

Exemplos Mongo DB

Edson Guimarães

Criar Banco de Dados DBUnip

- Comando:
 - use DBUnip



```
Select Command Prompt - mongo
MongoDB server version: 4.0.0
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] **           Read and write access to data and configuration is u
nrestricted.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---
> use DBUnip
switched to db DBUnip
>
```

Mostrar os Bancos Existentes

- Comando:
 - show dbs

```
Command Prompt - mongo
Questions? Try the support group
http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---
> use DBUnip
switched to db DBUnip
> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
mbaunip 0.000GB
>
```

Mostrar qual banco estamos conectados

- Comando:
 - db
- Sintaxe:
 - db.<nome da coleção>
- Se a collection não existe

```
Command Prompt - mongo
Server has startup warnings:
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

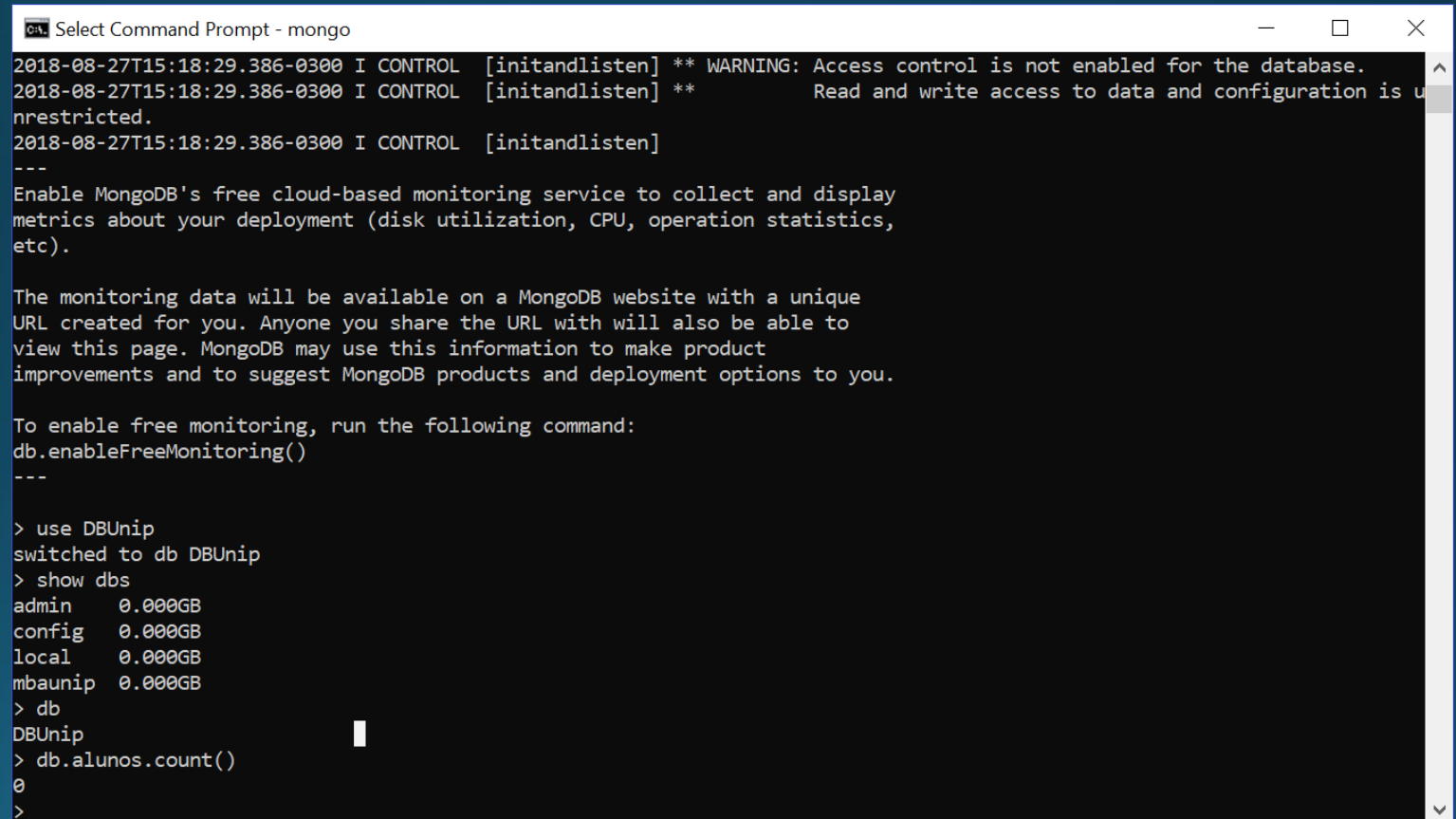
The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> use DBUnip
switched to db DBUnip
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
mbaunip    0.000GB
> db
DBUnip
>
```

Continuação....

