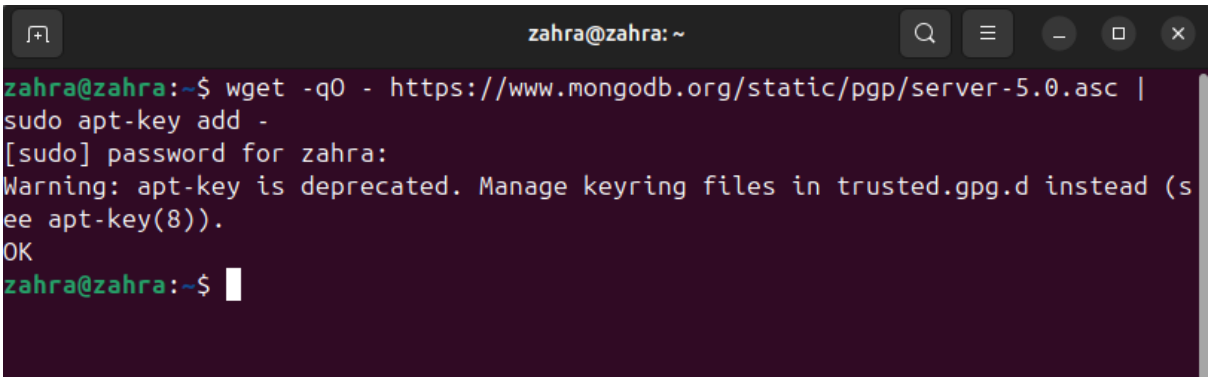
TP 1 : Installation et Manipulation de la BD NoSQL « MongoDB »

Binôme : Imane TOUIBA
Zahra KASMOUTI

1) Installation et configuration de MongoDB

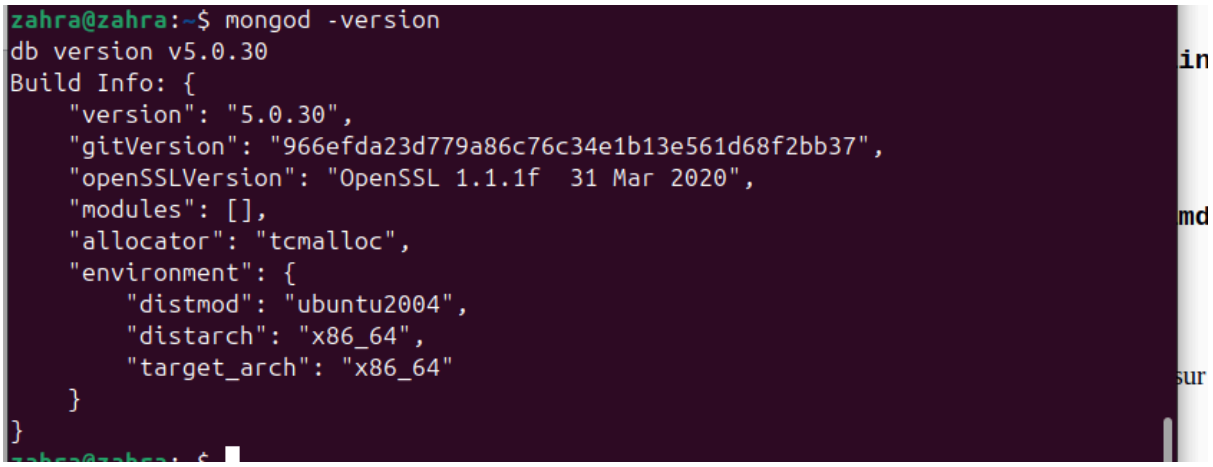
installation et configuration :

Étape 1 - Configuration du « apt Repository » :



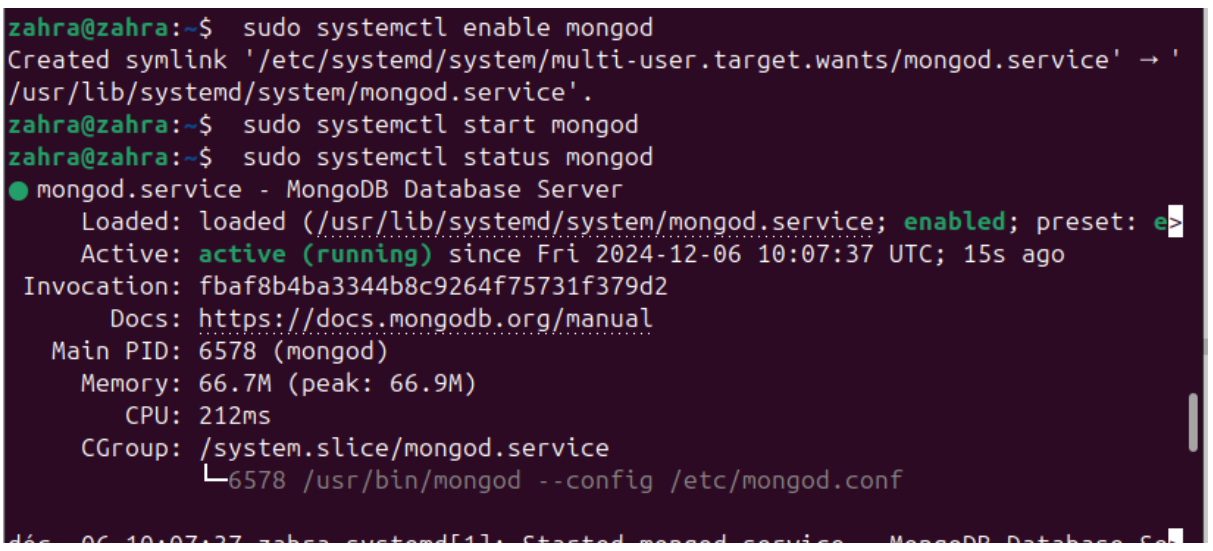
Étape 2 - Installation de mongoDB :

