CYCLE INGENIEUR – ING4

RAPPORT DE PROJET PDL

Intitulé du projet

Conception, développement de la plateforme Gestion des dépenses

---

Réalisé par

Ayoub Mekni

Encadrant

Houssem jemal

Année Universitaire
2024 - 2025

Chapitre 1 : Introduction

1.1 Présentation du projet

Le projet consiste en la création d'une application de gestion des dépenses permettant aux utilisateurs de soumettre, suivre et gérer leurs dépenses professionnelles ou personnelles. Cette application offre également aux administrateurs les moyens de valider ces dépenses et de gérer les utilisateurs ainsi que les catégories associées. L'objectif principal est de faciliter le contrôle budgétaire et la transparence dans la gestion des frais.

1.2 Contexte et justification

Dans un contexte où la gestion des dépenses est souvent complexe et chronophage, notamment dans les organisations et entreprises, il est essentiel de disposer d'un outil digitalisé performant. Ce projet vise à répondre à ce besoin en proposant une solution complète et accessible pour la saisie, la validation, et le suivi des dépenses avec justificatifs, tout en assurant la sécurité et la traçabilité des opérations.

La digitalisation des processus permet de réduire les erreurs, d'accélérer le traitement des demandes et de renforcer le contrôle financier, tout en offrant une meilleure visibilité aux utilisateurs et aux gestionnaires.

1.3 Étude de l'existant

Avant de développer cette application de gestion des dépenses, il est essentiel d'examiner les solutions existantes ainsi que les pratiques courantes dans ce domaine.

Solutions logicielles actuelles

Plusieurs logiciels de gestion des dépenses existent aujourd'hui sur le marché, allant des applications généralistes aux solutions dédiées aux entreprises. Parmi les plus répandues, on trouve :

- **Expensify, Concur, ou SAP Concur** : des outils professionnels largement utilisés, offrant des fonctionnalités complètes comme la soumission de notes de frais, l'intégration bancaire, la gestion des justificatifs, et des rapports avancés. Cependant, ces solutions sont souvent coûteuses et peuvent s'avérer complexes à personnaliser pour des besoins spécifiques.
- **Google Sheets ou Excel** : beaucoup d'entreprises utilisent encore des feuilles de calcul manuelles pour suivre leurs dépenses. Cette méthode est simple et accessible mais manque d'automatisation, de contrôle des accès, et peut être source d'erreurs.
- **Solutions open source** : des projets comme Odoo proposent des modules de gestion financière, mais nécessitent une installation et une configuration parfois complexes.

Limites des solutions existantes

- Coût élevé des solutions professionnelles pour les petites structures.
- Complexité d'utilisation et surdimensionnement des fonctionnalités dans certains cas.
- Manque de personnalisation selon les spécificités des utilisateurs.
- Absence parfois d'une intégration fluide entre la soumission, la validation et le suivi budgétaire.

Justification du projet

Face à ces constats, ce projet vise à proposer une solution plus légère, modulable, et adaptée aux besoins spécifiques des utilisateurs ciblés. L'objectif est de concevoir une application facile à utiliser, sécurisée, et

qui couvre l'essentiel des besoins sans complexité excessive, tout en assurant une bonne évolutivité pour de futures améliorations.

1.4 Objectifs du projet

Les objectifs clés de ce projet sont :

- Implémenter un système sécurisé d'authentification et de gestion des utilisateurs.
- Permettre aux utilisateurs de soumettre leurs dépenses avec justificatifs facilement.
- Offrir un tableau de bord intuitif pour visualiser, filtrer et analyser les dépenses.
- Mettre en place un workflow de validation des dépenses par un administrateur.
- Gérer les catégories de dépenses pour une meilleure classification.
- Contrôler le budget mensuel par utilisateur avec notifications en cas de dépassement.
- Fournir des fonctionnalités avancées telles que l'export des données et l'historique des actions.

1.3.4 Méthodologie utilisée

1.3.4.1 Méthode de gestion de projet – Le Framework adopté : Scrum

Le cadre **Scrum** a été adopté pour mener à bien ce projet. Il s'agit d'une méthode agile de gestion de projet qui repose sur :

- Des **sprints** courts et itératifs (1 à 2 semaines),
- Un **product backlog** (liste des fonctionnalités),
- Des rôles définis : **Product Owner**, **Scrum Master** et **Équipe de développement**,
- Des **réunions quotidiennes (daily meetings)** et des **revues de sprint**.

