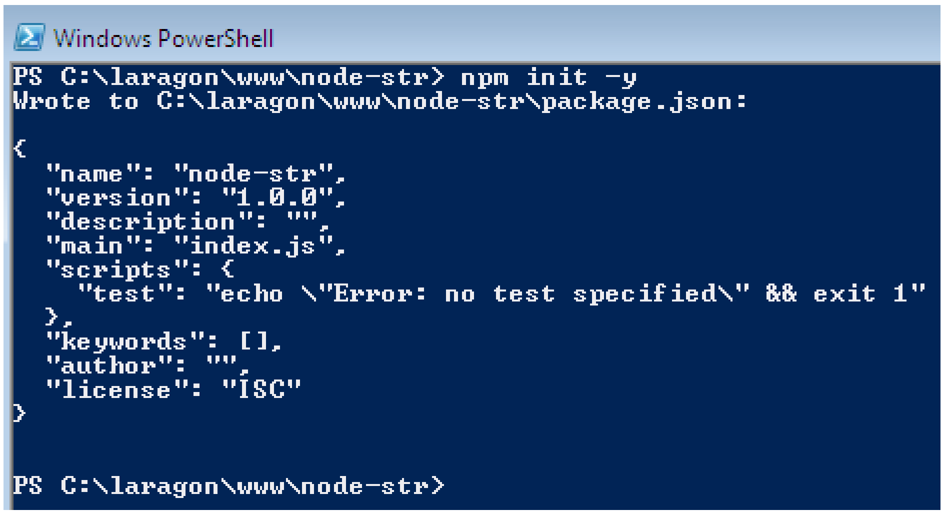
Criando API's com Node.JS - Aulas 1 a 33
balta.io (André Baltieri)

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

Resumo do curso feito por Roberto Pinheiro

Aula 02 - npm init e instalação dos pacotes



Windows PowerShell

```
PS C:\laragon\www\node-str> npm init -y
Wrote to C:\laragon\www\node-str\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"
}


PS C:\laragon\www\node-str>
```

package.json

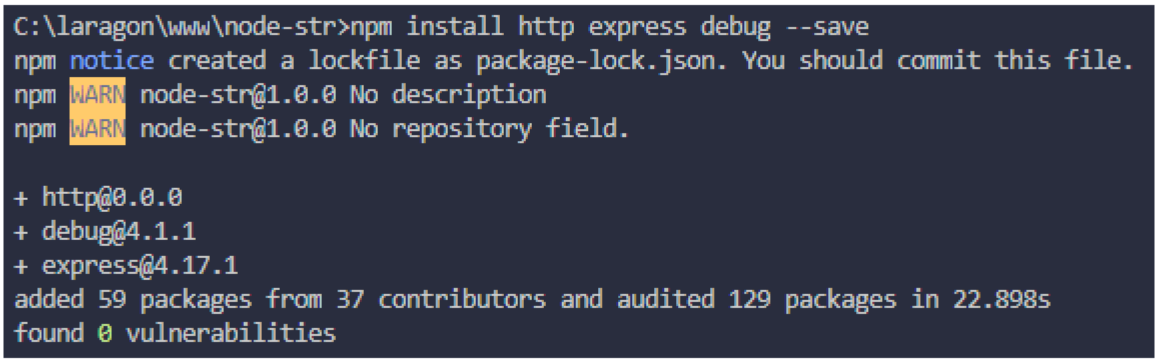
```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Instalando pacotes básicos

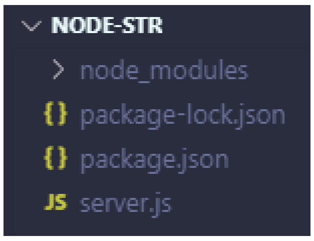
O comando a seguir irá instalar os pacotes:

- http
- express
- debug

npm install http express debug --save



```
C:\laragon\www\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.

+ http@0.0.0
+ debug@4.1.1
+ express@4.17.1
added 59 packages from 37 contributors and audited 129 packages in 22.898s
found 0 vulnerabilities
```



```
NODE-STR
  > node_modules
  {} package-lock.json
  {} package.json
  JS server.js
```

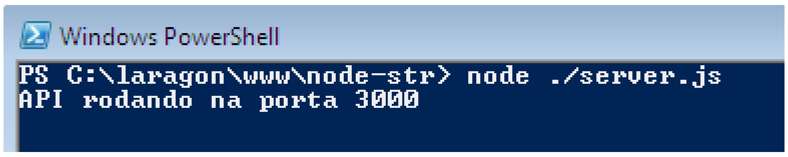
package.json

```
{
  "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": "^4.1.1",
    "express": "^4.17.1",
    "http": "0.0.0"
  }
}
```

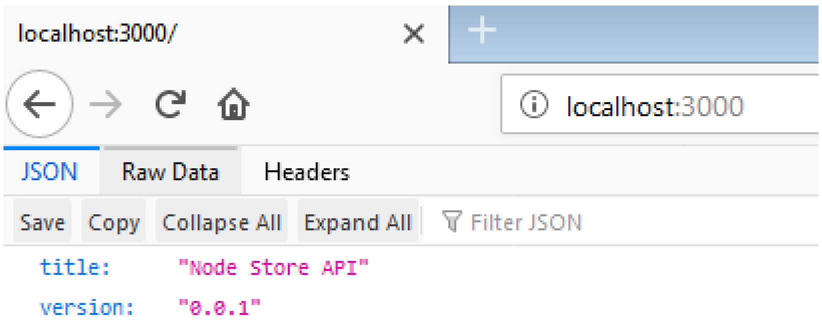
Aula 03 - Criando um servidor web

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);
```

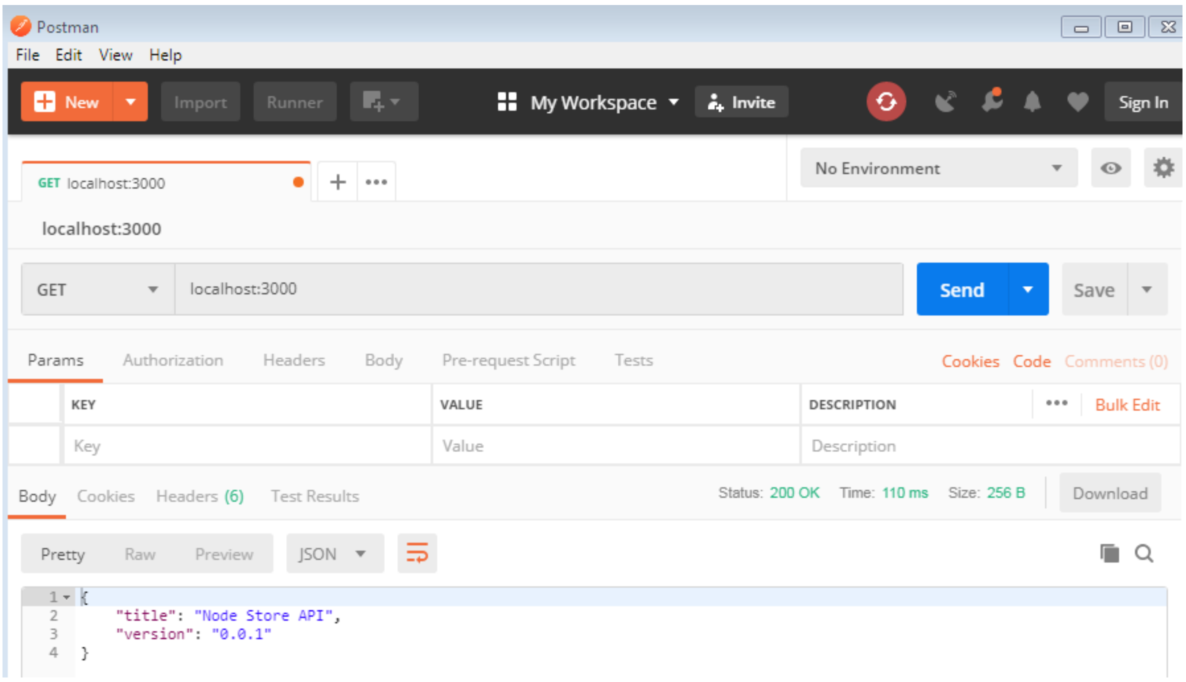


```
PS C:\laragon\www\node-str> node ./server.js
API rodando na porta 3000
```



```
title:    "Node Store API"
version:  "0.0.1"
```

Usando o Postman:



GET localhost:3000

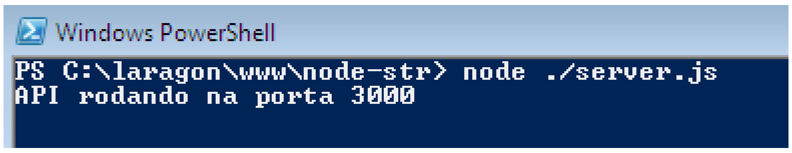
Status: 200 OK Time: 110 ms Size: 256 B

```
{
    "title": "Node Store API",
    "version": "0.0.1"
}
```

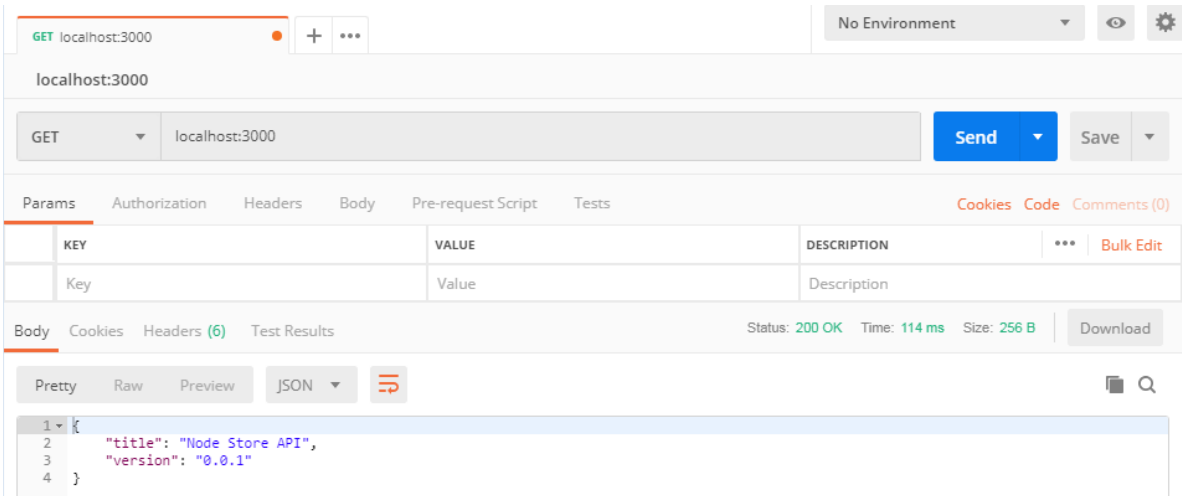
Aula 04 - Normalizando a porta

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;
}
```



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



Status: 200 OK Time: 114 ms Size: 256 B

```
{
    "title": "Node Store API",
    "version": "0.0.1"
}
```

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;
  }
}
```

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);
  }
```

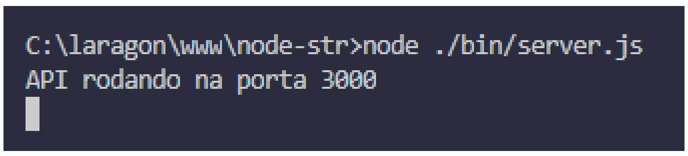
Aula 07 - Separando o servidor

bin/server.js

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

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

const server = http.createServer(app);

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);
}
```

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;
```



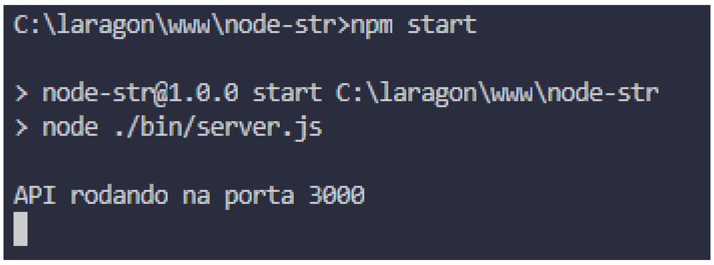
```
C:\laragon\www\node-str>node ./bin/server.js
API rodando na porta 3000
```

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

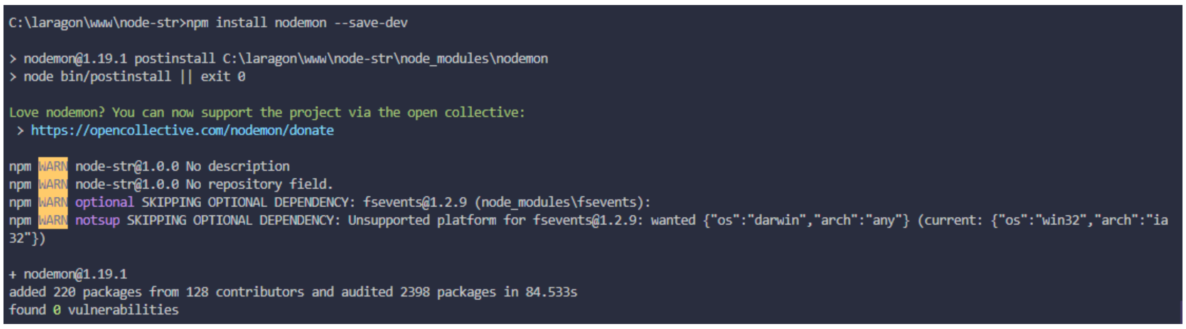


```
C:\laragon\www\node-str>npm start

> node-str@1.0.0 start C:\laragon\www\node-str
> node ./bin/server.js

API rodando na porta 3000
```

Aula 09 - Nodemon

Instalando o pacote nodemon

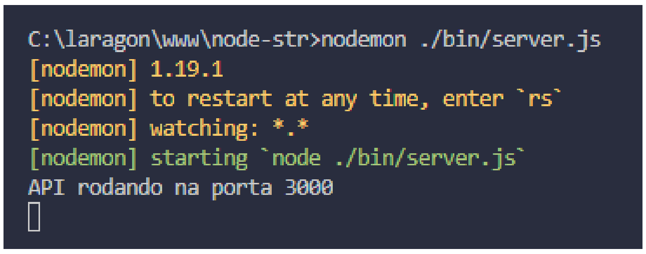
npm install nodemon --save-dev



```
C:\laragon\www\node-str>npm install nodemon --save-dev

> nodemon@1.19.1 postinstall C:\laragon\www\node-str\node_modules\nodemon
> node bin/postinstall || exit 0

Love nodemon? You can now support the project via the open collective:
 > https://opencollective.com/nodemon/donate

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@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"ia
32"})

+ nodemon@1.19.1
added 220 packages from 128 contributors and audited 2398 packages in 84.533s
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": {
    "debug": "^4.1.1",
    "express": "^4.17.1",
    "http": "0.0.0"
  },
  "devDependencies": {
    "nodemon": "^1.19.1"
  }
}
```

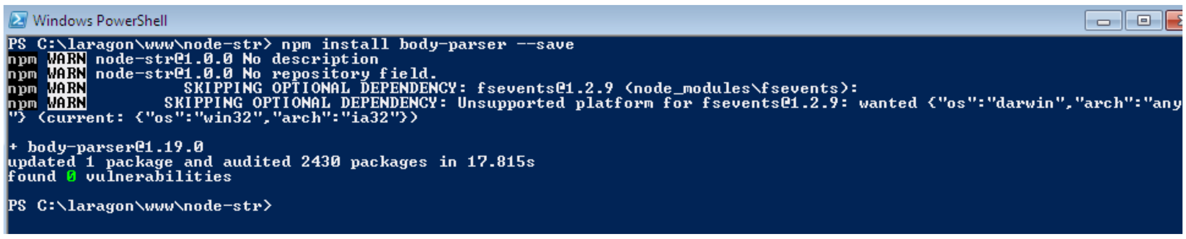


```
C:\laragon\www\node-str>nodemon ./bin/server.js
[nodemon] 1.19.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
```

Aula 10 - CRUD Rest

Instalando o pacote body-parser

```
npm install body-parser --save
```



Windows PowerShell

```
PS C:\laragon\www\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                 SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN               SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any
"} (current: {"os":"win32","arch":"ia32"})

+ body-parser@1.19.0
updated 1 package and audited 2430 packages in 17.815s
found 0 vulnerabilities

PS C:\laragon\www\node-str>
```

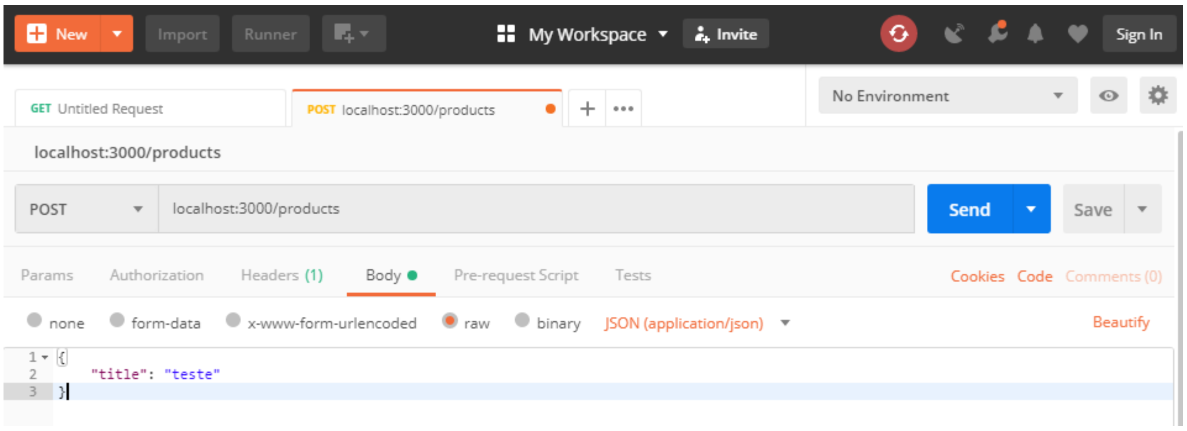
src/app.js

```
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);
});

