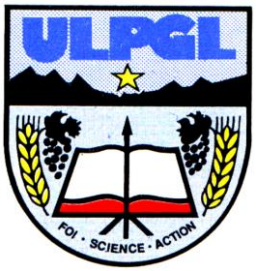
UNIVERSITE LIBRE DES PAYS DES GRANDS LACS

FACULTE DES SCIENCES ET TECHNOLOGIES

DEPARTEMENT DE GENIE ELECTRIQUE ET INFORMATIQUE



BP. 368 GOMA

www.ulpgl.net

CONCEPTION ET REALISATION D'UN SYSTEME DE GESTION DE PAIEMENT RELIANT LES AGENTS D'UNE BANQUE ET UNE UNIVERSITÉ : CAS DU SERVICE CASH EXPRESS D'EQUITY BCDC ET L'ULPGL

Travail présenté en vue de l'obtention du Diplôme de
Bachelor en Sciences de l'ingénieur
Option : Génie Informatique

Présenté par : ISHARA MUDAHAMA
Directeur : MSc. Ir. VINGI PATRICK
Encadreur : Ir. JOHNSON KISAMBA

ANNÉE ACADÉMIQUE 2023 - 2024

Dédicace

A mes très chers parents.

ISHARA MUDAHAMA Frédérick

Epigraphe

« Enseigne-nous à bien compter nos jours, Afin que nous appliquions notre cœur à la sagesse »

Psaumes 90 :12

Remerciements

Avant toute chose, je rends grâce à Dieu Tout-Puissant, source de toute sagesse, de m'avoir accordé la vie, la santé, la persévérance et la paix d'esprit nécessaires à la réalisation de ce travail de fin de cycle. Je tiens à exprimer ma profonde gratitude à l'Université Libre des Pays des Grands Lacs (ULPGL-Goma), en particulier à la Faculté des Sciences et Technologies, de m'avoir offert un cadre académique propice au développement de mes compétences tout au long de mon cursus.

Mes remerciements les plus sincères s'adressent au Master Ing. NZANZU VINGI Patrick, mon Directeur de mémoire, pour ses orientations précieuses, ses conseils méthodiques et sa bienveillance constante durant ce travail. Je remercie également l'Ir. JOHNSON KISAMBA, mon Encadreur, pour sa disponibilité, son appui technique et ses encouragements tout au long de ce projet.

Je tiens à saluer l'ensemble du corps professoral du Département de Génie Électrique et Informatique, et plus particulièrement le Professeur Alain Akwir, le Professeur Baraka Mushage Olivier ainsi que le Master Molo Mbasa Joachim, dont la passion pour les connaissances et la rigueur intellectuelle ont nourri en moi le goût d'apprendre, de comprendre et de persévérer. Une reconnaissance particulière va également à l'Ir. Koko Katumbi Pascal, bien qu'il n'appartienne pas à notre département, pour l'aide précieuse qu'il nous a apportée dans notre parcours.

À ma famille, je témoigne toute ma reconnaissance pour son amour inconditionnel, et plus particulièrement à mon père, M. Mudahama Sébastien, et à ma très chère mère, Mme Mauwa Marie-Claire, pour leurs sacrifices, soutien indéfectible et foi en moi. Ils ont été mes piliers solides.

Enfin, à mes collègues de promotion et à mes amis proches, je dis merci pour les échanges enrichissants, la camaraderie, les encouragements et tous les moments inoubliables partagés ensemble.

ISHARA MUDAHAMA Frédérick

Résumé

Ce mémoire s'inscrit dans une démarche de modernisation des services administratifs universitaires, en proposant la conception d'un système informatique baptisé **UniBank**, destiné à automatiser l'enregistrement des bordereaux de paiement et à simuler les paiements via Airtel Money, en connectant les services de la banque EquityBCDC à ceux de l'Université Libre des Pays des Grands Lacs (ULPGL-Goma). Face aux limites des procédures manuelles (files d'attente, pertes de justificatifs, absence d'intégration), cette solution, reposant sur une architecture microservices, intègre trois modules (celui pour la gestion d'utilisateurs, la gestion de paiements et la gestion de bordereaux) et s'appuie sur des technologies telles que **Spring Boot**, **Flutter**, **Vue.js**, **PostgreSQL**, **Kafka** et **MinIO**. Bien que certaines intégrations soient simulées, la structure est extensible et prête pour une mise en production réelle. Les résultats démontrent une amélioration notable de la fluidité du processus de paiement, tout en posant les bases d'une digitalisation complète, avec des perspectives futures comme l'usage de l'IA pour la reconnaissance automatique des bordereaux.

Mots-clés : bordereaux de paiement, paiement mobile, microservices, Spring Boot.

Abstract

This thesis is part of an effort to modernize university administrative services by designing an information system called *UniBank*, aimed at automating the registration of payment slips and simulating payments via Airtel Money. The system connects the services of EquityBCDC Bank's Cash Express with those of the Université Libre des Pays des Grands Lacs (ULPGL-Goma). In response to the limitations of manual procedures (queues, lost receipts, lack of integration), the proposed solution is built on a microservices architecture comprising three main modules: user management, payment processing, and slip management. It leverages technologies such as Spring Boot, Flutter, Vue.js, PostgreSQL, Kafka, and MinIO. While some integrations are simulated, the system is designed to be extensible and ready for real-world deployment. The results demonstrate a significant improvement in the efficiency of the payment process and lay the groundwork for a fully digitalized academic fee management system, with future prospects including the integration of artificial intelligence for automatic slip recognition.

Keywords: payment slips, mobile payment, microservices, Spring Boot.

Table des matières

Liste des figures

Liste des tableaux

Sigles et abréviations

ULPGL	:	Université Libre des Pays des Grands Lacs
SI	:	Système d'Informations
URL	:	U
JWT	:	Json Web Token
API	:	Application Programming Interface
DDD	:	Domain Driven Design
ERP	:	Enterprise Resource Planning
NFC	:	Near Field Communication
PSP	:	Prestataire de Service de Paiement
REST	:	Representational State Transfer
SGBDR	:	Système de Gestion de Base de Données Relationnelles
SGI	:	Système de Gestion d'Information
SOLID	:	**S**ingle responsibility, **O**pen/closed, **L**iskov substitution, **I**nterface segregation, **D**ependency inversion
SRP	:	Single Responsibility Principle
UAT	:	University Administrative Tools
UML	:	Unified Modelling Language

0. Introduction générale

0.1. Contexte et Généralités sur le thème

Dans l'accomplissement de ses diverses tâches, l'homme poursuit toujours un ou plusieurs objectifs. Dans la plupart de cas, il cherche à répondre à un besoin de son existence. Dans la poursuite de ces objectifs, l'homme met en place des outils, des méthodes et des stratégies qui lui permettent de réduire l'effort tout en atteignant ce qu'il s'est fixé. L'informatique est l'un de ces outils, elle permet d'automatiser certaines tâches que l'homme faisait en dépensant d'énormes efforts, et même des tâches qui, il y a quelques années, étaient considérées comme impossibles. [1]

Avec le développement de l'informatique, l'homme est capable d'effectuer des travaux à distance, chose qui auparavant nécessitait la présence physique d'une personne. Aujourd'hui il est aussi possible de faire des interventions dans des milieux à haut risque grâce au développement de l'informatique [1]. Réfléchissant sur les possibilités et opportunités que nous offre l'informatique, nous avons pensé à rendre plus optimale la gestion des paiements des frais au sein de l'Université Libre Des Pays des Grands Lacs.

Il y a quelques années encore, avant que la plupart de nos universités n'intègrent des systèmes informatisés pour la gestion de leurs différentes tâches, ces dernières étaient accomplies de manière laborieuse et archaïque. Heureusement, avec l'essor du numérique, ces méthodes fastidieuses tendent à disparaître progressivement, remplacées par des solutions de gestion plus performantes et adaptées aux besoins actuels. Dans cette dynamique, l'émergence des services de paiement mobile tels qu'Airtel Money ou M-Pesa ouvre la voie à des nouvelles possibilités. Il devient envisageable, par exemple, de proposer aux étudiants une alternative moderne pour le règlement de leurs frais académiques à distance, réduisant ainsi la dépendance aux guichets physiques et simplifiant les procédures administratives.

Malgré les avancées qu'a apportées le numérique dans le domaine du paiement, des défis subsistent, notamment en ce qui concerne la centralisation automatique des données

entre la banque et l'université. Cette lacune complique le processus de paiement et l'enregistrement des bordereaux, pénalisant ainsi les étudiants. C'est dans cette optique que, par notre travail, nous apportons une modique contribution à l'amélioration de la gestion des tâches administratives au sein de notre université.

0.2. Identification et formulation du problème

Comme mentionné précédemment, l'ULPGL-Goma fait partie des universités qui ont intégré des systèmes informatisés pour la gestion de leurs différentes tâches administratives, son principal outil étant l'UAT (University Administrative Tools), système qui prend en charge plusieurs modules, dont celui du paiement des frais académiques. Régulièrement, les paiements s'effectuent principalement aux guichets bancaires ou via Cash Express. Une fois le paiement effectué, l'étudiant doit se rendre à la comptabilité pour faire enregistrer son bordereau. Toutefois, cette procédure présente certaines limites :

0.2.1. Problèmes liés aux bordereaux papier

- Le personnel de la comptabilité qui effectue l'enregistrement des bordereaux se retrouve débordé et les étudiants se retrouvent obligés de faire la queue pour pouvoir enregistrer les bordereaux, ce qui entraine des longues durées d'attente dépendant de l'effectif des étudiants dans la file d'attente.
- Après paiement au cash express, l'étudiant reçoit un bordereau contenant les informations relatives à son paiement. Il peut arriver que le bordereau soit perdu ou son contenu soit effacé avant son enregistrement. L'étudiant devra alors faire la commande d'un duplicata de bordereau moyennant une amende de 2$.
- Après paiement au cash express, une raison pertinente peut être à la base du non enregistrement du bordereau à l'université par l'étudiant. Dans ce cas, rien ne prouve le payement de l'étudiant au niveau de l'Université. C'est qui est très désavantageux pour lui dans des situations de recouvrement par exclusion aux cours et/ou évaluations.

0.2.2. Absence d'alternative moderne pour les paiements directs

Les étudiants doivent passer par des guichets physiques (banque ou Cash Express), ce qui peut être contraignant en termes de temps et d'efforts.

0.3. Questions de recherche

Au vu des différents problèmes ci-hauts présentés nous formulons ces questions qui pourront nous guider dans cette recherche.

- Comment une conception et l'implémentation d'un système informatique de gestion des paiements reliant le Cash Express et l'université, et permettant aux étudiant un paiement direct des frais par Mobile Money, pourraient-elles offrir des solutions palliatives aux problèmes susmentionnés ?
- Quels sont les moyens technologiques adéquats pour établir un tel système ?
- Comment garantir la sécurité et la confidentialité des données dans un tel système ?

0.4. Formulation des hypothèses

A la suite des questions soulevées, nous nous proposons pour hypothèses que :

- La conception et l'implémentation d'une application combinant une interface mobile pour scanner les bordereaux de paiement et transmettre automatiquement les informations de paiement à la comptabilité, avec une interface web permettant à la fois l'accès à ces informations et le paiement direct des frais par les étudiants, amélioreraient l'efficacité du processus de paiement en établissant un lien direct entre le service Cash Express et l'université.
- L'utilisation de Spring Boot pour le backend, Flutter pour l'interface mobile, VueJs pour l'interface web, PostgreSQL pour la base de données et SerdiPay comme plateforme de paiement (PSP) constituerait une combinaison technologique optimale pour le développement d'un système de gestion des paiements académiques. Cette approche assurerait une performance optimale, une sécurité

renforcée, une expérience utilisateur fluide et une intégration transparente des différents modules, tout en permettant une évolutivité future du système.

- Avec l'implémentation de protocoles de cryptage (pour les mots de passe par exemple) et d'authentification, l'application garantirait la sécurité et la confidentialité des données transmises.

0.5. Justification du choix du sujet et motivations

Le paiement des frais académiques est l'une obligation pour les étudiants au sein de l'institution. Cependant, le processus actuel présente certaines lacunes, notamment lorsqu'il faut passer par la banque pour régler les frais, comme c'était déjà exposé précédemment. L'idéal serait d'automatiser l'enregistrement des bordereaux papier tout en intégrant des solutions de paiement mobile.

C'est dans cette optique que nous avons choisi de développer un système permettant d'enregistrer automatiquement les bordereaux et de réaliser le paiement des frais via Airtel Money.

La mise en place d'un tel système présente plusieurs avantages, notamment :

- Offrir aux étudiants une expérience de paiement simplifiée et accessible, adaptée aux contraintes actuelles.
- Une gestion plus fluide et transparente, grâce à la transmission immédiate des informations après paiement, facilitant ainsi le contrôle lors des opérations de recouvrement, un aspect bénéfique tant pour l'étudiant que pour l'administration.

0.6. Énoncé des objectifs de recherche

0.6.1. L'objectif général

Ce travail a pour objectif principal de concevoir et de réaliser un système (une application web et mobile) permettant l'enregistrement automatique des bordereaux de paiement, ainsi que le paiement mobile, en utilisant Airtel Money pour effectuer un transfert vers EquityBCDC. Bien que cette fonctionnalité de paiement réel ne soit pas encore

opérationnelle en raison des contraintes liées aux exigences de notre prestataire de service de paiement, une simulation visera à démontrer le processus et à préparer l'intégration future des solutions de paiement mobile adaptées aux conditions locales.

0.6.2. Les objectifs opérationnels/spécifiques

Pour arriver à bout de notre projet nous comptons :

- ✓ Identifier et documenter les exigences fonctionnelles et non fonctionnelles du système, en prenant en compte les besoins des utilisateurs et les contraintes techniques ;
- ✓ Concevoir une architecture modulaire et évolutive du système, assurant ainsi sa maintenance à long terme et sa capacité à s'adapter aux futures extensions ;
- ✓ Concevoir une base de données relationnelle optimisée pour le stockage et la gestion des informations des bordereaux de paiement, garantissant une efficacité maximale et une intégrité des données ;
- ✓ Implémenter le module de scan permettant de scanner les bordereaux de paiement avec une haute précision et une grande rapidité ;
- ✓ Intégrer une solution de paiement mobile, simulant le paiement des frais académiques via la plateforme SerdiPay, afin de faciliter les paiements à distance et réduire la dépendance aux guichets physiques ;
- ✓ Mettre en place un système de gestion des utilisateurs avec des niveaux d'accès et des permissions spécifiques, garantissant ainsi la sécurité et la confidentialité des informations traitées par l'application ;
- ✓ Créer une interface utilisateur intuitive et conviviale, facilitant l'utilisation du système par les utilisateurs.

0.7. Méthodologie et délimitation du travail

Pour atteindre les objectifs de notre projet, nous adopterons une approche méthodologique combinée. Tout d'abord, une méthode documentaire sera utilisée pour explorer les techniques et technologies couramment employées dans des systèmes

similaires. Cette phase nous permettra également d'identifier les meilleures pratiques et les améliorations possibles pour notre système.

Ensuite, pour la conception du système, nous adopterons une méthode expérimentale [2]. Cette approche permettra de construire progressivement le système, avec des cycles de tests réguliers pour évaluer ses fonctionnalités et ajuster les solutions en fonction des retours obtenus. Les tests réguliers nous permettront de valider les progrès du projet et de nous assurer que le système répond efficacement aux besoins identifiés.

Enfin, le périmètre de notre travail sera limité la gestion et l'enregistrement des bordereaux de paiement issus du Cash Express de la Banque EquityBCDC. Nous nous concentrerons principalement sur la simulation de paiements via Airtel Money, en tenant compte des solutions disponibles localement.

0.8. Structure du mémoire/ Subdivision du travail

Hormis l'introduction générale et la conclusion du travail, notre travail est subdivisé en trois chapitres :

1. Au premier chapitre, ***Concepts théoriques de base***, nous passons en revue toute la théorie nécessaire pour la compréhension de notre travail.
2. Au deuxième chapitre, ***Conception et Développement du Système***, nous allons détailler tout le processus de conception et de développement de notre système.
3. Au troisième chapitre, ***implémentation du Système UniBank***, qui est la phase finale de notre projet, nous allons présenter et expliquer le rendu de notre travail en nous basant sur la conception faite, puis nous présentons les résultats en capturant quelques interfaces utilisateurs du système.

