

Projeto de Banco de Dados e OO .NET

Bancos de Dados Não Relacionais e NodeJS

Vantagens do NoSQL

- NoSQL é um termo usado para descrever bancos de dados não relacionais de alto desempenho. Os bancos de dados NoSQL usam diversos modelos de dados, incluindo documentos, gráficos, chave-valor e colunares. Bancos de dados NoSQL são amplamente reconhecidos pela facilidade de desenvolvimento, desempenho escalável, alta disponibilidade e resiliência.

Correlação entre Bancos SQL e NoSQL

Banco de Dados
NoSQL

Coleção

Documento

```
{  
  _id: 1,  
  nome: "João",  
  sobrenome: "da Silva"  
}
```

Banco de Dados
SQL

Base de
Dados

Tabela

_id	nome	sobrenome
1	João	da Silva

No Schema

Banco de Dados

Coleção

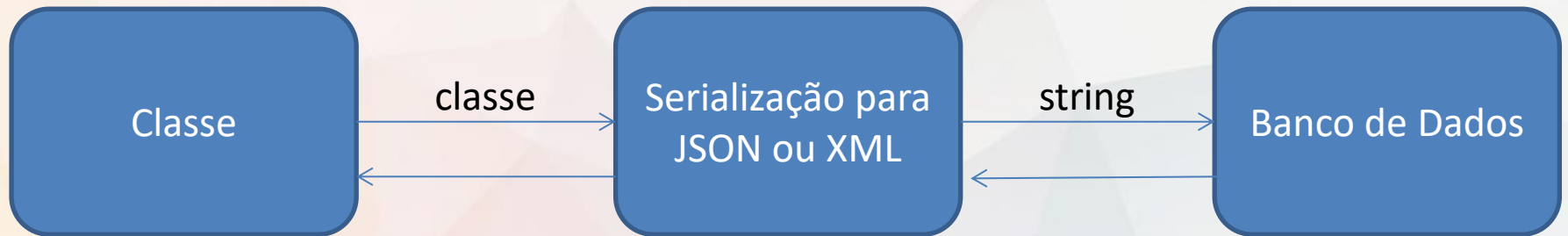
Documento

```
{  
  _id: 1,  
  nome: "João",  
  sobrenome: "da Silva"  
}
```

```
{  
  _id: 2,  
  nome: "Maria",  
  sobrenome: "da Silva",  
  idade: 22,  
  estado_civil: "casada",  
  casada_com: 1  
}
```

```
{  
  _id: 3,  
  nome: "Leonardo",  
  sobrenome: "da Silva",  
  idade: 1,  
  pais: [ {_id: 1}, {_id: 2} ]  
}
```

