



UNIVERSITÉ DES SCIENCES, DES TECHNIQUES ET DES
TECHNOLOGIES DE BAMAKO (USTTB)

FACULTÉ DES SCIENCES ET TECHNIQUES

MÉMOIRE DE MASTER 2
SPÉCIALITÉ : INGÉNIERIE LOGICIELLE & SYSTÈME D'INFORMATION

**CONCEPTION ET MISE EN ŒUVRE DE MICRO
SERVICES POUR LE SYSTÈME DE GESTION
INTÉGRÉ DE CERTAINES UNIVERSITÉS DE
BAMAKO**

PRÉSENTÉ PAR NAMAKAN KEITA

Sous la direction de
DR. JACQUELINE KONATE - MAÎTRE DE CONFÉRENCES

CO-ENCADREURS :

DR BOUKAYE TRAORE
DR ADAMA COULIBALY

MAÎTRE-ASSISTANT
MAÎTRE-ASSISTANT

Année 2022

*Dédicace
Je dédie ce mémoire :*

*À mon oncle feu Adama KEITA qui m'a beaucoup soutenu et aidé le long de mes études.
J'aurai aimé que ce jour soit en sa présence. Puisse Dieu dans sa miséricorde infinie,
l'accueillir dans son paradis !*

Remerciements

Mes remerciements vont tout premièrement à Dieu, le Tout-Puissant, pour la volonté, la santé et la patience qu'Il m'a accordé durant toutes ces années d'études.

J'adresse toute ma reconnaissance à la directrice de ce mémoire, Dr SOGOBA Jacqueline KONATÉ, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à la réalisation de ce mémoire. Je remercie également les co-encadreurs Dr Boukaye TRAORE et Dr Adama COULIBALY.

Je remercie toute l'équipe TELIMAN, particulièrement M. Abdoulaye MAIGA le Chief technical officer (CTO) de TELIMAN, pour son aide à la réalisation de ce travail et en acceptant que je puisse poursuivre mes études sans problème.

Je remercie également mon binôme Mohamed DIALLO et les autres, membres de l'équipe pour leur encouragement quotidien Yaya SANOGO, Paul O DIASSANA, Alou DIARRA, Dramane DIALLO, Aïcha DEMBELE et Pierre DIARRA.

Mes remerciements s'adressent à la Faculté des Sciences et Techniques pour la qualité de ses programmes et enseignements et particulièrement au DER de Mathématiques et Informatique qui est composé d'enseignants toujours disponibles et joignables pour assurer la bonne formation des étudiants.

Je souhaite adresser mes remerciements les plus sincères à tout le corps enseignant et administratif de l'Université des Sciences, des Techniques et des Technologies de Bamako (USTTB) pour la richesse et la qualité de leurs enseignements et qui déplient de grands efforts pour assurer une formation actualisée.

SIGLES ET ABRÉVIATIONS

API	Application Programming Interface
BD	Base de données
CD	Continuous Delivery
CI	Continuous Integration
CTO	Chief technical officer
CV	Curriculum Vitae
DB	Data Base
EC	Élément constitutif
FAPH	Faculté de Pharmacie
FDPRI	Faculté de Droit Privé
FDPU	Faculté de Droit Public
FHG	Faculté d'Histoire et de Géographie
FMOS	Faculté de Médecine et d'Odontostomatologie
FSAP	Faculté des Sciences Administratives et Politiques
FSEG	Faculté de Sciences Économique et de Gestion
FST	Faculté des Sciences et Techniques de Bamako
HTML	HyperText Markup Language
IDE	Integrated Development Environment
IP	Internet Protocol
ISA	Institut des Sciences Appliquées
IUDT	Institut Universitaire de Développement Territorial
IUG	Institut Universitaire de Gestion
JAR	Java ARchive
SGBD	Système de Gestion de Base de Données
SQL	Structured Query Language
UE	Unité d'enseignement
UML	Unified Modelling Language
USJPB	Université des sciences juridiques et politiques de Bamako
USSGB	Université des Sciences Sociales et de Gestion de Bamako
USTTB	Université des Sciences, des Techniques et des Technologies de Bamako
WAR	Web Application Archive

Table des matières

Remerciements	ii
SIGLES ET ABRÉVIATIONS	iii
1 Introduction	1
1.1 Présentation des Universités	1
1.1.1 L'USTTB	1
1.1.2 L'USSGB	2
1.1.3 L'USJPB	2
1.2 Contexte et problématique	2
2 Analyse des besoins et conceptions	4
2.1 Analyse des besoins	4
2.2 Analyse et conception du système	9
2.2.1 Diagramme de cas d'utilisation	9
2.2.2 Scénario et diagrammes de séquences	15
2.2.3 Diagramme de classes	23
3 Architecture et implémentation	31
3.1 Architecture du système	31
3.2 Mise en œuvre de l'application	33
3.2.1 Outils et technologies utilisés	33
3.2.2 Microservices	34
4 Méthodologie et approche de développement	41
4.1 La gestion de version du code ou versionning du code	43
4.2 Intégration continue et déploiement continu CI/CD.	43
4.2.1 Intégration continue CI	44
4.2.2 Distribution continue	44
4.3 Les branches principales	45
4.3.1 La branche develop	45
4.3.2 La branche staging	45
4.3.3 La branche prod	45
4.4 Docker	47
5 Tests fonctionnels	48

CONCLUSION ET PERSPECTIVES	50
Bibliographie	52
Références	53
ANNEXES	54
Annexe 1 : Interface Swagger-UI	54
Annexe 2 : Interface de Postman	55
Annexe 3 : Interface web - Liste des personnels administratifs	56
Annexe 4 : Interface web - Génération de la carte étudiant et l'attestation d'inscription	57
Annexe 5 : Interface web - Statistiques d'inscription par genre	58
Annexe 6 : test de la fonctionnalité authentification	59
Annexe 7 : Interface web - Les fonctionnalités testées	62

Liste des tableaux

2.1	Besoins fonctionnels	9
2.2	Description du diagramme de séquence du système relatif au cas d'utilisation "Se préinscrire"	18
2.3	Description du diagramme de séquence du système relatif au cas d'utilisation "Encaissement des paiements"	20
2.4	Description du diagramme de séquence du système relatif au cas d'utilisation "Inscription d'un candidat"	22
5.1	Fonctionnalités testées	67

Table des figures

2.1	Diagramme de cas d'utilisation 1	10
2.2	Diagramme de cas d'utilisation 2	11
2.3	Diagramme de cas d'utilisation 3	12
2.4	Diagramme de cas d'utilisation 4	13
2.5	Diagramme de cas d'utilisation 5	14
2.6	Diagramme de cas d'utilisation 6	14
2.7	Diagramme de séquence du système relatif au cas d'utilisation "Se préinscrire"	16
2.8	Diagramme de séquence du système relatif au cas d'utilisation "Encaissement des paiements"	19
2.9	Diagramme de séquence du système relatif au cas d'utilisation "Inscription d'un candidat"	21
2.10	Diagramme de classes, partie 1	23
2.11	Diagramme de classes, partie 2	25
2.12	Diagramme de classes, partie 3	28
2.13	Diagramme de classes, partie 4	30
3.1	Architecture microservices - source : docs.oracle.com	32
3.2	Architecture des dix microservices	34
3.3	Interface du microservice Eureka	35
3.4	API gateway illustration	36
3.5	Interface d'accueil de Spring Boot Admin montrant tous les microservice et leur statut	38
3.6	Détail d'un microservice	39
3.7	Journal d'évènements des microservices	40
4.1	Méthodologie agile (Figure tirée de [R1])	42
4.2	Tableau d'organisation des tâches (Figure tirée de [R2])	42
4.3	Illustration du CI/CD (Figure tirée de [R2])	44
4.4	Gestion des branches	46
4.5	Exemple de pipeline	46
4.6	Comparatif entre l'architecture docker et machine virtualisation (Figure tirée de [R3])	47
5.1	Couverture de test de la ressource structure	49
5.2	Code du test unitaire des getters d'une classe	49
5.3	Interface Swagger-UI	54
5.4	Interface de Postman	55

5.5	Interface web - Liste des personnels administratifs	56
5.6	Interface web - Génération de la carte étudiant et l'attestation d'inscription	57
5.7	Interface web - Statistiques d'inscription par genre	58
5.8	Code du test de la fonctionnalité authentification	60

Chapitre 1

Introduction

1.1 Présentation des Universités

L'université de Bamako était une université publique malienne entre 1996 et 2011. Elle est aussi connue sous le nom « d'université du Mali ».

En 2000, 19 714 étudiants et 538 enseignants occupaient les neuf (09) campus. En 2008/2009, l'université de Bamako comptait plus de 70 000 étudiants, près du double de l'effectif 2004/2005 (35 284).

L'université de Bamako dont l'effectif atteignait 80 000 étudiants en 2010/2011, est remplacée par 4 universités :

- L'université des Sciences Sociales et de Gestion de Bamako (U.S.S.G.B)
- L'université des Lettres et des Sciences Humaines de Bamako (U.L.S.H.B)
- L'université des Sciences, des Techniques et des Technologies de Bamako (U.S.T.T.B)
- L'université des Sciences Juridiques et Politiques de Bamako (U.S.J.P.B),

Ce projet concerne trois des quatre universités de Bamako : l'USTTB, l'USSGB et l'USJPB. Pour le cas de l'USJPB seule la Faculté des Sciences Administratives et Politiques (FSAP) est concernée.

1.1.1 L'USTTB

L'USTTB est un Établissement Public à caractère Scientifique, Technologique et Culturel (EPSTC). Elle est créée en septembre 2011. Elle est placée sous la tutelle du Ministre en charge de l'Enseignement Supérieur et de la Recherche Scientifique. Elle est composée de quatre (04) structures dont trois (03) facultés et d'un (01) Institut qui sont :

- La Faculté des Sciences et Techniques (FST)
- La Faculté de Médecine et d'Odontostomatologie (FMOS)
- La Faculté de Pharmacie (FAPH)
- L'Institut des Sciences Appliquées (ISA)

1.1.2 L'USSGB

L'USSGB est un Établissement Public à caractère Scientifique, Technologique et Culturel (EPSTC). Elle est créée en septembre 2011. Elle est composée de quatre (04) structures dont trois (02) facultés et d'un (02) Institut qui sont :

- La Faculté d'Histoire et de Géographie (FHG)
- La Faculté de Sciences Economique et de Gestion (FSEG)
- L'Institut Universitaire de Gestion (IUG)
- L'Institut Universitaire de Développement Territorial (IUDT)

1.1.3 L'USJPB

L'USJPB est le fruit d'une longue évolution. À l'origine, il y avait l'École d'Administration du Soudan créée en 1958, qui devint en 1963 l'École Nationale d'Administration (ENA) qui muera plus tard en Faculté des Sciences Juridiques et Économiques avant de devenir la faculté des Sciences Juridiques et Politiques, et enfin Université des Sciences Juridiques et Politiques.

L'USJPB est composée de trois (3) facultés :

- La Faculté de Droit Privé (FDPRI)
- La Faculté de Droit Public (FDPU)
- La Faculté des Sciences Administratives et Politiques (FSAP)

1.2 Contexte et problématique

Les nouvelles technologies transforment nos modes de vie et de consommation. L'information est au centre de cette révolution : elle est désormais numérisée et peut être transmise sur toutes sortes d'appareils de manière quasi instantanée. Il est impératif que les secteurs clés soient en phase avec cette avancée technologique. Les nouvelles technologies de l'information nous offrent de nombreux atouts pour y parvenir.

Le secteur de l'éducation et de l'enseignement reste énormément en marge de l'utilisation des nouvelles technologies au Mali. En effet, beaucoup de données sont manipulées dans ce secteur : les données personnelles des étudiants et du personnel enseignant et administratif, les inscriptions et les frais y afférents, les évaluations, les notes, les salles, les bâtiments, les emplois du temps, la gestion des effectivités des heures de cours, la gestion des heures supplémentaires, etc.

La gestion de ces données dans beaucoup de structures reste sous Excel et les gestionnaires de dossiers. Cette gestion des données n'est pas très pratique. Vu la taille importante de données, il n'est pas rare de perdre des données, d'avoir des données non cohérentes ou des duplications de données. Le support physique (papier) sont beaucoup utilisés, ce qui entraînent des coûts et de l'encombrement. L'accès à une donnée précise est assez difficile, le temps de recherche est assez conséquent et l'information complète est souvent morcelée,

il faut aller à droite et à gauche pour tout assembler. Pour avoir accès à des informations, il faut souvent faire des déplacements. La gestion des heures supplémentaires est un vrai casse pour les administrations.

Cette gestion des données est beaucoup fastidieuse et entraînent le plus souvent des désagrément et quelques fois des conséquences graves. Au vu de toute cette problématique, il est nécessaire d'avoir un outil de gestion intégré. Cet outil de gestion intégrée doit être le résultat d'un travail de rationalisation et d'automatisation des procédures administratives de l'université telles que l'inscription des étudiants, la gestion de leurs notes et de leurs parcours. Les avantages de ce travail seront, entre autres, la dématérialisation des procédures qui réduit grandement l'usage des supports de communication physique (papiers) et les déplacements inutiles, mais aussi et surtout une plus grande interactivité entre les différents acteurs du système.

C'est dans ce cadre que l'Université des Sciences Sociales et de Gestion de Bamako (USSGB) a initié un projet de mise en place d'un outil numérique (Système d'Information) pour la gestion intégrée de toute l'institution.

Les autres universités de Bamako sont confrontées à la même problématique décrite ci-dessus, il fut décidé par conséquent d'étendre le projet pour couvrir d'autres universités.

Chapitre 2

Analyse des besoins et conceptions

L'analyse des besoins et la conception représentent une phase importante du cycle de développement d'un logiciel.

2.1 Analyse des besoins

L'analyse des exigences se concentre sur les tâches qui déterminent les besoins ou les conditions pour répondre au produit ou projet nouveau ou modifié, en tenant compte des exigences éventuellement conflictuelles des différentes parties prenantes, en analysant, documentant, validant et gérant les logiciels ou configurations requises. [R4]

Nous avons utilisé Unified Modelling Language (UML) comme outil de modélisation. UML est un langage graphique permettant de visualiser, de spécifier, de construire et de documenter les artefacts d'un système à logiciel intensif. [B1]

L'objectif de la modélisation est de permettre l'obtention d'une vue d'ensemble du système avant sa mise en œuvre. Une première version du cahier des charges nous a été fourni par l'USSGB. Il s'ensuivit une conception du système à partir de la base du système existant à la FST. Le cahier de charges fut modifié au fil et à mesure pour s'adapter à toutes les universités du Mali.

Le public cible est une communauté d'enseignants, d'étudiants et d'administrateurs. Les fonctionnalités associées aux besoins exprimés sont présentées comme suit :

Besoins	Fonction
Gestion des candidatures	<ul style="list-style-type: none"> — Enregistrer les candidatures et avec les informations nécessaires avec possibilité d'y joindre des fichiers — Confirmer l'enregistrement de la candidature à l'intéressé — Traiter en ligne les candidatures — Notifier les résultats du traitement aux candidats
Préinscription des étudiants	<ul style="list-style-type: none"> — Compléter les informations du candidat accepté pour l'inscription — Catégoriser les candidats — Déterminer les frais dus par le candidat
Inscription des étudiants	<ul style="list-style-type: none"> — Identifier chaque étudiant en lui attribuant un numéro unique — Générer les attestations d'inscription — Générer les cartes d'étudiant avec code QR
Gestion des frais d'inscription et de scolarité	<ul style="list-style-type: none"> — Définir les différents frais d'inscription et/ou de scolarité par catégorie de candidat — Modifier les frais d'inscription et/ou de scolarité — Valider les paiements — Générer les justificatifs de paiement
Gestion du parcours des étudiants	<ul style="list-style-type: none"> — Vérifier la régularité des étudiants dans les évaluations — Faire les transferts des étudiants d'une filière à une autre — Générer une fiche de renseignement pour un étudiant

Gestion des UE et des filières	<ul style="list-style-type: none"> — Enregistrement des filières — Enregistrement des UE par filière
Gestion des notes	<ul style="list-style-type: none"> — Saisir des notes par évaluation — Calculer les moyennes avec possibilité de choisir les coefficients — Accéder aux notes de façon personnalisée — Générer les relevés de notes — Charger les anciennes notes dans le système
Gestion des réclamations de notes	<ul style="list-style-type: none"> — Formuler les réclamations — Traiter les réclamations de notes
Gestion des évaluations	<ul style="list-style-type: none"> — Enregistrer les évaluations (devoirs, examens, soutenances) — Générer la liste de présence — Générer le planning des évaluations — Rechercher les évaluations par mots clés, date, filière, UE, etc.
Gestion des documents délivrés par les universités	<ul style="list-style-type: none"> — Générer les attestations de réussite avec code QR crypté — Générer les relevés de notes par semestre et par étudiant avec code QR crypté
Authentification des documents délivrés par les universités	<ul style="list-style-type: none"> — Scanner le code QR des relevés de notes et les attestations pour vérifier leur authenticité

Publication des informations et documents internes	<ul style="list-style-type: none"> — Enregistrer des annonces rédigées ou scannées — Enregistrer les supports de cours avec possibilité de spécifier le public cible — Rechercher les annonces par mots clés, par date, par public cible, etc.
Gestion des emplois du temps	<ul style="list-style-type: none"> — Définir les séances de cours, de TD, TP par classe — Générer les emplois du temps des classes et des groupes de TD/TP — Partager les emplois du temps
Consultation de la liste des collaborateurs (trombinoscope)	<ul style="list-style-type: none"> — Identifier les collaborateurs par rôle, fonction, missions et coordonnées — Répertorier les compétences dans la communauté
Gestion des ressources humaines	<ul style="list-style-type: none"> — Enregistrer les enseignants et le personnel administratif et techniques — Gestion de la carrière de tout le personnel
Gestion des heures supplémentaires	<ul style="list-style-type: none"> — Enregistrer les heures effectuées par département et par semestre — Générer des états de paiement par banque — Consulter les heures supplémentaires par les enseignants avec possibilité de faire des réclamations

Gestion des salles et des équipements	<ul style="list-style-type: none"> — Enregistrer les bâtiments avec leur localisation — Enregistrer les salles par bâtiment avec leurs capacités — Enregistrer les équipements, leur état, leur caractéristique et leur emplacement — Réserver en ligne les salles et les équipements — Consulter la disponibilité des salles et des équipements — Enregistrer la mise hors service des équipements défectueux
Gestion du courrier	<ul style="list-style-type: none"> — Enregistrer le courrier — Consulter le courrier — Retracer le courrier
Gestion des offres de stages	<ul style="list-style-type: none"> — Enregistrer les offres de stages en provenance des entreprises — Postuler aux offres de stages
Gestion de la bibliothèque	<ul style="list-style-type: none"> — Enregistrer tous les ouvrages de la bibliothèque de l'Université — Enregistrer les prêts et les restitutions de livres — Suivre les délais de restitution et alerter en cas de dépassement de délai — Réserver des livres à l'avance

Génération de statistiques	<ul style="list-style-type: none"> — Faire des statistiques sur les inscriptions des étudiants par structure — Faire des statistiques par filière — Faire des statistiques par genre — Faire les statistiques sur les évaluations en dégagent les taux de réussite et d'échec par semestre, par an, par UE, etc. — Faire les statistiques sur les paiements des frais d'inscription — Faire les statistiques sur les courriers — Faire les statistiques sur les heures supplémentaires — Faire les statistiques sur le personnel par genre et en faisant des prévisions par rapport aux départs à la retraite — Faire les statistiques sur la fréquentation de la bibliothèque et les prêts d'ouvrages — Faire des statistiques suivant les indicateurs souhaités par les universités
Discussions et échanges	<ul style="list-style-type: none"> — Créer des fora de discussions et Chats entre collaborateurs — Faire des Foires Aux Questions (FAQ)

TABLE 2.1: Besoins fonctionnels

2.2 Analyse et conception du système

2.2.1 Diagramme de cas d'utilisation

Un diagramme de cas d'utilisation [B2] est une utilisation du système qui produit un résultat observable et fournit généralement de la valeur à une ou plusieurs entités qui interagissent avec le système. C'est une perspective externe sur le système du point de vue de l'utilisateur, mais ne montrent pas comment le système fonctionne en interne.