On a bien installé mongoDB v5.0.30



Gestion des services mongoDB :

Activation et démarrage du service MongoDB :



Le statut de MongoDB est bien active

Teste de la configuration :

```
> use mydb
switched to db mydb
> db.test.save( { kasmoutiTouiba: 100 } )
WriteResult({ "nInserted" : 1 })
> db.test.find()
{ "_id" : ObjectId("6752cef41e8f53b14f62c051"), "kasmoutiTouiba" : 100 }
>
```

2) Examination des requêtes MongoDB :

Restauration d'un fichier .bson :

```
zahra@zahra:~/Downloads$ mongorestore -d cinema -c films movieDetails.bson
2024-12-06T10:22:27.144+0000    checking for collection data in movieDetails.bso
n
2024-12-06T10:22:27.156+0000    restoring cinema.films from movieDetails.bson
2024-12-06T10:22:27.199+0000    finished restoring cinema.films (2295 documents,
 0 failures)
2024-12-06T10:22:27.199+0000    2295 document(s) restored successfully. 0 docume
nt(s) failed to restore.
zahra@zahra:~/Downloads$
```

Quelques testes sur la base de donnée crée

```
> db.films.findOne();
{
        "_id" : ObjectId("569190ca24de1e0ce2dfcd4f"),
        "title" : "Once Upon a Time in the West",
        "year" : 1968,
        "rated" : "PG-13",
        "released" : ISODate("1968-12-21T05:00:00Z"),
        "runtime" : 175,
        "countries" : [
                "Italy",
                "USA",
                "Spain"
        ],
        "genres" : [
                "Western"
        ],
        "director" : "Sergio Leone",
        "writers" : [
                "Sergio Donati",
                "Sergio Leone",
                "Dario Argento",
                "Bernardo Bertolucci",
```

Rhythmbox

Gestion des indexes :

1- Affichage des indexes :

```
> db.films.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
>
```

2- Création d'indexes :

```
> db.films.createIndex({"genre": 1}, {"sparse": true})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
>
```

3- Suppression des indexes :

```
> db.films.dropIndex("genre_1");
{ "nIndexesWas" : 2, "ok" : 1 }
>
```

lorsqu'on recherche un document donné, il est possible de connaître la stratégie effective permettant à MongoDB de le retrouver grâce à la commande **explain().**

```
> db.films.find({"actors":"Bruce Willis"}).explain()
{
        "explainVersion" : "1",
        "queryPlanner" : {
                "namespace" : "cinema.films",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "actors" : {
                                "$eq" : "Bruce Willis"
                        }
                },
                "queryHash" : "592671A4",
                "planCacheKey" : "70938073",
                "maxIndexedOrSolutionsReached" : false,
                "maxIndexedAndSolutionsReached" : false,
                "maxScansToExplodeReached" : false,
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "actors" : {
                                        "$eq" : "Bruce Willis"
                                }
```

Recherche de documents :

Utilisation d'un filtre simple

```
> db.films.find({"year":2012});
{ "_id" : ObjectId("569190cc24de1e0ce2dfcd58"), "title" : "West of Memphis", "ye
ar" : 2012, "rated" : "R", "released" : ISODate("2012-11-22T05:00:00Z"), "runtim
e" : 147, "countries" : [ "New Zealand", "USA" ], "genres" : [ "Documentary" ],
"director" : "Amy Berg", "writers" : [ "Amy Berg", "Billy McMillin" ], "actors"
: [ "Michael Baden", "Jason Baldwin", "Holly Ballard", "Jamie Clark Ballard" ],
"plot" : "An examination of a failure of justice in the case against the West Me
mphis Three.", "poster" : "http://ia.media-imdb.com/images/M/MV5BMjIzNDM3NjkzOV5
BMl5BanBnXkFtZTcwNjI5Nzg0OA@@._V1_SX300.jpg", "imdb" : { "id" : "tt2130321", "ra
ting" : 7.9, "votes" : 6627 }, "tomato" : { "meter" : 95, "image" : "certified",
 "rating" : 7.9, "reviews" : 111, "fresh" : 106, "consensus" : "Both a sobering
look at a true crime story and a scathing indictment of the American justice sys
tem, West of Memphis is a real-life horror story told with fury and compassion."
, "userMeter" : 87, "userRating" : 4.1, "userReviews" : 8482 }, "metacritic" : 8
0, "awards" : { "wins" : 0, "nominations" : 9, "text" : "Nominated for 1 BAFTA F
ilm Award. Another 9 nominations." }, "type" : "movie" }
{ "_id" : ObjectId("5692a13e24de1e0ce2dfcec7"), "title" : "HOUBA! On the Trail o
```

