Chapitre 2

Spécification des besoins

2.1 Introduction

Dans ce chapitre, nous allons définir avec précision les exigences de notre projet. Nous commencerons par analyser les besoins fonctionnels et non fonctionnels, qui sont cruciaux pour le développement de notre solution. Ensuite, nous identifierons les différents acteurs impliqués dans le système ainsi que leurs interactions avec celui-ci. Enfin, nous présenterons le backlog produit et le diagramme global des cas d'utilisation, permettant ainsi de structurer le processus de développement et d'assurer que notre application répond de manière efficace aux attentes des utilisateurs.

2.2 Identification des besoins fonctionnels

Un besoin fonctionnel est défini comme une exigence décrivant une action que le système doit pouvoir accomplir, sans prendre en compte de contraintes physiques. Ces besoins sont formulés du point de vue de l'utilisateur.[4] Notre système offre la capacité de :

- S'authentifier
- Échanger des messages
- Accéder aux statistiques
- Administrer ses données personnelles
- Gérer des tâches
- Affecter des développeurs à une tâche
- Gérer les projets
- Gérer les utilisateurs
- Gérer la phase de test
- Gérer les notes internes
- Consulter les tâches à accomplir
- Déposer son projet
- Tester des projets

2.3 Identification des besoins non fonctionnels

Les besoins non fonctionnels sont des critères de qualité essentiels pour l'exécution des besoins fonctionnels, ayant un impact indirect sur les résultats et la performance de l'utilisateur. Leur importance réside dans le fait qu'ils ne doivent en aucun cas être ignorés.[4] Pour garantir cela, il est nécessaire de satisfaire aux exigences suivantes :

- **Sécurité** : Notre plateforme doit garantir la confidentialité des données des utilisateurs ainsi qu'une protection solide de leurs informations personnelles, y compris les noms, adresses, numéros de téléphone et toutes autres données à caractère personnel. Cela nécessite la mise en œuvre de mesures de sécurité avancées, telles que des protocoles de cryptage, des pare-feu et des contrôles d'accès, afin de prévenir tout accès non autorisé.
- **Disponibilité** : Les services doivent être accessibles 24 heures sur 24 et 7 jours sur 7, permettant ainsi aux utilisateurs de réserver des hébergements à tout moment. Les temps d'arrêt doivent être minimisés afin d'éviter toute interruption pour les utilisateurs.
- **Compatibilité** : L'application doit être compatible avec divers navigateurs et appareils, permettant aux utilisateurs d'accéder au site depuis n'importe quelle plateforme. Des tests doivent être effectués sur plusieurs navigateurs et dispositifs pour garantir un fonctionnement optimal.
- **Performance et Utilisabilité** : Le site doit être rapide, réactif et intuitif, évitant ainsi les temps d'attente superflus et assurant une navigation fluide. Les pages doivent se charger rapidement, les menus et boutons clairement étiquetés, et les instructions faciles à suivre pour garantir une expérience utilisateur idéale.
- **Fiabilité** : Notre plateforme s'engage à garantir la fiabilité absolue de chaque service. Chaque processus est soigneusement conçu pour assurer l'intégrité des données et la continuité du service, offrant ainsi à nos utilisateurs une expérience sans faille et une confiance totale dans notre système.
- **Les erreurs** : L'application doit être conçue pour détecter et gérer les erreurs de manière efficace, en fournissant des messages d'erreur clairs et bien structurés. Ces messages orientent l'utilisateur, l'aidant à comprendre rapidement la nature du problème et les étapes à suivre pour le résoudre.

2.4 Identification des acteurs

Un acteur, qu'il s'agisse d'une personne physique ou morale, est une entité impliquée ou influencée par l'action ou le projet en question. Il est donc crucial de définir clairement l'action ou la série d'actions concernées afin d'identifier les acteurs et leur rôle respectif. Dans le cadre de notre plateforme, nous avons identifié deux acteurs qui interagissent directement avec notre système :

- **Superviseur** : C'est l'administrateur qui possède toutes les permissions de contrôle pour gérer les utilisateurs et les ressources, selon les critères établis.
- **Développeur** : Tout utilisateur de la plateforme a pour responsabilité d'exécuter les tâches qui lui sont assignées par le superviseur.

— **Testeur** : Toute personne ayant le rôle de testeur est chargée de valider la qualité et la conformité des projets développés. Il n'intervient que lorsqu'un projet lui est attribué par le superviseur.

