GROUPE 6 UTA

IGL.L2

ALI DABO &

KABLAN MARIE

Sous la direction du

DR JOHNSON.

Table of Content

Introduction

Dans le cadre de notre recherche, nous avons entrepris une analyse des modèles de processus de développement logiciel couramment utilisés. Cette analyse vise à comprendre les avantages et les inconvénients de chaque modèle afin de déterminer celui le plus approprié pour le développement d'une application de gestion de bibliothèque. Cette application sera conçue pour répondre aux besoins de gestion des utilisateurs, des livres, du stock, des emprunts, tout en offrant un tableau de bord pour visualiser les statistiques pertinentes.

Les Modeles de Procecus de developpement

Avant de détailler le modèle choisi pour notre projet , une analyse comparative des différents modèles de processus de développement en UML sera effectuée afin de choisir le plus adapté à notre projet.

Modèle en Transformation Automatique

Description

Le modèle en Transformation Automatique consiste à générer automatiquement du code à partir de spécifications détaillées. Il repose sur des outils spécialisés qui traduisent les spécifications en code, éliminant ainsi en grande partie l'intervention humaine dans le processus de développement.

Cas d'utilisation

Projets avec des spécifications très détaillées : Ce modèle est particulièrement adapté aux projets où les spécifications sont claires et bien définies, car il peut générer du code en fonction de ces spécifications.

Projets où la génération de code est fiable et efficace : Lorsque la génération de code à

partir de spécification est fiable, rapide et précise, ce modèle peut être efficacement utilisé.

Projets où la rapidité de développement est une priorité : Pour les projets nécessitant une mise en œuvre rapide, le modèle en Transformation Automatique peut être avantageux, car il permet de réduire le temps de développement en générant rapidement du code.

Avantages

Rapidité de développement : En automatisant la génération de code, ce modèle permet un développement plus rapide par rapport à la saisie manuelle.

Réduction des erreurs humaines : En éliminant ou en réduisant l'intervention humaine dans la traduction des spécifications en code, les erreurs humaines sont réduites.

Utilisation efficace des spécifications détaillées : Ce modèle tire parti des spécifications détaillées en les utilisant directement pour générer le code, assurant ainsi une correspondance étroite entre les spécifications et le produit final.

Inconvénients

Dépendance aux outils de génération de code : Ce modèle est fortement dépendant

des outils de génération de code. Si ces outils ne sont pas fiables ou ne répondent pas aux besoins spécifiques du projet, cela peut poser des problèmes.

Moins de flexibilité pour les modifications en cours de route : Étant donné que le code est généré à partir des spécifications, apporter des modifications ou des ajustements en cours de projet peut être plus difficile et moins flexible.

Difficulté à gérer les cas d'utilisation non prévus dans les spécifications initiales : Si de nouveaux cas d'utilisation ou des exigences non spécifiées initialement émergent, il peut être compliqué de les

intégrer dans le code généré sans modifications importantes.

Modèle en Prototypage

Description

Le modèle en Prototypage implique la création rapide de prototypes fonctionnels pour valider les concepts et les besoins des utilisateurs avant de développer la version finale de l'application. Ces prototypes sont des versions simplifiées de l'application qui permettent aux utilisateurs de visualiser et d'interagir avec les fonctionnalités clés.

Cas d'utilisation

Projets avec des exigences floues ou changeantes : Le modèle en Prototypage est particulièrement efficace pour les projets où les exigences sont en évolution ou mal définies au départ.

Projets où la validation des fonctionnalités est cruciale : Il est adapté aux projets où il est important de valider rapidement et efficacement les fonctionnalités avec les utilisateurs.

Projets où les utilisateurs finaux peuvent fournir un feedback rapide : Ce modèle est idéal pour les projets où la participation active des utilisateurs est possible pour

fournir des retours précieux sur les prototypes.

Avantages

Validation précoce des concepts : Les prototypes permettent de valider rapidement les idées et les fonctionnalités clés, réduisant ainsi le risque de développement dans la mauvaise direction.

Implémentation rapide des idées : Il permet une mise en œuvre rapide des idées en transformant rapidement les concepts en prototypes fonctionnels.

Possibilité d'ajuster les fonctionnalités en fonction des retours utilisateurs : Les

retours des utilisateurs sur les prototypes permettent d'apporter des ajustements précis et efficaces aux fonctionnalités avant le développement complet.

Inconvénients

Risque de divergence entre le prototype et le produit final : Il peut y avoir des différences entre le prototype et le produit final, ce qui peut conduire à des attentes irréalistes chez les utilisateurs.

Peut nécessiter des ressources supplémentaires pour le développement du prototype : La création de prototypes fonctionnels peut nécessiter des ressources

supplémentaires en termes de temps et de compétences.

Peut conduire à des attentes irréalistes chez les utilisateurs : Si les utilisateurs se concentrent trop sur le prototype, ils peuvent avoir des attentes irréalistes quant à la version finale du produit.

Modèle en Incrémental

Description

Le modèle en Incrémental consiste à diviser le projet en modules fonctionnels indépendants, qui sont développés et livrés un par un. Chaque module représente une

fonctionnalité ou une partie du système global, et est développé séparément avant d'être intégré au reste du projet.

Cas d'utilisation

Projets où des fonctionnalités peuvent être livrées progressivement : Ce modèle est idéal pour les projets où il est important de livrer des fonctionnalités au fur et à mesure qu'elles sont développées, plutôt que d'attendre la fin du projet pour tout livrer en une fois.

Projets avec des exigences qui évoluent au fil du temps : L'Incrémental permet une adaptation aux changements des exigences,

car les fonctionnalités peuvent être ajoutées ou modifiées à chaque itération.

Projets où chaque module peut être développé et testé séparément : La séparation des modules permet de les développer et de les tester indépendamment, assurant ainsi une meilleure qualité du produit final.

Avantages

Livraison progressive des fonctionnalités : Les utilisateurs peuvent commencer à utiliser l'application avec des fonctionnalités de base dès les premières étapes du développement.

Adaptabilité aux changements : Les modifications peuvent être apportées à chaque module ou itération, permettant au projet de s'adapter aux changements des exigences ou des besoins des utilisateurs.

T**ests par module pour une meilleure qualité** : Chaque module est développé et testé individuellement, ce qui garantit une meilleure qualité du code et une détection précoce des erreurs.

Inconvénients

Possibilité de difficultés d'intégration entre les modules : L'intégration de différents modules peut parfois poser des

défis, en particulier si les interfaces entre les modules ne sont pas bien définies.

Coût potentiellement plus élevé en raison des itérations successives : Le développement incrémental peut nécessiter plus de temps et de ressources en raison des itérations successives pour chaque module.

Risque de retard dans les livraisons si un module rencontre des problèmes : Si un module rencontre des problèmes ou des retards, cela peut affecter les livraisons ultérieures et le calendrier global du projet.

