# **O que são Middlewares do express?**

[Sugira uma melhoria](https://cursos.alura.com.br/course/node-js/section/9/exercise/1#)

Neste capítulo, estudamos o funcionamento do express para conseguir carregar dinamicamente todas as bibliotecas que fomos adicionando a aplicação conforme fomos desenvolvendo.

Aprendemos que isso é possível por conta dos Middlewares.

Explique com suas palavras o que você entendeu que sejam os Middlewares do express e como eles funcionam.

## **Resposta do instrutor**

Os Middlewares são funções cujo objetivo é possibilitar a interceptação do request e adicionar verificações e comportamentos sobre eles.

Vários plugins que utilizamos na aplicação junto do express fazem uso dos middlewares. É o caso do bodyparser ou do express-validator, para citar dois exemplos.

# **Onde mais utilizar Middlewares?**

[Sugira uma melhoria](https://cursos.alura.com.br/course/node-js/section/9/exercise/2#)

Utilizamos os Midlewares em várias situações. Principalmente para carregar bibliotecas de terceiros para a aplicação.

Em que outras possíveis situações você imagina que poderíamos utilizar esse conceito.

Discuta suas ideias com a gente!

## **Resposta do instrutor**

Poderíamos utilizar os Middlewares, por exemplo, para adicionar uma funcionalidade própria da nossa aplicação no ciclo de vida do express, como o tratamento de páginas não encontradas.

Se quiser ampliar o debate, abra uma dúvida no fórum! =)

# **Implementando um Middleware**

[Sugira uma melhoria](https://cursos.alura.com.br/course/node-js/section/9/exercise/3#)

Utilizamos Middlewares próprios para implementar o tratamento de páginas de erro e de páginas não encontradas.

Crie esses dois Middlewares na sua aplicação também.

Primeiro crie uma pasta “erros” dentro da pasta “views” e dentro dela crie os arquivos 404.ejs e 500.ejs.

Crie estes dois arquivos simples com suas respectivas mensagens informativas. Depois você pode melhorar suas implementações.

Crie os dois novos Middlewares no express.js

Teste acessar páginas não existentes ou forçar um erro de conexão com o banco para validar a implementação dos Middlewares.

## **Resposta do instrutor**

As páginas 404.ejs e 500.ejs podem ser colocadas dentro da nova pasta app/views/erros. Fica a seu critério pensar como fazer essas views.

O arquivo express.js com os dois novos Middlewares fica assim:

var express = require('express');  
var load = require('express-load');  
var bodyParser = require('body-parser');  
var expressValidator = require('express-validator');  
  
module.exports = function() {  
  
 var app = express();  
 app.use(express.static('./public'));  
 app.set('view engine', 'ejs');  
 app.set('views','./app/views');  
  
 app.use(bodyParser.urlencoded({extended: true}));  
 app.use(bodyParser.json());  
 app.use(expressValidator());  
  
 load('routes',{cwd: 'app',verbose:true})  
 .then('infra')  
 .into(app);  
  
 app.use(function(req, res, next){  
 res.status(404).render("erros/404");  
 });  
  
 app.use(function(error,req, res, next){  
 if(process.env.NODE\_ENV == 'production') {  
 res.status(500).render('errors/500');  
 return;  
 }  
 next(errors);  
 });  
 //tem que colocar na ordem, caso contrário ele passa pelo middleware e   
 //ainda não vai ter acontecido nenhum erro.  
  
 return app;  
};

Lembre que a ordem dos Middlewares importa para que quando um seja carregado, o express já tenha carregado todas as outras dependências das quais ele necessita.

Por isso é que estes dois que acabamos de criar devem aparecer por último na sequência.