

Desenvolvimento de Aplicativo Mobile

Material base desenvolvido pelo
Prof. Raphael Barreto

raphael.b.oliveira@docente.senai.br

Parâmetros na Rota, Rotas Aninhadas e Tela de Not Found

Desenvolvimento Front-End com React Native e Expo Router

SENAI 2026

NAVEGAÇÃO AVANÇADA



Sumário da Aula

1 Rotas Aninhadas

Organização de caminhos através de pastas e estrutura hierárquica

2 Parâmetros Dinâmicos

Criação de rotas com valores variáveis usando colchetes

3 Hook useLocalSearchParams

Captura e utilização de parâmetros nas rotas

4 Tela de Not Found

Criação de página personalizada para erros 404

5 Prática Completa

Implementação do código funcional completo

 CAPÍTULO 01

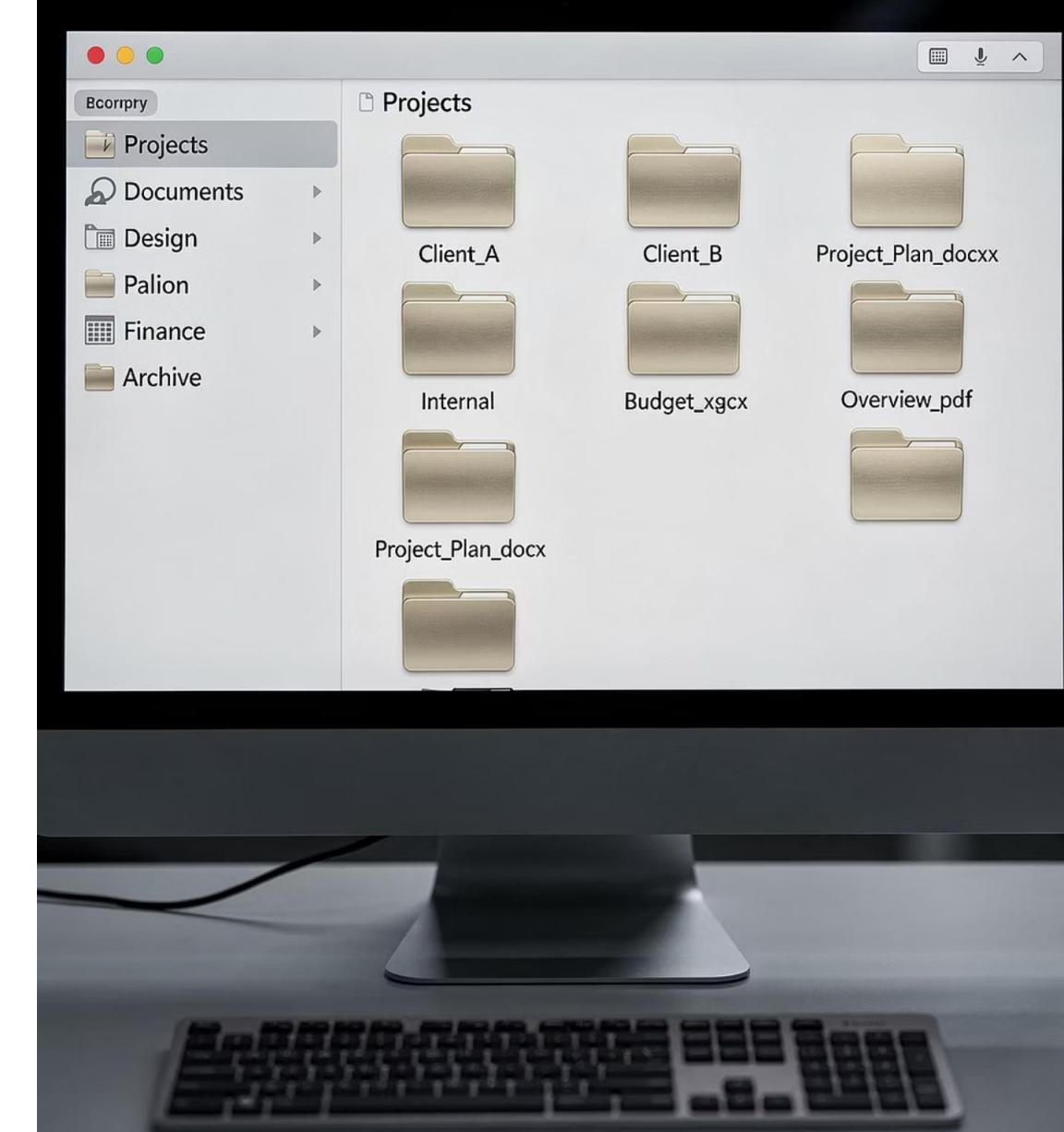
Rotas Aninhadas: Organizando a Navegação

Parâmetros na Rota, Rotas Aninhadas e Tela de Not Found

O Que São Rotas Aninhadas?