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

module.exports = app;
```

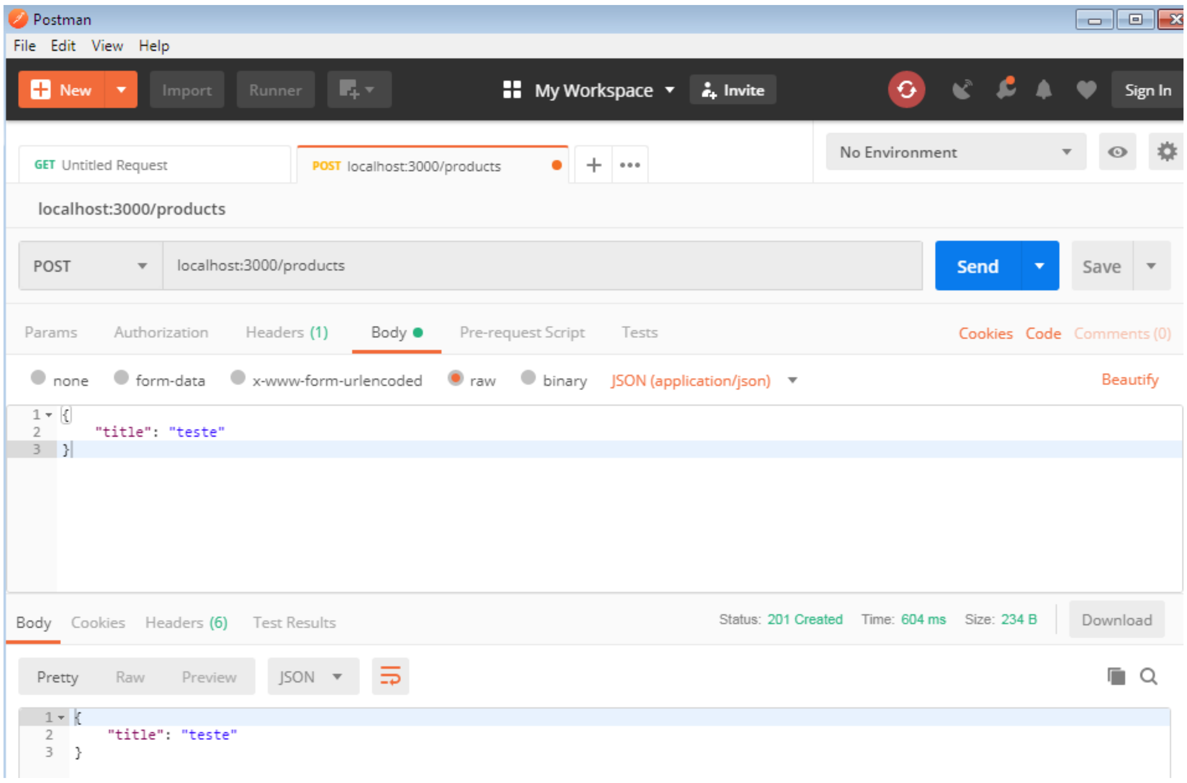
src/app.js

```
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 GET



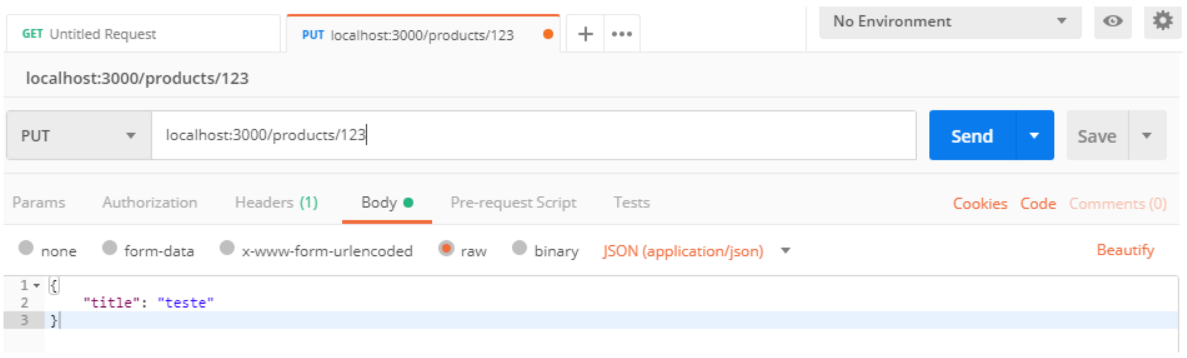
```
{
    "title": "teste"
}
```



```
{
    "title": "teste"
}
```

```
{
    "title": "teste"
}
```

put → método PUT

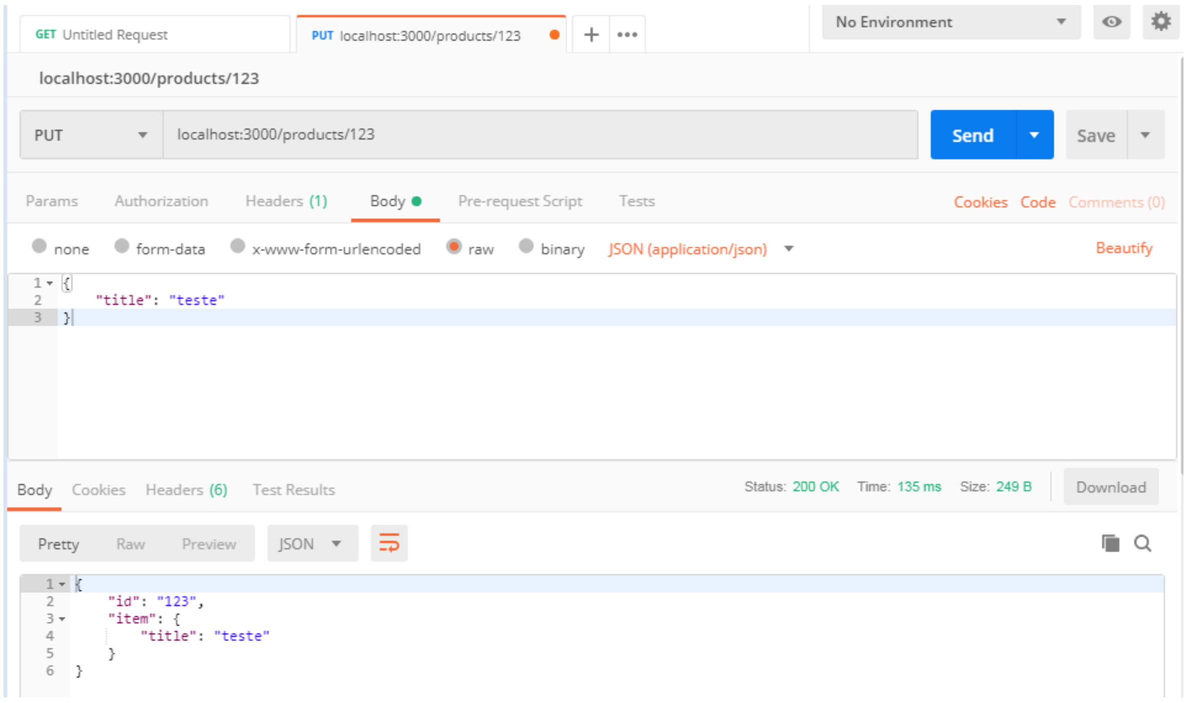


localhost:3000/products/123

PUT localhost:3000/products/123

JSON (application/json)

```
{
    "title": "teste"
}
```



localhost:3000/products/123

PUT localhost:3000/products/123

JSON (application/json)

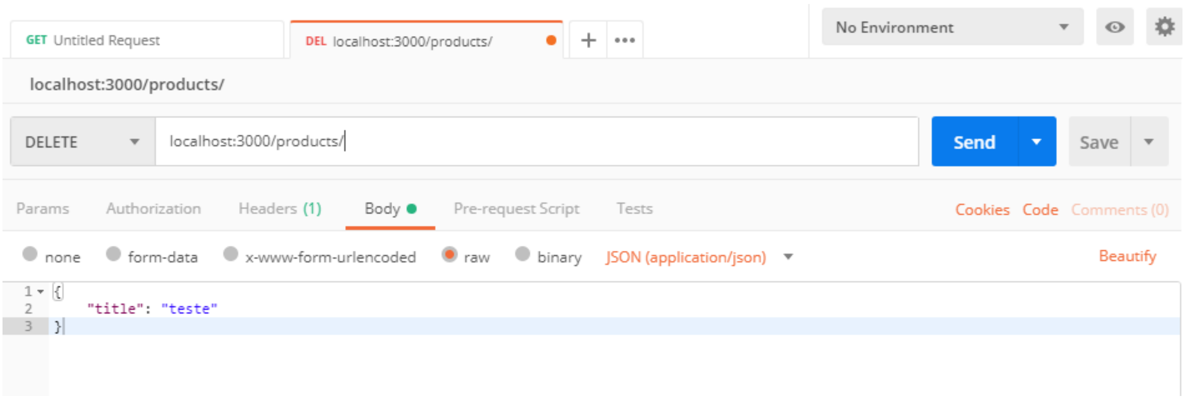
```
{
    "title": "teste"
}
```

Status: 200 OK Time: 135 ms Size: 249 B

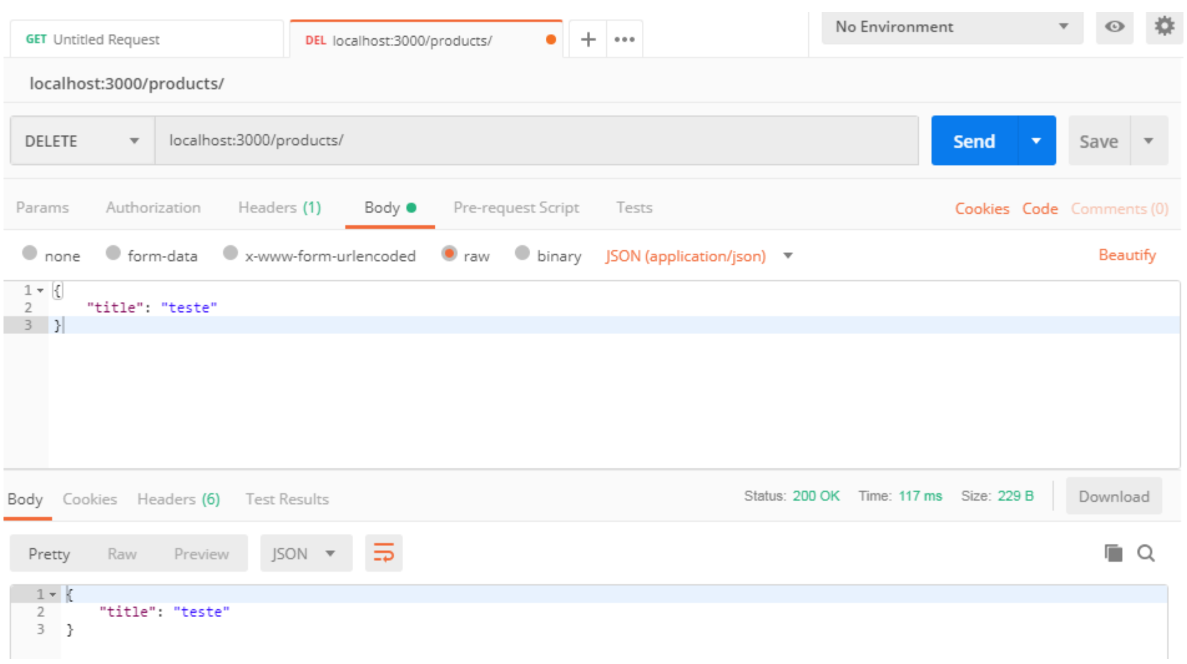
JSON

```
{
    "id": "123",
    "item": {
        "title": "teste"
    }
}
```

delete → método DELETE


GET Untitled Request
DEL localhost:3000/products/
No Environment
localhost:3000/products/
DELETE
localhost:3000/products/
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary JSON (application/json)
Beautify

```
{
    "title": "teste"
}
```


GET Untitled Request
DEL localhost:3000/products/
No Environment
localhost:3000/products/
DELETE
localhost:3000/products/
Send
Save
Params Authorization Headers (1) Body Pre-request Script Tests
Cookies Code Comments (0)
none form-data x-www-form-urlencoded raw binary JSON (application/json)
Beautify

```
{
    "title": "teste"
}
```

Body Cookies Headers (6) Test Results
Status: 200 OK Time: 117 ms Size: 229 B
Download
Pretty Raw Preview JSON

```
{
    "title": "teste"
}
```

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: "0.0.1"
   });
});

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

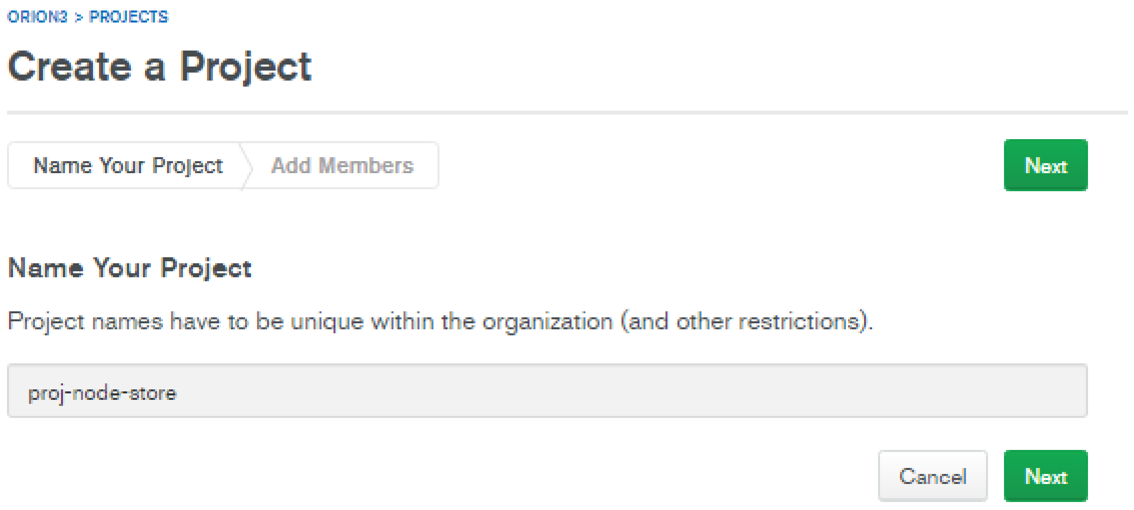
```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/product-controller");

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

module.exports = router;
```

Aula 13 - MongoDB Setup

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

- Crie uma organização com nome: Orion3

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



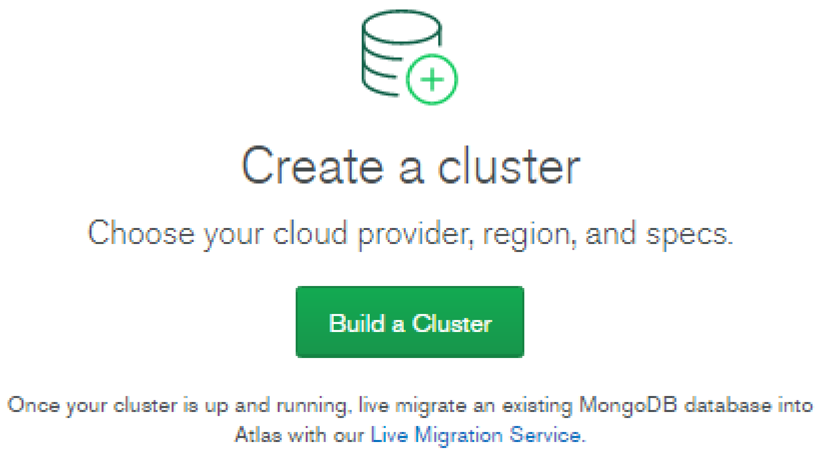
ORION3 > PROJECTS

Create a Project

Name Your Project

Project names have to be unique within the organization (and other restrictions).

proj-node-store



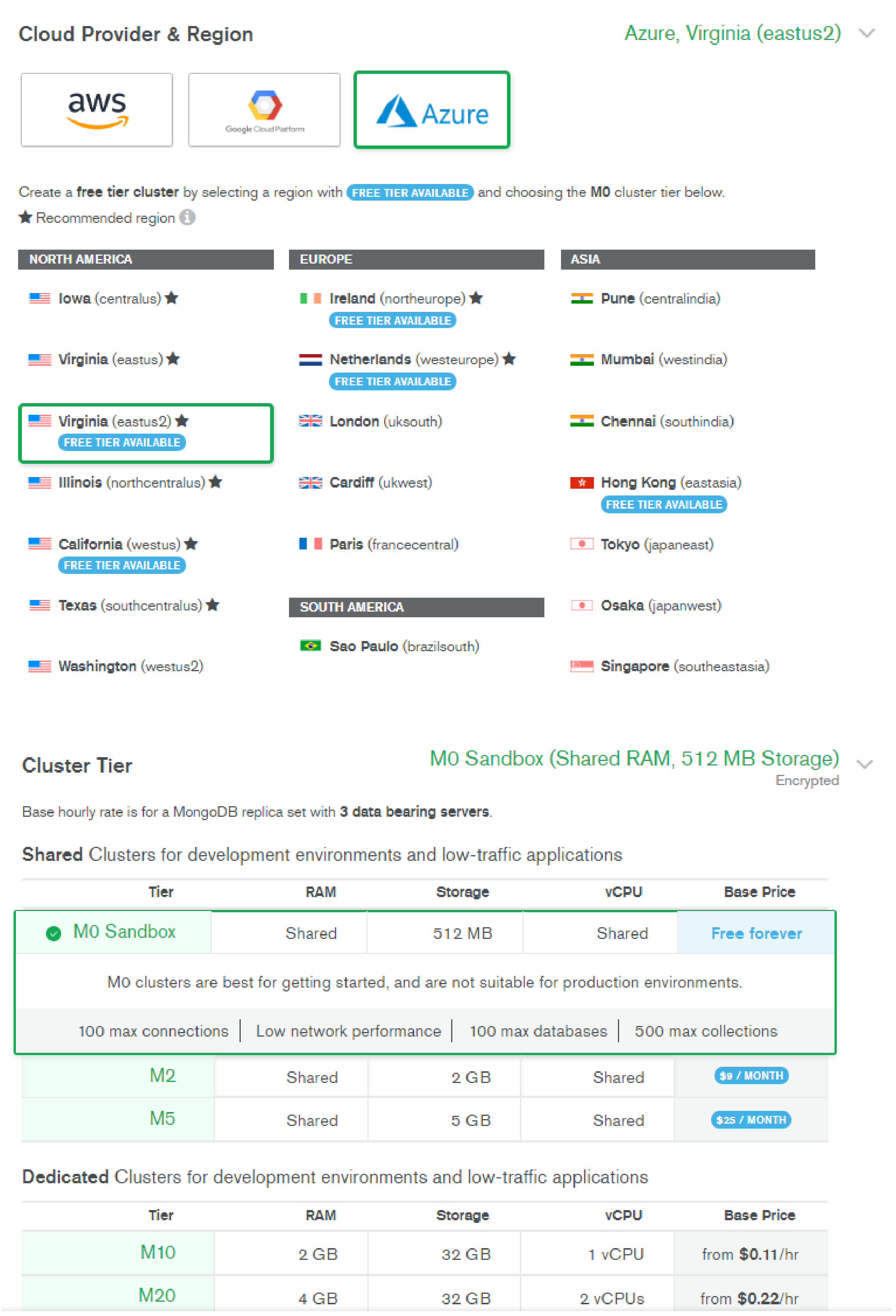
Create a cluster

Choose your cloud provider, region, and specs.

Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our Live Migration Service.

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

Configuração



Cloud Provider & Region — Azure, Virginia (eastus2)

Create a **free tier cluster** by selecting a region with FREE TIER AVAILABLE and choosing the **M0** cluster tier below.

★ Recommended region

NORTH AMERICA	EUROPE	ASIA
Iowa (centralus) ★	**Ireland** (northeurope) ★ FREE TIER AVAILABLE	**Pune** (centralindia)
Virginia (eastus) ★	**Netherlands** (westeurope) ★ FREE TIER AVAILABLE	**Mumbai** (westindia)
Virginia (eastus2) ★ FREE TIER AVAILABLE	**London** (uksouth)	**Chennai** (southindia)
Illinois (northcentralus) ★	**Cardiff** (ukwest)	**Hong Kong** (eastasia) FREE TIER AVAILABLE
California (westus) ★ FREE TIER AVAILABLE	**Paris** (francecentral)	**Tokyo** (japaneast)
Texas (southcentralus) ★	SOUTH AMERICA	**Osaka** (japanwest)
	Sao Paulo (brazilsouth)	
Washington (westus2)		**Singapore** (southeastasia)

Cluster Tier — M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted

Base hourly rate is for a MongoDB replica set with **3 data bearing servers**.