Ce choix s'explique par la flexibilité de Scrum et sa capacité à intégrer les retours clients rapidement tout au long du cycle de développement.

Voici ta section réorganisée, reformulée et **présentée de manière professionnelle** pour un rapport de stage ou de projet (en français), en tenant compte de la mise en page et des normes académiques :

1.4 Le Framework adopté : Scrum

La méthodologie **Scrum** est une méthode de gestion agile des projets, introduite en 2002. Son nom est emprunté au rugby et signifie « la mêlée », reflétant la collaboration étroite entre les membres de l'équipe.

Scrum repose sur le découpage du projet en itérations appelées **sprints**, d'une durée variant généralement entre **deux semaines et un mois**. À la fin de chaque sprint, l'équipe fournit un **produit partiel et fonctionnel**, considéré comme un livrable potentiel.

Avant chaque sprint, les tâches sont évaluées en **temps et en complexité** à l'aide de pratiques agiles telles que le **planning poker**. Ces estimations permettent de :

- Planifier précisément les livraisons,
- Estimer les charges de travail,
- Et de fournir des projections de coût au client.

Figure 1.2 – Cycle de vie de la méthode Scrum

1.4.1 Méthode de modélisation et de conception

J'ai adopté le langage de modélisation **UML (Unified Modeling Language)** comme formalisme de conception.

UML est un **langage visuel normalisé** qui permet de représenter les différents aspects d'un système logiciel à travers un ensemble de **diagrammes**. Ces derniers offrent plusieurs perspectives :

- structurelles (diagrammes de classes, objets…),
- comportementales (cas d'utilisation, séquences…),
- et d'interaction.

Ce formalisme facilite la compréhension globale du système, sa structure interne ainsi que les interactions entre ses différentes entités.

Figure 1.3 – Exemple de modélisation UML

1.5 Technologies et outils de travail

1.5.1 Environnement matériel

Au cours de mon stage, j'ai travaillé avec un **ordinateur portable Dell**, dont les caractéristiques sont les suivantes :

- **Modèle** : Dell portable
- **Processeur** : Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz – 2.30GHz
- **Mémoire RAM** : 12 Go
- **Architecture** : Système d'exploitation 64 bits, processeur x64

1.5.2 Environnement logiciel

Dans le cadre du développement de l'application, plusieurs outils logiciels ont été utilisés :

a) Éditeur de code : Visual Studio Code

![Logo Visual Studio Code]

Visual Studio Code est un éditeur de code source **léger, extensible et puissant**, développé par Microsoft. Il est compatible avec :

- Windows, macOS et Linux,
- de nombreux langages de programmation (JavaScript, Python, PHP, Java, etc.),
- et propose une **vaste bibliothèque d'extensions** permettant d'enrichir l'environnement de développement.

Figure 1.5 – Logo Visual Studio Code

XAMPP est une solution logicielle permettant de mettre en place rapidement un environnement serveur local, comprenant :

- **Apache** (serveur HTTP),
- **MySQL/MariaDB** (base de données),
- **PHP et Perl**.

Il est reconnu pour sa **simplicité d'installation** et ne nécessite pas de compétences techniques avancées. Il est compatible avec les systèmes Windows, Linux et macOS.

Figure 5 – Interface de XAMPP Control Panel v3.2.4

c) Navigateur Web : Google Chrome

![Logo Chrome]

Google Chrome est un navigateur rapide, sécurisé et multiplateforme développé par Google. Il est basé sur le projet open source **Chromium** et est disponible sur :

- Windows,
- macOS,
- Linux,
- Android,
- iOS.

Il a été utilisé tout au long du développement pour **tester et exécuter l'application web**.

Figure 1 – Navigateur Google Chrome

1.5.2 Environnement logiciel (suite)

d) Framework CSS : Bootstrap

Bootstrap est un **framework front-end** open-source développé par Twitter. Il permet de **créer des interfaces web responsives et modernes** rapidement grâce à une collection complète de composants et de styles prêts à l'emploi.

Bootstrap inclut :

- Des **fichiers CSS et JavaScript**,
- Des **composants HTML prédéfinis** (boutons, formulaires, cartes, modales, etc.),
- Et des **grilles flexibles** pour le responsive design.

