**Curso de Node.JS**

**Felipe MX**

https://www.youtube.com/watch?v=EGVPrfgl5jw&list=PLSx5HT-nMTzxfYej6YiT5WrWFXBa5CuT0&index=1

# Aula 01 - Conceitos básicos

## Conceitos de Backend

### API - (Application Programming Interface)

Uma definição fácil de entender seria: mecanismo para conectar sistema de softwares entre si. Ou seja, se compõe de um conjunto de instruções e regras que as aplicações podem seguir para se comunicarem.

### Backend

É a parte de um sistema que se encarrega de toda a lógica da aplicação. Não é diretamente acessível pelo usuário. Entre algumas das operações que realiza uma delas é atender as requisições dos recursos, assim como acessar o servidor de base de dados.

Algumas linguagens mais empregadas para programação de backend são: PHP, Ruby, C#, Java, Python, JavaScript, entre outras.

### API REST (REpresentational State Transfer)

(transferência de estado representacional)

É uma arquitetura que utiliza o protocolo http para realizar a execução de operações sobre os dados, opera sob a arquitetura cliente-servidor

Em outras palavras: permite o acesso e manipulação de recursos (dados, conjunto de dados) localizados em algum servidor web; as operações mais importantes em um sistema REST são: POST (criar), GET (ler/consultar), PUT (atualizar) e DELETE (apagar).

Para executar estas operações não se requer uma interface de usuário, a manipulação se faz através de uma URI.

### URI (Uniform Resource Identifier)

Identificador de recursos uniforme, ou seja, o caminho ou rota onde se localiza o recurso no sistema REST.

### O que faremos?

Desenvolveremos um backend básico para a manipulação de dados:

* Clientes
* Produtos
* Pedidos

### O que utizaremos?

NodeJS: ambiente de execução para JavaScript do lado do servidor, ou seja, com JavaScript programaremos o backend.

<https://nodejs.org>

npm: Sistema de gerenciamento de pacotes para NodeJS (é instalado ao se instalar o ambiente NodeJS)

MongoDB: motor/gerenciamento de base de dados não relacional, armazena os dados em documentos (coleções de objetos, documentos tipo JSON).

<https://www.mongodb.com>

Express: Servidor web ágil para NodeJS; através do qual montaremos nossa API REST.

### Requisitos antes de começar

* NodeJS (descarregue e instale a versão atual)
* MongoDB (descarregue e instale a versão atual de MongoDB Community Server)
* Compass: aplicação cliente para MongoDB
* Postman API Client: aplicação para interagir com API REST (pode se utilizar uma aplicação similar para provar uma API REST)
* Um IDE ou editor de código de sua preferência

md felipemx

cd felipemx

md store-api

cd store-api

## Inicializando o projeto Node.JS

npm init



code .



**package.json**

{

"name": "store-api",

"version": "1.0.0",

"description": "api rest",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [

"api"

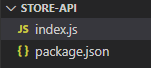
],

"author": "Roberto Pinheiro",

"license": "ISC"

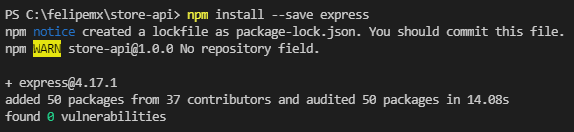
}

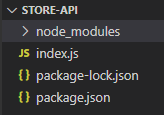
- Na pasta raiz do projeto adicione o arquivo index.js:



## Instalando o Express

npm install --save express





**package.json**

{

"name": "store-api",

"version": "1.0.0",

"description": "api rest",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [

"api"

],

"author": "Roberto Pinheiro",

"license": "ISC",

"dependencies": {

"express": "^4.17.1"

}

}

**index.js**

const express = require('express');

const app = express();

app.get('/', (req, res) => {

res.send('Ola mundo NodeJS!');

});

app.listen(5000, () => {

console.log('Servidor web Express em execução!');

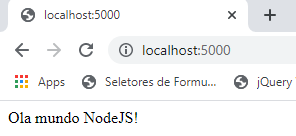
})

node index.js



- No browser:

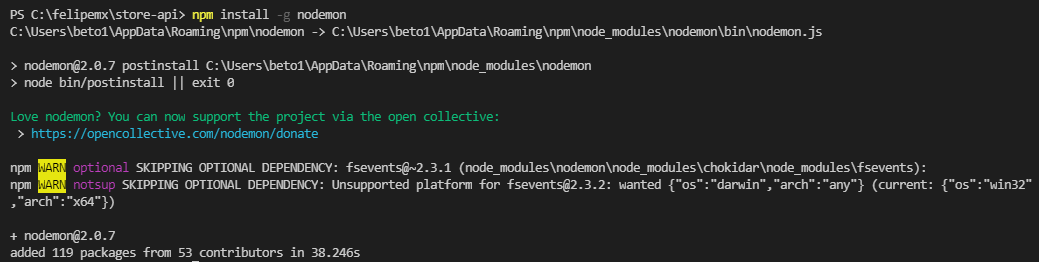
localhost:5000



## Instalando nodemon

npm install -g nodemon

- Instala o nodemon de forma global.



**package.json**

{

"name": "store-api",

"version": "1.0.0",

"description": "api rest",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "nodemon index.js"

},

"keywords": [

"api"

],

"author": "Roberto Pinheiro",

"license": "ISC",

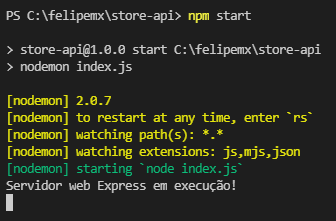
"dependencies": {

"express": "^4.17.1"

}

}

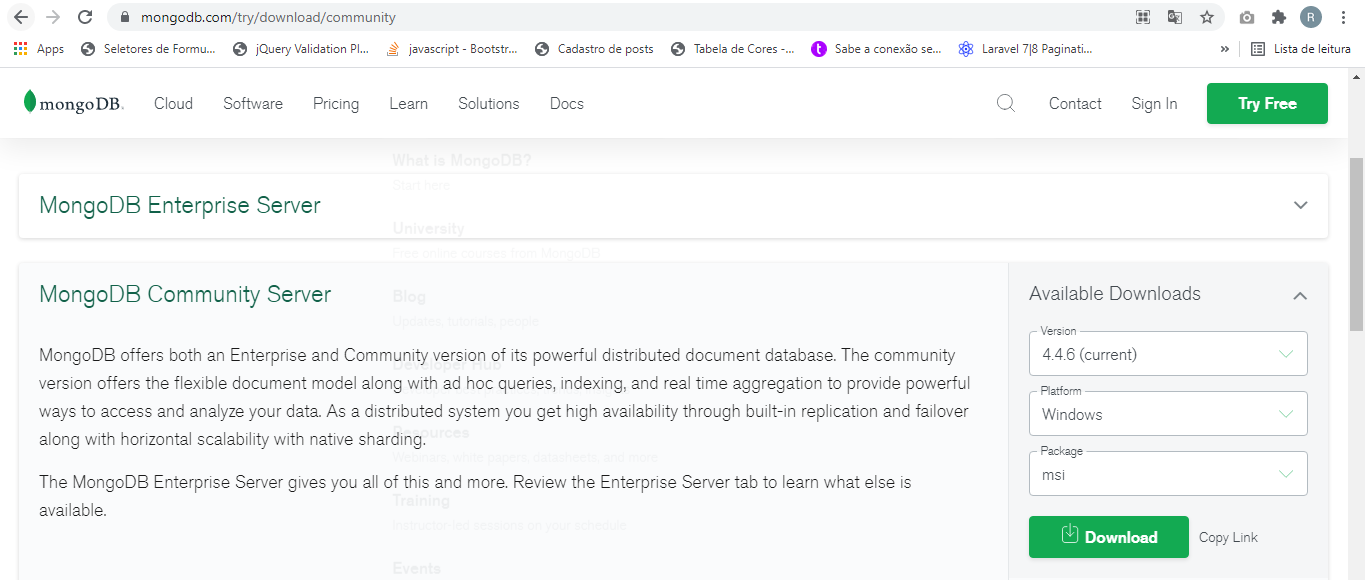
npm start

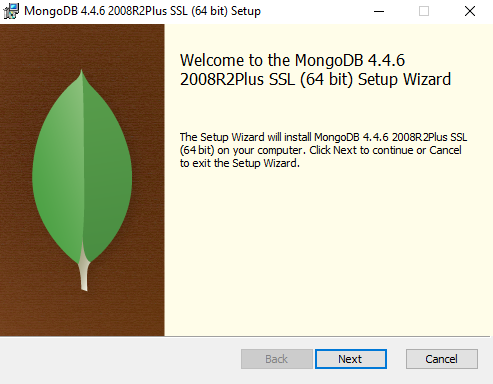


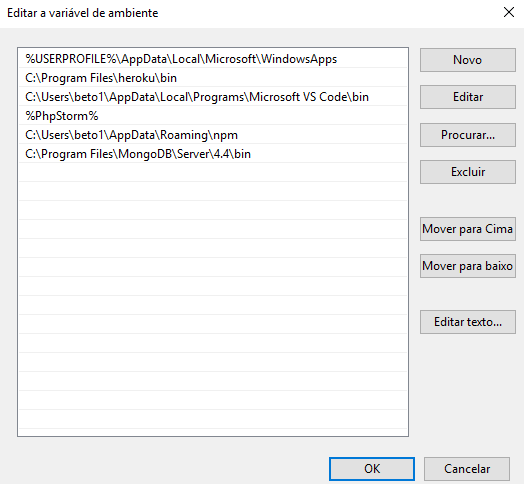
# Aula 02 - Iniciar MongoDB

- Baixe e instale o MongoDB

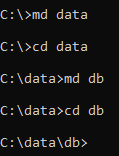
https://www.mongodb.com/try/download/community







- Dentro de C:\ crie a pasta data e dentro dela a subpasta db:

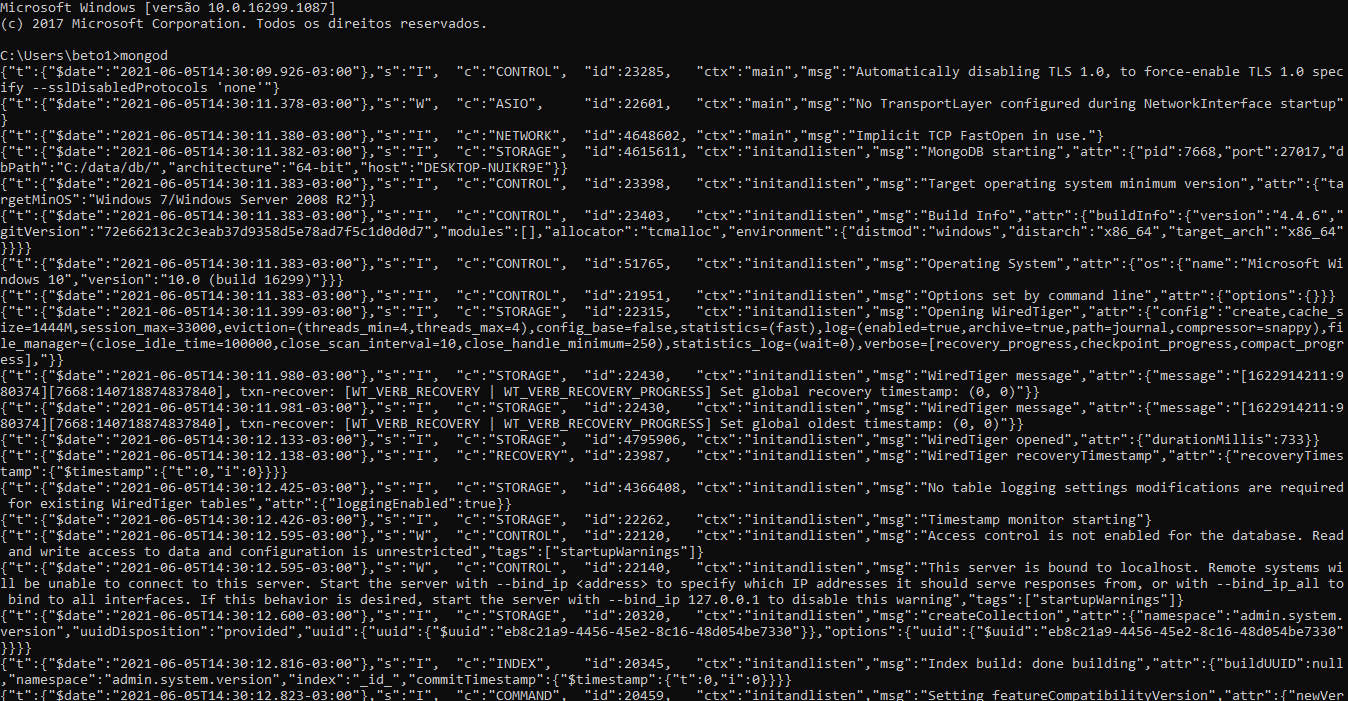


- No Linux - Ubuntu, dê permissão de escrita para C:/data/db:

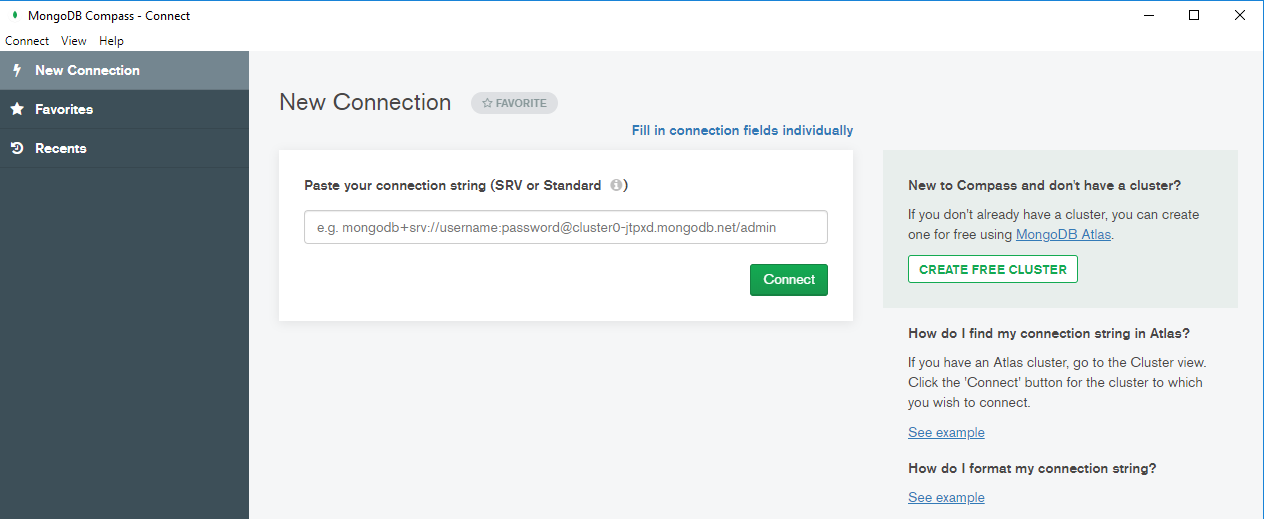
sudo chmod 777 /data/db

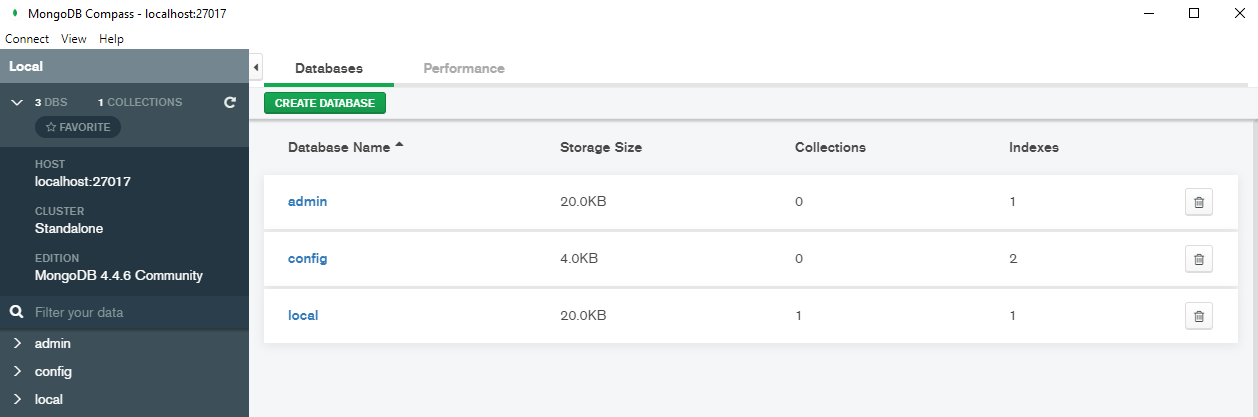
Executando o servidor do MongoDB

mongod



- Vamos utilizar Mongo Compact para conectar ao servidor.

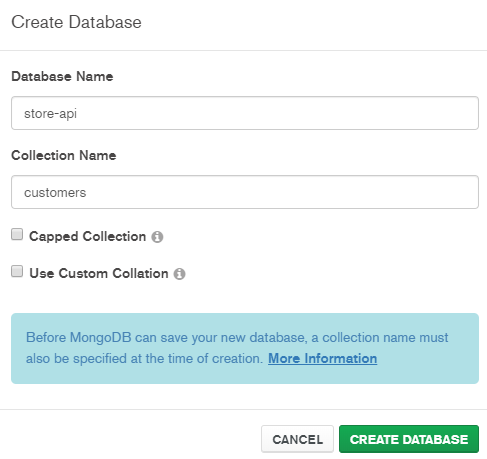




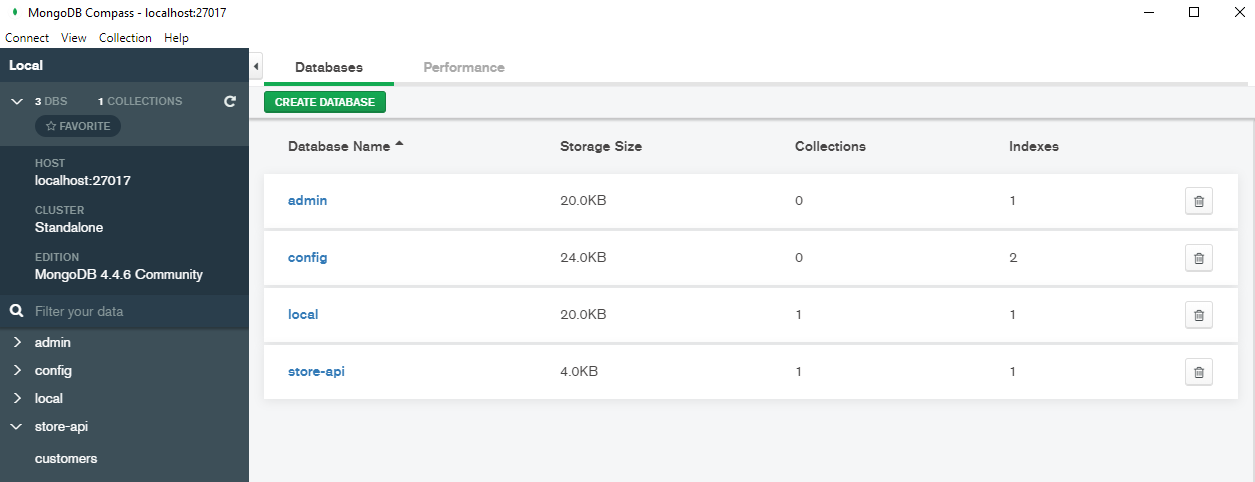
Clique no botão "CREATE DATABASE":

Database Name: store-api

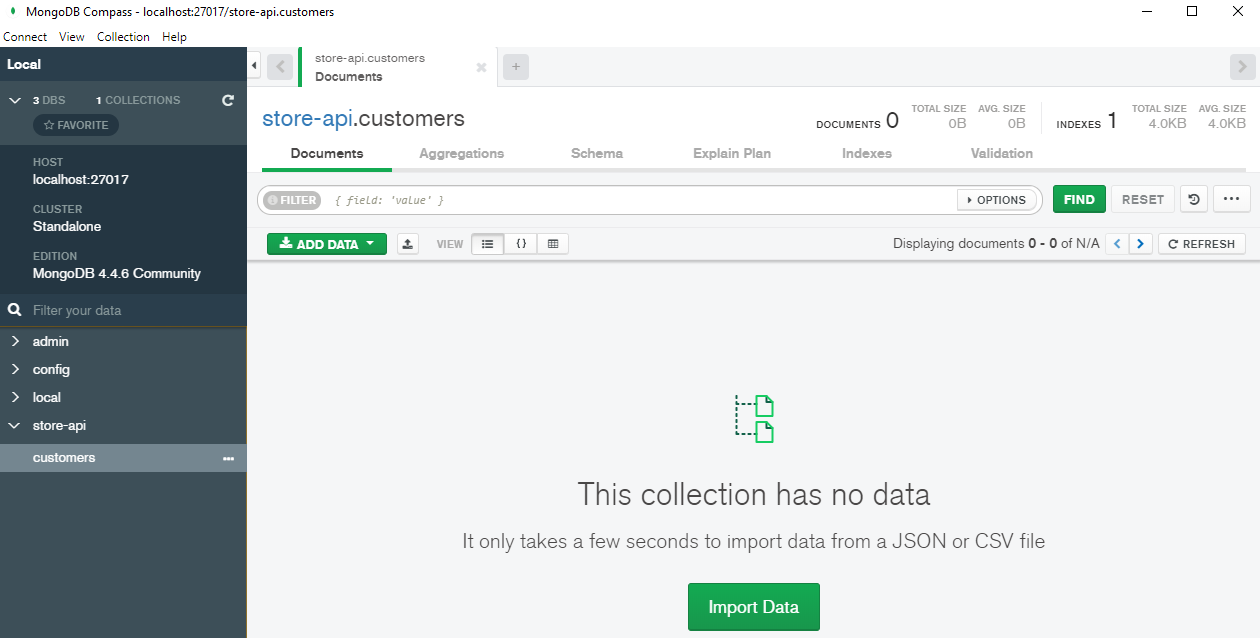
Collection Name: customers



- Clique no botão "CREATE DATABASE"



- Clique em "customers":



- Clique no botão "ADD DATA -> Insert Document":

"name": "Roberto",

"lastname": "Pinheiro",

"email": "roberto\_pinheiro@gmail.com",

"phone": "(11) 99905-3268"



- Clique no botão "INSERT"



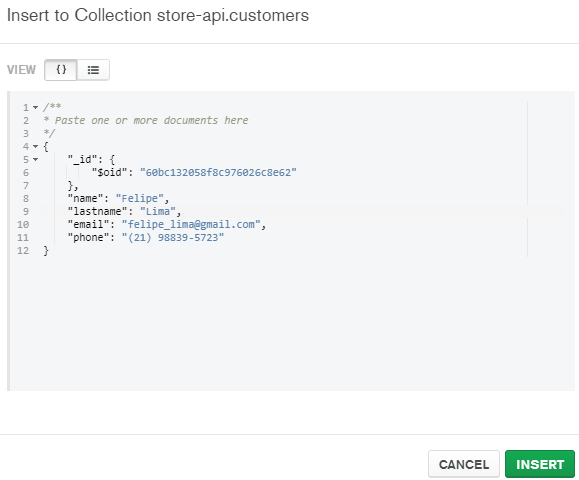
- Insira um novo documento:

"name": "Felipe"

"lastname": "Lima"

"email": "felipe\_lima@gmail.com"

"phone": "(21) 98839-5723"

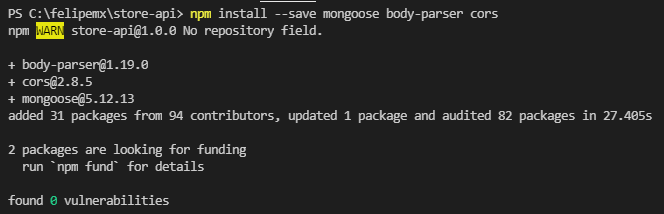




# Aula 03 - Conectar NodeJS a MongoDB Server

## Instalando as dependências Mongoose Body-Parser e CORs

npm install --save mongoose body-parser cors



**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

// Configurando mongoose

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/store-api', {

useNewUrlParser: true

});

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.get('/', (req, res) =>{

res.send('Ola mundo NodeJS!');

});

app.listen(5000, () => {

console.log('Servidor web Express em execução!');

})

- Na pasta raiz do projeto, adicione uma pasta chamada "models" e dentro dela adicione um arquivo chamado "Customer.js".

