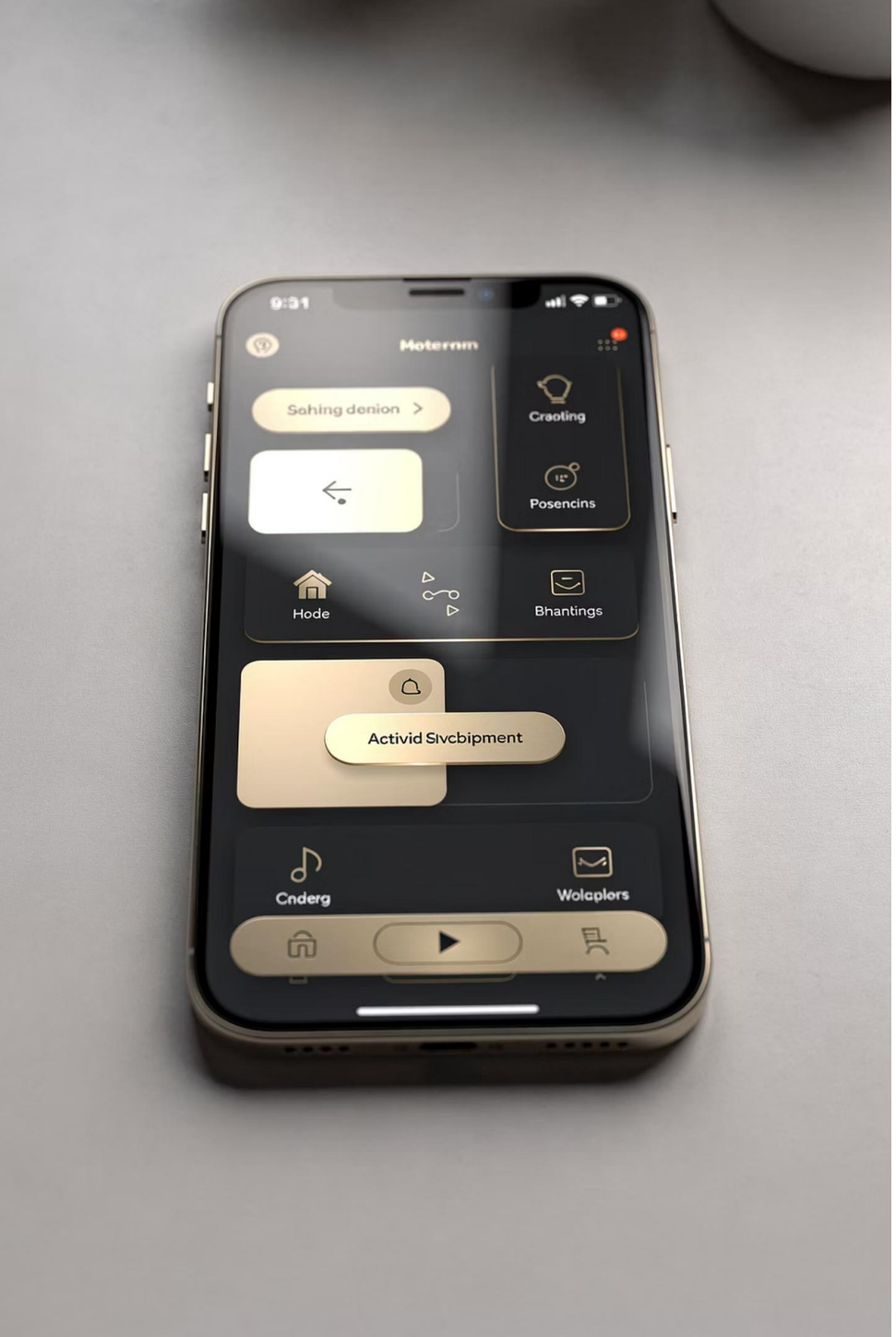
Desenvolvimento de Aplicativo Mobile

Material base desenvolvido pelo
Prof. Raphael Barreto

raphael.b.oliveira@docente.senai.br

2026

DEC - GEP



Navegação em Tabs (Abas) no React Native

