



# **TRABALHANDO COM DADOS**

NAVEGAÇÃO ENTRE TELAS EM  
UM APLICATIVO FLUTTER



## USANDO O WIDGET NAVIGATOR:

- *O Flutter fornece o widget Navigator para gerenciar a pilha de telas (ou rotas) em um aplicativo.*
- *Você pode usar o Navigator para empilhar uma nova tela na parte superior da pilha, empurrando-a para a frente, ou para remover uma tela da pilha, empurrando-a para trás.*

- Para navegar para uma nova tela usando o Navigator, você normalmente precisa executar as seguintes etapas:

- **IMPORTE O PACOTE DO FLUTTER:**

```
import 'package:flutter/material.dart';
```

- Crie uma função para lidar com a navegação:

```
void navigateToNewScreen(BuildContext context) {  
  Navigator.push(  
    context,  
    MaterialPageRoute(builder: (context) => NewScreen()),  
  );  
}
```

- CHAME A FUNÇÃO DE NAVEGAÇÃO EM RESPOSTA A UM EVENTO, COMO UM BOTÃO PRESSIONADO:

```
ElevatedButton(  
  onPressed: () {  
    navigateToNewScreen(context);  
  },  
  child: Text('Ir para a nova tela'),  
)
```

- NO EXEMPLO ACIMA, NEWSSCREEN() REPRESENTA A TELA PARA A QUAL VOCÊ DESEJA NAVEGAR. AO CHAMAR NAVIGATOR.PUSH, VOCÊ ESTÁ ADICIONANDO UMA NOVA ROTA À PILHA E A TELA SERÁ EMPURRADA PARA A FRENTE.

# **USANDO ROTAS NOMEADAS:**

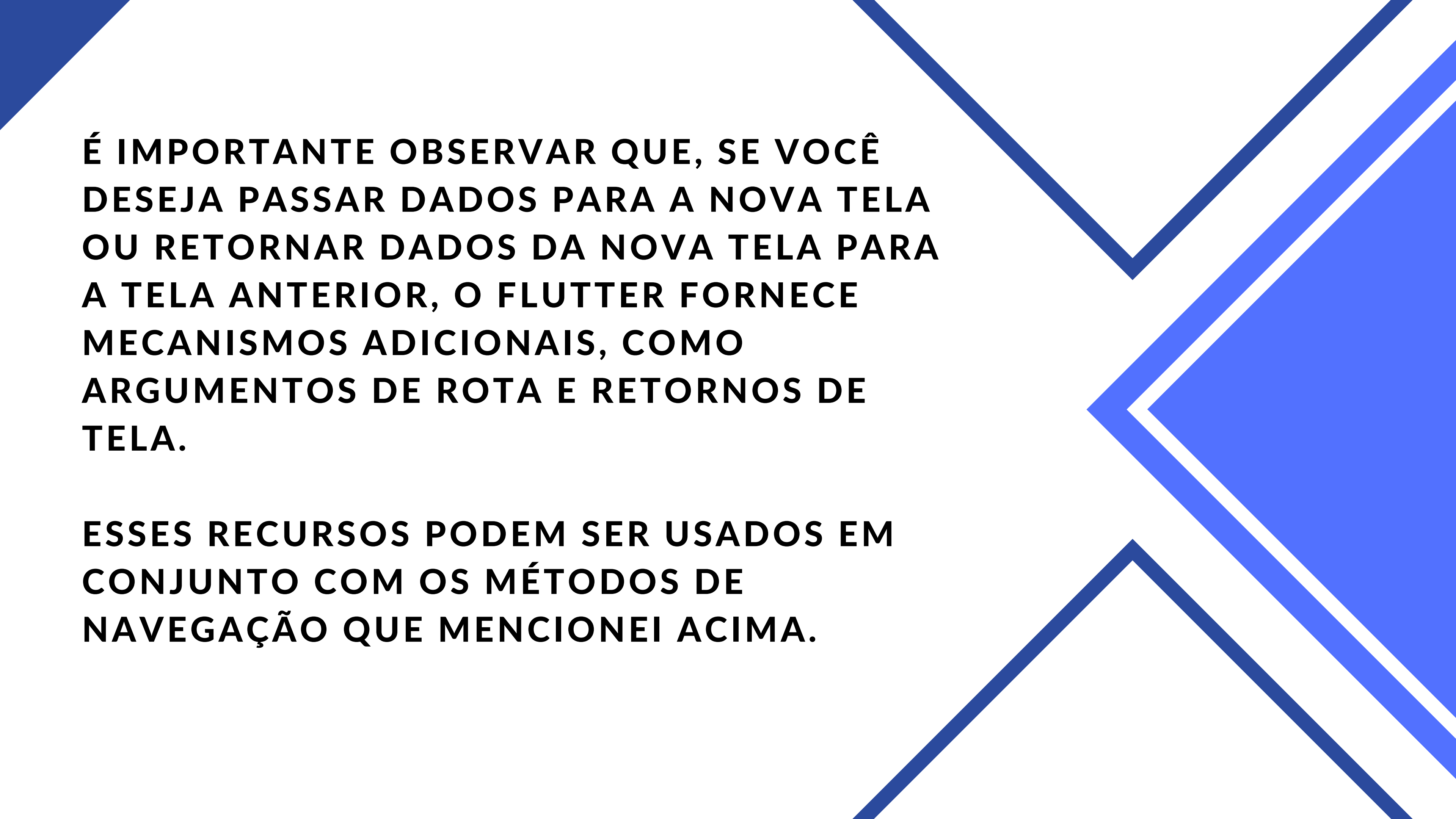
**O USO DE ROTAS NOMEADAS FORNECE UMA MANEIRA MAIS ESTRUTURADA DE NAVEGAR ENTRE TELAS, PERMITINDO QUE VOCÊ DEFINA UM NOME PARA CADA ROTA E NAVEGUE PARA ELA USANDO ESSE NOME. ISSO TORNA O CÓDIGO MAIS ORGANIZADO E FACILITA A MANUTENÇÃO.**