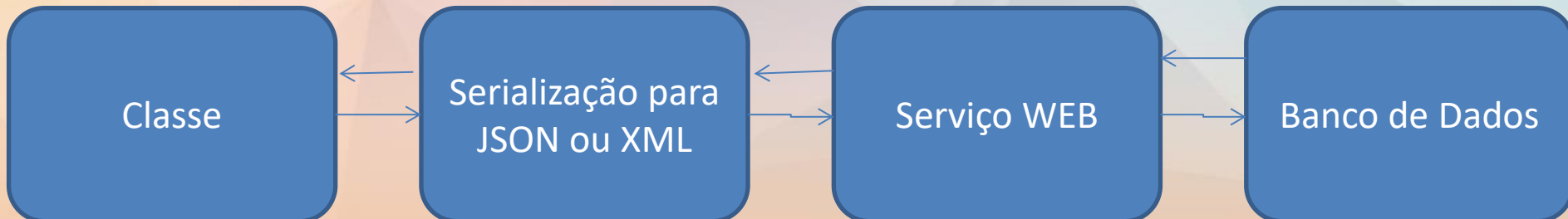
Facilidade na Persistência de Objetos



Não existe necessidade de modelar tabelas e criar relacionamento entre os dados

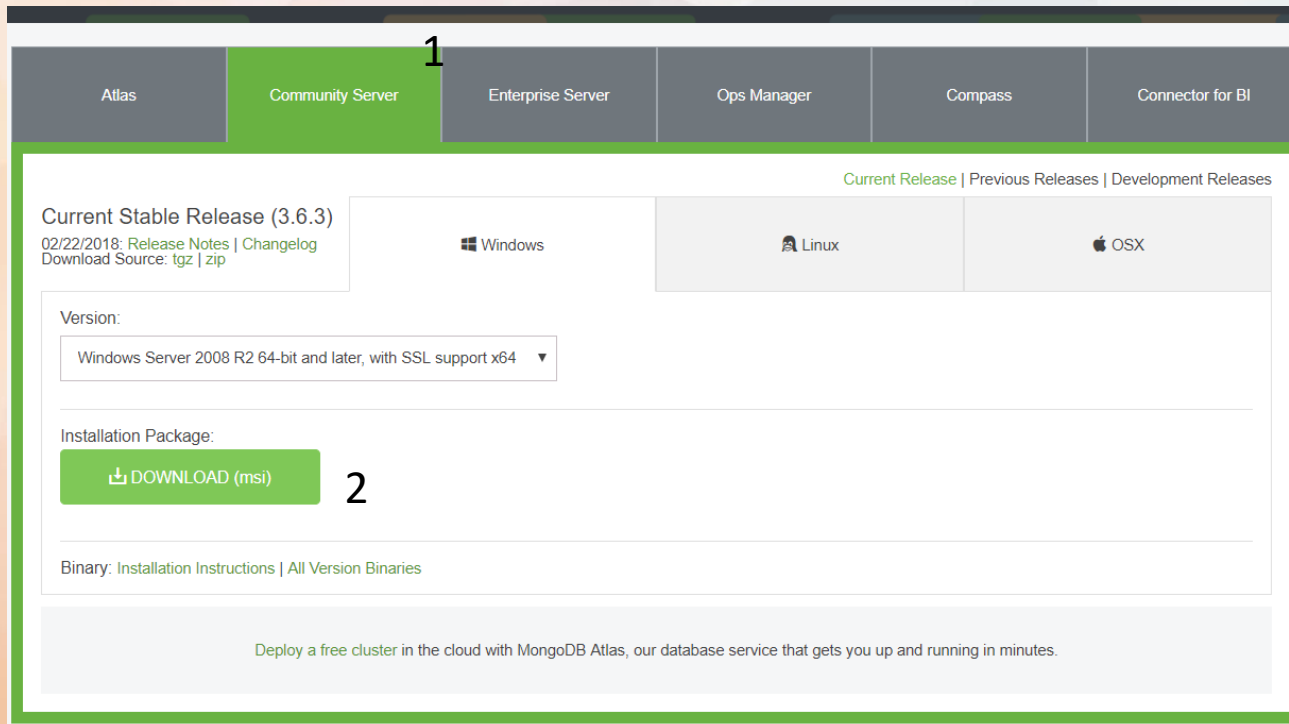
Facilmente utilizável com webservices REST

Dificuldade em manter o relacionamento de diversos dados, compensada com replicação dos dados



