

# Introduction à MongoDB

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. Installer mon environnement</b>	<b>3</b>
<b>III. Exercice : Appliquer la notion</b>	<b>17</b>
<b>IV. Fonctionnement de MongoDB</b>	<b>17</b>
<b>V. Exercice : Appliquer la notion</b>	<b>19</b>
<b>VI. Manipuler les documents [CRUD]</b>	<b>20</b>
<b>VII. Exercice : Appliquer la notion</b>	<b>26</b>
<b>VIII. Compass</b>	<b>26</b>
<b>IX. Exercice : Appliquer la notion</b>	<b>31</b>
<b>X. Essentiel</b>	<b>31</b>
<b>XI. Auto-évaluation</b>	<b>31</b>
A. Exercice final .....	31
B. Exercice : Défi .....	33
<b>Solutions des exercices</b>	<b>34</b>

## I. Contexte

**Durée :** 1h

**Environnement de travail :** Windows

**Pré-requis :** Aucun

### Contexte

La donnée est souvent présentée comme le nouvel or noir de ce siècle. Ceci est, en effet, confirmé par la course à la data qui oppose les géants du numérique, GAFA en tête, dont le modèle économique repose en grande partie sur le recueil, l'exploitation et la valorisation de toutes ces données recueillies. Les Systèmes de Gestion de Bases de Données ont donc dû évoluer pour être en capacité de traiter un volume gigantesque d'informations très hétérogènes, et s'adapter aux nouvelles architectures informatiques distribuées. C'est ainsi que les bases de données NOSQL - not only SQL - sont nées, et en particulier MongoDB que nous allons étudier plus en détail, et qui permet de stocker les informations sous forme de document.

## II. Installer mon environnement

### Objectifs

- Installer MongoDB en local sur son poste Windows
- Installer Compass, interface graphique permettant d'interagir avec sa base Mongo

### Mise en situation

Commençons donc par installer tous les outils qui nous seront nécessaires pour pouvoir créer une base de données MongoDB, et interagir avec celle-ci.

### Télécharger MongoDB

MongoDB peut être téléchargé depuis le site de l'éditeur<sup>1</sup>, en téléchargeant la version gratuite **Community**.

### Conseil

Plusieurs versions de MongoDB sont proposées sur le site de l'éditeur. Il faudra donc bien faire attention d'indiquer la bonne plateforme (Windows, Linux, MacOS...) en fonction de votre système d'exploitation. Concernant la version, il est d'usage d'installer la version stable la plus récente. Elle est indiquée comme version courante au niveau de la liste de choix sur le site éditeur.

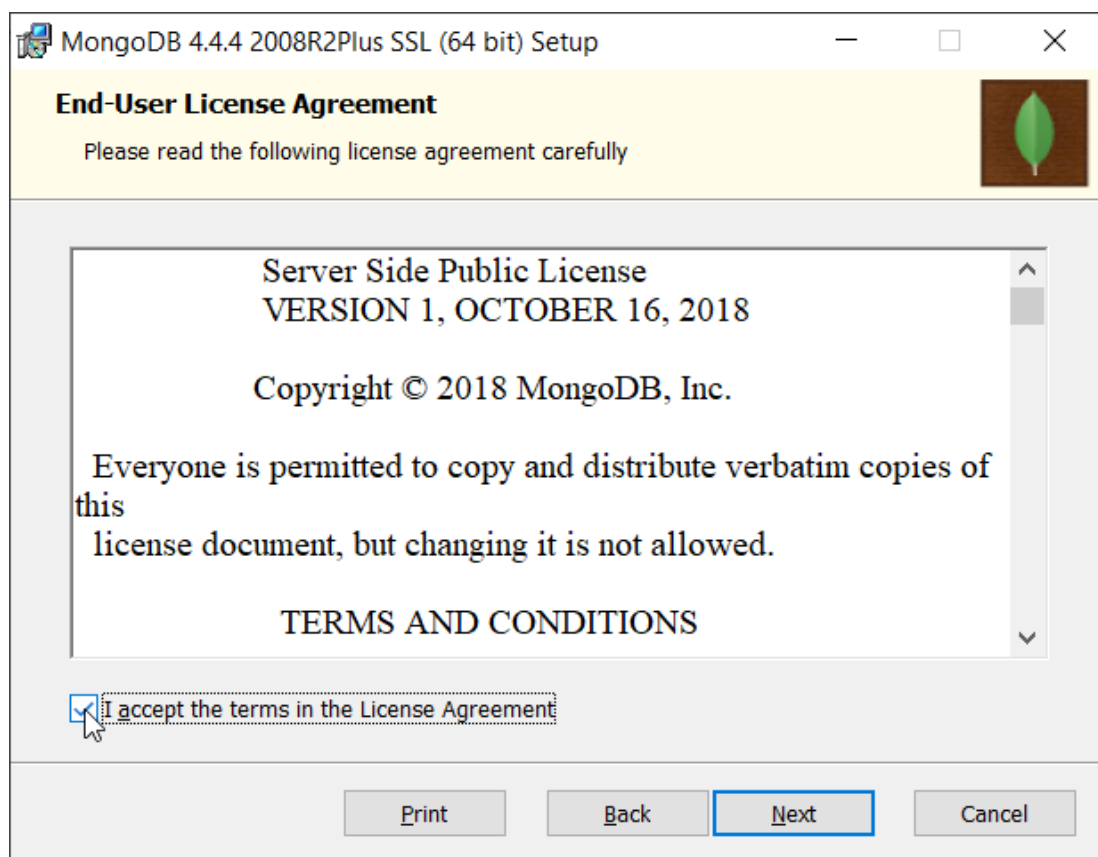
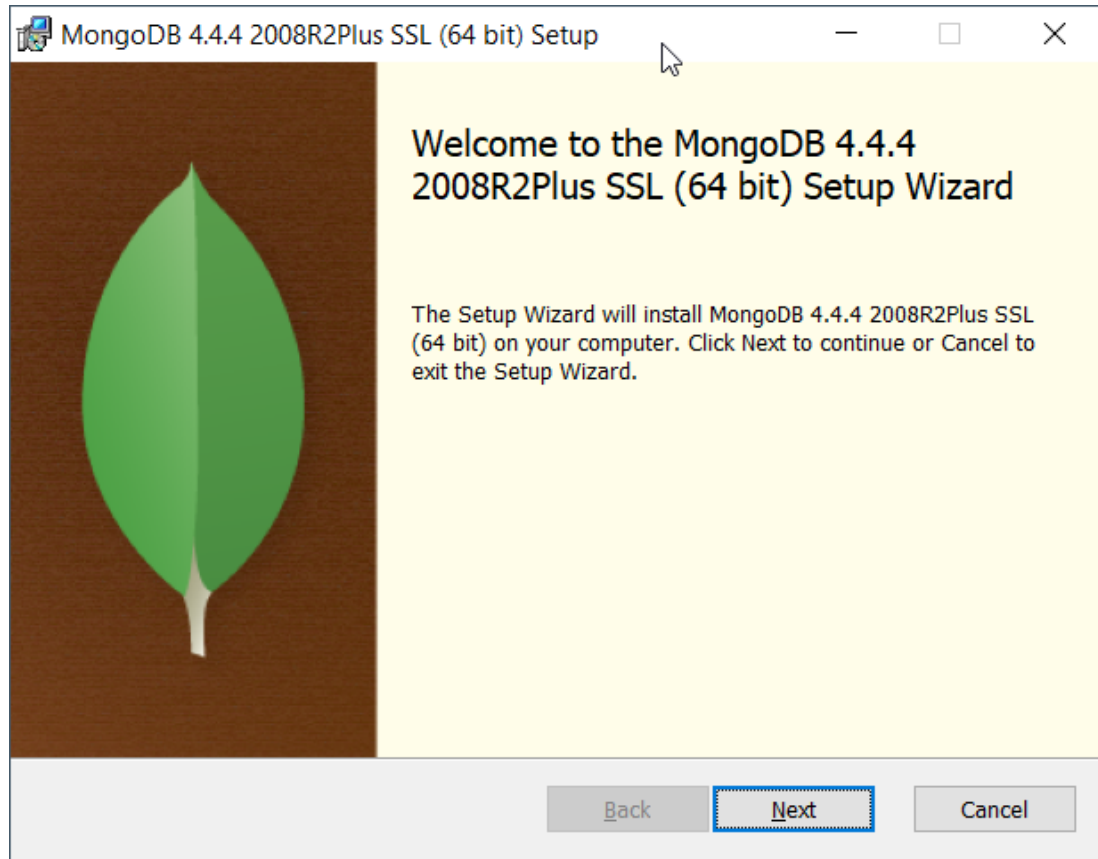
Concernant le Package, msi ou zip, le plus simple est de télécharger le msi qui pourra être exécuté directement sans avoir besoin de dézipper l'exécutable.

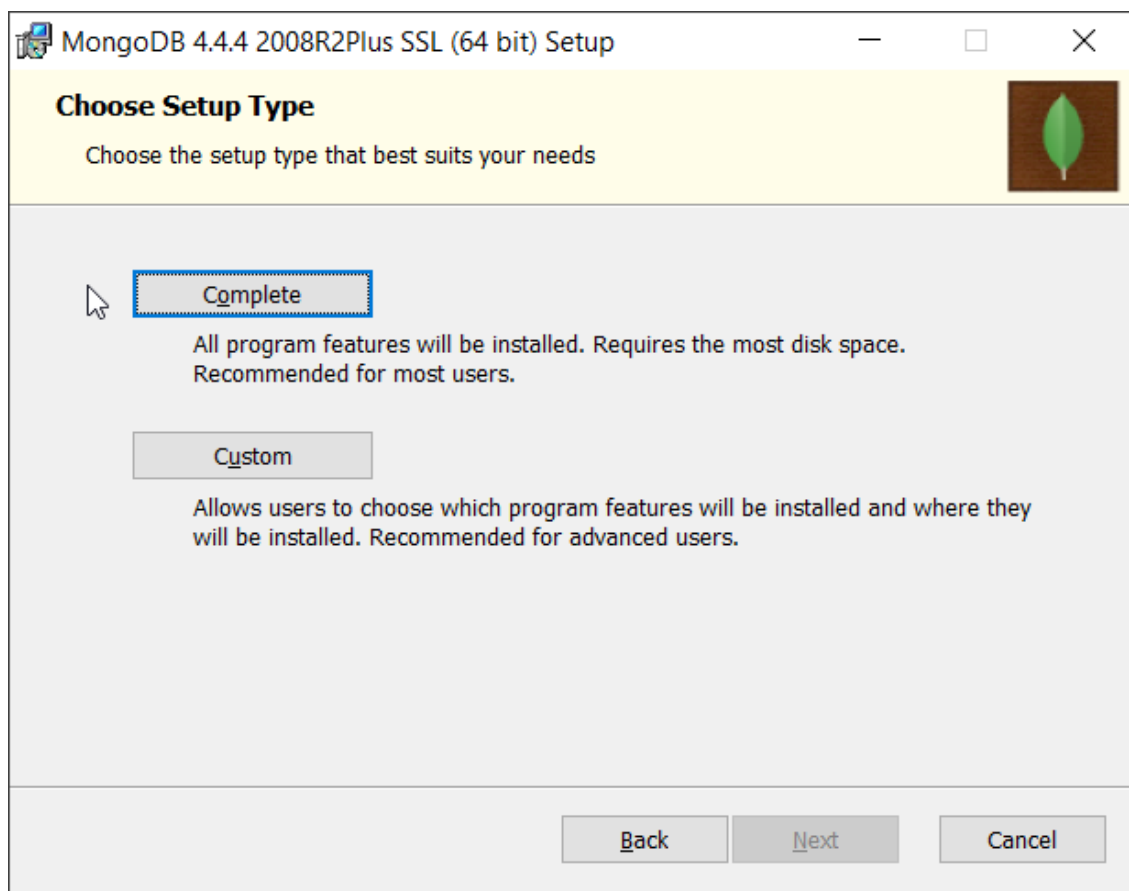
---

1 <https://www.mongodb.com/try/download/community>

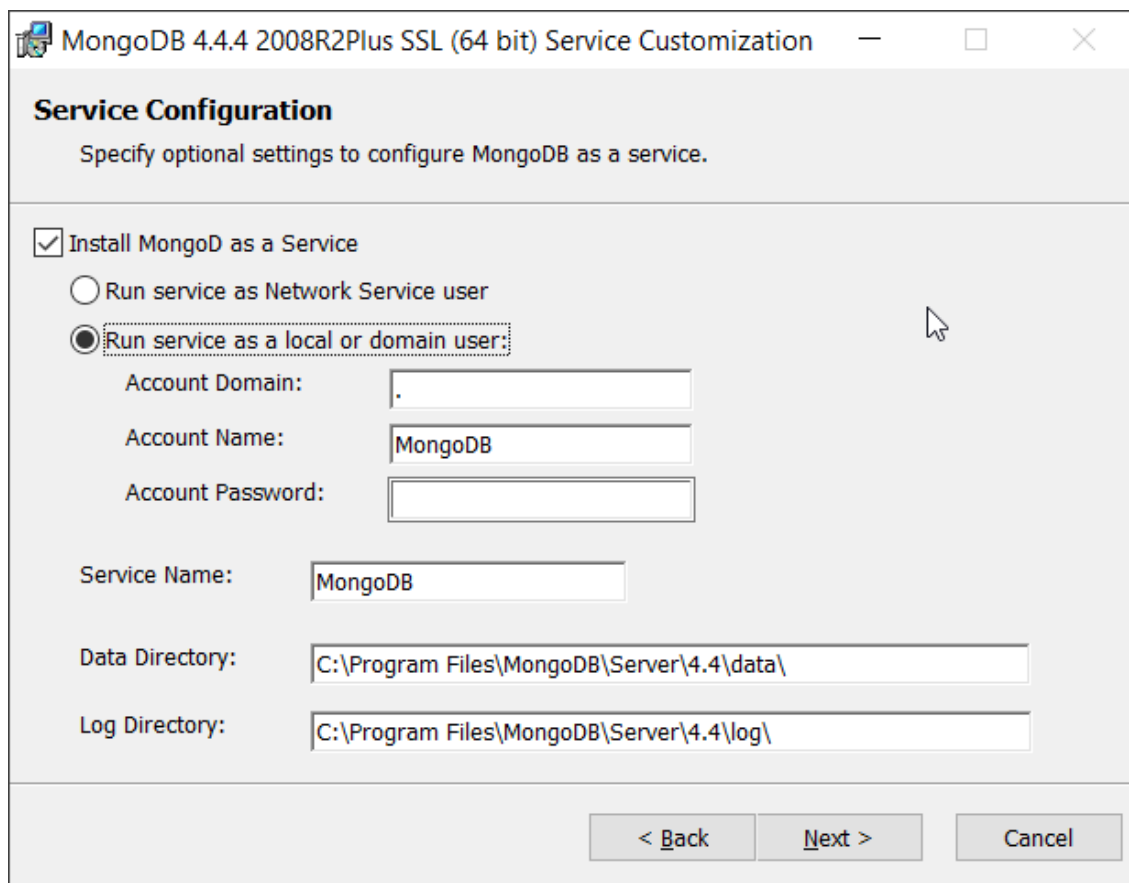
## Installer MongoDB et Compass

L'exécutable MongoDB est à présent téléchargé. Exécutez-le, et suivez la procédure d'installation.



