Une image contenant Police, Graphique, logo, graphisme

Description générée automatiquementLaCite collégiale

Le collège d'arts appliqués et de technologie

|  |  |
| --- | --- |
| Keven **Norton**  Jayson **Lefebvre**  Arthur **Kameni** | **Catalogue numérique de bibliothèque**  Une bibliothèque universitaire envisage de mettre en place un catalogue numérique facilitant la gestion et la consultation des livres disponibles. Cette application doit permettre une gestion simplifiée des ouvrages, un accès rapide aux informations, ainsi que des fonctionnalités d'emprunt et de retour des livres. |

Livrable 1

Une image contenant capture d’écran, cercle, obscurité, noir

Description générée automatiquement

Table des matières

[I. Introduction 3](#_Toc188913701)

[II. Objectifs 4](#_Toc188913702)

[III. Périmètre du projet 6](#_Toc188913703)

[IV. Spécifications Fonctionnelles et Techniques 8](#_Toc188913704)

[V. Méthodologie et Approche 12](#_Toc188913705)

[VI. Modélisation UML et Conception 14](#_Toc188913706)

[VII. Design d’Interfaces 33](#_Toc188913707)

[VIII. Planification et Gestion de Projet (Jira) 33](#_Toc188913708)

[IX. Validation et Maintenance 35](#_Toc188913709)

[A. Critères de validation coté équipe de développement 36](#_Toc188913710)

[B. Critères de validation coté client 37](#_Toc188913711)

[C. Modalités de Suivi et de Maintenance 37](#_Toc188913712)

[X. Création de compte Github 38](#_Toc188913713)

[XI. Annexes 38](#_Toc188913714)

# Introduction

**Contexte et Objectifs du Projet**

Les bibliothèques universitaires jouent un rôle essentiel dans l’accès à l’information et au savoir. Cependant, la gestion des collections de livres et documents reste souvent un défi logistique. Face à ce besoin, une bibliothèque universitaire souhaite mettre en place un catalogue numérique interactif qui simplifie la gestion des ouvrages, optimise les recherches et facilite les emprunts et retours.

L’objectif principal de ce projet est de concevoir une application numérique intuitive et efficace qui modernise la gestion des ressources de la bibliothèque, tout en améliorant l’expérience utilisateur pour les étudiants, les enseignants et le personnel. Pour assurer la réussite de ce projet, une architecture technique robuste a été définie, reposant sur des technologies modernes et éprouvées.

**Description Générale du Produit**

L’application se présente comme une solution numérique complète pour :

* **Enregistrer et importer des livres** avec génération automatique de codes ISBN ou QR.
* **Effectuer un inventaire en temps réel**, permettant un suivi précis des ouvrages disponibles.
* **Rechercher facilement des documents** à l’aide d’une interface intuitive ou via des outils tels que le scan de codes QR.
* **Gérer les emprunts et retours**, avec un suivi des délais et des alertes automatisées.
* **Fournir des statistiques et rapports** pour une gestion proactive des collections.

L’application est dotée d’une interface utilisateur moderne et conviviale, renforcée par une fiabilité des données accrue et une grande flexibilité pour s’adapter aux besoins spécifiques de la bibliothèque universitaire.

**Équipe et Responsabilités**

Le projet sera développé par une équipe multidisciplinaire, assurant une collaboration optimale pour atteindre les objectifs :

* **Chargé de Projet : Arthur**  
  Responsable de la coordination globale du projet, de la gestion des délais et de la communication avec les parties prenantes.
* **Développeur : Arthur, Keven, Jayson**  
  En charge du développement de l’application, incluant le front-end, le back-end, et l’intégration des bases de données.
* **Designer UX/UI : Arthur**  
  Chargé de concevoir une interface utilisateur intuitive et attractive, garantissant une expérience fluide et agréable pour tous les utilisateurs.
* **Architecte Logiciel : Keven, Arthur, Jayson**  
  Responsable de la conception de l’architecture technique, du choix des technologies, et de la garantie de la scalabilité et de la sécurité du système.
* **Qualité et Assurance (QA) : Jayson Lefebvre**  
  En charge des tests de l’application pour garantir sa fiabilité, sa stabilité et son respect des exigences fonctionnelles et techniques.
* **Scrum Master : Keven Norton**  
  Responsable de l’application des principes Agile/Scrum, facilitant la collaboration au sein de l’équipe et assurant la livraison des itérations du projet.

