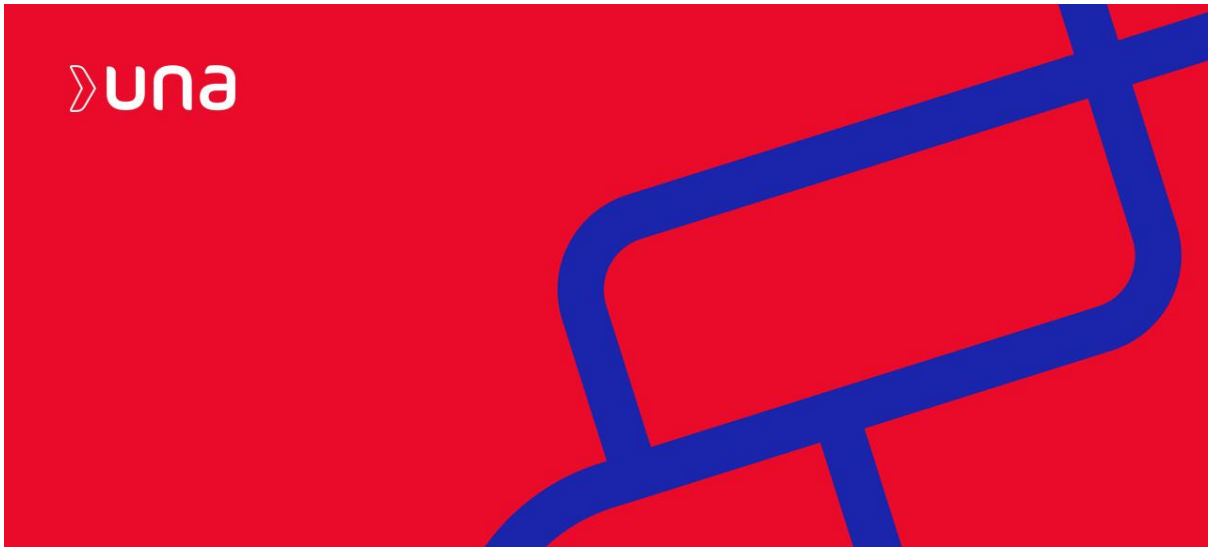


Flutter

Wesley Dias Maciel

2021/02

Centro Universitário UNA
Flutter
Wesley Dias Maciel
2021/02

Prática 17

Enviar Dados de uma Rota para Outra

- Parte 2 -

Documentação: https://flutter.dev/docs/cookbook/navigation/navigate-with-arguments, https://api.flutter.dev/flutter/widgets/Navigator-class.html

Objetivo: considerando um aplicativo com duas rotas, enviar argumentos da primeira rota para uma rota genérica. A rota genérica lê os argumentos recebidos e gera a segunda rota.

Nesta prática, você vai:

1) Criar duas rotas.
    - Definir a primeira rota.
    - Definir uma rota genérica que irá ler os argumentos recebidos e gerar uma segunda rota.
2) Definir uma classe cujo objeto encapsulará os argumentos enviados para a rota genérica.
3) Navegar da primeira rota para a rota genérica usando o método Navigator.pushNamed ().

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/.

