Cahier des Charges Fonctionnel – API Node.js JWT Authentication

1. Contexte & Objectifs

Ce document représente le cahier des charges fonctionnel sous forme de Product Backlog. Il précède la phase de développement et décrit les fonctionnalités attendues pour l'API d'authentification JWT basée sur Node.js.

2. Backlog Fonctionnel (Scrum)

Le tableau ci-dessous présente les User Stories, leurs descriptions et critères d'acceptation.

ID	Priorité	Epic (Module)	Titre de la Story	Description (Format User Story)	Critères d'Acceptation (Definition of Done)
US1	High	Authentification	Inscription (Sign Up)	**En tant qu'** utilisateur anonyme, **je veux** créer un compte avec mon email, nom d'utilisateur et mot de passe, **afin d'** accéder aux services de l'API.	- L'utilisateur est ajouté en base de données. - Le mot de passe est hashé (bcrypt) avant stockage. - Erreur 400 si l'email existe déjà.
US2	High	Authentification	Connexion (Sign In)	**En tant qu'** utilisateur enregistré, **je veux**	- Retourne un code 200 et le Token JWT si succès.

				m'authentifier avec mes identifiants, **afin de** recevoir un Token JWT d'accès sécurisé.	- Retourne 404/401 si utilisateur introuvable ou mot de passe faux. - Le Token contient l'ID et les rôles.
US3	Medium	Authentification	Validation des données	**En tant que** système (API), **je veux** vérifier le format des emails et la complexité des mots de passe, **afin de** garantir l'intégrité des données utilisateurs.	- Email doit être au format valide (regex). - Mot de passe > 6 caractères. - Rejet immédiat (400) si invalide.
US4	High	Gestion des Rôles	Accès Ressources Protégées	**En tant qu'** utilisateur authentifié, **je veux** accéder au contenu sécurisé (/api/test/user), **afin de** consulter mes données	- Accès autorisé si Header x-access-token est valide. - Accès refusé (401/403) si Token manquant ou invalide.

				privées.	
US5	High	Gestion des Rôles	Distinction des Rôles (RBAC)	**En tant qu'** administrateur ou modérateur, **je veux** avoir des droits d'accès spécifiques (/api/test/mod, /api/test/admin) , **afin de** gérer les ressources réservées à mon niveau.	- L'endpoint /mod est accessible uniquement au rôle Moderator. - L'endpoint /admin est accessible uniquement au rôle Admin. - Un utilisateur simple reçoit une 403 Forbidden.
US6	High	Sécurité	Gestion des Tokens	**En tant que** système, **je veux** rejeter automatiquement les Tokens expirés ou falsifiés, **afin de** prévenir les accès non autorisés.	- Le middleware authJwt vérifie la signature du Token. - Erreur TokenExpiredError gérée proprement (401).
US7	Medium	Sécurité	Protection	**En tant que**	- Les

			Injections	système, **je veux** assainir les entrées utilisateurs, **afin de** protéger la base MongoDB contre les injections NoSQL.	caractères spéciaux dans les champs email/password sont neutralisés. - Pas de fuite de données en cas de requête malveillante.

3. Contraintes & Exigences Non Fonctionnelles

Ces exigences garantissent la sécurité, l'architecture et la maintenabilité du projet.

ID	Catégorie	Exigence	Description & Critère de Validation
ENF-01	**Sécurité**	**Stockage sécurisé des mots de passe**	Le mot de passe doit être stocké uniquement sous forme hachée (utilisation de bcrypt). Aucun mot de passe en clair en DB.
ENF-02	**Sécurité**	**Externalisation des secrets**	La clé secrète JWT (JWT_SECRET) ne doit pas être codée en dur mais chargée via des variables

			d'environnement (process.env).
ENF-03	**Sécurité**	**Contenu du Token (Payload)**	Aucun token JWT ne doit contenir de données sensibles (mot de passe, PII). Seuls l'ID public et les rôles sont autorisés.
ENF-04	**Sécurité**	**Validation stricte des entrées**	Mise en place de validateurs pour éviter les injections NoSQL. Tout champ non attendu doit être ignoré ou rejeté.
ENF-05	**Architecture**	**Architecture Stateless**	Le système ne doit conserver aucune session serveur (pas de cookies de session). L'authentification repose à 100% sur le token.
ENF-06	**Maintenabilité**	**Documentation**	Le code doit être documenté et l'API décrite (README clair ou Swagger) pour faciliter la maintenance.
ENF-07	**Performance**	**Temps de réponse**	L'API doit répondre en moins de 200ms pour 95% des requêtes d'authentification sous charge

			normale.

4. Traçabilité & Liens avec la future campagne de tests

Chaque User Story du backlog correspondra à un futur Test Case dans Xray.
Les tests unitaires et tests API seront automatisés et intégrés dans une pipeline CI/CD.
Les modifications fonctionnelles futures doivent être accompagnées de leurs tests.

5. Priorisation (MVP / Roadmap)

1. Inscription (register)
2. Connexion (login) + génération de JWT
3. Accès aux routes protégées
4. Consultation du profil utilisateur
5. Mise à jour / suppression du compte
6. Validation des entrées et sécurité
7. Documentation API
8. (Optionnel) Mécanisme Refresh Token

6. Annexes / Spécifications Techniques Suggérées

- Utiliser Express ou un framework équivalent pour structurer l'API.
- Utiliser Json Web Token (JWT) pour la gestion des tokens.
- Utiliser une base de données adaptée (MongoDB, SQL…).
- Gérer les erreurs via des statuts RESTful.
- Ajouter des logs serveur (erreurs d'authentification, accès protégés…).
- Stocker la configuration dans des variables d'environnement.