Modèle en Code and Fix

Description

Le modèle en Code and Fix est un processus non structuré où le développement consiste à écrire du code sans spécifications claires, suivi de corrections au fur et à mesure que des problèmes sont identifiés. Il s'agit souvent d'une approche rapide pour obtenir un produit fonctionnel sans suivre un processus de développement formel.

Cas d'utilisation

Petits projets ou projets prototypes : Ce modèle est souvent utilisé pour de petits projets ou des prototypes où les exigences sont simples et peu susceptibles de changer.

Projets où les exigences sont simples et peu définies : Il est adapté aux projets où les exigences sont floues ou non spécifiées au départ.

Projets où le temps et les ressources sont limités : Le modèle en Code and Fix peut être utilisé lorsque les délais sont serrés et qu'il faut obtenir rapidement un produit fonctionnel.

Avantages

Rapidité initiale de développement : Étant donné qu'il n'y a pas de phase de conception détaillée, le développement peut commencer rapidement.

Flexibilité pour apporter des changements à tout moment : Les modifications peuvent être apportées au code à tout moment, sans nécessiter une planification ou une documentation détaillée.

Adapté aux projets expérimentaux ou de preuve de concept : Ce modèle est approprié pour tester des idées rapidement sans investir beaucoup de temps ou de ressources.

Inconvénients

Risque élevé de bugs et de problèmes de qualité : Étant donné le manque de planification et de tests approfondis, le

produit final peut contenir de nombreux bugs et problèmes de qualité.

Manque de structure peut entraîner des difficultés de maintenance : Le code développé selon ce modèle peut être difficile à comprendre et à maintenir en raison du manque de structure et de documentation.

Difficulté à estimer le temps et les coûts de développement : Sans processus formel, il peut être difficile d'estimer correctement le temps et les coûts nécessaires pour compléter le projet.

Modèle en Cascade (Waterfall)

Description

Le modèle en Cascade, également connu sous le nom de modèle en Waterfall, est un processus de développement logiciel linéaire et séquentiel. Il est constitué de plusieurs phases distinctes, où chaque phase doit être complétée avant de passer à la suivante. Les phases typiques incluent la définition des besoins, la conception, le développement, les tests, le déploiement et la maintenance.

Cas d'utilisation

Projets avec des exigences bien définies et stables : Ce modèle convient aux projets où les exigences sont clairement spécifiées dès

le début et peu susceptibles de changer tout au long du projet.

Projets où les changements sont peu probables : Il est adapté aux projets où les changements des exigences sont rares ou peu probables une fois que le processus de développement a commencé.

Projets où la planification est essentielle : Ce modèle est approprié pour les projets qui nécessitent une planification détaillée des tâches et des ressources avant le début du développement.

Avantages

Structure claire et linéaire : La séquence de phases bien définie offre une structure claire pour le développement du logiciel, ce qui facilite la planification et la gestion du projet.

Facilité de gestion et de suivi : Chaque phase est clairement délimitée, ce qui facilite la gestion et le suivi des progrès du projet.

Documentation approfondie à chaque étape : Chaque phase nécessite une documentation détaillée, ce qui permet une meilleure traçabilité des décisions prises et des fonctionnalités développées.

Inconvénients

Rigidité face aux changements : En raison de sa nature linéaire, le modèle en Cascade est moins adaptable aux changements des exigences en cours de projet.

Difficulté à détecter les erreurs tardivement : Les tests ne sont effectués qu'à la fin du processus, ce qui signifie que les erreurs peuvent ne pas être détectées avant cette phase, ce qui rend les corrections coûteuses et difficiles.

Livraison tardive des fonctionnalités aux utilisateurs : Les utilisateurs ne voient le produit qu'à la fin du développement, ce qui signifie qu'ils doivent attendre longtemps pour voir des résultats tangibles.

Modèle en V

Description

Le modèle en V est une extension du modèle en Cascade, mettant davantage l'accent sur les tests et la validation à chaque étape du développement. Il prend son nom de la forme de la lettre "V" représentant la séquence des phases de développement et de test qui se rejoignent à la fin du processus.

Cas d'utilisation

Projets nécessitant une forte validation et vérification : Le modèle en V est adapté aux projets où la qualité du produit final est

une priorité et nécessite une forte validation et vérification à chaque étape.

Projets où la qualité est une priorité : Il convient aux projets où la fiabilité, la robustesse et la qualité du logiciel sont essentielles.

Projets où les risques doivent être identifiés tôt : Ce modèle est approprié pour les projets où la gestion des risques est importante, car les tests sont effectués à chaque étape pour identifier les problèmes rapidement.

Avantages

Validation et vérification continues : Les tests sont effectués à chaque étape, ce qui

permet d'identifier et de corriger les problèmes rapidement.

Identification précoce des risques : Les tests précoces permettent d'identifier les risques potentiels dès le début du projet, ce qui permet une gestion proactive des risques.

Meilleure qualité du produit final : La validation et la vérification continues conduisent à un produit final de meilleure qualité avec moins de bugs et d'erreurs.

Inconvénients

Coût potentiellement plus élevé : En raison de la nécessité de tests à chaque étape, le

modèle en V peut être plus coûteux en termes de temps et de ressources.

Rigidité face aux changements : Comme le modèle en Cascade, le modèle en V peut être moins adaptable aux changements des exigences en cours de projet.

Possibilité de retard dans les livraisons: En raison des tests à chaque étape, il peut y avoir des retards dans les livraisons des fonctionnalités aux utilisateurs.

Modèle en Spirale

Description

Le modèle en Spirale est un processus de développement itératif et cyclique, qui combine les aspects de la gestion des risques avec les phases de développement en spirale. Il vise à gérer de manière proactive les risques tout au long du projet, en identifiant, évaluant et réduisant les risques à chaque itération.

Cas d'utilisation

Projets avec des aspects critiques ou incertains : Ce modèle est adapté aux projets où certains aspects du système sont critiques ou incertains et nécessitent une évaluation continue.

Projets où les risques doivent être gérés de manière proactive : Il convient aux projets où la gestion des risques est cruciale pour minimiser les impacts négatifs.

Projets où une version initiale est nécessaire rapidement : Ce modèle permet le développement d'une version initiale rapidement, avec des itérations pour ajouter des fonctionnalités et gérer les risques au fil du temps.

Avantages

Gestion proactive des risques : La méthode en Spirale met l'accent sur l'identification et la gestion précoce des

risques, ce qui permet de réduire les surprises et les impacts négatifs.

Flexibilité pour les changements : Avec ses itérations, le modèle en Spirale offre une flexibilité pour apporter des changements et des ajustements au projet au fur et à mesure que de nouvelles informations sont découvertes.

Développement itératif et progressif : Chaque itération ajoute des fonctionnalités au produit, permettant aux utilisateurs de voir une évolution progressive du système

Inconvénients

Complexité et coût potentiellement plus élevé : La gestion des risques et les itérations peuvent ajouter de la complexité et des coûts au projet.

Nécessite une expertise en gestion des risques : Pour mettre en œuvre efficacement le modèle en Spirale, une expertise en gestion des risques est nécessaire.

