**SERVIÇO NACIONAL DE APRENDIZAGEM INDUSTRIAL**

**SENAI/JOINVILLE NORTE**

**ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**



**FICHAMENTO : NODEJS**

MATEUS BRUNO DE ALECRIN

**JOINVILLE**

**Mateus Bruno de Alecrin**

**Fichamento : NodeJs**

Fichamento apresentado ao Sistema Nacional de Aprendizagem Industrial (SENAI) Campus NORTE / Joinville, como parte das exigências para a obtenção do título de tecnólogo em Análise e Desenvolvimento de Sistemas.

Professores: Silvio de Souza e Carlos Uchoa.

**JOINVILLE**

**2022**

# 1 NODEJS

## 1.1 O que é Node.js e por que é popular no desenvolvimento web?

“Assim nasce o Node.js, um ambiente de tempo de execução JavaScript, de código aberto, que permite a criação de servidores web eficientes” (ROCKETSEAT, 2023).

A simplicidade do código, somado à eficiência e escalabilidade oferecida pela arquitetura baseada em eventos do Node.js gerou um grande entusiasmo na comunidade dev. As primeiras versões do Node.js já estavam equipadas com recursos como a API Stream, que permitia o processamento eficiente de grandes volumes de dados, e a API Buffer, para manipulação de dados binários, uma necessidade para muitos servidores. [...]. Atualmente, Node.js é uma ferramenta vital para desenvolvedores web. Não só pela sua capacidade de executar JavaScript no servidor, juntamente com seu modelo de I/O não bloqueante, que o torna ideal para a criação de aplicações web de alto desempenho e altamente escaláveis, mas também pelas possibilidades de integração com inteligências artificiais, característica indispensável no cenário atual (ROCKETSEAT, 2023).

## 1.2 Qual é a diferença entre o Node.js e outras tecnologias de servidor, como o Apache?

Foi criado em 2009 pelo engenheiro de software Ryan Dahl como uma alternativa ao Apache HTTP Server, o servidor web mais popular da época. Dahl criticava algumas limitações do Apache, como o suporte relativamente baixo a múltiplas conexões, buscando resolver este problema com uma abordagem focada em eventos e execução assíncrona, sendo uma opção mais responsiva e com menos consumo de recursos do que os servidores “tradicionais” (MELO, 2020).

A principal característica do Node.js é sua execução ser single-thread, ou seja, os recursos computacionais são alocados apenas uma vez pelo tempo que a aplicação estiver sendo executada. Aplicações multi-thread, como as criadas com PHP, por exemplo, requerem a criação de uma thread a cada nova requisição, e ela não é executada enquanto a anterior não for finalizada (MELO, 2020).

## 1.3 Como você inicia um projeto Node.js usando o npm?

“Se você já tiver o Node e o npm e deseja começar a construir seu projeto, execute o comando npm init. Com isso, você dará procedimento a inicialização do seu projeto.” (L, 2023).

## 1.4 O que é o Express.js e qual é o seu papel no desenvolvimento web com Node.js?

“O Express.js é um Framework rápido e um dos mais utilizados em conjunto com o Node.js, facilitando no desenvolvimento de aplicações back-end e até, em conjunto com sistemas de templates, aplicações full-stack.” (ANDRADE, 2020).

O Express é um framework incrível e possui diversas características que facilitam o desenvolvimento de nossas aplicações. Dentre suas principais características, podemos citar: Possui um sistema de rotas completo; Possibilita o tratamento de exceções dentro da aplicação; Permite a integração de vários sistemas de templates que facilitam a criação de páginas web para suas aplicações; Gerencia diferentes requisições HTTP com seus mais diversos verbos; Feito para a criação rápida de aplicações utilizando um conjunto pequeno de arquivos e pastas (ANDRADE, 2020).

