



## DÉDICACE

À,

Ma famille



## REMERCIEMENTS

Nous aimerions témoigner notre profonde reconnaissance envers les personnes suivantes, qui se sont distinguées par leur engagement exceptionnel et leur précieux soutien dans la concrétisation de ce projet :

- Monsieur **Richard N'DRI**, responsable IT de AssurLand, pour sa précieuse assistance et son encadrement avisé durant mon stage. Sa contribution a grandement enrichi mon expérience professionnelle,
- Toute l'équipe AssurLand pour leur accueil chaleureux, leur collaboration exemplaire, et leur soutien constant tout au long de mon stage,
- Monsieur **Louagbeu Loua KPO**, Enseignant Chercheur à l'INP-HB, Directeur des Études ESI, et Chargé du Parcours TS STIC, pour ses précieux conseils,
- Monsieur **Adjé Louis ASSALÉ**, Enseignant Chercheur à l'INP-HB, responsable du département MATHS-INFO pour son assistance et sa disponibilité.



## SOMMAIRE

DÉDICACE .....	I
REMERCIEMENTS .....	II
SOMMAIRE .....	III
AVANT-PROPOS .....	V
LISTE DES SIGLES ET ABRÉVIATIONS .....	VI
LISTE DES TABLEAUX .....	VIII
LISTE DES FIGURES .....	IX
RÉSUMÉ .....	X
INTRODUCTION .....	1
PARTIE I : CADRE ET CONTEXTE DU PROJET .....	2
CHAPITRE I : PRÉSENTATION DU CADRE .....	3
I. PRÉSENTATION DE LA STRUCTURE D'ACCUEIL.....	3
II. DOMAINE D'ACTIVITÉS.....	3
III. ORGANIGRAMME HIÉRARCHIQUE .....	3
CHAPITRE II : DESCRIPTION DU PROJET .....	4
I. PRÉSENTATION DU CONTEXTE .....	4
II. OBJECTIFS DU PROJET .....	4
III. CAHIER DES CHARGES .....	6
IV. ÉTUDE DE L'EXISTANCE .....	8
V. MÉTHODE DE GESTION DE PROJET .....	9
VI. ORGANISATION DU TRAVAIL.....	13
PARTIE II : ÉTUDE CONCEPTUELLE .....	14
CHAPITRE III : APPROCHE MÉTHODOLOGIQUE .....	15
I. INTRODUCTION À LA MÉTHODE D'ANALYSE ET DE CONCEPTION .....	15
II. CHOIX DE LA MÉTHODE D'ANALYSE .....	19
CHAPITRE IV : CRÉATION DE L'ARCHITECTURE DU SYSTÈME.....	20
I. IDENTIFICATION DES ACTEURS ET DES CAS D'UTILISATION .....	20



# Conception et développement d'une API de paiement pour l'amélioration des opérations financières et de l'expérience client



II.	DIAGRAMMES DE SÉQUENCE .....	31
III.	DIAGRAMME DE CLASSES.....	35
PARTIE III : MISE EN OEUVRE.....		36
CHAPITRE V : TECHNIQUE D'ANALYSE .....		37
I.	ARCHITECTURE DU SYSTÈME.....	37
II.	OUTILS DE DÉVELOPPEMENT .....	38
CHAPITRE VI : RÉALISATION .....		44
I.	TESTS DES FONCTIONNALITÉS .....	44
II.	DÉPLOIEMENT .....	47
III.	ESTIMATION BUDGÉTAIRE .....	48
CONCLUSION.....		49
RÉFÉRENCES BIBLIOGRAPHIQUES.....		X
RÉFÉRENCES WEBGRAPHIQUES .....		XI
ANNEXES.....		XIII
TABLE DES MATIÈRES .....		XXXVI



## AVANT-PROPOS

Le décret 96-678 du 04/09/96, ayant donné naissance à l'Institut National Polytechnique Félix Houphouët Boigny (INP-HB) à Yamoussoukro, représente une étape cruciale dans la revitalisation du système éducatif ivoirien. Cette initiative audacieuse résultant de la fusion et de la restructuration de quatre écoles à Yamoussoukro vise à former une jeunesse compétente et à dynamiser les structures existantes.

L'INP-HB, en regroupant ces écoles au sein de six grandes écoles réparties sur trois sites, offre une diversité de formations et de possibilités pour les étudiants. Il prépare également ces futurs professionnels en proposant des projets pratiques dès la seconde année, permettant ainsi la mise en pratique des connaissances académiques et une meilleure intégration dans le monde professionnel.

Cette démarche est essentielle pour le développement de la jeunesse ivoirienne et la prospérité future du pays. L'INP-HB s'inscrit ainsi dans une vision d'excellence académique et de préparation des étudiants à relever les défis de l'avenir.



## LISTE DES SIGLES ET ABRÉVIATIONS

### A

API : Application Programming Interface

### H

HTTP : HyperText Transfer Protocol

HTTPS : HyperText Transfer Protocol Secure

### I

INP-HB : Institut National Polytechnique Félix Houphouët Boigny

IT : Information Technology

### M

MATHS-INFO : Mathématiques et Informatique

MERISE : Méthode d'Étude et de Réalisation Informatique pour les Systèmes d'Entreprise

### O

OAuth : Open Authorization

### P

PHP : Hypertext Preprocessor

### R

RESTful : Representational State Transfer

### S

SPA : Single Page Application

STIC : Sciences et Technologies de l'Information et de la Communication

### T

TLS : Transport Layer Security

TS : Technicien Supérieur



U

UML : Unified Modeling Language



## LISTE DES TABLEAUX

TABEAU 1: DESCRIPTION TEXTUELLE DE "TRAITER UN PAIEMENT" .....	25
TABEAU 2: DESCRIPTION TEXTUELLE DE "CONFIGURER DES OPTIONS DE PAIEMENT" .....	26
TABEAU 3: DESCRIPTION TEXTUELLE DE "GÉRER UN REMBOURSEMENT OU UNE ANNULATION" .....	27
TABEAU 4: DESCRIPTION TEXTUELLE DE "GÉRER LES UTILISATEURS ET LES PERMISSIONS" .....	29
TABEAU 5: COMPARAISON DES ARCHITECTURES .....	38
TABEAU 6: DIFFÉRENTS ENDPOINTS DE L'API .....	44
TABEAU 7: ESTIMATION BUDGÉTAIRE .....	48
TABEAU 8: DESCRIPTION TEXTUELLE DE "EFFECTUER UN PAIEMENT" .....	XVII
TABEAU 9: DESCRIPTION TEXTUELLE DE "CONSULTER L'HISTORIQUE DES PAIEMENTS" .....	XIX
TABEAU 10: DESCRIPTION TEXTUELLE DE "RECEVOIR DES NOTIFICATIONS DE PAIEMENT" .....	XXI
TABEAU 11: DESCRIPTION TEXTUELLE DE "GÉRER LES MOYENS DE PAIEMENT" .....	XXIII
TABEAU 12: DESCRIPTION TEXTUELLE DE "CONTESTER UN PAIEMENT" .....	XXV
TABEAU 13: DESCRIPTION TEXTUELLE DE "INITIER UNE DEMANDE DE PAIEMENT" .....	XXVII
TABEAU 14: DESCRIPTION TEXTUELLE DE "GÉNÉRER DES RAPPORTS FINANCIERS" .....	XXVIII
TABEAU 15: DESCRIPTION TEXTUELLE DE "CONFIGURER LES PARAMÈTRES DE L'API" .....	XXX
TABEAU 16: DESCRIPTION TEXTUELLE DE "AUTORISER UNE TRANSACTION" .....	XXXI
TABEAU 17: DIAGRAMME DE SÉQUENCE "NOTIFICATION" .....	XXXIII
TABEAU 18: DIAGRAMME DE SÉQUENCE "SÉCURITÉ" .....	XXXIV
TABEAU 19: DIAGRAMME DE SÉQUENCE DE "SURVEILLANCE ET DE REPORTING" .....	XXXIV
TABEAU 20: DIAGRAMME DE SÉQUENCE "INTÉGRATION AVEC LES APPLICATIONS TIERCES" .....	XXXV





## LISTE DES FIGURES

FIGURE 1: ORGANIGRAMME HIÉRARCHIQUE DE LA STRUCTURE D'ACCUEIL .....	3
FIGURE 2: ORGANIGRAMME DU TRAVAIL .....	13
FIGURE 3: COMPARAISON MERISE-UML .....	19
FIGURE 4: CAS D'UTILISATIONS .....	23
FIGURE 5: DIAGRAMME DES CAS D'UTILISATION MINIFIÉ .....	24
FIGURE 6: DIAGRAMME DE SÉQUENCE "TRAITEMENT DES TRANSACTIONS" .....	31
FIGURE 7: DIAGRAMME DE SÉQUENCE "AUTHENTIFICATION ET AUTORISATION" .....	32
FIGURE 8: DIAGRAMME DE SÉQUENCE "GESTION DES UTILISATEURS" .....	33
FIGURE 9: DIAGRAMME DE SÉQUENCE "GESTION DES ERREURS" .....	34
FIGURE 10: DIAGRAMME DE CLASSES .....	35
FIGURE 11: LOGO DU FRAMEWORK LARAVEL .....	39
FIGURE 12: LOGO DE LARAVEL PASSPORT .....	39
FIGURE 13: LOGO DE LARAVEL SANCTUM .....	39
FIGURE 14: LOGO DE ANGULARJS .....	40
FIGURE 15: LOGO DE LARAGON .....	40
FIGURE 16: LOGO DE MYSQL .....	41
FIGURE 17: LOGO DE POSTMAN .....	41
FIGURE 18: LOGO DE VISUAL STUDIO CODE .....	41
FIGURE 19: LOGO DE GIT .....	42
FIGURE 20: LOGO DE BITBUCKET .....	42
FIGURE 21: LOGO DE JIRA .....	43
FIGURE 22: CRÉER UN COMPTE UTILISATEUR .....	45
FIGURE 23: AUTHENTIFICATION UTILISATEUR .....	45
FIGURE 24: CONSULTER LE SOLDE .....	46
FIGURE 25: EFFECTUER UN PAIEMENT .....	46
FIGURE 26: VÉRIFIER UN PAIEMENT REÇU .....	46
FIGURE 27: GÉNÉRER DES RAPPORTS FINANCIERS .....	47
FIGURE 28: VOIR LES UTILISATEURS (ADMINISTRATEUR) .....	47



## RÉSUMÉ

Ce mémoire explore la conception et le développement d'une API de paiement visant à améliorer les opérations financières et l'expérience client. Avec l'essor du commerce électronique et des services en ligne, la nécessité d'une infrastructure de paiement sécurisée, rapide et efficace est devenue primordiale pour les entreprises modernes. L'objectif de ce projet est de fournir une solution technique qui permet aux entreprises d'intégrer facilement des fonctionnalités de paiement dans leurs systèmes, tout en assurant une expérience utilisateur optimale et sécurisée.

La première partie de ce mémoire présente une analyse des besoins en matière de paiement en ligne, en mettant l'accent sur les défis actuels auxquels les entreprises et les consommateurs sont confrontés. Cela inclut des aspects tels que la sécurité des transactions, la conformité aux réglementations, la diversité des méthodes de paiement, et la fluidité de l'expérience utilisateur.

Ensuite, la conception de l'API est détaillée, incluant les choix technologiques, les architectures de système, et les spécifications fonctionnelles. L'API proposée sera construite en utilisant des technologies modernes telles que RESTful services, OAuth pour l'authentification et l'autorisation, et des protocoles de sécurité comme HTTPS et TLS.

La phase de développement couvre la mise en œuvre de l'API, avec des exemples de code et des explications sur les modules principaux, tels que la gestion des transactions, l'intégration des passerelles de paiement, et les interfaces utilisateur pour les développeurs. Des tests approfondis sont effectués pour assurer la robustesse et la fiabilité de l'API.

Enfin, ce mémoire discute des résultats obtenus et de l'impact potentiel de l'API sur les opérations financières des entreprises et sur l'expérience client. Des études de cas et des retours d'expérience d'utilisateurs précoces sont présentés pour illustrer les bénéfices pratiques de l'API.

Ce travail conclut en soulignant les perspectives d'amélioration et d'évolution de l'API, en tenant compte des tendances émergentes dans le domaine des paiements électroniques et des retours utilisateurs pour une amélioration continu



## INTRODUCTION

Avec l'évolution constante de la technologie et la transformation numérique qui touche de nombreux secteurs, les entreprises doivent s'adapter rapidement pour répondre aux besoins changeants de leurs clients. Le domaine des paiements en ligne n'échappe pas à cette règle. Dans ce contexte en mutation, notre projet vise à proposer la conception et le développement d'une API de paiement destinée à améliorer les opérations financières et l'expérience client. Cette initiative vise à rendre les transactions en ligne plus sécurisées, plus rapides et plus efficaces pour les utilisateurs. Nous aspirons à créer une solution innovante qui profitera à la fois aux entreprises et aux consommateurs en simplifiant le processus de paiement et en offrant une expérience utilisateur optimale.

Dans ce mémoire, nous explorerons en détail les différentes étapes nécessaires à la réalisation de ce projet, depuis l'identification des besoins jusqu'à la phase de conception et de développement. Nous commencerons par présenter le projet dans son ensemble, puis nous définirons les exigences essentielles pour l'API. Ensuite, nous passerons à la phase conceptuelle, où nous élaborerons les principaux éléments de l'API en utilisant la méthode UML. La phase de développement couvrira la mise en œuvre technique de l'API, principalement avec le framework Laravel. Le mémoire se conclura par une synthèse de nos découvertes, les résultats obtenus, les perspectives d'amélioration de l'API, les références bibliographiques pertinentes et la liste des sources consultées.



# PARTIE I : CADRE ET CONTEXTE DU PROJET

Cette section du rapport comprend la présentation de l'environnement dans lequel le projet a été mené, ainsi qu'une description détaillée du projet.

## CHAPITRE I : PRÉSENTATION DU CADRE

---

Ce chapitre aura pour objectif de présenter le contexte dans lequel notre projet se déroule en mettant en évidence le rôle et les activités de ce cadre.

### I. PRÉSENTATION DE LA STRUCTURE D'ACCUEIL

Fondé en 2004 au Mali, AssurLand est un groupe panafricain spécialisé dans le courtage en assurance. Actuellement, la société étend son expertise dans quatre pays : la Côte d'Ivoire, le Sénégal, le Mali et l'Angola. AssurLand a fait de la satisfaction client sa priorité et s'engage à guider les entreprises, les institutions et les particuliers dans la sélection de la couverture d'assurance la mieux adaptée à leurs besoins spécifiques. Cette approche centrée sur le client reflète la vision de l'entreprise en matière de service et d'accompagnement.

### II. DOMAINE D'ACTIVITÉS

AssurLand opère dans le domaine du courtage en assurance. Leur expertise couvre probablement un large éventail de types d'assurance, notamment l'assurance santé, l'assurance automobile, l'assurance habitation, l'assurance responsabilité civile, l'assurance vie, et d'autres domaines d'assurance pour répondre aux besoins variés de leurs clients dans les pays où ils opèrent.

### III. ORGANIGRAMME HIÉRARCHIQUE

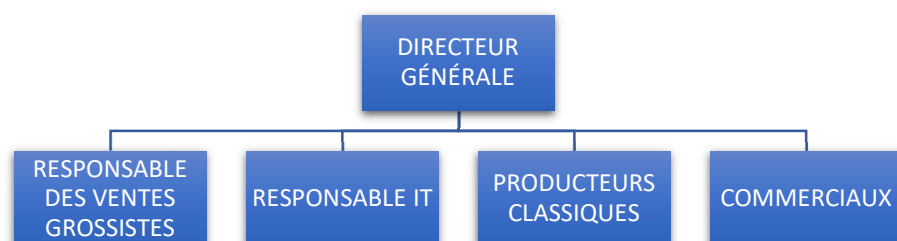


Figure 1: Organigramme hiérarchique de la structure d'accueil



## CHAPITRE II : DESCRIPTION DU PROJET

---

Ce chapitre vise à présenter en détail le projet en question, en définissant clairement ses objectifs ainsi que les spécifications détaillées énoncées dans le cahier des charges.

### I. PRÉSENTATION DU CONTEXTE

Avec l'omniprésence de la technologie qui transforme profondément nos modes de vie et de travail, la nécessité d'automatiser les processus dans le secteur des paiements est devenue incontournable. Ce besoin nous conduit à notre projet phare : « CONCEPTION ET DÉVELOPPEMENT D'UNE API DE PAIEMENT POUR L'AMÉLIORATION DES OPÉRATIONS FINANCIÈRES ET DE L'EXPÉRIENCE CLIENT ». Dans un monde en constante évolution, où les clients recherchent des solutions de paiement plus sécurisées, transparentes et adaptées à leurs besoins, cette initiative vise à révolutionner la manière dont les transactions financières sont effectuées et gérées.

L'essor du commerce électronique et des services en ligne a accentué la demande pour des systèmes de paiement efficaces et fiables. Les entreprises sont de plus en plus confrontées à des défis tels que la sécurité des transactions, la diversité des méthodes de paiement, et la nécessité de fournir une expérience utilisateur fluide et sans friction. Les consommateurs, quant à eux, exigent des solutions de paiement qui soient non seulement rapides et sécurisées, mais aussi faciles à utiliser.

Ce projet s'inscrit dans un cadre plus large d'innovation et d'adaptation aux nouvelles technologies, avec pour objectif de soutenir les entreprises dans leur transformation numérique et de contribuer à l'amélioration continue de l'expérience client.

### II. OBJECTIFS DU PROJET

#### II.1 Objectif général

L'objectif général de ce projet est de concevoir et développer une API de paiement sécurisée et efficace, visant à améliorer les opérations financières des entreprises et à optimiser l'expérience client. En utilisant des technologies modernes et des protocoles de sécurité avancés, cette API simplifiera les processus de paiement en ligne, garantira la sécurité des transactions, et offrira une interface utilisateur intuitive, répondant ainsi aux exigences des entreprises et des consommateurs.

## II.2 Objectifs spécifiques

### ❖ Analyser les besoins des utilisateurs et des entreprises :

- Identifier les exigences fonctionnelles et non fonctionnelles de l'API de paiement.
- Recueillir des retours d'expérience et des attentes des utilisateurs finaux et des entreprises utilisatrices.

### ❖ Concevoir l'architecture de l'API :

- Définir les modules principaux de l'API, tels que la gestion des transactions, l'intégration des passerelles de paiement, et la sécurité.
- Utiliser UML pour modéliser les différents composants et leur interaction.

### ❖ Développer l'API avec le framework Laravel :

- Implémenter les fonctionnalités principales de l'API en suivant les meilleures pratiques de développement.
- Assurer la sécurisation des transactions avec des protocoles tels que HTTPS et TLS.

### ❖ Intégrer des passerelles de paiement multiples :

- Faciliter l'intégration de différentes passerelles de paiement populaires pour offrir une diversité de choix aux utilisateurs.
- Garantir l'interopérabilité et la fluidité des transactions entre différentes passerelles.

### ❖ Mettre en place des mécanismes de sécurité robustes :

- Utiliser OAuth pour l'authentification et l'autorisation des utilisateurs.
- Implémenter des mesures de protection contre les fraudes et les attaques.

