Rapport de projet

Lino - Boîtes à livres

Razafindrakoto Nathan[a], **Hoang Quan Tran**[a] **and Lavolet Oscar**[a]

[a]*Université de Montréal*

L-E. Lafontant

Abstract—En utilisant MongoDB, Flutter, Dart et typescript, nous avons créé l'application Lino pour aider à promouvoir l'initiative du Club de Lecture de l'UdeM (CLUM), les boîtes à livres dispersées dans les différents emplacements de campus de l'Université.
L'application permet de situer les boîtes à livres, savoir quels livres sont dedans, avoir les information concernant ces mêmes livres, échanger avec les autres utilisateurs ou même demander des livres et recevoir des notifications vis-à-vis des livres demandés quand ceux-cis sont ajoutés à une boîte à livre.

Keywords— *IFT 3150, Lino, CLUM, UdeM, application mobile, Flutter, MongoDB*

Contents

1. Introduction

La lecture est souvent solitaire, et malgré le but des livres d'assembler des personnes autour d'une histoire, le rat de bibliothèque n'est pas particulièrement entouré. Heureusement, en cette époque moderne, avec nos téléphones, l'on peut partager sa lecture et ses avis, de plus, des initiatives ont été mises en place pour promouvoir le libre accès des livres.

Cependant, l'infrastructure actuelle limite la portée de l'initiative, en effet, l'information est mal distribuée, ce qui rend compliqué de trouver les boîtes à livres ou d'avoir une idée des livres qu'elles contiennent.

Ainsi, Lino a été conçu pour être l'outil parfait pour communiquer avec d'autres passionnés de livres et étendre l'initiative des boîtes à livre du Club de lecture de l'UdeM (CLUM). Ces petits paniers de bois dispersés dans l'enceinte de l'Université, contenant les livres généreusement donnés par la communauté étudiante, seront accesibles à travers l'interface de Lino, facilement triable et trouvables à travers un design intuitif et plaisant à la vue.

2. Objectifs

2.1. Faciliter le repérage de boîte à livre

Faisable à travers la page carte ou les différents outils de tris pour trouver les boîtes à livre

2.2. Assurer les échanges et la recherche de livres

Ces fonctions seront assurés par le système de requêtes et de notifications lors de la demande ou l'ajout d'un livre dans le réseau de livres

2.3. Développer la structure du réseau et les préférences littéraires

Les Tags (non implémentés) pourront aider à définir les préférences littéraires de chacun et les threads de la page forum

2.4. Encadrer l'organisation d'activités littéraires

non implémenté

3. Exigences

3.1. Besoins Fonctionnelles

· Pouvoir configurer une alerte pour être notifié lorsque X livre est inséré / enlevé dans une boîte à livre du campus

· Pouvoir connaître le contenu des boîtes à livres

· Pouvoir connaître les statistiques sur chaque livres (ratio prêt-temps ou nombre de prêts)

· Pouvoir générer/participer à un thread de discussion sur chaque livre emprunté qui disparaît lorsque aucune transaction n'a été faite sur le livre depuis au moins 60 jours

· Pouvoir inscrire un livre dans l'inventaire d'une boîte à livres en scannant/inscrivant son ISBN

3.2. Besoins non Fonctionnelles

· La plateforme est disponible 24/7

· La plateforme est sécurisée

· Les données des boîtes à livres sont à jour et précises au moins 95% du temps

· L'application est facile d'utilisation pour tout public

· L'application n'est pas très dépendante en ressources pour les téléphones

4. Analyse

4.1. Méthodologie/Approche

Le projet suit une approche agile avec des sprints hebdomadaires. Chaque semaine, les objectifs sont définis et révisés en fonction des progrès réalisés. La planification comprend l'élaboration des exigences, le prototypage, la conception, l'implémentation et la validation.

