Cahier des Charges

Portail d'interopérabilité des API.

Introduction :

Ce cahier des charges définit les spécifications fonctionnelles et non fonctionnelles pour la réalisation d'un **portail d'interopérabilité des API**. Cette plateforme web permettra aux acteurs d'exposer, de consommer et de gérer les API de manière sécurisée et centralisée.

Objectifs du Projet :

Développer une application Web permettant l'exposition, la consommation et la gestion des API de manière sécurisée et centralisée.

- Mettre en place un portail de gestion des API.
- Permettre aux providers d'enregistrer et de publier leurs API.
- Offrir aux consumers un accès aux API via des demandes d'abonnement.
- Assurer le suivi et la validation des actions via un admin.
- Gérer les demandes d'adhésion pour devenir provider ou consumer.
- Garantir un accès aux fiches techniques et documents des API sans authentification.

Acteurs et Rôles :

Administrateur :

- Gère les utilisateurs (providers et consumers).
- Valide les demandes d'adhésion pour devenir provider ou consumer.
- Valide les API soumises par les providers.
- Suit et approuve les demandes d'accès des consumers.
- Associe une nomenclature aux API validées.
- Ajoute des documents téléchargeables liés aux API.

Provider (Fournisseur d'API)

- Fait une demande d'adhésion pour devenir provider (validation par l'admin).
- Ajoute une nouvelle API au catalogue (avec documentation).
- Met à jour et supprime ses API.
- Consulte les API disponibles.
- Valide ou rejette les demandes d'accès des consumers.

Consumer (Consommateur d'API)

- Fait une demande d'adhésion pour devenir consumer (validation par l'admin).
- Recherche et filtre les API disponibles.
- Consulte la description et la fiche technique d'une API sans authentification.
- Télécharge les documents liés aux API sans authentification.
- Fait une demande d'accès à une API.

Besoins Fonctionnels

Gestion des Demandes d'Adhésion

- Un utilisateur peut faire une demande d'adhésion pour devenir provider ou consumer en remplissant un formulaire (Nom, Prénom, Email, Rôle souhaité, Organisation, Justification).
- L'admin valide ou rejette les demandes d'adhésion.
- Notification à l'utilisateur en cas d'acceptation ou de refus.

Gestion des API

- Un provider peut ajouter une API avec :
- Nom, description, URL d'accès, type d'authentification, version, etc.
- Documentation technique en PDF ou autre format.
- Un admin valide ou rejette l'ajout d'une API.
- Un consumer peut consulter et filtrer les API par :
- Catégorie, fournisseur, statut (public, privé), date d'ajout, etc.
- Possibilité de consulter la fiche technique d'une API sans authentification.
- Téléchargement des documents d'une API sans authentification.

Gestion des Demandes d'Accès aux API

- Un consumer peut demander l'accès à une API.
- Le provider reçoit et valide ou rejette la demande.
- L'admin suit les demandes et peut intervenir en cas de besoin.

Gestion des Utilisateurs

- Un admin ajoute et gère les utilisateurs (providers et consumers).
- Un provider peut gérer ses informations.
- Authentification et autorisation des utilisateurs selon leur rôle.

Besoins Non Fonctionnels

Sécurité

- Authentification et autorisation via JWT.
- Gestion des permissions selon les rôles.
- Protection des endpoints sensibles.

Performance et Scalabilité

- API REST optimisées et documentation via Swagger.
- Caching pour améliorer la rapidité des requêtes.

Accessibilité et Expérience Utilisateur

- Interface utilisateur ergonomique et responsive.
- Recherche avancée et filtres dynamiques pour le catalogue des API.

Maintenabilité

- Code structuré et documenté.
- Tests unitaires et d'intégration couvrant au moins 80 % du code.

Disponibilité

- Disponibilité garantie de 99,9 % pendant les heures ouvrables.

Documentation

- Guides utilisateur pour chaque acteur.
- Documentation technique pour la maintenance.

Contraintes Techniques

- Frontend : Angular

- Backend : Spring Boot

- Base de Données : MongoDB

- Notifications : Email (SMTP) et SMS

- Sécurité : Authentification avec gestion des rôles

Méthodologie de Travail – Scrum

La méthode Scrum sera utilisée avec des sprints de 2 semaines sur une durée totale de 4 mois.

Chaque sprint inclura :

- Planification du sprint,

- Développement et tests incrémentaux,

- Revue pour évaluer les livrables,

- Rétrospective pour améliorer le sprint suivant.

Outils Scrum :

- Backlog produit pour organiser les fonctionnalités.

- Kanban pour suivre les tâches (en cours, terminées, bloquées).

- Daily Scrum : réunion quotidienne (15 min) pour suivre l'avancement.

Planification Prévisionnelle en Sprints

Sprint 1 (Semaines 1-2) : Préparation de l'environnement : installation des outils (IDE, Git, MongoDB, Spring Boot, Angular).

Sprint 2 (Semaines 3-4) : Conception de l'architecture technique et définition des modèles de données.

Sprint 3 (Semaines 5-6) : Développement des fonctionnalités pour Administrateur (CRUD des comptes et nomenclatures).

Sprint 4 (Semaines 7-8) : Développement des fonctionnalités pour Consumer et Provider (CRUD).

Sprint 5 (Semaines 9-10) : Développement des notifications automatiques (paramétrage, envoi par email et SMS).

Sprint 6 (Semaines 11-12) : Développement du tableau de bord des KPI.

Sprint 7 (Semaines 13-14) : Tests fonctionnels, validation utilisateur et corrections des bugs.

Sprint 8 (Semaines 15-16) : Finalisation du projet, déploiement et rédaction du rapport final.

Livrables Attendus

1. Code source complet.

2. Rapport de stage détaillé.

Évaluation et Suivi

- Suivi hebdomadaire avec le tuteur.

- Démonstrations à chaque phase clé (fin de sprint).