### ❖ Tester l'API de manière exhaustive :

- Effectuer des tests unitaires, des tests d'intégration et des tests de performance pour garantir la fiabilité et la robustesse de l'API.
- Corriger les bugs et optimiser les performances en fonction des résultats des tests.

### ❖ Documenter l'API et former les utilisateurs :

- Créer une documentation détaillée pour les développeurs intégrant l'API.



- Organiser des sessions de formation et fournir des supports pédagogiques pour faciliter l'adoption de l'API par les utilisateurs.
- ❖ **Évaluer l'impact de l'API sur les opérations financières et l'expérience client :**
  - Recueillir des retours d'expérience des utilisateurs et des entreprises après la mise en œuvre de l'API.
  - Analyser les améliorations apportées aux processus de paiement et à la satisfaction des utilisateurs.

### III. CAHIER DES CHARGES

Ci-dessous, le cahier des charges détaillant les fonctionnalités de l'application :

#### **Pour les Clients :**

- ❖ **Parcourir une liste complète des méthodes de paiement disponibles**
  - Visualiser toutes les options de paiement acceptées, y compris les descriptions et les conditions d'utilisation.
- ❖ **Effectuer des paiements en ligne sécurisés**
  - Réaliser des transactions financières en utilisant différentes passerelles de paiement de manière sécurisée.
- ❖ **Recevoir des notifications et confirmations de paiement**
  - Obtenir des notifications en temps réel et des confirmations de paiement via email ou SMS.
- ❖ **Accéder à l'historique des transactions**
  - Visualiser et télécharger l'historique des paiements effectués, avec les détails de chaque transaction.
- ❖ **Gérer les informations de paiement**
  - Ajouter, modifier ou supprimer des informations de carte de crédit/débit ou d'autres modes de paiement.
- ❖ **Contacter le support client en ligne en cas de questions ou de problèmes**
  - Accéder à un support client réactif via chat en ligne, email ou téléphone pour toute assistance relative aux paiements.





**Pour l'Administrateur de la Plateforme :**

❖ **Gérer les méthodes de paiement disponibles**

- Ajouter, mettre à jour ou supprimer les différentes options de paiement proposées aux utilisateurs.

❖ **Superviser et gérer les transactions financières**

- Suivre toutes les transactions effectuées, traiter les paiements, et gérer les éventuels remboursements ou litiges.

❖ **Fournir un support en ligne aux clients**

- Assister les clients en cas de problèmes ou de questions relatifs aux paiements, via une interface de support dédiée.

❖ **Générer des rapports sur les performances financières**

- Créer des rapports détaillés sur les transactions, les revenus, les tendances de paiement et d'autres métriques financières.

❖ **Mettre en place des mécanismes de sécurité pour les transactions**

- Assurer la protection des données sensibles et la sécurité des transactions contre les fraudes et les attaques.

❖ **Assurer la conformité réglementaire et légale de l'API**

- Veiller à ce que l'API respecte toutes les normes et réglementations en matière de sécurité des paiements et de protection des données personnelles.

❖ **Suivre et analyser les comportements des utilisateurs**

- Utiliser des outils analytiques pour comprendre les habitudes de paiement des utilisateurs et améliorer l'expérience globale.

## IV. ÉTUDE DE L'EXISTANCE

### IV.1 Description de l'existant

Dans le contexte actuel des opérations financières et de l'expérience client, les transactions et les processus de paiement reposent largement sur des infrastructures traditionnelles. Les paiements sont souvent effectués via des méthodes conventionnelles telles que les virements bancaires, les paiements en espèces ou les transactions par carte de crédit. Les processus de paiement peuvent être gérés en interne par les entreprises elles-mêmes ou externalisés à des prestataires de services de paiement. Les clients peuvent accéder à divers canaux de paiement, notamment les guichets automatiques, les terminaux de paiement dans les magasins physiques et les portails en ligne des banques ou des fournisseurs de services financiers.

### IV.2 Critique de l'existant

#### IV.2.1 Forces du système

- Les infrastructures traditionnelles offrent une certaine familiarité aux clients, ce qui peut renforcer la confiance dans le processus de paiement,
- Les méthodes de paiement conventionnelles peuvent être considérées comme relativement stables et éprouvées.

#### IV.2.2 Faiblesses du système

- Les processus de paiement traditionnels peuvent être sujets à des retards, en particulier dans les transactions internationales, en raison de la complexité des systèmes de règlement interbancaire,
- Les frais de transaction peuvent être élevés, surtout lorsqu'il s'agit de transferts transfrontaliers,
- La sécurité des transactions peut être compromise en raison de la vulnérabilité aux fraudes et aux cyberattaques,
- L'expérience utilisateur peut être entravée par la nécessité de saisir manuellement des informations de paiement et par les temps d'attente lors du traitement des transactions.

#### IV.2.3 Solutions proposées

- Développement d'une API de paiement sécurisée et flexible permettant d'intégrer plusieurs méthodes de paiement et de simplifier le processus de transaction.



- Mise en place de mécanismes de sécurité avancés, tels que l'authentification à deux facteurs et le cryptage des données, pour protéger les transactions contre les menaces potentielles.
- Optimisation des flux de paiement pour réduire les délais de traitement et les frais associés.
- Amélioration de l'expérience utilisateur en proposant des interfaces conviviales et intuitives pour faciliter les paiements en ligne.
- Surveillance continue et mise à jour régulière de l'API pour garantir sa fiabilité et sa conformité aux normes de sécurité et de réglementation.

## **V. MÉTHODE DE GESTION DE PROJET**

### **V.1 La méthode Agile**

#### **V.1.1 Définition**

La méthode Agile est une approche de gestion de projet qui met l'accent sur l'incrémentation, l'itération et la collaboration étroite entre les membres de l'équipe et les parties prenantes. Elle vise à fournir des livrables fonctionnels régulièrement et à s'adapter rapidement aux changements de besoins ou de priorités.

#### **V.1.2 Principes de la méthode Agile**

Les principes fondamentaux de la méthode Agile incluent :

- La satisfaction du client par la livraison continue de logiciels de valeur.
- L'acceptation du changement, même tard dans le développement.
- La livraison fréquente de logiciels fonctionnels (toutes les deux semaines à deux mois).
- La collaboration quotidienne entre les développeurs et les parties prenantes.
- La construction de projets autour d'individus motivés et la confiance dans leur capacité à accomplir les tâches.
- La communication en face à face comme la forme de communication la plus efficace.
- La mesure du progrès principalement par la quantité de logiciels fonctionnels livrés.
- Le développement durable, en maintenant un rythme constant indéfini.
- L'excellence technique et la bonne conception améliorant l'agilité.
- La simplicité, ou l'art de maximiser la quantité de travail non fait, est essentielle.

- Les meilleures architectures, exigences et conceptions émergent d'équipes autoorganisées.
- Des ajustements réguliers pour devenir plus efficaces.

### V.1.3 Présentation de Scrum

Scrum est un cadre de travail Agile qui utilise des itérations courtes et fixes, appelées Sprints, généralement de deux à quatre semaines. Il se concentre sur la maximisation de la capacité de l'équipe à livrer rapidement des produits de valeur. Scrum se compose de rôles définis, d'événements, d'artéfacts et de règles, chacun ayant un objectif spécifique pour gérer le développement et la livraison de produits.

## V.2 Structure de Scrum

### V.2.1 Rôles dans Scrum

#### V.2.1.1 Scrum Master

Le Scrum Master est responsable de promouvoir et de soutenir Scrum. Il aide l'équipe à comprendre et à adopter les valeurs, principes et pratiques de Scrum, élimine les obstacles et facilite les événements Scrum.

#### V.2.1.2 Product Owner

Le Product Owner est responsable de maximiser la valeur du produit résultant du travail de l'équipe de développement. Il gère le Backlog Produit, en s'assurant qu'il est visible, transparent et compréhensible.

#### V.2.1.3 Équipe de développement

L'Équipe de Développement est composée de professionnels qui réalisent le travail de livraison d'un Increment potentiellement livrable à la fin de chaque Sprint. Ils sont auto-organisés et multifonctionnels.

### V.2.2 Évènements Scrum

#### V.2.2.1 Sprint

Un Sprint est une itération de développement de 1 à 4 semaines. Chaque Sprint commence avec une planification et se termine avec une revue et une rétrospective.



#### V.2.2.2 Planification du Sprint

C'est une réunion où l'équipe de Scrum planifie le travail à accomplir pendant le Sprint. Le Product Owner définit les objectifs du Sprint et l'Équipe de Développement sélectionne les éléments du Backlog Produit à compléter.

#### V.2.2.3 Réunion quotidienne

Une courte réunion quotidienne (15 minutes maximum) où l'équipe de développement synchronise ses activités et crée un plan pour les prochaines 24 heures.

#### V.2.2.4 Revue de Sprint

À la fin de chaque Sprint, l'équipe de Scrum et les parties prenantes se réunissent pour inspecter l'incrément et adapter le Backlog Produit si nécessaire.

#### V.2.2.5 Rétrospective de Sprint

Une réunion après la Revue de Sprint où l'équipe de Scrum réfléchit sur le Sprint passé et planifie des améliorations à mettre en œuvre dans le prochain Sprint.

### V.2.3 Artéfacts Scrum

#### V.2.3.1 Backlog Produit

Le Backlog Produit est une liste ordonnée de tout ce qui pourrait être nécessaire dans le produit. C'est la source unique de travail pour l'équipe de développement.

#### V.2.3.2 Backlog de Sprint

Le Backlog de Sprint est l'ensemble des éléments du Backlog Produit sélectionnés pour le Sprint, plus un plan pour délivrer l'incrément et atteindre l'objectif du Sprint.

#### V.2.3.3 Increment

Un Increment est la somme de tous les éléments du Backlog Produit complétés durant un Sprint et les Sprints précédents. Il doit être utilisable et répondre à la définition de "Terminé" (Definition of Done).

## V.2.4 Mise en œuvre de Scrum

### V.2.4.1 Initiation d'un projet Scrum

Définir les objectifs du projet, identifier les parties prenantes, former l'équipe Scrum et créer le premier Backlog Produit.

### V.2.4.2 Planification et Estimation

Utiliser des techniques de planification Agile, telles que la planification poker, pour estimer la taille des éléments du Backlog Produit et planifier les Sprints.

### V.2.4.3 Exécution et contrôle des Sprints

Suivre l'avancement du Sprint avec des outils comme le tableau Scrum et les graphiques de burndown. Assurer une communication constante et résoudre les obstacles.

### V.2.4.4 Suivi et Adaptation

À la fin de chaque Sprint, utiliser les revues et rétrospectives pour évaluer les résultats et adapter le processus et le Backlog Produit en conséquence.

## V.3 Avantages de Scrum

- Réactivité accrue aux changements de besoins,
- Amélioration de la qualité du produit grâce aux itérations fréquentes et aux retours constants,
- Meilleure collaboration et communication au sein de l'équipe et avec les parties prenantes,
- Livraison rapide de fonctionnalités de valeur,
- Transparence accrue sur l'avancement et les obstacles du projet.

Scrum, en tant que cadre Agile, offre une méthode structurée mais flexible pour gérer le développement de produits complexes. En mettant l'accent sur l'adaptation continue, la collaboration étroite et la livraison itérative de valeurs, Scrum aide les équipes à répondre efficacement aux besoins changeants des clients tout en maintenant un haut niveau de qualité.

## VI. ORGANISATION DU TRAVAIL

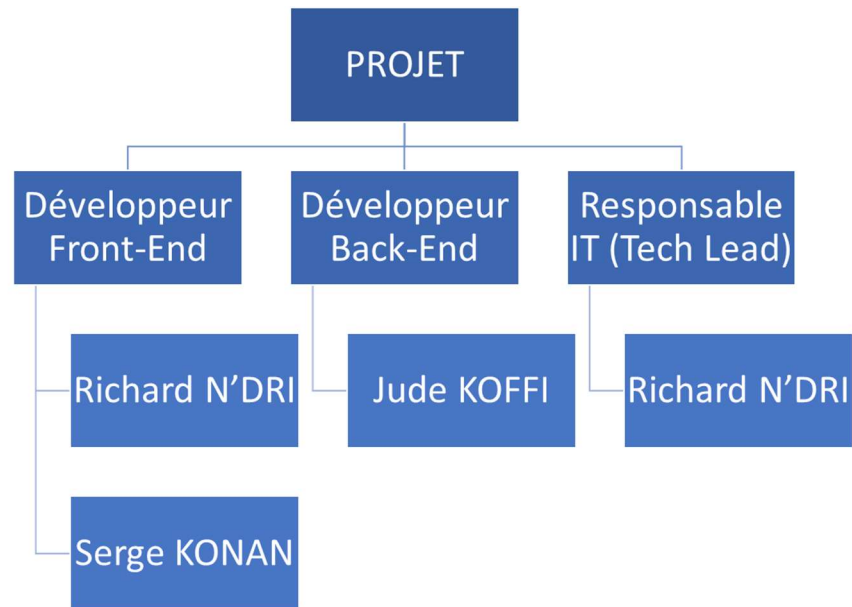


Figure 2: Organigramme du travail



## PARTIE II : ÉTUDE CONCEPTUELLE

Dans cette seconde section, nous aborderons la présentation des différentes méthodes de modélisation que nous avons étudiées, en sélectionnerons une, puis procéderons à la modélisation conceptuelle, organisationnelle et physique pour notre projet.



## CHAPITRE III : APPROCHE MÉTHODOLOGIQUE

---

Nous commencerons par présenter les différentes méthodes d'analyse, puis nous définirons la méthode d'analyse que nous avons choisie d'utiliser.

### I. INTRODUCTION À LA MÉTHODE D'ANALYSE ET DE CONCEPTION

#### I.1 Méthode d'analyse

L'objectif de la méthode d'analyse réside dans la formalisation des étapes initiales de la création d'un système, visant ainsi à répondre de manière appropriée aux besoins du client. Cette démarche implique de partir d'une description informelle des besoins exprimés par le client, complétée par des recherches dans le domaine fonctionnel, ainsi que par une analyse de l'état actuel, à savoir la manière dont les processus gérés par le système sont actuellement mis en œuvre chez le client.

La phase d'analyse consiste à énumérer les résultats attendus en termes de fonctionnalités, de performances, de robustesse, de maintenance, de sécurité, d'extensibilité, et autres. En parallèle, la phase de conception sert à décrire de manière claire, généralement à l'aide d'un langage de modélisation, le fonctionnement futur du système, facilitant ainsi sa mise en œuvre.

#### I.2 La méthode MERISE

##### I.2.1 Présentation

MERISE est une méthodologie de gestion de projets informatiques qui englobe la conception, le développement et la réalisation. Elle se fonde sur le principe de séparation des données et des traitements en différents modèles d'abstraction, ce qui garantit la pérennité du modèle. En général, les données subissent moins de modifications que les traitements. MERISE a vu le jour en 1978-1979 à la suite d'une consultation nationale organisée par le ministère de l'industrie français en 1977, dont l'objectif était de sélectionner les sociétés de conseil informatique chargées de définir la méthode de conception des systèmes d'informations.

##### I.2.2 Les aspects du cycle de développement

Le développement d'un projet selon la méthodologie MERISE repose sur trois éléments fondamentaux :

- La démarche ou cycle de vie,
- Le raisonnement ou cycle d'abstraction,
- La maîtrise ou cycle de décision.

### **I.2.2.1 La démarche ou cycle de vie**

La méthodologie MERISE définit les étapes suivantes pour mener à bien le cycle de vie d'un projet :

1. Le plan stratégique : Il s'agit de la planification globale des projets à réaliser dans l'entreprise à moyen terme. Cela inclut la définition de la qualité et de la quantité des ressources matérielles et humaines nécessaires, ainsi que les priorités à accorder aux différents projets.
2. L'étude préliminaire : Cette phase a pour objectif d'évaluer la faisabilité du projet (pour décider s'il doit être poursuivi ou abandonné) et de déterminer s'il est viable.
3. L'étude approfondie : Une fois la décision d'aller de l'avant prise, cette étape consiste à analyser en détail le système d'information pour mettre en évidence les règles de fonctionnement, telles que les mises à jour, les consultations, les calculs et les règles de gestion.
4. L'étude technique : Cette phase permet de prendre en compte des éléments tels que le choix de la technologie, la nature de la base de données, les options de sécurité, les environnements de développement et les spécifications des équipements nécessaires.
5. La phase de maintenance : Après l'installation, le logiciel est sujet à des erreurs (bugs) et aux évolutions constantes de l'environnement informatique. La maintenance vise à résoudre ces problèmes et à garantir le bon fonctionnement du système.
6. La remise en question : Dans certains cas, les changements dans l'environnement technique peuvent être si significatifs que la simple maintenance ne suffit pas. Il peut alors être nécessaire de revoir le cahier des charges du projet ou de décider de mettre fin définitivement à l'application.

### **I.2.2.2 Le raisonnement ou cycle d'abstraction**

Le design du système d'information est un processus en plusieurs étapes visant à créer un système fonctionnel qui reflète fidèlement la réalité physique. Chaque étape doit être validée en prenant en compte les résultats de la phase précédente. De plus, il est essentiel de vérifier l'alignement entre les données et les traitements en s'assurant que toutes les données nécessaires aux traitements sont disponibles et qu'il n'y a pas de données superflues. Ce processus est connu sous le nom de "cycle d'abstraction" pour la conception des systèmes d'information et se compose de quatre niveaux distincts.

#### I.2.2.2.1 Niveau conceptuel

Il s'agit d'une étape qui concerne des décisions de gestion fondamentales visant à identifier des éléments stables en dehors des moyens, des contraintes et de l'organisation à utiliser. Cette phase répond essentiellement à la question "QUOI ?".

#### I.2.2.2.2 Niveau organisationnel

Cette étape englobe la prise de décisions relatives à l'organisation des ressources humaines et matérielles, en définissant les acteurs impliqués et les postes de travail correspondants. Elle répond aux questions "QUI ? OÙ ? QUAND ?".

#### I.2.2.2.3 Niveau logique

Cette phase implique la prise de décisions concernant les moyens et les ressources informatiques, sans entrer dans les détails techniques spécifiques. Elle se situe au niveau du modèle relationnel, du diagramme des classes et des diagrammes de séquence d'objets. Cette étape répond à la question "COMMENT ?".

#### I.2.2.2.4 Niveau physique

Cette phase représente les décisions techniques prises et leur mise en œuvre en tenant compte de leurs particularités. Elle se situe au niveau du code dans un langage de programmation spécifique, et elle vise à traduire concrètement les choix techniques en réalisations informatiques.

#### I.2.2.3 La maîtrise ou cycle décision

Le cycle de décision englobe l'ensemble des décisions qui doivent être prises tout au long du cycle de vie d'un projet ou d'un système. Il inclut les choix stratégiques, tactiques et opérationnels nécessaires pour mener à bien le projet, résoudre les problèmes éventuels, et garantir l'atteinte des objectifs fixés.

### I.3 PU/UML

#### I.3.1 Processus Unifié (PU)

Le Processus Unifié (PU) est une méthode de développement de logiciels qui adopte une approche itérative et orientée vers l'architecture. Elle repose sur l'utilisation de cas d'utilisation pour atténuer les risques associés au projet. L'objectif principal du PU est de maîtriser la complexité des projets informatiques en minimisant les risques. Cette méthode est composée d'un ensemble de principes génériques qui peuvent être adaptés en fonction des caractéristiques spécifiques de chaque projet.

