RÉPUBLIQUE DU BÉNIN
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ D'ABOMEY-CALAVI

INSTITUT DE FORMATION ET DE RECHERCHE EN INFORMATIQUE

BP 526 Cotonou Tel : +229 55 028 070/ 53 973 060
https://www.ifri-uac.bj Courriel : contact@ifri.uac.bj

MÉMOIRE

pour l'obtention du

Diplôme de Licence en Informatique

Option : Génie Logiciel

Présenté par :
Stanislas Fructueux **HOUETO**

Projet de mise en place de l'Espace Numérique de Travail de l'Université d'Abomey Calavi (UAC) : Cas de la gestion académique des entités.

Sous la supervision :
Ing Pierre Jérôme **ZOHOU**

AIZAN Josky	Docteur	IFRI	Président
BOCO Charbel	Ingénieur	IFRI	Examinateur
ZOHOU Pierre Jérôme	Ingénieur	IFRI	Rapporteur

Année Académique : **2022-2023**

Sommaire

Dédicace ii

Remerciements iii

Résumé iv

Abstract v

Liste des figures vi

Acronymes vii

Glossaire viii

Introduction 1

1 Revue de littérature. 3

2 Méthode de conception et choix techniques 12

3 Résultat et discussion 25

Conclusion 31

Références bibliographiques 32

Références webographiques 33

Table des matières 34

Dédicace

À Dieu tout puissant.
À tous les membres de ma famille.

Remerciements

J'addresse mes sincères remerciements :

- Au Professeur Eugène C. EZIN , Directeur de l'Institut de Formation et de Recherche en Informatique (IFRI) ;
- Au Dr Gaston EDAH, Directeur Adjoint de l'Institut de Formation et de Recherche en Informatique (IFRI) ;
- À Monsieur Pierre Jérôme ZOHOU, pour m'avoir suivi tout au long de ce travail en dépit de ses multiples responsabilités, pour sa générosité et sa patience ;
- À tous les membres de l'administration et du corps professoral de l'IFRI pour l'accompagnement et leur formation inestimable ;
- Mr Ignace AZONHOUMON, mon maître de stage pour sa disponibilité et ses conseils dans la réalisation de ce travail.

Je remercie également les membres de jury qui ont accepté juger mon travail.

Résumé

La modernisation des systèmes académiques est devenue une nécessité incontournable, notamment au sein de l'Université d'Abomey-Calavi. Face à la complexité croissante de la gestion académique des entités, une problématique majeure émerge quant à la nécessité d'une plateforme numérique efficace et intégrée.

Dans ce contexte, notre solution repose sur la conception et l'implémentation d'une plateforme spécifiquement dédiée à la gestion académique des entités universitaires. Cette solution, basée sur une application web, utilise une architecture microservice pour assurer une interaction fluide avec les différents modules de l'université.

La réalisation de ce projet a suivi une approche méthodologique rigoureuse. Le développement du frontend a été effectué avec React JS, offrant une interface utilisateur intuitive et réactive. Le backend, construit avec Flask, garantit la robustesse et la sécurité des opérations. La modélisation a été réalisée en utilisant UML, assurant ainsi une compréhension claire de l'ensemble du système. L'application, frontend et backend, a été conteneurisée avec Docker pour une gestion simplifiée et une déploiement efficace. Les fonctionnalités principales concernent entre autre la gestion des performances académiques, les bilans et statistiques et d'avoir une vue d'ensemble du parcours académique des étudiants.

L'aboutissement de ce projet se traduit par une application web fonctionnelle, opérant de manière optimale en utilisant une base de données Postgres. L'architecture microservice permet une intégration harmonieuse avec les autres modules de l'ENT, offrant ainsi une solution complète et efficiente à la gestion académique des entités de l'Université d'Abomey-Calavi.

Mots clés : Espace numérique ; Gestion académique ; Application web ; Architecture Microservice.

Abstract

The modernization of academic systems has become an unavoidable necessity, especially within the University of Abomey-Calavi. Faced with the increasing complexity of academic entity management, a major issue arises regarding the need for an efficient and integrated digital platform.

In this context, our solution is based on the design and implementation of a web application specifically dedicated to academic entity management within the university. This solution, built as a web application, utilizes a microservices architecture to ensure smooth interaction with various university modules.

The project followed a rigorous methodological approach. The frontend development was carried out using React JS, providing an intuitive and responsive user interface. The backend, built with Flask, ensures robustness and security in operations. UML was used for clear system understanding. The application, both frontend and backend, was containerized with Docker for simplified management and efficient deployment. The main features include, among others, managing academic performance, generating reports and statistics, and having an overview of students academic progress.

The culmination of this project is a functional web application that operates optimally using a Postgres database. The microservices architecture allows seamless integration with other modules of the digital workspace, offering a comprehensive and efficient solution to academic entity management at the University of Abomey-Calavi.

Keywords:Digital Workspace ; Academic Management ; Web application ; Microservices.

Liste des figures

1.1 Corrélation entre les microservices de l'ENT 4
1.2 ERP Aimaira : Informations étudiant 7
1.3 ERP Aimara : Portail d'inscription côté administration 7
1.4 La suite logiciel GestStudent 8
1.5 Plateforme de gestion des notes de la FADESP : connexion 9
1.6 Interface d'ajout d'UE et d'ECUE 10
1.7 Interface d'insertion de note 10

2.1 Architecture Microservice 13
2.2 Diagramme de classe 14
2.3 Diagramme de cas d'utilisation 15
2.4 Diagramme de séquence du cas d'utilisation : importer notes étudiants. 16
2.5 Diagramme de séquence du cas d'utilisation : rechercher bulletin 17
2.6 Logo de docker. 18
2.7 interface du client d'API *Insomnia* 23

3.1 Interface de connexion au module 25
3.2 Dashboard principal 26
3.3 Page récapitulant les informations relatives à un enseignant. 27
3.4 Importation des notes 27
3.5 Formulaire d'inscription d'un nouvel enseignant dans le système. 28
3.6 Documentation swagger de l'API 29

Acronymes

API : Application Programming Interface , 12, 13, 22, 23

ECUE : Élément Constitutif d'Unité d'ensignement , 8, 9

ENT : Espace Numérique de Travail vi, , 3, 4, 34

ERP : Enterprise Resource Planning , 6

HTTP : HyperText Transfer Protocol , 22, 23

JSON : JavaScript Object Notation , 13

LDAP : Lightweight Directory Access Protocol , 29

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

UAC : Université d'Abomey-Calavi , 2, 9, 30

UE : Unité d'Enseignement , 8, 9

UML : Unified Modeling Language , 14–16, 34, 35

Glossaire

API RESTful : "Representational State Transfer" ; c'est un style d'architecture logicielle pour la conception de services web qui favorise la simplicité, la performance, la scalabilité et la réutilisabilité des services. Les API RESTful reposent sur un ensemble de principes et de contraintes qui définissent comment les ressources (par exemple, des données ou des fonctionnalités) doivent être exposées et manipulées via des requêtes HTTP. , 21

Bibliothèque : C'est un ensemble de fonctions, de classes, de modules ou de code réutilisable préécrit qui peut être utilisé par les développeurs pour effectuer des tâches courantes sans avoir à réinventer la roue. Les bibliothèques sont conçues pour simplifier le développement en fournissant une série de fonctionnalités prêtes à l'emploi, ce qui permet d'accélérer le processus de programmation et de réduire la complexité du code. On parle aussi de *librairie*.

Framework : C'est une structure logicielle générique qui fournit un ensemble de fonctionnalités et d'outils prêts à l'emploi pour faciliter le développement d'applications logicielles. Les frameworks sont conçus pour accélérer le processus de développement en offrant une base solide sur laquelle les développeurs peuvent construire leurs applications. , 24

JavaScript : JavaScript est un langage de programmation interprété et de haut niveau utilisé pour rendre interactif et dynamique les pages web. JavaScript est un langage polyvalent qui peut être utilisé à la fois côté client (dans le navigateur) et côté serveur (avec des environnements tels que Node.js).

Logiciel ERP : Enterprise Resource Planning (Progiciel de Gestion Intégré): c'est un logiciel qui permet à une organisation de gérer et d'automatiser l'ensemble de ses opérations et processus commerciaux et métier au sein d'un système intégré. Les logiciels ERP sont conçus pour aider les entreprises à optimiser leur efficacité, à rationaliser leurs opérations, à améliorer leur prise de décision et à favoriser la collaboration interne

Introduction Générale

La digitalisation est devenue un élément essentiel dans notre société moderne. Elle a un impact significatif dans de nombreux domaines de la vie quotidienne, de l'économie à la santé en passant par l'éducation et la communication. Les processus automatisés et numérisés permettent d'accomplir des tâches plus rapidement et de manière plus efficace. Cela peut réduire les coûts opérationnels et augmenter la productivité du personnel administratif.

Contexte et justification

L'Espace Numérique de Travail (ENT) de l'Université d'Abomey-Calavi (UAC) représente une plateforme web ambitieuse visant à réunir divers outils et services numériques dédiés à l'éducation et à la gestion universitaire. Son objectif principal est de centraliser l'accès à un ensemble crucial de ressources, incluant les cours, les inscriptions, les emplois du temps, les informations administratives, les résultats d'examens, ainsi que d'autres fonctionnalités essentielles de la vie universitaire.

Ce projet dans le cadre de ses activités, intègre un module dédié à la gestion académique des entités. Les entités de l'UAC sont confrontées à des défis de gestion liés à leurs activités académiques, englobant la gestion des performances académiques, des bulletins de notes, de l'emploi du temps, des offres de formations, des délibérations et statistiques, etc. Ces défis sont amplifiés par l'augmentation constante du nombre d'étudiants dans les facultés, écoles et instituts. Ainsi, les données générées par ces activités académiques doivent être partagées avec d'autres modules pour garantir une interopérabilité optimale.

