Sumário

[Cursos 3](#_Toc86667445)

[Comandos 4](#_Toc86667446)

[Métodos de Cursores (Cursor methods) 4](#_Toc86667447)

[Curso M001 5](#_Toc86667448)

[Teorema CAP (Consistency, Availability, Partition) 5](#_Toc86667449)

[Capítulo 1: O que é MongoDB? 5](#_Toc86667450)

[O que é um documento no MongoDB? 5](#_Toc86667451)

[O que é MongoDB Atlas? 5](#_Toc86667452)

[Criação do Cluster Atlas 🡪 MDBU 🡪 M001 🡪 Sandbox 5](#_Toc86667453)

[Visão geral da interface do usuário do Atlas 5](#_Toc86667454)

[Lab – Conectar no Atlas Cluster 5](#_Toc86667455)

[Capítulo 2: Importando, Exportando e Consultando Dados 6](#_Toc86667456)

[Como o MongoDB armazena dados? 6](#_Toc86667457)

[Importando e Exportando Dados 6](#_Toc86667458)

[Data Explorer 7](#_Toc86667459)

[Comando Find() 7](#_Toc86667460)

[Capítulo 3: Criação e Manipulação de Documentos 7](#_Toc86667461)

[Inserindo Novos Documentos – ObjectId 7](#_Toc86667462)

[Inserindo novos documentos – insert() e erros 8](#_Toc86667463)

[Inserindo novos documentos – insert() order 9](#_Toc86667464)

[Atualizando documentos - Data Explorer 9](#_Toc86667465)

[Atualizando documentos – Mongo Shell 9](#_Toc86667466)

[Deletando Documentos e Coleções (Documents and Collections) 10](#_Toc86667467)

[Capítulo 4: Advanced CRUD Operations 11](#_Toc86667468)

[Query Operators – Comparison (Operadores de consulta – comparação) 11](#_Toc86667469)

[Expressive Query Operators – Comparison (Operadores de consulta expressivas) 12](#_Toc86667470)

[Array Operators (Operadores de Vetores) 12](#_Toc86667471)

[Array Operators and Projection (Operadores de Vetores e Projeções) 13](#_Toc86667472)

[Query Arrays and Sub-Documents (Vetores de Consulta e Sub-Documentos) 14](#_Toc86667473)

[Capítulo 5: Aggregation Framework 14](#_Toc86667474)

[$group | $sum 14](#_Toc86667475)

[sort() | limit() 15](#_Toc86667476)

[Introdution to Indexes 16](#_Toc86667477)

[Introdution to Data Modeling (Introdução à Modelagem de Dados) 16](#_Toc86667478)

[Upsert (Update or Insert) 16](#_Toc86667479)

[Capítulo 6: Próximos Passos 17](#_Toc86667480)

[Atlas Features - More Data Explorer 17](#_Toc86667481)

[Atlas Products and Options 17](#_Toc86667482)

[What is MongoDB Compass? 17](#_Toc86667483)

[What is MongoDB Compass? (part 2) 17](#_Toc86667484)

[Why MongoDB? (optional) 19](#_Toc86667485)

[Curso M220P 19](#_Toc86667486)

[Capítulo 0 – Introduction and Setup 19](#_Toc86667487)

[Setting Up Atlas 19](#_Toc86667488)

[Visão Global 20](#_Toc86667489)

[Visão geral 20](#_Toc86667490)

[Camada de banco de dados 20](#_Toc86667491)

[Setting Up mflix 20](#_Toc86667492)

[Executando o aplicativo 20](#_Toc86667493)

[Executando os testes de unidade 20](#_Toc86667494)

[Capítulo 1 – Driver Setup 21](#_Toc86667495)

[MongoClient 21](#_Toc86667496)

# Cursos

[Minicurso gratuito Python - Hashtag Programação](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqa2ZOMGhqd0dMcXJwY29EanFweFIwVWV5bFhXZ3xBQ3Jtc0trN1kzR1JrYWc4NmNLb3V4a1QzYzIyeTM3NmQ1SFE0aGgwbHRnWGNRUzZLcC1nc2pCbXlKb0l5Y0oyRTVISmVEeEdnd2Vna3hkakpvTkk5Ykdza0dLRkV2eDhvV2l5Nm9YdWUySUpCaVlMaTNEX2ZSOA&q=http%3A%2F%2Fpages.hashtagtreinamentos.com%2Finscricao-minicurso-python-automacao-org%3Forigemurl%3Dhashtag_yt_org_minipython__eK0z5QbpKA)

MongoDB = mdvirtua@19

Link para o compass: mongodb+srv://m001-student:<password>@sandbox.qgari.mongodb.net/test

# Comandos

* show dbs 🡪 Lista os databases
* use <database> 🡪 Abre um database
* show collections 🡪 Lista as coleções dentro de um database
* db.<collection>.findOne() 🡪 Retorna somente o primeiro documento

usar somente quando se tem uma chave única (\_id)

* db.<collection>.find() 🡪 Retorna todos os documentos.

db.<collection>find( {<query>} , {<projections>} )

* db.<collection>.find().count() 🡪 Retorna a quantidade de documentos.
* db.<collection>.insert( json ) 🡪 Insere um documento
* db.<collection>.updateMany( json ) 🡪 Atualiza documentos

db.endereco.updateMany({"cidade":"São José dos Campos",{"$set":{"uf":"SP"}}})

