment.kt:

```kotlin
override fun onStart() {
    super.onStart()
    Log.d("CicloDeVidaFragment", "onStart chamado")
}

override fun onResume() {
    super.onResume()
    Log.d("CicloDeVidaFragment", "onResume chamado")
}

override fun onPause() {
    super.onPause()
    Log.d("CicloDeVidaFragment", "onPause chamado")
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Exemplo de ciclo de vida de um Fragment em MyFragment.kt:

```kotlin
override fun onPause() {
    super.onPause()
    Log.d("CicloDeVidaFragment", "onPause chamado")
}

override fun onStop() {
    super.onStop()
    Log.d("CicloDeVidaFragment", "onStop chamado")
}

override fun onDestroyView() {
    super.onDestroyView()
    Log.d("CicloDeVidaFragment", "onDestroyView chamado")
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Exemplo de ciclo de vida de um Fragment em MyFragment.kt:

```
    override fun onDestroy() {
        super.onDestroy()
        Log.d("CicloDeVidaFragment", "onDestroy chamado")
    }

    override fun onDetach() {
        super.onDetach()
        Log.d("CicloDeVidaFragment", "onDetach chamado")
    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Uso do ViewModel para Manter Dados ao Longo do Ciclo de Vida

Quando a Activity ou Fragment é recriado (por exemplo, ao girar a tela), os dados armazenados em variáveis normais são **perdidos**. O **ViewModel** permite armazenar dados de forma segura ao longo do ciclo de vida, mesmo quando a Activity/Fragment é recriada.

📌 Passos para usar o ViewModel:

1. Criar uma classe ViewModel
2. Criar uma instância do ViewModel na Activity ou Fragment
3. Observar os dados e atualizá-los conforme necessário

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 Exemplo de ViewModel para contar cliques do botão:

```
class MainViewModel : ViewModel() {
    var contador = MutableLiveData(0)

    fun incrementar() {
        contador.value = (contador.value ?: 0) + 1
    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Ciclo de Vida no Android

📌 **Uso na MainActivity.kt**

✅ **Vantagem do ViewModel:** Se o usuário girar a tela, o valor do contador **não será perdido**!

```
class MainActivity : AppCompatActivity() {
    private lateinit var viewModel: MainViewModel

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

        viewModel = ViewModelProvider(this).get(MainViewModel::class.java)

        val button = findViewById<Button>(R.id.button)
        val textView = findViewById<TextView>(R.id.textView)

        viewModel.contador.observe(this) { contador ->
            textView.text = "Contador: $contador"
        }

        button.setOnClickListener {
            viewModel.incrementar()
        }
    }
}
```

Activities e Fragments

No desenvolvimento Android, Activities e Fragments são dois componentes fundamentais para estruturar a interface e a navegação do aplicativo. Ambos são usados para exibir conteúdos e interagir com o usuário, mas têm propósitos e comportamentos diferentes.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Diferenças entre Activities e Fragments

Característica	Activity	Fragment
Independência	Funciona de forma independente.	Precisa ser hospedado dentro de uma `Activity`.
Gerenciamento pelo SO	Controlada diretamente pelo sistema Android.	Controlado pela `Activity` que o contém.
Ciclo de vida	Tem seu próprio ciclo de vida completo.	O ciclo de vida depende da `Activity` hospedeira.
Navegação	Pode abrir outra `Activity` com `Intent`.	Troca de `Fragments` geralmente é gerenciada por um `FragmentManager`.
Reutilização	Não pode ser reutilizada dentro de outra `Activity`.	Pode ser reutilizado em diferentes `Activities`.

📌 **Resumo:**

- Use **Activity** quando for um fluxo independente do app.
- Use **Fragment** quando precisar de partes reutilizáveis da interface dentro de uma mesma Activity.

Activities e Fragments

📌 Quando Usar Fragments em vez de Activities?