# Objectifs

Le but de projet est de concevoir à créer une application (mobile, desktop) sur la gestion d’une bibliothèque universitaire.

|  |  |  |
| --- | --- | --- |
| **No** | **Objectif opérationnel** | **Résultat opérationnel attendu** |
| **1** | **Enregistrement et importation des livres et documents** | **Ajouter manuellement ou importer des listes de livres et documents, garantissant un catalogage rapide et précis.** |
| **2** | **Inventaire en temps réel** | **Fournir une vue d'ensemble actualisée des ouvrages disponibles, permettant un suivi précis des collections.** |
| **3** | **Gestion des emprunts et retours** | **Suivre les prêts et retours avec mise à jour automatique de la disponibilité des ouvrages pour éviter les erreurs.** |
| **4** | **Accessibilité et efficacité** | **Garantir une interface conviviale pour une navigation simple et un accès rapide aux informations nécessaires.** |
| **5** | **Fiabilité des données** | **Éviter les doublons grâce à la vérification des codes ISBN uniques et une gestion automatisée des données.** |
| **6** | **Flexibilité** | **Permettre d’ajouter des catégories ou d'adapter les inventaires aux besoins spécifiques de la bibliothèque.** |
| **7** | **Génération de codes ISBN** | **Chaque livre enregistré se voit attribuer un code ISBN unique, facilitant son identification et son accès.** |

# Périmètre du projet

#### Le projet vise à développer une **application numérique pour une bibliothèque universitaire** afin de :

#### Simplifier la gestion des livres et documents.

#### Offrir un accès rapide et intuitif aux informations.

#### Proposer des fonctionnalités d’emprunt et de retour optimisées.

**Tableau des tâches incluses et exclusions du projet**

|  |  |  |
| --- | --- | --- |
| **No** | **Tâches incluses dans le projet** | **Exclusions du projet** |
| **1** | **Enregistrement ou importation des livres et documents** | **Gestion avancée des utilisateurs** |
| **2** | **Génération de codes ISBN pour chaque livre enregistré** | **Module de formation du personnel** |
| **3** | **Inventaire des ouvrages disponibles en temps réel** | **Intégration de supports multimédias** |
| **4** | **Recherche d'un livre via le code ISBN** | **Système de réservation de salles** |
| **5** | **Gestion des emprunts et retours, suivi de disponibilité** | **Rapports et analyses avancées** |
| **6** | **Gestion des champs pour titre, auteur(s), catégorie, et statut** | **Fonctionnalités de facturation et de paiement** |
| **7** | **Possibilité d'ajouter plusieurs auteurs pour un même livre** | **Intégrations natives avec des systèmes externes** |
| **8** | **Catégories prédéfinies et ajout de nouvelles catégories** | **Maintenance et support technique continu** |
| **9** | |  | | --- | | **Historique des modifications de la date d'acquisition** |  |  | | --- | |  | |  |
| **10** | **Mise à jour automatique du statut et de la disponibilité** |  |
| **11** | **Calcul automatique du nombre de livres disponibles** |  |

**Tableau des livrables clés du projet**

|  |  |  |
| --- | --- | --- |
| **No** | **Nom du livrable clé** | **Description** |
| **1** | **Cahier des charges et charte de projet** | **Document combiné décrivant les exigences fonctionnelles, techniques, la portée, les objectifs et les contraintes.** |
| **2** | **Application desktop** | **Version destinée à une gestion complète des livres et documents sur ordinateur (PC/Mac).** |
| **3** | **Application mobile** | **Version adaptée aux appareils mobiles, avec des fonctionnalités comme le scan QR/ISBN et la recherche rapide. (Changement possible)** |

# Spécifications Fonctionnelles et Techniques

**Exigences Fonctionnelles**

Pour assurer son bon fonctionnement, notre application doit satisfaire une série de besoins utilisateurs, présentée comme suit :

* 1. **Formulaire de Livre complet** : permettre la gestion détaillée des informations sur chaque livre :
     + Un identifiant unique : généré de manière automatique
     + **Titre du livre** : Champ texte (titre unique).
     + **Auteur(s)** : Champ texte avec possibilité d’ajouter plusieurs auteurs
     + **Catégorie** : Menu déroulant, avec option d’ajouter de nouvelles catégories.
     + **ISBN** : Champ texte unique, empêchant les doublons.
     + **Date d’acquisition** : Champ calendrier avec suivi de l’historique des modifications.
     + **Statut** : Menu déroulant indiquant disponibilité, prêt ou réservation.
     + **Emprunteur actuel** : Champ lié à la base de données des utilisateurs
     + **Dates d’emprunt et de retour prévue** : Champs calendrier
     + **Disponibilité des exemplaires** : Calcul automatique
  2. **Gestion des Emprunts et Retours** : offrir aux utilisateurs la possibilité d’emprunter :
     + Enregistrement des emprunts et retours avec mise à jour en temps réel.
     + Suivi des disponibilités et envoi de rappels automatisés pour les retours.
  3. **Recherche et Consultation :** offrir aux utilisateur la possibilité de rechercher des livres :
     + Affichage en temps réel des résultats et détails des livres
     + Recherche par **code ISBN/QR**, titre, auteur ou catégorie
  4. **Interface Utilisateur :** 
     + **Formulaires structurés** pour enregistrement, emprunt et retour
     + **Tableau de bord** centralisé avec statistiques (total des livres, emprunts en cours, etc.).
     + Personnalisation de l’interface selon les besoins spécifiques.
  5. **Sécurité et Gestion des Accès :** 
     + **Authentification robuste** pour contrôler l’accès.
     + **Journal des modifications** pour tracer toutes les actions et modifications.