Problématique

Au sein de l'Université d'Abomey-Calavi (UAC), la coexistence de multiples entités académiques, chacune délivrant des cours sous le régime du système LMD, pose un défi substantiel. Chaque entité se voit investir de la responsabilité cruciale de gérer efficacement ses données académiques.

Le problème émerge lorsque l'on considère la diversité des données et l'absence d'une structure logicielle intégrée au niveau de la scolarité. Cette lacune crée des obstacles significatifs pour assurer une uniformité dans la gestion des activités académiques. Les répercussions de cette difficulté s'étendent au-delà de la simple organisation des données, influant sur la qualité des services, la prise de décision, et ultimement sur la satisfaction des acteurs clés tels que les étudiants et le personnel administratif. Comment, alors, développer une solution capable de surmonter ces défis et garantir une gestion cohérente et efficiente des activités académiques au sein de chaque entité ? C'est la question fondamentale à laquelle ce travail s'efforcera de répondre.

Objectifs

L'objectif principal de ce travail consiste à mettre en place une plateforme web de gestion académique des entités de l'Université d'Abomey-Calaviuac. Elle doit être performante et efficace à travers une interface graphique intuitive et agréable. De façon concrète, il s'agira :

- de gérer les performances académiques des étudiants ;
- de permettre de faire les bilans des années académiques ;
- d'organiser les parcours des étudiants au sein du système ;
- de rendre disponible et interopérable le module.

Organisation du mémoire

Le présent travail est organisé en trois chapitres. Le premier chapitre fait un rappel des connaissances ainsi qu'une étude des solutions existantes liées à notre thème. Le deuxième chapitre expose les méthodes employées pour la conception de notre solution ainsi que les outils liés à sa réalisation. Le troisième et dernier chapitre examine les résultats obtenus, présente les interfaces de notre solution, et discute des éventuelles insuffisances liées à notre solution.

Chapitre 1

Revue de littérature.

Introduction

Dans le présent chapitre, nous faisons une étude de l'existant afin de mieux appréhender le problème auquel nous voulons apporter de solution. Ce qui nous permettra de ressortir les insuffisances et de formuler des hypothèses de recherches.

1.1 Présentation de l'architecture globale de l'Espace Numérique de Travail.

Le projet ENT-UAC adopte une architecture microservice, caractérisée par une découpe réfléchie de l'application en petits services autonomes. Chaque microservice est minutieusement conçu pour opérer de manière indépendante tout en interagissant avec les autres services afin de fournir une fonctionnalité globale. Cette approche a conduit à la création de six microservices distincts, à savoir :

- La Scolarité Centrale ;
- L'Espace Étudiant ;
- Le Module de Pré-inscription ;
- Le Module d'Administration ;
- Le Microservice des Commissions ;
- Le Module de Gestion des Entités.

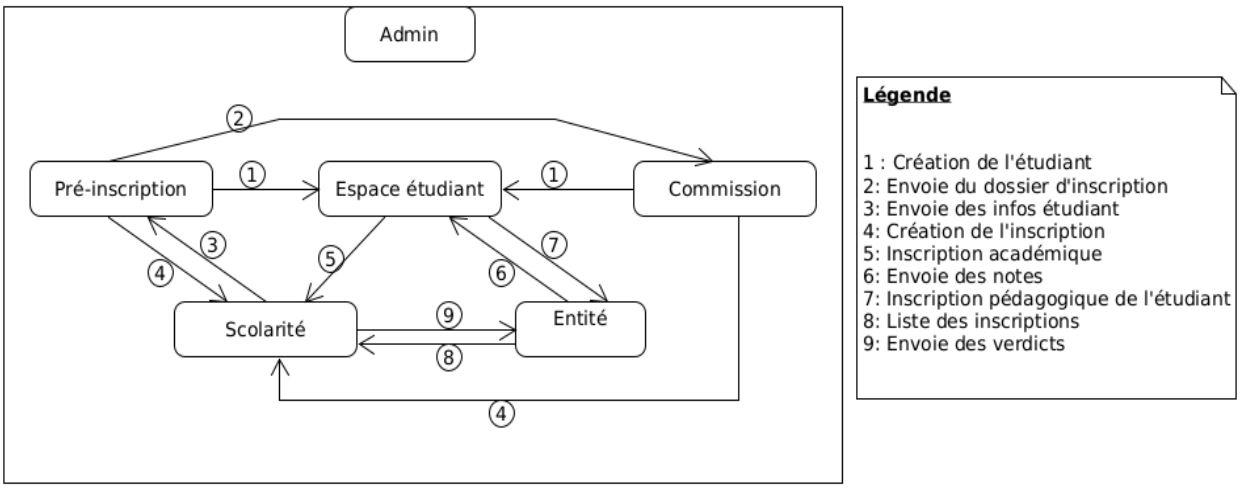


FIGURE 1.1 : Corrélation entre les microservices de l'ENT

1.2 Les entités académiques

1.2.1 Définition et clarification.

Une entité à l'université d'Abomey-Calavi est une structure sous la tutelle rectorale, disposant d'une offre de formation et qui dispense des cours et formations dans le cadre de ces offres. Nous avons ainsi donc les écoles de formations, les facultés de formation et les instituts de formation [4]. Un diplôme en fin de formation est délivré aux étudiants suivant le cursus de formation dans l'une de ces entités.

Parmi celles existantes on peut citer comme entité académique :

- L'École Polytechnique d'Abomey-Calavi ;
- L'Institut de Formation et de Recherche en Informatique.
- La Faculté des Sciences Agronomiques ;
- La Faculté des Sciences Économiques et de Gestion ;
- La Faculté des Sciences Politiques ;
- L'École Nationale d'Administration et de Magistrature ;
- L'École Nationale d'Économie Appliquée et de Management ;
- ...

L'Université d'Abomey-Calavi offre des formations en : [5]

- Licence ;
- Master ;
- Doctorat ;
- Diplôme d'ingénieur.

1.2.2 Les activités de gestion dans les entités académiques

Dans les entités académiques de l'université d'Abomey-Calavi plusieurs activités sont menées par le personnel pour gérer les étudiants. Entre autres, nous avons :

- Le suivi du parcours de l'étudiant dans l'entité ;
- La gestion et le suivi des cours sans oublier les emplois du temps ;
- La gestion des notes et bulletins ;
- La gestion des enseignants et formateurs dans l'entité ;
- Les délibérations suites aux évaluations terminales ;
- ...

1.3 Application web

Une application web, également connue sous le nom d'application en ligne, est un logiciel interactif accessible via un navigateur web, qui permet aux utilisateurs d'accomplir diverses tâches, telles que la gestion de données, la communication, le commerce électronique, ou d'autres activités en ligne. Contrairement aux applications traditionnelles, les applications web ne nécessitent généralement aucune installation sur l'ordinateur de l'utilisateur, car elles s'exécutent entièrement à partir du serveur distant, offrant ainsi une accessibilité et une flexibilité accrues.

1.3.1 Composition

Généralement les applications web sont composées d'une partie **frontend** et d'une partie **backend**.

1.3.1.1 Frontend

Le frontend représente le rendu visuel avec lequel l'utilisateur est en contact pour avoir une certaine expérience et une certaine interaction dans l'utilisation des fonctionnalités du logiciel. Il doit donc garantir une bonne ergonomie, être intuitif et s'adapter aux formats des écrans des terminaux qui l'utilisent.

1.3.1.2 Backend

La notion du backend fait référence à la logique interne du logiciel. C'est la partie invisible d'une application web ou d'un logiciel qui gère les aspects techniques tels que le stockage des données, le traitement des requêtes de l'utilisateur, la sécurité et la gestion des bases de données. Le backend communique avec le frontend en fournissant les données nécessaires et en exécutant des opérations en réponse aux demandes de l'utilisateur. Il est souvent développé en utilisant des langages de programmation tels que Python, Java, PHP ou Ruby, et il repose sur des serveurs et des bases de données pour fonctionner efficacement.

1.4 Microservice

Un microservice est une architecture de développement logiciel dans laquelle une application complexe est décomposée en plusieurs services logiciels autonomes et interopérables, appelés 'microservice'. Chaque microservice est conçu pour accomplir une tâche spécifique et unique, et il communique avec d'autres microservices au moyen d'APIs bien définies.

C'est cette architecture logicielle qui est adoptée pour l'ensemble du projet de l'espace numérique de l'université d'Abomey-Calavi. Ainsi, le module de gestion académique des entités représente un microservice qui interagit avec d'autres microservices du système (scolarité, espace étudiant, administration, etc).

1.5 Les solutions existantes

Dans l'optique de gérer les activités d'ordre académiques des entités dans le système LMD, plusieurs logiciels ont été développés. Ces solutions proposent des approches de résolutions sous forme d'application de bureau, d'applications mobiles ou d'ERP.

1.5.1 Aimaira ERP

C'est un logiciel qui se définit comme facilitateur des activités liées aux processus métiers dans l'enseignement supérieur spécialement dans le privé. L'ERP est en mode SaaS (Software as a Service), ce qui permet de maîtriser son coût. [6] Ce logiciel gère entre autre :

- La comptabilité de l'établissement ;
- La formation (Examens, bulletins, cours, ...)
- Le recrutement ;
- Les relation extérieurs (Parents, entreprises, partenaires, ...) ;
- ...

L'ERP aimaira est notemment utilisé dans les universités Européennes notamment en France. C'est un ERP qui est déployé en fonction du module de l'ERP auquel l'université partenaire souscrit.

