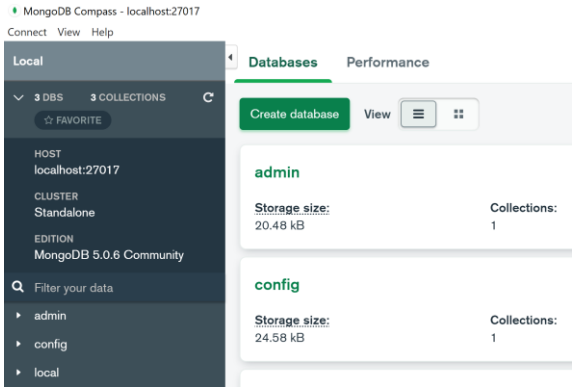
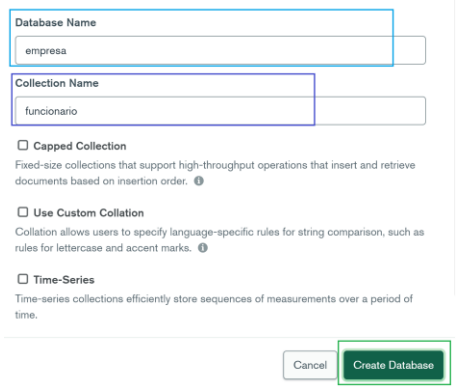
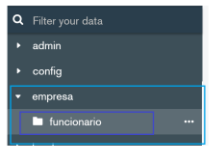
Inserir documentos no Mongo DB

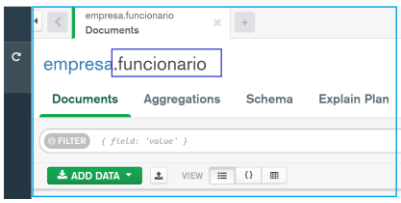
O mongoDB é o banco de dados não relacional mais popular, ele utiliza a estrutura de documentos e tem uma interface própria, o **MongoDB Compass:**



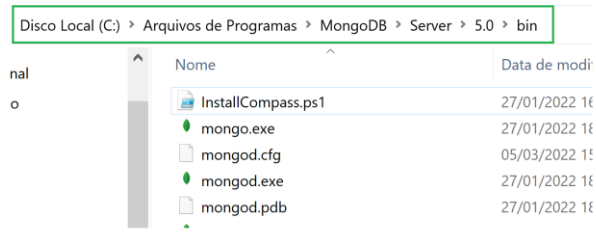
A **criação de um banco de dados** é bem simples, basta colocar o nome do banco de dados e o nome da coleção:





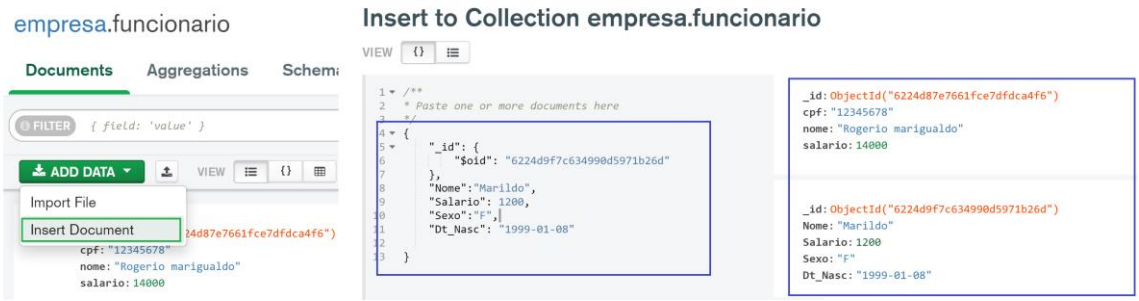


Podemos usar o mongo em **linha de comando** *(desde que entre no diretório certo)* **e inserir algum registro em formato JSON.** Todo registro no mongoDB tem uma **propriedade de ID** do documento, se não explicitarmos na inserção então ele **cria uma**:




```
> use empresa
switched to db empresa
> db.funcionario.insertOne({"cpf": "12345678", "nome":"Rogerio marigualdo", "salario":14000})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6224d87e7661fce7dfdca4f6")
}
>
```

Da pra fazer o mesmo pelo MongoDB Compass, mas ele possui algumas limitações *(não deixa instanciar um objeto de date por exemplo)*:

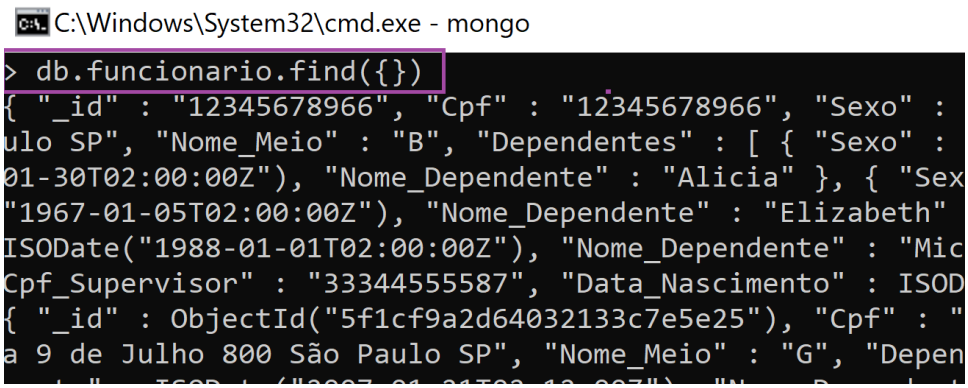


Para especificar o **valor do ID** do documento, basta explicitar a propriedade **"_id" com o valor que ela deverá usar:**

Obviamente é possível fazer a inserção de múltiplos registros passando um array de funcionários, por linha de comando usaria o ***db.funcionario.insertMany.***

Buscar documentos no Mongo DB

Por linha de comando da pra buscar todos os registros com o ***"Find({})",*** como resultado temos a busca de todos os registros apresentados com **formato Json,** não é muito bom pra ler.



O *find({})* também permite algumas sobrecargas, como passar um segundo parâmetro JSON com as propriedades que eu quero buscar, por exemplo *"find({}, {Nome: 1, Sexo: 1, Salario"* ***(1 é um booleano para true):***

```
> db.funcionario.find({}, {Nome: 1, Sexo: 1, Salario: 1})
{ "_id" : "12345678966", "Sexo" : "M", "Salario" : 30000 }
{ "_id" : ObjectId("5f1cf9a2d64032133c7e5e25"), "Sexo" : "F", "Salario" : 36000 }
{ "_id" : ObjectId("5f1cf9a2d64032133c7e5e26"), "Sexo" : "M", "Salario" : 34000 }
{ "_id" : ObjectId("5f1cf9a2d64032133c7e5e27"), "Sexo" : "F", "Salario" : 38000 }
{ "_id" : ObjectId("5f1cf9a2d64032133c7e5e28"), "Sexo" : "M", "Salario" : 18000 }
{ "_id" : ObjectId("5f1cf9a2d64032133c7e5e29"), "Sexo" : "F", "Salario" : 19000 }
{ "_id" : ObjectId("5f1cf9a2d64032133c7e5e2a"), "Sexo" : "F", "Salario" : 25000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e2b"), "Sexo" : "F", "Salario" : 41000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e2c"), "Sexo" : "M", "Salario" : 38000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e2d"), "Sexo" : "F", "Salario" : 12000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e2e"), "Sexo" : "F", "Salario" : 39000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e2f"), "Sexo" : "M", "Salario" : 55000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e30"), "Sexo" : "F", "Salario" : 18000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e31"), "Sexo" : "F", "Salario" : 41000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e32"), "Sexo" : "M", "Salario" : 19000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e33"), "Sexo" : "F", "Salario" : 43000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e34"), "Sexo" : "M", "Salario" : 25000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e35"), "Sexo" : "M", "Salario" : 22000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e36"), "Sexo" : "M", "Salario" : 18000 }
{ "_id" : ObjectId("5f1cf9a3d64032133c7e5e37"), "Sexo" : "F", "Salario" : 22000 }
Type "it" for more
```

Obviamente **propriedades de objetos** que estão **dentro desse Objeto maior na hierarquia** (no caso **Dependente** dentro de **Funcionário**) podem ser buscadas, basta navegar com o **"."**:

```
> db.funcionario.find({}, {_id: 0, "Nome": 1, "Sexo": 1, "Salario": 1, "Dependentes.Nome_Dependente": 1}).sort({"Nome": -1})
{ "Sexo" : "M", "Salario" : 30000, "Dependentes" : [ { "Nome_Dependente" : "Alicia" }, { "Nome_Dependente" : "Elizabeth" }, { "
dente" : "Michael" } ] }
{ "Sexo" : "F", "Salario" : 36000, "Dependentes" : [ { "Nome_Dependente" : "Claudio" }, { "Nome_Dependente" : "Juvenal" } ] }
{ "Sexo" : "M", "Salario" : 34000, "Dependentes" : [ { "Nome_Dependente" : "Katia" }, { "Nome_Dependente" : "Marcela" }, { "Nom
te" : "Maria" }, { "Nome_Dependente" : "Rodrigo" } ] }
{ "Sexo" : "F", "Salario" : 38000, "Dependentes" : [ { "Nome_Dependente" : "Jonas" }, { "Nome_Dependente" : "Orlando" } ] }
{ "Sexo" : "M", "Salario" : 18000, "Dependentes" : [ { "Nome_Dependente" : "Priscila" } ] }
{ "Sexo" : "F", "Salario" : 19000, "Dependentes" : [ { "Nome_Dependente" : "Clecio" } ] }
{ "Sexo" : "F", "Salario" : 25000 }
```

