G. Outras funcionalidades do Cluster que não foram pedidas no trabalho

---

- Leitura distribuída
- Escrita resiliente
- Backup e restore automáticos

Leitura Distribuída no MongoDB

A leitura distribuída permite que consultas sejam feitas em nós secundários do cluster, aliviando o primário e melhorando a performance de leitura.

Conceito

Você pode escolher diferentes estratégias para leitura, usando o `readPreference`:

Preferência	Descrição
PRIMARY_PREFERRED	Lê do primário, mas pode ler de secundários se o primário não estiver disponível
SECONDARY	Sempre lê dos secundários
SECONDARY_PREFERRED	Prefere os secundários, mas pode cair no primário
NEAREST	Lê do nó com menor tempo de resposta (ping)

Exemplo em Mongo Shell

```
db.getMongo().setReadPref("secondary")

db.cliente.find()
```

Você também pode usar `"nearest"`, `"primaryPreferred"` etc.

Escrita Resiliente no MongoDB

A escrita resiliente garante que os dados só sejam confirmados quando forem replicados em nós suficientes, trazendo maior confiabilidade.

O que é Write Concern?

O **write concern** define o nível de garantia exigido ao escrever um documento.

Parâmetros

Parâmetro	Função
w	Número de nós que precisam confirmar a escrita
j	Garante que a escrita seja persistida no journal

Exemplo em Mongo Shell

```
db.cliente.insertOne(
  { codigo: 4, nome: "Diego Leme Pires" },
  { writeConcern: { w: "majority", j: true } }
)
```

Exemplo em Python

```
from pymongo import MongoClient, WriteConcern


client = MongoClient("mongodb://localhost:27017/?replicaSet=replicaSet")
```

```
db = client["sistemaCorp"]

collection = db.get_collection("cliente",
write_concern=WriteConcern(w="majority", j=True))



collection.insert_one({"nome": "Lucas"})
```

O `write_concern` pode ser definido por operação ou por coleção.

Backup Automatizado com Python + `mongodump`

script que permite realizar **backups automáticos** do MongoDB utilizando o comando `mongodump`, com fallback para um caminho alternativo caso o executável não esteja no PATH do sistema. Além disso, ele suporta **agendamento diário** por horário definido.

Visão Geral

- Utiliza `subprocess` para executar o `mongodump`.
- Caso o comando falhe por ausência no PATH, usa o caminho completo do binário.
- Pode ser agendado para executar diariamente via `schedule`.
- Roda em segundo plano com uso de `threading`.

Estrutura das Funções

`executar_backup(backup_path, mongodump_path)`

```
def executar_backup(backup_path, mongodump_path):

    print("🗂️ Tentando executar mongodump padrão...")
```

```
    try:
        subprocess.run(
            ["mongodump", "--host=localhost", "--port=27017", f"--out=
{backup_path}"],
            check=True
        )
        print("Backup concluído com mongodump do sistema.")
    except FileNotFoundError:
        print("mongodump não está no PATH. Tentando com caminho completo...")
        try:
            subprocess.run(
                [mongodump_path, "--host=localhost", "--port=27017", f"--out=
{backup_path}"],
                check=True
            )
            print("Backup concluído com caminho alternativo.")
        except Exception as e:
            print("Falha ao realizar o backup com caminho alternativo.")
            print(f"Erro: {e}")
    except Exception as e:
        print("Falha ao realizar o backup com mongodump padrão.")
        print(f"Erro: {e}")
```

Parâmetros:

- `backup_path`: Caminho onde os dados serão salvos.
- `mongodump_path`: Caminho completo para o executável `mongodump.exe`.

`iniciar_agendador_em_thread(hora, backup_path, mongodump_path)`

```
def iniciar_agendador_em_thread(hora="00:00", backup_path="",
mongodump_path=""):

    def agendar():

        schedule.every().day.at(hora).do(executar_backup, backup_path,
mongodump_path)



        print(f"🔁 Backup agendado para às {hora}. Rodando em segundo
plano...")

        while True:

            schedule.run_pending()

            time.sleep(1)



    thread = threading.Thread(target=agendar, daemon=True)

    thread.start()
```

Parâmetros:

- `hora`: Horário diário no formato `"HH:MM"`.
- `backup_path`: Caminho para onde o backup será salvo.
- `mongodump_path`: Caminho completo até o `mongodump`.

Exemplo de Uso

```
from scripts.BackupAutomatico import iniciar_agendador_em_thread


hora = "23:45"

backup_path = r"C:\Users\lucas\OneDrive\Área de Trabalho\backupTest"

mongodump_path = r"C:\mongodb-tools\mongodump.exe"


iniciar_agendador_em_thread(hora, backup_path, mongodump_path)
```

Dicas

- Verifique se a pasta de destino realmente existe.
- O `mongodump.exe` precisa ser baixado manualmente do site da MongoDB.
- Para que o backup rode diariamente, mantenha o script em execução contínua (em background ou com um serviço).