Desenvolvimento de Aplicativo Mobile

Material base desenvolvido pelo
Prof. Raphael Barreto

raphael.b.oliveira@docente.senai.br

2026

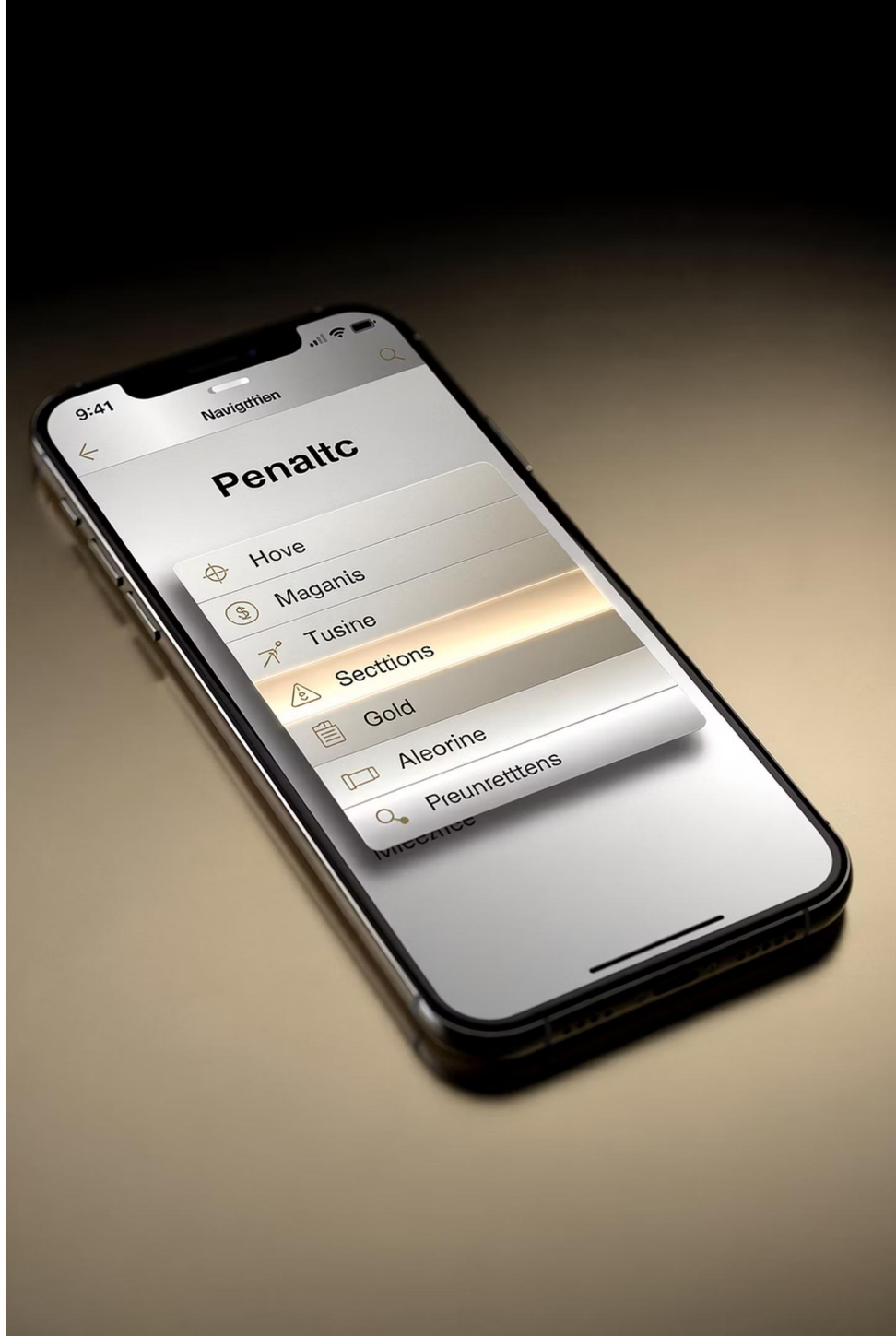
DEC - GEP

Navegação em Drawer (Menu Hambúrguer)

Uma introdução completa à implementação de navegação lateral em aplicativos React Native

FIRJAN SENAI

2026



Sumário da Apresentação

01

O que é Navegação em Drawer?

Entendendo o padrão de design e suas aplicações práticas

Conceito e Funcionalidade

A navegação em Drawer, também conhecida como menu hambúrguer, é um padrão de interface que apresenta um menu lateral deslizante acionado por um ícone de três linhas horizontais. Este componente é fundamental quando seu aplicativo possui múltiplas telas e você precisa oferecer acesso fácil e organizado a todas elas.

