

## Relatório — Infraestrutura MongoDB

Aluno: Caio Barroso | Disciplina: Infraestrutura MongoDB | Professor: Ricardo Barrionuevo Garcia

### Rubrica / Evidências (o que o avaliador procura)

Item da rubrica	Onde está no relatório	Evidência (print)
1) Diferenças e semelhanças (SQL vs NoSQL)	Parte 1.1	Texto
1) NoSQL é melhor/pior? Destaques e justificativa	Parte 1.2	Texto
1) Cenário NoSQL e cenário SQL	Parte 1.3	Texto
1) Tipos de NoSQL + características + fontes	Parte 1.4	Texto
2) A4 — Conteúdo das coleções	A4	evidencias/A04_conteudo_colecoes.png
2) A5 — Filtro igualdade em DEPARTAMENTO	A5	evidencias/A05_filtro_igualdade_departamento.png
2) A6 — Salário > 2000 em FUNCIONARIO	A6	evidencias/A06_salario_maior_2000.png
2) A7 — distinct(nome) em DEPENDENTE	A7	evidencias/A07_distinct_dependente_nome.png
2) A8 — update em uma coleção	A8	evidencias/A08_update_funcionario.png
2) A9 — delete/remove em uma coleção	A9	evidencias/A09_delete_dependente.png
3) B2 — Modelagem 1-N (referência)	B2	evidencias/B01_modelagem_1N.png
3) B3 — Modelagem N-N	B3	evidencias/B02_modelagem_NN.png
3) C2/C3/C4 — Índices/TTL/Text/Agregações (itens principais)	C Índices e C Agregações	evidencias/C01_indexar1_indices.png evidencias/C02_ttl_antes.png evidencias/C03_ttl_depois.png evidencias/C04_text_index_busca.png evidencias/C05_agregacoes_por_uf.png
4) D3/D4 — Replica set (criação + leitura no secundário)	D	evidencias/D01_mongod_rs_subindo.png evidencias/D02_rs_status.png evidencias/D13_insert_5_filmes_primary.png evidencias/D14_read_filmes_secondary.png
4) E3 — Sharding (mongos + add shards + shardCollection + distribuição)	E	evidencias/E06_mongos_conectado_27050.png evidencias/E07_add_shards_status.png evidencias/E08_enable_sharding_db.png evidencias/E09_shard_collection.png evidencias/E10_insert_1000_docs.png evidencias/E11_distribuicao_shards.png
4) F1/F2/F3 — Storage engines e Segurança	F1, F2, F3	evidencias/F01_mmapv1_storage_engine.png evidencias/F02_mmapv1_produtos_insert_4.png evidencias/F04_wiredTiger_storage_engine.png evidencias/F05_wiredTiger_lugares_insert_4.png evidencias/F09_secure_login_dba_connectionStatus.png evidencias/F11_createUser_CB_desenv.png evidencias/F12_login_CB_desenv.png evidencias/F13_insert_funcionarios_4.png
4) G9–G13 — Backup/kill/delete/restore/validação	G	evidencias/G11_backup_mongodump.png evidencias/G10_taskkill.png evidencias/G11_apagar_dados.png evidencias/G12_restore_ok.png evidencias/G13_contagem_colecoes_exerc4b.png

### Parte 1 — Conceitos

## 1.1 — 3 diferenças e 3 semelhanças entre SQL e NoSQL

### Diferenças (3)

- 1) SQL trabalha tipicamente com **esquema rígido** (tabelas/colunas e constraints); NoSQL tende a aceitar **esquema flexível**, facilitando evoluções incrementais do modelo.
- 2) SQL costuma priorizar **consistência forte** e transações com múltiplas entidades; NoSQL frequentemente prioriza **escala/alta disponibilidade**, aceitando consistência eventual em alguns cenários.
- 3) SQL é muito bom para dados relacionais e joins; NoSQL geralmente evita joins complexos (ou usa alternativas), apostando em **modelagem orientada ao acesso** e **distribuição horizontal**.

### Semelhanças (3)

- 1) Ambos existem para **persistência** e **consulta** de dados de forma confiável.
- 2) Ambos suportam **índices**, controle de acesso, logs e rotinas de backup/restore.
- 3) Em ambos, performance depende de modelagem + índices + padrão de consultas e de operação (monitoramento/capacidade).

## 1.2 — NoSQL é melhor ou pior que SQL?

### Resposta

Não existe “melhor/pior” universal. **NoSQL** se destaca em **escala horizontal**, alta taxa de leitura/escrita, dados semi-estruturados e evolução rápida de schema. **SQL** se destaca quando a prioridade é **integridade forte**, transações complexas e relacionamento intenso entre entidades (joins) com regras rígidas. A decisão correta depende de volume, padrão de acesso, SLA de consistência e custo operacional.

## 1.3 — Um cenário NoSQL e um cenário SQL

### Cenário NoSQL

Telemetria/eventos de aplicação e perfis de usuário que evoluem (campos variáveis), com necessidade de escalar por volume e consultas por atributos/ids. Ex.: logs, eventos de navegação, catálogo com campos opcionais.

### Cenário SQL

Sistema financeiro/ERP com regras fortes, auditoria e transações entre várias entidades (pedido/pagamento/faturamento), exigindo consistência e integridade referencial.

## 1.4 — Tipos de bancos NoSQL e características

### Resposta

- **Key-Value:** chave → valor. Muito rápido; bom para cache, sessão, feature flags, leitura direta por chave.
- **Document:** documentos (JSON/BSON). Flexível; bom para dados semi-estruturados e evolução rápida (ex.: MongoDB).
- **Wide-column/Column-family:** famílias de colunas; bom para escala distribuída e consultas por partição.
- **Graph:** nós/arestas; excelente para relacionamentos, recomendação, fraude e análise de redes.

Fontes:

- AWS (conceitos e categorias NoSQL): <https://docs.aws.amazon.com/whitepapers/latest/choosing-an-aws-nosql-database/conclusion.html>
- MongoDB (conceito de document database): <https://www.mongodb.com/resources/basics/databases/document-databases>

## Parte 2 — Prática (A: Instalações e criações)

## A4 — Mostrar o conteúdo de cada coleção

### O que foi pedido

Mostrar o conteúdo das coleções do database **CB\_rh**.

### Comandos

```
use CB_rh
db.CB_DEPARTAMENTO.find().pretty()
db.CB_FUNCIONARIO.find().pretty()
db.CB_DEPENDENTE.find().pretty()
```



evidencias/A04\_conteudo\_colecoes.png

## A5 — Filtro por igualdade em DEPARTAMENTO

### O que foi pedido

Filtro de igualdade na coleção **CB\_DEPARTAMENTO** por descrição.

### Comandos

```
use CB_rh
db.CB_DEPARTAMENTO.find({ descricao: "..." }).pretty()
```



evidencias/A05\_filtro\_igualdade\_departamento.png

## A6 — Funcionários com salário acima de 2000

### O que foi pedido

Mostrar em **CB\_FUNCIONARIO** quem tem salário > **2000**.