**models\Customer.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const customersSchema = new Schema({

name: {

type: String,

trim: true

},

lastname: {

type: String,

trime: true

},

email: {

type: String,

trim: true,

unique: true,

lowercase: true

},

phone: {

type: String,

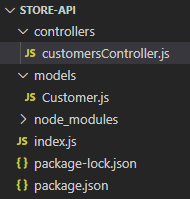
trim: true,

}

});

module.exports = mongoose.model('Customer', customer.Schema);

- Na pasta raiz do projeto, adicione uma pasta chamada "controllers" e dentro dela adicione um arquivo chamado "customersController.js".



**controllers\customersController.js**

const Customer = require('../models/Customer');

// primeira ação: list

exports.index = async (req, res) => {

try {

const customers = await Customer.find({});

res.json(customers);

} catch (error) {

console.log(error);

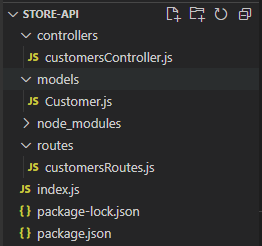
res.send(error);

next();

}

};

- Na pasta raiz do projeto, adicione uma pasta chamada "routes" e dentro dela adicione um arquivo chamado "customersRoutes.js".



**routes\customersRoutes.js**

const express = require('express');

const router = express.Router();

const customersController = require('../controllers/customersController');

module.exports = function() {

// get: /customers

router.get('/customers', customersController.index);

return router;

}

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const customersRoutes = require('./routes/customersRoutes');

const app = express();

// Configurando mongoose

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/store-api',

{

useNewUrlParser: true

});

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

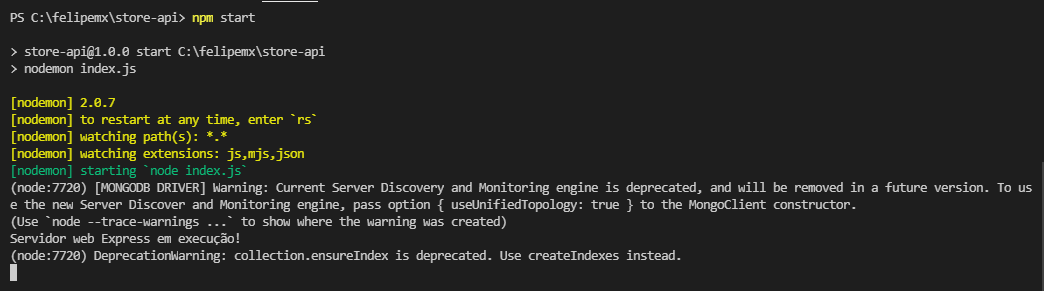
app.use('/', customersRoutes());

app.listen(5000, () => {

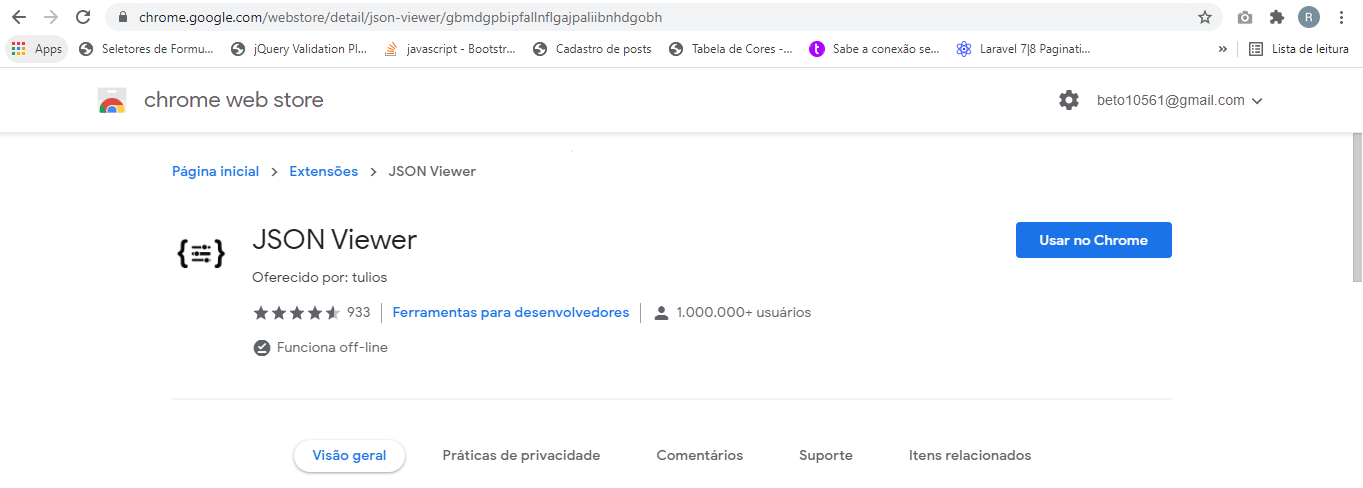
console.log('Servidor web Express em execução!');

})

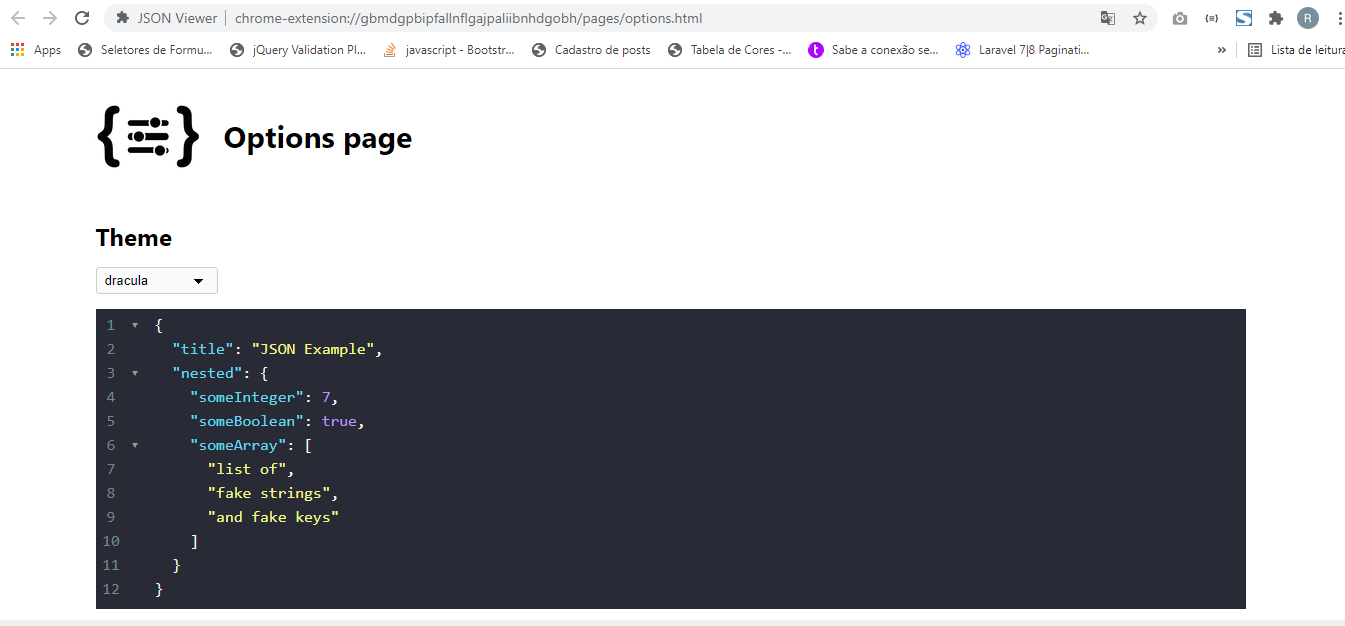
npm start



- Instale a extensão JSON Viewer no Google Chrome:

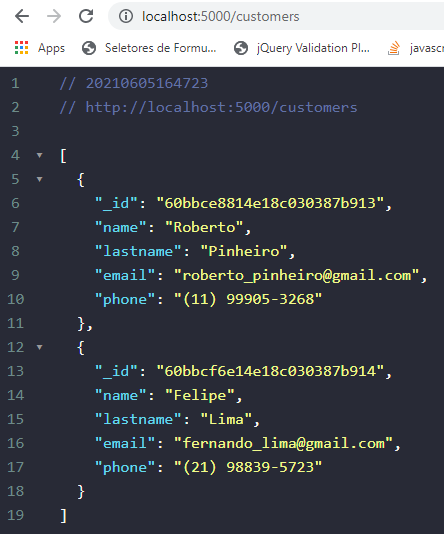


- Como tema, escolha: "dracula":



- No browser, entre com:

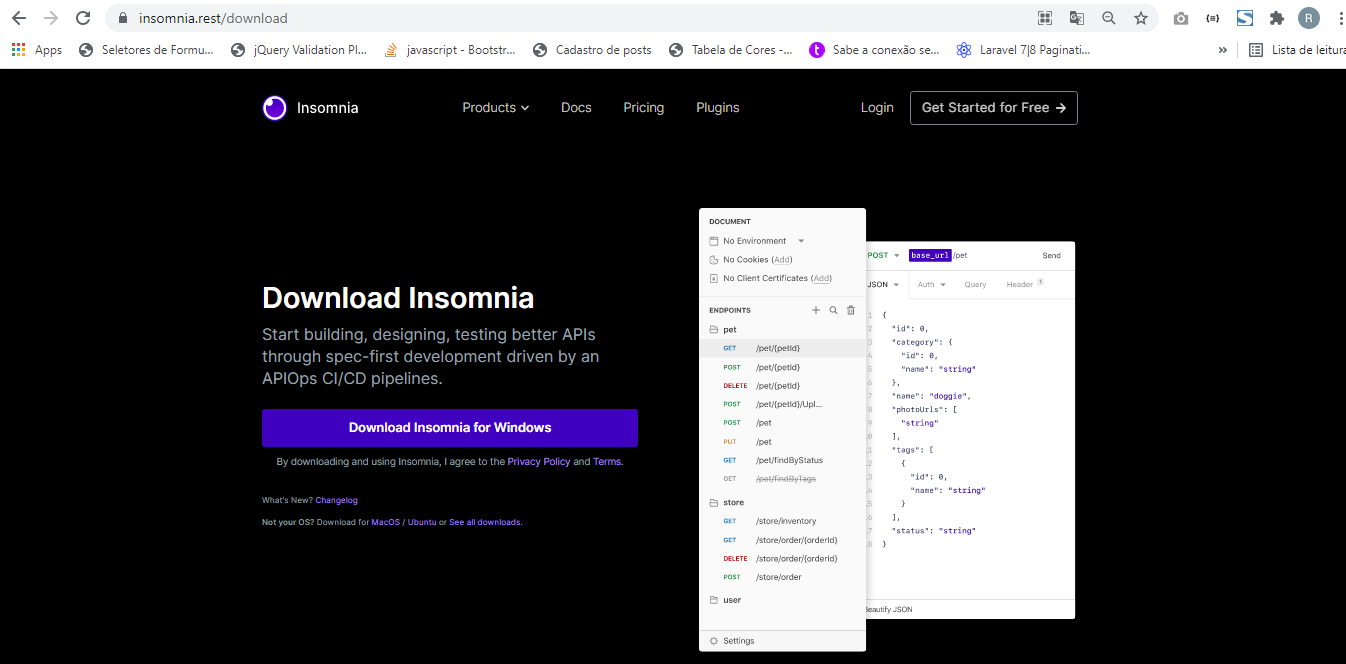
http://localhost:5000/customers

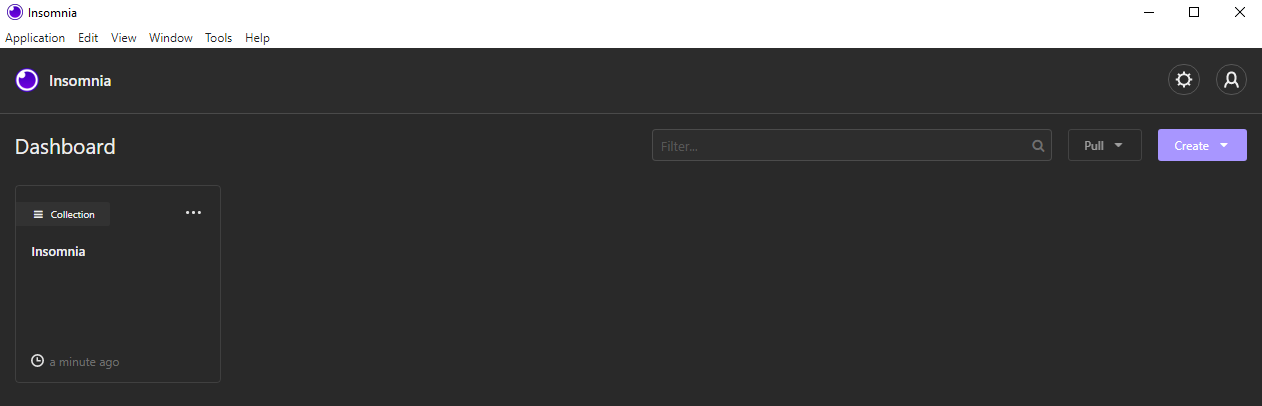


# Aula 04 - Utilizando Postman para interagir com nossa API REST

- Faça o download do Insomnia e o instale no computador:

https://insomnia.rest/download





**controllers\customersController.js**

const Customer = require('../models/Customer');

exports.create = async (req, res) => {

const customer = new Customer(req.body);

try {

await customer.save();

res.status(201).json({

message: "Cliente cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um cliente com o email: ${req.body.email}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

}

// listar clientes

exports.index = async (req, res) => {

try {

const customers = await Customer.find({});

res.json(customers);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

**routes\customersRoutes.js.js**

const express = require('express');

const router = express.Router();

const customersController = require('../controllers/customersController');

module.exports = function() {

// post: /customers

router.post('/customers', customersController.create);

// get: /customers

router.get('/customers', customersController.index);

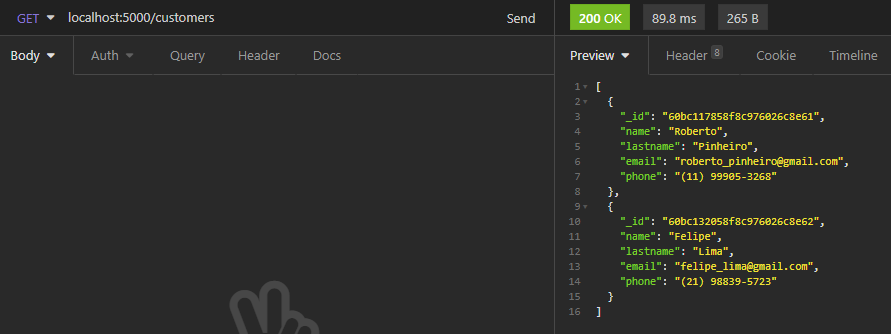
return router;

}

- No Insomnia:

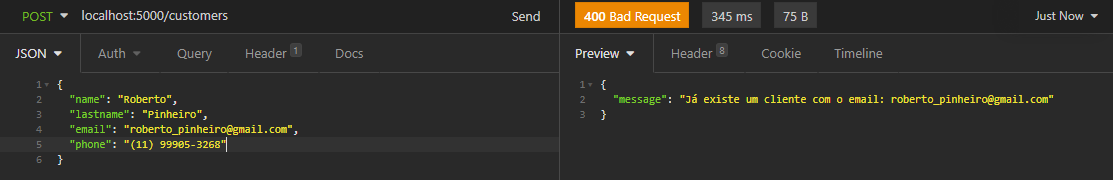
### Listando clientes

GET - localhost:5000/customers

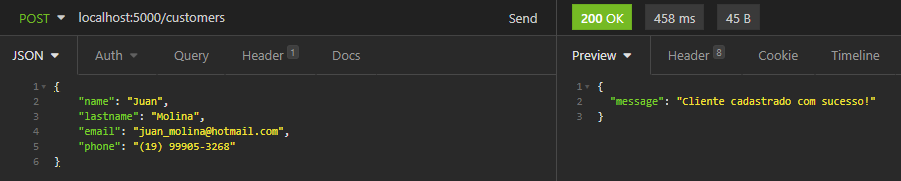


### Cadastrando cliente

POST - localhost:5000/customers

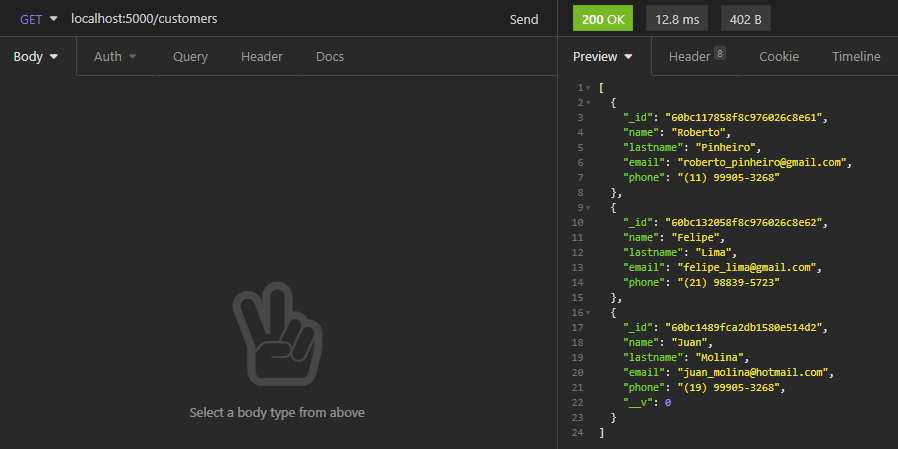


POST - localhost:5000/customers



### Listando clientes

GET - localhost:5000/customers



- No browser:

localhost:5000/customers



# Aula 05 - Consultar detalhes de um cliente por seu ID em nossa API REST

**routes\** **customersRoutes.js**

const express = require('express');

const router = express.Router();

const customersController = require('../controllers/customersController');

module.exports = function() {

// post: /customers

router.post('/customers', customersController.add);

// get: /customers

router.get('/customers', customersController.list);

// get: /customers/:id

router.get('/customers/:id', customersController.show);

return router;

};

**controllers\customersController.js**

const Customer = require('../models/Customer');

// cadastrar cliente

exports.create = async (req, res) => {

const customer = new Customer(req.body);

try {

await customer.save();

res.status(201).json({

message: "Cliente cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um cliente com o email: ${req.body.email}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

}

// listar clientes

exports.index = async (req, res) => {

try {

const customers = await Customer.find({});

res.json(customers);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

// exibir detalhes de um cliente por id

exports.show = async (req, res, next) => {

try {

const customer = await Customer.findById(req.params.id);

if(!customer){

res.status(404).json({

message: "O cliente não existe!"

