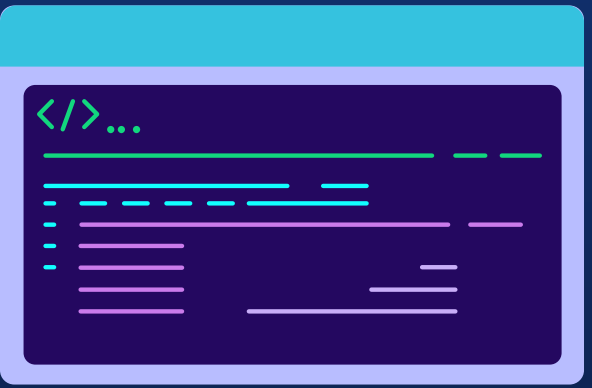
MongoDB



Utiliser MongoDB

Mongo DB

MongoDB

MongoDB est un moteur de base de données de orienté document, aussi dit “noSQL”.

- Elle permet de faire les requêtes de base du CRUD, et de la recherche, notamment par regex.
- Elle est compatible avec du scaling horizontal utilisant un système de sharding.
- Elle dispose d'un système très évolué de réplication
- Elle peut s’utiliser comme un système de fichier appelé GridFS
- Elle permet de faire de l'agrégation de données
- Elle permet l'exécution de JS directement dans la base

BDD Document

Une base de données document est dite semi-structurée.

Elle stocke une série de documents tous dans la même entité, comparé à une base relationnelle qui sépare les données en table.

Dans une base de données document, chaque document peut avoir des propriétés complètement différentes.

Les documents sont tous associés à une clef permettant de les identifier.

Les documents possèdent des métadata permettant de faire des recherches par champ

Une BDD Document est comme une base key-value mais plus spécialisée.

NoSQL

Le NoSQL date de la fin des années 60, mais le terme date du début des années 2000

Le besoin pour ce genre de bases de données prend son origine dans les entreprises du web 2.0, qui avaient besoin d'une simplicité de design, et de simplifier le scaling horizontal sur des clusters de machines.

Les bases NoSQL sont très utilisées dans le big data et le web en temps réel (permettre aux users de voir les informations en temps rẽel

Selon le cas dans lequel elles sont utilisées, les bases NoSQL peuvent se montrer plus ou moins rapides, et plus ou moins maintenables.

Setup

Installation

npm i mongoose

```
db.createUser(
  {
    user: "anon",
    pwd: "marbleCake",
    roles: [
      {
        role: "readWrite",
        db: "my_db"
      }
    ]
  }
);
db.createCollection("users");
```

```
# docker-compose
version: "3.8"

services:
 mongo:
  image: mongo:4.2
  container_name: mongo
  restart: always
  ports:
   - 27017:27017
  volumes:
   - ./data:/data/db
   - ./init-mongo.js:/docker-entrypoint-initdb.d/mongo-init.js:ro
  environment:
   MONGO_INITDB_ROOT_USERNAME: root
   MONGO_INITDB_ROOT_PASSWORD: root
   MONGO_INITDB_DATABASE: my_db

  networks:
   - mongo

networks:
 mongo:
```

Connexion

```
# index.ts
mongoose.connect("mongodb://anon:marbleCake@localhost:27017/my_db")
```

Lien de connexion mongoose :

mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[defaultauthdb][?options]]

Le lien permet de se connecter a plusieurs instances mongoose dans le cas d’un cluster.

Création de modele

Création de modèle

```
const userSchema = new mongoose.Schema({
   id: Number,
   name: String,
   email: String,
   password: String,
   created_at: Date,
   updated_at: Date
});
export const User = mongoose.model("Person", userSchema);
```

Sauvegarde de doc

Sauvegarde de doc

```
const user = new User(req.body);
   await user.save();
```

Récupération de doc

Person.find(callback)

Récupérer toutes les personnes.