Peut être difficile à planifier et à estimer : En raison de sa nature itérative, il peut être difficile de planifier et d'estimer correctement les délais et les coûts du projet.

Modèle en Vagues (Wave model)

Description

Le modèle en Vagues, également connu sous le nom de modèle wave, combine des aspects du modèle en V et des méthodes agiles. Il vise à offrir une certaine flexibilité tout en gardant une structure de développement claire. Le processus se déroule en vagues successives, où chaque vague représente un cycle de développement itératif.

Cas d'utilisation

Projets où la flexibilité est nécessaire mais une structure est également souhaitée : Ce modèle est adapté aux projets où il est important d'avoir une certaine flexibilité pour s'adapter aux changements, tout en gardant une structure claire pour le développement

Projets avec des parties du système plus stables que d'autres : Il convient aux projets où certaines parties du système sont bien définies et stables, tandis que d'autres parties sont plus sujettes aux changements.

Avantages

Flexibilité et adaptation aux changements : Le modèle en Vagues offre une certaine

flexibilité pour s'adapter aux changements des exigences ou des besoins des utilisateurs entre les vagues.

Structure de développement claire : Malgré sa flexibilité, le modèle en Vagues maintient une structure de développement claire avec des cycles itératifs définis.

Amélioration continue : Chaque vague permet une amélioration continue du produit, en ajoutant de nouvelles fonctionnalités ou en ajustant les existantes en fonction des retours des utilisateurs.

Inconvénients

Complexité de gestion : En raison de la combinaison de la flexibilité et de la structure, le modèle en Vagues peut être plus complexe à gérer que d'autres modèles.

Risque de dérive par rapport aux objectifs initiaux : La flexibilité peut entraîner une dérive par rapport aux objectifs initiaux du projet si les changements ne sont pas gérés de manière adéquate.

Besoin de compétences et de ressources appropriées : La mise en œuvre réussie du modèle en Vagues nécessite des compétences en gestion de projet et des ressources adéquates pour suivre les cycles itératifs.

Après une recherche approfondie, nous avons identifié sept modèles de processus de développement, la Transformation Automatique, le Prototypage, l'Incrémental, le Code-and-Fix, le Cascade (Waterfall), le V, le Spiral et les Vagues. Chaque modèle possède des caractéristiques uniques adaptées à des cas d'utilisation spécifiques.

Analyse Comparative

1.**Adaptabilité aux changements des exigences :** Le modèle Incrémental, ainsi que d'autres modèles itératifs tels que le Prototypage et la Spirale, sont plus flexibles

pour s'adapter aux changements, tandis que les modèles linéaires comme la Cascade sont moins adaptables.

2.**Rapidité de développement :** Les modèles itératifs comme le Prototypage, l'Incrémental et la Spirale permettent une livraison progressive de fonctionnalités, ce qui accélère le développement par rapport aux modèles linéaires comme la Cascade.

3.**Qualité du produit final** : Les modèles qui intègrent des tests continus à chaque étape, tels que l'Incrémental, la V et la Spirale, tendent à produire des produits finaux de meilleure qualité grâce à une détection précoce des erreurs.

4.**Gestion des risques**: Les modèles comme la Spirale et le modèle en Vagues, qui intègrent une gestion proactive des risques tout au long du processus, sont mieux équipés pour identifier, évaluer et atténuer les risques émergents.

5.**Complexité et coûts**: Les modèles itératifs peuvent avoir des coûts et une complexité plus élevés en raison de la gestion des itérations, tandis que les modèles linéaires peuvent être moins complexes mais risquent de nécessiter plus de ressources en cas de changements tardifs.

6.**Niveau de documentation:** Les modèles formels comme la Cascade et le V nécessitent généralement un niveau élevé de documentation à chaque étape, tandis que

les modèles plus agiles comme le Prototypage et l'Incrémental peuvent nécessiter moins de documentation pour permettre une flexibilité et un développement rapides.

Après avoir effectué une analyse comparative de chaque modèle, nous avons sélectionné le modèle Incrémental comme le plus approprié pour le développement de notre application de gestion de bibliothèque. Ce choix s'appuie sur plusieurs facteurs, notamment la nature itérative du modèle Incrémental, qui permet une flexibilité et une adaptation aux besoins changeants du projet.

Justification du choix de modèle en incrémental

Le modèle en Incrémental consiste à diviser le projet en modules fonctionnels indépendants, développés et livrés progressivement. Ses principaux cas d'utilisation sont les projets où des fonctionnalités peuvent être livrées de manière évolutive, où les exigences évoluent au fil du temps, et où chaque module peut être développé et testé séparément.

1.**Livraison progressive :** Avec la complexité inhérente à une application de

gestion de bibliothèque, la livraison progressive des fonctionnalités par étapes permet de démarrer rapidement tout en garantissant une qualité constante.

2.**Adaptabilité aux changements** : Les besoins d'une bibliothèque peuvent évoluer, nécessitant des ajustements dans l'application. Le modèle en Incrémental permet d'ajouter de nouvelles fonctionnalités ou de modifier les existantes à chaque itération, en réponse aux retours des utilisateurs ou aux nouveaux besoins identifiés.

3.**Tests et validation par module** : Chaque module étant développé et testé de manière

indépendante, cela garantit que chaque fonctionnalité répond aux besoins spécifiques de la bibliothèque et fonctionne de manière optimale.

4.**Réduction des risques :** En livrant par modules, les risques sont mieux maîtrisés. Les problèmes rencontrés dans un module n'affectent pas nécessairement les autres parties de l'application, ce qui permet une gestion plus efficace des risques.

5.**Implémentation progressive** : Les utilisateurs peuvent commencer à utiliser l'application plus rapidement, avec les fonctionnalités de base disponibles dès les premières étapes de développement, ce qui favorise l'adoption et la satisfaction des utilisateurs.

Model incremental

1.Planification et Définition des Besoins:

- Description: Cette étape implique la définition des besoins du système en concertation avec les parties prenantes. On établit les objectifs, les fonctionnalités principales et les contraintes.

- Diagramme UML: Diagramme de Cas d'Utilisation pour identifier les interactions entre les acteurs et le système, ainsi que les principales fonctionnalités.

2.Analyse et Conception du Système:

- Description: Cette étape consiste à détailler les besoins en fonctionnalités identifiées lors de la phase de planification. On conçoit l'architecture globale du système.

- Diagramme UML: Diagramme de Séquence pour représenter les interactions entre les différents composants du système.

3.Développement du Premier Module:

- Description: Dans cette étape, le premier module fonctionnel est développé. Il s'agit généralement d'une version de base du système avec des fonctionnalités essentielles.

- Diagramme UML: Diagramme de Classes pour représenter la structure statique du système avec ses classes et relations.

4.Tests du Premier Module:

- Description: Le premier module est testé pour s'assurer qu'il fonctionne correctement selon les spécifications définies.

- Diagramme UML: Diagramme d'Activité pour modéliser les différents processus et flux d'activités lors des tests.