});

}

res.json(customer);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

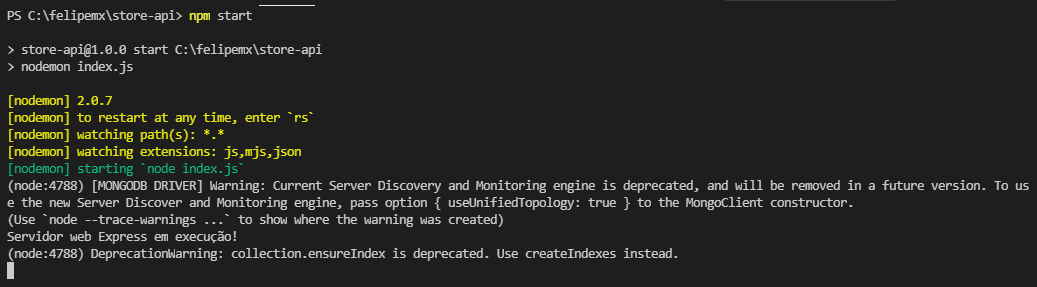
});

}

};

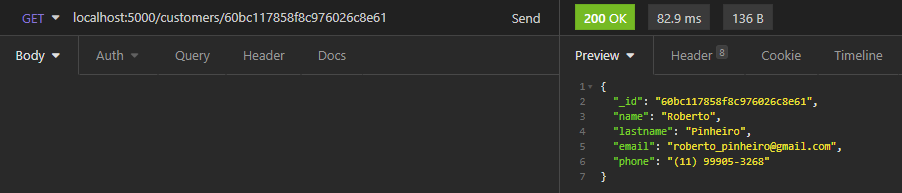
mongod

npm start

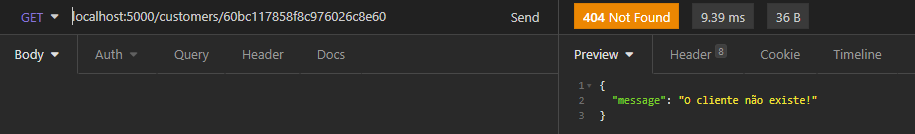


- No Insomnia:

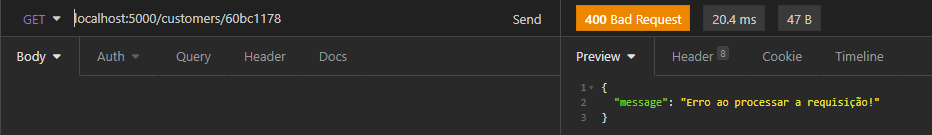
localhost:5000/customers/60bc117858f8c976026c8e61



localhost:5000/customers/60bc117858f8c976026c8e60



localhost:5000/customers/60bc1178



# Aula 06 - Atualizar um cliente em nossa API REST

**routes\customersRoutes.js**

const express = require('express');

const router = express.Router();

const customersController = require('../controllers/customersController');

module.exports = function() {

// post: /customers

router.post('/customers', customersController.create);

// get: /customers

router.get('/customers', customersController.index);

// get: /customers/:id

router.get('/customers/:id', customersController.show);

// put: /customers/:id

router.put('/customers/:id', customersController.update);

return router;

};

**controllers\customersController.js**

const Customer = require('../models/Customer');

exports.create = async (req, res) => {

const customer = new Customer(req.body);

try {

await customer.save();

res.status(201).json({

message: "Cliente cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um cliente com o email: ${req.body.email}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

}

// listar clientes

exports.index = async (req, res) => {

try {

const customers = await Customer.find({});

res.json(customers);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

// exibir detalhes de um cliente por id

exports.show = async (req, res, next) => {

try {

const customer = await Customer.findById(req.params.id);

if(!customer){

res.status(404).json({

message: "O cliente não existe!"

});

}

res.json(customer);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados do cliente

exports.update = async (req, res, next) => {

try {

const customer = await Customer.findOneAndUpdate(

{\_id: req.params.id},

req.body,

{new: true}

);

res.json({

message: 'Cliente atualizado com sucesso!'

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um cliente com o email: ${req.body.email}`

});

} else {

res.status(400).json({

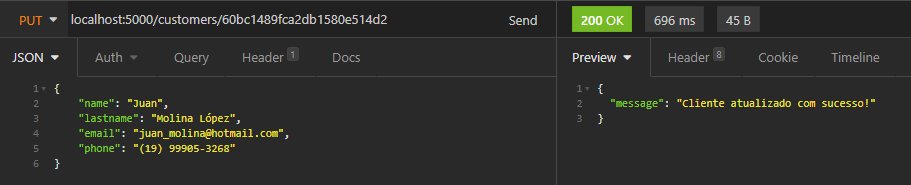
message: "Erro ao processar a requisição!"

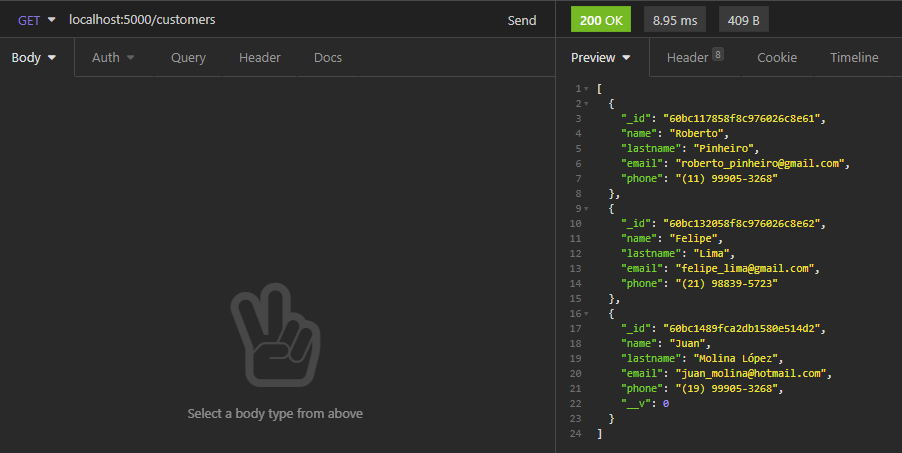
});

}

}

};





# Aula 07 - Eliminar um cliente em nossa API REST

**routes\customersRoutes.js**

const express = require('express');

const router = express.Router();

const customersController = require('../controllers/customersController');

module.exports = function() {

// post: /customers

router.post('/customers', customersController.create);

// get: /customers

router.get('/customers', customersController.index);

// get: /customers/:id

router.get('/customers/:id', customersController.show);

// put: /customers/:id

router.put('/customers/:id', customersController.update);

// delete: /customers/:id

router.delete('/customers/:id', customersController.delete);

return router;

};

**controllers\customersController.js**

const Customer = require('../models/Customer');

exports.create = async (req, res) => {

const customer = new Customer(req.body);

try {

await customer.save();

res.status(201).json({

message: "Cliente cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um cliente com o email: ${req.body.email}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

}

// listar clientes

exports.index = async (req, res) => {

try {

const customers = await Customer.find({});

res.json(customers);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

// exibir detalhes de um cliente por id

exports.show = async (req, res, next) => {

try {

const customer = await Customer.findById(req.params.id);

if(!customer){

res.status(404).json({

message: "O cliente não existe!"

});

}

res.json(customer);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados do cliente

exports.update = async (req, res, next) => {

try {

const customer = await Customer.findOneAndUpdate(

{\_id: req.params.id},

req.body,

{new: true}

);

res.json({

message: 'Cliente atualizado com sucesso!'

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um cliente com o email: ${req.body.email}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// Excluir um cliente por id

exports.delete = async (req, res, next) => {

try {

await Customer.findOneAndDelete({

\_id: req.params.id

});

res.json({

message: 'Cliente excluido com sucesso!'

});

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

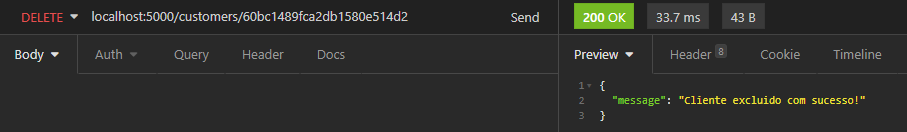
});

}

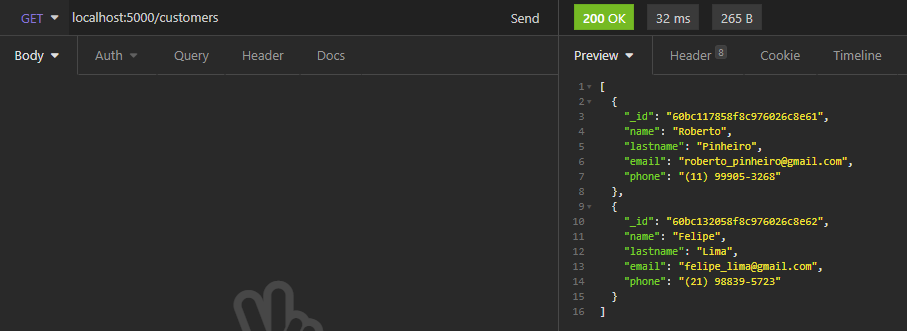
};

- No Insomnia:

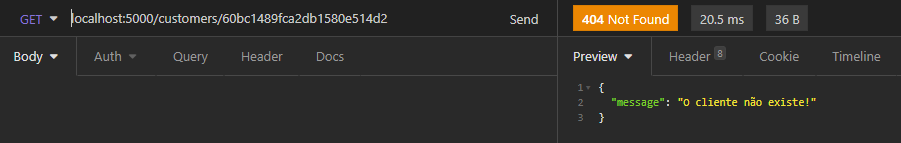
localhost:5000/customers/60bc1489fca2db1580e514d2



localhost:5000/customers



localhost:5000/customers/60bc1489fca2db1580e514d2



# Aula 08 - CRUD de Produtos

**models\Product.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const productsSchema = new Schema({

sku: {

type: String,

trim: true,

unique: true,

uppercase: true

},

name: {

type: String,

trime: true

},

description: {

type: String,

trim: true

},

price: {

type: Number

},

stock: {

type: Number,

default: 0

},

available: {

type: Boolean,

default: true

}

});

module.exports = mongoose.model('Product', productsSchema);

OBS.: Cada produto do estoque tem um SKU (Stock Keeping Unit) particular, que é um código formado por letras e números. Ao gerenciar o cadastro e retirada de produtos no seu estoque, ao invés de buscar pelo nome de cada produto, você procura pelo SKU.

**controllers\productsController.js**

const Product = require('../models/Product');

// cadastrar produto

exports.create = async (req, res) => {

const product = new Product(req.body);

try {

await product.save();

res.status(201).json({

message: "Produto cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um produto com o sku: ${req.body.sku}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// listar produtos

exports.index = async (req, res) => {

try {

const products = await Product.find({});

res.json(products);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

// exibir detalhes de um produto por id

exports.show = async (req, res, next) => {

try {

const product = await Product.findById(req.params.id);

if(!product){

res.status(404).json({

message: "O produto não existe!"

});

}

res.json(product);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados de um produto

exports.update = async (req, res, next) => {

try {

const product = await Product.findOneAndUpdate(

{\_id: req.params.id},

req.body,

{new: true}

);

res.json({

message: 'Produto atualizado com sucesso!'

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um produto com o sku: ${req.body.sku}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// Excluir um produto por id

exports.delete = async (req, res, next) => {

try {

await Product.findOneAndDelete({

\_id: req.params.id

});

res.json({

message: 'Produto excluido com sucesso!'

});

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

**routes\productsRoutes.js**

const express = require('express');

const router = express.Router();

const productsController = require('../controllers/productsController');

module.exports = function() {

// post: /products

router.post('/products', productsController.create);

// get: /products

router.get('/products', productsController.index);

// get: /products/:id

router.get('/products/:id', productsController.show);

// put: /products/:id

router.put('/products/:id', productsController.update);

// delete: products/:id

router.delete('/products/:id', productsController.delete);

return router;

};

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const customersRoutes = require('./routes/customersRoutes');

const productsRoutes = require('./routes/productsRoutes');

const app = express();

// Configurando mongoose

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/store-api',

{

useNewUrlParser: true

});

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.use('/', customersRoutes());

app.use('/', productsRoutes());

app.listen(5000, () => {

console.log('Servidor web Express em execução!');

})

### Cadastrando produtos

- No Insomnia:

POST - localhost:5000/products

{

"sku": "A11",

"name": "Ventilador",

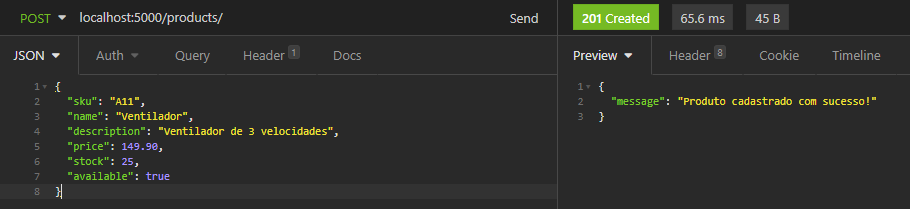
"description": "Ventilador de 3 velocidades",

"price": 149.90,

"stock": 25,

"available": true

}



POST - localhost:5000/products

{

"sku": "B12",

"name": "Liquidificador",

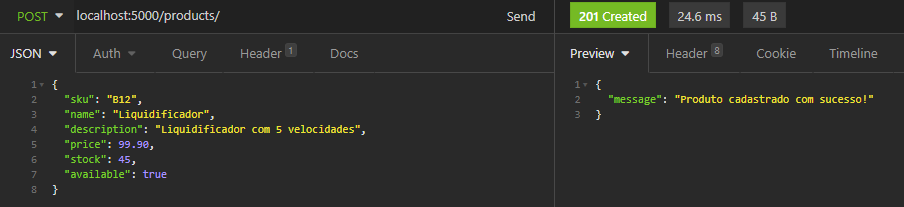
"description": "Liquidificador com 5 velocidades",

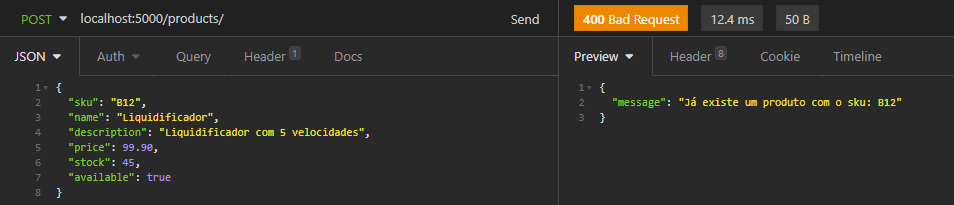
"price": 99.90,

"stock": 45,

"available": true

}





POST - localhost:5000/products

{

"sku": "E57",

"name": "Rádio relógio",

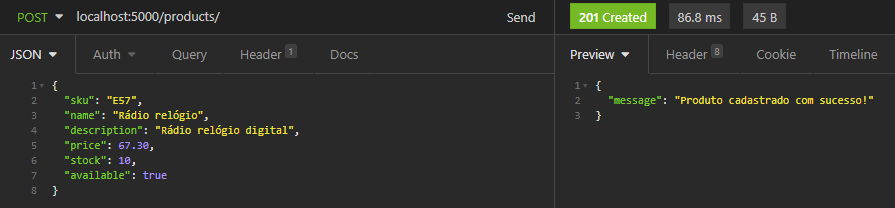
"description": "Rádio relógio digital",

"price": 67.30,

"stock": 10,

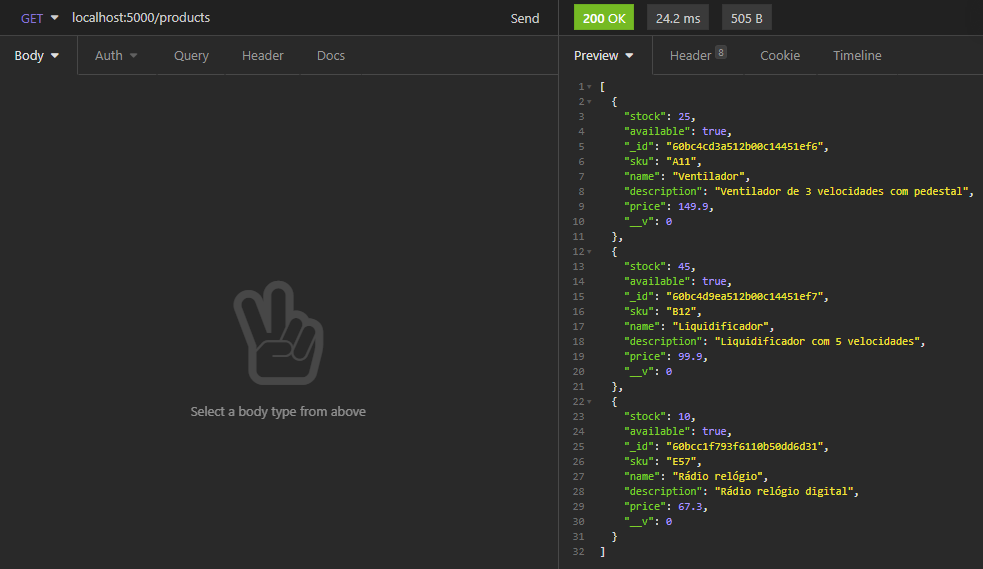
"available": true

}



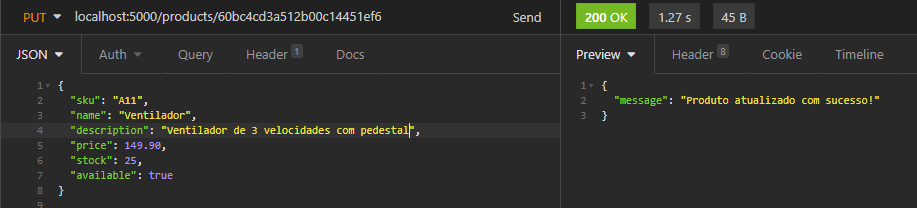
### Listando os produtos

GET - localhost:5000/products

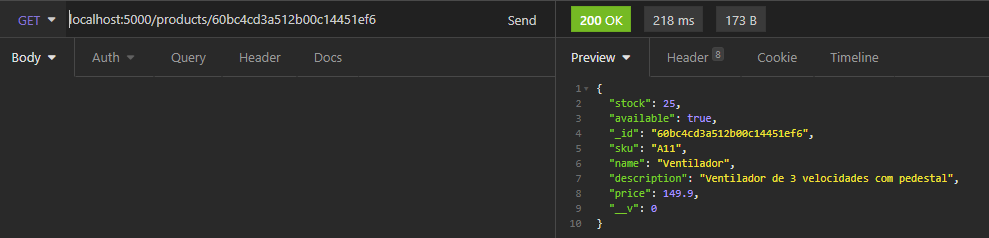


### Atualizando produtos

PUT - localhost:5000/products/60bc4cd3a512b00c14451ef6

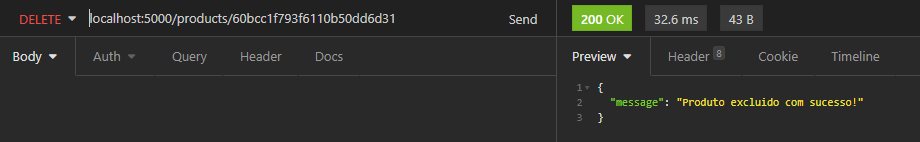


GET - localhost:5000/products/60bc4cd3a512b00c14451ef6



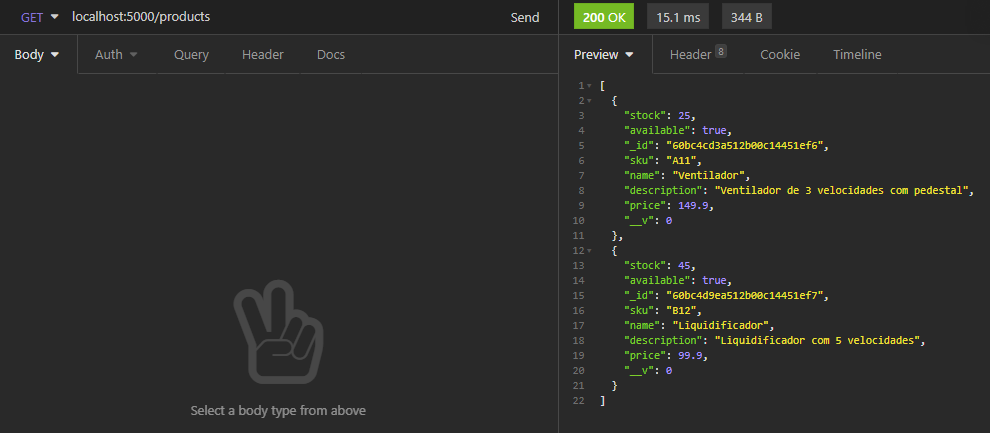
### Excluindo produto

DELETE - localhost:5000/products/60bcc1f793f6110b50dd6d31



### Listando produtos

GET - localhost:5000/products

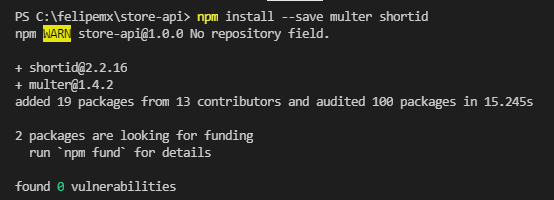


# Aula 09 - Subir imagem de produto

## Instalando multer e shortid

npm install --save multer shortid

multer será utilizado para processar e armazenar o artigo que recebemos na requisição e shortid utilizaremos para gerar id aleatório



**models\Product.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const productsSchema = new Schema({

sku: {

type: String,

trim: true,

unique: true,

uppercase: true

},

name: {

type: String,

trime: true

},

description: {

type: String,

trim: true

},

image: {

type: String

},

price: {

type: Number

},

stock: {

type: Number,

default: 0

},

available: {

type: Boolean,

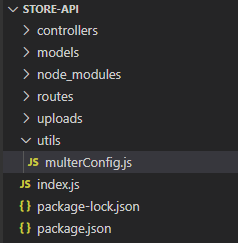
default: true

}

});

module.exports = mongoose.model('Product', productsSchema);

Na pasta raiz do projeto adicione duas pastas: uploads e utils e dentro desta última pasta adicione um arquivo chamado "multerConfig.js":



**utils\multerConfig.js**

// https://github.com/afelipelc/custom-snippets/blob/master/multerConfig.js

const multer = require('multer');

const shortid = require('shortid');

const multerConfig = {

storage: fileStorage = multer.diskStorage({

destination: (req, file, cb) => {

cb(null, \_\_dirname+'../../uploads/'); // los uploads se subirán en esta carpeta

},

filename: (req, file, cb) => {

// obtener la extensión del archivo

const extension = file.mimetype.split('/')[1];

// generar ID para ponerlo como nombre de imagen

cb(null, `${shortid.generate()}.${extension}`);

}

}),

fileFilter(req, file, cb) {

if ( file.mimetype === 'image/jpeg' || file.mimetype ==='image/png' ) { // solo aceptar imágenes

cb(null, true);

} else {

cb(new Error('Formato de imagem inválido!'))

}

},

}

module.exports = multerConfig;

**models\Product.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const productsSchema = new Schema({

sku: {

type: String,

trim: true,

unique: true,

uppercase: true

},

name: {

type: String,

trime: true

},

description: {

type: String,

trim: true

},

image: {

type: String

},

price: {

type: Number

},

stock: {

type: Number,

default: 0

},

available: {

type: Boolean,

default: true

}

});

module.exports = mongoose.model('Product', productsSchema);

**controllers\productsController.js**

const multer = require('multer');

const multerConfig = require('../utils/multerConfig');

const Product = require('../models/Product');

const upload = multer(multerConfig).single('image');

exports.fileUpload = (req, res, next) => {

upload(req, res, function(error) {

if(error){

res.json({message: error});

}

return next();

});

};

// cadastrar produto

exports.create = async (req, res) => {

const product = new Product(req.body);

try {

if (req.file && req.file.filename) {

product.image = req.file.filename;

}

await product.save();

res.status(201).json({

message: "Produto cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um produto com o sku: ${req.body.sku}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// listar produtos

exports.index = async (req, res) => {

try {

const products = await Product.find({});

res.json(products);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

// exibir detalhes de um produto por id

exports.show = async (req, res, next) => {

try {

const product = await Product.findById(req.params.id);

if(!product){

res.status(404).json({

message: "O produto não existe!"