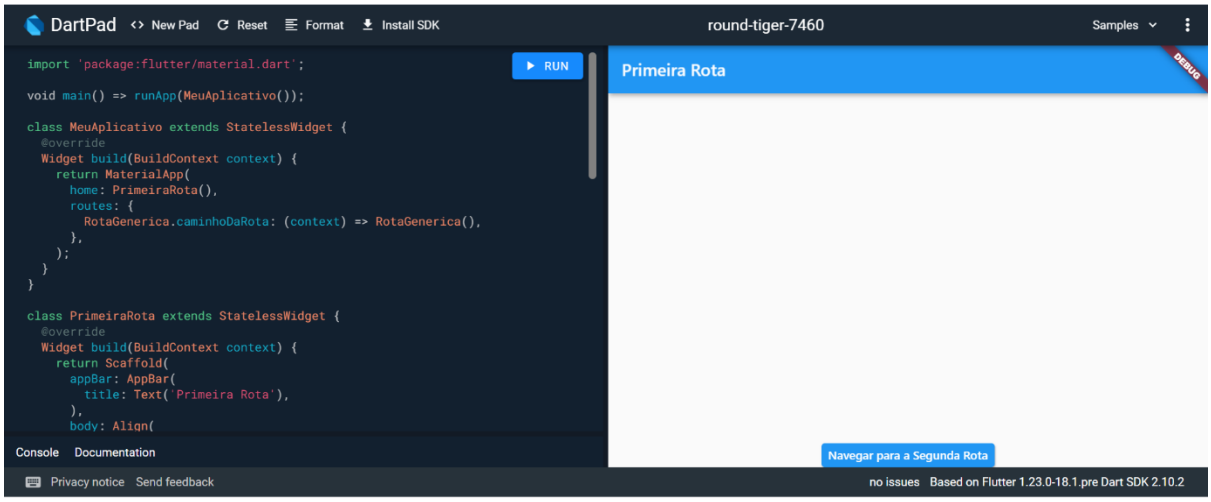
Criar duas Rotas

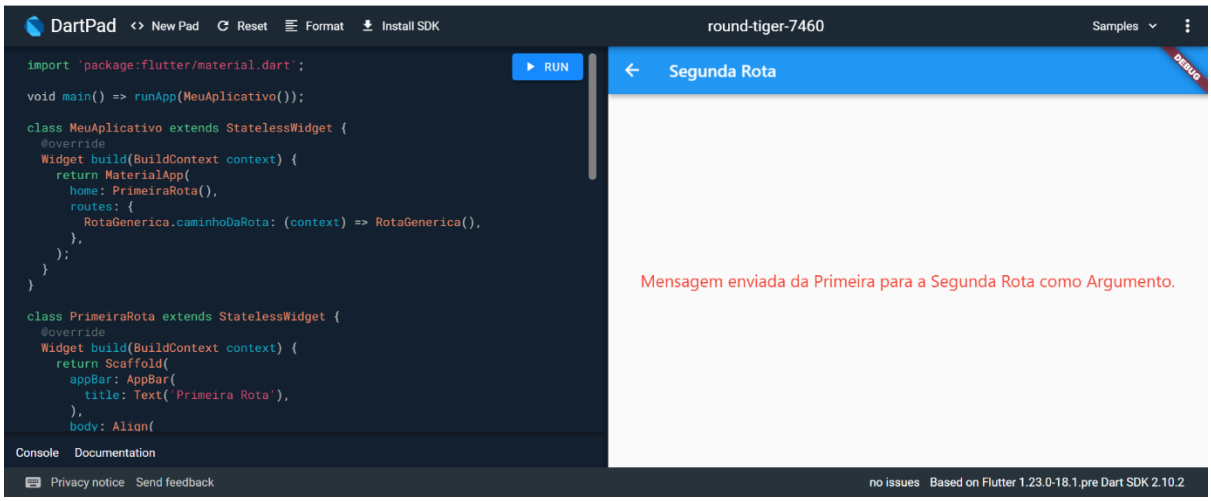
2) O exemplo abaixo cria a primeira rota e a rota genérica.

```
/*
 * Este exemplo exige que a linguagem Dart trate o código como
 * "null safe". Para que o SDK suporte "null safety", o arquivo
 * "pubspec.yaml" deve ter a seguinte instrução:
```

```
    environment:
      sdk: ">=2.12.0 <3.0.0"

 *
 * OBS 1: quando você quer especificar que uma variável pode
 *        armazenar um valor ou null, seu tipo deve ser declarado
 *        explicitamente como nullable. Isso é feito colocando um
 *        sinal de interrogação (?) após o nome do tipo.
 *
 * OBS 2: o sinal de exclamação (!) converte um tipo anulável
 *        (nullable) em um tipo não anulável (non-nullable).
*/

import 'package:flutter/material.dart';

void main() => runApp(MeuAplicativo());

class MeuAplicativo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: PrimeiraRota(),
      routes: {
        RotaGenerica.caminhoDaRota: (context) => RotaGenerica(),
      },
    );
  }
}

class PrimeiraRota extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Primeira Rota'),
      ),
      body: Align(
        alignment: Alignment.bottomCenter,
        child: ElevatedButton(
          child: Text("Navegar para a Segunda Rota"),
          onPressed: () {
            Navigator.pushNamed(
              context,
              RotaGenerica.caminhoDaRota,
              arguments: ArgumentosDaRota(
                'Segunda Rota',
```

```
                'Mensagem enviada da Primeira para a Segunda Rota como Argumen
to.',
              ),
            );
          },
        ),
      ),
    );
  }
}

class RotaGenerica extends StatelessWidget {
  static const caminhoDaRota = '/rotaGenerica';

  @override
  Widget build(BuildContext context) {
    ArgumentosDaRota argumentos =
        ModalRoute.of(context)!.settings.arguments as ArgumentosDaRota;

    return Scaffold(
      appBar: AppBar(
        title: Text(argumentos.titulo),
      ),
      body: Center(
        child: Text(
          argumentos.mensagem,
          style: TextStyle(
            fontSize: 20,
            color: Colors.red,
          ),
        ),
      ),
    );
  }
}

class ArgumentosDaRota {
  String titulo;
  String mensagem;

  ArgumentosDaRota(this.titulo, this.mensagem);
}
```





