République Algérienne Démocratique et Populaire
Ministère de L'Enseignement Supérieur et de la Recherche Scientifique

Université Abderrahmane MIRA – Béjaïa
Faculté des sciences Exactes
Département d'Informatique

Mémoire de fin de cycle

En vue de l'obtention du diplôme de Master Informatique

Option : Génie Logiciel

Présenté par :
MINDJOU Bilal

Thème

Conception et Réalisation d'une Application Mobile bancaire Cas pratique : BNA

Encadré par : Zoubeyr FARAH

M. Sofiane AISSANI	Maître Conférence Classe A (Univ-Béjaïa)	Examinateur
M. Farid KACIMI	Maître Conférence Classe B (Univ-Béjaïa)	Examinateur
M. Zoubeyr FARAH	Maître Conférence Classe A (Univ-Béjaïa)	Encadreur

Année 2019-2020

Remerciements

Pour commencer, je tiens à remercier en premier lieu le bon Dieu de m'avoir donné la force et le courage pour accomplir ce modeste travail.

Je tiens à remercier mon encadrant M.Zoubeyr FARAH, pour son orientation, sa confiance, et sa patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené à bon port.

Mes remerciements s'étendent également à M^{me} BARKA Yasmine pour le précieux temps qu'elle m'a consacré, aux personnes qui m 'ont apporté leur aide précieuse et qui ont contribué à l'élaboration de ce travail.

Un merci particulier à mes parents, pour leur amour, leur sacrifices et leur patiences. Je tiens également à remercier sincèrement les membres du jury ; Monsieur Sofaine AISSANI, et Monsieur Farid KACIMI pour l'intérêt qu'ils ont porté pour mon travail, pour les remarques et les conseils contribuant ainsi à son perfectionnement.

Enfin, je remercier, de tout coeur, tous ceux qui ont contribué de près ou de loin à la réalisation de ce mémoire.

Dédicaces

Rien n'est aussi beau á offrir que le fruit d'un labeur qu'on dédie du fond du coeur á ceux qu'on aime et qu'on remercie en exprimant la gratitude et la reconnaissance durant toute notre existence.

Je dédie ce mémoire :

A mes trés cher parent qui ont toujours été là pour moi, qui ont sacrifiè leur vie pour ma rèussite et m' ont èclairer le chemin par leurs conseils judicieux. J'espaire qu'un jour, je pourrais leur rendre un peu de ce qu'ils ont fait pour moi. Que dieu leur prête bonheur et longue vie ,

A mon frére et soeur et, mes cousins et cousines, à toutes ma famille,

A mes meilleurs amis et tous ceux qui me sont chers ,

A tous mes professeurs qui m'ont enseignés.

M. MINDJOU Bilal

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

BNA	Banque Nationale d'Algérie
CSS	Cascading Style Sheets
EDI	Environnement de Développement Intégré
GPL	General Public License
HTML	HyperText Markup Language
HTTP	Hyper Text Transfer Protocol
JSON	JavaScript Object Nation
JS	JavaScript
PHP	HyperText Préprocesseur
REST	REpresentational State Transfer
RPC	Remote Procedure Call
SDK	Software Developpement Kit
SE	Système d'Exploitation
SGBD	Système de Gestion de Base de Données
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
UML	Unified Modeling Language
UP	Unified Process
XML	Extensible Markup Language

Introduction Générale

Actuellement, le monde connaît une avancée technologique considérable dans tous les secteurs et celà grace à la généralisation de l'utilisation de l'Informatique.

Ainsi, l'essor des Nouvelles Technologies de l'information et de la Communication (NTIC) a donné naissance à de nouveaux équipements mobiles qui marque sans doute de façon décisive une évolution majeure ces dernières décennies. Cette évolution, notamment dans le domaine de l'informatique, a donné naissance à de nouveaux équipements mobiles tels que les Smartphones, les tablettes, etc. Le secteur bancaire n'a donc pas fait figure d'exception face au phénomène de l'émergence des NTIC. Les banques essaient alors de mettre à profit les avantages offerts par ces technologies en particulier l'accès à l'internet. Le défis actuel des banques est d'anticiper les demandes des consommateurs en matière de rapidité d'accès et des services de qualité. L'adaptation des services bancaires en ligne au profil de la clientèle et à ses attentes est considérée comme étant un des avantages les plus importants de l'utilisation des nouvelles technologies dans le secteur bancaire.

Il est évident que le client d'aujourd'hui est tout-à-fait différent de celui d'hier. Avec l'apparition des distributeurs automatiques de billets et ensuite les applications mobiles, les clients rencontrent de moins en moins leurs banquiers. Tout cela a modifié énormément la relation entre les banques et leurs clients puisque ces derniers peuvent désormais accéder aux services de leurs banques à n'importe quel moment et indépendamment de leur localisation.

Notre travail consiste à exploiter les nouvelles technologies dans le secteur bancaire, où nous allons utiliser les technologies web et mobile pour développer un système composé d'une application mobile qui permettra aux clients d'être plus proche de leur banque tout en étant mobile et cela en entrant dans l'air des nouvelles technologies mobiles et de créer des opportunités stratégiques et commerciales pour la banque.

Pour réaliser ce travail, nous avons dréssé un plan qui couvre l'ensemble des parties du projet, à commencer par la partie purement théorique jusqu'aux concepts de programmation et d'implémentation.

Notre travail est organisé en quatre chapitres, comme suit :

- Le premier chapitre intitulé «Généralités» regroupe des généralités sur l'organisme d'accueil BNA et les différentes parties du système que nous allons développer.
- Le deuxième chapitre intitulé «Étude préliminaire» donne un premier aperçu sur les différentes facettes et fonctionnalités qui seront présentées dans notre système.
- Le troisième chapitre intitulé «Analyse et Conception» donne plus de détails sur les fonctionnalités principales de notre système et c'est la base sur laquelle nous allons nous appuyer pour l'implémentation de ce dernier.
- Le quatrième et dernier chapitre intitulé «Implémentation» concerne la partie pratique c'est à dire les détails de la partie réalisation de notre système.

Chapitre 1

Géneralités

Introduction

Dans ce chapitre, nous présentons l'organisme d'accueil BNA, la société au sein de laquelle j'ai effectué mon stage pratique de fin d'études. Ensuite, nous présentons le cadre du projet qui permet de mieux comprendre le problème étudié et présenter le principe de fonctionnement des systèmes mobiles et de leurs applications dans le domaine métier des banques. .

Enfin, nous terminons ce chapitre par la présentation de la solution retenue qui sera détaillée plus loin dans le rapport.

1.1. Présentation de l'organisme d'accueil

1.1.1. Définition des banques

Les banques sont des entreprises ou des établissements, leur fonction principale consiste à collecter des ressources et distribuer des crédits, chaque banque est spécialisée selon son activité principale et sa clientèle[**1**].

1.1.2. Historique de la BNA

La Banque Nationale d'Algérie (BNA) a été crée le 13 juin 1966. Aux termes de ses statuts originels, la BNA a la qualité de banque de dépôts. Elle est chargée d'assurer le service financier des groupements professionnels, des entreprises et exploitation du secteur socialiste et du secteur public et de participer au contrôle de leur gestion.

La BNA exerçait toutes les activités d'une banque de dépôts. En outre, elle détenait le monopole du financement de l'agriculture, jusqu'à mars 1982 date à laquelle les pouvoirs publics ont décidé de l'opportunité de mettre en place une institution bancaire spécialisée, ayant pour vocation principale la prise en charge du financement et de la promotion du monde rural. Ainsi, a été crée la banque de l'agriculture et développement rural (BADR) à partir de la restructuration de la BNA. Dans le domaine industriel et commercial, la banque nationale contribuait au financement d'une bonne partie de ce secteur. Elle accordait des crédits à court terme aux secteurs étatiques et privés. Elle intervenait également dans le financement des crédits à moyen terme liés à la réalisation d'investissements planifiés productifs[**1**].

1.1.3. Organisation et Missions d'une agence de la BNA

Dans cette section, nous expliquons l'organisation d'une agence BNA, ses missions et ses objectifs principaux.

1.1.3.1. Organisation d'une agence de la BNA

L'agence entretient des relations fonctionnelles avec l'ensemble des structures de la banque. Selon les attributions qui lui sont conférées, l'agence est classée en fonction du niveau d'activité déployée. Elle relève des catégories suivantes[**2**] :

— Agence principale.

— Agence première catégorie.

— Agence de deuxième catégorie.

— Agence de troisième catégorie.

L'agence principale et celle de première catégorie sont dirigées par un directeur et deux directeurs adjoints selon leur importance et le nombre de clientèle géré.

L'agence de deuxième et celle de troisième catégorie sont dirigées par un directeur et un directeur adjoint. Elles sont respectivement restructurées en cinq et trois services.

1.1.3.2. Missions d'une agence de la BNA

La BNA exerce toutes les activités d'une banque de dépôts : elle assure notamment le service financier des groupements professionnels et des entreprises, elle traite toutes les opérations de banque, de change et de crédit dans le cadre de la législation et de la réglementation des banques et peut notamment [**2**] :

- Recevoir du public des dépôts de fonds, en compte ou autrement, remboursable à vue, à terme ou à échéance fixe, émettre des bons et des obligations : emprunts pour les besoins de son activité ;
- Effectuer et recevoir tout paiement en espèce, par chèque, virements, domiciliation, lettre de crédits et autres activités de banques ;
- Consentir sous toute forme de crédits, prêts ou avances avec ou sans garanties ;
- Louer tous les coffres et compartiments de coffres ;
- Servir d'intermédiaire pour l'achat, la souscription ou la vente de tous les effets publics, actions, obligations, plus généralement, de toutes les valeurs mobilières, ainsi que des métaux précieux ;
- Traiter toutes les opérations de change, au comptant ou à terme, contracter tous emprunts, prêts, nantissements, report de devises étrangères ;
- Financer par tous modes les opérations de commerce extérieur.

1.1.3.3. Objectifs de la BNA

La réaction des fonctions bancaires ainsi que le mode de fonctionnement des entreprises jouent un très grand rôle dans l'évolution de l'économie du pays.

En relation avec cette évolution, la BNA a pour objectifs de :

- S'adapter aux règles de la commercialité dans ses rapports avec sa clientèle commerciale qui connaît déjà de profonds changements dans ses structures et son organisation ;
- Améliorer sa rentabilité via un accroissement des ressources, contrepartie des crédits et par la promotion des services qui directement ou indirectement peuvent encore augmenter d'avantage cette rentabilité ;
- La préservation de ses propres équilibres ;
- Respecter les règles de gestion providentielle afin de créer de la monnaie, du crédit, des changes et les conditions les plus favorables à un développement ordonné de l'économie nationale.[**2**]

1.2. Problématique et objectif du travail

Nous présentons la problématique et l'objectif de notre travaille.

1.2.1. Problématique

Le service bancaire a manifesté le désir de trouver une solution pour mettre leur clientèle dans le confort, car de nos jours, le client n'a plus forcément le temps de se déplacer, afin d'obtenir des informations et des services concernant leur banque, ou carrément il ne connait pas l'agence la plus proche pour faire le retrait de son argent, et cela s'applique sur les clients de la banque BNA, ces dernier désirant effectuer des opérations à distance et obtenir ses informations sans contraintes temporelles, de plus, cela permettra aux agences de déminuer la pression sur leurs guichets.

L'informatique est presque devenue indispensable au niveau de tous les domaines de notre vie courante, c'est pourquoi il est nécessaire aujourd'hui pour une banque d'offrir à ses clients une application mobile permettant de réaliser des opérations en toute sécurité et à tout moment, de même elle doit proposer une partie administrative à ses employés. Ainsi nous résumons notre problématique dans les points suivants : Ainsi nous résumons notre problématique dans les points suivants :

— Comment intégrer la technologie du mobile à ce domaine ?

— Comment bien choisir les moyens pour réaliser une application qui répond aux besoins de ce service et de satisfaire leurs attentes en termes de fonctionnalités et d'ergonomie ?

— Quelle est la démarche à suivre afin d'aboutir à une telle solution pour réaliser un tel système ?

1.2.2. Objectif du travail

Pendant la période de stage pratique passé au niveau des services de la BNA, j'ai pu comprendre l'organisme et le fonctionnement de ce service, ce qui m'a permis après une étude et analyse préliminaire d'opter pour les propositions suivantes qui pourraient bien répondre aux questions posées précédemment.

— Comme présenté dans la figure 1.1, l'idée de ce projet dans le secteur bancaire est de réaliser une application mobile qui va satisfaire les besoins des clients dans le secteur bancaire et leur faire gagner du temps en limitant leur déplacement aux

agences et de bénéficier des services informationnels et services transactionnels sur sa banque ;

— Sécuriser la sauvegarde des données ;

— L'accès au niveau du serveur distant en se basant sur une architecture 3-tiers ;

— Une application web pour la partie administration pour les employés.

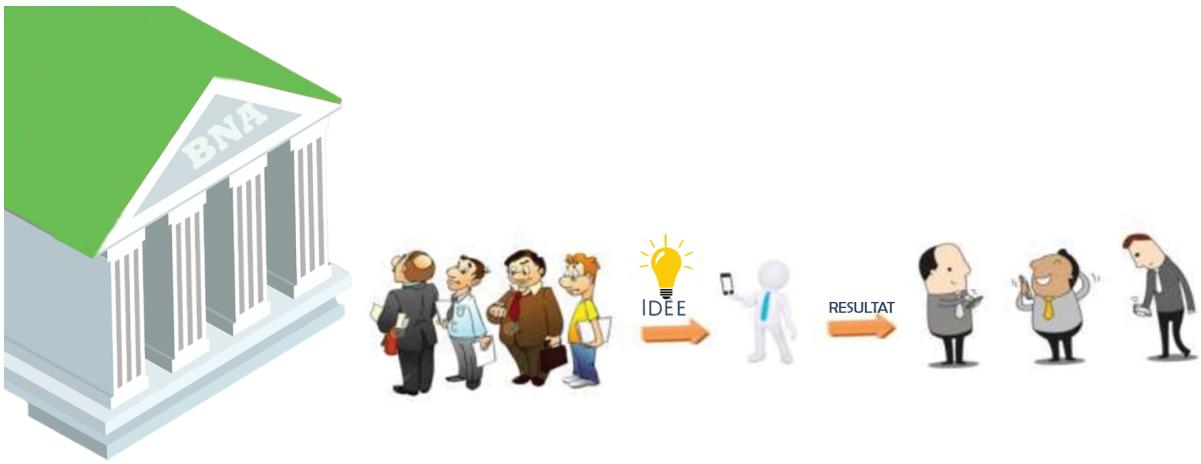


FIGURE 1.1 – Idée du projet.

1.3. Application mobile

Une application mobile n'est rien d'autre qu'un logiciel téléchargeable, que l'on installe facilement sur nos smartphones (téléphones mobiles intelligents), comme on ferait sur nos ordinateurs.

Il existe trois types d'applications mobiles selon leurs spécificités techniques :

Applications Natives : Ces applications sont liées au système d'exploitation sur lequel elles sont installées, car elles utilisent des caractéristiques reliées à celui-ci. Elle sont écrites dans un langage adapté au système d'exploitation en question.

Applications Web : Ce sont toutes les applications conçues grâce aux outils de développement web actuels (HTML, CSS, JavaScript, etc.). Elles sont accessibles sur tous les mobiles via un navigateur Web ce qui les rend plus intéressantes du point de vue financier, car les coûts de développement sont réduits vu qu'on développe une seule application qui est compatible avec tous les smartphones quelque soient leurs systèmes.

Applications Hybrides : Sont des applications qui incorporent les deux principes de développement précédemment cités : les caractéristiques des applications web et celles des applications natives. Elles pourront être distribuées sur les plate-formes de téléchargement telles que l'Apple Store (iOS), Play Store (Android) ou encore Windows Store (Windows Phone). L'utilisateur peut donc installer ces applications et consulter leur contenu sans avoir à passer par un navigateur web, car l'application elle même, incorpore un navigateur web intégré.

La figure suivante illustre les trois types cités :

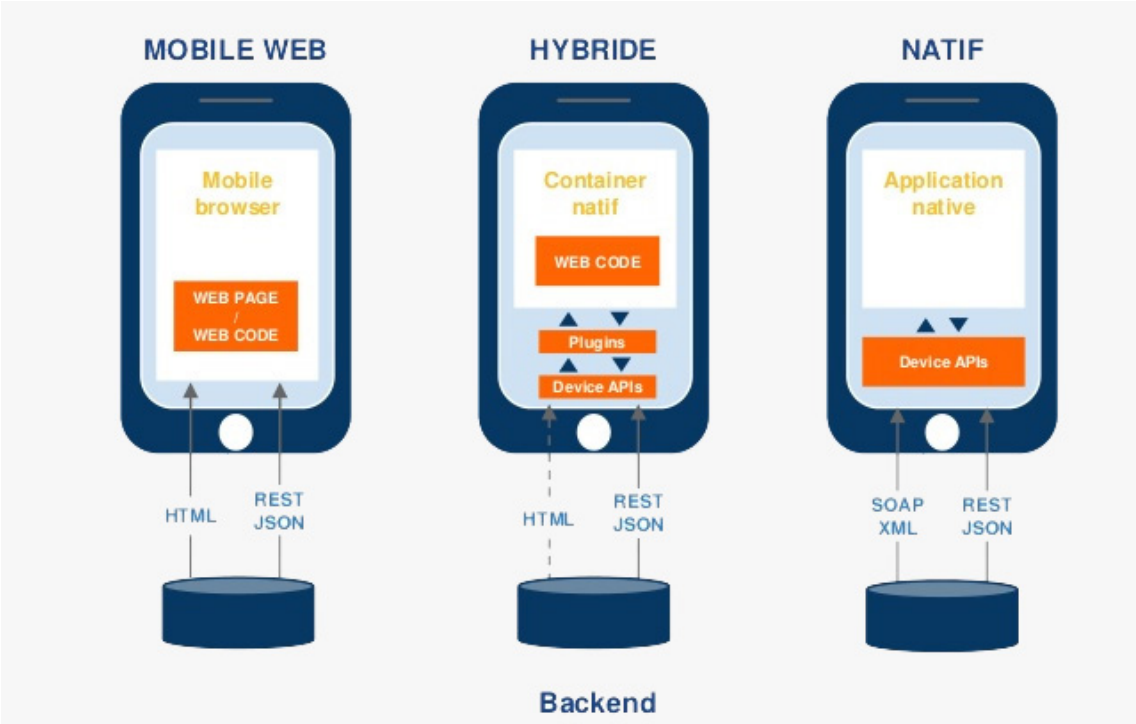


FIGURE 1.2 – Les différents types d'applications mobiles [**3**].

1.3.1. Système d'exploitation Android

Android est un système d'exploitation et plate-forme logicielle pour smartphones et tablettes créé par Google dont la première version beta a été proposée en Novembre 2007 [**4**].

1.3.1.1. Architecture d'Android

Android est en fait une plateforme qui inclut un système d'exploitation basé sur le noyau Linux, des middlewares (couches logicielles tierces) et des applications clés. Globalement divisée en quatre zones comme le montre le graphique suivant :

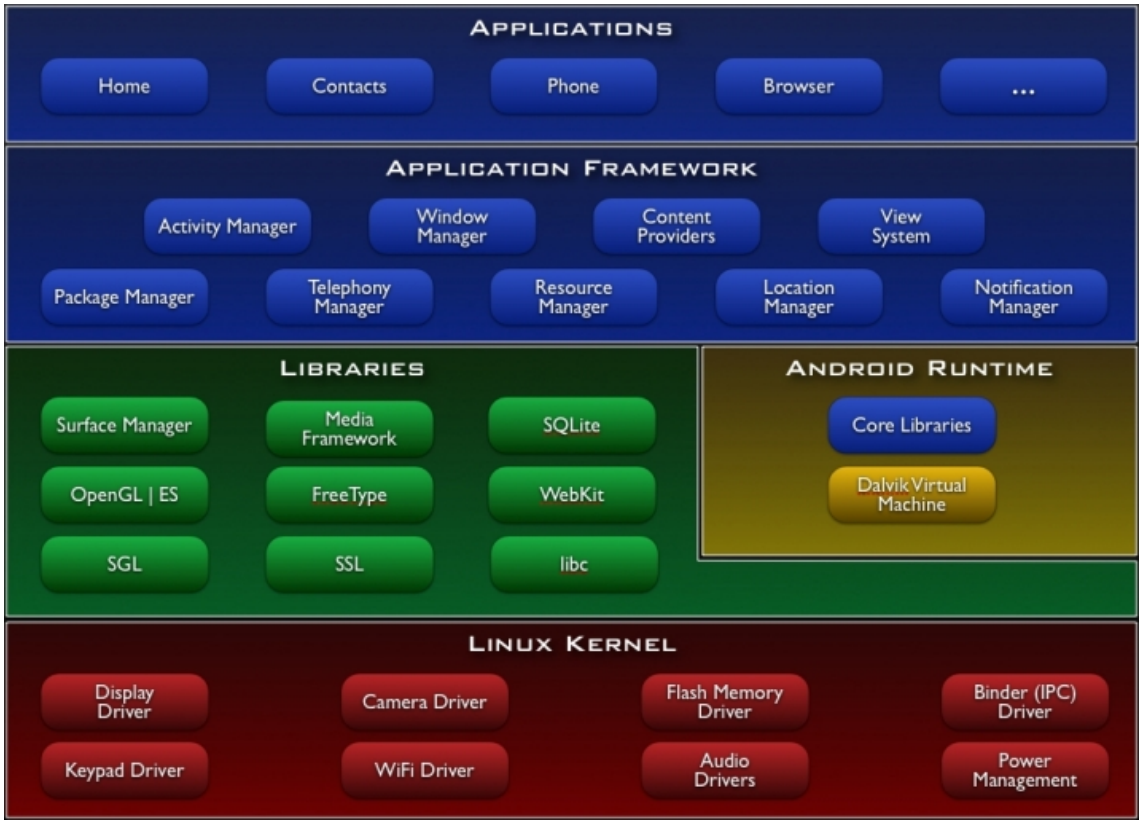


FIGURE 1.3 – L'architecture d'Android [**4**].

Ces éléments peuvent être décrits comme suit :

— **Applications** : Le projet Open Source Android contient plusieurs applications par défaut, comme le navigateur, l'appareil photo, la galerie, la musique, le téléphone et plus encore.

— **Application Framework** : C'est une API qui permet aux applications Android d'interagir avec le système Android.

— **Libraries and runtime** : C'est des bibliothèques assurant de nombreuses fonctions communes (le rendu graphique, le stockage de données, la navigation sur le Web, etc.) de l'Application Framework et du moteur Dalvik, ainsi que le noyau de bibliothèques Java pour exécuter des applications Android.

— **Linux kernel** : C'est la couche de communication avec le matériel sous-jacent, un noyau linux modifié et optimisé pour des systèmes avec ressources limitées

(Faible processeur, mémoire, batterie, ...).

Le noyau Linux, les bibliothèques et le moteur d'exécution sont encapsulés par l'Application Framework. Le développeur d'applications Android travail généralement avec les deux couches supérieures pour créer de nouvelles applications Android [**5**] .

1.3.1.2. Avantages de la plateforme Android

- Pas de Licence à obtenir, pas de dépenses pour la distribution et le développement.
- Développer des applications location-based en utilisant le GPS.
- Utiliser des cartes géographiques avec Google Maps.
- Recevoir et émettre des SMS, envoyer et recevoir des données sur le réseau mobile.
- Enregistrer et lire image, son et vidéo.
- Des outils de stockage de données partagés (SQLite en version Sandbox).
- Une accélération matérielle pour la 2D et la 3D.
- Des services et des applications qui peuvent tourner en tâche de fond : qui peuvent réagir au cours d'une action, à votre position dans la ville, à l'heure qu'il est, suivant l'identité de l'appelant.
- Une plateforme de développement qui favorise la réutilisation de composants logiciels et le remplacement des applications fournies .
- Un langage de programmation simple.
- Un des meilleurs environnements de test application par rapport aux principaux rivaux systèmes d'exploitation[**4**].