Acteur	Rôle
Superviseur	— S'authentifier — Échanger des messages — Accéder aux statistiques — Administrer ses données personnelles — Gérer des tâches — Affecter des développeurs à une tâche — Gérer les projets — Gérer les utilisateurs — Gérer la phase de test — Gérer les notes internes
Développeur	— S'authentifier — Échanger des messages — Accéder aux statistiques — Administrer ses données personnelles — Consulter les tâches à accomplir — Déposer son projet
Testeur	— S'authentifier — Échanger des messages — Accéder aux statistiques — Administrer ses données personnelles — Tester des projets

TABLE 2.1 – Description détaillée des acteurs

2.5 Diagramme de cas d'utilisation globale

Le diagramme de cas d'utilisation vise à identifier les attentes de chaque utilisateur vis-à-vis du système. Notre analyse des besoins repose sur la visualisation des interactions entre les utilisateurs et les fonctionnalités envisagées du système.[5]

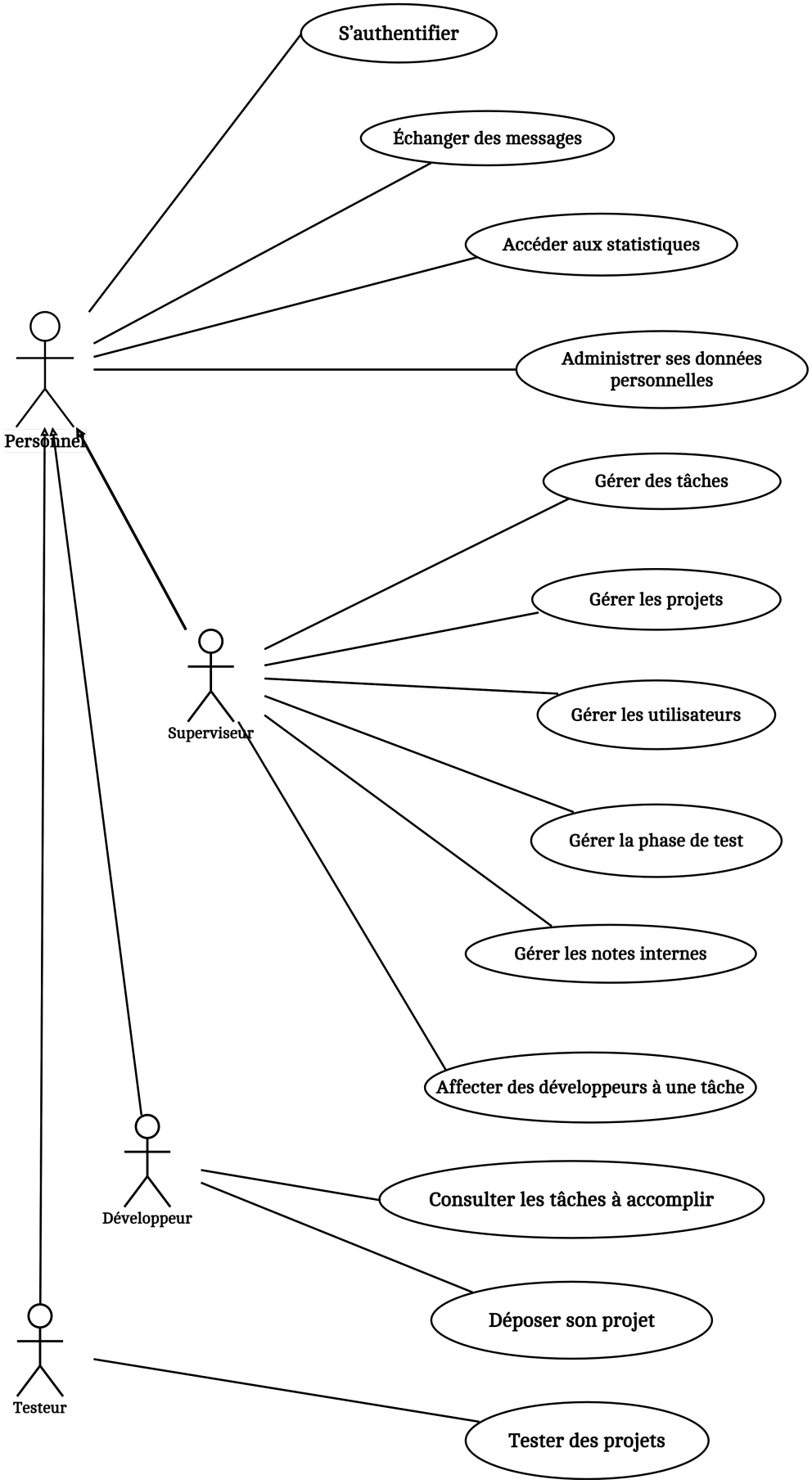

FIGURE 2.6 – Diagramme de cas d'utilisation

2.6 Backlog de produit

Après avoir déterminé les exigences fonctionnelles de notre système, nous exposons dans cette section le backlog du produit.

