



# **BACK-END**

# **NODE.JS**



# VAMOS USAR:

QUE

NODE.JS - FASTIFY - PRISMA - TYPESCRIPT - MONGODB

# INTRODUÇÃO AO BANCO DE DADOS

- **O que é um banco de dados?** Um banco de dados é basicamente um lugar onde você armazena informações para poder acessá-las e manipulá-las depois. Por exemplo, imagine uma planilha do Excel que guarda informações de clientes ou produtos – é como um banco de dados básico.
- **O que é o MongoDB?** MongoDB é um tipo de banco de dados NoSQL. Ao invés de usar tabelas e linhas (como os bancos de dados tradicionais), ele usa documentos em formato JSON (algo bem parecido com objetos no JavaScript) para armazenar informações. Isso facilita o trabalho com dados que não têm uma estrutura fixa.

# O QUE É NODE.JS?

- **Node.js** é uma plataforma que permite rodar JavaScript no servidor, fora do navegador. Antes do Node.js, o JavaScript era usado apenas no lado do cliente (navegador). Com o Node.js, você pode criar servidores web, trabalhar com bancos de dados, lidar com arquivos e muito mais — tudo usando a mesma linguagem que a gente já conhece no navegador, o JavaScript.
- **Exemplo simples:** Pense no Node.js como um garçom em um restaurante. Ele recebe pedidos (requisições) dos clientes (usuários), vai até a cozinha (banco de dados) buscar as informações, e entrega de volta ao cliente (navegador) o que foi pedido.

# O QUE É O FASTIFY?

- **O Fastify** é um framework que ajuda a gente a criar servidores e APIs de forma rápida e eficiente. Ele é como uma caixa de ferramentas que facilita a criação de rotas, tratamento de requisições, e envio de respostas. A grande vantagem do Fastify é que ele é muito rápido e otimizado, ideal para criar aplicativos de alto desempenho.
- **Comparação:** Se o Node.js é o garçom, o Fastify é o bloco de anotações que organiza melhor os pedidos, para que tudo funcione de forma mais rápida e organizada.

# O QUE É PRISMA?

- **Prisma** é uma ferramenta para interagir com o banco de dados de forma fácil e intuitiva. Ele gera um "ORM" (Object-Relational Mapping), que basicamente permite a gente escrever schemas para manipular o banco de dados, sem precisar escrever queries SQL complexas. Ele cria uma ponte entre o nosso código e o banco de dados MongoDB.
- **Exemplo prático:** Se o MongoDB é a cozinha do restaurante, o Prisma é o chefe de cozinha que traduz os pedidos do garçom (Node.js) em pratos de comida (dados no banco). O Prisma faz esse meio de campo e garante que os dados certos sejam manipulados.

# O QUE É TYPESCRIPT?

- **TypeScript** é uma versão melhorada do JavaScript, que adiciona "tipos". Ou seja, com ele, você pode dizer exatamente que tipo de dado uma variável deve ter, o que ajuda a evitar erros. É muito útil em projetos grandes, pois aumenta a segurança do código e ajuda a identificar erros antes mesmo do código rodar.
- **Exemplo simples:** É como se você estivesse construindo algo com peças de LEGO, e o TypeScript te avisa quando você está tentando encaixar uma peça que não se ajusta ao resto.

# COMO TUDO FUNCIONA JUNTO?

- **Node.js** é a plataforma onde tudo roda, permitindo a criação de servidores.
- **Fastify** é o framework que ajuda a construir as rotas e interações entre o usuário e o servidor de forma eficiente.
- **Prisma** faz a conexão entre o código e o banco de dados MongoDB, ajudando a manipular os dados sem se preocupar muito com queries complexas.
- **TypeScript** adiciona segurança ao código, garantindo que os dados e funções estejam corretos antes mesmo de rodar.

# INSTALANDO O NODE.JS

1

Acesse  
[nodejs.org](https://nodejs.org).

2

Baixe a  
versão LTS  
recomendada

3

Instale  
seguindo as  
instruções do  
instalador.

4

Verifique a  
instalação no  
terminal com:



node -v  
npm -v

# CRIANDO UM PROJETO NODE.JS

- Abra o terminal.
- Navegue para a pasta onde quer criar o projeto.
- Execute o comando:



```
mkdir meu-projeto  
cd meu-projeto  
npm init -y
```

- Isso cria o arquivo package.json.