**Exigences Techniques**

Les choix de méthodologie et techniques pour le développement de l’application prennent en considérations 3 critères :

1. Une base de données de grande taille pour un nombre limité d’utilisateurs
2. Une application simple, moderne, multiplateforme et facilement maintenable
3. Utiliser des technologies et des approches faisant partie du programme de formation des étudiants en charge du développement

**Développement Desktop**

* + - **Framework :** Windows Presentation Foundation (WPF)
    - **Architecture :** Modèle MVVM (Model-View-ViewModel)
    - **Langage :** C#
    - **Base de données :** SQL Server ou PostgreSQL
    - **Bibliothèque tierce :** Prism ou CommunityToolkit.Mvvm pour simplifier l'implémentation de MVVM.
    - **API :** ASP.NET Core pour la communication entre le backend et l'application. (Optionnel)
    - **Méthode de déploiement :**
  + **Microsoft Store :** L'application sera déployée via le Microsoft Store, permettant aux utilisateurs de télécharger et mettre à jour l'application facilement.

**Développement Mobile**

* + - **Android :**
  + **Langage :** Kotlin
  + **Framework :** Android Jetpack et Room pour la gestion locale des données.
  + **Base de données locale :** SQLite ou Realm pour le stockage temporaire et la synchronisation hors ligne.
  + **API Backend :** ASP.NET Core RESTful API ou gRPC pour une communication rapide et fiable entre le mobile et le backend.
  + **Méthode de déploiement :**
    - **Google Play Store :** L'application Android sera déployée sur le Google Play Store pour un accès facile à l'installation et aux mises à jour.
    - **iOS :**
  + **Langage :** Swift
  + **Framework :** SwiftUI pour l'interface utilisateur et Core Data pour le stockage local des données.
  + **API Backend :** ASP.NET Core RESTful API ou gRPC pour garantir des communications rapides et fiables.
  + **Méthode de déploiement :**
    - **App Store :** L'application iOS sera déployée via l'App Store, permettant une distribution fluide et la gestion des mises à jour.

**Conception UI/UX**

* + - **Outil de design :** Figma
    - **Guidelines :**
  + **Android :** Design respectant les standards Material Design pour une expérience utilisateur cohérente sur Android.
  + **iOS :** Design respectant les Human Interface Guidelines d'Apple pour garantir une interface intuitive et native sur iOS.

**Outils de collaboration et gestion**

* + - **Contrôle de version :** Git avec GitHub pour la gestion du code source et la collaboration entre les développeurs.
    - **Gestion des tâches :** Jira pour organiser, suivre et gérer les tâches tout au long du développement du projet

**Contraintes et Considérations**

Le projet de développement de l’application s’oppose à des contraintes multiples et qui ont été prises en considération dans le cadre du présent livrable :

1. **Normes de Sécurité :** Respecter les régulations sur la confidentialité des données (RGPD ou équivalent).
2. **Compatibilité Multiplateforme :** Fonctionnement fluide sur desktop et mobile.
3. **Performances :** Capacité à traiter rapidement une grande base de données.
4. **Ressources Limitées :** Contraintes en temps, budget et effectifs nécessitant une gestion optimale.

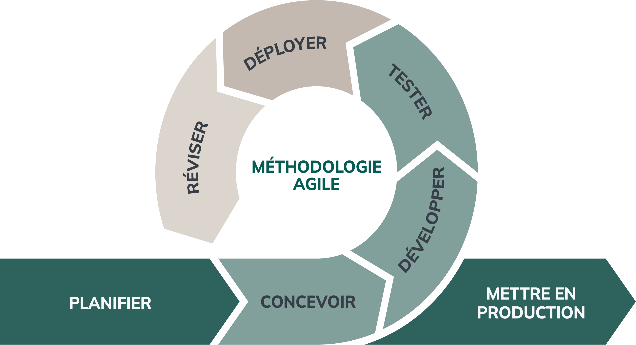
**Considérations**

1. **Scalabilité :** Prévoir des extensions futures (plus de livres, nouvelles fonctionnalités).
2. **Adoption Facile :** Une interface intuitive réduisant la formation nécessaire.
3. **Technologies Éducatives :** Technologies et approches correspondant au programme éducatif des étudiants impliqués dans le projet.

