

# LINFO1212

Projet d'approfondissement en informatique

## Persistence: MongoDB

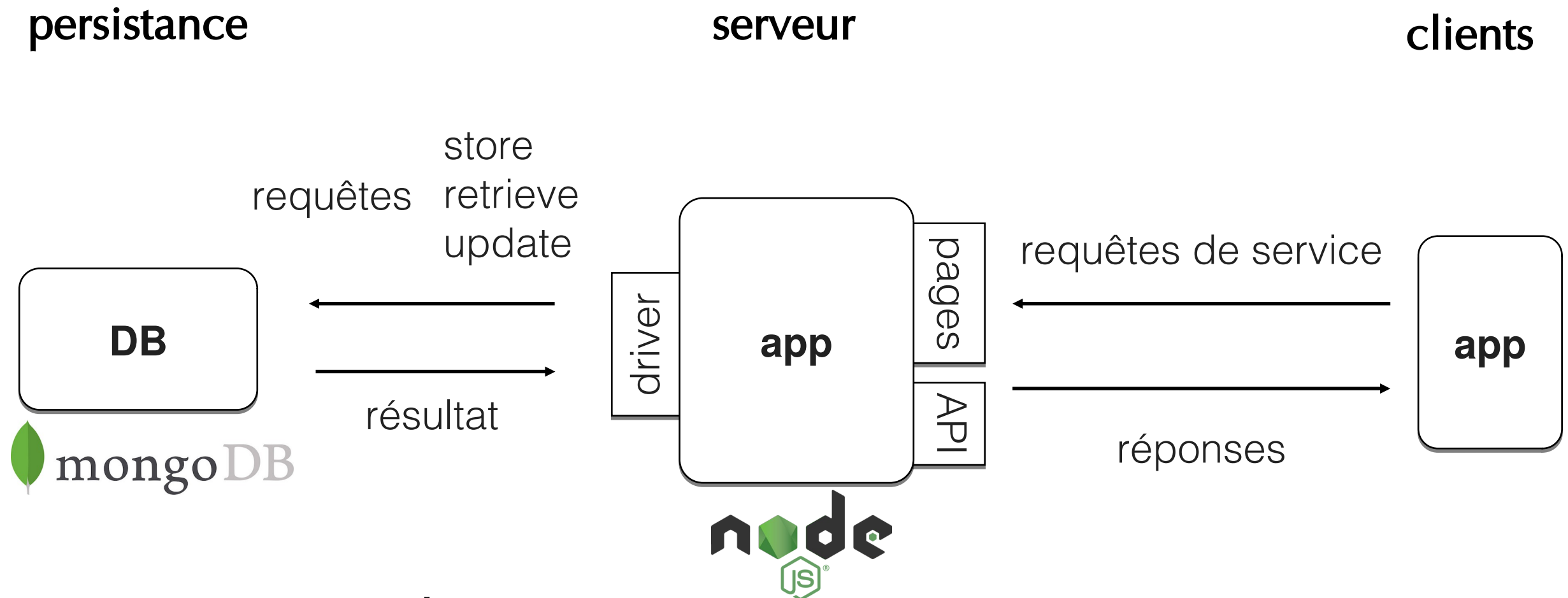
Siegfried Nijssen

*Contributions: Virginie Van den Schrieck, Sebastián González*

26 octobre 2019

# Architecture trois tiers

2



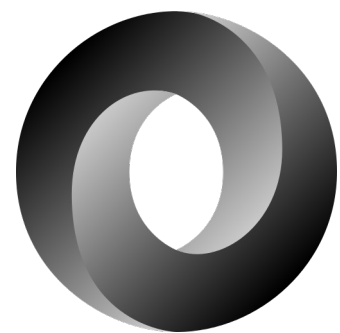
## Architecture trois tiers :

- Couche de présentation (interface d'utilisateur)
- Couche du métier (modèle)
- Couche des données (persistance)

Extension de l'architecture client–serveur

La plus utilisée pour les applications web

- Base de données « NoSQL » assez répandue
- Stockage de données sous forme de **documents**, en utilisant la “JavaScript Object Notation” (JSON)  
(Sous-ensemble de la syntaxe JavaScript pour objets)
- Une interface programmatique en JavaScript

