# DEDICACE

*A ma famille*,

# REMERCIEMENTS

Les travaux réalisés lors de notre stage de fin d’études ont été rendus possible grâce aux efforts conjugués de certaines personnes qui ont œuvré pour le bon déroulement du stage et à l’élaboration de ce présent rapport. Nous voudrions les remercier et leur témoigner notre profonde gratitude et notre infinie reconnaissance.

De ce fait nous commençons par remercier DIEU, pour la force, la santé et le courage qu’il nous a donnés pendant notre stage.

Nous adressons de façon particulière nos remerciements à :

* M. Jean-André AHIPEAUD, Président Directeur Général de J3A PARTNERS AM pour nous avoir accueillis au sein de son entreprise et pour le temps qu’il a consacré à nous apporter les outils méthodologiques indispensables à la conduite de notre projet ;
* M. Guessam Cyrille KOUADIO, Directeur Technique de J3A Finance, notre maître de stage, pour la formation technique reçue, son apport et son assistance dans notre intégration ;
* M. Kouassi Kan Juste Elie, ingénieur chez J3A Finance, pour l’accueil, la sympathie, et l’appui qu’il nous a témoigné durant notre stage ;

Nous exprimons également notre gratitude envers Dr. ASSIELOU Kouamé Abel, notre encadreur, pour son soutien inestimable, sa précieuse aide, et sa disponibilité tout au long de notre projet.

Nous souhaitons également remercier le corps professoral et administratif de l’Institut National Polytechnique Félix Houphouët-Boigny (INPHB) en général et celui de l’Ecole Supérieure des Industries en particulier qui a déployé de grands efforts pour nous assurer une formation de qualité

# SOMMAIRE

[DEDICACE I](#_Toc180103472)

[REMERCIEMENTS II](#_Toc180103473)

[SOMMAIRE III](#_Toc180103474)

[LISTE DES SIGLES ET ABREVIATIONS V](#_Toc180103475)

[LISTE DES FIGURES VIII](#_Toc180103476)

[LISTE DES TABLEAUX X](#_Toc180103477)

[AVANT PROPOS XI](#_Toc180103478)

[RESUME XII](#_Toc180103479)

[INTRODUCTION GENERALE 1](#_Toc180103480)

[PARTIE 1 : CADRE ET CONTEXTE DU PROJET 2](#_Toc180103481)

[CHAPITRE I : PRESENTATION DE L’ORGANISME D’ACCUEIL 3](#_Toc180103482)

[I. PRESENTATION DE J3A FINANCE 3](#_Toc180103483)

[II. ORGANISATION STRUCTURELLE DE J3A FINANCE 4](#_Toc180103484)

[CHAPITRE II : PRESENTATION DU PROJET 6](#_Toc180103485)

[I. ETUDE DE L’EXISTANT 6](#_Toc180103486)

[II. PRESENTATION GENERALE DU PROJET 9](#_Toc180103487)

[PARTIE 2 : ETUDE CONCEPTUELLE DU PROJET 12](#_Toc180103488)

[CHAPITRE III : METHODOLOGIE DU PROJET 13](#_Toc180103489)

[I. GENERALITES SUR LES METHODES D’ANALYSES 13](#_Toc180103490)

[II. PRESENTATION DE QUELQUES METHODES D’ANALYSE 14](#_Toc180103491)

[CHAPITRE IV : INITIALISATION 19](#_Toc180103492)

[I. SPECIFICATIONS DES BESOINS 19](#_Toc180103493)

[II. DELIMITATION DU SYSTEME 20](#_Toc180103494)

[CHAPITRE V : ELABORATION 26](#_Toc180103495)

[II. ANALYSE : DESCRIPTION TEXTUELLE DES CAS D’UTILISATIONS 27](#_Toc180103496)

[III. ELABORATION DES DIAGRAMMES DE SEQUENCES ET D’ACTIVITES 36](#_Toc180103497)

[IV. ELABORATION DU DIAGRAMME DE CLASSE 61](#_Toc180103498)

[PARTIE 3 : MISE EN ŒUVRE DU SYSTEME 63](#_Toc180103499)

[CHAPITRE VI : CONSTRUCTION 64](#_Toc180103500)

[I. ARCHITECTURE LOGICIELLE 64](#_Toc180103501)

[II. SYSTEME DE GESTION DE BASE DE DONNEES 67](#_Toc180103502)

[III. OUTILS D’IMPLEMENTATION 68](#_Toc180103503)

[IV. OUTILS D’ORGANISATION ET DE COLLABORATION 72](#_Toc180103504)

[V. RESULTATS DE L’IMPLEMENTATION 73](#_Toc180103505)

[CHAPITRES VII : TRANSITION 79](#_Toc180103506)

[I. TESTS 79](#_Toc180103507)

[II. DEPLOIEMENT 81](#_Toc180103508)

[V. COÛT DU PROJET 91](#_Toc180103511)

[CONCLUSION GENERALE ET PERSPECTIVES 92](#_Toc180103512)

[ANNEXES XI](#_Toc180103513)

[REFERENCES BIBLIOGRAPHIQUES XII](#_Toc180103514)

[REFERENCES WEBOGRAPHIQUES XIII](#_Toc180103515)

[GLOSSAIRE XIV](#_Toc180103516)

[TABLE DES MATIERES XV](#_Toc180103517)

# LISTE DES SIGLES ET ABREVIATIONS

**0-9**

**2TUP**: Two Tracks Unified Process

**A**

**API**: Application Programming Interface

**AXIAL** : Analyse et Conception de Système d’Information Assistés par Logiciel

**B**

**BD** : Base de Données

**C**

**CSRF**: Cross-Site Request Forgery

**CSS** : Cascading Style Sheets

**E**

**EDP-SATT** : Ecole Doctorale Polytechnique Sciences Agronomiques et Technologies de Transformation

**EDP-SIT**: Ecole Doctorale Polytechnique Sciences et Techniques de l’Ingénieur

**EFSPC**: Ecole des Formations Spécialisées et du Perfectionnement des Cadres

**ENSA**: Ecole Nationale Supérieure d’Agronomie

**ENSTP**: Ecole Nationale Supérieure des Travaux Publics

**EPGE**: Ecole Préparatoire aux Grandes Ecoles

**ESA** : Ecole Supérieure d’Agronomie

**ESAS** : Ecole Supérieure de l’Aéronautique et du Spatial

**ESCAE**: Ecole Supérieure de Commerce et d’Administration des Entreprises

**ESCPE** : Ecole Supérieure de la Chimie, du Pétrole et de l’Energie

**ESI**: Ecole Supérieure d’Industrie

**ESMG**: Ecole Supérieure des Mines et de Géologie

**ESTP**: Ecole Supérieure des Travaux Publics

**EssUP:** Essential Unified Process

**EUP:** Enterprise Unified Process

**H**

**HTML**: HyperText Markup Language

**HTTP**: HyperText Transfer Protocol

**I**

**IAB** : Institut Agricole de Bouaké

**IDE** : Environnement de Développement Intégré

**INPHB** : Institut National Polytechnique Felix Houphouët-Boigny

**INSET** : Institut National Supérieur de l’Enseignement technique

**J**

**JSF**: Java Server Faces

**JVM**: Java Virtual Machine

**Java EE** : Java Platform Entreprise Edition

**JSON** : JavaScript Object Notation

**M**

**MERISE** : Méthode d’Etude et de Réalisation Informatique pour les Systèmes d’Entreprise

**N**

**NoSQL**: Not Only SQL

**O**

**OVH**: Olé Van Herman

**P**

**PHP** : Hypertext Preprocessor

**PU** : Processus Unifié

**R**

**RAM**: Random Access Memory

**REST**: Representation State Transfer

**ROM**: Remote Procedure Call

**RUP**: Rational Unified Process

**S**

**SADT**: Structured Analysis and Design Technic

**SGBD**: Système de Gestion de Base de Données

**SGBDR**: Système de Gestion de Base de Données Relationnelle

**SI**: Système d’Information

**SQL** : Structured Query Language

**STIC** : Sciences et Technologies de l’Information et de la Communication

**U**

**UML**: Unified Modeling Language

**UP**: Unified Process

**X**

**XML**: eXtensible Mark-up Language

**XSS:** Cross-site Scripting

**XUP**: eXtreme Unified Process

# LISTE DES FIGURES

[Figure 1 : Organigramme de J3A Finance 4](#_Toc180103563)

[Figure 2 : Schéma de la méthodologie Scrum 5](#_Toc180103564)

[Figure 3: Diagramme de cas d'utilisation général "Client-Visiteur" 24](#_Toc180103565)

[Figure 4: Diagramme de cas d’utilisation général "Entreprise-Commerçant-Professionnel" 25](#_Toc180103566)

[Figure 5: Diagramme de cas d’utilisation général "Personnel" 26](#_Toc180103567)

[Figure 6 : Diagramme de séquence du cas "créer compte particulier" 37](#_Toc180103568)

[Figure 7: Diagramme de séquence du cas "créer compte business" 38](#_Toc180103569)

[Figure 8: Diagramme de séquence du cas "s'authentifier" 39](#_Toc180103570)

[Figure 9: Diagramme de séquence du cas "Recharger son compte" 40](#_Toc180103571)

[Figure 10: Diagramme de séquence du cas "Effectuer retrait" 41](#_Toc180103572)

[Figure 11: Diagramme de séquence du cas "effectuer virement" 42](#_Toc180103573)

[Figure 12: Diagramme de séquence du cas "Souscrire à une épargne" 43](#_Toc180103574)

[Figure 13: Diagramme de séquence du cas "Demander un crédit " 44](#_Toc180103575)

[Figure 14: Diagramme de séquence du cas "Valider demande de crédit " 45](#_Toc180103576)

[Figure 15: Diagramme de séquence du cas "Commander carte bancaire " 46](#_Toc180103577)

[Figure 16: Diagramme de séquence du cas "Vendre un produit " 47](#_Toc180103578)

[Figure 17: Diagramme d'activité "Créer compte de compte particulier" 49](#_Toc180103579)

[Figure 18: Diagramme d'activité "Création de compte business" 50](#_Toc180103580)

[Figure 19: Diagramme d'activité du cas "S'authentifier" 51](#_Toc180103581)

[Figure 20: Diagramme d'activité du cas "Recharger son compte" 52](#_Toc180103582)

[Figure 21: Diagramme d'activité du cas "effectuer retrait" 53](#_Toc180103583)

[Figure 22: Diagramme d'activité du cas "Effectuer virement" 54](#_Toc180103584)

[Figure 23: Diagramme d'activité du cas "Souscrire à une épargne" 56](#_Toc180103585)

[Figure 24:Diagramme d'activité du cas "Demander un crédit" 57](#_Toc180103586)

[Figure 25: Diagramme d'activité du cas "Valider un crédit" 58](#_Toc180103587)

[Figure 26: Diagramme d'activité du cas "Commander carte bancaire" 59](#_Toc180103588)

[Figure 27: Diagramme d'activité du cas "Vendre un produit" 60](#_Toc180103589)

[Figure 28: Diagramme de de classe 62](#_Toc180103590)

[Figure 29: Architecture 1-tiers 65](#_Toc180103591)

[Figure 30: Architecture 2-tiers 65](#_Toc180103592)

[Figure 31: Architecture 3-tiers 66](#_Toc180103593)

[Figure 34: Logo de PrimeFaces 71](#_Toc180103594)

[Figure 43: Capture "Interface de création de compte » version web 73](#_Toc180103595)

[Figure 44: Captures "interfaces d'authentification"" 74](#_Toc180103596)

[Figure 45: Capture "Dashboard administrateur" 74](#_Toc180103597)

[Figure 46: Capture "Dashboard client" version mobile 75](#_Toc180103598)

[Figure 47: Capture "Dashboard client" version web 75](#_Toc180103599)

[Figure 48: Capture "interface de virement" version web 76](#_Toc180103600)

[Figure 49: Capture "interface virement" version mobile 76](#_Toc180103601)

[Figure 50: Capture "simulateur de crédit" version web 77](#_Toc180103602)

[Figure 51: Capture "simulateur de crédit" version mobile 77](#_Toc180103603)

[Figure 52: Capture « interface de commande de carte bancaire » version Web 78](#_Toc180103604)

[Figure 56: Interface d'achat de nom de domaine sur Hostinger 87](#_Toc180103605)

[Figure 58: Diagramme de déploiement 90](#_Toc180103606)

[Figure 59: Diagramme de Gantt XI](#_Toc180103607)

# LISTE DES TABLEAUX

Tableau 1: Les déclinaisons du Processus Unifié [09] 15

Tableau 2: Interventions récurrentes des activités au niveau des phases 17

Tableau 3: Identification des acteurs 21

Tableau 4: Identification des cas d'utilisations 22

Tableau 5 : Description textuelle du cas "création de compte particulier" 27

Tableau 6: Description de cas d'utilisation "création de compte business" 28

Tableau 7: Description textuelle du cas d'utilisation "S'authentifier" 29

Tableau 8: Description textuelle du cas d'utilisation "recharger son compte" 30

Tableau 9: Description textuelle du cas d'utilisation "effectuer retrait" 30

Tableau 10:Description textuelle du cas d'utilisation "effectuer virement" 31

Tableau 11: Description textuelle du cas d'utilisation "souscrire à une épargne" 32

Tableau 12: Description textuelle du cas d'utilisation "demander un crédit" 33

Tableau 13: Description textuelle du cas d'utilisation "valider un crédit" 34

Tableau 14: Description de cas d'utilisation "commander une carte bancaire" 34

Tableau 15: Description textuelle du cas d'utilisation "vendre un produit" 35

Tableau 16: Tableau comparatif des architectures logicielles 66

Tableau 17: Tableau comparatif des systèmes de gestion de base de données 68

Tableau 18: Tableau comparatif des langages de programmation 69

Tableau 19: Tableau comparatif des Framework java 70

Tableau 20: Tableau comparatif des types d'hébergement 84

Tableau 21: Tableau comparatif des fournisseurs d'hébergement dédié 85

Tableau 22: Coût du projet 91

# AVANT PROPOS

La connaissance et la formation constituent des piliers du développement, du progrès et du succès économique d’un pays. C’est au regard de cette réalité et pour ne pas rester en marge du développement que la Côte d’Ivoire, dès les premières heures de son indépendance, a mis un accent particulier sur la formation et l’éducation. En effet, en moins de deux décennies d’indépendance, elle s’est dotée de structures de formation et de recherche dont l’IAB (Institut Agricole de Bouaké), l'ENSA (Ecole Nationale Supérieure d’Agronomie), l'INSET (Institut National Supérieur d’Enseignement Technique) et l'ENSTP (Ecole Nationale Supérieure des Travaux Publics). Etablissement à caractère administratif, l’Institut National Polytechnique Félix Houphouët-Boigny (INP-HB) de Yamoussoukro, créé par décret 96-678 du 04 Septembre 1996, naît de la fusion de ces quatre (4) grands établissements. Aujourd’hui, l’institut compte onze (11) grandes écoles qui sont :

* L’EPGE (Ecole Préparatoire aux Grandes Ecoles) ;
* L’EDP-STI (Ecole Doctorale Polytechnique Sciences et Techniques de l’Ingénieur) ;
* L’EDP-SATT (Ecole Doctorale Polytechnique Sciences Agronomiques et Technologies de Transformation) ;
* L’EFSPC (Ecole des Formations Spécialisées et du Perfectionnement des Cadres) ;
* L’ESA (Ecole Supérieure d’Agronomie) ;
* L’ESAS (Ecole Supérieure de l’Aéronautique et du Spatial) ;
* L’ESCAE (Ecole Supérieure de Commerce et d’Administration des Entreprises) ;
* L’ESI (Ecole Supérieure d’Industrie) ;
* L’ESMG (Ecole Supérieure des Mines et Géologie) ;
* L’ESCPE (Ecole Supérieure de la Chimie, du Pétrole et de l’Energie) ;
* L’ESTP (Ecole Supérieure des Travaux Publics).

Par ailleurs, l’ESI est chargée de la formation des ingénieurs et des techniciens supérieurs dans les principaux domaines de l’industrie, entre autres celui de l’informatique. Dans le cadre de cette formation, l’ESI initie en fin de cycle des stages pratiques en entreprise en vue d’amener ses élèves à confronter leurs connaissances acquises durant leur parcours académique aux réalités du monde socio-professionnel. C’est ainsi que J3A FINANCE nous a accueilli du 01 avril au 30 octobre 2024 afin d’effectuer le travail qui sanctionne ce mémoire.

# RESUME

Le présent rapport est le fruit d'un travail effectué au sein de la société J3A FINANCE, lors d’un stage qui s’est tenu dans la période du 01 avril au 31 Octobre 2024, en vue de l’obtention du diplôme d’ingénieur informaticien à l’issue du parcours STIC (Sciences et Technologies de l’Information et de la Communication) de l’INPHB de Yamoussoukro. Cette immersion avait pour objectif de réaliser et de déployer une application bancaire multiplateforme innovante conçue pour améliorer l’expérience utilisateur en offrant des services bancaires modernes et accessibles.

Pour modéliser notre système, faciliter l’organisation du travail et permettre un développement clair et fluide, nous nous sommes servis de la déclinaison EssUP du Processus Unifié et de son langage UML. Puis nous avons choisi l’approche agile SCRUM pour la gestion du projet.

Nous avons ensuite fait le choix du langage de programmation Java robuste, rapide et sécurisé pour la gestion de notre backend. Le Framework Spring Boot nous a servi à la création de l’API, ainsi que Maven pour la gestion des dépendances. En ce qui concerne la gestion des données, nous avons opté pour le système de gestion MySQL associé à Hibernate, qui assure le mapping entre la base de données et notre application Java.

**Mots clés** : Banque, API, EssUP, SCRUM, déploiement.

# INTRODUCTION GENERALE

Au cours des dernières décennies, nous avons été témoins de ce que les médias ont surnommé « la révolution des nouvelles technologies ». Cette révolution, caractérisée par la démocratisation de l'ordinateur personnel et l'accès massif au réseau Internet, a profondément transformé notre quotidien, marquant l'avènement de l'ère de l'informatique grand public. Aujourd'hui, ces technologies ne se contentent plus d'enrichir notre vie domestique ; elles s'imposent avec force dans la sphère professionnelle, devenant des outils indispensables à la conduite des affaires modernes.

L'évolution rapide des technologies de l'information n'a épargné aucun secteur, mais c'est peut-être dans le domaine bancaire et des services financiers que son impact se révèle le plus significatif. Loin de se cantonner à des pratiques conservatrices, ce secteur se distingue par sa capacité d'innovation continue. Les banques, qui jadis reposaient sur des systèmes traditionnels rigides, ont su tirer parti des avancées technologiques pour remodeler leurs activités, profitant pleinement de la diversité et de la richesse des nouvelles solutions numériques.

Parmi les transformations majeures qui ont redéfini le paysage bancaire, l'essor des banques en ligne se distingue comme l'une des plus grandes révolutions. Cette mutation ne concerne pas seulement les établissements bancaires eux-mêmes, mais également leurs clients, qui voient leurs interactions avec leurs institutions financières radicalement modifiées. Aujourd'hui, la banque en ligne s'impose comme une alternative crédible et compétitive à la banque traditionnelle, redéfinissant les codes du secteur bancaire.

C’est dans ce contexte d'innovation permanente et d'adaptation aux besoins modernes que nous avons eu la tâche d’assister le département informatique de J3A Finance dans son projet de : **« Réalisation et déploiement d’une application bancaire multiplateforme »**.

Dans le but de rendre compte de manière précise et analytique des six mois passés au sein de la société, il est pertinent de commencer par présenter le cadre du stage, à savoir la société J3A Finance ainsi que le contexte dans lequel le projet s’inscrit. Par la suite, nous mettrons en avant l’étude conceptuelle qui a été réalisée pour ce projet. Enfin, nous exposerons en détail la mise en œuvre de la solution développée dans le cadre de ce projet.

# PARTIE 1 : CADRE ET CONTEXTE DU PROJET

L’objet de cette partie est de réaliser une présentation générale de l’organisme d’accueil et le contexte dans lequel ce thème nous a été attribué. Cette présentation vise à fournir une compréhension claire de notre sujet.

# CHAPITRE I : PRESENTATION DE L’ORGANISME D’ACCUEIL

*Dans ce chapitre, nous présentons l’organisme d’accueil notamment son historique, ses prestations et son cadre de travail.*

## PRESENTATION DE J3A FINANCE

### Présentation

J3A PARTNERS AM est une société de gestion de Fonds de capital-risque (VC) enregistrée auprès du Financial Conduct Authrority (FCA 850258) à Londres et de l’Autorité des Marchés Financiers (EN-19000003) à Paris. Elle dispose en outre de l’agrément EUVECA délivré par l’Autorité Européenne des Marchés Financiers (ESMA). Elle dispose aussi d’un bureau de représentation à Abidjan depuis 2016 et d’un Start-up Studio sur le campus nord de l’INPHB à Yamoussoukro depuis décembre 2021.

Fondée en 2016 et basée à Abidjan, J3A FINANCE est la branche locale de J3A PARTNERS AM Sa mission principale consiste à développer des entreprises spécialisées dans les technologies financières (Fintech), les assurances (Insurtech) et la santé (Healthtech) pour le compte du European Performance Fund I et du West African Performance Fund I. Ces fonds d'investissement se consacrent spécifiquement au financement des entreprises opérant dans les secteurs des Fintech, Insurtech et Healthtech en Europe et en Afrique de l’Ouest (zone UEMOA).

### Missions et ambitions

J3A PARTNERS AM a pour objectif de donner aux entrepreneurs technologiques d'Afrique de l'Ouest toutes les ressources et l’encadrement dont ils ont besoin pour atteindre leurs objectifs. Elle fournira l'infrastructure pour construire leur entreprise.

Elle accélère leur vision et crée de la valeur grâce à un accès immédiat à un capital intelligent, un engagement inébranlable et l'accès à des talents exceptionnels. Elle a compris par expérience que toutes les start-ups ne sont pas égales, ce qui signifie que leurs objectifs et leurs parcours ne sont pas les mêmes. Adopter une approche flexible est donc au centre de sa politique d'investissement, avec un accélérateur adapté à chaque projet pour mettre en œuvre la bonne stratégie.

## ORGANISATION STRUCTURELLE DE J3A FINANCE

### Organigramme de l’entreprise

J3A FINANCE est une entreprise dirigée par un directeur général et composée de différents services, à savoir un service administratif, un service commercial, un service de gestion des ressources humaines et un service informatique dans lequel nous avons été accueilli.

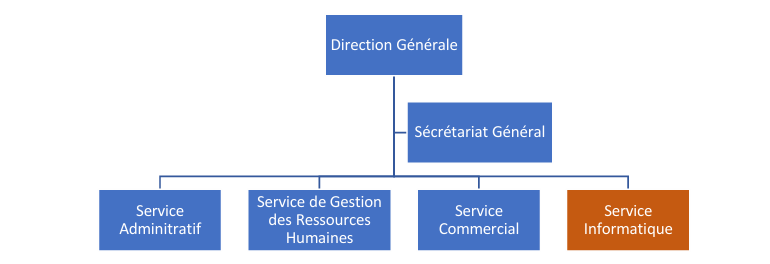


Figure 1 : Organigramme de J3A Finance

### Environnement de travail

J3A Finance accorde une attention particulière à la qualité de ses équipements, considérant cela comme un levier essentiel pour optimiser son efficacité et améliorer ses performances. Dans cette optique, l'entreprise met à la disposition de ses collaborateurs un ensemble d'équipements technologiques de pointe, tels que des ordinateurs performants et des imprimantes de haute qualité, afin de soutenir leur travail quotidien et de favoriser une meilleure productivité au sein de l'organisation.

Dans le souci d’aboutir à des produits qui donnent pleine satisfaction aux clients et aux utilisateurs, J3A FINANCE base son fonctionnement sur des méthodes agiles comme Scrum ou eXtreme Programming selon l’envergure des projets.

Une méthode Agile est une approche itérative et collaborative, capable de prendre en compte les besoins initiaux du client et ceux liés aux évolutions. Ainsi grâce à ces approches dites incrémentales, la création de produit est beaucoup plus fluide et rapide, les équipes sont autonomes, mieux orientées et travaillent plus rigoureusement. Elle permet une gestion efficace des tâches et favorise la collaboration au sein de l'équipe.

Dans la pratique, les projets sont décomposés en plusieurs phases séquentielles appelées sprints qui durent entre une (1) et quatre (4) semaines. Les sprints sont composés de réunions quotidiennes pour faire le point des réalisations de chaque journée et sont finalisés par des tests de vérification du produit.

Notons que dans le cadre de notre projet, nous avons jugé bon de procéder par la méthodologie Scrum. En utilisant cette méthodologie, nous nous assurons que le développement des applications se déroule de manière agile et efficiente, en répondant aux besoins changeants des utilisateurs et en assurant la livraison de produits de haute qualité.

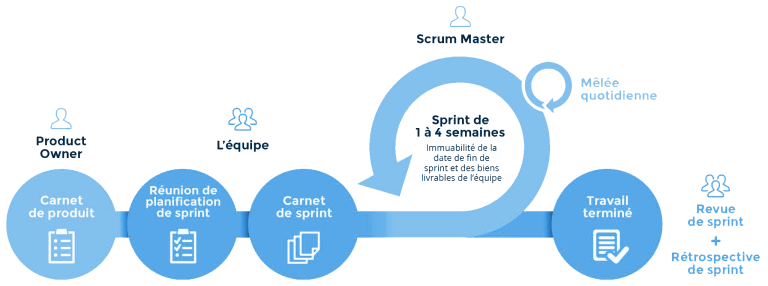


Figure 2 : Schéma de la méthodologie Scrum

Après avoir introduit l’organisme d’accueil, nous allons maintenant nous pencher plus en détail sur le projet qui a été soumis à notre étude. Nous examinerons les divers éléments qui ont conduit à sa conception, en mettant en lumière les motivations et les besoins spécifiques qui ont justifié sa mise en œuvre.

# CHAPITRE II : PRESENTATION DU PROJET

*Dans ce chapitre, nous nous concentrons sur les origines de notre thème en présentant d’abord le projet dans sa généralité et enfin l’état de l’art du projet.*

## ETUDE DE L’EXISTANT

### Présentation de l’existant

Une banque est une institution financière qui fournit des services bancaires, soit notamment de dépôt, de crédit et paiement. Le terme de banque peut désigner de façon générale le secteur bancaire.

Il existe à ce jour deux formes de banques en Côte d’Ivoire :

-Les banques traditionnelles ;

-Les néo-banques.

#### Les banques traditionnelles

Également appelée banque classique ou banque avec agences, une banque traditionnelle désigne des établissements financiers disposant de succursales physiques, souvent ancrés dans l'histoire. Ces banques, présentes sur le territoire avec des agences, ont généralement une longue expérience dans le secteur bancaire.

On distingue plusieurs types de banques, telles que les banques d'affaires, les banques dédiées aux particuliers, ainsi que les banques d'investissement. Dans la majorité des pays, ces institutions financières opèrent sous la supervision d'une autorité centrale, comme une banque centrale ou un organisme gouvernemental, qui veille à leur régulation et à la stabilité du système bancaire. On pourra notamment citer :

* Société Générale Côte d’Ivoire (SGCI) ;
* Banque Of Africa (BOA) ;
* Société Ivoirienne de Banque (SIB) ;
* NSIA Banque ;
* BICICI ;
* Banque Nationale d’Investissement (BNI).

Ces structures offrent à leurs clients une gamme de services traditionnels tels que :

* Un compte et une carte bancaire ;
* Le plus souvent un découvert autorisé ;
* Un chéquier ;
* Des comptes épargnes potentiels ;
* Des crédits ;
* Des produits d’assurances.

Le principal avantage des banques traditionnelles réside dans la possession d’agences physiques, permettant à leurs clients de se rendre sur place pour consulter un conseiller ou obtenir des informations directement auprès d'une équipe de professionnels.

Il est d’ailleurs bon de noter que la présence d’une agence à proximité de son logement est bien souvent un des motifs de choix principal de sa banque. Le fait d’avoir la possibilité de consulter son conseiller bancaire étant quelque chose d’assez rassurant. Mais c’est aussi un avantage pour les personnes peu adeptes du tout digital. En effet, il est possible de se rendre en agence pour réaliser l’intégralité de ses opérations allant du virement, à l’encaissement d’un chèque en passant par l’ouverture d’un contrat d’assurance.

#### Les néo-banques

Une néo-banque est une institution financière entièrement numérique, conçue dès le départ pour fonctionner sans agences physiques. Elle se concentre sur la fourniture de services bancaires de base, tels que les comptes courants et les paiements, à travers des plateformes mobiles ou web. Souvent, les néobanques offrent une expérience utilisateur simplifiée, des tarifs compétitifs et une grande adaptabilité. Contrairement à certaines banques en ligne, nombre d’entre elles n’ont pas de licence bancaire propre et collaborent avec des banques partenaires pour certaines opérations réglementées. On pourra citer notamment :

* Orange Bank Africapar l’opérateur téléphonique Orange**;**
* Djamo

Les néobanques présentent l'avantage de permettre l'ouverture d'un compte en quelques minutes depuis un smartphone ou un ordinateur. Peu de documents sont nécessaires, mais la plupart exigent au moins une pièce d'identité et une photo, parfois un justificatif de domicile. La facilité d'ouverture du compte est un élément clé. Une fois la demande effectuée en ligne, le compte est ouvert instantanément, et l'utilisateur reçoit sa carte bancaire gratuitement ou avec des frais réduits en quelques jours.

Le succès des néobanques repose aussi sur leurs faibles coûts de fonctionnement. Contrairement aux banques traditionnelles, elles ont moins de personnel, n'ont pas d'agences physiques, et leurs charges sont donc considérablement réduites.

