Prática 03

Estado

Documentação: https://flutter.dev/docs/development/ui/widgets-intro, https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html, https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html

Objetivo: apresentar as classes StatelessWidget e StatefulWidget.

No framework Flutter, a ideia central é que você construa sua interface a partir de widgets. A principal tarefa de um widget é implementar uma função build (). A função build () descreve esse widget em termos de outros widgets de nível inferior na árvore de renderização.

O framework Flutter constrói o widget até chegar em algum widget que represente um objeto da classe básica RenderObject. O objeto da classe RenderObject é responsável por calcular e descrever a geometria do widget.

Um widget descreve sua aparência a partir de sua configuração e estado atuais. Quando o estado de um widget muda, o widget reconstrói essa descrição, a fim de determinar as mudanças mínimas necessárias na árvore de renderização subjacente para a transição de um estado para o próximo.

Os widgets são descritos por classes. Ao escrever um aplicativo, você normalmente criará novos widgets que serão subclasses de StatelessWidget ou StatefulWidget. Seu widget será subclasse de StatelessWidget, se não precisar gerenciar algum estado. Por outro lado, seu widget será subclasse de StatefulWidget, caso faça a gerência de algum estado.

Para converter um widget que não gerencia estado em outro que gerencia, você precisa fazer duas coisas:

1) Converter a classe que estende StatelessWidget em uma classe de estado. Esta classe será responsável por determinar o estado do widget e terá que estender a classe State.

a. A classe State possui um método setState (). Você tem que usar o método setState () para alterar o estado interno de um objeto da classe State. Então, crie um método para promover a alteração de estado. Depois, passe esse método como parâmetro de setState ().

2) Criar uma nova classe que estende StatefulWidget. Essa classe terá que instanciar e retornar o objeto da classe de estado que você criou no passo anterior através do método createState ().

OBS: é uma boa prática de programação que o nome da classe de estado seja igual ao nome da classe que estende StatefulWidget concatenado com o string "State". No exemplo abaixo, a classe que estende StatefulWidget chama-se "PaginaInicial" e a classe que gerencia estado chama-se "PaginaInicial**State**":

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

void main() {
  runApp(MaterialApp(home: PaginaInicial()));
}

class PaginaInicial extends StatefulWidget {
  @override
  PaginaInicialState createState() {
    return PaginaInicialState();
  }
}

class PaginaInicialState extends State<PaginaInicial> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Página Inicial")),
    );
  }
}
```

Na definição da classe PaginaInicialState, a instrução State<PaginaInicial> informe que a classe é um estado para PaginaInicial:

```dart
class PaginaInicialState extends State<PaginaInicial>
```

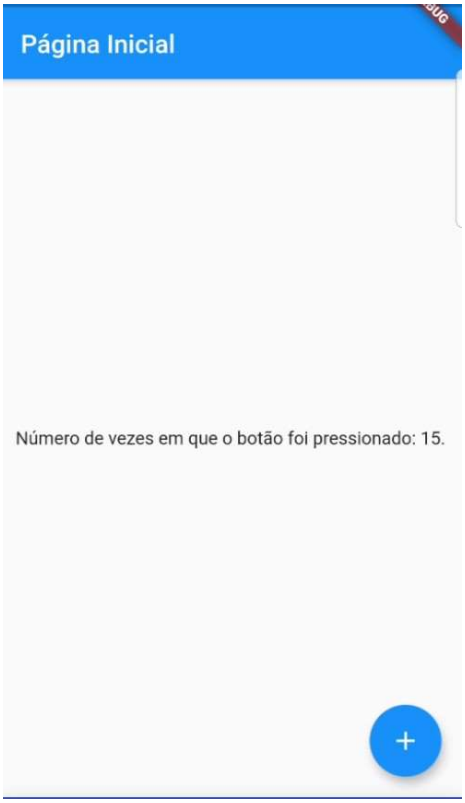
1) Crie um novo projeto Flutter com o exemplo abaixo, usando:
   a. https://zapp.run/, ou
   b. Visual Studio Code, ou;

c. https://dartpad.dev/, ou;
d. https://flutlab.io/ide, ou;
e. https://flutterstudio.app/, ou;
f. https://codemagic.io/.

2) O exemplo abaixo altera o estado de um widget ao clicar no botão apresentado na tela. Um método para incremento de uma variável foi implementado para promover a alteração de estado. Esse método foi associado ao evento de clicar no botão, onPressed.

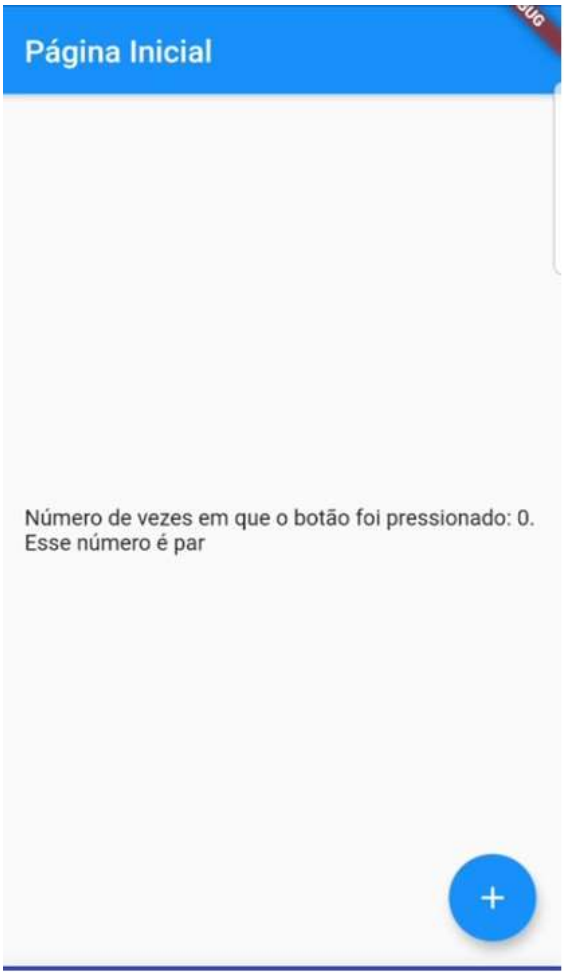
```
import 'package:flutter/material.dart';