Pour améliorer l'affichage, on peut utiliser la commande pretty.

```
> db.films.find({"year":2012}).pretty()
{
        "_id" : ObjectId("569190cc24de1e0ce2dfcd58"),
        "title" : "West of Memphis",
        "year" : 2012,
        "rated" : "R",
        "released" : ISODate("2012-11-22T05:00:00Z"),
        "runtime" : 147,
        "countries" : [
                "New Zealand",
                "USA"
        ],
        "genres" : [
                "Documentary"
        ],
        "director" : "Amy Berg",
        "writers" : [
                "Amy Berg",
```

Projections :

```
> db.films.find({"year":2012},{"title":1,"year":1,"actors":1}).pretty()
{
        "_id" : ObjectId("569190cc24de1e0ce2dfcd58"),
        "title" : "West of Memphis",
        "year" : 2012,
        "actors" : [
                "Michael Baden",
                "Jason Baldwin",
                "Holly Ballard",
                "Jamie Clark Ballard"
        ]
}
{
        "_id" : ObjectId("5692a13e24de1e0ce2dfcec7"),
        "title" : "HOUBA! On the Trail of the Marsupilami",
        "year" : 2012,
        "actors" : [
                "Jamel Debbouze",
                "Alain Chabat",
                "Fred Testot",
                "Lambert Wilson"
        ]
}
{
```

```
> db.films.find({"year":2012},{"title":1,"year":1,"actors":1, _id:0}).pretty()
{
        "title" : "West of Memphis",
        "year" : 2012,
        "actors" : [
                "Michael Baden",
                "Jason Baldwin",
                "Holly Ballard",
                "Jamie Clark Ballard"
        ]
}
{
        "title" : "HOUBA! On the Trail of the Marsupilami",
        "year" : 2012,
        "actors" : [
                "Jamel Debbouze",
                "Alain Chabat",
                "Fred Testot",
                "Lambert Wilson"
        ]
```

Recherche dans un tableau :

```
> db.films.find({"actors":"Leonardo DiCaprio"},{"title":1,"year":1,"actors":1, _
id:0}).pretty()
{
        "title" : "Shutter Island",
        "year" : 2010,
        "actors" : [
                "Leonardo DiCaprio",
                "Mark Ruffalo",
                "Ben Kingsley",
                "Max von Sydow"
        ]
}
{
        "title" : "Catch Me If You Can",
        "year" : 2002,
        "actors" : [
                "Leonardo DiCaprio",
                "Tom Hanks",
                "Christopher Walken",
                "Martin Sheen"
        ]
```

```
> db.films.find({"actors":"Leonardo DiCaprio", "year":2002},
... {"title":1,"year":1,"actors":1, _id:0}).pretty()
{
        "title" : "Catch Me If You Can",
        "year" : 2002,
        "actors" : [
                "Leonardo DiCaprio",
                "Tom Hanks",
                "Christopher Walken",
                "Martin Sheen"
        ]
}
```

```
> db.films.find({"actors":{$in:["Leonardo DiCaprio", "Tom Hanks"]}},{"title":1,"
year":1,"actors":1, _id:0}).pretty()
{
        "title" : "Shutter Island",
        "year" : 2010,
        "actors" : [
                "Leonardo DiCaprio",
                "Mark Ruffalo",
                "Ben Kingsley",
                "Max von Sydow"
        ]
}
{
        "title" : "Big",
        "year" : 1988,
        "actors" : [
                "Tom Hanks",
                "Elizabeth Perkins",
                "Robert Loggia",
                "John Heard"
        ]
}
{
        "title" : "Toy Story",
```

```
> db.films.find({"actors":{$all:["Leonardo DiCaprio", "Tom Hanks"]}},{"title":1,
"year":1,"actors":1, _id:0}).pretty()
{
        "title" : "Catch Me If You Can",
        "year" : 2002,
        "actors" : [
                "Leonardo DiCaprio",
                "Tom Hanks",
                "Christopher Walken",
                "Martin Sheen"
        ]
```

