



# NoSQL Databases

Curso RE\_START 2020  
2020-v0.0-PT

# Sumário

- › NoSQL
- › MongoDB
- › Setup
- › BSON
- › Shell
- › Demo

# NoSQL

## O que é NoSQL?

- › Os programadores trabalham com aplicações que criam volumes massivos de dados estruturados, semi- estruturados, não estruturados e polimórficos
- › São base de dados sem esquema formal, como as base de dados relacionais
- › Permitem guardar na mesma coleção documentos com esquemas diferentes

## Base de dados tipo NoSQL

- › Document: emparelham cada chave com uma estrutura de dados complexa chamada documento. Ex: MongoDB
- › Graph: guarda informação acerca de redes de dados, tal como conexões sociais. Ex: Neo4J
- › Key-value: são as mais simples bds NoSQL, onde cada item é guardado como um par chave/valor. Ex: Redis
- › Wide-column: são otimizados para consultas em grandes conjuntos de dados e guardam colunas de dados juntas, em vez de linhas. Ex: Cassandra

# NoSQL

## Vantagens

- › Lidar com grandes volumes de dados estruturados, semi-estruturados e não estruturados
- › Iterações ágeis de mudança de esquema
- › Linguagem de programação orientada a objetos mais flexível
- › Arquitetura distribuída e escalável, por oposição a arquiteturas monolíticas como dos SGBDRs
- › 'Auto-sharding' que significa suporte nativo a escalar base de dados horizontal e automaticamente por vários servidores