Chapitre 1
Concepts théoriques de base

Dans un monde en constante évolution numérique, la gestion des paiements est un enjeu fondamental pour les institutions financières et les établissements d'enseignement. L'intégration de solutions modernes dans le traitement des transactions permettrait d'améliorer la rapidité, la sécurité et la traçabilité des paiements. Dans le cadre d'un système reliant les cash express d'une banque et une université, il est essentiel de comprendre les mécanismes qui sous-tendent ces échanges financiers.

Ce chapitre présente d'abord les concepts théoriques de base relatifs aux systèmes de gestion de l'information. Il aborde ensuite les paiements et les bordereaux bancaires, qui jouent un rôle clé dans la gestion des transactions physiques. Il explore par après le paiement en ligne, une alternative de plus en plus adoptée pour simplifier les opérations financières. Enfin, le choix du Prestataire de Services de Paiement (PSP) sera abordé, en mettant en avant son rôle central dans la sécurisation et l'optimisation des transactions.

1.1 Systèmes de gestion d'information

Les systèmes de gestion d'information (SGI) constituent des outils fondamentaux dans le fonctionnement des organisations modernes, en particulier dans le secteur de l'éducation supérieure. Ils permettent de collecter, stocker, traiter et diffuser des informations nécessaires à la planification, au contrôle et à la prise de décision au sein d'une institution. [3]

Dans un contexte universitaire, un SGI vise à coordonner divers services tels que la gestion pédagogique, administrative et financière afin d'assurer une cohérence dans le suivi des étudiants, des enseignants, des ressources matérielles et des transactions financières.

Les SGI s'inscrivent dans une logique d'**automatisation** et de **rationalisation** des processus. Ils jouent un rôle central dans l'amélioration de l'efficience organisationnelle et de la qualité des services rendus aux usagers [3]. En centralisant les données et en facilitant leur accès structuré et sécurisé, ils réduisent considérablement les redondances, les erreurs humaines, et les lenteurs administratives. [4]

Structure et fonctionnement des SGI :

Traditionnellement, les SGI sont construits selon une architecture **centralisée**, comprenant trois couches principales [5] :

- **La couche de présentation**, qui constitue l'interface utilisateur via une application web, mobile ou desktop ;
- **La couche logique**, qui intègre les règles métiers et la gestion des processus internes ;
- **La couche de données**, généralement assurée par un système de gestion de bases de données relationnelles (SGBDR), centralisant toutes les informations de l'institution.

Ce modèle centralisé a longtemps dominé les systèmes d'information universitaires, en particulier dans les ERP classiques, car il permet une vision globale et intégrée de l'ensemble des activités de l'établissement.

Cependant, avec l'évolution des technologies et des exigences de **scalabilité**, de **flexibilité** et de **résilience**, une tendance claire se dessine vers des architectures **décentralisées**, notamment les **architectures orientées microservices**. Contrairement aux systèmes monolithiques, un SGI basé sur les microservices est composé d'un ensemble de services indépendants, chacun responsable d'un domaine métier précis par exemple : la gestion des inscriptions, la gestion des paiements, ou la gestion des ressources humaines. [6]

Chaque microservice dispose de sa **propre base de données**, ce qui garantit une autonomie complète, facilite la maintenance, et permet un déploiement indépendant de chaque composant. [7]

Cette approche permet de mieux répondre aux besoins dynamiques d'une université moderne, notamment en matière d'intégration de services tiers (paiement mobile, authentification fédérée, services cloud), et d'amélioration continue. En revanche, elle nécessite une gestion rigoureuse de la **cohérence interservices**, souvent assurée par des mécanismes asynchrones comme les files de messages ou les événements métier. [8]

Il est donc fondamental de comprendre à la fois les **fondements conceptuels** des SGI et leurs **formes d'implémentation moderne**, pour garantir une intégration fluide, cohérente et alignée avec les meilleures pratiques du développement logiciel actuel.

1.2 Paiements et bordereaux bancaires

Les paiements bancaires jouent un rôle essentiel dans les transactions financières, permettant aux particuliers et aux entreprises de transférer des fonds de manière sécurisée et efficace. Pour assurer la traçabilité et la gestion de ces opérations, les institutions financières utilisent des documents spécifiques, appelés bordereaux bancaires. Ces derniers servent à formaliser et à enregistrer les différentes formes de paiements effectués via le système bancaire.

1.2.1 Paiements bancaires

Un paiement bancaire est une transaction financière permettant de transférer des fonds d'un compte à un autre, généralement effectuée au sein d'une institution financière. Ces paiements sont essentiels pour le commerce, les services et les transactions quotidiennes.

Voici un aperçu des principaux moyens de paiement bancaires : [9] [10] [11]

- ***Carte bancaire*** : Un instrument de paiement physique ou virtuel permettant au titulaire d'un compte de réaliser des achats ou des retraits. Les cartes peuvent être de débit (débit immédiat ou différé), de crédit (avec réserve d'argent) ou prépayées (rechargeables).

Figure 1:carte bancaire Equity BCDC

Le principal avantage des cartes de crédit par exemple, est leur facilité d'utilisation pour les paiements en ligne, accessibles partout dans le monde et instantanément. Elles ne nécessitent aucun matériel ou logiciel spécifique. L'authentification repose sur le numéro de carte, le nom du titulaire et la date d'expiration. Pour renforcer la

sécurité des transactions, des systèmes comme Verified by Visa et MasterCard SecureCode ont été développés.

- ***Virement bancaire*** : Un transfert électronique de fonds d'un compte à un autre, souvent utilisé pour les paiements récurrents ou les transactions entre entreprises.
- ***Prélèvement automatique*** : Une autorisation donnée à une entreprise ou institution pour débiter automatiquement des fonds d'un compte bancaire à des dates régulières, couramment utilisé pour les factures récurrentes.
- ***Chèque*** : Document écrit par lequel une personne (le tireur) ordonne à sa banque de payer une somme déterminée à une autre personne (le bénéficiaire).

Figure 2: chèque bancaire Equity BCDC

- ***Portefeuille électronique (e-wallet)*** : Application ou service en ligne permettant de stocker des informations de paiement et d'effectuer des transactions électroniques via des dispositifs électroniques.
- ***Paiement mobile sans contact*** : Les paiements mobiles permettent d'effectuer des transactions via un smartphone ou une tablette, sans nécessiter de carte bancaire physique. Ils utilisent différentes technologies comme le **Near Field Communication (NFC)** pour les paiements sans contact, les **QR codes**, ou encore les applications bancaires dédiées. Des services comme **Apple Pay, Google Pay et MyAirtel** facilitent ces transactions en stockant les informations de paiement dans un environnement sécurisé. Cette méthode est rapide, pratique et de plus en plus adoptée par les commerçants.

- ***Monnaie électronique*** : La monnaie électronique désigne les valeurs monétaires stockées sous forme électronique et acceptées comme moyen de paiement par des entreprises ou des particuliers. Elle peut être utilisée pour des transactions en ligne ou hors ligne, sans passer par un compte bancaire traditionnel. Elle est généralement émise et régulée par une institution financière (ex. banques, prestataires de services de paiement).

Bien que cette liste ne soit exhaustive, elle regroupe les principaux moyens de paiement utilisé. En ce qui concerne nos paiements des frais académiques via les services de cash express, il s'agit bien des *virements bancaires* après que l'on ait déposé l'argent auprès de l'agent, on en est arrivé à cette conclusion après notre échange avec une des agents du cash express postée au niveau du Campus Salomon, mais également en nous basant sur les informations présentées ci-dessus.

1.2.2 Bordereaux bancaires

Un bordereau bancaire est un document officiel utilisé pour enregistrer et justifier une transaction financière effectuée auprès d'un établissement bancaire. Il peut s'agir d'un dépôt, d'un retrait, d'un virement ou de tout autre type d'opération bancaire. Ce document est généralement fourni par la banque ou généré par un système de gestion financière et doit être rempli avec précision afin d'assurer une bonne traçabilité des flux monétaires.
Particulièrement pour les dépôts de frais académiques au niveau du cash express, le bordereau qui est généré et remis est représenté par la *figure 3* :

Figure 3: bordereau de paiement equityBCDC

1.3 Le paiement en ligne

L'avènement du commerce électronique et la croissance d'Internet ont favorisé la numérisation du processus de paiement avec la fourniture de diverses méthodes de paiement en ligne telles que l'argent électronique, les cartes de débit, les cartes de crédit, le paiement sans contact, les portefeuilles mobiles, etc. En outre, les services fournis par le paiement mobile gagnent en popularité de jour en jour et montrent une transition en avançant vers un avenir propice de perspectives spéculatives en conjonction avec les innovations technologiques. [11]

1.3.1 Définitions

a. ***Paiement en ligne*** : est un arrangement d'échange monétaire entre les acheteurs et les vendeurs sur des conditions en ligne qui est facilité par un instrument financier

numérique (par exemple, des chèques électroniques, des numéros de carte de crédit codés ou de l'argent liquide sous forme numérique) soutenu par une banque, un médiateur ou un associé légitime. [11]

b. ***Monnaie électronique :*** correspond à une valeur monétaire qui est stockée sous une forme électronique (carte prépayée, le portefeuille électronique ou numérique appelé aussi e-wallet ...) émise contre la remise de fonds, et pouvant etre utilisée à des fins d'opération de paiement. [12]

c. ***Passerelle de paiement*** : En pratique, les passerelles de paiement servent de lien entre les organismes financiers responsables de l'échange d'argent et le site web du vendeur. [11]

1.3.2 Les acteurs du paiement en ligne [12]

a. ***Le client*** : il achète des biens et/ou services à partir du site d'un commerçant. L'utilisateur utilise un matériel informatique (exemple : ordinateur, etc.) connecté au réseau pour pouvoir effectuer des paiements.

b. ***Le commerçant*** : il vend des produits qui peuvent être soit des biens et/ou des services à des consommateurs qui les achètent à distance à travers un réseau.

c. ***Le prestataire de services de paiement (PSP)*** : comme va le montrer la *figure 2,* différentes factions sont incluses dans le processus de paiement en ligne. Une passerelle de paiement électronique est un élément fondamental pour les transactions en ligne et est censée garantir au client que l'échange est fiable et sûr dans tous les aspects de la sécurité. Exemples de PSP : Stripe, PayPal, Paystack, Flutterwave, CinetPay, etc.

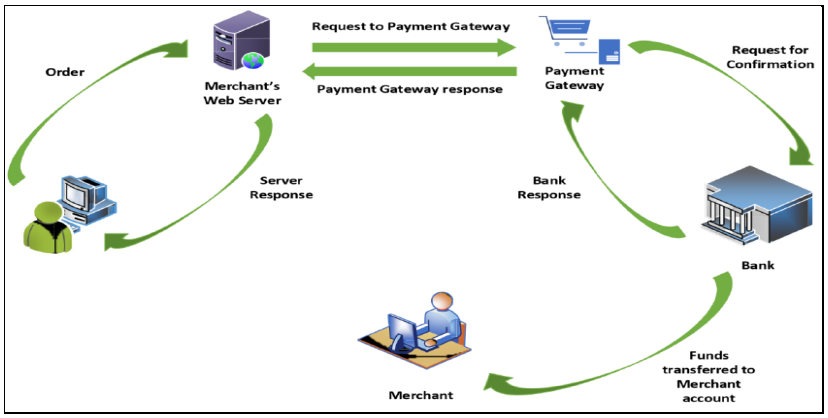


Figure 4 : les grandes étapes d'un paiement en ligne

d. ***La banque acquéreuse*** : c'est la banque du marchant, banque qui traite avec d'autres banques comme les banques du client par exemple, des maisons de télécommunication ou même avec des prestataires de service de paiement.

e. ***La banque émettrice*** : il s'agira de la banque du client, où l'argent sera prélevé au compte bancaire du client qui souhaite payer.

Dans notre cas il ne sera pas question d'une banque mais plutôt d'une maison de télécommunication précisément Airtel**.**

1.3.3 Inconvénients des solutions de paiement en ligne [14]

Les solutions de paiement en ligne présentent plusieurs avantages, cependant, comme toute technologie, elles présentent également des défauts et des défis à prendre en considération.

- ***Frais de transaction*** : Les prestataires facturent des frais sur chaque transaction, ce qui peut peser sur les entreprises, surtout celles avec de faibles marges. Comparer les offres et comprendre la tarification permet de choisir la meilleure option. Certains fournisseurs proposent des tarifs avantageux pour les petites entreprises et associations.
- ***Fraude et cybercriminalité*** : Les paiements en ligne sont exposés aux cyberattaques. Les entreprises doivent sécuriser les données (chiffrement, authentification à deux facteurs) et former leur personnel aux risques. La lutte contre la fraude implique les entreprises, les prestataires et les clients.
- ***Problèmes techniques et compatibilité*** : L'intégration des solutions de paiement peut être complexe. Des bugs ou interruptions peuvent affecter les paiements et l'expérience client. Choisir un prestataire fiable et tester la solution avant son déploiement limite ces risques.
- ***Confidentialité et protection des données*** : Les entreprises doivent protéger les données clients (numéros de carte, coordonnées bancaires) et respecter les réglementations. Le non-respect peut entraîner des sanctions et nuire à leur réputation.
- ***Difficultés pour les petites entreprises*** : Les coûts, la complexité technique et la sécurité peuvent être des obstacles. Comparer les offres, choisir une

solution simple à intégrer et s'informer sur les bonnes pratiques permet de surmonter ces défis.

1.4 Conclusion partielle

Nous venons de réaliser une vue d'ensemble sur les systèmes de gestion d'information et ceux de paiement. On a également présenté les bordereaux bancaires, document qui sert de pièce justificative pour la transaction qui a eu lieu. Parlant de paiement en ligne on a eu à voir que bien que celui-ci présente des avantages il a aussi bon nombre de défauts, et donc l'on se doit d'être regardant avant même son intégration. On a fini en faisant un choix sur ce qui allait nous servir comme prestataire de service de paiement, et c'est SerdiPay qui a été choisi comme PSP, choix qui a été expliqué.

Sur la base de ces éléments théoriques présentés, le chapitre suivant s'intéresse à la conception de notre système. Il décrit de manière progressive et structurée les différentes étapes du développement, en partant de l'identification des besoins jusqu'à l'architecture finale basée sur les microservices.

Chapitre 2
Conception et développement du système de gestion de paiement

Ce chapitre expose le processus de développement du système proposé, mené de manière méthodique et progressive, chaque étape s'appuyant sur les résultats de la précédente. Nous commençons par l'identification des besoins fonctionnels et non fonctionnels, ce qui nous permet de les formaliser à l'aide des diagrammes des cas d'utilisation. Ensuite, nous adoptons une démarche de conception orientée services, en introduisant l'architecture à base de microservices ainsi que les raisons ayant motivé ce choix architectural. Nous procédons alors à l'identification des microservices requis et à la description détaillée de leur conception, en nous appuyant sur le langage UML pour modéliser les cas d'utilisation et les différents domaines. Enfin, nous présentons l'architecture globale de notre système, en mettant en évidence l'intégration et les interactions entre les microservices.

2.1 Identification de besoins

Dans cette partie nous allons présenter les différents besoins logiciels du système proposé, besoins issus de nos analyses, mais également de nos échanges avec différents agents de l'université, ensuite nous allons exprimer de manière formelle ces besoins en utilisant les diagrammes de cas d'utilisation.

2.1.1 Présentation de la solution proposée

Le système que l'on cherche à développer vise dans un premier temps à faire un pont entre le service de cash express et le service de comptabilité de l'université, et dans un deuxième temps donner à l'étudiant la possibilité de payer ses différents frais par mobile money. Ce système assure la collecte des informations relatives aux bordereaux de paiement générés apres chaque paiement de l'étudiant au niveau du cash express. L'agent du cash express, au travers des interfaces conviviales, pourra scanner le bordereau de paiement et transmettre les informations liées au paiement au service de la comptabilité ou

celui du décanat pour d'autres fins. L'étudiant pourra aussi avoir un compte au sein du système, grâce auquel il pourra payer ses frais sans se présenter physiquement au niveau du cash express ni de la comptabilité.