SENAI 2026

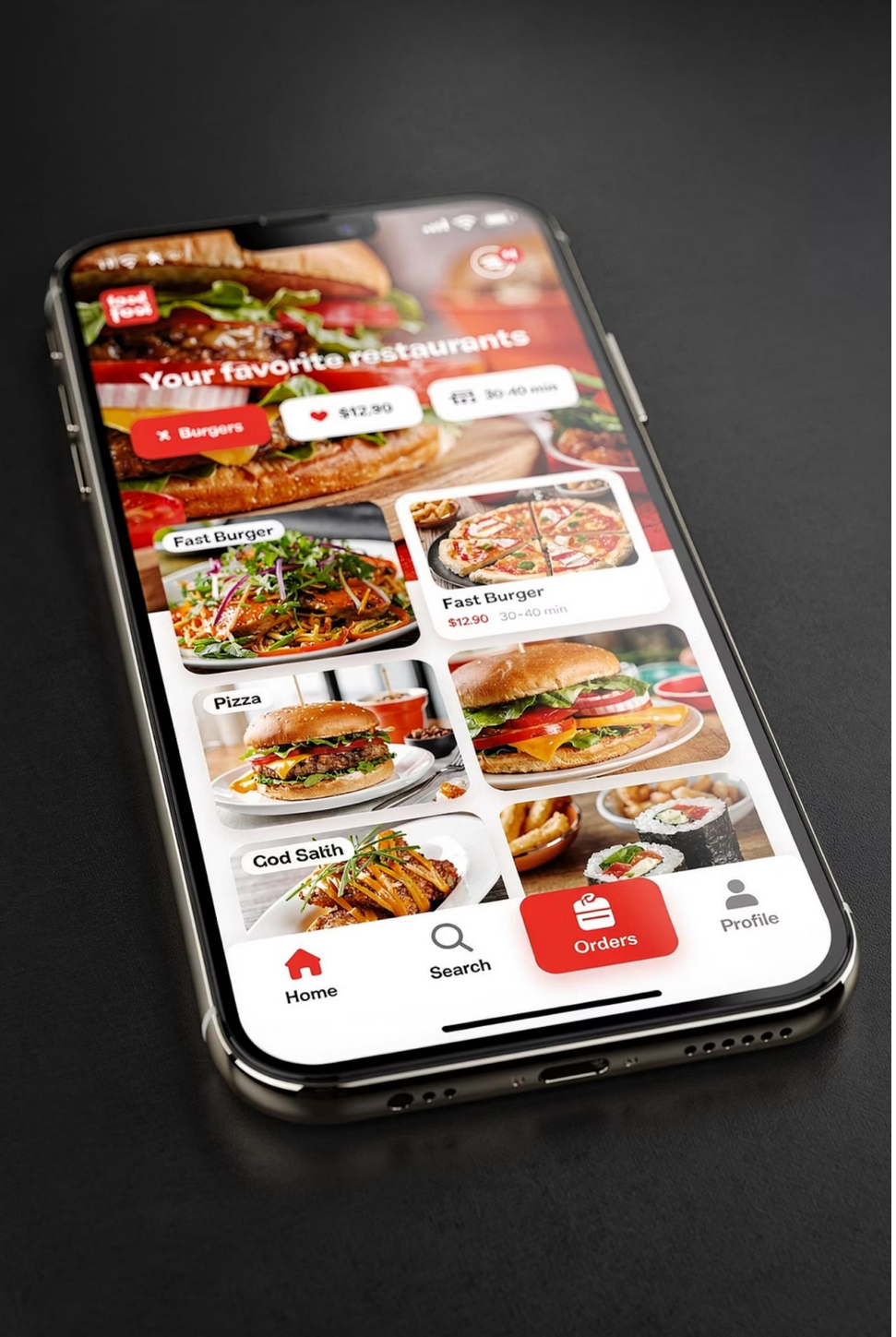
DESENVOLVIMENTO MOBILE

Sumário

01

Introdução à Navegação em Tabs

Compreendendo o conceito e aplicações práticas no desenvolvimento mobile



O Que São Tabs?