Rota Nomeada

3) O primeiro passo é definir uma rota nomeada no MaterialApp, usando o atributo routes. No exemplo, o caminho da rota foi informado no atributo caminhoDaRota da rota genérica.

```
class MeuAplicativo extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: PrimeiraRota(),
      routes: {
        RotaGenerica.caminhoDaRota: (context) => RotaGenerica(),
      },
    );
  }
}
```

Primeira Rota

4) Definição da primeira rota.

```
class PrimeiraRota extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Primeira Rota'),
      ),
      body: Align(
        alignment: Alignment.bottomCenter,
        child: ElevatedButton(
          child: Text("Navegar para a Segunda Rota"),
          onPressed: () {
            Navigator.pushNamed(
              context,
              RotaGenerica.caminhoDaRota,
              arguments: ArgumentosDaRota(
                'Segunda Rota',
                'Mensagem enviada da Primeira para a Segunda Rota como Argumento.',
              ),
            );
          },
        ),
      ),
    );
  }
}
```

Rota Genérica

5) Definição da rota genérica. A rota genérica lê os argumentos recebidos e gera uma segunda rota. Os argumentos são o título e uma mensagem para o corpo da segunda rota.

```
class RotaGenerica extends StatelessWidget {
  static const caminhoDaRota = '/rotaGenerica';

  @override
  Widget build(BuildContext context) {
    ArgumentosDaRota argumentos =
        ModalRoute.of(context)!.settings.arguments as ArgumentosDaRota;

    return Scaffold(
```

```
      appBar: AppBar(
        title: Text(argumentos.titulo),
      ),
      body: Center(
        child: Text(
          argumentos.mensagem,
          style: TextStyle(
            fontSize: 20,
            color: Colors.red,
          ),
        ),
      ),
    );
  }
}
```

Argumentos

6) Definição da classe cujo objeto encapsulará os argumentos enviados para a rota genérica. No exemplo, os argumentos são o título e mensagem de corpo para geração da segunda rota.

```
class ArgumentosDaRota {
  String titulo;
  String mensagem;

  ArgumentosDaRota(this.titulo, this.mensagem);
}
```