4.2. Choix technologiques

- **Frontend** : Flutter
- **Backend** : NodeJS avec Fastify
- **Base de données** : MongoDB
- **Langages** : TypeScript
- **Autres technologies** : NFC pour faciliter les échanges, Firebase pour l'authentification

5. Conception

5.1. Architecture haut niveau

L'application est structurée en trois principales composantes : l'interface utilisateur, le serveur backend et la base de données.

5.1.1. Interface utilisateur

- Développée en Flutter pour une compatibilité multiplateforme.
- Comprend des pages pour la carte des boîtes à livres, la recherche de livres, et la gestion des notifications.

5.1.2. Serveur backend

- Implémenté en NodeJS avec Fastify pour gérer les requêtes HTTP et les communications avec la base de données.
- Comprend des API pour la gestion des livres, des utilisateurs, et des statistiques.

5.1.3. Base de données

- MongoDB est utilisé pour stocker les informations sur les livres, les utilisateurs, et les transactions.
- Structure de données flexible pour supporter les modifications futures.

5.2. Architecture bas niveau

5.2.1. Schéma de données

Livre

- ID (unique)
- Titre
- Auteur
- ISBN
- Statut (disponible/emprunté)
- Historique des prêts

Utilisateur

- ID (unique)
- Nom d'utilisateur
- Email
- Historique des emprunts
- Livres favoris

Boîte à livres

- ID (unique)
- Localisation (coordonnées GPS)
- Liste des livres présents

5.2.2. Diagrammes des interactions

Diagramme API

- `getBookfromBookBox` : Récupère un livre par son ID.
- `addExistingBook` : Ajoute un livre à une boîte par son ID.
- `updateEcoImpact` : Met à jour l'impact écologique de l'utilisateur.
- `createThread` : Crée un fil de discussion.
- `loginUser` : Authentifie un utilisateur.

6. Application

6.1. Spécification de l'implémentation

6.1.1. Fonctions principales

- **Recherche de livres** : Filtrage par titre, auteur, genre.
- **Gestion des notifications** : Alertes pour nouveaux livres disponibles.
- **Historique des emprunts** : Suivi des livres empruntés et retournés.
- **Gestion des boîtes à livres** : Mise à jour de l'inventaire.

6.2. Interface utilisateur

Page d'accueil

- Carte des boîtes à livres avec filtres.
- Accès rapide à la recherche de livres.

Page de recherche

- Barre de recherche avec suggestions.
- Filtres avancés pour affiner la recherche.

Page de notification

- Liste des notifications récentes.
- Paramètres de gestion des alertes.

7. Tests

7.1. Stratégie de test

7.1.1. Tests unitaires

Chaque composant du code est testé individuellement pour s'assurer de son bon fonctionnement.

7.1.2. Tests d'intégration

Les interactions entre les différents composants sont vérifiées pour garantir une intégration fluide.

7.1.3. Tests d'acceptation

Les fonctionnalités sont testées dans leur ensemble pour valider qu'elles répondent aux exigences définies.

7.2. Résultats des tests

- **Tests unitaires** : 95% de couverture.
- **Tests d'intégration** : Toutes les API fonctionnent comme prévu.
- **Tests d'acceptation** : 90% des scénarios validés avec succès.

8. Conclusion

8.1. Résumé

Le projet Lino a permis de développer une application mobile innovante pour faciliter l'accès aux livres dans les boîtes à livres du campus de l'Université de Montréal. Grâce à l'utilisation de technologies modernes et une méthodologie de développement agile, l'équipe a réussi à créer une solution qui répond aux besoins des étudiants et du Club de Lecture de l'UdeM.

8.2. Perspectives

- **Améliorations futures** : Intégration de nouvelles fonctionnalités comme les tags littéraires et l'organisation d'événements.
- **Déploiement** : Planification du lancement de l'application sur les stores d'applications.
- **Évaluation continue** : Suivi des retours utilisateurs pour des améliorations continues.

8.3. Remerciements

Nous tenons à remercier notre professeur L-E. Lafontant pour son encadrement tout au long de ce projet. Nous remercions également tous les membres du Club de Lecture de l'UdeM pour leur soutien et leurs suggestions précieuses.