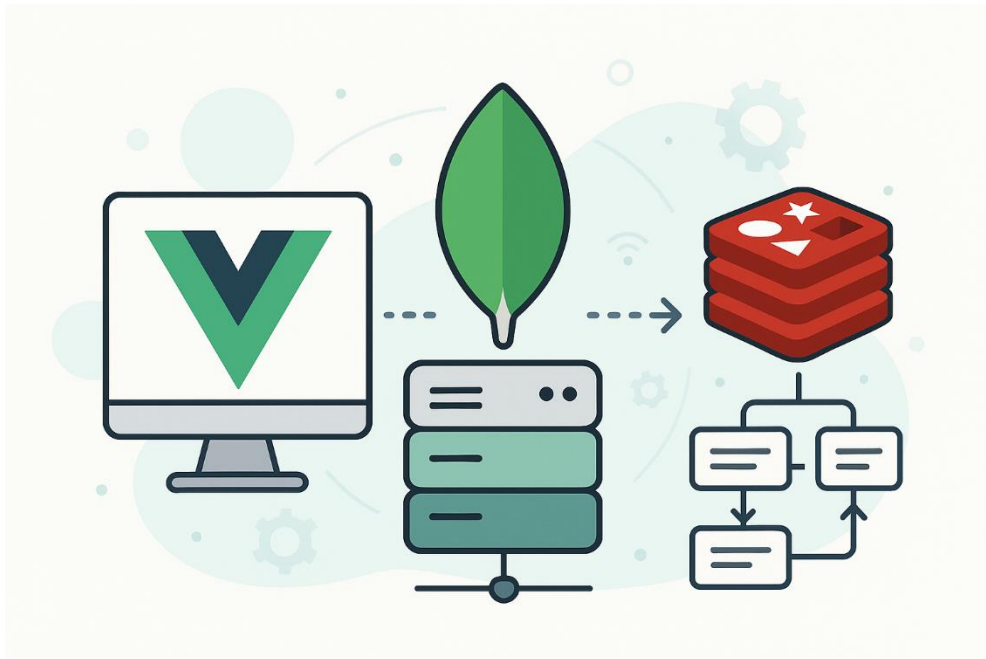


# TodoApp

---



OpenAI. (2025, 19.05). ChatGPT (Version 4o). [[Génération d'image pour le projet](#)]

Eithan Sanchez Filipe – MID2A  
Vennes  
24p  
M. Meylan

# Table des matières

---

<b>1</b>	<b>SPÉCIFICATIONS.....</b>	<b>3</b>
1.1	TITRE.....	3
1.2	DESCRIPTION.....	3
<b>2</b>	<b>EXPLICATIONS DU CODE.....</b>	<b>3</b>
2.1	IMPORTATION.....	3
2.2	ROLE/USER ADMINISTRATEUR .....	3
2.3	ROLE/USER UTILISATEUR .....	4
2.4	ROLE/USER MANAGER.....	4
2.5	BACKUP .....	5
<b>3</b>	<b>CONCLUSION.....</b>	<b>5</b>

# 1 SPÉCIFICATIONS

## 1.1 Titre

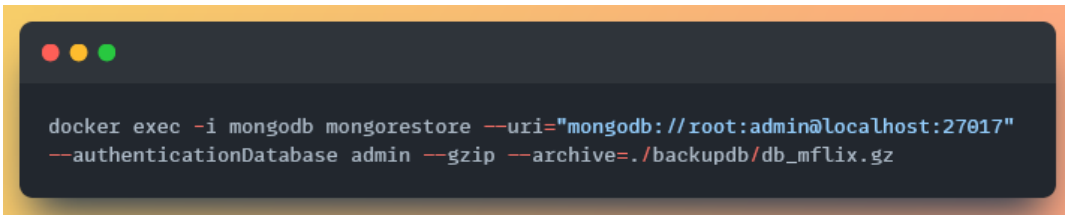
### TodoApp

## 1.2 Description

Reprendre une application « Todo List » puis transformer la base de données en MySQL en mongoDB. Exécuter quelques commandes et les documenter.

