

MÉMOIRE DE FIN D'ÉTUDES POUR L'OBTENTION DU DIPLÔME DE LICENCE EN INFORMATIQUE

PRÉSENTÉ PAR

TAREK-YACINE ATBI & IKRAM MESSADI

Conception d'un système intelligent pour la gestion d'un campus universitaire

Encadrant

Monsieur Aribi Noureddine

Code licence

17/2021

Jury

Monsieur/Madame

Soutenu le

19 juin 2022

2021/2022

REMERCIEMENTS

Nous tenons à remercier toutes les personnes qui ont contribuer de près ou de loin à la réalisation de ce mémoire.

En premier lieu, nous remercions notre encadrant Monsieur Aribi Nourredine, pour nous avoir encouragé dans cette démarche et nous avoir mis à disposition le temps nécessaire à sa réalisation. Nous le remercions pour son engagement et son soutien ainsi que pour la pertinence de ses remarques et de ses feed-back ainsi que son souci du détail, qui ont abouti à la réalisation de ce mémoire.

Nos remerciements vont également aux membres du jury pour avoir accepter d'examiner notre travail et de l'enrichir par leurs propositions.

Nous souhaitons aussi adresser nos remerciements au corps enseignant et administratif de l'université d'Oran 1 qui a contribué à la réussite de nos études universitaires.

RÉSUMÉ

A la lumière du développement et d'importantes avancées technologiques des outils et dispositifs de télécommunication (e.g. Smartphone, Tablet, Réseaux 4G, GPS, ...), l'Intelligence Artificielle (IA) connaît un regain d'intérêt sans précédent, notamment dans le domaine des systèmes de recommandation, impactant ainsi les activités humaines et la relation Humain/-Machine au quotidien. Ces avancées ouvrent de vastes perspectives en termes d'innovation technologique, informatique et d'automatisation dans les différents domaines et activités humaines. Ce travail s'intéresse à l'exploitation de ces évolutions dans le but de construire un système intelligent pour la gestion d'un campus universitaire. L'objectif de ce projet de licence consiste à concevoir et mettre en œuvre un système de gestion intelligente de la vie étudiante dans un campus universitaire. Il consiste à étendre les capacités des systèmes d'information et de gestion classiques actuels et accroître leurs performances avec un système d'aide à la décision innovant, dynamique et évolutif. Ce système intelligent est concrétisé à travers un système de recommandation (meilleurs cours/livres, formations, loisir, ...) ; un système de géolocalisation (la carte du campus, l'ajout et la navigation des points d'intérêts) ; un système temps réel (temps restant de passage de bus, tramway, ...). Le système proposé sera déployé sur un dispositif mobile (Smart-Phone ou Tablet). L'outil développé dans le cadre de ce projet a été testé et validé sur le cas d'étude du Campus de l'université Oran1.

Abstract

Under the light of development, the various important technologies and telecommunication tools (e.g. smartphones, tablets, 4G networks, GPS ...), Artificial intelligence has witnessed a remarkable recovery, particularly in the domain of recommendation systems therefore, impacting the human activities and the daily human/machine relationship. These advancements open vast perspectives in terms of technological innovation, computational as well as automating in various fields alongside human activities. This work's main interest is the exploitation of these evolutions with an aim of constructing an intelligent system for the management of a university campus. The objective of this license degree project is the conception and implementation of an intelligent system for the student's life management within the campus.

It mainly consists of widening the capacities of both the information systems as well as the traditional, classical management besides increasing their performance with an innovative, dynamic decision support system. This sophisticated system is embodied through a recommendation system (best courses/books, training, leisure...) ; a dynamic geolocation system (the campus map/adding and browsing points of interest ...) ; a real time system (remaining time for the next bus, tram car...). The proposed system will be deployed on mobile devices (smartphones or tablets). The following application which was developed under this project has been tested and validated as a study case of the campus of Oran 1 university.

الموجز

تحت ضوء إزدهار وتطور أدوات وتقنيات الإتصال (مثل : الهاتف النقالة، الأجهزة اللوحية تحت ضوء ازدهار وتطور أدوات وتقنيات الإتصال (مثل : الهاتف النقالة، الأجهزة اللوحية الذكية، شبكات بحث، أنظمة تحديد الموضع)، الذكاء الاصطناعي قد اهتماما لم يسبق له مثيل، ولا سيما في مجال أنظمة الترجيح، مؤثرا بذلك على الأنشطة الإنسانية و العلاقة الرابطة بين الإنسان والآلة على الأسس اليومي. هذه التقدمات تفتح منظورات شاسعة فيما يخص الابتكارات التكنولوجية، المعلوماتية والتقنية بالعديد من المجالات.

هذا العمل يهتم باستغلال هذه التطورات بهدف تصميم نظام ذكي لتسهيل حرم جامعي. إن الهدف من مشروع التخرج هذا هو تصميم وتشغيل نظام ذكي لتسهيل الحياة الطلابية داخل الحرم الجامعي. يسعى توسيع نطاق أنظمة المعلوماتية والتسيير المعتادة الكلاسيكية الحالية والرفع من أدائها عن طريق نظام ابتكاري ذكي للإعانة على إتخاذ القرارات. إن هذا الهيكل الذكي يجمع نظام ترجيح (أفضل الدورات الدورات التدريبية) نظام تتبع و تحديد الموقع الجغرافي (خريطة الحرم الجامعي / ميزة تحديد الموضع الجغرافية لأهم المناطق) إضافة إلى أساس توقيت فعلي (الوقت المتبقى للحافلة المقلبة، الترام و ما إلى غير ذلك).

تمت برجمة هذا النظام على هيئة تطبيق قابل للتحميل على الأجهزة المحمولة بما فيها الهاتف و كذا الأجهزة اللوحية الذكية، إضافة أنه قد تم التحقق من صحة هذا النظام من خلال تجربته كنموذج دراسي على الحرم الجامعي لجامعة وهران .

TABLE DES MATIÈRES

| | |
|--|-----------|
| Table des Figures | 7 |
| Liste des tables | 8 |
| 1 Introduction Générale | 1 |
| 2 Applications mobiles multiplateformes et géolocalisation | 2 |
| 2.1 Introduction | 2 |
| 2.2 Types de développement mobile | 2 |
| 2.3 Plateformes mobile multiplateforme ou hybride | 3 |
| 2.3.1 Plateforme Flutter | 4 |
| 2.3.2 Plateforme Ionic | 5 |
| 2.3.3 Plateforme React Native | 6 |
| 2.3.4 Comparaison des Frameworks | 7 |
| 2.4 Tiers Backend des applications multiplateforme | 9 |
| 2.4.1 Langages et environnement d'exécution | 9 |
| 2.4.2 Langages du Web dynamique : Javascript et Typescript | 10 |
| 2.4.3 Environnement d'exécution NodeJs | 10 |
| 2.4.4 Environnement d'exécution Express | 11 |
| 2.5 La géolocalisation | 11 |
| 2.5.1 Définition de la géolocalisation | 11 |
| 2.5.2 Le système GPS | 12 |
| 2.5.3 Techniques de géolocalisation | 13 |
| 2.6 Conclusion | 13 |
| 3 Les systèmes de recommandation | 14 |
| 3.1 Introduction | 14 |
| 3.2 Principe des systèmes de recommandation | 14 |
| 3.3 Filtrage collaboratif | 15 |

| | | |
|----------------------------|---|-----------|
| 3.3.1 | Filtrage basé sur la mémoire | 16 |
| 3.3.2 | Filtrage Basé sur un modèle | 20 |
| 3.4 | Filtrage basé sur le contenu | 20 |
| 3.5 | Filtrage hybride | 21 |
| 3.6 | Conclusion | 22 |
| 4 | Conception et mise en œuvre | 23 |
| 4.1 | Introduction | 23 |
| 4.2 | Cahier de charge | 23 |
| 4.3 | Architecture globale de l'application | 24 |
| 4.3.1 | Les besoins fonctionnels | 24 |
| 4.3.2 | Les besoins non fonctionnels | 25 |
| 4.4 | Modélisation UML | 25 |
| 4.4.1 | Diagramme de cas d'utilisation général | 25 |
| 4.4.2 | Analyse et conception du service " Géolocalisation " | 26 |
| 4.4.2.1 | Diagramme de cas d'utilisation du service de géolocalisation | 27 |
| 4.4.2.2 | Le diagramme de séquence | 27 |
| 4.4.3 | Analyse et conception du service " Bibliothèque " | 28 |
| 4.4.3.1 | Diagramme de cas d'utilisation du service Bibliothèque . . | 28 |
| 4.4.3.2 | Les diagrammes de séquence | 29 |
| 4.5 | Système d'information | 32 |
| 4.5.1 | Modèle conceptuel de données (MCD) | 32 |
| 4.5.2 | Modèle logique de données (MLD) | 32 |
| 4.5.3 | Modèle physique de données (MPD) | 33 |
| 4.5.3.1 | Schéma normalisé de la BDD | 33 |
| 4.5.3.2 | Remplissage de la BDD | 35 |
| 4.5.3.3 | Quelques requêtes et résultats | 35 |
| 4.6 | Réalisation | 37 |
| 4.6.1 | Environnement matériel | 37 |
| 4.6.2 | Environnement logiciel | 38 |
| 4.6.3 | Environnement de programmation | 39 |
| 4.6.4 | Elaboration de l'API | 40 |
| 4.6.5 | Simulation et résultats | 42 |
| 4.6.5.1 | Schéma de navigation de l'application | 42 |
| 4.6.5.2 | Premier cas d'utilisation : authentification | 42 |
| 4.6.5.3 | Deuxième cas d'utilisation : recommandation et réservation d'ouvrage | 44 |
| 4.6.5.4 | Troisième cas d'utilisation : la carte du Campus et la géolo- calisation | 46 |
| 4.7 | Conclusion | 48 |
| Conclusion générale | 49 | |
| Bibliographie | 52 | |

TABLE DES FIGURES

| | | |
|------|---|----|
| 2.1 | Frameworks les plus utilisés par les développeurs mobiles de 2019 à 2021 [9] | 4 |
| 2.2 | Architecture de Flutter [3] | 5 |
| 2.3 | Architecture Apache Cordova [5] | 6 |
| 2.4 | Architecture de React Native [6] | 7 |
| 2.5 | Architecture Flutter et le serveur en NodeJs | 9 |
| 2.6 | Représentation graphique de la liaison entre Typescript et Javascript | 10 |
| 2.7 | Architecture de l'environnement d'exécution NodeJs | 11 |
| 2.8 | Principe de fonctionnement du système GPS | 12 |
| 2.9 | Géolocalisation <i>Indoor</i> , <i>Outdoor</i> et <i>semi-outdoor</i> | 13 |
| 3.1 | Les types des systèmes de recommandation | 15 |
| 3.2 | Exemple illustrant l'extraction des éléments pertinents dans le scénario <i>item-based</i> à l'aide du Filtrage Collaboratif. | 17 |
| 3.3 | Organigramme d'un processus de filtrage collaboratif basé sur l'utilisateur . . | 20 |
| 3.4 | Le processus de la recherche d'information dans un système CBF. | 21 |
| 3.5 | Une abstraction d'un système basé sur le filtrage hybride. | 22 |
| 4.1 | Architecture globale de l'application | 24 |
| 4.2 | Diagramme de cas d'utilisation général | 26 |
| 4.3 | Diagramme de cas d'utilisation du service géolocalisation | 27 |
| 4.4 | Diagramme de séquence du service de géolocalisation | 28 |
| 4.5 | Diagramme de cas d'utilisation du service bibliothèque | 29 |
| 4.6 | Diagramme de séquence du service Bibliothèque : partie <i>Système de recommandation</i> | 30 |
| 4.7 | Diagramme de séquence du service Bibliothèque : partie <i>Gestion des ouvrages</i> | 31 |
| 4.8 | Le Modèle conceptuel de données du projet | 32 |
| 4.9 | Script SQL permettant la création de la table Étudiant (<i>Student</i>) | 34 |
| 4.10 | Script SQL permettant la création de la table Ouvrage (<i>Textbook</i>) | 34 |
| 4.11 | Script SQL permettant la création de la table Point d'interet (<i>Point of Interest</i>) | 35 |

| | |
|---|----|
| 4.12 Script SQL permettant l'insertion de deux points d'intérêts dans la table <i>Point d'intérêt</i> . | 35 |
| 4.13 La première requête SQL | 36 |
| 4.14 Le résultat d'exécution de la 1ère requête | 36 |
| 4.15 La deuxième requête SQL | 36 |
| 4.16 Le résultat d'exécution de la 2ème requête | 37 |
| 4.17 La troisième requête SQL | 37 |
| 4.18 Le résultat d'exécution de la 3ème requête | 37 |
| 4.19 Structure de notre API | 40 |
| 4.20 L'objet de la réponse en JSON | 41 |
| 4.21 Structure de notre projet Flutter | 41 |
| 4.22 Schéma de navigation de notre application | 42 |
| 4.23 Interface 'Connexion' et 'Inscription' | 43 |
| 4.24 Corps de la méthode connectUser() | 43 |
| 4.25 Gestion d'authentification par le serveur | 44 |
| 4.26 La fonction derrière le filtrage collaboratif. | 45 |
| 4.27 Interfaces du service Bibliothèque | 45 |
| 4.28 Guide de géolocalisation manuelle envoyé par le département. | 46 |
| 4.29 Photos envoyé par le département. | 46 |
| 4.30 Interfaces du service de géolocalisation | 47 |
| 4.31 Le code derrière l'interface du service de géolocalisation | 48 |

LISTE DES TABLEAUX

| | | |
|-----|---|----|
| 2.1 | Résumé des Frameworks [7, 8] | 8 |
| 3.1 | Échantillon des évaluations des livres par cinq utilisateurs | 18 |
| 3.2 | Résultat de calcul de similarité en utilisant la corrélation de Pearson | 19 |
| 3.3 | Résultat de calcul de similarité en utilisant la distance Euclidienne | 19 |
| 3.4 | Résultat de calcul de similarité en utilisant l'indice de Jaccard | 19 |
| 4.1 | Acteurs et leurs rôles | 25 |

ABRÉVIATIONS

POI Point Of Interest.

SR Système de Recommandation.

API Application Programming Interface.

CBF Content Based Filtering.

CF Collaborative Filtering.

SQL Standard Query Language.

URI Uniform Ressource Identifier

CHAPITRE 1

INTRODUCTION GÉNÉRALE

Au cours des dernières années, de plus en plus de personnes possèdent un Smartphone ou encore une tablette. Le développement mobile est un phénomène qui a pris beaucoup d'ampleur en quelques années.

Aujourd’hui, il n’y a plus de doute possible sur l’importance d’avoir un site accessible par mobile, surtout que de nos jours les moyens de communication les plus utilisés sont les Smartphones. Tout cela nous a motivé à développer une application mobile pour la gestion d’un campus universitaire réalisé au sein du département d’informatique dans le cadre de projet de fin d’étude de Licence Informatique spécialité Système d’information.

Au début de cette année universitaire 2021/2022, nous avons pu constater beaucoup de problèmes qui concernent l’orientation au sein de notre campus, beaucoup d’étudiants ignorent les emplacements exactes de plusieurs lieux et points d’intérêt de notre campus ainsi que les invités qui se déplacent pour assister ou participer à des événements (e.g. concours, séminaires, soutenances, conférences, portes ouvertes, etc.). Un autre problème qu’on a pu identifier est l’emprunt de livre sans avoir une connaissance au préalable de son contenu ni des avis des ex-lecteurs de ce livre.

Donc, un besoin est apparu qui consiste à faciliter la localisation des points d’intérêt à l’intérieur du campus ainsi que mémoriser et garder les évaluations des lecteurs après chaque emprunt d’ouvrage afin de faciliter la recommandation.

Au final, l’objectif de ce travail est de développer une application mobile multi-plateforme (pour que tous les étudiants puissent l’utiliser, quelque soit le système installé sur leurs mobiles) qui gère un campus universitaire tout en donnant l’accès aux étudiants locaux et en organisant ses services. L’axe principal est la mise en œuvre d’une architecture optimisée de géolocalisation à l’intérieur d’un campus universitaire pour les étudiants, personnels et invités en plus d’un système de recommandations de livres ou ouvrages dédiés aux étudiants et chercheurs.

CHAPITRE 2

APPLICATIONS MOBILES MULTIPLATEFORMES ET GÉOLOCALISATION

2.1 Introduction

Le marché du mobile a connu une croissance très rapide ces dernières années. Depuis 2015, les utilisateurs naviguent davantage sur les *Smartphones* ou les tablettes que sur les ordinateurs car on n'utilise plus les *Smartphones* que pour passer des appels et les géants du mobile et les logiciels informatiques le savent déjà. De tout cela, le plus motivant est que les applications mobiles se multiplient d'année en année dans tous les domaines.

Pour rendre le système plus efficace et efficient, plusieurs types d'applications sont disponibles (application de bureau, Web, mobile) dans ce qui suit on présente un aperçu plus ou moins approfondi sur les applications mobiles ainsi qu'au Framework Flutter.

2.2 Types de développement mobile

Dans le domaine du *Smartphone*, plusieurs types d'applications mobiles existent. Selon l'usage auquel elle est destinée, une application mobile peut être développée pour réaliser une fonction précise ou à fournir une fonctionnalité réduite (ou étendant ou dupliquant) d'un site web ou encore avoir un mélange de fonctionnalités des précédentes. La création d'une application mobile mène à beaucoup de questions concernant le développement, la fonctionnalité et l'ergonomie. Ainsi, avant de se lancer dans le développement d'une application mobile, il est indispensable de bien choisir le type d'application à développer et de bien argumenter ce choix [1].

1. **Application native** : Une application native est une application mobile qui fait l'objet de développements distinct pour chacun des systèmes d'exploitation utilisé (iOS, Android, etc.), et ainsi elle fonctionne de façon optimale avec tous les types d'appareils compatibles. Elle a un développement plus complexe que les autres car

les langages de programmation sont différents selon les systèmes d'exploitation. Par exemple, les applications pour l'Apple Store sont développées soit en Objective-C ou en Swift mais les applications pour Google Play sont développées en Java et Kotlin. Les applications Windows sont développées en C [2].

2. **Application web mobile :** Les applications mobiles web sont une version réduite des applications web dédiées pour les appareils mobiles. L'accès à ce genre d'application est actionné par un navigateur web. Par conséquent, ces applications sont indépendantes des plateformes et caractéristiques des appareils mobiles. Nécessairement, ces applications exigent une connexion internet pour leurs utilisations [1].
3. **Application hybride :** Entre l'application native et l'application web, on trouve l'application multiplateforme (en anglais *Cross-platform*) ou hybride. Comparées aux applications natives les applications hybrides sont moins chères et plus rapides à développer.

Une application hybride a une conception plus simple et rapide que les applications natives car seulement une version du code est déployée pour les différents systèmes d'exploitation (IOS, Android, Windows). Lors du développement de l'application un Software Développement Kit (SDK) est utilisé pour avoir accès à une librairie d'ensemble de briques pré-construites adaptées aux différents systèmes d'exploitation et permettant la transposition en code binaire (langage natif). Les développements en « cross-platform » ont l'avantage de réduire la maintenance et de faciliter l'ajout de fonctionnalités car il n'y a qu'une seule version du code [2].

2.3 Plateformes mobile multiplateforme ou hybride

À la différence des implémentations natives, les Framework multiplateformes génèrent des applications compatibles avec plusieurs plateformes. La quasi-totalité de ces outils prennent en compte à la fois le développement pour Android et iOS. Les principaux outils multiplateformes en ce moment sont Ionic, Flutter et React Native (voir figure 2.1). On donne ci-après les détails sur ces plateformes hybrides.

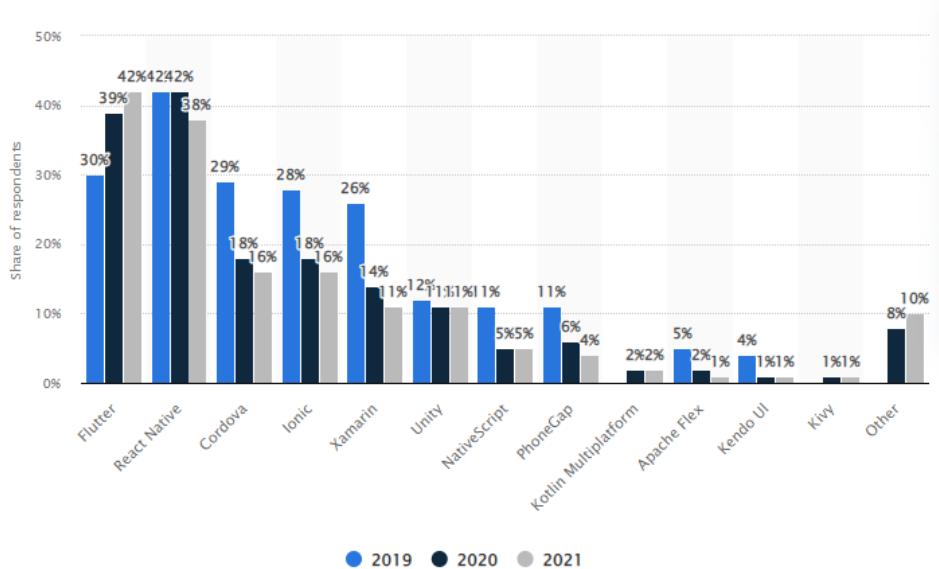


FIGURE 2.1 – Frameworks les plus utilisés par les développeurs mobiles de 2019 à 2021 [9]

2.3.1 Plateforme Flutter

La première version de Flutter date de décembre 2018, c'est un framework développé par Google. Ce framework permet, tout comme React Native, de développer des applications mobiles pour différentes plateformes comme Android, iOS, Windows ou MacOS. Flutter malgré son jeune âge est utilisé par de grands groupes internationaux comme : Alibaba, BMW, eBay ou encore Tencent.

Les widgets sont au cœur de ce framework. Un widget est n'importe quel composant de l'application. Un widget peut représenter un bouton, mais un widget permet aussi de styliser une interface en permettant de centrer un élément. Un widget peut aussi être le composant qui permet le défilement d'une liste. Par exemple, pour centrer un bouton, le widget « bouton » devra être un enfant du widget « centrer » [3].

- Architecture :** L'architecture de Flutter est en trois couches. La première couche est appelée « Dart Framework », c'est simplement le code de l'application écrite en langage Dart. La deuxième couche est appelée « Engine », elle permet de faire le lien entre le code écrit par le développeur et l'appareil mobile. Enfin, la troisième couche est appelée « Embedder », elle permet de faire, comme son nom l'indique, l'intégration de la deuxième couche, en faisant la connexion avec le système natif de l'appareil mobile (voir figure 2.2).

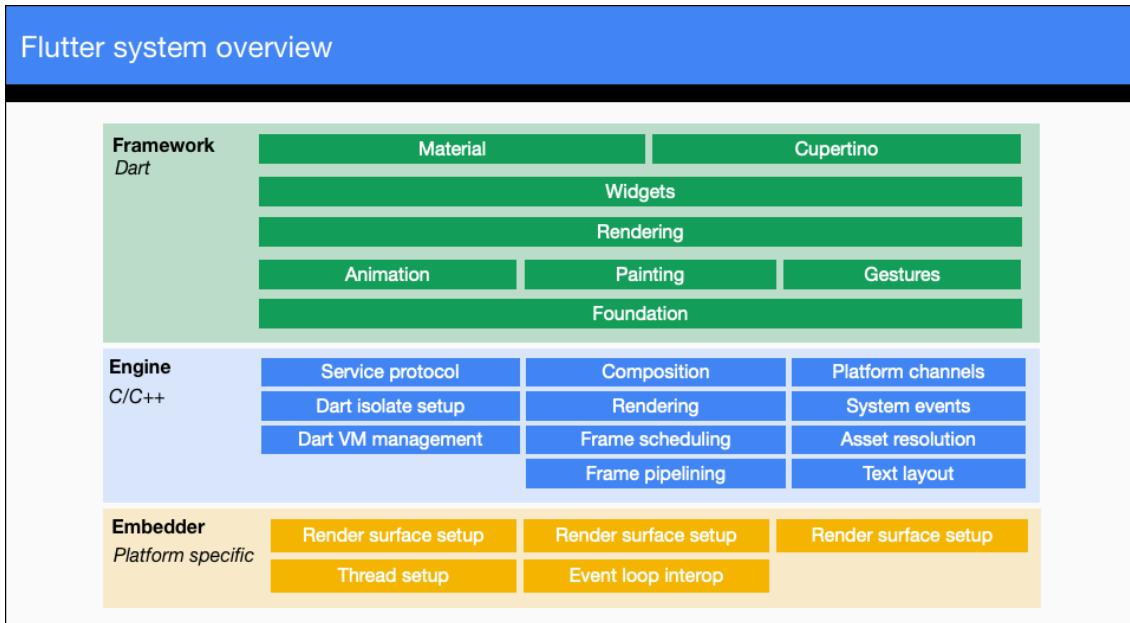


FIGURE 2.2 – Architecture de Flutter [3]

2. **Dart :** Dart est un langage de programmation créé par Google. C'est un langage qui permet de créer des applications sur n'importe quelle plateforme. Ce langage est optimisé et spécialisé dans la création d'interfaces utilisateur. Dart est un langage qui permet de créer des fonctions asynchrone qui permettent donc de ne pas bloquer le rendu de l'application. Contrairement à d'autres langages qui offrent la possibilité d'exécuter des processus en parallèle, Dart propose l'utilisation d'« isolates ». Un Isolates est un peu comme un thread à la différence que sa mémoire n'est pas partagée avec les autres Isolates, c'est d'ailleurs pour cela qu'il porte ce nom. Deux Isolates peuvent quand même se passer des informations via des messages et lorsqu'un Isolate reçoit un message, il peut exécuter une certaine portion de code [3].

2.3.2 Plateforme Ionic

Le Framework multiplateforme Ionic a été élaboré avec Apache Cordova et Angular. Avec cet outil, les applications mobiles développées sont utilisables sur plus d'une plateforme mobile. Elles fonctionnent sur Androïd et iOS. La conception de ces applications est classique. Elle se fait à l'aide de HTML, d'Angular ou de CSS. En tant qu'outil multiplateforme, Ionic permet aux entreprises de développer leur application mobile avec un minimum de dépenses. L'entreprise n'a pas besoin de faire appel à plusieurs ingénieurs logiciels pour la conception d'une application mobile [4].

1. **Architecture :** Comme décrit précédemment, Ionic est une surcouche du Framework Angular qui, lui-même, utilise le Framework Apache Cordova pour la création de l'APK¹(*Android Package Kit*). Le fonctionnement de ces Frameworks vont être expliqués ci-dessous.

1. APK est un format de fichiers pour le système d'exploitation Android.

2. Apache Cordova : La création de l'APK et le test de l'application sur le téléphone mobile sont une étape essentielle du développement. Il existe plusieurs façons de créer un APK. Apache Cordova propose d'encapsuler facilement l'application dans un paquetage lisible par les systèmes d'exploitation mobiles et offre des accès aux composants natifs du téléphone (GPS, appareil photo, accéléromètre, etc.) via une collection de plugins intégrés à Apache Cordova (voir figure 2.3) [5].

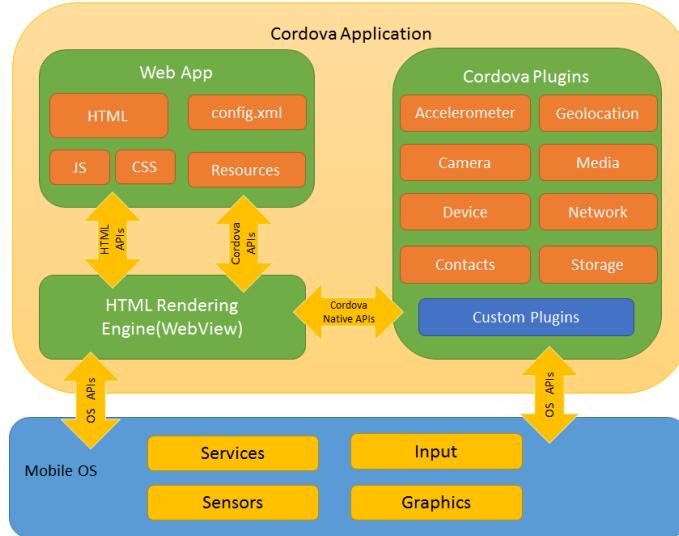


FIGURE 2.3 – Architecture Apache Cordova [5]

3. Angular : Angular est un Framework offrant la possibilité de créer des pages WEB, des applications mobiles de manière simplifiée grâce à des systèmes de déclaration de modèles, d'injection de dépendances et de binding de données. Angular va relier de manière simplifiée les données de la partie logique à l'affichage et inversement.

2.3.3 Plateforme React Native

Framework créé par Facebook, React Native utilise comme langage de programmation JavaScript (et est couplé à l'implémentation mobile du Framework ReactJS). Il se distingue des applications hybrides par le fait qu'il emploie les composants natifs du téléphone. Le JavaScript est langage souple et rapidement assimilable. Par sa syntaxe, il permet une grande diversité dans l'écriture. L'orientation composante du Framework permet également de réutiliser différentes parties du code sans avoir à les réécrire [8].

1. Architecture : L'architecture de React Native comprend trois modules : le noyau, l'application mobile et l'application Web. Le noyau est exécuté à l'aide du Framework Redux et contient la logique métier et l'état de la demande. Ce module est responsable de la réutilisation du code qui est annexé aux applications mobiles et Web en tant que sous-module git (voir figure 2.4) [6].

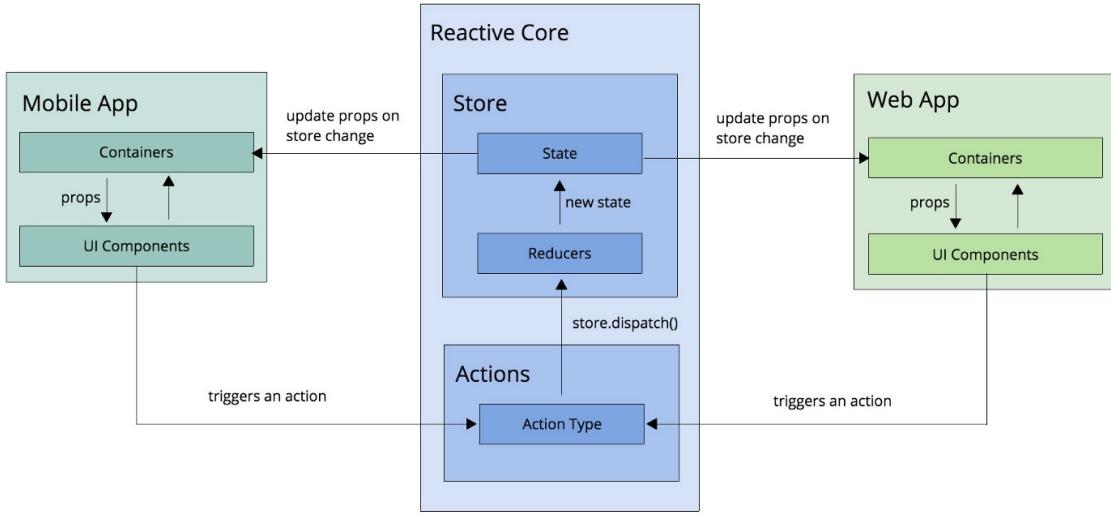


FIGURE 2.4 – Architecture de React Native [6]

2.3.4 Comparaison des Frameworks

Il n'y a pas de framework parfait et chaque framework a des avantages et inconvénients. Choisissez celui qui convient le mieux en fonction des besoins du développeur. Par exemple, Ionic est facile à apprendre et fournit une documentation facile à utiliser, et couvre la plupart des besoins d'un développeur. Cependant, les performances et la sécurité de l'application ne sont pas aussi bonnes que l'application mobile d'origine. De plus, React Native est connu pour ses performances idéales et son fonctionnement rapide par rapport aux applications Ionic. Son défaut fondamental est qu'il est considéré comme une faille de sécurité car il est open source.

En outre, Flutter se démarque de ses concurrents (React Native, NativeScript, Ionic) par ses performances et offre aussi les meilleures interfaces utilisateur néanmoins Flutter est un nouveau Framework pour la communauté en ce moment et pas très populaire. Mais, il est intensément annoncé par Google qui montre qu'ils veulent en faire une chose majeure dans le monde mobile. Le tableau ci-dessous (tableau 2.1) représente une comparaison de ces trois frameworks :

| | Ionic | Flutter | React Native |
|--------------------------------|---|--|--|
| Année de sortie | 2013 | 2018 | 2015 |
| Créateur | Drifty.co | Google | Facebook |
| Langage | Technologie Web | Dart | JavaScript |
| Performance | Ses performances ne sont pas aussi similaires à celles des React Native ou Flutter | Flutter est le plus performants par rapport à ses concurrents. | Les performances qu'il fournit sont très similaires aux applications natives |
| Communauté | C'est le deuxième Framework le plus populaire après React | Flutter est un nouveau Framework pour la communauté en ce moment et pas très populaire | Le Framework le plus populaire |
| Plateformes | iOS, Android et PWA | iOS, Android, PWA et UWP | Android, iOS et UWP |
| Interface utilisateur | Ionic interface utilisateur n'utilise pas du tout d'éléments natifs et rend tout en HTML et CSS | Il fournit les meilleures interfaces utilisateur | Ses modules s'associent aux contrôleurs d'interface utilisateur natifs |
| Applications populaires | JustWatch, Pacifica, Nation wide | Hamilton, Alibaba, BMW | Facebook, Instagram, Airbnb, UberEats |

TABLE 2.1 – Résumé des Frameworks [7, 8]

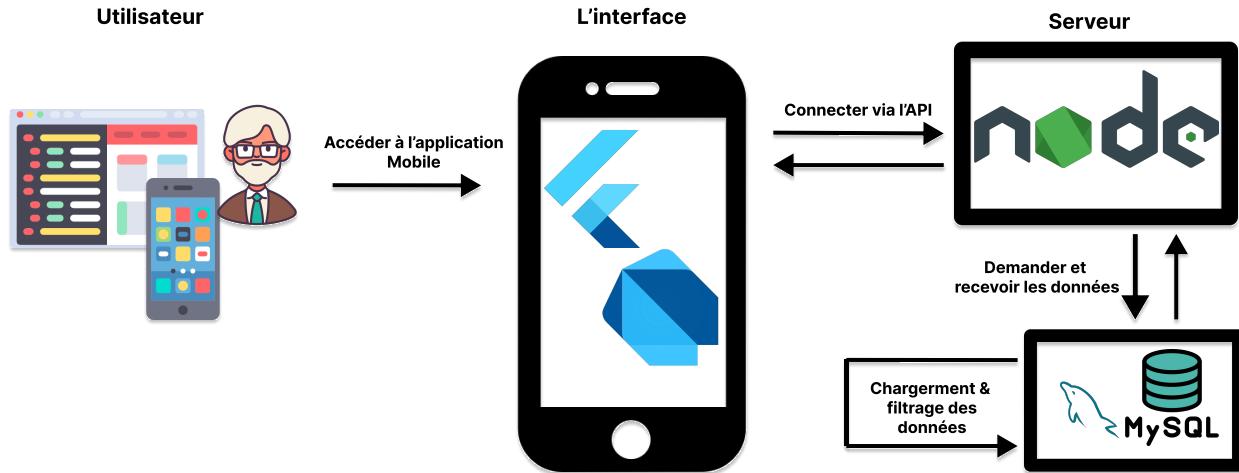


FIGURE 2.5 – Architecture Flutter et le serveur en NodeJs

2.4 Tiers Backend des applications multiplateforme

En général, les applications mobiles se composent de deux tiers. Les utilisateurs peuvent accéder au frontend tandis que la zone invisible pour l'utilisateur est le backend ou autrement référencé par "le développement côté serveur". Il s'agit d'un type de développement informatique qui se focalise sur l'architecture métier d'une application, les scripts et tout code exécutés par le serveur. En d'autres termes, tout processus arrière-plan nécessaire qui est exécuté côté serveur d'une application web ou mobile. La figure 2.5 illustre la communication entre un serveur en Nodejs et une interface mobile en Flutter.

Le terme backend est principalement utilisé en raison de la non-visibilité du code et son exécution par rapport à l'interface présentée aux utilisateurs. Le développement backend englobe d'innombrables services rudimentaires qui représentent la base solide pour n'importe quelle application tel qu'un SGBD. Le développement backend est principalement basé sur la logique d'un modèle client-serveur.

2.4.1 Langages et environnement d'exécution

Il existe une variété des langages de développement coté serveur qui permettent la programmation explicite du traitement effectué par un serveur ainsi que son interaction avec le système de gestion de base de données. Certains de ces langages sont Python, Java, PHP et Javascript. D'une autre part, les Frameworks représentent une boîte à outils ou un *toolbox* écrits dans un langage de programmation serveur spécifique. Cette boîte à outils permet, à la fois, la création rapide et robuste de la logique d'une application web ou mobile. Nul doute qu'un développement avec un framework évite de réécrire des fonctionnalités essentielles et d'exporter plus facilement les programmes pour différentes utilisations. Certaines frameworks du développement backend sont : *Flask*, *Laravel*, *Django* et *ExpressJs*.

Pour le développement du backend de notre système, on a choisi Typescript comme lan-

gage côté serveur, Nodejs pour l'environnement d'exécution de base et ExpressJs comme framework.

2.4.2 Langages du Web dynamique : Javascript et Typescript

Javascript est le premier langage de script pour le web, orienté objet, léger, interprété et conforme aux spécifications ECMAScript.

Traditionnellement utilisé comme outil de développement côté client, il est également devenu un outil majeur de développement mobile multiplateforme comme une technologie de base pour un grand nombre des plates-formes, telles que Apache Cordova/PhoneGap et React Native. Par ailleurs, Typescript est un langage de programmation orienté objet qui apporte une encapsulation de la version Javascript conforme au ECMAScript 5 (voir la figure 2.6). Il est développé par Microsoft, Il est basé sur un compilateur qui compile le code Typescript en Javascript [29].

Destinée à faciliter le développement d'applications JavaScript à grande échelle, TypeScript offre un système de modules, des classes, des interfaces et un système de typage riche.

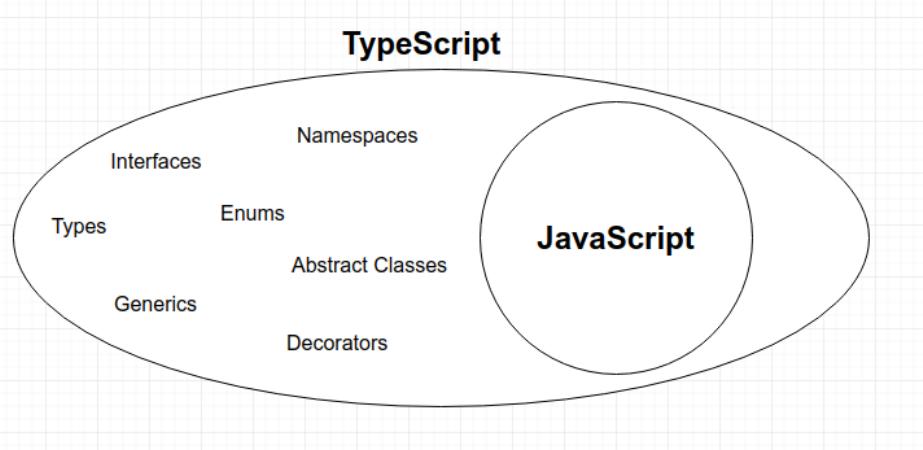


FIGURE 2.6 – Représentation graphique de la liaison entre Typescript et Javascript

2.4.3 Environnement d'exécution NodeJs

NodeJs est un environnement d'exécution mono thread, open source et multiplateforme, qui permet l'exécution du code Javascript côté serveur. Il est écrit en C, C++ et Javascript. NodeJs est basé sur une architecture E/S non bloquante. Il s'appuie sur un système de boucles d'événement *Event loop* qui le rend robuste, efficace et lui permet de supporter une charge volumineuse des requêtes simultanées. Grâce à son architecture asynchrone, cet environnement d'exécution permet d'éviter les attentes en se basant sur un mécanisme puissant. Tout d'abord, les nouvelles requêtes sont placées dans une file d'attente qui est gérée éventuellement par la boucle d'événement. La boucle filtre les requêtes et lance seulement celles qui sont bloquantes parallèlement dans un seul pool de thread appelé le *worker*. Simultanément, elle traite les requêtes non bloquantes restantes en leur envoyant une réponse. La

figure -2.7- résume cette architecture.

Parce qu'il utilise peu de thread, Nodejs consomme moins de ressource, ce qui en fait un environnement d'exécution rapide et efficace. Il permet également l'intégration d'un vaste ensemble des packages grâce à son écosystème NPM.

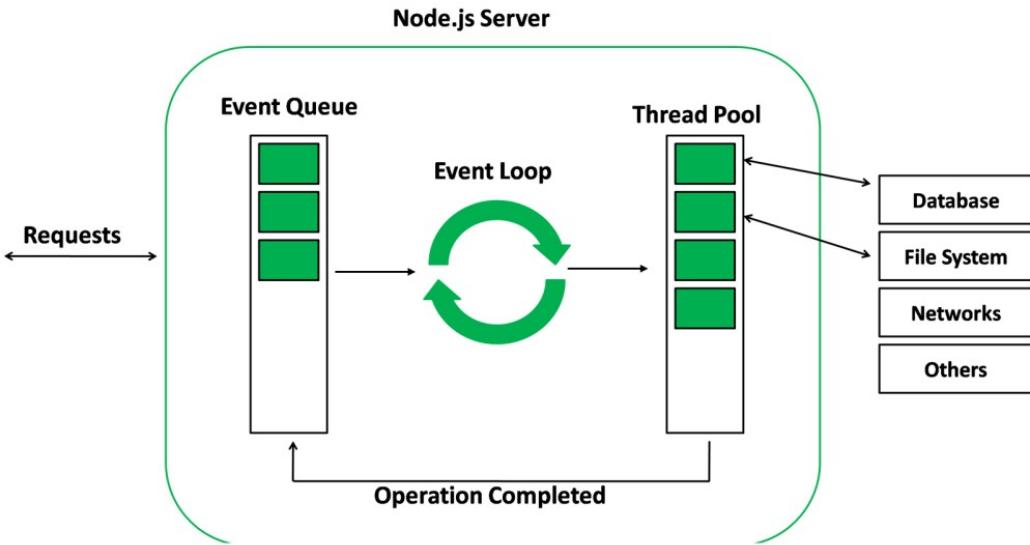


FIGURE 2.7 – Architecture de l'environnement d'exécution NodeJs

2.4.4 Environnement d'exécution Express

Express est un framework minimaliste, flexible et open source basé sur l'architecture MVC (*Model-View-Controller*). De plus, il est le framework standard dans le développement serveur en NodeJs. Il offre une vaste collection des fonctionnalités inévitables pour la création d'un serveur comme les fonctionnalités de routage, caching et rendering. Non seulement la flexibilité du framework ExpressJs ouvert la porte aux développeurs d'intégrer de nombreuses d'autre bibliothèques selon le besoin et de choisir d'autres architectures comme backend, mais il facilite la procédure de débogage grâce à son module *debug* qui trace explicitement les points d'erreurs d'une exécution.

2.5 La géolocalisation

2.5.1 Définition de la géolocalisation

La géolocalisation est un procédé permettant de positionner un objet, un véhicule, ou une personne sur un plan ou une carte à l'aide de ses coordonnées géographiques. Cependant, il est intéressant de pouvoir accéder à ces cartes lors de l'utilisation d'applications spécifiques avec nos smartphones grâce au système de positionnement global GPS (*Global Positioning System*) embarqué.

2.5.2 Le système GPS

Le système GPS est un système basé sur une constellation de 24 satellites d'observation dans la terre émettant en permanence des signaux datés vers le globe et sur un réseau de stations de surveillance de ces satellites au sol le GPS a pour rôle de fournir via un récepteur approprié des données spatiales temporelle (latitude, longitude, élévation,...).

Récepteur GPS. Ces appareils spécialisés sont conçus pour recevoir et analyser les signaux transmis par les réseaux de satellites afin de déterminer une géolocalisation.

Il existe différents **types de récepteurs GPS.** Certains sont de simples récepteurs qui se limitent à produire des données de géolocalisation transmises ensuite à d'autres appareils. D'autres intègrent un écran et peuvent positionner l'utilisateur sur une *carte numérique*, ou être exploités en lien avec d'autres appareils comme un *système de pilotage automatique* dans le monde maritime ou de calcul d'itinéraire routier. D'autres appareils électroniques beaucoup plus courants intègrent un récepteur GPS, c'est notamment le cas de la plupart des *smartphones* récents.

Il est intéressant de noter que la vitesse de transmission des ondes radio étant **constante**, le délai de réception du signal est donc proportionnel à la distance qui sépare le récepteur de chaque satellite émetteur.

Les Réseaux de Satellites. Il est possible de déterminer une position géographique du récepteur en croisant les données issues d'au moins trois satellites (voir la figure 2.8) et idéalement de quatre pour une meilleure précision. Des techniques de correction et de filtrage permettent de compenser les nombreux paramètres qui peuvent influer sur la transmission et la réception des signaux. On comprend qu'un système de géolocalisation nécessite un nombre important de satellites pour pouvoir couvrir la surface du globe avec une qualité suffisante et un minimum de redondance pour pouvoir pallier d'éventuels dysfonctionnements. Les deux principales constellations de satellites de géolocalisation sont l'américaine **Navstar GPS** et l'europeenne **Galileo** [32].

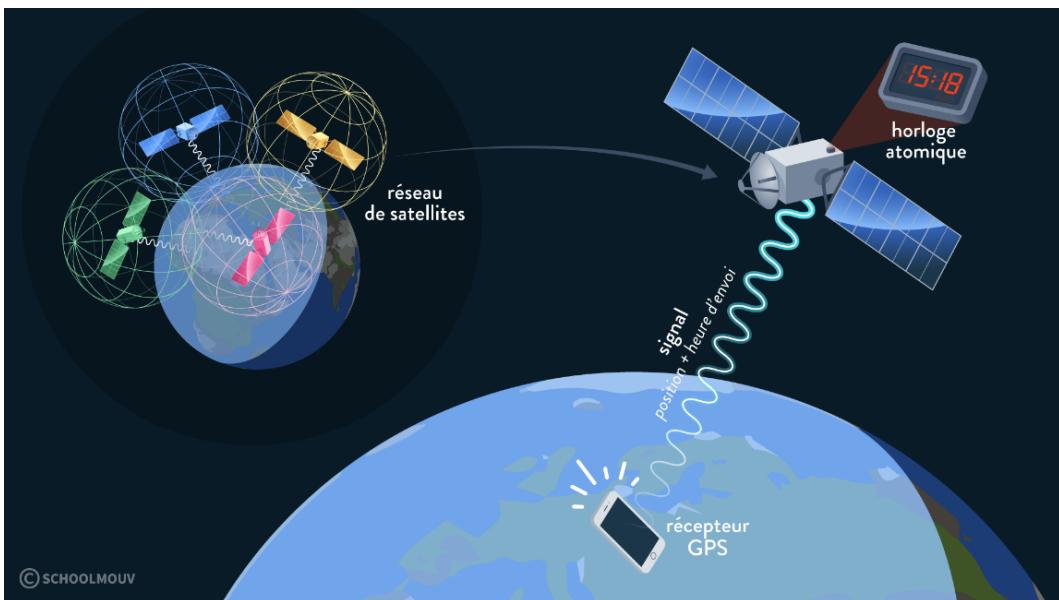


FIGURE 2.8 – Principe de fonctionnement du système GPS

2.5.3 Techniques de géolocalisation

On distingue deux principaux types de géolocalisation :

Géolocalisation Outdoor. Cette technologie repose sur un système de localisation de type GPS, embarqué sur le mobile pour permettre d'en déterminer la position et de la transmettre selon un certain mode de communication (SMS, GSM, satellite...). Cette information sera ensuite acheminée à un centre localisé, au sein duquel l'ensemble des informations seront traitées et pourront être consultées.

Géolocalisation Indoor. Dans le cadre de bâtiments, la solution GPS n'est pas la plus adaptée ne permettant pas une précision suffisante puisque sa marge d'erreur est d'environ 10 mètres, il apparaît difficile d'apprécier le service. On trouve en l'occurrence une véritable solution avec une localisation par Wi-fi, Bluetooth et RFID. Les deux premiers sont clairement les plus intéressants sur les supports mobiles. Ils sont compatibles avec 100% des smartphones et la plupart des tablettes. Le bluetooth apporte notamment une précision utile pour se localiser avec précision en indoor (figure 2.9).

| Environnement | Outdoor | Semi-outdoor | Indoor |
|---------------|---------------------------|--------------------|---------------------------|
| Definition | l'extérieur d'un bâtiment | près d'un bâtiment | l'intérieur d'un bâtiment |
| Exemple | | | |
| Scene | | | |

FIGURE 2.9 – Géolocalisation *Indoor*, *Outdoor* et *semi-outdoor*

2.6 Conclusion

Ce chapitre a présenté les principaux frameworks de développement des applications mobiles multiplateformes, suivie d'une comparaison entre les frameworks qui dominent actuellement le marché. Ensuite, il a brièvement abordé les technologies du développement côté serveur (Backend ou Back-office) en se focalisant surtout sur l'environnement d'exécution NodeJs.

Notre principale motivation derrière le choix du framework Flutter se résume à sa riche documentation et la grande communauté qui l'utilise, et qui reste en croissance continue. De plus, Flutter offre des performances quasi natives et assure une expérience utilisateur cohérente sur tous les systèmes d'exploitation supportés par les smartphones actuels.

CHAPITRE 3

LES SYSTÈMES DE RECOMMANDATION

3.1 Introduction

L'arrivée du WEB 2.0 a bouleversé la relation entre les producteurs et les consommateurs en plusieurs niveaux. Par donner l'accès à tous le monde de partager, créer et ajouter son propre contenu, les informations du Web ont connu une croissance exponentielle qui a conduit à l'apparition de divers services e-business et e-commerce. Ainsi que l'émergence des plateformes de réseaux sociaux tels que Facebook qui ont principalement encouragé l'utilisateur à partager, recommander et coopérer des informations sur leurs activités, goûts, intérêts et préférences. Cela n'a donné lieu qu'à une surabondance d'informations liées à un objet[13].

À notre époque de technologie et de Big Data, le système de recommandation est utilisé pour résoudre les problèmes liés à la surcharge d'informations sur le Web. De plus, il constitue un moyen de comprendre le profil d'un utilisateur et de lui suggérer des recommandations appropriées pour ses prochaines interactions avec le système, tout en tenant compte par exemple de son historique de la navigation, son contexte des achats ou un feedback implicite ou explicite [12].

Ce présent chapitre présente le principe derrière les systèmes de recommandations, leurs différents types ainsi que le mécanisme suivi par chaque approche.

3.2 Principe des systèmes de recommandation

Depuis leur naissance en 1992, les systèmes de recommandation (SR) ont été exploités dans plusieurs domaines comme l'exploitation des usages sur le web, la recherche d'informations et également le e-commerce [12]. Une implémentation populaire d'un SR est par Amazon pour la suggestion des articles similaires, Netflix et Youtube.

Un système de recommandation est une approche qui s'adresse spécifiquement au problème de la suggestion des objets appropriés à l'utilisateur par rapport à son historique et ses relations

avec d'autres utilisateurs éventuellement. Ils génèrent généralement un certain nombre de suggestions tout en basant sur des techniques d'intelligence artificielle.

Les systèmes de recommandation utilisent principalement deux méthodes de filtrage pour proposer des recommandations personnalisées aux utilisateurs, à savoir le filtrage collaboratif et le filtrage basé sur le contenu (voir la figure 3.1). Le premier utilise les opinions des utilisateurs similaires à l'utilisateur actif. Le deuxième utilise uniquement les préférences de l'utilisateur actif. Le filtrage collaboratif est considéré comme la méthode la plus populaire et la plus répandue dans les systèmes de recommandation [33].

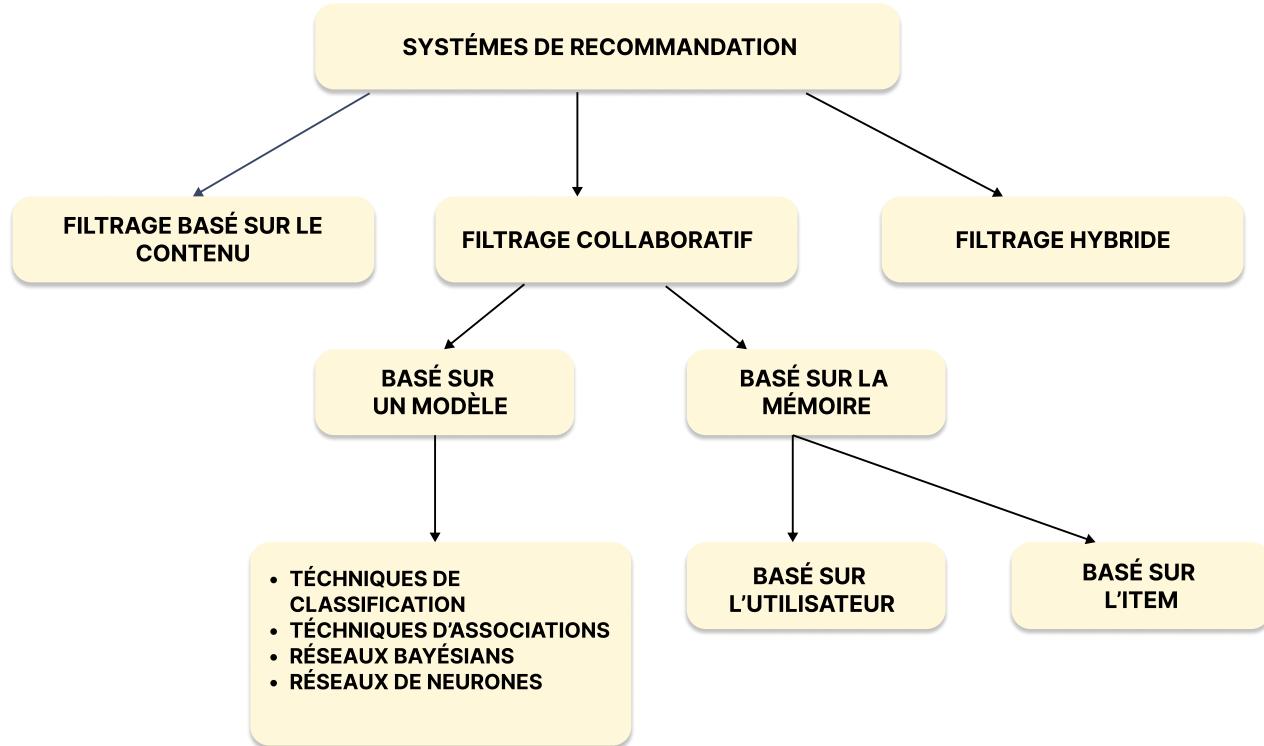


FIGURE 3.1 – Les types des systèmes de recommandation

Parmi les techniques utilisés par les systèmes de recommandation, nous allons citer les techniques les plus implémentées. On commence par le Filtrage Collaboratif.

3.3 Filtrage collaboratif

Les systèmes de recommandation collaboratifs associent au profil de l'utilisateur cible une recommandation en se basant sur les profils d'autres utilisateurs ayant utilisé le même système mais avec des préférences similaires. Si d'autres utilisateurs ont évalué un ensemble d'éléments de la même façon que l'utilisateur cible, le système de recommandation prédit que l'utilisateur cible pourrait également aimer de nouveaux éléments qui n'ont pas encore été vus, mais que des utilisateurs similaires ont aimé. L'hypothèse sous-jacente dans le filtrage collaboratif est que les notes fournies par les utilisateurs représentent des opinions assez constantes qui

peuvent être analysées pour fournir une estimation raisonnable de la préférence de l'utilisateur cible sur un item qu'il n'a pas encore noté [13].

Les systèmes de filtrage collaboratif regroupent les notes, reconnaissent les similitudes entre les personnes et produisent des recommandations fondées sur des comparaisons entre les utilisateurs. Il se divise en deux catégories, le filtrage collaboratif basé sur mémoire et celui basé sur un modèle.

3.3.1 Filtrage basé sur la mémoire

Cette méthode est dite *memory-based* ou heuristique car, toute note laissée par les utilisateurs est conservée par le système et elle est éventuellement utilisée pour déterminer la similarité entre eux et l'utilisateur cible c'est pour cela qu'elle est nommée *Word Of Mouth* ou "bouche à l'oreille". Également, une note peut être une appréciation d'un item, un centre d'intérêt ou un contexte de recherche.

Il existe deux approches qui peuvent être utilisées pour tirer des recommandations en se basant sur un système de filtrage collaboratif à base de mémoire. Ces deux approches sont mathématiquement assez similaires. Néanmoins, il existe une différence conceptuelle entre les deux :

1. **Approche basé sur l'utilisateur** : Dans ce cas, les *ratings* fournis par les utilisateurs similaires à l'utilisateur cible sont utilisés pour faire des recommandations pour l'utilisateur u . La prédiction est calculée comme la moyenne pondérée des *ratings* pour l'item i fournis par le voisinage de l'utilisateur u [13].
2. **Approche basé sur les items** : Pour un item i , le vote par un utilisateur u qui ne l'a pas encore voté est trouvé en prenant les I items les plus similaires qui ont été noté par l'utilisateur u en calculant le vote basé sur les votes de ces I items [13] (voir la figure 3.2).

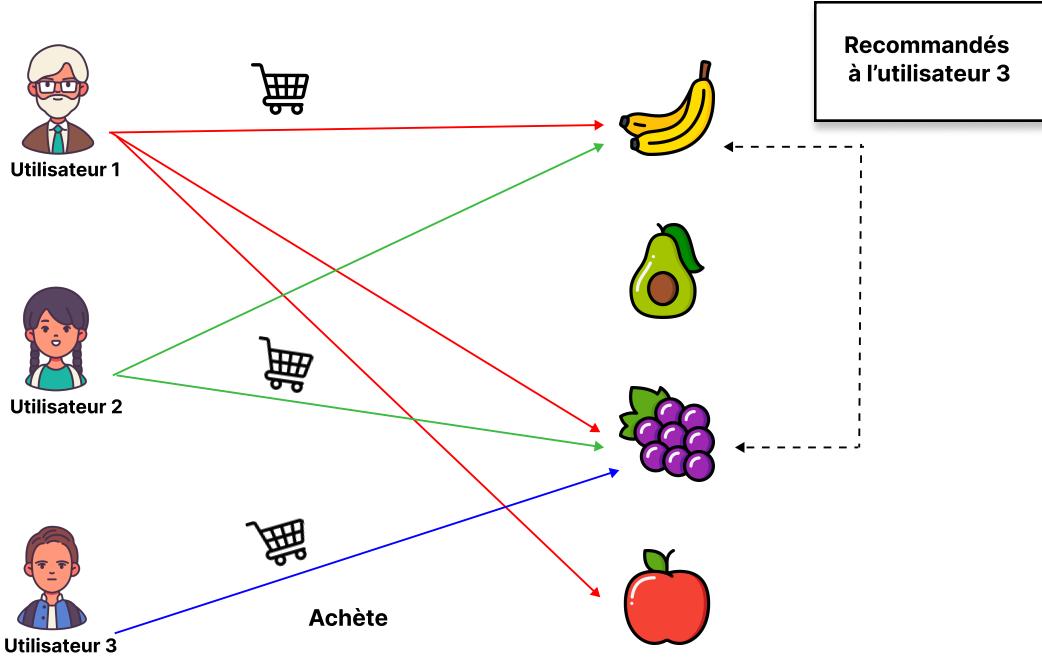


FIGURE 3.2 – Exemple illustrant l'extraction des éléments pertinents dans le scénario *item-based* à l'aide du Filtrage Collaboratif.

Les algorithmes de filtrage collaboratif passent par deux étapes essentielles : (1) Phase de calcul de voisinage et (2) la phase de calcul de la similarité.

Etape 1 : Calcul de voisinage

En se basant sur le profil ainsi que le *rating* laissé par l'utilisateur u_i , le système cherche tous les utilisateurs u_j qui lui sont les plus similaires tenant compte de l'appréciation laissé par l'utilisateur u_i (les utilisateurs similaires, avec lesquels il partage les mêmes préférences) [21].

Etape 2 : Calcul de similarité Pour que le système puisse déterminer quels seront les utilisateurs les plus similaires à l'utilisateur cible, plusieurs techniques sont utilisées. On cite quelques techniques de la similarité vectorielle ci-dessous :

- **Corrélation de Pearson :** La corrélation de Pearson est une méthode issue des statistiques. Elle est aussi très utilisée dans le domaine des systèmes de recommandation pour mesurer la similarité entre deux usagers. La formule ci-dessous, calcule cette valeur pour deux usagers A et B [17] :

$$\mathcal{W}(A, B) = \frac{\sum_{j=1}^n (V_{A,j} - \bar{V}_A)(V_{B,j} - \bar{V}_B)}{\sqrt{\sum_{j=1}^n (V_{A,j} - \bar{V}_A)^2 \sum_{j=1}^n (V_{B,j} - \bar{V}_B)^2}} \quad (3.1)$$

où :

- $V_{A,j}$: Vote de l'utilisateur A pour l'item j
- n : Nombre totale des items ayant été voté à la fois par les utilisateurs A et B
- \bar{V}_A : La moyenne des évaluations de l'utilisateur A sur l'ensemble des n items co-évalués par A et B.

- **Distance Euclidienne** : Une méthode classique qui s'agit d'une mesure de la longueur du segment ayant les deux points comme extrémités. Elle est le plus souvent utilisée pour comparer les profils des répondants entre les variables ou les deux extrêmes représentées les utilisateurs. Ci-dessous la formule pour un espace à n dimension :

$$D(A, B) = \sqrt{\sum_{j=1}^n (A_j - B_j)^2} \quad (3.2)$$

Comme la mesure de similarité est comprise toujours entre 0 et 1, c'est possible d'utiliser la formule de normalisation suivante [16] :

$$\text{Similarity}(A, B) = \frac{1}{1 + D(A, B)} \quad (3.3)$$

- **L'indice de Jaccard (*Jaccard Similarity*)** : Parfois appelé coefficient de similarité de Jaccard. Il s'agit de la comparaison des membres appartenant au deux ensembles différents pour voir quels membres sont partagés et lesquels sont distincts. C'est une mesure de similarité pour les deux ensembles de données, avec un rang de résultat entre 0 à 100%. Plus le pourcentage est élevé, plus les deux populations sont semblables. Elle est noté par la formule suivante :

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (3.4)$$

où :

- A : l'ensemble des items notés par l'utilisateur cible.
- B : l'ensemble des items notés par l'utilisateur voisin.

Exemple :

Afin d'illustrer cette mesure de similarité, on prend un exemple illustratif pour le cas d'une recommandation d'un livre pour un utilisateur u_α à partir des *ratings* entre 0 et 5, où 5 est le meilleur rating, laissés par d'autres utilisateurs u_i (voir la table 3.1).

| Utilisateur / livre _i | L1 | L2 | L3 | L4 | L5 |
|----------------------------------|-------|-------|-------|-------|-----|
| U_1 | 4.5/5 | 5/5 | 4.5/5 | 4/5 | 3/5 |
| U_2 | | 4/5 | 5/5 | 4.5/5 | 5/5 |
| U_3 | 2/5 | | 3.5/5 | | |
| U_4 | 3/5 | 4/5 | 5/5 | 3.5/5 | |
| U_α | | 4.5/5 | | 5/5 | 4/5 |

TABLE 3.1 – Échantillon des évaluations des livres par cinq utilisateurs

Calculons maintenant la similarité entre U_α et les quatre utilisateurs en appliquant les mesures de similarité précédentes. Commençons par la Corrélation de Pearson, le résultat est donné ci-dessous (voir la table 3.2) :

| Utilisateurs | \mathbf{U}_1 | \mathbf{U}_2 | \mathbf{U}_3 | \mathbf{U}_4 |
|--------------------------------------|----------------|----------------|----------------|----------------|
| $\mathbf{W}(\mathbf{A}, \mathbf{B})$ | -0.273 | 0.5021 | -0.9337 | -0.2844 |

TABLE 3.2 – Résultat de calcul de similarité en utilisant la corrélation de Pearson

Par suite, il est bien clair que l'utilisateur le plus similaire à notre utilisateur cible est U_2 . La table 3.3 résume les résultats obtenus par la distance Euclidienne avec la normalisation :

| Utilisateurs | \mathbf{U}_1 | \mathbf{U}_2 | \mathbf{U}_3 | \mathbf{U}_4 |
|--------------------------------------|----------------|----------------|----------------|----------------|
| $\mathbf{D}(\mathbf{A}, \mathbf{B})$ | 0.131 | 0.1626 | 0.1020 | 0.1212 |

TABLE 3.3 – Résultat de calcul de similarité en utilisant la distance Euclidienne

Finalement, on applique la similarité de Jaccard. La table ci-dessous -3.4- illustre les résultats obtenus :

$$Jacc(U_\alpha, U_1) = \frac{|\{U_\alpha \cap U_1\}|}{|\{U_\alpha \cup U_1\}|} = \frac{|\{L2, L4, L5\}|}{|\{L1, L2, L3, L4, L5\}|} = \frac{3}{5} = 0.6 \quad (3.5)$$

On obtient :

| Utilisateurs | \mathbf{U}_1 | \mathbf{U}_2 | \mathbf{U}_3 | \mathbf{U}_4 |
|-----------------|----------------|----------------|----------------|----------------|
| Similarity(A,B) | 0.6 | 0.75 | 0 | 0.4 |

TABLE 3.4 – Résultat de calcul de similarité en utilisant l'indice de Jaccard

Et donc on peut conclure que les utilisateurs les plus similaires à U_α sont U_2 et U_1 respectivement.

Afin de déterminer la prochaine recommandation à U_α , on sélectionne les livres lus et notés par les utilisateurs similaires mais qui n'appartient pas à l'ensemble des livres lus par U_α puis on calcule leur note moyenne.

$L2 = \frac{4.5+5}{2} = 4.75$. L1 n'appartient pas à l'ensemble des livres notés par U_2 . Puisque L1 et L2 sont bien notés par U_1 et U_2 , ils seront recommandés à l'utilisateur cible.

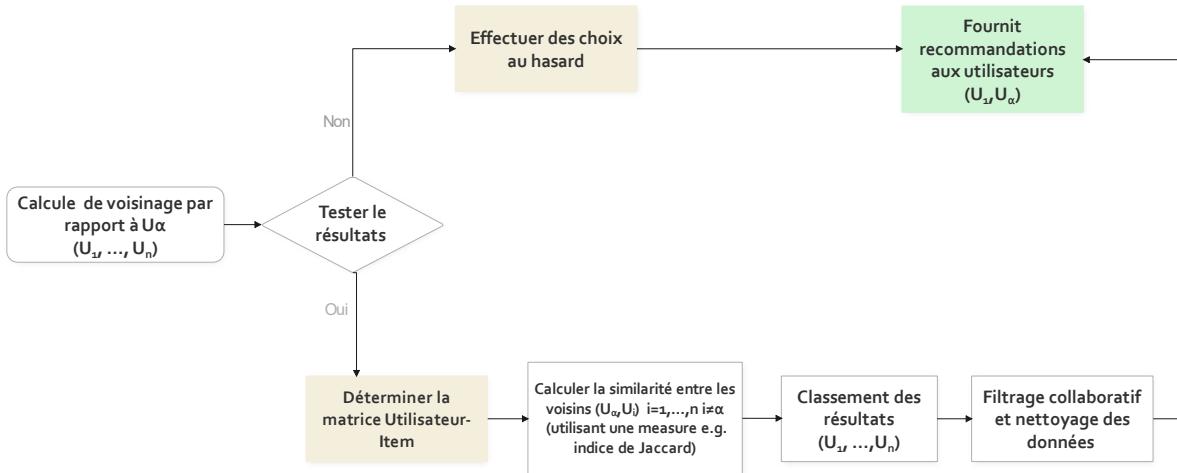


FIGURE 3.3 – Organigramme d'un processus de filtrage collaboratif basé sur l'utilisateur

3.3.2 Filtrage Basé sur un modèle

Dans une approche basé sur un modèle les algorithmes se basent sur les évaluations précédentes et les profils stockées des utilisateurs, mais cette fois-ci on ne calcule pas directement les prédictions mais on construit des modèles qui s'apprennent à partir des données des utilisateurs après les avoir classifiés dans des groupes. Une fois les groupes ou les modèles d'usagers sont trouvés, la prédition pour un usager donné est générée automatiquement à partir de son profil.

Pour la classification des utilisateurs dans des groupes, plusieurs méthodes sont utilisées. Par exemple une des méthodes les plus utilisées dans le domaine du Machine Learning c'est la méthode de *clustering* ou *classification*, une deuxième est celle des réseaux Bayésiens (*Bayesian networks*).

3.4 Filtrage basé sur le contenu

Elle est considérée comme l'approche la plus ancienne. Le filtrage basé sur le contenu (*Content-based filtering*) s'appuie sur le contenu des documents (thèmes abordés) pour les comparer à un profil lui-même constitué de thèmes. Son but c'est de trouver la réponse pour la question suivante : pourquoi l'usager, à qui la recommandation est destinée, a donné une haute valeur à certains items qu'il a évalués dans le passé ? Une fois cette question résolue, le système cherche parmi les nouveaux items ceux qui maximisent ces caractéristiques pour les lui recommander (voir figure 3.4) .

Par exemple, le profil de l'utilisateur cible peut contenir une liste des thèmes que l'utilisateur aime bien ou qu'il n'aime pas. Lors de l'arrivée d'un nouveau document, le système compare la représentation du document avec le profil pour prédire la satisfaction de l'utilisateur sur ce document .

Une telle approche ne requiert pas une grande communauté des utilisateurs ou un long

historique d'utilisation [12].

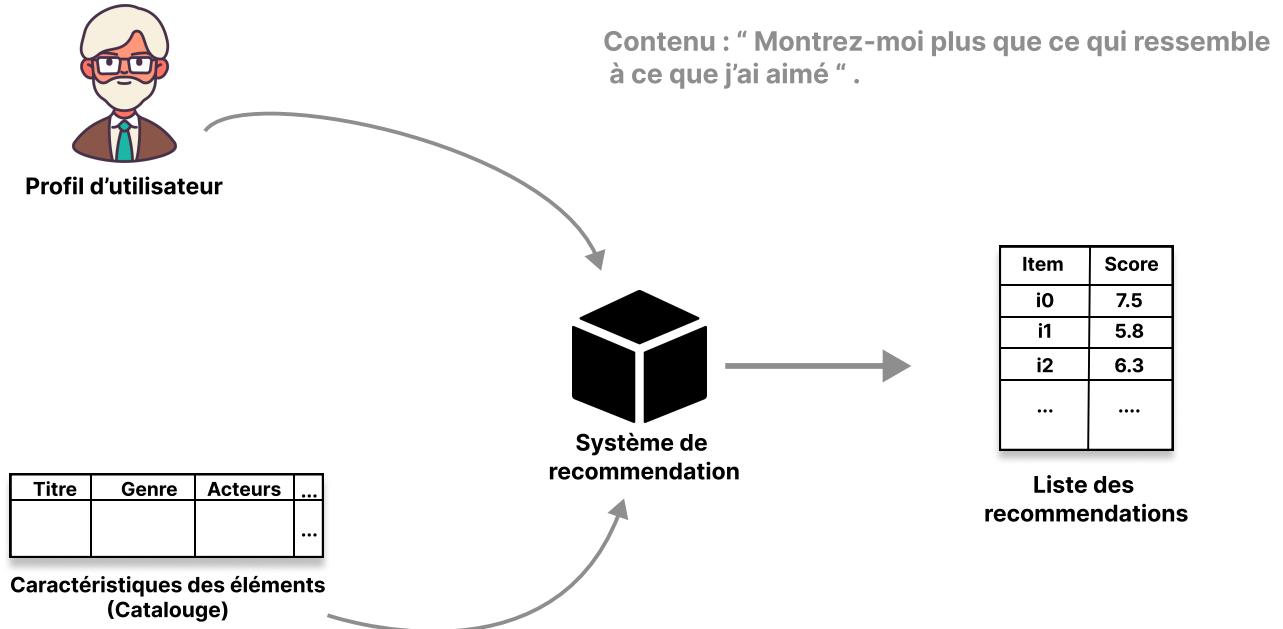


FIGURE 3.4 – Le processus de la recherche d'information dans un système CBF.

Cette méthode est souvent implémentée dans la recommandation d'items des type textuel, comme les articles de journaux, les pages Web. D'ailleurs, cette méthode s'appuie sur les techniques de recherche d'informations.

3.5 Filtrage hybride

L'approche hybride est la combinaison des deux méthodes traditionnelles précédentes (voir figure 3.5). Ajouter à cela les méthodes basées sur d'autres techniques en provenance des statistiques et de l'apprentissage automatique. De nombreux systèmes reposent sur le filtrage hybride. D'ailleurs, il n'y a aucune raison pour que plusieurs techniques différentes du même type ne puissent pas être hybridées. En effet, plusieurs études ont démontré que les approches hybrides peuvent fournir des recommandations plus précises [14].

En général, l'hybridation s'effectue en deux phases : appliquer séparément le filtrage collaboratif et autres techniques de filtrage pour générer des recommandations candidates, et combiner ces ensembles de recommandations préliminaires selon certaines méthodes telles que la pondération, la mixtion, la cascade et la commutation, afin de produire les recommandations finales pour les utilisateurs [15].

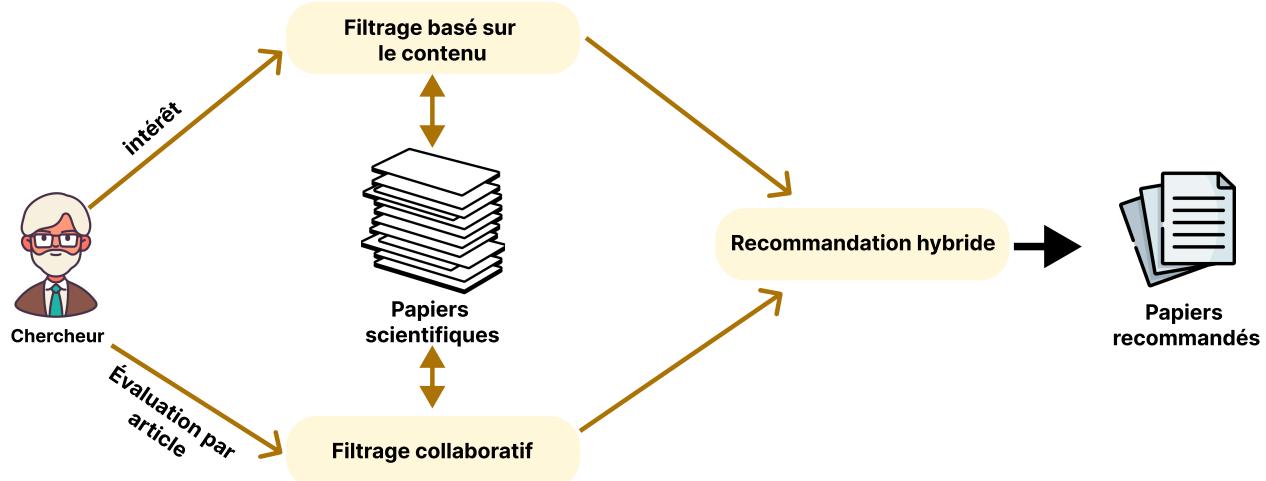


FIGURE 3.5 – Une abstraction d'un système basé sur le filtrage hybride.

3.6 Conclusion

Ce chapitre a abordé brièvement les différentes approches qui représentent le cerveau derrière les systèmes de recommandation d'aujourd'hui tout en explorant les mesures de similarité les plus utilisées dans ces derniers.

Dans le cadre de notre application, nous avons choisi d'implémenter un système de filtrage collaboratif basé sur la mémoire qui s'appuie sur la similarité de Jaccard pour la détermination des utilisateurs similaires.

CHAPITRE 4

CONCEPTION ET MISE EN ŒUVRE

4.1 Introduction

Afin d'aboutir à une meilleure organisation et une bonne maîtrise du travail et donc arriver à déployer de meilleures applications, il est nécessaire de suivre une démarche méthodologique rigoureuse. Pour cela le choix d'une méthode d'analyse et de conception est d'une très grande importance. Ainsi, une fois le travail de modélisation est achevé, on arrive à l'étape du codage (implémentation ou programmation) et test qui consiste à traduire dans un langage de programmation des différentes fonctionnalités définies lors des phases d'analyse et de conception, et enfin vérifier, à travers une pile de tests logiciel, que les fonctionnalités développées ne comportent pas de bug, sinon les fixer le cas échéant. Le but étant d'avoir en résultat une application qui réalise correctement toutes les tâches attendues par ses utilisateurs. A cette fin, il faut bien choisir les logiciels ainsi que les langages adaptés pour la mise en place de l'application.

A travers ce chapitre, nous détaillerons toutes les étapes méthodologiques de conception et de mise en œuvre, afin d'aboutir à un outil opérationnel et déployable sur des dispositifs mobiles (e.g. Smartphones). Nous exposons ensuite des cas d'utilisations réels illustrant les fonctionnalités développées et motivant l'intérêt d'utiliser notre application.

4.2 Cahier de charge

Nous proposons dans ce projet les modèles conceptuels d'une application mobile multiplateformes facilitant la gestion d'un campus universitaire (celui de l'université Oran1 à titre d'exemple) tout en donnant l'accès aux étudiants locaux (et aux invités par le biais d'un accès restreint) aux différents services dont ils ont besoin. L'axe principal est la mise en œuvre d'une architecture optimisée de géolocalisation à l'intérieur d'un campus universitaire pour les étudiants, personnels et invités en plus d'un système de recommandations de livres ou ouvrages dédiés aux étudiants et aux chercheurs.

4.3 Architecture globale de l'application

Le schéma ci-dessous (Figure 4.1) illustre l'architecture globale de notre application mobile. Cette figure comporte trois tiers importants : **(1) Tiers Client** : qui se connecte à l'application en utilisant soit son propre compte ou bien en mode invité ; **(2) Tiers Front-end** : qui héberge la partie visuelle de l'application de gestion de campus avec ses différentes interfaces aux services offerts par le tiers Serveur ; **(3) Tiers Serveur** : qui héberge toute la logique métier de l'application et ses différents services, y compris la gestion des accès, la récupération et le transfert des données se trouvant à la base de données vers le tier Front-end.

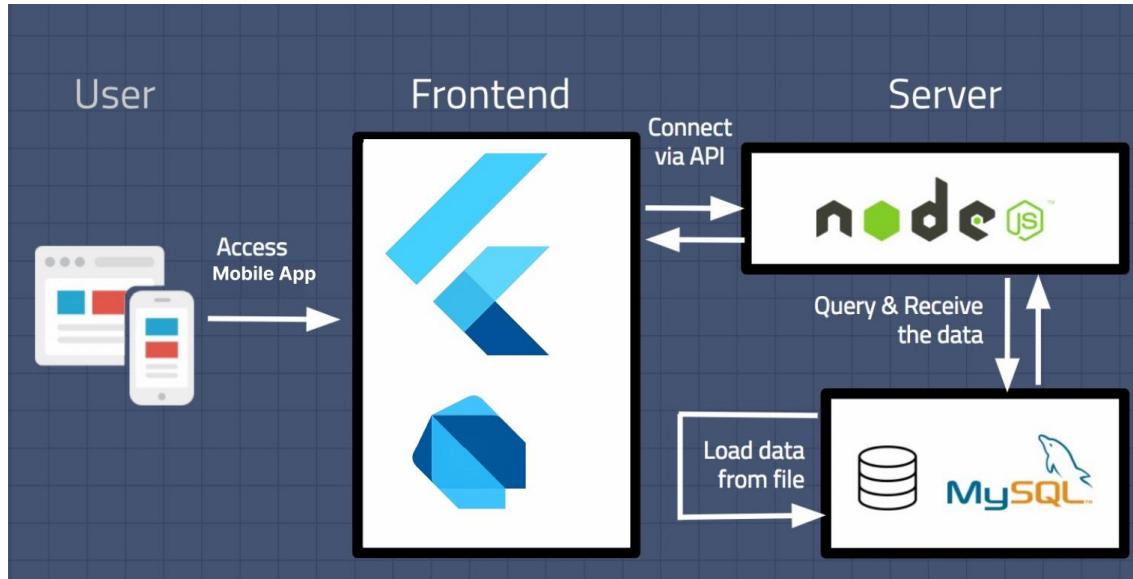


FIGURE 4.1 – Architecture globale de l'application

4.3.1 Les besoins fonctionnels

Un système ne devient opérationnel que s'il satisfait les besoins fonctionnels (besoins métiers). Principalement, l'application doit couvrir les besoins fonctionnels représentés dans le tableau 4.1 associés à leurs acteurs en offrant deux services principaux, le service géolocalisation et le service bibliothèque. Avant d'accéder à ces deux services, il faut d'abord que l'étudiant s'authentifie.

L'authentification permet de s'assurer de l'identité de l'utilisateur et de garantir la sécurité des données.

| Acteurs | Rôle |
|------------------------|---|
| Administrateur | <ul style="list-style-type: none"> - Gérer les tables (étudiants et services) - Valider/Refuser les demandes d'ajout des Point d'intérêt (PI) - Suivre les états des emprunts d'ouvrages |
| Utilisateur (étudiant) | <ul style="list-style-type: none"> - Authentification - Récupérer le mot de passe oublié - Se localiser, chercher et naviguer vers les PI - Demander l'ajout d'un PI - Réserver/Annuler l'emprunt d'un ouvrage - Évaluer un ouvrage |

TABLE 4.1 – Acteurs et leurs rôles

4.3.2 Les besoins non fonctionnels

Les besoins non fonctionnels sont des exigences qui portent sur le comportement du système, mais permettent d'identifier ses exigences internes et externes qui garantissent sa performance et le respect des exigences de l'utilisateur.

- **Performance** : notre application doit assurer un temps de réponse minimum tout en répondant aux besoins du manipulateur.
- **La simplicité** : chaque utilisateur pourra utiliser cette application d'une manière facile et claire.
- **L'ergonomie de l'interface** : les interfaces doivent être simples et conviviales
- **La modularité de l'application** : avoir un code simple, facile à maintenir et à comprendre en cas de besoin.

4.4 Modélisation UML

UML (*Unified Modeling Language*), se définit comme un langage de modélisation graphique et textuel destiné à comprendre et à définir des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML modélise l'ensemble des données et des traitements en élaborant des différents diagrammes.

4.4.1 Diagramme de cas d'utilisation général

La figure 4.2 ci-dessous illustre le diagramme de cas d'utilisation général qui permet de distinguer les différents services que l'étudiant peut avoir accès, en plus du rôle de l'administrateur dans la gestion de ces derniers.

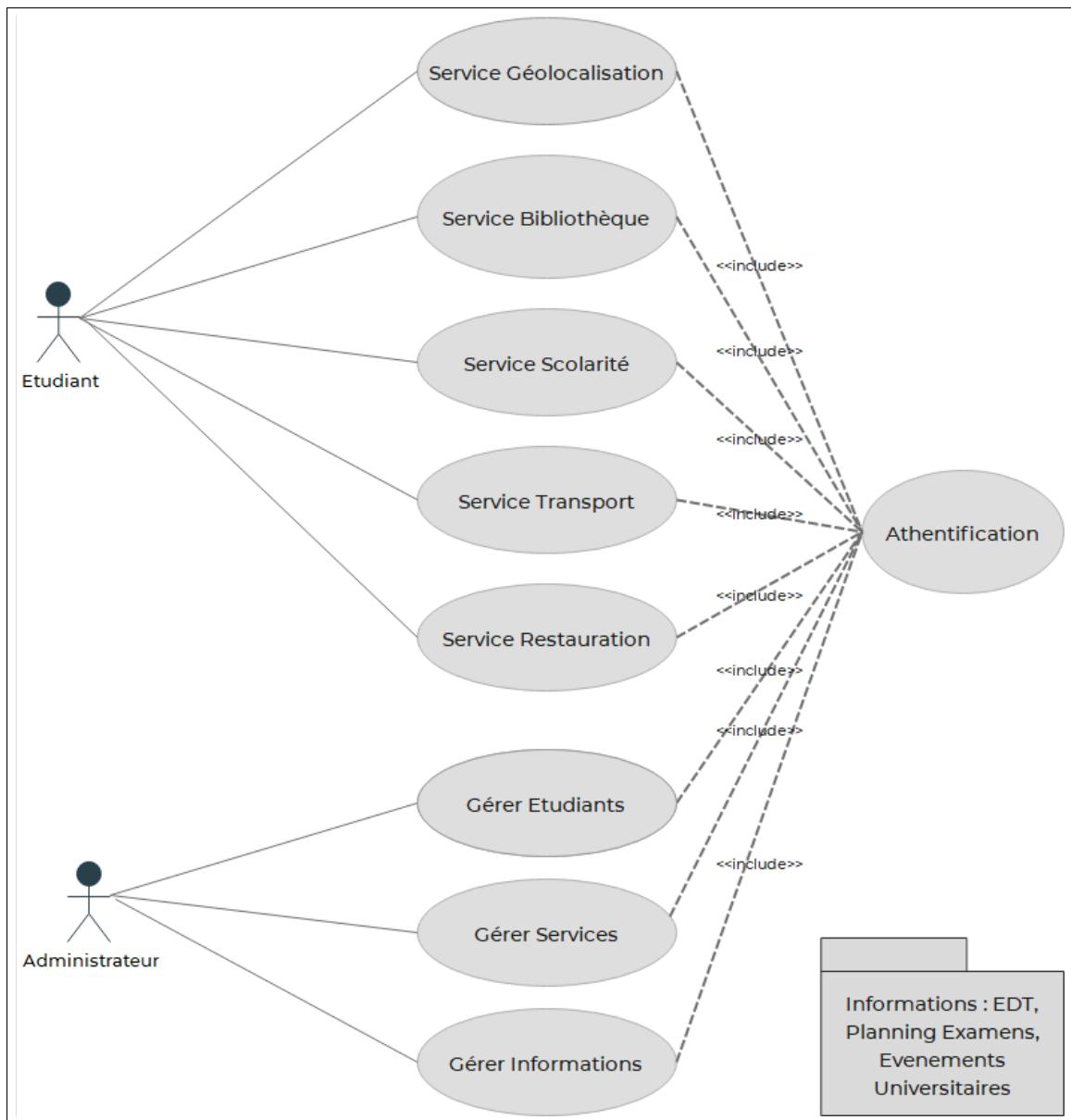


FIGURE 4.2 – Diagramme de cas d'utilisation général

Dans ce qui suit, nous allons détailler deux principaux services (**Géolocalisation** et **Bibliothèque**) en laissant les autres services comme des perspectives qui pourront être développer et réaliser dans l'avenir.

4.4.2 Analyse et conception du service " Géolocalisation "

Dans le service géolocalisation, l'accès se fait directement après l'authentification, l'étudiant bénéfice de plusieurs fonctionnalités qui seront décrites par le cas d'utilisation et le diagramme de séquence qui suivent.

4.4.2.1 Diagramme de cas d'utilisation du service de géolocalisation

Ce diagramme (voir figure 4.3) illustre bien le fonctionnement du service de géolocalisation, il représente les différentes fonctionnalités proposés pour l'étudiant ainsi que le rôle de l'administrateur dans la gestion de ce service.

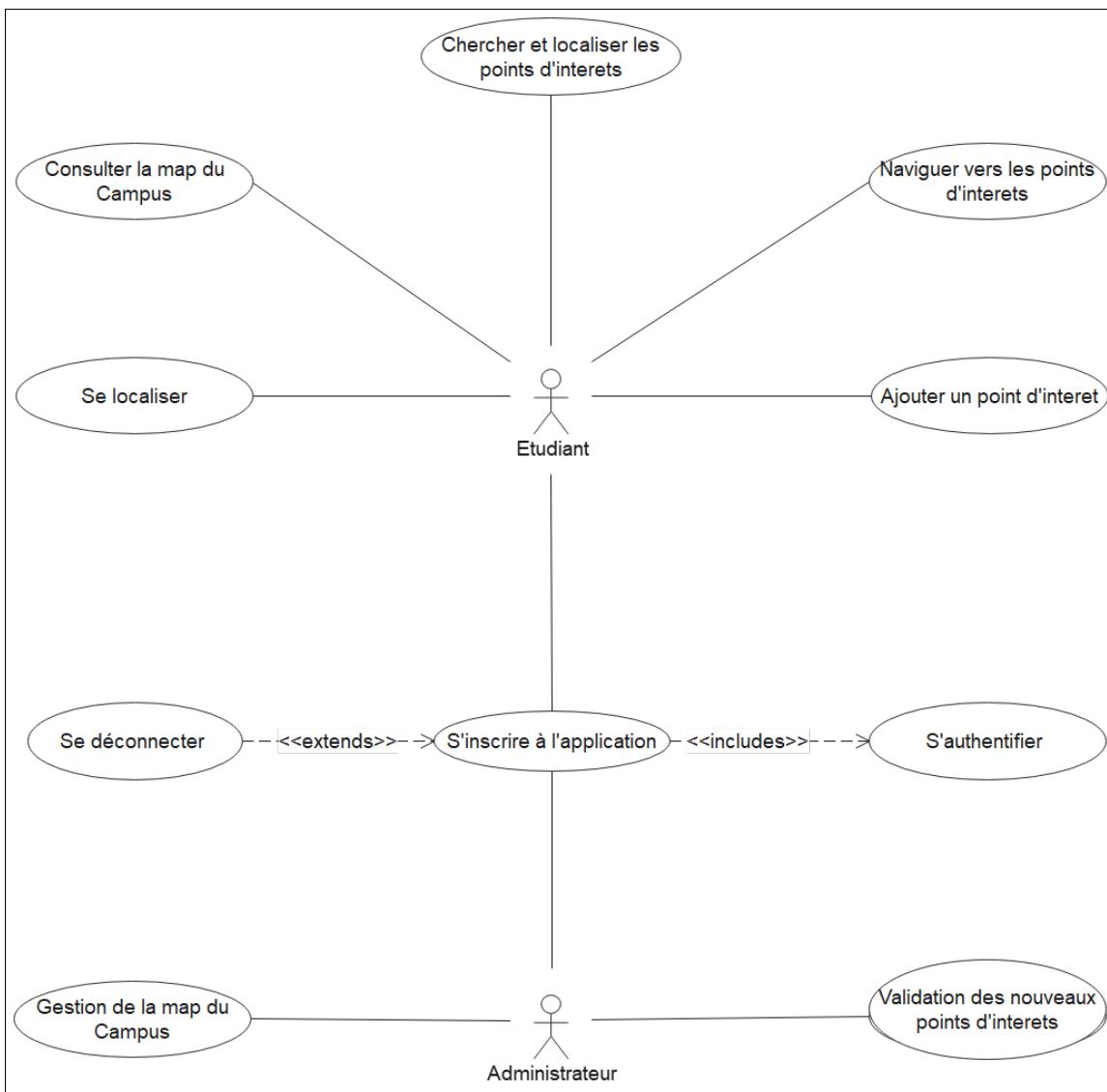


FIGURE 4.3 – Diagramme de cas d'utilisation du service géolocalisation

4.4.2.2 Le diagramme de séquence

Le diagramme de séquence suivant représente les différentes collaborations entre les objets de notre application selon un point de vue temporel. Dans ce diagramme nous mettons l'accent sur l'accès à la cartographie du campus en plus de la recherche des différents points d'intérêts (voir figure 4.4).

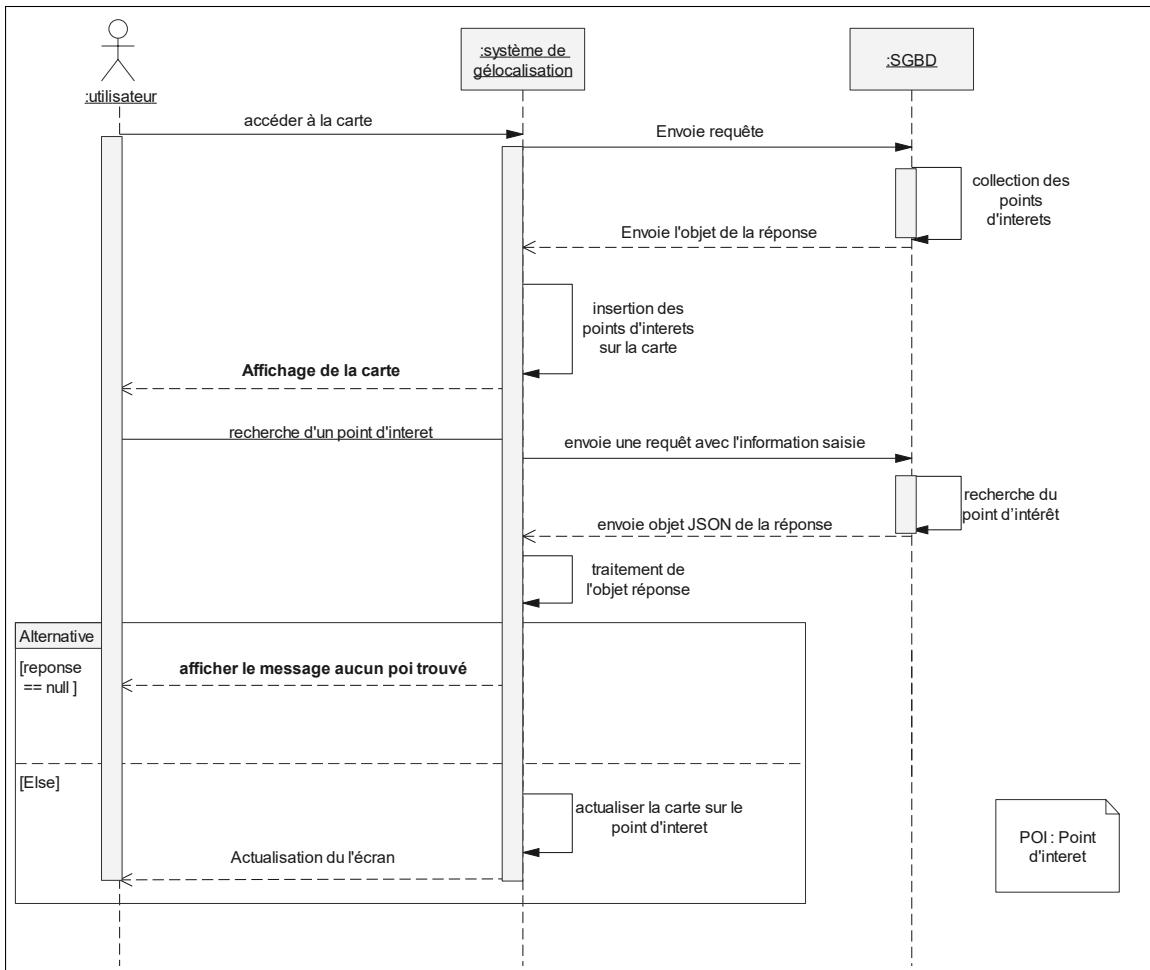


FIGURE 4.4 – Diagramme de séquence du service de géolocalisation

4.4.3 Analyse et conception du service " Bibliothèque "

Dans le service bibliothèque, l'accès au service se fait directement comme pour le service géolocalisation sauf pour la fonctionnalité de réservation d'ouvrages ,La condition c'est que l'étudiant soit d'abord adhérent au sein de la bibliothèque donc quand un étudiant essaye de réserver un ouvrage,le serveur vérifie automatiquement si l'étudiant est un adhérent ou pas, si l'étudiant est adhérent ; alors il pourra réserver l'ouvrage, sinon la demande de réservation lui sera refusée.

4.4.3.1 Diagramme de cas d'utilisation du service Bibliothèque

Ce diagramme (voir Figure 4.5) illustre bien le fonctionnement du service Bibliothèque, il représente les différentes fonctionnalités proposés pour l'étudiant y compris la partie de recommandation ainsi que le rôle de l'administrateur dans la gestion de ce service.

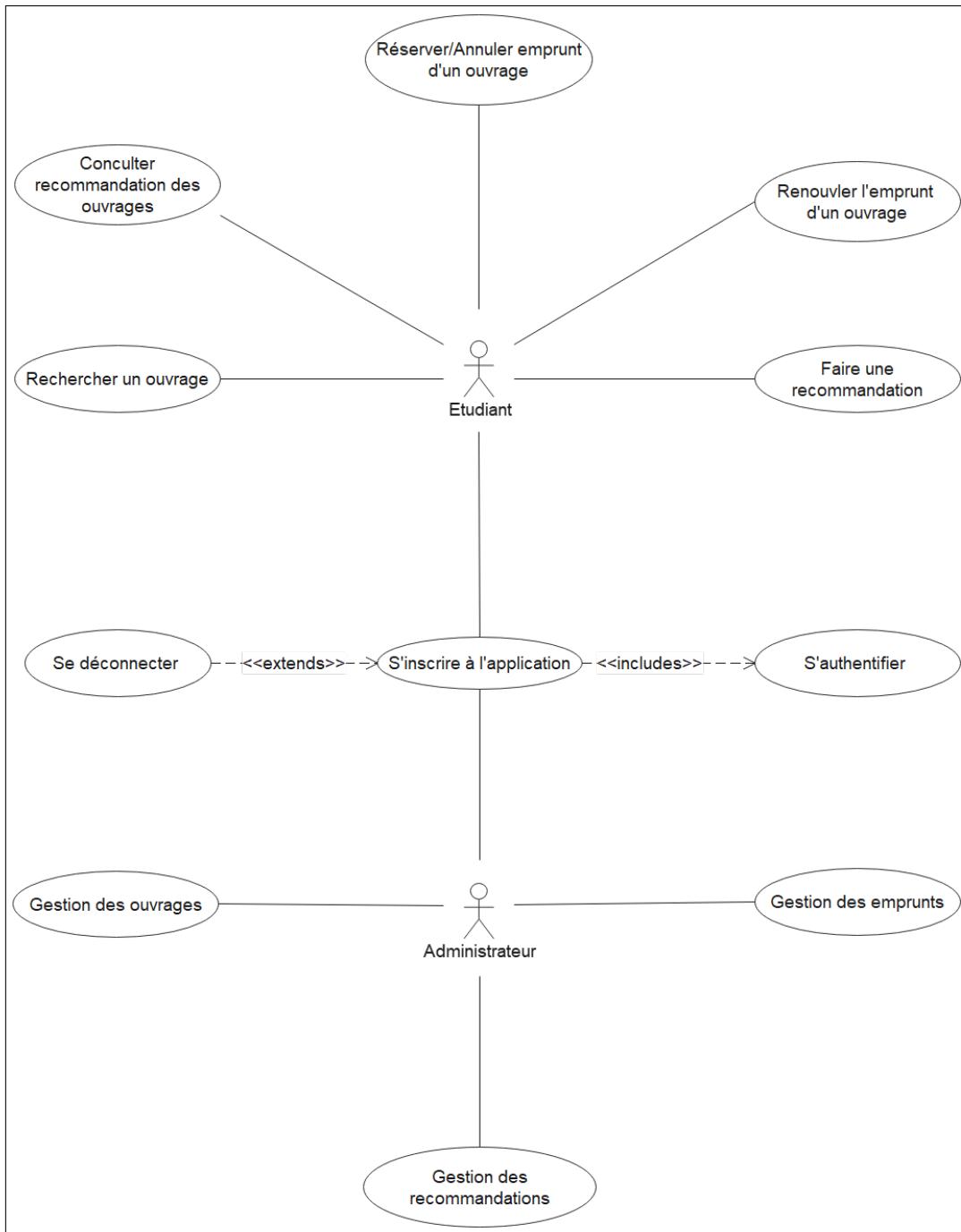


FIGURE 4.5 – Diagramme de cas d'utilisation du service bibliothèque

4.4.3.2 Les diagrammes de séquence

Nous allons détailler quelques exemples de diagrammes de séquence des cas d'utilisations que nous avons déjà cité au-dessus.

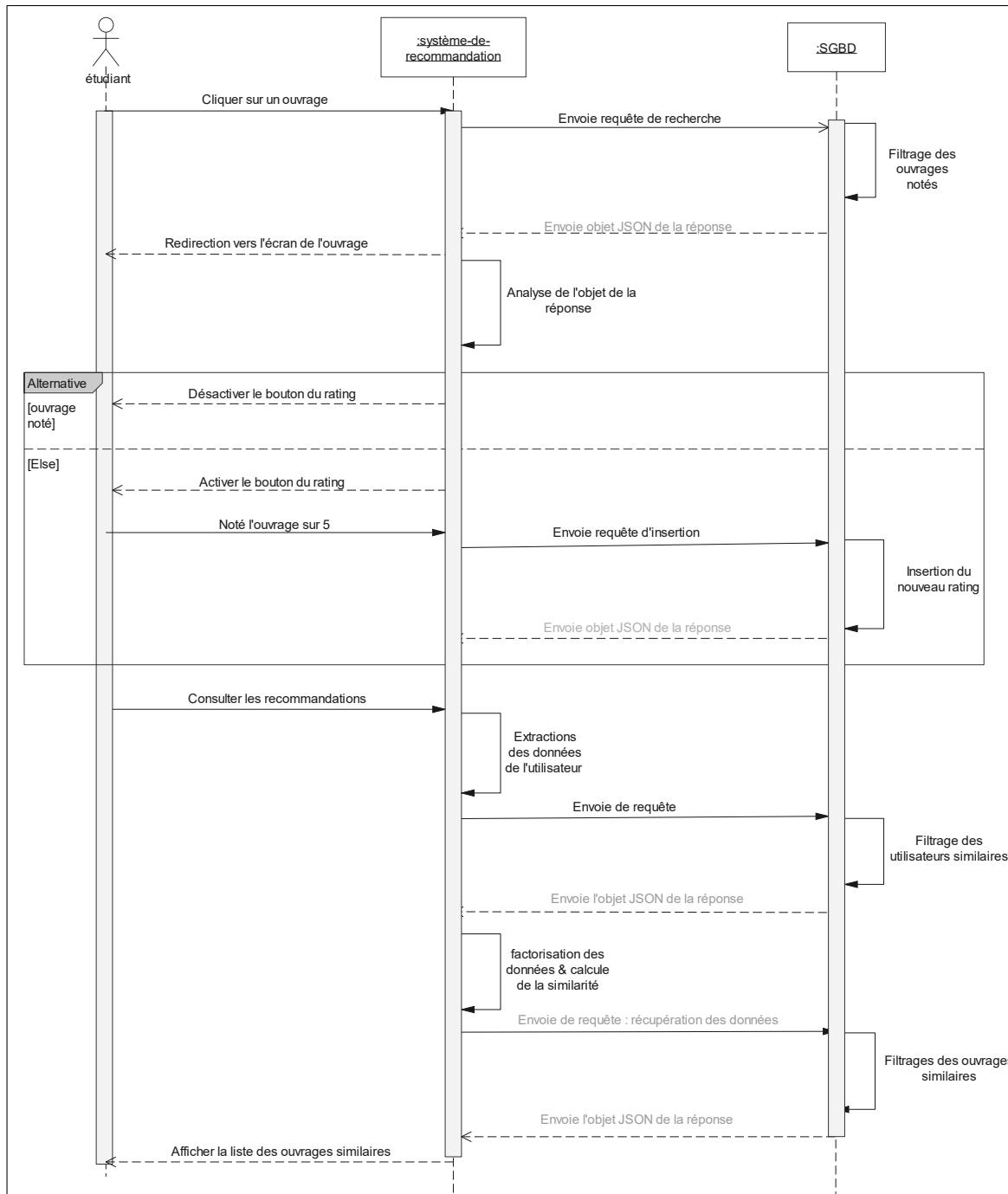


FIGURE 4.6 – Diagramme de séquence du service Bibliothèque : partie *Système de recommandation*

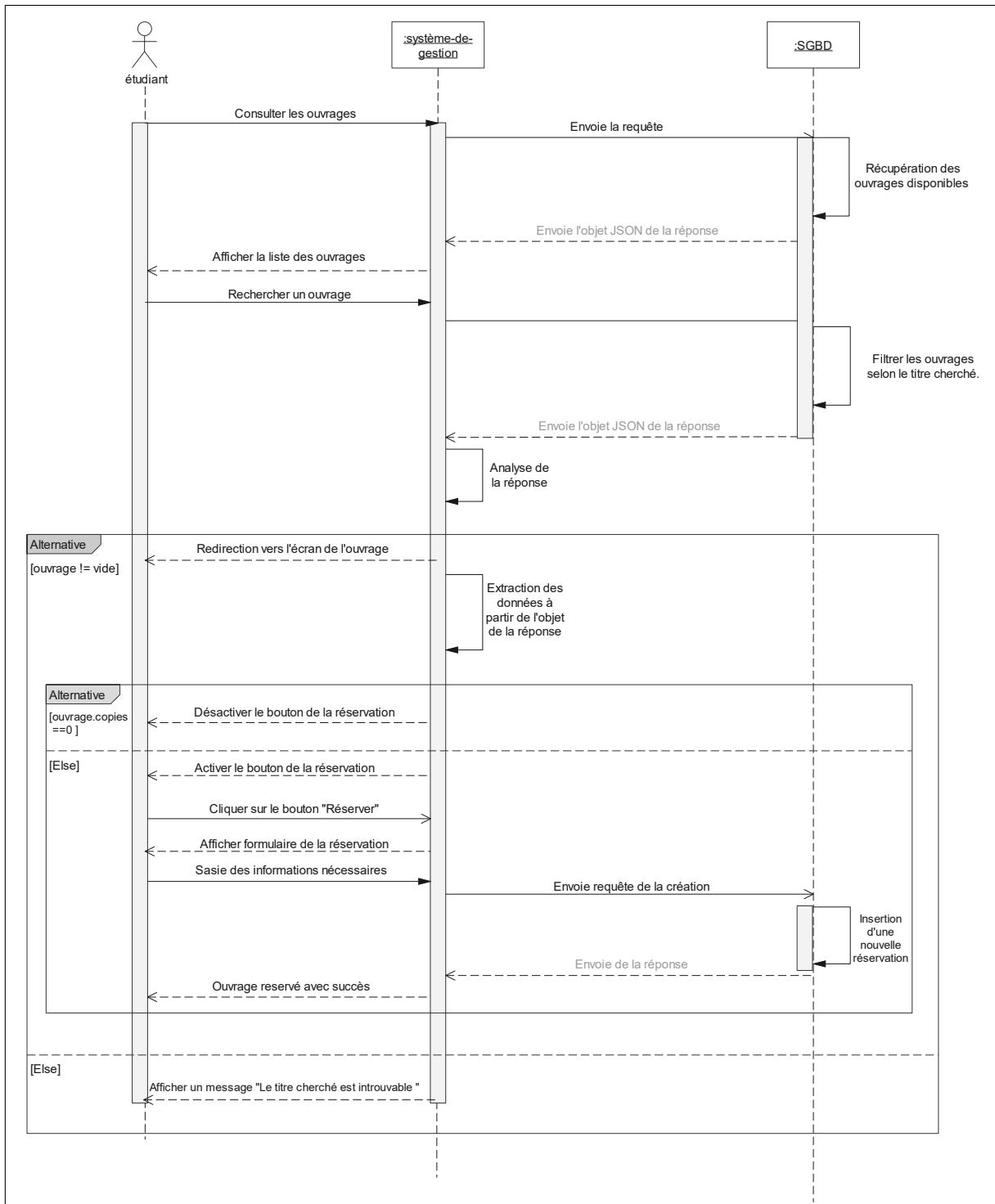


FIGURE 4.7 – Diagramme de séquence du service Bibliothèque : partie *Gestion des ouvrages*

4.5 Système d'information

4.5.1 Modèle conceptuel de données (MCD)

Le modèle conceptuel des données (MCD) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités [20].

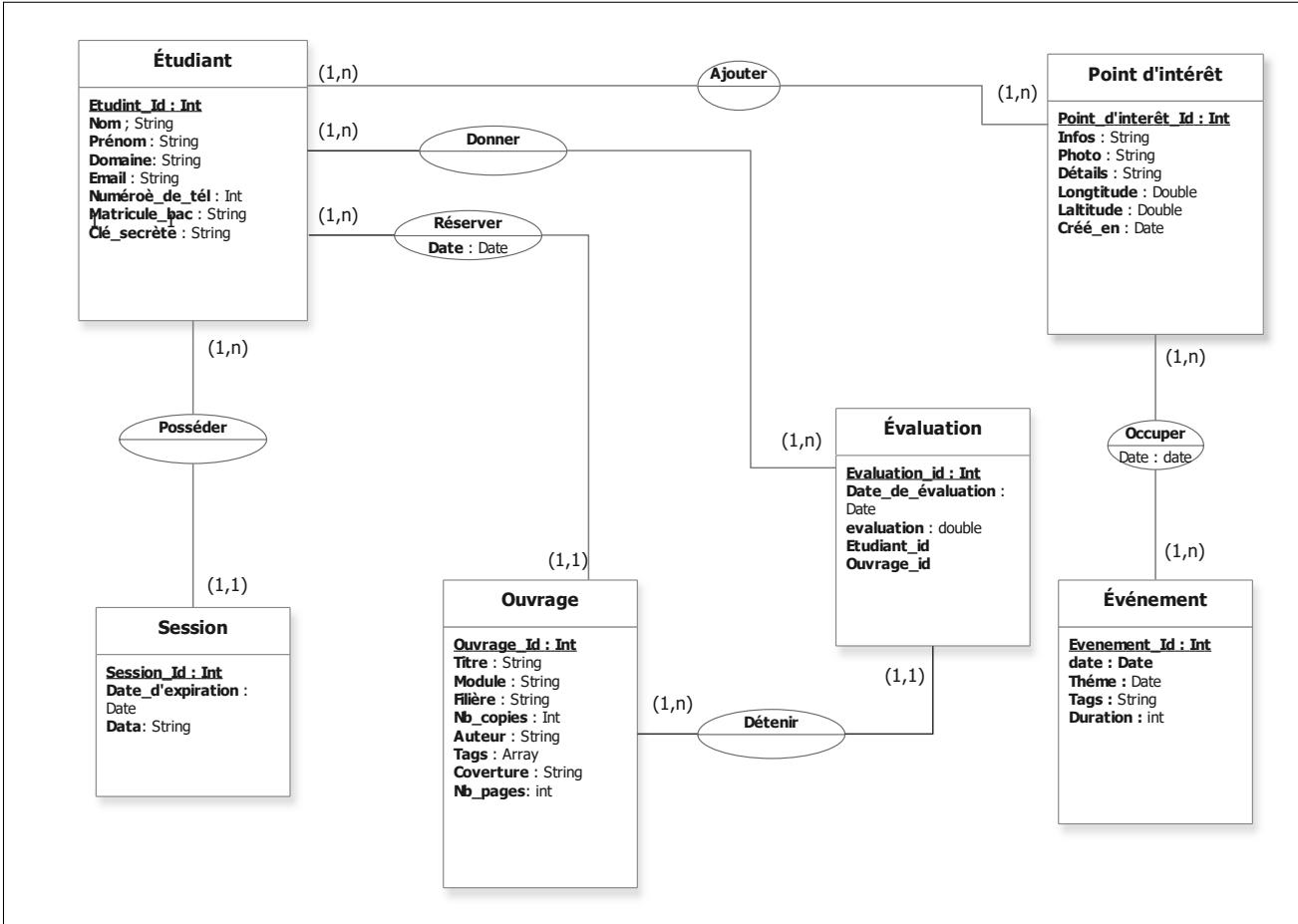


FIGURE 4.8 – Le Modèle conceptuel de données du projet

4.5.2 Modèle logique de données (MLD)

Le modèle logique représente une abstraction du schéma global de la base de données. Il établit la structure des éléments de données et les relations entre eux. Il est indépendant de la base de données physique qui détaille la façon dont les données seront mises en œuvre. Ce modèle est utilisé comme modèle pour les données utilisées. Il fournit plus de détail sur les éléments de la modélisation conceptuelles.

Le modèle ci-dessous représente le modèle logique de notre base de données :

- Étudiant (**Étudiant ID**, nom, prénom, domaine, date de naissance, email, numéro de tél, matricule bac, clé secrète).
- Point d'intérêt (**Point Of Interest ID**, information, longitude, latitude, détails).
- Ouvrage (**Ouvrage ID**, titre, auteur, couverture, description, domaine, nb pages, nb copies, catégories).
- Session (**Session ID**, date d'expiration, data).
- Évaluation (**Évaluation ID**, date d'évaluation, évaluation, Étudiant ID, Ouvrage ID)
- Évènement (**Évènement ID**, date, thème, tags, durée)
- Possède (**Étudiant ID**, **Session ID**).
- Donner(**Évaluation ID**, **Étudiant ID**).
- Détenir(**Ouvrage ID**, **Évaluation ID**).
- Réserver(**Étudiant ID**, **Ouvrage ID**, **Date**).
- Localiser(**Évènement ID**, **Point d'intérêt ID**).

Explication du MLD proposé :

- *Étudiant ID, Point d'intérêt ID, Ouvrage ID Session ID, Évaluation ID et Évènement ID* représentent respectivement les clés primaires des tables suivantes : *Étudiant, Point d'intérêt, Ouvrage, Évaluation, Évènement* et *Session*.
- La clé primaire de la table *Possède* est composée de la clé primaire de la table *Utilisateur* et la table *Session* qui sont ici des clés étrangères.
- L'attribut *étudiant_id* représente la clé étrangère *Étudiant ID*.
- La clé primaire de la table *Réserver* est composée des clés primaires des tables *Étudiant* et *Ouvrage* qui sont ici des clés étrangères et l'attribut *Date*.
- La clé primaire de la table *Localiser* est composée des clés primaires des tables *Point d'intérêt* et *Évènement* qui sont ici des clés étrangères.

4.5.3 Modèle physique de données (MPD)

Présentons maintenant le schéma physique des tables qui s'agit du résultat de l'implémentation du modèle logique de la BDD normalisée en SQL.

4.5.3.1 Schéma normalisé de la BDD

Le schéma des tables *Étudiant*, *Ouvrage* et *Point d'intérêt* qui représentent le noyeux de la base de données.

```

1 CREATE TABLE `students` (
2     `id` int NOT NULL AUTO_INCREMENT,
3     `first_name` varchar(255) NOT NULL,
4     `last_name` varchar(255) NOT NULL,
5     `field` varchar(255) NOT NULL,
6     `birth_date` datetime NOT NULL,
7     `email` varchar(255) NOT NULL,
8     `phone_number` int NOT NULL,
9     `matricule_bac` varchar(255) NOT NULL,
10    `cle_secrete` varchar(255) NOT NULL,
11    `created_at` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
12    `updated_at` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6),
13    PRIMARY KEY (`id`),
14    UNIQUE KEY (`email`),
15    UNIQUE KEY (`phone_number`),
16    UNIQUE KEY (`matricule_bac`),
17    UNIQUE KEY (`cle_secrete`)
18 );

```

FIGURE 4.9 – Script SQL permettant la création de la table Étudiant (*Student*)

```

1 CREATE TABLE `textbook` (
2     `id` int NOT NULL AUTO_INCREMENT,
3     `title` varchar(255) NOT NULL DEFAULT '',
4     `author` text NOT NULL,
5     `cover_image_link` varchar(255) NOT NULL,
6     `desc` text,
7     `field` varchar(255) NOT NULL,
8     `nb_copies` int NOT NULL DEFAULT '1',
9     `page_count` int NOT NULL,
10    `created_at` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
11    `updated_at` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6),
12    `categories` text NOT NULL,
13    PRIMARY KEY (`id`)
14 );

```

FIGURE 4.10 – Script SQL permettant la création de la table Ouvrage (*Textbook*)

```

● ○ ●
1 CREATE TABLE `point_of_interest` (
2   `added_on` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6),
3   `updated_at` datetime(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6),
4   `id` int NOT NULL AUTO_INCREMENT,
5   `longitude` double NOT NULL,
6   `latitude` double NOT NULL,
7   `icon` varchar(255) NOT NULL DEFAULT '',
8   `titre` varchar(255) NOT NULL,
9   `contact_info` varchar(255) NOT NULL,
10  `cover` text NOT NULL,
11  `details` text NOT NULL,
12  `floors` tinyint NOT NULL,
13  PRIMARY KEY (`id`),
14  UNIQUE KEY (`titre`)
15 );

```

FIGURE 4.11 – Script SQL permettant la création de la table Point d’interet (*Point of Interest*)

4.5.3.2 Remplissage de la BDD

Examinons maintenant le script SQL chargé d’ensemencer la base de données avec les données nécessaires au fonctionnement de notre système.

Pour notre système, nous devions recueillir toutes les données (longitude et de latitude) pour les endroits cibles sur le campus. Les données suivantes sont réelles.

```

● ○ ●
1 INSERT INTO `point_of_interest` VALUES ('2022-06-13 09:44:24.571814','2022-06-13 09:44:24.571814',688,-0.631498,35.662024,'college','Salles des conférences TALAHIT','','https://lh3.googleusercontent.com/pw/AM-JKLVBblz2xBCXC4rme5amEQ7c11NizA4a8FX2UHE59HXo1B8eni5Q9Bt-z-KG83W3QKLHcyNX1n1EA
AuqlkEhTr9NN3Ns6gSS4_1ylfaU_phKYSU7ikDg6Xaqz7_6HLT-PU69gV6PYt9HS9iuXHg59UOs=w1280-h960-no','','0'),
2 ('2022-06-13 09:44:24.578216','2022-06-13 09:44:24.578216',689,-0.628728,35.660766,'college','Re
staurant universitaire','','https://lh3.googleusercontent.com/pw/AM-JKLUiH7N6IBxy_hyL1YJANltQ1rF
mkGrjLosVqkf3afOKm9FKyCyTBW765qGTStSppk7YqGlVaYUtjrGpKVBHnMhvF6BCkv1We0trXgqzF_wGu8V-bWAnWCHTkRl
oajtnTmNkH36Gzw18mJKYrk06eM_7=w1258-h943-no','','0');
3

```

FIGURE 4.12 – Script SQL permettant l’insertion de deux points d’intérêts dans la table *Point d’intérêt*.

4.5.3.3 Quelques requêtes et résultats

Nous montrerons l’exécution de quelques requêtes SQL pour illustrer l’efficacité de notre modèle de base de données.

1. Affichage de la liste des étudiants qui ont un compte dans l'application (i.e. les utilisateurs inscrits) (voir les Figures 4.13 et 4.14).



```
1 SELECT matricule,first_name,last_name FROM campus_univ.students
2 RIGHT JOIN campus_univ.users
3 ON campus_univ.students.id = campus_univ.users.studentIdId
4 ORDER BY matricule DESC;
```

FIGURE 4.13 – La première requête SQL

| | matricule | first_name | last_name |
|---|-----------|--------------|-------------|
| ▶ | 6 | Kip | Jacquelyn |
| | 4 | Jesselyn | Elna |
| | 34 | Anjela | Oneida |
| | 33 | Enrika | Daffy |
| | 32 | Ysabel | Constantine |
| | 30 | Vicki | Fiona |
| | 3 | Bonnee | Maddalena |
| | 29 | Sheilah | Sabina |
| | 28 | Lib | Berry |
| | 27 | Dione | Darleen |
| | 26 | Kristyn | Madelin |
| | 25 | Reena | Marla |
| | 12 | Brynna | Johnath |
| | 104 | Tarek Yacine | Atbi |
| | 103 | Ikram | Messadi |

FIGURE 4.14 – Le résultat d'exécution de la 1ère requête

2. Lister toutes les bibliothèques du campus (Figures 4.15 et 4.16).



```
1 SELECT titre, latitude, longitude FROM campus_univ.point_of_interest WHERE titre LIKE 'Bib%';
2
```

FIGURE 4.15 – La deuxième requête SQL

| # | titre | latitude | longitude |
|---|--|-------------------|---------------------|
| 1 | Bibliothèque : Faculté des sciences exactes | 35.66268953183995 | -0.6300375240752701 |
| 2 | Bibliothèque : Faculté des sciences sociales | 35.662885 | -0.62746 |

FIGURE 4.16 – Le résultat d'exécution de la 2ème requête

3. Lister le nombre de page des ouvrages réservés par l'étudiant ayant le matricule 104 ainsi que son nom & son prénom (Figures 4.17 et 4.18).

```

1
2  SELECT title, page_count, first_name, last_name FROM campus_univ.textbook AS txt,
3   campus_univ.reservation AS resv,
4   campus_univ.students AS std,
5   campus_univ.users AS us
6  WHERE resv.userId = 354
7  AND resv.textbook_id = txt.id
8  AND resv.userId=us.id AND us.studentIdId = std.id;

```

FIGURE 4.17 – La troisième requête SQL

| # | title | page_count | first_name | last_name |
|---|---------------------|------------|------------|-----------|
| 1 | Android in Practice | 500 | Ikram | Messadi |
| 2 | SOA Security | 512 | Ikram | Messadi |

FIGURE 4.18 – Le résultat d'exécution de la 3ème requête

4.6 Réalisation

Cette section est consacrée aux outils et technologies utilisés dans la conception et la réalisation de notre application mobile et ses fonctionnalités, l'implémentation de notre propre API ainsi que l'interface et des exemples sur quelques scénarios d'utilisation de l'application.

4.6.1 Environnement matériel

Pour le développement de notre système nous avons utilisé nos deux PC. Le premier PC est doté d'un processeur Intel i7-8750H, 2.21 GHz, 8Go de RAM et est équipé d'un système

d'exploitation Linux Mint édition Cinnamon 20.3. Nous avons exploité ce PC surtout pour le développement de la partie textitServeur de l'application. Le deuxième PC est équipé d'un processeur i5-5200U, 2.21 GHz , 8Go de RAM et un système d'exploitation Windows 10 édition Pro architecture 64-bit. Les deux machines ont été exploités pour le développement de l'application.

Notre application a été testée sur nos deux Smartphones. Le premier Smartphone est un Samsung S6 Galaxy Edge Plus avec un Octa-core CPU, 4Go RAM et un Android 7.0.

Le deuxième Smartphone est un Huawei AUM-L29 avec un QS 430 CPU, 2Go RAM et un Android 8.0.

4.6.2 Environnement logiciel

Nous listons ci-après les environnements logiciels que nous avons exploité pour la réalisation de notre projet de fin d'étude.



Visual Studio Code est un éditeur de code source développé par Microsoft pour Windows, Linux et macOS. Il comprend Git intégré et la prise en charge du débogage, de la mise en évidence de la syntaxe, de l'achèvement intelligent du code, des extraits de code et de la refactorisation du code. Il est hautement personnalisable, permettant aux utilisateurs de modifier le thème, les raccourcis clavier, les préférences et d'installer des extensions qui ajoutent des fonctionnalités supplémentaires [23].



MySQL Workbench est un outil graphique permettant de travailler avec les serveurs et les bases de données MySQL, il couvre cinq sujets principaux : SQL Development, Data Modeling (Design), Server Administration, Data Migration, MySQL Enterprise Support [24].



Postman est une application permettant de tester des API, il regroupe chaque test d'API dans une collection, permettant de mutualiser leurs URLs et authentifications en plus de l'importation et exportation des données en JSON [25].



GitHub est un service web d'hébergement et de gestion collaborative de développement de logiciels basé sur le programme Git. De manière résumée, à travers GitHub

vous pouvez accéder aux fichiers d'un projet que quelqu'un d'autre a créé ou bien créer votre projet et permettre à d'autres utilisateurs d'y accéder. L'historique de l'évolution des projets dans github est également conservé, ce qui, au besoin, permet de pouvoir revenir à des versions antérieures. À cela s'ajoute bien d'autres fonctionnalités [27].

4.6.3 Environnement de programmation



Flutter est un kit de développement de logiciel (SDK) d'interface utilisateur open-source créé par Google il est utilisé pour développer des applications multiplateformes pour Android, iOS, Linux, Mac, Windows, Google Fuchsia et le web à partir d'une seule base de code [3].



MySQL est un SGBD (Système de Gestion de Base de Données). Son rôle est d'enregistrer des données de manière organisée afin de vous aider à les retrouver facilement plus tard. C'est grâce à MySQL que vous pourrez enregistrer la liste des membres de votre site, etc. Le langage qui permet de communiquer avec la base de données s'appelle SQL [28].



TypeScript est un langage de programmation qui ajoute des types optionnels à JavaScript ainsi que la création de classes et d'interfaces, l'import de modules, tout en conservant l'approche non-constricte de JavaScript. Il peut être installé via outil NPM (*Node Package Manager*) [8].



JSON «JavaScript Object Notation» est un format d'échange de données, facile à lire par un être humain et il est interprété par la machine. Il est basé sur JavaScript, il utilise des conventions qui sont communes à toutes les langages de programmation [30].



Apparu en 2004, Google Maps est un service gratuit de cartographie en ligne. La possibilité de naviguer à travers le monde entier de manière fluide avec une simple connexion Internet a fait de ce projet un véritable succès. Le succès de cette API a été et reste toujours la mise à disposition d'une interface de programmation gratuite à destination des développeurs, leur permettant d'intégrer de la cartographie dynamique dans leurs applications [19].

Git

Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, auteur du noyau Linux. En 2016, il s'agit du logiciel de gestion de versions le plus populaire qui est utilisé par plus de douze millions de personnes [30].

4.6.4 Elaboration de l'API

Pour le développement de l'API de notre application nous avons choisi l'environnement d'exécution Nodejs v14.17.3, le framework Express v4.17.3, le langage Typescript v4.6.2 et JSON pour un langage d'échange de données textuelles.

Anatomie de notre API

- **dist** : un clone de l'arborescence **src** dont tous les fichiers sont **.js**.
- **node_modules** : tous les packages nécessaires au fonctionnement de l'API sont stockés ici.
- **src/controllers** : contient tous les services fournis par l'API sous forme de fonctions exportables.
- **src/data** : contient des fichiers **.json** des données nécessaire au remplissage de la base de données.
- **src/entities** : contient toutes les fonctions nécessaire pour la gestion de la base de données.
- **src/routes** : contient les points de terminaison de l'API.
- **src/types** : contient les types énumérés.
- **package.json** : représente le fichier de configuration de l'application.
- **tsconfig.json** : il s'agit d'un fichier de configuration du compilateur Typescript.
- **.env** : il s'agit d'un fichier contenant les variables d'environnement.

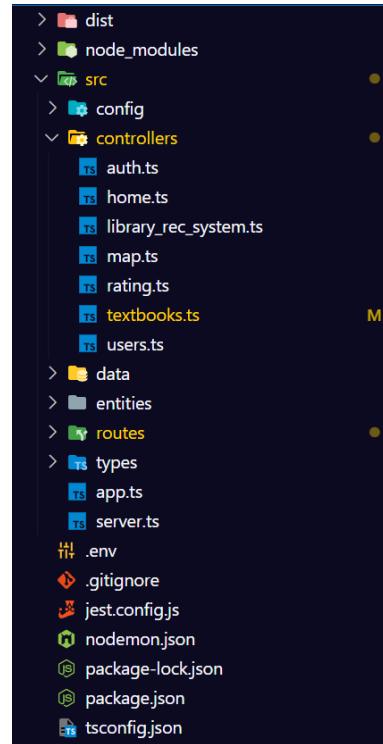


FIGURE 4.19 – Structure de notre API

Afin d'illustrer la récupération des informations en exploitant l'API, prenons un exemple d'une demande d'un ouvrage spécifique tout en envoyant une requête à l'URI **http://localhost:8000/api/textbooks/:id**. La figure (4.20) suivante représente le résultat récupéré via l'outil **Postman**.

The screenshot shows a JSON response from a REST API. The response body is a single object with the following structure:

```

1  "found_textbook": {
2      "id": 554,
3      "title": "Unlocking Android",
4      "author": [
5          "W. Frank Ableson",
6          "Charlie Collins",
7          "Robi Sen"
8      ],
9      "cover_image_link": "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-images/ableson.jpg",
10     "desc": "Unlocking Android: A Developer's Guide provides concise, hands-on instruction for the Android operating system and development tools. This book teaches important architectural concepts in a straightforward writing style and builds on this with practical and useful examples throughout.",
11     "field": "Computer Science",
12     "nb_copies": 8,
13     "page_count": 416,
14     "created_at": "2022-04-14T13:38:26.507Z",
15     "updated_at": "2022-04-14T13:38:26.507Z",
16     "categories": [
17         "Open Source",
18         "Mobile",
19         "Operating System"
20     ]
21 }
22
23

```

FIGURE 4.20 – L'objet de la réponse en JSON

Notre API est un composant logiciel que nous avons créé et intégré dans le code source du système d'information afin de séparer les deux parties Front-end et Back-end et donner une meilleure connectivité aux utilisateurs présentée par les différentes interfaces que nous allons présenter par la suite.

La figure ci-dessous 4.21 illustre la structure de notre projet Flutter

Anatomie de notre projet Flutter

- **android/** : contient les fichiers et dossiers nécessaires à l'exécution de l'application sur un système d'exploitation **Android**. Ces fichiers et dossiers sont autogénérés lors de la création du projet de flutter.
- **assets/** : le dossier contient tous les fichiers statiques intégrés dans l'application (images, icons, font .. etc).
- **ios/** : Comme le dossier Android, ce dossier contient les fichiers requis par l'application pour exécuter son code sur le système d'exploitation **iOS**. Les fichiers principaux sous le dossier ios sont le dossier *Assets.xcassests* et le fichier *info.plist*.
- **lib/** : c'est le dossier le plus important du projet, utilisé pour écrire la plupart du code. Par défaut, le dossier lib contient le fichier *main.dart*, qui est le point d'entrée de l'application.
- **.packages** : ce fichier contient le chemin de chaque package et bibliothèque utilisés dans le projet.
- **pubspec.yaml** : ce fichier est utilisé pour ajouter des métadonnées et une configuration spécifique à notre application. il permet la configuration des dépendances telles que les ressources d'image, les polices et les versions d'applications.

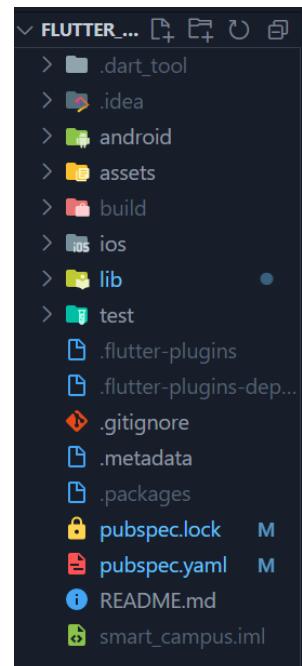


FIGURE 4.21 – Structure de notre projet Flutter

4.6.5 Simulation et résultats

Dans ce qui suit, nous allons présenter un schéma de navigation ainsi que quelques interfaces de l'application mobile et un aperçu de quelques fonctions métiers.

4.6.5.1 Schéma de navigation de l'application

Un schéma de navigation de l'application est un diagramme qui représente une vue globale sur les pages fournies par l'application sous forme d'une arborescence. Il s'agit donc d'un enchaînement logique des services. Son but est d'éclairer et illustrer l'emplacement de chaque page lors de l'utilisation.

Pour notre application il existe deux modes de navigation, le mode **authentifié** et le mode **invité(e)** et par la suite, l'accès aux différentes pages de l'application dépend du mode d'utilisation. La figure (4.22) ci-dessous illustre l'arborescence de la navigation.

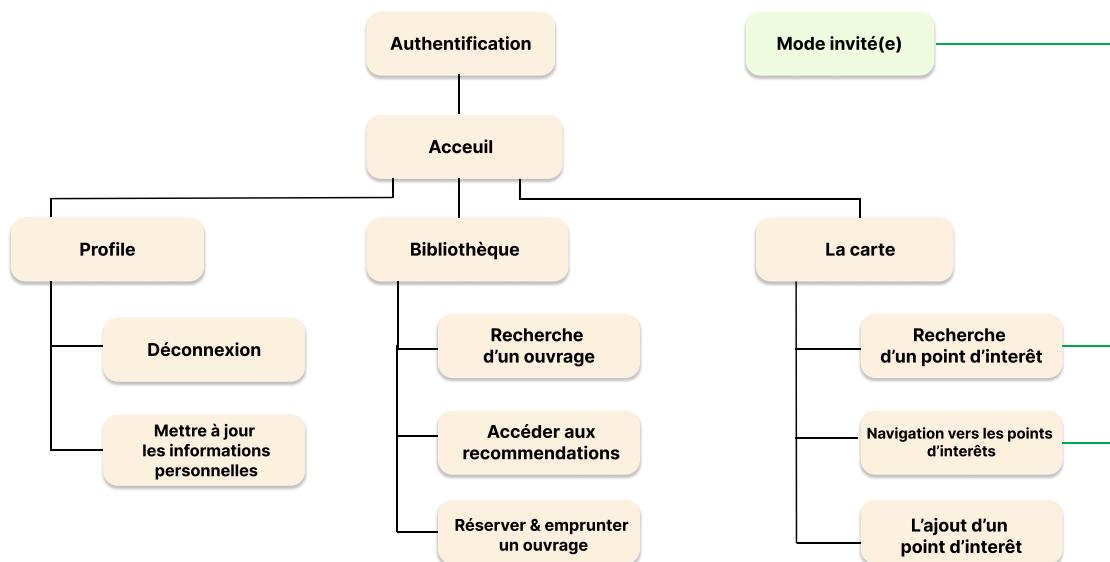


FIGURE 4.22 – Schéma de navigation de notre application

4.6.5.2 Premier cas d'utilisation : authentification

La figure 4.24 offre un aperçu sur la page d'authentification ainsi que la page d'inscription. Dans cette interface, nous pouvons voir les deux champs '*ID*' et '*Mot de passe*' que l'étudiant doit saisir, puis il aura à cliquer sur le bouton '*Connexion*' afin de s'authentifier.

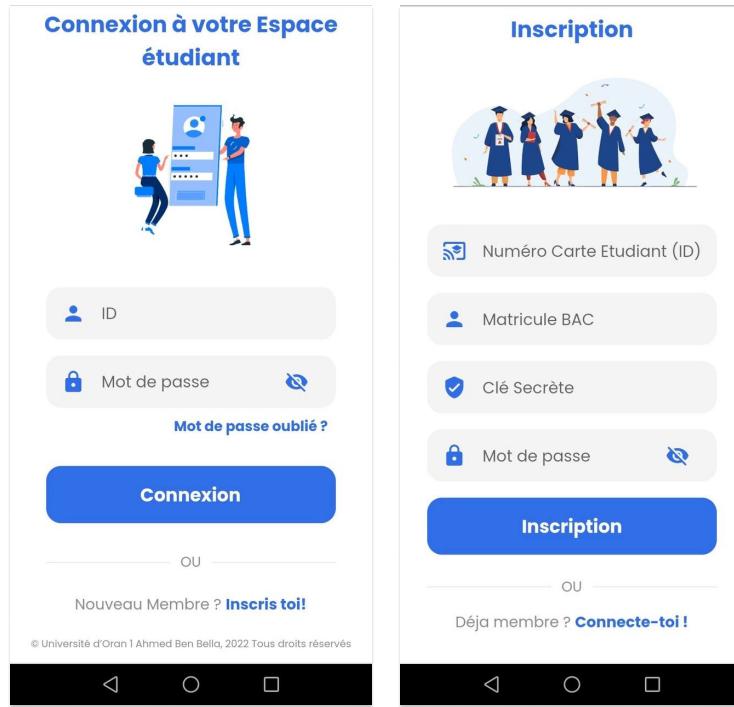


FIGURE 4.23 – Interface 'Connexion' et 'Inscription'

```

1 void connectUser() async {
2     final id = idInput.text;
3     final password = passwordInput.text;
4     User user = new User(id, password);
5     var res = await network.postRequest(user, "auth/login");
6     if (res.statusCode == 200) {
7         final data = jsonDecode(res.body);
8
9         SharedPreferences prefs = await SharedPreferences.getInstance();
10        prefs.setString('token', jsonDecode(res.body)['token']);
11        setState(() {
12            isLoading = false;
13        });
14        Fluttertoast.showToast(
15            msg: '' + data['msg'] + '',
16            toastLength: Toast.LENGTH_SHORT,
17            gravity: ToastGravity.BOTTOM,
18            backgroundColor: Color.fromARGB(255, 24, 0, 90),
19            textColor: Color.fromARGB(255, 255, 255, 255));
20        Navigator.push(
21            context, MaterialPageRoute(builder: (context) => CampusMap()));
22    } else {
23        setState(() {
24            errorMessage = true;
25            isLoading = false;
26        });
27    }
28}

```

FIGURE 4.24 – Corps de la méthode connectUser()

La méthode connectUser() permet d’authentifier l’étudiant en envoyant l’objet ‘user’ pour une vérification.

```
● ● ●  
1 export const login: RequestHandler = async (req: Request, res: Response) => {  
2   const { matricule, password } = req.body;  
3  
4   //check if the user exists in the table of the users  
5   const user: User | undefined = await getRepository(User).findOne(  
6     { matricule },  
7     { relations: ["student_id"] }  
8   );  
9   console.log(user);  
10  if (!user) {  
11    return res.status(404).json({ msg: "user was not found" });  
12  }  
13  
14  //check if the password matches  
15  const isCorrectPassword: boolean = await bcrypt.compare(  
16    password,  
17    user.password  
18  );  
19  
20  if (!isCorrectPassword)  
21    return res.status(401).json({ msg: "Wrong password" });  
22  
23  //generate a new session for the user  
24  req.session.user_id = user.id;  
25  const token = jwt.sign({ id: user.id }, "secretKey");  
26  return res.status(200).json(  
27    { msg: `Utilisateur - ${user.student_id.last_name} a été connecté avec succès`,  
28      token,  
29    }  
30  );
```

FIGURE 4.25 – Gestion d’authentification par le serveur

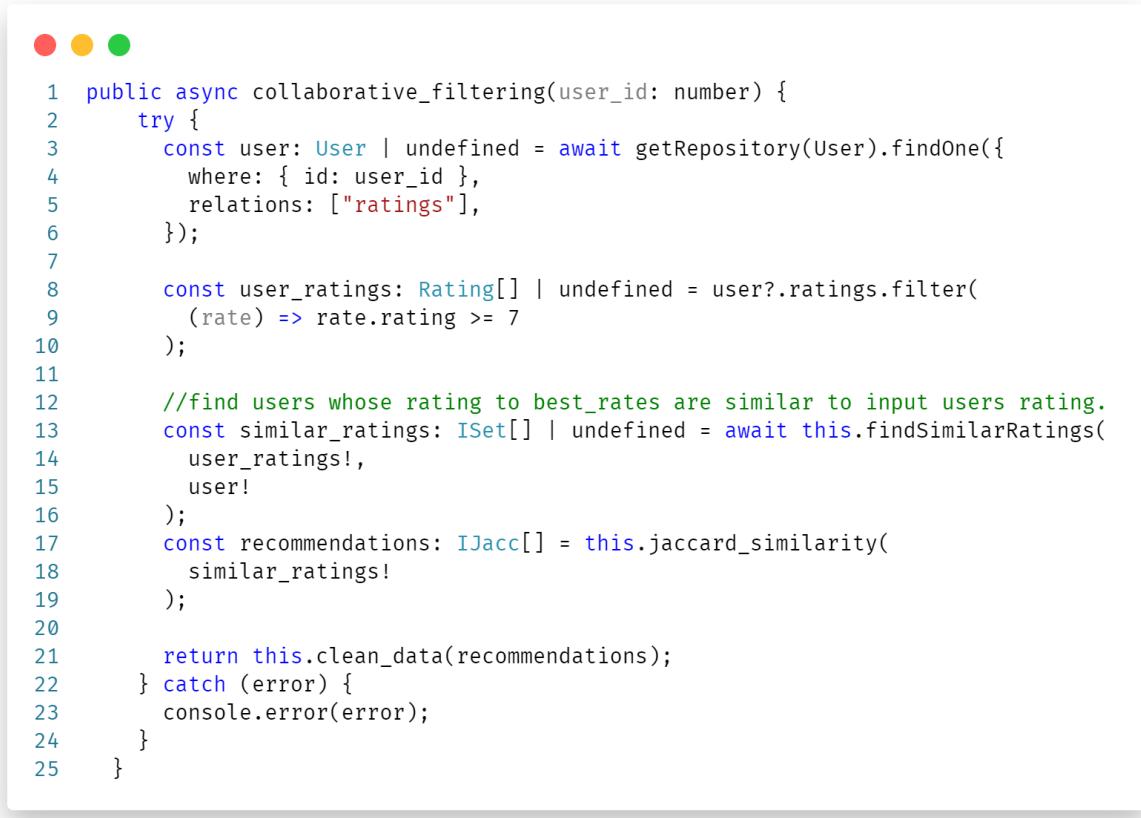
Le code côté serveur dans la figure 4.25 vérifie si l’identifiant et le mot de passe envoyé par l’interface existent dans la table des utilisateurs, si oui il crée une session pour l’utilisateur afin de naviguer dans l’application sinon il envoie un message d’erreur.

4.6.5.3 Deuxième cas d’utilisation : recommandation et réservation d’ouvrage

A travers ce service, l’utilisateur est d’abord accueilli par une liste d’ouvrages recommandés utilisant la technique de **filtrage collaboratif** basée sur la mémoire présentée dans le troisième chapitre. La figure 4.26 illustre le code derrière la recommandation.

De plus, L’utilisateur a la possibilité de chercher et réserver un ouvrage tout en spécifiant le nom de l’ouvrage et en cliquant sur le bouton de la réservation présent dans la page. Ainsi

que l'utilisateur a la possibilité d'évaluer les ouvrages avec une note allant de worst-rate à best-rate.



```

1  public async collaborative_filtering(user_id: number) {
2      try {
3          const user: User | undefined = await getRepository(User).findOne({
4              where: { id: user_id },
5              relations: ["ratings"],
6          });
7
8          const user_ratings: Rating[] | undefined = user?.ratings.filter(
9              (rate) => rate.rating >= 7
10         );
11
12         //find users whose rating to best_rates are similar to input users rating.
13         const similar_ratings: ISet[] | undefined = await this.findSimilarRatings(
14             user_ratings!,
15             user!
16         );
17         const recommendations: IJacc[] = this.jaccard_similarity(
18             similar_ratings!
19         );
20
21         return this.clean_data(recommendations);
22     } catch (error) {
23         console.error(error);
24     }
25 }

```

FIGURE 4.26 – La fonction derrière le filtrage collaboratif.

La figure (4.27) suivante illustre les interfaces du service Bibliothèque.

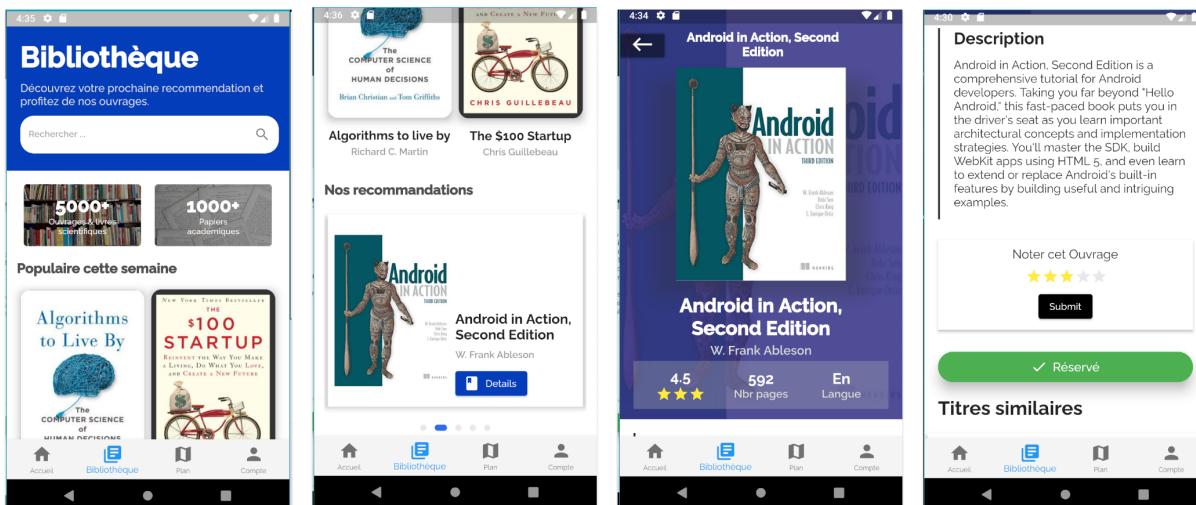


FIGURE 4.27 – Interfaces du service Bibliothèque

4.6.5.4 Troisième cas d'utilisation : la carte du Campus et la géolocalisation

Les figures 4.28 et 4.29 montre l'ancien usage mis à disposition des étudiants et enseignants par le département d'informatique, pour la localisation des lieux de **déroulement des examen**, soutenances, etc. (e.g. ISTA), pour cela notre application propose à travers ses interfaces une manière plus pratique et dynamique de localiser ces lieux et de naviguer vers ces derniers ainsi que la possibilité d'avoir plus d'informations (Numéro de téléphone, Adresse mail, Horaires d'ouvertures, Détails, ...).

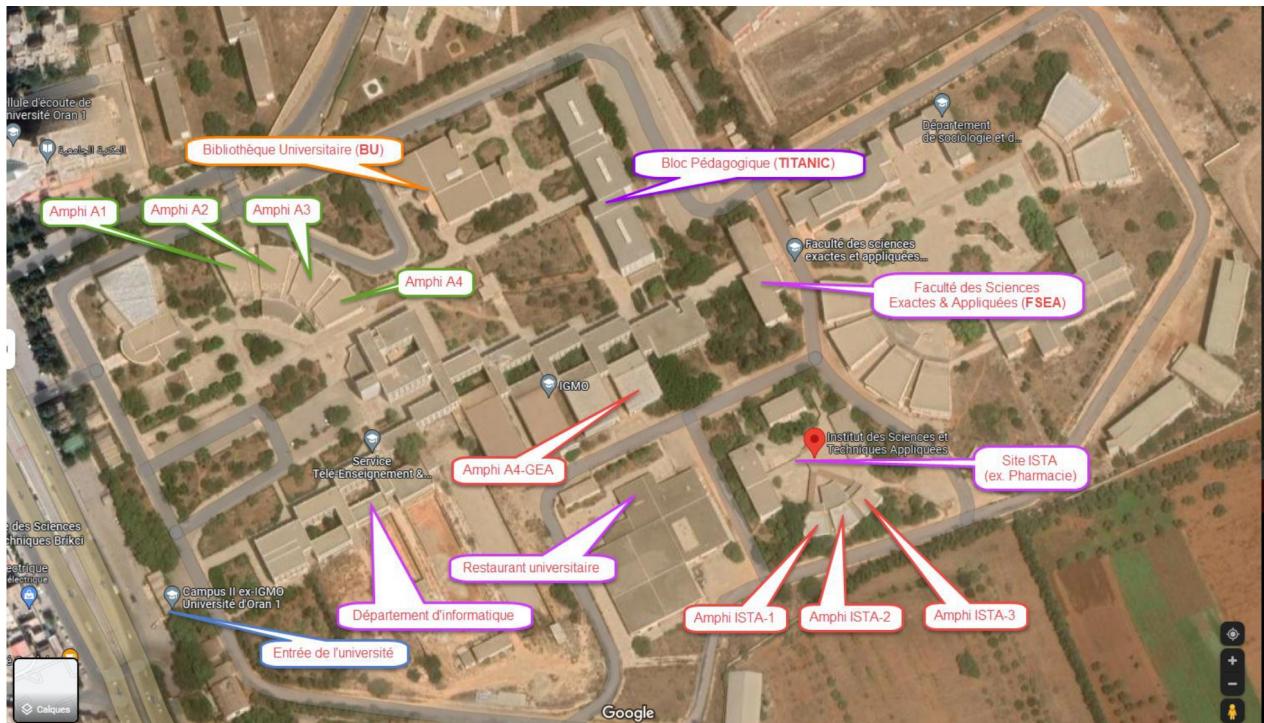


FIGURE 4.28 – Guide de géolocalisation manuelle envoyé par le département.

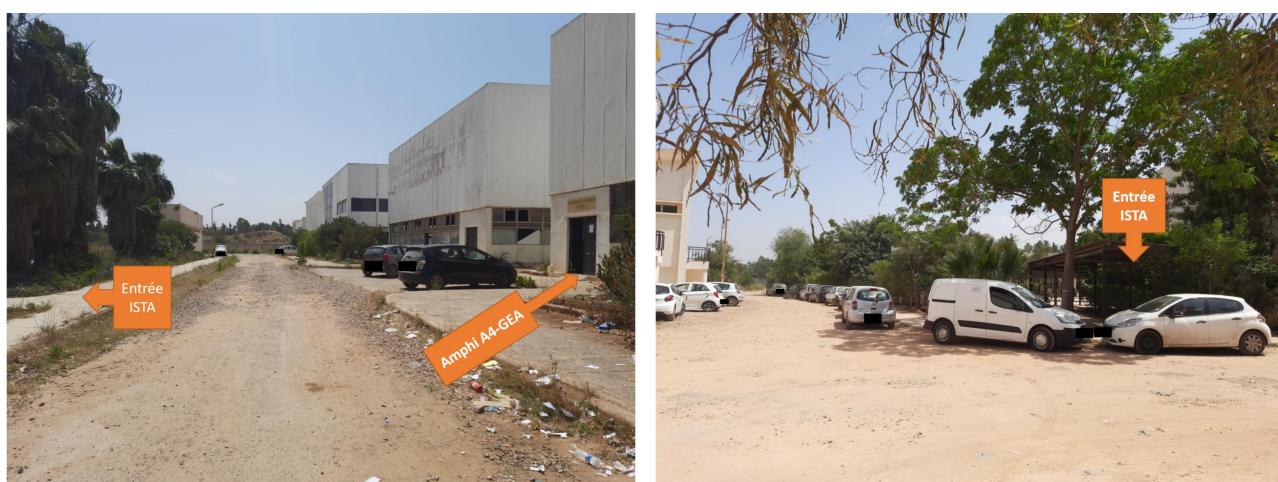


FIGURE 4.29 – Photos envoyé par le département.

Les figures suivantes 4.30 et 4.28 illustrent respectivement les interfaces du service géolocalisation ainsi que le code dart derrière.

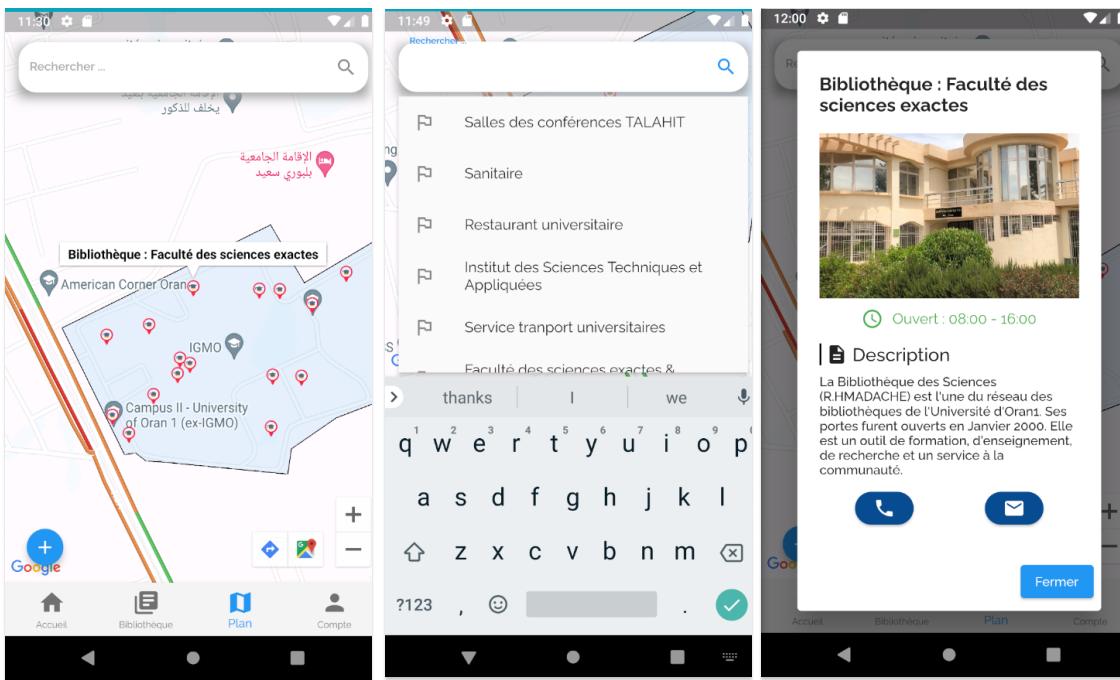


FIGURE 4.30 – Interfaces du service de géolocalisation



```
1  @override
2      Widget build(BuildContext context) {
3          //if the user's location hasn't been fetched yet, display loader on the screen
4          if (myLocation == null) {
5              return MapLoader();
6          }
7          return Scaffold(
8              body: AnimatedOpacity(
9                  opacity: _opacity,
10                 duration: const Duration(seconds: 3),
11                 child: Stack(children: <Widget>[
12                     GoogleMap(
13                         mapToolbarEnabled: true,
14                         zoomControlsEnabled: true,
15                         polygons: _campusPolygon,
16                         myLocationButtonEnabled: true,
17                         mapType: MapType.terrain,
18                         markers: _markers,
19                         compassEnabled: false,
20                         trafficEnabled: true,
21                         indoorViewEnabled: true,
22                         myLocationEnabled: false,
23                         initialCameraPosition: _currentLocation!,
24                         onMapCreated: (controller) {
25                             _controller.complete(controller);
26                         },
27                     )

```

FIGURE 4.31 – Le code derrière l’interface du service de géolocalisation

4.7 Conclusion

Ce dernier chapitre est une présentation des aspects pratiques liés au processus de développement de notre projet et de ce que nous avons pu accomplir. Tout d’abord, nous avons présenté une architecture globale de notre application pour expliquer l’idée globale de l’approche suivie. Ensuite une étude conceptuelle du projet par les diagrammes UML (cas d’utilisation et séquence) et un modèle du système d’information. En outre, nous avons précisé brièvement les langages et les outils impliqués dans notre environnement de travail ainsi que l’élaboration d’une API propre au système et enfin, les différentes vues de l’application mobile avec un aperçu de quelques fonctions métiers.

CONCLUSION GÉNÉRALE

L'objectif de ce projet de fin d'étude était la conception et la réalisation d'un système intelligent de gestion d'un campus universitaire sous forme d'une application mobile multi-plateformes, pour notre université Oran1 Ahmed ben bella.

Nous avons choisi de nous focaliser sur la conception et la mise en œuvre des deux principaux services dont nous avons constatés un vrai besoin par les enseignants, les invités et les étudiants y compris nous-mêmes.

Le cœur de notre projet a consisté au développement d'une carte de localisation au sein du campus, un système de recommandation avec la gestion des réservations des ouvrages tout en utilisant les dernières technologies telles que Git, le framework Flutter et l'environnement d'exécution Nodejs.

Nous avons tenu compte de l'existant afin de collecter tous les besoins possibles des utilisateurs de l'application et ça nous a permis de fixer une structure globale de l'application. De plus, nous nous sommes assurés de respecter toutes les normes de conception mis en disposition afin de réaliser les fonctionnalités de l'application.

La réalisation de ce projet nous a aidé à approfondir les notions acquises pendant notre cursus universitaire en cycle de licence et à pratiquer ces dernières à travers une variété de technologies tout en suivant les méthodes de modélisation et conception fondamentales et par la suite nous préparer au monde professionnel.

Enfin, nous espérons sincèrement que ce modeste travail servira d'excellentes ressources pour tout ceux qui souhaitent s'engager à des projets semblables. Finalement, c'était l'un des principaux objectifs de ce dernier.

Perspectives

Notre réalisation de ce projet n'était qu'une première version. Nous croyons fermement que ce projet présente un grand potentiel. En effet, les principaux points de vue qui sous-tend ce

projet sont les suivants :

Nous avons choisi de développer une API complète pour notre application afin que celle-ci puisse être utilisée par d'autres applications futures ou même plus, pour le développement d'une version ou une extension Web pour l'application.

En raison de la durée limitée du projet, nous avons choisi de nous concentrer sur deux services principaux, mais nous avons laissé la porte ouverte aux contributions futures concernant les autres services.

Dans le cadre d'une application mobile dédiée au service universitaire plusieurs fonctionnalités peuvent être proposées et notre application peut toutes les englober.

BIBLIOGRAPHIE

- [1] Mane, GHERARI Mlle. Contribution à L'évolution des Architectures Logicielles des Systèmes Intensifs. Tebessa : s.n. Thèse de doctorat.
- [2] Comprendre le développement d'application (2018, 30 mai). MBA MCI. Consulté 24 avril 2022, à l'adresse <https://mbamci.com/comprendre-le-developpement-d-application/>
- [3] Développement d'une application de calendrier partagé, React Native ou Flutter. Neuen-schwander Hervé. Consulté 24 avril 2022, à l'adresse https://doc.rero.ch/record/329789/files/M_moire_HNeunschwander.pdf
- [4]] I. (s. d.). Ionic Framework - Ionic Documentation. Ionic Docs. Consulté mars 2020, à l'adresse <https://ionicframework.com/docs>
- [5] Architectural overview of Cordova platform - Apache Cordova. (s. d.). Apache Cordova. Consulté mai 2020, à l'adresse <https://cordova.apache.org/docs/en/latest/guide/overview/index.html>
- [6] Aguilar, C. (2018, 18 juin). Reactive Core architecture for React Native and React applications. Medium. Consulté 27 mai 2020, à l'adresse <https://medium.com/kuralabs-engineering/reactive-core-architecture-for-react-native-and-react-applications-d590daf4ef8a>
- [7] Jaiswal, S. (2018). Difference between Ionic and React Native - javatpoint. www.javatpoint.com. <https://www.javatpoint.com/ionic-vs-react-native>
- [8] Schüpbach, L. (2018, September). Développement d'une application mobile d'échange de parking, quel Framework choisir ? (Thèse de Magister). Haute École de Gestion de Genève (HEG-GE). <https://doc.rero.ch/record/323721/files/Memoire.pdf>
- [9] Statista. (2020, juin). Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020 [Illustration]. statista. <https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/>
- [10] Understanding TypeScript by Gavin Bierman, Martín Abadi Mads Torgersen Conference paper. part of the Lecture Notes in Computer Science book series (LNCS, volume 8586). Consulté le 19 avril 2022 à l'adresse https://link.springer.com/chapter/10.1007/978-3-662-44202-9_11

- [11] A Survey on Recommendation System by Debasish Das, LaxmanSahoo and SujoyDatta from *International Journal of Computer Applications (0975 -8887) Volume 160 -No 7, February 2017*. Consulté le 6 Mai 2022
- [12] Les Moteurs et systèmes de recommandation par Imad Saleh. Consulté le 6 Mai 2022
- [13] Systèmes de recommandation par Elsa Negre. Consulté le 6 Mai 2022
- [14] Un nouveau système de filtrage collaboratif basé sur le modèle des espaces de communautés An Te NGUYEN, Thèse de doctorat. Consulté le 8 Mai 2022
- [15] Étude des systèmes de recommandations et mise en pratique des algorithmes. par Bastin, Jérémie. Thèse de Master. Consulté le 8 Mai 2022
- [16] Collaborative Filtering: basic ideas(slides based on chapter 2 of Programming Collective Intelligence book by Toby Segaran) par Fernando Lobo, diapo n°=7. Consulté le 9 Mai 2022
- [17] Un système de gestion et de recommandation d'articles de recherche par Naak Amine. Consulté le 08 Mai 2022
- [18] Standford paper - Chapter 9 Recommendation Systems. Consulté le 27 Mai 2022.
- [19] Développer avec les API Google Maps. Consulté le 12 Juin 2022.
- [20] Diagramme MCD <https://web.maths.unsw.edu.au/~lafaye/CCM/merise/mcd.htm#:~:text=Le%20mod%C3%A8le>
- [21] Idir Benouaret. "Un système de recommandation contextuel et composite pour la visite personnalisée de sites culturels". Autre [cs.OH]. Université de Technologie de Compiègne, 2017. Consulté le 04 Juin 2022.
- [22] Système de recommandation, Wikipédia. Consulté le 6 Mai à l'adresse https://fr.wikipedia.org/wiki/Système_de_recommandation
- [23] Visual Studio Code. https://en.wikipedia.org/wiki/Visual_Studio_Code.
- [24] MySQL Workbench <https://downloads.mysql.com/docs/workbench-en.pdf>
- [25] Postman [https://fr.wikipedia.org/wiki/Postman_\(logiciel\)](https://fr.wikipedia.org/wiki/Postman_(logiciel))
- [26] Trello, un outil de gestion de projet en ligne <https://www.pedagogie.ac-aix-marseille.fr/upload/docs/application/pdf/2020-03/trello.pdf>
- [27] GitHub <http://projet.eu.org/pedago/sin/tutos/github.pdf>
- [28] MySQL <https://dspace.univ-tlemcen.dz/bitstream/112/9263/1/Gestion-des-Emplois-du-Temps.PDF>
- [29] Typescript <https://www.npmjs.com/package/typescript>
- [30] Amosse EDOUARD. cour COMPRENDRE L'ARCHITECTURE DES WEB SERVICES REST.
- [31] Git <https://fr.wikipedia.org/wiki/Git>.
- [32] Gélocalisation <https://www.schoolmouv.fr/enseignants/cours/geolocalisation/fiche-de-cours>
- [33] Ricci, F., Rokach, L., Shapira, B. (2011). Introduction to Recommender Systems Handbook. In : Ricci, F., Rokach, L., Shapira, B., Kantor, P. (eds) Recommender Systems Handbook. Springer, Boston, MA.