# MongoDB

## O que é?

    MongoDB é um banco de dados não relacional (NoSQL) do tipo Documento (document-base). Nesse tipo de banco de dados, cada registro é pensado como um documento, e todos os dados desse documento ficam armazenadas junto dele. Ele pode conter dados sem um esquema prévio, em que cada documento possui seu conjunto de chaves e valores independente dos outros documentos.

## Estruturas

### Database

    O database é seu banco de dados, que vai possuir um conjunto de collections.

### Collection

    Uma coleção (collection) é um conjunto de documentos, seria o equivalente a uma tabela em bancos SQL.

### Document

Um documento é como se fosse um objeto em JSON (javascript object notation) que você usa no seu código. Em outros bancos de dados, o documento pode estar em outros formatos também, como XML, mas no MongoDB utiliza somente JSON.

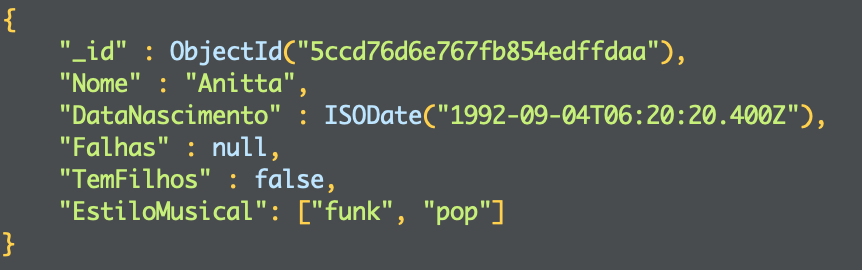
O JSON é uma estrutura de chaves (nome, dataNascimento, etc) e valores, que podem ser dos tipos: **string, number, boolean, null, object, array**.

O MongoDB armazena esses documentos JSON no formato **BSON** (Binary JSON), para ser mais eficiente na codificação e decodificação dos dados. O BSON também fornece alguns alguns tipos a mais, como **Date** e **ObjectId**.

Exemplo de um documento no MongoDB da coleção Diva:



Lembrando que dois documentos dentro da mesma coleção não precisam ter os mesmos campos. Exemplo de outro documento da mesma coleção Diva:



#### ObjectId

Cada documento no MongoDB possui um identificador único, que é gerado automaticamente e armazenado no campo **\_id**. Ele é um **ObjectId**, que contém dentro dele informações do timestamp da criação do documento, assim dá para ordenar seus documentos por esse campo e você não precisa ter um segundo campo como "DataCriação". Mais detalhes: [MongoDB ObjectId](https://www.tutorialspoint.com/mongodb/mongodb_objectid.htm)

# Instalação

Acesse o guia de instalação e siga o passo a passo.

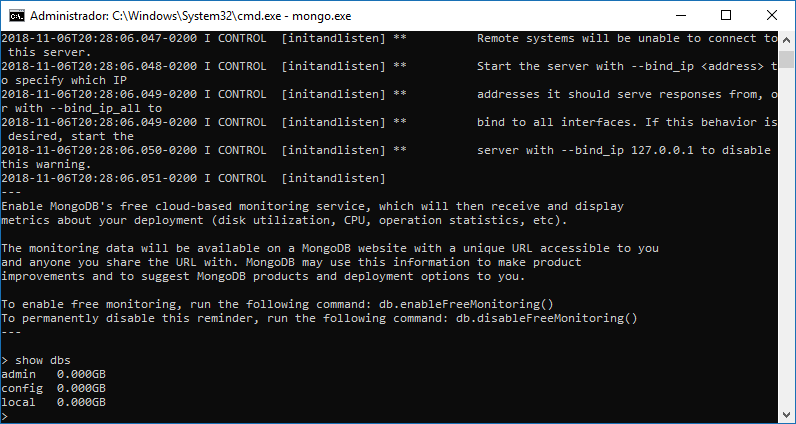
Ir até a pasta de instalação – *‘C:\Program Files\MongoDB\Server\4.0\bin’* e executar **mongod.exe**

O comando **mongod** faz com o que processo do banco de dados inicie.

O comando **mongo** faz com que você se conecte com uma instância específica do mongo.

Abra outro CMD/Prompt de Comando, entre na pasta: ‘C:\Program Files\MongoDB\Server\4.0\bin’, e execute **mongo.exe**

Por padrão, estes são os bancos criados:



## Manipulando o MongoDB pela linha de comando

<https://docs.mongodb.com/manual/crud/>

## Criando databases

**use** nomeDB

**show dbs** – não irá mostrar nenhum pois não tem nenhuma coleção inserida nele.

**show collections**

## Importando dados

**mongoimport** serve para importar dados de um determinado documento (Excel, JSON, CSV) para o banco de dados do mongodb. Abra um novo CMD e execute o comando abaixo:

**mongoimport --db todo -c tasks --type csv --file tasks.csv --headerline**

**mongoimport --db reprograma -c alunas --jsonArray --file alunas.json**

## Comandos Gerais

**show dbs** – mostrar todos os bancos de dados

**show collections** – mostrar todas as coleções

**use reprograma** – colocar o banco de dados em uso

**db.dropDatabase()** – deletar todo o banco de dados

**db.alunas.count()** – contar a quantidade de registros inserida

## Read

**db.alunas.findOne()** – seleciona o primeiro registro

**db.alunas.find()** – seleciona todos os registros da coleção selecionada

**db.alunas.find().pretty()** – busca todos os registros da coleção com uma melhor apresentação

**db.alunas.find().limit(2)** – busca os dois primeiros registros da coleção

#### Filtro

**db.alunas.find({ “nome” : “Helena” }).pretty() –** Realizando filtro. Busca apenas a aluna aonde o nome será igual a Helena. Ele é case sensitive. Isso significa que, caso você digite **Nome** no valor da chave, ele não irá ser encontrado.