Son objectif principal est de **simplifier et accélérer** le développement d'interfaces utilisateurs cohérentes et adaptatives, tout en respectant les bonnes pratiques du web.

Figure 8 – Framework Bootstrap

e) Outil de versionnage : Git Bash

Git Bash est un outil permettant d'utiliser **Git** via une **interface en ligne de commande** sur les systèmes Windows. Il émule un environnement UNIX (type Bash) pour permettre aux utilisateurs d'exécuter des commandes Git et des scripts Bash dans un terminal.

Git Bash est constitué de :

- Git (le système de gestion de versions distribué),
- Bash (un shell Unix populaire),
- et plusieurs utilitaires Bash courants.

C'est un outil essentiel pour gérer le **versionnage du code source**, collaborer avec d'autres développeurs via des plateformes comme GitHub ou GitLab, et automatiser certaines tâches de développement.

Figure 10 – Git Bash sous Windows

f) Framework Backend : Spring Boot

Spring Boot est un framework Java open source utilisé pour créer des applications backend robustes et évolutives. Il permet de développer rapidement des applications web en se basant sur le framework Spring tout en réduisant la configuration manuelle grâce à son approche convention over configuration.
Ses avantages :

- Intégration facile avec des bases de données (JPA, JDBC)
- Gestion des dépendances via Maven ou Gradle
- Intégration avec Spring Security pour la gestion d'authentification et d'autorisation
- Création d'API REST performantes

Figure 11 – Spring Boot

g) Framework Frontend : Angular

Angular est un framework TypeScript développé par Google pour la création d'applications web côté client (frontend). Il permet de concevoir des interfaces utilisateur dynamiques, performantes et maintenables.
Parmi ses fonctionnalités clés :

- Liaison bidirectionnelle des données (two-way data binding)
- Architecture basée sur des composants
- Système de routage intégré
- Communication simplifiée avec des API REST via le service HttpClient
- CLI puissant pour la génération et le déploiement de code

Figure 12 – Angular

1.6 Architecture de l'application

L'architecture de l'application repose sur une **architecture en couches** respectant le modèle **client-serveur**. Elle est composée principalement de deux parties : le **front-end** développé avec **Angular**, et le **back-end** développé avec **Spring Boot**.

1.6.1 Architecture générale

L'application suit une **architecture en trois tiers** :

- **Couche de présentation (Front-end)** :
  Développée avec **Angular**, cette couche représente l'interface utilisateur. Elle est chargée de l'interaction avec l'utilisateur final à travers un navigateur web. Elle consomme les services exposés par l'API REST du back-end.
- **Couche métier et logique applicative (Back-end)** :
  Implémentée avec **Spring Boot**, elle contient toute la logique métier de l'application. Elle gère les traitements, les règles de gestion, la sécurité, ainsi que l'exposition des API REST nécessaires à la communication avec le front-end.
- **Couche de persistance (Base de données)** :
  Cette couche est responsable de la gestion des données via **Spring Data JPA**. Elle assure l'accès, la lecture et l'écriture dans une base de données relationnelle (comme MySQL ou PostgreSQL).

1.6.2 Communication entre les composants

La communication entre Angular et Spring Boot se fait via des **API REST** utilisant le protocole **HTTP/HTTPS**. Angular envoie des requêtes (GET, POST, PUT, DELETE) au serveur, et reçoit les réponses au format **JSON**.

1.6.3 Avantages de l'architecture adoptée

- **Séparation des responsabilités** : chaque couche est indépendante et peut être développée, testée et maintenue séparément.
- **Scalabilité** : possibilité de faire évoluer le front-end ou le back-end indépendamment.
- **Réutilisabilité** : les services REST peuvent être réutilisés dans d'autres applications (mobile, desktop...).
- **Performance** : Angular permet une exécution rapide côté client, et Spring Boot offre des performances élevées côté serveur.

Chapitre 2 : Planification du Backlog produit

2.1 Introduction

Dans ce chapitre, je présente la phase de planification du projet. Cette phase est essentielle pour définir les besoins fonctionnels à travers les **user stories** et organiser les tâches à accomplir dans le **Backlog produit**. Avant d'établir le backlog, il est important d'identifier les **acteurs principaux** du système, c'est-à-dire les types d'utilisateurs qui interagiront avec l'application.

