Rapport Projet UA2



Fait par

Caleb Kastner

À soumettre le 31 mars 2024

Cours de Programmation Orientée Objet en C#

Kodzo Michel Aladji

La Cité Collégiale

Programmation Informatique, 2ème session

Table des matières

[Introduction 3](#_Toc162821705)

[Analyse des Besoins 3](#_Toc162821706)

[Fonctionnalités Principales : 3](#_Toc162821707)

[Conception du système 5](#_Toc162821708)

[Diagramme de classe UML 5](#_Toc162821709)

[Explication des classes, des attributs et méthodes 6](#_Toc162821710)

[Explication du fonctionnement logique 7](#_Toc162821711)

[Implémentation 8](#_Toc162821712)

[Problème avec la méthode VerificationInformation : 8](#_Toc162821713)

[Problème avec la méthode Connexion : 8](#_Toc162821714)

[Écran encombré après être rendu loin dans le programme : 8](#_Toc162821715)

[Ajout de messages lorsqu’un livre n’existait pas dans la bibliothèque : 8](#_Toc162821716)

[Problème qu’un livre réservé ne pouvait pas être emprunté : 9](#_Toc162821717)

[Problème d’exception de type lors de l’inscription : 9](#_Toc162821718)

[Ajout de vérification si l’entrée est nulle : 9](#_Toc162821719)

[Difficulté à réaliser correctement le diagramme de classes UML : 10](#_Toc162821720)

[Résultats et Discussion 11](#_Toc162821721)

[Fonctionnalités réussies 11](#_Toc162821722)

[Fonctionnalités qui n'ont pas pu être implémentées : 11](#_Toc162821723)

[Conclusion 11](#_Toc162821724)

[Références 12](#_Toc162821725)

[Annexes 12](#_Toc162821726)

## Introduction

Dans le cadre du projet UA2 de programmation C#, chaque équipe était chargée de concevoir et développer un type spécifique de système informatique, avec des contraintes et des exigences précises. Notre mission consistait à créer un système de gestion de bibliothèque dynamique. Cette plateforme vise à donner l'expérience de recherche, d'emprunt, de retour, et de réservation de livres pour les utilisateurs. Pour cela, chaque livre est enregistré avec des informations détaillées telles que le titre et l'auteur, tandis que les utilisateurs sont identifiés de manière unique par leur nom et leurs informations de contact. Un historique complet des transactions de chaque utilisateur est également maintenu, incluant les dates précises d'emprunt et de retour.

Ce rapport présente une analyse des besoins qui ont guidé la conception de notre système, en mettant l'accent sur la facilité d'utilisation et l'accessibilité et tout en discutant les défis techniques rencontrés et les solutions adoptées pour les surmonter. Enfin, le trajet de notre projet sera exposé, retraçant le développement depuis la phase de conception initiale jusqu'à la mise en œuvre finale.

## Analyse des Besoins

### Fonctionnalités Principales :

En lisant la description du projet du système de gestion d’une bibliothèque, plusieurs fonctionnalités nécessaires sont très évidentes. Dans la section suivante, ces fonctionnalités vont être présentée, ainsi que quelques autres qui s’avèrent cruciales au bon fonctionnement du système.

#### Gestion des Livres :

**Ajouter un livre :** Permettre aux bibliothécaires d'ajouter de nouveaux livres à la collection avec les informations pertinentes telles que le titre, l'auteur, et autres détails pertinents.

**Rechercher un livre :** Offrir aux utilisateurs la possibilité de rechercher des livres dans la base de données de la bibliothèque en fonction de critères tels que le titre, l'auteur, ou le genre.

**Afficher les détails d'un livre :** Permettre aux utilisateurs de consulter les informations détaillées d'un livre spécifique.

#### Gestion des Emprunts :

**Emprunter un livre :** Permettre aux utilisateurs inscrits de procéder à l'emprunt de livres disponibles.

**Retourner un livre :** Autoriser les utilisateurs à retourner les livres empruntés une fois qu'ils ont terminé leur lecture.

**Réservation de livre :** Offrir la possibilité de réserver un livre qui n'est pas actuellement disponible.

#### Gestion des Utilisateurs :

**Inscription des utilisateurs :** Permettre aux nouveaux utilisateurs de s'inscrire en fournissant leur nom et leurs informations de contact.

**Consulter l'historique des emprunts :** Offrir aux utilisateurs la possibilité de consulter leur historique d'emprunts, y compris les livres empruntés précédemment et les dates associées.

#### Gestion des Prêts :

**Enregistrer les détails du prêt :** Enregistrer les informations relatives aux emprunts, y compris les dates de prêt et de retour, ainsi que les détails de l'utilisateur emprunteur et du livre emprunté.

## Conception du système

Afin de réaliser le système de gestion et implémenter les besoins, il va falloir planifier comment concevoir le tout, et savoir par où commencer. Grâce à la liste de fonctionnalités qui vont devoir être implémentés, la conception du système devient beaucoup plus facile. Les classes et les méthodes qu’il va falloir ajouter au système vont se baser sur la section précédente. Cette section va explorer la conception du système, la logique, les liens, et plus sur comment le système va fonctionner.

En premier, il fallait faire un diagramme de classe UML, en démontrant les classes, les attributs et les méthodes. Cependant, cette tâche a été sans démonstration, formation ou exemple à suivre, rendant très difficile de faire exactement ce que le professeur désirait. En suivant des guides en ligne, notre équipe a fini avec ceci :

### Diagramme de classe UML

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Le diagramme UML ci-dessus nous permet de voir les différentes classes, les différents attributs et les différentes méthodes.

### Explication des classes, des attributs et méthodes

#### Classe « Bibliothèque »

Premièrement, la classe centrale de notre système est nommée « Bibliothèque ». Elle englobe une liste de tous les livres disponibles dans la bibliothèque, les informations relatives aux utilisateurs inscrits, ainsi qu'un historique complet des emprunts effectués, permettant ainsi de suivre les prêts de manière efficace. Cette classe est dotée de plusieurs méthodes essentielles au bon fonctionnement de la bibliothèque, parmi lesquelles : une méthode pour ajouter des livres à la collection, une méthode pour enregistrer les utilisateurs, et une autre pour consigner les emprunts dans l'historique des prêts. Elle propose également une méthode de vérification des informations des utilisateurs lors de leur connexion, une méthode de recherche dans le catalogue de livres, et enfin, une méthode pour lister tous les livres disponibles.

La pertinence de ces fonctionnalités est évidente au regard des besoins opérationnels d'une bibliothèque et des spécifications du projet, soulignant leur importance cruciale pour son fonctionnement optimal.

#### Classe « GenerateurDonneesTest »

Deuxièmement, une classe intitulée « GenerateurDonneesTest » comporte deux méthodes spécifiques destinées à vérifier le bon fonctionnement du système par la création de livres et d'utilisateurs fictifs. Cette classe joue un rôle clé dans le test des différentes fonctionnalités telles que les emprunts, les retours, les réservations, et bien d'autres. Elle établit un lien direct avec la classe « Livre » pour la génération de livres fictifs, et avec la classe « Utilisateur » pour la création d'utilisateurs fictifs.

#### Classe « Utilisateur »

Troisièmement, la classe « Utilisateur » regroupe divers attributs pour stocker les informations personnelles de l'utilisateur, incluant prénom, nom, adresse, détails de connexion, et autres. Elle est enrichie de plusieurs méthodes facilitant l'inscription, la connexion, ainsi que la consultation des informations relatives au compte de l'utilisateur, telles que les emprunts en cours, les réservations, l'historique des emprunts et les informations générales. Une interaction est prévue entre cette classe et la classe « Bibliothèque » afin d'ajouter automatiquement un nouvel utilisateur à la liste d'utilisateurs lors de son inscription. Un lien est également établi avec la classe « Emprunt », autorisant ainsi l'utilisateur à emprunter des livres.

#### Classe « Emprunt »

Quatrièmement, la classe « Emprunt » se concentre exclusivement sur la gestion des prêts de livres, avec des attributs décrivant chaque emprunt pour maintenir un historique détaillé.

#### Classe « Livre »

Finalement, la classe « Livre » conserve les informations essentielles sur chaque livre, telles que le titre, l'ISBN, l'auteur, ainsi que le statut courant de l'ouvrage. Lorsqu'un livre est emprunté, son statut passe à « Emprunté », et une date de retour est fixée à 31 jours après la date d'emprunt. En cas de retour ou de réservation du livre, son statut est mis à jour en conséquence.

### Explication du fonctionnement logique

Bien qu'il soit essentiel de définir les différentes classes et leurs attributs, il est tout aussi important de comprendre le fonctionnement global du système et ce qui se produit lors du démarrage du programme. Lorsque le programme démarre, il génère automatiquement 9 utilisateurs, chacun associé à des informations telles que leur numéro (Prénom1, Nom1, Adresse1, etc.), ainsi que 9 livres avec des informations correspondantes à leur ordre (Livre1, Auteur1, etc.). Ensuite, l'écran d'accueil de la bibliothèque s'affiche, proposant le choix entre s'inscrire ou se connecter.

L'utilisateur peut alors choisir de se connecter, ce qui déclenche une recherche dans la liste des utilisateurs, gérée par la classe Bibliothèque, ou de s'inscrire pour créer son propre compte, ajoutant ainsi une nouvelle instance à la liste des Utilisateurs. Une fois connecté ou inscrit, l'utilisateur est dirigé vers la page d'accueil du système, offrant diverses fonctionnalités telles que la visualisation des emprunts et des réservations, l'emprunt ou le retour de livres, et bien d'autres encore.

## Implémentation

Dans la section suivante, nous allons discuter de l'implémentation des classes et d'autres aspects dans le logiciel Visual Studio 2022, tout en abordant les difficultés rencontrées et les solutions adoptées pour les surmonter.

### Problème avec la méthode VerificationInformation :

La méthode VerificationInformation comparait les attributs numeroDeCarteConnexion et motDePasseConnexion de l'objet utilisateur avec ceux de tous les utilisateurs enregistrés dans la bibliothèque. Cependant, ces attributs n’étaient pas encore définis dans la classe Utilisateur.

**Solution trouvée :** Utiliser les attributs appropriés de la classe Utilisateur, tels que NumeroCarte et MotDePasse et les implémenter.

### Problème avec la méthode Connexion :

La méthode Connexion de la classe Utilisateur ne permettait pas à l'utilisateur de réessayer de se connecter en cas d'échec de la première tentative. Cela était un gros problème puisque l’utilisateur devait recommencer le programme après chaque essai manqué, ce qui s’avérait très long.

**Solution trouvée :** Utiliser une boucle do-while dans la méthode Connexion pour permettre à l'utilisateur de réessayer tant qu'il n'a pas fourni les bonnes informations d'identification.

### Écran encombré après être rendu loin dans le programme :

Après de longues heures consacrées au dépannage et à la correction des erreurs successives, l'écran du terminal de commande s'est trouvé encombrée, créant un grand ennui pour la lisibilité de l'interface pour l'utilisateur. Les lignes de texte s'accumulaient, rendant l'expérience d'utilisation de plus en plus confuse.

**Solution trouvée :** En utilisant la commande Console.Clear(), nous avons pu rafraîchir l'écran, ce qui a permis de nettoyer l'affichage. En ajustant également certaines pages, l'écran est devenu beaucoup plus propre et l'expérience utilisateur en a été grandement améliorée.

### Ajout de messages lorsqu’un livre n’existait pas dans la bibliothèque :

Le problème initial était que lorsque l'utilisateur recherchait un livre qui n'existait pas dans la bibliothèque, le message généré ne donnait pas une indication claire de la raison pour laquelle la réservation n'était pas possible. Le message générique "Le livre n'est pas disponible" n'informait pas précisément l'utilisateur sur la nature du problème.

**Solution trouvée :** Nous avons modifié le message d'erreur pour inclure une explication plus précise. En particulier, le message d'erreur modifié indique désormais que le livre recherché n'existe pas dans la bibliothèque ou n'est pas disponible, clarifiant la situation pour l'utilisateur.

### Problème qu’un livre réservé ne pouvait pas être emprunté :

En raison de la logique de la méthode "Emprunter", il était uniquement possible d'emprunter un livre qui avait un statut "Disponible". Ainsi, lorsqu'un livre était réservé par un utilisateur, il devenait impossible de l'emprunter par la suite.

**Solution trouvée :** Pour remédier à ce problème, il a été nécessaire d'ajouter un argument supplémentaire qui permettait l'emprunt d'un livre lorsque son statut était marqué comme disponible, mais également lorsqu'il était réservé par le même utilisateur qui était connecté.

### Problème d’exception de type lors de l’inscription :

Lorsqu'un utilisateur saisissait ses informations et que le type d'entrée n'était pas correct (par exemple, un string à la place d'un entier pour le numéro de téléphone), le programme affichait une erreur.

**Solution trouvée :** Pour remédier à ce problème, une boucle do-while a été ajoutée avec un booléen, permettant de redemander à l'utilisateur de saisir le bon type d'entrée tant que ce n'était pas le cas. Cette approche a été utilisée pour la saisie du numéro de téléphone et du mot de passe, assurant ainsi une saisie correcte des données.

### Ajout de vérification si l’entrée est nulle :

Dans plusieurs parties du code, si l'entrée était nulle, le programme s'arrêtait et affichait une erreur. De même, si l'utilisateur faisait une erreur en appuyant accidentellement sur une touche, le programme plantait et nécessitait un redémarrage complet. Pour remédier à cette situation, où le programme s'arrêtait brusquement et affichait des erreurs en cas d'entrée nulle ou d'erreur utilisateur, une solution efficace consiste à implémenter une gestion des exceptions.

**Solution trouvée :** En utilisant des boucles do-while, il fut possible de redemander une entrée tant qu’une des possibilités n’étaient pas atteinte. De cette manière, si l'utilisateur entre une valeur nulle, la demande d’entrée est réaffichée, invitant l'utilisateur à fournir à nouveau la valeur attendue, plutôt que de laisser le programme planter. Cette approche renforce la robustesse du programme et améliore l'expérience utilisateur en cas d'erreur de saisie.

### Difficulté à réaliser correctement le diagramme de classes UML :

L'un des défis rencontrés lors de la planification du système a été la création du diagramme de classes UML. Il s'est avéré très ardu de comprendre comment élaborer un tel diagramme lorsque nous n'avions jamais appris ni même vu un exemple de diagramme UML auparavant.

**Solution trouvée :** Nous avons entrepris des recherches pour obtenir une compréhension générale de ce à quoi devrait ressembler un diagramme UML, puis nous avons approfondi nos investigations pour comprendre plus précisément comment procéder. Cependant, cette tâche s'est avérée plus complexe que prévu, car les descriptions d'un diagramme UML variaient considérablement. Malgré cela, nous avons fait de notre mieux pour représenter fidèlement le système.

## Résultats et Discussion

Dans la section suivante, nous allons parler des fonctionnalités implémentées et celles qui n'ont pas pu être implémentées.

### Fonctionnalités réussies

Tout d'abord, le générateur de données de test s'est avéré être une fonctionnalité cruciale qui a grandement facilité le processus de débogage. En effet, cette fonctionnalité a permis de tester les différentes actions que l'utilisateur pouvait entreprendre sans avoir à s'inscrire préalablement.

En outre, l'obtention de livres de test s'est révélée être un atout majeur pour évaluer les fonctionnalités telles que l'emprunt, la réservation et le retour.

L'inscription des utilisateurs constitue une autre fonctionnalité essentielle qui a rendu le système accessible à tous, permettant à n'importe qui de s'inscrire facilement.

Enfin, la fonction de rafraîchissement de l'écran et l'option de retour en arrière après avoir terminé une action ont été des réussites importantes car elles ont amélioré l'expérience utilisateur en assurant une navigation fluide et intuitive dans le système.

### Fonctionnalités qui n'ont pas pu être implémentées :

Certaines fonctionnalités n'ont pas pu être implémentées en raison de contraintes de temps et de connaissances. Par exemple, l'option permettant de revenir à l'écran de connexion et de se connecter à nouveau n'a pas été mise en œuvre, de même que la connexion à une base de données. Cette dernière n'a pas pu être réalisée en raison de nos limites de connaissances.

De plus, l'ajout d'un livre à partir de la ligne de commande n'a pas été possible car l'utilisateur n'aurait pas eu accès à une telle fonctionnalité. Pour permettre cette fonction, il aurait fallu la possibilité de se connecter en tant qu'administrateur.

## Conclusion

Ce projet a représenté un défi qui a permis à l’équipe d'approfondir et d'appliquer les connaissances acquises en classe. Grâce à une révision approfondie et à un travail acharné, nous avons pu créer une plateforme fonctionnelle répondant aux exigences spécifiques du projet.

Malgré les obstacles rencontrés, tels que les difficultés avec le diagramme de classes UML et les problèmes d'implémentation des méthodes, notre équipe a su surmonter ces défis en trouvant des solutions appropriées. L'ajustement des méthodes, la gestion des exceptions et l'amélioration de l'interface utilisateur ont contribué à renforcer la robustesse et l'accessibilité du système.

En somme, ce projet UA2 a été une étape cruciale dans notre parcours de développement logiciel, nous dotant de compétences précieuses et nous préparant à relever de nouveaux défis dans le monde de la programmation orientée objet en C#.

## Références

<https://gatineau.ent.sirsidynix.net/client/fr_CA/default/?dt=list>

<https://www.bibliothequedequebec.qc.ca/>

<https://www.w3schools.com/cs/index.php>

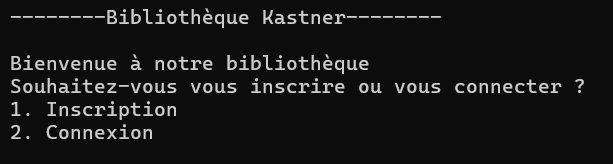
<https://creately.com/blog/fr/uncategorized-fr/tutoriel-sur-les-diagrammes-de-classe/>

## Annexes

#### Répertoire GitHub vers le code source :

<https://github.com/calebk5/GestionBibliotheque>

#### Captures d’écran :



Une image contenant texte, capture d’écran, Police, noir

Description générée automatiquement