✔️ Usar Fragments quando:

✅ O app precisa exibir múltiplas telas dentro da mesma Activity (ex.: abas).
✅ Quer reaproveitar a mesma interface em diferentes contextos.
✅ Precisa dividir uma tela em várias partes independentes (ex.: **layouts responsivos** para tablet e celular).
✅ Deseja facilitar a navegação com o **Navigation Component** do Jetpack.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 **Exemplo: Aplicativo de Notas**

- 🏠 **Tela principal** → Lista de notas (Fragment A).
- 📄 **Detalhes da nota** → Exibe o conteúdo (Fragment B).
- 📱 Em celulares: a Activity muda para exibir um Fragment.
- 🖥️ Em tablets: a Activity pode exibir ambos os Fragments ao mesmo tempo.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

❌ **Usar Activities quando:**

🚫 A tela tem um escopo independente do resto do app.
🚫 É um fluxo principal do aplicativo (ex.: Tela de Login, Tela Inicial).
🚫 O desempenho pode ser impactado ao trocar de tela (Fragments são mais leves).

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

Já que os **Fragments não são independentes**, muitas vezes precisamos fazer com que eles se comuniquem com a Activity ou entre si.

1. **Comunicação de Fragment → Activity**

O Fragment pode chamar um método diretamente da Activity.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

📌 **Exemplo:** Enviar uma mensagem do Fragment para a Activity.

No Fragment (MyFragment.kt)

```
class MyFragment : Fragment(R.layout.fragment_my) {

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val button = view.findViewById<Button>(R.id.button_send)
        button.setOnClickListener {
            (activity as? MainActivity)?.receberMensagem("Olá, Activity!")
        }
    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

Na Activity MainActivity.kt

```
class MainActivity : AppCompatActivity() {

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

    fun receberMensagem(mensagem: String) {
        Toast.makeText(this, mensagem, Toast.LENGTH_SHORT).show()
    }
}
```

✅ **Explicação:** Quando o botão do Fragment é pressionado, ele chama um método da Activity para exibir um Toast.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

2. Comunicação de Activity → Fragment

A Activity pode chamar diretamente um método do Fragment.

📌 **Exemplo:** Activity passando dados para um Fragment.

Na Activity (MainActivity.kt)

```
class MainActivity : AppCompatActivity() {

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

        val fragment = supportFragmentManager.findFragmentById(R.id.fragment_container) as? MyFragment
        fragment?.atualizarTexto("Novo texto da Activity!")
    }
}
```

Activities e Fragments

📌 Comunicação entre Fragments e Activities

No Fragment (MyFragment.kt)

```
class MyFragment : Fragment(R.layout.fragment_my) {

    fun atualizarTexto(novoTexto: String) {
        view?.findViewById<TextView>(R.id.textView)?.text = novoTexto
    }
}
```

✅ **Explicação:** A Activity obtém uma referência do Fragment e chama um método para atualizar o texto.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

Se dois Fragments precisam trocar informações, a abordagem recomendada é usar a Activity como intermediária.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

📌 **Exemplo:** Fragment A envia dados para Fragment B via Activity.

No Fragment A (FragmentA.kt)

```
class FragmentA : Fragment(R.layout.fragment_a) {

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val button = view.findViewById<Button>(R.id.button_send)
        button.setOnClickListener {
            (activity as? MainActivity)?.enviarParaFragmentB("Olá, Fragment B!")
        }
    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Activities e Fragments

📌 Comunicação entre Fragments e Activities

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Na Activity (MainActivity.kt)

```
class MainActivity : AppCompatActivity() {

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

    fun enviarParaFragmentB(mensagem: String) {
        val fragmentB = supportFragmentManager.findFragmentById(R.id.fragment_b) as? FragmentB
        fragmentB?.receberMensagem(mensagem)
    }
}
```

Activities e Fragments

📌 Comunicação entre Fragments e Activities

No Fragment B (FragmentB.kt)

```
class FragmentB : Fragment(R.layout.fragment_b) {

    fun receberMensagem(mensagem: String) {
        view?.findViewById<TextView>(R.id.textView)?.text = mensagem
    }
}
```

✅ Explicação:

- FragmentA chama um método da Activity.
- A Activity localiza FragmentB e passa os dados para ele.
- FragmentB atualiza sua interface com a mensagem recebida.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

A navegação no Android refere-se à transição entre diferentes telas do aplicativo, seja entre Activities ou Fragments. Há diversas formas de implementar essa navegação, dependendo da arquitetura do app e da experiência do usuário.