});

}

res.json(product);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados de um produto

exports.update = async (req, res, next) => {

try {

let newProduct = req.body;

if (req.file && req.file.filename) {

newProduct.image = req.file.filename;

} else {

const product = await Product.findById(req.params.id);

newProduct.image = product.image;

}

const productUpdated = await Product.findOneAndUpdate(

{\_id: req.params.id},

newProduct,

{new: true}

);

res.json({

message: 'Produto atualizado com sucesso!'

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um produto com o sku: ${req.body.sku}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// Excluir um produto por id

exports.delete = async (req, res, next) => {

try {

await Product.findOneAndDelete({

\_id: req.params.id

});

res.json({

message: 'Produto excluido com sucesso!'

});

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

**routes\productsRoutes.js**

const express = require('express');

const router = express.Router();

const productsController = require('../controllers/productsController');

module.exports = function() {

// post: /products

router.post('/products', productsController.fileUpload, productsController.create);

// get: /products

router.get('/products', productsController.index);

// get: /products/:id

router.get('/products/:id', productsController.show);

// put: /products/:id

router.put('/products/:id', productsController.fileUpload, productsController.update);

// delete: products/:id

router.delete('/products/:id', productsController.delete);

return router;

};

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const customersRoutes = require('./routes/customersRoutes');

const productsRoutes = require('./routes/productsRoutes');

const app = express();

// Configurando mongoose

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/store-api',

{

useNewUrlParser: true

});

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.use('/', customersRoutes());

app.use('/', productsRoutes());

app.use(express.static('uploads'));

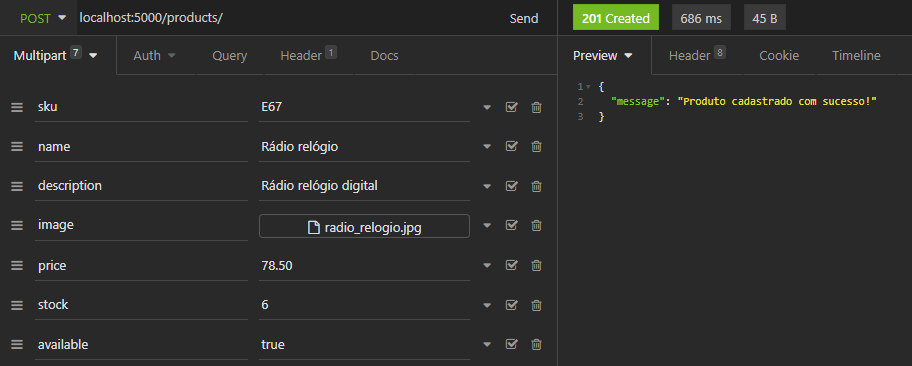
app.listen(5000, () => {

console.log('Servidor web Express em execução!');

})

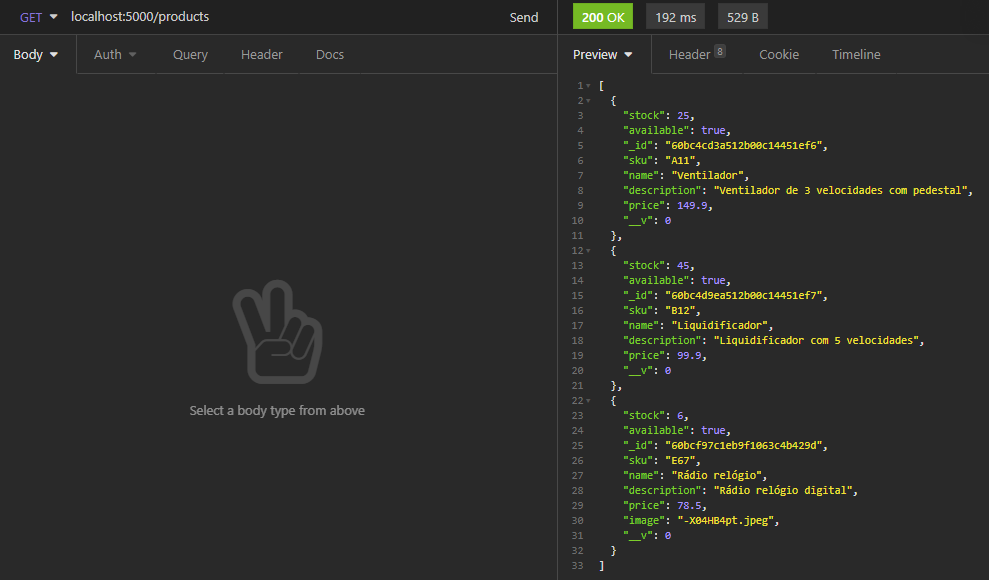
### Cadastrando um novo produto

POST - localhost:5000/products/



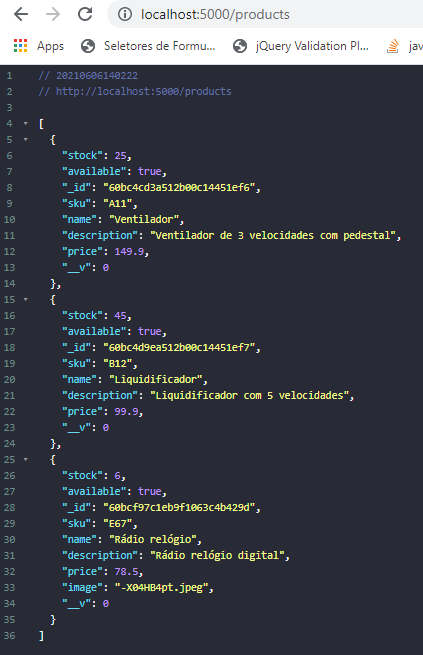
### Listando produtos

GET - localhost:5000/products



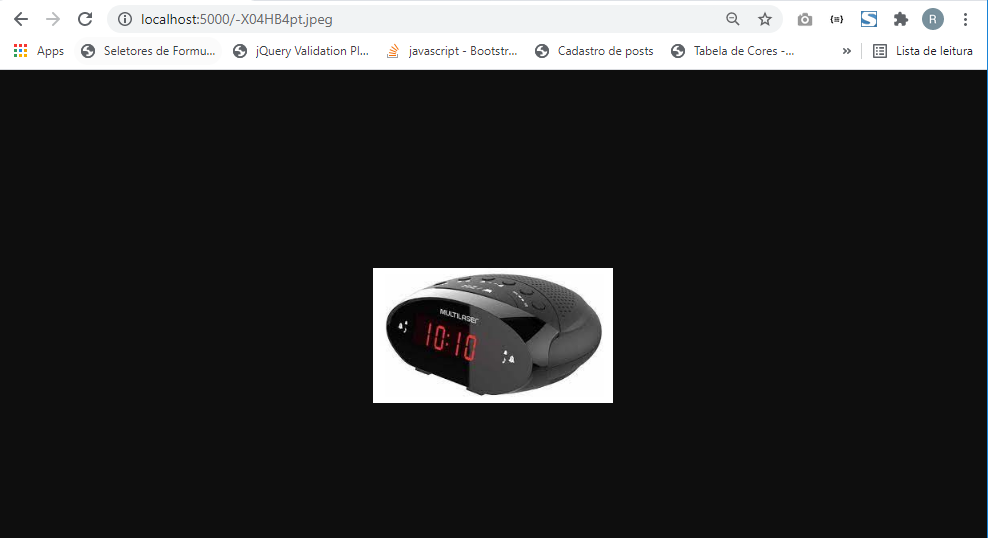
- No browser:

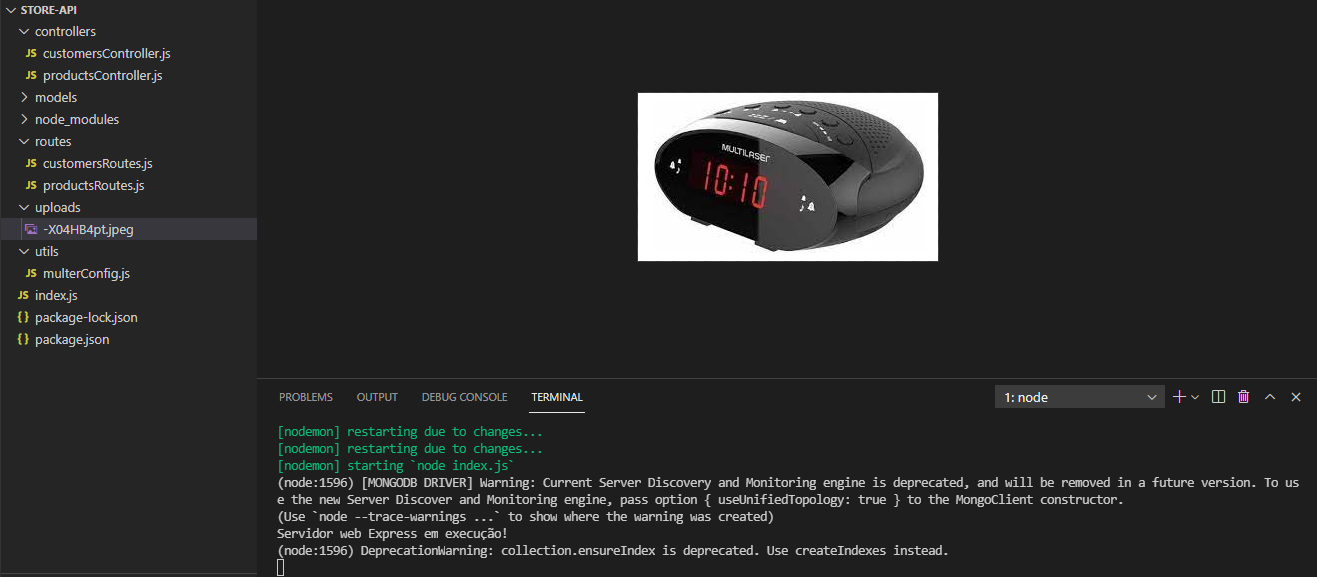
http://localhost:5000/products



### Imagem do produto

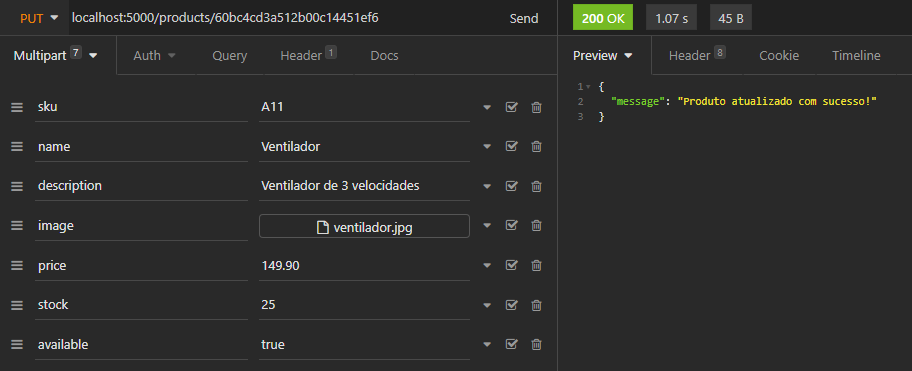
http://localhost:5000/-X04HB4pt.jpeg



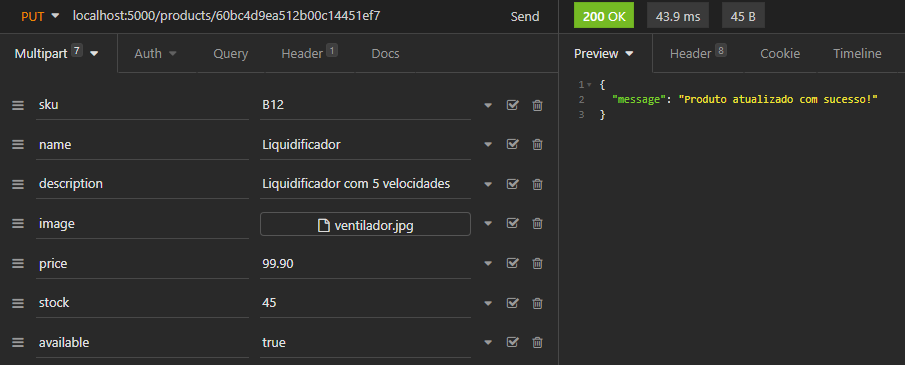


### Atualizando produtos

PUT - localhost:5000/products/60bc4cd3a512b00c14451ef6

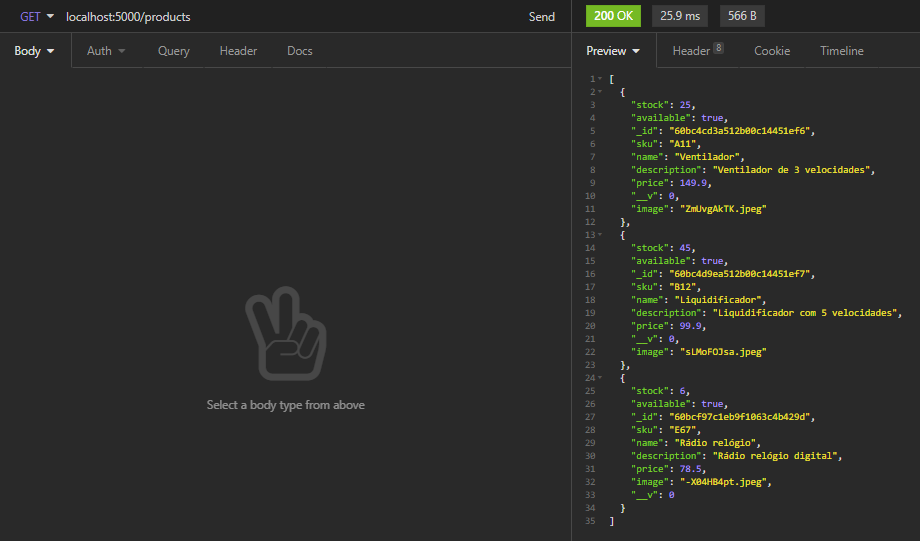


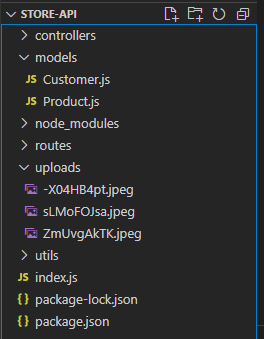
PUT - localhost:5000/products/60bc4d9ea512b00c14451ef7



### Listando produtos

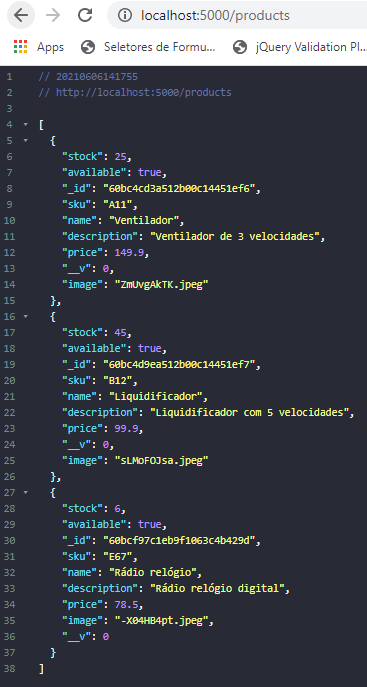
GET - localhost:5000/products





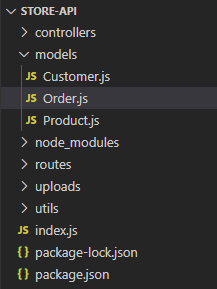
- No browser:

http://localhost:5000/products



# Aula 10 - Registrar pedido e mostrar a listagem de pedidos

- Dentro da pasta "models" adicione um arquivo chamado "Order.js":



**models\Order.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const ordersSchema = new Schema({

created: {

type: Date,

default: Date.now

},

customer: {

type: Schema.ObjectId,

ref: 'Customer'

},

products: [{

product: {

type: Schema.ObjectId,

ref: 'Product'

},

unitPrice: {

type: Number

},

quantity: {

type: Number

},

amount: {

type: Number

}

}],

totalAmount: {

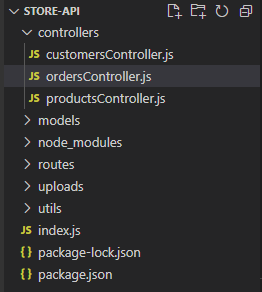
type: Number

}

});

module.exports = mongoose.model('Order', ordersSchema);

- Dentro da pasta "controllers" adicione um arquivo chamado "ordersController.js":



**controllers\ordersController.js**

const Order = require('../models/Order');

// cadastrar pedido

exports.create = async (req, res, next) => {

try {

const order = new Order(req.body);

await order.save();

res.json(order);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// listar pedidos

exports.index = async (req, res, next) => {

try {

const orders = await Order.find({})

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(orders);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

**routes\ordersRoutes.js**

const express = require('express');

const router = express.Router();

const ordersController = require('../controllers/ordersController');

module.exports = function() {

// post: /orders

router.post('/orders', ordersController.create);

// get: /orders

router.get('/orders', ordersController.index);

return router;

};

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const customersRoutes = require('./routes/customersRoutes');

const productsRoutes = require('./routes/productsRoutes');

const ordersRoutes = require('./routes/ordersRoutes');

const app = express();

// Configurando mongoose

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/store-api',

{

useNewUrlParser: true

});

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.use('/', customersRoutes());

app.use('/', productsRoutes());

app.use('/', ordersRoutes());

app.use(express.static('uploads'));

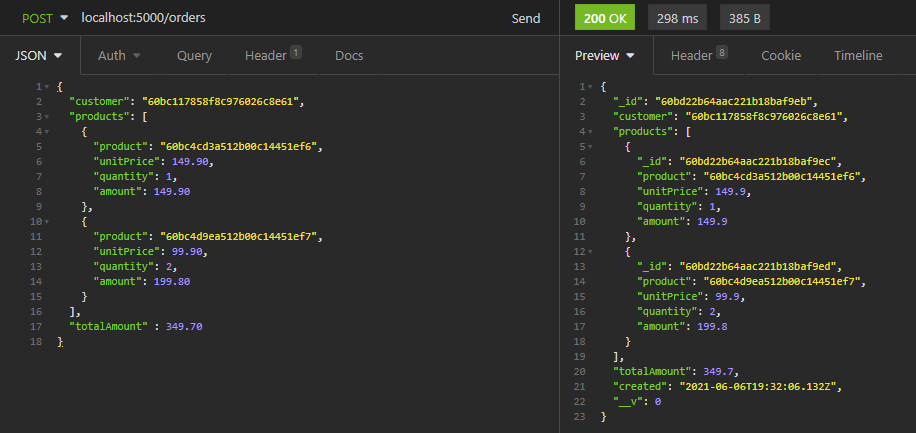
app.listen(5000, () => {

console.log('Servidor web Express em execução!');

})

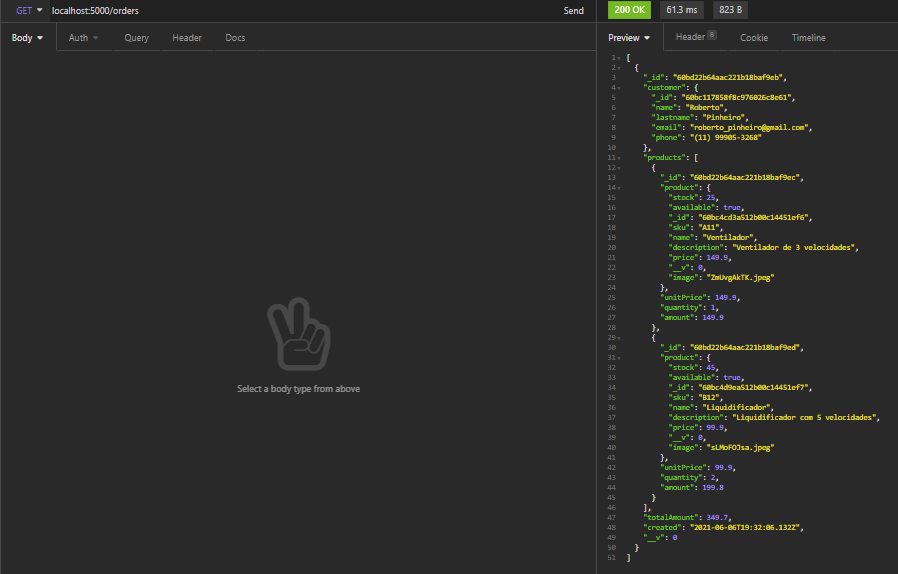
### Cadastrando um pedido

POST - localhost:5000/orders



### Listando pedidos

GET - localhost:5000/orders

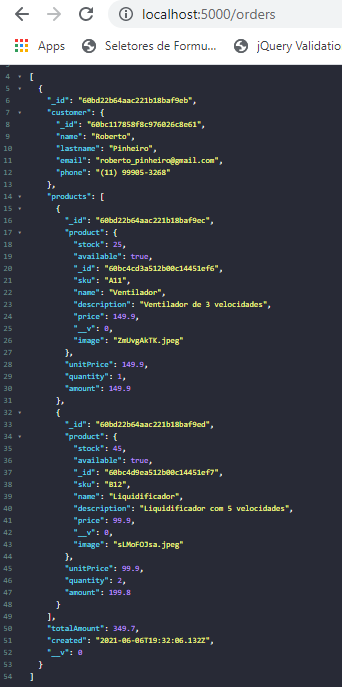


- No Mongo Compass:



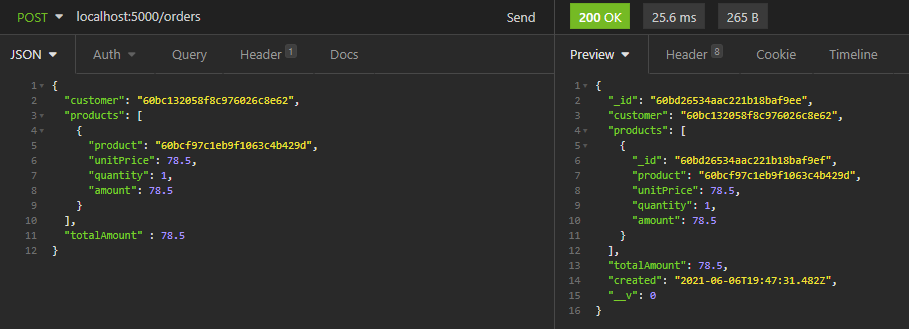
- No browser:

http://localhost:5000/orders



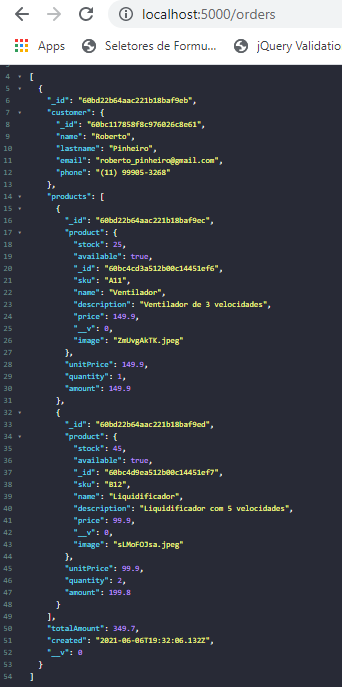
### Cadastrando um novo pedido

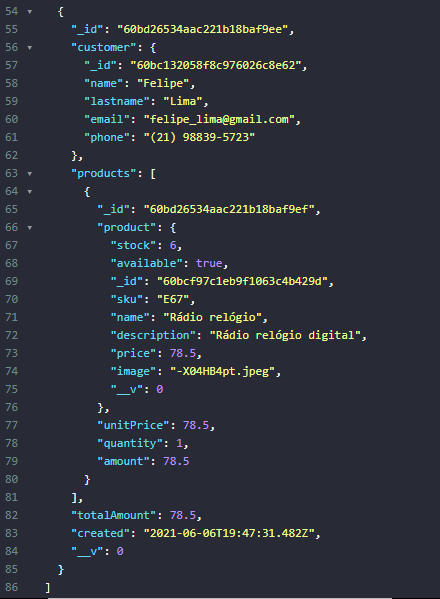
POST - localhost:5000/orders



### Listando pedidos

- No browser:





# Aula 11 - Atualizar pedido

## Exibir dados de um pedido específico

**controllers\ordersController.js**

const Order = require('../models/Order');

// cadastrar pedido

exports.create = async (req, res, next) => {

try {

const order = new Order(req.body);

await order.save();

res.json(order);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// listar pedidos

exports.index = async (req, res, next) => {

try {

const orders = await Order.find({})

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(orders);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// exibir detalhes de um pedido por id

exports.show = async (req, res, next) => {

try {

const order = await Order.findById(req.params.id)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

if(!order){

res.status(404).json({

message: "O pedido não existe!"