Shared Clusters for development environments and low-traffic applications

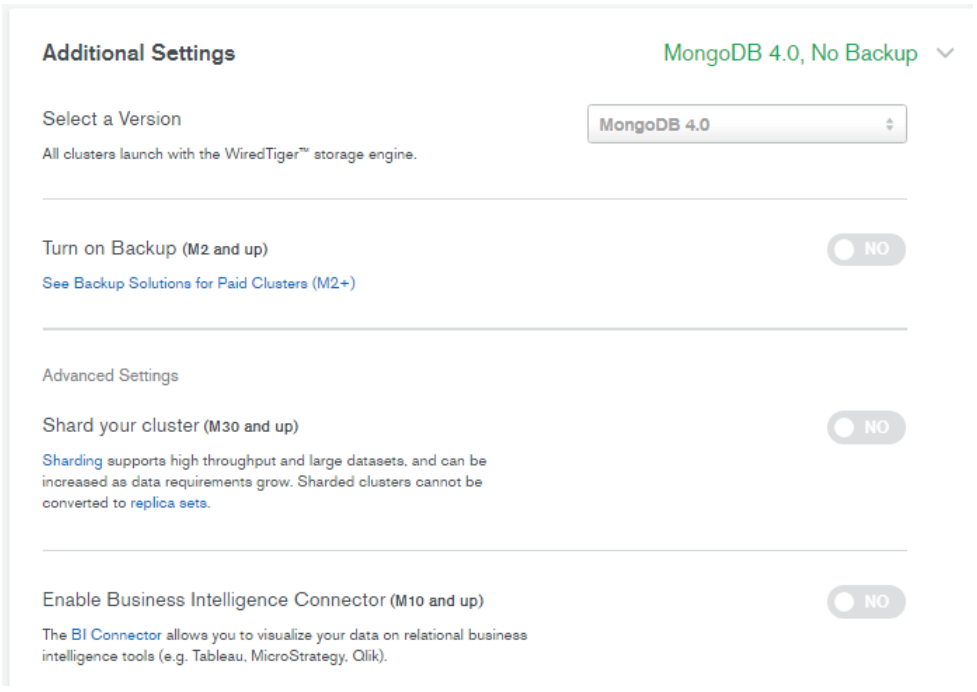
Tier	RAM	Storage	vCPU	Base Price
✓ M0 Sandbox	Shared	512 MB	Shared	Free forever
M2	Shared	2 GB	Shared	$9 / MONTH
M5	Shared	5 GB	Shared	$25 / MONTH

M0 clusters are best for getting started, and are not suitable for production environments.

100 max connections | Low network performance | 100 max databases | 500 max collections

Dedicated Clusters for development environments and low-traffic applications

Tier	RAM	Storage	vCPU	Base Price
M10	2 GB	32 GB	1 vCPU	from **$0.11**/hr
M20	4 GB	32 GB	2 vCPUs	from **$0.22**/hr



Additional Settings — MongoDB 4.0, No Backup

Select a Version: MongoDB 4.0

All clusters launch with the WiredTiger™ storage engine.

Turn on Backup (M2 and up): NO

See Backup Solutions for Paid Clusters (M2+)

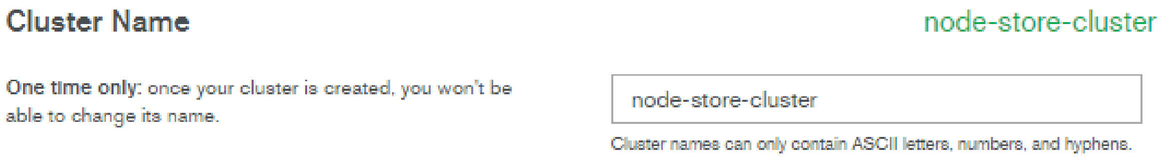
Advanced Settings

Shard your cluster (M30 and up): NO

Sharding supports high throughput and large datasets, and can be increased as data requirements grow. Sharded clusters cannot be converted to replica sets.

Enable Business Intelligence Connector (M10 and up): NO

The BI Connector allows you to visualize your data on relational business intelligence tools (e.g. Tableau, MicroStrategy, Qlik).



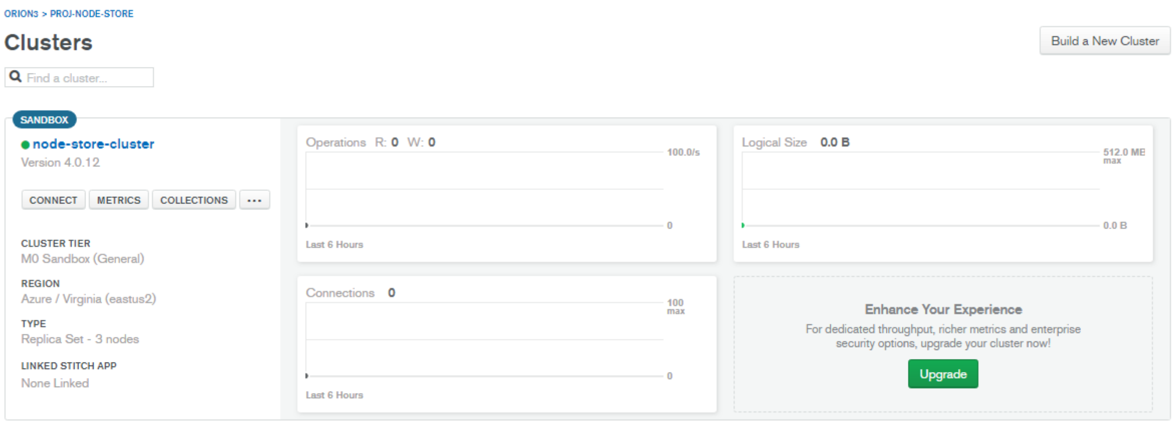
Cluster Name — node-store-cluster

One time only: once your cluster is created, you won't be able to change its name.

node-store-cluster

Cluster names can only contain ASCII letters, numbers, and hyphens.

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



ORION3 > PROJ-NODE-STORE

Clusters

Build a New Cluster

Find a cluster...

SANDBOX

● node-store-cluster

Version 4.0.12

CONNECT | METRICS | COLLECTIONS | ...

CLUSTER TIER: M0 Sandbox (General)

REGION: Azure / Virginia (eastus2)

TYPE: Replica Set - 3 nodes

LINKED STITCH APP: None Linked

Operations R: 0 W: 0 — 100.0/s — 0 — Last 6 Hours

Logical Size 0.0 B — 512.0 MB max — 0.0 B — Last 6 Hours

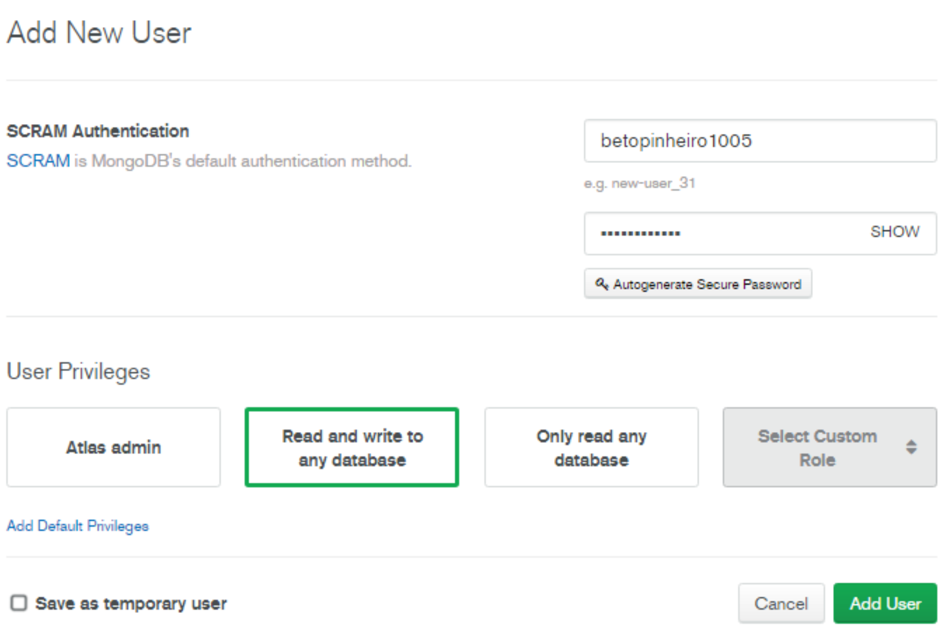
Connections 0 — 100 max — 0 — Last 6 Hours

Enhance Your Experience

For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!

Upgrade

- Crie um usuário para acessar o banco de dados.



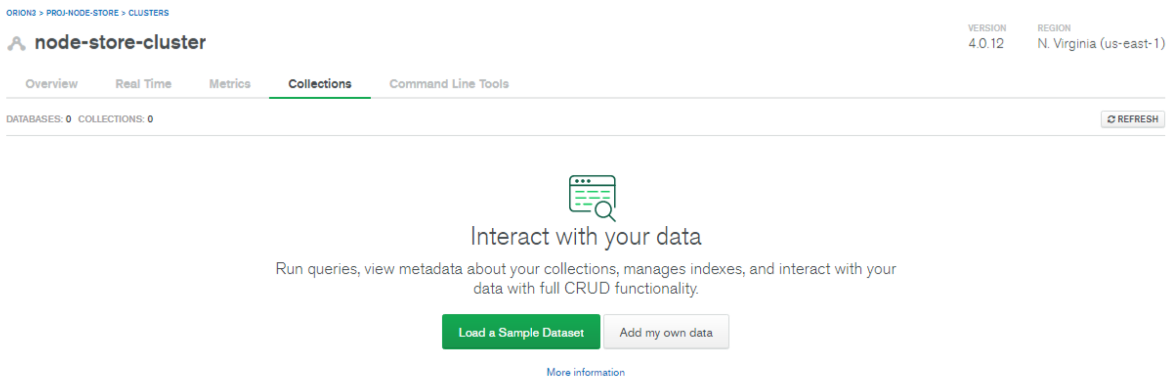
Add New User

SCRAM Authentication
SCRAM is MongoDB's default authentication method.

betopinheiro1005
e.g. new-user_31
••••••••••••• SHOW
Autogenerate Secure Password

User Privileges

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



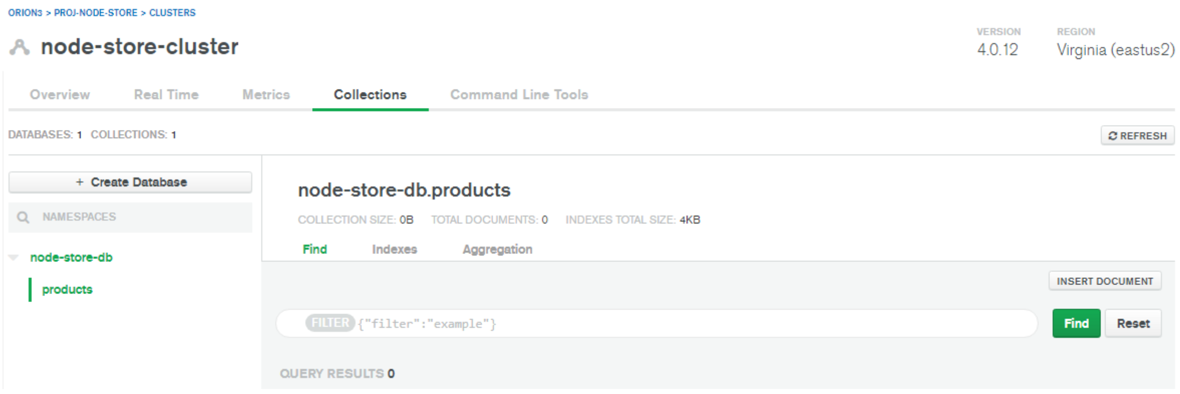
ORION3 > PROJ-NODE-STORE > CLUSTERS
node-store-cluster
VERSION 4.0.12 REGION N. Virginia (us-east-1)

DATABASES: 0 COLLECTIONS: 0

Interact with your data

Run queries, view metadata about your collections, manages indexes, and interact with your data with full CRUD functionality.

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



ORION3 > PROJ-NODE-STORE > CLUSTERS
node-store-cluster
VERSION 4.0.12 REGION Virginia (eastus2)

DATABASES: 1 COLLECTIONS: 1

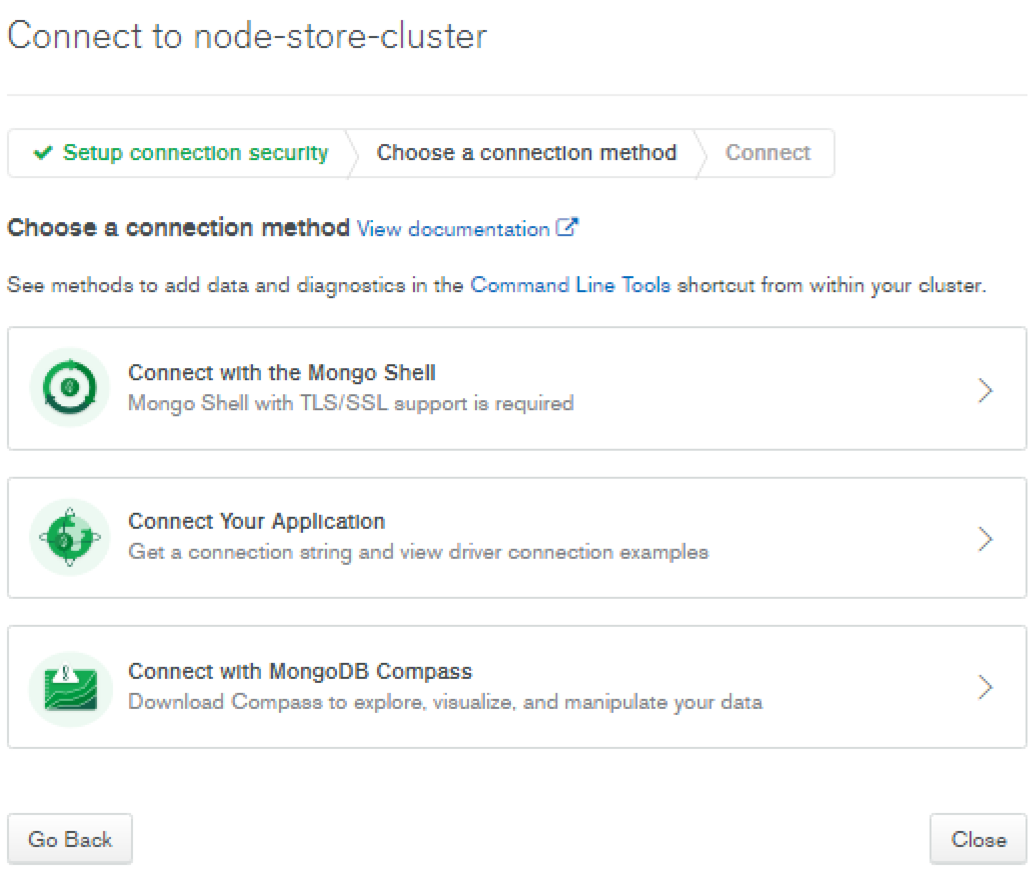
node-store-db
products

node-store-db.products
COLLECTION SIZE: 0B TOTAL DOCUMENTS: 0 INDEXES TOTAL SIZE: 4KB

```
{"filter":"example"}
```

QUERY RESULTS 0

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


Connect to node-store-cluster

✓ Setup connection security | Choose a connection method | Connect

Choose a connection method View documentation

See methods to add data and diagnostics in the Command Line Tools shortcut from within your cluster.

Connect with the Mongo Shell
Mongo Shell with TLS/SSL support is required

Connect Your Application
Get a connection string and view driver connection examples

Connect with MongoDB Compass
Download Compass to explore, visualize, and manipulate your data

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

- Copie a string de conexão:

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

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

- Baixe e instale o programa Studio 3T.

- Opção de outra interface gráfica:

RoboMongo

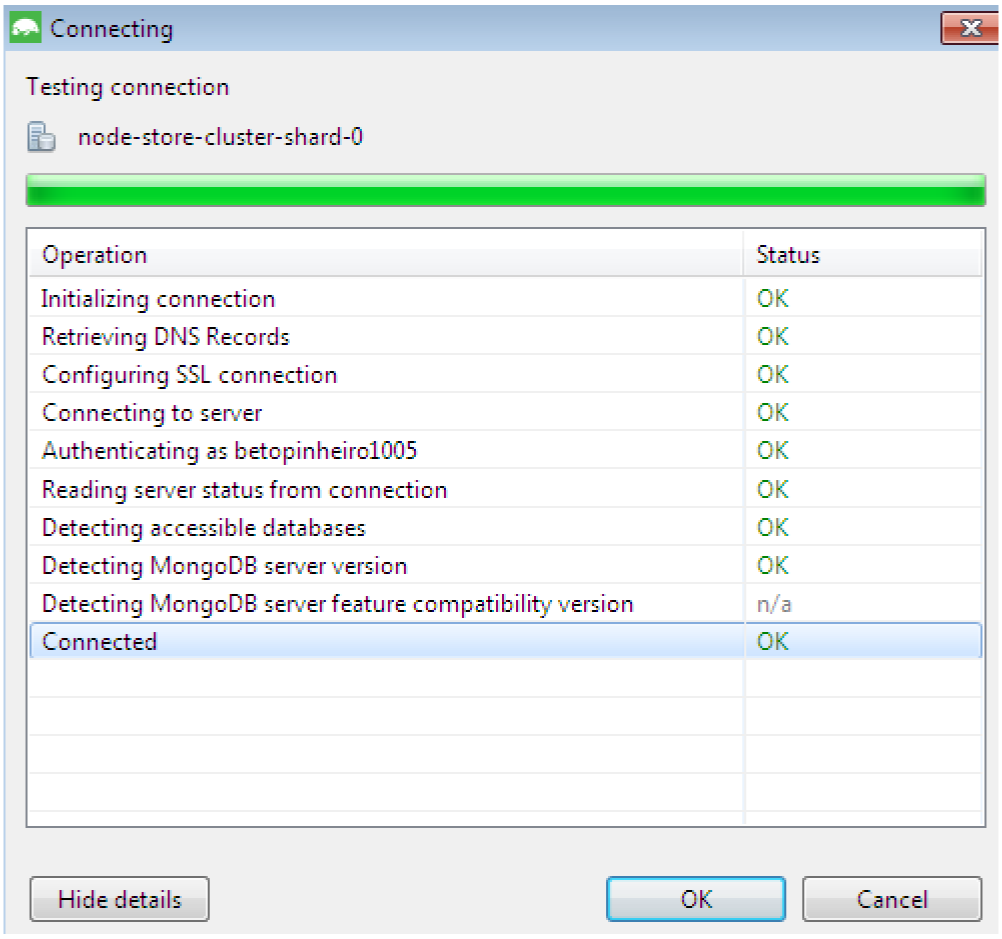
- Clique no botão Connect.

- Clique em New Connection.

