# **Test Technique - Développeur Fullstack (MEAN Stack)**

## **Contexte**

Vous êtes chargé de développer une plateforme de **blog collaboratif multi-auteurs** en utilisant **MongoDB, Express.js, Angular et Node.js** (**MEAN Stack**). Cette plateforme doit permettre aux utilisateurs de créer, modifier et commenter des articles avec une gestion avancée des permissions et une architecture **scalable et performante**

## **Objectifs du test**

* Démontrer votre maîtrise du **MEAN stack**.
* Implémenter une **architecture scalable** et **sécurisée**.
* Optimiser les **requêtes et performances**.
* Gérer des **permissions dynamiques** et un **système de rôles avancé**.
* Intégrer **WebSockets** pour la gestion des commentaires en temps réel.
* Fournir une **interface utilisateur fluide et réactive** en **Angular**.

## **Fonctionnalités à implémenter**

### **Gestion des utilisateurs & authentification (JWT + Refresh Token)**

* Un utilisateur peut s’inscrire et se connecter via une API sécurisée.
* Implémenter un système de rôles (Admin, Éditeur, Rédacteur, Lecteur).
* Utiliser **JWT + Refresh Token** pour l’authentification et les permissions.
* Permettre aux admins de gérer les rôles via une interface Angular.

### **Gestion des articles (CRUD avancé avec permissions dynamiques)**

* Un utilisateur peut **créer** un article avec un titre, un contenu, une image et des étiquettes.
* Un **Éditeur** ou un **Admin** peut modifier n’importe quel article.
* Un **Rédacteur** peut uniquement modifier ses propres articles.
* Seul un **Admin** peut supprimer des articles.
* Stocker les articles dans MongoDB et **optimiser les requêtes avec Mongoose**.

### **Commentaires en temps réel (Socket.io)**

* Un utilisateur connecté peut commenter un article.
* Les commentaires doivent être **imbriqués** (réponses aux commentaires).
* L’auteur d’un article doit être **notifié en temps réel** via **WebSockets**.

### **Sécurité avancée & bonnes pratiques**

* **Rate Limiting** sur l’API pour éviter les abus (DDoS).
* Sécuriser les données sensibles (hashing des mots de passe avec **bcrypt**).
* Implémenter **CORS**.

## **Durée estimée**

* **16 à 20 heures** réparties sur plusieurs jours.
* Le test peut être réalisé en plusieurs étapes.

## **Critères d’évaluation**

* **Qualité du code** (lisibilité, structure, modularité).
* **Bonne gestion des permissions et sécurité**.
* **Performances et optimisation des requêtes**.
* **Expérience utilisateur fluide et réactive**.
* **Capacité à utiliser des technologies avancées (ElasticSearch, Redis, WebSockets, Microservices)**.

### **Livrables attendus**

Un **repository Git** contenant :

* Le code source **backend** et **frontend**.
* Une **documentation** (README.md) expliquant :
  + Comment installer et exécuter le projet.
  + La structure du projet.
  + Les choix techniques réalisés.

### **Bonus (pour se démarquer)** - Implémentation d’un **service de notifications push** (Web Push API).

### **Dashboard Analytics (Angular + Chart.js/Recharts)**

* Afficher des **statistiques sur les articles** : vues, likes, partages.
* Un **Admin** peut voir les tendances des articles (graphique interactif).

### **Architecture Microservices (Bonus)**

* Découper l’application en **3 microservices** :
* **UserService** → Gère l’authentification et les utilisateurs.
* **ArticleService** → Gère les articles et les commentaires.
* **NotificationService** → Gère les WebSockets et les notifications.