});

next();

}

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

**routes\ordersRoutes.js**

const express = require('express');

const router = express.Router();

const ordersController = require('../controllers/ordersController');

module.exports = function() {

// post: /orders

router.post('/orders', ordersController.create);

// get: /orders

router.get('/orders', ordersController.index);

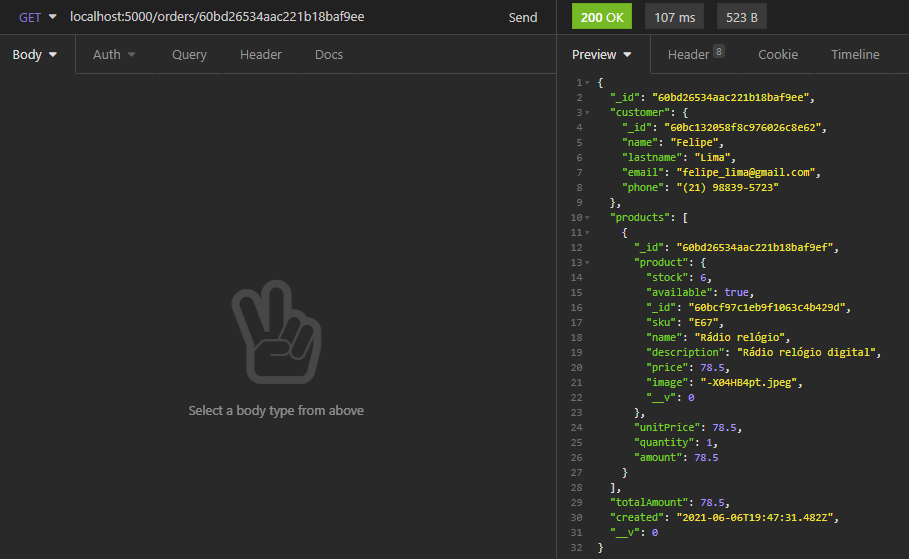
// get: /orders/:id

router.get('/orders/:id', ordersController.show);

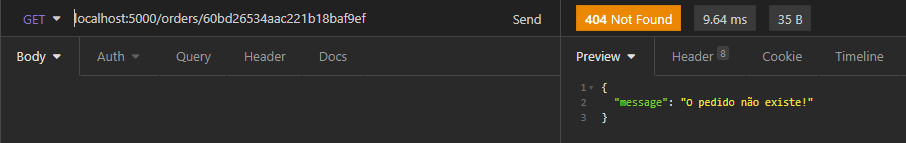
return router;

};

GET - localhost:5000/orders/60bd26534aac221b18baf9ee



GET - localhost:5000/orders/60bd26534aac221b18baf9ef



## Atualizando um pedido

**controllers\ordersController.js**

const Order = require('../models/Order');

// cadastrar pedido

exports.create = async (req, res, next) => {

try {

const order = new Order(req.body);

await order.save();

res.json(order);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// listar pedidos

exports.index = async (req, res, next) => {

try {

const orders = await Order.find({})

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(orders);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// exibir detalhes de um pedido por id

exports.show = async (req, res, next) => {

try {

const order = await Order.findById(req.params.id)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

if(!order){

res.status(404).json({

message: "O pedido não existe!"

});

next();

}

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados de um pedido

exports.update = async (req, res, next) => {

try {

const order = await Order.findOneAndUpdate(

{ \_id: req.params.id },

req.body,

{ new: true }

)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

}

**routes\ordersRoutes.js**

const express = require('express');

const router = express.Router();

const ordersController = require('../controllers/ordersController');

module.exports = function() {

// post: /orders

router.post('/orders', ordersController.create);

// get: /orders

router.get('/orders', ordersController.index);

// get: /orders/:id

router.get('/orders/:id', ordersController.show);

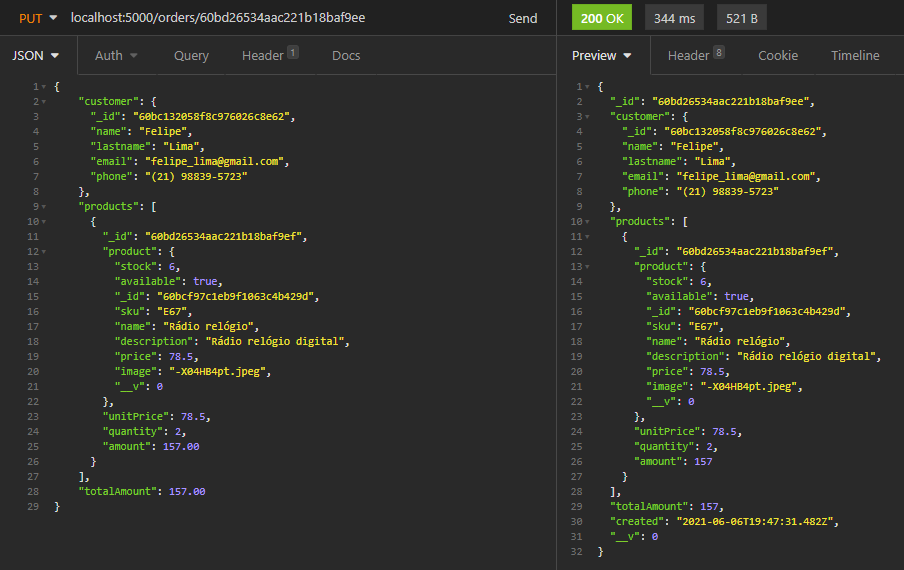
// put: /orders/:id

router.put('/orders/:id', ordersController.update);

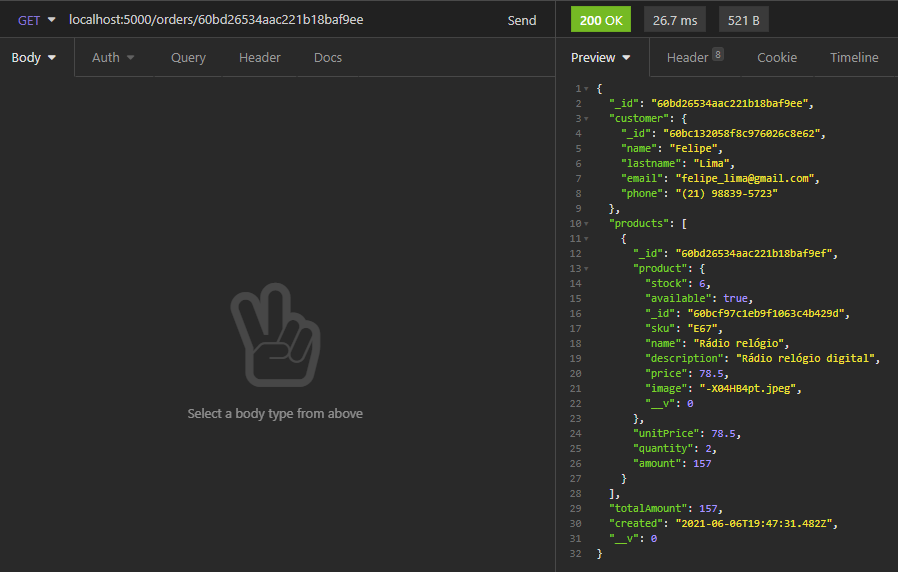
return router;

};

PUT - localhost:5000/orders/60bd26534aac221b18baf9ee



GET - localhost:5000/orders/60bd26534aac221b18baf9ee



# Aula 12 - Eliminar pedido

**controllers\ordersController.js**

const Order = require('../models/Order');

// cadastrar pedido

exports.create = async (req, res, next) => {

try {

const order = new Order(req.body);

await order.save();

res.json(order);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// listar pedidos

exports.index = async (req, res, next) => {

try {

const orders = await Order.find({})

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(orders);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// exibir detalhes de um pedido por id

exports.show = async (req, res, next) => {

try {

const order = await Order.findById(req.params.id)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

if(!order){

res.status(404).json({

message: "O pedido não existe!"

});

next();

}

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados de um pedido

exports.update = async (req, res, next) => {

try {

const order = await Order.findOneAndUpdate(

{ \_id: req.params.id },

req.body,

{ new: true }

)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

}

// Excluir um pedido por id

exports.delete = async (req, res, next) => {

try {

await Order.findOneAndDelete({

\_id: req.params.id

});

res.json({

message: 'Pedido excluido com sucesso!'

});

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

**routes\ordersRoutes.js**

const express = require('express');

const router = express.Router();

const ordersController = require('../controllers/ordersController');

module.exports = function() {

// post: /orders

router.post('/orders', ordersController.create);

// get: /orders

router.get('/orders', ordersController.index);

// get: /orders/:id

router.get('/orders/:id', ordersController.show);

// put: /orders/:id

router.put('/orders/:id', ordersController.update);

// delete: /orders/:id

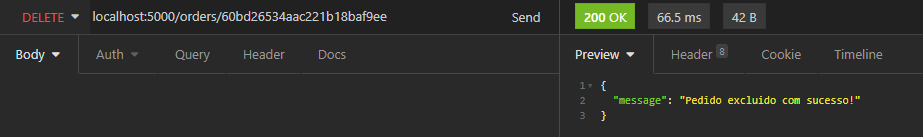
router.delete('/orders/:id', ordersController.delete);

return router;

};

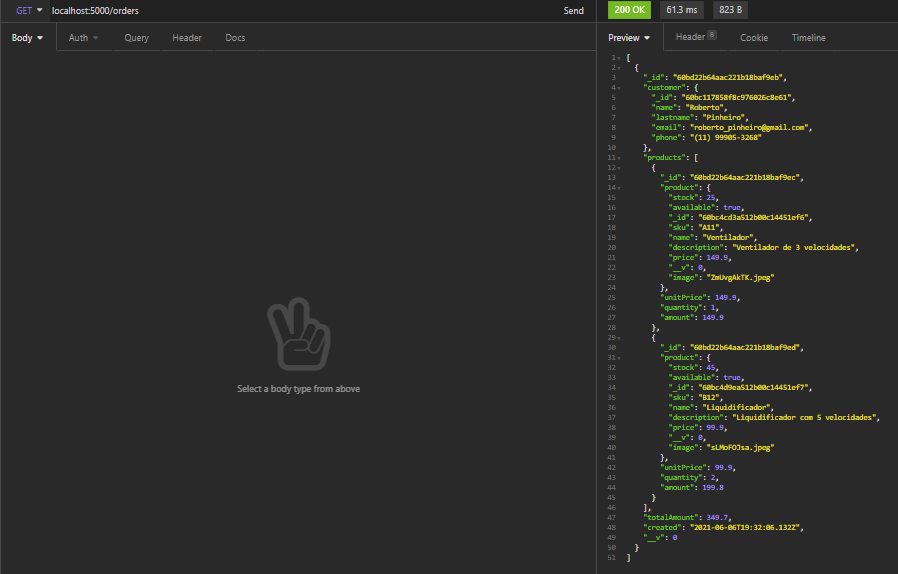
### Excluindo um pedido

DELETE - localhost:5000/orders/60bd26534aac221b18baf9ee



### Listando pedidos

GET - localhost:5000/orders



# Aula 13 - Obter pedidos por cliente

**controllers\ordersController.js**

const Order = require('../models/Order');

// cadastrar pedido

exports.create = async (req, res, next) => {

try {

const order = new Order(req.body);

await order.save();

res.json(order);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// listar pedidos

exports.index = async (req, res, next) => {

try {

const orders = await Order.find({})

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(orders);

} catch (error) {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

};

// exibir detalhes de um pedido por id

exports.show = async (req, res, next) => {

try {

const order = await Order.findById(req.params.id)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

if(!order){

res.status(404).json({

message: "O pedido não existe!"

});

next();

}

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados de um pedido

exports.update = async (req, res, next) => {

try {

const order = await Order.findOneAndUpdate(

{ \_id: req.params.id },

req.body,

{ new: true }

)

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(order);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

}

// Excluir um pedido por id

exports.delete = async (req, res, next) => {

try {

await Order.findOneAndDelete({

\_id: req.params.id

});

res.json({

message: 'Pedido excluido com sucesso!'

});

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// exibir pedidos por cliente

exports.byCustomer = async (req, res, next) => {

try {

const orders = await Order.find({ customer: req.params.id })

.populate('customer')

.populate({

path: 'products.product',

model: 'Product'

});

res.json(orders);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

**routes\ordersRoutes.js**

const express = require('express');

const router = express.Router();

const ordersController = require('../controllers/ordersController');

module.exports = function() {

// post: /orders

router.post('/orders', ordersController.create);

// get: /orders

router.get('/orders', ordersController.index);

// get: /orders/:id

router.get('/orders/:id', ordersController.show);

// get: /orders/customer/:id

router.get('/orders/customer/:id', ordersController.byCustomer);

// put: /orders/:id

router.put('/orders/:id', ordersController.update);

// delete: /orders/:id

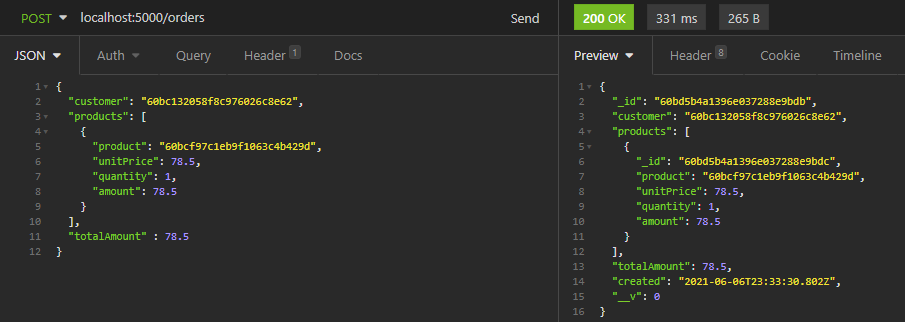
router.delete('/orders/:id', ordersController.delete);

return router;

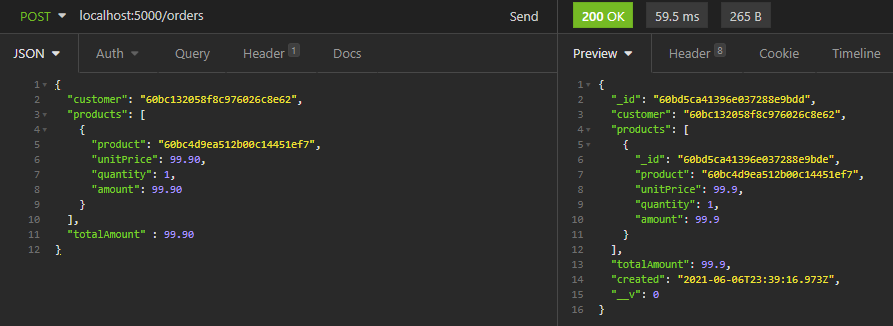
};

- Cadastre mais dois pedidos:

POST - localhost:5000/orders



POST - localhost:5000/orders



## Listando pedidos por um cliente específico

GET - localhost:5000/orders/customer/60bc132058f8c976026c8e62







# Aula 14 - Buscar produtos por nome

**controllers\productsController.js**

const multer = require('multer');

const multerConfig = require('../utils/multerConfig');

const Product = require('../models/Product');

const upload = multer(multerConfig).single('image');

exports.fileUpload = (req, res, next) => {

upload(req, res, function(error) {

if(error){

res.json({message: error});

}

return next();

});

};

// cadastrar produto

exports.create = async (req, res) => {

const product = new Product(req.body);

try {

if (req.file && req.file.filename) {

product.image = req.file.filename;

}

await product.save();

res.status(201).json({

message: "Produto cadastrado com sucesso!"

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um produto com o sku: ${req.body.sku}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// listar produtos

exports.index = async (req, res) => {

try {

const products = await Product.find({});

res.json(products);

} catch (error) {

console.log(error);

res.send(error);

next();

}

};

// exibir detalhes de um produto por id

exports.show = async (req, res, next) => {

try {

const product = await Product.findById(req.params.id);

if(!product){

res.status(404).json({

message: "O produto não existe!"

});

}

res.json(product);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

// atualizar dados de um produto

exports.update = async (req, res, next) => {

try {

let newProduct = req.body;

if (req.file && req.file.filename) {

newProduct.image = req.file.filename;

} else {

const product = await Product.findById(req.params.id);

newProduct.image = product.image;

}

const productUpdated = await Product.findOneAndUpdate(

{\_id: req.params.id},

newProduct,

{new: true}

);

res.json({

message: 'Produto atualizado com sucesso!'

});

} catch (error) {

if(error.code === 11000){

res.status(400).json({

message: `Já existe um produto com o sku: ${req.body.sku}`

});

} else {

res.status(400).json({

message: "Erro ao processar a requisição!"

});

}

}

};

// Excluir um produto por id

exports.delete = async (req, res, next) => {

try {

await Product.findOneAndDelete({

\_id: req.params.id

});

res.json({

message: 'Produto excluido com sucesso!'

});

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

};

exports.search = async (req, res, next) => {

try {

const products = await Product.find({

name: new RegExp(req.params.query, 'i')

});

res.json(products);

} catch (error) {

res.status(400).json({

message: 'Erro ao processar a requisição!'