As tabs, ou abas, são uma forma de navegação muito comum em aplicativos móveis modernos. Elas aparecem como uma barra fixa, geralmente na parte inferior da tela, contendo botões que representam as principais seções do aplicativo.

No exemplo do iFood, temos cinco tabs principais: Início, Busca, Loja, Pedidos e Perfil. Essa barra permanece sempre visível, permitindo que o usuário navegue rapidamente entre as áreas mais importantes do aplicativo com apenas um toque.

Quando Usar Navegação em Tabs?

Categorias Principais

Quando você tem 3 a 5 seções principais que precisam de acesso rápido

Navegação Frequente

Para áreas que o usuário precisa acessar constantemente durante o uso

Organização Clara

Quando as categorias são distintas e bem definidas

Vantagens das Tabs

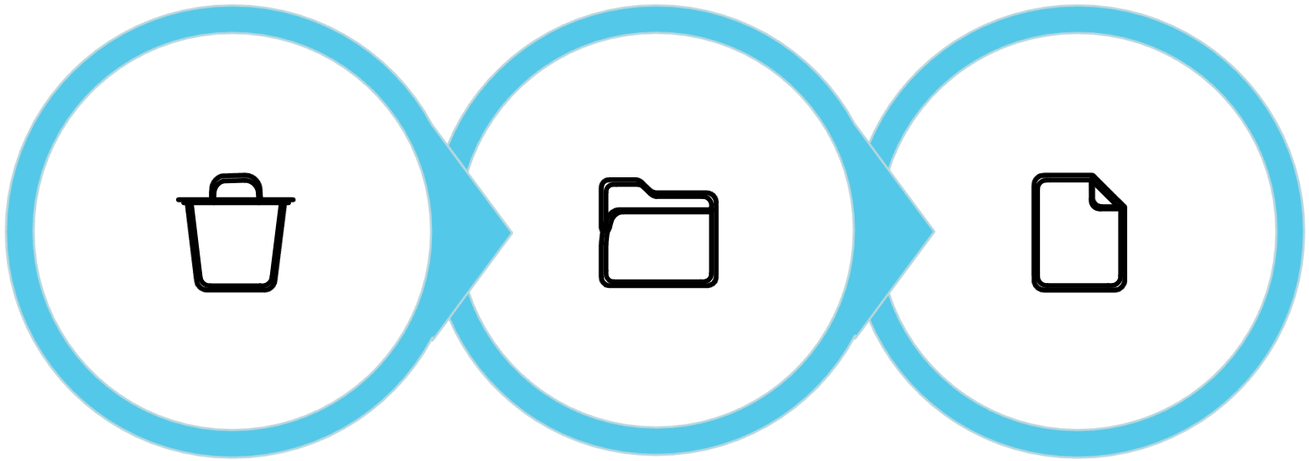
- Acesso imediato às principais funcionalidades
- Redução de cliques para navegação
- Orientação visual clara para o usuário
- Padrão familiar em apps mobile

02

Estrutura Base do Projeto

Organizando arquivos e preparando a configuração inicial

Preparando o Ambiente



Limpar Navegação

Criar Pastas

Mover Arquivos

Estrutura de Pastas Inicial

Antes da Implementação

```
app/
├── _layout.jsx
├── index.jsx
└── +not-found.jsx
└── components/
|   └── settings.jsx
|   └── user.jsx
└── products/
|   └── [id].jsx
|   └── index.jsx
```

Nossa estrutura inicial com navegação simples por stacks

Depois da Reorganização

```
app/
├── tabs/
|   └── products/
|       └── _layout.jsx
|       └── [id].jsx
|       └── index.jsx
|   ├── _index.jsx
|   ├── _layout.jsx
|   ├── settings.jsx
├── +not-found.jsx
├── _layout.jsx
├── index.jsx
└── user.jsx
```

Estrutura organizada para suportar navegação em tabs

Criando a Pasta (tabs)

O primeiro passo é criar uma pasta chamada `(tabs)` dentro do diretório `app`. O uso de parênteses no nome da pasta é uma convenção do Expo Router que indica um grupo de rotas com layout compartilhado.

