

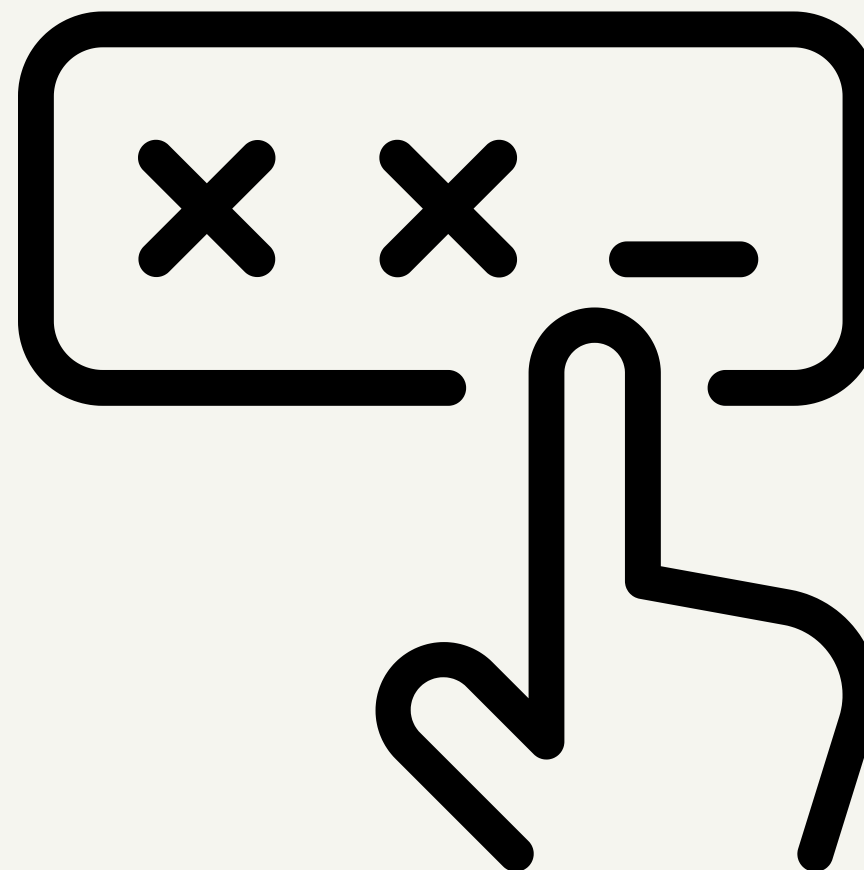


3^o Ano - Técnico em Informática para Internet

Dieimes Nunes de Souza
dieimes.souza@docente.pr.senac.br

CAMPO DE TEXTO

- Os campos de texto possibilita os usuários a interagir com o aplicativo;
- Imagine nos formulários que desenvolvemos em HTML e PHP que continha os campos de texto;



SCAFFOLD

- Scaffold é um widget que fornece uma estrutura visual básica para um aplicativo.
- O Scaffold normalmente envolve outros widgets para construir a interface do usuário do aplicativo.

```
1  import 'package:flutter/material.dart';
2
3  class Pagina lista extends StatelessWidget{
4      Widget build(BuildContext context){
5          return Scaffold(
6              body: TextField(),
7          ); // Scaffold
8      }
9  }
```

SCAFFOLD - PODE DEFINIR DIFERENTES ELEMENTOS, COMO:

- appBar: Uma barra superior que geralmente contém um título e, opcionalmente, ações, como botões.
- body: O conteúdo principal da tela, onde você pode colocar widgets como Container, Column, ListView, etc.
- drawer: Um menu lateral que desliza para a direita ou para a esquerda, comumente usado para opções de navegação adicionais.
- bottomNavigationBar: Uma barra de navegação inferior que permite alternar entre várias telas ou guias.

VAMOS CENTRALIZAR O CAMPO DE TEXTO

Basta inserir o comando Center e depois child: TextField(). Veja a imagem abaixo:

```
3  class Pagina lista extends StatelessWidget{  
4      Widget build(BuildContext context){  
5          return Scaffold(  
6              body: Center(child: TextField(),)  
7          ); // Scaffold  
8      }  
9  }
```

ATENÇÃO

Se as indicações de erros estiver incomodando, então inserir o comando conforme a imagem abaixo no arquivo "analysis_options.yaml".

```
24 rules:
25   prefer_const_constructors: false
26   # avoid_print: false # Uncomment to disable the `avoid_print` ru
27   # prefer_single_quotes: true # Uncomment to enable the `prefer_s
28
```

MARGENS PARA O CAMPO DE TEXTO

- Utilize o comando `Padding` para obter margens na sua caixa de texto. Veja a imagem ao lado:

```
3  class Pagina_lista extends StatelessWidget {  
4      Widget build(BuildContext context) {  
5          return Scaffold(  
6              body: Center(  
7                  child: Padding(  
8                      padding: const EdgeInsets.all(20),  
9                      child: TextField(),  
10                 ), // Padding  
11             ), // Center  
12         ); // Scaffold  
13     }  
14 }
```

MARGENS PARA O CAMPO DE TEXTO

```
8 padding: const EdgeInsets.all(20),
```

A linha `padding: const EdgeInsets.all(20)` é usada para definir o espaçamento interno (padding) de um widget em todas as direções. O `EdgeInsets.all()` é um construtor da classe `EdgeInsets` que cria um objeto de preenchimento com valores iguais em todas as direções.