2.1.2 Les spécifications fonctionnelles

Après quelques analyses, voici les spécifications fonctionnelles les plus importantes du futur système :

- ✓ Le système doit permettre à chaque utilisateur de s'authentifier.
- ✓ Le système doit permettre à l'agent du cash Express de pouvoir scanner un bordereau de paiement, après ce scan, pouvoir envoyer les informations liées au paiement ainsi que le fichier pdf contenant l'image du bordereau.
- ✓ Le système doit permettre à l'agent du décanat de pouvoir consulter les différents paiements des frais relatifs à sa faculté, ceux ayant été approuvés et enregistrés par le service de comptabilité ainsi que ceux ne l'étant pas encore.
- ✓ Le système doit permettre à l'agent du décanat de pouvoir imprimer une liste d'étudiant ayant payé, liste qui sera produite selon un filtre donné (le type de frais, la promotion, une plage de dates où le paiement aurait été effectué).
- ✓ Le système doit permettre à l'agent du décanat de pouvoir approuver et enregistrer tout paiement d'un autre type que le frais académique.
- ✓ Le système doit permettre à l'agent de la comptabilité de pouvoir consulter, approuver les paiements qui ont été sauvegardés par l'agent du cash Express, paiement du type frais académique uniquement.
- ✓ Le système doit permettre à l'agent de la comptabilité de pouvoir consulter tous les agents actifs utilisant le système, ou les bloquer si besoin, ceux ne l'étant plus (c'est-à-dire qui ont déjà été bloqués) mais aussi tous les étudiants utilisant le système.
- ✓ Le système doit permettre à l'étudiant de pouvoir consulter sa situation financière, mais également de payer ses frais.

2.1.3 Les spécifications non fonctionnelles

Une exigence non fonctionnelle définit l'attribut de qualité d'un système logiciel. Ils représentent un ensemble de normes utilisées pour juger du fonctionnement spécifique d'un

système. Les exigences non fonctionnelles sont essentielles pour garantir la convivialité et l'efficacité de l'ensemble du système logiciel. Ne pas répondre aux exigences non fonctionnelles peut entraîner des systèmes qui ne parviennent pas à satisfaire aux besoins des utilisateurs. [15]

Pour le système proposé ici nous avons identifié les spécifications non fonctionnelles suivantes :

- ✓ ***La sécurité*** **:** Les données des utilisateurs doivent être protégées contre les accès non autorisés.
- ✓ ***La scalabilité*** **:** Le système doit pouvoir être facilement adapté à l'augmentation des données, des fonctionnalités et des utilisateurs.
- ✓ ***L'utilisabilité*** **:** Le système doit disposer des interfaces utilisateurs claires et intuitives, avec une navigation cohérente.
- ✓ ***La maintenance*** **:** le système doit être facilement maintenable dans le temps, permettant une mise en œuvre continue des modifications futures sans perturbation majeure des autres sous-systèmes.

2.1.4 Les cas d'utilisations

Les cas d'utilisation sont une technique de capture des besoins fonctionnels du système, ils décrivent les interactions entre les utilisateurs du système et le système lui-même, ils sont une description "narrative" de comment est utilisé le système. Les cas d'utilisation sont basés sur des scénarios qui sont un séquencement des évènements se passant entre l'utilisateur et le système. [2]

2.1.4.1 Identification des acteurs du système

Les utilisateurs du système sont appelés « acteurs » ; un acteur identifie un rôle joué dans le système et pas une personne. [8] Partant des besoins spécifiés dans la section précédente nous pouvons retenir les acteurs suivants : **l'agent du cash express**, **l'agent du décanat**, **l'API de SerdiPay, l'API de UAT**, **l'étudiant** et **le comptable** (qui aura également le rôle d'administrateur du système).

2.1.4.2 Identification des cas d'utilisation

Reprenons un à un dans le *Tableau 1* les cas d'utilisations du système par acteur (les six acteurs identifiés précédemment) et définissons les différentes façons dont ils auront à utiliser le système.

Tableau 1: les cas d'utilisations du système

Acteurs	Cas d'utilisations
Agent du cash express	o Consulter tous ses bordereaux scannés o Scanner un bordereau o Apres scan, pouvoir renseigner et envoyer les informations du paiement
Agent du décanat	o Consulter tous les paiements pour sa faculté o Imprimer une liste de paiements (impression suivant un type de frais spécifique et/ou une certaine plage de date que l'agent choisi) o Valider un paiement d'un autre type que frais académique o Enregistrer un paiement d'un autre type que frais académique
Etudiant	o Pouvoir consulter toute sa situation financière o Pouvoir payer ses différents frais
API de UAT	o Charger et fournir les informations d'un utilisateur lors de la création de son compte (particulièrement les étudiants et agents de l'université) o Fournir les informations relatives à la situation financière de l'étudiant
API de SerdiPay	o Transférer le fond depuis le compte mobile money de l'étudiant vers le compte bancaire de l'université
Comptable	o Consulter tous les paiements du type frais académique o Valider un paiement o Enregistrer un paiement

	o Générer un matricule avec le quel un agent du cash express pourrait s'inscrire dans le système o Consulter tous les matricules disponibles o Consulter tous les utilisateurs du système (agents comme étudiants) o Bloquer l'accès au système à un utilisateur

Notons que ce n'est pas tout comme cas d'utilisations car un autre est engendré par la spécification non fonctionnelle relative à la sécurité. Comme la protection des données des utilisateurs contre les accès non autorisés est primordiale, chaque utilisateur pourra alors s'authentifier avant d'accéder aux fonctionnalités lui réservées dans le système.
C'est ainsi qu'en plus des cas d'utilisation ci-haut, nous pouvons augmenter : **S'authentifier** pour chaque utilisateur.

2.2 Choix de l'architecture et conception du système

L'architecture d'un programme ou d'un système informatique est la façon dont ses différentes parties sont organisées, avec leurs caractéristiques et la manière dont elles interagissent. [2]
Il est donc crucial de noter que l'architecture du système joue un rôle central dans la satisfaction de ses exigences de qualité de service (exigences non fonctionnelles). En effet, bien que les besoins fonctionnels puissent être réalisés avec n'importe quelle architecture, l'architecture choisi détermine dans quelle mesure le système répondra aux exigences de qualité telles que la scalabilité, la maintenabilité, l'utilisabilité et la sécurité.

2.2.1 Choix de l'architecture

2.2.1.1 Analyses des architectures possibles pour notre système

a. ***Architecture monolithique :*** L'architecture monolithique repose sur une seule base de code qui regroupe toutes les fonctionnalités du système. Ce type d'architecture est simple à mettre en place au début d'un projet et facilite le déploiement initial. Cependant, à mesure que l'application évolue, cette architecture devient difficile à

maintenir et à faire évoluer. Le couplage fort entre les différentes parties du code complique les mises à jour et limite la scalabilité horizontale. [16]

b. ***Architecture en couches (ou n-tiers) :*** C'est une architecture couramment utilisée, notamment dans les systèmes traditionnels. Elle sépare les responsabilités en plusieurs couches : présentation, logique métier, accès aux données, etc. Bien qu'elle permette une certaine séparation des préoccupations, cette approche reste souvent rigide dans les cas d'évolution rapide ou d'interopérabilité avec plusieurs interfaces (mobile, web, API externes). [16]

c. ***Architecture microservices :*** L'architecture microservices consiste à diviser l'application en une suite de services autonomes, chacun répondant à un domaine fonctionnel précis. Chaque microservice (qui est une application à part entière) est déployé indépendamment, dispose de sa propre base de données et communique avec les autres services via des API (souvent REST) ou par des messages (asynchrones). [7]

Même si les architectures monolithiques et en couches restent valables dans de nombreux contextes, les besoins spécifiques de notre système rendent l'approche microservices plus appropriée. La section suivante présente les raisons de ce choix.

2.2.1.2 Justification des microservices

Le choix d'une architecture microservices se justifie pleinement dans le cadre de ce projet pour plusieurs raisons : [7]

- ***Modularité et découplage fort*** : le système implique plusieurs acteurs (étudiant, agent cash express, agent décanat, agent comptabilité), chacun ayant des responsabilités précises. Une architecture microservices permet de créer des modules indépendants qui gèrent leurs propres logiques et données.
- ***Scalabilité*** : certains services, comme le service de paiement ou celui pour la gestion des utilisateurs (particulièrement pour le rôle *étudiant*), peuvent connaître une charge plus élevée que d'autres. Les microservices permettent de scaler indépendamment chaque service en fonction de la demande.
- ***Maintenance facilitée*** : chaque service peut être mis à jour, modifié ou redéployé sans impacter les autres, ce qui est idéal pour un système devant évoluer (ex. : ajout

de nouveaux moyens de paiement à l'avenir, analyse et validation des bordereaux avant enregistrement de ceux-ci soit par des modèles d'apprentissage automatique, etc.).

- ***Sécurité renforcée*** : grâce à des mécanismes de contrôle d'accès au niveau de chaque service, il est possible de mieux compartimenter les privilèges et les données sensibles (ex. : approbation des paiements, accès aux identités).

En plus de ses nombreux avantages techniques, notre choix s'est également orienté vers cette architecture dans une optique d'apprentissage, afin de consolider nos compétences et de valoriser notre démarche en matière d'ingénierie logicielle.

2.2.1.3 Vue d'ensemble de l'architecture

L'architecture du système repose sur une approche **microservices**, où chaque fonctionnalité clé de l'application est implémentée sous forme de service autonome, interagissant avec les autres via des API.

La *figure 3* présente un aperçu global de l'architecture envisagée.

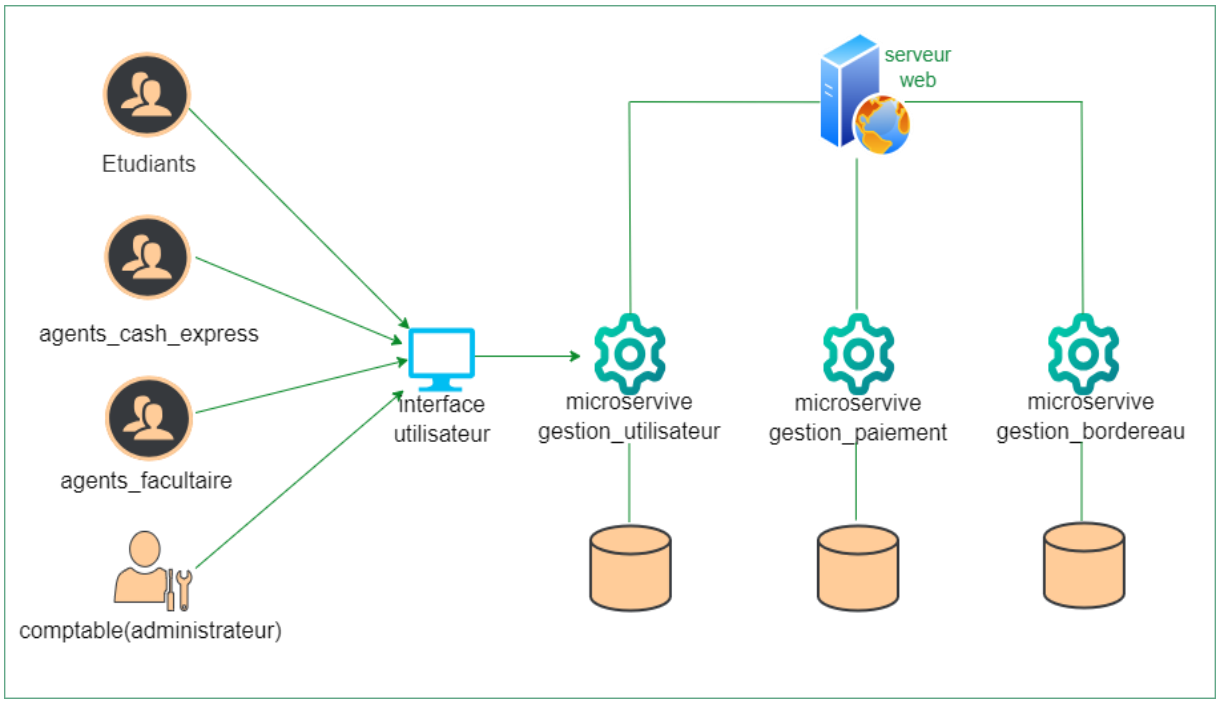


Figure 5 : vue d'ensemble sur l'architecture du système

2.2.2 Découpage du système en microservices

L'un des défis de l'utilisation de l'architecture de microservices est qu'il n'existe pas d'algorithme concret et bien défini pour décomposer un système en services. Comme pour la plupart des développements de logiciels, c'est une sorte d'art. Pour aggraver les choses, si on décompose un système de manière incorrecte, on construit un *monolithe distribué*, un système composé de services couplés qui doivent être déployés ensemble. [7]

2.2.2.1 Principes de découpage

Le découpage en microservices repose sur plusieurs principes d'architecture logicielle destinés à garantir la clarté, la modularité et la maintenabilité du système. Dans notre cas, nous avons défini **trois microservices** : users_mgt, payment_mgt et slips_mgt. Ce choix est principalement guidé par les concepts du *Domain-Driven Design (DDD)*, les *principes SOLID*, et les bonnes pratiques de *l'architecture microservices*.

- ***Le Bounded Context (DDD)***

Selon Eric Evans [17], *un bounded context* représente une frontière claire dans laquelle un modèle métier donné est défini et cohérent. Dans notre application :

- ✓ users_mgt : gère exclusivement les utilisateurs et leurs rôles,
- ✓ payment_mgt : traite les transactions financières,
- ✓ slips_mgt : s'occupe des bordereaux justificatifs.

Ce découpage évite le chevauchement des responsabilités et permet de faire évoluer chaque domaine de manière autonome.

- ***Responsabilité unique (SRP)***

Conformément au *principe de responsabilité unique* de Robert C. Martin [18], chaque microservice est responsable d'un seul aspect métier, ce qui facilite la maintenance et réduit les impacts lors de modifications.

- ***Faible couplage, forte cohésion***

Nous avons veillé à ce que chaque service soit fortement cohérent, c'est-à-dire que ses fonctions soient étroitement liées autour d'une même responsabilité, et faiblement couplé, en limitant ses dépendances aux autres services via des interfaces API bien définies. Cette approche s'inscrit dans les bonnes pratiques recommandées par Sam Newman [6], et favorise la maintenabilité, l'évolutivité et la résilience du système.

- ***Autonomie des services***

Chaque microservice dispose de sa **propre base de données**, conformément aux recommandations de Martin Fowler [19]. Cela garantit leur indépendance et réduit les risques de conflits d'accès aux données.

2.2.2.2 Description des microservices

A présent que nous avons découpé notre système et identifié les différents microservices, définissons maintenant dans le *Tableau 2* les différentes responsabilités pour chaque micorservice.

Tableau 2 : Définition des responsabilités

Microservices	Responsabilités (besoins fonctionnels)
User_mgt	o Créer un compte o Blocage d'accès o Générer matricule o Gérer son compte
Payment_mgt	o Enregistrement d'un paiement associé à chaque bordereau o Consulter tous les paiements o Charger les informations relatives à la situation financière de l'étudiant o Initier un paiement en ligne o Valider un paiement
Slips_mgt	o Enregistrer une pièce justificative pour tout paiement o Consulter tous les bordereaux o Générer un bordereau pour un paiement qui s'est effectué en ligne

2.2.2.3 Communication inter-services

Dans une architecture basée sur les microservices, la manière dont les services interagissent entre eux est cruciale pour garantir la cohérence, la performance et la résilience du système.

a. Patterns de communication inter-services

Plusieurs *design patterns* ont émergé pour faciliter les échanges efficaces et résilients entre microservices. Voici les principaux retenus dans le cadre du système proposé :

- ***Service Discovery***

Dans un environnement distribué, les services sont souvent dynamiques et scalables. Le pattern *Service Discovery* permet à un service client de localiser dynamiquement l'adresse du service cible via un registre (ex. : **Eureka**, **Consul**). [6]
Ce mécanisme évite de coder en dur les URLs et facilite l'ajout/retrait de services à la volée.