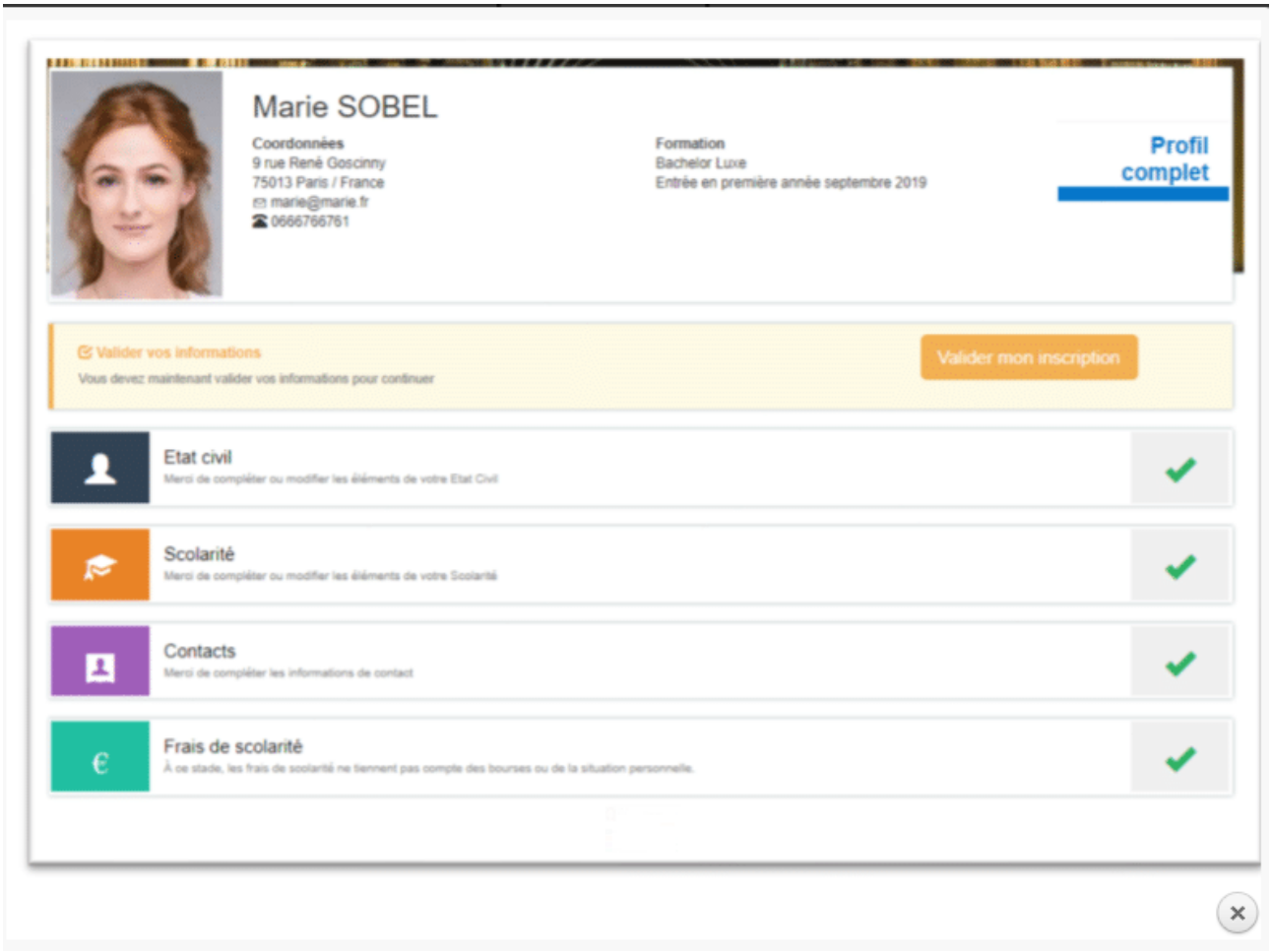


FIGURE 1.2 : ERP Aimaira : Informations étudiant

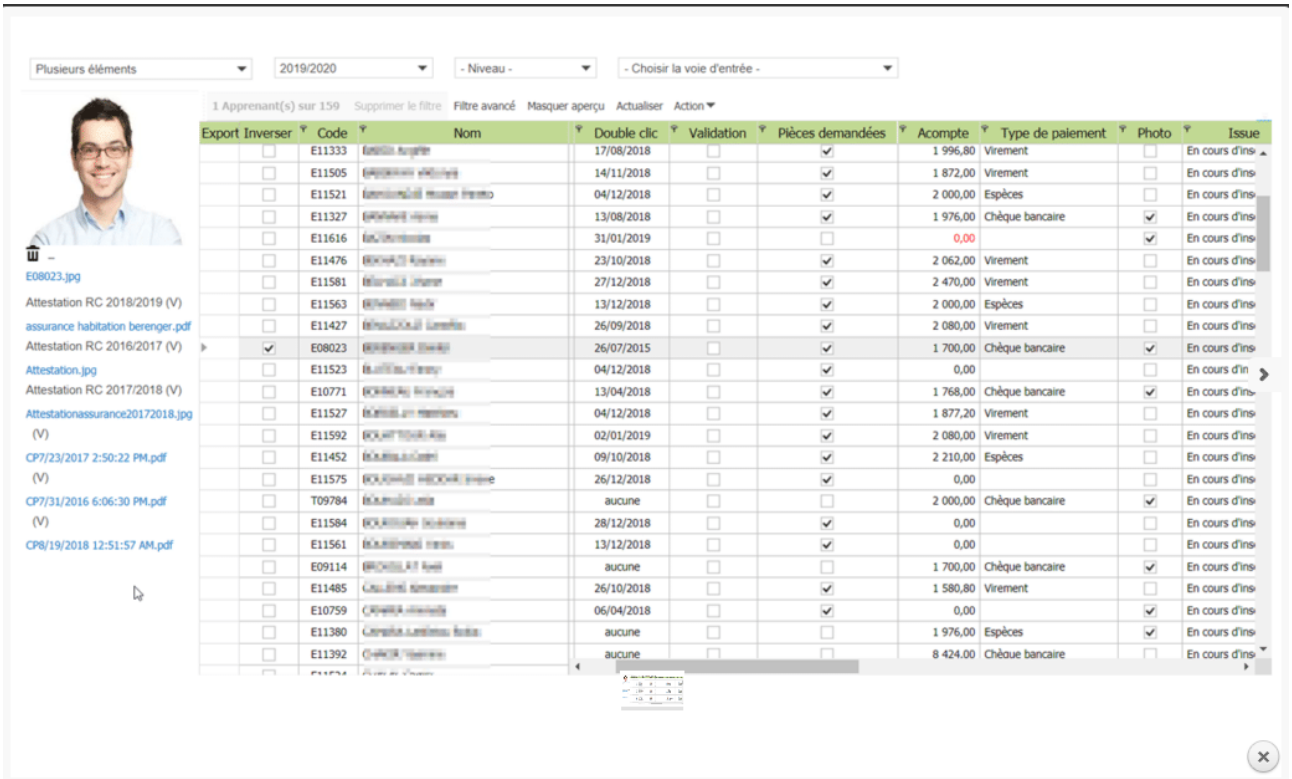


FIGURE 1.3 : ERP Aimara : Portail d'inscription côté administration

1.5.2 GestStudent

GestStudent est un progiciel de gestion dédié au système LMD. Il a été développé par IITECH Bénin, une Société à Responsabilité Limité (SARL) implantée en Rébublique du Bénin. L'entreprise propose un logiciel disponible en version desktop (bureau), mobile et web disposant de plusieurs fonctionnalités, dont les principales sont :

- La pré-inscription et la validation des inscriptions des étudiants ;
- La génération des cartes d'étudiants et relevé de notes ;
- Les examens en ligne ;
- La gestion des notes, UE et ECUE ;
- La proclamation des résultats académiques consultables en ligne.