PERSONALIZAÇÃO

Adicione o comando decoration: InputDecoration dentro do TextField();

O comando decoration: InputDecoration é usado para fornecer a decoração visual de um campo de entrada (input field) em um formulário no Flutter;

```
child: TextField(  
  | decoration: InputDecoration(),  
), // TextField
```

PERSONALIZAÇÃO

- O comando `labelText` é usado para definir o texto exibido como rótulo acima de um campo de texto;
- O comando `hintText` é usado para exibir um texto de dica (placeholder) dentro de um campo de texto (input field) quando ele está vazio.
- O comando `border: OutlineInputBorder()` é usado para definir a aparência da borda de um campo de texto;

Não se preocupe em decorar esses comandos, saiba que eles existem e podem ser encontrados na internet e pelas inteligências artificiais;

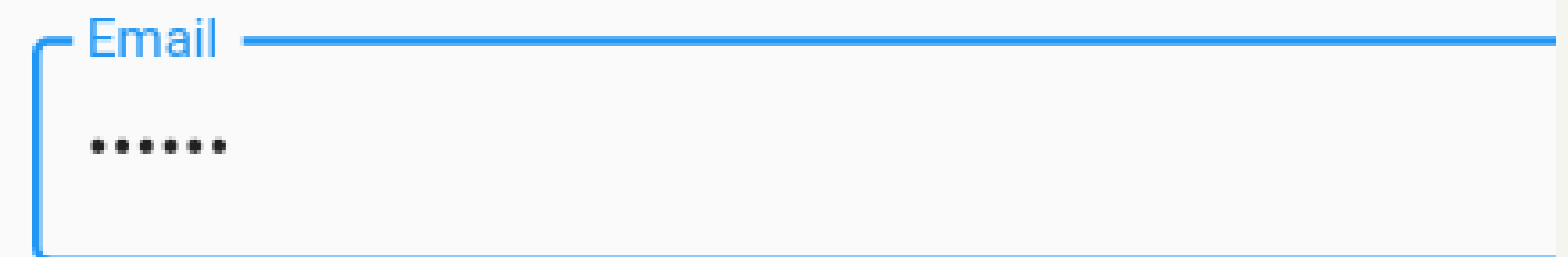
PERSONALIZAÇÃO

```
decoration: InputDecoration(  
  labelText: 'Email',  
  hintText: 'exemplo@exemplo.com',  
  border: OutlineInputBorder(),  
) // InputDecoration
```

PERSONALIZAÇÃO

- O comando **obscureText: true** é usado para ocultar o texto digitado em um campo de entrada.
- O texto digitado no campo de entrada será substituído por pontos ou asteriscos para ocultar o seu conteúdo real. Isso impede que o texto digitado seja facilmente legível por outras pessoas que possam estar olhando para a tela.

```
child: TextField(  
  decoration: InputDecoration(  
    labelText: 'Email',  
    hintText: 'exemplo@exemplo.com',  
    border: OutlineInputBorder(),  
  ), // InputDecoration  
  obscureText: true,  
), // TextField
```




PERSONALIZAÇÃO - COMANDOS

- O comando **obscuringCharacter** permite definir um caractere personalizado que substituirá o texto oculto no campo de entrada quando obscureText estiver definido como true; **(Fora do decoration);**
- O comando **prefixText** é usado para adicionar um texto prefixo a um campo de entrada; **(Dentro do decoration);**
- O comando **suffixText** é usado para adicionar um texto sufixo a um campo de entrada. Esse texto sufixo é exibido **após o campo** de entrada e é útil para adicionar informações adicionais ou contextuais ao campo. **(Dentro do decoration);**

PERSONALIZAÇÃO - COMANDOS

- O comando **keyboardType: TextInputType** é usado para definir o tipo de teclado virtual que será exibido quando o campo de entrada (input field) recebe o foco e o usuário interage com ele no Flutter.
- **TextInputType.text:** Teclado de texto padrão para entrada de texto geral.
- **TextInputType.number:** Teclado numérico para entrada de números.
- **TextInputType.phone:** Teclado numérico com caracteres especiais para entrada de números de telefone.
- **TextInputType.emailAddress:** Teclado para entrada de endereço de e-mail.
- **TextInputType.datetime:** Teclado para entrada de data e hora.
- **TextInputType.url:** Teclado para entrada de URLs.

PERSONALIZAÇÃO - TEXTO

```
style: TextStyle(  
  fontSize: 40,  
  fontWeight: FontWeight.w700, // Negrito ou ...  
  color:  colors.purple,  
), // TextStyle
```

O comando que você mencionou, `style: TextStyle(...)`, é usado para definir o estilo do texto em um widget de texto no Flutter. Ele permite personalizar várias propriedades do texto, como tamanho da fonte, peso da fonte e cor.

PERSONALIZAÇÃO - TEXTO

A propriedade **fontSize** define o tamanho da fonte do texto, em pontos. Por exemplo, `fontSize: 40` define o tamanho da fonte como 40 pontos.

A propriedade **fontWeight** define o peso da fonte do texto. Nesse caso, `FontWeight.w700` define um peso de fonte negrito (700 na escala de peso de fonte). Você também pode usar outros valores de `FontWeight`, como `FontWeight.bold` para obter um peso de fonte ainda mais forte.

A propriedade **color** define a cor do texto. Nesse exemplo, `Colors.purple` define a cor do texto como roxo. Você pode usar qualquer cor definida na classe `Colors` ou especificar uma cor personalizada usando `Color(0xFF123456)`.

PERSONALIZAÇÃO - TEXTO

A **classe TextStyle** fornece várias outras propriedades para personalizar ainda mais o estilo do texto, como `fontFamily` para definir a família da fonte, `letterSpacing` para ajustar o espaçamento entre letras, `wordSpacing` para ajustar o espaçamento entre palavras, entre outros.

Dúvidas e Comentários

Fique à vontade para fazer deste um debate aberto à perguntas e esclarecimentos antes de prosseguirmos.