Rotas aninhadas permitem organizar seus caminhos de navegação de forma hierárquica, similar à estrutura de pastas no seu computador. Ao invés de ter todas as páginas no mesmo nível, você pode agrupá-las logicamente.

Por exemplo, todas as páginas relacionadas a produtos ficam dentro de `/products`, como `/products/lista` ou `/products/detalhes`. Isso facilita a manutenção e torna a estrutura do projeto mais intuitiva e escalável.



Criando a Estrutura de Pastas

Antes da Organização

Arquivos soltos na pasta app:

- index.jsx
- settings.jsx
- user.jsx
- products.jsx
- product-detail.jsx

Estrutura desorganizada e difícil de escalar conforme o projeto cresce.

Depois da Organização

Estrutura hierárquica com pasta products:

- app/index.jsx
- app/_layout.jsx
- App/components/setting.jsx
- app/components/user.jsx
- app/products/index.jsx (substituiu products por index)
- app/products/detalhes.jsx (substituiu product-detail por detalhes)

Agrupamento lógico facilita navegação e manutenção do código.

O Arquivo Index: Página Principal da Rota

Função do Index

Dentro de cada pasta, o arquivo `index.jsx` representa a página principal daquela rota. Quando o usuário acessa `/products`, é o `index` que será renderizado.

Convenção de Nomenclatura

O nome do componente React pode ser qualquer um, mas o **nome do arquivo** deve ser exatamente `index.jsx`. Isso permite que o Expo Router identifique automaticamente a rota raiz.

```
import { Text, View } from "react-native"
import { styles } from "../../styles/styles"
import { Link } from "expo-router"

export default function Products() {
  return (
    <View style={[styles.container, { backgroundColor: "#FAEDCB" }]}>
      <Text>Lista de Produtos</Text>
      <Link href="/products/detalhes"> Ver Detalhes do Produto <img alt="arrow icon" style={{ height: 15px }} /></Link>
    </View>
  )
}
```

Configurando Rotas Aninhadas no Layout

Após criar a estrutura de pastas, é necessário registrar as rotas no componente `layout.jsx`. A Stack reconhece automaticamente a hierarquia baseada nos nomes dos arquivos e pastas.

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

export default function RootLayout() {
  return (
    <>
      <StatusBar style="light" backgroundColor="#E94560" />
      <Stack screenOptions={{ headerStyle: { backgroundColor: "#E94560" }, headerTintColor: "#FFFFFF" }}>
        <Stack.Screen name="index" options={{ headerTitle: "Home" }} />
        <Stack.Screen name="components/settings" options={{ headerTitle: "Configurações" }}/>
        <Stack.Screen name="components/user" options={{ headerTitle: "Usuário" }}/>
        <Stack.Screen name="products/index" options={{ headerTitle: "Produtos" }} />
        <Stack.Screen name="products/detalhes" options={{ headerTitle: "Detalhes do Produto" }} />
      </Stack>
    </>
  )
}
```

Note que o `name` segue o caminho da pasta seguido pelo nome do arquivo. A barra (/) indica a hierarquia da navegação.

🔗 CAPÍTULO 02

Parâmetros Dinâmicos nas Rotas

Parâmetros na Rota, Rotas Aninhadas e Tela de Not Found



A Necessidade de Parâmetros Dinâmicos

Quando trabalhamos com listas de itens - produtos, usuários, posts - precisamos que a mesma página exiba diferentes conteúdos baseados em um identificador único. Ao invés de criar uma rota separada para cada produto, usamos parâmetros dinâmicos.

Por exemplo, `/products/1`, `/products/2` e `/products/100` podem todas usar o mesmo componente, apenas alterando o ID que é passado como parâmetro na URL.

Na página principal de produtos `app/products/index.jsx` iremos substituir o trecho de Código:

`<Link href="/products/detalhes"> Ver Detalhes do Produto → </Link>`
pelo seguinte código:

```
<Link push href="/products/1">Produto 1 → </Link>
<Link push href="/products/2">Produto 2 → </Link>
```

Sintaxe de Parâmetros com Colchetes

Arquivo Estático

detalhes.jsx

Rota fixa: /products/detalhes

Arquivo Dinâmico

[id].jsx

Rota variável: /products/1,
/products/2, etc.

Como Funciona

Os **colchetes** no nome do arquivo indicam que aquele segmento da rota é um parâmetro variável. O nome dentro dos colchetes (`id` no exemplo) será o nome da variável que você poderá acessar no código.

Você escolhe o nome que fizer mais sentido: `[id]`, `[slug]`, `[productId]`, etc.

Atualizando a Stack com Parâmetros

Quando renomeamos o arquivo para incluir parâmetros dinâmicos, devemos atualizar também a configuração da Stack no `layout.jsx`:

```
<Stack.Screen name="products/[id]" options={{ headerTitle: "Detalhes do Produto" }} /></Stack>
```

O `name` deve refletir exatamente o caminho do arquivo, incluindo os colchetes. Isso informa ao Expo Router que essa rota aceita um parâmetro chamado `id`.

- ❑ **Importante:** Sempre que modificar a estrutura de navegação, feche e abra novamente o aplicativo para garantir que as mudanças sejam reconhecidas corretamente.

Capturando Parâmetros com useLocalSearchParams

```
// app/products/[id].jsx
import { useLocalSearchParams } from "expo-router"

export default function ProductDetail() {
  const { id } = useLocalSearchParams()

  return (
    <View>
      <Text>Produto ID: {id}</Text>
    </View>
  )
}
```

Como Usar o Hook

O hook `useLocalSearchParams()` retorna um objeto contendo todos os parâmetros da rota atual. Usamos desestruturação para extrair o parâmetro específico que precisamos.

O nome da variável (`id`) deve corresponder ao nome usado nos colchetes do arquivo. Com isso, você pode usar o valor dinamicamente em qualquer lugar do componente.

Navegando para Rotas com Parâmetros

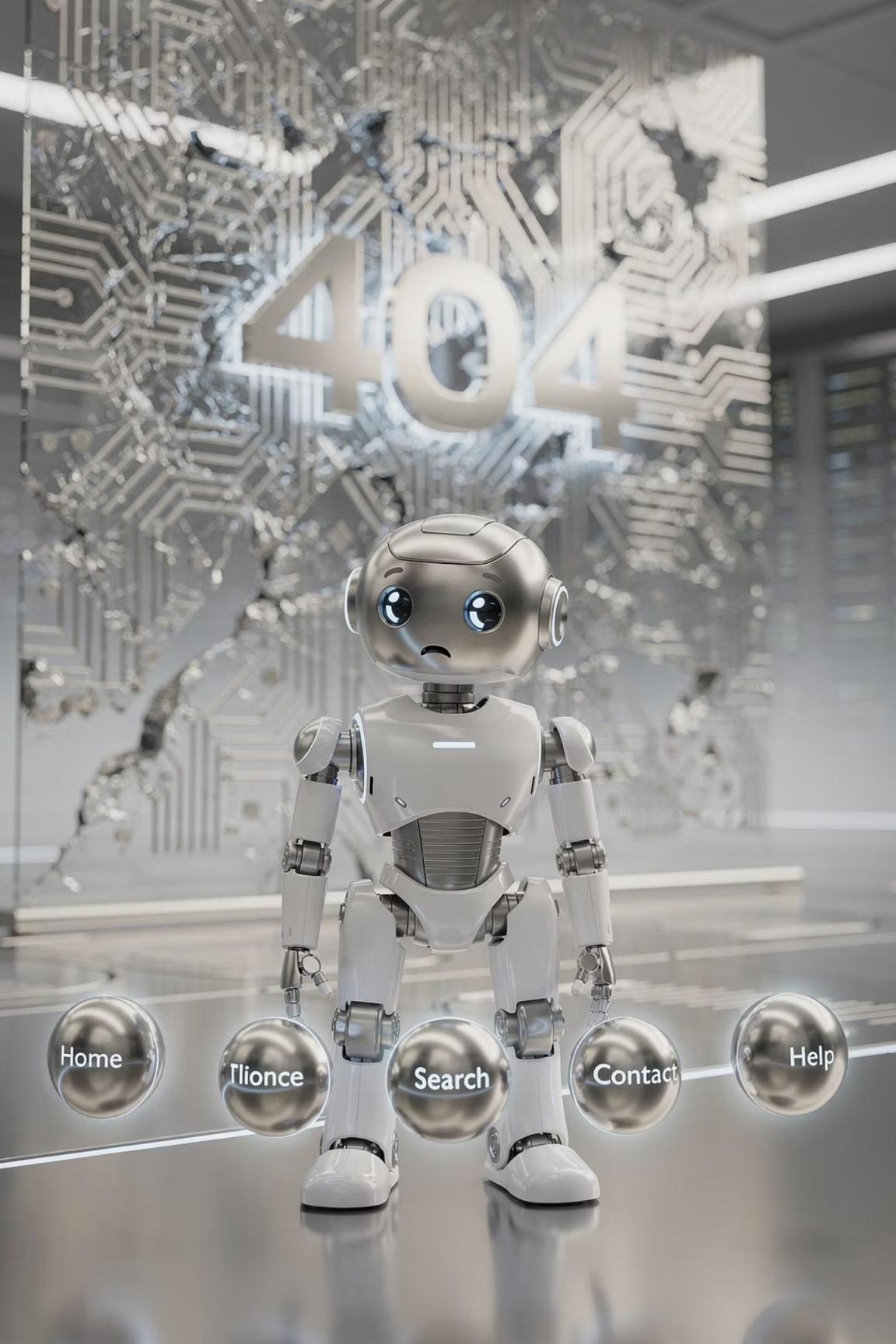
Para navegar para uma rota dinâmica, você simplesmente passa o valor desejado no caminho:

```
// app/products/index.jsx
<Link push href="/products/1">Produto 1 ➔</Link>
<Link push href="/products/2">Produto 2 ➔</Link>
```

⚠ CAPÍTULO 03

Criando uma Tela de Not Found Personalizada

Parâmetros na Rota, Rotas Aninhadas e Tela de Not Found



Por Que Personalizar a Página de Erro?

Por padrão, o Expo Router exibe uma página genérica de erro quando o usuário tenta acessar uma rota inexistente. Embora funcional, essa tela padrão não reflete a identidade visual do seu aplicativo.

Criar uma página de Not Found personalizada permite oferecer uma experiência mais amigável ao usuário, com mensagens claras e opções de navegação para retornar às áreas principais do app.

Criando o Arquivo +not-found.jsx

Convenção de Nomenclatura

O arquivo deve ser criado na raiz da pasta `app` com o nome exato:

`+not-found.jsx`

O símbolo `+` no início é obrigatório e indica que este é um arquivo especial de erro. Não use letras maiúsculas no nome.

```
import { router } from "expo-router"
import { Pressable, Text, View } from "react-native"
import { styles } from "../styles/styles"

export default function NotFound() {
  const goToHome = () => {
    router.replace("/")
  }
  return (
    <View style={[styles.container, { backgroundColor: "#F7D9C4" }]}>
      <Text>Essa página não existe</Text>
      <Pressable onPress={goToHome}>
        <Text> Ir para Home ➔</Text>
      </Pressable>
    </View>
  )
}
```

Registrando a Rota de Erro na Stack

A tela de Not Found deve ser registrada como a **última rota** na Stack. Isso garante que, se nenhuma outra rota corresponder ao caminho acessado, o usuário será direcionado para esta página de erro.

```
<Stack screenOptions={{ headerStyle: { backgroundColor: "#E94560" }, headerTintColor: "#FFFFFF" }}>
  <Stack.Screen name="index" options={{ headerTitle: "Home" }} />
  <Stack.Screen name="components/settings" options={{ headerTitle: "Configurações" }}/>
  <Stack.Screen name="components/user" options={{ headerTitle: "Usuário" }}/>
  <Stack.Screen name="products/index" options={{ headerTitle: "Produtos" }} />
  <Stack.Screen name="products/[id]" options={{ headerTitle: "Detalhes do Produto" }} />
  <Stack.Screen name="+not-found" options={{ headerTitle: "ERRO", headerLeft: () => null }} />
</Stack>
```

A ordem é importante: rotas específicas primeiro, rota de erro por último.

para testar a página de erro, registre em app/products/index.js uma rota inexistente:

```
<Link push href="/rssrsrss"> Ir para ERRO ➔</Link>
```

Usando Replace ao Invés de Push

1

Problema com Push

Se usar navegação normal (`href` ou `router.push`), o usuário pode pressionar "voltar" e retornar à página de erro, criando uma experiência confusa.

2

Solução com Replace

Use `router.replace("/")` para substituir a página de erro pela Home na pilha de navegação. Assim, ao pressionar "voltar", o usuário não retorna ao erro.

```
const goToHome = () => {
  router.replace("/")
}
```

Estrutura Completa do Projeto

Principais Aprendizados

- Rotas aninhadas organizam a navegação em hierarquias lógicas usando pastas
- Arquivos `index.jsx` representam a rota raiz de cada pasta
- Colchetes `[nome]` criam parâmetros dinâmicos nas rotas
- Hook `useLocalSearchParams()` captura valores dos parâmetros
- Arquivo `+not-found.jsx` personaliza a página de erro 404
- Usar `router.replace()` evita retorno indesejado à página de erro

app/_layout.jsx

Configuração da Stack com todas as rotas

app/index.jsx

Página inicial com links de navegação

app/components/settings.jsx

Página de configurações com links de navegação

app/components/user.jsx

Página do usuário com links de navegação

app/products/index.jsx

Lista de produtos com links dinâmicos

app/products/[id].jsx

Detalhes do produto com parâmetro

app/+not-found.jsx

Página de erro personalizada

Firjan SENAI