### Comandos

```
use CB_rh
db.CB_FUNCIONARIO.find({ salario: { $gt: 2000 } }).pretty()
```



evidencias/A06\_salario\_maior\_2000.png

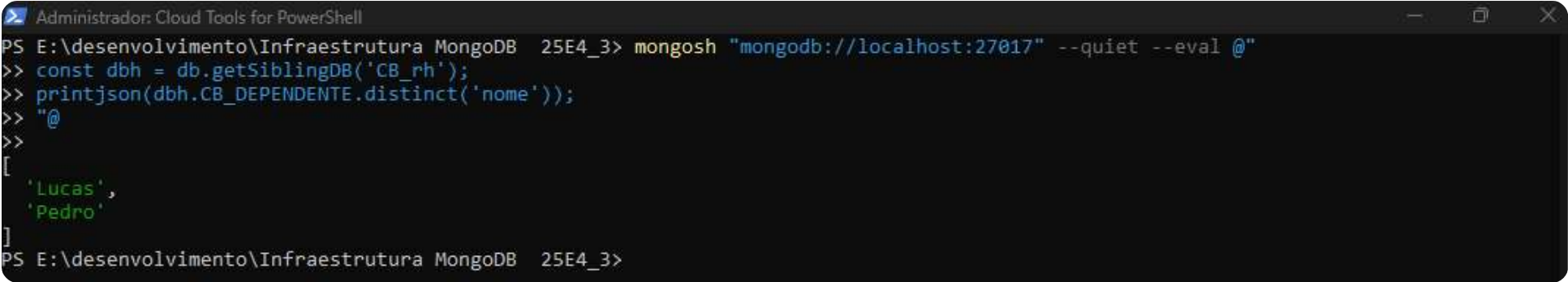
## A7 — Distinct no atributo nome em DEPENDENTE

### O que foi pedido

Executar **distinct** no atributo **nome** na coleção **CB\_DEPENDENTE**.

### Comandos

```
use CB_rh
db.CB_DEPENDENTE.distinct("nome")
```



evidencias/A07\_distinct\_dependente\_nome.png

## A8 — Update em uma coleção

### O que foi pedido

Executar um **update** em uma coleção.

### Comandos

```
use CB_rh
db.CB_FUNCIONARIO.updateOne(
  { _id: ... },
  { $set: { ... } }
)
db.CB_FUNCIONARIO.find({ _id: ... }).pretty()
```



evidencias/A08\_update\_funcionario.png

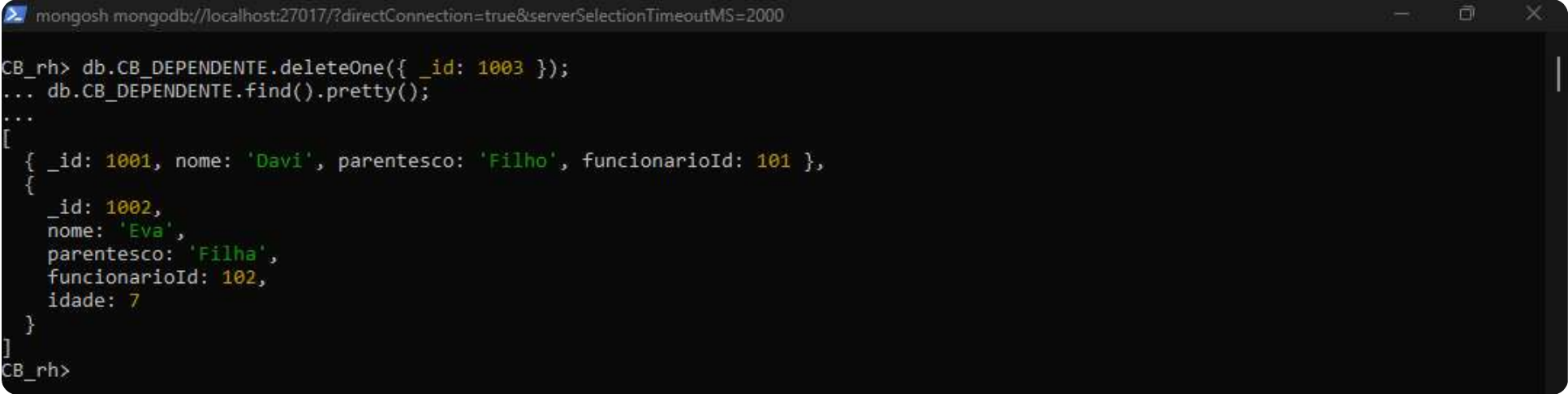
## A9 — Delete (remove) em uma coleção

### O que foi pedido

Executar um **delete/remove** em uma coleção.

### Comandos

```
use CB_rh
db.CB_DEPENDENTE.deleteOne({ _id: ... })
db.CB_DEPENDENTE.find().pretty()
```



evidencias/A09\_delete\_dependente.png

## Parte 2 — Prática (B: Modelagem)



B2 — Modelagem 1-N com referência

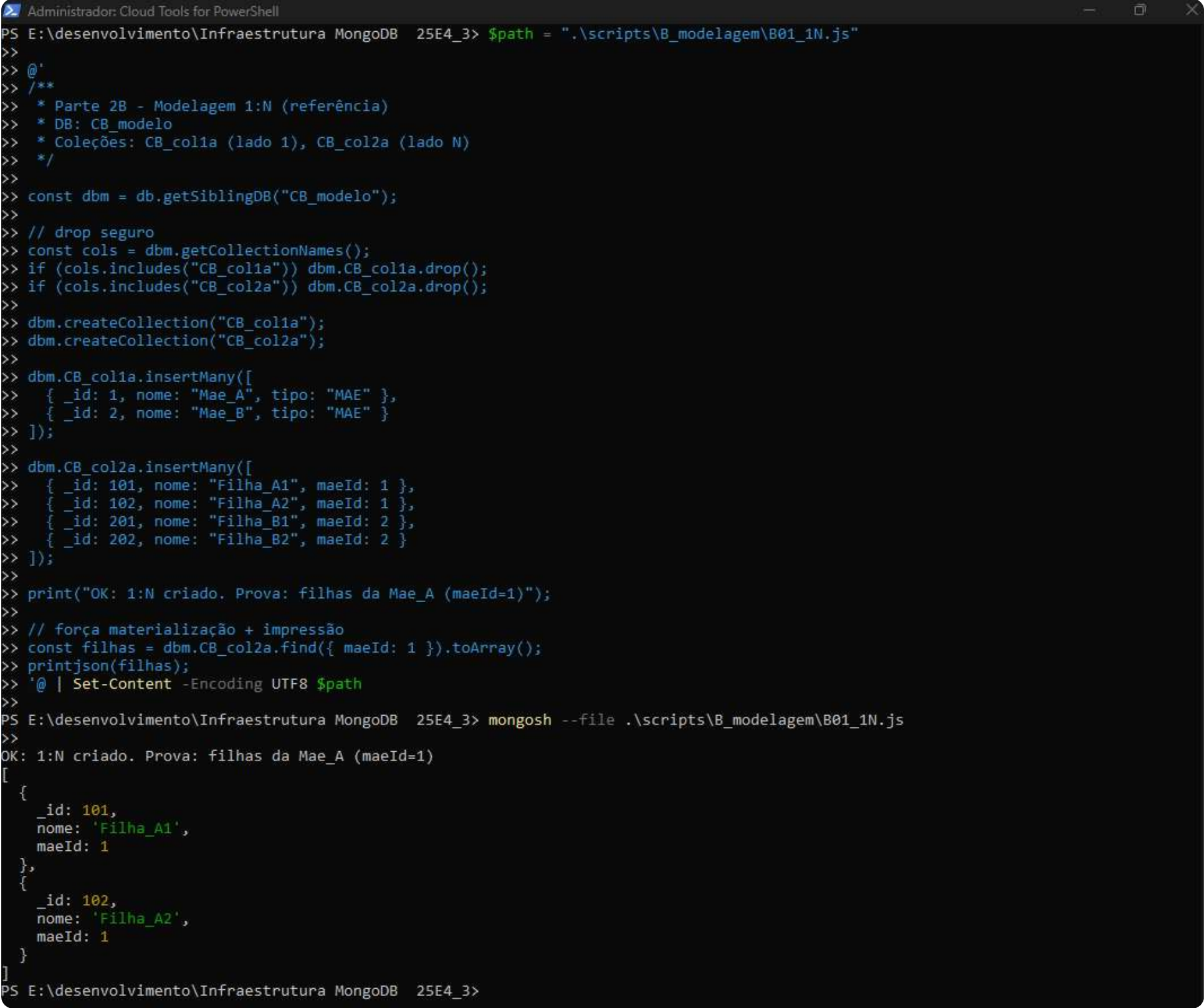
Comandos (resumo)

```
use CB_modelo
db.CB_col1a.insertMany([ {...}, {...} ])

db.CB_col2a.insertMany([
  { ..., col1a_id: ObjectId("...") },
  { ..., col1a_id: ObjectId("...") },
  { ..., col1a_id: ObjectId("...") },
  { ..., col1a_id: ObjectId("...") }
])
```