Dentro dessa pasta, criaremos um arquivo `_layout.jsx` que será responsável por configurar toda a estrutura de navegação em tabs. Este arquivo é fundamental pois define como as abas serão organizadas e apresentadas ao usuário.

03

Implementação das Tabs

Criando a navegação em abas passo a passo

Primeiro Código: Layout das Tabs

Vamos criar o arquivo `app/(tabs)/_layout.jsx` com a configuração básica:

```
import { Tabs } from "expo-router"

export default function TabsLayout() {
  return (
    <Tabs>
      <Tabs.Screen name="index"/>
      <Tabs.Screen name="settings"/>
    </Tabs>
  )
}
```

Este código importa o componente `Tabs` do Expo Router e define duas telas: `index` (nossa Home) e `settings` (Configurações).

Movendo os Arquivos para (tabs)

Próximo Passo Importante

Agora precisamos mover os arquivos `index.jsx` e `settings.jsx` para dentro da pasta `(tabs)`. Isso é essencial para que o Expo Router reconheça essas páginas como parte da navegação em tabs.

Atenção: Ao mover os arquivos, você precisará ajustar os imports dos styles, adicionando

```
../../styles/styles
```

no caminho para subir um nível na hierarquia de pastas.

Atualizando o Layout Principal

No arquivo `app/_layout.jsx`, precisamos informar que temos tabs na aplicação:

```jsx
import { Stack } from "expo-router"
import { StatusBar } from "expo-status-bar"

export default function RootLayout() {
  return (
    <>
      <Stack>
        <Stack.Screen name="tabs" options={{ headerShown: false }} />
        <Stack.Screen name="+not-found" options={{ headerTitle: "Erro" }} />
      </Stack>
      <StatusBar style="auto" />
    </>
  )
}
```

O `headerShown: false` é crucial para evitar a duplicação de headers na tela.

Redirecionando para a index Principal

Precisamos redirecionar nossa index para tabs:

```
import { Redirect } from "expo-router"

export default function Index() {
  return <Redirect href="/tabs" />
}
```

Testando a Implementação Básica

Neste momento, ao executar o aplicativo, você já verá a navegação em tabs funcionando! Aparecerá uma barra inferior com dois botões, permitindo alternar entre as telas Home e Settings.

É normal que a aparência ainda não esteja ideal — os ícones podem aparecer quebrados e as cores podem não corresponder ao tema do app. Vamos resolver isso nas próximas etapas de personalização.

04

Personalização Visual

Ajustando cores, ícones e estilos da barra de tabs

Configurando Cores do Header

Vamos adicionar `screenOptions` ao nosso tabs/_Layout para personalizar a aparência:

```
<Tabs screenOptions={{
    headerStyle: { backgroundColor: "#E94560“
    headerTintColor: "#FFFFFF",
    tabBarActiveTintColor: "#E94560",
    tabBarInactiveTintColor: "gray"
   }}
   >
```

Propriedades Explicadas

- `headerStyle` - Define o estilo do cabeçalho superior
- `headerTintColor` - Cor do texto no header
- `tabBarActiveTintColor` - Cor da tab ativa
- `tabBarInactiveTintColor` - Cor das tabs inativas

Dica Importante

Use as cores do tema Firjan SENAI: #54c8e8 (azul principal), #0082AD (azul escuro), #343739 (cinza escuro) e #747372 (cinza médio) para manter a identidade visual institucional.

Adicionando Títulos Personalizados

Podemos melhorar os nomes exibidos nas tabs usando a propriedade `options` tabs/_layout.jsx:

```
import { Tabs } from "expo-router"

export default function TabsLayout() {
    return (
        <Tabs screenOptions={{
            headerStyle: { backgroundColor: "#E94560" },
            headerTintColor: "#FFFFFF",
            tabBarActiveTintColor: "#E94560",
            tabBarInactiveTintColor: "gray"
        }}
        >
            <Tabs.Screen name="index" options={{ title: "Home" }} />
            <Tabs.Screen name="settings" options={{ title:" Configurações "}}/>
        </Tabs>
    )
}
```

Agora, ao invés de mostrar "index“ e "settings“, o aplicativo exibirá "Home“ e "Configurações“ tanto no header superior quanto na barra de tabs inferior, proporcionando uma experiência mais profissional e intuitiva.