### Critique de l’existant

#### Les banques traditionnelles

L’un des atouts majeurs des banques traditionnelles réside dans la proximité avec la clientèle. Cependant il faut noter une forte baisse du nombre visites en agence. Aujourd’hui, beaucoup de consommateurs et notamment les jeunes réalisent de plus en plus d’opérations directement en ligne. Il arrive ainsi fréquemment que des banques traditionnelles décident de fermer des agences pour cause de manque de fréquentation.

Les banques traditionnelles, malgré leur ancrage historique et leur proximité physique avec les clients, présentent plusieurs inconvénients qui peuvent limiter leur attrait. L'un des principaux points faibles réside dans les frais élevés qu'elles imposent, liés à la gestion de leur réseau d'agences physiques, au personnel et aux infrastructures nécessaires à leur fonctionnement. De plus, les horaires d'ouverture restreints de ces agences peuvent poser problème pour les clients ayant des emplois du temps chargés, les obligeant parfois à se déplacer à des heures peu pratiques. Les processus administratifs dans ces banques sont souvent plus longs et délicats, en raison de la nécessité de passer par plusieurs étapes manuelles, ce qui peut être frustrant pour les clients en quête de rapidité et d'efficacité.

#### Les néo-banques

Bien que les deux concepts soient relativement proches en termes de fonctionnement, la plupart des néobanques se différencient des banques en ligne par l'étendue de leurs services financiers, généralement plus restreinte. En effet, toutes ne possèdent pas de licence bancaire et se concentrent principalement sur des services tels que les paiements, les transferts de fonds internationaux ou la gestion de comptes.

En ce qui concerne la fiabilité et la confiance, les néo-banques présentent certains défis. Bien que ces institutions soient généralement bien adaptées aux besoins numériques des utilisateurs modernes, la perception de leur fiabilité peut varier. Les néo-banques ne disposent souvent pas des mêmes antécédents historiques que les banques traditionnelles, ce qui peut susciter des doutes chez certains clients quant à leur solidité financière et à leur capacité à gérer des crises économiques ou des fraudes. De plus, étant principalement des plateformes numériques, les néo-banques peuvent être perçues comme plus vulnérables aux cyberattaques ou aux interruptions de service, ce qui pourrait affecter la continuité des services et la protection des données personnelles.

## PRESENTATION GENERALE DU PROJET

### Contexte du projet

J3A FINANCE s'engage fermement à soutenir l'émergence des Fintechs, Insurtechs, et Healthtechs en Afrique de l'Ouest. Pour ce faire, l'entreprise a entrepris la conception et la mise en place d'une banque en ligne innovante, destinée à promouvoir l'inclusion financière et à faciliter l'accès aux services financiers essentiels. Une banque en ligne se distingue par son fonctionnement principalement via Internet, que ce soit à travers un site web ou une application mobile. Cette initiative a pour objectif de surmonter les défis auxquels font face de nombreuses personnes, notamment celles résidant dans des zones rurales ou issues de groupes économiquement vulnérables, en leur offrant la possibilité d'ouvrir un compte bancaire, d'accéder au crédit, ou d'effectuer des paiements électroniques. En se positionnant dans la dynamique de digitalisation croissante des services financiers, cette application vise à accroître l'accessibilité et à offrir une commodité sans précédent aux utilisateurs. En favorisant l'inclusion financière, J3A FINANCE participe activement à la réduction de la pauvreté, à la stimulation de l'activité économique, et à la promotion d'une société plus équitable. Ce projet s'inscrit également dans la transformation en cours du secteur financier, qui doit s'adapter aux besoins évolutifs des utilisateurs et à l'expansion de la digitalisation. De plus, il remédie aux insuffisances des institutions bancaires traditionnelles en Afrique de l'Ouest, telles que l'éloignement entre les clients et leurs banques, ainsi que la complexité des opérations bancaires, particulièrement pour les personnes moins expérimentées.

### Problématique du projet

Dans le contexte global, une grande partie de la population demeure exclue des services financiers traditionnels en raison de divers obstacles tels que l'éloignement géographique, le manque d'infrastructures bancaires adéquates, la complexité des processus d'ouverture de compte, et la faible littératie financière. Ces défis limitent l'accès à des services essentiels comme l'épargne, le crédit, et les paiements électroniques, freinant ainsi le développement économique et social, en particulier pour les individus vivant dans des zones reculées ou appartenant à des groupes économiquement vulnérables.

La problématique centrale de ce projet réside dans la nécessité de développer une solution bancaire qui soit à la fois accessible, simple d'utilisation, et sécurisée. Cette solution doit être capable de combler les lacunes des institutions bancaires traditionnelles, tout en s'adaptant aux besoins d'une population diversifiée. Le défi est de créer une banque en ligne qui puisse surmonter les obstacles techniques, logistiques, et réglementaires liés à la digitalisation des services financiers, tout en garantissant une adoption large et une expérience utilisateur fluide et inclusive.

### Objectifs du projet

Le principal objectif de notre travail est de développer une solution informatique qui permet de répondre aux besoins d’une banque en ligne.

De l’objectif principal, énoncé précédemment, découle plusieurs autres dits spécifiques. Les objectifs spécifiques représentent dans la pratique un éclatement de l’objectif général et permettent de l’atteindre plus facilement. Dans notre cas, ce sont les suivants :

* Développer une application qui permet aux utilisateurs d'accéder aux services bancaires via plusieurs canaux, tels que les applications mobiles et les sites web, afin de maximiser la portée et de répondre aux besoins d'une clientèle diversifiée ;
* Concevoir une interface utilisateur intuitive et personnalisée, qui s'adapte aux besoins individuels des utilisateurs, offrant ainsi une navigation simplifiée et des fonctionnalités facilement accessibles.
* Offrir des services bancaires aux individus qui sont traditionnellement sous-bancarisés ou non-bancarisés, en supprimant les barrières d'accès grâce à une plateforme accessible et conviviale ;
* Assurer la confidentialité et la sécurité des données : l'objectif est de mettre en place des mesures de sécurité robustes pour protéger les données des clients. Cela comprend le stockage sécurisé des informations personnelles, la conformité aux réglementations en matière de protection des données et la prévention des violations de sécurité ;

Garantir que la plateforme soit hautement disponible et performante, en mettant en place des solutions d'hébergement robustes.

### Contraintes et planification

Ce projet nécessite une communication entre une application web et une application mobile. Pour faciliter cette communication, il est essentiel de mettre en place une base de données centralisée. Cette interaction avec la base de données se fait par le biais d'une API RESTful, qui fournit les données à la partie frontend de l'application mobile.

De plus, la plateforme doit se conformer aux normes de sécurité les plus rigoureuses afin de protéger les informations financières des utilisateurs contre les cyberattaques, les violations de données, et les fraudes. Elle doit également garantir une gestion éthique des données, en préservant leur confidentialité et en évitant toute exploitation abusive ou non autorisée. Par ailleurs, il est impératif qu’elle reste opérationnelle 24/7, sans la moindre interruption de service.

Enfin, le délai accordé pour réaliser ce projet est de huit (08) mois. Et pour respecter le délai, nous avons dû mettre en place un planning bien avant de démarrer le projet. Ce planning est résumé dans le diagramme de planification des tâches en annexe 1.

# PARTIE 2 : ETUDE CONCEPTUELLE DU PROJET

Dans cette partie, il sera question d’une part d’une étude perspicace des besoins afin d’avoir une meilleure compréhension du système à concevoir et d’autre part de la présentation des différents diagrammes qui découlent de cette étude.

# CHAPITRE III : METHODOLOGIE DU PROJET

*L’analyse du cahier des charges de notre projet laisse entrevoir une partie non négligeable du fonctionnement du système à implémenter. Cependant, l'utilisation d'une méthode d’analyse et de conception constitue un impératif pour conduire à bien un projet de développement informatique. Par conséquent, nous effectuons ici le choix d’une méthode avant d’aborder en profondeur l’étude des besoins des utilisateurs.*

## GENERALITES SUR LES METHODES D’ANALYSES

Une méthode d'analyse et de conception informatique vise à structurer les étapes initiales du développement d'un système pour mieux répondre aux besoins d’un client. Et parmi toutes les approches existantes, les plus utilisées sont la méthode MERISE et le Processus Unifié (PU) allié au langage de modélisation UML.

### Méthodes cartésiennes

Les méthodes fonctionnelles ou cartésiennes sont les méthodes de conception de systèmes d’information de la première génération. Elles visent à décomposer une application de manière hiérarchique en plusieurs sous-applications. Les fonctionnalités de chaque sous-application sont ensuite progressivement détaillées en sous-fonctions simples, qui peuvent être facilement codées dans un langage de programmation spécifique. Un exemple de méthode utilisée dans ce contexte est le SADT (Structured Analysis and Design Technique).

### Méthodes systémiques

Les méthodes systémiques sont les méthodes de conception de systèmes d’information de la seconde génération. La démarche systémique permet de modéliser le domaine à étudier pour mieux le comprendre. Elle le décompose en sous domaines (sous-systèmes) et chaque partie est étudiée en relation avec l’ensemble pour en faire une combinaison des différentes parties. Nous pouvons citer comme exemple les méthodes MERISE et AXIAL.

### Méthodes objets

La méthode objet est une évolution de la méthode systémique, dans laquelle l'objet est considéré comme étant au cœur de tout système. Chaque objet possède un comportement vis-à-vis de son environnement et des caractéristiques qui lui sont propres. L'approche objet favorise l'indépendance entre les objets, les données et les méthodes, permettant ainsi aux programmes de partager les mêmes objets. À titre d'exemples concrets, nous pouvons mentionner le Processus Unifié (PU) utilisant le langage de modélisation UML (Unified Modeling Language) ainsi que l'Object Modeling Technique (OMT).

Lors de notre étude, nous allons accorder une attention particulière à la méthode systémique et à la méthode objet, qui occupaient une place centrale dans notre formation académique.

## PRESENTATION DE QUELQUES METHODES D’ANALYSE

### Présentation de MERISE

La méthode MERISE (Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise) faisant partie des méthodes systémiques a été développée au début des années 1970 par René Colletti et Roland Moreno. Bien qu'elle ait perdu en popularité depuis l'avènement d'UML et des approches orientées objet, MERISE reste une méthode de modélisation structurée et conceptuelle appréciée dans certains contextes spécifiques. Elle dispose de plusieurs modèles, classés en quatre niveaux : conceptuel, organisationnel, logique et physique. Elle se décline en deux (2) principales approches et sera présentée dans les sections suivantes.

#### La démarche classique

C’est la démarche « par défaut » de la méthode. La conception se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. De plus, étant donné que les données sont distinctes des traitements, il est nécessaire de vérifier leur adéquation, afin de s'assurer que toutes les données requises pour les traitements sont disponibles et qu'aucune donnée inutile n'est présente.

#### La démarche rapide

La démarche « RAD » (Rapid Application Development), apparue au début des années 90, s'est développée en réaction aux méthodes en cascade, considérées comme trop rigides et lourdes pour le développement d'applications de petite et moyenne taille. Plutôt que de s'opposer totalement à la méthode classique, elle met l'accent sur une participation active des utilisateurs, la maîtrise des coûts et des délais, un cycle itératif de conception, réalisation et amélioration, ainsi qu'un périmètre fonctionnel réduit et bien défini dès le départ du projet.

### Présentation du Processus Unifiée

Le Processus Unifié (PU), ou « Unified Process » (UP) en anglais, est une méthode de développement de logiciels orientée objets. Il se distingue par une approche itérative et incrémentale, guidée par les cas d'utilisation et axée sur l'architecture ainsi que sur les modèles UML. Ce processus englobe toutes les activités de conception et de réalisation, organisées en cycles de développement composés de quatre phases : la création, l'élaboration, la construction et la transition. Chaque phase comprend plusieurs itérations. Le Processus Unifié répète un certain nombre de fois une série de cycles constituant la vie d’un système. Tout se conclut par la livraison d’une version du produit et s’articule en quatre (04) phases :

* **Identification des besoins** : Définition du champ d’action du projet. Elle est réalisée par l’identification des cas d’utilisation. La phase d’identification des besoins définit ce que doit faire le système.
* **Analyse** : Spécification du plan du projet, des exigences et des bases de l’architecture. Le plan est spécifié en un ensemble d’itérations.
* **Conception** : Réalisation du produit.
* **Déploiement** : Test, validation et transfert du produit vers les utilisateurs finaux. A chaque phase une évaluation est faite pour la validation.

L'architecture logicielle de la méthode MERISE repose principalement sur les traitements, qui sont moins stables que les données. Cela rend cette approche moins adaptée à la conception de logiciels de grande envergure. En revanche, le Processus Unifié, étant adaptable aux spécificités du projet et de l'environnement de travail, plusieurs déclinaisons en découlent avec des caractéristiques différentes que nous présentons dans le tableau ci-dessous.

Tableau 1: Les déclinaisons du Processus Unifié [09]

|  |  |  |
| --- | --- | --- |
| Sigles | Déclinaisons | Particularités |
| PU | *Processus Unifié* | Préceptes généraux de la méthode |
| RUP | *Rational Unified Process* | Méthode propriétaire de Rational Software (IBM) sur laquelle PU a été basée |
| EUP | *Enterprise Unified Process* | Intègre les phases et les activités de post implantation pour couvrir le cycle de vie du logiciel en production jusqu’à son retrait de la production ; la dénomination est une marque commerciale. |
| XUP | *eXtreme Unified Process* | Hybride intégrant UP avec extreme Programming. |
| AUP | *Agile Unified Process* | Un hybride combinant certains éléments de l'Unified Process (UP) avec des principes des méthodes agiles, diffusé sous forme de site web documentant la méthode, mais abandonné depuis 2006 |
| 2TUP | *two Tracks Unified Process* | Capable de prendre en compte les imprévus et les contraintes liés aux changements rapides et continus des systèmes d’information. |
| EssUP | Essential Unified Process | Proposé par la société Ivar Jacobson Consulting, c’est une version allégée du processus unifié (UP) qui intègre des principes agiles. Il se compose d'un ensemble de pratiques dites « essentielles » que l'on peut sélectionner et combiner librement, offrant ainsi une grande flexibilité d'utilisation. |

### Présentation de la déclinaison EssUP

EssUP est un processus conçu par IJI (Ivar Jacobson International) pour améliorer le développement des logiciels modernes. C’est en fait une méthode qui intègre avec prudence les pratiques du Processus Unifié et des méthodes agiles, possédant ainsi un certain nombre de pratiques simples et éprouvées qui peuvent être utilisées comme base pour tout type de développement. Par ailleurs, il se concentre sur les éléments essentiels applicables aux projets, fournit des conseils sur la mise en œuvre d'une approche cohérente, met l'accent sur l'amélioration des compétences des personnes impliquées dans le développement et ajoute suffisamment de mécanismes pour réduire les risques liés aux projets.

Le cycle de vie de la méthode EssUP hérite complètement des phases du Processus Unifié. Il prévoit ainsi les étapes suivantes :

* **L’initialisation** : il s’agit ici de lever les ambiguïtés sur les besoins et les exigences du système à concevoir, d’en identifier les acteurs et d’en délimiter la portée ;
* **L’élaboration** : elle permet de préciser la plupart des cas d'utilisation et de concevoir l'architecture du système. Au terme de cette phase, l’on doit être à mesure de prévoir les ressources nécessaires à l’achèvement du projet ;
* **La construction** : elle est principalement axée sur l’implémentation. Son objectif est de produire un logiciel qui répond aux besoins prioritaires des utilisateurs. A la fin de cette phase, les développeurs doivent fournir une version exécutable du système ;
* **La transition** : c’est la phase qui finalise le produit. Il s'agit de s'assurer que le système répond réellement aux attentes des utilisateurs, de repérer les éventuelles défaillances, de combler les lacunes et d'ajuster le produit en fonction du contexte de travail des utilisateurs.

Les phases précitées répètent à chaque itération des pratiques appelées activités. L’on distingue six (6) activités qui sont :

* **L’identification des besoins** : cette activité permet d’inventorier les besoins principaux du système à concevoir. De plus, elle permet de recenser les besoins fonctionnels (qui conduisent à l'élaboration des modèles de cas d'utilisation) ainsi que les besoins non fonctionnels ;
* **L’analyse** : Son objectif est d'obtenir une compréhension approfondie des besoins et des attentes du client. Elle produit une spécification détaillée des exigences basées sur les cas d'utilisation, en les organisant de manière à rendre le futur système facilement compréhensible ;
* **La conception** : Elle offre une bonne compréhension des contraintes associées à la phase de construction. En effet, elle sert de point de départ pour l'implémentation en créant une abstraction claire et transparente de cette dernière ;
* **L’implémentation** : Il s'agit de l'implémentation du système en code source, scripts, exécutables et d'autres éléments comparables. Les objectifs clés de cette étape incluent la planification de l'intégration des composants pour chaque itération, ainsi que la production de classes et de sous-systèmes en code source ;
* **Les tests** : Ils permettent de valider les résultats de l'implémentation en évaluant la structure. Cette activité se concentre sur l'assurance qualité du logiciel ;
* **Le déploiement** : c’est un ensemble d’étapes qui visent à placer le logiciel dans son environnement cible afin qu'il soit prêt à être utilisé.

Notons que le PU gère le processus de développement selon deux (2) axes : un axe vertical qui représente les enchaînements d'activités et un axe horizontal représentant le temps et les enchaînements de phases. Le tableau ci-dessous présente la façon dont les activités interviennent le plus souvent au niveau des phases :

Tableau 2: Interventions récurrentes des activités au niveau des phases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Initialisation | Elaboration | Construction | Transition |
| Besoins | X |  |  |  |
| Analyse |  | X |  |  |
| Conception |  | X |  |  |
| Implémentation |  |  | X |  |
| Tests |  |  |  | X |
| Déploiement |  |  |  | X |

Le tableau ci-dessus est une présentation simplifiée des phases et activités du processus. Il montre, en effet, les manifestations récurrentes des activités au niveau des phases et non leurs uniques interventions.

En outre, le Processus Unifié étant itératif et incrémental, plusieurs itérations peuvent avoir lieu dans une même phase. Et durant chaque itération, plusieurs activités du processus peuvent être effectuées. Pour que notre travail soit facilement compréhensible, nous présenterons les phases d’EssUP dans leur ordre usuel en indiquant à chaque fois les différentes itérations ayant contribué à la réalisation desdites phases.

La méthode de gestion du projet étant bien définie, nous allons donc présenter dans les lignes qui suivent, son application sur le projet.

# CHAPITRE IV : INITIALISATION

*L’initialisation est la première phase du processus EssUP. Il s’agit ici de lever les ambiguïtés sur les besoins et les exigences du système à concevoir, d’en identifier les acteurs et d’en délimiter la portée.*

## SPECIFICATIONS DES BESOINS

### Spécifications fonctionnelles

Cette section est dédiée à la description des exigences fonctionnelles pour les différents utilisateurs de l'application.

* **L’ouverture de compte en ligne** : permettre aux clients de créer un compte bancaire ;
* **La gestion des comptes** : Avoir la possibilité de consulter ses comptes, mettre à jour certaines informations d’un compte, restreindre un compte ou le fermer ;
* **La réalisation des traitements** : Avoir la possibilité de réaliser certains traitements comme les virements (internes, externe, programmés ou permanents), les retraits, les dépôts, les achats ou les paiements ;
* **La gestion des produits bancaires d’épargne** : permettre aux clients de souscrire aux différentes épargnes proposées par la banque ;
* **La gestion des prêts**: Réaliser des demandes de prêts, les analyser en vue de les valider ou pas, rembourser manuellement ou automatiquement des prêts ;
* **La gestion des cartes bancaires** : Avoir la possibilité de visualiser, paramétrer, bloquer ou encore ajuster les limites de paiements et de retraits des cartes bancaires ;
* **La gestion des paramètres bancaires** : Les administrateurs doivent pouvoir paramétrer les informations utiles au fonctionnement de la banque ;
* **Le service d’aide à la clientèle** : Répondre aux préoccupations des clients soit par appel téléphonique, soit par messagerie instantanée, soit par un assistant virtuel gouverné par une intelligence artificielle ;
* **La gestion du personnel**: administrateurs doivent pouvoir gérer (ajouter, supprimer ou modifier) le personnel de la banque.
* **Alerte notifications push pour les transactions et évènements importants** : Les utilisateurs doivent recevoir des notifications pour des activités spécifiques sur leurs comptes.

### Spécifications non-fonctionnelles

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

* **La confidentialité** : Les informations destinées à chaque utilisateur ne peuvent être consultées que par les personnes dûment autorisées ;
* **L’authenticité** : Les utilisateurs doivent prouver leur identité par l’usage de codes d’accès. Cela permet de gérer les droits d’accès aux ressources concernées ;
* **L’intégrité** : Les données doivent correspondre aux attentes des utilisateurs et ne doivent pas être modifiées de manière accidentelle, illégale ou malveillante. En d'autres termes, les informations traitées doivent être précises et complètes ;
* **La disponibilité** : L’accès aux ressources du système d’information doit être permanent et sans faille durant les plages d’utilisation prévues. Les services et ressources sont accessibles rapidement et régulièrement ;
* **La non-répudiation** : Aucun utilisateur ne doit être en mesure de contester les opérations qu'il a effectuées dans le cadre de ses actions autorisées, et aucun tiers ne doit pouvoir revendiquer les actions d'un autre utilisateur.
* **La convivialité de l’interface graphique** : l’application doit fournir une interface conviviale et simple pour tout type d’utilisateur car elle présente le premier contact de l’utilisateur avec l’application et par le biais de celle-ci on découvrira ses fonctionnalités.

## DELIMITATION DU SYSTEME

### Identification des acteurs

Un acteur est un utilisateur type de système. Il représente une responsabilité par rapport au système ou un rôle plutôt qu’une personne physique. Il est donc une entité externe qui interagit directement avec le système en émettant et en recevant des messages.

Le système comporte quatre groupes d’utilisateurs à savoir :

* Les visiteurs ;
* Les clients (Particuliers, entreprises, commerçants et professionnels) ;
* Le personnel (Administrateurs, analystes crédit et conseillers clients) ;
* Les caissiers.

Tableau 3: Identification des acteurs

|  |  |
| --- | --- |
| Acteurs | Rôles |
| Visiteur | Les visiteurs sont des utilisateurs non enregistrés qui explorent la plateforme pour en savoir plus sur les services bancaires proposés. |
| Particulier | Les particuliers sont des individus qui utilisent la plateforme pour gérer leurs finances personnelles. |
| Entreprise | Les entreprises représentent les organisations ou sociétés qui utilisent la plateforme pour gérer leurs finances professionnelles. |
| Commerçant | Les commerçants sont des petits entrepreneurs ou indépendant qui gèrent leurs activités professionnelles et utilisent des solutions de paiement adaptées. |
| Professionnel | Les professionnels regroupent les travailleurs indépendants qui utilisent la plateforme pour gérer leurs activités professionnelles. |
| Administrateur | Les administrateurs sont responsables de la supervision générale de la plateforme. Ils ont un accès complet aux paramètres et fonctionnalités, leur permettant de gérer la sécurité, d'assurer la conformité aux réglementations, et de configurer les services offerts. |
| Conseiller client | Les conseillers clients sont au cœur de la relation avec les clients. Ils accompagnent les clients dans la gestion de leurs comptes et les orientent vers des produits bancaires adaptés à leurs besoins spécifiques. |
| Analyste crédit | Les analystes crédit sont chargés d'évaluer les demandes de prêt soumises par les clients. Leur travail consiste à analyser les dossiers de crédit, à évaluer la solvabilité des emprunteurs et à déterminer les risques associés à chaque prêt. |
| Caissier | Le caissier est affecté à un point de vente et est chargé d’alimenter le compte des clients ou de les fournir de l’argent en espèces débité de leur solde virtuel. |

### Identification des cas d’utilisation

Un cas d’utilisation est un texte qui décrit l’interaction et les dialogues entre l’acteur et le système. Les cas d’utilisations sont une technique puissante pour consigner et traduire le comportement détaillé du système.

Le tableau ci-dessous répertorie les différentes façons dont les acteurs précédemment identifiés utilisent le système :

Tableau 4: Identification des cas d'utilisations

|  |  |
| --- | --- |
| Acteurs | Cas d’utilisation |
| Visiteur | * S’inscrire * Consulter les informations sur les produits de la banque * S’inscrire aux newsletters |
| Client | * (Tous les cas d’utilisation du visiteur) * S’authentifier * Changer de plan * Gérer ses comptes (consulter, alimenter, fermer) * Effectuer un retrait * Recharger son compte * Effectuer un virement * Programmer un virement * Demander un prêt * Rembourser manuellement un prêt * Simuler une demande de crédit * Souscrire aux produits bancaires d’épargnes * Résilier une épargne * Commander une carte bancaire * Paramétrer une carte * Gérer un bénéficiaire (ajouter, modifier et supprimer) * Imprimer son RIB * Imprimer son relevé de compte * Contacter le service client * Trouver une agence |
| Professionnel | * Gestion des clients * Créer un devis * Créer une facture * Envoyer un devis * Envoyer une facture * Justifier une sortie * Catégoriser une transaction |
| Entreprise | * (Tous les cas d’utilisations du professionnel) * Gérer ses employés (ajouter, modifier, supprimer) * Attribuer une carte bancaire à un employé * Payer un salaire |
| Commerçant | * (Tous les cas d’utilisations du professionnel) * Vendre un produit * Imprimer son relevé de vente * Consulter statistique de vente |
| Caissier | * S’authentifier * Effectuer un retrait * Effectuer un dépôt |
| Conseiller client | * S’authentifier * Discuter avec un client * Valider la création de compte * Imprimer code QR commerçant |
| Analyste crédit | * S’authentifier * Valider une demande de prêt |
| Administrateur | * (Tous les cas d’utilisations des analystes crédits) * (Tous les cas d’utilisations des conseillers client) * Gérer les clients et les caissiers * Gérer le personnel * Gérer les produits bancaires * Gérer les plans * Fermer un compte * Restreindre un compte * Paramétrer les taux * Consulter les statistiques |
| Super administrateur | * (Tous les cas d’utilisations de l’administrateur) * Gérer les administrateurs * Gérer les comptes VIP * Gérer les rôles et permissions des utilisateurs |

### Diagramme de cas d’utilisation

Un diagramme des cas d’utilisation (DCU) représente le comportement d’un système, d’un sous-système, d’une classe ou d’un composant tel qu'il est perçu par un utilisateur externe. Il décompose les fonctionnalités du système en unités cohérentes appelées cas d'utilisation, qui sont significatives pour les acteurs impliqués. Ces cas d'utilisation permettent de traduire les besoins des utilisateurs d'un système, offrant ainsi une perspective centrée sur l'utilisateur.

Dans les diagrammes ci-dessous, les acteurs principaux sont représentés par des bonhommes, les acteurs secondaires par des rectangles, les cas d’utilisation par des ellipses, les relations élémentaires entre acteurs et cas par des traits simples et les relations d’inclusion par des traits accompagnés de la mention « include ».