Quando quer fazer uma busca encadeada por título, devemos colocar o ponto para identificar qual chave interna estamos buscando.

**db.alunas.find({ "livros.titulo" : "Clean Code A" }).pretty()**

**db.alunas.find({ "livros.leu" : true }).pretty()**

### Operadores

**Cuidado:** Quando são colocadas duas condições pro mesmo campo simultaneamente, ele irá considerar apenas o primeiro filtro da lista - **db.alunas.find({ "nome" : "Helena" }, { "nome" : "Aya" }).pretty()**

#### AND

**db.alunas.find({ "nome" : "Helena", "nasceuEmSp" : true }).pretty()**

**db.alunas.find({ $and: [{ "nome" : "Helena"}, {"ativa" : true }] }).pretty()**

#### OR

**db.alunas.find({ $or: [{ "nome" : "Helena"}, {"nome" : "Aya" }] }).pretty()**

#### IN

**db.alunas.find({ "nome" : { $in : ["Helena", "Aya"] } }).pretty()**

#### LIKE

**db.alunas.find({ "nome" : /h/ }).pretty() –** um nome que contenha um *h* em qualquer parte.

**db.alunas.find({ "nome" : /h/i }).pretty()**

**db.alunas.find({ "nome" : /a$/ }).pretty() –** um nome que termine com a letra *a*

**db.alunas.find({ "nome" : /^H/ }).pretty()** – um nome que comece com a letra *H*

**db.alunas.find({ "nome" : /^h/i }).pretty()**

i – ignoreCase - <https://www.w3schools.com/jsref/jsref_regexp_i.asp>

#### Ordenação

**db.alunas.find().sort({ \_id: 1}).pretty()** - ordena pelo campo **\_id** de forma crescente

**db.alunas.find().sort({ nome: 1}).pretty()**

**db.alunas.find().sort({ \_id: -1}).pretty()** - ordena pelo campo **\_id** de forma decrescente

**db.alunas.find().sort({ nome: -1}).pretty()**

**db.alunas.find().sort({ nome: -1}).limit(1).pretty()** - ordena pelo campo **nome** de forma crescente e pega o primeiro registro

**db.alunas.find().limit(1).pretty() –** seleciona somente o primeiro registro

## Create

**-- inserir apenas um registro na instrução**

db.alunas.insert(

{

"nome": "Sabrina",

"dateOfBirth": "1996-02-12T08:30:00.000Z",

"nasceuEmSp": false,

"livros": [

{"titulo":"MySQL" , "leu" : true},

{"titulo": "MongoDB", "leu" :false},

{"titulo": "Java", "leu":false},

{"titulo": "Clean Architecture", "leu" :true}

]

}

)

**-- inserir vários registros em uma única instrução**

db.alunas.insertMany([

{

"nome": "Helena B",

"dateOfBirth": "1993-03-18T08:30:00.000Z",

"nasceuEmSp": true,

"livros": [

{"titulo":"C#" , "leu" : true},

{"titulo": "JavaScript", "leu" :true},

{"titulo": "Java", "leu":true},

{"titulo": "Clean Architecture", "leu" :true}

]

}

,

{

"nome": "Helena C",

"dateOfBirth": "1993-03-18T08:30:00.000Z",

"nasceuEmSp": false,

"livros": [

{"titulo":"NodeJS" , "leu" : true},

{"titulo": "Java", "leu":true},

{"titulo": "Clean Architecture", "leu" :true}

]

}

])

-- verificar se os registros foram inseridos corretamente.

**db.alunas.find({}).sort({ \_id : -1 }).limit(3).pretty()**

## Update

Caso eu faça isso, sem settar os valores que eu quero no campo, o documento inteiro será atualizado com apenas o campo que foi passado.

db.alunas.update(

{ "nome" : "Helena C" },

{ "nasceuEmSp" : false }

)

**db.alunas.find({}).sort({ \_id : -1 }).limit(1).pretty()**

db.alunas.update(

{ "nome" : "Helena B" },

{

$set : { "nasceuEmSp" : true }

}

)

Os operadores de update:

$set: muda o valor de um campo;

$unset: remove um campo do documento;

$rename: muda o nome de um campo do documento;

$inc: quando um campo numérico precisa ser incrementado (ou decrementado, usando um valor negativo);

$mul: quando um campo numérico precisa ser multiplicado;

## Delete

**db.alunas.remove({ "\_id" : ObjectId("5dab90e10d61146c420264aa") })**

**db.alunas.remove({ "\_id" : ObjectId("5dab90e10d61146c420264ab") })**