Criando API's com Node.JS
balta.io (André Baltieri)

https://www.youtube.com/watch?v=wDWdqlYxfcw&list=PLHlHvK2lnJndvvycjBqQAbgEDqXxKLoqn

Resumo do curso feito por Roberto Pinheiro

Github - Código fonte:

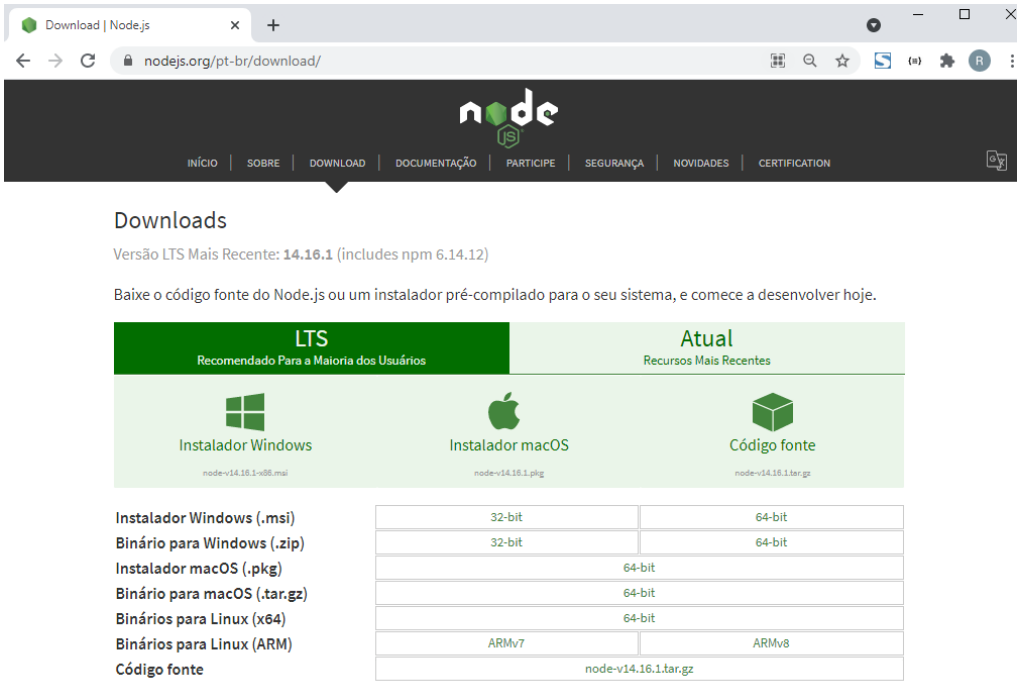
https://github.com/balta-io/1972

Aula 01 - Instalação Node, NPM e VS Code

- Dentro de C:\ crie uma pasta chamada balta e dentro dela crie uma subpasta chamada nodejs. Dentro dela crie uma subpasta chamada node-str. Essa será a pasta do nosso projeto inicial.

- Baixe e instale o Node:

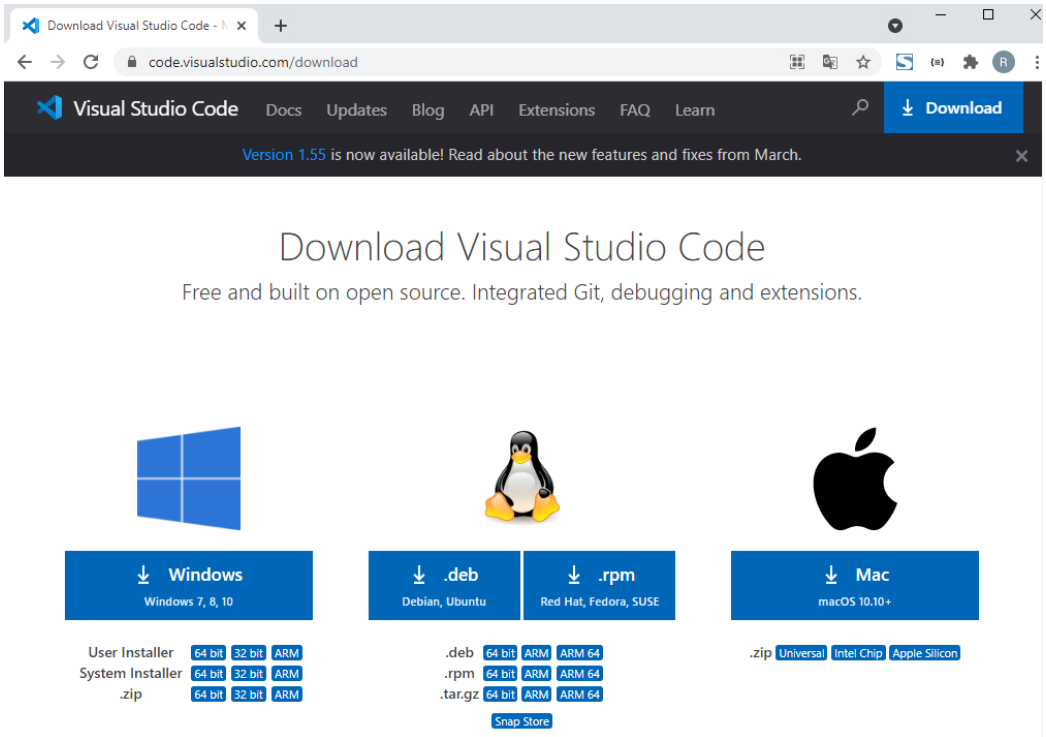
https://nodejs.org/pt-br/download/



- Baixe a versão LTS

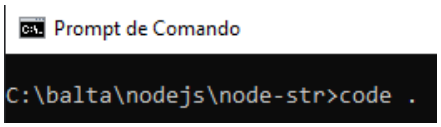
- Baixe e instale o Visual Studio Code

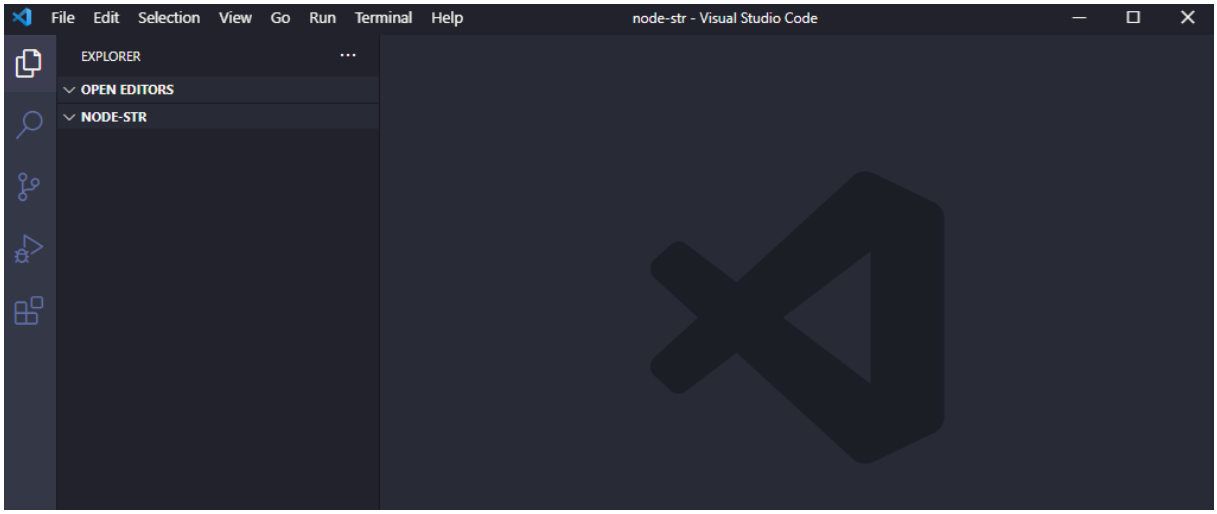
https://code.visualstudio.com/download



- Na pasta do projeto, entre com:

code .





- Abra um terminal Powershell e entre com os seguintes comandos:

node --version

```
PS C:\balta\nodejs\node-str> node --version
v14.15.3
```

npm --version

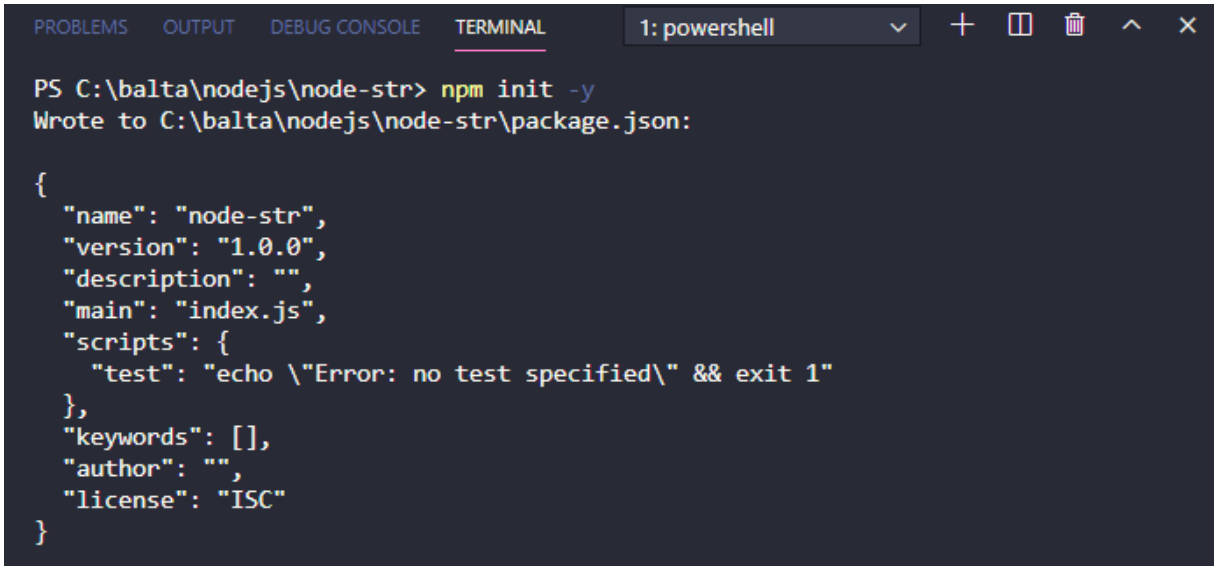
```
PS C:\balta\nodejs\node-str> npm --version
6.14.9
```

Aula 02 - npm init e instalação dos pacotes

Inicializando uma aplicação do Node

- No terminal, na pasta do projeto, entre com o comando:

npm init -y



- É criado um arquivo chamado package.json

package.json

```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

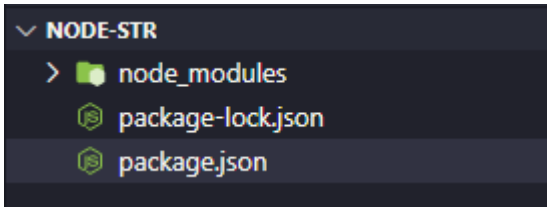
Instalando pacotes básicos

O comando a seguir irá instalar os pacotes:

- http
- express
- debug

npm install http express debug --save

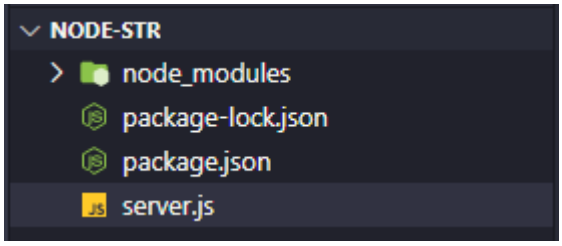
```
PS C:\balta\nodejs\node-str> npm install http express debug --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.

+ express@4.17.1
+ debug@4.3.1
+ http@0.0.1-security
added 59 packages from 38 contributors and audited 59 packages in 23.813s
found 0 vulnerabilities
```



package.json

```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "debug": "^4.3.1",
    "express": "^4.17.1",
    "http": "0.0.1-security"
  }
}
```

- Crie um arquivo chamado server.js



server.js

```
'use strict'

console.log('Testando...');
```

```
PS C:\balta\nodejs\node-str> node ./server.js
Testando...
```

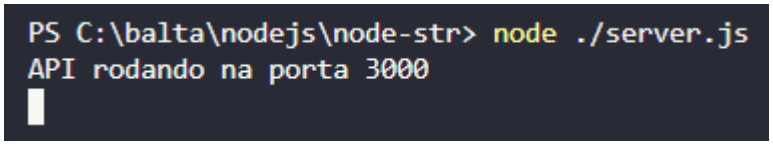
Aula 03 - Criando um servidor web

server.js

```
'use strict'

const http = require('http');
const debug = require('debug')('nodestr: server');
const express = require('express');

const app = express();
const port = 3000;
app.set('port', port);

const server = http.createServer(app);
const router = express.Router();

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.1"
    });
});

app.use('/', route);

server.listen(port);
console.log('API rodando na porta ' + port);
```

- Rode o servidor:

node ./server.js

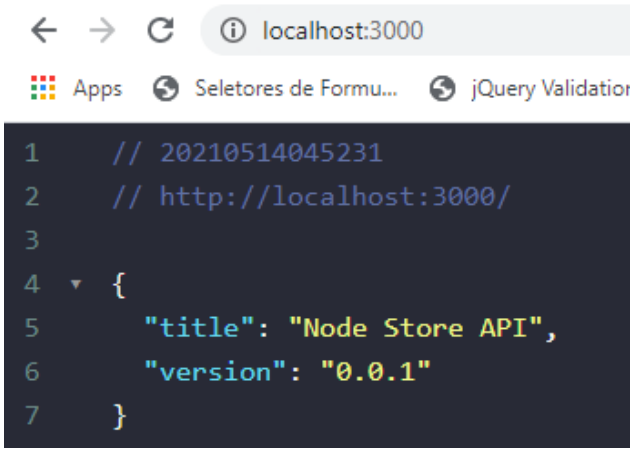


- Instale a extensão JSON Viewer para Google Chrome.

- Selecione o tema Dracula:

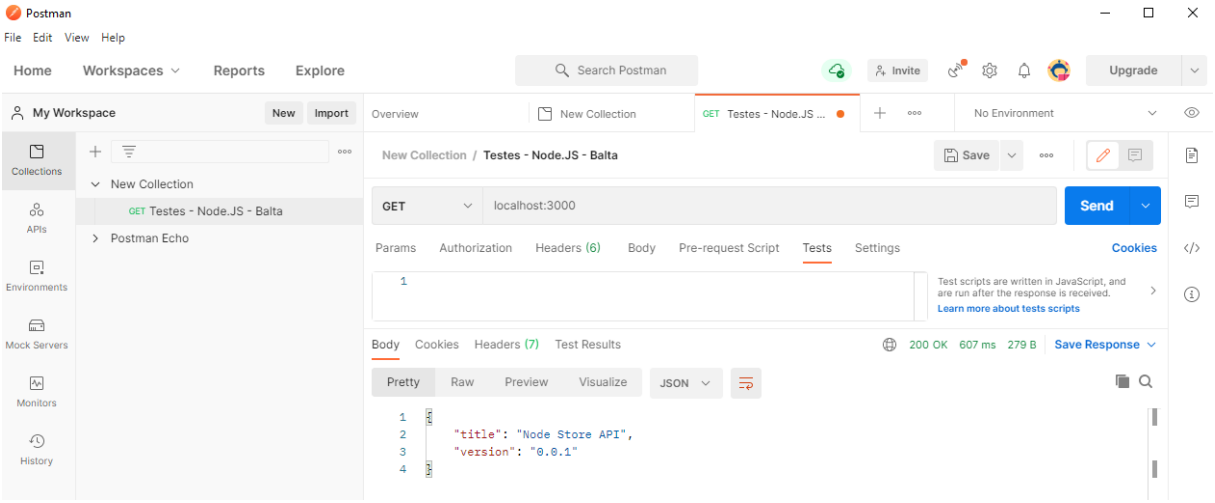
- No browser:

http://localhost:3000/



- Baixe e instale o Postman. Ele será utilizado para simular requisições.

Usando o Postman:



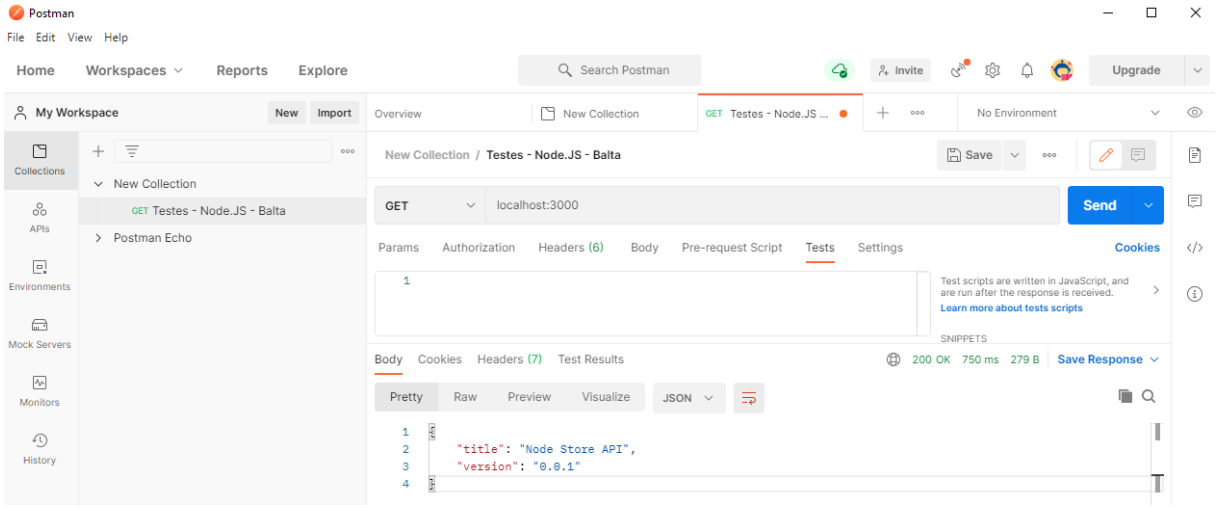
Aula 04 - Normalizando a porta

server.js

```
'use strict'

const http = require('http');
const debug = require('debug')('nodestr: server');
const express = require('express');

const app = express();
const port = normalizePort(process.env.PORT) || '3000';
app.set('port', port);

const server = http.createServer(app);
const router = express.Router();

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.1"
    });
});

app.use('/', route);

server.listen(port);
console.log('API rodando na porta ' + port);

function normalizePort(val){
    const port = parseInt(val, 10);

    if(isNaN(port)){
        return val;
    }

    if(port >= 0){
        return port;
    }

    return false;
}
```

node ./server.js

```
PS C:\balta\nodejs\node-str> node ./server.js
API rodando na porta 3000
```



Aula 05 - Gerenciando erros do servidor

server.js

```
'use strict'

const http = require('http');
const debug = require('debug')('nodestr: server');
const express = require('express');

const app = express();
const port = normalizePort(process.env.PORT) || '3000';
app.set('port', port);

const server = http.createServer(app);
const router = express.Router();

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.1"
    });
});

app.use('/', route);

server.listen(port);
server.on('error', onError);
console.log('API rodando na porta ' + port);

function normalizePort(val){
    const port = parseInt(val, 10);

    if(isNaN(port)){
        return val;
    }

    if(port >= 0){
        return port;
    }

    return false;
}

function onError(error) {
    if (error.syscall !== 'listen') {
      throw error;
    }
```

```
  const bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}
```

Criando um erro

- No Visual Studio Code, abra e execute o servidor em dois terminais:

node ./server.js

```
PS C:\balta\nodejs\node-str> node ./server.js
API rodando na porta 3000
Pipe 3000 is already in use
```

Aula 06 - Iniciando o Debug

server.js

```
'use strict'

const http = require('http');
const debug = require('debug')('nodestr: server');
const express = require('express');

const app = express();
const port = normalizePort(process.env.PORT) || '3000';
app.set('port', port);

const server = http.createServer(app);
const router = express.Router();

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.1"
    });
});

app.use('/', route);

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);
console.log('API rodando na porta ' + port);

function normalizePort(val){
    const port = parseInt(val, 10);

    if(isNaN(port)){
        return val;
    }

    if(port >= 0){
        return port;
    }

    return false;
}
```

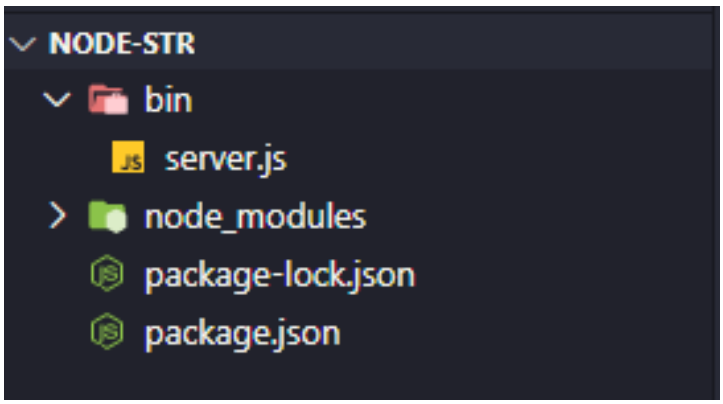
```
function onError(error) {
    if (error.syscall !== 'listen') {
      throw error;
    }

    const bind = typeof port === 'string'
      ? 'Pipe ' + port
      : 'Port ' + port;

    switch (error.code) {
      case 'EACCES':
        console.error(bind + ' requires elevated privileges');
        process.exit(1);
        break;
      case 'EADDRINUSE':
        console.error(bind + ' is already in use');
        process.exit(1);
        break;
      default:
        throw error;
    }
  }

  function onListening() {
    const addr = server.address();
    const bind = typeof addr === 'string'
      ? 'pipe ' + addr
      : 'port ' + addr.port;
    debug('Listening on ' + bind);
  }
```

node ./server.js

```
PS C:\balta\nodejs\node-str> node ./server.js
API rodando na porta 3000
```

Aula 07 - Separando o servidor

- No diretório raiz da aplicação, crie uma pasta chamada bin. E mova o arquivo server.js para dentro dela.



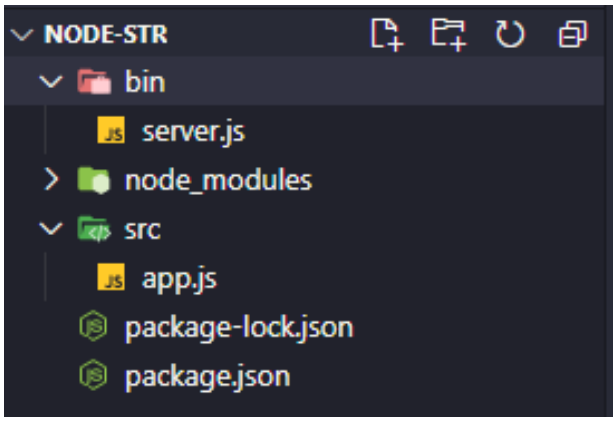
- No arquivo bin/server.js altere a linha da const app e remova o trecho de código especificado:

bin/server.js

```
const app = require('../src/app');
const http = require('http');
const debug = require('debug')('balta:server');

const express = require('express');

~~const app = express();~~

const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

const server = http.createServer(app);

~~const router = express.Router();~~

~~const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.1"
    });
});~~

~~app.use('/', route);~~

server.listen(port);
```

```
server.on('error', onError);
server.on('listening', onListening);
console.log('API rodando na porta ' + port);

function normalizePort(val) {
  const port = parseInt(val, 10);

  if (isNaN(port)) {
    return val;
  }

  if (port >= 0) {
    return port;
  }

  return false;
}

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  const bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

function onListening() {
  const addr = server.address();
  const bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

- No diretório raiz da aplicação, crie uma nova pasta chamada src e dentro dela insira um arquivo chamado app.js:



src/app.js

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

const app = express();
const router = express.Router();

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.1"
    });
});

app.use('/', route);

module.exports = app;
```

node ./bin/server.js

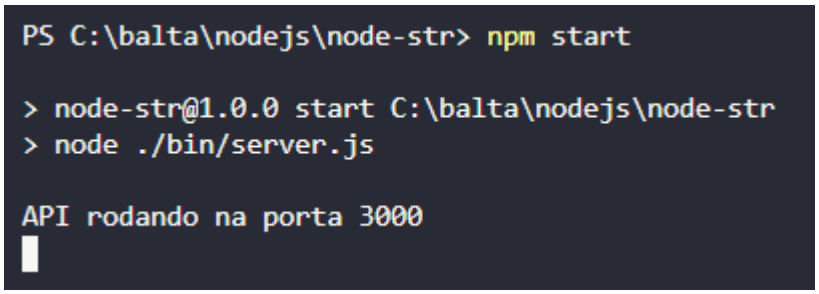
```
PS C:\balta\nodejs\node-str> node ./bin/server.js
API rodando na porta 3000
```

Aula 08 - Configurando o npm start

package.json

```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node ./bin/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "debug": "^4.1.1",
    "express": "^4.17.1",
    "http": "0.0.0"
  }
}
```

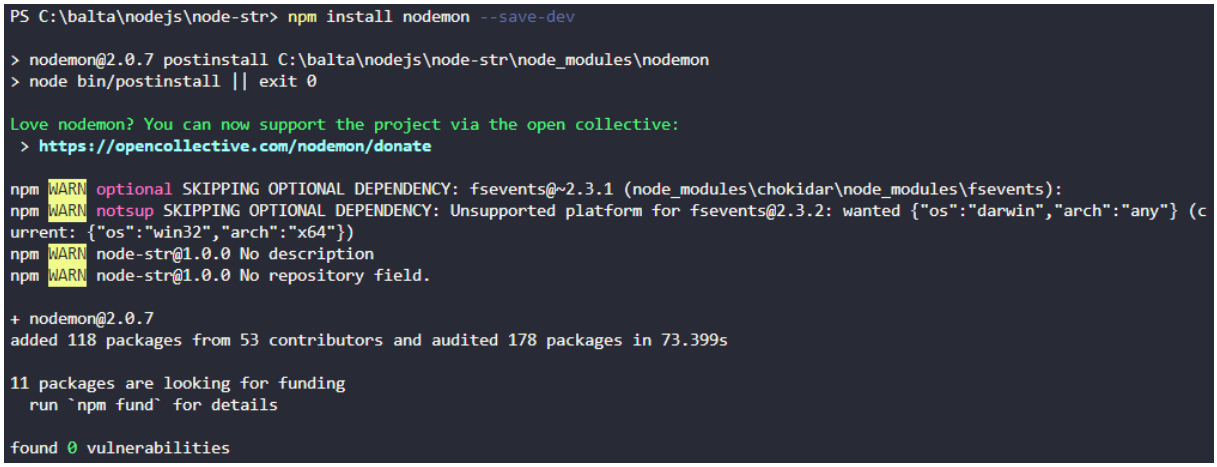
npm start



Aula 09 - Nodemon

Instalando o pacote nodemon

npm install nodemon --save-dev



package.json

```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node ./bin/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "debug": "^4.3.1",
    "express": "^4.17.1",
    "http": "0.0.1-security"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

nodemon ./bin/server.js

```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
```

- Fazendo uma alteração no arquivo src/app.js, ao salvá-lo:

src/app.js

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

const app = express();
const router = express.Router();

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.2"
    });
});

app.use('/', route);

module.exports = app;
```

```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
[nodemon] restarting due to changes...
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
```

Aula 10 - CRUD Rest

Instalando o pacote body-parser

npm install body-parser --save

```
PS C:\balta\nodejs\node-str> npm install body-parser --save
npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (c
urrent: {"os":"win32","arch":"x64"})

+ body-parser@1.19.0
updated 1 package and audited 179 packages in 10.138s

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

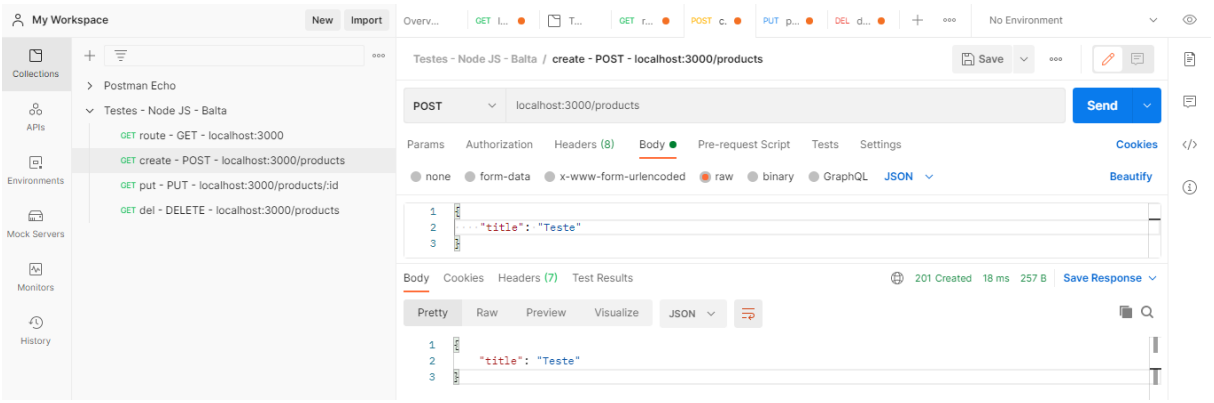
src/app.js

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

const app = express();
const router = express.Router();

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

const route = router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.2"
    });
});

const create = router.post('/', (req, res, next) => {
    res.status(201).send(req.body);
});

app.use('/', route);
app.use('/products', create);

module.exports = app;
```

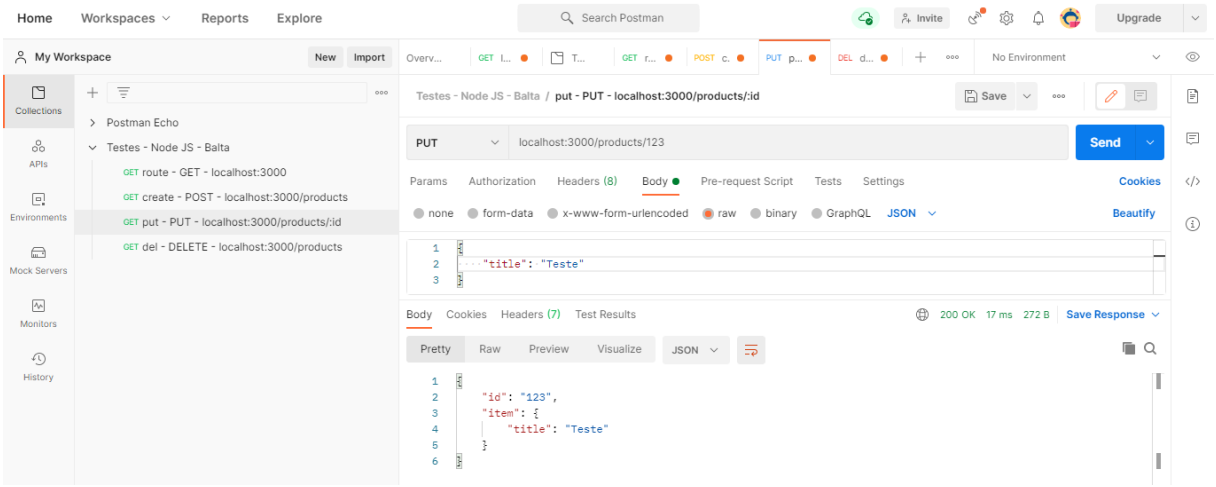
src/app.js

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

const app = express();
const router = express.Router();

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

const route = router.get('/', (req, res, next) => {
   res.status(200).send({
      title: "Node Store API",
      version: "0.0.1"
   });
});

const create = router.post('/', (req, res, next) => {
   res.status(201).send(req.body);
});

const put = router.put('/:id', (req, res, next) => {
   const id = req.params.id;
   res.status(200).send({
      id: id,
      item: req.body
   });
});

const del = router.delete('/', (req, res, next) => {
   res.status(200).send(req.body);
});

app.use('/', route);
app.use('/products', create);
app.use('/products', put);
app.use('/products', del);

module.exports = app;
```