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

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

- Faça o teste de conexão:


Connecting

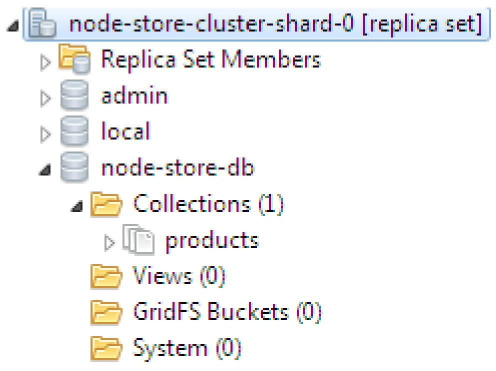
Testing connection

node-store-cluster-shard-0

Operation	Status
Initializing connection	OK
Retrieving DNS Records	OK
Configuring SSL connection	OK
Connecting to server	OK
Authenticating as betopinheiro1005	OK
Reading server status from connection	OK
Detecting accessible databases	OK
Detecting MongoDB server version	OK
Detecting MongoDB server feature compatibility version	n/a
Connected	OK

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

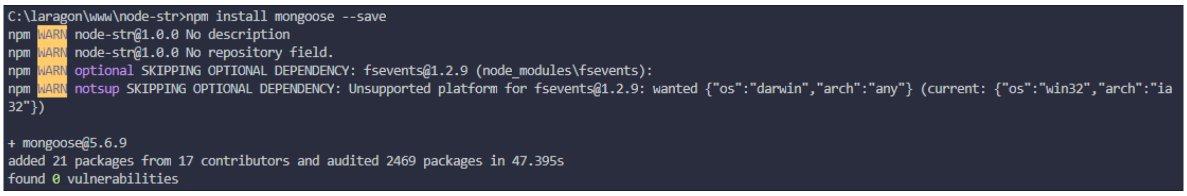
- Clique no botão "Connect"


```
node-store-cluster-shard-0 [replica set]
  Replica Set Members
  admin
  local
  node-store-db
    Collections (1)
      products
    Views (0)
    GridFS Buckets (0)
    System (0)
```

Aula 14 - Mongoose

Instalação do Mongoose

```
npm install mongoose --save
```



```
C:\laragon\www\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@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"ia
32"})

+ mongoose@5.6.9
added 21 packages from 17 contributors and audited 2469 packages in 47.395s
found 0 vulnerabilities
```

package.json

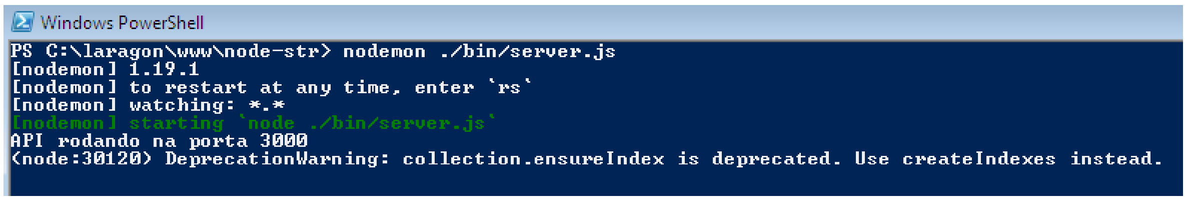
```
{
  "name": "node-str",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "node ./bin/server.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "debug": "^4.1.1",
    "express": "^4.17.1",
    "http": "0.0.0",
    "mongoose": "^5.6.9"
  },
  "devDependencies": {
    "nodemon": "^1.19.1"
  }
}
```

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:<password>@node-store-cluster-nlcnv.azure.mongodb.net/node-store-db?retryWrites=true&w=majority", { useNewUrlParser: true });

// 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;
```

Atenção: Na string de conexão, substituia <password> pela senha cadastrada na mlab para esse banco de dados.

Se estiver tudo ok, o servidor irá rodar normalmente:



```
PS C:\laragon\www\node-str> nodemon ./bin/server.js
[nodemon] 1.19.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node ./bin/server.js`
API rodando na porta 3000
(node:30120) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

Aula 15 - Models

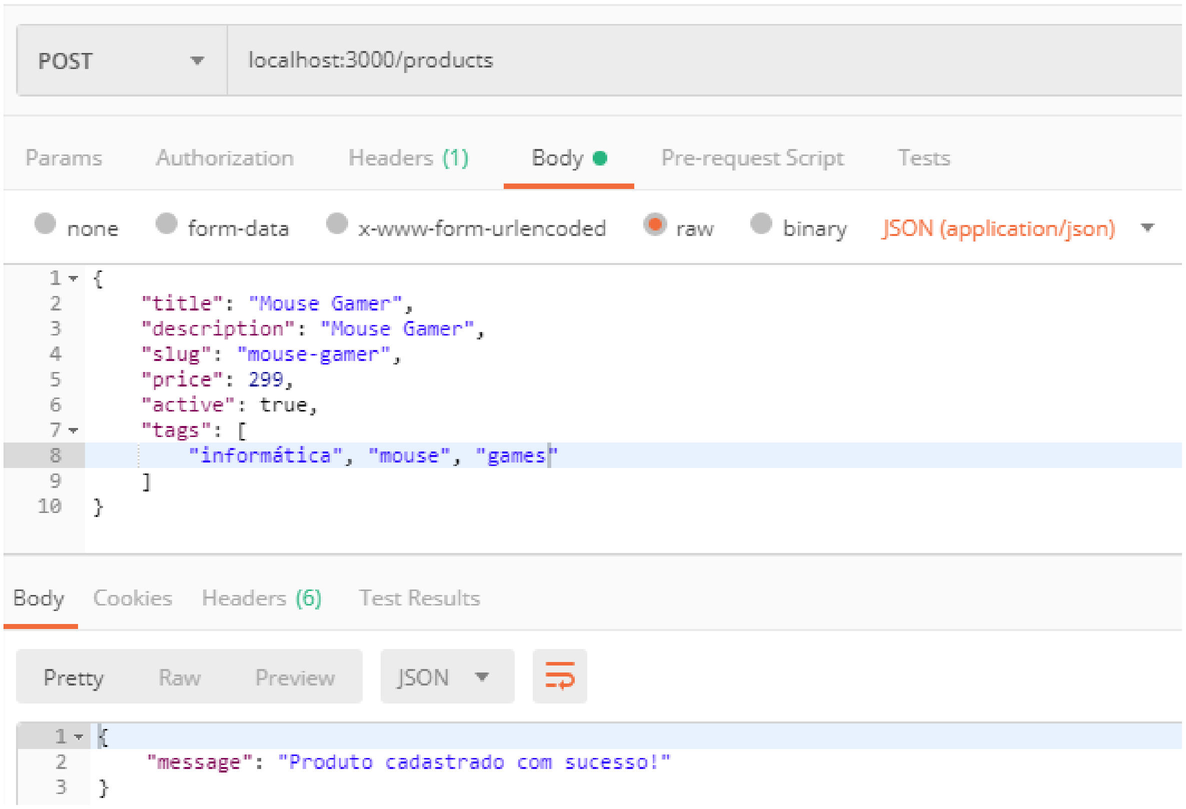
src/models/product.js

```
'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/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",    {    useNewUrlParser: true });

// 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;
```

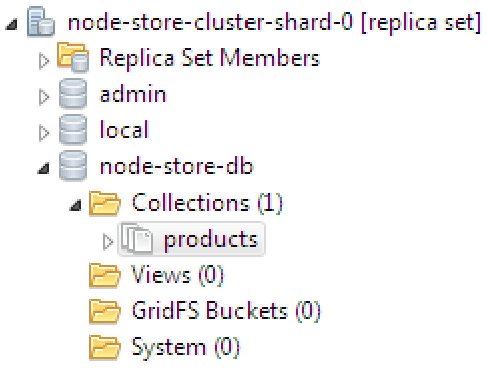


localhost:3000/products

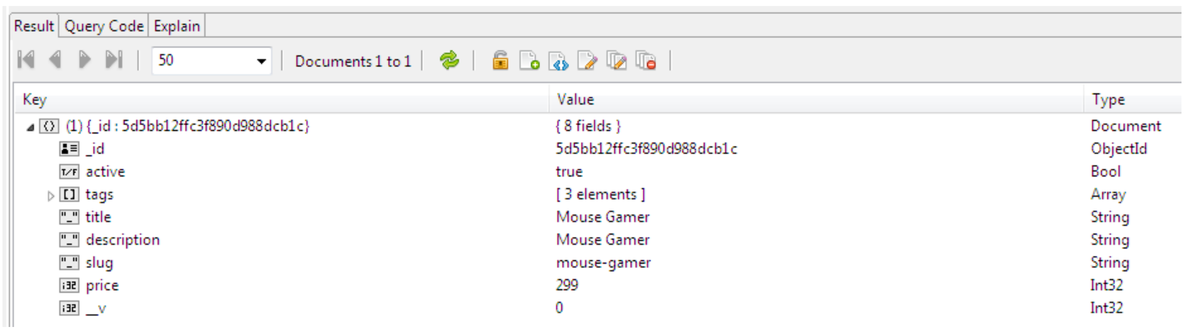
```
{
    "title": "Mouse Gamer",
    "description": "Mouse Gamer",
    "slug": "mouse-gamer",
    "price": 299,
    "active": true,
    "tags": [
        "informática", "mouse", "games"
    ]
}
```

```
{
    "message": "Produto cadastrado com sucesso!"
}
```

No Studio 3T:

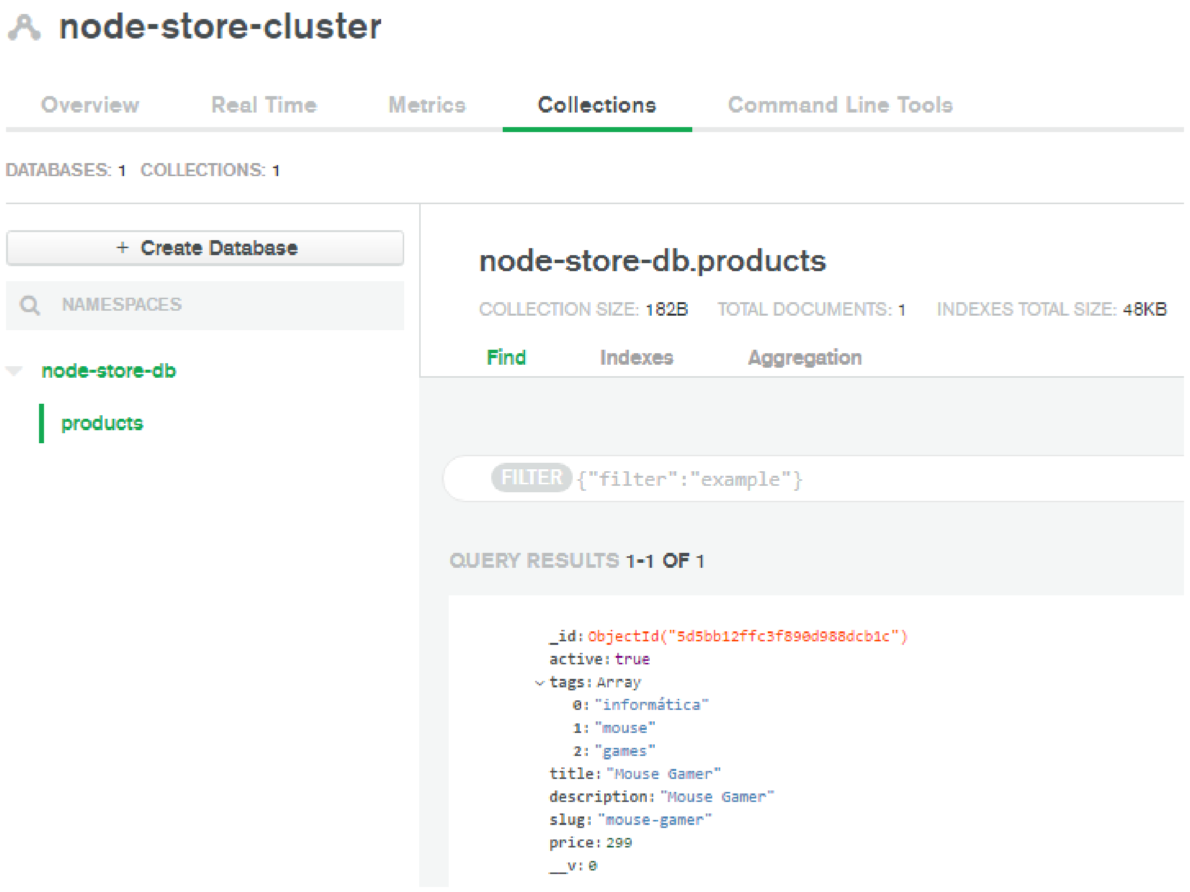


```
node-store-cluster-shard-0 [replica set]
    Replica Set Members
    admin
    local
    node-store-db
        Collections (1)
            products
        Views (0)
        GridFS Buckets (0)
        System (0)
```



Key	Value	Type
(1) {_id : 5d5bb12ffc3f890d988dcb1c}	{ 8 fields }	Document
_id	5d5bb12ffc3f890d988dcb1c	ObjectId
active	true	Bool
tags	[3 elements]	Array
title	Mouse Gamer	String
description	Mouse Gamer	String
slug	mouse-gamer	String
price	299	Int32
__v	0	Int32

No mlab:



```
_id: ObjectId("5d5bb12ffc3f890d988dcb1c")
active: true
tags: Array
    0: "informática"
    1: "mouse"
    2: "games"
title: "Mouse Gamer"
description: "Mouse Gamer"
slug: "mouse-gamer"
price: 299
__v: 0
```

Aula 17 - Listando os produtos

src/controllers/product-controller.js

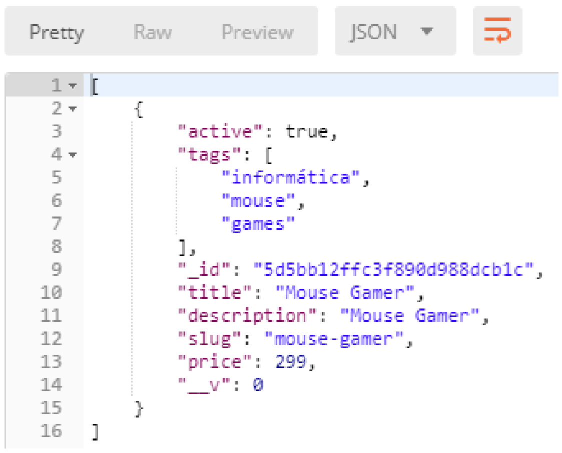
```
'use strict';

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;
```

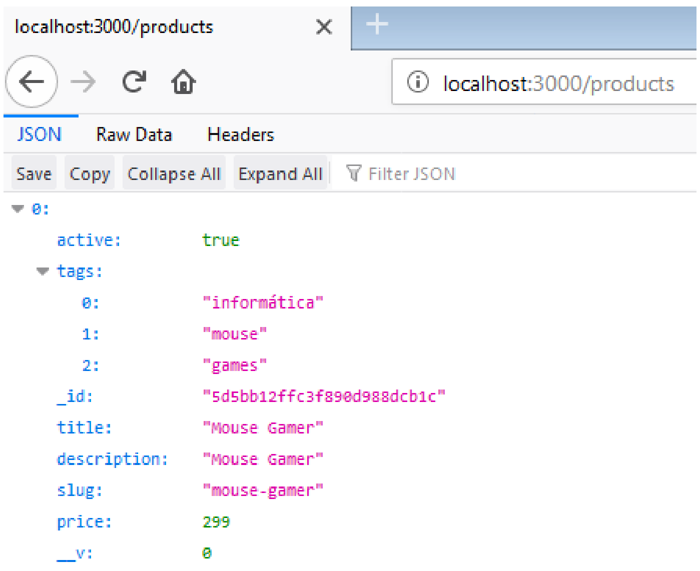
Testando no Postman

GET - localhost:3000/products



```
[
    {
        "active": true,
        "tags": [
            "informática",
            "mouse",
            "games"
        ],
        "_id": "5d5bb12ffc3f890d988dcb1c",
        "title": "Mouse Gamer",
        "description": "Mouse Gamer",
        "slug": "mouse-gamer",
        "price": 299,
        "__v": 0
    }
]
```

No navegador

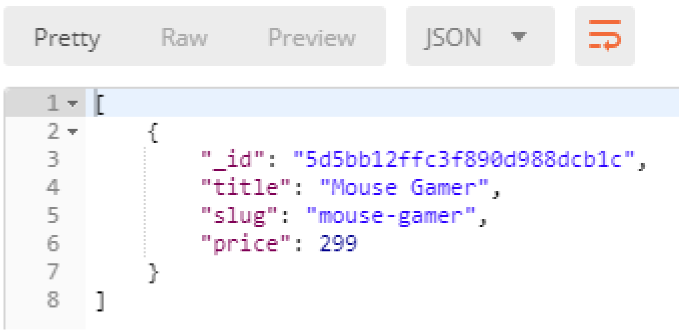


```
0:
    active:         true
    tags:
        0:          "informática"
        1:          "mouse"
        2:          "games"
    _id:            "5d5bb12ffc3f890d988dcb1c"
    title:          "Mouse Gamer"
    description:    "Mouse Gamer"
    slug:           "mouse-gamer"
    price:          299
    __v:            0
```

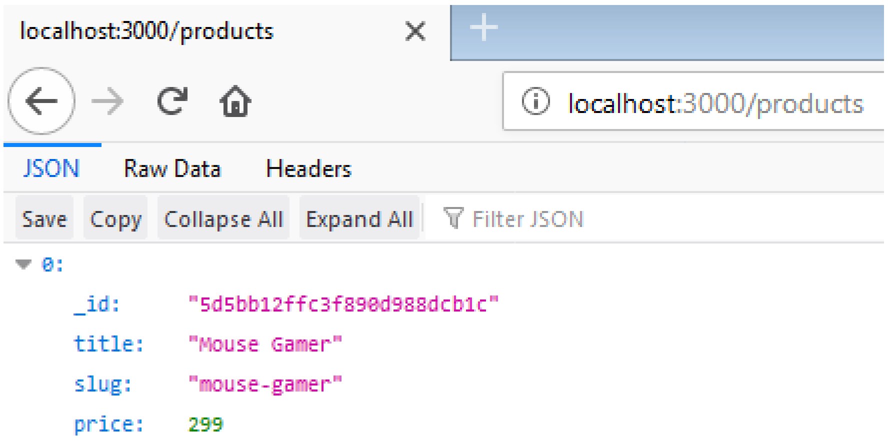
Exibindo apenas alguns campos

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);
};
```



```
[
    {
        "_id": "5d5bb12ffc3f890d988dcb1c",
        "title": "Mouse Gamer",
        "slug": "mouse-gamer",
        "price": 299
    }
]
```



```
0:
    _id:      "5d5bb12ffc3f890d988dcb1c"
    title:    "Mouse Gamer"
    slug:     "mouse-gamer"
    price:    299
```

Aula 18 - Listando um produto pelo slug

src/controllers/product-controller.js

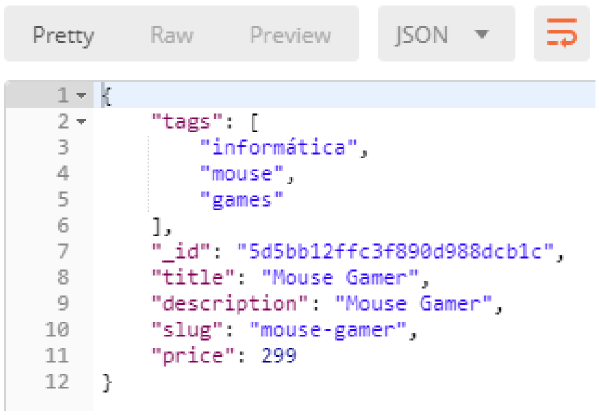
```
'use strict';

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;
```

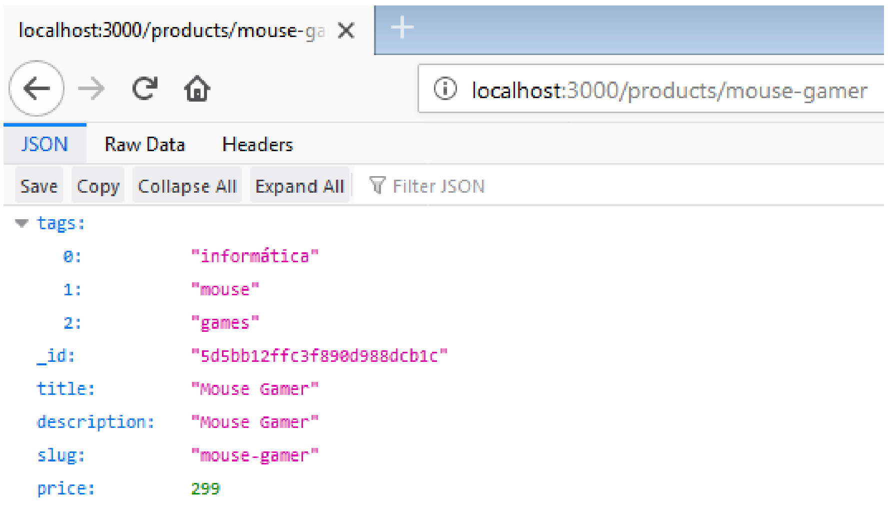
Testando no Postman

GET - localhost:3000/products/mouse-gamer