* db.<collection>.updateOne( json ) 🡪 Atualiza o primeiro documento que encontra

usar somente quando se tem uma chave única (\_id)

* db.<collection>.deleteOne({"\_id":123456}) 🡪 Exclui o primeiro documento que encontra

usar somente quando se tem uma chave única (\_id)

* db.<collection>.deleteMany({"test": 3}) 🡪 Exclui todos os documentos na condição
* db.<collection>.drop() 🡪 Exclui a coleção
* $inc
* $set
* $unset
* $eq 🡪 Equal to
* $ne 🡪 Not Equal to
* $gt 🡪 Greater than
* $gte 🡪 Greater than or equal to
* $lt 🡪 Less than
* $gte 🡪 Less than or equal to
* $and
* $or
* $nor
* $not
* $expr
* $all 🡪 Para verificar se todos os elementos do vetor estão no vet do doc.
* $size 🡪 Para limitar o tamanho do vetor
* $elemMatch 🡪 Pesquisa em elementos de um objeto
* $regex 🡪 Contém o elemento texto.
* db.aggregate() 🡪 Função de agregação, similar ao find()
* db.aggregate() 🡪 Função de agregação, similar ao find()
* $group
* $sum

# Métodos de Cursores (Cursor methods)

São aplicados aos resultados (resultsets) dos cursores e não aos dados.

* pretty()
* count()
* sort()
* limit()

**cursor = movies.find( { "cast": "Salma Hayek" } )**

**from bson.json\_util import dumps**

**print(dumps(cursor, indent=2))**

# Curso M001

## Teorema CAP (Consistency, Availability, Partition)

Partição | Consistência | Disponibilidade 🡪 Não é possível termos as três coisas em um único instante no mundo distribuído.

## Capítulo 1: O que é MongoDB?

### **O que é um documento no MongoDB?**

Documento - uma maneira de organizar e armazenar dados como um conjunto de pares de valores de campo.

Campo - um identificador exclusivo para um ponto de dados.

Valor - dados relacionados a um determinado identificador.

Coleção - um armazenamento organizado de documentos no MongoDB, geralmente com campos comuns entre os documentos. Pode haver muitas coleções por banco de dados e muitos documentos por coleção.

### O que é MongoDB Atlas?

O Atlas é um banco de dados na nuvem totalmente gerenciado, construído para uma ampla gama de aplicativos com o MongoDB.

O Atlas gerencia os detalhes da criação de clusters para você, o que simplifica a sobrecarga operacional de execução. Isso inclui a implantação em provedores de nuvem e regiões de sua escolha.

O Atlas também é uma ótima maneira de experimentar as novas ferramentas e recursos do MongoDB, e é por isso que o escolhemos para este curso. Possui diferentes níveis de preços, dependendo do tamanho e do uso de seu banco de dados. Neste curso usaremos o nível gratuito do Atlas que permite implantar um cluster gratuito. Um cluster de nível gratuito é um conjunto de réplicas de três servidores com **512 megabytes** de armazenamento. Os clusters de nível gratuito **nunca expiram**. No nível gratuito você terá acesso um subconjunto de recursos do Atlas como **Gráficos** e **Região**.

Conjunto de réplicas - algumas máquinas (instâncias) conectadas que armazenam os mesmos dados para garantir que, se algo acontecer a uma das máquinas, os dados permanecerão intactos. Vem da palavra replicar - copiar algo.

Instância - uma única máquina localmente ou na nuvem, rodando um determinado software, no nosso caso é o banco de dados MongoDB.

Cluster - grupo de servidores que armazenam seus dados.

### Criação do Cluster Atlas 🡪 MDBU 🡪 M001 🡪 Sandbox

* Nome de usuário: **m001-student**
* Senha: **m001-mongodb-basics**

### Visão geral da interface do usuário do Atlas

O painel de navegação superior permite acessar serviços.

O painel de navegação lateral esquerdo se relaciona com o gerenciamento do cluster.

No curso vamos interagir com o cluster de duas maneiras diferentes

* DataExplorer, no qual você pode navegar clicando no botão <Collections>
* Shell

Usar o ip-address permite acesso de qualquer lugar.

### Lab – Conectar no Atlas Cluster

Usar o comando abaixo:

mongo "mongodb+srv://sandbox.qgari.mongodb.net/mvpDB" --username m001-student

ou

mongo "mongodb+srv://sandbox.qgari.mongodb.net/mvpDB" --username m001-student –password m001-mongodb-basics

## Capítulo 2: Importando, Exportando e Consultando Dados

### Como o MongoDB armazena dados?

O MongoDB armazena dados no formato BSON internamente e pela rede.

JSON 🡪 JavaScript Object Notation (Notação de Objeto JavaScript)

* Amigável
* Legível
* Baseado em texto que tem leitura lenta
* Pouco eficiente em termos de espaço (preocupação do BD)
* Suporte apenas um número limitado de tipos de dados básicos

BSON 🡪 Binary JSON

* Otimizado para velocidade
* Otimizado para espaço
* Otimizado para flexibilidade
* Alto desempenho
* Foco em propósitos gerais
* Aceita tipos data
* Aceita tipos binários
* Permite inteiros de tamanhos diferentes e numéricos com níveis de precisão (float, dougle, decimal128)
* Permite que comparações e cálculos ocorram diretamente com os dados numéricos.

<https://www.mongodb.com/json-and-bson>

### Importando e Exportando Dados

Para saber mais sobre outros formatos suportados pelo mongoimport , [verifique esta página de documentação](https://docs.mongodb.com/manual/reference/program/mongoimport/#compatibility).