O termo "drawer" (gaveta, em inglês) descreve perfeitamente o comportamento: o menu desliza lateralmente como uma gaveta sendo aberta, revelando opções de navegação que ficam ocultas quando não estão em uso. Isso economiza espaço valioso na interface, mantendo a tela limpa e focada no conteúdo principal.

Quando Utilizar a Navegação em Drawer

Cenários Ideais

O Drawer é especialmente útil quando você tem mais de cinco telas principais no aplicativo, ultrapassando o limite recomendado para navegação por tabs

Vantagens Principais

- **Organização de múltiplas opções de navegação sem sobrecarregar a interface**
- **Acesso rápido a funcionalidades secundárias e configurações**
- **Flexibilidade para adicionar ou remover itens conforme necessário**
- **Familiaridade para usuários, presente em aplicativos populares como Mercado Livre**

02

Configuração Inicial

Preparando o ambiente e instalando dependências necessárias

Instalação das Dependências

Para implementar a navegação em Drawer no seu projeto React Native com Expo Router, é necessário instalar bibliotecas específicas que fornecem os componentes e funcionalidades necessárias. A documentação oficial do Expo lista todas as dependências requeridas para este tipo de navegação.

Importante: Parar a Aplicação

Antes de instalar novas dependências, sempre pare a execução da aplicação no terminal. Instalar pacotes enquanto o app está rodando pode causar problemas de reconhecimento das novas bibliotecas. Após a instalação, reinicie o servidor de desenvolvimento.

```
npm install @react-navigation/drawer react-native-gesture-handler react-native-reanimated
```

Estrutura Inicial do Projeto

Antes de implementar o Drawer, é importante entender a estrutura de navegação já existente no projeto. No nosso caso, temos uma navegação por tabs funcionando, com telas de Home, Configurações e Produtos, além de uma tela de Usuários que está "solta" e será integrada ao menu Drawer.

app/tabs/_layout.jsx

Gerencia a navegação por tabs inferior com três abas principais

app/tabs/products/_layout.jsx

Implementa Stack Navigator dentro da aba de produtos

app/_layout.jsx

Layout raiz onde implementaremos o Drawer Navigator

app/user.jsx

Tela que será acessível através do menu Drawer

03

Implementação do Drawer

Transformando o layout raiz e integrando o menu lateral

Substituindo Stack por Drawer

O primeiro passo para implementar a navegação em Drawer é modificar o arquivo de layout principal (app/_layout.jsx), substituindo o componente Stack pelo componente Drawer. Esta mudança transforma fundamentalmente como a navegação de nível superior funciona no aplicativo.

Importação do Componente

```
import Drawer from "expo-router/drawer"
```

O componente Drawer vem diretamente do expo-router, mantendo a integração com o sistema de roteamento baseado em arquivos.

Substituição no Layout Stack por Drawer

```
<Drawer>
        <Drawer.Screen name="tabs" options={{ headerShown: false }} />
        <Drawer.Screen name="user" options={{ headerTitle: "Usuário" }} />
        <Drawer.Screen name="+not-found" options={{ headerTitle: "Erro" }} />
</Drawer>
```

Cada tela ou grupo de telas agora é registrado como Drawer.Screen.

Comportamento Inicial do Drawer

Após implementar o Drawer, se ocê retirar o headerShown de tabs, você notará que um botão de menu hambúrguer aparece automaticamente na barra de navegação. Por padrão, o Drawer exibe todas as telas registradas no menu lateral, incluindo tanto a navegação por tabs quanto a tela de usuário.
No entanto, este comportamento padrão geralmente não é o ideal. O menu mostra todas as rotas disponíveis, incluindo elementos técnicos como "(tabs)" e "not-found", que não deveriam ser visíveis para o usuário final. Além disso, a apresentação é bastante básica, sem ícones ou formatação personalizada.

Para criar uma experiência de usuário profissional e intuitiva, precisaremos personalizar completamente o conteúdo do menu Drawer, controlando exatamente quais itens aparecem, como são apresentados e qual é seu comportamento ao serem selecionados.