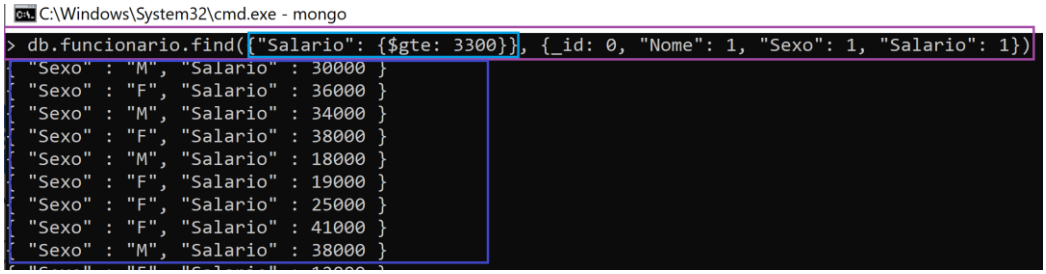
*****algumas condições como ordem de busca, limite e etc podem ser encadeados *****

Filtros simples

O primeiro "{}" dentro do Find() serve como um filtro simples onde basta passar a propriedade e o valor (funciona como um critério de igualdade), então buscar por Sexo masculino seria : ***"find({"Sexo": M}, {xxxxxxxxx})":***

```
> db.funcionario.find({"Sexo": "M"}, {_id: 0, "Nome": 1, "Sexo": 1, "Salario": 1})
{ "Sexo" : "M", "Salario" : 30000 }
{ "Sexo" : "M", "Salario" : 34000 }
{ "Sexo" : "M", "Salario" : 18000 }
{ "Sexo" : "M", "Salario" : 38000 }
{ "Sexo" : "M", "Salario" : 55000 }
{ "Sexo" : "M", "Salario" : 19000 }
{ "Sexo" : "M", "Salario" : 25000 }
{ "Sexo" : "M", "Salario" : 22000 }
{ "Sexo" : "M", "Salario" : 18000 }
{ "Sexo" : "M", "Salario" : 22000 }
```

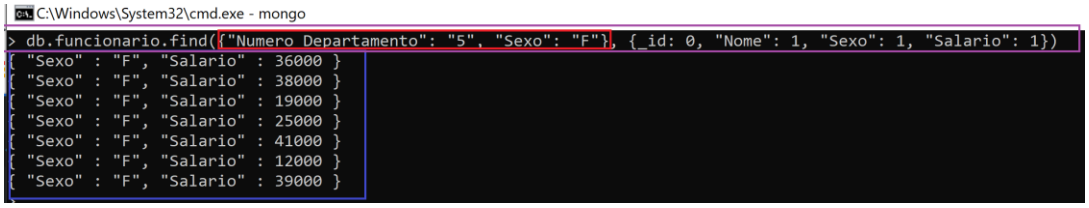
Filtros como **maior/menor** são representados por **$gt/lt *(Greather than, less than)***, **maior/menor ou igual** acrescenta o **"e" *(equals)***, então uma busca por funcionário com salário maior ou igual a 3300 seria: **"find({"Salario": {$gte: 3300}}, {xxxxxxxx})":**



Para operar gte/lte com data é necessário usar o new Date(xxxx)*

Filtros lógicos

Se eu quiser atender mais condições no Find() basta usar **","** no primeiro **"{}"**. Ele vai se comportar como um **"and"/"or"**, então se eu quisesse buscar todos os funcionários Femininos do Departamento 5 eu faria: **"find({"Numero_Departamento": "5", "Sexo": "F"}, {xxxxxxx})":**



Mas nada impede de explicitar o uso desses filtros com *find($and: [{"xxxx"}, {"xxx"}], {xxxx})*