République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane Mira de Bejaia
Faculté des Sciences Exactes
Département d'Informatique

RAPPORT DE PROGET AGL

Développement rapide d'une application Web pour la gestion d'une bibliothèque.

Réaliser par :

- BERKATI Farah
- BENYOUB Adel
- AYADENE Meriem
- ARAR Manal

Etudiants en M2GL (G1)

Année universitaire : 2023/2024

Table des matières

INTRODUCTION GENERAL : 4

CHAPITRE 1 :PRESENTATION DES AGL'S 5

Introduction : 6

Définition du terme Atelier de Génie Logiciel (AGL) : 6

Catégories d'AGL : 6

Critères d'adoption d'un AGL : 6

Les avantages des AGL : 7

Les différents types des ateliers génie logiciel : 7

Conclusion : 7

CHAPITRE 2 :PRESENTATION DE LA METHODE SCRUM 8

Introduction 9

Définition de la méthode Scrum: 9

Principes Fondamentaux de Scrum : 9

Rôles dans Scrum 9

Artefacts Scrum : 10

Événements Scrum : 10

Conclusion 10

CHAPITRE 3 :SPECIFICATION DES EXIGENCE DU SYSTEME 11

Introduction : 12

Exigences fonctionnelles selon chaque acteur de système : 12

Exigences non fonctionnelles : 13

Conclusion : 13

CHAPITRE 4 :PLANNIFICATION DE PROJET ET PESENTATION DES UTILES 14

Introduction : 15

Planification Agile avec Scrum : 15

1. présentation d'outils (ClickUp) : 15
2. Les rôles Scrum : 16
3. planification des sprints : 16
4. Product backlog : 16

Planification et gestion des taches de projet : 18

1. Définition des tâches : 18
2. Diagramme de PERT 19
3. Diagramme de Gantt 19

Conclusion : 20

CHAPITRE 5 : CONCEPTION ET REALISATION 21
Introduction : 22
Conception et Modélisation UML : 22
1. Présentation des outils : Visual Paradigme 22
2. Diagramme de Cas d'utilisation 22
3. Diagramme de Séquences : 24
4. Diagramme de Class : 25
5. Modèle relationnel : 26
Réalisation et Programmation : 27
1. Prototype et Design : 27
a) Présentation des outils : FIGMA 27
b) Carte Graphique : 27
c) Design UI/UX 28
2. Codage et développement : 29
CONCLUSION GENERALE: 32
RESUME: 32

Table des illustrations :

Figure 1 : Méthodologie SCRUM 9
Figure 2 : Logo ClickUp. 15
Figure 3 : Tableau de Produck Backlog 17
Figure 4 : Tableau des Taches Planifier 18
Figure 5 : Diagramme de PERT. 19
Figure 6 : Diagramme de Gantt. 20
Figure 7 : Logo Visual paradigme 22
Figure 8 : Diagramme de Cas d'utilisation 23
Figure 9 : Diagramme Séquence demande Emprunt 24
Figure 10 : Diagramme de Séquence Réstitution 25
Figure 11 Diagramme de Class de Conception 26
Figure 12 :Logo Figma 27
Figure 13 : Carte Graphique 27

INTRODUCTION GENERAL :

Dans le monde numérique d'aujourd'hui, la gestion efficace de l'information est essentielle, surtout dans des domaines cruciaux comme les bibliothèques. Les bibliothèques, en tant que piliers de connaissances, ont évolué au fil des ans pour devenir des centres d'apprentissage dynamiques, nécessitant des systèmes de gestion modernes et sophistiqués. Notre projet ambitieux de développement d'une application web pour la gestion d'une bibliothèque répond à cette nécessité croissante en exploitant les dernières avancées en matière de génie logiciel.

L'objectif central de notre projet est de créer une application de gestion de bibliothèque robuste, conviviale et hautement efficace. Pour atteindre cet objectif, nous allons mettre en œuvre les Ateliers de Génie Logiciel (AGL), un ensemble d'outils et de méthodologies qui vont révolutionner notre approche du développement logiciel. Les AGL offrent une structure qui favorise la collaboration, automatise les tâches répétitives et garantit la qualité, accélérant ainsi le cycle de développement.

CHAPITRE 1 : PRESENTATION DES AGL'S

Introduction :

Dans ce premier chapitre nous allons présenter les concepts essentiels des Ateliers de Génie Logiciel (AGL), et comprendre leur importance dans le contexte du développement des applications et logiciels. Nous allons examiner les avantages clés des AGL en matière d'efficacité, de collaboration et de qualité des logiciels.

Définition du terme Atelier de Génie Logiciel (AGL) :

Un ensemble cohérent d'outils informatiques formant un environnement d'aide à la conception, au développement et à la mise au point de logiciels d'application spécialisés. On retrouvera par exemple dans un AGL des dictionnaires de données, des outils permettant de réaliser des diagrammes, pour faciliter la phase d'analyse et de conception des applications. Puis des générateurs de code ainsi que des aides à la mise au point (encore appelés débogueurs) viendront accélérer la production et la finalisation de l'application.

Catégories d'AGL :

Les AGL peuvent être classés selon plusieurs aspects :

- Richesse du support : ensemble d'outils, outils intégrés, aide à la démarche.
- Type de problèmes : logiciels embarqués, temps réel, "business applications", applications métiers
- Type de projet d'ingénierie logicielle : développement logiciel (cycle de vie), intégration de systèmes, système à base de connaissance.
- Ampleur du projet : complexité, nombres de participants, durée ...
- Gestion des ressources du projet : les considérations managériales des ressources mises en œuvre dans le projet sont-elles prises en compte (planification, ordonnancement, …).
- Phase du cycle de développement prises en compte : conception et/ou développement.

Critères d'adoption d'un AGL :

Choisir d'utiliser un AGL pose certains questionnements :

- Investissement de ressources : Coût d'adoption d'une technologie AGL.
- Aide et Support technique disponible : évaluation à long terme de l'exploitation du logiciel.
- Méthodes et processus de GL existants dans l'entreprise : adéquation entre ce qui est fait par les 'acteurs' et ce qui est proposé par les outils
- Montée en charge : aussi bien en termes d'ampleur du projet que de la performance des applications générées avec l'outil.

Les avantages des AGL :

- **Accélération du Développement :** Les AGL automatisent de nombreuses tâches de développement, ce qui accélère le processus de création de logiciels complexes.
- **Collaboration :** Les AGL favorisent la collaboration entre les membres de l'équipe grâce à des outils de communication en temps réel et de partage de code.
- **Gestion des Versions :** Les AGL intègrent des systèmes de gestion de versions comme Git, ce qui permet de suivre l'évolution du code source, de gérer les branches et de fusionner les modifications de manière transparente.
- **Amélioration de la Qualité :** Les AGL offrent des outils de test automatisés, d'analyse statique du code et de détection des erreurs, ce qui contribue à améliorer la qualité du logiciel.

1. **Flexibilité et Adaptabilité :** Les AGL sont flexibles et peuvent être adaptés aux besoins spécifiques du projet, permettant ainsi aux équipes de travailler selon leurs préférences et leurs méthodologies.

Les différents types des ateliers génie logiciel :

On distingue essentiellement deux types d'AGL selon la nature des outils intégrés :

1. **Les environnements de conception (upper-case) :** Axés sur l'analyse et la conception, ces ateliers intègrent des outils tels que des éditeurs de diagrammes (avec vérification syntaxique), des dictionnaires de données et des générateurs de code.

Exemple : **BOUML** est un logiciel de création de diagrammes UML. Permet de dessiner des diagrammes suivant la norme UML 2.0, Il est multiplateformes.

2. **Les environnements de développement (lower-case**) : ces ateliers se concentrent sur les phases d'implémentation et de test du processus logiciel. Ils intègrent généralement des éditeurs (éventuellement dirigés par la syntaxe), des générateurs d'interfaces homme/machine, des SGBD, des compilateurs, optimiseurs, debuggers.