1.4. Services web

Nous présentons dans cette partie les services web et les différents protocoles qui permettent de communiquer avec ce dernier.

1.4.1. Définition d'un service web

Un service web est un composant logiciel identifié par une URI, dont les interfaces publiques peuvent être appelées en utilisant plusieurs langages. Sa définition peut être découverte par d'autres systèmes logiciels.

Les services web peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages (ex :XML) portés par les protocoles Internet[**6**].

1.4.2. Protocoles de communication avec les services web

Simple Object Access Protocol (SOAP) : SOAP est un protocole d'accès aux services web basés sur des normes qui existent depuis un certain temps et bénéficie de tous les avantages de l'utilisation à long terme. Initialement développé par Microsoft, SOAP est vraiment pas aussi simple que son acronyme le semble [**7**].

REpresentational State Transfer (REST) : Il vise à résoudre les problèmes rencontrés avec SOAP et fournir une méthode vraiment simple, pour accéder à des services web. Cependant, SOAP est parfois plus facile à utiliser, REST peut causer parfois quant à lui ses propres problèmes. Les deux techniques ont des problèmes à prendre en compte au moment de décider quel protocole utiliser [**7**].

XML-RPC : XML-RPC fait partie des approches de services web les plus simples et permet aux ordinateurs d'appeler facilement des procédures sur d'autres ordinateurs.

XML-RPC réutilise l'infrastructure créée à l'origine pour les communications entre humains afin de prendre en charge les communications entre les programmes sur les ordinateurs. Le langage XML (Extensible Markup Language)fournit un vocabulaire permettant de décrire les appels de procédure distante(RPC), qui sont en suite transmis entre ordinateurs à l'aide du protocole HTTP (Hyper Text Transfer Protocol) [**8**].

1.5. Conclusion

Dans ce premier chapitre, nous avons présenté le contexte et la problématique abordés dans ce projet. nous avons présenté brièvement l'organisme d'accueil et les applications mobiles. Dans le chapitre suivant, nous entamerons la phase de spécification des besoins qui nous permettra de modéliser et de rédiger un cahier des charges incluant les besoins à satisfaire, les besoins auxquels, notre application est censée répondre et proposer une solution aux besoins.

Chapitre 2

ÉTUDE PRÉLIMINAIRE

Introduction

Dans ce chapitre, nous présentons les différentes parties de notre projet, la démarche adoptée pour l'élaboration de notre projet ainsi que la première phase qui est l'expression des besoins.

2.1. La méthodologie

Le choix du processus de développement à utiliser se fait selon plusieurs critères, tels que la complexité, le type de projet, le délai de livraison, le coût de développement et les compétences de l'équipe.

Notre choix c'est porté sur l'UP pour sa simplicité et sa compatibilité avec notre projet et comme langage de modélisation l'UML.

2.1.1. Processus UP (Unified Process)

2.1.1.1. Définition des principaux concepts et schéma d'ensemble

Le processus unifé décrit qui fait quoi, comment et quand les travaux sont réalisés tout au long du cycle de vie du projet. Quatre concepts d'UP répondent à ces questions [**9**] :

— Rôle (qui ?)
— Activité (comment ?)
— Artefact (quoi ?)
— Workflow (quand ?)

— **Rôle**
Un rôle définit le comportement et les responsabilités d'une ressource ou d'un groupe de ressources travaillant en équipe. Le rôle doit être considéré en termes de « casquette » qu'une ressource peut revêtir sur le projet. Une ressource peut jouer plusieurs rôles sur le projet.

— **Activité**
Les rôles ont des activités qui définissent le travail qu'ils effectuent. Une activité est une unité de travail qu'une ressource, dans un rôle bien précis, peut effectuer et qui produit un résultat dans le cadre du projet. L'activité a un but clairement établi, généralement exprimée en termes de création ou de mise à jour d'artefacts, comme un modèle , une classe ou un planning. Les ressources sont affectées aux activités selon leurs compétences et leur disponibilité.

— **Artefact**
Un artefact est un ensemble d'informations qui est produit, modifié ou utilisé par le processus. Les artefacts sont les produits effectifs du projet. Les artefacts sont utilisés comme input par les ressources pour effectuer une activité et sont le résultat d'output d'activités du processus.

— **Workflow**
Une énumération de tous les rôles, activités et artefacts ne constitue pas un processus. En effet, il est nécessaire d'avoir une façon de décrire des séquences d'activités mesurables qui produisent un résultat de qualité montre l'interaction entre les ressources. Le workflow est une séquence d'activités qui produit un résultat mesurable. En UML, il peut être exprimé par un diagramme de séquence, un diagramme de communication ou un diagramme d'activités.

— **Schéma d'ensemble**

UP peut être décrit suivant deux dimensions traduites en deux axes :

— Un axe horizontal représentant le temps et montrant l'aspect dynamique du processus. Sur cet axe, le processus est organisé en phases et itérations ;

— Un axe vertical représentant l'aspect statique du processus. Sur cet axe,le processus est organisé en activités et workflows.

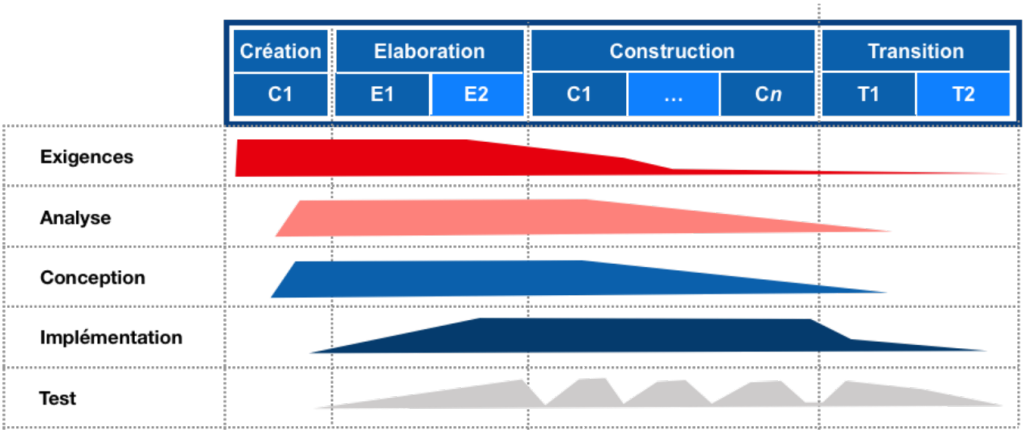


FIGURE 2.1 – Schéma Processus unifié (UP) .

2.1.1.2. Phases et itérations du processus (aspect dynamique)

Le processus unifié, organisé en fonction du temps, est divisé en quatre phases successives :[**10**]

— Inception (Analyse des besoins) ;

— Élaboration ;

— Construction ;

— Transition.

— **Inception (Analyse des besoins)**
Cette phase correspond à l'initialisation du projet où l'on mène une étude d'opportunité et de faisabilité du système à construire. Une évaluation des risques est aussi réalisée dès cette phase. En outre, une identification des principaux cas d'utilisation accompagnée d'une description générale est modélisée dans un diagramme de cas d'utilisation afin de définir le périmètre du projet.

— **Élaboration**

Cette phase reprend les résultats de la phase d'inception et élargit l'appréciation de la faisabilité sur la quasi-totalité des cas d'utilisation. Ces cas d'utilisation se retrouvent dans le diagramme des cas d'utilisation qui est ainsi complété. Cette phase a aussi pour but d'analyser le domain et technique du système à développer afin d'aboutir à une architecture stable. Ainsi, toutes les exigences non recensées dans les cas d'utilisation, comme par exemple les exigences de performances du système, seront prises en compte dans la conception et l'élaboration de l'architecture.

— **Construction**

Une énumération de tous les rôles, activités et artefacts ne constitue pas un processus. En effet, il est nécessaire d'avoir une façon de décrire des séquences d'activités mesurables qui produisent un résultat de qualité montre l'interaction entre les ressources. Le workflow est une séquence d'activités qui produit un résultat mesurable. En UML, il peut être exprimé par un diagramme de séquence, un diagramme de communication ou un diagramme d'activité.

— **Transition**

Après les opérations de test menées dans la phase précédente, il s'agit dans cette phase de livrer le produit pour une exploitation réelle. C'est ainsi que toutes les actions liées au déploiement sont traitées dans cette phase.

2.1.1.3. Activités du processus (aspect statique)

— **Expression des besoins :**

UP propose d'appréhender l'expression des besoins en se fondant sur une bonne compréhension du domaine concerné pour le système à développer et une modélisation des procédures du système existant [**10**]. Ainsi, UP distingue deux types de besoins :

- Les besoins fonctionnels qui conduisent à l'élaboration des cas d'utilisations ;
- Les besoins non fonctionnels (techniques) qui aboutissent à la rédaction d'une matrice des exigences.

— **Analyse :**

L'analyse permet une formalisation du système à développer en réponse à l'expression des besoins formulée par les utilisateurs. L'analyse se concrétise par l'élaboration de tous les diagrammes donnant une représentation du système de

séquence, d'activité, d'état-transition, etc).

— **Conception :**
La conception prend en compte les choix d'architecture et technique retenus pour le développement et l'exploitation du système. La conception permet d'étendre la représentation des diagrammes effectuée au niveau de l'analyse en y'intégrant les aspects techniques plus proches des préoccupations physiques.

— **Implémentation :**
Cette phase correspond à la production du logiciel sous forme de composants, de bibliothèques ou de fichiers. Cette phase reste, comme dans toutes les autres méthodes, la plus lourde en charge par rapport à l'ensemble des autres phases (au moins 40%).

— **Test :**
Les tests permettent de vérifier :
  — La bonne implémentation de toutes les exigences (fonctionnelles et techniques) ;
  — Le fonctionnement correct des interactions entre les objets ;
  — La bonne intégration de tous les composants dans le logiciel.

2.1.2. Langage UML (Unified Modeling Language)

Il se définit comme un langage de modélisation graphique et textuel destiné à comprendre et à définir des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML modélise l'ensemble des données et des traitements en élaborant différents diagrammes[**10**] .

2.1.2.1. Présentation générale des diagrammes

UML dans sa version 2 propose treize diagrammes qui peuvent être utilisés dans la description d'un système. Ces diagrammes sont regroupés dans deux grands ensembles comme le montre la Figure 2.2 .

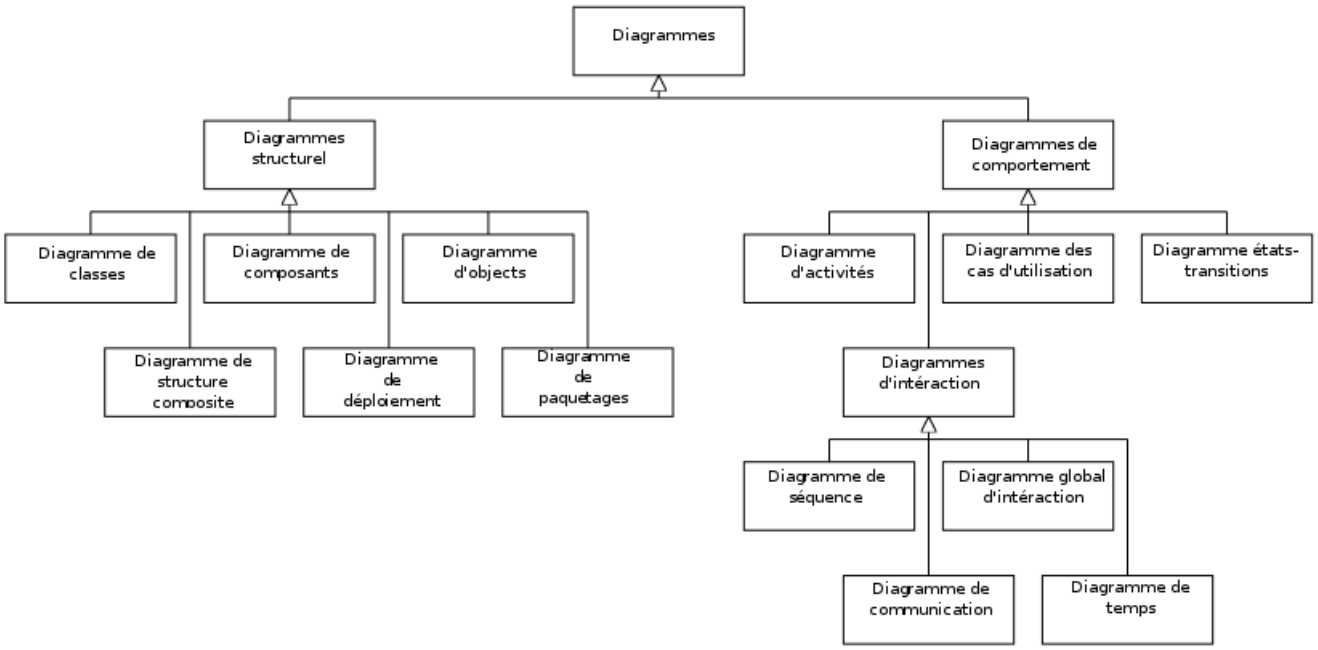


FIGURE 2.2 – Les Diagrammes UML.

— **Les diagrammes structurels :** Ces diagrammes, au nombre de six, ont vocation à représenter l'aspect statique d'un système(classes, objets, composants...).

— **Les diagrammes de comportement :** Ces diagrammes représentent la partie dynamique d'un système réagissant aux événements et permettant de produire les résultats attendus par les utilisateurs. Sept diagrammes sont proposés par UML.

UML 2 décrit les concepts et le formalisme de ces treize diagrammes mais ne propose pas de démarche de construction couvrant l'analyse et la conception d'un système.

Les diagrammes que nous allons présenter sont les diagrammes que nous avons jugé nécessaires à utiliser dans notre conception.

— **Diagramme de cas d'utilisation :**
Ce diagramme est destiné à représenter les besoins des utilisateurs par rapport au système. Il constitue un des diagrammes les plus structurants dans l'analyse d'un système.[**11**]

— **Diagramme de séquence système :**
Le diagramme de séquence représente la succession chronologique des opérations réalisées par un acteur. Il indique les objets que l'acteur va manipuler et les opérations qui font passer d'un objet à l'autre.[**11**]

— **Diagramme d'interaction :**
Ce diagramme permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets.[**11**]

— **Diagramme de classes :**
Le diagramme de classe constitue l'un des pivots essentiels de la modélisation avec UML. En effet, ce diagramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les données et les traitements dont elle est responsable pour elle-même et vis-à-vis des autres classes. Les traitements sont matérialisés par des opérations. Le détail des traitements n'est pas représenté directement dans le diagramme de classe.[**11**]

2.2. Présentation du projet

Cette section comporte les différentes informations de l'application mobile, ainsi que l'application web de contrôle d'accès.

2.2.1. Rappel du cadre méthodologique

La présente étude de cas porte sur une banque (BNA), c'est une forme d'agrégation de produits qui propose en effet à ses clients des produits, qui est en fait un bouquet de services : simuler un crédit, consulter le compte, trouver les agences, effectuer des transactions bancaires, faire des commandes de carte/chéquier, permettre d'éffectuer des oppositions de carte/chéquier. Les données nécessaires à l'élaboration d'une commande proviennent de l'administrateur.

Dans ce cas, notre application mobile que nous bâtissons EasyBNA, consultée directement par deux catégories d'utilisateurs : les visiteurs et les clients de la banque.

— Les visiteurs sont les personnes qui cherchent des informations sur la banque, et devenir les futurs clients.
— Les clients de la banque sont ceux qui ont tous les privilèges des fonctionnalités de l'application mobile.

L'application envisagée, dans le secteur bancaire est une application mobile client

riche serveur sur une plateforme Android. Cette application, comme le montre la figure 2.3, permet à un client d'accéder à partir d'un terminal mobile pour bénéficier des services sur ça banque en toute sécurité après vérification des paramètres d'accès au niveau du serveur distant.

Le réseau internet via la suite des protocoles TCP/IP permet la connexion entre le terminal mobile et le serveur d'application et la transmission de messages se fait par le langage JSON qui constitue un moyen de communication. Le terminal mobile envoi les informations afin de s'authentifier et effectuer une demande de réservation au serveur qui décode ces données et les analyse afin d'accepter ou de refuser l'accès à ses données.

Avec ce projet, j'ai appris comment utiliser une architecture 3-tier et les webservices. Cette architecture divise l'application en trois parties. Le client (téléphone) se connecte à un serveur (Middleware) via des webservices et ce serveur interroge la base de données.

FIGURE 2.3 – Interaction entre le client et le serveur.

2.2.2. Description générale

— **Environnement**

Il faut utiliser :

- Un serveur web :Apache.
- Un SGBD : MySQL.

— **Caractéristiques de nos applications :**

- L'application mobile :
  — Responsive : S'adapte à toutes les tailles d'écran des appareils mobiles ;
  — S'adapte aux deux modes d'affichage mobile, portrait et paysage.
  — **Disponibilité :** Disponible (24h/24h) et (7j/7) et depuis n'importe où dans le monde.
  — **Flexible et scalablé :** Peut-être améliorée et avoir plus de fonctionnalités.
- L'application web :
  — Responsive : L'utilisation de Bootstrap est recommandée, pour l'adaptation à tout type de support (mobile, tablette et ordinateur) ;
  — Compatible avec les dernières versions des navigateurs : Chrome, Firefoxet IE.
- Il est important aussi de respecter les points qui seront définis dans la charte graphique.

— **Caractéristiques des utilisateurs :**

Notre application cible tous les clients de la banque, souhaitant informatiser leurs services bancaires et les rendre en ligne et aussi, les visiteurs qui sont curieux de connaitre les différents produits offerts par BNA et qui pourront à l'avenir être les futurs clients.

2.2.3. Charte graphique

La charte graphique est l'identité visuelle d'une application, entreprise où marque, c'est un guide qui détermine tous les éléments graphiques (logo, couleur, police, typographie et maquette), leurs utilisations et leurs caractéristiques. Elle permet de donner une cohérence afin de donner à l'utilisateur des repères et lui faciliter sa visite.

2.2.3.1. Logotype (logo)

Est une représentation graphique qui sert à identifier de manière unique et immédiate une application. Concernant le logo de notre application mobile et web nous avons imaginé un logo qui respecte les principes de notre projet : efficacité et modernité.

Les figures 2.4 et 2.5 représentent les deux apparences du logo de notre projet, qui reflètent la thématique de ce dernier.

FIGURE 2.4 – L'apparence du logo dans l'application mobile.

FIGURE 2.5 – L'apparence du logo dans l'application web.

Concernant le logo, nous avons combiné deux types :

- Le monogramme : L'initiale choisie est celle du mot Easy, qui est la lettre E.
- Le logotype : Nous avons choisi de laisser le mot BNA (pour déterminer le domaine de l' application) en s'appuyant seulement sur sa typographie

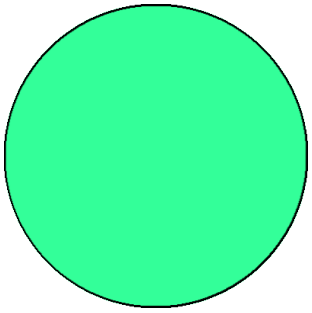
  En rassemblant le tout, nous obtenons un nom explicite et en rapport avec le thème et le nom de l'application.

2.2.3.2. Couleur du système

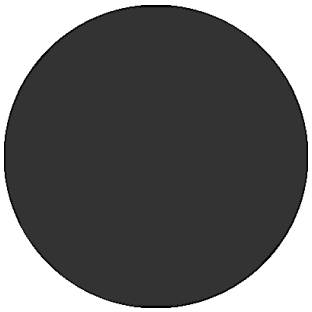
Couleur principale

Sont les couleurs qui dominent l'application, notre choix est porté sur ces deux couleurs pour attirer l'intention de l'utilisateur avec une certaine douceur et qu'ils sont liées à les couleur utilisée par la BNA.

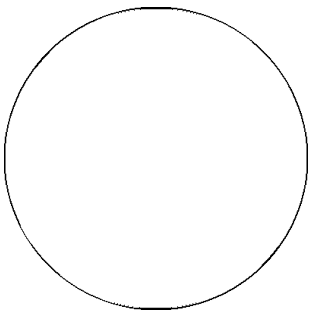
. **Vert [#00a885] :**
Une couleur qui a pour but d'attirer l'attention du visiteur de notre système.



. **Noir [#454545] :**
Une couleur qui reflète l'élégance et la simplicité.



. **Blanc [#ffffff] :**
Une couleur mise en arrière plan pour faire ressortir les éléments placés sur les pages et les mettre en valeur.



2.2.3.3. Typographie

Typographie des titres

. **Police :** Roboto-regular.

. **Font size :** 1.25 em

Typographie des paragraphes

. **Police :** Sans-serif..

. **Font size :** 0.938 em.

2.2.3.4. Maquettes

Le maquettage est une méthode de conception d'interface qui permet de proposer des interfaces conformes aux attentes et besoins de l'utilisateur. Elle permet également à la BNA de s'assurer que les besoins de l'utilisateur sont adaptés ou non au projet.

La maquette ci-dessous représente l'interface de la page d'accueil de notre application mobile.

FIGURE 2.6 – Maquette de la page d'accueil «application mobile»

La maquette ci-dessous représente l'interface de la page d'accueil de notre application web.

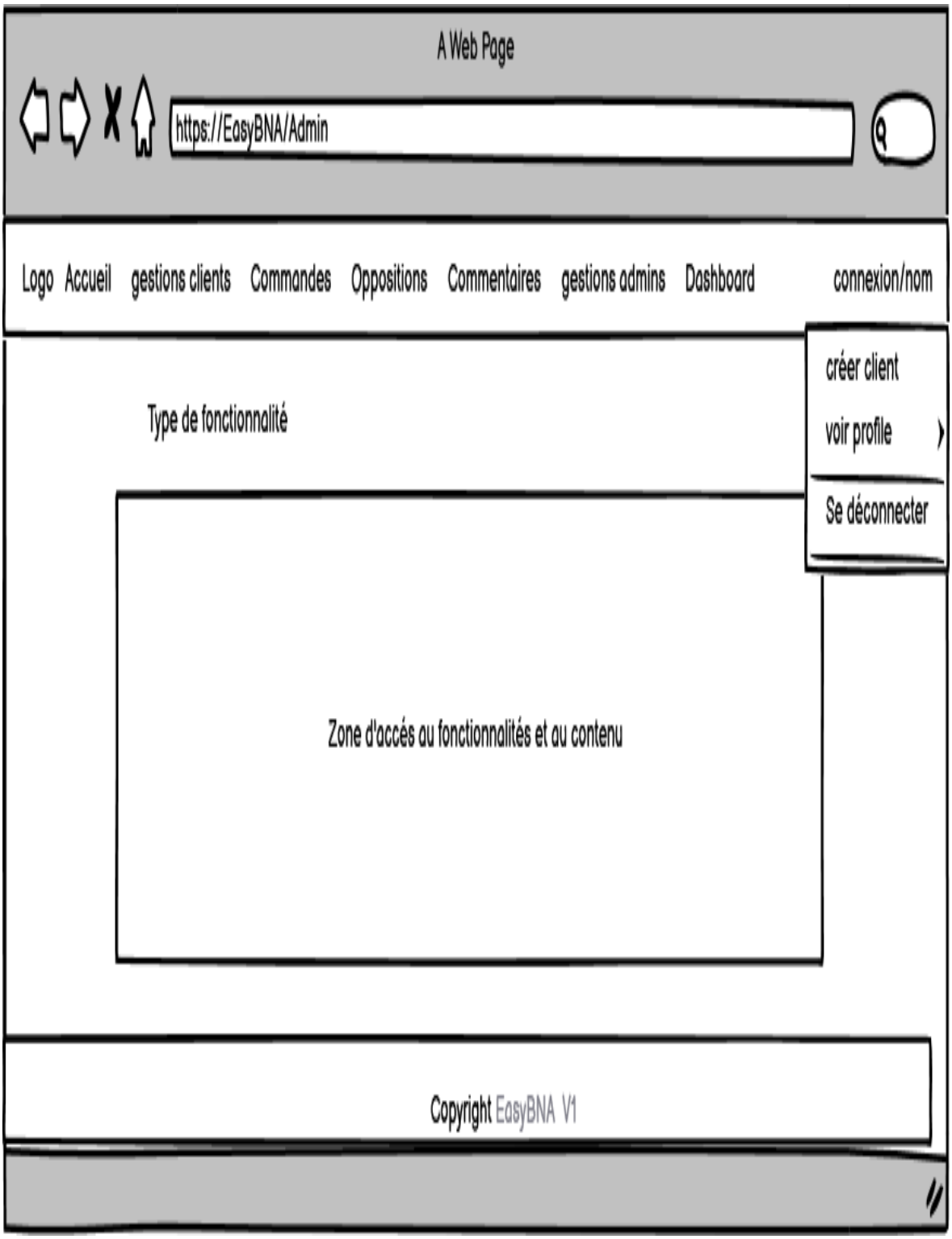


FIGURE 2.7 – Maquette de la page d'accueil «application web»

2.3. Spécification des besoins

C'est une phase décisive du processus de développement d'une application. Elle permet d'identifier les acteurs, de formaliser les besoins fonctionnels et non fonctionnels, et de déduire les différents cas d'utilisations à partir des besoins fonctionnels.