Evidência

evidencias/B01\_modelagem\_1N.png



evidencias/B01\_modelagem\_1N.png

### B3 — Modelagem N-N

#### Comandos (resumo)

```
use CB_modelo
db.CB_col1b.insertMany([ {...}, {...} ])
db.CB_col2b.insertMany([ {...}, {...} ])
// se houver coleção de ligação, ela foi criada e populada no mesmo f
```

#### Evidência

evidencias/B02\_modelagem\_NN.png

```
Administrador: Cloud Tools for PowerShell
PS E:\desenvolvimento\Infraestrutura MongoDB 25E4_3> $path = ".\scripts\B_modelagem\B02_NN.js"
>>
>> @'
>> /**
>>  * Parte 2B - Modelagem N:N (referência via coleção de relacionamento)
>>  * DB: CB_modelo
>>  * Coleções: CB_col1b, CB_col2b, CB_rel1b2b
>>  */
>>
>> const dbm = db.getSiblingDB("CB_modelo");
>>
>> const cols = dbm.getCollectionNames();
>> if (cols.includes("CB_col1b")) dbm.CB_col1b.drop();
>> if (cols.includes("CB_col2b")) dbm.CB_col2b.drop();
>> if (cols.includes("CB_rel1b2b")) dbm.CB_rel1b2b.drop();
>>
>> dbm.createCollection("CB_col1b");
>> dbm.createCollection("CB_col2b");
>> dbm.createCollection("CB_rel1b2b");
>>
>> // Entidade A: 3 docs
>> dbm.CB_col1b.insertMany([
>>   { _id: 1, nome: "A1" },
>>   { _id: 2, nome: "A2" },
>>   { _id: 3, nome: "A3" }
>> ]);
>>
>> // Entidade B: 3 docs
>> dbm.CB_col2b.insertMany([
>>   { _id: 10, nome: "B10" },
>>   { _id: 20, nome: "B20" },
>>   { _id: 30, nome: "B30" }
>> ]);
>>
>> // Relações (6 docs) para provar N:N
>> dbm.CB_rel1b2b.insertMany([
>>   { _id: 1001, aId: 1, bId: 10 },
>>   { _id: 1002, aId: 1, bId: 20 },
>>   { _id: 1003, aId: 2, bId: 20 },
>>   { _id: 1004, aId: 2, bId: 30 },
>>   { _id: 1005, aId: 3, bId: 10 },
>>   { _id: 1006, aId: 3, bId: 30 }
>> ]);
>>
>> print("OK: N:N criado.");
>>
>> print("Prova 1: relações de A1 (aId=1)");
>> printjson(dbm.CB_rel1b2b.find({ aId: 1 }).toArray());
>>
>> print("Prova 2: A's relacionados ao B20 (bId=20)");
>> printjson(dbm.CB_rel1b2b.find({ bId: 20 }).toArray());
>> '@ | Set-Content -Encoding UTF8 $path
>>
PS E:\desenvolvimento\Infraestrutura MongoDB 25E4_3> mongosh --file .\scripts\B_modelagem\B02_NN.js
>>
OK: N:N criado.
Prova 1: relações de A1 (aId=1)
[
  {
    _id: 1001,
    aId: 1,
    bId: 10
  },
  {
    _id: 1002,
    aId: 1,
    bId: 20
  }
]
Prova 2: A's relacionados ao B20 (bId=20)
[
  {
    _id: 1002,
    aId: 1,
    bId: 20
  },
  {
    _id: 1003,
    aId: 2,
    bId: 20
  }
]
PS E:\desenvolvimento\Infraestrutura MongoDB 25E4_3>
```

evidencias/B02\_modelagem\_NN.png

## Parte 2 — Prática (C: Índices e Agregações)



## C — Índices (indexar1/indexar2) e TTL

### Comandos (resumo)

```
use CB_indeagreg

db.CB_indexar1.createIndex({ campoAsc: 1 })
db.CB_indexar1.createIndex({ campoUnico: -1 }, { unique: true })
db.CB_indexar1.createIndex({ campoA: 1, campoB: 1 })
db.CB_indexar1.createIndex({ tags: 1 })
db.CB_indexar1.createIndex({ opcional: 1 }, { sparse: true })
db.CB_indexar1.createIndex({ expiraEm: 1 }, { expireAfterSeconds: 20

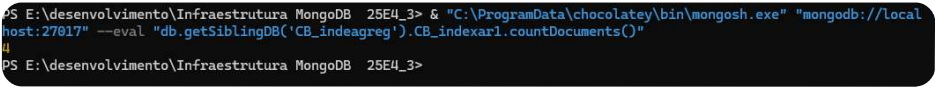
db.CB_indexar2.createIndex({ a:"text", b:"text", c:"text", d:"text" })
db.CB_indexar2.find({ $text: { $search: "..." } })
```

### Evidências

evidencias/C01\_indexar1\_indices.png  
evidencias/C02\_ttl\_antes.png  
evidencias/C03\_ttl\_depois.png  
evidencias/C04\_text\_index\_busca.png