Exemple : **WebDev** (pour la conception d'application web) et **WinDev** Mobile (pour la conception d'application mobile)

Conclusion :

Nous avons vu dans cette section, les fondamentales des AGLs en précisant leurs types, leurs avantages dans le développement rapide des logiciels de qualité, et on peut déduire que même si ces AGL ont atteint un niveau élevé, tant en termes de couverture des différentes étapes du processus logiciel, ils restent un sujet controversé et sont loin d'être au niveau attendu.

CHAPITRE 2 : PRESENTATION DE LA METHODE SCRUM

Introduction

Dans ce chapitre, nous plongerons dans les profondeurs de la méthodologie Scrum, un cadre Agile qui sera le pilier de notre approche de gestion de projet. Nous explorerons les principes fondamentaux de Scrum, ses rôles clés, ses artefacts cruciaux et ses événements essentiels. Comprendre Scrum est essentiel, car cela guidera notre équipe tout au long du développement de notre application de gestion de bibliothèque, permettant ainsi une adaptation agile aux changements et un rythme de développement soutenu.

Définition de la méthode Scrum:

La méthode Scrum est une méthode agile de gestion de projets informatiques privilégiant la communication, et facilitant les réorientations opportunes. C'est désormais la méthode privilégiée pour les démarches dites "agiles». Fort de son succès dans l'univers informatique, elle est maintenant déployée en entreprise comme nouvelle organisation du fonctionnement en mode projet.

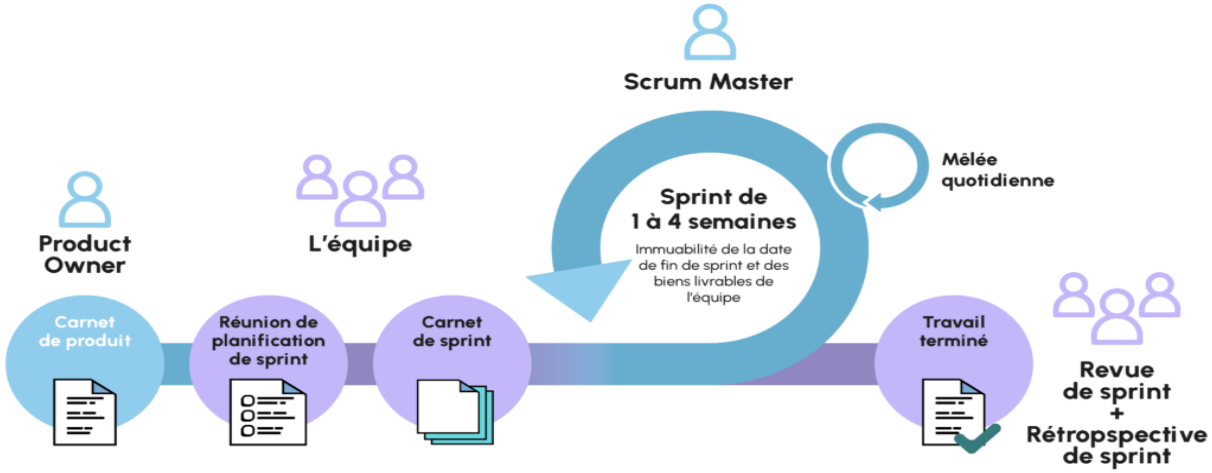


Figure 1 : Méthodologie SCRUM

Principes Fondamentaux de Scrum :

Scrum est une méthodologie Agile basée sur la transparence, l'inspection et l'adaptation. Les principes clés de Scrum sont la collaboration, la communication fréquente et le développement itératif. Il encourage l'auto-organisation des équipes, la flexibilité face aux changements de priorité et l'attention constante à la qualité.

Rôles dans Scrum :

- **Product Owner :** Le Product Owner est le gardien de la vision du produit. Il définit les fonctionnalités du produit, priorise le backlog et s'assure que l'équipe de développement comprend clairement les exigences.
- **Scrum Master :** Le Scrum Master est le facilitateur du processus Scrum. Il s'assure que l'équipe suit les principes de Scrum, supprime les obstacles, et facilite les réunions Scrum, aidant ainsi l'équipe à rester concentrée et à progresser.

- **Équipe de Développement :** L'équipe de développement est composée de professionnels qui réalisent le travail. Ils sont auto-organisés, inter fonctionnels et ont tous les talents nécessaires pour créer le produit.

Artefacts Scrum :

- **Product Backlog :** Le Product Backlog est une liste dynamique et priorisée des fonctionnalités à développer. Il est géré par le Product Owner et représente les besoins du client pour le produit.
- **Sprint Backlog :** Le Sprint Backlog est une liste des tâches que l'équipe de développement prévoit de réaliser pendant le sprint en cours. Il est créé à partir du Product Backlog et est géré par l'équipe de développement.
- **Incrément :** L'Incrément est la somme de toutes les fonctionnalités achevées au terme d'un sprint. Il doit être potentiellement livrable, c'est-à-dire qu'il doit être fonctionnel et prêt à être mis en production.

Événements Scrum :

- **Sprint :** Un Sprint est une période de temps fixe (généralement deux à quatre semaines) pendant laquelle l'équipe de développement travaille pour créer un Incrément potentiellement livrable.
- **Sprint Planning :** Au début de chaque Sprint, l'équipe de développement et le Product Owner se réunissent pour définir les objectifs du Sprint et sélectionner les tâches du Product Backlog à réaliser.
- **Daily Scrum :** Chaque jour pendant le Sprint, l'équipe de développement se réunit pour partager les progrès, discuter des obstacles et planifier le travail pour les prochaines 24 heures.
- **Sprint Review :** À la fin de chaque Sprint, l'équipe de développement présente l'Incrément lors d'une réunion appelée Sprint Review. Le Product Owner évalue l'Incrément et décide des prochains objectifs.
- **Sprint Retrospective :** Après la Sprint Review, l'équipe de développement se réunit en Sprint Rétrospective pour réfléchir sur le Sprint écoulé, identifier ce qui a bien fonctionné et ce qui peut être amélioré, afin d'ajuster leurs pratiques pour les Sprints suivants.

Conclusion

En somme, la méthodologie Scrum représente un cadre souple et collaboratif pour la gestion de projets. En mettant l'accent sur la transparence, la communication régulière et le développement itératif, elle favorise une adaptation fluide aux changements et une attention constante à la qualité du produit. Avec des rôles bien définis, des artefacts clés et des événements structurés, Scrum offre un chemin clair pour la réalisation de produits en phase avec les attentes des clients. En choisissant Scrum comme fondement pour notre projet de développement d'une application de gestion de bibliothèque, nous nous engageons à adopter une approche agile, permettant ainsi une gestion efficace du changement et une progression continue du produit.

CHAPITRE 3 : SPECIFICATION DES EXIGENCE DU SYSTEME

Introduction :

La spécification des exigences, également connue sous le nom de documentation, est un processus consistant à noter toutes les exigences du système et de l'utilisateur sous la forme d'un document. Ces exigences doivent être claires, complètes, exhaustives et cohérentes

Exigences fonctionnelles selon chaque acteur de système :

Bibliothécaires :

1. **Ajout de livres :** Les bibliothécaires doivent pouvoir ajouter de nouveaux en fournissant des détails tels que le titre, l'auteur, le genre et le numéro d'inventaire.
2. **Mise à jour des informations :** Les bibliothécaires doivent être en mesure de mettre à jour les informations sur les livres existants, y compris les détails sur les exemplaires disponibles et empruntés.
3. **Gestion des exemplaires :** Les bibliothécaires doivent pouvoir ajouter de nouveaux exemplaires d'ouvrages, spécifiant leur référence et leur localisation dans les rayons.
4. **Gestion des emprunts et des restitutions :** Les bibliothécaires doivent pouvoir enregistrer les emprunts d'ouvrages par les abonnés, y compris la date d'emprunt et la date de retour prévue. Ils doivent également enregistrer les restitutions d'ouvrages, marquant ainsi la fin des emprunts.
5. **Génération de rapports d'emprunt :** Les bibliothécaires doivent pouvoir générer un rapport quotidien de l'ensemble des emprunts (date, heure, code) et un rapport quotidien détaillé de tous les emprunts de la journée, incluant les informations sur les abonnés, les ouvrages, les dates et les heures.