# Méthodologie et Approche

**Approche de Développement Choisie : Agile**

Nous avons choisi la méthodologie **Agile** pour le développement de ce projet pour plusieurs raisons :

* 1. **Flexibilité améliorée** : La méthodologie Agile nous offre la flexibilité nécessaire pour mieux répondre aux évolutions des besoins des utilisateurs, aux changements de priorités et aux situations imprévues.
  2. **Itérations rapides** : Avec des cycles de développements courts, nous pouvons rapidement produire des versions fonctionnelles de l’application. Cela permet de faciliter les tests, recueillir les retours des utilisateurs, et effectuer des ajustements continus du produits.
  3. **Communication constante** : L’approche Agile encourage une communication continue entre les membres de l’équipe et les parties prenantes, y compris les utilisateurs, garantissant ainsi une compréhension commune du projet.
  4. **Gestion proactive des risques** : La méthode Agile nous permet de repérer et de gérer les risques plus rapidement au cours du processus de développement, ce qui aide à mieux anticiper les défis potentiels.
  5. **Livraison progressive** : En adoptant la méthodologie Agile, nous avons la possibilité de livrer des fonctionnalités de manière progressive, ce qui nous permet de recueillir des retours clients tout au long de l’avancement du projet.

**Phases du Projet et Jalons Clés**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase** | **Nom du Jalons** | **Description** | **Date de Début** | **Date de Fin** |
| **Phase 1** | Approbation de la Charte de Projet | La charte du projet est approuvée et finalisée par toutes les parties prenantes. | |  | | --- | | 2025-01-06 | | |  | | --- | | 2025-01-10 | |
|  | Finalisation du Cahier des Charges et Charte de Projet | Les exigences, les fonctionnalités et les spécifications techniques du projet sont définies et validées. | 2025-01-11 | 2025-01-15 |
|  | Planification globale | Le calendrier détaillé du projet, incluant les ressources, les délais et jalons, est approuvé. | 2025-01-16 | 2025-01-20 |
|  | Finalisation de la Conception Technique | La conception technique du système, incluant les bases de données, interfaces utilisateurs et mesures de sécurité, est terminée. | 2025-01-21 | 2025-01-28 |
| **Phase 2** | Développement de l'Application Desktop – Gestion de livres | Développement de l’application desktop pour la gestion des livres et documents. | 2025-01-29 | 2025-02-11 |
|  | Tests unitaires de l'Application Desktop & Emprunt de livres | Les tests unitaires pour l'application desktop sont réalisés et validés. Ainsi que l’implémentation de l’emprunt de livre | 2025-02-12 | 2025-02-18 |
|  | Tests d'Intégration pour l'Application Desktop – Retour de livres | L'intégration des modules de l'application desktop est réalisée et les tests sont effectués. & Implémentation de retour de livre | 2025-02-19 | 2025-02-25 |
|  | Impression d’un ticket d’emprunt | Intégration du ticket d’emprunt | 2025-02-26 | 2025-03-04 |
|  | Développement de l'Application Mobile | Développement de l’application mobile pour la recherche et la vérification des ouvrages. | 2025-03-05 | 2025-03-19 |
|  | Tests unitaires de l'Application Mobile | Les tests unitaires pour l'application mobile sont réalisés et validés. | 2025-03-20 | 2025-04-03 |
|  | Tests d'Intégration pour l'Application Mobile | L'intégration des modules de l'application mobile est réalisée et les tests sont effectués. | 2025-04-04 | 2025-04-15 |
| **Phase 3** | Lancement des versions finales | La version finale de l'application est déployée et mise en service. | 2025-04-15 | 2025-04-15 |

# Modélisation UML et Conception

1. **Justification de l’utilisation de l’UML**

L'UML (Unified Modeling Language) est une norme largement adoptée dans le développement logiciel pour la modélisation des systèmes. L'utilisation de l'UML dans ce projet présente plusieurs avantages :

1. **Clarté et Communication :**
   * L'UML permet de créer des représentations visuelles claires des différents aspects du système. Cela facilite la communication entreles membres de l'équipe de développement, les parties prenantes, ainsi que les utilisateurs finaux. En outre, il aide à réduire les malentenduset les erreurs dans la compréhension du système.
2. **Documentation Structurée :**
   * L'UML fournit une documentation structurée et cohérente qui peut être utilisée tout au long du cycle de vie du projet. Cela permet de suivre l'évolution du système et de faciliter sa maintenance future.
3. **Analyse et Conception :**
   * L'UML permet d'identifier les besoins fonctionnels du système, de concevoir l'architecture du logiciel et de décrire les interactions entre les composants du système. Cela aide à mieux comprendre les exigences et à s'assurer que le système sera conforme aux besoins des utilisateurs.
4. **Flexibilité et Adaptabilité :**
   * Grâce à ses différents diagrammes, l'UML peut être utilisé à différents stades du développement, depuis la phase d'analyse des exigences jusqu'à la phase de conception détaillée.
5. **Diagrammes UML**

Voici les diagrammes UML qui seront utilisés pour modéliser le système de gestion de la bibliothèque :

* 1. **Diagramme de Cas d’Utilisation**

**Une image contenant texte, diagramme, ligne, capture d’écran

Description générée automatiquement**

**Une image contenant diagramme, capture d’écran, texte

Description générée automatiquement**

Description textuelle des cas d’utilisation

**Description textuelle**

**Identification :**

- Nom du cas : Authenticate

- Objectif : Permettre à un utilisateur de s'authentifier sur le système.