# MongoDB

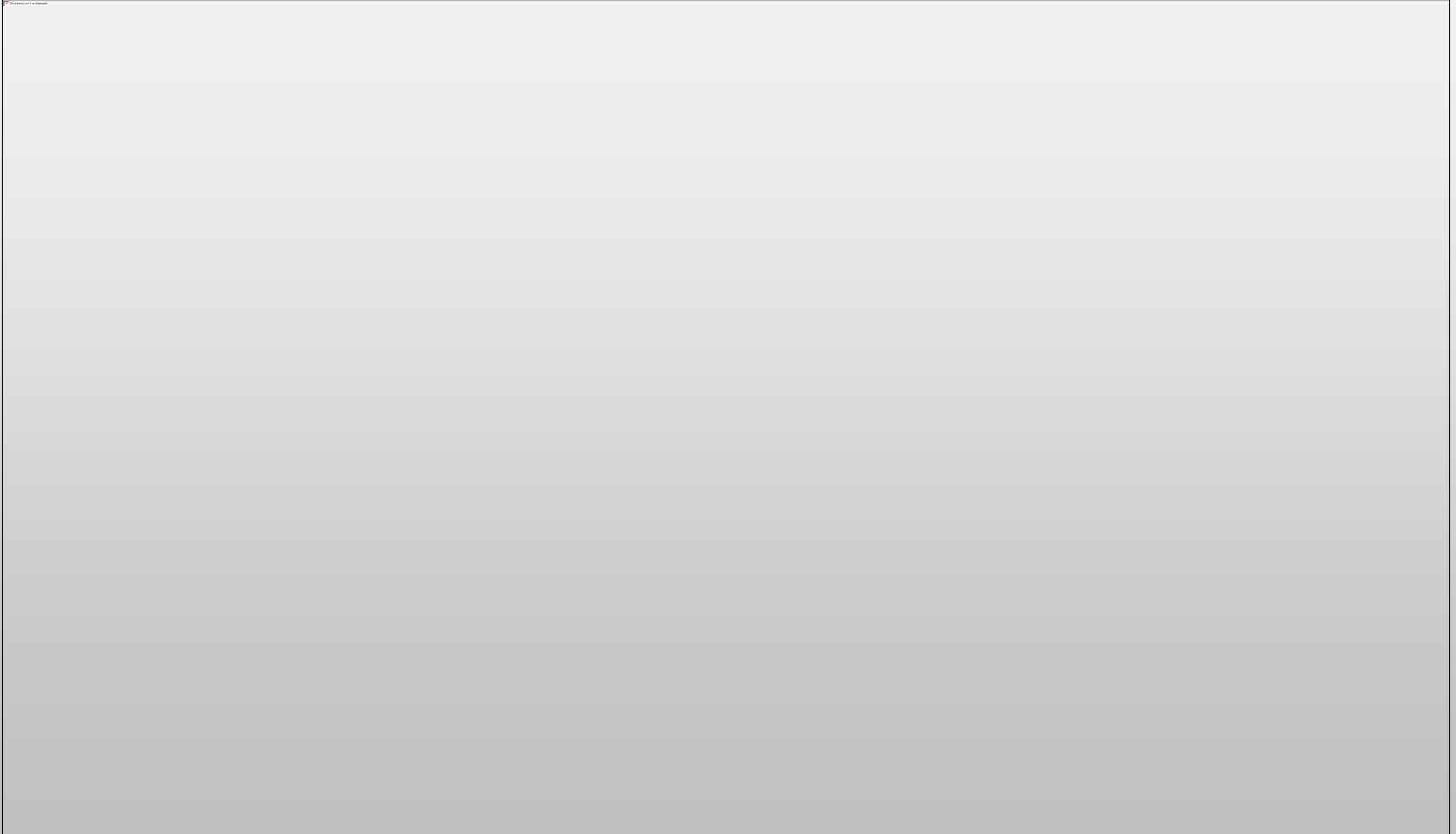
## MongoDB

- › É uma base de dados NoSQL
- › Plataforma livre e de código aberto
- › Programa de base de dados orientado a documentos
- › Usa documentos do tipo JSON (BSON – Binary JSON)

## Estrutura

- › Database > Collection > Document
- › Exemplos:
  - › Compras > Produto > { produto: 'caneta', preco: 5.25 }
  - › Compras > Produto > { produto: 'caneta', tipo: 'tinta permanente', cor: 'preto', preco: 5.25 }

# MongoDB



# Setup

## Download

- › MongoDB Community On-premises 6.0.1 (agosto-2022)
  - › <https://www.mongodb.com/try/download/community>
  - › C:\Program Files\MongoDB\Server\6.0.1\bin
- › Mongo Shell

## Componentes

- › Servidor: mongo.exe
- › Linha de comandos (shell): mongo.exe
- › GUIs:
  - › [MongoDB Compass](#)
  - › [Studio 3T for MongoDB](#)
- › Editor de texto:
  - › [Notepad++](#)
  - › ...