Antes de Iniciar

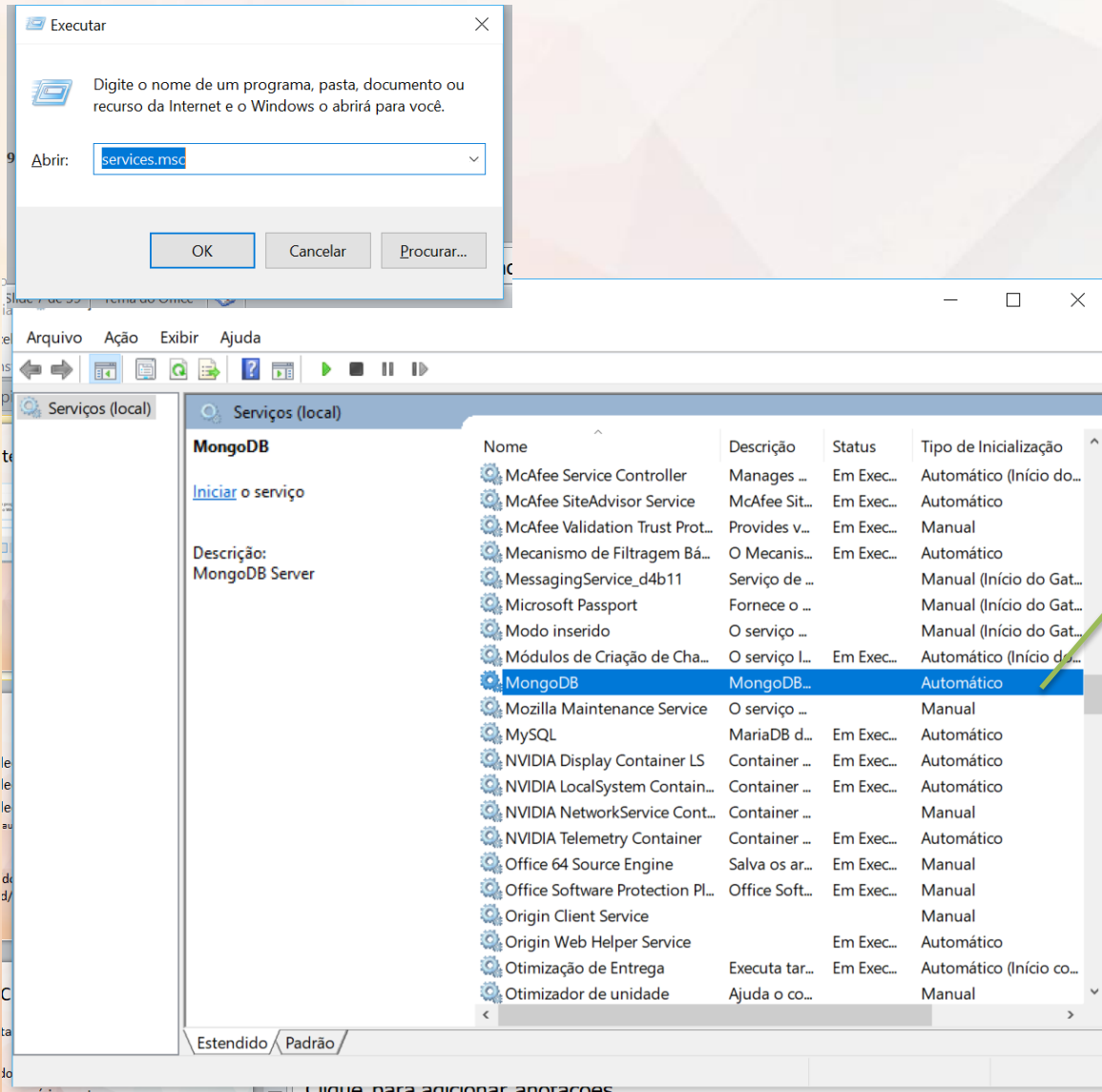
- É necessário instalar os dois software abaixo
 - NodeJs: <https://nodejs.org/en/>
 - Mongo Community Server: <https://www.mongodb.com/download-center>



Colocando o Mongo DB como Serviço do Windows

- Instale o mongoDB na sua máquina.
- Crie uma pasta c:\mongo
- Abrir o Console do Windows (CMD) como Administrador
 - CMD> cd C:\Program Files\MongoDB\Server\3.6\bin
 - CMD> mongod --remove
 - CMD> mongod --install --dbpath=C:\mongo --logpath=C:\mongo\log.txt