Navegação

7) No atributo onPressed do botão da primeira rota, permitir a navegação para a rota genérica através do método Navigator.pushNamed (). O atributo arguments do método Navigator.pushNamed () recebe um objeto da classe ArgumentosDaRota. Quando o usuário clicar no botão, esse objeto será enviado como argumento para a rota genérica.

```
          onPressed: () {
            Navigator.pushNamed(
              context,
              RotaGenerica.caminhoDaRota,
              arguments: ArgumentosDaRota(
                'Segunda Rota',
                'Mensagem enviada da Primeira para a Segunda Rota como Argumento.',
              ),
            );
          },
```

Leitura dos Argumentos

Na rota genérica, a leitura dos argumentos é realizada através da classe ModalRoute:

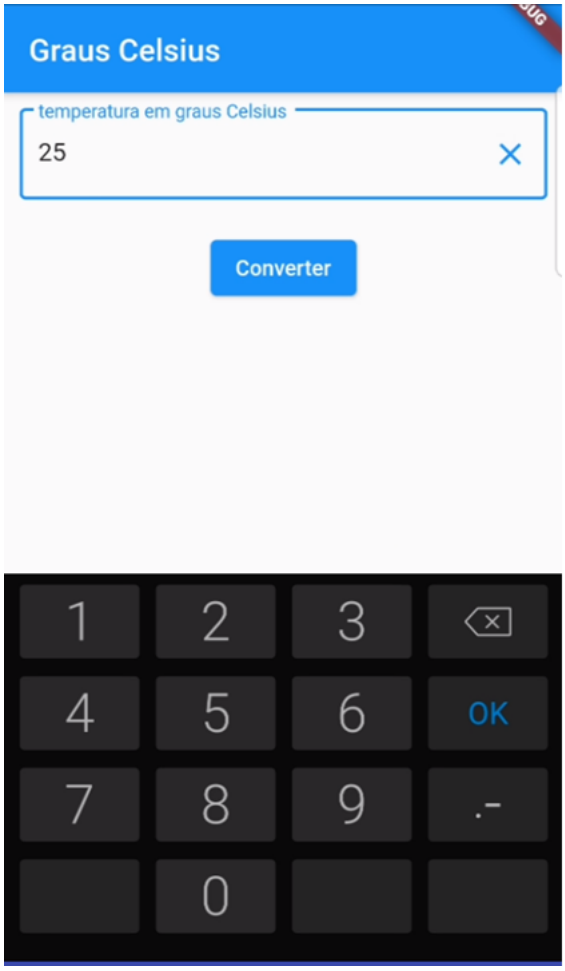
```
ArgumentosDaRota argumentos =
      ModalRoute.of(context)!.settings.arguments as ArgumentosDaRota;
```

Exercício

1) Altere o algoritmo apresentado nesta prática, gerando um conversor de temperatura. O aplicativo deve ler uma temperatura em graus Celsius e apresentar a temperatura correspondente em graus Fahrenheit. A temperatura em graus Celsius deve ser lida na primeira rota e enviada como argumento para a segunda rota. A segunda rota deve calcular a temperatura correspondente em graus Fahrenheit e apresentar o resultado. Fórmula:

$$°F = °C \times 1,8 + 32$$

Em que °F é a temperatura em graus Fahrenheit e °C é a temperatura em graus Celsius.



Dicas:

a) Teclado numérico:

```
        TextField(
             controller: temperaturaCelsiusController,
             keyboardType: TextInputType.number,
             decoration: InputDecoration(
               suffixIcon: IconButton(
                 onPressed: () => temperaturaCelsiusController.clear(),
                 icon: Icon(Icons.clear),
               ),
               border: OutlineInputBorder(),
               labelText: 'temperatura em graus Celsius',
             ),
           ),
```