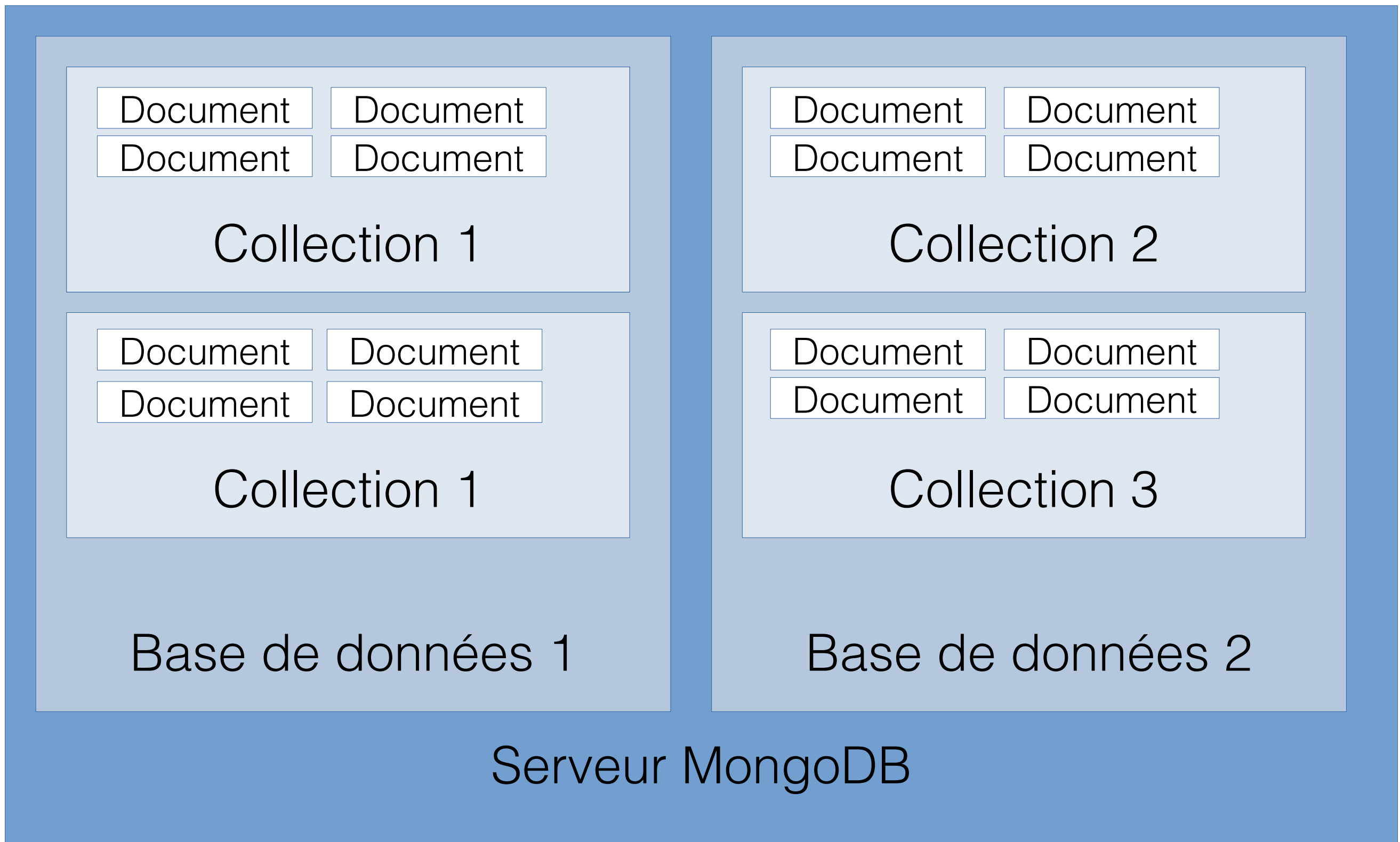


# JSON

Chaîne  
de caractères

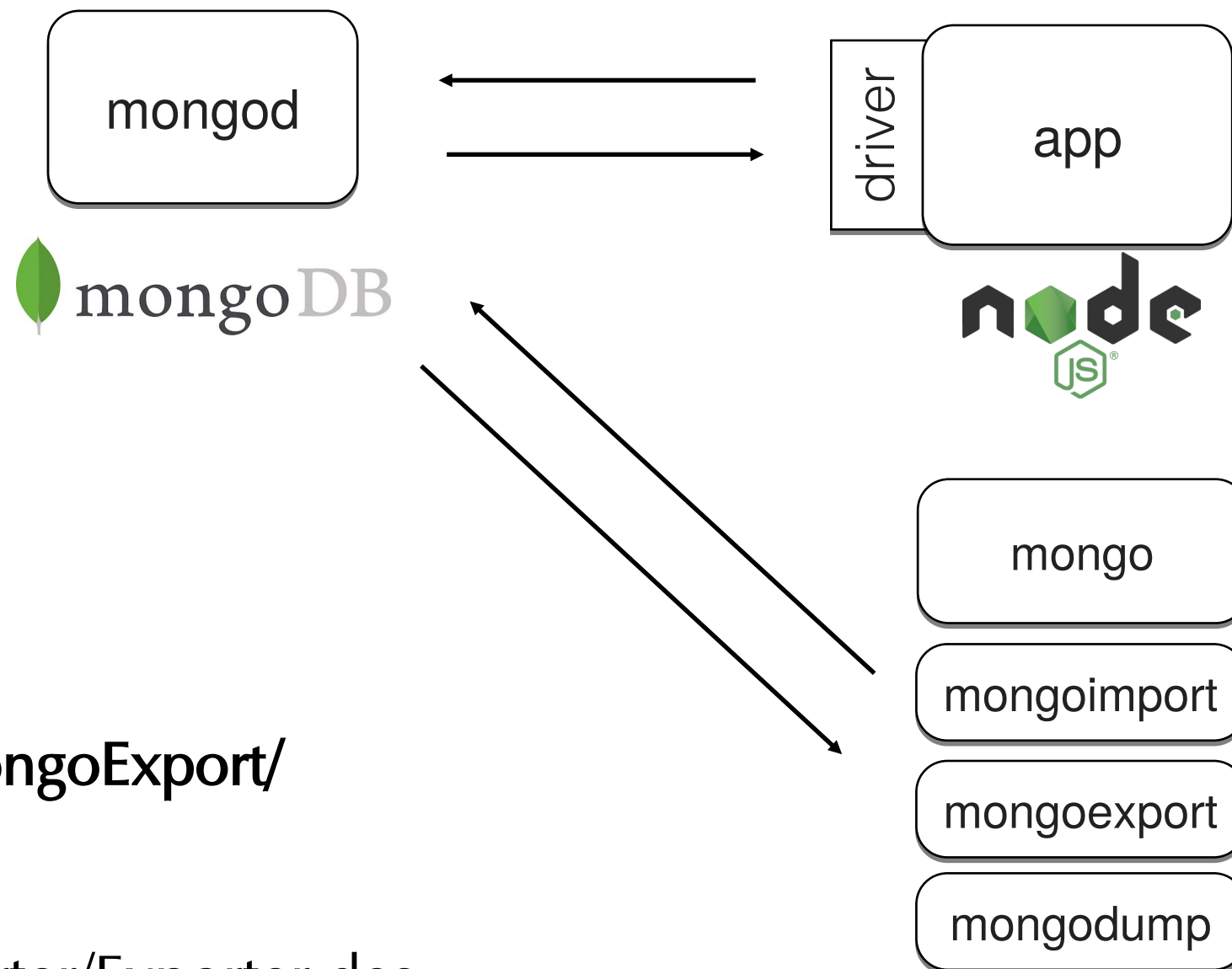
## UN DOCUMENT

```
{
  "class" : "algebra",
  "students" : [
    {
      "name" : "Susan",
      "activities" : [
        {
          "name" : "soccer",