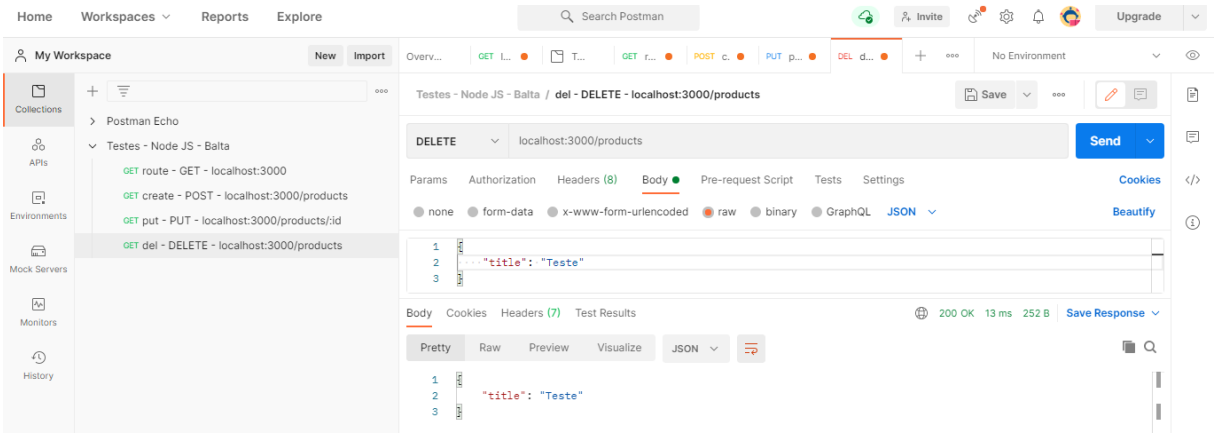
create → método POST



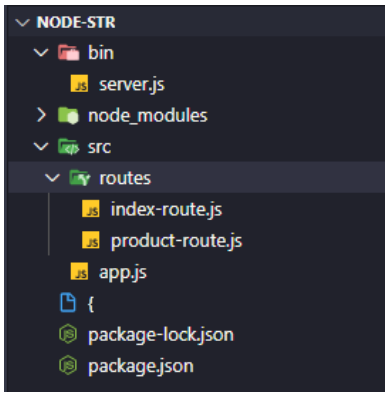
put → método PUT



del → método DELETE



Aula 11 - Rotas



src/routes/index-route.js

```
const express = require('express');
const router = express.Router();

router.get('/', (req, res, next) => {
    res.status(200).send({
        title: "Node Store API",
        version: "0.0.2"
    });
});

module.exports = router;
```

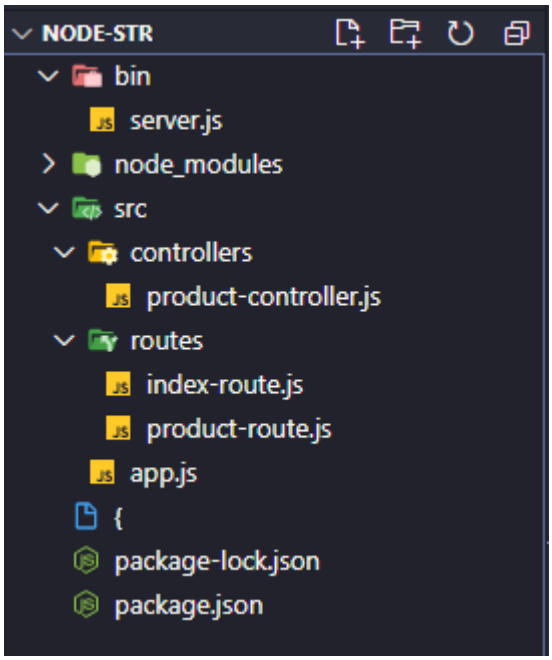
src/routes/product-route.js

```
const express = require('express');
const router = express.Router();

router.post('/', (req, res, next) => {
    res.status(201).send(req.body);
});

router.put('/:id', (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
});

router.delete('/', (req, res, next) => {
    res.status(200).send(req.body);
});

module.exports = router;
```

src/app.js

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

const app = express();
const router = express.Router();

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);

module.exports = app;
```

Aula 12 - Controllers



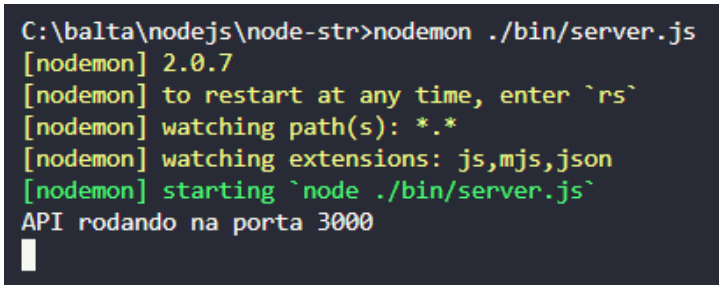
src/controllers/product-controller.js

```
'use strict';

exports.post = (req, res, next) => {
    res.status(201).send(req.body);
};

exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

src/routes/product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/', controller.delete);

module.exports = router;
```

nodemon ./bin/server.js

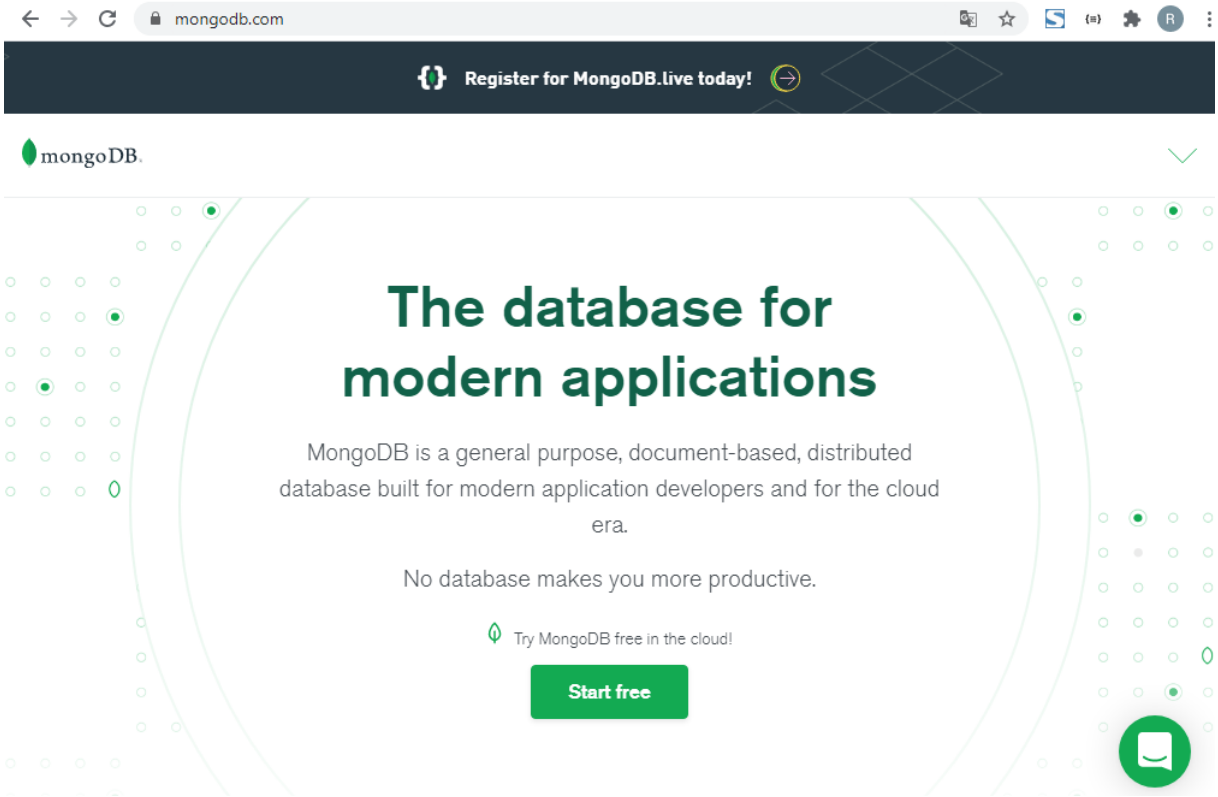


- Faça os testes com o Postman para verificar se tudo continua funcionando normalmente.

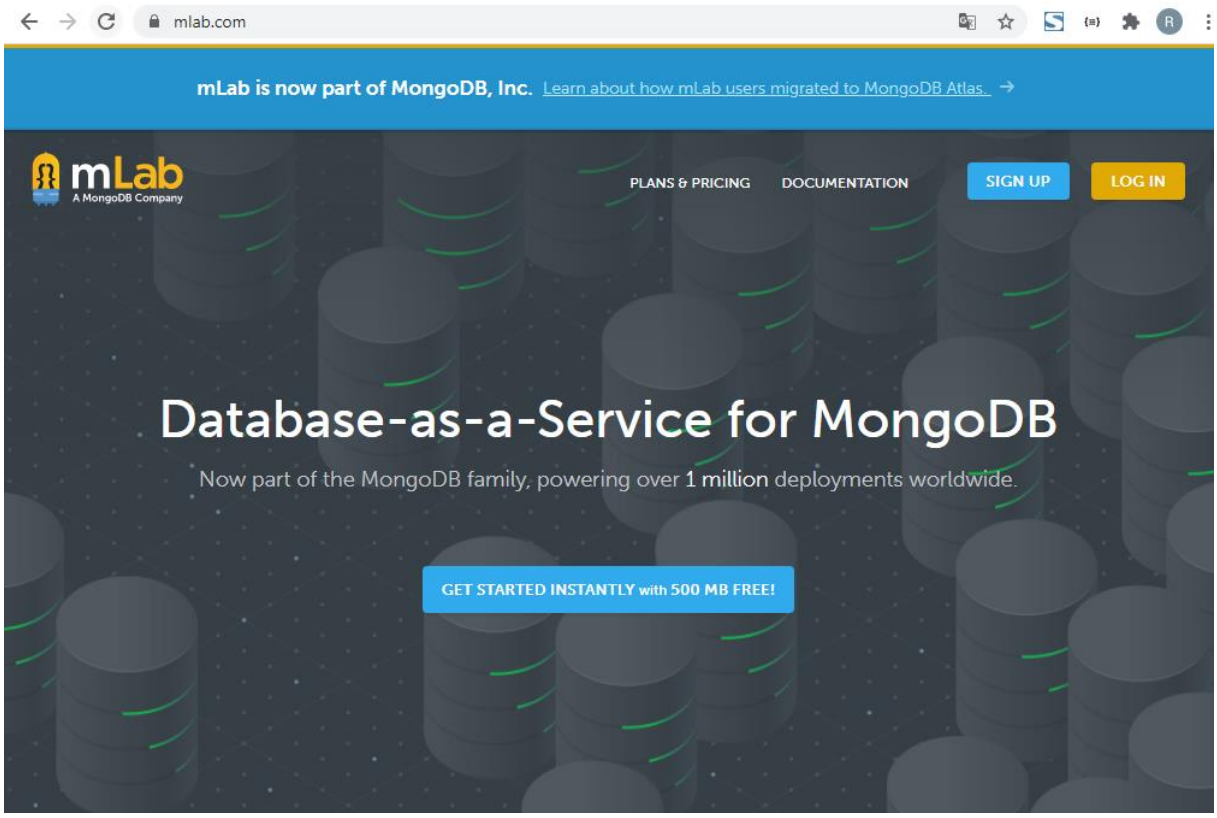
Aula 13 - MongoDB Setup

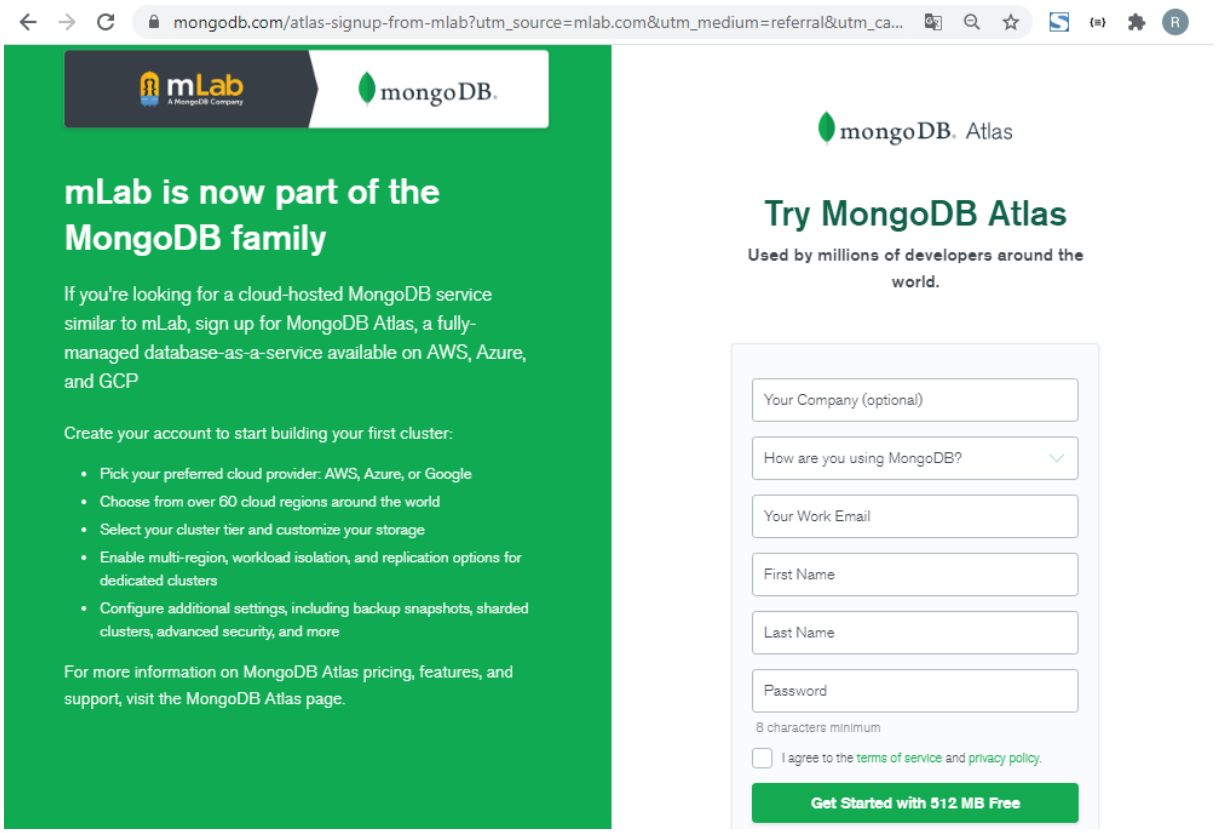
- Baixe e instale o MongoDB.

www.mongodb.com



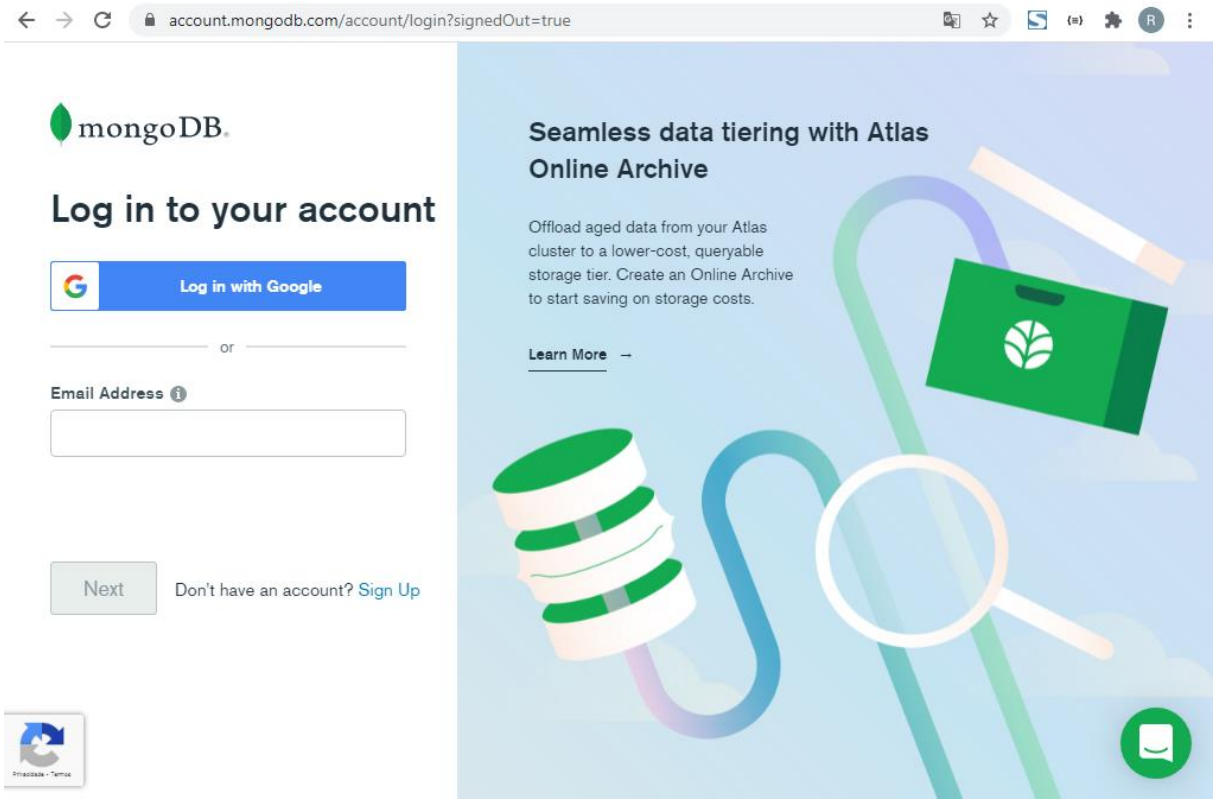
- Acesse https://mlab.com e abra uma conta (SIGN UP).





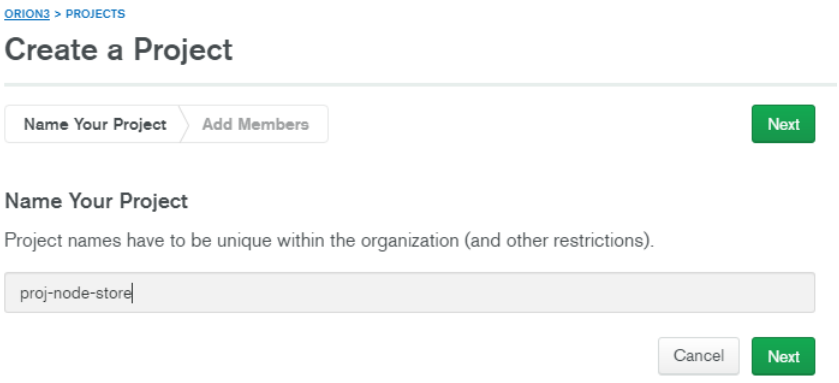
- Faça o login:

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

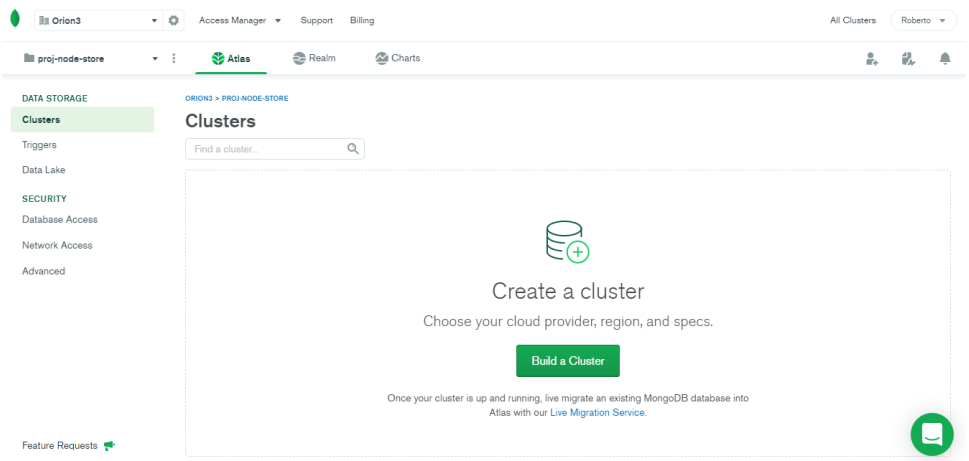


- Crie uma organização com nome: Orion3

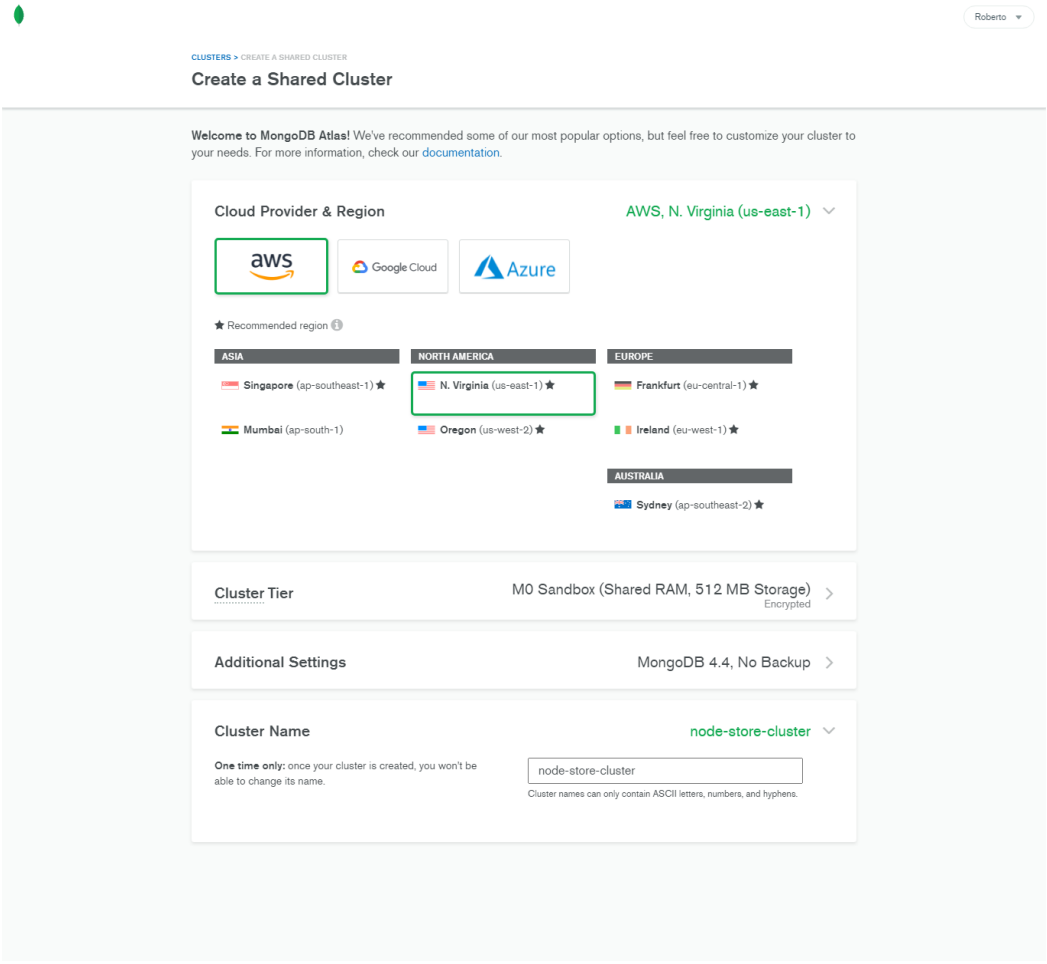
- Crie um novo projeto chamado: proj-node-store



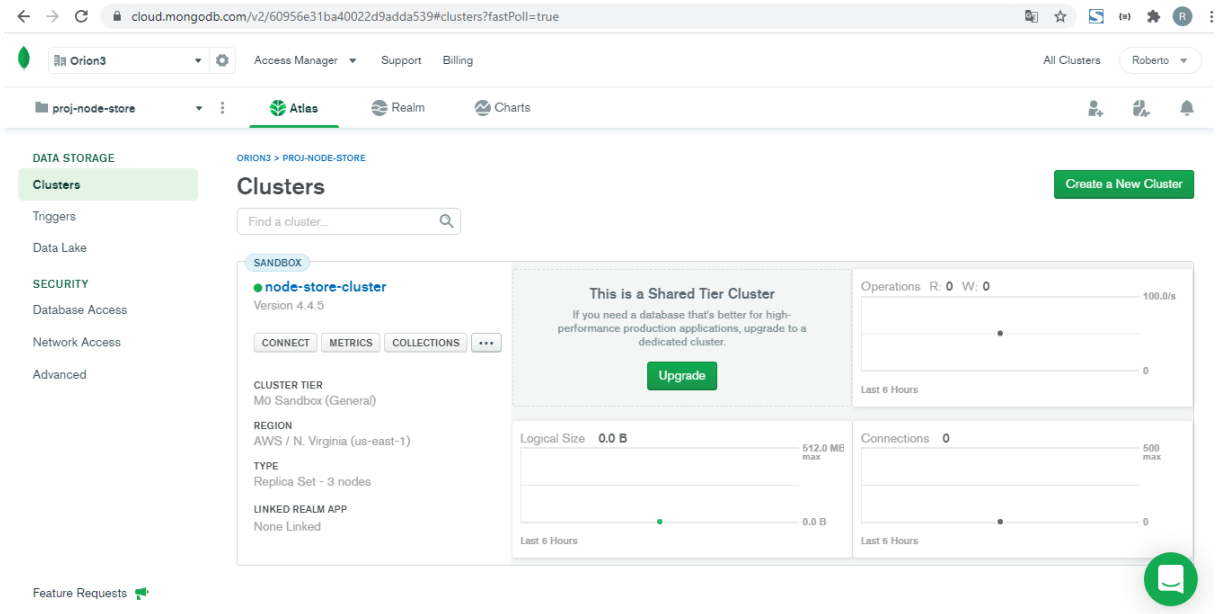
- Crie um cluster para este projeto:



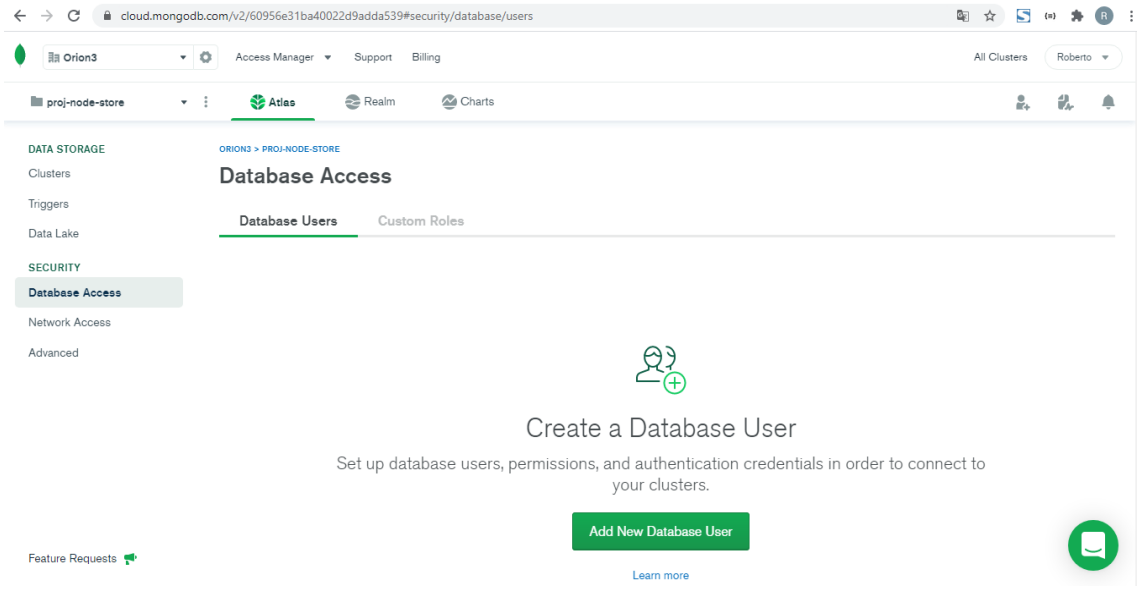
- Crie um cluster com nome: node-store-cluster

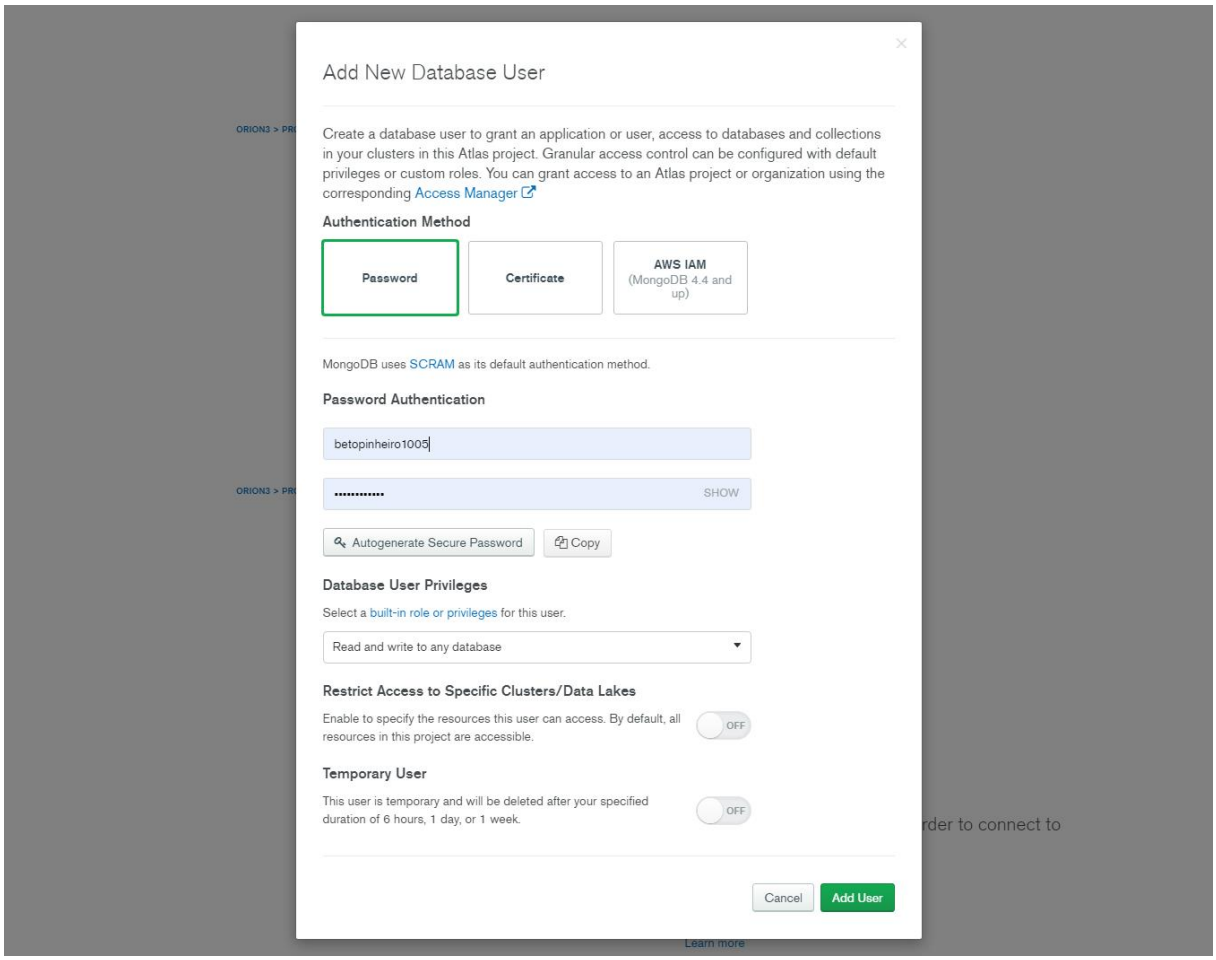


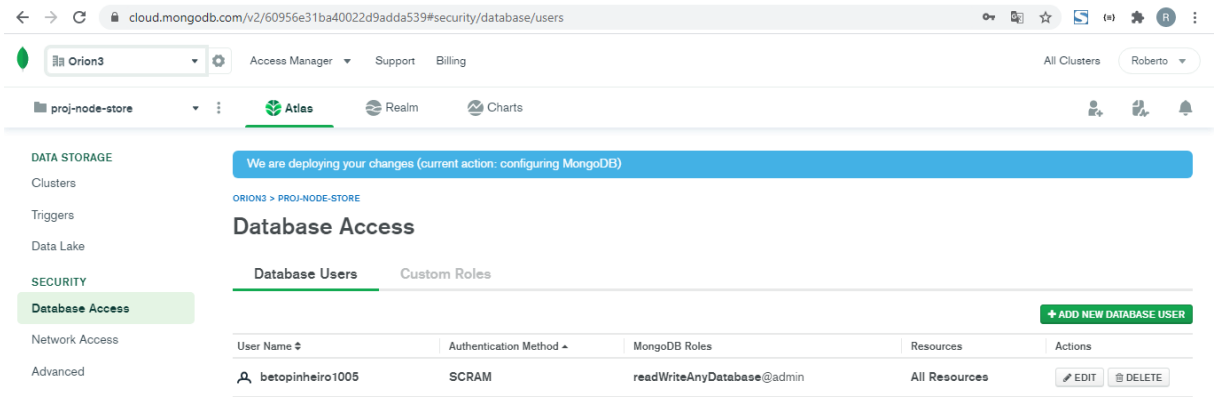
- Clique no botão "Create Cluster" e aguarde o cluster ser criado (leva um bom tempo):



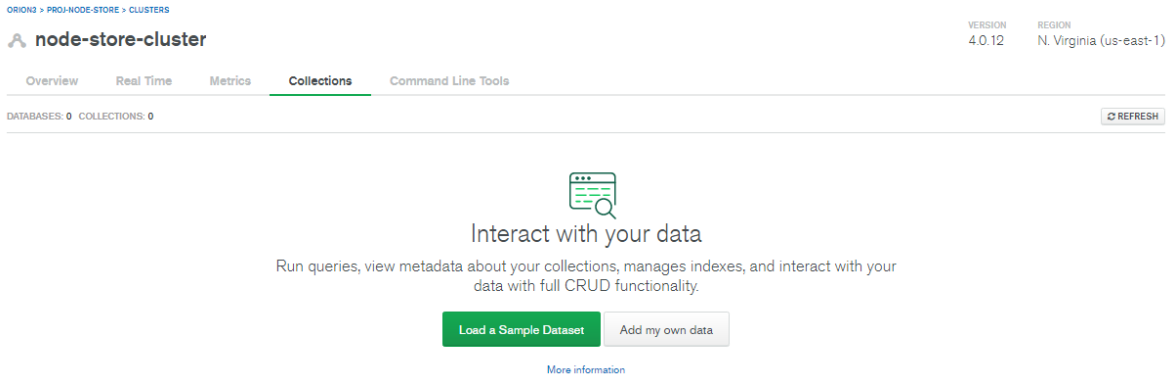
- Crie um usuário para acessar o banco de dados.



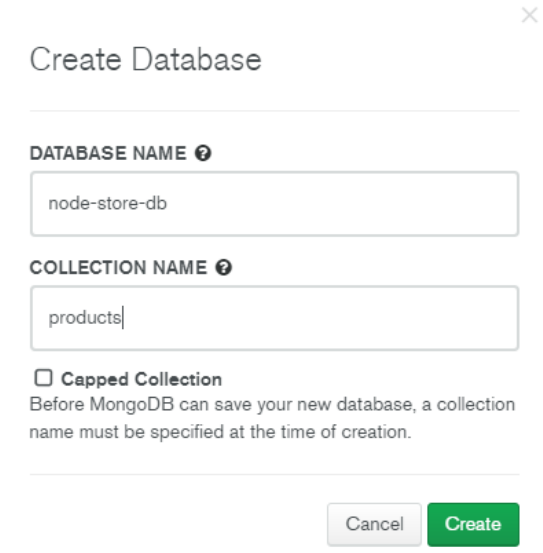

Add New Database User
Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding Access Manager
Authentication Method
Password
Certificate
AWS IAM
(MongoDB 4.4 and up)
MongoDB uses SCRAM as its default authentication method.
Password Authentication
betopinheiro1005
SHOW
Autogenerate Secure Password
Copy
Database User Privileges
Select a built-in role or privileges for this user.
Read and write to any database
Restrict Access to Specific Clusters/Data Lakes
Enable to specify the resources this user can access. By default, all resources in this project are accessible.
OFF
Temporary User
This user is temporary and will be deleted after your specified duration of 6 hours, 1 day, or 1 week.
OFF
Cancel
Add User
rder to connect to
Learn more

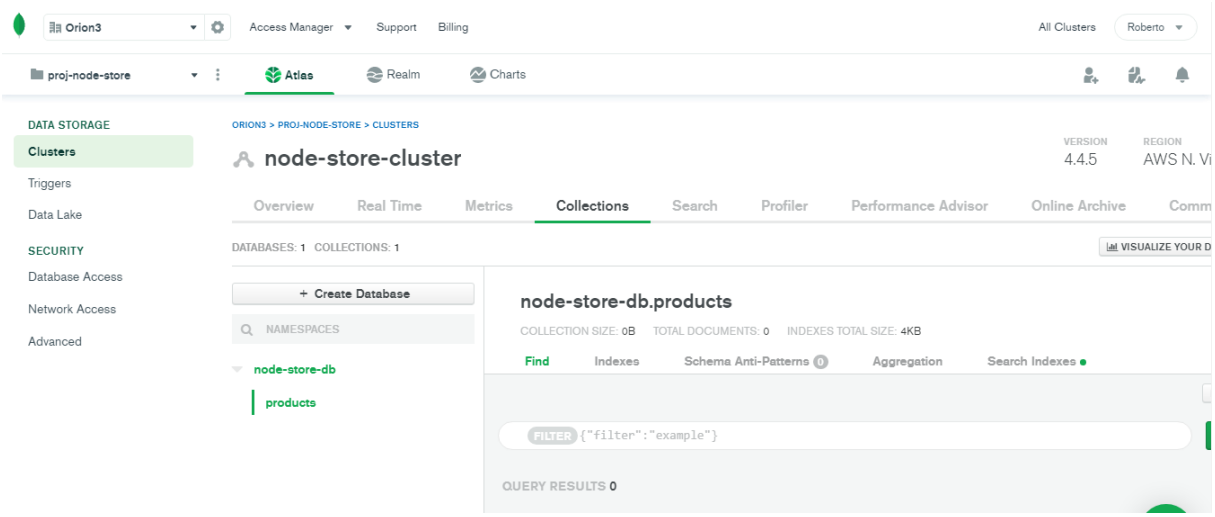

cloud.mongodb.com/v2/60956e31ba40022d9adda539#security/database/users
Orion3
Access Manager
Support
Billing
All Clusters
Roberto
proj-node-store
Atlas
Realm
Charts
DATA STORAGE
Clusters
Triggers
Data Lake
SECURITY
Database Access
Network Access
Advanced
We are deploying your changes (current action: configuring MongoDB)
ORION3 > PROJ-NODE-STORE
Database Access
Database Users
Custom Roles
+ ADD NEW DATABASE USER
User Name
Authentication Method
MongoDB Roles
Resources
Actions
betopinheiro1005
SCRAM
readWriteAnyDatabase@admin
All Resources
EDIT
DELETE

- Em node-store-cluster, clique na aba "Collections"

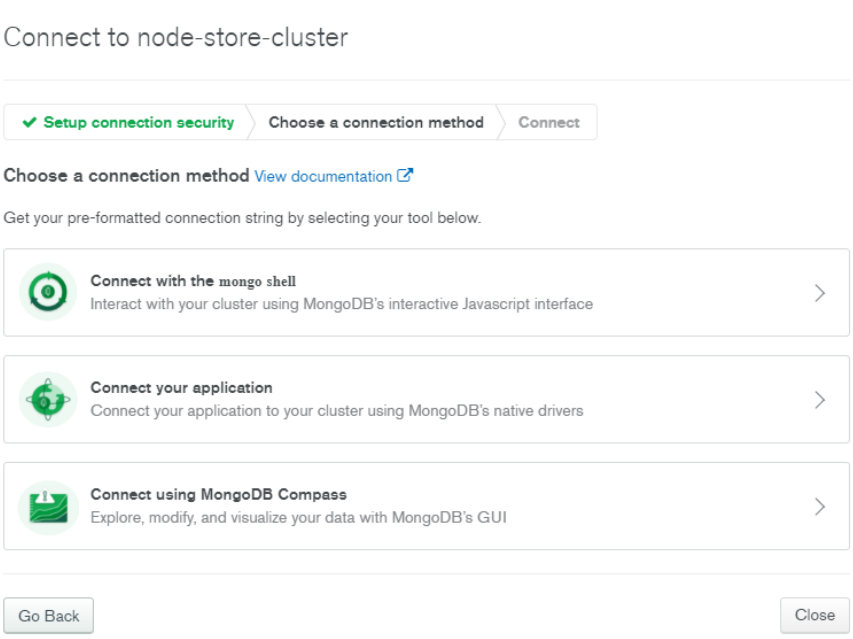


- Crie o database node-store-db com a collection products.

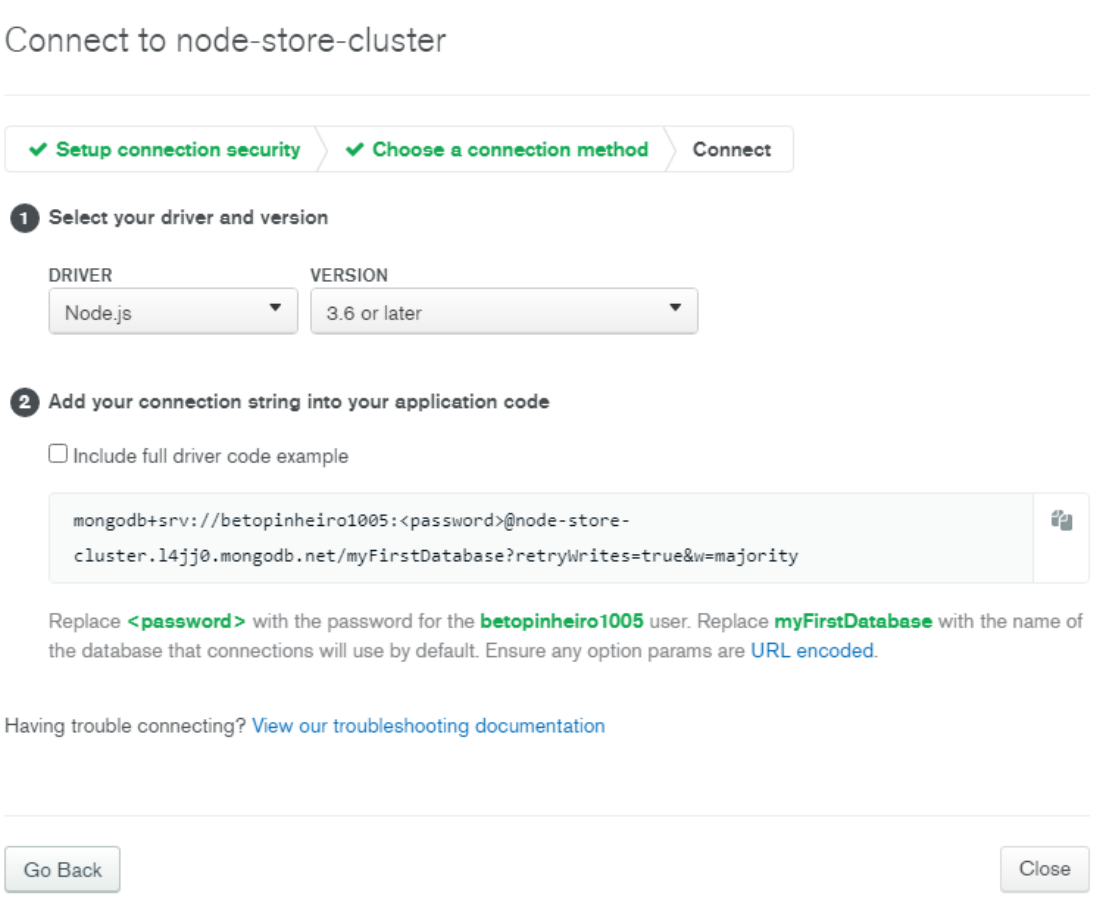




- Em node-store-cluster, clique no botão "Connect":



- Escolha o método de conexão: "Connect Your Application":



- Copie a string de conexão:

mongodb+srv://betopinheiro1005:<password>@node-store-cluster.l4jj0.mongodb.net/myFirstDatabase?retryWrites=true&w=majority

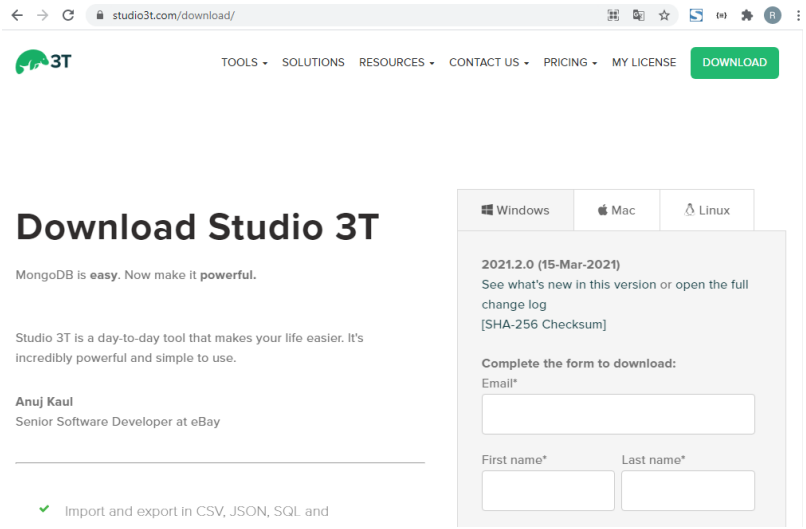
- Substitua myFirstDatabase por node-store-db
- Substitua <password> pela senha do usuário.

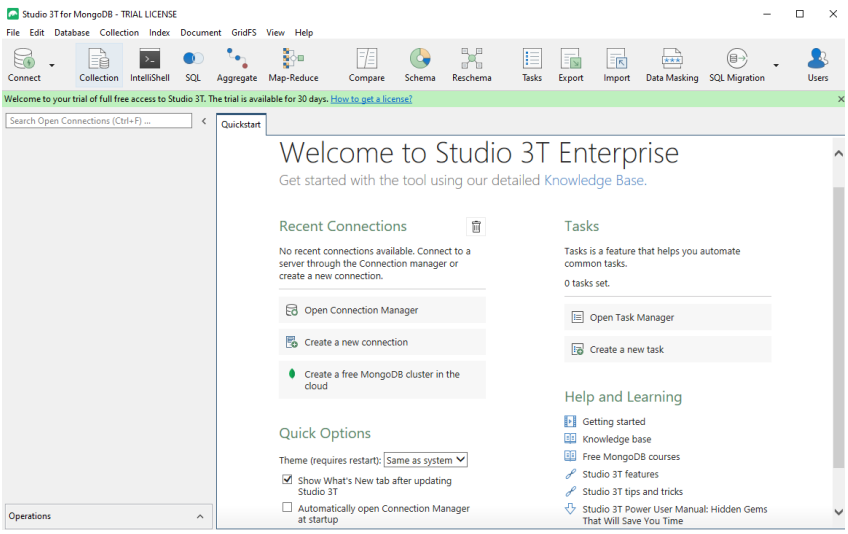
mongodb+srv://betopinheiro1005:angstron1005@node-store-cluster.l4jj0.mongodb.net/node-store-db?retryWrites=true&w=majority

---

- Baixe e instale o programa Studio 3T.

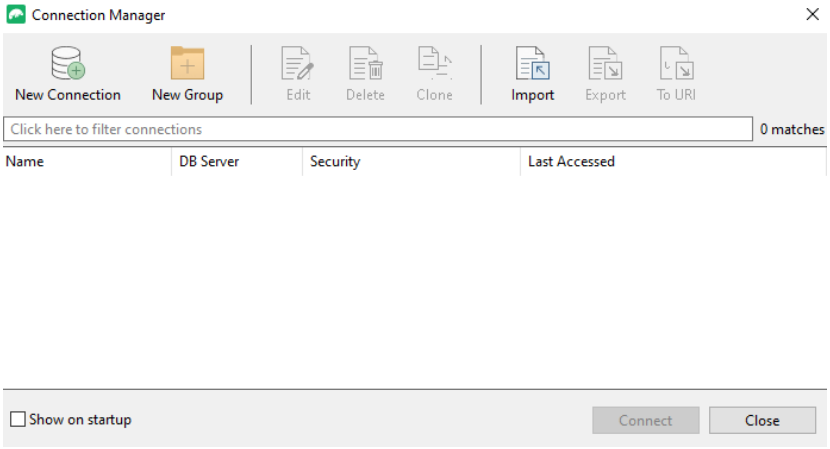
https://studio3t.com/download/





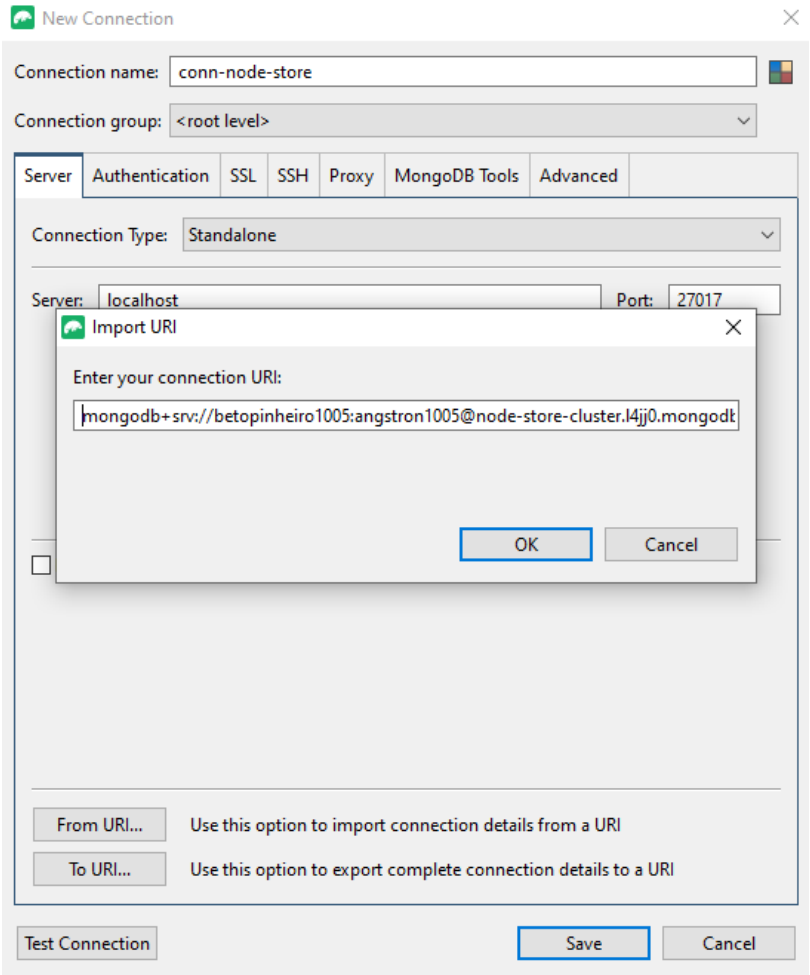
- Clique no botão Connect.

- Clique em New Connection.

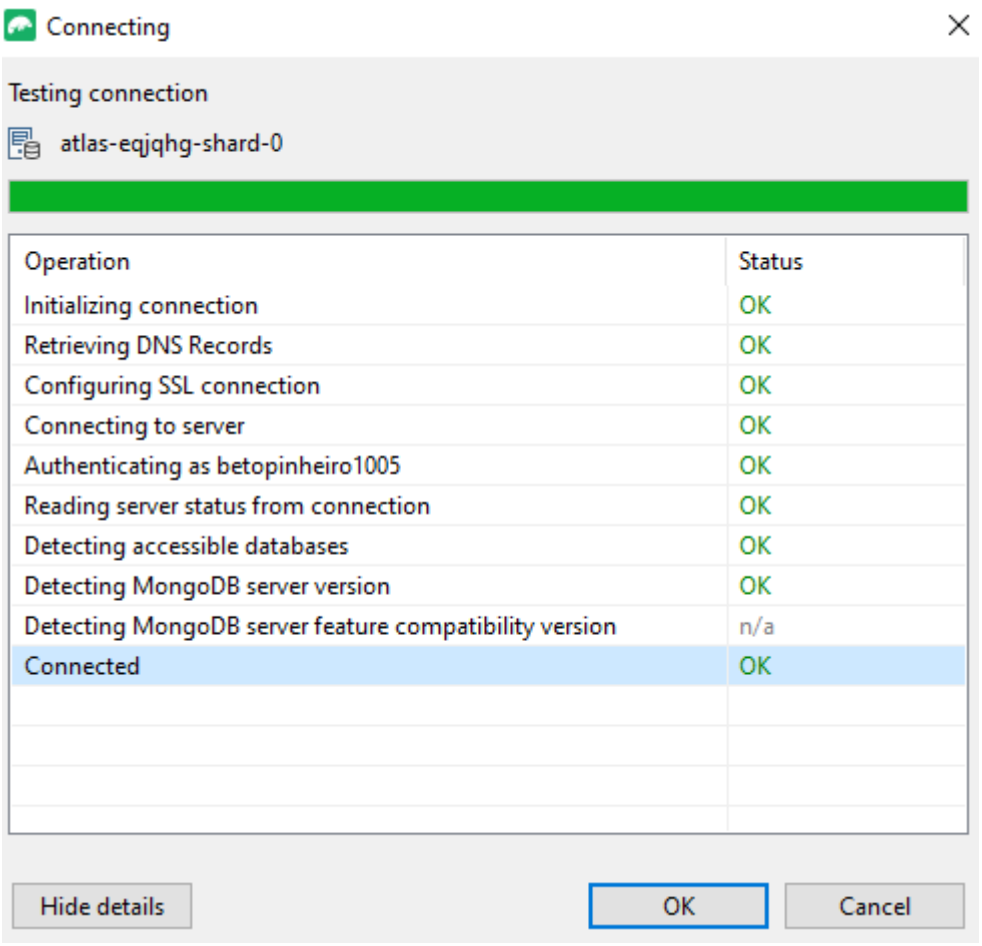


- Dê o nome para a conexão: conn-node-store.

- Clique no botão From URI e cole a string de conexão.

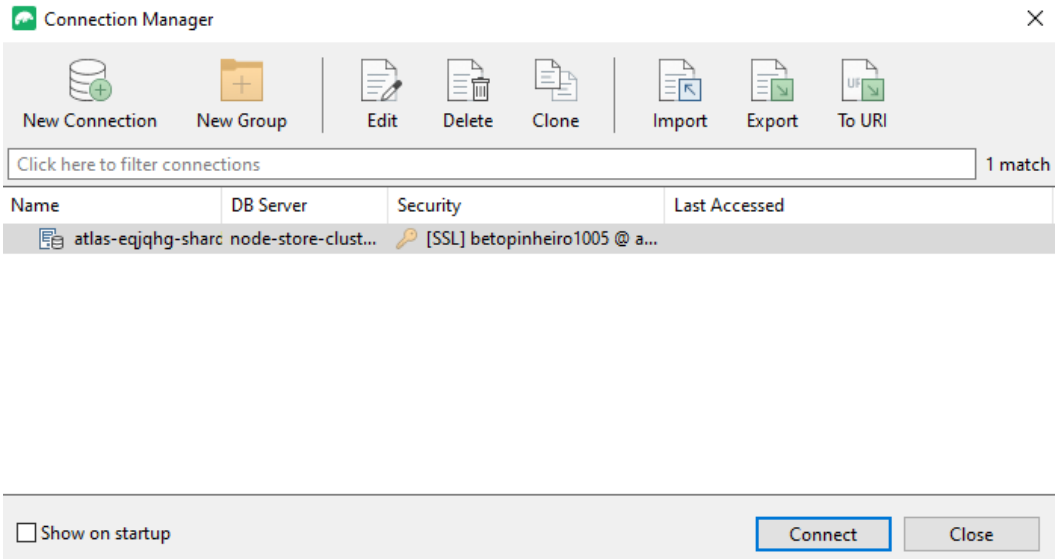


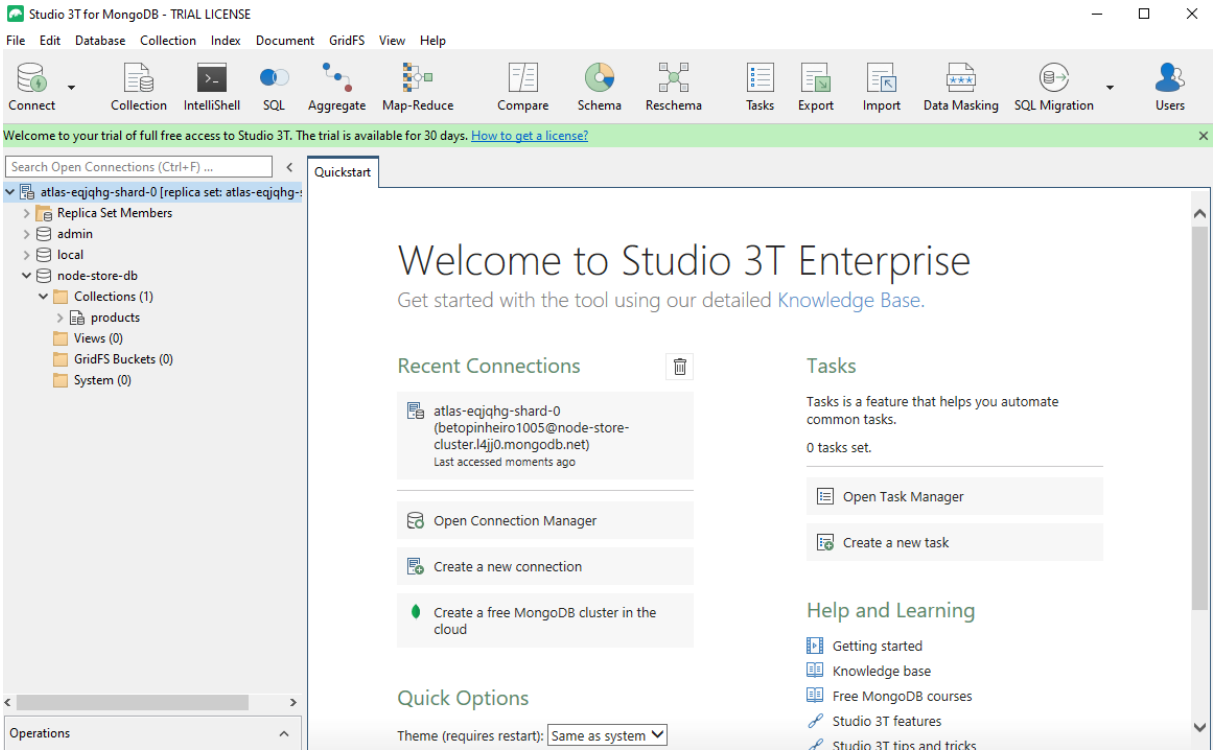
- Faça o teste de conexão:



- Se estiver tudo ok, clique no botão "Save" para salvar a configuração.

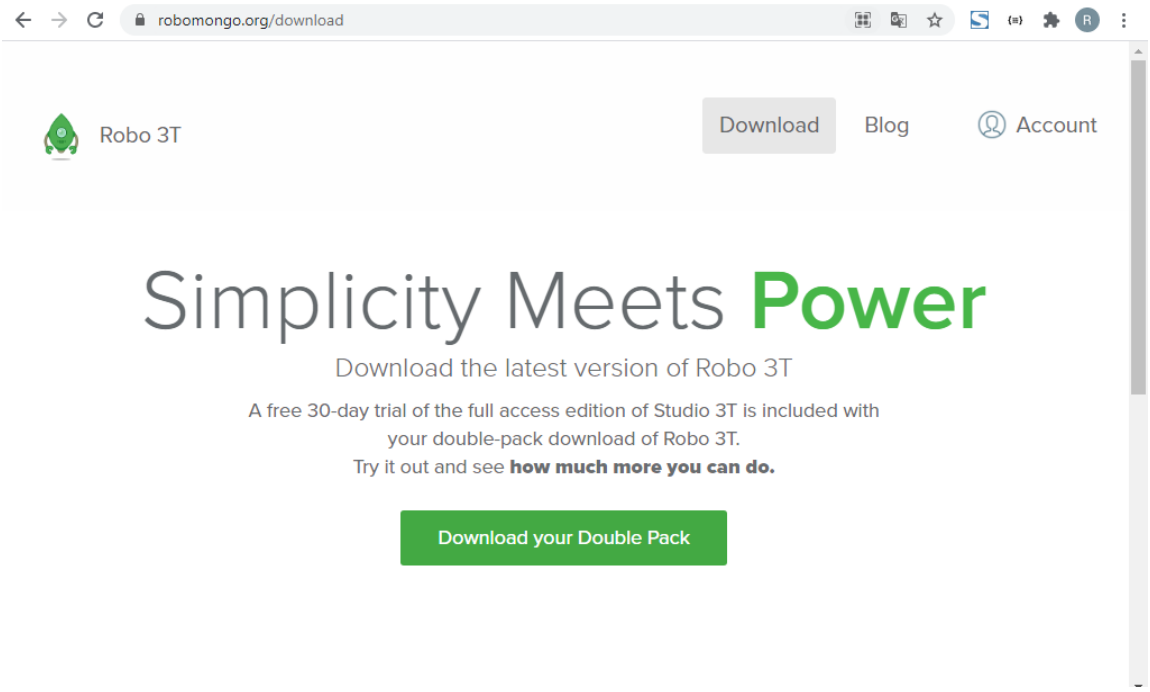
- Clique no botão "Connect"





- Outro possível serviço

https://robomongo.org/download



Aula 14 - Mongoose

Instalação do Mongoose

- Para fazer a conexão com o banco de dados vamos utilizar um pacote chamado Mongoose. Para instalá-lo use o comando:

npm install mongoose --save