Implementando Ícones nas Tabs

Código para Ícones

```
import { Tabs } from "expo-router"
import { FontAwesome } from "@expo/vector-icons"

export default function TabsLayout() {
    return (
        <Tabs screenOptions={{
            headerStyle: { backgroundColor: "#E94560" },
            headerTintColor: "#FFFFFF",
            tabBarActiveTintColor: "#E94560",
            tabBarInactiveTintColor: "gray"
        }}
        >
            <Tabs.Screen
            name="index"
            options={{ title: "Home", tabBarIcon: ({ color }) => (<FontAwesome size={28} name="home" color={color} />) }} />
            <Tabs.Screen
            name="settings"
            options={{ title: " Configurações ", tabBarIcon: ({ color }) => (<FontAwesome size={28} name="cog" color={color} />) }} />
        </Tabs>
    )
}
```

A propriedade `tabBarIcon` recebe uma função que retorna o componente do ícone. O parâmetro `color` é fornecido automaticamente e muda conforme a tab está ativa ou inativa, seguindo as cores definidas em `tabBarActiveTintColor` e `tabBarInactiveTintColor`.

Escolhendo Ícones Apropriados

Home

`name="home"` - Ícone de casa para a página inicial

Configurações

`name="cog"` - Engrenagem para configurações

Produtos

`name="shopping-bag"` - Sacola para produtos

Visite icons.expo.fyi para explorar todos os ícones disponíveis do FontAwesome e escolher os mais adequados para seu aplicativo.

05

Combinando Tabs e Stacks

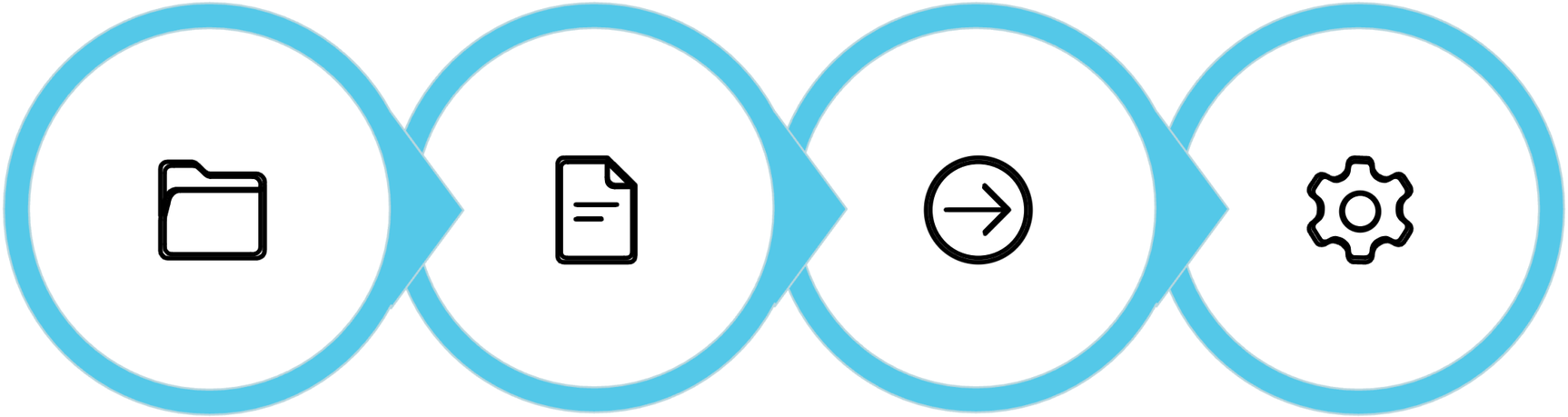
Integrando diferentes tipos de navegação no mesmo aplicativo

Por Que Combinar Tabs e Stacks?

Em aplicativos reais, é comum ter uma navegação em tabs onde cada tab contém sua própria pilha (stack) de navegação. Por exemplo, na tab de Produtos, você pode ter uma lista de produtos e, ao clicar em um, navegar para a tela de detalhes — tudo isso sem sair da tab.