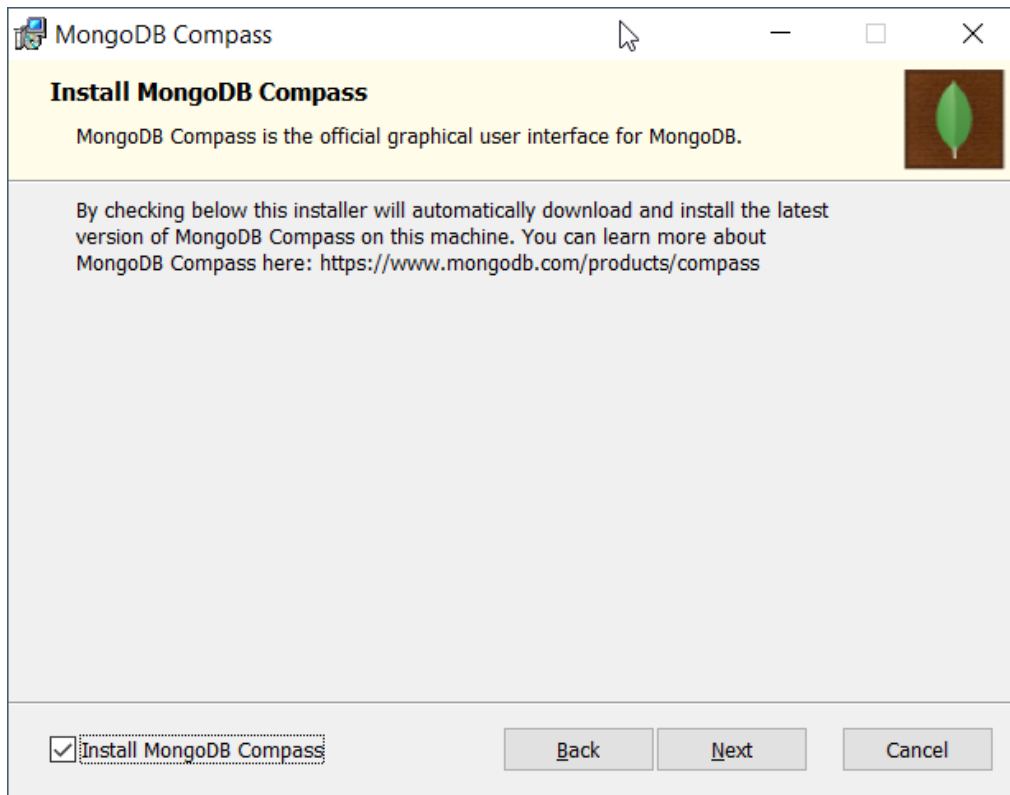


Choisir l'installation Complète.

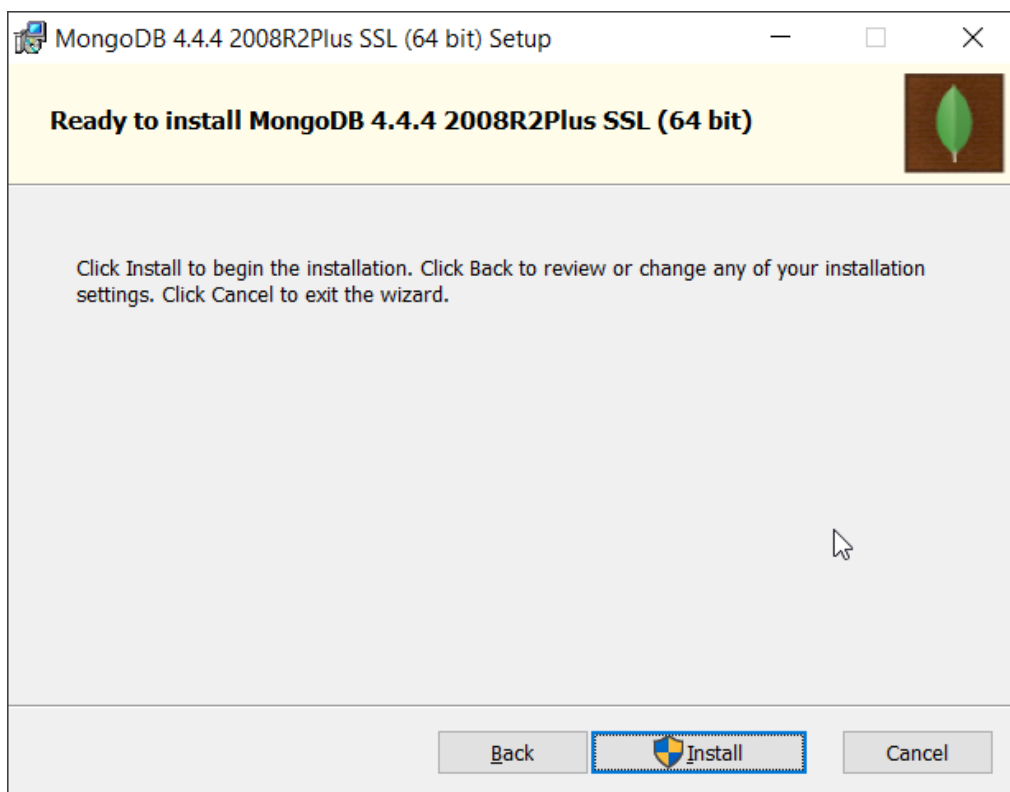


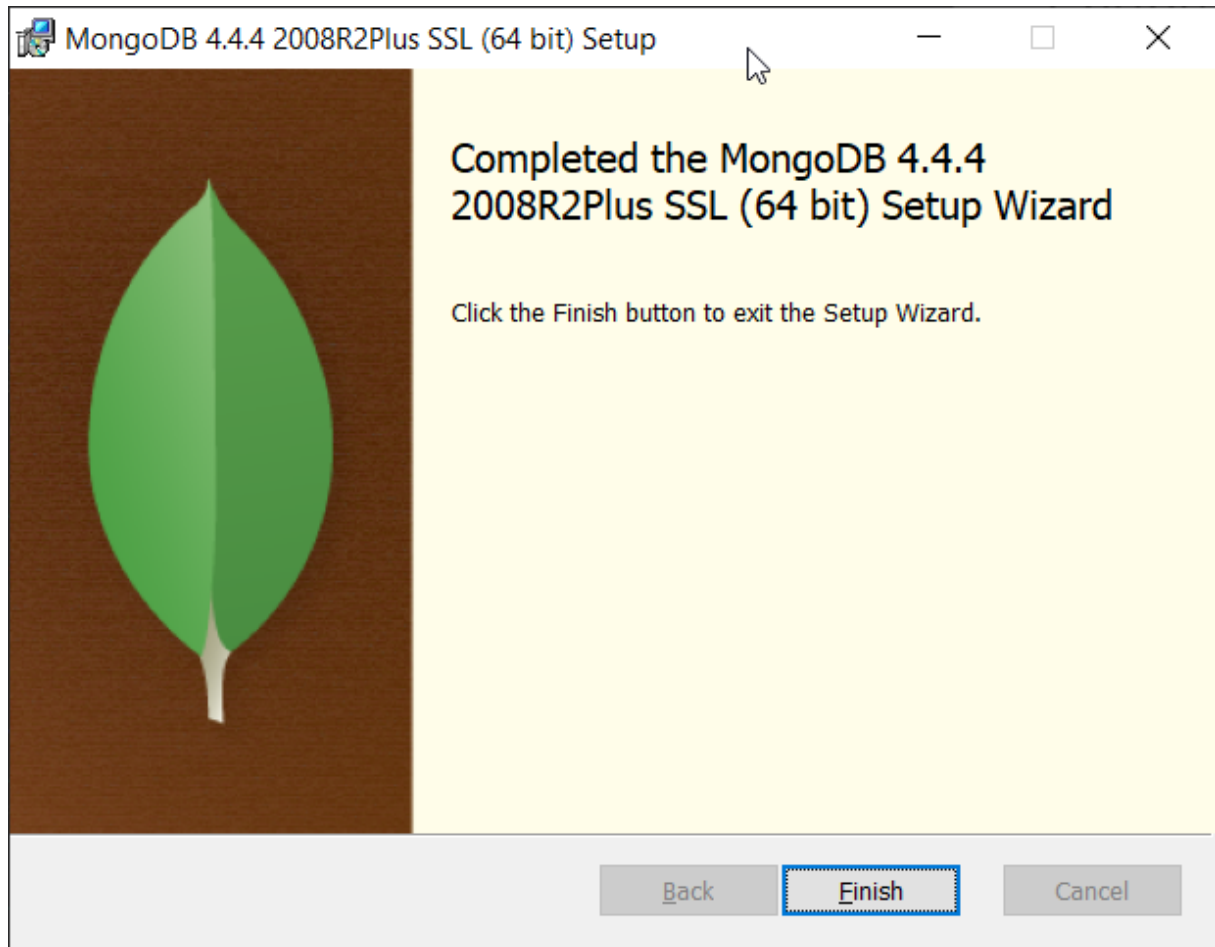
On laisse cocher l'option `Install MongoDB as a Service`, ce qui permettra de ne pas être obligé de lancer l'instruction `mongod` à chaque fois que nous voulons travailler sur notre base MongoDB en local.

Il est tout à fait possible de modifier les dossiers dans lesquels seront stockés les données ainsi que les fichiers log, mais pas d'utilité dans le cadre de cette introduction à MongoDB.



On profite de l'installation de MongoDB pour installer également l'application Compass.



**Rappel** Pour ouvrir une invite de commandes Windows

Cliquez sur la touche Windows de votre clavier (ou cliquez sur le menu Démarrer en bas à gauche de votre écran), tapez `cmd`, vérifiez que **Invite de commandes** est trouvée par Windows, et tapez sur votre touche `Entrée`.

**Vérification de la bonne installation de MongoDB et variable d'environnement**

Avec le système d'exploitation Windows, pour pouvoir utiliser un exécutable, il est nécessaire par défaut de se positionner dans le répertoire où l'application a été installée.

Pour vérifier la bonne installation de MongoDB, il est donc nécessaire d'ouvrir une invite de commandes Windows, se positionner dans le répertoire ad hoc, et vérifier si la version de MongoDB installée est bien retournée :

```
1 Microsoft Windows [version 10.0.19041.804]
2 (c) 2020 Microsoft Corporation. Tous droits réservés.
3
4 C:\Users\sebastienv>cd C:\Program Files\MongoDB\Server\4.4\bin
5
6 C:\Program Files\MongoDB\Server\4.4\bin>mongo --version
7 MongoDB shell version v4.4.4
8 Build Info: {
9   "version": "4.4.4",
10  "gitVersion": "8db30a63db1a9d84bdcad0c83369623f708e0397",
11  "modules": [],
12  "allocator": "tcmalloc",
13  "environment": {
14    "distmod": "windows",
```

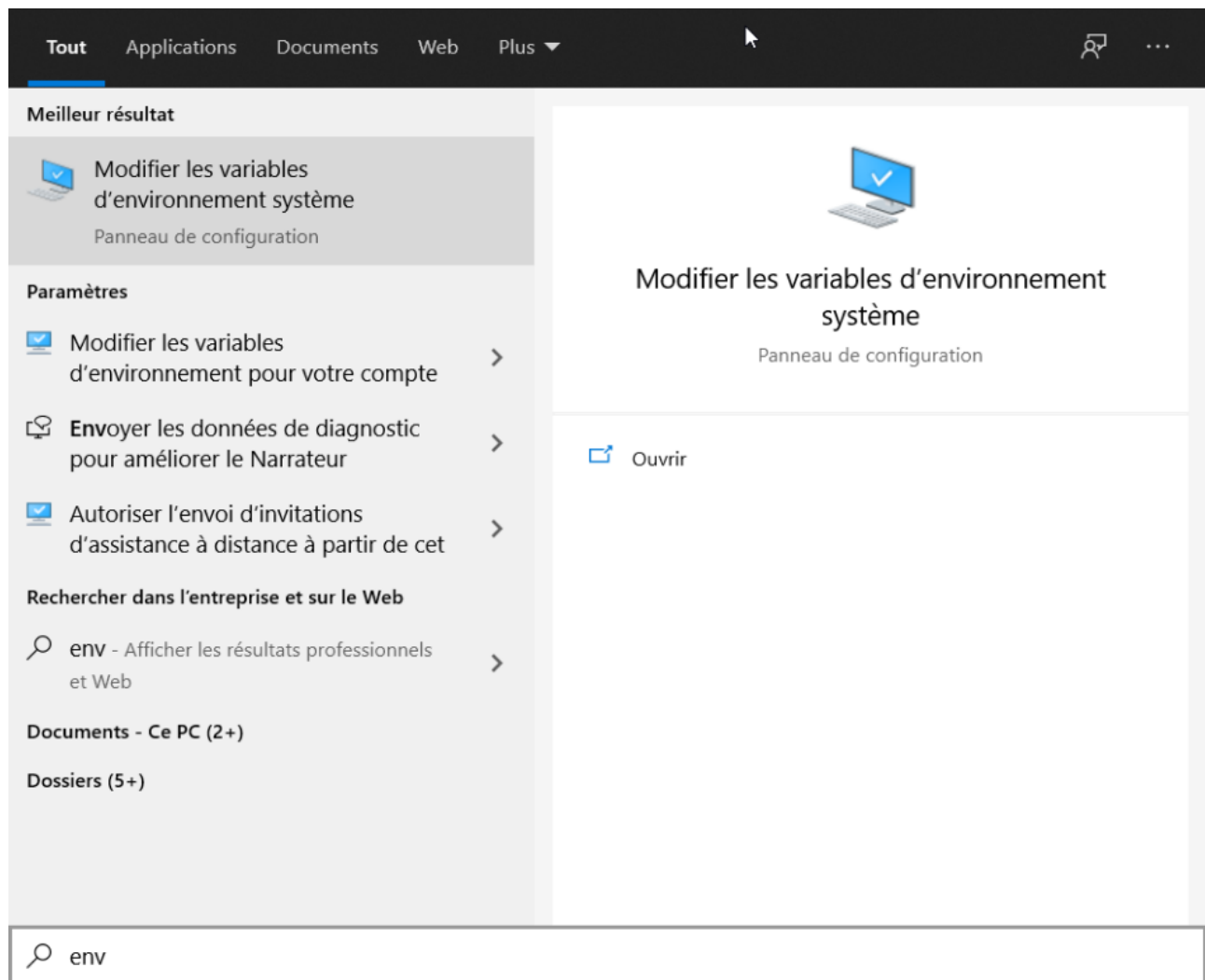
```

15     "distarch": "x86_64",
16     "target_arch": "x86_64"
17   }
18 }
19
20 C:\Program Files\MongoDB\Server\4.4\bin>