Le callback prend en paramètres “error” et “response”

Person.find({ attribut: valeur }, callback)

Récupérer toutes les personnes qui ont des attributs équivalent à ceux recherchés

Le callback prend en paramètres “error” et “response”

Person.find({ attribut: “valeur” }, “attribut”, callback)

Récupérer uniquement le nom des personnes qui ont des attributs équivalent à ceux recherchés

Le callback prend en paramètres “error” et “response”

Person.findOne({ attribut: “valeur” }, callback)

Récupérer une seule personne qui a des attributs équivalent à ceux recherchés

Le callback prend en paramètres “error” et “response”

Person.findById(id, callback)

Récupérer la personne avec l’ID demandé

Le callback prend en paramètres “error” et “response”

modification de doc

Person.update(conditions, nouvelles_vals, callback)

Modifier toutes les personnes correspondantes à ‘conditions’ en leur appliquant les nouvelles val

Le callback prend en paramètres “error” et “response”

Person.findOneAndUpdate(conditions, nouvelles_vals, callback)

Modifier une seule personne correspondante à ‘conditions’ en lui appliquant les nouvelles val

Le callback prend en paramètres “error” et “response”

Person.findByIdAndUpdate(id, nouvelles_vals, callback)

Modifier la personne à l’ID spécifié en lui appliquant les nouvelles val

Le callback prend en paramètres “error” et “response”

Person.remove(conditions, callback)

supprime toutes les personnes qui correspondent aux conditions

Le callback prend en paramètres “error” et “response”

Person.findOneAndRemove(conditions, callback)

supprime la première personne qui correspond aux conditions

Le callback prend en paramètres “error” et “response”

Person.findByIdAndRemove(id, callback)

supprimer la personne à l’ID spécifié

Le callback prend en paramètres “error” et “response”

Sécurité

Hashage de mot de passe

Hasher un mdp consiste à le transformer de façon irréversible.

On peut alors comparer le hash de deux mots de passes pour savoir s’ils étaient initialement identiques.

Salt Hashing

Le salt hashing consiste à hasher une chaine de caractère, en utilisant une chaîne de caractère aléatoire appelée “salt”.

Le salt est utilisé lors du calcul du hash de la chaîne de caractère puis stocké avec elle pour les comparaisons par la suite

Salt Hashing

Le salt hashing consiste à hasher une chaîne de caractère, en utilisant une chaîne de caractère aléatoire appelée “salt”.

Le salt est utilisé lors du calcul du hash de la chaîne de caractère puis stocké avec elle pour les comparaisons par la suite

Implementation

import crypto as ‘crypto’

Générer un sel

```
let generateSalt = rounds => {
    if (rounds >= 15) {
        throw new Error(`${rounds} is greater than 15,Must be less that 15`);
    }
    if (typeof rounds !== 'number') {
        throw new Error('rounds param must be a number');
    }
    if (rounds == null) {
        rounds = 12;
    }
    return crypto.randomBytes(Math.ceil(rounds / 2)).toString('hex').slice(0,
rounds);
};
```

Générer un hash

```
let hasher = (password, salt) => {
    let hash = crypto.createHmac('sha512', salt);
    hash.update(password);
    let value = hash.digest('hex');
    return {
        salt: salt,
        hashedpassword: value
    };
};
```

Générer un hash

```
let hash = (password, salt) => {
    if (password == null || salt == null) {
        throw new Error('Must Provide Password and salt values');
    }
    if (typeof password !== 'string' || typeof salt !== 'string') {
        throw new Error('password must be a string and salt must either be a salt
string or a number of rounds');
    }
    return hasher(password, salt);
};
logger(hash('Wisdom', generateSalt(12)))
```

Comparer

```
let compare = (password, hash) => {

    if (password == null || hash == null) {
        throw new Error('password and hash is required to compare');
    }
    if (typeof password !== 'string' || typeof hash !== 'object') {
        throw new Error('password must be a String and hash must be an Object');
    }
    let passwordData = hasher(password, hash.salt);
    if (passwordData.hashedpassword === hash.hashedpassword) {
        return true;
    }
    return false
};
```

Comparer

```
let compare = (password, hash) => {

    if (password == null || hash == null) {
        throw new Error('password and hash is required to compare');
    }
    if (typeof password !== 'string' || typeof hash !== 'object') {
        throw new Error('password must be a String and hash must be an Object');
    }
    let passwordData = hasher(password, hash.salt);
    if (passwordData.hashedpassword === hash.hashedpassword) {
        return true;
    }
    return false
};
```

Comparer

```
let compare = (password, hash) => {

    if (password == null || hash == null) {
        throw new Error('password and hash is required to compare');
    }
    if (typeof password !== 'string' || typeof hash !== 'object') {
        throw new Error('password must be a String and hash must be an Object');
    }
    let passwordData = hasher(password, hash.salt);
    if (passwordData.hashedpassword === hash.hashedpassword) {
        return true;
    }
    return false
};
```