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  int numeroVezes = 0;

  void cliqueDoBotao() {
    numeroVezes = numeroVezes + 1;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("Página Inicial"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
                "Número de vezes em que o botão foi pressionado:
$numeroVezes."),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(cliqueDoBotao);
        },
```

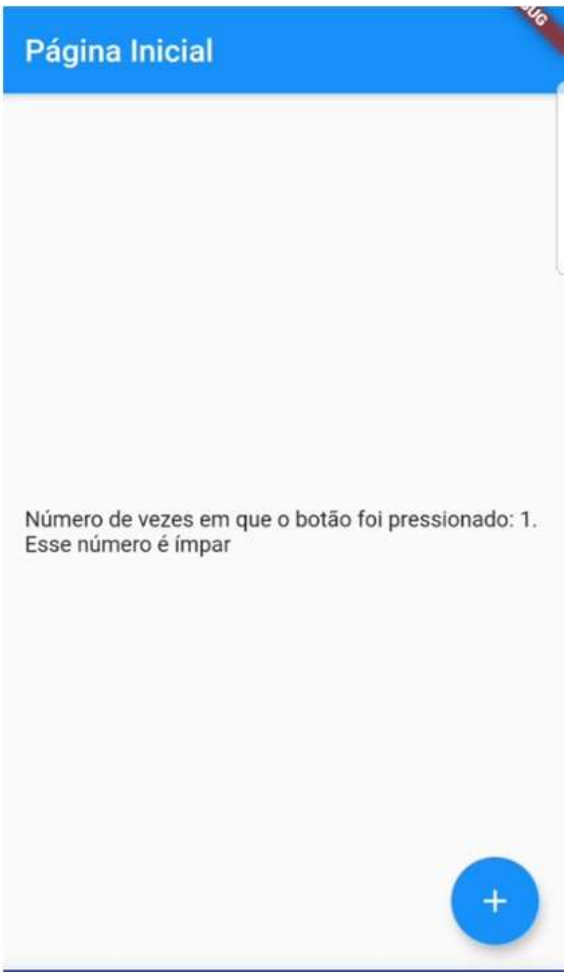
```
        child: const Icon(Icons.add),
      ),
    );
  }
}
```



Exercício

1) Altere o exemplo desta prática, para que ele informe se o número de vezes em que o botão foi pressionado é par ou ímpar.





Dica:

```
  int numeroVezes = 0;
  String mensagem = "Número de vezes em que o botão foi pressionado: 0.\nEsse
número é par";

  void cliqueDoBotao() {
    numeroVezes = numeroVezes + 1;
    mensagem = "Número de vezes em que o botão foi pressionado: $numeroVezes.\
nEsse número é ${numeroVezes % 2 == 0 ? "par" : "ímpar"}";
  }
```

Entrada & Saída

Documentação: https://api.flutter.dev/flutter/material/TextField-class.html, https://api.flutter.dev/flutter/widgets/TextEditingController-class.html, https://api.flutter.dev/flutter/material/ElevatedButton-class.html

Objetivo: apresentar widgets de entrada e saída.

Classe TextField

A classe TextFiled gera um campo de texto do material design. Um campo de texto permite que o usuário insira texto, seja com teclado de hardware ou com teclado virtual na tela.

O campo de texto chama o método de callback onChanged sempre que o usuário altera o texto no campo. Por outro lado, se o usuário indicar que terminou de digitar o texto no campo (por exemplo, pressionando um botão no teclado virtual), o campo de texto chama o método de callback onSubmitted.

Para controlar o texto que é exibido no campo de texto, use o atributo controller. O atributo controller pode receber um objeto da classe TextEditingController; por exemplo, para definir o valor inicial do campo de texto ou ler o texto informado pelo usuário. É uma boa prática descartar um TextEditingController quando ele não for mais necessário. Isso garantirá o descarte de quaisquer recursos usados pelo objeto.

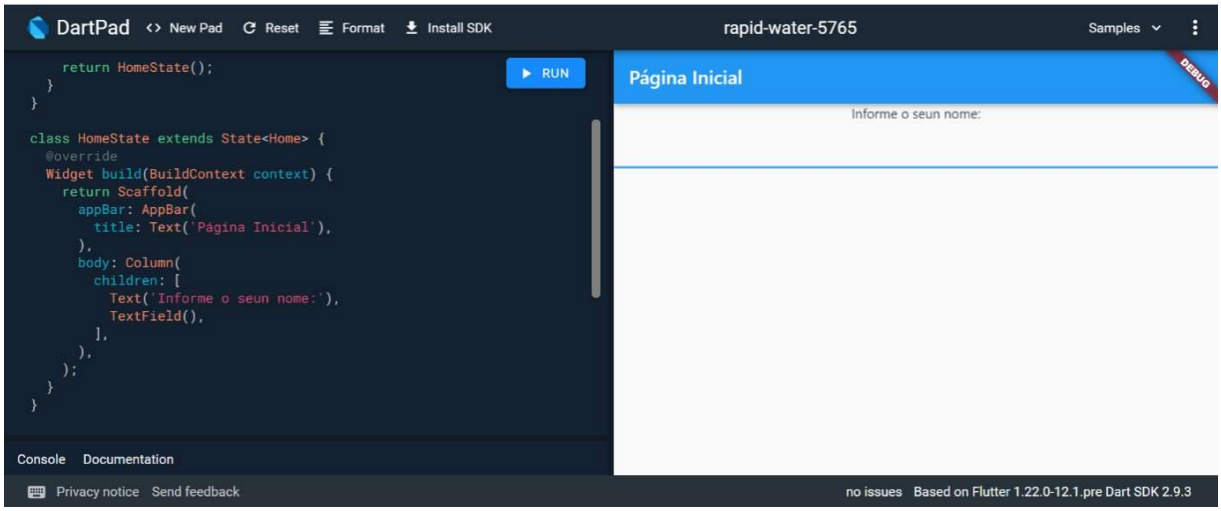
O campo de texto possui um atributo decoration que serve para configurar a aparência do campo. Por padrão, o atributo decoration desenha uma barra divisória abaixo do campo de texto. Entretanto, você pode usar a propriedade decoration para alterar essa aparência padrão; por exemplo, adicionando um rótulo ou um ícone. Se você definir o atributo decoration como null, toda a configuração visual (decoração, layout) do campo de texto será removida. Se o atributo decoration for diferente de null (que é o padrão), o campo de texto exigirá que um de seus ancestrais seja um widget Material.

OBS: para incluir um campo de texto em um formulário (Form) com outros widgets de tipo FormField, use um TextFormField.

3) Exemplo de utilização da classe TextField.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
```