Membres de la Bibliothèque (lecteur/Abonné) :

1. **Recherche de livres** : Les membres de la bibliothèque doivent pouvoir rechercher des livres dans la base de données en utilisant des critères tels que le titre, l'auteur, ou la catégorie. Ils doivent pouvoir consulter les détails des exemplaires disponibles, y compris le nombre d'exemplaires disponibles et leur emplacement dans la bibliothèque.
2. **Emprunts d'ouvrages :** Les membres de la bibliothèque doivent pouvoir emprunter des ouvrages en utilisant leur identifiant d'abonné. Le système doit vérifier le nombre maximum d'ouvrages empruntables (trois) par abonné.
3. **Prolongation d'emprunts :** Les membres de la bibliothèque doivent pouvoir demander la prolongation exceptionnelle du délai d'emprunt d'un ouvrage de trois à cinq semaines. La prolongation doit être autorisée uniquement si aucune réservation n'a été faite pour l'ouvrage.
4. **Gestion des réservations :** Les membres de la bibliothèque doivent pouvoir réserver des ouvrages qui sont actuellement empruntés par d'autres membres. Ils doivent également pouvoir annuler des réservations.

Gestionnaire/Administrateur :

1. **Gestion des abonnés :** Le gestionnaire doit pouvoir gérer les inscriptions des abonnés, enregistrer leurs informations personnelles et définir la durée de validité de leur abonnement. Il doit avoir la possibilité de définir le statut d'abonné pour les étudiants, les enseignants et les abonnés externes.
2. **Gestion des abonnés externes :** Le gestionnaire doit limiter le nombre d'abonnés externes à environ 10% des inscrits chaque année. Il doit permettre aux abonnés externes de s'inscrire exceptionnellement en échange d'une cotisation.
3. **Gestion des pénalités :** L'administrateur doit être en mesure de gérer les pénalités imposées aux emprunteurs en cas de retards ou de non-restitution d'ouvrages dans les délais. Le système doit bloquer temporairement l'accès aux emprunts pour les emprunteurs pénalisés.

Exigences non fonctionnelles :

- **Sécurité** : Garantir la confidentialité et la sécurité des données des abonnés.
- **Performances** : Assurer des temps de réponse rapides et une gestion efficace des opérations.
- **Convivialité** : Fournir une interface utilisateur intuitive et conviviale.
- **Extensibilité** : Permettre l'ajout facile de nouvelles fonctionnalités à l'avenir.
- **Maintenabilité** : Faciliter la maintenance, les mises à jour et les corrections de bugs.
- **Disponibilité** : Assurer un accès continu pendant les heures d'ouverture avec une maintenance minimale.
- **Scalabilité** : S'adapter à la variation de la charge en ajoutant des ressources au besoin.
- **Intégration** : S'intégrer avec d'autres systèmes et bases de données universitaires.

Conclusion :

Une spécification des exigences est un document qui décrit les besoins spécifiques d'un projet ou d'un système. La spécification des exigences est importante car elle sert de base à tous les travaux futurs sur le projet. La spécification des exigences logicielles (SRS) est différente de la spécification des exigences métier (BRS), bien qu'elles soient liées

CHAPITRE 4 : PLANNIFICATION DE PROJET ET PESENTATION DES UTILES

Introduction :

La planification est un élément essentiel pour la réussite de tout projet, et notre projet de gestion de bibliothèque ne fait pas exception. Notre approche combine deux méthodes complémentaires : la planification Agile avec Scrum et la gestion traditionnelle de projet.

Dans la première partie de notre rapport, nous explorerons les principes agiles avec un focus sur les rôles Scrum, les sprints, et le Product Backlog.

La seconde partie se concentre sur la planification traditionnelle, avec la définition des tâches, le Diagramme de Pert pour visualiser les dépendances, et le Diagramme de Gantt pour la gestion temporelle. Cette combinaison vise à assurer une planification robuste et adaptable pour mener à bien notre projet de gestion de bibliothèque.

Planification Agile avec Scrum :

1. présentation d'outils (ClickUp) :

Figure 2 : Logo ClickUp.

ClickUp est une plateforme de gestion de projet en ligne qui vise à simplifier et à centraliser la collaboration au sein des équipes. Conçu pour répondre aux besoins variés des projets, ClickUp offre une approche flexible et personnalisable pour la gestion des tâches, la collaboration et la communication

ClickUp propose un éventail de fonctionnalités qui peuvent être adaptées aux principes de la gestion agile. En raison de ses caractéristiques polyvalentes et adaptatives, nous avons conclu que ClickUp serait un choix optimal pour répondre aux besoins spécifiques de notre projet. La flexibilité offerte par la définition des sprints et Product Backlog, permet une gestion agile efficace. De plus, la capacité à suivre en temps réel l'avancement des tâches, à assigner des responsabilités et à gérer les dépendances, offre une adaptabilité cruciale. Cette souplesse, associée à une interface conviviale, favorise une collaboration

transparente au sein de l'équipe et facilite la prise de décision proactive.

2. Les rôles Scrum :

Dans le cadre de la méthodologie Scrum, l'identification des personnes pour chaque rôle est une étape cruciale pour assurer la clarté des responsabilités et la cohésion de l'équipe. Voici une suggestion de personnes pour chaque rôle :

- **Product owner :** Définir la vision du projet, prendre des décisions éclairées sur les fonctionnalités à développer c'est le rôle de Mm Djebbar.
- **Scrum master :** Personne chargée de faciliter l'application de Scrum au sein de l'équipe dans notre cas ça sera le membre BERKATI Farah
- **Équipe de Développement :** Responsable de la réalisation des éléments du Product Backlog lors des sprints. Représenter par les tous les s de group

3. Planification des sprints :

La planification des sprints est une composante clé de la méthodologie Scrum, où un sprint est défini comme une itération de développement de courte durée, généralement d'une à quatre semaines. Chaque sprint a pour objectif de livrer une version incrémentale du produit, ajoutant de nouvelles fonctionnalités ou améliorations. Ces périodes définies permettent à l'équipe de se concentrer sur des objectifs spécifiques, favorisant la flexibilité et l'adaptation aux changements.

Au cours de notre projet, nous avons planifié et exécuté quatre sprints successifs. Ces sprints sont clairement détaillés dans le Product Backlog, démontrant notre engagement envers la livraison continue de fonctionnalités de valeur. Chaque sprint a permis à l'équipe de développement de travailler sur des éléments prioritaires du Product Backlog, favorisant ainsi une approche itérative et incrémentale pour atteindre les objectifs du projet.

4. Product backlog :

Le Product Backlog, élément central de la méthodologie Scrum, est une liste dynamique et priorisée de toutes les fonctionnalités à développer. Il agit comme un carnet de commandes évolutif, reflétant la vision globale du produit et s'ajustant continuellement en fonction des changements de priorités, des retours d'utilisateurs et des exigences émergentes.

Les fonctionnalités du produit sont formulées sous forme de "User Stories", courtes et centrées sur les utilisateurs, offrant ainsi une compréhension claire des besoins à satisfaire.

La priorisation constante du Product Backlog permet de maximiser la valeur livrée à chaque itération et de garantir que l'équipe de développement se concentre sur les éléments les plus cruciaux pour le succès du projet.