Sequência de conexão SRV - um formato específico usado para estabelecer uma conexão entre seu aplicativo e uma instância do MongoDB. [Clique aqui para saber mais](https://docs.mongodb.com/manual/reference/connection-string/#connections-dns-seedlist).

Código usado nesta palestra:

**BSON**

mongodump --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample\_supplies"

**JSON**

mongoexport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample\_supplies" --collection=sales --out=sales.json

**BSON**

mongorestore --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample\_supplies" --drop dump

**JSON**

mongoimport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample\_supplies" --drop sales.json

Outros formatos suportados [ver documentação](https://docs.mongodb.com/manual/reference/program/mongoimport/#compatibility).

*Sequência de conexão SRV* - um formato específico usado para estabelecer uma conexão entre seu aplicativo e uma instância do MongoDB. [Clique aqui para saber mais](https://docs.mongodb.com/manual/reference/connection-string/#connections-dns-seedlist).

### Data Explorer

*Namespace* - A concatenação do nome do banco de dados e do nome da coleção é chamada de namespace.

Vimos a coleção sample\_training.zips e emitimos as seguintes consultas:

* {"estado": "NY"}
* {"estado": "NY", "cidade": "ALBANY"}

### Comando Find()

Nesta lição, usamos os seguintes comandos:

Conecte-se ao cluster Atlas:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

Sintaxe utilizada:

mongo "mongodb+srv://m001-student:m001-mongodb-basics@sandbox.qgari.mongodb.net/admin"

show dbs

use sample\_training

show collections

db.zips.find({"state": "NY"})

**it** iterates through the cursor.

db.zips.find({"state": "NY"}).count()

db.zips.find({"state": "NY", "city": "ALBANY"})

db.zips.find({"state": "NY", "city": "ALBANY"}).pretty()

## Capítulo 3: Criação e Manipulação de Documentos

### Inserindo Novos Documentos – ObjectId

**Leitura Opcional**

Se você quiser ler mais sobre o tipo de dados ObjectId e a função ObjectId() , que gera valores ObjectId, [verifique esta excelente página de documentação](https://docs.mongodb.com/manual/reference/method/ObjectId/index.html#objectid).

### Inserindo novos documentos – insert() e erros

Para saber mais sobre a [Validação de Esquema, verifique nossa excelente página de documentação sobre o assunto](https://docs.mongodb.com/manual/core/schema-validation/index.html#schema-validation) .

Nesta lição, usamos os seguintes comandos:

mongoimport --uri="mongodb+srv://<username>:<password>@<cluster>.mongodb.net/sample\_supplies" sales.json

**Comando utilizado:**

**mongoimport --uri="mongodb+srv://m001-student:m001-mongodb-basics@admin.mongodb.net/sample\_supplies" sales.json**

**Etapa um** : conectar-se ao cluster Atlas

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

**Etapa dois** : navegue até o banco de dados de que precisamos:

use sample\_training

**Etapa três** , obtenha um documento aleatório da coleção:

db.inspections.findOne();

**Etapa quatro** , copie este documento aleatório e tente inseri-lo na coleção:

db.inspections.insert({

"\_id" : ObjectId("56d61033a378eccde8a8354f"),

"id" : "10021-2015-ENFO",

"certificate\_number" : 9278806,

"business\_name" : "ATLIXCO DELI GROCERY INC.",

"date" : "Feb 20 2015",

"result" : "No Violation Issued",

"sector" : "Cigarette Retail Dealer - 127",

"address" : {

"city" : "RIDGEWOOD",

"zip" : 11385,

"street" : "MENAHAN ST",

"number" : 1712

}

})

db.inspections.insert({

"id" : "10021-2015-ENFO",

"certificate\_number" : 9278806,

"business\_name" : "ATLIXCO DELI GROCERY INC.",

"date" : "Feb 20 2015",

"result" : "No Violation Issued",

"sector" : "Cigarette Retail Dealer - 127",

"address" : {

"city" : "RIDGEWOOD",

"zip" : 11385,

"street" : "MENAHAN ST",

"number" : 1712

}

})

db.inspections.find({"id" : "10021-2015-ENFO", "certificate\_number" : 9278806}).pretty()

### Inserindo novos documentos – insert() order

Nesta lição, usamos os seguintes comandos:

Insert three test documents:

db.inspections.insert([ { "test": 1 }, { "test": 2 }, { "test": 3 } ])

Insert three test documents but specify the **\_id** values:

db.inspections.insert([{ "\_id": 1, "test": 1 },{ "\_id": 1, "test": 2 },

{ "\_id": 3, "test": 3 }])

Find the documents with **\_id**: **1**

db.inspections.find({ "\_id": 1 })

Insert multiple documents specifying the **\_id** values, and using the "**ordered**": **false** option.

db.inspections.insert([{ "\_id": 1, "test": 1 },{ "\_id": 1, "test": 2 },

{ "\_id": 3, "test": 3 }],{ "ordered": false })

Insert multiple documents with **\_id**: 1 with the default "**ordered**": **true** setting

db.inspection.insert([{ "\_id": 1, "test": 1 },{ "\_id": 3, "test": 3 }])

View **collections** in the active **db**

show collections

Switch the active **db** to **training**

use training

View all available databases

show dbs

### *Atualizando documentos - Data Explorer*

### *Atualizando documentos – Mongo Shell*

Para saber mais sobre todos os [operadores de atualização](https://docs.mongodb.com/manual/reference/operator/update/#id1) disponíveis [em MQL, visite nossa excelente página de documentação](https://docs.mongodb.com/manual/reference/operator/update/#id1) .

