# Mongodb


- MongoDB est un système de gestion de base de données NoSQL, auquel il est facile d'accéder depuis Node.js en raison de son format de stockage, le BSON (du JSON binaire).
- Une base MongoDB est structurée en collections, composées elles-mêmes de documents.
- Chaque document est un objet JavaScript, mais toutes les propriétés n'ont pas besoin d'être stockées.
- Cela permet de gagner de la place sur l'espace de stockage nécessaire pour la base de données.

- ***Exemple*** :

{ "_id" : ObjectId("5c5e0511f59b8121641bbe21"), "nom" : "Tounsi", "age": 34, "prenom": "Mohamed" }

{ "_id" : ObjectId("5c5e0ba614128f24f872a0a1"), "nom" : "ElArbi", "age": 23 }

{ "_id" : ObjectId("5c5ee355cd9c200710bdabf6"), "nom" : "Gabsi", age: 37 }

{ "_id" : ObjectId("5c5ee35ccd9c200710bdabf7"), "nom" : "Sfaxi", "prenom": "Hèdia" }

***Tel que _id est une propriété générée automatiquement par MongoDB et correspond à une clef primaire.***

# Installer MongoDB

- Pour télécharger et installer le pilote officiel MongoDB, ouvrir le terminal de commande et exécuter ce qui suit qui permet de télécharger et d'installer le package mongodb :

**npm install mongodb --save**

- Node.js peut utiliser ce module pour manipuler les bases de données MongoDB :

```
var mongo = require('mongodb');
```

# Création d’une base de données

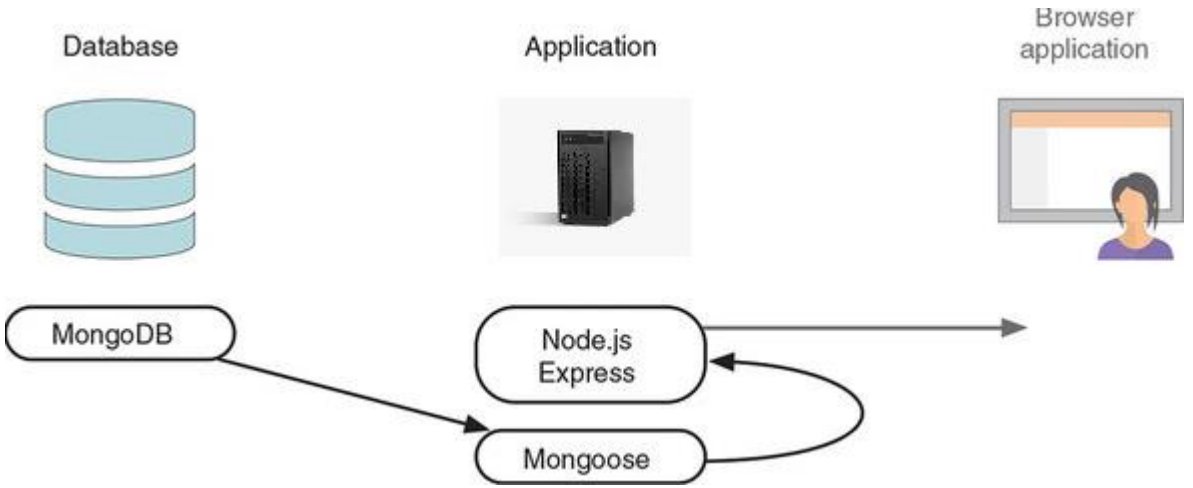
- Pour créer une base de données dans MongoDB, il faut commencer par créer un objet MongoClient, puis spécifier une URL de connexion avec l'adresse IP correcte et le nom de la base de données qu’on souhaite créer.
- MongoDB créera la base de données si elle n'existe pas et établira une connexion avec elle.

- ***Exemple*** :

```
// utiliser le driver mongodb
const MongoClient = require('mongodb').MongoClient
//spécifier l'URL de la connexion
const url = 'mongodb://127.0.0.1/base1'
//connexion à la base de données
MongoClient.connect(url, {
    useNewUrlParser: true,
    useUnifiedTopology: true
  }, (err, client) => {
  if (err) {
    console.error(err)
    return
  }
console.log("Connecté à la base de données") ;
 //sélectionner une base de données en utilisant la méthode client.db()
const db = client.db('base1')
});
```

# Mongoose

- Mongoose est ce qui va permettre à Node.js de communiquer avec le serveur MongoDB : L'application Node / Express interagit avec MongoDB via Mongoose.