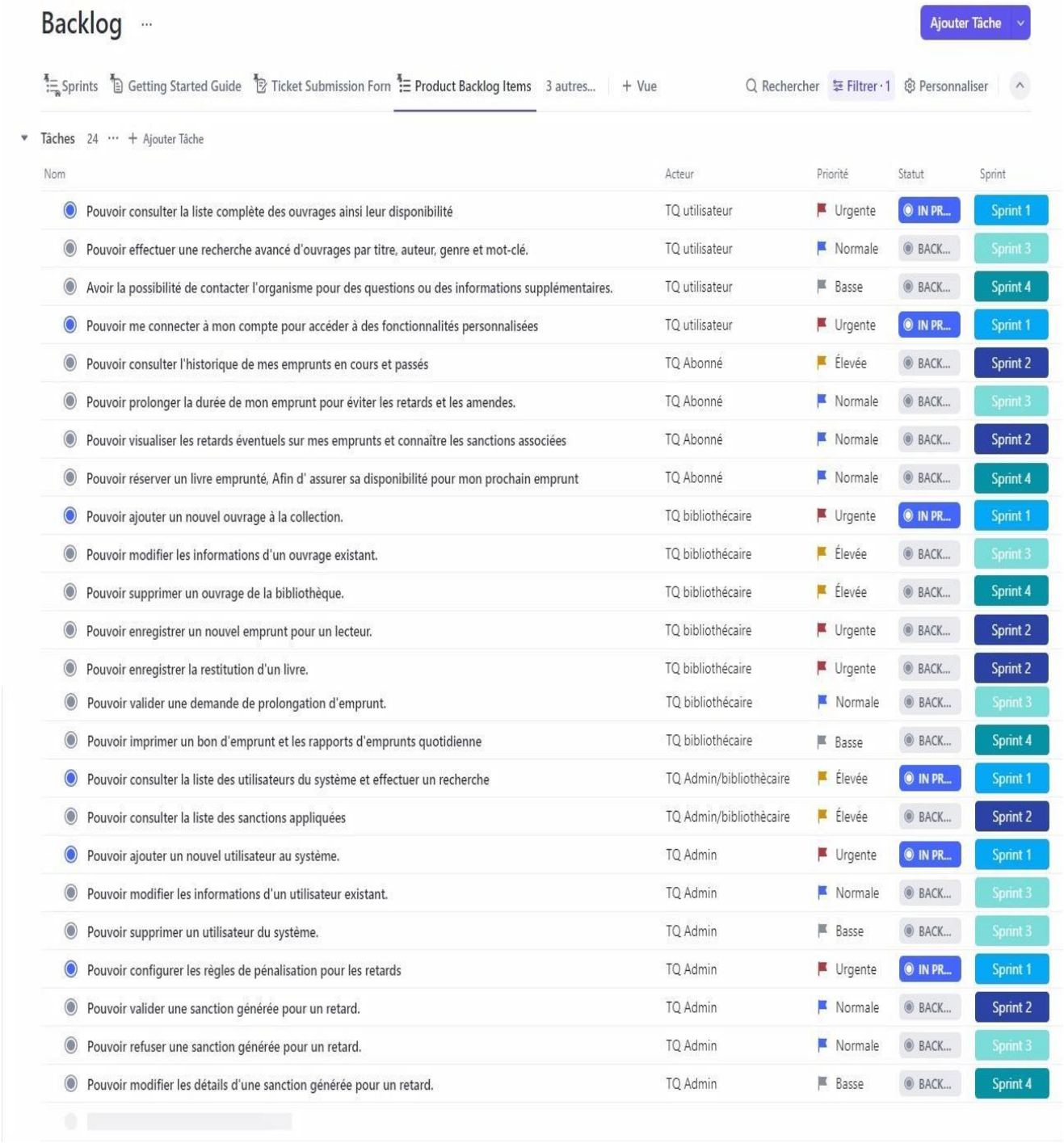


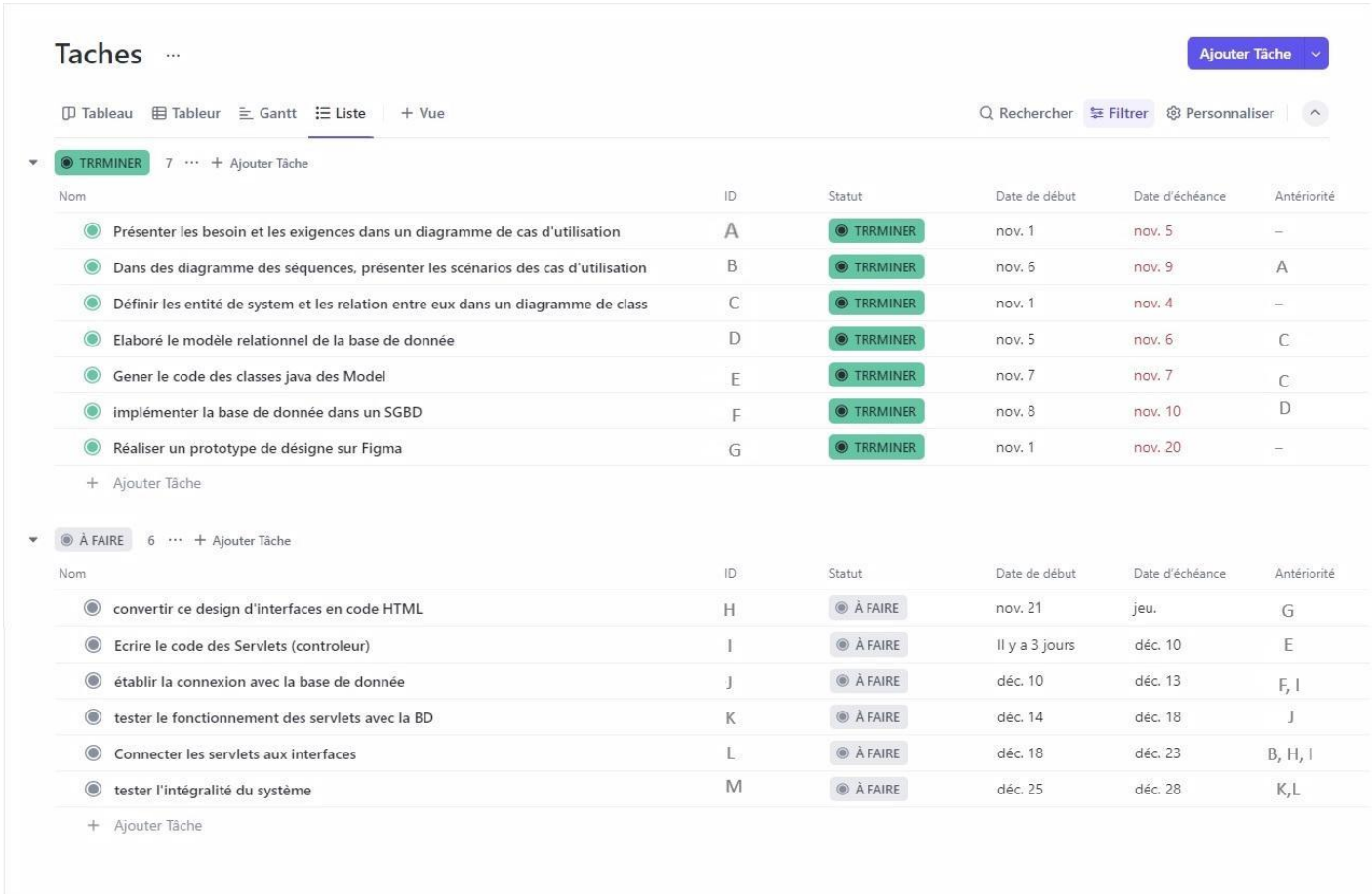
Figure 3 : Tableau de Produck Backlog

Planification et gestion des taches de projet :

1. Définition des tâches :

La définition des tâches dans un projet constitue un pilier essentiel de la gestion, apportant une clarté cruciale sur les objectifs spécifiques, simplifiant la répartition des responsabilités, permettant une planification temporelle réaliste, identifiant les interdépendances entre les activités, facilitant le suivi de l'avancement, et renforçant la communication au sein de l'équipe. Ces aspects fondamentaux établissent une base robuste pour une gestion proactive et couronnée de succès du projet.