```

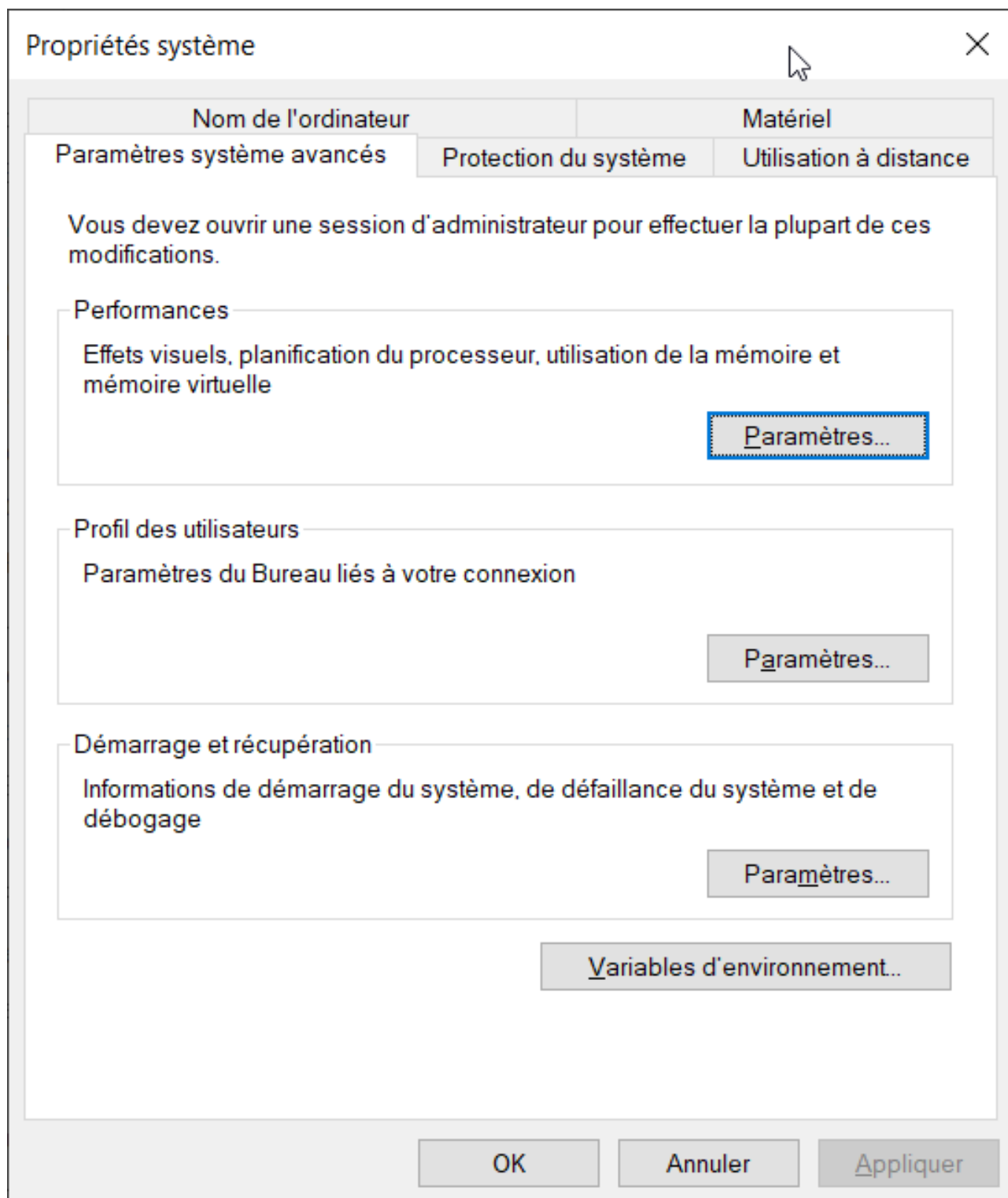
La version de MongoDB a bien été affichée, ce qui confirme sa bonne installation. Néanmoins, il serait problématique de devoir se repositionner dans le répertoire d'installation de MongoDB à chaque fois que nous souhaitons y accéder. Une solution simple est d'ajouter le dossier d'installation `C:\Program Files\MongoDB\Server\4.4\bin` au niveau du PATH dans les variables d'environnement du système Windows.

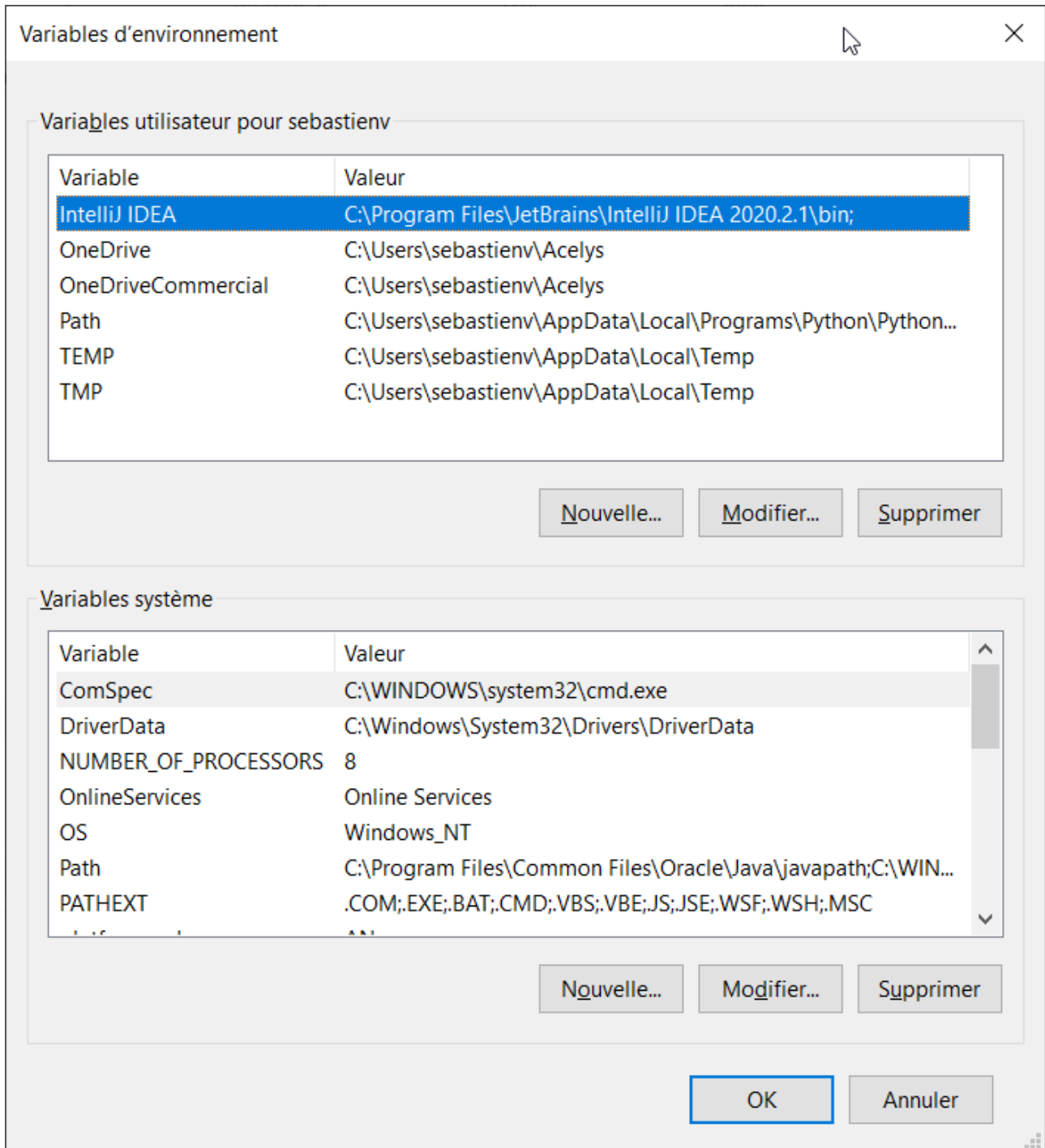
Commencez par rechercher **Modifier les variables d'environnement système** au niveau du menu Démarrer.



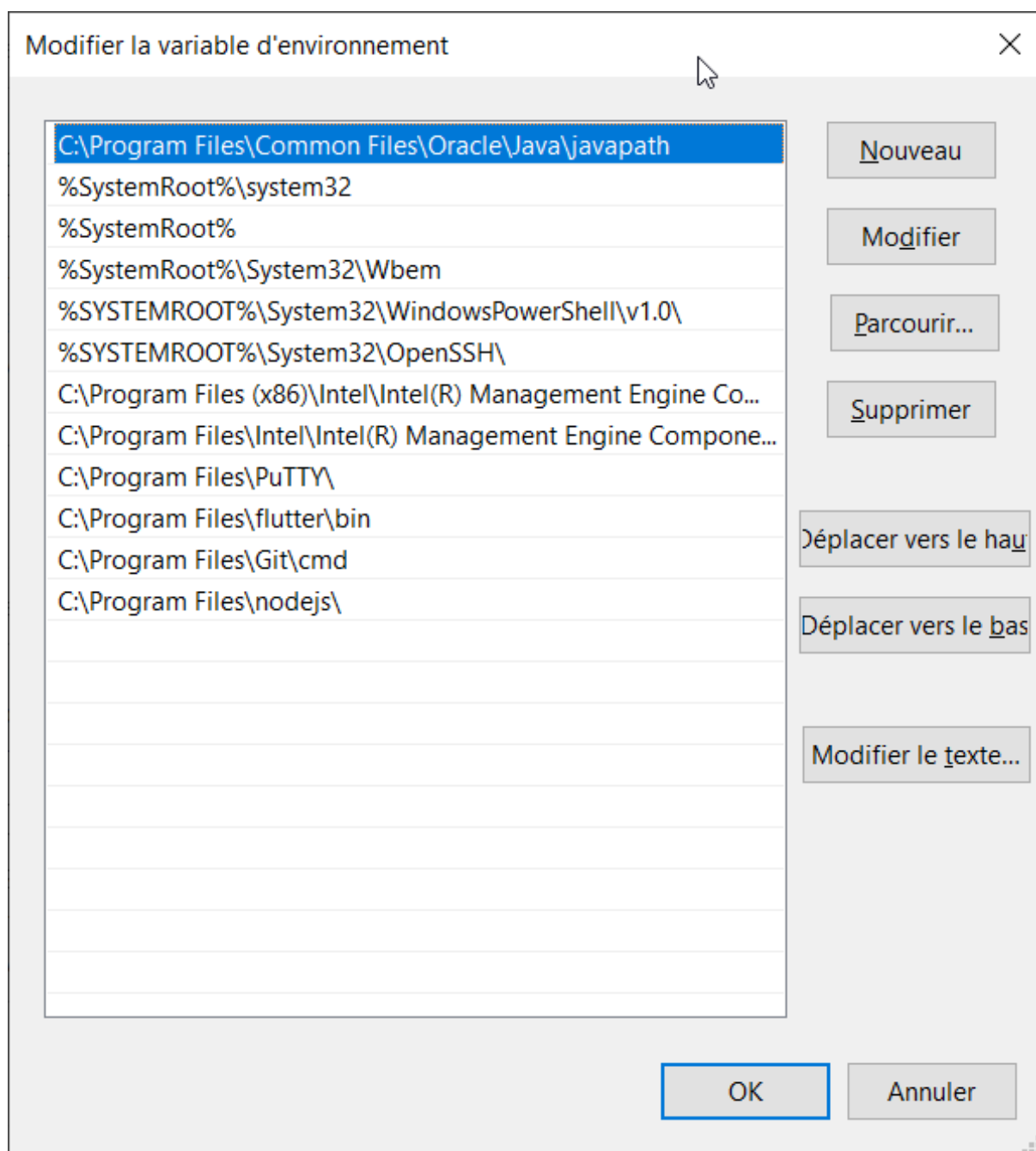
Cliquez ensuite sur le bouton **Variables d'environnement**







Cliquez sur la variable `PATH` dans **Variables Système**, et cliquez sur Modifier.



Cliquez sur Nouveau, collez le chemin `C:\Program Files\MongoDB\Server\4.4\bin` où MongoDB a été installé, et validez votre saisie.

MongoDB peut à présent être lancé depuis n'importe quel dossier.

#### **Méthode** Installer MongoDB et Compass sur Mac OS X

La méthode la plus simple et complète pour installer MongoDB sur Mac OS X est de passer par Homebrew, vous pouvez vérifier que ce dernier est installé sur votre système grâce à la commande :

```
1 $ Brew -v
```

```
Homebrew 3.3.2  
Homebrew/homebrew-core (git revision 60c7446230f; last commit 2021-11-04)
```

Dans le cas où Homebrew est installé la version s'affichera comme sur la capture d'écran (la version peut varier) et vous pouvez passer à l'installation de mongoDB. A contrario si le terminal vous indique : `brew: command not found`, cela signifie que Homebrew doit être installé sur votre système, pour cela nous devons installer la version actuelle de Homebrew grâce à la commande :

```
1 $ /bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Attention à ne pas copier de retour à la ligne !

Si pendant l'installation d'Homebrew, un message vous demande d'installer les Command-Line Tools de Xcode, vous pouvez le faire avec la commande :

```
1 $ xcode-select --install
```

Maintenant que votre installation de Homebrew est prête, vous pouvez commencer l'installation de mongoDB.

Pour commencer nous allons récupérer le dépôt officiel de mongoDB avec Homebrew :

```
1 $ brew tap mongodb/brew
```

```
=> Tapping mongodb/brew
Cloning into '/usr/local/Homebrew/Library/Taps/mongodb/homebrew-brew'...
remote: Enumerating objects: 794, done.
remote: Counting objects: 100% (291/291), done.
remote: Compressing objects: 100% (208/208), done.
remote: Total 794 (delta 144), reused 139 (delta 80), pack-reused 503
Receiving objects: 100% (794/794), 173.59 KiB | 4.96 MiB/s, done.
Resolving deltas: 100% (382/382), done.
Tapped 14 formulae (30 files, 238.4KB).
```

Grâce à l'ajout du dépôt officiel, nous pouvons installer directement mongoDB grâce à la commande :

```
1 $ brew install mongodb-community@5.0
```

Attention : la version installée ici est la 5.0 vue comme la dernière version actuelle sur la page de téléchargement de la version community, cependant, vous devrez peut-être spécifier une version plus récente, selon le moment où vous installez votre version.

```
=> mongodb-community
To start mongodb/brew/mongodb-community now and restart at login:
brew services start mongodb/brew/mongodb-community
Or, if you don't want/need a background service you can just run:
mongod --config /usr/local/etc/mongod.conf
```

Après quelque temps, téléchargement et installation vous devriez voir ces 5 lignes apparaître.

Comme dit sur les 5 dernières lignes le service mongoDB peut-être lancées grâce à la commande :

```
1 $ brew services start mongodb-community@5.0
```

Et peut-être arrêté grâce à la commande :

```
1 $ brew services stop mongodb-community@5.0
```

La base de données mongoDB sera lancé en service (cela signifie qu'il tournera en arrière-plan et sera lancé avec le mac).

À partir du moment où le service est lancé vous pouvez accéder au shell mongoDB avec la commande :

```
1 $ mongosh
```

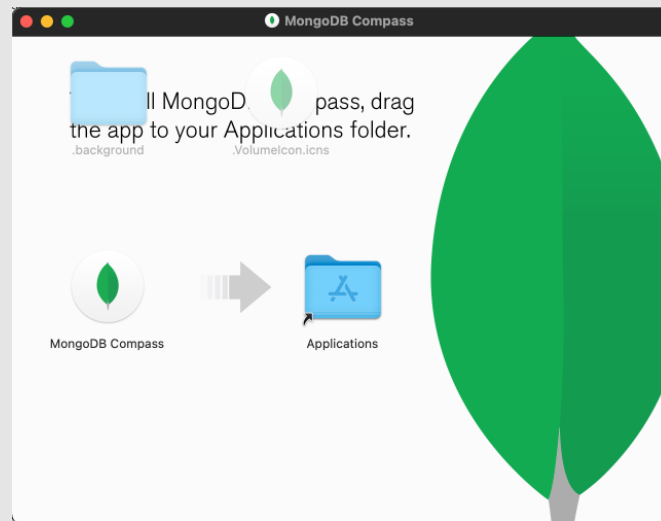
```
Current Mongosh Log ID: 6197d506827ac6e03c2579d7
Connecting to: mongodb://127.0.0.1:27017/?directConnection=true&server
SelectionTimeoutMS=2000
Using MongoDB: 5.0.3
Using Mongosh: 1.1.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to Mon
goDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting:
2021-11-19T17:46:58.098+01:00: Access control is not enabled for the databas
e. Read and write access to data and configuration is unrestricted
-----
```

La dernière partie de l'installation consiste en l'installation de l'outil mongoDB Compass. Il faudra vous rendre sur le site mongoDB Compass pour télécharger le dmg, une fois téléchargé il ne reste plus qu'à l'installer comme une app standard en ouvrant l'image dmg et déplaçant le fichier MongoDB Compass dans le dossier Applications.



### Méthode Installer MongoDB, MongoDB Shell et Compass sur une distribution Linux

#### MongoDB Server

Pour installer MongoDB sur une distribution Linux ( Ubuntu dans les exemples qui vont suivre ), vous devez récupérer les dépendances pré-requis :