- Comando:
- `db.alunos.count()`



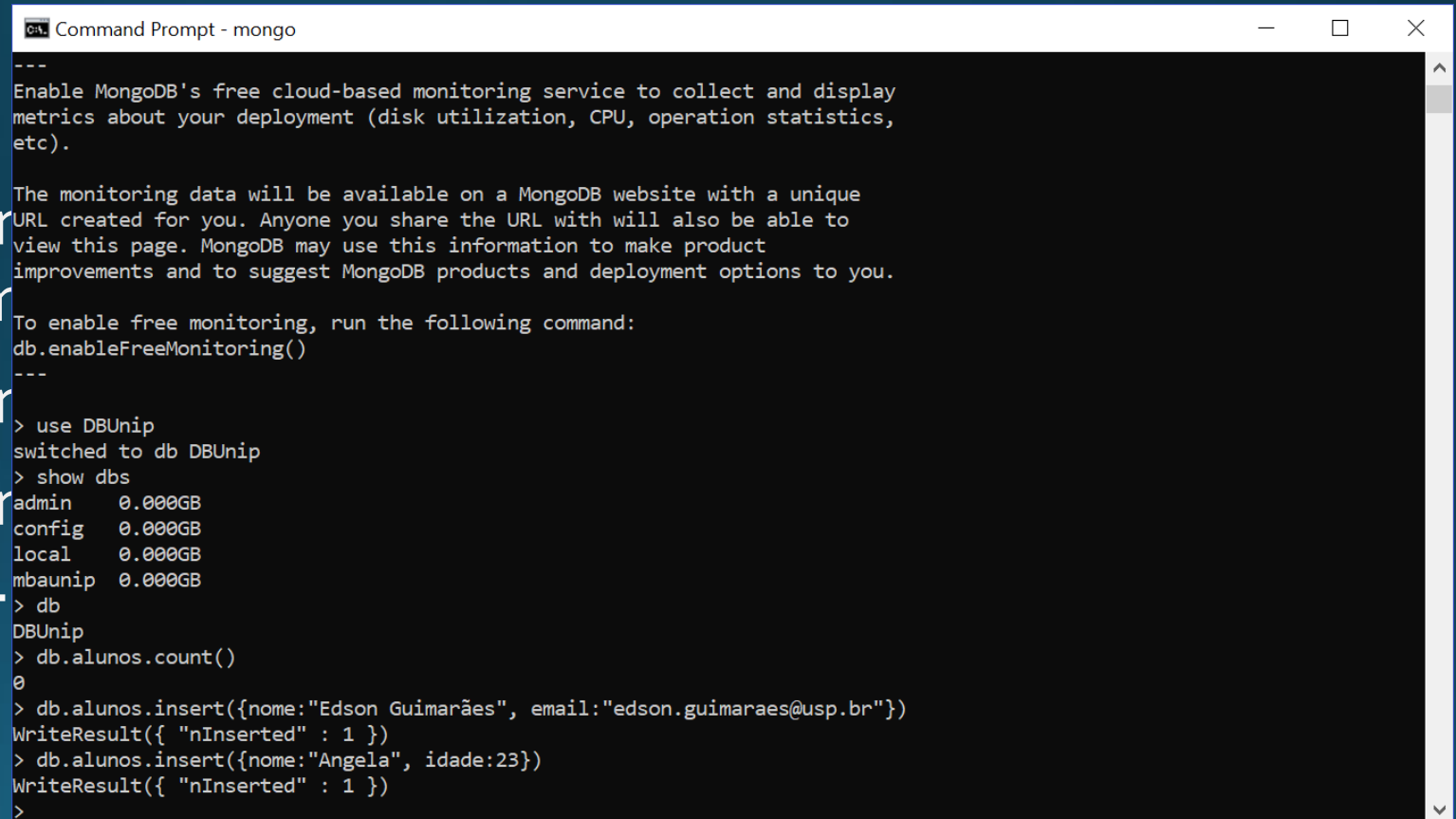
```
Select Command Prompt - mongo
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten] **      Read and write access to data and configuration is u
nrestricted.
2018-08-27T15:18:29.386-0300 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---
> use DBUnip
switched to db DBUnip
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
mbaunip    0.000GB
> db
DBUnip
> db.alunos.count()
0
>
```