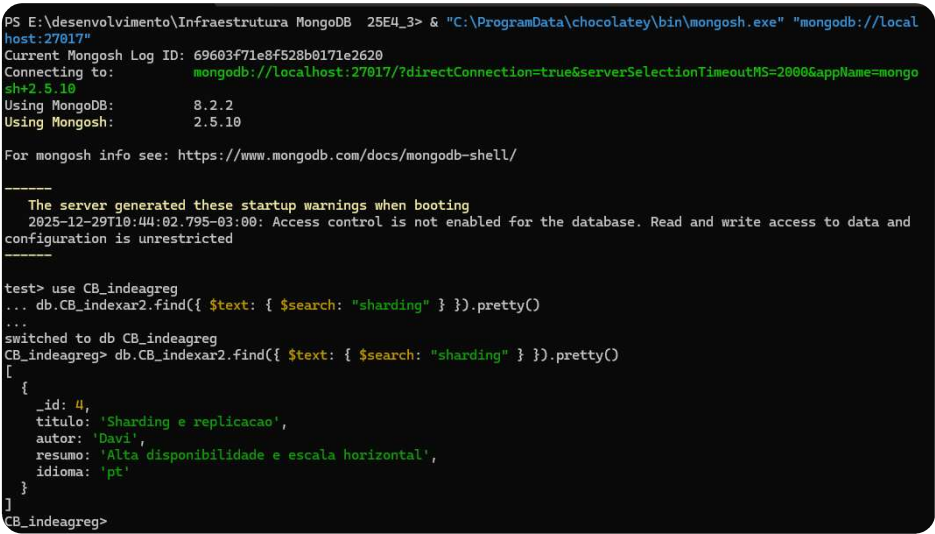
evidencias/C01\_indexar1\_indices.png



evidencias/C02\_ttl\_antes.png



evidencias/C03\_ttl\_depois.png



evidencias/C04\_text\_index\_busca.png

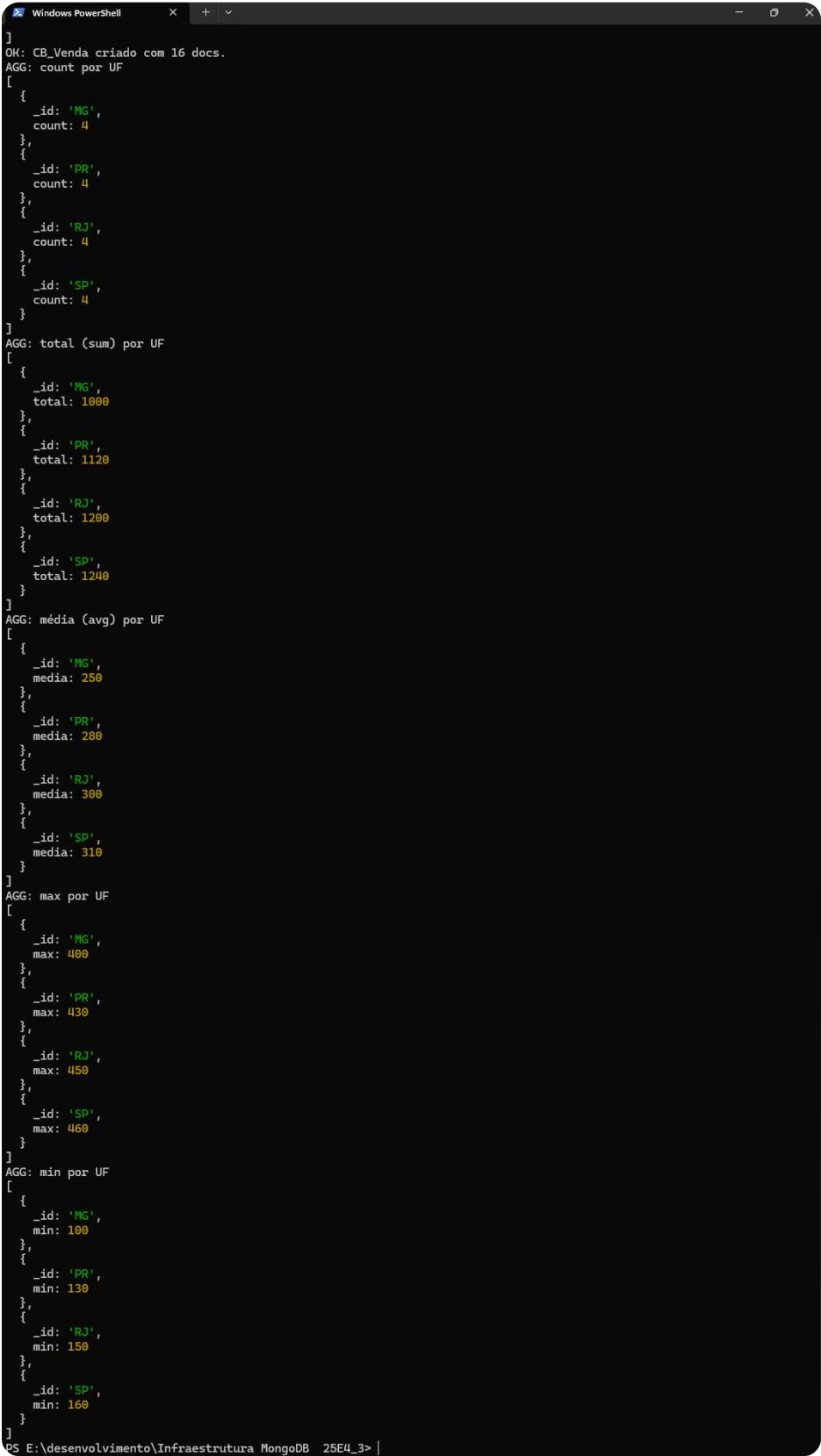
## C — Agregações por UF em CB\_Venda

### Comando (resumo)

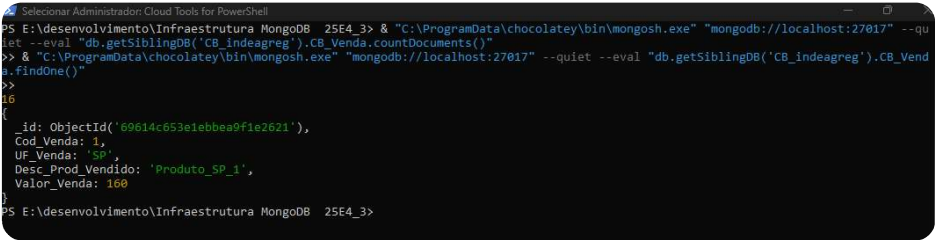
```
use CB_indeagreg
db.CB_Venda.aggregate([
  { $group: {
    _id: "$UF_Venda",
    qtd: { $sum: 1 },
    total: { $sum: "$Valor_Venda" },
    media: { $avg: "$Valor_Venda" },
    maior: { $max: "$Valor_Venda" },
    menor: { $min: "$Valor_Venda" }
  }},
  { $sort: { _id: 1 } }
])
```