5.Intégration du Premier Module:

- Description: Une fois le premier module testé et validé, il est intégré au système existant. Les interactions entre les modules sont vérifiées et testées.

- Diagramme UML: Diagramme de Déploiement pour décrire la configuration matérielle du système et comment les composants logiciels sont déployés.

6.Développement des Modules Suivants :

- Description: Les étapes 3 à 5 sont répétées pour chaque nouveau module à ajouter. Chaque module apporte de nouvelles fonctionnalités au système.

- Diagramme UML: Selon les besoins spécifiques de chaque module, différents types de diagrammes UML peuvent être utilisés, tels que les diagrammes de Séquence, de Classes, d'État, etc.

7.Tests et Intégration des Modules Suivants:

- Description: Les nouveaux modules développés sont testés et intégrés au système existant. Les interactions entre tous les modules sont vérifiées.

- Diagramme UML: Diagrammes spécifiques pour chaque module, selon les besoins de tests et d'intégration.

En somme le modèle en Incrémental se distingue comme le choix le plus adapté pour le développement de notre application de gestion de bibliothèque, grâce à sa capacité de livraison progressive, son adaptabilité aux changements, ses tests par module et sa réduction des risques. En optant pour ce modèle, nous pouvons garantir le développement d'une application robuste, flexible et répondant parfaitement aux besoins changeants des utilisateurs d'une bibliothèque.

Cahier des Charges - Application de Gestion de Bibliothèques

Le cahier des charges de notre projet détaille les objectifs, les besoins fonctionnels et non fonctionnels, ainsi que les contraintes techniques et temporelles. Il fournit également une vue d'ensemble du déroulement du projet, y compris la planification, la documentation et les responsabilités.

Présentation du Projet

Contexte

Le projet vise à développer une application de gestion d’une bibliothèque moderne afin de faciliter la gestion des livres, des utilisateurs, des emprunts, et du stock. Cette application permettra une gestion efficace et conviviale pour les administrateurs et les adhérents de la bibliothèque.

Objectifs

- Automatiser la gestion des livres et des utilisateurs pour plus d'efficacité.
- Offrir une plateforme conviviale pour les adhérents et les administrateurs.
- Assurer la disponibilité et la gestion efficace du stock.

- Implémenter un système d'emprunt avec alertes pour les retours en retard.

Description de l’Existant

Actuellement, la gestion de la bibliothèque est manuelle, ce qui entraîne des inefficacités, des retards dans les emprunts et des difficultés de suivi du stock.

Critères d’Acceptabilité du Produit

- Interface utilisateur intuitive et conviviale.
- Fonctionnalités complètes pour la gestion des livres, des utilisateurs, des emprunts et du stock.

- Notifications et alertes efficaces pour les adhérents et les administrateurs.
- Gestion sécurisée des données et de l'authentification.

Expression des Besoins

Besoins Fonctionnels

Gestion des Utilisateurs

- Ajout, Modification, Suppression d'un utilisateur.
- Profil Administrateur avec droits spécifiques.
- Authentification sécurisée.

Gestion des Livres

- Ajout, Modification, Suppression d'un livre.
- Information sur le titre, l'auteur, la date de parution, la page de couverture, la collection, et un code unique.
- Vérification de la disponibilité et de l'approvisionnement.
- Limite d'emprunt (max 2 livres par adhérent).
- Alerte par e-mail pour les retards de retour.

Gestion des Emprunts

- Enregistrement, Retour des livres.
- Contrôle des délais (maximum 2 semaines).

- Pénalité pour les retards.
- Restriction : pas d'emprunt supplémentaire si un livre est déjà emprunté.

Tableau de Bord Statistique

- Statistiques sur les livres empruntés, disponibles, les utilisateurs ayant emprunté et n'ayant pas encore rendu les livres.
- Diagrammes circulaires pour visualiser les statistiques.

Besoins Non Fonctionnels

Sécurité

- Authentification sécurisée pour les utilisateurs et les administrateurs.
- Protection des données personnelles.

Performances

- Temps de réponse rapide pour une expérience utilisateur fluide.
- Capabilité pour une gestion efficace du stock et des utilisateurs.

Convivialité

- Interface utilisateur intuitive et facile à utiliser.
- Notifications claires et précises pour les utilisateurs.

Contraintes

Coûts

Budget alloué pour le développement et la maintenance de l'application.

Délais

Date de livraison fixée au 24 mai 2024.

Contraintes Techniques

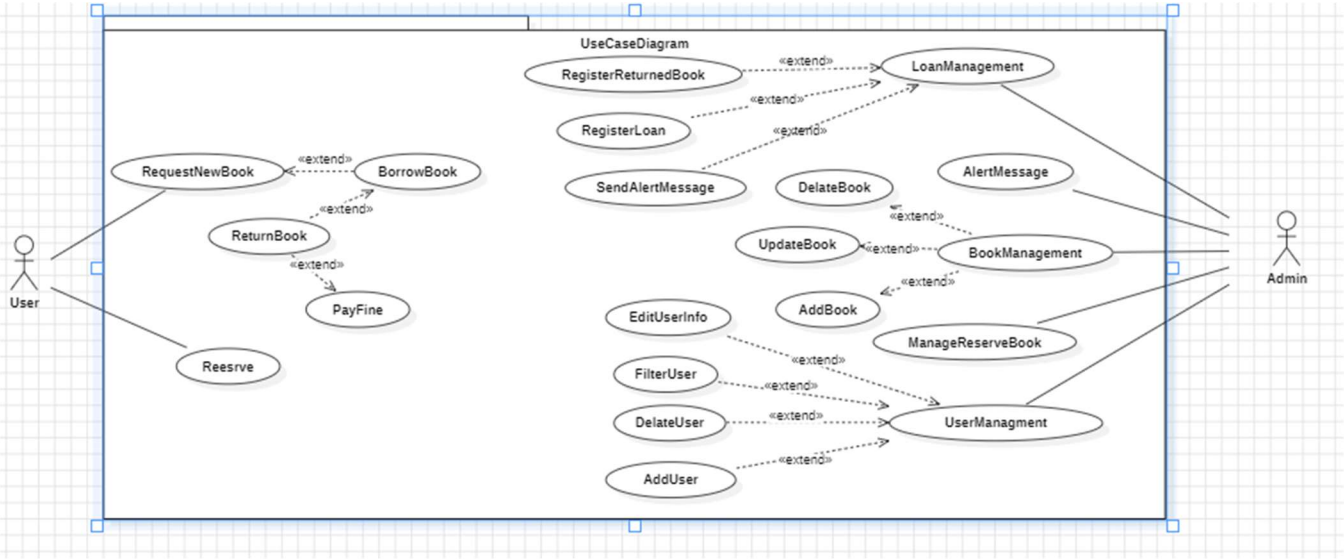
Langage de programmation : Java.

Base de données relationnelle : MySQL

Modèle de développement : Incrémental.