2.2 Identification des profils utilisateurs

Un **acteur** est une entité externe qui interagit avec le système pour atteindre un objectif précis. Ces interactions permettent de déduire les besoins fonctionnels du système, qui seront exprimés sous forme de **user stories**.

Dans le cadre de mon application de **gestion des dépenses**, trois profils utilisateurs principaux ont été identifiés :

Administrateur :

L'administrateur est chargé de la gestion globale de l'application. Il peut :

- Gérer les utilisateurs (ajout, modification, suppression),
- Consulter les rapports globaux de dépenses,
- Modifier les catégories de dépenses,
- Superviser les activités de tous les utilisateurs.

Utilisateur (standard) :

L'utilisateur est une personne qui utilise l'application pour gérer ses dépenses personnelles. Il peut :

- Ajouter une dépense ou une recette,
- Consulter son historique de dépenses,
- Visualiser des graphiques ou des rapports sur ses finances,
- Gérer ses catégories de dépenses.

2.2.1 Liste des User Stories

ID	User Story	Priorité	Acteur	Fonctionnalité liée
US1	En tant qu'utilisateur, je veux m'authentifier pour accéder à l'application.	Haute	Utilisateur/Admin	UC1: S'authentifier
US2	En tant qu'utilisateur, je veux soumettre une dépense pour remboursement.	Haute	Utilisateur	UC3: Soumettre une dépense
US3	En tant qu'utilisateur, je veux télécharger un justificatif avec ma dépense.	Haute	Utilisateur	UC8: Télécharger justificatifs
US4	En tant qu'utilisateur, je veux visualiser mon	Moyenne	Utilisateur	UC9: Visualiser

ID	User Story	Priorité	Acteur	Fonctionnalité liée
	tableau de bord.			tableau de bord
US5	En tant qu'administrateur, je veux valider ou rejeter une dépense.	Haute	Admin	UC4: Valider une dépense
US6	En tant qu'administrateur, je veux consulter des rapports de dépenses.	Moyenne	Admin	UC5: Consulter les rapports
US7	En tant qu'administrateur, je veux gérer les catégories de dépenses.	Moyenne	Admin	UC6: Gérer les catégories
US8	En tant qu'administrateur, je veux définir des budgets mensuels par utilisateur.	Moyenne	Admin	UC7: Définir les budgets
US9	En tant qu'administrateur, je veux gérer les comptes utilisateurs.	Moyenne	Admin	UC2: Gérer les utilisateurs
US10	En tant qu'utilisateur, je veux être notifié du statut de mes dépenses (soumise, approuvée, rejetée, remboursée).	Basse	Utilisateur	Dépend de l'état StatutDepense
US11	En tant qu'utilisateur, je veux consulter l'historique de mes dépenses.	Moyenne	Utilisateur	UC9 (tableau de bord)
US12	En tant qu'utilisateur, je veux voir si mes dépenses respectent le budget alloué.	Moyenne	Utilisateur	Lié à Budget + Dépenses

Réalise	Nom de la tâche	Tâches techniques	Priorité	Estimation (points)
Release 1	Authentification	- Créer modèle Utilisateur- Implémenter l'authentification (JWT/session)- Créer l'écran de connexion	Haute	5
Release 1	Soumission d'une dépense avec justificatif	- Créer entité Dépense- Formulaire de soumission- Upload de justificatif- Liaison à l'utilisateur connecté	Haute	8
Release 1	Visualisation des dépenses	- Tableau de bord- Filtrage par statut- Statistiques/Graphiques	Moyenne	5
Release 1	Téléchargement des justificatifs	- Stockage fichiers- Lien fichier dans la DB- Sécurité d'accès	Haute	5
Release 1	Validation des dépenses	- Vue Admin des dépenses- Actions Approuver/Rejeter- Changement du statut	Moyenne	8
Release 1	Gestion des catégories	- Créer entité Catégorie- Ajouter/Modifier/Supprimer une catégorie- Associer à une dépense	Moyenne	5
Release 1	Gestion du budget utilisateur	- Créer entité Budget- Lier à utilisateur- Définir un montant limite mensuel- Vérification avant soumission dépense	Moyenne	8
Release 2	Inscription des utilisateurs	- Formulaire d'inscription- Validation email (optionnelle)- Enregistrement en base de données	Haute	5
Release 2	Gestion des utilisateurs (admin)	- Lister les utilisateurs- Ajouter/Modifier/Supprimer un utilisateur-	Haute	8

		Attribuer des rôles		
Release 2	Filtrage et recherche des dépenses	- Ajout de filtres (date, montant, catégorie) - Recherche par mots-clés	Moyenne	5
Release 2	Export des dépenses en PDF/Excel	- Générer un export PDF ou Excel- Ajouter un bouton "Exporter" dans l'interface	Basse	5
Release 2	Notifications	- Alerte pour dépenses dépassant le budget- Notification de validation/rejet	Moyenne	5
Release 2	Historique des actions	- Suivi des dépenses modifiées ou supprimées- Audit utilisateur	Basse	5

Sprint Backlog – Sprint 1 (2 semaines)

User Story ID	User Story	Tâches techniques	Priorité	Estimation (points)
US1	En tant qu'utilisateur, je veux pouvoir m'authentifier afin d'accéder à l'application.	- Créer modèle Utilisateur (connexion)- Implémenter l'authentification (JWT/session) - Créer l'écran de connexion	Haute	5
US2	En tant qu'utilisateur, je veux soumettre une dépense avec justificatif pour remboursement.	- Créer entité Dépense- Créer formulaire de soumission- Ajouter champ de téléchargement de justificatif- Lier à l'utilisateur connecté	Haute	8
US4	En tant qu'utilisateur, je veux télécharger un justificatif pour chaque dépense.	- Ajouter stockage fichiers- Lier chemin fichier au champ JustificatifPath- Gérer sécurité accès fichiers	Haute	5

Sprint Backlog – Sprint 2 (2 semaines)

User Story ID	User Story	Tâches techniques	Priorité	Estimation (points)

User Story ID	User Story	Tâches techniques	Priorité	Estimation (points)
US3	En tant qu’utilisateur, je veux visualiser mes dépenses sur un tableau de bord.	- Créer interface Tableau de bord- Afficher dépenses par statut- Intégrer graphiques simples (optionnel)	Moyenne	5
US5	En tant qu’administrateur, je veux pouvoir valider une dépense soumise.	- Interface pour voir toutes les dépenses- Boutons pour approuver/rejeter- Changer statut de la dépense	Moyenne	8

Chapitre 3 : Planification des Releases

3.1 Introduction

Ce chapitre détaille la planification des deux premières releases du projet, avec les user stories planifiées, leurs priorités, estimations, ainsi que la conception associée pour garantir une bonne organisation et une compréhension claire du fonctionnement du système.

3.2 Sprint 1

User Story ID	User Story	Tâches techniques	Priorité	Estimation (points)
US1	Authentification	- Créer modèle Utilisateur - Implémenter authentification (JWT/session) - Créer écran de connexion	Haute	5
US2	Soumission d'une dépense avec justificatif	- Créer entité Dépense - Formulaire de soumission - Upload de justificatif - Liaison utilisateur connecté	Haute	8
US4	Téléchargement des justificatifs	- Stockage fichiers - Lien fichier dans la base de données - Sécurisation accès	Haute	5

Objectifs Sprint 1

- Permettre aux utilisateurs de s'authentifier de manière sécurisée.
- Offrir une fonctionnalité de soumission de dépenses avec justificatif.
- Assurer la gestion et la sécurité des fichiers justificatifs.

3.3 Sprint 2

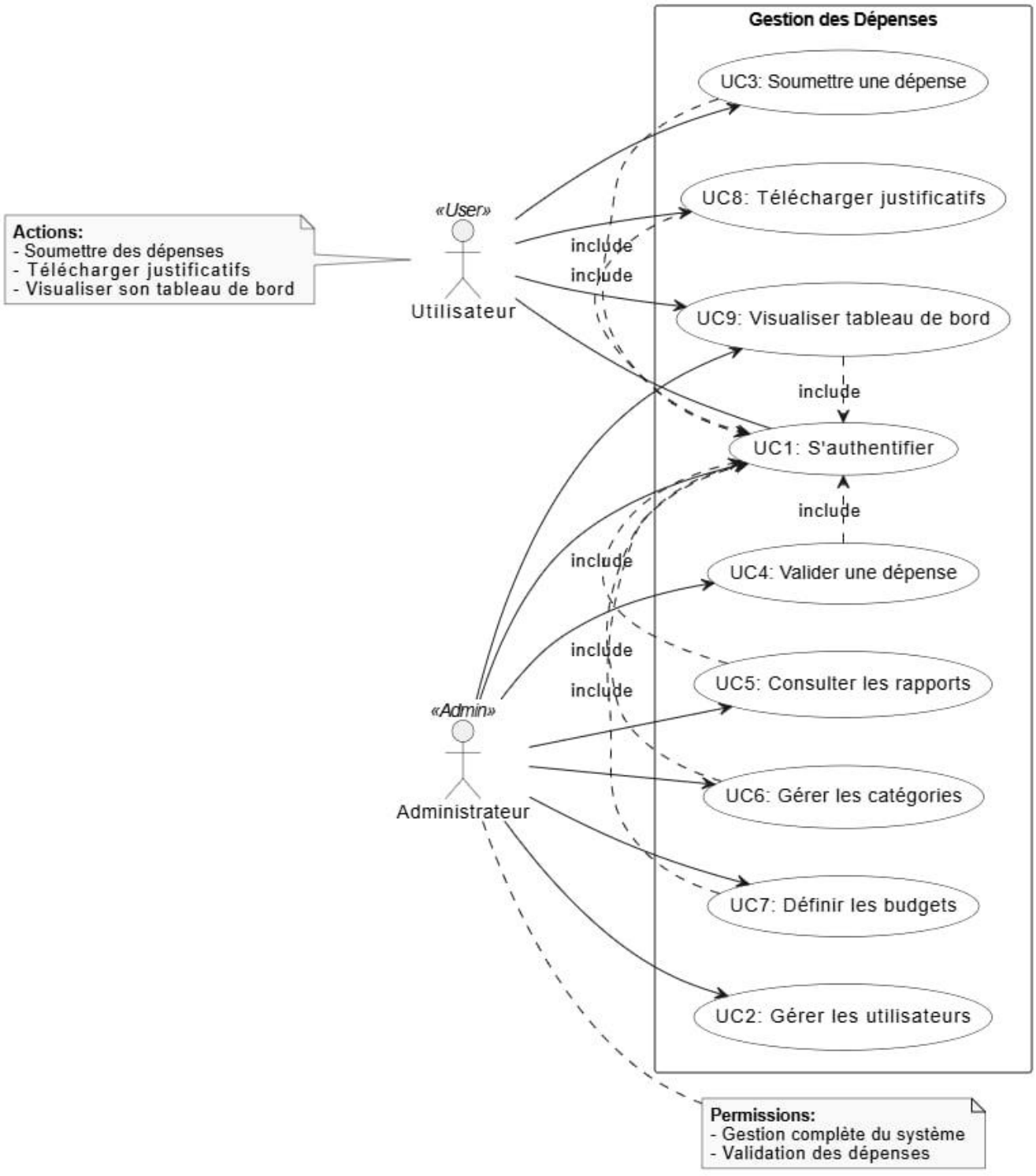
User Story ID	User Story	Tâches techniques	Priorité	Estimation (points)
US3	Visualisation des dépenses	- Création d'un tableau de bord - Affichage par statut - Intégration de graphiques simples	Moyenne	5
US5	Validation des dépenses	- Interface admin - Actions approuver/rejeter - Mise à jour statut	Moyenne	8

Objectifs Sprint 2

- Fournir une interface pour consulter et filtrer les dépenses.
- Permettre aux administrateurs de valider ou rejeter les dépenses.