- Acteurs : Utilisateur, Système d'authentification (secondaire)

- Date : 11/12

- Responsables : Admin

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur veut se connecter à l'application.

- Préconditions : L'utilisateur a un compte existant.

- Enchaînement nominal :

1. L'utilisateur saisit son nom d'utilisateur et son mot de passe.

2. Le système vérifie les informations d'identification.

3. Le système permet l'accès si les informations sont correctes.

- Enchaînements alternatifs :

1. En (2) : Si les informations sont incorrectes, le système affiche un message d'erreur.

- Postconditions :

- L'utilisateur est authentifié et peut accéder aux fonctionnalités de l'application.

- Une session utilisateur est ouverte.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Le processus d'authentification doit être sécurisé.

- Confidentialité : Les informations d'identification ne doivent pas être divulguées.

- Contraintes liées à l'interface homme-machine :

- La connexion doit être intuitive et rapide.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Register

- Objectif : Permettre à un utilisateur de créer un nouveau compte.

- Acteurs : Utilisateur

- Date : 11/12

- Responsables : Admin

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur veut créer un compte.

- Préconditions : Aucun compte existant pour cet utilisateur.

- Enchaînement nominal :

1. L'utilisateur remplit le formulaire d'inscription avec les informations requises (nom, email, mot de passe).

2. Le système vérifie la validité des informations saisies.

3. Le système crée un nouveau compte et enregistre les informations.

- Enchaînements alternatifs :

1. En (2) : Si les informations sont invalides, le système demande des corrections.

- Postconditions :

- Le compte utilisateur est créé.

- L'utilisateur peut se connecter avec les nouvelles informations.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les données doivent être enregistrées correctement.

- Confidentialité : Les informations personnelles doivent être sécurisées.

- Contraintes liées à l'interface homme-machine :

- Le formulaire d'inscription doit être simple et facile à comprendre.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Reserve a book

- Objectif : Permettre à un utilisateur de réserver un livre.

- Acteurs : Utilisateur

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur veut réserver un livre.

- Préconditions : L'utilisateur doit être connecté.

- Enchaînement nominal :

1. L'utilisateur sélectionne un livre disponible.

2. Le système vérifie la disponibilité du livre.

3. Le système réserve le livre pour l'utilisateur.

4. Le système notifie l'utilisateur de la réservation réussie.

- Enchaînements alternatifs :

1. En (2) : Si le livre n'est pas disponible, le système notifie l'utilisateur.

- Postconditions :

- Le livre est réservé pour l'utilisateur.

- Une notification est envoyée à l'utilisateur.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les réservations doivent être mises à jour en temps réel.

- Confidentialité : Les informations sur les réservations doivent être sécurisées.

- Contraintes liées à l'interface homme-machine :

- Le processus de réservation doit être fluide et rapide.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Request a new book

- Objectif : Permettre à un utilisateur de demander l'ajout d'un nouveau livre au catalogue.

- Acteurs : Utilisateur

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur veut demander un nouveau livre.

- Préconditions : L'utilisateur doit être connecté.

- Enchaînement nominal :

1. L'utilisateur remplit un formulaire de demande de nouveau livre.

2. Le système enregistre la demande et l'envoie au bibliothécaire.

3. Le bibliothécaire examine la demande.

4. Le système notifie l'utilisateur de l'état de la demande (approuvée ou rejetée).

- Enchaînements alternatifs :

1. En (3) : Si le bibliothécaire rejette la demande, l'utilisateur est notifié.

- Postconditions :

- La demande de nouveau livre est enregistrée.

- L'utilisateur est informé de la décision.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les demandes doivent être traitées de manière cohérente.

- Confidentialité : Les informations sur les demandes doivent être protégées.

- Contraintes liées à l'interface homme-machine :

- Le formulaire de demande doit être clair et facile à remplir.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Loan a book

- Objectif : Permettre à un utilisateur d'emprunter un livre.

- Acteurs : Utilisateur, Bibliothécaire (secondaire)

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur veut emprunter un livre.

- Préconditions : L'utilisateur doit être connecté et avoir une réservation ou choisir un livre disponible.