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Navegação entre Activities com Intent

Uma **Activity** representa uma única tela do app. Para navegar de uma Activity para outra, usamos **Intents**.

Criando uma Navegação Entre Activities

Exemplo: Vamos abrir a SecondActivity a partir da MainActivity ao clicar em um botão.

1. Criar a segunda Activity (SecondActivity.kt):

```
class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)
    }
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Navegação entre Activities com Intent

2. Criar um botão na interface da MainActivity (activity_main.xml):

```
<Button
    android:id="@+id/button_navigate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ir para Segunda Tela" />
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Navegação entre Activities com Intent

3. Criar a navegação na MainActivity.kt usando Intent:

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

        val button = findViewById<Button>(R.id.button_navigate)
        button.setOnClickListener {
            val intent = Intent(this, SecondActivity::class.java)
            startActivity(intent)
        }
    }
}
```

Explicação:

Criamos um Intent, informando que queremos abrir a SecondActivity

Chamamos startActivity(intent), que inicia a nova tela.

Navegação entre Telas

📌 Navegação entre Fragments com Navigation Component

O Navigation Component do Jetpack facilita a navegação entre Fragments, mantendo um histórico correto e permitindo a passagem de dados facilmente.

Configurando a navegação com Navigation Component

1. Adicionar a dependência no build.gradle (Module: app):

```
dependencies {
    implementation 'androidx.navigation:navigation-fragment-ktx:2.7.5'
    implementation 'androidx.navigation:navigation-ui-ktx:2.7.5'
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Navegação entre Fragments com Navigation Component

2. Criar um NavHostFragment no layout (activity_main.xml):

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/nav_host_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:navGraph="@navigation/nav_graph" />
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Navegação entre Fragments com Navigation Component

```xml
<?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"
    xmlns:tools="http://schemas.android.com/tools"
    app:startDestination="@id/fragmentA">

    <fragment
        android:id="@+id/fragmentA"
        android:name="com.example.app.FragmentA"
        android:label="Fragment A" >
        <action
            android:id="@+id/action_fragmentA_to_fragmentB"
            app:destination="@id/fragmentB" />
    </fragment>

    <fragment
        android:id="@+id/fragmentB"
        android:name="com.example.app.FragmentB"
        android:label="Fragment B" />
</navigation>
```

3. Criar um gráfico de navegação (res/navigation/nav_graph.xml):

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Navegação entre Fragments com Navigation Component

4. **Navegar entre Fragments usando NavController**:
No FragmentA.kt:

```
class FragmentA : Fragment(R.layout.fragment_a) {
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val button = view.findViewById<Button>(R.id.button_navigate)
        button.setOnClickListener {
            findNavController().navigate(R.id.action_fragmentA_to_fragmentB)
        }
    }
}
```

Explicação:

- Criamos um gráfico de navegação (nav_graph.xml).
- Usamos findNavController().navigate(R.id.action_fragmentA_to_fragmentB) pa
  mudar para o próximo Fragment.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Passagem de Dados entre Telas

1. Passagem de Dados Entre Activities

Podemos enviar dados de uma Activity para outra usando **Intent Extras**.