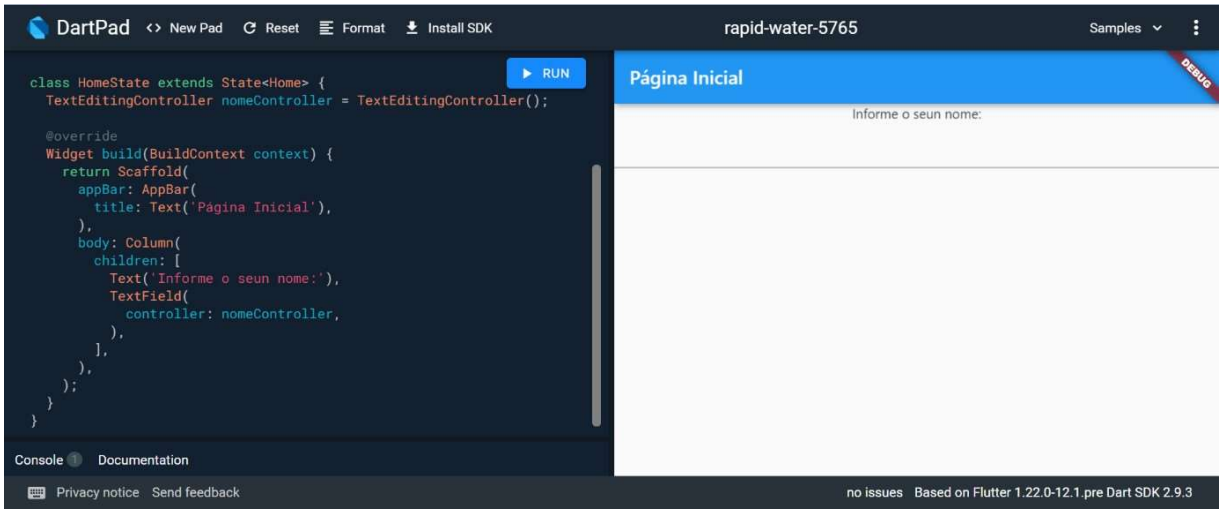
```
    return HomeState();
  }
}

class HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: const [
          Text('Informe o seu nome:'),
          TextField(),
        ],
      ),
    );
  }
}
```



4) Associar um controlador TextEditingController ao atributo controller do campo de texto.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}
```

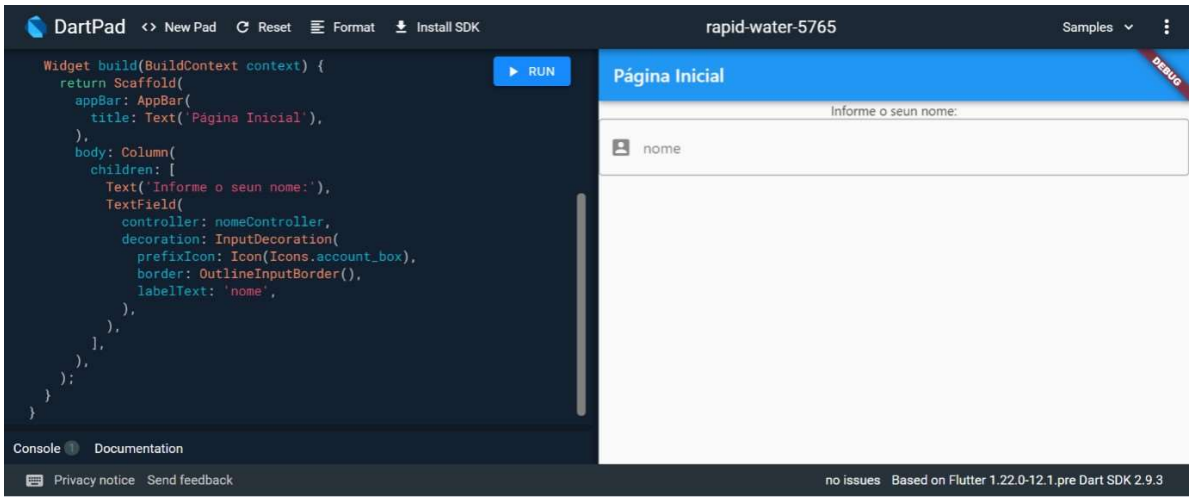
```dart
class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
          ),
        ],
      ),
    );
  }
}
```

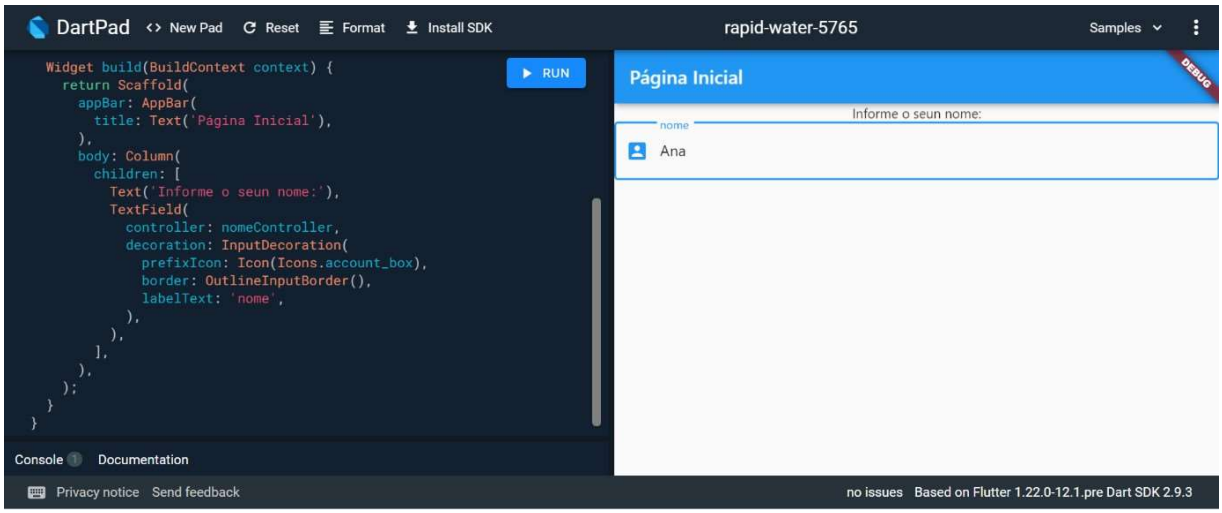


5) Associar uma configuração visual ao atributo decoration. Um ícone é apresentado dentro do campo de texto à esquerda.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
```

```dart
  }
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
            decoration: const InputDecoration(
              prefixIcon: Icon(Icons.account_box),
              border: OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
        ],
      ),
    );
  }
}
```