Comandos usados ​​nesta lição:

*Connect to your Atlas Cluster.*

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

*Use the* sample\_training *database as your database in the following commands.*

use sample\_training

*Find all documents in the* zips *collection where the* zip *field is equal to* "12434".

db.zips.find({ "zip": "12534" }).pretty()

*Find all documents in the* zips *collection where the* city *field is equal to* "HUDSON".

db.zips.find({ "city": "HUDSON" }).pretty()

*Find how many documents in the* zips *collection have the* city *field equal to* "HUDSON".

db.zips.find({ "city": "HUDSON" }).count()

*Update all documents in the* zips *collection where the* city *field is equal to* "HUDSON" *by adding* 10 *to the current value of the* "pop" *field.*

db.zips.updateMany({ "city": "HUDSON" }, { "$inc": { "pop": 10 } })

*Update a single document in the* zips *collection where the* zip *field is equal to* "12534" *by setting the value of the* "pop" *field to* 17630.

db.zips.updateOne({ "zip": "12534" }, { "$set": { "pop": 17630 } })

*Update a single document in the* zips *collection where the* zip *field is equal to* "12534" *by setting the value of the* "population" *field to* 17630.

db.zips.updateOne({ "zip": "12534" }, { "$set": { "population": 17630 } })

*Find all documents in the* grades *collection where the* student\_id *field is* 151 *, and the* class\_id *field is* 339.

db.grades.find({ "student\_id": 151, "class\_id": 339 }).pretty()

*Find all documents in the* grades *collection where the* student\_id *field is* 250 *, and the* class\_id *field is* 339.

db.grades.find({ "student\_id": 250, "class\_id": 339 }).pretty()

*Update one document in the* grades *collection where the* student\_id *is ``250`` \*, and the* class\_id *field is* 339 *, by adding a document element to the* "scores" *array.*

db.grades.updateOne({ "student\_id": 250, "class\_id": 339 },

{ "$push": { "scores": { "type": "extra credit",

"score": 100 }

}

})

### Deletando Documentos e Coleções (Documents and Collections)

Para deletar uma collection 🡪 db.<collection>.drop()

Para deletar um documento 🡪 db.<collection>.deleteOne() ou deleteMany()

*Commands used in this lesson:*

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

use sample\_training

*Look at all the docs that have* test *field equal to* 1.

db.inspections.find({ "test": 1 }).pretty()

*Look at all the docs that have* test *field equal to* 3.

db.inspections.find({ "test": 3 }).pretty()

*Delete all the documents that have* test *field equal to* 1.

db.inspections.deleteMany({ "test": 1 })

*Delete one document that has* test *field equal to* 3.

db.inspections.deleteOne({ "test": 3 })

*Inspect what is left of the* inspection *collection.*

db.inspection.find().pretty()

*View what collections are present in the* sample\_training *collection.*

show collections

*Drop the* inspection *collection.*

db.inspection.drop()

## Capítulo 4: Advanced CRUD Operations

### Query Operators – Comparison (Operadores de consulta – comparação)

#### Update Operators (Operadores de Atualização)

Exemplo: **$inc**, **$set**, **$unset**

Enable us to modify data in the database

#### Query Operators (Operadores de Consulta)

Provide additional ways to locate data within the database

#### $ has multiple uses

Precedes MQL operators

Precedes Aggregation pipeline stages

Allows Access to Field Values

#### Comparison Operators (Operadores de Comparação)

**$eq** = **EQ**ual to

**$gt** > **G**reater **T**han

**$gte** >= **G**reater **T**han or **E**qual to

**$ne** = **N**ot **E**qual to

**$lt** < **L**ess **T**han

**$lte** <= **L**ess **T**han or **E**qual to

**{ <field>: { <operator>: <value> } }**

Implicit **$and**

Find which student ids are > 25 and < 100 in the sample training. Grades collection.

{“$and”: [{“sdudent\_id”: {“$gt”: 25}},{“sdudent\_id”: {“$lt”: 100}}]}

**Better**

{“student\_id”: {“$gt”:25, “$lt”: 100}}

[Dados históricos do Citi Bike](https://www.citibikenyc.com/system-data)

Esta lição usou os seguintes comandos:

Conecte-se ao cluster Atlas:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

*Switch to this database:*

use sample\_training

*Find all documents where the* tripduration *was less than or equal to* 70 *seconds and the* usertype *was not* Subscriber:

db.trips.find({ "tripduration": { "$lte" : 70 },

"usertype": { "$ne": "Subscriber" } }).pretty()

*Find all documents where the* tripduration *was less than or equal to* 70 *seconds and the* usertype *was* Customer *using a redundant equality operator:*

db.trips.find({ "tripduration": { "$lte" : 70 },

"usertype": { "$eq": "Customer" }}).pretty()

*Find all documents where the* tripduration *was less than or equal to* 70 *seconds and the* usertype *was* Customer *using the implicit equality operator:*

db.trips.find({ "tripduration": { "$lte" : 70 },

"usertype": "Customer" }).pretty()