Na MainActivity.kt (Enviando dados)

```
val intent = Intent(this, SecondActivity::class.java)
intent.putExtra("NOME", "João")
startActivity(intent)
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Passagem de Dados entre Telas

Na SecondActivity.kt (Recebendo dados)

```
val nome = intent.getStringExtra("NOME")
Toast.makeText(this, "Nome recebido: $nome", Toast.LENGTH_SHORT).show()
```

Explicação:

- Usamos putExtra() para enviar dados.
- Na outra Activity, usamos getStringExtra() para recuperar os dados.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Passagem de Dados entre Telas

2. Passagem de Dados Entre Fragments com Safe Args

O **Safe Args** é a forma recomendada para passar dados entre Fragments com o Navigation Component.

1. **Adicionar a dependência no build.gradle:**

```
dependencies {
    classpath "androidx.navigation:navigation-safe-args-gradle-plugin:2.7.5"
}
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Passagem de Dados entre Telas

2. **Criar um argumento no nav_graph.xml:**

```
<fragment
    android:id="@+id/fragmentB"
    android:name="com.example.app.FragmentB">
    <argument
        android:name="mensagem"
        app:argType="string" />
</fragment>
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Passagem de Dados entre Telas

3. **Enviar os dados no FragmentA.kt:**

```
val action = FragmentADirections.actionFragmentAToFragmentB("Olá, Fragment B!")
findNavController().navigate(action)
```

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO

Navegação entre Telas

📌 Passagem de Dados entre Telas

4. **Receber os dados no FragmentB.kt:**