Recherche avancée :

```
> db.films.findOne({},{title:1,year:1,awards:1,_id:0})
{
        "title" : "Once Upon a Time in the West",
        "year" : 1968,
        "awards" : {
                "wins" : 4,
                "nominations" : 5,
                "text" : "4 wins & 5 nominations."
        }
}
```

```
> db.films.find({"awards.wins":7},
... {title:1,year:1,awards:1,_id:0}).pretty()
{
        "title" : "How the West Was Won",
        "year" : 1962,
        "awards" : {
                "wins" : 7,
                "nominations" : 5,
                "text" : "Won 3 Oscars. Another 7 wins & 5 nominations."
        }
}
{
        "title" : "Brazilian Western",
        "year" : 2013,
        "awards" : {
                "wins" : 7,
                "nominations" : 6,
                "text" : "7 wins & 6 nominations."
        }
}
```

```
> db.films.find({"awards.wins":{$ne:0}},
... {title:1,year:1,awards:1,_id:0}).pretty()
{
        "title" : "Once Upon a Time in the West",
        "year" : 1968,
        "awards" : {
                "wins" : 4,
                "nominations" : 5,
                "text" : "4 wins & 5 nominations."
        }
}
{
        "title" : "Wild Wild West",
        "year" : 1999,
        "awards" : {
                "wins" : 10,
                "nominations" : 11,
                "text" : "10 wins & 11 nominations."
        }
```

```
> db.films.find({"awards.wins":{$gte:80}},
... {title:1,year:1,awards:1,_id:0}).pretty()
{
        "title" : "Beasts of the Southern Wild",
        "year" : 2012,
        "awards" : {
                "wins" : 91,
                "nominations" : 119,
                "text" : "Nominated for 4 Oscars. Another 91 wins & 119 nominati
ons."
        }
}
{
        "title" : "Lost in Translation",
        "year" : 2003,
        "awards" : {
                "wins" : 90,
                "nominations" : 100,
                "text" : "Won 1 Oscar. Another 90 wins & 100 nominations."
        }
}
{
        "title" : "The Tree of Life"
```

Trier les résultats :

```
> db.films.find({"awards.wins":{$gte:80}},
... {title:1,year:1,awards:1,_id:0}).sort({"awards.wins":-
... 1}).pretty()
{
        "title" : "No Country for Old Men",
        "year" : 2007,
        "awards" : {
                "wins" : 148,
                "nominations" : 122,
                "text" : "Won 4 Oscars. Another 148 wins & 122 nominations."
        }
}
{
        "title" : "There Will Be Blood",
        "year" : 2007,
        "awards" : {
                "wins" : 103,
                "nominations" : 123,
                "text" : "Won 2 Oscars. Another 103 wins & 123 nominations."
        }
}
```

Insertion des documents dans mongoDB :

```
> use test
switched to db test
> db.test.insertOne({"name":"zahra"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("67533cc6fc53fc9410ab9c58")
}
>
```

```
> db.test.find()
{ "_id" : ObjectId("67533cc6fc53fc9410ab9c58"), "name" : "zahra" }
>
```

```
> db.test.insertOne({"name":"zahra"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("67533dbdfc53fc9410ab9c59")
}
> db.test.find()
{ "_id" : ObjectId("67533cc6fc53fc9410ab9c58"), "name" : "zahra" }
{ "_id" : ObjectId("67533dbdfc53fc9410ab9c59"), "name" : "zahra" }
>
```

```
> db.test.insertOne({_id:0,"name":"zahira"})
{ "acknowledged" : true, "insertedId" : 0 }
>
```

```
> db.test.insertOne({_id:0,"name":"imane"})
WriteError({
        "index" : 0,
        "code" : 11000,
        "errmsg" : "E11000 duplicate key error collection: test.test index: _id_
 dup key: { _id: 0.0 }",
        "op" : {
                "_id" : 0,
                "name" : "imane"
        }
}) :
WriteError({
        "index" : 0,
        "code" : 11000,
        "errmsg" : "E11000 duplicate key error collection: test.test index: _id_
 dup key: { _id: 0.0 }",
        "op" : {
                "_id" : 0,
                "name" : "imane"
        }
})
WriteError@src/mongo/shell/bulk_api.js:465:48
```

```
> db.test.insertOne({"name":"toto","age":35})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("67533ed3fc53fc9410ab9c5a")
}
> db.test.find()
{ "_id" : ObjectId("67533cc6fc53fc9410ab9c58"), "name" : "zahra" }
{ "_id" : ObjectId("67533dbdfc53fc9410ab9c59"), "name" : "zahra" }
{ "_id" : 0, "name" : "zahira" }
{ "_id" : ObjectId("67533ed3fc53fc9410ab9c5a"), "name" : "toto", "age" : 35 }
>
```

```
> db.test.insert([{"name":"mohamed"},{"name": "othman", "ville": ["casablanca","
rabat"]}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Suppression des documents dans mongoDB :

```
> db.test.drop()
true
> db.test.find()
>
```

```
> db.test.insertOne({"name":"zahra"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("6753413efc53fc9410ab9c5f")
}
> db.test.insertOne({"name":"zahra"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("67534140fc53fc9410ab9c60")
}
```