```
C:\balta\nodejs\node-str>npm install mongoose --save
npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (c
urrent: {"os":"win32","arch":"x64"})

+ mongoose@5.12.7
added 29 packages from 92 contributors and audited 211 packages in 33.806s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

package.json

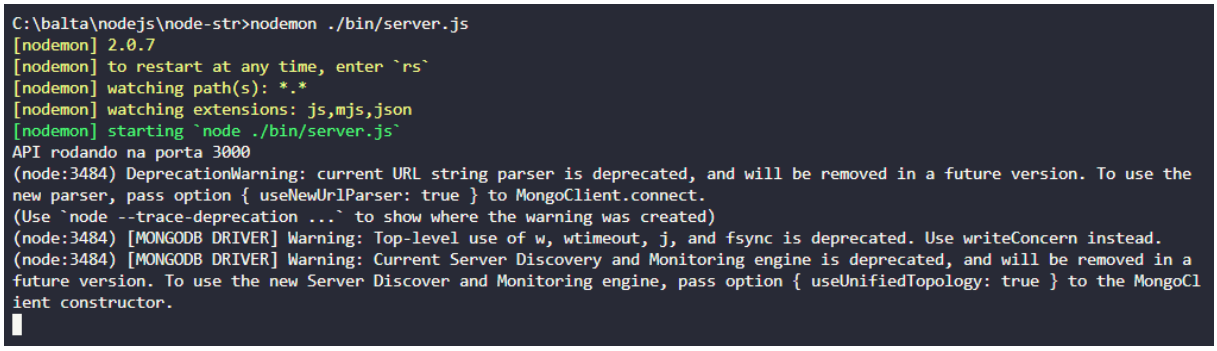
```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node ./bin/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "debug": "^4.3.1",
    "express": "^4.17.1",
    "http": "0.0.1-security",
    "mongoose": "^5.12.7"
  },
  "devDependencies": {
    "nodemon": "^2.0.7"
  }
}
```

src/app.js

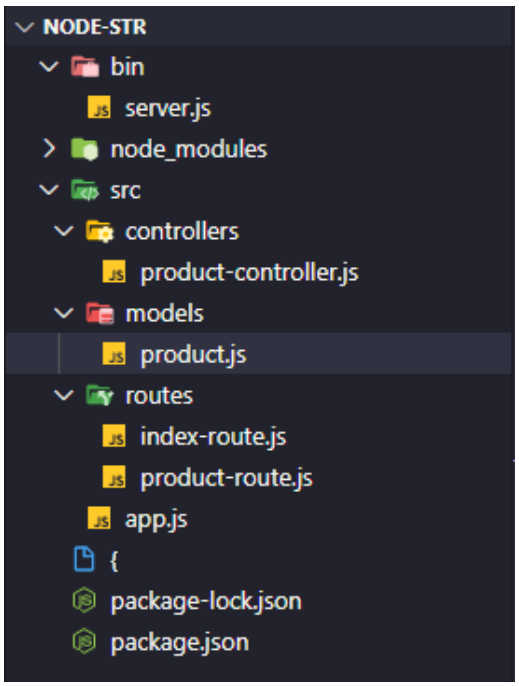
```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-store-cluster.l4jj0.mongodb.net/node-store-db?retryWrites=true&w=majority");

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);

module.exports = app;
```

