

University of El Oued

Faculty of Exact Sciences

Department of Computer Science

Lab Report 5: NoSQL Databases

MongoDB Setup & CRUD Operations

Student:

Mohammed Seddik Lifa

Specialization:

Master II: AI & Data Science

Date:

11-10-2025

1 Objective

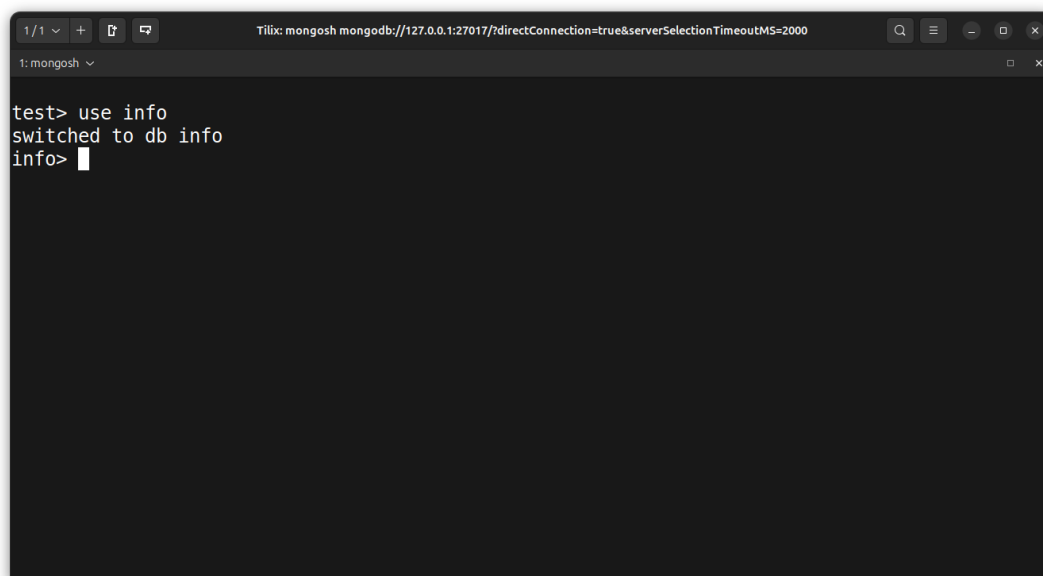
The primary objective of this lab is to install MongoDB, configure the background service, and perform fundamental Create, Read, Update, and Delete (CRUD) operations using the modern mongosh client. This report demonstrates the successful execution of these tasks.

2 Implementation & Proof of Execution

2.1 A. Create a Database

Instruction: Create and switch to a new database named `info` using the command `use info`.

Execution Proof: The shell successfully switched context to the new database.

A screenshot of a terminal window titled 'Tilix: mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000'. The terminal shows the prompt 'test>' followed by the command 'use info'. The output is 'switched to db info' and the prompt changes to 'info>'.

```
1/1 + [ ] [ ]
Tilix: mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
t: mongosh
test> use info
switched to db info
info>
```

Figure 1: Command execution: Switching to database 'info'

2.2 B. Create a Collection (Insert Documents)

Instruction: Insert a dataset into the `produits` collection. The dataset includes details (manufacturer, price, and options) for a Macbook Pro, Macbook Air, and Thinkpad X230.