# Setup

## Pasta das base de dados

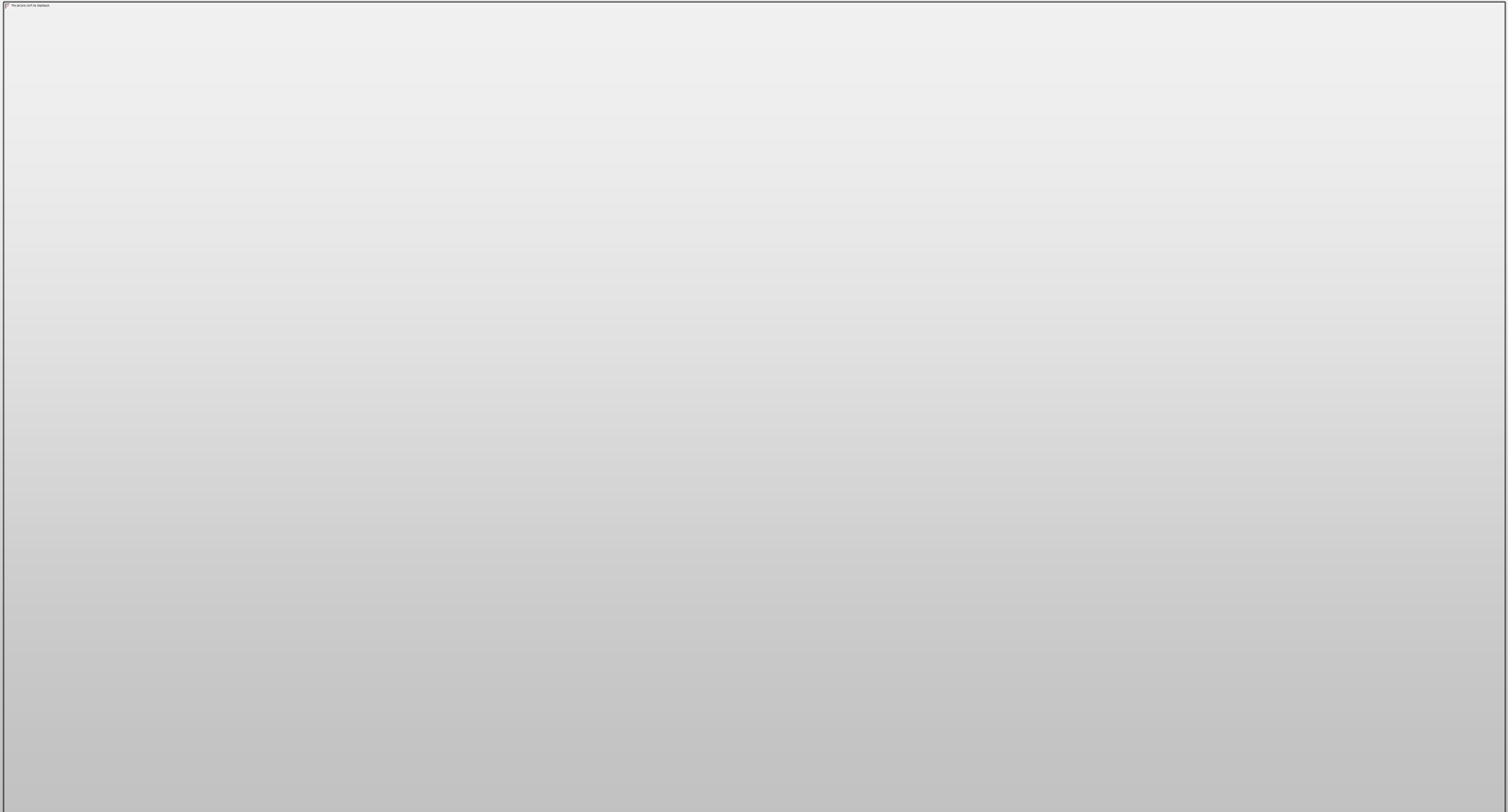
- › Configurar a variável de ambiente Path com o caminho completo para a diretoria bin
  - › "C:\Program Files\MongoDB\Server\6.0\bin"
- › Abrir uma janela "Command Prompt" e digitar o commando "mongod --version"