#### 3.1. Diagramme de cas d’utilisation général « visiteur-Client »

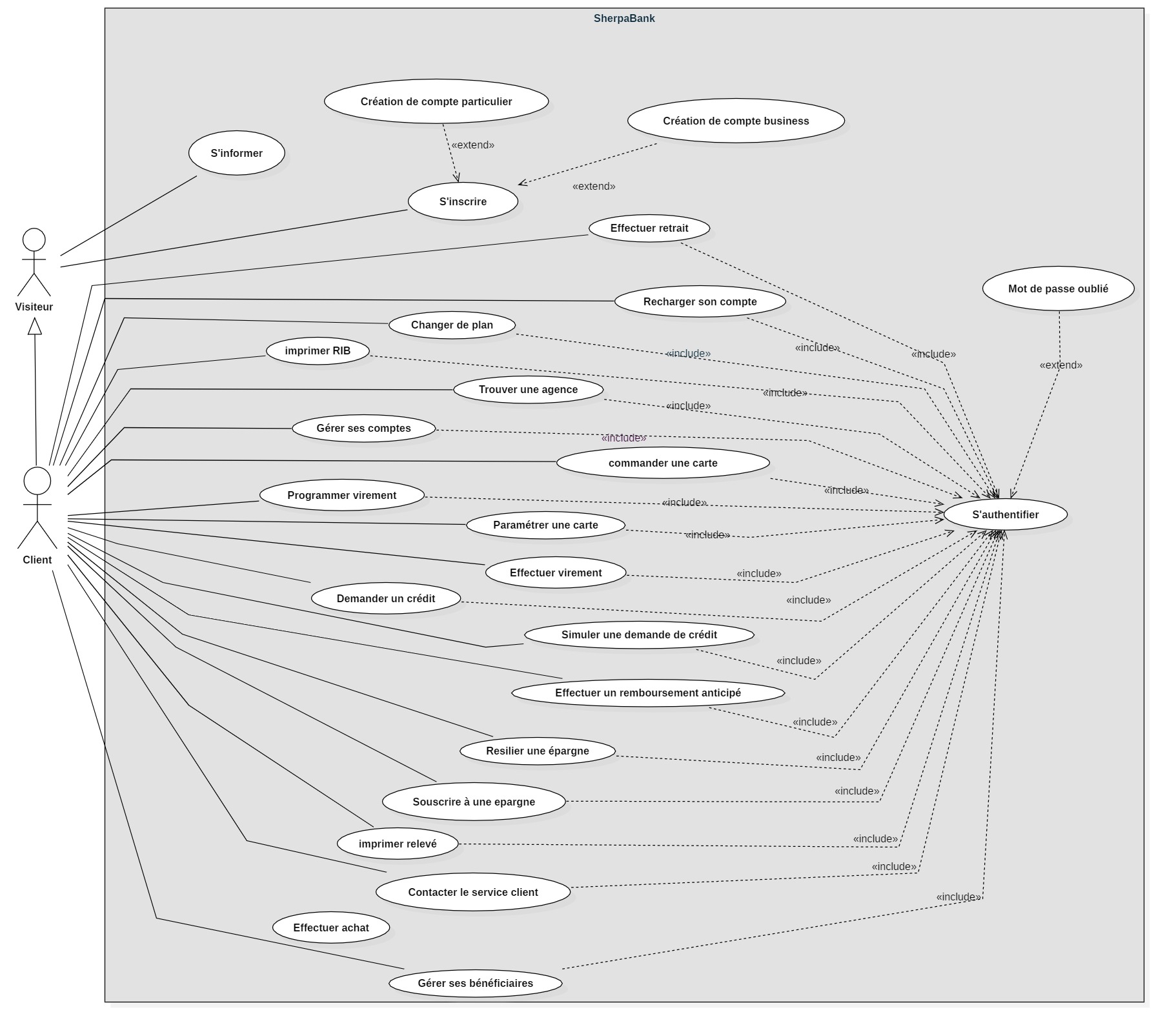


Figure 3: Diagramme de cas d'utilisation général "Client-Visiteur"

#### Diagramme de cas d’utilisation général « Entreprise-Commerçant-Professionnel »

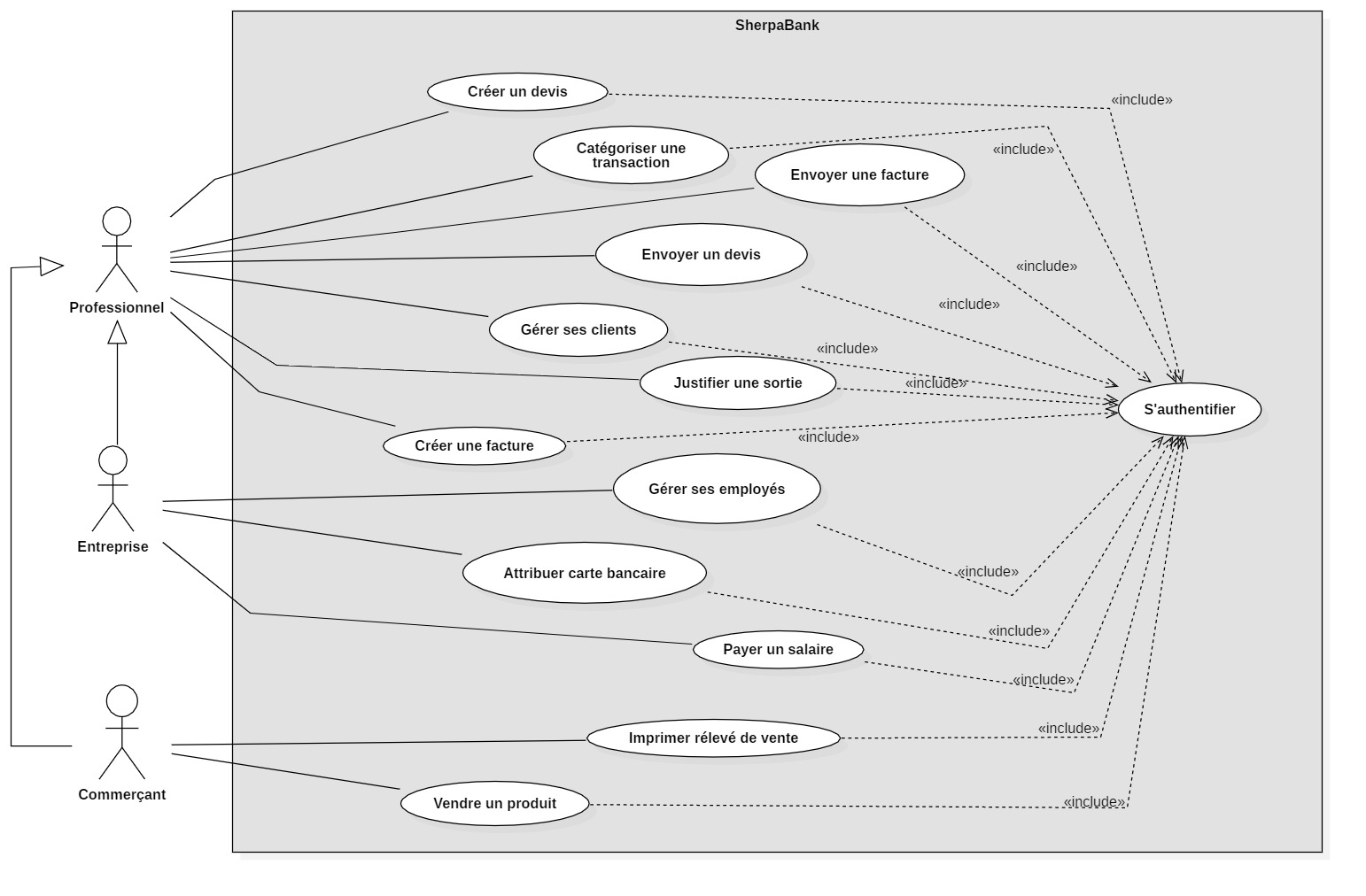


Figure 4: Diagramme de cas d’utilisation général "Entreprise-Commerçant-Professionnel"

#### Diagramme de cas d’utilisation général « Personnel »

# CHAPITRE V : ELABORATION

Figure 5: Diagramme de cas d’utilisation général "Personnel"

*L'élaboration constitue la deuxième étape de la méthode EssUP. Elle vise à détailler la majorité des cas d'utilisation et à concevoir l'architecture du système. À l'issue de cette phase, il doit être possible d'estimer les ressources requises pour finaliser le module.*

## ANALYSE : DESCRIPTION TEXTUELLE DES CAS D’UTILISATIONS

La description textuelle d’un cas d’utilisation est une fiche descriptive de ce cas d’utilisation. Elle permet de :

* Clarifier le déroulement de la fonctionnalité ;
* Décrire la chronologie des actions qui devront être réalisées ;
* D’identifier les parties redondantes pour en déduire des cas d’utilisation plus précis qui seront utilisés par inclusion, extension ou généralisation/spécialisation
* D’indiquer d’éventuelles contraintes déjà connues et dont les développeurs vont devoir tenir compte lors de la réalisation du logiciel.

Ces contraintes peuvent être de nature diverse. Pour chacun des cas d’utilisation préalablement identifiés, nous présentons une fiche de description à l’aide des tableaux suivants.

### Cas « S’inscrire »

#### Création de compte particulier

Tableau 5 : Description textuelle du cas "création de compte particulier"

|  |  |
| --- | --- |
| Titre | Créer un compte particulier |
| Résumé : Décrit le processus de création d’un compte particulier | |
| Acteurs principaux | Visiteur, Client |
| Acteur secondaire | Service de messagerie |
| **Descriptions des enchaînements** | |
| Précondition(s) | L’utilisateur n’a pas de compte particulier |
| Postcondition(s) | Le compte particulier est créé |
| Scénario nominal | 1.L’utilisateur clique sur le bouton de création de compte particulier  2.Le système affiche le formulaire de création d’un nouveau compte particulier  3.L’utilisateur remplit le formulaire et clique sur le bouton de validation  4.Le système vérifie les informations saisies  5.Le système envoie un code OTP par SMS au numéro de téléphone saisit par l’utilisateur  6.Le client renseigne le code OTP et confirme toutes les informations  7.Le système vérifie le code OTP saisit  8.Le système crée un nouveau compte particulier pour l’utilisateur et le redirige vers la page de connexion avec un message de bienvenue |
| **Scénario alternatif** | |
| A5 : Le numéro d’identification existe déjà dans la base de données | 1.Le système affiche un message d’erreur  2.Retour au point 2 du scénario nominal |

#### Création de compte Business

Tableau 6: Description de cas d'utilisation "création de compte business"

|  |  |
| --- | --- |
| Titre | Créer un compte business |
| Résumé : Décrit le processus de création d’un compte business | |
| Acteur principal | Visiteur, Client |
| Acteurs secondaires | Service de messagerie, Conseiller client |
| **Description des enchainements** | |
| Précondition(s) | L’utilisateur n’a pas de compte business |
| Postcondition(s) | Le compte business de l’utilisateur est créé |
| Scénario nominal | 1.L’utilisateur clique sur le bouton de création de compte et choisit le bon type de compte business  2.Le système affiche le formulaire de création associé à la sélection de l’utilisateur  3.L’utilisateur remplit le formulaire ainsi que les documents demandés et clique sur le bouton de validation  4.Le système vérifie les informations saisies  5. Le système envoie un code OTP par SMS au numéro de téléphone saisit par l’utilisateur  6.Le client renseigne le code OTP et confirme toutes les informations pour effectuer la demande de création de compte business  7.Le système vérifie le code OTP saisit  8. Le système informe le client par SMS que sa demande a été prise en compte et le redirige vers la page d'accueil.  9.Le conseille client consulte la liste des demandes de création de compte business  10.Le conseiller client valide la demande de l’utilisateur à travers la console d’administration  11.Le système transmet un lien à l’utilisateur par mail ou par SMS pour la finalisation de création de compte  12.L’utilisateur clique sur lien, remplit le formulaire associé et clique sur le bouton de validation  13.Le système vérifie les informations saisies  14. Le système crée un nouveau compte business pour l’utilisateur et le redirige vers la page de connexion avec un message de bienvenue |
| **Scénarios alternatifs** | |
| A5 : Le numéro d’identification existe déjà dans la base de données | 1.Le système affiche un message d’erreur  2.Retour au point 2 du scénario nominal |
| A11 : Les documents fournis par l’utilisateur sont invalides ou illisibles | 1.Le conseiller client rejette la demande en entrant le motif de rejet  2.Le système transmet le motif de rejet par mail ou par SMS a l’utilisateur  3.Le système met fin au processus |

### Cas « S’authentifier »

Tableau 7: Description textuelle du cas d'utilisation "S'authentifier"

|  |  |
| --- | --- |
| Titre | S’authentifier |
| Résumé : Processus d’authentification au système qui intervient à chaque lancement de l’application. | |
| Acteur principal | Client, Personnel, Caissier |
| Acteur secondaire |  |
| **Description des enchainements** | |
| Précondition(s) | L’utilisateur a un compte |
| Postcondition(s) | L’utilisateur est connecté à la plateforme |
| Scénario nominal | 1.L’utilisateur clique sur le bouton « se connecter »  2.Le système affiche le formulaire de connexion  3.L’utilisateur saisit son numéro de téléphone et son mot de passe et clique sur valider  4.Le système vérifie les informations saisies  5.Le système affiche le tableau de bord associé |
| **Scénarios alternatifs** | |
| A5 : Le informations saisies sont erronées | 1.Le système empêche la progression  2.Retour au point 2 du scénario nominal |

### Cas « Recharger son compte »

Tableau 8: Description textuelle du cas d'utilisation "recharger son compte"

|  |  |
| --- | --- |
| Titre | Recharger son compte |
| Résumé : Décrit le processus d’alimentation d’un compte. | |
| Acteur principal | Client |
| Acteur secondaire | Caissier |
| **Description des enchainements** | |
| Précondition(s) | Le client et le caissier sont authentifiés |
| Postcondition(s) | -Le compte du client est alimenté  -Le caissier est débité du montant du rechargement |
| Scénario nominal | 1.Le caissier clique sur le bouton « alimenter un compte »  2.Le système affiche le formulaire de rechargement de compte  3.Le client clique sur le bouton « dépôt et retrait »  4.Le système affiche le code QR du client  5.Le caissier scanne le code QR du client, saisit le montant du rechargement et confirme la transaction  6.Le système effectue une vérification  7.Le système déduit le montant saisit sur le compte du caissier et met à jour le compte du client  8.Le système envoie une notification de succès à chacun des acteurs. |
| **Scénario alternatif** | |
| A7 : Le caissier ne dispose pas de fonds nécessaires pour effectuer la transaction | 1.Le système affiche un message d’erreur  2.Retour au point 2 du scénario nominal |

### Cas « Effectuer un retrait »

Tableau 9: Description textuelle du cas d'utilisation "effectuer retrait"

|  |  |
| --- | --- |
| Titre | Effectuer un retrait |
| Résumé : Décrit le processus d’alimentation d’un compte. | |
| Acteur principal | Client |
| Acteur secondaire | Caissier |
| **Description des enchainements** | |
| Précondition(s) | - Le client et le caissier sont authentifiés  - Le client doit avoir suffisamment de fonds disponibles pour effectuer le retrait |
| Postcondition(s) | -Le compte du client est débité du montant du retrait  -Le compte du caissier est alimenté |
| Scénario nominal | 1. Le caissier clique sur le bouton « initier un retrait »  2. Le système affiche le formulaire de retrait  3. Le client affiche clique sur le bouton d’affichage de code QR  4. Le système affiche le code QR du client  5. Le caissier scanne le code QR du client, saisit le montant du retrait et confirme la transaction  6. Le système envoie un message au client concernant la demande de retrait ainsi qu’un formulaire de confirmation  7. Le client saisit son mot de passe pour confirmer le retrait  8. Le système vérifie le mot de passe saisit  9. Le système vérifie les informations du client  10. Le système débite le compte du client et met à jour le compte du caissier  11. Le système envoie une notification de succès à chacun des acteurs. |
| **Scénarios alternatifs** | |
| A7 : Le mot de passe est erroné après trois tentative | Le système annule la transaction |
| A8 : Le montant du retrait est supérieur au montant minimum requis | 1. Le système envoie un message d’erreur  2. Le système annule la transaction |

### Cas « Effectuer un virement »

Tableau 10:Description textuelle du cas d'utilisation "effectuer virement"

|  |  |
| --- | --- |
| Titre | Effectuer un virement |
| Résumé : Ce cas détaille le processus par lequel un utilisateur peut transférer de l'argent de son compte vers un autre. | |
| Acteur principal | Client |
| Acteur secondaire |  |
| **Description des enchainements** | |
| Précondition(s) | - Le client émetteur est authentifiée  - Le compte du destinataire existe |
| Postcondition(s) | - Le compte de l’émetteur est débité du montant de transfert  - Le compte du destinataire est crédité du montant de transfert |
| Scénario nominal | 1. Le client intègre la section virement de l’application et déclenche une demande de virement.  2. Le système affiche un formulaire dans lequel le client peut sélectionner le compte de départ, le compte de destination et le montant à transférer.  3. Le client entre les informations nécessaires dans le formulaire.  4. Le client valide et soumet le formulaire  5. Le système vérifie les informations  6. Le système effectue le transfert et met à jour les soldes des comptes concernés  7. Le système envoie une notification de succès aux propriétaires des comptes de la transaction. |
| **Scénario alternatif** | |
| A5 : Le compte de départ n'a pas suffisamment de fonds | 1. Le système affiche un message d'erreur au client  2. Retour au point 2 du scénario nominal |

### Cas « Souscrire à une épargne »

Tableau 11: Description textuelle du cas d'utilisation "souscrire à une épargne"

|  |  |
| --- | --- |
| Titre | Souscrire à une épargne |
| Résumé : Décrit le processus de souscription à une épargne | |
| Acteur principal | Client |
| Acteur secondaire |  |
| **Description des enchainements** | |
| Précondition(s) | - Le client est authentifié  -Le client est sur la page de souscription aux produits bancaires d’épargne |
| Postcondition(s) | -Le compte du client est débité du montant de la souscription  -Le compte épargne du client est créé |
| Scénario nominal | 1.Le client choisit le type d’épargne auquel il veut souscrire  2.Le système affiche le formulaire de souscription  3. Le client remplit le formulaire et clique sur le bouton de validation  4.Le système vérifie les informations  5.Le système crée le compte épargne du client et lui envoie un message de succès |
| **Scénarios alternatifs** | |
| A5 : Le client ne dispose pas de fonds nécessaires pour la souscription | 1.Le système affiche un message d’erreur  2.Retour au point 2 du scénario nominal |
| A5 : Le données saisit par le client sont erronées | 1.Le système affiche un message d’erreur  2.Retour au point 3 du scénario nominal |

### Cas « Demander un crédit »

Tableau 12: Description textuelle du cas d'utilisation "demander un crédit"

|  |  |
| --- | --- |
| Titre | Demander un crédit |
| Résumé : Décrit le processus de demande de crédit d’un client | |
| Acteur principal | Client |
| Acteur secondaire |  |
| **Description des enchainements** | |
| Précondition(s) | - Le client est authentifié  - Le client n’a pas de demande de crédit en cours  - Le client n’a pas de crédit à rembourser |
| Postcondition(s) | La demande de crédit est effectuée |
| Scénario nominal | 1. Le client choisit sur le type de crédit voulu et clique sur le bouton « faire la demande »  2. Le système affiche le formulaire de demande de crédit  3. Le client remplit le formulaire et le soumet  4. Le système vérifie les informations saisies  5. Le système vérifie l’éligibilité du client  6. Le système enregistre la demande  7. Le système envoie un message de succès |
| **Scénarios alternatifs** | |
| A5 : Le client ne respecte les conditions d’éligibilité | 1. Le système affiche un message d’erreur  2. Retour au point 2 du scénario nominal |
| A6 : Le montant saisit est inférieur au montant maximum de l’emprunt automatique | 1. La demande de prêt est acceptée et le compte courant du client est crédité  2. Le système envoie une notification de succès au client |

### Cas « valider un crédit »

Tableau 13: Description textuelle du cas d'utilisation "valider un crédit"

|  |  |
| --- | --- |
| Titre | Valider un crédit |
| Résumé : Décrit le processus de demande de crédit d’un client | |
| Acteur principal | Analyste crédit |
| Acteur secondaire | Client |
| **Description des enchainements** | |
| Précondition(s) | - L’analyste crédit est authentifié  - La demande de crédit du client a été enregistré |
| Postcondition(s) | Le crédit est accordé ou refusé |
| Scénario nominal | 1. L’analyste crédit clique sur la demande de prêt à analyser  2. Le système affiche la demande de façon détaillée avec toutes les informations de la demande et du client.  3. L’analyse crédit analyse la demande du client  4. L’analyste crédit accepte ou refuse le prêt en cliquant sur un bouton.  5. Le système met à jour le statut de la demande (accordé ou refusé).  6. Le système renvoie le résultat de la demande au client à travers une notification |
| **Scénario alternatif** | |
| A3 : Les documents fournis sont insuffisants, illisibles ou invalides | 1.L’analyste crédit envoie un message au client à travers le système pour lui demander de mettre à jour sa demande  2.Le client met à jour sa demande  3. Retour au point 1 |

### Cas « Commander une carte bancaire »

Tableau 14: Description textuelle du cas d'utilisation "commander une carte bancaire"

|  |  |
| --- | --- |
| Titre | Commander une carte bancaire |
| Résumé : Décrit le processus de commande carte bancaire | |
| Acteur principal | Client |
| Acteur secondaire |  |
| **Description des enchainements** | |
| Précondition(s) | - Le client est authentifié  - La client est dans la section des cartes bancaires  - Le client ne dispose pas de la carte bancaire souhaité |
| Postcondition(s) | Le client dispose d’une carte bancaire virtuelle |
| Scénario nominal | 1. Le client clique sur le bouton « ajouter une carte bancaire »  2. Le système affiche un modal avec la liste des différentes cartes bancaires auxquels il a accès en fonction de son plan  3. Le client sélectionne la carte qui l’intéresse et soumet son choix  4. Le système effectue une vérification  5.Le système débite le montant de la carte sur le compte du client  5.Le système attribue au client une carte virtuelle en attendant la livraison de la carte physique  7.Le système envoie un message de succès |
| **Scénario alternatif** | |
| A5 : Le client ne dispose pas de fonds nécessaire | 1.Le système envoie un message d’erreur au client  2.Retour au point 2 du scénario mobile |

### Cas « Vendre un produit »

Tableau 15: Description textuelle du cas d'utilisation "vendre un produit"

|  |  |
| --- | --- |
| Titre | Vendre un produit |
| Résumé : Décrit le processus de vente d’un produit | |
| Acteur principal | Commerçant |
| Acteur secondaire | Client |
| **Description des enchainements** | |
| Précondition(s) | - Le commerçant et le client sont authentifiés |
| Postcondition(s) | -Le compte commerçant est crédité  -Le compte client est débité |
| Scénario nominal | 1. Le commerçant clique sur le bouton « vendre un produit »  2. Le système affiche un formulaire de vente  3. Le commerçant remplit le formulaire et le soumet  4. Le système génère un code QR  5.Le client scanne le code et confirme la transaction  6.Le système effectue une vérification  7.Le système débite le compte client et crédite le compte commerçant  9.Le système envoie un message de succès |
| **Scénario alternatif** | |
| A6 : Le client ne dispose pas de fonds nécessaire | 1.Le système envoie un message d’erreur au client  2.Retour au point 2 du point nominal |

## ELABORATION DES DIAGRAMMES DE SEQUENCES ET D’ACTIVITES

### Diagrammes de séquence

Les diagrammes de séquence représentent les interactions entre différents objets dans un seul cas d'utilisation. Ils montrent comment les éléments d'un système collaborent pour accomplir une fonction, ainsi que l'ordre dans lequel ces interactions se déroulent lors de l'exécution d'un cas d'utilisation spécifique.

Chaque diagramme est structuré en une ligne temporelle, débutant en haut et se déroulant de haut en bas pour illustrer la succession des interactions. Chaque objet est associé à une colonne, tandis que les messages échangés sont symbolisés par des flèches.

#### Cas « S’inscrire »

##### Création de compte particulier

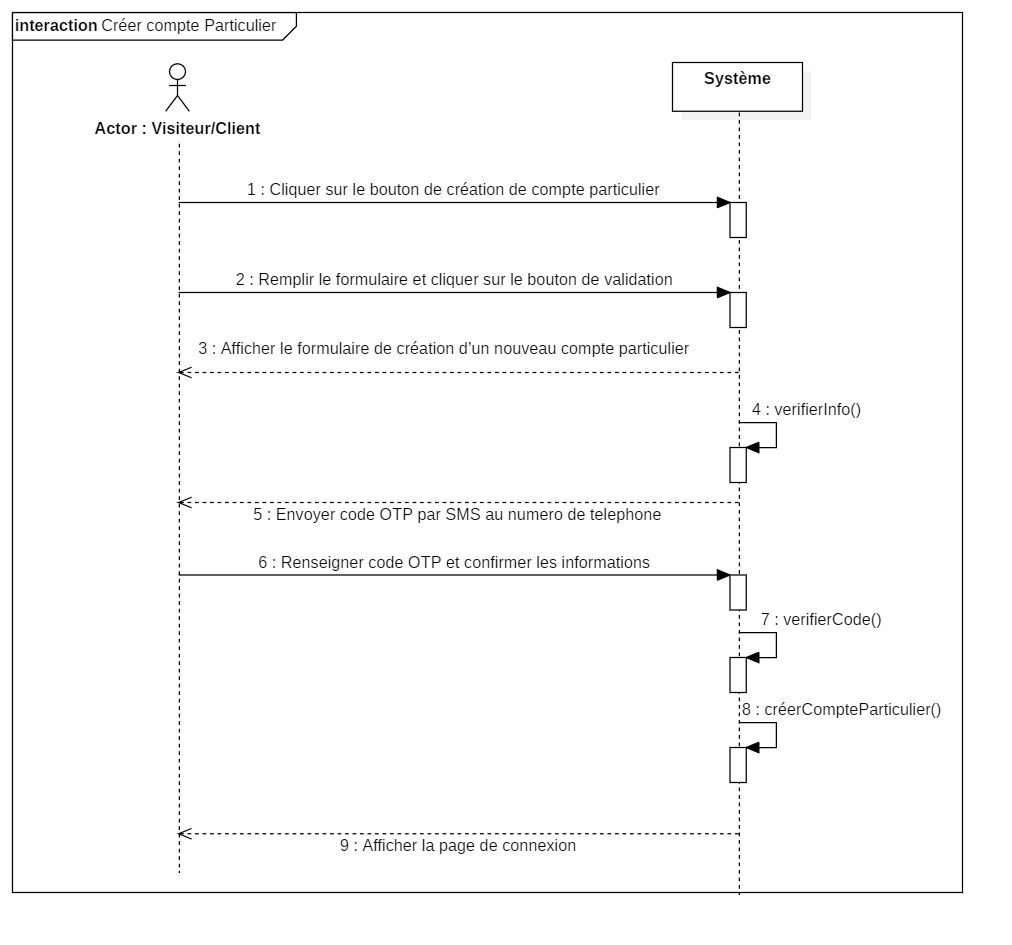


Figure 6 : Diagramme de séquence du cas "créer compte particulier"

##### Création de compte business

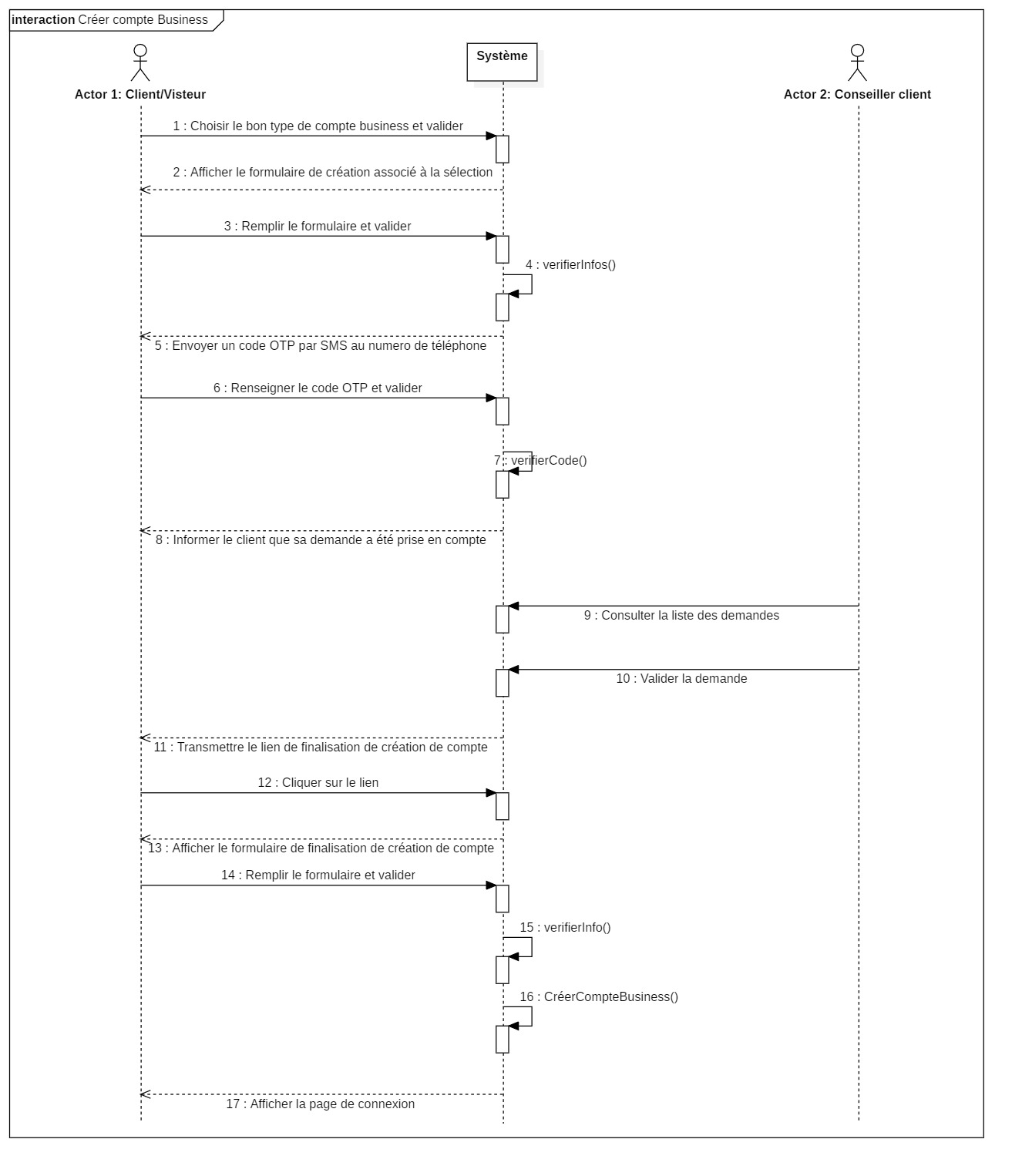


Figure 7: Diagramme de séquence du cas "créer compte business"

#### Cas « S’authentifier »

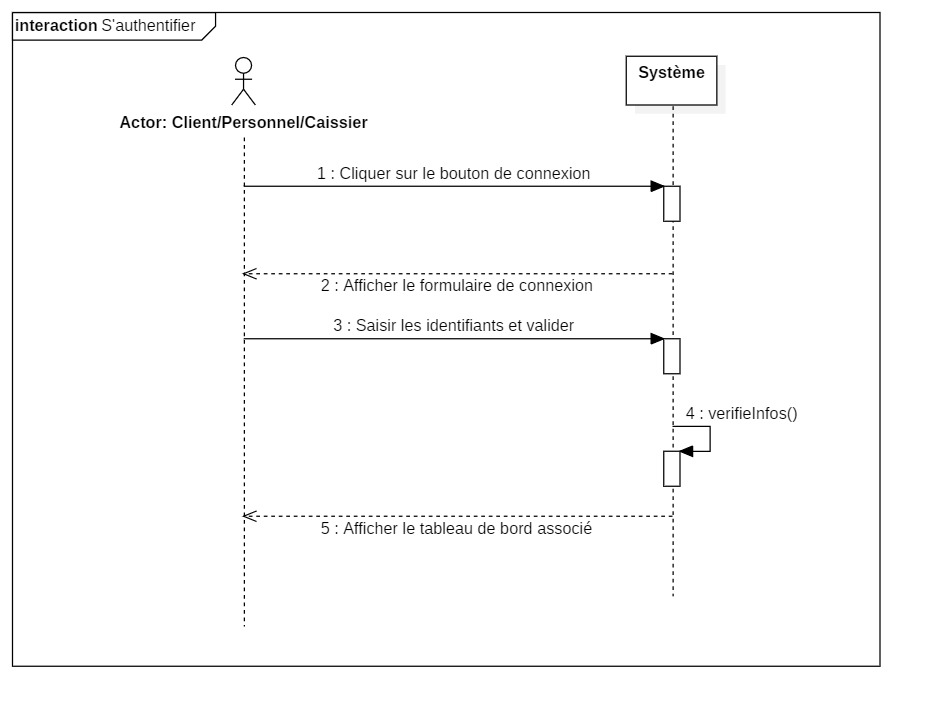


Figure 8: Diagramme de séquence du cas "s'authentifier"