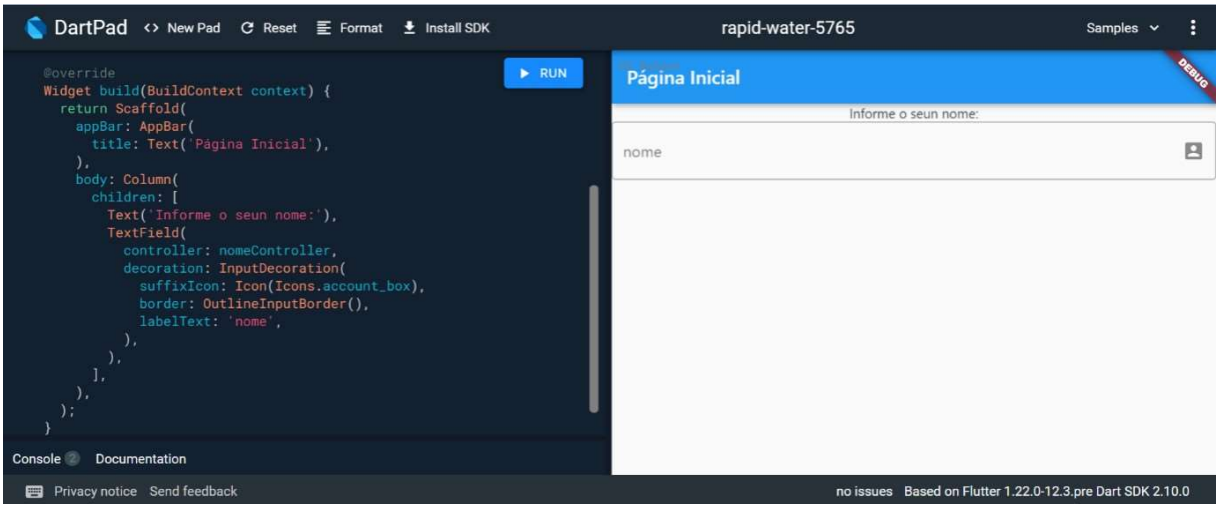


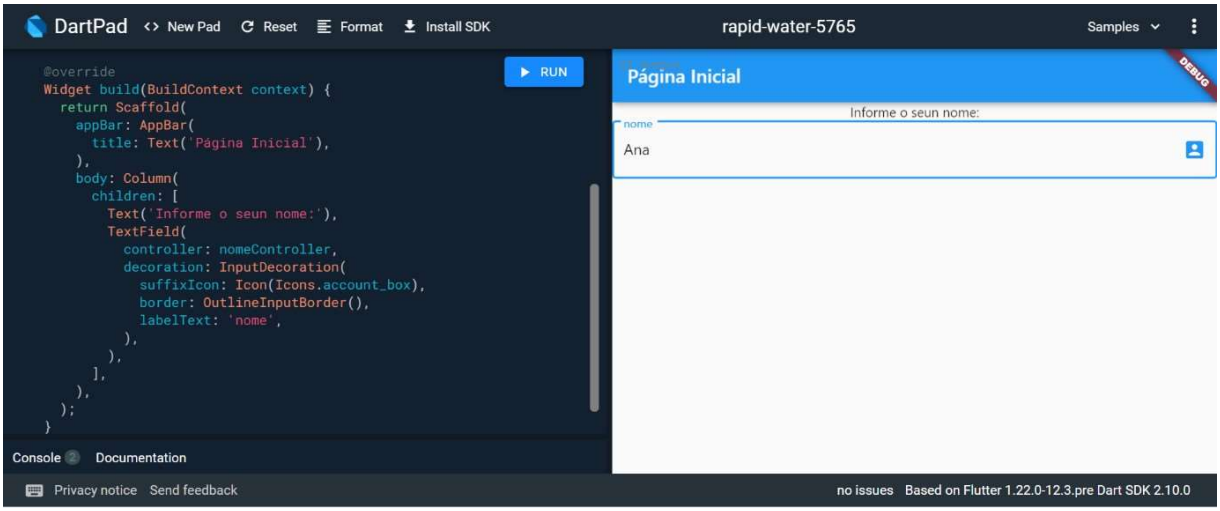


6) Apresentar o ícone dentro do campo de texto à direita.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
            decoration: const InputDecoration(
              suffixIcon: Icon(Icons.account_box),
              border: OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
```

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

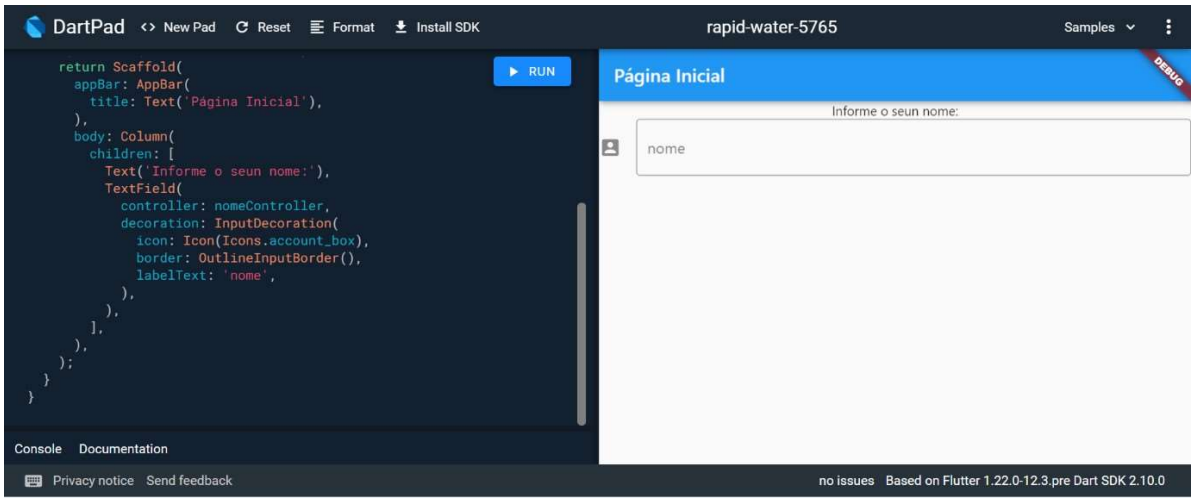


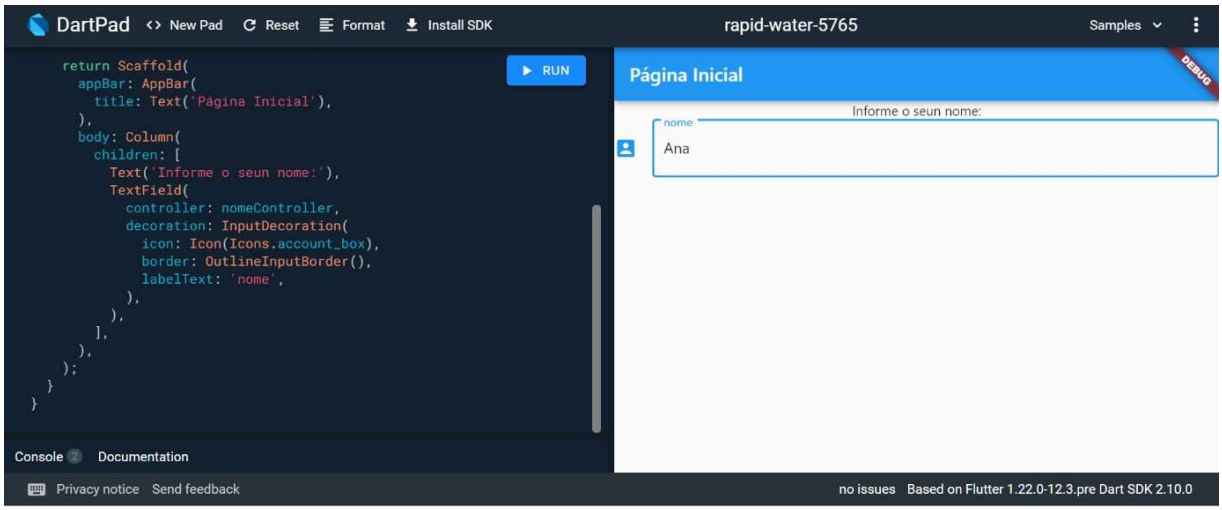


7) Apresentar o ícone fora do campo de texto.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
```

```dart
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
            decoration: const InputDecoration(
              icon: Icon(Icons.account_box),
              border: OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
        ],
      ),
    );
  }
}
```