## Base de dados por omissão

- › admin
- › config
- › local



# MongoDB



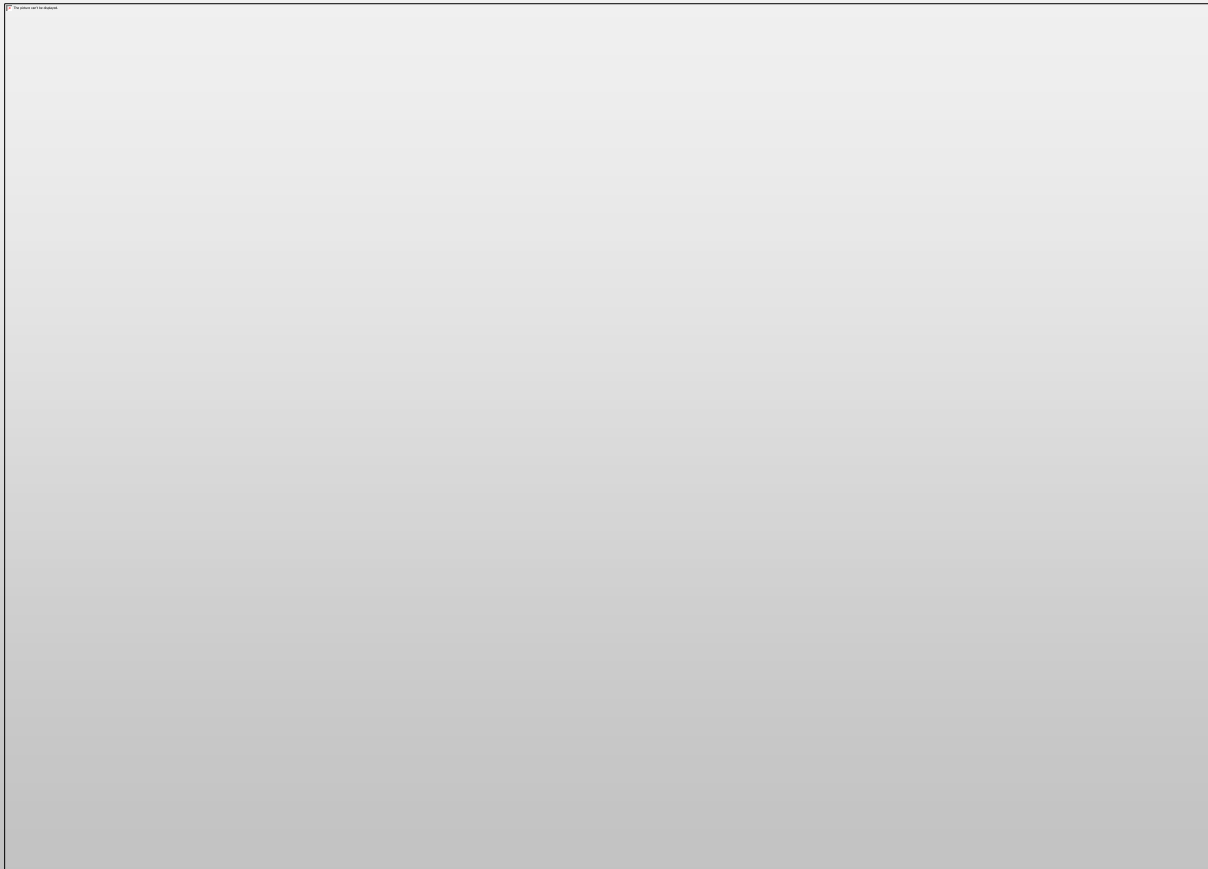
# BSON

## Formato de dados

- › O formato nativo de dados do MongoDB é BSON
- › MongoDB representa documentos JSON num formato binary-encoded chamado BSON



