
HELLO
WORLD

MIDDLEWARES

E TRATAMENTO DE ERROS

MIDDLEWARES DE ROTEAMENTOS

Middleware é um termo utilizado para descrever funções que têm acesso ao objeto de requisição (req), ao objeto de resposta (res) e à próxima função de middleware na pilha de processamento (next). Os middlewares podem executar código, modificar a requisição e a resposta, encerrar a requisição e chamar a próxima função na pilha.

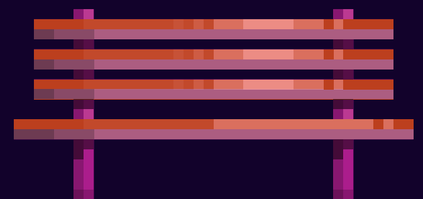
Estrutura Básica de um Middleware

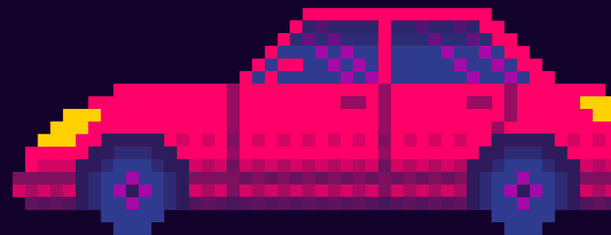
Um middleware típico no Express possui a seguinte estrutura:

```
function myMiddleware(req, res, next) {
  console.log('Middleware executado!');

  next();
}

```





COMO FUNCIONA

- **Recepção da Requisição:** Quando uma requisição é feita ao servidor, o Express passa pela pilha de *middlewares*.
- **Execução da Lógica:** Cada *middleware* pode processar a requisição e a resposta, podendo modificar os objetos.
- **Encaminhamento:** O *middleware* deve chamar a função *next()* para passar o controle para o próximo *middleware* na pilha. Se *next()* não for chamado, a requisição será encerrada, e a resposta não será enviada.

TIPOS DE MIDDLEWARES

01

Middlewares de Aplicação: Associados a uma instância do Express, são usados para gerenciar requisições em toda a aplicação.

```
const express = require('express');
const app = express();

app.use(myMiddleware); // Usando o middleware em toda a aplicação
```

02

Middlewares de Rota: Associados a uma rota específica.

```
app.get('/users', myMiddleware, (req, res) => {
    res.send('Lista de usuários');
});
```

03

Middlewares de Erro: Usados para tratar erros. Devem ter quatro argumentos: *err, req, res, next.*

```
function errorHandler(err, req, res, next) {
    console.error(err);
    res.status(500).send('Ocorreu um erro!');
}
```

MIDDLEWARES MANIPULADORES DE ERROS

Como Funciona?

1. *Captura de Erros*: Um middleware de erro é chamado quando um erro é passado para a função next(). Por exemplo, em uma rota, você pode capturar um erro e passá-lo para o middleware de erro:

```
app.get('/users', (req, res, next) ⇒ {
  try {
      // Lógica que pode causar um erro
      throw new Error('Erro de exemplo');
  } catch (error) {
      next(error); // Passa o erro para o middleware de erro
  }
});
```

2. *Ordem dos Middlewares*: O middleware de erro deve ser definido depois de todas as rotas e middlewares. Isso garante que ele capture os erros que ocorrerem durante o processamento.

3. *Tratamento de Erros*: Dentro do middleware de erro, você pode personalizar a resposta com base no tipo de erro, logar detalhes do erro ou até mesmo enviar notificações.

EXEMPLOS COMUNS DE MIDDLEWARES DE ERRO

01

Middleware de Erro Genérico
Um middleware simples que captura erros e envia uma resposta genérica

```
function genericErrorHandler(err, req, res, next) {
  console.error(err); // Log do erro no console
  res.status(500).json({ error: 'Algo deu errado!' }); // Resposta JSON
}
```

02

Middleware de Erro Personalizado
Um middleware que fornece mensagens de erro mais específicas com base no tipo de erro

```
function customErrorHandler(err, req, res, next) {
  if (err.isJoi) { // Exemplo usando Joi para validação
      return res.status(400).json({ error: err.details[0].message });
  }

  res.status(err.status || 500).json({
      message: err.message || 'Ocorreu um erro interno no servidor.',
  });
}
```

03

Middleware de Erro para Validação
Se você estiver usando bibliotecas como Joi ou express-validator para validação de dados, você pode criar um middleware que trata especificamente de erros de validação:

```
function validateUser(req, res, next) {
    const schema = Joi.object({
        username: Joi.string().min(3).required(),
        email: Joi.string().email().required(),
    });

    const { error } = schema.validate(req.body);
    if (error) {
        return next(error); // Passa o erro para o middleware de erro
    }
    next(); // Validação bem-sucedida
}
```


THANK
YOU