
UFR SCIENCES ET TECHNOLOGIES DEPARTEMENT

D'INFORMATIQUE

# Mémoire de fin d'études

Pour l'obtention du diplôme de master

Mention Informatique ; Spécialité Génie Logiciel

Sujet :

## Architecture micro-services pour la dématérialisation des procédures et documents administratifs : cas de l'Université Assane SECK de Ziguinchor

Présenté et soutenu publiquement le 09/05/2022 par :

M. Ibrahima MARE

**Membres du jury**

- Pr. Youssou FAYE (**Président**)
- Dr. Ibrahima DIOP (**Encadrant**)
- Pr. Youssou DIENG (**Co-encadrant**)
- M. Bassirou DIENE (**Rapporteur**)
- Dr. Papa Alioune CISSE (**Examinateur**)

**Sous la direction de :**

- Dr. Ibrahima DIOP
- Pr. Youssou DIENG

**Sous la supervision de :**

- Pr. Youssou FAYE

***Année universitaire : 2021-2022***

# RESUME

Depuis sa création, l'université Assane Seck de Ziguinchor ne cesse d'évoluer dans plusieurs domaines. Cette évolution progressive peut rendre complexe sa gestion dans l'avenir. Raisons pour lesquelles les autorités ont la volonté d'anticiper sur la complexité de la gestion de l'université. Ainsi pour bien mener à la bonne gestion de l'institut, l'usage de systèmes informatiques est indispensable. C'est pourquoi on veut passer à la dématérialisation de toutes les procédures et documents administratifs de l'université.

Ces enjeux qui accompagnent cette dématérialisation sont compris par les autorités administratives de l'université. En effet, il existe des logiciels utilisables au quotidien pour alléger le travail. On peut en citer : SICOMA (Système Informatique pour la Comptabilité des Matières), GRH-Paie (Gestion des Ressources Humaines et Paie), GESBUDGET (Gestion du Budget) et d'autres. Ils sont tous basés sur l'approche d'architectures monolithiques.

En effet, depuis 2019 des enseignants du département d'informatique, la DISI (Direction de l'Informatique et des Systèmes Informations) et la DRH (Direction des Ressources Humaines) ont écrit un projet portant sur la dématérialisation des procédures et documents administratifs. L'objectif général de ce projet est de réaliser un système global et sécurisé centré sur l'organisation et le personnel. Il permettra aussi de donner des statistiques, d'aider à la prise de décisions, à la recherche d'informations et à l'archivage. Cependant avec une vision globale d'un tel système, il n'y a pas encore eu un travail qui tente de se pencher de la question.

C'est dans cette optique que notre sujet de mémoire vient en appui, proposer une architecture basée sur les micro-services. Cette architecture permettra à l'université Assane Seck de Ziguinchor d'avoir un système global et flexible pour la dématérialisation de ses procédures et documents administratifs.

# ABSTRACT

Since its creation, the Assane Seck University of Ziguinchor has continued to evolve in several areas. This gradual evolution can make it complex to manage in the future. Reasons why the authorities have the will to anticipate the complexity of the management of the university. Thus, to lead to the good management of the institute, the use of computer systems is essential. This is why we want to move to the dematerialization of all administrative procedures and documents of the university.

These issues that accompany this dematerialization are understood by the administrative authorities of the university. Indeed, there are software that can be used on a daily basis to lighten the work. These include: SICOMA (Computer System for Materials Accounting), GRH-Payroll (Human Resources and Payroll Management), GESBUDGET (Budget Management) and others. They are all based on the monolithic architecture approach.

Indeed, since 2019 some teachers of the computer science department, the DISI (Computer Science and Information Systems Department) and the HRD (Human Resources Department) have written a project on the dematerialization of administrative procedures and documents. The general objective of this project is to achieve a global and secure system centered on the organization and the personnel. It will also provide statistics, help with decision-making, information research and archiving. However, with a global vision of such a system, there has not yet been a work that attempts to address the issue.

It is in this perspective that our subject comes in support, proposing an architecture based on micro-services. This architecture will allow the Assane Seck University of Ziguinchor to have a global and flexible system for the dematerialization of its administrative procedures and documents.

# REMERCIEMENTS

*Tout d'abord, nous tenons à remercier ALLAH (SWT) le tout puissant, nous le rendons grâce pour nous avoir donné la volonté, la santé et la patience pour élaborer ce travail. Nous prions également sur son noble prophète Seydina MOUHAMED (PSL).*

*Ensuite, nous remercions nos deux encadrants Dr. Ibrahima DIOP et Pr. Youssou DIENG pour leur engagement, rigueur, dévouement et suivi permanant qu'ils ont portés sur notre travail depuis le début jusqu'à son aboutissement.*

*Nous tenons aussi à remercier les autres membres du jury dont le président Pr. Youssou FAYE, M. Bassirou DIENE et Dr. Papa Alioune CISSE qui ont accepté d'évaluer notre travail.*

*Je veux aussi témoigner mes remerciements à l'ensemble les enseignants du département d'Informatique, les intervenants externes et l'équipe pédagogique pour toutes connaissances théoriques et pratiques qu'ils m'ont apportés tout au long de ma formation.*

*Je remercie tous mes camarades de promotion : jamais je n'ai vu autant de coopération dans une promotion. Ces années de formation furent un plaisir et nul doute que notre réussite individuelle passe avant tout par ces liens que nous avons pu tisser et l'ambiance qui régnait.*

*Nos vifs remerciements vont également à l'endroit de ma famille d'accueil à Ziguinchor, qui m'a intégré et m'a considéré comme leur fils : grâce à elle je ne me souciais de rien depuis mon arrivée dans cette région, donc un grand merci à M. Ibrahima BA et sa femme Mariama TOURE à M. Sidiya DIENG et safemme Nafissatou NDIAYE.*

*Nous tenons vraiment à remercier tous les amis et camarades de près ou de loin qui m'ont apporté un soutien moral ou physique pour la réussite de ce travail.*

*Un grand merci à toutes et à tous.*

# DEDICACES

*C'est avec un immense plaisir que je dédie ce modeste mémoire de fin d'études :*

*À mes très chers parents Djimby DIONGUE et Alassane MARE qui ont tout fait pour moi et qui ne cessent de manifester leurs soutiens et prières à mon égard. Qu'ALLAH leurs donne longue vie.*

*À mon oncle : Mbaye DIONGUE, qui m'a élevé depuis tout petit et qui s'est sacrifié nuit et jour pour mon éducation et ma réussite.*

*À mes oncles : Macoura DIONGUE, Mame Ndiaga DIONGUE et Ibrahima THIAM, pour leurs soutiens et encouragements qu'elles m'ont apportés depuis tout petit.*

*À ma grand-mère Anta SECK et mon cousin Moussa DIONGUE qui représentaient beaucoup pour moi. Que le bon Dieu leur renouvelle miséricorde et les accueils au paradis céleste.*

*À mes tantes : Fatou DIONGUE, Maguette DIONGUE, Mbène THIAM et Astou THIAM pour leurs soutiens et encouragements qu'ils m'ont toujours apportés depuis tout petit.*

*À mes très chers frères et sœurs, particulièrement à Fatou SOW, Bathie MARE et Talla HANN.*

*À mon ami et frère Sidiya DIENG et sa femme Nafissatou NDIAYE. Je n'ai même pas les mots pour les qualifier, ils m'ont toujours conseillé et encouragé pour la réussite de ce mémoire.*

*À mes parents de Ziguinchor, Monsieur Ibrahima BA et madame Mariama TOURE ainsi que toute la famille BA.*

*À tous mes amis, plus particulièrement Anna BAKHOUM, Adja Ndickou DIOP, Camir MALACK, Korka DIALLO, Tahirou DIALLO, Mamadou DIALLO et Ababacar DIATTA. Vos soutiens, conseils et encouragements me tiennent vraiment à cœur.*

# SOMMAIRE

# LISTE DE FIGURES

# LISTE DE TABLEAUX

# LISTE DES ABREVIATIONS

| | |
|---|---|
| **API :** | Application Programming Interface |
| **CDAP :** | Chef de Division Administratif du Personnel |
| **DOM :** | Document Object Model |
| **DRH :** | Direction des Ressources Humaines |
| **DCOM :** | Distributed Component Object Mode |
| **HTTP :** | HyperText Transfer Protocol |
| **IaaS :** | Infrastructure as a Service |
| **JSON :** | JavaScript Object Notation |
| **PER :** | Personnel Enseignant et de Recherche |
| **PATS :** | Personnel Administratif Technique et Service |
| **RMI :** | Remote Invocation Method |
| **REST :** | REpresentational State Transfer |
| **RPC :** | Remote Procedure Call |
| **SOA :** | Architectures Orientées Services |
| **UASZ :** | Université Assane Secck de Ziguinchor |
| **UI :** | Interface utilisateur |

# INTRODUCTION GENERALE

La première révolution numérique de l'administration consiste à la dématérialisation des procédures. Aujourd'hui la plupart des services publics comme privés spécialisés (bibliothèques, entreprises, universités) ont un système informatique permettant de gérer les inscriptions, les réservations de documents, l'accès à l'information, etc. Développer l'accès aux documents et procédures administratives de manière dématérialisée, permettra d'augmenter la qualité de services publics, de développer la transversalité dans l'administration, et de réduire les coûts économiques et environnementaux qui sont induits par les procédures encore traitées sur papier.

L'université Assane SECK de Ziguinchor depuis sa création, tente d'améliorer son système d'information en mettant en place des systèmes informatiques de gestion, lui permettant d'automatiser les tâches et de dématérialiser les documents et les procédures administratifs qu'elle produit. Cette volonté de dématérialisation les procédures et les documents administratifs lui a value aujourd'hui de disposer des outils informatiques pouvant alléger le travail quotidien du personnel, bien qu'il reste beaucoup à faire pour les améliorer et les compléter. Ainsi plusieurs travaux d'étudiants en informatiques sur l'idée de la dématérialisation ont été réalisés et d'autres sont en cours de réalisation à travers leur mémoire de fin d'études.

Les travaux réalisés dans le cadre de ces mémoires sont très riches en termes de contenu. Cependant, tous ces systèmes conçus sont basés sur une architecture d'approche monolithique et elle est associée aux systèmes d'anciennes générations. Il s'agit d'une unique pile d'applications qui rassemble toutes les fonctionnalités au sein d'une seule application. Elle est étroitement couplée, tant au niveau des interactions entre les services qu'au niveau des processus de développement et de déploiement. La mise à jour ou la mise à l'échelle d'un seul composant d'une application monolithique peut avoir des effets sur toute l'application, ainsi que sur son infrastructure sous-jacente.
Après chaque changement apporté au code de l'application, il faut publier à nouveau l'application tout entière. De ce fait, la communication et le partage de données restent impossible dans une telle architecture.

Face à ces problèmes, nous avons réfléchi sur une solution faisant recours aux architectures modernes qui essaient de décomposer les services en fonctionnalités ou en capacités métier,

pour fournir davantage d'agilité. Notre solution consiste à proposer une architecture logicielle dont les composants sont autonomes mais aussi partagent des informations via des services qu'elles s'offrent mutuellement. Cette architecture sera basée sur l'approche micro-services. Ces derniers désignent à la fois une architecture et une approche de développement logiciel, qui consiste à décomposer les applications en éléments plus simples, indépendants les uns des autres. Chacun de ces éléments est un micro-service.

Les micro-services sont distribués et faiblement couplés, ils n'ont donc pas d'incidence les uns sur les autres. Ces caractéristiques offrent des avantages au niveau de l'évolutivité dynamique et de la tolérance aux pannes : il est possible de mettre à l'échelle des services individuels sans nécessiter une infrastructure lourde, et d'effectuer leur basculement sans affecter les autres services. Avec une architecture de micro-services, l'objectif est de fournir rapidement des logiciels de qualité mais aussi de développer plusieurs micro-services simultanément. De plus, puisque les services sont déployés indépendamment, on n'a pas à recréer ou redéployer toute l'application après chaque modification.

En effet, le présent mémoire décrit les étapes de la mise en œuvre de notre solution et les notions de base. Il est divisé en cinq grands (5) chapitres dont le :

- **CHAPITRE 1 : Description et contexte justificatif du sujet,** composé de trois sections principales dont la première fait la présentation du projet, la deuxième traite le diagnostic de l'existence, la troisième et dernière section réservée à la problématique.
- **CHAPITRE 2 : Etat de l'art sur les micro-services** qui comporte quatre sections, la première porte sur les micro-services, la deuxième fait la comparaison de l'architecture monolithique et micro-services, la troisième porte sur l'architecture micro-services et la dernière parle de micro-services et les APIs.
- **CHAPITRE 3 : Spécification des besoins fonctionnels** qui est composé de deux grandes sections, dont la première porte l'identification des acteurs et la deuxième porte sur l'identification des besoins fonctionnels
- **CHAPITRE 4 : Conception et implémentation des micro-services,** qui est divisé en deux grandes sections dont la première fait l'objet de la conception et la deuxième est consacrée à l'implémentation.
- **CHAPITRE 5 : Présentation et fonctionnement des micro-services** composé de deux sections, la première porte sur la présentation des micro-services et la deuxième évoque le fonctionnement de nos prototypes micro-services.

Notre document sera bouclé par une conclusion générale et des perspectives pour ce travail.

# CHAPITRE 1 : Description et contexte justificatif du sujet

Ce premier chapitre introductif sera divisé en trois sections. Dans la première, nous allons faire la présentation du projet, ensuite parler du diagnostic de l'existant à l'UASZ dans la seconde section. Dans la troisième et dernière section, nous évoquerons la problématique.

## 1.1 Présentation du projet

L'université Assane Seck de Ziguinchor (UASZ) est un établissement public à caractère administratif, doté de la personnalité juridique et de l'autonomie financière. Elle a pour missions de former des cadres supérieurs, de contribuer à la recherche scientifique et de promouvoir le développement de nos valeurs culturelles. L'exécution des tâches dévolues à l'université, conformément aux dispositions, est répartie entre des cellules administratives et techniques organisées en directions comme il ressort du tableau des organigrammes. L'université a une organisation et un personnel chargé de l'administration, de la mise en œuvre des programmes d'enseignement pour les étudiants et de l'impulsion de la recherche. Elle dispose de biens matériels et de biens financiers pour l'accompagnement de la communauté universitaire composée du personnel PER, PATS et des étudiants, dans l'atteinte de ses objectifs de service public. Dans sa mission de service public, l'université a mis en place une disposition pour permettre la circulation, l'exploitation et l'archivage du courrier. L'intérêt de cette disposition est d'éviter les omissions ou les pertes de courriers mais aussi d'officialiser les dates de réception pour les documents ayant valeur juridique ou contractuelle (offre de services par exemple). L'université dispose aussi d'un réseau avec une architecture et de systèmes d'informations toujours sollicitées pour l'atteinte de ses objectifs de service public. Ainsi, nous proposons la figure suivante qui décompose la gestion administrative et pédagogique de l'UASZ dans huit (8) sous domaines [1]. Ces derniers sont représentés par la figure 1-1 ci-dessous :

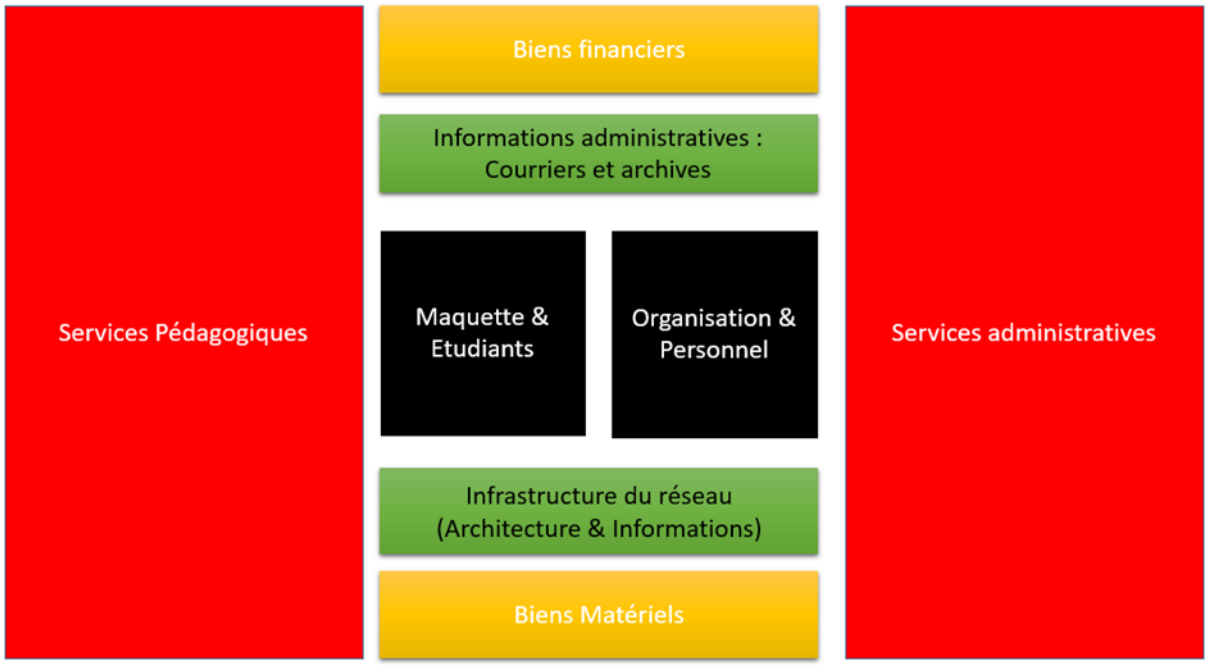


*Figure 1-1 : Les sous domaines de la gestion administrative et pédagogique de l'UASZ [1].*

## 1.2 Diagnostic de l'existence

La **figure 1-1** précédente montre le découpage en sous domaines de la gestion administrative et pédagogique de l'UASZ. Ce découpage sera le point de départ pour :

- faire un diagnostic de l'existant en termes d'outils informatiques au sein de l'université;
- cerner les besoins actuels de l'UASZ pour la dématérialisation dans les procédures et documents administratifs.

Ainsi, ce découpage est structuré en huit points traitant de l'état des lieux de la numérisation à l'UASZ dans les huit (8) sous domaines cités plus haut.

### 1.2.1 Organisation et personnel de l'UASZ

Le manuel de procédure est l'outil dont dispose aujourd'hui l'UASZ pour décrire sa gestion à travers sa composition, ses missions et ses procédures administratives. Ainsi dans le manuel de procédure, il est décrit les organes (le conseil d'administration, le Recteur, le conseil de gestion) et la structuration (les UFRs, les départements) de l'UASZ. La confection de ce manuel de procédure n'est pas sans difficulté. En effet, il est écrit comme un rapport au format WORD sans dispositif de travail collaboratif. Une application informatique avec un tel dispositif permettrait non seulement la diminution de la charge de travail mais aussi aiderait à la lisibilité et à la génération automatique en PDF en cas d'évolution mineur. En effet, l'institution évolue

avec son organigramme et ses procédures. Avec le format actuel, toute évolution des procédures implique une relecture et une modification susceptible d'engendrer des contradictions. Avec un système de mise à jour automatique certains changements mineurs dans les procédures peuvent être automatisés pour permettre d'avoir toujours un manuel à jours. En plus de décrire l'organisation de l'institution, une telle application peut aussi permettre de décrire le personnel dont dispose l'UASZ. Cela permettrait de disposer :

- des coordonnées (matricule, nom, prénom, adresse,…) ;
- des informations personnelles (nom partenaire, nombre d'enfants, …) ;
- des Curriculums Vita (CV) ;
- des dossiers personnels contenant toutes autres informations ou documents pour prouver le contenu du CV (attestations de formations, les articles scientifiques,…).

La rédaction du manuel de procédure est longue et pénible parce que l'UASZ ne dispose pas d'un logiciel de description et de gestion des informations sur son organisation et son personnel. Cette pénibilité se manifeste aussi dans la gestion du personnel. C'est ce qui fait qu'aujourd'hui le bureau du DRH est encombré de papiers (voir les figures 1-2 et 1-3).


*Figure 1-2 : Bureau du DRH [1]*

Il faut aussi souligner le risque encouru en cas d'incendie vu que tous les documents administratifs du personnel de l'UASZ sont dans le bureau du DRH, sans système informatique de sauvegarde. En 2018, un incendie au Rectorat de l'UGB avait fait disparaitre intégralement les dossiers de la DRH (voir la figure 1-3).

*Figure 1-3 : Bureau du DRH de l'UGB (à droite), Bureau du SG de l'UGB (à gauche) [1].*

### 1.2.2 Gestion des informations pédagogiques

Pour la gestion des informations pédagogiques on utilise à l'UASZ un logiciel fournit par l'AUF. [2]

Ce logiciel permet aux chefs de services pédagogiques des UFR de :

- effectuer les inscriptions pédagogiques ;
- générer les listes d'émargement lors des devoirs et examens ;
- gérer les notes et les délibérations ;
- imprimer les attestations de réussites.

Une version web de ce logiciel est en cours de développement par l'AUF, selon le chef de service pédagogique de l'UFR ST. Cette version va alléger le travail du CSP en intégrant les acteurs (enseignants, étudiants, …) dans le processus de la gestion pédagogique.

La finalité de la gestion des informations pédagogiques est aussi de pouvoir retracer le parcours d'un étudiant de l'UASZ même après 20 ans.

### 1.2.3 Gestion des biens matériels

Dans le manuel de procédure, les matières de l'Université Assane Seck de Ziguinchor sont séparées entre les matières du rectorat et les matières des UFR et pour la gestion de ces matières trois rubriques sont proposées :

- l'organisation des matières ;
- la gestion des matières ; et
- la gestion du stock.

Actuellement l'UASZ ne dispose pas d'un outil informatique pour la gestion des matières et du stock. Ce qui existent aujourd'hui, c'est deux outils informatiques l'un nommé SICOMA [2] et l'autre nommé SAGESTOCK. [4]

SAGESTOCK est sous une licence payante. Il est utilisé par le comptable des matières de l'UFR des Sciences et Technologies pour la gestion de stock de façon générale, dispose de fonctionnalités (comme la vente de produits) qui ne sont pas utilisées dans le contexte de la gestion du stock d'UFR ST. SAGESTOCK dans sa version 1.0 ne dispose pas de fonctionnalités pour la réception de matières, la gestion des bon d'entrés, et la gestion des bon de sortie, le livre-journal, etc. Parce qu'il n'est pas conçu pour gestion de la comptabilité des matières mais pour la gestion de stock. C'est pourquoi le comptable des matières l'utilise pour la gestion de son stock mais pour ce qui est de la gestion des matières ce dernier le fait de façon manuelle.

SICOMA est développé, lors d'un mémoire de master, par un étudiant de l'UASZ. Il est actuellement utilisé à la direction de la gestion du patrimoine et de la maintenance pour la gestion des matières du Rectorat. SICOMA dispose des fonctionnalités pour la réception de matières avec la production de PV, la gestion des bon d'entré et des bons de sortie et le suivie des opérations enregistrées par jour dans un livre-journal.

L'UASZ dispose d'une application pour la gestion des biens matériels [5], mais il faut noter que certaines tâches pour la comptabilité des matières se font toujours manuellement. Ce qui la rend fastidieuse et justifiant certains problèmes que rencontre notre institution pour sa comptabilité des matières.

Nous allons ici mettre en évidence d'autres manquements du système de gestion des biens :

- **le catalogue des biens** : une matière donnée ne dispose pas de catalogue permettant d'avoir un aperçu de l'élément avec toutes ces caractéristiques ;

- **le référencement des matières** : la matière n'est pas référencée à l'UASZ et on ne dispose pas de système d'identification unique du matériel qui peut être fait par un code à barre et permettre une lecture automatique des références sur les matières de l'UASZ par lecteur de code à barre;
- **la traçabilité des matières** : pour un élément donné, nous ne gardons pas ses mouvements à travers les différentes structures ainsi que ses différents possesseurs au fil du temps ;
- **les alertes pour contrôler le stock et le déplacement des matières ;**
- **la sécurité** : une matière déjà attribuée peut être déplacé vers un endroit où il n'est pas censé être. Comme dans les figures 1-4 et 1-5 suivantes les tables qui doivent être dans les salles sont souvent déplacées dans les couloirs ou sous les arbres ;

*Figure 1-4 : Suivi du déplacement et l'endroit des matières [6].*

*Figure 1-5 : Manque de tables dans les salles de classes [6].*

### 1.2.4 Gestion des composants et de l'architecture du réseau informatique

Les composants du réseau de l'UASZ sont divers et variés. On peut citer comme exemples de composants : les serveurs, les switch, le câblage, etc.

L'architecture du réseau permet de voir comment ces composants sont organisés dans le réseau informatique de l'UASZ. Cette architecture est importante, car elle permet aux administrateurs de pouvoir mieux situer les problèmes.

Les composants du réseau informatique de l'UASZ sont en évolution car l'université et ses besoins évoluent. Ainsi l'architecture du réseau évolue aussi dans l'espace et dans le temps. Il est donc important de disposer d'une application qui permet de gérer les composants et l'architecture du réseau de l'UASZ. Cette application peut nous donner une cartographie des composants et de l'architecture du réseau informatique.
Ce qui sera très utile lors de changement d'équipe. Une nouvelle équipe d'administrateurs de disposer d'informations sur le réseau et de se baser sur ces informations pour améliorer le service et évite de toujours repenser le réseau de l'UASZ.

### 1.2.5 Gestion des biens financiers

La gestion des biens financiers de l'UASZ est décomposée, dans le manuel de procédures [7], en deux parties, que sont : la gestion budgétaire et la mobilisation et le décaissement des ressources. La gestion budgétaire se charge de la préparation, de l'élaboration et de l'approbation, de l'exécution et du suivi du budget. Il faut aussi, dans la partie mobilisation et décaissement des ressources, gérer la mobilisation des ressources (venant des bailleurs de fonds, de l'Etat et de l'UASZ), la trésorerie et le règlement des achats de biens, travaux et services.
La gestion des biens financiers de l'UASZ dispose d'un outil informatique **GESBUDGET** [8] pour la gestion budgétaire (aider à la confection, le Vote et le Déroulement du budget). Ce système de gestion du budget est d'une architecture client lourd (monoposte) ce qui fait qu'il utilise des documents Excel pour recueillir les informations sur le budget. Ce qui ne favorise pas une gestion collaborative, transparente et sécurisée des informations financières de l'UASZ. Car tous les acteurs n'accèdent pas au système de façon autonome et selon leur niveau de responsabilité (ou profile) et ainsi faire leurs tâches et être alerté en cas d'opération dans le budget.