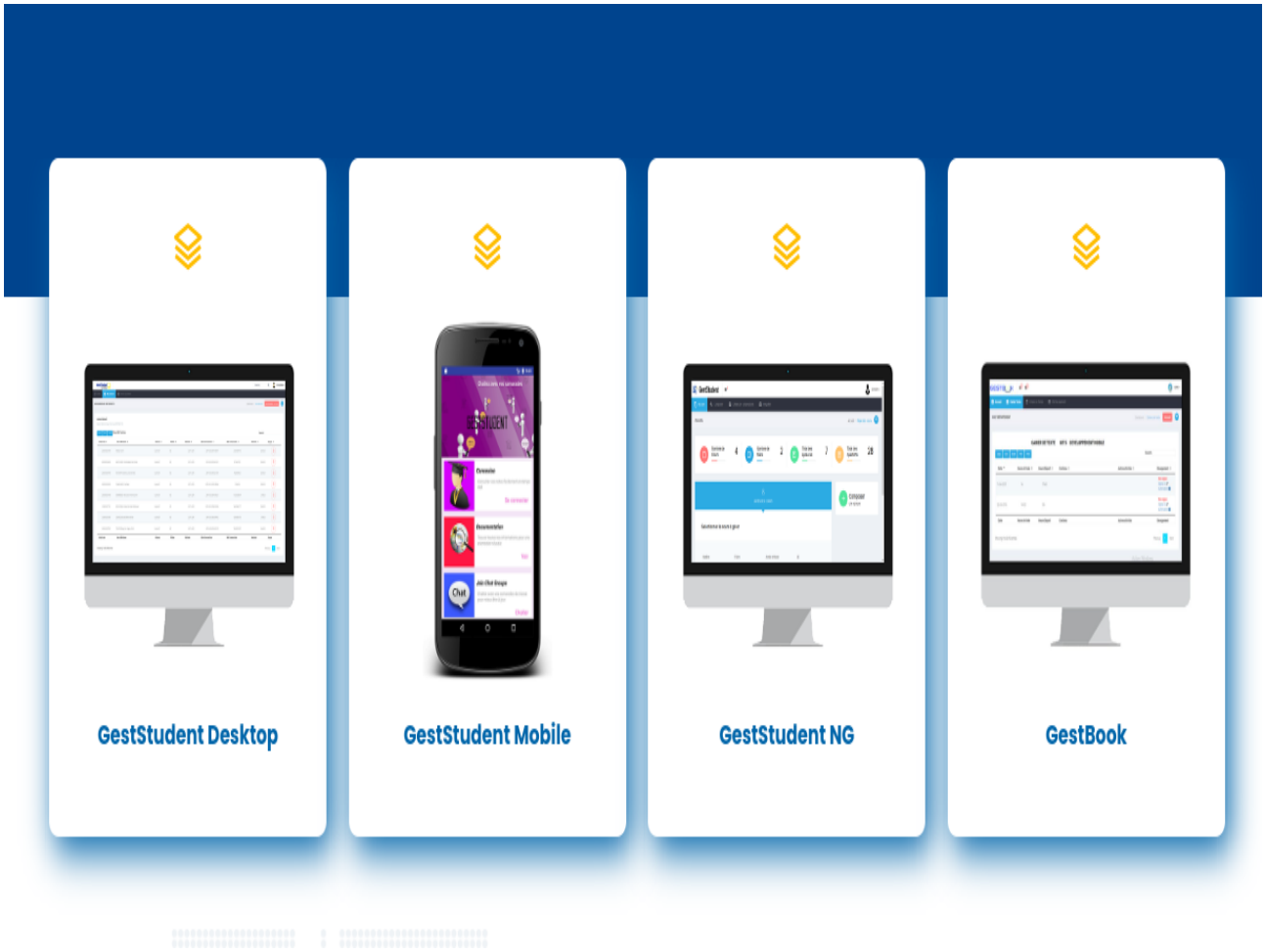


FIGURE 1.4 : La suite logiciel GestStudent

Le progiciel est séparé en plusieurs modules qui sont :

- GestStudent Administration : pour la configuration générale du progiciel ;
- GestStudent Validation : pour la validation des inscriptions ;

- GestStudent NG : pour les évaluations en ligne ;
- GestStudent Resultat : pour la gestion des notes, délibération et résultat.

La version mobile est disponible sur les appareils Android. Il prend en charge une interface pour l'administration globale du logiciel et une autre interface permettant aux étudiants de se connecter à leur espace sans frais (examens, inscription et ré-inscription, informations diverses). Ce logiciel est utilisé par les instituts comme l'Ecole Polytechnique d'Abomey-Calavi (EPAC), la Haute École de Commerce et de Management (HECM), et l'École Supérieur le Faucon (ESF-Faucon).

1.5.3 Egaster

Egaster est un logiciel de suivi de performance académique de la Faculté de Droit et de Science Politique (FADESP) de l'UAC. Il a été conçu pour permettre aux étudiants et aux parents d'étudiants de cette faculté de consulter aisément les notes des examens et aussi permettre au personnel de la faculté de gérer les processus académiques plus aisément.

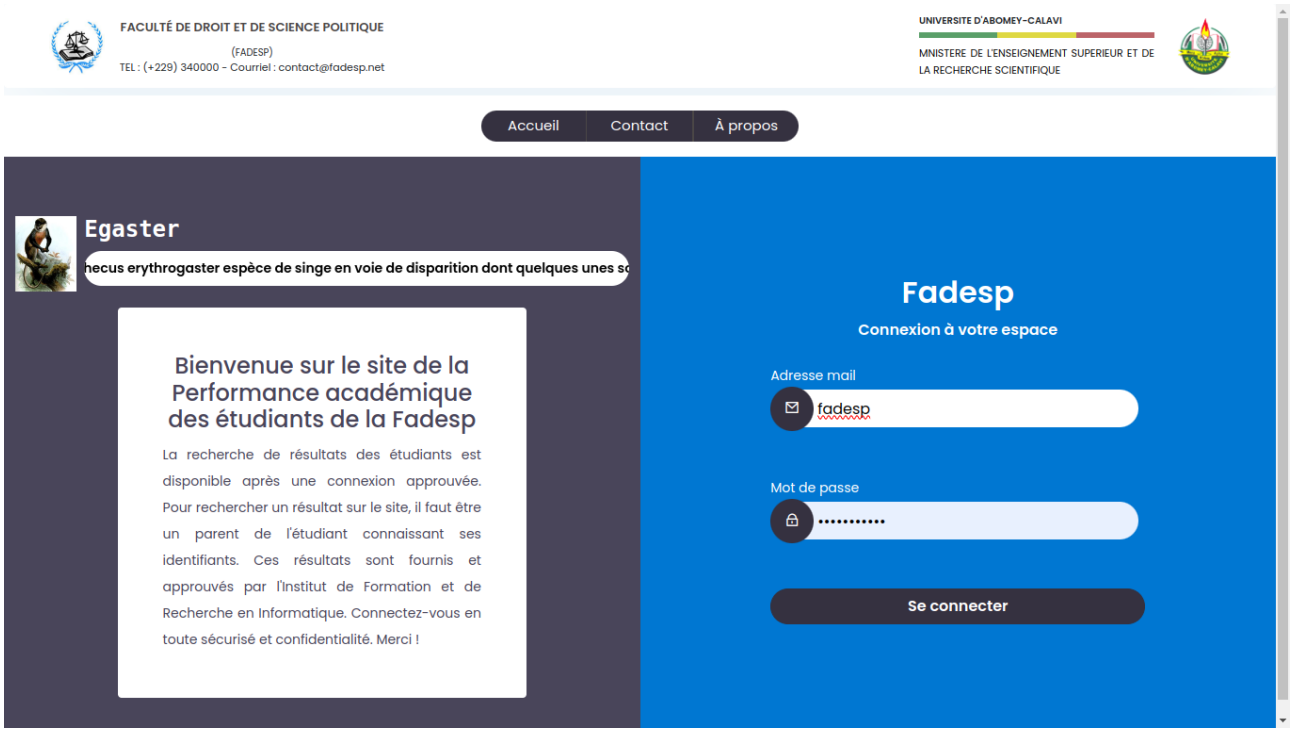


FIGURE 1.5 : Plateforme de gestion des notes de la FADESP : connexion

Fonctionnalités

Cet outils permet entre autre :

- d'ajouter des UE et ECUE ;
- d'enregistrer les notes des étudiants par ECUE ;
- de calculer la moyenne des étudiants par ECUE et par UE ;
- de produire le procès verbal de délibération de chaque année académique ;
- de gérer de formations, les années académiques, ...