```
{
    "tags": [
        "informática",
        "mouse",
        "games"
    ],
    "_id": "5d5bb12ffc3f890d988dcb1c",
    "title": "Mouse Gamer",
    "description": "Mouse Gamer",
    "slug": "mouse-gamer",
    "price": 299
}
```

No navegador


localhost:3000/products/mouse-ga
localhost:3000/products/mouse-gamer
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
tags:
    0:          "informática"
    1:          "mouse"
    2:          "games"
_id:            "5d5bb12ffc3f890d988dcb1c"
title:          "Mouse Gamer"
description:    "Mouse Gamer"
slug:           "mouse-gamer"
price:          299
```

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
```

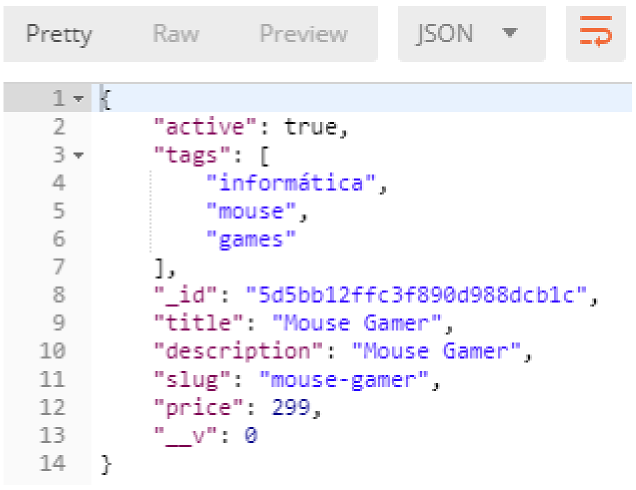
```
    });
};

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;
```

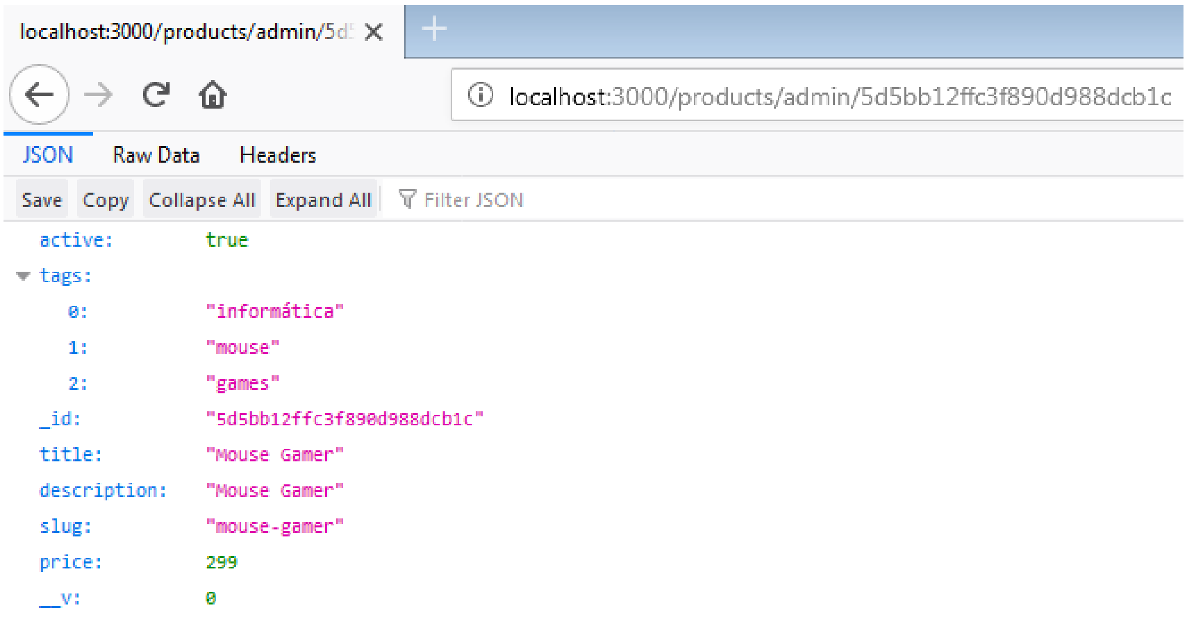
Testando no Postman

GET - localhost:3000/products/admin/5d5bb12ffc3f890d988dcb1c



```
{
    "active": true,
    "tags": [
        "informática",
        "mouse",
        "games"
    ],
    "_id": "5d5bb12ffc3f890d988dcb1c",
    "title": "Mouse Gamer",
    "description": "Mouse Gamer",
    "slug": "mouse-gamer",
    "price": 299,
    "__v": 0
}
```

Testando no navegador


localhost:3000/products/admin/5d
localhost:3000/products/admin/5d5bb12ffc3f890d988dcb1c
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON

```
active:         true
▼ tags:
   0:           "informática"
   1:           "mouse"
   2:           "games"
_id:            "5d5bb12ffc3f890d988dcb1c"
title:          "Mouse Gamer"
description:    "Mouse Gamer"
slug:           "mouse-gamer"
price:          299
__v:            0
```

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
});
```

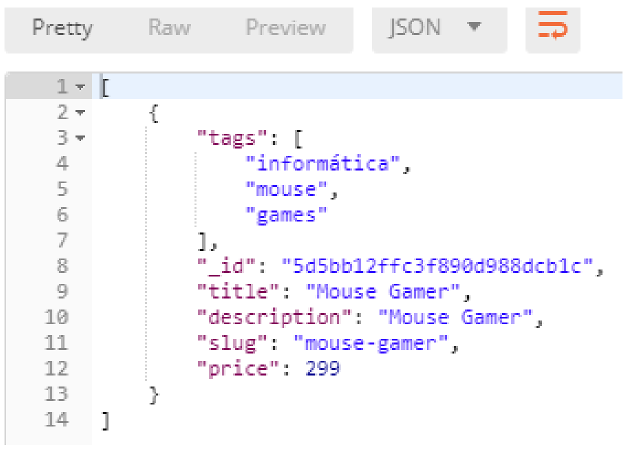
```
    });
};

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



```
[
    {
        "tags": [
            "informática",
            "mouse",
            "games"
        ],
        "_id": "5d5bb12ffc3f890d988dcb1c",
        "title": "Mouse Gamer",
        "description": "Mouse Gamer",
        "slug": "mouse-gamer",
        "price": 299
    }
]
```

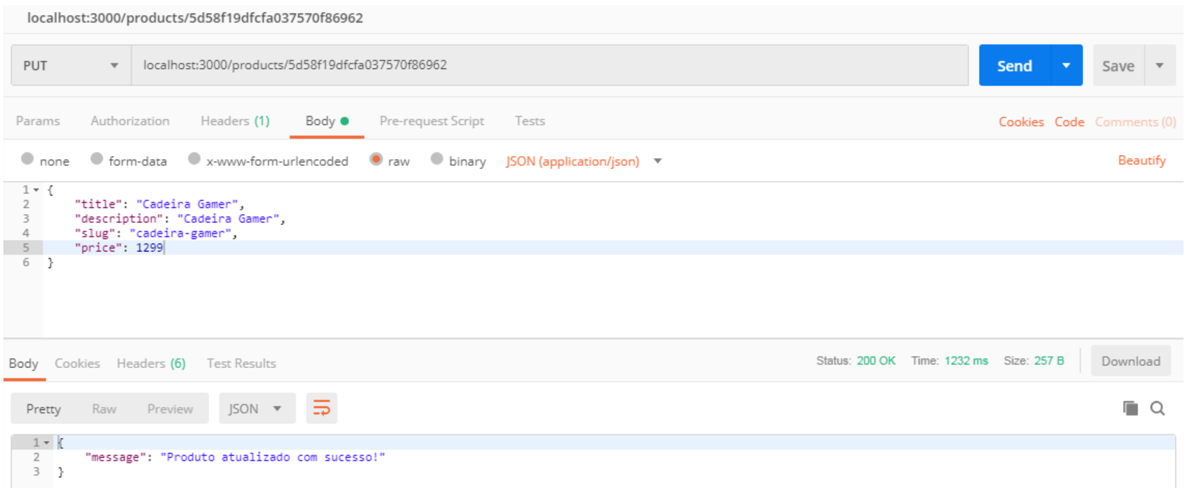
Aula 21 - Atualizando um produto

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/



localhost:3000/products/5d58f19dfcfa037570f86962

PUT localhost:3000/products/5d58f19dfcfa037570f86962

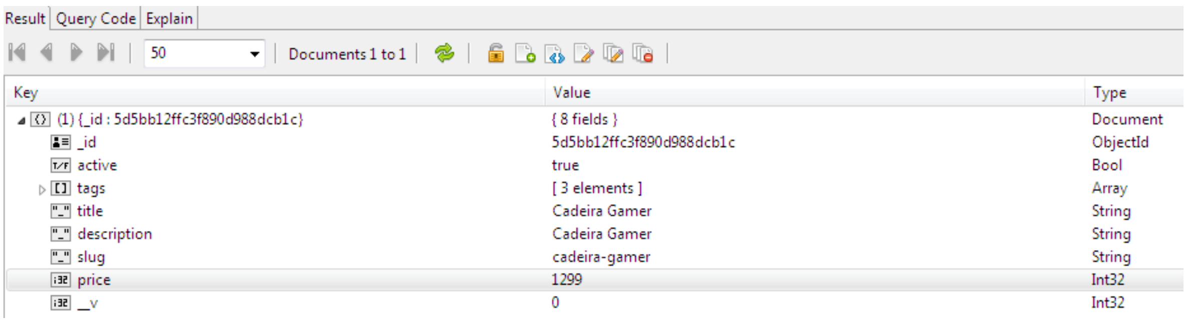
Body: raw, JSON (application/json)

```
{
    "title": "Cadeira Gamer",
    "description": "Cadeira Gamer",
    "slug": "cadeira-gamer",
    "price": 1299
}
```

Status: 200 OK Time: 1232 ms Size: 257 B

```
{
    "message": "Produto atualizado com sucesso!"
}
```

No Studio 3T

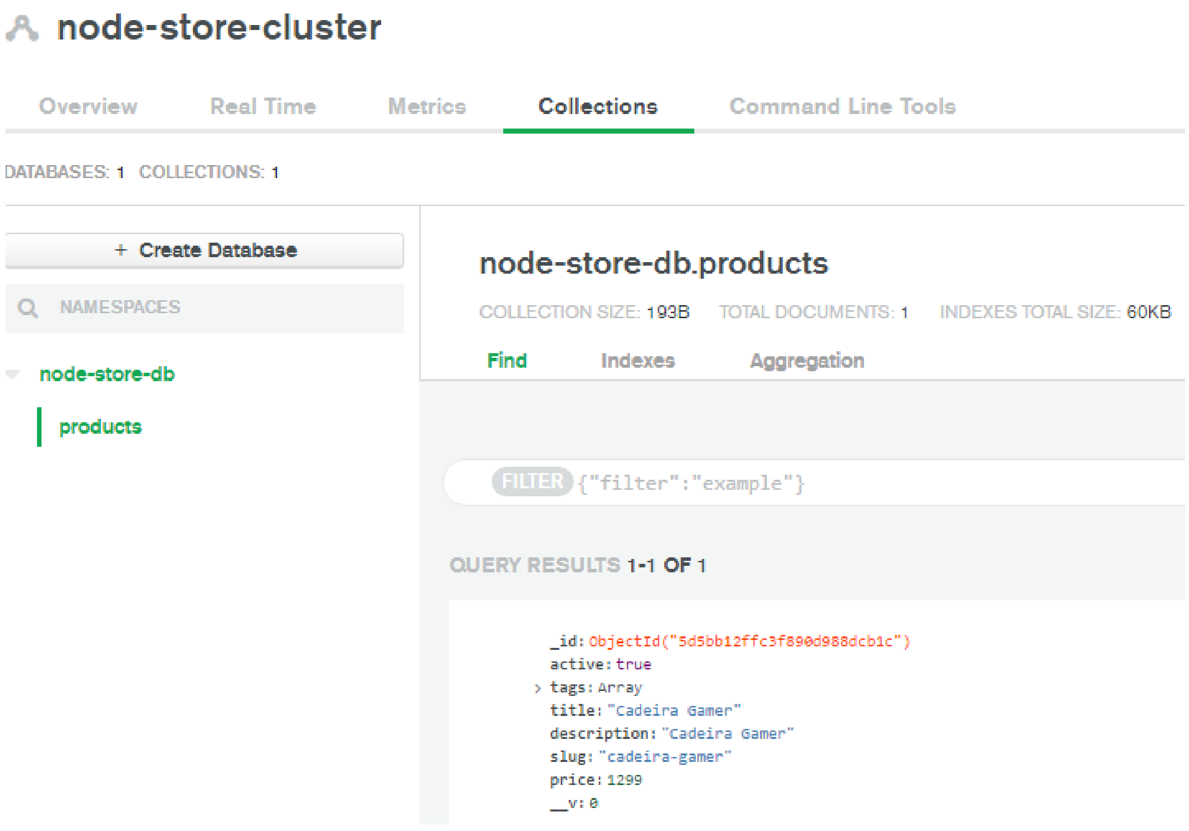


Result | Query Code | Explain

50 Documents 1 to 1

Key	Value	Type
(1) {_id : 5d5bb12ffc3f890d988dcb1c}	{ 8 fields }	Document
_id	5d5bb12ffc3f890d988dcb1c	ObjectId
active	true	Bool
tags	[3 elements]	Array
title	Cadeira Gamer	String
description	Cadeira Gamer	String
slug	cadeira-gamer	String
price	1299	Int32
__v	0	Int32

No mlab



node-store-cluster

Overview | Real Time | Metrics | Collections | Command Line Tools

DATABASES: 1 COLLECTIONS: 1

+ Create Database

NAMESPACES

node-store-db

products

node-store-db.products

COLLECTION SIZE: 193B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 60KB

Find | Indexes | Aggregation

FILTER {"filter":"example"}

QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("5d5bb12ffc3f890d988dcb1c")
active: true
> tags: Array
title: "Cadeira Gamer"
description: "Cadeira Gamer"
slug: "cadeira-gamer"
price: 1299
__v: 0
```

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
});
```

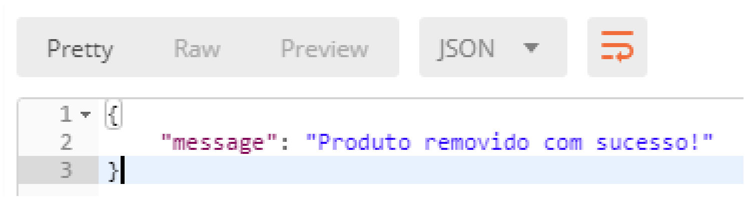
```
    });
};

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

DELETE - localhost:3000/products/5d5bb12ffc3f890d988dcb1c



```
{
    "message": "Produto removido com sucesso!"
}
```

```
}
```

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;
}
```

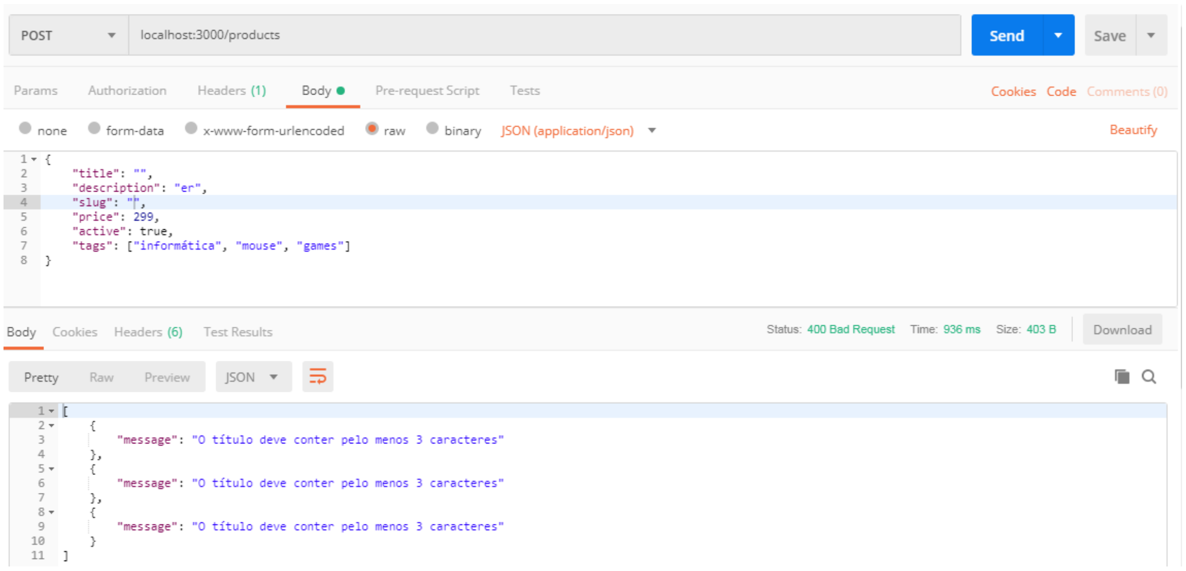
```
module.exports = ValidationContract;
```

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 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;
    }

    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
        });
    });
};
```

Testando no Postman



localhost:3000/products

raw — JSON (application/json)

```
{
    "title": "",
    "description": "er",
    "slug": "",
    "price": 299,
    "active": true,
    "tags": ["informática", "mouse", "games"]
}
```

Status: 400 Bad Request Time: 936 ms Size: 403 B

```
[
    {
        "message": "O título deve conter pelo menos 3 caracteres"
    },
    {
        "message": "O título deve conter pelo menos 3 caracteres"
    },
    {
        "message": "O título deve conter pelo menos 3 caracteres"
    }
]
```

Aula 24 - Repositórios

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

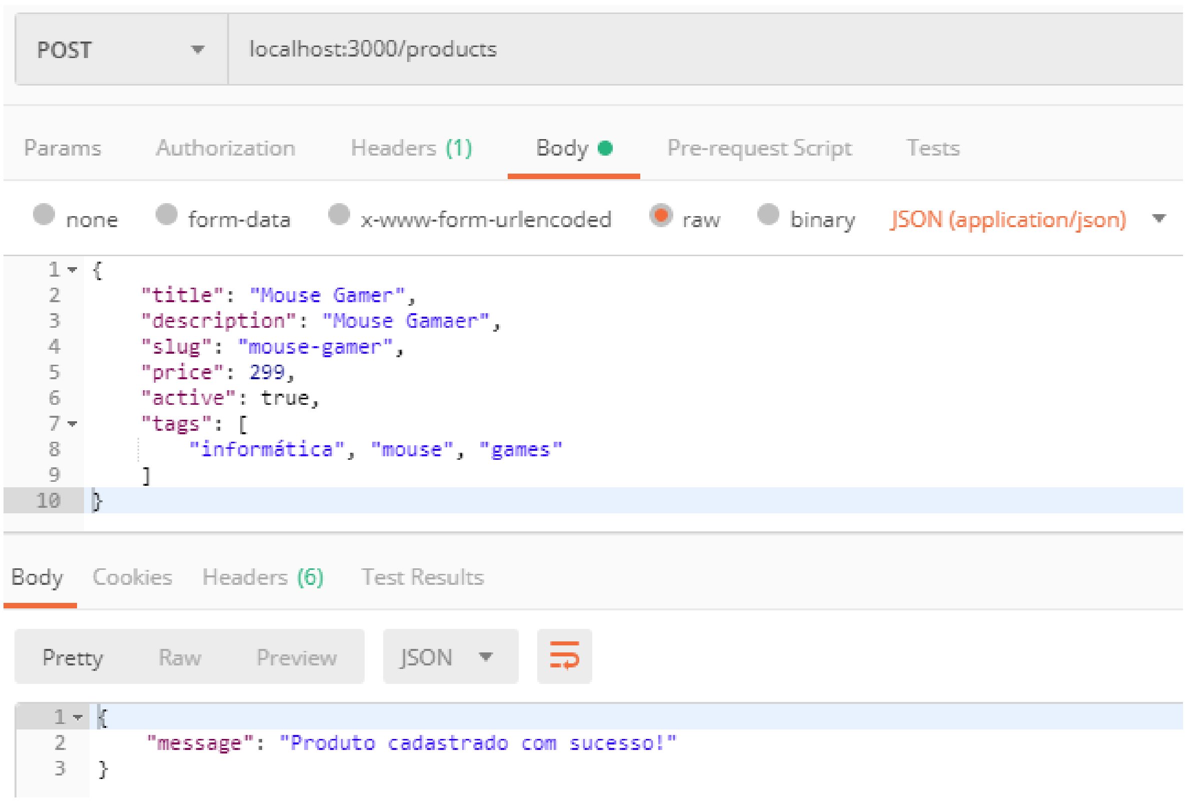
```
'use strict';

const 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 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;
   }

   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
        });
    });
};
```