});

}

}

**routes\productsRoutes.js**

const express = require('express');

const router = express.Router();

const productsController = require('../controllers/productsController');

module.exports = function() {

// post: /products

router.post('/products', productsController.fileUpload, productsController.create);

// get: /products

router.get('/products', productsController.index);

// get: /products/:id

router.get('/products/:id', productsController.show);

// put: /products/:id

router.put('/products/:id', productsController.fileUpload, productsController.update);

// delete: products/:id

router.delete('/products/:id', productsController.delete);

// get: /products/search/:query

router.get('/products/search/:query', productsController.search);

return router;

};

- Cadastre dois novos produtos

{

"sku": "A12",

"name": "Ventilador",

"description": "Ventilador de parede",

"price": 139.50,

"stock": 12,

"available": true

}

POST - localhost:5000/products/



{

"sku": "A13",

"name": "Ventilador",

"description": "Ventilador de teto",

"price": 109.50,

"stock": 10,

"available": true

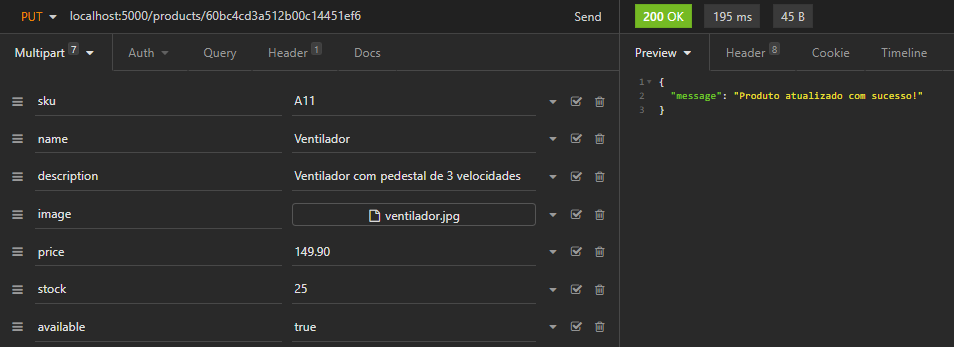
}

POST - localhost:5000/products/



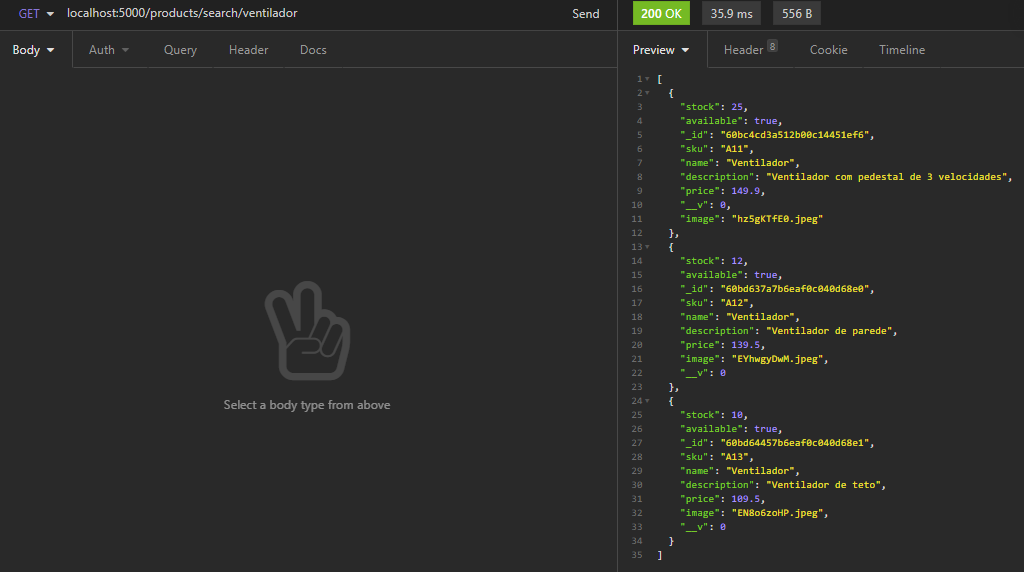
### Atualizando dados de um ventilador

PUT - localhost:5000/products/60bc4cd3a512b00c14451ef6

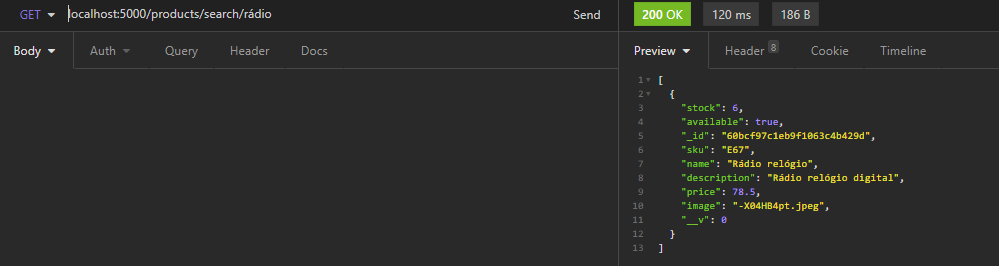


### Listando ventiladores cadastrados

GET - localhost:5000/products/search/ventilador



localhost:5000/products/search/rádio

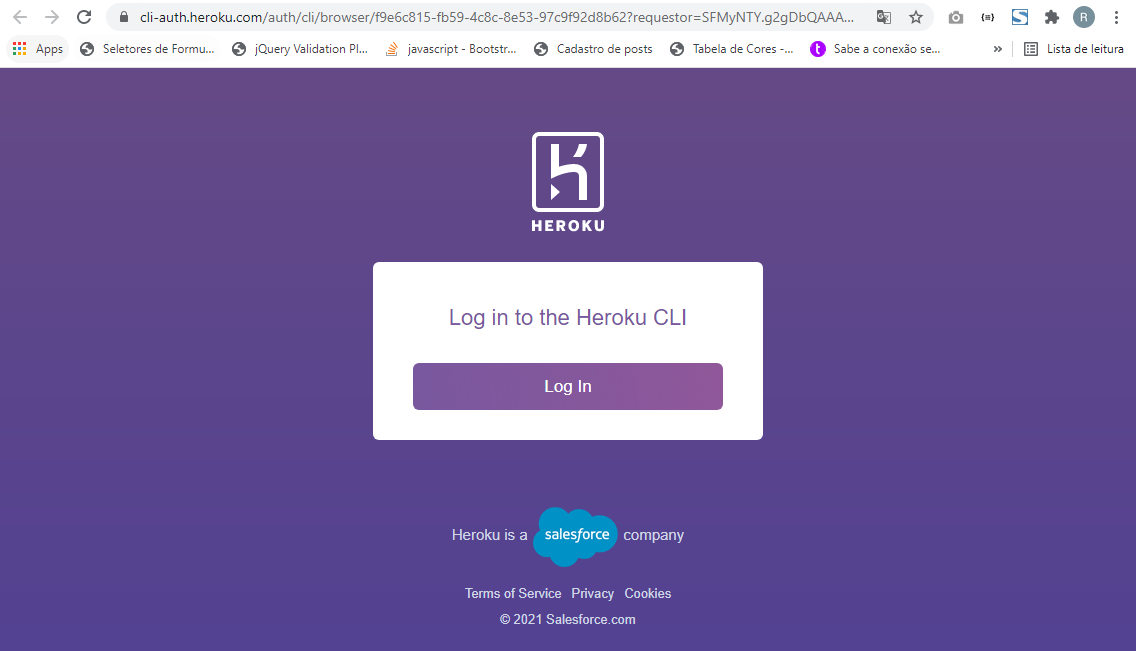


# Aula 15 - Deploy da aplicação em Heroku e MongoDB Atlas

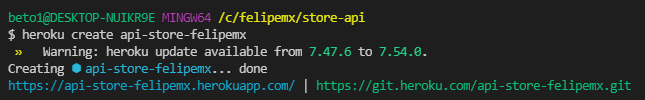
- Instale Heroku CLI em seu computador.

- No terminal, entre com o comando:

heroku login



heroku create api-store-felipemx



## Configurações

**package.json**

{

"name": "store-api",

"version": "1.0.0",

"description": "api rest",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"dev": "nodemon index.js",

"start": "node index.js"

},

"keywords": [

"api"

],

"author": "Roberto Pinheiro",

"license": "ISC",

"dependencies": {

"body-parser": "^1.19.0",

"cors": "^2.8.5",

"express": "^4.17.1",

"mongoose": "^5.12.13",

"multer": "^1.4.2",

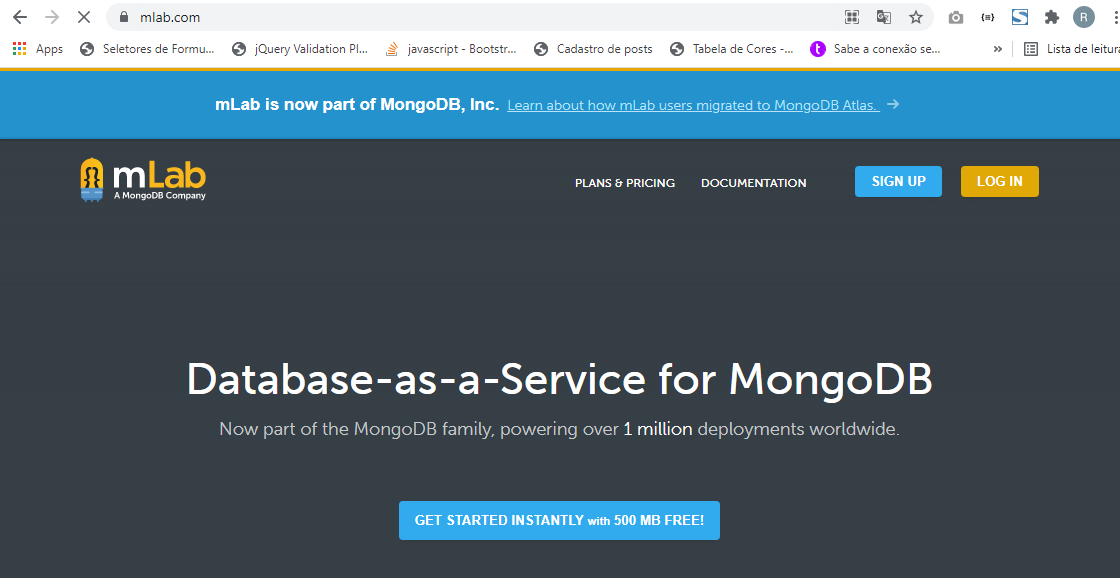
"shortid": "^2.2.16"

}

}

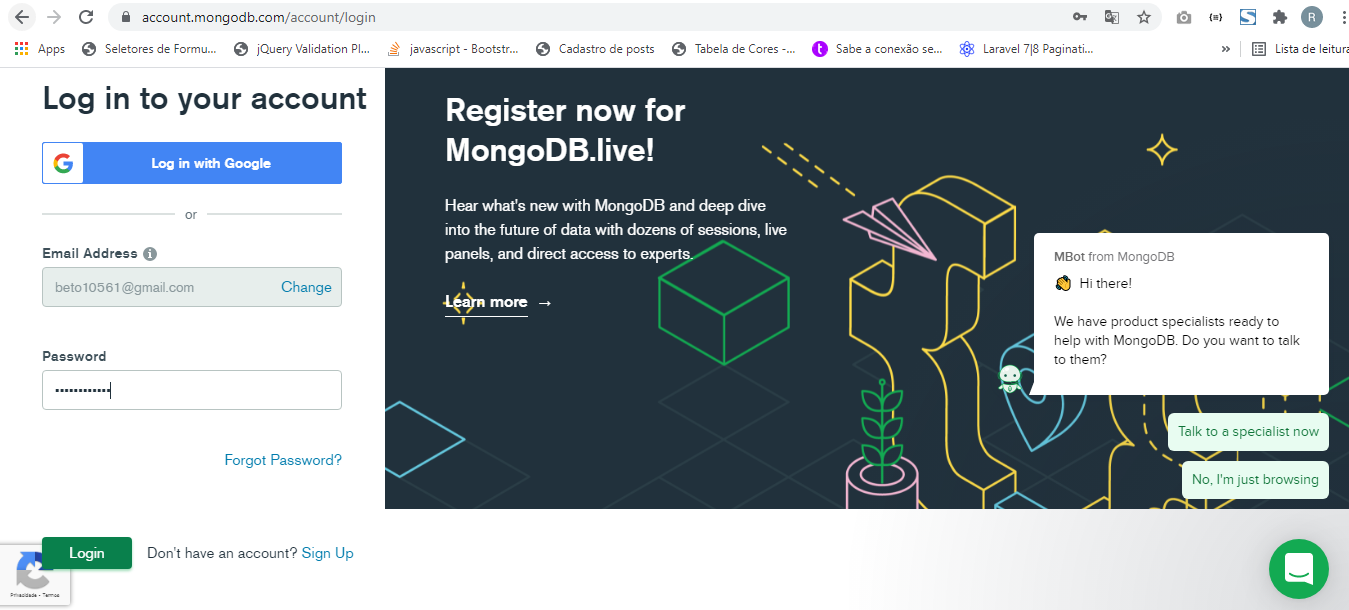
- Acesse o site do mlab:

https://mlab.com/

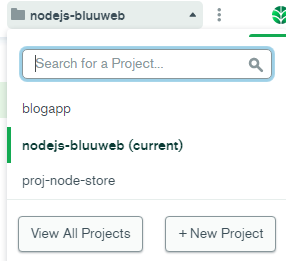


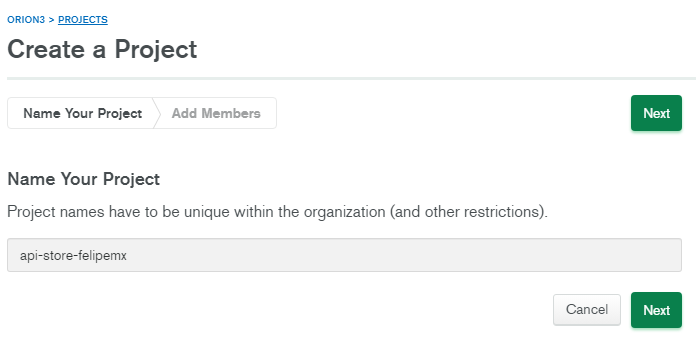
- Crie uma conta e faça o login.

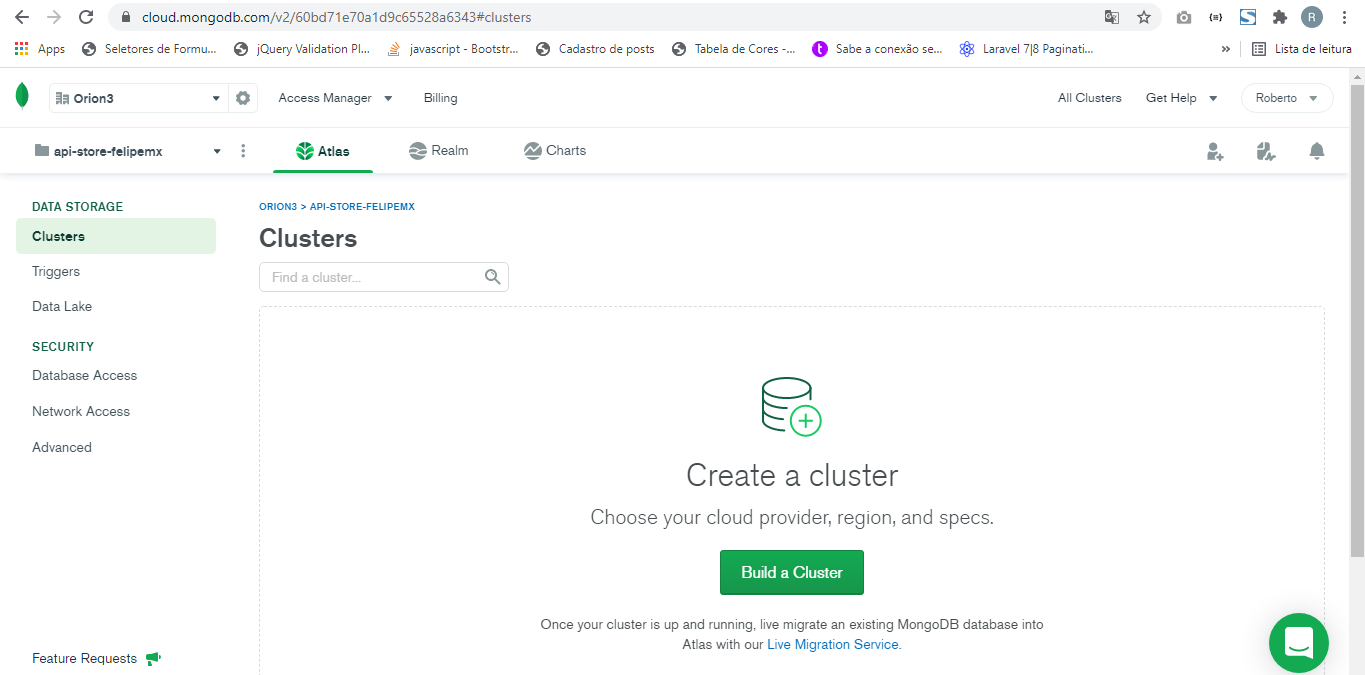
https://account.mongodb.com/account/login



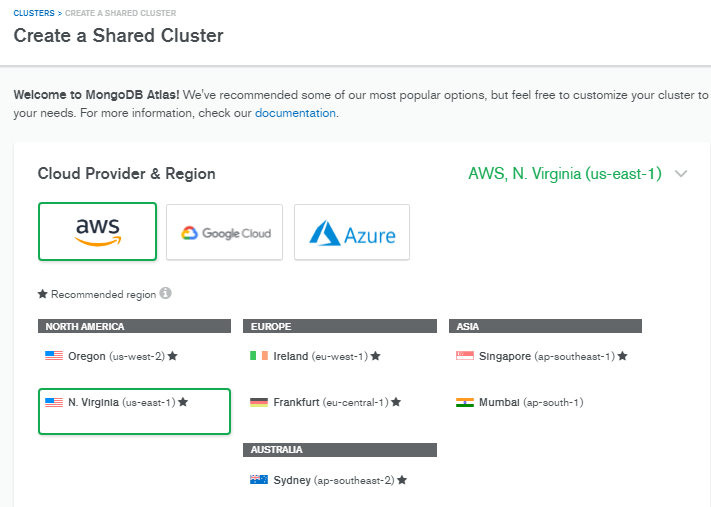
- Crie um novo projeto chamado "api-store-felipemx"

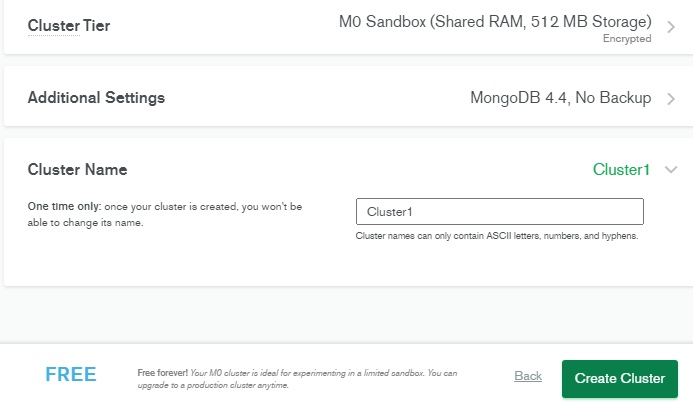




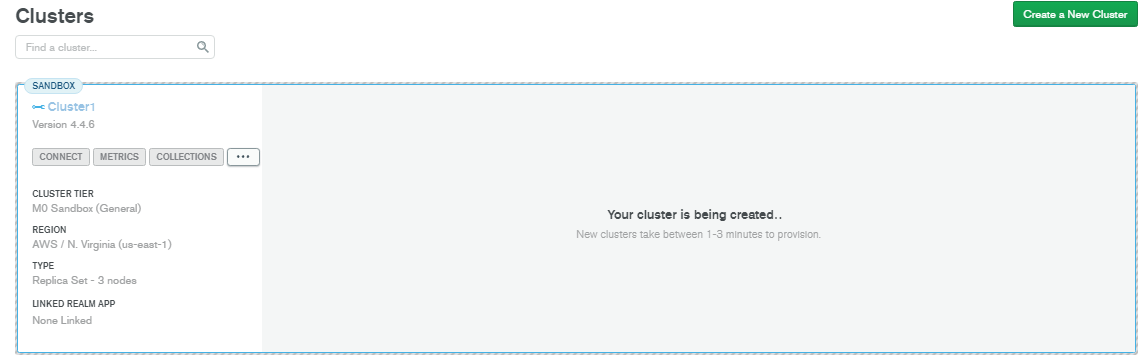


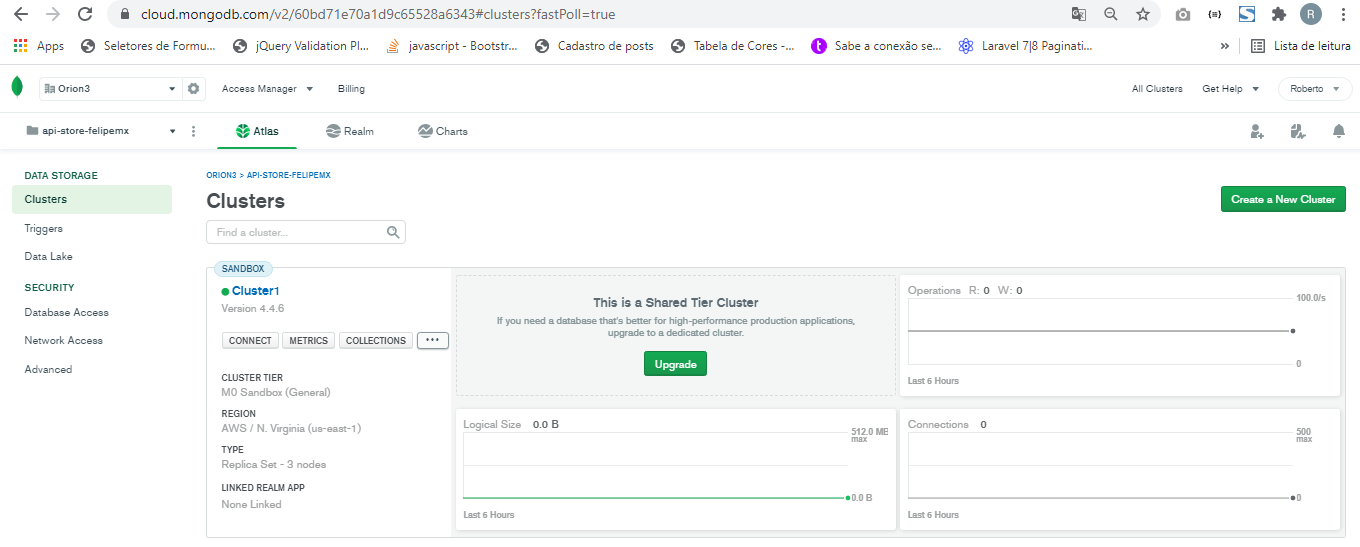
- Crie um cluster (FREE)



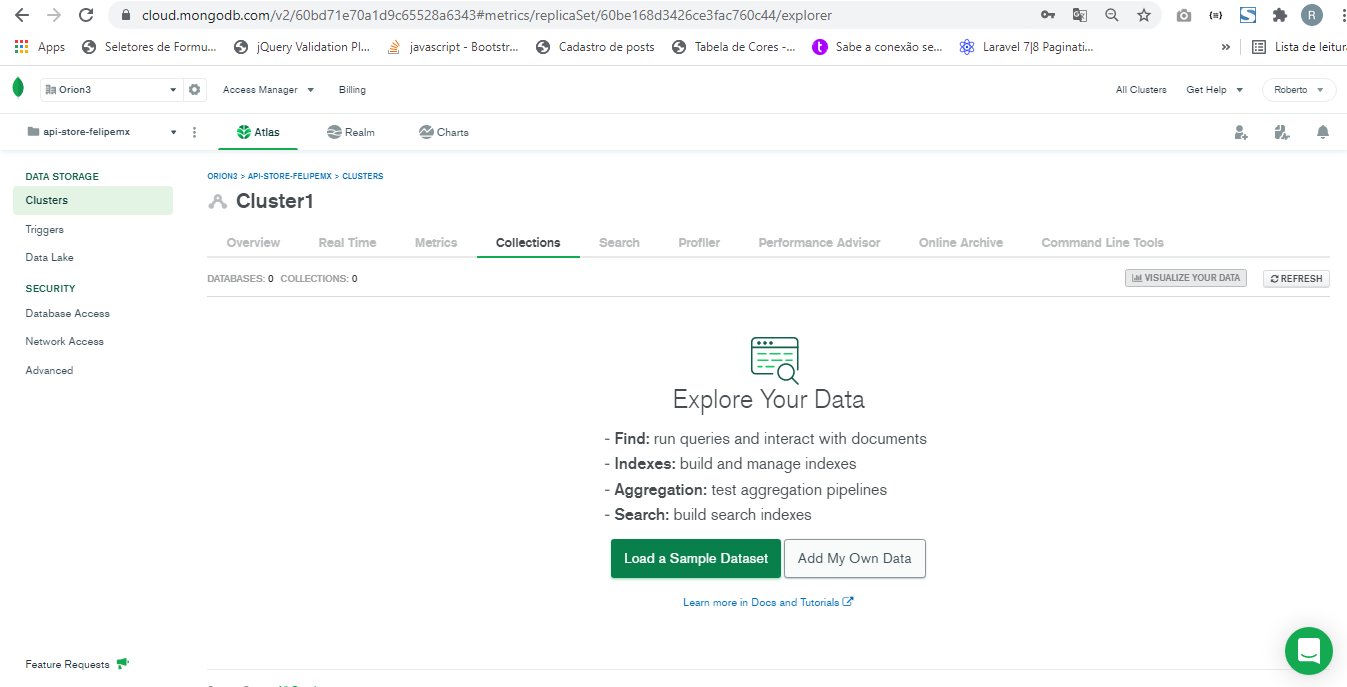


- Aguarde a criação do cluster (de 1 a 3 minutos).