2.3.1. Identification des besoins

Besoins fonctionnels : L'application doit répondre aux besoins fonctionnels qui sont :

Coté Client :

- Consulter les informations personnelles du client.
- Consulter les informations bancaires du client.
- Permettre aux clients de connaître les différents crédits offerts par sa banque :
  - Crédits immobiliers ;
  - Crédits automobiles ;
  - Crédits confort ;
  - Crédits financement de l'investissement.
- Permettre aux clients de simuler les différents types de crédits ;
- Effectuer des transactions bancaires ;
- Consulter l'historique des transactions bancaires ;
- Localiser les différents emplacements des agences de notre banque ;
- Commande de chéquier et de carte bancaires ;
- Permettre aux clients d'effectuer une opposition sur sa carte bancaire ou chéquier ;
- Recevoir des notifications mobiles ;
- Faire un commentaire.

Coté Administrateur :

- L'administrateur principal peut ajouter, modifier, supprimer un administrateur simple ;
- Permettre la gestion des clients de la banque ;
- Consulter la liste des oppositions de cartes bancaires ou chéquiers ;

- L'administrateur peut supprimer un commentaire ;
- L'administrateur peut répondre à une commande de carte bancaire ou chéquier ;

Besoins non fonctionnels : L'application doit remplir des critères non fonctionnels comme :

- **Graphisme** : Les principales couleurs utilisées sont le gris, le vert clair, le blanc et ceci parce qu'elles n'agressent pas les yeux ;
- **Ergonomie** : L'application doit offrir une interface simple et facile d'utilisation ;
- **La fiabilité** : L'utilisateur devra recevoir les alertes programmées même si l'application cesse de fonctionner ;
- **L'utilisabilité** : L'utilisateur doit pouvoir maîtriser le fonctionnement de l'application facilement et rapidement ;
- **Performance** : On est amené à vérifier notre comptes plusieurs fois par jour. L'application doit être rapidement accessible pour ne pas contrarier l'utilisateur .
- **Sécurité** : La sécurité est assurée par la confidentialité des données des clients à travers l'authentification et le chiffrement des données.

2.3.2. Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation sont des diagrammes utilisés pour donner une vision globale du comportement fonctionnel d'une application. Ils permettent de recueillir, d'analyser, et d'organiser les besoins. Il s'agit donc de la première étape UML d'analyse d'un système [**12**]. Les éléments d'un diagramme de cas d'utilisation sont :

Acteur : Un acteur est l'idéalisation d'un rôle joué par une personne externe, un processus ou une chose qui interagit avec un système. On le représente par un petit bonhomme avec son nom inscrit en dessous.

Cas d'utilisation : Est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement, et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise, donc, un service rendu par le système, sans imposer le mode de réalisation de ce service. On le représente par une ellipse contenant le nom du cas.

Relations d'association : Une relation d'association est un chemin de communication entre un acteur et un cas d'utilisation et est représenté par un trait continu.

2.3.2.1. Identification des acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.[**12**]

Dans le cas du service bancaire, on a quatre acteurs : l'administrateur principal, l'administrateur simple et le client et enfin le visteur.

Visiteur : Le visiteur peut visualiser tous les services bancaire, mais ne peut accéder aux services reservé aux clients que s'il devient client, c'est-à-dire que son compte sera confirmé dans la base de données de la BNA ;

Client : Le client peut accéder à tous les services de l'application bancaire ;

Administrateur simple : L'administrateur va gérer les services clientèle, commande, opposition, commentaire et chacun des services disposent d'un ou plusieurs administrateurs. Les administrateurs vont gérer ce service en utilisant une application web conçu pour cela.

Administrateur Principal : Gestion des administrateurs simples.

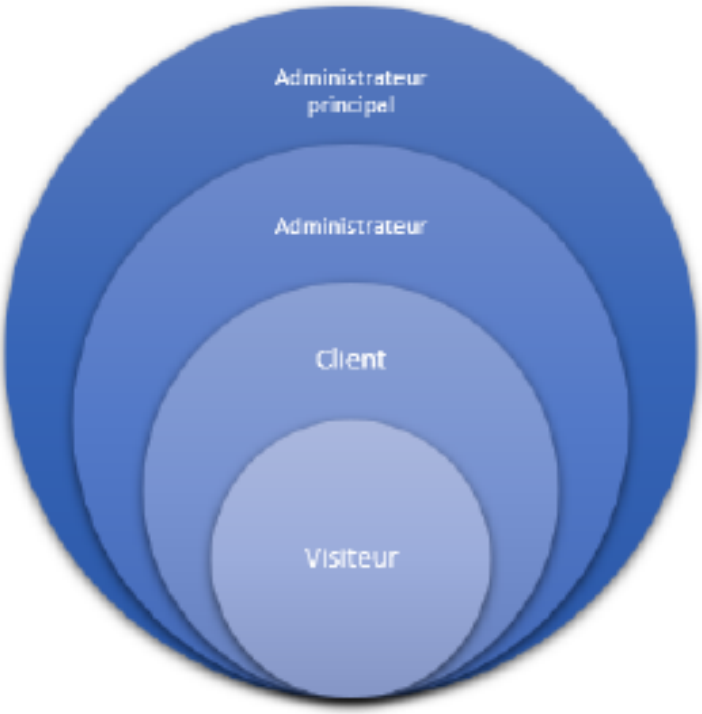


FIGURE 2.8 – Figure illustrant la gestion des droits

Dans ce qui suit, nous décrivons le diagramme de cas d'utilisation global du service bancaire.

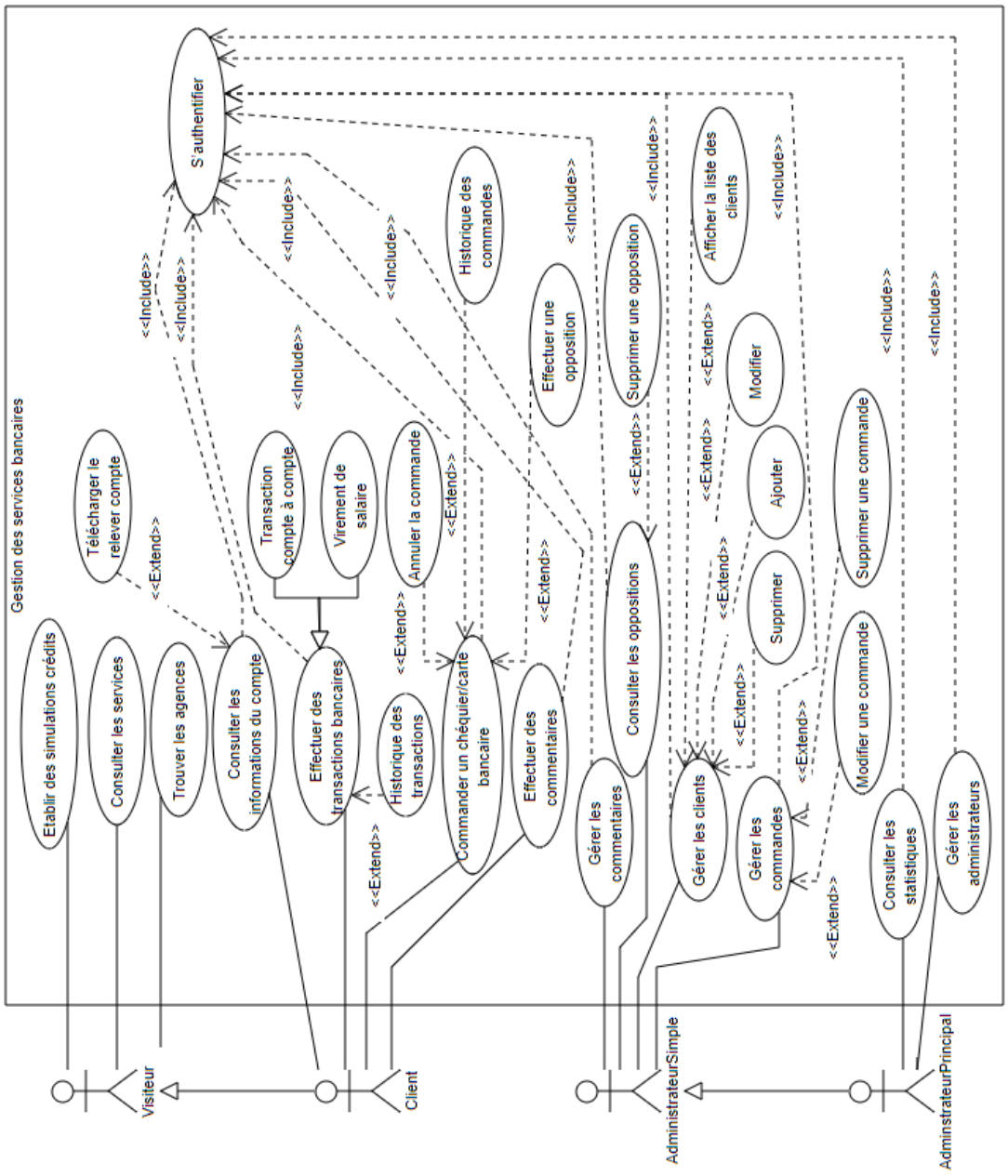


FIGURE 2.9 – Diagramme de cas d'utilisation général.

Dans le but de mieux cerner les fonctionnalités propres à chaque acteur je vais vous présenter ci-dessous les diagrammes des cas d'utilisations de ces derniers.

Diagramme de cas d'utilisation(Administrateur principal) Ce diagramme ci-dessous décrit les fonctionnalités relatives à l'administrateur principal qui hérite des fonctionnalités de l'administrateur simple.

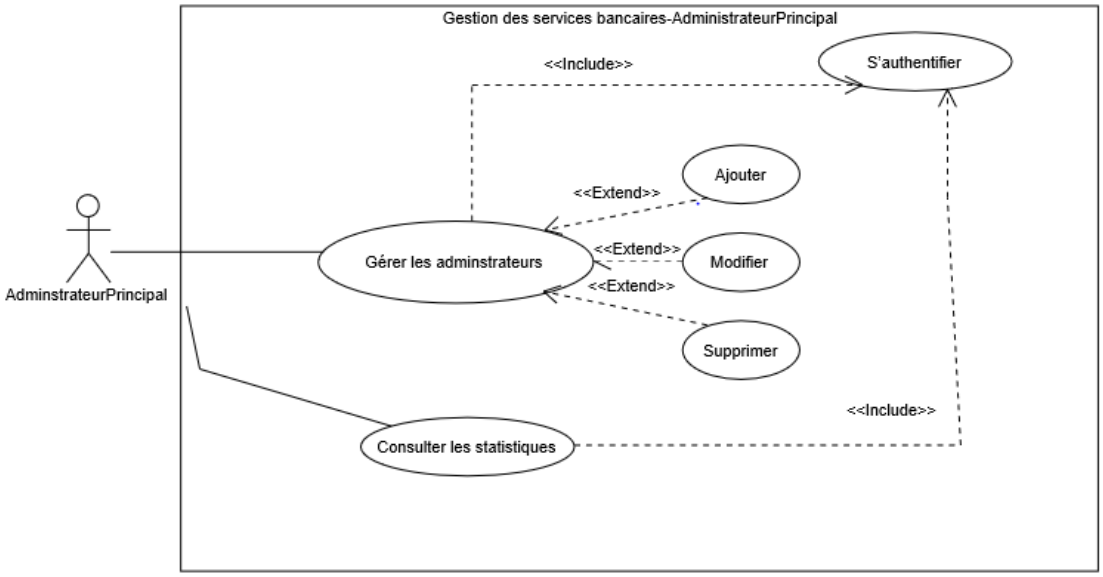


FIGURE 2.10 – Fonctionnalités relatives à l'administrateur principal

Diagramme de cas d'utilisation (administrateur simple) Ce diagramme ci-dessous décrit les fonctionnalités relatives à l'administrateur simple.

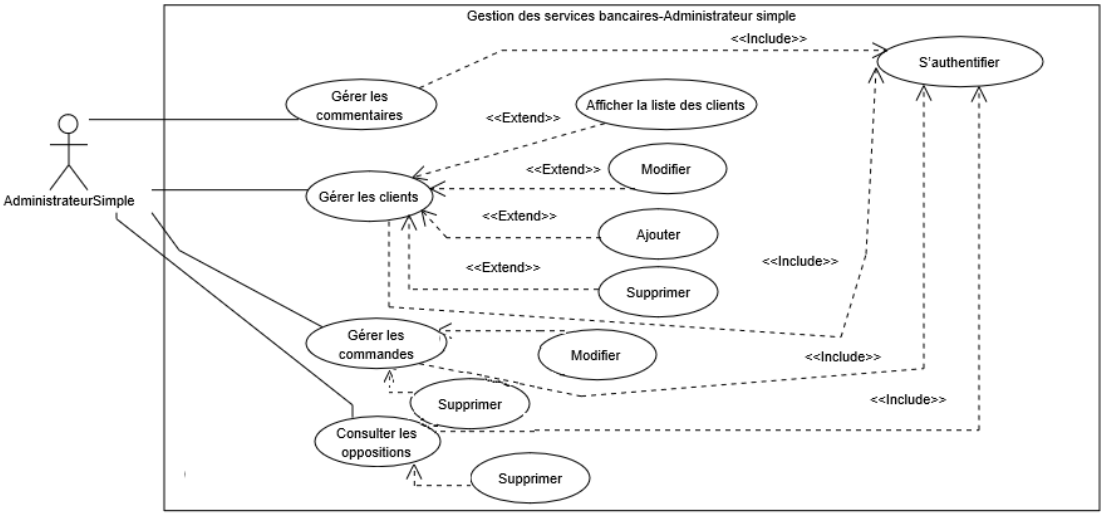


FIGURE 2.11 – Fonctionnalités relatives à l'administrateur simple

Diagramme de cas d'utilisation (visiteur) Ce diagramme ci-dessous décrit les fonctionnalités relatives au visiteur.

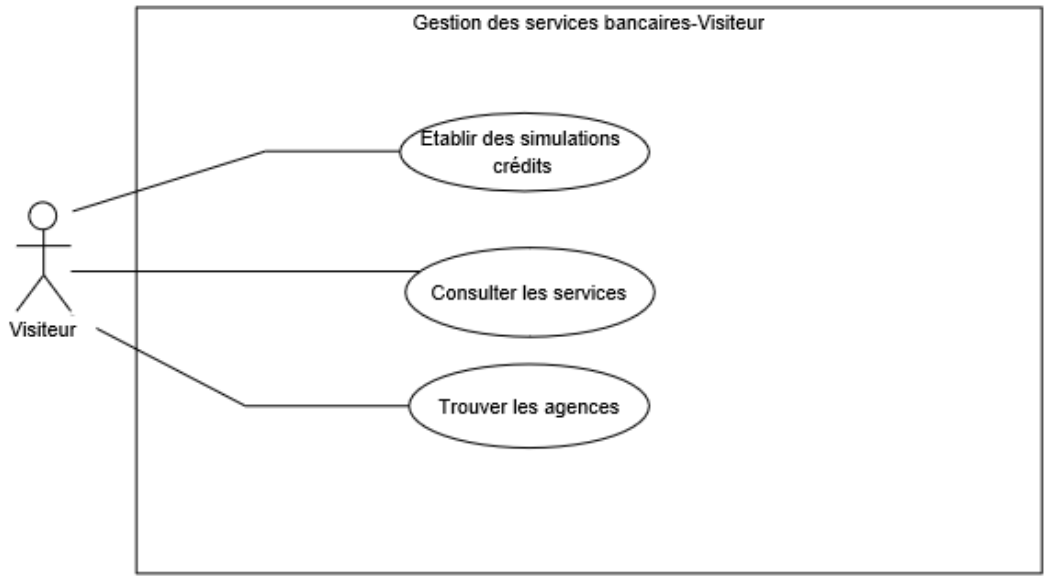


FIGURE 2.12 – Fonctionnalités relatives au visiteur

Diagramme de cas d'utilisation (Client) Ce diagramme ci-dessous décrit les fonctionnalités relatives au client en plus des fonctionnalités visiteur.

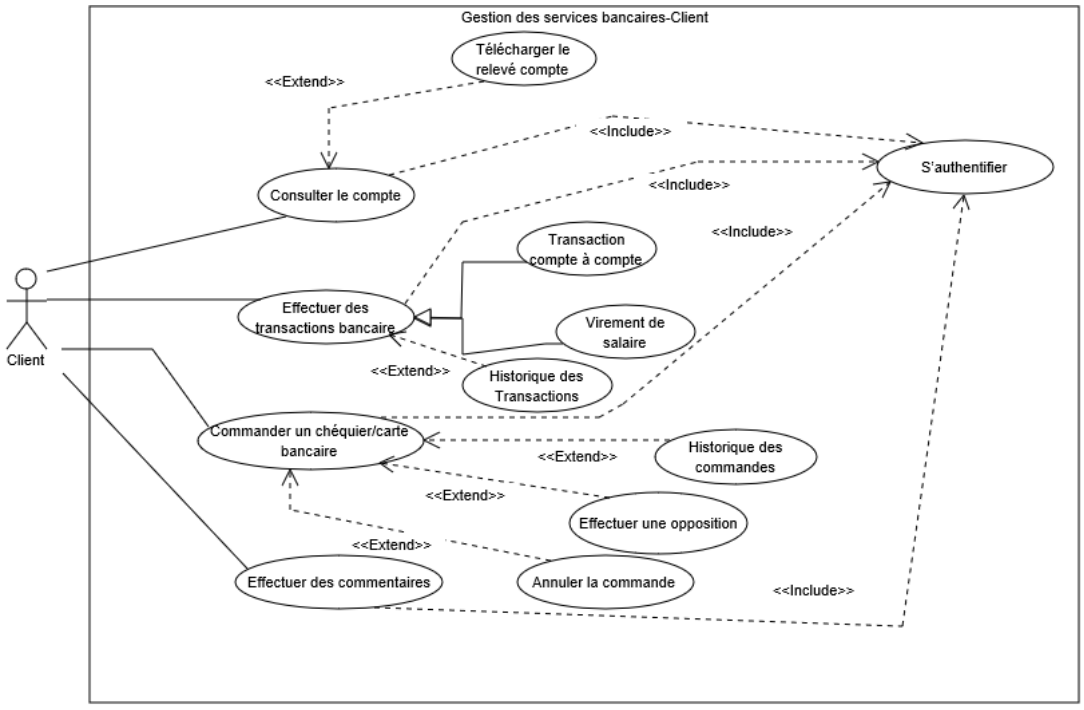


FIGURE 2.13 – Fonctionnalités relatives au client

Afin de mieux éclaircir les diagrammes de cas d'utilisation nous allons faire une description textuelle des cas d'utilisation.

Dans le tableau ci-dessous sont décrits les différents cas d'utilisation associés à

leur(s) acteur(s) :

N	Cas d'utilisation	Acteur
1	Authentification.	Client/Administrateur
2	Etablir des simulations crédits.	Client/Visiteur
3	Consulter les services.	
4	Trouver les agences.	
5	Consulter le compte.	Client
6	Effectuer des transactions bancaires.	
7	Commander un chéquier et carte bancaire.	
8	Effectuer des commentaires.	
9	Gérer les commentaires.	Administrateur
10	Gérer les clients.	Admin-S/Admin-P
11	Gérer les commandes.	
12	Consulter les oppositions	
13	Gérer les administrateurs Simple	Admin-P
14	Consulter les statistiques	

TABLE 2.1 – Cas d'utilisation du système à réaliser

2.3.2.2. Description textuelle de quelques cas d'utilisation

Dans le tableau2.2 sont décrits les cas d'utilisation que nous avons choisi pour la suite de la modélisation.

Cas d'utilisation	Acteur	Description
Authentification.	Client, Administrateur, Administrateur principale.	Permet aux différents utilisateurs de s'authentifier à notre système.
Consulter compte.	Client.	Permet au client de consulter son compte et voir son solde sur l'application et cela après s'être authentifié.
Transaction bancaire.	Client.	Permet au client de faire des transactions de type compte à compte ou un versement salaire et la possibilité de voir l'historique de ses transactions et aussi voir le montant de son solde après la transaction.
Simulation du crédit.	Client,visiteur.	Permet au client et au visiteur de consulter tous les services de crédit offert, et aussi une simulation sur les différents crédits de BNA.
Commander un chéquier/Carte.	Client.	Permet au client de faire une commande de carte bancaire ou chéquier et permettre de faire des oppositions. en fonction des limites approprier à l'application.
Gérer les clients.	Administrateur simple,Administrateur principal.	Permet à l'administrateur de consulter, d'ajouter, modifier, supprimer les comptes des clients.
Gérer les administrateurs.	Administrateur principal.	Permet à l'administrateur principal de consulter, d'ajouter, modifier, supprimer les comptes des autres administrateurs.

TABLE 2.2 – Description des cas d'utilisation choisis pour la suite de la modélisation.

2.4. Conclusion

Dans ce chapitre, nous avons entamé la présentation de notre système, le processus UP et le langage de modélisation UML qui nous a permis de traduire nos besoins fonctionnels en cas d'utilisations que nous allons détailler dans le chapitre suivant.

Chapitre 3

Analyse et Conception

Introduction

Après avoir tracé les grandes lignes de la phase de spécification des besoins, nous allons maintenant mettre l'accent sur deux phases importantes du processus de développement de notre systéme qui sont l'analyse des besoins et la conception. dans ce chapitre, nous présenterons les différents diagrammes séquence, d'interactions associées au cas d'utilisation. Par la suite on va définir le diagramme de classe à réaliser, celui associé à notre système en précisant les classes et les relations existantes entre elles. Enfin, à partir de cette dernière et en appliquant les règles de passage de diagramme de classe vers le modèle relationnel nous allons aboutir à notre MLD ce qui nous servira dans l'implémentation de notre base de données et définir un dictionnaire de données.

3.1. Analyse des besions

Comme dans tout projet informatique nous devons passer par l'analyse des besoins où nous présentons le diagramme de séquence système et le pattern MVC.