Cadastrando um produto

POST - localhost:3000/products

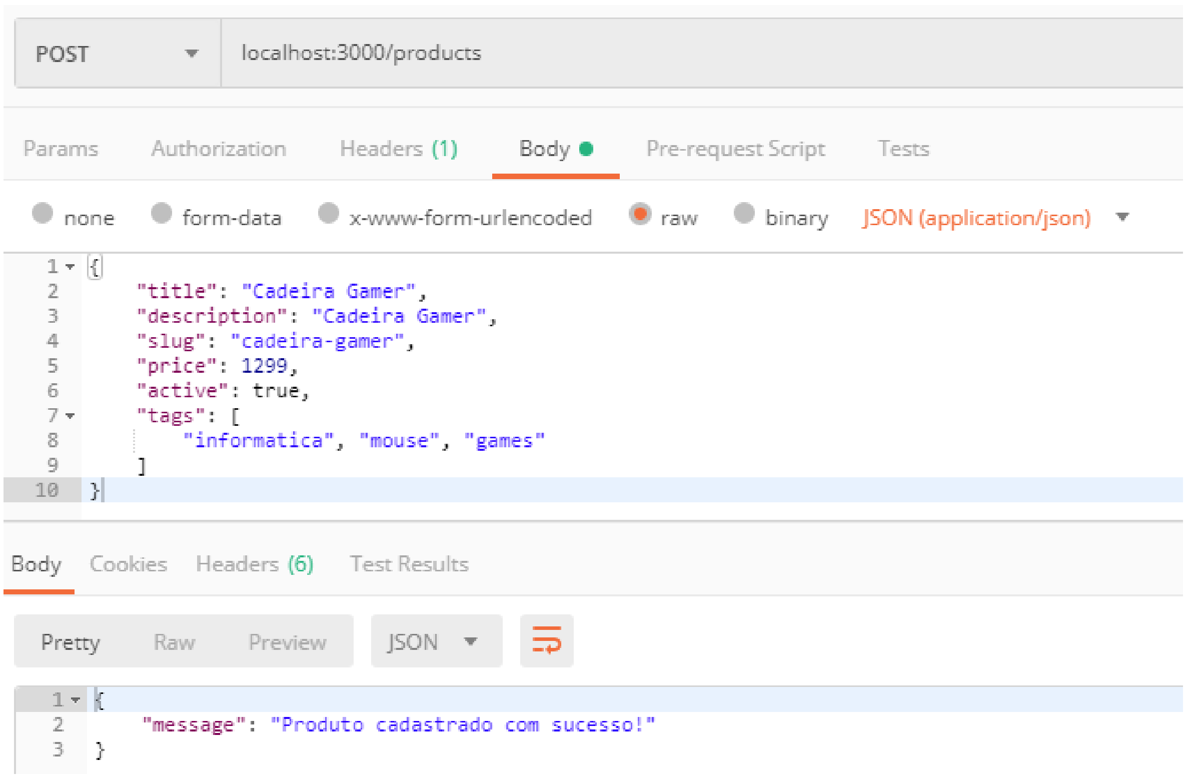


```
{
    "title": "Mouse Gamer",
    "description": "Mouse Gamaer",
    "slug": "mouse-gamer",
    "price": 299,
    "active": true,
    "tags": [
        "informática", "mouse", "games"
    ]
}
```

```
{
    "message": "Produto cadastrado com sucesso!"
}
```

Cadastrando um segundo produto

POST - localhost:3000/products

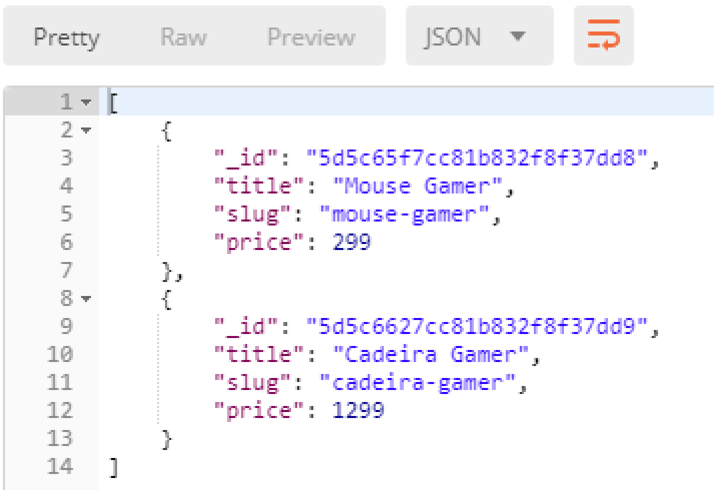


```
{
    "title": "Cadeira Gamer",
    "description": "Cadeira Gamer",
    "slug": "cadeira-gamer",
    "price": 1299,
    "active": true,
    "tags": [
        "informatica", "mouse", "games"
    ]
}
```

```
{
    "message": "Produto cadastrado com sucesso!"
}
```

Listando os produtos

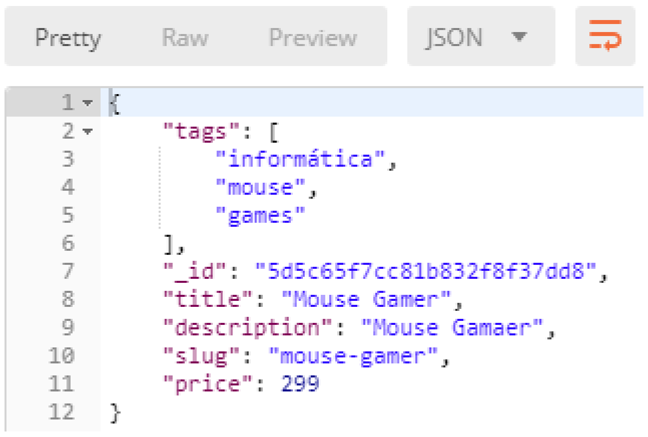
GET - localhost:3000/products



```
[
    {
        "_id": "5d5c65f7cc81b832f8f37dd8",
        "title": "Mouse Gamer",
        "slug": "mouse-gamer",
        "price": 299
    },
    {
        "_id": "5d5c6627cc81b832f8f37dd9",
        "title": "Cadeira Gamer",
        "slug": "cadeira-gamer",
        "price": 1299
    }
]
```

Exibindo dados de um produto por slug

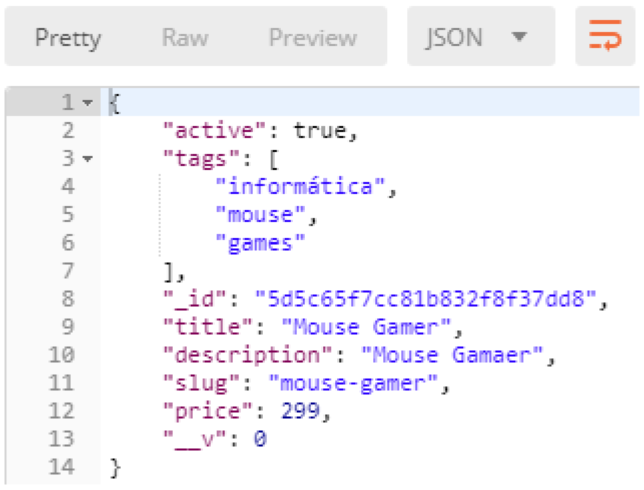
GET - localhost:3000/products/mouse-gamer



```
{
    "tags": [
        "informática",
        "mouse",
        "games"
    ],
    "_id": "5d5c65f7cc81b832f8f37dd8",
    "title": "Mouse Gamer",
    "description": "Mouse Gamaer",
    "slug": "mouse-gamer",
    "price": 299
}
```

Exibindo dados de um produto por id

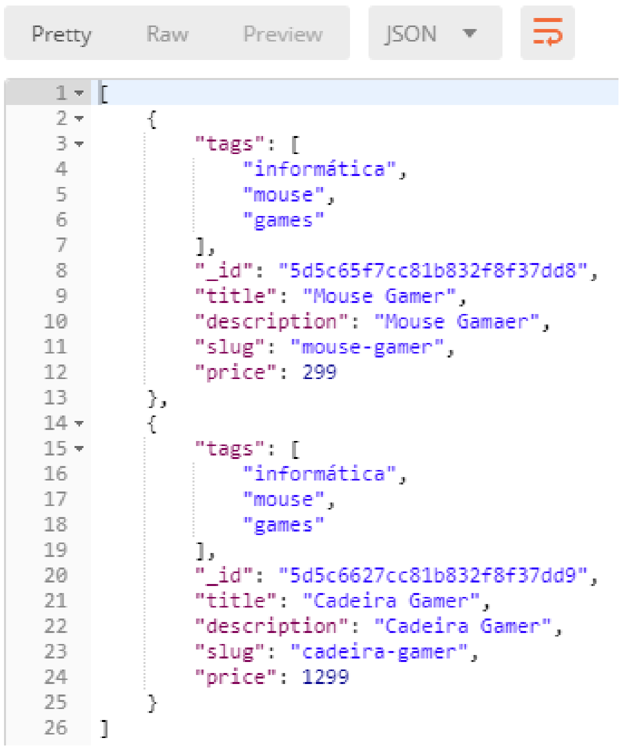
localhost:3000/products/admin/ 5d5c65f7cc81b832f8f37dd8



```
{
    "active": true,
    "tags": [
        "informática",
        "mouse",
        "games"
    ],
    "_id": "5d5c65f7cc81b832f8f37dd8",
    "title": "Mouse Gamer",
    "description": "Mouse Gamaer",
    "slug": "mouse-gamer",
    "price": 299,
    "__v": 0
}
```

Exibindo produtos por tag

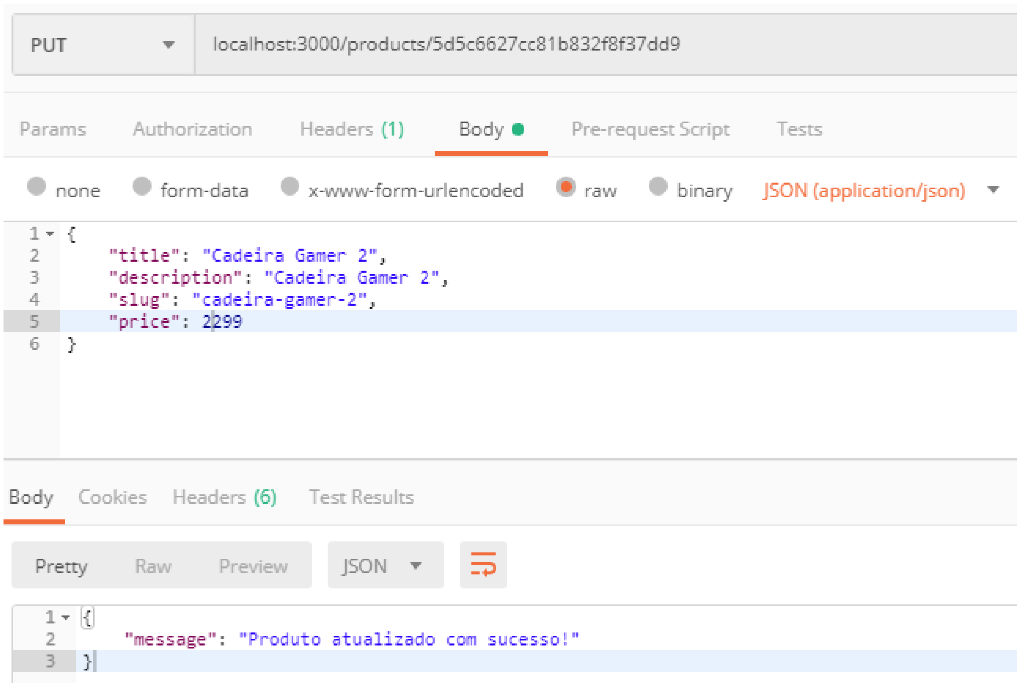
GET - localhost:3000/products/tags/games



```
[
    {
        "tags": [
            "informática",
            "mouse",
            "games"
        ],
        "_id": "5d5c65f7cc81b832f8f37dd8",
        "title": "Mouse Gamer",
        "description": "Mouse Gamaer",
        "slug": "mouse-gamer",
        "price": 299
    },
    {
        "tags": [
            "informática",
            "mouse",
            "games"
        ],
        "_id": "5d5c6627cc81b832f8f37dd9",
        "title": "Cadeira Gamer",
        "description": "Cadeira Gamer",
        "slug": "cadeira-gamer",
        "price": 1299
    }
]
```

Atualizando dados de um produto

PUT - localhost:3000/products/5d5c6627cc81b832f8f37dd9

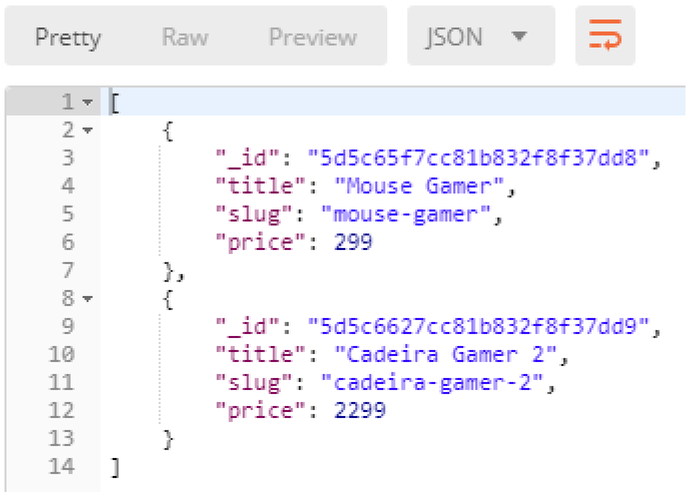


```
{
    "title": "Cadeira Gamer 2",
    "description": "Cadeira Gamer 2",
    "slug": "cadeira-gamer-2",
    "price": 2299
}
```

```
{
    "message": "Produto atualizado com sucesso!"
}
```

Listando os produtos

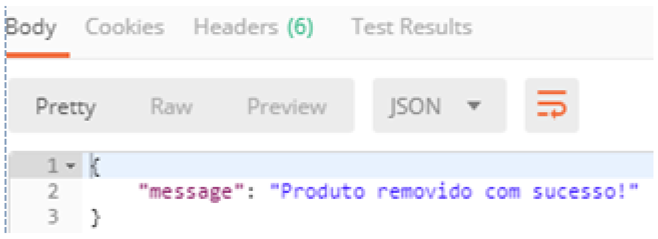
GET - localhost:3000/products



```
[
    {
        "_id": "5d5c65f7cc81b832f8f37dd8",
        "title": "Mouse Gamer",
        "slug": "mouse-gamer",
        "price": 299
    },
    {
        "_id": "5d5c6627cc81b832f8f37dd9",
        "title": "Cadeira Gamer 2",
        "slug": "cadeira-gamer-2",
        "price": 2299
    }
]
```

Excluindo um produto

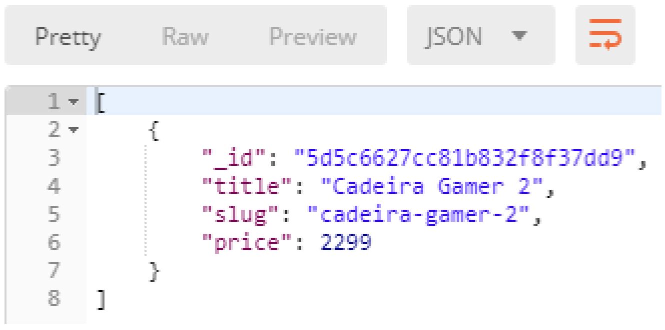
DELETE - localhost:3000/products/5d5c6627cc81b832f8f37dd9



```
{
    "message": "Produto removido com sucesso!"
}
```

Listando os produtos

GET - localhost:3000/products



```
[
    {
        "_id": "5d5c6627cc81b832f8f37dd9",
        "title": "Cadeira Gamer 2",
        "slug": "cadeira-gamer-2",
        "price": 2299
    }
]
```

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 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(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

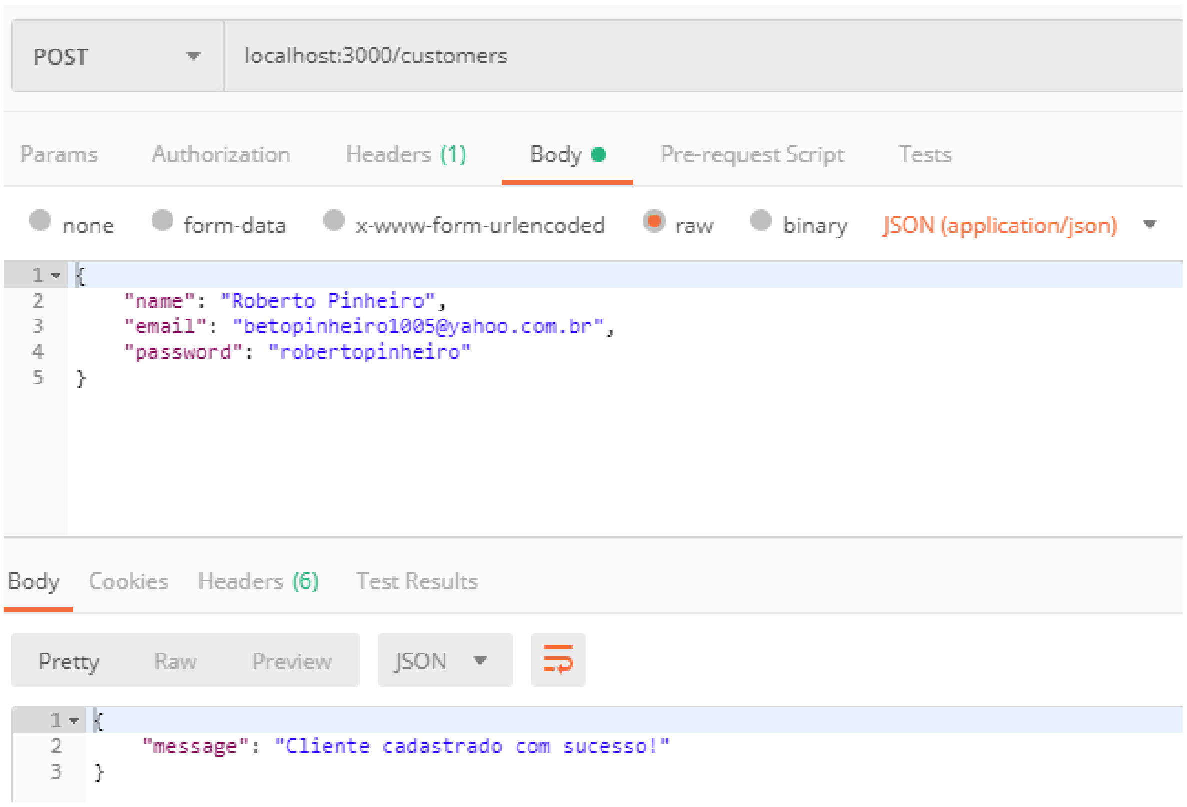
```
const express = require('express');
const router = express.Router();
const controller = require("../controllers/customer-controller");

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;
```