Le tableau ci-dessous offre une représentation claire et structurée des tâches essentielles de notre projet



Taches ···

Ajouter Tâche

Tableau | Tableur | Gantt | Liste | + Vue | Rechercher | Filtrer | Personnaliser

TRRMINER 7 ··· + Ajouter Tâche

Nom	ID	Statut	Date de début	Date d'échéance	Antériorité
Présenter les besoin et les exigences dans un diagramme de cas d'utilisation	A	TRRMINER	nov. 1	nov. 5	–
Dans des diagramme des séquences, présenter les scénarios des cas d'utilisation	B	TRRMINER	nov. 6	nov. 9	A
Définir les entité de system et les relation entre eux dans un diagramme de class	C	TRRMINER	nov. 1	nov. 4	–
Elaboré le modèle relationnel de la base de donnée	D	TRRMINER	nov. 5	nov. 6	C
Gener le code des classes java des Model	E	TRRMINER	nov. 7	nov. 7	C
implémenter la base de donnée dans un SGBD	F	TRRMINER	nov. 8	nov. 10	D
Réaliser un prototype de désigne sur Figma	G	TRRMINER	nov. 1	nov. 20	–

+ Ajouter Tâche

À FAIRE 6 ··· + Ajouter Tâche

Nom	ID	Statut	Date de début	Date d'échéance	Antériorité
convertir ce design d'interfaces en code HTML	H	À FAIRE	nov. 21	jeu.	G
Ecrire le code des Servlets (controleur)	I	À FAIRE	Il y a 3 jours	déc. 10	E
établir la connexion avec la base de donnée	J	À FAIRE	déc. 10	déc. 13	F, I
tester le fonctionnement des servlets avec la BD	K	À FAIRE	déc. 14	déc. 18	J
Connecter les servlets aux interfaces	L	À FAIRE	déc. 18	déc. 23	B, H, I
tester l'intégralité du système	M	À FAIRE	déc. 25	déc. 28	K,L

+ Ajouter Tâche

Figure 4 : Tableau des Taches Planifier

2. Diagramme de PERT

Le diagramme de Pert, acronyme de Program Evaluation and Review Technique, est un outil de gestion de projet qui permet de représenter graphiquement les différentes étapes et les dépendances entre les tâches.

Chaque tâche est représentée par un nœud, et les flèches entre les nœuds indiquent les dépendances et le flux logique du projet. Le diagramme de Pert offre une visualisation claire du chemin critique, c'est-à-dire la séquence d'activités qui détermine la durée totale du projet. Il permet également d'identifier les tâches qui peuvent être effectuées en parallèle.

Le diagramme ci-dessous offre une représentation visuelle des tâches de projet définies dans le tableau des tâches

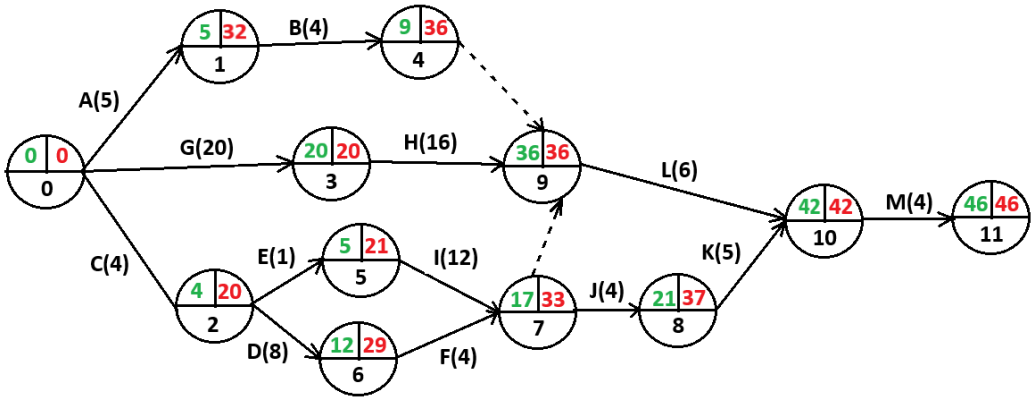


Figure 5 : Diagramme de PERT.

3. Diagramme de Gantt

Le diagramme de Gantt est un outil de gestion de projet visuel qui illustre les tâches du projet sur une ligne de temps. Il offre une représentation graphique des activités planifiées, de leur séquence et de leur durée. Chaque tâche est représentée par une barre horizontale dont la longueur indique la durée prévue, permettant ainsi de visualiser clairement le début, la fin et le chevauchement des différentes phases du projet.

Le diagramme de Gantt généré à partir du tableau des tâches offre une vue chronologique des activités planifiées. Chaque tâche est représentée par une barre horizontale positionnée sur la ligne de temps. Cette représentation visuelle facilite la compréhension des échéances, des chevauchements et des périodes critiques du projet.

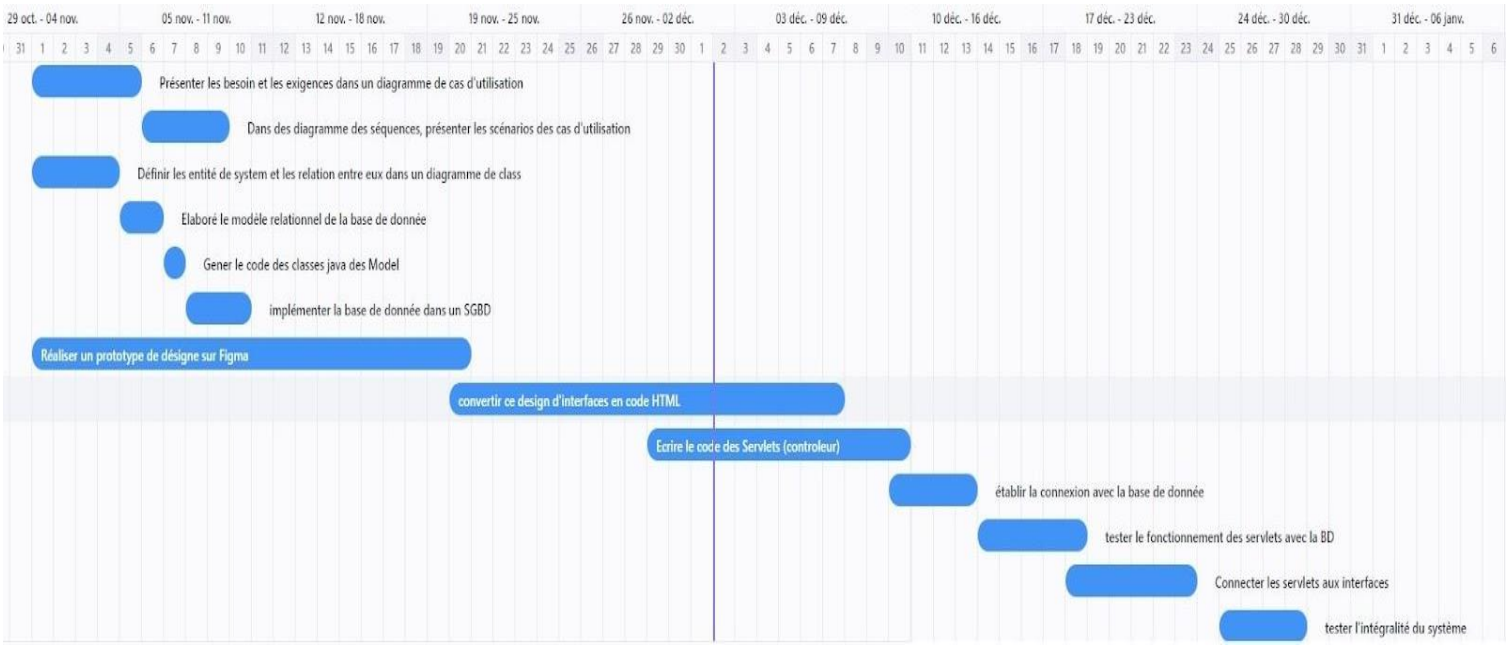


Figure 6 : Diagramme de Gantt.