Le PU est organisé en deux axes principaux : l'axe vertical, qui regroupe les activités en fonction de leur nature et qui décrit l'aspect statique du processus en termes de composants, de processus, d'activités, de séquences et de parties prenantes ; et l'axe horizontal, qui représente le temps et détaille le déroulement du cycle de vie du processus, en mettant en évidence les cycles, les phases et les itérations.

### I.3.2 Unified Modeling Language (UML)

UML, acronyme de "Unified Modeling Language" (Langage de Modélisation Unifié en français), est un langage de modélisation graphique basé sur des pictogrammes. Il offre une méthode normalisée pour représenter la conception d'un système, notamment dans le contexte du développement logiciel et de la conception orientée objet. UML est largement reconnu comme un standard et est géré par l'Object Management Group (OMG).

### I.4 Comparaison des méthodes d'analyse

<b>Aspect</b>	<b>MERISE</b>	<b>UML</b>
<b>Utilisation principale</b>	<b>Conception de bases de données et systèmes d'information.</b>	<b>Conception de logiciels, systèmes d'information et modélisation de processus métier.</b>
<b>Méthode d'analyse</b>	<b>Utilise des schémas conceptuels, logiques et physiques pour représenter la structure des données.</b>	<b>Utilise des diagrammes de classe pour représenter la structure des données.</b>
<b>Diagramme principaux</b>	<b>Diagramme Entité-Relation, Diagramme de Flux, Diagramme de Niveau.</b>	<b>Diagramme de classe, Diagramme de séquence, Diagramme d'activité, etc.</b>
<b>Approche</b>	<b>Structurée, met l'accent sur la modélisation des données et des processus.</b>	<b>Orientée objet, met l'accent sur la modélisation des objets et de leurs interactions.</b>
<b>Formalisme</b>	<b>Moins formel, utilise des notations spécifiques à MERISE</b>	<b>Plus formel, utilise des notations normalisées dans l'industrie</b>

<b>Notations</b>	<b>Propres notations spécifiques à MERISE telles que les rectangles, les losanges, etc.</b>	<b>Utilise des notations normalisées comme les flèches, les stéréotypes, les classes.</b>
<b>Domaines d'application</b>	<b>Principalement orienté vers les systèmes d'information et les bases de données.</b>	<b>Applicable à un large éventail de domaines, y compris les logiciels, l'ingénierie des systèmes, le développement web.</b>

Figure 3: Comparaison MERISE-UML

## II. CHOIX DE LA MÉTHODE D'ANALYSE

Nous avons choisi d'opter pour UML pour la conception de notre projet d'API de paiement en raison de sa reconnaissance en tant que standard industriel dans le domaine du développement logiciel. UML offre une méthodologie de modélisation standardisée, permettant une communication claire et cohérente au sein de l'équipe de développement et avec les parties prenantes. Son abstraction et sa variété de diagrammes, tels que les diagrammes de cas d'utilisation, de classes et de séquence, nous permettent de visualiser efficacement les besoins fonctionnels du système et son architecture. En choisissant UML, nous nous assurons également de bénéficier de la flexibilité et de l'évolutivité nécessaires pour répondre aux changements et aux évolutions du projet au fil du temps.

## CHAPITRE IV : CRÉATION DE L'ARCHITECTURE DU SYSTÈME

---

Dans ce chapitre, nous mettons en œuvre la conception de notre système en appliquant la méthodologie UML.

### I. IDENTIFICATION DES ACTEURS ET DES CAS D'UTILISATION

#### I.1 Les acteurs

Les acteurs sont des éléments essentiels dans la représentation des interactions avec le système. Ils peuvent prendre diverses formes, telles que des individus, des machines externes ou des interactions spécifiques avec le système. On distingue généralement deux catégories : les acteurs principaux, qui interagissent directement avec le système pour son utilisation, et les acteurs secondaires, qui assurent des tâches administratives et de maintenance pour garantir le bon fonctionnement du système pour les acteurs principaux.

Les acteurs principaux et secondaires identifiés sont :

- Acteurs principaux
  1. Utilisateur (Client)
  2. Administrateur (Gestionnaire du système)
  3. Développeur (Responsable de l'intégration)
  4. Banque ou Institution Financière
  5. Fournisseur de Services de Paiement
- Acteurs secondaires
  1. Commerçant (Vendeur)
  2. Système d'API de Paiement
  3. Fournisseur de Services d'Authentification

## I.2 Les cas d'utilisation

CAS D'UTILISATION	DESCRIPTION
<b>CLIENT</b>	
<b>Effectuer un paiement</b>	Permet aux utilisateurs d'envoyer et de recevoir des paiements en ligne.
<b>Consulter l'historique des paiements</b>	Permet aux utilisateurs de consulter les détails de leurs transactions passées.
<b>Recevoir des notifications de paiement</b>	Alerte les utilisateurs sur les paiements reçus ou effectués.
<b>Gérer les moyens de paiement</b>	Permet aux utilisateurs d'ajouter, de supprimer ou de modifier leurs méthodes de paiement.
<b>Contester un paiement</b>	Permet aux utilisateurs de signaler et de contester des transactions incorrectes ou frauduleuses.
<b>COMMERÇANT</b>	
<b>Initier une demande de paiement</b>	Permet aux vendeurs de créer une demande de paiement pour leurs produits ou services à envoyer aux clients.
<b>Vérifier un paiement reçu</b>	Permet aux vendeurs de confirmer la réception de paiements pour leurs ventes.
<b>Gérer un remboursement ou une annulation</b>	Donne aux vendeurs la capacité de traiter les remboursements ou les annulations de transactions.
<b>Générer des rapports financiers</b>	Permet aux vendeurs de créer des rapports détaillés sur leurs activités financières, y compris les ventes, les revenus, etc.
<b>Configurer des options de paiement</b>	Offre aux vendeurs la possibilité de définir et de personnaliser les modes de paiement acceptés pour leurs produits.
<b>Recevoir des notifications de paiement</b>	Alerte les vendeurs sur les paiements reçus pour leurs ventes.
<b>ADMINISTRATEUR SYSTÈME</b>	
<b>Surveiller les performances de l'API</b>	Permet aux administrateurs de suivre et d'analyser les performances et la disponibilité de l'API.

<b>Gérer les utilisateurs et les permissions</b>	Permet aux administrateurs de contrôler l'accès des utilisateurs et de définir leurs autorisations.
<b>Assurer la sécurité et la conformité</b>	Permet aux administrateurs de mettre en place des mesures de sécurité et de veiller à la conformité réglementaire.
<b>Effectuer des mises à jour et des backups</b>	Permet aux administrateurs de mettre à jour le système et de sauvegarder les données.
<b>Gérer les logs et les audits</b>	Permet aux administrateurs de surveillance et d'analyser les journaux d'activité et les audits pour détecter les anomalies.
<b>Configurer les paramètres de l'API</b>	Permet aux administrateurs de définir et d'ajuster les paramètres de l'API selon les besoins.
<b>DÉVELOPPEUR INTÉGRATEUR</b>	
<b>Intégrer l'API dans une application tierce</b>	Implémenter les fonctionnalités de l'API dans une application externe de manière fluide.
<b>Tester les fonctionnalités de l'API</b>	Évaluez le bon fonctionnement de l'API en effectuant des tests pour garantir sa performance et sa fiabilité.
<b>Assurer la compatibilité avec d'autres systèmes</b>	Veille à ce que l'API puisse interagir efficacement avec d'autres systèmes et applications.
<b>Fournir de la documentation et des exemples de code</b>	Crée et partage une documentation détaillée ainsi que des exemples de code pour faciliter l'intégration et l'utilisation de l'API.
<b>Déboguer et résoudre les problèmes techniques</b>	Identifier et corriger les erreurs et les problèmes techniques rencontrés lors de l'utilisation de l'API.
<b>BANQUE OU INSTITUTION FINANCIÈRE</b>	
<b>Autoriser une transaction</b>	Approuve ou rejette les transactions financières en fonction des règles et des paramètres définis par l'institution.
<b>Traiter un paiement</b>	Effectuer les étapes nécessaires pour transférer des fonds entre les comptes des clients ou vers des tiers.
<b>Gérer les litiges et les rétro facturations</b>	Gère les situations où les clients contestent une transaction ou demandent un remboursement.



<b>Garantir la conformité réglementaire</b>	Assure le respect des lois et des réglementations en vigueur dans le secteur financier.
<b>Fournir des relevés de transaction</b>	Génère et mis à disposition des clients des relevés détaillés de leurs transactions financières.
<b>FOURNISSEUR DE SERVICES DE PAIEMENT</b>	
<b>Intégrer les différentes méthodes de paiement</b>	Permet aux commerçants d'accepter une variété de méthodes de paiement.
<b>Faciliter les transactions entre les banques et les commerçants</b>	Assurer la liaison entre les institutions financières et les commerçants pour faciliter le traitement des paiements.
<b>Gérer les risques et la fraude</b>	Met en place des systèmes et des procédures pour détecter et prévenir les activités frauduleuses.
<b>Proposer des solutions de paiement personnalisés</b>	Offre des options de paiement adaptées aux besoins spécifiques des commerçants et de leurs clients.

Figure 4: Cas d'utilisations

## I.3 Diagramme des cas d'utilisation

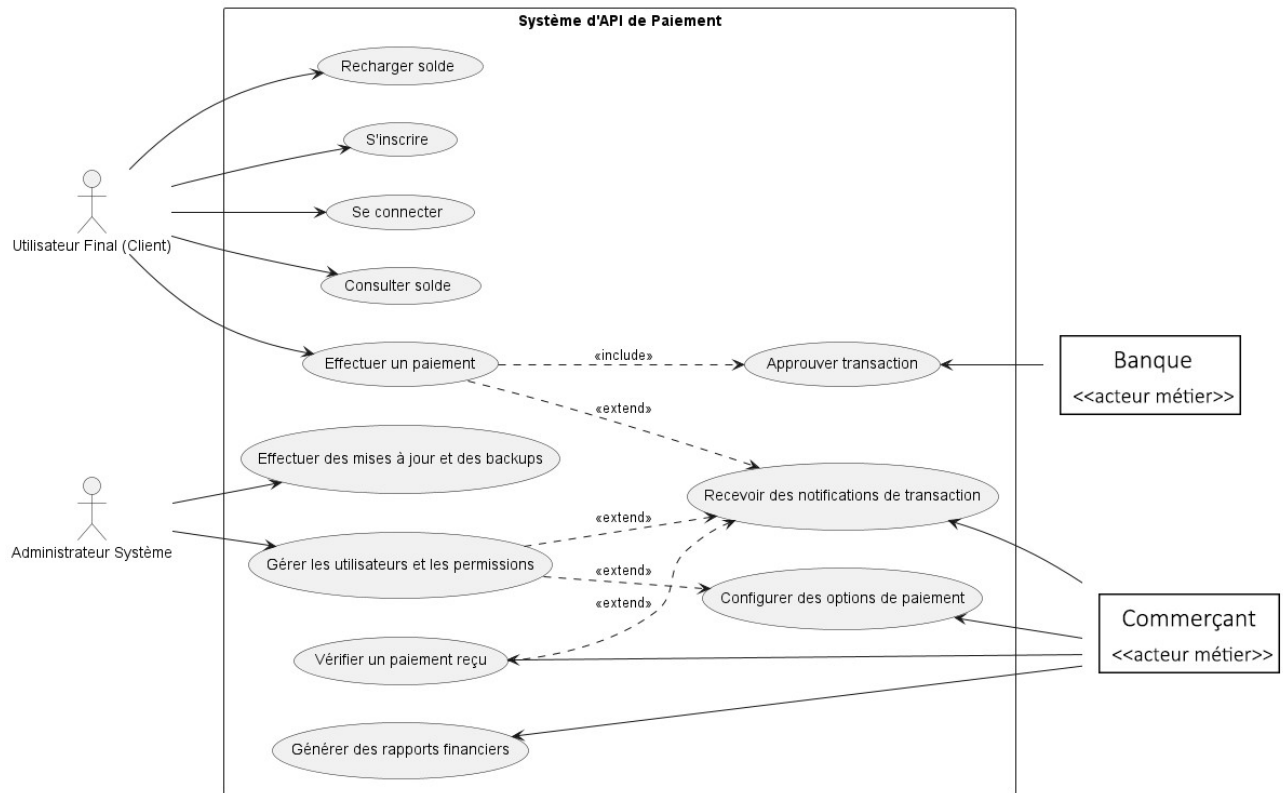


Figure 5: Diagramme des cas d'utilisation minifié

## I.4 Descriptions textuelles des cas d'utilisation

### I.4.1 Traiter un paiement

Tableau 1: Description textuelle de "Traiter un paiement"

<b>Nom du Cas d'Utilisation</b>	<b>Traiter un paiement</b>
<b>Acteur Principal</b>	Système de Gestion des Transactions
<b>Acteurs Secondaires</b>	Utilisateur Final (Client), API de Paiement, Système de Gestion des Utilisateurs
<b>Objectif</b>	Permettre au système de gestion des transactions de traiter avec succès un paiement autorisé, en transférant les fonds du compte de l'utilisateur final vers le compte du commerçant et en enregistrant toutes les informations pertinentes associées à la transaction.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>1. La transaction a été autorisée avec succès par le système de gestion des transactions.</li> <li>2. Les informations de paiement et les détails de la transaction ont été validés et sont prêts à être traités.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>1. Le système de gestion des transactions récupère les informations de la transaction autorisée, y compris le montant, les détails du commerçant, les informations de paiement, etc.</li> <li>2. Le système transfère les fonds du compte de l'utilisateur final vers le compte du commerçant en utilisant les informations de paiement validées lors de l'autorisation de la transaction.</li> <li>3. Le système enregistre toutes les informations pertinentes associées à la transaction, telles que le montant, la date, le commerçant, le numéro de transaction, etc.</li> <li>4. Le système génère un reçu de transaction contenant les détails de la transaction, y compris le montant, la date, le commerçant, etc.</li> <li>5. Le reçu est envoyé à l'utilisateur final et/ou au commerçant pour confirmation du paiement et comme preuve de la transaction réussie.</li> </ol>
<b>Scénarios Alternatifs</b>	<b>Échec du Transfert de Fonds :</b>

	<ul style="list-style-type: none"> <li>Si le transfert de fonds échoue pour une raison quelconque, telle qu'un problème technique, une indisponibilité du service de paiement, etc., le système doit annuler la transaction, avertir les parties concernées et enquêter sur la cause du problème.</li> </ul>
<b>Postconditions</b>	Le paiement est traité avec succès, les fonds sont transférés du compte de l'utilisateur final vers le compte du commerçant, et toutes les informations de transaction sont enregistrées de manière sécurisée dans le système.

#### I.4.2 Configurer des options de paiement

Tableau 2: Description textuelle de "Configurer des options de paiement"

<b>Nom du Cas d'Utilisation</b>	<b>Configurer des options de paiement</b>
<b>Acteur Principal</b>	Administrateur du Système
<b>Objectif</b>	Permettre à l'administrateur du système de configurer et de gérer les différentes options de paiement disponibles pour les utilisateurs finaux, afin de répondre aux besoins spécifiques de l'entreprise ou de l'organisation.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>L'administrateur du système doit être authentifié et avoir les autorisations nécessaires pour accéder aux options de configuration de paiement.</li> <li>Les différentes méthodes de paiement doivent être intégrées au système et disponibles pour la configuration.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>L'administrateur du système accède à la section "Configurer des options de paiement" via l'interface d'administration du système.</li> <li>L'administrateur du système examine les méthodes de paiement actuellement disponibles et décide des modifications à apporter.</li> <li>Pour chaque méthode de paiement activée, l'administrateur du système peut spécifier des paramètres et des configurations supplémentaires.</li> </ol>

	<ol style="list-style-type: none"> <li>4. L'administrateur du système passe en revue les modifications apportées aux options de paiement pour s'assurer de leur exactitude.</li> <li>5. Une fois validées, les modifications sont enregistrées dans le système et deviennent effectives pour les utilisateurs finaux.</li> </ol>
<b>Scénarios Alternatifs</b>	<p><b>Échec de Validation :</b></p> <ul style="list-style-type: none"> <li>• Si une erreur est détectée lors de la validation des modifications, le système affiche un message d'erreur et invite l'administrateur à corriger les paramètres invalides avant d'enregistrer les modifications.</li> </ul>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Les options de paiement sont configurées et mises à jour selon les préférences spécifiées par l'administrateur du système.</li> <li>2. Les modifications apportées aux options de paiement sont enregistrées dans le système et prêtes à être utilisées par les utilisateurs finaux lors des transactions.</li> </ol>

#### I.4.3 Gérer un remboursement ou une annulation

Tableau 3: Description textuelle de "Gérer un remboursement ou une annulation"

<b>Nom du Cas d'Utilisation</b>	<b>Gérer un remboursement ou une annulation</b>
<b>Acteur Principal</b>	Utilisateur Final (Client)
<b>Acteurs Secondaires</b>	<ol style="list-style-type: none"> <li>1. Commerçant (Vendeur)</li> <li>2. Service Client</li> <li>3. Banque ou Institution Financière</li> </ol>
<b>Objectif</b>	Permettre à l'utilisateur final d'initier et de gérer des demandes de remboursement ou d'annulation pour des transactions de paiement existantes, afin de récupérer les fonds ou d'annuler les paiements en cas de besoin.