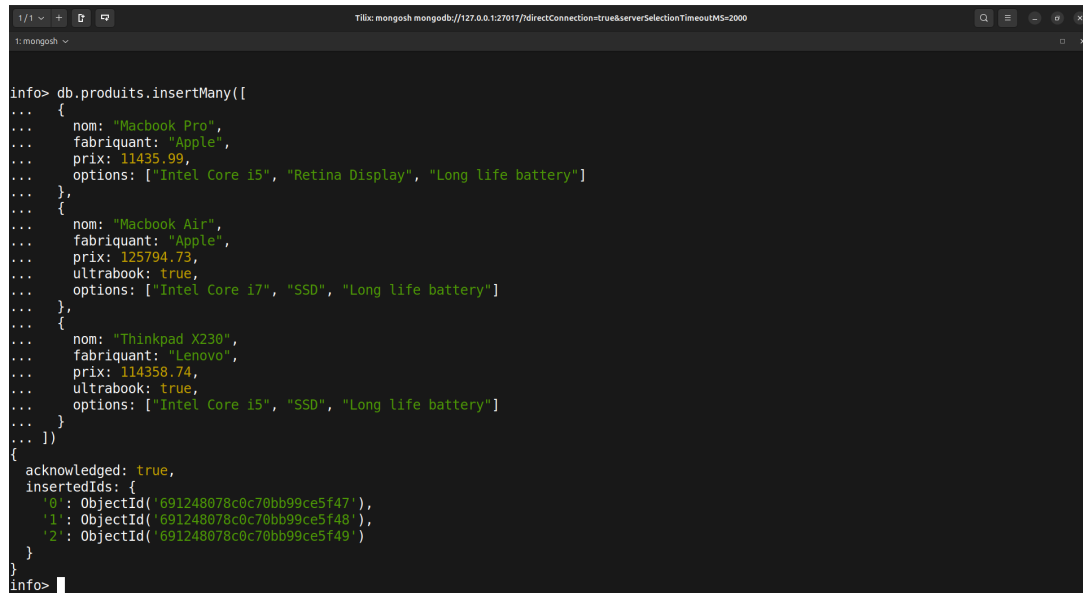
Code Pattern Executed:

```
1 db.produits.insertMany([
2   { nom: "Macbook_Pro", fabricant: "Apple", prix: 11435.99 ... },
3   { nom: "Macbook_Air", fabricant: "Apple", ultrabook: true ... },
4   { nom: "Thinkpad_X230", fabricant: "Lenovo", ultrabook: true ... }])
```

5 | 1)

Listing 1: Insertion Query Structure

Execution Proof: The output shows acknowledged: true and three generated ObjectIDs.

A screenshot of a MongoDB terminal window. The terminal shows a command to insert three documents into the 'produits' collection. The documents are for 'Macbook Pro', 'Macbook Air', and 'Thinkpad X230'. The output shows that the insertion was successful, with 'acknowledged: true' and three 'insertedIds' provided as ObjectIds.

```
1/1 ~ + [?] [?]
Title: mongosh mongodbs://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
1: mongosh

info> db.produits.insertMany([
...   {
...     nom: "Macbook Pro",
...     fabricant: "Apple",
...     prix: 11435.99,
...     options: ["Intel Core i5", "Retina Display", "Long life battery"]
...   },
...   {
...     nom: "Macbook Air",
...     fabricant: "Apple",
...     prix: 125794.73,
...     ultrabook: true,
...     options: ["Intel Core i7", "SSD", "Long life battery"]
...   },
...   {
...     nom: "Thinkpad X230",
...     fabricant: "Lenovo",
...     prix: 114358.74,
...     ultrabook: true,
...     options: ["Intel Core i5", "SSD", "Long life battery"]
...   }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('691248078c0c70bb99ce5f47'),
    '1': ObjectId('691248078c0c70bb99ce5f48'),
    '2': ObjectId('691248078c0c70bb99ce5f49')
  }
}
info> |
```

Figure 2: Successful insertion of 3 documents with ObjectIDs

2.3 C. Read (Query) Documents

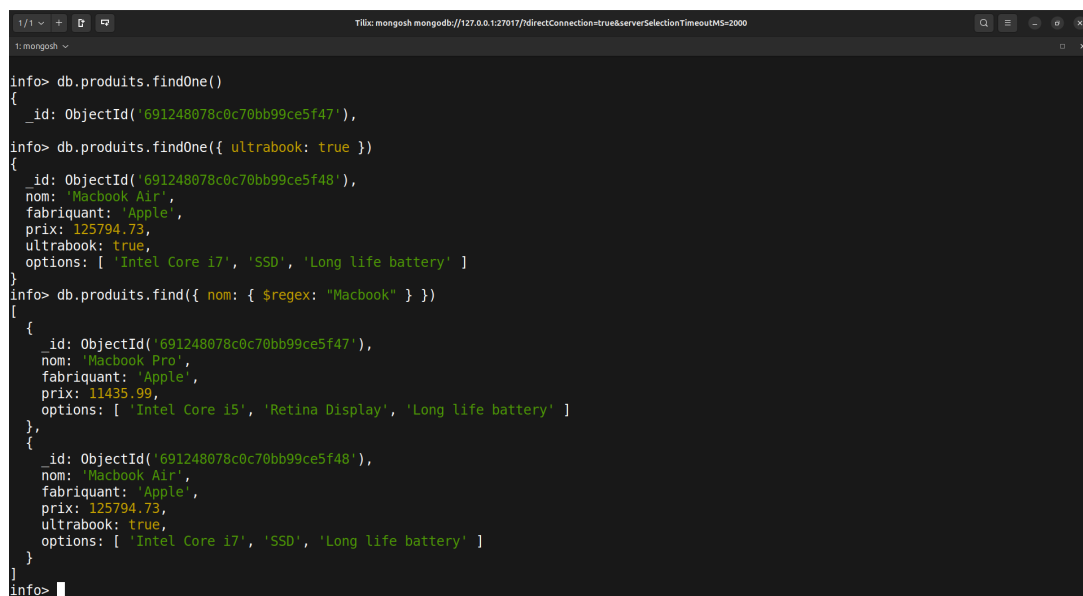
Instruction: Perform various queries to retrieve data. Specifically:

1. Retrieve a single document.
2. Filter documents where ultrabook is true.
3. Use Regex to find products starting with "Macbook".

Queries Run:

```
1 db.produits.findOne()
2 db.produits.findOne({ ultrabook: true })
3 db.produits.find({ nom: { $regex: "Macbook" } })
```

Execution Proof:



```
Tiix: mongosh mongod://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
1: mongosh
info> db.produits.findOne()
{
  _id: ObjectId('691248078c0c70bb99ce5f47'),
}
info> db.produits.findOne({ ultrabook: true })
{
  _id: ObjectId('691248078c0c70bb99ce5f48'),
  nom: 'Macbook Air',
  fabricant: 'Apple',
  prix: 125794.73,
  ultrabook: true,
  options: [ 'Intel Core i7', 'SSD', 'Long life battery' ]
}
info> db.produits.find({ nom: { $regex: "Macbook" } })
[
  {
    _id: ObjectId('691248078c0c70bb99ce5f47'),
    nom: 'Macbook Pro',
    fabricant: 'Apple',
    prix: 11435.99,
    options: [ 'Intel Core i5', 'Retina Display', 'Long life battery' ]
  },
  {
    _id: ObjectId('691248078c0c70bb99ce5f48'),
    nom: 'Macbook Air',
    fabricant: 'Apple',
    prix: 125794.73,
    ultrabook: true,
    options: [ 'Intel Core i7', 'SSD', 'Long life battery' ]
  }
]
info>
```

Figure 3: Query Results: findOne, Boolean filtering, and Regex matching

2.4 D. Delete Documents

Instruction: Remove specific data from the collection:

- Delete all documents where manufacturer is "Apple".
- Delete a specific document by its unique _id.

Queries Run:

```
1 // Delete all Apple products
2 db.produits.deleteMany({ fabricant: "Apple" })
3
```

```
4 // Delete specific ID (Thinkpad)
5 db.produits.deleteOne({ _id: thinkpad_id })
6
7 // Verify Empty Collection
8 db.produits.find()
```

Execution Proof: The output confirms deletedCount: 2 for Apple products and deletedCount: 1 for the Thinkpad. The final find command returns nothing, confirming the collection is empty.

A screenshot of a MongoDB shell session. The window title is 'Tilix: mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000'. The prompt is '1: mongosh >'. The user enters 'db.produits.deleteMany({ fabricant: "Apple" })', and the output is '{ acknowledged: true, deletedCount: 2 }'. The user then enters 'db.produits.deleteOne({ _id: thinkpad_id })', and the output is '{ acknowledged: true, deletedCount: 1 }'. The user enters 'info>' and the output is 'info>'. The user enters 'db.produits.find()' and the output is 'info>'. The user enters 'info>' and the output is 'info>'.

```
1: mongosh > db.produits.deleteMany({ fabricant: "Apple" })
{ acknowledged: true, deletedCount: 2 }
1: mongosh > db.produits.deleteOne({ _id: thinkpad_id })
{ acknowledged: true, deletedCount: 1 }
1: mongosh > info>
info>
1: mongosh > db.produits.find()
info>
1: mongosh > info>
```

Figure 4: Deletion operations and final verification