```
1 cmd > sudo apt-get install libcurl4 openssl liblzma5
```

```
studi-ubuntu@studiubuntu-VM:~$ sudo apt-get install libcurl4 openssl liblzma5
[sudo] Mot de passe de studi-ubuntu :
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
libcurl4 est déjà la version la plus récente (7.68.0-1ubuntu2.7).
libcurl4 passé en « installé manuellement ».
liblzma5 est déjà la version la plus récente (5.2.4-1ubuntu1).
liblzma5 passé en « installé manuellement ».
openssl est déjà la version la plus récente (1.1.1f-1ubuntu2.9).
openssl passé en « installé manuellement ».
Les paquets suivants ont été installés automatiquement et ne sont plus nécessaires :
  libfprint-2-tod1 libllvm10
Veuillez utiliser « sudo apt autoremove » pour les supprimer.
0 mis à jour, 0 nouvellement installés, 0 à enlever et 0 non mis à jour.
```

Il faudra ensuite récupérer le fichier tarball souhaité en sélectionnant sur le site : MongoDB Community Server > La version notifié current > La version Linux utilisée > Le format tgz

### MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

---

The database is also offered as a fully-managed service with [MongoDB Atlas](#). Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

[Give it a try with a free, highly-available 512 MB cluster.](#)

<https://www.mongodb.com/try/download/community>

#### Available Downloads

Version  
5.0.4 (current)

Platform  
Ubuntu 20.04

Package  
tgz

Download Copy Link

[Current releases & packages](#)  
[Development releases](#)  
[Archived releases](#)  
[Changelog](#)  
[Release Notes](#)

Il faudra vous placer dans le dossier ou le fichier à été téléchargé et extraire le contenu (dans l'exemple le fichier à été mis dans le dossier Téléchargements) :

```
1 cmd > cd <Nom Du Dossier>
2 cmd > tar -zxvf <Nom Du Fichier>
```

```
studi-ubuntu@studiubuntu-VM:~$ cd Téléchargements/
studi-ubuntu@studiubuntu-VM:~/Téléchargements$ ls
mongodb-linux-x86_64-ubuntu2004-5.0.4.tgz
studi-ubuntu@studiubuntu-VM:~/Téléchargements$ tar -zxvf mongodb-linux-x86_64-ubuntu2004-5.0.4.tgz
mongodb-linux-x86_64-ubuntu2004-5.0.4/LICENSE-Community.txt
mongodb-linux-x86_64-ubuntu2004-5.0.4/MPL-2
mongodb-linux-x86_64-ubuntu2004-5.0.4/README
mongodb-linux-x86_64-ubuntu2004-5.0.4/THIRD-PARTY-NOTICES
mongodb-linux-x86_64-ubuntu2004-5.0.4/bin/install_compass
mongodb-linux-x86_64-ubuntu2004-5.0.4/bin/mongo
mongodb-linux-x86_64-ubuntu2004-5.0.4/bin/mongod
mongodb-linux-x86_64-ubuntu2004-5.0.4/bin/mongos
studi-ubuntu@studiubuntu-VM:~/Téléchargements$
```

Dans l'exemple, vous pouvez remarquer le ls qui permet d'afficher nos fichiers contenus dans notre dossier, cela facilite l'obtention du nom du fichier.

Un petit conseil : après avoir taper tar -zxvf mongo, vous pouvez appuyer sur TAB, le reste du nom du fichier devrait se remplir automatiquement.

Pour installer les fichiers binaires il faudra les déplacer dans le dossier local binaire :

```
1 cmd > sudo cp <Chemin Du Dossier Extraire>/bin/* /usr/local/bin/
```

```
studi-ubuntu@studiubuntu-VM:~/Téléchargements$ sudo cp mongodb-linux-x86_64-ubuntu2004-5.0.4/bin/* /usr/local/bin/
[sudo] Mot de passe de studi-ubuntu :
```

Le raccourcis TAB pour l'auto-complétion du nom du dossier est utilisable aussi dans cette commande.

MongoDB est à présent installé, vous pouvez le vérifier en tapant : `mongodb --version`

```
studi-ubuntu@studiubuntu-VM:~/Téléchargements/mongosh-1.14-linux-x64$ mongod --version
db version v5.0.4
Build Info: {
  "version": "5.0.4",
  "gitVersion": "62a84ede3cc9a334e8bc82160714df71e7d3a29e",
  "opensslVersion": "OpenSSL 1.1.1f 31 Mar 2020",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "ubuntu2004",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```

Il reste malgré tout à créer les dossiers qui permettront le fonctionnement de mongoDB :

```
1 cmd > sudo mkdir -p /var/lib/mongo
2 cmd > sudo mkdir -p /var/log/mongodb
3 cmd > sudo chown `whoami` /var/lib/mongo
4 cmd > sudo chown `whoami` /var/log/mongodb
```

```
studi-ubuntu@studiubuntu-VM:~$ sudo mkdir -p /var/lib/mongo
[sudo] Mot de passe de studi-ubuntu :
studi-ubuntu@studiubuntu-VM:~$ sudo mkdir -p /var/log/mongodb
studi-ubuntu@studiubuntu-VM:~$ sudo chown `whoami` /var/lib/mongo
studi-ubuntu@studiubuntu-VM:~$ sudo chown `whoami` /var/log/mongodb
```

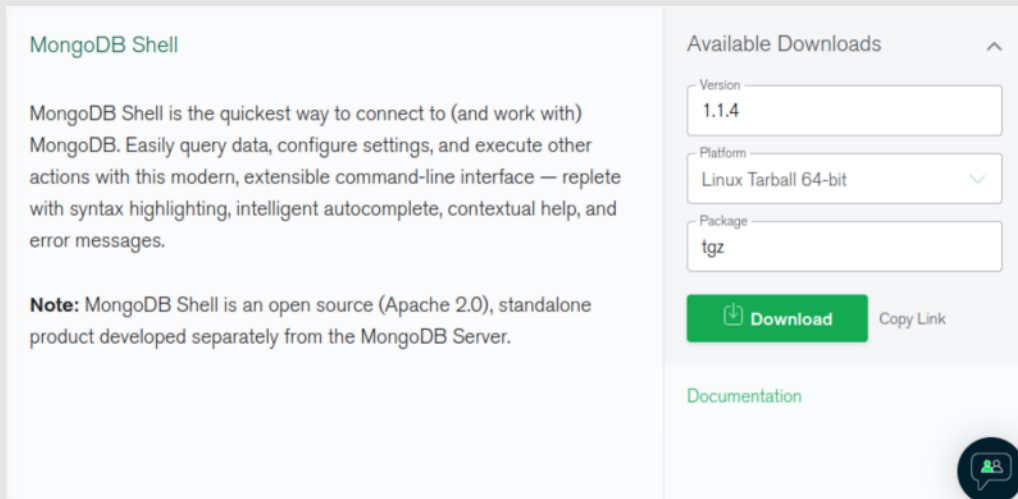
Enfin nous pourrons lancer le serveur avec :

```
1 cmd > mongod --dbpath /var/lib/mongo --logpath /var/log/mongodb/mongod.log --fork
```

```
studi-ubuntu@studiubuntu-VM:~/Téléchargements$ mongod --dbpath /var/lib/mongo -
-logpath /var/log/mongodb/mongod.log --fork
about to fork child process, waiting until server is ready for connections.
forked process: 2433
child process started successfully, parent exiting
```

## MongoDB Shell

Pour commencer, nous allons récupérer le fichier tgz de MongoDB Shell sur le site de téléchargement mongoDB.  
MongoDB Shell > La version la plus récente > Linux Tarball 64-bit > Le format tgz



Nous allons nous rendre dans le dossier où le fichier a été téléchargé, l'extraire et autoriser ces fichiers à s'exécuter :

```
1 cmd > cd <Nom Du Dossier>
2 cmd > tar -zxvf <Nom Du Fichier>
3 cmd > cd <Nom Du Dossier Extraît>
4 cmd > chmod +x bin/mongosh
5 cmd > chmod +x bin/mongocryptd-mongosh
```

```
studi-ubuntu@studiubuntu-VM:~/Téléchargements$ tar -zxvf mongosh-1.1.4-linux-x64.tgz
mongosh-1.1.4-linux-x64/
mongosh-1.1.4-linux-x64/LICENSE-mongocryptd
mongosh-1.1.4-linux-x64/LICENSE-mongosh
mongosh-1.1.4-linux-x64/README
mongosh-1.1.4-linux-x64/THIRD_PARTY_NOTICES
mongosh-1.1.4-linux-x64/bin/
mongosh-1.1.4-linux-x64/bin/mongosh.1.gz
mongosh-1.1.4-linux-x64/bin/mongocryptd-mongosh
mongosh-1.1.4-linux-x64/bin/mongosh
studi-ubuntu@studiubuntu-VM:~/Téléchargements$ cd mongosh-1.1.4-linux-x64/
studi-ubuntu@studiubuntu-VM:~/Téléchargements/mongosh-1.1.4-linux-x64$ chmod +x bin/mongosh
studi-ubuntu@studiubuntu-VM:~/Téléchargements/mongosh-1.1.4-linux-x64$ chmod +x bin/mongocryptd-mongosh
```

Le raccourci TAB fonctionnera en commençant le nom par mongosh puis en appuyant sur TAB.

Il faudra ensuite copier les fichiers binaires dans le dossier local binaire:

```
1 cmd > sudo cp bin/* /usr/local/bin/
```

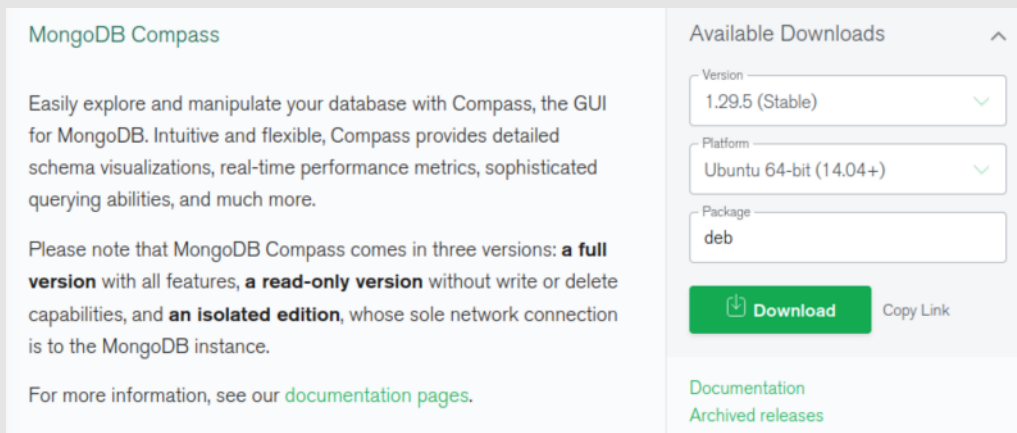
```
studi-ubuntu@studiubuntu-VM:~/Téléchargements/mongosh-1.1.4-linux-x64$ sudo cp bin/* /usr/local/bin/
[sudo] Mot de passe de studi-ubuntu :
studi-ubuntu@studiubuntu-VM:~/Téléchargements/mongosh-1.1.4-linux-x64$
```

MongoDB Shell est à présent installé, vous pouvez le vérifier en regardant la version avec mongosh --version

```
studi-ubuntu@studiubuntu-VM:~$ mongosh --version
1.1.4
```

## MongoDB Compass

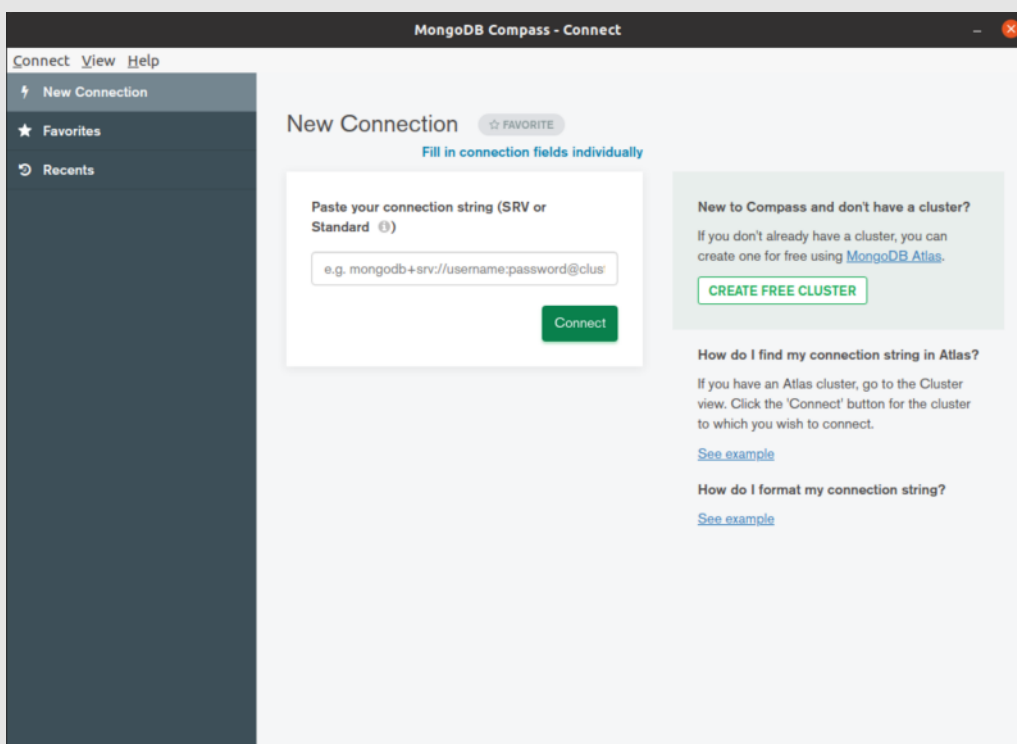
Pour installer MongoDB Compass, il faudra récupérer le fichier .deb ou .rpm selon la distribution linux sur le site :  
MongoDB Compass > La version notifiée stable> Votre plate-forme > Le package deb ou rpm



Puis il faudra installer le paquet :

```
1 cmd > sudo dpkg -i <Fichier>
```

Puis il sera possible de le lancer avec la commande `mongodb-compass`



## Syntaxe À retenir

- MongoDB peut s'installer sur tout système d'exploitation.
- La procédure d'installation est très simple puisqu'il suffit de télécharger l'exécutable d'installation sur le site officiel de l'éditeur puis suivre le processus d'installation (pour Windows).
- L'installation pour Mac OS X passe par Homebrew, et permet de l'installer rapidement.



## Exercice : Appliquer la notion

[solution n°1 p.35]

### Exercice

Il est possible d'installer MongoDB en local sur son poste, quel que soit son système d'exploitation.

- ☐ Vrai
- ☐ Faux

### Exercice

Compass est un outil indispensable pour manipuler une base de données MongoDB.

- ☐ Vrai
- ☐ Faux

### Exercice

Mon installation s'est réalisée sur mon poste avec succès.

- ☐ Vrai
- ☐ Faux

## IV. Fonctionnement de MongoDB

### Objectifs

- Comprendre les notions de collections et documents
- Savoir comment MongoDB stocke les informations

### Mise en situation

Maintenant que MongoDB est installé, il est temps de passer aux choses sérieuses et de voir comment ce système de gestion de bases de données fonctionne. Après avoir vu les notions de document et de collection, nous reviendrons sur le format de stockage des informations dans MongoDB.