Backlog de produit	**Priorité**	**Estimation**	**Planification**
En tant que personnel, je peux m'authentifier.	1	Moyen	Sprint 0
En tant que personnel, je peux échanger des messages.	1	Moyen	Sprint 0
En tant que personnel, je peux accéder aux statistiques.	1	Moyen	Sprint 0
En tant que personnel, je peux administrer mes données personnelles.	1	Moyen	Sprint 0
En tant que superviseur, je peux gérer des tâches.	1	Moyen	Sprint 1
En tant que superviseur, je peux affecter des développeurs à une tâche.	1	Moyen	Sprint 1
En tant que superviseur, je peux gérer les projets.	2	Moyen	Sprint 1
En tant que superviseur, je peux gérer les utilisateurs.	2	Moyen	Sprint 1
En tant que superviseur, je peux gérer la phase de test.	2	Moyen	Sprint 1
En tant que superviseur, je peux gérer les notes internes.	2	Moyen	Sprint 1
En tant que développeur, je peux consulter les tâches à accomplir.	2	Moyen	Sprint 2
En tant que développeur, je peux déposer mon projet.	3	Moyen	Sprint 2
En tant que testeur, je peux tester des projets.	3	Moyen	Sprint 2

Table 2.2 – Backlog de produit

2.7 Identification des besoins techniques

Un besoin technique se réfère aux conditions, ressources et outils nécessaires pour assurer le bon fonctionnement et la mise en œuvre de notre projet. Dans ce chapitre, nous allons explorer les différents aspects de l'environnement de travail et de l'environnement logiciel, qui jouent un rôle important dans la réalisation de notre solution.[6]

2.7.1 Environnement de travail

Le développement de cette plateforme est réalisé à l'aide d'un ordinateur portable disposant des caractéristiques décrites dans le tableau 2.3.

Caractéristique	**Détails**
Système d'exploitation	Windows 11
Type du système	Système d'exploitation 64 bits, processeur x64
Processeur	Intel Core i7
Mémoire RAM	32 Go
Stockage	256 Go SSD et 1 To HDD
Carte graphique	NVIDIA GeForce GTX 1650

TABLE 2.3 – Caractéristiques du PC

2.7.2 Environnement logiciel

Dans notre projet, nous avons utilisé les logiciels suivants :

Logiciel	Définition
	Zoom est une plateforme de vidéoconférence multiplateforme offrant réunions, webinaires, partage d'écran, chat de groupe et enregistrement, facilitant le travail à distance et les échanges en ligne. [7]
	Postman est une plateforme pour la gestion et le test des API. Elle permet aux développeurs de créer, exécuter et documenter des requêtes HTTP de manière intuitive, sans compétences en programmation. Avec une interface graphique conviviale, Postman facilite l'organisation des requêtes, le partage entre équipes et l'automatisation des tests, tout en offrant des outils pour simuler des environnements et générer de la documentation API.[8]
	Node.js est un environnement d'exécution JavaScript côté serveur basé sur le moteur V8 de Google Chrome. Il permet aux développeurs de créer des applications web rapides et évolutives en utilisant JavaScript.[9]
	IntelliJ IDEA est un IDE professionnel pour Java et Kotlin, alliant complétion intelligente, refactorisation et analyse statique à une intégration poussée d'outils de versionnage, de build et de débogage, le tout dans une interface ergonomique, personnalisable et multiplateforme pour accélérer le développement et la gestion de projets. [10]
	MySQL est un système de gestion de base de données relationnelle open-source largement utilisé dans le développement d'applications web. Il offre une performance élevée, une fiabilité et une facilité d'utilisation pour stocker et gérer les données.[11]
	Visual Studio Code est un éditeur de code source léger, extensible et multiplateforme développé par Microsoft. Il offre une interface conviviale, une intégration avec de nombreux outils et une large gamme d'extensions pour améliorer la productivité des développeurs.[12]

TABLE 2.4 – Environnement Logiciel

2.7.3 Technologies utilisées

Dans notre projet, nous avons utilisé les technologies suivantes :

