



Para saber mais: Mensagens de erro

A definição de mensagens de erro para problemas de validação é uma parte importante do desenvolvimento. Isso se dá, pois é a partir delas que conseguimos informar ao cliente exatamente o que falhou no envio dos dados. A boa notícia é que é bem fácil resolver isso através do atributo `message`, como visto em aula. Contudo, há algumas exceções! Existem alguns decorators que recebem informações opcionais ou não nos primeiros parâmetros, como é o caso do `@IsEmail()` que recebe como primeiro parâmetro um objeto JS opcional para configuração específica desse decorator. Sendo assim, para configurar uma mensagem para a validação `@IsEmail()` deveríamos fazer algo como:

```
@IsEmail({}, {  
  message: 'Email inválido!'  
})
```

[COPIAR CÓDIGO](#)

Veja que no primeiro parâmetro é passado um objeto JS vazio representando as configurações opcionais desse validador específico. Nesse caso, não utilizamos nenhuma dessas configurações.

Portanto, segue um bom exemplo da classe `Usuario` com todas as mensagens de erro. Inclusive, o instrutor que vos escreve a utilizará no restante do curso. Então é importante copiá-la para o seu projeto para ter uma paridade com o que será visto nos próximos vídeos.

```
export class Usuario {  
  id: number;  
  
  @IsNotEmpty({  
    message: 'nomeDeUsuario é obriga  
  })  
  @IsString({  
    message: 'nomeDeUsuario precisa  
  })
```

```
nomeDeUsuario: string;

@IsEmail({}, {
  message: 'email precisa ser um e
}))
email: string;

@IsNotEmpty({
  message: 'senha é obrigatório.'
}))
senha: string;

@IsNotEmpty({
  message: 'nomeCompleto é obrigat
}))
nomeCompleto: string;

dataDeEntrada: Date;
}
```

[COPIAR CÓDIGO](#)

Para saber mais sobre os decorators de validação, segue o link da documentação do class-validator : <https://github.com/typestack/class-validator#validation-decorators>

[. \(https://github.com/typestack/class-validator#validation-decorators\)](https://github.com/typestack/class-validator#validation-decorators)