## 1.5 Explique o conceito de middleware no contexto do Express.js.

Para ficar mais claro, vou associar o Express à brincadeira de telefone sem fio. Sim, isso mesmo, mas o que uma coisa tem a ver com a outra? Bom o Express é aquele que passa a primeira informação e nela contém tudo que é relacionado ao serviços Web e a seus protocolos e métodos. Vindo a primeira informação ela passa por uma fila, e assim como na brincadeira, o middleware pode alterar, adicionar ou apagar informações. É importante dizer que no meio do caminho pode-se “acabar” com a brincadeira, como é o caso do res.render, res.send ou res.end que emitem os cabeçalhos de respostas (protocolo HTTP) [...]. (RIBEIRO, 2018).

Figura 01 - Fluxo do middleware no Express.js



Fonte: Ribeiro (2020).

## 1.6 Como você roteia solicitações HTTP em um aplicativo Express?

O Roteamento refere-se à definição de terminais do aplicativo (URIs) e como eles respondem às solicitações do cliente. Para obter uma introdução a roteamento, consulte Roteamento básico. O código a seguir é um exemplo de uma rota muito básica.

| **var** **express =** **require('express');**  **var** **app =** **express();**  // respond with "hello world" when a GET request is made to the homepage  **app.get('/',** **function(req, res) {**  **res.send('hello world');**  **});** |
| --- |

(EXPRESSJS, 2023).

## 1.7 O que é o middleware de análise de corpo (body-parser) e por que é útil em um aplicativo Express?

“Body Parser NodeJS em si não sabe converter os dados da requisição para o formato que queremos, o body-parser é um middleware capaz de converter o body da requisição para vários formatos. Um desses formatos é o json, exatamente o que queremos.” (LIMA, 2019).

## 1.8 Quais são os principais métodos HTTP e como eles são usados em rotas Express?

Um método de roteamento é derivado a partir de um dos métodos HTTP, e é anexado a uma instância da classe express. o código a seguir é um exemplo de rotas para a raiz do aplicativo que estão definidas para os métodos GET e POST.

| // GET method route  **app.get('/',** **function** **(req, res) {**  **res.send('GET request to the homepage');**  **});**  // POST method route  **app.post('/',** **function** **(req, res) {**  **res.send('POST request to the homepage');**  **});** |
| --- |

O Express suporta os seguintes métodos de roteamento que correspondem aos métodos HTTP: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, propfind, proppatch, unlock, report, mkactivity, checkout, merge, m-search, notify, subscribe, unsubscribe, patch, search, e connect (EXPRESSJS, 2023).

## 1.9 Como você lida com erros em um aplicativo Express?

Uma forma de fazer o gerenciamento de erros de forma mais profissional é centralizando-o através de um error middleware. O Express é todo organizado através de middlewares de processamento e existe um middleware especial que chamamos de Error Middleware que serve como destino default para todos os erros gerados e não tratados no event loop. Enquanto que um middleware normal possui 3 parâmetros (req, res e next), o Error Middleware é criado passando um parâmetro a mais, que na verdade deve ser o primeiro: error. Além disso, ele deve ser o último middleware na sua cadeia de processamento.

| **app.get('/teste3',** **(req, res, next)** **=>{**  **throw new Error('teste3 deu erro');**  **});**  **app.use((error, req, res, next)** **=>** **{**  **console.log('error middleware');**  **res.sendStatus(500);**  **});** |
| --- |

No exemplo acima, criei o error middleware logo após a última rota. Experimente subir novamente sua aplicação e testar a rota 3. Verá que apesar de ela não possuir try/catch, seu fluxo será redirecionado para o error middleware automaticamente quando o erro estourar, simplificando bastante o tratamento de erros pois agora ele estará centralizado (DUARTE, 2021).