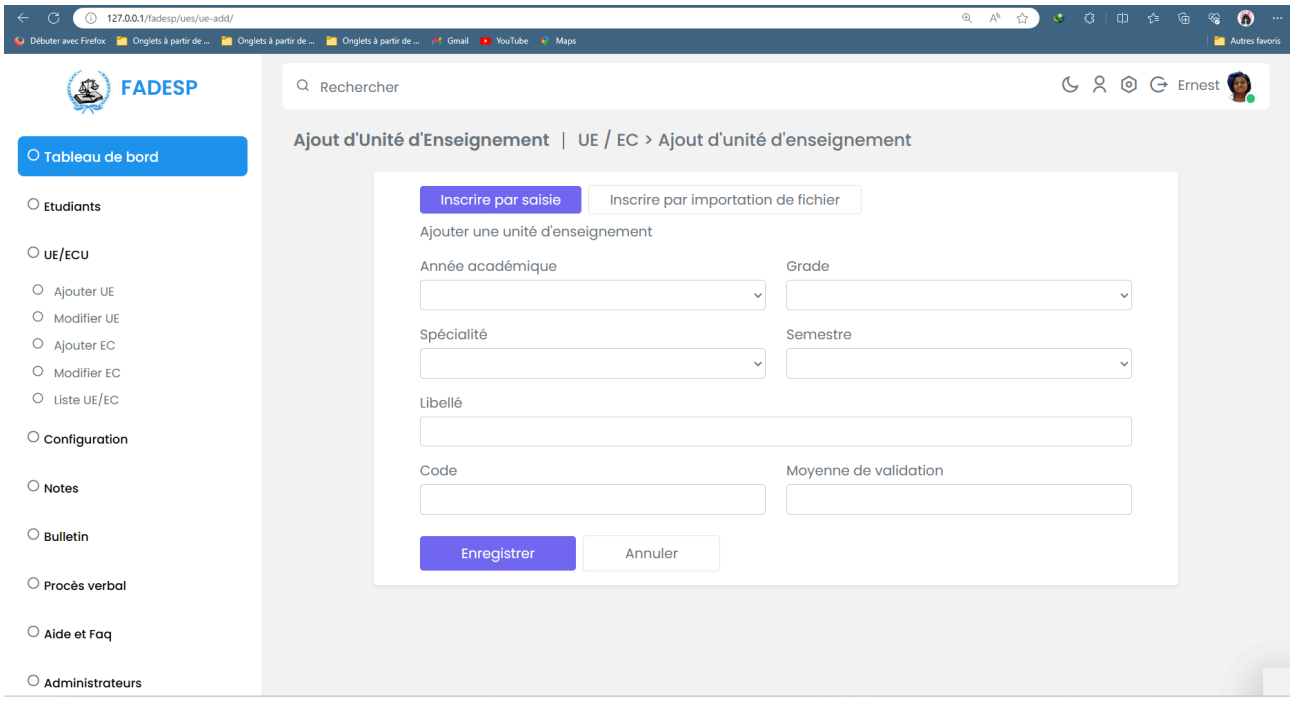


FIGURE 1.6 : Interface d'ajout d'UE et d'ECUE

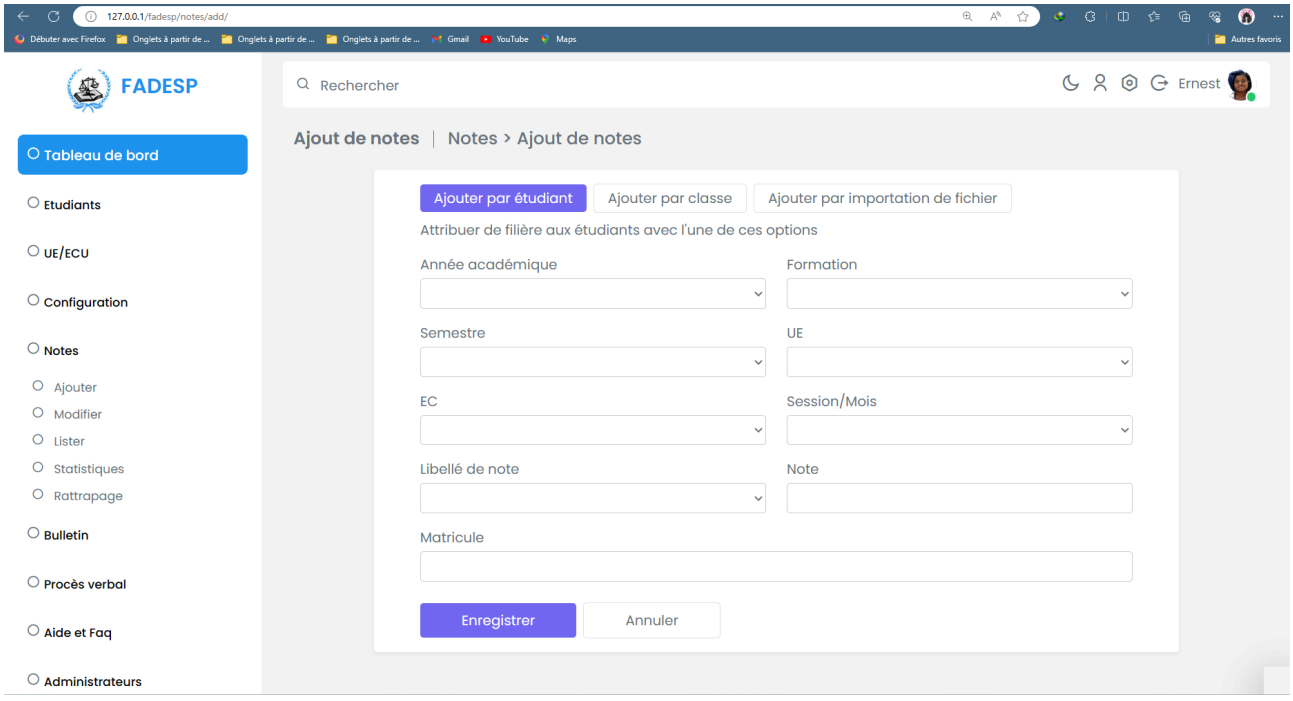


FIGURE 1.7 : Interface d'insertion de note

Ces solutions développées permettent de pallier à un problème de gestion des notes dans une entité. Mais notons que ces traitements ne sont fait que de façon isolée, ce qui ne permet pas à la scolarité centrale de disposer de certaines informations relatives au parcours des étudiants, leurs entités respectives ou encore de faire des statistiques à l'endroit des organismes partenaires ; car l'un des objectifs de ce projet de *l'espace numérique de travail* est de mettre en place un système hautement performant et interopérable.

1.6 Hypothèses de recherche

Les outils précédemment mentionnés ouvrent la voie à une solution qui s'inscrit parfaitement dans le cadre de la conception de l'Espace Numérique de Travail de l'Université d'Abomey-Calavi (UAC). Cette solution est conçue pour respecter les spécifications suivantes :

- Elle doit offrir une expérience utilisateur conviviale, intuitive et ergonomique.
- Elle doit être capable de gérer les activités internes liées au fonctionnement académique des entités.
- Elle doit être en mesure d'interagir avec les autres modules (microservices) de l'Espace Numérique de Travail de l'Université d'Abomey-Calavi.

Conclusion

Ce chapitre nous a permis de parcourir les solutions existantes utilisées pour la gestion académique des entités. Nous avons présenté les fonctionnalités importantes mises en oeuvres par ces logiciels. Le but de notre travail est de proposer un application web répondant au besoin de gestion académique des entités . Dans le prochain chapitre, nous présenterons notre méthode de conception et les choix techniques opérés.

Chapitre 2

Méthode de conception et choix techniques

Introduction

Dans le but d'atteindre les objectifs précédemment cités, une méthodologie et des choix techniques ont été opérés. En se basant sur les hypothèses de recherche évoquées dans le chapitre précédent, une application web a été produite, ce qui permettra de mieux prendre en charge les activités académiques qui incombent à chaque entité. Dans ce chapitre nous présentons la méthode de travail utilisée pour exécuter le projet, l'architecture, les outils et techniques utilisés et présentons son fonctionnement général.

2.1 Architecture

Généralité sur les microservices

L'architecture microservice est un modèle d'architecture logicielle qui repose sur le découpage d'une application en petits services autonomes, indépendants les uns des autres. Chaque service est responsable d'une fonctionnalité ou d'une tâche spécifique de l'application et communique avec d'autres services au moyen d'API bien définies. Ce modèle favorise la flexibilité, la scalabilité et la facilité de déploiement des applications. [9]

Les microservices communiquent généralement entre eux via des protocoles tels que HTTP (HyperText Transfer Protocol) ou le HTTPS (HTTP Secure). Les microservices permettent une flexibilité, une évolutivité et une facilité de déploiement améliorées, car chaque service peut être développé, testé, déployé et mis à jour indépendamment. [2] Cette architecture favorise également une meilleure résilience et une répartition des charges efficace, ce qui en fait une approche populaire et robuste pour le développement d'applications modernes.

L'architecture en microservice est largement adoptée pour la conception de l'ensemble du système de notre Espace Numérique de Travail. Chaque module est vue comme une unité indépendante communiquant par API avec les autres. Un service sert d'API Gateway, exposé au client, qui va permettre de prendre chaque requête et faire une redirection vers le service approprié en interne et ensuite de retourner le résultat au client concerné.

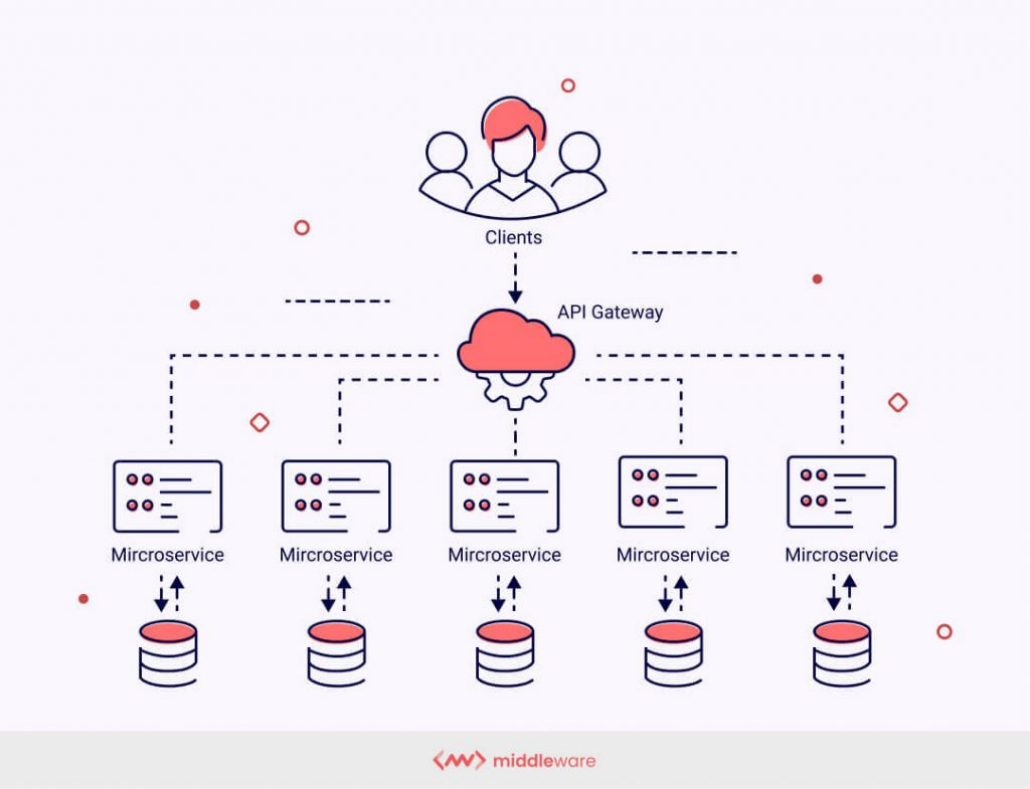


FIGURE 2.1 : Architecture Microservice

Les API

Une API (Application Programming Interface ou Interface de Programmation Applicative) est un ensemble de règles et de protocoles qui permettent à deux logiciels différents de communiquer entre eux. Elle définit les méthodes et les structures de données que les développeurs peuvent utiliser pour accéder aux fonctionnalités ou aux données d'un service ou d'une application, souvent sans avoir à connaître les détails internes de la mise en œuvre. Les formats de données utilisés pour effectuer des échanges d'informations avec une API sont le JSON et le XML.

