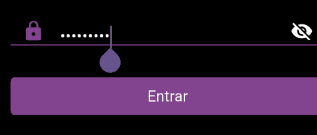
child: TextField(

                      obscureText: true,



Agora vai ficar dessa forma

Lembrar que no ícone adicionamos para ele ficar como se não tivesse mostrando nada, mas quando a gente clicar no botão, queremos deixar ele exibindo a senha.

suffixIcon: Icon(

                          Icons.visibility\_off,

                          color: Colors.white,

                        ),

Para isso vamos declarar uma variável bool

Forma certa de criar a variável é aqui.

class \_MyHomePageState extends State<MyHomePage> {

  String email = "";

  String senha = "";

  bool isObscureText = true;

Aqui toda vez que atualizar a tela ela vai voltar para o estado inicial, no caso senha “” email “” e isObs.. retornando sempre para true.

  @override

  Widget build(BuildContext context) {

    String email = "";

    String senha = "";

    bool isObscureText = true;

Colocamos ela para abrir como true, porque o texto vai abrir escondido já.

child: TextField(

                      obscureText: isObscureText,

Aqui onde estava o true vai receber a variável, que esta como verdadeiro, para o texto ficar escondido.



Agora vamos fazer um ternário mudando isso.

suffixIcon: InkWell(

                          onTap: () {

                            setState(() {

                              isObscureText = !isObscureText;

                            });

                          },

                          child: Icon(

                            isObscureText

                                ? Icons.visibility\_off

                                : Icons.visibility,

                            color: Colors.white,

                          ),

                        ),

**suffixIcon:**  
Esse é um parâmetro do InputDecoration, usado em TextField ou TextFormField. Ele adiciona um ícone no final do campo de texto (lado direito).

🔹 **InkWell(...)**  
Esse widget detecta toques (taps) e fornece um efeito visual de “ripple” (ondinha) ao ser clicado. Ele envolve o ícone e permite que ele seja **interativo**.

🔹 **onTap: () { setState(() { ... }); }**  
Aqui é onde a mágica acontece: quando o ícone é tocado, ele chama setState, informando ao Flutter que o estado mudou e que o widget deve ser reconstruído.

isObscureText = !isObscureText;

Esse trecho **inverte o valor da variável**: se estava true (texto oculto), vira false (texto visível), e vice-versa.

🔹 **child: Icon(...)**  
Esse é o ícone em si. A linha abaixo usa um operador condicional:

isObscureText ? Icons.visibility\_off : Icons.visibility

— Se isObscureText for true, mostra o ícone de olho cortado (visibility\_off), indicando que o texto está oculto.  
— Se for false, mostra o olho aberto (visibility), indicando que o texto está visível.

🔹 **color: Colors.white**  
Define que o ícone será branco, pra combinar com o restante do estilo da tela

## GestureDetector

Outra forma de fazer é usar o GestureDetector

Ele serve para detectar ações do usuário.

Nesse por exemplo vamos ter a opção de segurar o dedo em cima ele fazer algo...

