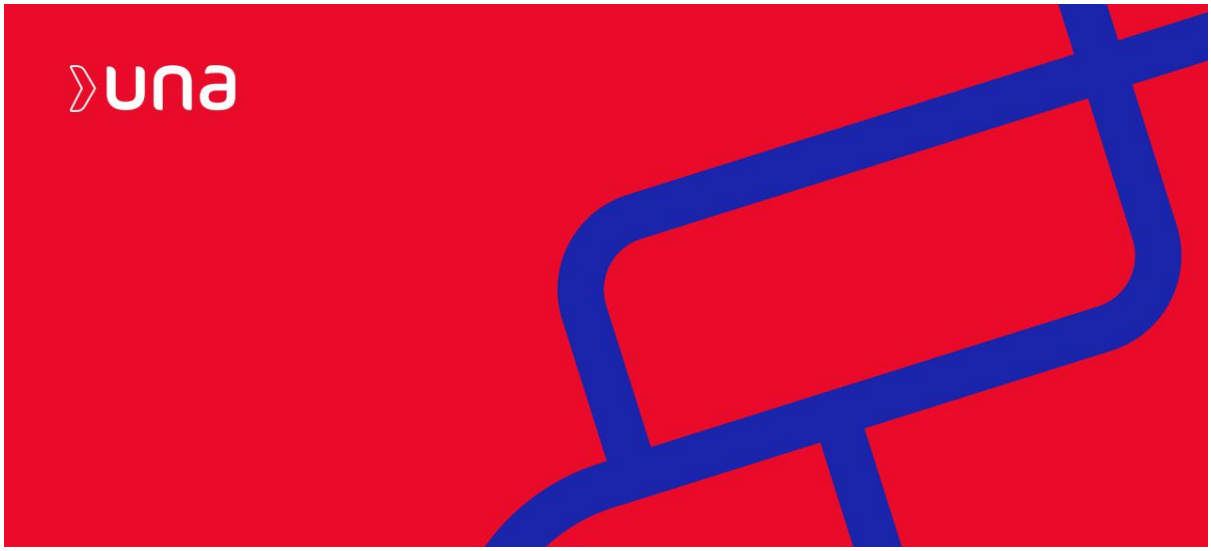


Flutter

Wesley Dias Maciel

2021/02

Centro Universitário UNA
Flutter
Wesley Dias Maciel
2021/02

Prática 12

Navegação com Rotas Nomeadas

Documentação: https://flutter.dev/docs/cookbook/navigation/named-routes, https://api.flutter.dev/flutter/widgets/Navigator-class.html

Objetivo: apresentar o processo de navegação através de rotas nomeadas.

Na prática anterior, você aprendeu como navegar para uma nova tela. Você criou uma rota e a empilhou através do widget Navigator. Entretanto, uma outra solução possível é definir uma rota nomeada e usá-la para navegação. Para trabalhar com rotas nomeadas, você deve usar o método Navigator.pushNamed (). Nesta prática, replicamos a funcionalidade da prática original, mas demonstrando o uso de rotas nomeadas.

Nesta prática, você vai:

1) Criar duas telas.
2) Definir as rotas nomeadas.
3) Navegar para a segunda tela usando o método Navigator.pushNamed ().
4) Retornar à primeira tela usando Navigator.pop ().

1) Crie um novo projeto Flutter, usando:
    a. Visual Studio Code, ou;
    b. https://dartpad.dev/, ou;
    c. https://flutlab.io/, ou;
    d. https://flutterstudio.app/, ou;
    e. https://codemagic.io/.