<b>Préconditions</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur final doit être authentifié et avoir accès à son compte.</li> <li>2. La transaction de paiement à rembourser ou à annuler doit être enregistrée dans le système avec les détails pertinents.</li> <li>3. Le système doit fournir une interface utilisateur pour initier et suivre les demandes de remboursement ou d'annulation.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur final accède à la section "Gérer un remboursement ou une annulation" via le portail utilisateur ou l'application mobile.</li> <li>2. Le système affiche une liste des transactions récentes, y compris les paiements effectués.</li> <li>3. L'utilisateur final sélectionne la transaction spécifique à rembourser ou à annuler dans la liste des transactions récentes.</li> <li>4. L'utilisateur final choisit l'option pour initier une demande de remboursement ou d'annulation pour la transaction sélectionnée.</li> <li>5. L'utilisateur final fournit une raison détaillée pour la demande de remboursement ou d'annulation.</li> <li>6. L'utilisateur final passe en revue les détails de la demande de remboursement ou d'annulation pour s'assurer de leur exactitude.</li> <li>7. L'utilisateur final confirme la demande et la soumet au système.</li> </ol>
<b>Scénarios Alternatifs</b>	<p><b>Transaction Non Trouvée :</b></p> <ul style="list-style-type: none"> <li>• Si la transaction spécifiée n'est pas trouvée dans la liste des transactions récentes, le système affiche un message d'erreur et invite l'utilisateur à vérifier à nouveau ou à contacter le service client.</li> </ul> <p><b>Échec de Soumission :</b></p> <ul style="list-style-type: none"> <li>• Si l'envoi de la demande de remboursement ou d'annulation échoue en raison d'un problème technique ou de connectivité, le</li> </ul>

	système affiche un message d'erreur et invite l'utilisateur à réessayer ultérieurement.
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. La demande de remboursement ou d'annulation est initiée et transmise au service client, au commerçant ou à la banque pour traitement.</li> <li>2. Le statut de la transaction est mis à jour pour refléter la demande de remboursement ou d'annulation.</li> </ol>

#### I.4.4 Gérer les utilisateurs et les permissions

Tableau 4: Description textuelle de "Gérer les utilisateurs et les permissions"

<b>Nom du Cas d'Utilisation</b>	<b>Gérer les utilisateurs et les permissions</b>
<b>Acteur Principal</b>	Administrateur du Système
<b>Objectif</b>	Permettre à l'administrateur du système de gérer les utilisateurs et leurs permissions associées dans le système de paiement, afin de garantir un accès sécurisé et approprié aux différentes fonctionnalités et données.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>1. L'administrateur du système doit être authentifié et avoir les autorisations nécessaires pour accéder aux fonctionnalités de gestion des utilisateurs et des permissions.</li> <li>2. Les utilisateurs doivent être enregistrés dans le système avec des identifiants uniques et des rôles attribués.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur du système accède à la section "Gérer les utilisateurs" via l'interface d'administration du système.</li> <li>2. Le système affiche une liste des utilisateurs enregistrés dans le système, y compris leurs identifiants, leurs rôles et leurs permissions actuelles.</li> <li>3. L'administrateur sélectionne un utilisateur spécifique dans la liste et choisit l'option pour modifier ses permissions.</li> <li>4. L'administrateur passe en revue les modifications apportées aux permissions de l'utilisateur pour s'assurer de leur exactitude.</li> </ol>



	5. Une fois validées, les modifications sont enregistrées dans le système et deviennent effectives immédiatement.
<b>Scénarios Alternatifs</b>	<b>Échec de Validation :</b> <ul style="list-style-type: none"><li>• Si une erreur est détectée lors de la validation des modifications, le système affiche un message d'erreur et invite l'administrateur à corriger les paramètres invalides avant d'enregistrer les modifications.</li></ul>
<b>Postconditions</b>	<ol style="list-style-type: none"><li>1. Les modifications apportées aux utilisateurs et à leurs permissions sont enregistrées dans le système et deviennent effectives immédiatement.</li><li>2. Les utilisateurs peuvent accéder aux fonctionnalités autorisées en fonction de leurs permissions.</li></ol>



## II. Diagrammes de séquence

### II.1 Traitement des transactions

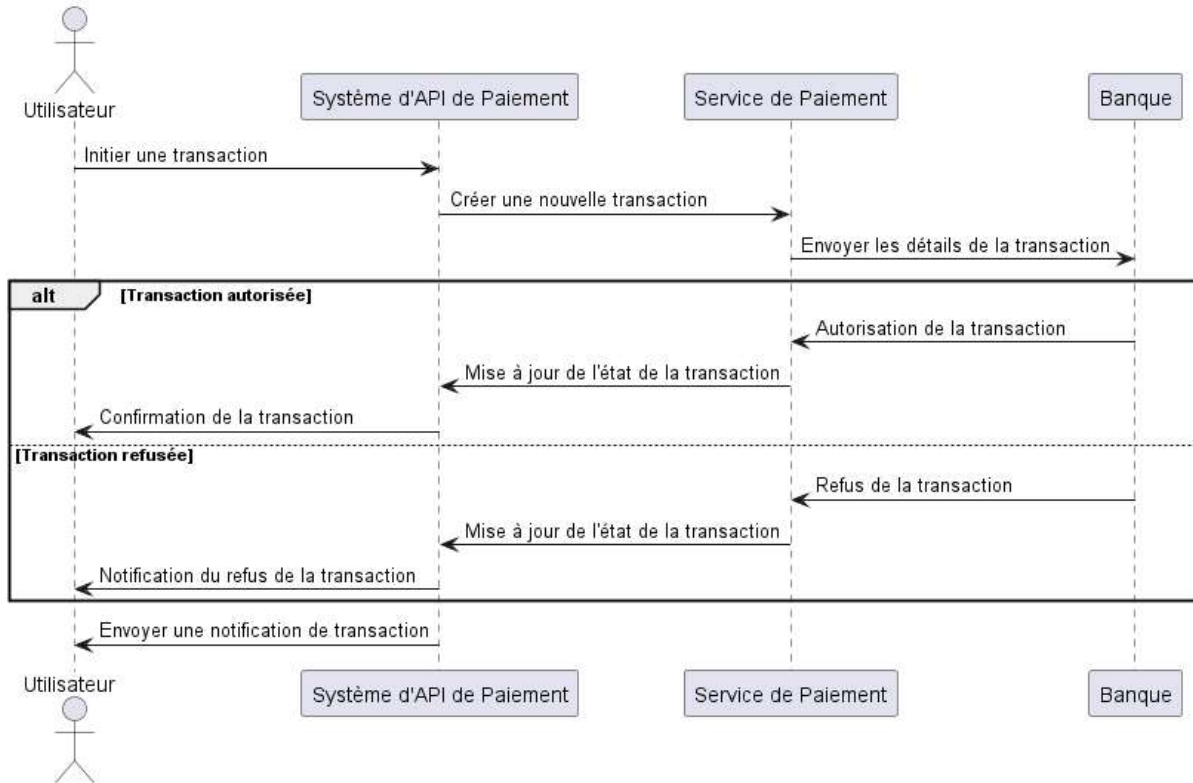


Figure 6: Diagramme de séquence "Traitement des transactions"

## II.2 Authentification et autorisation

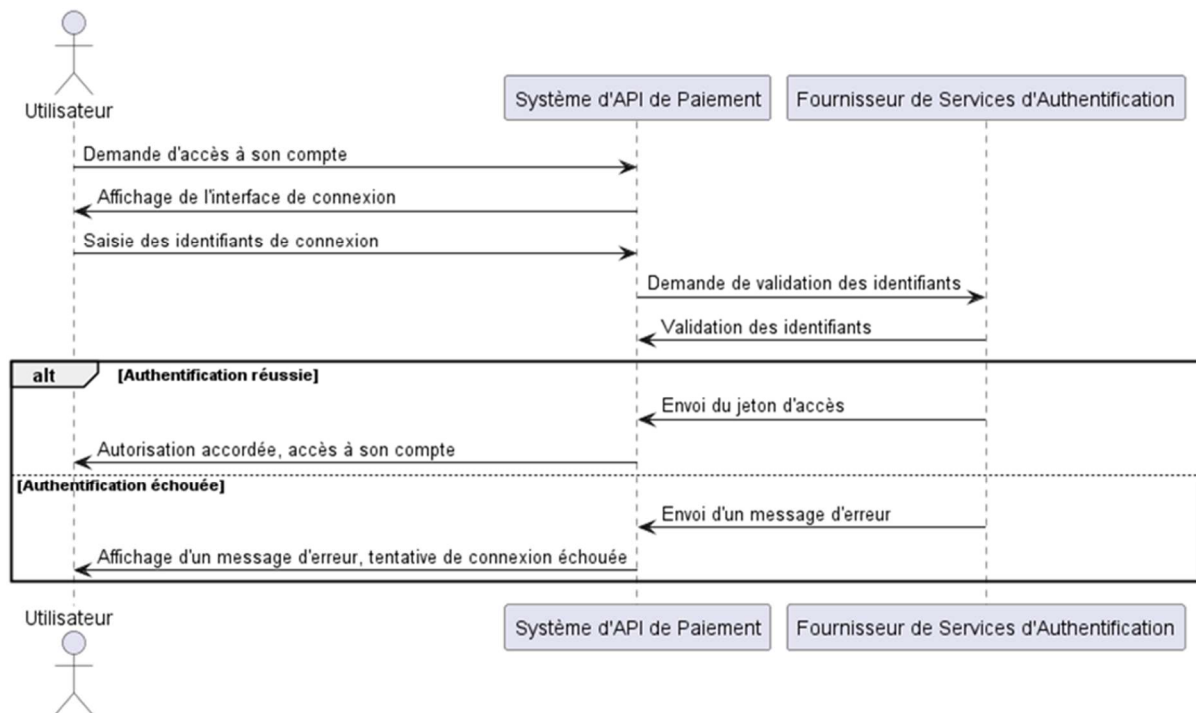


Figure 7: Diagramme de séquence "Authentification et autorisation"

### II.3 Gestion des utilisateurs

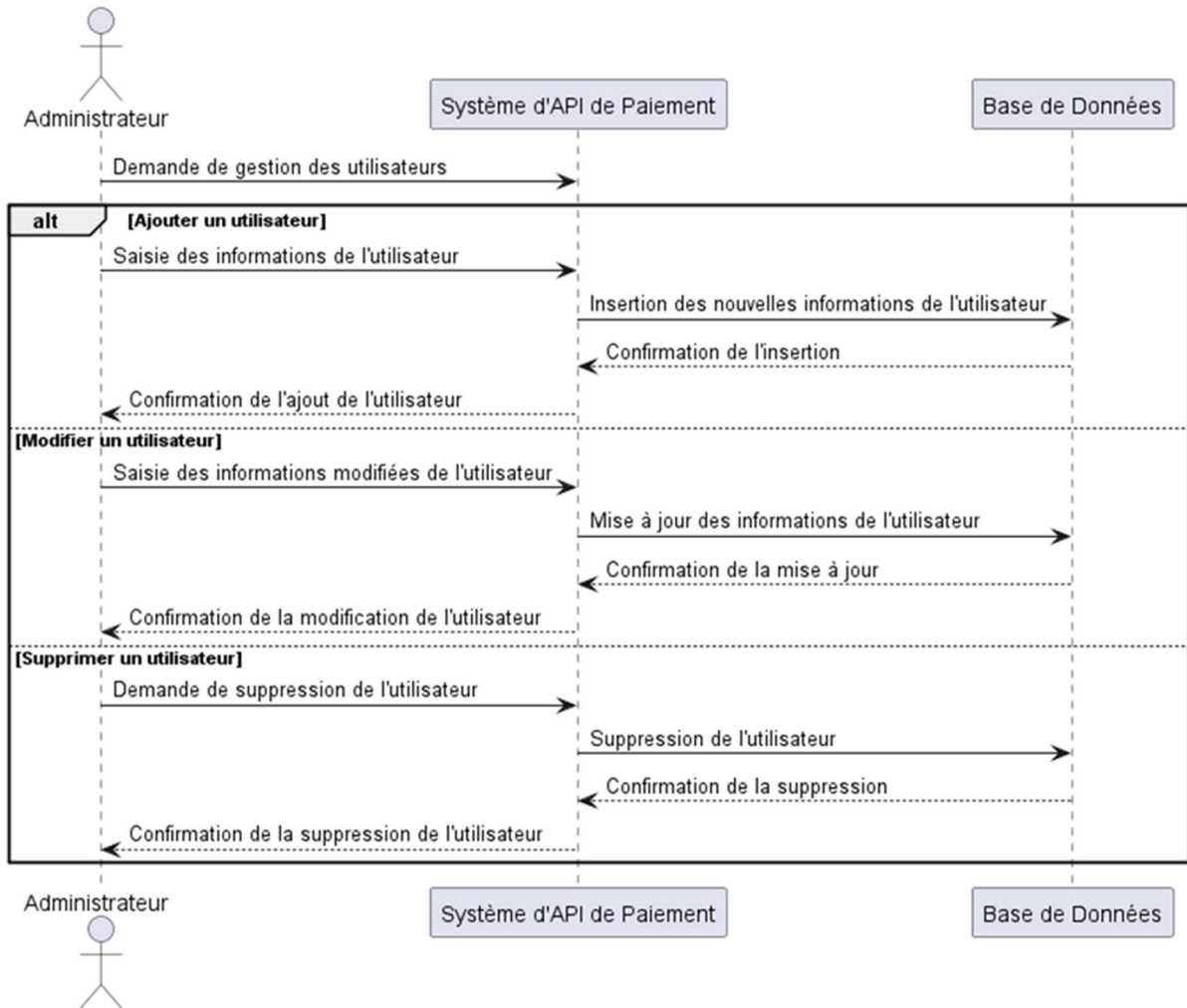


Figure 8: Diagramme de séquence "Gestion des utilisateurs"

## II.4 Gestion des erreurs

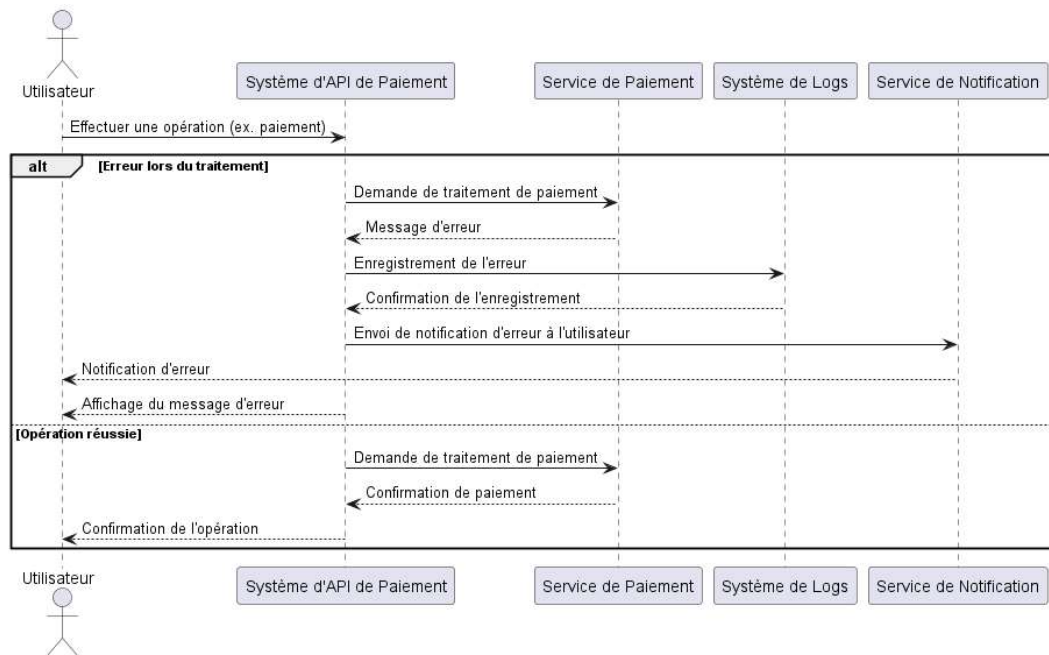


Figure 9: Diagramme de séquence "Gestion des erreurs"

## III. Diagramme de classes

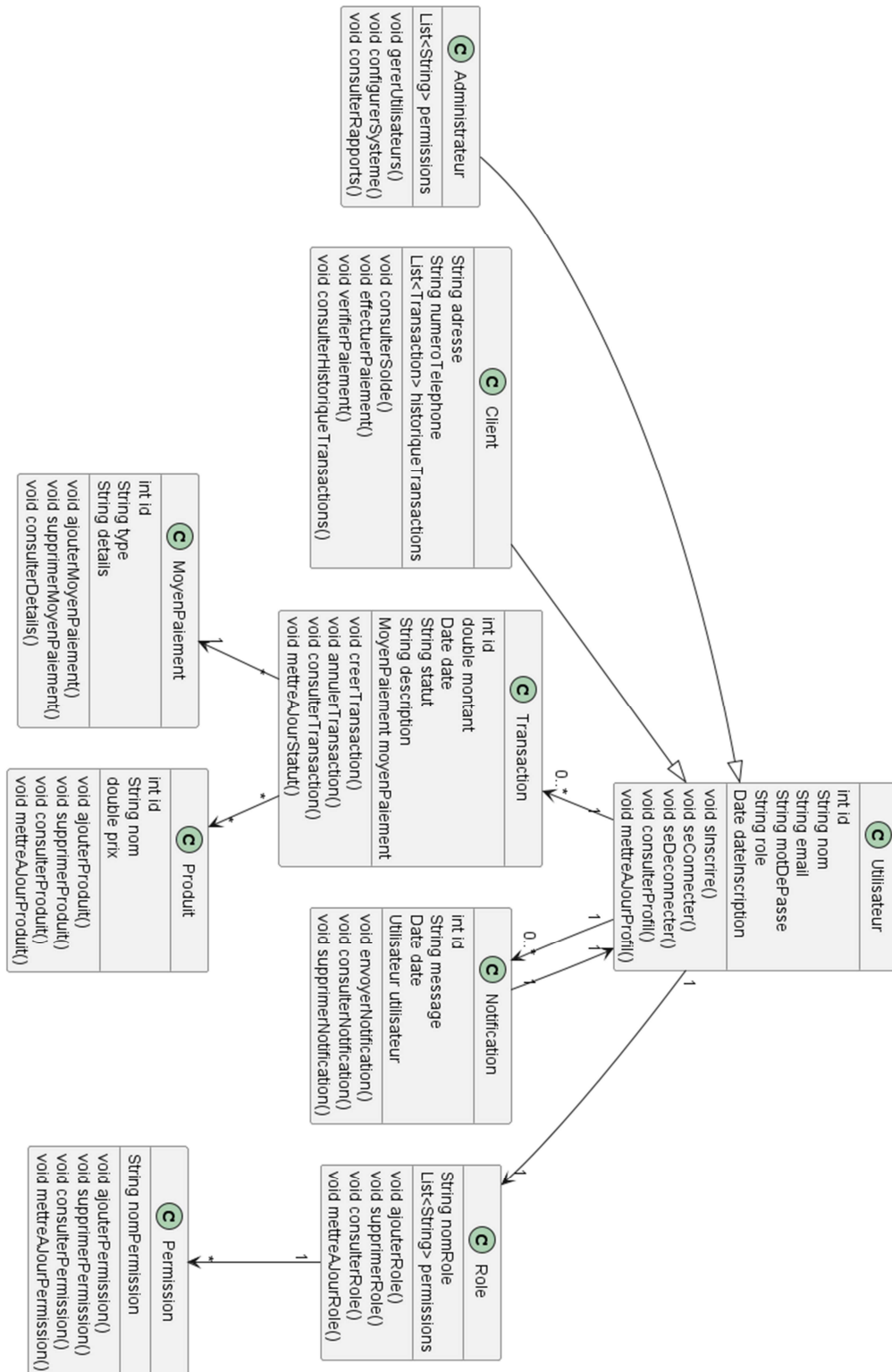


Figure 10: Diagramme de classes



## PARTIE III : MISE EN OEUVRE

Dans cette section, nous présenterons les éléments et les ressources qui ont été utilisés pour accomplir cette tâche, ainsi que pour réaliser notre système avec succès.

## CHAPITRE V : TECHNIQUE D'ANALYSE

---

Dans cette section, nous allons détailler les composants et les ressources qui ont été mobilisés pour accomplir cette tâche et pour concrétiser notre système.

### I. ARCHITECTURE DU SYSTÈME

La conception d'applications repose généralement sur une structure en trois couches bien définies : la couche de données, la couche de traitement et la couche de présentation.

#### I.1 Présentation des couches

##### I.1.1 La couche de données

La couche de données englobe à la fois le stockage des données et les mécanismes qui permettent à l'application d'accéder et de manipuler ces données de manière à les rendre exploitables dans la phase de traitement.

##### I.1.2 La couche de traitement

La couche de traitement englobe à la fois les tâches que l'application doit exécuter sur les données et les opérations nécessaires pour répondre aux actions entreprises par l'utilisateur. Elle représente le cœur de la logique de l'application où les données sont traitées en fonction des besoins et des interactions de l'utilisateur.

##### I.1.3 La couche de présentation

La couche de présentation est responsable de la mise en œuvre de la logique de présentation de l'application et facilite l'interaction entre l'application et l'utilisateur. Elle constitue la partie visible et conviviale du système, servant ainsi d'interface utilisateur.