#### Cas « Recharger son compte »

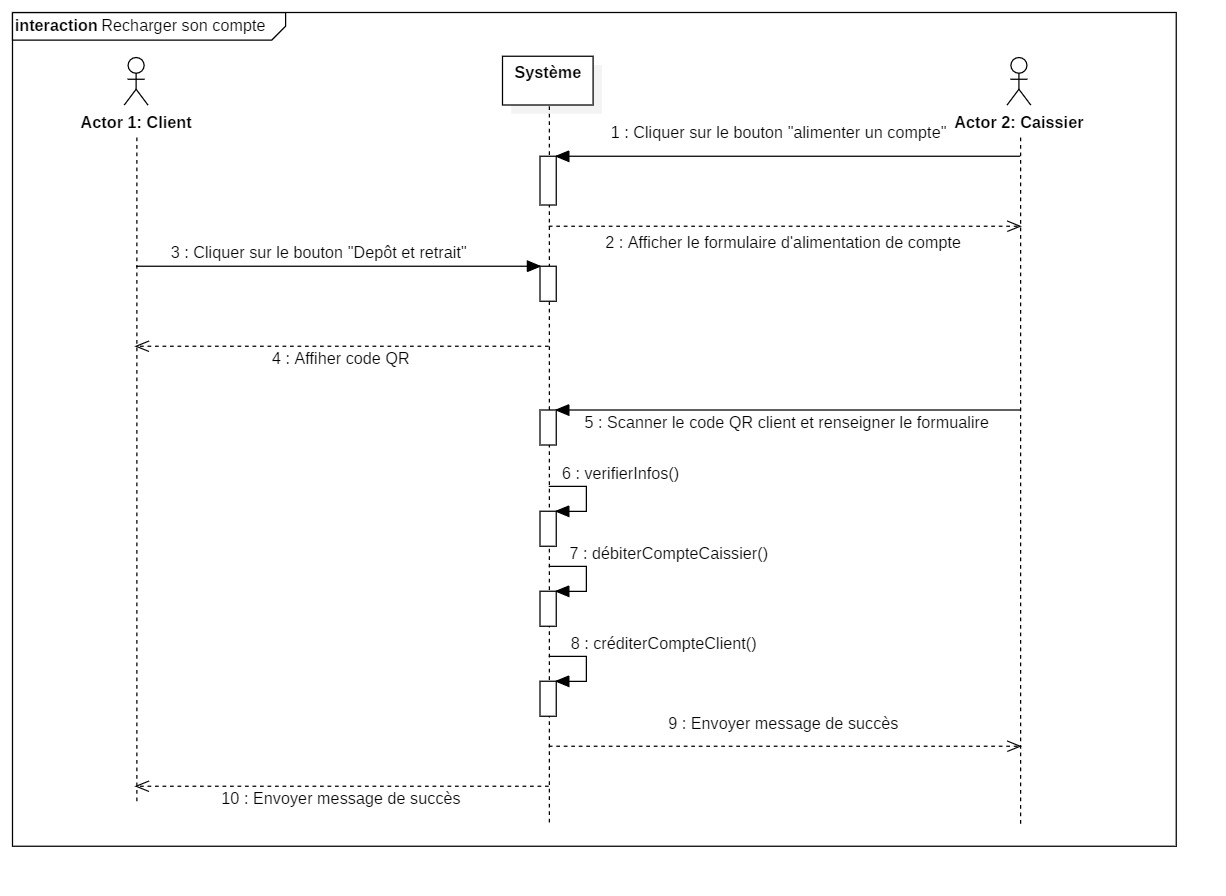


Figure 9: Diagramme de séquence du cas "Recharger son compte"

#### Cas « Effectuer retrait »

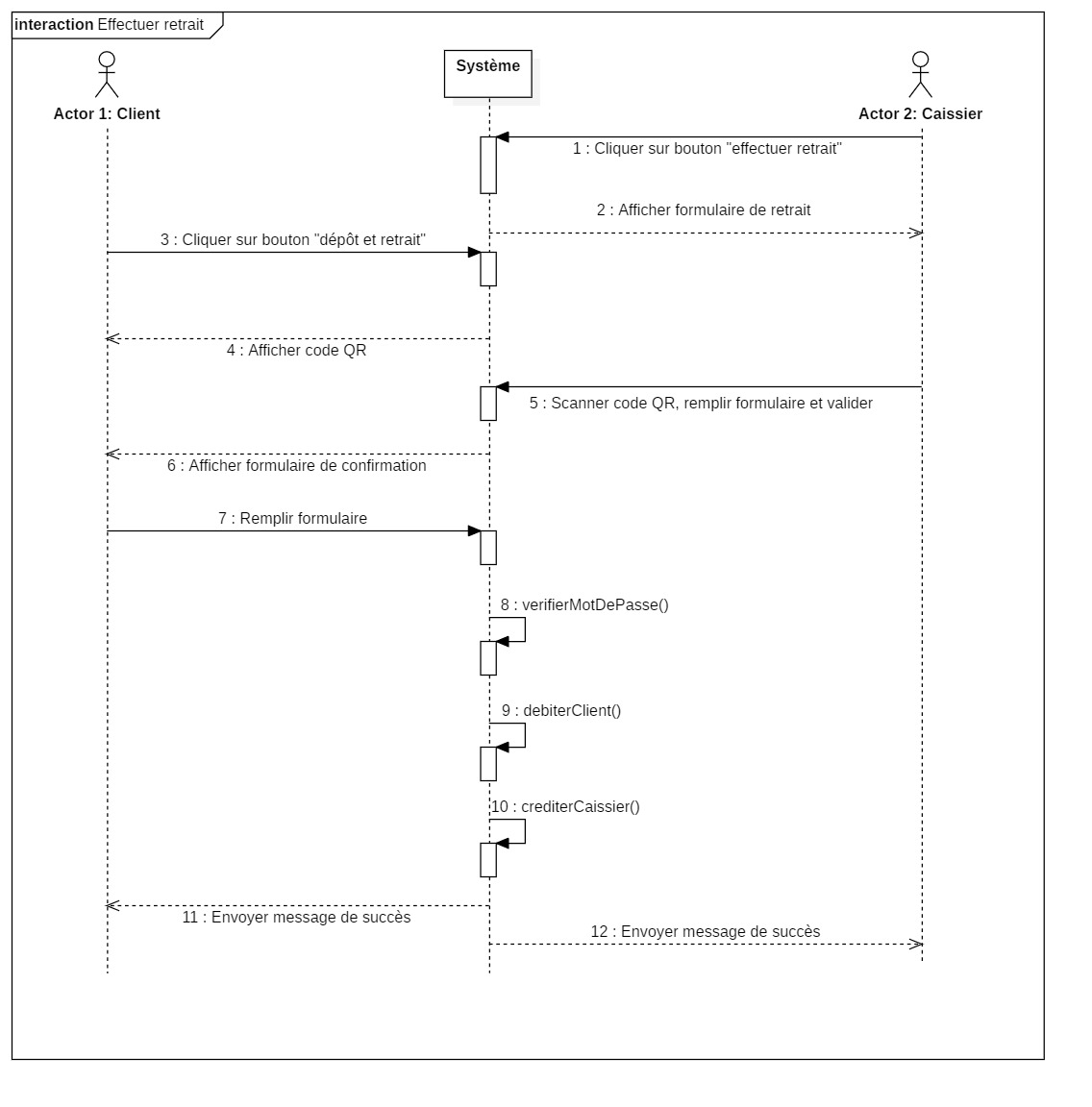


Figure 10: Diagramme de séquence du cas "Effectuer retrait"

#### Cas « Effectuer virement »

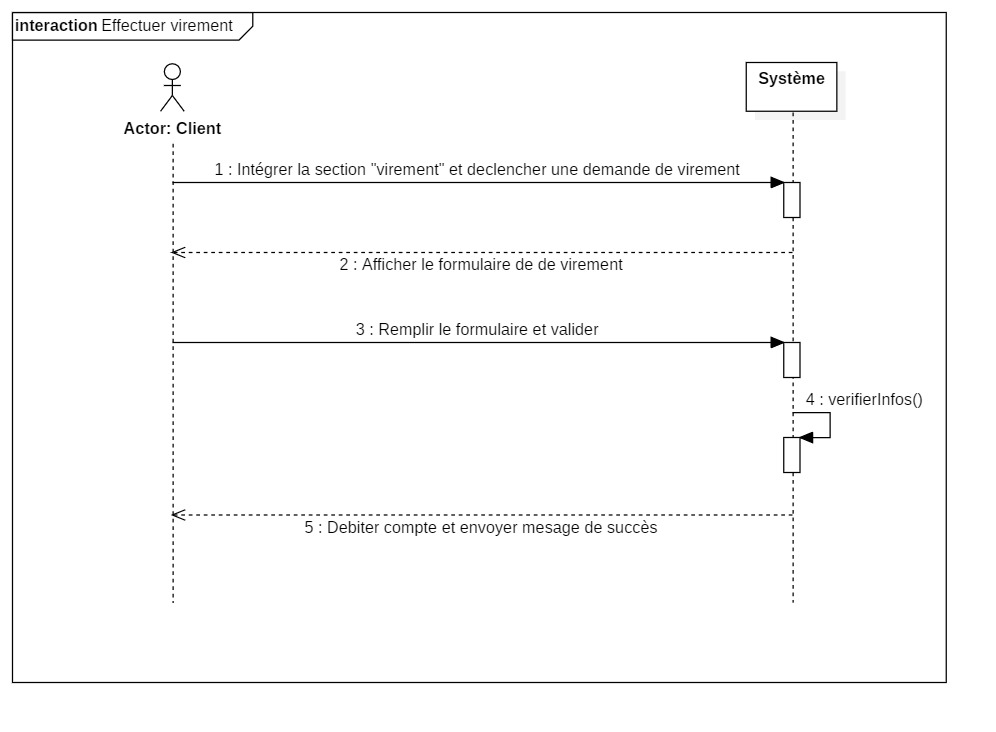


Figure 11: Diagramme de séquence du cas "effectuer virement"

#### Cas « Souscrire à une épargne »

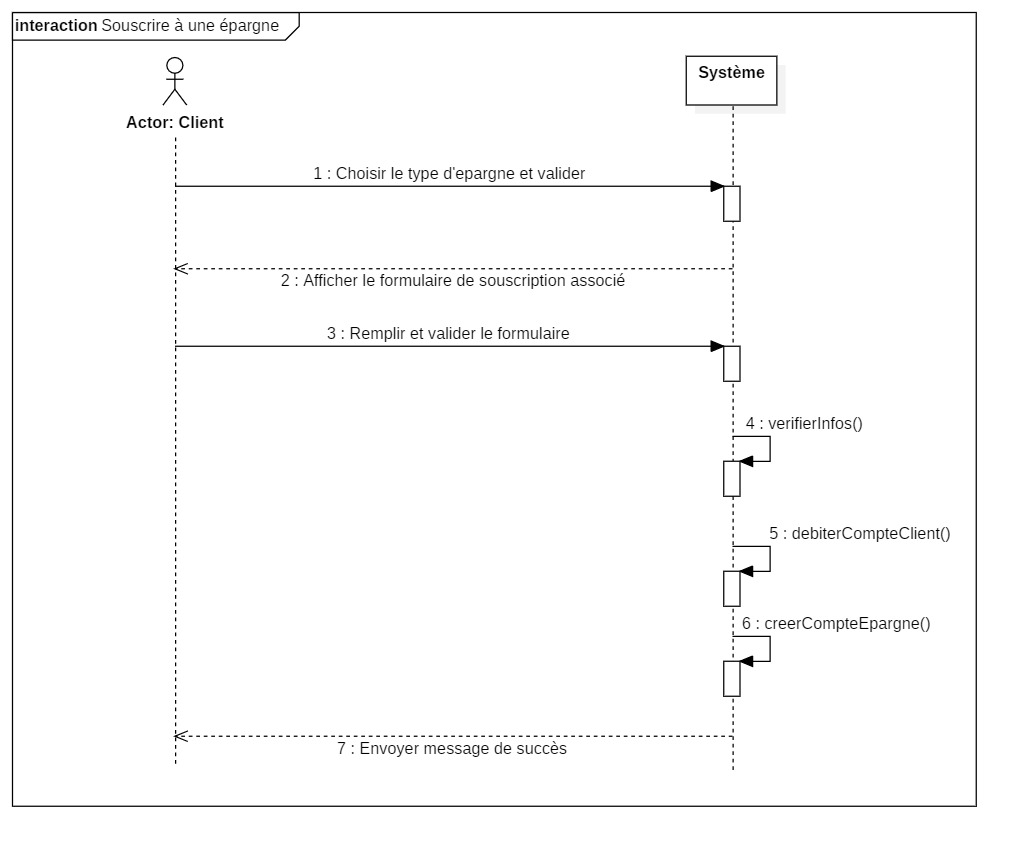


Figure 12: Diagramme de séquence du cas "Souscrire à une épargne"

#### Cas « Demander un crédit »

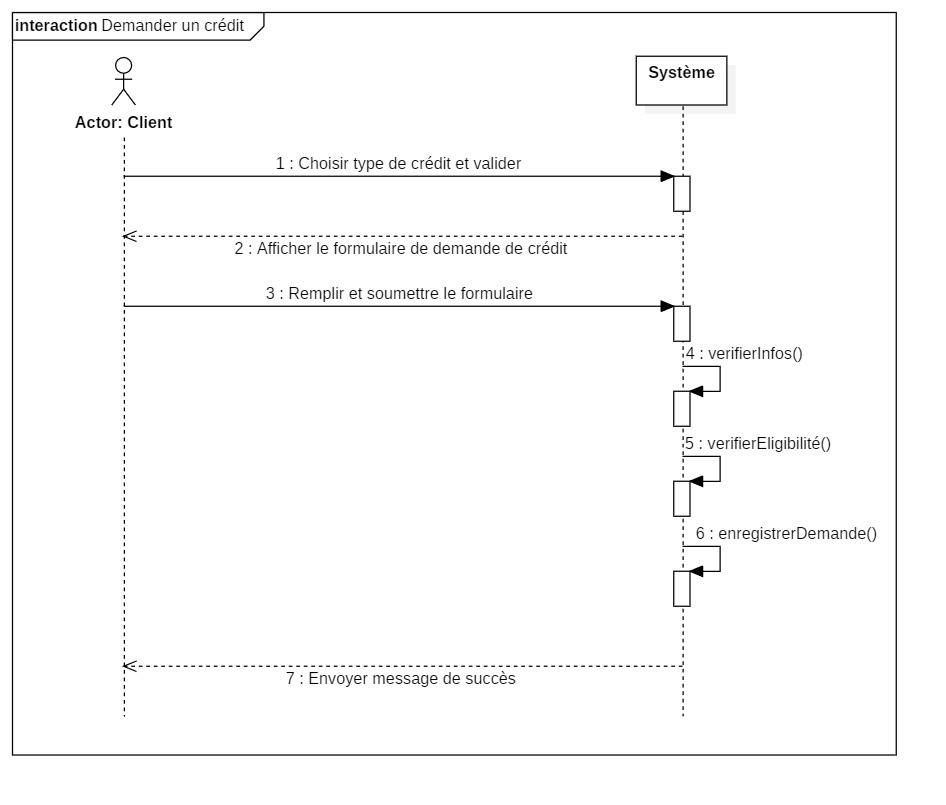


Figure 13: Diagramme de séquence du cas "Demander un crédit "

#### Cas « Valider demande de crédit »

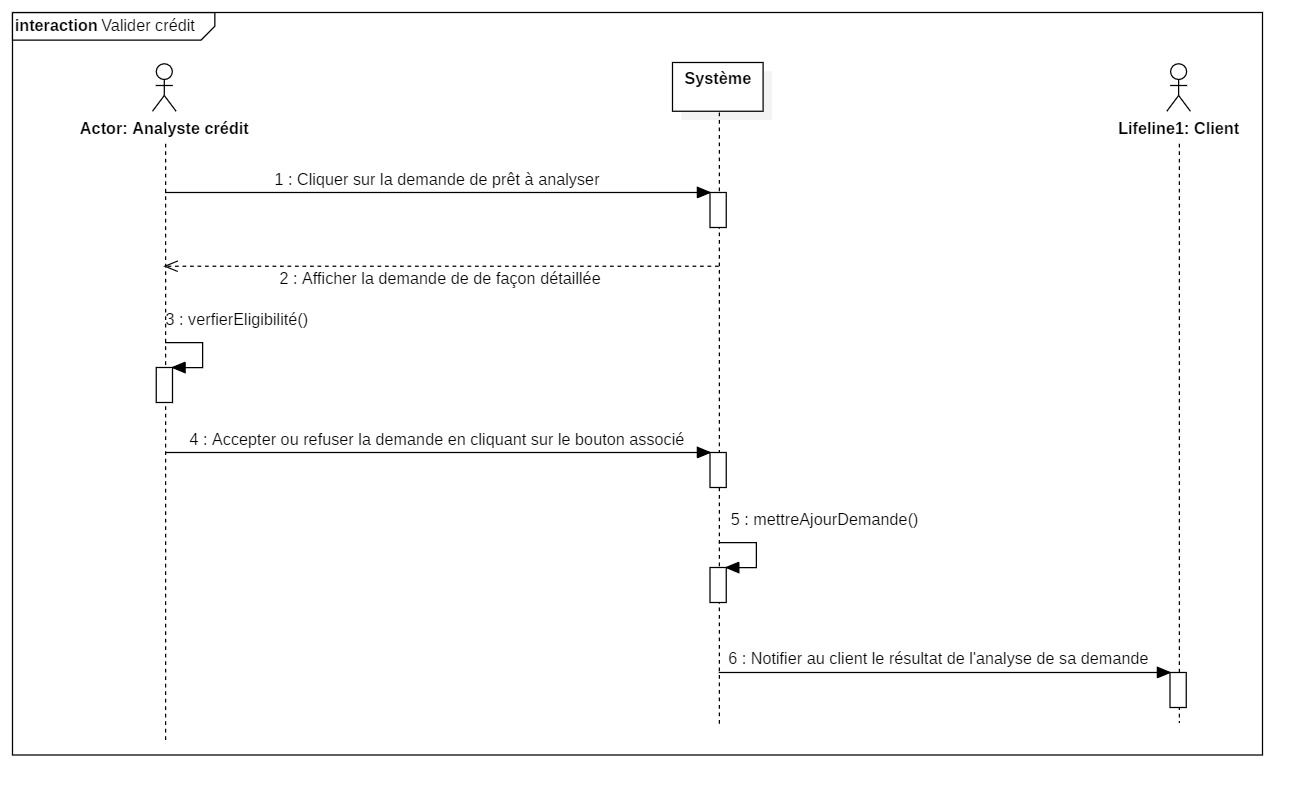


Figure 14: Diagramme de séquence du cas "Valider demande de crédit "

#### Cas « Commander carte bancaire »

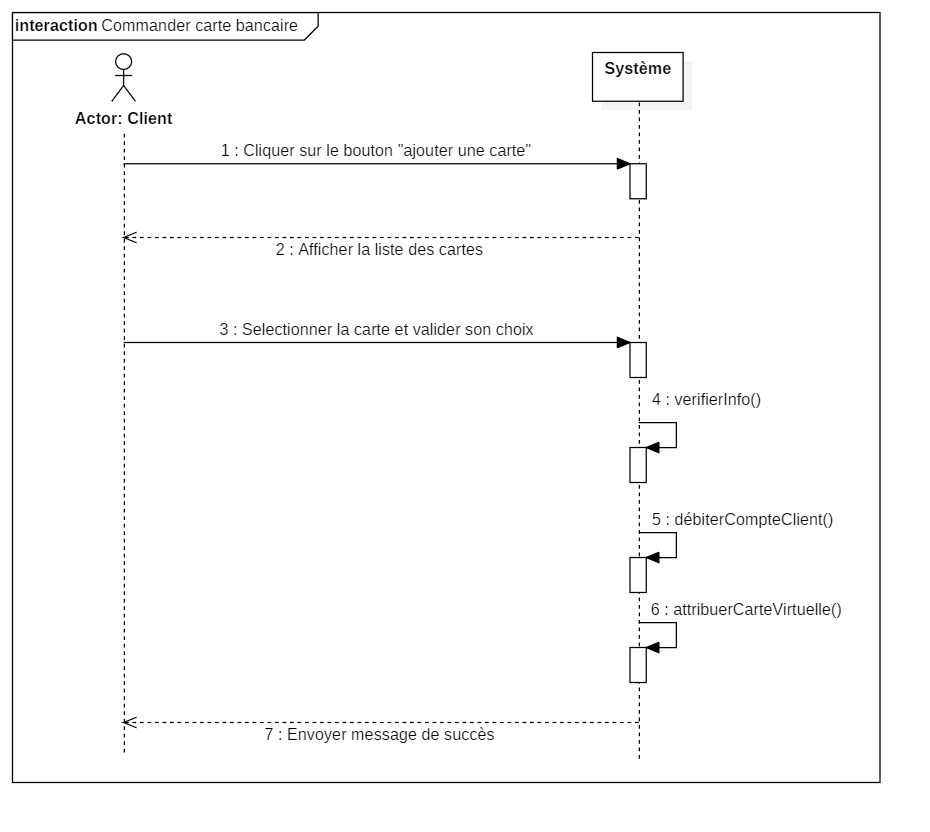


Figure 15: Diagramme de séquence du cas "Commander carte bancaire "

#### Cas « Vendre un produit »

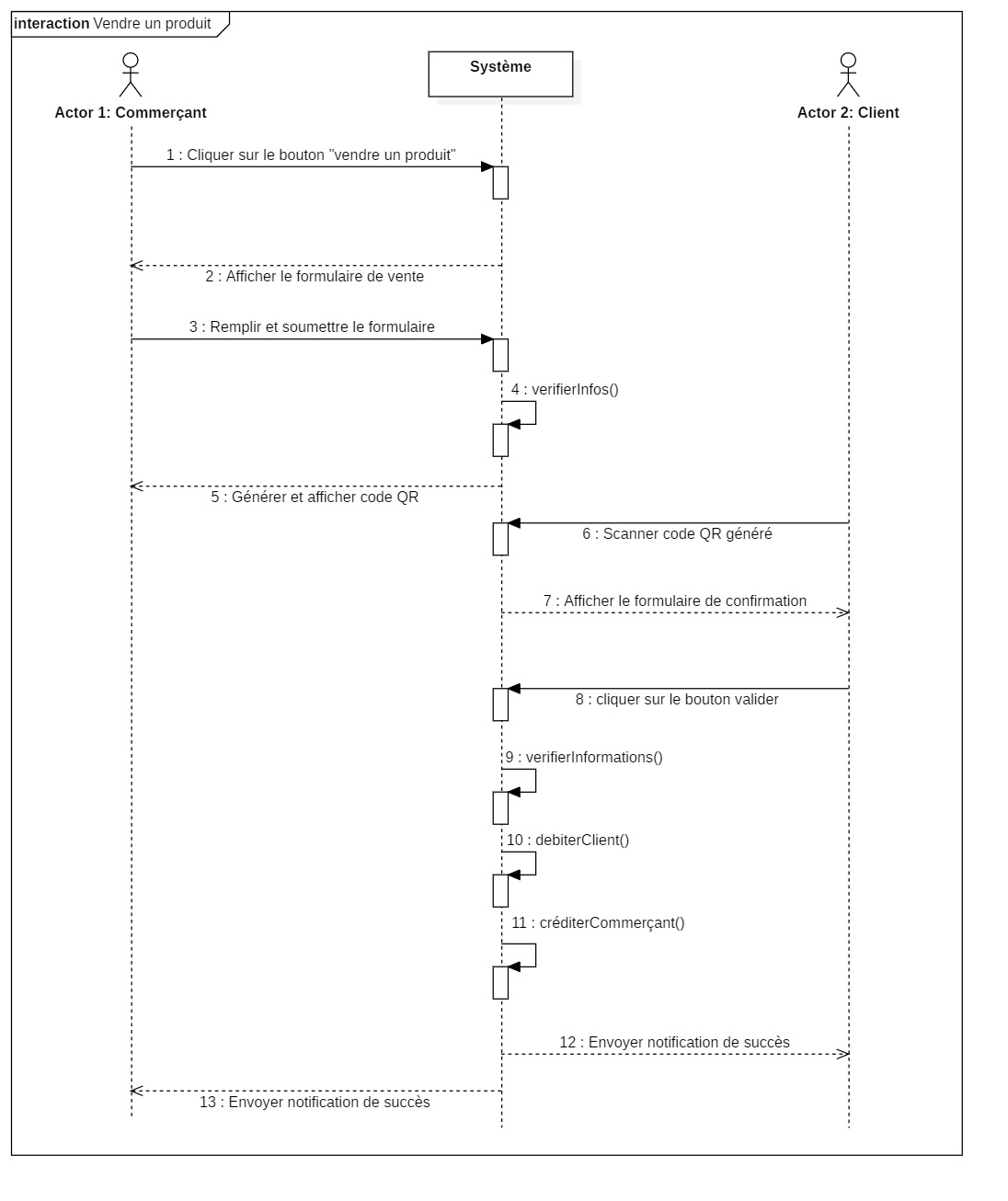


Figure 16: Diagramme de séquence du cas "Vendre un produit "

### Diagrammes d’activités

Un diagramme d'activité est un type d'organigramme UML (Unified Modeling Language) qui montre le flux d'une activité à une autre dans un système ou un processus. Il est utilisé pour décrire les différents aspects dynamiques d'un système et est appelé « diagramme de comportement » car il décrit ce qui doit se passer dans le système modélisé.

Les diagrammes d'activité permettent également de visualiser des systèmes très complexes. C'est pourquoi les diagrammes d'activité sont souvent utilisés dans la modélisation des processus métier ou pour décrire les étapes d'un diagramme de cas d'utilisation au sein des organisations. Ils montrent les différentes étapes d'une activité et l'ordre dans lequel elles sont présentées. Ils peuvent également montrer le flux de données entre les activités.