Technologie	Définition
	Angular est un framework basé sur TypeScript, développé par Google, qui permet de créer des applications web monopage performantes grâce à une architecture par composants. Il facilite le développement en offrant des fonctionnalités avancées comme la liaison de données, la gestion des composants et l'injection de dépendances.[13]
	Spring Boot est un framework Java qui simplifie la création d'applications Spring autonomes et prêtes pour la production, en réduisant la configuration et en offrant des fonctionnalités comme l'auto-configuration, un serveur web embarqué et une gestion facilitée des dépendances. Il permet aux développeurs de se concentrer sur la logique métier tout en accélérant le développement et le déploiement d'applications web ou de microservices. [14]
	HTML (HyperText Markup Language) est le langage de balisage standard utilisé pour créer des pages web. Il définit la structure et le contenu des pages en utilisant des balises. [15]
	CSS (Cascading Style Sheets) est un langage de feuille de style utilisé pour contrôler la présentation visuelle des pages web écrites en HTML. Il permet de définir la mise en forme, la couleur, la disposition et d'autres aspects visuels d'un site web. [16]
	TypeScript (TS) est un langage de programmation open-source, qui étend JavaScript en ajoutant un typage statique optionnel pour améliorer la sécurité et la maintenabilité du code, tout en restant compatible avec tout code JavaScript valide. [17]

TABLE 2.5 – Technologies utilisées

2.8 Méthodologie de conception

2.8.1 Méthodologie Agile

La méthodologie Agile est une approche itérative et collaborative pour la gestion de projets qui se concentre sur la livraison continue de produits fonctionnels, en mettant l'accent sur la flexibilité, l'adaptabilité et la communication constante entre les membres de l'équipe. Contrairement aux méthodologies traditionnelles, l'Agile favorise des cycles de développement courts, des retours fréquents des parties prenantes et une capacité à s'adapter rapidement aux changements. En intégrant des principes tels que la collaboration, la réactivité aux besoins changeants du projet et la livraison incrémentielle, l'approche Agile permet aux équipes de travailler de manière plus efficace et de produire des résultats de haute qualité tout en répondant aux attentes du client de manière continue.[18]

2.8.2 Scrum

Scrum est un cadre de travail Agile spécifique qui se concentre sur la gestion de projets complexes en mettant l'accent sur la collaboration, la transparence et l'adaptabilité. Basé sur des itérations courtes appelées «**sprints**», généralement de deux à quatre semaines, Scrum encourage les équipes à se concentrer sur des objectifs clairs et à livrer des fonctionnalités utilisables à la fin de chaque sprint. Les rôles clés dans Scrum incluent le Product Owner, chargé de définir les besoins du produit, le Scrum Master, responsable de faciliter le processus et d'éliminer les obstacles, et l'équipe de développement, chargée de réaliser le travail. Les cérémonies régulières telles que la planification du sprint, la revue de sprint et la rétrospective aident à maintenir la transparence et à favoriser l'amélioration continue au sein de l'équipe.

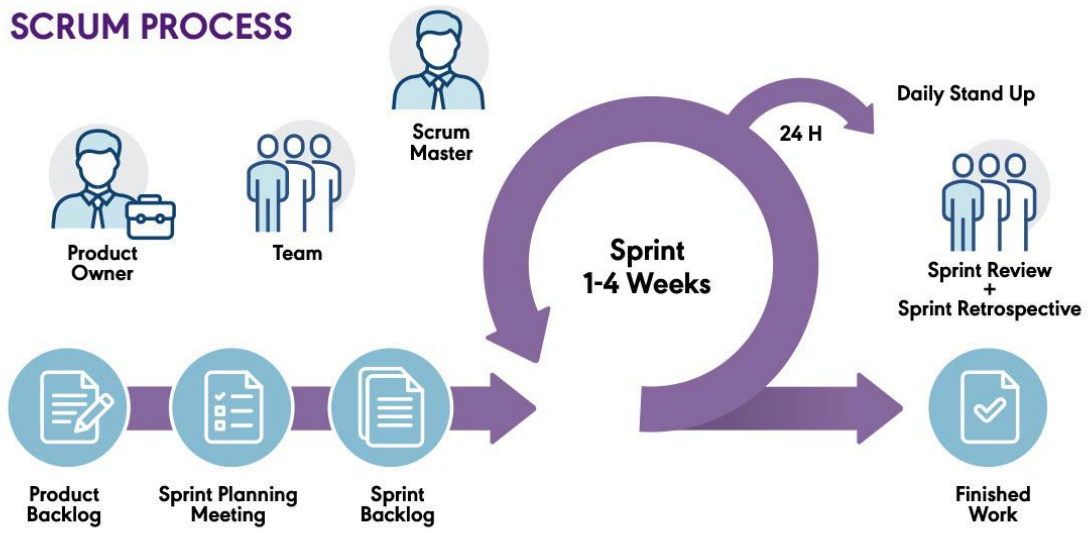

FIGURE 2.7 – Processus Scrum [19]

2.9 Conclusion

Dans ce chapitre, nous avons présenté le diagramme de cas d'utilisation, facilitant ainsi la décomposition fonctionnelle de notre système.

Dans le prochain chapitre, nous aborderons le premier release en détaillant le processus de raffinement, la conception et la réalisation.