- Se estiver tudo ok, o servidor irá rodar normalmente:

nodemon ./bin/server.js



Aula 15 - Models



src/models/product.js

```
'use strict';

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const schema = new Schema({
    title: {
        type: String,
        required: true,
        trim: true
    },
    slug: {
        type: String,
        required: [true, 'O slug é obrigatório'],
        trim: true,
        index: true,
        unique: true
    },
    description: {
        type: String,
        required: true
    },
    price: {
        type: Number,
        required: true
    },
```

```
    active: {
        type: Boolean,
        required: true,
        default: true
    },
    tags: [{
        type: String,
        required: true
    }]
});

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

Aula 16 - Criando um produto

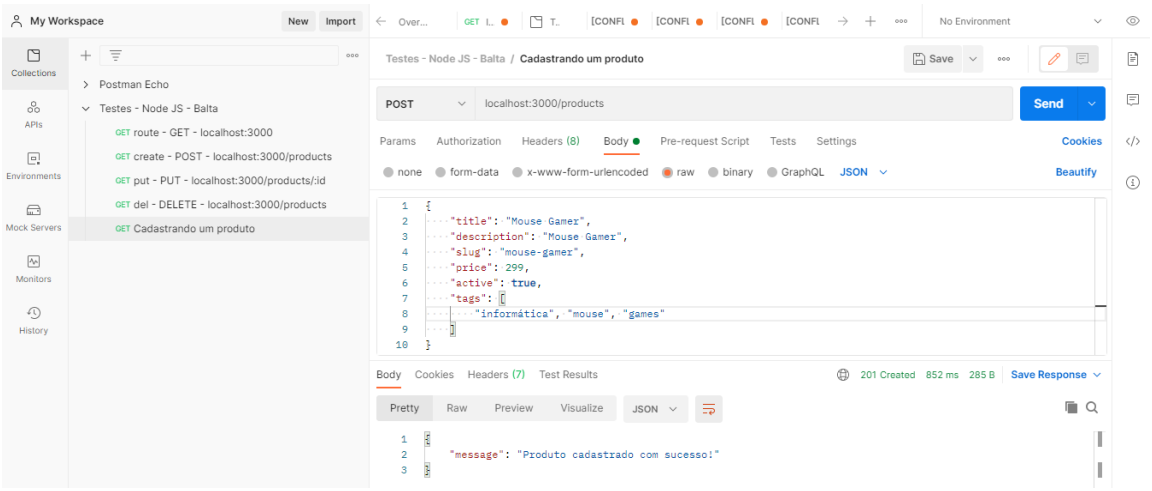
src\controllers\product-controller.js

```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({ message: "Produto cadastrado com sucesso!" });
    }).catch(e => {
        res.status(400).send({ message: "Falha ao cadastrar o produto!", data: e });
    });
};

exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

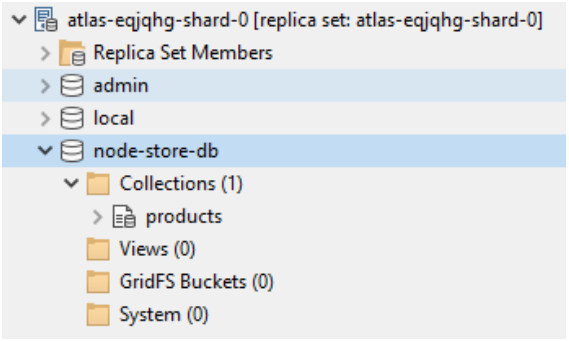
src\app.js

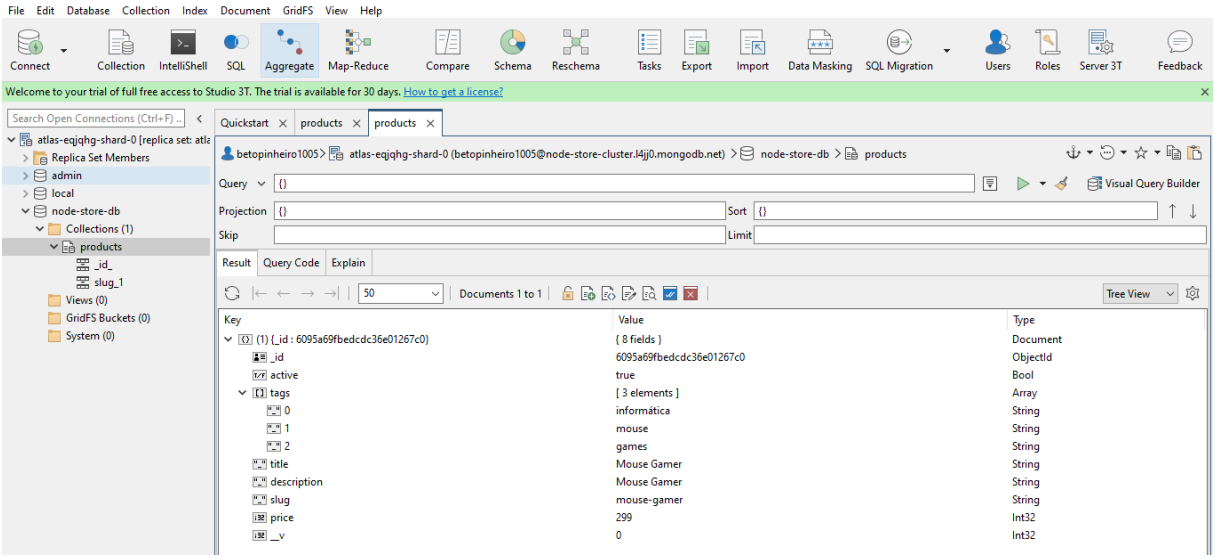
```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-str-f9kvu.mongodb.net/test?retryWrites=true&w=majority");

// Carrega os models
const Product = require('./models/product');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);

module.exports = app;
```

Usando o Postman para cadastrar um produto

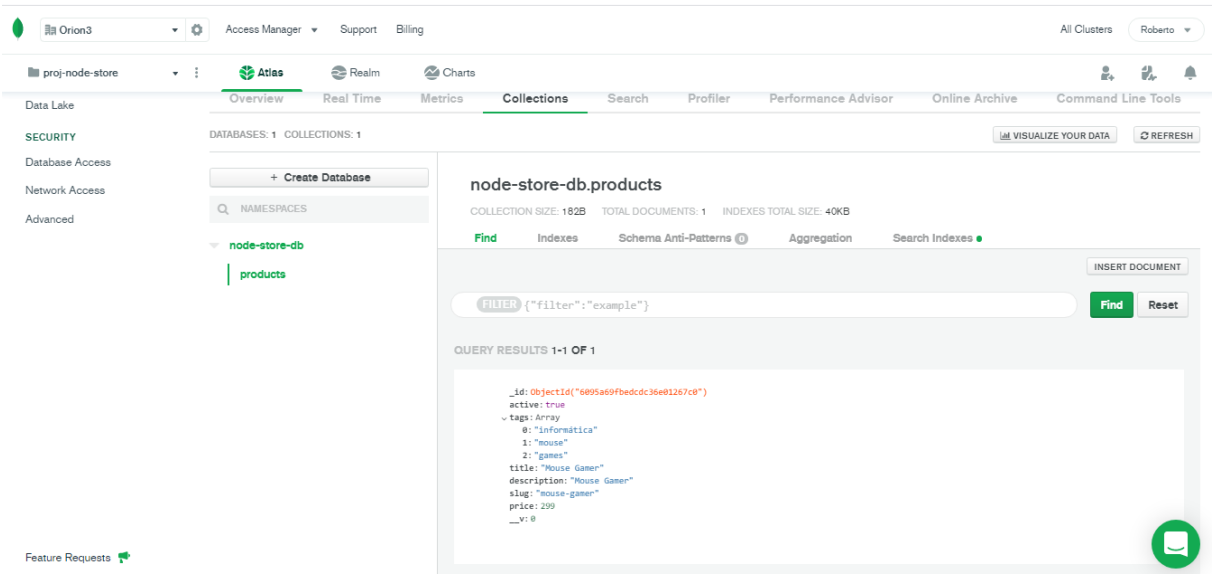


No Studio 3T:





No mlab:


Orion3
Access Manager
Support
Billing
All Clusters
Roberto
proj-node-store
Atlas
Realm
Charts
Data Lake
SECURITY
Database Access
Network Access
Advanced
Overview
Real Time
Metrics
Collections
Search
Profiler
Performance Advisor
Online Archive
Command Line Tools
DATABASES: 1 COLLECTIONS: 1
VISUALIZE YOUR DATA
REFRESH
+ Create Database
NAMESPACES
node-store-db
products
node-store-db.products
COLLECTION SIZE: 182B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 40KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
INSERT DOCUMENT
FILTER {"filter":"example"}
Find
Reset
QUERY RESULTS 1-1 OF 1
_id: ObjectId("6095a69fbedcdc36e01267c0")
active: true
tags: Array
0: "informática"
1: "mouse"
2: "games"
title: "Mouse Gamer"
description: "Mouse Gamer"
slug: "mouse-gamer"
price: 299
__v: 0
Feature Requests

Aula 17 - Listando os produtos

src/controllers/product-controller.js

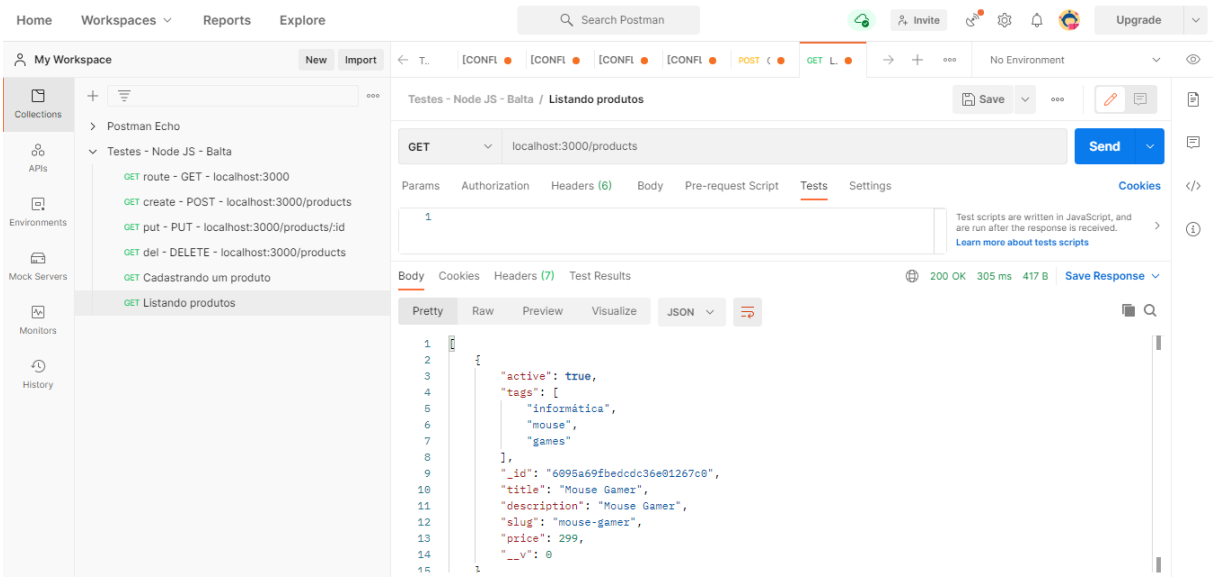
```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({}).then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e
});
    });
};

exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

src/routes/product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

router.get('/', controller.get);
router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/', controller.delete);

module.exports = router;
```

Testando no Postman

GET - localhost:3000/products



No navegador

http://localhost:3000/products

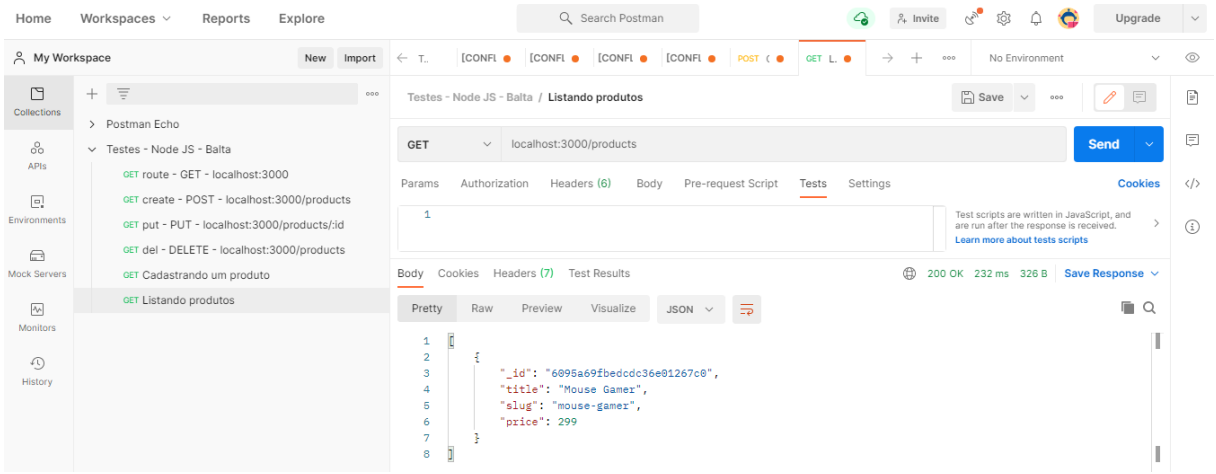
Exibindo apenas alguns campos

src/controllers/product-controller.js

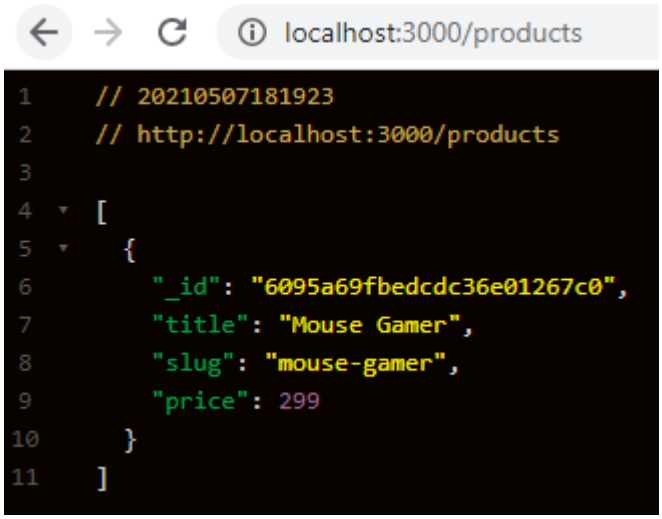
```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e
});
    });
};
```

```
exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

- No Postman:



- No browser:



Aula 18 - Listando um produto pelo slug

src/controllers/product-controller.js

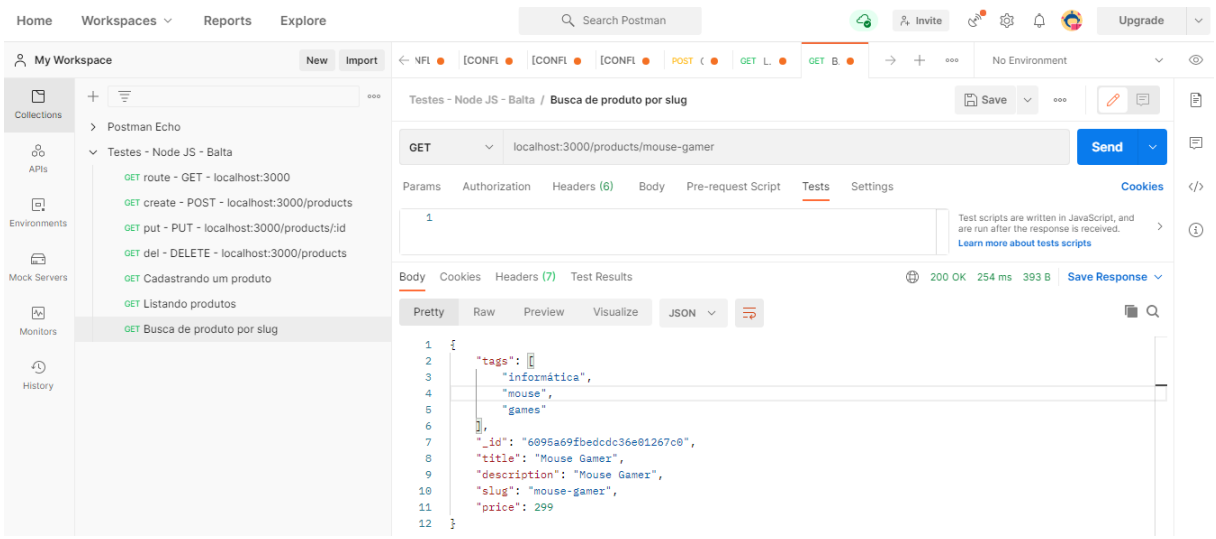
```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getBySlug = (req, res, next) => {
    Product.findOne({ slug: req.params.slug, active: true }, 'title description price slug tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e });
    });
};

exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

src/routes/product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

router.get('/', controller.get);
router.get('/:slug', controller.getBySlug);
router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/', controller.delete);

module.exports = router;
```

Testando no Postman

GET - localhost:3000/products/mouse-gamer



No navegador

Aula 19 - Listando um produto pelo id

src/controllers/product-controller.js

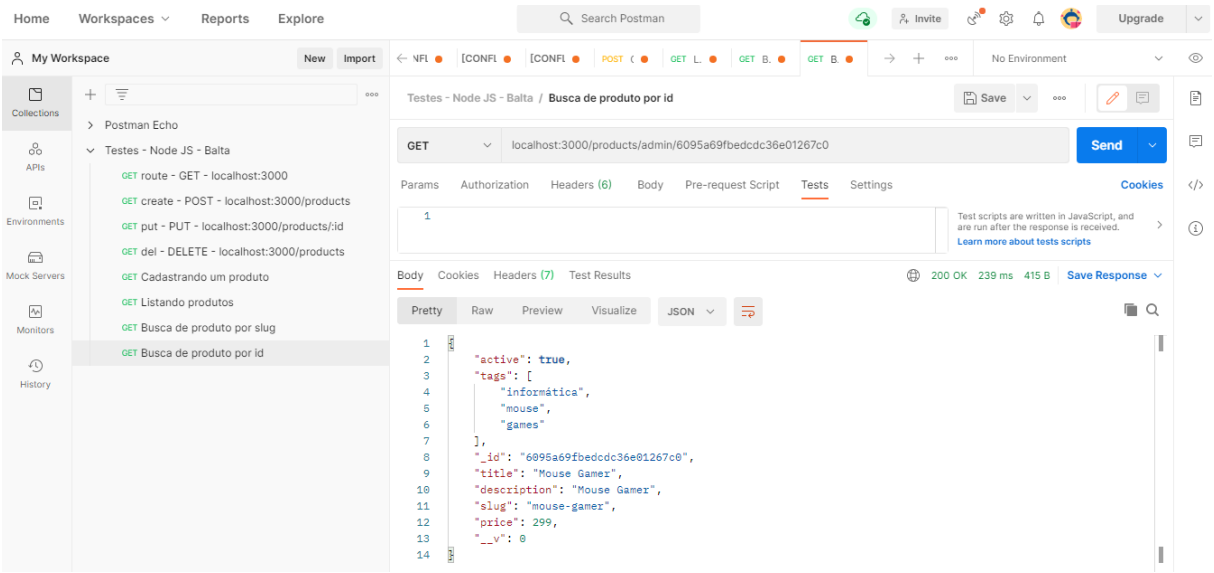
```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getBySlug = (req, res, next) => {
    Product.findOne({ slug: req.params.slug, active: true }, 'title description price slug
tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getById = (req, res, next) => {
    Product.findById({_id: req.params.id}).then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e });
    });
};

exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

src/routes/product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

router.get('/', controller.get);
router.get('/:slug', controller.getBySlug);
router.get('/admin/:id', controller.getById);
router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/', controller.delete);

module.exports = router;
```

Testando no Postman

localhost:3000/products/admin/6095a69fbedcdc36e01267c0



Testando no navegador

http://localhost:3000/products/admin/6095a69fbedcdc36e01267c0

Aula 20 - Listando os produtos de uma tag

src/controllers/product-controller.js

```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getBySlug = (req, res, next) => {
    Product.findOne({ slug: req.params.slug, active: true }, 'title description price
slug tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getById = (req, res, next) => {
    Product.findById({_id: req.params.id}).then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getByTag = (req, res, next) => {
    Product.find({tags: req.params.tag, active: true}, 'title description price slug
tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e
});
```

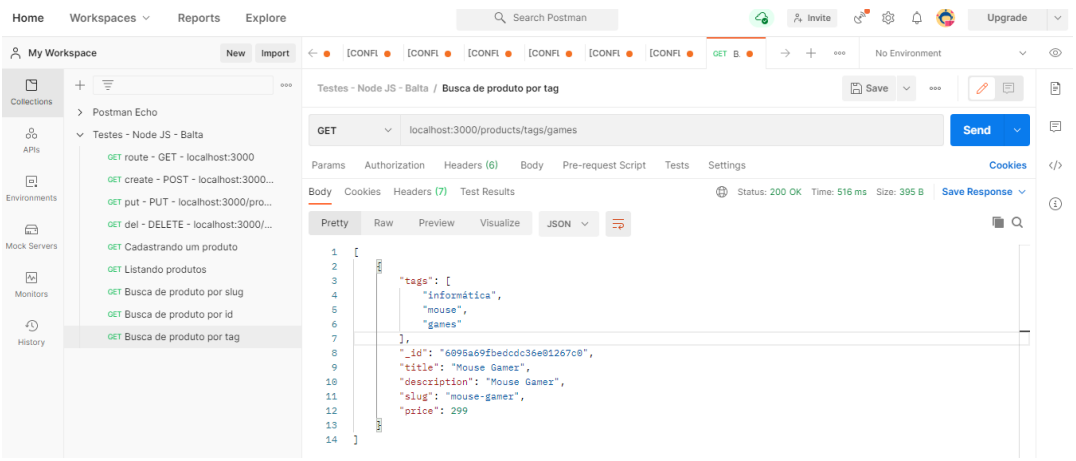
```
    });
};

exports.put = (req, res, next) => {
    const id = req.params.id;
    res.status(200).send({
        id: id,
        item: req.body
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

src/routes/product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

router.get('/', controller.get);
router.get('/:slug', controller.getBySlug);
router.get('/admin/:id', controller.getById);
router.get('/tags/:tag', controller.getByTag);
router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/', controller.delete);

module.exports = router;
```

Testando no Postman

GET - localhost:3000/products/tags/games



Aula 21 - Atualizando um produto

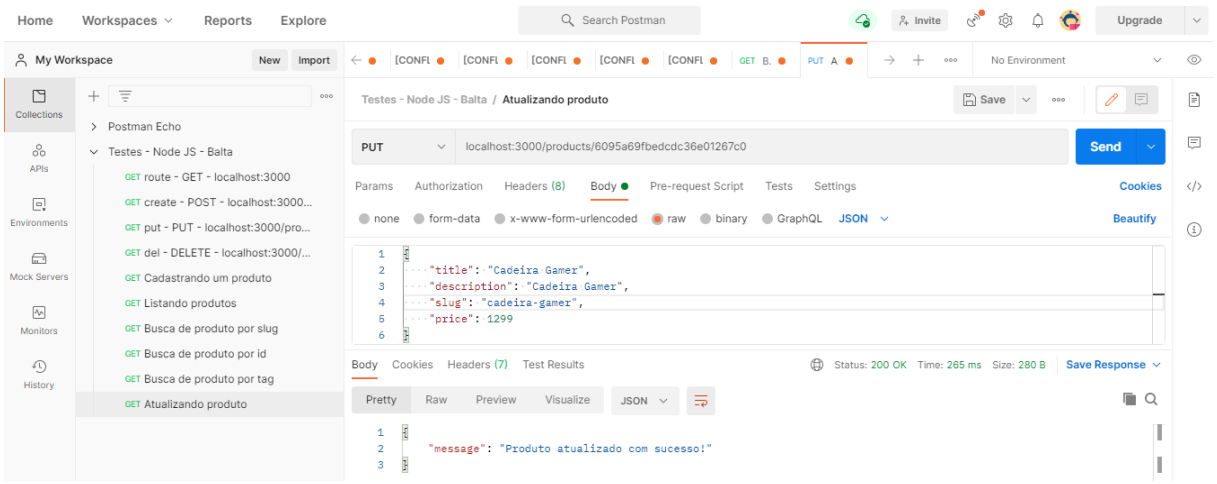
src/controllers/product-controller.js

```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getBySlug = (req, res, next) => {
    Product.findOne({ slug: req.params.slug, active: true }, 'title description price slug
tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getById = (req, res, next) => {
    Product.findById({_id: req.params.id}).then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getByTag = (req, res, next) => {
    Product.find({tags: req.params.tag, active: true}, 'title description price slug
tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e });
    });
};
```

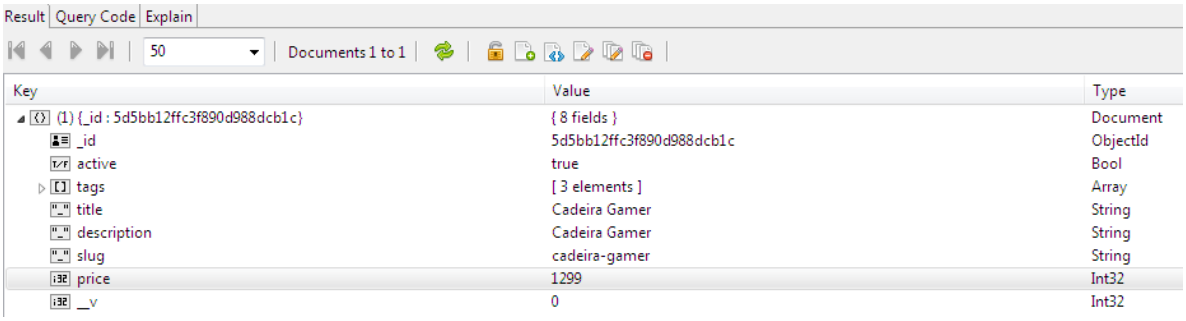
```
exports.put = (req, res, next) => {
    Product.findByIdAndUpdate(req.params.id, {
        $set: {
            title: req.body.title,
            description: req.body.description,
             slug: req.body.slug,
            price: req.body.price
        }
    }).then(x => {
        res.status(200).send({
            message: "Produto atualizado com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao atualizar produto!", data: e
        });
    });
};

exports.delete = (req, res, next) => {
    res.status(200).send(req.body);
};
```

Testando no Postman

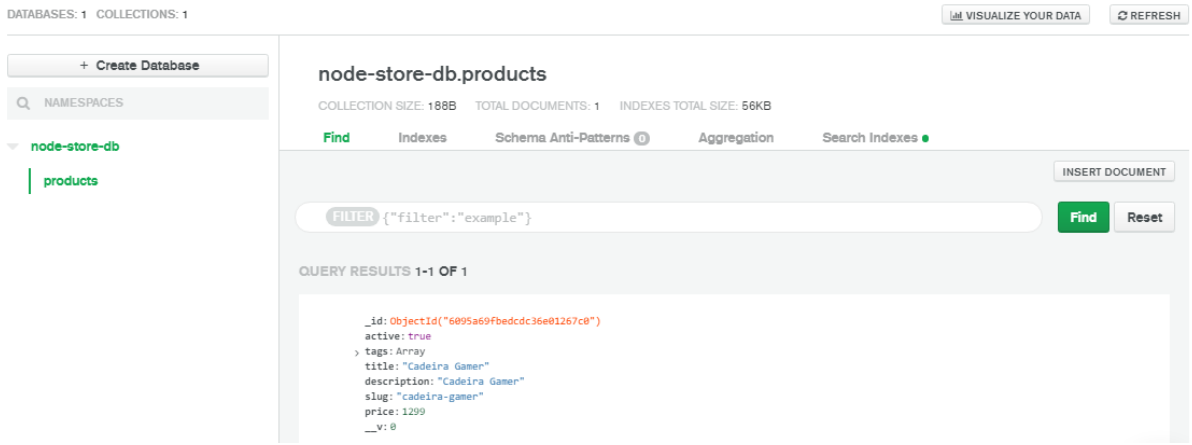
PUT - localhost:3000/products/6095a69fbedcdc36e01267c0



No Studio 3T



No mlab



Aula 22 - Excluindo um produto

src/controllers/product-controller.js

```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getBySlug = (req, res, next) => {
    Product.findOne({ slug: req.params.slug, active: true }, 'title description price slug
tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getById = (req, res, next) => {
    Product.findById({_id: req.params.id}).then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.getByTag = (req, res, next) => {
    Product.find({tags: req.params.tag, active: true}, 'title description price slug
tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });;
};

exports.post = (req, res, next) => {
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e });
    });
};
```

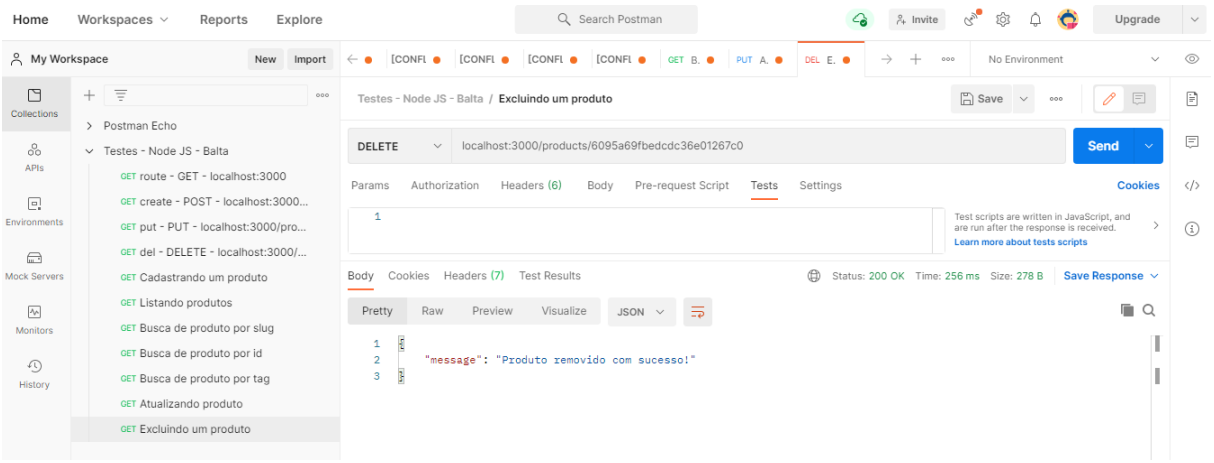
```
exports.put = (req, res, next) => {
    Product.findByIdAndUpdate(req.params.id, {
        $set: {
            title: req.body.title,
            description: req.body.description,
            slug: req.body.slug,
            price: req.body.price
        }
    }).then(x => {
        res.status(200).send({
            message: "Produto atualizado com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao atualizar produto!", data: e
        });
    });
};

exports.delete = (req, res, next) => {
    Product.findOneAndRemove(req.params.id).then(x => {
        res.status(200).send({
            message: "Produto removido com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao remover produto!", data: e
        });
    });
};
```

src/routes/product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

router.get('/', controller.get);
router.get('/:slug', controller.getBySlug);
router.get('/admin/:id', controller.getById);
router.get('/tags/:tag', controller.getByTag);
router.post('/', controller.post);
router.put('/:id', controller.put);
router.delete('/:id', controller.delete);

module.exports = router;
```

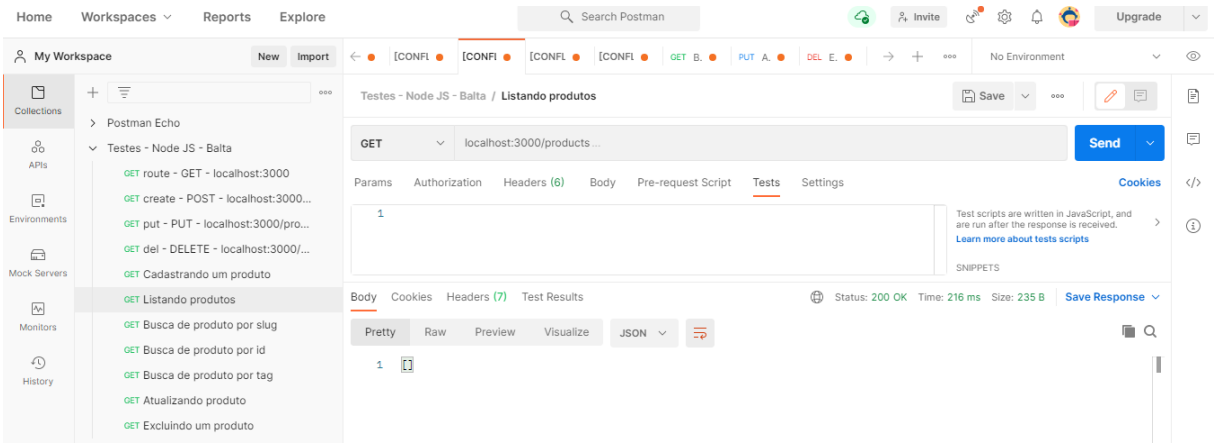
Testando no Postman

localhost:3000/products/6095a69fbedcdc36e01267c0

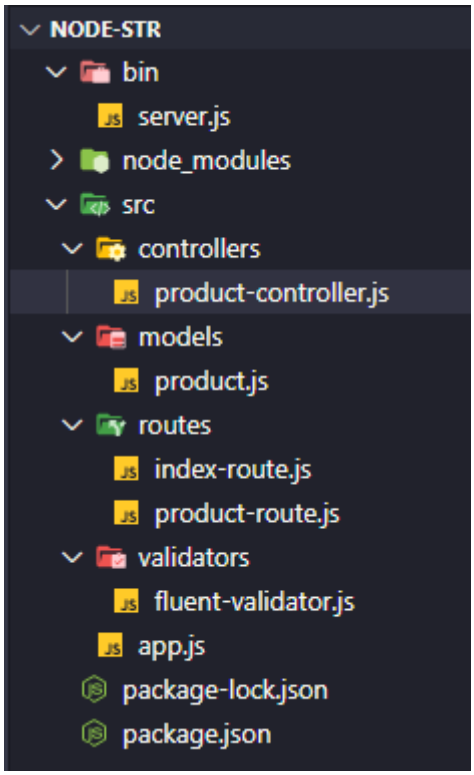


- Agora, ao listar os produtos:

GET - localhost:3000/products



Aula 23 - Validações



src/validators/fluent-validator.js

```
'use strict';

let errors = [];

function ValidationContract() {
    errors = [];
}

ValidationContract.prototype.isRequired = (value, message) => {
    if (!value || value.length <= 0)
        errors.push({ message: message });
}

ValidationContract.prototype.hasMinLen = (value, min, message) => {
    if (!value || value.length < min)
        errors.push({ message: message });
}

ValidationContract.prototype.hasMaxLen = (value, max, message) => {
    if (!value || value.length > max)
        errors.push({ message: message });
}

ValidationContract.prototype.isFixedLen = (value, len, message) => {
    if (value.length != len)
        errors.push({ message: message });
}

ValidationContract.prototype.isEmail = (value, message) => {
    var reg = new RegExp(/^\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*$/);
    if (!reg.test(value))
        errors.push({ message: message });
}

ValidationContract.prototype.errors = () => {
    return errors;
}

ValidationContract.prototype.clear = () => {
    errors = [];
}

ValidationContract.prototype.isValid = () => {
    return errors.length == 0;
}

module.exports = ValidationContract;
```

src/controllers/product-controller.js

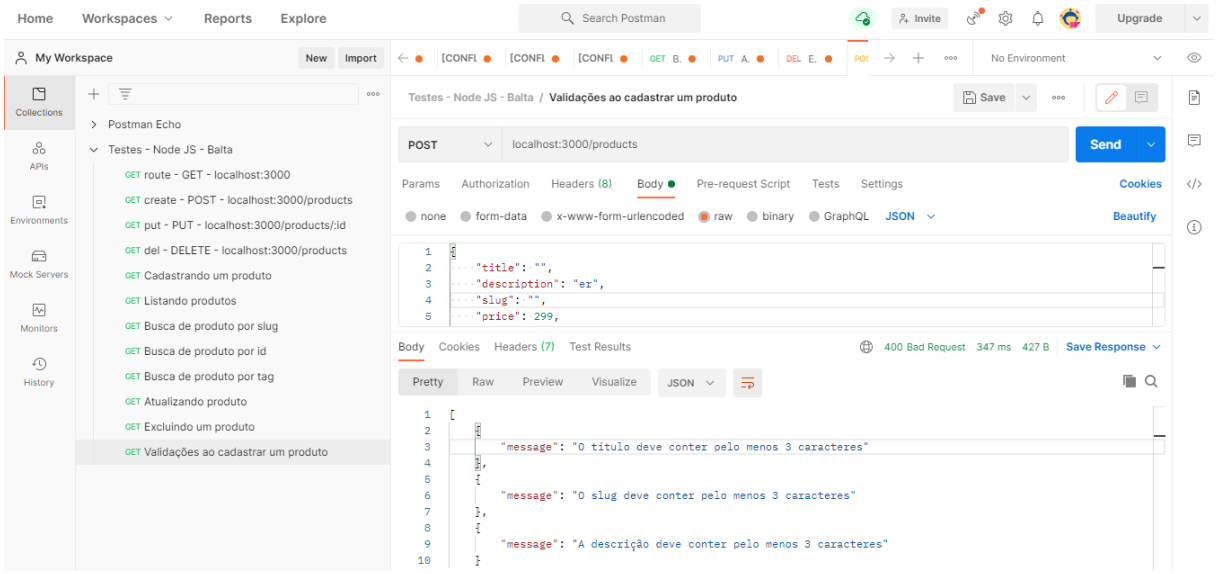
```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');
const ValidationContract = require('../validators/fluent-validator');

exports.get = (req, res, next) => {
    Product.find({ active: true }, 'title price slug').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

exports.getBySlug = (req, res, next) => {
    Product.findOne({ slug: req.params.slug, active: true }, 'title description price slug tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

exports.getById = (req, res, next) => {
    Product.findById({_id: req.params.id}).then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

exports.getByTag = (req, res, next) => {
    Product.find({tags: req.params.tag, active: true}, 'title description price slug tags').then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

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

    let contract = new ValidationContract();
    contract.hasMinLen(req.body.title, 3, 'O título deve conter pelo menos 3 caracteres');
    contract.hasMinLen(req.body.slug, 3, 'O slug deve conter pelo menos 3 caracteres');
    contract.hasMinLen(req.body.description, 3, 'A descrição deve conter pelo menos 3 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }
```

```
    var product = new Product(req.body);
    product.save().then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e });
    });
};

exports.put = (req, res, next) => {
    Product.findByIdAndUpdate(req.params.id, {
        $set: {
            title: req.body.title,
            description: req.body.description,
            slug: req.body.slug,
            price: req.body.price
        }
    }).then(x => {
        res.status(200).send({
            message: "Produto atualizado com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao atualizar produto!", data: e
        });
    });
};

exports.delete = (req, res, next) => {
    Product.findOneAndRemove(req.body.id).then(x => {
        res.status(200).send({
            message: "Produto removido com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao remover produto!", data: e
        });
    });
};
```

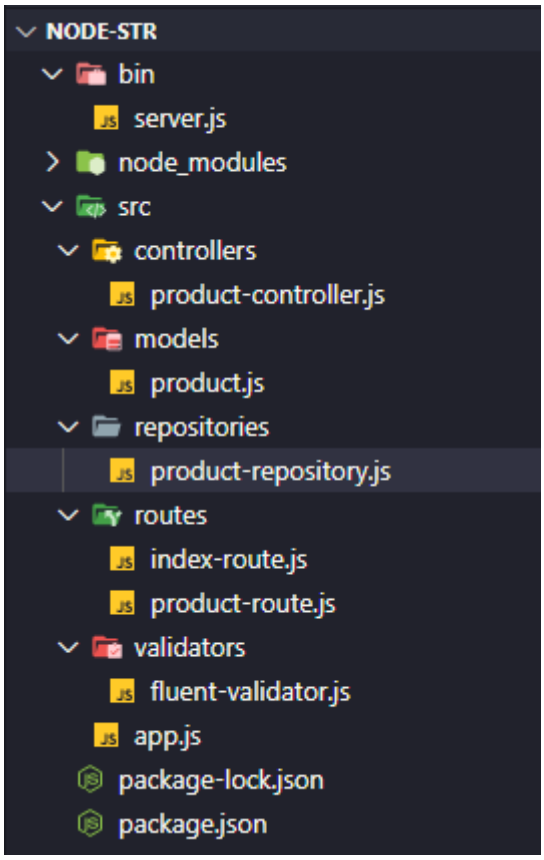
nodemon ./bin/server.js

```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:9048) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:9048) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:9048) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoCl
ient constructor.
(node:9048) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

Testando no Postman



Aula 24 - Repositórios



src/repositories/product-repository.js

```
'use strict';
const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = () => {
    return Product.find({
        active: true
    }, 'title price slug');
}

exports.getBySlug = (slug) => {
    return Product
        .findOne({
            slug: slug,
            active: true
        }, 'title description price slug tags');
}

exports.getById = (id) => {
    return Product
        .findById(id);
}
```

```
exports.getByTag = (tag) => {
    return Product
        .find({
            tags: tag,
            active: true
        }, 'title description price slug tags');
}

exports.create = (data) => {
    var product = new Product(data);
    return product.save();
}

exports.update = (id, data) => {
    return Product
        .findByIdAndUpdate(id, {
            $set: {
                title: data.title,
                description: data.description,
                price: data.price,
                slug: data.slug
            }
        });
}

exports.delete = (id) => {
    return Product
        .findOneAndRemove(id);
}
```

src/controllers/product-controller.js

```
'use strict';

const mongoose = require('mongoose');
const Product = mongoose.model('Product');
const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/product-repository');

exports.get = (req, res, next) => {
    repository
    .get()
    .then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

exports.getBySlug = (req, res, next) => {
    repository
    .getBySlug(req.params.slug)
    .then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

exports.getById = (req, res, next) => {
    repository
    .getById(req.params.id)
    .then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};

exports.getByTag = (req, res, next) => {
    repository
    .getByTag(req.params.tag)
    .then(data => {
        res.status(200).send(data);
    }).catch(e => {
        res.status(400).send(e);
    });
};
```

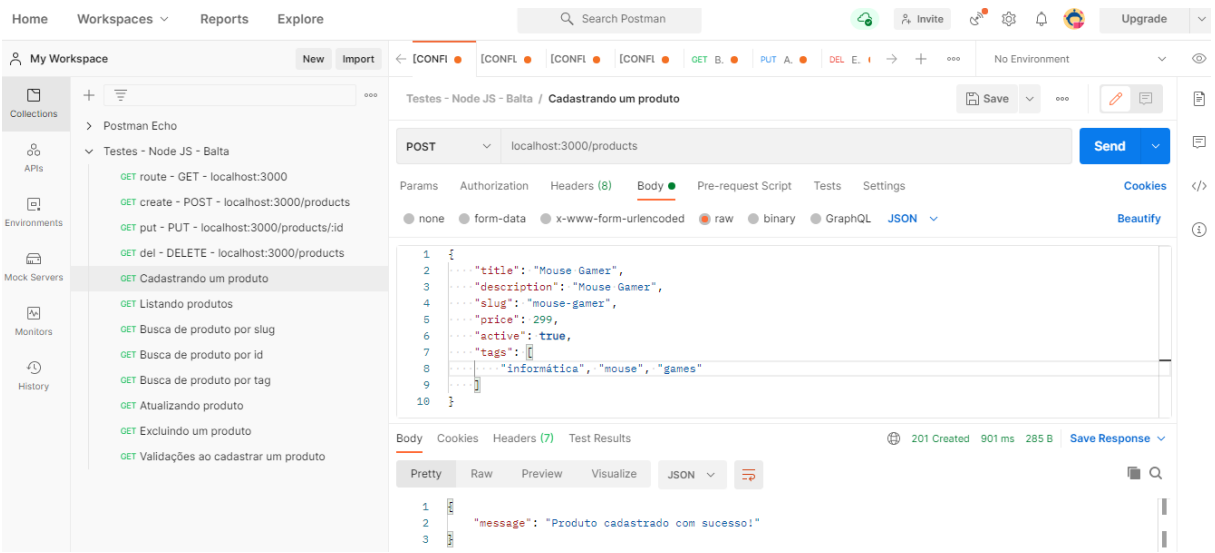
```
exports.post = (req, res, next) => {

    let contract = new ValidationContract();
    contract.hasMinLen(req.body.title, 3, 'O título deve conter pelo menos 3 caracteres');
    contract.hasMinLen(req.body.slug, 3, 'O slug deve conter pelo menos 3 caracteres');
    contract.hasMinLen(req.body.description, 3, 'A descrição deve conter pelo menos 3
caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    repository
    .create(req.body)
    .then(x => {
        res.status(201).send({message: 'Produto cadastrado com sucesso!'});
    }).catch(e => {
        res.status(400).send({message: 'Falha ao cadastrar o produto!', data: e });
    });
};

exports.put = (req, res, next) => {
    repository
    .update(req.params.id, req.body)
    .then(x => {
        res.status(200).send({
            message: "Produto atualizado com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao atualizar produto!", data: e
        });
    });
};

exports.delete = (req, res, next) => {
    repository
    .delete(req.body.id)
    .then(x => {
        res.status(200).send({
            message: "Produto removido com sucesso!"
        });
    }).catch(e => {
        res.status(400).send({
            message: "Falha ao remover produto!", data: e
        });
    });
};
```

nodemon ./bin/server.js

```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:2428) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:2428) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:2428) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoCl
ient constructor.
(node:2428) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

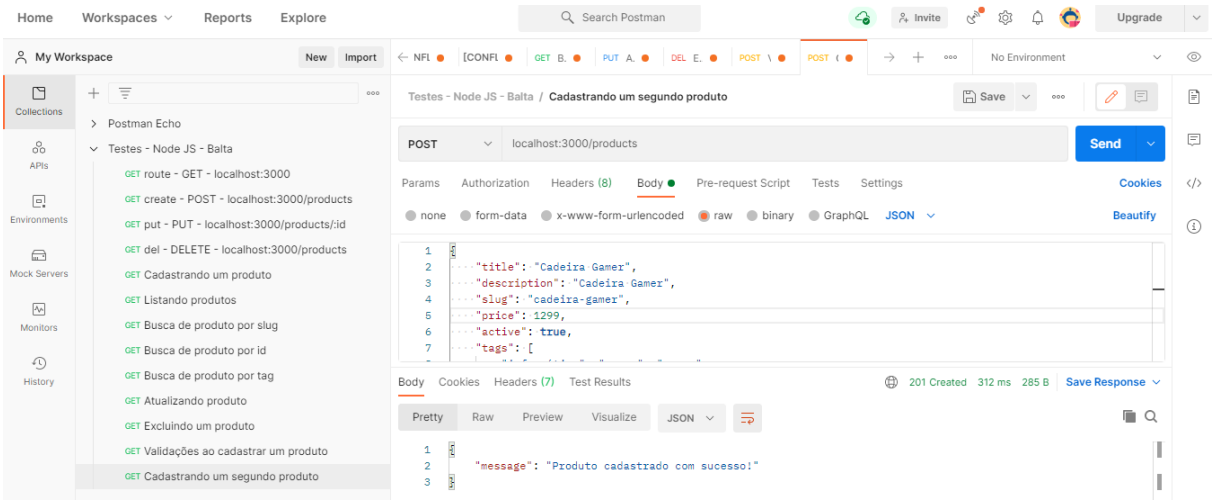
Cadastrando um produto

POST - localhost:3000/products



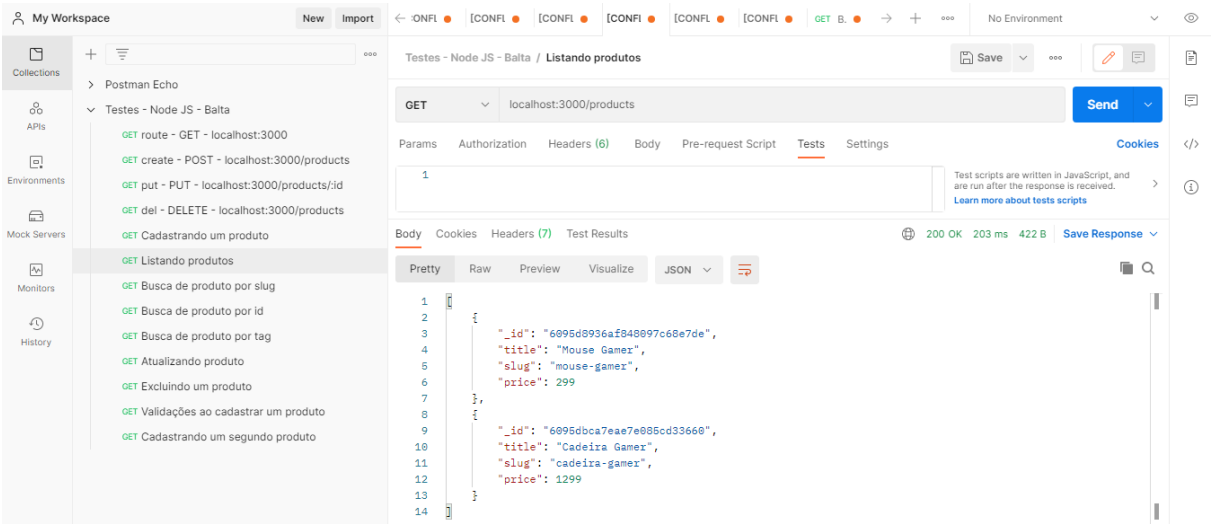
Cadastrando um segundo produto

POST - localhost:3000/products



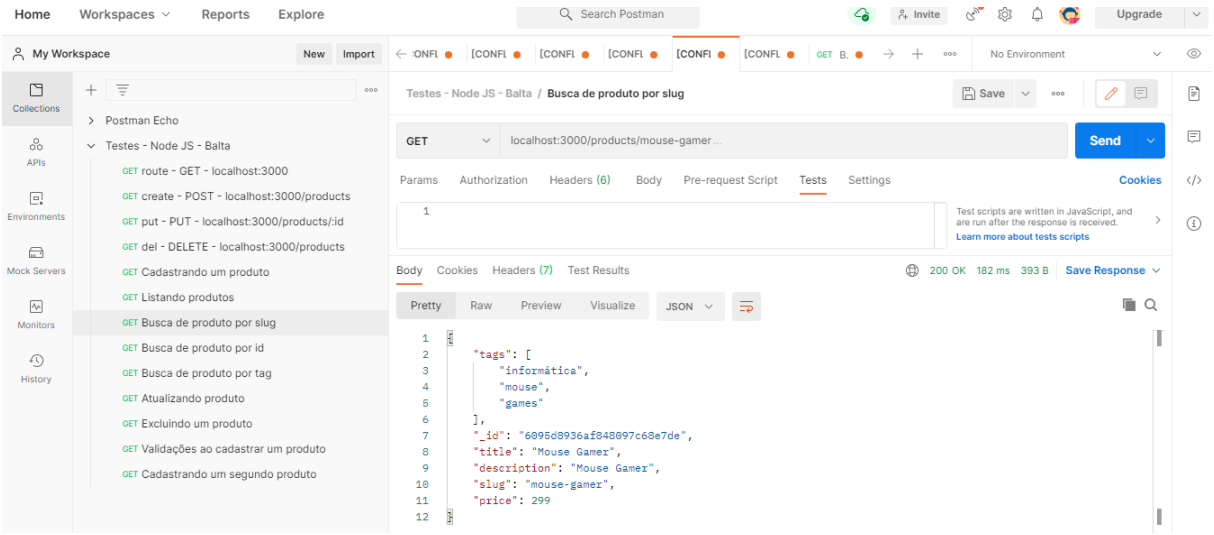
Listando os produtos

GET - localhost:3000/products



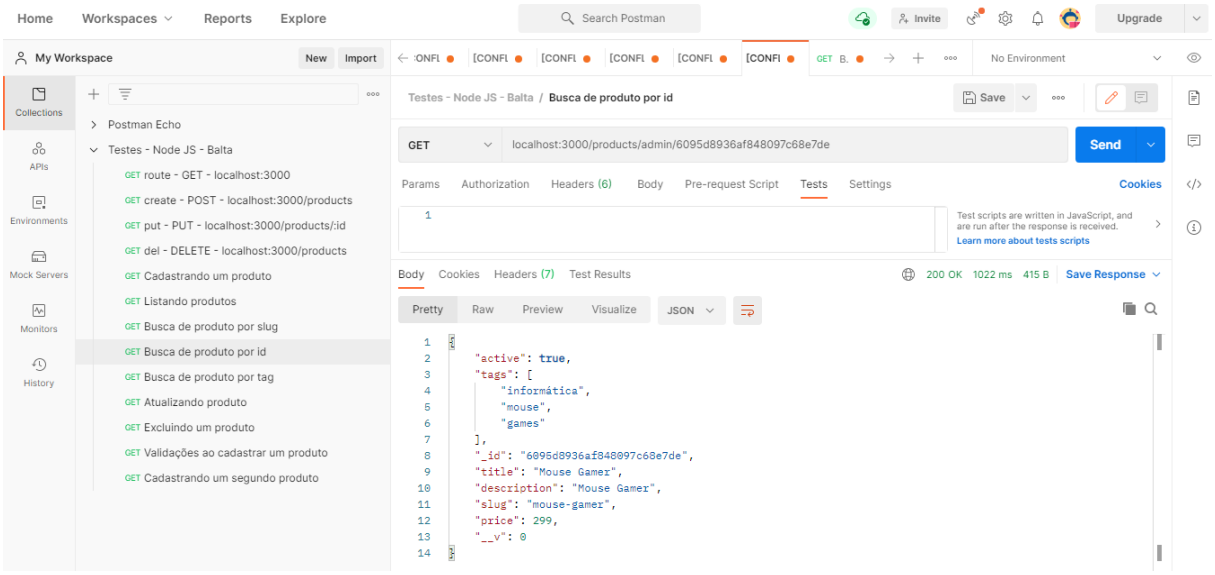
Exibindo dados de um produto por slug

GET - localhost:3000/products/mouse-gamer



Exibindo dados de um produto por id

localhost:3000/products/admin/6095d8936af848097c68e7de



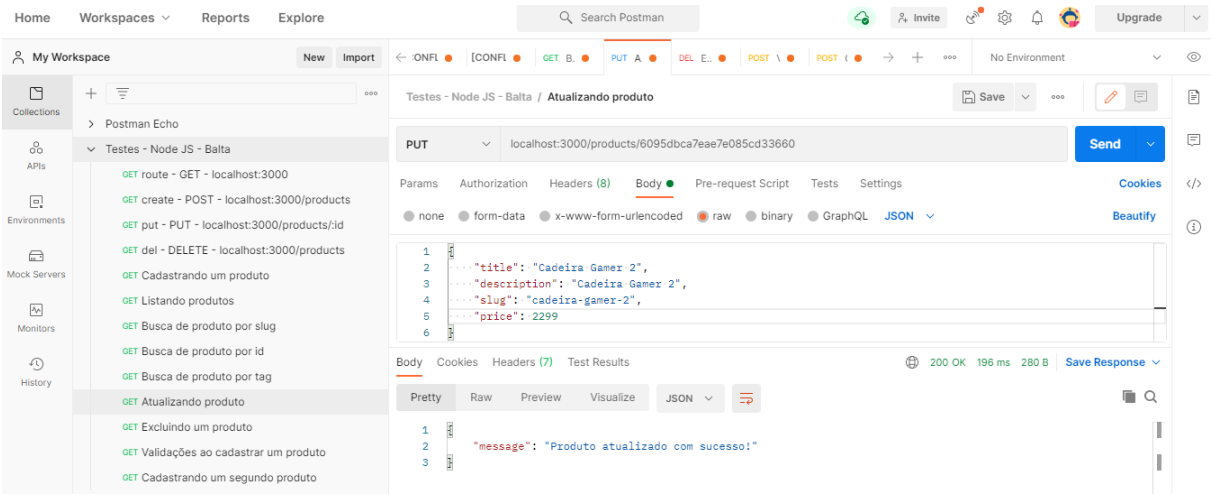
Exibindo produtos por tag

GET - localhost:3000/products/tags/games


```
[
    {
        "tags": [
            "informática",
            "mouse",
            "games"
        ],
        "_id": "6095d8936af848097c68e7de",
        "title": "Mouse Gamer",
        "description": "Mouse Gamer",
        "slug": "mouse-gamer",
        "price": 299
    },
    {
        "tags": [
            "informática",
            "mouse",
            "games"
        ],
        "_id": "6095dbca7eae7e085cd33660",
        "title": "Cadeira Gamer",
        "description": "Cadeira Gamer",
        "slug": "cadeira-gamer",
        "price": 1299
    }
]
```

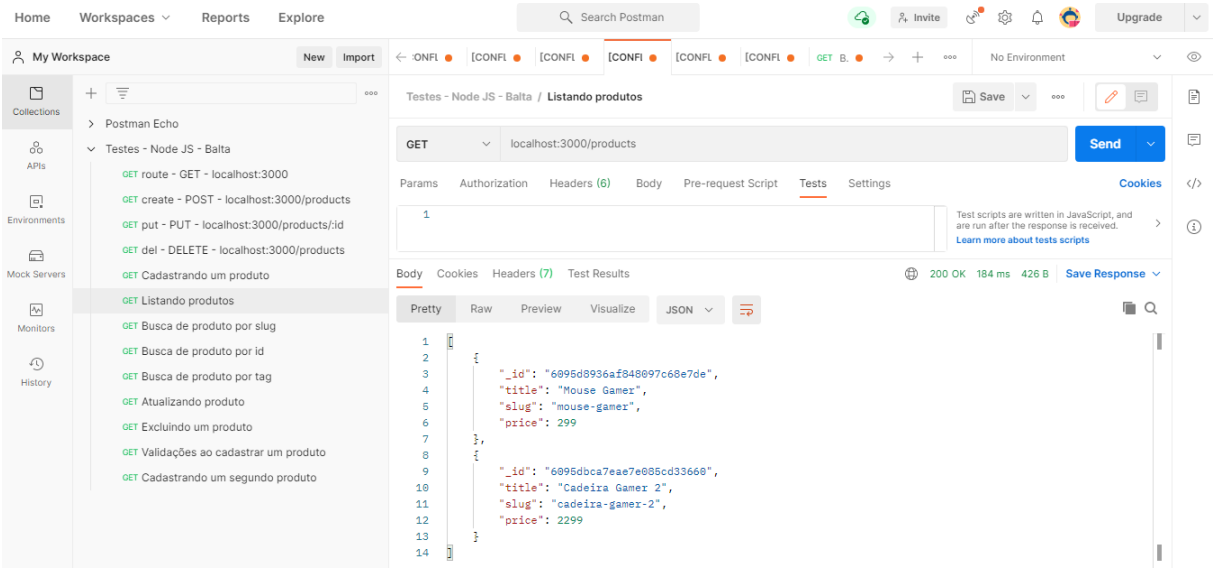
Atualizando dados de um produto

localhost:3000/products/6095dbca7eae7e085cd33660



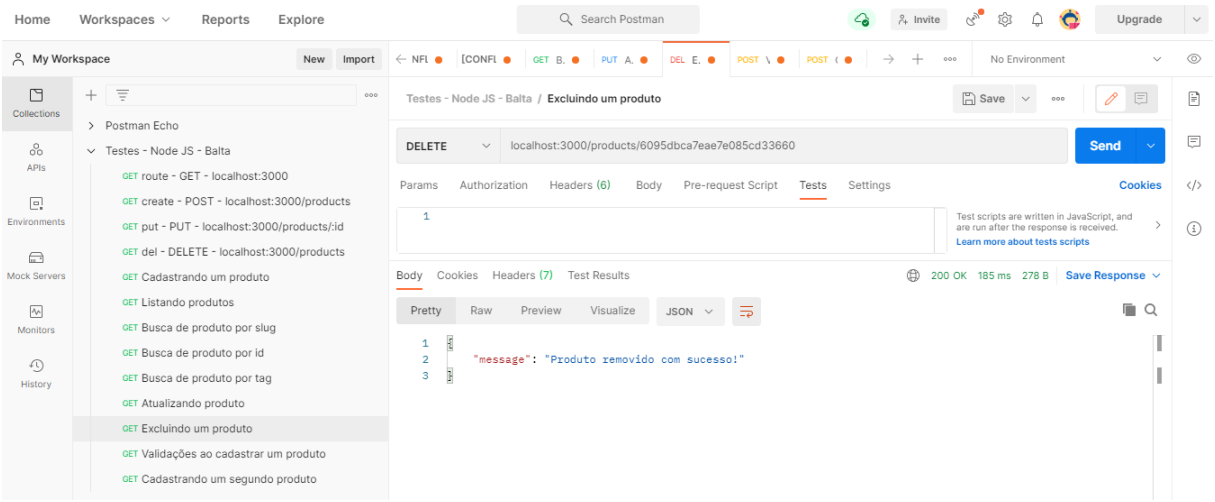
Listando os produtos

GET - localhost:3000/products



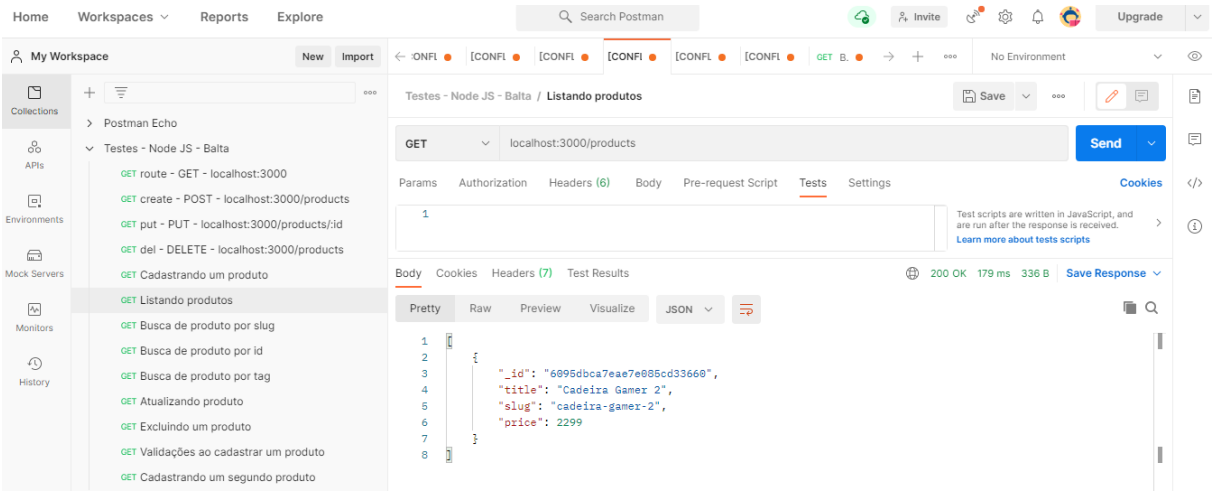
Excluindo um produto

DELETE - localhost:3000/products/5d5c6627cc81b832f8f37dd9



Listando os produtos

GET - localhost:3000/products



Aula 25 - Async / Await

src/repositories/product-repository.js

```
'use strict';
const mongoose = require('mongoose');
const Product = mongoose.model('Product');

exports.get = async() => {
    const res = await Product.find({
        active: true
    }, 'title price slug');
    return res;
}

exports.getBySlug = async(slug) => {
    const res = await Product
        .findOne({
            slug: slug,
            active: true
        }, 'title description price slug tags');
    return res;
}

exports.getById = async(id) => {
    const res = await Product
        .findById(id);
    return res;
}

exports.getByTag = async(tag) => {
    const res = Product
        .find({
            tags: tag,
            active: true
        }, 'title description price slug tags');
    return res;
}

exports.create = async(data) => {
    var product = new Product(data);
    await product.save();
}

exports.update = async(id, data) => {
    await Product
        .findByIdAndUpdate(id, {
            $set: {
                title: data.title,
                description: data.description,
                price: data.price,
                slug: data.slug
            }
        });
}
```

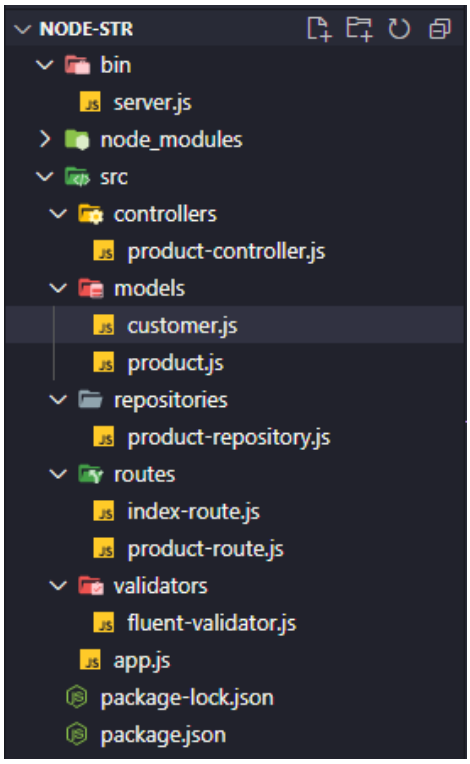
```
exports.delete = async(id) => {
    await Product
        .findOneAndRemove(id);
}
```

src/controllers/product-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/product-repository');

exports.get = async(req, res, next) => {
    try {
        var data = await repository.get();
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.getBySlug = async(req, res, next) => {
    try {
        var data = await repository.getBySlug(req.params.slug);
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.getById = async(req, res, next) => {
    try {
        var data = await repository.getById(req.params.id);
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.getByTag = async(req, res, next) => {
    try {
        const data = await repository.getByTag(req.params.tag);
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
```

```
}

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.title, 3, 'O título deve conter pelo menos 3 caracteres');
    contract.hasMinLen(req.body.slug, 3, 'O título deve conter pelo menos 3 caracteres');
    contract.hasMinLen(req.body.description, 3, 'O título deve conter pelo menos 3
caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create(req.body);
        res.status(201).send({
            message: 'Produto cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

exports.put = async(req, res, next) => {
    try {
        await repository.update(req.params.id, req.body);
        res.status(200).send({
            message: 'Produto atualizado com sucesso!'
        });
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

exports.delete = async(req, res, next) => {
    try {
        await repository.delete(req.body.id)
        res.status(200).send({
            message: 'Produto removido com sucesso!'
        });
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

Aula 26 - Revisitando os Models: Customer



src/models/customer.js

```
'use strict';

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const schema = new Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    password: {
        type: String,
        required: true
    }
});

module.exports = mongoose.model('Customer', schema);
```

src/app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-
store-cluster-nlcnv.mongodb.net/node-str-db?retryWrites=true&w=majority", {
useNewUrlParser: true });