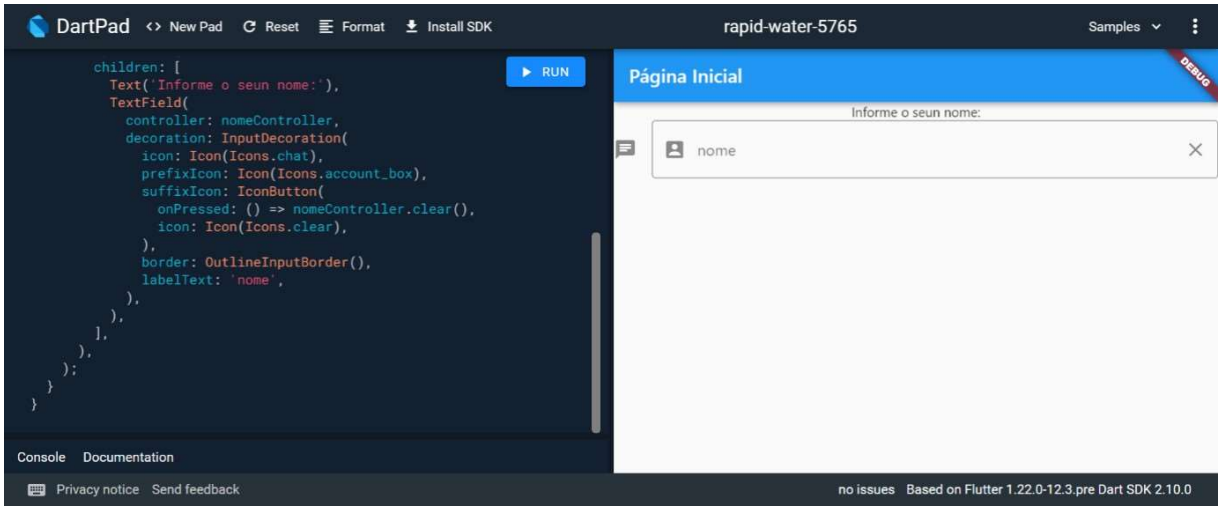


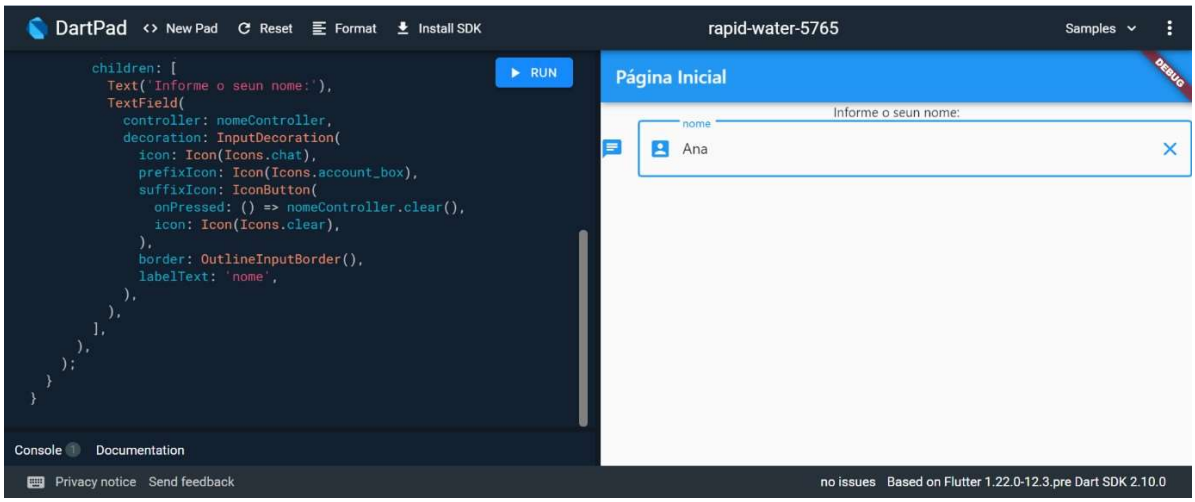


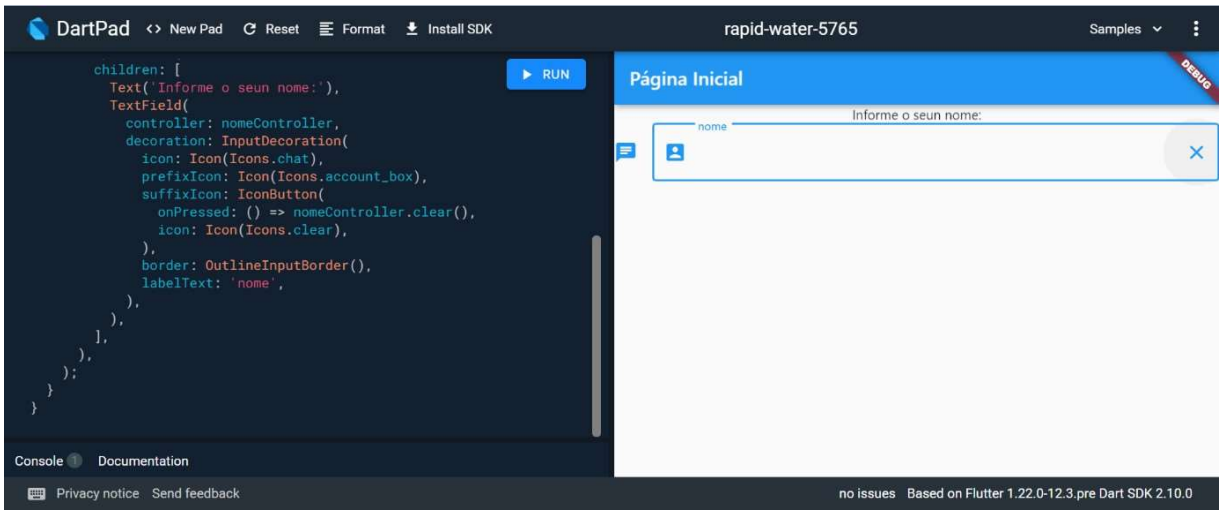
8) Apresentar os três ícones simultaneamente. O ícone da direita recebe uma função que apaga o texto informado no campo de texto quando esse ícone é clicado.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
            decoration: InputDecoration(
              icon: const Icon(Icons.chat),
              prefixIcon: const Icon(Icons.account_box),
              suffixIcon: IconButton(
                icon: const Icon(Icons.clear),
```