Manipulação de Dados

- Comando:
- Inserir dados
- `db.alunos.insert({nome:"Edson Guimarães", email:"edson.guimaraes@usp.br"})`
- `db.alunos.insert({nome:"Angela", idade:23})`
- `db.alunos.insert({nome:"pa@gmail.com", idade:23})`



```
Command Prompt - mongo
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---
> use DBUnip
switched to db DBUnip
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
mbaunip    0.000GB
> db
DBUnip
> db.alunos.count()
0
> db.alunos.insert({nome:"Edson Guimarães", email:"edson.guimaraes@usp.br"})
WriteResult({ "nInserted" : 1 })
> db.alunos.insert({nome:"Angela", idade:23})
WriteResult({ "nInserted" : 1 })
>
```

Manipulação de Dados

- Comando:
- Consulta
- `db.alunos.find()`
- `db.alunos.find().pretty()`

```
Command Prompt - mongo
> db.alunos.count()
0
> db.alunos.insert({nome:"Edson Guimarães", email:"edson.guimaraes@usp.br"})
WriteResult({ "nInserted" : 1 })
> db.alunos.insert({nome:"Angela", idade:23})
WriteResult({ "nInserted" : 1 })
> db.alunos.insert({nome:"Paulo André", idade:30, email:"pa@gmail.com"})
WriteResult({ "nInserted" : 1 })
> db.alunos.find()
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela", "idade" : 23 }
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "idade" : 30, "email" : "pa@gmail.com" }
> db.alunos.find().pretty()
{
  "_id" : ObjectId("5b8940d19948b86a47417bd2"),
  "nome" : "Edson Guimarães",
  "email" : "edson.guimaraes@usp.br"
}
{
  "_id" : ObjectId("5b8941c99948b86a47417bd3"),
  "nome" : "Angela",
  "idade" : 23
}
{
  "_id" : ObjectId("5b8950599948b86a47417bd4"),
  "nome" : "Paulo André",
  "idade" : 30,
  "email" : "pa@gmail.com"
}
>
```

Manipulação de Dados

- Comando:
- Consulta
- `db.alunos.find({nome:"Angela" })`
- `db.alunos.findOne()`
- `db.alunos.find({nome:"Angela", idade:1})`

```
Command Prompt - mongo
> db.alunos.find({nome:"Angela"})
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela", "idade" : 23 }
> db.alunos.find({nome:"Angela"}, {idade:0})
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela" }
> db.alunos.find({nome:"Angela"}, {idade:1})
... ^C
> db.alunos.find({nome:"Angela"}, {idade:1})
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "idade" : 23 }
> db.alunos.find({nome:"Edson Guimarães"})
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
> db.alunos.find()
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela", "idade" : 23 }
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "idade" : 30, "email" : "pa@gmail.com" }
> db.alunos.find({nome:"Paulo André"}, {idade:0})
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "email" : "pa@gmail.com" }
>
```


Manipulação de Dados

- Comando:
- Update
- db.alunos.update()
- db.alunos.update()
- Verificar como está
- db.alunos.update({nome:"Angela Rochetti Alterada"})

```
Select Command Prompt - mongo
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
> db.alunos.find()
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela", "idade" : 23 }
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "idade" : 30, "email" : "pa@gmail.com" }
> db.alunos.find({nome:"Paulo André"}, {idade:0})
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "email" : "pa@gmail.com" }
> db.alunos.update({nome:"Angela"}, {nome:"Angela Rochetti"})
2018-08-31T11:47:54.766-0300 E QUERY [js] SyntaxError: missing } after property list @(shell):1:40
> db.alunos.update({nome:"Angela"}, {nome:"Angela Rochetti"})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.alunos.find()
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela Rochetti" }
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "idade" : 30, "email" : "pa@gmail.com" }
> db.alunos.insert({nome:"Angela Rochetti 2", idade:30})
WriteResult({ "nInserted" : 1 })
> db.alunos.find()
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela Rochetti" }
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "idade" : 30, "email" : "pa@gmail.com" }
{ "_id" : ObjectId("5b8957e39948b86a47417bd5"), "nome" : "Angela Rochetti 2", "idade" : 30 }
> db.alunos.update({nome:"Angela Rochetti 2"}, {$set:{nome:"Angela Rochetti Alterada"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.alunos.find()
{ "_id" : ObjectId("5b8940d19948b86a47417bd2"), "nome" : "Edson Guimarães", "email" : "edson.guimaraes@usp.br" }
{ "_id" : ObjectId("5b8941c99948b86a47417bd3"), "nome" : "Angela Rochetti" }
{ "_id" : ObjectId("5b8950599948b86a47417bd4"), "nome" : "Paulo André", "idade" : 30, "email" : "pa@gmail.com" }
{ "_id" : ObjectId("5b8957e39948b86a47417bd5"), "nome" : "Angela Rochetti Alterada", "idade" : 30 }
>
```