La figure 2.1 présente l'ensemble des cas d'utilisations des acteurs « Étudiant » et « Candidat ».

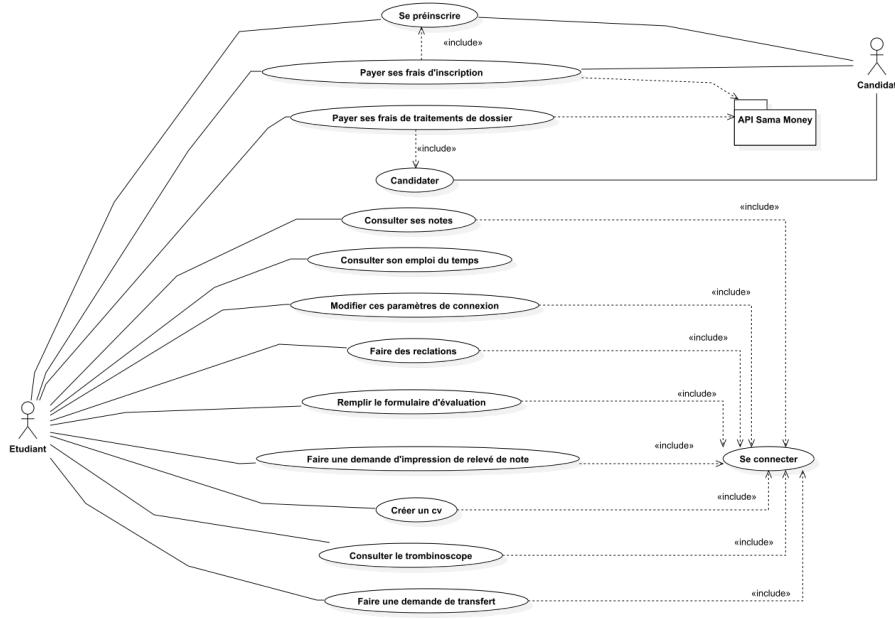


FIGURE 2.1 – Diagramme de cas d'utilisation 1

Un candidat peut :

- se préinscrire dans une option,
- payer ses frais d'inscriptions après sa préinscription. Le paiement des frais d'inscriptions dépend de l'API (Application Programming Interface ou « interface de programmation d'application ») Sama money
- candidater dans une option,
- payer ses frais de traitements de dossier après sa candidature. Le paiement des frais d'inscriptions dépend de l'API Sama money

Un étudiant après s'être connecté au système peut :

- consulter ses notes,
- consulter son emploi du temps,
- Modifier ces paramètres de connexion (nom d'utilisateur, mot de passe),
- Faire des réclamations,
- Remplir le formulaire d'évaluation d'une unité d'enseignement (UE),
- Faire une demande d'impression de relevé de note,
- Créer un Curriculum vitæ (CV),
- Consulter le trombinoscope,
- Faire une demande de transfert dans une autre option.

Un étudiant sans s'être connecté au système peut :

- se préinscrire dans une option,

- payer ses frais d’inscriptions après sa préinscription. Le paiement des frais d’inscriptions dépend de l’API (Application Programming Interface ou « interface de programmation d’application ») Sama money
- candidater dans une option,
- payer ses frais de traitements de dossier après sa candidature. Le paiement des frais d’inscriptions dépend de l’API Sama money,

La figure 2.2 présente l’ensemble des cas d’utilisations des acteurs « Enseignant » et « Chef de département ».

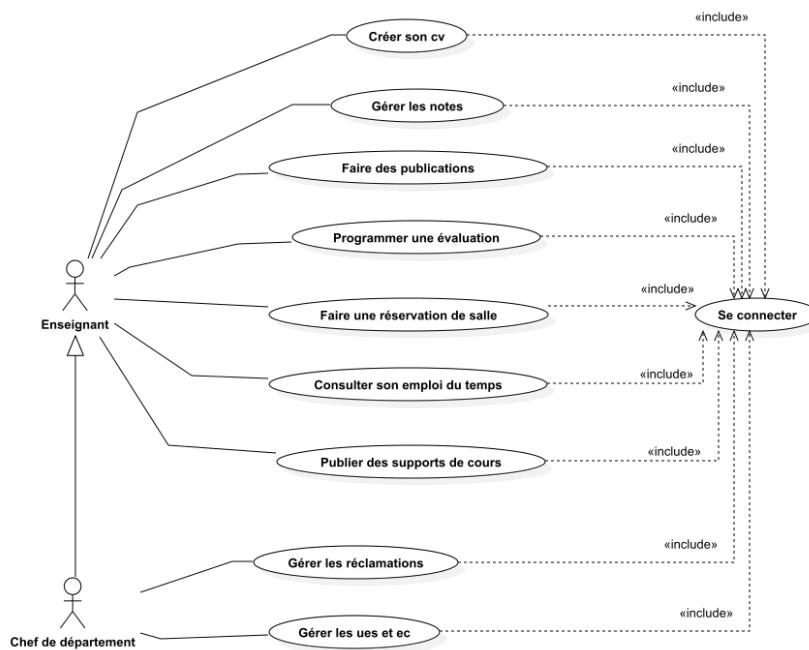


FIGURE 2.2 – Diagramme de cas d’utilisation 2

Le chef de département hérite de l’enseignant, de ce fait le chef de département hérite de tous les cas d’utilisation de l’enseignant.

L’enseignant après s’être connecté au système peut :

- se préinscrire dans une option,
- créer son CV,
- gérer les notes,
- faire des publications,
- programmer une évaluation,
- réserver une salle,
- consulter son emploi du temps,

- publier des supports de cours.

Le chef de département après s'être connecté peut :

- gérer les réclamations,
- gérer les UE et éléments constitutifs (EC).

La figure 2.3 présente l'ensemble des cas d'utilisations des acteurs « Comptable », « Chef comptable », « Secrétaire » et « Secrétaire principale ».

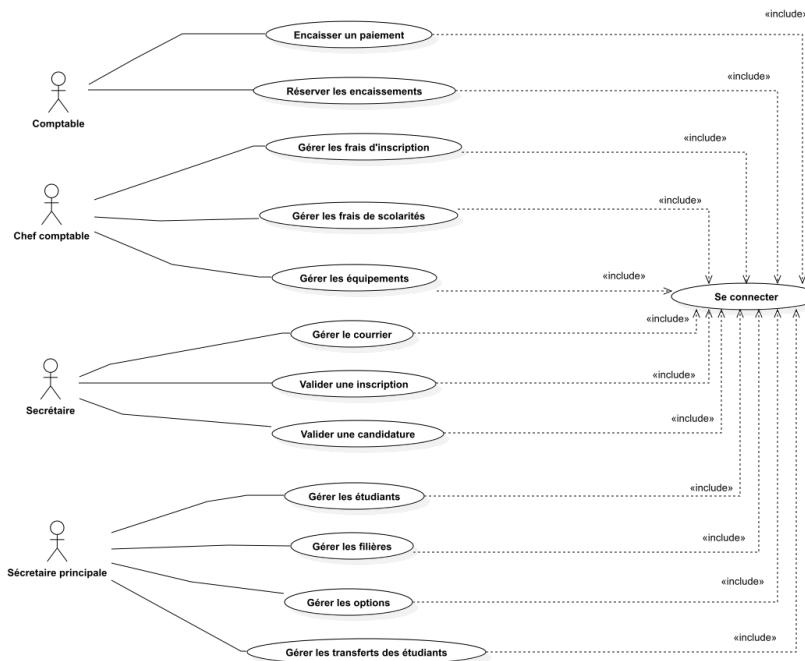


FIGURE 2.3 – Diagramme de cas d'utilisation 3

Le comptable après s'être connecté au système peut :

- encaisser un paiement (frais d'inscription, de scolarité, de traitement de dossier, de relevé de note, de carte scolaire, de diplôme),
- reversé ses encaissements au chef comptable,

Le chef comptable après s'être connecté au système peut :

- gérer les frais d'inscription, de scolarité, de traitement de dossier, de relevé de note, de carte scolaire, de diplôme,
- gérer les équipements,

Le secrétaire après s'être connecté au système peut :

- gérer le courrier,
- valider une inscription,

- valider une candidature,

Le secrétaire principal après s'être connecté au système peut :

- gérer les étudiants,
- gérer les filières,
- gérer les options,
- gérer les transferts des étudiants

La figure 2.4 présente l'ensemble des cas d'utilisations de l'acteur « Bibliothécaire ».

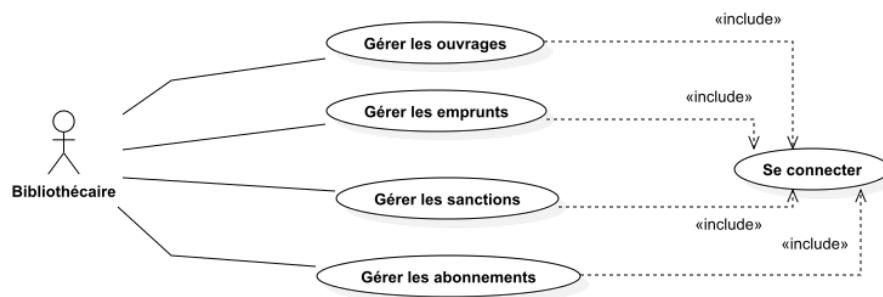


FIGURE 2.4 – Diagramme de cas d'utilisation 4

Le bibliothécaire après s'être connecté au système peut :

- gérer les ouvrages (livres, articles, mémoires des étudiants),
- gérer les emprunts,
- gérer les abonnements à la bibliothèque,
- gérer les sanctions (les sanctions peuvent survenir après qu'un ouvrage a été endommagé, rendu en retard...),

La figure 2.5 présente l'ensemble des cas d'utilisations des acteurs « Administrateurs » et « Utilisateur ». Tous les types d'acteurs sont des utilisateurs excepté l'acteur candidat. La figure 2.6 présente l'héritage entre « Utilisateur » et tous les acteurs.

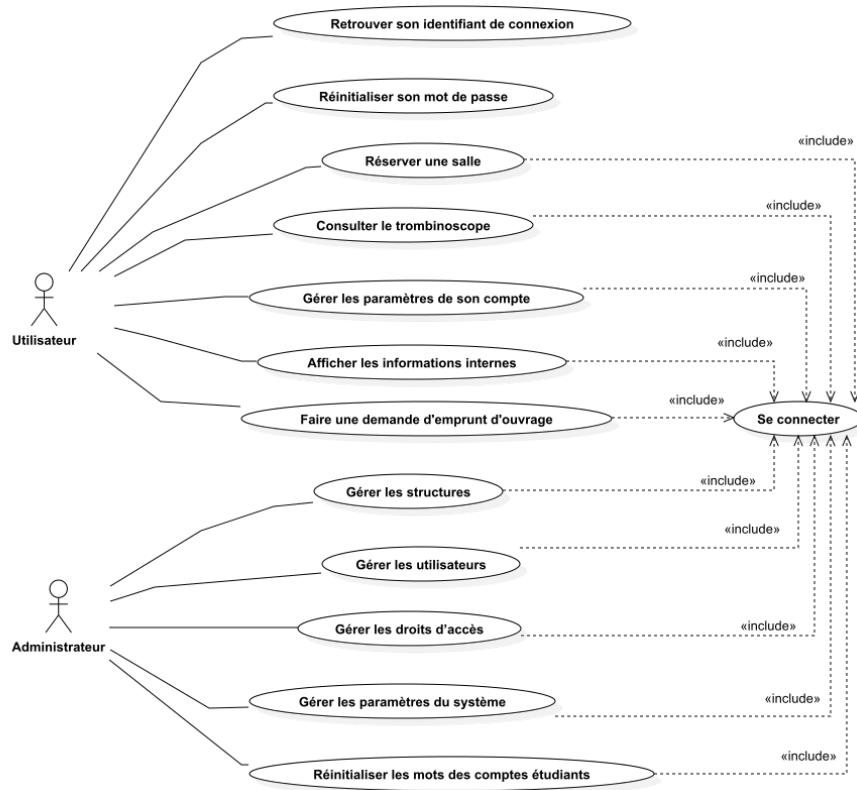


FIGURE 2.5 – Diagramme de cas d'utilisation 5

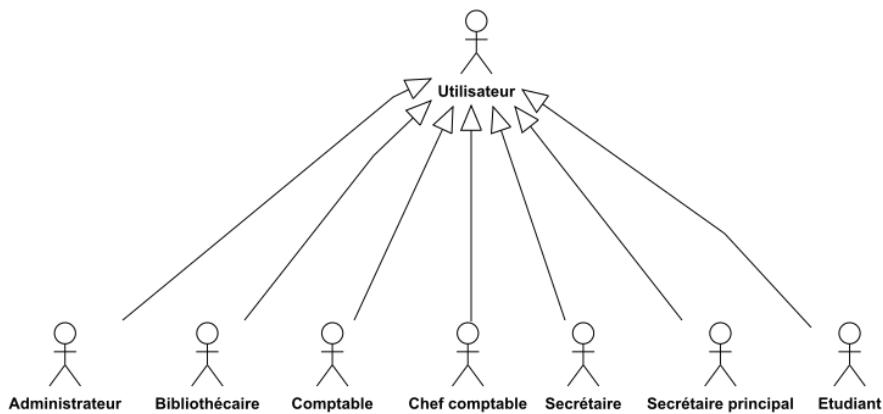


FIGURE 2.6 – Diagramme de cas d'utilisation 6

Un utilisateur après s'être connecté au système peut :

- réserver une salle,
- consulter le trombinoscope,

- gérer les paramètres de son compte (changer son identifiant et mot de passe),
- afficher les informations internes de sa structure,
- faire une demande d'emprunt d'ouvrage.

Un utilisateur sans s'être connecté au système peut :

- retrouver son identifiant de connexion,
- réinitialiser son mot de passe.

Un administrateur après s'être connecté au système peut :

- gérer les structures,
- gérer les utilisateurs dans sa globalité : il peut ajouter ou mettre à jour les informations utilisateur,
- gérer les droits d'accès des utilisateurs,
- gérer les paramètres du système (conditions de passage, évènements, années scolaires...),
- réinitialiser les mots des comptes étudiants.

2.2.2 Scénario et diagrammes de séquences

Un scénario représente une succession particulière d'enchaînements s'exécutant du début à la fin du cas d'utilisation, un enchaînement étant l'unité de description de séquences d'actions. Un cas d'utilisation contient en général un scénario nominal et plusieurs enchaînements alternatifs qui se terminent de façon normale ou par des échecs. On parle donc tantôt de cas favorable pour un succès, tantôt de cas défavorable pour un échec.

Un diagramme de séquence [B2] décrit la séquence des opérations au cours d'un scénario, d'un cas d'utilisation du système et détermine quel objet exécute chaque opération.

Scénario et diagramme de séquence du cas d'utilisation préinscription

La figure 2.7 présente les diagrammes de séquence correspondants aux scenarii favorables et alternatifs présentés ci-dessus.

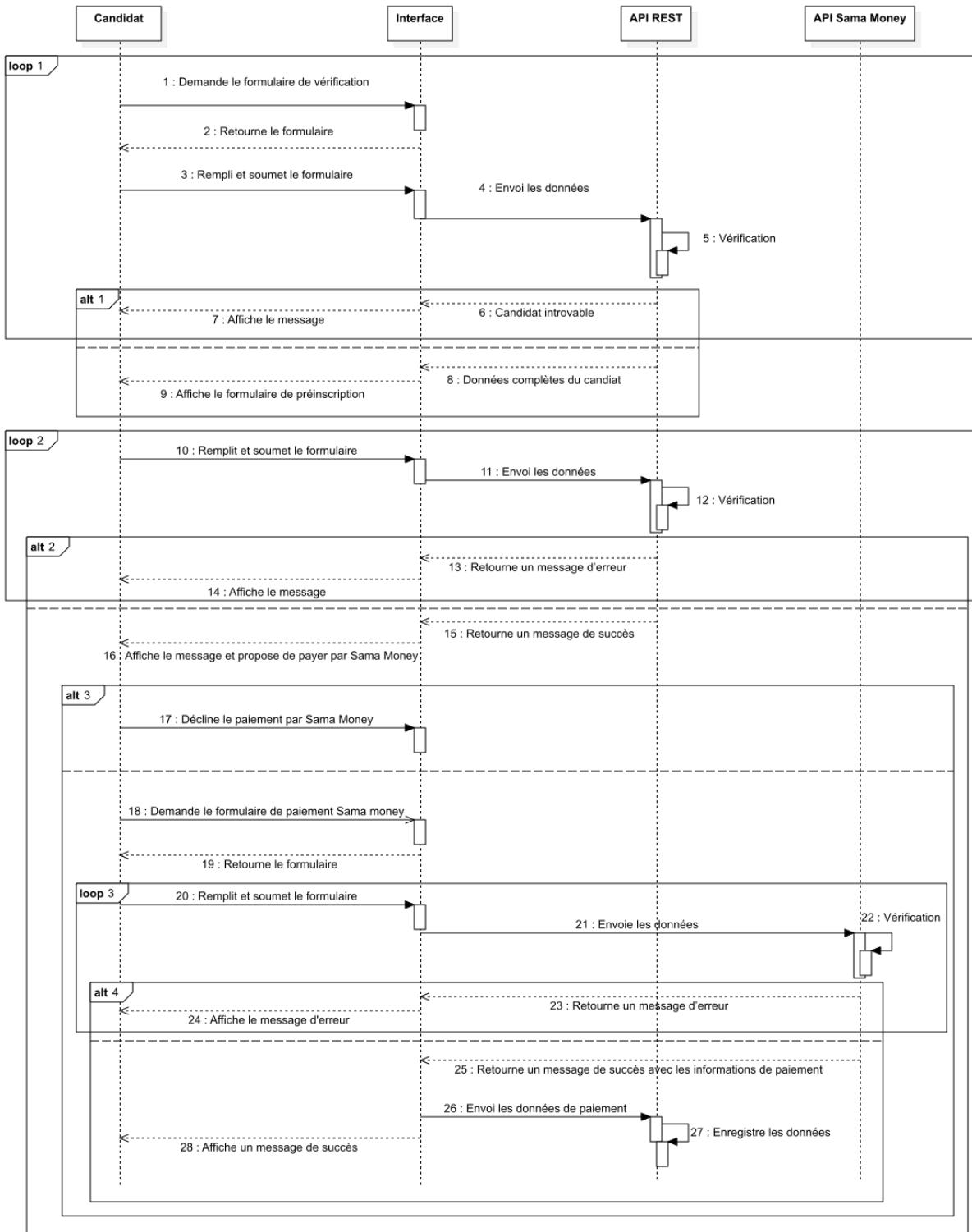


FIGURE 2.7 – Diagramme de séquence du système relatif au cas d'utilisation "Se préinscrire"

La description de la figure 2.7 se trouve dans le tableau 2.2.

Acteur possible	Candidat
Pré-conditions	Néant
Post conditions	Le candidat est préinscrit
Scénario nominal	<ol style="list-style-type: none"> 1. Le système front (interface) lui présente le formulaire de vérification 2. Le candidat remplit et soumet le formulaire au système 3. Le système front (interface) envoie les données à l'API REST 4. L'API REST vérifie les données et envoie la réponse au système front 5. Le système front présente le formulaire de préinscription 6. Le candidat remplit et soumet le formulaire au système front 7. Le système front (interface) envoie les données à l'API REST 8. L'API REST vérifie les données et met à jour les données du candidat et envoie la réponse au système front 9. Le système front (interface) affiche un message de succès et propose le paiement par Sama money et affiche le formulaire de paiement 10. Le candidat remplit et soumet le formulaire au système front 11. Le système front (interface) envoie les données à l'API Sama Money 12. L'API Sama Money vérifie les données, enregistre le paiement et renvoie les données de paiement au système front 13. Le système front (interface) affiche un message de succès et envoie les données paiement à l'API REST 14. L'API REST sauvegarde les données dans la base de données.