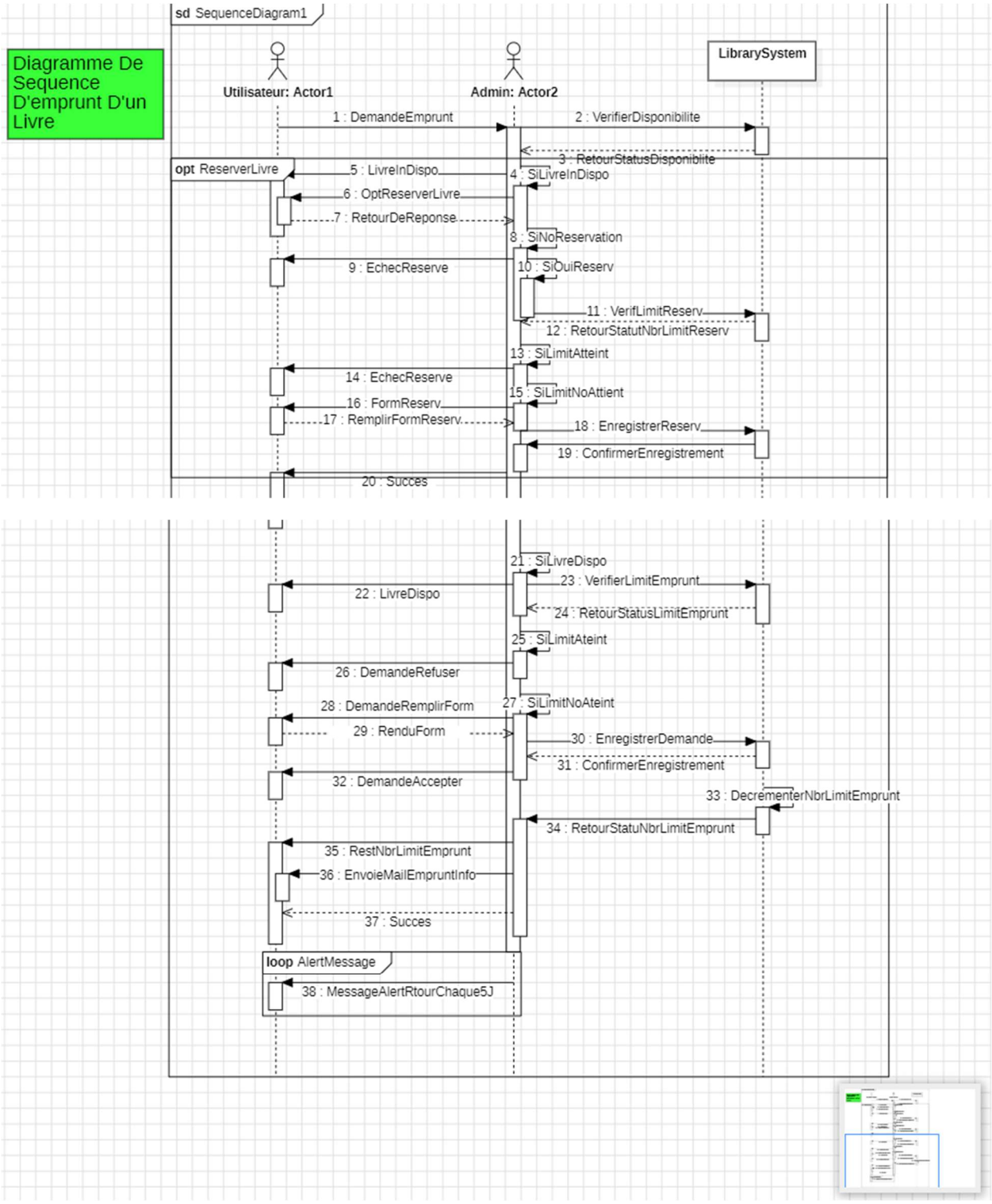
Diagrammes Uml intervenant a chaque Etape du Model Incremental

Planification et Définition des Besoins : **Diagramme de Cas d'Utilisation** pour identifier les interactions entre les acteurs et le système, ainsi que les principales fonctionnalités

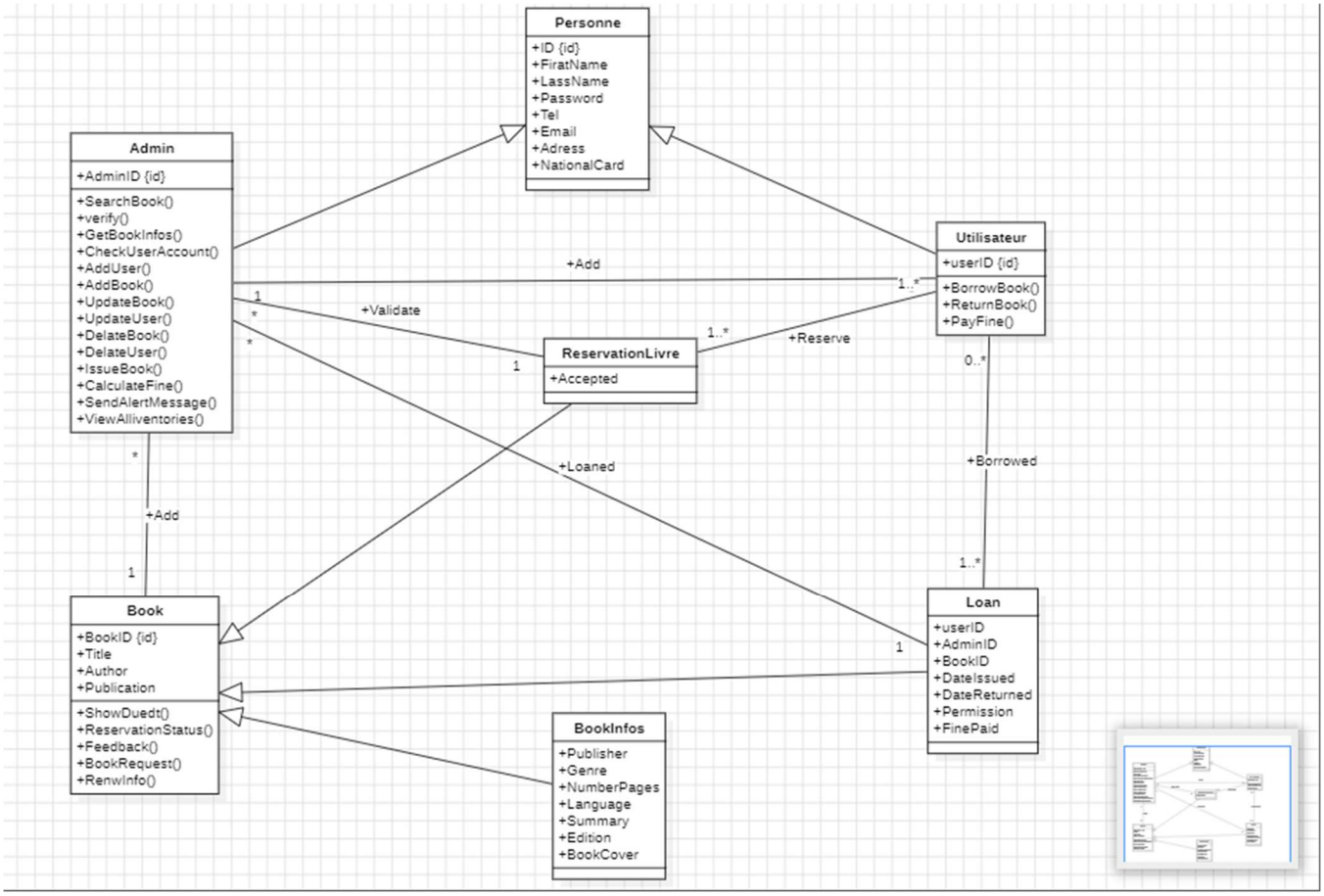


Analyse et Conception du Système : **Diagramme de Séquence** pour représenter les interactions entre les différents composants du système.