Criando um cliente (customer)

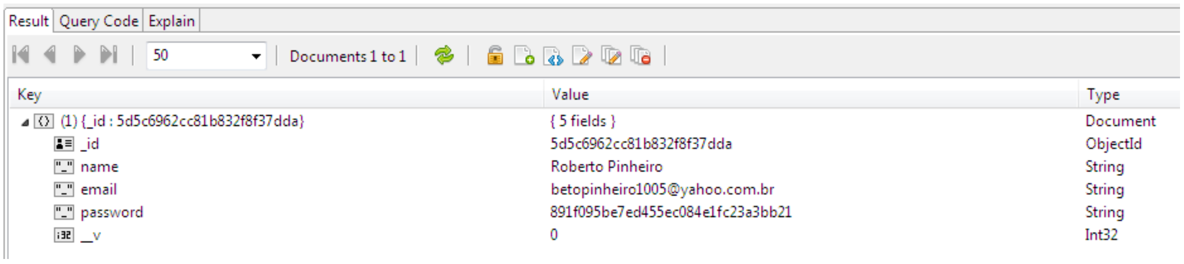
POST - localhost:3000/customers



```
{
    "name": "Roberto Pinheiro",
    "email": "betopinheiro1005@yahoo.com.br",
    "password": "robertopinheiro"
}
```

```
{
    "message": "Cliente cadastrado com sucesso!"
}
```

No Studio 3T:

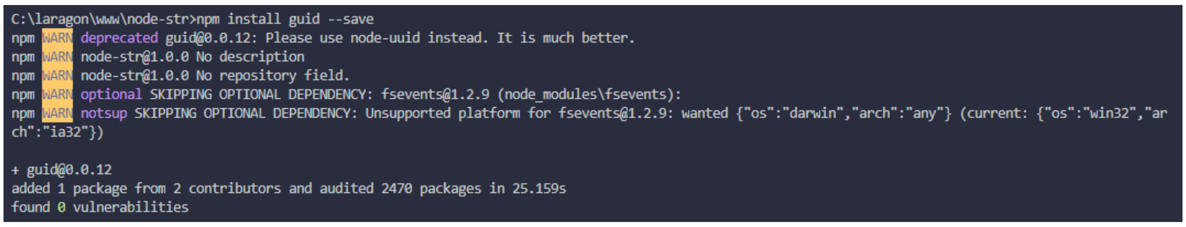


Key	Value	Type
(1) {_id : 5d5c6962cc81b832f8f37dda}	{ 5 fields }	Document
_id	5d5c6962cc81b832f8f37dda	ObjectId
name	Roberto Pinheiro	String
email	betopinheiro1005@yahoo.com.br	String
password	891f095be7ed455ec084e1fc23a3bb21	String
__v	0	Int32

Aula 29 - Revisitando os Controllers: Order

Instalação do pacote guid

```
npm install guid --save
```



```
C:\laragon\www\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@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","ar
ch":"ia32"})

+ guid@0.0.12
added 1 package from 2 contributors and audited 2470 packages in 25.159s
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/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();
}
```

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;
```

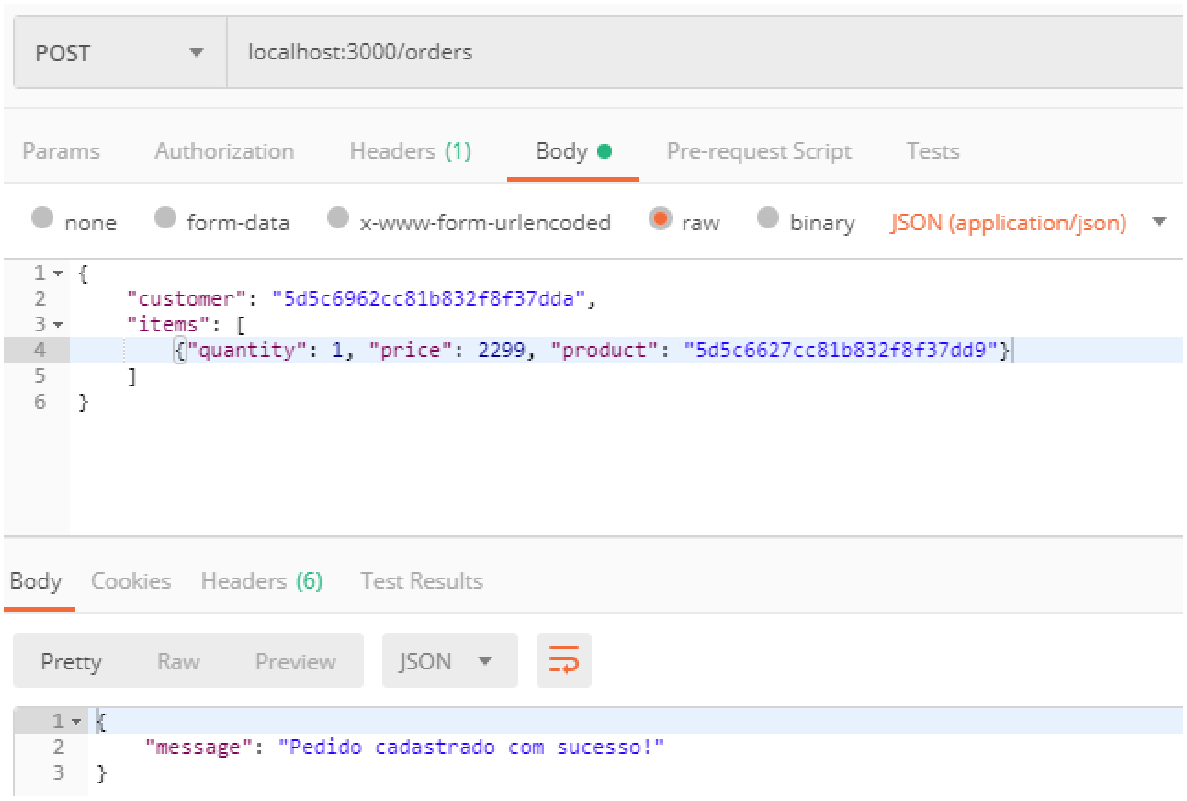
src/app.js

```
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;
```

Testando no Postman

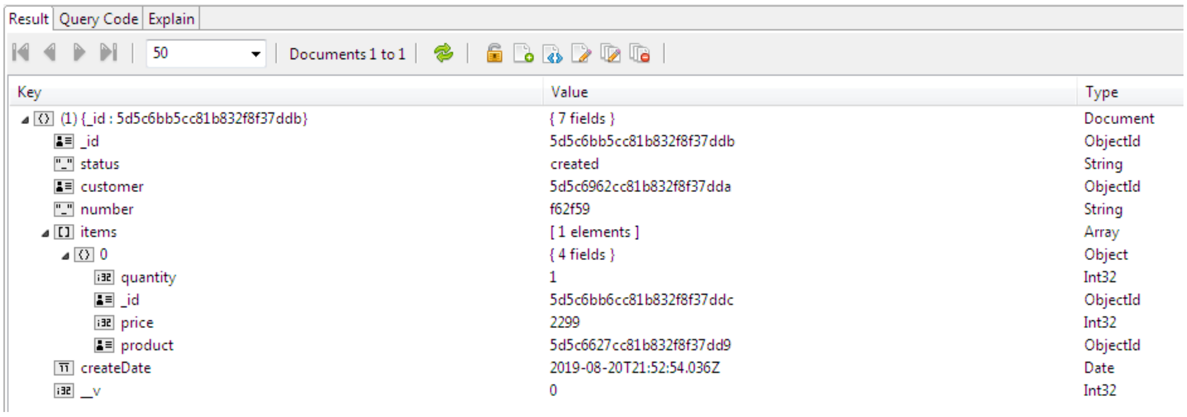
POST - localhost:3000/orders



```
{
    "customer": "5d5c6962cc81b832f8f37dda",
    "items": [
        {"quantity": 1, "price": 2299, "product": "5d5c6627cc81b832f8f37dd9"}
    ]
}
```

```
{
    "message": "Pedido cadastrado com sucesso!"
}
```

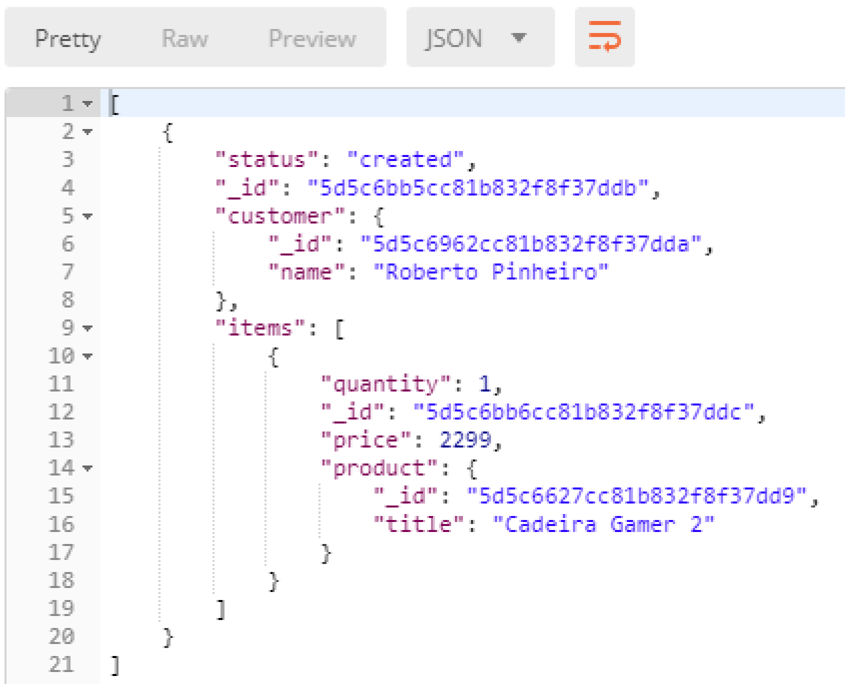
No Studio 3T:



Key	Value	Type
(1) {_id : 5d5c6bb5cc81b832f8f37ddb}	{ 7 fields }	Document
_id	5d5c6bb5cc81b832f8f37ddb	ObjectId
status	created	String
customer	5d5c6962cc81b832f8f37dda	ObjectId
number	f62f59	String
items	[1 elements]	Array
0	{ 4 fields }	Object
quantity	1	Int32
_id	5d5c6bb6cc81b832f8f37ddc	ObjectId
price	2299	Int32
product	5d5c6627cc81b832f8f37dd9	ObjectId
createDate	2019-08-20T21:52:54.036Z	Date
__v	0	Int32

Listando os pedidos no Postman

GET - localhost:3000/orders



```
[
    {
        "status": "created",
        "_id": "5d5c6bb5cc81b832f8f37ddb",
        "customer": {
            "_id": "5d5c6962cc81b832f8f37dda",
            "name": "Roberto Pinheiro"
        },
        "items": [
            {
                "quantity": 1,
                "_id": "5d5c6bb6cc81b832f8f37ddc",
                "price": 2299,
                "product": {
                    "_id": "5d5c6627cc81b832f8f37dd9",
                    "title": "Cadeira Gamer 2"
                }
            }
        ]
    }
]
```

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:<password> @node-
store-cluster-nlcnv.mongodb.net/node-str-db?retryWrites=true&w=majority',
   sendgridKey: 'TBD',
   containerConnectionString: 'TBD'
}
```

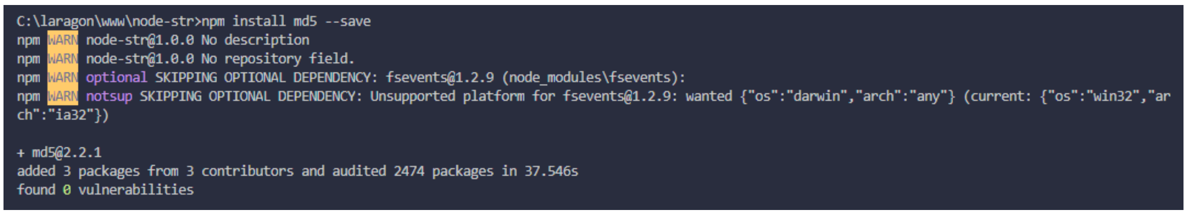
src/app.js

```
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, { 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;
```

Aula 31 - Encriptando a senha

Instalação do md5

```
npm install md5 --save
```



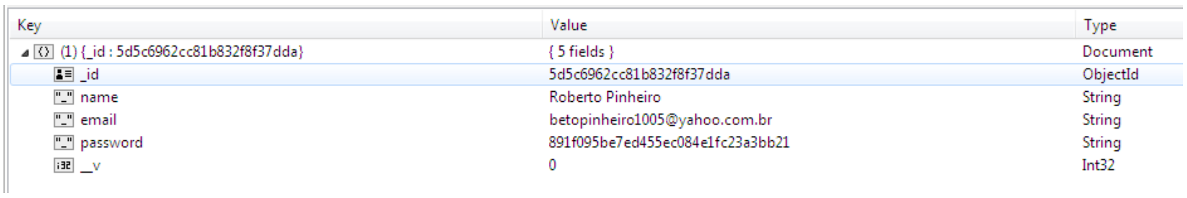
```
C:\laragon\www\node-str>npm install md5 --save
npm WARN node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","ar
ch":"ia32"})

+ md5@2.2.1
added 3 packages from 3 contributors and audited 2474 packages in 37.546s
found 0 vulnerabilities
```

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'
        });
```

```
  }
};
```

No Studio 3T



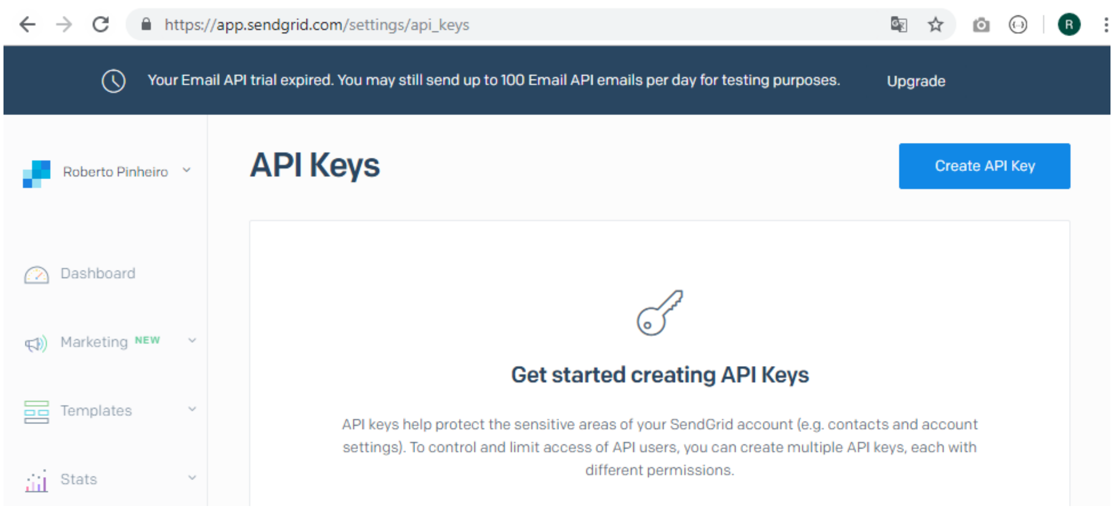
```
Key                                      Value                                   Type
(1) {_id : 5d5c6962cc81b832f8f37dda}     { 5 fields }                            Document
    _id                                  5d5c6962cc81b832f8f37dda                ObjectId
    name                                 Roberto Pinheiro                        String
    email                                betopinheiro1005@yahoo.com.br           String
    password                             891f095be7ed455ec084e1fc23a3bb21        String
    __v                                  0                                       Int32
```

Aula 32 - Enviando email de boas vindas

SendGrid

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

- Crie uma API Key

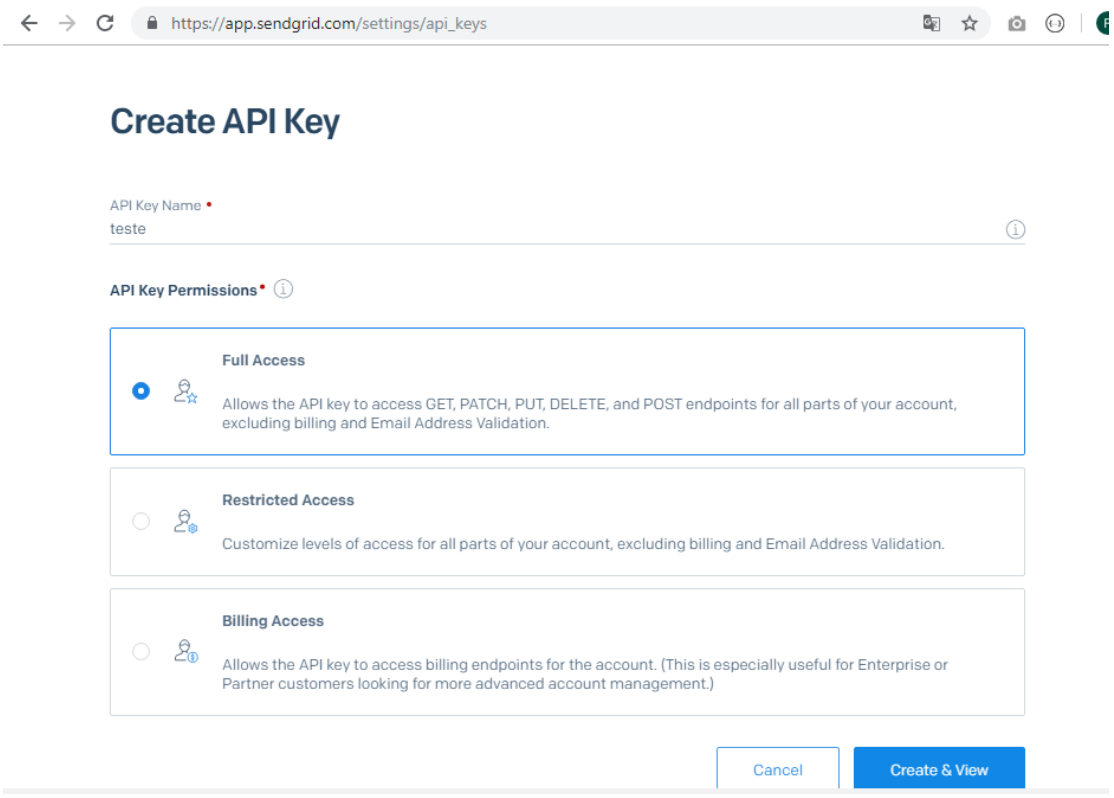

https://app.sendgrid.com/settings/api_keys

Your Email API trial expired. You may still send up to 100 Email API emails per day for testing purposes.

Get started creating API Keys

API keys help protect the sensitive areas of your SendGrid account (e.g. contacts and account settings). To control and limit access of API users, you can create multiple API keys, each with different permissions.

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


https://app.sendgrid.com/settings/api_keys

Create API Key

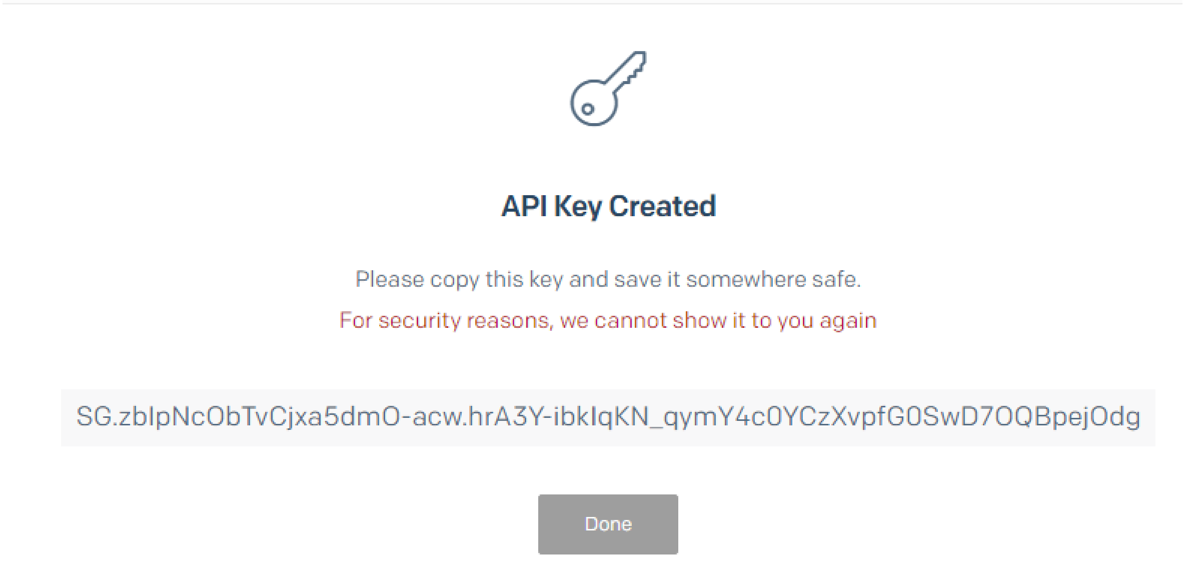
API Key Name *
teste

API Key Permissions*

[✓ Full Access]
Allows the API key to access GET, PATCH, PUT, DELETE, and POST endpoints for all parts of your account, excluding billing and Email Address Validation.

Restricted Access
Customize levels of access for all parts of your account, excluding billing and Email Address Validation.

Billing Access
Allows the API key to access billing endpoints for the account. (This is especially useful for Enterprise or Partner customers looking for more advanced account management.)

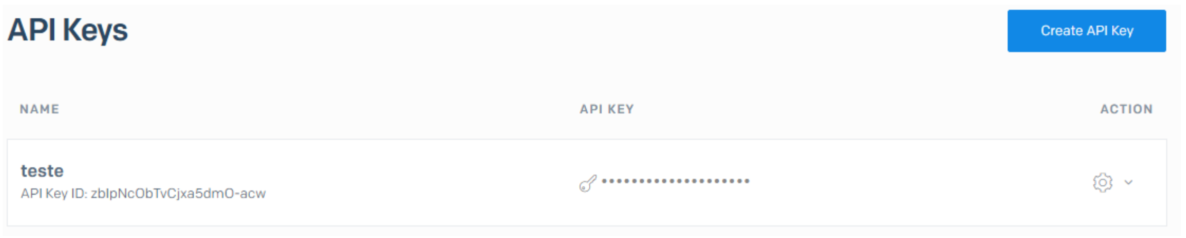


API Key Created

Please copy this key and save it somewhere safe.

For security reasons, we cannot show it to you again

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



API Keys

NAME	API KEY	ACTION
teste API Key ID: zbIpNcObTvCjxa5dmO-acw	••••••••••••••••••••	

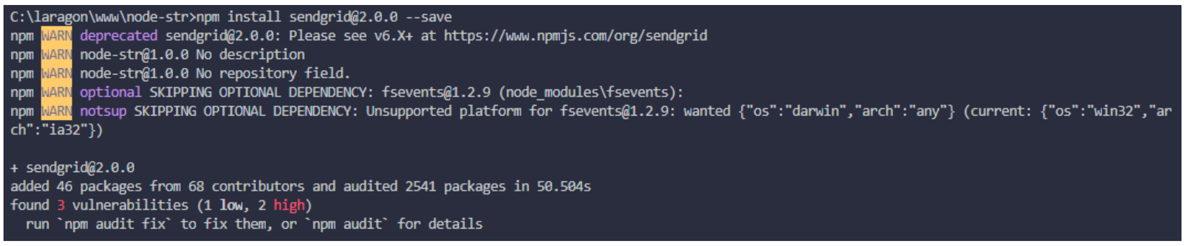
- Copie a chave e cole-a no arquivo config.js:

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:<password> @node-store-cluster-nlcnv.mongodb.net/node-str-db?retryWrites=true&w=majority',
    sendgridKey: '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',
    containerConnectionString: 'TBD'
}
```