Conclusion :

La combinaison stratégique de méthodes agiles comme Scrum avec des outils traditionnels a abouti à une planification solide et flexible pour notre projet de gestion de bibliothèque. Cette approche hybride a favorisé une collaboration fluide, une focalisation précise sur les objectifs tout en restant adaptable aux évolutions, assurant ainsi une gestion proactive et réussie.

CHAPITRE 5 : CONCEPTION ET REALISATION

Introduction :

Afin d'aboutir à une meilleure organisation. Nous devons tout naturellement avoir recours un processus de conception orientée objet qui est une démarche de développement et souvent utilisé conjointement au langage UML qui va nous permettre de comprendre et de décrire les besoins de spécifier et documenter le système.

Conception et Modélisation UML :

1. Présentation des outils : Visual Paradigme

Figure 7 : Logo Visual paradigme

Visual Paradigm est un environnement de développement avancé qui se distingue comme un Atelier de Génie Logiciel (AGL) de premier plan, dédié à la conception et à la modélisation de systèmes. Avec une interface conviviale et des fonctionnalités puissantes, Visual Paradigm offre une plateforme intégrée permettant aux développeurs de créer des modèles conceptuels, logiques et physiques .Cet AGL permet de visualiser et d'organiser efficacement les concepts du système, d'identifier les relations entre les composants et de générer automatiquement du code source de haute qualité. Grâce à ses fonctionnalités de modélisation avancées et à sa capacité de génération de code, Visual Paradigm se positionne comme un outil essentiel pour accélérer le processus de développement, tout en garantissant la cohérence et la qualité architecturale du projet.

2. Diagramme de Cas d'utilisation

En langage de modélisation unifié (UML), un diagramme de cas d'utilisation peut servir à résumer les informations des utilisateurs de votre système (également appelés acteurs) et leurs interactions avec ce dernier. La création de ce type de diagramme UML requiert un ensemble de symboles et de connecteurs spécifiques. Lorsqu'ils sont bien conçus, les diagrammes de cas d'utilisation peuvent aider votre équipe à collaborer.

Composants d'un diagramme de cas d'utilisation :

- **Les acteurs** : utilisateurs qui interagissent avec un système. Un acteur peut être une personne, une organisation ou un système externe qui interagit avec votre application ou votre système, il s'agit nécessairement d'objets externes qui produisent ou consomment des données.
- **Le système** : séquence spécifique d'actions et d'interactions entre les acteurs et le système. Un système peut également être appelé scénario.
- **Les objectifs** : résultat final de la plupart des cas d'utilisation. Un diagramme réussi doit décrire les activités et les variantes utilisées pour atteindre l'objectif.

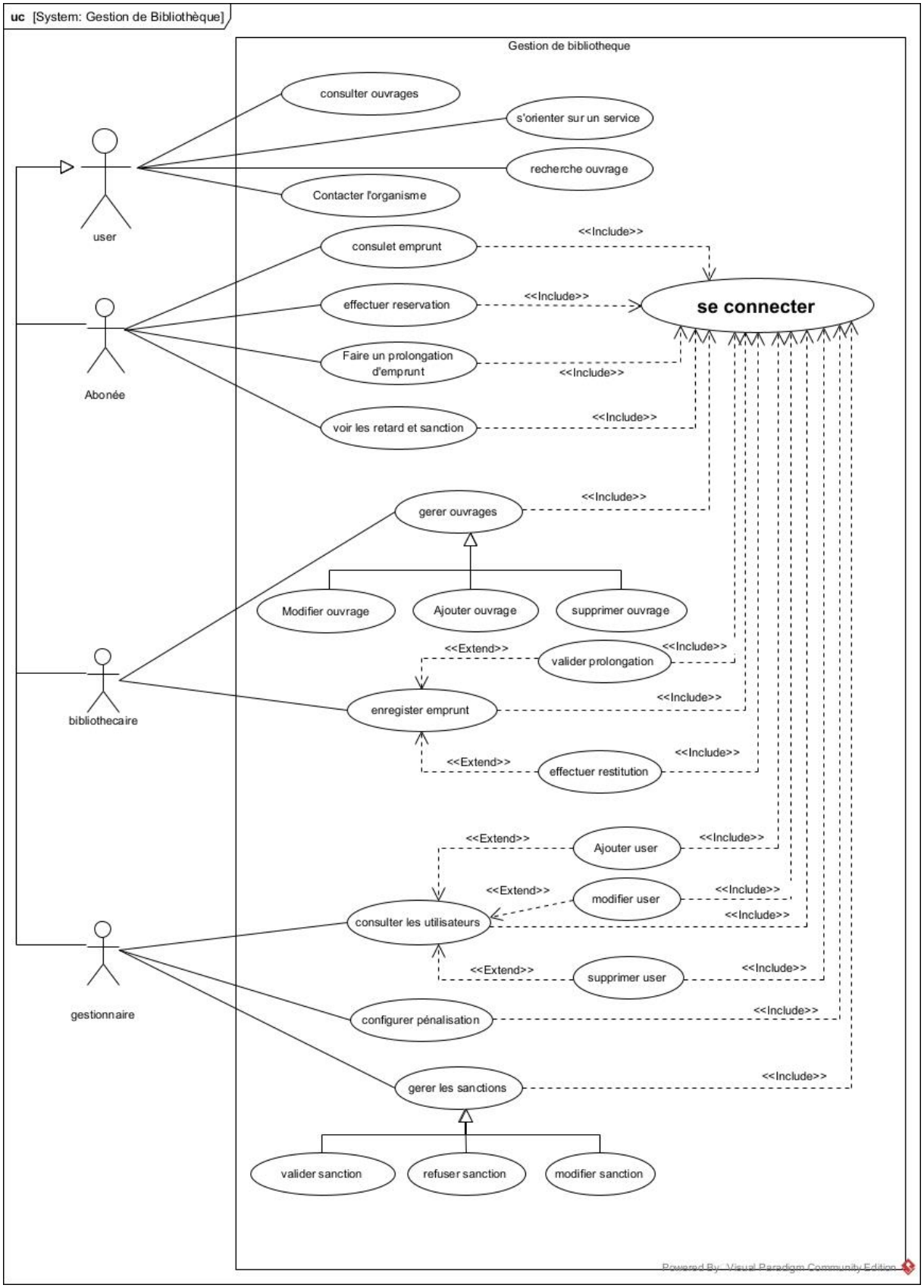


Figure 8 : Diagramme de Cas d'utilisation

3. Diagramme de Séquences :

Un diagramme de séquence est un diagramme d'interaction qui expose en détail la façon dont les opérations sont effectuées : quels messages sont envoyés et quand ils le sont. Les diagrammes de séquence sont organisés en fonction du temps. Le temps s'écoule au fur et à mesure que vous parcourez la page. Les objets impliqués dans l'opération sont répertoriés de gauche à droite en fonction du moment où ils prennent part dans la séquence de messages