## 1.10 O que é uma API RESTful e como o Express pode ser usado para criar uma?

Para o nosso artigo, como o objetivo é mostrar como criar uma API Restful, vamos criar um web service Restful que expõe informações de produtos como : nome do produto, descrição, preço, estoque e se o produto está ativo. Com base nessa premissa, vamos utilizar REST para desenvolvermos no protocolo HTTP e vamos criar um serviço que permita consultar, atualizar, inserir e excluir informações de produtos. Para isso, vamos implementar um serviço usando os verbos HTTP: GET , PUT, POST, DELETE e PATCH. Para alcançar o nosso objetivo vamos definir as tarefas que iremos realizar em nosso projeto:

* Definir o modelo que representa o produto as as informações que iremos expor;
* Definir um banco de dados para persistir as informações;
* Definir as rotas usadas em nossa aplicação para definir a localização do recurso;
* Definir o tratamento das requisições e respostas usando o protocolo HTTP;

Dessa forma vamos iniciar criando em nosso projeto 3 pastas :

* models - onde vamos definir o modelo;
* routes - onde vamos definir as rotas;
* controllers - onde vamos definir o tratamento das requisições e respostas (MACORATTI, 2023).

**REFERÊNCIAS BIBLIOGRÁFICAS**

ANDRADE, Ana Paula de. **O que é o Express.js?**: características do express.js. 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-o-express-js>. Acesso em: 08 out. 2023.

ANDRADE, Ana Paula de. **O que é o Express.js?**: de volta ao express.js. 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-o-express-js>. Acesso em: 08 out. 2023.

DUARTE, Luiz. **Error Handling em Node.js com Express**: error middleware. 2021. Disponível em: <https://www.luiztools.com.br/post/error-handling-em-node-js-com-express/>. Acesso em: 08 out. 2023.

EXPRESSJS. **Roteamento**. 2023. Disponível em: <https://expressjs.com/pt-br/guide/routing.html>. Acesso em: 08 out. 2023.

EXPRESSJS. **Roteamento**: métodos de roteamento. 2023. Disponível em: <https://expressjs.com/pt-br/guide/routing.html>. Acesso em: 08 out. 2023.

L, Andrei. **O Que É npm? Introdução Básica para Iniciantes**: começar um projeto com npm. 2023. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-npm#:~:text=Se%20voc%C3%AA%20j%C3%A1%20tiver%20o,a%C3%A0%20inicializa%C3%A7%C3%A3o%20do%20seu%20projeto.&text=Esse%20comando%20funciona%20como%20uma%20ferramenta%20para%20criar%20o%20arquivo%20package>. Acesso em: 08 out. 2023.

LIMA, Tiago. **Criando uma API RESTful com NodeJS e Express — Inicializando o projeto e o método GET**: body parser. Body Parser. 2019. Disponível em: <https://medium.com/xp-inc/https-medium-com-tiago-jlima-developer-criando-uma-api-restful-com-nodejs-e-express-9cc1a2c9d4d8>. Acesso em: 08 out. 2023.

MACORATTI, José Carlos. **Node - Criando uma API Restful com Express acessando o Mongodb - III**: objetivo e estrutura do projeto. 2023. Disponível em: <https://macoratti.net/17/04/node\_restmgdb3.htm>. Acesso em: 08 out. 2023.

MELO, Diego. **O que é Node.js? [Guia para iniciantes]**: como funciona?. 2020. Disponível em: <https://tecnoblog.net/responde/o-que-e-node-js-guia-para-iniciantes/>. Acesso em: 08 out. 2023.

MELO, Diego. **O que é Node.js? [Guia para iniciantes]**: o que é node.js?. 2020. Disponível em: <https://tecnoblog.net/responde/o-que-e-node-js-guia-para-iniciantes/>. Acesso em: 08 out. 2023.

RIBEIRO, Higo. **Iniciando com middlewares no Express.js**: começando com middlewares. 2018. Disponível em: <https://blog.rocketseat.com.br/middlewares-no-express-js/>. Acesso em: 08 out. 2023.

ROCKETSEAT. **Node.js e a revolução do desenvolvimento web**. 2023. Disponível em: <https://blog.rocketseat.com.br/node-js-e-a-revolucao-do-desenvolvimento-web/>. Acesso em: 08 out. 2023.