- ***Event-Driven Communication (Kafka/Event Bus)***

Ce pattern repose sur une architecture pilotée par les événements, dans laquelle les microservices communiquent via des messages publiés dans un bus (ex. : **Kafka**, **RabbitMQ**). Cela permet un découplage fort entre producteurs et consommateurs. [20]
Utile pour les opérations asynchrones comme celles impliquant la participation de plusieurs services.

b. Approche adoptée pour notre système

Pour notre système, la communication inter-services repose sur une approche hybride et bien orchestrée autour des patterns ci-dessus :

- ✓ Un Service Discovery via ***Eureka*** permettra aux microservices users_mgt, payments_mgt et slips_mgt de s'enregistrer et de se découvrir dynamiquement.
- ✓ Les interactions critiques et immédiates (ex. : vérification d'un utilisateur avant paiement) seront effectuées via des **API REST synchrones**, avec une gestion des erreurs via des *Circuit Breakers* (ex. : Resilience4j).
- ✓ Les opérations déclenchées par événements (ex. : notification d'un paiement enregistré) seront gérées via ***Kafka***, suivant un modèle ***event-driven***.

Cette combinaison permet d'assurer **robustesse**, **scalabilité**, et **souplesse** dans la gestion des échanges entre services.

2.2.3 Conception logique du système

Cette section s'intéresse en premier temps en une présentation brève du langage de modélisation UML, apres quoi vient la conception proprement dite de notre système.
La démarche adoptée s'appuie sur une méthodologie de conception par service, dans laquelle chaque composant est conçu et modélisé de manière isolée, tout en garantissant son intégration harmonieuse dans l'ensemble du système. Cette approche contribue à une gestion plus efficace de la complexité, à une meilleure résilience, ainsi qu'à une évolutivité accrue. Pour formaliser cette conception, le langage de modélisation UML sera utilisé, à travers l'élaboration de diagrammes dédiés à la représentation structurelle et comportementale de chaque microservice.

2.2.3.1 Présentation du langage UML [21]

- **Définition**

Le langage UML (Unified Modeling Language) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre des systèmes logiciels complexes par leur structure aussi bien que leur comportement.

- **Types des diagrammes UML**

L'UML utilise des éléments et les associe de différentes manières pour former des diagrammes qui représentent les aspects statiques ou structurels d'un système, ainsi que des diagrammes comportementaux qui capturent les aspects dynamiques d'un système.
Dans la suite, nous allons utiliser les diagrammes suivants pour la modélisation de chaque microservice :

- ***Le diagramme de cas d'utilisation :*** Pour représenter les acteurs du système et leurs interactions avec ce dernier.
- ***Le diagramme de séquence :*** Sert à illustrer la chronologie des échanges entre les objets du système, en précisant comment ils collaborent et dans quel ordre les messages sont échangés pour accomplir une fonctionnalité spécifique

- ***Le diagramme de classe :*** Pour explorer les concepts de différents sous domaines du système (les microservices).

2.2.3.2 Le microservice « User_mgt »

Le microservice ***user_mgt*** est responsable de la **gestion des utilisateurs** au sein du système. Il prend en charge les opérations comme celles liées à l'enregistrement, la génération des matricules par l'administrateur pour les agents du cash express, la gestion des profils des utilisateurs. Ce service centralise également les rôles, les permissions et les informations personnelles, garantissant ainsi une **séparation des responsabilités** et une **sécurité renforcée**. Il expose une API REST permettant aux autres microservices de vérifier l'identité des utilisateurs ou de récupérer certaines informations nécessaires, tout en assurant la **confidentialité** et la **cohérence des données** utilisateur dans l'ensemble du système.

- **Diagramme de cas d'utilisation**

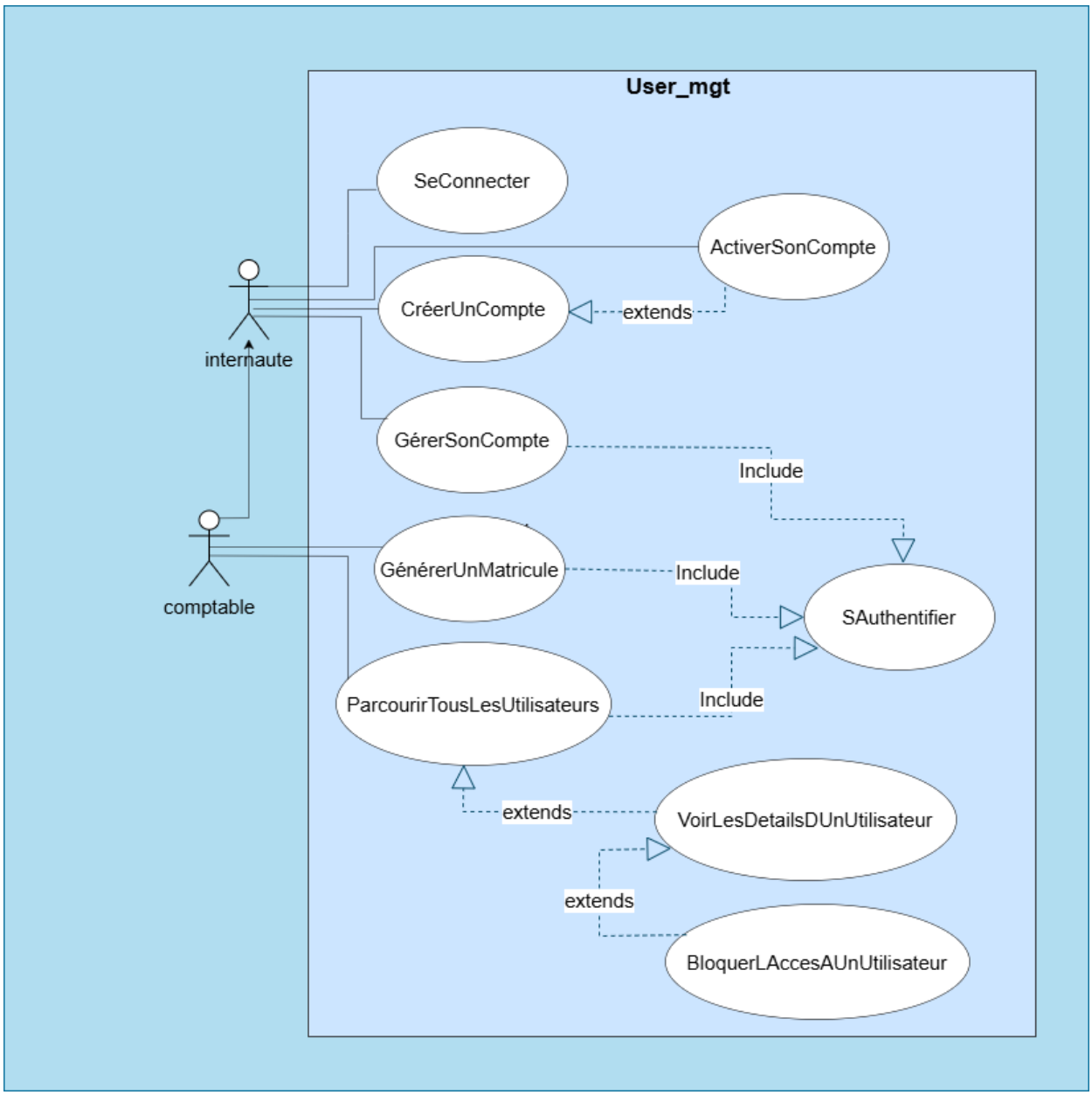


Figure 6:Digramme de cas d'utilisation du microservice user_mgt

- **Digramme de classe**

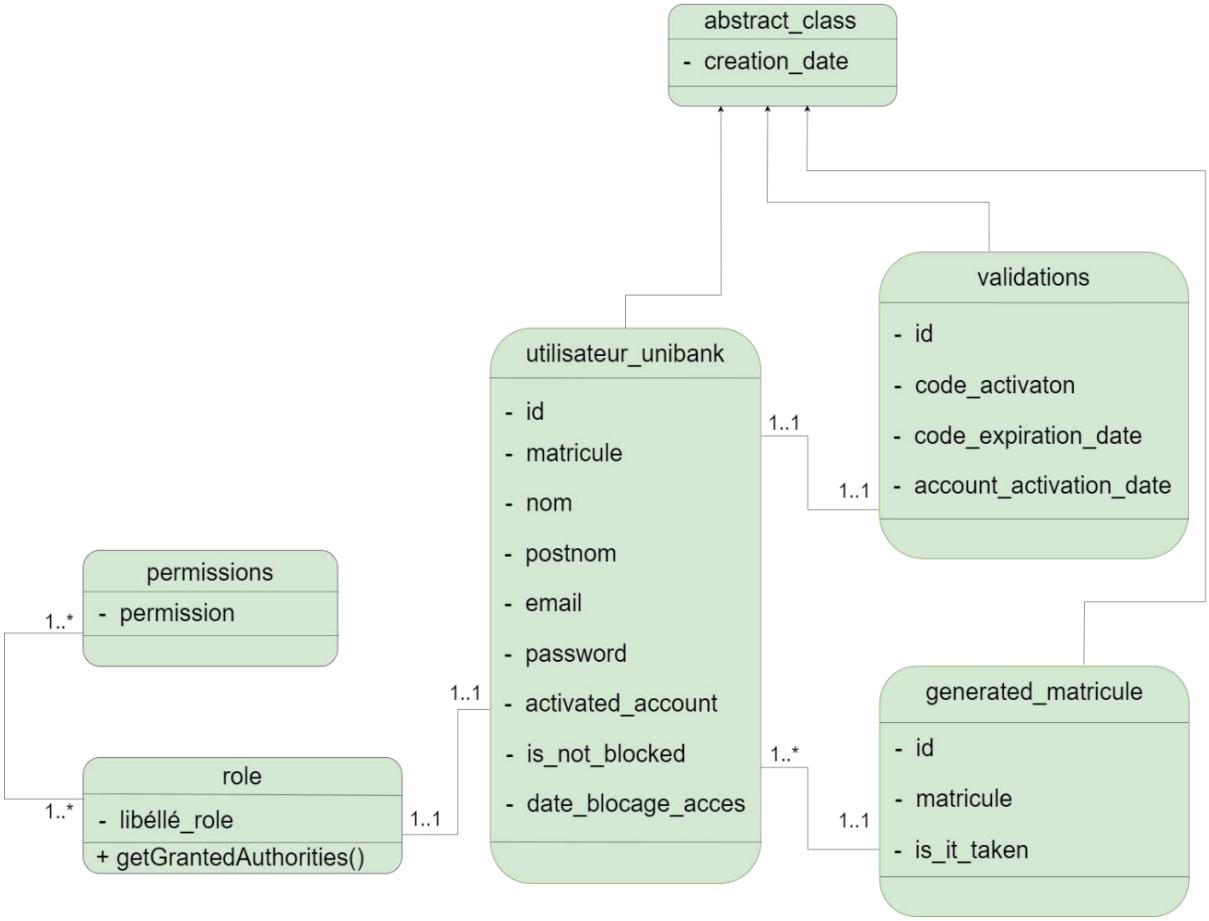


Figure 7:Diagramme de classe du service user_mgt

- **Diagramme de séquence « Scénario : Création et activation d'un compte utili**

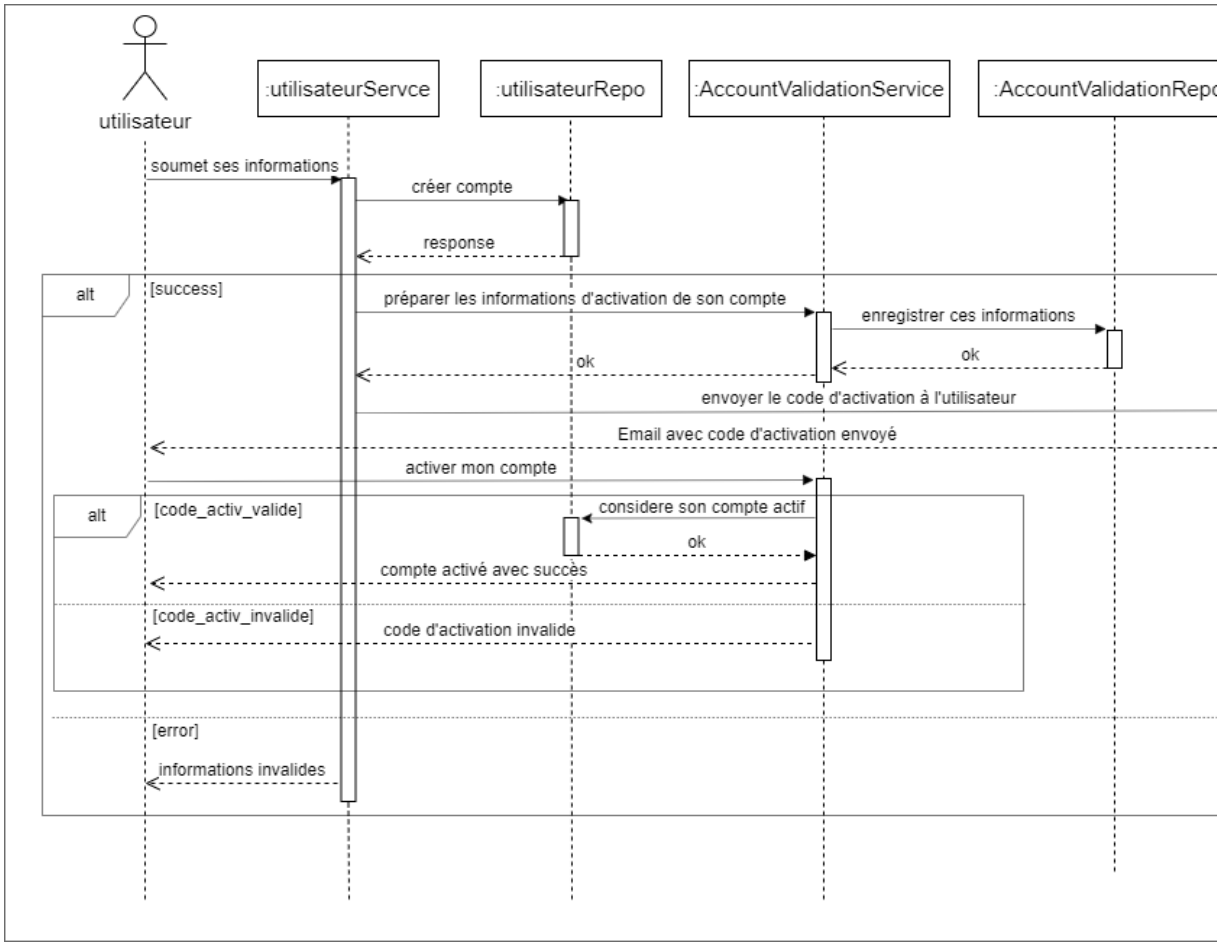


Figure 8:Diagramme de séquence pour les scénarios "créer et activer mon compte

2.2.3.3 Le microservice « Slips_mgt »

Ce microservice est dédié à la gestion des pièces justificatives associées aux opérations de paiement, en particulier les bordereaux. Lorsqu'un paiement est effectué et qu'un bordereau est scanné, le service réagit à l'événement émis par le module de gestion des paiements. Il assure alors l'enregistrement du bordereau correspondant. Dans le cas d'un paiement en ligne, il prend en charge la génération automatique du bordereau avant de procéder à son enregistrement. Ce mécanisme garantit la traçabilité et l'intégrité des justificatifs de paiement, quel que soit le canal utilisé.