3.1.1. Architecture MVC

C'est un modèle d'architecture qui cherche à séparer nettement les couches de présentation (UI : User Interface), métier (BLL : Business Logic Layer) et d'accès aux données (DAL : Data Access Layer). Le but étant d'avoir une dépendance minimale

entre les différentes couches de l'application, ainsi les modifications effectuées sur n'importe quelle couche de l'application n'affectent pas les autres couches[**29**].

— **Couche Modèle :** La couche modèle représente la partie de l'application qui exécute la logique métier. Cela signifie qu'elle est responsable de récupérer les données, de les convertir selon des concepts chargés de sens pour l'application, tel que le traitement, la validation, l'association et beaucoup d'autres tâches concernant la manipulation des données. À première vue, l'objet « Modèle » peut être vu comme la première couche d'interaction avec n'importe quelle base de données de l'application. Mais plus globalement, il fait partie des concepts majeurs autour desquels l'application est exécutée[**29**].

— **Couche Vue :** La vue retourne une présentation des données venant du modèle. Etant séparée par les objets « modèles », elle est responsable de l'utilisation des informations dont elle dispose pour produire une interface de présentation de l'application. Par exemple, de la même manière que la couche modèle retourne un ensemble de données, la vue utilise ces données pour fournir une page HTML ou un résultat XML formaté, la couche vue n'est pas seulement limitée au HTML/XML ou à la représentation en texte de données, elle peut aussi être utilisée pour offrir une grande variété de formats en fonction des besoins[**29**].

— **Couche Contrôleur :** Cette partie gère la logique du code qui prends des décisions. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP , Java. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès)[**29**].

La figure suivante schématise le rôle de chacun de ces éléments.

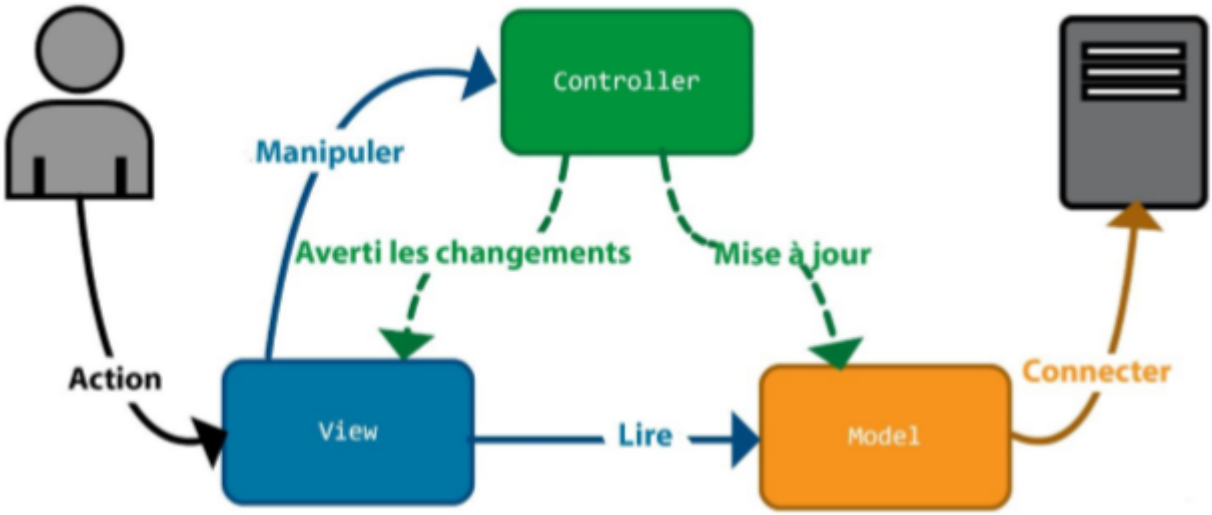


FIGURE 3.1 – Architecture du modèle MVC.[**29**]

3.1.2. Diagramme de séquence système

Ce diagramme permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets. Pour passer du diagramme de cas d'utilisation aux diagrammes d'interactions on doit passer par une étape intermédiaire : diagramme de séquence système où est décrit le scénario nominal du cas d'utilisation. Un diagramme des séquence système montre les interactions entre un acteur et le système (représenté en tant que boîte noire)[**13**].

3.1.3. Diagramme de séquence authentification

La figure 3.2 décrit l'enchaînement séquentiel des échanges entre un client et le système lors de l'authentification (connexion).

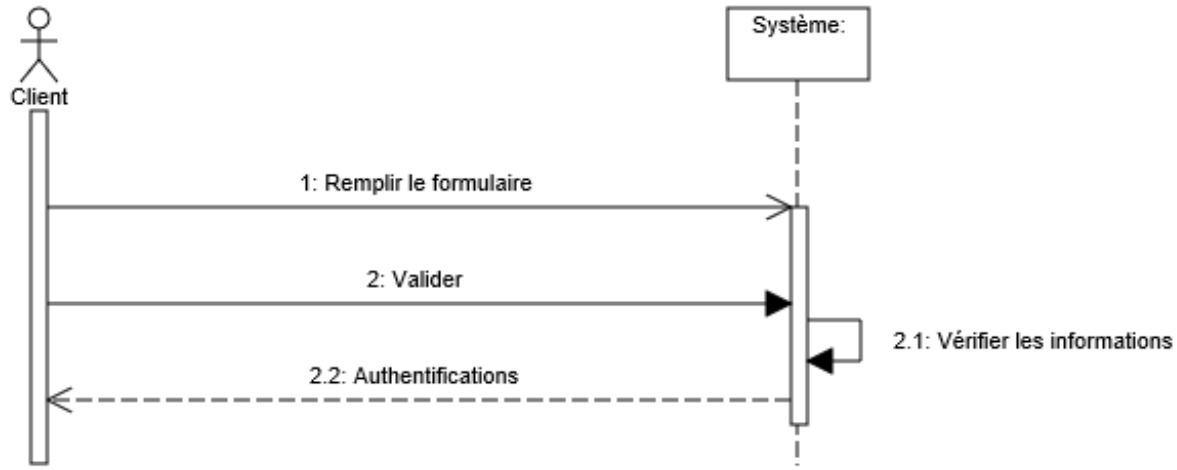


FIGURE 3.2 – Diagramme de séquence «Authentification ».

3.1.4. Diagramme de séquence simulation du crédit

La figure 3.3 décrit l'enchaînement séquentiel des échanges entre un client et le système lorsque ce dernier veut faire une simulation d'un crédit.

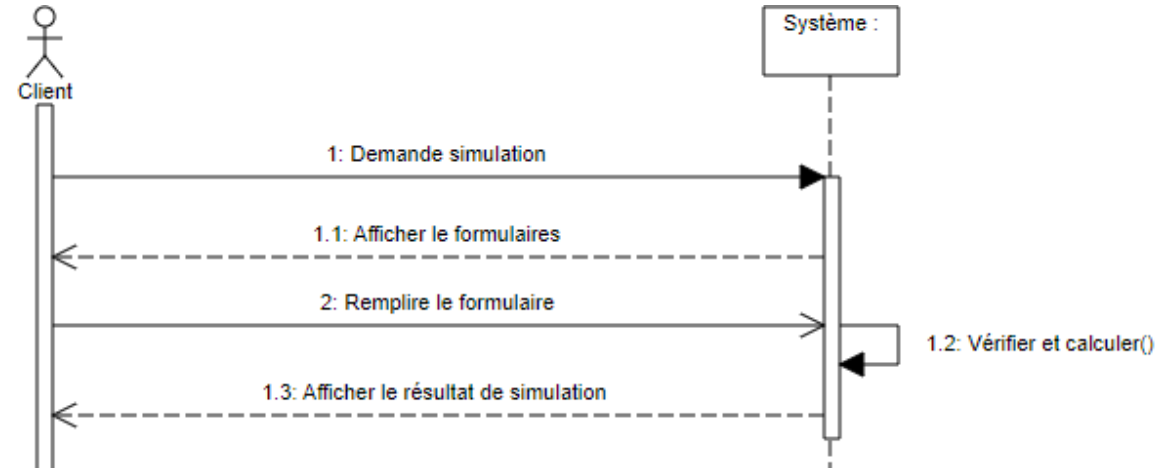


FIGURE 3.3 – Diagramme séquence « Simulation du crédit ».

3.1.5. Diagramme de séquence transacation

La figure 3.4 décrit l'enchaînement séquentiel des échanges entre un client et le système lorsque ce dernier veut faire une transaction.

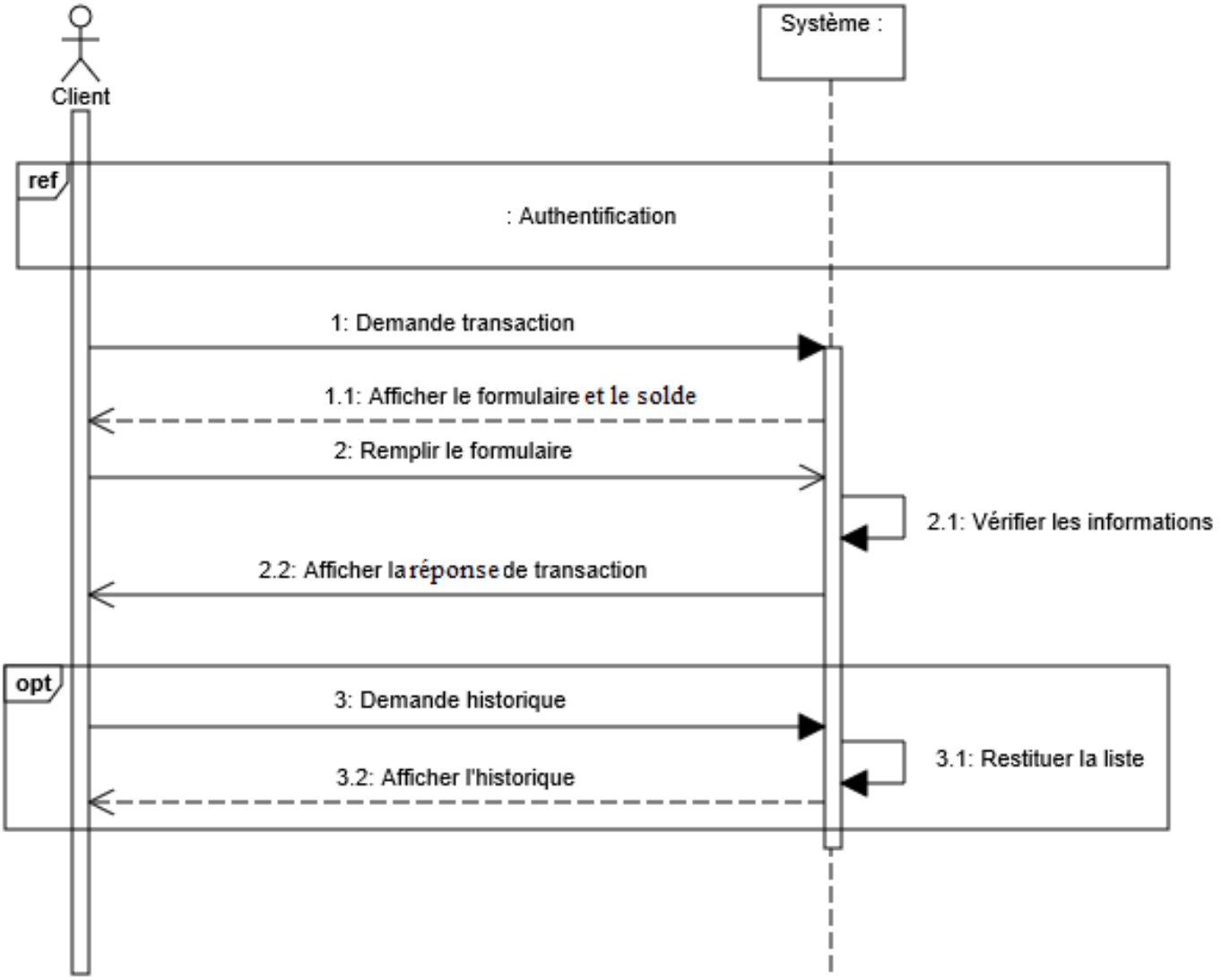


FIGURE 3.4 – Diagramme de séquence « Transacation ».

3.1.6. Diagramme de séquence commande de chéquier et carte bancaire

La figure 3.5 décrit l'enchaînement séquentiel des échanges entre un client et le système lorsque ce dernier veut faire une commande.

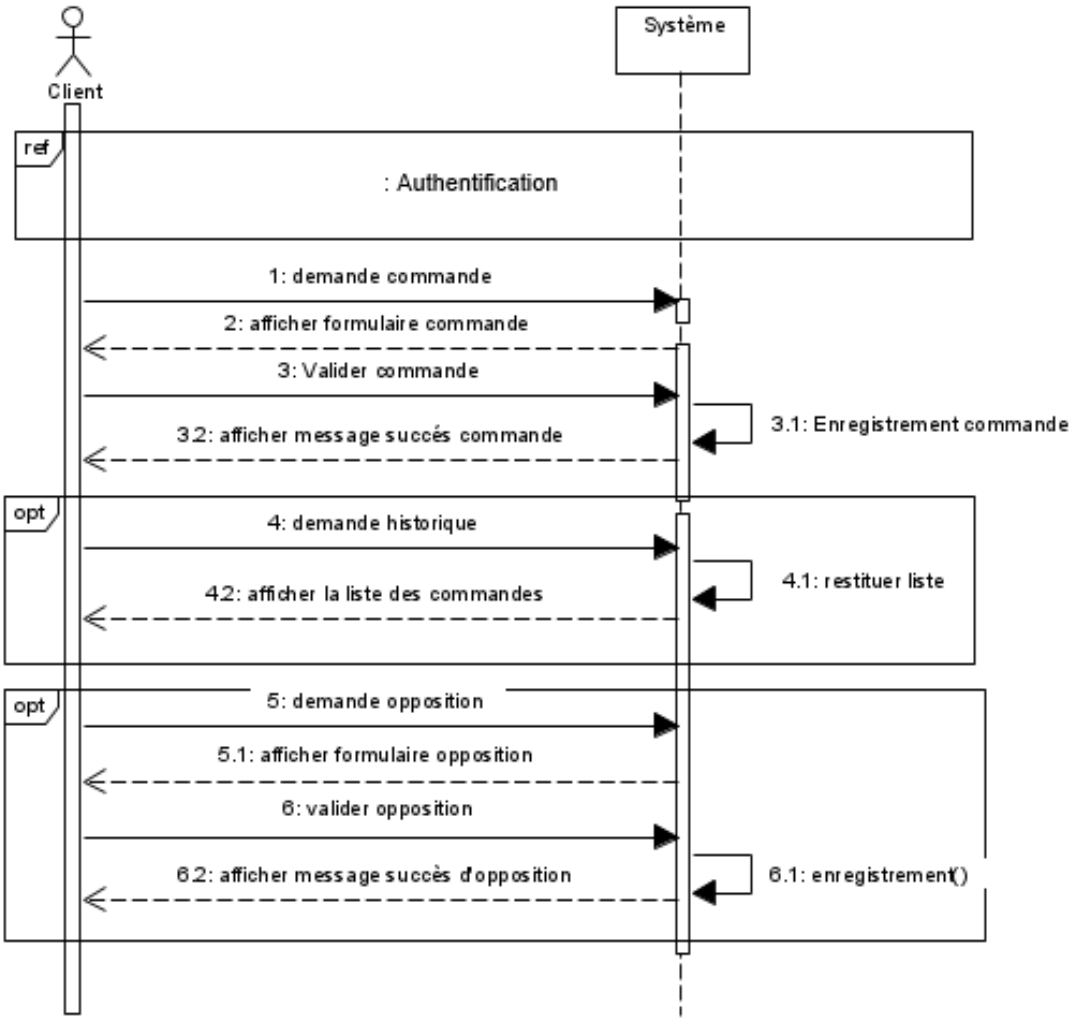


FIGURE 3.5 – Diagramme de séquence « Commande de chéquier et carte bancaire ».

3.1.7. Diagramme de séquence consulter le compte

La figure 3.6 décrit l'enchaînement séquentiel des échanges entre un client et le système lorsque ce dernier veut consulter les informations de son compte.

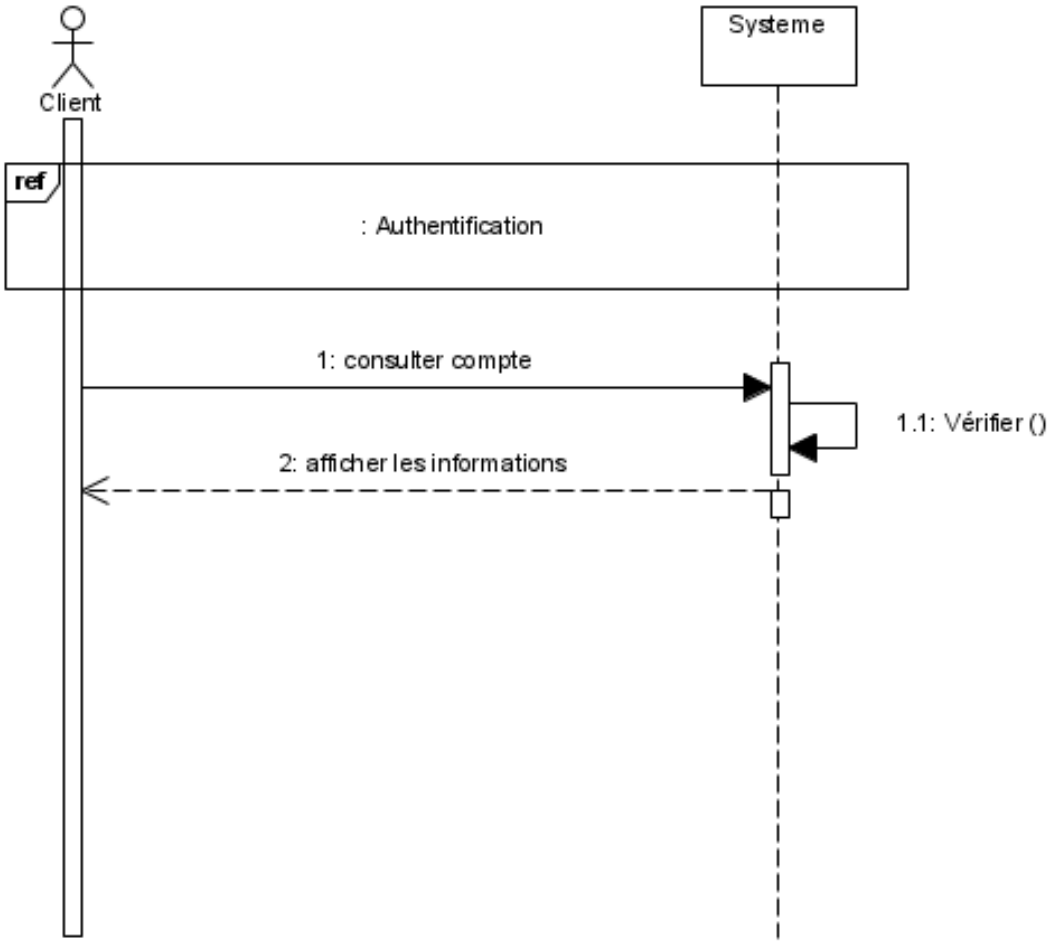


FIGURE 3.6 – Diagramme d'interaction « Consulter le compte ».

3.1.8. Diagramme de séquence gérer les clients

La figure 3.7 décrit l'enchaînement séquentiel des échanges entre un administrateur simple et le système lorsque ce dernier veut gérer les clients.

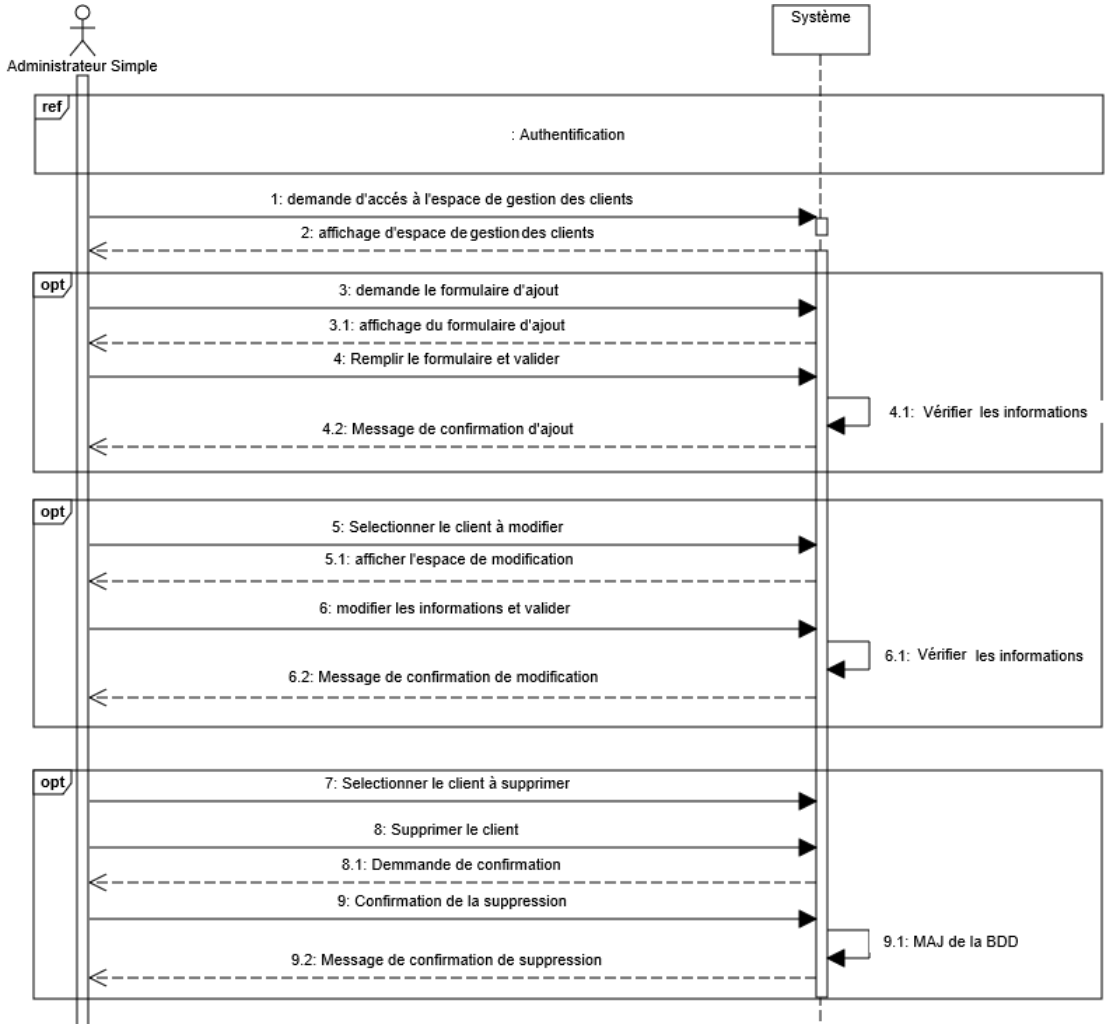


FIGURE 3.7 – Diagramme de séquence « Gérer les clients ».

3.1.9. Diagramme de séquence gérer les administrateurs

La figure 3.8 décrit l'enchaînement séquentiel des échanges entre un administrateur principale et le système lorsque ce dernier veut gérer les administrateurs simples.