```
> db.test.find({"_id":ObjectId("67534140fc53fc9410ab9c60")})
{ "_id" : ObjectId("67534140fc53fc9410ab9c60"), "name" : "zahra" }
> db.test.deletOne({"_id":ObjectId("67534140fc53fc9410ab9c60")})
uncaught exception: TypeError: db.test.deletOne is not a function :
@(shell):1:1
> db.test.deleteOne({"_id":ObjectId("67534140fc53fc9410ab9c60")})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

MAJ des documents dans mongoDB :

1- Ajouter ou remplacer un champ existant avec $set

```
> db.test.update({name:"zahra"},{$set:{ville:"rabat"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.test.find()
{ "_id" : ObjectId("6753413efc53fc9410ab9c5f"), "name" : "zahra", "ville" : "rab
at" }
>
```

```
> db.test.find()
{ "_id" : ObjectId("6753413efc53fc9410ab9c5f"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("675343fafc53fc9410ab9c61"), "name" : "mohamed", "age" : 40 }
{ "_id" : ObjectId("6753441afc53fc9410ab9c62"), "name" : "zaidouni", "ville" : "
casablanca" }
{ "_id" : ObjectId("675344a7fc53fc9410ab9c63"), "name" : "zahra" }
>
```

```
> db.test.update({name:"zahra"},{$set:{ville:"casablanca"}}, {multi:true})
WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 1 })
> db.test.find()
{ "_id" : ObjectId("6753413efc53fc9410ab9c5f"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("675343fafc53fc9410ab9c61"), "name" : "mohamed", "age" : 40 }
{ "_id" : ObjectId("6753441afc53fc9410ab9c62"), "name" : "zaidouni", "ville" : "
casablanca" }
{ "_id" : ObjectId("675344a7fc53fc9410ab9c63"), "name" : "zahra", "ville" : "cas
ablanca" }
>
```

2- Incrémenter un champ numérique existant avec $inc

```
> db.test.update({name:"mohamed"},{$inc:{age:1} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.test.find()
{ "_id" : ObjectId("6753413efc53fc9410ab9c5f"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("675343fafc53fc9410ab9c61"), "name" : "mohamed", "age" : 41 }
{ "_id" : ObjectId("6753441afc53fc9410ab9c62"), "name" : "zaidouni", "ville" : "
casablanca" }
{ "_id" : ObjectId("675344a7fc53fc9410ab9c63"), "name" : "zahra", "ville" : "cas
ablanca" }
>
```

3 - MAJ du tableau avec $push ou pull

```
> db.test.insert({"name": "othman", "ville":
... ["casablanca","rabat"]})
WriteResult({ "nInserted" : 1 })
> db.test.update({"name": "othman"}, {$push:{ville:"tanger"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.test.find()
{ "_id" : ObjectId("6753413efc53fc9410ab9c5f"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("675343fafc53fc9410ab9c61"), "name" : "mohamed", "age" : 44 }
{ "_id" : ObjectId("6753441afc53fc9410ab9c62"), "name" : "zaidouni", "ville" : "
casablanca" }
{ "_id" : ObjectId("675344a7fc53fc9410ab9c63"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("6753471bfc53fc9410ab9c64"), "name" : "othman", "ville" : [ "
casablanca", "rabat", "tanger" ] }
>
```

```
> db.test.update({"name": "othman"}, {$pull:{ville:"rabat"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.test.find()
{ "_id" : ObjectId("6753413efc53fc9410ab9c5f"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("675343fafc53fc9410ab9c61"), "name" : "mohamed", "age" : 44 }
{ "_id" : ObjectId("6753441afc53fc9410ab9c62"), "name" : "zaidouni", "ville" : "
casablanca" }
{ "_id" : ObjectId("675344a7fc53fc9410ab9c63"), "name" : "zahra", "ville" : "cas
ablanca" }
{ "_id" : ObjectId("6753471bfc53fc9410ab9c64"), "name" : "othman", "ville" : [ "
casablanca", "tanger" ] }
>
```

3)Implémentation d'une application avec Node.js et MongoDB et réalisation des opérations CRUD

node.js installé :

```
zahra@zahra:~/Downloads$ node -v
v20.10.0
zahra@zahra:~/Downloads$
```

npm init :

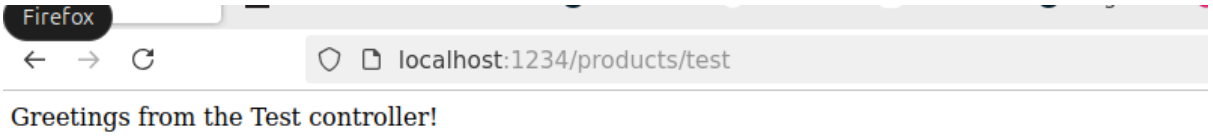
```
package name: (productsapp)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/zahra/Documents/productsapp/package.json:

{
  "name": "productsapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
```

lancement de serveur node :

```
zahra@zahra:~/Documents/productsapp$ node app.js
Server is up and running on port number 1234
```

test de l'application :

```
zahra@zahra:~/Documents/productsapp$ node app.js
Server is up and running on port number 1234
```