- **Diagramme de cas d'utilisation**

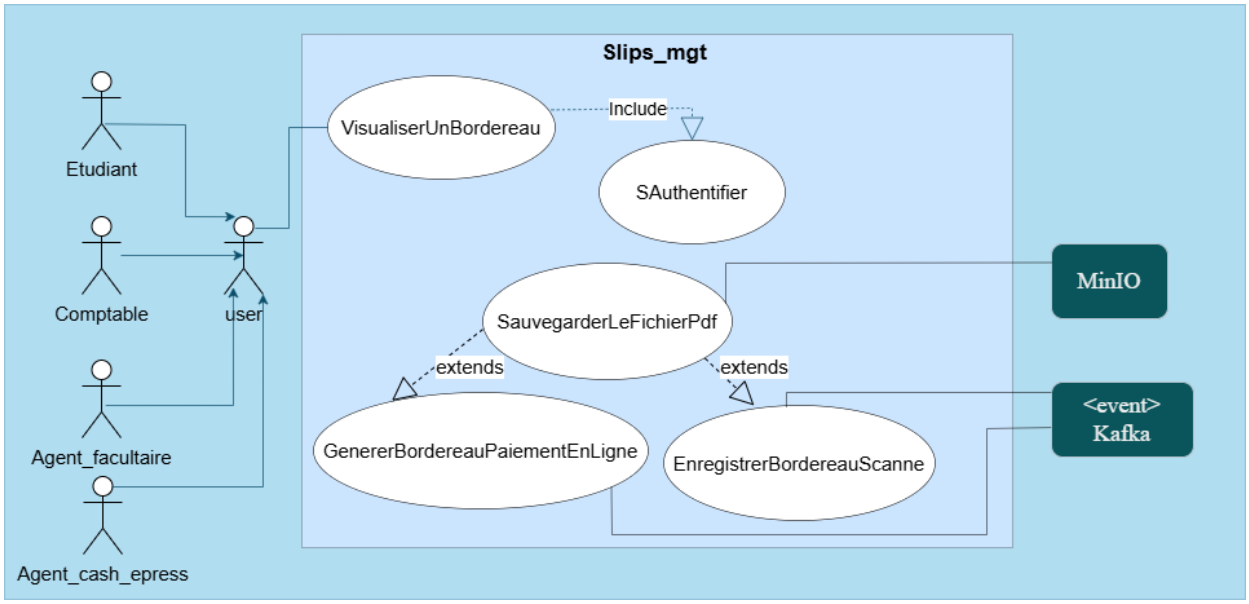


Figure 9:Diagramme de cas d'utilisation du microservice Slips_mgt

- **Diagramme de classe**

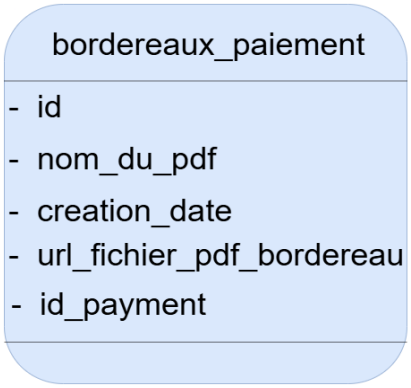


Figure 10:Diagramme de classe du service Slips_mgt

- **Diagramme de séquence « Générer et enregistrer un bordereau »**

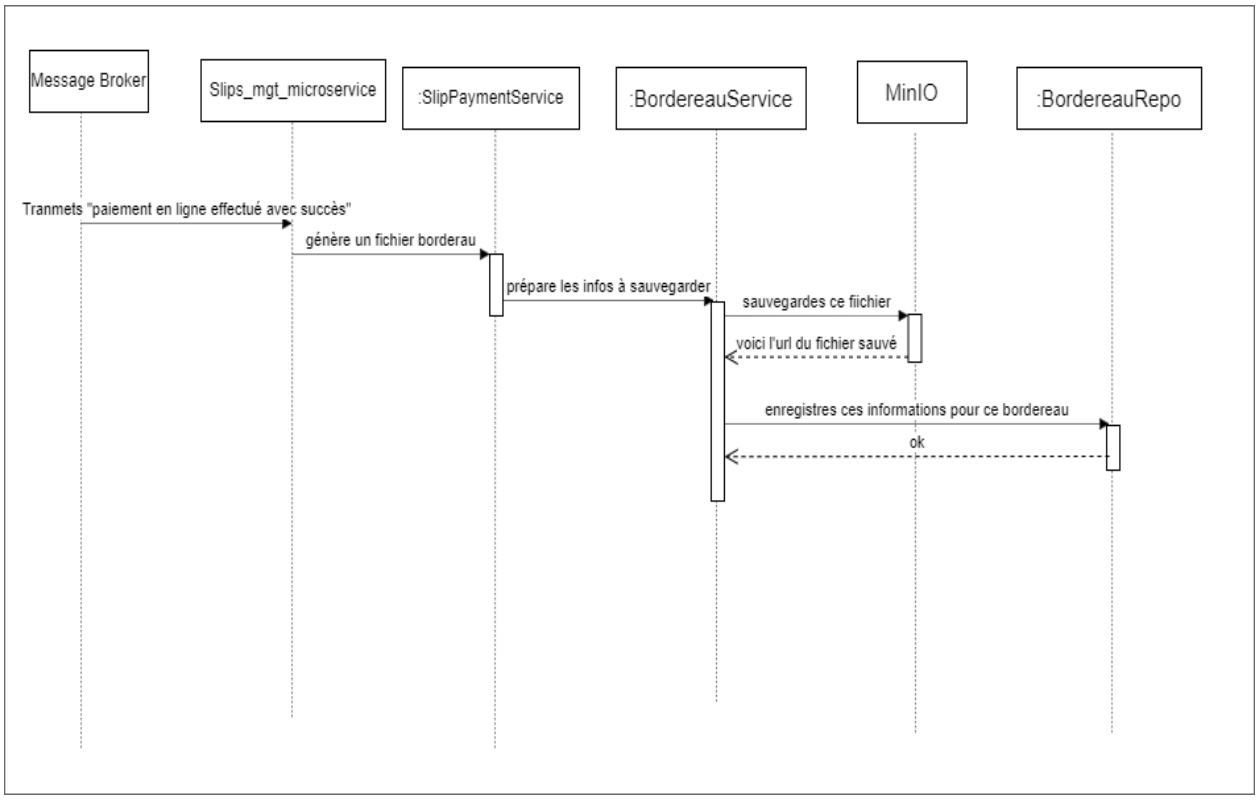


Figure 11:Diagramme de séquence « scénario : Générer et enregistrer un bordereau »

2.2.3.4 Le microservice « Payment_mgt »

Ce microservice se charge de toutes les opérations liées au paiement au sein du système. Il assure notamment l'enregistrement des paiements effectués auprès du service Cash Express. Il permet également la validation et la consultation des paiements par les différents utilisateurs. Dans le cadre des processus de validation, le microservice peut être amené à contacter une API de l'UAT. Cela lui permet de transmettre les informations relatives au paiement à valider, afin de les enregistrer dans le système de l'université.

- **Diagramme de cas d'utilisation**

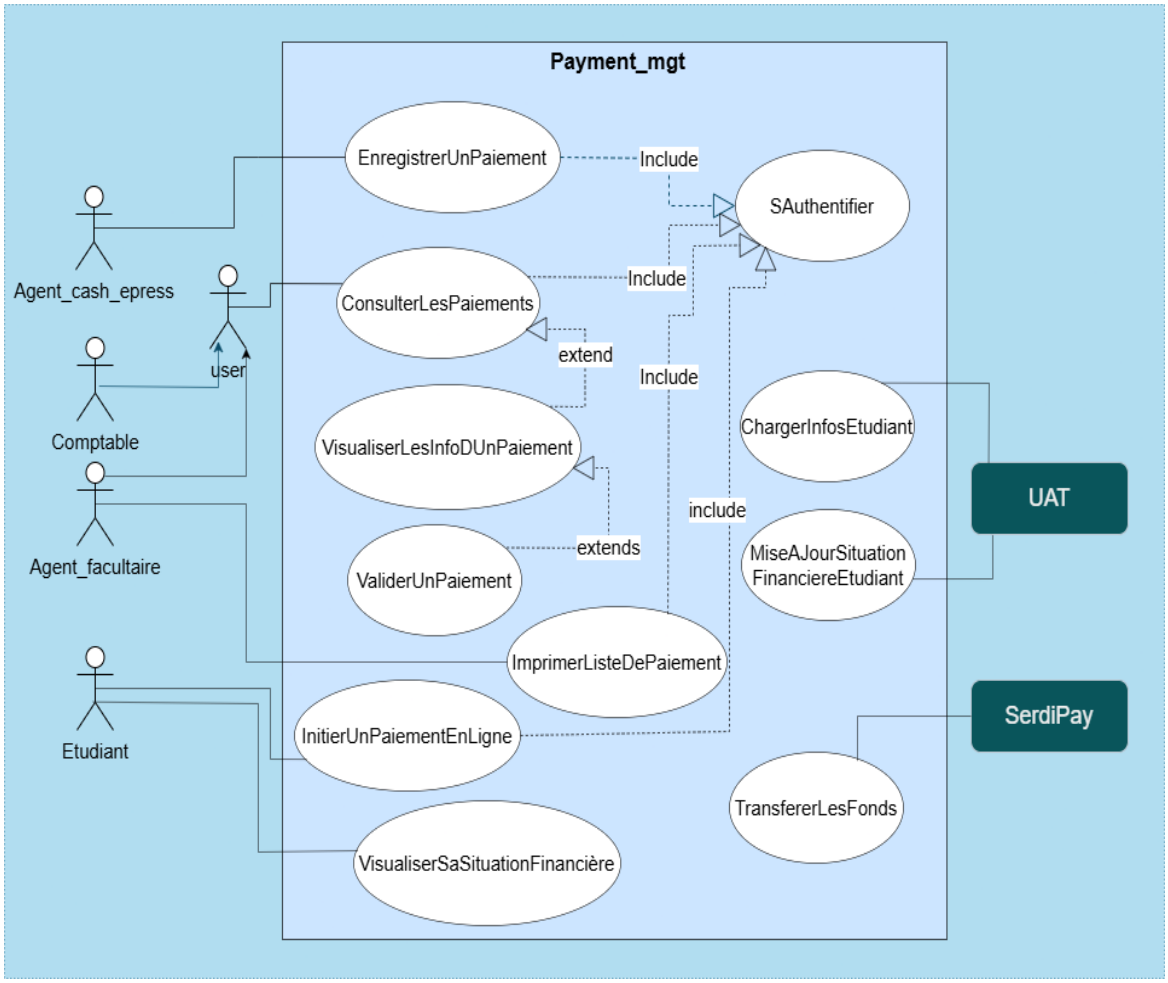


Figure 12:Diagramme de cas d'utilisation du microservice Payment_mgt

- **Diagramme de classe**

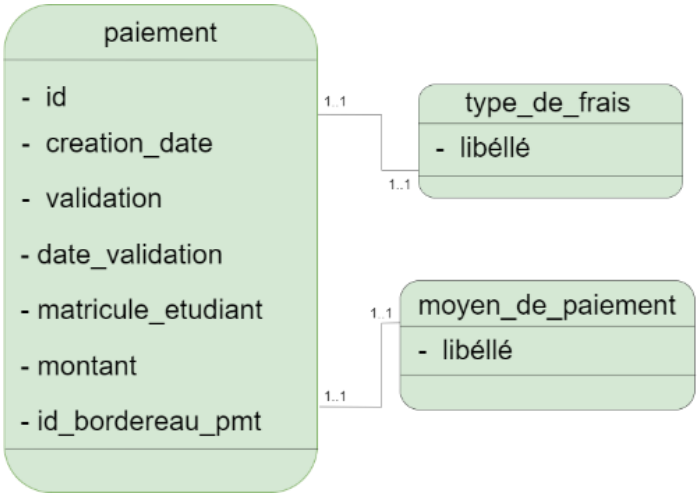


Figure 13:Diagramme de classe du service Payment_mgt

- **Diagramme de séquence de « Validation d'un paiement »**

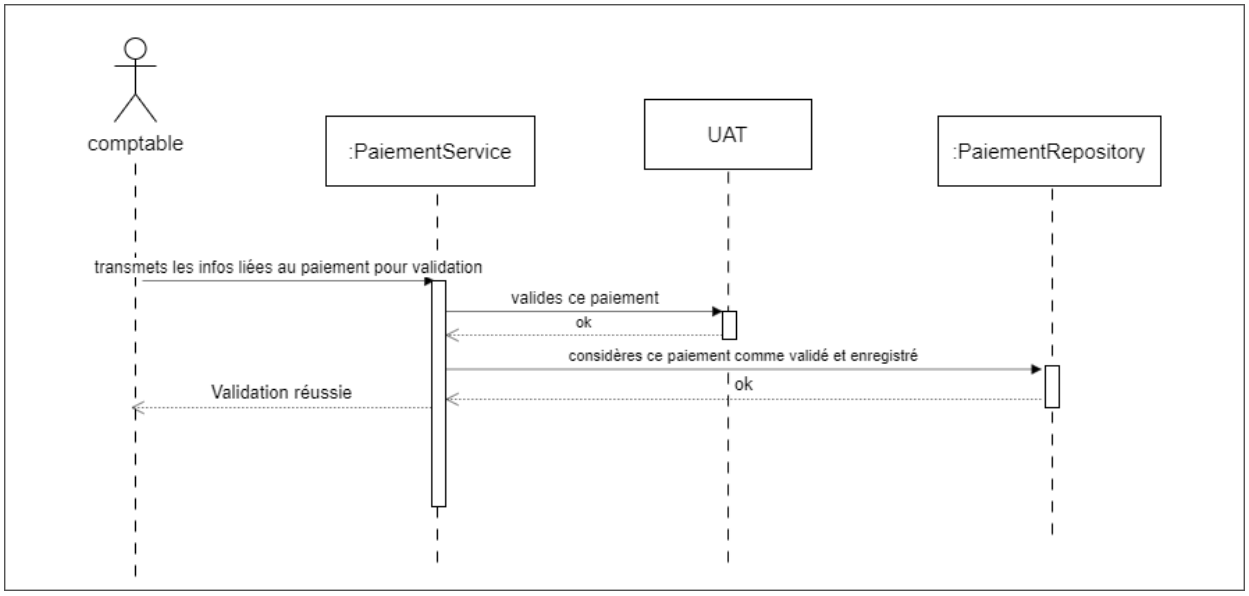


Figure 14:Diagramme de séquence "scénario : validation d'un paiement"