### Documents

Un **document** est une information stockée dans MongoDB, pouvant être assimilé à un objet JSON. Un **document** est donc composé d'un ensemble d'informations sous la forme **champ/valeur**, où **champ** indique le sens à donner à cette information, et où **valeur** pourra être de différents types. Nous reviendrons sur ces différents types dans la prochaine section.

```
1 {  
2   "firstName": "Bob",  
3   "lastName": "Sponge",  
4   "age": 32,  
5   "hobbies": [  
6     "Soccer",  
7     "Table tennis"  
8   ]  
9 }
```

Voici un exemple de document pouvant être stocké dans MongoDB. Nous remarquerons au passage que la valeur d'un champ peut également prendre la forme d'un tableau.

## Collections

Contrairement aux bases de données relationnelles où les informations sont stockées sous forme de table, avec un schéma rigide pour chaque table, les informations seront regroupées dans MongoDB sous forme de collection.

Une collection se composera alors de documents ayant un sens commun, même s'ils n'ont pas nécessairement la même structure. Des documents présents dans une même collection partageront donc la plupart du temps des informations communes, représentant la logique métier.

### Exemple Trois documents qui pourraient composer une collection véhicule

```

1 [
2   {
3     type:"car",
4     motors: 1,
5     energy:"electric",
6     color: "red",
7     owner:{
8       firstname: "Bob",
9       lastname: "Ine"
10    },
11    location:{
12      type: "Point",
13      coordinates: [ 3.94886, 43.55757 ]
14    }
15  },
16  {
17    type:"plane",
18    motors: 2,
19    company: "Air France",
20    owner:{
21      firstname: "Erik",
22      lastname: "Lost"
23    },
24    location:{
25      type: "Point",
26      coordinates: [ 3.95870, 43.58463 ]
27    }
28  },
29  {
30    type:"boat",
31    motors: 4,
32    homeport: "Palavas",
33    owner:{
34      firstname: "Merill",
35      lastname: "Stubing"
36    },
37    location:{
38      type: "Point",
39      coordinates: [ 3.92171, 43.53199 ]
40    }
41  }
42 ]

```

L'exemple ci-dessus illustre bien trois documents représentant chacun un objet véhicule, ayant des caractéristiques communes : `type`, `motors`, `owner`, `location`. Pour autant, la force du NOSQL est de pouvoir apporter des informations supplémentaires sur certains documents, et uniquement lorsque cela est nécessaire. Ici, la voiture a pu se voir préciser les informations `energy` et `color`, l'avion l'information `company`, et le bateau son `homeport`.

**Remarque**

Les recherches géospatiales ne seront pas abordées dans cette initiation à MongoDB, mais il est intéressant de savoir qu'à partir d'un champ `location` tel que défini précédemment, il serait tout à fait possible, grâce à MongoDB, de rechercher tous les véhicules à moins de 3 km de notre propre position !

**Format de stockage BSON**

Il est important de bien comprendre que les documents présents dans MongoDB ne seront pas directement stockés au format JSON, mais au format BSON, Binary JSON. Ce format n'est pas directement lisible par un utilisateur (comme tout fichier binaire). Pourquoi donc s'arrêter alors sur cette précision puisque nous n'aurons jamais à analyser ou comprendre directement une donnée BSON ? Et bien cela permet de comprendre pourquoi il est possible de stocker une grande diversité de types d'information : *string*, *object*, *objectId*, *bool*, *date*, *null*, *timestamp*, ...

Vous pouvez retrouver la liste exhaustive des types disponibles au sein de la documentation officielle<sup>1</sup>.

**Syntaxe****À retenir**

- Une collection est l'équivalent d'une table dans un modèle relationnel, et permet de regrouper des documents ayant un sens commun, et partageant des caractéristiques communes, appelées champs.
- Un document est une information présente sous la forme d'un objet JSON, plus ou moins complexe. Ce document est stocké dans MongoDB en BSON.

**Complément**

- Liste des types disponibles sur MongoDB<sup>2</sup>

**Exercice : Appliquer la notion**

[solution n°2 p.35]

## Exercice

MongoDB stocke les informations au format JSON, ce qui est bien pratique pour interagir avec des API REST.

- ☐ Vrai
- ☐ Faux

## Exercice

Le BSON a l'avantage de fournir une très grande liste de types possibles pour nos données, bien plus que les types proposés par JSON.

- ☐ Vrai
- ☐ Faux

<sup>1</sup> <https://docs.mongodb.com/manual/reference/bson-types/>

<sup>2</sup> <https://docs.mongodb.com/manual/reference/bson-types/>

## Exercice

### Un Document...

- ☐ se représente sous la forme d'un JSON, et se compose de paires champ/valeur
- ☐ est stocké dans une table MongoDB
- ☐ doit être créé dans une collection
- ☐ peut contenir une infinité de champs

## VI. Manipuler les documents [CRUD]

### Objectifs

- Créer un document ou plusieurs documents dans une collection
- Accéder aux documents présents dans une collection
- Appliquer un filtre ou une projection
- Mettre à jour un document
- Supprimer un document ou une collection

### Mise en situation

Le but de ce cours sera de créer une application de gestion des visiteurs. Cette application devra pouvoir enregistrer des informations sur un visiteur, en particulier le motif de la visite, ainsi que la personne visitée dans l'entreprise. Le visiteur devra pouvoir indiquer son départ en fin de visite. Vous avez validé avec le client qu'il s'agira de créer une application pour tablette, et que votre application mobile stockera les informations dans une base MongoDB.

Dès maintenant, vous allez pouvoir utiliser les principales fonctions MongoDB dites **CRUD [Create, Read, Update, Delete]**, pour modéliser nos visiteurs sous forme de document au sein d'une collection *visitors*.

#### Remarque Utilisation de Mongo Shell

De très nombreux langages Back (Java, PHP, Node.JS ...) pourront bénéficier d'un driver spécifique pour leur permettre de se connecter à une base de données MongoDB et de manipuler les documents et collections. Dans ce cours, il est possible d'interagir directement dans la base de donnée à partir du Mongo Shell, ce qui permettra de bien comprendre les instructions principales, sans être lié à une technologie Back en particulier.

### Création

Commençons par se connecter à notre base de données MongoDB et se positionner sur une nouvelle collection que l'on va nommer `welcome`. Pour cela, il faut ouvrir un terminal, qui peut être intégré à votre IDE favori, ou simplement une invite de commandes Windows. Lancez la commande `mongo`.

```
1 C:\Users\sebastienv>mongo
2 MongoDB shell version v4.4.4
3 connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
4 Implicit session: session { "id" : UUID("b3cc3b12-b59e-4641-9170-1a1ac9fdc0fa") }
5 MongoDB server version: 4.4.4
6 ---
7 The server generated these startup warnings when booting:
8   2021-03-13T08:01:07.058+01:00: Access control is not enabled for the database. Read
9   and write access to data and configuration is unrestricted
10 ---
```

```

11      Enable MongoDB's free cloud-based monitoring service, which will then receive and
12      display metrics about your deployment (disk utilization, CPU, operation statistics, etc).
13
14      The monitoring data will be available on a MongoDB website with a unique URL
15      accessible to you and anyone you share the URL with. MongoDB may use this information to make product
16      improvements and to suggest MongoDB products and deployment options to you.
17
18      To enable free monitoring, run the following command: db.enableFreeMonitoring()
19      To permanently disable this reminder, run the following command:
20      db.disableFreeMonitoring()
21  ---
22  >

```

On commence par consulter la liste des databases existantes avec la commande : `show dbs`

```

1 > show dbs
2 admin  0.000GB
3 config 0.000GB
4 local  0.000GB
5 >

```

Nous indiquons à MongoDB la database que l'on souhaite utiliser avec la commande : `use nomDeLaBase`. Si celle-ci n'existe pas, elle sera automatiquement créée à la première création de document

```

1 > use welcome
2 switched to db welcome
3 >

```

Il est possible d'utiliser la méthode `insertOne()` pour créer un nouveau document dans la collection `visitors`. Cette méthode prend en paramètre le document à créer.

```

1 > db.visitors.insertOne({firstname:"Dupont",lastname:"Jacques",startedAt:ISODate(),visited:
2   ["SVE","PWR"]})
3   {
4     "acknowledged" : true,
5     "insertedId" : ObjectId("604da2ce558700c6a80bce73")
6   }
7 >

```

`acknowledged` à `True` indique que l'instruction a bien été prise en compte par MongoDB.

La méthode retourne également l'identifiant unique du document créé, au format `ObjectId`.

Créons quelques visiteurs complémentaires, en imaginant que ceux-ci n'ont pas tous rempli les mêmes informations facultatives sur le formulaire proposé au niveau de la tablette d'accueil.

Lorsque plusieurs documents sont à créer, il est possible d'utiliser la méthode `insertOne()` en itérant sur chaque document, mais il est également possible d'utiliser la méthode `insertMany()`, qui prendra en paramètre un tableau de documents à créer.

```

1 > db.visitors.insertMany([
2   {firstname:"Martin",lastname:"Durand",society:"Studi",
3     reason:"Atelier de travail", startedAt:ISODate(),visited:["SVE"]},
4   {firstname:"Pierre",lastname:"Robes",
5     ... startedAt:ISODate(),visited:["PWR", "ERR"], reason: "Business"}])
6   {
7     "acknowledged" : true,
8     "insertedIds" : [
9       ObjectId("604da430558700c6a80bce74"),
10      ObjectId("604da430558700c6a80bce75")
11    ]
12  }
13 >

```

La méthode retourne alors un tableau d'**ObjectId**, représentant la liste des identifiants des documents créés.

Il est possible ensuite d'utiliser la méthode `count()` qui permet de compter le nombre de documents présents dans notre collection `visitors`.

```
1> db.visitors.countDocuments()
```

## L'id unique pour les documents MongoDB

L'`ObjectId` dans un document mongoDB pourrait être comparé à la colonne `id` (unique et en clé primaire) pour les tables de données SQL, il s'agit d'un champ en type BSON qui portent le nom : `_id`, qui est unique et identifie le document.

Elle est composée de :

- 4 octets représentant le timestamp actuel (nombre de seconde depuis le 1er janvier 1970)
- 5 octets représentant un nombre unique propre à une machine et un processus (3 pour la machine / 2 pour le processus)
- 3 octets commençant par une valeur aléatoire

L'`ObjectId` permet donc d'effectuer des actions sur un document en étant sûr d'affecter celui que l'on vise.

## Lecture

La méthode `find()` va permettre de consulter tous les documents présents dans une collection.

```
1> db.visitors.find()
2 { "_id" : ObjectId("604da2ce558700c6a80bce73"), "firstname" : "Dupont", "lastname" :
  "Jacques", "startedAt" : ISODate("2021-03-14T05:44:46.694Z"), "visited" : [ "SVE", "PWR" ] }
3 { "_id" : ObjectId("604da430558700c6a80bce74"), "firstname" : "Martin", "lastname" : "Durand",
  "society" : "Studi", "reason" : "Atelier de travail", "startedAt" : ISODate("2021-03-
  14T05:50:40.973Z"), "visited" : [ "SVE" ] }
4 { "_id" : ObjectId("604da430558700c6a80bce75"), "firstname" : "Pierre", "lastname" : "Robes",
  "startedAt" : ISODate("2021-03-14T05:50:40.973Z"), "visited" : [ "PWR", "ERR" ], "reason" :
  "Business" }
5 >
```

Il est important de remarquer qu'un champ `"_id"` a été automatiquement créé par MongoDB à la création de chaque document. Il s'agit d'un identifiant unique de type `ObjectId`. Nous aurions pu préciser une valeur pour ce champ à la création des collections, à condition de bien veiller à l'unicité des valeurs renseignées. Nous avons ici fait le choix de laisser MongoDB alimenter l'`id` avec un identifiant quelconque.

Très souvent, il n'est pas nécessaire de récupérer toutes les informations présentes dans la collection, mais uniquement certains documents vérifiant une condition. Il est pour cela tout à fait possible d'ajouter un paramètre à la fonction `find()`, afin de préciser le filtre souhaité.

Voici toutes les personnes ayant visité le collaborateur SVE :

```
1> db.visitors.find({visited: "SVE"})
2 { "_id" : ObjectId("604da2ce558700c6a80bce73"), "firstname" : "Dupont", "lastname" :
  "Jacques", "startedAt" : ISODate("2021-03-14T05:44:46.694Z"), "visited" : [ "SVE", "PWR" ] }
3 { "_id" : ObjectId("604da430558700c6a80bce74"), "firstname" : "Martin", "lastname" : "Durand",
  "society" : "Studi", "reason" : "Atelier de travail", "startedAt" : ISODate("2021-03-
  14T05:50:40.973Z"), "visited" : [ "SVE" ] }
4 >
```

La méthode retourne alors les deux visiteurs de SVE, et a exclu Pierre Robes du résultat retourné.

Nous venons sans le savoir d'utiliser notre premier opérateur, l'opérateur égal `$eq`, de manière implicite.

En effet, nous aurions pu utiliser la version non abrégée pour obtenir le même résultat.

```
1> db.visitors.find({visited: {$eq: "SVE"}})
2 { "_id" : ObjectId("604da2ce558700c6a80bce73"), "firstname" : "Dupont", "lastname" :
  "Jacques", "startedAt" : ISODate("2021-03-14T05:44:46.694Z"), "visited" : [ "SVE", "PWR" ] }
3 { "_id" : ObjectId("604da430558700c6a80bce74"), "firstname" : "Martin", "lastname" : "Durand",
  "society" : "Studi", "reason" : "Atelier de travail", "startedAt" : ISODate("2021-03-
  14T05:50:40.973Z"), "visited" : [ "SVE" ] }
4 >
```

La liste des opérateurs disponibles est longue, mais on peut retrouver en particulier les opérateurs de comparaison suivants :

- `$ne` : non égal
- `$gt` / `$gte` : supérieur / supérieur ou égal
- `$lt` / `$lte` : inférieur / inférieur ou égal
- `$in` / `$nin` : inclus / non inclus dans une liste de valeurs

Recherchons par exemple tous ceux qui n'ont pas visité SVE :

```
1> db.visitors.find({visited: {$ne: "SVE"}})
2 { "_id" : ObjectId("604da430558700c6a80bce75"), "firstname" : "Pierre", "lastname" : "Robes",
  "startedAt" : ISODate("2021-03-14T05:50:40.973Z"), "visited" : [ "PWR", "ERR" ], "reason" :
  "Business" }
3 >
```

Il est également possible d'utiliser des opérateurs logiques lorsqu'il s'agit de rechercher des documents selon plusieurs conditions :

- `$and` : retourne le document si plusieurs conditions sont réunies
- `$or` : retourne le document si au moins une condition est réunie
- `$nor` : retourne le document si aucune des conditions n'est remplie
- `$not` : retourne le document si la condition n'est pas remplie

Recherchons par exemple tous les visiteurs de SVE de l'entreprise Studi :

```
1> db.visitors.find({$and: [{society: "Studi"},{visited: "SVE"}]})
2 { "_id" : ObjectId("6051d14b11978b5d3571a287"), "firstname" : "Martin", "lastname" : "Durand",
  "society" : "Studi", "reason" : "Atelier de travail", "startedAt" : ISODate("2021-03-
  17T09:52:11.943Z"), "visited" : [ "SVE" ] }
3 >
```

Il est également possible de restreindre les informations retournées, et de ne fournir que certains champs au moment de retourner le résultat. On parle alors de **projection**. Il faut alors indiquer un second paramètre à la méthode `find()` pour lui préciser les colonnes que l'on souhaite conserver au niveau du résultat de la requête.

```
1> db.visitors.find({visited: "SVE"},{firstname: 1, lastname: 1, startedAt: 1})
2 { "_id" : ObjectId("604da2ce558700c6a80bce73"), "firstname" : "Dupont", "lastname" :
  "Jacques", "startedAt" : ISODate("2021-03-14T05:44:46.694Z") }
3 { "_id" : ObjectId("604da430558700c6a80bce74"), "firstname" : "Martin", "lastname" : "Durand",
  "startedAt" : ISODate("2021-03-14T05:50:40.973Z") }
4 >
```

Seuls les champs `firstname`, `lastname` et `startedAt` sont alors retournés, à l'exception du champ `_id` qui est retourné même s'il n'a pas été précisé au niveau de la projection souhaitée. Si nous ne souhaitons pas transmettre cette information, il est alors nécessaire de le préciser de manière explicite.

```
1> db.visitors.find({visited: "SVE"},{firstname: 1, lastname: 1, startedAt: 1, _id: 0})
2 { "firstname" : "Dupont", "lastname" : "Jacques", "startedAt" : ISODate("2021-03-
  14T05:44:46.694Z") }
3 { "firstname" : "Martin", "lastname" : "Durand", "startedAt" : ISODate("2021-03-
  14T05:50:40.973Z") }
4 >
```

#### Remarque

La méthode `find()` est un outil extrêmement puissant pour filtrer les informations souhaitées, et mériterait un chapitre entier. N'hésitez donc pas à consulter la documentation officielle de MongoDB<sup>1</sup> à son sujet.