# INSTALAÇÃO E INICIACÃO TYPESCRIPT

- Vá no terminal



```
npm install typescript --save-dev  
tsc -init
```

- Isso cria o arquivo tsconfig.json, onde você pode configurar várias opções do TypeScript



# INSTALAÇÃO DO MONGODB

- Vá no terminal



```
npm install mongodb
```

- instala o driver oficial do MongoDB para Node.js no seu projeto.



# CONFIGURANDO O SERVIDOR COM FASTIFY

- Vá no terminal



```
npm install fastify @fastify/cors
```

- instala o framework Fastify e o plugin CORS para Fastify no seu projeto.



# TYPESCRIPT NO AMBIENTE NODE.JS.

- Vá no terminal



```
npm install tsx --save-dev
```

- instala o pacote tsx como uma dependência de desenvolvimento no seu projeto.



# INSTALAÇÃO DO PRISMA CLIENT

- Vá no terminal



```
npm install @prisma/client  
npx prisma init
```

- Instala o Prisma Client, permitindo interagir com o banco de dados no código.
- Prepara o Prisma, criando arquivos de configuração, e cria o arquivo .env



Prisma

# CONFIGURAÇÃO DO MONGODB

- **Passo 1:** Crie uma conta gratuita no MongoDB (escolha na região de São Paulo).
- **Passo 2:** Configure o acesso ao MongoDB na aba "Network Access", permitindo o acesso de qualquer IP.

# CRIAÇÃO DA CONTA NO MONGODB

**Crie uma conta**

Veja tudo o que o Atlas pode oferecer gratuitamente

**Entrar com o Google**

**Nome\***

**Connect to Clients**

1. Set up connection security

2. Choose a connection method

3. Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

1. Add a connection IP address

Your current IP address (189.89.215.220) has been added to enable local connectivity. Only an IP address you add to your Access List will be able to connect to your project's clusters. Add more later in [Network Access](#).

2. Create a database user

This first user will have `atlasAdmin` permissions for this project.

We autogenerated a username and password. You can use this or create your own.

You'll need your database user's credentials in the next step. Copy the database user password.

**Username**: testeaula2025    **Password**: aula123456789    **Copy**

**Create Database User**

**Choose a connection method**

**Atlas**

Welcome to Atlas. Let's build something great.

Help us tailor your experience by taking a minute to answer the questions below.

GETTING TO KNOW YOU

What is your primary goal?

Select

How long have you been developing software with MongoDB?

Select

GETTING TO KNOW YOUR PROJECT

What programming language are you primarily building on MongoDB with?

Select

What type(s) of data will your project use?

You can choose as many as you want

**Configurations**

Name: Clientes

Provider: AWS

Region: São Paulo (sa-east-1)

Tag (optional):

Quick setup

Automate security setup

Preload sample dataset

**M10** \$0.12/hour

Dedicated cluster for development environments and low-traffic applications.

Storage: 10 GB   RAM: 2 GB   vCPU: 2 vCPUs

**Serverless** \$0.10/1M reads

For application development and testing, or workloads with variable traffic.

Storage: Up to 1TB   RAM: Auto-scale   vCPU: Auto-scale

**M0** Free

For learning and exploring MongoDB in a cloud environment.

Storage: 512 MB   RAM: Shared   vCPU: Shared

**Configurations**

Name: Cluster0

Quick setup

Automate security setup

Preload sample dataset

I'll do this later

Go to Advanced Configuration   Create Deployment

# ORGANIZAÇÃO DA ESTRUTURA DO PROJETO

- **Passo 1:** Crie a pasta src, onde ficarão os arquivos da API.
- **Passo 2:** Dentro da pasta src, crie os arquivos:
  - **server.ts** (onde vamos configurar o servidor)
  - **routes.ts** (onde vamos configurar as rotas da API)

# IMPLEMENTAÇÃO DAS FUNCIONALIDADES DA API

- **Passo 1:** Crie as seguintes pastas dentro de src:
  - services (para manipular os dados e interagir com o banco de dados)
  - controllers (onde as requisições da API serão tratadas)
- **Passo 2:** Dentro de services, crie o arquivo CreateCustomerService.ts.
- **Passo 3:** Dentro de controllers, crie o arquivo CreateCustomerController.ts.
- **O que é** o Service logica da aplicação
- **O que é** o Controller controla o que eu preciso mandar para o usuário visualizar e cadastrar"

# CONCLUSÃO

**Resumo:** O projeto foi configurado com Node.js, Prisma e MongoDB. Criamos rotas, serviços e controladores para gerenciar os dados de clientes, além de implementar o tratamento de erros.



# OBRIDO!



JOÃO VICTOR NAZARETH DE SOUZA