#### Logic Operators (Operadores Lógicos)

**$and** Match **all** of the specified query clauses

**$or** **At least one** of the query clauses is matched

**$nor** **Fail to match** both given clauses (Retorna todos os que não correspondem a ambas as cláusulas)

**{<operator>: [{statment1},{statement2},...]}**

**$not Negates** the query requirement

**{$not: {statment}}**

[Logical Query Operators Documentation](https://docs.mongodb.com/manual/reference/operator/query-logical/)

This lesson used the following commands:

Connect to the Atlas cluster:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

Switch to this database:

use sample\_training

Find all documents where airplanes CR2 or A81 left or landed in the KZN airport:

db.routes.find({ "$and": [ { "$or" :[ { "dst\_airport": "KZN" },

{ "src\_airport": "KZN" }

] },

{ "$or" :[ { "airplane": "CR2" },

{ "airplane": "A81" } ] }

]}).pretty()

### Expressive Query Operators – Comparison (Operadores de consulta expressivas)

#### Compara campos de mesmo tipo.

#### $expr allows the use of aggregation expressions within the query language

{ $expr: { <expression> } }

**$expr** allow us to use variables and conditional statements

$ denotes the use of na operator

$ addresses the field value

This lesson used the following commands:

*Connect to the Atlas cluster:*

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

*Switch to this database:*

use sample\_training

*Find all documents where the trip started and ended at the same station:*

db.trips.find({ "$expr": { "$eq": [ "$end station id", "$start station id"] }

}).count()

*Find all documents where the trip lasted longer than* 1200 *seconds, and started and ended at the same station:*

db.trips.find({ "$expr": { "$and": [ { "$gt": [ "$tripduration", 1200 ]},

{ "$eq": [ "$end station id", "$start station id" ]}

]}}).count()

### Array Operators (Operadores de Vetores)

**$push** nos permite adicionar um elemento em um array.

**$push** transforma um campo em um campo de matriz se ele era de um tipo diferente.

**$all** para verificar todos os elementos do vetor independente da ordem no documento.

.find( {"$all": ["texto1","texto2","texto3"]} )

**$size** restringir o tamanho do vetor do documento.

{ <array field>: { "$size": <number> } }

**Erro na aula**

Em 0:18, há um erro factual nesta lição. A funcionalidade correta do operador $ push é que, quando o campo não é uma matriz, a operação falhará. No entanto, se o campo estiver ausente no documento a ser atualizado, $push adiciona o campo da matriz com o valor como seu elemento.

Para saber mais sobre todos os operadores de  [update operators in MQL, visit our excellent documentation page](https://docs.mongodb.com/manual/reference/operator/update/#id1).

This lesson used the following commands:

Connect to the Atlas cluster:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

Switch to this database:

use sample\_airbnb

Find all documents with exactly 20 amenities which include all the amenities listed in the query array:

db.listingsAndReviews.find({ "amenities": {

"$size": 20,

"$all": [ "Internet", "Wifi", "Kitchen",

"Heating", "Family/kid friendly",

"Washer", "Dryer", "Essentials",

"Shampoo", "Hangers",

"Hair dryer", "Iron",

"Laptop friendly workspace" ]

}

}).pretty()

### Array Operators and Projection (Operadores de Vetores e Projeções)

db.<collection>.find( { <query> } , { <projection> } )

1 – include the field, 0 – exclude the field

Commands used in this lesson:

*Connect to the Atlas cluster:*

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

*Switch to this database:*

use sample\_airbnb

*Find all documents with exactly* 20 *amenities which include all the amenities listed in the query array, and display their* price *and* address:

db.listingsAndReviews.find({ "amenities":

{ "$size": 20, "$all": [ "Internet", "Wifi", "Kitchen", "Heating",

"Family/kid friendly", "Washer", "Dryer",

"Essentials", "Shampoo", "Hangers",

"Hair dryer", "Iron",

"Laptop friendly workspace" ] } },

{"price": 1, "address": 1}).pretty()

*Find all documents that have* Wifi *as one of the amenities only include* price *and* address *in the resulting cursor:*

db.listingsAndReviews.find({ "amenities": "Wifi" },

{ "price": 1, "address": 1, "\_id": 0 }).pretty()

*Find all documents that have* Wifi *as one of the amenities only include price and address in the resulting cursor, also exclude* [``](https://university.mongodb.com/mercury/M001/2021_September_7/chapter/Chapter_4_Advanced_CRUD_Operations/lesson/5f3725a904e9ffb277d970f7/lecture#id1)"maximum\_nights"[``](https://university.mongodb.com/mercury/M001/2021_September_7/chapter/Chapter_4_Advanced_CRUD_Operations/lesson/5f3725a904e9ffb277d970f7/lecture#id3)*. \*\*This will be an error:\**

db.listingsAndReviews.find({ "amenities": "Wifi" },

{ "price": 1, "address": 1,

"\_id": 0, "maximum\_nights":0 }).pretty()

*Switch to this database:*

use sample\_training

*Get one document from the collection:*

db.grades.findOne()

*Find all documents where the student in class* 431 *received a grade higher than* 85 *for any type of assignment:*

db.grades.find({ "class\_id": 431 },

{ "scores": { "$elemMatch": { "score": { "$gt": 85 } } }

}).pretty()

*Find all documents where the student had an extra credit score:*

db.grades.find({ "scores": { "$elemMatch": { "type": "extra credit" } }

}).pretty()

### Query Arrays and Sub-Documents (Vetores de Consulta e Sub-Documentos)

To learn more about [querying arrays using MQL visit our excellent documentation page](https://docs.mongodb.com/manual/tutorial/query-arrays/). To learn more about the $regex operator [check out this documentation page](https://docs.mongodb.com/manual/reference/operator/query/regex/index.html).