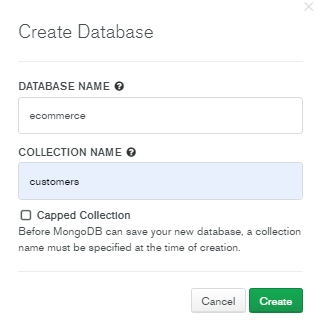




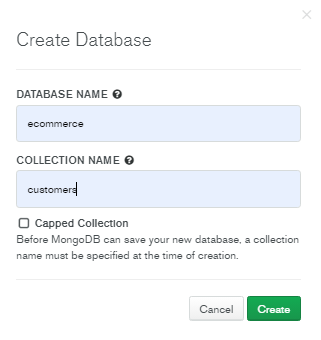
- Clique no botão "COLLECTIONS":

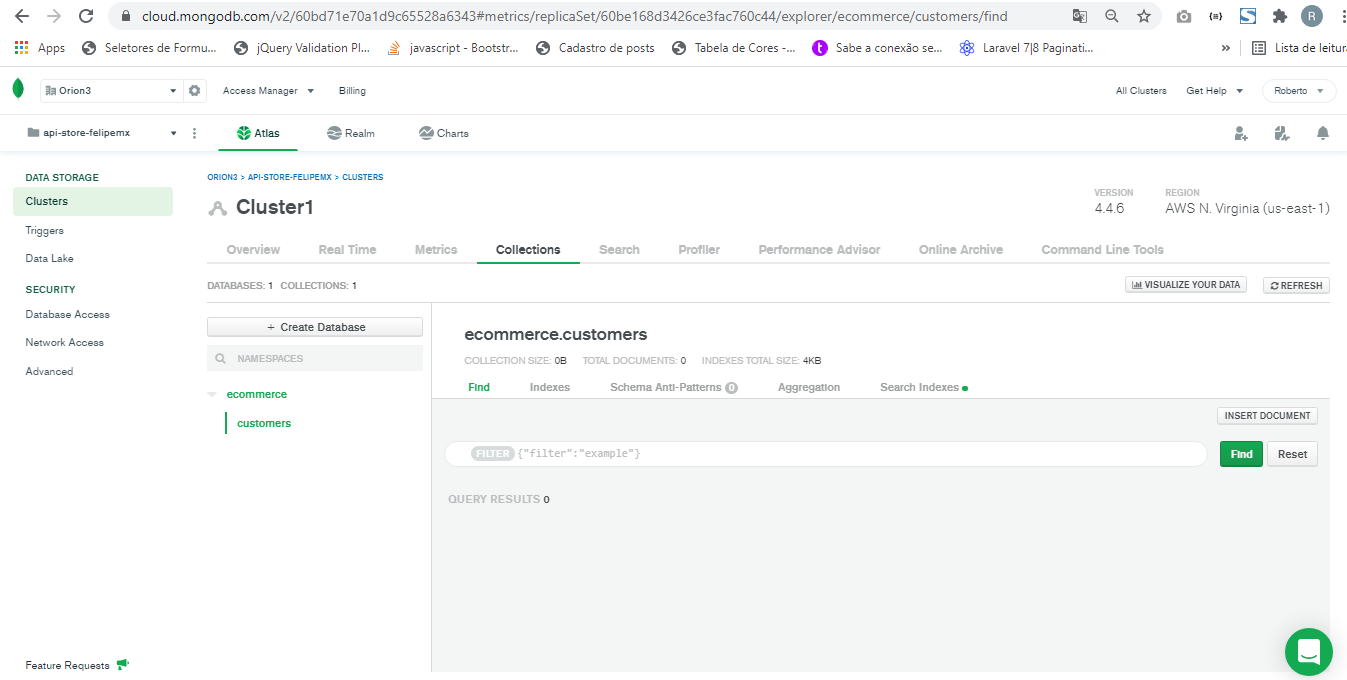


- Clique no botão "Add My Own Data":

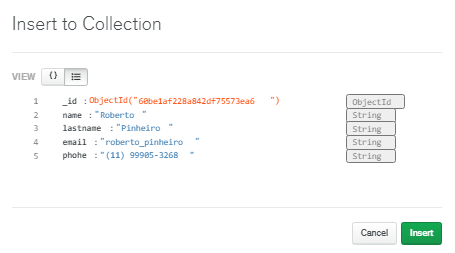


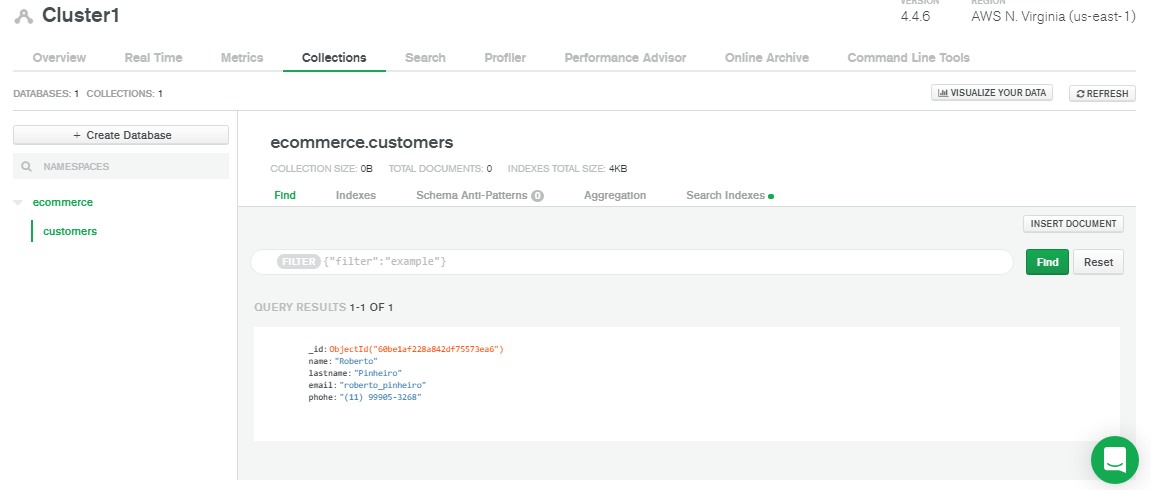
- Clique no botão "Create":



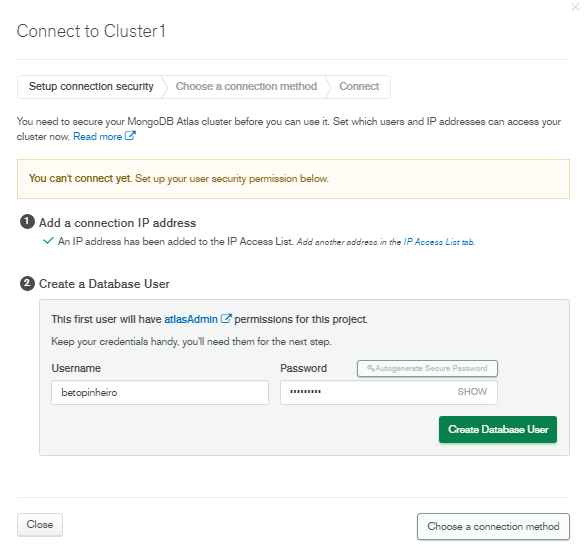


- Clique no botão "INSERT DOCUMENT"

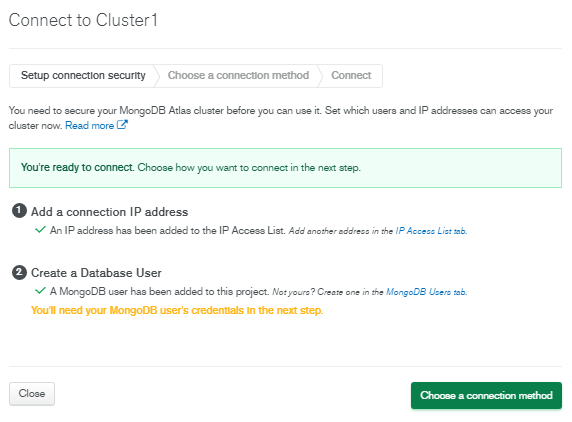




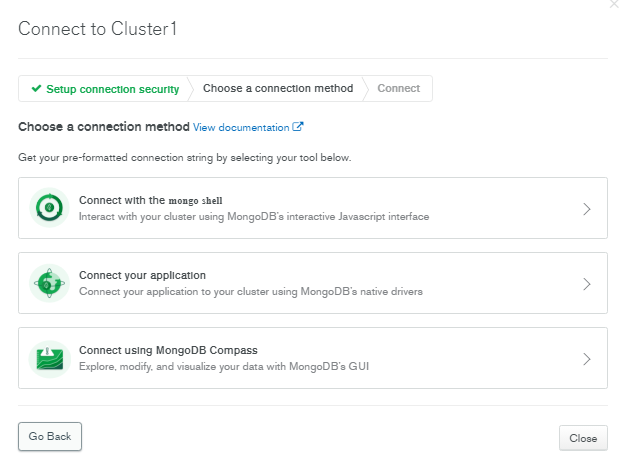
- Em "Cluster1" clique no botão "CONNECT":



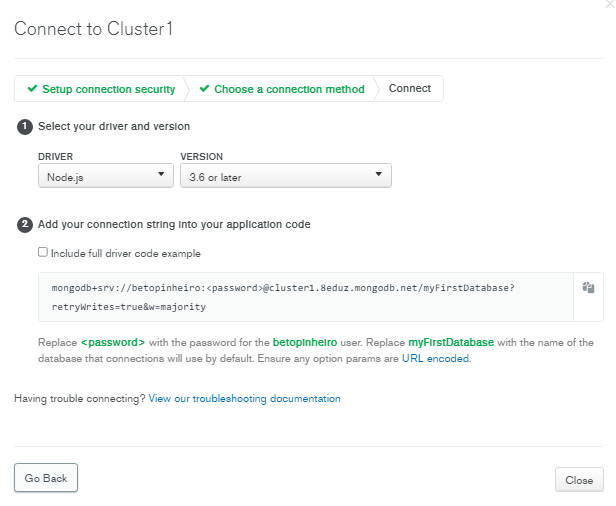
- Clique no botão "Create Database User":



- Clique no botão "Choose a connection method":



- Selecione a opção "Connect your application":



String de conexão:

mongodb+srv://betopinheiro:<password>@cluster1.8eduz.mongodb.net/myFirstDatabase?retryWrites=true&w=majority

Na string de conexão acima, substitua <password> pela senha e myFirstDatabase por ecommerce.

mongodb+srv://betopinheiro:abc123456@cluster1.8eduz.mongodb.net/ecommerce?retryWrites=true&w=majority

## Trabalhando em modo development (local - database: store-api)

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const customersRoutes = require('./routes/customersRoutes');

const productsRoutes = require('./routes/productsRoutes');

const ordersRoutes = require('./routes/ordersRoutes');

const app = express();

// Configurando mongoose

mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/store-api',

{

useNewUrlParser: true

})

.then(db => console.log('Base de dados conectada!'))

.catch(error => console.log(error));

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.use('/', customersRoutes());

app.use('/', productsRoutes());

app.use('/', ordersRoutes());

app.use(express.static('uploads'));

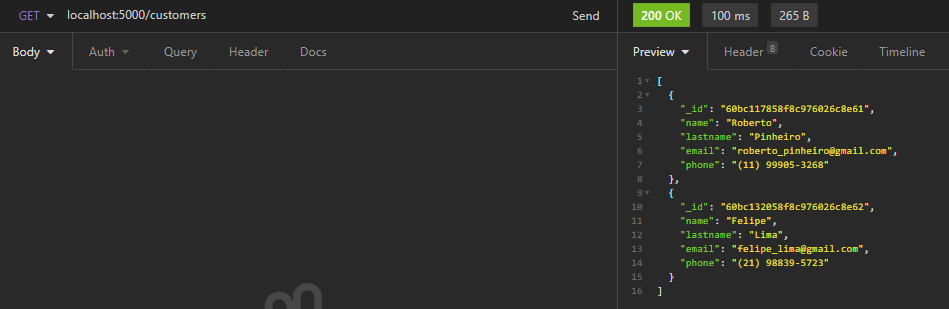
// server port

app.listen(process.env.PORT || 5000, () => {

console.log('Servidor web Express em execução!');

})

GET - localhost:5000/customers



GET - localhost:5000/products







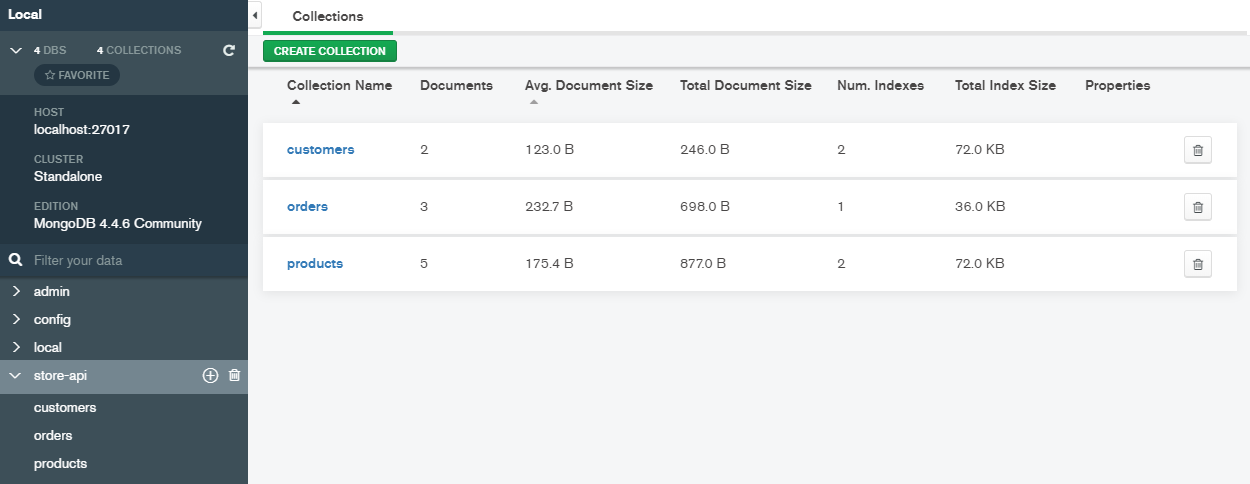
GET - localhost:5000/orders











## Trabalhando em modo development (local - database: ecommerce)

### Alterações no arquivo index.js

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

const customersRoutes = require('./routes/customersRoutes');

const productsRoutes = require('./routes/productsRoutes');

const ordersRoutes = require('./routes/ordersRoutes');

const app = express();

mongoose.connect("mongodb+srv://betopinheiro:abc123456@cluster1.8eduz.mongodb.net/ecommerce?retryWrites=true&w=majority",

{

useNewUrlParser: true

})

.then(db => console.log('Base de dados conectada!'))

.catch(error => console.log(error));

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.use('/', customersRoutes());

app.use('/', productsRoutes());

app.use('/', ordersRoutes());

app.use(express.static('uploads'));

// server port

app.listen(process.env.PORT || 5000, () => {

console.log('Servidor web Express em execução!');

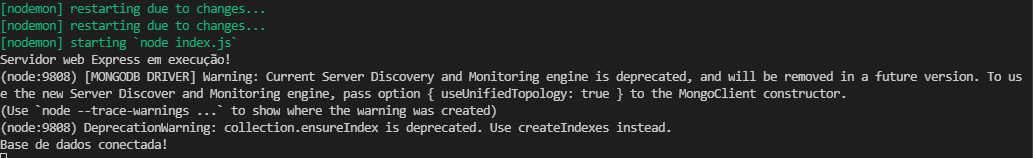
})

- Rode o servidor do Mongo:

mongod

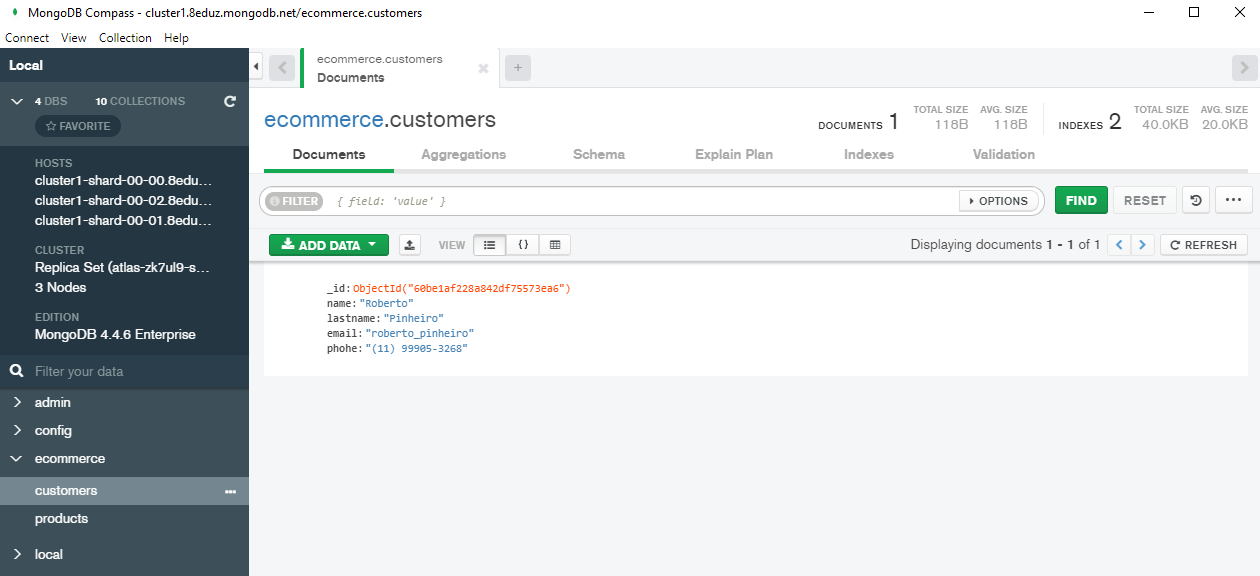
- Rode o servidor da aplicação:

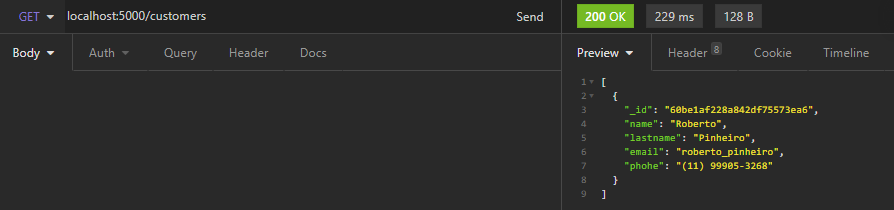
npm run dev

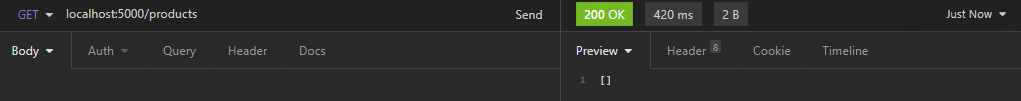


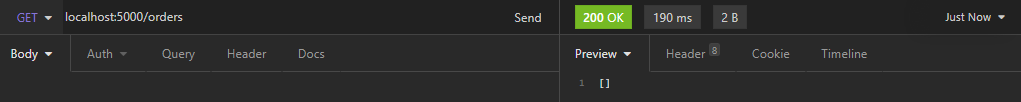
- Crie uma nova conexão no MongoDB Compass, com a string de conexão:

mongodb+srv://betopinheiro:abc123456@cluster1.8eduz.mongodb.net/ecommerce?retryWrites=true&w=majority









### Cadastrando clientes

{

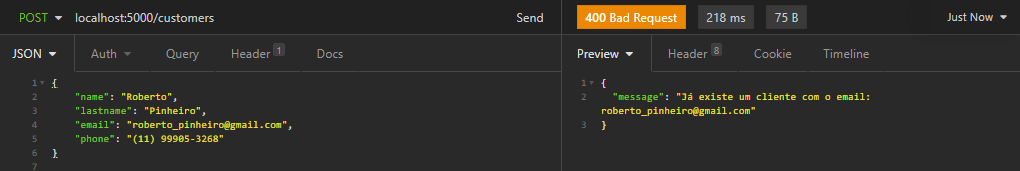
"name": "Roberto",

"lastname": "Pinheiro",

"email": "roberto\_pinheiro@gmail.com",

"phone": "(11) 99905-3268"

}



{

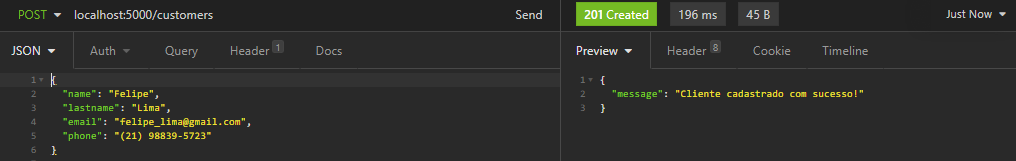
"name": "Felipe",

"lastname": "Lima",

"email": "felipe\_lima@gmail.com",

"phone": "(21) 98839-5723"