2.2 Méthodes

2.2.1 Modélisation

2.2.2 UML

UML (Unified Modeling Language) est un langage de modélisation graphique essentiel pour la conception et la documentation des systèmes informatiques. Dans le cadre de ce travail, UML a été employé comme une méthodologie de modélisation permettant de représenter visuellement les différentes facettes de la structure, concernant les acteurs, les cas d'utilisation, et le flow d'exécution des actions liées à des fonctionnalités.

2.2.2.1 Diagramme UML de classe

Un diagramme de classe est une représentations des différents objets du système ainsi que les relations entre elles.

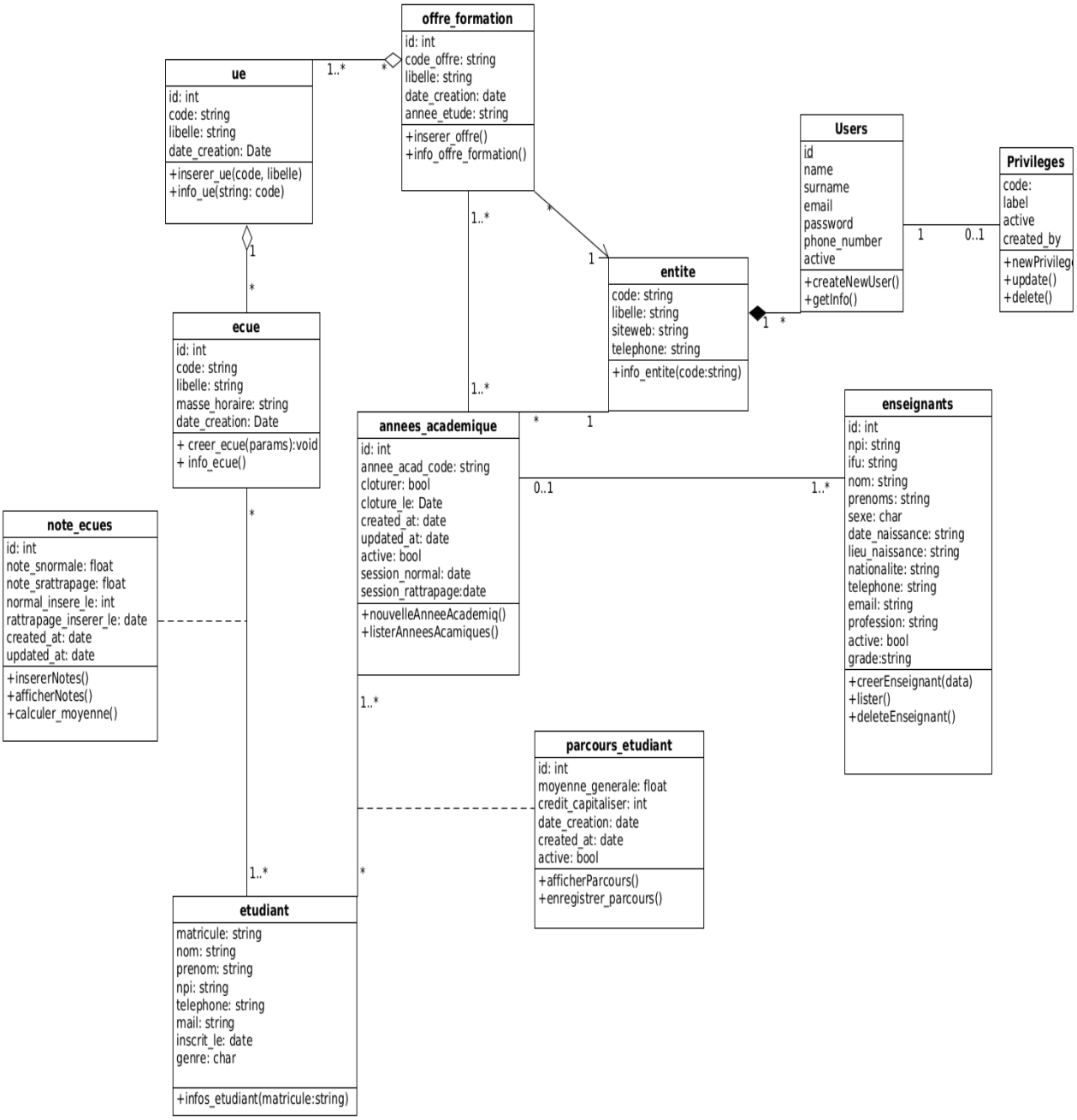


FIGURE 2.2 : Diagramme de classe

2.2.2.2 Diagramme UML de cas d'utilisation

Définition

Un diagramme de cas d'utilisation, également appelé diagramme de cas d'utilisation UML, est un type de diagramme de modélisation qui représente les interactions entre les acteurs (utilisateurs ou systèmes externes) et un système logiciel. Un diagramme de cas d'utilisation permet de décrire les fonctionnalités et les comportements d'un système du point de vue de ses utilisateurs.

La représentation visuelle ci-dessous va illustrer les différents scénario d'interaction entres les acteurs identifiés du système avec le logiciel ainsi que les relations entre les cas d'utilisation répertoriés.

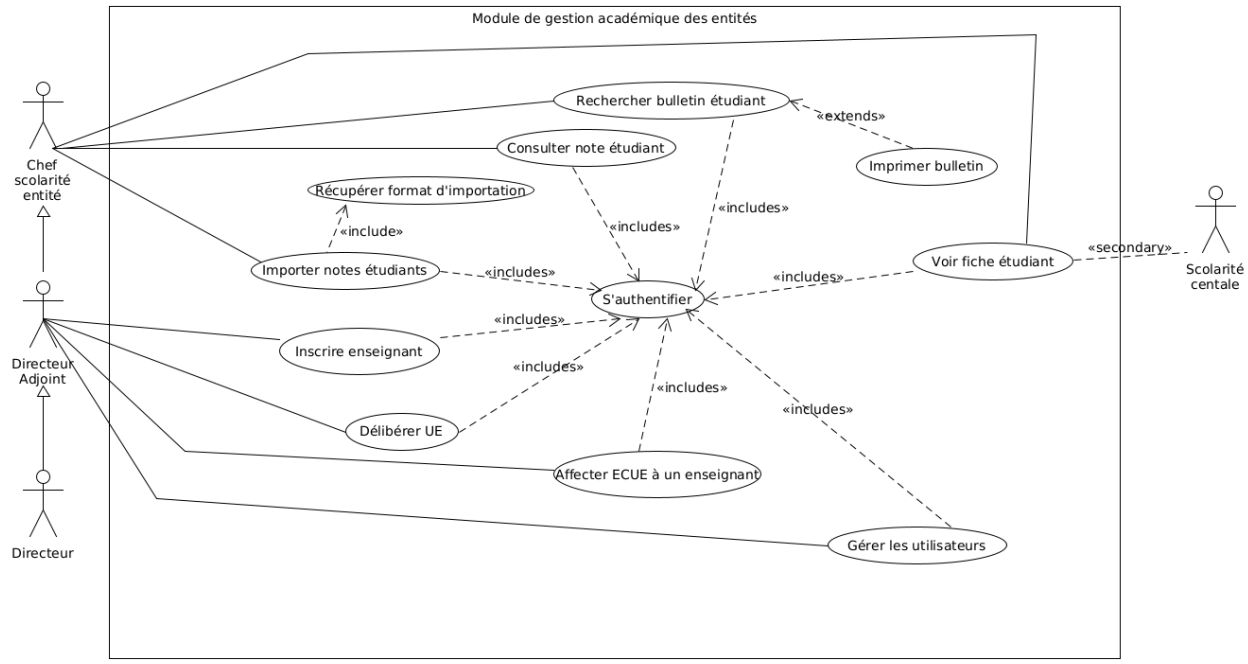


FIGURE 2.3 : Diagramme de cas d'utilisation

2.2.2.3 Diagramme de séquence

Définition

Un diagramme de séquence UML est un type de diagramme de modélisation qui représente les interactions entre les objets ou les composants d'un système logiciel au fil du temps. Les diagrammes de séquence sont largement utilisés dans l'ingénierie logicielle pour visualiser le déroulement des scénarios ou cas d'utilisation, des opérations, des messages et des collaborations entre les objets.

Nous présentons ici, le diagramme UML de séquence relatif au cas d'utilisation **importer notes étudiants** et **rechercher bulletin étudiant**.