Commands used in this lesson:

Connect to the Atlas cluster:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

use sample\_training

db.trips.findOne({ "start station location.type": "Point" })

db.companies.find({ "relationships.0.person.last\_name": "Zuckerberg" },

{ "name": 1 }).pretty()

db.companies.find({ "relationships.0.person.first\_name": "Mark",

"relationships.0.title": { "$regex": "CEO" } },

{ "name": 1 }).count()

db.companies.find({ "relationships.0.person.first\_name": "Mark",

"relationships.0.title": {"$regex": "CEO" } },

{ "name": 1 }).pretty()

db.companies.find({ "relationships":

{ "$elemMatch": { "is\_past": true,

"person.first\_name": "Mark" } } },

{ "name": 1 }).pretty()

db.companies.find({ "relationships":

{ "$elemMatch": { "is\_past": true,

"person.first\_name": "Mark" } } },

{ "name": 1 }).count()

## Capítulo 5: Aggregation Framework

### $group | $sum

Com MQL (.find) podemos filtrar e atualizar dados. Com Aggregation podemos calcular e remodelar dados.

{ $group:

{

**\_id**: <expression>, //Group By Expression

<field1>: { <accumulator1> : <expression1> }, ...} }

{ $group:

{

**\_id**: "**$address.country**", //Group By Expression

<field1>: { <accumulator1> : <expression1> }, ...} }

To learn more about aggregation, take our [Aggregation Course](https://university.mongodb.com/courses/M121/about)!

Commands used in this lesson:

Connect to the Atlas cluster:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

Switch to this database:

use sample\_airbnb

Find all documents that have Wifi as one of the amenities. Only include price and address in the resulting cursor.

db.listingsAndReviews.find({ "amenities": "Wifi" },

{ "price": 1, "address": 1, "\_id": 0 }).pretty()

Using the aggregation framework find all documents that have Wifi as one of the amenities``\*. Only include\* ``price and address in the resulting cursor.

db.listingsAndReviews.aggregate([

{ "$match": { "amenities": "Wifi" } },

{ "$project": { "price": 1,

"address": 1,

"\_id": 0 }}]).pretty()

Find one document in the collection and only include the address field in the resulting cursor.

db.listingsAndReviews.findOne({ },{ "address": 1, "\_id": 0 })

Project only the address field value for each document, then group all documents into one document per address.country value.

db.listingsAndReviews.aggregate([ { "$project": { "address": 1, "\_id": 0 }},

{ "$group": { "\_id": "$address.country" }}])

Project only the address field value for each document, then group all documents into one document per address.country value, and count one for each document in each group.

db.listingsAndReviews.aggregate([

{ "$project": { "address": 1, "\_id": 0 }},

{ "$group": { "\_id": "$address.country",

"count": { "$sum": 1 } } }

])

### *sort() | limit()*

db.<collection>.find().sort( { "field1":1 , "field2”: -1 } )

1 = ascendente

-1 = descendente

.find().sort().limit(n) ⬄ .find.limit(n).sort()

There is another fun method called skip() that might come in handy with sort() and limit(). [Read more about it on the documentation page.](https://docs.mongodb.com/manual/reference/method/cursor.skip/index.html)

Commands used in this lesson:

Connect to the Atlas cluster:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

use sample\_training

db.zips.find().sort({ "pop": 1 }).limit(1)

db.zips.find({ "pop": 0 }).count()

db.zips.find().sort({ "pop": -1 }).limit(1)

db.zips.find().sort({ "pop": -1 }).limit(10)

db.zips.find().sort({ "pop": 1, "city": -1 })

### Introdution to Indexes

To learn more about performance and indexing with MongoDB, take our [MongoDB Performance Course](https://university.mongodb.com/courses/M201/about)!

Commands used in this lesson:

mongo "mongodb+srv://<username>:<password>@<cluster>.mongodb.net/admin"

use sample\_training

db.trips.find({ "birth year": 1989 })

db.trips.find({ "start station id": 476 }).sort( { "birth year": 1 } )

db.trips.createIndex({ "birth year": 1 })

db.trips.createIndex({ "start station id": 1, "birth year": 1 })

### Introdution to Data Modeling (Introdução à Modelagem de Dados)

Modelagem de dados - uma maneira de organizar campos em um documento para oferecer suporte ao desempenho do aplicativo e aos recursos de consulta.

To learn more about data modeling with MongoDB, take our [Data Modeling Course](https://university.mongodb.com/courses/M320/about)! Check out our [documentation](https://docs.mongodb.com/manual/core/data-modeling-introduction/) and [blog](https://www.mongodb.com/blog/post/building-with-patterns-a-summary).

**Regra mais importante**: Os dados são armazenados da maneira como são usados.

Sempre considerar a forma como são consultados. Tudo que é regularmente consultado junto deve ser armazenado junto para uma recuperação rápida.

### Upsert (Update or Insert)

É um híbrido de inserção e atualização.

db.collection.updateOne( {<query>} , {<update>} , {**"upsert": true**} )

Commands used in this lesson:

db.iot.updateOne({ "sensor": r.sensor, "date": r.date,

"valcount": { "$lt": 48 } },

{ "$push": { "readings": { "v": r.value, "t": r.time } },

"$inc": { "valcount": 1, "total": r.value } },

{ "upsert": true })

## Capítulo 6: Próximos Passos

### Atlas Features - More Data Explorer

Try out Atlas Search by [following this tutorial](https://docs.atlas.mongodb.com/reference/atlas-search/tutorial/#fts-tutorial-ref)!