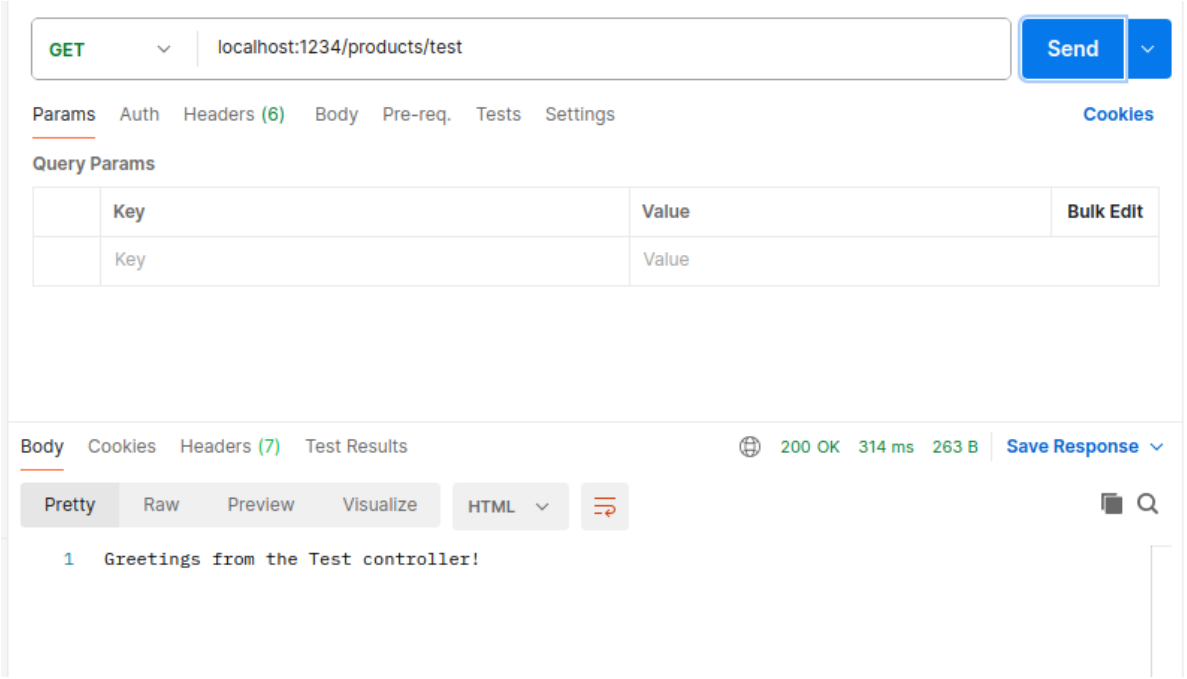


Greetings from the Test controller!

installation de postman :

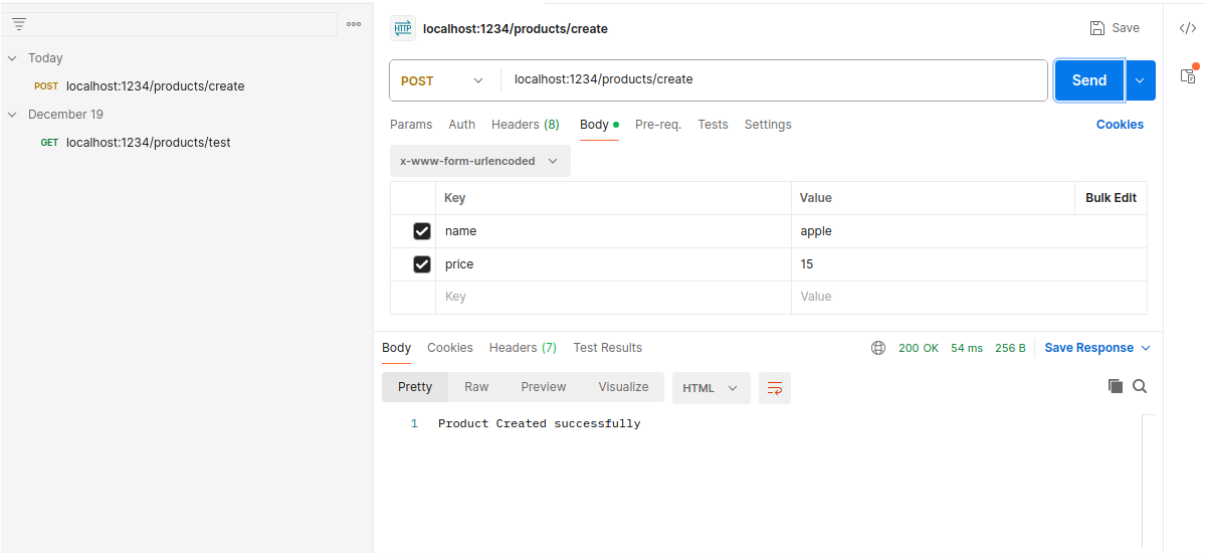
```
zahra@zahra:~/Documents/productsapp$ sudo snap install postman
[sudo] password for zahra:
postman (v11/stable) 11.23.3 from Postman, Inc. (postman-inc✓) installed
zahra@zahra:~/Documents/productsapp$
```

connection à l’application avec postman :