### Evidências

evidencias/C05\_agregacoes\_por\_uf.png  
evidencias/C07\_venda\_count\_16.png



evidencias/C05\_agregacoes\_por\_uf.png



evidencias/C07\_venda\_count\_16.png

## Parte 2 — Prática (D: Replicação)

## D — Replica Set CB\_rsposmit (primário, 2 secundários, árbitro)

### Comandos (resumo)

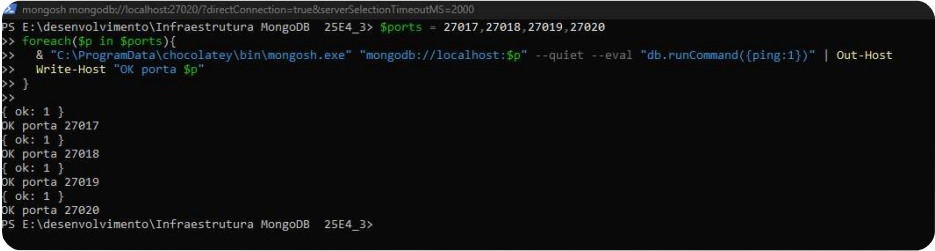
```
rs.initiate(...)
rs.add("127.0.0.1:<sec1>")
rs.add("127.0.0.1:<sec2>")
rs.addArb("127.0.0.1:<arb>")

use CB_rsposmit
db.CB_col_filmes.insertMany([ {...},{...},{...},{...},{...} ])

// nó secundário
rs.slaveOk()
db.CB_col_filmes.find().pretty()
```

### Evidências

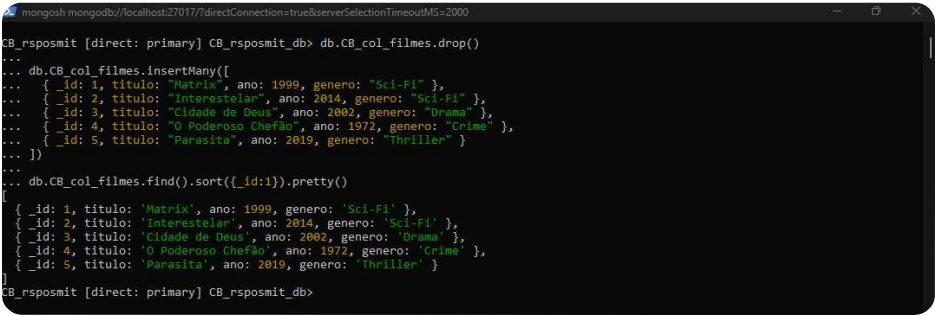
evidencias/D01\_mongod\_rs\_subindo.png  
evidencias/D02\_rs\_status.png  
evidencias/D13\_insert\_5\_filmes\_primary.png  
evidencias/D14\_read\_filmes\_secondary.png



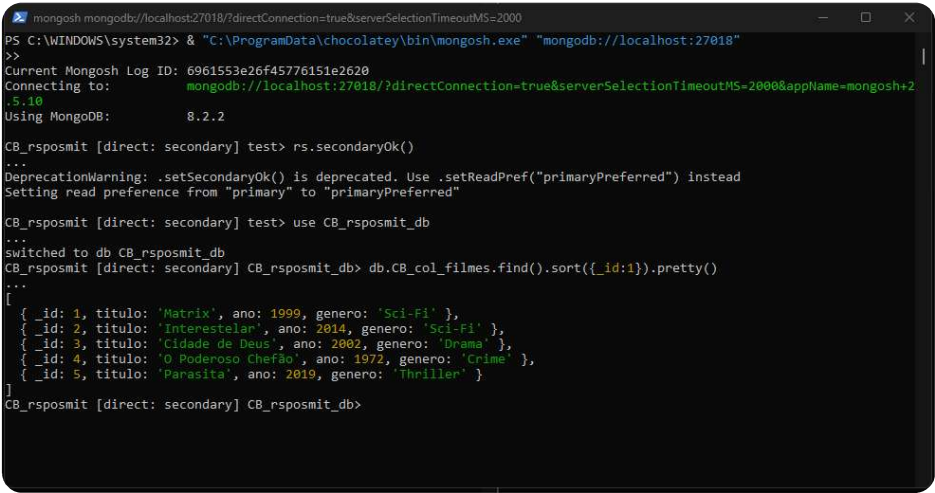
evidencias/D01\_mongod\_rs\_subindo.png



evidencias/D02\_rs\_status.png



evidencias/D13\_insert\_5\_filmes\_primary.png



evidencias/D14\_read\_filmes\_secondary.png

## Parte 2 — Prática (E: Particionamento / Sharding)

## E — Sharding (configRS + 4 shards + mongos)

### Comandos (resumo)

```
sh.addShard("127.0.0.1:27200")
sh.addShard("127.0.0.1:27201")
sh.addShard("127.0.0.1:27202")
sh.addShard("127.0.0.1:27203")

sh.enableSharding("CB_algumdb")
sh.shardCollection("CB_algumdb.CB_algumacol", { chave: 1 })

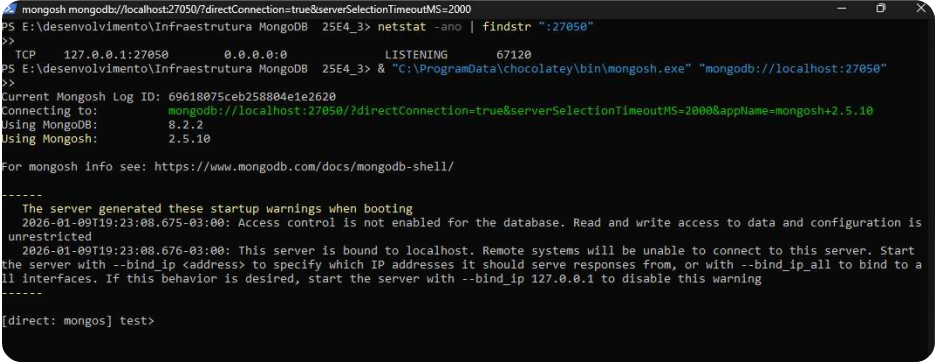
for (var i=1; i<=1000; i++){
  db.CB_algumacol.insert({ chave:i, payload:"x" })
}

db.CB_algumacol.getShardDistribution()
```

### Evidências

- evidencias/E06\_mongos\_conectado\_27050.png
- evidencias/E07\_add\_shards\_status.png
- evidencias/E08\_enable\_sharding\_db.png
- evidencias/E09\_shard\_collection.png
- evidencias/E10\_insert\_1000\_docs.png
- evidencias/E11\_distribuicao\_shards.png





evidencias/E06\_mongos\_conectado\_27050.png



evidencias/E07\_add\_shards\_status.png

```
mongosh mongodb://localhost:27050/?directConnection=true&serverSelectionTimeoutMS=2000

[direct: mongos] test> sh.enableSharding("CB_sharded")
...
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1767997716, i: 21 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1767997716, i: 21 })
}
[direct: mongos] test>
```

evidencias/E08\_enable\_sharding\_db.png

```
mongosh mongodb://localhost:27050/?directConnection=true&serverSelectionTimeoutMS=2000

[direct: mongos] test> use CB_sharded
switched to db CB_sharded
[direct: mongos] CB_sharded> db.CB_items.createIndex({ cod: 1 })
cod_1
[direct: mongos] CB_sharded> sh.shardCollection("CB_sharded.CB_items", { cod: 1 })
{
  collectionsharded: 'CB_sharded.CB_items',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1767997810, i: 41 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1767997810, i: 40 })
}
[direct: mongos] CB_sharded>
```

