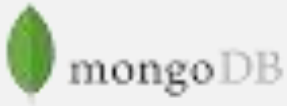
Mongoose



Mongoose

Qu’est-ce que Mongoose ?

Mongoose est une **bibliothèque JavaScript** (un **ODM**, *Object Data Modeling*) pour **Node.js**, utilisée pour interagir avec **MongoDB**.

En d’autres termes :
C’est une **couche intermédiaire** entre ton code Node.js et ta base MongoDB.
Elle te permet d’écrire du code structuré et cohérent, tout en profitant de la flexibilité du NoSQL.

Mongoose

Le rôle de Mongoose

MongoDB est un système **non structuré** : tu peux insérer à peu près n'importe quoi. Mais cela peut vite devenir le chaos.

C'est là que Mongoose intervient :

Sans Mongoose	**Avec Mongoose**
Les documents peuvent avoir des formes différentes	Tous les documents suivent un **schéma défini**
Tu dois gérer toi-même les validations	Mongoose **valide automatiquement** les données
Tu dois écrire les requêtes MongoDB manuellement	Tu as des **méthodes haut niveau** (find, save, update, etc.)
Tu gères les relations à la main	Tu peux utiliser **populate()** pour lier les données

Mongoose

Connexion à MongoDB

```
import { connect } from "mongoose";

connect("mongodb://127.0.0.1:27017/ma_base")
  .then(() => console.log("Connecté à MongoDB"))
  .catch(err => console.error("Erreur MongoDB :", err));


app.listen(3000, () => console.log("Serveur lancé sur http://localhost:3000"));
```

Mongoose

Définir un Schéma

Un **Schema** décrit la structure des documents dans une collection :

```
import { Schema, model } from "mongoose";

const userSchema = new Schema({
    nom: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    motdepasse: { type: String, required: true },
    role: { type: String, enum: ["user", "admin"], default: "user" },
});

export default model("User", userSchema);
```

Les principales méthodes *Mongoose* sur un modèle (ex : User)

1. Lecture — Rechercher / Lire des documents

Méthode	Description	Exemple
find()	Renvoie **tous** les documents correspondant au filtre. Retourne un **tableau**.	User.find({ role: "admin" })
findOne()	Renvoie le **premier document** correspondant au filtre. Retourne **un seul objet** ou null.	User.findOne({ email: "test@mail.com" })
findById()	Recherche par **identifiant MongoDB** (_id).	User.findById("66eab2f...")
countDocuments()	Compte le nombre de documents correspondant au filtre.	User.countDocuments({ role: "user" })
distinct()	Renvoie les valeurs distinctes d'un champ donné.	User.distinct("role")
exists()	Vérifie si au moins un document correspond. Retourne un booléen.	User.exists({ email: "..." })

Les principales méthodes *Mongoose* sur un modèle (ex : User)

2. Écriture — Création et insertion

Méthode	Description	Exemple
create()	Crée un ou plusieurs documents et les sauvegarde directement.	User.create({ nom: "Toto", email: "toto@mail.fr" })
insertMany()	Insère plusieurs documents à la fois (plus rapide que plusieurs .save()).	User.insertMany([{...}, {...}])
save()	Sauvegarde une instance d'un document créé avec new User().	const u = new User(data); u.save();

Les principales méthodes *Mongoose* sur un modèle (ex : User)

3. Mise à jour — Modifier les documents

Méthode	Description	Exemple
updateOne()	Met à jour **le premier document** correspondant au filtre.	User.updateOne({ nom: "Toto" }, { $set: { role: "admin" } })
updateMany()	Met à jour **tous les documents** correspondant au filtre.	User.updateMany({ actif: false }, { $set: { actif: true } })
findByIdAndUpdate()	Recherche par ID et met à jour le document. Retourne (par défaut) **l'ancienne version**, ou la nouvelle si { new: true }.	User.findByIdAndUpdate(id, { role: "admin" }, { new: true })
findOneAndUpdate()	Comme findByIdAndUpdate() mais avec un filtre personnalisé.	User.findOneAndUpdate({ email: "x@mail.com" }, { age: 30 })
replaceOne()	Remplace totalement un document (pas seulement certains champs).	User.replaceOne({ nom: "Toto" }, { nom: "Tata", email: "tata@mail.fr" })

Les principales méthodes *Mongoose* sur un modèle (ex : User)

4. Suppression — Effacer les documents

Méthode	Description	Exemple
deleteOne()	Supprime le premier document trouvé correspondant au filtre.	User.deleteOne({ nom: "Toto" })
deleteMany()	Supprime tous les documents correspondant au filtre.	User.deleteMany({ actif: false })
findByIdAndDelete()	Supprime un document selon son _id.	User.findByIdAndDelete(id)
findOneAndDelete()	Supprime le premier document correspondant au filtre.	User.findOneAndDelete({ email: "..." })

Les principales méthodes *Mongoose* sur un modèle (ex : User)

5. Méthodes avancées et outils

Méthode	Description	Exemple
aggregate()	Permet d'utiliser les **pipelines MongoDB** ($match, $group, $project, etc.)	User.aggregate([{ $match: { role: "admin" } }])
populate()	Récupère les documents liés (équivalent d'une jointure).	User.find().populate("amis")
lean()	Retourne des objets JS "plats" au lieu d'instances Mongoose (plus rapide).	User.find().lean()
select()	Sélectionne uniquement certains champs.	User.find().select("nom email")
sort()	Trie le résultat selon un ou plusieurs champs.	User.find().sort({ nom: 1 })
limit() / skip()	Pagination : limite le nombre de résultats ou saute des documents.	User.find().skip(10).limit(5)
exec()	Exécute la requête et renvoie une promesse (utile en async/await).	await User.find().exec()

Les principales méthodes *Mongoose* sur un modèle (ex : User)

6. Méthodes d'instance (sur un document précis)

Méthode	Description	Exemple
save()	Sauvegarde un document modifié.	user.nom = "Nouveau"; await user.save();
remove()	Supprime ce document de la base.	await user.remove();
validate()	Vérifie les contraintes du schéma sans sauvegarder.	await user.validate();
toJSON() / toObject()	Transforme le document Mongoose en objet simple.	user.toJSON()