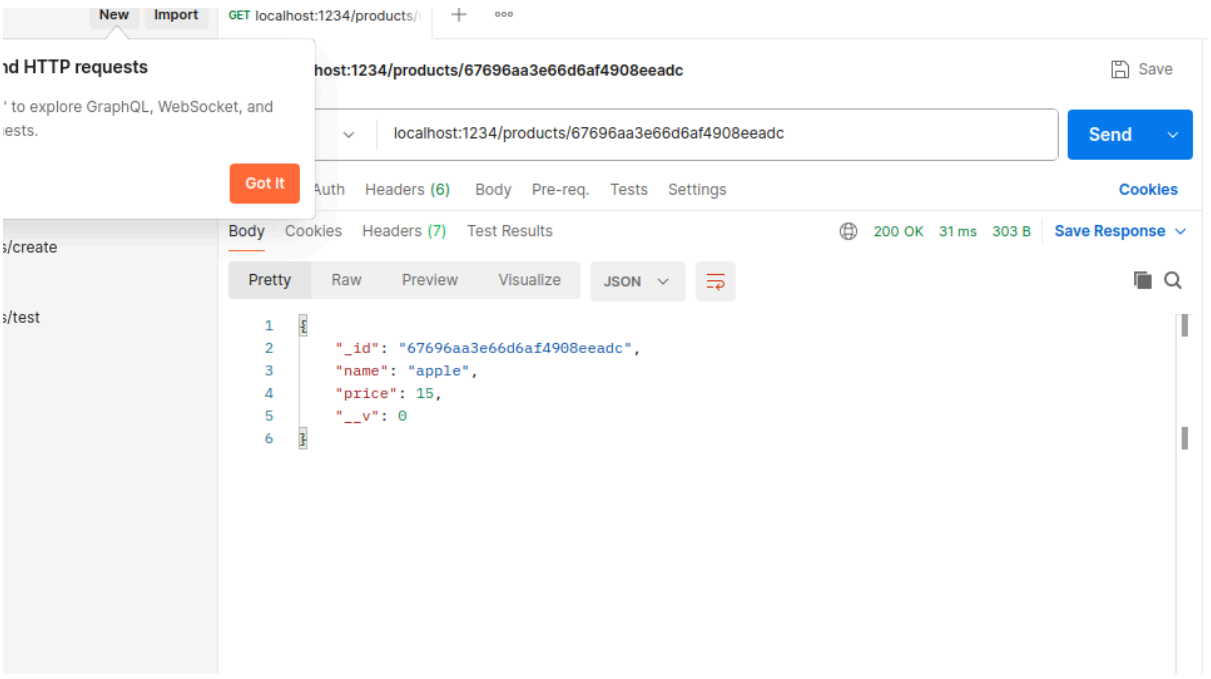
Implémentation des endpoints :

CREATE

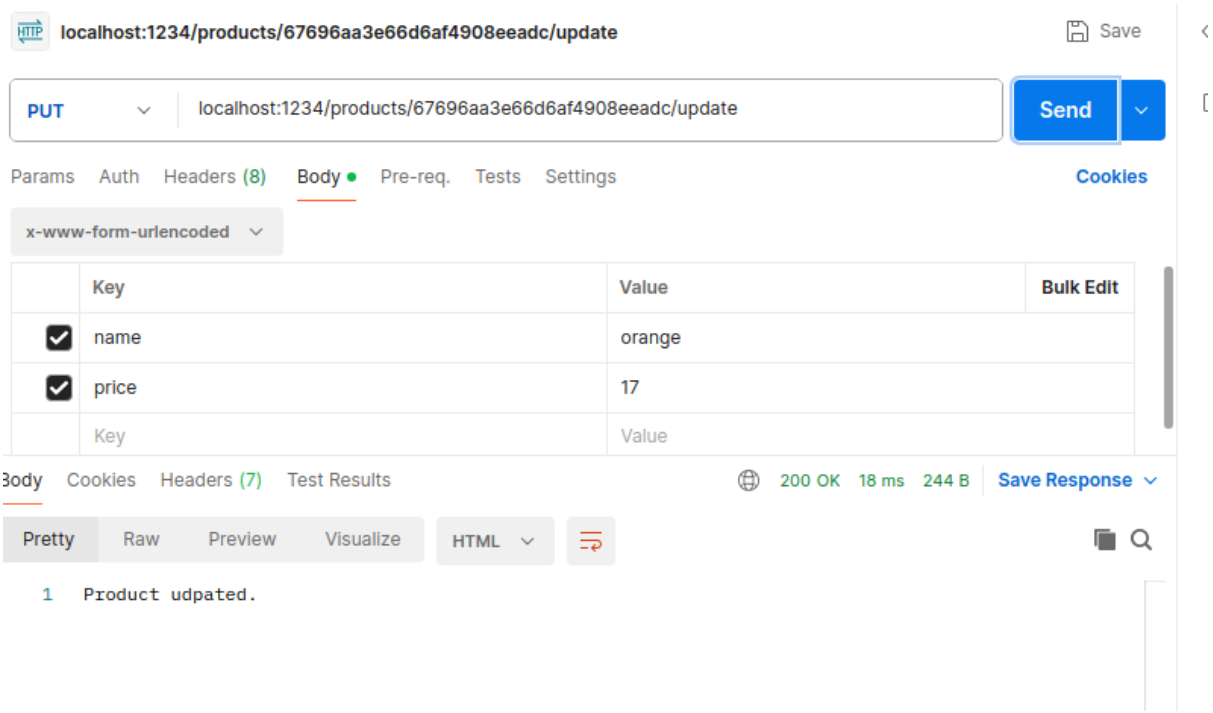


```
> show dbs
admin        0.000GB
cinema       0.001GB
config       0.000GB
local        0.000GB
mydb         0.000GB
test         0.000GB
tp_database  0.000GB
> use tp_database
switched to db tp_database
> db.products.find()
{ "_id" : ObjectId("67696aa3e66d6af4908eeadc"), "name" : "apple", "price" : 15,
"__v" : 0 }
>
```