evidencias/E09\_shard\_collection.png

```
[direct: mongos] CB_sharded> for (let i = 1; i <= 1000; i++) {
...   db.CB_items.insertOne({ cod: i, nome: "Item_" + i })
... }
...
{
  acknowledged: true,
  insertedId: ObjectId('696181a5ceb258804e1e2a08')
}
```

evidencias/E10\_insert\_1000\_docs.png

```
[direct: mongos] CB_sharded> db.CB_items.countDocuments()
... db.CB_items.getShardDistribution()
...
Shard sh4RS at sh4RS/localhost:27203
{
  data: '48KiB',
  docs: 1000,
  chunks: 1,
  'estimated data per chunk': '48KiB',
  'estimated docs per chunk': 1000
}
---
Totals
{
  data: '48KiB',
  docs: 1000,
  chunks: 1,
  'Shard sh4RS': [
    '100 % data',
    '100 % docs in cluster',
    '49B avg obj size on shard'
  ]
}
[direct: mongos] CB_sharded>
```

evidencias/E11\_distribuicao\_shards.png

## Parte 2 — Prática (F: Storage Engines e Segurança)

## F1 — Instância mmapv1 + CB\_produtos (4 docs)

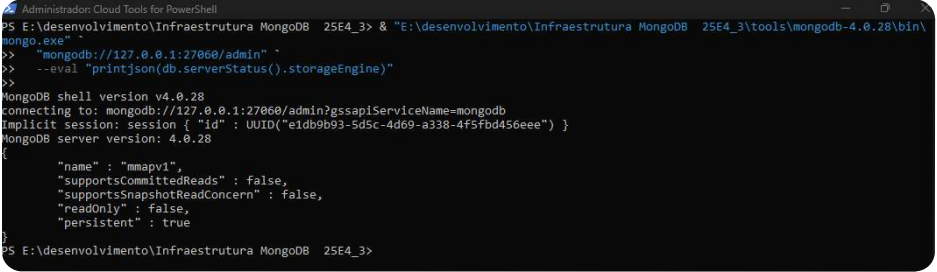
### Comandos (resumo)

```
mongod --port 27060 --dbpath <...> --logpath <...> --logappend --storage
mongo mongodb://127.0.0.1:27060/admin --eval "db.serverStatus().storage"

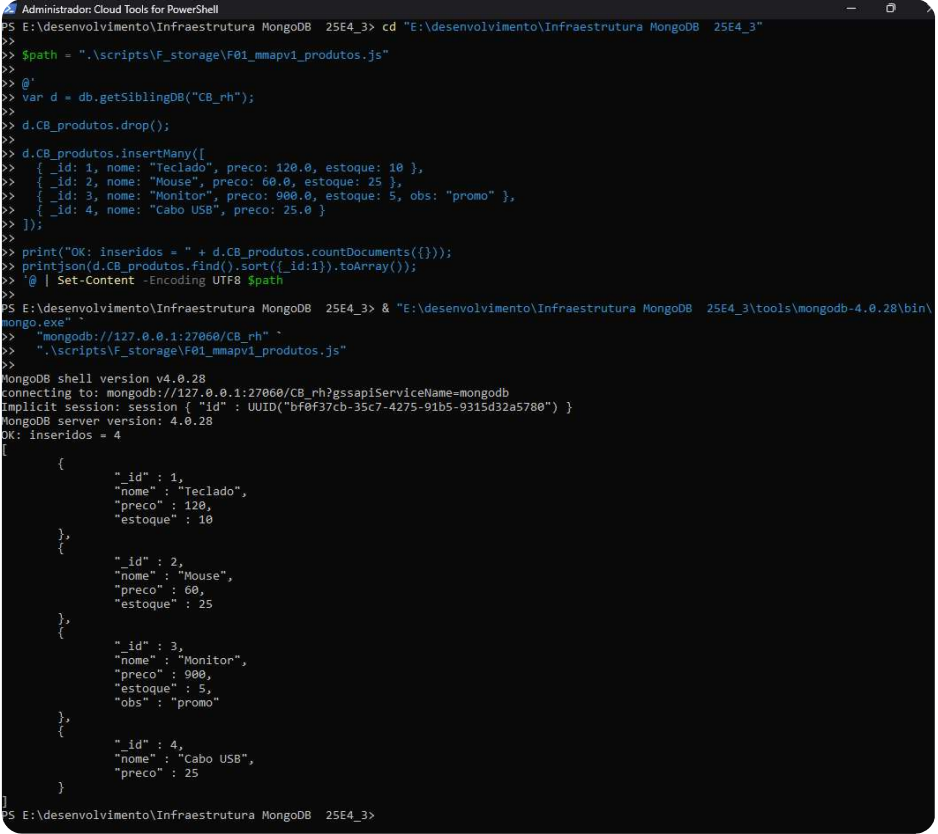
use CB_rh
db.CB_produtos.insertMany([ { ... }, { ... }, { ... }, { ... } ])
```

### Evidências

evidencias/F01\_mmapv1\_storage\_engine.png  
evidencias/F02\_mmapv1\_produtos\_insert\_4.png



evidencias/F01\_mmapv1\_storage\_engine.png



evidencias/F02\_mmapv1\_produtos\_insert\_4.png

## F2 — Instância wiredTiger + CB\_lugares (4 docs)

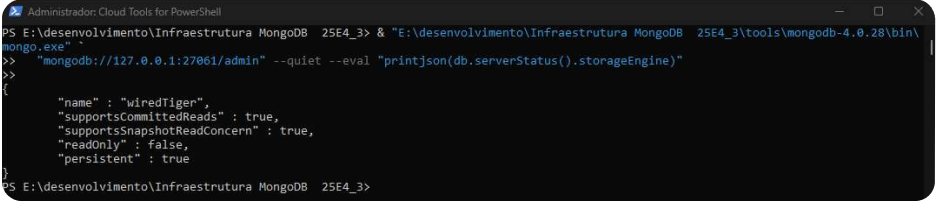
### Comandos (resumo)

```
mongod --port 27061 --dbpath <...> --logpath <...> --logappend --storage
mongo mongodb://127.0.0.1:27061/admin --eval "db.serverStatus().storage"

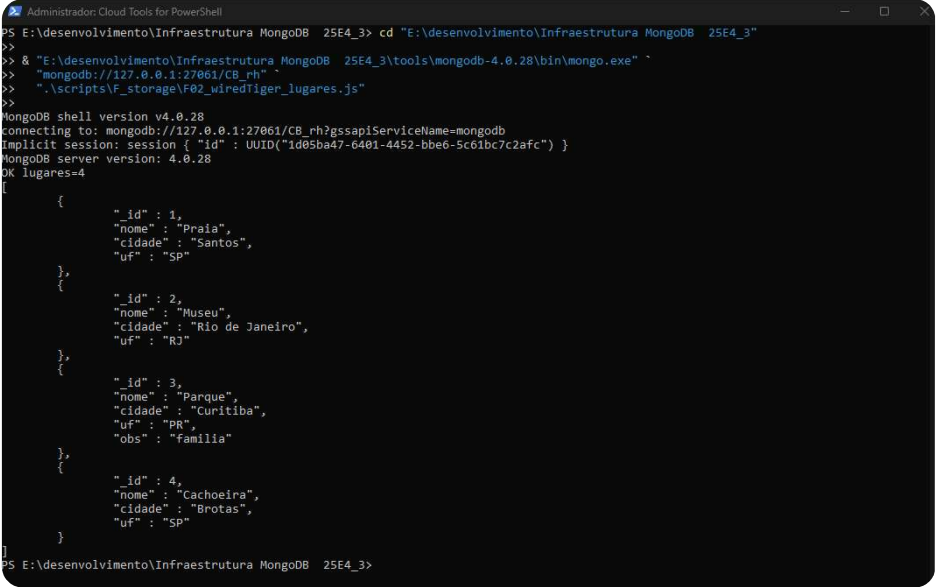
use CB_rh
db.CB_lugares.insertMany([ { ... }, { ... }, { ... }, { ... } ])
```