3.4 Diagramme de cas d'utilisation

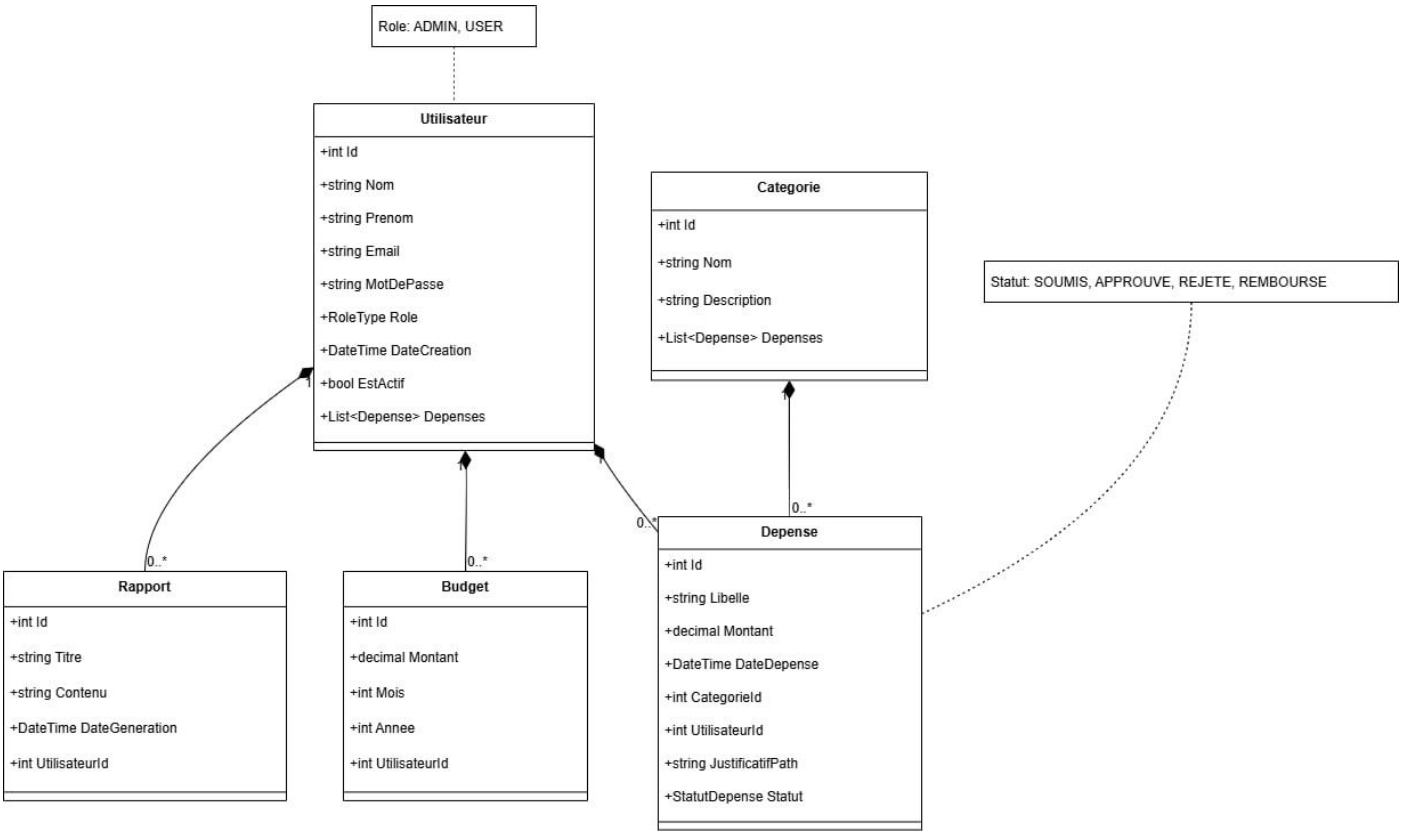
Le diagramme de cas d'utilisation présente les interactions principales entre les utilisateurs (utilisateur simple et administrateur) et le système :



Ce diagramme permet de clarifier les fonctionnalités clés et les rôles des utilisateurs.

3.4.2 Diagramme de classes

Le diagramme de classes détaille les entités principales impliquées dans les sprints 1 et 2, notamment :



Cette modélisation facilite la conception de la base de données et l'implémentation des fonctionnalités.

Conclusion

Le projet de gestion des dépenses a permis de développer une application web fonctionnelle facilitant la soumission, le suivi et la validation des dépenses. Grâce à l'utilisation de **Spring Boot** pour le backend, **Angular** pour le frontend et **Git** pour le versionnage, l'application offre une interface fluide, sécurisée et moderne. Les fonctionnalités clés comme l'authentification, le dépôt de justificatifs et la validation des dépenses ont été réalisées avec succès, répondant aux besoins des utilisateurs et des administrateurs.