#### Cas « S’inscrire »

##### Création de compte de compte particulier

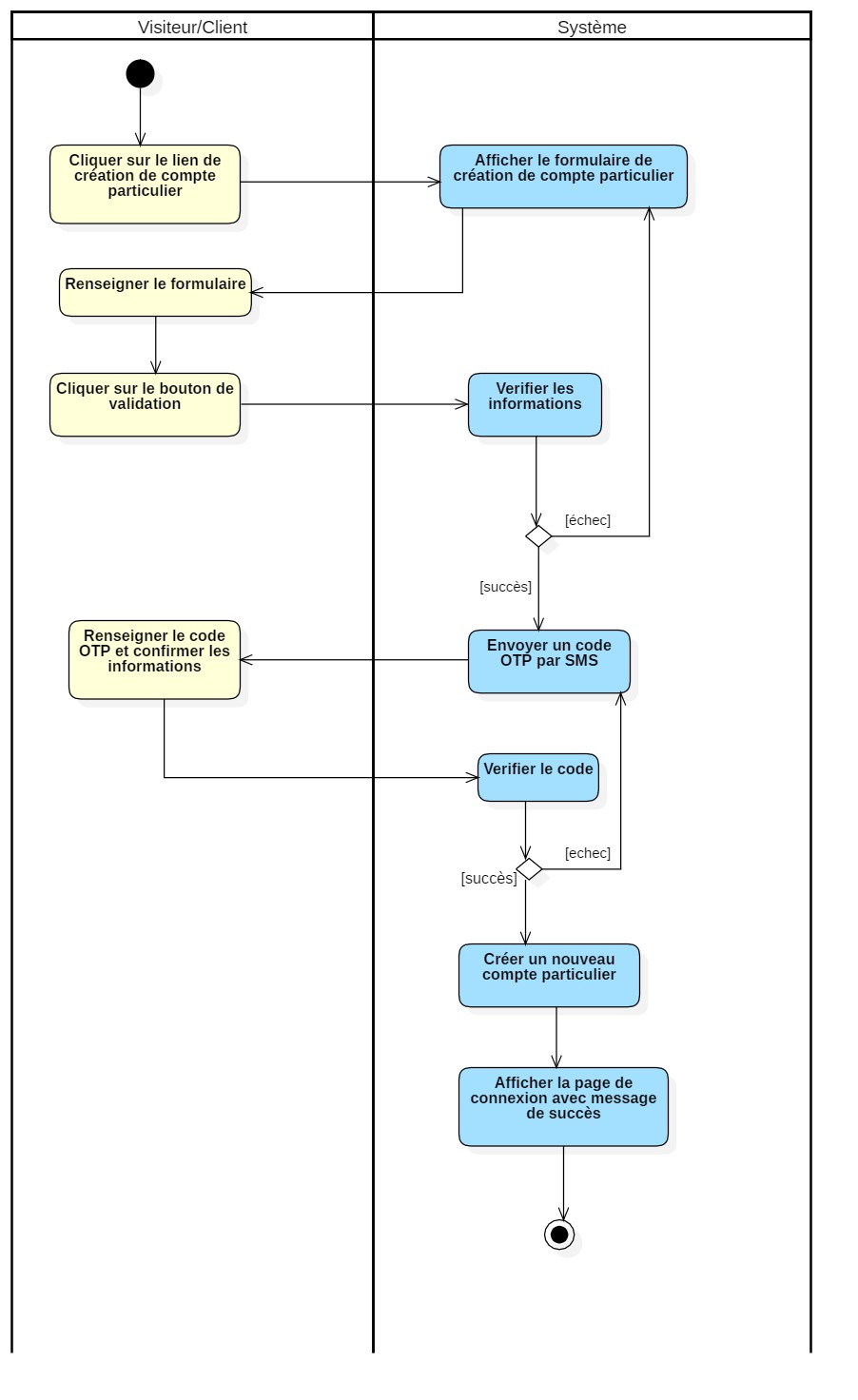


Figure 17: Diagramme d'activité "Créer compte de compte particulier"

##### Création de compte business

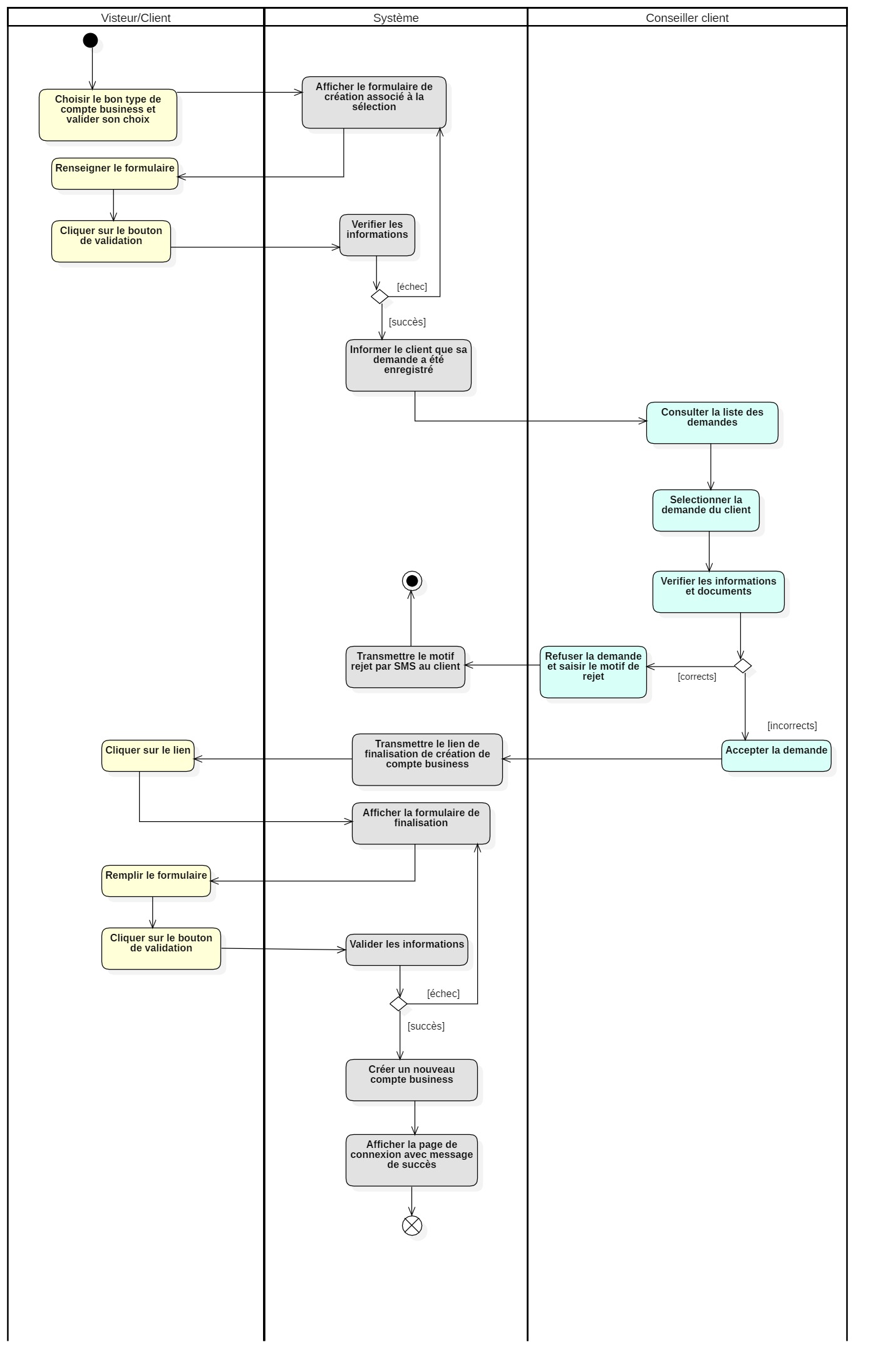


Figure 18: Diagramme d'activité "Création de compte business"

#### Cas « S’authentifier »

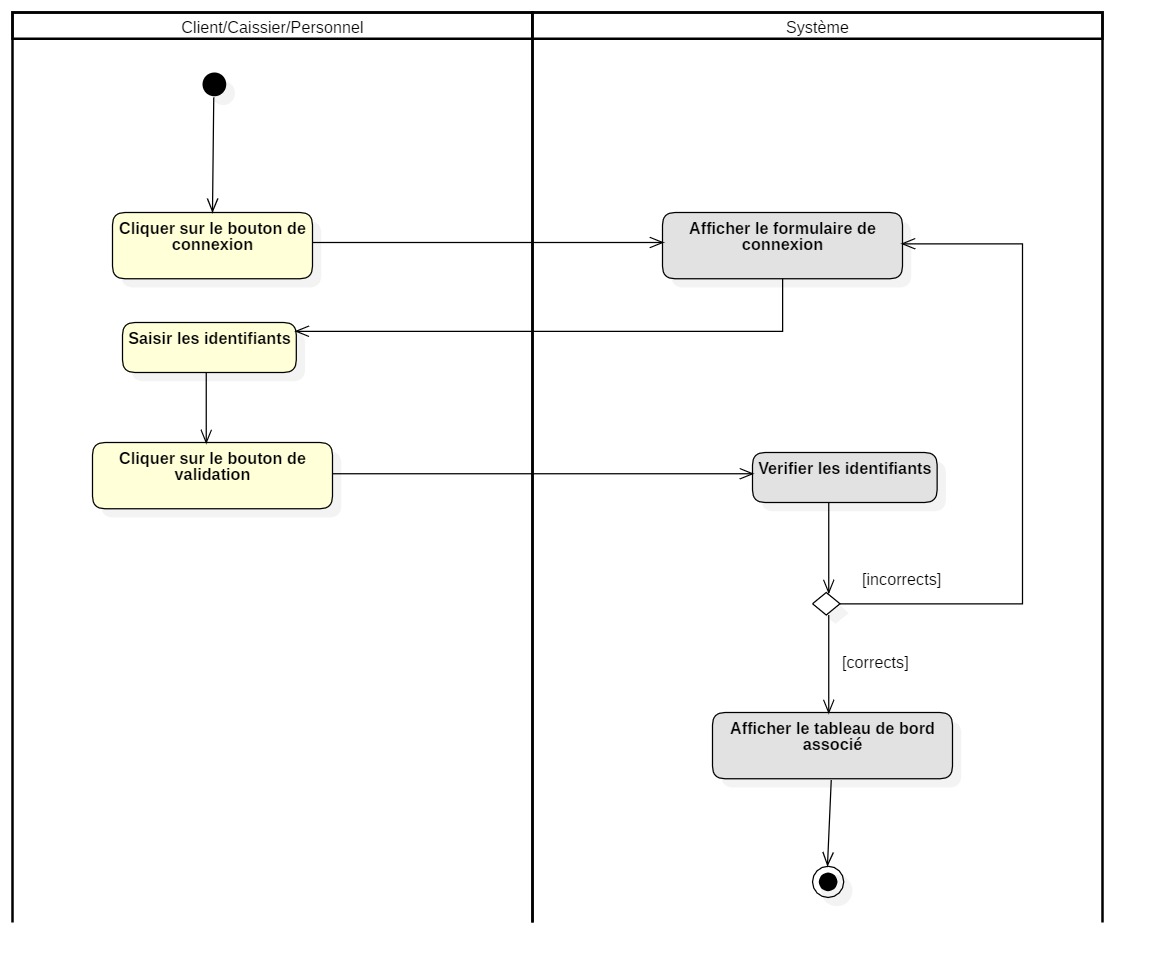


Figure 19: Diagramme d'activité du cas "S'authentifier"

#### Cas « Recharger son compte »



Figure 20: Diagramme d'activité du cas "Recharger son compte"

#### Cas « Effectuer un retrait »

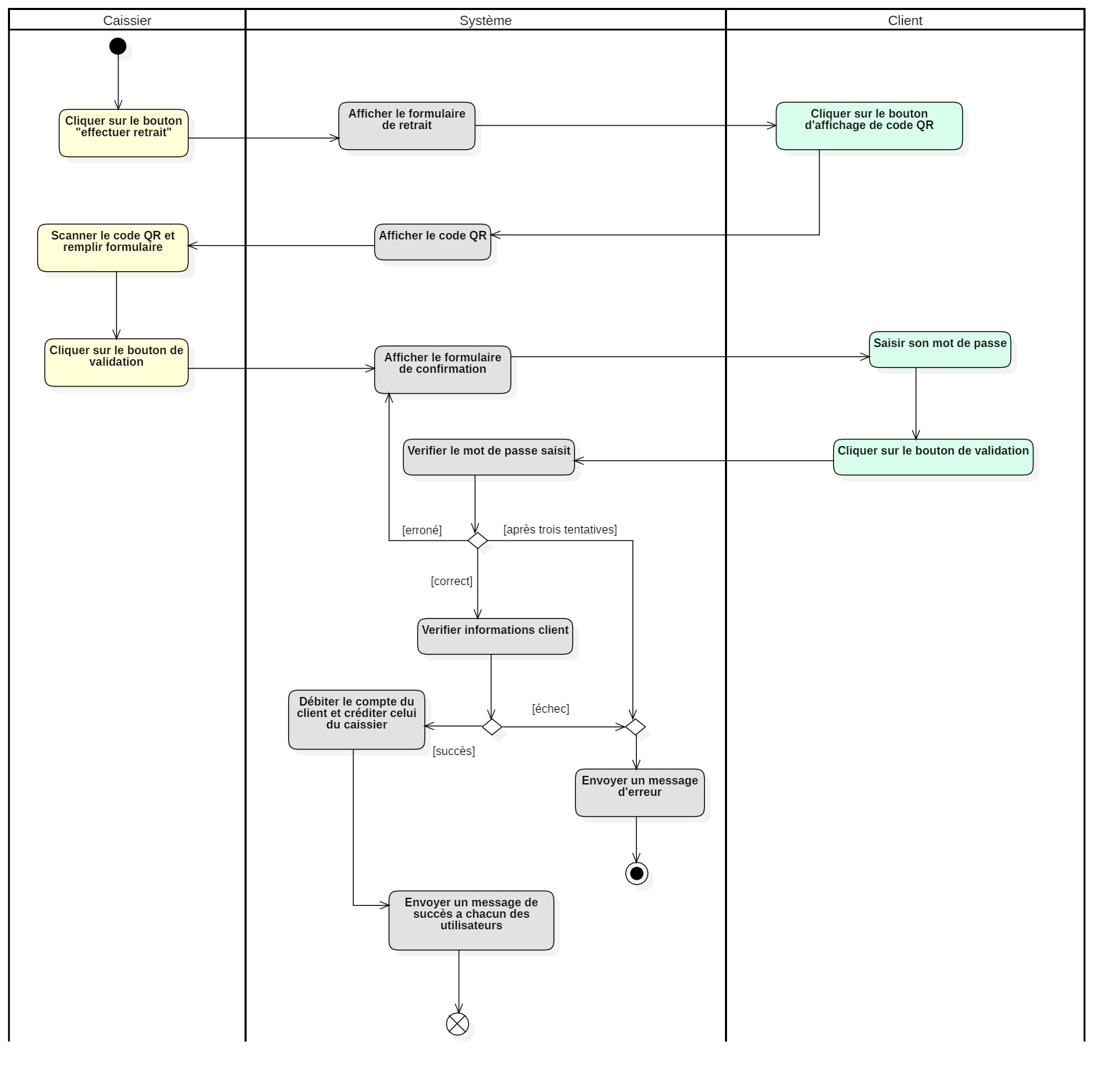


Figure 21: Diagramme d'activité du cas "effectuer retrait"

#### Cas « Effectuer virement »

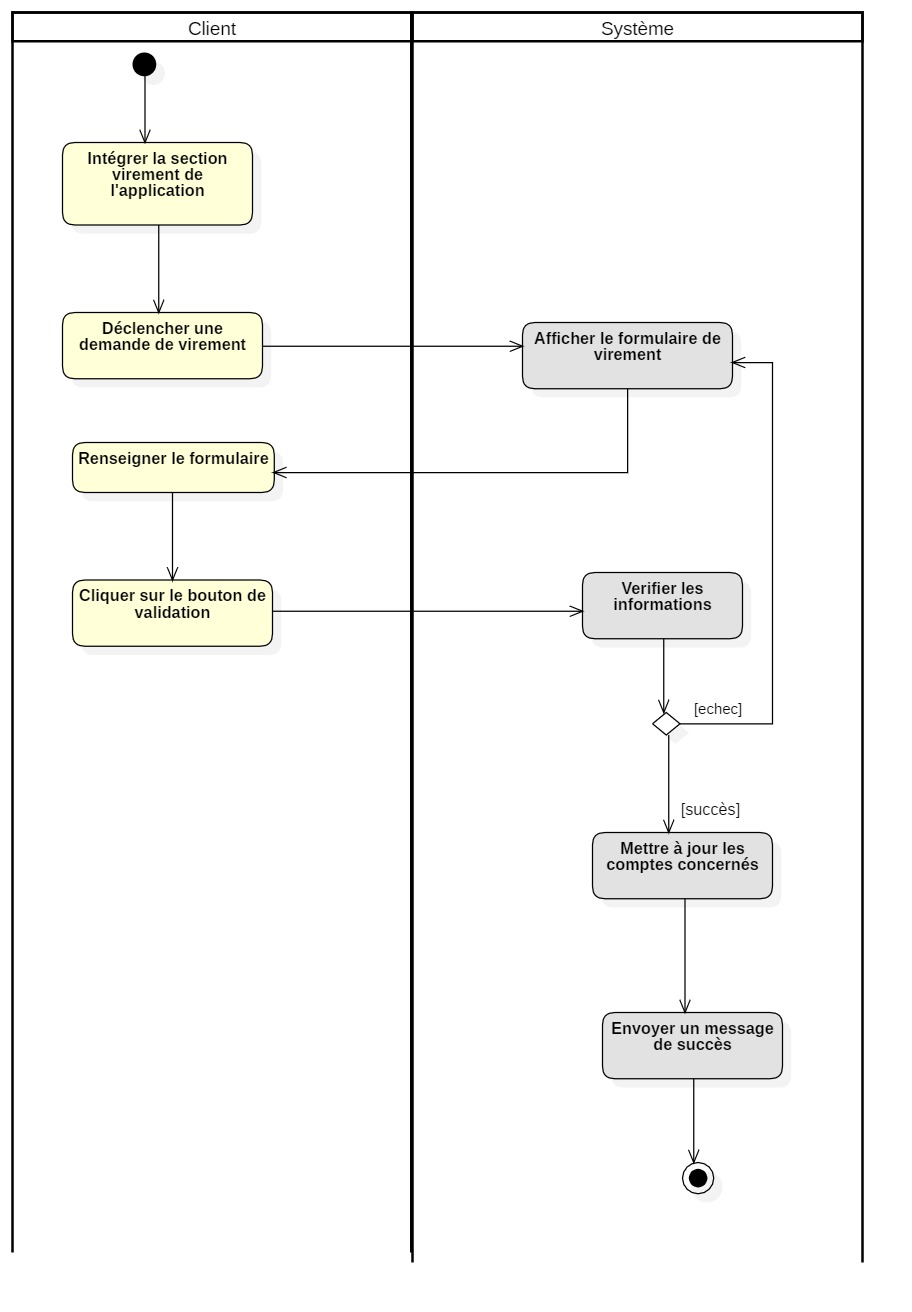


Figure 22: Diagramme d'activité du cas "Effectuer virement"

#### Cas « Souscrire à une épargne »

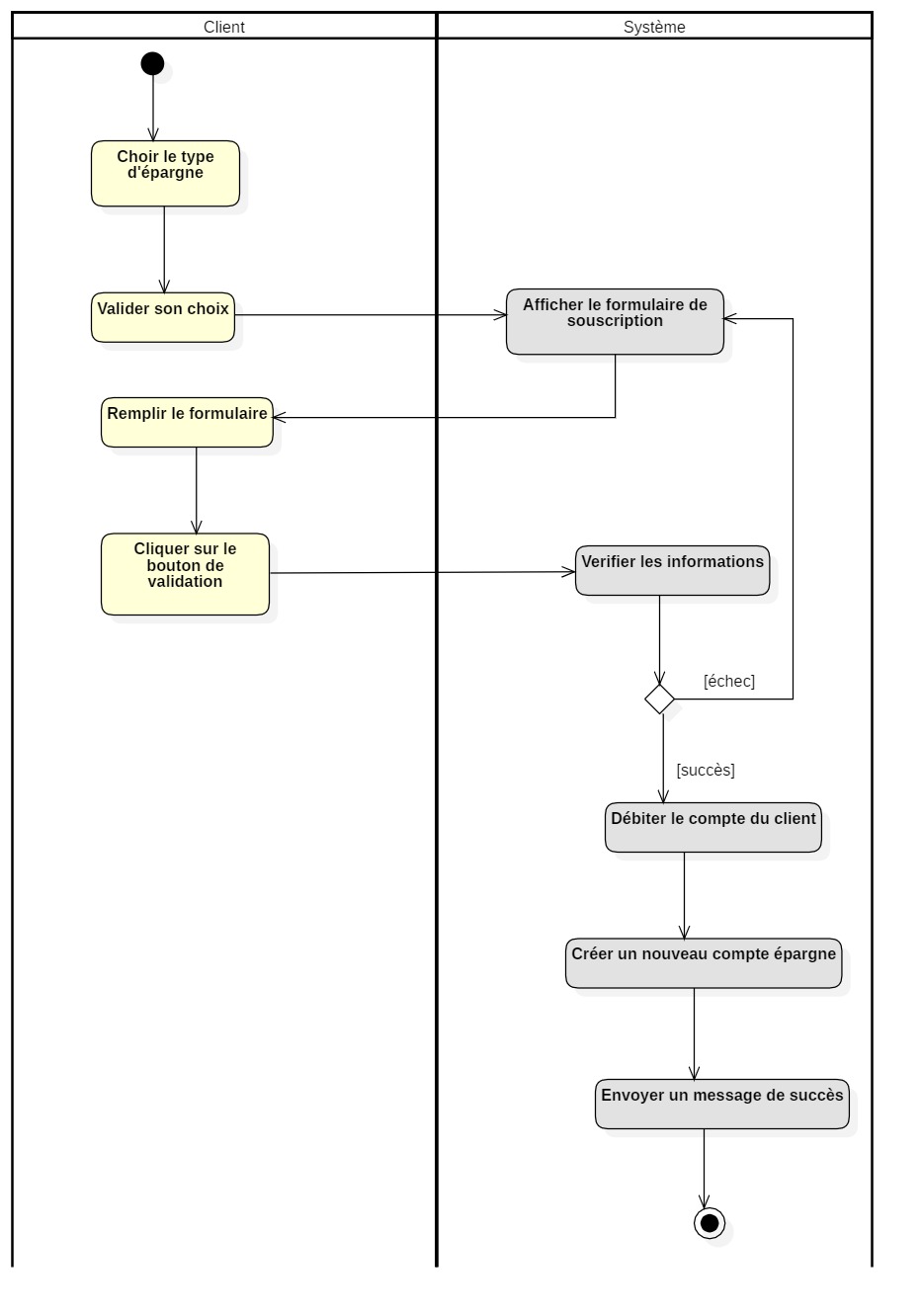


Figure 23: Diagramme d'activité du cas "Souscrire à une épargne"

#### Cas « Demander un crédit »

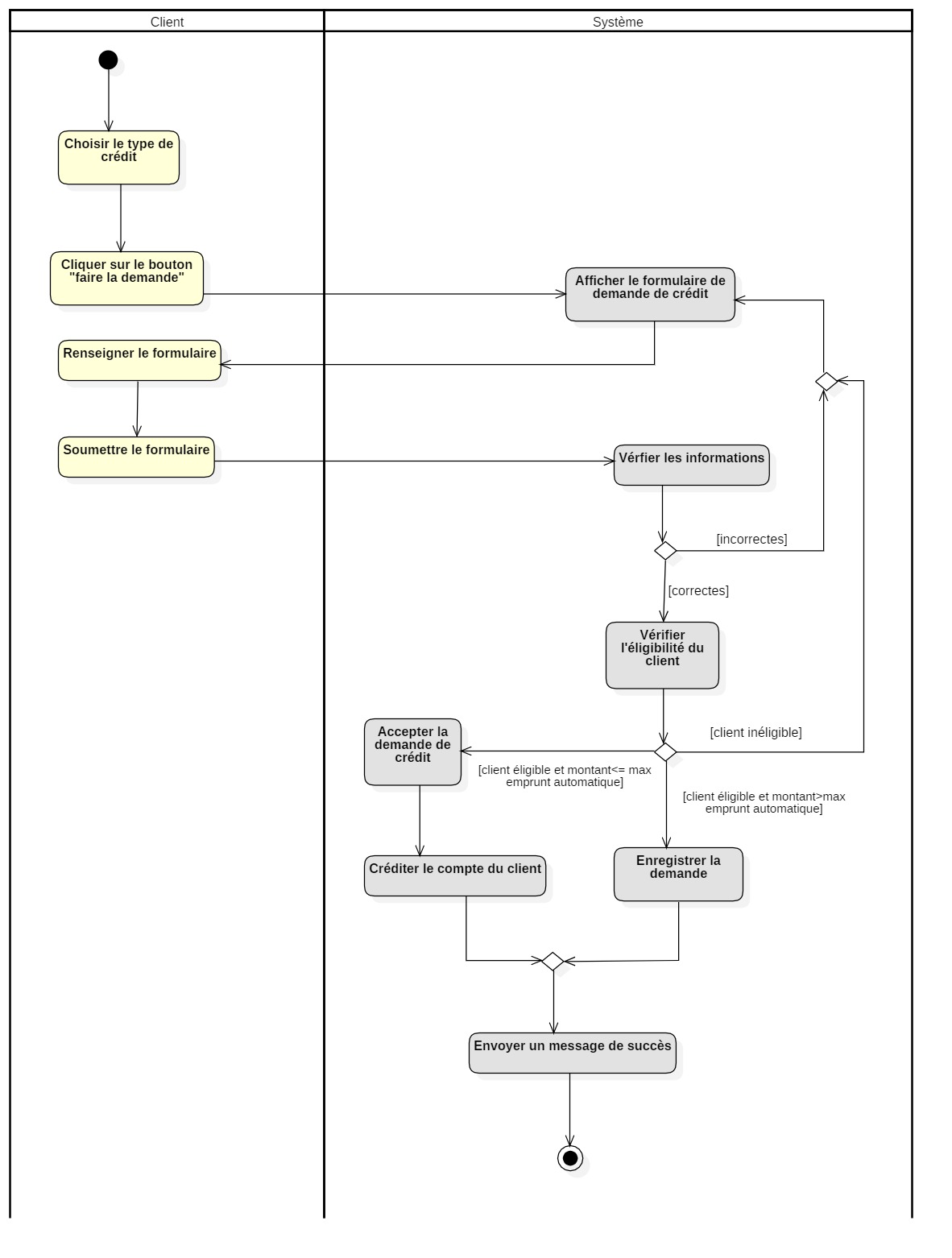


Figure 24:Diagramme d'activité du cas "Demander un crédit"

#### Cas « Valider crédit »

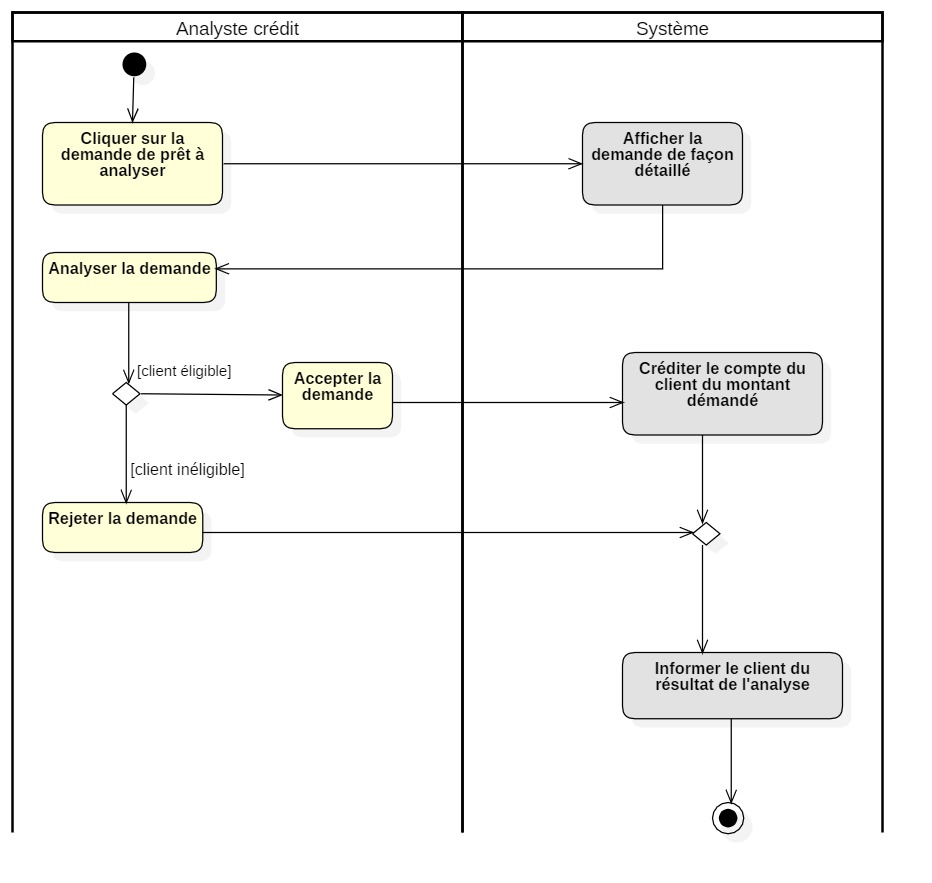


Figure 25: Diagramme d'activité du cas "Valider un crédit"

#### Cas « Commander carte bancaire »

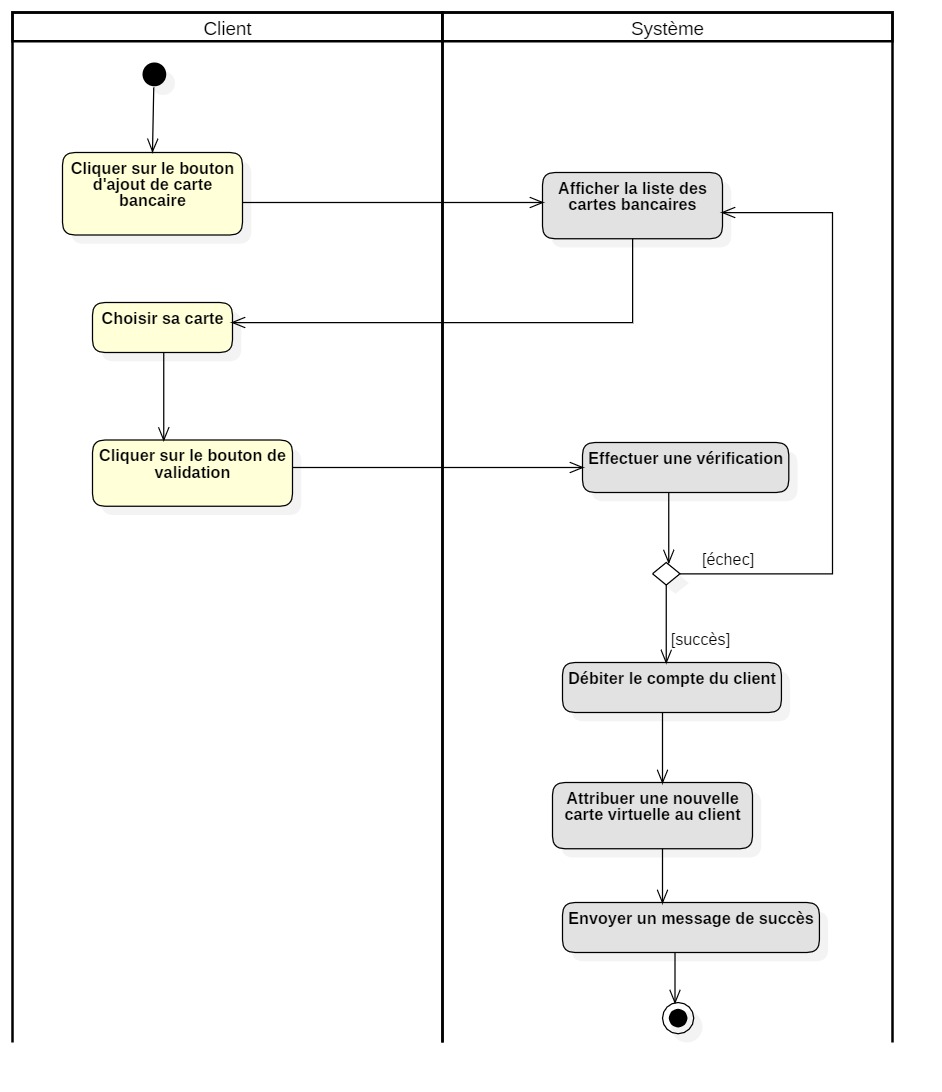


Figure 26: Diagramme d'activité du cas "Commander carte bancaire"