```
                onPressed: () => nomeController.clear(),
              ),
              border: const OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
        ],
      ),
    );
  }
}
```





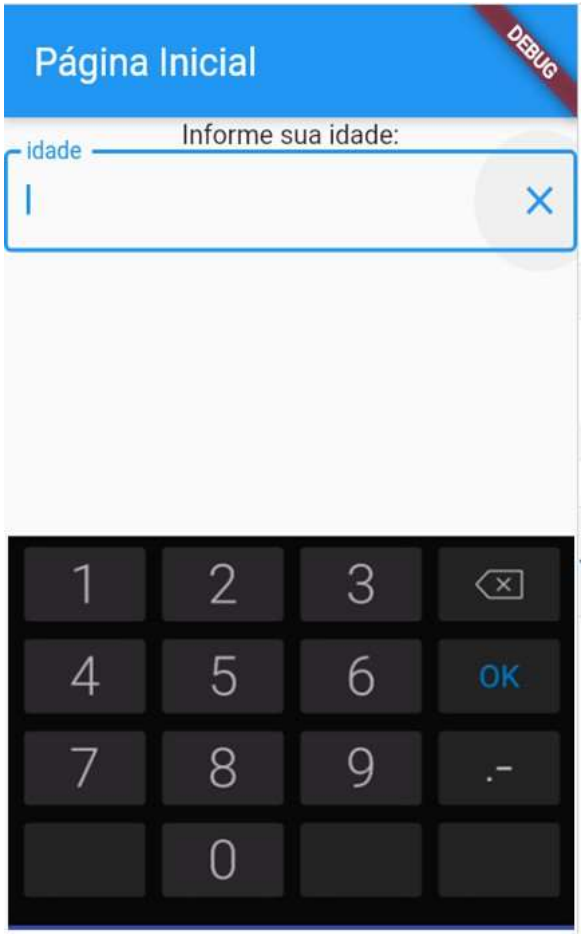


Teclado numérico

9) Criar um teclado numérico:

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController idadeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe sua idade:'),
          TextField(
            controller: idadeController,
            keyboardType: TextInputType.number,
            decoration: InputDecoration(
              suffixIcon: IconButton(
                onPressed: () => idadeController.clear(),
```

```
                icon: const Icon(Icons.clear),
              ),
              border: const OutlineInputBorder(),
              labelText: 'idade',
            ),
          ),
        ],
      ),
    );
  }
}
```



Borda

10) Borda com cantos arredondados:

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

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

class Home extends StatefulWidget {
  @override
```

```
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController idadeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe sua idade:'),
          TextField(
            controller: idadeController,
            keyboardType: TextInputType.number,
            decoration: InputDecoration(
              suffixIcon: IconButton(
                onPressed: () => idadeController.clear(),
                icon: const Icon(Icons.clear),
              ),
              border: const OutlineInputBorder(
                borderRadius: BorderRadius.all(Radius.circular(15)),
              ),
              labelText: 'idade',
            ),
          ),
        ],
      ),
    );
  }
}
```

Classe ElevatedButton

A classe ElevatedButton gera um "botão elevado"", ou "botão em relevo", do Material Design. Você deve usar botões elevados para adicionar dimensão a layouts que são geralmente planos, como listas longas ou em áreas com espaços amplos. Evite usar botões elevados em widgets que também são elevados, como caixas de diálogo ou cartões.

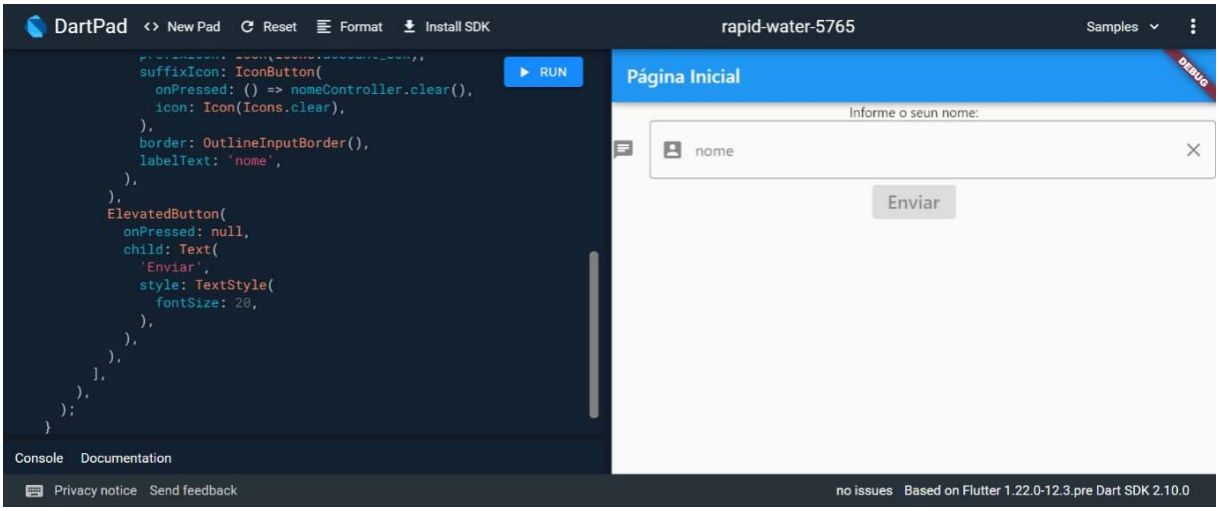
Um botão elevado é baseado em um widget cujo atributo Material.elevation aumenta quando o botão é pressionado. Um botão elevado possui os widgets filhos Text e Icon.

Botões elevados também possuem os atributos onPressed e onLongPress. Esses atributos devem receber as funções que serão executadas quando esses eventos ocorrerem. Por outro lado, se esses atributos receberem o valor null, o botão será desabilitado.

11) Apresentar um botão desabilitado. O atributo onPressed recebe o valor null.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
            decoration: InputDecoration(
              icon: const Icon(Icons.chat),
              prefixIcon: const Icon(Icons.account_box),
              suffixIcon: IconButton(
                onPressed: () => nomeController.clear(),
                icon: const Icon(Icons.clear),
              ),
              border: const OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
          const ElevatedButton(
```