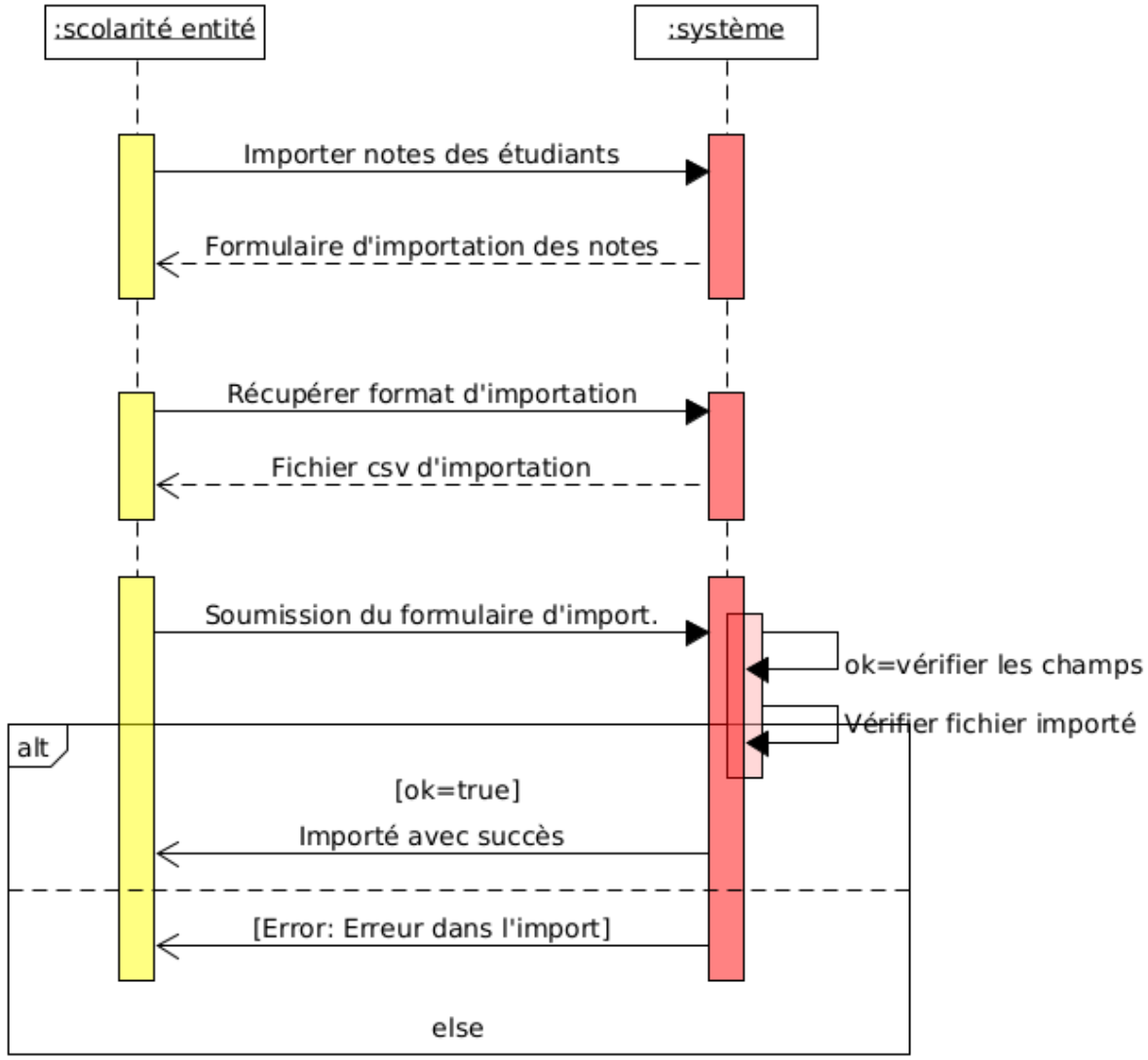


FIGURE 2.4 : Diagramme de séquence du cas d'utilisation : importer notes étudiants.

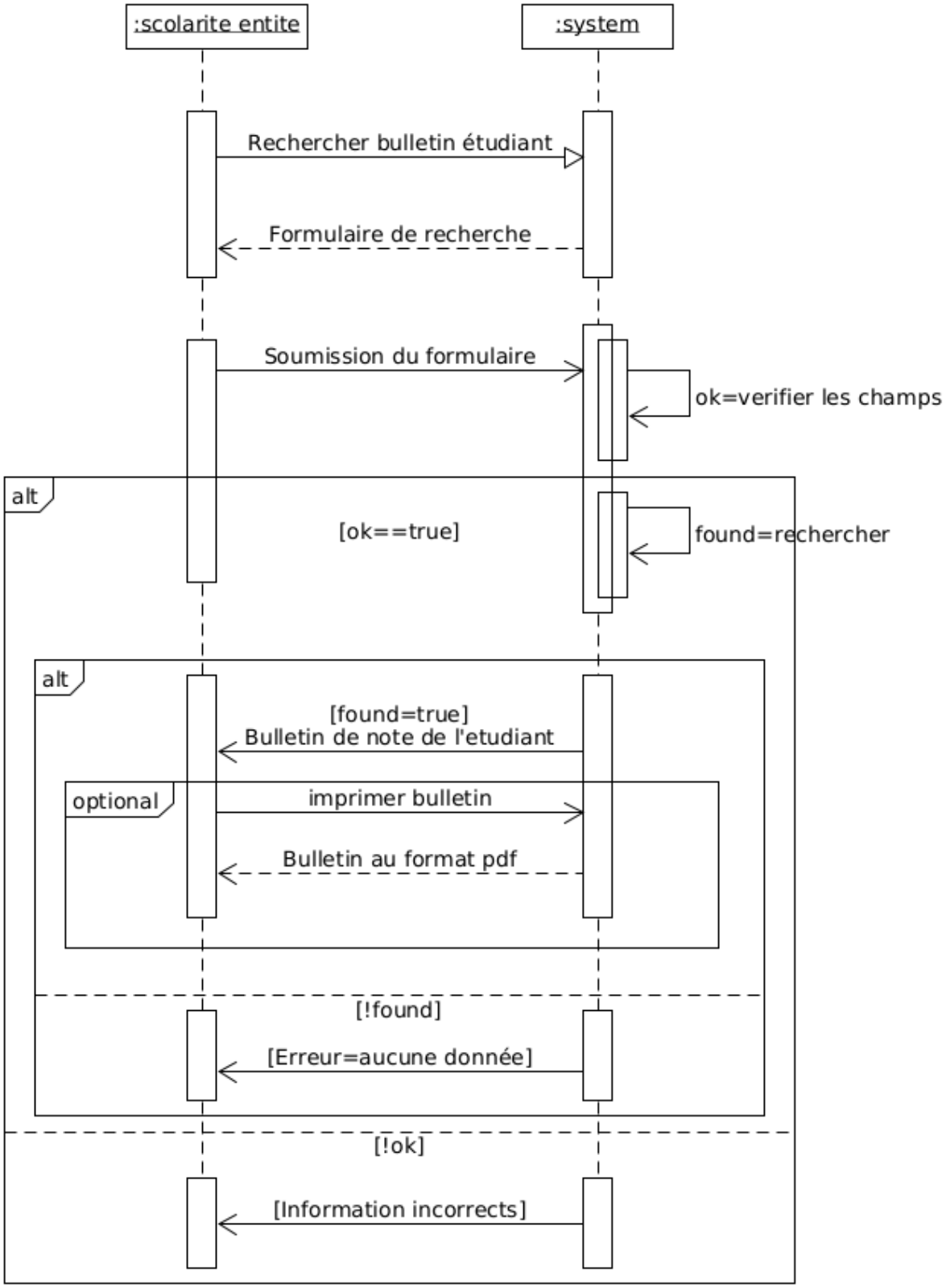


FIGURE 2.5 : Diagramme de séquence du cas d'utilisation : rechercher bulletin

2.3 Outils

2.3.1 Intégration continue et versionning

Docker

Docker est une plate-forme open source de conteneurisation qui permet aux développeurs de créer, de distribuer et d'exécuter des applications dans des environnements isolés appelés 'conteneurs'. Chaque conteneur inclut tout ce dont une application a besoin pour fonctionner, y compris le code, les bibliothèques, les dépendances et les variables d'environnement. Docker simplifie le processus de développement, de test et de déploiement en offrant une solution de conteneurisation légère et portable. [1]

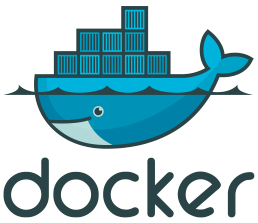


FIGURE 2.6 : Logo de docker.

Une des caractéristiques principales de Docker est sa facilité d'utilisation. En effet toute la configuration nécessaire (variables d'environnement, ports, volumes, fichiers de code, images, ...) est renseignée dans un fichier appelé *Dockerfile* et qui est utilisé pour construire et lancer l'application.

```
1  FROM python:3
2
3  RUN mkdir /app
4
5  WORKDIR /app
6
7  COPY requirements.txt /app/requirements.txt
8
9  RUN pip install -r /app/requirements.txt
10
11 COPY . /app
12
13 EXPOSE 5000
14
15 CMD python3 app.py
```

Listing 2.1 : Dockerfile pour la configuration et le lancement de module entité

Dans l'optique de faciliter la gestion centralisée des conteneurs, surtout dans un contexte multi-conteneurs (il s'agit de plusieurs microservices), la configuration est écrite dans un fichier `docker-compose` d'extension **yml** [8] [7]. La configuration est donc reproductible, centralisée, avec la possibilité d'interdépendre les services lancés. Avec une possibilité de définir les variables d'environnement, on sépare les environnements de *développement* et de *production*. Nous avons donc :

- l'orchestration des conteneurs : plusieurs conteneurs sont utilisés et représentent chacun un service (chacun utilisant un port, un volume, une réseau, des variables d'environnement, etc.)
- La gestion des dépendances ;
- Une configuration centralisée ;
- Une grande facilité de déploiement ;
- Une reproductibilité accrue : les membres de la chaîne de développement et de déploiement, reproduiront et lanceront l'application sans problème.

```
version: '3'
services:
  gestion_entite:
    build: ./services/gestion_entite
    container_name: gestion_entite
    hostname: gestion_entite
    ports:
      - '5000:5000'
    environment:
      - ENV=dev
      - DB_USER=entite_user
      - DB_PASSWORD=peace
      - DB_HOST=entite_db
      - DB_PORT=5432
      - DB_NAME=scolarite
      - DB_SCHEMA=public
      - RABBITMQ_HOST=rabbitmq
      - RABBITMQ_USER=uac-rabbit
      - RABBITMQ_PASSWORD=uac-rabbit
      - RABBITMQ_VHOST=rabbit_vhostc
    volumes:
      - ./services/gestion_entite:/app

  rabbitmq:
    image: rabbitmq:3.8-management
    container_name: rabbitmq
    hostname: 'rabbit'
    ports:
      - '5672:5672'
      - '8080:15672'
    environment:
      - RABBITMQ_DEFAULT_USER=uac-rabbit
      - RABBITMQ_DEFAULT_PASS=uac-rabbit
      - RABBITMQ_DEFAULT_VHOST=rabbit_vhost

  database:
    # depends_on:
    #   - formation
    container_name: entite_db
    image: postgres
    env_file: ./services/database_env.env
    ports:
      - "5432:5432"
    volumes:
      - database-data:/var/lib/postgresql/data

volumes:
  database-data:
```

Listing 2.2 : Dockerfile pour la configuration et le lancement de module entité

2.3.2 Développement

2.3.2.1 Python

Python est un langage de programmation multiparadigme et multiplateforme. C'est un langage de programmation de haut niveau, interprété et polyvalent, qui est largement utilisé dans le développement logiciel, la science des données, l'automatisation, le développement web, l'intelligence artificielle et de nombreux autres domaines.