Isso permite que cada seção do app mantenha seu próprio histórico de navegação independente, melhorando significativamente a experiência do usuário.

Criando a Estrutura de Produtos



Colocar a pasta de produtos em tabs

Adicionar _layout

Mover páginas

Configurar stack

Estrutura de Arquivos Completa

```
app/
├── tabs/
|   └── products/
|       └── _layout.jsx
|       └── [id].jsx
|       └── index.jsx
|   ├── _index.jsx
|   ├── _layout.jsx
|   ├── settings.jsx
├── +not-found.jsx
├── _layout.jsx
├── index.jsx
└── user.jsx
```

Arquivos Principais

- `app/_layout.jsx` - Layout raiz
- `tabs/_layout.jsx` - Configuração das tabs
- `products/_layout.jsx` - Stack de produtos

Organização Clara

Cada nível de navegação tem seu próprio arquivo de layout, facilitando manutenção e escalabilidade do projeto.

Configurando o Layout de Produtos

Arquivo `app/tabs/products/_layout.jsx`:

```
import { Stack } from "expo-router"

export default function ProductsLayout() {
    return (
        <Stack
            screenOptions={{
                headerStyle: {
                    backgroundColor: "#E94560"
                },
                headerTintColor: "#FFFFFF"
            }}
        >
            <Stack.Screen name="index" options={{ headerTitle: "Lista de Produtos" }} />
            <Stack.Screen name="[id]" options={{ headerTitle: "Detalhes do Produto" }} />
        </Stack>
    )
}
```

Esta configuração cria uma pilha de navegação dentro da tab de produtos, permitindo navegar entre a lista de produtos e os detalhes de cada produto mantendo o histórico de navegação.

Adicionando a Tab de Produtos

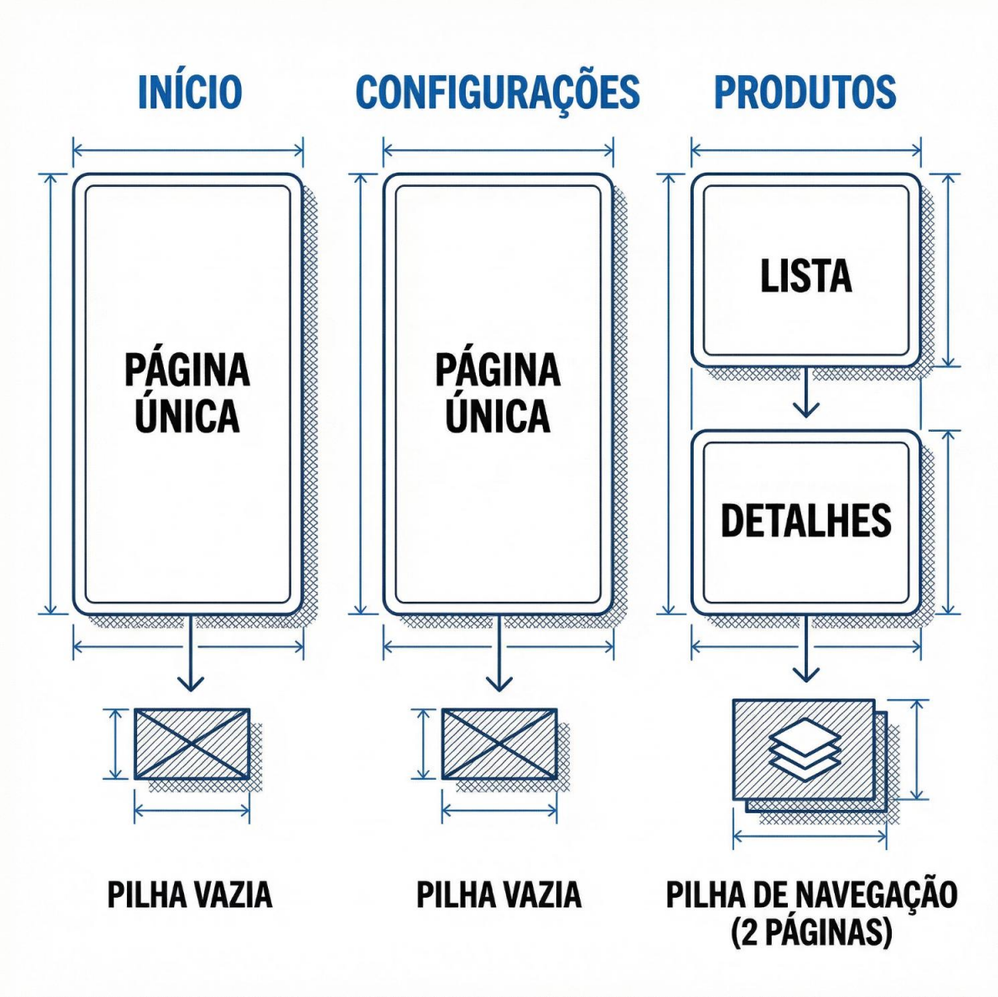
No arquivo `(tabs)/_layout.jsx`, adicione a nova tab:

```
import { Tabs } from "expo-router"
import { FontAwesome } from "@expo/vector-icons"

export default function TabsLayout() {
    return (
        <Tabs screenOptions={{
            headerStyle: { backgroundColor: "#E94560" },
            headerTintColor: "#FFFFFF",
            tabBarActiveTintColor: "#E94560",
            tabBarInactiveTintColor: "gray"
        }}
        >
            <Tabs.Screen
                name="index"
                options={{ title: "Home", tabBarIcon: ({ color }) => (<FontAwesome size={28} name="home" color={color} />) }} />
            <Tabs.Screen
                name="settings"
                options={{ title: "Configurações", tabBarIcon: ({ color }) => (<FontAwesome size={28} name="cog" color={color} />) }} />
            <Tabs.Screen
                name="products"
                options={{ title: "Produtos", tabBarIcon: ({ color }) => (<FontAwesome size={28} name="shopping-bag" color={color} />) }} />
        </Tabs>
    )
}
```

Importante: O `headerShown: false` é essencial aqui para evitar headers duplicados, já que a stack de produtos tem seu próprio header.

Comportamento da Navegação Híbrida



Vantagens da Abordagem

- Cada tab mantém seu estado de navegação independente
- Ao voltar para uma tab, você retorna exatamente onde parou
- Navegação intuitiva e familiar aos usuários
- Flexibilidade para ter tabs simples e tabs complexas

Quando você navega da Home para Produtos, vê a lista; clica em um produto e vê os detalhes. Se mudar para Settings e voltar, ainda estará nos detalhes do produto.

Boas Práticas de Navegação

Limite de Níveis

Evite mais de 3-4 níveis de navegação para não confundir o usuário

Orientação Clara

Sempre indique visualmente onde o usuário está na navegação

Foco nas Principais

Mantenha 3-5 tabs principais, não sobrecarregue a barra

Consistência

Use padrões similares em todas as seções do app

Todos os Arquivos da Implementação

app/_layout.jsx

Layout principal com Stack e configuração das tabs

app/tabs/_layout.jsx

Configuração das tabs com cores, ícones e títulos

app/tabs/index.jsx

Página Home - tela inicial do aplicativo

app/tabs/settings.jsx

Página de Configurações

app/tabs/products/_layout.jsx

Stack de navegação para produtos

app/tabs/products/index.jsx

Lista de produtos com links

app/tabs/products/[id].jsx

Detalhes de um produto específico

Todos esses arquivos trabalham juntos para criar uma navegação fluida e intuitiva, combinando tabs e stacks de forma eficiente.

Conclusão e Próximos Passos

Parabéns! Você aprendeu a implementar navegação em tabs no React Native usando Expo Router. Agora você é capaz de criar aplicativos com navegação profissional, combinando tabs e stacks para oferecer uma experiência de usuário fluida e intuitiva.

1

Aprendemos

Criar tabs, personalizar aparência, combinar com stacks

2

Próximo Tema

Navegação com Drawers - o último tipo de navegação

3

Continue Praticando

Experimente criar suas próprias combinações de navegação

Firjan SENAI