Ces trois niveaux, à savoir la couche de données, la couche de traitement et la couche de présentation, peuvent être organisés de différentes manières, ce qui permet de distinguer plusieurs architectures applicatives, notamment l'architecture à un seul niveau, l'architecture à deux niveaux et l'architecture à trois niveaux.

#### I.2 Comparaison des différentes architectures

Pour choisir l'architecture la mieux adaptée à notre système, nous avons effectué une analyse comparative des différentes architectures disponibles. Vous trouverez ci-dessous un tableau qui résume cette comparaison entre ces différentes architectures.

Tableau 5: Comparaison des architectures

Architecture	Description	Avantages	Inconvénients
Architecture à un seul niveau	Toutes les fonctionnalités sont regroupées en une seule couche.	Simplicité adaptée aux petites applications.	Manque de modularité, évolutivité limitée.
Architecture à deux niveaux	Division en deux couches principales : données et présentation.	Amélioration de la modularité.	Traitement inclus dans la présentation.
Architecture à trois niveaux	Séparation distincte des données, du traitement et de la présentation.	Modularité, évolutivité, maintenance aisée.	Plus de complexité, nécessite une planification détaillée.

## I.3 Architecture retenue

L'architecture trois tiers s'avère idéale pour le développement d'une API en raison de sa capacité à séparer les préoccupations en trois couches distinctes : la présentation, la logique métier et la persistance des données. Cette séparation permet une gestion modulaire du code, favorisant ainsi la maintenance, la mise à jour et l'extension du système sans perturber les autres parties. De plus, cette structure optimise les performances en répartissant efficacement la charge de travail, tout en facilitant la détection et la résolution des problèmes. En isolant les différentes couches, l'architecture trois tiers simplifie également la mise à l'échelle du système en permettant d'ajouter des ressources là où elles sont nécessaires, contribuant ainsi à une API robuste, flexible et facile à maintenir.

## II. OUTILS DE DÉVELOPPEMENT

### II.1 Laravel

Laravel est un framework PHP open-source, réputé pour sa simplicité et son élégance. Il offre une syntaxe expressive et des fonctionnalités puissantes pour le développement d'applications web robustes et évolutives.



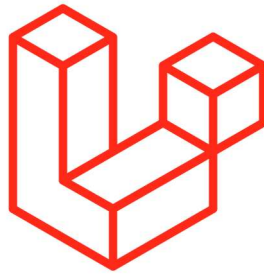


Figure 11: Logo du  
framework Laravel

## II.2 Laravel Passport

Laravel Passport est un package d'authentification API pour le framework Laravel, conçu pour simplifier l'implémentation de l'authentification OAuth2. Il offre une solution complète pour gérer les jetons d'accès, les clients OAuth, et les utilisateurs, tout en fournissant une sécurité robuste et des outils pour une gestion aisée.



Figure 12: Logo de Laravel Passport

## II.3 Laravel Sanctum

Laravel Sanctum est un package qui fournit une API de token d'authentification pour les applications web SPA et les API mobiles. Il permet une authentification stateless, basée sur des jetons, offrant ainsi une solution sécurisée et flexible pour protéger les routes de l'application.



Figure 13: Logo de Laravel Sanctum

## II.4 AngularJS

AngularJS est un framework JavaScript open-source développé par Google. Il est largement utilisé pour la création d'applications web dynamiques et interactives côté client. AngularJS offre une structure robuste pour le développement d'applications web basées sur des composants.



Figure 14: Logo de AngularJS

## II.5 Laragon

Laragon est une plateforme de développement local pour les développeurs web, offrant un environnement complet incluant Apache, MySQL, PHP et autres outils essentiels. Il propose des fonctionnalités avancées telles que la création de domaines virtuels et la gestion des bases de données, en plus de prendre en charge des outils tiers comme Composer, Git et Node.js.



Figure 15: Logo de Laragon

## II.6 MySQL

MySQL est un système de gestion de base de données relationnelle open-source. Il est largement utilisé pour stocker et gérer les données dans les applications web. MySQL offre une grande fiabilité, une performance élevée et une grande compatibilité avec de nombreuses plateformes.



Figure 16: Logo de  
MySQL

## II.7 Postman

Postman est une plateforme de développement d'API qui permet aux développeurs de créer, de tester et de déboguer des API plus rapidement et plus facilement. Il offre une interface conviviale pour envoyer des requêtes HTTP, inspecter les réponses et automatiser les tests d'API.



Figure 17: Logo de  
Postman

## II.8 Visual studio code

Visual Studio Code est un éditeur de code source léger et puissant, développé par Microsoft. Il offre une large gamme de fonctionnalités, telles que la coloration syntaxique, la complétion



Figure 18: Logo de  
Visual Studio Code

automatique, le débogage intégré, et une vaste sélection d'extensions pour améliorer la productivité des développeurs.

## II.9 Git

Git est un système de contrôle de version distribué utilisé pour suivre les modifications du code source pendant le développement logiciel. Il permet aux développeurs de collaborer efficacement sur des projets, de gérer les versions du code et de revenir en arrière en cas de besoin.



Figure 19: Logo de Git

## II.10 Bitbucket

Bitbucket est une plateforme de gestion de code source basée sur Git, développée par Atlassian. Il offre des fonctionnalités de gestion de projet, de suivi des problèmes, et de collaboration en équipe, en plus de l'hébergement de dépôts Git privés et publics.



Figure 20: Logo de Bitbucket

## II.11 Jira

Jira est un logiciel de suivi de problèmes et de gestion de projet, également développé par Atlassian. Il permet aux équipes de planifier, de suivre et de gérer les tâches et les projets de manière efficace, en offrant une visibilité et un contrôle complets sur le cycle de vie du développement logiciel.



Figure 21: Logo de Jira

## CHAPITRE VI : RÉALISATION

Dans cette section, nous allons introduire les tests de l'API.

### I. TESTS DES FONCTIONNALITÉS

#### I.1 Définition des Endpoints

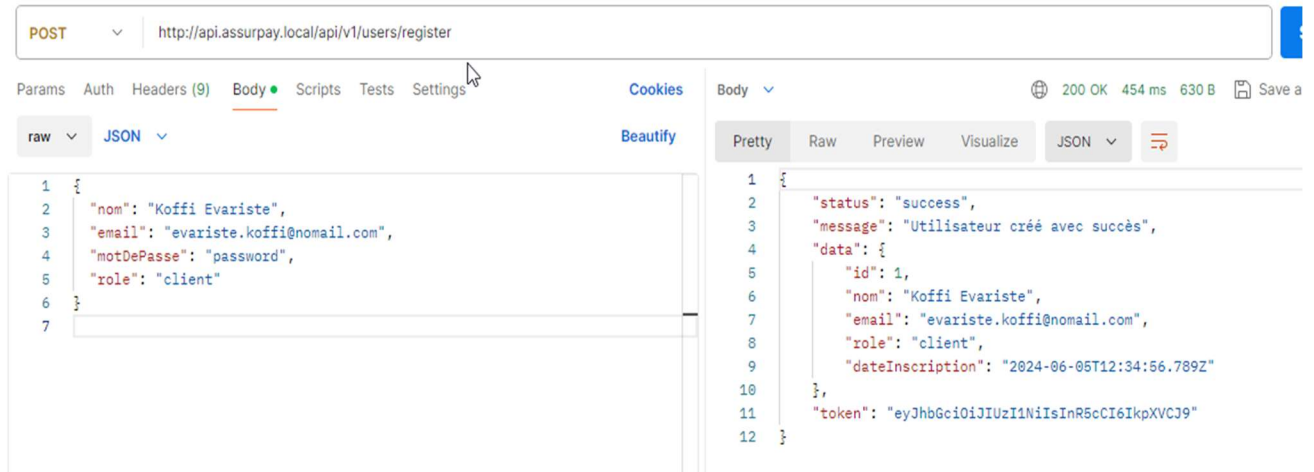
Un endpoint est une URL spécifique où une application peut accéder aux ressources fournies par le serveur de l'API. Chaque endpoint correspond à une fonction particulière de l'API, permettant aux clients (comme les applications web ou mobiles) d'interagir avec le serveur de l'API pour réaliser différentes actions, comme récupérer des données, les mettre à jour, les supprimer ou en ajouter de nouvelles.

Tableau 6: Différents endpoints de l'API

Action	Méthode	URL	Description
<b>Créer un compte utilisateur</b>	POST	/api/v1/users/register	Permet à un nouvel utilisateur de créer un compte en fournissant les informations nécessaires.
<b>Authentification utilisateur</b>	POST	/api/v1/users/login	Authentifie un utilisateur en vérifiant ses identifiants et retourne un token d'accès.
<b>Consulter le solde</b>	GET	/api/v1/users/{userId}/balance	Permet à un utilisateur de consulter son solde actuel.
<b>Effectuer un paiement</b>	POST	/api/v1/payments	Effectue un paiement à partir du compte de l'utilisateur vers un autre compte.
<b>Vérifier un paiement reçu</b>	GET	/api/v1/payments/{paymentId}	Vérifie le statut d'un paiement reçu par un utilisateur.
<b>Générer des rapports financiers</b>	GET	/api/v1/reports/financial	Génère un rapport financier détaillé pour un utilisateur ou un commerçant.
<b>Gérer les utilisateurs et les</b>	GET, POST,	/api/v1/admin/users	Permet à l'administrateur de

permissions (Administrateur)	PUT, DELETE	gérer les utilisateurs et les permissions.
---------------------------------	----------------	---

## I.2 Tests des endpoints



The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://api.assurpay.local/api/v1/users/register`
- Body (Request):**

```

1 {
2   "nom": "Koffi Evariste",
3   "email": "evariste.koffi@nomail.com",
4   "motDePasse": "password",
5   "role": "client"
6 }
7

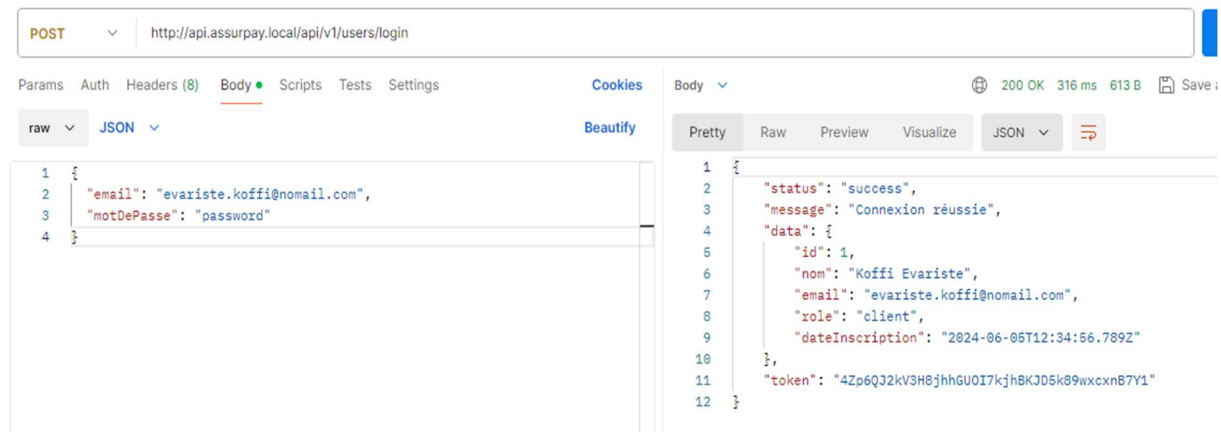
```
- Body (Response):**

```

1 {
2   "status": "success",
3   "message": "Utilisateur créé avec succès",
4   "data": {
5     "id": 1,
6     "nom": "Koffi Evariste",
7     "email": "evariste.koffi@nomail.com",
8     "role": "client",
9     "dateInscription": "2024-06-05T12:34:56.789Z"
10  },
11   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9"
12 }

```
- Status:** 200 OK, 454 ms, 630 B

Figure 22: Créer un compte utilisateur



The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `http://api.assurpay.local/api/v1/users/login`
- Body (Request):**

```

1 {
2   "email": "evariste.koffi@nomail.com",
3   "motDePasse": "password"
4 }

```
- Body (Response):**

```

1 {
2   "status": "success",
3   "message": "Connexion réussie",
4   "data": {
5     "id": 1,
6     "nom": "Koffi Evariste",
7     "email": "evariste.koffi@nomail.com",
8     "role": "client",
9     "dateInscription": "2024-06-05T12:34:56.789Z"
10  },
11   "token": "4Zp6QJ2kV3H8jhGU0I7kjhBKJD5k89wxcn87Y1"
12 }

