



Flutter

Wesley Dias Maciel

2021/02

Prática 11

Alternar entre Telas

Documentação: <https://flutter.dev/docs/cookbook/navigation/navigation-basics>,
<https://api.flutter.dev/flutter/widgets/Navigator-class.html>,
<https://api.flutter.dev/flutter/material/MaterialPageRoute-class.html>

Objetivo: apresentar o widget Navigator e MaterialPageRoute para navegar entre telas.

Os aplicativos podem possuir várias telas para exibir informação. Por exemplo, um aplicativo pode ter uma tela que exibe produtos. Quando o usuário toca na imagem de um produto, uma nova tela exibe detalhes sobre o produto. No Flutter, as telas e páginas são chamadas de rotas. Uma rota é apenas um widget.

Nesta prática, você vai:

- 1) Criar duas rotas.
- 2) Navegar até a segunda rota usando Navigator.push ().
- 3) Retornar à primeira rota usando Navigator.pop ().

- 1) Crie um novo projeto Flutter, usando:
 - a. Visual Studio Code, ou;
 - b. <https://dartpad.dev/>, ou;
 - c. <https://flutlab.io/ide>, 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.

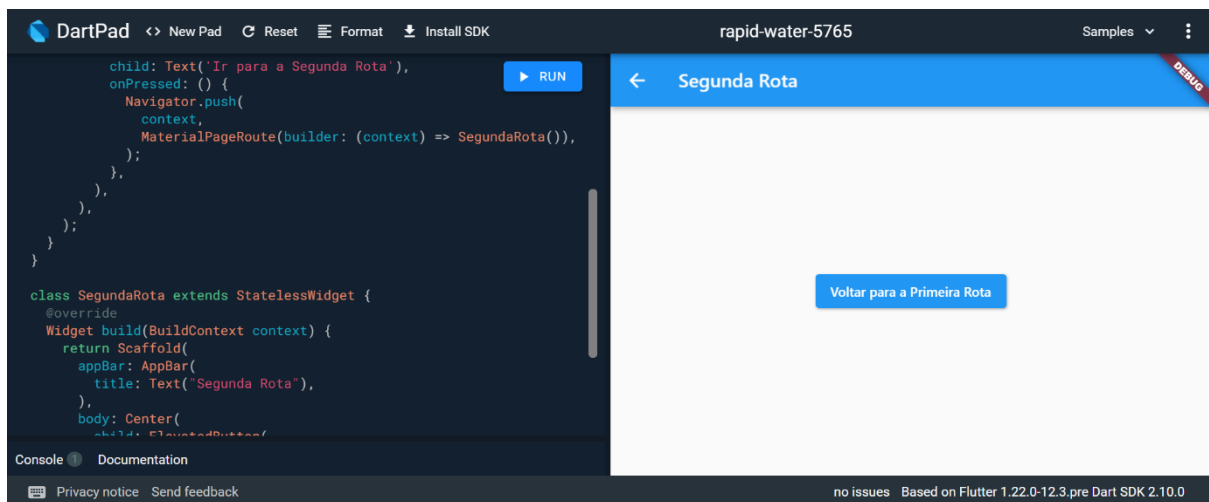
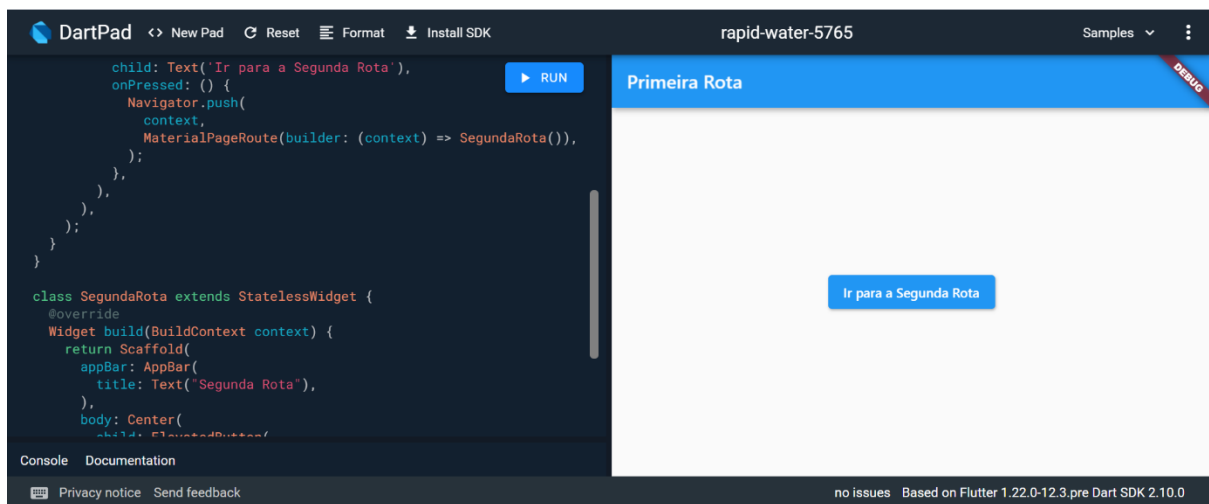
```
import 'package:flutter/material.dart';

void main() => runApp(
  MaterialApp(
    home: PrimeiraRota(),
  ),
);
```

```
);

class PrimeiraRota extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Primeira Rota'),
      ),
      body: Center(
        child: ElevatedButton(
          child: Text('Ir para a Segunda Rota'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SegundaRota()),
            );
          },
        ),
      ),
    );
  }
}

class SegundaRota extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Segunda Rota"),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pop(context);
          },
          child: Text('Voltar para a Primeira Rota'),
        ),
      ),
    );
  }
}
```