- Pour cela, et afin de pouvoir l'utiliser, il faut installer Mongoose.

**npm install mongoose --save**

- Une fois le module installé, on peut l'utiliser au sein du code en faisant appel au module :

**var mongoose = require('mongoose');**

- Une fois mongoose appelé, il faut instancier la connexion :

```
mongoose.connect('mongodb://localhost', function(err) {   if (err) { throw err; }   });
```

# Les schémas

- Mongoose utilise des « Schemas » pour modéliser les données.
- Il permet de définir les types de variables et de structurer les données.
- Pour créer un Schema avec Mongoose, il suffit d'utiliser l'objet **Schema** :
- ***Exemple*** :

```
var commentaireArticleSchema = new mongoose.Schema({
pseudo : { type : String, match: /^[a-zA-Z0-9-_]+$/ },
contenu : String,
date : { type : Date, default : Date.now }
});
```

- On peut aussi faire en sorte que des valeurs soient aussi des schémas.
- ***Exemple*** :

```
var articleSchema = new mongoose.Schema({
  auteur : Schema.ObjectId,
  contenu : String,
  date : { type : Date, default : Date.now },
  commentaires : [ commentaireArticleSchema ],
  votes : {
    plus : { type : Number, min : 0 },
    moins : { type : Number, min : 0 }
  }
});
```

# Les Models

- Un model permet d'insérer des données dans MongoDB en respectant le schéma précisé et d’utiliser des requêtes dessus.
- Pour créer un model :

```
var CommentaireArticleModel = mongoose.model('commentaires',
commentaireArticleSchema);
```

- Cela va créer un model nommé « commentaires » à partir du schéma CommentaireArticleSchema et qui va le renvoyer dans la variable CommentaireArticleModel.
- On peut également, créer le model dans un fichier .js en incluant Mongoose et en le récupérant dans un autre fichier en appelant :

```
var CommentaireArticleModel = mongoose.model('commentaires');
```

- Le model va créer une collection automatiquement nommée « commentaires » (si elle n'existe pas dès qu'un élément sera inséré dedans).

# Les instances de Model

- Pour créer une instance de Model, il suffit de faire :

```
var monCommentaire = new CommentaireArticleModel({ pseudo : 'Alio' });
// On rajoute le contenu du commentaire, possible de le faire lors de l'instanciation
monCommentaire.contenu = 'Salut, super article sur Mongoose !';
```

- Cela va créer une instance du Model CommentaireArticleModel, cependant il n'est pas encore sauvegardé dans MongoDB, pour cela il suffit d'appeler la méthode **save**() de l'instance créée.

```
monCommentaire.save(function (err) {
  if (err) { throw err; }
  console.log('Commentaire ajouté avec succès !');
});
```

- ***Exemple*** :

```
// Inclusion de Mongoose
var mongoose = require('mongoose');

// On se connecte à la base de données
mongoose.connect('mongodb://localhost/blog', function(err) {
  if (err) { throw err; }
});

// Création du schéma pour les commentaires
var commentaireArticleSchema = new mongoose.Schema({
  pseudo : { type : String, match: /^[a-zA-Z0-9-_]+$/ },
  contenu : String,
  date : { type : Date, default : Date.now }
});
```

```
// Création du Model pour les commentaires
var CommentaireArticleModel = mongoose.model('commentaires', commentaireArticleSchema);

// On crée une instance du Model
var monCommentaire = new CommentaireArticleModel({ pseudo : 'Alio' });
monCommentaire.contenu = 'Salut, super article sur Mongoose !';

// On le sauvegarde dans MongoDB !
monCommentaire.save(function (err) {
  if (err) { throw err; }
  console.log('Commentaire ajouté avec succès !');
  // On se déconnecte de MongoDB maintenant
  mongoose.connection.close();
});
```