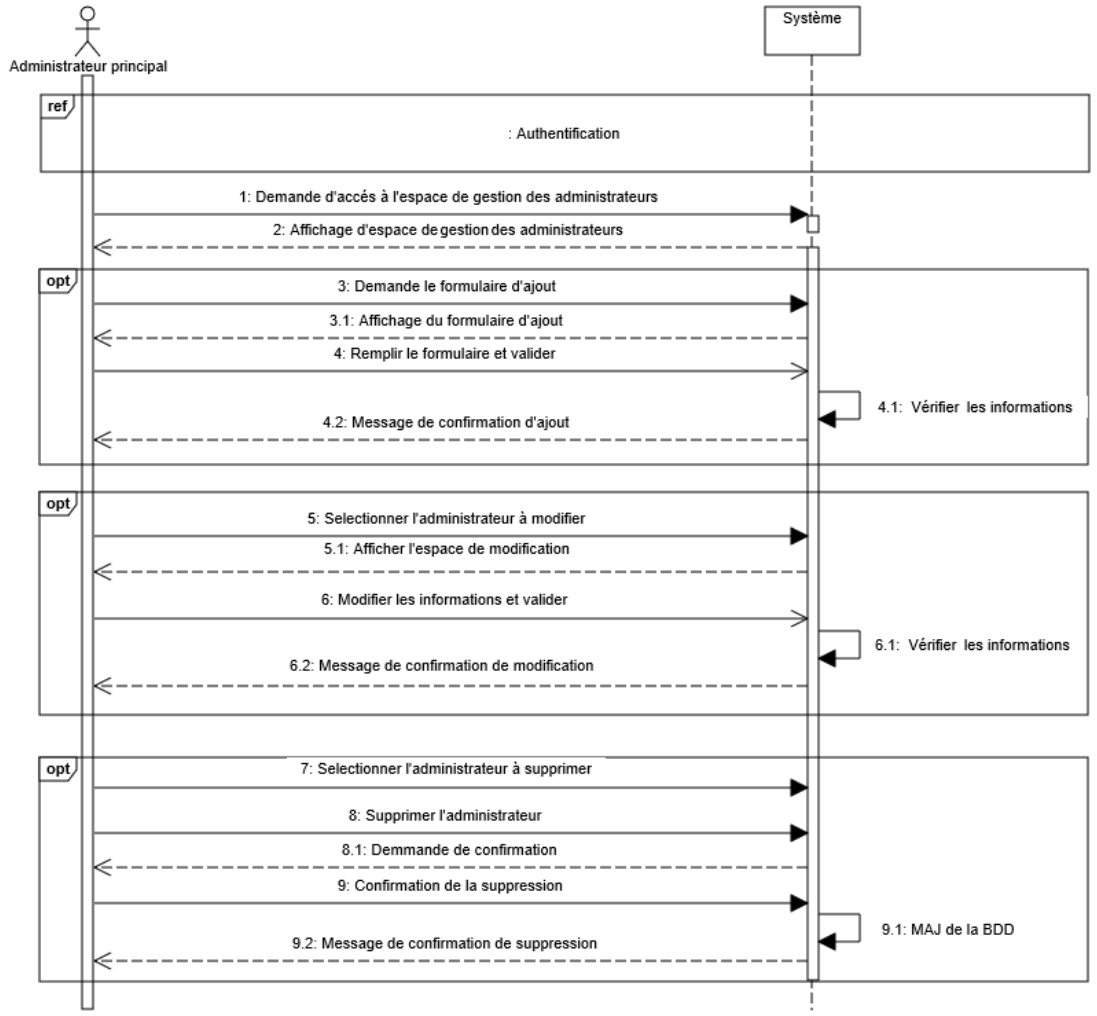


FIGURE 3.8 – Diagramme de séquence « Gérer les administrateurs ».

3.2. Conception

Nous présentons dans cette partie les diagrammes d'interactions et le diagramme de classe de notre système.

3.2.1. Diagramme d'interactions

Un diagramme d'interaction est un diagramme de séquence système détaillé où le système vu comme une boîte noire par un ensemble d'objets de classes différentes tout en respectant les règles suivantes :[**13**]

— Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.

— Les dialogues peuvent interagir avec les contrôles.

— Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.

— Les entités ne peuvent interagir qu'entre elles.

Le formalisme est donné dans l'exemple type présenté à la figure 3.9

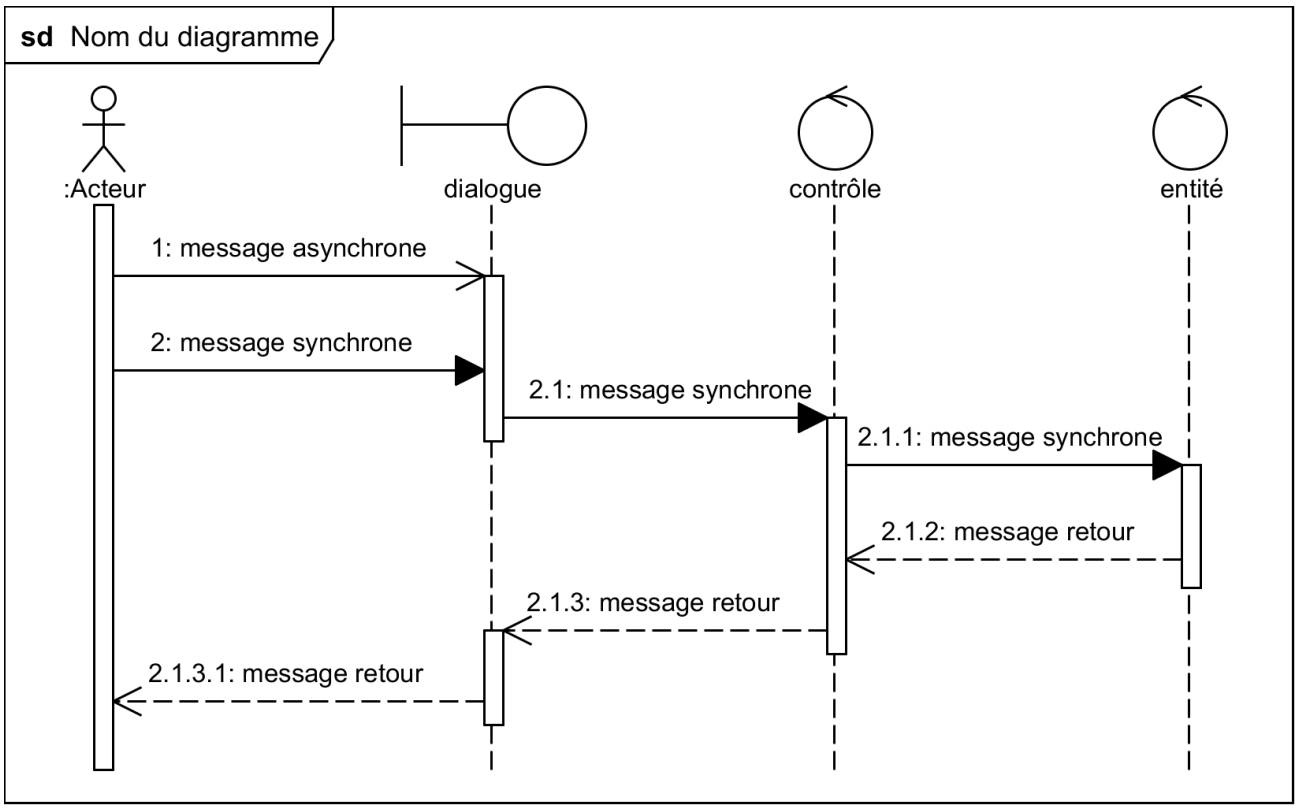


FIGURE 3.9 – Formalisme du diagramme d'interactions

3.2.2. Diagramme d'interactions authentification

La figure 3.10 décrit l'enchaînement des échanges entre l'utilisateur (administrateurs ou clients) et les différents objets participant au cas d'utilisation «authentifier».

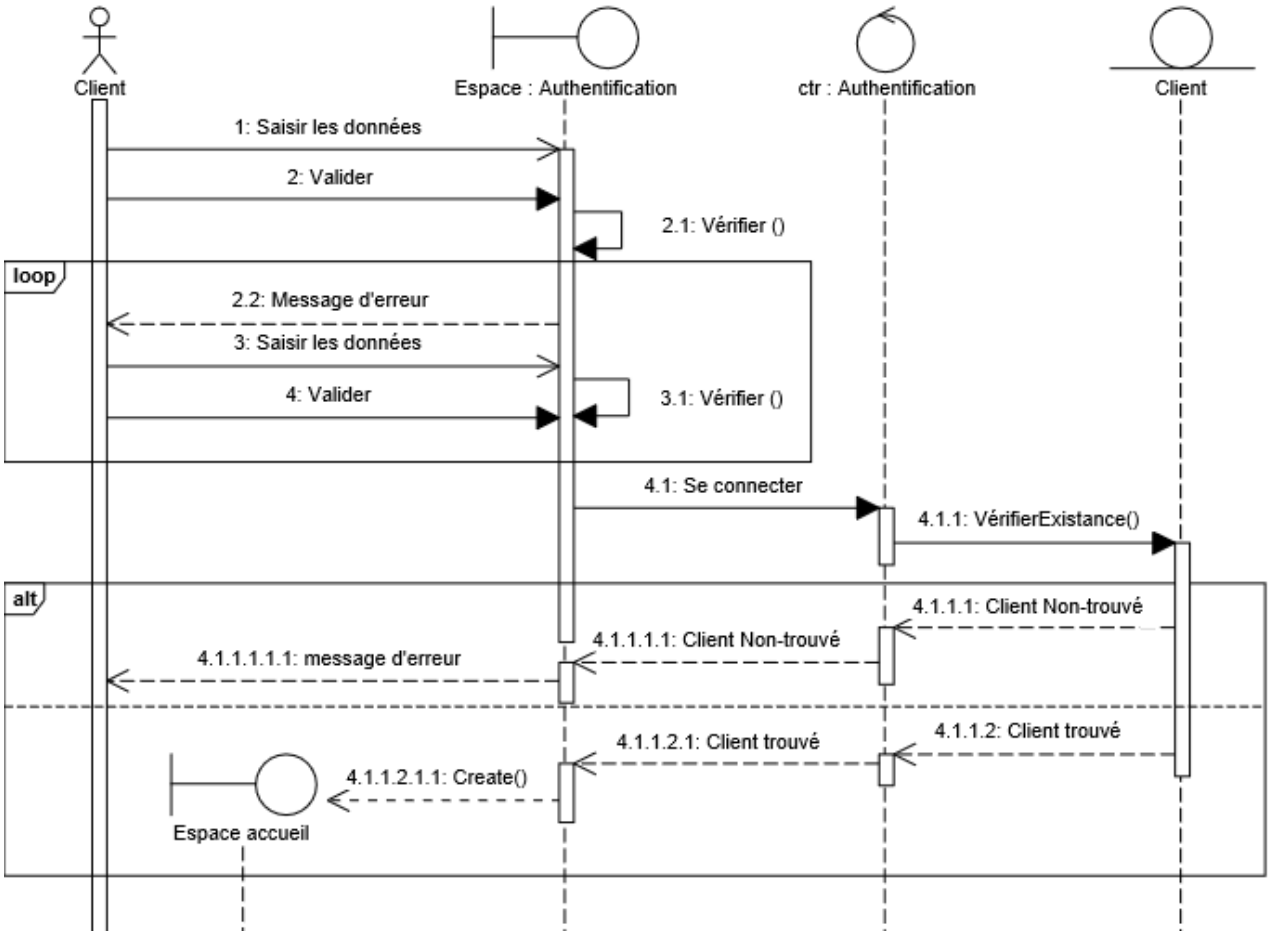


FIGURE 3.10 – Diagramme d'interactions «Authentification ».

3.2.3. Diagramme d'interactions simulation du crédit

La figure 3.11 décrit le diagramme d'interactions entre un client et les différents objets participant au cas d'utilisation «Etablir des Simulations crédits».

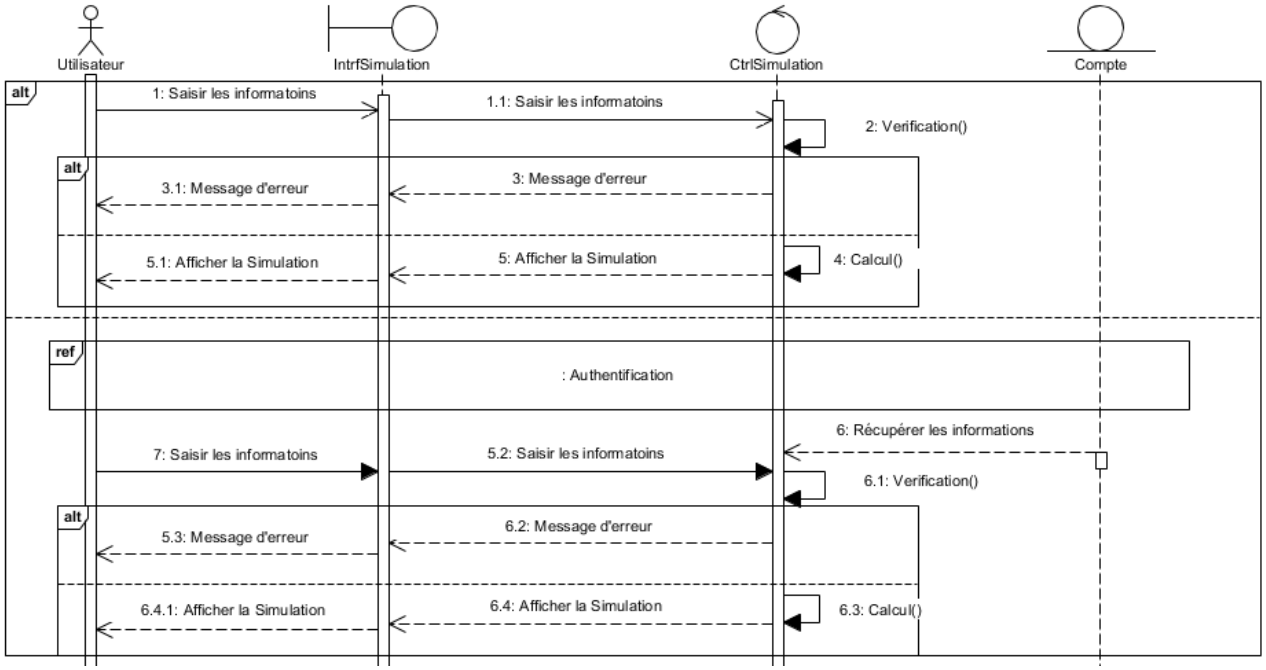


FIGURE 3.11 – Diagramme d'interactions « Simulation du crédit ».

3.2.4. Diagramme d'interactions transacation bancaire

La figure 3.12 décrit le diagramme d'interactions entre un client et les différents objets participant au cas d'utilisation «Effectuer des transactions bancaires».

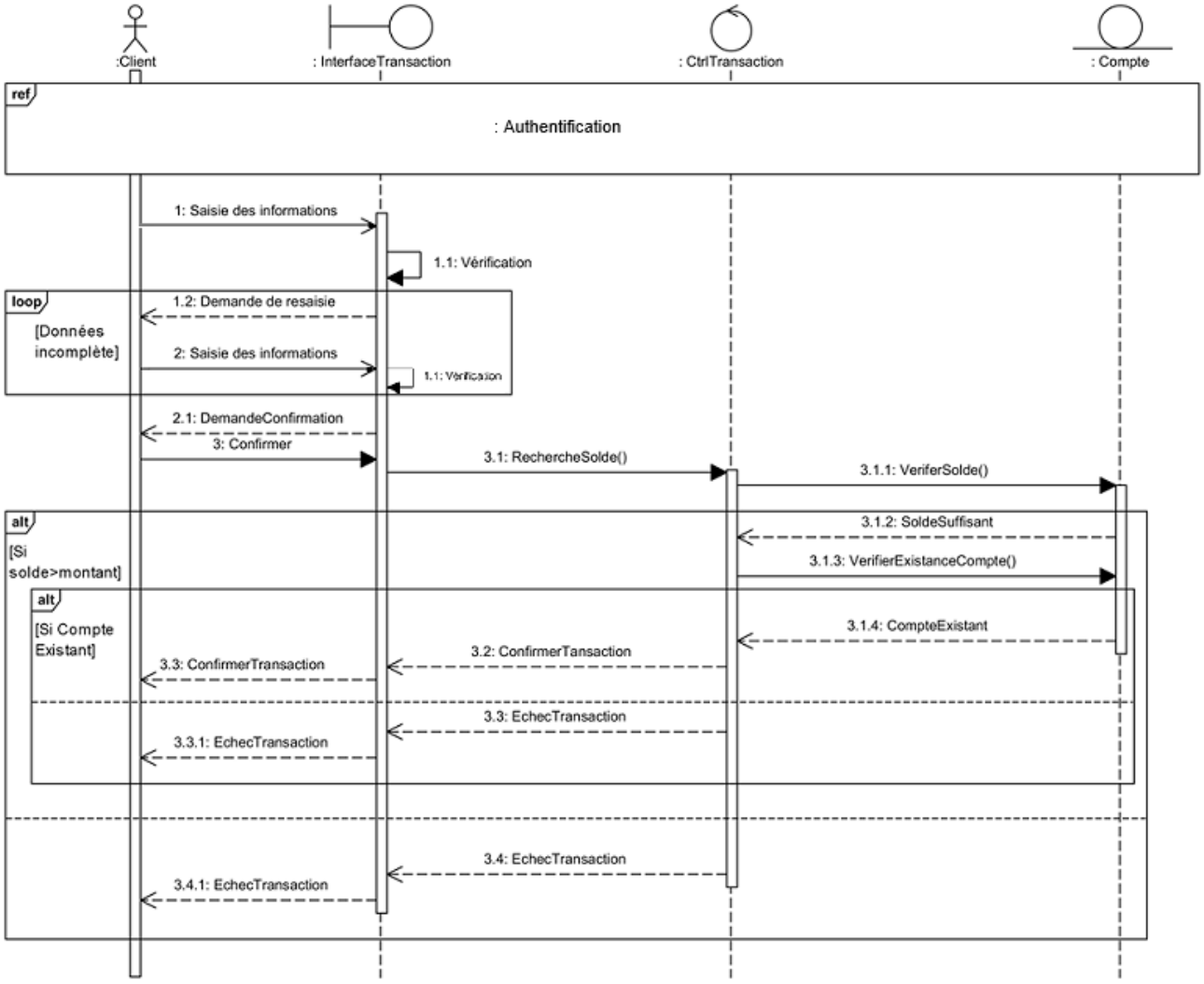


FIGURE 3.12 – Diagramme d'interactions « Transacation bancaire ».

3.2.5. Diagramme d'interactions commande de chéquier et carte bancaire

La figure 3.13 décrit le diagramme d'interactions entre un client et les différents objets participant au cas d'utilisation traité.

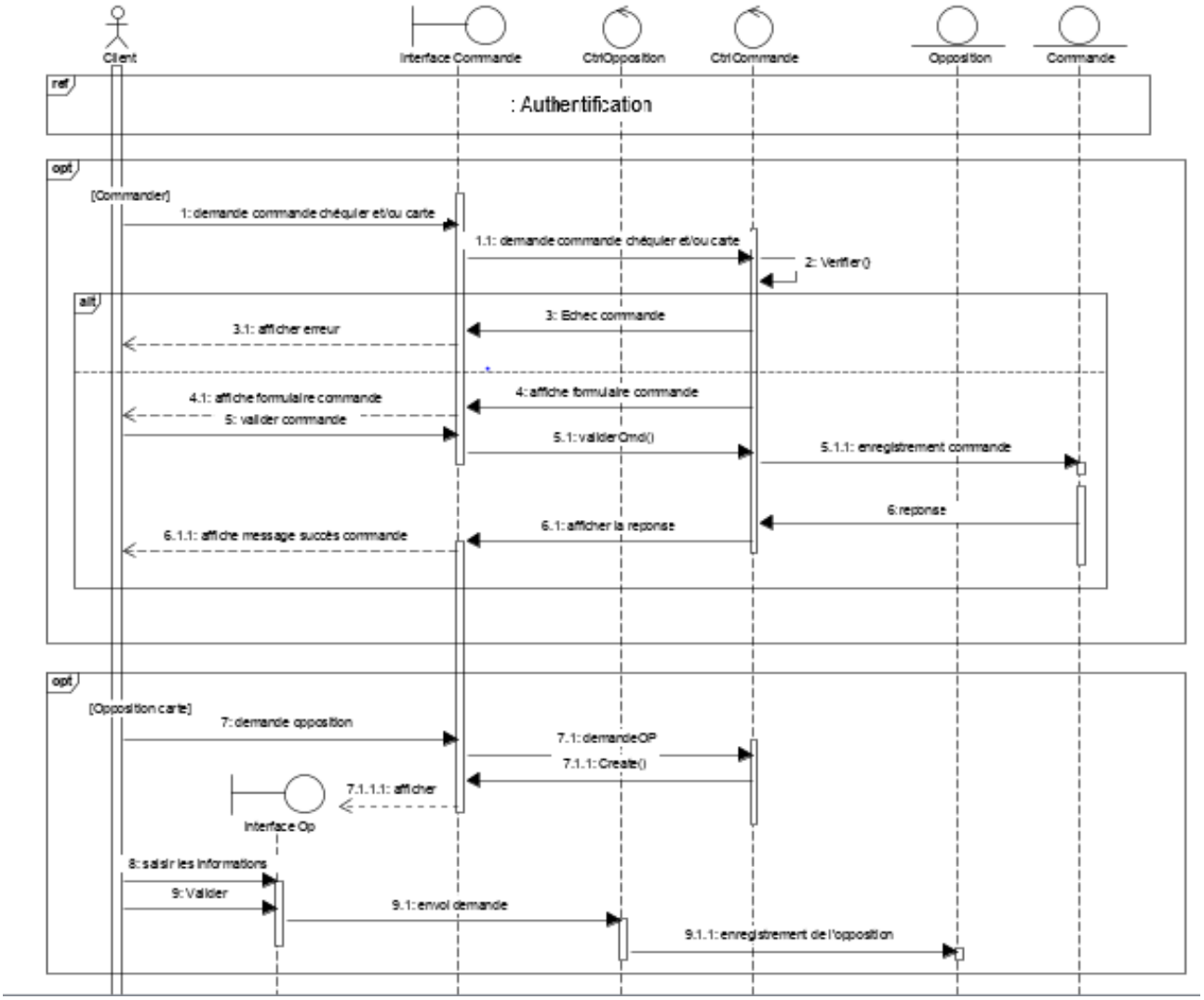


FIGURE 3.13 – Diagramme d'interactions « Commande de chéquier ou carte bancaire ».

3.2.6. Diagramme d'interactions consulter le compte

La figure 3.14 décrit le diagramme d'interactions entre un client et les différents objets participant au cas d'utilisation traité.