Exercício

- Comando:
- Update
 - Crie um novo registro igual à um existente
 - Atualize este registro (substituir valor de um campo)
 - Verifique o que ocorreu
- Explique Porque isto ocorreu (sua impressão)

Exercício

- Comando:
- Update
 - Insira dois documentos com propriedade nome igual
 - `db.alunos.update({nome:"Nome que Vc inseriu"}, {$set:{nome:"Novo Nome"}}, {multi:true})`
 - Verifique o que ocorreu
- Explique Porque isto ocorreu (sua impressão)

Robomongo

The screenshot displays the Robomongo 3T - 1.2 application window. The interface includes a menu bar (File, View, Options, Window, Help) and a toolbar with icons for connecting to a database, opening a folder, saving, and running queries. The left sidebar shows a tree view of the database structure, including 'Unip (4)' with sub-items like 'System', 'admin', 'local', 'config', and 'mbaunip'. Under 'mbaunip', there is a 'Collections (1)' folder containing the 'alunos' collection. The main workspace shows a query editor with the command `db.getCollection('alunos').find({})` and a results pane displaying a single document from the 'alunos' collection. The results pane includes a table with columns 'Key', 'Value', and 'Type'.

Robo 3T - 1.2

File View Options Window Help

Unip (4)

- System
 - admin
 - local
- config
- mbaunip
 - Collections (1)
 - alunos
 - Functions (0)
 - Users

db.getCollection('alunos').find({})