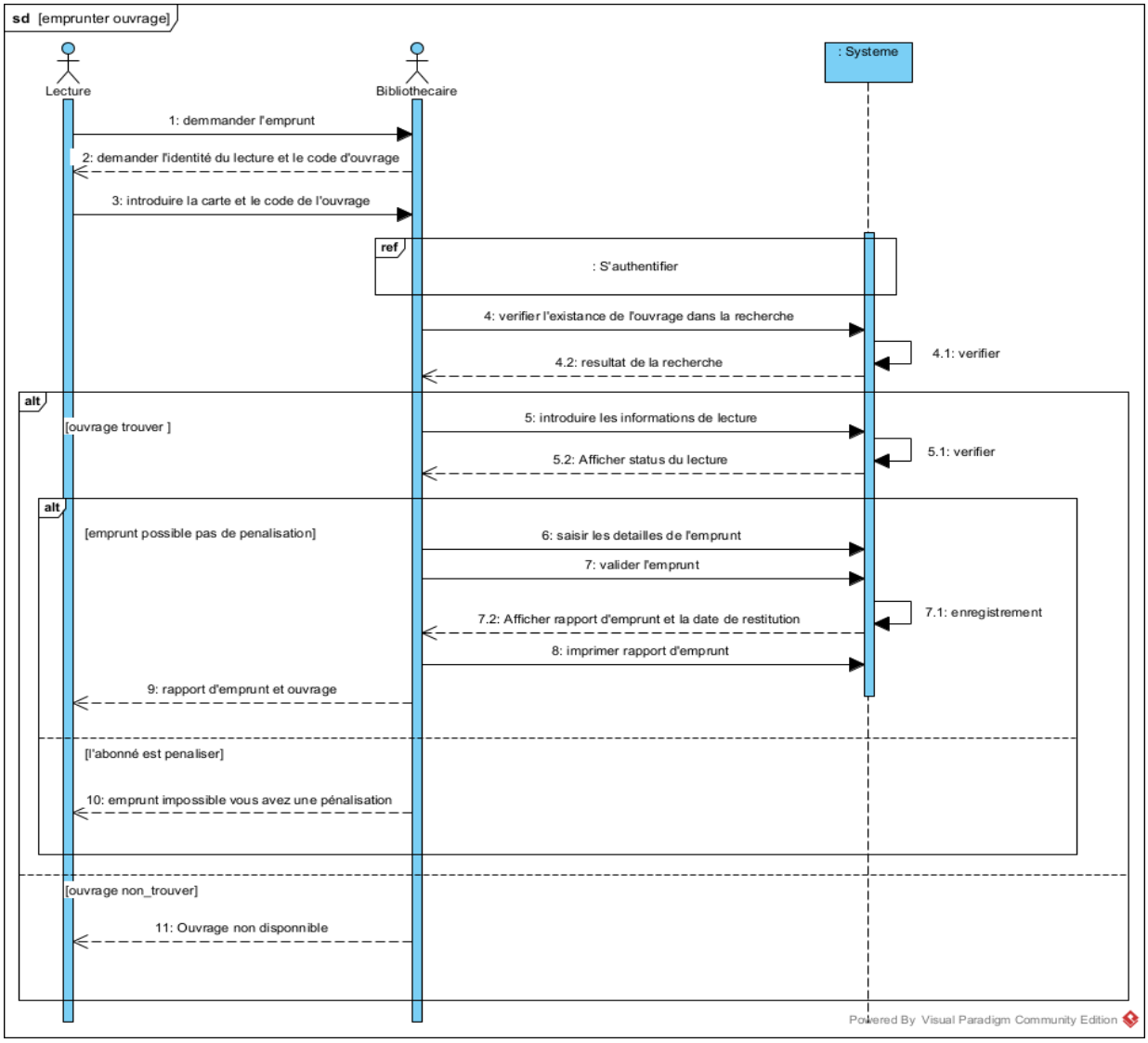


Figure 9 : Diagramme Séquence demande Emprunt

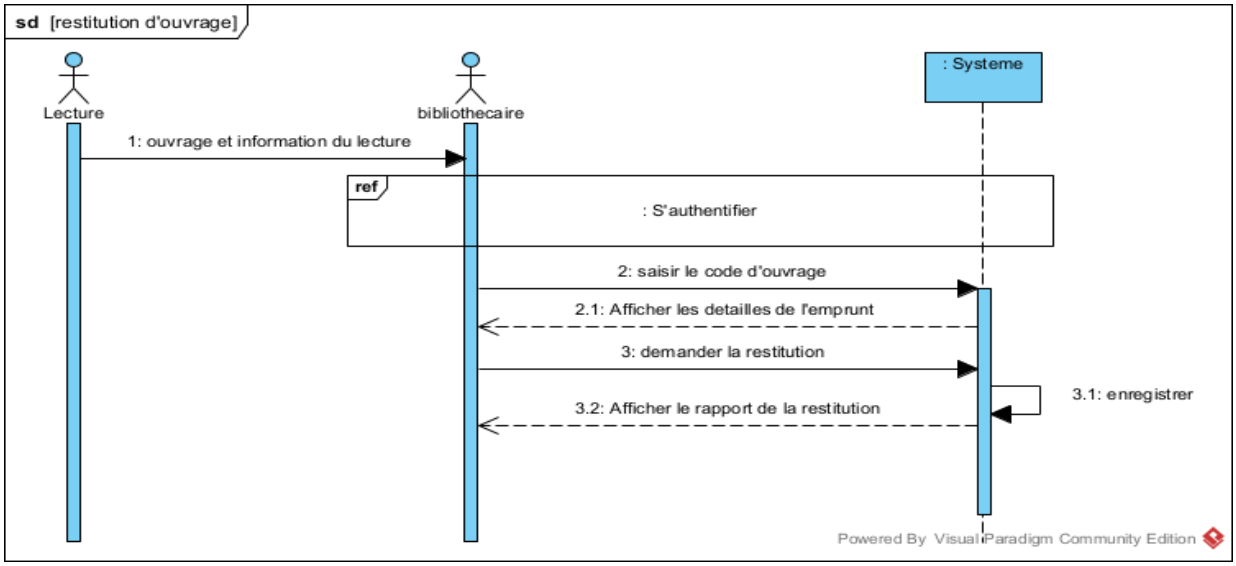


Figure 10 : Diagramme de Séquence Réstitution

4. Diagramme de Class :

Ce diagramme expose la structure statique du système en identifiant les classes, leurs attributs et leurs relations. Il offre une vue détaillée des entités du système et de leurs interactions