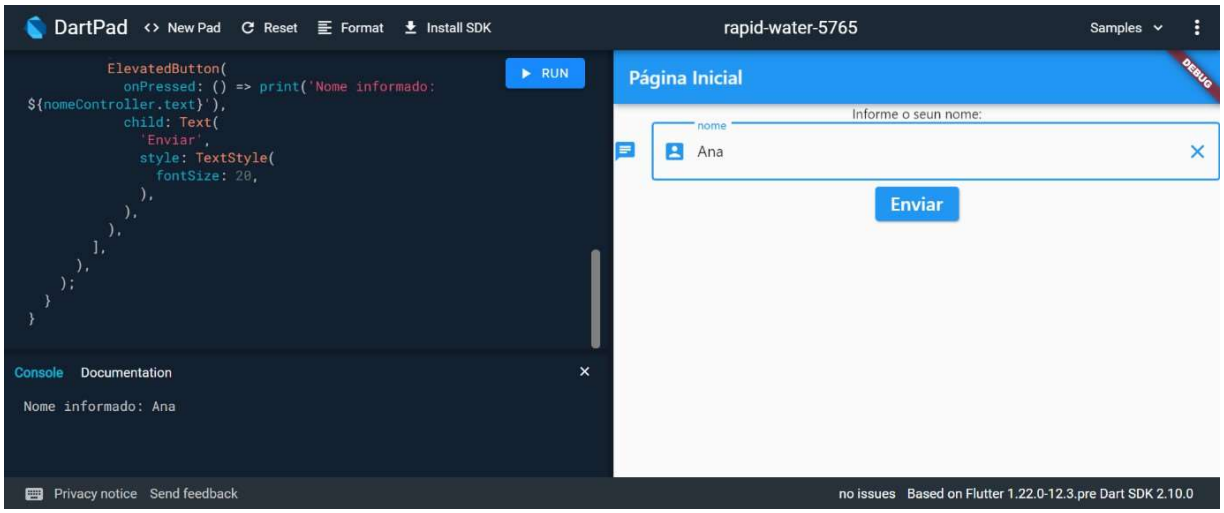
```
            onPressed: null,
            child: Text(
              'Enviar',
              style: TextStyle(
                fontSize: 20,
              ),
            ),
          ),
        ),
      ],
    ),
  );
 }
}
```



12) Apresentar um botão habilitado. O atributo onPressed recebe uma função que imprime no console o nome informado.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();

  @override
```

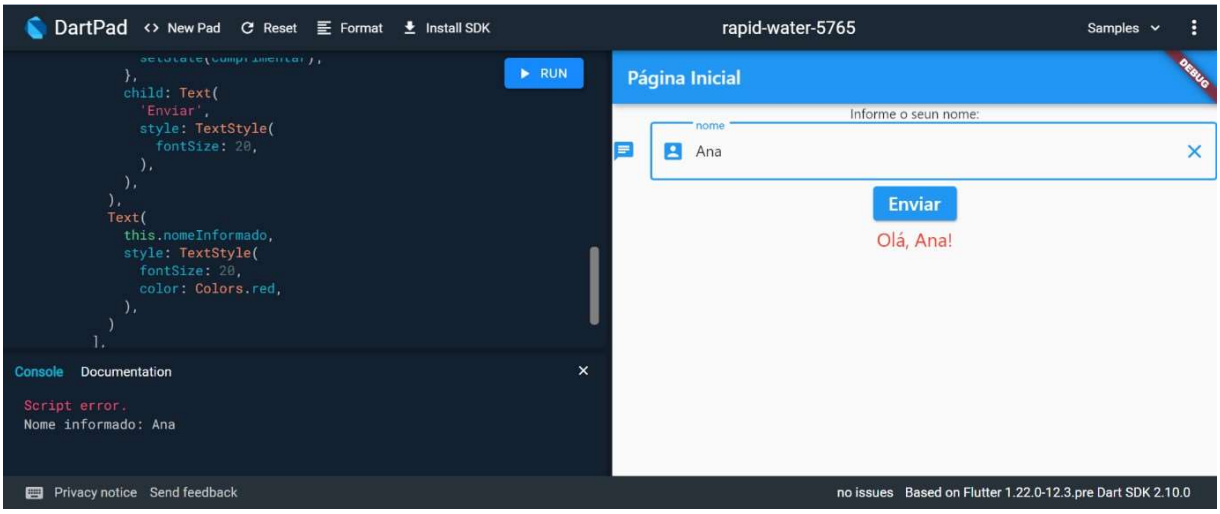
```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
          TextField(
            controller: nomeController,
            decoration: InputDecoration(
              icon: const Icon(Icons.chat),
              prefixIcon: const Icon(Icons.account_box),
              suffixIcon: IconButton(
                onPressed: () => nomeController.clear(),
                icon: const Icon(Icons.clear),
              ),
              border: const OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
          ElevatedButton(
            onPressed: () => print('Nome informado: ${nomeController.text}'),
            child: const Text(
              'Enviar',
              style: TextStyle(
                fontSize: 20,
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```



13) O exemplo abaixo apresenta na tela o nome informado. Como há uma alteração de estado do widget, precisamos usar o método **setState ()** quando o botão é pressionado. No exemplo, o método setState () recebe o método cumprimentar () como parâmetro. O método cumprimentar () gera um string para apresentação na tela.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController nomeController = TextEditingController();
  String mensagem = "";

  cumprimentar() {
    mensagem = 'Olá, ${nomeController.text}!';
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
      ),
      body: Column(
        children: [
          const Text('Informe o seu nome:'),
```