// Carrega os models
const Product = require('./models/product');
const Customer = require('./models/customer');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);

module.exports = app;
```

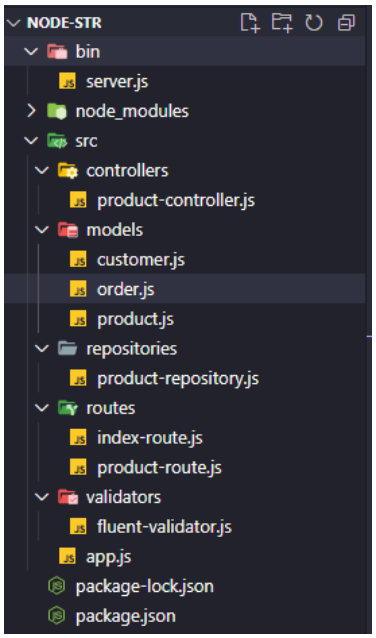
Aula 27 - Revisitando os Models: Order

src/models/order.js

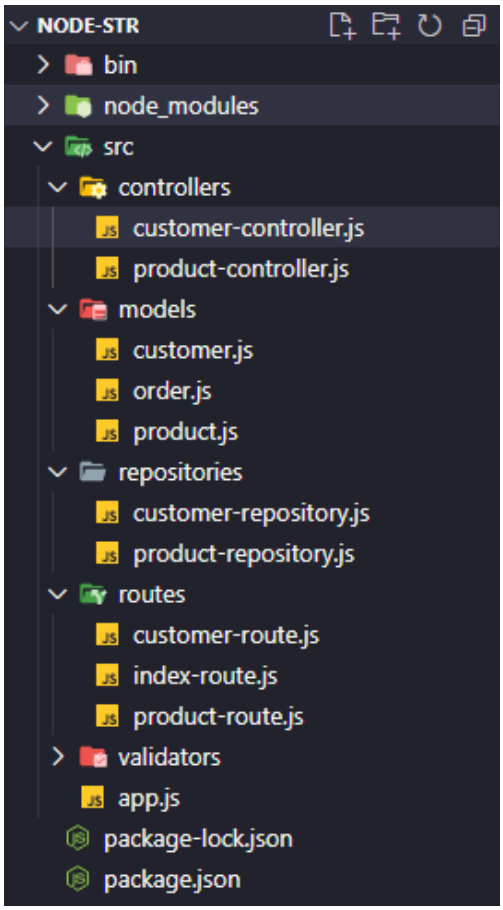
```
'use strict';

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const schema = new Schema({
    customer: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Customer'
    },
    number: {
        type: String,
        required: true
    },
    createDate: {
        type: Date,
        required: true,
        default: Date.now
    },
    status: {
        type: String,
        required: true,
        enum: ['created', 'done'],
        default: 'created'
    },
    items: [{
        quantity: {
            type: Number,
            required: true,
            default: 1
        },
        price: {
            type: Number,
            required: true
        },
        product: {
            type: mongoose.Schema.Types.ObjectId,
            ref: 'Product'
        }
    }],
});

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

src/app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-store-cluster-nlcnv.mongodb.net/node-str-db?retryWrites=true&w=majority", { useNewUrlParser: true });

// Carrega os models
const Product = require('./models/product');
const Customer = require('./models/customer');
const Order = require('./models/order');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);

module.exports = app;
```



Aula 28 - Revisitando os Controllers: Customer



src/controllers/customer-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O nome deve conter pelo menos 3 
caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 
caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }
```

```
    try {
        await repository.create(req.body);
        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

src/repositories/customer-repository.js

```
'use strict';
const mongoose = require('mongoose');
const Customer = mongoose.model('Customer');

exports.create = async(data) => {
    var customer = new Customer(data);
    await customer.save();
}
```

src/routes/customer-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/customer-controller");

router.post('/', controller.post);

module.exports = router;
```

src/app.js

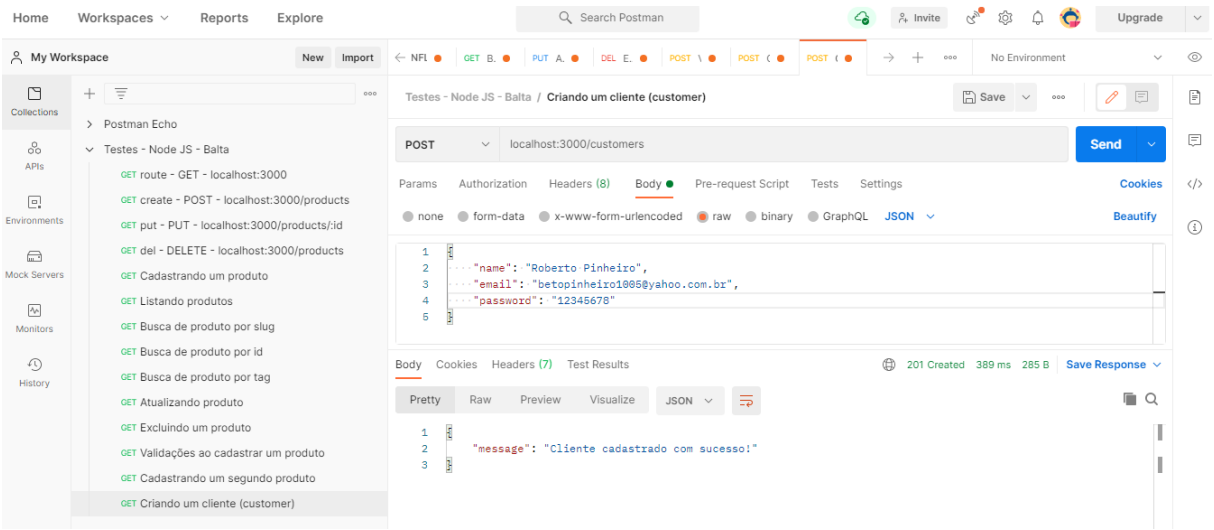
```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-store-cluster-nlcnv.mongodb.net/node-str-db?retryWrites=true&w=majority", {
useNewUrlParser: true });

// Carrega os models
const Product = require('./models/product');
const Customer = require('./models/customer');
const Order = require('./models/order');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');
const customerRoute = require('./routes/customer-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);
app.use('/customers', customerRoute);

module.exports = app;
```

nodemon ./bin/server.js

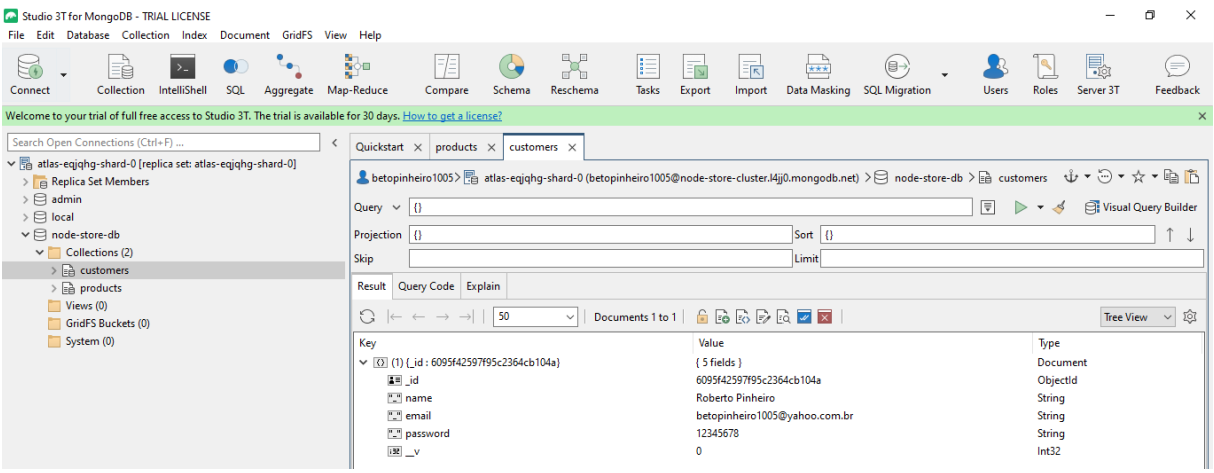
```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:9060) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:9060) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:9060) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(node:9060) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

Criando um cliente (customer)

POST - localhost:3000/customers



No Studio 3T:



Aula 29 - Revisitando os Controllers: Order

src/repositories/order-repository.js

```
'use strict';
const mongoose = require('mongoose');
const Order = mongoose.model('Order');

exports.get = async(data) => {
    var res = await Order.find({}, 'name status customer items')
    .populate('customer', 'name')
    .populate('items.product', 'title');
    return res;
}

exports.create = async(data) => {
    var order = new Order(data);
    await order.save();
}
```

Instalação do pacote guid

npm install guid --save

```
C:\balta\nodejs\node-str>npm install guid --save
npm WARN deprecated guid@0.0.12: Please use node-uuid instead. It is much better.
npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (c
urrent: {"os":"win32","arch":"x64"})

+ guid@0.0.12
added 1 package from 2 contributors and audited 212 packages in 9.7s

13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

src/controllers/order-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/order-repository');
const guid = require('guid');

exports.get = async(req, res, next) => {
    try {
        var data = await repository.get();
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.post = async(req, res, next) => {
    try {
        await repository.create({
            customer: req.body.customer,
            number: guid.raw().substring(0, 6),
            items: req.body.items
        });
        res.status(201).send({
            message: 'Pedido cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

src/routes/order-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/order-controller");

router.get('/', controller.get);
router.post('/', controller.post);

module.exports = router;
```

src/app.js

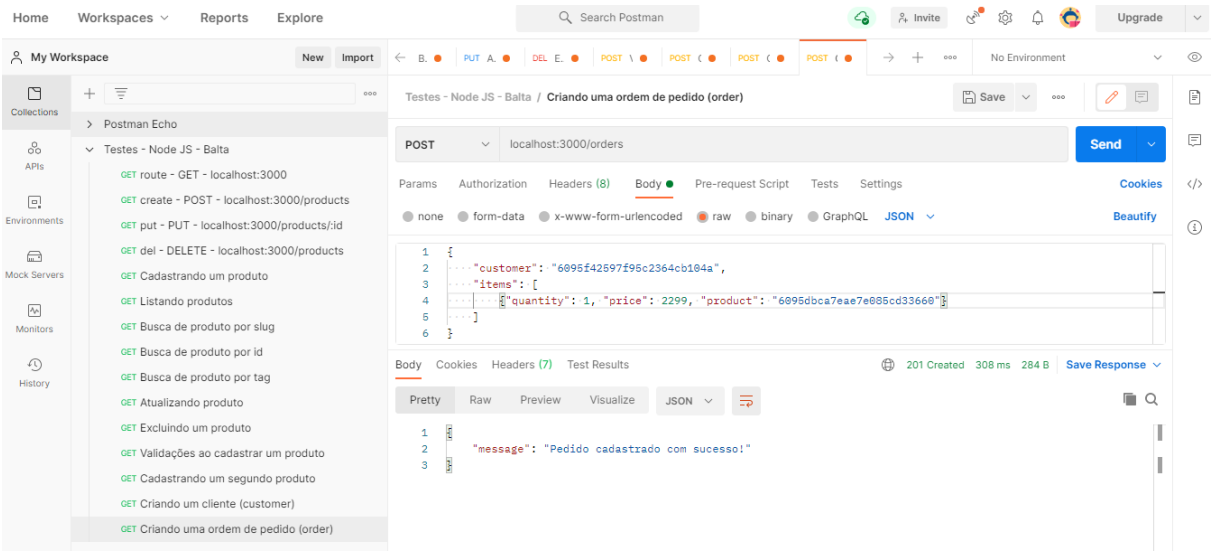
```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-store-cluster-nlcnv.mongodb.net/node-str-db?retryWrites=true&w=majority", {
useNewUrlParser: true });

// Carrega os models
const Product = require('./models/product');
const Customer = require('./models/customer');
const Order = require('./models/order');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');
const customerRoute = require('./routes/customer-route');
const orderRoute = require('./routes/order-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);
app.use('/customers', customerRoute);
app.use('/orders', orderRoute);

module.exports = app;
```

nodemon ./bin/server.js

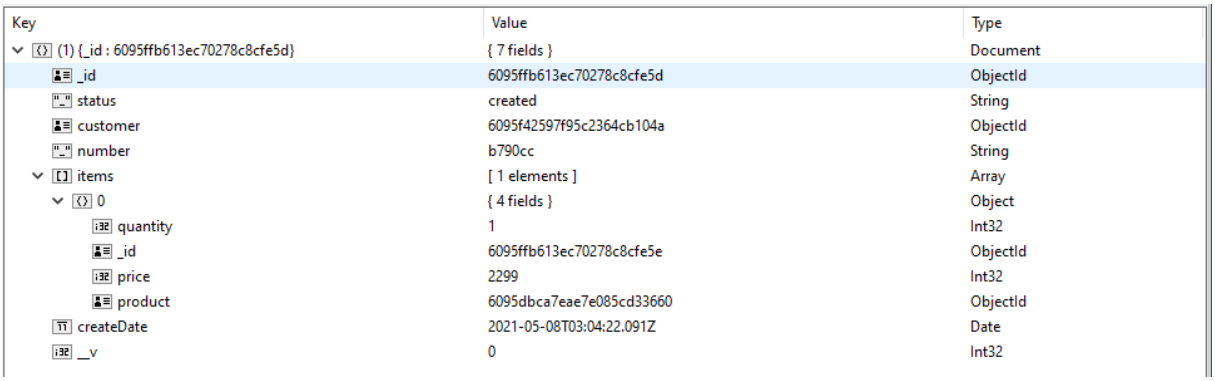
```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:10124) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
 new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:10124) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:10124) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
 future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
(node:10124) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

Testando no Postman

POST - localhost:3000/orders



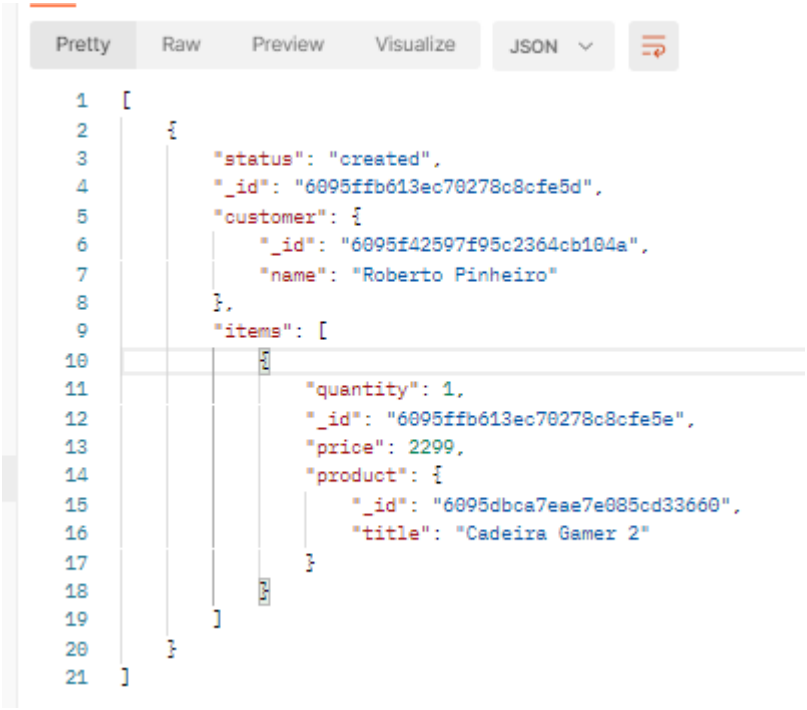
No Studio 3T:



Key	Value	Type
(1) {_id : 6095ffb613ec70278c8cfe5d}	{ 7 fields }	Document
_id	6095ffb613ec70278c8cfe5d	ObjectId
status	created	String
customer	6095f42597f95c2364cb104a	ObjectId
number	b790cc	String
items	[1 elements]	Array
0	{ 4 fields }	Object
quantity	1	Int32
_id	6095ffb613ec70278c8cfe5e	ObjectId
price	2299	Int32
product	6095dbca7eae7e085cd33660	ObjectId
createDate	2021-05-08T03:04:22.091Z	Date
__v	0	Int32

Listando os pedidos no Postman

GET - localhost:3000/orders



Aula 30 - Arquivo de configurações

src/config.js

```
global.SALT_KEY = 'f5b99242-6504-4ca3-90f2-05e78e5761ef';
global.EMAIL_TMPL = 'Olá, <strong>{0}</strong>, seja bem vindo à Node Store!';

module.exports = {
    connectionString: 'mongodb+srv://betopinheiro1005:angstron1005@node-store-
cluster.l4jj0.mongodb.net/node-store-db?retryWrites=true&w=majority',
    sendgridKey: 'TBD',
    containerConnectionString: 'TBD'
}
```

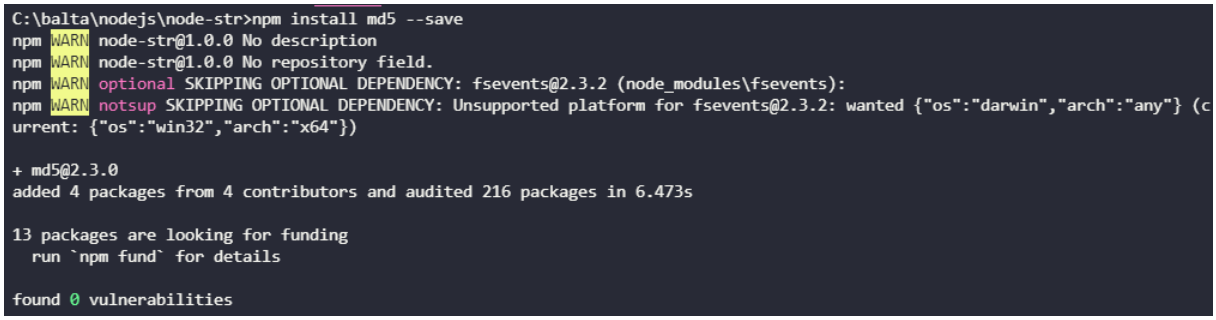
src/app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const config = require('./config');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect(config.connectionString);

// Carrega os models
const Product = require('./models/product');
const Customer = require('./models/customer');
const Order = require('./models/order');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');
const customerRoute = require('./routes/customer-route');
const orderRoute = require('./routes/order-route');

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

app.use('/', indexRoute);
app.use('/products', productRoute);
app.use('/customers', customerRoute);
app.use('/orders', orderRoute);

module.exports = app;
```

Aula 31 - Encriptando a senha

Instalação do md5

npm install md5 --save



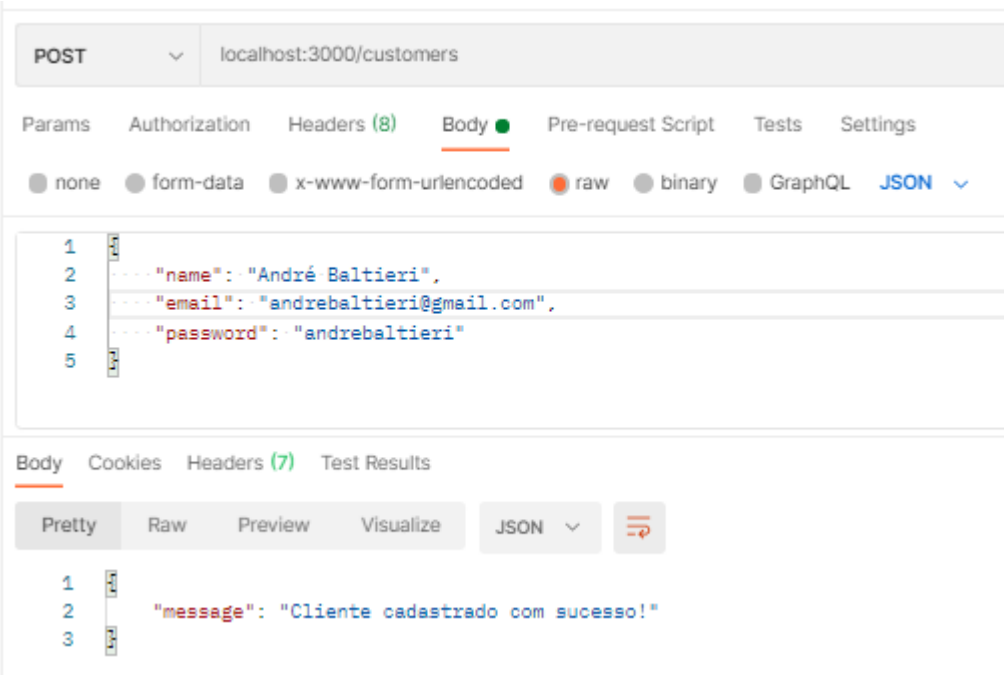
src/controllers/customer-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O título deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });
        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
```

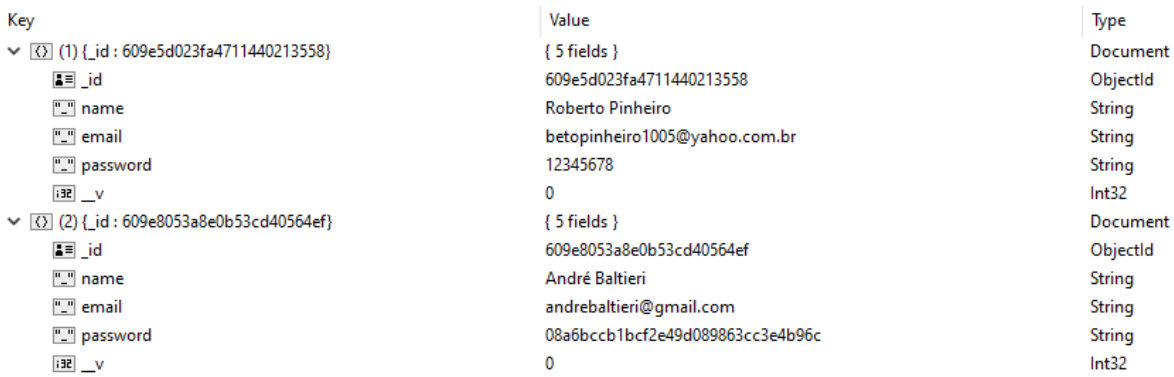
```
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

Criando um segundo cliente

POST - localhost:3000/customers



No Studio 3T

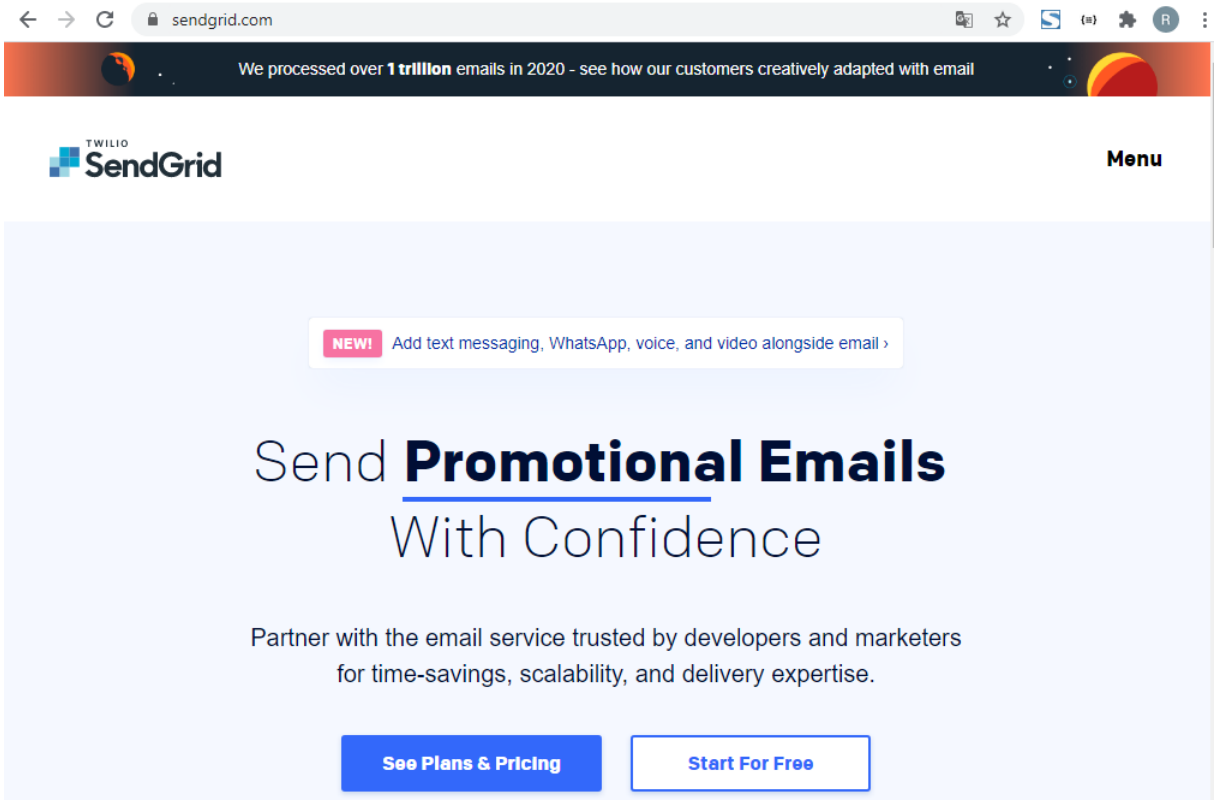


- Repare que a senha do segundo cliente está encriptada.

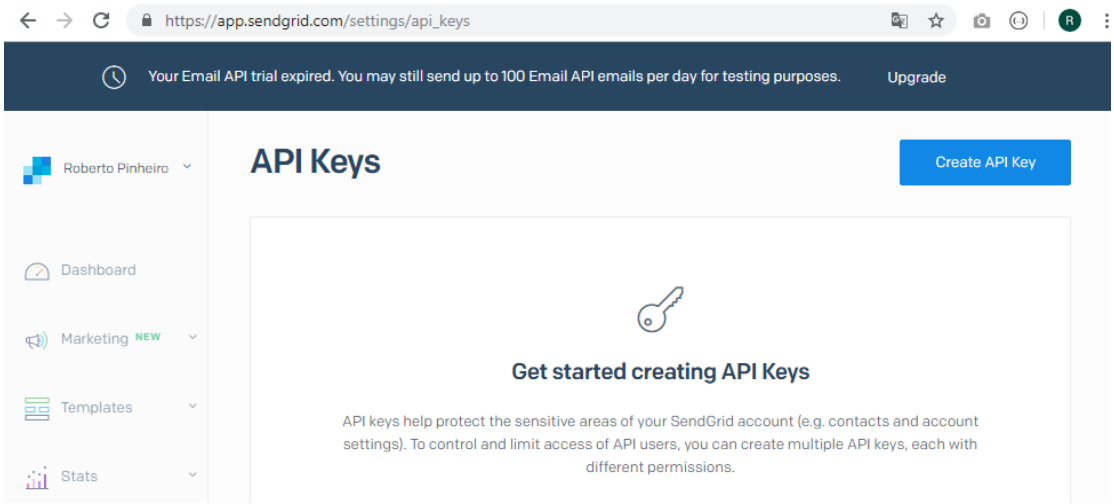
Aula 32 - Enviando email de boas vindas

SendGrid

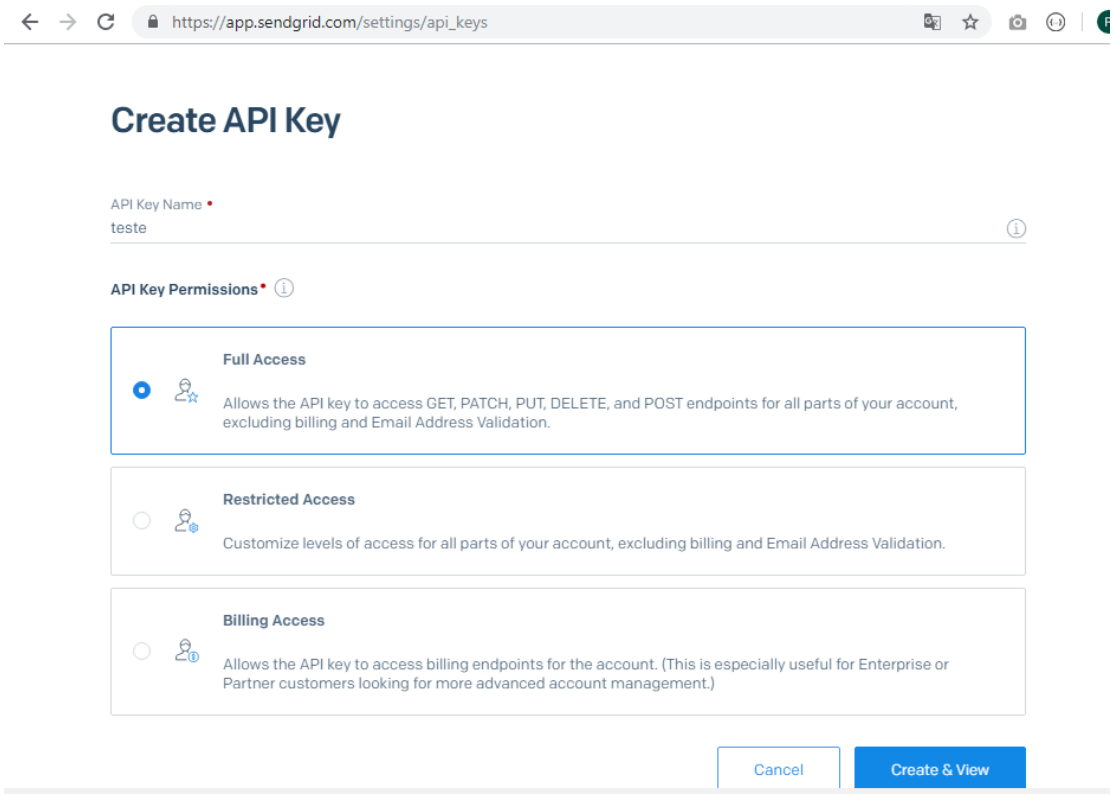
- Acesse https://sendgrid.com/ e abra uma conta.

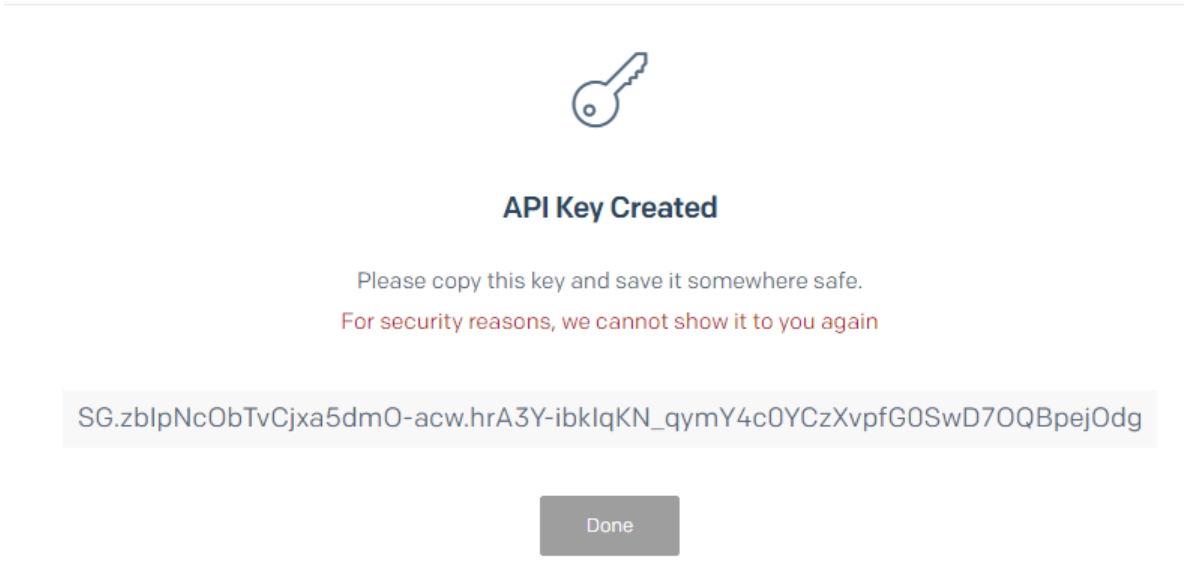


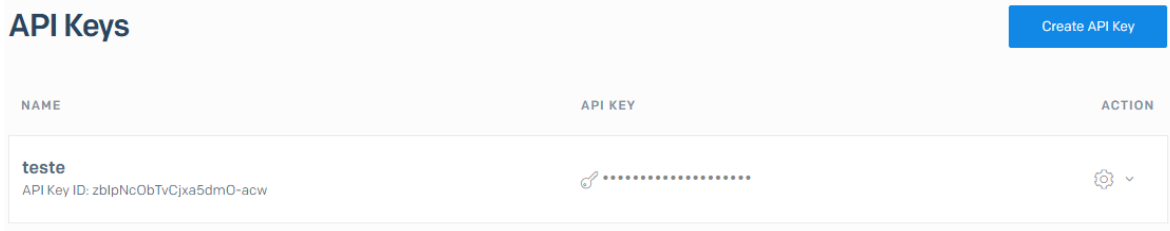
- Crie uma API Key



- Nomeie a API de teste e dê acesso total (full access):







- Copie a chave e cole-a no arquivo config.js:

src/config.js

```
global.SALT_KEY = 'f5b99242-6504-4ca3-90f2-05e78e5761ef';
global.EMAIL_TMPL = 'Olá, <strong>{0}</strong>, seja bem vindo à Node Store!';

module.exports = {
    connectionString: 'mongodb+srv://betopinheiro1005:angstron1005@node-store-cluster.l4jj0.mongodb.net/node-store-db?retryWrites=true&w=majority ',
    sendgridKey: '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',
    containerConnectionString: 'TBD'
}
```

Instalação do SendGrid

npm install sendgrid@2.0.0 --save

```
C:\balta\nodejs\node-str>npm install sendgrid@2.0.0 --save
npm WARN deprecated sendgrid@2.0.0: Please see v6.X+ at https://www.npmjs.com/org/sendgrid
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
npm WARN deprecated har-validator@5.1.5: this library is no longer supported
npm WARN deprecated core-js@2.6.12: core-js@<3 is no longer maintained and not recommended for usage due to the number of iss
ues. Please, upgrade your dependencies to the actual version of core-js@3.