# JSON



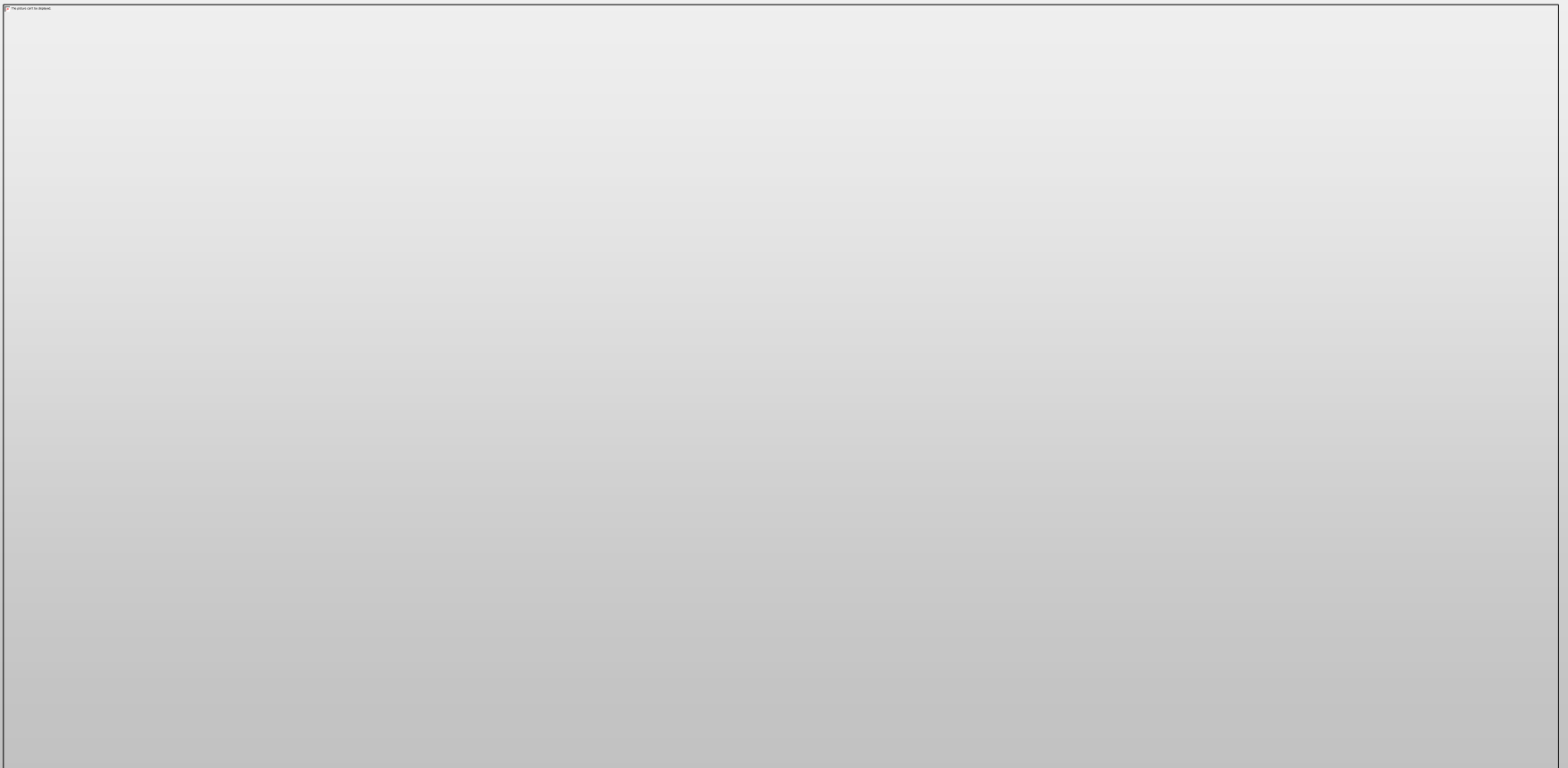
# Shell

## Shell para objetos (base de dados e coleções)

Objetivo	Comando
Mostrar bds	show dbs
Criar/usar bd	use [nome BD] e.g. use pessoas
Apagar bd	db.dropDatabase()
Criar coleção explicitamente	db.createCollection( "pessoas" )
Criar coleção implicitamente	db.people.insertOne({greeting: "Hello"}) A coleção é criada ao inserir um documento na coleção
Listar coleções	db.getCollectionNames()
Apagar coleção	db.pessoas.drop()
Remover bd	db.dropDatabase()

# CRUD

## CRUD Operations

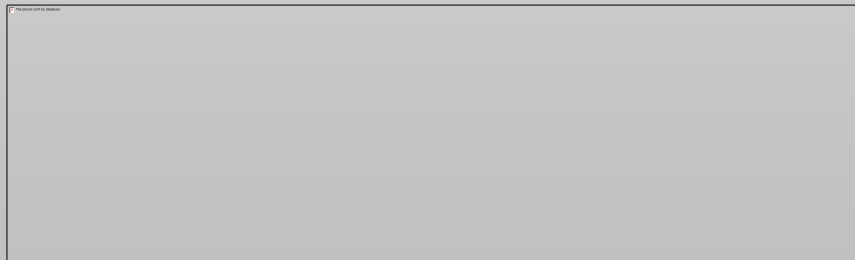
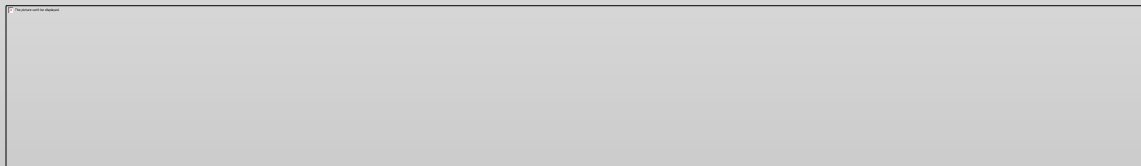


## Create: insertOne

Inserir um documento na coleção

```
db.people.insertOne( {  
  "firstName": "Bastien",  
  "lastName": "Themann",  
  "email": "bthemann0@patch.com",  
  "gender": "Male"  
})
```