```
          TextField(
            controller: nomeController,
            decoration: InputDecoration(
              icon: const Icon(Icons.chat),
              prefixIcon: const Icon(Icons.account_box),
              suffixIcon: IconButton(
                onPressed: () => nomeController.clear(),
                icon: const Icon(Icons.clear),
              ),
              border: const OutlineInputBorder(),
              labelText: 'nome',
            ),
          ),
          ElevatedButton(
            onPressed: () {
              print('Nome informado: ${nomeController.text}');
              setState(cumprimentar);
            },
            child: const Text(
              'Enviar',
              style: TextStyle(
                fontSize: 20,
              ),
            ),
          ),
          Text(
            mensagem,
            style: const TextStyle(
              fontSize: 20,
              color: Colors.red,
            ),
          )
        ],
      ),
    );
  }
}
```

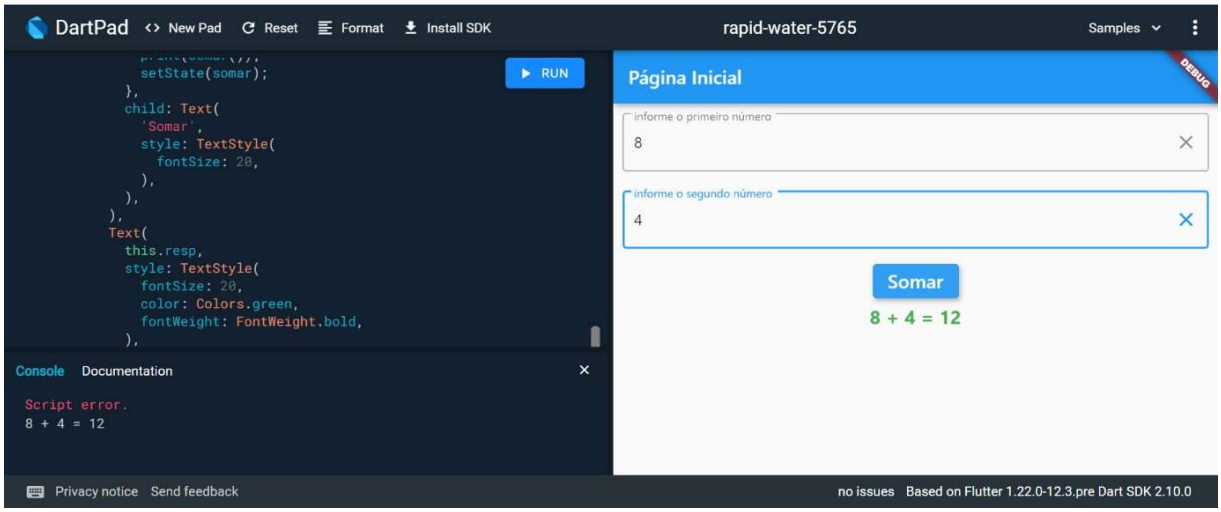


14) No próximo exemplo, o usuário informa dois números inteiros. O aplicativo apresenta a soma dos números na tela. Os números são lidos como strings. Portanto, é preciso convertê-los para inteiro antes de realizar a soma.

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

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

class Home extends StatefulWidget {
  @override
  HomeState createState() {
    return HomeState();
  }
}

class HomeState extends State<Home> {
  TextEditingController num01Controller = TextEditingController();
  TextEditingController num02Controller = TextEditingController();
  String resp = "";

  somar() {
    int num01 = int.parse(num01Controller.text);
    int num02 = int.parse(num02Controller.text);
    int soma = num01 + num02;
    resp = '$num01 + $num02 = $soma';
    return resp;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Página Inicial'),
```

```
      ),
      body: Column(
        children: [
          Container(
            margin: const EdgeInsets.all(10),
            child: TextField(
              controller: num01Controller,
              decoration: InputDecoration(
                suffixIcon: IconButton(
                  onPressed: () => num01Controller.clear(),
                  icon: const Icon(Icons.clear),
                ),
                border: const OutlineInputBorder(),
                labelText: 'informe o primeiro número',
              ),
            ),
          ),
          Container(
            margin: const EdgeInsets.all(10),
            child: TextField(
              controller: num02Controller,
              decoration: InputDecoration(
                suffixIcon: IconButton(
                  onPressed: () => num02Controller.clear(),
                  icon: const Icon(Icons.clear),
                ),
                border: const OutlineInputBorder(),
                labelText: 'informe o segundo número',
              ),
            ),
          ),
          ElevatedButton(
            onPressed: () {
              print(somar());
              setState(somar);
            },
            child: const Text(
              'Somar',
              style: TextStyle(
                fontSize: 20,
              ),
            ),
          ),
          Text(
            resp,
            style: const TextStyle(
              fontSize: 20,
              color: Colors.green,
```

```
              fontWeight: FontWeight.bold,
            ),
          )
        ],
      ),
    );
  }
}
```

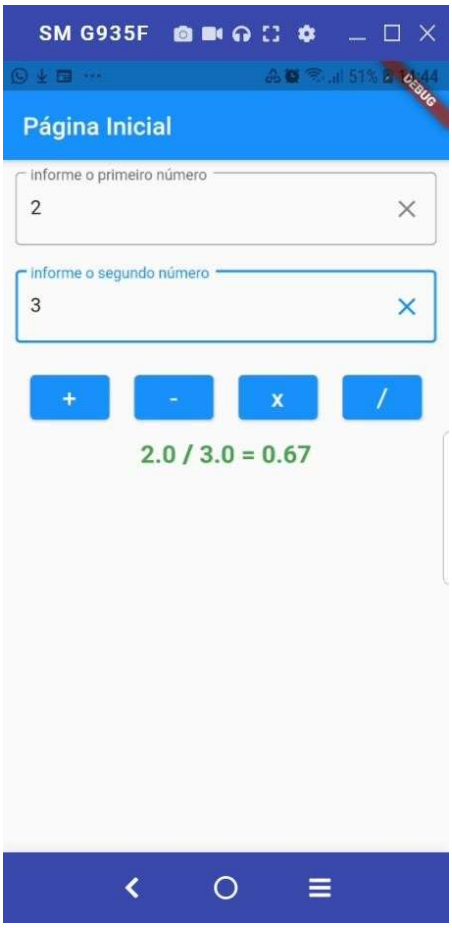


Exercício

2) Usando o framework Flutter, escreva o algoritmo de uma calculadora que permite realizar as 4 operações básicas: somar, subtrair, multiplicar e dividir. O usuário deve informar números reais.

   Dica:

```
dividir() {
  double num01 = double.parse(this.num01Controller.text); //te
  double num02 = double.parse(this.num02Controller.text);
  double result = num01 / num02;
  resp = '$num01 / $num02 = ' + result.toStringAsFixed(2);
  return resp;
}
```



3) Usando o framework Flutter, escreva o algoritmo que leia o peso e altura de uma pessoa e que calcule o índice de massa corporal (IMC) dessa pessoa. O algoritmo também deve apresentar na tela a classificação apresentada na tabela abaixo.

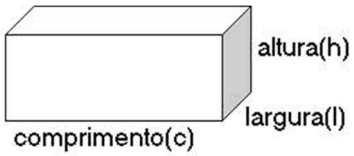
   Dica:
   IMC = peso / (altura × altura)

$$\text{IMC} = \frac{\text{Peso}}{\text{Altura} \times \text{Altura}}$$

IMC	Classificação
< 16	Magreza grave
16 a < 17	Magreza moderada
17 a < 18,5	Magreza leve
18,5 a < 25	Saudável
25 a < 30	Sobrepeso
30 a < 35	Obesidade Grau I
35 a < 40	Obesidade Grau II (severa)
≥ 40	Obesidade Grau III (mórbida)

4) Usando o framework Flutter, escreva o algoritmo que leia o comprimento, a largura e a altura de um paralelepípedo e que retorne o volume desse sólido.

Dica:
volume = comprimento × largura × altura



$V = c \times l \times h$