Instalação do SendGrid

```
npm install sendgrid@2.0.0 --save
```



```
C:\laragon\www\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 node-str@1.0.0 No description
npm WARN node-str@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","ar
ch":"ia32"})

+ sendgrid@2.0.0
added 46 packages from 68 contributors and audited 2541 packages in 50.504s
found 3 vulnerabilities (1 low, 2 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

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": "^4.1.1",
    "express": "^4.17.1",
    "guid": "0.0.12",
    "http": "0.0.0",
    "md5": "^2.2.1",
    "mongoose": "^5.6.9",
    "sendgrid": "^2.0.0"
  },
  "devDependencies": {
    "nodemon": "^1.19.1"
  }
}
```

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
    });
}
```

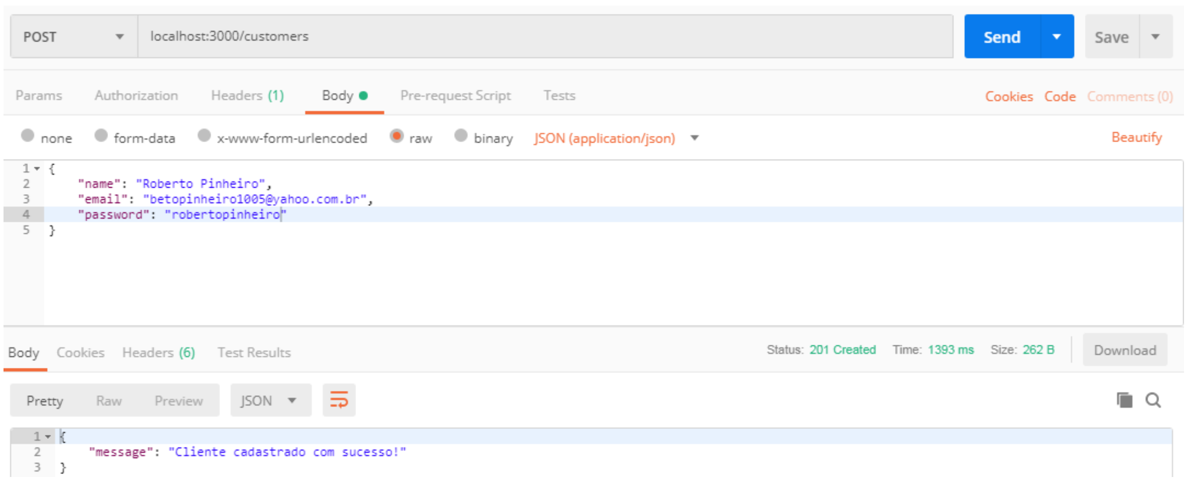
src/controllers/customer-controller.js

```
'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



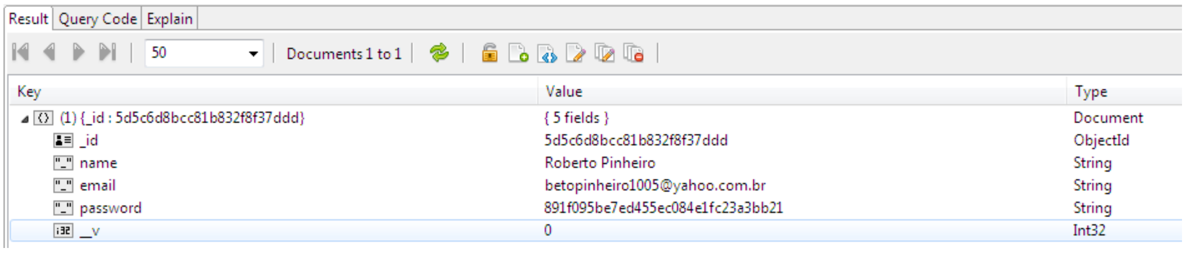
POST localhost:3000/customers

Body: raw — JSON (application/json)

```
{
    "name": "Roberto Pinheiro",
    "email": "betopinheiro1005@yahoo.com.br",
    "password": "robertopinheiro"
}
```

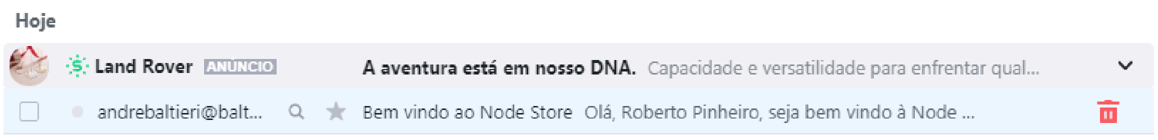
Status: 201 Created Time: 1393 ms Size: 262 B

```
{
    "message": "Cliente cadastrado com sucesso!"
}
```



Documents 1 to 1

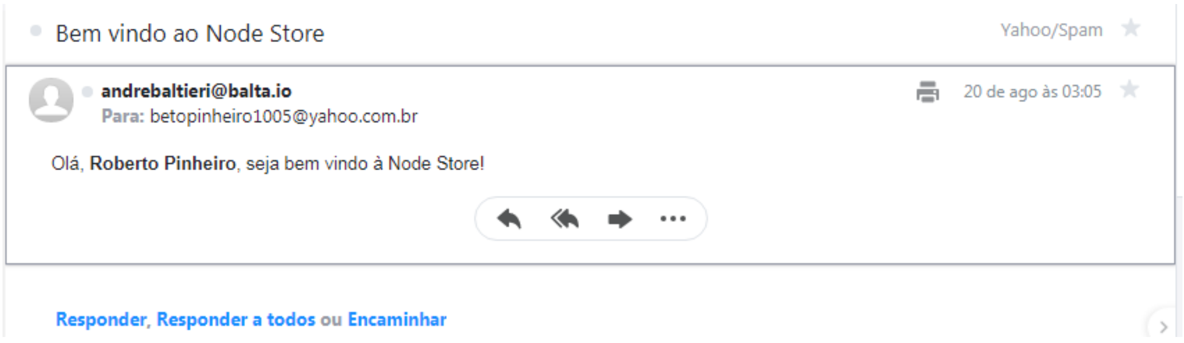
Key	Value	Type
(1) {_id : 5d5c6d8bcc81b832f8f37ddd}	{ 5 fields }	Document
_id	5d5c6d8bcc81b832f8f37ddd	ObjectId
name	Roberto Pinheiro	String
email	betopinheiro1005@yahoo.com.br	String
password	891f095be7ed455ec084e1fc23a3bb21	String
__v	0	Int32



Hoje

Land Rover ANÚNCIO A aventura está em nosso DNA. Capacidade e versatilidade para enfrentar qual...

andrebaltieri@balt... Bem vindo ao Node Store Olá, Roberto Pinheiro, seja bem vindo à Node ...


Bem vindo ao Node Store

Yahoo/Spam

andrebaltieri@balta.io
Para: betopinheiro1005@yahoo.com.br

20 de ago às 03:05

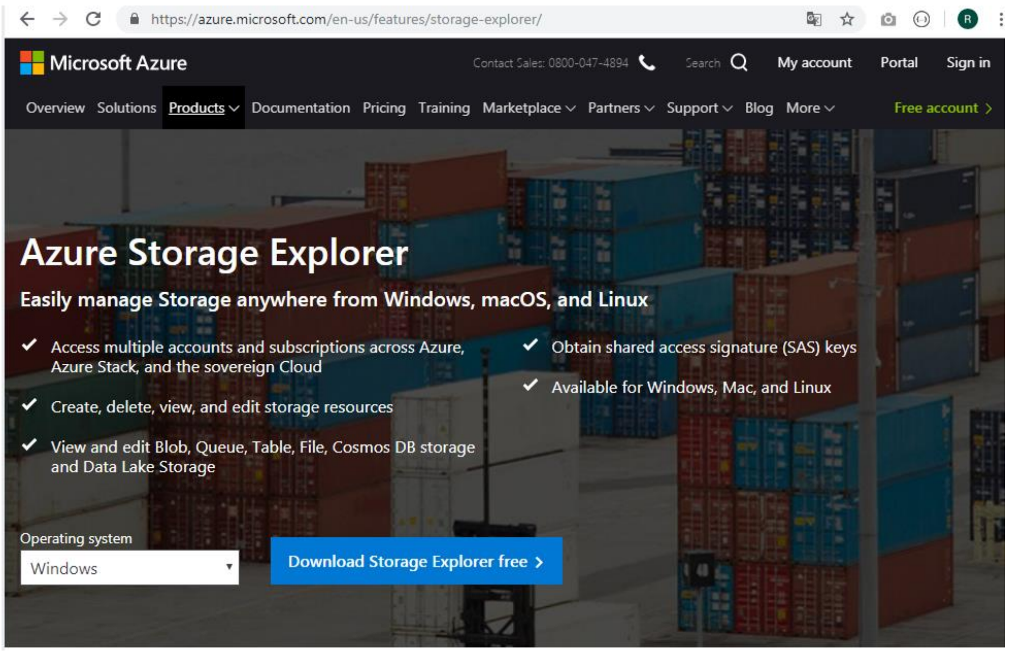
Olá, **Roberto Pinheiro**, seja bem vindo à Node Store!

Responder, Responder a todos ou Encaminhar

Aula 33 - Upload da imagem do produto

Acesse a URL:

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


https://azure.microsoft.com/en-us/features/storage-explorer/
Microsoft Azure
Contact Sales: 0800-047-4894
Search
My account
Portal
Sign in
Overview Solutions Products Documentation Pricing Training Marketplace Partners Support Blog More
Free account

Azure Storage Explorer

Easily manage Storage anywhere from Windows, macOS, and Linux

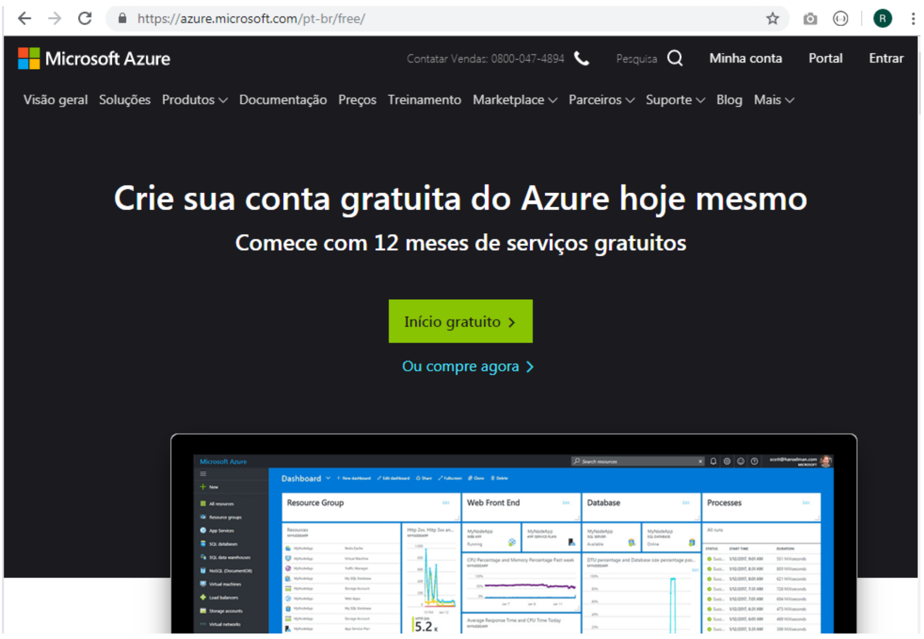
- Access multiple accounts and subscriptions across Azure, Azure Stack, and the sovereign Cloud
- Create, delete, view, and edit storage resources
- View and edit Blob, Queue, Table, File, Cosmos DB storage and Data Lake Storage
- Obtain shared access signature (SAS) keys
- Available for Windows, Mac, and Linux

Operating system: Windows

- Baixe e instale o programa Storage Explorer free.

- Crie sua conta no Azzure:

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

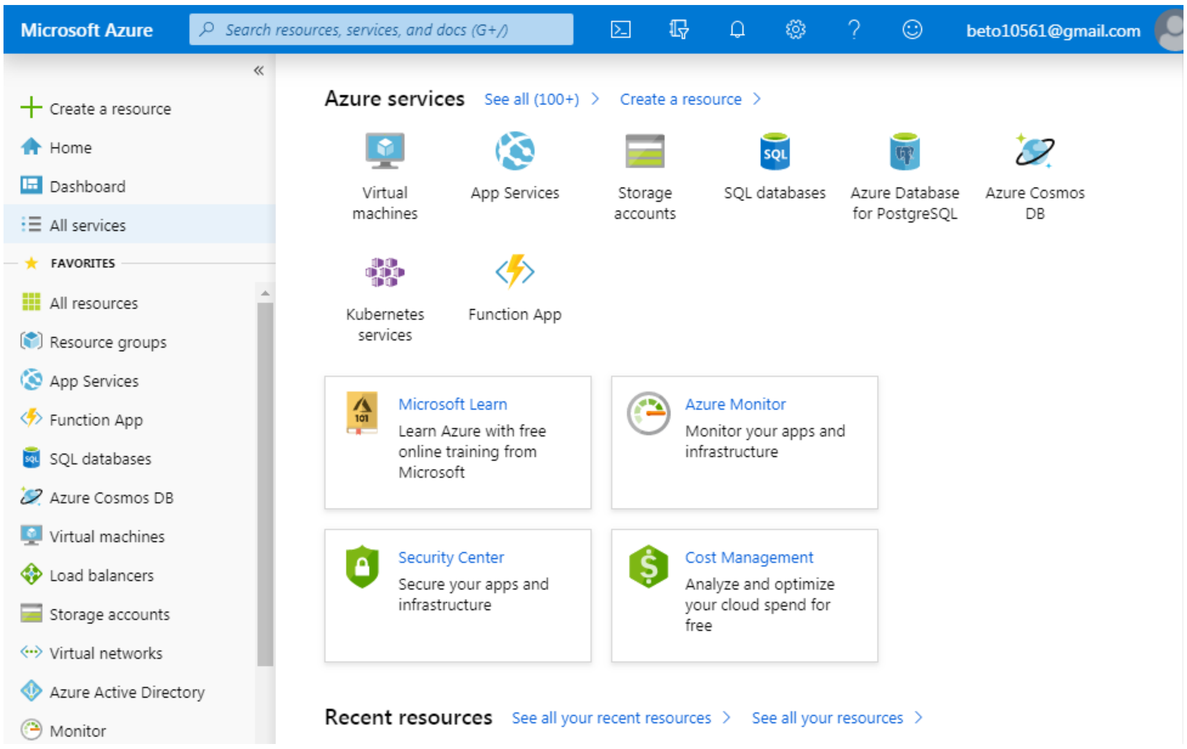

https://azure.microsoft.com/pt-br/free/
Microsoft Azure
Contatar Vendas: 0800-047-4894
Pesquisa
Minha conta
Portal
Entrar
Visão geral Soluções Produtos Documentação Preços Treinamento Marketplace Parceiros Suporte Blog Mais

Crie sua conta gratuita do Azure hoje mesmo

Comece com 12 meses de serviços gratuitos

Início gratuito
Ou compre agora
Dashboard
Resource Group
Web Front End
Database
Processes

- Acesse o portal.



- Crie um Storage Account.