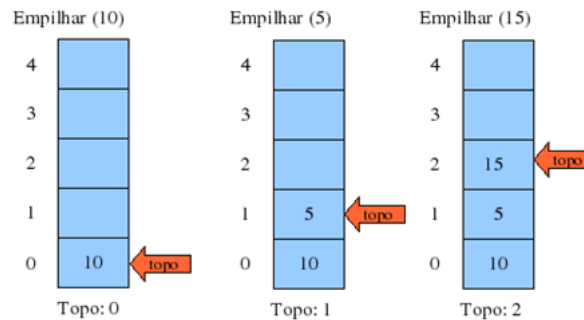


Navigator.push ().

- 3) Observe que a navegação até a segunda rota é realizada empregando o método `Navigator.push ()` no evento `onPressed` do botão da primeira rota. O parâmetro `context` armazena uma referência para o widget pai na hierarquia de widgets.

```
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => SegundaRota()),
  );
},
```

O método `push ()` empilha as rotas gerenciadas pelo Navigator. O empilhamento é semelhante ao que ocorre quando criamos uma pilha de pratos de cozinha. O primeiro prato é a base da pilha. Os pratos vão sendo colocados uns sobre os outros. O último prato é o topo da pilha. Exemplo:



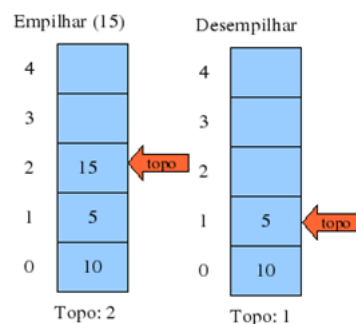
A transição de uma rota para outra pode ser feita através de um widget da classe `MaterialPageRoute`. O widget da classe `MaterialPageRoute` faz a transição para a nova rota usando uma animação específica da plataforma.

Navigator.pop()

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

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

O método `pop ()` desempilha as rotas gerenciadas pelo Navigator. O desempilhamento sempre ocorre a partir do topo da pilha. Assim, as retiradas sempre acontecem eliminando primeiro o prato que estiver no topo da pilha. Exemplo:



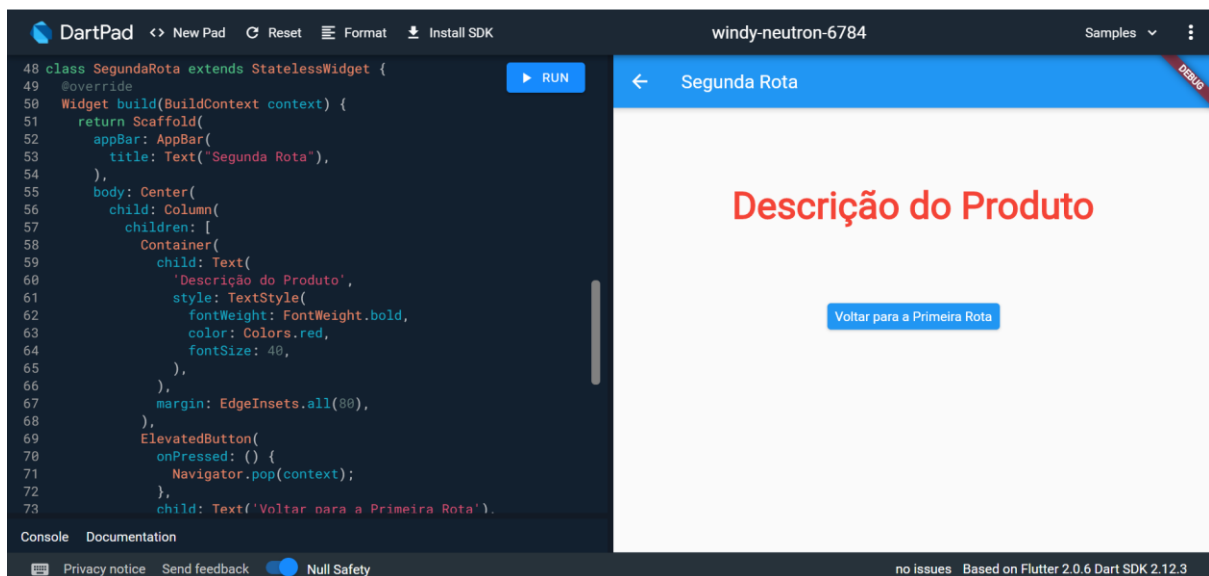
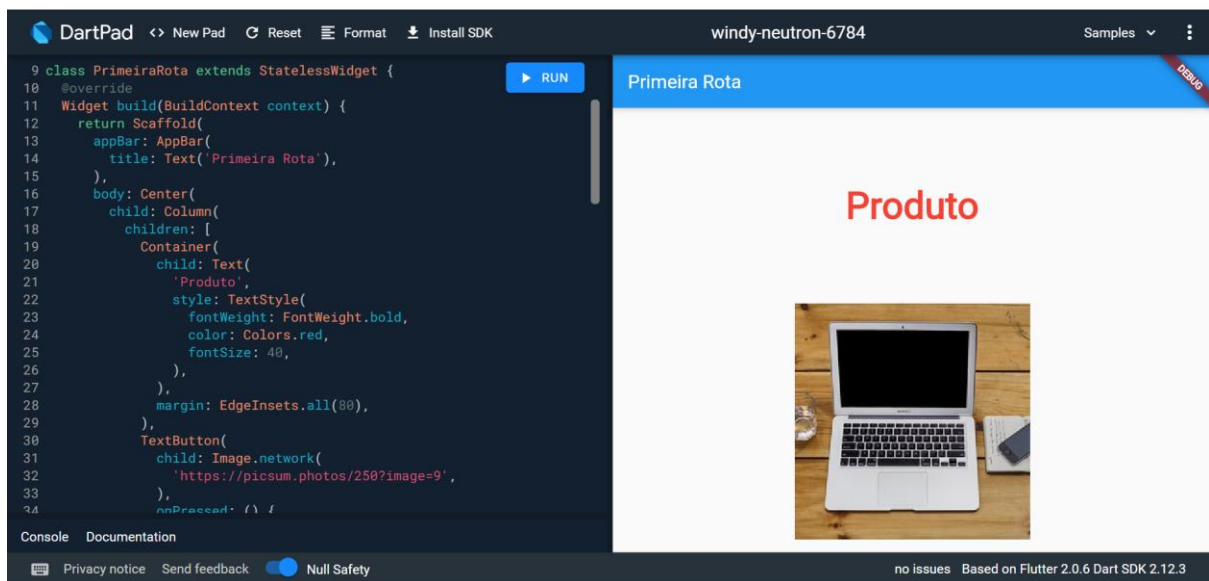
- 5) O exemplo abaixo apresenta uma tela de produto que leva para uma tela de descrição do produto. A aplicação é escrita inserindo uma imagem em um widget `TextButton`. Ao clicar na imagem da primeira rota, ocorre uma transição para a segunda rota. O inverso ocorre clicando-se no botão da segunda rota.

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

void main() => runApp(
  MaterialApp(
    home: PrimeiraRota(),
  ),
);

class PrimeiraRota extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Primeira Rota'),
      ),
      body: Center(
        child: Column(
          children: [
            Container(
              child: Text(
                'Produto',
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Colors.red,
                  fontSize: 40,
                ),
              ),
            ),
            margin: EdgeInsets.all(80),
          ],
        ),
        child: TextButton(
          child: Image.network(
            'https://picsum.photos/250?image=9',
          ),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SegundaRota()),
            );
          },
        ),
      ),
    );
  }
}
```

```
    ),  
  );  
}  
}  
  
class SegundaRota extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Segunda Rota"),  
      ),  
      body: Center(  
        child: Column(  
          children: [  
            Container(  
              child: Text(  
                'Descrição do Produto',  
                style: TextStyle(  
                  fontWeight: FontWeight.bold,  
                  color: Colors.red,  
                  fontSize: 40,  
                ),  
            ),  
            margin: EdgeInsets.all(80),  
          ),  
          ElevatedButton(  
            onPressed: () {  
              Navigator.pop(context);  
            },  
            child: Text('Voltar para a Primeira Rota'),  
          ),  
        ],  
      ),  
    );  
  }  
}
```



underline

6) Na linguagem Dart, o "underline" pode ser usado como nome de variável. Exemplos:

a)

```
void main() {
  int _ = 7;
  print(_);
}
```

b)


```
void main() {  
  int _ = 7, __ = 3;  
  print(_ + __);  
}
```

c)

```
void func(double _) => print(2 * _);  
void main() {  
  func(7);  
}
```

d)

```
double max(double _, double __, double ___) {  
  double resp = _;  
  if (resp < __) resp = __;  
  if (resp < ___) resp = ___;  
  return resp;  
}  
  
void main() {  
  print(max(7, 8, 9));  
}
```

Unused Parameter

- 7) Às vezes, a assinatura de uma função ou de um método exige um parâmetro, mas a implementação não usa esse parâmetro. Nesse caso, é idiomático nomear o parâmetro não utilizado como _ ("underline"). Se a função ou método tiver vários parâmetros não usados, use "underlines" adicionais para evitar colisões de nomes: _, __, ___, etc.

Exemplo:

```
import 'package:flutter/material.dart';  
  
void main() => runApp(MaterialApp(home: PrimeiraRota()));  
  
class PrimeiraRota extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(title: Text('Primeira Rota')),  
      body: Center(  
        child: Column(  

```

```
children: [
  Container(
    child: Text(
      'Produto',
      style: TextStyle(
        fontWeight: FontWeight.bold,
        color: Colors.red,
        fontSize: 40,
      ),
    ),
    margin: EdgeInsets.all(80),
  ),
  TextButton(
    child: Image.network('https://picsum.photos/250?image=9'),
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(builder: (_) => SegundaRota()),
      );
    },
  ),
],
),
);
}
}

class SegundaRota extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Segunda Rota"),
      ),
      body: Center(
        child: Column(
          children: [
            Container(
              child: Text(
                'Descrição do Produto',
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Colors.red,
                  fontSize: 40,
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

```
        margin: EdgeInsets.all(80),
      ),
      ElevatedButton(
        onPressed: () {
          Navigator.pop(context);
        },
        child: Text('Voltar para a Primeira Rota'),
      ),
    ],
  ),
),
);
}
```

Exercício

- 1) O algoritmo abaixo apresenta um álbum de fotografias. O algoritmo usa o widget ListView para criar uma lista com número indeterminado de elementos. A lista pode ser rolada para que o usuário veja os elementos.

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

void main() => runApp(
  MaterialApp(
    home: Home(),
  ),
);

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Álbum"),
        backgroundColor: Colors.green,
      ),
      body: Padding(
        padding: const EdgeInsets.all(8.0),
        child: ListView(
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: Image.network(
```

```
        "https://images.pexels.com/photos/213781/pexels-photo-213781.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213782/pexels-photo-213782.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213783/pexels-photo-213783.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213784/pexels-photo-213784.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213785/pexels-photo-213785.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213786/pexels-photo-213786.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213787/pexels-photo-213787.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: Image.network(
        "https://images.pexels.com/photos/213788/pexels-photo-213788.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
```

```
child: Image.network(  
  "https://images.pexels.com/photos/213789/pexels-photo-  
213789.jpeg?auto=compress&cs=tinysrgb&dpr=1&w=500"),  
),  
],  
,  
,  
,  
);  
}  
}
```

Usando o framework Flutter, altere o algoritmo para que o usuário possa ver uma tela que descreva cada foto do álbum, como apresentado abaixo. Ao clicar numa foto, o usuário deve ser direcionado para a tela de descrição. A tela de descrição deve possuir um botão para retornar à tela principal.