- Enchaînement nominal :

1. L'utilisateur sélectionne un livre à emprunter.

2. Le système vérifie la disponibilité du livre.

3. Le bibliothécaire confirme l'emprunt.

4. Le système met à jour l'état du livre comme emprunté.

- Enchaînements alternatifs :

1. En (2) : Si le livre n'est pas disponible, le système notifie l'utilisateur.

- Postconditions :

- Le livre est marqué comme emprunté.

- Une notification est envoyée à l'utilisateur.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les emprunts doivent être enregistrés de manière fiable.

- Confidentialité : Les informations sur les emprunts doivent être sécurisées.

- Contraintes liées à l'interface homme-machine :

- Le processus d'emprunt doit être intuitif et rapide.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Update catalog

- Objectif : Permettre au bibliothécaire de mettre à jour le catalogue des livres.

- Acteurs : Bibliothécaire

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un bibliothécaire veut mettre à jour le catalogue.

- Préconditions : Le bibliothécaire doit être connecté.

- Enchaînement nominal :

1. Le bibliothécaire accède à la section de gestion du catalogue.

2. Le bibliothécaire ajoute, modifie ou supprime des informations sur les livres.

3. Le système enregistre les modifications et met à jour le catalogue.

- Enchaînements alternatifs :

1. En (2) : Si les informations sont invalides, le système demande des corrections.

- Postconditions :

- Le catalogue est mis à jour.

- Les changements sont visibles pour les utilisateurs.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les mises à jour doivent être enregistrées correctement.

- Confidentialité : Les informations sur les mises à jour doivent être sécurisées.

- Contraintes liées à l'interface homme-machine :

- L'interface de mise à jour doit être simple et efficace.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Manage user account

- Objectif : Permettre à l’administrateur de gérer les comptes des utilisateurs.

- Acteurs : Admin

- Date : 11/12

- Responsables : Admin

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un bibliothécaire veut gérer les comptes des utilisateurs.

- Préconditions : Le bibliothécaire doit être connecté.

- Enchaînement nominal :

1. Le bibliothécaire accède à la section de gestion des comptes.

2. Le bibliothécaire ajoute, modifie ou supprime des comptes utilisateurs.

3. Le système enregistre les modifications.

- Enchaînements alternatifs :

1. En (2) : Si les informations sont invalides, le système demande des corrections.

- Postconditions :

- Les comptes utilisateurs sont mis à jour.

- Les modifications sont effectives immédiatement.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Pay fine

- Objectif : Permettre à un utilisateur de payer une amende pour un livre retourné en retard.

- Acteurs : Utilisateur, Système de paiement (secondaire)

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur doit payer une amende pour un livre en retard.

- Préconditions : L'utilisateur doit être connecté.

- Enchaînement nominal :

1. L'utilisateur accède à la section des amendes dans son compte.

2. L'utilisateur sélectionne l'amende à payer.

3. Le système affiche le montant de l'amende et les options de paiement.

4. L'utilisateur saisit les informations de paiement.

5. Le système vérifie les informations de paiement et effectue la transaction.

6. Le système notifie l'utilisateur du paiement réussi.

- Enchaînements alternatifs :

1. En (5) : Si les informations de paiement sont incorrectes, le système demande à l'utilisateur de les ressaisir.

- Postconditions :

- L'amende est payée.

- Une notification est envoyée à l'utilisateur.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les transactions doivent être sécurisées.

- Confidentialité : Les informations de paiement doivent être protégées.

- Contraintes liées à l'interface homme-machine :

- Le processus de paiement doit être simple et intuitif.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Feedback

- Objectif : Permettre à un utilisateur de donner son avis sur les services de la bibliothèque.

- Acteurs : Utilisateur

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un utilisateur veut donner son avis.

- Préconditions : L'utilisateur doit être connecté.

- Enchaînement nominal :

1. L'utilisateur accède à la section des avis dans son compte.

2. L'utilisateur remplit le formulaire de feedback avec ses commentaires et suggestions.

3. Le système enregistre le feedback et notifie l'utilisateur de sa soumission.

- Enchaînements alternatifs :

1. En (2) : Si le formulaire est incomplet, le système demande à l'utilisateur de le compléter.

- Postconditions :

- Le feedback est enregistré.

- L'utilisateur est informé de la réception de son avis.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les avis doivent être enregistrés sans perte de données.

- Confidentialité : Les commentaires doivent être anonymes si demandé.

- Contraintes liées à l'interface homme-machine :

- Le formulaire de feedback doit être clair et facile à remplir.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Send overdue notification

- Objectif : Notifier un utilisateur lorsqu'un livre emprunté est en retard.

- Acteurs : Système, Utilisateur (secondaire)

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un livre emprunté devient en retard.

- Préconditions : L'utilisateur doit avoir emprunté un livre.