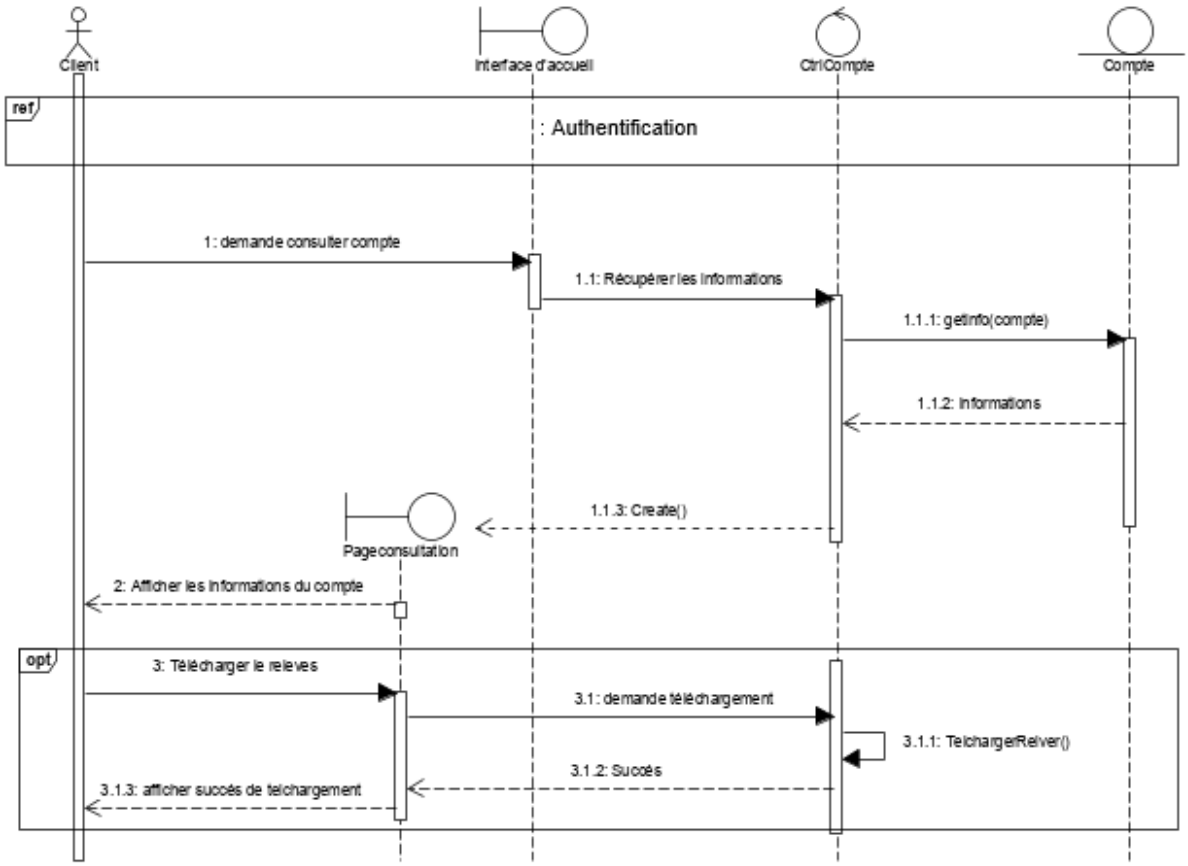


FIGURE 3.14 – Diagramme d'interactions « Consulter le compte ».

3.2.7. Diagramme d'interactions gérer les clients

La figure 3.15 décrit le diagramme d'interactions entre un administrateur simple et les différents objets participant au cas d'utilisation traité.

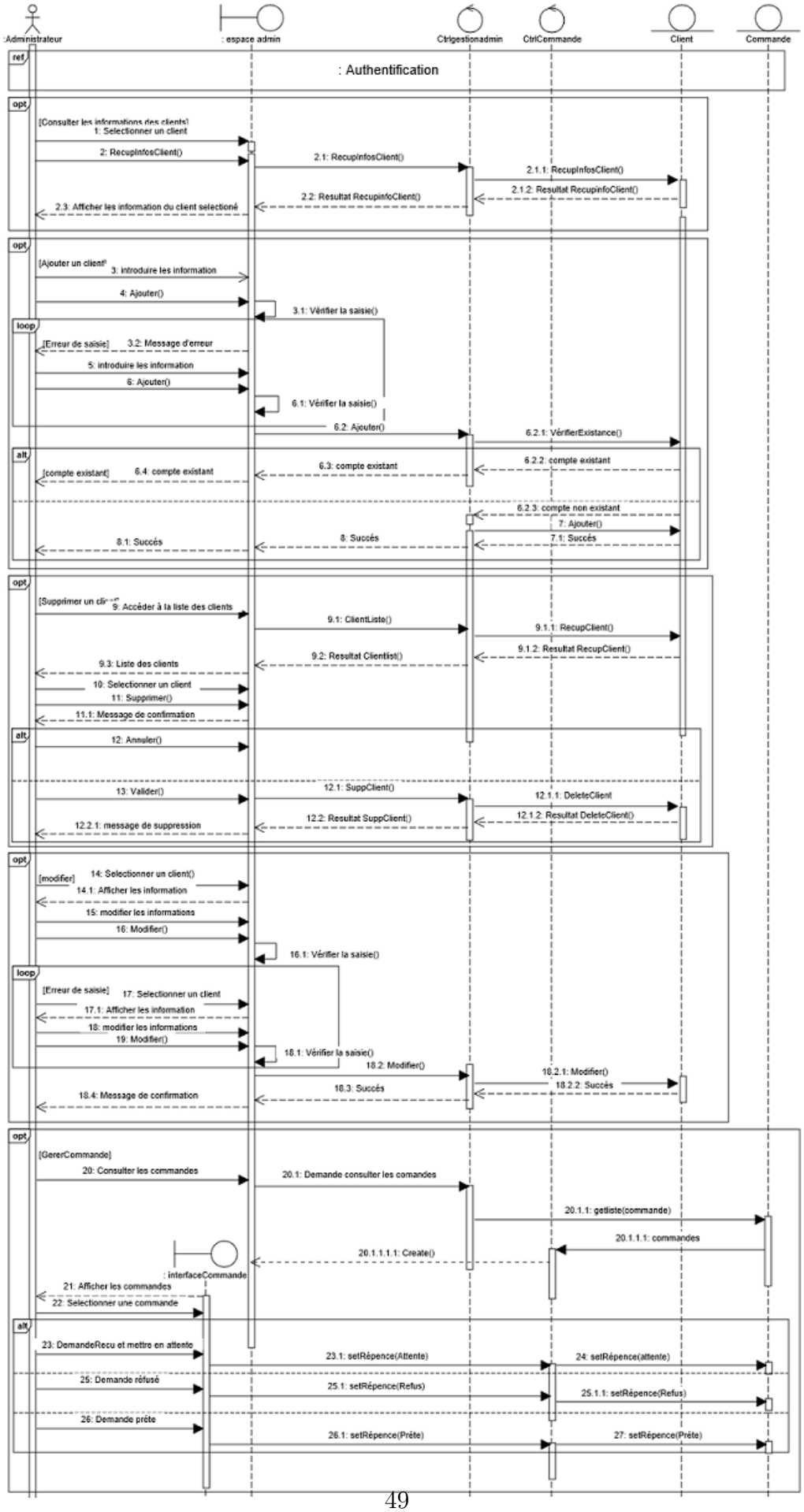


FIGURE 3.15 – Diagramme d'interactions « Gérer les clients ».

3.2.8. Diagramme d'interactions gérer les administrateurs

La figure 3.16 décrit le diagramme d'interactions entre un administrateur principale et les différents objets participant au cas d'utilisation traité.

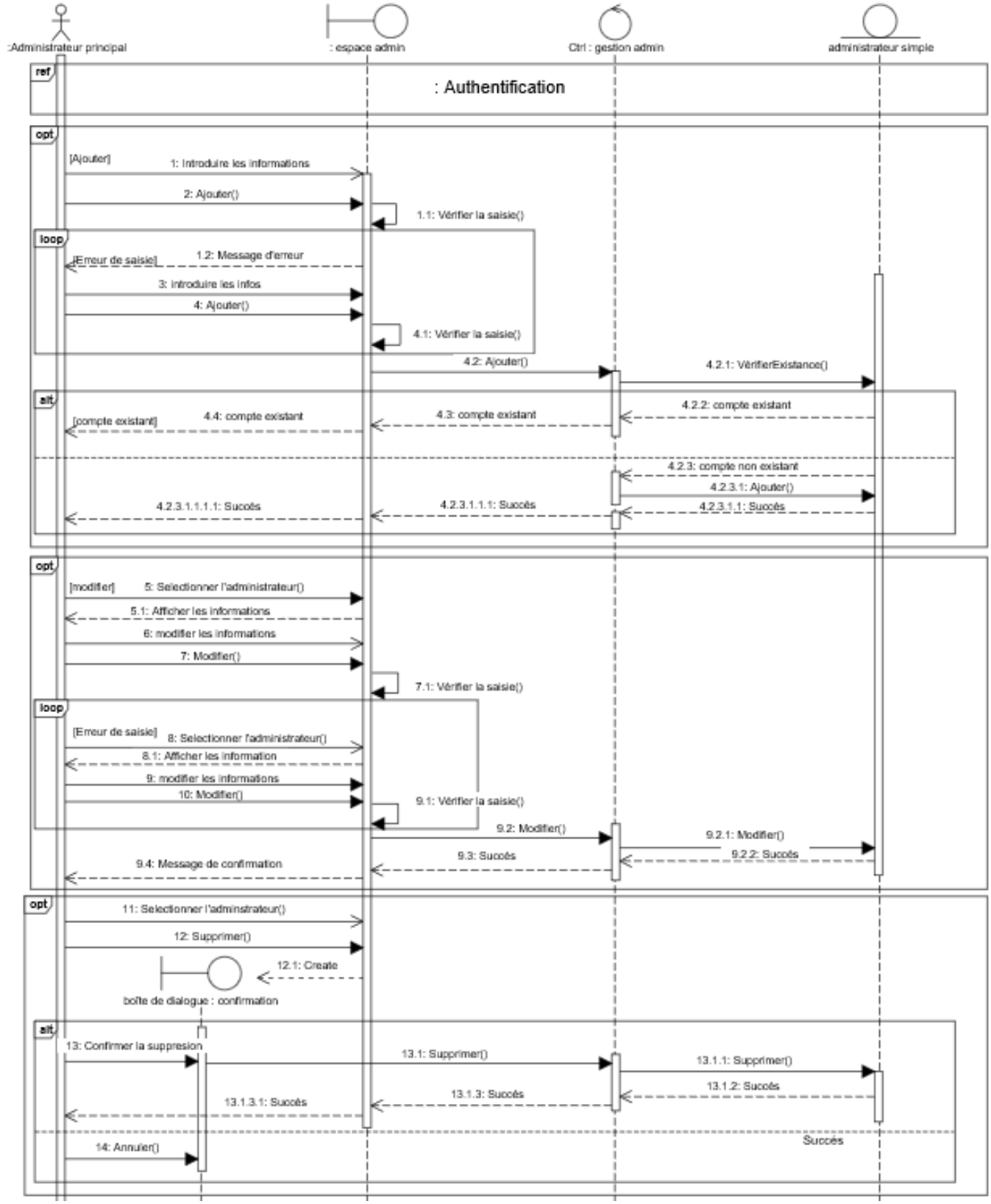


FIGURE 3.16 – Diagramme d'interactions « Gérer les administrateurs ».

3.2.9. Diagramme de classe

Le diagramme de classe représente la description statique du système en intégrant dans chaque classe la partie dédiée aux données et celle consacrée aux traitements. C'est le diagramme pivot de l'ensemble de la modélisation d'un système.[**14**]

3.2.9.1. Description des classes

Le diagramme ci-dessous représente le diagramme de classe de notre système.

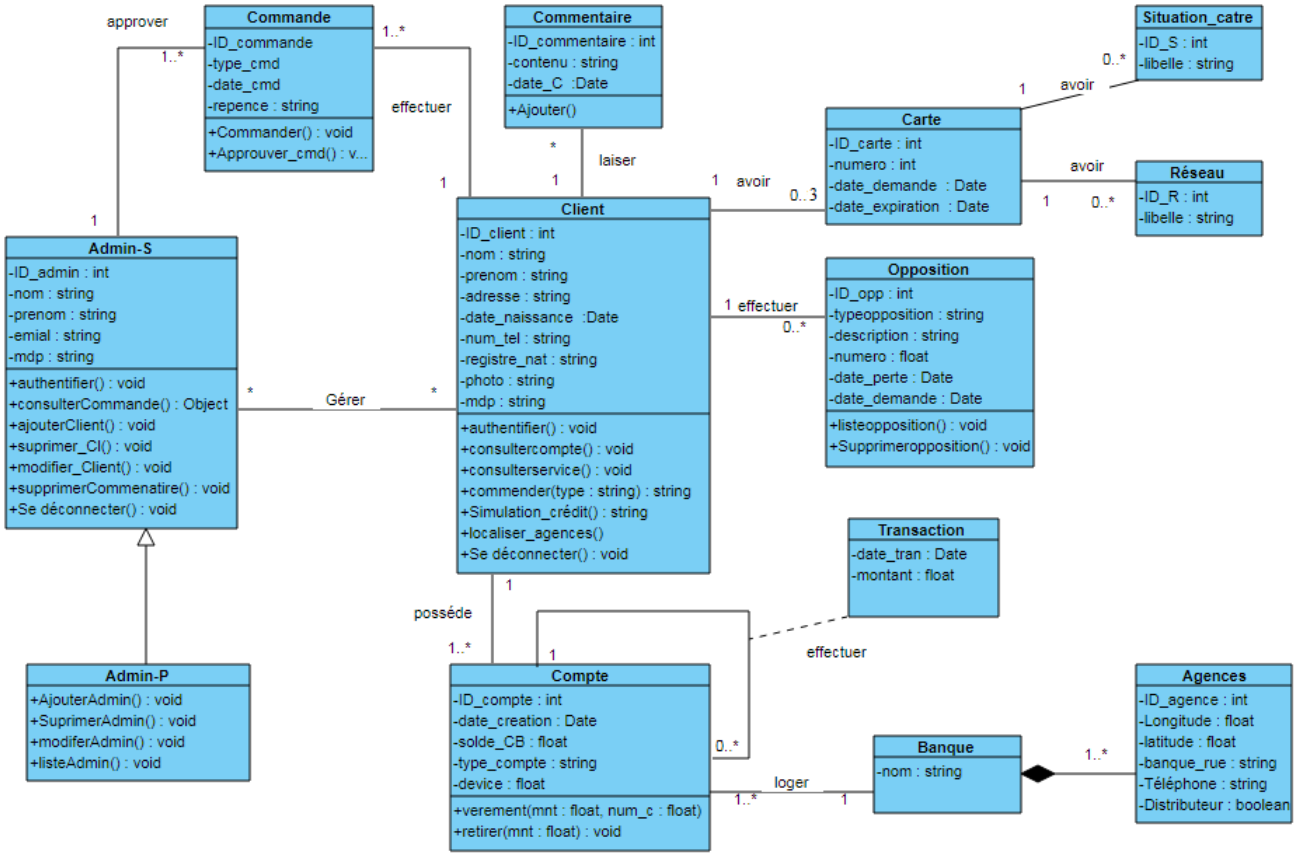


FIGURE 3.17 – Diagramme de classe du système.

Le tableau ci-dessous représente la description des classes de notre système.

Client	Regroupe les informations des clients ayant interagi avec notre application mobile.
Administrateur principal	Représente la classe de l'administrateur qui gère tous les administrateurs de l'application web.
Administrateur simple	Regroupe les administrateurs qui gèrent la plateforme.
Commande	Regroupe les informations des commandes envoyées pour être réalisées.
Transaction	Regroupe les informations sur les transactions entre les clients.
Commentaire	Regroupe les commentaires des clients sur l'application mobile.
Compte	Regroupe les informations sur le compte de chaque client.
Opposition	Regroupe les oppositions émises par les clients.
Agences	Regroupe les informations des agences.

TABLE 3.1 – Description des classes du système.

3.2.10. Dictionnaire de données

Dans le tableau ci-dessous sont décrites et expliquées toutes les données qui sont relatives aux classes de notre système.

Classe	Attribut	Description	type	Taille
Client	ID_Client	Identifiant de client	int	11
	nom	Nom de client	varchar	30
	prenom	Prénom de client	varchar	30
	adresse	Adresse de client	varchar	50
	date_naissance	Date naissance de client	dateTime	/
	num_tel	Numéro téléphone de client	int	11
	registre_nat	Numéro de carte d'identité	int	11
	photo	Photo profil de client	varchar	250
	mdp	Mot de passe de Client	varchar	20
Compte	ID_compte	L'identifiant de compte	int	11
	date_creation	Date de création de compte	dateTime	/
	solde_CB	Solde de compte d'un client	float	11
	type_compte	Type de compte	vatchar	20
	device	Solde device d'un client	float	11
Administrateur	ID_admin	Identifiant de l'administrateur	int	11
	nom	Nom de l'administrateur	varchar	30
	prenom	Prénom de l'administrateur	varchar	30
	email	Adresse email de l'administrateur	varchar	50
	mdp	Mot de passe de l'admin	varchar	20
commentaire	ID_commentaire	Identifiant de commentaire	int	11
	contenu	Contenu de commentaire	text	/
	Note	Note donnée à l'application	int	5
	date_C	Date de commentaire	dateTime	/
Commande	ID_Commande	Identifiant de Commande	int	11
	type_cmd	Type de commande	varchar	30
	date_cmd	Date de commande	varchar	30
	status	Reponse de commande	dateTime	/
Opposition	ID_opp	Identifiant du l'opposition	int	11
	type_oppositttion	Type de l'opposition	varchar	30
	date_perte	Date de perte	dateTime	/
	date_demande	Date d'opposition	dateTime	/
Carte	ID_carte	Identifiant de Carte	int	11
	numéro	Numéro de carte	int	11
	date_demande	Date de demande de la carte	dateTime	/
	date_expiration	Date d'expiration de la carte	dateTime	/
Agence	ID_Agence	Identifiant de l'agence	int	11
	Longitude	Longitude de l'agence	entier	30
	laltitude	Laltitude de l'agence	entier	30
	banque_rue	Rue où se trouve l'agence	varchar	30
	Téléphone	Le numéro de téléphone	varchar	30
	Distributeur	La présence d'un distributeur	boolean	/
Situation_carte	ID_S	Identifiant de situation de carte	int	11
	libelle	Libellé de la carte	dateTime	/
Transaction	date_tran	Date de la transaction	dateTime	/
	montant	Le montant de la transaction	entier	30
Réseau	ID_R	Identifiant du réseau de la carte	int	11
	libelle	Réseau carte	dateTime	/

TABLE 3.2 – Dictionnaire de données

3.2.11. Modèle relationnel

Le modèle relationnel représente la base de données comme un ensemble de tables, sans se préoccuper de la façon dont les informations sont stockées dans la machine. Les relations constituent donc la structure logique du modèle relationnel. Au niveau physique, le système est libre d'utiliser n'importe quelle technique de stockage dès lors qu'il est possible de relier ces structures à des relations au niveau logique. Les relations ne représentent donc qu'une abstraction de l'enregistrement physique des données en mémoire [**15**].

Les règles utilisées pour le passage du modèle du domaine vers le modèle relationnel sont [**14**] :

— **Règle 1(Transformation des classes) :** Chaque classe du diagramme de classe devient une relation, il faut choisir un attribut de la classe pouvant jouer le rôle de clé (le rôle de l'identifiant).

Transformation des associations : Nous distinguons trois familles d'associations :

— **Règle 2 (Association un-à-plusieurs) :** Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.

— **Règle 3 (Association plusieurs-à-plusieurs) :** La classe-association devient une relation. La clé primaire de cette relation est la concaténation des identifiants des classes connectées à l'association, chaque attribut devient clé étrangère si la classe connectée dont il provient devient une relation en vertu de la règle 1. Les attributs de l'association (classe-association) doivent être ajoutés à la nouvelle relation. Ces attributs ne sont ni clé primaire, ni clé étrangère.

— **Règle 4 (Association un-à-un) :** Il faut ajouter un attribut de type clé étrangère dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association. Si les deux multiplicités minimales sont à un, il est préférable de fusionner les deux classes en une seule

— **Règle 5 (Transformation de l'héritage) :** Trois décompositions sont possibles pour traduire une association d'héritage en fonction des contraintes existantes :

- **Décomposition par distinction :** Il faut transformer chaque sous-classe en une relation, la clé primaire de la surclasse, migre dans la relation issue de la sous-classe(s) et devient à la fois clé primaire et clé étrangère.

- **Décomposition descendante :** S'il existe une contrainte de totalité ou de partition sur l'association d'héritage, il est possible de ne pas traduire la relation issue de la surclasse. Il faut alors faire migrer tous ses attributs dans la(les) relation(s) issue(s) de la(des) sous-classe(s).
- **Décomposition ascendante :** Il faut supprimer la relation issue de la sous-classe et faire migrer les attributs dans la relation issue de la surclasse.

Dans mon contexte, j'ai obtenu le schéma relationnel suivant :

— Client(**ID_Client**, nom, prenom, adresse, date_naissance, num_tel, registre_nat, photo,mdp).

— Compte(**ID_compte**, date_creation, slode_CB, type_compte, device, *#ID_Client*).

— Administrateur(**ID_admin**, nom, prenom, email, mdp).

— Commande (**ID_commande**, type_cmd, description, date_cmd, repence, *#ID_Client*, *#ID_admin*).

— Opposition (**ID_opp**, type_opposition, description, numero, date_perte, date_demande, *#ID_Client*).

— Commentaire(**ID_commentaire**, contenu, date_C,*#ID_Client*).

— Transaction(*#ID_Client1 ,#ID_Client2 ,date_tran*, montant).

— Carte(**ID_carte**, nemero, date_demande,date_expiration,*#ID_Client*).

— Situation_carte(**ID_S**, libelle,*#ID_carte*).

— Réseau(**ID_R**, libelle,*#ID_carte*).

— Banque(**nom**).

— Agences(**ID_agence**, longitude, laltitude, banque_rue, Téléphone, Distributeur).

3.2.12. Organigramme de la plateforme

Les figures ci-dessous représente les schémas de navigation sur notre système.

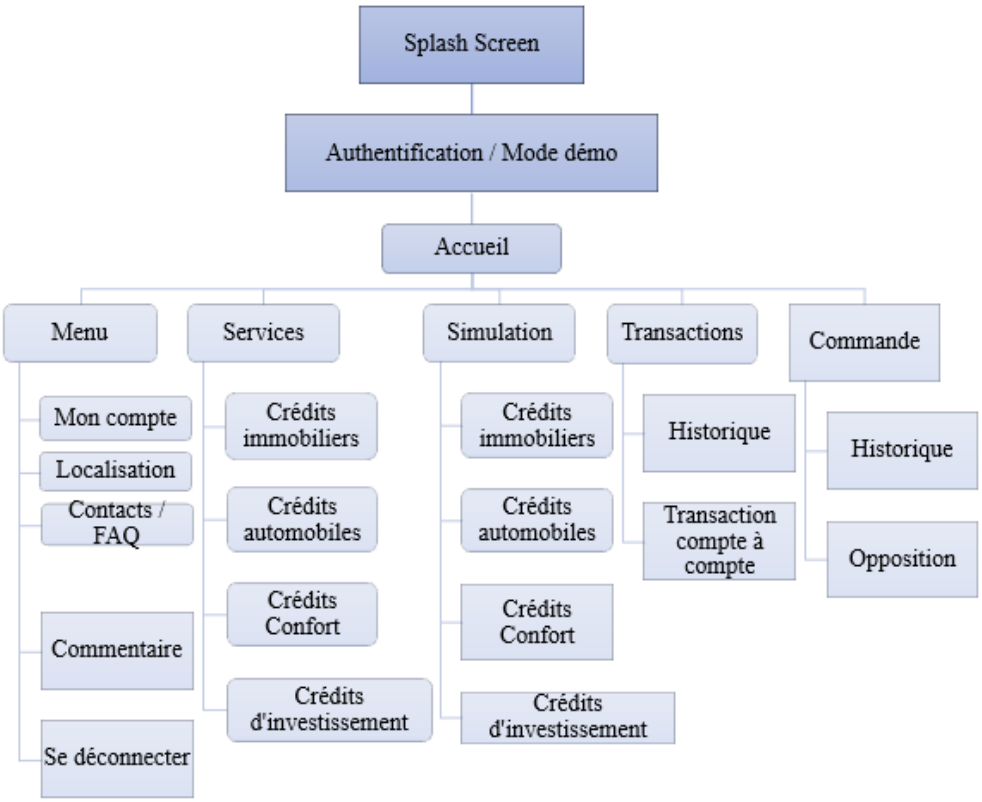


FIGURE 3.18 – Organigramme de l'application mobile

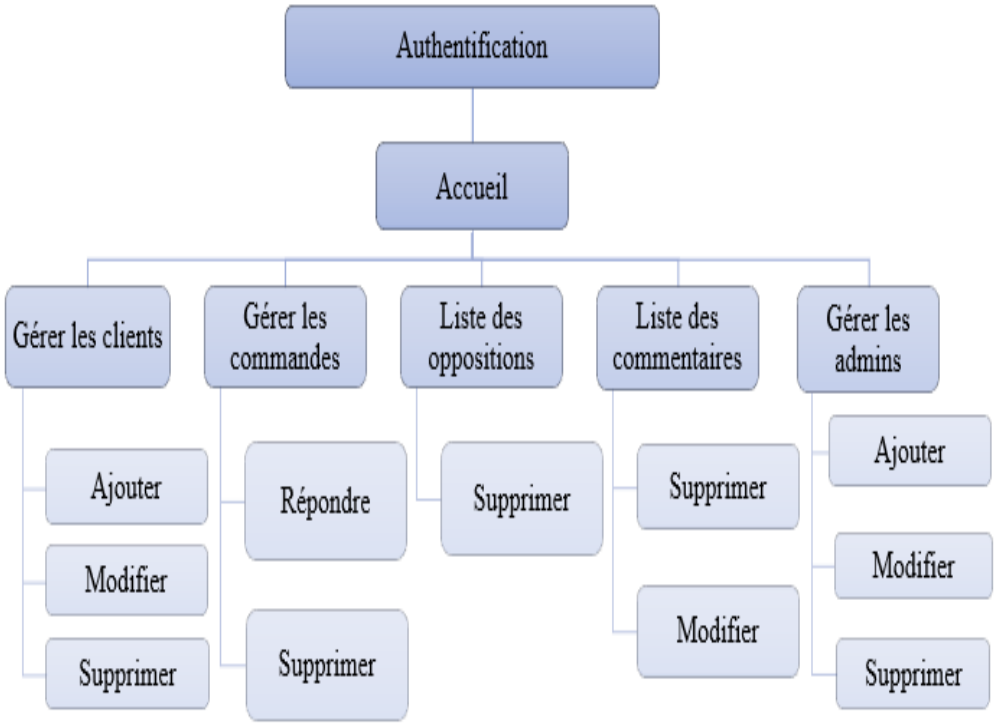


FIGURE 3.19 – Organigramme de l'application web

3.3. Conclusion

Dans ce chapitre nous avons vu les différents diagrammes de la conception commençant par le diagramme de séquence système passant au diagramme plus détaillé qui est le diagramme d'interactions, et en dernier le diagramme de classe, en présentant le passage de notre diagramme de classe en un modèle relationnel qui nous servira à implémenter la base de données et à la fin on a schématisé nos plateformes par deux organigrammes.