Confirmar se o documento foi inserido na coleção



- de onde vem o atributo `_id`?
- para que serve a função `pretty()`

## Create: insertMany

Inserir um documento na coleção

```
db.people.insertMany([
  {
    "firstName": "Bastien",
    "lastName": "Themann",
    "email": "bthemann0@patch.com",
    "gender": "Male"
  },
  {
    "firstName": "Windham",
    "lastName": "Corrington",
    "email": "wcorrington1@examiner.com",
    "gender": "Male"
  }
])
```

## Read: find vs findOne

Selecionar todos os elementos de uma coleção

- `db.people.find()`
- `db.people.find( {} )`

Selecionar documentos cujos nome seja "Bastien"

- `db.people.find( { "firstName": "Bastien" } )`
- Selecionar apenas um documento
- `db.people.findOne()`
- `db.people.findOne( { "firstName": "Bastien" } )`



## Read: deleteOne vs deleteMany

Apagar um único documento de uma coleção

- `db.people.deleteOne( { "firstName": "Bastien" } )`

(apesar de haver 2 coleções com o primeiro nome “Bastien”, apenas uma delas foi apagado)

Apagar todos os documentos de uma coleção que satisfaçam um determinado filtro

- `db.people.deleteMany( { "firstName": "Bastien" } )`

(apaga todos os documentos cujo primeiro nome é “Bastien”)

Apagar todos os documentos de uma coleção

- `db.people.deleteMany( {} )`

# Projeção

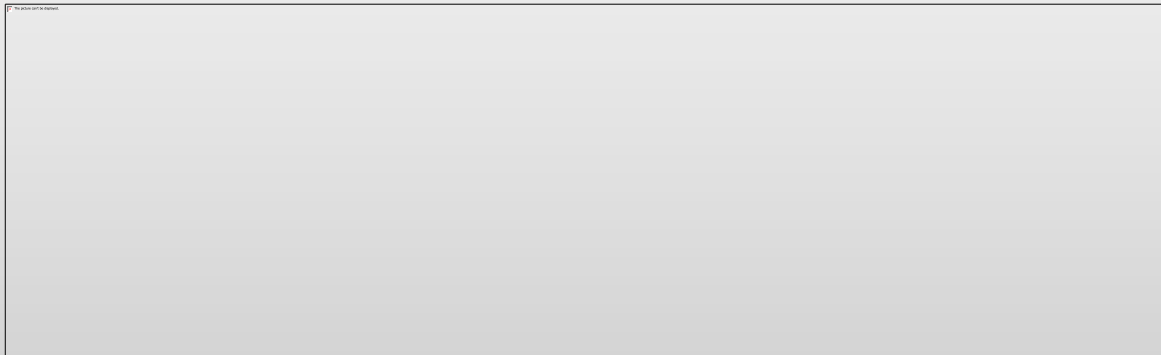
© Primavera Engenharia

## Find: Projeção

Apenas o atributo “firstName”.

- `db.people.find({}, { "firstName": 1 })`

(`_id` é incluído por omissão)



Todos os atributos excepto “firstName”

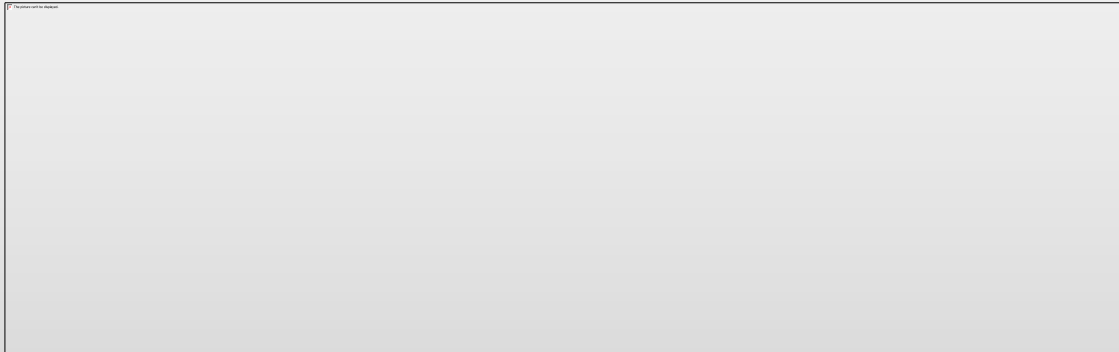
- `db.people.find({}, { "firstName": 0 })`

Como listar apenas o atributo “firstName” sem o `_id`?