Diagramme De Sequence D'emprunt D'un Livre

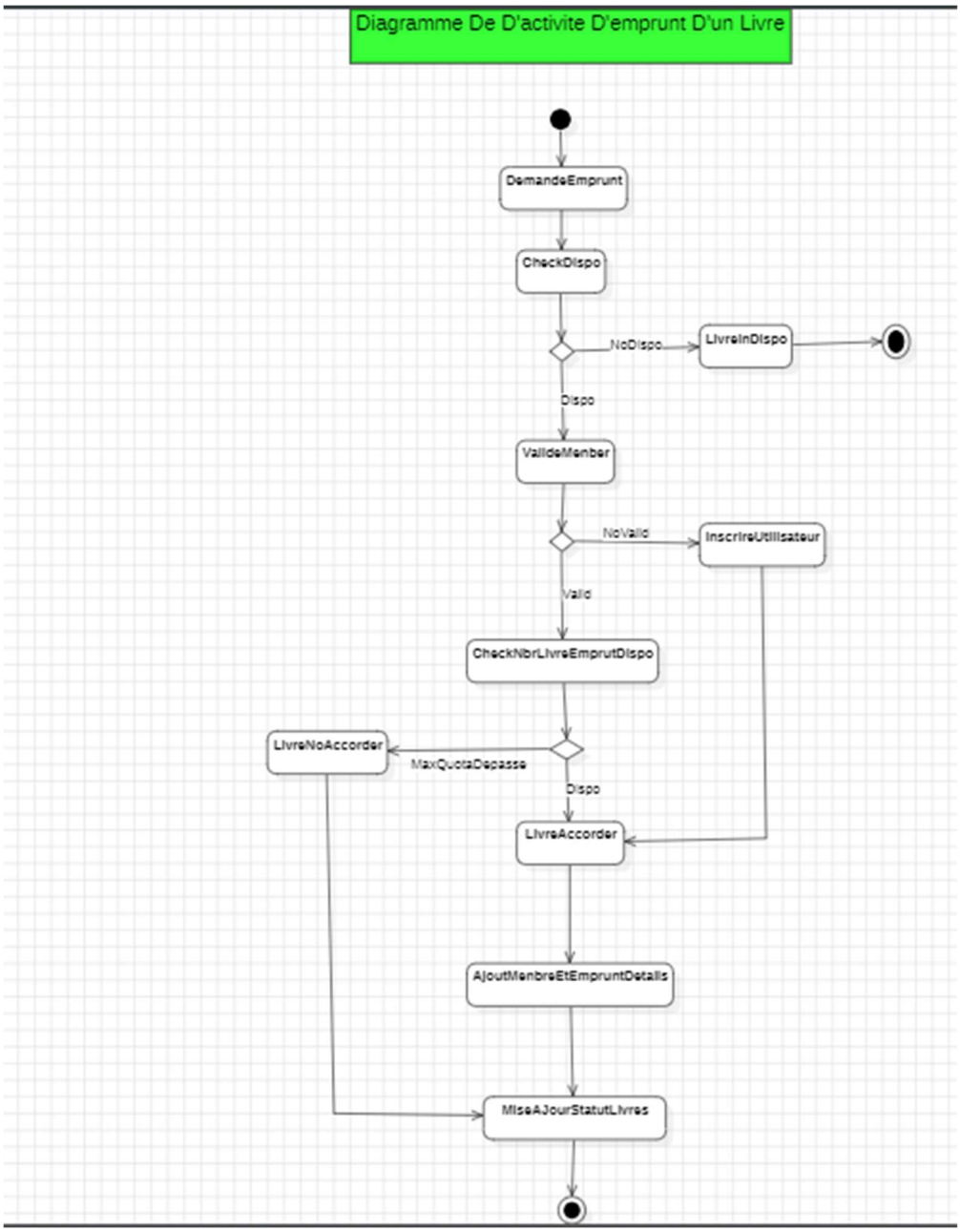


Développement du Premier Module : **Diagramme de Classes** pour représenter la structure statique du système avec ses classes et relations.

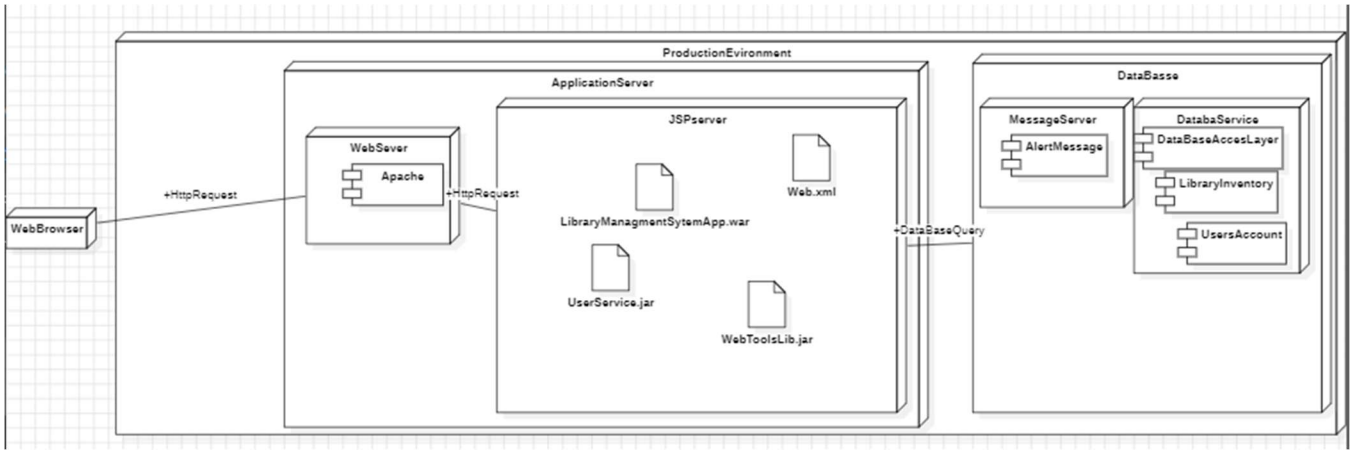


Tests du Premier Module: **Diagramme d'Activité** pour modéliser les différents processus et flux d'activités lors des tests.