Le prochain chapitre sera consacré à la réalisation de notre application en implémentant les solutions proposées.

Chapitre 4

Implémentation

Introduction

Ce dernier chapitre est consacré à la partie pratique de notre projet. Dans un premier temps, nous énumérons les différents outils de développement qui nous ont permis de mener à bien notre application mobile et web. Ensuite, nous présenterons les différents langages de programmation utilisés, les librairies, et enfin les différentes interfaces de note application.

4.1. Environnement et outils de développement

4.1.1. Android Studio

Android Studio est un environnement de développement intégré (EDI) permettant de développer des applications Android. Développé par Google, il se base sur l'EDI IntelliJ de JetBrains. Il offre les outils nécessaires pour développer des applications mobiles natives destinées à Android. Il permet d'éditer des fichiers Java/Kotlin pour la partie programmation et des fichiers XML pour la partie graphique[**16**].

4.1.2. Visual Studio Code

Visual Studio Code est un éditeur de code source léger mais puissant qui s'exécute sur votre bureau et est disponible pour Windows, macOS et Linux. Il est livré avec un

support intégré pour JavaScript, TypeScript et Node.js et possède un riche écosystème d'extensions pour d'autres langages (tels que C,C++, C, Java, Python, PHP, Go) et des runtimes (tels que .NET et Unity)[**17**].

4.1.3. Git et GitHub

Git est un logiciel libre de gestion de versions, sous licence GPL2. GitHub est un service web de gestion et d'hébergement de projet de développement logiciel utilisant le logiciel Git (qui, pour l'anecdote, a été racheté le 4 de ce mois par Microsoft).[**21**]

Cet outil a été un véritable atout pour mon projet. Il m'a permis de travailler de manière fastidieux et méthodique avec la gestion des versions, sur un code source unique, et de suivre l'évolution du travail en temps réel. Le lecteur peut accéder et suivre l'évolution de mon dépôt en visitant le lien suivant : `https://github.com/mindjou06/prototypapp`

4.1.4. WampServer

WampServer est une plate-forme de développement Web sous Windows pour des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PhpMyAdmin pour gérer plus facilement vos bases de données[**18**].

4.1.5. PhpMyAdmin

PhpMyAdmin est un outil logiciel gratuit écrit en PHP, destiné à gérer l'administration de MySQL sur le Web. Prend en charge une large gamme d'opérations sur MySQL tel que la gestion des bases de données, des tableaux, des colonnes, des relations, des index, des utilisateurs, des autorisations, etc. ses opérations peuvent être effectuées via l'interface utilisateur, alors il offre aussi la possibilité d'exécuter directement une instruction SQL[**25**].

4.1.6. Apache

Le logiciel libre Apache HTTP est un serveur HTTP créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du World Wide Web. Il

est distribué selon les termes de la licence Apache[**19**].

4.1.7. MySQL

Le terme MySQL, pour My Structured Query Language, désigne un serveur de base de données distribué sous licence libre GNU (General Public License). Il peut être utilisé sur de nombreux systèmes d'exploitation (Windows,MacOS,etc.). Il supporte les langages informatiques SQL et SQL/PSM. Dans la pratique, le serveur MySQL peut se résumer à un lieu de stockage et d'enregistrement des données[**20**].

4.1.8. SDK de Android

Le SDK (Software Developpement Kit) d'Android est un ensemble d'outils de développement essentiel au développement d'applications mobiles Android. Il inclut différents outils tel qu'un débogueur, un émulateur basé sur QEMU, et un ensemble de bibliothéques logicielles auxquels vient s'ajouter une documentation des plus riches[**28**].

4.2. Langage de programmation

Java est un puissant langage de programmation orienté objet. Il a la particularité d'être portable, c'est-à-dire qu'il est possible d'exécuter les programmes écrits en Java sous n'importe quel système d'exploitation grâce à la JVM incluse dans le JDK[**21**].

XML eXtensible Markup Language (Langage de balisage extensible) est un métalangage informatique de balisage générique. Il permet de structurer des données grâce à des balises et est utilisé essentiellement pour décrire les interfaces graphiques Android et autres fichiers de données globales[**8**].

HTML5 C'est la dernière revision majeure du HTML, il désigne un langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web[**20**].

CSS3 CSS est un langage de mise en forme d'un document HTML. Il définit les règles de style et de disposition appliquées aux éléments d'un document html. On utilise le CSS pour modifier le style de n'importe quel élément html pour corriger ses dimensions, couleurs, bordures, etc[**20**].

JavaScript JavaScript, a été créé pour permettre un accès par script à tous les éléments d'un document HTML. En d'autres termes, il offre une possibilité d'interaction dynamique de l'utilisateur, comme la vérification de validité d'une adresse courriel dans des formulaires d'entrée d'informations, et afficher des invites c'est-à-dire des messages de demande de confirmation[**20**].

JSON JSON (JavaScript Object Notation - Notation Objet issue de JavaScript) est un format léger d'échange de données. Il est facile à lire ou à écrire pour des humains et aisément analysable ou générable par des machines. Il est basé sur un sous-ensemble du langage de programmation JavaScript. JSON est un format texte complètement indépendant de tout langage[**26**].

PHP PHP (HyperText Préprocesseur) est un langage de scripts open source, conçu spécialement pour le développement d'applications web. Il peut être intégré facilement au HTML. Il s'exécute de côtés serveur qui permet la génération des pages HTML dynamiquement et permet aussi la connectivité étendue aux bases de données[**20**].

SQL Le langage SQL (Structured Query Language) est un langage informatique utilisé pour exploiter des bases de données. Il permet de façon générale la définition, la manipulation et le contrôle de sécurité de données. Dans la pratique, le langage SQL est utilisé pour créer des tables, ajouter des enregistrements sous forme de lignes, interroger une base de données, la mettre à jour,ou encore gérer les droits d'utilisateurs de cette base de donnée[**20**].

4.3. Frameworks utilisés

Symfony 5 C'est un ensemble de composants PHP, ainsi qu'un Framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web. Il fournit une méthodologie (conventions d'écriture et d'organisation, discipline du code produit, utilisation du modèle MVC)[**22**].

Doctrine 2 C'est un ORM (Object-Relational Mapping) pour PHP, intégré par défaut dans Symfony, il permet de donner l'illusion de travailler avec une base de données objet alors que l'on est sur une base de données relationnelle. Grâce à cela le développeur ne fait plus de requête SQL (sauf cas spécifiques rares) et travailles

avec ses objets[**23**].

Bootstrap Bootstrap est une infrastructure de développement frontale, gratuite et open source pour la création de sites et d'applications Web. L'infrastructure Bootstrap repose sur HTML, CSS et JavaScript(JS) pour faciliter le développement de sites et d'applications réactives et tout-mobile. La conception réactive permet à une page ou une application Web de détecter la taille et l'orientation de l'écran du visiteur pour adapter automatiquement l'affichage[**24**].

JQuery jQuery, est une bibliothèque JavaScript gratuite, libre et multiplateforme. Compatible avec l'ensemble des navigateurs Web, elle a été conçue et développée pour faciliter l'écriture de scripts. Il s'agit du framework JavaScript le plus connu et le plus utilisé. Il permet d'agir sur les codes HTML, CSS, JavaScript et AJAX et s'exécute essentiellement côté client[**27**].

jQuery est une bibliothèque JavaScript qui permet aux développeurs Web d'ajouter des fonctionnalités supplémentaires à leurs sites Web. Il est open source. Au cours des dernières années, jQuery est devenue la bibliothèque JavaScript la plus populaire utilisée dans le développement Web .

Sécurité La sécurité dans Symfony est robuste et très poussée, vous pouvez la contrôler facilement. Filtrage des inputs, authentification, gestion des sessions, protection contre les injections et les failles XSS, CSRF, Cette série d'outils aborde les différentes mécaniques que tout développeur Symfony peut utiliser, afin d'assurer la sécurité et la fiabilité de l'application à développer.

4.4. Quelques interfaces de notre système

Dans ce qui suit, nous présentons quelques interfaces de l'application mobile et web de notre système.

4.4.1. Interfaces de l'application Android

Nous présentons les interfaces de notre application mobile pour la partie client.

4.4.1.1. Interface d'authentification

La figure 4.1 présente l'interface d'authentification, c'est la deuxieme interface affichée à l'utilisateur au lancement de l'application après le splash screen. Son principal

composant est une vue qui offre aux utilisateurs un service pour passer en mode visiteur 'demo' et d'authentification pour les clients.

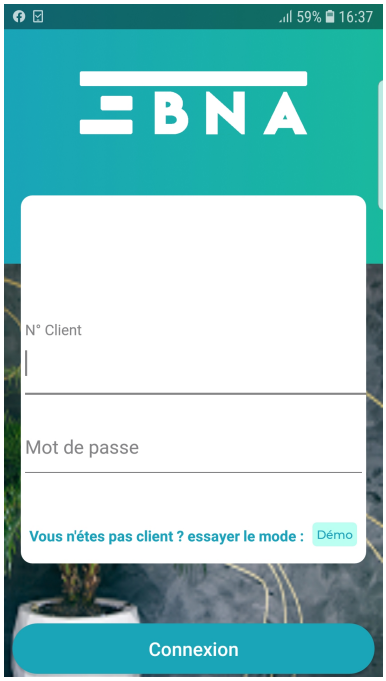


FIGURE 4.1 – Interface connexion.

4.4.1.2. Interfaces d'accueil

La figure 4.2 présente l'interface d'accueil, son principal composant est une vue qui regroupe les différentes fonctionnalités préalablement conçu pour les clients. Chaque fonctionalité est affichée avec une icone auquel elle lui correspond. Aussi, un menu latéral pour naviguer entre les différents fragements et activités de l'application, et enfin un menu qui ouvre une liste déroulante pour voir d'autres fonctionnalités tel que (localisation des agences BNA, voir le compte du client , déconnexion, etc).

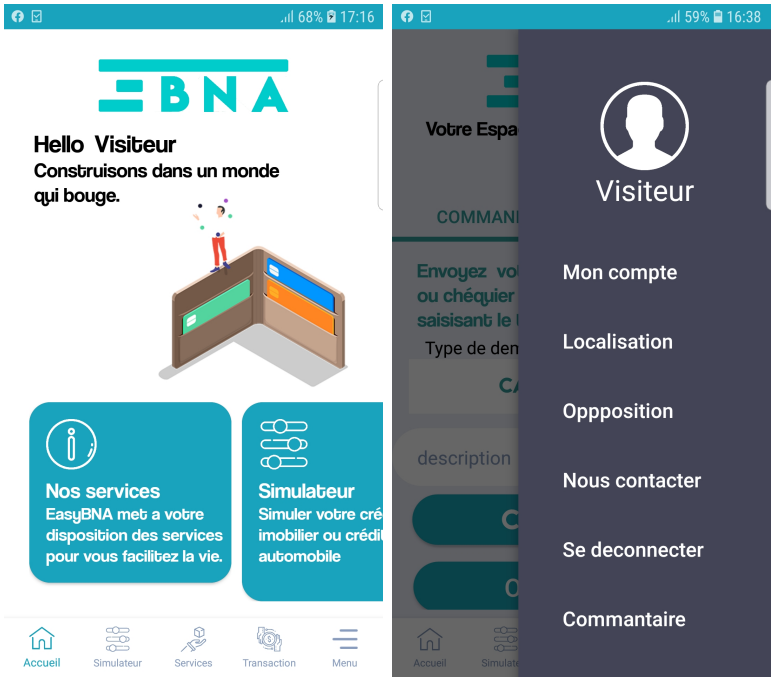


FIGURE 4.2 – Interfaces d'accueil

4.4.1.3. Interfaces Service

La figure 4.3 représente les crédits offerts par la BNA, les clients ont la possibilités de faire une simulation virtuele de crédit choisi.

(a) Crédit Immobilier

(b) Crédit Confort

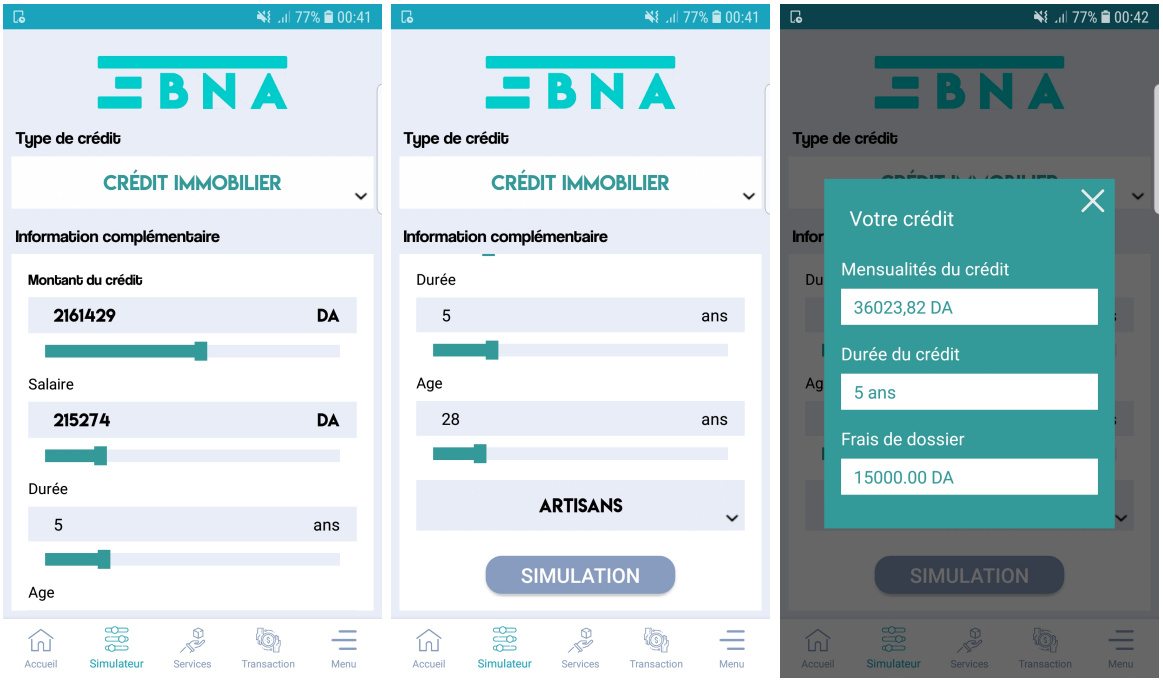
(c) Crédit d'Investissement

(d) Crédit Auto

FIGURE 4.3 – Interfaces des services

4.4.1.4. Interfaces Simulation

La figure 4.4 représente l'interface de simulation d'un crédit bancaire. Le client est invité à remplir les champs, à choisir le type de crédit et sa profession.



(a) Simulation crédit (b) Simulation crédit suite (c) Resultat simulation

FIGURE 4.4 – Interfaces simulation.

4.4.1.5. Interface localisation des agences

La figure 4.5 représente l'interface de localisation avec la Google Map qui contiendra des markers qui représentent les défferentes agences de BNA au niveau de la wilaya de béjaia, en zoomant, et en cliquant sur le marker le nom de l'agence, sa spécialité ainsi que son numéro de téléphone s'afficheront sur une petite fenêtre.

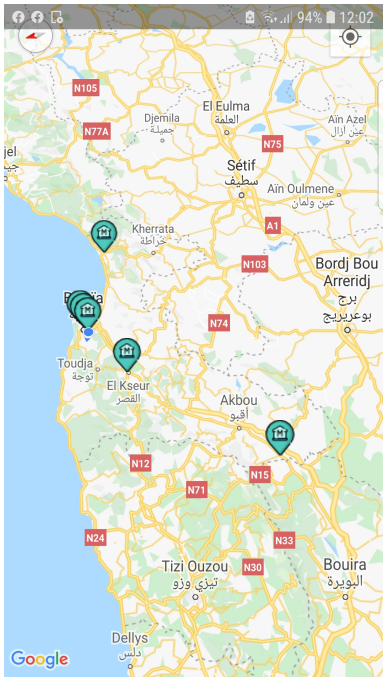


FIGURE 4.5 – Interface localisation

4.4.1.6. Interface transaction bancaire

La figure 4.6 représente l'interface de la transation bancaire. Le client pourra voir son solde en compte en dinar et il est invité à remplir le montant de la transaction et choisir à quel destinataire sera verser le montant et enfin il paura visualiser l'historique des transactions effectuées.

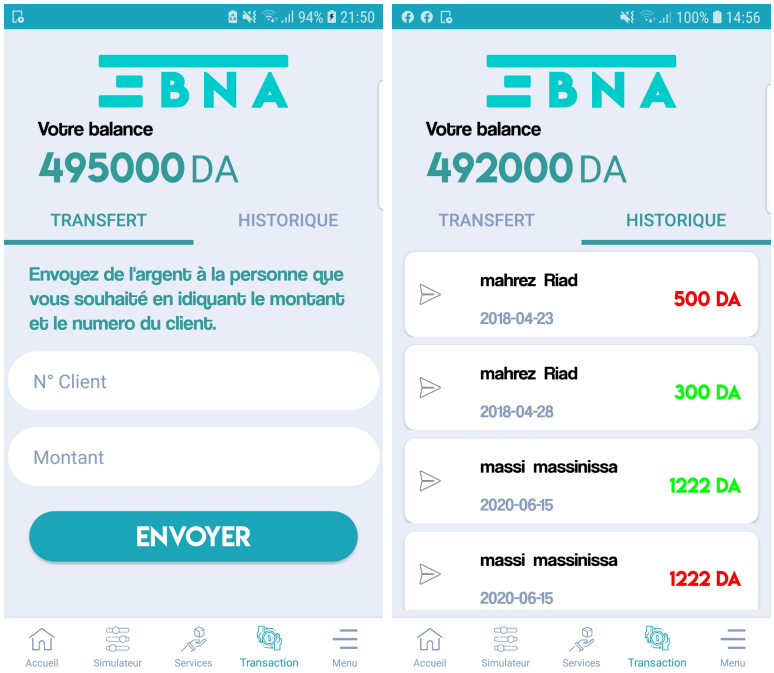


FIGURE 4.6 – Interfaces transaction bancaire.

4.4.1.7. Interface commande

La figure 4.7 représente l'interface de commande de carte ou de chéquier. Le client pourra suivre sa demmande et il est invité à remplir une description , à choisir quel type de commande et il paura visualiser l'historique de ses commandes avec leur réponse et enfin il paura effectuer une oppostion en cas de vol ou de perte de ces cartes ou son chéquier.

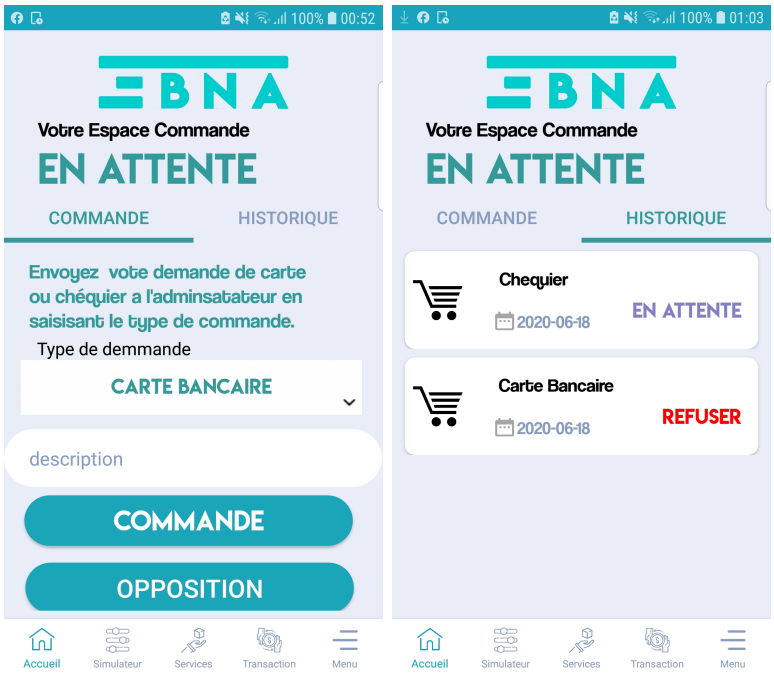


FIGURE 4.7 – Interface commande.

4.4.2. Interfaces de l'application d'administrateur

Nous présentons quelques interfaces de l'application web d'administration.

4.4.2.1. Page d'authentification

La figure 4.8 représente l'interface qui permet aux administrateurs de s'authentifier au sein de notre application web.

