**Installation, exécution et Structure**

1. **Installation et Exécution du Projet**

| **Partie** | **Prérequis** | **Commandes d'Installation et Lancement** |
| --- | --- | --- |
| **Global** | Node.js (v22.20.0), Angular CLI (20.3.6), MongoDB (instance locale ou Atlas) | MONGO\_URI=mongodb+srv://PROJET:bePeUpyYJV\_22S-@plateformeblog.ysgtpek.mongodb.net/  JWT\_ACCESS\_SECRET  JWT\_REFRESH\_SECRET  VAPID\_PUBLIC\_KEY  VAPID\_PRIVATE\_KEY VAPID\_EMAIL |
| **Backend** | Configurer un fichier .env avec MONGO\_URI, JWT\_SECRET, etc. | 1. cd backend/ 2. npm install 3. npm run dev(nodemon) |
| **Frontend** | Assurez-vous que l'URL de l'API est correcte dans l'environnement Angular. | 1. cd frontend/ 2. npm install 3. ng serve |
| **Accès** |  | Application disponible sur http://localhost:4200/ et API sur http://localhost:5000/ (selon la variable base\_url Postman). |

1. **Structure et Choix Techniques**

| **Section** | **Détails techniques** |
| --- | --- |
| Backend | Node.js (v22.20.0) + Express + MongoDB/Mongoose. |
| Frontend | Angular (Angular CLI: 20.3.6). |
| Temps Réel | Socket.io pour les commentaires et notifications. |
| écurité | JWT + Refresh Token Hashing (bcrypt) |
| Architecture | Utilisation de l'Architecture en Couches (Contrôleurs, Services, Modèles) côté Express et organisation par Modules/Composants côté Angular |

**Tests unitaires pour valider les rôles et permissions**

1. **Validation des Rôles et Permissions (Clé du Test)**

Identifiants de Test (À créer et valider par vous) : Vous devez créer un utilisateur pour chaque rôle dans votre base de données et fournir leurs accès : profil par défaut lecteur il inscrit et par la suite gère la base de donnes pour modifier rôle admin pour la gestion de rôle de la partie front.

D’autre part-il exicte 3 tableaux dans la base de donnes users comments et articles.

| Rôle | Email / Nom d'utilisateur (Exemple) | Mot de Passe (Exemple) | Permissions Clés |
| --- | --- | --- | --- |
| Admin | admin@blog.com | admin123 | Gère les rôles, Modifie/Supprime tout article. |
| Éditeur | editeur@blog.com | editeur123 | Modifie tout article. |
| Rédacteur | redacteur@blog.com | redacteur123 | Crée des articles, Modifie *ses* articles uniquement. |
| Lecteur | lecteur@blog.com | lecteur123 | Lit les articles, ajoute des commentaires. |

1. **Scénario de Test pour le Correcteur :**
   1. Connexion : Exécuter la requête Login dans Postman avec les identifiants Admin pour récupérer le {{token}} (voir Livrable 2).
   2. Test de Modification Restreinte : Se connecter avec redacteur@blog.com pour créer un article (Article A). Tenter ensuite de modifier cet article (Succès attendu). Changer le token pour celui de editeur@blog.com et tenter de modifier l'Article A (Succès attendu).
   3. Test d'Autorisation (Échec) : Revenir au token du Rédacteur et tenter de modifier un article créé par l'Éditeur ou un autre Rédacteur (Réponse HTTP 403 Forbidden attendue).
   4. Test de Suppression (Admin) : Utiliser le token Admin pour la requête Delete Article (Réponse HTTP 200/204 attendue).

**Annexe : Fichier Blog API - MEAN Stack.postman\_collection.json (Optimisé) avec le repos git**

1. Collection JSON Exportée

2. Script de Test (À insérer par vous dans la requête "Login" Postman)

Dans la requête Auth > Login, allez dans l'onglet Tests et ajoutez le script JavaScript suivant. Il récupère le JWT et le stocke dans la variable d'environnement token pour toutes les requêtes suivantes :

JavaScript

// Postman Tests pour récupérer le Token JWT et le stocker en variable d'environnement

var jsonData = JSON.parse(responseBody);

// S'assurer que la réponse contient la propriété 'token' (ou 'accessToken')

If (jsonData && jsonData.token) {

// Stocker le token dans la variable 'token' de l'environnement ou de la collection

pm.collectionVariables.set("token", jsonData.token); // Utiliser pm.environment.set si vous utilisez un environnement

console.log ("Token JWT récupéré et stocké dans la variable 'token'.");

} else {

console.log ("Erreur : Token non trouvé dans la réponse.");

}

// Optionnel: Récupérer l'ID utilisateur (si nécessaire pour des tests Admin/Rôles)

if (jsonData && jsonData.user && jsonData.user.\_id) {

pm.collectionVariables.set ("user\_id", jsonData.user.\_id);

}