### Evidências

evidencias/F04\_wiredTiger\_storage\_engine.png  
evidencias/F05\_wiredTiger\_lugares\_insert\_4.png



evidencias/F04\_wiredTiger\_storage\_engine.png



evidencias/F05\_wiredTiger\_lugares\_insert\_4.png



### F3 — Segurança (auth), usuário dba, usuário CB\_desenv, coleção CB\_funcionarios

#### Comandos (resumo)

```
mongod --port 27062 --dbpath <...> --logpath <...> --logappend --auth

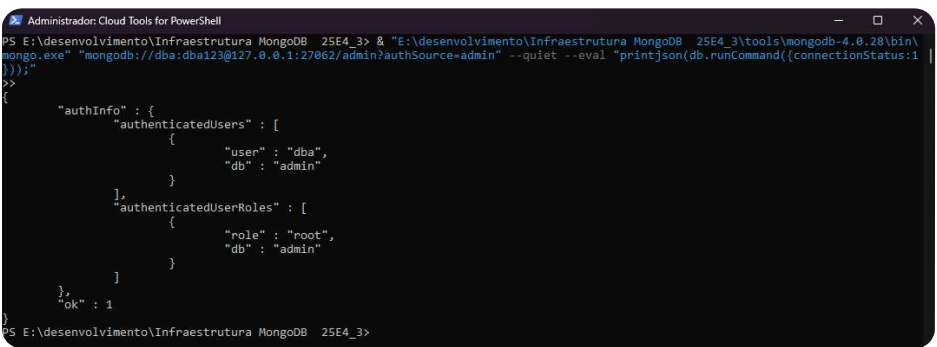
mongo mongodb://127.0.0.1:27062/admin --eval "db.createUser({user:'dba',pwd:'admin',roles:[{role:'root',db:'admin'}]})"

mongo "mongodb://dba:dba123@127.0.0.1:27062/admin" --eval "db.getSiblingDB('CB_rh').createUser({user:'CB_desenv',pwd:'dev123',roles:[{role:'readWrite',db:'CB_rh'}]})"

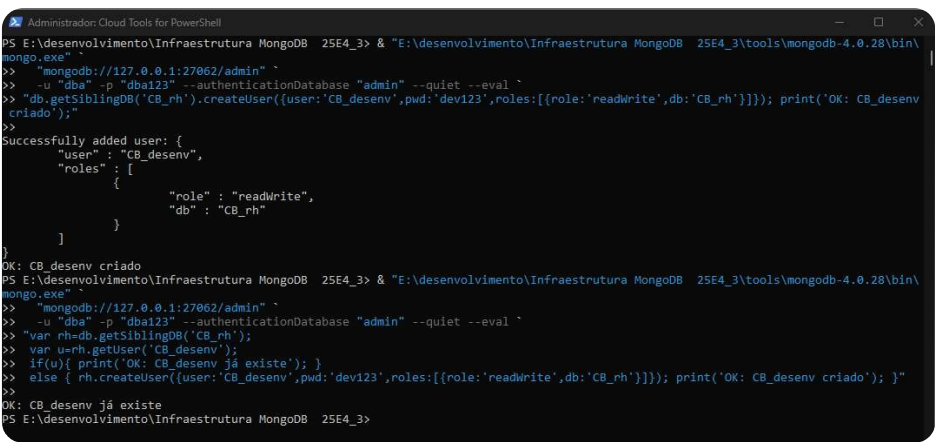
mongo "mongodb://CB_desenv:dev123@127.0.0.1:27062/CB_rh" --eval "db.CB_funcionarios.insertMany([
  { _id:1, nome:'Ana', cargo:'Dev', salario:3500},
  { _id:2, nome:'Bruno', cargo:'DBA', salario:5000, obs:'admin'},
  { _id:3, nome:'Carla', cargo:'Analista', salario:3200},
  { _id:4, nome:'Diego', cargo:'Estagiario', salario:1500}
])"
```

#### Evidências

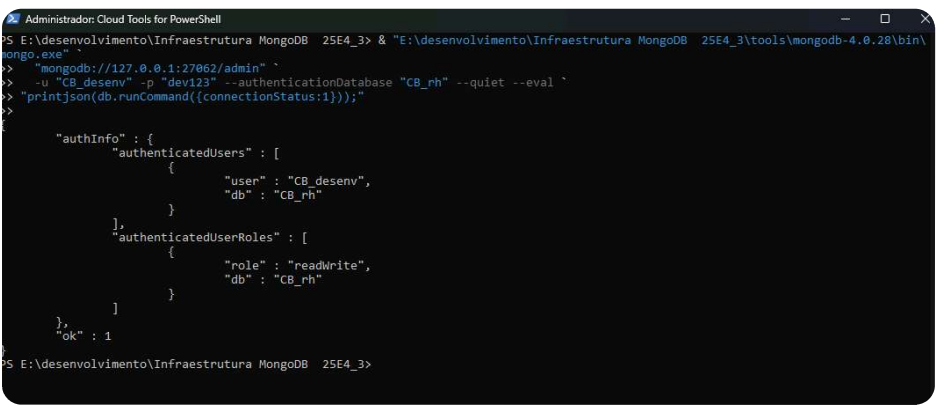
evidencias/F09\_secure\_login\_dba\_connectionStatus.png  
evidencias/F11\_createUser\_CB\_desenv.png  
evidencias/F12\_login\_CB\_desenv.png  
evidencias/F13\_insert\_funcionarios\_4.png



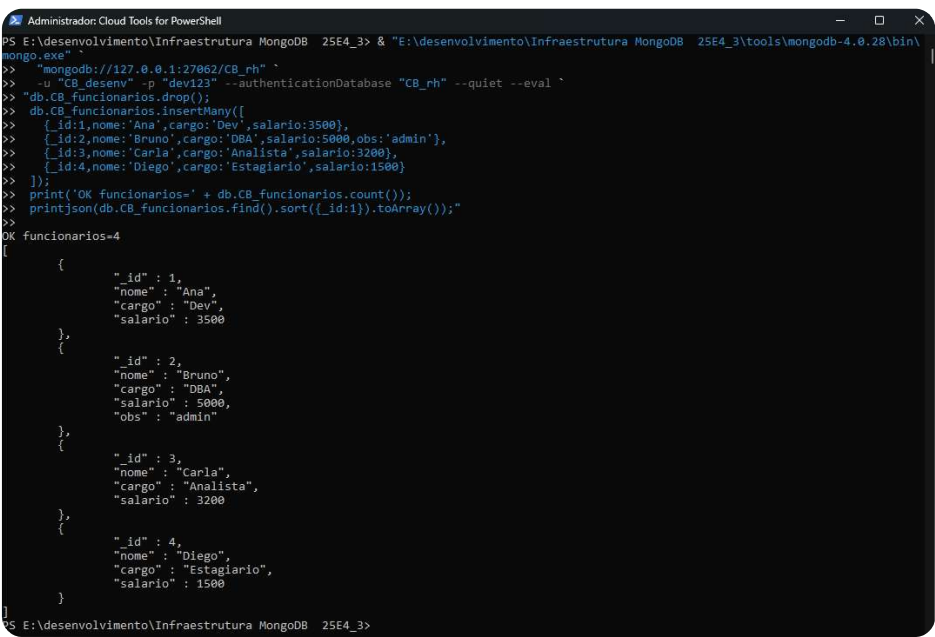
evidencias/F09\_secure\_login\_dba\_connectionStatus.png