b) Método conversor:

```
converter(double celsius) => celsius * 1.8 + 32;
```

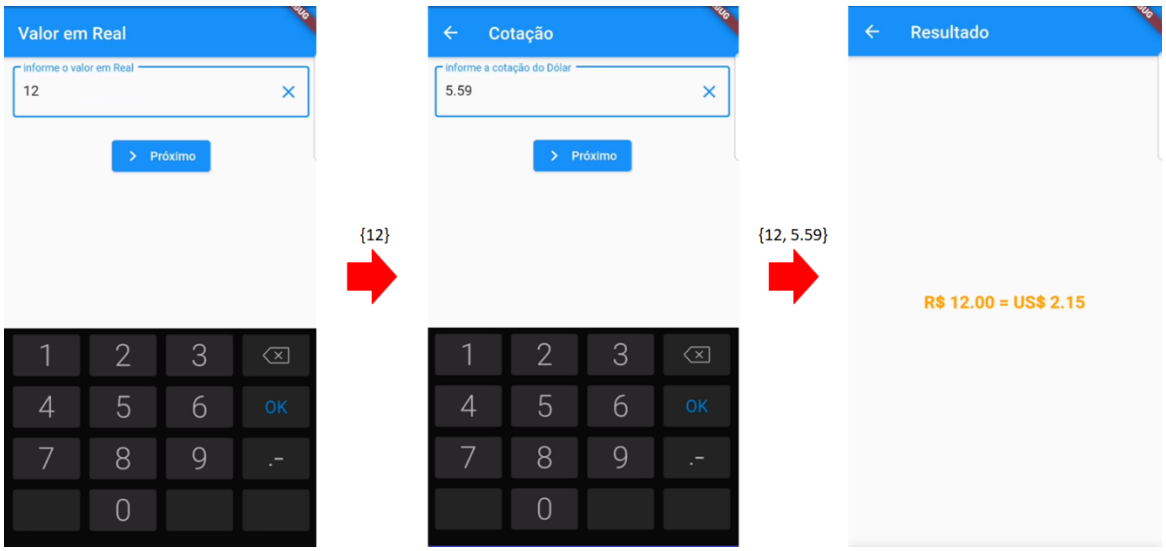
c) Precisão:

```
            Text(
              'Graus Celsius: ${argumentos.celsius.toStringAsFixed(2)}',
              style: TextStyle(
                fontSize: 20,
                color: Colors.green,
              ),
            ),
```

d) Classe para encapsular os argumentos:

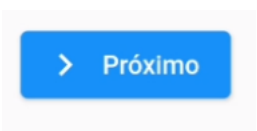
```
class ArgumentosDaRota {
  String titulo;
  double celsius;

  ArgumentosDaRota(this.titulo, this.celsius);
}
```

2) Usando o framework Flutter, escreva um aplicativo para conversão de Dólar em Real. O aplicativo deve possuir 3 rotas. A primeira rota deve ler o valor em Real e enviá-lo como argumento para a segunda rota. A segunda rota deve ler a cotação do Dólar e enviá-la, juntamente com o valor em real, para a terceira rota. O envio deve ser feito como argumento. A terceira rota deve calcular e apresentar o resultado da conversão.



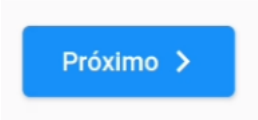
Dica:

a) Exemplos de botão:




```
          Container(
            margin: EdgeInsets.all(10),
```

```
      child: ElevatedButton.icon(
        label: const Text('Próximo'),
        icon: Icon(Icons.navigate_next),
        onPressed: () {
          Navigator.pushNamed(
            context,
            SegundaRota.caminhoDaRota,
            arguments: double.parse(valorEmRealController.text),
          );
        },
      ),
    ),
```