```
- Status:** 200 OK, 316 ms, 613 B

Figure 23: Authentification utilisateur

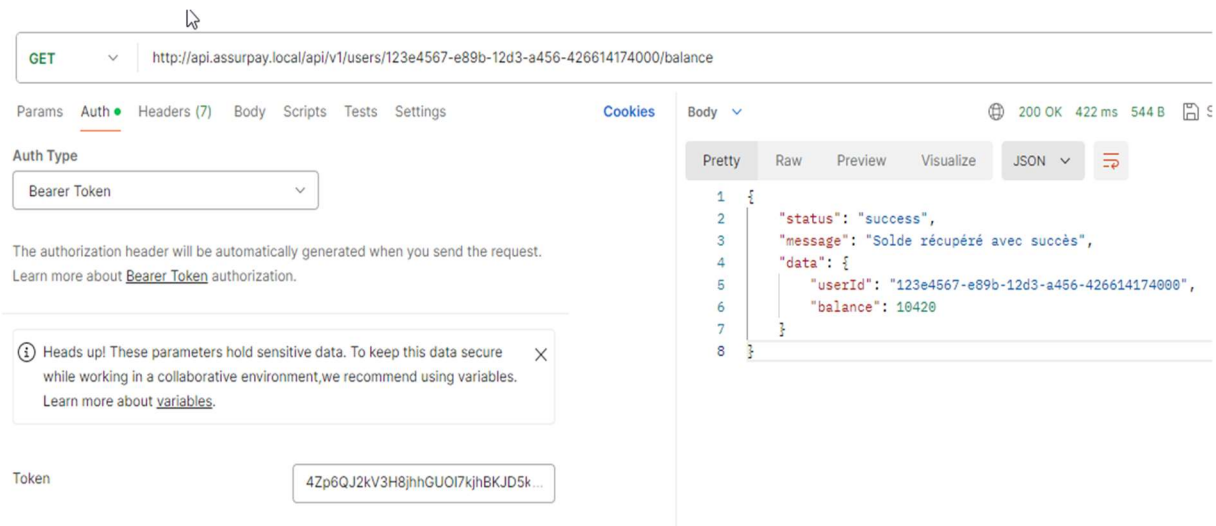


Figure 24: Consulter le solde

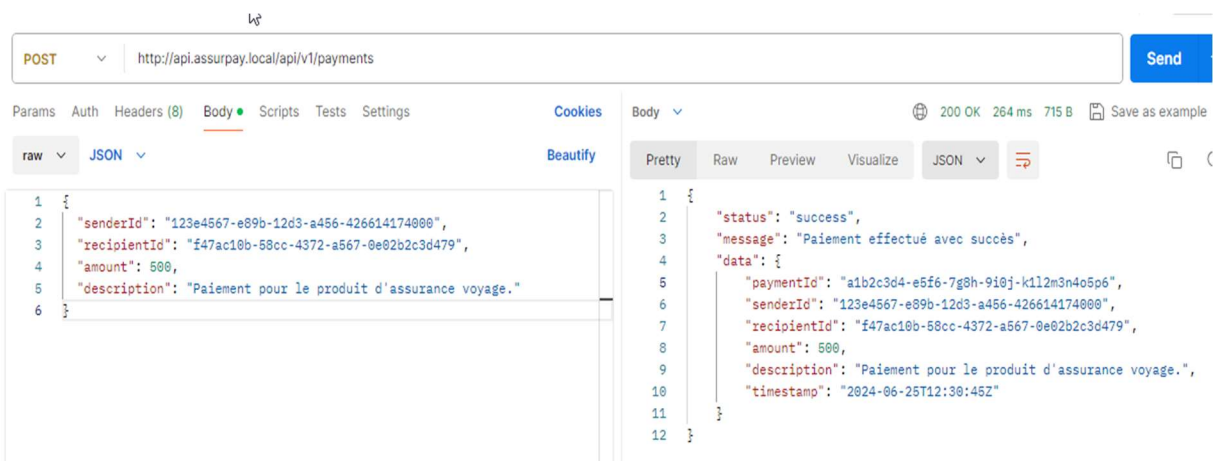


Figure 25: Effectuer un paiement

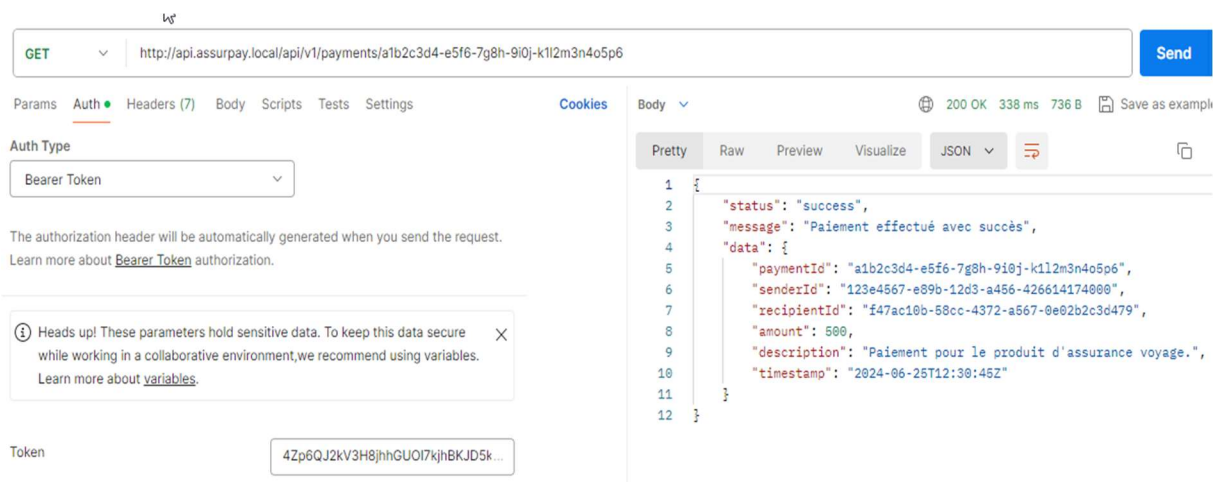


Figure 26: Vérifier un paiement reçu



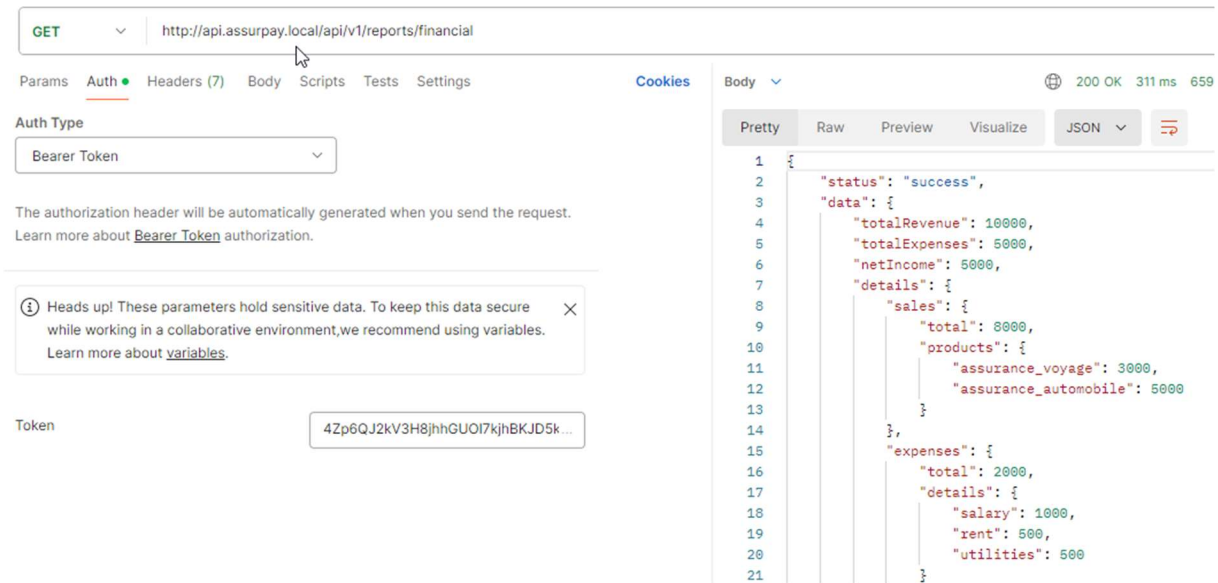


Figure 27: Générer des rapports financiers

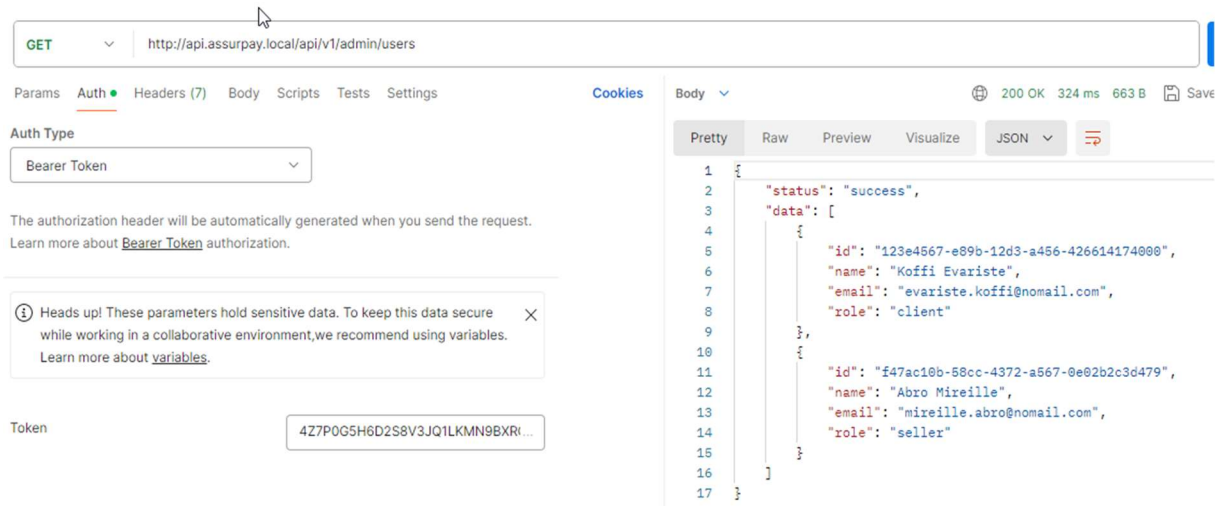


Figure 28: Voir les utilisateurs (Administrateur)

## II. DÉPLOIEMENT

Le déploiement est le processus de mise en ligne d'une application pour qu'elle soit accessible aux utilisateurs finaux. Cela inclut la configuration de l'environnement serveur, la migration des fichiers et des bases de données, ainsi que la mise en place des configurations nécessaires pour assurer le bon fonctionnement de l'application. Une fois déployée, l'application est disponible sur un serveur et peut être utilisée par les clients.

Dans le cadre de notre projet, l'application sera hébergée sur Hostinger, une plateforme d'hébergement web réputée pour sa fiabilité et ses performances. Hostinger offrira les



ressources nécessaires pour assurer un accès rapide et sécurisé à notre application, garantissant ainsi une expérience utilisateur optimale.

### III. ESTIMATION BUDGÉTAIRE

Tableau 7: Estimation budgétaire

Ressources	Nombre	Durée (mois)	Total
Développeur	3	6	3 600 000
Hébergement	1	12	31 200
Total			3 631 200

## CONCLUSION

La réalisation de ce mémoire sur la conception et le développement d'une API de paiement pour l'amélioration des opérations financières et de l'expérience client a permis de mettre en lumière plusieurs aspects cruciaux du développement d'applications financières modernes.

Au cours de ce projet, nous avons développé une API robuste et sécurisée capable de gérer différentes méthodes de paiement, notamment les cartes de crédit, les virements bancaires et les portefeuilles électroniques comme Orange Money, Moov Money, MTN Mobile Money et Wave. La modularité et la flexibilité de l'API ont été des éléments clés pour assurer une intégration fluide avec les systèmes existants des commerçants et des institutions financières.

L'architecture de l'API a été conçue en utilisant les meilleures pratiques de développement logiciel, y compris le modèle MVC, des principes de conception orientée objet, et les méthodologies Agile pour une gestion efficace du projet. L'utilisation de diagrammes UML a permis de structurer et de visualiser les différents composants et leurs interactions de manière claire et concise.

Pour les commerçants, l'API offre une solution complète pour gérer les transactions, les remboursements, et les notifications, améliorant ainsi l'efficacité opérationnelle et la satisfaction des clients. Les clients bénéficient d'une interface utilisateur simplifiée pour effectuer des paiements, consulter leur historique de transactions et recevoir des notifications en temps réel.

Le développement de cette API a également posé des défis techniques, notamment en matière de sécurité des transactions et de protection des données personnelles. Ces défis ont été relevés en intégrant des mécanismes de sécurité avancés tels que l'authentification à deux facteurs, le chiffrement des données et la gestion des permissions utilisateur.



## RÉFÉRENCES BIBLIOGRAPHIQUES

### ❖ Ouvrages

- [1] M. Adjé Louis ASSALÉ. UML2.2\_diapos, 2022, 124 pages.

### ❖ Mémoires

- [1] Kodana Adama DIARRA. Conception et réalisation des API REST d'une application de gestion d'activités de paiement d'une église catholique : cas de l'église Saint-Joseph, Yamoussoukro, INP-HB : ESI, 2022-2023, 99 pages,
- [2] Venance ZEBA. Mise en place d'une plateforme de paiement de facture en ligne e-facture, Yamoussoukro, INP-HB : ESI, 2022-2023, 54 pages.



## RÉFÉRENCES WEBOGRAPHIQUES

- [1] Guide sur la méthode Agile. [En ligne], consulté le 25/04/2024 à 14h32, disponible à l'adresse : <https://www.atlassian.com/fr/agile>
- [2] Introduction à Scrum. [En ligne], consulté le 25/04/2024 à 16h45, disponible à l'adresse : <https://www.scrum.org/resources/what-is-scrum>
- [3] Les avantages et inconvénients de la méthode Agile. [En ligne], consulté le 26/04/2024 à 09h22, disponible à l'adresse : <https://emplois.ca.indeed.com/conseils-carriere/developpement-carriere/methode-agile>
- [4] Comprendre les API et leur importance. [En ligne], consulté le 26/04/2024 à 10h40, disponible à l'adresse : <https://www.ibm.com/fr-fr/cloud/learn/api>
- [5] Bonnes pratiques de conception d'API. [En ligne], consulté le 26/04/2024 à 14h50, disponible à l'adresse : <https://learn.microsoft.com/fr-fr/azure/architecture/best-practices/api-design>
- [6] L'utilisation d'UML pour la modélisation de systèmes. [En ligne], consulté le 27/04/2024 à 08h15, disponible à l'adresse : <https://www.uml-diagrams.org/>
- [7] Introduction à la méthode Merise. [En ligne], consulté le 27/04/2024 à 09h37, disponible à l'adresse : [https://fr.wikipedia.org/wiki/Merise\\_\(informatique\)](https://fr.wikipedia.org/wiki/Merise_(informatique))
- [8] Outils et langages de programmation pour les API. [En ligne], consulté le 27/04/2024 à 11h50, disponible à l'adresse : <https://www.techniques-ingenieur.fr/base-documentaire/archives-th12/archives-automatique-et-ingenierie-systemes-tias0/archive-1/langages-de-programmation-pour-api-norme-iec-1131-3-s8030/>
- [9] Comparaison des SGBD populaires. [En ligne], consulté le 27/04/2024 à 14h05, disponible à l'adresse : <https://www.developpez.net/forums/d779158/bases-donnees/comparaison-sgbd/>
- [10] Plateformes de paiement en ligne. [En ligne], consulté le 27/04/2024 à 15h23, disponible à l'adresse : <https://www.moneris.com/fr/>



[11] Sécurité des transactions en ligne. [En ligne], consulté le 28/04/2024 à 10h00, disponible à l'adresse : <https://www.kaspersky.fr/resource-center/preemptive-safety/online-shopping>

[12] Gestion des notifications dans les systèmes distribués. [En ligne], consulté le 28/04/2024 à 11h45, disponible à l'adresse : <https://www.confluent.io/fr-fr/learn/distributed-systems/>

[13] Les outils de collaboration en développement logiciel. [En ligne], consulté le 28/04/2024 à 13h30, disponible à l'adresse : <https://www.atlassian.com/fr/software>



## ANNEXES

### **Cahier des charges**

**Titre du projet :** Conception et Développement d'une API de Paiement pour l'Amélioration des Opérations Financières et de l'Expérience Client.

**Contexte :** Avec l'essor du commerce électronique et des services en ligne, les entreprises doivent offrir des solutions de paiement sécurisées, efficaces et conviviales pour répondre aux attentes croissantes des consommateurs. Ce projet vise à développer une API de paiement intégrable, sécurisée et facile à utiliser pour améliorer les opérations financières et l'expérience utilisateur.

**Objectif général :** Concevoir et développer une API de paiement sécurisée et efficace, simplifiant les processus de paiement en ligne et garantissant la sécurité des transactions.

#### **Fonctionnalités principales :**

- Gestion des transactions financières.
- Intégration des passerelles de paiement multiples.
- Authentification et autorisation sécurisées.
- Interface utilisateur intuitive pour les développeurs.

#### **Technologies utilisées :**

- Framework Laravel pour le développement.
- Protocoles de sécurité comme HTTPS et TLS.
- OAuth pour l'authentification et l'autorisation.

#### **Besoins fonctionnels :**

##### **❖ Gestion des Transactions :**

- ❖ Créer, lire, mettre à jour et supprimer des transactions.
- ❖ Supporter les paiements récurrents.
- ❖ Générer des reçus et des rapports de transaction.

❖ **Intégration des Passerelles de Paiement :**

- Intégrer les principales passerelles de paiement (PayPal, Stripe, etc.).
- Permettre la sélection et la configuration de la passerelle de paiement par l'utilisateur.

❖ **Sécurité des Transactions :**

- Utiliser HTTPS pour toutes les communications.
- Implémenter OAuth pour sécuriser les accès.
- Protéger contre les fraudes et les attaques (ex : SQL injection, CSRF).

❖ **Expérience Utilisateur :**

- Fournir une documentation API complète et claire.
- Créer des interfaces de test pour les développeurs.

**Besoins non fonctionnels :**

❖ **Performance :**

- Assurer des temps de réponse rapides.
- Gérer efficacement un grand nombre de transactions simultanées.

❖ **Fiabilité :**

- Garantir un uptime de 99.9%.
- Implémenter des mécanismes de récupération en cas de panne.

❖ **Scalabilité :**

- Permettre une montée en charge facile pour supporter l'augmentation du volume des transactions.

❖ **Sécurité :**

- Protéger les données sensibles des utilisateurs.
- Respecter les normes de sécurité PCI-DSS.





### **Architecture :**

- Utilisation d'une architecture RESTful.
- Séparation claire entre le backend et les services API.

### **Modélisation UML :**

- Diagrammes de cas d'utilisation pour définir les interactions des utilisateurs.
- Diagrammes de classes pour la structure des données.
- Diagrammes de séquence pour les flux de transactions.

### **Spécifications Techniques :**

- Utilisation de Laravel pour le backend.
- Base de données relationnelle (MySQL/PostgreSQL).
- Système de gestion des versions via Git.

### **Phases de Développement :**

#### **❖ Phase de Préparation :**

- Installation des outils de développement.
- Configuration de l'environnement de développement.

#### **❖ Phase de Conception :**

- Création des modèles de données.
- Conception des endpoints de l'API.

#### **❖ Phase de Développement :**

- Implémentation des fonctionnalités principales.
- Intégration des passerelles de paiement.
- Développement des mécanismes de sécurité.

#### **❖ Phase de Test :**

- Tests unitaires et d'intégration.



- Tests de performance et de sécurité.

❖ **Phase de Déploiement :**

- Déploiement sur un serveur de production.
- Configuration des services de monitoring et de maintenance.

**Plan de Test :**

- **Tests Unitaires** : Validation de chaque composant individuel.
- **Tests d'Intégration** : Vérification de l'interaction entre les différents modules.
- **Tests de Performance** : Mesure des temps de réponse et de la capacité à gérer un grand nombre de transactions.
- **Tests de Sécurité** : Vérification des vulnérabilités et des failles potentielles.

**Critères de Validation :**

- Conformité aux besoins fonctionnels et non fonctionnels.
- Respect des normes de sécurité et de performance.
- Satisfaction des utilisateurs finaux et des développeurs.

**Documentation :**

- Documentation technique pour les développeurs (endpoints, paramètres, exemples d'utilisation).
- Manuel utilisateur pour les entreprises intégrant l'API.

**Formation :**

- Sessions de formation pour les développeurs internes.
- Webinaires et supports pédagogiques pour les utilisateurs externes.

**Plan de Maintenance :**

- Suivi et résolution des bugs.
- Mises à jour régulières pour améliorer les fonctionnalités et la sécurité.

## Support :

- Support technique via email et chat.
- Base de connaissances en ligne pour les utilisateurs.

## Calendrier Prévisionnel :

- Phase de Préparation : 1 mois
- Phase de Conception : 1 mois
- Phase de Développement : 3 mois
- Phase de Test et de Déploiement : 1 mois

## Budget Prévisionnel :

- Coûts de développement (salaires, outils, licences).
- Coûts de test et de déploiement.
- Coûts de maintenance et de support.

## Description textuelle des cas d'utilisation

Tableau 8: Description textuelle de "Effectuer un paiement"

Nom du Cas d'Utilisation	Effectuer un paiement
Acteur Principal	Utilisateur Final (Client)
Acteurs Secondaires	<ol style="list-style-type: none"><li>1. Commerçant (Vendeur)</li><li>2. Banque ou Institution Financière</li><li>3. Fournisseur de Services de Paiement (PSP)</li></ol>
Objectif	Permettre à l'utilisateur final de réaliser une transaction de paiement pour l'achat de biens ou de services via l'API de paiement de manière sécurisée et efficace.
Préconditions	<ol style="list-style-type: none"><li>1. L'utilisateur final doit être authentifié et autorisé à effectuer des paiements.</li><li>2. L'utilisateur final doit avoir un moyen de paiement valide (carte de crédit, compte bancaire, portefeuille électronique) enregistré ou prêt à être utilisé.</li></ol>

	<ol style="list-style-type: none"> <li>3. Le commerçant doit avoir intégré l'API de paiement et être prêt à accepter les paiements.</li> <li>4. - La connexion à l'API de paiement et aux services de la banque ou du PSP doit être opérationnelle.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur final sélectionne les produits ou services à acheter sur la plateforme du commerçant.</li> <li>2. L'utilisateur final choisit l'option de paiement pour finaliser l'achat.</li> <li>3. L'API de paiement présente à l'utilisateur final une interface pour entrer les détails de paiement (par exemple, informations de la carte de crédit, détails du compte bancaire).</li> <li>4. L'utilisateur final saisit les informations requises et les soumet.</li> <li>5. L'API de paiement valide les informations de paiement pour vérifier leur exactitude et leur validité (par exemple, numéro de carte valide, date d'expiration).</li> <li>6. Si les informations sont invalides, l'utilisateur final est invité à les corriger et à resoumettre.</li> <li>7. L'API de paiement envoie les informations de paiement à la banque ou au PSP pour l'autorisation de la transaction.</li> <li>8. La banque ou le PSP procède à l'authentification de l'utilisateur final (par exemple, via 3D Secure) et à l'autorisation de la transaction.</li> <li>9. Si l'authentification échoue, l'utilisateur final est notifié et invité à réessayer.</li> <li>10. Si l'autorisation est réussie, l'API de paiement traite la transaction.</li> <li>11. La banque ou le PSP transfère les fonds du compte de l'utilisateur final au compte du commerçant.</li> <li>12. L'API de paiement envoie une confirmation de la transaction à l'utilisateur final.</li> <li>13. Le commerçant est notifié de la réception du paiement.</li> </ol>

	14. La transaction est enregistrée dans le système pour un suivi et une traçabilité ultérieure.
<b>Scénarios Alternatifs</b>	<ol style="list-style-type: none"> <li>1. Si les informations de paiement fournies par l'utilisateur final sont invalides, l'API de paiement affiche un message d'erreur et invite l'utilisateur à corriger les informations et à resoumettre (retour à l'étape 2).</li> <li>2. Si l'authentification de l'utilisateur final échoue, l'API de paiement affiche un message d'erreur et invite l'utilisateur à réessayer ou à utiliser une autre méthode de paiement (retour à l'étape 2).</li> <li>3. Si la transaction est refusée par la banque ou le PSP, l'API de paiement affiche un message indiquant le refus et invite l'utilisateur à utiliser une autre méthode de paiement ou à contacter sa banque (retour à l'étape 2).</li> <li>4. Si le traitement du paiement échoue en raison d'un problème technique, l'API de paiement affiche un message d'erreur et invite l'utilisateur à réessayer plus tard.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Le paiement est autorisé, traité et confirmé.</li> <li>2. L'utilisateur final reçoit une confirmation de la transaction.</li> <li>3. Le commerçant est notifié de la réception du paiement.</li> <li>4. La transaction est enregistrée dans le système pour un suivi ultérieur.</li> </ol>

Tableau 9: Description textuelle de "Consulter l'historique des paiements"

<b>Nom du Cas d'Utilisation</b>	<b>Consulter l'historique des paiements</b>
<b>Acteur Principal</b>	Utilisateur Final (Client)
<b>Acteurs Secondaires</b>	<ol style="list-style-type: none"> <li>1. Commerçant (Vendeur)</li> <li>2. Administrateur Système</li> </ol>



<b>Objectif</b>	Permettre à l'utilisateur final de visualiser l'historique de ses transactions de paiement pour suivre ses dépenses, vérifier les paiements effectués, et obtenir des informations sur chaque transaction.
<b>Préconditions</b>	<ol style="list-style-type: none"><li>1. L'utilisateur final doit être authentifié et autorisé à accéder à ses informations de paiement.</li><li>2. Les transactions de paiement doivent être enregistrées dans le système avec les détails pertinents.</li></ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"><li>1. L'utilisateur final se connecte à la plateforme du commerçant avec ses identifiants.</li><li>2. Le système vérifie les informations d'authentification et accorde l'accès.</li><li>3. L'utilisateur final navigue vers la section "Historique des paiements" de la plateforme.</li><li>4. L'API de paiement récupère les transactions de l'utilisateur final à partir de la base de données.</li><li>5. L'API de paiement présente à l'utilisateur final une liste de ses transactions de paiement.</li><li>6. Chaque entrée de la liste inclut des détails tels que l'identifiant de la transaction, le montant, la date, le commerçant, et le statut (réussi, en attente, échoué).</li><li>7. L'utilisateur final utilise des filtres pour affiner la liste des transactions (par exemple, par date, montant, commerçant).</li><li>8. L'utilisateur final peut rechercher des transactions spécifiques en utilisant des mots-clés ou des critères précis.</li><li>9. L'utilisateur final clique sur une transaction pour voir des détails supplémentaires (par exemple, les articles achetés, les méthodes de paiement utilisées, les références de transaction).</li><li>10. L'utilisateur final a l'option de télécharger l'historique des paiements au format PDF ou CSV.</li><li>11. L'utilisateur final peut également imprimer l'historique des paiements pour ses archives personnelles.</li></ol>

<b>Scénarios Alternatifs</b>	<ol style="list-style-type: none"> <li>1. Si aucune transaction n'est trouvée pour l'utilisateur final, l'API de paiement affiche un message indiquant qu'il n'y a pas d'historique de paiements disponible.</li> <li>2. Si l'authentification échoue, l'utilisateur final est invité à réessayer avec les bonnes informations d'identification ou à réinitialiser son mot de passe.</li> <li>3. Si une erreur technique empêche l'accès ou l'affichage de l'historique des paiements, un message d'erreur est affiché et l'utilisateur est invité à réessayer plus tard ou à contacter le support client.</li> </ol>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur final accède à une liste de ses transactions passées avec des détails tels que les montants, les dates, et les statuts des paiements.</li> <li>2. L'utilisateur peut filtrer et rechercher des transactions spécifiques.</li> <li>3. L'utilisateur peut télécharger ou imprimer l'historique des paiements pour ses archives personnelles.</li> </ol>

Tableau 10: Description textuelle de "Recevoir des notifications de paiement"

<b>Nom du Cas d'Utilisation</b>	<b>Recevoir des notifications de paiement</b>
<b>Acteur Principal</b>	Utilisateur Final (Client)
<b>Acteurs Secondaires</b>	<ol style="list-style-type: none"> <li>1. Commerçant (Vendeur)</li> <li>2. Administrateur Système</li> <li>3. Banque ou Institution Financière</li> </ol>
<b>Objectif</b>	Informar l'utilisateur final en temps réel des événements importants liés à ses transactions de paiement, tels que la confirmation des paiements, les échecs de transaction, les remboursements et les rétro facturations, afin d'améliorer l'expérience utilisateur et la transparence.