Unip localhost:27017 mbaunip

```
db.getCollection('alunos').find({})
```

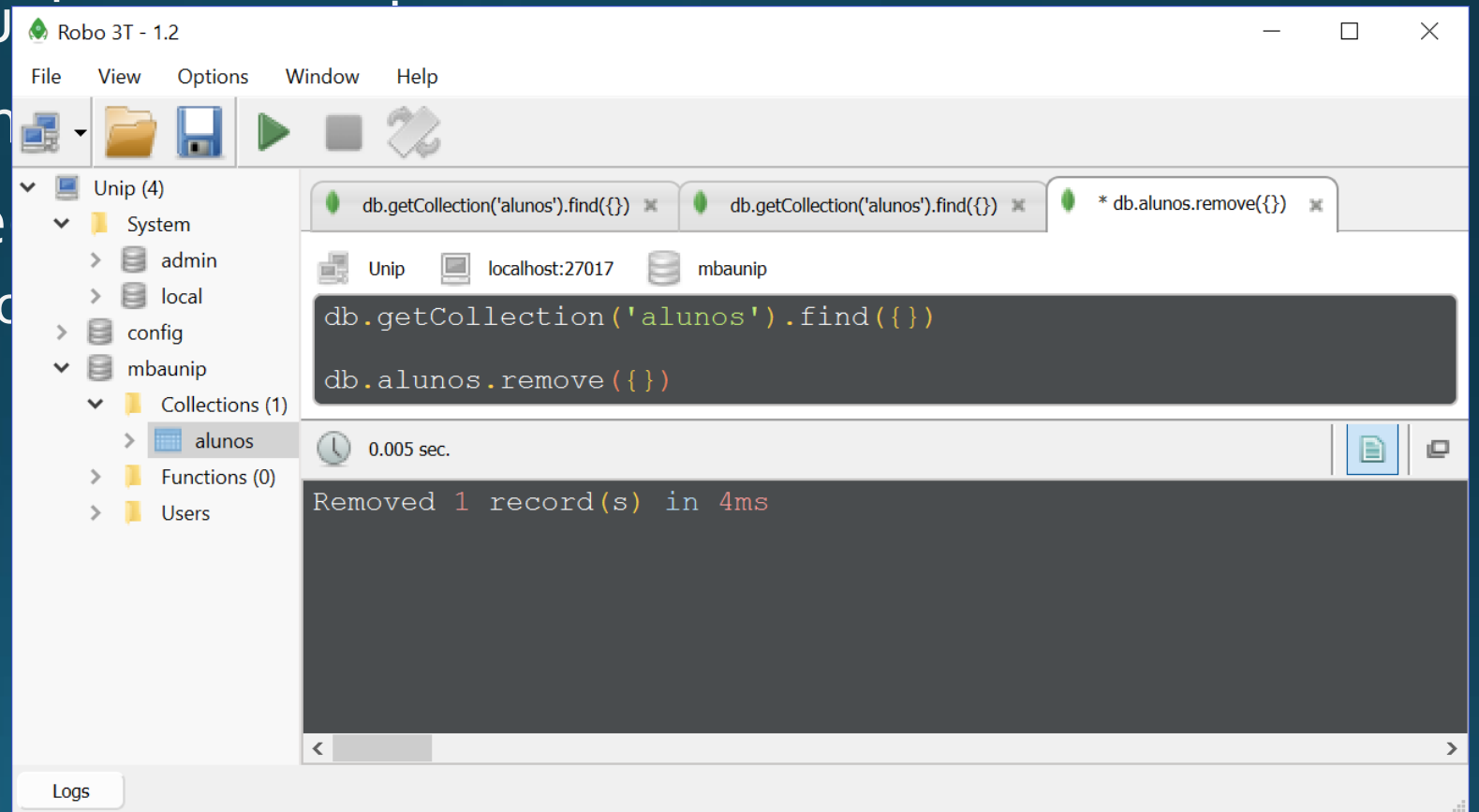
alunos 0.001 sec. 0 50

Key	Value	Type
(1) ObjectId("5b89601119bfba099c3...")	{ 4 fields }	Object
_id	ObjectId("5b89601119bfba099c390187")	ObjectId
nome	Edson	String
idade	30.0	Double
email	edson.guimaraes@usp.br	String

Logs

Robomongo

- Comandos são iguais ao MongoDB
- Comando para remover um documento
- `db.alunos.remove()`
 - Removerá todos os documentos da coleção



Robomongo

- Criar os documentos da tabela abaixo:

Nome	Idade	Cidade	EstadoCivil
José da Silva	20	Bauru	Casado
Antonio José	30	Bauru	Solteiro
Ana Maria	25	Araraquara	Solteira
Maria Clara	30	Ribeirão Preto	Casada

Robomongo

- Consultar todas as pessoas com idade ≥ 25
 - `db.alunos.find({idade:{$gte:25}})` dica -> Case sensitive
- Consultar todas as pessoas com idade > 25
 - `db.alunos.find({idade:{$gt:25}})`
- Consultar todas as pessoas com idade ≤ 25
 - `db.alunos.find({idade:{$lte:25}})`
- Consultar todas as pessoas com idade < 25
 - `db.alunos.find({idade:{$lt:25}})`
- Consultar todas as pessoas com idade $= 30$
 - `db.alunos.find({idade:{$eq:30}})`
 - `db.alunos.find({idade:30})`

Robomongo

- Consultar todas as pessoas que o nome comece com "José"
 - `db.alunos.find({nome:{$regex:/^José/}})`
- Consultar todas as pessoas que possui "José" no nome
 - `db.alunos.find({nome:{$regex:/.*José.*/}})`
- Consultar todas os documentos em que o nome possua números decimais
 - `db.alunos.find({nome:{$regex:/^\d.*$/}})`
- Consultar todas as pessoas com nome igual a "José da Silva" ou idade = 30
 - `db.alunos.find({$or: [{nome:"José da Silva"}, {idade:30}]})`
- Consultar todas as pessoas que possua "José" no nome e idade = 20
 - `db.alunos.find({$and: [{nome:{$regex:/.*José.*/}}, {idade:20}]})`

Robomongo

- Retornar todas as pessoas em ordem crescente de idade
 - `db.getCollection('alunos').find({}).sort({idade:1})`
- Retornar todas as pessoas em ordem decrescente de idade
 - `db.getCollection('alunos').find({}).sort({idade:-1})`
- Retornar apenas os campos “idade” e “estadocivil” das pessoas em ordem decrescente de idade
 - `db.getCollection('alunos').find({}, {idade:1, estadocivil:1}).sort({idade:-1})`
- Retornar apenas os campos “idade” e “estadocivil” das pessoas em ordem decrescente de idade, desde que o campo idade contenha valor
 - `db.getCollection('alunos').find({idade:{$exists:true}}, {idade:1, estadocivil:1}).sort({idade:-1})`
- Limitar a quantidade de documentos recuperadas
 - `db.alunos.find({}).limit(1)`