> core-js@2.6.12 postinstall C:\balta\nodejs\node-str\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (c
urrent: {"os":"win32","arch":"x64"})

+ sendgrid@2.0.0
added 141 packages from 200 contributors and audited 357 packages in 190.45s

15 packages are looking for funding
  run `npm fund` for details

found 7 vulnerabilities (3 low, 1 moderate, 3 high)
```

package.json

```
{
 "name": "node-str",
 "version": "1.0.0",
 "description": "",
 "main": "index.js",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "start": "node ./bin/server.js"
 },
 "keywords": [],
 "author": "",
 "license": "ISC",
 "dependencies": {
   "body-parser": "^1.19.0",
   "debug": "^4.3.1",
   "express": "^4.17.1",
   "guid": "0.0.12",
   "http": "0.0.1-security",
   "md5": "^2.3.0",
   "mongoose": "^5.12.7",
   "sendgrid": "^2.0.0"
 },
 "devDependencies": {
   "nodemon": "^2.0.7"
 }
}
```

src/services/email-service.js

```
'use strict';
var config = require('../config');
var sendgrid = require('sendgrid')(config.sendgridKey);

exports.send = async (to, subject, body) => {
    sendgrid.send({
        to: to,
        from: 'andrebaltieri@balta.io',
        subject: subject,
        html: body
    });
}
```

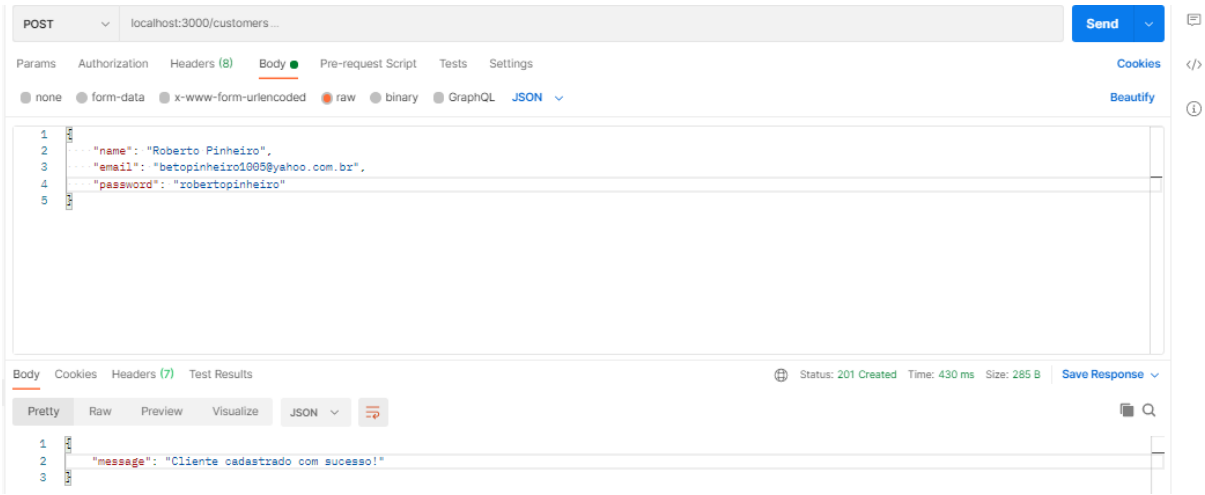
src/controllers/customer-controller.js

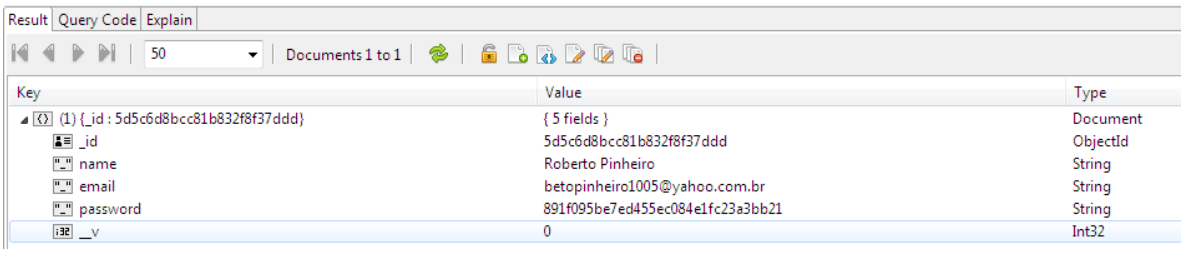
```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');
const emailService = require('../services/email-service');

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O título deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });

        emailService.send(
            req.body.email,
            'Bem vindo ao Node Store',
            global.EMAIL_TMPL.replace('{0}', req.body.name)
        );

        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

- No Studio 3T, apague os clientes cadastrados.

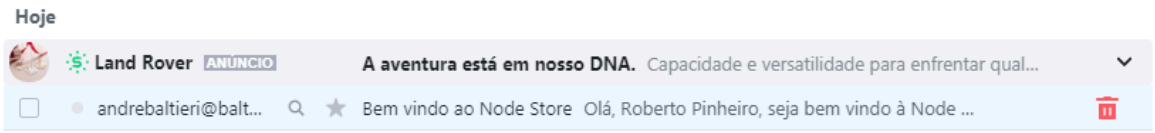
- Crie um novo cliente.

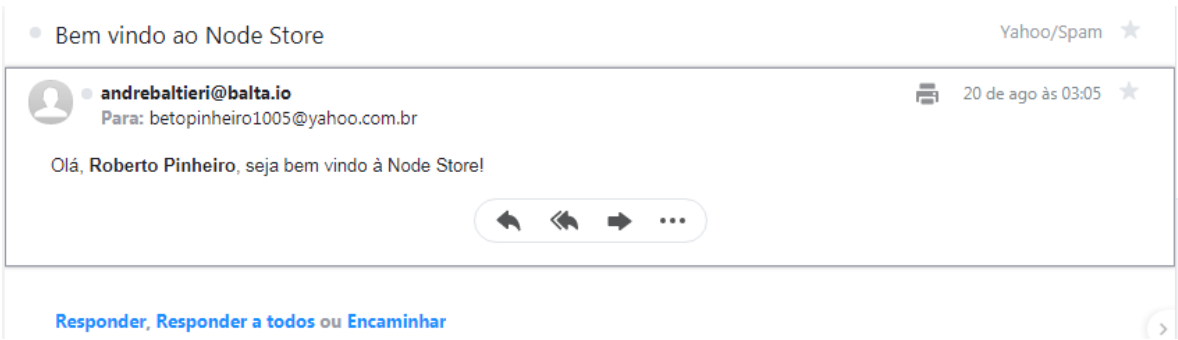
No Postman





- Ao realizar o cadastro, o cliente irá receber em seu email a seguinte mensagem:

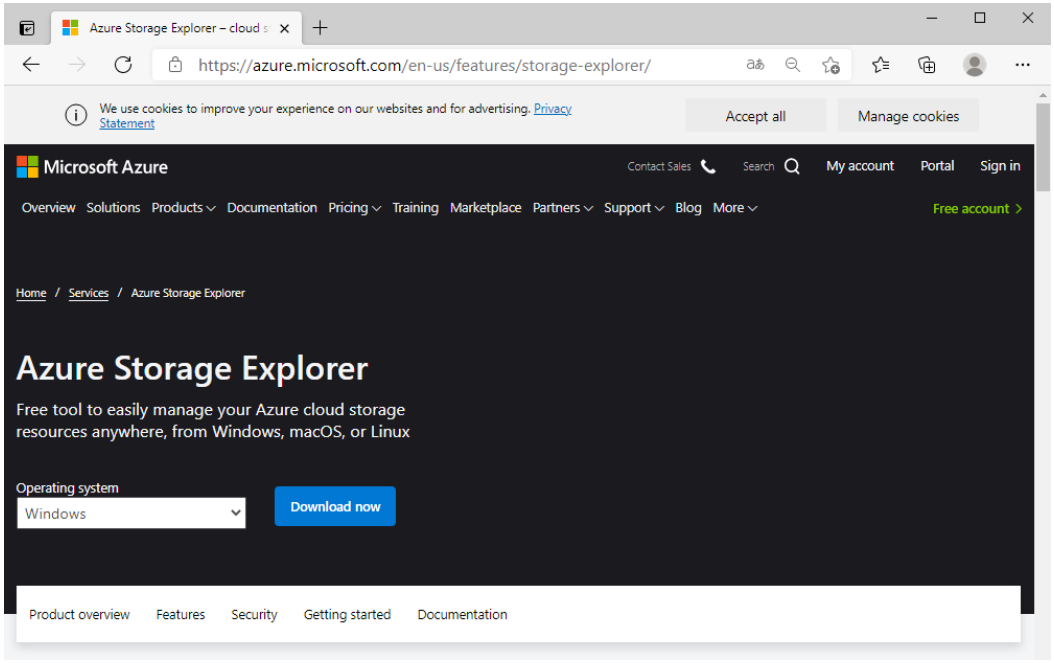




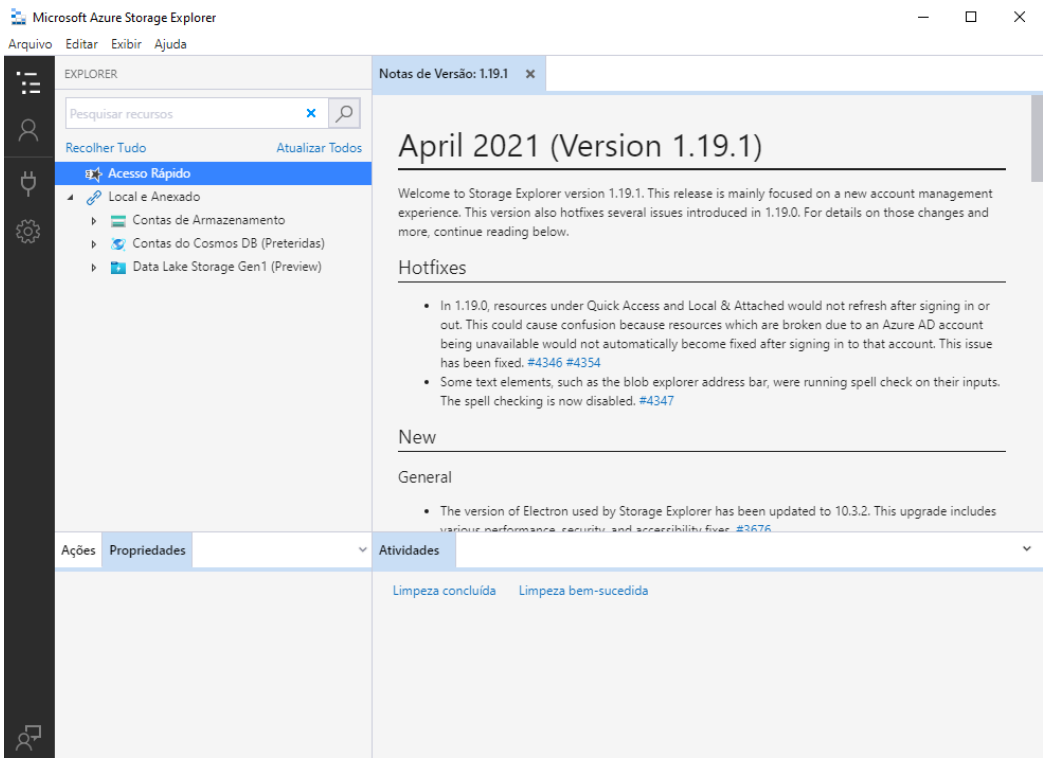
Aula 33 - Upload da imagem do produto

Acesse a URL:

https://azure.microsoft.com/en-us/features/storage-explorer/

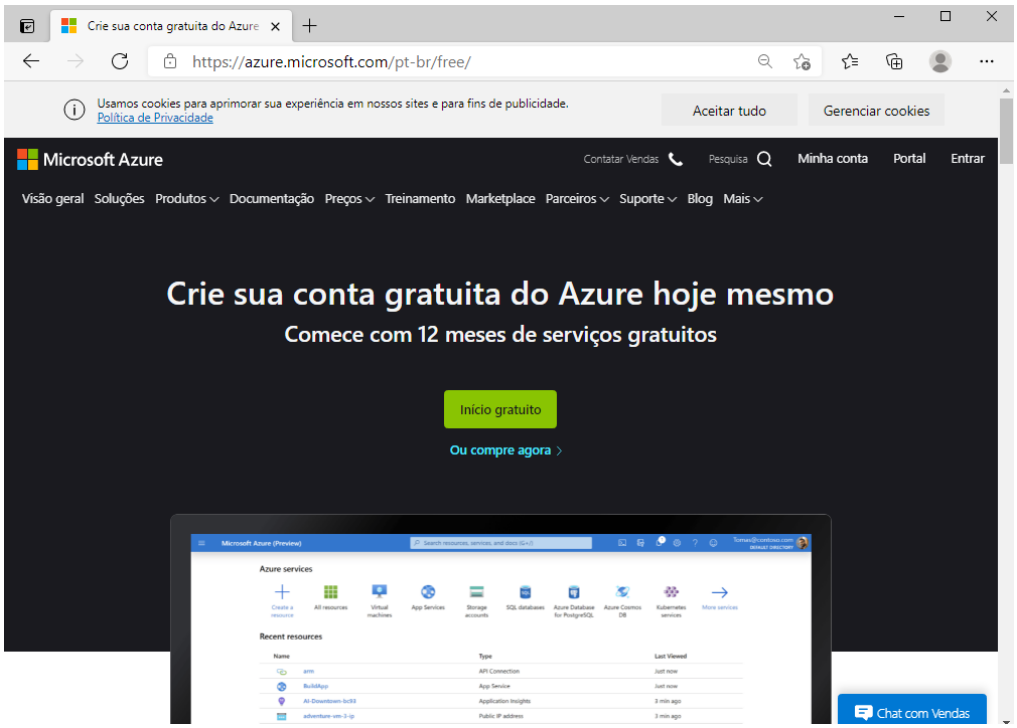


- Baixe e instale o programa Microsoft Azure Storage Explorer free.

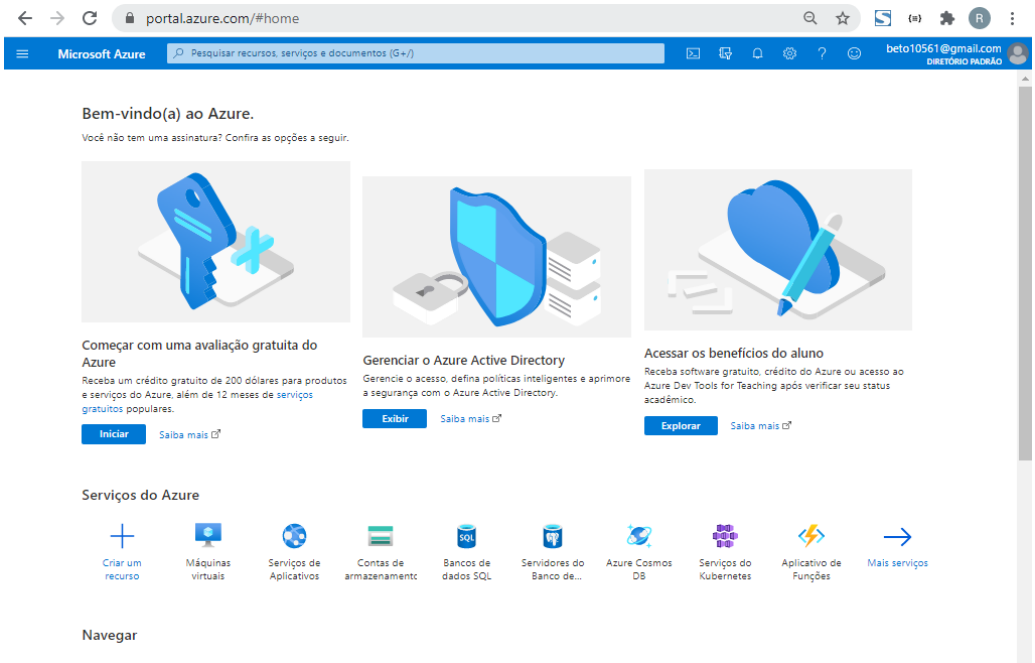


- Crie sua conta no Azzure:

https://azure.microsoft.com/pt-br/free/



- Acesse o portal.



- Crie um Storage Account.

Aula 34 - Autenticação

- Vamos utilizar autenticação via token com JWT.

Instalando o pacote JWT

npm install jsonwebtoken@7.4.0 --save

```
C:\balta\nodejs\node-str>npm install jsonwebtoken@7.4.0 --save
npm WARN deprecated joi@6.10.1: This version has been deprecated in accordance with the hapi support policy (hapi.im/support)
. Please upgrade to the latest version to get the best features, bug fixes, and security patches. If you are unable to upgrad
e at this time, paid support is available for older versions (hapi.im/commercial).
npm WARN deprecated topo@1.1.0: This version has been deprecated in accordance with the hapi support policy (hapi.im/support)
. Please upgrade to the latest version to get the best features, bug fixes, and security patches. If you are unable to upgrad
e at this time, paid support is available for older versions (hapi.im/commercial).
npm WARN deprecated hoek@2.16.3: This version has been deprecated in accordance with the hapi support policy (hapi.im/support
). Please upgrade to the latest version to get the best features, bug fixes, and security patches. If you are unable to upgra
de at this time, paid support is available for older versions (hapi.im/commercial).
npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (c
urrent: {"os":"win32","arch":"x64"})

+ jsonwebtoken@7.4.0
added 13 packages from 18 contributors and audited 370 packages in 43.908s

15 packages are looking for funding
  run `npm fund` for details

found 9 vulnerabilities (3 low, 3 moderate, 3 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

src\services\auth-service.js

```
'use strict';
const jwt = require('jsonwebtoken');

exports.generateToken = async (data) => {
    return jwt.sign(data, global.SALT_KEY, { expiresIn: '1d' });
}

exports.decodeToken = async (token) => {
    var data = await jwt.verify(token, global.SALT_KEY);
    return data;
}

exports.authorize = function (req, res, next) {
    var token = req.body.token || req.query.token || req.headers['x-access-token'];

    if (!token) {
        res.status(401).json({
            message: 'Acesso Restrito'
        });
    } else {
        jwt.verify(token, global.SALT_KEY, function (error, decoded) {
            if (error) {
                res.status(401).json({
                    message: 'Token Inválido'
                });
            } else {
                next();
            }
        });
    }
};
```

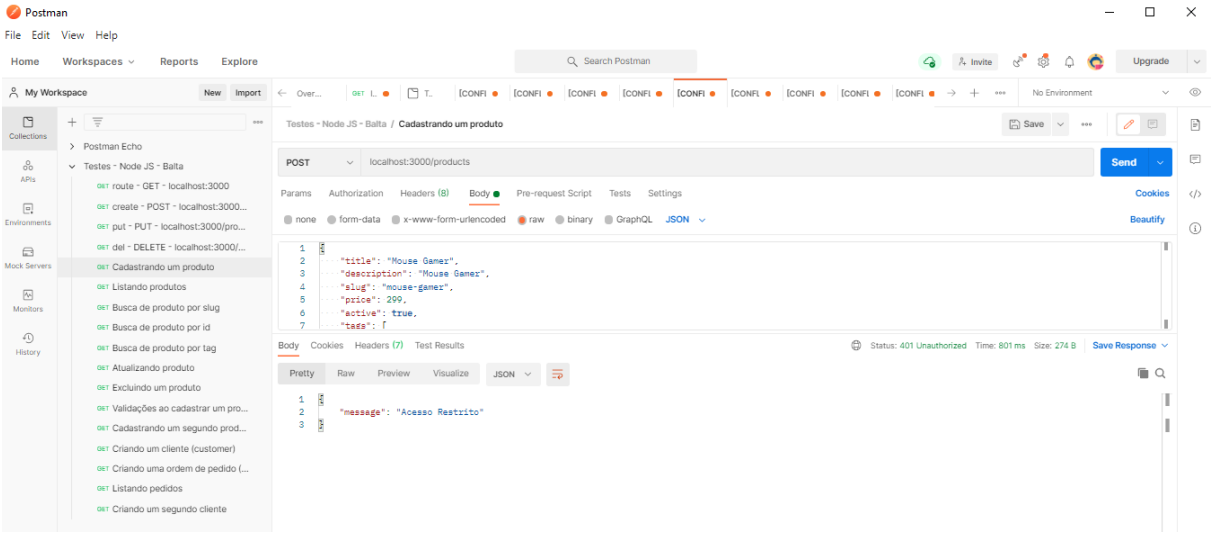
src\routes\product-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");
const authService = require('../services/auth-service');

router.get('/', controller.get);
router.get('/:slug', controller.getBySlug);
router.get('/admin/:id', controller.getById);
router.get('/tags/:tag', controller.getByTag);
router.post('/', authService.authorize, controller.post);
router.put('/:id', authService.authorize, controller.put);
router.delete('/:id', authService.authorize, controller.delete);

module.exports = router;
```

nodemon ./bin/server.js

```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:14384) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
 new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:14384) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:14384) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
 future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoC
lient constructor.
(node:14384) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

- Ao tentar criar um produto:



src\repositories\customer-repository.js

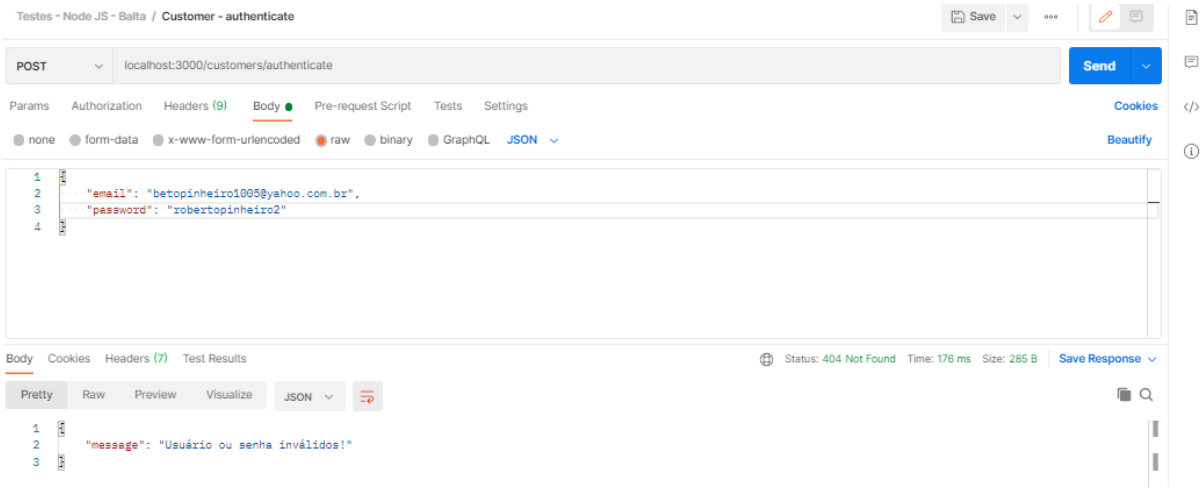
```
'use strict';
const mongoose = require('mongoose');
const Customer = mongoose.model('Customer');

exports.create = async(data) => {
    var customer = new Customer(data);
    await customer.save();
}

exports.authenticate = async(data) => {
    const res = await Customer.findOne({
        email: data.email,
        password: data.password
    });
    return res;
}
```

src\controllers\customer-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');
const authService = require('../services/auth-service');

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O nome deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        }
);
        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

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

    try {
        const customer = await repository.authenticate({
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });

        if(!customer){
            res.status(404).send({
                message: 'Usuário ou senha inválido(s)!'
```

```
            });
            return;
        }

        const token = await authService.generateToken({
            email: customer.email,
            name: customer.name
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

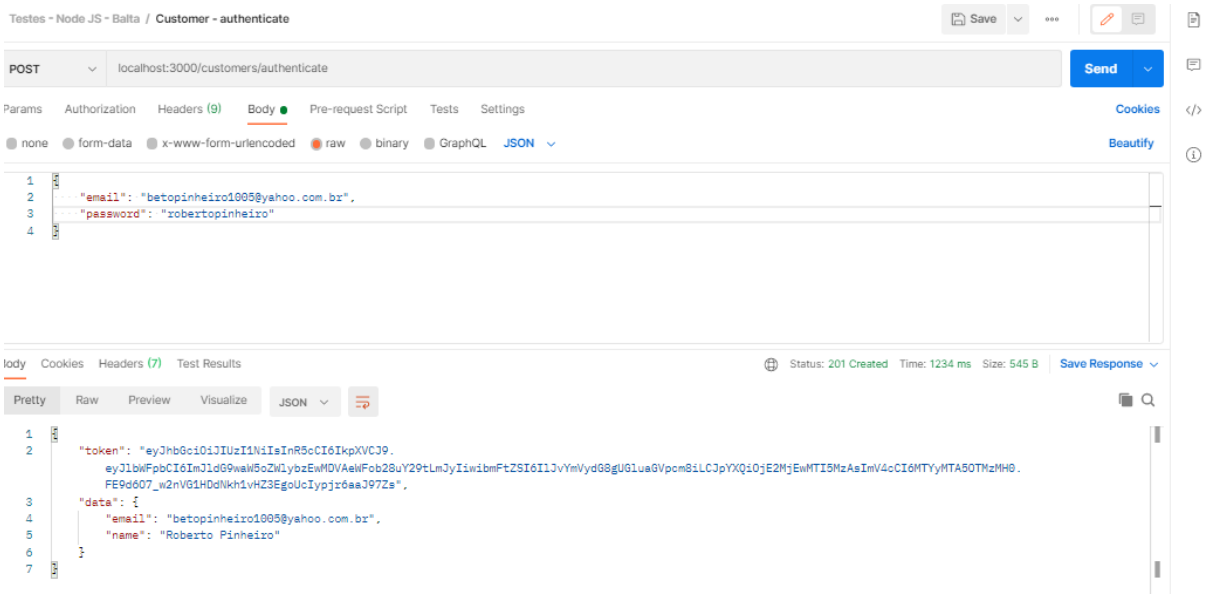
- No Postman, passando senha errada:



src\routes\customer-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/customer-controller");

router.post('/', controller.post);
router.post('/authenticate', controller.authenticate);

module.exports = router;
```

No Postman, passando a senha correta:




```
{
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImJldG9waW5oZWlybzEwMDVAe
WFob28uY29tLmJyIiwibmFtZSI6IlJvYmVydG8gUGluaGVpcm8iLCJpYXQiOjE2MjEwMTI5Mz
AsImV4cCI6MTYyMTA5OTMzMH0.FE9d6O7_w2nVG1HDdNkh1vHZ3EgoUcIypjr6aaJ97Zs",
   "data": {
      "email": "betopinheiro1005@yahoo.com.br",
      "name": "Roberto Pinheiro"
   }
}
```


- No Postman, na requisição que cadastra um produto, na aba Header, insira a key:

x-access-token

- Em value, cole o valor do token obtido ao cadastrar o cliente.