04

Personalização do Menu

Criando um menu customizado com ícones e comportamento definido

Criando o Componente CustomDrawerContent no layout principal

Para personalizar completamente o conteúdo do menu Drawer, criamos um componente customizado que define exatamente quais itens aparecem e como se comportam. Este componente substitui o menu padrão, dando controle total sobre a aparência e funcionalidade.

```
import { StatusBar } from "expo-status-bar"
import Drawer from "expo-router/drawer"
import { DrawerContentScrollView, DrawerItem } from "@react-navigation/drawer"
import { FontAwesome } from "@expo/vector-icons"
import { router } from "expo-router"

const CustomDrawerContent = (props) => {
  return (
    <DrawerContentScrollView>
      <DrawerItem
        icon={({ color }) => (<FontAwesome name="user" size={20} color={color} />)}
        label="Usuário"
        onPress={() => router.push("/user")}
      />
    </DrawerContentScrollView>
  )
}
```

O DrawerContentScrollView permite que o menu role verticalmente caso tenha muitos itens, enquanto cada DrawerItem representa uma opção de navegação individual com ícone, label e ação de clique.

Configurando o Drawer para Usar Conteúdo Customizado

Após criar o componente CustomDrawerContent, precisamos instruir o Drawer a utilizá-lo em vez do menu padrão. Isso é feito através da propriedade drawerContent no componente Drawer principal.

Configuração no Layout

```
<Drawer
  drawerContent={() =>
    <CustomDrawerContent />
  }
  screenOptions={{
    headerStyle: {
      backgroundColor: "#004a80"
    },
    headerTintColor: "#FFFFFF"
  }}
></Drawer>
```

Elementos Importantes

- A prop drawerContent recebe uma função que retorna o componente customizado
- ScreenOptions define estilos globais para todas as telas do Drawer
- Cores seguem o padrão visual da Firjan SENAI
- O menu agora exibe apenas os itens explicitamente definidos

Adicionando o Botão Drawer no Layout de Tabs

Por padrão, o botão do menu Drawer aparece apenas na navbar mais externa. Para melhorar a experiência do usuário, devemos adicionar o botão em cada tela onde queremos que o menu esteja acessível. Isso é especialmente importante quando trabalhamos com múltiplos níveis de navegação.

Posicionamento no Header

Adicione o DrawerToggleButton na propriedade headerRight (ou headerLeft) dentro do screenOptions do layout de tabs. Este botão funciona automaticamente, abrindo e fechando o menu quando pressionado.

Personalização Visual

Use a propriedade tintColor para definir a cor do ícone, garantindo contraste adequado com o fundo da navbar. No padrão Firjan SENAI, utilizamos branco (#FFFFFF) sobre o azul institucional.

```
headerLeft: () => (
  <DrawerToggleButton tintColor="#FFFFFF" />
)
```