Execute o comando services.msc

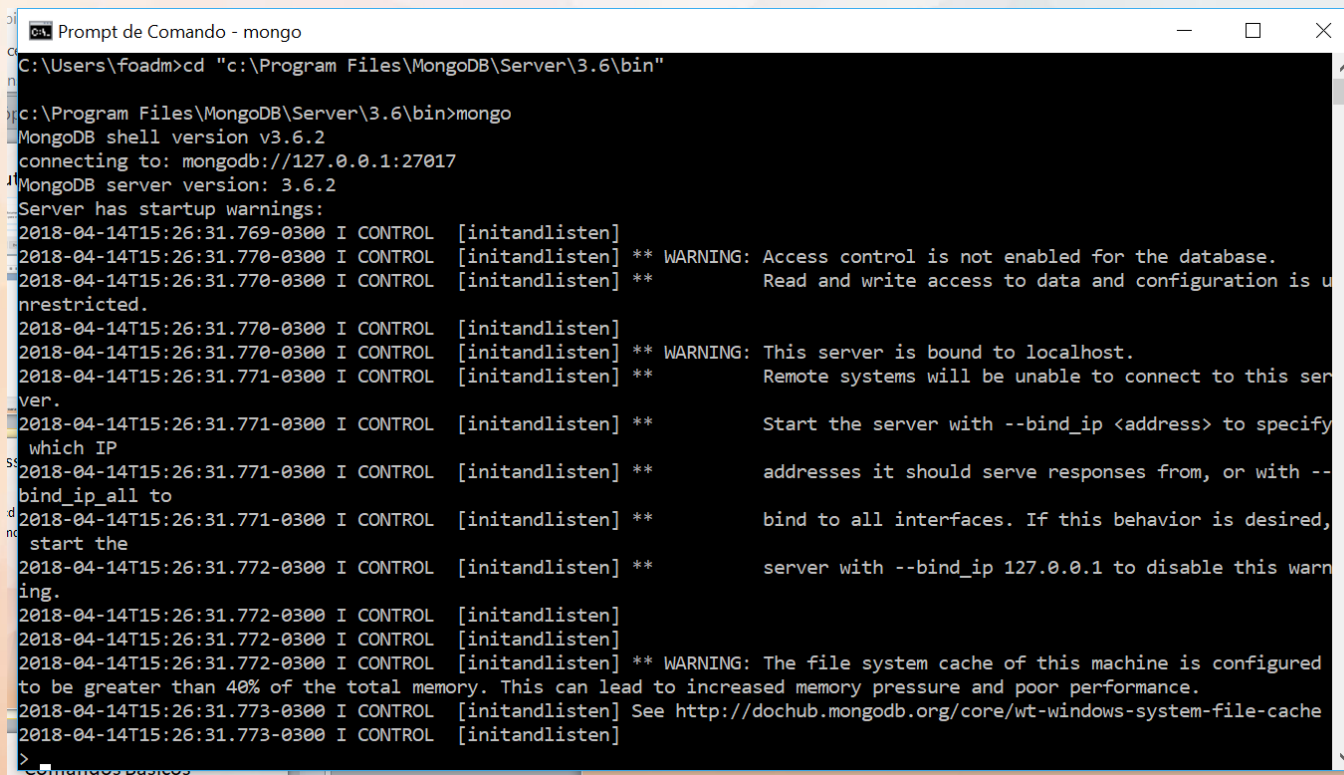


Clique com o botão direito
selecione iniciar.

Isso iniciará o serviço do
banco de dados do mongo
usualmente na porta 27017

Acessando o Mongo via Linha de Comando

- CMD> cd C:\Program Files\MongoDB\Server\3.6\bin
- CMD> mongo



```
Prompt de Comando - mongo
C:\Users\foadm>cd "c:\Program Files\MongoDB\Server\3.6\bin"

c:\Program Files\MongoDB\Server\3.6\bin>mongo
MongoDB shell version v3.6.2
connecting to: mongod://127.0.0.1:27017
MongoDB server version: 3.6.2
Server has startup warnings:
2018-04-14T15:26:31.769-0300 I CONTROL [initandlisten]
2018-04-14T15:26:31.770-0300 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-04-14T15:26:31.770-0300 I CONTROL [initandlisten] **      Read and write access to data and configuration is u
nrestricted.
2018-04-14T15:26:31.770-0300 I CONTROL [initandlisten]
2018-04-14T15:26:31.770-0300 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-04-14T15:26:31.771-0300 I CONTROL [initandlisten] **      Remote systems will be unable to connect to this ser
ver.
2018-04-14T15:26:31.771-0300 I CONTROL [initandlisten] **      Start the server with --bind_ip <address> to specify
which IP
2018-04-14T15:26:31.771-0300 I CONTROL [initandlisten] **      addresses it should serve responses from, or with --
bind_ip_all to
2018-04-14T15:26:31.771-0300 I CONTROL [initandlisten] **      bind to all interfaces. If this behavior is desired,
start the
2018-04-14T15:26:31.772-0300 I CONTROL [initandlisten] **      server with --bind_ip 127.0.0.1 to disable this warn
ing.
2018-04-14T15:26:31.772-0300 I CONTROL [initandlisten]
2018-04-14T15:26:31.772-0300 I CONTROL [initandlisten]
2018-04-14T15:26:31.772-0300 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured
to be greater than 40% of the total memory. This can lead to increased memory pressure and poor performance.
2018-04-14T15:26:31.773-0300 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-04-14T15:26:31.773-0300 I CONTROL [initandlisten]
>
```