### Atlas Products and Options

To learn more about **Realm** [check out our Realm documentation](https://docs.mongodb.com/realm/).

To learn about application development with MongoDB take a [course from our Developer series](https://university.mongodb.com/learning_paths/developer).

Check out this [**Charts** tutorial](https://docs.mongodb.com/charts/saas/tutorial/order-data/order-data-tutorial-overview/)!

### What is MongoDB Compass?

Follow this link to [download MongoDB Compass](https://www.mongodb.com/try/download/compass).

To learn more about performance and indexing with MongoDB, take our [MongoDB Performance Course](https://university.mongodb.com/courses/M201/about)!

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

### What is MongoDB Compass? (part 2)

[Documentation about the Validation tab in Compass](https://docs.mongodb.com/compass/current/validation/).

[Schema Validation with MongoDB](https://docs.mongodb.com/manual/core/schema-validation/) Documentation.

[JSON Schema Validation - Locking down your model the smart way](https://www.mongodb.com/blog/post/json-schema-validation--locking-down-your-model-the-smart-way).

[JSON Schema Validation - Checking Your Arrays](https://www.mongodb.com/blog/post/json-schema-validation--checking-your-arrays)

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

### Why MongoDB? (optional)

At this point you have learned enough to start having a meaningful conversation about why developers, DBAs, and companies choose MongoDB as their database.

This can also help you decide whether MongoDB is the right choice for you.

To help you learn more about specific use cases, we put together case studies and articles about some of our customers and community members.

We also added a link to the MongoDB Developer Hub and Community Forums where you can ask questions, learn new things, and connect with others who are diving into MongoDB content and products.

Take a look and feel free to browse for more information.

* [MongoDB Developer Hub](https://developer.mongodb.com/)
* [MongoDB Community Forums](https://developer.mongodb.com/community/forums/)
* [Case study: Bosch Leads Charge into Internet of Things](https://www.mongodb.com/customers/bosch)
* [Case study: Forbes](https://www.mongodb.com/blog/post/forbes-cloud-migration-helps-worlds-biggest-media-brand-continue-standard-digital-innovation)
* [Case study: SEGA](https://www.mongodb.com/blog/post/sega-hardlight-migrates-to-mongodb-atlas-simplify-ops-improve-experience-mobile-gamers)
* [How the Financial Sector Uses MongoDB](https://www.mongodb.com/industries/financial-services)

# Curso M220P

## Capítulo 0 – Introduction and Setup

* URI 🡪 Uniform Resource Identifier
* Structure of the URI string

**Mongodb+srv://<username>:<password>@<host>/<database>**

* + SRV – Serve como um DNS para os host que iremos acessar.
  + @ define o host do registro srv. Neste caso não aponta para um host de servidor, mas hospeda os registros SRV do nome. Aqui identificaremos o servidor, mesmo que esteja em um cluster
  + **database** será o banco de dados que conterá as credenciais que utilizamos para o acesso.

### Setting Up Atlas

Database criado 🡪 mflix

Username 🡪 m220student

Password 🡪 m220password

String de conexão com o compass 🡪 mongodb+srv://m220student:m220password@mflix.o7o8g.mongodb.net/test

This lecture provides instructions for installing and configuring MongoDB Atlas and Compass, as well as importing the sample dataset we'll be using throughout the course.

**Note**

If you need to run this course using a local instance of MongoDB, you can download the sample data from [here](https://atlas-education-staging.s3.amazonaws.com/sampledata.archive.gz) and then import it using the [``mongorestore``](https://docs.mongodb.com/manual/reference/program/mongorestore/) command:

mongorestore --gzip --archive=sampledata.archive.gz

Here are a few interesting reference links for you to follow:

* [Download Compass](https://www.mongodb.com/download-center?jmp=university#compass).
* [Aggregation pipeline builder reference](https://docs.mongodb.com/compass/master/aggregation-pipeline-builder/)
* [Export to Language documentation](https://docs.mongodb.com/compass/master/export-pipeline-to-language/).

### Visão Global

# Visão geral

Teremos uma série de instruções README para poder configurar nosso aplicativo MFLIX ​​com sucesso. Com o aplicativo MFLIX, você aprenderá a criar e compartilhar uma conexão de banco de dados, realizar as operações básicas Criar, Ler, Atualizar e Excluir por meio do driver, lidar com erros, utilizar as melhores práticas do MongoDB e muito mais.

Mflix é composto por dois componentes principais:

* RADME.rst; Contém instruções detalhadas para configurar o ambiente e instruções detalhadas sobre o leiaut do projeto
* data: contém os dados de origem que usaremos para a aplicação.
* dotini: será modificado e renomeado para o curso. Conterá configurações para que o aplicativo possa se conectar ao Atlas, além de informações que a aplicação usará internamente.
* Mflix/api: Contém manipuladores de raiz para o aplicativo.
* Mflix/build: Contém o Frontend.
* Frontend : toda a funcionalidade da IU já está implementada para você, o que inclui o aplicativo interno React com o qual você não precisa se preocupar.
* Back - end : O projeto que fornece o serviço necessário ao aplicativo. O fluxo de código já está implementado, exceto algumas funções.
* Alguns HTMLs serão editados.
* Db.py: É onde a maior parte do foco do estudo se encontrará.
* Run.py: Não modificar
* Tests: Cnotém os testes de unidades. Ao longo do curso você terá labs chamados tickets.