Diagramme d’activite d’emprunt d’un Livre



Intégration du Premier Module**:**
Diagramme de Déploiement pour décrire la configuration matérielle du système et comment les composants logiciels sont déployés.



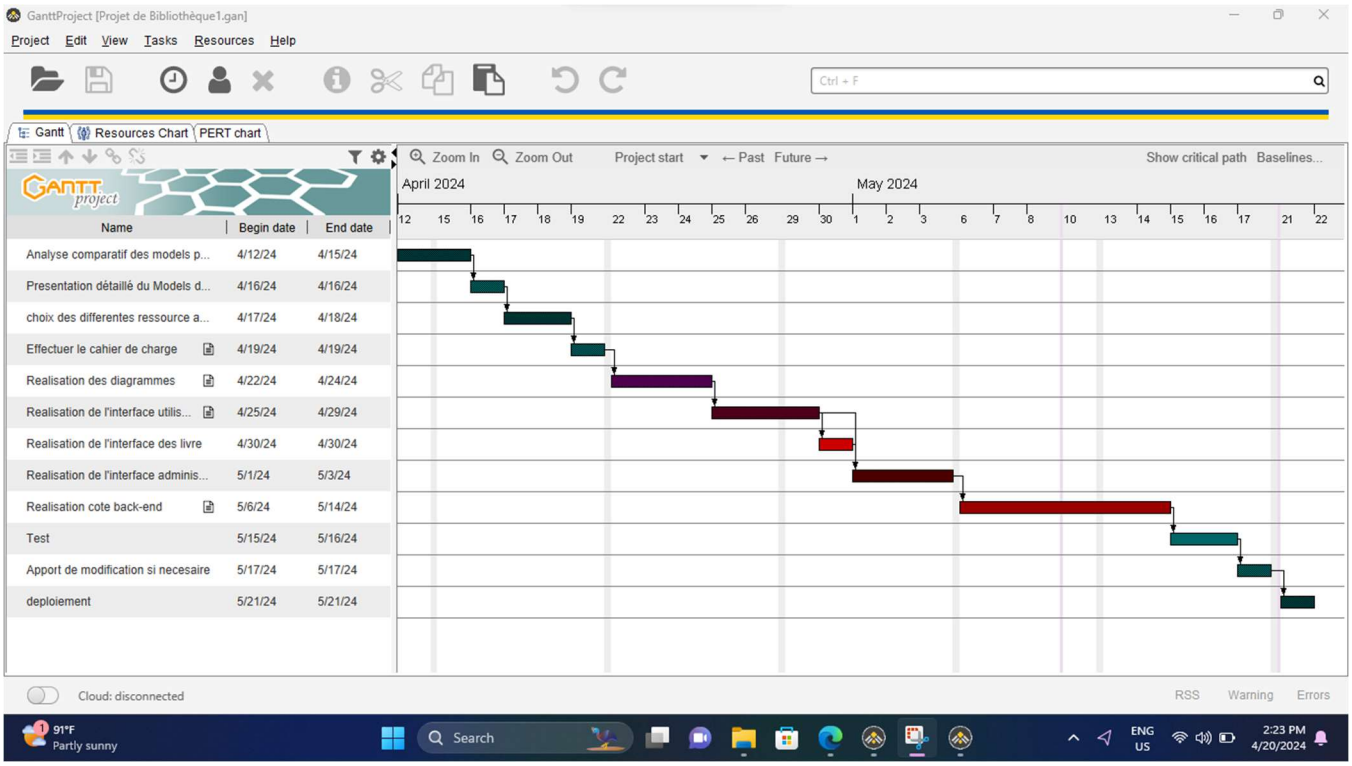
Déroulement du Projet

Planification

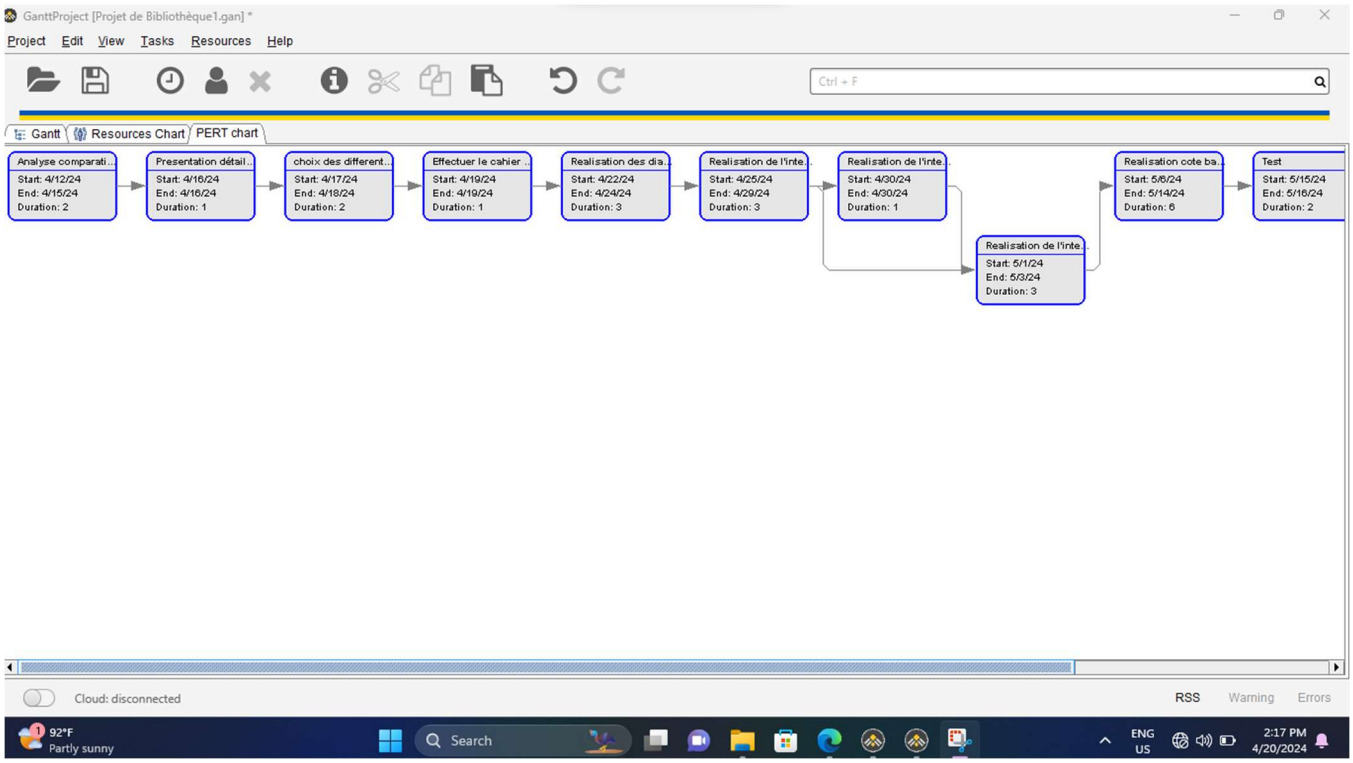
- Identification des étapes de développement selon le modèle Incrémental.