#### Cas « Vendre un produit »

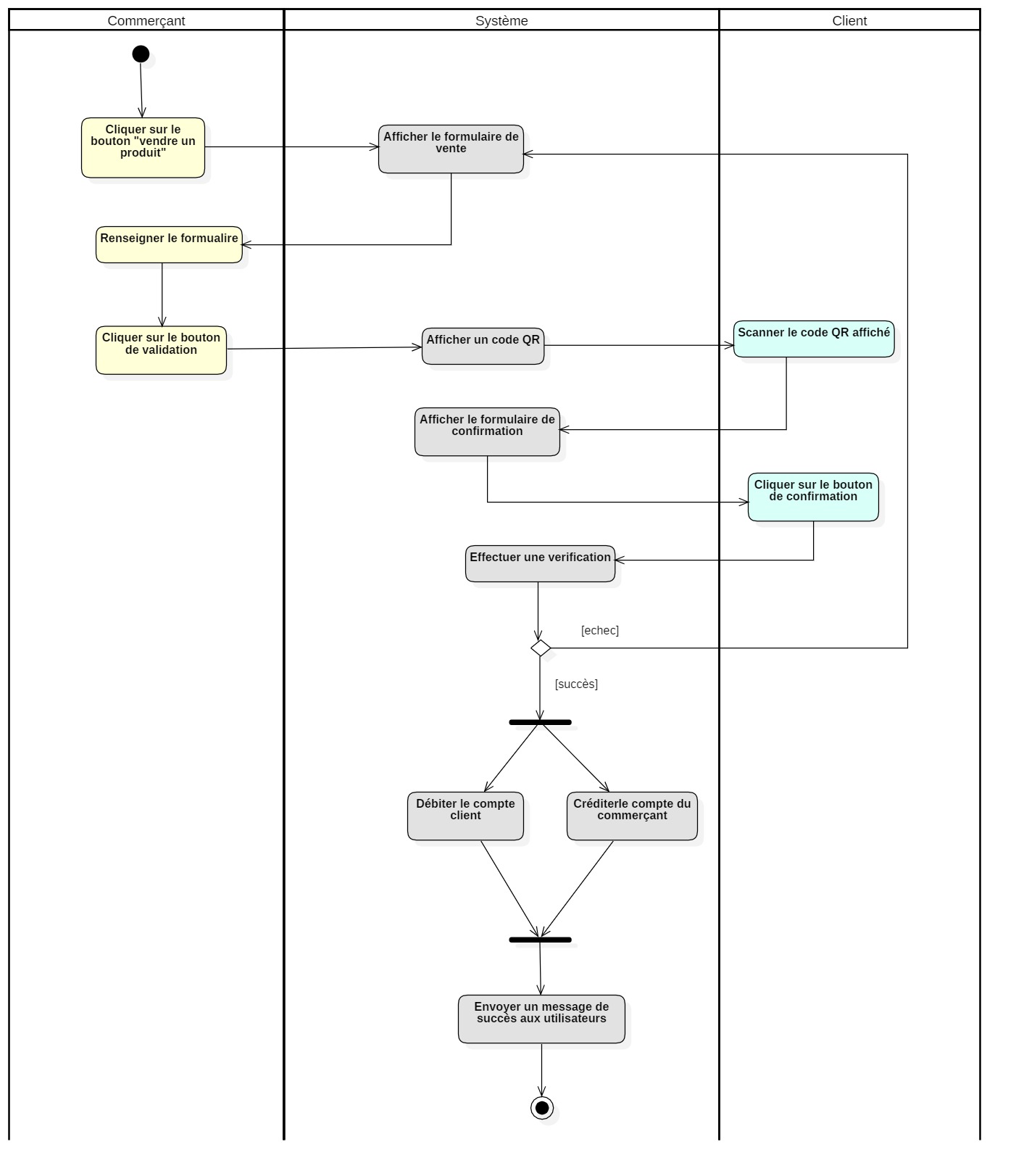


Figure 27: Diagramme d'activité du cas "Vendre un produit"

## ELABORATION DU DIAGRAMME DE CLASSE

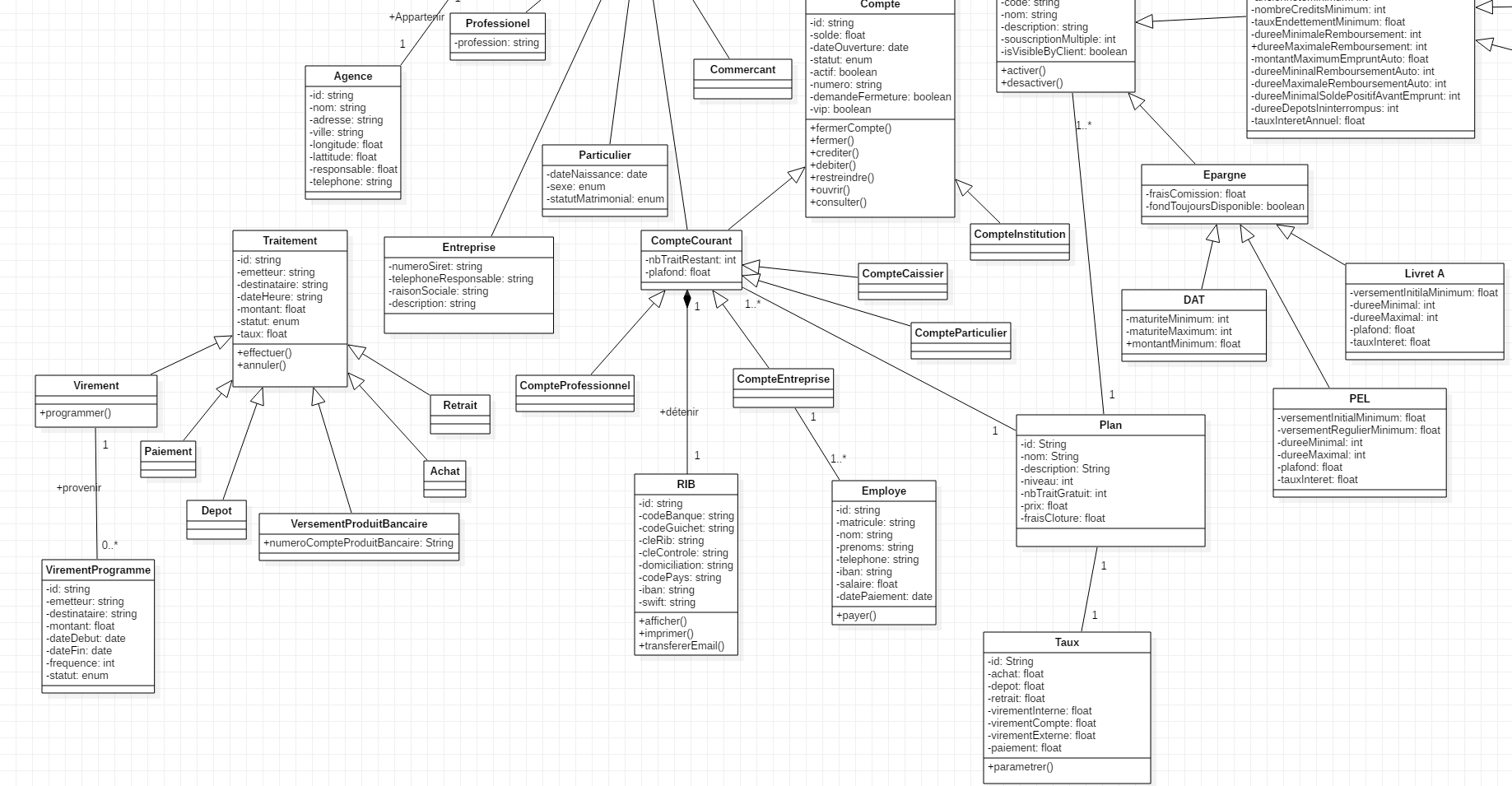


Figure 28: Diagramme de de classe

# PARTIE 3 : MISE EN ŒUVRE DU SYSTEME

Cette partie abordera en premier lieu l’implémentation de notre système puis dans un second volet présentera les détails non négligeables inhérents à la réalisation d’une application.

# CHAPITRE VI : CONSTRUCTION

*La construction est la troisième phase du processus EssUP. Elle est principalement axée sur l’implémentation et a pour objectif de produire un logiciel répondant aux besoins prioritaires des utilisateurs. A la fin de cette phase, nous devons fournir une version exécutable du système.*

## ARCHITECTURE LOGICIELLE

L’architecture logicielle se définit par la façon dont un système informatique est organisé en composantes et comment celles-ci interagissent. Cette idée s'inscrit également dans le cadre du cycle de développement logiciel. Il est essentiel d'avoir une vision globale des aspects fonctionnels d'une application tout en tenant compte de ses caractéristiques non fonctionnelles. Un mauvais choix d'architecture peut entraîner de grave conséquence, compromettant ainsi le succès technologique et commercial d'un projet.

Une application informatique peut être structurée en trois niveaux distincts d'abstraction:

* **Couche Présentation**: Autrement dite Couche IHM (Interface Homme Machine), c’est la partie rendue du logiciel. Elle permet l’interaction d’application avec l’utilisateur.
* **Couche Logique applicative**: Autrement dite Couche métier ou Couche traitement, elle décrit les traitements à exécuter afin de répondre aux besoins de l’utilisateur.
* **Couche Données**: Autrement dite Couche de persistance, elle englobe tous les mécanismes permettant de gérer les informations stockées par l'application

Le découpage et la répartition des trois niveaux d’abstraction d’une application informatique permettent de distinguer plusieurs types d’architecture à savoir :

### Architecture 1-tiers

Une architecture 1-tiers désigne une application dans laquelle la logique du back-end, la base de données et l'interface utilisateur sont toutes situées sur le même serveur. Elle implique l'hébergement de tous les composants essentiels d'une application logicielle ou d'une technologie sur un seul serveur ou une seule plateforme, tels que l'interface, le middleware et les données du back-end. Certains la désignent également comme tout programme pouvant être installé et exécuté sur un seul ordinateur.



Figure 29: Architecture 1-tiers

### Architecture 2-tiers

Une architecture à deux niveaux fait référence à une architecture logicielle dans laquelle une couche de présentation s'exécute sur un client et une couche de données est disponible sur un serveur. La séparation de ces deux composants dans deux emplacements physiques différents donne lieu à une architecture à deux niveaux, par opposition à une architecture à un seul niveau. L'application côté client établit une connexion avec le serveur afin de communiquer avec la couche de données. Le côté serveur est chargé de fournir des fonctionnalités de traitement des requêtes et de gestion des transactions.

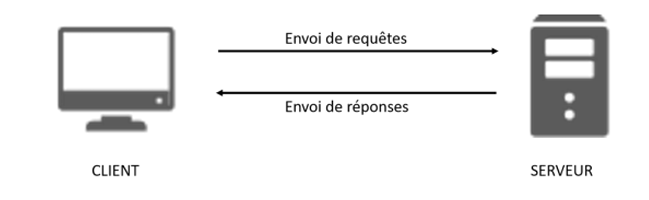


Figure 30: Architecture 2-tiers

### Architecture 3-tiers

Dans une architecture à 3 niveaux, la couche présentation, la couche métier et la couche base de données sont situées sur des machines distinctes. Contrairement à l'architecture à 2 niveaux, 2 serveurs effectuent les tâches principales de cette application. Un serveur est destiné à la couche métier et l'autre à la couche base de données.

Cette architecture est utilisée pour concevoir une application Web. La couche client demande les données. Le serveur d'applications agit comme une interface entre le client et la couche serveur. La couche application aide le client à communiquer avec le serveur. Elle reçoit la demande du client et l'envoie à la base de données, qui répond ensuite et les informations requises parviennent au client en passant par la couche application.

La couche de base de données stocke toutes les données et répond en fonction de la logique de la couche application. Cette couche augmente également la sécurité de l'application. L'objectif principal de l'architecture à 3 niveaux est de séparer l'application cliente et la base de données physique, d'assurer l'indépendance des données du programme et de prendre en charge plusieurs vues.



Figure 31: Architecture 3-tiers

Après avoir exposé les différents types d’architectures logicielles, il convient ensuite de procéder à une comparaison de chacun d'eux avant de poursuivre notre analyse.

Tableau 16: Tableau comparatif des architectures logicielles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Architectures | Sécurité | Optimisation des ressources | Richesse de l’interface utilisateur | Simplicité d’administration |
| 1-tiers | Moyen | Bon | Moyen | Bon |
| 2-tiers | Bon | Moyen | Bon | Moyen |
| 3-tiers | Bon | Bon | Bon | Moyen |

Au vu du tableau ci-dessus, et compte tenu du cadre professionnel et institutionnel du projet, notre choix s’est porté sur une architecture 3-tiers puisqu’elle offre une grande marge d’évolution et est assez flexible. Nous pouvons compter aussi au nombre des motivations de ce choix, la possibilité d’appliquer une sécurité à tous les niveaux de manière indépendantes, la réduction des charges du poste client ce qui le rend plus simple à manipuler pour les non informaticiens, et en outre la grande fiabilité et disponibilité du système.

## SYSTEME DE GESTION DE BASE DE DONNEES

Un système de gestion de base de données (SGBD) est un système logiciel permettant de créer et de gérer des bases de données. Un SGBD permet aux utilisateurs finaux de créer, protéger, lire, mettre à jour et supprimer des données dans une base de données. Il gère également la sécurité, l'intégrité des données et la concurrence des bases de données.

Il existe différents types de systèmes de gestion de base de données :

* **Le modèle relationnel (ou SQL)** : les données relationnelles sont structurées sous forme de tables, ce qui présente l'avantage de faciliter leur gestion tout en assurant la cohérence des informations stockées ;
* **Le modèle non relationnel (ou noSQL)** : les données peuvent être structurées différemment d'un tableau, dans un format optimisé selon les besoins spécifiques du type de données stockées (comme des valeurs simples, des graphes, des documents JSON, etc.) ;
* **Le modèle hiérarchique** : les données sont organisées selon une structure hiérarchique descendante. Toutefois, si l'un des nœuds est supprimé, toutes les données qui en dépendent le sont également, ce qui compromet la pérennité des informations ;
* **Les données orientées réseau** : ce modèle facilite l'établissement de connexions entre les données, qu'elles soient ascendantes, descendantes ou multiples.
* **Les données orientées objet** : ce système de gestion permet de gérer des structures de données complexes.
* **Les données orientées document** : Il organise les données sous forme de tableaux avec des documents pour consigner les informations.

Voici un aperçu de quelques SGBD disponibles avant de prendre une décision éclairée :

* **Microsoft SQL Server** est un système de gestion de base de données relationnelle créé par Microsoft. Depuis son lancement en 1989, SQL Server a connu une évolution significative, se transformant en une plateforme complète pour la gestion des informations d'entreprise, adaptée à une variété d'applications.
* **Oracle Database** est l'une des solutions de bases de données commerciales les plus reconnues qui organise les données en tables et colonnes interconnectées. Également appelée système de gestion de base de données relationnelle Oracle (SGBDRO), cette solution se caractérise par sa flexibilité et sa scalabilité. Elle permet de stocker les données sous forme d'objets tout en maintenant les relations entre celles-ci.
* **MySQL** est la SGBD open source la plus plébiscitée au monde.
* C'est un système de gestion de bases de données relationnelles SQL optimisé pour des performances élevées en matière de lecture. D'après DB-Engine, MySQL est la deuxième base de données la plus populaire, juste derrière Oracle Database. Elle alimente de nombreuses applications bien connues, telles que Facebook, Twitter, Netflix, Uber, Airbnb, Shopify et Booking.com.
* **MongoDB** est un système de gestion de bases de données orienté documents, pouvant être déployé sur un nombre illimité d'ordinateurs et ne requérant pas de schéma de données prédéfini. Écrit en C++, il est distribué sous une licence SSPL pour le serveur et les outils, sous licence Apache pour les pilotes, et sous licence Creative Commons pour la documentation. Il fait partie intégrante du mouvement NoSQL.

**Choix du Système de Gestion de Bases de Données**

Le tableau ci-dessous résume une étude comparative des différents SGBD examinés dans le but de sélectionner celui qui convient le mieux à notre projet.

Tableau 17: Tableau comparatif des systèmes de gestion de base de données

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Langages | Rapidité | Gratuité | Portabilité | Simplicité  d’administration |
| SQL Server | Très bon | Non | Oui | Non |
| Oracle | Très bon | Non | Oui | Non |
| MySQL | Bon | Oui | Oui | Oui |
| MongoDB | Bon | Oui | Oui | Oui |

Après une analyse approfondie, nous avons opté pour MySQL en raison de ses nombreux avantages par rapport à ses concurrents, ce qui en fait une solution plus avantageuse pour la gestion des données de notre projet.

## OUTILS D’IMPLEMENTATION

L’implémentation d’une application passe impérativement par la sélection d’un ensemble d’outils de développement adéquats. Dans notre cas, il a fallu tenir compte de plusieurs contraintes tant techniques que organisationnelles.

### Langages de programmation et Framework

#### Backend

Le développement backend fait référence au processus de conception, de création, de déploiement et de gestion de la base de données, du serveur, de l'application et de tout autre processus exécuté en arrière-plan d'un site Web ou d'une application. Le backend est responsable de la gestion de la logique métier et de la communication entre le frontend de l'utilisateur et la base de données ou d'autres services. La tâche principale du backend est de garantir que toutes les requêtes envoyées depuis le frontend sont traitées efficacement, en renvoyant les données appropriées ou en exécutant des actions spécifiées, telles que les opérations CRUD (Créer, Lire, Mettre à jour, Supprimer) sur les données. Le backend assure la sécurité, la gestion des données et l'intégration avec d'autres systèmes ou API.

Nous avons d’abord eu à réaliser une étude avant de choisir le langage de programmation adéquat.

Tableau 18: Tableau comparatif des langages de programmation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caractéristiques** | **Java** | **Python** | **PHP** | **JavaScript** |
| Performance | Bonne | Bonne | Moyenne | Bonne (avec Node.js) |
| Facilité d’apprentissage | Moyenne | Bonne | Bonne | Bonne |
| Ecosystème de bibliothèques | Vaste | Large | Large | Large |
| Gestion de la concurrence | Bonne (multithreading) | Limitée | Limitée | Excellente (asynchrone) |
| Support Communauté | Fort | Fort | Fort | Fort |
| Extensibilité | Bonne | Bonne | Bonne | Bonne |
| Utilisation dans les entreprises | Courante | Émergente | Courante | Émergente |
| Performance en temps réel | Excellente | Bonne | Moyenne | Bonne |
| Sécurité | Solide | Variable | Variable | Variable |

Notre choix s’est porté vers Java comme langage serveur car il offre la combinaison idéale de performances, de sécurité, de gestion de la concurrence et d'écosystème, nécessaire pour créer une application bancaire performante, évolutive et sécurisée. De plus, de nombreuses applications bancaires et financières critiques ont été développées en Java, ce qui démontre sa fiabilité et sa capacité à répondre aux exigences du secteur.

Ensuite, il était nécessaire de choisir un Framework solide basé sur le langage de programmation sélectionné.

Tableau 19: Tableau comparatif des Framework java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caractéristiques** | **Spring Boot** | **Jakarta EE** | **Micronaut** | **Quarkus** |
| Popularité | Très élevée | Elevée | Faible | Faible |
| Performance | Bonne | Bonne | Excellente | Excellente |
| Evolutivité | Excellente | Excellente | Excellente | Excellente |
| Particularités | - Prise en charge de l'inversion de contrôle et de l'injection de dépendances  - Intégration de Spring Security pour la sécurité  - Facilité de création d'API REST  - Large écosystème de modules | - Basé sur les spécifications Java EE  - Intégration de la sécurité, de la persistance et des transactions  - Grande compatibilité avec les serveurs d'application Java | - Faible empreinte mémoire et temps de démarrage rapide  - Prise en charge native de GraalVM  -Annotations similaires à Spring pour la configuration | - Conçu pour les conteneurs et le déploiement cloud  -Performances élevées grâce à l'utilisation de GraalVM  - Démarrage ultra-rapide  - Prise en charge des normes Java EE |
| Sécurité | Solide | Solide | Solide | Solide |
| Documentation | Très complète | Complète | Bonne | Bonne |

Spring Boot fait partie intégrante de l'écosystème Spring, l'un des Framework Java les plus populaires et complets. Il propose un large éventail de bibliothèques et de modules prêts à l'emploi pour gérer divers aspects tels que la sécurité, la persistance des données, les transactions et la gestion des sessions. Grâce à sa configuration simplifiée, Spring Boot facilite l'intégration avec d'autres technologies et services, comme les bases de données, les services web, les API REST et les systèmes de messagerie, ce qui en fait un choix idéal pour le développement d'applications bancaires modernes.

#### Frontend

Le développement frontend désigne le processus de création et de mise en œuvre des composants côté client d'un site web ou d'une application, avec lesquels les utilisateurs interagissent directement. Il consiste à concevoir et développer l'interface utilisateur (UI) et l'expérience utilisateur (UX) qui constituent la partie visible d'une application.

Pour mener à bien la réalisation de ce projet nous avons utilisé les outils suivants :

* Les langages du niveau web client : **HTML**, **CSS** et **JavaScript** ;
* **PrimeFaces** : C’est une bibliothèque de composants d'interface utilisateur open source populaire pour JavaServer Faces (JSF), une spécification Java permettant de créer des interfaces utilisateur basées sur des composants pour les applications Web. Elle fournit un ensemble complet de composants d'interface utilisateur, tels que des boutons, des tableaux, des graphiques, des boîtes de dialogue, etc., permettant aux développeurs de créer des applications Web dynamiques et réactives.



Figure 34: Logo de PrimeFaces

* **Flutter :** Il s’agit d’un Framework créé par Google en 2015 pour la création d’applications mobiles natives multiplateformes (Android et iOS). Il utilise le langage Dart qui est orienté objet avec une syntaxe de type C++ et peut se compiler en code natif (Java pour Android ou Objective C pour iOS) ou en JavaScript.

### Environnements de développement intégré

* **Niveau Web**:

Éclipse est un environnement de développement intégré (IDE) polyvalent, extensible et open source. Il offre une interface utilisateur personnalisable, une gestion complète des projets, des fonctionnalités avancées de développement et une large gamme de plugins. Grâce à son intégration avec des Framework et des outils populaires, Éclipse facilite le développement de logiciels dans différents langages. Sa communauté active fournit des mises à jour régulières et du support. Eclipse est un choix populaire pour les développeurs en raison de sa flexibilité, de son extensibilité et de son adoption répandue dans l'industrie du développement de logiciels. Il est utilisé dans le cadre de notre projet pour Spring Boot et PrimeFaces.

* **Niveau mobile :**

Android Studio est un environnement de développement proposé par Google, conçu pour créer, tester et optimiser des applications et des outils pour Android. Il inclut le SDK (Software Development Kit) du système de Google. En plus de sa version pour Windows, Android Studio est également disponible pour Linux et Mac.

## OUTILS D’ORGANISATION ET DE COLLABORATION

* **Serveur web de développement**

WampServer (anciennement appelé WAMP5) est une plateforme de développement Web de type WAMP, qui permet d'exécuter des scripts PHP en local, sans connexion à un serveur externe. Il constitue un environnement regroupant trois serveurs (Apache, MySQL et MariaDB), un interpréteur PHP, et phpMyAdmin pour la gestion des bases de données MySQL via une interface Web.

* **Testeur d’API**

Postman est une plateforme dédiée aux API, conçue pour faciliter la création et l’utilisation d’API. Elle simplifie chaque étape du cycle de vie des API et optimise la collaboration, permettant ainsi de développer des API de meilleure qualité plus rapidement. Avec Postman, il est possible de créer et d’exécuter des requêtes HTTP, de les sauvegarder dans un historique pour les rejouer ultérieurement, et surtout de les organiser en collections. Cette organisation permet de regrouper des requêtes de manière fonctionnelle, comme pour une séquence d’ajout d’articles au panier ou un processus d’identification.

* **Versionnage**

GitHub est une plateforme de développement collaboratif qui permet aux développeurs de travailler ensemble sur des projets en ligne. Basée sur le système de gestion de versions Git, elle facilite le suivi de l'évolution du code source d'un projet et la gestion des contributions des différents collaborateurs. GitHub propose de nombreuses fonctionnalités, telles que la signalisation de problèmes, la soumission de solutions, la création et la gestion de branches de code, ainsi que l'automatisation de tâches via des services de déploiement continu.

* **Gestionnaire de projet**

Jira est un outil de gestion de projets, de suivi des bugs et des incidents, développé par Atlassian et lancé pour la première fois en 2002. Initialement conçu pour aider les équipes de développement logiciel à suivre les erreurs et à gérer les versions de leur code, Jira a progressivement évolué pour devenir une solution complète de gestion de projets. Il offre des fonctionnalités adaptées aux développeurs pour suivre le cycle de vie des logiciels, mais s'adresse également aux intervenants non techniques, tels que les chefs de projet ou les équipes de marketing, en leur permettant de suivre l'avancement des tâches, de gérer les priorités et de coordonner les efforts des différentes équipes. Grâce à sa flexibilité, Jira peut être utilisé dans divers secteurs pour la planification, la gestion des workflows et la collaboration sur des projets complexes.