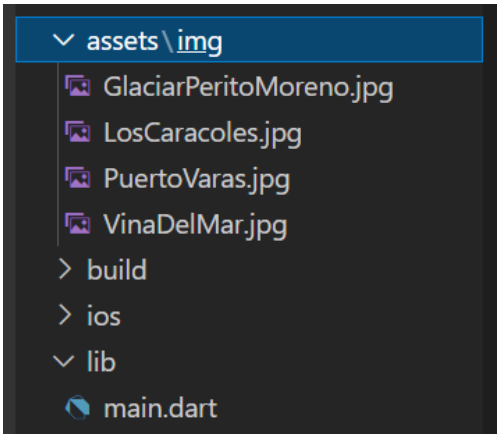



```
    Container(
      margin: EdgeInsets.all(10),
      child: ElevatedButton(
        child: Padding(
          padding: EdgeInsets.fromLTRB(5, 7, 0, 7),
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: <Widget>[
              Text('Próximo'),
              Icon(
                Icons.navigate_next,
              )
            ],
          ),
        ),
        onPressed: () {
          Navigator.pushNamed(
            context,
            SegundaRota.caminhoDaRota,
            arguments: double.parse(valorEmRealController.text),
          );
        },
      ),
    ),
```

3) O exemplo abaixo apresenta um carrossel de imagens. Altere o exemplo para que:
    a. Ao clicar numa imagem, o usuário seja redirecionado para uma segunda rota responsável por apresentar informação sobre a imagem selecionada, como título e descrição.

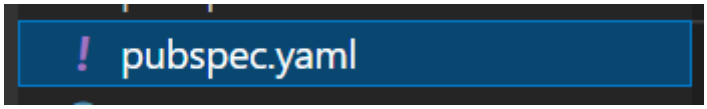
b. A primeira rota deve enviar a informação da imagem como argumento para a segunda rota.

No exemplo, as imagens devem ser armazenadas localmente. Para isso, crie a estrutura de diretórios "assets/img" na raiz do seu projeto. Armazene 4 imagens de sua preferência no diretório "img":



OBS: lembre-se de atualizar os nomes dos arquivos de suas imagens no código de exemplo.

Em seguida especifique a estrutura de diretórios "assets/img" no arquivo "pubspec.yaml" de seu projeto:



No arquivo "pubspec.yaml", procure a seção "assets". Exclua o comentário dessa seção e informe a estrutura de diretório "assets/img", como apresentado abaixo:

```
  # To add assets to your application, add an assets section, like this:
assets:
  - assets/img
  #   - images/a_dot_burr.jpeg
  #   - images/a_dot_ham.jpeg
```

Código:

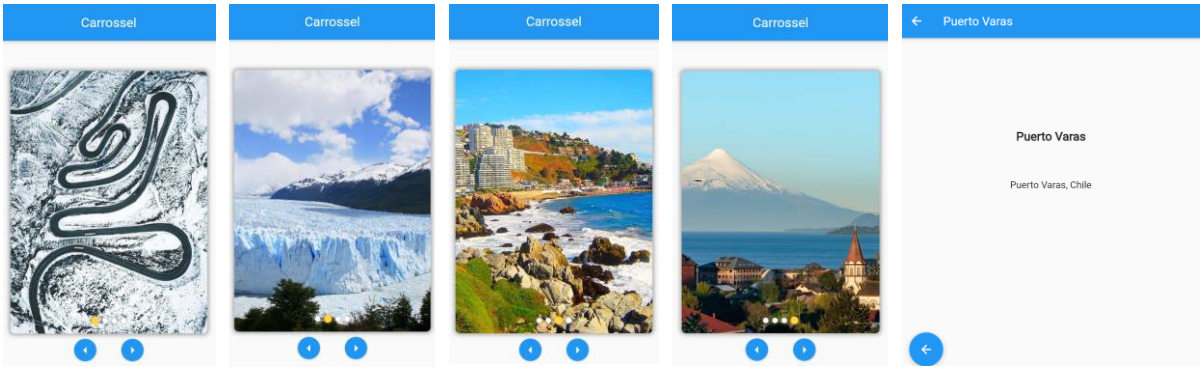
```
import 'package:flutter/material.dart';

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

```dart
class Carrossel extends StatefulWidget {
  @override
  CarrosselState createState() => CarrosselState();
}

class CarrosselState extends State<Carrossel> {
  int indice = 0;