- Répartition des tâches et estimation des délais.

- La planification originale suit le diagramme de Gantt suivant :



- Diagramme de PERT :



Plan d'Assurance Qualité

- Tests unitaires, Tests d'intégration, Tests de validation.

- Revues de code régulières pour garantir la qualité du code.

Documentation

- Documentation technique détaillée.

- Manuel d'utilisation pour les adhérents et les administrateurs.

Responsabilités

- Maîtrise d’Ouvrage : Validation des besoins et du produit final.
- Maîtrise d’Œuvre : Développement et mise en œuvre de l'application.

Spécifications Techniques

- Architecture : Architecture 3 tiers

(Présentation, Logique Métier, Données). Dans le contexte d'un projet de bibliothèque, où la gestion des données (livres, membres, emprunts, etc.) et la logique métier (gestion des prêts, réservations, notifications, etc.) sont cruciales, l'architecture MVC offre une approche structurée et flexible pour

développer une application robuste et évolutive. Elle permet de répondre efficacement aux besoins de l'application tout en facilitant sa maintenance et son évolution à long terme.

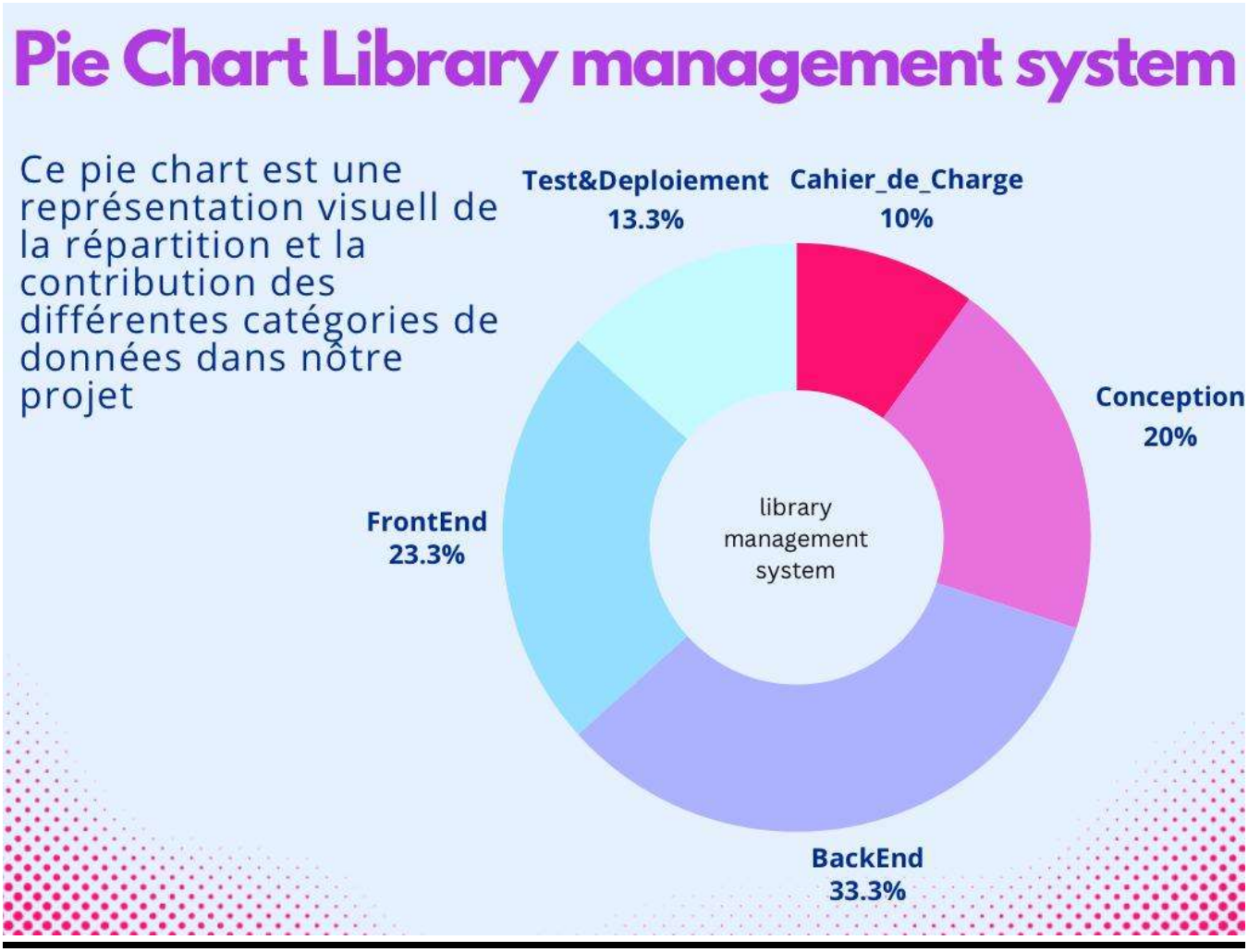
- IDE : Utilisation Netbeans IDE for Java Developers pour le développement.
- Framework graphique : Utilisation de Java Swing pour l'interface graphique.
- Base de données : Utilisation de MySQL comme base de données relationnelle.
- Gestion de version : Utilisation de Git pour le visionnage du code.
- Gestion de Projet : Utilisation de GanttProject comme outil de gestion de projet.

- Canvas Pour la representation du diagramme circulaire

Livrables en Génie Logiciel

1. Code Source de l'Application.
2. Documentation Technique.
3. Manuel d'Utilisation.
4. Base de Données.
5. Rapport d'Assurance Qualité.

Diagramme Circulaire (Pie Chart) du Projet



Depot Git

Retrouver ce projet complet sur notre depot Git en cliquant sur ce lien https://github.com/alsondab/Bibliot_G6.git

Ce projet est basé sur le modèle de développement incrémental, avec une répartition des tâches et une estimation des délais. Un plan d'assurance qualité est établi, comprenant des tests unitaires, d'intégration et de validation, ainsi que des revues de code régulières. Les livrables incluent le code source de l'application, la documentation technique, le manuel d'utilisation, la base de données et un rapport d'assurance qualité.

En conclusion, notre recherche et notre analyse nous ont permis de choisir le modèle

Incrémental comme le plus adapté pour le développement de notre application de gestion de bibliothèque. Nous sommes convaincus que ce choix nous permettra d'atteindre nos objectifs de manière efficace et de fournir une solution de haute qualité à nos utilisateurs.

Références

“Dr JOHNSON”, Exemple de cahier de charges.pdf, 2024.

Blackbox, AI

Pressman, Roger S. “Software engineering a practitioner's approach.” McGraw-Hill

Education, 2014.
Larman, Craig. “Agile and iterative development: a manager's guide.” Addison-Wesley Professional, 2004.
Brooks, Frederick P. “No silver bullet—essence and accidents of software engineering.” Computer 20.4 (1987): 10-19.