<b>Préconditions</b>	<ol style="list-style-type: none"><li>1. L'utilisateur final doit être authentifié et avoir des moyens de contact valides (par exemple, adresse e-mail, numéro de téléphone) enregistrés dans le système.</li><li>2. Le système de notification doit être configuré et opérationnel.</li><li>3. Les événements de paiement doivent être enregistrés et suivis par l'API de paiement.</li></ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"><li>4. L'utilisateur final configure ses préférences de notification via le portail utilisateur (par exemple, choisir de recevoir des notifications par e-mail, SMS, ou via une application mobile).</li><li>5. Une transaction de paiement est initiée, confirmée, échoue, ou subit une autre action nécessitant une notification (par exemple, remboursement, rétro facturation).</li><li>6. L'API de paiement enregistre l'événement pertinent dans le système et génère une notification correspondante.</li><li>7. Le système de notification envoie la notification à l'utilisateur final selon ses préférences configurées (e-mail, SMS, notification push).</li><li>8. La notification inclut les détails de l'événement (par exemple, montant, date, statut de la transaction, identifiant de la transaction).</li><li>9. L'utilisateur final reçoit la notification sur le moyen de contact choisi.</li><li>10. L'utilisateur final peut consulter les détails de la notification pour prendre les actions nécessaires (par exemple, vérifier un paiement réussi, contacter le support en cas d'échec de la transaction).</li><li>11. L'API de paiement enregistre la confirmation que la notification a été envoyée et reçue (par exemple, statut de livraison de l'e-mail ou du SMS).</li></ol>



<b>Scénarios Alternatifs</b>	<p>12. Si l'utilisateur final n'a pas configuré ses préférences de notification, le système utilise les paramètres par défaut (par exemple, e-mail) pour envoyer les notifications.</p> <p>13. Si l'envoi de la notification échoue (par exemple, adresse e-mail incorrecte, numéro de téléphone invalide), le système enregistre l'échec et réessaie l'envoi ou notifie l'utilisateur par un autre moyen disponible.</p> <p>14. L'utilisateur final peut choisir de se désabonner des notifications non critiques via le portail utilisateur, ce qui met à jour ses préférences de notification.</p>
<b>Postconditions</b>	<p>1. L'utilisateur final reçoit des notifications pour les événements de paiement pertinents.</p> <p>2. Les notifications sont enregistrées dans le système pour un suivi ultérieur.</p>

Tableau 11: Description textuelle de "Gérer les moyens de paiement"

<b>Nom du Cas d'Utilisation</b>	<b>Gérer les moyens de paiement</b>
<b>Acteur Principal</b>	Utilisateur Final (Client)
<b>Objectif</b>	Permettre à l'utilisateur final de gérer les différents moyens de paiement associés à son compte, tels que les cartes de crédit, les comptes bancaires, les portefeuilles électroniques, afin de faciliter les transactions futures et de garantir la sécurité des paiements.
<b>Préconditions</b>	<p>1. L'utilisateur final doit être authentifié et avoir accès à son compte.</p> <p>2. L'utilisateur final doit avoir au moins un moyen de paiement enregistré dans le système.</p> <p>3. Le système doit fournir une interface utilisateur pour gérer les moyens de paiement.</p>
<b>Scénario Principal</b>	4. L'utilisateur final accède à la section "Gérer les moyens de paiement" via le portail utilisateur ou l'application mobile.

	<ol style="list-style-type: none"> <li>5. Le système affiche une liste des moyens de paiement actuellement enregistrés par l'utilisateur final.</li> <li>6. Chaque moyen de paiement est présenté avec des détails tels que le type (carte de crédit, compte bancaire, portefeuille électronique), les quatre derniers chiffres du numéro, et la date d'expiration.</li> <li>7. L'utilisateur final sélectionne l'option pour ajouter un nouveau moyen de paiement.</li> <li>8. L'utilisateur final entre les informations requises pour le nouveau moyen de paiement (par exemple, numéro de carte de crédit, date d'expiration, nom sur la carte).</li> <li>9. Le système valide les informations et ajoute le nouveau moyen de paiement à la liste.</li> <li>10. L'utilisateur final sélectionne un moyen de paiement existant à supprimer.</li> <li>11. Le système confirme la suppression et retire le moyen de paiement de la liste.</li> <li>12. L'utilisateur final sélectionne un moyen de paiement existant à modifier.</li> <li>13. L'utilisateur final modifie les informations nécessaires (par exemple, met à jour la date d'expiration de la carte de crédit).</li> <li>14. Le système valide les modifications et met à jour le moyen de paiement dans la liste.</li> </ol>
<b>Scénarios Alternatifs</b>	<ol style="list-style-type: none"> <li>15. Si aucun moyen de paiement n'est enregistré pour l'utilisateur final, le système affiche un message indiquant qu'aucun moyen de paiement n'est disponible et invite l'utilisateur à en ajouter un.</li> <li>16. Si les informations fournies pour ajouter ou modifier un moyen de paiement ne sont pas valides, le système affiche un message d'erreur et invite l'utilisateur à corriger les informations.</li> </ol>

	17. Avant de supprimer définitivement un moyen de paiement, le système demande une confirmation de la part de l'utilisateur final pour éviter les suppressions accidentelles.
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Les moyens de paiement de l'utilisateur final sont mis à jour conformément à ses actions (ajout, suppression, modification).</li> <li>2. Les changements apportés aux moyens de paiement sont enregistrés dans le système pour une utilisation future lors des transactions.</li> </ol>

Tableau 12: Description textuelle de "Contester un paiement"

<b>Nom du Cas d'Utilisation</b>	<b>Contester un paiement</b>
<b>Acteur Principal</b>	Utilisateur Final (Client)
<b>Acteurs Secondaires</b>	<ol style="list-style-type: none"> <li>1. Commerçant (Vendeur)</li> <li>2. Service Client</li> <li>3. Banque ou Institution Financière</li> </ol>
<b>Objectif</b>	Permettre à l'utilisateur final de contester une transaction de paiement en cas de problème ou de litige avec le commerçant, tels que des biens ou services non reçus, des produits défectueux, ou des fraudes, afin de demander un remboursement ou une résolution du litige.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur final doit être authentifié et avoir accès à son compte.</li> <li>2. La transaction de paiement à contester doit être récente et enregistrée dans le système.</li> <li>3. Le commerçant doit être identifiable et avoir un accord de traitement des litiges avec le service client.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>1. L'utilisateur final accède à la section "Contester un paiement" via le portail utilisateur ou l'application mobile.</li> <li>2. L'utilisateur final sélectionne la transaction de paiement spécifique à contester dans la liste des transactions récentes.</li> </ol>



	<ol style="list-style-type: none"><li>3. L'utilisateur final fournit une raison détaillée pour contester la transaction (par exemple, produit non reçu, service non conforme, fraude).</li><li>4. L'utilisateur final télécharge ou fournit toute preuve ou documentation pertinente pour appuyer sa contestation (par exemple, captures d'écran, factures, communications avec le commerçant).</li><li>5. L'utilisateur final confirme sa demande de contestation et la soumet au système.</li><li>6. Le système enregistre la demande de contestation et la transmet au service client ou à la banque pour enquête et résolution.</li></ol>
<b>Scénarios Alternatifs</b>	<p><b>Transaction Non Trouvée :</b></p> <ul style="list-style-type: none"><li>• Si aucune transaction correspondante n'est trouvée pour la contestation, le système affiche un message indiquant qu'aucune transaction récente n'est disponible pour la contestation.</li></ul> <p><b>Demande Incomplète :</b></p> <ul style="list-style-type: none"><li>• Si l'utilisateur final ne fournit pas toutes les informations requises pour la contestation, le système affiche un message d'erreur et invite l'utilisateur à fournir les informations manquantes.</li></ul> <p><b>Échec de Soumission :</b></p> <ul style="list-style-type: none"><li>• Si la soumission de la contestation échoue en raison d'un problème technique, le système affiche un message d'erreur et invite l'utilisateur à réessayer plus tard.</li></ul>
<b>Postconditions</b>	<ol style="list-style-type: none"><li>1. Une demande de contestation de paiement est initiée et transmise au service client ou à la banque pour enquête et résolution.</li><li>2. L'utilisateur final reçoit une confirmation de la réception de la demande de contestation.</li></ol>

Tableau 13: Description textuelle de "Initier une demande de paiement"

Nom du Cas d'Utilisation	Initier une demande de paiement
Acteur Principal	Utilisateur Final (Client)
Acteurs Secondaires	<ol style="list-style-type: none"> <li>1. Commerçant (Vendeur)</li> <li>2. Service Client</li> </ol>
Objectif	Permettre à l'utilisateur final d'envoyer une demande de paiement à un tiers (généralement un autre utilisateur ou un commerçant) pour solliciter un paiement pour des biens ou des services fournis.
Préconditions	<ol style="list-style-type: none"> <li>1. L'utilisateur final doit être authentifié et avoir accès à son compte.</li> <li>2. Le destinataire de la demande de paiement doit être identifié et accessible dans le système.</li> <li>3. Le système doit fournir une interface utilisateur pour initier et gérer les demandes de paiement.</li> </ol>
Scénario Principal	<ol style="list-style-type: none"> <li>1. L'utilisateur final accède à la section "Initier une demande de paiement" via le portail utilisateur ou l'application mobile.</li> <li>2. L'utilisateur final sélectionne l'option pour créer une nouvelle demande de paiement.</li> <li>3. L'utilisateur final saisit les informations du destinataire de la demande de paiement (par exemple, nom, adresse e-mail, numéro de téléphone).</li> <li>4. Le système recherche et identifie le destinataire dans la base de données.</li> <li>5. L'utilisateur final entre le montant de la demande de paiement ainsi que les détails pertinents (par exemple, motif de la demande, date d'échéance).</li> <li>6. L'utilisateur final passe en revue les informations de la demande de paiement pour s'assurer de leur exactitude.</li> <li>7. L'utilisateur final confirme la demande de paiement et l'envoie au destinataire.</li> </ol>

<b>Scénarios Alternatifs</b>	<p><b>Destinataire Non Trouvé :</b></p> <ul style="list-style-type: none"> <li>Si le système ne parvient pas à trouver le destinataire basé sur les informations fournies, il affiche un message d'erreur et invite l'utilisateur à vérifier les informations saisies.</li> </ul> <p><b>Montant Incorrect :</b></p> <ul style="list-style-type: none"> <li>Si le montant spécifié pour la demande de paiement est invalide ou non conforme aux limites prédéfinies, le système affiche un message d'erreur et invite l'utilisateur à corriger le montant.</li> </ul> <p><b>Échec de l'Envoi :</b></p> <ul style="list-style-type: none"> <li>Si l'envoi de la demande de paiement échoue en raison d'un problème technique ou de connectivité, le système affiche un message d'erreur et invite l'utilisateur à réessayer ultérieurement.</li> </ul>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>La demande de paiement est envoyée au destinataire avec les détails pertinents.</li> <li>Le destinataire peut accepter ou rejeter la demande de paiement.</li> <li>Le statut de la demande de paiement est mis à jour dans le système.</li> </ol>

Tableau 14: Description textuelle de "Générer des rapports financiers"

<b>Nom du Cas d'Utilisation</b>	<b>Générer des rapports financiers</b>
<b>Acteur Principal</b>	Administrateur du Système
<b>Objectif</b>	Permettre à l'administrateur du système d'accéder et de générer des rapports financiers basés sur les données des transactions de paiement,

	afin de fournir des informations précieuses sur l'état financier de l'entreprise ou de l'organisation.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>1. L'administrateur du système doit être authentifié et avoir les autorisations nécessaires pour accéder aux rapports financiers.</li> <li>2. Les données des transactions de paiement doivent être enregistrées et accessibles dans le système.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>1. L'administrateur du système accède à la section "Générer des rapports financiers" via l'interface d'administration du système.</li> <li>2. L'administrateur spécifie les critères de filtrage pour le rapport financier, tels que la période de temps, le type de transaction, le mode de paiement, etc.</li> <li>3. Le système utilise les critères spécifiés pour extraire les données des transactions de paiement pertinentes de la base de données.</li> <li>4. Le système analyse les données et génère un rapport financier avec les informations demandées, telles que le total des ventes, les revenus nets, les remboursements, etc.</li> </ol>
<b>Scénarios Alternatifs</b>	<p><b>Aucune Donnée Disponible :</b></p> <ul style="list-style-type: none"> <li>• Si aucune donnée n'est disponible pour les critères spécifiés, le système affiche un message d'avertissement indiquant qu'aucun rapport ne peut être généré pour la période ou les critères sélectionnés.</li> </ul> <p><b>Échec de Génération du Rapport :</b></p> <ul style="list-style-type: none"> <li>• Si la génération du rapport échoue en raison d'un problème technique ou d'une erreur système, le système affiche un message d'erreur et invite l'administrateur à réessayer ultérieurement ou à contacter le support technique.</li> </ul>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. Un rapport financier est généré avec les informations demandées et mis à disposition de l'administrateur du système.</li> </ol>

	2. Les données utilisées pour générer le rapport sont sécurisées et protégées contre tout accès non autorisé.
--	---

Tableau 15: Description textuelle de "Configurer les paramètres de l'API"

<b>Nom du Cas d'Utilisation</b>	<b>Configurer les paramètres de l'API</b>
<b>Acteur Principal</b>	Développeur / Administrateur du Système
<b>Objectif</b>	Permettre au développeur ou à l'administrateur du système de configurer les paramètres de l'API de paiement pour adapter son comportement et ses fonctionnalités en fonction des besoins spécifiques de l'entreprise ou des utilisateurs.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>1. Le développeur ou l'administrateur doit être authentifié et avoir les autorisations nécessaires pour accéder aux fonctionnalités de configuration de l'API.</li> <li>2. L'API de paiement doit être intégrée au système et accessible pour la configuration.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>3. Le développeur ou l'administrateur accède à la section "Configurer les paramètres de l'API" via l'interface d'administration ou de développement.</li> <li>4. Le système affiche les paramètres actuels de l'API de paiement, y compris les options de configuration disponibles.</li> <li>5. Le développeur ou l'administrateur sélectionne les paramètres qu'il souhaite modifier et spécifie les nouvelles valeurs ou options de configuration.</li> <li>6. Les paramètres peuvent inclure des options telles que les clés d'API, les limites de requêtes, les stratégies de sécurité, etc.</li> <li>7. Le développeur ou l'administrateur passe en revue les modifications apportées aux paramètres de l'API pour s'assurer de leur exactitude.</li> <li>8. Une fois validées, les modifications sont enregistrées dans le système et deviennent effectives immédiatement.</li> </ol>



<b>Scénarios Alternatifs</b>	<b>Échec de Validation :</b> <ul style="list-style-type: none"> <li>Si une erreur est détectée lors de la validation des modifications, le système affiche un message d'erreur et invite le développeur ou l'administrateur à corriger les paramètres invalides avant d'enregistrer les modifications.</li> </ul>
<b>Postconditions</b>	Les modifications apportées aux paramètres de l'API sont enregistrées et prennent effet immédiatement, affectant le comportement de l'API dans le système.