}



{

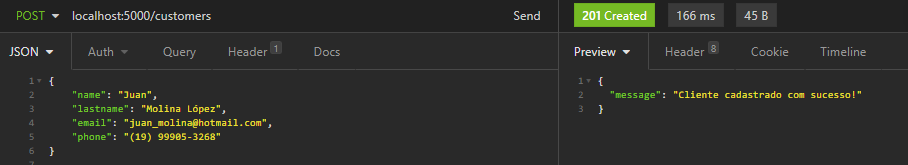
"name": "Juan",

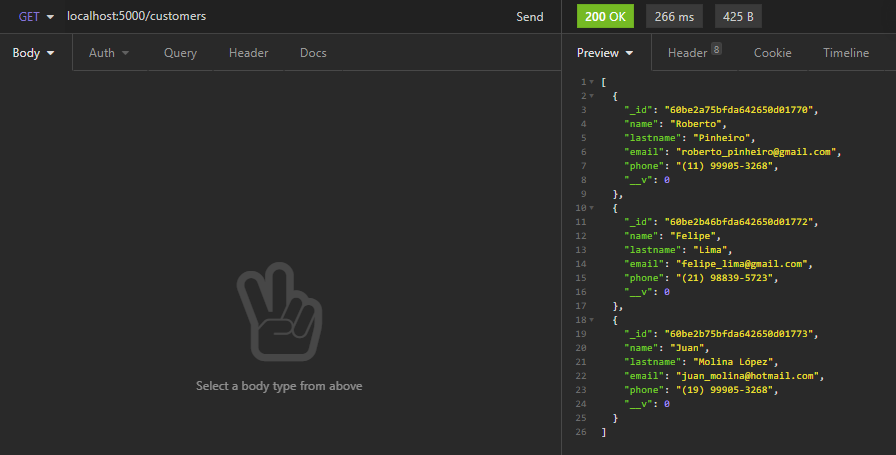
"lastname": "Molina López",

"email": "juan\_molina@hotmail.com",

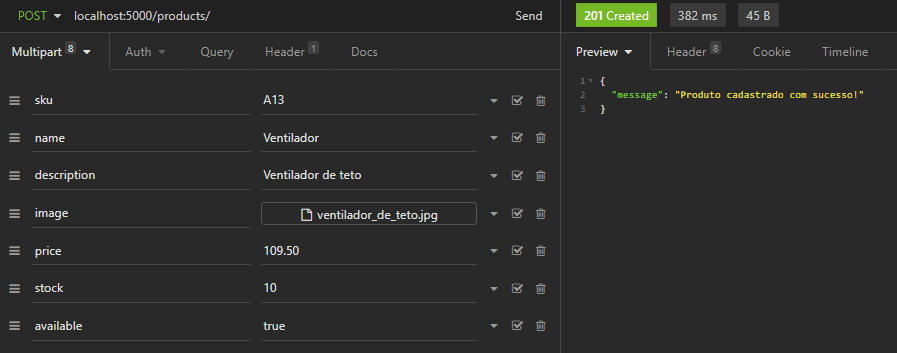
"phone": "(19) 99905-3268"

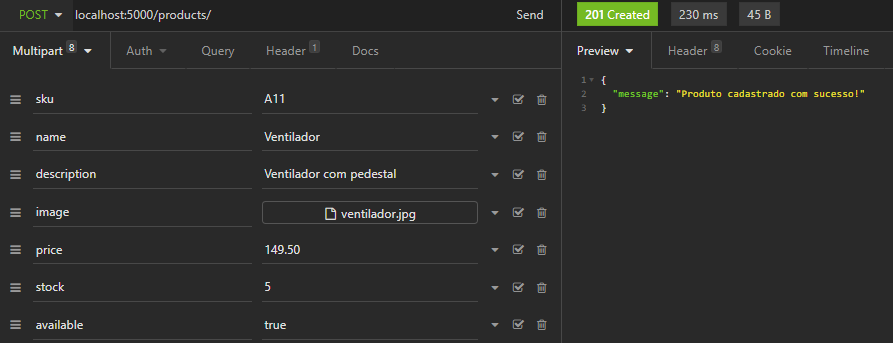
}

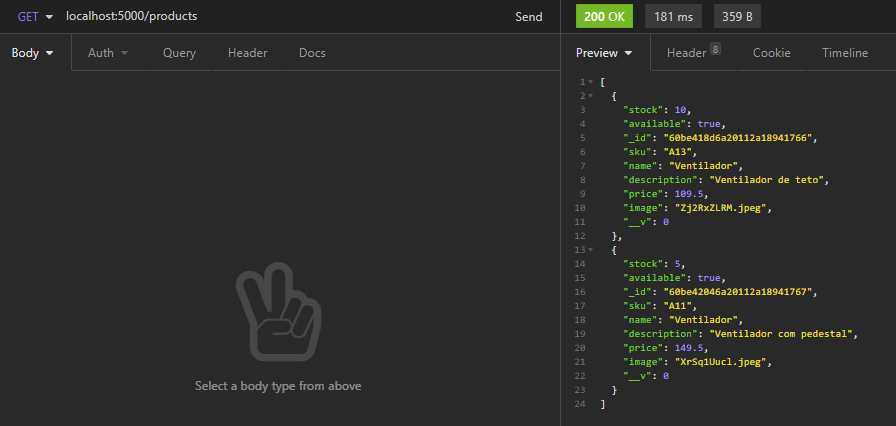




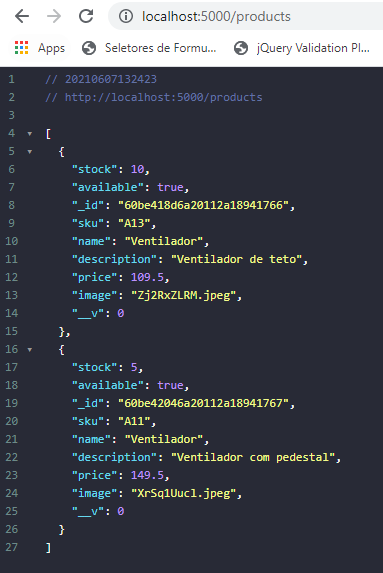
### Cadastrando produtos











### Instalando o pacote dotenv

npm install --save dotenv

## Deploy

- Adicione a pasta raiz do projeto um arquivo chamado ".gitignore":

**index.js**

const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');

const cors = require('cors');

// Rotas

const customersRoutes = require('./routes/customersRoutes');

const productsRoutes = require('./routes/productsRoutes');

const ordersRoutes = require('./routes/ordersRoutes');

const app = express();

app.get('/', (req, res) => {

res.send('Página inicial');

});

// Conexão com o mongoDB

// Configurando mongoose

mongoose.Promise = global.Promise;

require('dotenv').config();

// Conexão a base de dados

const uri = `mongodb+srv://${process.env.USER}:${process.env.PASSWORD}@cluster1.8eduz.mongodb.net/${process.env.DBNAME}?retryWrites=true&w=majority`;

mongoose.connect(uri, {useNewUrlParser: true, useUnifiedTopology: true})

.then(() => {

console.log('Base de dados conectada!');

}).catch((error) => {

console.log(error);

});

// Habilitar body-parser

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

// Habilitar cors

app.use(cors());

app.use('/', customersRoutes());

app.use('/', productsRoutes());

app.use('/', ordersRoutes());

app.use(express.static('uploads'));

// server port

const port = process.env.PORT || 5000;

app.listen(port, () => {

console.log(`Server running at port ` + port);

});

**.env**

USER=betopinheiro

PASSWORD=abc123456

DBNAME=ecommerce

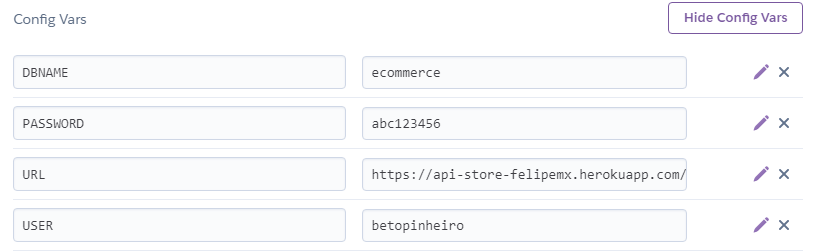
URL=https://api-store-felipemx.herokuapp.com/

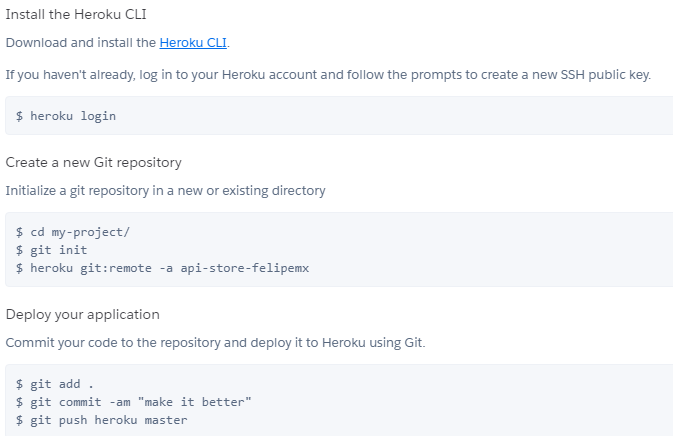
**.gitignore**

/node\_modules

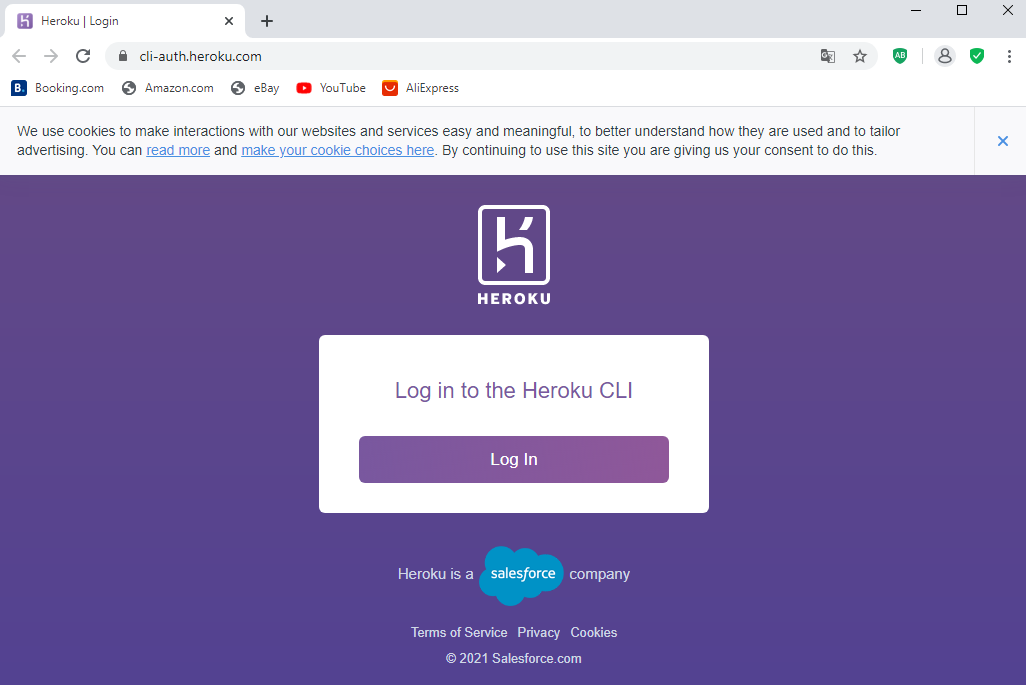
.env

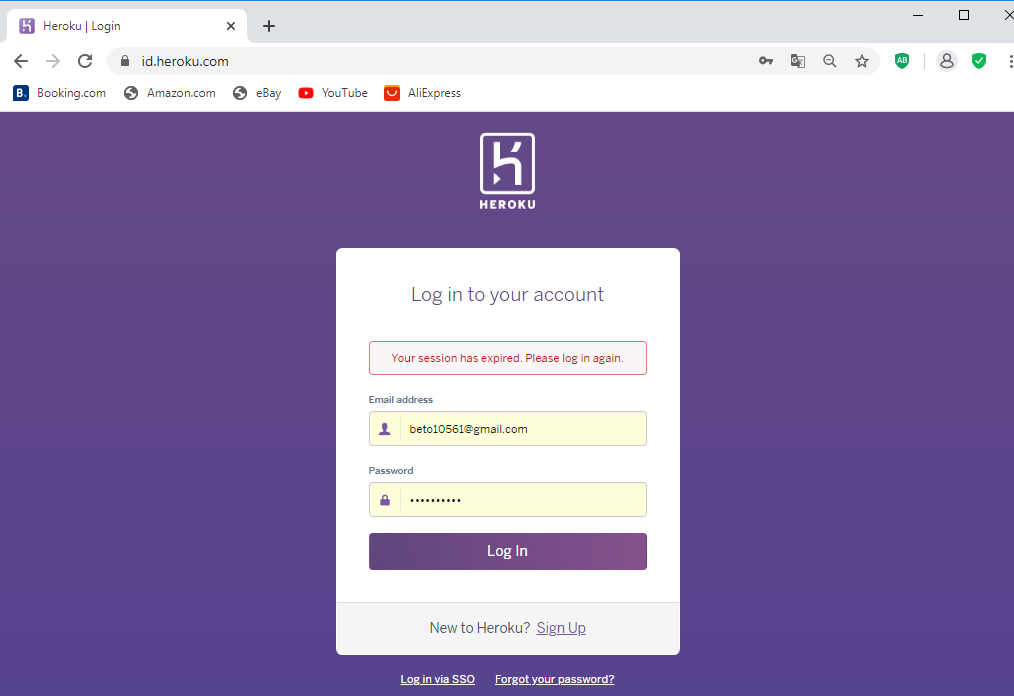
### Variáveis de ambiente no Heroku



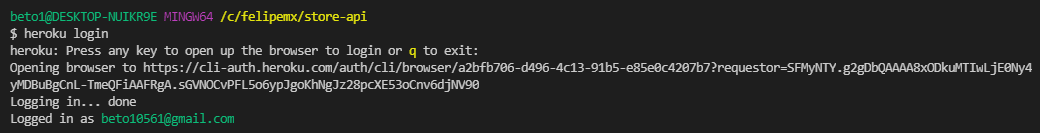


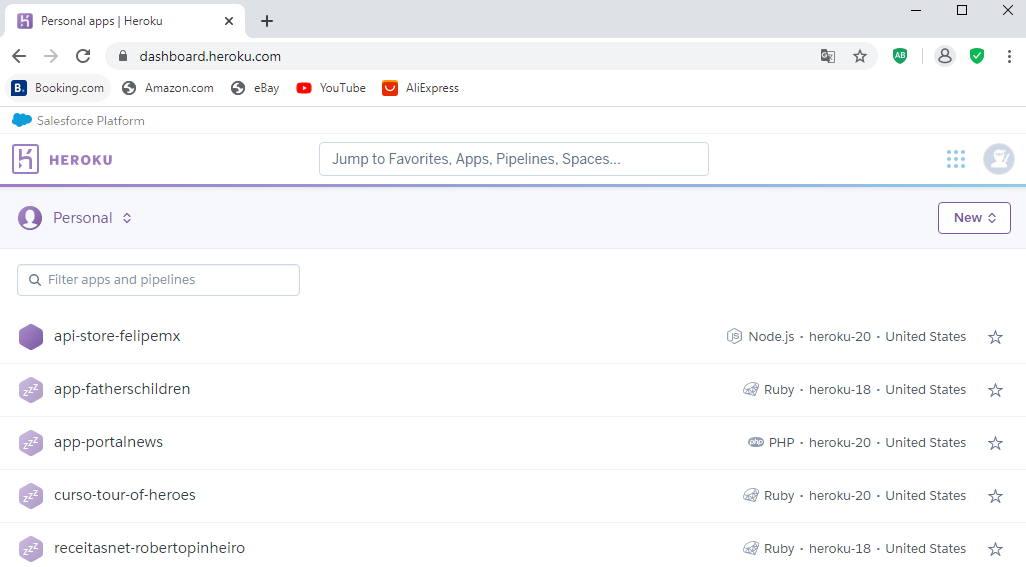
heroku login

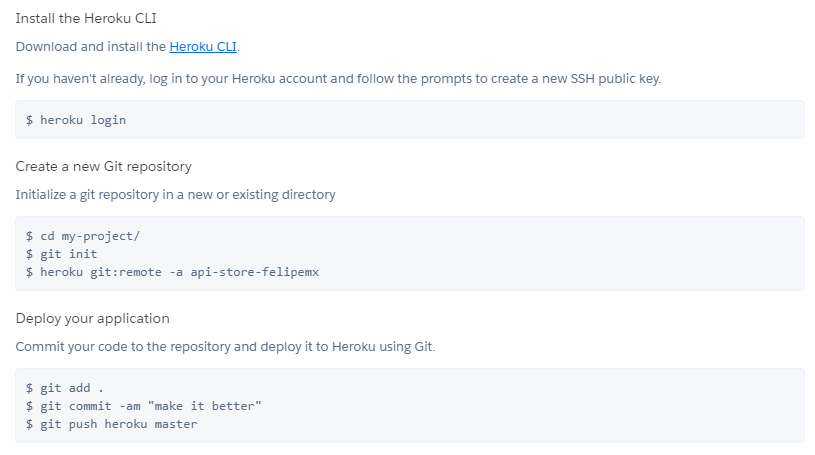




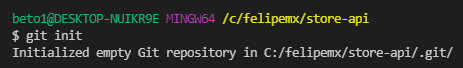




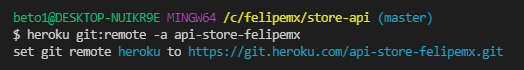




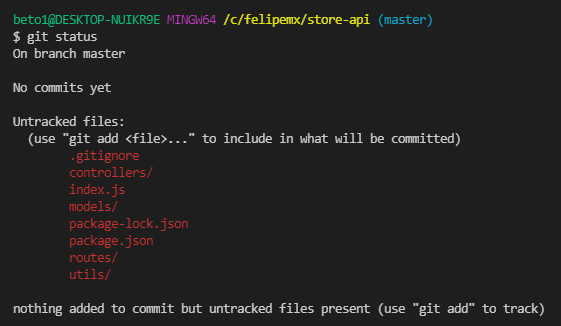
git init



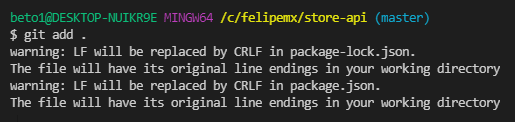
heroku git:remote -a api-store-felipemx



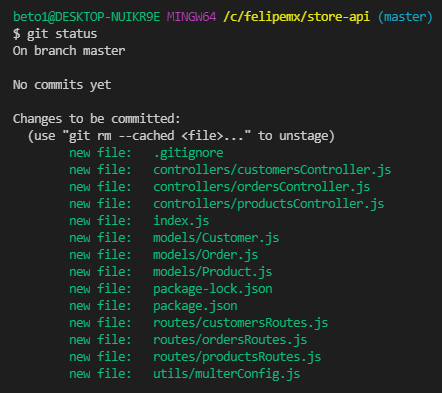
git status



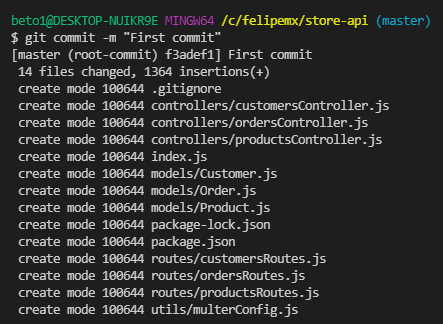
git add .



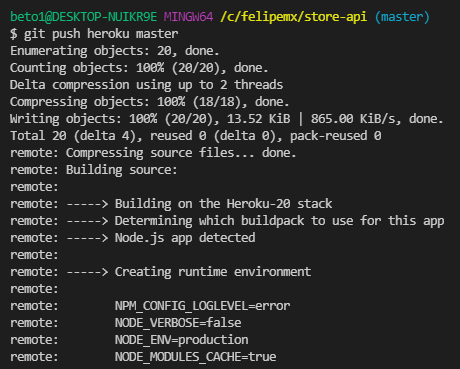
git status



git commit -m "First commit"



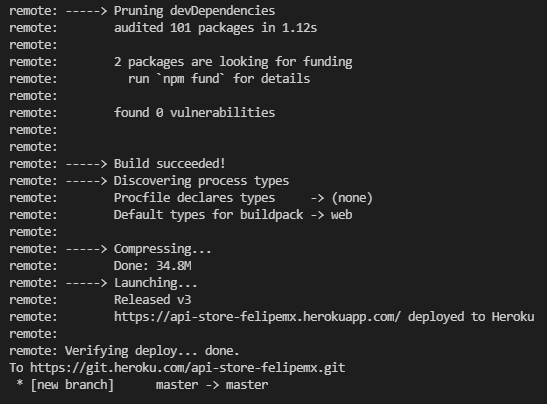
git push heroku master



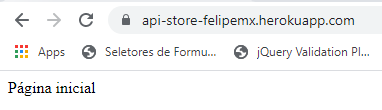
.

.

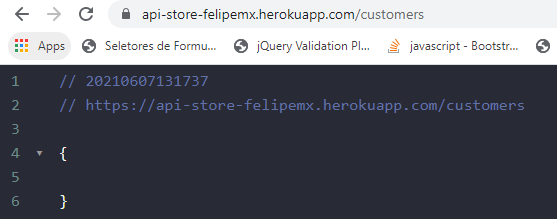
.

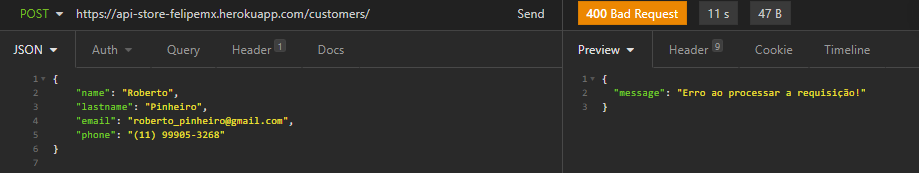


https://api-store-felipemx.herokuapp.com/



https://api-store-felipemx.herokuapp.com/customers





???????????