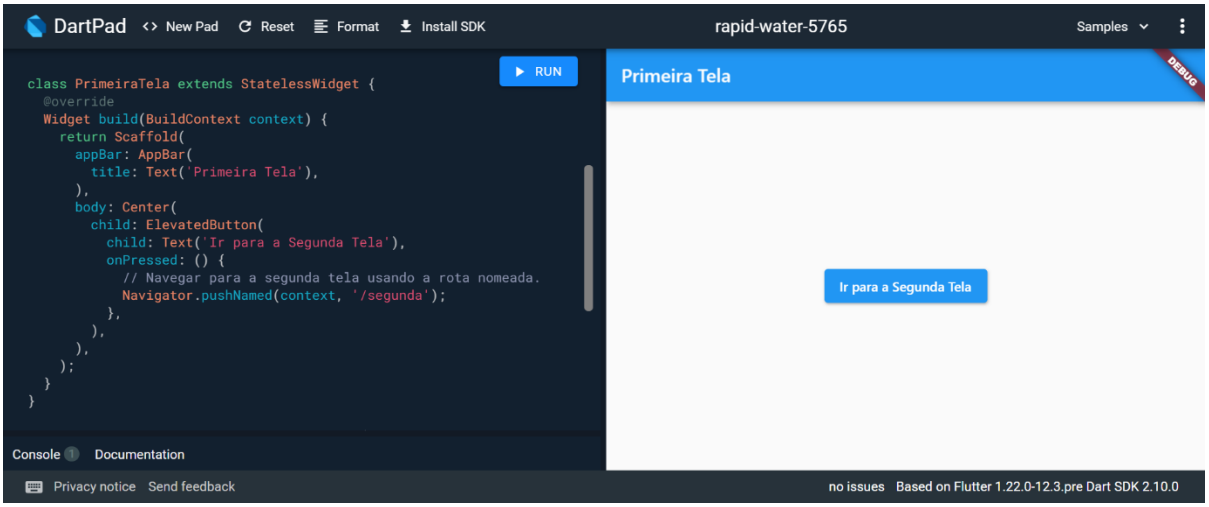
Criar as duas Rotas

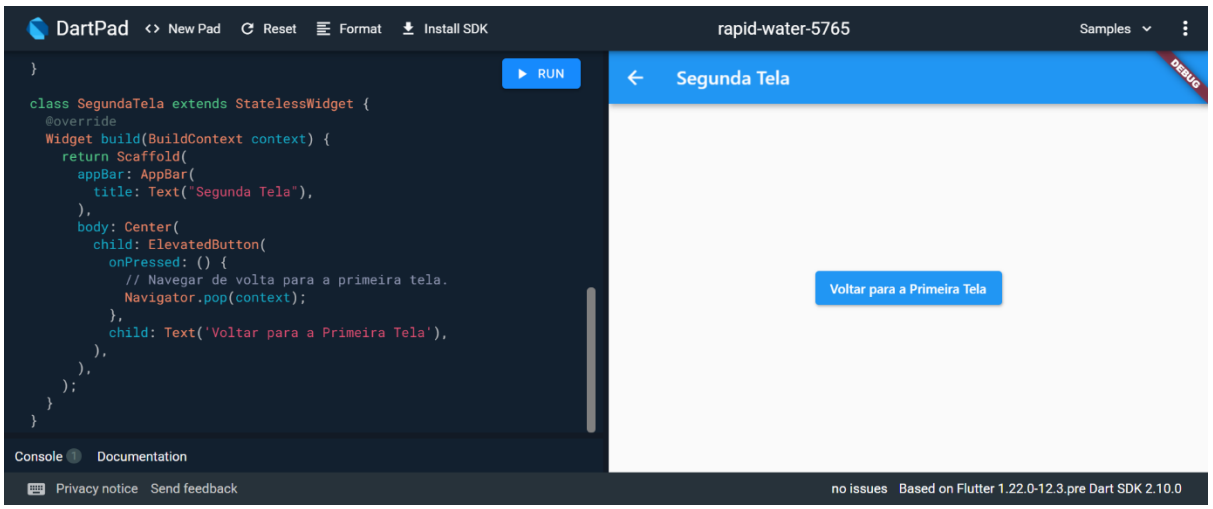
2) O exemplo abaixo cria as duas rotas. Cada rota é um widget.

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

void main() => runApp(
      MaterialApp(
```

```
        // Inicie o aplicativo com a rota nomeada "/".
        // Neste exemplo, o aplicativo inica no widget PrimeiraTela.
        initialRoute: '/',
        routes: {
          // Quando navegar pela rota nomeada "/", construa o widget PrimeiraT
ela.
          '/': (_) => PrimeiraTela(), //_ = context

          // Quando navegar pela rota nomeada "/segunda", construa o widget Se
gundaTela.
          '/segunda': (_) => SegundaTela(), //_ = context

        },
      ),
    );

class PrimeiraTela extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Primeira Tela'),
      ),
      body: Center(
        child: ElevatedButton(
          child: Text('Ir para a Segunda Tela'),
          onPressed: () {
            // Navegar para a segunda tela usando a rota nomeada.
            Navigator.pushNamed(context, '/segunda');
          },
        ),
      ),
    );
  }
}

class SegundaTela extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Segunda Tela"),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Navegar de volta para a primeira tela.
            Navigator.pop(context);
```

```
          },
          child: Text('Voltar para a Primeira Tela'),
        ),
      ),
    );
  }
}
```





Definir as rotas nomeadas

3) As rotas são definidas através dos atributos initialRoute e routes no construtor do widget MaterialApp, formando uma tabela de rotas.

```
    MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (_) => PrimeiraTela(), //_ = context
```

```
        '/segunda': (_) => SegundaTela(), //_ = context
      },
    ),
```

O atributo initialRoute define com qual rota nomeada o aplicativo deve começar. Essa é a rota inicial.

OBS: ao usar o atributo initialRoute, não use o atributo home do widget MaterialApp.

O atributo routes define as rotas nomeadas disponíveis e os widgets a serem construídos ao se navegar por elas.

Navigator.pushNamed ()

4) Com os widgets e rotas definidos na tabela de rotas, a navegação da primeira para a segunda tela ocorre usando o método Navigator.pushNamed (). Assim, no método build () do widget PrimeiraTela, atualizamos a função associada ao atributo onPressed:

```
        onPressed: () {
          Navigator.pushNamed(context, '/segunda');
        },
```

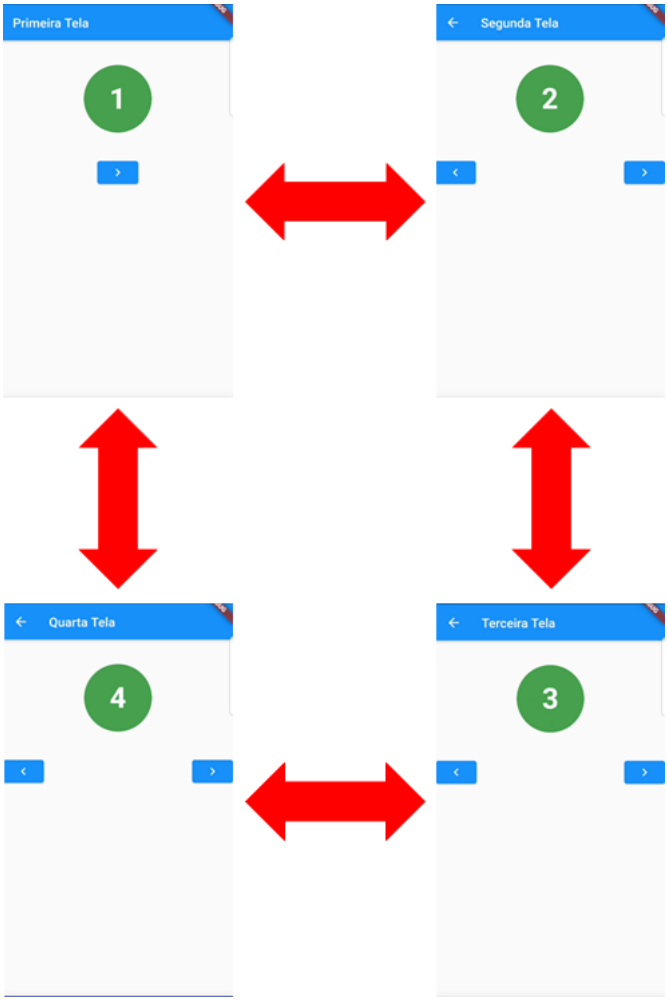
Navigator.pop()

5) Observe que a navegação de volta para a primeira tela é realizada empregando o método Navigator.pop () no atributo onPressed do botão da segunda tela. O parâmetro context é usado para assegurar o retorno ao widget pai do widget corrente na hierarquia de widgets.

```
        onPressed: () {
          Navigator.pop(context);
        },
```

Exercício

1) Usando o framework Flutter, escreva um algoritmo que implemente a transição de telas apresentada na figura abaixo.



Dica:

```
class SegundaTela extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Segunda Tela"),
      ),
      body: Center(
        child: Column(
          children: [
            Container(
              child: Text(
                '2',
                style: TextStyle(
```

```
                fontSize: 45,
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            ),
            decoration: BoxDecoration(
              shape: BoxShape.circle,
              color: Colors.green,
            ),
            padding: EdgeInsets.all(40),
            margin: EdgeInsets.all(25),
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              ElevatedButton(
                child: Icon(Icons.navigate_before),
                onPressed: () {
                  Navigator.pop(context);
                },
              ),
              ElevatedButton(
                child: Icon(Icons.navigate_next),
                onPressed: () {
                  Navigator.pushNamed(context, '/terceira');
                },
              ),
            ],
          )
        ],
      ),
    ),
  );
 }
}
```