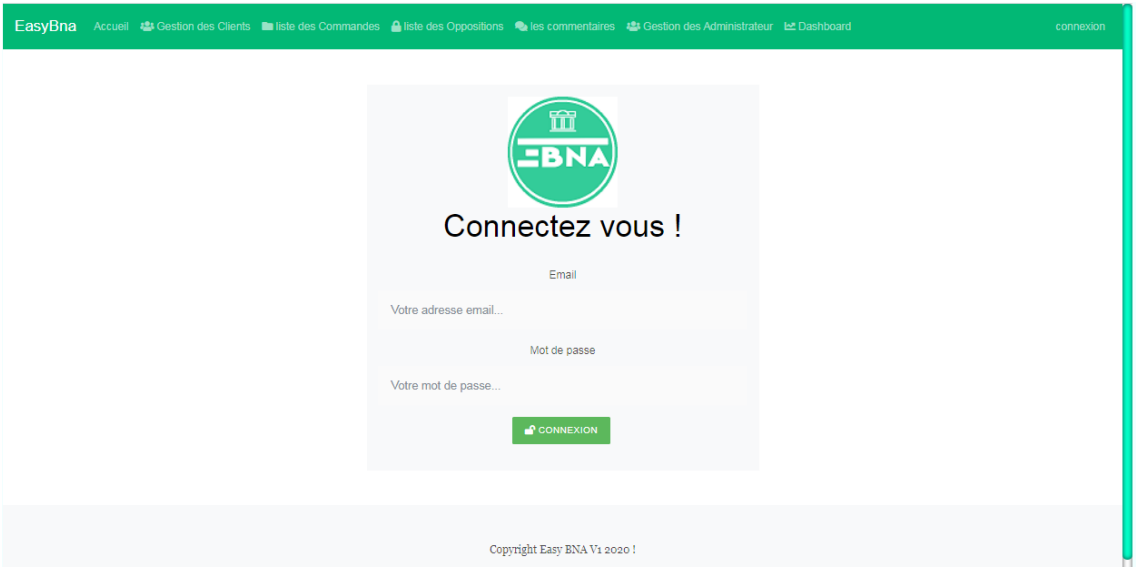


FIGURE 4.8 – Page d'authentification.

4.4.2.2. Page Gestion des Clients

La figure 4.9 représente l'interface qui permet à l'administrateur de gérer les clients.

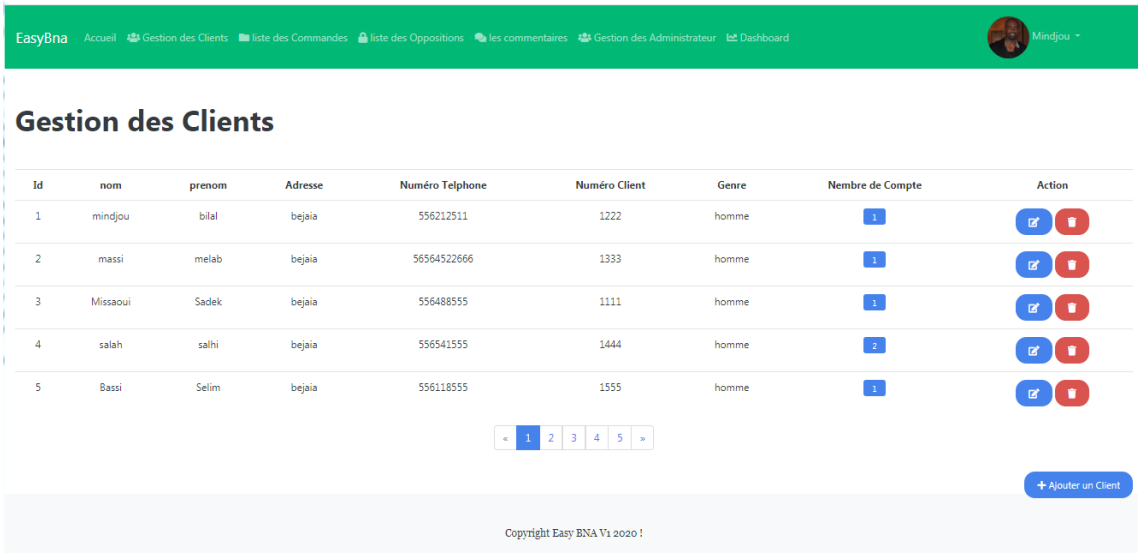


FIGURE 4.9 – Page gestion des clients.

4.4.2.3. Page Gestion des Commandes

la figure 4.10 représente l'interface qui donne la possibilité à l'administrateur de consulter les commandes des clients et pouvoir leur répondre.

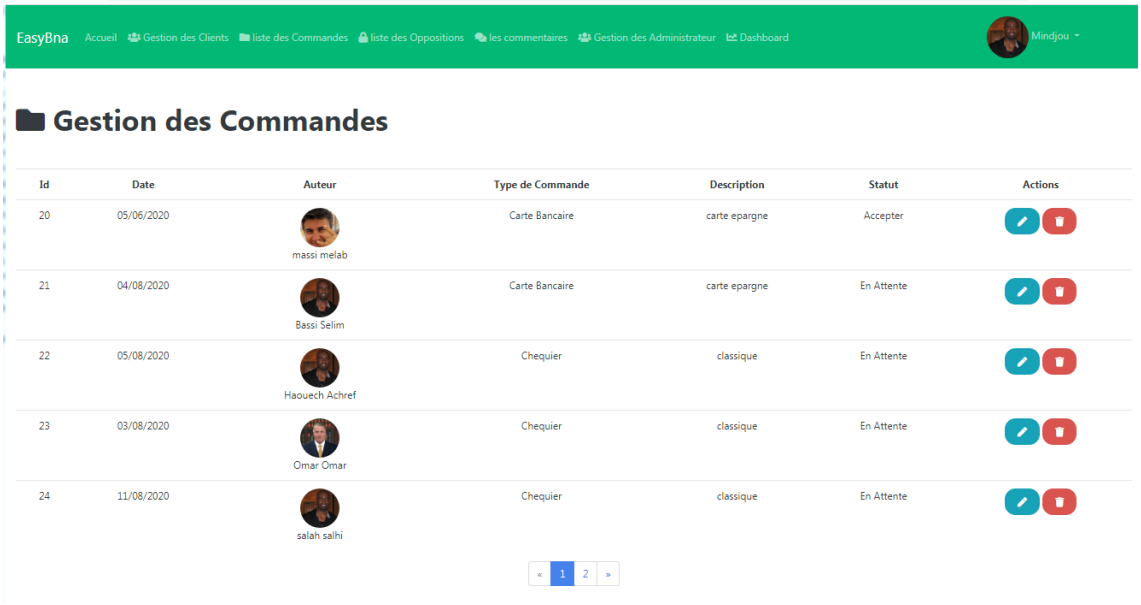


FIGURE 4.10 – Page consulter les commandes.

4.5. Conclusion

Dans ce dernier chapitre, nous avons mis en avant tout ce qui est langages, frameworks, outils et logiciels qui ont contribué à l'élaboration de notre application (web et mobile) ainsi que les principales fonctionnalités en exposant quelques interfaces de l'application accompagnées de leurs descriptions afin d'avoir une vue générale de notre application.

Conclusion générale et perspectives

Ce travail a été réalisé dans le cadre de notre projet de fin de cycle master en Génie logiciel. Il a consisté en l'élaboration d'un système bancaire constitué d'une application Android et d'une application web permettant une gestion optimale du temps du client, et ce grâce à une étude du contexte et de la problématique. Nous avons commencé par dégager un cahier de charges, analysé les différentes fonctionnalités attendues dans la future application. L'application à été modélisée avec UML en suivant le processus de développement UP. nous avons, finalement, implémenté ces dernières en ayant recours à l'environnent de développement Android Studio, au langage Java, PHP et XML, framework symfony, GitHub ainsi que Git comme outils de travail de versioning, et en exploitant des librairies Open Source comme «ColorPicker» et «WeekView».

Nous avons réussi à concevoir une application mobile *'EasyBNA'* qui donne la possibilité au client d'être plus proche de sa banque et de recevoir des notifications, d'avoir une vue détaillée sur les différents services offerts par la banque et pleines d'autres fonctionnalités utiles et aussi une application web pour l'administration. Les applications réalisées peuvent être facilement utilisée par les clients et les employés de la banque et ne demande pas un grand temps d'adaptation.

Malgré tout le travail fourni, nous sommes conscients que plusieurs aspects de notre système peuvent et doivent être améliorés. On peut citer à titre d'exemple l'optimisation du code pour permettre une meilleure exploitation des ressources. En guise de perspectives, nous aspirons à enrichir l'application (mobile et web) avec d'autres fonctionnalités telles que les transactions avec des clients d'autres banques, et le suivi et l'évaluation des crédits.

Bibliographie

[1] BNA, *presentation-de-la-bna*, Disponible sur <https ://bna.dz/fr/a-propos-d e-la-BNA/presentation-de-la-bna.html> (consulté le 2 juin 2020).

[2] Finance, *Les risques et les garanties bancaires*, Disponible sur <https ://memoireonline.com> (consulté le 16/05/2020).

[3] *Application mobile* , Disponible sur <https ://taktilcommunication.com/blog/> (consulté le 27/04/2020).

[4] Jean-François LALANDE, *Développement Android*, INSA Centre Val de Loire, 2016.

[5] *Openclassrooms*, Disponible sur <https ://openclassrooms.com> (consulté le 16/03/2020).

[6] w3, *web services*, Disponible sur <https ://www.w3.org> (consulté le 16/02/2020).

[7] supinfo, *Protocoles de communication avec les services web*, Disponible sur < https ://www.supinfo.com > (consulté le 10/03/2020).

[8] *Oreilly* , Disponible sur <https ://www.oreilly.com/library/view/programming-web-services > (consulté le 10/03/2020).

[9] R. P. Franck Vallée, *UML 2 en action, de l'analyse des besoins à la conception*, $4^{ème}$ édition, 2007.

[10] A. V. André Pascal, *Développement de logiciels avec UML 2 et OCL*, ellipses, 2013.

[11] JOSEF GABAY, DAVID GABAY, *UML2 Analyse et Conception*, Université de Québec, 1^{re} édition, 2009.

[12] Laurent Audibert, *UML 2 de l'apprentissage a la pratique* , Disponible sur <https ://laurent-audibert.developpez.com/Cours-UML/> (consulté le 10/03/2020).

[13] NIEDERCORN, *UML : Cas d'utilisation*, Lycée technique «La Briquerie» 57100 Thionville, Disponible sur <http ://niedercorn.free.fr/iris/iris1/uml/uml10.pdf> (consulté le 05/05/2020).

[14] CHRISTIAN SOUTOU, *UML 2 pour les bases de données*, Eyrolles, Paris, 2007.

[15] RIM CHAABANE, *Le modèle de données relationnel*, Disponible sur <http ://www.ai.univ-paris8.fr> (consulté le 01/05/2020).

[16] *Android*, Disponible sur <https ://android-studio.fr> (consulté le 01/05/2020).

[17] *Code*, Disponible sur <https ://code.visualstudio.com> (consulté le 12/04/2020).

[18] *Wamp*, Disponible sur <https ://wampserver.com> (consulté le 12/04/2020).

[19] *Apache*, Disponible sur <https ://www.apache.orgr> (consulté le 12/04/2020).

[20] ROBIN NIXON, *Développer un site web en php, mysql, javascript jquery, css3 et html5*, $4^{\text{ème}}$ édition, traduction de William Piette, Edition O'Reilly, 2016.

[21] *Github*, Disponible sur <https ://github.com> (consulté le 12/04/2020).

[22] *Symfony*, Disponible sur <https ://www.symfony.com> (consulté le 12/04/2020).

[23] *Doctrine*, Disponible sur <https ://doctrine.org> (consulté le 12/04/2020).

[24] *Bootstrap*, Disponible sur <https ://W3Schools.com> (consulté le 12/04/2020).

[25] *Project*, Disponible sur <https ://projet-plume.org/fiche/phpmyadmin> (consulté le 12/04/2020).

[26] *JSON*, Disponible sur <https ://json.org> (consulté le 12/04/2020).

[27] *JQuery*, Disponible sur <https ://jquerymobile.com> (consulté le 12/04/2020).

[28] FREDERIC ESPIAU, «Créez des applications pour Android», 2012.

[29] Disponible sur <https ://book.cakephp.org> (consulté le 12/04/2020).

Annexe A

Annexe

A.1. Interfaces opposition

La figure A.1(a) représente les informations en cas de perte ou vol de la carte bancaire ou du chéquier .

La figure A.1(b) représente l'inteface pour faire une demande d'opposition du client.

(a) information sur l'opposition (b) faire une opposition

FIGURE A.1 – Interfaces d'Opposition

A.2. Interface Contact

La figure A.2 représente les informations pour contacter le service administratif de la BNA .

FIGURE A.2 – Interface Contact

A.3. Interface commentaire

La figure A.3 représente l'espace pour faire un commentaire sur l'application mobile.

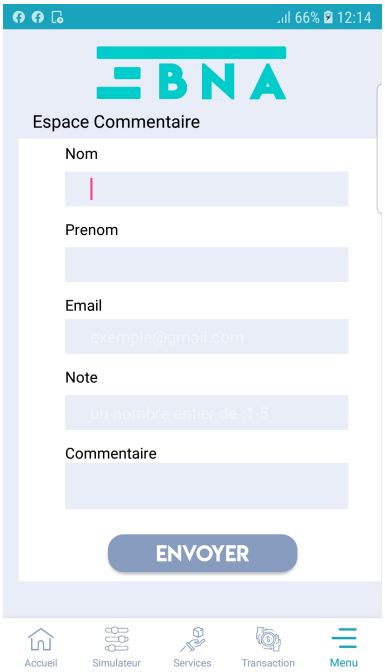


FIGURE A.3 – Interface Commantaire

A.4. Interface mon compte

La figure A.4 représente les informations personnelles du client et lui permet aussi de pouvoir se déconnecter .

FIGURE A.4 – Interface mon compte

A.4.0.1. Page d'accueil

Cette figure A.5 est la page d'accueil après avoir fait l'authentification.

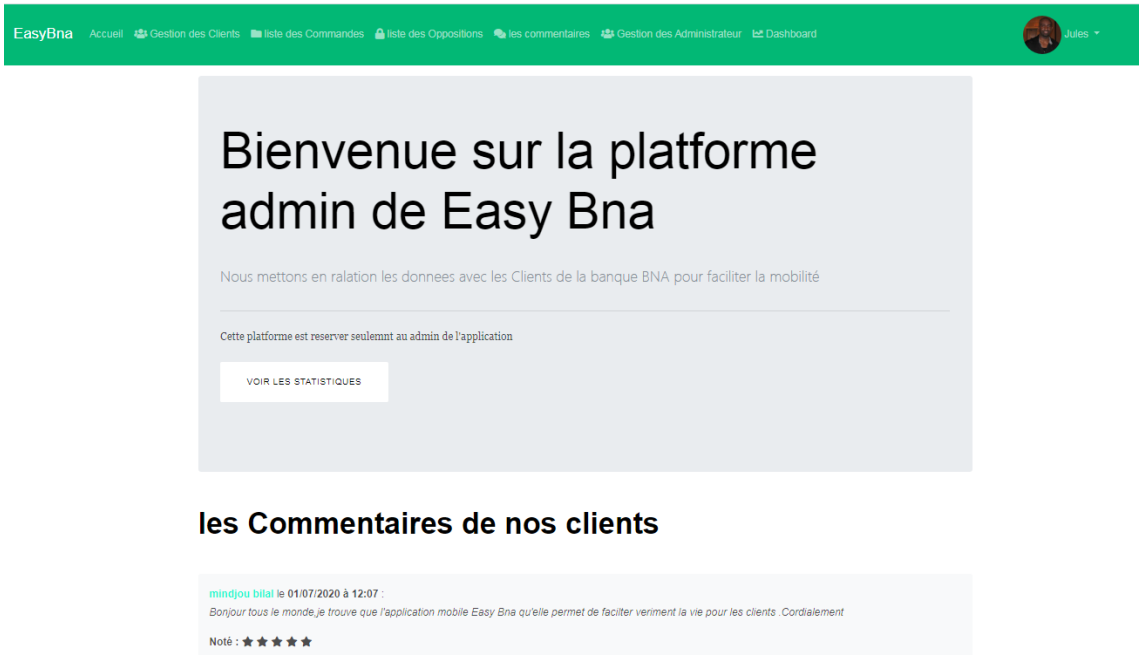


FIGURE A.5 – Page d'accueil.

A.4.0.2. Page Gestion des commentaires

Sur cette figure A.6 l'administrateur peut lire, modifier ou supprimer les commentaires.

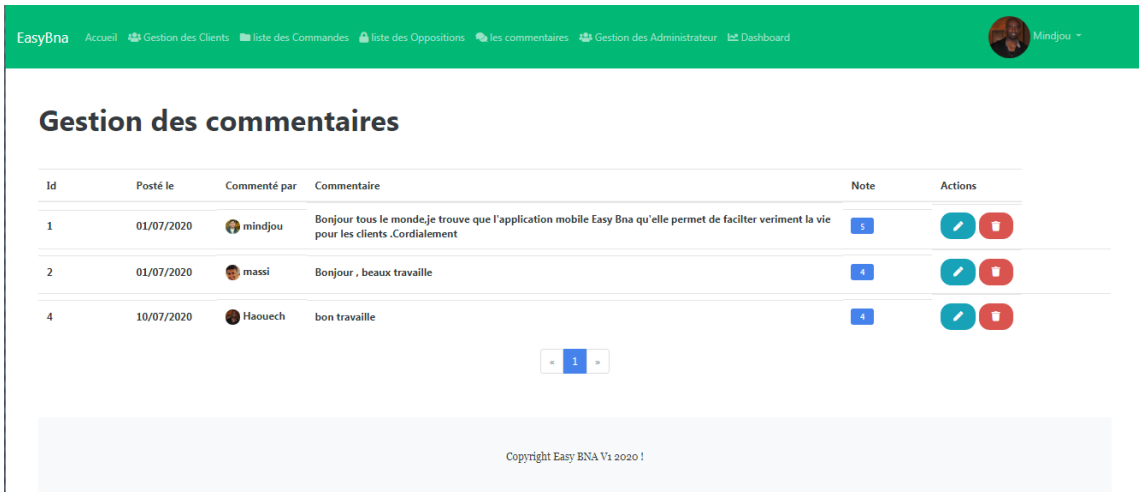


FIGURE A.6 – Page gestion des commentaires.

A.4.0.3. Page liste des oppositions

Sur cette figure A.8 l'administrateur peut lire, supprimer les oppositions faite par les clients.

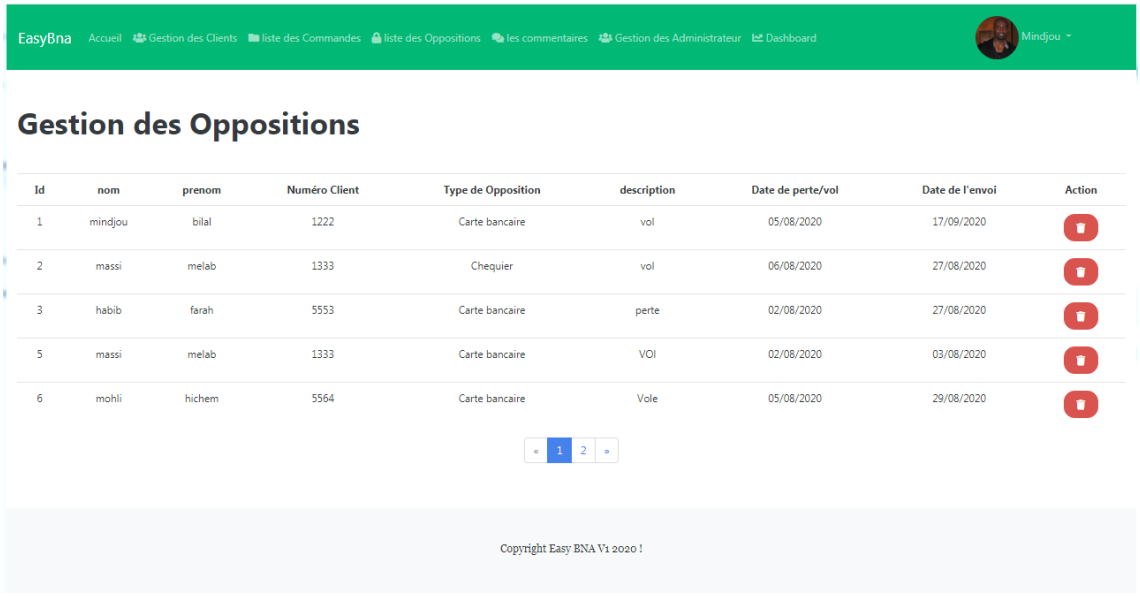


FIGURE A.7 – Page liste des oppositions.

A.4.0.4. Page Statistique

Sur cette figure A.8 l'administrateur principal peut consulter des informations sur la base de données de notre application.

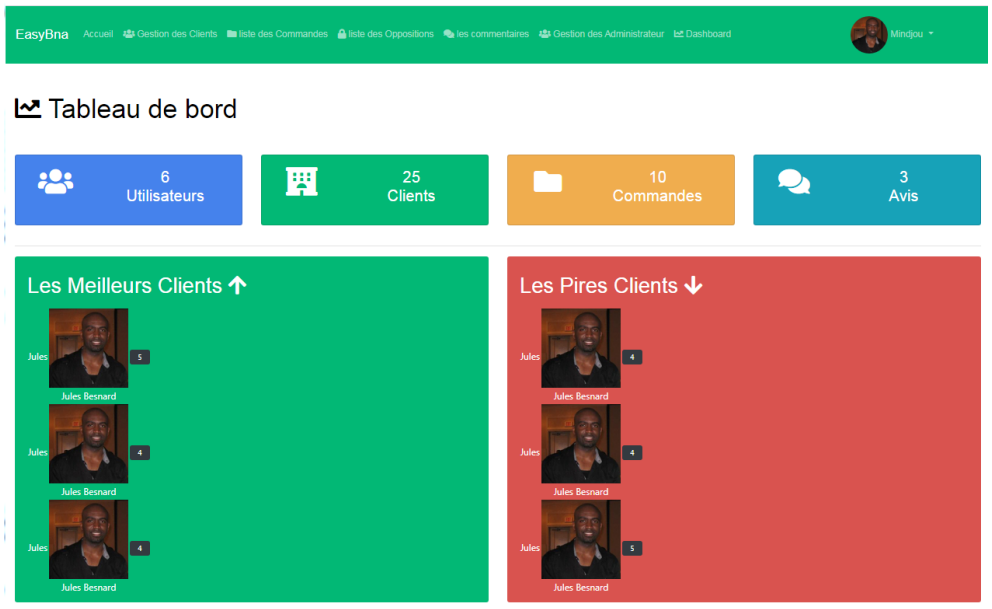


FIGURE A.8 – Page Statistique.

Résumé

Ce document a été rédigé en vue de l'obtention du diplôme de master en génie logiciel. Au cours de cette décennie, l'industrie de la Technologies de l'information et de la communication (TIC) a connu une progression fulgurante suite à l'apparition des smartphones. Voulant exploiter ces technologies pour aider les clients de la BNA à mieux gérer leur temps et à être plus libre, nous avons réalisé une application android qui permet aux clients de la BNA de gérer plusieurs services bancaire, faire des transactions et des commandes, des oppositions de carte/chéquier, etc ainsi que la possibilité de recevoir des alertes. D'autre part, une application web où l'administrateur utilise cette solution pour ajouter et modifier les clients, répondre aux commandes, et consulter les opposition. Pour mener à terme ce projet, nous avons d'abord analysé les besoins, proposé une solution que nous avons modélisé grâce à UML, et implémenté en utilisant Android Studio, Symfony, Java, PHP, XML, SQL, Git, GitHub, et d'autres librairies open source.

Mots clés : *Application web, Application mobile, Android, BNA, Services bancaire.*

Abstract

This document was written for a Master's degree in software engineering .During this decade, the information and communication technology (ICT) industry has experienced tremendous growth following the emergence of smartphones. Wanting to use these technologies to help BNA clients manage their time better and be more free, we have create an android application that allows BNA clients to manage several banking services, allows transactions and orders, card / checkbook oppositions, etc. and the possibility of receiving alerts. On the other hand, a web application where the administrator uses this solution to add and modify customers, distribute orders, and consult oppositions. To complete this project, we first analyzed the needs, proposed a solution that we modeled using UML, and implemented using Android Studio, Symfony, Java, PHP, XML, SQL, Git, GitHub, and other open source libraries.

Keywords : *Web Application, Mobile application, Android, BNA, banking services.*