2.2.4 Schéma global de l'architecture

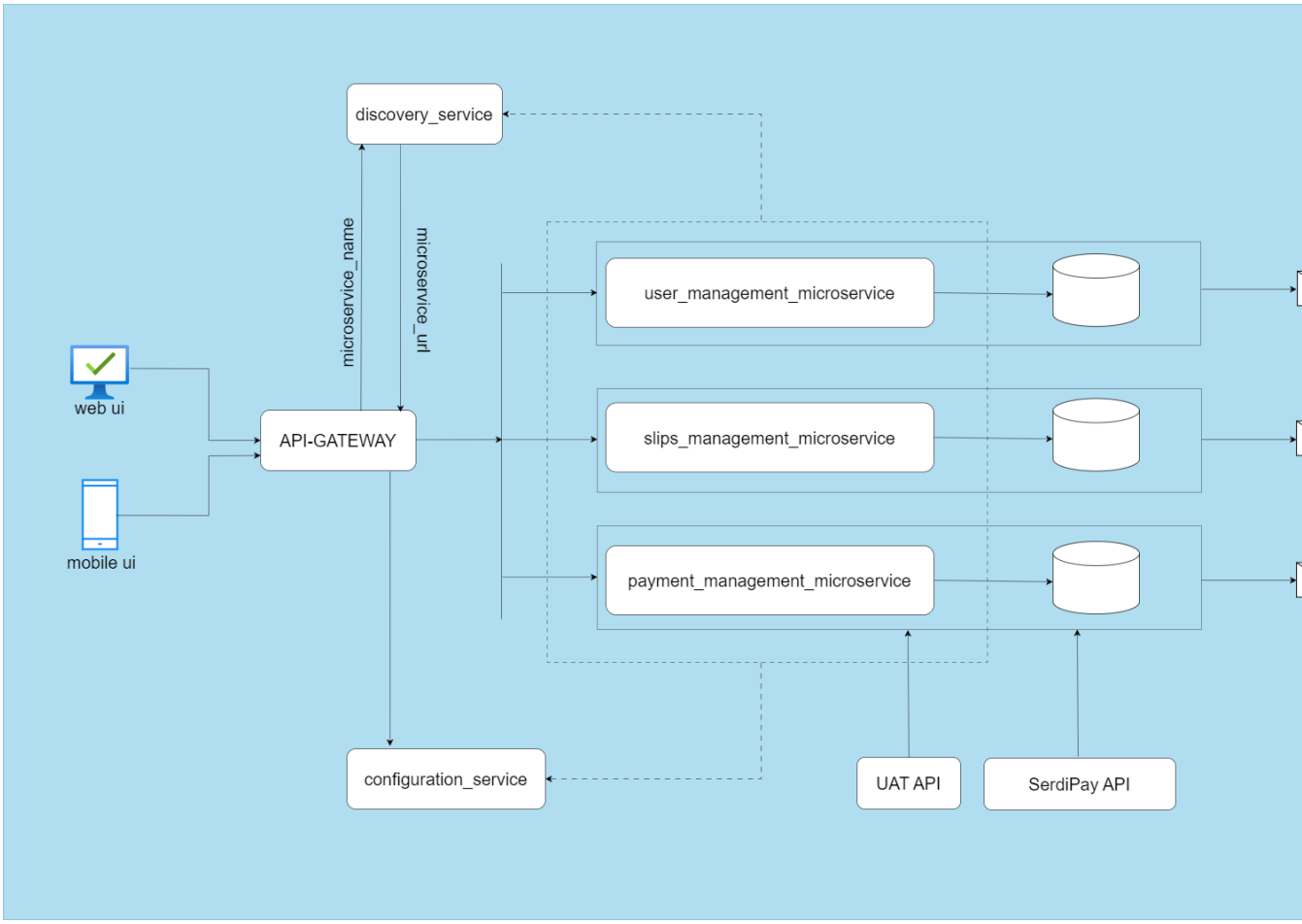


Figure 15:Architecture globale du système

2.3 Conclusion partielle

Dans ce chapitre, nous avons présenté l'approche méthodologique ayant conduit à la conception du système de gestion de paiement proposé. Après une analyse minutieuse des besoins fonctionnels et non fonctionnels, nous avons opté pour une architecture basée sur les microservices, garantissant ainsi modularité, évolutivité et résilience du système.

Les diagrammes UML élaborés nous ont permis de formaliser les interactions entre les différents acteurs et les modules du système, offrant une vue claire et structurée de son fonctionnement. La conception ainsi réalisée constitue une base solide pour le développement et la mise en œuvre du système, tout en assurant une flexibilité nécessaire pour les évolutions futures.

Le chapitre suivant sera consacré à l'implémentation pratique du système, en mettant en lumière les technologies choisies, les outils utilisés, ainsi que les différentes interfaces utilisateur.

Chapitre 3
Implémentation du système UniBank

Après avoir défini les besoins du système et élaboré son architecture au chapitre précédent, ce chapitre est consacré à la mise en œuvre concrète du système que nous avons appelé UniBank. Il détaille l'environnement de développement utilisé, les technologies choisies, ainsi que les différentes étapes techniques de l'implémentation.

La répartition des modules dans une architecture microservices a permis une meilleure isolation des responsabilités et une évolutivité facilitée. La communication entre ces services s'est appuyée non seulement sur des API REST pour les interactions synchrones, mais aussi sur **Apache Kafka** en tant que message broker, pour assurer une communication asynchrone, résiliente et scalable entre certains services critiques.

Nous abordons également l'intégration de la base de données, la communication avec les différents services externes, le développement des interfaces utilisateur, ainsi que la mise en place des mécanismes de sécurité et de gestion des accès, indispensables à la fiabilité du système.

3.1 Environnement de développement et outils utilisés

Le développement du système **UniBank** s'est appuyé sur une pile technologique moderne, adaptée à une architecture basée sur les microservices, et sélectionnée pour sa robustesse, sa facilité d'intégration, et sa compatibilité avec les besoins du projet.

3.1.1 Langages et frameworks

- **Java (Spring Boot)** : utilisé pour développer les différents microservices backend. Spring Boot permet de créer des applications Java prêtes à l'emploi avec une configuration minimale, tout en intégrant des mécanismes puissants pour la sécurité, la gestion des dépendances et la communication interservices.
- **Flutter** : utilisé pour le développement de l'interface mobile, offrant une interface multiplateforme performante, fluide et responsive.

- **VueJs** : utilisé pour le développement de l'interface web de notre application. Ce framework JavaScript est léger, progressif, et facilite la création d'interfaces utilisateur dynamiques.

3.1.2 Système de gestion de base de données

- **PostgreSQL** : choisi comme système de gestion de base de données relationnelle, pour sa performance, sa fiabilité, et son support étendu pour les requêtes complexes. Chaque microservice dispose de sa propre base de données, conformément aux principes de l'architecture microservices.

3.1.3 Le choix du PSP

Pour la mise en place du système de paiement, nous avons choisi d'utiliser **SerdiPay** comme passerelle de paiement, son choix se justifie par sa facilité d'intégration et sa documentation détaillée.

L'architecture développée dans ce projet reste modulaire et adaptable à d'autres prestataires de services de paiement qui seraient disponibles localement (ex : PayDunya, DPO Group, API bancaire d'Equity BCDC, etc.). Ainsi, si l'université souhaite une implémentation réelle, il suffira de remplacer l'intégration de SerdiPay par celle d'un PSP compatible, grâce à un partenariat avec celui-ci.

3.1.4 Communication entre microservices

- **REST APIs** : mises en œuvre pour permettre les échanges synchrones entre les microservices via des points d'entrée bien définis.
- **Apache Kafka** : utilisé comme **message broker** pour assurer la communication asynchrone entre certains services critiques (comme l'enregistrement d'un bordereau ou la création du paiement associé), garantissant ainsi une meilleure résilience du système et un découplage efficace entre les services.

3.1.5 Intégration de services externes

- **SerdiPay :** tel que mentionné précédemment, SerdiPay constitue la passerelle retenue pour l'implémentation de la fonctionnalité de paiement en ligne. Toutefois,

après échanges avec leur service commercial, il a été précisé que l'accès à leur API est exclusivement réservé aux entités juridiques ayant conclu un accord préalable avec eux, ce qui exclut les développeurs individuels. Pour pallier cela, nous avons développé un service fictif (mock service), dockerisé, qui simule le comportement de leur API. Ce service est intégré au système afin de reproduire le processus de paiement. Ainsi, si une intégration réelle devait être envisagée à l'avenir, il suffirait d'adapter certains paramètres pour remplacer ce service par l'API officielle.

- **UAT :** on aura également besoin de faire appel au système de l'université pour certains besoins de notre application comme présenté précédemment. N'ayant pas accès à celui-ci, nous avons également développé un service fictif qui simule les fonctionnalités de l'UAT dont on a besoin. Ainsi plus tard on pourra passer à une intégration réelle sans des modifications majeures.
- **MinIO** : un système de stockage d'objets compatible avec S3, que l'on a utilisé pour sauvegarder les différents fichiers pdf de nos bordereaux scannés ou générés pour ainsi permettre aux utilisateurs de pouvoir les consulter selon le besoin.

3.1.6 Sécurité

- **Spring Security + JWT (JSON Web Token)** : utilisé pour l'authentification et l'autorisation, garantissant la protection des données et la sécurisation des accès au système de manière stateless.

3.2 Présentation des interfaces utilisateur

L'expérience utilisateur constitue un aspect essentiel du système UniBank. Afin de répondre aux besoins spécifiques de chaque type d'utilisateur (l'agent du cash-express, le comptable, l'étudiant ainsi que l'agent facultaire), deux entités applicatives principales ont été développées : une application mobile et une application web.

a. Application mobile (Flutter)

L'application mobile est principalement destinée aux agents du cash-express pour leurs différentes interactions avec le système. Les figures 16 et 17 présentent un aperçu sur ce que sont nos interfaces mobiles.

Cette interface présente l'écran d'accueil de notre application mobile, ici l'agent du cash express voit ses récents enregistrements effectués (les quatre derniers), une possibilité de filtre et de recherche, mais aussi on peut y voir d'autres boutons qui parlent d'eux même.

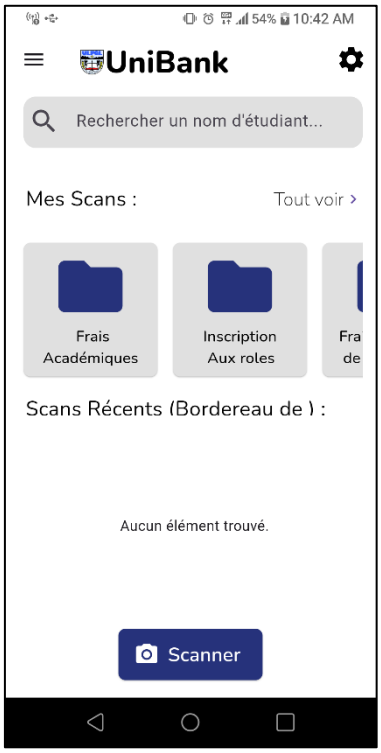


Figure 16:interface d'accueil et bouton de scan

L'interface ci-dessous s'agit de du formulaire que l'agent du cash express remplit après avoir scanné le bordereau de paiement.

Figure 17: formulaire d'informations relatives au paiement

b. Application web (VueJs)

Cette application est destinée à la fois au comptable (admin du système), aux agents du décanat ainsi qu'aux étudiants.

Cette première interface s'agit de l'interface de connexion, où l'utilisateur devra rentrer son matricule et son mot de passe.

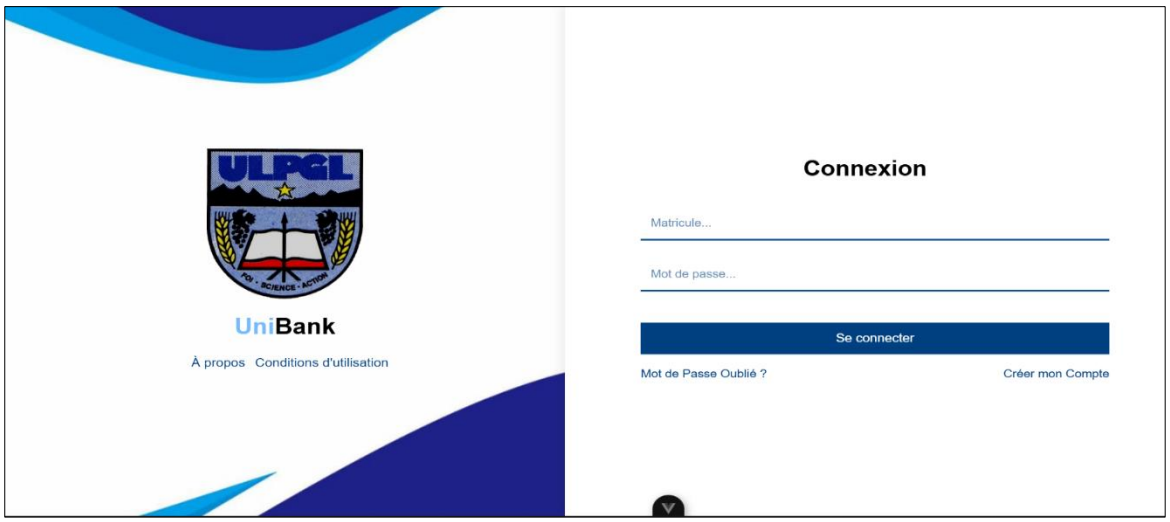


Figure 18:interface de connexion appli web

- **Interfaces propres au comptable**

Il s'agit ici de l'interface d'accueil que voit notre comptable juste après sa connexion, il y voit une vision d'ensemble sur sa gestion dans le système, y voit également les récents payements qui ont été effectués (encore une fois les quatre derniers), mais aussi grâce à la liste de menu à gauche il a accès à un certain nombre de fonctionnalité.

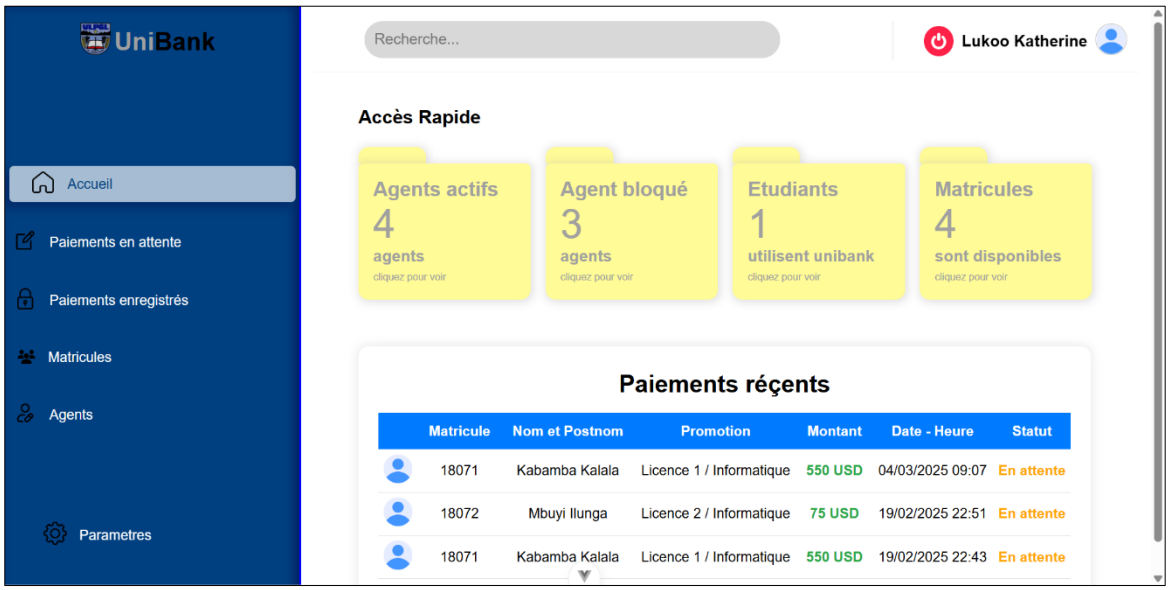


Figure 19:interface d'accueil et visualisation des paiements

Lorsque le comptable clique sur un payement pour en visualiser les détails, il s'affiche un pop-up montrant le bordereau associé au payement, mais aussi deux possibilités s'offrent à lui, il peut se décider de fermer la nouvelle interface, ou soit valider le paiement en cliquant sur le bouton ***« Enregistrer le Paiement »***

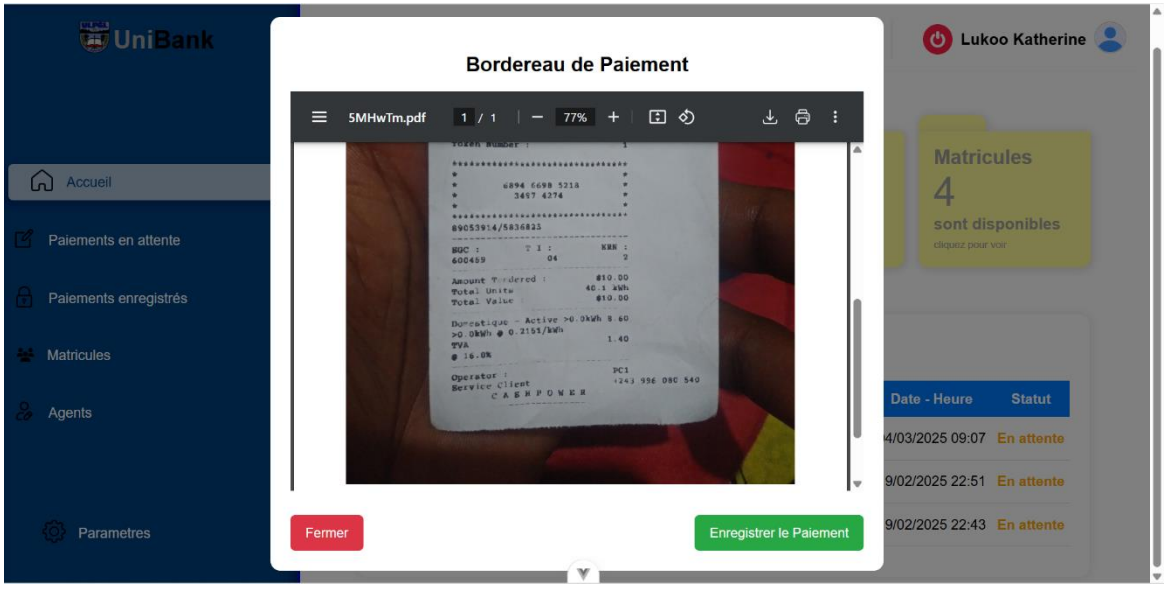


Figure 20:visualisation d'un bordereau

- **Interfaces propres à l'agents facultaire**

L'interface ci-dessous est l'espace d'accueil que voit notre agent facultaire juste après sa connexion, il y voit directement les différents paiements récents, une possibilité de recherche s'offre à lui ainsi que des options de filtre suivant le type de frais, tout à gauche il a également une liste de menu où grâce à laquelle il peut effectuer différentes tâches.