- **DECLARE AS ROTAS NOMEADAS NO MÉTODO MATERIALAPP OU CUPERTINOAPP:**

```
MaterialApp(  
  // outras configurações...  
  routes: {  
    '/': (context) => HomeScreen(),  
    '/new': (context) => NewScreen(),  
  },  
);
```

- **NAVEGUE PARA UMA NOVA TELA USANDO O NOME DA ROTA:**

```
void navigateToNewScreen(BuildContext context) {  
  Navigator.pushNamed(context, '/new');  
}
```



**É IMPORTANTE OBSERVAR QUE, SE VOCÊ DESEJA PASSAR DADOS PARA A NOVA TELA OU RETORNAR DADOS DA NOVA TELA PARA A TELA ANTERIOR, O FLUTTER FORNECE MECANISMOS ADICIONAIS, COMO ARGUMENTOS DE ROTA E RETORNOS DE TELA.**

**ESSES RECURSOS PODEM SER USADOS EM CONJUNTO COM OS MÉTODOS DE NAVEGAÇÃO QUE MENCIONEI ACIMA.**

# GERENCIAMENTO DE ESTADO EM UM APLICATIVO FLUTTER

NO DESENVOLVIMENTO DE APLICATIVOS FLUTTER, O GERENCIAMENTO DE ESTADO É UMA PARTE FUNDAMENTAL PARA MANTER A CONSISTÊNCIA E A SINCRONIZAÇÃO DOS DADOS ENTRE OS DIFERENTES COMPONENTES DA INTERFACE DO USUÁRIO.

EXISTEM VÁRIAS ABORDAGENS PARA O GERENCIAMENTO DE ESTADO EM FLUTTER, INCLUINDO O USO DE BIBLIOTECAS EXTERNAS, COMO O PROVIDER, MOBX, REDUX, ENTRE OUTRAS. ALÉM DISSO, O PRÓPRIO FLUTTER POSSUI UM MECANISMO INTERNO CHAMADO "GERENCIAMENTO DE ESTADO LOCAL" QUE PERMITE CONTROLAR O ESTADO EM UM ÚNICO WIDGET.



# GERENCIAMENTO DE ESTADO LOCAL (STATEFUL WIDGETS): USE WIDGETS STATEFUL PARA CONTROLAR O ESTADO DENTRO DE UM ÚNICO WIDGET.

1

## **PROVIDER:**

*Uma biblioteca popular que permite compartilhar estado eficientemente entre widgets, seguindo o conceito de Injeção de Dependência. É útil para atualizar dados de forma reativa.*

2

## **MOBX**

*Uma biblioteca poderosa que usa observáveis e reações para rastrear dependências entre os dados e atualizar automaticamente a interface do usuário. É útil para gerenciar estados complexos ou reativos.*

3

## **REDUX:**

*Gerenciamento de estado que usa um store centralizado para armazenar todo o estado do aplicativo e implementar um fluxo unidirecional para atualizar e recuperar dados, adequado para aplicativos com grande quantidade de estado compartilhado.*