### 1.2.6 Gestion de l'information administrative

L'UASZ gère deux formes de courrier à savoir : courrier physique et courrier électronique. Pour cela, elle dispose de procédures visant à expliquer la circulation, l'exploitation et l'archivage du courrier arrivé (le courrier provenant de l'extérieur de l'université ou des UFR) et du courrier départ (le courrier destiné à l'extérieur).

La gestion et l'archivage du courrier à l'UASZ reste jusqu'à ce jour essentiellement manuel. C'est ce qui produit des documents pour la gestion du courrier. Ces documents sont :

- Registres courrier (départ et arrivée) ;
- Fiche d'exploitation du courrier ;
- Fiche de circulation ;
- Fiche interne de validation ;
- Bordereau d'envoi de courrier ;
- Chrono de classement.

Ainsi, l'université doit disposer d'outils informatiques pour la gestion et l'archivage du courrier. Un tel outil pourra être comme un **système de description** (de dématérialisation du Registre du courrier, de la Fiche d'exploitation du courrier, de la Fiche de circulation, de la Fiche interne de validation, du Bordereau d'envoi de courrier, du Chrono de classement) **d'alerte**, **de numérisation** et **d'archivage** du courrier physique de l'UASZ.

### 1.2.7 Gestion des services rendus

L'université, en plus de son service d'enseignement et de recherche, offre des services aux étudiants et à son personnel (PER et PATS). Ainsi, elle donne aux étudiants des services pour :

- **inscription académique** : on dispose d'un système informatique [9] pour l'inscription académique des étudiants. Mais même avec ce système, on note que l'inscription n'est pas rapide et reste pénible pour les étudiants et pour la scolarité de l'UASZ. La cause est que ce système n'intègre pas les avancés des technologies du web, du payement (en ligne ou mobile) pour l'amélioration du service d'inscription académique ;
- **inscription pédagogique** : on dispose d'un système de gestion pédagogique [10] pour pédagogiques ;
- **alerte d'évènements ou d'informations universitaires importants** : Aujourd'hui l'université ne dispose pas d'un système pour alerter les étudiants des évènements ou informations importants (tels que les dates de début et de fin des semestres, les fêtes, les

horaires des services en période de vacances ou de fêtes, …). Ce qui existe aujourd'hui, c'est la mailing liste des étudiants ;

- **disponibilité des emplois du temps** : il existe un système informatique pour la gestion des emplois du temps [11]. Ce système met en entrée beaucoup d'informations comme : les maquettes, les choix des enseignants, les salles et les préférences pour générer tous les emplois du temps pour un semestre donné. Le logiciel génère et envoie tous les emplois du temps des classes ;
- **consultation des résultats** : par affiche et non par mail;
- **réclamation de notes** : ce font chez le chef service pédagogique et non en ligne;
- **réception des attestations et des diplômes** : Ce qu'on note, aujourd'hui, c'est que les étudiants reçoivent des attestations et pratiquement pas de diplômes ou rarement dans les Université Sénégalaise ;
- etc.

Les services de l'UASZ offerts à son personnel (PER et PATS) peuvent permettre aux ayants droit de :

- **disposer de leur bulletin de salaire** : Actuellement, l'UASZ dispose d'un logiciel nommé GRH-Paie [12], mais on imprime les bulletins de salaires par trois mois et on modifie les variables de salaire seulement avant le 15 de chaque mois [13];
- **demander leurs congés** : Il n'existe pas de système informatique pour la gestion des congés à l'UASZ, alors que la majeure partie des agents prennent **des congés fractionnés**. Le suivi des congés des agents est donc rendu plus difficile avec ce fractionnement qu'il est possible de faire pendant trois (3) ans. Ce qui cause un véritable problème pour la gestion des congés à l'UASZ [14] ;
- **demander des imputations** : Il n'existe pas de système informatique pour la gestion des imputations à l'UASZ. Ce qui existe c'est des formulaires en papier, qu'il faut remplir et signer. Ce qui reste très pénible pour la DRH de l'UASZ et les agents obligés de faire le déplacement ;
- **demander un voyage d'étude et déposer son rapport de voyage d'étude** : Il n'existe pas de système informatique pour la gestion des voyages d'étude. Ce qui existe c'est un formulaire qui est envoyé par mail. Ce formulaire est rempli et signé par l'enseignant, puis acheminer par voie hiérarchique vers le vice-recteur chargé de la recherche. Ce qui reste très pénible pour les enseignants et le vise rectorat chargé de la recherche et de la coopération de l'UASZ ;

- **demander des formations et déposer son rapport de participation** : Il n'existe pas de système informatique pour la gestion des demande de formation. Ce qui existe c'est un formulaire qui est envoyé par mail. Ce formulaire est rempli et signé par l'agent, puis acheminer par voie hiérarchique vers le Rectorat.
- **être alerté des évènements ou informations universitaires importants** : Aujourd'hui l'université ne dispose pas d'un système pour alerter le personnel des évènements ou informations importants (tels que les dates de début et de fin des semestres, les fêtes, les horaires des services en période de vacances ou de fêtes, …). Ce qui existe aujourd'hui, c'est la mailing liste du personnel (PER et PATS) ;
- **choisir ses enseignements** : L'UASZ ne dispose pas de logiciel pour le choix des enseignements. Ainsi selon [15], « l'Université Assane SECK de Ziguinchor (UASZ) utilise depuis très longtemps des fichiers Excel, pour la gestion de la répartition des enseignements. Ces fichiers ne répondent pas entièrement aux attentes des utilisateurs, car, non seulement, renferment beaucoup de manquements en termes de fonctionnalité (détection des conflits lors des choix, visualisation des répartitions pour chaque département, etc.), mais aussi des problèmes en termes d'usage (manque d'autonomie, de commodité et de rapidité dans le processus, difficulté de constituer le fichier final des répartitions) ». La centralisation des fichiers au vice-rectorat est très difficile car elle se fait par échange de mail ;
- **disposer des emplois du temps** : à l'UASZ, on dispose de système de gestion des emplois du temps. Mais d'un système centralisé au tour d'un seul acteur, qui fait tout et envoie les emplois du temps à tout le monde à la fin. Ce qui est un réel problème pour une gestion collaborative et transparente ;
- **disposer d'un cahier de texte** : dans le déroulement des cours à l'UASZ, rare sont les départements, qui utilisent un cahier de texte en format papier. Il est très utile, car contenant les informations du déroulement de chaque cours. Avec les informations sur ce cahier de texte, on pourra savoir l'évolution du semestre, sans envoyer des mails (comme ça ce fait actuellement) pour demander aux enseignants de donner les niveaux d'exécution de leurs cours ;
- **déclarer ses heures supplémentaires et de vacations** : à partir des informations dans les cahiers de texte et des choix d'enseignements, on peut déduire les déclarations des heures supplémentaires et de vacations. Ce qui est très pénible aujourd'hui pour les enseignants de l'université. Ceci parce que sans cahier de texte, même au format papier,

l'enseignant qui déroule son cours sans l'écrire et se trouve à la fin du semestre dans l'obligation de se rappeler de toutes ces informations ;

- **recruter de nouveaux collègues** : Aujourd'hui, le recrutement d'un enseignant ne se passe pas de manière facile et transparente. Parce qu'on ne dispose pas de système informatique qui publie les postes (avec le contenu du dossier de candidature, la grille de sélection, …), donne la possibilité de dépôt des dossiers et sélection par ordre de mérite suivant la grille et le contenu de dossier reçus. Un tel système aiderait les enseignants à choisir le meilleur candidat dans la transparence, la facilité et la rapidité ;
- etc.

A côté de ce contexte d'utilisation de l'informatique dans la gestion des affaires de l'UASZ, il existe des enjeux énormes et une volonté affichée dans notre institution de dématérialiser les documents et procédures administratives, comptables et financières.

## 1.3 Problématique

L'université Assane Seck de Ziguinchor connait une croissance depuis sa mise en œuvre. Ainsi de grands chantiers qui sont actuellement en cours peuvent constituer des problèmes de complicités dans l'avenir ; une institution comme celle-ci a besoin de s'acquérir de moyens plus performants et rapides favorisant pleinement sa gestion dans sa globalité. Malgré l'existence de plusieurs systèmes d'informations comme : SICOMA, GRH-Paie, GESBUDGET, GesPédagogique, GestAcadémique etc, il y a aussi plusieurs mémoires d'étudiants d'informatiques qui sont soutenues ou encours sur la dématérialisation des documents et procédures administratives, comptables et financières. On peut citer quelques de ces mémoires :

- Automatisation de la comptabilité des matières à l'UASZ ;
- Conception et Développement d'une application informatique pour l'automatisation de la gestion des congés au sein de la DRH de l'UASZ et la dématérialisation des documents administratifs qui s'y rapportent ;
- Automatisation de la gestion des biens matériels de l'UFR des Sciences et technologies ;
- Automatisation de la Gestion des Salaires et Dématérialisation du Bulletin de Salaire du personnel de l'UASZ ;
- Automatisation de la gestion administrative des départements ;
- Etc.

Les travaux réalisés dans le cadre de ces mémoires sont très riches en termes de contenu. Cependant, leurs objectifs n'étaient pas encore de proposer une architecture générale pour la dématérialisation à l'UASZ. Ainsi, il n'y a pas encore eu un travail qui tente d'avoir une vision globale de la question.

Raison pour lesquelles, nous avons proposé une solution pouvant répondre aux différents problèmes qui sont liées à la dématérialisation des documents et procédures administratives, comptables et financières à l'UASZ.

### 1.3.1 Solution proposée

Les problèmes liés à la dématérialisation des documents et procédures administratives ne manquent toujours pas à l'UASZ malgré l'existence de plusieurs systèmes informatiques qu'elle dispose actuellement et ceux en cours de réalisation. Raisons pour lesquelles nous avons proposé une architecture logicielle générale, compète, évolutive et basée sur les micro-services. Cette solution va répondre aux besoins liés à la bonne gestion de l'institut et une fois mise en place, va permettre de :

- d'anticiper sur les problèmes de complexité qui viendront dans l'avenir en pensant dès maintenant à l'utilisation de l'informatique dans sa gestion ;
- découper les systèmes informatiques existants en micro-services afin de les simplifier pour mieux les gérer ;
- développer d'autres micro-services suivant un ordonnancement tiré de cette architecture ;
- Implémenter les micro-services de façon séparée et indépendante.
- de faire communiquer les micro-services par des services qu'elles partagent mutuellement.
- Utiliser des langages différents pour l'implémentation des micro-services.

### 1.3.2 Objectif général

L'objectif général de ce projet est de réaliser un système global et sécurisé centré sur l'organisation et le personnel et composé de plusieurs sous-systèmes autonomes qui communiquent entre eux pour échanger des informations. Ce système a pour but de donner des statistiques, d'aider à la prise de décisions, à la recherche d'informations et à l'archivage.

### 1.3.3 Objectifs spécifiques

Les objectifs spécifiques du projet sont :

- Proposer une architecture logicielle générale dont les composants sont autonomes mais partagent des informations via des services qu'elles s'offrent mutuellement ;
- Implémenter par des modules et des services suivant un ordonnancement tiré de l'architecture ;
- Tester et utiliser ces composants au fur à mesure ;
- Disposer de code sources propriétaires ;
- Faire de nouvelles versions au fur à mesure qu'on utilise les outils ;
- etc.

A travers ce premier chapitre, nous avons fait la présentation de notre projet dans la première section ainsi que le diagnostic de l'existence dans la deuxième section. Et la dernière section était consacrée à la problématique. Nous pouvons passer au chapitre suivant qui est réservé à l'état de l'art sur les micro-services.

# CHAPITRE 2 : Etat de l'art sur les micro-services

Ce chapitre est devisé en quatre sections, la première dans laquelle nous allons parler des généralités sur les micro-services afin de bien les comprendre. Dans la seconde section nous nous intéresserons sur l'architecture monolithique. Dans la troisième section nous allons parler de l'architecture micro-services. Et la dernière section sera consacrée aux micro-services et API.

## 2.1 Les micro-services

Les micro-services désignent à la fois une architecture et une approche de développement logiciel, qui consiste à décomposer les applications en éléments les plus simples, indépendants les uns des autres. Chacun de ces composants ou processus est un micro-service. Ils sont distribués et faiblement couplés, ils n'ont donc pas d'incidence les uns sur les autres. Ces caractéristiques offrent des avantages au niveau de l'évolutivité dynamique et de la tolérance aux pannes : il est possible de mettre à l'échelle des services individuels sans nécessiter une infrastructure lourde, et d'effectuer leur basculement sans affecter les autres services. [16]

Un micro-service est une petite application destinée à gérer peu de fonctionnalités. C'est une seule brique de l'application globale, une brique indépendante ayant une seule responsabilité indépendante.

Ainsi plusieurs développeurs peuvent travailler sur leurs services individuels en même temps, sans qu'il y ait besoin de mettre à jour toute l'application. Cela permet de réduire la durée du développement et de publier de nouvelles fonctions plus souvent.

La figure 2-1 nous donne un peu en détail sur l'approche micro-services.

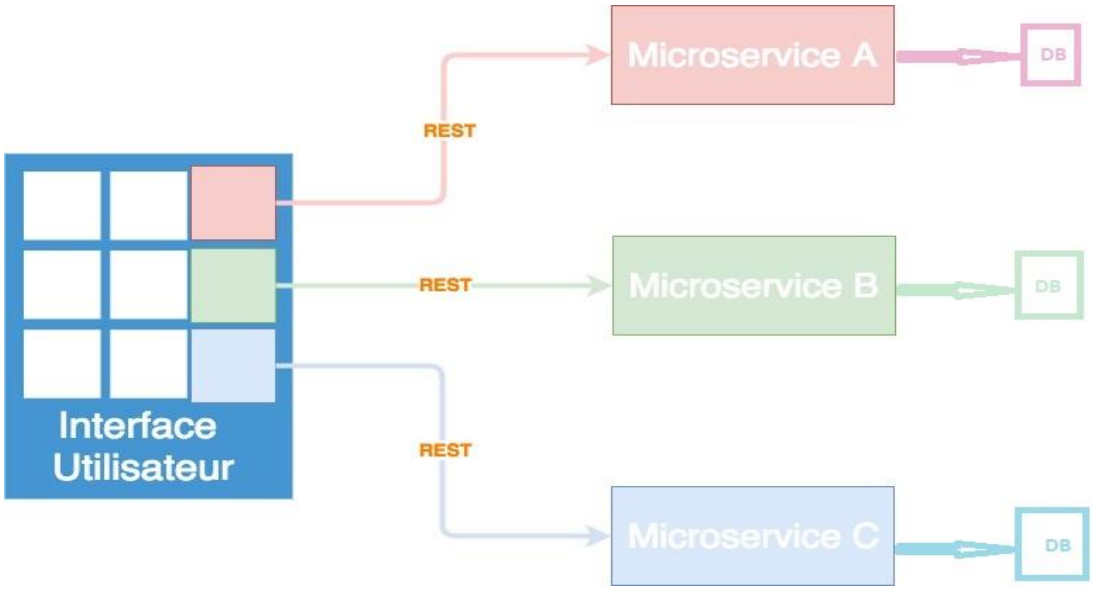


*Figure 2-1 : Composants micro-services d'une application.*

Au niveau de la performance d'une application, les micro-services offrent un certain nombre d'avantages. L'application est décomposée en plusieurs parties, permettant un partage des tâches plus efficaces au sein des équipes de développement. En effet, dans une telle approche, chaque micro-service est indépendant des autres et gère ses propres données (pouvant aller jusqu'à posséder sa propre instance de base de données).

Grâce à cette indépendance, les avantages sont non négligeables pour les équipes de développement qui peuvent désormais maîtriser uniquement un langage pour développer les services qui les concernent.

**Au niveau du service en lui-même,** les équipes peuvent :

- déployer et redéployer un service sans impacter l'ensemble de l'application ;
- effectuer des tests ;
- écrire dans un langage de programmations de leur choix (Go, Ruby, Python, Java, etc.) car les micro-services permettent une interopérabilité au travers du Protocol HTTP.

**Au niveau de la scalabilité de l'application**, cette isolation permet une gestion plus fine des ressources. En cas de besoin, il est facile d'allouer plus de ressources en déployant des instances supplémentaires pour une fonctionnalité de l'application (e.g. la fonction de recherche). On a alors un contrôle de la scalabilité qui est très puissant.

**Au niveau de la gestion des incidents,** une application en micro-service est plus robuste face aux pannes. Seule l'utilisation du composant concerné sera impactée, le reste de l'application pouvant tourner de manière indépendante. [17]

### 2.1.1 Evolution

Plus qu'une nouvelle révolution dans les architectures informatiques, les micro-services sont une évolution de l'architecture SOA. Une évolution logique qui résulte d'un long retour d'expérience dans la mise en place d'architecture SOA, et qui vient gommer certains de ses défauts.

Les architectures informatiques peuvent été assimilées aux modes de l'habillement : une mode chasse la précédente. Sans remonter à l'époque des anciennes, nous constatons ainsi que le client serveur a cédé la place aux architectures web qui a donné naissance au SOA (architecture orientée services). Aujourd'hui, l'architecture informatique en vogue, c'est les micro-services.

Avec l'adoption des protocoles standard XML et HTTP pour la communication entre plateformes, l'architecture orientée services a tenté de définir un ensemble de normes d'interopérabilité. Ainsi, les normes applicables aux services Web qui trouvaient leur origine dans le protocole SOAP (Simple Object Access Protocol) ont été confiées à un comité baptisé Oasis. Il est notamment composé de représentants de Computer Associates, Fujitsu, IBM, Novell et Sun. A travers cet effort, l'Oasis prévoyait simplifier les déploiements en environnements hétérogènes.

Des éditeurs comme IBM, Tibco, Microsoft et Oracle ont commencé à proposer des produits d'intégration applicative pour l'entreprise fondée sur les principes de l'architecture orientée services. Malgré le succès remporté par ces produits auprès des entreprises, les jeunes sociétés Web 2.0 (qui désigne le deuxième âge d'internet et du Web) ont commencé à se tourner vers le protocole REST (REpresentational State Transfer) pour l'informatique distribuée. Alors que JavaScript gagnait du terrain, JSON (JavaScript Object Notation) et REST sont rapidement devenus des normes de fait pour le Web. [18]

### 2.1.2 Caractéristiques micro-services

Une des principales caractéristiques des micro-services est qu'ils sont des unités fonctionnelles logiques de grande précision, chacun exprime un processus autonome, responsable d'une seule fonctionnalité. Et par conséquent, il sera facile de détecter et de remplacer un service défectueux sans perturber les autres services.

Les micro-services se distinguent aussi par leur exploitation des capacités des différentes technologies dans une seule architecture micro-services pour, bien évidemment, réaliser des

tâches spécifiques. En fait, grâce à l'isolation des services nous pouvons basculer facilement d'une technologie à une autre.

Au niveau de l'autonomie, tous les services des composants d'une architecture de micro-services peuvent être développés, déployés, gérés et mis à l'échelle sans affecter le fonctionnement des autres services. Les services n'ont pas besoin de partager leur code ou leur implémentation avec d'autres services. Les composants individuels communiquent par le biais d'API bien définies.

En termes de spécialisation, chaque service est conçu pour un ensemble de fonctionnalités et se concentre sur la résolution d'un problème spécifique. Si les développeurs apportent plus de code à un service au fil du temps, et que le service devient complexe, ce dernier peut être fragmenté en services plus petits. [19]

## 2.2 Architecture monolithique

Une architecture monolithique est un concept de développement logiciel permettant de concevoir des applications dites monolithiques. Une application monolithique est une application qui, centralise tous les besoins, dans un seul bloc applicatif, où toutes les couches communiquent entre elles par des appels de méthodes et elle est réalisée avec une seule technologie. De plus les données sont centralisées dans une seule base de données. [20]
La figure 2-2 illustre l'exemple d'une architecture monolithique.

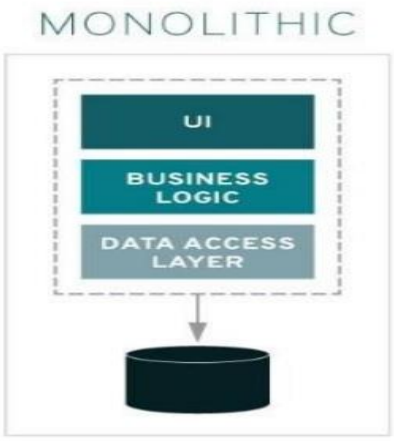


*Figure 2-2 : Architecture monolithique*

Cette architecture monolithique représente le modèle traditionnel unifié de conception d'un programme informatique. Dans ce contexte, « monolithique » signifie former d'un seul bloc.

Un logiciel monolithique est conçu pour être autonome ; ses composants sont interconnectés et interdépendants plutôt qu'associés de manière flexible comme dans le cas des programmes modulaires. Dans ce type d'architecture étroitement intégrée, chaque composant et ceux qui lui sont associés doivent être présents pour permettre un bon fonctionnement du système.

### 2.2.1 Limites d'une architecture monolithique

Tout système informatique devra garantir son évolutivité et sa maintenabilité, surtout que les besoins et les exigences fonctionnelles du client se développent au fur et à mesure. Dans une application monolithique, tout changement conduira vers un système plus complexe. De plus, pour chaque modification nous devrons accéder, dans certains cas, sur tout le système et relire presque tout le code ; ce qui affecte généralement la qualité de code d'une part et rend la maintenance difficile et pénible d'autre part. [21]

Nous pouvons aller plus loin que ceci, après cette modification, nous devrons tout redéployer parce que l'application monolithique est un seul bloc applicatif, donc tout changement, même s'il est minime, conduira à un déploiement complet de toute l'application.

De plus dans une telle approche d'architecture, dès le début du développement d'applications, le code source est entièrement intégré dans une unité de déploiement unique. C'est l'approche « monolithique ». Et par conséquent, si le développeur souhaite apporter la moindre modification à une application existante, il doit effectuer une mise à jour complète accompagnée d'un processus d'assurance qualité spécifique, ce qui rallonge les temps de déploiement et accroit la charge de travail. Si l'application est mise en ligne et que les mises à jour introduisent des erreurs, il faut mettre l'application hors ligne, la déconstruire et la corriger. [22]

Bien vrai que, les architectures classiques ont fait leurs preuves dans plusieurs projets, mais à un moment donné elles deviennent trop coûteuses à faire évoluer d'où l'essor d'un nouveau style architectural : Aujourd'hui, l'architecture micro-services répond idéalement à cette problématique.

## 2.3 Architecture micro-services

L'architecture micro-services est une méthode utilisée pour le développement de systèmes logiciels. Cette architecture permet de traiter une application comme un ensemble de petits services et chaque service est considéré comme un projet à part.

L'approche micro-services est totalement différente de l'approche monolithique dans la conception, le développement et aussi dans le déploiement. En effet, une architecture micro-services est un style architectural logiciel qui décompose les applications complexes en des processus élémentaires, spécialisés pour certaines fonctionnalités.

La décomposition des processus dépend entièrement du métier de l'application et de sa complexité. L'approche micro-services est apparue pour résoudre les problèmes que présentent les applications monolithiques.

Ainsi en cassant une application monolithique, elle devient une série de services indépendants, chacun codé par le langage le plus adéquat, testé et déployé indépendamment des autres. Ces services sont des entités communicantes via des API. Et c'est ici que l'enjeu de l'architecture micro-services persiste, c'est à dire que la décomposition en un ensemble d'unités fonctionnelles joue un rôle majeur dans la réussite d'une telle conception.

La figure 2-3 ci-dessous montre la casse d'une application monolithique en micro-services.

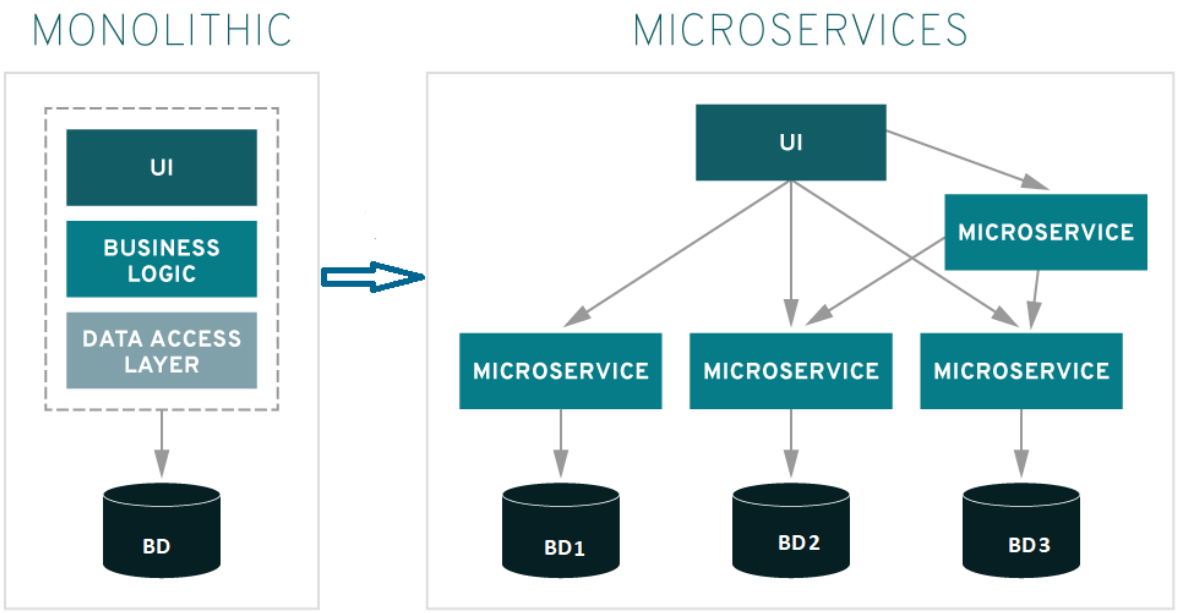


*Figure 2-3 : Architecture monolithique vers architecture micro-services.*

Une architecture de micro-services se compose d'une collection de petits services autonomes et chaque service doit mettre en œuvre une fonctionnalité bien définie. Les services sont associés librement et communiquent via des contrats API.