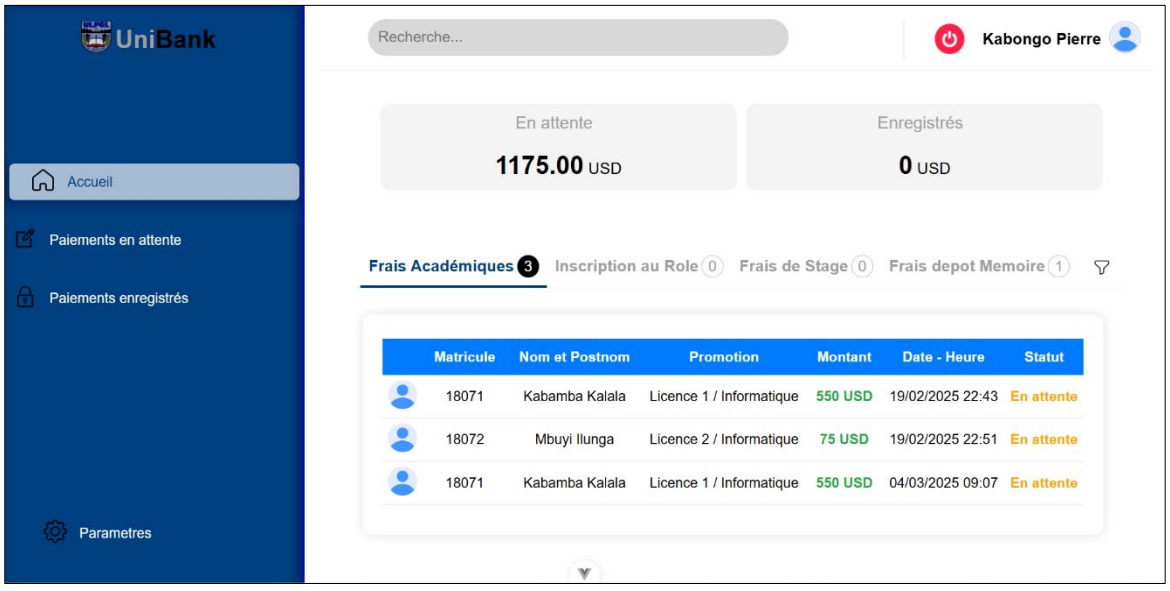


Figure 21:interface d'accueil et visualisation des paiements

- **Interfaces propres à l'étudiant**

L’interface qui suit présente l’espace étudiant au sein de notre système, il voit d’une manière détaillée toute sa situation financière, il y a même un graphique présentant ses tendances de paiement sur tout un semestre, mais également il peut accéder à tout son historique de paiement.

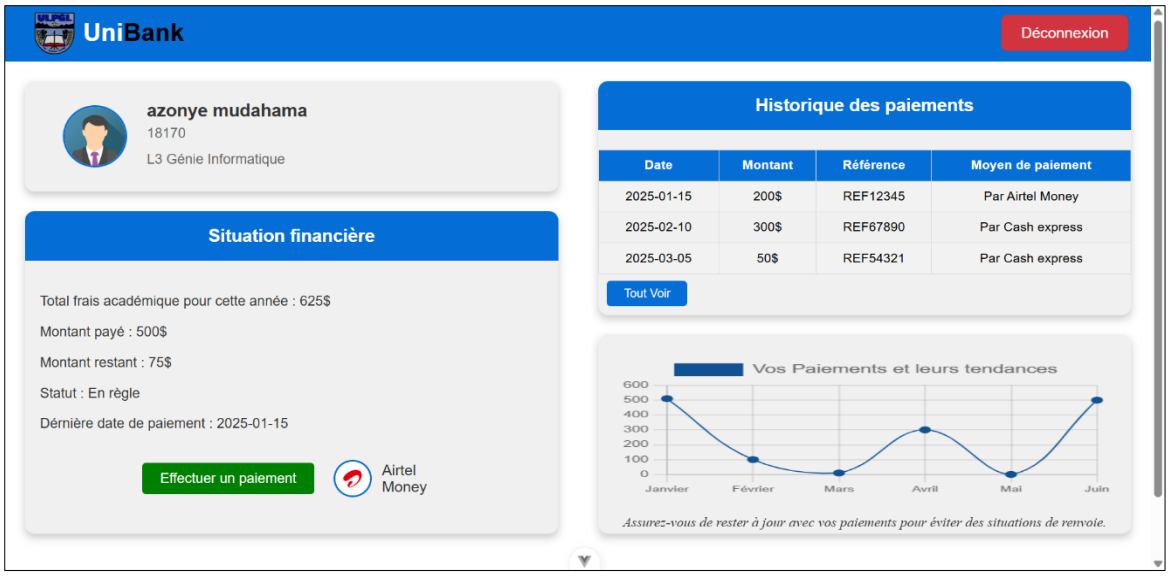


Figure 22:interface d'accueil et visualisation de son état financier

L’étudiant en cliquant sur le bouton ***« Effectuer un paiement »*** se voit affiché ce pop-up lui permettant d’indiquer toutes les informations requises pour pouvoir effectuer son paiement via Airtel-Money.

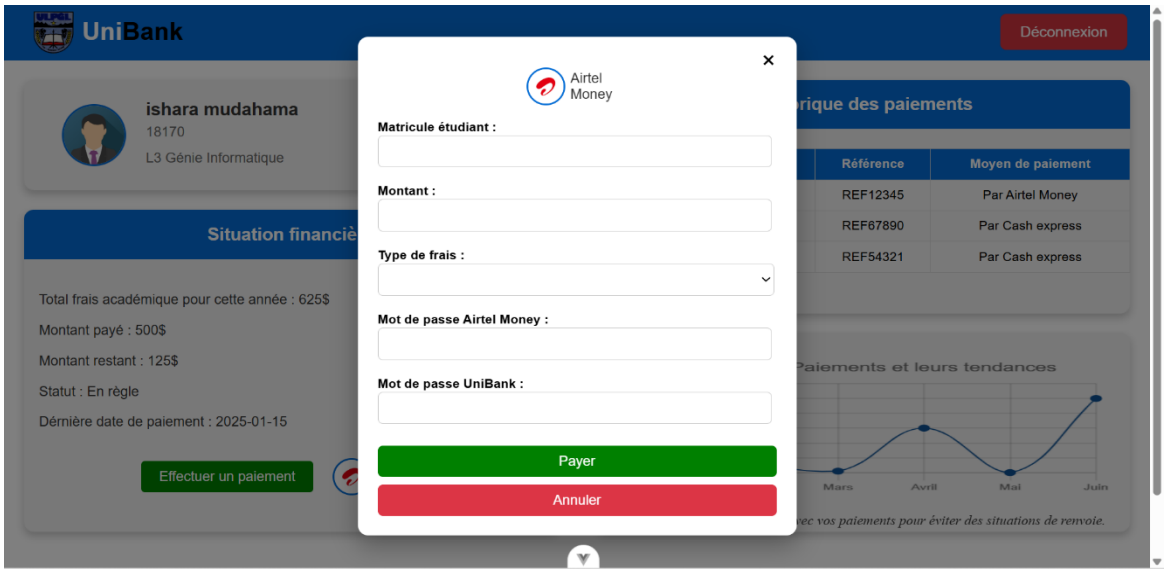


Figure 23:formulaire de paiement en ligne

3.3 Conclusion partielle

Ce chapitre a présenté en détail l'implémentation du système UniBank, concrétisant ainsi les choix d'architecture et de conception évoqués précédemment. À travers une approche fondée sur les microservices, nous avons pu développer un système modulaire, évolutif et résiliant, répondant efficacement aux exigences de performance et de sécurité propres à un système de gestion de paiement.

Nous avons exposé l'environnement de développement adopté, les technologies retenues tant pour le backend (Spring Boot) que pour les interfaces utilisateur (Flutter et VueJs), ainsi que les solutions de stockage et de communication (PostgreSQL, Apache Kafka, MinIO). L'intégration de services tiers, bien qu'encore limitée au mode test, montre également l'ouverture du système à des évolutions futures.

Enfin, la présentation des interfaces utilisateurs met en lumière la volonté de concevoir une expérience utilisateur fluide, intuitive et adaptée à chaque profil d'utilisateur. Le travail réalisé dans cette phase constitue ainsi un socle technique solide pour l'exploitation du système et pour ses futures extensions.

Conclusion générale

À l'issue de ce travail portant sur la conception et la réalisation d'un système de gestion de paiement reliant les cash express d'une banque et une université, nous pouvons affirmer que les objectifs initiaux ont été globalement atteints. En effet, partant des problématiques identifiées au sein de notre institution, notamment les limites liées à l'enregistrement manuel des bordereaux de paiement et l'absence d'alternatives modernes de paiement à distance, nous avons proposé une solution technologique palliative et adaptée au contexte local.

Avant de conclure notre travail, il est préférable de revenir aux hypothèses émises lors de l'introduction du travail, et réponses aux questions formulées dans l'introduction, nous pouvons établir ce qui suit :

La première hypothèse, selon laquelle l'automatisation de l'enregistrement des bordereaux combinée à un module de paiement direct améliorerait significativement le processus de gestion des paiements, a été confirmée. Le système permet une réduction notable des délais de traitement et une meilleure traçabilité des paiements.

La seconde hypothèse, portant sur l'adéquation des outils technologiques retenus, est partiellement confirmée. Les technologies choisies ont effectivement permis de construire une solution robuste, cependant l'impossibilité d'établir une connexion réelle avec le prestataire SerdiPay et le système de l'université constitue une limite temporaire.

La troisième hypothèse, affirmant que l'implémentation de mécanismes de sécurité (authentification et chiffrement) assurerait la confidentialité des données, a été validée par la mise en place de solutions éprouvées telles que Spring Security et JWT.

Pour les perspectives futures, nous recommandons :

- La signature de partenariats avec des prestataires de services de paiement pour rendre opérationnelle la fonctionnalité de paiement mobile ;

- L’intégration complète du système avec le SI de l’université (UAT) ;

En somme, ce travail constitue une première contribution technique et conceptuelle en vue d’une transition vers une gestion numérique plus fluide et sécurisée des paiements universitaires. Il apporte une modeste avancée dans la réflexion autour de la digitalisation des services administratifs dans le contexte académique local.

Bibliographie

[1] H. G. &. F. Monjoie, Introduction aux sciences informatiques, Liege: Sart Tilman, 2005.

[2] M. Vingi, «Cours de Génie Logiciel,» 2024.

[3] J. A. O'Brien et G. M. Marakas, Management Information Systems, New York: McGraw-Hill Education, 2011.

[4] K. C. Laudon et J. P. Laudon, Management Information Systems: Managing the Digital Firm, Boston: Pearson, 2020.

[5] J. S. Valacich et C. Schneider, Information Systems Today: Managing in the Digital World, Boston: Pearson, 2016.

[6] S. Newman, Building Microservices: Designing Fine-Grained Systems, Sebastopol: O'Reilly Media, 2015.

[7] C. Richardson, Microservices Patterns: With Examples in Java, Shelter Island: Manning Publications, 2018.

[8] M. Fowler, Patterns of Enterprise Application Architecture, Boston: Addison-Wesley, 2019.

[9] «Moyen de paiement,» Wikipédia, Fevrier 2025. [En ligne]. Available: https://fr.wikipedia.org/wiki/Moyen_de_paiement. [Accès le Mars 2025].

[10] «Sortir sa carte bancaire, un geste en voie de disparition ?,» Le Monde, [En ligne]. Available: https://www.lemonde.fr/argent/article/2024/10/19/sortir-sa-carte-bancaire-un-geste-en-voie-de-disparition_6355663_1657007.html. [Accès le Mars 2025].

[11] B. U. I. e. a. Khan, «A compendious study of online payment systems: Past developments, present impact, and future considerations.,» *International Journal of Advanced Computer Science and Applications.,* p. 8, 2017.

[12] D. S. B, «cours de commerce électronique : chap2. Le paiement électronique,» 2023. [En ligne]. Available: https://fr.scribd.com/document/638564721/Chapitre-2-le-paiement-electronique. [Accès le Mars 2025].

[13] WORLDLINE, « Votre guide pour des paiements en ligne en toute simplicité,» [En ligne]. Available: https://worldline.com/content/dam/worldline/local/fr-fr/documents/ebook/ebook-worldline-payments-guide-fr.pdf. [Accès le Mars 2025].

[14] «Les avantages et inconvénients des solutions de paiement en ligne pour les entreprises e-commerce,» IMAGINA, [En ligne]. Available: https://imagina.com/fr/blog/article/paiement-en-ligne/. [Accès le Mars 2025].

[15] Guru99, «Différence entre les exigences fonctionnelles et non fonctionnelles,» Guru99, 2023. [En ligne]. Available: https://www.guru99.com/fr/functional-vs-non-functional-requirements.html. [Accès le Avril 2025].

[16] M. Richards, Software Architecture Patterns, Sebastopol: O'Reilly Media, 2015.

[17] E. Evans, Domain-Driven Design: Tackling Complexity in the Heart of Software, Boston: Addison-Wesley, 2004.

[18] R. C. Martin, Agile Software Development: Principles, Patterns, and Practices, Upper Saddle River: Prentice Hall, 2002.

[19] M. Fowler, «Microservices,» 2014. [En ligne]. Available: https://martinfowler.com/articles/microservices.html. [Accès le Avril 2025].

[20] R. Daigneau, Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services, Boston: Addison-Wesley, 2011.

[21] A. G. Gloire, «CONCEPTION ET REALISATION D'UN SYSTEME D'IMPLICATION PARENTALE SCOLAIRE POUR LES ECOLES PRIMAIRES DE LA VILLE DE GOMA,» Goma, 2023.