Alternatives	<p>4.a L'API REST n'a pas trouvé de candidat correspondant aux informations du formulaire soumis à l'étape 2</p> <ul style="list-style-type: none"> — Le système signale l'échec au candidat et lui propose à nouveau le formulaire. Le cas d'utilisation redémarre à l'étape 1 du scénario nominal <p>8.a L'API REST n'a pas pu mettre à jour dans la base des données les informations du formulaire soumis à l'étape 6</p> <ul style="list-style-type: none"> — Le système signale l'échec au candidat et lui propose à nouveau le formulaire de l'étape 5. <p>9.a Le candidat décline le paiement par Sama money, il est rédigé vers la page d'accueil.</p> <p>12.a L'API Sama n'a pas pu prélever les frais d'inscriptions au compte fournis</p> <ul style="list-style-type: none"> — Le système signale l'échec au candidat et lui propose à nouveau le formulaire de l'étape 9.
Exigences supplémentaires	<ul style="list-style-type: none"> — Tous les champs du formulaire sont obligatoires à l'exception de la rue et la porte — Les documents doivent être obligatoirement au format PDF ou JPEG

TABLE 2.2: Description du diagramme de séquence du système relatif au cas d'utilisation "Se préinscrire"

Scénario et diagramme de séquence du cas d'utilisation encaissement des paiements

La figure 2.8 présente les diagrammes de séquence correspondants aux scenarii favorables et alternatifs présentés ci-dessus.

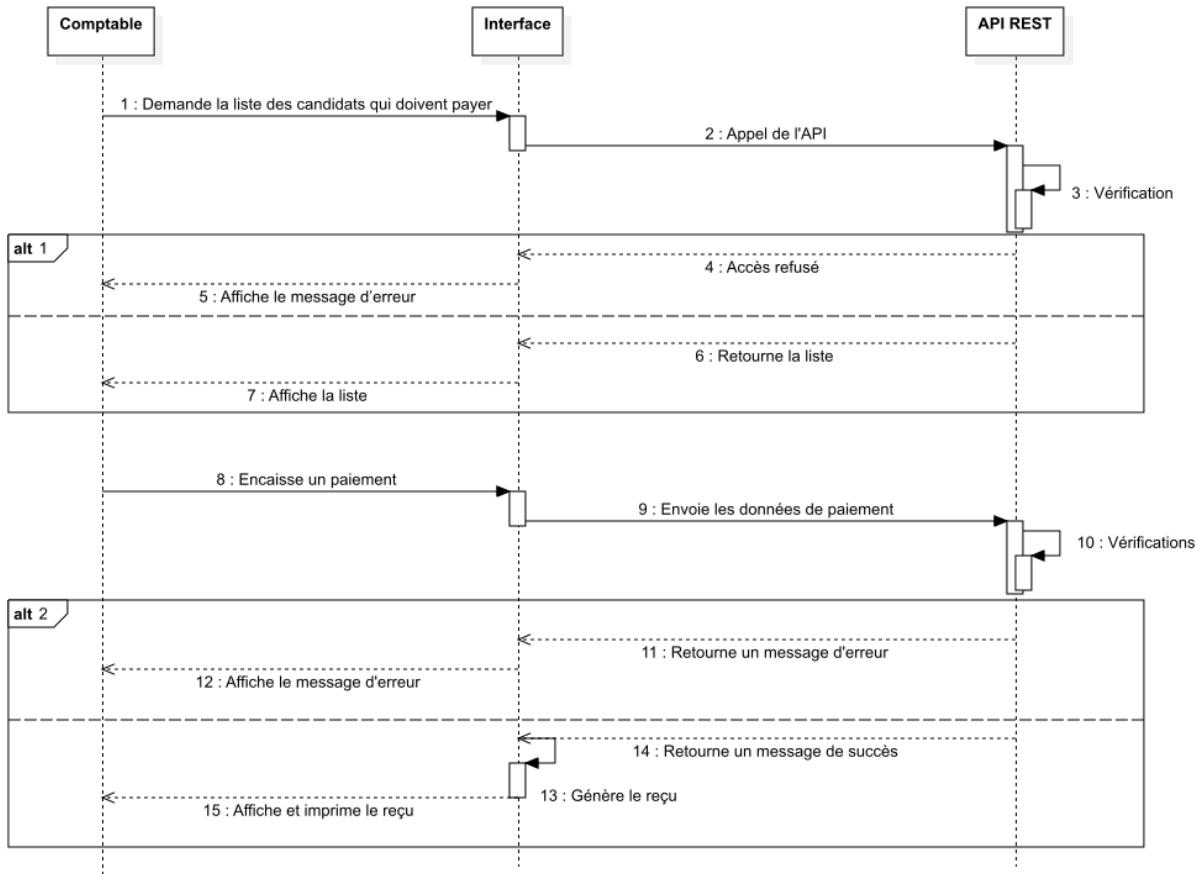


FIGURE 2.8 – Diagramme de séquence du système relatif au cas d'utilisation "Encaissement des paiements"

La description de la figure 2.8 se trouve dans le tableau 2.3.

Acteur possible	Comptable
Pré-conditions	Néant
Post conditions	Frais d'inscription payé
Scénario nominal	<ol style="list-style-type: none"> 1. Le comptable demande la liste des candidats qui doivent payer leur frais d'inscription 2. Le système front (interface) envoie la requête à l'API REST 3. L'API REST vérifie le rôle du comptable et envoie la liste au système front 4. Le système front (interface) affiche la liste des candidats 5. Le comptable valide le paiement d'un candidat 6. Le système front (interface) envoie les données de paiement à l'API REST 7. L'API REST vérifie les données et met à jour les données de préinscription du candidat et envoie les données et envoie la réponse au système front 8. Le système front (interface) génère l'attestation de paiement, l'affiche et l'imprime
Alternatives	<p>3.a Le comptable n'a pas le droit requis pour voir la liste, l'API retourne un message d'erreur</p> <p>7.a L'API REST n'a pas pu valider le paiement. Les raisons peuvent être les suivantes :</p> <ul style="list-style-type: none"> — Le candidat a déjà payé les frais d'inscription — Le numéro de quittance existe déjà pour un autre paiement.
Exigences supplémentaires	

TABLE 2.3: Description du diagramme de séquence du système relatif au cas d'utilisation "Encaissement des paiements"

Scénario et diagramme de séquence du cas d'utilisation inscription d'un candidat

La figure 2.9 présente les diagrammes de séquence correspondants aux scénarios favorables et alternatifs présentés ci-dessus.

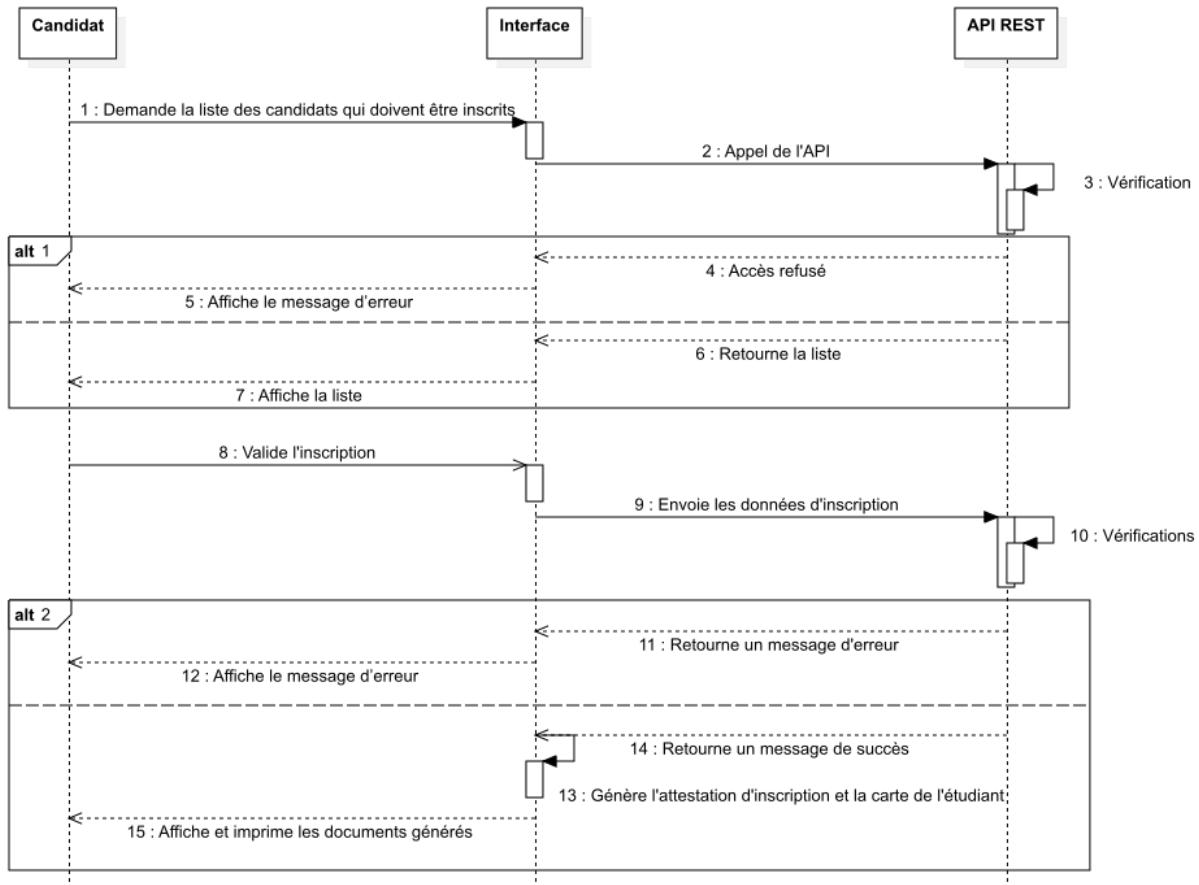


FIGURE 2.9 – Diagramme de séquence du système relatif au cas d'utilisation "Inscription d'un candidat"

La description de la figure 2.9 se trouve dans le tableau 2.4.

Acteur possible	Comptable
Pré-conditions	Néant
Post conditions	Frais d'inscription payé
Scénario nominal	<ol style="list-style-type: none"> 1. Le secrétaire demande la liste des candidats qui doivent être inscrits 2. Le système front (interface) envoie la requête à l'API REST 3. L'API REST vérifie le rôle du secrétaire et envoie la liste au système front 4. Le système front (interface) affiche la liste des candidats 5. Le secrétaire valide l'inscription d'un candidat 6. Le système front (interface) envoie les données de paiement à l'API REST 7. L'API REST vérifie les données et met à jour les données de préinscription du candidat, créé un compte étudiant et envoie les données et envoie la réponse au système front 8. Le système front (interface) génère l'attestation de d'inscription, la carte étudiante, l'affiche et l'imprime
Alternatives	<p>3.a Le secrétaire n'a pas le droit requis pour voir la liste, l'API retourne un message d'erreur</p> <p>7.a L'API REST n'a pas pu valider l'inscription. Les raisons peuvent être les suivantes :</p> <ul style="list-style-type: none"> — Le candidat a déjà payé inscrit — Le candidat a une adresse email, numéro de téléphone déjà existant.
Exigences supplémentaires	

TABLE 2.4: Description du diagramme de séquence du système relatif au cas d'utilisation "Inscription d'un candidat"

2.2.3 Diagramme de classes

L'analyse des besoins du système, la modélisation des cas d'utilisation et les scénarii nous ont permis de dresser une des différents concepts qui sera utilisé par le système. Un diagramme de classes [B3] est un modèle statique qui montre les classes et les relations entre les classes qui restent constantes dans le système au fil du temps. Le diagramme de classes décrit les classes, qui incluent à la fois des comportements et des états, avec les relations entre les classes. Compte tenu de la taille du diagramme de classes, pour apporter plus de lisibilité, le diagramme fut scindé en quatre parties. Les figures 2.10, 2.11, 2.12 et 2.13 représentent les différentes parties de notre diagramme de classes.

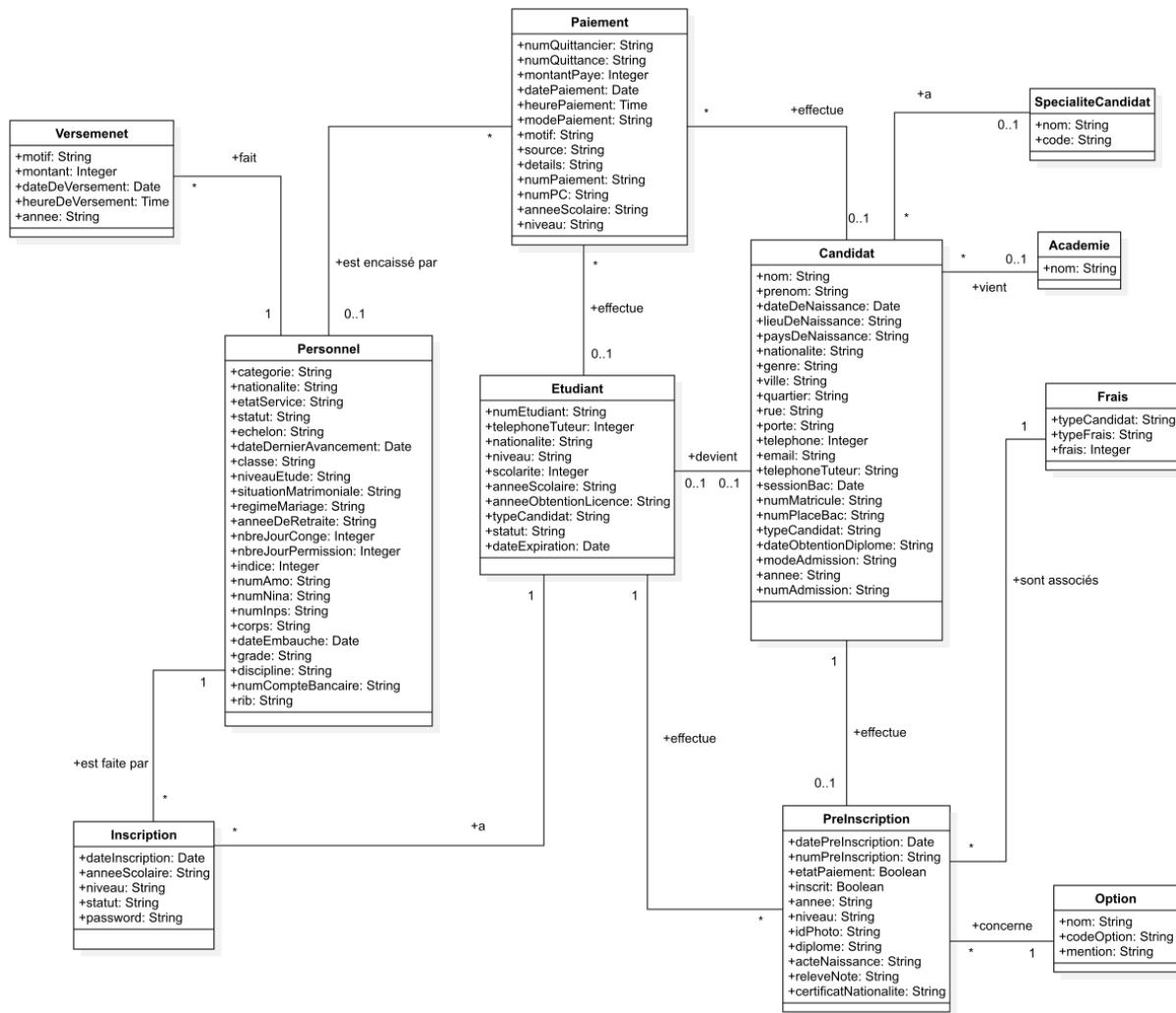


FIGURE 2.10 – Diagramme de classes, partie 1

La description de la figure 2.10 se trouve ci-dessous.

- un **CANDIDAT** peut effectuer une seule **PREINSCRIPTION**, une **PREINSCRIPTION** concerne un seul **CANDIDAT** ;

- Une **PREINSCRIPTION** se fait dans une seule **OPTION**, une **OPTION** peut être concernée par zéro ou plusieurs **PREINSCRIPTION** ;
- une **OPTION** appartient à une seule **FILIERE**, une **FILIERE** au moins une **OPTION** ;
- des **FRAIS** sont associés à zéro ou une plusieurs **PREINSCRIPTION**, une **PREINSCRIPTION** a un seul **FRAIS** ;
- un **CANDIDAT** peut effectuer zéro ou plusieurs **PAIEMENT**, un **PAIEMENT** est effectué par zéro ou un **CANDIDAT** ;
- un **CANDIDAT** peut devenir un ou zéro étudiant **ETUDIANT**, un **ETUDIANT** est un ou zéro **CANDIDAT** ;
- un **ETUDIANT** peut effectuer zéro ou plusieurs **PREINSCRIPTION**, une **PREINSCRIPTION** peut concerner un seul **ETUDIANT** ;
- un **ETUDIANT** peut effectuer zéro ou plusieurs **PAIEMENT**, un **PAIEMENT** est effectué par zéro ou un **ETUDIANT** ;
- une **INSCRIPTION** concerne un seul **ETUDIANT**, un **ETUDIANT** a zéro ou plusieurs **INSCRIPTION**
- une **INSCRIPTION** est enregistré par un seul **PERSONNEL**, un **PERSONNEL** enregistre zéro ou plusieurs **INSCRIPTION**
- un **PERSONNEL** encaisse zéro ou plusieurs **PAIEMENT**, un **PAIEMENT** est encaissé par zéro ou un **PERSONNEL**
- un **PERSONNEL** effectue zéro ou plusieurs **VERSEMENT**, un **VERSEMENT** est effectué un seul **PERSONNEL**.

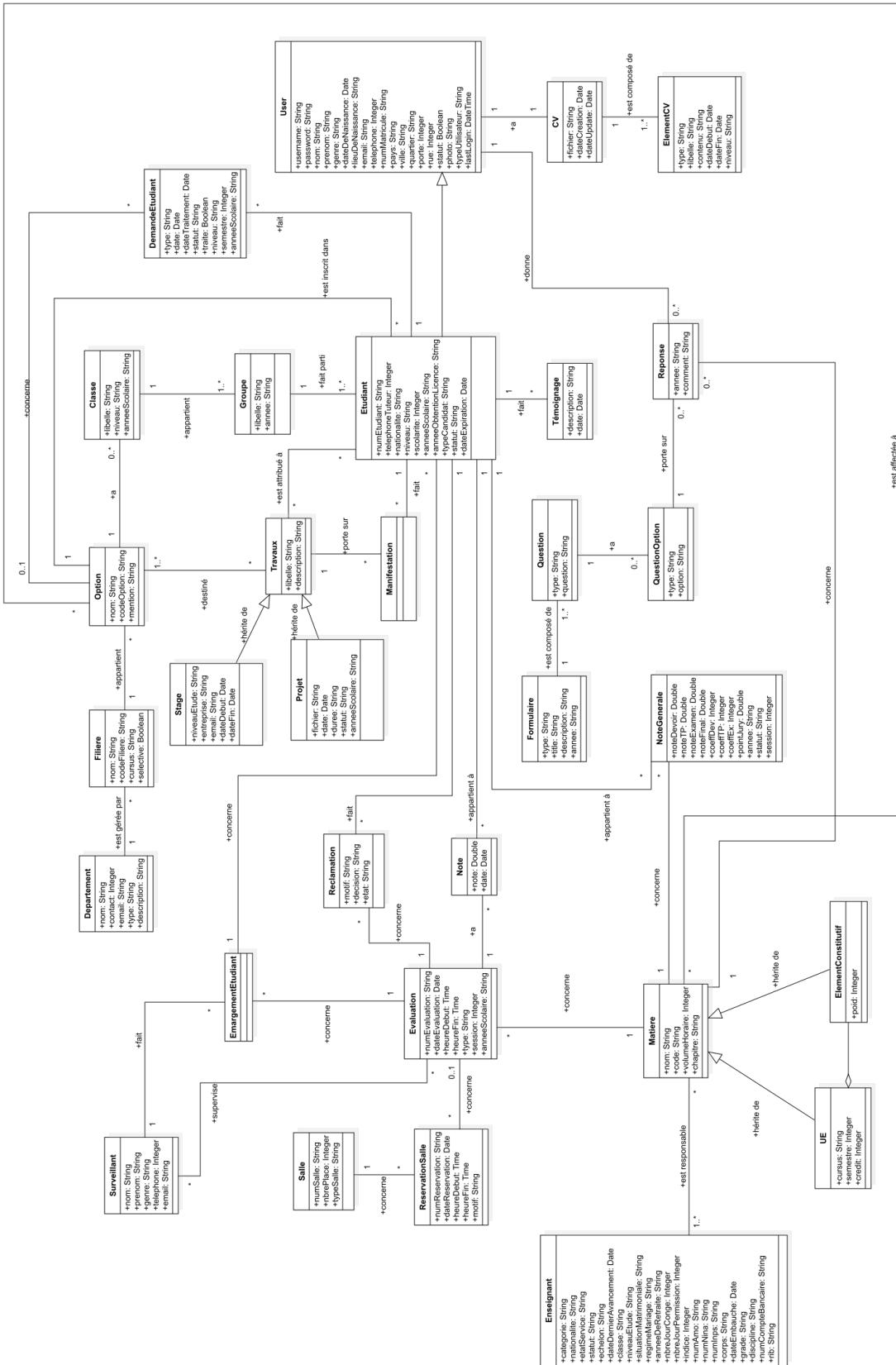


FIGURE 2.11 – Diagramme de classes, partie 2

La description de la figure 2.11 se trouve ci-dessous.

- un **ETUDIANT** hérite d'un **USER** ;
- un **USER** a un seul **CV**, un **CV** appartient à un seul **USER** ;
- un **CV** est composé d'un ou plusieurs **ELEMENTCV**, un **ELEMENTCV** est associé à un seul **CV** ;
- un **ETUDIANT** peut faire zéro ou plusieurs **TEMOIGNAGE**, un **TEMOIGNAGE** est fait par un seul **ETUDIANT** ;
- un **ETUDIANT** appartient à un seul **GROUPE**, un **GROUPE** est composé d'au moins un **ETUDIANT** ;
- un **GROUPE** appartient à une seule **CLASSE**, une **CLASSE** est composée d'au moins un **GROUPE** ;
- un **CLASSE** appartient à une seule **OPTION**, une **OPTION** a zéro ou plusieurs **CLASSE** ;
- une **FILIERE** est gérée à un seul **DEPARTEMENT**, un **DEPARTEMENT** a zéro ou plusieurs **FILIERE** ;
- un **ETUDIANT** peut faire zéro ou plusieurs **DEMANDEETUDIANT**, une **DEMANDEETUDIANT** est faite par un seul **ETUDIANT** ;
- une **DEMANDEETUDIANT** peut concerner zéro ou une **OPTION**, une **OPTION** peut être concernée par zéro ou plusieurs **DEMANDEETUDIANT** ;
- un **PROJET** hérite de **TRAVAUX** ;
- un **STAGE** hérite de **TRAVAUX** ;
- un **ETUDIANT** peut être affecté à un ou plusieurs **TRAVAUX**, un "**TRAVAUX**" est traité par un ou plusieurs **ETUDIANT** ;
- un **ETUDIANT** peut faire zéro ou une **MANIFESTATION** pour un "**TRAVAUX**", une **MANIFESTATION** est faite par un seul **ETUDIANT** et concerne un seul "**TRAVAUX**" ;
- un "**TRAVAUX**" est pour une seule **OPTION**, une **OPTION** a zéro ou plusieurs "**TRAVAUX**" ;
- un **ETUDIANT** a zéro ou plusieurs **NOTE**, une **NOTE** appartient à un seul **ETUDIANT** ;
- une **NOTE** est pour une seule **EVALUATION**, une **EVALUATION** a zéro ou plusieurs **NOTE** ;
- une UE hérite de **MATIERE** ;
- un **ELEMENTCONSTITUTIF** hérite de **MATIERE** ;
- une **EVALUATION** concerne une seule **MATIERE**, une **MATIERE** est concernée par zéro ou plusieurs **EVALUATION** ;
- un **ELEMENTCONSTITUTIF** est agrégation de **UE** ;
- une **RESERVATIONSALLE** est faite pour zéro ou une **EVALUATION**, **EVALUATION** a zéro ou plusieurs **RESERVATIONSALLE** ;

- une **RESERVATIONSALLE** est faite pour une seule **SALLE**, une **SALLE** a zéro ou plusieurs **RESERVATIONSALLE** ;
- un **SURVEILLANT** supervise zéro ou plusieurs **EVALUATION**, une **EVALUATION** est supervisé par zéro ou plusieurs **SURVEILLANT** ;
- un **SURVEILLANT** fait zéro ou plusieurs **EMARGEMENTETUDIANT**, un **EMARGEMENTETUDIANT** est fait par un seul **SURVEILLANT** ;
- un **EMARGEMENTETUDIANT** est faite pour une seule **EVALUATION**, une **EVALUATION** a zéro ou plusieurs **EMARGEMENTETUDIANT** ;
- un **ETUDIANT** a un seul **EMARGEMENTETUDIANT** pour une seule **EVALUATION** donnée, pour une **EVALUATION** donnée zéro plusieurs **ETUDIANT** ont un seul **EMARGEMENTETUDIANT** ;
- un **ETUDIANT** fait zéro ou plusieurs **RECLAMATION**, une **RECLAMATION** est faite par un seul **ETUDIANT** ;
- une **RECLAMATION** est faite pour une seule **EVALUATION**, une **EVALUATION** est concernée par zéro ou plusieurs **RECLAMATION** ;
- une **MATIERE** est dispensée par un ou plusieurs **ENSEIGNANT**, un **ENSEIGNANT** dispense zéro ou plusieurs **MATIERE** ;
- une **NOTEGENERALE** concerne une seule **MATIERE**, une **MATIERE** est concernée par zéro ou plusieurs **NOTEGENERALE** ;
- un **ETUDIANT** a zéro ou plusieurs **NOTEGENERALE**, une **NOTEGENERALE** appartient à un seul **ETUDIANT** ;
- un **FORMULAIRE** est composé d'un ou plusieurs **QUESTION**, une **QUESTION** concerne un seul **FORMULAIRE** ;
- une **QUESTION** a zéro ou plusieurs **QUESTIONOPTION**, une **QUESTIONOPTION** concerne une seule **QUESTION** ;
- une **QUESTIONOPTION** a zéro ou plusieurs **RESPONSE**, une **RESPONSE** concerne une seule **QUESTIONOPTION** ;
- une **RESPONSE** concerne zéro ou une **UE**, une **UE** est concernée par zéro ou plusieurs **RESPONSE** ;
- une **RESPONSE** concerne zéro ou une **ELEMENTCONSTITUTIF**, une **ELEMENTCONSTITUTIF** est concernée par zéro ou plusieurs **RESPONSE** ;
- une **RESPONSE** est donnée par un seul **USER**, un **USER** donne zéro ou plusieurs **RESPONSE** ;
- une **MATIERE** est dispensée dans zéro ou plusieurs **OPTION**, dans une **OPTION** est dispensée zéro ou plusieurs **MATIERE**.

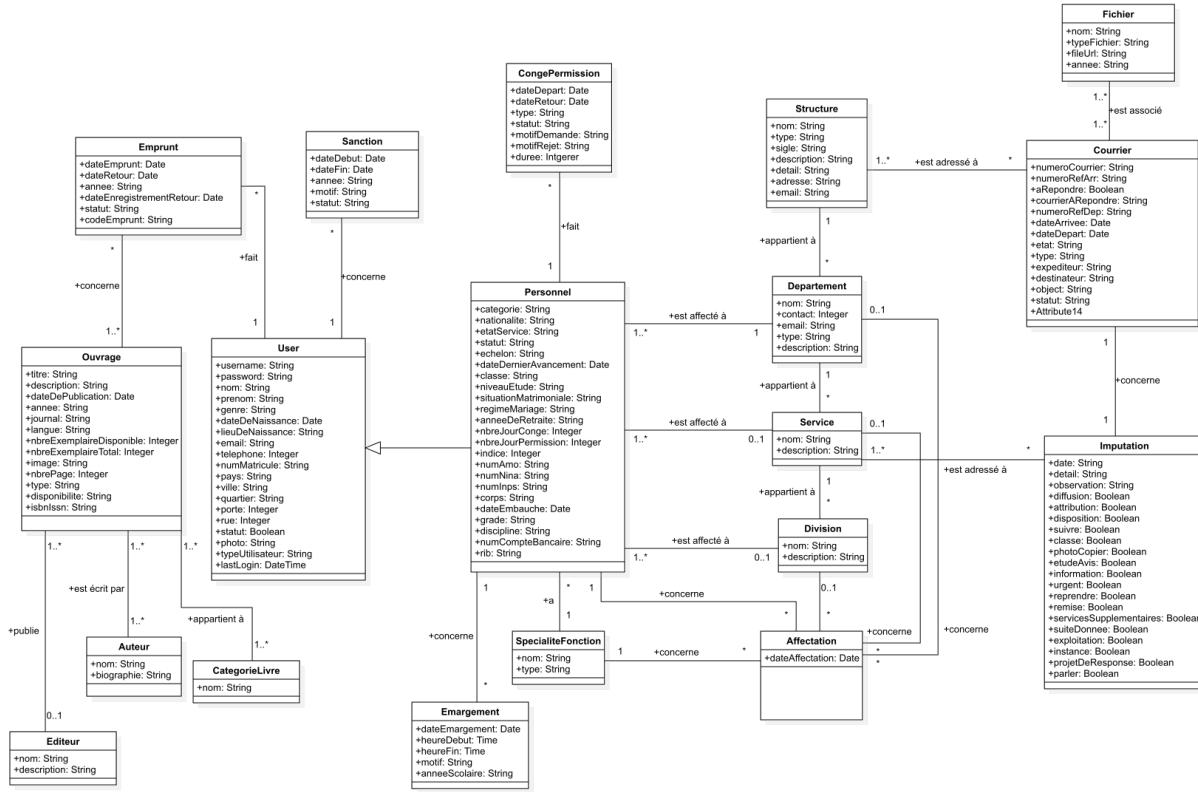


FIGURE 2.12 – Diagramme de classes, partie 3

La description de la figure 2.12 se trouve ci-dessous.

- un **PERSONNEL** hérite d'un **USER** ;
- un **PERSONNEL** est affecté à un seul **DEPARTEMENT**, un **DEPARTEMENT** est composé d'un ou plusieurs **PERSONNEL** ;
- un **PERSONNEL** est affecté à zéro ou un seul **SERVICE**, un **DEPARTEMENT** est composé d'un ou plusieurs **PERSONNEL** ;
- un **PERSONNEL** est affecté à zéro ou une seule **DIVISION**, une **DIVISION** est composée d'un ou plusieurs **PERSONNEL** ;
- un **PERSONNEL** a zéro ou plusieurs **AFFECTATION**, une **AFFECTATION** concerne un seul **PERSONNEL** ;
- un **PERSONNEL** a une **SPECIALITEFONCTION**, une **SPECIALITEFONCTION** concerne zéro ou plusieurs **PERSONNEL**
- une **SPECIALITEFONCTION** est liée à zéro ou plusieurs **AFFECTATION**, une **AFFECTATION** a une seule **SPECIALITEFONCTION** ;
- une **AFFECTATION** concerne zéro ou une seule **DIVISION**, une **DIVISION** est concernée par zéro ou plusieurs **AFFECTATION** ;
- une **AFFECTATION** concerne zéro ou un seul **SERVICE**, un **SERVICE** est concerné par zéro ou plusieurs **SERVICE** ;

- une **AFFECTATION** concerne zéro ou un **DEPARTEMENT**, un **DEPARTEMENT** est concerné par zéro ou plusieurs **AFFECTATION** ;
- une **DIVISION** appartient à un seul **SERVICE**, un **SERVICE** a zéro ou plusieurs **DIVISION** ;
- un **SERVICE** appartient à un seul **DEPARTEMENT**, un **DEPARTEMENT** a zéro ou plusieurs **SERVICE** ;
- un **DEPARTEMENT** appartient à une seule **STRUCTURE**, une **STRUCTURE** a zéro ou plusieurs **DEPARTEMENT** ;
- une **STRUCTURE** reçoit zéro ou plusieurs **COURRIER**, un **COURRIER** est destiné à une seule **STRUCTURE** ;
- un **FICHIER** est associé à un ou plusieurs **COURRIER**, un **COURRIER** possède un ou plusieurs **FICHIER** ;
- un **COURRIER** a une seule **IMPUTATION**, une **IMPUTATION** concerne un seul **COURRIER** ;
- une **IMPUTATION** est faite pour au moins un seul **SERVICE**, un **SERVICE** est concerné par zéro ou plusieurs **IMPUTATION** ;
- un **PERSONNEL** prend zéro ou plusieurs **CONGE PERMISSION**, un **CONGE PERMISSION** concerne un seul **PERSONNEL** ;
- un **PERSONNEL** est concerné par zéro ou plusieurs **EMARGEMENTENSEIGNANT**, **EMARGEMENTENSEIGNANT** concerne un seul **PERSONNEL** ;
- un **USER** est concerné par zéro ou plusieurs **SANCTION**, une **SANCTION** concerne un seul **USER** ;
- un **USER** peut faire zéro ou plusieurs **EMPRUNT**, un **EMPRUNT** est fait par un seul **USER** ;
- un **EMPRUNT** concerne zéro ou plusieurs **OUVRAGE**, un **OUVRAGE** est concerné par zéro ou plusieurs **EMPRUNT** ;
- un **OUVRAGE** a un ou plusieurs **AUTEUR**, un **AUTEUR** écrit au moins un **OUVRAGE** ;
- un **OUVRAGE** est édité par zéro ou un **EDITEUR**, un **EDITEUR** édite au moins un **OUVRAGE** ;
- un **OUVRAGE** appartient à un ou plusieurs **CATEGORIELIVRE**, une **CATEGORIELIVRE** a au moins un **OUVRAGE**.

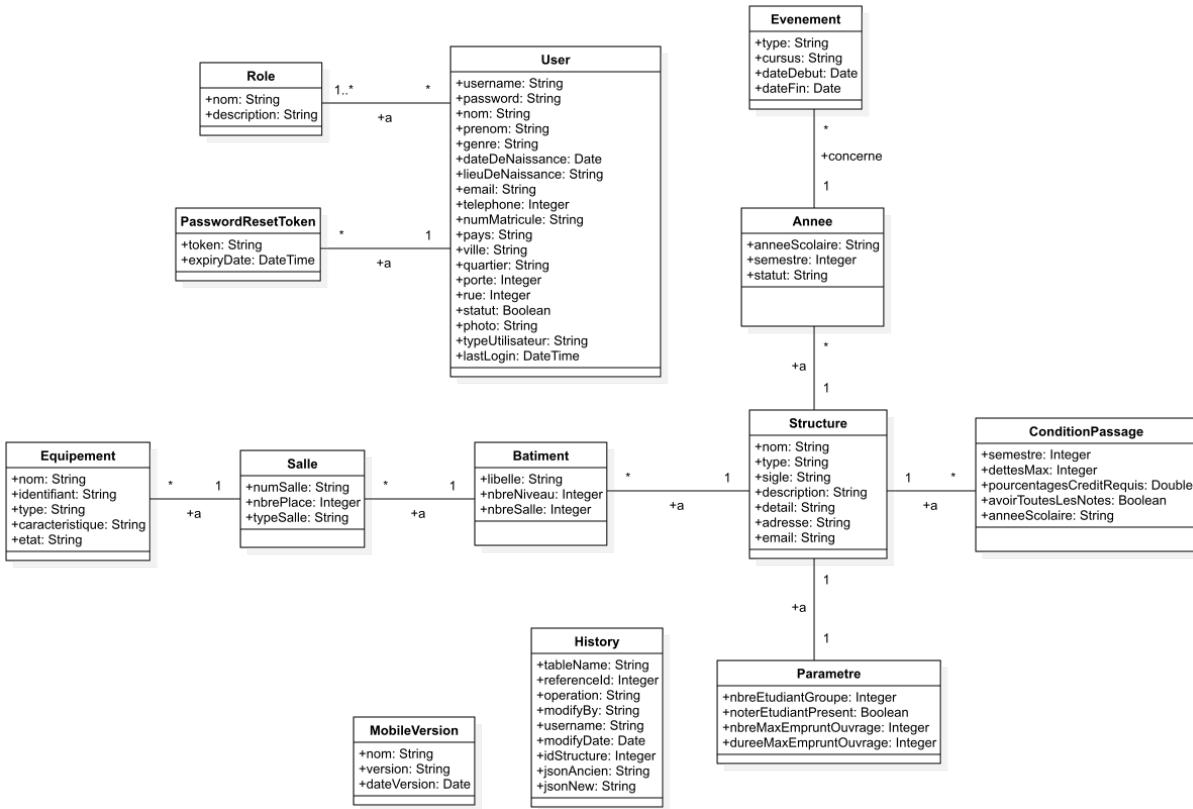


FIGURE 2.13 – Diagramme de classes, partie 4

La description de la figure 2.13 se trouve ci-dessous.

- une **STRUCTURE** a zéro ou plusieurs **CONDIDITIONPASSAGE**, une **CONDIDITIONPASSAGE** concerne une seule **STRUCTURE** ;
- une **STRUCTURE** a un seul **PARAMETRE**, un **PARAMETRE** concerne une seule **STRUCTURE** ;
- une **ANNEE** est définie pour une seule structure **STRUCTURE**, une **STRUCTURE** a zéro ou plusieurs **ANNEE** ;
- un **EVENEMENT** est défini pour une seule **ANNEE**, une **ANNEE** a zéro ou plusieurs **EVENEMENT** ;
- un **BATIMENT** appartient à une seule **STRUCTURE**, une **STRUCTURE** possède zéro ou plusieurs **BATIMENT** ;
- une **SALLE** appartient à une seule **BATIMENT**, un **BATIMENT** possède zéro ou plusieurs **SALLE** ;
- un **EQUIPEMENT** est dans une seule **SALLE**, une **SALLE** contient zéro ou plusieurs **EQUIPEMENT** ;
- un **USER** a au moins un **ROLE**, un **ROLE** est associé à zero ou plusieurs **USER** ;
- un **USER** a zéro ou plusieurs **PASSWORDRESTTOKEN**, un **PASSWORDRESTTOKEN** est généré pour un seul **USER**.

Chapitre 3

Architecture et implémentation

3.1 Architecture du système

L’architecture logicielle décrit d’une manière symbolique et schématique les différents éléments d’un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. L’architecture microservices a été adoptée pour la réalisation du projet.

L’architecture microservice [B4] est un modèle de mise en œuvre de la logique métier au sein d’une organisation à l’aide de petits services à usage unique. La figure 3.1 illustre l’architecture microservices. Cette approche contraste avec la méthode traditionnelle de construction de services monolithiques. Chaque microservice s’exécute sur son propre processus et communique entre eux via, par exemple, des API RESTful. Il existe diverses raisons de choisir une approche plutôt qu’une autre, et aucune approche n’est absolument meilleure ou pire que l’autre. Nous avons choisi l’architecture microservice pour les raisons suivantes :

1. **La séparant la logique métier** : avec l’architecture monolithique, toute la logique métier est implantée dans un seul projet global. Le code source peut devenir assez complexe et enchevêtré. La séparation de la logique métier réduit ces risques.
2. **La haute disponibilité** : il n’est pas rare d’avoir des bugs dans une partie du code source, cela peut entraîner l’arrêt de l’application. Avec une architecture monolithique, il n’y a qu’une seule application, une erreur sur une partie joue sur toute l’application et entraîne un arrêt total. Avec l’architecture microservice seul le microservice concerné par le bug s’arrête, les autres microservices continueront de fonctionner.
3. **Les déploiements fréquents** : Avec la méthodologie agile (nous y reviendrons dans le chapitre 4) les déploiements sont fréquents, avec l’architecture microservice, seul le microservice concerné est redéployé. L’ensemble de l’application n’est pas paralysé.
4. **Flexibilité technologique** : Bien que cet aspect n’ait pas utilisé dans le cadre de ce projet, l’architecture microservice permet d’utiliser différents langages de programmation. Chaque microservice peut-être développé dans un langage différent.

L’architecture microservices offre d’autres avantages. Cependant, la gestion des microservices peut s’avérer complexe, nous avons été confrontés à ce problème. Beaucoup d’outils ont été développés par d’autres développeurs pour faciliter cette gestion.

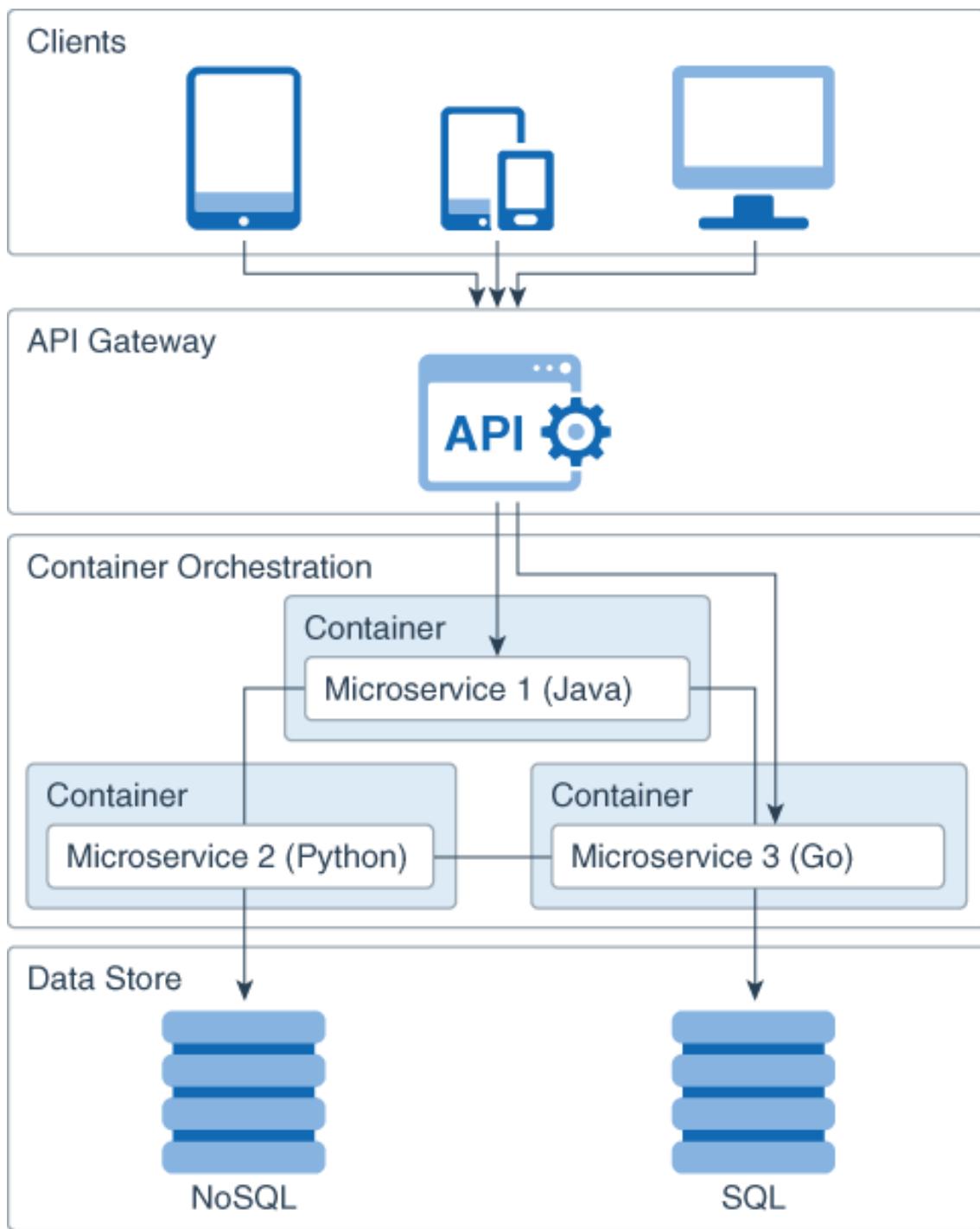


FIGURE 3.1 – Architecture microservices - source : docs.oracle.com

3.2 Mise en œuvre de l'application

3.2.1 Outils et technologies utilisés

Ci-dessous la description des outils et technologies qui ont été utilisés tout au long la mise en œuvre :

- **StarUML** [R5] est un modélisateur logiciel sophistiqué visant à prendre en charge une modélisation agile et concise.
Il a été utilisé pour l'édition des diagrammes de cas d'utilisations, de classes et de séquences que nous avons produits.
- **Java** [R6] est un langage de programmation largement utilisé pour coder des applications. C'est un choix populaire parmi les développeurs depuis plus de deux décennies. Java est un langage multiplateforme, orienté objet et centré sur le réseau qui peut être utilisé comme une plateforme en soi. Il s'agit d'un langage de programmation rapide, sécurisé et fiable pour tout coder, des applications mobiles, des logiciels d'entreprise, des applications Big Data et des technologies côté serveur.
- **SpringBoot** [R7] est un framework Java. Spring Boot facilite la création d'applications Spring autonomes de qualité production que vous pouvez "exécuter simplement".
Spring Boot a été utilisé pour le développement des différents microservices.
- **Docker** [R8] est une plate-forme ouverte pour le développement, la livraison et l'exécution d'applications. Docker vous permet de séparer vos applications de votre infrastructure afin que vous puissiez livrer rapidement des logiciels.
- **PostgreSQL** [R9] est un puissant système de base de données relationnelle objet open source avec plus de 30 ans de développement actif qui lui a valu une solide réputation de fiabilité, de robustesse des fonctionnalités et de performances.
Il a été utilisé pour la gestion des bases de données.
- **Postman** [R10] est une plate-forme API pour la création et l'utilisation d'API. Postman simplifie chaque étape du cycle de vie des API et rationalise la collaboration afin que vous puissiez créer de meilleures API plus rapidement.
Postman a été utilisé pour tester les endpoints des microservices.
- **GitLab** [R11] est une plateforme DevOps complète proposée sous la forme d'une application unique. Elle révolutionne le développement, la sécurité, l'exploitation et la collaboration entre les équipes. Créez, testez et déployez des logiciels plus rapidement en n'utilisant qu'une seule solution.
Gitlab a été utilisé pour héberger code et pour des tests et déploiement automatisés.
- **IntelliJ IDEA** [R12] est un environnement de développement très puissant qui permet de développer des applications Web complètes, grâce à ses outils intégrés, à la prise en charge de JavaScript et des technologies associées, et à la prise en charge avancée des frameworks populaires tels que Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus, Helidon. Il a été utilisé comme environnement de développement dans le cadre du projet.

3.2.2 Microservices

Le système est divisé en dix microservices. La figure 3.2 schématise les dix microservices.

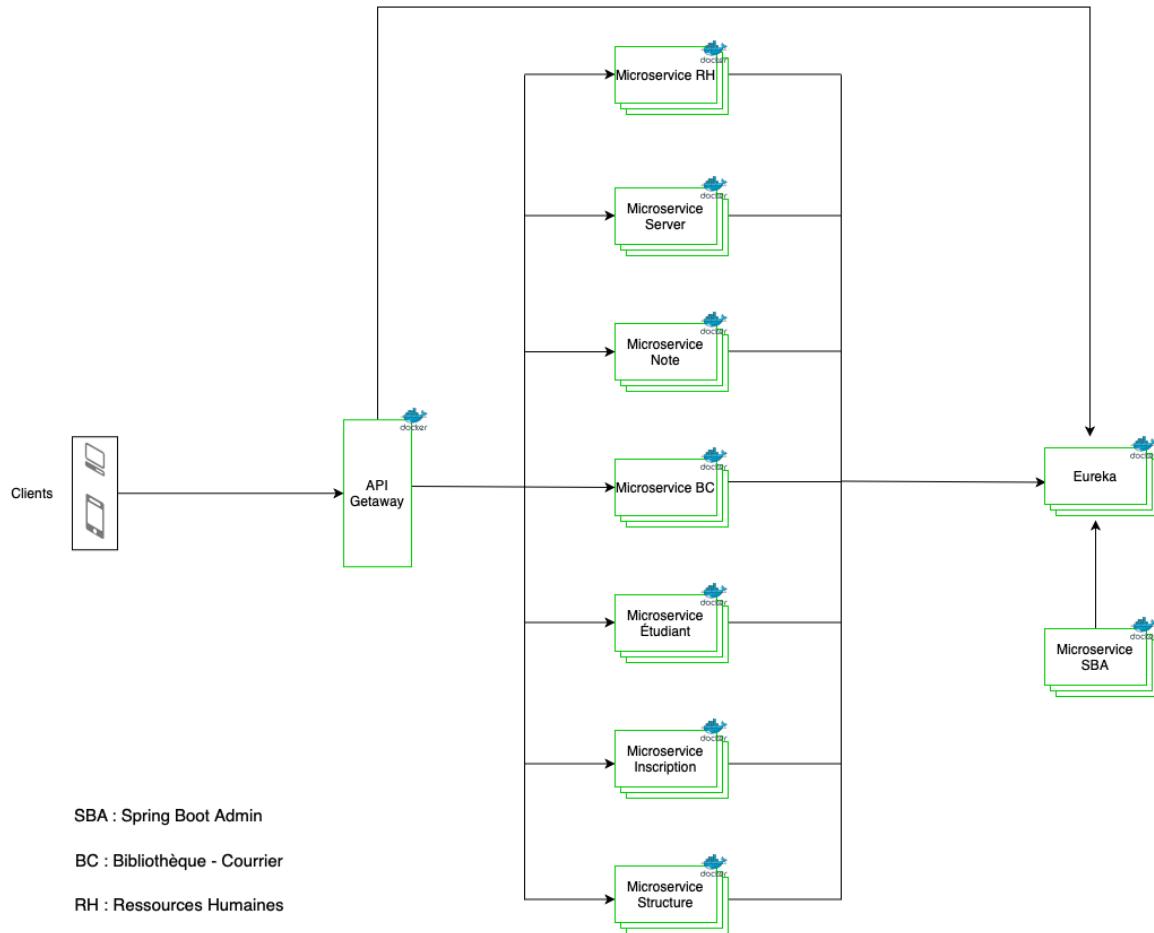


FIGURE 3.2 – Architecture des dix microservices

Les dix microservices sont présentés dans cette section.

Microservice Discovery Server

Une application basée sur l'architecture microservices s'exécute généralement dans des environnements virtualisés ou conteneurisés. Le nombre d'instances d'un service et ses emplacements changent de manière dynamique. Nous devons savoir où se trouvent ces instances et leurs noms pour permettre aux requêtes d'arriver au microservice cible.

Le Discovery Server nous aide à savoir où se trouve chaque instance. De cette manière, un Discovery Server agit comme un registre dans lequel les adresses de toutes les instances sont suivies.

Dans le cadre du projet, nous avons utilisé **Spring Cloud Netflix – Eureka** comme Discovery Server. Eureka [R13] est un système très stable, testé dans de grands déploiements

cloud avec des dizaines de milliers de noeuds. Eureka fournie également une interface web sur laquelle il est possible de visualiser les microservices qui sont enregistrés. La figure 3.3 est l'interface web d'Eureka avec des instances enregistrées.

The screenshot shows the Spring Eureka dashboard. At the top, there's a header with the Eureka logo and navigation links for 'HOME' and 'LAST 1000 SINCE STARTUP'. Below the header, the 'System Status' section displays various configuration parameters:

Environment	N/A	Current time	2022-08-15T23:07:22 +0000
Data center	N/A	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	17
		Renews (last min)	0

The 'DS Replicas' section lists instances currently registered with Eureka, each with its application name, AMIs, Availability Zones, and status:

Application	AMIs	Availability Zones	Status
ADMIN-SERVER	n/a (1)	(1)	UP (1) - 453ce698344b:admin-server:9091
API-GATEWAY-MICROSERVICE	n/a (1)	(1)	UP (1) - a155ceefaf50:api-gateway-microservice:9090
BIBLIOTHEQUE-COURRIER-MICROSERVICE	n/a (1)	(1)	UP (1) - 8a30ee1702cf:bibliothque-courrier-microservice:9096
ETUDIANT-MICROSERVICE	n/a (1)	(1)	UP (1) - 85f5ef72cb8c:etudiant-microservice:9094
INSCRIPTION-MICROSERVICE	n/a (1)	(1)	UP (1) - 2704e0ceeff7:inscription-microservice:9097
NOTE-MICROSERVICE	n/a (1)	(1)	UP (1) - 3c3d5f6e8ad3:note-microservice:9095
RESSOURCES-HUMAINES-MICROSERVICE	n/a (1)	(1)	UP (1) - 9ccbb2d942b:ressources-humaines-microservice:9093
SERVER-MICROSERVICE	n/a (1)	(1)	UP (1) - 3238f022b9c7:server-microservice:9098
STRUCTURE-MICROSERVICE	n/a (1)	(1)	UP (1) - 242e7af9083b:structure-microservice:9092

The 'General Info' section provides memory usage statistics:

Name	Value
total-avail-memory	39mb
num-of-cpus	6
current-memory-usage	36mb (92%)

FIGURE 3.3 – Interface du microservice Eureka

Microservice API gateway

L'API gateway est utilisé comme le point d'entrée de l'application. Les applications clientes ne s'adressent qu'à ce microservice et il s'occupe de router la requête vers le microservice correspondant.

Nous avons utilisé Spring Cloud Gateway comme API gateway.

Spring Cloud Gateway [R14] fournit une bibliothèque pour créer une passerelle API au-dessus de Spring WebFlux. Spring Cloud Gateway vise à fournir un moyen simple, mais efficace d'acheminer vers les API et de leur fournir des préoccupations transversales telles que : la sécurité, la surveillance/les métriques et la résilience.

Il récupère tous les microservices et ainsi que leur adresse IP auprès de Eureka.

La figure 3.4 illustre le fonctionnement d'une API gateway.

Microservice Server

Le microservice Server implémente les fonctionnalités suivantes :

- Authentification,
- Gestion des utilisateurs,
- Envoie de notification et de mail,

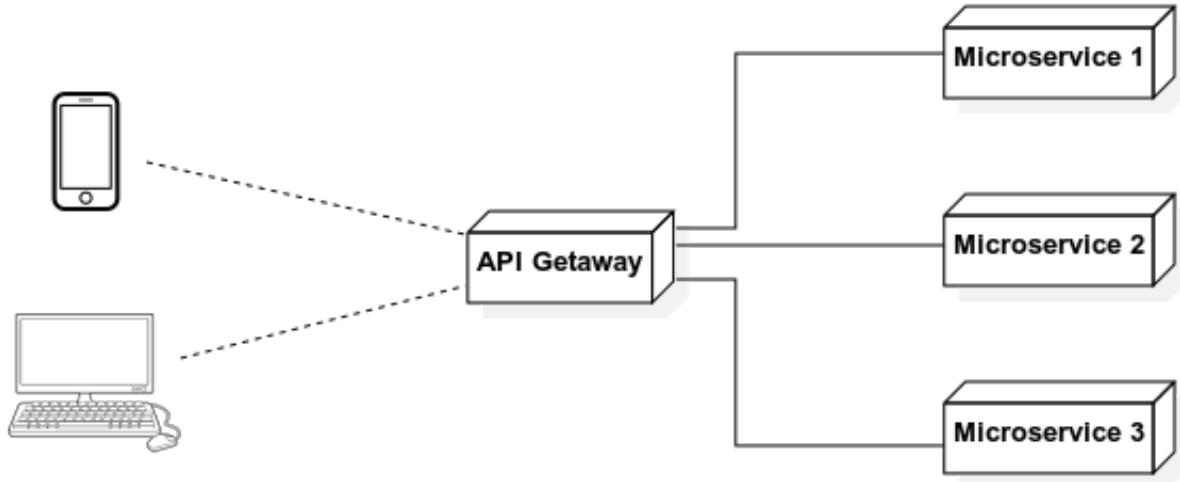


FIGURE 3.4 – API gateway illustration

- Téléversement de fichier.

Ce microservice sert aussi à déployer les nouvelles migrations dans la base de données.

Microservice Structure

Il implémente les fonctionnalités suivantes :

- Gestion des structures,
- Gestion des bâtiments,
- Gestion des salles et des équipements,
- Gestion des années scolaires,
- Gestion des paramètres du système.

Microservice Ressources humaines

Ce microservice implémente tout ce qui est en rapport avec la gestion des ressources humaines. Les fonctionnalités implémentées par ce microservice sont :

- Gestion des départements, services et divisions,
- Gestion du personnel,
- Gestion des affectations, carrières et congés du personnel,
- Gestion des effectivités et émargement des heures des enseignants.

Il permet également de donner les statistiques sur les personnels et les étudiants.

Microservice Inscription

Il implémente les fonctionnalités :

- Gestion des académies,
- Gestion des candidats,
- Gestion des frais d’inscriptions, de candidature, de scolarité, leur paiement et leur versement,
- Préinscription, candidature et inscription des étudiants et candidats,
- Affichage de la liste de passage.

Microservice Étudiant

Il implémente les fonctionnalités :

- Gestion des étudiants, groupes et classes,
- Gestion des filières et options,
- Gestion des projets et des stages,
- Gestion des témoignages des étudiants,
- Gestion des demandes des étudiants.

Microservice Note

Ce microservice implémente les fonctionnalités suivantes :

- Gestion des unités d’enseignements et éléments constitutifs,
- Gestion des évaluations,
- Gestion des notes,
- Gestion des réclamations,
- Gestion des sujets.

Microservice Bibliothèque-Courrier

Ce microservice implémente les fonctionnalités suivantes :

- Gestion des ouvrages,
- Gestion des courriers et des imputations.

Microservice Spring Boot Admin

L’une des difficultés avec l’architecture microservice est la gestion et surveillance de chaque microservice. Plus il y a des microservices, plus cette gestion est complexe.

Spring Boot Admin [R15] de codecentric est un projet communautaire pour gérer et surveiller des applications Spring Boot. Les applications s’enregistrent auprès de notre Spring Boot Admin Client (via HTTP) ou sont découvertes à l’aide de Spring Cloud (par exemple, Eureka, Consul). L’interface utilisateur n’est qu’une application Vue.js au-dessus des points de terminaison Spring Boot Actuator.

Spring Boot Admin permet de voir en tant réel le statut de chaque microservice, les métriques sur la performance et l’utilisation de mémoire, de voir leur log afin de facilement

débuguer les erreurs, ...

Dans le cadre de ce projet, Spring Boot Admin utilise Eureka pour administrer tous les microservices enregistrés auprès de lui. La figure 3.5 est l'accueil de l'interface web de Spring Boot Admin. Nous pouvons visualiser sur cette page tous les microservices enregistrés auprès d'Eureka ainsi que leur statut et le nombre d'instances. Il y a trois statuts :

- **Le vert** : ce statut indique que le microservice est en ligne et complètement fonctionnel,
- **Le gris** : ce statut indique que le microservice est indisponible, cela peut être dû à une perte de connexion entre le microservice et Eureka ou le microservice s'est arrêté.
- **Le rouge** : ce statut indique que le microservice est en ligne, mais n'est pas complètement fonctionnel.

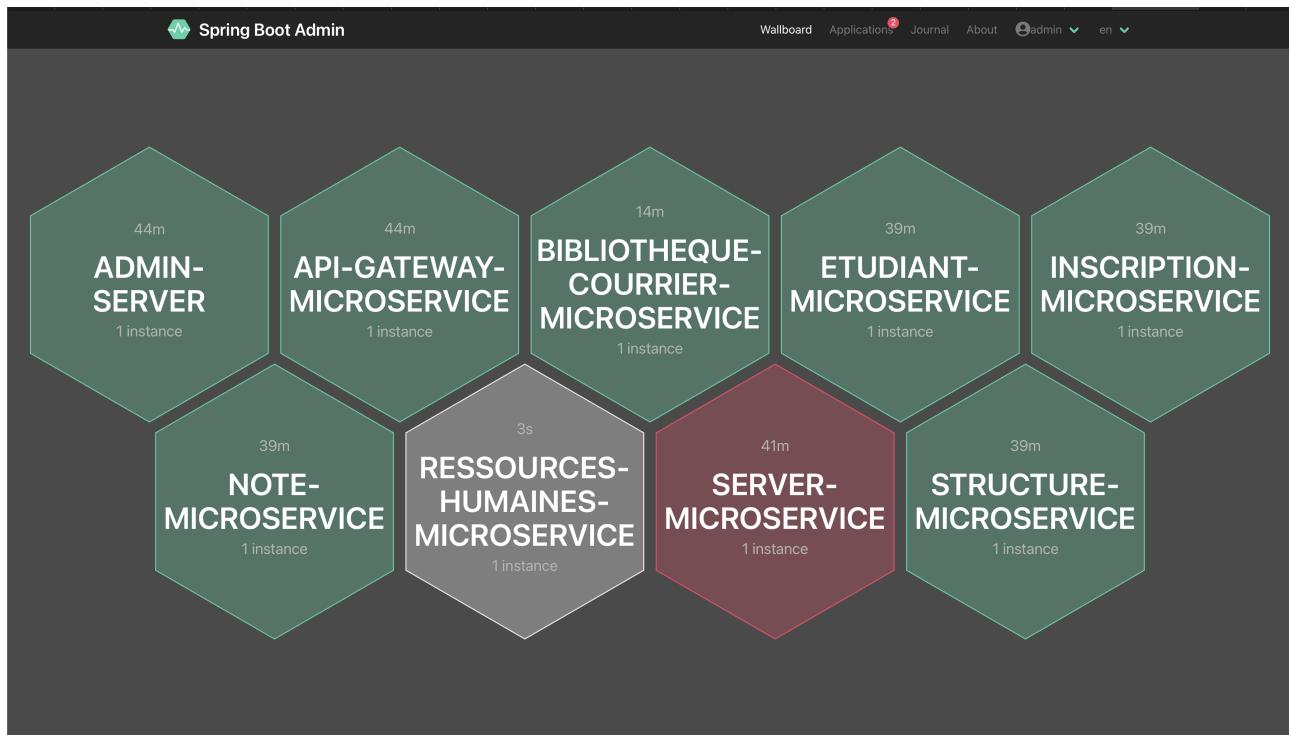


FIGURE 3.5 – Interface d'accueil de Spring Boot Admin montrant tous les microservice et leur statut

La figure 3.6 est la page détail d'un microservice. Sur cette page, nous pouvons visualiser les métriques, log, etc du microservice en temps réel.

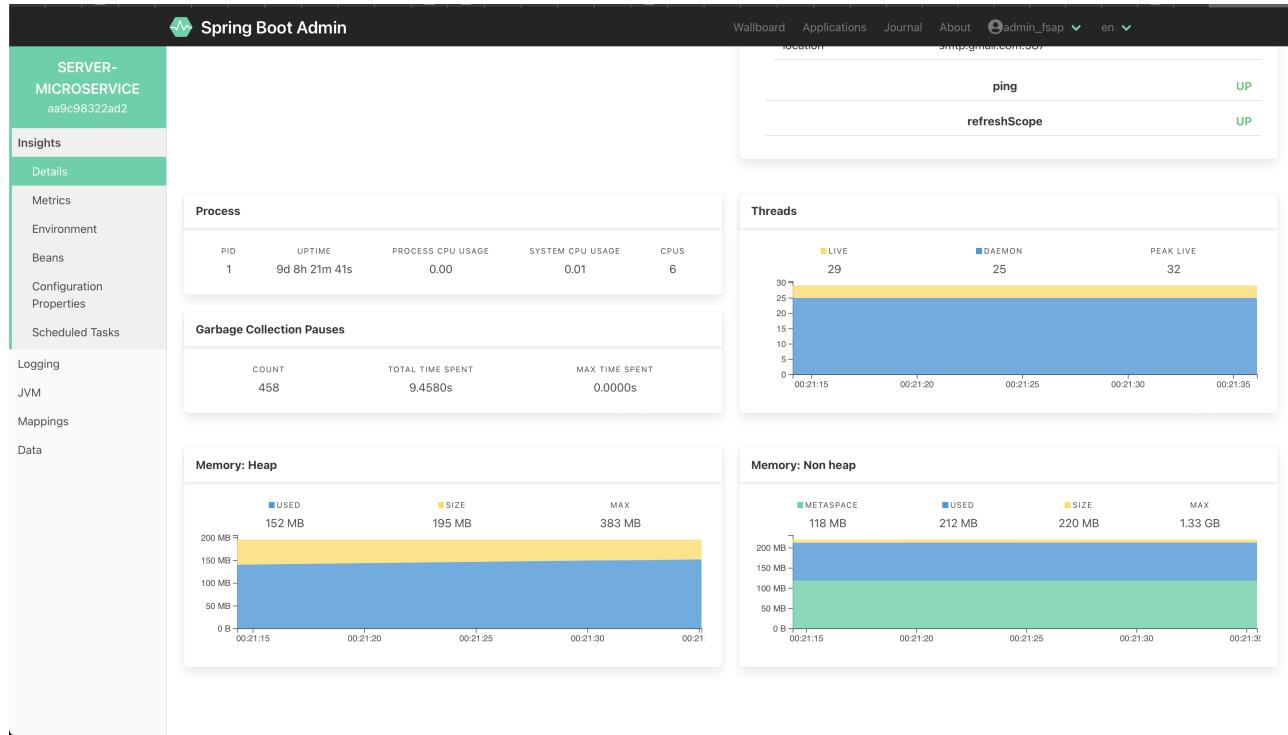


FIGURE 3.6 – Détail d'un microservice

la figure 3.7 illustre le journal d'évènements des microservices

The screenshot shows the Spring Boot Admin interface with the title 'Journal d'évènements'. The table has four columns: 'Application', 'Instances', 'Temps', and 'Evènement'. The data shows frequent status changes for a single instance of 'SERVER-MICROSERVICE' over a period of about two days.

Application	Instances	Temps	Evènement
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 09:03:59.026	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 09:02:06.794	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 08:57:41.604	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 08:55:38.745	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 08:55:38.547	STATUS_CHANGED (OFFLINE)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 08:51:08.765	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 08:49:09.289	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	10/04/2022 08:49:08.584	STATUS_CHANGED (OFFLINE)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 14:30:18.858	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 14:28:04.250	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 14:18:58.757	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 14:14:38.851	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 14:14:38.566	STATUS_CHANGED (OFFLINE)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 13:13:28.875	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 13:11:14.791	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 12:43:08.741	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 12:41:08.781	STATUS_CHANGED (DOWN)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 12:41:08.569	STATUS_CHANGED (OFFLINE)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 11:17:38.921	STATUS_CHANGED (UP)
SERVER-MICROSERVICE	79c53efd61ac	09/27/2022 11:15:45.332	STATUS_CHANGED (DOWN)

FIGURE 3.7 – Journal d'évènements des microservices

Chapitre 4

Méthodologie et approche de développement

Initialement, la durée prévue pour le projet était d'un an après la validation du cahier charges. Le cahier de charges fut modifié durant l'exécution du projet, ce qui entraîna le non-respect du délai initial. Le travail a démarré en novembre 2020 et se poursuit encore. Nous avons livré au fur et à mesure les différents modules du système après les tests de validation.

Un système d'information a été développé par **Kadidiatou DJIBO** et **Aminata GUEYE** pour le périmètre d'une faculté [B5]. Ce système permet entre autres l'inscription des étudiants, la gestion de leurs parcours, leurs notes, etc. Un autre système d'information pour la gestion des ressources humaines de l'USTTB a été développé par **Laya Guindo** et **Yacouba Koné** [B6]. Ceux deux systèmes sont indépendants, mais utilisent pourtant beaucoup de données communes. Leurs périmètres sont limités, plusieurs structures ne peuvent pas l'utiliser communément. D'où la nécessité de faire un autre système qui les engloberait. Le travail débute par la compréhension du contexte du sujet. Ensuite, nous nous sommes inspirés des deux systèmes cités ci-dessus, afin de constituer une base de travail.

Pour le choix de la méthodologie de développement, il faut prendre en considération le périmètre et les intervenants du projet, les contraintes telles que le temps alloué et le nombre de personnes mobilisées et enfin les spécificités de l'architecture. En analysant ces facteurs, nous avons découpé le projet en plusieurs itérations. À chaque itération, une partie du projet est réalisée.

Pour ce faire, nous avons choisi d'adopter la méthode Agile SCRUM comme méthodologie. Agile [B7] est un ensemble de méthodes et de méthodologies qui aident votre équipe à penser plus efficacement, à travailler plus efficacement et à prendre de meilleures décisions. Ces méthodes et méthodologies abordent tous les domaines du génie logiciel traditionnel, y compris la gestion de projet, la conception et l'architecture des logiciels et l'amélioration des processus.

Scrum est un framework qui est utilisé pour implémenter la méthode Agile de développement et de gestion de projet.

La figure 4.1 est une illustration de la méthodologie Agile SCRUM.

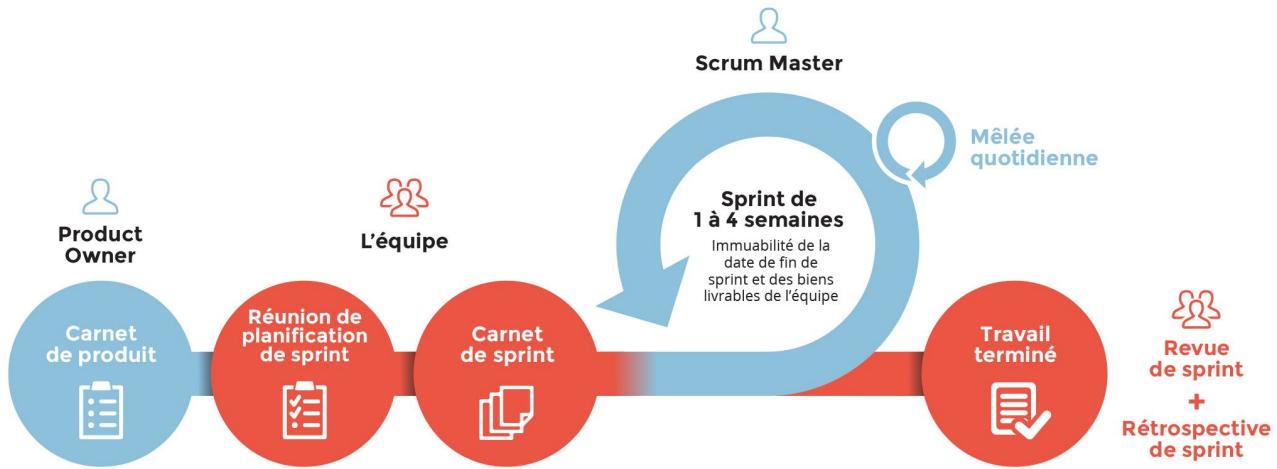


FIGURE 4.1 – Méthodologie agile (Figure tirée de [R1])

la figure 4.2 illustre l’organisation des tâches avec la méthodologie agile.

- **Récits** : L’ensemble des tâches du projet,
- **A faire** : Les tâches qui doivent être réalisées pour l’itération actuelle,
- **En cours** : Les tâches qui sont en cours de réalisations,
- **En test** : Les tâches réalisées, mais pas encore testées,
- **Terminées** : Les tâches réalisées et testées.

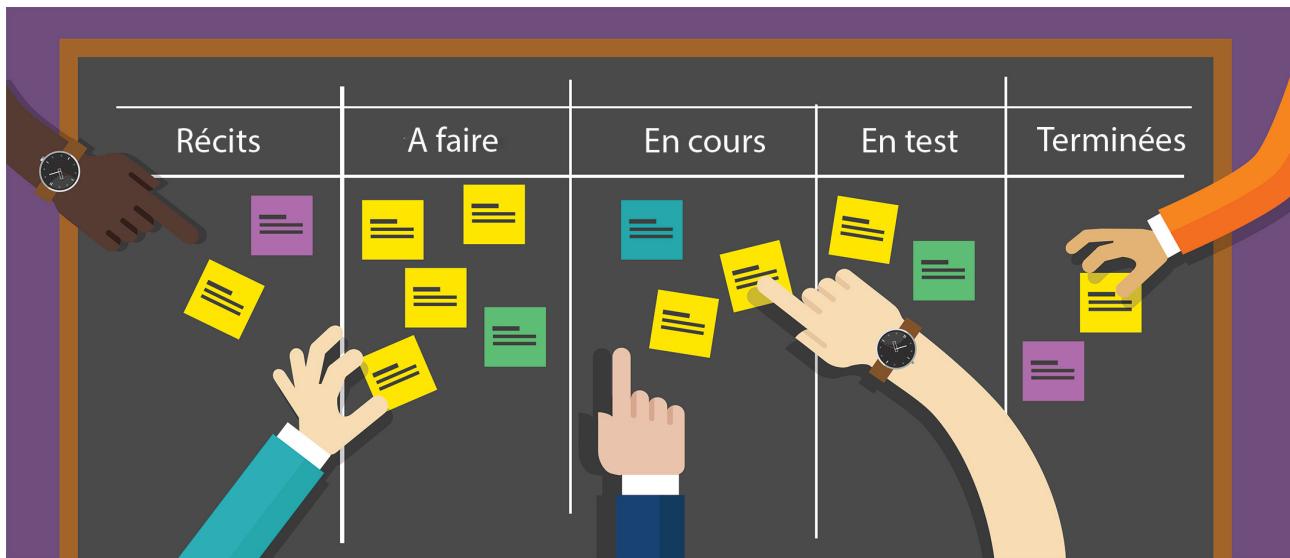


FIGURE 4.2 – Tableau d’organisation des tâches (Figure tirée de [R2])

Nous avons choisi l'approche DevOps.

Le terme DevOps [B8] est un mélange de développement (représentant les développeurs de logiciels, y compris les programmeurs, les testeurs) et d'opération (représentant les experts qui mettent le logiciel en production et gèrent l'infrastructure de production). DevOps [B8] décrit des pratiques qui rationalisent le processus de livraison de logiciels, en mettant l'accent sur l'apprentissage en diffusant les commentaires de la production au développement et en améliorant le temps entre le développement et la livraison.

Nous avons adopté deux pratiques DevOps : **L'intégration et livraison continues (CI/CD)** et **la gestion de version du code**

4.1 La gestion de version du code ou versionning du code

Un logiciel de versioning permet de stocker et de gérer du code, de façon collaborative. Chaque modification apportée au code est enregistrée. Dans le cadre de ce projet, nous avons utilisé Git comme logiciel de versionning. Git propose un système de gestion de branches. Une branche est une nouvelle version du code partant d'un référentiel principal. Ils peuvent être créés et supprimés à la volée, cependant nous avons utilisé trois branches principales :

- develop,
- staging,
- prod

Ces branches sont pérennes. Les détails de chaque branche se trouvent à la **section 4.3**

4.2 Intégration continue et déploiement continu CI/CD.

L'approche Intégration continue (CI), déploiement continu (CD) [R16] introduit l'automatisation au niveau des étapes de développement des applications, cela permet d'augmenter la fréquence de livraison des applications. L'intégration continue, la distribution continue et le déploiement continu sont les principaux concepts liés à l'approche CI/CD. L'approche CI/CD représente une solution aux problèmes posés par l'intégration de nouvelles parties de code pour les équipes de développement et d'exploitation (ce qu'on appelle en anglais « integration hell », ou l'enfer de l'intégration).

Nous avons utilisé Gitlab CI/CD pour l'intégration et la distribution continues. GitLab CI/CD est une fonctionnalité de GitLab permettant de mettre en place des pipelines de CI/CD pour n'importe quel projet, qu'il soit nouveau ou existant, pourvu qu'il utilise Git. Git [R17] est un système de contrôle de version distribué, gratuit et open source conçu pour gérer des projets, avec rapidité et efficacité.

La figure 4.3 illustre le cycle du CI/CD

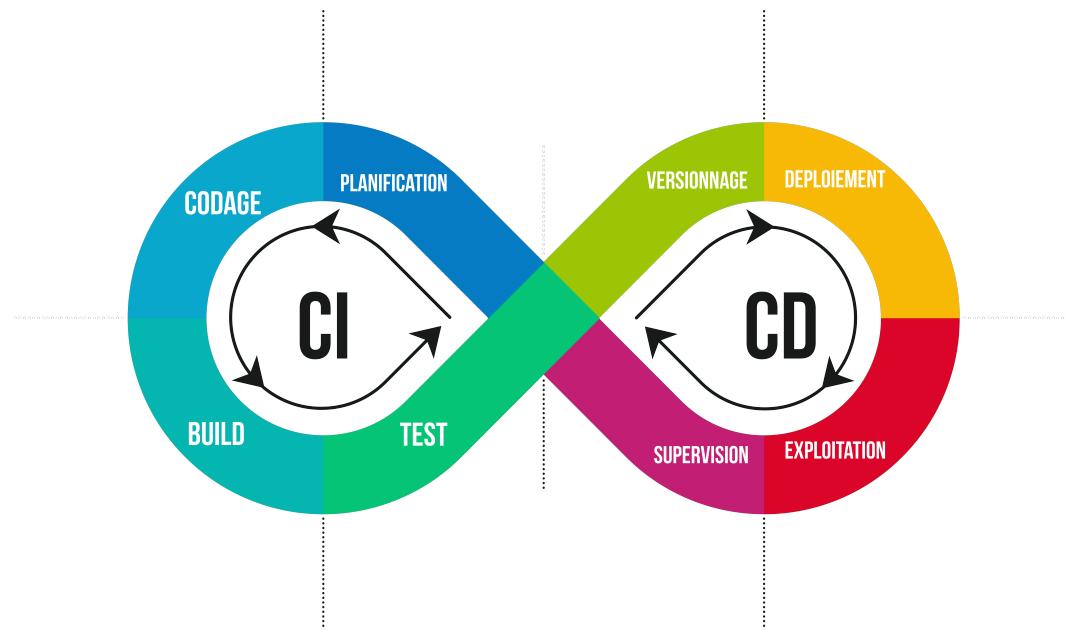


FIGURE 4.3 – Illustration du CI/CD (Figure tirée de [R2])

4.2.1 Intégration continue CI

L'intégration continue (CI) désigne la pratique qui consiste à intégrer automatiquement les changements de code réalisés par les équipes de développement dans un projet. Il s'agit d'une bonne pratique permettant aux développeurs de logiciels de fusionner fréquemment des changements de code dans un dépôt central où les builds et les tests s'exécutent ensuite. Des outils automatisés sont utilisés pour valider le nouveau code avant son intégration. Il s'articule autour d'un système de contrôle de version du code source (Git dans notre projet).

4.2.2 Distribution continue

La distribution ou livraison continue est une étape au-delà de l'intégration continue. La distribution continue automatise le déploiement de l'application. La distribution continue permet de disposer d'une base de code toujours prête à être déployée dans un environnement de production.

Cependant, avec la distribution continue, les déploiements sont déclenchés manuellement. La distribution continue vérifie le code automatiquement, mais elle nécessite une intervention humaine pour déclencher manuellement et stratégiquement le déploiement des modifications. Métrologie Agile Scrum l'oblige, l'ensemble du projet a été découpé en petites tâches. Chaque semaine, certaines de ces tâches sont assignées à l'équipe pour être implémentées.

4.3 Les branches principales

4.3.1 La branche *develop*

La branche ***develop*** est la branche qui sert de base, la branche par défaut. Pour chaque nouvelle tâche, la branche créée est faite à partir de la branche ***develop*** sauf exceptions. Une fois la tâche terminée, la branche de la tâche est renommée en ***develop_nom_branche*** ou ***develop-nom_branche*** et elle est "pushée" sur le dépôt de code (repository). Cette action déclenche le CI.

Il y a d'abord un test unitaire qui s'effectue sur les nouvelles modifications. Si le test échoue, le processus se bloque et le développeur est notifié via Slack que le test a échoué. Si le test passe, les nouvelles modifications sont fusionnées avec la branche ***develop***. S'ensuit alors le build, le test des nouvelles modifications fusionnées avec le code existant. À chaque échec une notification est envoyée sur Slack.

À la fin de ces processus, un trigger manuel est créé pour déclencher la fusion du ***develop*** avec la branche suivante ***staging***

4.3.2 La branche *staging*

Le code sur cette branche est destinée à être déployée sur **serveur de test**. Les commits directs sur cette branche ne sont pas permises. Les nouvelles modifications lui sont ajoutées grâce au trigger manuel qui permet de fusionner la branche ***develop*** et ***staging***. Après la fusion, le nouveau code est "buildé", testé ensuite une image Docker (nous parlons en détail dans la section **section 4.4**) est créée et sauvegardée sur Gitlab. Comme pour la branche ***develop*** si une étape échoue, une notification est envoyée sur Slack.

À la fin de ces trois (3) processus, un trigger manuel est créé pour déclencher le déploiement de la nouvelle image Docker sur le serveur de test. Une fois cet trigger déclencher, le déploiement s'effectue et un nouveau trigger manuel est créé pour fusionner la branche ***staging*** avec la branche ***prod***.

4.3.3 La branche *prod*

Le code sur cette branche est destinée à être déployée sur **serveur de production**. La logique est la même avec la branche de ***staging***. À la fin du processus, un trigger manuel est créé pour déployer l'image Docker sur les différents serveurs de productions.

Cependant, il y a une spécificité avec la branche ***prod***. Il arrive souvent qu'il y ait des bugs sur le serveur de production qui doivent être rapidement corrigées. Suivre le chemin classique ***develop -> staging -> prod*** est assez long. Pour corriger et déploier rapidement une correction, nous créons une branche du nom ***prod_hot_fix***. Après correction et "push" de la branche, ce code est fusionné avec la branche ***prod***. S'ensuit alors les processus de "build", test, création de la nouvelle image Docker et création du trigger manuel pour le déploiement. Le code est aussi fusionné avec la branche ***staging*** et ***develop*** pour garder une bonne cohérence.

Ci-dessous la figure 4.4 qui permet de comprendre notre gestion des branches.

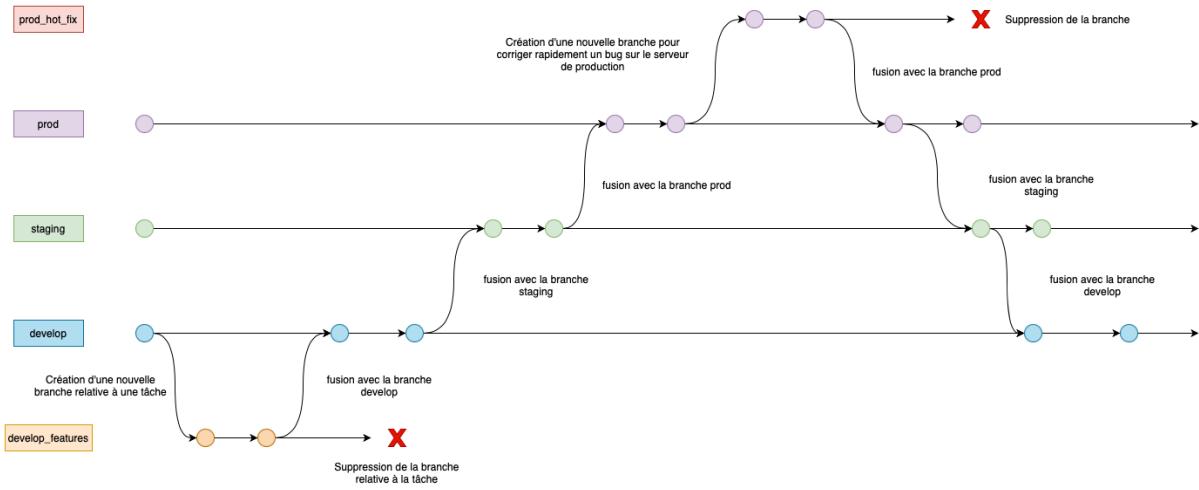


FIGURE 4.4 – Gestion des branches

La figure 4.5 est un exemple de pipeline.

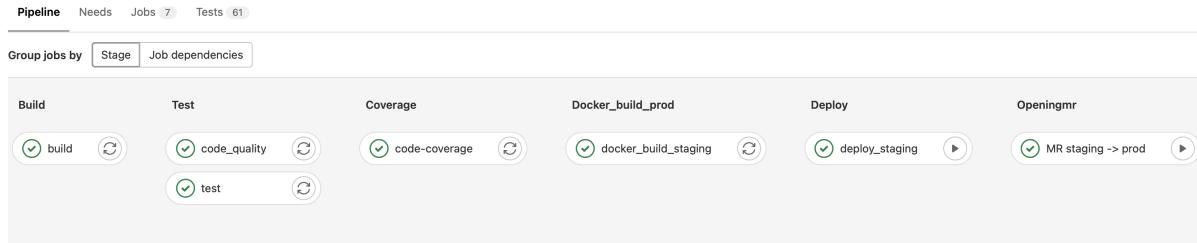


FIGURE 4.5 – Exemple de pipeline

4.4 Docker

Il existe plusieurs façons de déployer une application Spring Boot (Java). La façon la plus commune est de builder (convertir le code source en application exécutable) l'application en **Web Application Archive (WAR)** et le déployer sur un serveur d'application web java comme **Tomcat**, **Wildfly**, etc. La seconde façon la plus commune est de builder l'application en **Java ARchive (JAR)** et l'exécuter avec le **Java Runtime Environment (JRE)**.

Dans ces deux cas, il faut au préalable faire des installations et des configurations sur la machine sur laquelle l'application est déployée. Ces configurations prennent du temps et peuvent être différentes d'un système d'exploitation à un autre.

Pour le déploiement, nous avons opté pour l'utilisation de Docker. Docker [B9] est un programme de ligne de commande, un démon d'arrière-plan et un ensemble de services distants qui adoptent une approche logistique pour résoudre les problèmes logiciels courants et simplifier l'expérience d'installation, d'exécution, de publication et de suppression de logiciels. Pour ce faire, il utilise une technologie UNIX appelée **conteneurs**. Les conteneurs peuvent s'apparenter à de la virtualisation, en effet, ils tournent sur des environnements complètement isolés. La différence avec la virtualisation est que les conteneurs n'utilisent pas de ressources matérielles, ils s'interfacent directement avec le noyau de l'hôte. Cette architecture permet aux conteneurs d'être moins gourmands en ressources.

L'utilisation de Docker évite de faire des configurations sur les machines hôtes, ce qui représente un gain de temps énorme. Nos microservices sont déployés sur des conteneurs et fonctionnent de manière totalement isolée.

La figure 4.6 est un Comparatif entre l'architecture docker et machine virtualisation.

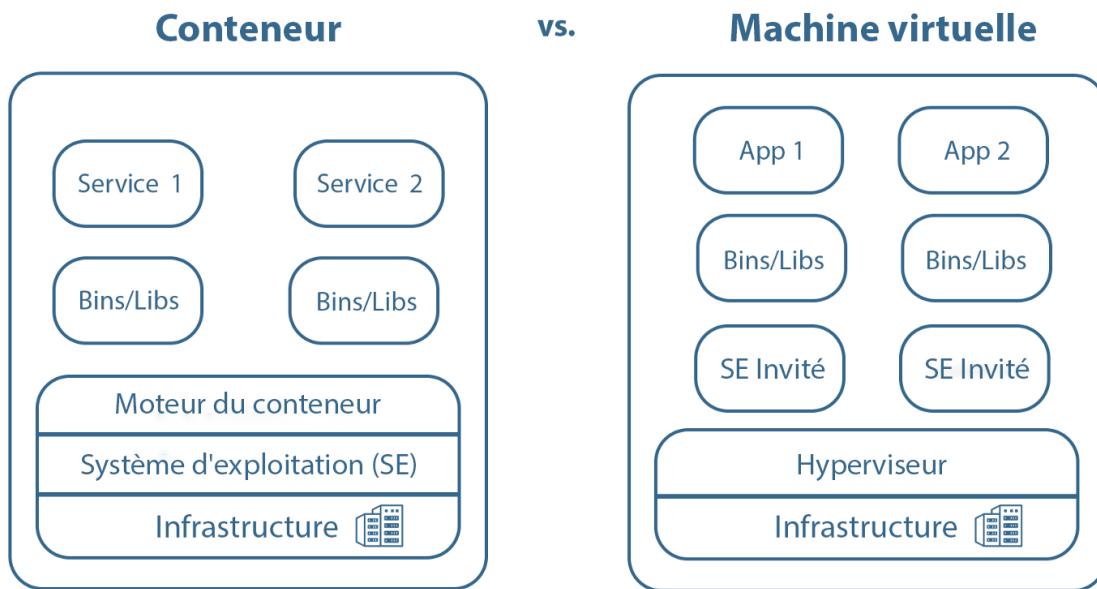


FIGURE 4.6 – Comparatif entre l'architecture docker et machine virtualisation (Figure tirée de [R3])

Chapitre 5

Tests fonctionnels

« Souvent sous-évalué et jugé peu digne d'intérêt par les développeurs, le test n'en est pas moins un maillon crucial de l'ingénierie logicielle et un pilier fondamental sans lequel aucun logiciel ne peut être mis sur le marché avec la qualité nécessaire. » [R18].

Le test a pour but principal d'identifier des comportements problématiques (bugs) du logiciel. Ces comportements peuvent être des erreurs de logique, des erreurs fonctionnelles, des dégradations de certaines fonctionnalités suite à l'ajout de nouvelles fonctionnalités ou correction de bugs, etc.

Il existe de nombreux types de tests de logiciels : tests d'acceptation, tests d'intégration, tests unitaires, tests fonctionnels, tests de performance, tests de régression, tests de charge, tests d'utilisation, tests de configuration, etc [B10].

Dans le cadre du projet, nous avons réalisé des tests unitaires et d'intégration.

Un test unitaire [R18] permet de valider qu'une unité logicielle fonctionne comme prévu. Une unité est un extrait de code d'une application testable. Cela nous permet de nous assurer que ces extraits de code continueront à fonctionner modifications.

Tests d'intégration [R18] : permet de s'assurer que l'ensemble des composants logiciels fonctionne bien ensemble. Les tests d'intégration font parties intégrantes de l'approche CI/CD.

Nous avons utilisé **Spring Boot Starter Test** et **Junit** pour les tests unitaires et d'intégrations. Les tests unitaires et d'intégrations sont implémentés dans le code source dans le répertoire **src/test**. En général, les tests unitaires et d'intégrations sont faits en utilisant une base de données **H2**. La base de données est temporaire, elle est créée au début du test et effacé à la fin. Les bases de données H2 sont très légères, rapide et on peut les persister dans un fichier si nécessaire. Cela est pratique, mais présente un gros inconvénient. Les bases de données supportent une version allégée du **SQL (Structured Query Language)**, il est assez différent de PostgreSQL. Pour cette raison, nous avons décidé d'utiliser une base PostgreSQL lors des tests pour avoir une uniformité.

Les tests sont effectués lors du processus de CI sur un conteneur Docker créé pour simuler l'environnement de production de l'application. Les tests peuvent être également exécutés localement à travers IntelliJ.

Nous avons utilisé le plugin **JaCoCo** pour avoir la couverture du test. Ce plugin est in-

tégré et projet et génère un fichier **HTML (HyperText Markup Language)** à la fin des tests indiquant le pourcentage de code testé. La figure 5.1 présente l'interface web de la couverture des tests de la ressource structure.

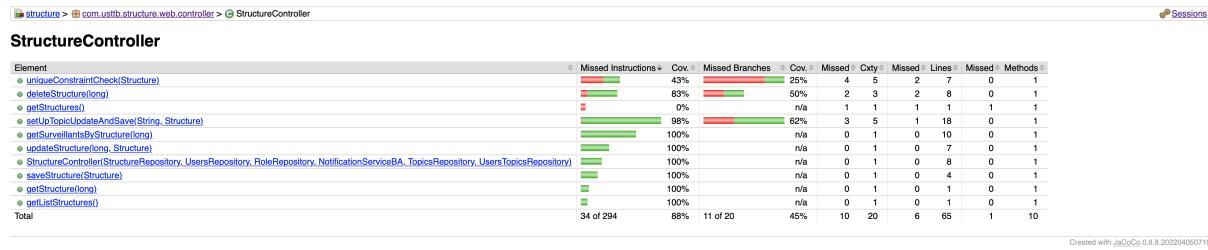


FIGURE 5.1 – Couverture de test de la ressource structure

Exemple de test unitaire : cas test des *getters*

Spring Boot utilise la programmation orientée objet, dans ce paradigme les attributs de la classe sont privés et on peut y accéder grâce aux *getters* et les modifier grâce aux *setters*. La figure 5.2 présente le code du test unitaire des *getters*.

```

1 usage  nnnamakan10 *
208 @
209     public static  <T extends ModelBase> void executeAllGetters(T bean){
210         Class<?> currentClass = bean.getClass();
211
212         findGetters(currentClass).forEach(
213             method -> {
214                 try {
215                     method.invoke(bean);
216                 } catch (IllegalAccessException | InvocationTargetException e2){
217                 }
218             }
219         );
220     }
221

```

FIGURE 5.2 – Code du test unitaire des getters d'une classe

CONCLUSION ET PERSPECTIVES

En définitive, l'objectif du projet traité dans ce document consistait en la conception et la mise en œuvre d'une application pour la gestion des universités : USTTB, USSGB et USJPB(FSAP). Pour cela, nous avons réalisé une application interactive basée sur les microservices permettant de gérer les différents traitements du système et de satisfaire les besoins des différents utilisateurs impliqués dans les différents processus.

La Faculté de Sciences Économique et de Gestion (FSEG) fut la première structure qui a commencé l'utilisation de la plateforme en mars 2021. Progressivement, la Faculté des Sciences et Techniques (FST), la Faculté d'Histoire et de Géographie (FHG) et la Faculté des Sciences Administratives et Politiques (FSAP) ont adopté et utilisé la plate-forme. Ainsi, nous avons régulièrement recueilli les retours des utilisateurs (feedbacks) pour mieux adapter le système à leurs besoins et leur faciliter son utilisation. De ce fait, de nouveaux besoins ont surgi pendant l'utilisation, ce qui a entraîné la modification du cahier des charges initial, ce qui justifie l'option de la gestion agile de notre projet.

Le système a beaucoup facilité la gestion des inscriptions, des notes, des ressources humaines... Le système a dématérialisé beaucoup de processus comme la demande de transfert des étudiants d'une option à une autre.

Les microservices ont été consommés par une application web et mobile. Les utilisateurs interagissent avec ces applications.

À l'état actuel, la très grande majorité des tâches du cahier de charge a été réalisé. Le projet est en phase d'achèvement.

Ce travail fut une grande expérience pour nous. Il nous a permis d'approfondir nos connaissances acquises pendant le Master Informatique, comme la conception de la base de données, la programmation, etc. Il nous a permis d'apprendre le CI/CD, l'utilisation de l'architecture microservice, Docker, etc. Il nous appris surtout à mieux optimiser une application pour que le temps de réponse soit le plus optimal possible et la collaboration avec d'autres équipes de développeurs.

Avec le système actuel, les universités ne sont pas interconnectées. En perspective, nous envisageons :

- d'étendre le projet au niveau de tout l'enseignement supérieur et d'interconnecté toutes les différentes structures,
- d'améliorer de certaines fonctionnalités déjà existantes,
- de perfectionner notre architecture microservice en séparant la base de données en

- plusieurs bases de données, en utilisant plusieurs instances par microservice,
- d'orchestrer les conteneurs Docker avec Kubernetes,
- d'automatiser des tests d'intégration entre les microservices

Bibliographie

- [B1] BAUMANN Henriette & BAUMANN Philippe GRÄSSE, Patrick. *UML 2.0 in Action : A Project-based Tutorial.* Packt Publishing Ltd, 32 Lincoln Road, Olton, Birmingham, B27 6PA, UK, 2005.
- [B2] Howard Podeswa. *UML FOR THE IT BUSINESS ANALYST.* Course Technology, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2014.
- [B3] Barbara Haley Wixom & David Tegarden Alan Dennis. *SYSTEMS ANALYSIS & DESIGN.* Wiley, 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2014.
- [B4] Thomas Hunter II. *Advanced Microservices.* Apress, 233 Spring Street New York City United States NY 10013, 2012.
- [B5] Kadidiatou DJIBO & Aminata GUEYE. *Conception et mise en oeuvre d'un système d'information pour la Faculté des Sciences et Techniques (FST).* Mémoire de Master - 2017.
- [B6] Laya Yibé GUINDO & Yacouba Koné. *Conception et développement d'une application de gestion des ressources humaines et du courrier pour l'USTTB.* Mémoire de Master - 2020.
- [B7] Andrew Stellman & Jennifer Greene. *Learning Agile.* O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2010.
- [B8] Michael Hüttermann. *DevOps for Developers.* Apress, 233 Spring Street New York City United States NY 10013, 2012.
- [B9] Jeff Nickoloff. *Docker in Action.* Manning Publications Co, 20 Baldwin Road PO Box 761 Shelter Island, NY 11964, 2016.
- [B10] Jacqueline KONATE. *Génie Logiciel.* Unité d'enseignement, Master - 2020.

Références

- [R1] Bubble plan. <https://bubbleplan.net>. [dernier accès octobre 2022].
- [R2] Abode stock. <https://stock.adobe.com>. [dernier accès octobre 2022].
- [R3] Dev genius. <https://devgenius.io>. [dernier accès octobre 2022].
- [R4] Wikipedia contributors. Analyse des besoins. https://stringfixer.com/fr/Requirements_analysis, 2022. [dernier accès juillet 2022].
- [R5] Staruml. <https://staruml.io>. [dernier accès août 2022].
- [R6] Java. <https://java.com>. [dernier accès octobre 2022].
- [R7] Springboot. <https://spring.io/projects/spring-boot>. [dernier accès août 2022].
- [R8] Docker overview. <https://docs.docker.com>. [dernier accès août 2022].
- [R9] Postgresql. <https://www.postgresql.org>. [dernier accès août 2022].
- [R10] Postman. <https://postman.com>. [dernier accès août 2022].
- [R11] Gitlab. <https://gitlab.com>. [dernier accès août 2022].
- [R12] Jetbrains. <https://www.jetbrains.com>. [dernier accès août 2022].
- [R13] Eureka 2.0 motivations. <https://github.com/Netflix/eureka/wiki/Eureka-2.0-Motivations>. [dernier accès août 2022].
- [R14] Spring cloud gateway. <https://spring.io/projects/spring-cloud-gateway>. [dernier accès août 2022].
- [R15] Spring boot admin reference guide. <https://codecentric.github.io/spring-boot-admin/2.5.1>. [dernier accès août 2022].
- [R16] Qu'est-ce que l'approche ci/cd ? <https://www.redhat.com/fr/topics/devops/what-is-ci-cd>. [dernier accès août 2022].
- [R17] Git. <https://git-scm.com>. [dernier accès août 2022].
- [R18] Qu'est-ce que le test logiciel ? <https://ibm.com/ca-fr/topics/software-testing>. [dernier accès octobre 2022].

ANNEXES

Annexe 1 : Interface Swagger-UI

La figure 5.3 présente l'interface **Swagger-UI**. Cette interface est la documentation des microservices. Elle est utilisée par les développeurs web et mobile pour consommer les fonctionnalités des microservices.

The screenshot shows the Swagger-UI interface for the USSGB RESTFull API. At the top, there is a green header bar with the Swagger logo and a dropdown menu labeled "Select a spec" set to "default". Below the header, the title "USSGB RESTFull API" is displayed, along with the base URL "[Base URL: 194.163.155.205:9098 /]" and the API documentation URL "http://194.163.155.205:9098/v2/api-docs". Below this, there is a brief description of the API for developers, links to "Terms of service" and "Contact", and a CC BY-SA 3.0 license notice. On the right side of the main content area, there is a "Authorize" button with a lock icon. The main content area is divided into sections: "Authentification" (Auth Controller), "Files" (Files Controller), "Init" (Init Controller), and "Partage Controller" (Partage Controller). Each section contains a list of API endpoints with their methods, URLs, and descriptions. For example, under "Files", there are five endpoints: POST /files (Upload file and save it), GET /files/{fileType}/{filename} (getFile), POST /files/inscription (postInscriptionFile), POST /files/simpleUpload/{fileType} (Upload file without in serveur without save it in fichier table), and PUT /files/updateUsersProfilePics/{id_user} (Update user profile photo). The "PUT" endpoint has a lock icon next to it, indicating it requires authentication.

FIGURE 5.3 – Interface Swagger-UI

Annexe 2 : Interface de postman

La figure 5.4 présente l'interface de **Postman**. Postman permet de faire des requêtes vers les API Rest exposés par les microservices.

The screenshot shows the Postman application interface. On the left, there's a sidebar with various project and collection options. The main area shows a collection named "USSOB RESTFull" containing several requests. One request is selected, labeled "GET getStructures". The "Params" tab is active, showing a single parameter "Key": "Value". Below the request details, the "Body" tab is selected, displaying a JSON response. The response body is as follows:

```
1 {
2   "id": 1,
3   "nom": "Facult\u00e9 des Sciences et des Techniques",
4   "type": "FACULTE",
5   "sigle": "FST",
6   "description": "Facult\u00e9 des Sciences et des Techniques\\n",
7   "detail": null,
8   "adresse": "Colline de Badalabougou Rue L\u00e9opold Sedar Senghor",
9   "email": null,
10  "telephone": null,
11  "logo": "163.97.143.133/usstb/assets/uploads/logo/161875775446fst_logo.png"
12 },
13 {
14   "id": 3,
15   "nom": "FSAP",
16   "type": "FACULTE",
17   "sigle": "FSAP",
18   "description": "axerty",
19   "detail": "axerty",
20   "adresse": "Sogeniko, Bamako Mali.",
21   "email": null,
22   "telephone": null,
23   "logo": "173.212.252.70/fsap/assets/uploads/logo/164312031511123640382_176695824086567_3782844324490680546_n-2.jpg"
24 },
25 }
```

FIGURE 5.4 – Interface de Postman

Annexe 3 : Interface web - Liste des personnels administratifs

La figure 5.5 présente la liste des personnels de l'interface web.

The screenshot shows the FST web application interface. The left sidebar contains navigation links such as Historiques, Réinitialisation de passe, Statistiques, Liste de passage, Gestion générale, Filières/Options et UEs, Comptabilité, G.R.H., Personnel administratif (which is selected), Vacataire, Enseignants, AFFECTATIONS (with sub-links: Affection des enseignants, Affection des personnels, Affection aux DER), Scolarité, and Enseignant. The main content area is titled "LISTE DU PERSONNEL ADMINISTRATIF" and features a "Filtre" section with a search input. Below is a table with columns: Nom, Prenom, Téléphone, Spécialité/Fonction, and Statut. The table lists 12 administrative personnel entries, each with an "Action" button. The footer of the page displays the year "2020-2021 USSTB".

	Nom	Prenom	Téléphone	Spécialité/Fonction	Statut	Action
<input type="checkbox"/>	Boré	Aminata	90776017	Secrétaire	Fonctionnaire civil	Action
<input type="checkbox"/>	BOUARE	Nourou-Dine Mohamed	76022747	Informatique	Fonctionnaire civil	Action
<input type="checkbox"/>	Coulibaly	Awa	77020737	Secrétaire	Fonctionnaire civil	Action
<input type="checkbox"/>	COULIBALY	Nouhoum	69371182	Scolarité	Contractuel	Action
<input type="checkbox"/>	Dembélé	Kalifa	77412150	Informatique	Contractuel	Action
<input type="checkbox"/>	DIAKITE	Aminata S	76461568	Secrétaire	Fonctionnaire civil	Action
<input type="checkbox"/>	DIAKITE	Mahadi	76016972	Secrétaire	Fonctionnaire civil	Action
<input type="checkbox"/>	Diallo	Maïmouna	66997931	Secrétaire	Fonctionnaire civil	Action
<input type="checkbox"/>	Haidara	Mohamed	76114476	Comptable	Vacataire	Action
<input type="checkbox"/>	KABAYO	Sekou	76132565	Comptabilité	Fonctionnaire civil	Action
<input type="checkbox"/>	Konaré	Astan Moussokoro	76420230	Secrétaire	Fonctionnaire civil	Action
<input type="checkbox"/>	Sacko	Alliou	79321177	Informatique	Fonctionnaire civil	Action
<input type="checkbox"/>	Sissoko	Madina Kahou	65622324	Secrétaire	Fonctionnaire civil	Action

FIGURE 5.5 – Interface web - Liste des personnels administratifs

Annexe 4 : Interface web - Génération de la carte étudiante et l'attestation d'inscription

La figure 5.6 présente la génération de la carte étudiante et l'attestation d'inscription depuis l'interface web.

The screenshot displays the FST web application interface. On the left, a sidebar lists various administrative functions: Scolarité, Enseignant, Evaluation et notation, Secrétariat, Candidature à traiter, Candidatures approuvées, Candidatures rejetées, Admis/Non admis, Validation des inscriptions, Cahier de texte, Congé & permission, Gestion courriers, Plan De Carrière, Gestion des grades, and Bibliothèque. The main content area shows a student profile for the academic year 2022-2023. The profile details include:

- N° étudiant: FST2021MA000073
- Nom: BAGAYOKO
- Prénom: Ibrahim
- Né(e) le: 18-12-2001
- A: Bamako
- Nationalité: Malienne
- Soumazard: Kalaban Coro Koulobieni
- Téléphone: 78884652
- Parcours: MPCI - LI
- Fait le: 03-10-2022

Below the profile is a QR code. To the right of the profile is a note: "Cette carte ne peut être utilisée que par le signataire autorisé. En cas de découverte, veuillez déposer cette carte à l'(e) FST. En cas de perte, l'étudiant(e) est tenu d'informer le service scolaire et orientation du rectorat en fournissant un certificat de perte". Below this note are two circular seals: "Université des Sciences, des Techniques et des Technologies de Bamako" and "FACULTÉ DES SCIENCES ET DES TECHNIQUES". The text "Colline de Badalabougou Rue Léopold Sedar Senghor" appears below the seals. At the bottom of the profile section is the heading "Attestation d'inscription". A statement reads: "Je soussigné(e), Secrétaire Principal(e) de l'(e) Faculté des Sciences et des Techniques (FST), atteste que: l'étudiant ayant les informations ci-dessous, est inscrit dans notre FACULTÉ. Détails de l'étudiant". Below this is a table with two rows: "Année universitaire" and "2022 - 2023". On the far right, there is a sidebar titled "TROMBINOSCOPE" with a dropdown menu for "Niveau" set to "L1". Below this is a table with columns "Niveau", "Option", "Scolarité", and "Action". Several rows are listed, each with a "Actions" button. The first row is for "Niveau: LESE, Option: 2, Action: Actions". Subsequent rows follow a similar pattern with different codes like LBM-SVT, LRT, LOG, LRL, LM-MPCI, LRME, LSB-SVT, LRME, and LQL.

FIGURE 5.6 – Interface web - Génération de la carte étudiant et l'attestation d'inscription

Annexe 5 : Interface web - Statistiques d'inscription par genre

La figure 5.6 présente l'interface web des statistiques d'inscription par genre

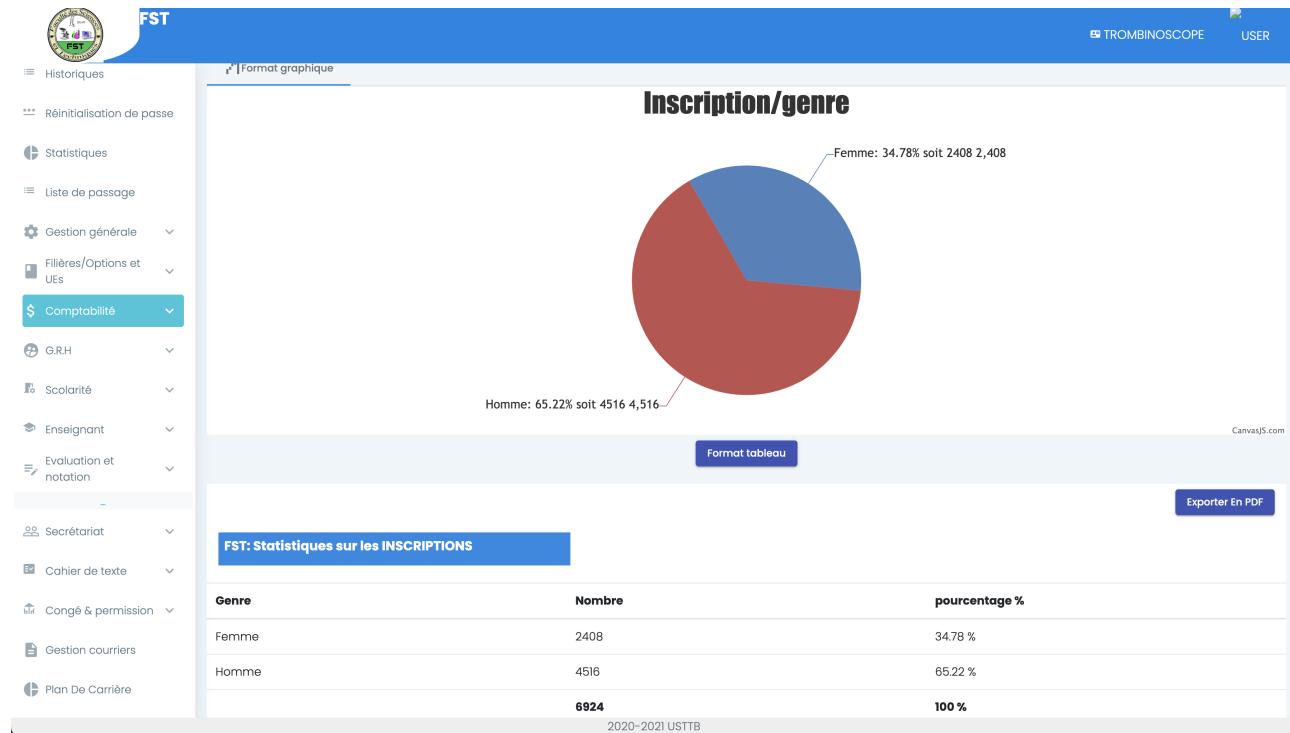


FIGURE 5.7 – Interface web - Statistiques d'inscription par genre

Annexe 6 : Test de la fonctionnalité authentification

Cette fonctionnalité permet aux utilisateurs du système de s'authentifier. La grande majorité des requêtes de l'API est sécurisée. Un jeton est requis pour accéder aux requêtes sécurisées. Ce jeton est retourné après l'authentification de l'utilisateur. Il est crypté et contient certaines informations de l'utilisateur, ces droits d'accès ainsi que la durée de validité du jeton.

L'utilisateur doit fournir son identifiant ainsi que son mot de passe pour s'authentifier. Si l'une de ces deux informations est incorrecte, l'authentification échoue. L'authentification peut échouer avec un identifiant et mot de passe correct si le compte de l'utilisateur est désactivé. Le bon fonctionnement de cette fonctionnalité est primordiale pour le système. La figure 5.8 présente le code du test de la fonctionnalité authentification. Nous avons testé tous les scénarios possibles. Dans chaque scénario, une requête est envoyée et juste après, nous vérifions le statut de la réponse en faisant des affirmations. Si le statut de la réponse est **400** cela veut dire que l'authentification n'a pas marché. Un statut **200** indique que l'authentification a réussi. Le test de la fonctionnalité est validé si toutes les affirmations sont vérifiées.

```

38     @Test
39     public void authenticate(@Autowired DataSource dataSource) throws Exception {
40         try (Connection conn = dataSource.getConnection()) {
41             ScriptUtils.executeSqlScript(conn, new ClassPathResource("cleanDB.sql"));
42         }
43
44         Users user = ModelGenerators.getUser();
45         user.setUsername("test_login");
46
47         //Test utilisateur non enregistré
48         AuthRequest credentials = new AuthRequest(user.getUsername(), password: "test");
49         MvcResult result = authRequest(credentials);
50         assertEquals(result.getResponse().getStatus(), actual: 400);
51
52         //Test Utilisateur enregistré avec un mot de passe incorrect
53         usersRepository.save(user);
54         result = authRequest(credentials);
55         assertEquals(result.getResponse().getStatus(), actual: 400);
56
57         //Test Utilisateur enregistré avec un nom d'utilisateur incorrect
58         credentials = new AuthRequest(username: "test", ModelGenerators.PASSWORD_TEST);
59         result = authRequest(credentials);
60         assertEquals(result.getResponse().getStatus(), actual: 400);
61
62         //Test Utilisateur enregistré avec compte désactivé
63         user.setStatut(false);
64         usersRepository.save(user);
65         credentials = new AuthRequest(user.getUsername(), ModelGenerators.PASSWORD_TEST);
66         result = authRequest(credentials);
67         assertEquals(result.getResponse().getStatus(), actual: 400);
68
69         //Test Utilisateur enregistré avec les bons accès et le compte activé
70         user.setStatut(true);
71         usersRepository.save(user);
72         result = authRequest(credentials);
73         assertEquals(result.getResponse().getStatus(), actual: 200);
74     }

```

FIGURE 5.8 – Code du test de la fonctionnalité authentification

- **Ligne 40-42** : cette partie du code permet de vider la base de données de test ;
- **Ligne 44-45** : Un utilisateur est généré aléatoirement mais n'est pas enregistré dans la base ;
- **Ligne 48-50** : une requête est envoyée pour essayer d'authentifier l'utilisateur qui a été généré précédemment. À la dernière ligne de ce bout de code nous affirmons que la réponse a un statut **400** ;
- **Ligne 53-55** : l'utilisateur généré est sauvegardé dans la base et une requête est envoyée pour essayer de l'authentifier. Un bon identifiant et un mauvais mot de passe sont utilisés comme paramètres de la requête. À la dernière ligne de ce bout de code nous affirmons que la réponse a un statut **400** ;
- **Ligne 58-60** : une requête est envoyée pour essayer d'authentifier le même utilisateur. Un mauvais identifiant et un bon mot de passe sont utilisés comme paramètres

de la requête. À la dernière ligne de ce bout de code nous affirmons que la réponse a un statut **400** ;

- **Ligne 63-67** : le compte de l'utilisateur est désactivé. une requête est envoyée pour essayer d'authentifier utilisateur. Un bon identifiant et un bon mot de passe sont utilisés comme paramètres de la requête. À la dernière ligne de ce bout de code nous affirmons que la réponse a un statut **400** ;
- **Ligne 70-73** : le compte de l'utilisateur est activé. Une requête est envoyée pour essayer d'authentifier utilisateur. Un bon identifiant et un bon mot de passe sont utilisés comme paramètres de la requête. À la dernière ligne, de ce bout de code, nous affirmons que la réponse a un statut **200** ;

Annexe 7 : Les fonctionnalités testées

Le tableau 5.1 indique l'ensemble des fonctionnalités testées.

Ressources principales	Fonctionnalités
Courrier et Imputation	<ul style="list-style-type: none">— Création d'un courrier— Mise à jour d'un courrier— Recherche d'un courrier en fonction de certains paramètres— Suppression d'un courrier— Répondre un courrier— Imputation d'un courrier— Mise à jour d'une imputation
Éditeur	<ul style="list-style-type: none">— Création d'un éditeur— Mise à jour d'un éditeur— Recherche d'un éditeur en fonction de certains paramètres— Suppression d'un éditeur
Ouvrage	<ul style="list-style-type: none">— Création d'un ouvrage— Mise à jour d'un ouvrage— Recherche d'un ouvrage en fonction de certains paramètres— Suppression d'un ouvrage
Demande étudiant	<ul style="list-style-type: none">— Création d'une demande étudiant— Mise à jour d'une demande étudiant— Recherche d'une demande étudiant en fonction de certains paramètres— Suppression d'une demande étudiant

Filière	<ul style="list-style-type: none"> — Création d'une filière — Mise à jour d'une filière — Recherche d'une filière en fonction de certains paramètres — Suppression d'une filière — Affectation d'une spécialité du bac à une filière
Témoignage	<ul style="list-style-type: none"> — Création d'un témoignage — Mise à jour d'un témoignage — Recherche d'un témoignage en fonction de certains paramètres — Suppression d'un témoignage
Candidat	<ul style="list-style-type: none"> — Création d'un candidat — Mise à jour d'un candidat — Recherche d'un candidat en fonction de certains paramètres — Suppression d'un candidat — Vérification d'un candidat avant sa préinscription ou candidature
Candidature	<ul style="list-style-type: none"> — Création d'une candidature — Mise à jour d'une candidature — Validation d'une candidature — Acceptation d'une candidature — Recherche d'une candidature en fonction de certains paramètres — Suppression d'une candidature

Paiement	<ul style="list-style-type: none"> — Création d'un paiement — Mise à jour d'un paiement — Recherche d'un paiement en fonction de certains paramètres — Suppression/Annulation d'un paiement
Évaluation	<ul style="list-style-type: none"> — Création d'une évaluation — Mise à jour d'une évaluation — Recherche d'une évaluation en fonction de certains paramètres — Suppression d'une évaluation — Récupérations des salles disponibles pour une évaluation selon son créneau horaire — Récupération de la liste de présence d'une évaluation donnée — Affectation de surveillant à une évaluation — Récupération des surveillants affectées à une évaluation — Récupération de la liste des étudiants concernés par une évaluation — Récupération de la liste des étudiants noté, et non noté — Récupération des groupes concernés par une évaluation
Note	<ul style="list-style-type: none"> — Création d'une note — Mise à jour d'une note — Recherche d'une note en fonction de certains paramètres — Suppression d'une note

Note générale	<ul style="list-style-type: none"> — Création d'une note générale — Calcul d'une note générale — Mise à jour d'une note générale — Recherche d'une note générale en fonction de certains paramètres — Suppression d'une note générale — Attribution de point de jury à une note générale
Unité d'enseignement	<ul style="list-style-type: none"> — Création d'une unité d'enseignement — Mise à jour d'une unité d'enseignement — Recherche d'une unité d'enseignement en fonction de certains paramètres — Suppression d'une unité d'enseignement — Affectation d'une unité d'enseignement à des enseignants — Affectation d'une unité d'enseignement à des options de parcours étudiant
Congé/Permission	<ul style="list-style-type: none"> — Création d'une demande de congé/permission — Mise à jour d'une demande de congé/permission — Recherche d'une demande de congé/permission en fonction de certains paramètres — Suppression d'une demande de congé/permission — Traitement d'une demande de congé/permission
Personnel	<ul style="list-style-type: none"> — Création d'un personnel — Mise à jour d'un personnel — Recherche d'un personnel en fonction de certains paramètres — Suppression d'un personnel

Authentification	<ul style="list-style-type: none"> — Authentification d'utilisateur
Rôle	<ul style="list-style-type: none"> — Création d'un rôle — Mise à jour d'un rôle — Recherche d'un rôle en fonction de certains paramètres — Suppression d'un rôle
Année	<ul style="list-style-type: none"> — Création d'une année — Mise à jour d'une année — Recherche d'une année en fonction de certains paramètres — Suppression d'une année
Bâtiment	<ul style="list-style-type: none"> — Création d'un bâtiment — Mise à jour d'un bâtiment — Recherche d'un bâtiment en fonction de certains paramètres — Suppression d'un bâtiment
Condition de passage	<ul style="list-style-type: none"> — Création d'une condition de passage — Mise à jour d'une condition de passage — Recherche d'une condition de passage en fonction de certains paramètres — Suppression d'une condition de passage

Équipement	<ul style="list-style-type: none"> — Création d'un équipement — Mise à jour d'un équipement — Recherche d'un équipement en fonction de certains paramètres — Suppression d'un équipement
Salle	<ul style="list-style-type: none"> — Création d'une salle — Mise à jour d'une salle — Recherche d'une salle en fonction de certains paramètres — Suppression d'une salle
Structure	<ul style="list-style-type: none"> — Création d'une structure — Mise à jour d'une structure — Recherche d'une structure en fonction de certains paramètres — Suppression d'une structure

TABLE 5.1: Fonctionnalités testées