Avec une architecture de micro-services, une application est développée à l'aide de composants indépendants, qui exécutent chaque processus de l'application sous forme de service. Ces services communiquent par le biais d'une interface bien définie et à l'aide d'API légères. Les services sont développés pour des capacités métier, et chaque service exécute une seule fonction. Étant donné qu'ils sont exécutés de manière indépendante, chaque service peut être mis à jour, déployé ou mis à l'échelle pour répondre aux demandes des fonctions spécifiques d'une application.

La figure 2-4 qui suit représente une architecture micro-services.

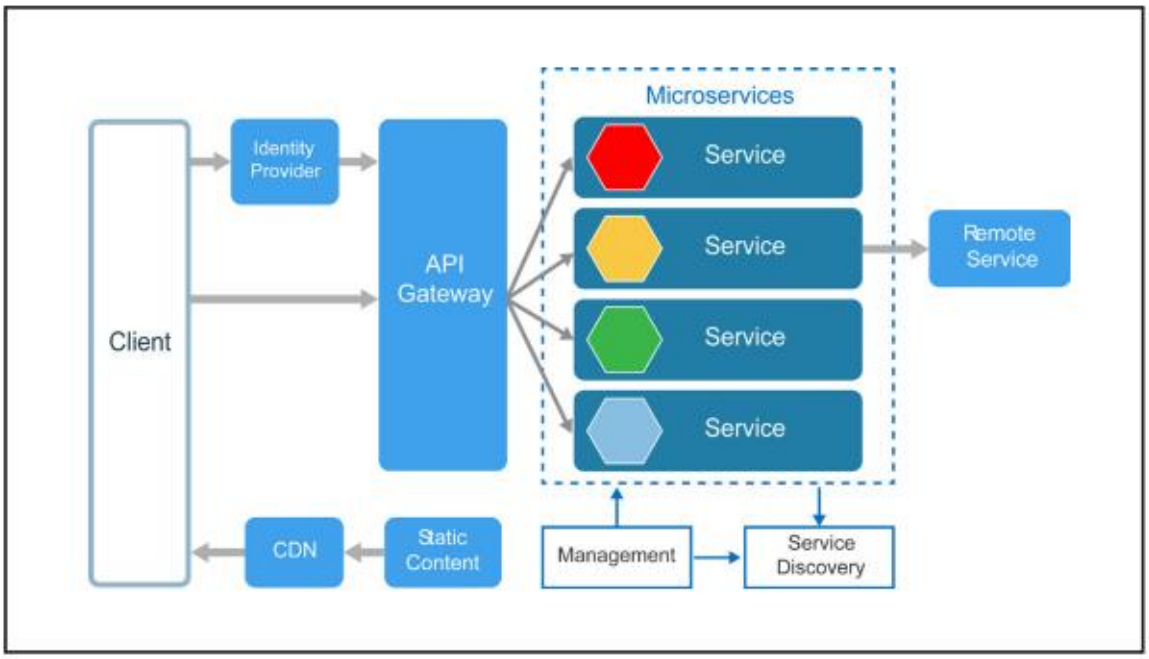


*Figure 2-4 : Architecture micro-services*

À certains égards, les micro-services représentent l'évolution naturelle des architectures orientées services (SOA), mais il y a des différences entre les micro-services et la SOA. Voici quelques caractéristiques qui permettent de définir un micro-service :

- dans une architecture de micro-services, les services sont de petites tailles, indépendantes et faiblement couplés entre eux ;
- chaque service constitue une base de code distincte, qui peut être gérée par une équipe de développement restreinte ;
- les services peuvent être déployés de manière indépendante. Une équipe peut mettre à jour un service existant sans avoir à recréer et à redéployer toute l'application ;

- les services sont eux-mêmes chargés de la persistance de leurs données ou de leur état externe, à la différence du modèle classique, où la persistance des données est gérée par une couche de données distincte ;
- les services communiquent entre eux à l'aide d'API bien définies. Les détails de la mise en œuvre interne de chaque service ne sont pas visibles par les autres services ;
- les services n'ont pas besoin de présenter une pile technologique, des bibliothèques ou des infrastructures identiques ;

Outre les services eux-mêmes, une architecture de micro-services type présente un certain nombre d'autres composants :

**Gestion** : Le composant de gestion est chargé notamment de la mise en place des services sur les nœuds, de l'identification des défaillances et du rééquilibrage des services entre les nœuds**.**

**Découverte des services** : Assure la conservation d'une liste des services et des nœuds où ils se trouvent. Permet la consultation des services pour trouver le point de terminaison d'un service.

**Passerelle API** : La passerelle API constitue le point d'entrée pour les clients. Les clients n'appellent pas les services directement. Au lieu de cela, ils appellent la passerelle API, qui transfère l'appel aux services appropriés sur le back end. La passerelle API peut agréger les réponses de plusieurs services et renvoyer la réponse agrégée. [23]

### 2.3.1 Transition des micro-services

Les applications actuelles s'appuient sur l'intégration et le déploiement continus pour obtenir une itération rapide. Pour tirer parti de cette approche, l'application est subdivisée en petites unités fonctionnelles indépendantes. Chaque unité est attribuée à une équipe qui est chargée de l'améliorer. Grâce à ce fonctionnement, les équipes peuvent livrer rapidement de nouvelles versions des micro-services sans perturber d'autres parties de l'application.

L'évolution de l'Internet des objets (IoT) et de la communication machine-à-machine (M2M) exige de structurer différemment les modules d'application. Chaque module doit s'occuper d'une seule tâche qui intervient dans le workflow global. Pour chacune des parties d'une application, les développeurs choisissent les langages, les frameworks et les outils les plus appropriés. [24]

La figure 2-5 montre la transition d'un monolithique vers micro-services.

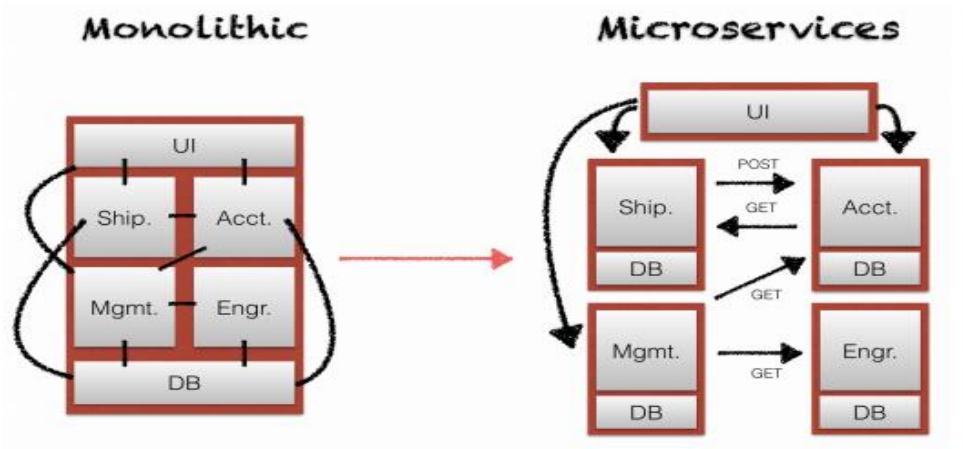


*Figure 2-5 : Transition vers les micro-services*

La technologie des conteneurs, permet de porter le code d'un environnement à l'autre. Les développeurs peuvent transférer en toute transparence le code écrit sur leurs machines de développement vers des machines virtuelles, ou vers le Cloud. Chaque conteneur qui s'exécute est un ensemble complet, du système d'exploitation au code d'exécution de la tâche.

L'infrastructure sous forme de code est un concept puissant : il permet aux développeurs de gérer l'infrastructure sous-jacente via un programme. On peut ainsi provisionner, configurer et orchestrer de façon dynamique quelques centaines de serveurs virtuels.
Combinée aux conteneurs, cette fonctionnalité offre de puissants outils comme Kubernetes pour le déploiement dynamique de clusters exécutant des micro-services.

Pour chacune des parties d'une application, les développeurs choisissent les langages, les frameworks et les outils les plus appropriés. Ainsi, une grande application peut se composer de micro-services écrits en Node.js, Ruby on Rails, Python, R et Java. Chaque micro-service est écrit dans le langage le plus adapté à la tâche.

C'est également le cas de la couche de stockage permanent. Les applications Web ont toujours plus besoin de stockage objet, de stockage de données semi-structurées et structurées, et de cache en mémoire pour garantir la persistance. Les micro-services facilitent l'adoption d'une stratégie polyglotte en termes de langages de codage et de bases de données. [24]

### 2.3.2 Les avantages de l'architecture micro-services

Avec les micro-services, développeurs et opérateurs peuvent développer et déployer des applications auto-adaptatives. Chaque micro-service étant autonome et indépendant, il est facile de détecter et de remplacer un service défectueux sans perturber d'autres services.

La technologie des micro-services permet aux entreprises d'investir dans des composants modulables et réutilisables. A l'inverse des applications monolithiques, celles à base de micro-services peuvent évoluer de manière sélective.

Un avantage avancé est que lors d'un besoin critique en une ressource, seul le micro-service lié sera augmenté, contrairement à la totalité de l'application dans une architecture classique, par exemple une architecture trois tiers. Cependant, le coût de mise en place, en raison des compétences requises, est parfois plus élevé.

Adrian Cockford, le concepteur de l'architecture de Netflix, l'une des architectures informatiques en micro-services les plus avancées actuellement, a qualifié ce type d'architecture d'une SOA "à grain fin" car, dans un tel environnement, ce sont des centaines, des milliers de services qui sont instanciés chaque jour pour faire face aux requêtes des utilisateurs. Et chacune d'elles peut mobiliser des centaines de micro-services pour délivrer la réponse demandée.

Au lieu de lancer plusieurs instances du serveur d'applications, on peut faire monter en charge un micro-service donné à la demande. Quand la charge se déplace vers d'autres parties de l'application, un micro-service utilisé antérieurement subit une baisse de charge parallèlement à la montée en charge d'un autre. On valorise mieux l'infrastructure sous-jacente : inutile désormais de provisionner de nouvelles machines virtuelles, il suffit de provisionner de nouvelles instances de micro-services sur les machines existantes.

Les développeurs et les administrateurs pourront choisir les technologies de pointe qui fonctionnent le mieux avec tel ou tel micro-service. Ils pourront combiner divers systèmes d'exploitation, langages, frameworks, environnements d'exécution, bases de données et outils de contrôle.

Enfin, la technologie des micro-services permet aux entreprises d'investir dans des composants modulables et réutilisables. Chaque micro-service fonctionne comme une brique de Lego qui

peut être enfichée dans une pile d'applications. Si les entreprises investissent dans un jeu de micro-services élémentaires, elles pourront ensuite construire par assemblage des applications répondant à diverses utilisations. [24]

## 2.4 Micro-services et API

Les **micro-services** relèvent d'un type d'architecture de développement qui permet de déléguer des fonctionnalités à des systèmes tiers. Et les **API** sont les éléments qui permettent à ces micro-services de partager des informations entre eux. Elles servent de portes d'entrée par lesquelles les développeurs internes et externes peuvent accéder aux données d'un micro-service ou utiliser ses fonctionnalités. [25]

La figure 2-6 qui suit représente la quelques micro-services et API.

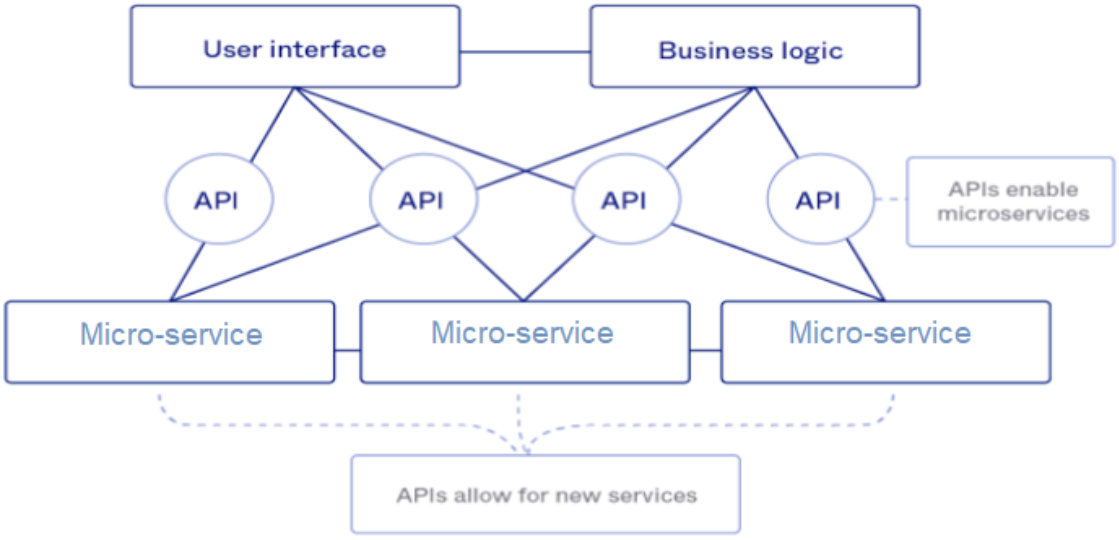


*Figure 2-6 : Micro-services et API*

Au moyen des API, l'architecture en micro-services donne aux développeurs l'opportunité de déléguer des fonctionnalités et d'enrichir leurs applications avec des services d'experts du secteur.

### 2.4.1 Utilisation d'API dans notre cas

Notre système global que nous envisageons est basé sur une architecture micro-services. Cette architecture sera composée de plusieurs micro-services qui se communiquent et se partagent mutuellement de données. Il est aussi possible de faire communiquer directement les clients et nos micro-services. Ce fonctionnement est préférable et plus adapté dans le cadre d'application

au domaine fonctionnel restreint mais dans notre cas la communication directe sera difficile à gérer. Comme notre architecture est évolutive, plus le nombre de micro-services mis à disposition sera important plus il sera compliqué de savoir quel micro-service devra être appelé. Pour optimiser donc le fonctionnement de notre système, il nous faudra se poser plusieurs questions à savoir : comment limiter le nombre de requêtes envoyées au backend ? Comment gérer des problèmes tels que la sécurité, l'utilisation de protocoles non compatibles internet ?

Pour gérer ces différentes problématiques on peut faire usage d'une passerelle d'API ou API Gateway « service-gateway » dans notre architecture.

#### 2.4.1.1 API gateway

Une API Gateway est une passerelle entre les API exposées par des micro-services (ou une application) et un front consommant ces API son rôle est de :

- Servir de point d'entrée au système.
- Filtrer les requêtes venant des autres applications.
- Réaliser la transformation de protocoles ou de données, l'utilisation de protocoles non compatibles internet.
- Centraliser l'implémentation de la sécurité des micro-services.
- Participer à la disponibilité et l'évolutivité de l'application.
- Contribuer à la simplicité de la consommation des API par les applications consommatrices.

Comme le montre la figure 2-7, l'API Gateway est le point d'entrée de notre système, le client ne doit savoir que son adresse et elle se charge de le router vers le micro-service qui répond à sa demande de façon transparente.

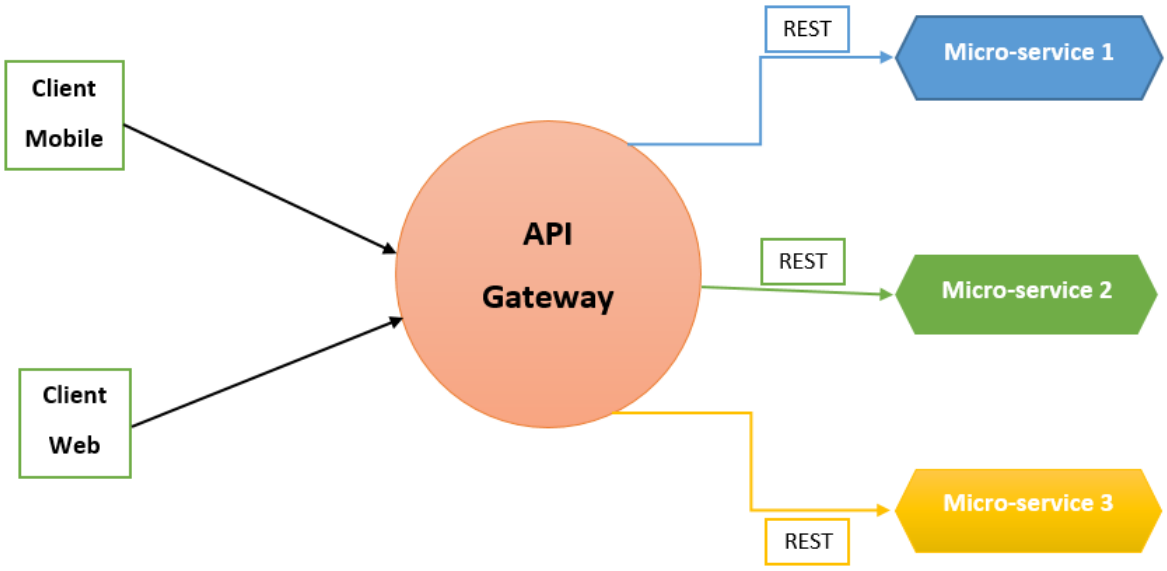


*Figure 2-7 : API gateway*

#### 2.4.1.2 Pourquoi avons-nous chosi l'API gateway plutôt qu'une communication directe de client à micro-service ?

Dans notre architecture, plus le nombre de micro-services est important plus il est compliqué de savoir lequel devra être appelé pour satisfaire la demande d'une application cliente. Et souvent une application cliente fait appelle à plusieurs micro-services pour collecter l'ensemble des données nécessaires dont elle a besoin.

Sans la passerelle d'API ou API gateway, Ces applications clientes envoient les requêtes directement aux micro-services ; et cela peut avoir les incidences suivantes :

- Les applications doivent obligatoirement connaître précisément tous les micro-services (Nom, adresse et numéro de port) de l'architecture pour pouvoir les appeler.
- Augmentation de la latence réseau avec la multiplication des allers-retours entre l'application cliente et les micro-services.
- Garantir la sécurité des données devient également plus compliqué puisque tous les services sont exposés publiquement ; ce qui augmente ainsi le risque d'attaque.

En outre, dans la mesure où l'API Gateway est implémentée tout comme les micro-services, il permet de présenter aux applications consommatrices uniquement les informations nécessaires, et aussi de gérer plus simplement le versioning des micro-services.

### 2.4.2 Méthode de mise en place d'un API gateway

L'API Gateway se doit d'être indépendante et se place entre le client et le backend. En tant que couche d'abstraction des micro-services, elle agit comme un proxy inversé en redistribuant les requêtes vers les différents micro-services correspondants. Elle devient donc un point d'entrée à l'application. Il est possible d'implémenter, dans cette couche, des fonctionnalités supplémentaires comme l'authentification, le cache ou encore la sécurité.

L'API Gateway se doit également d'être répliquée et « Load Balancée » pour éviter de devenir le « Single Point Of Failure » de l'application. Ainsi, elle ne doit pas non plus devenir monolithique et regrouper l'ensemble des appels vers les micro-services. Les API Gateway pourront être séparées par domaine fonctionnel et par utilisation, voire par plateforme (desktop, web, mobile). Cela permettra d'équilibrer la charge et d'avoir les justes données à fournir au client.

Les implémentations d'authentification ou de cache peuvent être déléguées à la Gateway évitant par la même occasion une implémentation propre à chaque micro-service ce qui améliore les délais d'implémentation en réduisant la charge, facilite l'écriture et la lecture du code des micro-services. Mais aussi en externalisant certaines portions, facilite l'évolutivité puisque ces implémentations sont « centralisées » donc plus faciles à faire évoluer et les micro-services s'abstenant de ces implémentations sont eux aussi plus faciles à maintenir et à faire évoluer. [26]

Nous avons à travers ce chapitre fait l'état de l'art des micro-services pour mieux cerner les concepts de base liés à une architecture micro-services et à notre projet en général. Le prochain chapitre abordera la spécification des besoins fonctionnels.

# CHAPITRE 3 : Spécification des besoins fonctionnels

Dans cette section, il est nécessaire de délimiter le contexte du sujet en faisant une étude fonctionnelle du futur système. Il s'agit donc d'identifier l'ensemble des acteurs agissant au système global et d'identifier les besoins fonctionnels.

## 3.1 Identification des acteurs

Un acteur est l'archétype de l'utilisateur (personne, processus externe, …) qui interagit avec le système. Par défaut, c'est un acteur principal, c'est-à-dire qu'il agit directement sur le système et en attend des résultats ou bien, il peut être un acteur secondaire qui est souvent sollicité pour des informations supplémentaires.

*Tableau 3-1: Tableau d'identification des acteurs*

| Acteurs | Rôles | Autorisation d'accès |
|---|---|---|
| **Recteur** | Supervise la gestion admirative | Il a accès à certaines fonctionnalités ou services concernant la partie de la gestion administrative et pédagogique |
| **Vice-recteur R & C** | Il est le responsable de la gestion de certaines tâches administratives sous la supervision du recteur | Il est autorisé à accéder sur les fonctionnalités ou services que le recteur lui a délégués. |
| **Vice-recteur EVU** | Il supervise la gestion pédagogique dont il est responsable et sous contrôle du recteur. | Il a accès à certaines fonctionnalités ou services concernant la gestion pédagogique |
| **PER** | - Coté administratif il fournit des informations personnelles au système pour bénéficier des autres services qui sont autour de la gestion des informations du personnel. | Il est autorisé à accéder à certaines fonctionnalités liées à la gestion |

| | | |
|---|---|---|
| | - Coté pédagogique il est responsable de certaines fonctionnalités liées à la gestion des examens, des devoirs, des notes etc. | administratives et pédagogique. |
| **Acteurs** | Rôles | Autorisation d'accès |
| **PATS** | - Coté administratif il donne des informations personnelles au système pour ensuite bénéficier des autres services qui sont autour de la gestion des informations du personnel.<br>- Coté pédagogique il est responsable de certaines tâches liées à la gestion pédagogique. | Il est autorisé à accéder à certaines fonctionnalités liées à la gestion pédagogique. |
| **Vacataire** | Renseigner des informations personnelles au niveau du système de gestion des heures de vacations. | Il est autorisé à accéder à certaines fonctionnalités liées à la gestion des heures de vacations. |
| **Directeur d'UFR** | Il supervise la gestion des tâches administratives et pédagogiques de son UFR. | Il est autorisé à accéder à certaines fonctionnalités liées à la gestion administrative et pédagogique au sein de son UFR. |
| **Chef de département** | Il supervise la gestion des taches pédagogique de son département. | Il est autorisé sur certaines fonctionnalités de la gestion pédagogique au sein de son département. |

| | | |
|---|---|---|
| **Responsable filière** | Il est le responsable de gestions taches pédagogique de la filière sous le contrôle du chef du département. | L'accès est autorisé sur certaines fonctionnalités de la gestion pédagogique. |

| Acteurs | Rôles | Autorisation d'accès |
|---|---|---|
| **DRH** | Il est le responsable de la gestion du recrutement du personnel, du traitement des salaires, des congés et des absences. | Il a accès sur certaines fonctionnalités liées à la gestion du recrutement du personnel, du traitement des salaires, des congés et des absences. |
| **Secrétaire** | Il assure certaines tâches administratives ou pédagogiques sous le contrôle de son supérieur hiérarchique. | Il est autorisé à toutes les fonctionnalités qui lui sont déléguées par son supérieur hiérarchique |
| **Comptable des matières de l'université ou d'UFR** | Il assure la gestion des biens matériels. | Il a accès aux fonctionnalités ou services concernant la gestion des biens matériels |
| **CDAP** | Il gère le suivi des congés et les agents en congé. | Il est autorisé à accéder à certaines fonctionnalités liées au suivi des congés du personnel. |
| **Etudiant** | Il fournit des informations personnelles au système afin de bénéficier des fonctionnalités liées à la gestion administrative et pédagogiques. | Il est autorisé pour accéder à certaines fonctionnalités lui concernant. |

| | | |
|---|---|---|
| **Administrateur** | Il gère la partie administration du système global. | Accès autorisé pour tous les services. |

## 3.2 Identification des besoins fonctionnels

Les fonctionnalités sont les actions ou services pouvant être rendus par le système en cas de sollicitation de l'acteur concerné. Ces services peuvent être externes (nécessitant un acteur) ou bien, interne, c'est-à-dire non-visible, mais fonctionnant en interne s'il y a le démarrage d'un processus donné. Dans notre étude, nous avons pu inclure l'ensemble des fonctionnalités dans les huit (8) sous domaines qui constituent le découpage de l'UASZ pour mener à bien sa gestion administrative, pédagogique et financière.

Le découpage en sous-domaines effectué est représenté au niveau de la figure 1-1 de notre premier chapitre (chapitre 1). Il nous permet d'avoir une vision claire sur l'architecture que nous envisageons afin de mieux cerner l'ensemble des besoins fonctionnels dans chaque sous domaines. Ces sous-domaines seront découpés en micro-services, et l'ensemble de ces micro-services constitueront l'architecture générale du système global.

Dans cette section nous allons premièrement faire le découpage de chaque sous-domaine en des micro-services et deuxièmement donner la description de chacun d'entre eux.

### 3.2.1 Découpage en micro-services de la gestion administrative

Il est constitué de plusieurs applications indépendantes et autonomes dont une qui est centrale et qui communiquent avec les autres en les exposant ses services qu'elles peuvent utiliser en cas de besoin. Ces applications sont des micro-services. Ces derniers partagent des données via des services qu'ils offrent mutuellement afin d'assurer la bonne gestion administrative à l'UASZ.

Ces micro-services constituent l'architecture de la gestion administrative mais aussi mettent en évidence l'ensemble des acteurs qui interviennent dans ces systèmes via un portail web.

La figure 3-2 ci-dessous montre en détails cette architecture.

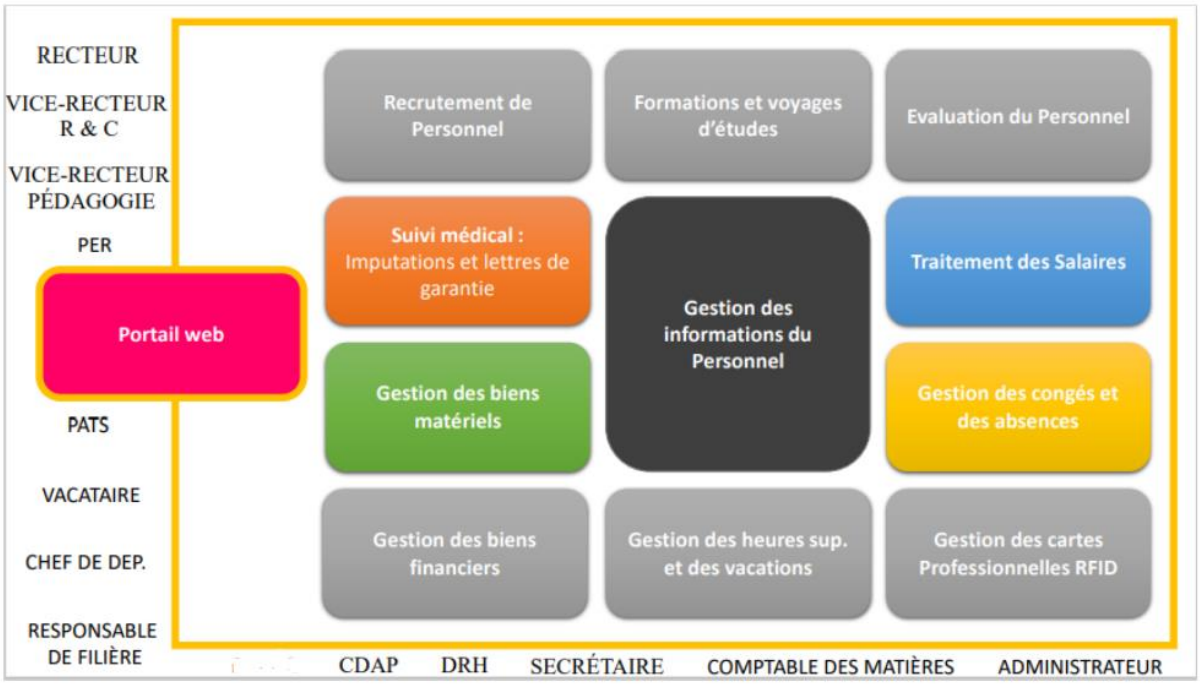


*Figure 3-1 : Architecture pour la gestion administrative de l'UASZ.*

- Description des micro-services de l'architecture

*Tableau 3-2 : Gestion des informations du personnel*

| **Nom du micro-service** | **Gestion des informations du personnel** |
|---|---|
| **Résumé** | Centraliser et mettre à jour les informations sur le personnel pour les fournir aux différents modules de gestion d'informations de l'UASZ. |
| **Fonctionnalités** | • Recueil et mise à jour des informations sur le personnel avec des dépôts de documents justificatifs en ligne.<br>• Validation des documents déposés en ligne par la DRH avant sa prise en charge dans le dossier du personnel et pour tout ce qui suit. |
| **Liens entrants (←)** | • « Recrutement de personnel »<br>• « Evaluation du Personnel » |
| **Liens sortants (→)** | • « Gestion des congés et des absences »<br>• « Traitement des salaires »<br>• « Gestion des biens matériels »<br>• « Suivi médical : Imputations et lettres de garanties » |

| | |
|---|---|
| | • « Formations et voyages d'études »<br>• « Gestion des cartes professionnelles RFID »<br>• « Gestion des biens financiers »<br>• « Gestion des heures sup et des vacations » |

*Tableau 3-3 : Gestion des congés et des absences*

| **Nom du micro-service** | **Gestion des congés et des absences** |
|---|---|
| **Résumé** | L'autonomisation de la gestion des congés et des absences au sein de la DRH de l'UASZ et la dématérialisation des documents administratifs qui s'y rapportent. |
| **Fonctionnalités** | • Ouverture et fermeture de sessions congés.<br>• Automatisation des du calcul des jours de congés selon du personnel et les jours restants dans les sessions précédentes.<br>• Dématérialisation des documents administratifs liés aux congés.<br>• Envoie des plannings de congés aux personnels par mail.<br>• Automatisation des procédures de la gestion des congés et absences.<br>• Historique de congés du personnel.<br>• Statistiques.<br>• Validation des documents déposés en ligne par la DRH avant sa prise en charge dans le dossier du personnel et pour tout ce qui suit. |
| **Lien entrant (←)** | • « Gestion des dossiers du personnel » |
| **Lien sortant (→)** | • « Traitement des salaires » |

*Tableau 3-4 : Traitement des salaires*

| **Nom du micro-service** | **Traitement des salaires** |
|---|---|
| **Résumé** | Automatisation de la procédure de gestion des salaires et dématérialisation du bulletin de salaire (ou bulletin de paye) à l'UASZ. |

| | |
|---|---|
| **Fonctionnalités** | • Automatisation du calcul des salaires.<br>• Dématérialisation du bulletin de salaire.<br>• Envoie des bulletins de salaire aux personnels par email avec notification SMS chaque mois.<br>• Historique des bulletins de salaire du personnel.<br>• Statistiques |
| **Liens entrants (←)** | • « Gestion des congés et des absences »<br>• « Gestion des biens financiers »<br>• « Gestion des dossiers du personnel » |
| **Liens sortants (→)** | Néant |

*Tableau 3-5 : Gestion des biens financiers*

| **Nom du micro-service** | **Gestion des biens financiers** |
|---|---|
| **Résumé** | L'automatisation de la gestion des biens matériels en s'appuyant sur l'utilisation des codes à barres pour l'identification de chaque matériel et l'automation de la saisie des informations. |
| **Fonctionnalités** | • Ouverture et fermeture d'une session budgétaire<br>• Proposition du budget<br>• Amendement et gestion de l'évolution (versionning) de la proposition de budget.<br>• Vote du budget.<br>• Gestion du déroulement du budget (par ses lignes)<br>• Rapport fin budget<br>• Historiques<br>• Statistiques |
| **Lien entrant (←)** | • « Gestion des dossiers du personnel » |
| **Liens sortants (→)** | Néant |

*Tableau 3-6 : Gestion des biens matériels*

| Nom du micro-service | Gestion des biens matériels |
|---|---|
| **Résumé** | L'automatisation de la gestion des biens financiers à la gestion du budget. |
| **Fonctionnalités** | • Faciliter la saisie par lecture code à barre.<br>• Gérer l'entrer ou la réception de biens.<br>• Gérer les sorties de biens.<br>• Gérer l'attribution ou affectation de biens aux matériels.<br>• Faire un suivi et une maintenance des biens matériels. |
| **Lien entrant (←)** | • « Gestion des dossiers du personnel » |
| **Liens sortants (→)** | • « Gestion des dossiers du personnel »<br>• « Gestion des biens financiers » |

*Tableau 3-7 : Suivi médical : Imputations et lettres de garantie*

| Nom du micro-service | Suivi médical : Imputations et lettres de garantie |
|---|---|
| **Résumé** | Automatisation du suivi médical avec possibilité de demande en ligne des imputations et lettre de garantie. |
| **Fonctionnalités** | • Production d'imputations et lettres de garantie.<br>• Demande en ligne d'imputations et lettre de garantie.<br>• Historiques des demandes d'imputations et lettres de garantie. |
| **Lien entrant (←)** | • « Gestion des dossiers du personnel » |
| **Liens sortants (→)** | Néant |

*Tableau 3-8 : Gestion des heures sup. et des vacations*

| Nom du micro-service | Gestion des heures sup. et des vacations |
|---|---|
| **Résumé** | L'automatisation de la gestion du payement des heures sup et des heures de vacations. |
| **Fonctionnalités** | • Gérer les heures sup et les de vacations. |

| | |
|---|---|
| | • Faciliter les paiements avec les transferts d'argent.<br>• Historique des paiements.<br>• Statistiques. |
| **Liens entrants (🡐)** | • « Gestion des dossiers du personnel »<br>• « Gestion des biens financiers » |
| **Liens sortants (🡒)** | Néant |

*Tableau 3-9 : Gestion des formations*

| **Nom du micro-service** | **Gestion des formations** |
|---|---|
| **Résumé** | L'automatisation de la gestion des formations à l'UASZ. |
| **Fonctionnalités** | • Demande de formations.<br>• Publication des formations. |
| **Liens entrants (🡐)** | • « Gestion des dossiers du personnel »<br>• « Gestion des biens financiers » |
| **Liens sortants (🡒)** | Néant |

*Tableau 3-10 : Voyages d'études*

| **Nom du micro-service** | **Voyages d'études** |
|---|---|
| **Résumé** | L'autonomisation de la gestion des voyages d'études |
| **Fonctionnalités** | • Publication des listes des VE chaque année.<br>• Remplissage du formulaire de demande VE.<br>• Tableau de bord sur le déroulement des VE.<br>• Remplissage et Edition rapport. |
| **Liens entrants (🡐)** | • « Gestion des dossiers du personnel »<br>• « Gestion des biens financiers » |
| **Liens sortants (🡒)** | Néant |

*Tableau 3-11 : Evaluation du personnel*

| **Nom du micro-service** | **Evaluation du personnel** |
|---|---|
| **Résumé** | L'autonomisation de la gestion de l'évaluation du personnel |
| **Fonctionnalités** | • Gestion des grilles d'évaluation.<br>• Gérer les heures de pointage. |

| | |
|---|---|
| **Liens entrants (←)** | • « Gestion des dossiers du personnel »<br>• « Gestion des biens financiers » |
| **Liens sortants (→)** | Néant |

*Tableau 3-12 : Recrutement du Personnel*

| **Nom du micro-service** | **Recrutement du Personnel** |
|---|---|
| **Résumé** | L'autonomisation de la gestion du recrutement du personnel de l'UASZ. |
| **Fonctionnalités** | • Gestion des grilles de recrutement.<br>• Publication d'avis de recrutement.<br>• Dépôt de dossier.<br>• Classement des dossiers pour un poste selon une grille de recrutement. |
| **Liens entrants (←)** | • « Gestion des dossiers du personnel »<br>• « Gestion des biens financiers » |
| **Liens sortants (→)** | Néant |

### 3.2.2 Découpage en micro-services de la gestion pédagogique

Il est constitué de plusieurs applications indépendantes et autonomes dont une qui est centrale et qui communiquent avec les autres en exposant ses services qu'elles peuvent utiliser en cas de besoin. Ces applications sont des micro-services qui partagent des données via des services qu'elles offrent mutuellement afin d'assurer la bonne gestion pédagogique à l'UASZ.

Ces micro-services constituent l'architecture de la gestion pédagogique mais aussi mettent en évidence l'ensemble des acteurs qui interviennent dans ces systèmes via un portail web.

La figure 3-2 ci-dessous montre en détails cette architecture.

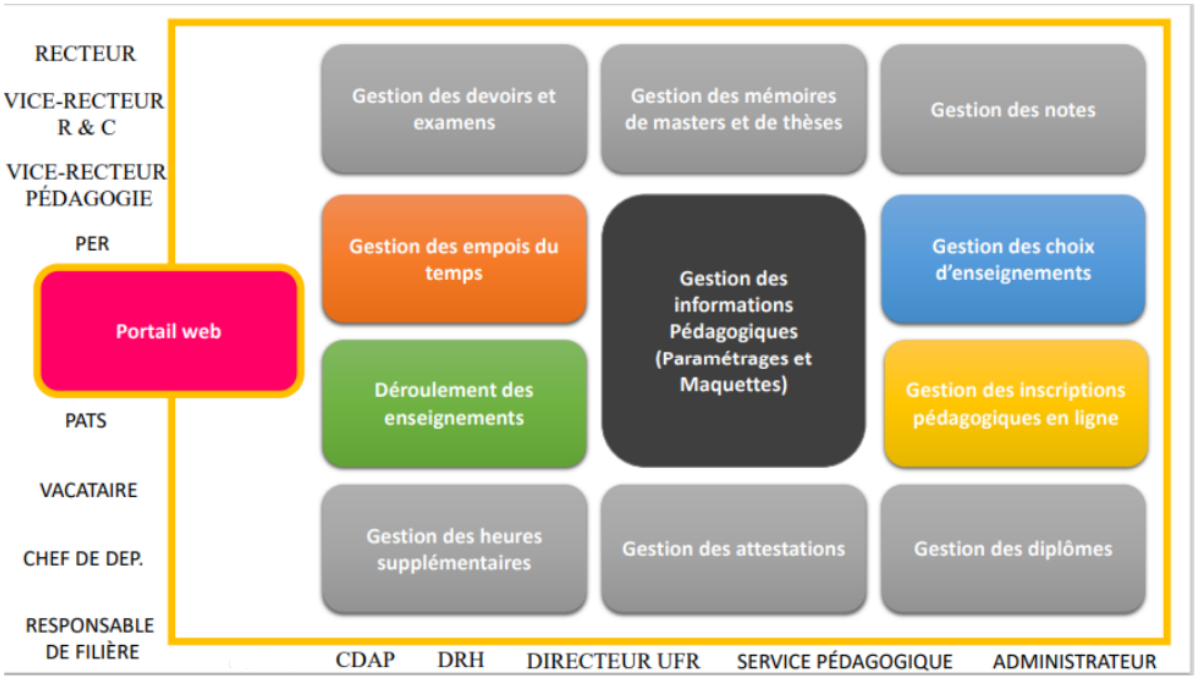


*Figure 3-2 : Architecture pour la gestion pédagogique de l'UASZ.*

- Description des micro-services de l'architecture

*Tableau 3-13 : Gestion des informations pédagogiques*

| **Nom du micro-service** | **Gestion des informations pédagogiques** |
|---|---|
| **Résumé** | Centraliser et mettre à jour les informations pédagogiques pour les fournir aux différents modules de gestions d'informations pédagogiques de l'UASZ. |
| **Fonctionnalités** | • Gérer les maquettes des filières<br>• Gérer les différents paramètres pédagogiques. |
| **Liens entrants (←)** | Néant |
| **Liens sortants (→)** | • « Gestion des devoirs et examens »<br>• « Gestion des mémoires de master et de thèses »<br>• « Gestion des notes »<br>• « Gestion des emplois du temps »<br>• « Déroulement des enseignements »<br>• « Gestion des heures supplémentaires »<br>• « Gestion des attestations »<br>• « Gestion des diplômes »<br>• « Gestion des inscriptions pédagogiques » |

| | |
|---|---|
| | • « Gestion des choix d'enseignements » |

*Tableau 3-14 : Gestion des inscriptions pédagogiques en ligne*

| **Nom du micro-service** | **Gestion des inscriptions pédagogiques en ligne** |
|---|---|
| **Résumé** | L'autonomisation du choix des enseignements à l'UASZ. |
| **Fonctionnalités** | • Ouverture et fermeture d'une année académique.<br>• Liste des inscriptions académiques par formation.<br>• Liste des choix (UE et EC) possibles<br>• Choix des UE et EC.<br>• Vérification des choix des UE et EC.<br>• Inscription pédagogiques et impression des choix de l'étudiant.<br>• Sauvegarde des informations de l'inscription pédagogique.<br>• Edition des listes des promotions et leurs groupes par UE ou EC<br>• Historisation des informations.<br>• |
| **Liens entrants (←)** | • « Gestion des information pédagogiques »<br>• « Gestion des inscriptions académiques » |
| **Liens sortants (→)** | • « Gestion des devoirs et examens »<br>• « Gestion des mémoires de master et de thèses »<br>• « Gestion des notes » |

| **Nom du micro-service** | **Gestion des devoirs et examens** |
|---|---|
| **Résumé** | Automatisation de la gestion des devoirs et examens des étudiants à l'UASZ. |
| **Fonctionnalités** | • Planifier une évaluation.<br>• Gérer les évaluations des étudiants.<br>• Obtenir la liste d'émargement des étudiants.<br>• Surveiller une évaluation à distance.<br>• Remplir les notes d'évaluation. |
| **Liens entrants (←)** | • « Gestion des informations pédagogiques » |

| | |
|---|---|
| | • « Gestion des inscriptions pédagogiques »<br>• « Gestion des notes » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-15 : Gestion des mémoires de masters et de thèses*

| **Nom du micro-service** | **Gestion des mémoires de masters et de thèses** |
|---|---|
| **Résumé** | Automatisation de la gestion des mémoires de master et de thèses. |
| **Fonctionnalités** | • Programmation de soutenance de master.<br>• Programmation de soutenance de thèses.<br>• Edition des rapports et procès-verbaux des soutenances |
| **Liens entrants (←)** | • « Gestion des informations pédagogiques »<br>• « Gestion des inscriptions pédagogiques » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-16 : Gestion des notes*

| **Nom du micro-service** | **Gestion des notes** |
|---|---|
| **Résumé** | Automatisation de la gestion des notes d'étudiants. |
| **Fonctionnalités** | • Gérer les notes d'étudiants. |
| **Liens entrants (←)** | • « Gestion des informations pédagogiques »<br>• « Gestion des inscriptions pédagogiques » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-17 : Gestion des emplois du temps*

| **Nom du micro-service** | **Gestion des emplois du temps** |
|---|---|
| **Résumé** | Automatisation de la gestion des emplois du temps. |
| **Fonctionnalités** | • Gestion des emplois du temps |
| **Lien entrant (←)** | • « Gestion des informations pédagogiques » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-18 : Déroulement des enseignements*

| Nom du micro-service | Déroulement des enseignements |
|---|---|
| **Résumé** | Automatisation du déroulement des enseignements à l'UASZ. |
| **Fonctionnalités** | • Gestion du déroulement des enseignements.<br>• Consulter la réparation des enseignements selon le calendrier |
| **Liens entrants (←)** | • « Gestion des informations pédagogiques »<br>• « Gestion des inscriptions pédagogiques » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-19 : Gestion des heures sup. et des vacations*

| Nom du micro-service | Gestion des heures sup. et des vacations |
|---|---|
| **Résumé** | Automatisation de la gestion des heures supplémentaires et des vacations. |
| **Fonctionnalités** | • Gérer les heures supplémentaires.<br>• Gérer les vacations. |
| **Lien entrant (←)** | • « Gestion des informations pédagogiques » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-20 : Gestion des attestations*

| Nom du micro-service | Gestion des attestations |
|---|---|
| **Résumé** | Automatisation de la gestion des attestations de réussite des étudiants. |
| **Fonctionnalités** | • Gérer les attestations de réussite.<br>• Vérifier disponibilité.<br>• Signaler erreurs attestation.<br>• Télécharger attestation. |
| **Liens entrants (←)** | • « Gestion des informations pédagogiques »<br>• « Gestion des inscriptions pédagogiques » |
| **Liens sortants (→)** | • Néant |

*Tableau 3-21 : Gestion des choix d'enseignement*

| Nom du micro-service | Gestion des choix d'enseignement |
|---|---|

| **Résumé** | Automatisation de la gestion des choix d'enseignements à l'UASZ. |
|---|---|
| **Fonctionnalités** | • Ouverture et Fermeture de la réparation des enseignements.<br>• Gestion des enseignements.<br>• Choisir des enseignements.<br>• Calculer les heures d'enseignement.<br>• Signaler et régler les conflits de choix. |
| **Lien entrant (←)** | • « Gestion des informations pédagogiques » |
| **Liens sortants (→)** | Néant |

*Tableau 3-22 : Gestion des diplômes*

| **Nom du micro-service** | **Gestion des diplômes** |
|---|---|
| **Résumé** | Automatisation de la gestion des diplômes à l'UASZ. |
| **Fonctionnalités** | • Consulter les types de diplômes.<br>• Vérifier disponibilité d'un diplôme.<br>• Demander retrait d'un diplôme.<br>• Télécharger diplôme. |
| **Liens entrants (←)** | • « Gestion des informations pédagogiques » |
| **Liens sortants (→)** | Néant |

### 3.2.3 Découpage en micro-services de la gestion des composants et du réseau informatique

Il est constitué de plusieurs applications indépendantes et autonomes dont une qui est centrale et qui communiquent avec les autres en exposant ses services qu'elles peuvent utiliser en cas de besoin. Ces applications sont des micro-services. Ces derniers partagent des données via des services qu'ils offrent mutuellement afin d'assurer la bonne gestion des composants et du réseau informatique à l'UASZ.

Ces micro-services constituent l'architecture de la gestion des composants et du réseau informatique mais aussi mettent en évidence l'ensemble des acteurs qui interviennent dans ces systèmes via un portail web.

La figure 3-3 ci-dessous montre en détails cette architecture.

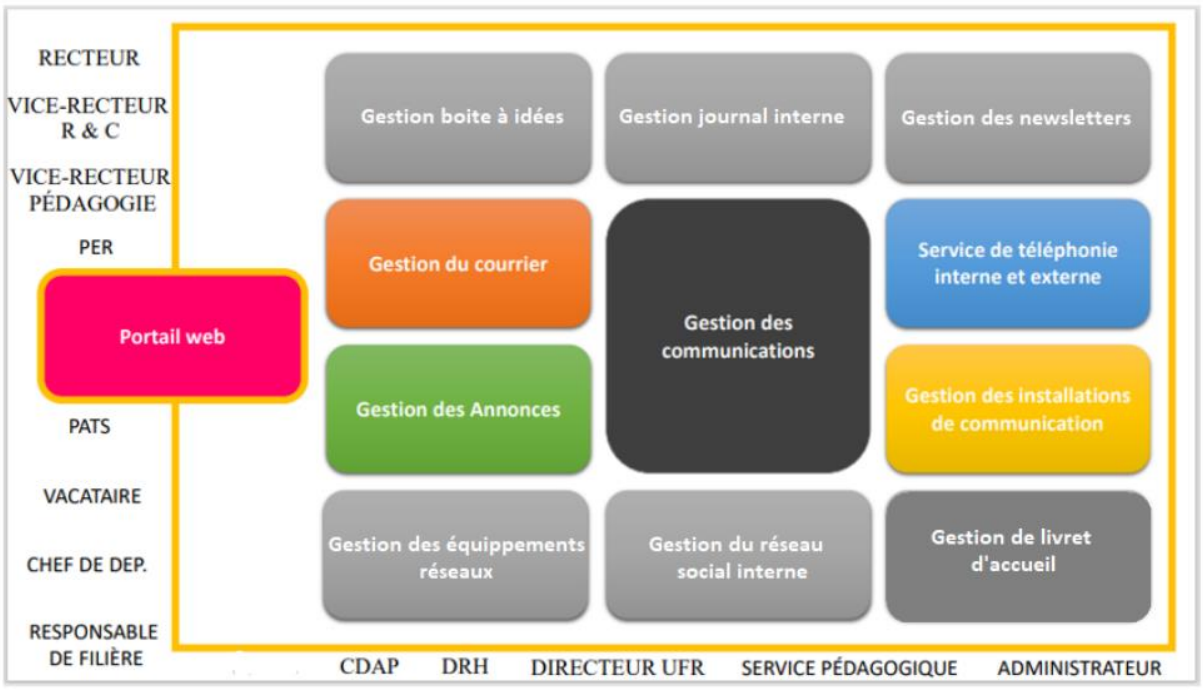


*Figure 3-3 : Architecture pour la gestion des composants et l'architecture du réseau informatique de l'UASZ.*

- Description des micro-services de l'architecture

*Tableau 3-23 : Gestion des communications*

| **Nom du micro-service** | **Gestion des communications** |
|---|---|
| **Résumé** | L'automation de la gestion des communications à l'UASZ. |
| **Fonctionnalités** | • Gérer les communications internes.<br>• Gérer les informations internes.<br>• Gérer les accès aux outils.<br>• Diagnostiquer et gérer les problèmes techniques à distance.<br>• Sécuriser les communications. |
| **Liens entrants (←)** | Néant |
| **Liens sortants (→)** | • « Gestion des communications »<br>• « Gestion des courriers »<br>• « Gestion des annonces »<br>• « Services de téléphonie interne et externe »<br>• « Gestion des installations de communication »<br>• « Gestion boite à idées »<br>• « Gestion journal interne » |

*Tableau 3-24 : Gestion des courriers*

| Nom du micro-service | Gestion des courriers |
|---|---|
| **Résumé** | L'automation de la gestion administrative du courrier inclut l'ensemble des tâches et processus de son envoie et de sa réception et de son traitement. |
| **Fonctionnalités** | • Faciliter les envois et réceptions de courriers.<br>• Gérer le traitement des courriers.<br>• Gérer la traçabilité des courriers.<br>• Historisation des courriers. |
| **Liens entrants (←)** | Néant |
| **Lien sortant (→)** | • « Gestion des communications » |

*Tableau 3-25 : Gestion des annonces*

| Nom du micro-service | Gestion des annonces |
|---|---|
| **Résumé** | Système permettant la gestion des annonces à l'UASZ. |
| **Fonctionnalités** | • Gérer les annonces.<br>• Publier une annonce |
| **Liens entrants (←)** | Néant |
| **Lien sortant (→)** | • « Gestion des communications » |

*Tableau 3-26 : Services de téléphonie interne et externe*

| Nom du micro-service | Services de téléphonie interne et externe |
|---|---|
| **Résumé** | Système de gestion de la téléphonie interne et externe pour gérer les communications téléphoniques à l'UASZ. |
| **Fonctionnalités** | • Gérer la téléphonie interne et externe.<br>• Définir les horaires d'ouverture.<br>• Gérer les groupes d'appels.<br>• Gérer les suivis d'appels. |
| **Liens entrants (←)** | Néant |
| **Lien sortant (→)** | • « Gestion des communications » |

*Tableau 3-27 : Gestion du réseau social interne*

| Nom du micro-service | Gestion du réseau social interne |
|---|---|
| **Résumé** | Mise en place d'un réseau social interne qui s'inscrit dans la continuité de l'intranet permettant au personnel de disposer d'un profil. C'est un moyen supplémentaire d'échanger de manière instantanée. |
| **Fonctionnalités** | • Gérer les conversations<br>• Gérer les communications instantanées.<br>• Définir des groupes par services<br>• Etablir des restrictions. |
| **Liens entrants (←)** | Néant |
| **Lien sortant (→)** | • « Gestion des communications » |

*Tableau 3-28: Gestion de livret d'accueil*

| Nom du micro-service | Gestion de livret d'accueil |
|---|---|
| **Résumé** | Système de gestion d'un livret d'accueil pour une intégration réussie du personnel nouvellement recruté. |
| **Fonctionnalités** | • Consulter l'historique de l'université.<br>• Consulter l'organigramme de l'université.<br>• Consulter le règlement intérieur du site.<br>• Accéder à toutes les informations facilitant l'intégration du nouvel ou de la nouvelle recru(e). |
| **Liens entrants (←)** | Néant |
| **Lien sortant (→)** | • « Gestion des communications » |

Ce chapitre faisait l'objet de l'analyse des besoins fonctionnels. Dans un premier lieu, nous avons fait l'identification des acteurs. Dans un second lieu, nous avons détaillé la spécification des besoins fonctionnels. Le chapitre qui suit sera donc consacré à la conception et à l'implémentation des micro-services.

# CHAPITRE 4 : Conception et implémentation des micro-services

Après avoir détaillé la section spécification des besoins, il sera question d'expliquer les grands points de la conception et l'implémentation de nos différents micro-services. L'implémentation va s'appuyer intégralement sur les résultats découlant de la section conception. La réalisation de ces micro-services nécessite un choix minutieux de technologies et de langages à utiliser. Dans la première section nous allons parler de la conception générale et dans la deuxième section nous allons détailler l'implémentation en présentant l'ensemble des outils et technologies utilisés.

## 4.1 Conception générale

La phase de conception est une phase essentielle, elle nécessite une analyse rigoureuse et une proposition de modèle adapté pour la réalisation du système. Notre conception reposera sur deux points essentiels à savoir l'architecture générale pour l'UASZ et l'architecture micro-services technique proposée pour l'implémentation des prototypes.

### 4.1.1 Architecture micro-services générale pour le cas de l'UASZ

L'objectif général de cette architecture est de permettre aux équipes de développement, de réaliser un système global, sécurisé et centré sur l'organisation et le personnel. Elle est composée de plusieurs sous-systèmes qui découlent des architectures citées dans notre chapitre précédent. Avec une telle architecture, nous pourront fournir rapidement des logiciels de qualité pour l'université. Il est possible grâce à cette architecture de développer plusieurs micro-services simultanément. De plus, puisque les services qu'ils partagent sont déployés indépendamment, nous n'avons pas à recréer ou redéployer toute l'application après chaque modification.

La figure 4-1 suivante nous en donne plus en détail.

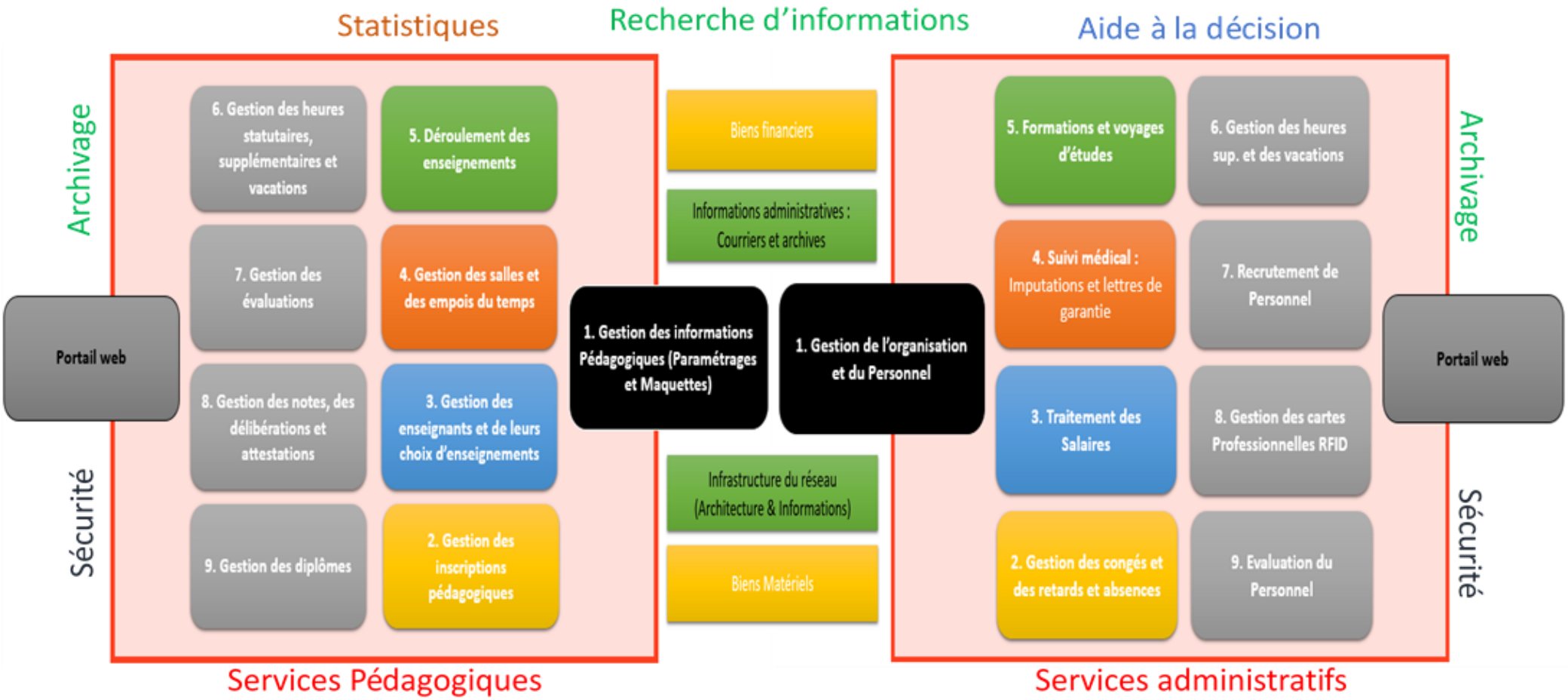


*Figure 4-1 : Architecture micro-services générale du système pour la dématérialisation des documents et procédures pédagogiques et administratives [15]*

- Description de quelques paramètres de l'architecture

*Tableau 4-1: Description de quelques paramètres de l'architecture*

| Paramètres | Description |
|---|---|
| **Archivage** | Comme les archives physiques, les archives électroniques répondent aux mêmes règles tout en se distinguant par des particularités spécifiques, nécessitant une gestion d'archive particulière. |
| **Aide à la décision** | Le système doit nous garantir l'ensemble des techniques qui permettent en cas de besoin d'effectuer la meilleure prise de décision possible. |
| **Recherche d'informations** | Elle étudie la meilleure manière de retrouver des informations dans le système. Celui-ci est composé de documents d'une ou plusieurs bases de données, qui sont décrits par un contenu associé. |
| **Sécurité** | Elle est indispensable pour la protection de l'intégrité du système contre les attaques, les dommages ou les accès non autorisés. |
| **Statistiques** | Elles permettent la collecte de données, leur traitement, leur analyse, l'interprétation des résultats afin de rendre ces données compréhensibles par tous. |

Cette architecture met en évidence le découpage clair en sous-domaine de la gestion administrative et pédagogique de l'université Assane SECK de Ziguinchor. Afin de réussir la réalisation d'un système global et sécurisé, chaque sous-système sera considéré comme une application à part autonome qui gère ses propres données. Ses sous-domaines représenteront

des micro-services qui se communiquent sur des API en partageant des informations via des services qu'ils exposent.

Les micro-services sont distribués et faiblement couplés, ils n'ont donc pas d'incidence les uns sur les autres. Ces caractéristiques offrent des avantages au niveau de l'évolutivité dynamique et de la tolérance aux pannes dans le système souhaité.

### 4.1.2 Architecture micro-services technique pour l'implémentation

En se référant de figure 4-1 précédente, nous voyons une décomposition nette de la gestion administrative et pédagogique de l'UASZ en plusieurs sous-domaines. Cette décomposition nous permettra de bien structurer nos besoins et nous facilitera l'adoption de l'approche micro-service afin de réaliser dans l'avenir un système global et sécurisé. Ce système sera centré non seulement sur l'organisation et le personnel mais aussi composé de plusieurs micro-services qui communiquent entre eux pour échanger des informations. Dans ce contexte, nous devons donc mettre en place une architecture micro-services technique en procédant comme suit :

- définir l'API Gateway : le point d'entrée unique pour l'ensemble de toute application basée sur les micro-services, présentant l'API de chaque micro-service ;
- proposer une approche pour la découverte de services : "Service Discovery" qui garde le catalogue des instances disponibles de chaque micro-service. Il dispose de ses propres mécanismes pour avoir le catalogue à jour à tout moment ;
- définir la communication inter-processus dans l'architecture micro-services en utilisant l'API REST ;
- gérer les données : chaque micro-service possède son propre magasin de données.

D'après la figure 4-2 toujours, il existe de nombreux micro-services à mettre en place suivant l'architecture. Et pour que la solution ou le système global à réaliser fonctionne de façon correcte, les micro-services doivent communiquer et interagir les uns avec les autres.

La figure 4-2 ci-dessous représente l'architecture pour l'implémentation et la communication des micro-services tirés de l'architecture générale :

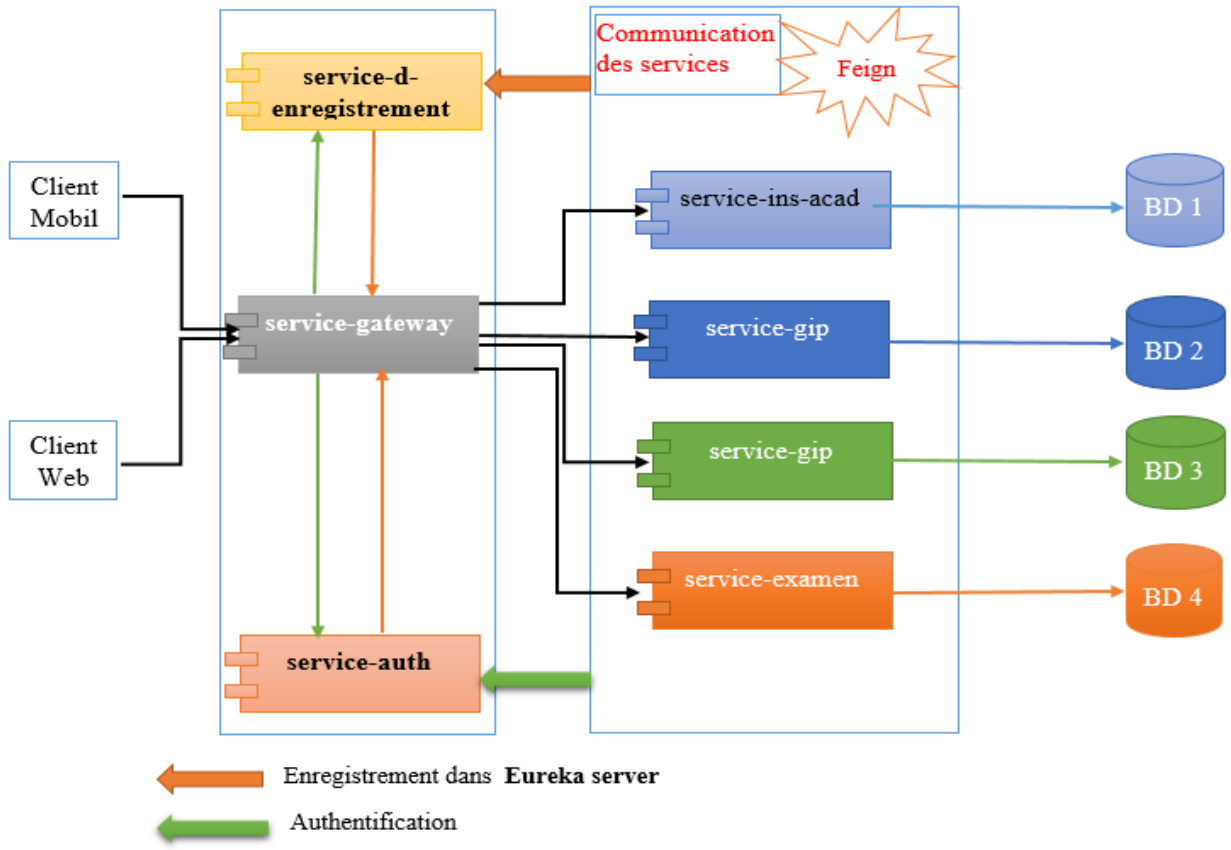


*Figure 4-2 : l'architecture micro-services technique pour l'implémentation*

Comme cette architecture est une architecture évolutive, nous pouvons y mettre en cas de besoin plusieurs composants. Et comme nous le montre l'architecture générale représentée par la figure 4-1, l'UASZ a besoin d'une telle architecture pour l'ensemble de ses systèmes existants ou à venir pour bien mener à son projet de dématérialisation des procédures et documents administratives.

Dans cette architecture proposée, nous avons juste représenté quelques composants que nous allons utiliser dans la suite pour les implémenter en tant que prototypes. Cela nous permettra de bien savoir l'intérêt de l'approche micro-services et de consolider notre travail.

La table 4-1 nous présente la description des composants que constitue cette architecture.

*Table 4-1 : Description des micro-services*

| Composants | Description |
|---|---|
| service-info-ped | Micro-service pour la gestion des informations pédagogiques (UE, EC etc.). |
| service-ins-acad | Micro-service pour la gestion des inscriptions académiques. |
| service-gip | Micro-service ayant la responsabilité de la gestion des inscriptions pédagogiques. |
| service-examen | Micro-service ayant la responsabilité de la gestion des examens. |
| service-d-enregistrement | Micro-service pour gérer les différentes instances des micro-services du système. Chaque micro-service s'enregistre auprès de ce micro-service. |
| service-gateway | Micro-service pour filtrer et répartir les différents appels à l'API vers les autres micro-services. |
| service-auth | Micro-service exposant les opérations d'authentification et de sécurité au sein de l'API du système. Par exemple, la connexion et la déconnexion d'un utilisateur se fera via ce micro-service. |

## 4.2 Implémentation

Après avoir choisi l'architecture micro-services à adopter. Nous devons comprendre l'interaction des services avant d'essayer d'optimiser leur implémentation. Ensuite, même si les architectures micro-services nous apportent une certaine rapidité, nous devons optimiser ces gains de vitesse en permanence. Il faut donc que nous fassions preuve de flexibilité en choisissant les technologies adaptées pour déployer notre architecture.

### 4.2.1 Langage utilisée

Ce qui est formidable avec les micro-services, c'est que les développeurs ne sont pas obligés d'utiliser toujours les mêmes langages ou les mêmes frameworks. Les composants peuvent être donc créés avec Ruby, Java, JavaScript/Node js ou PHP. Les programmeurs peuvent continuer d'utiliser les langages et les outils de développement qu'ils maîtrisent, qu'ils aiment et auxquels ils font confiance. Pour cela nous avons décidé de développer nos micro-services avec la technologie Java/JEE. Par ailleurs, il existe trois frameworks populaires pour développer une application en micro-services : Dropwizard, Spring Boot et WildFly Swarm..

Pour l'implémentation de nos micro-services, nous avons choisi le langage Java et les technologies offertes par Spring boot et Spring Cloud.

### 4.2.2 Technologies utilisées

#### 4.2.2.1 Mysql

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multithread et multi-utilisateur.

#### 4.2.2.2 Apache Tomecate

Apache Tomcat, souvent appelé serveur Tomcat, est un conteneur de servlets Java open-source développé par Apache Software Foundation (ASF). Tomcat implémente plusieurs spécifications Java EE, notamment Java Servlet, JavaServer Pages (JSP), Java EL et WebSocket, et fournit un environnement de serveur Web HTTP « pur Java » dans lequel le code Java peut s'exécuter. Tomcat est un serveur HTTP à part entière. De plus, il gère les servlets et les JSP (par un compilateur Jasper compilant les pages JSP pour en faire des servlets). Tomcat a été écrit en langage Java. Il peut donc s'exécuter via la machine virtuelle Java sur n'importe quel système d'exploitation la supportant.

#### 4.2.2.3 Spring framework

Spring est conçu comme une sorte de boîte à outils, au contraire d'autres Framework. Les modules peuvent être utilisés de façon indépendante. Spring est d'ailleurs disponible sous deux formes, celle d'un jar (librairie Java) unique et celle de plusieurs fichiers jar permettant de ne rajouter au projet que la partie que l'on souhaite utiliser (i.e. Spring Core, Spring Remoting, etc.). D'autre part, Spring fournit non seulement des services de type fonctionnel comme par

exemple les transactions mais est également utile d'un point de vue conceptuel en améliorant la qualité du design. Cela par l'utilisation quasi-systématique d'interface.

#### 4.2.2.4 Spring boot

Spring Boot est un projet ou un micro Framework qui a notamment pour but de faciliter la configuration d'un projet Spring et de réduire le temps alloué au démarrage d'un projet. Pour arriver à remplir cet objectif, Spring Boot se base sur plusieurs éléments :

Un site web (https://start.spring.io/) qui vous permet de générer rapidement la structure de votre projet en y incluant toutes les dépendances Maven nécessaires à votre application. Cette génération est aussi disponible via le plugin Eclipse STS.

#### 4.2.2.5 Spring cloud

Aujourd'hui, Spring Cloud & Netflix ont fait un grand bond en avant sur la mise en place et le déploiement des architectures de micro-services. Apportant ainsi pas mal de solutions sur tout ce dont on a besoin pour exposer et faire communiquer des micro-services de la manière la plus efficace et sécurisée possible et de plus avec une parfaite compatibilité avec les outils cloud native.

Dans cette section, il s'agira de voir comment est-ce qu'on peut déployer notre architecture de micro-services dans un écosystème Spring Cloud & Netflix. Pour ce faire, nous allons montrer étape par étape, comment nous avons implémenté nos micro-services avec Spring boot et Spring Cloud.

##### a) Serveur Discovery

Le serveur de discovery appelé dans notre cas « **service-d-enregistrement** » (voir figure 4-2) est un service où tous les micro-services de l'écosystème s'enregistrent en renseignant leurs informations d'identification et de routage : nom, adresse IP et numéro de port. Dans spring cloud, les deux services de discovery les plus populaires sont **Eureka** et **Consul**. Et dans notre cas nous avons choisi Eureka.

##### b) Client Discovery

Le client discovery est une dépendance à importer et à configurer au niveau de chaque micro-service. Ce composant se chargera à notre place de communiquer les micro-services avec le serveur de discovery. Assurant ainsi en background la découverte de service et le load Balancing à chaque fois que l'on veut contacter un micro-service avec RestTemplate ou un client Feign. (Voir figure 4-2).

### 4.2.3 Réalisation des micro-services avec Spring Boot

Puisque chacun de nos micro-services est une application à part, Spring boot nous permet de créer facilement et rapidement un projet puis composer notre application selon nos besoins. Pour créer un projet avec Spring boot, nous allons sur le site https://start.spring.io/, dans la partie **project Metadata** on renseigne les champs comme le nom du package, le nom de notre micro-service et toutes les autres informations nécessaires. Dans la partie **Dependancies,** on chosit les dépendances que nous avons besoin pour le micro-service.

La figure ci-dessous illustre les détails du projet de notre premier micro-service « **service-gip** » qui gère les inscriptions pédagogiques.

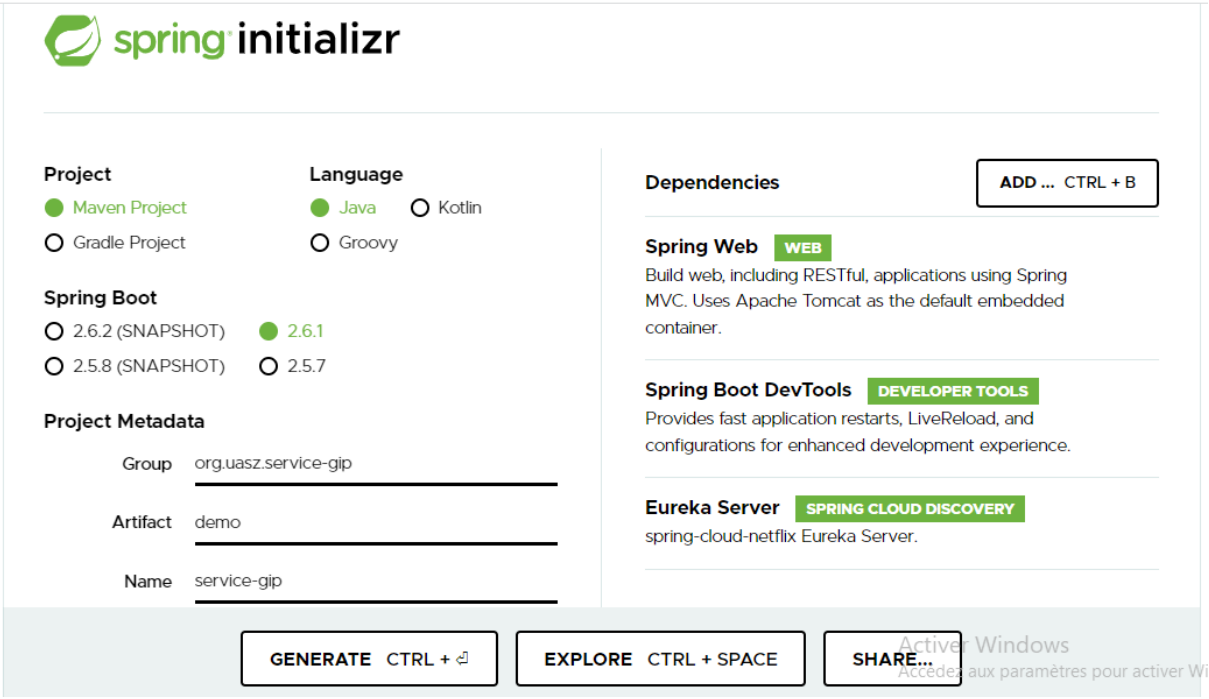


*Figure 4-3 : Création d'un projet avec spring boot en ligne*

Après avoir généré le projet avec toutes ses dépendances nécessaires, on l'importe dans notre IDE eclipse STS.

Après la mise en place de toute les micro-services avec chacun toutes ses fonctionnalités, il nous faut maintenant les faire communiquer pour que le système global puisse répondre correctement aux attentes des utilisateurs.

### 4.2.4 Outils pour faire communiquer les micro-services

Dans le modèle d'architecture micro-service, un système distribué s'exécute sur plusieurs machines différentes et chaque service est un composant ou un processus d'une application d'entreprise. Cela signifie que ces services sur les multiples machines doivent gérer les demandes des clients de cette application d'entreprise. Parfois, tous ces services collaborent pour traiter ces demandes. Ainsi, tous les services interagissent à l'aide d'un mécanisme de communication inter-services. [27]

#### 4.2.4.1 Mise en place de serveur discovery et de client discovery pour la communication des micro-services

Pour que notre architecture de micro-services puisse respecter les critères d'élasticité et de couplage lâche, et assurer une communication inter-micro-services fluide, il faut que nous disposions d'un bon système de découverte de services et d'équilibrage de charge. De sorte que nos micro-services puissent à tout moment connaître la liste de tous leurs homologues en cours d'exécution et les informations permettant de les contacter, à savoir : nom, adresse IP et numéro de port. Pour cela, nous allons nous servir de l'opportunité que nous a offerte Spring Cloud avec ses deux composants à savoir le **serveur discovery** et le **client discovery.** Ces deux composants nous permettront de disposer d'un équilibrage de charge et de la découverte de service.

a) Serveur discovery (Eureka)

**Eureka** est une application permettant la localisation d'instances de services. Elle se caractérise par une partie serveur et une partie cliente. La communication entre les parties se fait via les API Web exposées par le composant serveur. Ces services doivent être créés en tant que clients Eureka, ayant pour objectif de se connecter et s'enregistrer sur un serveur Eureka. De ce fait, les clients vont pouvoir s'enregistrer auprès du serveur et périodiquement donner des signes de vie. Le service Eureka (composant serveur) va pouvoir conserver les informations de localisation desdits clients afin de les mettre à disposition aux autres services. [28]

Nous avons nommé ce serveur dans notre cas « **service-d-enregistrement »** et il est considéré comme un micro-service appart.

Il convient donc de mettre en place un serveur Eureka et de transformer les services (voir figure 4-2) de notre système en clients Eureka. Fort heureusement, Spring Cloud et son projet

supportant la plupart des applications de Netflix, ont intégré Eureka dans leur solution, créant ainsi l'annotation : @EnableEurekaServer. Cette annotation fait partie du module *spring-cloud-starter-eureka-server* que nous devons intégrer comme dépendance dans notre projet spring.

Pour déployer notre serveur de discovery, nous avons dû créer une application Spring Boot au sein de laquelle, nous avons embarqué un serveur eureka. Pour cela, il nous suffit juste d'ajouter la dépendance suivante au niveau du fichier pom.xml.

```
17      <properties>
18          <java.version>1.8</java.version>
19          <spring-cloud.version>Hoxton.SR1</spring-cloud.version>
20          <maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>
21      </properties>
22
23      <dependencies>
24          <dependency>
25              <groupId>org.springframework.cloud</groupId>
26              <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
27          </dependency>
28
29          <dependency>
30              <groupId>org.springframework.boot</groupId>
31              <artifactId>spring-boot-starter-test</artifactId>
32              <scope>test</scope>
33              <exclusions>
34                  <exclusion>
35                      <groupId>org.junit.vintage</groupId>
36                      <artifactId>junit-vintage-engine</artifactId>
37                  </exclusion>
38              </exclusions>
39          </dependency>
40      </dependencies>
41
```

*Figure 4-4 : Fichier pom.xml de « service-d-enregistrement »*

Après avoir créé le projet, nous devons juste ajouter l'annotation @EnableEurekaServer dans la classe principale.

La figure 4-5 présente la classe principale de notre micro-service « **service-d-enregistrement** » pour le compléter :

```java
package org.uasz.serviceergmt.servicedenregistrement;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class ServiceDEnregistrementApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServiceDEnregistrementApplication.class, args);
    }

}
```

*Figure 4-5 : micro-service « service-d-enregistrement »*

### b) Client discovery (Feign)

**Feign** est un client HTTP qui facilite grandement l'appel des API exposées par les différents micro-services. Il est donc capable de créer et d'exécuter des requêtes HTTP basées sur les annotations et informations que l'on fournit. C'est un peu l'équivalent en code de Postman. [29]

Nous avons créé notre serveur d'enregistrement afin de recevoir les informations des différents services, et de se déplacer au centre de l'architecture. Nous allons créer maintenant nos services clients à savoir **« service-ins-acad »**, **« service-info-ped »**, **« service-gip »** et **« service-examen »** en les appliquant l'annotation *@EnableFeignClients.*

Les figures suivantes représentent la classe main de chacun de nos micro-services avec l'annotation *@EnableFeignClients* juste en haut des noms des classes :

```
13  import org.uasz.insacad.serviceinsacad.entites.Filiere;
14  import org.uasz.insacad.serviceinsacad.entites.Ufr;
15
16  @SpringBootApplication
17  @EnableFeignClients
18  public class ServiceInsAcadApplication {
19
20      public static void main(String[] args) {
21          SpringApplication.run(ServiceInsAcadApplication.class, args);
22
23          }
24      @Bean
25      CommandLineRunner star(RepositoryRestConfiguration configuration){
26
27          return args->{
28              configuration.exposeIdsFor(Etudiant.class);
29              configuration.exposeIdsFor(Filiere.class);
30              configuration.exposeIdsFor(Departement.class);
31              configuration.exposeIdsFor(Ufr.class);
32              /*etudiantRepository.save(new Etudiant("Dieng", "Sidiya", new Date(), "dieng@gmail.com","
33              etudiantRepository.save(new Etudiant("Mare", "Ibrahima", new Date(), "mareg@gmail.com","G
34              etudiantRepository.save(new Etudiant("Fall", "Anta", new Date(), "anta@gmail.com","Genie
35              etudiantRepository.save(new Etudiant("Bakhoum", "Ana", new Date(), "dieng@gmail.com","Gen
36              etudiantRepository.save(new Etudiant("Diop", "Fatou", new Date(), "dopg@gmail.com","Genie
37              etudiantRepository.save(new Etudiant("Fall", "Astou", new Date(), "fall@gmail.com","Genie
38              etudiantRepository.save(new Etudiant("Dieng", "Mamadou", new Date(), "diengmad@gmail.com"
39          };
40      }
41
42  }
```

*Figure 4-6 : micro-service « service-ins-acad »*

```
14  import org.uasz.infoped.serviceinfoped.entites.Ue;
15  import org.uasz.infoped.serviceinfoped.entites.Ufr;
16
17  @SpringBootApplication
18  public class ServiceInfoPedApplication {
19
20      public static void main(String[] args) {
21          SpringApplication.run(ServiceInfoPedApplication.class, args);
22      }
23
24      @Bean
25      CommandLineRunner star(UeRepository ueRepository, EcRepository ecRepository, FiliereRepository filiereRepository,
26              RepositoryRestConfiguration configuration){
27          return args->{
28              configuration.exposeIdsFor(Ufr.class);
29              configuration.exposeIdsFor(Departement.class);
30              configuration.exposeIdsFor(Filiere.class);
31              configuration.exposeIdsFor(Ue.class);
32              configuration.exposeIdsFor(Ec.class);
33
34              /*Filiere f1=filiereRepository.save(new Filiere("Mathematique"));
35              Filiere f2=filiereRepository.save(new Filiere("Genie Logiciel"));
36
37              Ue ue1=ueRepository.save(new Ue("Genie Logiciel", "M1","semestre 1",f2));
38              Ue ue2=ueRepository.save(new Ue("Modèle de Calcul", "M1","semestre 1",f2));
39              Ue ue3=ueRepository.save(new Ue("Programmation", "M1","semestre 1",f2));
```

*Figure 4-7 : micro-service « service-info-ped »*

```java
import org.uasz.servicegip.servicegip.entites.Ue;
import org.uasz.servicegip.servicegip.entites.Ufr;
//import org.uasz.servicegip.servicegip.entites.IP;

@SpringBootApplication
@EnableFeignClients
public class ServiceGipApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServiceGipApplication.class, args);
    }

    @Bean
    CommandLineRunner star(RepositoryRestConfiguration configuration ) {
        return args -> {
        configuration.exposeIdsFor(Annee.class);
        configuration.exposeIdsFor(Ip.class);
        configuration.exposeIdsFor(Ue.class);
        configuration.exposeIdsFor(Choix.class);
        configuration.exposeIdsFor(Filiere.class);
        configuration.exposeIdsFor(Etudiant.class);
        configuration.exposeIdsFor(Ec.class);
        configuration.exposeIdsFor(Ufr.class);
        configuration.exposeIdsFor(Departement.class);
        };
    }
        /*
```

*Figure 4-8 : micro-service « service-gip »*

```java
import org.uasz.examen.serviceexamen.entites.Ec;
import org.uasz.examen.serviceexamen.entites.Etudiant;
import org.uasz.examen.serviceexamen.entites.Examen;
import org.uasz.examen.serviceexamen.entites.Filiere;
import org.uasz.examen.serviceexamen.entites.Ue;
import org.uasz.examen.serviceexamen.entites.Ufr;

@SpringBootApplication
@EnableFeignClients
public class ServiceExamenApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServiceExamenApplication.class, args);
    }
    @Bean
    CommandLineRunner star(RepositoryRestConfiguration configuration,ExamenRepository examenRepository,
            InfoPedService infoPedService){
        return args->{
            configuration.exposeIdsFor(Etudiant.class);
            configuration.exposeIdsFor(Examen.class);
            configuration.exposeIdsFor(Ufr.class);
            configuration.exposeIdsFor(Departement.class);
            configuration.exposeIdsFor(Filiere.class);
            configuration.exposeIdsFor(Ue.class);
            configuration.exposeIdsFor(Ec.class);
        };
    }

}
```

*Figure 4-9 : micro-service « service-examen »*

```
52      <dependency>
53          <groupId>org.springframework.cloud</groupId>
54          <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
55      </dependency>
56      <dependency>
57          <groupId>org.springframework.cloud</groupId>
58          <artifactId>spring-cloud-starter-openfeign</artifactId>
59      </dependency>
```

*Figure 4-10 : Les dépendances nécessaires*

#### 4.2.4.2 Localisation du serveur d'enregistrement

Nous avons déjà mis en place la relation client/serveur. Cependant nos différents micro-services et le serveur Eureka ne peuvent pas encore se communiquer, car ils ne savent pas où se trouve le serveur Eureka. Il nous faut donc les indiquer la localisation du serveur, et aussi configurer ce dernier.

Pour se faire nous allons dans le fichier *application.propreties* qui se trouve au niveau l'arborescence de chacun de nos services afin de les configurer. On y configure notamment Eureka ainsi que le nom du micro-service pour pouvoir l'utiliser plus tard comme identifiant de service. Les deux captures suivantes nous donnent en détail les configurations respectives du serveur d'enregistrement des micro-services clients.

```
1  server.port=8761
2  eureka.client.register-with-eureka=false
3  eureka.client.fetch-registry=false
4  eureka.client.eureka-server-total-connections=5000
5  eureka.client.eureka-server-read-timeout-seconds=5000
```

*Figure 4-11 : Fichier de configuration du serveur d'enregistrement*

```
1  server.port=8081
2  spring.application.name=service-ins-acad
3  spring.cloud.discovery.enabled=true
4  eureka.client.service-url.defaltZone=http://localhost:8761/eurka
5  eureka.client.eureka-server-read-timeout-seconds=5000
6  spring.datasource.url=jdbc:mysql://localhost:3306/bd_insacad?zeroDateTimeBehavior=CONVERT_TO_NULL&serverTimezone=UTC
7  spring.datasource.username=root
8  spring.datasource.password=
9  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
10 spring.jpa.hibernate.ddl-auto=update
11 spring.servlet.multipart.max-file-size=2MB
12 spring.servlet.multipart.max-request-size=2MB
```

*Figure 4-12 : Fichier de configuration de « service-ins-acad »*

Ici nous avons juste donné l'exemple du fichier de configuration pour le seul micro-service « service-ins-acad » mais c'est presque pareil pour tous les autres micro-services. Seuls les noms de ces derniers changent d'un fichier à un autre.

Nos micro-services bénéficient à présent du client Eureka, qui va enregistrer nos instances de micro-services à chaque démarrage. Les voici sur la figure 4-13 après son démarrage, il est accessible sur l'adresse : http://localhost:8761/

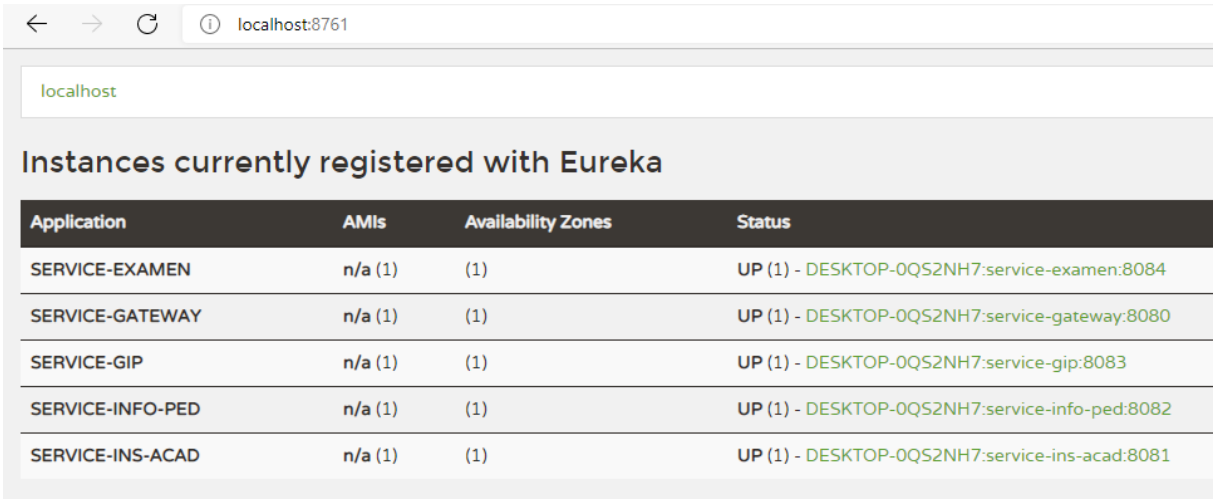


*Figure 4-13 : Serveur d'enregistrement avec Eureka*

Tous les micro-services sont maintenant enregistrés dans le serveur d'enregistrement Eureka. Ils sont donc identifiables sur leur nom, leur adresse IP et leur numéro de port.

Nous avons à travers ce chapitre évoqué deux sections essentielles. Dans la première section nous avons parlé des technologies et langages que nous avons utilisés pour l'implémentation de nos micro-services. Puis, nous avons consacré la deuxième section à la communication des micro-services. Le chapitre suivant sera l'objet de la présentation et du fonctionnement de nos prototypes micro-services.

# CHAPITRE 5 : Présentation et fonctionnement de nos prototypes micro-services

Dans cette partie nous allons nous servir des résultats découlant de la section conception afin d'expliquer les grands points de ce chapitre. Ainsi nous allons premièrement faire la présentation de chacun de nos prototypes micro-services suivant notre architecture micro-services technique. Deuxièmement il sera questions d'expliquer les points saillants du fonctionnement de chacun de ces micro-services.

## 5.1 Présentation des micro-services

Notre architecture micro-services technique (voir figure 4-2) qui nous permet implémenter nos prototypes, est composée de sept (7) micro-services à savoir :

- **services-ins-acad :** Micro-services qui gère la gestion des inscriptions académiques
- **service-info-ped :** Micro-services qui a la charge de la gestion des informations pédagogiques ;
- **service-gip :** Micro-services chargé de gérer les inscriptions pédagogiques ;
- **service-examen :** Micro-services qui gère la gestion des évaluations des étudiants ;
- **service-d-enregistrement :** Micro-service pour gérer les différentes instances des micro-services du système. Il est indispensable à la communication des autres micro-services car il les enregistre tous pour leur identification sur leur nom, leur adresse IP et leur numéro de pour ;
- **service-getway :** Micro-service pour filtrer et répartir les différents appels à l'API vers les autres micro-services ;
- **service-auth :** Micro-service exposant les opérations d'authentification et de sécurité au sein de l'API du système. Par exemple, la connexion et la déconnexion d'un utilisateur se fera via ce micro-service.

Chacun de ces micro-services cités est un projet que nous avons réalisé à part avec l'IDE eclipse STS. Ils sont autonomes et chacun gère ses propres données.

D'après la figure 5-1 ci-dessous, nous avons à gauche l'ensemble des micro-services que nous avons mis en place et à droite lorsqu'ils qu'ils ont été démarrés.

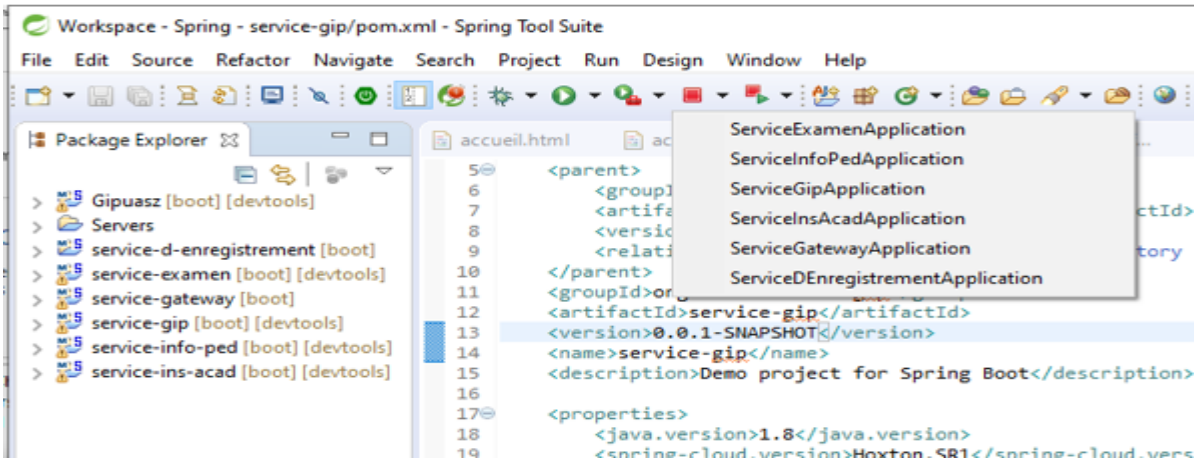


*Figure 5-1 : Les micro-services après démarrage*

Après l'implémentation de nos prototypes, les clients web peuvent maintenant consommer les services rendus par chaque micro-service. Et pour faciliter la compréhension, nous avons dans la réalisation des front-end de chaque prototype, différencié les couleurs des interfaces. Ceci nous permettra de bien identifier chacun de nos micro-services et ses différentes fonctionnalités. Les captures d'écrans suivant présentent l'interface d'accueil de chacun.

### 5.1.1 Micro-services « service-ins-acad »

C'est le micro-service qui gère la gestion des inscriptions académiques. Il invoque les services du micro-service « **service-info-ped** » et offre des services au micro-service « **service-gip** ».


*Figure 5-2 : Template du micro-service « service-ins-acad »*

### 5.1.2 Micro-services « service-info-ped »

C'est le micro-service qui assure la gestion des informations pédagogiques. Il communique avec les micro-services : « **service-ins-acad** », « **service-gip** » et « service-examen » pour leur offrir ses services qu'il expose.


*Figure 5-3 : Template du micro-service « service-ins-acad »*

### 5.1.3 Micro-service « service-gip »

C'est le micro-service qui gère la gestion des inscriptions péagogiques. Il invoque les services des micro-services « **service-info-ped** » et « **service-ins- acad** » mais aussi offre des services au micro-service « service-examen ».


*Figure 5-4 : Template du micro-service « service-gip »*

### 5.1.4 Micro-service « service-gip »

Ce micro-service est réalisé pour la gestion des évaluations (examen, devoir etc.). Il invoque les services des micro-services : « **service-info-ped** » et « **service-gip** ».


*Figure 5-5 : Template du micro-service « service-examen »*

## 5.2 Fonctionnement de nos prototypes micro-services

Dans cette section nous allons d'abord parler du fonctionnement interne des micro-services. Ensuite montrer le fonctionnement externe de ces derniers en exploitant quelques fonctionnalités à travers les interfaces graphiques présentés dans la section précédente.

### 5.2.1 Fonctionnement interne

Bien qu'ils soient développés et déployés indépendamment, aucun de nos micro-services n'a pas sa raison d'exister seul. Il existe de nombreux micro-services dans notre architecture, et pour que système globale envisagée fonctionne bien, ces micro-services doivent communiquer et interagir les uns avec les autres. Cette communication implique des invocations de services, c'est-à-dire chaque micro-service peut demander des services à un autre micro-service qui à son tour sollicite un service via un autre pour satisfaire la demande. Cela nécessite donc une bonne gestion des accès pour la sécuriser de nos micro-services et du système global.

#### 5.2.1.1 Gestion des accès avec le micro-service « **service-auth** »

La sécurité est toujours une préoccupation transversale majeure pour toute application. Nous avions au paravent une seule couche de sécurité dans le service back-end monolithique qui sécurisait toutes les demandes d'applications front-end. Avec l'architecture micro-services, il n'y a plus un seul service back-end pour gérer l'authentification. Les applications frontales invoquent plusieurs services principaux, qui peuvent à leur tour appeler d'autres services. Comme nos micro-services ont de nombreux services qui traitent chacun des responsabilités distinctes, la solution évidente pour la sécurité est de mettre en place un micro-service d'authentification et d'autorisation. C'est là que nous intégrons **Keycloak** dans notre micro-service « **service-auth** » car il fournit les fonctionnalités dont nous avons besoin pour sécuriser nos micro-services.

#### 5.2.1.2 Présentation de Keyclaok

Keycloak est une solution open source de gestion des identités (identity manager) et des accès destinés aux applications et aux services modernes. Cela facilite la sécurisation des applications et des services Il offre un large éventail de fonctionnalités, telles que la connexion unique, l'authentification et l'autorisation, la connexion sociale, l'authentification multifactorielle et la gestion centralisée des utilisateurs. Il fournit plusieurs fonctionnalités :

- **Authentification unique :** les utilisateurs s'authentifient avec Keycloak plutôt qu'avec des applications individuelles. Cela signifie que nos applications n'ont pas à traiter avec les formulaires de connexion, l'authentification des utilisateurs et le stockage des utilisateurs. Une fois connecté à Keycloak, les utilisateurs ne doivent plus se connecter pour accéder à une autre application. Ceci s'applique également à la déconnexion. Keycloak fournit une déconnexion unique, ce qui signifie que les utilisateurs ne doivent

se déconnecter qu'une seule fois pour se déconnecter de toutes les applications utilisant Keycloak.

- **Courtage d'identité et connexion sociale :** l'activation de la connexion avec les réseaux sociaux est facile à ajouter via la console d'administration. Il suffit de sélectionner le réseau social que nous souhaitons ajouter. Aucun code ou modification de notre application n'est requis. Keycloak peut également authentifier les utilisateurs avec les fournisseurs d'identité OpenID Connect ou SAML 2.0 existants. Encore une fois, il s'agit simplement de configurer le fournisseur d'identité via la console d'administration. La figure 5-6 illustre les réseaux sociaux et les courtages d'identité que keycloak supporte.

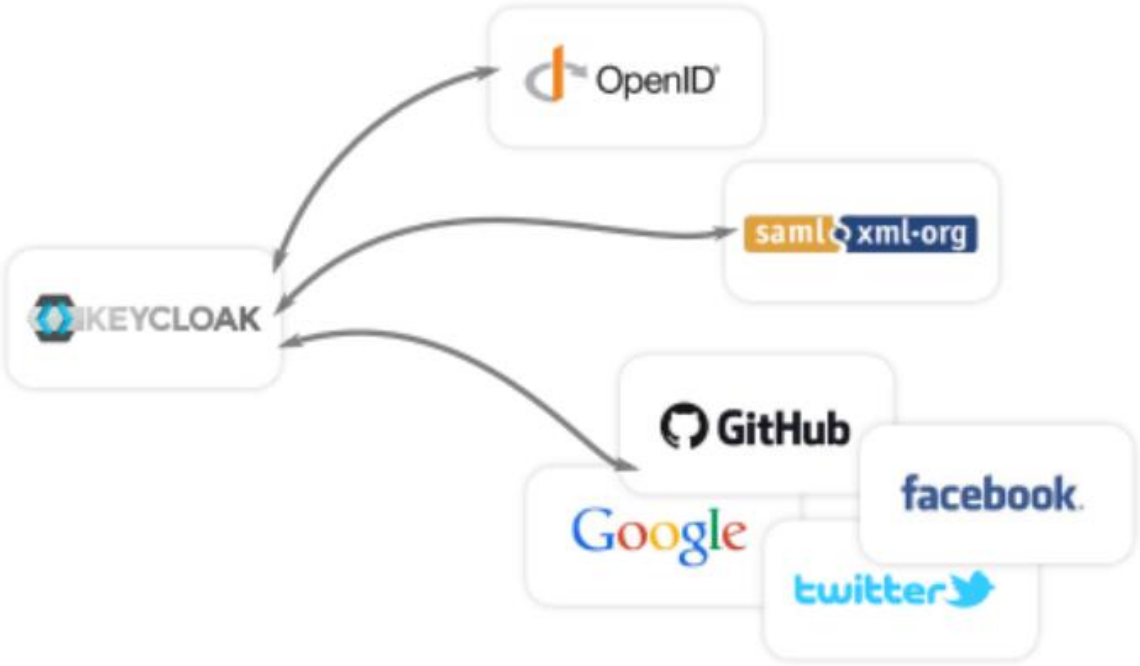


*Figure 5-6 : Courtage d'identité et connexion sociale*

- **Console d'administration :** grâce à la console d'administration, les administrateurs peuvent gérer de manière centralisée tous les aspects du serveur Keycloak. Ils peuvent activer et désactiver diverses fonctionnalités. Ils peuvent configurer le courtage d'identité et la fédération d'utilisateurs. Ils peuvent créer et gérer des applications et des services, et définir des politiques d'autorisation détaillées. Ils peuvent également gérer les utilisateurs, notamment les autorisations et les sessions.

- **Services d'autorisation :** si l'autorisation par rôle ne couvre pas nos besoins, Keycloak fournit également des services d'autorisation détaillés. Cela nous permet de gérer les autorisations pour tous nos services à partir de la console d'administration Keycloak et nous donne le pouvoir de définir exactement les stratégies dont nous avons besoin.

#### 5.2.1.3 Service de découverte « **service-d-enregistrement** »

Dans notre architecture, les micro-services doivent communiquer les uns avec les autres afin d'assurer le bon fonctionnement du système. C'est le service de découverte qui gère la façon d'enregistrement et de localisation des micro-services déployés. Nous avons choisi dans notre cas le service de découverte **Eureka** de Netflix qui est un Framework de service de découverte conçu pour les déploiements. Les instances des micro-services enregistrés chez le service de découverte Eureka sont présentés par la figure 5-7 suivant :

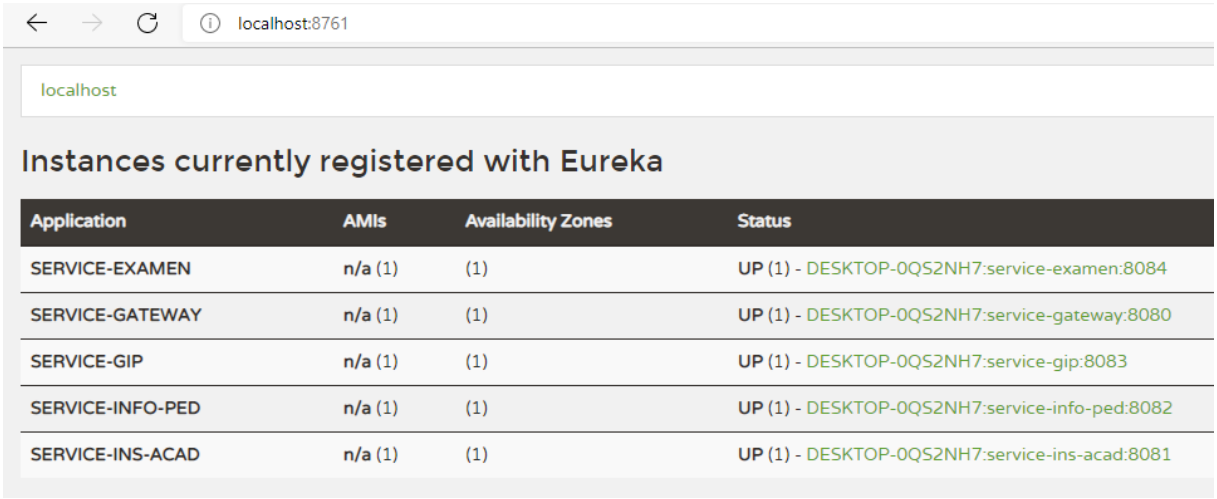


*Figure 5-7 : Interface de service découverte Eureka*

Après avoir été enregistrés, les micro-services sont à partir de là identifiables par leur nom, leur adresse IP et leur numéro de port.

#### 5.2.1.4 API gateway micro-service « **service-gateway** »

Notre architecture en micro-services implique l'utilisation de plusieurs services qui peuvent changer ou modifier d'emplacement. Ce changement ne doit être visible coté clients. Pour assurer la transparence entre le partitionnement en micro-services et la navigation de client nous avons utilisé une passerelle nommée « service-gateway » qui est aussi un micro-service. Elle est le point d'entrée du système, le client ne doit savoir que son adresse et elle se charge de le router vers le micro-service qui répond à son besoin en toute transparence.

Toutes les requêtes lancées seront filtrées et réparties par cette passerelle vers les autres micro-services. Pour faciliter la compréhension, prenons un exemple d'une séquence où un client http lance une requête demandant un service à savoir liste des étudiants ayant satisfait leur inscription académique afin de procéder à leur inscription pédagogique.

Lorsque le client lance la requête HTTP ici noté « **req1** » dont l'adresse est : http//localhost :8080/SERVICE-GIP/listeEtudiants, cette requête passe d'abord par le micro-service gateway « **service-gateway** » répondant sur le port 8080. Il contact avec la requête « **req2** » le micro-service « **service-d-enregistrement** » qui enregistre tous les noms, adresses IP et le numéro de port de chaque micro-service (voir figure 5-8). Il renseigne au gateway le nom, l'adresse et le numéro de port de « **service-gip** », le gateway lance encore une deuxième requête « **req3** » pour informer le micro-service concerné « **service-gip** ». Ce dernier contact à son tour le micro-service « **service-d-enregistrement** » avec la requête « **req4** ». Il lui donne le nom, l'adresse et le numéro port de « **service-ins-acad** ». Le micro-service « **service-gip** » contact avec la requête « **req5** » le micro-service « **service-ins-acad** ». Ce dernier lui donne sa réponse « **reponse 1** » qu'il va transmettre ensuite au « **service-gateway** » avec la réponse « **reponse 2** ». Le gateway transmet enfin cette réponse par « **reponse 3** » au client demandeur.

La figure 5-8 suivante illustre le processus d'une demande de service faite par un client http et adressée au micro-service « **service-gip** »

**Req1** : http//localhost :8080/SERVICE-GIP/listeEtudiants

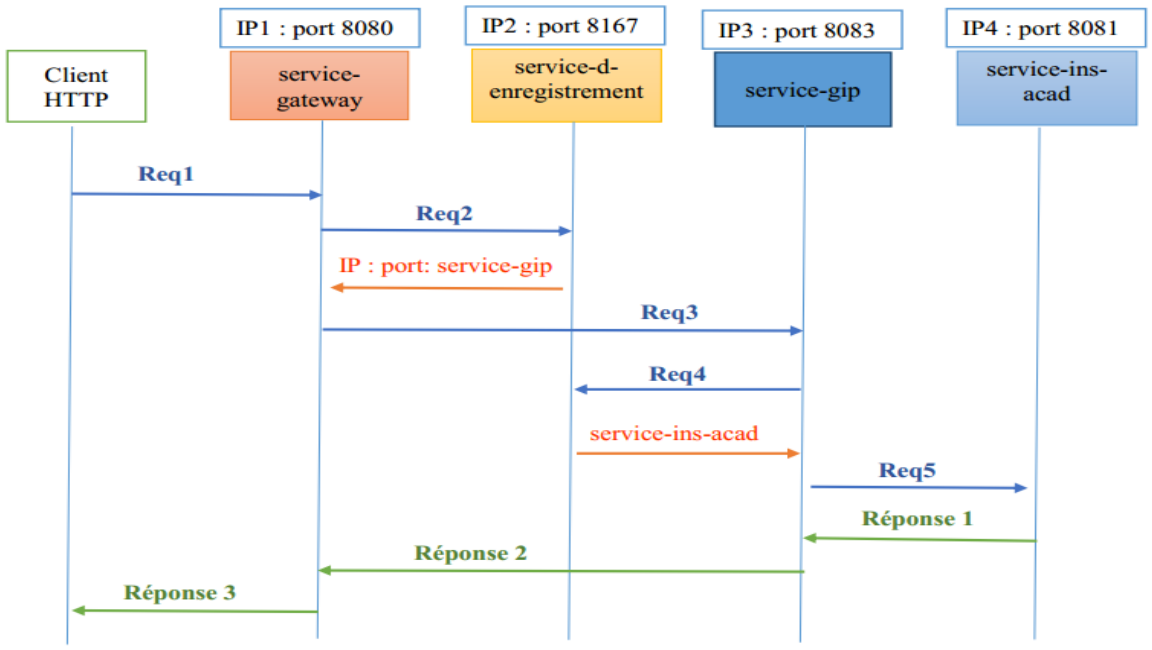


*Figure 5-8 : Communication entre « service-gip» et «service-ins-acad»*

### 5.2.2 Fonctionnement externe

Dans cette section, nous allons montrer l'ensemble des interfaces graphiques que nous avons obtenues avec quelques fonctionnalités. Nous allons aussi voir comment à travers ces interfaces graphiques les micro-services invoque-t-ils des services d'un autre micro-service.

#### 5.2.2.1 Micro-service « service-ins-acad »

Ce micro-service est le responsable de la gestion des inscriptions académiques. Il communique avec le micro-service « **service-info-ped** » par un lien entrant et « **service-gip** » par un lien sortant. Dans notre exemple ci-dessus figure 5-8, micro-service « **service-gip** » sollicitait un service au micro-service « **service-ins-acad** » à savoir la liste des étudiants ayant fait leur inscription académique selon la filière et le niveau. Cette liste d'étudiants est obtenue par « **service-ins-acad** » après avoir uploadé un fichier CSV qui les contient.

Les captures qui suivent résument le processus avec l'indication de la flèche rouge.

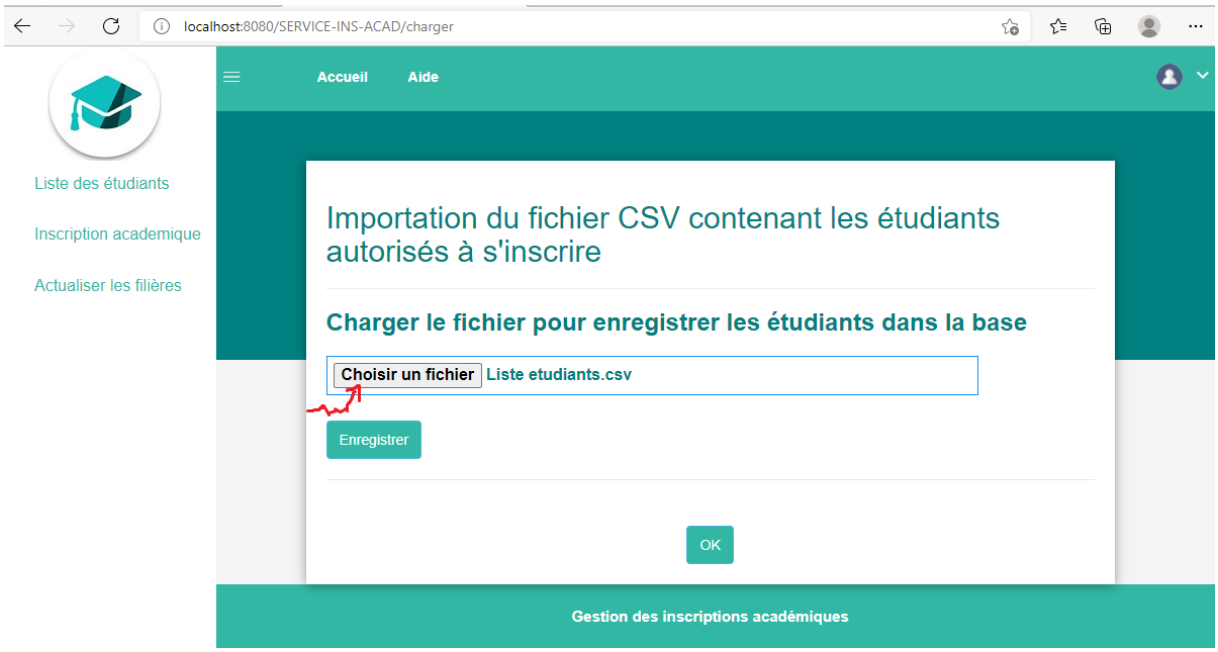


*Figure 5-9 : Interface de choix d'un fichier*

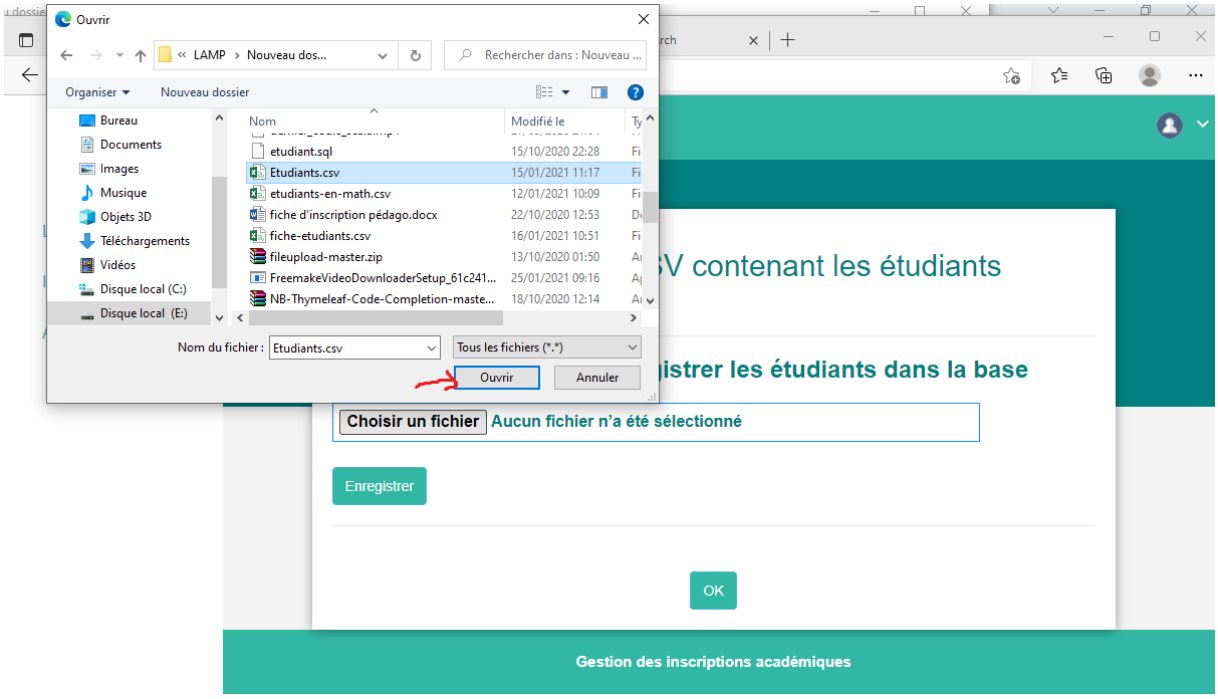


*Figure 5-10 : Interface de sélection d'un fichier*

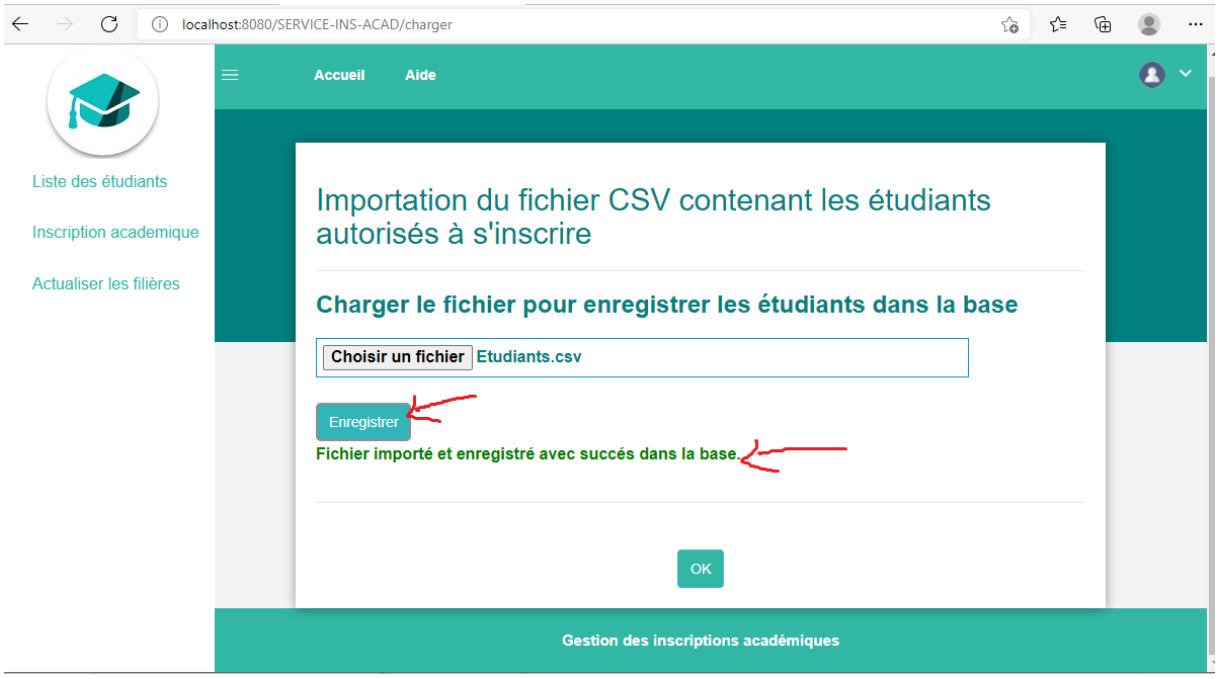


*Figure 5-11 : Interface d'enregistrement d'un fichier dans la base*

#### 5.2.2.2 Demande d'un service par un client http

Le résultat du processus de la demande du client peut être visualisé sur l'interface graphique du micro-service « ***service-gip*** ». Pour ce faire l'utilisateur doit au préalable s'authentifier via l'interface graphique du micro-service « **service-auth** » représenté par la figure 5-12 suivante.

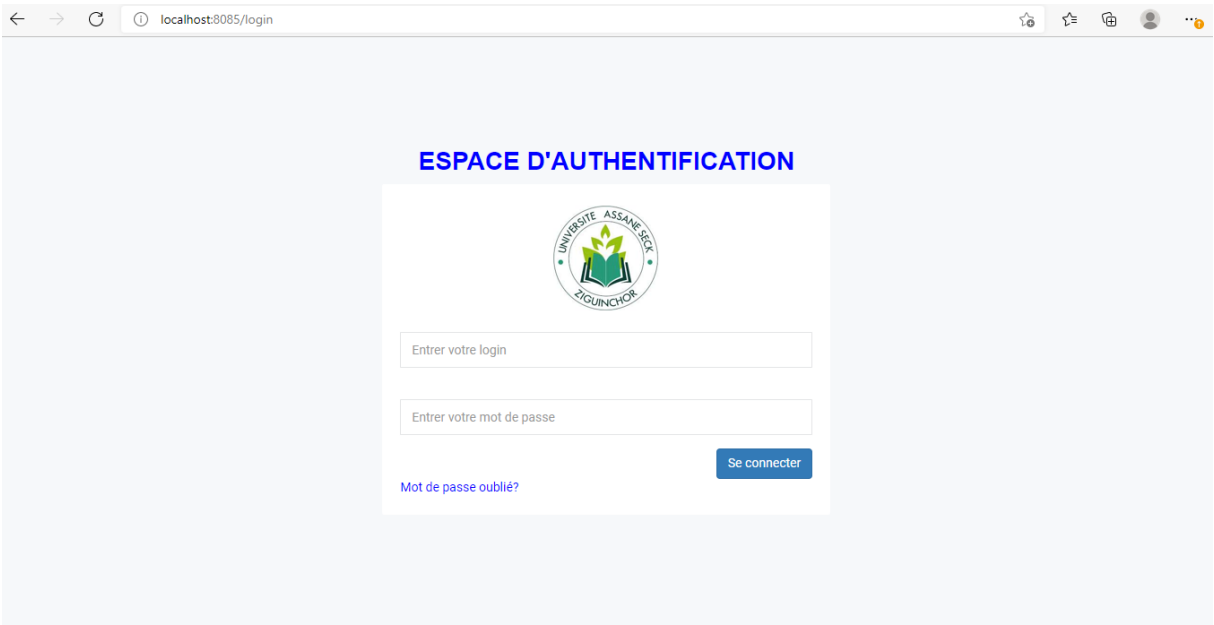


*Figure 5-12 : Page d'authentification*

Lorsque l'utilisateur s'authentifie il est dirigé vers la page d'accueil du micro-service « service-gip » (voir figure 5-4 plus haut). Il clique sur le lien intitulé « **Liste des étudiants** » et la requête est lancée soit l'adresse : http//localhost:8080/SERVICE-GIP/listeEtudiants.

La figure 5-13 ci-dessous montre résultat obtenu après lancement de cette requête.

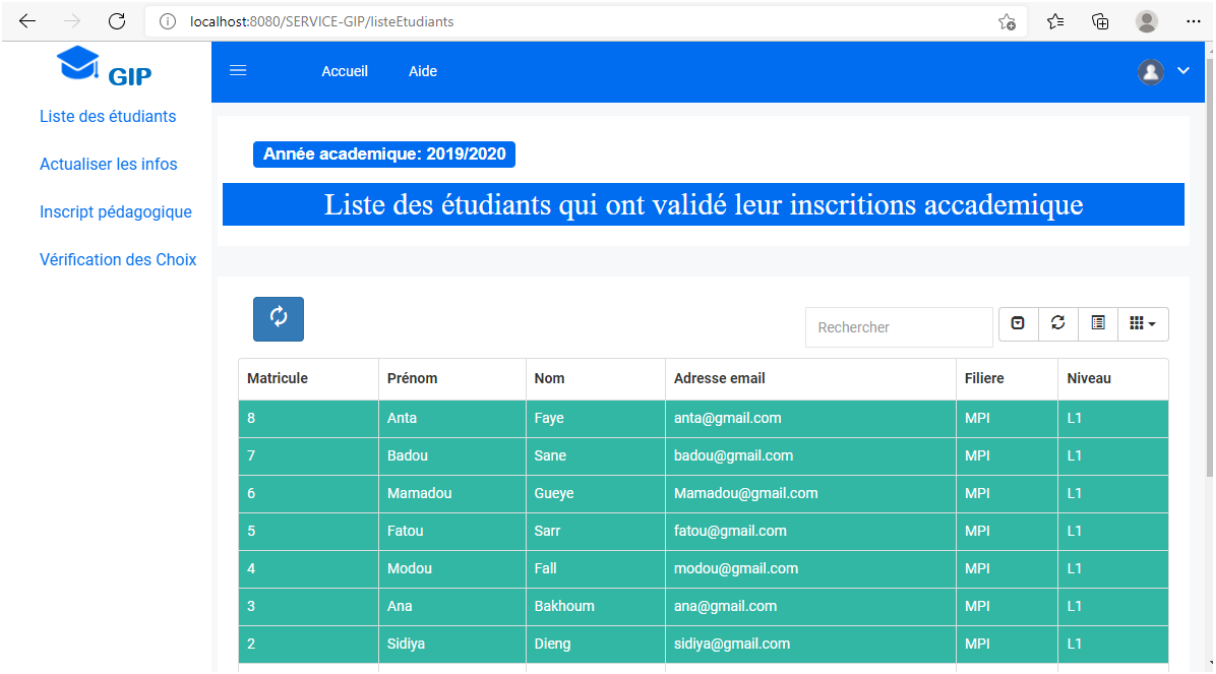


*Figure 5-13 : Résultat de la requête demandant la liste des étudiants*

L'interface graphique du micro-service « service-gip » est de couleur bleue. Donc ici la couleur sur la liste des étudiants obtenue montre que les données sont issues du micro-service « **service-ins-acad** » car elle est identique à la couleur de son interface graphique.

Pour faciliter la compréhension nous avions choisi d'utiliser des couleurs différentes pour l'interface graphique de chaque micro-service. Ici comme c'est le micro-service *« service-gip »* (couleur bleue) qui demande un service au micro-service *« service-ins-acad »* (couleur mélangée), ce qui explique la différence de couleurs obtenues sur la liste des étudiants au niveau de la figure 5-8 suivant.

#### 5.2.2.3 Inscription pédagogique d'un étudiant

Après l'obtention de la liste des étudiants demandée, le micro-service « ***service-gip*** » garde ces informations dans sa propre base de données. De ce fait en cas de panne ou de disfonctionnement du micro-service *« **service-ins-acad** »*, son fonctionnement ne sera pas empêché. Donc l'utilisateur peut continuer à faire l'inscription pédagogique des étudiants. Et pour faire cette inscription, le micro-service *« **service-gip** »* a cette fois ci besoin de contacter le micro-service *« **service-info-ped** »* afin d'obtenir les informations pédagogiques (UEs, ECs, maquettes etc).

Le processus d'inscription est illustré par les figures suivantes avec une flèche rouge qui indique les différentes étapes à suivre du début à la fin de l'inscription :

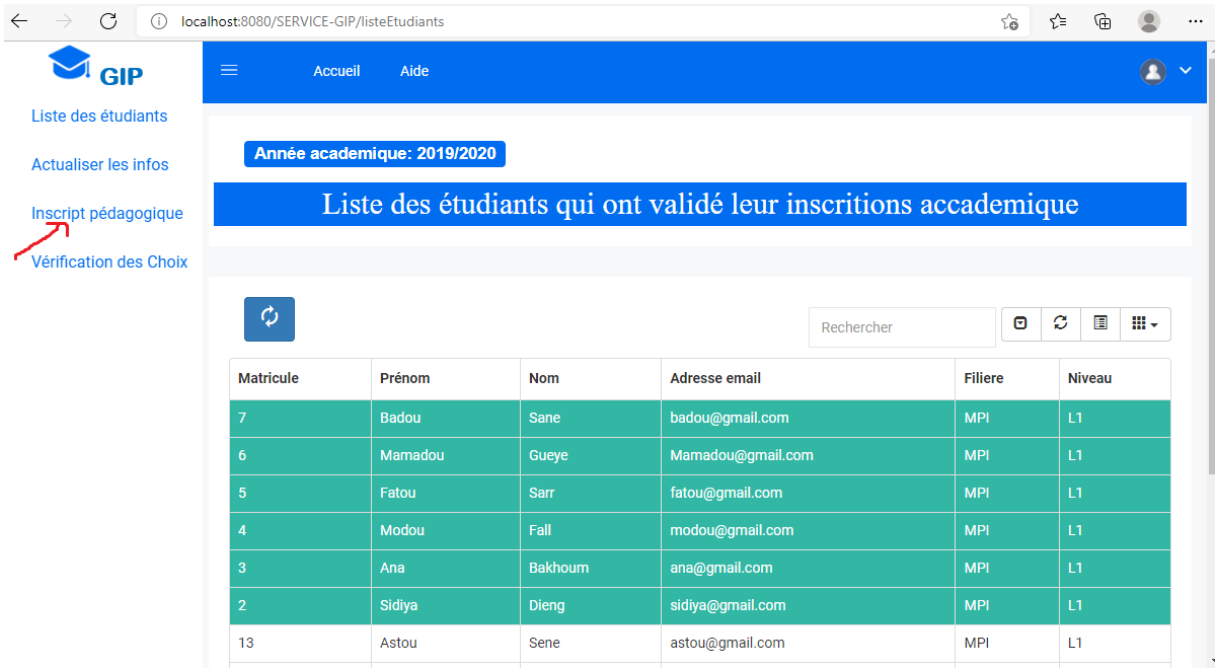


*Figure 5-14 : Interface pour faire une inscription pédagogique*

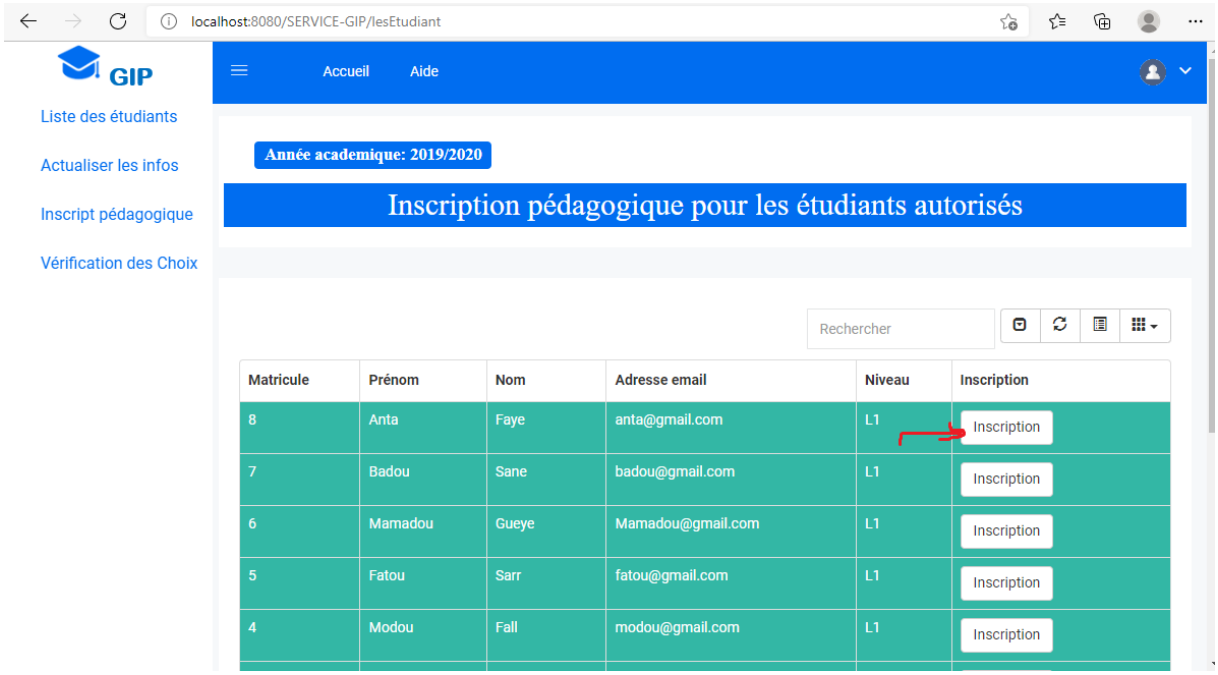


*Figure 5-15 : Listes des étudiant à inscrire*

Ici en appuyant sur le bouton Inscription, une requête portant l'adresse http//localhost :8080/SERVICE-INFO-PED/choix est à nouveau lancée. Cela montre que le micro-service « **service-info-ped** » est à son tour sollicité afin d'obtenir la liste des enseignements à choisir. La figure 5-13 ci-dessous nous en détaille plus avec la couleur verte montrant que les données sont issues de « service-info-ped » car identique à celle de son interface (**voir figure 5-3** plus haut)

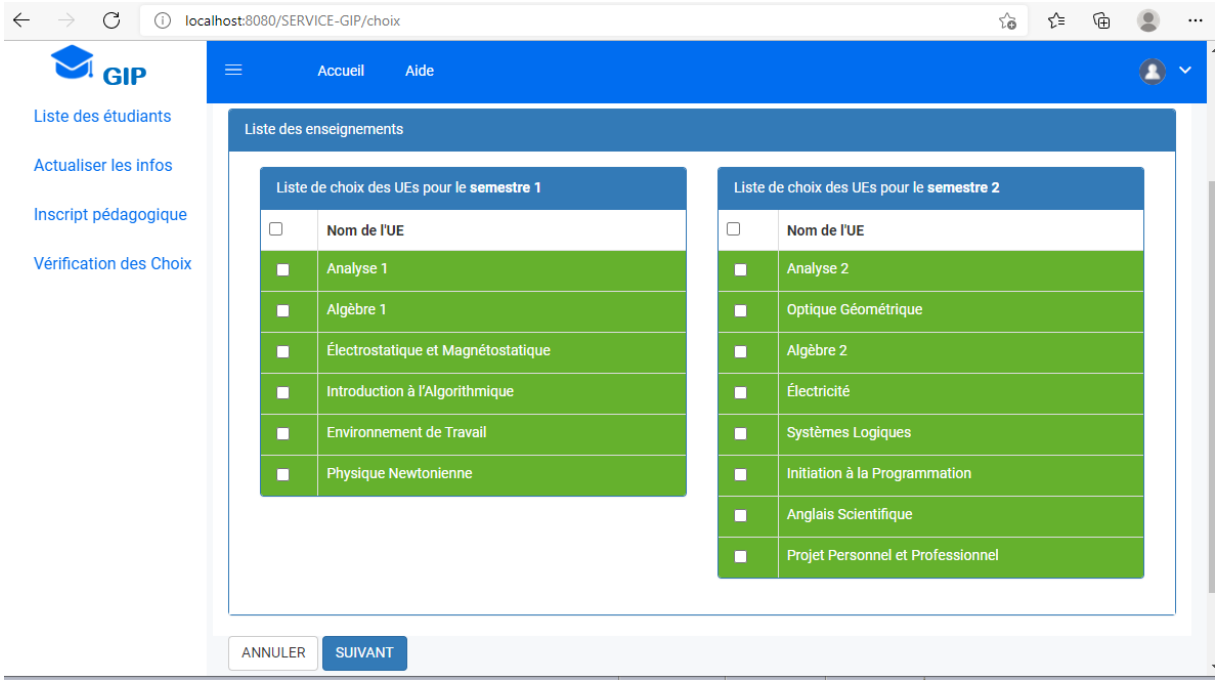


*Figure 5-16 : Liste de choix des matières*

Choix des enseignements puis clique du bouton SUIVANT

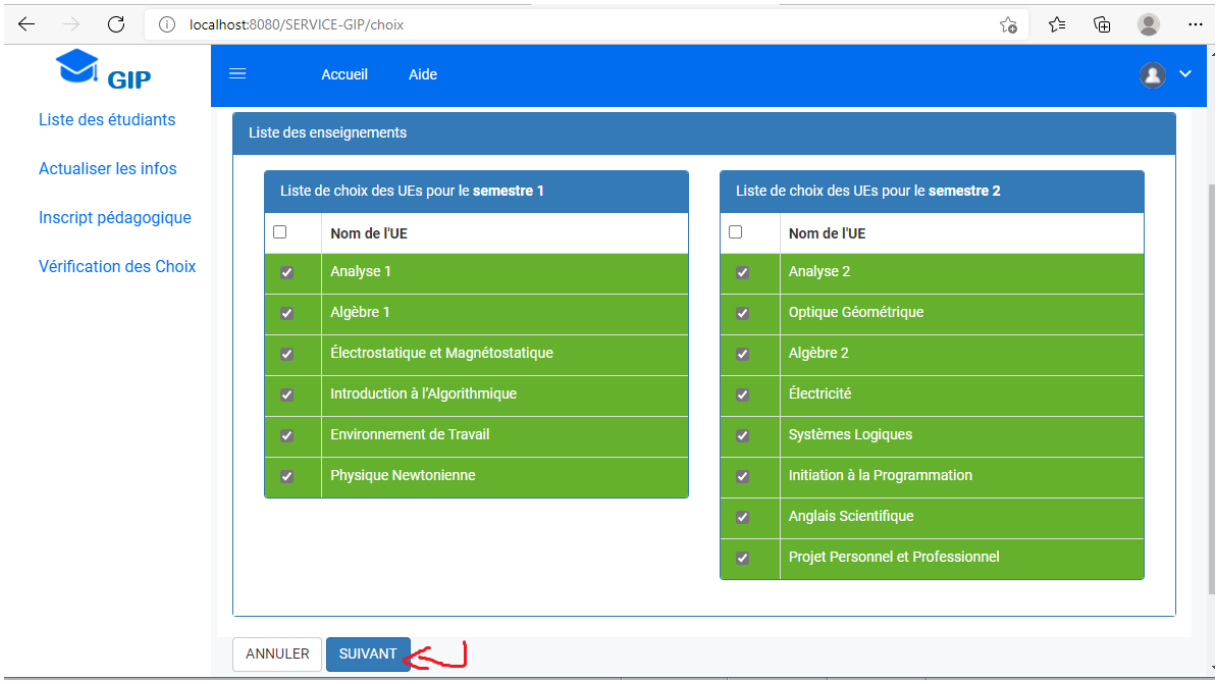


*Figure 5-17 : Liste des matières choisies*

Renseignement de la date d'inscription puis clique du bouton 


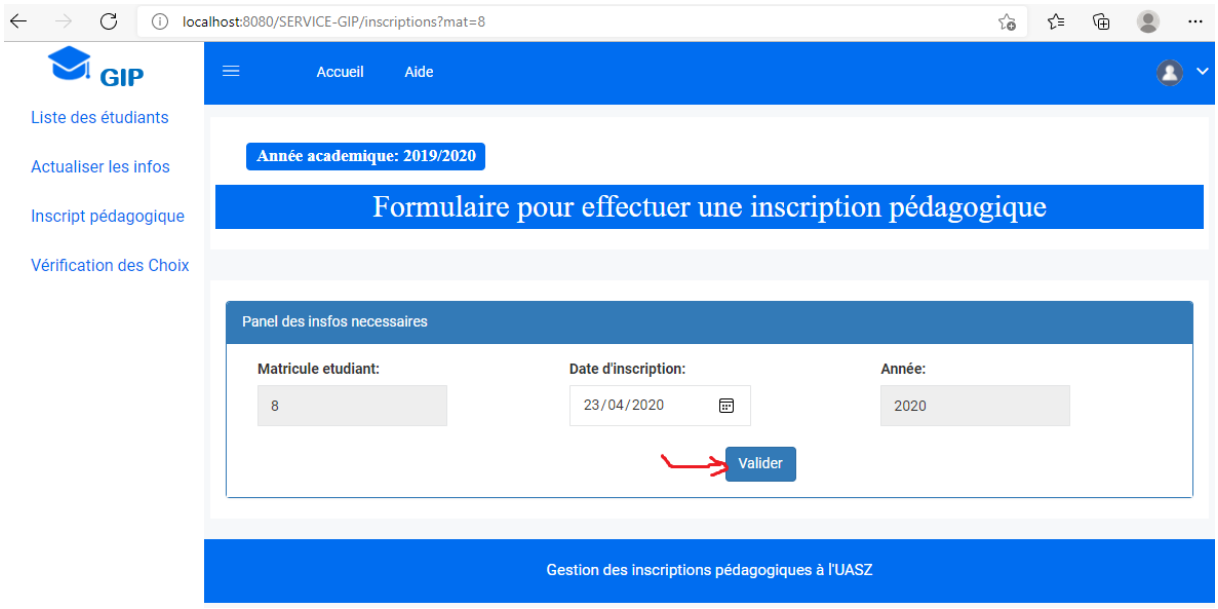


*Figure 5-18 : formulaire de validation d'une inscription*

Cette interface montre la fin du processus d'inscription avec la confirmation encadrée en rouge.

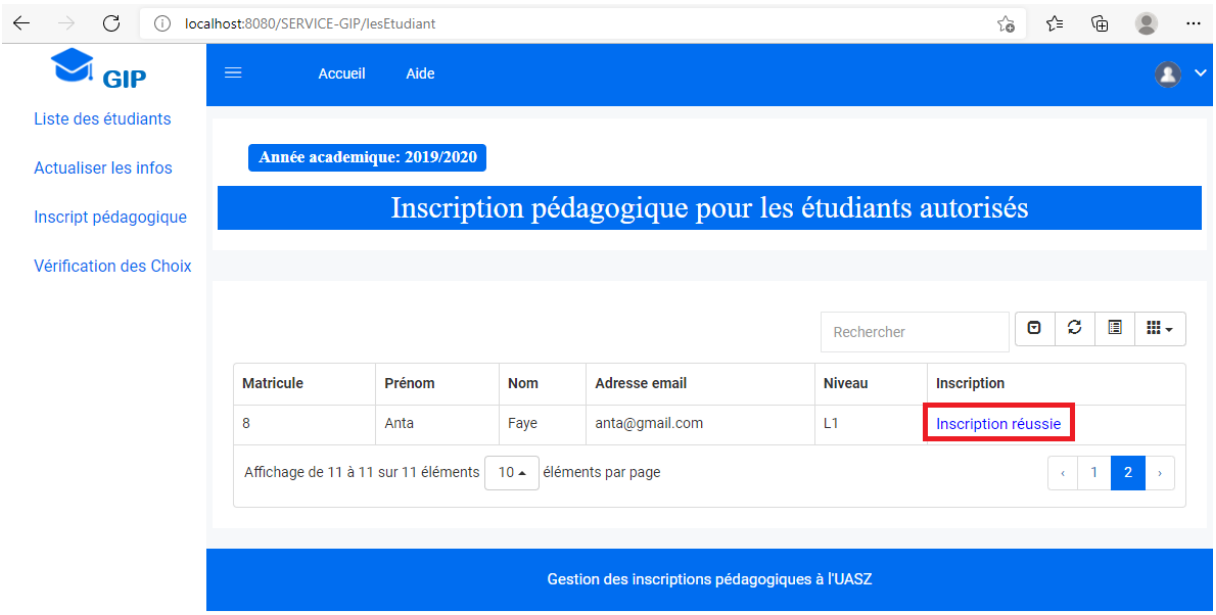


*Figure 5-19 : Résultat obtenu après inscription d'un étudiant*

### 5.2.2.4 Vérification de choix d'un étudiant

Clique du bouton 

.Cette fois-ci aucun micro-services n'est sollicité, cette requête appelle juste une fonctionnalité au sein du méme micro-service.

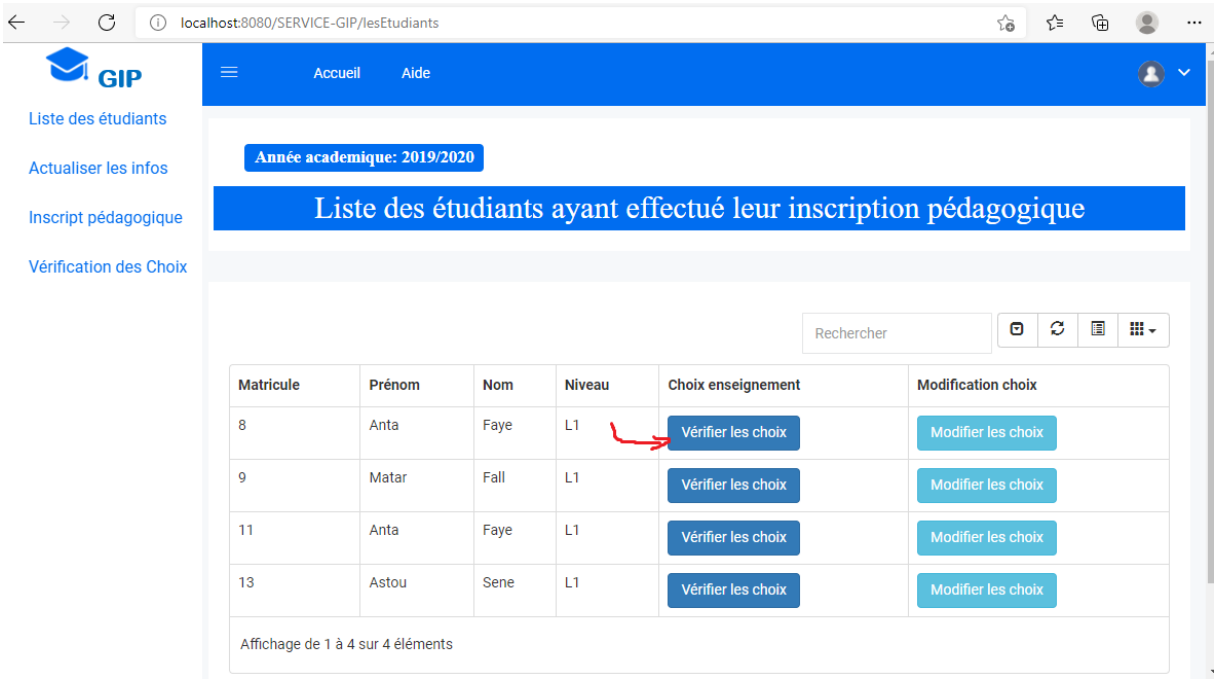


*Figure 5-20 : Interface de vérification et de modification de choix*

Le résultat de la requête est représenté dans cette interface.

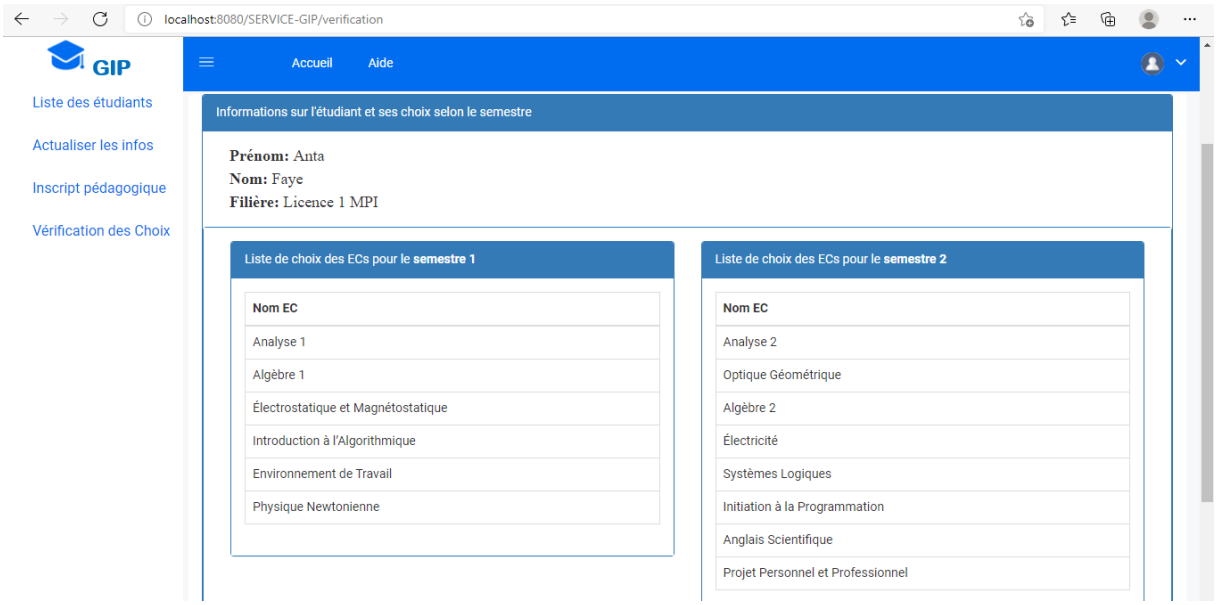


*Figure 5-21 : Résultat obtenu après vérification de choix*

### 5.2.2.5 Invocation de service par le micro-service « service-examens »

Une fois que l'inscription pédagogique terminé, le micro-service « ***service-examen*** » responsable de la gestion des évaluations peut contacter le micro-service « ***service-gip*** » pour avoir la liste d'émargement des étudiant à chaque fois qu'une évaluation est programmée.
Les captures suivantes avec l'indication de la flèche rouge décrivent le processus d'une planification d'une évaluation et l'obtention de la liste des étudiants qui doivent subir cette évaluation.

Après son authentification l'utilisateur clique sur le lien intitulé « **examen** » comme le montre la figure 5-22


*Figure 5-22 : Interface d'accueil de service-examen*

Clique du bouton 

et remplissage du formulaire. Ici le micro-service « service-info-ped » est solliciter pour choisir une matière selon la filière et le niveau.


*Figure 5-23 : Interface de planification d'une évaluation*

Remplissage du formulaire puis clique du bouton 


*Figure 5-24 : Formulaire de planification d'une évaluation*

Clique du bouton 

. Ici le micro-service « **service-gip** » est sollicité afin d'obtenir la liste d'émargement des étudiants.

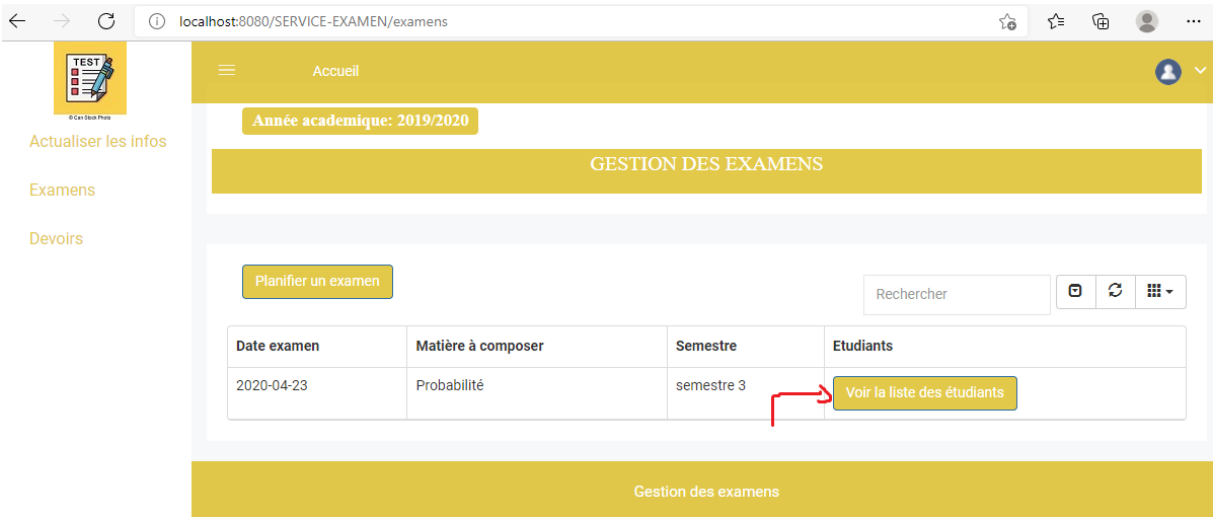


*Figure 5-25 : Liste des évaluations programmées*

Résultat obtenu pour voir la liste des étudiants concernés par cette évaluation. Ici la couleur bleue montre que les données sont fournies par le micro-service « **service-gip** ».

La figure 5-26 ci-dessous nous montre le résultat obtenu.


*Figure 5-26 : Résultat obtenu après consultation*

À travers ce dernier chapitre, qui a bouclé notre travail, nous avons commencé par faire la présentation de nos prototypes micro-services avant de détailler sur leur fonctionnement. Nous avons présenté une description des fonctionnalités principales de quelques micro-services en utilisant des captures d'écrans pour leur illustration.

# CONCLUSION GENERALE

Le travail qui nous a été confié portait sur une architecture basée les micro-services pour la dématérialisation des procédures et des documents administratifs pour le cas de l'UASZ. Cette idée de dématérialisation avait été portée par certaines autorités de l'université notamment celles de la DISI, des enseignant du département informatique et la DRH. En effet, depuis 2019 ils ont établi un projet [1] dont l'objectif est de faire la dématérialisation des procédures et des documents administratifs de cet institut.

Ceci pour permettre à l'université de résoudre les problématiques liées aux systèmes d'information dont elle dispose actuellement. Mais aussi d'anticiper les sur les problèmes de complexités que ces systèmes ne pourront pas gérer dans l'avenir à cause de leur architecture monolithique et l'impossibilité de partager mutuellement des données.

Pour permettre à l'université d'adapter son système d'information aux nouveaux systèmes modernes, nous avons été confiés à un travail dont l'objectif est de proposer une architecture évolutive. Permettant aussi aux composants de cette architecture de pouvoir communiquer et partager des données via des services qu'ils offrent mutuellement. Ceci afin de pouvoir répondre aux exigences des autorités qui ont porté le projet en vue de dématérialiser toutes procédures et documents administratifs de l'université.

Pour atteindre les objectifs de notre travail, nous avons divisé ce mémoire en cinq (5) grands chapitres. Le premier a été réservé à la description et au contexte justificatif du sujet. Dans le second chapitre nous avons fait l'état de l'art des micro-services. Nous avons consacré le troisième chapitre à la spécification des besoins fonctionnels. Dans le quatrième chapitre, nous avons détaillé sur la conception et l'implémentation des micro-services. Et le cinquième et dernier chapitre a été l'objet de la présentation et le fonctionnement de nos prototypes micro-services.

En perspectives nous envisageons dans l'avenir avec la collaboration des autorités universitaires :

- de mettre en application notre solution afin d'avoir un système global et performant pour l'UASZ ;

- de faire une refonte progressive faisant une transition de l'existence vers l'approche micro-services ;
- d'étendre nos prototypes pour réaliser les premiers micro-services du système global ;
- déployer les micro-services dans des conteneurs dédiés.

# BIBLIOGRAPHIE

[1] Youssou DIENG, Alousseynou FALL, Youssou FAYE, Serigne DIAGNE, Marie NDIAYE, Ibrahima DIOP, Khadim DRAME et Lamine FATY, « Projet de recherche et développement logiciel » rédigé en 2019.

[2] Université Assane SECK de Ziguinchor, AUF « Logiciel pour la gestion pédagogique »

[3] M. Lamine FATY, SICOMA « Automatisation de la comptabilité des matières à l'Université Assane SECK de Ziguinchor », mémoire de master 2 soutenu le 29/10/2014.

[4] Université Assane SECK de Ziguinchor, Manuel d'installation et d'utilisation de SAGESTOCK.

[5] M. Elhadji Mamadou Korka DIALLO, « Automatisation de la gestion des biens matériels de l'UFR des Sciences et Technologies », mémoire de master 2 soutenu le 29/03/2018.

[6] Photos prises par Dr. Dame DIOP de l'UFR LASHU.

[7] Université Assane SECK de Ziguinchor, Manuel de procédure de l'Université Assane SECK de Ziguinchor.

[8] Université Assane SECK de Ziguinchor, GESBUDGET « logiciel de gestion des biens financiers ».

[9] Université Assane SECK de Ziguinchor, COCKTAIL « Système de gestion des inscriptions académiques des étudiants »

[10] Université Assane SECK de Ziguinchor, SCHOOLPEDAG « Système de gestion des inscriptions pédagogiques des étudiants »

[11] Université Assane SECK de Ziguinchor, « Système de gestion des emplois du temps »

[12] Université Assane SECK de Ziguinchor, GRH-Paie « Logiciel de gestion de paiement des salaires »

[13] M. Camir MALACK, « Automatisation de la gestion des salaires et dématérialisation du bulletin de salaire du personnel de l'UASZ », mémoire de master 2 soutenu le 29/03/2018.

[14] M. Cheikh Souleymane BADIANE, « Conception et Développement d'une application informatique pour l'automatisation de la gestion des congés au sein de la DRH de l'UASZ et la

dématérialisation des documents administratifs qui s'y rapportent », mémoire de master 2 soutenu le 29/03/2018.

[15] UASZ, M. Henry DIALLO « Conception et Développement d'une application pour la gestion des répartitions d'enseignement à l'UASZ » mémoire de master soutenu le 07/11/2018.

# WEBOGRAPHIE

[16] https://www.redhat.com/fr/topics/cloud-native-apps/what-is-an-application-architecture consulté le 12/05/2019.

[17] https://www.expertsolutions.com/microservices-et-securite/ consulté le 09/05/2018.

[18] GUEDIRI Mossaab HAMIED Ali, « Middleware Orienté Micro-service pour les Applications Internet des Objets» disponible sur le site : https://www.expertsolutions.com/microservices-et-securite/ consulté le 09/05/2018.

[19] https://aws.amazon.com/fr/microservices/ consulté le 08/07/2019.

[20] https://crc.re/5wgg8/6ea3eb-architecture-d%27un-r%C3%A8seau-informatique-definition consulté le 08/07/2019.

[21] https://123dok.net/document/6zkg5opq-middleware-oriente-microservice-pour-les-applications-internet-des-objets.html consulté le 09/05/2018.

[22] https://www.okta.com/fr/blog/2021/02/microservices/ consulté le 16/02/2021

[23] https://azure.microsoft.com/en-us/campaigns/cloud-application-architecture-guide/ consulté le 15/10/2021.

[24] https://123dok.net/document/6zkg5opq-middleware-oriente-microservice-pour-les-applications-internet-des-objets.html consulté le 09/05/2018.

[25] https://www.okta.com/fr/blog/2021/02/microservices/ consulté le 16/02/2021.

[26] https://www.softfluent.fr/blog/microservices-api-gateway/ consulté le 29/04/2021.

[27] https://www.dineshonjava.com/microservices-inter-service-communication/ consulté le 29/04/2021

[28] https://blog.ippon.fr/2015/10/16/rex-architecture-orientee-microservices-avec-netflix-oss-et-spring-article-2/ consulté le 19/06/2021

[29] https://openclassrooms.com/fr/courses/4668216-optimisez-votre-architecture-microservices/5176263-faites-communiquer-vos-microservices-grace-a-feign#:~:text=Feign%20est%20un%20client%20HTTP,%C3%A9quivalent%20en%20code%20de%20Postman consulté le 12/12/2020.

.