evidencias/F11\_createUser\_CB\_desenv.png



evidencias/F12\_login\_CB\_desenv.png



evidencias/F13\_insert\_funcionarios\_4.png

## Parte 2 — Prática (G: Depuração, Backup e Restore)



## G — Backup/kill/delete/restore/validação (DB exerc4b)

### Sequência executada

- Backup (mongodump) da instância
- Finalização do processo (taskkill)
- Exclusão do diretório de dados
- Restore (mongorestore)
- Validação das contagens por coleção

### Comandos (resumo)

```
mongodump --host 127.0.0.1 --port 27070 --out "E:\...\mongo-G\backup"

taskkill /PID <pid> /F

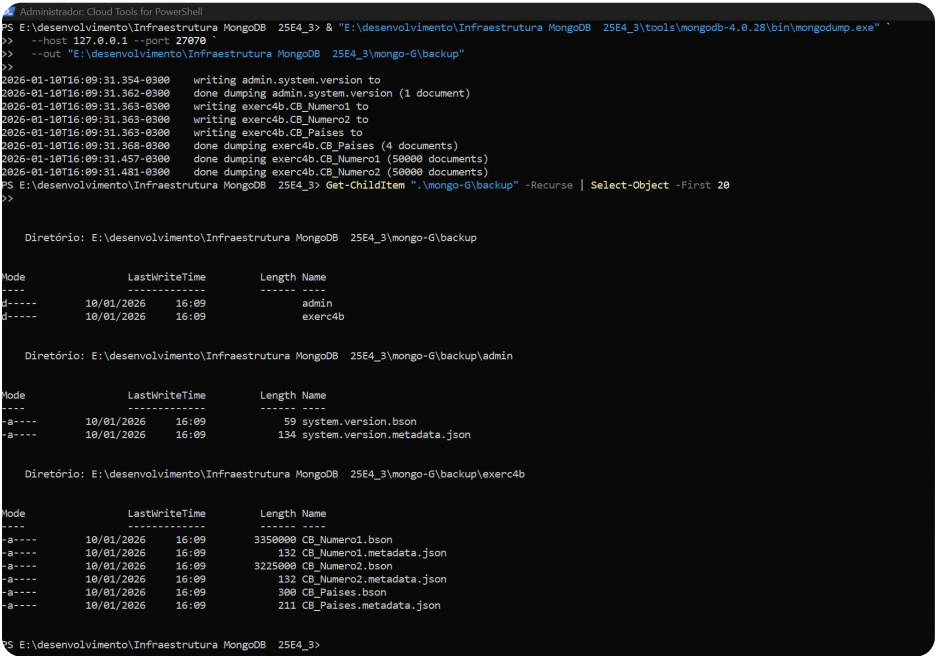
Remove-Item "E:\...\mongo-G\data\*" -Recurse -Force

mongorestore --host 127.0.0.1 --port 27070 --drop "E:\...\mongo-G\backup"

use exerc4b
db.getCollectionNames().forEach(function(c){
    print(c + " => " + db.getCollection(c).count());
});
```

### Evidências exigidas na rubrica

evidencias/G11\_backup\_mongodump.png  
evidencias/G10\_taskkill.png  
evidencias/G11\_apagar\_dados.png  
evidencias/G12\_restore\_ok.png  
evidencias/G13\_contagem\_colecoes\_exerc4b.png



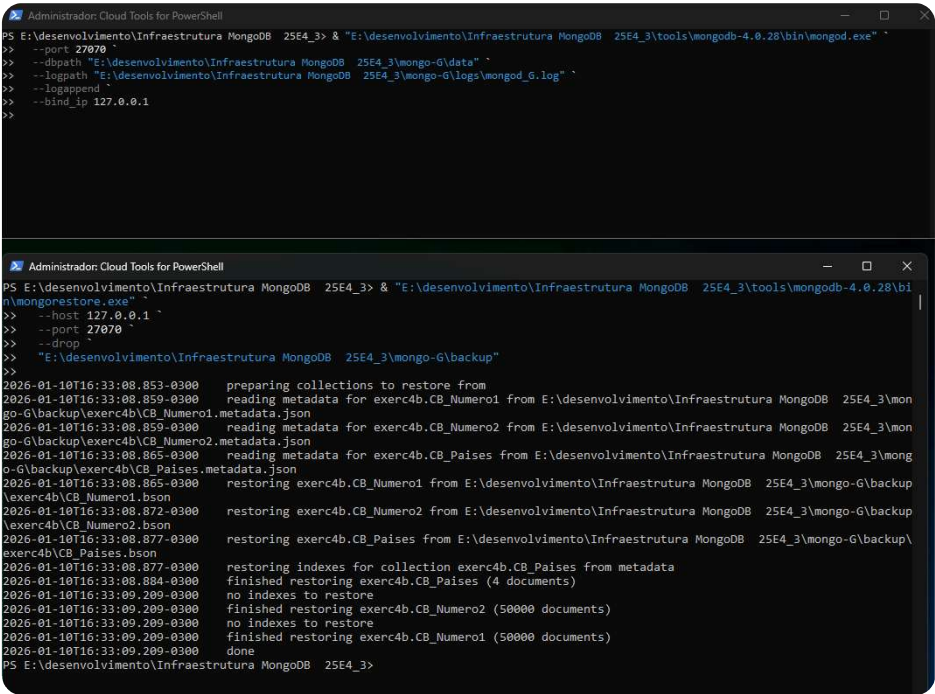
evidencias/G11\_backup\_mongodump.png

### G10 taskkill

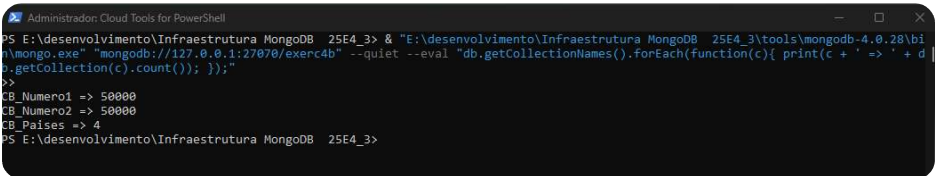
evidencias/G10\_taskkill.png



evidencias/G11\_apagar\_dados.png



evidencias/G12\_restore\_ok.png



evidencias/G13\_contagem\_colecoes\_exerc4b.png