Annexes

Annexes

Annexe 1 : Codes sources de toutes les parties du système proposé

- ✓ ***Les différents microservices du système :*** http://bit.ly/4aq2ClL
- ✓ ***L'application web qui fait appel à notre backend :*** https://bit.ly/49oRZhW
- ✓ ***L'application mobile utilisée par l'agent Cash-Express :*** https://bit.ly/3YKUtkw
- ✓ ***Le service fictif simulant l'api du prestataire de paiement :*** https://bit.ly/3YKUHYU
- ✓ ***Le service fictif simulant certaines fonctionnalités de UAT :*** https://bit.ly/3Y4W0lp

Annexe 2 : Extrait de code : logique métier du service gestion de paiement

```
package ulpgl.unibank.payment_management.services.implementations;

import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.kafka.core.KafkaTemplate;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import ulpgl.unibank.payment_management.dtos.*;
import
ulpgl.unibank.payment_management.dtos.fromUniversity.EtudiantResponse;
import
ulpgl.unibank.payment_management.exceptions.PaymentFailedException;
import
ulpgl.unibank.payment_management.exceptions.UnauthorizedException;
import ulpgl.unibank.payment_management.models.Payment;
import ulpgl.unibank.payment_management.models.enums.PaymentMode;
import ulpgl.unibank.payment_management.repository.PaymentRepository;
import ulpgl.unibank.payment_management.services.PaymentService;
import ulpgl.unibank.payment_management.services.externals.*;
import ulpgl.unibank.payment_management.services.mappers.PaymentMapper;
import
ulpgl.unibank.payment_management.services.utils.PaymentsListFileGenerato
r;
import
ulpgl.unibank.unibankshareddtos.shared_dtos.CashExpressPaymentSuccessEve
nt;
import
ulpgl.unibank.unibankshareddtos.shared_dtos.OnlinePaymentSuccessEvent;

import java.io.IOException;
import java.math.BigDecimal;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
```

```
import java.time.temporal.TemporalAdjusters;
import java.util.*;

@Slf4j
@AllArgsConstructor
@Service
public class PaymentServiceImpl implements PaymentService {

    private final PaymentRepository paymentRepository;
    private final UniversityAgentService universityAgentService;
    private final UniversityStudentService universityStudentService;
    private final ExternalPaymentsNeeds externalPaymentsNeeds;
    private final ContactSerdiPayAPI contactSerdiPayAPI;
    private final GetTheUserCredentials getTheUserCredentials;
    private final KafkaTemplate<String, Object> kafkaTemplate;
    private final PasswordEncoder passwordEncoder;

    @Override
    public void
registerCashExpressPayment(CashExpressPaymentRegisterRequest requestDto)
throws IOException, Exception {

        String matriculeUser =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        String studentRegNumber =
requestDto.getMatriculeEtudiantPayant();
        EtudiantResponse payingStudent =
universityStudentService.getStudentByMatricule(studentRegNumber);

        Payment newPayment =
setNewCashExpressPaymentInfomations(requestDto, matriculeUser,
payingStudent);

        Payment savedPayment = paymentRepository.save(newPayment);
        CashExpressPaymentSuccessEvent eventInfo =
CashExpressPaymentSuccessEvent.builder()
                .paymentId(savedPayment.getId())

.fichierBordereau(requestDto.getFichierBordereau().getBytes())
                .nameThatWillGetTheSlip(
                        payingStudent.getNom()+"
"+payingStudent.getPostnom()+" - "+
                        payingStudent.getPromotion()+"
"+payingStudent.getDepartement()
                )
                .build();

        kafkaTemplate.send("cash-express-payment-created-
dev",eventInfo);
        log.info("✅ CashExpressPaymentSuccessEvent envoyé avec succès
au topic Kafka ...");

        PaymentMapper.toPaymentDto(savedPayment);
    }
```

```
    private static Payment
setNewCashExpressPaymentInfomations(CashExpressPaymentRegisterRequest
requestDto, String matriculeUser, EtudiantResponse payingStudent) {
        Payment newPayment = new Payment();
        BigDecimal montantPayed = new
BigDecimal(requestDto.getMontant());
        newPayment.setMontant(montantPayed);
        newPayment.setTypeFrais(requestDto.getTypeDeFrais());

newPayment.setPaymentMode(PaymentMode.paiement_au_niveau_du_cashExpress)
;
        newPayment.setUserAuthorMatricule(matriculeUser);

newPayment.setMatriculeEtudiant(requestDto.getMatriculeEtudiantPayant())
;
        newPayment.setNom_complet(payingStudent.getNom()+" "+
payingStudent.getPostnom());

newPayment.setFaculte(payingStudent.getDepartement().getFaculte().getNom
());

newPayment.setDepartement_promotion(payingStudent.getPromotion().getNom(
)+" - "+ payingStudent.getDepartement().getNom());
        return newPayment;
    }

    @Override
    public boolean validateAndSavePayment(String idPaiement) {
        Optional<Payment> paiementToValidate =
paymentRepository.findById(Integer.valueOf(idPaiement));
        if (paiementToValidate.isPresent()) {
            PaymentValidationRequest preparedRequest =
PaymentMapper.toPaymentValidationRequest(paiementToValidate.get());
            Boolean response =
externalPaymentsNeeds.validatePayment(preparedRequest);
            if (response){

paiementToValidate.get().setLastModifiedDate(LocalDateTime.now());
                paiementToValidate.get().setEnregistrementStatus(true);
                paymentRepository.save(paiementToValidate.get());
                return response;
            }
            return response;
        }
        return false;
    }

    @Override
    public PaymentResponse
initiateOnlinePayment(OnlinePaymentInitiateRequest request) {

        String matriculeUser =
SecurityContextHolder.getContext().getAuthentication()
                .getName();

        String connectedUserPassword =
```

```
getTheUserCredentials.getUserPassword(matriculeUser);
        boolean isThisRequestPasswordCorrect =
passwordEncoder.matches(request.getUnibankPassword(),
connectedUserPassword);

        if (isThisRequestPasswordCorrect) {
            String studentRegNumber = request.getStudentMatricule();
            EtudiantResponse payingStudent =
universityStudentService.getStudentByMatricule(studentRegNumber);
            String receiverAccount =
request.getType_frais().equals("frais_academique") ?
"00018000280003722120002" : "00018000280003722120003";

            OnlinePaymentTransactionRequest transactionRequest =
OnlinePaymentTransactionRequest.builder()
                    .amount(Double.parseDouble(request.getAmount()))
                    .senderNumber(payingStudent.getPhoneNumber())
                    .receiverAccount(receiverAccount)
                    .build();

            OnlinePaymentResponse onlinePaymentResponse =
contactSerdiPayAPI.initiateAnOnlinePayment(transactionRequest);

            if (onlinePaymentResponse.getStatus().equals("SUCCESS")) {

                Random random = new Random();
                int number = 10000 + random.nextInt(90000);
                String refNumber = "REF" + String.valueOf(number);

                Payment newPayment =
setNewPaymentInfoFromOnlinePayement(request, matriculeUser,
payingStudent, refNumber);
                DateTimeFormatter formatter =
DateTimeFormatter.ofPattern("dd-MM-yyyy");

                paymentRepository.save(newPayment);

                OnlinePaymentSuccessEvent eventInfo =
OnlinePaymentSuccessEvent.builder()
                        .matricule(request.getStudentMatricule())
                        .studentFullName(payingStudent.getNom()+"
"+payingStudent.getPostnom())

.studentPromotion(payingStudent.getPromotion().getNom())
                        .amountPayed(request.getAmount()+"USD")

.paymentDate(newPayment.getCreatedDate().format(formatter))
                        .paymentReference(refNumber)
                        .nameThatWillGetTheSlip(
                                payingStudent.getNom()+"
"+payingStudent.getPostnom()+" - "+
                                payingStudent.getPromotion().getNom()+"
"+payingStudent.getDepartement().getNom()
                        )
                        .paymentId(newPayment.getId())
                        .typeFrais(newPayment.getTypeFrais())
```

```
                        .build();

                kafkaTemplate.send("online-payment-success-
dev",eventInfo);
                log.info("✅ OnlinePaymentSuccessEvent envoyé avec
succès au topic Kafka ... with id : "+eventInfo.getPaymentId());

                return PaymentMapper.toPaymentDto(newPayment);
            } else {
                throw new PaymentFailedException("Échec de la
transaction : statut renvoyé = " + onlinePaymentResponse.getStatus());
            }

        } else {
            throw new UnauthorizedException("Mot de passe Unibank
incorrect pour l'utilisateur connecté.");
        }
    }


    private static Payment setNewPaymentInfoFromOnlinePayement(
            OnlinePaymentInitiateRequest request,
            String matriculeUser,
            EtudiantResponse payingStudent,
            String paymentRef) {
        Payment newPayment = new Payment();
        BigDecimal montantPayed = new BigDecimal(request.getAmount());
        newPayment.setMontant(montantPayed);
        newPayment.setTypeFrais(request.getType_frais());
        newPayment.setPaymentMode(PaymentMode.paiement_par_mobileMoney);
        newPayment.setPayment_reference(paymentRef);
        newPayment.setUserAuthorMatricule(matriculeUser);
        newPayment.setMatriculeEtudiant(request.getStudentMatricule());
        newPayment.setNom_complet(payingStudent.getNom()+" "+
payingStudent.getPostnom());

newPayment.setFaculte(payingStudent.getDepartement().getFaculte().getNom
());

newPayment.setDepartement_promotion(payingStudent.getPromotion().getNom(
)+" - "+ payingStudent.getDepartement().getNom());
        return newPayment;
    }

    @Override
    public List<PaymentResponse> findPaiementByTypeFrais(String
typeFrais, boolean status) {

        String connectedAgentFacultaireUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        String faculteAgent =
universityAgentService.getAgentByMatricule(connectedAgentFacultaireUsern
ame).getFaculte().getNom();

        return
```

```
paymentRepository.findByTypeFraisAndEnregistrementStatus(typeFrais,
status).stream()
                .map(PaymentMapper::toPaymentDto)
                .filter(paymentResponse -> {
                    String faculteEtudiant =
paymentResponse.getFaculte();
                    return
faculteEtudiant.equalsIgnoreCase(faculteAgent);
                })
                .toList();
    }

    @Override
    public List<PaymentResponse> findAllPaiementsAcademicFees(boolean
status) {
        return
paymentRepository.findByTypeFraisAndEnregistrementStatus("frais_academiq
ue", status).stream()
                .map(PaymentMapper::toPaymentDto)
                .toList();
    }

    @Override
    public List<PaymentResponse> findTheFourLatestPayement() {
        return
paymentRepository.findTop4ByTypeFraisOrderByCreatedDateDesc("frais_acade
mique")
                .stream()
                .map(PaymentMapper::toPaymentDto)
                .toList();
    }

    @Override
    public String calculateSumOfPaymentPerType(String type, Boolean
status) {

        String connectedAgentFacultaireUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        String faculteAgent =
universityAgentService.getAgentByMatricule(connectedAgentFacultaireUsern
ame).getFaculte().getNom();

        BigDecimal sum =
paymentRepository.findByTypeFraisAndEnregistrementStatus(type, status)
                .stream()
                .filter(payment -> {
                    return
faculteAgent.equalsIgnoreCase(payment.getFaculte());
                })
                .map(Payment::getMontant)
                .reduce(BigDecimal.ZERO, BigDecimal::add);

        return sum.toString();
    }
```

```
    @Override
    public StudentFinancialStatus loadStudentFinancialStatus() {
        String connectedStudentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        EtudiantResponse student =
universityStudentService.getStudentByMatricule(connectedStudentUsername)
;

        BigDecimal totalPayed =
paymentRepository.findByTypeFrais("frais_academique")
                .stream().filter(payment -> {
                    return
connectedStudentUsername.equalsIgnoreCase(payment.getMatriculeEtudiant()
);
                })
                .map(Payment::getMontant)
                .reduce(BigDecimal.ZERO, BigDecimal::add);

        BigDecimal totalFraisPromotion = new
BigDecimal(student.getPromotion().getTotal_frais());
        BigDecimal fraisRestant =
totalFraisPromotion.subtract(totalPayed);
        LocalDateTime lastPaymentDate =
paymentRepository.findDerniereDatePayementParMatriculeEtudiant(connected
StudentUsername);
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-
MM-yyyy");
        boolean isThereDatas = lastPaymentDate != null;

        return StudentFinancialStatus.builder()
                .status("En règle")
                .totalFrais(student.getPromotion().getTotal_frais())
                .fraisRestant(fraisRestant.toString())
                .totalPayed(totalPayed.toString())
                .lastPayementDate(isThereDatas ?
lastPaymentDate.format(formatter) : "-")
                .build();
    }

    @Override
    public List<PaymentResonseThatStudentAsksFor>
loadTheThreeLatestStudentPayment() {
        String connectedStudentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        return
paymentRepository.findTop3ByMatriculeEtudiantOrderByCreatedDateDesc(conn
ectedStudentUsername)
                .stream()
                .map(PaymentMapper::toPaymentResonseThatStudentAsksFor)
                .toList();
    }

    @Override
    public List<PaymentResonseThatStudentAsksFor>
```

```
loadAllTheStudentPayment() {
        String connectedStudentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        return
paymentRepository.findAllByMatriculeEtudiantOrderByCreatedDateDesc(conne
ctedStudentUsername)
                .stream()
                .map(PaymentMapper::toPaymentResonseThatStudentAsksFor)
                .toList();
    }

    @Override
    public List<TotalAmountPaymentsPerMonth> loadMyPaymentsTendancy() {
        String connectedStudentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        List<TotalAmountPaymentsPerMonth> totalAmountPaymentsPerMonths =
new ArrayList<>();

        LocalDateTime maintenant = LocalDateTime.now();
        for (int i = 0; i < 5; i++) {
            LocalDateTime debutMois =
maintenant.minusMonths(i).withDayOfMonth(1);
            LocalDateTime finMois =
debutMois.with(TemporalAdjusters.lastDayOfMonth())

.withHour(23).withMinute(59).withSecond(59).withNano(999_999_999);

            String nomMois = debutMois.getMonth().toString();

            totalAmountPaymentsPerMonths.add(new
TotalAmountPaymentsPerMonth(
                    nomMois,
                    getTotalAmountPayments(connectedStudentUsername,
debutMois, finMois)
                    ));
        }

        return totalAmountPaymentsPerMonths;
    }

    @Override
    public List<String> getDepartementByFaculte() {
        String connectedAgentFacultaireUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        String faculteAgent =
universityAgentService.getAgentByMatricule(connectedAgentFacultaireUsern
ame).getFaculte().getNom();
        return paymentRepository.getPromotionByFaculty(faculteAgent);
    }

    @Override
    public List<String> getAvailableTypeFraisByDepartment(String
department) {
```

```
        return paymentRepository.getTypeFraisByPromotion(department);
    }

    @Override
    public List<PaymentResponse>
getPaymentsForTheirPrinting(PrintPaymentsListRequest request, String
faculte) {

        LocalDateTime startDate =
(LocalDate.parse(request.getFirstDate())).atStartOfDay();
        LocalDateTime endDate =
(LocalDate.parse(request.getLastDate())).atStartOfDay();

        return paymentRepository.getPaymentsForTheirPrinting(
                faculte,
                request.getDepartment(),
                request.getTypeFrais(),
                startDate,
                endDate
        ).stream().map(PaymentMapper::toPaymentDto).toList();
    }

    @Override
    public byte[] generatePaymentListInPdfFile(String htmlContext) {
        return PaymentsListFileGenerator.getBytes(htmlContext);
    }

    public String getTotalAmountPayments(String matricule, LocalDateTime
startDate, LocalDateTime endDate) {
        BigDecimal amountFound =
paymentRepository.getTotalPaymentsByStudentAndDateRange(matricule,
startDate, endDate);
        return amountFound != null ? amountFound.toString() : "0";
    }









    @Override
    public List<ForCashExpressPaymentResponse>
findAllMyOwnBorderauxButPerTypeFrais(String typeFrais) {
        String connectedCashExpreAgentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        return paymentRepository

.findAllByTypeFraisAndUserAuthorMatriculeOrderByCreatedDateDesc(typeFrai
s, connectedCashExpreAgentUsername)
                .stream()
                .map(PaymentMapper::mapToForCashExpressPaymentResponse)
                .toList();
    }

    @Override
```

```
    public List<ForCashExpressPaymentResponse> findMyLastTwoScans() {
        String connectedCashExpreAgentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        return paymentRepository

.findTop2ByUserAuthorMatriculeOrderByCreatedDateDesc(connectedCashExpreA
gentUsername)
                .stream()
                .map(PaymentMapper::mapToForCashExpressPaymentResponse)
                .toList();
    }

    @Override
    public List<ForCashExpressPaymentResponse> findAllMyOwnBorderaux() {
        String connectedCashExpreAgentUsername =
SecurityContextHolder.getContext().getAuthentication()
                .getName();
        return paymentRepository

.findAllByUserAuthorMatriculeOrderByCreatedDateDesc(connectedCashExpreAg
entUsername)
                .stream()
                .map(PaymentMapper::mapToForCashExpressPaymentResponse)
                .toList();
    }


}
```