Cahier des Charges - Application de Gestion de Projets Agile

1. Introduction

L'objectif de cette application est de faciliter la gestion de projets selon la méthode Agile, en permettant de suivre le développement de nouvelles fonctionnalités à travers des **User Stories**, organisées en **Epics** et stockées dans un **Product Backlog**. Les **User Stories** seront ensuite sélectionnées et traitées dans le cadre de sprints à travers un **Sprint Backlog**. L'application devra permettre au Product Owner, aux développeurs et aux Scrum Masters de suivre l'évolution des tâches tout au long du projet.

2. Objectifs

- Suivre le **Product Backlog** et les **Sprint Backlogs** d'un projet Agile.
- Organiser les **User Stories** en **Epics**.
- Prioriser les **User Stories** en fonction de leur importance.
- Gérer les **Tasks** associées à chaque **User Story**.
- Affecter les tâches aux développeurs en fonction de leur disponibilité et de leurs compétences.
- Assurer le suivi de l'état d'avancement des User Stories et des Tasks (par exemple : **To Do**, **In Progress**, **Done**).
- Permettre la gestion des sprints et leur suivi.

3. Public Cible

- **Product Owner** : Gestion du Product Backlog, priorisation des User Stories, définition des Epics.
- **Scrum Master** : Suivi du déroulement des sprints, gestion de l'avancement des tâches.
- **Développeurs** : Mise à jour des User Stories et des tâches en fonction de l'avancement.

4. Fonctionnalités

4.1. Gestion du Product Backlog

- Création et gestion des **Product Backlogs**.
- Ajout, modification, et suppression des **User Stories** dans un **Product Backlog**.
- Priorisation des **User Stories** au sein du **Product Backlog**.
- Lien des **User Stories** à des **Epics**.
-

4.2. Critères pour Prioriser un Backlog

Facteurs de Priorisation :

1. **Valeur Métier** : Quel est l’impact sur les utilisateurs ou l’entreprise ?

2. **Urgence** : Est-ce nécessaire pour une prochaine livraison ou dépendance ?
3. **Complexité/Coût** : Difficulté d'implémentation (Story Points).
4. **Risques** : Faut-il réduire une incertitude (ex. : aspect technique non maîtrisé) ?
5. **Dépendances** : Cette tâche débloque-t-elle d'autres fonctionnalités ?

Techniques de Priorisation :

- **MoSCoW** : Must Have, Should Have, Could Have, Won't Have.
- **Valeur vs Effort** : Comparer l'impact et l'effort.
- **WSJF (Weighted Shortest Job First)** : Utilisé en SAFe, basé sur la valeur business et le temps d'attente.

4.3. Gestion des Epics

- Création et gestion des **Epics**.
- Lien des **User Stories** à des **Epics** (optionnel mais recommandé).
- Visualisation des **User Stories** liées à chaque **Epic**.

4.4. Gestion du Sprint Backlog

- Création et gestion des **Sprints** (Sprint 1, Sprint 2, etc.).
- Sélection des **User Stories** depuis le **Product Backlog** pour les inclure dans le **Sprint Backlog**.
- Gestion des **Tasks** associées à chaque **User Story** dans le **Sprint Backlog**.
- Suivi de l'état des **User Stories** et **Tasks** dans le **Sprint Backlog** (statuts : **To Do**, **In Progress**, **Done**).

4.5. Gestion des User Stories

- Ajout, modification, et suppression des **User Stories**.
- Lien des **User Stories** avec des **Epics**.
- Définition des **critères d'acceptation** pour chaque **User Story**.
- Suivi du **status** de chaque **User Story** (ex: **To Do**, **In Progress**, **Done**).
- Priorisation des **User Stories** dans le **Product Backlog**.

4.6. Gestion des Tasks

- Création et gestion des **Tasks** pour chaque **User Story**.
- Suivi du **status** de chaque **Task** (ex: **To Do**, **In Progress**, **Done**).

5. Architecture Technique

5.1. Backend

- **Framework** : Spring Boot
- **Langage** : Java
- **Base de données** : MySQL/PostgreSQL (ou toute autre base relationnelle)
- **Gestion des transactions** : Spring Transaction Management

- **Sécurité** : Spring Security (authentification via JWT, rôle utilisateur)
- **API** : RESTful API (utilisation de Spring MVC)
- **Tests** : JUnit, Mockito pour les tests unitaires et d'intégration.

6. Modèle de Données

6.1. Entities principales :

- **UserStory** : titre, description, priorité, statut, épic lié, ProductBacklog, SprintBacklog
- **Epic** : titre, description, liste des User Stories
- **ProductBacklog** : nom, liste des epics, User Stories
- **SprintBacklog** : nom, liste des User Stories, liste des Tasks
- **Task** : titre, description, statut, UserStory

6.2. Relations entre entités :

- Une **UserStory** peut être associée à un ou plusieurs **Epics**.
- Une **UserStory** est associée à un **ProductBacklog** et peut être déplacée dans un **SprintBacklog**.
- Un **SprintBacklog** contient plusieurs **UserStories** et **Tasks**.

6.3. Product Backlog

- Liste des **User Stories** avec possibilité de filtrer, trier et prioriser.
- Ajout et modification des **User Stories** et **Epics**.

6.4. Sprint Backlog

- Liste des **User Stories** sélectionnées pour le Sprint en cours.
- Visualisation de l'état des **Tasks** liées à chaque **User Story**.

6.5. User Stories & Tasks

- Interface pour ajouter, modifier, supprimer des **User Stories**.
- Vue détaillée des **Tasks** associées aux **User Stories**.

7. Gestion des Utilisateurs

- **Authentification** : Inscription et connexion des utilisateurs (Product Owner, Scrum Master, Développeur).
- **Gestion des rôles** : Product Owner, Scrum Master, Développeur.
- **Permissions** : Le Product Owner peut créer/modifier les User Stories et Epics, le Scrum Master peut gérer les sprints, et les développeurs peuvent gérer les tâches.

8. Suivi et Reporting

- **Suivi des progrès** : Indicateurs de performance comme le burndown chart, la progression des sprints.
- **Historique des sprints** : Historique des User Stories complétées, en cours, ou à venir.
- **Reporting personnalisé** : Tableau de bord avec des indicateurs pour chaque utilisateur.

9. Exigences Non Fonctionnelles

- **Sécurité** : Protection des données utilisateur avec des pratiques sécurisées de gestion des sessions (JWT, HTTPS).
- **Performance** : Temps de réponse rapide pour les interactions avec l'API, même avec un grand nombre de **User Stories** et **Tasks**.
- **Scalabilité** : Possibilité de faire évoluer l'application pour gérer un grand nombre de projets et utilisateurs.
- **Accessibilité** : Interface claire et simple à utiliser.

10. Livrables

- Code source de l'application.
- Documentation technique (architecture, API, base de données).
- Documentation utilisateur pour la prise en main de l'application.

Ce cahier des charges présente une vue d'ensemble des fonctionnalités et de la structure de l'application. Bien sûr, certains détails devront être affinés au fur et à mesure du développement.