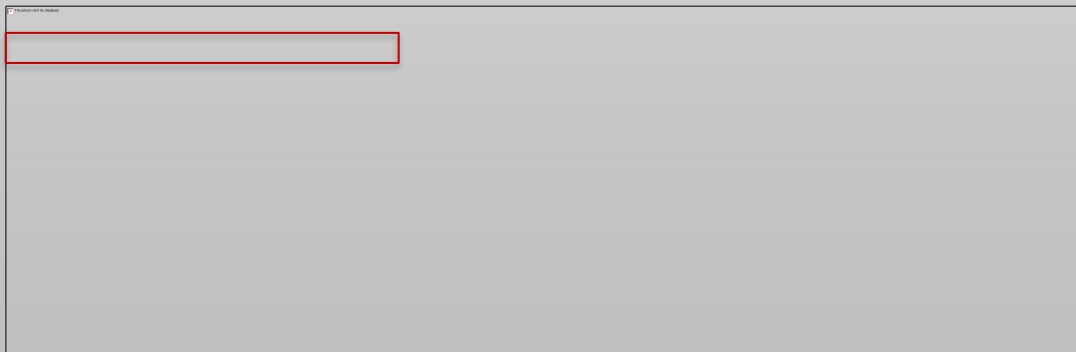
## Find: updateOne

Listar “firstName” da coleção “people”.

```
- db.people.find({}, { "firstName": 1, _id:0 })
```



```
-db.people.updateOne( {"firstName": "Noach"},  
  {$set:{"firstName" : "Noe"}})
```



# Shell

## Shell para documentos (linhas)

Objetivo	Comando
Inserir 1 documento	<code>db.pessoas.insertOne( {nome: "mrs", tipo: "formadora" } )</code>
Inserir vários documentos	<code>db.pessoas.insertMany( [ { nome: "ana", tipo: "fornecedor" }, { nome: "amélia", tipo: "cliente", nif: "123456789" }, { nome: "temp" } ] )</code>
Atualizar 1 documento	<code>db.pessoas.updateOne( { nome: "mrs" }, { \$set: { nome: "milena" } } )</code>
Apagar 1 documento	<code>db.pessoas.deleteOne( { nome: "temp" } )</code>
Encontrar documentos	<code>db.pessoas.find( { nome: "ana" } );</code>
Listar documentos	<code>db.pessoas.find();</code>
Listar documentos formatados	<code>db.pessoas.find().pretty();</code>

### CRUD operations

C → create | insert

R → read | select

U → update | update

D → delete | delete

## Demo

1. Listar as bds
2. Criar e usar a base de dados 'pessoas'
3. Criar a coleção 'pessoas'
4. Listar as bds
5. Listar as coleções
6. Criar, um a um, dois documentos com os atributos 'nome' e 'tipo'
7. Listar os documentos
8. Inserir dois documentos, simultaneamente, com atributos diferentes
9. Encontrar um documento
10. Atualizar um documento
11. Apagar um documento
12. Listar os documentos formatados
13. Apagar a collection
14. Apagar a bd

## Demo - solução

```
1. show dbs
2. use pessoas
3. db.createCollection( "pessoas" )
4. db.getCollectionNames()
5. db.pessoas.insertOne( {nome: "mrs", tipo: "formadora" } )
6. db.pessoas.insertOne( { nome: "a", tipo: "aluna" } )
7. db.pessoas.find()
8. db.pessoas.insertMany( [ { nome: "amélia", tipo: "fornecedora", produto:
    "pcs" }, { nome: "milena", nif: "123456789", tipo: "cliente" } ] )
9. db.pessoas.find( { tipo: "aluna" } )
10. db.pessoas.updateOne( { nome: "mrs"}, { $set: { nome : "temp", tipo:
    "temp" } } )
11. db.pessoas.updateOne( { tipo: "aluna" }, { $set: { tipo : "formanda" } } )
12. db.pessoas.deleteOne( { nome: "temp" } )
13. db.pessoas.find().pretty()
```