          "type" : "sport"
        },
        {
          "name" : "band",
          "type" : "music"
        }
      ]
    }
  ]
}
```



# MongoDB et autres clients

9



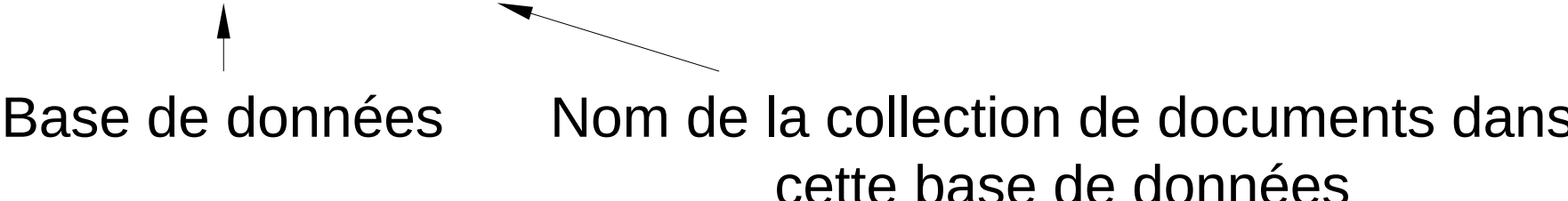
**MongoImport/MongoExport/  
MongoDump**