- Enchaînement nominal :

1. Le système vérifie la date de retour des livres empruntés.

2. Le système identifie les livres en retard.

3. Le système envoie une notification à l'utilisateur pour les livres en retard.

- Enchaînements alternatifs :

- Aucune

- Postconditions :

- L'utilisateur est informé des livres en retard.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les notifications doivent être envoyées en temps opportun.

- Confidentialité : Les informations sur les emprunts doivent être protégées.

- Contraintes liées à l'interface homme-machine :

- La notification doit être claire et fournir des instructions sur la marche à suivre.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Send reservation available notification

- Objectif : Notifier un utilisateur qu'un livre réservé est disponible.

- Acteurs : Système, Utilisateur (secondaire)

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'un livre réservé par un utilisateur devient disponible.

- Préconditions : L'utilisateur doit avoir réservé un livre.

- Enchaînement nominal :

1. Le système vérifie la disponibilité des livres réservés.

2. Le système identifie les livres disponibles.

3. Le système envoie une notification à l'utilisateur pour les livres disponibles.

- Enchaînements alternatifs :

- Aucune

- Postconditions :

- L'utilisateur est informé de la disponibilité du livre réservé.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les notifications doivent être envoyées immédiatement.

- Confidentialité : Les informations sur les réservations doivent être protégées.

- Contraintes liées à l'interface homme-machine :

- La notification doit être claire et fournir des instructions sur la marche à suivre.

-----------------------------------------------------------------------------------------------------------------------------

**Description textuelle**

**Identification :**

- Nom du cas : Send reservation canceled notification

- Objectif : Notifier un utilisateur qu'une réservation a été annulée.

- Acteurs : Système, Utilisateur (secondaire)

- Date : 11/12

- Responsables : Bibliothécaire

- Version : 1.0

**Séquences :**

- Le cas d'utilisation commence lorsqu'une réservation d'un livre est annulée.

- Préconditions : L'utilisateur doit avoir réservé un livre.

- Enchaînement nominal :

1. Le système identifie les réservations annulées.

2. Le système envoie une notification à l'utilisateur pour les réservations annulées.

- Enchaînements alternatifs :

- Aucune

- Postconditions :

- L'utilisateur est informé de l'annulation de la réservation.

**Rubriques optionnelles :**

- Contraintes non fonctionnelles :

- Fiabilité : Les notifications doivent être envoyées immédiatement.

- Confidentialité : Les informations sur les réservations doivent être protégées.

- Contraintes liées à l'interface homme-machine :

- La notification doit être claire et fournir des instructions sur la marche à suivre.

-----------------------------------------------------------------------------------------------------------------------------

* 1. **Diagramme de Classe**

**Une image contenant texte, diagramme, ligne, Plan

Description générée automatiquement**

* 1. **Diagramme d’Objet**

**Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement**

* 1. **Diagrammes de Séquence**

**Une image contenant texte, capture d’écran, diagramme

Description générée automatiquement**

**Une image contenant texte, capture d’écran, diagramme, conception

Description générée automatiquement**

**Une image contenant texte, capture d’écran, diagramme, nombre

Description générée automatiquement**

* 1. **Diagrammes d’Activité**

**Une image contenant diagramme, croquis, ligne, Parallèle

Description générée automatiquement**

**Une image contenant diagramme, ligne, texte, capture d’écran

Description générée automatiquement**

**Une image contenant diagramme, texte, ligne, Plan

Description générée automatiquement**

* 1. **Diagrammes d’Etats-Transitions**

**Une image contenant texte, capture d’écran, diagramme, ligne

Description générée automatiquement**

**Une image contenant texte, diagramme, capture d’écran, Plan

Description générée automatiquement**

**Une image contenant texte, capture d’écran, diagramme, Rectangle

Description générée automatiquement**

# Design d’Interfaces

**Veuillez consulter ce lien pour toute description d’interfaces présente dans l’application :**<https://www.figma.com/design/vfVEY1QkUdbWP5od229RVJ/Library-Management-System?node-id=0-1&t=DwiZpB7A2RoP71co-1>

# Planification et Gestion de Projet (Jira)

**Une image contenant texte, logiciel, nombre, Icône d’ordinateur

Description générée automatiquement**

**Une image contenant texte, logiciel, Icône d’ordinateur, capture d’écran

Description générée automatiquement**

**Une image contenant texte, logiciel, Icône d’ordinateur, Page web

Description générée automatiquement**

**Une image contenant texte, logiciel, nombre, Icône d’ordinateur

Description générée automatiquement**

**Une image contenant texte, nombre, logiciel, Icône d’ordinateur

Description générée automatiquement**

# Validation et Maintenance

Les modalités de validation varient en fonction des besoins spécifiques du projet et des parties prenantes.

Il y aura des validations effectuées par toutes les parties prenantes du projet, et chacun aura ses propres critères pour approuver la validation.

## Critères de validation coté équipe de développement

* 1. **Succès des tests logiciels dans divers contextes**, avec des vérifications pour chaque fonctionnalité, comprenant les saisies et les résultats escomptés.
  2. **Tests de charge et de réactivité** du logiciel face aux requêtes multiples, simulant également la performance sur un ordinateur et un téléphone avec des caractéristiques de processeur et de mémoire vive similaires aux équipements du laboratoire. Cela inclut même les scénarios de multitâche sur l’ordinateur concerné.
  3. **Protection des données** avec un cryptage robuste, assorti de tests sur la résilience de la base de données et du logiciel face aux attaques extérieures, telles que les virus et autres menaces.

#### Accessibilité de l’interface utilisateur, avec :

* + - Un bon contraste et une palette de couleurs harmonieuse,
    - Des menus et boutons clairs, intuitifs, et bien représentatifs des fonctions proposées,
    - Des « infobulles » et des "infoBox" sur certaines fonctions complexes en mode « survol » pour faciliter l’utilisation sans consulter la documentation,
    - Compatibilité avec les lecteurs d’écran et navigation au clavier.
  1. **Uniformité de conception sur toutes les pages**, avec un modèle cohérent pour toutes les fonctionnalités.
  2. **Messages d'erreur clairs**, accompagnés de liens vers des solutions adéquates en cas d’import de fichiers mal formatés ou autres erreurs.
  3. **Performance d’exécution rapide**, avec une intégration fluide dans l’infrastructure informatique existante du laboratoire, même pour le traitement de grandes quantités de données et de requêtes sur le serveur.
  4. **Sécurité et gestion des accès** aux fonctionnalités et données, différenciées pour chaque employé en fonction de leur rôle et de leurs responsabilités.
  5. **Documentation complète et manuel d'utilisation** pour permettre aux nouveaux utilisateurs de trouver facilement les informations et les étapes nécessaires pour réaliser les opérations souhaitées.

## Critères de validation coté client

Concernant l'utilisateur, l'évaluation portera sur l'ensemble de ces aspects. Nous prévoyons de proposer des formulaires de mesure du niveau de satisfaction permettant aux employés de donner des notes de 1 à 5 à chaque fonctionnalité après les tests Beta. Nous offrons également aux utilisateurs la possibilité de :

* Ajouter leurs commentaires dans des champs libres pour recueillir leurs retours et attentes, et surtout pour identifier tout décalage entre les fonctionnalités attendues et celles fournies,
* Suggérer des ajustements.

En conclusion, si la majorité des utilisateurs attribuent une note supérieure à 4 au logiciel, le projet sera considéré comme réussi, et la livraison pourra alors avoir lieu.

## Modalités de Suivi et de Maintenance

Comme pour toute application ou service proposé, il est crucial que nos applications ne demeurent pas figées dans leur version initiale. Elles doivent être évolutives et s'adapter au fil du temps pour diverses raisons.

Premièrement, du point de vue fonctionnel, il peut arriver qu’un problème technique, non identifié lors des tests, apparaisse après plusieurs jours ou mois d’utilisation par les clients.

Ensuite, de nouvelles règles de sécurité ou des mises à jour peuvent être nécessaires, notamment à la suite des évolutions du système d’exploitation, ou en raison de la découverte de nouvelles failles ou menaces dans l’environnement numérique.

Pour une meilleure compréhension de l'application, une FAQ sera incluse, accompagnée de tutoriels vidéo expliquant clairement l’utilisation des fonctionnalités proposées.

L’ajout de nouvelles fonctionnalités pourra aussi être envisagé à l’avenir, en fonction de l'évolution de la clinique ou de ses besoins en gestion des dossiers médicaux. Cependant, avant leur intégration, les nouvelles fonctionnalités devront être soigneusement étudiées par les équipes DevOps et développement, pour en évaluer la faisabilité et assurer une implémentation sécurisée sans risques de dysfonctionnements.

Il est donc essentiel d'assurer un service de maintenance, de suivi et de mise à jour pour la clinique après la livraison du système de gestion des dossiers médicaux.

Enfin, nous mettrons en place divers moyens de communication afin de faciliter aux médecins et administrateurs l’accès à l’assistance, que ce soit par courriel, téléphone, ou même par des sessions de support en ligne.

# Création de compte Github

Lien vers le repositery github : [github](https://github.com/AKameni1/LMS)

# Annexes

* <https://etcdigital.fr/blog/quest-ce-que-la-methodologie-agile/>
* [https://learn.microsoft.com/en-us/visualstudio/get- started/csharp/tutorial-wpf?view=vs-2022](https://learn.microsoft.com/en-us/visualstudio/get-%20started/csharp/tutorial-wpf?view=vs-2022)
* Charte de projet v.1.1