# 2 EXPLICATIONS DU CODE

## 2.1 Importation

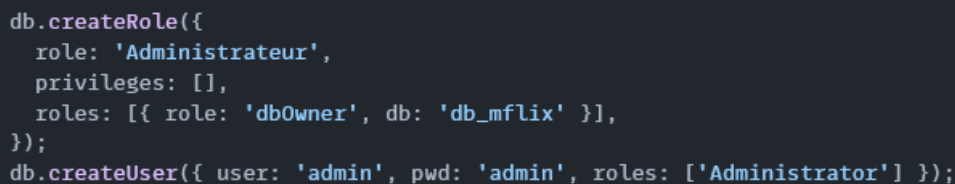


```
docker exec -i mongodb mongorestore --uri="mongodb://root:admin@localhost:27017"
--authenticationDatabase admin --gzip --archive=./backupdb/db_mflix.gz
```

Premièrement il faut importer la base de données pour ce faire on doit exécuter une commande dans le container docker. C'est à ça que « docker exec » sert à exécuter une commande dans le container « -i » sert à dire que le cli doit renvoyer les résultats ou les prompts. Pour importer les données on utilise l'outil mongorestore qui sert à restaurer à partir de fichiers binaires. Pour que mongorestore fonctionne il faut lui donner :

- --uri (string de connexion peut aussi être remplacé par --host, --port, --password et --username)
- --authenticationDatabase (base de données d'authentification)
- --gzip (indication que le fichier est compressé)
- --archive (fichier qu'on veut restaurer)

## 2.2 Role/User Administrateur



```
db.createRole({
  role: 'Administrateur',
  privileges: [],
  roles: [{ role: 'dbOwner', db: 'db_mflix' }],
});
db.createUser({ user: 'admin', pwd: 'admin', roles: ['Administrator'] });
```

Pour créer un rôle on fait db.createRole, on spécifie le nom dans ce cas Administrateur. Pour simplifier au lieu d'écrire chaque privilège on utilise le rôle dbOwner qui permet de lire/écrire/modifier, indexer, récupérer des statistiques, modifier les rôles et utilisateurs d'une base de données. Après avoir créé un rôle il faut créer l'utilisateur avec ce rôle. Pour ça on utilise db.createUser, il faut spécifier le nom d'utilisateur puis le mot de passe. On peut le faire manuellement ou avec passwordPrompt() et la personne concernée vient taper le mot de passe. Ensuite on spécifie le rôle qui est celui qu'on vient de créer.

## 2.3 Role/User Utilisateur

```
db.createRole({
  role: 'Utilisateur',
  privileges: [
    {
      resource: { db: 'db_mflix', collection: 'movies' },
      actions: ['find'],
    },
    {
      resource: { db: 'db_mflix', collection: 'comments' },
      actions: ['find', 'insert', 'update'],
    },
  ],
  roles: [],
});
db.createUser({ user: 'user', pwd: 'user', roles: ['User'] });
```

Pour le rôle utilisateur on lui donne des droits car il n'y a pas de rôle qui nous permet de faire ça plus efficacement. Ces droits lui permettent de lire le document movies et de lire/écrire/modifier sur le document comments.

## 2.4 Role/User Manager

```
db.createRole({
  role: 'Manager',
  privileges: [
    {
      resource: { db: 'db_mflix', collection: 'users' },
      actions: ['find'],
    },
    {
      resource: { db: 'db_mflix', collection: 'movies' },
      actions: ['find', 'update', 'remove'],
    },
    {
      resource: { db: 'db_mflix', collection: 'comments' },
      actions: ['find', 'update', 'remove'],
    },
  ],
  roles: [],
});
db.createUser({ user: 'manager', pwd: 'manager', roles: ['Manager'] });
```

Pour le rôle manager on lui donne le droit de lire le document users puis de lire/modifier/supprimer les documents movies et comments

## 2.5 Backup

```
docker exec -i mongodb mongodump --uri=mongodb://root:admin@localhost:27017
--authenticationDatabase admin --db=db_mflix --gzip --archive=./backupdb/backup_db_mflix.gz
```

Pour faire un backup d'une base de données il faut utiliser l'outil mongodump. Cet outil fait des exports binaire du contenu d'une base de données. Pour que la commande fonction on lui spécifie :

- --uri (string de connexion peut aussi être remplacer par --host, --port, --password et --username)
- --authenticationDatabase (base de données d'authentification)
- --db (on spécifie la base de données qu'on veut exporter)
- --gzip (indication que le fichier doit être compressé )
- --archive (le chemin dans lequel l'export sera)

## 3 CONCLUSION

Ce projet m'a permis de mettre en pratique les choses apprises dans le module 165, en faisant une migration d'une application qui utilisait MySQL vers MongoDB. C'est grâce à ce projet que j'ai pu approfondir mes connaissances, notamment sur Redis et MongoDB. J'ai bien aimé avoir un exemple réel d'utilisation de Redis, même si dans une application aussi petite, on n'en a pas vraiment besoin. Pour la deuxième partie du projet, cela ressemblait fortement aux tests déjà effectué dans le module. J'aurais aimé quelque chose que nous n'aurions pas encore vu ou expérimenté.