Outils pour Importer/Exporter des  
données

**Mongo: Console d'administration**

**MongoD**

- Écrit en C++.
- Lancement en ligne de commandes :  
> `mongod --dbpath db/`

- Exécution sur la ligne de commande
- Importation d'une base de données :
  - > mongoimport -d course -c grades grades.json

Base de données      Nom de la collection de documents dans cette base de données
- Exportation d'une base de données sous format JSON, [CSV](#) et [TSV](#) :
  - > mongoexport -d course -c grades

- Commandes données dans l'outil mongo  
(Pas sur la ligne de commande de votre système d'exploitation)
- Base de données courante :  
    > db
- Voir bases de données disponibles :  
    > show dbs
- Changer la base de données courante :  
    > use course
- Consulter le contenu d'une collection en utilisant une notation JavaScript:  
    > db.grades.find()
- Insérer un document JSON dans une collection :  
    > db.grades.insert({ "student": "Steve", "assignment": "hw4", "grade": 100 })
- Utiliser un document comme « descripteur de requête » :  
    > db.grades.find({ "student": "Steve" })

# Opérations de base pour la persistance des données

<u>CRUD</u>	MongoDB	HTTP
<b>C</b> reate	Insert	POST
<b>R</b> ead	Find	GET
<b>U</b> ppdate	Update	PUT
<b>D</b> elete	Remove	DELETE

En MongoDB, ces opérations de base existent en tant qu'API d'une librairie (le « driver ») en NodeJS



**C**RU**D** : Création des données

```
db.inventory.insert( { _id: 10, item: "card", qty: 15 } )
```

```
> db.inventory.update({ type: "book", item : "journal" },  
    { $set : { qty: 10 } })
```

**CRUD : Lecture des données**

## **findOne()**

- Lecture d'un document choisi aléatoirement :
  - > `db.people.findOne()`
- Lecture d'un document qui a un champ et valeur donnés :
  - > `db.people.findOne({ name: "Jones" })`

## **find()**

- Lecture de tous les documents présents dans une collection :
  - > `db.people.find()`
- Lecture des documents qui ont un champ et valeur donnés :
  - > `db.people.find({ name: "Jones", type: "employee" })`

## Inégalités

- Sélection d'âge  $> 18$  (**greater than**) :
  - > `db.people.find({ age: { $gt: 18 } })`
- Sélection d'âge  $> 18$  et statut d'employé :
  - > `db.people.find({ age: { $gt: 18 }, type: "employee" })`
- Sélection d'âge  $> 18$  **et** d'âge  $\leq 65$  :
  - > `db.people.find({ age: { $gt: 18, $lte: 65 } })`
- Autres opérateurs dans la même famille : `$gte` ( $\geq$ ), `$lt` ( $<$ ).

**CRUD : Mise à jour des données**

## Substitution

- Le premier paramètre est une interrogation, le deuxième paramètre est le document qui remplace le document existant (sauf pour l'ObjectId):
  - `db.people.update({ type: "employee" }, { name: "Jones", type: "employee" });`

## Mise à jour des champs

- Affecter une nouvelle âge à Tom (ou la créer si pas existante) :
  - > `db.people.update({ name: "Tom" }, { $set: { age: 30 } })`

**CRUD : Élimination des données**



- Élimination par l'exemple :
  - > `db.people.remove({ "name": "Tom" })`
- Élimination avec opérateurs d'interrogation :
  - > `db.people.remove({ "name": { $gt: "M" } })`
- Élimination d'une collection entière :
  - > `db.people.drop()`

- Code **synchrone** à la console mongo :

```
let doc = db.coll.findOne();
```

- On ne peut donner une autre commande que après la commande précédente a terminée

- Code **asynchrone** en node.js :

```
app.get('/', (req, res) => {  
  dbo.collection('grades').findOne({student:"Amanda"}, (err, doc) => {  
    res.render('mongodb-express', doc);  
  });  
});
```

- On ne s'intéresse pas à la valeur renvoyée par un appel asynchrone.
- Le callback reçoit le résultat et l'utilise dès qu'il est disponible.