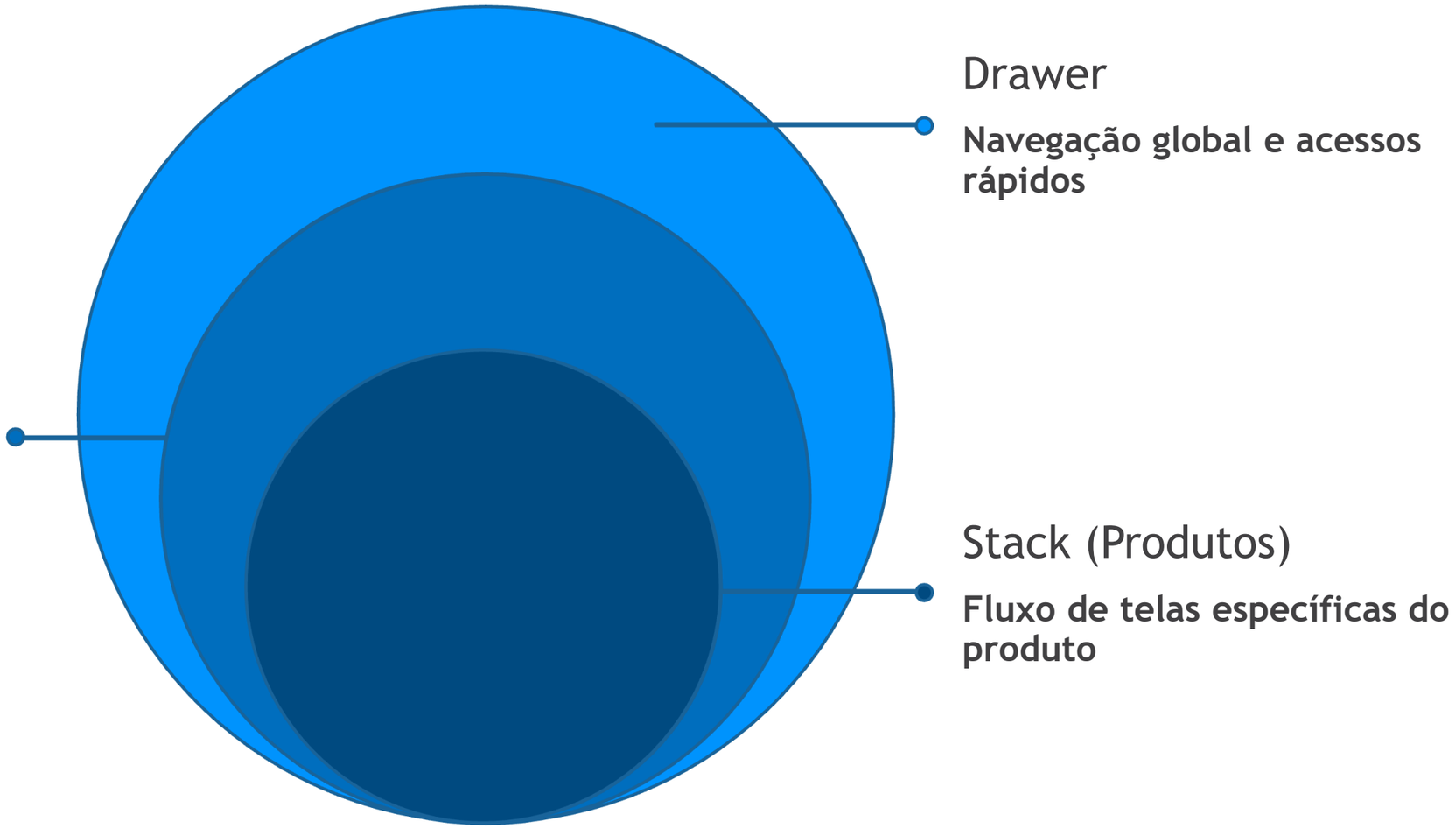
05

Integração Completa

Combinando Stack, Tabs e Drawer em um sistema de navegação coeso

Gerenciando Múltiplos Níveis de Navegação

Quando trabalhamos com aplicativos que combinam diferentes tipos de navegação (Stack, Tabs e Drawer), é fundamental entender a hierarquia e onde cada navbar aparece. No nosso projeto, temos três níveis de navegação interconectados.



Ocultando Headers Redundantes

Com múltiplos níveis de navegação, frequentemente acabamos com headers duplicados na tela. Para manter uma interface limpa e profissional, devemos ocultar os headers redundantes usando a opção headerShown: false.

Estratégia de Ocultação

- Oculte o header do Drawer principal, pois o header das Tabs será visível
- Mantenha o header das Tabs visível com o botão do Drawer
- Para o Stack de Produtos, oculte o header das Tabs e mostre o header do Stack
- Adicione o DrawerToggleButton em cada header que permanece visível

Esta abordagem garante que sempre há exatamente um header na tela, evitando confusão visual e desperdício de espaço.

Configuração no Layout do Drawer

```
<Drawer.Screen
  name="(tabs)"
  options={{
    headerShown: false
  }}
/>
```

Configuração no Products Layout

```
headerLeft: () => (<DrawerToggleButton
tintColor="#FFFFFF" />)
```

06

Código Funcional Completo

Implementação final integrada de todos os componentes

Revisão Final e Próximos Passos

Implementamos com sucesso a navegação em Drawer, criando um menu lateral personalizado com ícones e comportamento customizado. A integração com os outros tipos de navegação (Stack e Tabs) demonstra como combinar diferentes padrões para criar experiências de usuário sofisticadas e intuitivas.

O que Aprendemos

- Instalação e configuração de dependências do Drawer
- Criação de menu lateral personalizado com DrawerContentScrollView
- Integração de ícones e navegação programática
- Gerenciamento de múltiplos níveis de navegação
- Boas práticas para headers e botões de menu

Próximo Módulo

Nas próximas aulas, exploraremos funcionalidades específicas de dispositivos móveis que não existem na web. Aprenderemos a trabalhar com recursos nativos como gestos touch avançados, sensores do dispositivo, câmera, geolocalização e outras capacidades exclusivas do ambiente mobile.

> "A navegação em Drawer é essencial para aplicativos complexos com muitas telas. Dominar sua implementação e integração com outros padrões de navegação é fundamental para criar experiências mobile profissionais."

Firjan SENAI