1 <https://docs.mongodb.com/manual/tutorial/query-documents/>

## Mise à jour

Il est tout à fait possible de modifier un document existant, ce qui va nous donner l'occasion de découvrir deux nouveaux opérateurs.

Nous venons justement de nous rendre compte que Jacques Dupont a mal renseigné son formulaire et a inversé son nom et prénom, ce qui s'est alors répercuté sur notre document en base de données.

Utilisons alors la méthode `updateOne(param1, param2)` afin de mettre à jour ce document. Cette méthode prend en premier paramètre un filtre pour identifier le document à mettre à jour, le second paramètre indique les modifications à réaliser.

### Remarque Un unique document mis à jour

Dans le cas où le filtre appliqué en paramètre 1 de cette méthode `updateOne()` ne serait pas assez restrictif et retournerait plusieurs documents, **seul le premier document serait alors mis à jour**.

```
1 > db.visitors.updateOne({firstname: "Dupont"}, {$set: {firstname: "Jacques", lastname:
  "Dupont", updatedAt: ISODate()}})
2 { "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
3 >
```

La requête a bien été prise en compte (`"acknowledged" : true`), un document a été trouvé par rapport au filtre indiqué en premier paramètre (`"matchedCount" : 1`), et un document a bien été modifié (`"modifiedCount" : 1`).

```
1 > db.visitors.find({firstname: "Jacques"})
2 { "_id" : ObjectId("604da2ce558700c6a80bce73"), "firstname" : "Jacques", "lastname" :
  "Dupont", "startedAt" : ISODate("2021-03-14T05:44:46.694Z"), "visited" : [ "SVE", "PWR" ],
  "updatedAt" : ISODate("2021-03-14T07:33:26.900Z") }
3 >
```

Après vérification avec la méthode `find()`, nous constatons en effet que le nom et prénom ont bien été inversés comme nous le souhaitions, et un champ contenant la date de mise à jour du document a également été ajouté.

Une autre méthode existe, et a l'avantage de retourner le document concerné par la modification ; il s'agit de la méthode `findOneAndUpdate()`. Par défaut, cette méthode retournera le document tel qu'il existait **avant** sa mise à jour. Il est tout à fait possible d'ajouter un paramètre à l'exécution de cette méthode pour obtenir le document **après** mise à jour.

Une alternative pour reprendre les informations concernant la visite de Jacques Dupont aurait été la suivante :

```
1 > db.visitors.findOneAndUpdate({firstname: "Dupont"}, {$set: {firstname: "Jacques", lastname:
  "Dupont", updatedAt: ISODate()}}, {returnNewDocument: true})
2 {
3   "_id" : ObjectId("604da2ce558700c6a80bce73"),
4   "firstname" : "Jacques",
5   "lastname" : "Dupont",
6   "startedAt" : ISODate("2021-03-14T05:44:46.694Z"),
7   "visited" : [
8     "SVE",
9     "PWR"
10  ],
11   "updatedAt" : ISODate("2021-03-14T07:43:51.229Z")
12 }
13 >
```

L'objet modifié est alors retourné.



**Remarque**

- C'est bien l'option `{returnNewDocument: true}` qui a permis de récupérer le document modifié. Si cette option n'est pas précisée, c'est le document avant modification qui est retourné.
- En fonction du terminal utilisé pour exécuter les instructions Mongo, cette option peut être inactive. Il faudra alors utiliser le paramètre suivant en remplacement: `{returnOriginal: false}` (utilisation d'un plugin Mongo dans certains IDE par exemple).

Il est également intéressant de connaître un second opérateur de mise à jour : `$unset`. Cet opérateur sera utilisé afin de supprimer un champ (ou plusieurs) d'un document.

Le champ `updatedAt` a été ajouté à notre document concernant la visite réalisée par Jacques Dupont. Finalement, il ne sera pas utile et le choix est fait de supprimer cette information dans notre base de données :

```
1 > db.visitors.findOneAndUpdate({firstname: "Jacques"}, {$unset: {updatedAt: ""}},
2 {
3     "id" : ObjectId("604da2ce558700c6a80bce73"),
4     "firstname" : "Jacques",
5     "lastname" : "Dupont",
6     "startedAt" : ISODate("2021-03-14T05:44:46.694Z"),
7     "visited" : [
8         "SVE",
9         "PWR"
10    ]
11 }
12 >
```

Une liste non exhaustive d'opérateurs de mise à jour, est disponible également au niveau de la documentation officielle<sup>1</sup> :

- `$inc` : pour incrémenter la valeur d'un champ
- `$min` : fait la mise à jour uniquement si la valeur indiquée est plus petite que la valeur actuellement stockée au niveau du document
- `$max` : même fonctionnement qu'avec le min.
- `$rename` : pour renommer un champ (sans impacter sa valeur)
- `$setOnInsert` : pour appliquer une mise à jour uniquement si le document n'existe pas encore. N'est utilisable que si le paramètre `{ upsert: true }` a été passé à la méthode de mise à jour, pour indiquer que l'update doit être remplacé par un insert si le document n'existe pas encore.

## Suppression

Nous souhaitons supprimer la visite stockée en base pour Pierre Robes qui a été créée à tort. Rien de plus simple puisque MongoDB met à notre disposition les méthodes suivantes :

- `deleteOne()`
- `deleteMany()`
- `findOneAndDelete()`

Ces méthodes s'utilisent à l'identique des méthodes de mise à jour équivalentes.

```
1 > db.visitors.deleteOne({firstname: "Pierre"})
2 { "acknowledged" : true, "deletedCount" : 1 }
3 >
```

Si toute la collection doit être supprimée, il faut utiliser la méthode `drop()`

<sup>1</sup> <https://docs.mongodb.com/manual/reference/operator/update/>

```
1 > db.visitors.drop()
2 true
3 >
```

### Syntaxe À retenir

- `insertOne()` et `insertMany()` pour créer un ou plusieurs documents dans une collection
- `updateOne()`, `updateMany()`, `findOneAndUpdate()` pour mettre à jour un ou plusieurs documents dans une collection
- `deleteOne()`, `deleteMany()`, `findOneAndDelete()` pour supprimer un ou plusieurs documents dans une collection
- `find()`, `findOne()` pour récupérer un ou plusieurs documents dans une collection
- `drop()` pour supprimer une collection

### Complément

- Liste de drivers disponibles pour connecter une application Back (API par exemple) à une base MongoDB<sup>1</sup>
- Liste des opérateurs d'update<sup>2</sup>
- Liste des opérateurs logiques et de comparaison<sup>3</sup>
- <sup>4</sup>Documentation officielle sur la méthode `find()`<sup>5</sup>

## VII. Exercice : Appliquer la notion

### Question 1

[solution n°3 p.36]

Jacques Dupont a enfin terminé sa visite.

Connectez-vous à MongoDB via un terminal et exécutez l'instruction permettant d'enrichir le document correspondant à sa visite pour y ajouter la date de fin de visite dans un champ nommé `endedAt`.

#### Indice :

Vous avez le choix entre la méthode `updateOne()` ou la méthode `findOneAndUpdate()`

### Question 2

[solution n°4 p.36]

Supprimez la visite de Martin qui a été créée à tort en base de données.

#### Indice :

`deleteOne()`

## VIII. Compass

1 <https://docs.mongodb.com/drivers/>

2 <https://docs.mongodb.com/manual/reference/operator/update/>

3 <https://docs.mongodb.com/manual/reference/operator/aggregation/>

4 <https://docs.mongodb.com/manual/reference/operator/update/>

5 <https://docs.mongodb.com/manual/tutorial/query-documents/>

## Objectifs

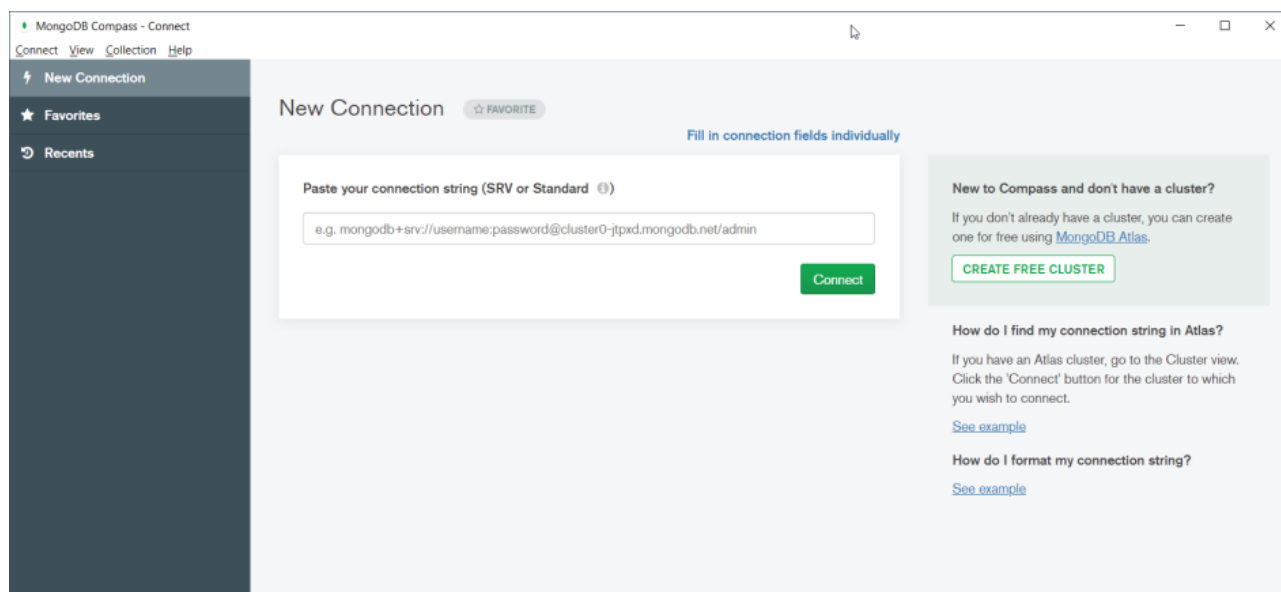
- Se connecter à sa base de données locale avec Compass
- Consulter une collection avec Compass
- Manipulation de documents avec Compass
- Aller plus loin...

## Mise en situation

Il serait intéressant d'avoir un outil visuel qui permette d'interagir avec une base de données MongoDB. En effet, les instructions vues précédemment sont incontournables lorsqu'il s'agit de mettre en place un traitement back pour une application web ou mobile. Elles ne sont pas, par contre, très user-friendly. Compass a été mis en place pour répondre à cette attente.

## Connexion

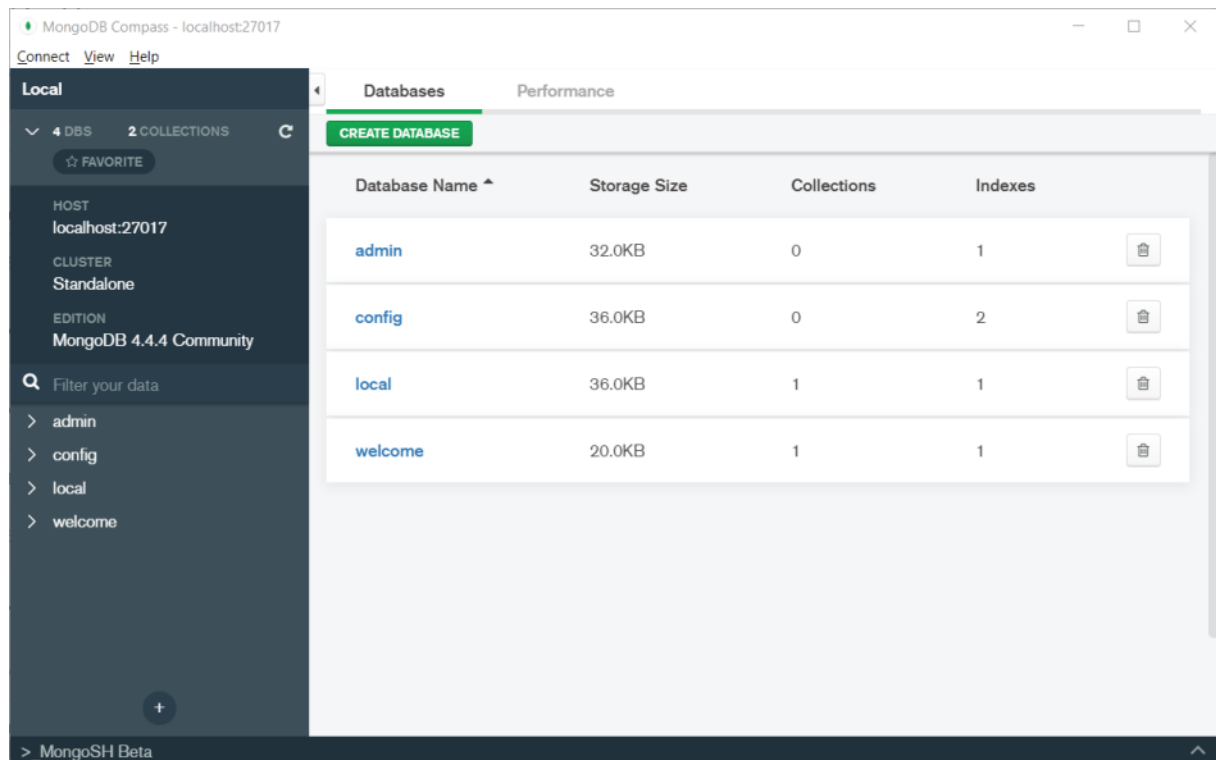
Compass a été installé en même temps que MongoDB. Il suffit alors de rechercher l'application MongoDBCompass et de l'ouvrir.



La connexion vers la base de données MongoDB s'établit en précisant une chaîne de connexion. Lorsqu'il s'agit d'accéder à une base de données locale, la chaîne de connexion sera la suivante :

```
1 mongodb://localhost:27017
```

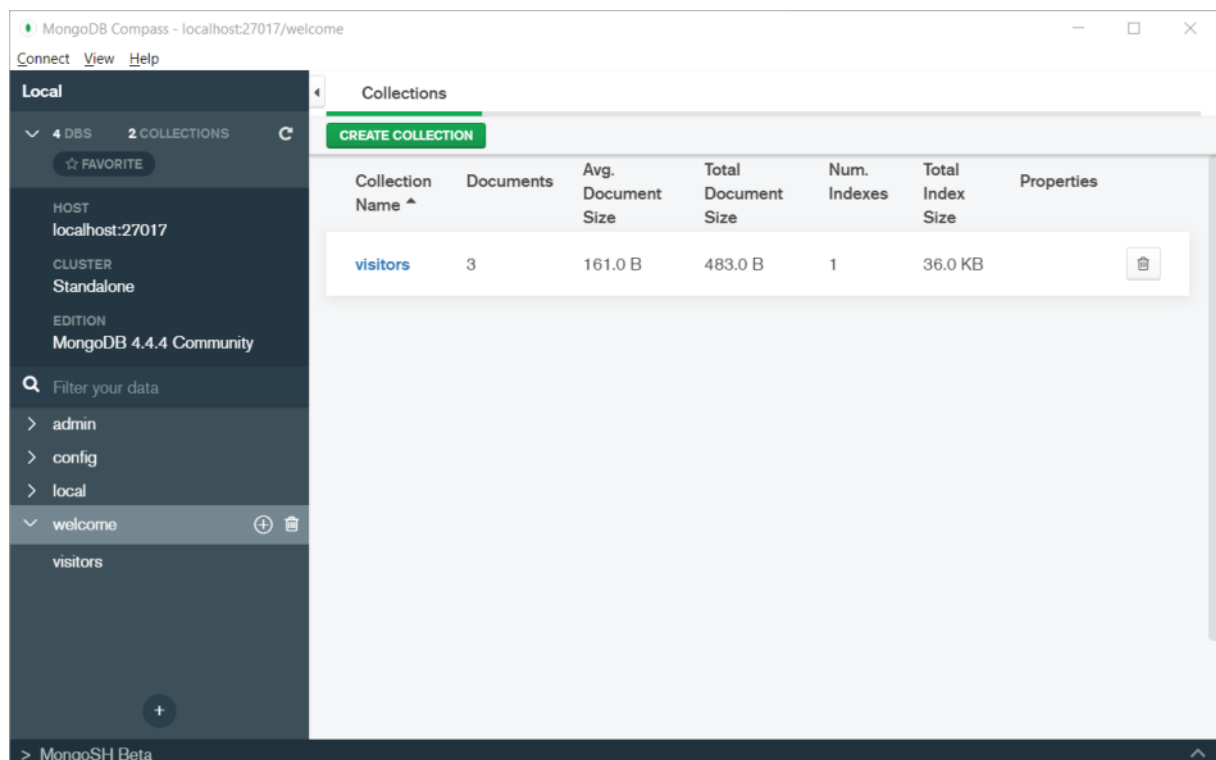
Après avoir validé la connexion, vous pourrez alors accéder aux collections et documents présents dans cette base.



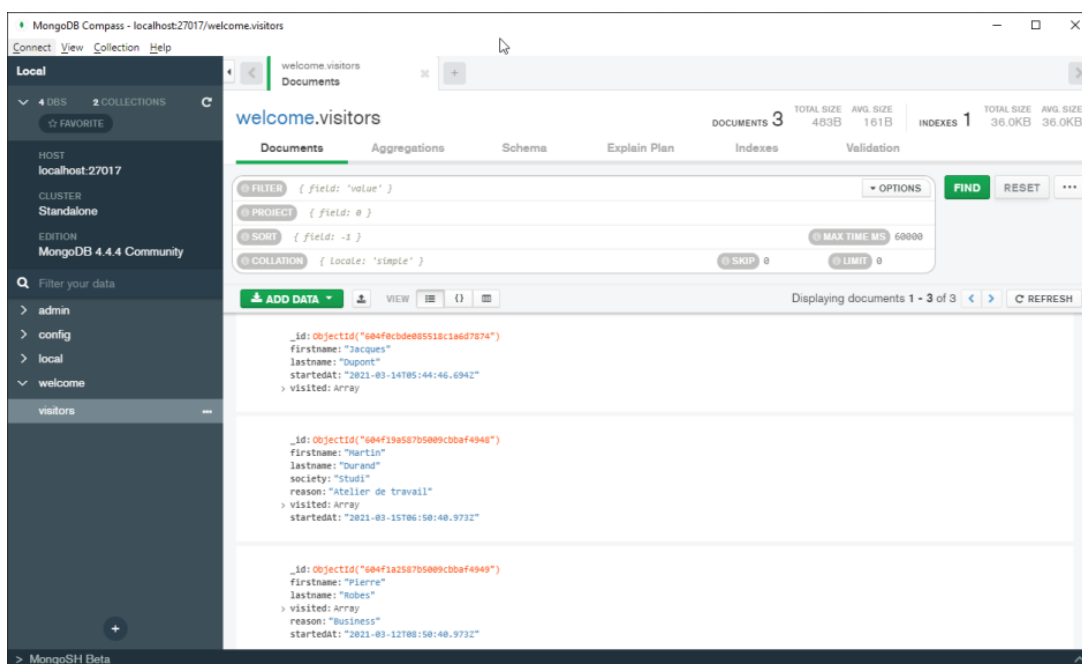
Compass nous permettra toutes les opérations principales pouvant s'appliquer à une base de données, collection ou document.

## Consultation

En cliquant sur `welcome`, Compass nous présente toutes les collections existantes, avec également la possibilité de créer une nouvelle collection.

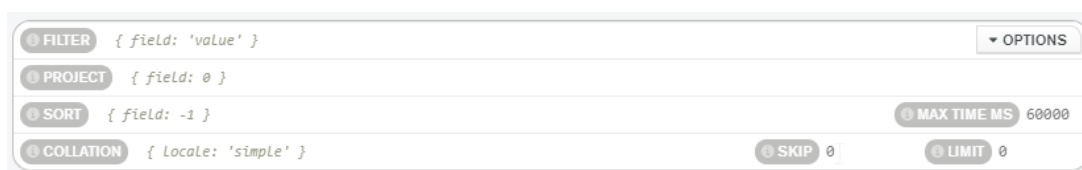


La collection `visitors` se compose de trois documents. Cliquons sur `visitors` pour en voir le détail.



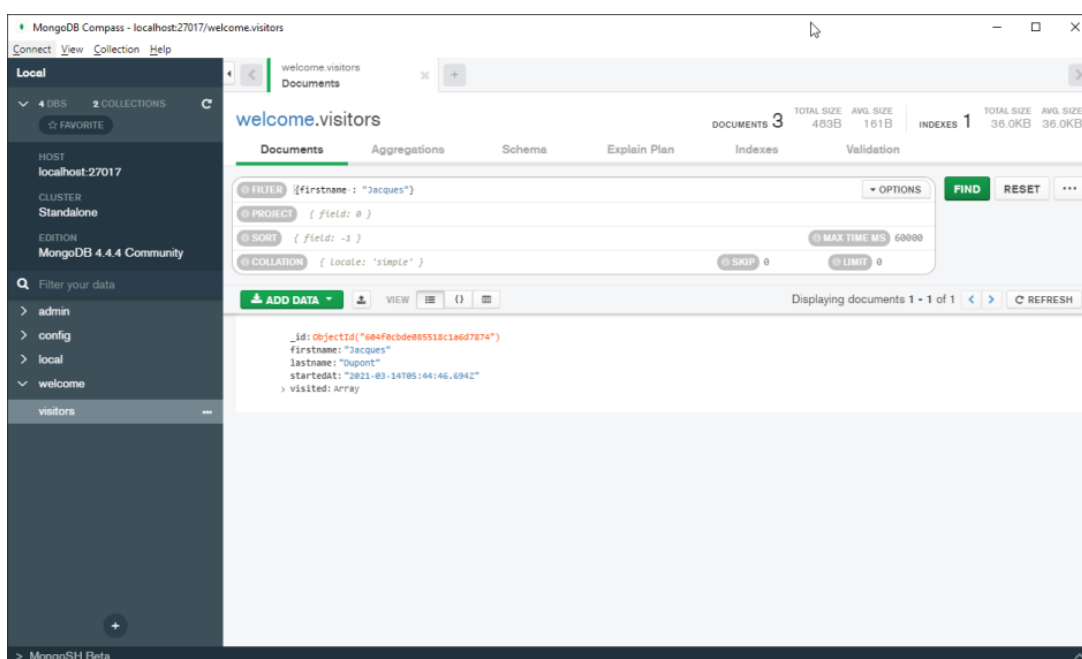
L'affichage des documents est alors beaucoup plus visuel que via un terminal en ligne de commande.

En cliquant sur **OPTIONS**, il sera alors possible d'appliquer des filtres, projections et tris.

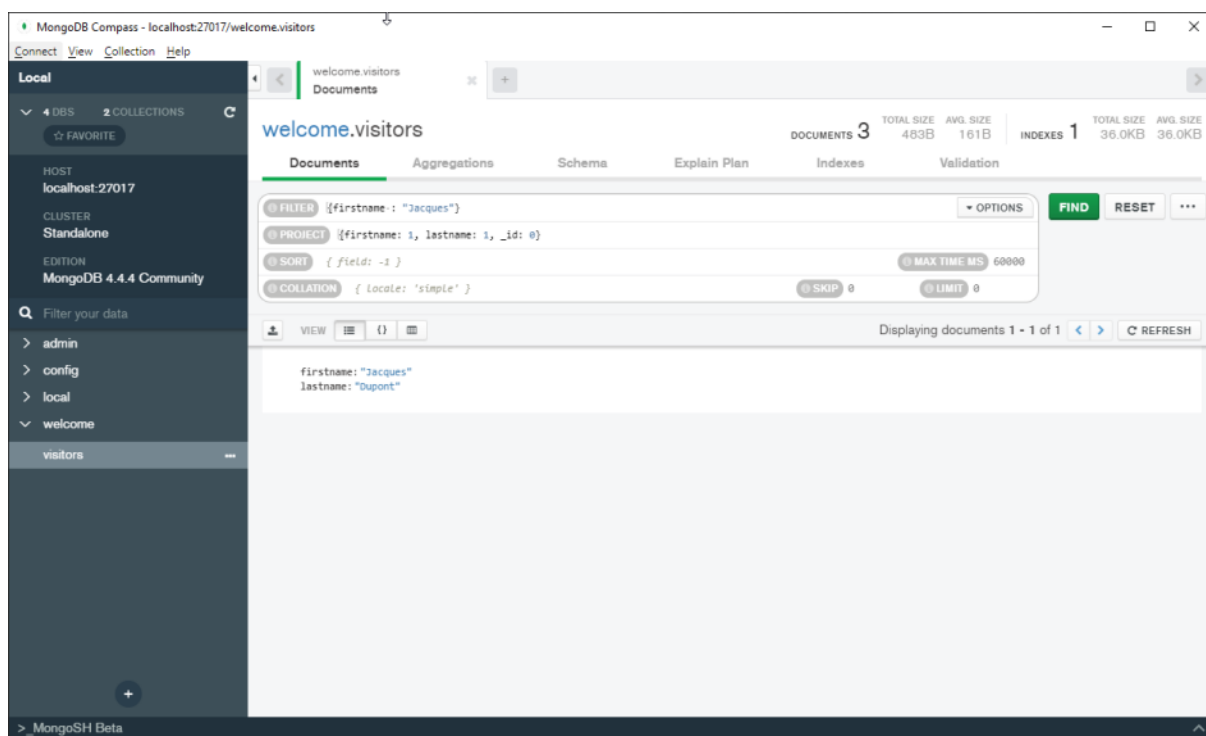


La syntaxe à utiliser est la même que celle que nous avons précédemment vu en utilisant la méthode `find()`.

Pour filtrer sur le visiteur ayant pour prénom Jacques, il est alors possible d'appliquer le filtre `{firstname : "Jacques"}`



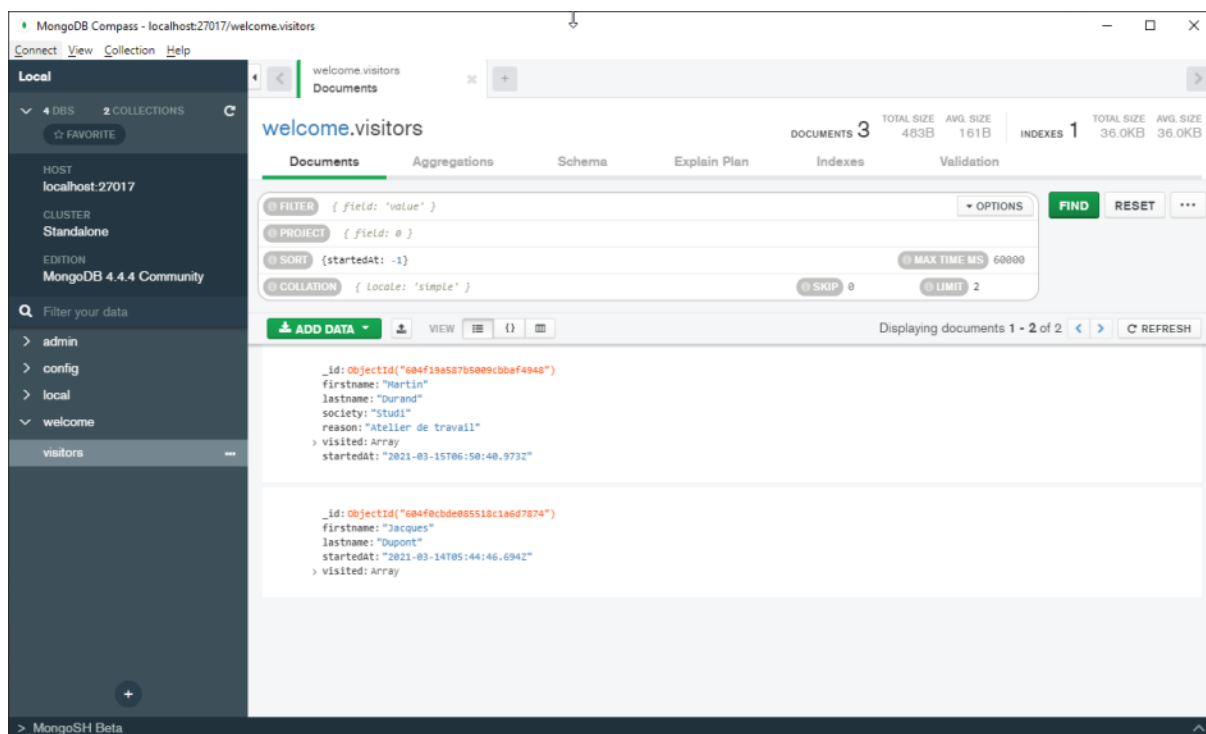
et d'ajouter une projection pour n'obtenir que le nom et prénom :



Le bouton **RESET** permet de supprimer le filtre.

Concernant la possibilité de trier, il suffit de préciser les critères de tri, avec pour valeur 1 pour trier de manière croissante, et -1 pour un tri décroissant.

Ci-dessous la liste des visiteurs triés par date décroissante et avec une limite de résultat fixée à deux documents.



**Syntaxe**   **À retenir**

- Compass permet de visualiser plus simplement les informations
- Compass permet de faire toutes les mises à jour possibles sur les données sans connaître toutes les méthodes vues précédemment

**Exercice : Appliquer la notion**

[solution n°5 p.36]

Exercice

Le port par défaut pour MongoDB est le port 27017.

- ☐ Vrai
- ☐ Faux

Exercice

Compass ne permet pas de se connecter à une base de données distante.

- ☐ Vrai
- ☐ Faux

**X. Essentiel****XI. Auto-évaluation****A. Exercice final****Exercice 1**

[solution n°6 p.37]

Exercice

MongoDB est une base de données orientée documents.

- ☐ Vrai
- ☐ Faux

Exercice

Une base de données NoSQL doit être utilisée uniquement lorsqu'il y a de gros volumes d'informations à stocker, afin de faciliter la scalabilité horizontale de notre application.

- ☐ Vrai
- ☐ Faux

Exercice

L'instruction `db.visitors.find({})` retourne ...

- ☐ Tous les documents présents dans la collection `visitors`.
- ☐ Uniquement le premier document de la collection `visitors`.

Exercice

L'opérateur `$set` permet d'indiquer les informations à mettre à jour lorsqu'on utilise la méthode `updateOne()`.

- ☐ Vrai
- ☐ Faux

#### Exercice

L'opérateur `$unset` permet de réinitialiser la valeur d'un champ avec une chaîne de caractères vide.

- ☐ Vrai
- ☐ Faux

#### Exercice

Parmi ces commandes de recherche, quelle(s) sont celle(s) qui sont correcte(s) ?

- ☐ `db.visitors.find({$and: [{society: "Banque de France"}, {visited: "SVE"}]})`
- ☐ `db.visitors.find({firstname: "Pierre", $and: [{society: "Studi"}, {visited: "SVE"}]})`
- ☐ `db.visitors.find({$nor: {visited: "PBN"}, {visited: "SVE"}})`
- ☐ `db.visitors.find({$or: [{society: "Banque de France"}, {society: "Studi"}]})`

#### Exercice

Quelle est la méthode utilisée pour supprimer une collection ?

- ☐ `delete()`
- ☐ `suppr()`
- ☐ `drop()`

#### Exercice

Quels sont les opérateurs logiques acceptables dans une requête MongoDB ?

- ☐ `$and`
- ☐ `$first`
- ☐ `$last`
- ☐ `$or`
- ☐ `$nor`
- ☐ `$for`
- ☐ `$not`
- ☐ `$eq`

#### Exercice

Quels sont les opérateurs de comparaison acceptables dans une requête MongoDB ?



- ☐ \$eq
- ☐ \$gta
- ☐ \$lt
- ☐ \$nin
- ☐ \$lte
- ☐ \$gte
- ☐ \$ne
- ☐ \$in
- ☐ \$gt
- ☐ \$null

### Exercice

Quels sont les types de données valables en BSON ?

- ☐ Double
- ☐ Triple
- ☐ String
- ☐ ObjectID
- ☐ Short
- ☐ Int
- ☐ Long
- ☐ Array
- ☐ Géolocalisation
- ☐ Date

### B. Exercice : Défi

Les visites suivantes ont été réalisées dans l'entreprise et vous devez les stocker dans la collection `visitors` de la base de données `welcome`.

Commencez par exécuter l'instruction suivante pour ajouter toutes ces visites en base de données.

```
1 db.visitors.insertMany([
2   {firstname: "Arthur", lastname: "Péachepé", startedAt: "2021-02-10T14:28:22.273Z",
  visited: ["SVE"], endedAt: "2021-02-10T16:32:22.273Z"},
3   {firstname: "Benjamin", lastname: "Swift", startedAt: "2021-03-15T11:17:22.273Z", visited:
  ["PWR"], endedAt: "2021-03-15T13:32:22.273Z"},
4   {firstname: "Charles", lastname: "Java", startedAt: "2021-03-09T15:32:22.273Z", visited:
  ["PWR", "ERR"], endedAt: "2021-03-09T16:28:22.273Z"},
5   {firstname: "Daniela", lastname: "Flutter", startedAt: "2021-03-17T09:28:22.273Z",
  visited: ["SVE"], endedAt: "2021-03-17T11:40:22.273Z"},
6   {firstname: "Erik", lastname: "Spoutnik", startedAt: "2021-03-15T09:53:52.273Z", visited:
  ["SVE", "SRU"], endedAt: "2021-03-15T17:52:52.273Z"},
7   {firstname: "Franck", lastname: "Vincent", startedAt: "2021-03-15T17:30:22.273Z", visited:
  ["SVE"], endedAt: "2021-03-15T17:40:22.273Z"},
8   {firstname: "Géraldine", lastname: "Colo", startedAt: "2021-03-15T17:18:22.273Z", visited:
  ["SVE", "FB0"], endedAt: "2021-03-15T19:32:22.273Z"},
```

```

9   {firstname: "Hector", lastname: "Cé", startedAt: "2021-02-14T11:18:22.273Z", visited:
["JSN"], endedAt: "2021-02-14T15:59:22.273Z"},
10  {firstname: "Ilyana", lastname: "péaud", startedAt: "2021-01-03T09:28:22.273Z", visited:
["SVE"], endedAt: "2021-01-03T10:43:22.273Z"},
11  {firstname: "Justine", lastname: "Daniel", startedAt: "2021-01-03T15:45:22.273Z", visited:
["PWR"], endedAt: "2021-01-03T18:00:22.273Z"},
12  {firstname: "Kévin", lastname: "Water", startedAt: "2021-01-18T14:28:22.273Z", visited:
["BHT"], endedAt: "2021-01-18T16:44:22.273Z"},
13  {firstname: "Laura", lastname: "Peausi", startedAt: "2021-03-17T11:12:22.273Z", visited:
["SVE"]},
14  {firstname: "Marie", lastname: "Paul", startedAt: "2021-03-17T10:00:22.273Z", visited:
["ERR"]}]
15  ])
```

### Question 1

[solution n°7 p.40]

Quelle instruction permet de retrouver le nombre de visites réalisées ?

#### Indice :

Ne pas oublier de se positionner sur la bonne base de données avec l'instruction `use`

#### Indice :

`count()` devrait pouvoir répondre au besoin.

### Question 2

[solution n°8 p.40]

Affichez les visites ayant été réalisées depuis le 15/03/2021.

### Question 3

[solution n°9 p.40]

Affichez les visites ayant été réalisées le 15/03/2021.

#### Indice :

`$and`

### Question 4

[solution n°10 p.40]

Affichez toutes les visites qu'a reçu "SVE".

### Question 5

[solution n°11 p.40]

Affichez toutes les visites qu'a reçu "SVE" et uniquement lui.

### Question 6

[solution n°12 p.40]

Affichez toutes les visites en cours (sans champ `endedAt`), par utilisation de l'opérateur `exists`<sup>1</sup> :

#### Indice :

Suivez le lien en cliquant sur `exists`, et vous devriez avoir toute l'aide nécessaire.

## Solutions des exercices


<sup>1</sup> <https://docs.mongodb.com/manual/reference/operator/query/exists/>

**Exercice p. 17 Solution n°1****Exercice**

Il est possible d'installer MongoDB en local sur son poste, quel que soit son système d'exploitation.

☒ Vrai

☐ Faux


 Il est tout à fait possible d'installer MongoDB quel que soit son OS : Windows, Linux, ou MacOS.

**Exercice**

Compass est un outil indispensable pour manipuler une base de données MongoDB.

☐ Vrai

☒ Faux


 Une base MongoDB peut tout à fait être manipulée sans outil graphique tel que Compass, qui apporte néanmoins du confort en termes d'accès ou manipulation des informations. Nous reviendrons sur cet outil en fin de cours, puisqu'on privilégiera dans un premier temps le Mongo Shell pour bien comprendre les notions de base.

**Exercice**

Mon installation s'est réalisée sur mon poste avec succès.

☒ Vrai

☐ Faux


 Il est important que l'installation soit terminée avant de poursuivre sur la suite du cours.

**Exercice p. 19 Solution n°2****Exercice**

MongoDB stocke les informations au format JSON, ce qui est bien pratique pour interagir avec des API REST.

☐ Vrai

☒ Faux

 Même si les documents se représentent sous la forme d'un objet JSON, les informations sont stockées en base au format BSON, Binary JSON.

**Exercice**

Le BSON a l'avantage de fournir une très grande liste de types possibles pour nos données, bien plus que les types proposés par JSON.

☒ Vrai

☐ Faux

- Q BSON est un format étendant les possibilités du JSON avec, par exemple, des données temporelles, binaires et bien d'autres !

### Exercice

Un Document...

- ☒ se représente sous la forme d'un JSON, et se compose de paires champ/valeur
- ☐ est stocké dans une table MongoDB  
*Il n'y a pas de notion de table dans une base NOSQL MongoDB.*
- ☒ doit être créé dans une collection
- ☐ peut contenir une infinité de champs  
*Il ne peut pas excéder 16Mo.*

#### p. 26 Solution n°3

```
1 > db.visitors.updateOne({firstname:"Jacques"},{$set: {endedAt: ISODate()}})
2 { "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
3 > db.visitors.find({firstname:"Jacques"})
4 { "_id" : ObjectId("604f0cbde085518c1a6d7874"), "firstname" : "Jacques", "lastname" :
  "Dupont", "startedAt" : "2021-03-14T05:44:46.694Z", "visited" : [ "SVE", "PWR" ], "endedAt" :
  ISODate("2021-03-15T14:28:22.273Z") }
5 >
```

#### p. 26 Solution n°4

```
1 > db.visitors.deleteOne({firstname:"Martin"})
2 { "acknowledged" : true, "deletedCount" : 1 }
3 > db.visitors.find({firstname:"Martin"}).count()
4 0
5 >
```

#### Exercice p. 31 Solution n°5

### Exercice

Le port par défaut pour MongoDB est le port 27017.

☒ Vrai

☐ Faux


- Q Bien qu'il soit possible de le changer et d'en choisir un à sa guise, le port 27017 est le port utilisée par défaut par MongoDB.

### Exercice

Compass ne permet pas de se connecter à une base de données distante.

☐ Vrai

☒ Faux

-  Pour se connecter à une base de données distante, il suffira de modifier la chaîne de connexion au moment de la création de la connexion pour indiquer l'IP de la machine contenant la base de données.


### Exercice p. 31 Solution n°6

#### Exercice

MongoDB est une base de données orientée documents.

☒ Vrai

☐ Faux


-  Les documents sont des fichiers au format BSON.

#### Exercice

Une base de données NoSQL doit être utilisée uniquement lorsqu'il y a de gros volumes d'informations à stocker, afin de faciliter la scalabilité horizontale de notre application.

☐ Vrai

☒ Faux


-  Les bases NOSQL sont très adaptées pour gérer d'énormes volumes d'informations, mais peuvent convenir parfaitement pour les plus petits volumes, lorsque les informations à stocker dans l'application ne suivent pas un schéma strict.

#### Exercice

L'instruction `db.visitors.find({})` retourne ...

☒ Tous les documents présents dans la collection `visitors`.

☐ Uniquement le premier document de la collection `visitors`.


-  La méthode `find()` retourne tous les documents vérifiant une condition. La condition `{}` n'indiquant aucune restriction, tous les documents sont alors retournés.

#### Exercice

L'opérateur `$set` permet d'indiquer les informations à mettre à jour lorsqu'on utilise la méthode `updateOne()`.

☒ Vrai

☐ Faux


-  La méthode `updateOne()` permet, comme on l'a vu, de modifier un document de nos collections. Si le premier paramètre de `updateOne()` permet d'ajouter des filtres pour trouver le fichier à modifier, l'opérateur `$set` arrive à la suite et permet de spécifier les données à modifier sur ce document. Dans le cas où les critères de recherches trouveraient plusieurs documents, les éléments modificateurs spécifiés dans `$set` seront appliqués au premier document de la liste.

#### Exercice

L'opérateur `$unset` permet de réinitialiser la valeur d'un champ avec une chaîne de caractères vide.

☐ Vrai

☒ Faux

 Cet opérateur ne réinitialise pas un champ à la valeur "", mais le supprime du document.

### Exercice


Parmi ces commandes de recherche, quelle(s) sont celle(s) qui sont correcte(s) ?

- ☒ `db.visitors.find({$and: [{society: "Banque de France"}, {visited: "SVE"}]})`  
Cette commande fonctionnera.
- ☒ `db.visitors.find({firstname: "Pierre", $and: [{society: "Studi"}, {visited: "SVE"}]})`  
Cette commande fonctionnera.
- ☐ `db.visitors.find({$nor: {visited: "PBN"}, {visited: "SVE"}})`  
Cette commande ne fonctionnera pas, il manque des crochets.
- ☒ `db.visitors.find({$or: [{society: "Banque de France"}, {society: "Studi"}]})`  
Cette commande fonctionnera.

### Exercice

Quelle est la méthode utilisée pour supprimer une collection ?

- ☐ `delete()`
- ☐ `suppr()`
- ☒ `drop()`

 La méthode utilisée pour supprimer une collection est `drop()`.

### Exercice

Quels sont les opérateurs logiques acceptables dans une requête MongoDB ?

- ☒ `$and`  
Cet opérateur est un opérateur logique accepté dans les requêtes MongoDB.
- ☐ `$first`  
Cet opérateur n'est pas un opérateur logique accepté dans les requêtes MongoDB.
- ☐ `$last`  
Cet opérateur n'est pas un opérateur logique accepté dans les requêtes MongoDB.
- ☒ `$or`  
Cet opérateur est un opérateur logique accepté dans les requêtes MongoDB.
- ☒ `$nor`  
Cet opérateur est un opérateur logique accepté dans les requêtes MongoDB.
- ☐ `$for`  
Cet opérateur n'est pas un opérateur logique accepté dans les requêtes MongoDB.
- ☒ `$not`  
Cet opérateur est un opérateur logique accepté dans les requêtes MongoDB.

☐ \$eq

*Cet opérateur n'est pas un opérateur logique. Il est accepté par MongoDB mais c'est un opérateur de comparaison.*

**Exercice**

---

Quels sont les opérateurs de comparaison acceptables dans une requête MongoDB ?

☒ \$eq

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **égal à**.*

☐ \$gta

*Cet opérateur n'est pas un opérateur de comparaison acceptable.*

☒ \$lt

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **inférieur à**.*

☒ \$nin

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **n'inclue pas**.*

☒ \$lte

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **inférieur ou égal à**.*

☒ \$gte

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **supérieur ou égal à**.*

☒ \$ne

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **non égal**.*

☒ \$in

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **inclus**.*

☒ \$gt

*Cet opérateur est bien un opérateur de comparaison acceptable et signifie : **supérieur à**.*

☐ \$null

*Cet opérateur n'est pas un opérateur de comparaison acceptable.*

**Exercice**

---

Quels sont les types de données valables en BSON ?

☒ Double

*C'est un type de données valable en BSON*

☐ Triple

*Ce type de données n'existe pas en BSON*

☒ String

*C'est un type de données valable en BSON*

☒ ObjectID

*C'est un type de données valable en BSON*

☐ Short

*Ce type de données n'existe pas en BSON*

- ☒ Int  
*C'est un type de données valable en BSON*
- ☒ Long  
*C'est un type de données valable en BSON*
- ☒ Array  
*C'est un type de données valable en BSON*
- ☐ Géolocalisation  
*Ce type de données n'existe pas en BSON*
- ☒ Date  
*C'est un type de données valable en BSON*

**p. 34 Solution n°7**

```
1 db.visitors.find().countDocuments()
```

**p. 34 Solution n°8**

```
1 db.visitors.find({startedAt: {$gt: "2021-03-15"}})
```

**p. 34 Solution n°9**

```
1 db.visitors.find({$and:[{startedAt: {$gte: "2021-03-15"}},{startedAt: {$lt: "2021-03-16"}}]})
```

**p. 34 Solution n°10**

```
1 db.visitors.find({visited: "SVE"})
```

**p. 34 Solution n°11**

```
1 db.visitors.find({visited: ["SVE"]})
```

**p. 34 Solution n°12**

```
1 db.visitors.find({endedAt : {$exists: false}})
```