# **USO DE WIDGETS PARA GERENCIAMENTO DE ESTADO:**

***NO FLUTTER, OS WIDGETS SÃO A  
PRINCIPAL FORMA DE GERENCIAR O  
ESTADO DENTRO DA HIERARQUIA DA  
INTERFACE DO USUÁRIO. EXISTEM  
ALGUNS TIPOS DE WIDGETS QUE  
VOCÊ PODE USAR PARA LIDAR COM O  
ESTADO EM SEU APLICATIVO:***

# STATELESS WIDGET:

1. **UM WIDGET "STATELESS" É IMUTÁVEL, O QUE SIGNIFICA QUE ELE NÃO POSSUI ESTADO INTERNO MUTÁVEL. ELE É CONSTRUÍDO APENAS COM BASE NOS PARÂMETROS FORNECIDOS. AO ATUALIZAR O ESTADO, UM NOVO WIDGET STATELESS É CONSTRUÍDO PARA REFLETIR AS ALTERAÇÕES. É ADEQUADO PARA COMPONENTES ESTÁTICOS QUE NÃO PRECISAM ATUALIZAR SEU ESTADO.**

## STATELESS WIDGET:

Um widget "stateless" é imutável, o que significa que ele não possui estado interno mutável. Ele é construído apenas com base nos parâmetros fornecidos. Ao atualizar o estado, um novo widget stateless é construído para refletir as alterações. É adequado para componentes estáticos que não precisam atualizar seu estado.

## STATEFUL

Um widget "stateful" tem um estado interno mutável que pode ser atualizado ao longo do tempo. Ele mantém o estado em uma classe separada chamada 'State', que é vinculada ao widget e gerencia as atualizações de estado. Ao chamar 'setState()', o Flutter reconstruirá o widget com base nas alterações do estado interno.

## INHERITEDWIDGET:

Um InheritedWidget é um widget especial do Flutter que permite compartilhar dados entre widgets descendentes de forma eficiente. Ele define um valor que pode ser acessado por qualquer widget descendente. Quando o valor é atualizado, os widgets descendentes que dependem dele são reconstruídos automaticamente.

# **TÉCNICAS PARA EVITAR PROBLEMAS DE PERFORMANCE AO GERENCIAR ESTADO**

AO GERENCIAR O ESTADO EM UM APLICATIVO FLUTTER, É IMPORTANTE CONSIDERAR ALGUMAS TÉCNICAS PARA EVITAR PROBLEMAS DE DESEMPENHO.

AQUI ESTÃO ALGUMAS DICAS PARA OTIMIZAR O GERENCIAMENTO DE ESTADO E MELHORAR O DESEMPENHO:

1. UTILIZE O MÍNIMO DE ESTADO NECESSÁRIO.
2. ATUALIZE O ESTADO DE FORMA EFICIENTE.
3. EVITE RECONSTRUÇÕES DESNECESSÁRIAS DE WIDGETS.
4. UTILIZE WIDGETS OTIMIZADOS PARA LISTAS.
5. MEMORIZE RESULTADOS COMPUTACIONALMENTE INTENSIVOS.
6. GERENCIE O ESCOPO DO ESTADO ADEQUADAMENTE.
7. OTIMIZE O ACESSO A DADOS EXTERNOS.

*Essas técnicas ajudarão a melhorar o desempenho do seu aplicativo Flutter, garantindo uma utilização eficiente do estado e evitando operações desnecessárias que possam impactar a experiência do usuário.*

# FONTES

[HTTPS://BR.ATSIT.IN/ARCHIVES/32663](https://br.atsit.in/archives/32663)

[HTTPS://DOCS.FLUTTER.DEV/PERF/BEST-PRACTICES](https://docs.flutter.dev/perf/best-practices)

[HTTPS://ACERVOLIMA.COM/FLUTTER-ROTAS-NOMEADAS/](https://acervolima.com/flutter-rotas-nomeadas/)

[HTTPS://BLOG.CISNE.DEV/NAVEGACAO-POR-ROTAS-EM-FLUTTER-USANDO-O-NUVIGATOR/](https://blog.cisne.dev/navegacao-por-rotas-em-flutter-usando-o-navigator/)

[HTTPS://ATELIWARE.COM/BLOG/GERENCIAMENTO-DE-ESTADO-EM-FLUTTER](https://ateliware.com/blog/gerenciamento-de-estado-em-flutter)

[HTTPS://DOCS.FLUTTER.DEV/DATA-AND-BACKEND/STATE-MGMT/OPTIONS](https://docs.flutter.dev/data-and-backend/state-mgmt/options)