Você só implementará as funções que chamam diretamente ao MongoDB.

# Camada de banco de dados

Usaremos o MongoDB Atlas , o banco de dados oficial como serviço (DBaaS) do MongoDB, então você não precisará gerenciar o componente de banco de dados por conta própria. No entanto, você ainda precisará instalar o MongoDB localmente para acessar as ferramentas de linha de comando que interagem com o Atlas, para carregar dados no MongoDB e potencialmente fazer alguma exploração de seu banco de dados com o shell.

As seguintes seções README estão aqui para ajudá-lo a configurar este curso.

### Setting Up mflix

**Download Course Materials - Handouts (1)**

* [m220/mflix-python.zip](https://s3.amazonaws.com/edu-downloads.10gen.com/M220P/2021/October/static/handouts/m220/mflix-python.zip). Descompactar o conteúdo em um diretório de trabalho.
* Instalar o anaconda [Windows](https://www.anaconda.com/download/). Selecionar para adicionar ao path.
* No diretório descompacado do mflix-python executar os comandos abaixo:

# enter mflix-python folder

**cd mflix-python**

# create a new environment for MFlix

**conda create --name mflix**

# activate the environment

**conda activate mflix**

* Dependências da biblioteca Python

Pode ser que você tenha que executar como administrador. Executando algumas vezes você consegue atualizar tudo sem erros.

pip install -r requirements.txt

Extra: Para instalar o Pycharm do Anaconda: <https://www.anaconda.com/pycharm>

### 

### Executando o aplicativo

No diretório mflix-python , você pode encontrar um arquivo chamado dotini .

Abra este arquivo e insira sua string de conexão Atlas SRV conforme direcionado no comentário. Esta é a informação que o driver usará para se conectar. Certifique-se de não quebrar sua conexão Atlas SRV entre aspas:

MFLIX\_DB\_URI = mongodb + srv: // ...

Renomeie este arquivo para .ini com o seguinte comando:

mv dotini\_unix .ini # no Unix

ren dotini\_win .ini # no Windows

Para iniciar o MFlix, execute o seguinte comando:

python run.py

### Executando os testes de unidade

Para executar os testes de unidade deste curso, você usará o pytest e precisa ser executado a partir do diretório mflix-python . Cada laboratório do curso contém um módulo de testes de unidade que você pode chamar individualmente com um comando como o seguinte:

pytest -m LAB\_UNIT\_TEST\_NAME

Cada tíquete conterá o comando para executar os testes de unidade específicos desse tíquete. Por exemplo, para executar o teste de Tíquete de Conexão, seu comando shell será:

pytest -m connection

## Capítulo 1 – Driver Setup

As implementações serão realizadas no arquivo db.py.

Devemos ler os arquivos de testes para entender melhor o que será enviado para a API.

### MongoClient

**Executando os Notebooks Jupyter**

Esta lição, como muitas lições neste curso, foi criada usando o Jupyter Notebook. Esses notebooks facilitam uma abordagem mais modular para escrever Python.

O Jupyter foi incluído em requirements.txt , portanto, já deve estar instalado em seu ambiente.

Para abrir e executar o notebooks neste curso, navegue até o diretório da aplicação e inicie o servidor **jupyter notebook**:

cd mflix-python/notebooks

jupyter notebook

Isso deve fornecer a você um link de URL para o servidor do notebook (este servidor está sendo executado em sua máquina) e você pode colá-lo no navegador para visualizar os blocos de notas. Navegue entre as células no notebook com as teclas de seta. Você pode executar cada célula do notebook com Shift-Enter . Sinta-se à vontade para editar as células individuais e executar novamente o bloco de notas.

A string SRV nos notebooks não funcionará, porque o cluster Atlas de demonstração para este curso foi desligado. Substitua por sua própria string de conexão.

### First Read

### Ticket: Projection

* Verificar se o arquivo ini criado se chama realmene “.ini”

### Cursor Methods and Agregation Equivalents

* cursor.limit()
* print(dumps(cursor,indent=2)
* pipeline
* cursor.find().sort(‘year’,1) //ascending ou -1 para descending
* cursor.find().sort([(“year”,-1) , (“title”, 1)])
* movies.find().skip(14) //Apenas os documentos que não pulamos aparecerá no cursor.

Sumário:

* .limit() == $limit
* .sort() == $sort
* .skip() == $skip

#### User-Facing Backend

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

Uma imagem contendo Linha do tempo

Descrição gerada automaticamente

Texto, Linha do tempo

Descrição gerada automaticamente com confiança média

#### Sorting

Texto

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

#### Skipping

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Este pipeline vai contar o número de documentos que será retornado no campo “num\_movies”

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Retornou só um porque existem 15 documentos e os primeiros 14 foram pulados.

Texto

Descrição gerada automaticamente

Pulamos os 10 filmes mais antigos de Sam Raimi

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

### Basic Aggregation

$match >> $project >> $group

## Capítulo 3 – Admin Backend

Read isolation

Read concern local | Majority |

### Bulk Writes

Db.stock. bulkWrite([

{ updateOne: {...} ,

updateOne: {...}

}

] ,

{ ordered: false }

)

## Capítulo 4 – Resiliency