Université Moulay Ismaïl
Faculté des Sciences et Techniques Errachidia
Département d'Informatique
LST en Génie Logiciel

Rapport du Projet de Fin d'Études

En vue de l'obtention du diplôme Licence en Sciences et Techniques
Option : Génie Logiciel

Thème

Conception et réalisation d'une application mobile : « eBook Store »

Réalisé par :
AIT MAKHAD Abdelilah
BADRY Ibrahym
EDDAHBY Mohamed

Encadré par :
Prof. YAAKOUBI Fouad

Devant le jury composé de :
Prof. BAATAOUI Aziz
Prof. TAIFI Khaddouj
Prof. YAAKOUBI Fouad

A.U 2021-2022

Dédicace

À nos chers parents, pour leurs amour et sacrifices.

À nos adorables frères, sœurs pour leur patience.

À nos proches amis et toute notre grande famille, pour leurs soutient et encouragements.

À toutes les personnes qui nous connaissent de près ou de loin seulement pour leur existence.

Remerciements

Tout au long de notre carrière universitaire, nous avons eu le plaisir de faire connaissance de plusieurs personnes, qui directement ou indirectement ont contribué à notre formation et nous ont permis de progresser d'une année à l'autre.

En premier lieu, nous remercions notre professeur Monsieur Fouad Yaakoubi qui n'a pas cessé de nous encourager pendant la durée du projet, pour ses conseils toujours très avisés ainsi que pour sa générosité en matière de formation et d'encadrement.

Ces remerciements vont aussi au corps professoral et administratif de la Faculté des Sciences et Techniques d'Errachidia, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nos remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre projet en acceptant d'examiner ce travail.

Finalement, nous adressons nos plus sincères remerciements à tous nos familles pour leurs énormes sacrifices et nos amis(es), avec qui nous avons passé des moments très agréables à la FSTE, pour leur assistance dans les moments difficiles et qui nous ont toujours encouragés au cours de la réalisation de ce projet.

Merci infiniment !

Résumé

Notre objectif principal est de développer une application mobile hybride (cross-plateforme) fiable et flexible, qui est une boutique de livres électroniques (e-book store). Pour cela nous avons dû, s'initier à concevoir en UML, manipuler des langages comme Dart et Flutter et maîtriser un environnement de développement dédié à ce type d'application. Dans notre application nous avons choisi la plateforme de développement Android Studio et VS Code. Notre application consiste à consulter et acheter les livres facilement.

Enfin, il faut noter que comme tous les projets, celui-ci est destiné à évoluer au fil du temps.

Mots clés: Flutter, Dart, Android, Cross-platform, Book, Store.

Table des matières

Liste des figures

Liste des abréviations

API : Application Programming Interface
FSTE : Faculté Sciences et Technique Errachidia
IDE : Integrated Development Environment
GPS : Global Positioning System
GSM : Global System for Mobile
iOS : iPhone Operating System
JOSN : JavaScript Object Notation
NoSQL : Not only Structured Query Language
OS : Operating System
RAM : Random Access Memory
REST : REpresentational State Transfer
SDK : Software Development Kit
SGBD : Système de Gestion de Base de Données
UMI : Université Moulay Ismaïl
UML : Unified Modelling Language
URL : Uniform Resource Locator

Introduction générale

Ces dernières années, le marché du mobile s'est développé très rapidement. Les utilisateurs naviguent plus sur leurs smartphones ou tablettes que sur les ordinateurs, car nous n'utilisons plus les Smartphones que pour passer des appels. Le plus motivant dans tout cela, c'est que la demande en application mobile ne fait qu'augmenter d'année en année et dans tous les domaines.

Tant que le marché du Smartphone est en croissance continue, bien sûr le marché des applications mobile le sera également. De telles applications mobiles qu'on retrouve chez tout le monde permettent de jouer, de consulter des e-mails, de naviguer sur les réseaux sociaux, d'accéder à des services administratifs, d'émettre des requêtes vers une base de données, etc. De manière générale, et dans le cadre de notre projet, une application mobile est très utile pour supporter certaine options mobiles (service à distance, publicité, message d'urgence, notification administrative, …) d'un système d'information donné.

Présentation du cadre de notre projet :

La mobilité devient un facteur clé de l'informatique ce qui permet aux utilisateurs d'accéder à leurs systèmes d'information n'importe quand et à n'importe où. Les téléphones deviennent de véritables petits ordinateurs et offrent des capacités encore sous-exploitées.

Dans ce contexte, nous proposons l'étude, la conception, et l'implémentation d'une application mobile et l'intégrer dans un système de gestion e-commerce des livres, pour offrir l'ensemble de fonctionnalités du système aux utilisateurs.

Chapitre 1 : Environnement Mobile et Développement des Applications Multi-plateformes

1. Introduction :

Le marché de la téléphonie portable connaît actuellement une véritable révolution, d'un simple téléphone portable pour émettre des appels à un téléphone évolué doté de capacités proches d'un véritable ordinateur appelé Smartphone. Légers, puissants et intelligents, ces Smartphones remplaceront de plus en plus l'équipement de téléphonie standard dans les boutiques. Les applications mobiles pourront être utilisées par tout le monde.

Lors du développement d'application mobile native, les application Android sont écrites en Java, et celles pour iOS en Swift et Objective-C. Il ne fait aucun doute que cette approche apporte des applications sans défaut et très performantes. Mais, le développement que cette approche suit est très chronophage et coûteux car le même code doit être écrit deux fois. Quelles sont les solutions pour développer une application mobile et quelle solution choisir ?

2. Environnement mobile et développement des applications :

2.1 Smartphone :

Un smartphone désigne un téléphone portable multifonctions qui a la capacité de naviguer sur Internet, lire des musiques et des films, équipé d'une puce GPS, d'un écran tactile, qui peut évoluer avec le temps à l'aide de mises à jour, et qui a la capacité de télécharger et installer de nouvelles applications.

2.2 Systèmes d'exploitation mobile :

Un système d'exploitation est un ensemble de programmes de base qui contrôle le fonctionnement interne d'un ordinateur et dirige son exploitation par les logiciels applicatifs.

Cette définition s'applique aussi aux Smartphones que nous retrouvons sur le marché, car ils possèdent les mêmes caractéristiques et structures qu'un ordinateur à savoir un système d'exploitation, un processeur, une mémoire interne RAM et une mémoire de stockage auquel s'ajoute un ensemble de ports de connexion de périphériques qu'il doit gérer.

a. Android

Android est le système d'exploitation mobile développé par Google. Il est utilisé par plusieurs Smartphones et tablettes. Les exemples incluent Google Pixel 4, la série Samsung Galaxy, et Sony Xperia les nouveaux du marché comme les Smartphones Xiaomi et Oppo pour ne citer que ceux-là.

Le système d'exploitation Android est basé sur le noyau Linux. Contrairement à iOS d'Apple, Android est open source, ce qui signifie que les développeurs peuvent modifier et personnaliser le système d'exploitation de chaque téléphone. Par conséquent, différents téléphones basés sur Android ont souvent des interfaces graphiques différentes même s'ils utilisent le même système d'exploitation.

b. iOS

iOS est un système d'exploitation mobile développé par Apple. Il s'appelait à l'origine iPhone OS, mais a été renommé iOS en juin 2009. L'iOS fonctionne actuellement sur iPhone, iPod Touch et iPad.

2.3 Application mobile

Une application mobile est un logiciel spécifiquement développé pour fonctionner sur un périphérique mobile à interface tactile. Cette définition prise au sens large inclut les tablettes et les montres intelligentes. Mais si l'on n'apporte pas de précision supplémentaire, tout le monde comprend que vous parlez d'une application développée pour les téléphones intelligents.

a. Caractéristiques des applications mobiles

La conception et l'implémentation des applications mobiles sont fortement liées aux systèmes d'exploitation de l'appareil mobile sur lequel ces applications vont être installées et exécutées.

Cependant, il reste toujours quelques caractéristiques communes aux applications mobiles malgré la diversité des systèmes d'exploitation mobiles et la disparité des plates-formes de systèmes d'exploitation qui les gèrent.

- Temps de lancement : l'application doit se lancer en moins de 5-6 seconds.
- Haute performance : garantir constamment une haute performance et une rapide réactivité afin de répondre aux exigences de l'utilisateur qui cherche la fiabilité dans l'application utilisée.
- Pas d'étapes intermédiaires : afin d'atteindre le but d'exécution de l'application.
- Consommation d'énergie : l'optimisation du processus de l'application pour garantir une exploitation efficace de l'énergie.

b. Types d'applications mobiles

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.

3.3.1 Applications natives

Les applications natives sont des programmes développés pour exécuter certaines fonctionnalités et être déployables dans des plateformes ou appareils mobiles particuliers. Ces derniers permettent aux applications de prendre l'avantage de leurs caractéristiques, leurs applications préinstallées par défaut (ex., calendrier et contact), et aussi les technologies qu'elles fournissent comme la Caméra, le GPS (localisation) ou encore l'espace de stockage et les versions ou les types des périphériques.

Les applications natives peuvent être téléchargées depuis des boutiques en ligne publiques ou privées, ensuite être installées sur l'appareil mobile ou bien elles peuvent exister par défaut avec l'appareil mobile (ex. Contact, Messagerie et Calendrier, etc..).

✓ **Avantages**

- Une meilleure performance parce qu'elles utilisent les logiciels et le matériel valable dans l'appareil mobile.
- Comme ces applications sont déjà installées dans l'appareil et utilisent les données déjà existantes, elles peuvent être exécutées hors ligne sans besoin d'Internet.
- Chaque application est distinguée par son logo pour attirer l'attention de l'utilisateur.

– **Inconvénients**

- L'inconvénient majeur de ce type d'application, c'est qu'elles sont fortement liées avec l'appareil sur lequel elles sont installées, ce qui engendre l'impossibilité d'évoluer ou d'exploiter de nouvelles technologies.
- Rendre les applications natives déployables dans différents types d'appareils/plateformes (c.-à-d. réécrire le code dans différents langages) est une tâche consommatrice en termes de temps.

- L’exclusivité des applications natives pour un type d’appareil mobile bien particulier diminuera le nombre des utilisateurs et le gain de leurs ventes.

3.3.2 Applications mobiles web

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.

Lors du développement des applications mobiles web, il est essentiel de :

- Définir le texte et les images statiques de l’application en utilisant HTML5.
- Définir le style et la présentation de l’application en utilisant par exemple CSS.
- Définir l’interaction et l’animation au moyen d’un outil comme JavaScript.

Une meilleure utilisation de ce genre d’application est motivée par la possibilité d’une réorganisation de leurs contenus en fonction des caractéristiques de l’appareil mobile.

✓ **Avantages**

- L’avantage le plus important des applications mobiles web est la capacité de déploiement sur les multiples plateformes indépendamment du type d’appareil mobile.
- Moins cher, facile et rapide à développer.
- Accès facile en utilisant un simple URL sans le besoin d’installation ou le téléchargement des compléments.

– **Inconvénients**

- Les navigateurs traditionnels sont plus performants que les navigateurs des appareils mobiles.
- Les applications mobiles web ne peuvent pas exploiter les fonctionnalités offertes par les logiciels et les hardwares des appareils mobiles.
- La performance des applications mobiles web dépend de la vitesse et l’état de la connexion via Internet ou les réseaux de téléphonie mobile GSM.

3.3.3 Applications mobiles multi-plateformes (hybrides)

Lors du développement des applications mobiles natives, les applications Android sont écrites en Java et celles d'iOS en Swift et Objective-C. Cette approche fournit des applications sans faille et très performantes. Mais, le chemin de développement que cette approche suit est assez long et coûteux car le même code doit être écrit deux fois. Ainsi,

la solution pour faire les choses de manière transparente (sans investir beaucoup de ressources) est de préconiser un développement d'applications mobiles multi-plateformes (hybrides).

Les applications mobiles multi-plateformes (hybrides) sont la combinaison entre les applications natives et les applications mobiles web. La synergie entre ces deux types d'application résulte en une réduction de temps, d'effort de développement, de prix et de maintenance. Ces applications sont téléchargeables via des boutiques en ligne, installées sur l'appareil et exécutées depuis une simple icône comme les applications natives. Les applications mobiles hybrides sont créées pour être exécutées sur plusieurs plateformes.

3. Développement mobile multi-plateformes ou hybride :

Cette approche permet aux développeurs d'écrire le code une fois et d'en appliquer quelques morceaux sur d'autres plateformes pour réduire le temps de développement. React Native, Xamarin, Ionic et Flutter sont des exemples d'outils les plus populaires.

Après beaucoup de recherches et de comparaisons nous avons choisi flutter pour développer notre application mobile, alors qu'est-ce que flutter ? et pourquoi le choisir ?

3.1 Choix de langage de programmation :

Le langage de programmation utilisée va beaucoup influer sur le projet et la manière dont celui-ci sera développé, en fonction des avantages et des inconvénients du langage. Il est important de bien étudier le langage, pour éviter de devoir changer de langage en cours de projet, ce qui constituerai une perte de temps considérable Le choix du langage s'est finalement porté sur Dart.

Dart est un langage de programmation optimisé pour les applications sur plusieurs plateformes. Il est développé par Google et est utilisé pour créer des applications mobiles, de bureau, de serveur et web.

Dart est un langage orienté objet à ramasse-miettes avec une syntaxe de type C++. Dart peut se compiler en code natif ou en JavaScript.

Figure 01 : Logo de Dart

3.2 Choix de Framework :

Pour que l'application soit robuste, facile à faire évoluer et réalisable en un temps minimum, un Framework représente un outil idéal.
Il existe une grande quantité de Framework. Chacun présentant des avantages et des inconvénients.
C'est pour cela nous avons choisi flutter pour ces grands avantages par rapport aux autres framework.

Flutter est un kit de développement logiciel (SDK) d'interface utilisateur open-source créé par Google. Il est utilisé pour développer des applications pour Android, iOS, Linux, Mac, Windows, Google Fuchsia et le web à partir d'une seule base de code.
Les applications Flutter sont écrites en Dart et utilisent de nombreuses fonctionnalités avancées du langage.

Figure 02 : Logo de Flutter

3.3 Les caractéristiques de Flutter :

- ✓ **Base de code unique pour Android et iOS**

Cette approche permet de simplifier et de réduire le temps et le coût de développement, et la maintenance est également une tâche facile.

- ✓ **Fonction de rechargement à chaud (hot reload)**

Cette fonction permet aux développeurs et aux concepteurs de voir les changements au fur et à mesure qu'ils codent.

- ✓ **Open-source et par Google**

Flutter est un choix populaire parmi les développeurs en raison de l'énorme soutien de la communauté.

- ✓ **Programmation Dart**

Flutter utilise un langage de programmation facile à apprendre et à mettre en œuvre, appelé Dart, qui est le langage de programmation général de Google.

Flutter a montré son potentiel quelques années seulement après son lancement officiel. Ce framework permet non seulement d'accélérer le processus de développement d'apps, mais aussi de gagner du temps et de l'argent.

- ✓ **Flutter : Très rentable**

Comparé aux autres frameworks de développement d'applications multi-plateformes, Flutter est plus rentable. Il répond à toutes les exigences de chaque entreprise, quels que soient son modèle et sa taille.

Grâce à sa fonction de réutilisation du code, Flutter permet aux développeurs de gagner du temps. Le concept « Write Once, Run Everywhere » (écrire une fois, exécuter partout) se vérifie puisqu'un seul code est utilisé pour développer une application pour plusieurs plateformes.

Les petites et moyennes entreprises peuvent choisir cette plateforme pour créer des applications rapides avec les fonctionnalités souhaitées et d'excellentes conceptions. Ici, le coût du développement d'une application reste faible car les applications Flutter prennent peu de temps à développer.

- ✓ **Flutter : Un développement plus rapide**

La popularité croissante de Flutter a de nombreuses raisons. La meilleure d'entre elles est d'offrir les résultats les plus rapides. Les développeurs peuvent déboguer et tester les codes rapidement. Le respect des délais, la réduction des coûts et les autres ressources et efforts nécessaires au développement d'une application sont nettement moins importants avec Flutter.

- ✓ **Flutter : Excellente expérience utilisateur**

L'interface utilisateur basée sur Flutter peut être installée virtuellement sur n'importe quelle plateforme. Elle possède son propre moteur de rendu qui permet aux développeurs de conserver l'interface utilisateur telle quelle tout en passant à une autre plateforme.

Par conséquent, les utilisateurs d'applications peuvent bénéficier d'une expérience excellente et identique à celle d'une application native sur diverses plateformes.

- ✓ **Flutter : Gamme de widgets**

Flutter dispose d'une large gamme de widgets. Ces widgets ont des capacités étendues qui permettent aux développeurs de construire facilement des interfaces complexes.

Les plugins et les widgets tiers contribuent ensemble à rendre le processus de développement transparent. Le kit d'outils d'interface utilisateur rend le processus de développement d'applications Flutter plus intuitif.

- ✓ **Flutter : Productivité améliorée**

Flutter dispose d'une fonction de rechargement à chaud qui permet aux développeurs et aux concepteurs de se coordonner efficacement et de vérifier les changements immédiatement sans modifier le code.

Pendant le processus de développement de l'application, toutes les modifications de l'interface utilisateur sont visibles pour les développeurs et les concepteurs. Cela leur permet d'économiser du temps et des efforts.

- ✓ **Flutter : Sécurité**

Google a conçu le framework Flutter en tenant compte de tous les problèmes de sécurité des applications modernes. On peut trouver des plugins fiables et bien testés dans Flutter pour atténuer les risques de sécurité tels que les failles d'authentification des utilisateurs, les injections de code malveillant et les fuites de données.

- ✓ **Flutter : Performance**

En tant que framework multi-plateforme, Flutter offre des performances inégalées par rapport à ses concurrents. Flutter compile les conceptions vers le code natif. Contrairement à React Native, Flutter rend les widgets directement à partir de la bibliothèque native plutôt que de télécharger les bibliothèques et les composants sur l'appareil avant de les rendre.

4. Conclusion

Nous avons présenté les différents types de systèmes d'exploitation pour Smartphones et appareils mobiles, les différents types d'application mobile ainsi que leurs avantages et inconvénients.

Pour répondre à notre problématique, nous avons choisi d'utiliser la solution multi-plateformes, qui nous offrent juste ce qu'il nous faut pour développer une application à la fois pas très lourde et assez riche en fonctionnalités native et multi-plateformes. Et nous avons justifié notre choix de développer notre application avec Flutter.

Chapitre 2 : Spécification des besoins

1. Introduction

La spécification des besoins représente la première phase du cycle de développement d'une application. Elle doit décrire sans ambiguïté l'application à développer.

Dans ce chapitre nous allons spécifier l'ensemble des besoins fonctionnels et non fonctionnels liés à notre application. Ensuite, nous allons modéliser les spécifications semi-formelles des besoins à l'aide des diagrammes de cas d'utilisation.

2. Spécification des besoins fonctionnels

Dans cette partie nous identifions les acteurs de notre organisme en ligne afin de pouvoir dégager les besoins fonctionnels.

2.1 Identification des acteurs

L'application doit fournir un ensemble de fonctionnalités aux clients. En effet, l'application Permettra aux clients d'effectuer leurs achats et de lancer des commandes.
Nous allons maintenant énumérer les acteurs susceptibles d'interagir avec le système.
Tout d'abord, nous commençons par définir ce qui est un acteur.

Définition d'un acteur : un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié.

Les acteurs de notre application sont :

- ***Le Client :*** toute personne qui fait l'inscription dans l'application et il peut, par la suite utiliser l'application, procéder à l'achat d'un article et réserver des produits.
- ***L'administrateur :*** c'est le superviseur qui contrôle, qui doit gérer le bon fonctionnement de l'application et rectifie l'application Java pour assurer le bon fonctionnement de système.

2.2 Analyse des besoins fonctionnels

Dans la suite, nous désignons par centres d'intérêts les différents services offerts par notre application :

3.3.1 Côté client :

- *Inscription :* Cette fonctionnalité obligatoire pour permettre au client d'avoir une visibilité et accès sur le contenu de l'application sur les ressources qu'elle contient. Un compte est attribué à chaque client inscrit, qui lui permet de consulter, réserver et commander les produits (livres électroniques).
  Le système doit permettre au client qui a déjà fait l'inscription de passer à l'étape de l'authentification.
- *Consulter le catalogue :* Le client dispose d'un accès au catalogue qui contient l'ensemble des produits vendus par le magasin. Ainsi qu'il peut consulter les détails d'un livre.
- *Ajouter un produit au panier :* Le système doit permettre au client de réserver des livres à partir des détails du produit dans l'interface « Catalogue ».
- *Gérer le panier :*
  - Voir les détails d'un achat.
  - Annuler la réservation d'un produit.
- *Gérer son compte :*
  - Accéder aux détails de son compte.
  - Actualiser ses informations.

3.3.2 Côté administrateur :

Après l'authentification l'administrateur sera redirigé vers son application administrateur qui lui permet de :

- *Gérer les utilisateurs de l'application admin :*
  - Ajouter un utilisateur.
  - Modifier un utilisateur.
  - Supprimer un utilisateur.
- *Liste des clients :*
  - Afficher la liste des clients et leurs réservations.
- *Gérer les catégories :*
  - Ajouter une catégorie.
  - Modifier une catégorie.
  - Supprimer une catégorie.
- *Gérer les livres :*
  - Ajouter un produit.
  - Modifier un produit.
  - Supprimer un produit.
- *Gérer les commandes des clients :*

- Valider les commandes d'un client.

3. Spécification des besoins non fonctionnels

Il s'agit des besoins qui caractérisent le système. Ce sont des besoins en matière de performance, de type de matériel ou le type de conception.

Les besoins non fonctionnels de notre application se résument à :

➢ Architecture : client/serveur

➢ SGBD de type : SQLite/Firebase

➢ Systèmes d'exploitation : Android, iOS, …

➢ Langages de programmation : Dart et Flutter

➢ Sécurité : C'est primordial pour l'application, étant donné qu'il gère des données confidentielles, l'authentification est indispensable.

➢ Performance : fluidité du logiciel et la minimisation du temps de réponse.

➢ Les contraintes du matériel : L'application sera installée sur un téléphone mobile à OS Android.

Certains besoins non fonctionnels sont généraux et ne peuvent pas être rattachés à un cas d'utilisation particulier.

4. Conclusion

La spécification des besoins nous a permis de définir les fonctionnalités de notre application, ce qui nous mène à entamer la phase de la conception (technique et graphique) pour assurer une bonne mise en œuvre d'un système fonctionnel répondant aux besoins cités.

Chapitre 3 : Analyse et Conception

1. Introduction

Après avoir défini la spécification du projet, la phase de conception vient pour mieux l'éclaircir. Dans une première partie, nous entamerons la conception technique en décrivant l'architecture générale de notre système ainsi que les vues statiques du système en utilisant les diagrammes UML appropriés. Ensuite, nous présentons la conception graphique de notre projet.

2. Méthodologie du projet

Afin de concevoir et développer notre application, nous avons opté pour le modèle de cycle de vie en V. Ce choix revient au fait que ce cycle est le plus efficace avec son principe de travail qui nécessite la vérification de chaque étape et la possibilité de corriger les fautes avant de se lancer vers l'étape suivante.

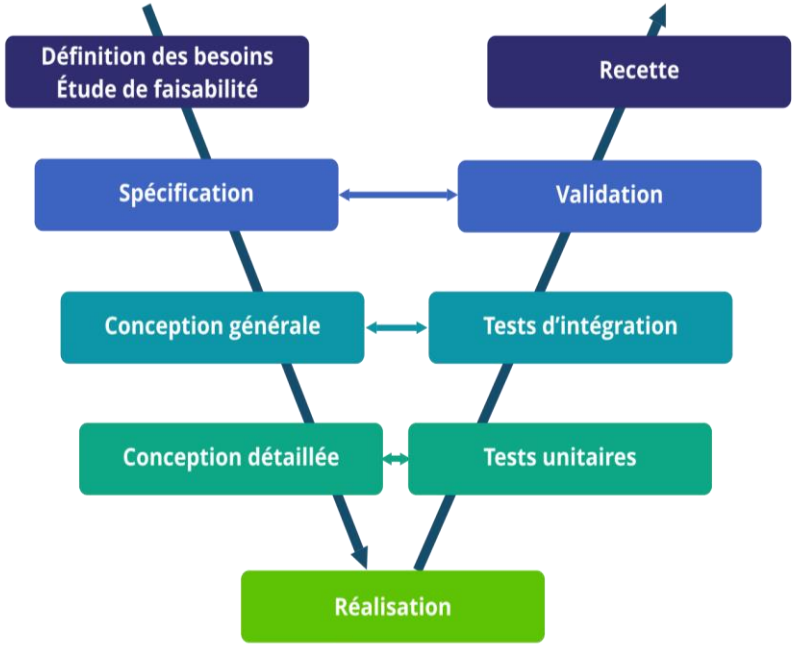

Figure 03 : Schéma de cycle de vie en V

3. Conception détaillée

Nous entamons la conception détaillée de l'application. Nous mettons en disposition une vue statique de notre application représentée par le diagramme de classes et les diagrammes de séquences.

3.1 Description de la vue statistique (diagramme de classes)

Le diagramme de classes est considéré comme le plus important de la modélisation orienté objet. Il s'agit d'une vue statique du fait qu'on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier. Le diagramme de classes retenu à la fin de la conception est le suivant :

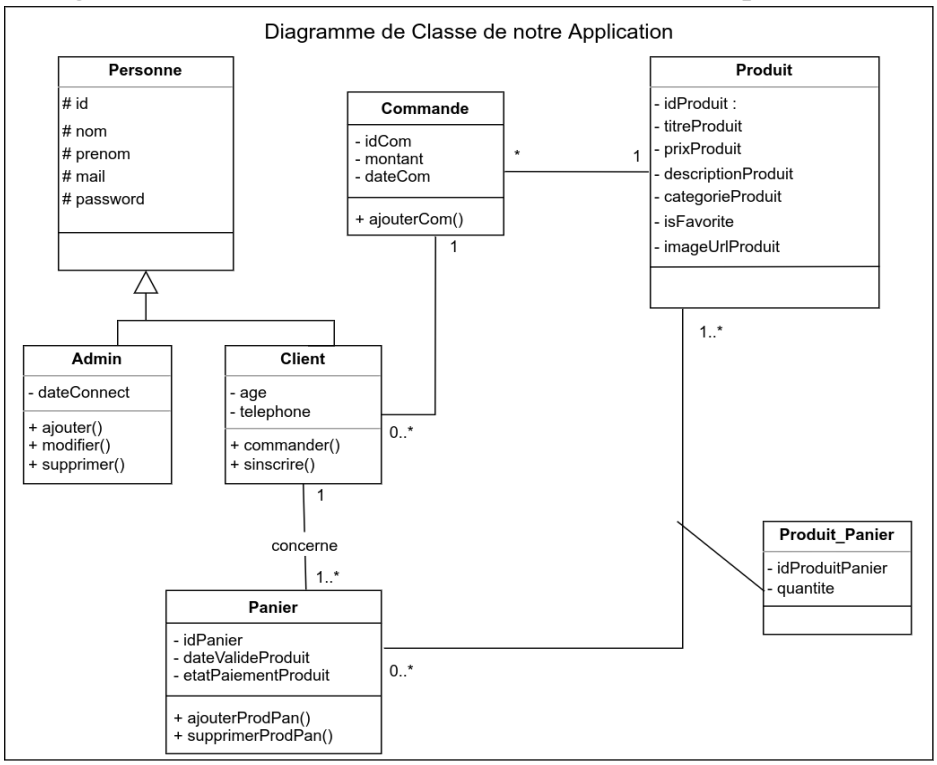

Figure 04 : Diagramme de classe de notre application

- **Description textuelle** :

* La classe Personne dérivent deux classe (Admin et Client) chaque personne à son nom, prenom, email, password.

* La classe Admin a le rôle de gérer les comptes admins aussi comptes clients, il peut ajouter modifier ou supprimer une livre ou un catalogue, chaque admin a une dateConnect comme un attribut.

* La classe Client peut consulter les catalogues et faire chercher les livre, peut faire inscription et gérer son compte. Un client peut posséder un panier. Chaque client a plusieurs attributs ...

* La classe Panier a les attributs suivants : idPanier et aussi dateValidLivre, chaque panier concerne un seul client, dans un panier on ajoute au moins un livre.

* Un livre peut ajouter au 0 ou plusieurs Panier, chaque livre caractérisé par son idLivre son titre et son prix aussi une description conclut et un catalogueLivre, aussi une imageUrl ....

3.2 Diagramme de cas d'utilisation global Administrateur-Client :

La figure suivante désigne le diagramme de cas d'utilisation global Administrateur-Client dans lequel nous allons mettre en évidence les services offerts par l'application.

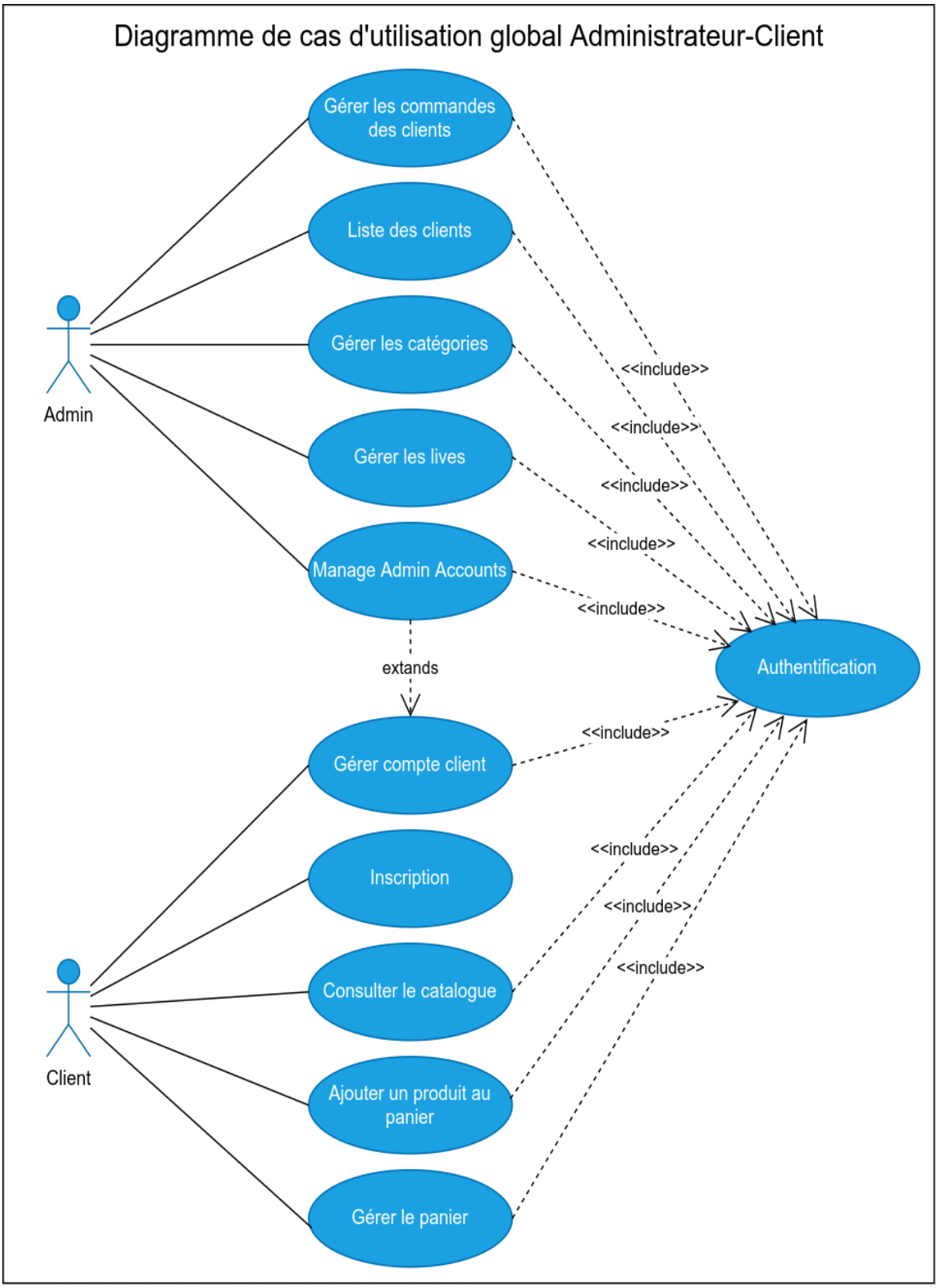

Figure 05 : Diagramme de cas d'utilisation global « Administrateur-Client »

3.3 Diagrammes de séquences

3.3.1 Diagramme de séquences d'inscription

La figure suivante désigne le diagramme de séquence d'inscription qui représente graphiquement les interactions d'inscription entre le client et l'application et SGBD.

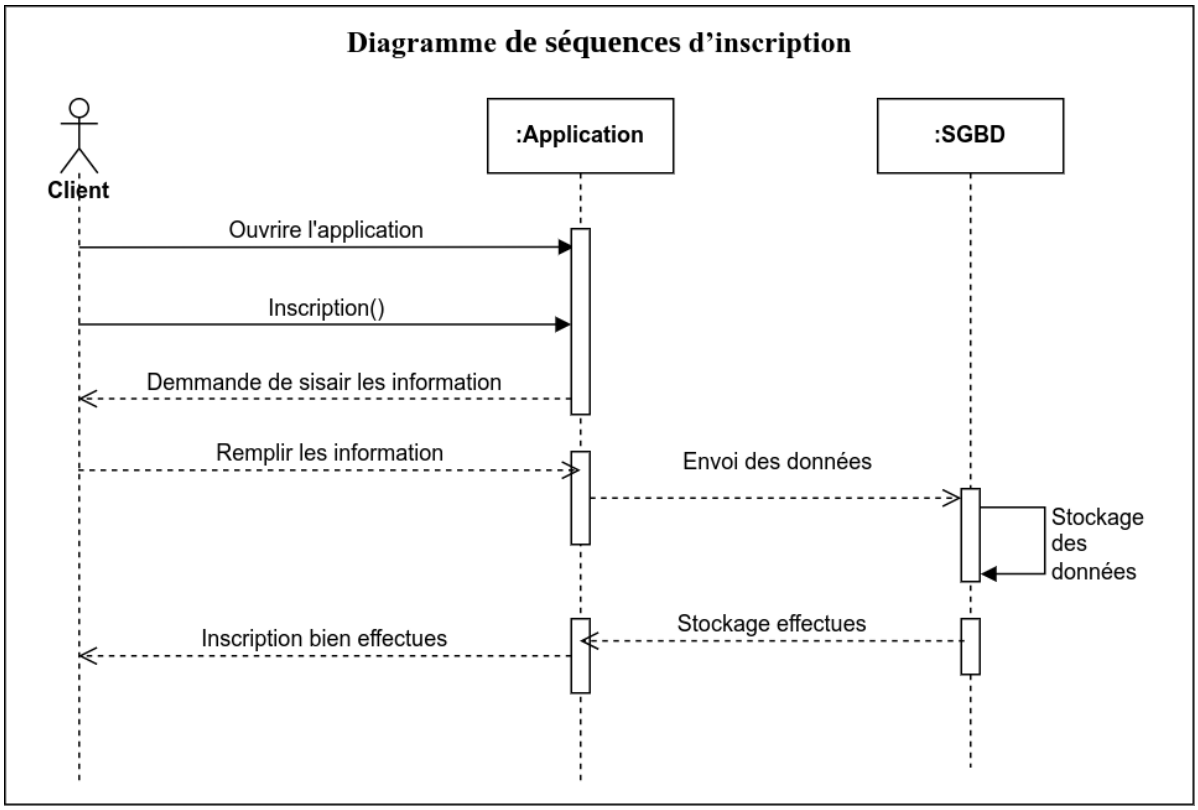

Figure 06 : Diagramme de séquences d'« Inscription »

L'utilisateur saisit les données d'inscription. L'application va vérifier la validation et lui répondre par la suite soit par passage à l'interface de consultation des livres ou soit par un message d'erreur.

3.3.2 Diagramme de séquences d'authentification

La figure suivante désigne le diagramme de séquence d'authentification qui représente graphiquement les interactions d'authentification entre le client et l'application et SGBD.

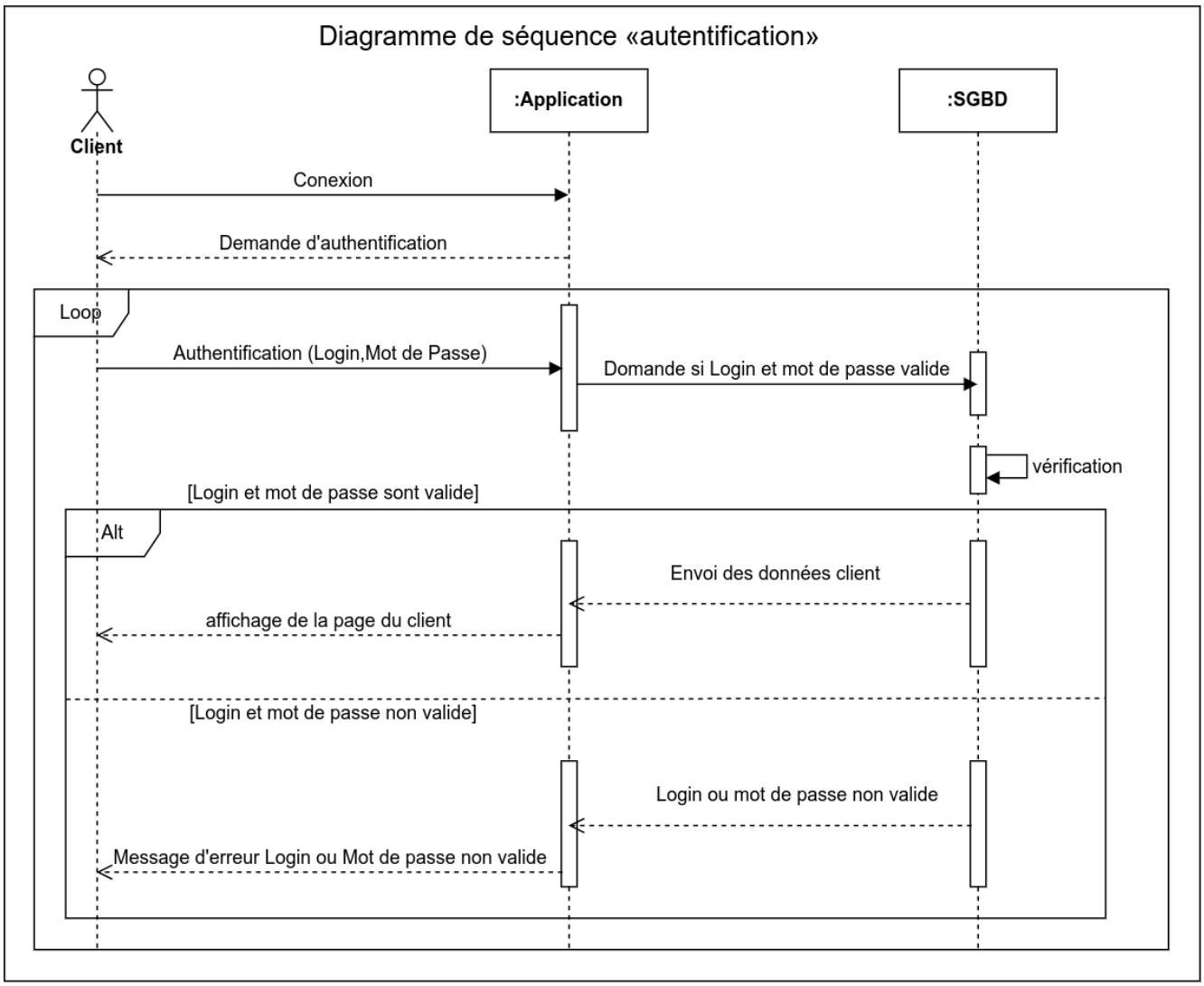

Figure 07 : Diagramme de séquences d'« Authentification »

L'utilisateur (Client) saisit les données d'authentification. Le système va vérifier ses droits d'accès dans la base de données et lui répondre par la suite soit par passage à l'interface de consultation des livres ou soit par un message d'erreur.

3.3.3 Diagramme de séquences « Consultation de catalogue des livres »

La figure suivante désigne le diagramme de séquence de consultation de catalogue des livres qui représente graphiquement les interactions entre le client et l'application et SGBD.

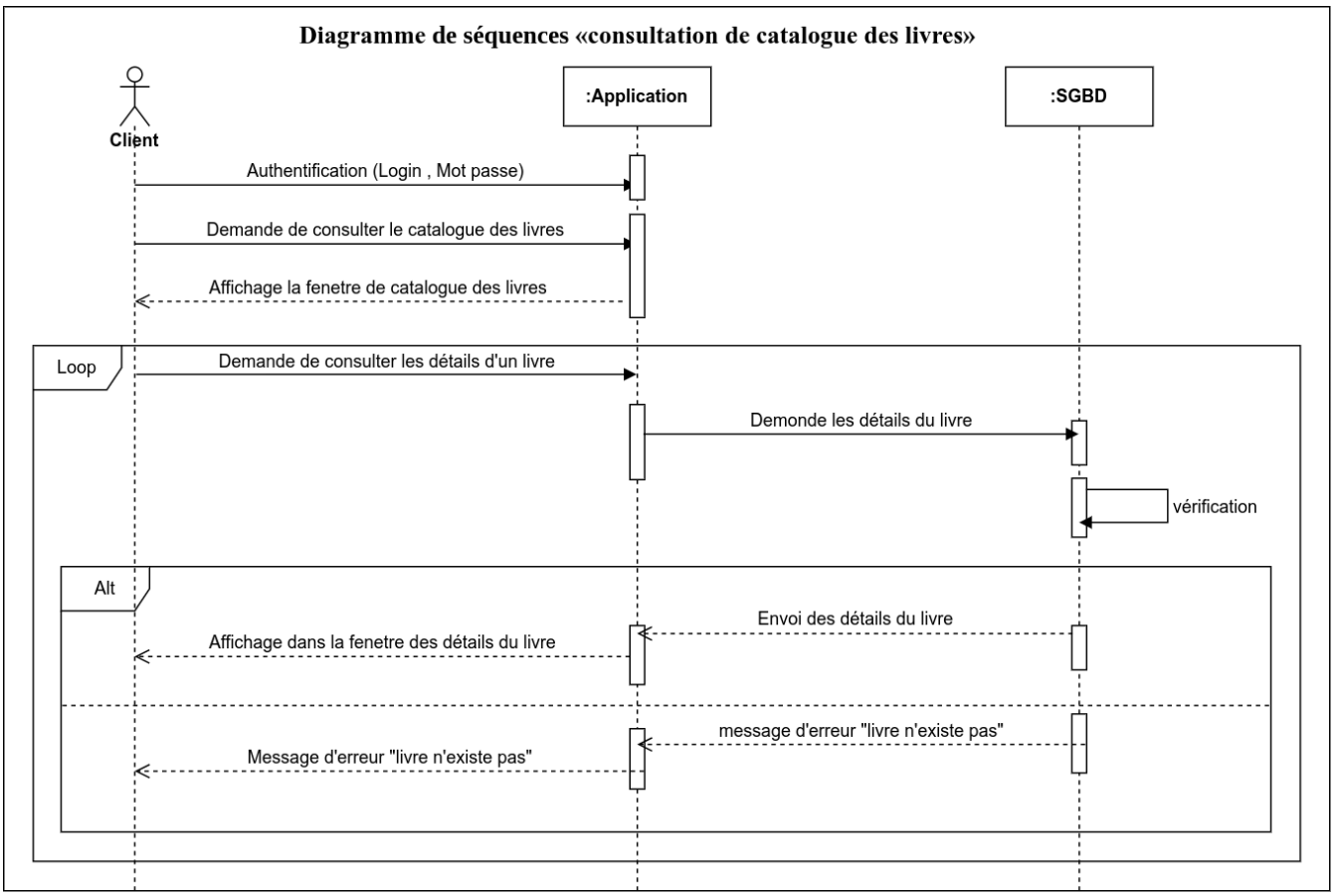

Figure 08 : Diagramme de séquences « Consultation de catalogue des livres »

Après que le client faire son authentification, il sera rédigé vers l'interface de catalogue, puis faire une demande de consultation a les détails d'un livre choisi pour naviguer vers les détails de ce livre, l'application sera vérifiée dans la base de données si le livre existe-il et lui répondre soit par afficher les détails de livre soit par un message d'erreur qui dire que le livre n'existe pas.

3.3.4 Diagramme de séquences « Ajout d'un livre au panier »

La figure suivante désigne le diagramme de séquence pour ajouter un livre au panier, il représente graphiquement les interactions entre le client et l'application et SGBD.

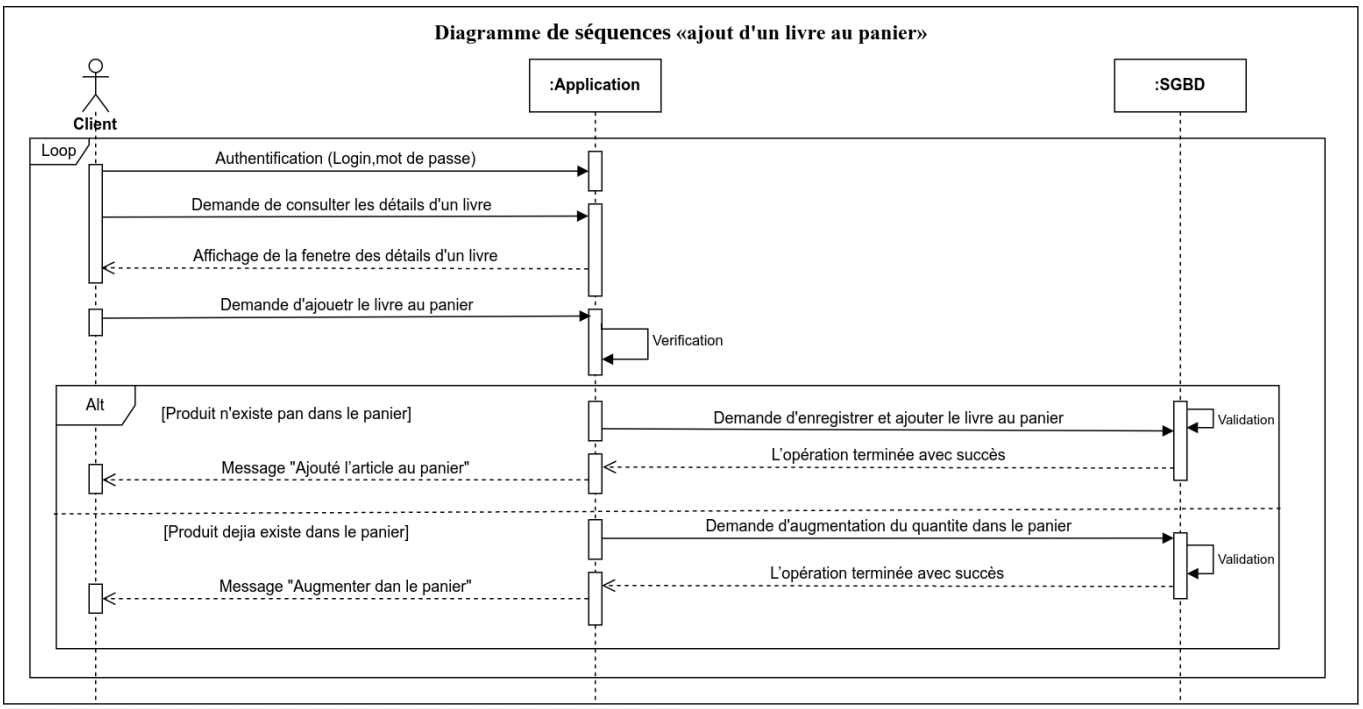

Figure 09 : Diagramme de séquences « Ajout d'un livre au panier »

Après l'authentification la page des catalogues sera affichée après avoir choisi un livre, le client demande d'ajouter un livre au panier, l'application fait la validation et ajouter dans la base de données et afficher le message qui dit que le livre est bien réservé. Pour faire une augmentation de quantité le client fait la demande d'ajouter encore une fois ou le nombre qui veut.

3.3.5 Diagramme de séquences « Gestion du panier »

La figure suivante désigne le diagramme de séquence de gestion du panier de client, il représente graphiquement les interactions entre le client et l'application et SGBD.

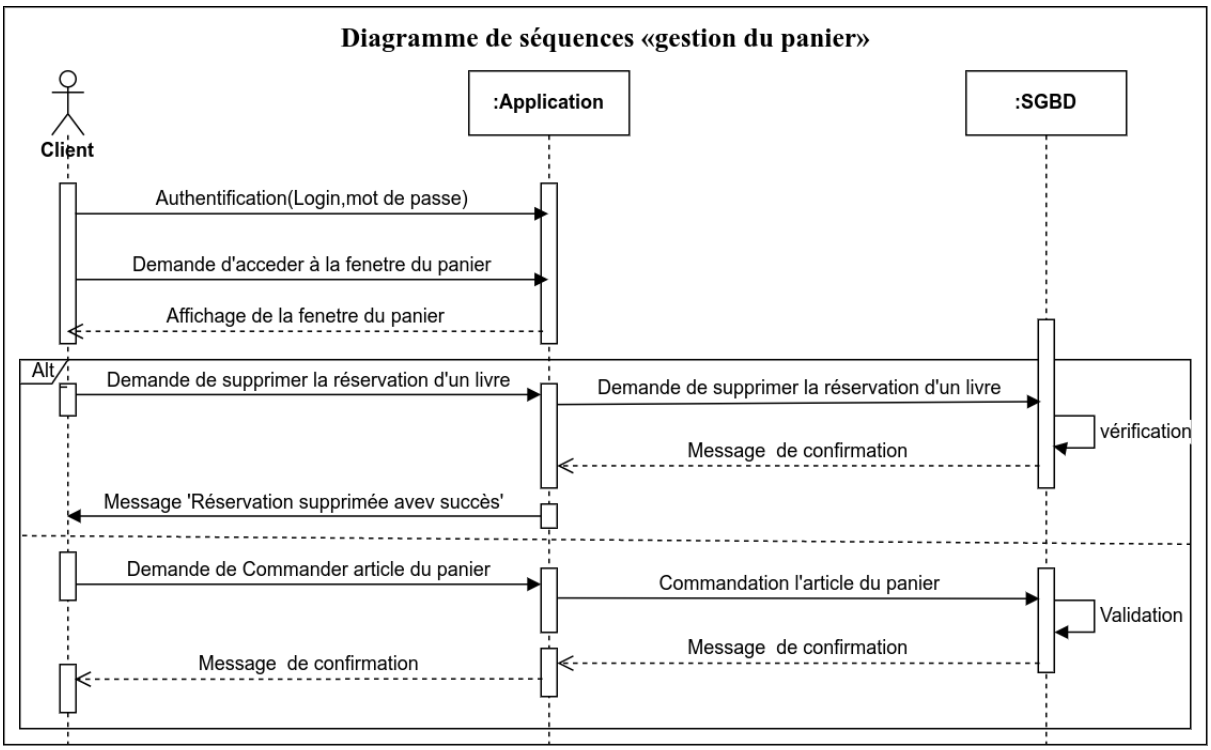

Figure 10 : Diagramme de séquences « Gestion du panier »

Après l'authentification, le client consulte au panier il a la possibilité de supprimer ou bien de confirmer une commande, pour supprimer un livre, le client demande la suppression de la réservation, l'application se transmet ce message à la base de données et après la confirmation de suppression dans la base de données, le message de suppression s'affiche sur l'application, et pour faire l'enregistrement d'une commande il se fait comme suit :

Tout d'abord le client demande l'enregistrement de la commande, l'application fait passer l'ordre à la base de données et directement le message de confirmation de commande s'affiche au client par l'application.

4. Conclusion

Ce chapitre présente les différents aspects de conception de l'application ainsi que les différents diagrammes intervenants dans cette conception tels que les diagrammes de cas d'utilisation, les diagrammes des séquences, le diagramme de classe.

La phase de conception était importante pour pouvoir visualiser le fonctionnement de notre application d'une façon abstraite, et à partir de laquelle nous avons pu passer à la phase de la réalisation.

Chapitre 4 : Réalisation

1. Introduction

Dans ce chapitre, nous expliquons l'implémentation des différents services utilisés ensuite nous présenterons les différentes interfaces de notre application.

2. Utilisation de firebase

Pour la conception de base de données et nous avons choisi Firebase car elle a des API intuitives regroupées dans un SDK unique. Ces API gagne le temps et réduit le nombre d'intégrations gérés par mon application.

Après avoir créé la base de données Firebase, il faut la relier à notre application Android pour cela, il faut installer les dépendances dont nous aurons besoin. Une configuration de firebase pour Android est nécessaire.

Les informations dans cette base sont stockées sous format JSON. Ce dernier contient la description, la latitude, la longitude et éventuellement la date d'expiration. On intègre les objets JSON dans Firebase et on ajoute des marqueurs avec la description enregistrée

3. Outils et environnement du développement de l'application

Dans cette partie nous allons présenter les différents outils utilisés pour la mise en place de notre solution

3.1 IDEs

- **VS Code**

Visual Studio Code est un éditeur de code open source extensible développé par Microsoft pour Windows, Linux et macOS.

Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégré. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

Figure 11 : Logo VS Code

- **Android Studio**

Android Studio est un environnement de développement pour développer des applications mobiles Android. Il est basé sur IntelliJ IDEA et utilise le moteur de production Gradle. Il peut être téléchargé sous les systèmes d'exploitation Windows, macOS, Chrome OS et Linux. Il a aussi plusieurs extensions pour Dart et Flutter.

Figure 12 : Logo Android Studio

3.2 Contrôle de version

- **Git**

Git est de loin le système de contrôle de version le plus largement utilisé aujourd'hui. Git est un projet open source avancé, qui est activement maintenu. À l'origine, il a été développé en 2005 par Linus Torvalds, le créateur bien connu du noyau du système d'exploitation Linux. De plus en plus de projets logiciels reposent sur Git pour le contrôle de version, y compris des projets commerciaux et en open source. Les développeurs qui travaillent avec Git sont bien représentés dans le pool de talents disponible, et la solution fonctionne bien sur une vaste gamme de systèmes d'exploitation et d'environnements de développement intégrés (IDE).

Figure 13 : Logo Git

- **Github**

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Ce site est développé en Ruby on Rails et Erlang. GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres.

Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches, et un wiki pour chaque projet. Le site est devenu le plus important dépôt de code au monde, utilisé comme dépôt public de projets libres ou dépôt privé d'entreprises. En 2018, GitHub est acquis par Microsoft.

Figure 14 : Logo Github

3.3 Autres outils

- **Draw.io**

draw.io ou **diagrams.net** est un logiciel de dessin graphique multi-plateforme gratuit et open source développée en HTML5 et JavaScript. Son interface peut être utilisée pour créer des diagrammes tels que des organigrammes, des structures filaires, des diagrammes UML, des organigrammes et des diagrammes de réseau

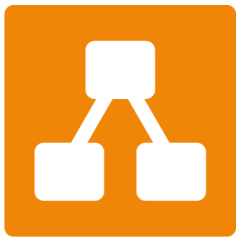

Figure 15 : Logo Draw.io

- **JSON**

JSON (JavaScript Object Notation) un format d'encodage de données structurées en JavaScript qui date de 2002. Il permet de représenter de l'information structurée. Il n'est pas rare d'associer le couple REST/JSON pour des raisons de simplicité. Son avantage est de fournir un support pour une écriture simple et légère au format texte, relativement compréhensible par les développeurs JavaScript, mais aussi - et surtout - d'être nativement interprété.

Figure 16 : Logo JSON

Figure 17 : Exemple de format JSON

- **Firebase**

Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application (Android, iOS, Javascript, Node.js, Java, Flutter, Dart, Unity, PHP,

C++ ...). Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de l'authentification sociale (Google , Facebook, Twitter et Github), et des notifications, ou encore des autres services.

Figure 18 : Logo Firebase

4. Présentation de l'application

4.1 Icône de l'application

C'est l'image qui va identifier notre application par rapport aux autres sur un appareil mobile.

Figure 19 : Icône de l'application

4.2 Les interfaces de l'application mobile

4.2.1 Interface d'authentification

Une fois l'application lancée, ce sera la première interface que l'utilisateur devra utiliser. C'est dans celle-ci que l'utilisateur doit saisir ses données pour pouvoir s'authentifier.

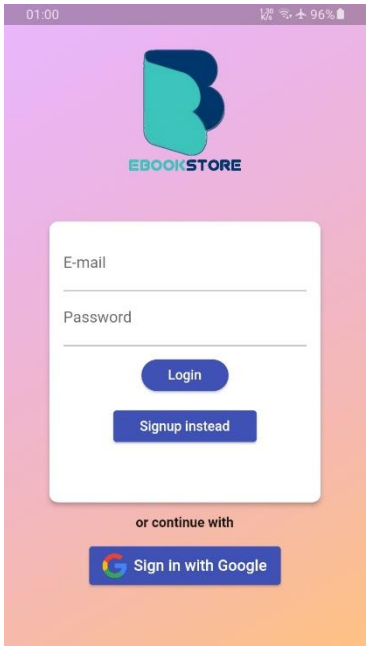

Figure 20 : Interface d'authentification

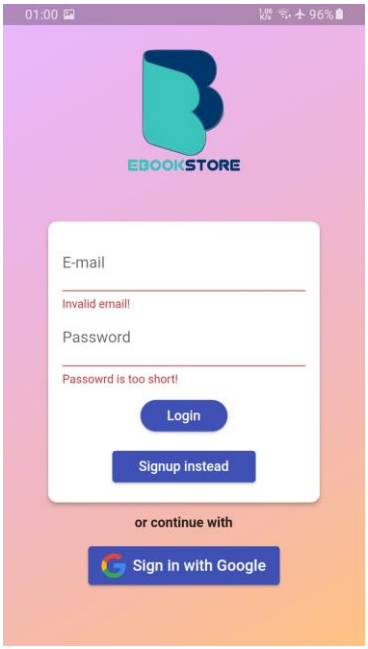

Figure 21 : Interface d'authentification échoué

- Dans le cas où les champs sont vides un message d'erreur sera affiché pour remplir les champs obligatoires.

4.2.2 Interface d'inscription

Une fois que l'utilisateur clique sur s'inscrire de l'interface d'authentification, il sera redirigé vers l'interface d'inscription. C'est dans celle-ci que l'utilisateur devra saisir ses informations.

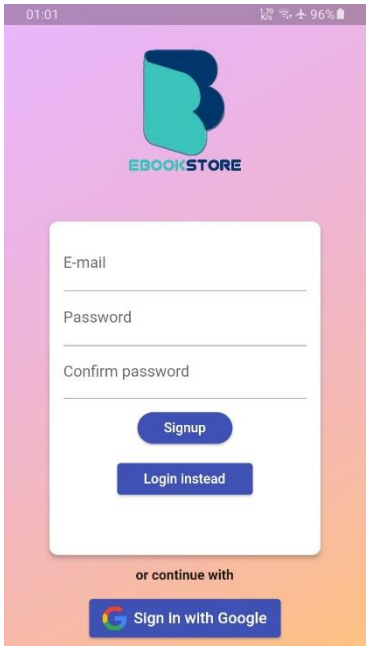

Figure 22 : Interface d'inscription

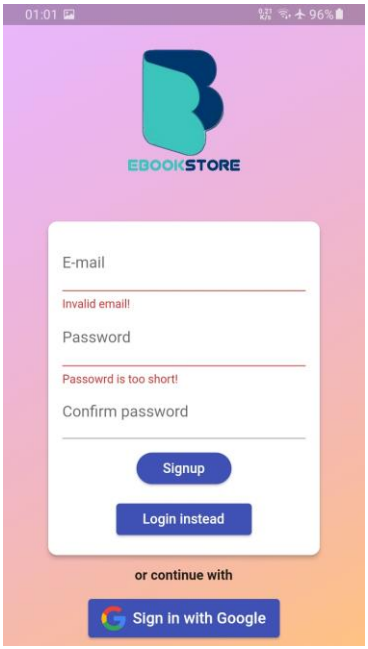

Figure 23 : Interface d'inscription échoué

- Dans le cas où les champs sont vides, un message d'erreur s'affiche pour remplir les informations obligatoires.

4.2.3 Les autres interfaces principales de l'application

Lorsque l'utilisateur est déjà inscrit et quand il s'authentifie avec succès cela lui permettra d'accéder à toutes les fonctionnalités de l'application :

Dans le premier écran après la connexion, nous voyons l'aperçu de la grille des produits où nous pouvons rechercher les livres que nous voulons, nous pouvons double appuyez sur chaque livre pour le marquer comme favori ou cliquez dans le cœur sous chaque article et aussi nous pouvons l'ajouter au panier en cliquant sur l'icône du panier [Figure 24].

En cliquant sur chaque livre, nous naviguons vers un autre écran où nous pouvons voir les détails du livre, un bouton pour ajouter le livre au panier et un autre pour le marquer comme favori [Figure 25].

Figure 24 : Interface principale

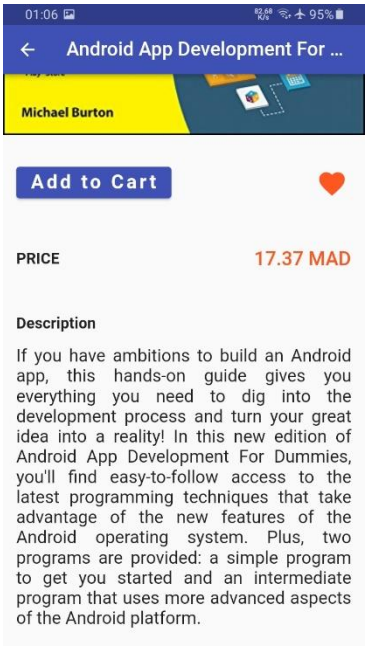

Figure 25 : Interface détailles d'un livre

Figure 26 : Interface des catégories

Figure 27 : Message d'ajout d'un livre au panier

En cliquant sur le deuxième onglet dans la barre du bas. L'application navigue vers l'écran des catégories [Figure 26].

En cliquant sur le bouton pour ajouter le livre au panier, nous voyons une petite fenêtre contextuelle en bas indiquant le succès de l'opération et une option pour annuler

l’opération [Figure 28]. En cliquant sur le troisième onglet dans la barre du bas, l’application naviguer vers le panier.

Nous pouvons glisser chaque élément de droite à gauche pour le supprimer du panier [Figure 29], et après cela, une fenêtre contextuelle s’affiche pour confirmer ou annuler [Figure 30].

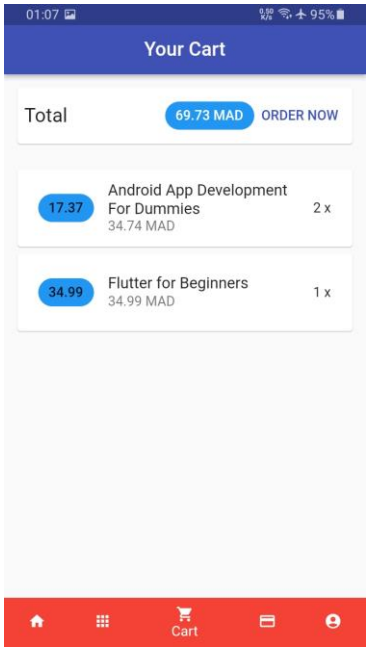

Figure 28 : Interface de panier

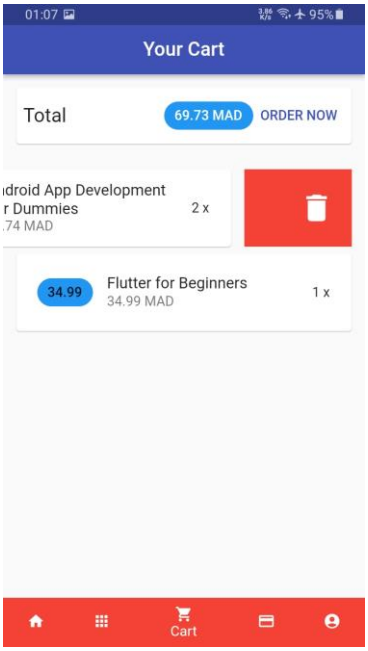

Figure 29 : Suppression d’un livre du panier

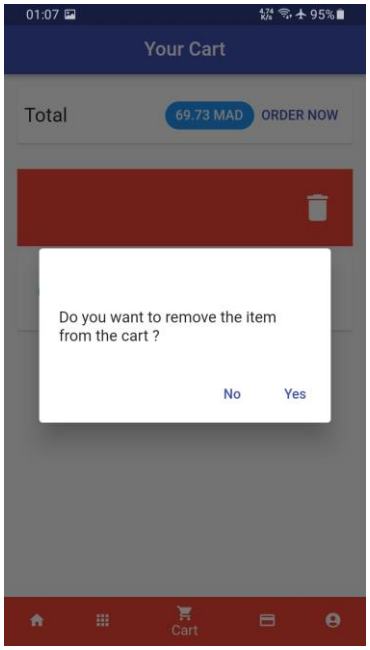

Figure 30 : Message suppression d’un livre du panier

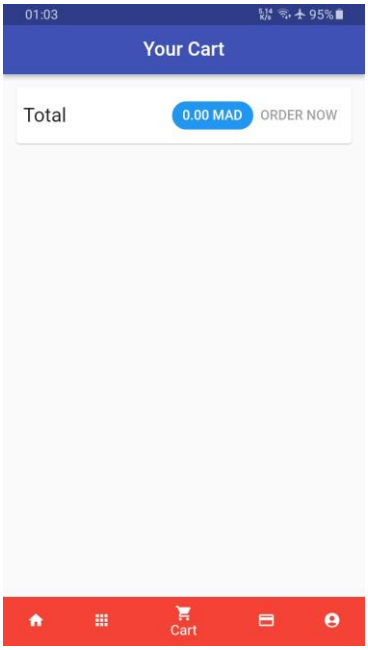

Figure 31 : Interface du panier vide

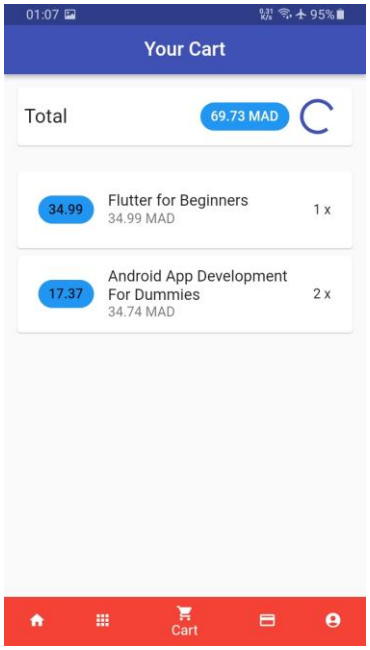

Figure 32 : Commande d'un livre

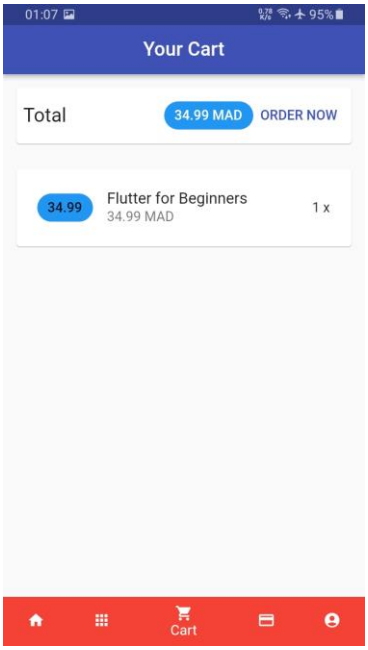

Figure 33 : Interface du paiement

Pour terminer la commande d'un article du panier, nous cliquons sur le bouton « Order Now ».

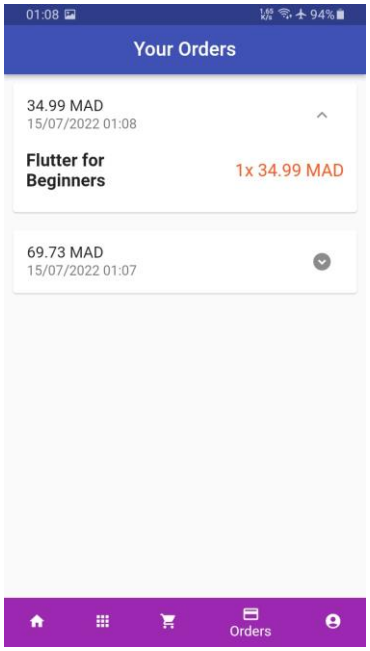

Figure 34 : Interface histoire des commandes

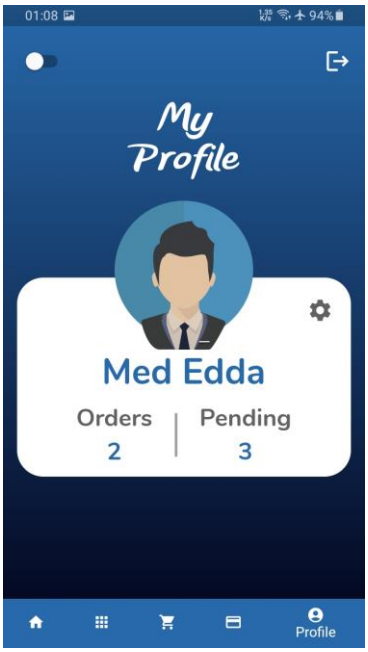

Figure 35 : Interface de profile

Le quatrième écran montre l'histoire de tous les commandes que nous avons faites. Figures [34].

Le dernier onglet affiche notre profil avec le nombre de commandes complètes et en attente (articles du panier) ainsi que le nom et prénom. Dans le coin supérieur droit, c'est le bouton de déconnexion et le bouton gauche permet d'activer ou de désactiver le mode sombre Figures [35].

Figure 36 : Interface catégories en mode sombre

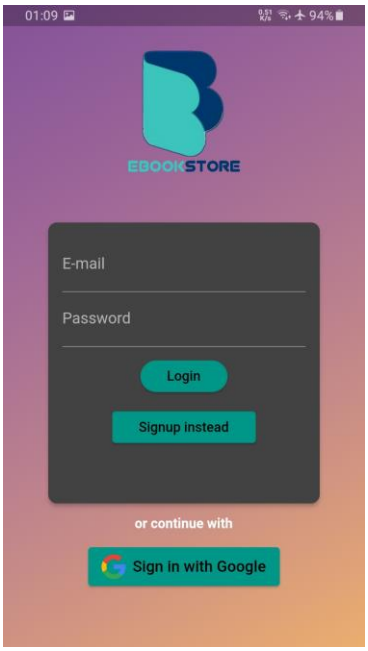

Figure 37 : Interface login en mode sombre

5. Conclusion

Ce chapitre a été consacré à la réalisation et présentation de la solution, nous avons commencé par la présentation des différents outils utilisés pour le développement de notre application, et nous avons terminé par la présentation des différentes interfaces de notre application.

Conclusion générale

A la fin de notre cursus en licence, nous avons été chargés de réaliser un projet de fin d'études. Notre travail est basé sur le développement d'un programme sur les technologies mobiles (smart phone) cross-plateforme. Ceci nous a amené à découvrir une nouvelle plateforme de développement et à enrichir notre savoir et notre expérience.

Ce projet s'inscrit dans le cadre d'une licence d'informatique en Sciences et Techniques option Génie Logiciel au sein de l'UMI, Faculté des Sciences et Techniques d'Errachidia.

Au cours de la phase de réalisation de l'application, on a élaboré une étude préalable sur les smart phones et son importance sur le plan social afin de préciser le but principal pour la future application. Cette phase a constitué le point de départ pour l'étape d'analyse et de spécification des besoins. Une fois nos objectifs fixés, nous avons enchaîné avec la conception afin de mener à bien notre projet. Nous avons procédé à la phase de réalisation au cours de laquelle nous avons appris le nouveau langage Dart et sa framework Flutter et appliqué toutes nos connaissances pour le réaliser.

Pour conclure, notre travail peut être sujet à des extensions et à des modifications. En effet, nous envisageons de présenter plusieurs fonctionnalités à savoir la possibilité pour les utilisateurs de contacter l'administrateur et aussi compléter la plateforme d'administration et une barre de recherche et intégrer un system de paiement comme Stripe, ...

Références

Bibliographie

[1] Modélisation Conceptuelle de Données Une Démarche Pragmatique de Patrick Bergougnoux
[2] UML et les Design patterns de Craig Larman
[3] Flutter - Développez vos applications mobiles multiplateformes avec Dart de Julien Trillard

Webographie

[1] https://docs.flutter.dev/
[2] https://firebase.com/
[3] https://github.com/
[4] https://docs.dart.dev/
[5] https://firebase.flutter.dev/
[6] https://app.diagrams.net/
[7] https://material.io/
[8] https://flutterawesome.com/
[9] https://pub.dev/
[10] https://udemy.com/
[11] https://www.maximelb.com/quel-framework-pour-developper-une-application-mobile-en-2020/
[12] https://www.ambient-it.net/react-native-vs-xamarin-vs-ionic-vs-flutter/
[13] https://citronnoir.com/creer-une-application-mobile/flutter-loutil-de-developpement-dapplication-moderne-cree-par-google/