Aula 35 - Recuperando dados do usuário logado

src\routes\order-route.js

```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/order-controller");
const authService = require('../services/auth-service');

router.get('/', authService.authorize, controller.get);
router.post('/', authService.authorize, controller.post);

module.exports = router;
```

src\controllers\customer-controller.js

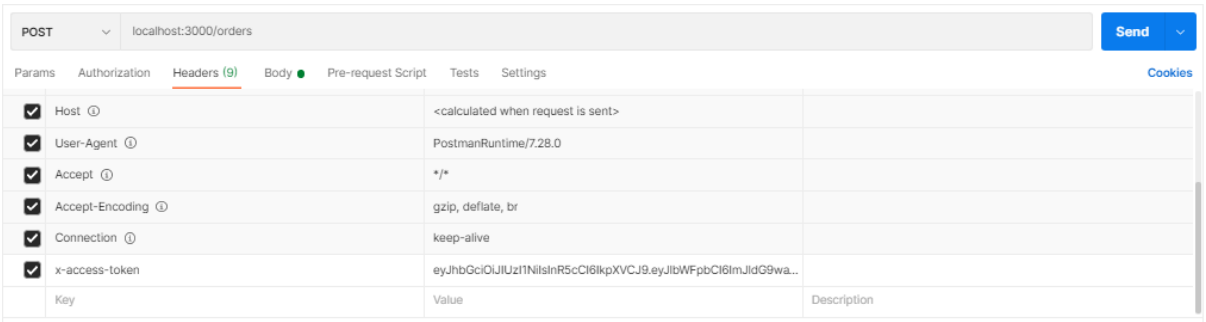
```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');
const emailService = require('../services/email-service');

const authService = require('../services/auth-service');

exports.get = async(req, res, next) => {
    try {
        var data = await repository.get();
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O nome deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            // password: req.body.password
```

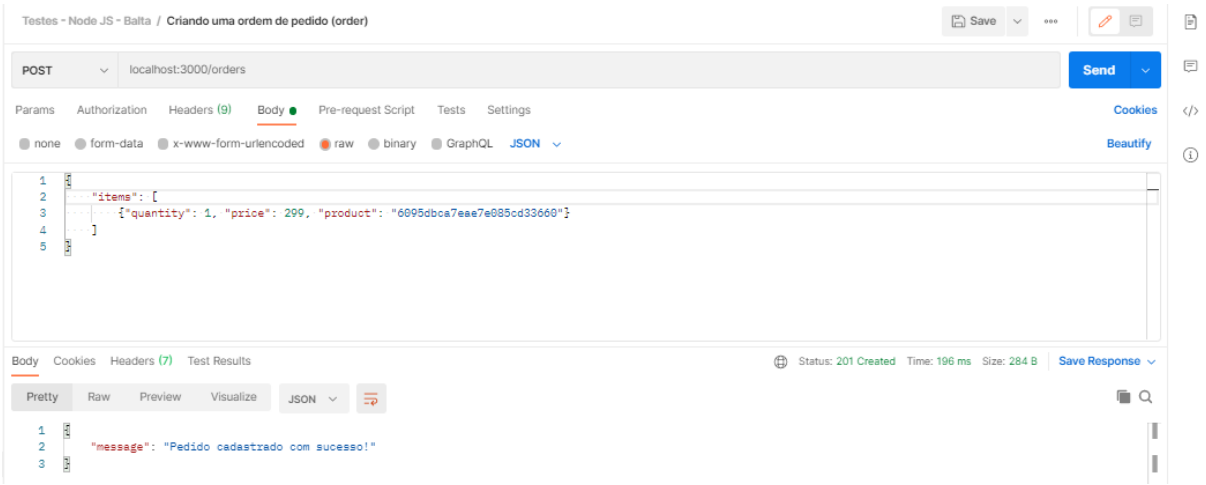
```
            password: md5(req.body.password + global.SALT_KEY)
        });

        emailService.send(
            req.body.email,
            'Bem vindo ao Node Store',
            global.EMAIL_TMPL.replace('{0}', req.body.name)
        );

        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

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

    try {
        const customer = await repository.authenticate({
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });

        if(!customer){
            res.status(404).send({
                message: 'Usuário ou senha inválidos!'
            });
            return;
        }

        const token = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

src\controllers\order-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/order-repository');
const guid = require('guid');
const authService = require('../services/auth-service');

exports.get = async(req, res, next) => {
   try {
      var data = await repository.get();
      res.status(200).send(data);
   } catch (e) {
      res.status(500).send({
         message: 'Falha ao processar sua requisição'
      });
   }
}

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

      const token = req.body.token || req.query.token || req.headers['x-access-token'];
      const data = await authService.decodeToken(token);

      await repository.create({
         customer: data.id,
         number: guid.raw().substring(0, 6),
         items: req.body.items
      });
      res.status(201).send({
         message: 'Pedido cadastrado com sucesso!'
      });
   } catch (e) {
      console.log(e);
      res.status(500).send({
         message: 'Falha ao processar sua requisição'
      });
   }
};
```

- No Postman:





Key	Value	Type
(1) {_id : 6095ffb613ec70278c8cfe5d}	{ 7 fields }	Document
(2) {_id : 609ec3dcee14102584139e92}	{ 6 fields }	Document
_id	609ec3dcee14102584139e92	ObjectId
status	created	String
number	5ac399	String
items	[1 elements]	Array
0	{ 4 fields }	Object
quantity	1	Int32
_id	609ec3dcee14102584139e93	ObjectId
price	299	Int32
product	6095dbca7eae7e085cd33660	ObjectId
createDate	2021-05-14T18:39:24.387Z	Date
__v	0	Int32

Aula 36 - Refresh Token

src\controllers\customer-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');
const emailService = require('../services/email-service');

const authService = require('../services/auth-service');

exports.get = async(req, res, next) => {
    try {
        var data = await repository.get();
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O nome deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            // password: req.body.password
            password: md5(req.body.password + global.SALT_KEY)
        });

        emailService.send(
            req.body.email,
            'Bem vindo ao Node Store',
            global.EMAIL_TMPL.replace('{0}', req.body.name)
        );

        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
```

```
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

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

    try {
        const customer = await repository.authenticate({
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });

        if(!customer){
            res.status(404).send({
                message: 'Usuário ou senha inválidos!'
            });
            return;
        }

        const token = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

exports.refreshToken = async(req, res, next) => {
    try {
        const token = req.body.token || req.query.token || req.headers['x-access-token'];
        const data = await authService.decodeToken(token);

        const customer = await repository.getById(data.id);

        if (!customer) {
            res.status(404).send({
                message: 'Cliente não encontrado'
            });
            return;
        }
```

```
        const tokenData = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name,
            roles: customer.roles
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

src\repositories\customer-repository.js

```
'use strict';
const mongoose = require('mongoose');
const Customer = mongoose.model('Customer');


exports.get = async() => {
    const res = await Customer.find({
    }, 'name email password');
    return res;
}

exports.create = async(data) => {
    var customer = new Customer(data);
    await customer.save();
}

exports.authenticate = async(data) => {
    const res = await Customer.findOne({
        email: data.email,
        password: data.password
    });
    return res;
}

exports.getById = async(id) => {
    const res = await Customer.findById(id);
    return res;
}
```

src\routes\customer-route.js

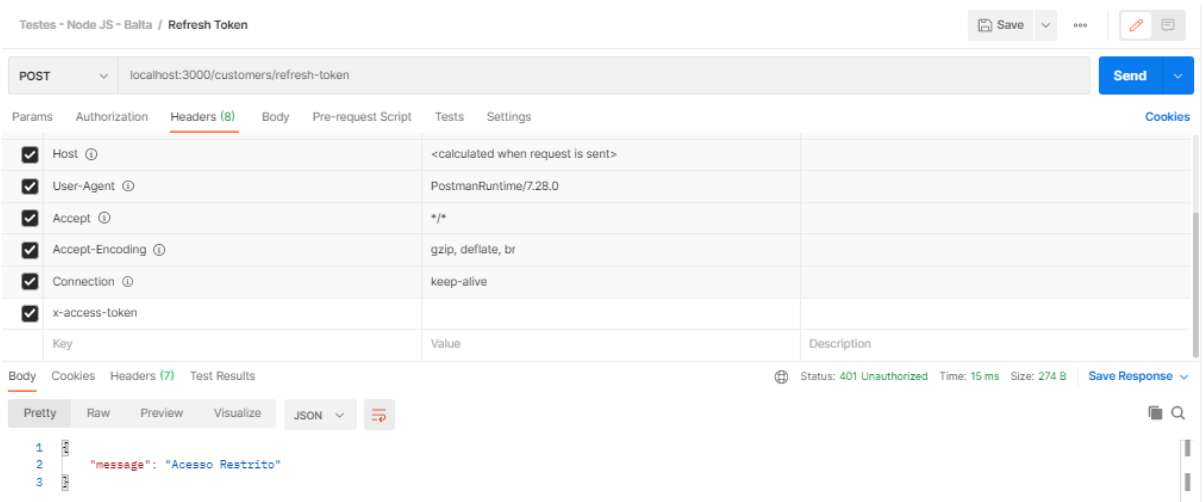
```
'use strict';

const express = require('express');
const router = express.Router();
const controller = require("../controllers/customer-controller");
const authService = require('../services/auth-service');

router.get('/', controller.get);
router.post('/', controller.post);
router.post('/authenticate', controller.authenticate);
router.post('/refresh-token', authService.authorize, controller.refreshToken);

module.exports = router;
```

nodemon ./bin/server.js

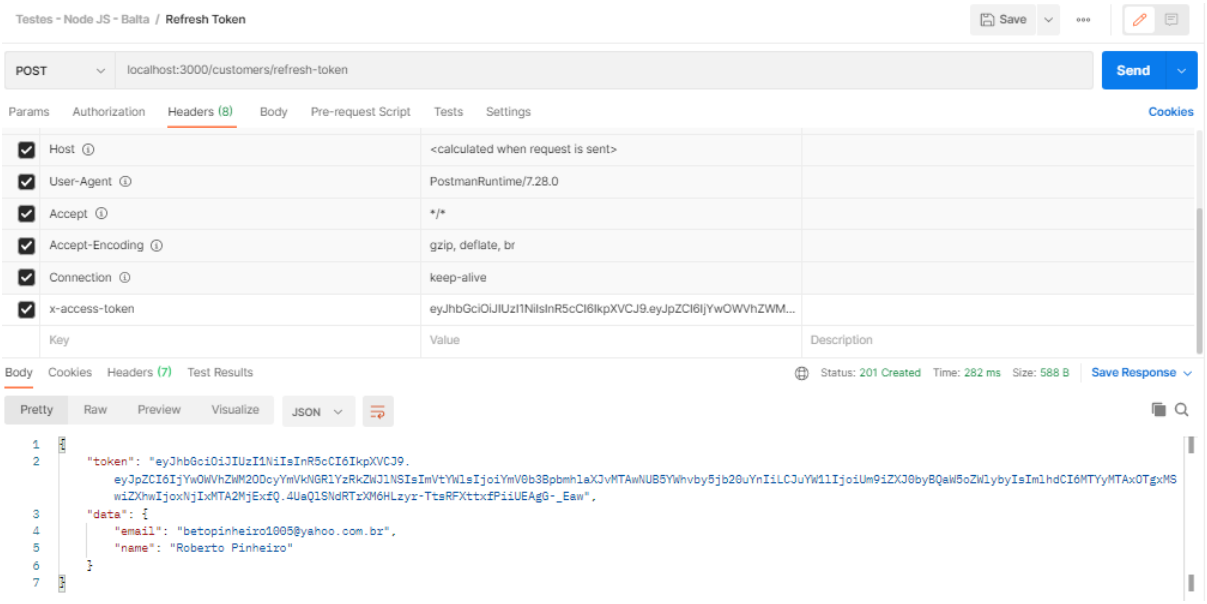
```
C:\balta\nodejs\node-str>nodemon ./bin/server.js
[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:7280) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the
new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:7280) [MONGODB DRIVER] Warning: Top-level use of w, wtimeout, j, and fsync is deprecated. Use writeConcern instead.
(node:7280) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a
future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoCl
ient constructor.
(node:7280) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

- No Postman, crie a requisição localhost:3000/customers/refresh-token:

- Sem o token na key x-access-token da aba Headers:



- Com o token ainda válido, na key x-access-token da aba Headers, ao clicar no botão Send é gerado um novo token:




```
{
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYwOWVhZWM2ODcyYmVkNGRlYzRkZWJlNSIsImVtYWlsIjoiYmV0b3BpbmhlaXJvMTAwNUB5YWhvby5jb20uYnIiLCJuYW1lIjoiUm9iZXJ0byBQaW5oZWlyby IsImlhdCI6MTYyMTAxOTgxMSwiZXhwIjoxNjIxMTA2MjExfQ.4UaQlSNdRTrXM6HLzyr-TtsRFXttxfPiiUEAgG-_Eaw",
   "data": {
      "email": "betopinheiro1005@yahoo.com.br",
      "name": "Roberto Pinheiro"
   }
}
```

Aula 37 - Autorização

src\models\customer.js

```
'use strict';

const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const schema = new Schema({
    name: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    password: {
        type: String,
        required: true
    },
    roles: [{
        type: String,
        required: true,
        enum: ['user', 'admin'],
        default: 'user'
    }]
});

module.exports = mongoose.model('Customer', schema);
```

src\controllers\customer-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');
const emailService = require('../services/email-service');

const authService = require('../services/auth-service');

exports.get = async(req, res, next) => {
    try {
        var data = await repository.get();
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
```

```
    }
}

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O nome deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY),
            roles: ["user"]
        });

        emailService.send(
            req.body.email,
            'Bem vindo ao Node Store',
            global.EMAIL_TMPL.replace('{0}', req.body.name)
        );

        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

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

    try {
        const customer = await repository.authenticate({
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });

        if(!customer){
            res.status(404).send({
                message: 'Usuário ou senha inválidos!'
            });
            return;
        }
```

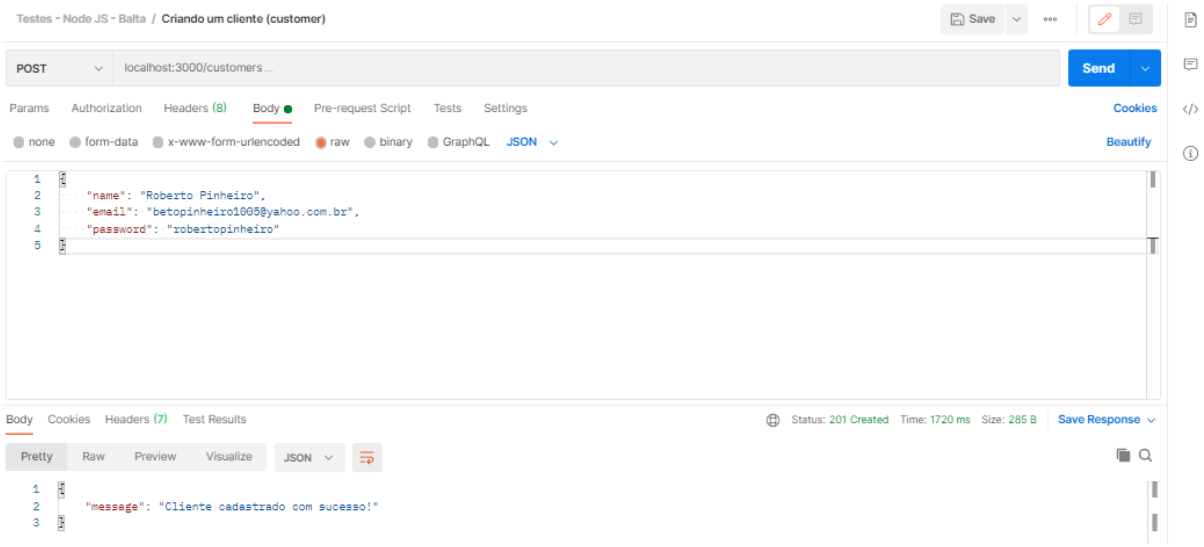
```
        const token = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

exports.refreshToken = async(req, res, next) => {
    try {
        const token = req.body.token || req.query.token || req.headers['x-access-token'];
        const data = await authService.decodeToken(token);

        const customer = await repository.getById(data.id);

        if (!customer) {
            res.status(404).send({
                message: 'Cliente não encontrado'
            });
            return;
        }

        const tokenData = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

- No Studio 3T, exclua o cliente.

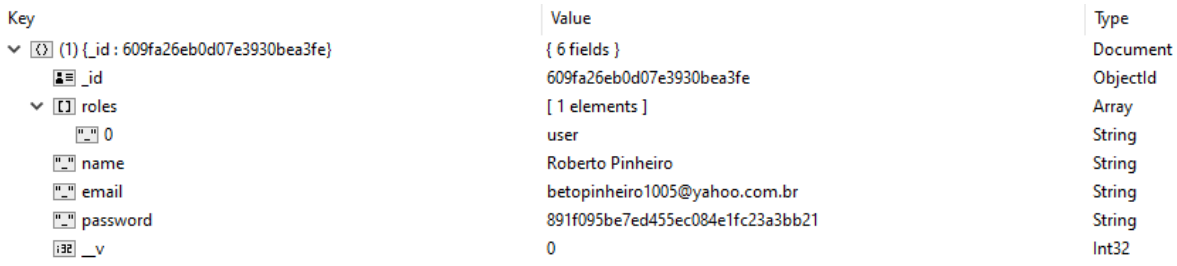
- Rode o servidor:

nodemon ./bin/server.js

- No Postman, crie um novo cliente(customer).



- No 3T Studio:



- Repare que o customer foi criado com role user (valor default).

- Para permitir que somente as pessoas com o role admin possam criar, editar ou excluir produtos, é necessário executar alguns passos a seguir:

src\services\auth-service.js

```
'use strict';
const jwt = require('jsonwebtoken');

exports.generateToken = async (data) => {
    return jwt.sign(data, global.SALT_KEY, { expiresIn: '1d' });
}

exports.decodeToken = async (token) => {
    var data = await jwt.verify(token, global.SALT_KEY);
    return data;
}

exports.authorize = function (req, res, next) {
    var token = req.body.token || req.query.token || req.headers['x-access-token'];

    if (!token) {
        res.status(401).json({
            message: 'Acesso Restrito'
        });
    } else {
        jwt.verify(token, global.SALT_KEY, function (error, decoded) {
            if (error) {
                res.status(401).json({
                    message: 'Token Inválido'
                });
            } else {
                next();
            }
        });
    }
};

exports.isAdmin = function (req, res, next) {
    var token = req.body.token || req.query.token || req.headers['x-access-token'];

    if (!token) {
        res.status(401).json({
            message: 'Token Inválido'
        });
    } else {
        jwt.verify(token, global.SALT_KEY, function (error, decoded) {
            if (error) {
                res.status(401).json({
                    message: 'Token Inválido'
                });
            } else {
                if (decoded.roles.includes('admin')) {
                    next();
                } else {
                    res.status(403).json({
                        message: 'Esta funcionalidade é restrita para administradores'
                    });
                }
            }
        });
```

```
    }
};
```

src\controllers\customer-controller.js

```
'use strict';

const ValidationContract = require('../validators/fluent-validator');
const repository = require('../repositories/customer-repository');
const md5 = require('md5');
const emailService = require('../services/email-service');

const authService = require('../services/auth-service');

exports.get = async(req, res, next) => {
    try {
        var data = await repository.get();
        res.status(200).send(data);
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
}

exports.post = async(req, res, next) => {
    let contract = new ValidationContract();
    contract.hasMinLen(req.body.name, 3, 'O nome deve conter pelo menos 3 caracteres');
    contract.isEmail(req.body.email, 'Email inválido');
    contract.hasMinLen(req.body.password, 6, 'A senha deve conter pelo menos 6 caracteres');

    // Se os dados forem inválidos
    if (!contract.isValid()) {
        res.status(400).send(contract.errors()).end();
        return;
    }

    try {
        await repository.create({
            name: req.body.name,
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY),
            roles: ["user"]
        });

        emailService.send(
            req.body.email,
            'Bem vindo ao Node Store',
            global.EMAIL_TMPL.replace('{0}', req.body.name)
        );

        res.status(201).send({
            message: 'Cliente cadastrado com sucesso!'
        });
```

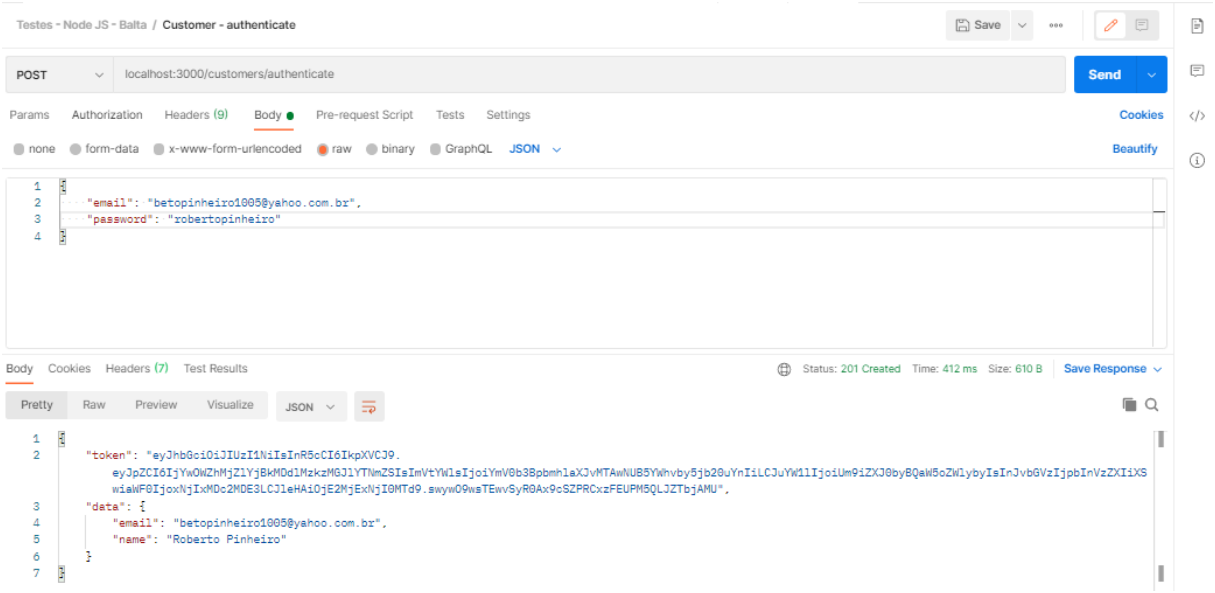
```
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

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

    try {
        const customer = await repository.authenticate({
            email: req.body.email,
            password: md5(req.body.password + global.SALT_KEY)
        });

        if(!customer){
            res.status(404).send({
                message: 'Usuário ou senha inválidos!'
            });
            return;
        }

        const token = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name,
            roles: customer.roles
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        console.log(e);
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};

exports.refreshToken = async(req, res, next) => {
    try {
        const token = req.body.token || req.query.token || req.headers['x-access-token'];
        const data = await authService.decodeToken(token);

        const customer = await repository.getById(data.id);

        if (!customer) {
            res.status(404).send({
                message: 'Cliente não encontrado'
            });
            return;
```

```
        }

        const tokenData = await authService.generateToken({
            id: customer._id,
            email: customer.email,
            name: customer.name,
            roles: customer.roles
        });

        res.status(201).send({
            token: token,
            data: {
                email: customer.email,
                name: customer.name
            }
        });
    } catch (e) {
        res.status(500).send({
            message: 'Falha ao processar sua requisição'
        });
    }
};
```

src\routes\product-route.js

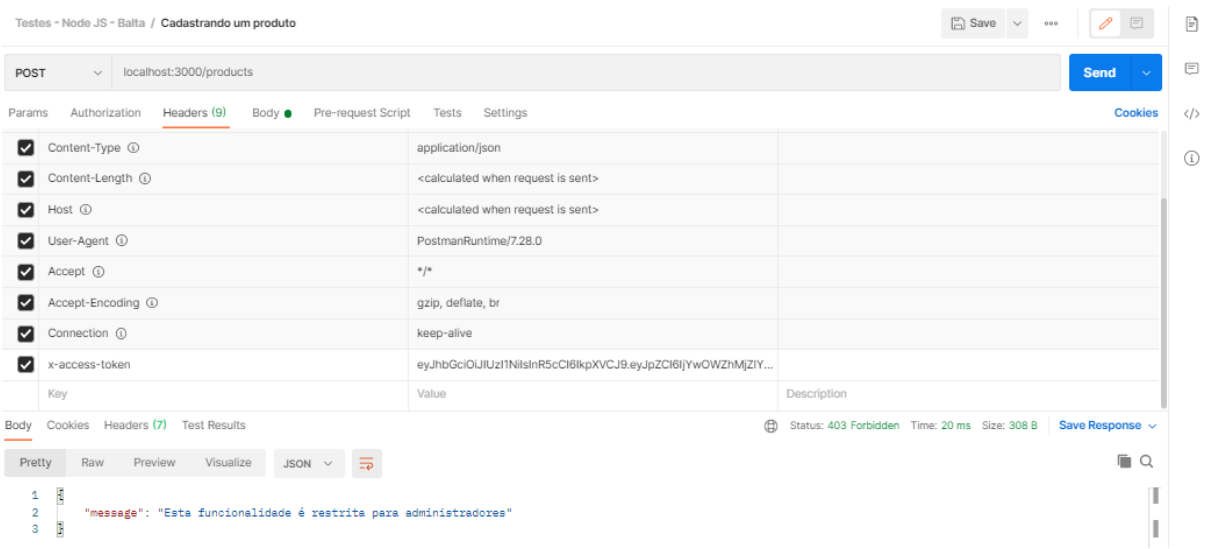
```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");
const authService = require('../services/auth-service');

router.get('/', controller.get);
router.get('/:slug', controller.getBySlug);
router.get('/admin/:id', controller.getById);
router.get('/tags/:tag', controller.getByTag);
router.post('/', authService.isAdmin, controller.post);
router.put('/:id', authService.isAdmin, controller.put);
router.delete('/:id', authService.isAdmin, controller.delete);

module.exports = router;
```

- No Postman, crie um novo authenticate, para gerar um novo token:

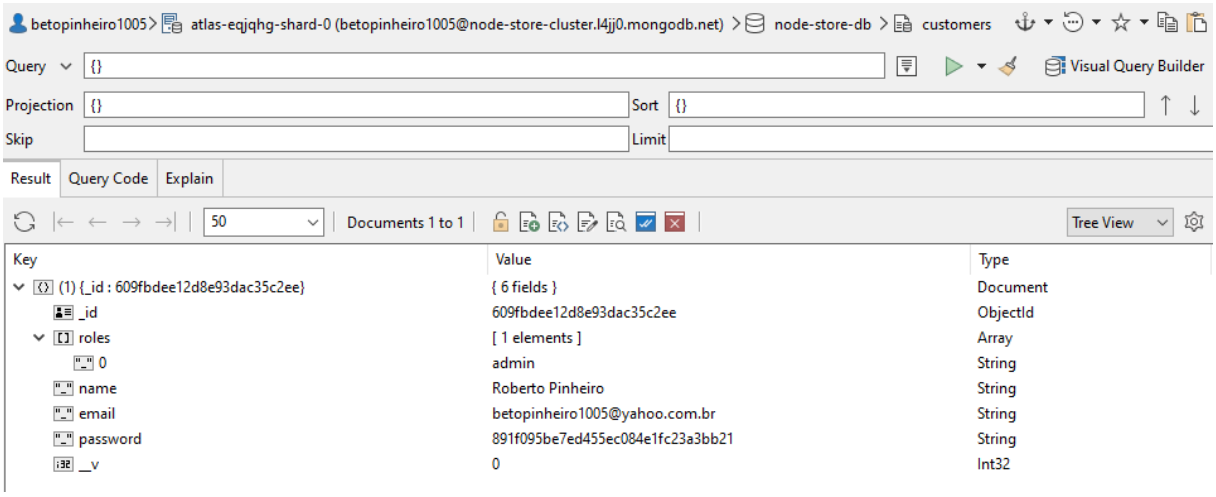



```
{
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYwOWZhMjZlYjBkMDdlMzkzMGJlYTNmZSIsImVtYWlsIjoiYmV0b3BpbmhlaXJvMTAwNUB5YWhvby5jb20uYnIiLCJuYW1lIjoiUm9iZXJ0byBQaW5oZWlybyIsInJvbGVzIjpbInVzZXIiXSwiaWF0IjoxNjIxMDc2MDE3LCJleHAiOjE2MjExNjI0MTd9.swywO9wsTEwvSyR0Ax9cSZPRCxzFEUPM5QLJZTbjAMU",
   "data": {
      "email": "betopinheiro1005@yahoo.com.br",
      "name": "Roberto Pinheiro"
   }
}
```

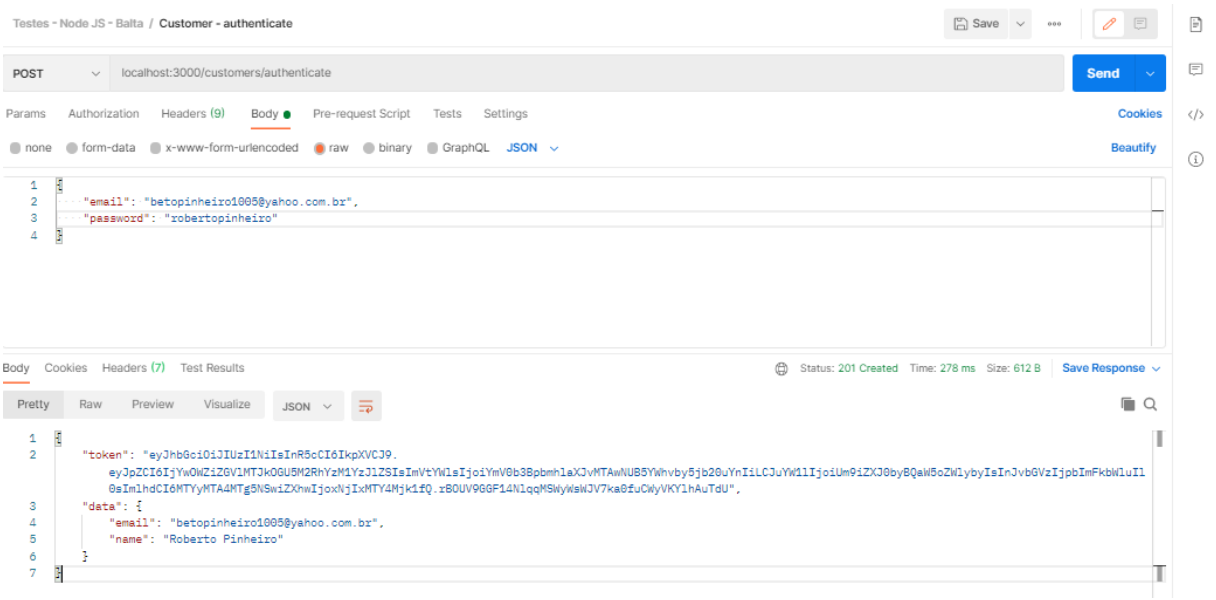

- Tente cadastrar um novo produto. Copie e cole o token acima na key x-access-token (aba Headers) e em seguida clique no botão Send.



- No 3T Studio edite o role do cliente (customer). Altere de user para admin:

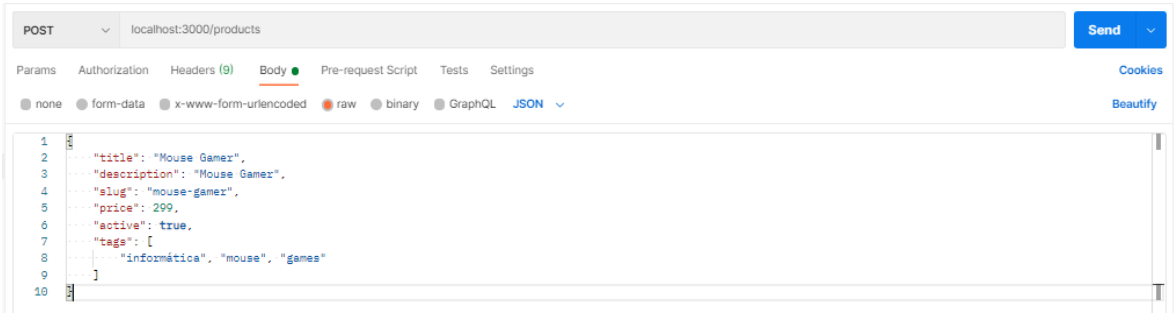


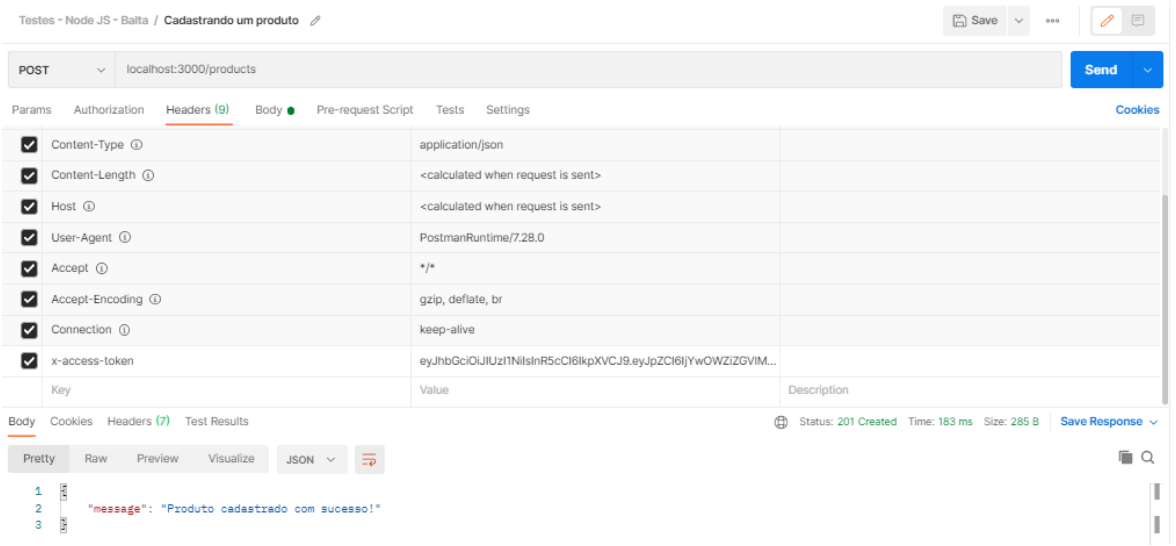
- Na autenticação atual o customer é apenas user. Portanto, agora é necessário fazer uma nova autenticação para tratá-lo como admin:

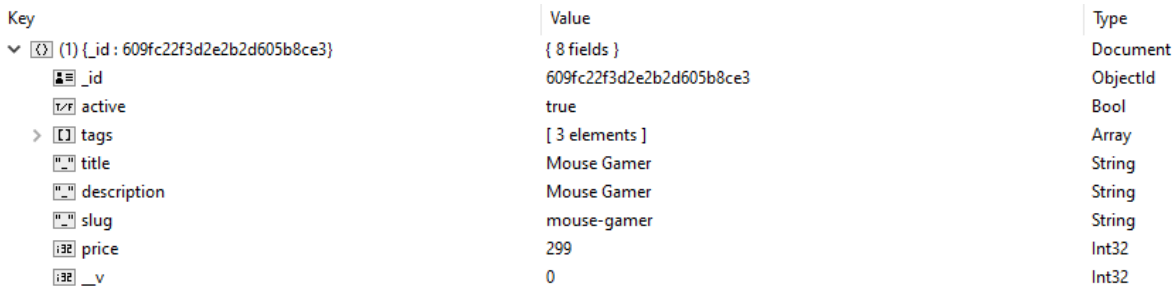



```
{
   "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYwOWZiZGVlMTJkOGU5M2RhYzM1YzJlZSIsImVtYWlsIjoiYmV0b3BpbmhlaXJvMTAwNUB5YWhvby5jb20uYnIiLCJuYW1lIjoiUm9iZXJ0byBQaW5oZWlybyIsInJvbGVzIjpbImFkbWluIl0sImlhdCI6MTYyMTA4MTg5NSwiZXhwIjoxNjIxMTY4Mjk1fQ.rBOUV9GGF14NlqqMSWyWsWJV7ka0fuCWyVKYlhAuTdU",
   "data": {
      "email": "betopinheiro1005@yahoo.com.br",
      "name": "Roberto Pinheiro"
   }
}
```

- Tente cadastrar um novo produto. Copie e cole o token acima na key x-access-token (aba Headers) e em seguida clique no botão Send.







Aula 38 - Outros

src\app.js

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const router = express.Router();

// Conecta ao banco
mongoose.connect("mongodb+srv://betopinheiro1005:angstron1005@node-store-
cluster.l4jj0.mongodb.net/node-store-db?retryWrites=true&w=majority");

// Carrega os models
const Product = require('./models/product');
const Customer = require('./models/customer');
const Order = require('./models/order');

// Carrega as rotas
const indexRoute = require('./routes/index-route');
const productRoute = require('./routes/product-route');
const customerRoute = require('./routes/customer-route');
const orderRoute = require('./routes/order-route');

app.use(bodyParser.json({
    limit: '5mb'
}));

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

// Habilita o CORS
app.use(function (req, res, next) {
    res.header('Access-Control-Allow-Origin', '*');
    res.header('Access-Control-Allow-Headers', 'Origin, X-Requested-With, Content-Type,
Accept, x-access-token');
    res.header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
    next();
});

app.use('/', indexRoute);
app.use('/products', productRoute);
app.use('/customers', customerRoute);
app.use('/orders', orderRoute);

module.exports = app;
```