```
val args: FragmentBArgs by navArgs()
val mensagem = args.mensagem
Toast.makeText(requireContext(), mensagem, Toast.LENGTH_SHORT).show()
```

Explicação:

- O Safe Args cria classes automáticas para passar os dados de forma segura.
- FragmentADirections.actionFragmentAToFragmentB("mensagem") cria a ação com argumentos.
- No FragmentB, usamos navArgs() para receber os dados.

Atividade

1. Criar um app com duas telas:
   a. **Tela A:** Botão para navegar para a **Tela B**
   b. **Tela B:** Recebe um dado da **Tela A** e exibe na interface
2. Implementar a navegação usando:
   a. Intent para troca de **Activities**
   b. Navigation Component para troca de **Fragments**
3. Simular mudanças no ciclo de vida usando Log.d() para entender os métodos do ciclo de vida

Bibliografia

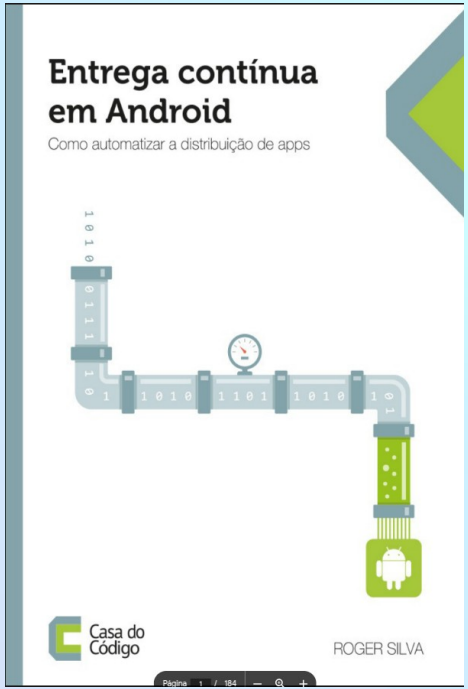
RESENDE, Kassiano. Kotlin com android: crie aplicativos de maneira fácil e divertida. São Paulo, SP: Casa do Código, 2018. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 26 fev. 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/213016

SILVA, Roger. Entrega contínua em android: como automatizar a distribuição de apps. São Paulo, SP: Casa do Código, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 26 fev. 2025.

https://plataforma.bvirtual.com.br/Acervo/Publicacao/212670

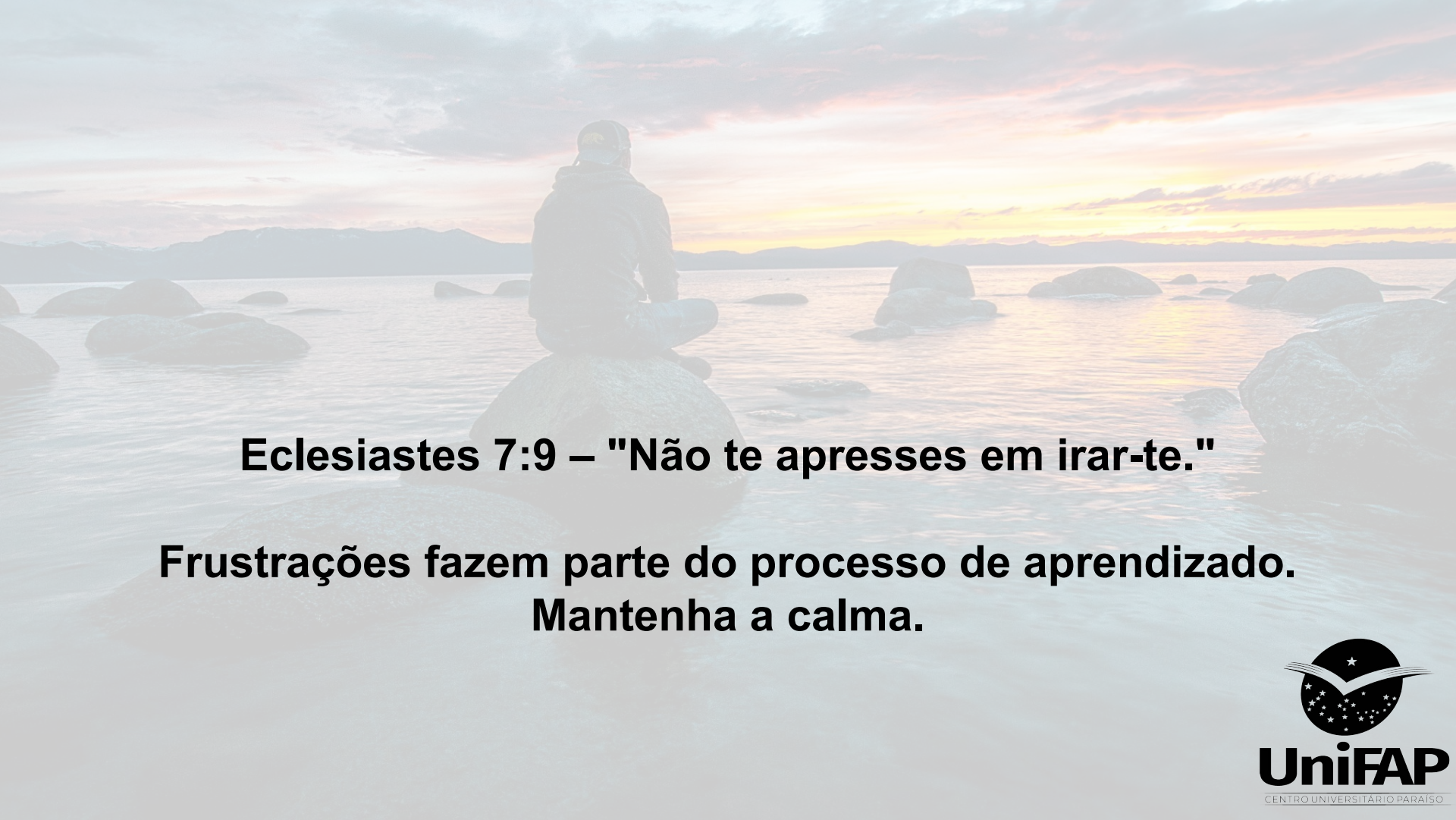




Perguntas?!



UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



Eclesiastes 7:9 – "Não te apresses em irar-te."

Frustrações fazem parte do processo de aprendizado.
Mantenha a calma.

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO



Obrigado!

UniFAP
CENTRO UNIVERSITÁRIO PARAÍSO