Criando um Database

- use [database]
- Comando: use facet
- Não é necessário create.

```
> use facet  
switched to db facet
```

Inserindo um registro numa Collection

- `db.[collection].insert({JSON})`
- Comando:
 - `db.alunos.insert({nome:"João", turma:"Web Mobile"});`
 - `db.alunos.insert({nome:"José", turma:"Eng. Software"});`
 - `db.alunos.insert({nome:"Foad", turma:"Web Mobile"});`
- Não é necessário create.

```
> db.alunos.insert({nome:"Foad", turma:"Web Mobile"});  
WriteResult({ "nInserted" : 1 })
```

Lendo registros de uma Collection

- `db.[collection].find()`
- Comando: `db.alunos.find()`

```
> db.alunos.find();
{ "_id" : ObjectId("5ac911fc71e56080c24aef82"), "nome" : "Foad", "turma" : "Web Mobile" }
{ "_id" : ObjectId("5ac9130f71e56080c24aef83"), "nome" : "José", "turma" : "Eng. Software" }
{ "_id" : ObjectId("5ac9132371e56080c24aef84"), "nome" : "João", "turma" : "Web Mobile" }
```

Fazendo Queries

- `db.[collection].find({parametro: valor})`
- Comando: `db.alunos.find({nome: "José"})`

```
> db.alunos.find({nome:"José"});  
{ "_id" : ObjectId("5ac9130f71e56080c24aef83"), "nome" : "José", "turma" : "Eng. Software" }
```

- Comando: `db.alunos.find({turma: "Web Mobile"})`

```
> db.alunos.find({turma: "Web Mobile"})  
{ "_id" : ObjectId("5ac911fc71e56080c24aef82"), "nome" : "Foad", "turma" : "Web Mobile" }  
{ "_id" : ObjectId("5ac9132371e56080c24aef84"), "nome" : "João", "turma" : "Web Mobile" }
```

- `db.alunos.find({nome:"Jose"}).collation({ locale: "pt", strength: 1 })`

```
> db.alunos.find({nome:"Jose"}).collation( { locale: "pt", strength: 1 } )  
{ "_id" : ObjectId("5ac9130f71e56080c24aef83"), "nome" : "José", "turma" : "Eng. Software" }
```

Outros exemplos de Consulta

- `db.restaurants.find({ "grades.score": { $gt: 30 } })`
- `db.restaurants.find({ "grades.score": { $lt: 10 } })`
- `db.restaurants.find({ "cuisine": "Italian",
"address.zipcode": "10075" })`
- `db.restaurants.find({ $or: [{ "cuisine": "Italian" }, {
"address.zipcode": "10075" }] })`

Usando projeções

- `db.[collection].find([query], [projeção])`
- `db.alunos.find({turma: "Web Mobile"}, {_id:0, nome:1})`

```
> db.alunos.find({turma: "Web Mobile"}, {_id:0, nome:1})
{ "nome" : "Foad" }
{ "nome" : "João" }
```

Alteração de Documentos

- `db.alunos.findAndModify({ query: {nome:"José"}, update: { $inc: {nota: 9.5} }})`

```
{ "_id" : ObjectId("5ac9130f71e56080c24aef83"), "nome" : "José", "turma" : "Eng. Software", "nota" : 9.5 }
```

- `db.alunos.update({nome:"João"}, { $inc: {nota: 8.5} })`

```
{ "_id" : ObjectId("5ac9132371e56080c24aef84"), "nome" : "João", "turma" : "Web Mobile", "nota" : 8.5 }
```


