Projet Web – Secure Docs

Nest.js, GraphQL, Message Queuing, CI/CD et Tests

Objectif

Par groupe de 3 à 4 étudiants, vous devez développer une plateforme sécurisée de gestion documentaire, permettant à des utilisateurs authentifiés de créer, lire et organiser des documents numériques, tout en intégrant des outils de qualité logicielle et de déploiement continu.

En fin de TD, vous présenterez votre projet, vos choix d'architecture, ainsi qu'une démonstration technique.

Prérequis techniques

- Utilisation de Nest.js
- API construite en GraphQL (Code First)
- Intégration d'un système de Message Queuing (BullMQ + Redis)
- Tests automatisés (unitaires et intégration)
- Intégration continue (CI) avec GitHub Actions
- Déploiement avec Docker (option : Render / Heroku)

1- Étude de faisabilité

- Étudiez le fonctionnement de NestJS, son installation, son architecture modulaire.
- Analysez-les avantages/inconvénients de GraphQL dans le contexte d'une API documentaire.

Liens :

https://docs.nestjs.com/

https://graphql.org/

2- Mise en place du projet

- Initialisez le projet avec Nest CLI
- Créez un contrôleur Health check répondant "OK"
- Installez Redis via docker-compose.yml
- Intégrez BullMQ pour la gestion asynchrone :
- Créez une queue
- Ajoutez un job à partir du contrôleur health
- Créez un consumer
- Logguez le traitement du job

Liens :

https://docs.nestjs.com/techniques/queues

https://hub.docker.com/_/redis

3- Configuration GraphQL

- Installez @nestjs/graphql
- Configurez GraphQL en mode Code First
- Ajoutez un premier résolveur retournant { result: "ok" }
- Testez votre API avec Postman ou GraphQL Playground

https://docs.nestjs.com/graphql/quick-start

4- Conception de l'architecture

- Modélisez vos entités :
    - Utilisateur
    - Document
    - (optionnel) Historique ou Log

- Organisez votre projet :
    - Modèles / DTO / services / résolveurs
    - Gestion des rôles : admin, user

Exemples : https://github.com/nestjs/nest/tree/master/sample/23-graphql-code-first

5- Développement des APIs

- Résolveurs :
    - getDocumentsByUser()
    - getDocumentById()

- Mutations :
    - createDocument(title, description, fileUrl)
    - deleteDocument(id)
    - (bonus) updateDocument()

- Données stockées en mémoire dans un premier temps

6- Intégration du Message Queuing

- Lors de la création ou suppression d'un document :
- Envoyer un événement dans une queue
- Le consumer loggue et traite l'événement (audit, analytics, etc.)

7- Intégration continue

- Créez un dépôt GitHub
- Configurez une GitHub Action :
    - npm install
    - npm run lint
    - npm run test
    - nest build
- Créez une image Docker
- Testez-la localement
- Modifiez l'action GitHub pour builder l'image

https://github.com/features/actions

8- Tests automatisés

- Installez Jest
- Écrivez des tests unitaires pour :
- Les services
- Les résolveurs
- Bonus : tests d'intégration avec base en mémoire

https://docs.nestjs.com/fundamentals/testing

9- Déploiement continu

- Modifiez la GitHub Action pour :
    - Pusher votre image Docker sur DockerHub
    - Déployer automatiquement via Render ou Heroku à chaque push sur main

https://docs.render.com/web-services#deploy-from-a-container-registry

10- Tests d'intégration

- Créez une collection Postman pour tester vos APIs
- Automatisez ces tests avec Newman
- Intégrez-les dans la pipeline GitHub Actions

11- Interface utilisateur (bonus fortement recommandé)

- Affichez la liste des documents de l'utilisateur
- Affichez les détails d'un document
- Créez et supprimez un document via l'UI
- Framework libre : React, Vue, Angular...

12- Authentification

- Utilisez une lib comme Auth0 ou Passport.js avec JWT
- Protégez les routes sensibles
- Associez les documents à l'utilisateur authentifié

https://developer.auth0.com/

13- Gestion de fichiers (bonus)

- Implémentez l'upload de fichier
- Stockez localement (dans /uploads) ou utilisez un cloud storage
- Liez chaque document à une URL de fichier

14- Base de données

- Intégrez Prisma et une base PostgreSQL
- Modifiez les services pour stocker les documents et utilisateurs en base
- Déployez votre base sur Render PostgreSQL

https://www.prisma.io/docs