## RESULTATS DE L’IMPLEMENTATION

### Interface de création de compte

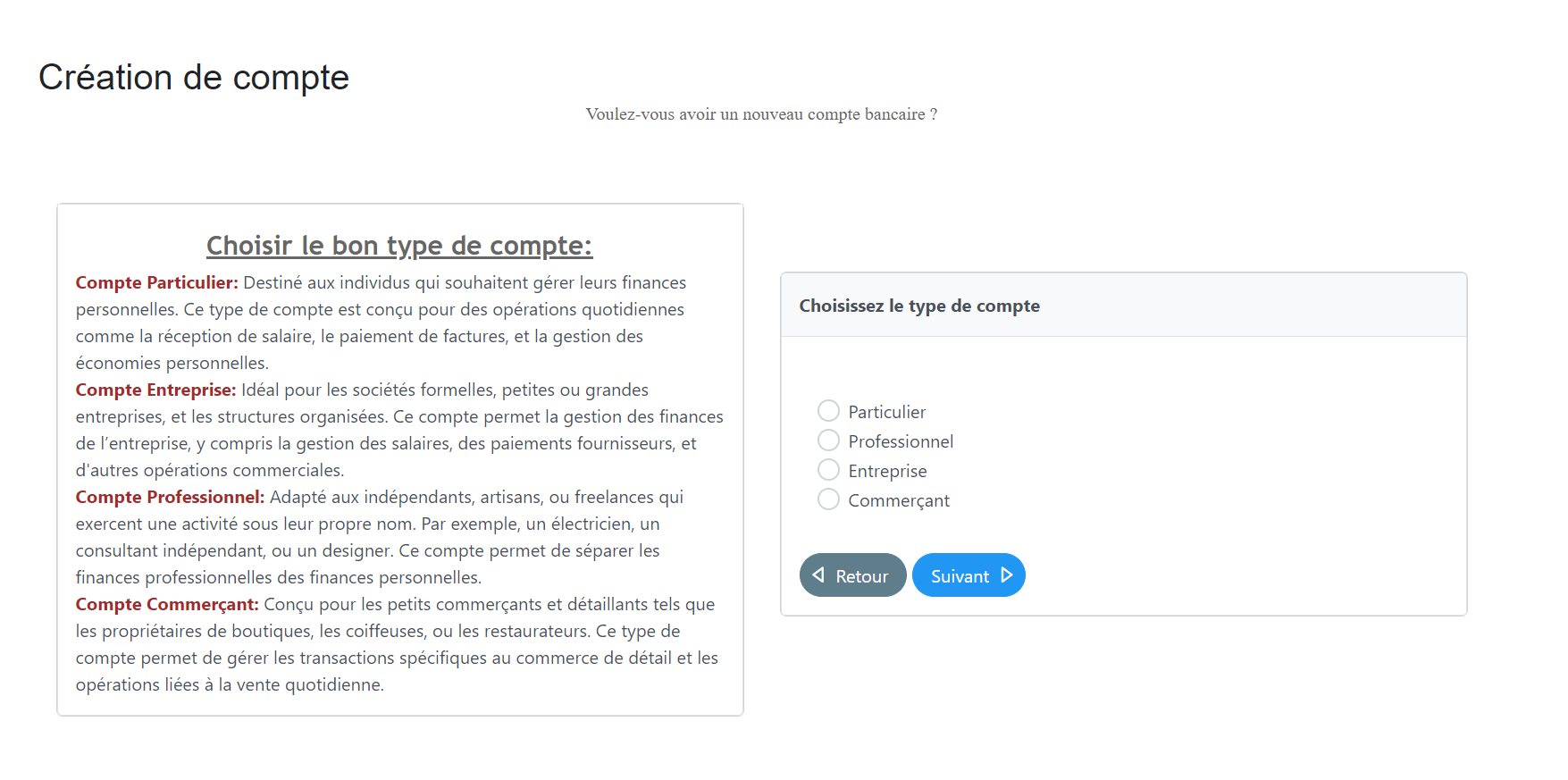


Figure 43: Capture "Interface de création de compte » version web

### Authentification

Figure 44: Captures "interfaces d'authentification""

### Tableau de bord administrateurs

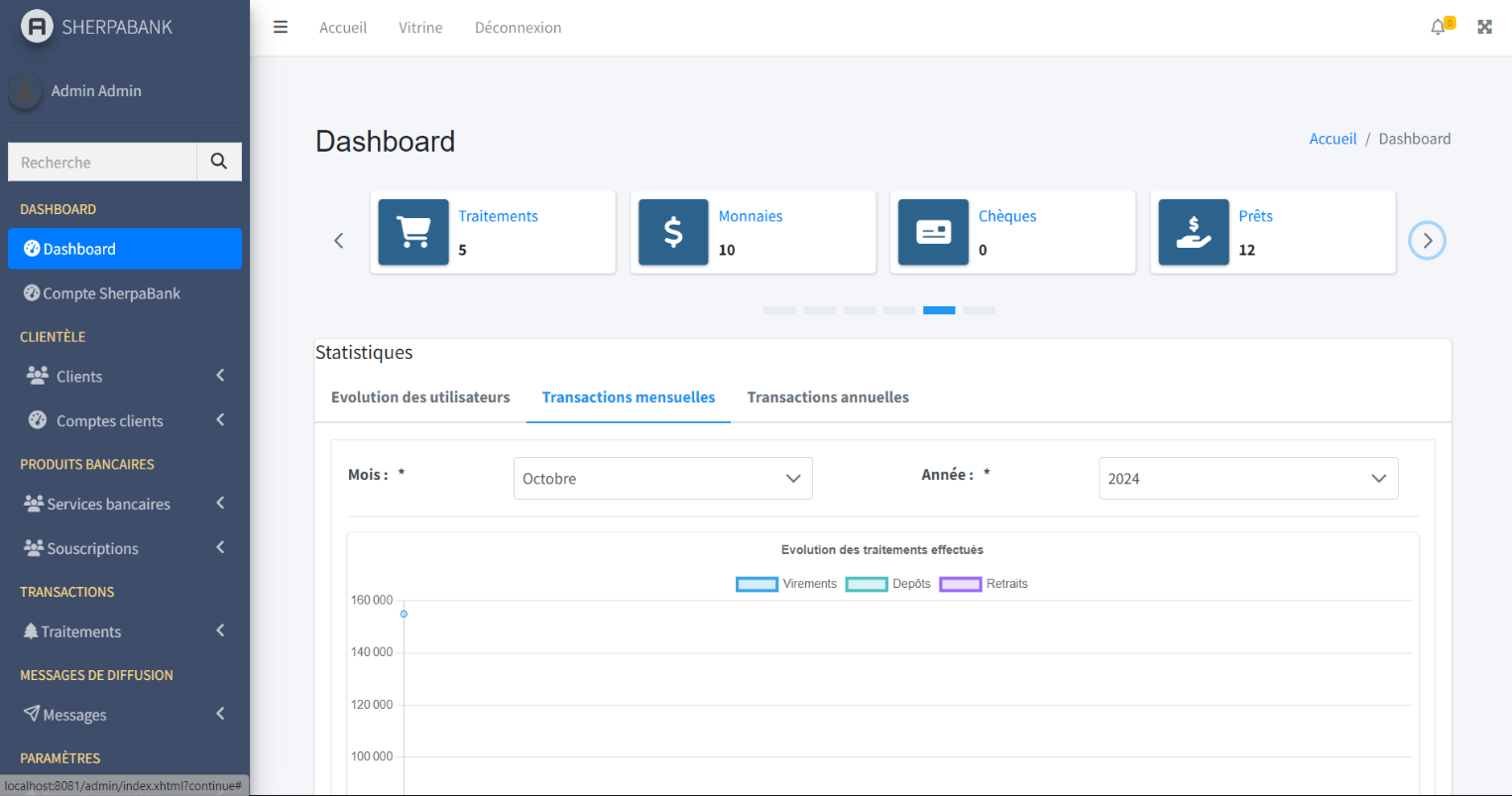


Figure 45: Capture "Dashboard administrateur"

### Tableau de bord client

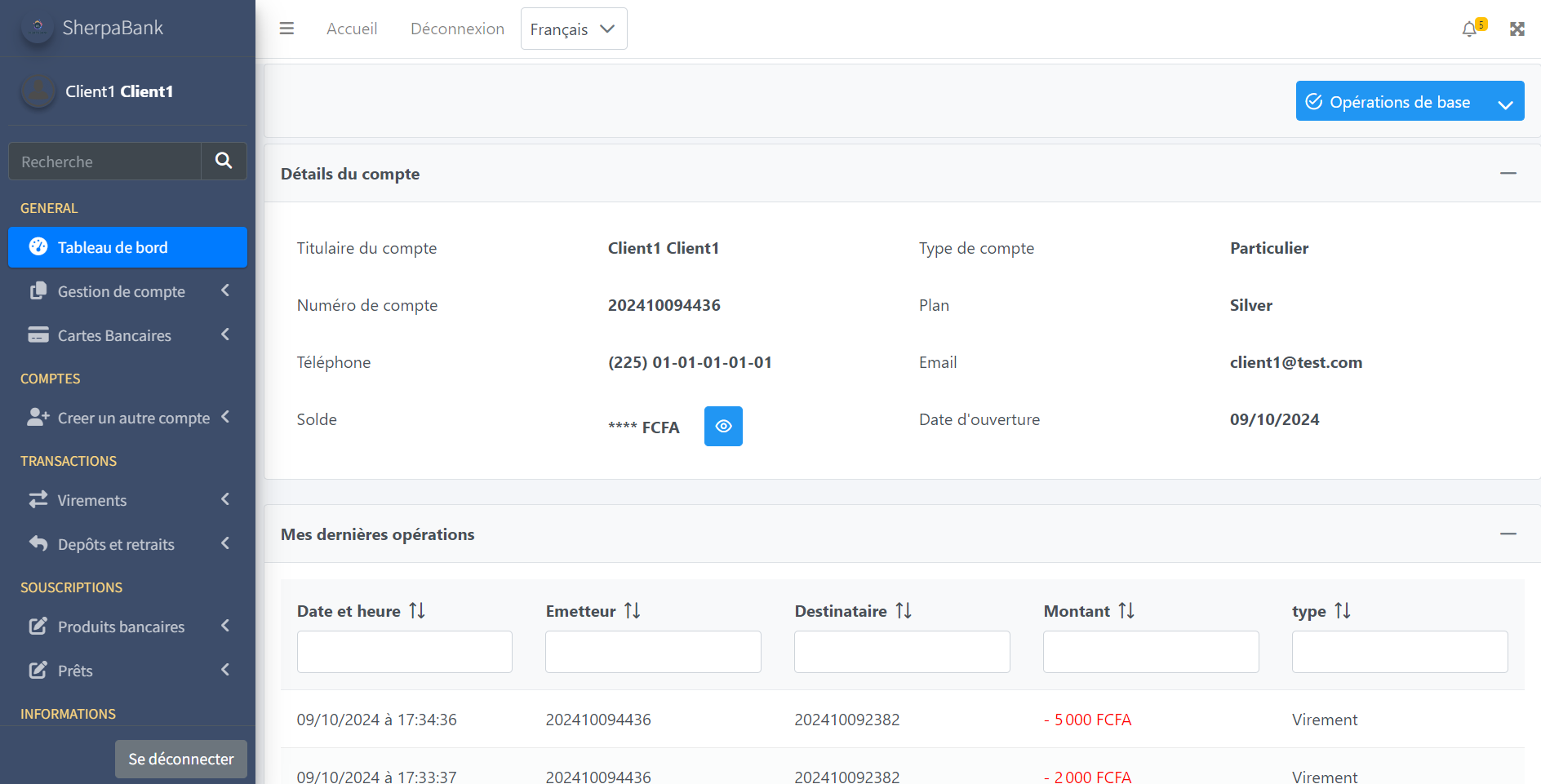
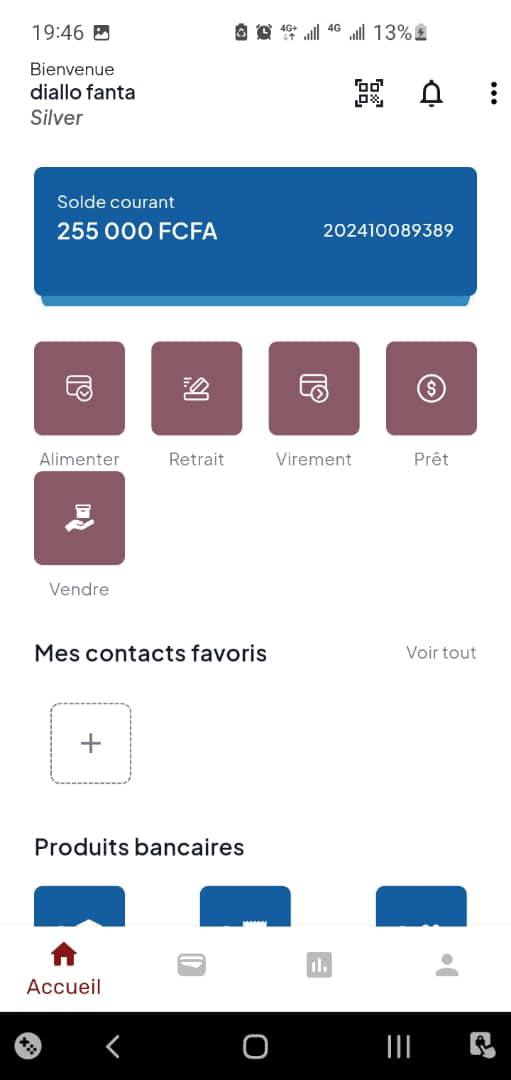


Figure 46: Capture "Dashboard client" version mobile

Figure 47: Capture "Dashboard client" version web

### Interface virement

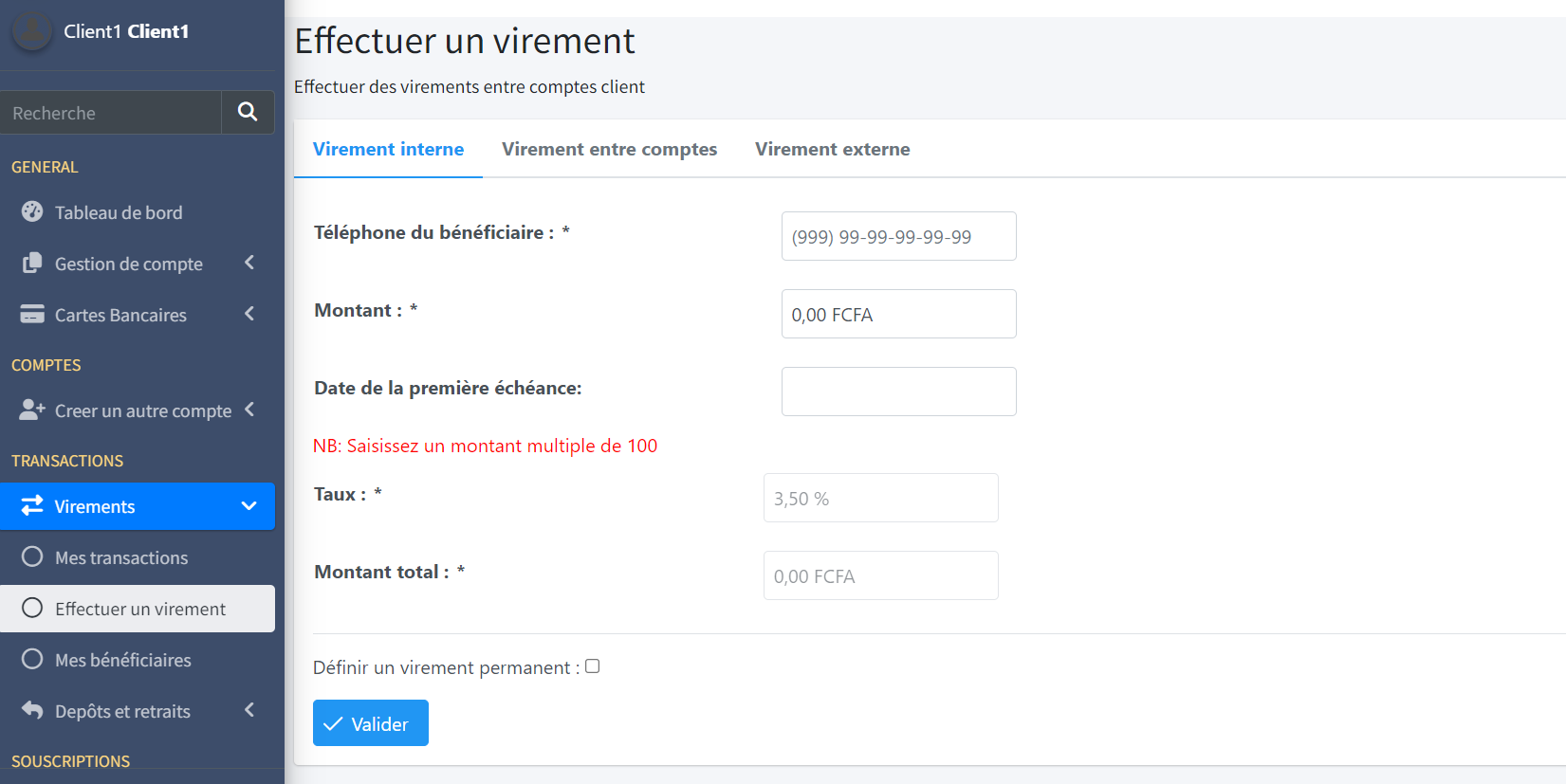


Figure 48: Capture "interface de virement" version web

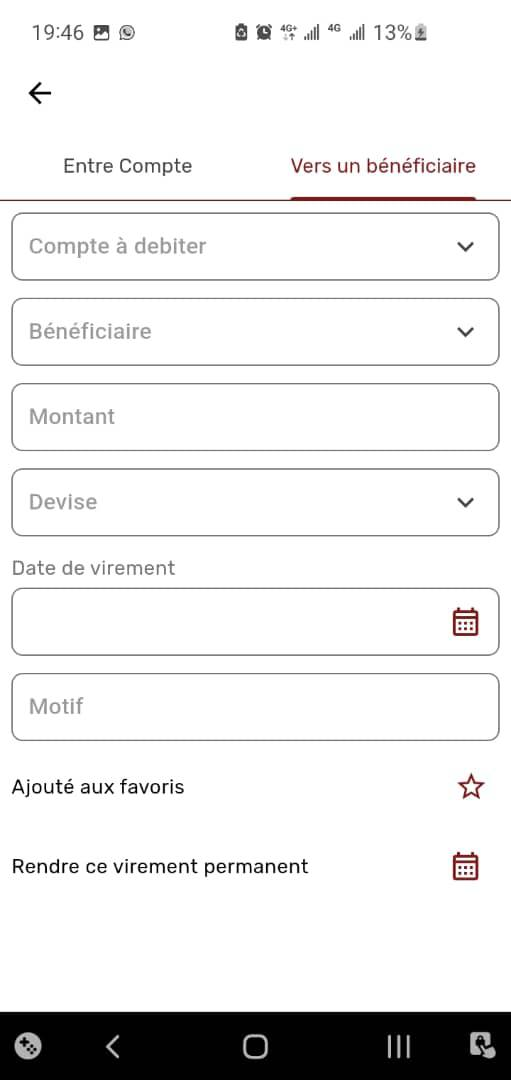


Figure 49: Capture "interface virement" version mobile

### Simulateur de crédit

Figure 50: Capture "simulateur de crédit" version web



Figure 51: Capture "simulateur de crédit" version mobile

### Interface de commande de carte bancaire



Figure 52: Capture « interface de commande de carte bancaire » version Web

# CHAPITRES VII : TRANSITION

*La transition est la quatrième et dernière phase de la méthode EssUP. C’est celle qui finalise le produit. Il s’agit ici de vérifier si le système offre véritablement les services exigés, de détecter les défaillances, de combler les manques et d’adapter le produit à l’environnement de travail des utilisateurs.*

## TESTS

Les tests basés sur le code constituent un aspect crucial du développement logiciel, car ils garantissent l'intégrité et la qualité du code. Ils impliquent de tester systématiquement le code pour identifier les bugs, les défauts et les vulnérabilités avant de déployer le logiciel. En mettant en œuvre un plan de test robuste, les développeurs peuvent minimiser les risques d'erreurs dans le code de production et améliorer la satisfaction des clients.

Les techniques de test de codage jouent un rôle crucial pour assurer le succès de tout projet de développement logiciel. Ces techniques impliquent une variété d'approches de test, allant de l'évaluation de petits composants de code à l'évaluation de la fonctionnalité globale de l'application. Plongeons-nous dans les cinq composants principaux des techniques de test de codage :

### Tests unitaires

Les tests unitaires sont un aspect fondamental des tests logiciels, dans lequel les composants ou fonctions individuels d'une application logicielle sont testés de manière isolée. Cette méthode garantit que chaque unité du logiciel fonctionne comme prévu. En se concentrant sur de petites parties gérables de l'application, les tests unitaires permettent d'identifier et de corriger les bugs dès le début du processus de développement, améliorant ainsi considérablement la qualité et la fiabilité du code.

* Les tests unitaires ont pour objectifs :
* L’isolation d’une section de code ;
* La vérification de l’exactitude du code ;
* La correction des bugs ;
* La réutilisation du code.

Pour les tests unitaires de nos codes nous avons utilisé les outils suivants :

* JUnit en Java pour l’application web ;
* Flutter test pour l’application mobile.

### Tests d’intégration/système

Les tests d'intégration système (TIS), ou simplement tests d'intégration, sont une étape du processus de test logiciel où différents composants ou modules d'un système sont testés ensemble afin de vérifier leur bon fonctionnement en tant qu'unité complète. Ces tests vérifient que le système intégré fonctionne comme prévu, garantissant que toutes les parties fonctionnent de manière transparente lorsqu'elles travaillent ensemble.

Les tests d’intégration système sont cruciaux pour plusieurs raisons :

* Assurer la fonctionnalité de bout en bout ;
* Amélioration de la fiabilité du système ;
* Faciliter une expérience utilisateur transparente ;
* Découverte des problèmes de performances ;

### Tests fonctionnels

Les tests fonctionnels sont un type de test logiciel qui vérifie la fonctionnalité d'un système logiciel ou d'une application en vérifiant que le système se comporte conformément aux exigences fonctionnelles spécifiées et répond aux besoins commerciaux prévus.

L'objectif des tests fonctionnels est de valider les fonctionnalités, les capacités et les interactions du système avec différents composants. Il s'agit de tester les entrées et les sorties du logiciel, la manipulation des données, les interactions des utilisateurs et la réponse du système à divers scénarios et conditions. Les tests fonctionnels ne visent qu'à vérifier si un système fonctionne comme prévu.

Pour les tests fonctionnels, on peut utiliser Selenium, une suite d'outils permettant d'automatiser les interactions avec les navigateurs Web. Il s'intègre aisément avec divers Framework de test dans plusieurs langages de programmation, facilitant ainsi l'automatisation des tests pour les interfaces graphiques Web.

### Tests de régression

Les tests de régression sont un type de test dans le cycle de développement logiciel qui s'exécute après chaque modification pour garantir que celle-ci n'entraîne aucune interruption involontaire. Les tests de régression répondent à un problème courant auquel les développeurs sont confrontés : l'apparition d'anciens bugs lors de l'introduction de nouvelles modifications.

En règle générale, il s'agit d'écrire un test pour un bug connu et de réexécuter ce test après chaque modification de la base de code. Cela vise à identifier immédiatement tout changement qui réintroduit un bug.

Comme outils on peut utiliser Playwright pour exécuter des tests de régression pour les applications Web sur différents navigateurs et Appium pour vérifier la fonctionnalité des applications mobiles.

### Tests d’acceptation

Les tests d'acceptation sont le processus d'évaluation d'une application logicielle pour s'assurer qu'elle répond aux exigences métier spécifiées et aux besoins des utilisateurs avant sa publication. Ils impliquent généralement la validation des fonctionnalités, des performances et de la convivialité pour confirmer que le logiciel est prêt à être déployé et satisfera les utilisateurs finaux.

Les tests d'acceptation constituent une étape finale cruciale du cycle de vie du développement logiciel. Ils visent à vérifier que le logiciel répond aux exigences métier et aux besoins des utilisateurs avant sa sortie. Ils servent de point de contrôle d'assurance qualité pour garantir que le logiciel est fonctionnel, fiable et convivial.

## DEPLOIEMENT

Le déploiement d'applications, également connu sous le nom de déploiement de logiciels, est le processus d'installation, de configuration, de mise à jour et d'activation d'une application ou d'une suite d'applications qui rendent un système logiciel disponible à l'utilisation, comme la facilitation d'une certaine URL sur un serveur.

Le déploiement d'applications est l'une des étapes les plus importantes du processus de développement logiciel, car la stratégie utilisée pour créer, tester et déployer aura un impact direct sur la rapidité avec laquelle une application peut répondre aux changements des préférences ou des exigences des composants et, plus important encore, sur la qualité de chaque changement.

### Sécurité

Il est impossible de conclure la réalisation de notre projet sans aborder la question de la sécurité, qui en est l'un des piliers fondamentaux. La sécurité constitue une stratégie préventive visant à réduire les risques de dysfonctionnement, à éviter les incohérences dans les données, à garantir la confidentialité et à prévenir les intrusions dans le système. Pour un projet informatique de cette envergure, diverses techniques de sécurisation doivent être mises en œuvre à différents niveaux : physique, réseau, transport et applicatif. Concernant la couche applicative, plusieurs solutions ont été adoptées :

* L’utilisation de Spring Security pour contourner les failles de sécurité courantes que rencontre le langage JAVA ;
* Le chiffrement des données à caractère personnel (nom, prénoms, mot de passe, numéro de téléphone, adresse, email, numéro de compte, solde, etc…) dans la base de données de sorte à les rendre incompréhensibles pour qui que ce soit ayant accès à la base de données ;
* L'utilisation des UUID (Identifiants Uniques Universels) en tant qu'alternative aux clés primaires auto-incrémentées dans la base de données. Il s'agit d'un standard d'identifiants générés aléatoirement, garantissant une unicité globale à travers toute l'application.
* L'utilisation de jetons de validité (tokens) pour l'authentification : Cette méthode, basée sur JWT (JSON Web Token), repose sur la création et l'échange sécurisé de jetons, lesquels chiffrent un identifiant utilisateur, un nonce, et un horodatage ;
* Des mesures contre le Cross-Site Scripting (XSS) incluent le retraitement systématique du code HTML généré par l'application avant son envoi au navigateur, ainsi que le filtrage des variables affichées ou stockées afin d'éliminer certains caractères malveillants ;
* L'utilisation de jetons dans les formulaires permet de prévenir les attaques de type Cross-Site Request Forgery (CSRF ou XSRF). Cette vulnérabilité des services d'authentification web consiste à inciter un utilisateur authentifié à exécuter, à son insu, une requête HTTP falsifiée visant une action interne du site, en utilisant ses propres droits ;
* La gestion de droits et d’autorisations en fonction du type d’utilisateur connecté. Cela garantit qu’un utilisateur réalise effectivement les actions qui lui sont autorisées.
* L’association des API à des jetons de validité uniques (tokens). Cela permet de s'assurer qu'une personne non authentifiée ne puisse pas accéder ou exécuter ces API, empêchant ainsi toute tentative d'utilisation non autorisée via des outils comme Postman.

### Hébergement

L'hébergement en informatique fait référence à la mise à disposition d'espace et de ressources sur des serveurs pour stocker, gérer et diffuser des données et des applications sur Internet. Cela permet aux utilisateurs d'accéder à des sites web, des applications et d'autres contenus numériques. De plus, l'hébergement englobe des services complémentaires, tels que la gestion de la sécurité, la sauvegarde des données et l'assistance technique, qui assurent la disponibilité et la performance des services en ligne.[14]

#### Type d’hébergement

Dans le cadre de la mise en production de notre application bancaire multiplateforme, il est essentiel de choisir l’hébergement le plus adapté parmi les cinq (5) grands types existants :

* **Hébergement local (On-premise)** : Un hébergement On-Premise, ou “sur site” en français, désigne l'utilisation des infrastructures de stockage de l'entreprise elle-même. Dans ce modèle d'hébergement, l'utilisateur conserve toutes ses données sur un serveur installé directement au sein de l'organisation ;
* **Hébergement mutualisé**: L'hébergement mutualisé consiste à partager les ressources et la capacité d'un serveur avec d'autres sites web, ce qui permet de diminuer les coûts. Ce type d'hébergement est particulièrement adapté aux petites entreprises qui ont un budget restreint. Parmi les fournisseurs de services qui proposent ce type d'hébergement, on peut mentionner Hostinger, o2switch et PlanetHoster ;
* **Hébergement VPS**: L’hébergement VPS se définit comme un type de service où un serveur physique est divisé en plusieurs serveurs virtuels, chacun fonctionnant de manière indépendante avec ses propres ressources dédiées, telles que la puissance de calcul, la mémoire vive (RAM) et l’espace de stockage. Chaque serveur virtuel (VPS) dispose de son propre système d'exploitation et de ses configurations, offrant une plus grande flexibilité et contrôle.
* À la différence de l’hébergement partagé, où les ressources sont mutualisées entre différents utilisateurs, le VPS garantit une isolation des environnements, permettant ainsi d’éviter les interférences et d’optimiser la gestion des performances pour chaque instance.
* **Hébergement dédié**: L'hébergement dédié est une option d'hébergement dans laquelle un ou plusieurs serveurs physiques sont dédiés à un seul client professionnel. Le client a un contrôle total sur la machine, ce qui lui permet de l'optimiser en fonction de ses besoins spécifiques, notamment en termes de performances et de sécurité. Le fournisseur d'hébergement fournit le serveur physique et l'environnement, les services associés et le support technique.
* **Hébergement Cloud computing**: L'hébergement cloud fait référence à un type de service d'hébergement Web dans lequel les ressources et l'infrastructure nécessaires pour héberger un site Web ou une application sont réparties sur plusieurs serveurs virtuels. Ces serveurs virtuels font partie d'un réseau plus vaste de serveurs physiques interconnectés appelés « cloud ». L 'hébergement cloud utilise la technologie de virtualisation pour créer plusieurs serveurs virtuels au sein d'un serveur physique, chacun fonctionnant indépendamment avec des ressources allouées distinctes.

Nous allons effectuer une comparaison des différents types d’hébergement afin de sélectionner la meilleure option pour la suite du travail :

Tableau 20: Tableau comparatif des types d'hébergement

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Critères** | **On-premise** | **Mutualisé** | **VPS** | **Dédié** | **Cloud computing** |
| Coût | Élevé | Faible | Modéré | Élevé à modéré | Variable |
| Scalabilité | Faible | Faible | Moyenne | Faible à moyenne | Très élevée |
| Maintenance | Élevée | Faible | Faible à moyenne | Moyenne | Faible |
| Contrôle | Total | Faible | Moyenne | Élevé | Variable |
| Performance | Élevée | Variable | Moyenne à élevée | Élevée | Très élevée |
| Sécurité | Très élevée | Moyenne | Élevée | Élevée | Très élevée |

Le meilleur choix ici semble être l’hébergement cloud. Cependant, bien que le cloud offre une grande scalabilité et flexibilité, il exige une certaine expertise en gestion des infrastructures cloud, ce qui peut être un obstacle pour certaines entreprises. De plus, les ressources partagées peuvent entraîner des performances variables et des préoccupations de sécurité, car les données sensibles sont hébergées sur des serveurs tiers, compliquant ainsi la conformité réglementaire. En revanche, **l'hébergement dédié** garantit un contrôle total sur l'environnement, des performances constantes et une sécurité renforcée, ce qui en fait un choix préférable pour notre projet.

#### Fournisseur d’hébergement

Pour la suite du travail, il est essentiel de choisir le fournisseur d'hébergement dédié le plus adapté aux besoins de la plateforme bancaire, en tenant compte des performances, de la gestion des ressources, et des exigences de sécurité. Une comparaison approfondie des différentes solutions disponibles permettra de garantir une infrastructure stable et évolutive.

* **LWS** : LWS (Ligne Web Services) est un fournisseur d’hébergement web français qui offre une gamme complète de services d’hébergement, allant de l’hébergement mutualisé à l’hébergement dédié. Fondé en 1999, LWS est reconnu pour ses solutions adaptées aux particuliers comme aux entreprises, avec des offres compétitives en termes de prix et de performance.

En plus des serveurs dédiés, LWS propose également des services comme l'enregistrement de noms de domaine, des certificats SSL, et des solutions de création de site web.

* **OVHCloud :** OVH est l'un des principaux fournisseurs d'hébergement en Europe, fondé en 1999 en France. Il est particulièrement reconnu pour ses services d’hébergement cloud et ses solutions de serveurs dédiés. OVH propose une large gamme de serveurs dédiés, avec des ressources allouées de manière exclusive, garantissant de hautes performances, une grande flexibilité et un contrôle total sur la configuration du serveur. Leurs serveurs dédiés sont adaptés aussi bien aux entreprises qu’aux utilisateurs exigeants, offrant un bon rapport qualité-prix.
* **Plesk :** Plesk est un fournisseur d'hébergement qui se spécialise dans la gestion de serveurs dédiés, offrant une interface conviviale et des outils puissants pour simplifier l'administration des serveurs. Fondé en 2000, Plesk est particulièrement apprécié des développeurs et des agences web grâce à sa capacité à gérer plusieurs sites, applications et services à partir d'un seul tableau de bord.

Les serveurs dédiés proposés par Plesk sont optimisés pour la performance, la sécurité et la scalabilité, ce qui les rend adaptés aux entreprises de toutes tailles. Plesk permet également une intégration facile avec des solutions tierces, facilitant ainsi le déploiement d'applications web complexes.

Nous allons effectuer une comparaison des fournisseurs d’hébergement pour la suite du travail :

Tableau 21: Tableau comparatif des fournisseurs d'hébergement dédié

|  |  |  |  |
| --- | --- | --- | --- |
| **Critère** | **LWS** | **OVHCloud** | **Plesk** |
| Type de service | Hébergement partagé et dédié | Hébergement cloud et dédié | Gestion de serveurs dédiés |
| Interface | Interface simple et conviviale | Console de gestion avancée | Interface intuitive pour la gestion |
| Performances | Performances correctes pour les petites et moyennes entreprises | Haute performance, scalabilité | Optimisé pour les applications web |
| Support technique | Support disponible par téléphone et chat | Support en ligne, documentation complète | Support technique et documentation en ligne |
| Options de personnalisation | Options limitées pour les serveurs dédiés | Personnalisation complète des serveurs | Personnalisation via des extensions et plugins |
| Sécurité | Protection DDoS, sauvegardes automatiques | Sécurité avancée, firewall et sauvegardes | Intégration de fonctionnalités de sécurité |
| Tarification | Tarifs compétitifs pour les petits budgets | Large gamme de prix selon les besoins | Tarification variable en fonction des options |

Pour la suite du travail, le choix de **Plesk** comme fournisseur d’hébergement dédié se justifie par plusieurs aspects. Tout d’abord, Plesk offre une interface de gestion particulièrement intuitive, facilitant la gestion des serveurs et des applications, un atout précieux pour les utilisateurs à la recherche d'efficacité. De plus, son environnement flexible, qui permet de gérer plusieurs plateformes (Windows et Linux), offre une grande adaptabilité aux besoins spécifiques des projets. Enfin, Plesk propose des solutions robustes pour la sécurité et les mises à jour automatiques, garantissant une gestion simplifiée des serveurs tout en maintenant des niveaux de sécurité élevés. Ces caractéristiques font de Plesk un choix stratégique pour un hébergement dédié fiable et facile à gérer.

### Nom de domaine



#### Acquisition d’un nom de domaine

Pour garantir une présence en ligne optimale et sécurisée, il est essentiel de sélectionner un registraire fiable pour l'achat et la gestion du nom de domaine. Ce choix joue un rôle crucial dans la configuration du domaine, la gestion des sous-domaines, ainsi que l'intégration des certificats de sécurité nécessaires à la protection des données sur notre plateforme bancaire.

Parmi les registraires existants nous avons choisi **Hostinger** en raison de son excellent rapport qualité-prix et des fonctionnalités incluses. Contrairement à d'autres options souvent plus coûteuses avec des frais supplémentaires pour certains services, Hostinger propose une gestion simple et complète des domaines, incluant la gestion DNS et la protection de la confidentialité sans frais cachés. De plus, son interface conviviale et ses options d'assistance facilitent la gestion, tout en offrant une bonne intégration avec d'autres services, ce qui le rend plus pratique et abordable que d'autres alternatives.

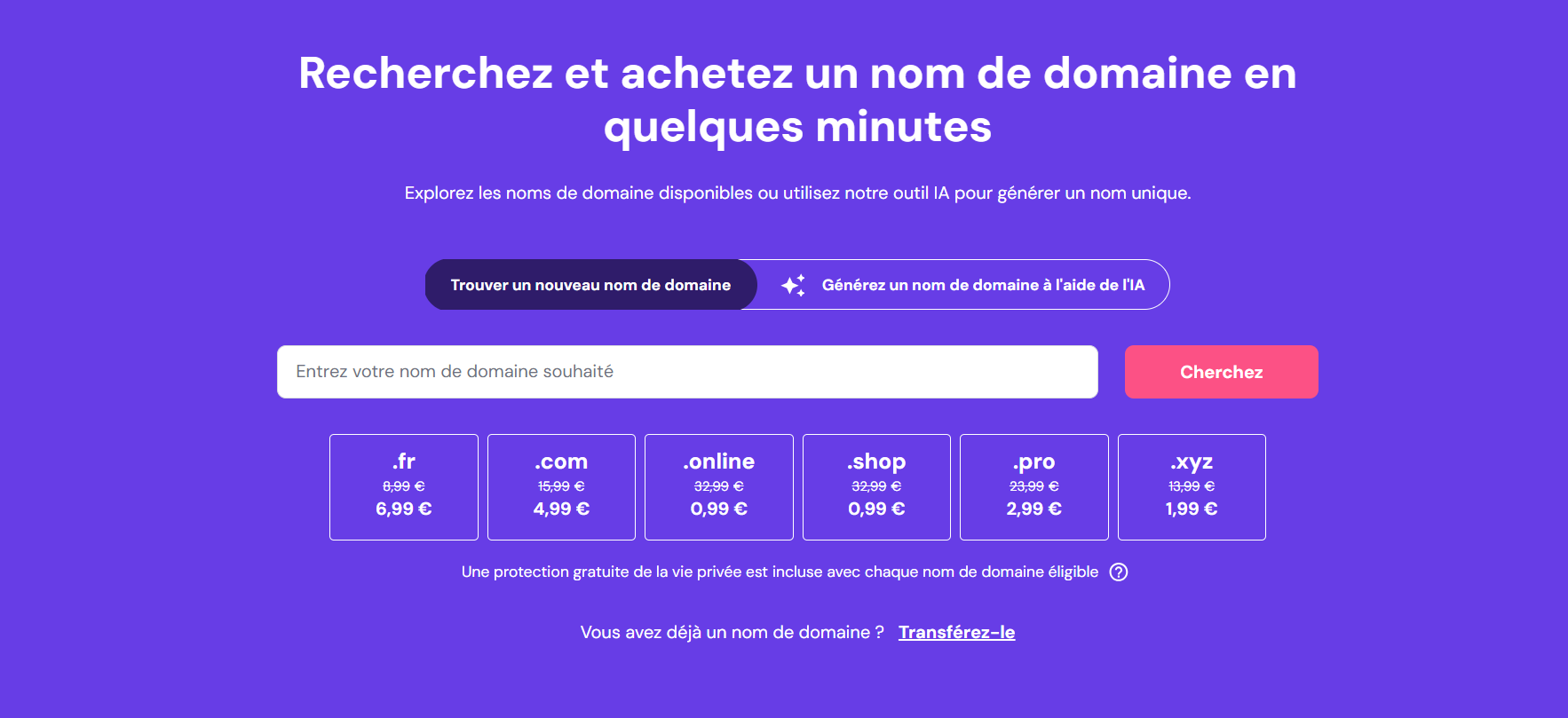


Figure 56: Interface d'achat de nom de domaine sur Hostinger

#### Configuration DNS

Pour assurer le bon fonctionnement de notre plateforme bancaire en ligne, il est indispensable de configurer correctement le DNS afin de lier notre nom de domaine au serveur dédié. Cette étape permet de diriger le trafic vers notre infrastructure, d’assurer la disponibilité des services, et de garantir la sécurité des échanges via une gestion efficace des enregistrements DNS.

Pour accomplir cette configuration, nous suivrons les étapes suivantes :

* Ajout de domaine sur l’interface de Plesk en utilisant le nom du domaine acheté sur Hostinger ;
* Accéder aux paramètres du domaine acheté sur son compte Hostinger afin de configurer les enregistrements DNS nécessaires ;
* Dans la zone DNS sur Hostinger nous allons configurer les enregistrements principaux pour que le domaine pointe vers l’adresse IP de notre serveur dédié :
* Enregistrement **A** pour lier le domaine principal à l'IP de notre serveur dédié sous Plesk.
* Enregistrement **CNAME** afin d’associer un nom d’alias tel que **www** à notre nom de domaine.
* Une fois la configuration terminée et enregistrée nous attendons la propagation DNS qui peut prendre quelques minutes à 24h ;
* Sécurisation des communications en utilisant un certificat SSL comme **Let’s Encrypt** proposé gratuitement par Plesk ;

### Optimisation



#### Conteneurisation

La conteneurisation est un processus de déploiement de logiciels qui regroupe les applications avec toutes les bibliothèques, fichiers, configurations et binaires nécessaires à leur exécution dans une seule image exécutable. Cela isole les applications et leur permet de s'exécuter, en partageant uniquement le noyau du système d'exploitation avec la machine hôte.

Dans le cadre de notre projet, la conteneurisation pourrait grandement simplifier le déploiement et la gestion de notre application. Voici quelques avantages spécifiques à envisager :

* **Isolation des services** : Chaque composant de notre application (API, base de données) peut être isolé dans un conteneur séparé, garantissant une meilleure sécurité et évitant les conflits de dépendances ;
* **Portabilité**: Les conteneurs peuvent être exécutés sur n’importe quel environnement (serveur dédié, cloud, local), ce qui rend les déploiements et les migrations plus simples et prévisibles ;
* **Scalabilité**: Avec l'utilisation d'orchestrateurs, il est possible d'ajuster dynamiquement le nombre de conteneurs en fonction des variations de la demande. Cela est particulièrement avantageux pour une application bancaire nécessitant une haute disponibilité et une gestion efficace des pics de trafic.
* **Evolutivité**: Les conteneurs offrent la possibilité d'ajouter de nouvelles fonctionnalités, mises à jour et améliorations sans perturber les applications initiales. Ainsi, ils permettent aux applications de se développer tout en minimisant l'utilisation des ressources.

Pour la conteneurisation nous utiliserons Docker qui est un logiciel open source qui permet de créer et d’utiliser des conteneurs Linux. C’est probablement la technologie la plus connue et la plus utilisée dans le monde en matière de conteneur.

#### Intégration continue/Déploiement continu

L'intégration continue (CI) et le déploiement continu (CD) sont des pratiques fondamentales dans le développement logiciel moderne, particulièrement dans des projets complexes comme le nôtre. Ces pratiques permettent d’automatiser les tests, la construction, et le déploiement du code, garantissant ainsi un développement plus rapide et plus fiable.

**L'intégration continue** consiste à intégrer fréquemment le code de chaque développeur dans une branche principale, avec des tests automatisés pour vérifier que les nouvelles modifications n'introduisent pas de bugs. Dans le contexte de notre projet à chaque changement de code (comme l'ajout d'une nouvelle fonctionnalité ou la correction d'un bug), le code est testé automatiquement via des pipelines CI.

**Le déploiement continu** va au-delà en automatisant l'intégralité du processus de mise en production. Une fois que le code passe les tests dans la phase CI, il est automatiquement déployé dans un environnement de staging ou directement en production. Dans notre projet, cela peut se traduire par :

* La mise à jour automatique de notre serveur dédié (ou conteneur) via Docker à chaque validation de code réussie.
* Le déploiement de nouvelles fonctionnalités sans interruption de service, ce qui est crucial dans un environnement bancaire où la disponibilité est essentielle.

Sur Plesk nous avons la possibilité de lier notre dépôt Git à notre serveur dédié ce qui peut être considéré comme action de CI/CD même si cela reste relativement basique par rapport à des solutions complètes existantes.

En liant notre dépôt Git à Plesk nous aurons la possibilité d’automatiser une partie du déploiement mais sur le long terme il serait préférable d’utiliser une solution comme GitHub Actions ou Jenkins.

SherpaBank étant développée avec le SDK Flutter, il est donc possible de générer une version spécifique pour les plateformes Android et iOS sans modifier le code source. L’application mobile sera déployée sur Play Store pour les utilisateurs d’Android et sur App Store pour les utilisateurs d’iOS. Pour illustrer cette opération, nous avons recours à un diagramme de déploiement. Ce type de diagramme représente la structure physique des ressources matérielles du système et met en évidence la distribution des différents composants sur ces infrastructures. Chaque ressource étant représentée par un nœud, le diagramme de déploiement décrit la répartition des composants sur ces nœuds et les connexions qui existent entre eux ou entre les composants. Les diagrammes de déploiement existent sous deux formes : spécification et instance.

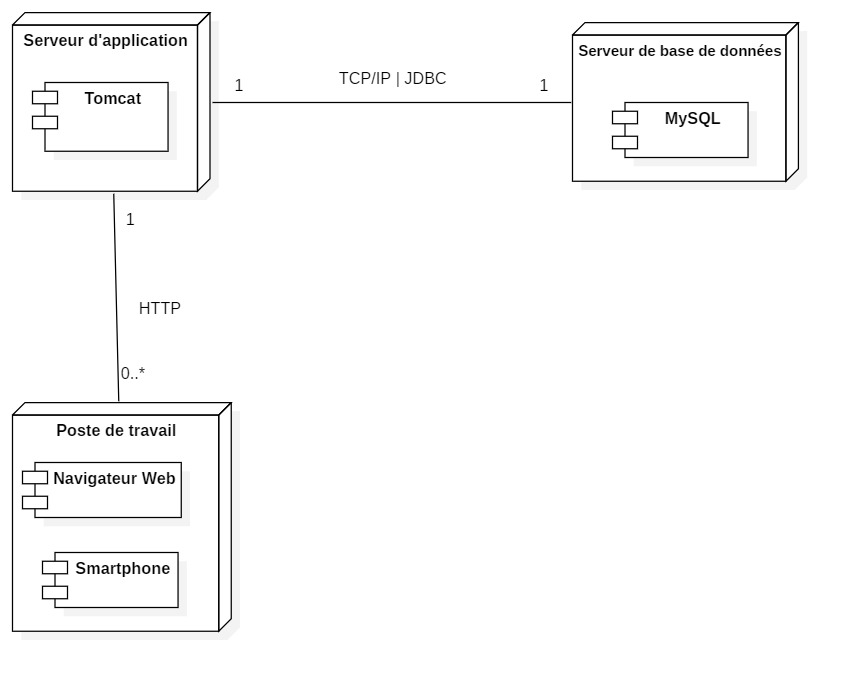


Figure 58: Diagramme de déploiement

## COÛT DU PROJET

Pour assurer le succès de notre projet, une estimation des coûts est indispensable. Celle-ci prend en compte trois facteurs principaux : la durée effective de la mise en œuvre du système (distincte de la période de stage), le matériel requis, ainsi que la main-d'œuvre mobilisée pour le développement. Nous avons principalement opté pour des solutions open-source, ce qui a considérablement réduit les dépenses.

Tableau 22: Coût du projet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ressources | Quantité | Durée | Prix unitaire (F CFA) | Prix total (F CFA) |
| Ordinateurs | 6 |  | 400.000 | 2.400.000 |
| Connexion internet 10Mbps |  | 8 mois | 30.000 | 240.000 |
| Ingénieurs | 6 | 8 mois | 500.000 | 24.000.000 |
| Service Sms Twilio | 18.250 | 1 an | 5 | 91.250 |
| Hébergement web |  | 1 an | 60.000 | 60.000 |
| Hébergement Play Store |  | Illimité | 15.500 | 15.500 |
| Hébergement App Store |  | 1 an | 62.000 | 62.000 |
| **Total** | | | | **26.868 .750** |

Un montant total de 26.868 .750FCFA est donc nécessaire pour mettre en place notre solution.

Cette phase de transition finalise donc notre travail. En passant par toutes les phases de la méthode EssUP/UML, nous sommes arrivés au terme de notre étude. Achevons désormais notre présentation

# CONCLUSION GENERALE ET PERSPECTIVES

En définitive, notre projet de fin d’étude, réalisé au compte de la société J3A FINANCE, consistait à réaliser et à déployer une application bancaire multiplateforme permettant aux utilisateurs d'accéder aux services financiers à la fois sur des supports web et mobiles. Pour y parvenir, nous avons procédé à une analyse approfondie des solutions existantes et du cahier de charges afin de mieux comprendre les besoins métiers et de mettre en place une solution adéquate. La méthode agile EssUP du processus unifié a été notre guide tout au long de cette étude. Elle nous a permis d’achever la conception du système en fournissant, en plus, les diagrammes qui ont mis en exergue les données et les traitements relatifs au système.

Par ailleurs, ce projet a été d’un apport considérable, non seulement à notre formation personnelle dans le métier d’ingénieur logiciel mais aussi à notre intégration dans le monde socio-professionnel. En outre, il nous a permis d’approfondir nos connaissances sur la méthode EssUP/UML, le développement web, le développement mobile ainsi que la gestion de projet en équipe grâce au cadre méthodologique Scrum.

Finalement nous sommes satisfaits d’avoir réalisé la quasi-totalité des objectifs qui nous ont été confiés. Les perspectives futures de ce projet incluent principalement le déploiement de la multiplateforme en environnement de production, afin de la rendre accessible à une plus grande échelle. D'autres optimisations et ajouts de fonctionnalités pourront également être envisagés pour répondre aux besoins évolutifs des utilisateurs.

# ANNEXES

**ANNEXE 1** : Diagramme de planification des tâches

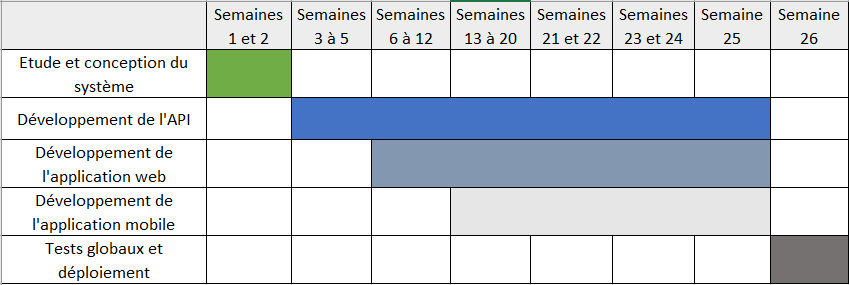


Figure 59: Diagramme de Gantt

# REFERENCES BIBLIOGRAPHIQUES

**MEMOIRES**

**[01]** ZEMMOURI Amel & BOULAICHE Nassima, Conception et réalisation d’une application mobile pour la gestion des services bancaires, Mémoire : Informatique. Centre Universitaire Abd elhafid Boussouf Mila : 2016

**[02]** Guessam Cyrille KOFFI, Conception et réalisation d'un comparateur-souscripteur d'assurance en ligne, Mémoire : Informatique. Institut National Polytechnique Felix Houphouët Boigny (INP-HB) : 2021

**[03]** Kan Juste Elfried KOUASSI, Conception et réalisation d’une application bancaire avec étude d’un module de calcul des taux d’intérêt d’emprunts à base d’intelligence artificielle, Mémoire : Informatique. Institut National Polytechnique Felix Houphouët Boigny (INP-HB) : 2023

**[04]** AIT ZEGGANE Belaid & OULMANE Hakim, Conception et réalisation d’une application banque en ligne, Mémoire : Informatique. Université Mouloud MAMMERI de Tizi-Ouzou

**OUVRAGES**

**[05]** Gilles, ROY. Informatique (en ligne). Québec : Presses de l’Université du Québec, 2007, 534 Pages. [Consulté le 20 mai 2024].

**[06]** Laurent, AUDIBERT. UML 2.0. France : Institut Universitaire de Technologie de Villetaneuse – Département Informatique, 128 Pages. [Consulté le 07 juin 2024].

**[07]** Adjé, ASSALE. Cours UML. Yamoussoukro, Côte d'Ivoire, 124 Pages.

[Consulté le jeudi 25 juin 2024].

**[08]** Bi Tra, Gooré. Services Réseaux. Yamoussoukro, Côte d’Ivoire, 135 Pages. [Consulté le 23 juillet 2024]

# REFERENCES WEBOGRAPHIQUES

**[09]** Processus Unifié [Consulté le 16 juin 2024 à 20 : 36]

<https://fr.wikipedia.org/wiki/Processus_unifi%C3%A9#:~:text=Le%20processus%20unifi%C3%A9%20est%20une,architecture%20et%20les%20mod%C3%A8les%20UML>.

**[10]** Système de gestion de base de données [consulté le 14 août 2024 à 15:31]

<https://www.opendatasoft.com/fr/glossaire/systemes-de-gestion-de-base-de-donnees/#:~:text=donn%C3%A9es%20(SGBD)%20%3F-,Un%20syst%C3%A8me%20de%20gestion%20de%20base%20de%20donn%C3%A9es%20est%20un,aspects%20primaires%20de%20la%20database>.

**[11]** Les banques traditionnelles [consulté le 17 août 2024 à 9:24].

<https://skaleet.com/blog/au-dela-de-la-banque-traditionnelle-la-course-aux-ecosystemes>

**[12]** Le taux de bancarisation en Côte d’Ivoire [consulté le 19 août 2024 à 10:50]

<https://www.yeclo.com/le-taux-de-bancarisation-en-cote-divoire#:~:text=Selon%20un%20rapport%20publi%C3%A9%20par%20la%20BCEAO%20en,%2820%2C4%25%29%2C%20du%20S%C3%A9n%C3%A9gal%20%2819%2C6%25%29%20et%20du%20Niger%20%289%2C1%25%29>

**[13]** Différence entre banque en ligne et néo-banque [consulté le 19 2024 août à 11:20]

<https://www.bforbank.com/blog/parlons-argent/banques-ligne-neo-banques-differences>

**[14]** Hébergement Informatique [consulté le 04 septembre 2024 à 15h13]

<https://fcmicro.net/hebergement-informatique-comment-cela-fonctionne/>

**[15]** Conteneurisation informatique [consulté le 13 septembre 2024 à 19:39]

<https://www.qiminfo.ch/conteneurisation-informatique/>

# GLOSSAIRE

**Cross-site Scripting** : Le Cross-Site Scripting (XSS) est une faille de sécurité web qui permet l'injection de scripts malveillants dans une page, ce qui peut déclencher des actions indésirées sur les navigateurs des utilisateurs lorsqu'ils accèdent à cette page.

**Fintech** : Le terme Fintech est une contraction de « Finance » et « Technologie ». Il fait référence aux entreprises qui utilisent les technologies de l'information et de la communication pour proposer des services financiers de manière innovante.

**Framework** : Ensemble de concepts, de règles, de pratiques et de composants logiciels préétablis qui fournissent une structure de base pour le développement d'applications. Il s'agit d'un cadre de travail qui facilite la conception, le développement et la maintenance de logiciels en fournissant des fonctionnalités prêtes à l'emploi.

**Healthtech** : Ce terme résulte de la combinaison des mots « santé » et « technologie ». Il désigne les entreprises qui intègrent des technologies de pointe, comme l'intelligence artificielle, l'Internet des objets, la télémédecine, les applications mobiles ou encore la biotechnologie, dans le but d'améliorer les soins de santé, la prévention, le diagnostic, le traitement et la gestion des données médicales

**Injection SQL** : L’injection SQL est un groupe de méthodes d'exploitation de failles de sécurité d'une application interagissant avec une base de données. Elle permet d'injecter dans la requête SQL en cours un morceau de requête non prévu par le système et pouvant compromettre la sécurité.

**Insurtech** : contraction des mots "insurance" (assurance) et "technology" (technologie), désigne les startups innovantes œuvrant pour la digitalisation et l’optimisation des pratiques traditionnelles dans le secteur de l’assurance.

**Traitement** : Une opération financière spécifique, telle qu'un paiement, un virement, un retrait ou un dépôt effectué par un client ou une banque.

# TABLE DES MATIERES

[DEDICACE I](#_Toc180103372)

[REMERCIEMENTS II](#_Toc180103373)

[SOMMAIRE III](#_Toc180103374)

[LISTE DES SIGLES ET ABREVIATIONS V](#_Toc180103375)

[LISTE DES FIGURES VIII](#_Toc180103376)

[LISTE DES TABLEAUX X](#_Toc180103377)

[AVANT PROPOS XI](#_Toc180103378)

[RESUME XII](#_Toc180103379)

[INTRODUCTION GENERALE 1](#_Toc180103380)

[PARTIE 1 : CADRE ET CONTEXTE DU PROJET 2](#_Toc180103381)

[CHAPITRE I : PRESENTATION DE L’ORGANISME D’ACCUEIL 3](#_Toc180103382)

[I. PRESENTATION DE J3A FINANCE 3](#_Toc180103383)

[1. Présentation 3](#_Toc180103384)

[2. Missions et ambitions 3](#_Toc180103385)

[II. ORGANISATION STRUCTURELLE DE J3A FINANCE 4](#_Toc180103386)

[1. Organigramme de l’entreprise 4](#_Toc180103387)

[2. Environnement de travail 4](#_Toc180103388)

[CHAPITRE II : PRESENTATION DU PROJET 6](#_Toc180103389)

[I. ETUDE DE L’EXISTANT 6](#_Toc180103390)

[1. Présentation de l’existant 6](#_Toc180103391)

[2. Critique de l’existant 8](#_Toc180103392)

[II. PRESENTATION GENERALE DU PROJET 9](#_Toc180103393)

[1. Contexte du projet 9](#_Toc180103394)

[2. Problématique du projet 9](#_Toc180103395)

[3. Objectifs du projet 10](#_Toc180103396)

[4. Contraintes et planification 10](#_Toc180103397)

[PARTIE 2 : ETUDE CONCEPTUELLE DU PROJET 12](#_Toc180103398)

[CHAPITRE III : METHODOLOGIE DU PROJET 13](#_Toc180103399)

[I. GENERALITES SUR LES METHODES D’ANALYSES 13](#_Toc180103400)

[1. Méthodes cartésiennes 13](#_Toc180103401)

[2. Méthodes systémiques 13](#_Toc180103402)

[3. Méthodes objets 13](#_Toc180103403)

[II. PRESENTATION DE QUELQUES METHODES D’ANALYSE 14](#_Toc180103404)

[1. Présentation de MERISE 14](#_Toc180103405)

[2. Présentation du Processus Unifiée 14](#_Toc180103406)

[3. Présentation de la déclinaison EssUP 16](#_Toc180103407)

[CHAPITRE IV : INITIALISATION 19](#_Toc180103408)

[I. SPECIFICATIONS DES BESOINS 19](#_Toc180103409)

[1. Spécifications fonctionnelles 19](#_Toc180103410)

[2. Spécifications non-fonctionnelles 20](#_Toc180103411)

[II. DELIMITATION DU SYSTEME 20](#_Toc180103412)

[1. Identification des acteurs 20](#_Toc180103413)

[2. Identification des cas d’utilisation 21](#_Toc180103414)

[3. Diagramme de cas d’utilisation 23](#_Toc180103415)

[CHAPITRE V : ELABORATION 26](#_Toc180103416)

[II. ANALYSE : DESCRIPTION TEXTUELLE DES CAS D’UTILISATIONS 27](#_Toc180103417)

[1. Cas « S’inscrire » 27](#_Toc180103418)

[2. Cas « S’authentifier » 29](#_Toc180103419)

[3. Cas « Recharger son compte » 30](#_Toc180103420)

[4. Cas « Effectuer un retrait » 30](#_Toc180103421)

[5. Cas « Effectuer un virement » 31](#_Toc180103422)

[6. Cas « Souscrire à une épargne » 32](#_Toc180103423)

[7. Cas « Demander un crédit » 33](#_Toc180103424)

[8. Cas « valider un crédit » 34](#_Toc180103425)

[9. Cas « Commander une carte bancaire » 34](#_Toc180103426)

[10. Cas « Vendre un produit » 35](#_Toc180103427)

[III. ELABORATION DES DIAGRAMMES DE SEQUENCES ET D’ACTIVITES 36](#_Toc180103428)

[1. Diagrammes de séquence 36](#_Toc180103429)

[2. Diagrammes d’activités 48](#_Toc180103430)

[IV. ELABORATION DU DIAGRAMME DE CLASSE 61](#_Toc180103431)

[PARTIE 3 : MISE EN ŒUVRE DU SYSTEME 63](#_Toc180103432)

[CHAPITRE VI : CONSTRUCTION 64](#_Toc180103433)

[I. ARCHITECTURE LOGICIELLE 64](#_Toc180103434)

[1. Architecture 1-tiers 64](#_Toc180103435)

[2. Architecture 2-tiers 65](#_Toc180103436)

[3. Architecture 3-tiers 65](#_Toc180103437)

[II. SYSTEME DE GESTION DE BASE DE DONNEES 67](#_Toc180103438)

[III. OUTILS D’IMPLEMENTATION 68](#_Toc180103439)

[1. Langages de programmation et Framework 68](#_Toc180103440)

[2. Environnements de développement intégré 71](#_Toc180103441)

[IV. OUTILS D’ORGANISATION ET DE COLLABORATION 72](#_Toc180103442)

[V. RESULTATS DE L’IMPLEMENTATION 73](#_Toc180103443)

[1. Interface de création de compte 73](#_Toc180103444)

[2. Authentification 74](#_Toc180103445)

[3. Tableau de bord administrateurs 74](#_Toc180103446)

[4. Tableau de bord client 75](#_Toc180103447)

[5. Interface virement 76](#_Toc180103448)

[6. Simulateur de crédit 77](#_Toc180103449)

[7. Interface de commande de carte bancaire 78](#_Toc180103450)

[CHAPITRES VII : TRANSITION 79](#_Toc180103451)

[I. TESTS 79](#_Toc180103452)

[1. Tests unitaires 79](#_Toc180103453)

[2. Tests d’intégration/système 80](#_Toc180103454)

[3. Tests fonctionnels 80](#_Toc180103455)

[4. Tests de régression 80](#_Toc180103456)

[5. Tests d’acceptation 81](#_Toc180103457)

[II. DEPLOIEMENT 81](#_Toc180103458)

[1. Sécurité 81](#_Toc180103459)

[2. Hébergement 82](#_Toc180103460)

[3. Nom de domaine 86](#_Toc180103461)

[4. Optimisation 88](#_Toc180103463)

[V. COÛT DU PROJET 91](#_Toc180103465)

[CONCLUSION GENERALE ET PERSPECTIVES 92](#_Toc180103466)

[ANNEXES XI](#_Toc180103467)

[REFERENCES BIBLIOGRAPHIQUES XII](#_Toc180103468)

[REFERENCES WEBOGRAPHIQUES XIII](#_Toc180103469)

[GLOSSAIRE XIV](#_Toc180103470)

[TABLE DES MATIERES XV](#_Toc180103471)