READ

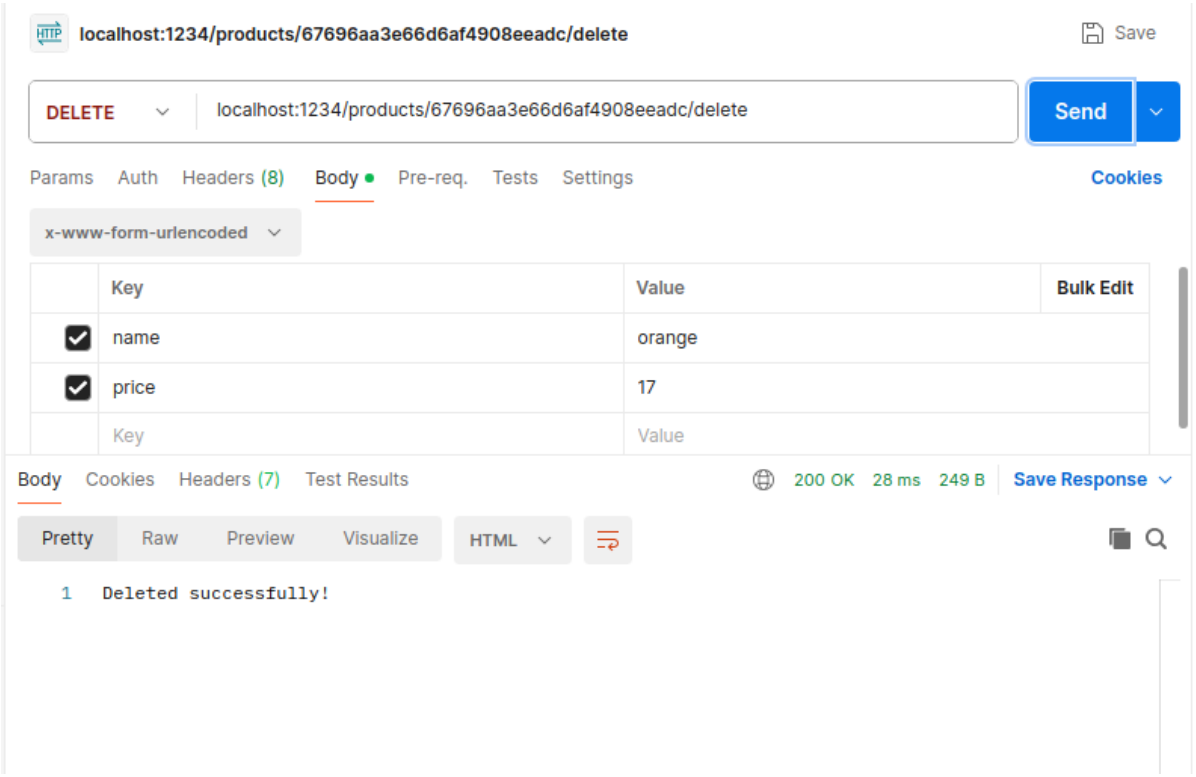


UPDATE




```
> use tp_database
switched to db tp_database
> db.products.find()
{ "_id" : ObjectId("67696aa3e66d6af4908eeadc"), "name" : "orange", "price" : 17,
 "__v" : 0 }
>
```

DELETE




```
> use tp_database
switched to db tp_database
> db.products.find()
>
```