Tableau 16: Description textuelle de "Autoriser une transaction"

<b>Nom du Cas d'Utilisation</b>	<b>Autoriser une transaction</b>
<b>Acteur Principal</b>	Système de Gestion des Transactions
<b>Acteurs Secondaires</b>	Utilisateur Final (Client), API de Paiement, Système de Gestion des Utilisateurs
<b>Objectif</b>	Permettre au système de gestion des transactions d'autoriser une transaction financière initiée par un utilisateur final, en vérifiant les informations de paiement, les fonds disponibles et en effectuant toutes les vérifications de sécurité nécessaires.
<b>Préconditions</b>	<ol style="list-style-type: none"> <li>L'utilisateur final a initié une transaction via une interface utilisateur.</li> <li>Les informations de paiement fournies par l'utilisateur ont été validées et transmises au système de gestion des transactions.</li> </ol>
<b>Scénario Principal</b>	<ol style="list-style-type: none"> <li>Le système de gestion des transactions reçoit les informations de transaction initiées par l'utilisateur final, y compris le montant, les détails du commerçant, les informations de paiement, etc.</li> <li>Le système vérifie l'exactitude et la validité des informations de paiement fournies par l'utilisateur final, y compris les détails de la carte de crédit, les coordonnées bancaires, etc.</li> </ol>

	<ol style="list-style-type: none"> <li>3. Le système vérifie la disponibilité des fonds sur le compte de l'utilisateur final ou la limite de crédit disponible, en tenant compte du montant de la transaction et de tout frais associé.</li> <li>4. Le système autorise la transaction et procède au traitement du paiement.</li> <li>5. Les fonds sont réservés ou transférés conformément aux détails de la transaction, et un identifiant de transaction unique est généré.</li> <li>6. Une fois la transaction autorisée, le système génère un reçu de transaction contenant les détails de la transaction, tels que le montant, la date, le commerçant, etc.</li> <li>7. Le reçu est envoyé à l'utilisateur final et/ou au commerçant pour confirmation de la transaction.</li> </ol>
<b>Scénarios Alternatifs</b>	<p><b>Échec de Vérification des Informations de Paiement :</b></p> <ul style="list-style-type: none"> <li>• Si les informations de paiement fournies par l'utilisateur final sont invalides ou incorrectes, le système rejette la transaction et informe l'utilisateur final de la raison du rejet.</li> </ul> <p><b>Fonds Insuffisants :</b></p> <ul style="list-style-type: none"> <li>• Si les fonds disponibles sur le compte de l'utilisateur final sont insuffisants pour couvrir le montant de la transaction, le système rejette la transaction et informe l'utilisateur final du solde insuffisant.</li> </ul>
<b>Postconditions</b>	<ol style="list-style-type: none"> <li>1. La transaction est autorisée avec succès et les fonds sont réservés ou transférés conformément aux détails de la transaction.</li> <li>2. Un reçu ou une confirmation de la transaction est généré et envoyé à l'utilisateur final et/ou au commerçant.</li> </ol>

## Diagrammes de séquence

Tableau 17: Diagramme de séquence "Notification"

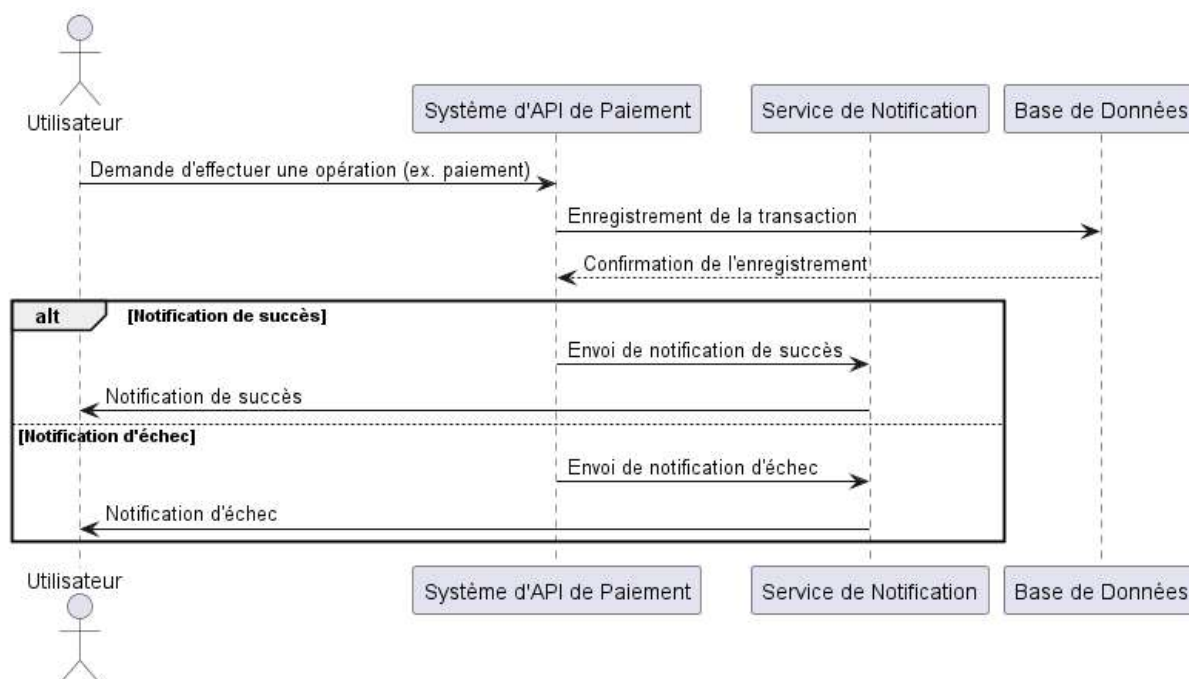


Tableau 19: Diagramme de séquence de "surveillance et de reporting"

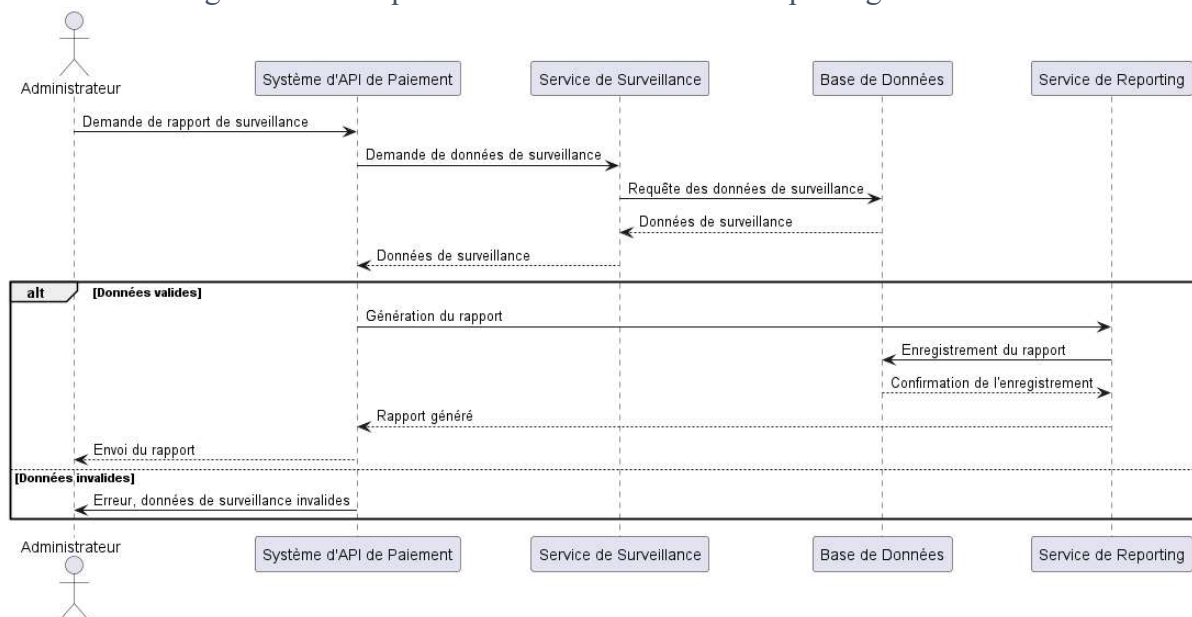


Tableau 18: Diagramme de séquence "sécurité"

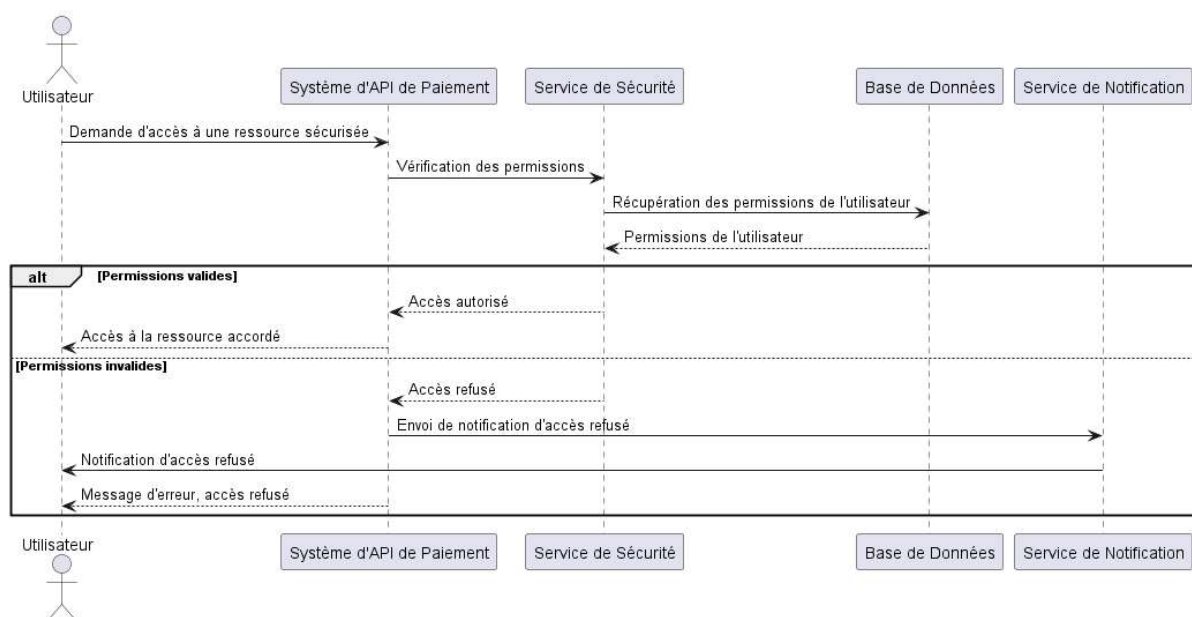
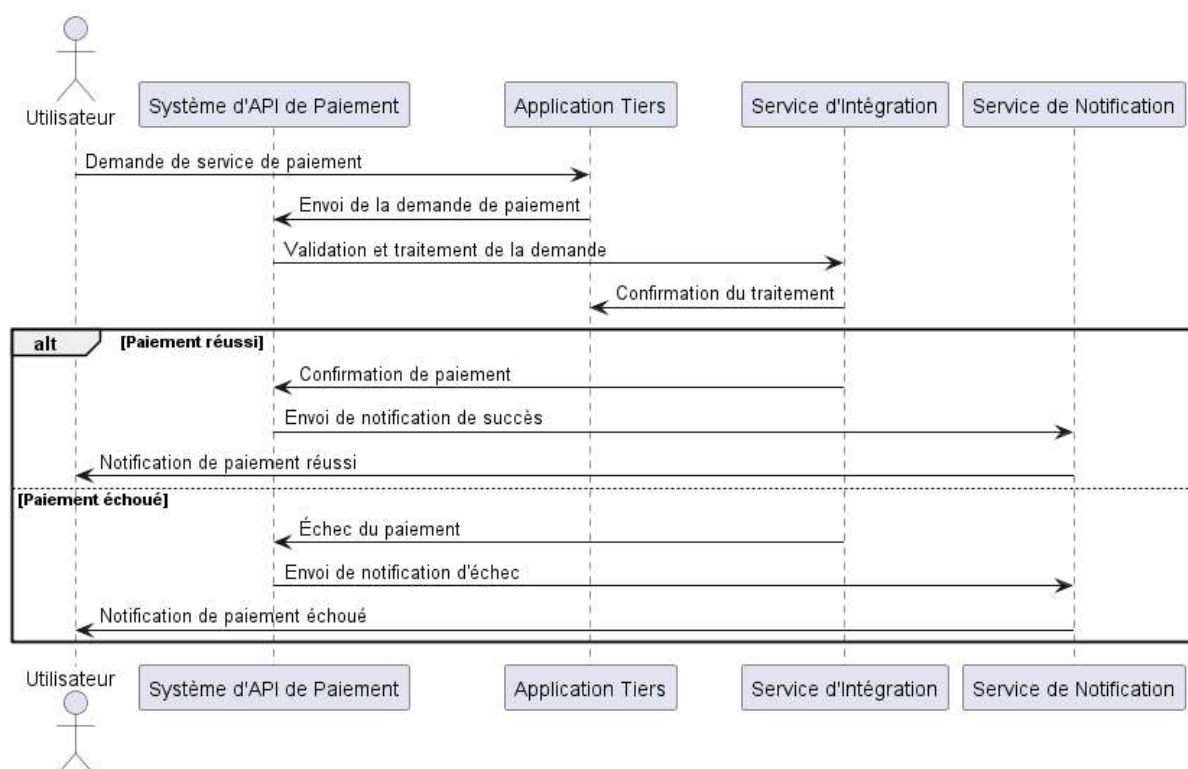


Tableau 20: Diagramme de séquence "intégration avec les applications tierces"





## TABLE DES MATIÈRES

DÉDICACE .....	I
REMERCIEMENTS .....	II
SOMMAIRE .....	III
AVANT-PROPOS .....	V
LISTE DES SIGLES ET ABRÉVIATIONS .....	VI
LISTE DES TABLEAUX .....	VIII
LISTE DES FIGURES .....	IX
RÉSUMÉ .....	X
INTRODUCTION .....	1
PARTIE I : CADRE ET CONTEXTE DU PROJET .....	2
CHAPITRE I : PRÉSENTATION DU CADRE .....	3
I.    PRÉSENTATION DE LA STRUCTURE D'ACCUEIL.....	3
II.   DOMAINE D'ACTIVITÉS.....	3
III.  ORGANIGRAMME HIÉRARCHIQUE .....	3
CHAPITRE II : DESCRIPTION DU PROJET .....	4
I.    PRÉSENTATION DU CONTEXTE .....	4
II.   OBJECTIFS DU PROJET .....	4
II.1  OBJECTIF GÉNÉRAL .....	4
II.2  OBJECTIFS SPÉCIFIQUES .....	5
III.  CAHIER DES CHARGES .....	6
IV.   ÉTUDE DE L'EXISTANCE .....	8
IV.1  DESCRIPTION DE L'EXISTANT .....	8
IV.2  CRITIQUE DE L'EXISTANT .....	8
IV.2.1  Forces du système .....	8
IV.2.2  Faiblesses du système .....	8
IV.2.3  Solutions proposées.....	8
V.    MÉTHODE DE GESTION DE PROJET .....	9
V.1  LA MÉTHODE AGILE .....	9
V.1.1  Définition.....	9



V.1.2	Principes de la méthode Agile .....	9
V.1.3	Présentation de Scrum .....	10
V.2	STRUCTURE DE SCRUM.....	10
V.2.1	Rôles dans Scrum.....	10
V.2.1.1	Scrum Master .....	10
V.2.1.2	Product Owner .....	10
V.2.1.3	Équipe de développement .....	10
V.2.2	Évènements Scrum .....	10
V.2.2.1	Sprint .....	10
V.2.2.2	Planification du Sprint .....	11
V.2.2.3	Réunion quotidienne .....	11
V.2.2.4	Revue de Sprint .....	11
V.2.2.5	Rétrospective de Sprint .....	11
V.2.3	Artéfacts Scrum .....	11
V.2.3.1	Backlog Produit.....	11
V.2.3.2	Backlog de Sprint .....	11
V.2.3.3	Increment .....	11
V.2.4	Mise en œuvre de Scrum .....	12
V.2.4.1	Initiation d'un projet Scrum .....	12
V.2.4.2	Planification et Estimation.....	12
V.2.4.3	Exécution et contrôle des Sprints .....	12
V.2.4.4	Suivi et Adaptation .....	12
V.3	AVANTAGES DE SCRUM .....	12
VI.	ORGANISATION DU TRAVAIL.....	13
PARTIE II :	ÉTUDE CONCEPTUELLE .....	14
CHAPITRE III :	APPROCHE MÉTHODOLOGIQUE .....	15
I.	INTRODUCTION À LA MÉTHODE D'ANALYSE ET DE CONCEPTION .....	15
I.1	MÉTHODE D'ANALYSE.....	15
I.2	LA MÉTHODE MERISE.....	15
I.2.1	Présentation .....	15
I.2.2	Les aspects du cycle de développement .....	15
I.2.2.1	La démarche ou cycle de vie .....	16
I.2.2.2	Le raisonnement ou cycle d'abstraction .....	16
I.2.2.2.1	Niveau conceptuel.....	17
I.2.2.2.2	Niveau organisationnel .....	17
I.2.2.2.3	Niveau logique .....	17
I.2.2.2.4	Niveau physique.....	17
I.2.2.3	La maîtrise ou cycle décision .....	17



I.3	PU/UML .....	17
I.3.1	<i>Processus Unifié (PU)</i> .....	17
I.3.2	<i>Unified Modeling Language (UML)</i> .....	18
I.4	COMPARAISON DES MÉTHODES D'ANALYSE .....	18
II.	CHOIX DE LA MÉTHODE D'ANALYSE .....	19
CHAPITRE IV : CRÉATION DE L'ARCHITECTURE DU SYSTÈME.....		20
I.	IDENTIFICATION DES ACTEURS ET DES CAS D'UTILISATION .....	20
I.1	LES ACTEURS .....	20
I.2	LES CAS D'UTILISATION .....	21
I.3	DIAGRAMME DES CAS D'UTILISATION .....	24
I.4	DESCRIPTIONS TEXTUELLES DES CAS D'UTILISATION .....	25
I.4.1	<i>Traiter un paiement</i> .....	25
I.4.2	<i>Configurer des options de paiement</i> .....	26
I.4.3	<i>Gérer un remboursement ou une annulation</i> .....	27
I.4.4	<i>Gérer les utilisateurs et les permissions</i> .....	29
II.	DIAGRAMMES DE SÉQUENCE .....	31
II.1	TRAITEMENT DES TRANSACTIONS .....	31
II.2	AUTHENTIFICATION ET AUTORISATION.....	32
II.3	GESTION DES UTILISATEURS.....	33
II.4	GESTION DES ERREURS .....	34
III.	DIAGRAMME DE CLASSES.....	35
PARTIE III : MISE EN OEUVRE.....		36
CHAPITRE V : TECHNIQUE D'ANALYSE .....		37
I.	ARCHITECTURE DU SYSTÈME.....	37
I.1	PRÉSENTATION DES COUCHES .....	37
I.1.1	<i>La couche de données</i> .....	37
I.1.2	<i>La couche de traitement</i> .....	37
I.1.3	<i>La couche de présentation</i> .....	37
I.2	COMPARAISON DES DIFFÉRENTES ARCHITECTURES .....	37
I.3	ARCHITECTURE RETENUE .....	38
II.	OUTILS DE DÉVELOPPEMENT .....	38
II.1	LARAVEL .....	38
II.2	LARAVEL PASSPORT .....	39





II.3	LARAVEL SANCTUM .....	39
II.4	ANGULARJS.....	40
II.5	LARAGON .....	40
II.6	MYSQL .....	40
II.7	POSTMAN .....	41
II.8	VISUAL STUDIO CODE.....	41
II.9	GIT .....	42
II.10	BITBUCKET.....	42
II.11	JIRA.....	43
<b>CHAPITRE VI : RÉALISATION .....</b>		<b>44</b>
<b>I.</b>	<b>TESTS DES FONCTIONNALITÉS .....</b>	<b>44</b>
I.1	DÉFINITION DES ENDPOINTS.....	44
I.2	TESTS DES ENDPOINTS.....	45
<b>II.</b>	<b>DÉPLOIEMENT .....</b>	<b>47</b>
<b>III.</b>	<b>ESTIMATION BUDGÉTAIRE .....</b>	<b>48</b>
<b>CONCLUSION.....</b>		<b>49</b>
<b>RÉFÉRENCES BIBLIOGRAPHIQUES.....</b>		<b>X</b>
<b>RÉFÉRENCES WEBOGRAPHIQUES .....</b>		<b>XI</b>
<b>ANNEXES.....</b>		<b>XIII</b>
<b>TABLE DES MATIÈRES .....</b>		<b>XXXVI</b>