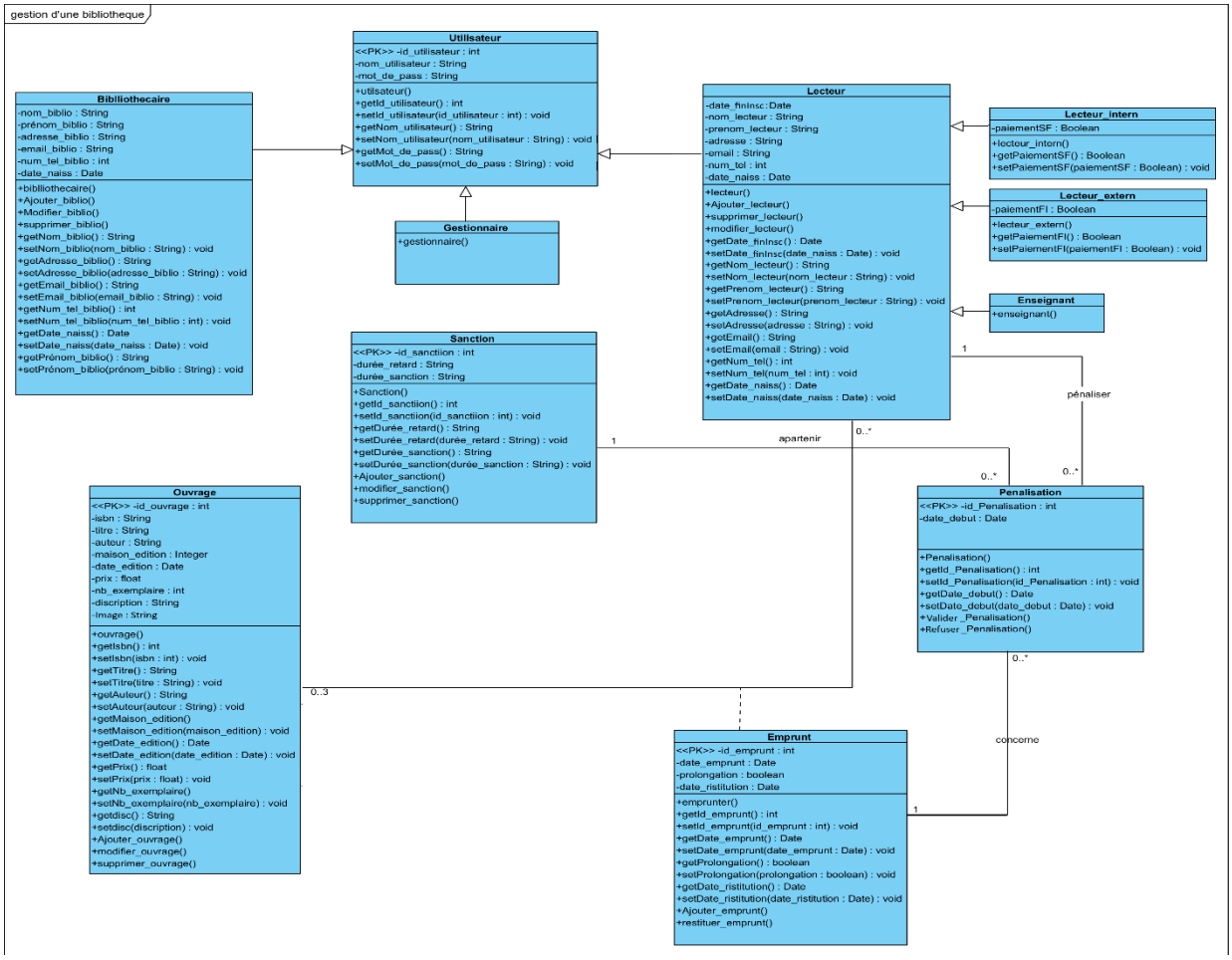


Figure 11 Diagramme de Class de Conception

5. Modèle relationnel :

Ce modèle traduit la structure du système en schémas relationnels, décrivant comment les données sont organisées et liées dans la base de données.

Gestionnaire (**id_utlisateur**, nom_utilisateur, mot_de_pass)

Bibliothécaire (**id_utlisateur**, nom_utilisateur, mot_de_pass, nom_biblio, prénom_biblio, email_biblio, adresse_biblio, num_tel_biblio,date_naiss)

Lecteur (**id_utlisateur**, nom_utilisateur, mot_de_pass,nom_lecteur, prénom_lecteur, adresse_lecteur, email_lecteur, num_tel, date_naiss, date_finInsc, type, paiementSF, paimentFI)

Ouvrage (**id_ouvrage**, isbn, titre, auteur, maison_edition, date_edition, prix, nb_exemplaire, discription)

Emprunt (id_emprunt, date_emprunt, prolongation, date_restitution**,#id_utilisateur,#id_ouvrage**)

Sanction (id_sanction, durée_retard, durée_sanction)

Pénalisation (id_Penatisation, date_debut, **#id_sanction, #id_utilisateur, #id_emprunt**)

Réalisation et Programmation :

1. Prototype et Design :

a) Présentation des outils : FIGMA

Figure 12 :Logo Figma

Figma est une plateforme de conception et de prototypage en ligne prisée pour sa collaboration en temps réel. Elle offre un environnement de travail intuitif pour créer des interfaces utilisateur, des maquettes et des prototypes interactifs. Son atout majeur réside dans sa capacité à permettre à plusieurs utilisateurs de travailler simultanément sur un même projet, favorisant la création collaborative, la visualisation en temps réel des modifications et la création de prototypes interactifs pour tester l'expérience utilisateur, le tout accessible depuis différents appareils grâce à sa version web, ce qui en fait un outil puissant et flexible pour les équipes de conception et de développement.

b) Carte Graphique :

Figure 13 : Carte Graphique

c) Design UI/UX

2. Codage et développement :

a) Présentation d'outils et d'environnement de développement :

Pour développer notre système, on a utilisé des différentes technologies notamment :

- **Java Entreprise Edition :**

Java EE est une plateforme basée sur le langage Java. Elle est une extension de la plateforme Java SE (Java Standard Edition) à laquelle est ajouté un ensemble de classes, méthodes et des fonctions sont rassemblées dans une multitude d'interfaces de programmation (API) définissant la façon dont les composants informatiques s'invoquent l'un l'autre comme les servlets, les pages JSP et les beans.

- **Architecture Java EE**

L'architecture d'une application Java EE se découpe idéalement en au moins trois couches :

* La couche cliente (présentation) : permet de dialoguer avec l'utilisateur. Elle est composée d'une application stand-alone, application web ou applet.

* La couche métier : encapsule les traitements (dans des composants EJB) .

* La couche de données : stocke des données (Fichiers, Bases de données relationnelles ou XML, Annuaires d'entreprise, …)

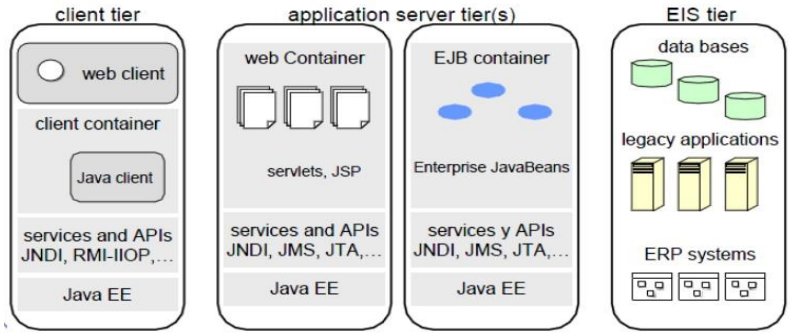


Figure : architecture JavaEE.

- **Apache Tomcat :**

Apache Tomcat ou simplement Tomcat est un serveur d'applications, plus précisément un conteneur web libre de servlets et JSP. Issu du projet Jakarta, c'est un des nombreux projets de l'Apache Software foundation. Il implémente les spécifications des servlets et des JSP du Java Community Process, est paramétrable par des fichiers XML et des propriétés, et inclut des outils pour la configuration et la gestion. Il comporte également un serveur HTTP.

Tomcat a été écrit en langage Java. Il peut donc s'exécuter via la machine virtuelle Java sur n'importe quel système d'exploitation la supportant.

Figure : Logo du serveur Apache Tomcat.

- **Eclipse**

Eclipse IDE est un environnement de développement intégré libre, extensible et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Il est principalement écrit en java.

Figure : Logo du logiciel eclipse.

- **MySQL**

C'est un système de gestion de bases de données relationnelles, son rôle est d'enregistrer des données de manière organisée afin d'aider à les retrouver facilement.

Figure : Logo du système MySQL

- **WampServer**

WampServer est une plate-forme de développement Web basée sur Windows pour les applications Web de type Wamp (Windows Apache MySQL, PHP), permettant d'utiliser localement le serveur Apache2, le langage de script PHP et la base de données MySQL. Il existe également PHPMyAdmin pour une gestion plus facile de la base de données.

Figure : Logo de la plateforme WampServer

- **Git**

Git est le système de contrôle de version le plus utilisé aujourd'hui. Il permet d'enregistrer des versions de projets et de basculer entre eux et de manipuler le code de projet avec sécurité et flexibilité.

Figure : Logo du système Git

- **GitHub**

GitHub est une plateforme open source de contrôle de version qui permet aux développeurs de stocker et de partager leur code publiquement ou en privé. Le site est aussi un espace de collaboration. Tous les utilisateurs peuvent contribuer aux projets publiés sur GitHub en suggérant des modifications. Une partie du succès de GitHub réside dans la façon dont le site a facilité ce processus. Pour éviter que les utilisateurs ne se dérangent en modifiant le programme en même temps, chacun télécharge le code sur son propre ordinateur, apporte des modifications et le publie sur GitHub après vérification.

Ce site est basé sur Git. Des espaces de discussion mis à la disposition de tous les développeurs pour discuter de leurs programmes et contributions.

Figure : Logo du service GitHub

CONCLUSION GENERALE:

RESUME: