République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abderrahmane MIRA de Bejaia
Faculté des Sciences Exactes

Mémoire de fin de cycle
En vue d'obtenir le diplôme de master professionnel en génie logiciel

Conception et réalisation d'une application Android de gestion commerciale Cas d'étude : SAFE SOFT d'Alger

Réalisé par :
Mr Arezki TIGMIT.
Mr Mohand Saïd TAYEB CHERIF.

Devant le jury composé de :
Examinatrice: Mme Hayette KHALED
Examinatrice : Mme Dalila KESSIRA
Promotrice : Mme Lina BACHIRI.

Promotion : 2019/2020

Remerciements

J’adresse mes vifs remerciements :

En premier lieu, au dieu tout-puissant de nous avoir donné la santé, la volonté, le courage et de nous avoir fourni la force pour achever ce modeste travail…

Nous ne saurons oublier nos parents, pour leur contribution, leur soutien et leur patience. Nos proches, nos amis et toutes les personnes qui nous ont aidés de près ou de loin tout au long de notre parcours éducatif.

À notre promotrice M^{me} Lina BACHIRI pour son encadrement, son soutien sans failles et sa disponibilité. Ces conseils, ses suggestions de lecture, ses commentaires et ses corrections ont été très précieux pour mener à bien ce modeste travail.

Aux membres du jury M^{me} Hayette KHALED et M^{me} Dalila KESSIRA pour l’intérêt qu’ils ont porté à notre recherche en acceptant d’examiner notre travail et de l’enrichir par leurs propositions.

À M^{me} Hayette KHALED, M^{me} Dalila KESSIRA pour avoir fait l’immense honneur d’accepter de juger mon travail.

À l’entreprise « Safe Soft » pour nous avoir ouvert ses portes, accueilli et offert la possibilité de faire notre stage pratique et de développer notre application sous son aile.

À Mr Wassim CHAGUETMI, tout particulièrement, pour avoir été un si bon encadrant au sein de l’entreprise Safe Soft, et pour nous avoir transmis un peu de son expérience et de son savoir-faire.

Dédicaces A. TIGMIT

Je dédie ce modeste travail :

A mon très cher père qui, sans lui, je ne serais pas la personne que je suis et qui a veillé a ce que je ne manque de rie tout au long de ma vie, pour avoir été le meilleur père que je puisse avoir.

A ma chère mère qui, par sa tendresse, sa bienveillance et son amour, pour m'avoir soutenu, aidé, aimé et si bien élevé pour être qui je suis.

A mon cher petit frère Anis pour avoir été un si bon frère pour moi.

A mes cousins et cousines pour leur présence et leur soutien.

A mes oncles et tantes pour avoir été là pour moi quand j'avais besoin d'eux et plus particulièrement Naima et sa famille pour tout ce qu'ils ont fait pour moi que je ne saurais compter.

A toute ma famille qu'elle soit proche ou lointaine.

A toi ma compagne adorée, Aziza.K pour tout le bonheur que tu as apporté à ma vie.

Et a tous mes amis pour avoir été les meilleurs amis avec lesquels j'ai passé de si bons moments de folie.

Dédicaces M. S. TAYEB CHERIF

Je dédie ce modeste travail :

À ma très chère maman, qui a su me donner l'attention, l'affection, l'aide et l'amour qui m'ont permis d'achever ce projet dans de bonnes conditions.

À mon très cher père qui a toujours répondu présent dans les moments les plus difficiles. Son soutien et son encouragement m'ont toujours donné la force de poursuivre mes études. Aucune dédicace ne pourra compenser les sacrifices de mes parents.

À mes frères Yanis, Massi et ma sœur Kenza.

À mes oncles, mes tantes, mes cousins, mes cousines et toute ma famille.

À mon amie Sidali DILMI qui m'a apporté une aide précieuse.

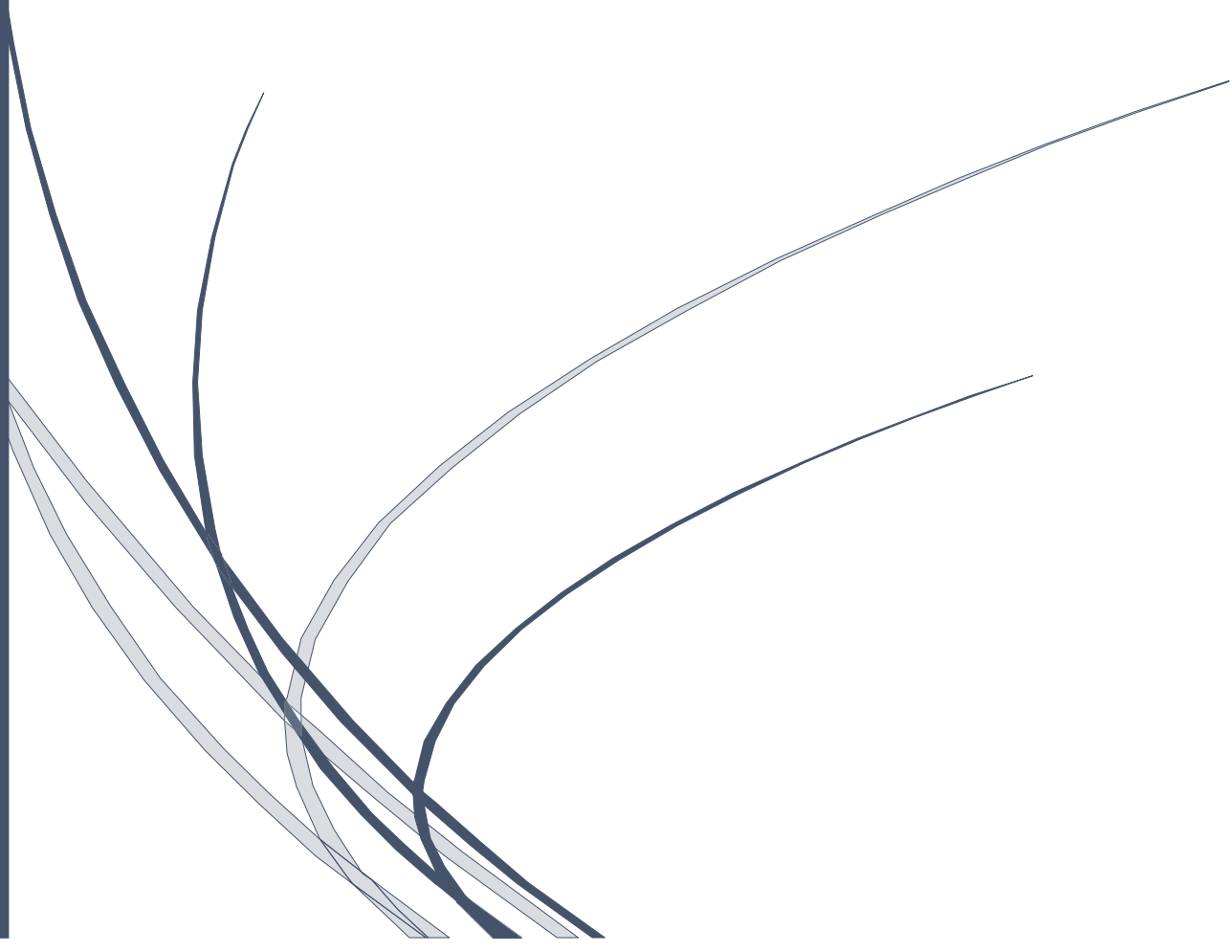
À tous ceux qui sont chers et proches de mon cœur.

Table des matières

Table des figures

Table des tableaux

Introduction Générale



Introduction générale

Aujourd'hui nous somme dans l'aire de la téléphonie mobile. Chaque personne possédé son smartphone qui gère probablement toute s vie ; de son planning quotidien aux achats qu'il fait.

Conçu pour accompagner les personnes n'importe ou n'importe quand, le smartphone est le parfait outil pour les entreprises pour la commercialisation de leurs produits, promouvoir leur marque, ou encore rapprocher leurs produits au maximum de leurs clients. De plus, grâce à la connexion permanente de ce téléphone intelligent au réseau internet via connexion WI-FI ou 4G permet au client d'être au premier rang pour profiter au maximum des produits en faisant ses courses pour le client et de gérer plus facilement ces ventes pour le gérant.

En gros, l'industrie de la téléphonie mobile ou du smartphone permet de :

-Rapprocher le client de l'entreprise et du produit sans trop d'efforts et sans déplacement.

-Faciliter les transactions d'achat et de ventes entre le client et l'industrie.

-Suivre les tendances et les nouveautés pour améliorer le markéting et la mise en vente de leurs produits des produits.

Dans ce contexte précis, entre en jeu notre projet d'étude qui consiste à « Concevoir et de réaliser une application Android de gestion commerciale » en mettant en œuvre les connaissances acquises durant nos deux cycles au sein de l'université A/Mira de Béjaïa.

Ce mémoire est composé de quatre chapitres :

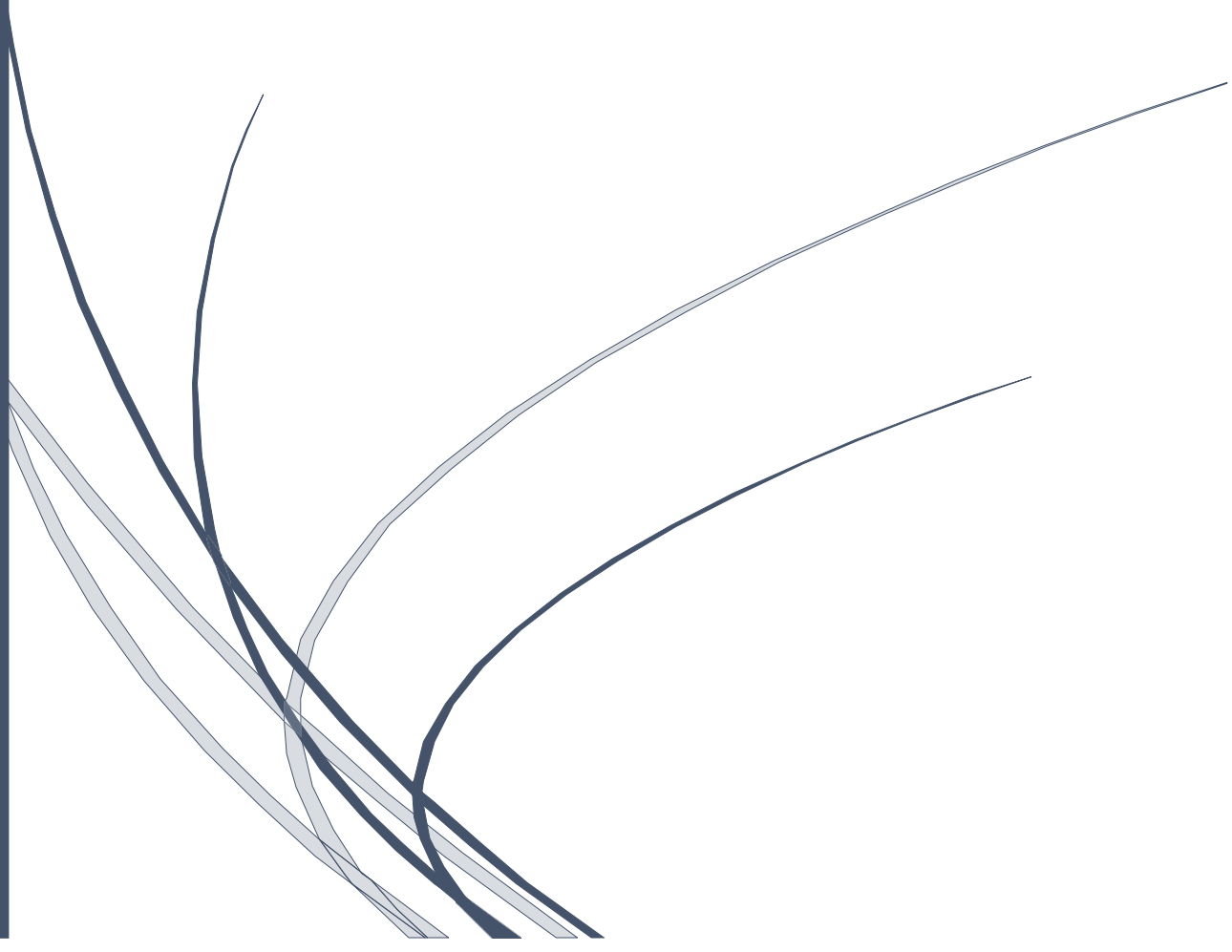
Le premier chapitre est une présentation globale des applications mobiles, de l'entreprise « Safe Soft » où s'est déroulé notre stage, et de la méthode de conception pour laquelle nous avons opté.

Le deuxième chapitre consiste à présenter les besoins fonctionnels et non fonctionnels de notre projet et le suivi du processus de son développement.

Le troisième chapitre, lui, est réservé à la partie conception de l'application. Nous présentons dans ce dernier l'architecture de notre application et les différents diagrammes d'interaction system et de conception.

Dans le quatrième chapitre, nous mettons en avant la partie implémentation, et la présentation de quelques interfaces graphiques.

Chapitre I Applications mobiles et méthodologie de conception



Chapitre I Applications mobiles et méthodologie de conception

Introduction

De nos jours, nous pouvons dire que les appareils mobiles sont nos compagnons de tous les jours. Ils prennent une place considérable dans nos vies quotidiennes grâce à leur utilité dans presque tous les domaines de la personne.

Nous présentons dans ce chapitre une vue générale de l'informatique mobile. Ensuite nous nous pencherons sur le système d'exploitation Android, ces fonctionnalités, son architecture et son cycle de vie. Enfin, nous finirons ce chapitre avec quelques notions et informations sur le Processus Unifié (UP).

Partie 1 Applications Mobiles

I.1. Informatique mobile

I.1.1. Définition

La mobilité est une caractéristique de quelque chose qui peut se déplacer ou être déplacé, changé de position et de situation.

La première façon à considérer l'informatique mobile c'est de fournir aux utilisateurs appareil portable. L'utilisation de ces appareils reste immersive, c'est-à-dire que l'utilisation de ces appareils requiert l'attention de l'utilisateur et applique tout mouvement indépendamment de celui-ci. Cette approche est souvent appelée nomade, bien que ce terme puisse prendre d'autres significations dans d'autres domaines, tels que dans les systèmes de distribution où le nomadisme est la capacité d'un système à fournir un environnement de travail de chaque utilisateur à chaque poste. **[1]**

I.1.2. Appareils mobiles

Un appareil mobile (traduction littérale du terme anglophone « mobile devise » est un appareil informatique portatif utilisable de manière autonome lors d'un déplacement. Les appareils mobiles sont de petite taille, certains peuvent être mis dans les poches. Ils sont typiquement dérivés des téléphones mobiles, et permettent d'accéder au Web, de lire du courrier électronique, de prendre des photos, de jouer à des jeux vidéo, d'écouter de la musique, de regarder des clips vidéo ou bien de télécharger des applications. Ils peuvent également comporter un calendrier ou un carnet d'adresses. **[2]**

C'est l'usage et l'ergonomie de ces appareils qui définit le groupe d'appareils où l'on peut les placer. Et on peut distinguer plusieurs genres d'appareils mobiles comme par exemple :

a. Smartphone

Un smartphone est un ordinateur de poche qui combine les fonctions d'un téléphone mobile avec celles d'un ordinateur (accéder au web, envoyer des e-mails, et écouter de la

musique). En d'autres termes, c'est 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. C'est le cas de l'iPhone d'Apple par exemple. **[2][3]**

Figure 0.1. Téléphone portable.

b. Tablette tactile

Une tablette tactile est un ordinateur portable très léger et sans clavier. Le texte est entré manuscrit à l'aide d'un stylo, en touchant l'écran, par la parole. La puissance de calcul d'une tablette est inférieure à celle d'un ordinateur de bureau. Sur les tablettes convertibles, l'écran peut être retourné, et l'appareil peut ainsi être utilisé comme un notebook. **[2]**

Figure 0.2. Tablette.

Ces appareils sont à mi-chemin entre le téléphone et l'ordinateur personnel. La majorité de ces appareils sont équipés d'un système d'exploitation pour smartphone. Elles sont utilisées pour des activités proches de celles d'un ordinateur personnel. **[2]**

c. Ultraportable

Un ultraportable ou ultra-portable désigne un ordinateur portable de taille très réduite et de masse minimum dont la principale qualité est de pouvoir être transporté et utilisé n'importe où avec un encombrement minimum tout en conservant de bonnes performances. Leur taille est comprise entre celle des smartphones (environ format A6) et des ordinateurs portables (environ format A4). **[2]**

Figure 0.3. Ultraportable.

I.2. Applications mobiles

I.2.1. Définition

L'application mobile se définit comme étant logiciel téléchargeable et qu'on installe facilement sur son smartphone comme nous le faisons avec tout logiciel sur notre PC portable.

C'est un logiciel applicatif téléchargeable (gratuitement, payante ou mix avec des contenus ou fonctionnalités payantes) sur un appareil mobile (baladeur, smartphone, tablette tactile) via une plateforme de téléchargement adapté au système d'exploitation (Google Play, App Store, Windows Store). **[4] [5]**

Pour télécharger une application sur un téléphone mobile, il existe différentes possibilités :

- Transfert depuis un ordinateur via un câble de connexion.
- À partir d'un service mobile.
- Via une boutique logicielle accessible depuis un téléphone mobile (App Store d'Apple, Windows Market Place, Nokia OVI, ANDROID Market, etc.).
- Le cas échéant. L'application est dite native ; elle est déjà dans le téléphone lors de l'achat du téléphone (l'opérateur ou le fabricant l'a ajouté comme fonction de base).

I.2.2. Histoire des applications mobiles

Pour commencer, les applications mobiles sont utilisées par les gens à des fins sociales. Les réseaux sociaux sont disponibles pour les appareils mobiles pour permettre aux utilisateurs de saluer leurs amis et de publier des cris, même sans l'aide de leurs unités informatiques. Les gens peuvent partager des idées et s'abonner à de nouvelles informations.

L'industrie des applications mobiles a commencé lorsque l'iPhone d'Apple et son iTunes Store ont été introduits en 2007. C'était aussi le moment où les utilisateurs ont commencé à utiliser leurs appareils mobiles pour surfer sur Internet. Seuls les professionnels utilisent leurs appareils mobiles pour consulter les e-mails et les mises à jour. Mais avec l'iPhone, la façon dont les consommateurs ordinaires utilisaient leur appareil mobile pour plus que des appels téléphoniques vocaux était un bon début.

À mesure que les consommateurs ordinaires utilisent leurs téléphones mobiles, le besoin de fournir des applications mobiles adaptées augmente également. De nombreuses entreprises développent leurs propres versions de Smartphones et de plates-formes pour offrir plus de mobilité dans l'accès au World Wide Web.

Pour cette raison, les développeurs mobiles ont réalisé qu'il était nécessaire de catapulter les capacités WWW mobiles vers des progiciels ou des applications mobiles plus adaptés. Ceux-ci fonctionneront comme des sites Web mais avec une utilisation plus conviviale et conçue pour les écrans plus petits. **[6]**

Alors que la demande d'applications mobiles augmente, de nombreuses entreprises ainsi que Skyline Apps Technologies réclament de répondre à ce besoin. Des petites industries aux grandes entreprises, les applications mobiles ont été utilisées pour effectuer une tâche spécifique. Les investissements ont été acheminés vers cette entreprise car à partir

d'aujourd'hui, de nombreuses personnes passent leur temps à utiliser leurs appareils mobiles et, par conséquent, leur ordinateur personnel et même les ordinateurs portables ont été utilisés moins longtemps.

Skyline Apps a développé Mobile App Maker pour permettre aux utilisateurs de créer leurs propres applications en fonction de leur style et de leurs préférences. Ce créateur d'applications mobiles est un logiciel de clic et de glisser qui est facile et simple. Un simulateur est également utilisé pour afficher l'affichage en temps réel sur le smartphone réel.

Les applications mobiles sont également utilisées dans le domaine de l'enseignement, dans le domaine médical et dans d'autres domaines. Même les enfants ont été initiés aux applications mobiles à des fins de divertissement et à des fins éducatives. Les amateurs de musique créent leur propre liste de lecture, vidéo ou musique et d'autres contenus qu'ils souhaitent partager avec les utilisateurs de leur application mobile.

Ainsi, les applications mobiles sont nées, mais beaucoup doivent encore venir et être utilisées dans tous les domaines de l'activité humaine. **[6]**

I.2.3. Domaines d'application mobiles

De la même façon qu'il existe plusieurs types de sites web (site vitrine, blog, site e-commerce, etc.), il existe également différents domaines d'applications mobiles aux usages et objectifs différents. **[6]**

Et parmi ces différents domaines on peut citer :

a. Applications mobiles synonymes d'utilitaires

Les applications dites « utilitaires » sont développées pour résoudre rapidement des problèmes spécifiques. Leur objectif : faire économiser du temps aux mobinautes et réduire leurs efforts. Parmi ces applications on retrouve la célèbre calculatrice, l'alarme, la lampe de poche, le scanner, etc.

b. Applications mobiles pour divertir

Jeux, musiques, films et vidéos… ces genres d'applications mobiles n'ont qu'un but : divertir. Les applications mobiles divertissantes peuvent aussi avoir un rôle éducatif. Une agence digitale ou de graphisme peut par exemple développer un jeu basé sur la connaissance des couleurs. Concernant les applications liées à la musique il n'y a pas que celles basées sur l'écoute. Certaines peuvent aider à composer et créer des morceaux, d'autres à enregistrer du son ou encore à regarder des concerts en live.

c. Applications mobiles pour un accompagnement au quotidien

Nous sommes de plus en plus connectés : nous voulons savoir combien de kilomètres nous avons parcouru lors de notre séance de sport, connaître les calories brûlées, le nombre de marches montées dans la journée… Le suivi de notre condition physique et de nos symptômes est permanent.

d. Applications mobiles pour suivre l'actualité

Toutes les chaînes d'information disposent de leur propre application mobile pour permettre aux mobinautes de suivre l'actualité en direct. Et plusieurs entreprises proposent

des applications pour suivre la météo, le sport l'actualité, la politique… et tout ça en temps en heure.

e. Applications mobiles en support du site e-commerce

Si vous avez un site e-commerce vous devriez sérieusement envisager de développer une application mobile. De plus en plus d'utilisateurs effectuent leurs achats en ligne, et pas seulement depuis un ordinateur : près de 80% des mobinautes achètent depuis leur smartphone.

Ce genre d'application mobile doit prendre en charge tout le processus d'achat : de la présentation du produit au paiement en passant par les informations liées au compte (commandes en cours, coupons de réduction, prochaine livraison, etc.). **[7]**

I.2.4. Types d'application mobiles

Pour chaque développeur, il existe différents types d'applications mobiles. Chaque type a ses caractéristiques et son utilité pour l'utilisateur. Nous distinguons alors plusieurs solutions pour satisfaire les atteintes du client : applications natives, applications web, applications hybrides et applications Flash.

a. Applications mobiles natives

Une application native est une application mobile qui est développée spécifiquement pour un des systèmes d'exploitation utilisé par les smartphones et tablettes (iOS, Android, etc.). Les applications natives pour iPhones sont par exemple développées avec le langage de développement Objective-C.

Le fait de développer une application native permet généralement d'utiliser toutes les fonctionnalités liées au système d'exploitation visé (GPS, accéléromètre, appareil photo, etc.) et permet également de proposer des applications généralement plus riches que les web applications en HTML5. Une fois téléchargées et installées certaines applications peuvent par ailleurs être utilisées sans connexion Internet. **[8]**

b. Applications mobiles web

Une application web mobile désigne un site web adapté pour les mobiles. L'application n'a pas besoin d'être téléchargée et installée puisqu'il s'agit d'un simple site web accessible via un navigateur mobile. **[9]**

c. Applications hybrides

Une application hybride est une application utilisant le navigateur web intégré du support (Smartphone ou tablette) et les technologies Web (HTML, CSS et JavaScript) pour fonctionner sur différents OS (iOS, Android, Windows Phone, etc.). Une telle application utilise les fonctionnalités natives des smartphones. Elle peut être distribuée sur les plateformes d'applications telles que l'App Store, le Google Play, etc.

Les performances d'une application hybride sont souvent inférieures à celles d'une application native, développées spécifiquement pour IOS et Android. Cette technologie ne convient que dans certains cas, si les besoins de développement sont simples.

d. Applications Flash

Comme pour les applications HTML5/JavaScript, une application Flash est une solution de développements multiplateformes, qui permet d'accéder à la plupart des

ressources des smartphones et tablettes : multitouche, accéléromètre, GPS, bases de données locales SQLite. Ces applications embarquent le runtime AIR dans leur code.

Nous pouvant citer Flash Builder et Flex en version 4.5 comme applications flash permettent de packager des applications ANDROID, iOS et BlackBerry Tablet OS.

I.3. System d'exploitation Android

I.3.1. Définition

Android est le système d'exploitation mobile créé par Google. Il équipe la majorité des téléphones portables du moment (smartphones). Son principal concurrent est Apple avec l'iPhone. Android est un système vous permettant de personnaliser votre téléphone, télécharger des applications (navigateur Internet, GPS, Facebook...). Android équipe également les tablettes tactiles. **[10]**

Google a souhaité, en développant Android, créer un système facile à utiliser, qui équipera la majorité des smartphones et tablettes dans le monde. C'est un pari réussi. Le système vous permet de télécharger de nouvelles applications répondant à vos besoins. Votre téléphone, tablette seront donc uniques et vous ressemble. Des milliers de développeurs programment chaque jour des nouvelles applications pour votre téléphone. **[11]**

Il est basé essentiellement sur la simplicité d'utilisation et surtout sur une capacité de personnalisation importante présentant un argument commercial de poids. Pour promouvoir ce système d'exploitation open source, Google lui a conféré des alliés puissants réunis au sein de l'Open Handset Alliance tel que Samsung, Motorola, Sony Ericsson et LG.

Android, à nos jours, a connu plusieurs versions représentées ci-dessous :

Figure 0.4. Evolution des versions Android.

I.3.2 Architecture d'Android

Android est basé sur un kernel linux 2.6.xx, au-dessus du kernel il y a "le hardware abstraction layer" qui permet de séparer la plateforme logique du matériel. Au-dessus de cette couche d'abstraction on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android.

Au-dessus des librairies on retrouve l'Android Runtime, cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle exécutant les applications.

Au-dessus la couche "Android Runtime" et des librairies cœurs on retrouve le Framework permettant au développeur de créer des applications. Enfin au-dessus du Framework il y a les applications. **[12]**

Tout cela représenté dans la figure suivante :

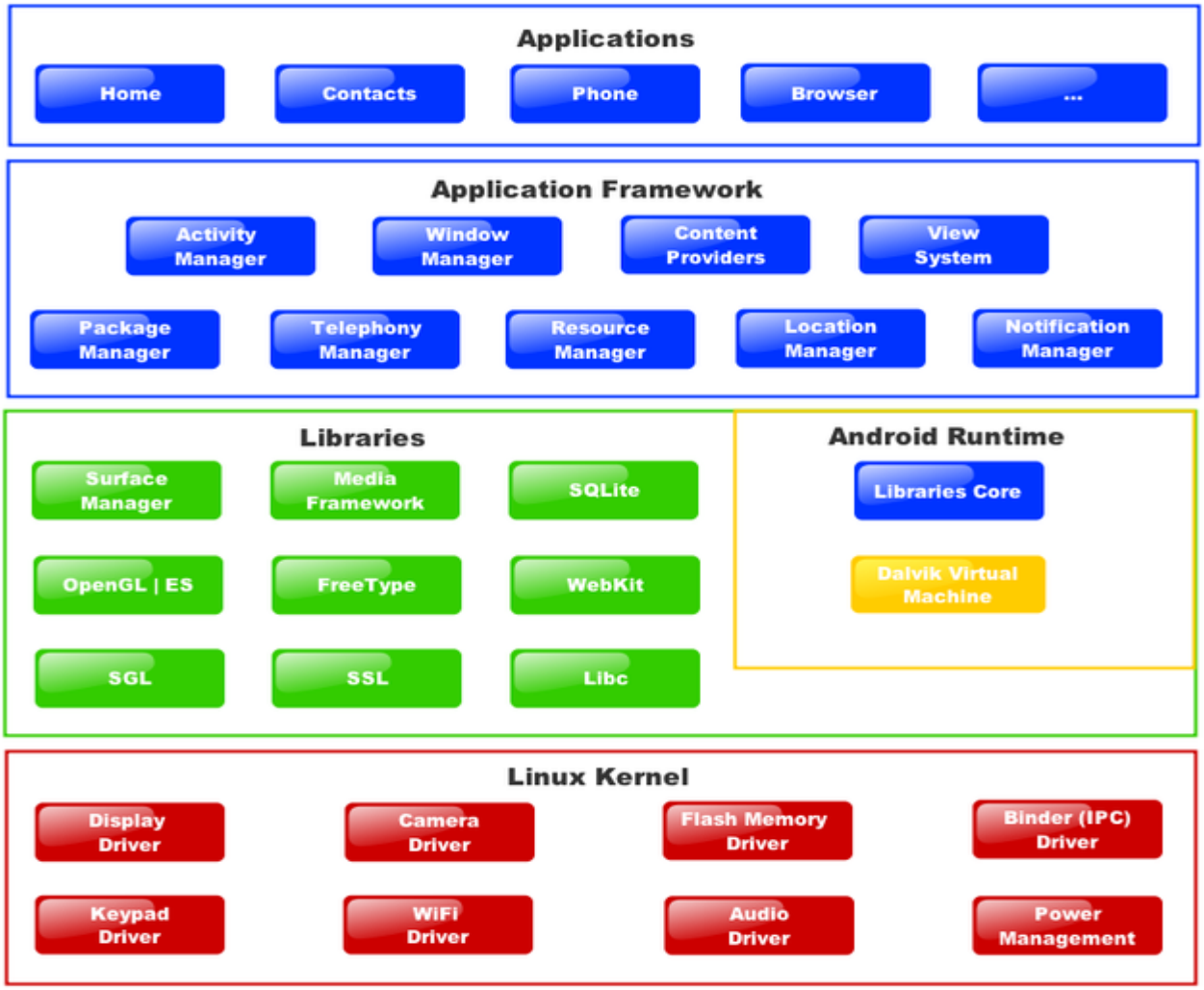


Figure 0.5. Architecture d'Android.

a. Applications

ANDROID est fourni avec un ensemble d'application dont un client-email, une application SMS, un calendrier, un service de cartographie, un navigateur tout écrit en JAVA.

b. Framework de développement

En fournissant une plateforme de développement ouverte, ANDROID offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes. Les développeurs sont libres de profiter du matériel périphérique et informations sur la localisation d'accès, exécuter des services d'arrière-plan, définir des alarmes, ajouter des notifications à la barre d'état, etc.

c. Bibliothèques

ANDROID dispose d'un ensemble de librairie C / C++ utilisé par les différents composants du système ANDROID. Elles sont offertes aux développeurs à travers le Framework ANDROID. En voici quelques-unes :

- **Système de bibliothèque C :** une mise en œuvre dérivée de BSD de la bibliothèque C standard du système (libc), destinés aux systèmes embarqués basés sur Linux.
- **Médiathèques basées sur PacketVideo de OpenCore :** les librairies permettant la lecture et l'enregistrement audio et vidéo, ainsi que la gestion des fichiers image.

d. ANDROID Runtime

ANDROID inclut un ensemble de librairie de base offrant la plupart des fonctionnalités disponibles dans les bibliothèques de base du langage de programmation Java.

e. Noyau Linux

ANDROID est basé sur un noyau linux 2.6 mais ce n'est pas linux. Il ne possède pas de système de fenêtrage natif. ANDROID n'est pas linux mais il est basé sur son noyau. Pourquoi sur un noyau linux.

Le noyau linux possède un système de gestion mémoire et de processus reconnu pour sa stabilité et ses performances. Le kernel linux est entièrement open source et il y a une communauté de développeurs qui l'améliorèrent et rajoutent régulièrement des pilotes des nouveaux périphériques. C'est pour cette raison, l'équipe en charge du noyau a décidé d'utiliser un noyau linux. **[12]**

I.3.3 Cycle de vie d'Android

Le système, pour des raisons de priorisation d'activités (coup de téléphone), peut tuer une activité quand il a besoin de ressources. Pour cette raison, aucune activité ne peut penser pouvoir vivre jusqu'au bout de son traitement, elle doit être considérée comme un humain. Elle peut avoir un accident, être à l'hôpital et/ou mourir.

Une activité possède quatre états que sont :

- **« Active » :** l'activité est lancée par l'utilisateur, elle s'exécute au premier plan.
- **« En Pause » :** l'activité est lancée par l'utilisateur, elle s'exécute et est visible, mais elle n'est plus au premier plan. Une notification ou une autre activité lui a volé le focus et une partie du premier plan.
- **« Stoppée » :** l'activité a été lancée par l'utilisateur, mais n'est plus au premier plan et est invisible. L'activité ne peut interagir avec l'utilisateur qu'avec une notification.
- **« Morte » :** l'activité n'est pas lancée.

Le schéma suivant indique le cycle de vie d'une activité et les méthodes appelées lors des changements d'état :

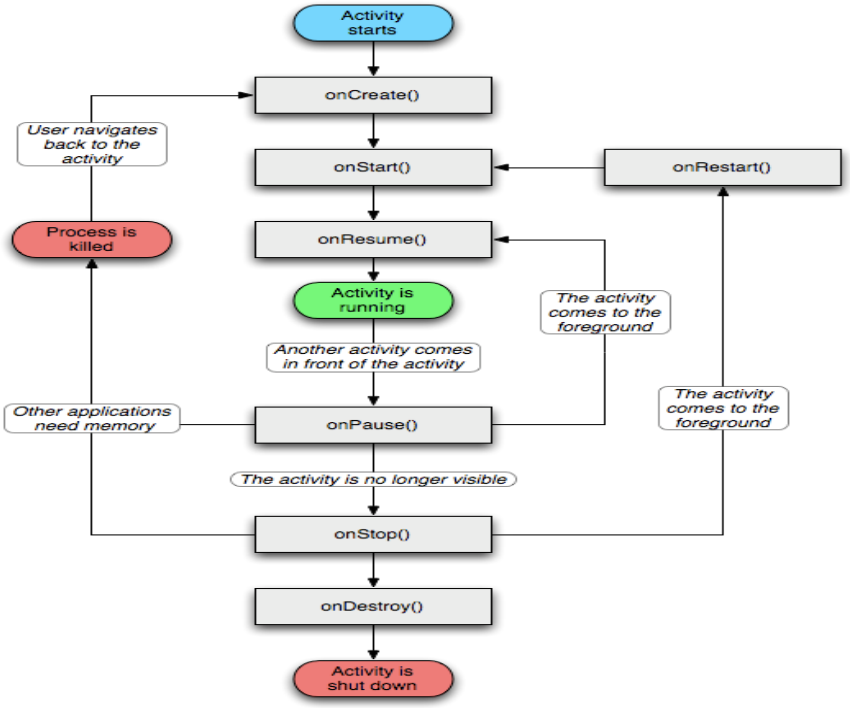


Figure 0.6. Cycle de vie d'Android.

Ainsi, les différentes méthodes sont présentées ci-dessous.

- Méthode onCreate est appelée au premier lancement de l'activité, Si l'activité est ressuscitée, le bundle passé en paramètre sera celui sauvegardé par onSaveInstanceState, Si l'état du terminal change et que l'activité est associée à cet état (passage du mode portrait au mode paysage). Elle configure les IHM et tous les traitements d'initialisation qui ne sont effectués qu'une seule fois au lancement de l'activité.
- Méthode onDestroy est appelée lors de la mort de l'activité (soit naturelle, soit par le système). Parfois, l'urgence du système détruira l'activité sans même appeler onDestroy. Cette méthode doit libérer les ressources allouées dans la méthode onCreate. La seule méthode qui est appelée avec certitude avant la destruction de l'activité est onPause ().

- Méthodes onStart est la fonction qui permet de démarrer les programmes
  , onRestart permet de redemarrer le systeme
  et onStop qui permet d'arrêter totalement le système et elles n'ont pas grand intérêt.

- Méthodes onPause et onResume sont celles dans lesquelles l'activité doit sauvegarder ses états et les restituer.

- Méthode onResume est appelée immédiatement avant que l'activité ne passe au premier plan. De ce fait, c'est le bon endroit pour reconstruire les données affichées par l'IHM et mettre celle-ci à jour, quitte à lancer un thread de reconstruction des données.

Inversement, la méthode onPause est la seule par laquelle il est certain que l'application passera avant de se faire détruire. Il convient donc de sauvegarder l'état de l'application dans cette méthode.

- Sauvegarde et la restauration s'effectuent au travers des méthodes onSaveInstance et onRestoreInstance. L'objet Bundle est passé en paramètre de l'une et renvoyé par l'autre. L'objet Bundle est une carte (map) intelligente qui stocke des paires typées (Key, Object) et possède l'ensemble des méthodes pour enregistrer et récupérer ces paires (**putBoolean**(**String** key, boolean value) et boolean **getBoolean**(**String** key) par exemple et ce pour la plupart des types connus).

Toute la subtilité du mécanisme résidé dans le choix fait par le développeur des données à sauvegarder. Il ne doit pas vouloir y mettre la terre entière et être précis dans ce qu'il choisit de stocker et de restaurer. **[13]**

Partie 2 Méthodologie de conception

I.4. Méthode UP (Unified Process) :

I.4.1. Définition

Le processus unifié est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel. Il est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétence et à différentes tailles de l'entreprise. Le document suivant présente sous la forme d'une note les concepts associés à ce processus. **[14] [15]**

a. UP est itératif

L'itération est une répétition d'une séquence d'instructions ou d'une partie de programme un nombre de fois fixé à l'avance ou tant qu'une condition définie n'est pas remplie, dans le but de reprendre un traitement sur des données différentes. Elle qualifie un traitement ou une procédure qui exécute un groupe d'opérations de façon répétitive jusqu'à ce qu'une condition bien définie soit remplie. **[15]**

Une itération prend en compte un certain nombre de cas d'utilisation et traite en priorité les risques majeurs.

b. UP est piloté par les cas d'utilisation UML

Le but principal d'un système informatique est de satisfaire les besoins du client. Le processus de développement sera donc accès sur l'utilisateur. Les cas d'utilisation permettent d'illustrer ces besoins. Ils détectent puis décrivent les besoins fonctionnels (du point de vue de l'utilisateur), et leur ensemble constitue le modèle de cas d'utilisation qui dicte les fonctionnalités complètes du système.

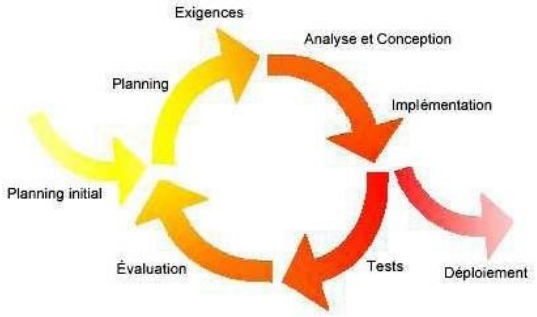


Figure 0.7. Cycle de vie d'UP.

I.4.2. Objectif d'UP

L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques.

UP est un ensemble de principes génériques adapté en fonction des spécificités des projets. UP répond aux préoccupations suivantes :

- **QUI** participe au projet ?
- **QUOI**, qu'est-ce qui est produit durant le projet ?
- **COMMENT** doit-il être réalisé ?
- **QUAND** est réalisé chaque livrable ?

I.4.3. Architecture bidirectionnelle :

UP gère le processus de développement par deux axes.

L'axe vertical représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités, d'enchaînements, d'artefacts et de travailleurs.

L'axe horizontal représente le temps et montre le déroulement du cycle de vie du processus ; cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en termes de cycles, de phases, d'itérations et de jalons. **[15]**

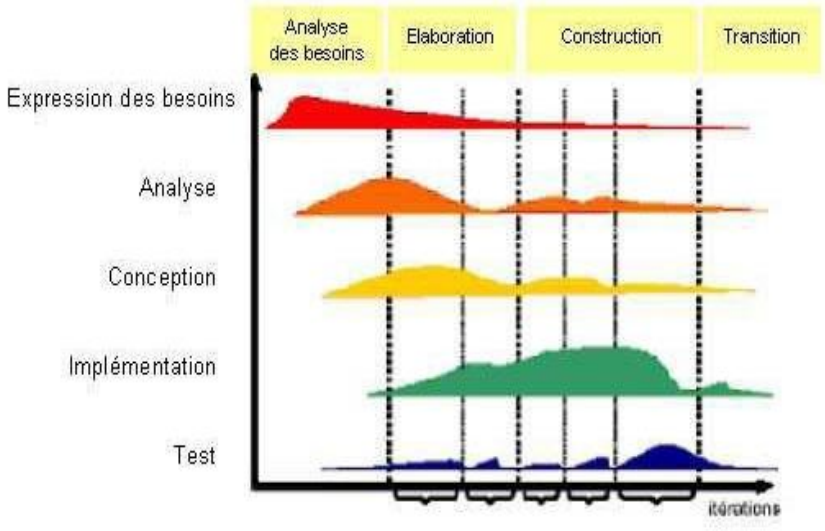


Figure 0.8. Gestion du processus de développement par UP.

UP répète un certain nombre de fois une série de cycles qui s'articulent autour de quatre phases

- Analyse des besoins.
- Elaboration.
- Construction.
- Transition.

Pour mener efficacement un tel cycle, les développeurs ont besoin de toutes les représentations du produit logiciel :

- Un modèle de cas d'utilisation.
- Un modèle d'analyse : détailler les cas d'utilisation et procéder à une première répartition du comportement.
- Un modèle de conception : finissant la structure statique du système sous forme de sous-systèmes, de classes et interfaces graphiques.
- Un modèle d'implémentation : intégrant les composants.
- Un modèle de déploiement : définissant les nœuds physiques des ordinateurs.
- Un modèle de test : décrivant les cas de test vérifiant les cas d'utilisation.
- Une représentation de l'architecture.

I.4.4. Phases d'UP

Le processus UP comporte quatre phases essentielles pour développer une application qui sont :

a. Expression des besoins

L'expression des besoins comme son nom l'indique, permet de définir les différents besoins :

- Inventorier les **besoins principaux** et fournir une liste de leurs fonctions.
- Recenser les **besoins fonctionnels** (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation.
- Appréhender les **besoins non fonctionnels** (technique) et livrer une liste des exigences.

Le modèle de cas d'utilisation présente le système du point de vue de l'utilisateur et représente sous forme de cas d'utilisation et d'acteur, les besoins du client.

b. Elaboration

L'élaboration reprend les éléments de la phase d'analyse des besoins et les précise pour arriver à une spécification détaillée de la solution à mettre en œuvre.

L'élaboration permet de préciser la plupart des cas d'utilisation, de concevoir l'architecture du système et surtout de déterminer l'architecture de référence. Au terme de cette phase, les chefs de projet doivent être en mesure de prévoir les activités et d'estimer les ressources nécessaires à l'achèvement du projet. Les tâches à effectuer dans la phase élaboration sont les suivantes :

- Créer une architecture de référence.
- Identifier les risques, ceux qui sont de nature à bouleverser le plan, le coût et le calendrier.
- Définir les niveaux de qualité à atteindre.
- Formuler les cas d'utilisation pour couvrir les besoins fonctionnels et planifier la phase de construction.
- Elaborer une offre abordant les questions de calendrier, de personnel et de budget.

c. Construction :

La construction est le moment où l'on construit le produit. L'architecture de référence se métamorphose en produit complet. Le produit contient tous les cas d'utilisation que les chefs de projet en accord avec les utilisateurs ont décidé de mettre au point pour cette version.

d. Transition :

Le produit est en version bêta. Un groupe d'utilisateurs essaye le produit et détecte les anomalies et défauts. Cette phase suppose des activités comme la formation des utilisateurs clients, la mise en œuvre d'un service d'assistance et la correction des anomalies constatées.

Conclusion

Durant cette étude de l'environnement Android, les applications mobiles et leur influence dans le monde d'aujourd'hui, nous avons eu une idée globale sur le déroulement du

développement de notre projet. De plus, nous avons compris comment utiliser le Processus Unifié (UP) dans notre projet.

Le chapitre suivant, nous présenterons l'étude des besoins fonctionnels et non fonctionnels de notre application.

Chapitre II Etude de l'existant et spécification des besoins

Chapitre II Etude de l'existant et spécification des besoins

Introduction

Pour le développement de notre application, nous allons nous baser sur le processus unifié. En effet, dans la première phase nous allons effectuer la spécification des besoins, ensuite dans la seconde phase nous allons élaborer l'analyse des besoins, puis nous allons effectuer la conception dans la troisième phase et enfin la dernière phase doit être consacrée à la réalisation suivie par les tests de l'application.

Dans ce chapitre, nous allons présenter l'organisme « Safe Soft » ou nous avons effectué notre stage et faire une étude sur une application de commercialisation de produits existants. Nous présentons la spécification et l'analyse des besoins.

II.1. Présentation de l'organisme d'accueil

Safe Soft est une entreprise Algérienne basée à Mohammadia Mall, El Mohammadia Alger en Algérie. Elle est spécialisée dans le développement de logiciels, vente de matériel informatique destiné à l'utilisation de leurs logiciels et l'installation de différents réseaux informatiques ou de sécurité. Crée le 1er avril 2012, elle compte 11 employés et plus de 20 partenaires agréés dans le territoire national. L'entreprise possède 2 gérants Sebti BOU ABD ELLAH et Mounir Dahmani.

- ✓ **Site web:** https://safesoft-dz.com/.

II.1.1 Secteur d'activité

Safe Soft travail dans plusieurs secteurs d'activités. Parmi eux on trouve :

- Location et vente de matériels informatiques.
- Vente en détail de matériels de bureau
- Agence de communication.
- Bureau d'études et d'orientation en informatique.
- Installation de réseaux et de serveurs.
- Réparation de matériels informatiques et bureautiques.

II.1.2 Rôle du personnel

- ✓ **Les gérants**

Les deux gérants sont chargés de la gestion des employés dans l'entreprise et des relations avec les partenaires.

- ✓ **Le superviseur**

Il est responsable de la communication entre les employés internes de l'entreprise et la supervision des projets et des contrats.

- ✓ **Les ingénieurs et techniciens**

Ils sont chargés du développement, de la maintenance et de l'installation des logiciels informatiques et des différents réseaux.

- ✓ **Service d'accueil**

Ce service gère l'accueil et la commercialisation des différents services et produits de l'entreprise.

- ✓ **Le service après-vente**

Il est responsable de la maintenance des appareils défectueux et des retours de produits en cas de panne et de mise à jour de produits.

II.2. Etude de l'existant

II.2.1. Description et présentation

Jumia est une entreprise de commerce électronique présente sur le marché africain et fondée en 2012. La plateforme de Jumia est une « marketplace » ou place de marché, qui met en relation des vendeurs et des acheteurs, en mettant à leur disposition un service logistique, permettant l'expédition et la livraison des colis en plus d'un service de paiement.

L'entreprise est présente dans plus d'une dizaine de pays africains dont l'Algérie, le Maroc, la Tunisie, le Kenya, l'Égypte, l'Ouganda, le Sénégal, le Ghana, l'Afrique du Sud et la Côte d'Ivoire. Plus de 5 000 personnes travaillent directement avec Jumia, et presque 100 000 personnes indirectement sur le continent. Le 12 avril 2019, Jumia est la première entreprise technologique uniquement consacrée au marché africain cotée à la bourse de New York (JMIA - NYSE). **[16]**

Cette application marchande offre ces services :

- Vente en ligne des jeux, DVD, musique, tv...
- Vente des téléphones, des appareilles photo.
- Vente d'accessoires.
- Vente des électroménagers.
- Vente des vêtements, des chaussures, des maisons brico...
- Vente des logiciels et des matériels informatiques : PC, imprimantes, périphériques réseaux...

- ➢ **Objectif**

C'est une version simplifiée du site « Jumia » développé essentiellement pour faire des achats en ligne de produits flash d'un catalogue web.

- ➢ **Site web:**

https://jumia.fr.uptodown.com/android/telecharger.

- ➢ **Googleplay**

https://play.google.com/store/apps/details?id=com.jumia.android&hl=fr&gl=US.

- **Public visé**

Cette application est destinée aux utilisateurs disposants des appareils téléphoniques intelligents.

II.2.2. Fonctionnalités de l'application

II.2.2.1. Aperçu de la page d'accueil

La page principale est l'interface graphique d'accueil de cette application. Elle représente la première confrontation entre le client et l'application.

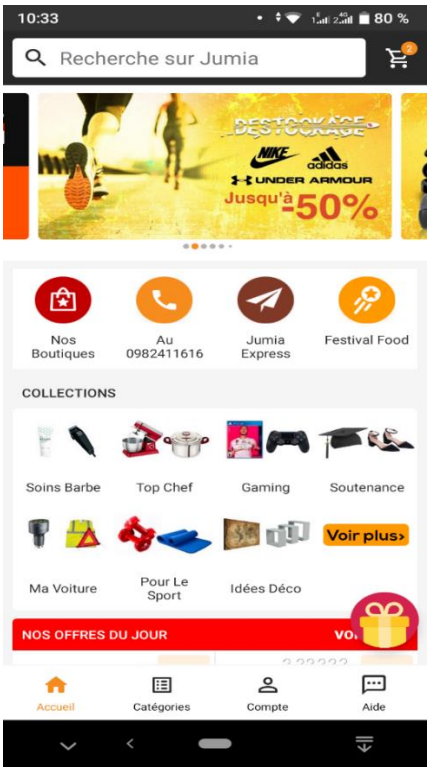


Figure 0.1. Interface « Acueil » de l'application Jumia.

II.2.2.2. Aperçu d'une page interne de l'application « Catalogue »

La page interne de cette application représente le catalogue des produits par catégories qui contient des liens vers une description détaillée des produits.

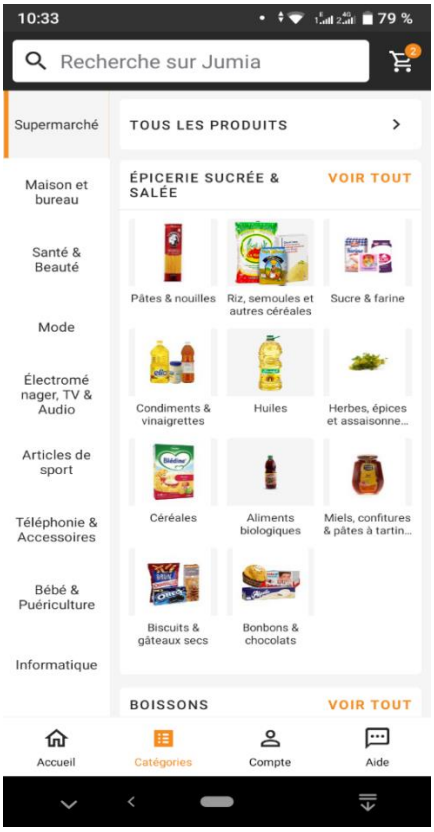


Figure 0.2. Interface « Catalogue » de l'application Jumia.

- L'application possède une lecture horizontale qui permet une lecture plus rapide.
- Respecte le critère de lisibilité et de visibilité
- Cette application permet à l'utilisateur de naviguer dans un catalogue web qui contient tous les produits organisés selon leurs types.

II.2.2.3. Recherche de produits

L'application offre aux clients une barre de recherche pour trouver plus facilement le produit qu'il désire acheter et lui propose un vaste choix de produits avec d'autres de même catégorie que le produit recherché.

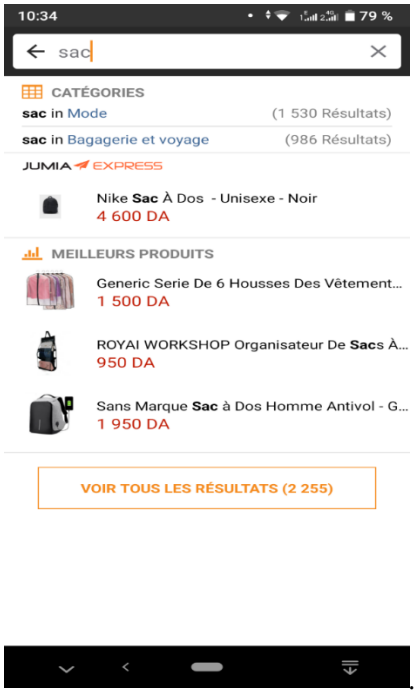


Figure 0.3. Interface « Recherche » de l'application Jumia.

II.2.2.4. Inscription et l'authentification

Le visiteur a la possibilité de s'inscrire aux services de « Jumia », cela lui permettra de consulter son compte, ses achats...

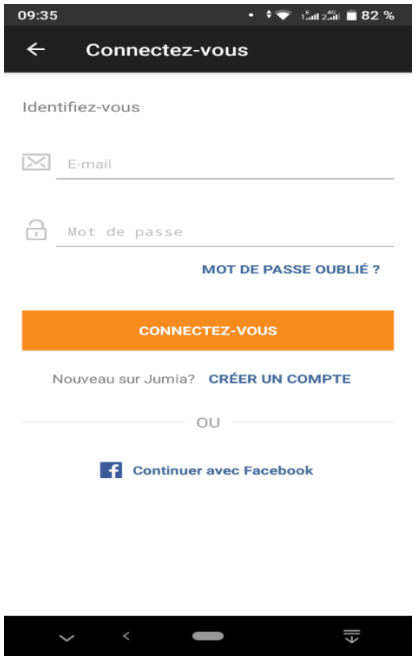


Figure 0.4. Interface « Authentification » de l'application Jumia.

II.3. Spécification des besoins

La spécification de besoins constitue la phase de départ de toute application à développer dans laquelle nous allons identifier les besoins de notre application. Nous distinguons des besoins fonctionnels qui présentent les fonctionnalités attendues de notre application et les

besoins non fonctionnels pour éviter le développement d'une application non satisfaisante ainsi de trouver un accord commun entre les spécialistes et les utilisateurs pour réussir le projet.

II.3.1. Identification des acteurs

Définition d'un acteur : un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositifs matériels ou autres système) qui interagit directement avec le système étudié **[17]**.

Les acteurs de notre application sont :

- **Le Client :** cet acteur est un individu ayant déjà créé un compte sur notre application, il peut donc suivre le processus d'achat des produits en toute sécurité sachant que notre système doit être l'unique responsable de la confidentialité des données personnelles de ses clients.
- **L'administrateur :** c'est le superviseur qui contrôle, qui doit gérer l'application pour assurer le bon fonctionnement de système.

II.3.2. Spécification des besoins fonctionnels

- **Côté Client**
  - ✓ Inscription.
  - ✓ Consulter le catalogue des produits.
  - ✓ Rechercher des produits.
  - ✓ Contacter l'organisme.
  - ✓ Faire une commande.
  - ✓ Afficher la liste de ses commandes
    - Voir les détails d'un achat.
    - Annuler la réservation d'un produit.
    - Ajouter une commande.
  - ✓ Gérer son compte
    - Accéder aux détails de son compte.
    - Modifier ou actualiser ses informations.
  - ✓ Gérer le panier de choix
    - Ajouter un produit au panier.
    - Supprimer un produit du panier.
    - Voir détails du produit.
- **Côté Administrateur** : Après l'authentification, l'administrateur sera redirigé vers son application administrateur qui lui permet de :
  - ✓ Afficher liste des clients
    - Afficher la liste des clients et leurs réservations.
    - Contacter un client.
    - Gérer les clients
      - Afficher les informations d'un client.
      - Modifier un client.
      - Supprimer un client.

- ✓ Gérer le catalogue
  - Ajouter un produit.
  - Modifier un produit.
  - Supprimer un produit.
  - Voir les détails d'un produit.
- ✓ Afficher toutes les commandes des clients
  - Gérer les commandes des clients
    - Valider les commandes d'un client.
    - Annuler les commandes d'un client.
- ✓ Contacter un client.
- ✓ Modifier les informations de l'organisme.

II.3.3. Spécification des besoins non fonctionnels [18]

Les besoins non fonctionnels décrivent toutes les contraintes techniques, ergonomiques et esthétiques auxquelles est soumis le système pour sa réalisation et pour son bon fonctionnement. Et ce qui concerne notre application, nous avons dégagé les besoins suivants :

- ✓ La disponibilité : l'application doit être disponible pour être utilisé par n'importe quel utilisateur.
- ✓ La sécurité de l'accès aux informations critiques : nous devons prendre en considération la confidentialité des données de clients surtout au niveau de l'authentification. Pour cela nous devons restreindre l'accès à ces informations à l'administrateur.
- ✓ La fiabilité : les données fournies par l'application doivent être fiables.
- ✓ La convivialité de l'interface graphique : l'application doit fournir une interface conviviale et simple pour tout type d'utilisateur car elle présente le premier contact de l'utilisateur avec l'application et par le biais de celle-ci on découvrira ses fonctionnalités.
- ✓ Une solution ouverte et évoluée : l'application peut être améliorée par l'ajout d'autres modules pour garantir la souplesse, l'évolutivité et l'ouverture de la solution. La possibilité de retourner au menu principal de l'application à partir de n'importe quelle fenêtre de celle-ci.

II.3.4. Diagramme de cas d'utilisation [19]

Le diagramme des cas d'utilisation (Use Case Diagram) constitue la première étape de l'analyse UML en :

- Modélisant les besoins des utilisateurs.
- Identifiant les grandes fonctionnalités et les limites du système.
- Représentant les interactions entre le système et ses utilisateurs.

Le diagramme des cas d'utilisation apporte une vision utilisateur et pas absolument une vision informatique. Il ne nécessite aucune connaissance informatique et l'idéal serait qu'il soit réalisé par le client.

Le diagramme des cas d'utilisation n'est pas un inventaire exhaustif de toutes les fonctions du système. Il ne liste que des fonctions générales essentielles et principales sans rentrer dans les détails.

Les éléments d'un diagramme des cas d'utilisation

- **Les acteurs :** Avant de rechercher les besoins, la première tâche consiste à définir les limites du système (c.à.d. ce qui est inclus ou pas dans le système), puis à identifier les différentes entités intervenantes sur le système. Ces entités sont appelés acteurs. Les acteurs se représentent sous la forme d'un petit personnage (stick man) ou sous la forme d'une case rectangulaire (appelé classeur) avec le mot clé « actor ». Chaque acteur porte un nom.

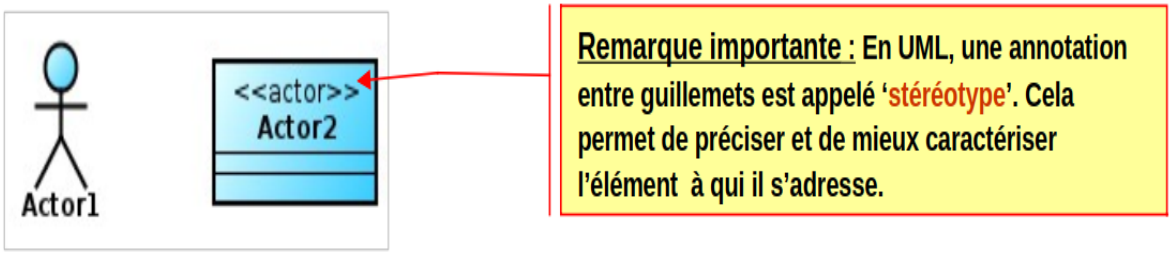


Figure 0.5. Exemple de représentation d'un acteur sous la forme d'un classeur. **[19]**

Un acteur est un utilisateur externe au système. Cela peut être :

- ✓ Une personne.
- ✓ Du matériel (capteurs, moteurs, relais…).
- ✓ Un autre système.

Quelquefois, nous utilisons :

- ✓ Le **stick man** si l'acteur est humain
- ✓ Le **classeur** si l'acteur est du matériel ou un autre système.

- **Les cas d'utilisation :** Le cas d'utilisation représente une fonctionnalité du système (visible de l'extérieur du système). Un cas d'utilisation se représente par une ellipse contenant le nom du cas d'utilisation (phrase commençant par un verbe à l'infinitif) et optionnellement un stéréotype au-dessus du nom.

Figure 0.6. Exemple de représentation d'un cas d'utilisation. **[19]**

Les différents cas d'utilisation peuvent être représentés à l'intérieur d'un même rectangle représentant les limites du système.

- **Relation entre acteurs et cas d'utilisation**
  - ✓ **La relation d'association**
    - A chaque acteur est associé un ou plusieurs cas d'utilisations, la relation d'association peut aussi être appelée relation de communication.
    - Elle est représentée par un trait reliant l'acteur et le cas d'utilisation. Nous pouvons rajouter sur ce trait un stéréotype qui va préciser la relation de communication.
  - ✓ **Multiplicité :** Lorsqu'un acteur peut interagir plusieurs fois avec un cas d'utilisation, il est possible d'ajouter une multiplicité sur l'association du côté du cas d'utilisation. Le symbole * signifie plusieurs. Exactement n s'écrit tout simplement n, n..m signifie entre n et m, etc. Préciser une multiplicité sur une relation n'implique pas nécessairement que les cas sont utilisés en même temps.
- **Les relations entre cas d'utilisation**
  - ✓ **Relation d'inclusion :** La relation d'inclusion sert à enrichir un cas d'utilisation par un autre cas d'utilisation. Dans un diagramme des cas d'utilisation, cette relation est représentée par une flèche pointillée reliant les 2 cas d'utilisation et munie du stéréotype « include ».
  - ✓ **Relation d'extension :** Comme la relation d'inclusion, la relation d'extension enrichit un cas d'utilisation par un autre cas d'utilisation de sous fonction mais celui-ci est optionnel. Cette relation est représentée par une flèche en pointillée reliant les 2 cas d'utilisation et munie du stéréotype « extend ».
  - ✓ **Point d'extension :** L'extension peut intervenir à un point précis du cas étendu. Ce point s'appelle le point d'extension. Il porte un nom, qui figure dans un compartiment du cas étendu sous la rubrique point d'extension, et est éventuellement associé à une contrainte indiquant le moment où l'extension intervient. Une extension est souvent soumise à condition. Graphiquement, la condition est exprimée sous la forme d'une note.

  - ✓ **Relation de généralisation ou de spécialisation**
    - Il est également possible de spécialiser un cas d'utilisation en un autre cas d'utilisation, Nous obtenons alors un sous-cas d'utilisation. Le sous-cas d'utilisation hérite du comportement du sur-cas d'utilisation. Le sous-cas d'utilisation hérite aussi de toutes les associations du sur-cas (relations d'association avec les acteurs, relations d'inclusions, et relations d'extensions).
    - Quelquefois, le sur-cas d'utilisation est abstrait (c'est-à-dire qu'il ne peut pas être instancié). Il correspond à un comportement partiel et sert uniquement de base pour les sous-cas d'utilisation qui en hériteront. La relation de généralisation est représentée par une flèche avec une

extrémité triangulaire. Le nom d'un cas d'utilisation abstrait est écrit en italique (ou accompagné du stéréotype « abstract »).

- **Type d'acteurs et relation entre acteurs**
  - ✓ **Acteurs principaux et secondaires**
    - A chaque cas d'utilisation est associé un ou plusieurs acteurs.
    - Un acteur est principal pour le cas d'utilisation auquel il est lié si ce cas d'utilisation lui rend un service.
    - Les autres acteurs liés à ce cas d'utilisation sont dits secondaires.
    - Normalement, Il ne peut y avoir qu'un seul acteur principal par cas d'utilisation.
    - En général, l'acteur principal sollicite le cas d'utilisation alors que l'acteur secondaire est sollicité par le cas d'utilisation.
    - Un acteur peut être principal pour un cas d'utilisation et secondaire pour un autre cas d'utilisation.
    - Si nous désirons indiquer si l'acteur est principal ou secondaire pour un cas d'utilisation, nous pouvons ajouter les stéréotypes « primary » ou « secondary » sur la relation d'association entre l'acteur et le cas d'utilisation.
  - ✓ **La relation de généralisation :** La seule relation possible entre 2 acteurs est la généralisation (même comportement et même représentation graphique que la relation de généralisation entre 2 cas d'utilisation).

II.3.5. Diagramme de cas d'utilisation global Administrateur-Client

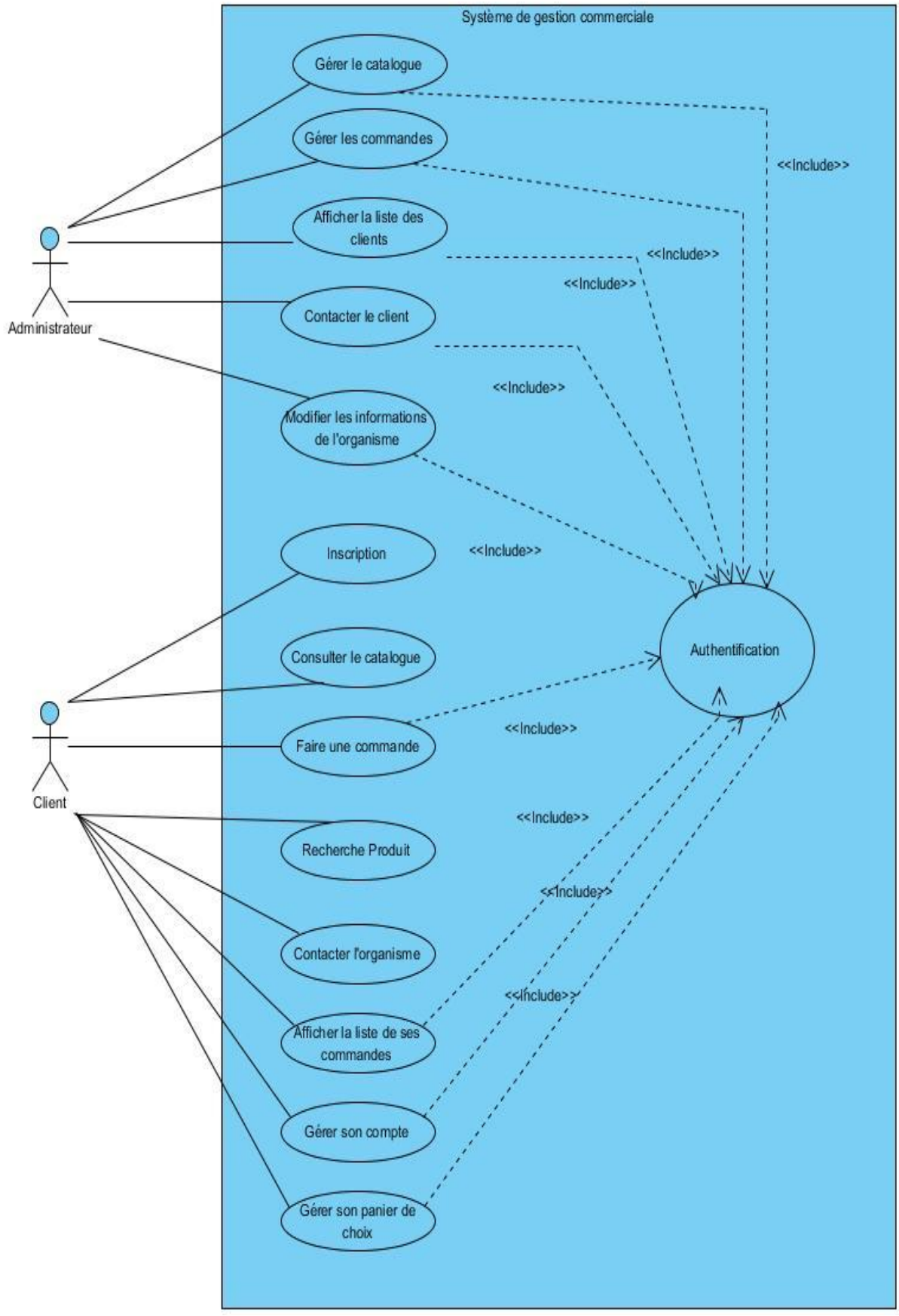


Figure 0.7. Diagramme de cas d'utilisation global.

II.4. Description textuelle des cas d'utilisation

II.4.1. Description des cas d'utilisations pour « Administrateur »

II.4.1.1. Cas d'utilisation « Authentification » pour l'administrateur

Nous allons décrire le diagramme de cas d'utilisation « Authentification » pour l'administrateur.

a. Description de diagramme de cas d'utilisation « Authentification » pour l'administrateur

Sommaire d'identification	
Titre :	Authentification de l'administrateur.
But :	S'authentifier.
Résumer :	L'administrateur introduit son login et son mot de passe.
Acteur :	Administrateur.
Description des enchaînements	
Pré conditions	Poste conditions
L'administrateur doit avoir un compte.	Accès à la page d'accueil de l'administrateur.
Scenario nominal	
➢ Demande d'accès au système. ➢ Le système lui répond par l'affichage de la boite d'authentification. ➢ L'administrateur saisie son login et son mot de passe. ➢ Le système vérifie les champs. ➢ Le système vérifie l'existence de l'administrateur. • Si l'administrateur existe dans la base de données alors il va réussir à s'authentifier, sinon le système renvoi un message d'erreur (l'administrateur n'existe pas). • Si l'authentification est réussie, il aura accès à la page d'accueil, sinon le système réinitialise la page d'authentification.	
Enchainement alternatif	
E1. Champs obligatoires vides. ➢ Le système affiche un message d'erreur (champ vide). ➢ Le scenario reprend de l'étape 3. E2. Login ou mot de passe non valide. ➢ Le système affiche un message d'erreur (login ou mot de passe non valide). ➢ Le scenario reprend de l'étape 3.	

Tableau II.1. Description de diagramme de cas d'utilisation « Authentification » pour l'administrateur.

b. Diagramme de cas d'utilisation « Authentification » pour l'administrateur

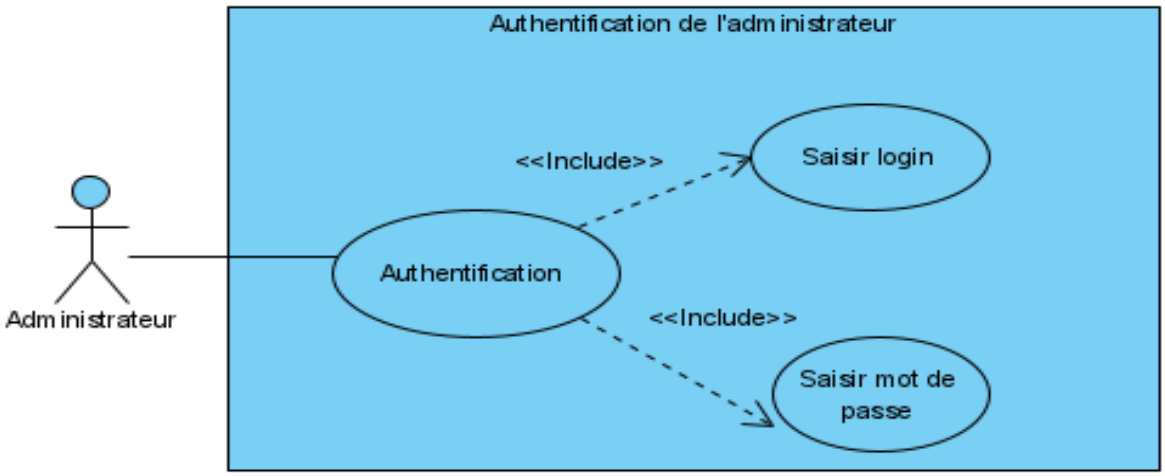


Figure 0.8. Diagramme de cas d'utilisation « Authentification » pour l'administrateur.

II.4.1.2. Cas d'utilisation « Gestion de catalogue des produits » pour l'administrateur

Nous allons décrire le diagramme de cas d'utilisation « Gestion de catalogue des produits » pour l'administrateur.

a. Description de diagramme de cas d'utilisation « Gestion de catalogue des produits » pour l'administrateur

Sommaire d'identification	
Titre :	Gestion de catalogue des produits.
But :	Gérer le catalogue des produits.
Résumer :	L'administrateur va gérer la liste du catalogue en ajoutant, modifiant ou supprimant un produit.
Acteur :	Administrateur.
Description des enchaînements	
Pré conditions	Poste conditions
➢ L'administrateur est authentifié. ➢ L'administrateur choisie le produit à ajouter, modifier ou supprimer.	Produit ajouté, modifier ou supprimer.
Scenario nominal	
➢ Demande d'accès au système. ➢ Le système lui répond par l'affichage de l'interface de gestion de catalogue des produits. ➢ L'administrateur choisie l'action à réaliser : ajouter, modifier ou supprimer un produit. ➢ Le système vérifie l'existence du produit dans la base de données. ➢ Le système affiche la réussite ou l'échec de l'opération.	

Enchainement alternatif
E1. Le produit existe déjà. Dans le cas de l'ajout : ➢ Le système affiche un message d'erreur 'produit déjà existant'. ➢ Le scenario reprend de l'étape 2. Dans le cas de la modification ou de la suppression : ➢ Le système affiche un message 'Produit mise à jour avec succès' ou 'Produit supprimé avec succès'. ➢ Le scenario reprend de l'étape 2. E2. Produit n'existe pas. Dans le cas de l'ajout : ➢ Le système affiche un message 'Produit ajouté avec succès '. ➢ Le scenario reprend de l'étape 2. Dans le cas de la modification ou de la suppression : ➢ Le système affiche un message d'erreur 'Vous devez sélectionner un produit'. ➢ Le scenario reprend de l'étape 2.

Tableau II.2. Description de diagramme de cas d'utilisation « Gestion de catalogue des produits » pour l'administrateur.

b. Diagramme de cas d'utilisation « Gestion de catalogue des produits » pour l'administrateur

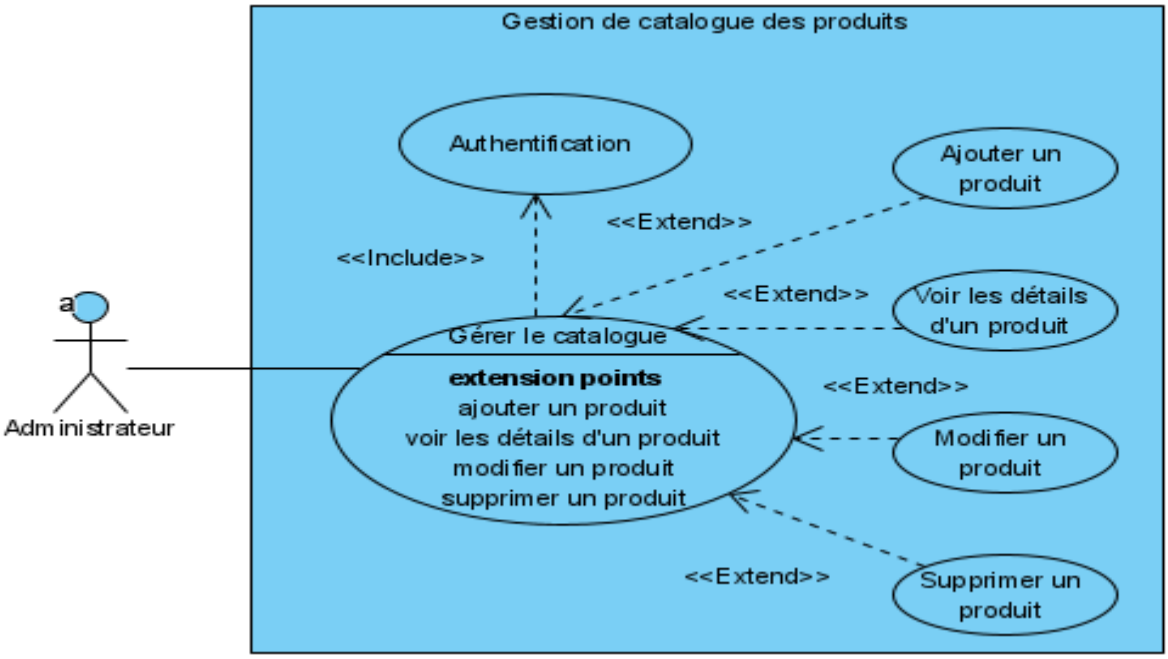


Figure 0.9. Diagramme de cas d'utilisation « Gestion de catalogue des produits » pour l'administrateur.

II.4.1.3. Cas d'utilisation « Afficher liste des clients » pour l'administrateur

Nous allons décrire le diagramme de cas d'utilisation « Afficher liste des clients » pour l'administrateur.

a. Description de diagramme de cas d'utilisation « Afficher liste des clients »

<table>
<tr><td colspan="2">Sommaire d'identification</td></tr>
<tr><td>Titre :</td><td>Afficher liste des clients.</td></tr>
<tr><td>But :</td><td>Afficher la liste des clients ou gérer les clients.</td></tr>
<tr><td>Résumer :</td><td>➢ L'administrateur va afficher la liste des clients.
➢ Une fois la liste des clients est affichée l'administrateur a le choix modifier, supprimer, afficher les informations ou les réservations d'un client.
➢ Le client peur aussi contacter le client.</td></tr>
<tr><td>Acteur :</td><td>Administrateur.</td></tr>
<tr><td colspan="2">Description des enchaînements</td></tr>
<tr><td>Pré conditions</td><td>Poste conditions</td></tr>
<tr><td>➢ L'administrateur est authentifié.
➢ L'administrateur choisie le client à modifier, supprimer ou même d'afficher ses informations ou ses commandes.
➢ L'administrateur choisie le client à contacter.</td><td>Compte modifié ou non.

Compte supprimé ou non.

Liste des clients est vide.</td></tr>
<tr><td colspan="2">Scenario nominal</td></tr>
<tr><td colspan="2">➢ Demande d'accès au système.
➢ L'administrateur choisie un client pour : contacter, modifier, supprimer, afficher ses informations ou ses commandes.
➢ Le système affiche la réussite ou l'échec de l'opération.</td></tr>
<tr><td colspan="2">Enchainement alternatif</td></tr>
<tr><td colspan="2">E1. Si la liste des clients est vide :

➢ Le système affiche un message 'liste des clients est vide'.
E2. Sinon : Le système affiche l'interface liste des clients.

Dans le cas de la modification :

Si le client existe déjà :

➢ Le système affiche un message d'erreur 'client déjà existant'.
➢ Le scenario reprend de l'étape 3.</td></tr>
</table>

Sinon : Le système affiche un message 'compte modifié avec succès'.

Dans le cas de la suppression :

- Le système affiche un message 'compte supprimé avec succès'.
- Le scenario reprend de l'étape 2.

Dans le cas de l'affichage des informations d'un client :

- Le système affiche l'interface correspondante.

Dans le cas de l'affichage des réservations d'un client :

S'il existe au moins une commande :

- Le système affiche l'interface correspondante.

Sinon : Le système affiche un message 'liste des commandes vide'.

Tableau II.3. Description de diagramme de cas d'utilisation « Afficher liste des clients » pour l'administrateur.

b. Diagramme de cas d'utilisation « Afficher liste des clients » pour l'administrateur

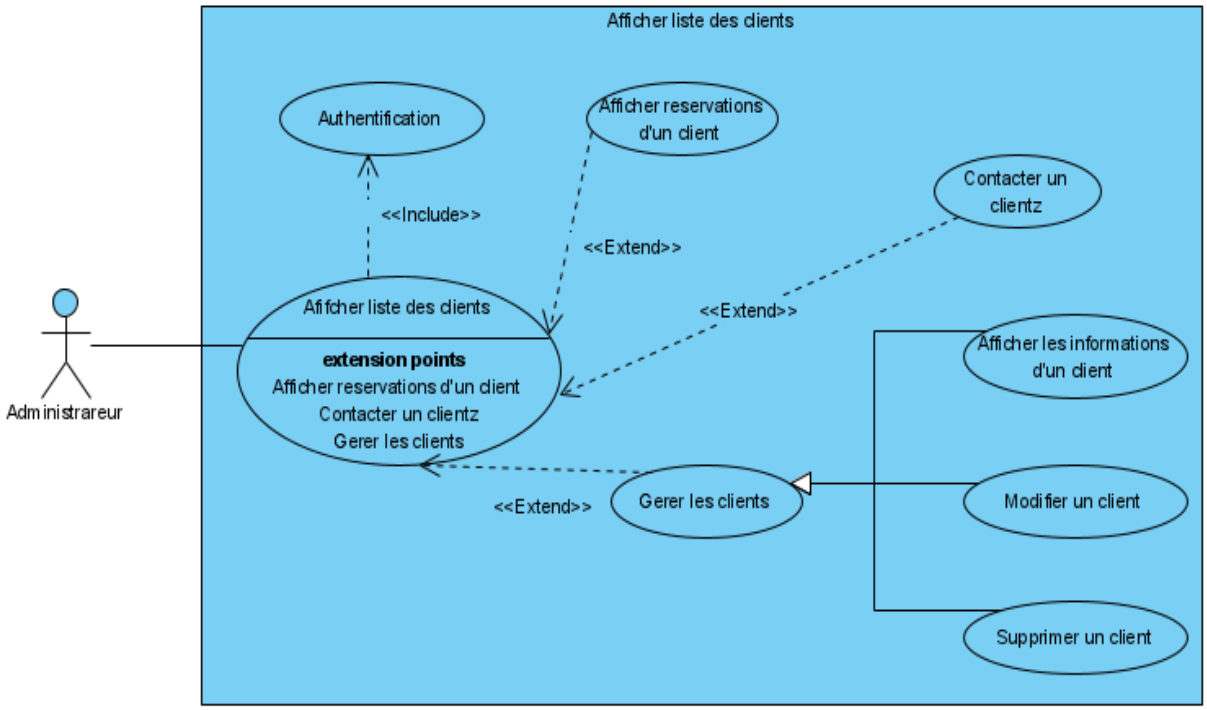


Figure 0.10. Diagramme de cas d'utilisation « Afficher liste des clients » pour l'administrateur.

II.4.1.4. Cas d'utilisation « Gestion des commandes » pour l'administrateur

Nous allons décrire le diagramme de cas d'utilisation « Gestion des commandes » pour l'administrateur.

a. Description de diagramme de cas d'utilisation « Gestion des commandes » pour l'administrateur

Sommaire d'identification	
Titre :	Gestion des commandes.
But :	Gérer les commandes.
Résumer :	L'administrateur va gérer la liste des commandes des clients soit par valider ou annuler une commande.
Acteur :	Administrateur.
Description des enchaînements	
Pré conditions	Poste conditions
➢ L'administrateur est authentifié. ➢ L'administrateur choisi la commande à valider ou à annuler.	Valider ou annuler une commande.
Scenario nominal	
➢ Demande d'accès à la liste des commandes. ➢ Le système lui répond par l'affichage de la liste des commandes. ➢ L'administrateur choisie l'action à réaliser : valider ou annuler une commande. ➢ Le système affiche la réussite de l'opération.	

Tableau II.4. Description de diagramme de cas d'utilisation « Gestion des commandes » pour l'administrateur.

b. Diagramme de cas d'utilisation « Gestion des commandes » pour l'administrateur

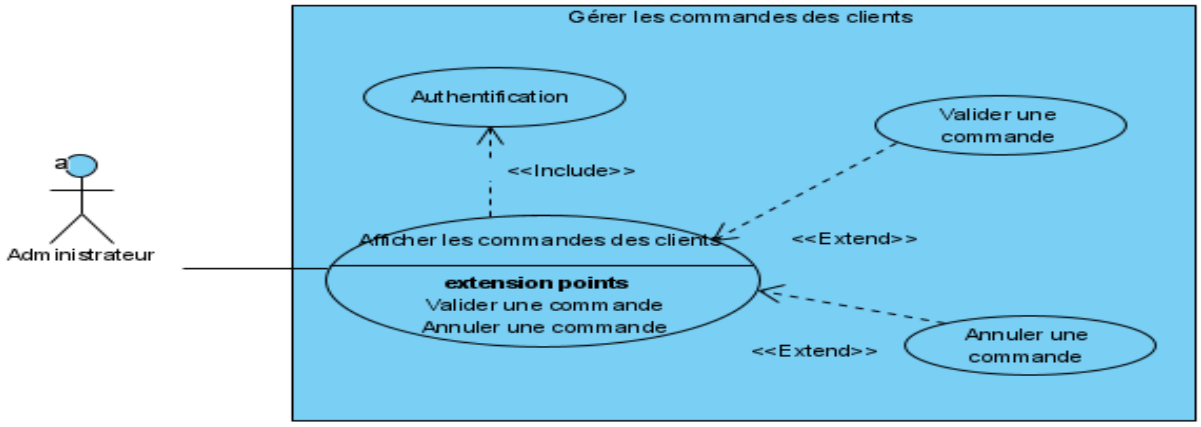


Figure 0.11. Diagramme de cas d'utilisation « Gestion des commandes » pour l'administrateur.

II.4.2. Description des cas d'utilisations pour « Client »

II.4.2.1 Cas d'utilisation « Recherche d'un produit » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Recherche d'un produit » pour le client.

a. Description de diagramme de cas d'utilisation « Recherche d'un produit » pour le client

Sommaire d'identification	
Titre :	Recherche d'un produit.
But :	Rechercher un produit.
Résumer :	➢ Le client va lancer une recherche grâce à un moteur de recherche, il tape un mot clé et le système affiche tous les produits qui répondent à la requête. ➢ Une fois le produit trouvé, le client aura le choix de consulter les détails d'un produit, de l'ajouter à son panier ou de réserver le produit.
Acteur :	Client.
Description des enchaînements	
Pré conditions	Poste conditions
Le client consulte le catalogue des produits.	Produit trouvé ou non.
Scenario nominal	
➢ Le client saisie un mot clé. ➢ Le système cherche tous les produits en lien avec le mot clé. ➢ Le système affiche les résultats. ➢ Le client peut choisir une action : • Ajouter un produit à son panier. • Consulter les détails d'un produit. • Faire une commande sur un produit.	

Tableau II.5. Description de diagramme de cas d'utilisation « Recherche d'un produit » pour le client.

Diagramme de cas d'utilisation « Recherche d'un produit » pour le client

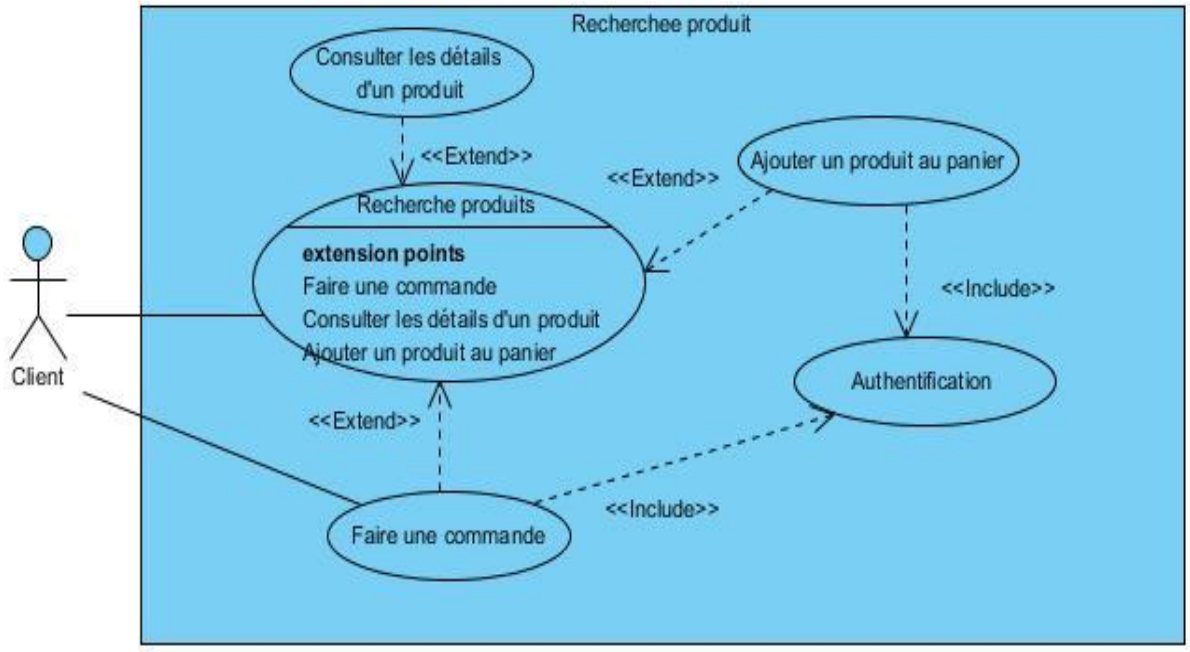


Figure 0.12. Diagramme de cas d'utilisation « Recherche d'un produit » pour le client.

II.4.2.2. Cas d'utilisation « Consultation de catalogue des produits » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Consultation de catalogue des produits » pour le client.

a. Description de diagramme de cas d'utilisation « Consultation de catalogue des produits » pour le client

Sommaire d'identification	
Titre :	Consultation du catalogue.
But :	Consulter le catalogue.
Résumer :	➢ Le client va consulter le catalogue en le sélectionnant. Le système répond à la requête en l'affichant. ➢ Après, le client peut trier le catalogue et aura le choix de réserver un produit, d'ajouter un produit à son panier ou de consulter les détails d'un produit.
Acteur :	Client.
Description des enchaînements	
Pré conditions	Poste conditions
Aucune.	Aucune.
Scenario nominal	

➢ Le client demande la consultation du catalogue. ➢ Le système affiche le catalogue. ➢ Le client peut choisir une action : • Ajouter un produit à son panier. • Consulter les détails d'un produit. • Faire une commande sur un produit. • Trier le catalogue.

Tableau II.6. Description de diagramme de cas d'utilisation « Consultation de catalogue des produits » pour le client.

b. Diagramme de cas d'utilisation « Consultation de catalogue des produits » pour le client

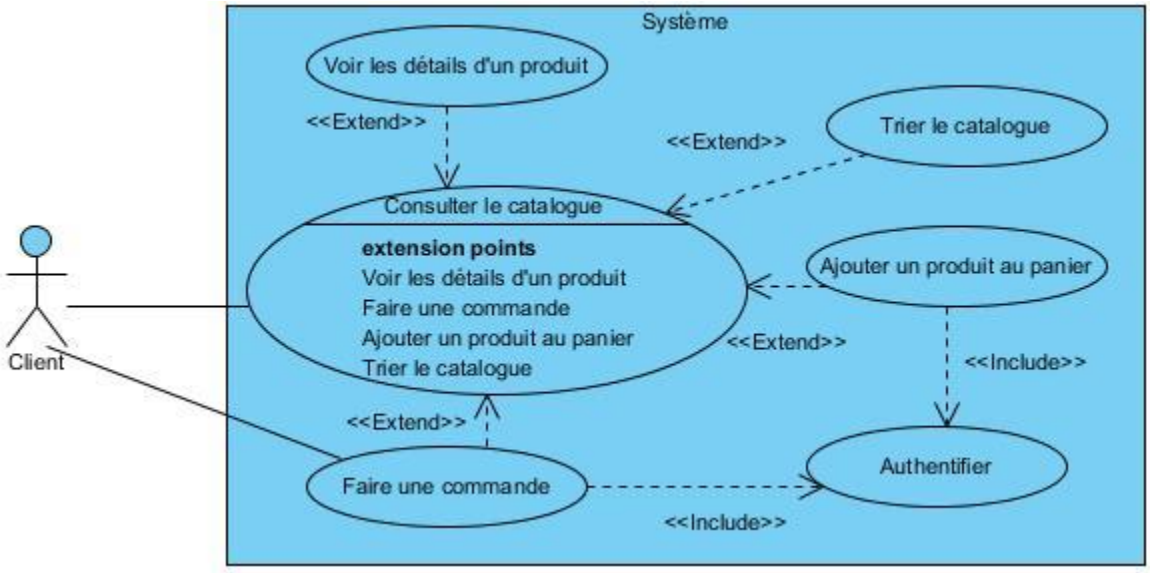


Figure 0.13. Diagramme de cas d'utilisation « Consultation de catalogue des produits » pour le client.

II.4.2.3. Cas d'utilisation « Contacter l'organisme » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Contacter l'organisme » pour le client.

a. **Description de diagramme de cas d'utilisation « Contacter l'organisme » pour le client**

Sommaire d'identification	
Titre :	Contacter l'organisme.
But :	Contacter l'organisme.
Résumer :	Le client peut contacter et voir les informations générales concernant l'organisme.
Acteur :	Client.
Description des enchaînements	

Pré conditions	Poste conditions
Le client est authentifié.	Aucune.
Scenario nominal	
➢ Le client demande de consulter les informations de l'organisme. ➢ Le système affiche les informations au client. ➢ Le client contacte l'organisme.	

Tableau II.7. Description de diagramme de cas d'utilisation « Contacter l'organisme » pour le client.

b. Diagramme de cas d'utilisation « Contacter l'organisme » pour le client

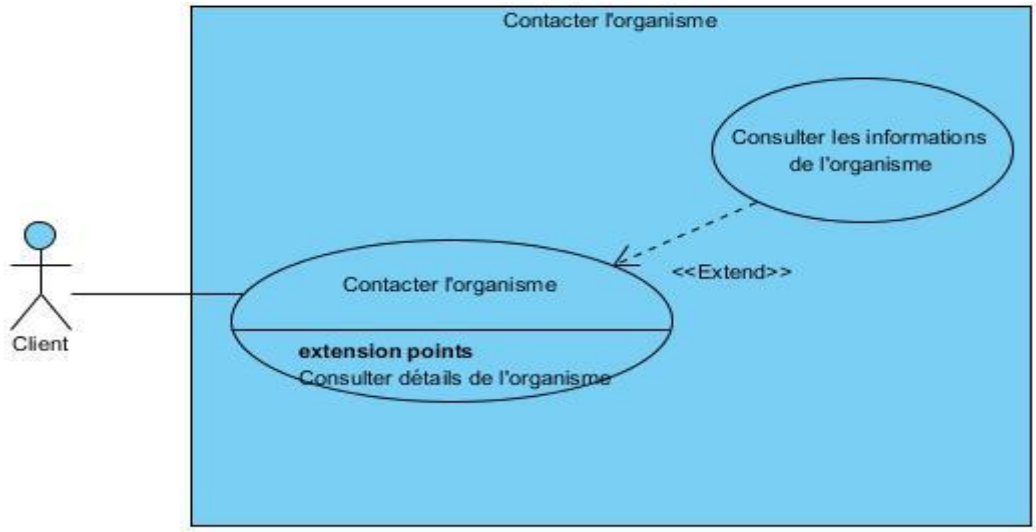


Figure 0.14. Diagramme de cas d'utilisation « Contacter l'organisme » pour le client.

II.4.2.4. Cas d'utilisation « Faire une commande » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Faire une commande » pour le client.

a. Description de diagramme de cas d'utilisation « Faire une commande » pour le client

Sommaire d'identification	
Titre :	Faire une commande.
But :	Réserver un produit.
Résumer :	Le client consulte un produit et clique sur « réserver le produit ». Le système enregistre la commande.
Acteur :	Client.
Description des enchaînements	

Pré conditions	Poste conditions
➢ Le client est authentifié. ➢ Le client consulte une des interfaces (Catalogue, Recherche, Mon Panier).	Produit réservé. Produit non réservé.
Scenario nominal	
1. Le client consulte les détails d'un produit. 2. Le client choisie de réserver le produit. 3. Le système vérifie la disponibilité du nombre d'unités du produit dans la base de données. 4. Le système enregistre la commande du client ou affiche un message s'erreur.	
Enchainement alternatif	
E1. Le nombre d'unités du produit n'est pas disponible. ➢ Le système affiche un message d'erreur 'Réservation du produit non accomplie, stock achevé'. ➢ Le scenario reprend de l'étape 1. E2. Le nombre d'unités du produit disponible. ➢ Le système diminue le nombre d'unités de ce produit dans la quantité de stock dans la base de données. ➢ Le système affiche un message 'Réservation du produit réussie'. ➢ Le scenario reprend de l'étape 1. E3. Le produit n'est pas vendu pendent trois jours à partir de la date de réservation. ➢ Le système modifie la base en augmentant à un nombre d'unité donné de la quantité du stock de ce produit.	

Tableau II.8. Description de diagramme de cas d'utilisation « Faire une commande » pour le client.

b. Diagramme de cas d'utilisation « Faire une commande » pour le client

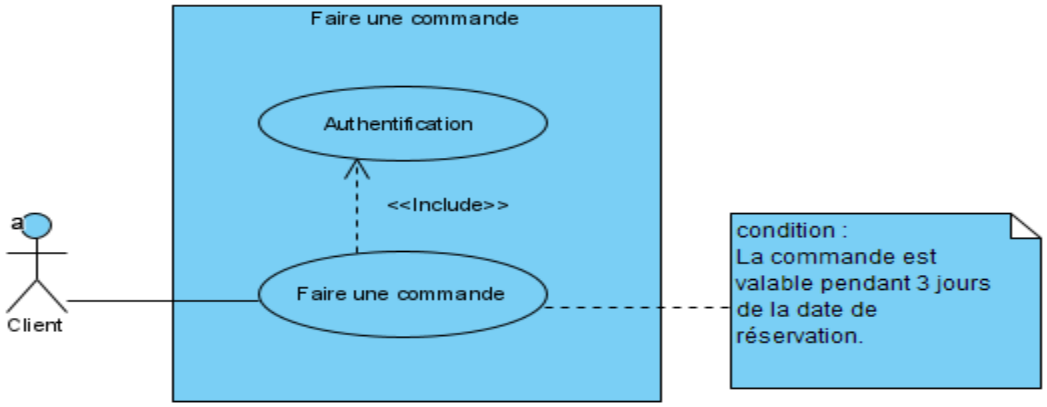


Figure 0.15. Diagramme de cas d'utilisation « Faire une commande » pour le client.

II.4.2.5. Cas d'utilisation « Gestion du panier » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Afficher la liste de ses commandes » pour le client.

a. Description de diagramme de cas d'utilisation « Afficher la liste de ses commandes » pour le client

<table>
<tr><td colspan="2">Sommaire d'identification</td></tr>
<tr><td>Titre :</td><td>Afficher la liste de ses commandes.</td></tr>
<tr><td>But :</td><td>Gérer les commandes.</td></tr>
<tr><td>Résumer :</td><td>Le client va gérer ses commandes en consultant les détails de chaque achat comme il peut annuler la réservation d'un produit non payé ou ajouter une commande.</td></tr>
<tr><td>Acteur :</td><td>Client.</td></tr>
<tr><td colspan="2">Description des enchaînements</td></tr>
<tr><td>Pré conditions</td><td>Poste conditions</td></tr>
<tr><td>➢ Le client est authentifié.
➢ Le client accède à sa liste des commandes.</td><td>L'affichage ou non de la liste de commandes.</td></tr>
<tr><td colspan="2">Scenario nominal</td></tr>
<tr><td colspan="2">1. Le client demande d'afficher sa liste des commandes.
2. Le système affiche la liste des commandes du client.
3. Le client peut choisir une action :
• Consulter les détails d'un achat.
• Annuler la réservation d'un produit.
• Ajouter une commande.</td></tr>
<tr><td colspan="2">Enchainement alternatif</td></tr>
<tr><td colspan="2">E1. La liste des commandes est vide.

➢ Le système affiche un message d'erreur ' la liste des commandes est vide '.
➢ Le scenario reprend de l'étape 1.
E2. La liste des commandes n'est pas vide.

➢ Le système affiche la liste des commandes.</td></tr>
</table>

Tableau II.9. Description de diagramme de cas d'utilisation « Afficher la liste de ses commandes » pour le client.

b. Diagramme de cas d'utilisation « Afficher la liste de ses commandes » pour le client

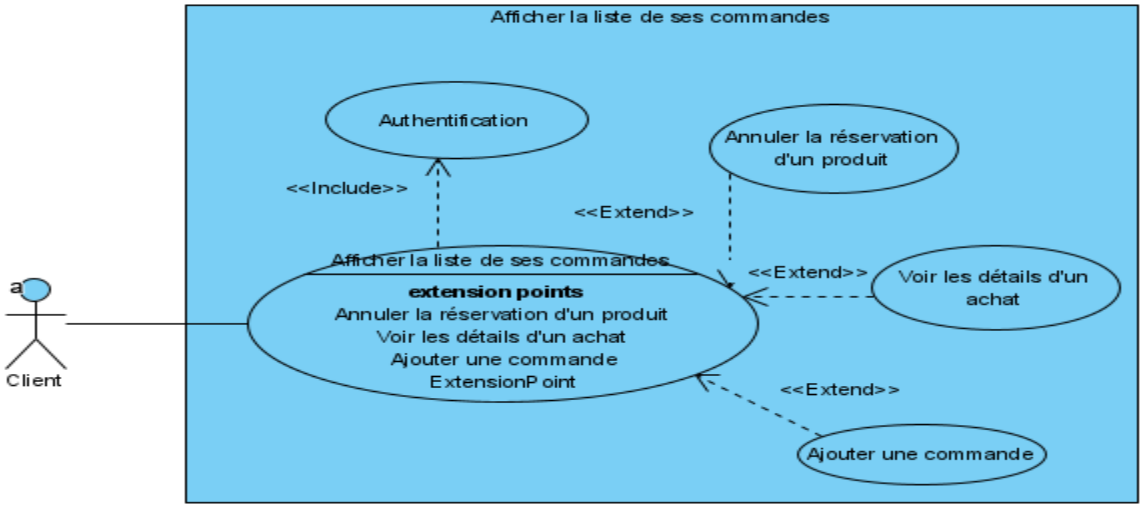


Figure 0.16. Diagramme de cas d'utilisation « Afficher la liste de ses commandes » pour le client.

II.4.2.6. Cas d'utilisation « Gérer son compte » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Gérer son compte » pour le client.

a. Description de diagramme de cas d'utilisation « Gérer son compte » pour le client

Sommaire d'identification	
Titre :	Gérer son compte.
But :	Gérer compte du client.
Résumer :	Le client va gérer son compte en consultant les détails de son compte comme il peut mettre à jour ses informations.
Acteur :	Client.
Description des enchaînements	
Pré conditions	Poste conditions
➢ Le client est authentifié. ➢ Le client accède à son compte.	Aucune.
Scenario nominal	
➢ Le client demande d'accéder à son compte. ➢ Le système affiche les détails de son compte. ➢ Le client peut Modifier les informations de son compte.	

Tableau II.10. Description de diagramme de cas d'utilisation « Gérer son compte » pour le client.

b. Diagramme de cas d'utilisation « Gérer son compte » pour le client

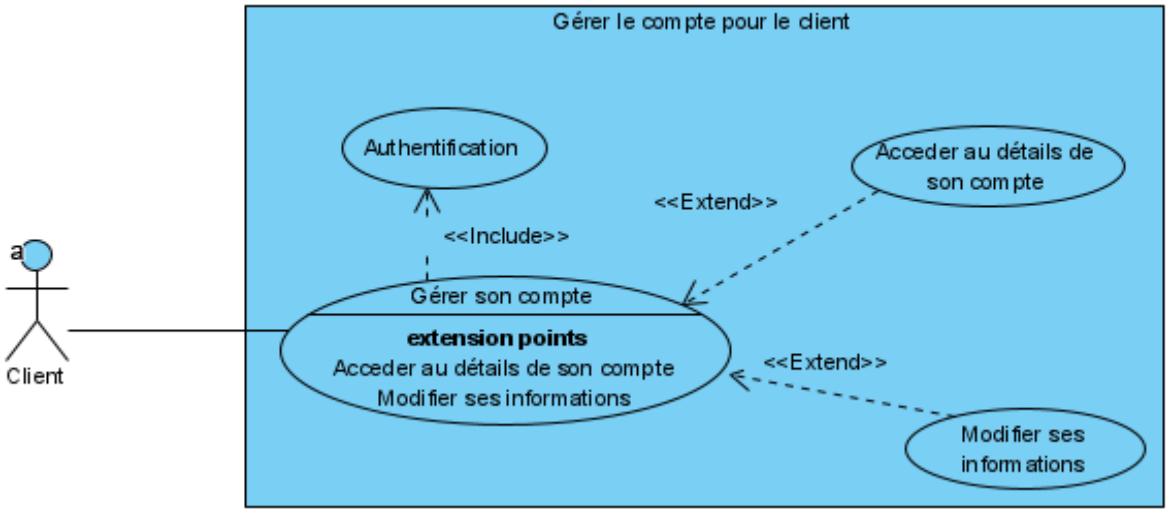


Figure 0.17. Diagramme de cas d'utilisation « Gérer son compte » pour le client.

II.4.2.7. Cas d'utilisation « Gestion du panier » pour le client

Nous allons décrire le diagramme de cas d'utilisation « Gestion du panier » pour le client.

a. Description de diagramme de cas d'utilisation « Gestion du panier » pour le client

Sommaire d'identification	
Titre :	Gestion du panier.
But :	Gérer son panier.
Résumer :	Le client peut ajouter, supprimer ou consulter un produit dans son panier.
Acteur :	Client.
Description des enchaînements	
Pré conditions	Poste conditions
➢ Le client est authentifié.	Aucune.
Scenario nominal	
➢ Le client demande la consultation de son panier. ➢ Le système affiche le panier du client. ➢ Le client peut choisir une action à réaliser : • Consulter les détails d'un produit. • Ajouter un produit au panier. • Supprimer un produit du panier.	

Tableau II.11. Description de diagramme de cas d'utilisation « Gestion du panier » pour le client.

b. Diagramme de cas d'utilisation « Gestion du panier » pour le client

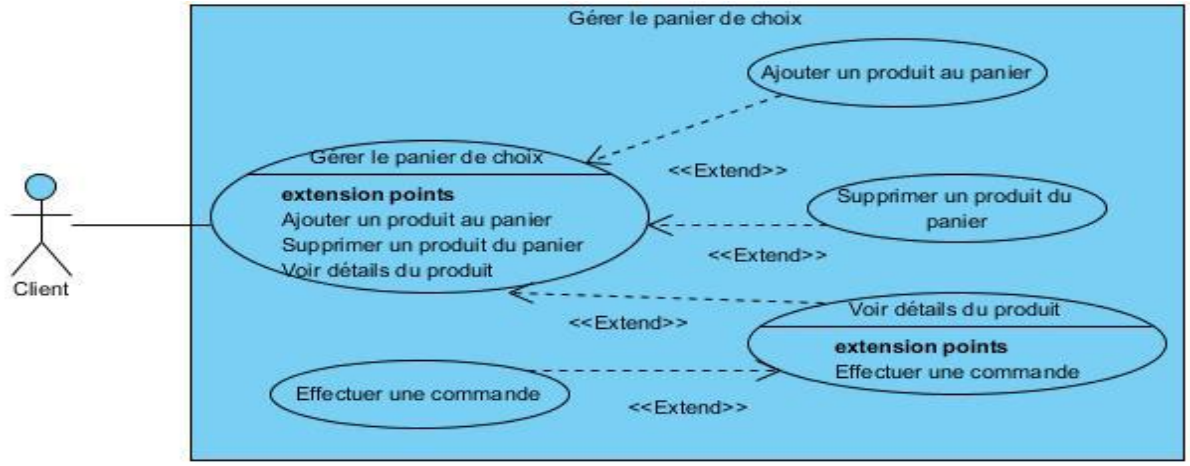


Figure 0.18. Diagramme de cas d'utilisation « Gestion du panier » pour le client.

II.5. Phase d'analyse

II.5.1. Programmation orienté objet

La programmation classique ou procédurale telle que le développeur la connaissait à travers des langages de programmation comme Pascal, C etc. Traite les programmes comme un ensemble de données sur lesquelles agissent des procédures. Les procédures sont les éléments actifs et importants, les données devenant des éléments passifs qui traversent l'arborescence de programmation procédurale en tant que flot d'informations **[19]**.
La programmation orientée objet (POO) est un paradigme au sein de la programmation informatique. Il s'agit d'une représentation des choses, un modèle cohérent partagé à travers différents langages qui permettent son usage (Python, Java, C++). Le but de la POO consiste à définir et faire interagir entre eux des objets **[20]**.
Pour le développement de notre application, nous allons baser sur la programmation orientée objet.

II.5.2. Pourquoi utiliser la programmation orientée objet ?

La programmation orientée objet amène certains avantages par rapport à la programmation procédurale.

Tout d'abord elle permet d'obtenir un code plus réutilisable. En effet, les types d'objets créés peuvent servir de base pour d'autres objets en ne développant que les évolutions nécessaires.

Ensuite cette programmation offre un code plus compréhensible. En effet, chaque objet va contenir ses propriétés et ses méthodes. Il est donc aisé de voir ce qu'une fonction manipule et à quoi correspondent les variables disponibles.

Enfin la programmation orientée objet amène un code modulaire. En effet, chaque type d'objet ne communique avec les autres types que par des interfaces connues et définies. Cet isolement permet donc de développer facilement d'autres modules pour interagir avec les interfaces déjà utilisées **[21]**.

II.5.3. Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

D'après Laurent AUDIBERT **[22]**, le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les cas d'utilisation ne réalisent donc pas une partition des classes du diagramme de classes.

Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de Programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation, etc.

- ✓ **Classe**

Une classe est une représentation abstraite d'un d'ensemble d'objets, elle contient les informations nécessaires à la construction de l'objet (c'est-à-dire la définition des attributs et des méthodes).

La classe peut donc être considérée comme le modèle, le moule ou la notice qui va permette la construction d'un objet. Nous pouvons encore parler de type (comme pour une donnée).

On dit également qu'un objet est l'instance d'une classe (la concrétisation d'une classe).

Une classe est représentée par un rectangle (appelé aussi classeur) divisé en 3 compartiments : nom, attributs et les méthodes **[19]**.

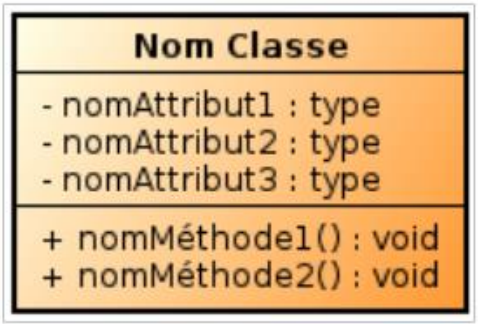


Figure 0.19. Représentation d'une classe. **[19]**

✓ **Les associations**

Elle indique qu'une classe est en relation avec une autre pendant un certain laps de temps. La ligne de vie des deux objets concernés n'est cependant pas associée étroitement (un objet peut être détruit sans que l'autre le soit nécessairement).

L'association est représentée par un simple trait continu, reliant les deux classes. Le fait que deux instances soient ainsi liées permet la navigation d'une instance vers l'autre, et vice-versa (chaque classe possède un attribut qui fait référence à l'autre classe). **[19]**

Figure 0.20. Représentation d'une classe. **[19]**

✓ **La cardinalité (ou multiplicité)**

La cardinalité indique le nombre d'instances de classe étant en relation avec la classe située à l'autre extrémité de l'association. En l'absence de spécification, la cardinalité vaut 1.

Cardinalité	Signification
0..1	Zéro ou une fois
1..1 (ou 1)	Une et une seule fois
0..* (ou *)	De zéro à plusieurs fois
1..*	De une à plusieurs fois
m..n	Entre **m** et **n** fois
n..n (ou n)	**n** fois

Figure 0.21. Les différentes cardinalités et leurs significations. [**19]**

✓ **Relation d'héritage**

Le mécanisme d'héritage permet de mettre en relation des classes ayant des caractéristiques communes (attributs et comportements) en respectant une certaine filiation.

L'héritage indique qu'une classe B est une spécialisation d'une classe A. La classe B (appelé classe fille, classe dérivée ou sous classe) hérite des attributs et des méthodes de la classe A (appelée classe mère, classe de base ou super classe). Il se représente par un triangle vide afin d'indiquer le sens de la généralisation. **[19]**

II.5.4. Modèle du domaine [23]

La modélisation des besoins par des cas d'utilisation s'apparente à une analyse fonctionnelle classique. L'élaboration du modèle des classes du domaine permet d'opérer une transition vers une véritable modélisation objet.

La phase d'analyse du domaine permet d'élaborer la première version du diagramme de classes. Elle est appelée modèle du domaine. Ce modèle doit définir les classes qui

modélisent les entités ou concepts présents dans le domaine (on utilise aussi le terme de métier) de l'application. Il s'agit donc de produire un modèle des objets du monde réel dans un domaine donné.

Il faut absolument utiliser le vocabulaire du métier pour nommer les classes et leurs attributs. Les classes du modèle du domaine ne doivent pas contenir d'opération, mais seulement des attributs. Les étapes à suivre pour établir ce diagramme sont :

- Identifier les entités ou concepts du domaine.
- Identifier et ajouter les associations et les attributs.
- Organiser et simplifier le modèle en éliminant les classes redondantes et en utilisant l'héritage.

Nous allons présenter le modèle de domaine associé à notre système dans la figure suivante :

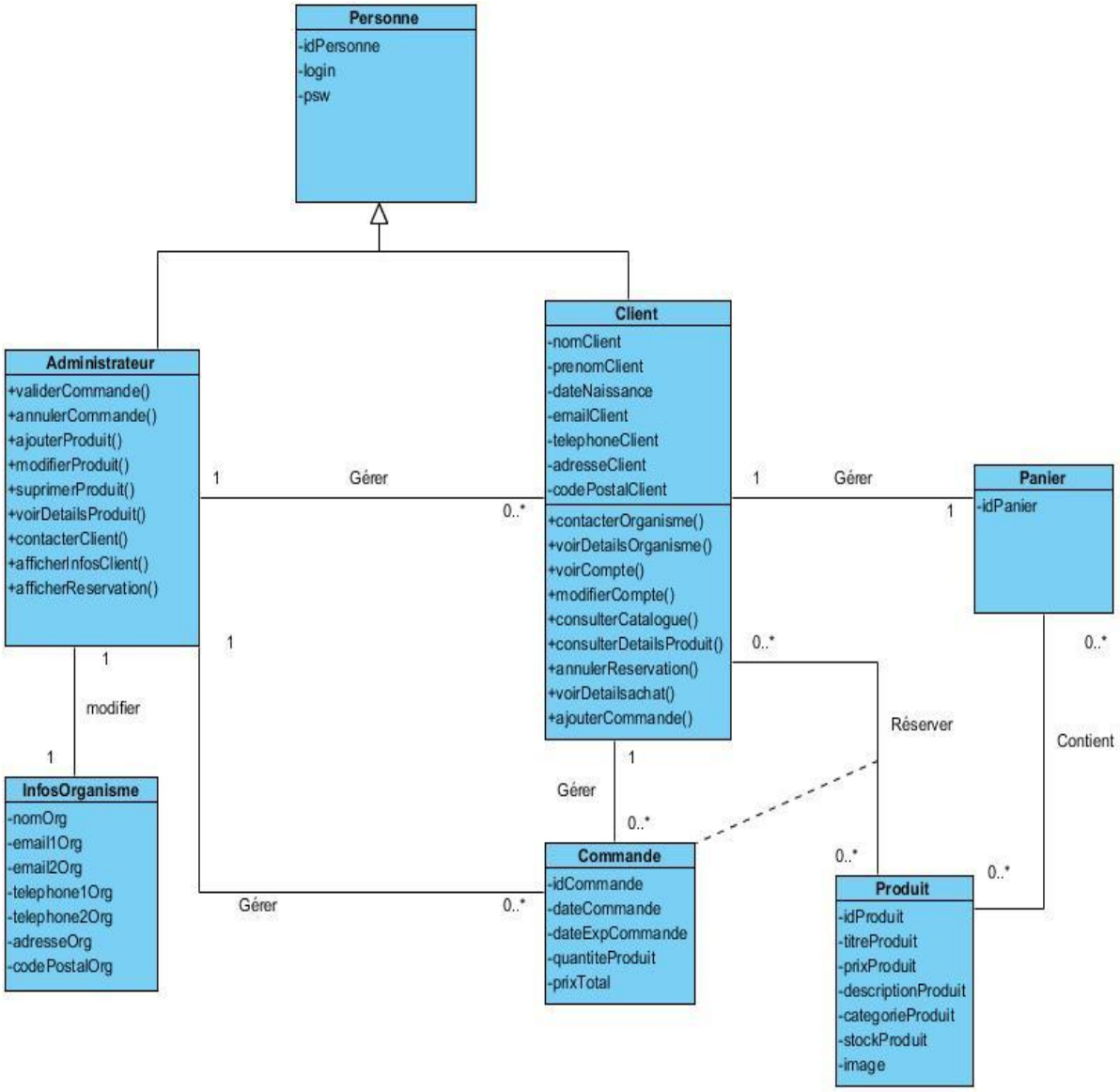
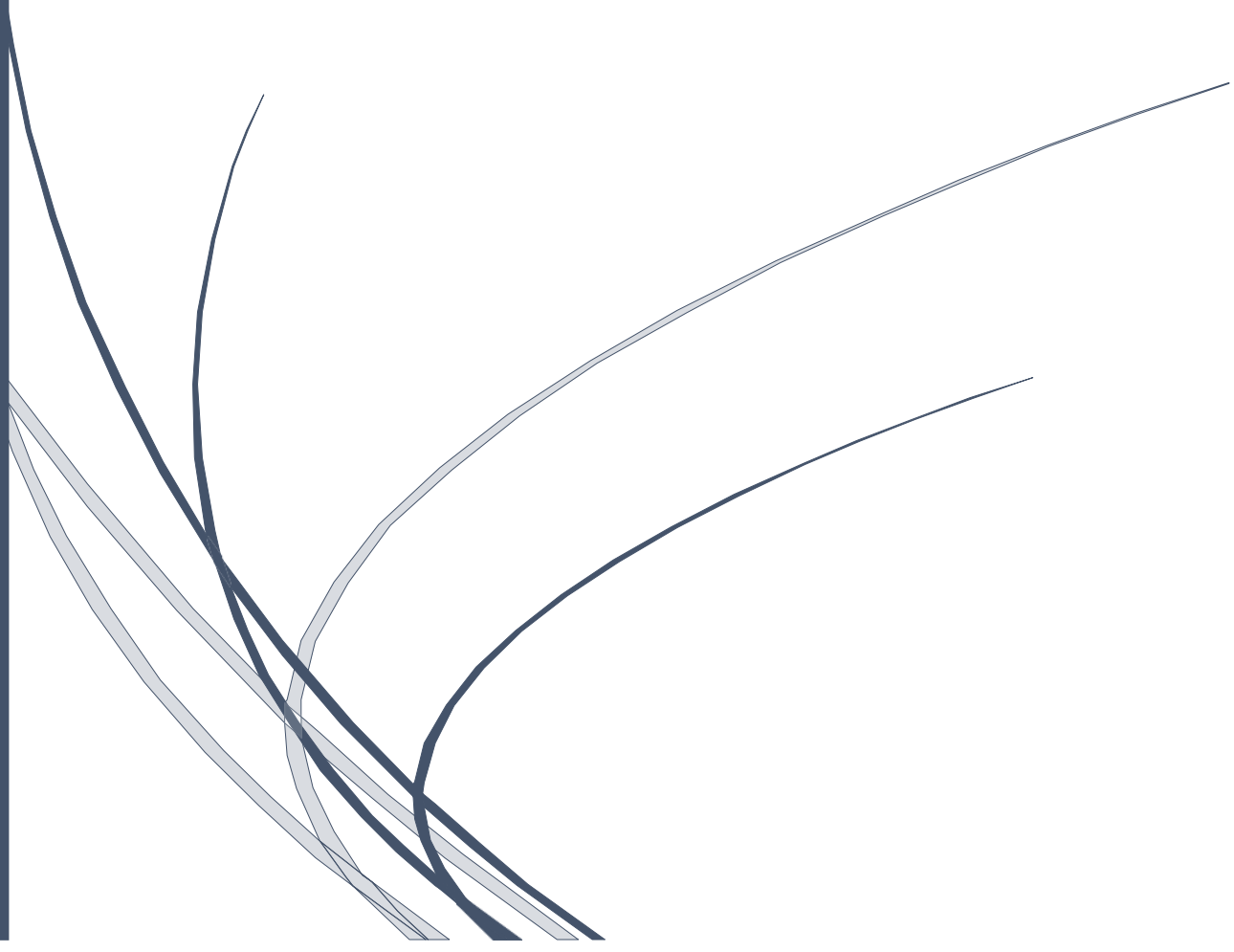


Figure 0.22. Model du domaine.

Conclusion

Cette phase avait pour intérêt de présenter l’organisme d’accueil, un exemple d’application de gestion commerciale et les besoins fonctionnels et non fonctionnels de notre application ; qui consiste à mettre en place une démarche de développement. Nous avons essayé d’exprimer le fonctionnement de notre système en se basant principalement sur les diagrammes de cas d’utilisation et le modèle de domaine. Nous pouvons ainsi entamer la prochaine étape qui consiste à présenter la phase de conception.

Chapitre III Conception



Chapitre III Conception

Introduction

Après avoir tracé les grandes lignes de phase d'analyse et spécification des besoins, mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie d'un logiciel, la phase de conception.

Nous passons, à travers ce chapitre, à décrire la conception élue pour réaliser convenablement le travail demandé. Pour ce faire, nous choisissons de donner en premier lieu une idée sur l'architecture générale que nous allons suivre dans la partie réalisation de notre projet. Puis, dans un deuxième lieu nous allons détailler notre choix conceptuel à travers des diagrammes d'interaction système et des diagrammes de classes conceptuelles.

III.1. Conception architecturale

Il est primordial à la conception de tout système informatique de choisir le modèle d'architecture qui lui sera adéquat pouvant assurer un bon fonctionnement, des meilleures performances ainsi que la réutilisation et l'interconnexion fiable de ce système avec d'autres.

C'est à cet effet que nous optons pour le modèle MVC qui sera également très pratique pour gérer l'interaction entre les différents composants de notre application Android.

Nous décrivons cette architecture dans la section suivante.

III.1.1. Architecture MVC

Le pattern MVC (Modèle, Vue, contrôleur) permet de bien organiser son code source. Il va nous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts **[24]** :

- ✓ **Modèle :** cette partie gère les données de l'application. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc les requêtes SQL.
  Parfois, les données ne sont pas stockées dans une base de données. C'est plus rare, mais on peut être amené à aller chercher des données dans des fichiers. Dans ce cas, le rôle du modèle est de faire les opérations d'ouverture, de lecture et d'écriture de fichier.
- ✓ **Vue :** La vue correspond à l'interface avec laquelle l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic sur l'écran, sélection d'une entrée, boutons, etc). Ces différents événements sont envoyés au contrôleur. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.
- ✓ **Contrôleur :** Le contrôleur sert de base à récupérer les informations, de les traiter en fonction des paramètres demandés par la vue (par l'utilisateur), puis de renvoyer à la

vue les données afin d'être affichées. C'est donc l'élément qui va utiliser les données pour les envoyer à la vue.

L'interaction entre ces trois couches est décrite à l'aide de la figure suivante :

Figure III.1. Architecture MVC. **[25]**

Les avantages apportés par l'architecture MVC sont **[26]**:

- La séparation des données de la vue et du contrôleur (ce qui permet une conception claire et efficace de l'application)
- Une indépendance des données, de l'affichage et des actions (ce qui donne plus de souplesse pour la maintenabilité et l'évolutivité du système).
- Un gain de temps de maintenance et d'évolution de l'application.

III.2 Diagrammes d'interactions système

III.2.1 Définition du diagramme de séquence

La description de la vue dynamique de notre application est réalisée à travers les différents diagrammes d'interaction système. En effet, un diagramme d'interaction système décrit les interactions entre les objets du système. Nous devons, donc, utiliser les différents composants de l'architecture MVC pour mieux éclaircir les tâches, pour cela :

- L'utilisateur émet une requête.
- Le contrôleur intercepte la requête de l'utilisateur.
- Le contrôleur détermine quelle partie du modèle est concernée et quelle vue y est associée.
- Le modèle traite les interactions avec les données, applique les règles métier et renvoie les données au contrôleur.
- Le contrôleur sélectionne la vue et lui renseigne les données.
- La vue présente les données à l'utilisateur.

III.2.2. Diagrammes d'interactions système pour l'administrateur

III.2.2.1. Diagramme d'interaction système du cas « Authentifier » pour l'administrateur

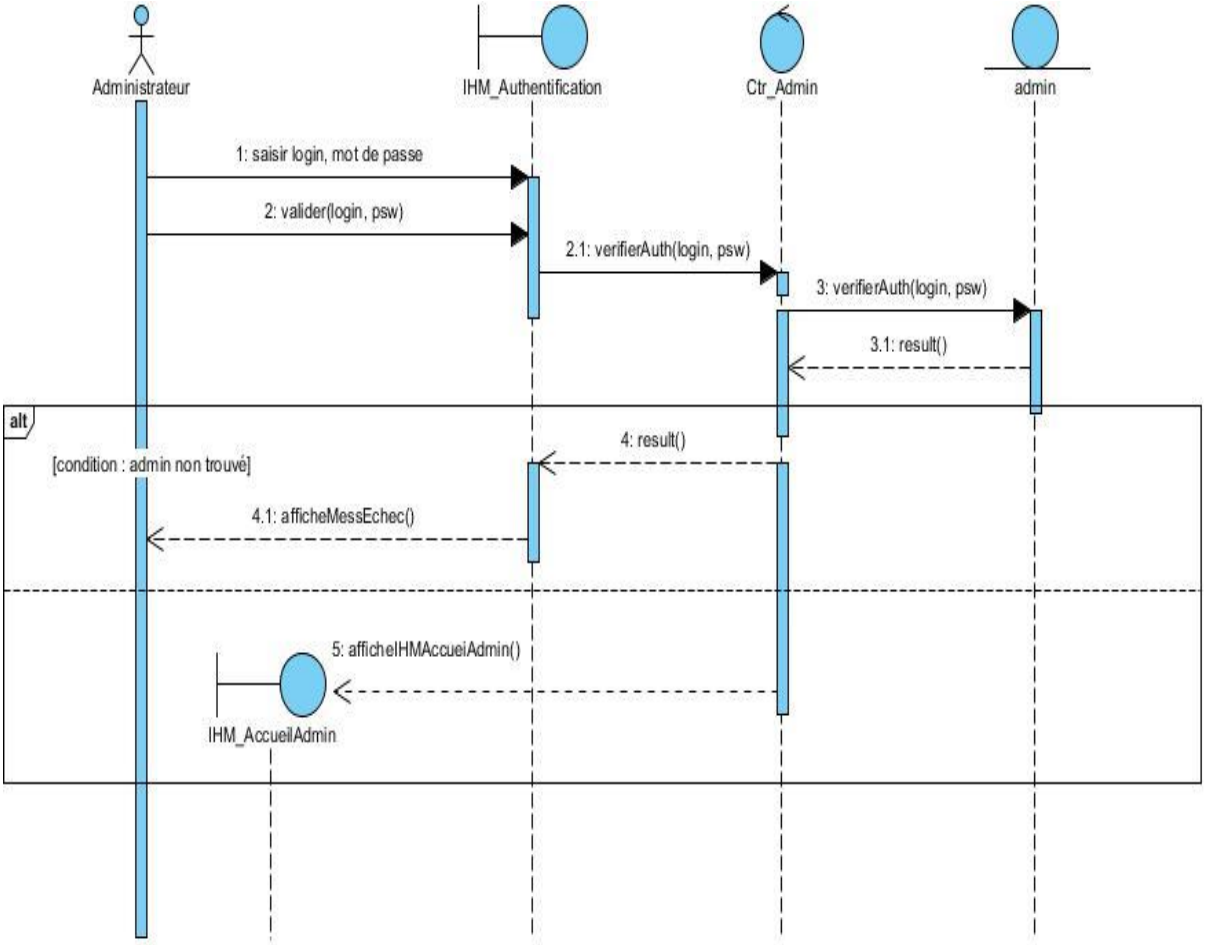


Figure III.2. Diagramme d'interaction système du cas « Authentifier ».

III.2.2.2. Diagramme d'interaction système du cas « Afficher la liste des clients » pour l'administrateur

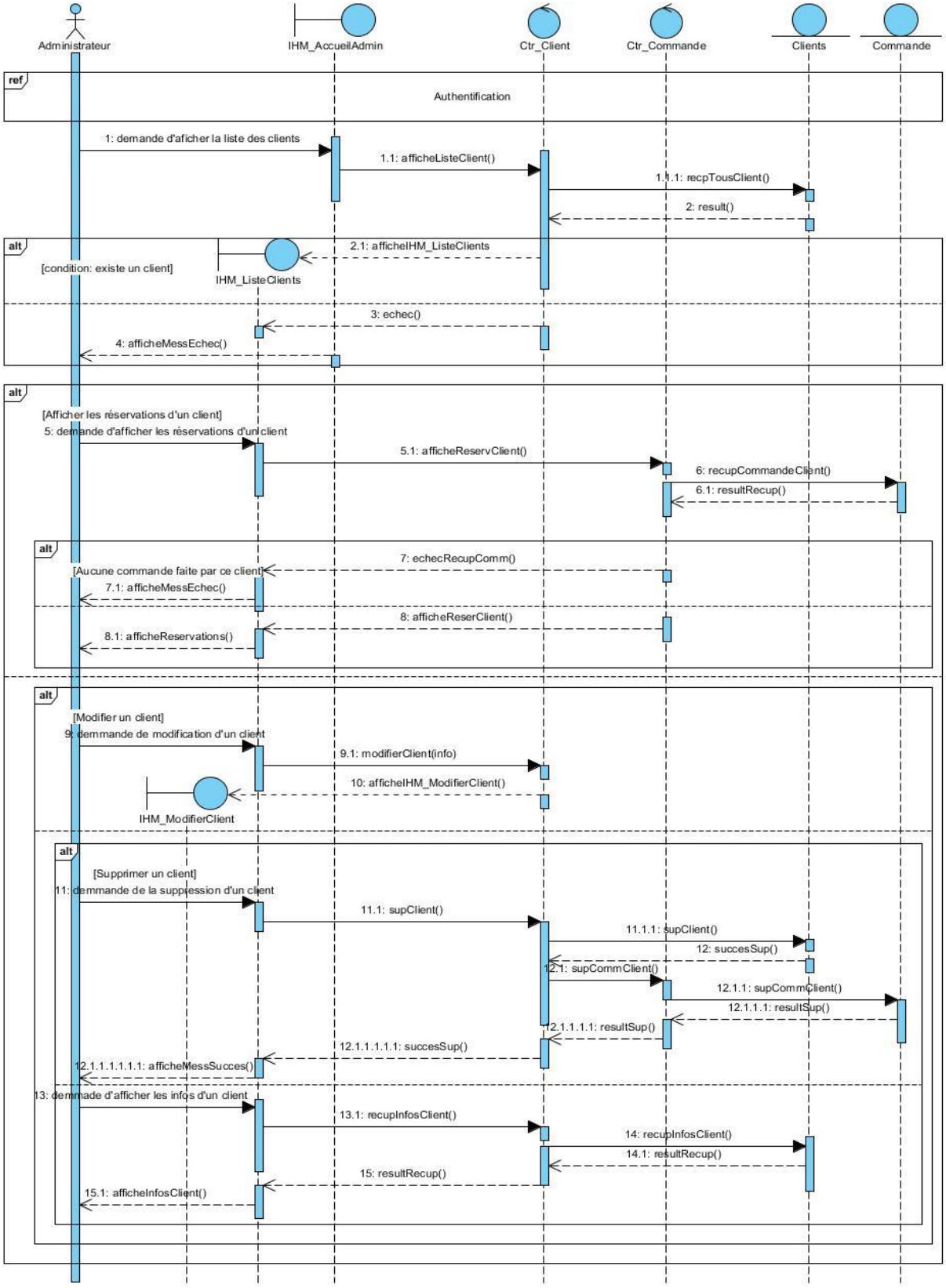


Figure III.3. Diagramme d'interaction système du cas « Afficher la liste des clients » pour l'administrateur.

III.2.2.3. Diagramme d'interaction système du cas « Ajouter un produit » pour l'administrateur

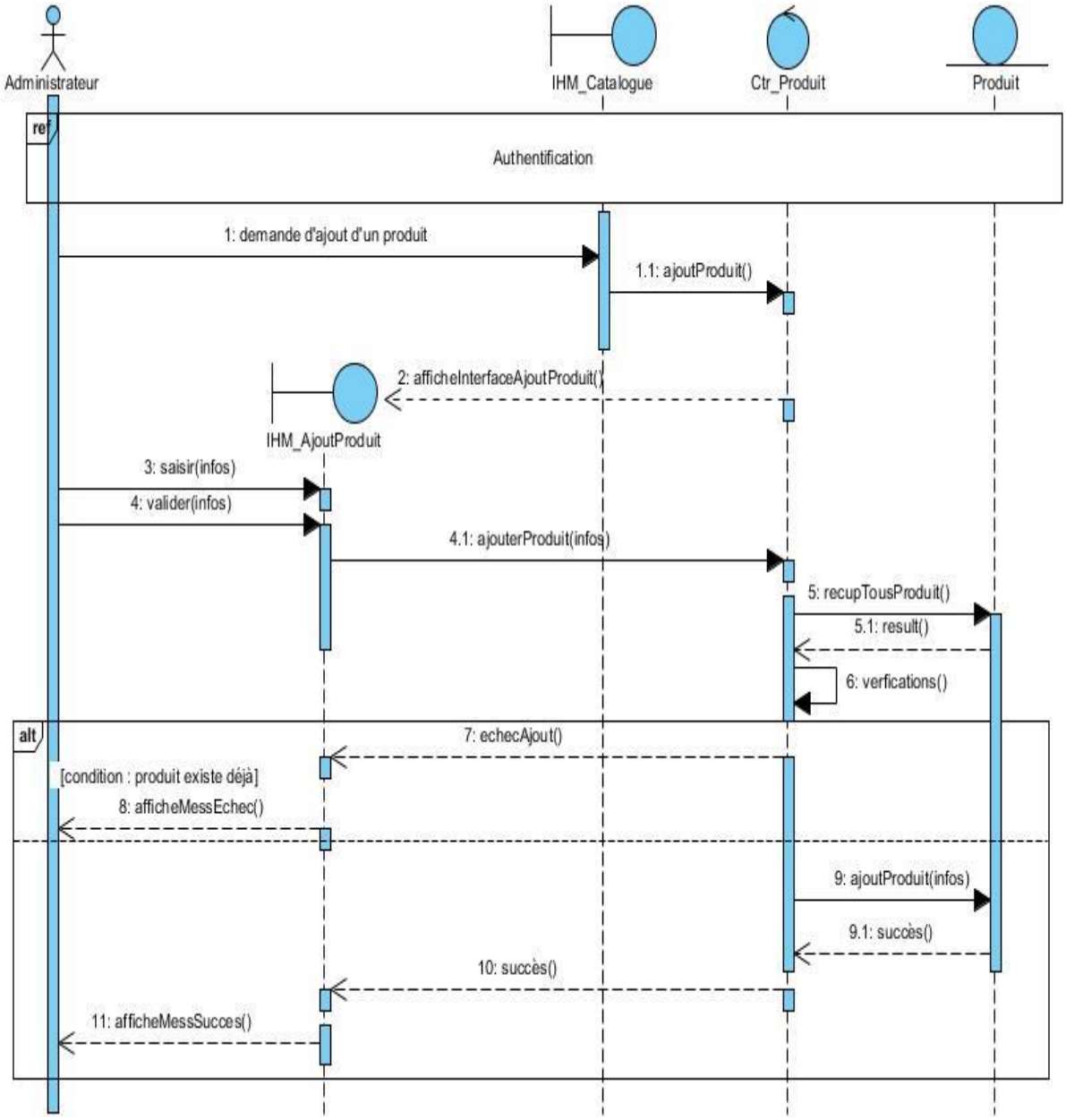


Figure III.4. Diagramme d'interaction système du cas « Ajouter un produit » pour l'administrateur.

III.2.2.4. Diagramme d'interaction système du cas « Modifier un produit » pour l'administrateur

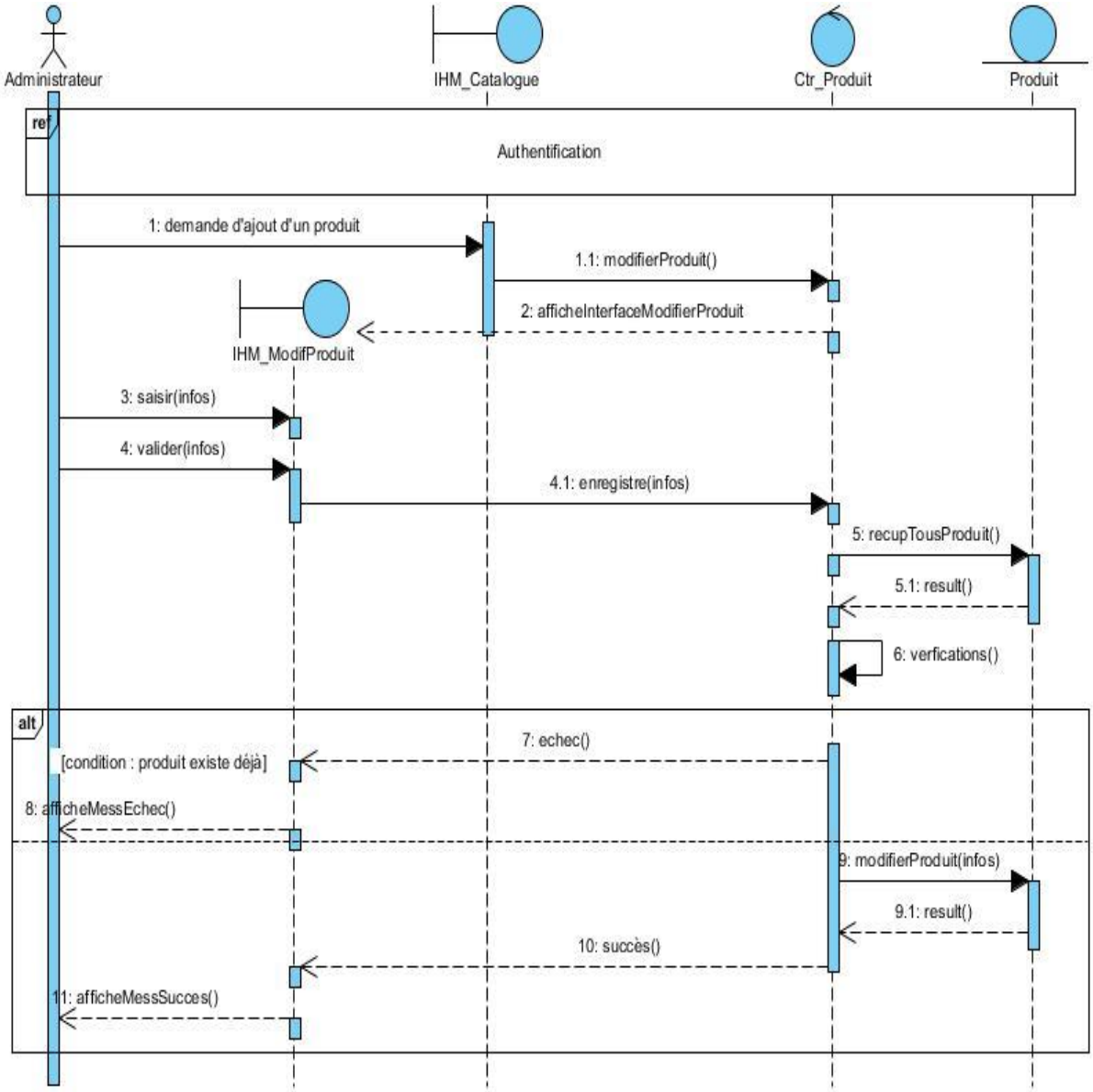


Figure III.5. Diagramme d'interaction système du cas « Modifier un produit » pour l'administrateur.

III.2.2.5. Diagramme d'interaction système du cas « Supprimer un produit » pour l'administrateur

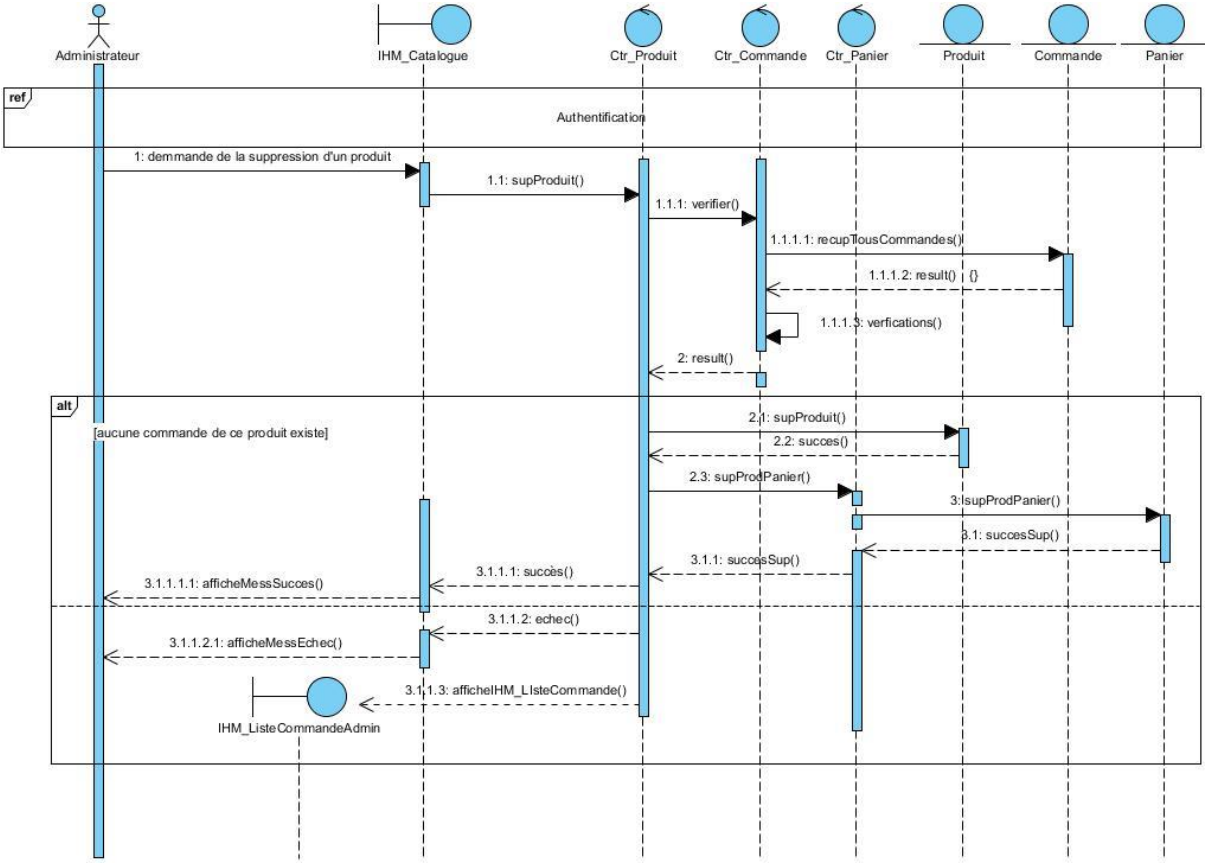


Figure III.6. Diagramme d'interaction système du cas « Supprimer un produit » pour l'administrateur.

III.2.2.6 Diagramme d'interaction système du cas « Annuler une commande » pour l'administrateur

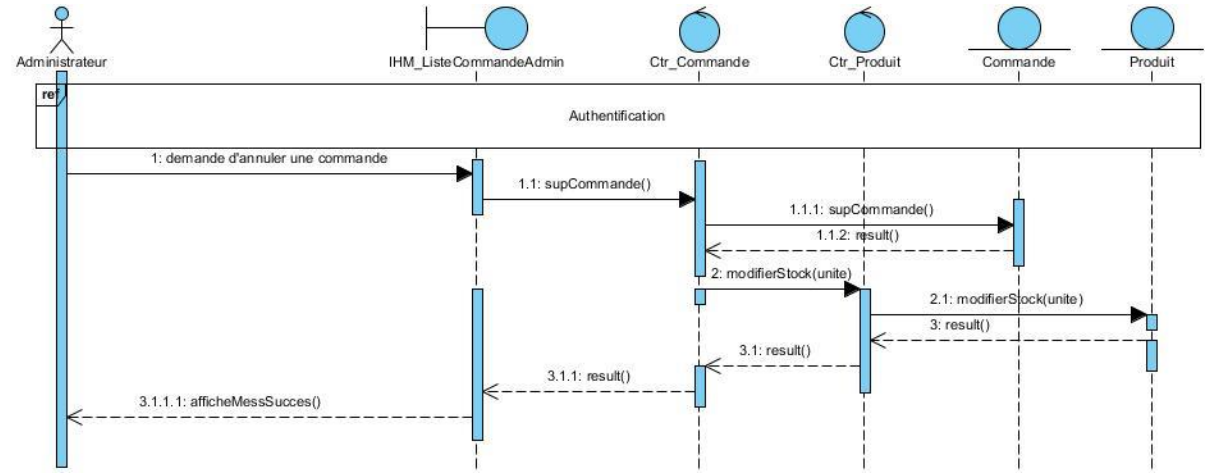


Figure III.7. Diagramme d'interaction système du cas « Annuler une commande » pour l'administrateur.

III.2.3. Diagrammes d'interactions système pour le Client

III.2.3.1. Diagramme d'interaction système du cas « Consulter le catalogue » pour le client

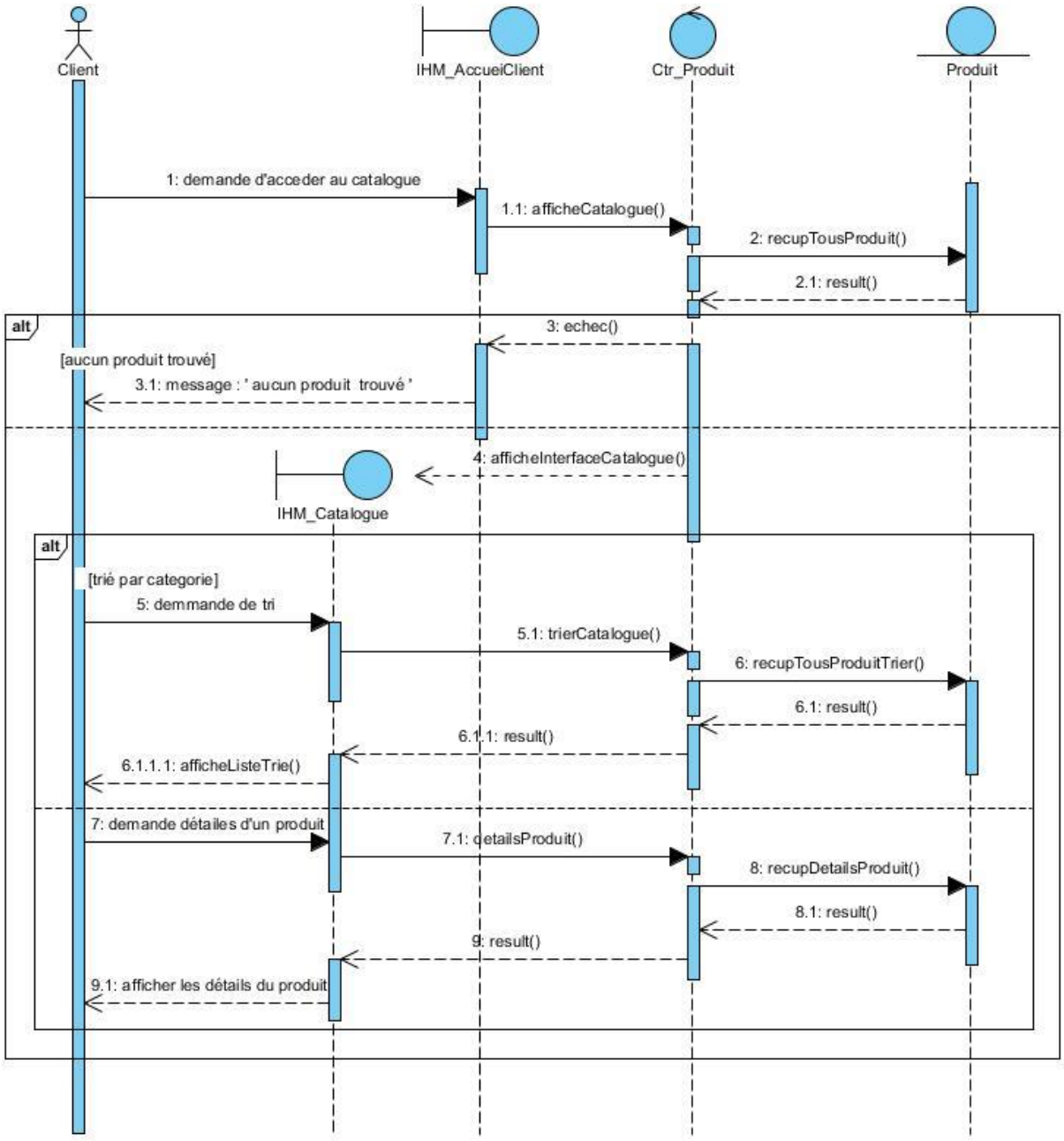


Figure III.8. Diagramme d'interaction système du cas « Consulter le catalogue » pour le client.

III.2.3.2. Diagramme d'interaction système du cas « Faire une commande » pour le client

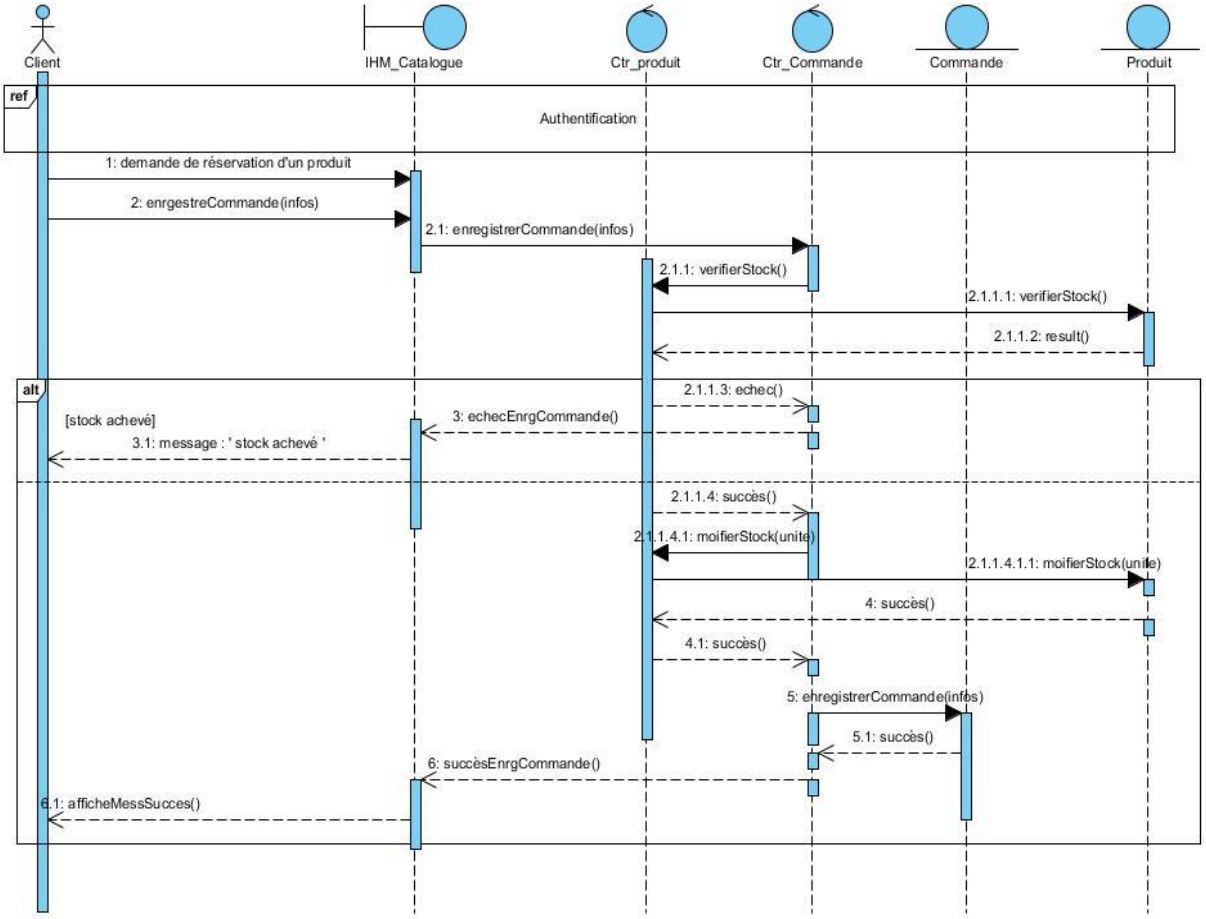


Figure III.9. Diagramme d'interaction système du cas « Faire une commande » pour le client.

III.2.3.3. Diagramme d'interaction système du cas « Annuler une commande » pour le client

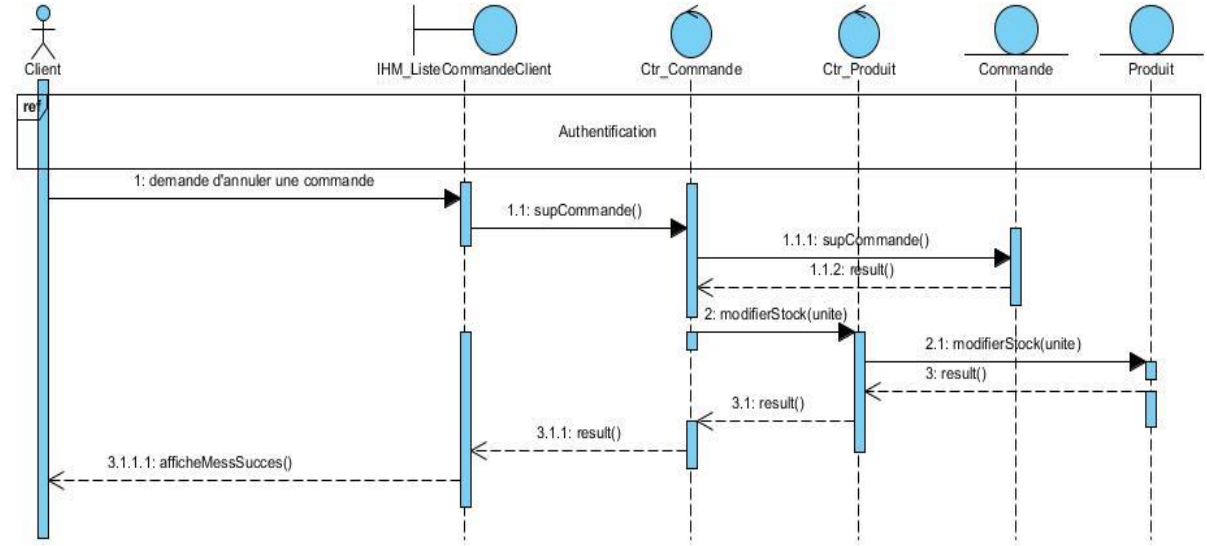


Figure III.10. Diagramme d'interaction système du cas « Annuler une commande » pour le client.

III.2.3.4. Diagramme d'interaction système du cas « Afficher la liste de ses commandes » pour le client

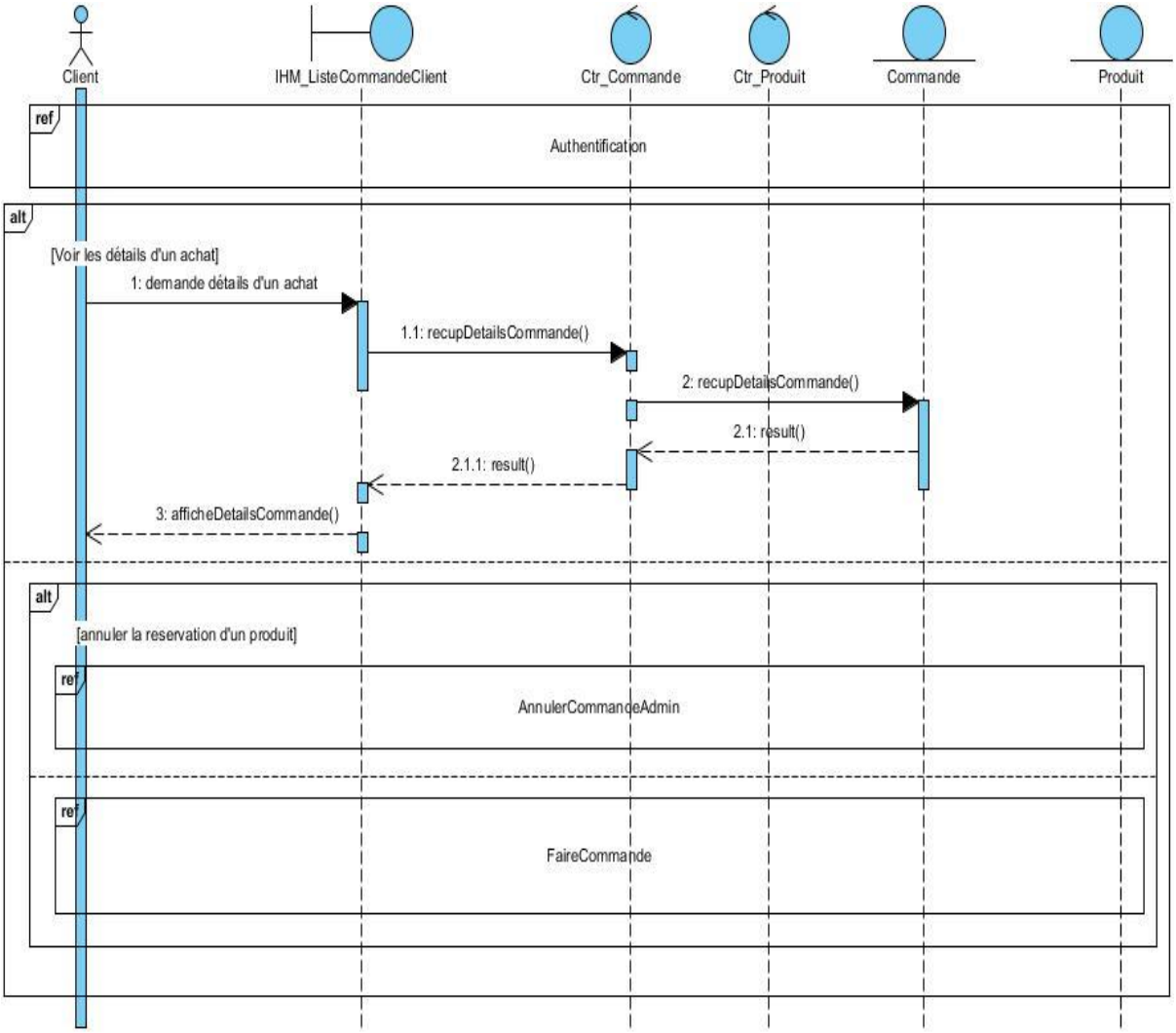


Figure III.11. Diagramme d'interaction système du cas « Afficher la liste de ses commandes » pour le client.

III.2.3.5. Diagramme d'interaction système du cas « Recherche un produit » pour le client

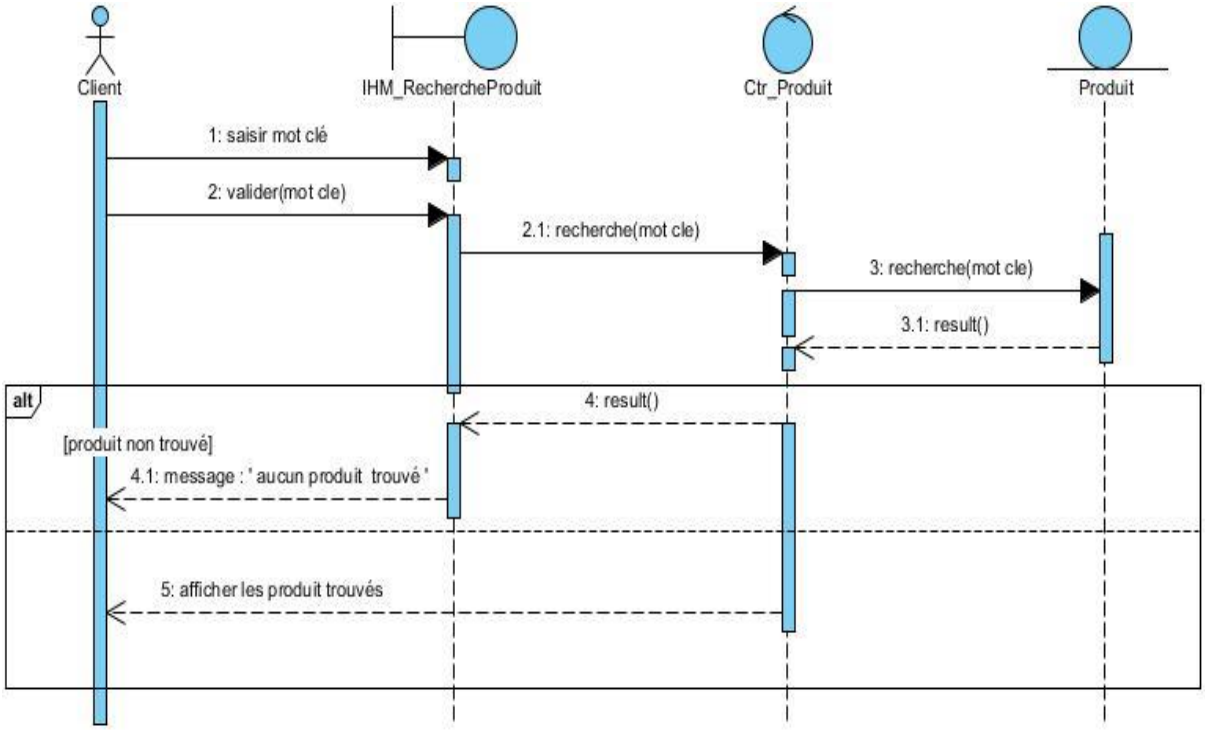


Figure III.12. Diagramme d'interaction système du cas « Recherche un produit » pour le client.

III.2.3.6. Diagramme d'interaction système du cas « Gérer son compte » pour le client

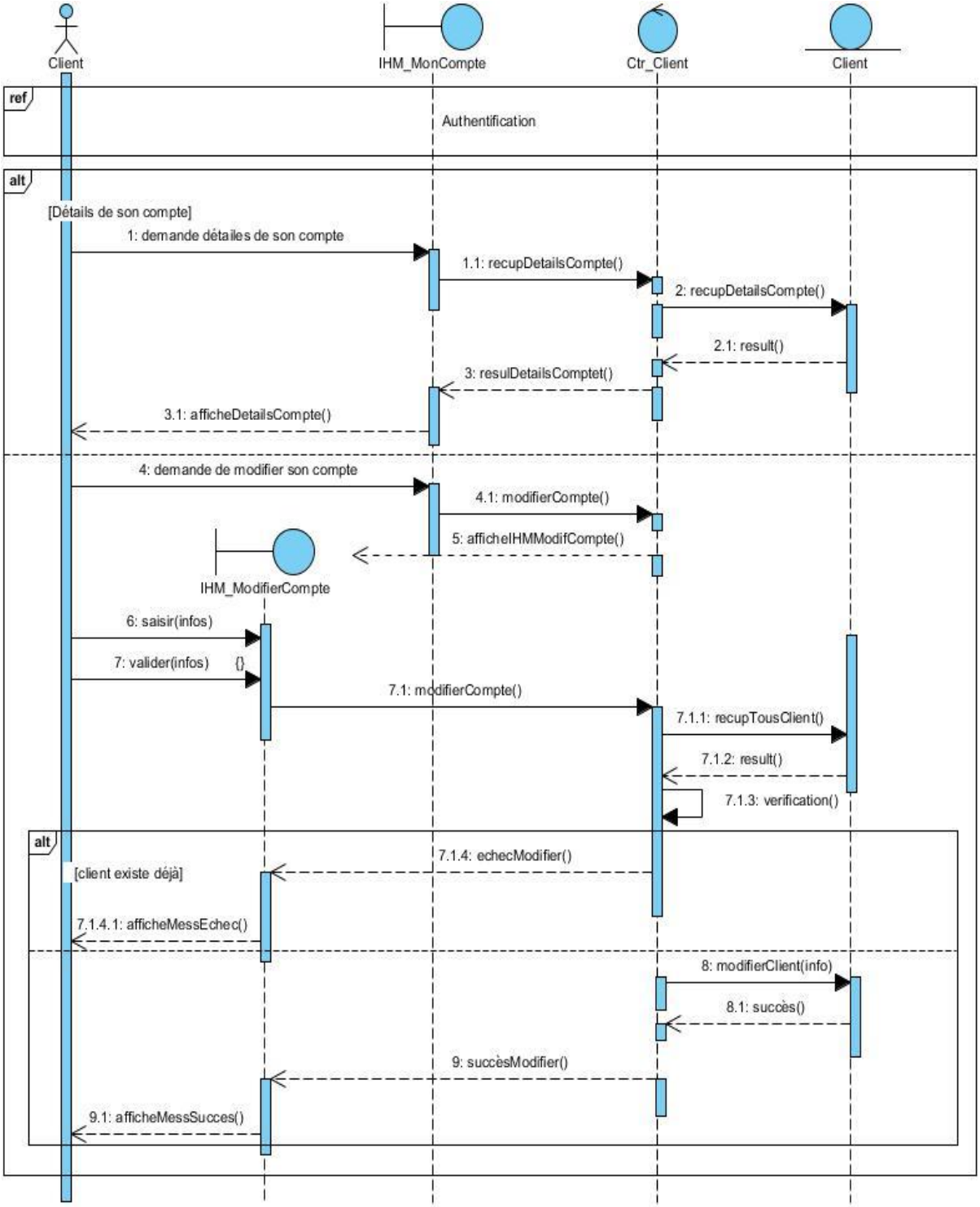


Figure III.13. Diagramme d'interaction système du cas « Gérer son compte » pour le client.

III.2.3.7. Diagramme d'interaction système du cas « Gérer son panier » pour le client

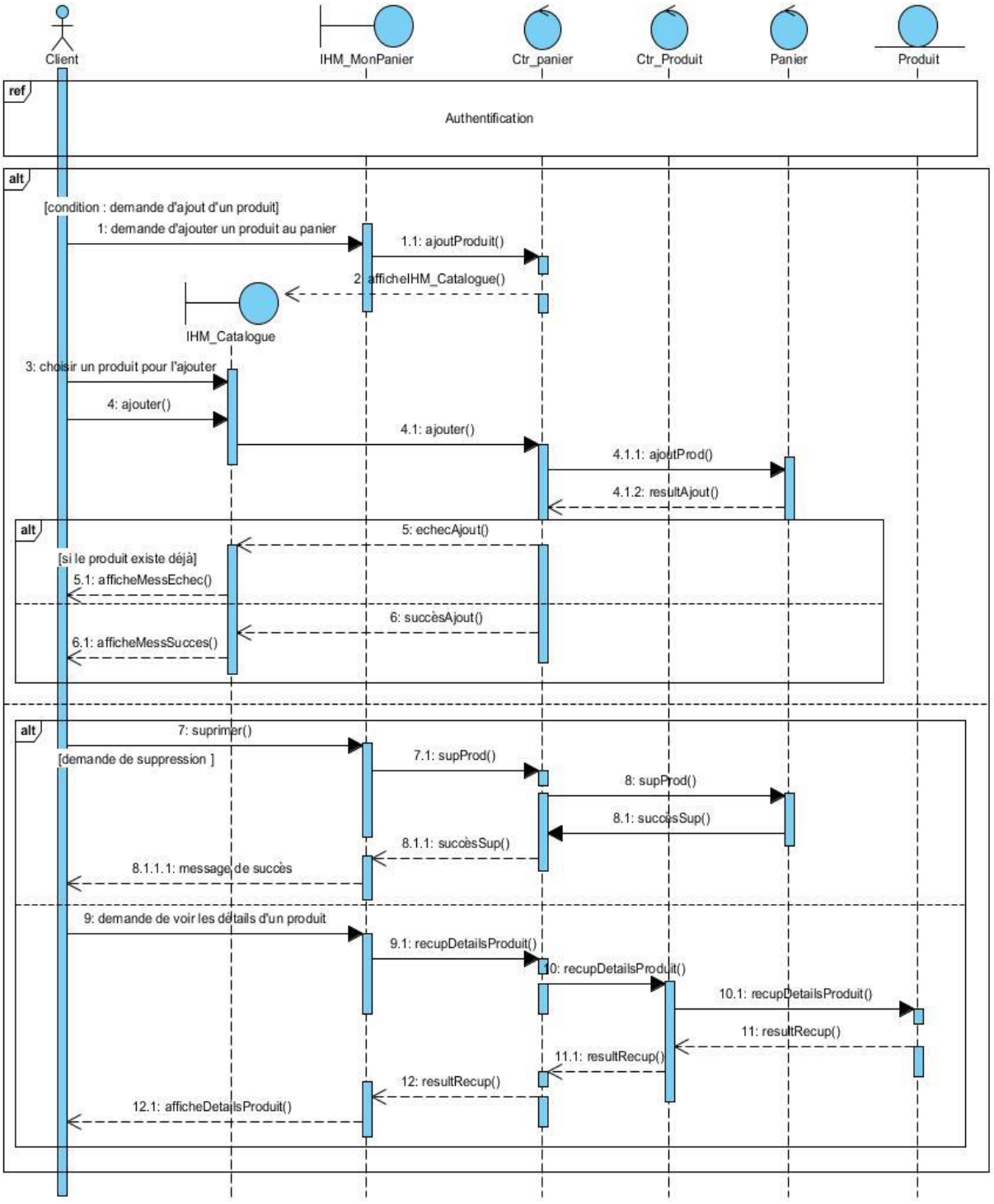


Figure III.14. Diagramme d'interaction système du cas « Gérer son panier » pour le client.

III.3. Diagramme de classes de conception

L'objectif de cette étape est de produire le diagramme de classes qui servira pour l'implémentation.

Le diagramme de classes de conception modélise trois types de classes : les dialogues, les contrôles et les entités ainsi que leurs relations. Il faut maintenant le compléter en précisant les opérations privées des différentes classes. Il faut prendre en compte les choix techniques, comme le choix du langage de programmation, le choix des différentes bibliothèques utilisées (notamment pour l'implémentation de l'interface graphique), etc.

Pour une classe, le couplage est la mesure de la quantité d'autres classes auxquelles elle est connectée par des associations, des relations de dépendance, etc. Durant toute l'élaboration du diagramme de classes de conception, il faut veiller à conserver un couplage faible pour obtenir une application plus évolutive et plus facile à maintenir. L'utilisation des design patterns est fortement conseillée lors de l'élaboration du diagramme de classes de conception **[23]**.

Nous allons présenter les diagrammes de classes de conception de notre système pour L’administrateur et le client.

III.3.1. Diagramme de classes de conception pour l'administrateur

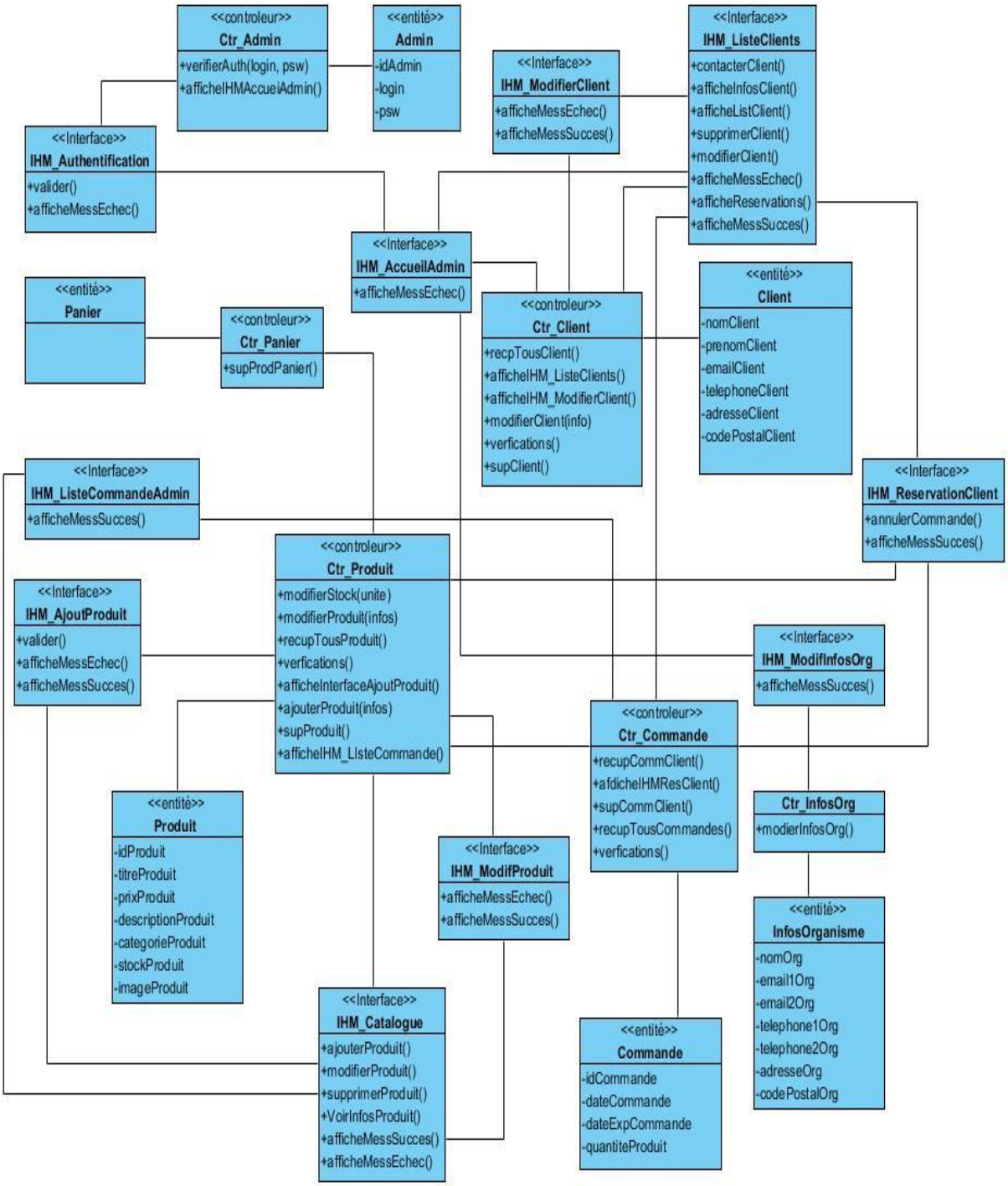


Figure III.15. Diagramme des classes de conception pour l'administrateur.

III.3.2. Diagramme de classes de conception pour le client

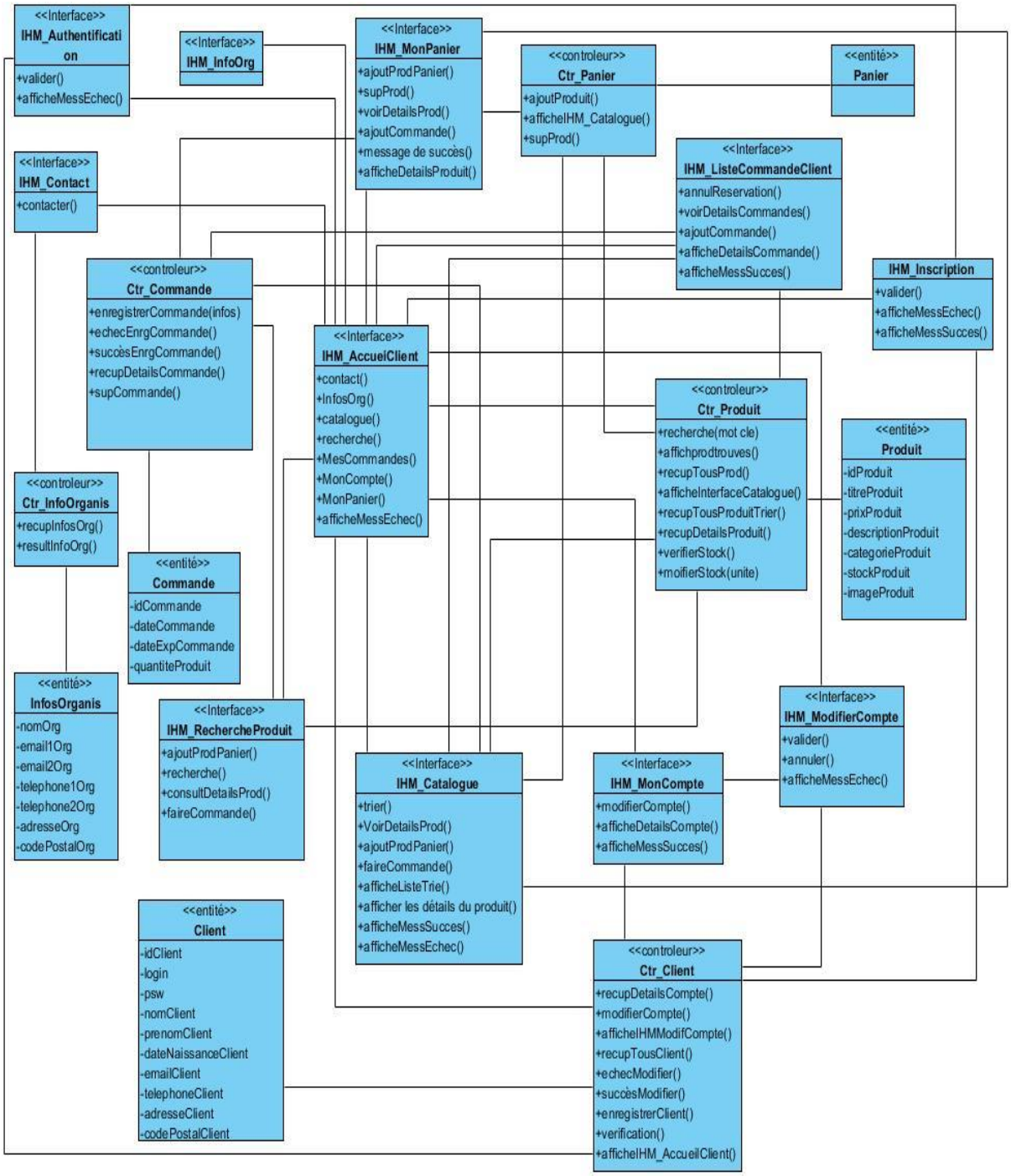


Figure III.16. Diagramme des classes de conception pour le client.

III.4. Modèle relationnel

Le concepteur d'une base de données relationnelle doit élaborer un schéma relationnel de la base de données.

Cette activité consiste à définir toutes les relations de la base de données et leurs attributs.

III.4.1. Règles de passage au modèle relationnel

Les règles utilisées pour le passage du diagramme de classes de notre application web au modèle Relationnel sont les suivantes **[27]** :

- Chaque classe devient une relation. Les attributs de la classe deviennent des attributs de la relation. Si la classe possède un identifiant, il devient la clé primaire de la relation, sinon, il faut ajouter une clé primaire arbitraire
- Pour représenter une association 1 vers 1 entre deux relations, la clé primaire de l'une des relations doit figurer comme clé étrangère dans l'autre relation.
- Pour représenter une association 1 vers plusieurs, on procède comme pour une association 1 vers 1, excepté que c'est forcément la relation du côté plusieurs qui reçoit comme clé étrangère la clé primaire de la relation du côté 1.
- Pour représenter une association du type plusieurs vers plusieurs, il faut introduire une nouvelle relation dont les attributs sont les clés primaires des relations en association et dont la clé primaire est la concaténation de ces deux attributs.
- Le cas est proche de celui d'une association plusieurs vers plusieurs, les attributs de la classe-association étant ajoutés à la troisième relation qui représente, cette fois-ci, la classe-association elle-même.
- Il faut mettre un # après chaque clé étrangère et un tiré en dessous du chaque clé primaire.

Après avoir appliqué les règles de passage cité précédemment sur le modèle du domaine de notre système, nous avons abouti au schéma relationnel de la base de données suivant:

Administrateur (idAdmin, loginAdmin, pswAdmin).

Client (idClient, loginClient, pswClient, nomClient, prenomClient,dateNaissanceClient, adresseClient, codePostalClient).

InfosOraganisme (nomOrganisme, email1Organisme, email2Organisme, telephone1Organisme, telephone2Organisme, adresseOrganisme, codePostalOrganisme).

Produit (idProduit, titreProduit, prixProduit, descriptionProduit, categorieProduit, stockProduit, imageProduit).

Commande (idCommande, idClient#, idProduit#, dateCommande, dateExpCommande, quantiteProduit, prixTotal).

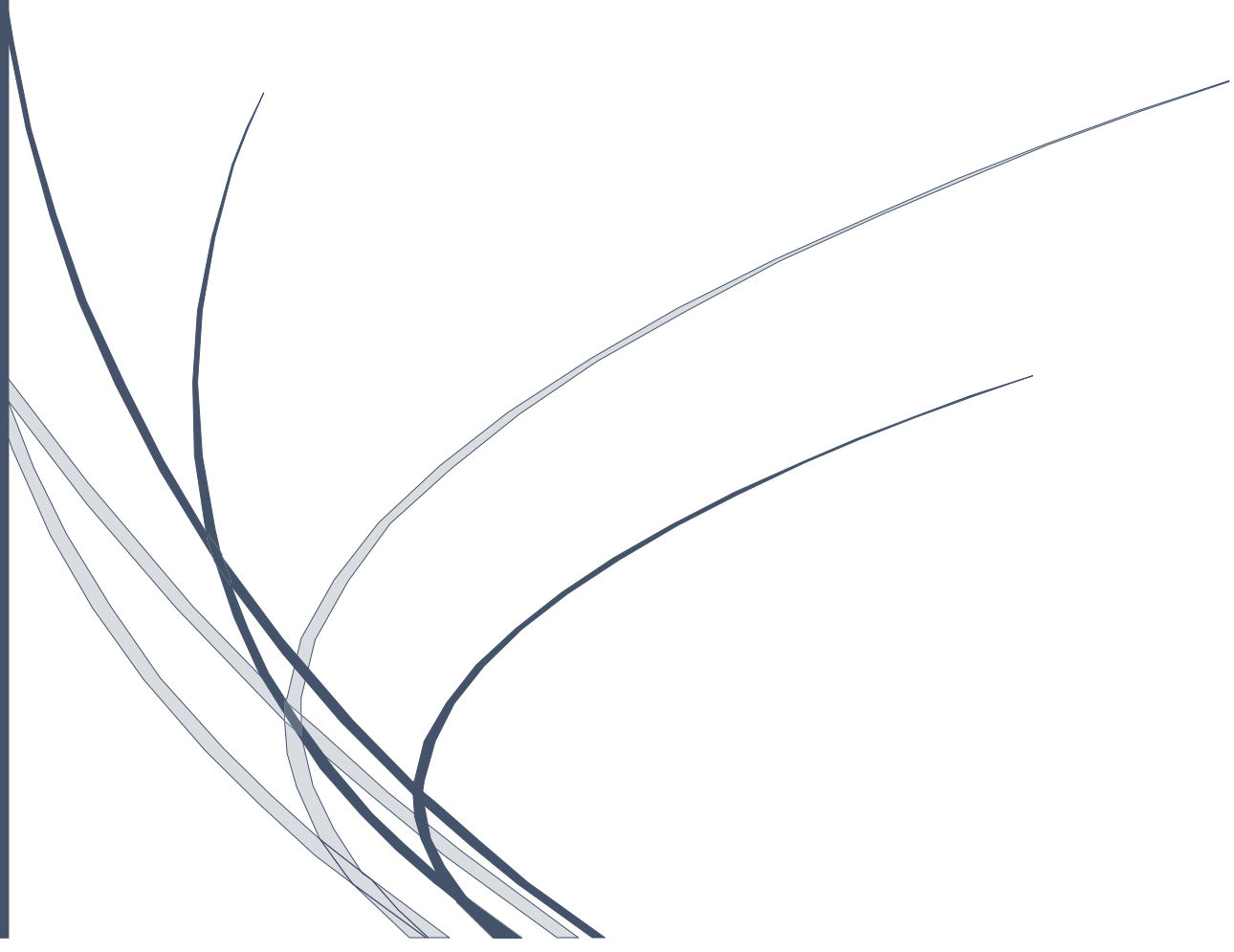
Panier (idPanier, idClient#, idProduit#).

Conclusion

Cette phase de conception avait pour intérêt de présenter les différentes étapes de conception de l'application tout en évoluant dans le niveau de détail, et doit par conséquent aboutir immédiatement à l'implémentation avec une vision claire des aspects fonctionnels ainsi qu'organisationnels de l'application.

Maintenant, notre application est prête à être codée. Dans le chapitre suivant, nous allons nous intéresser à l'implémentation de notre système en se basant sur la conception détaillée de ce chapitre.

Chapitre IV Implémentation et tests



Chapitre IV Implémentation et tests

Introduction

Dans ce chapitre, nous allons nous intéresser à la description des environnements matériels et logiciels qui nous ont permis de réaliser notre projet. Nous passons ensuite à la phase test d'intégration dans laquelle nous allons présenter les différentes interfaces de notre application.

IV.1. Environnement du développement de l'application

IV.1.1. Environnement matériel

Deux PCs DELL avec :

Processeur : Intel(R) Core(TM) I7 3537U CPU @ 2,00GHz 2,50GHz.

Mémoire(RAM) : 8Go.

Disque dur : 1To.

IV.1.2. Environnement logiciel

IV.1.2.1. Android Studio :

Android Studio est l'environnement de développement intégré (IDE) officiel pour le développement d'applications Android, basé sur IntelliJ IDEA.

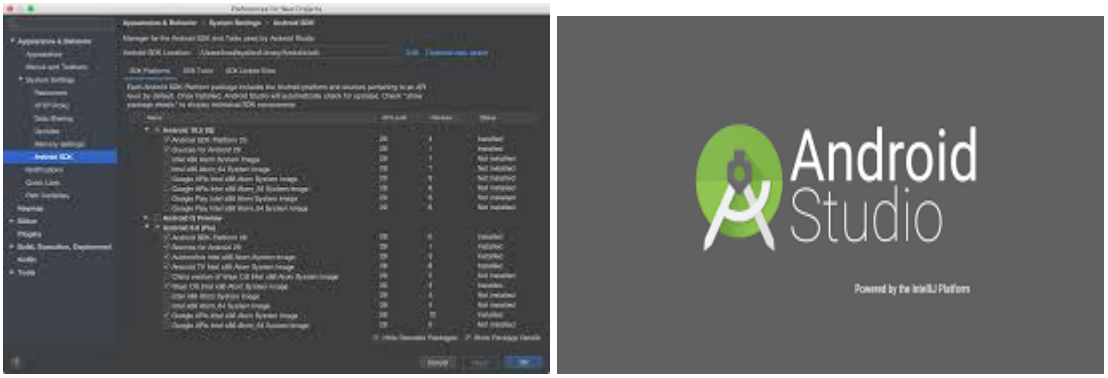


Figure I0.1. Android Studio.

IV.1.2.2. JDK

Le JDK est un logiciel qui contient une variété d'outils et d'utilitaires qui permettent de développer, empaqueter, surveiller et déployer des applications qui se construisent pour n'importe quelle plate-forme Java standard (java SE, java ME, java EE).

IV.1.2.3. IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré (IDE) pour les langages JVM conçu pour maximiser la productivité des développeurs. Il effectue les tâches routinières et répétitives en fournissant une complétion de code intelligent et une analyse de code statique. **[28]**

IV.1.2.4. PgAdmin 4

Postgres pgAdmin 4 permet de créer tous sorts d'objets du serveur de bases de données PostgreSQL. Ces objets peuvent être des bases de données (BDD), des schémas, des tables, des utilisateurs… Cet outil permet également d'exécuter des requêtes SQL. **[29]**

IV.1.2.5. Postgres

PostgreSQL est un puissant système de base de données relationnelle objet open source qui utilise et étend le langage SQL combiné à de nombreuses fonctionnalités qui stockent et mettent à l'échelle en toute sécurité les charges de travail de données les plus complexes. Les origines de PostgreSQL remontent à 1986 dans le cadre du projet POSTGRES de l'Université de Californie à Berkeley et comptent plus de 30 ans de développement actif sur la plate-forme principale. **[30]**

Figure I0.2. PostgreSQL.

IV.1.3. Langages de programmation

IV.1.3.1. Kotlin

Kotlin est un langage de programmation à usage général, gratuit, open source, à typage statique, initialement conçu pour la JVM (Java Virtual Machine) et Android, qui combine des fonctionnalités de programmation orientées objet et fonctionnelles. Il est axé sur l'interopérabilité, la sécurité, la clarté et le support de l'outillage. Kotlin est né chez JetBrains, la société derrière IntelliJ IDEA, en 2010, il est open source depuis 2012. L'équipe Kotlin compte actuellement plus de 90 membres à plein temps de JetBrains, et le projet Kotlin sur GitHub compte plus de 300 contributeurs. JetBrains utilise Kotlin dans plusieurs de ses produits, y compris son produit phare IntelliJ IDEA. **[31]**

Figure I0.3. Kotlin.

IV.1.3.2. PHP

PHP est un langage open source côté serveur utilisé pour créer des pages Web dynamiques. Il peut être intégré au HTML. PHP est généralement utilisé en conjonction avec une base de données.

Figure I0.4. PHP.

IV.1.3.3. SQL

Le langage SQL (Structured Query Language) peut être considéré comme le langage d'accès normalisé aux bases de données. Il est aujourd'hui supporté par la plupart des produits commerciaux que ce soit par les systèmes de gestion de bases de données micro tel que Access ou par les produits plus professionnels tels qu'Oracle. Il a fait l'objet de plusieurs normes ANSI/ISO dont la plus répandue aujourd'hui est la norme SQL2 qui a été définie en 1992. **[32]**

IV.1.3.4. SQLite

SQLite est une base de données SQL Open Source qui stocke les données dans un fichier texte sur un appareil. Android est livré avec une implémentation de base de données SQLite intégrée. SQLite prend en charge toutes les fonctionnalités de la base de données relationnelle. Pour accéder à cette base de données, vous n'avez pas besoin d'établir de connexion.

IV.1.3.5. XML

XML est un langage de balisage flexible pour les documents électroniques structurés. Le langage XML (Extensible Markup Language) est un langage de programmation couramment utilisé par les services d'échange de données (comme les flux de blog) pour envoyer des informations entre des systèmes autrement incompatibles. Il est lisible par les humains et les ordinateurs et est basé sur SGML (Standard Generalized Markup Language), une norme internationale pour les documents électroniques. **[33]**

IV.1.3.6. JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) est un format d'échange de données léger, il est facile pour les humains de lire et d'écrire. Il est facile pour les machines d'analyser et de générer. Il est basé sur un sous-ensemble de la norme du langage de programmation JavaScript ECMA-262 3e édition, décembre 1999. JSON est un format de texte complètement indépendant du langage, mais qui utilise des conventions familières aux programmeurs de la famille de

langages C, notamment C, C ++, C #, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal. **[34]**

IV.1.4. Bibliothèques (APIs)

IV.1.4.1. Anko

Anko est une bibliothèque Kotlin qui rend le développement d'applications Android plus rapide et plus facile. Il rend le code propre et facile à lire, et permet d'oublier les aspérités du SDK Android pour Java.Anko se compose de plusieurs parties:

- Anko Commons : une bibliothèque légère remplie d'aides pour les intentions, les dialogues, la journalisation, etc.
- Anko Layouts : un moyen rapide et sûr d'écrire des mises en page dynamiques Android;
- Anko SQLite : une collection de requêtes DSL et d'analyseurs pour Android SQLite.
- Anko Coroutines : utilitaires basés sur la bibliothèque kotlinx.coroutines.

IV.1.4.2. API RESTful

RESTful est un style architectural pour une interface de programme d'application (API) qui utilise des requêtes HTTP pour accéder et utiliser des données. Ces données peuvent être utilisées pour les types de données GET, PUT, POST et DELETE, qui se réfèrent à la lecture, la mise à jour, la création et la suppression d'opérations concernant les ressources.

IV.1.4.3. Retrofit :

Retrofit est un client REST pour Java et Android. Cela rend relativement facile la récupération et le téléchargement de JSON (ou d'autres données structurées) via un service Web basé sur REST.

IV.1.4.4. Room

Room est une bibliothèque de persistance faisant partie des composants d'architecture Android. Elle permet de facilite le travail en réduisant la quantité de code standard et en vérifiant les requêtes SQL au moment de la compilation.

IV.1.4.5. RXjava

RxJava est une implémentation open-source compatible JVM de la bibliothèque ReactiveX conçue pour aider à travailler avec des flux de données asynchrones dans un style de programmation réactif, et sans avoir à écrire une tonne de rappels. **[35]**

L'utilisation de RxJava dans les projets Android présente de nombreux avantages, mais certains des plus importants sont :

- **Simplifier le code.**
- **Un flux de travail cohérant.**
- **Simplifier le multithreading :** Les applications mobiles modernes doivent pouvoir effectuer plusieurs tâches à la fois, mais en tant qu'environnement à un seul thread, Android n'est pas exactement un multitâche naturel. Android fournit plusieurs outils

intégrés pour créer des threads supplémentaires. Rxjava donne l'accès à certaines opérations comme subscribeOn et observeOn,ce qui facilite beaucoup la création et la gestion de threads supplémentaires.

- Transformations de données complexes.

IV.1.5. L'outil de conception

IV.1.5.1. Visual Paradigm for UML Enterprise Edition

Il s'agit d'un outil de conception logicielle adapté aux projets logiciels agiles. Il prend en charge UML, BPMN, DFD, SysML. Il prend également en charge les cas d'utilisation, le wireframe, l'ingénierie de code, etc. **[36]**

Figure I0.5. PostgreSQL.

IV.2. Test d'intégration

IV.2.1. Scénario d'exécution

Nous présenterons dans ce qui suit quelques interfaces de notre application.

IV.2.1.1 Coté administrateur

a. L'interface « Accueil » pour l'administrateur

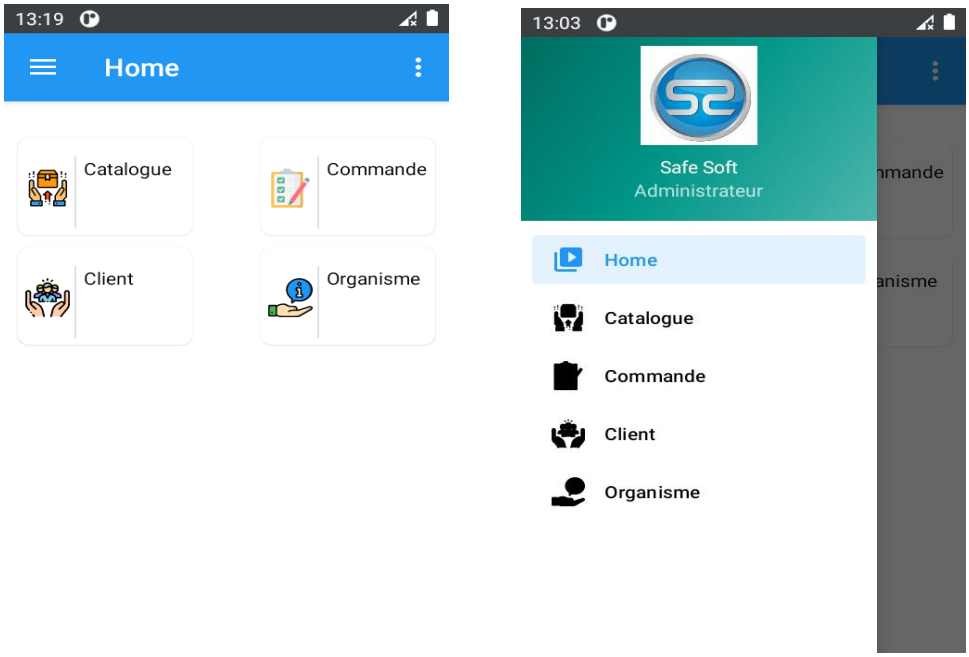


Figure I0.6. L'interface « Accueil » pour l'administrateur.

b. L'interface « Ajout produit » pour l'administrateur

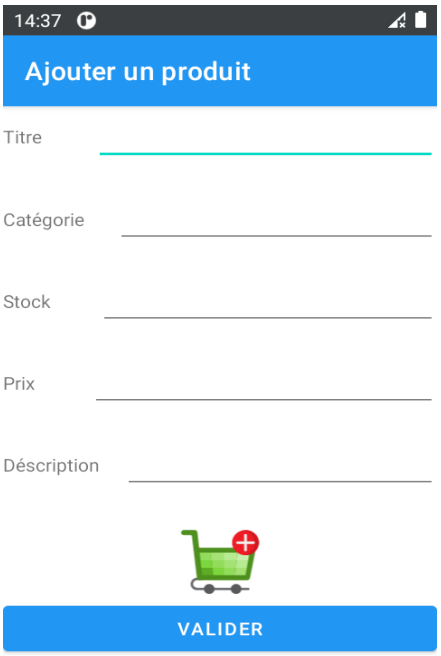


Figure I0.7. L'interface « Ajout produit » pour l'administrateur.

c. L'interface « Détails produit » pour l'administrateur

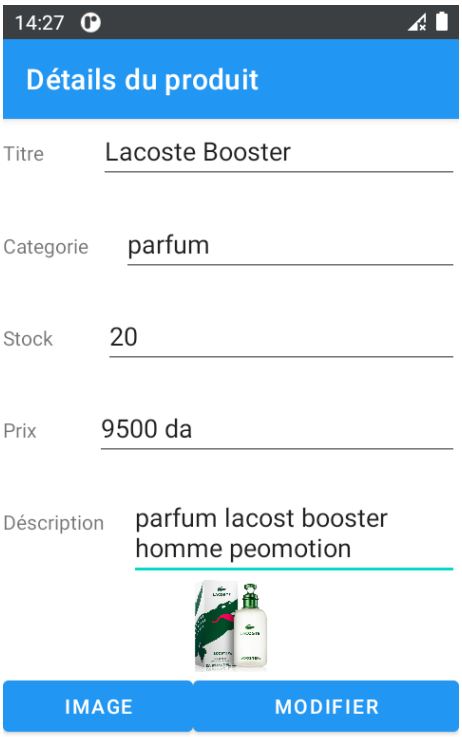


Figure I0.8. L'interface « Détails produit » pour l'administrateur.

d. L'interface « Catalogue » pour l'administrateur

Figure I0.9. L'interface « Catalogue » pour l'administrateur.

IV2.1.2. Coté client

a. L'interface « Inscription » pour le client

b. L'interface « Authentification » pour le client

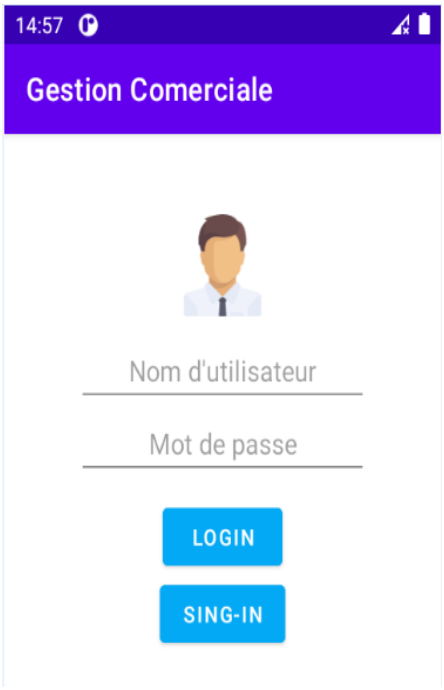


Figure I0.10. *L'interface « Authentification » pour le client.*

c. L'interface « Catalogue » pour le client

Figure I0.11. L'interface « Catalogue » pour le client.

d. L'interface « Contacts » pour le client

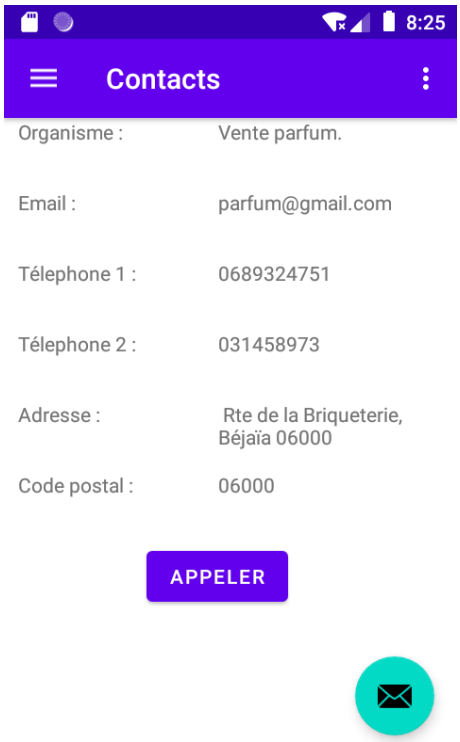


Figure I0.12. L'interface « Contacts » pour le client.

L'interface « détails du produit » pour le client

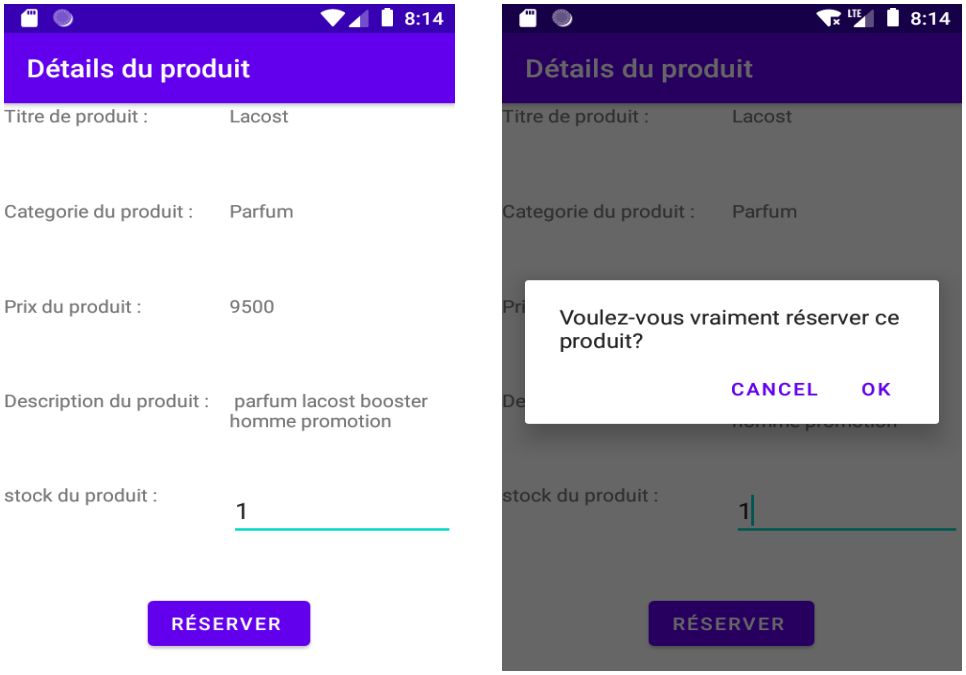


Figure I0.13. *L'interface « Détails produits » pour le client.*

e. L'interface « Mon compte » pour le client

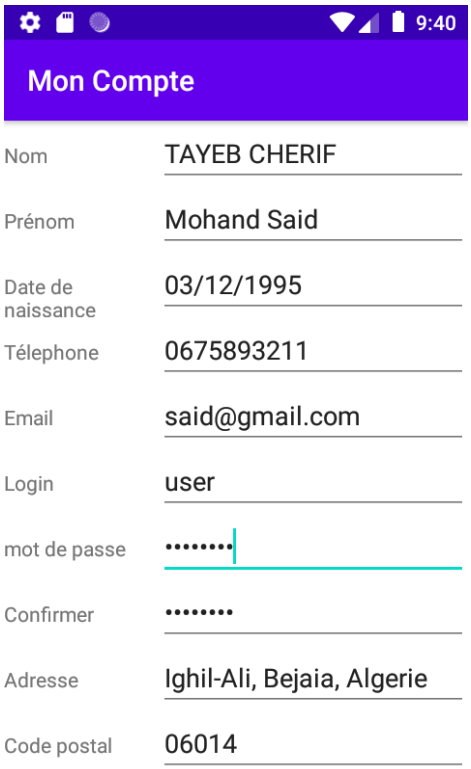


Figure I0.14. *L'interface « Mon compte » pour le client.*

f. L'interface « Modifier compte » pour le client

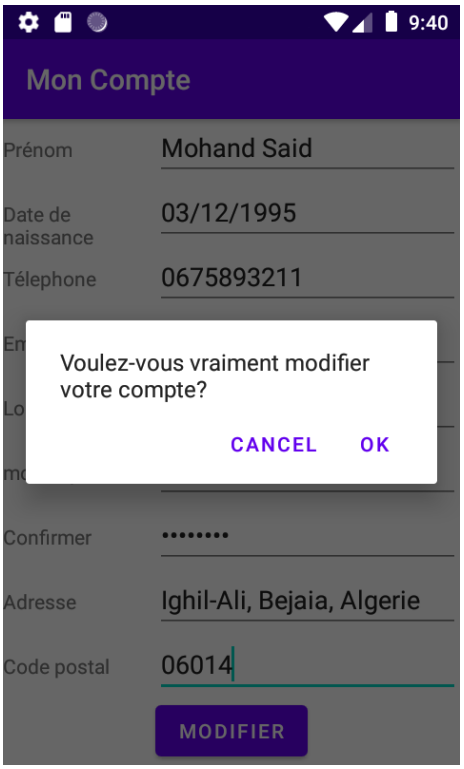


Figure I0.15. L'interface « Modifier compte » pour le client.

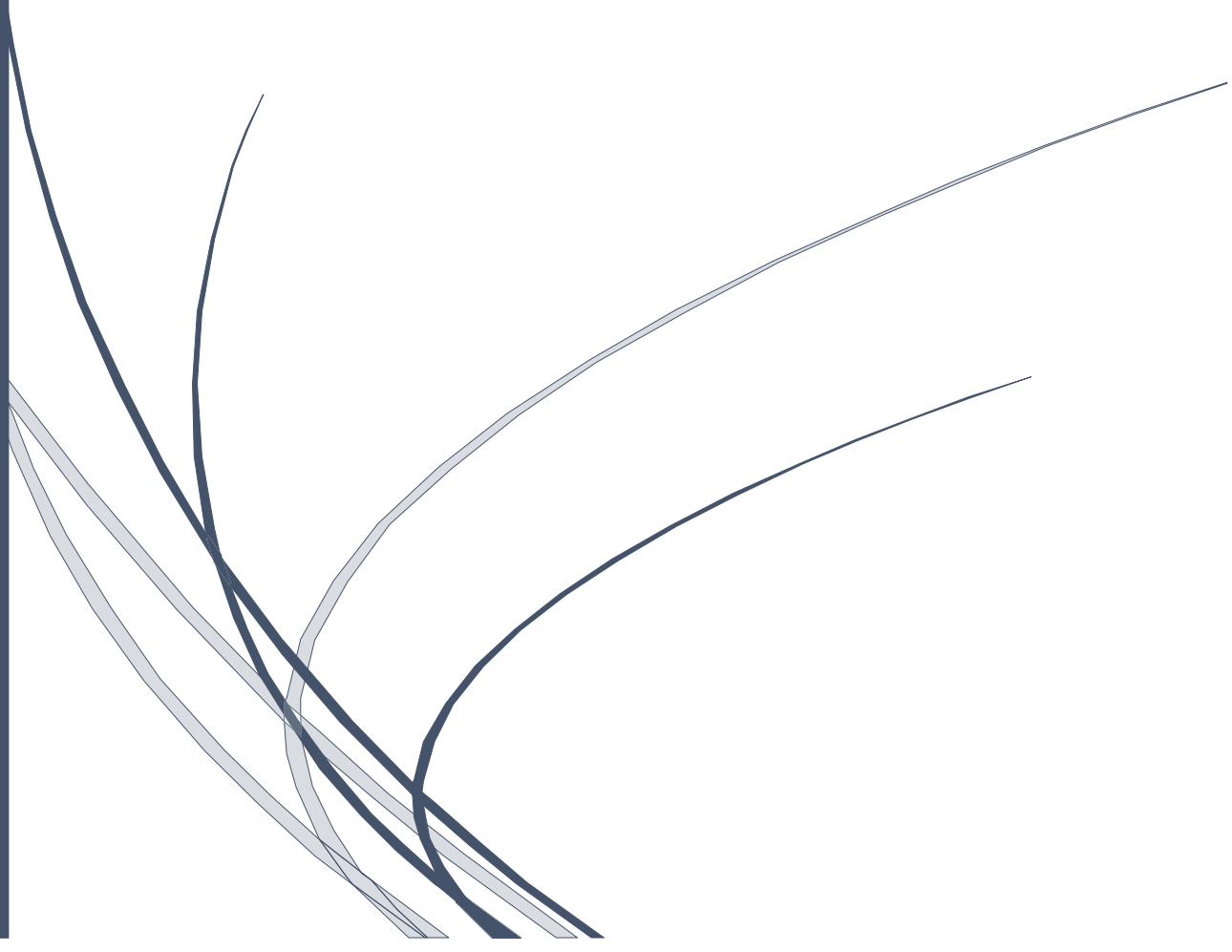
g. L'interface « Mon Panier » pour le client

Figure I0.16. L'interface « Mon panier » pour le client.

Conclusion

Dans ce chapitre nous avons présenté l'environnement matériel et logiciel utilisé dans la réalisation. Ensuite, nous avons fourni quelques captures d'écran décrivant des interfaces de notre application mobile.

Conclusion Générale



Conclusion générale

Tout au long de ce rapport, nous avons présenté les différentes étapes de développement de ce projet. Le Processus Unifié nous a aidés dans la capture des besoins spécifiques de notre application qui est une étape cruciale pour mieux assimiler le système déjà existant. Ensuite vient l'analyse qui a permis la conception d'une architecture de base pour le développement de notre programme. Puis, la conception, qui s'est basé sur l'utilisation de UML comme langage de modélisation. Après ça, l'implémentation avec laquelle nous avons développé notre application en tenant compte de l'architecture matériel et l'environnement logiciel. Et enfin la phase de test afin de finaliser et de mener à bien notre réalisation de notre application Android.

Ce projet nous a donné l'opportunité de nous imprégner de la vie professionnelle ans un milieu réel, d'avoir un début d'expérience significatif et de travailler en groupe. Il nous a permis de d'accroitre notre champ de compétence, comment compter sur ses capacités, être bien organisés pour accomplir dans les meilleurs délais et les meilleures conditions les tâches qui nous ont été confiées.

Après notre initiation au domaine Android, nous avons appris à travailler avec de nouvelles technologies pour arriver à développer notre application et nous avons conçu une application fiable et fonctionnelle. Toutefois, il est difficile de prétendre avoir créé une application parfaite, mais nous espérons avoir répondu tant soit peu aux attentes de l'entreprise.

Références

[1] https:// www.birger.technology/frblog/?p=50, consulter le 02/09/2020.

[2] https://fr.wikipedia.org/wiki/Appareil_mobile#Smartphone, consulter le 02/09/2020.

[3] https://cours-informatique-gratuit.fr/dictionnaire/smartphone/, consulter le 03/09/2020.

[4] https://www.taktilcommunication.com/blog/applications-mobile/definition-typologie applications -mobiles.html, consulter le 05/09/2020.

[5] https://www.wearecom.fr/dictionnaire/application-mobile/, consulter le 03/09/2020.

[6] https://skylineapps.wordpress.com/2012/01/16/the-birth-of-the-mobile-applications/, consulter le 04/09/2020

[7] https://www.codeur.com/blog/type-dapplication-mobile/, consulter le 04/09/2020.

[8]https://www.definitionsmarketing.com/definition/applicationnative/#:~:text=Une%20application%20native%20est%20une,langage%20de%20d%C3%A9veloppement%20Objective%2DC, consulter le 06/09/2020.

[9] http://glossaire.infowebmaster.fr/application-web-mobile/#:~:text=Une%20application%20web%20mobile%20d%C3%A9signe,accessible%20via%20un%20navigateur%20mobile, consulter le 08/09/2020.

[10] https://cours-informatique-gratuit.fr/dictionnaire/android/, consulter le 08/09/2020.

[11] http://www-igm.univ-mlv.fr/~dr/XPOSE2008/android/archi_comp.html, consulter le 09/09/2020.

[12] K. MRABET N. TRABELSI, Application de messagerie simple sur Android : Rapport de projet de VAP RSM.Université TELECOM SudParis, 2010/2011, consulter le 10/09/2020.

[13]https://mathias-seguy.developpez.com/tutoriels/android/comprendre-cyclevie-activite/, consulter le 14/09/2020.

[14] https://sabricole.developpez.com/uml/tutoriel/unifiedProcess/, consulter le 11/09/2020

[15] http://fdigallo.online.fr/cours/uml.pdf, consulter le 15/09/2020.

[16] https://group.jumia.com/, consulter le 02/09/2020.

[17] http://dev1-0.com/uploads/media/2TUP_SPEC_DEV1.pdf, consulter le 05/09/2020.

[18] https://tchammpa-tn.blogspot.com/2017/10/ la-gestion-des-services-de-la- societe4.html, consulter le 14/09/2020.

[19] http://remy-manu.no-ip.biz/, consulter le 16/09/2020.

[20] https://www.jedha.co /blog/quest-ce-que-la-programmation-orientee-objet, consulter le 18/09/2020.

[21]http://www.opentuto.com/pourquoi-utiliser-la-programmation-orientee-objet/#:~:text=La%20programmation%20orient%C3%A9e%20objet%20am%C3%A8ne,d%C3%A9veloppant%20que%20les%20%C3%A9volutions%20n%C3%A9cessaires, consulter le 18/09/2020.

[22] https://laurent-audibert.developpez.com/Cours-UML/, consulter le 21/09/2020.

[23] https://laurent-audibert.developpez.com/Cours-UML/?page=mise--oeuvre-uml, consulter le 22/09/2020.

[24]https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php /4678736-comment-fonctionne-une-architecturemvc#:~:text=Le%20pattern%20MVC%20 permet %20de, les%20donn %C3%A9es%20de%20votre%20site, consulter le 26/09/2020.

[25]https://www.google.com/search?q=MVC&sxsrf=ALeKk02SAE1zSdJwGvLbnSrJ_tobFlMnzA:1602174634029&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiSp7fPtaXsAhXwyIUKHWagCAUQ_AUoAXoECBkQAw&biw=1366&bih=657#imgrc=lS3PmaUddoGIKM, consulter le 26/09/2020.

[26] https://www.memoireonline.com/03/12/5548/m_Rapport-de-stage-sur-le-projet-Locate-my-car-google-map-android10.html, consulter le 27/09/2020.

[27] https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes, consulter le 29/09/2020.

[28] https://www.jetbrains.com/help/idea/discover-intellij-idea.html, consulter le 08/10/2020.

[29] http://tutoriels.meddeb.net/postgres-pgadmin-4utilisation/#:~:text=Postgres%20pgAdmin%204%20permet%20de,d'ex%C3%A9cuter%20des%20requ%C3%AAtes%20SQL, consulter le 10/10/2020.

[30] https://www.postgresql.org/about/, consulter le 10/10/2020.

[31] https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html, consulter le 11/10/2020.

[32] https://laurent-audibert.developpez.com/Cours-BD/?page=langage-sql, consulter le 13/10/2020.

[33] https://www.investopedia.com/terms/x/extensible-markup-language-xml.asp, consulter le 15/10/2020.

[34] https://www.json.org/json-en.html, consulter le 15/10/2020.

[35] https://code.tutsplus.com/tutorials/connect-to-an-api-with-retrofit-rxjava-2-and-kotlin--cms-32133, consulter le 16/10/2020.

[36] https://stackshare.io/visual-paradigm, consulter le 16/10/2020.

Résumé

Les entreprises commerciales ont comme but de commercialiser des produits et des services. Et pour mieux faire leurs travaux, ils cherchent toujours à rapprocher au maximum le client de leurs produits tout en essayant de le mettre le plus à l'aise possible.

Vu que la plupart des personnes possèdent un smartphone, il est plus pratique de développer une application mobile pour que cette personne se trouve, elle ait toujours un œil sur chacun des produits que propose l'entreprise commerciale.

Cette application mobile facilite l'achat, la vente de produits et la communication entre l'entité client et l'entreprise commerciale. Dans le cadre de notre projet de fin d'études, nous proposons une solution de vente et d'achat qui dispose des caractéristiques suivantes :

- Un système central (serveur) qui s'appuie sur une API RESTful et une base de données PostgreSQL et qui s'occupe de la gestion des ventes d'une entreprise commerciale et des achats de leurs clients.
- Une application mobile qui offrira essentiellement des fonctionnalités de gestion de vente et de clientèle pour l'entreprise, de gestion d'achat pour la clientèle et un service de communication entre ces deux entités.
- Une base de données locale permettant la gestion des données en cas de rupture de connexion avec la base de données distante.

Mots clés : Android, Gestion commerciale, Application mobile, UP, UML.

Abstract

The purpose of commercial enterprises is to market products and services. And in order to do their job better, they always try to bring the customer as close as possible to their products while trying to make them as comfortable as possible.

Since most people own a smartphone, it is more practical to develop a mobile application in order to always have an eye on each of the products offered by the commercial company, wherever that person is.

This mobile application facilitates the purchase, sale of products and communication between the customer entity and the commercial enterprise. Within the framework of our graduation project; we propose a sale and purchasing solution that has the following features:

- A central system (server) that relies on a RESTful API and a PostgreSQL database and takes care of managing the sales of a commercial company and the purchases of their customers.
- A mobile application that will essentially offer sales and customer management functionalities for the company, purchase management for the customers and a communication service between these two entities.
- A local database allowing data management in the event of a loss of connection with the remote database.

Keywords: Android, Business management, Mobile application, UP, UML.