  List<String> imagens = [
    'img/LosCaracoles.jpg',
    'img/GlaciarPeritoMoreno.jpg',
    'img/VinaDelMar.jpg',
    'img/PuertoVaras.jpg',
  ];

  void anterior() =>
      setState(() => indice = indice > 0 ? indice - 1 : imagens.length - 1);

  void posterior() =>
      setState(() => indice = indice < imagens.length - 1 ? indice + 1 : 0);

  Stack imagem() => Stack(
        children: [
          Container(
            height: 400,
            width: 300,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(7),
              image: DecorationImage(
                image: AssetImage(imagens[indice]),
                fit: BoxFit.cover,
              ),
              boxShadow: [
                BoxShadow(
                  color: Colors.grey,
                  spreadRadius: 2,
                  blurRadius: 5,
                )
              ],
            ),
          ),
          Positioned(
            top: 375,
            left: 25,
            right: 25,
            child: PainelPontos(
              numeroPontos: imagens.length,
```

```
            indice: indice,
          ),
        ),
      ],
    );

Row cursor() => Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Padding(
          padding: const EdgeInsets.all(8),
          child: ElevatedButton(
            child: Icon(Icons.arrow_left),
            onPressed: anterior,
            style: ElevatedButton.styleFrom(
              shape: CircleBorder(),
              padding: EdgeInsets.all(13),
            ),
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8),
          child: ElevatedButton(
            child: Icon(Icons.arrow_right),
            onPressed: posterior,
            style: ElevatedButton.styleFrom(
              shape: CircleBorder(),
              padding: EdgeInsets.all(13),
            ),
          ),
        ),
      ],
    );

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Carrossel'),
      centerTitle: true,
    ),
    body: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        imagem(),
        cursor(),
      ],
```

```
      ),
    );
  }
}

class PainelPontos extends StatelessWidget {
  final int? numeroPontos;
  final int? indice;

  PainelPontos({this.numeroPontos, this.indice});

  Widget fotoInativa() {
    return Padding(
      padding: EdgeInsets.only(left: 3, right: 3),
      child: Container(
        height: 8,
        width: 8,
        decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.circular(4),
          boxShadow: [
            BoxShadow(
              color: Colors.grey,
              spreadRadius: 0.3,
              blurRadius: 3,
            )
          ],
        ),
      ),
    );
  }

  Widget fotoAtiva() {
    return Padding(
      padding: EdgeInsets.only(left: 3, right: 3),
      child: Container(
        height: 11,
        width: 11,
        decoration: BoxDecoration(
          color: Colors.amberAccent,
          borderRadius: BorderRadius.circular(5),
          boxShadow: [
            BoxShadow(
              color: Colors.orangeAccent,
              spreadRadius: 0.3,
              blurRadius: 3,
            )
```

```
          ],
        ),
      ),
    );
  }

  List<Widget> gerarPainelPontos() {
    List<Widget> pontos = [];

    for (int i = 0; i < this.numeroPontos!; i++)
      pontos.add(i == indice ? fotoAtiva() : fotoInativa());

    return pontos;
  }

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: this.gerarPainelPontos(),
      ),
    );
  }
}
```



Dica:

```
  List<Map> imagens = [
    {
      'arquivo': 'img/LosCaracoles.jpg',
      'titulo': 'Los Caracoles',
      'descricao': 'Los Caracoles, Chile'
    },
    {
      'arquivo': 'img/GlaciarPeritoMoreno.jpg',
      'titulo': 'Glaciar Perito Moreno',
```

```
    'descricao': 'Glaciar Perito Moreno, Argentina'
  },
  {
    'arquivo': 'img/VinaDelMar.jpg',
    'titulo': 'Viña Del Mar',
    'descricao': 'Viña Del Mar, Chile'
  },
  {
    'arquivo': 'img/PuertoVaras.jpg',
    'titulo': 'Puerto Varas',
    'descricao': 'Puerto Varas, Chile'
  },
];
```