Exclusão de Documentos

- `db.[collection].deleteOne({JSON})`
- `db.[collection].deleteMany({JSON})`
- `db.[collection].remove({JSON})`
- `db.alunos.remove({nome: "Foad"});`

```
> db.alunos.remove({nome: "Foad"});  
WriteResult({ "nRemoved" : 1 })
```

Backend Mongo com Node.js

- Vamos criar um backend usando Node.js
- Instale o Node.js
- Baixe o arquivo do GitHub
 - <https://github.com/foadmkn/RESTfulAPITutorial>
- Extraia em uma pasta
- Entre na pasta com o CMD do Windows
- Digite **npm install**
- Rode usando o comando **node server.js**

Entendendo o nosso server

verifyToken

```
app.route('/tasks')
  .get(verifyToken, todoList.list_all_tasks)
  .post(verifyToken, todoList.create_a_task);

app.route('/tasks/:taskId')
  .get(verifyToken, todoList.read_a_task)
  .put(verifyToken, todoList.update_a_task)
  .delete(verifyToken, todoList.delete_a_task);

var auth = require('../controllers/authController');
app.route('/register').post(auth.register);
app.route('/login').post(auth.login);
```

/tasks

GET – lista todos task

POST – cria novo task

/tasks/[id]

GET – lê uma task

PUT – atualiza uma task

DELETE – exclui uma task

/register

POST – Cria um novo usuário
retornando um token JWT

/login

POST – Faz login retornando um
token JWT

Models

Model User

```
var UserSchema = new Schema({
  username: {
    type: String,
    Required: 'Kindly enter the username'
  },
  created_date: {
    type: Date,
    default: Date.now
  },
  password: {
    type: String,
    Required: 'Kindly enter the password'
  }
});
```

Model Task

```
var TaskSchema = new Schema({
  name: {
    type: String,
    Required: 'Kindly enter the name of the task'
  },
  created_date: {
    type: Date,
    default: Date.now
  },
  status: {
    type: String,
    enum: ['pending', 'ongoing', 'completed'],
    default: 'pending'
  }
});
```

VerifyToken

```
var jwt = require('jsonwebtoken');
var config = require('../config');
function verifyToken(req, res, next) {
  var token = req.headers['x-access-token'];
  if (!token)
    return res.status(403).send({ auth: false, message: 'No token provided.' });
  jwt.verify(token, config.secret, function(err, decoded) {
    if (err)
      return res.status(500).send({ auth: false, message: 'Failed to authenticate token.' });
    // if everything good, save to request for use in other routes
    req.userId = decoded.id;
    next();
  });
}
module.exports = verifyToken;
```

Http Header

Verifica o token

```
exports.login = function(req, res) {
  User.findOne({ username: req.body.username }, function (err, user) {
    if (err) return res.status(500).send('Error on the server.');
```

```
    if (!user) return res.status(404).send('No user found.');
```

```
    var passwordIsValid = bcrypt.compareSync(req.body.password, user.password);
```

```
    if (!passwordIsValid) return res.status(401).send({ auth: false, token: null });
```

```
    var token = jwt.sign({ id: user._id }, config.secret, {
```

```
      expiresIn: 86400 // expires in 24 hours
```

```
    });
```

```
    res.status(200).send({ auth: true, token: token });
```

```
  });
}
```

secret string
somente
conhecido pelo
server

Cria um token

JSON Web Tokens

- JWT - JSON Web Tokens
- Definidos pelo padrão [RFC 7519](#)
- Servem para garantir segurança entre duas parte
- Header

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

- Payload

```
{  
  "sub": "1234567890",  
  "name": "John Doe",  
  "admin": true  
}
```

- Verify Signature

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  base64UrlEncode(secretToken))
```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjE0NjQ5OTY0Mj0.TjVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

Mais informações

- <https://docs.mongodb.com/manual/>
- <https://github.com/generalgmt/RESTfulAPITutorial>
- <https://medium.freecodecamp.org/securing-node-js-restful-apis-with-json-web-tokens-9f811a92bb52>