C'est le framework **Flask** qui est utilisé pour fournir les *endpoints* ou *route* d'API de notre microservice.

`Flask` est un framework web léger et open source écrit en Python. Il est conçu pour simplifier le développement d'applications web en offrant une structure de base tout en permettant aux développeurs de faire des choix sur les composants et les bibliothèques à utiliser. Il présente l'avantage d'être adapté à la création d'API RESTful ce qui en fait un choix populaire pour le développement d'applications web basées sur des services web.[8]

2.3.2.2 Bibliothèques tiers en backend

Plusieurs librairies sont utilisés dans la conception de ce backend, et qui ont été, pour la plupart installé comme dépendance au projet et en complément au framework Flask. Nous avons principalement :

1. **Flask-CORS** : Flask-CORS est une extension de Flask qui permet de gérer les autorisations CORS (Cross-Origin Resource Sharing) pour les requêtes HTTP entre domaines différents. Il est utilisé pour contrôler l'accès aux ressources du backend depuis des domaines différents.

2. **Flask-Swagger-UI** : Flask-Swagger-UI est une extension Flask qui facilite la documentation de votre API Flask en générant automatiquement une interface Swagger pour vos endpoints. Cela permet aux utilisateurs de votre API de visualiser et de tester les endpoints de manière conviviale.

3. **Requests** : Le *package* Requests est une bibliothèque Python populaire pour effectuer des requêtes HTTP. Il est utilisé pour envoyer des requêtes HTTP vers d'autres services web, récupérer des données et interagir avec des API externes depuis votre backend Flask.

4. **Pandas** : c'est une bibliothèque Python utilisée pour la manipulation et l'analyse de données. Elle est utilisée pour le traitement de données, notamment la lecture, la transformation et l'analyse de fichiers CSV, de feuilles de calcul Excel et d'autres formats de données.

5. **Openpyxl** : c'est une bibliothèque Python permettant de lire, de manipuler et de créer des fichiers Excel au format XLSX.

6. **Flasgger** : c'est une extension Flask qui facilite la génération de documentation Swagger pour les endpoints d'API. Il est souvent utilisé conjointement avec Flask-Swagger-UI pour documenter les API de manière claire et structurée. La documentation généré par ce package permet aussi de tester les endpoints documentés.

7. **Psycopg2** : Psycopg2 est un adaptateur PostgreSQL pour Python. Il est utilisé pour interagir avec une base de données en Postgres, de la connexion au lancement de requêtes.

8. **CSV** (module Python standard) : Le module CSV est une partie standard de la bibliothèque Python. Il est utilisé pour la lecture et l'écriture de fichiers CSV, un format de données couramment utilisé pour stocker des données tabulaires.

2.3.2.3 Procédure d'installation des bibliothèques

L'installation de ces bibliothèques (*package* ou *librairy*) python sus-cités est possible grâce à un outil logiciel, plus précisément un gestionnaire de *package* propre à python appelé **pip**.

Ainsi, l'installation d'un package python se fait avec la commande suivante : `pip install < nom_package >`; où < nom_package > est le nom du package à installer. L'installation nécessite une connexion à internet, puisque les packages sont téléchargés depuis des dépôts de package en ligne. Celui de **pip** est PyPI (Python Package Index).

2.3.2.4 Visual Studio Code

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et MacOs. Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégrer. Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

2.3.3 Base de données

2.3.3.1 Postgresql

C'est un système de gestion de base de données relationnelle (SGBDR) open source, robuste et extensible. Il est largement utilisé dans le développement de logiciels, la gestion des données et les applications web en raison de sa puissance, de sa fiabilité et de sa conformité aux normes SQL.

2.3.3.2 DBeaver

DBeaver est un outil de gestion de base de données open source et multiplateforme. Il est utilisé pour se connecter, gérer, requêter et visualiser différentes bases de données relationnelles (Postgres, MySQL, MS SQLServer, SQLite, IBM Db2, MariaDB, etc). Dans notre contexte, il nous permet d'administrer notre base de données **postgres**.

2.3.4 Test d'API

Insomnia C'est une application client qui est utilisée pour lancer des requêtes HyperText Transfer Protocol (HTTP) pour tester les routes d'Application Programming Interface (API). Il dispose d'une interface graphique conviviale au lancement de diverses types de requêtes. On commence par le choix de la méthode de la requête HTTP (GET, POST, PUT, DELETE, PATCH, etc). Ensuite on renseigne le lien à contacter pour la requête, sans oublier le contenu (corps de la requête).

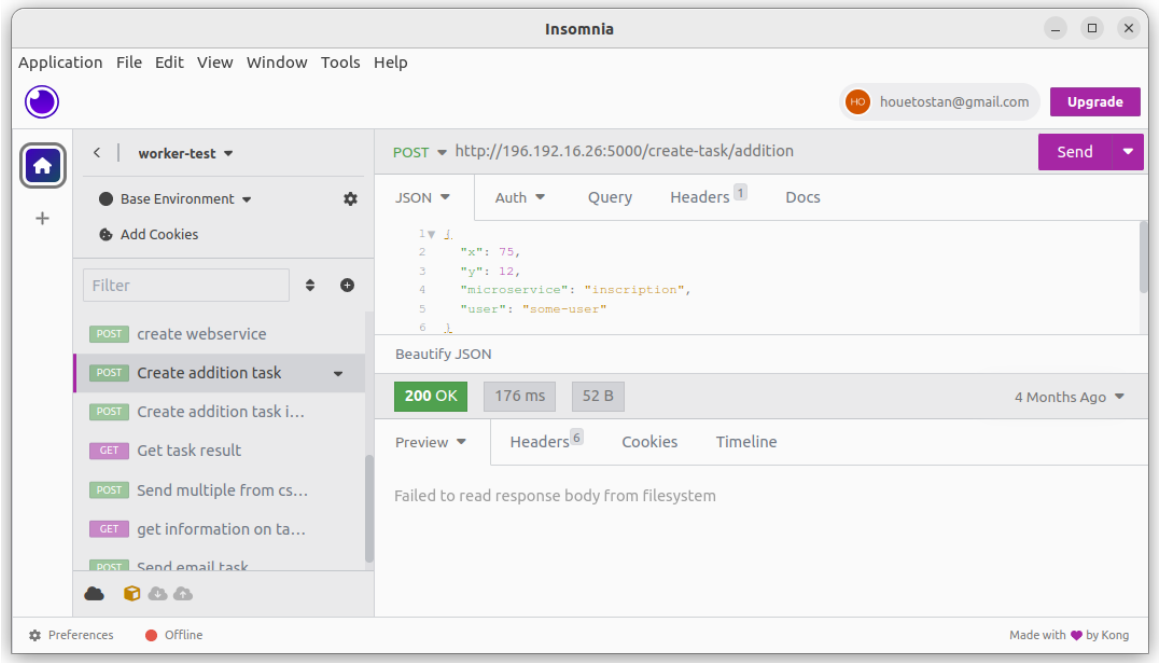


FIGURE 2.7 : interface du client d'API *Insomnia*

2.3.5 Frontend

Pour la mise en place de l'espace utilisateur, nous avons utilisé la librairie frontend écrit en Javascript appelé `REACT JS`.

React, également connu sous le nom de React.js ou ReactJS, est une bibliothèque JavaScript open source largement utilisée pour la création d'interfaces utilisateur (UI) interactives et réactives. Développée et maintenue par la société META (ex Facebook), React permet aux développeurs de construire des applications web complexes en morcelant l'interface utilisateur en composants réutilisables. [10] Le but principal de cette bibliothèque est de faciliter la création d'application web monopage, via la création de composants dépendant d'un état et générant une page (ou portion) HTML à chaque changement d'état.[11]

Le frontend envoie donc les requêtes HTTP pour interagir avec l'application (l'API) en backend.

Ainsi, pour effectuer ses requêtes http, nous utilisons le package Axios, que nous installons avec NPM (Node Package manager).

NodeJs et NPM

Node.js, souvent appelé simplement "Node" est un environnement d'exécution JavaScript côté serveur open source qui permet aux développeurs de créer des applications web et des services réseau hautement évolutifs et performants. Node.js utilise le moteur JavaScript V8 de Google Chrome pour exécuter du code JavaScript côté serveur, ce qui en fait un choix populaire pour les applications web en temps réel et les API RESTful. [3]

NPM (Node Packet Manager) est le gestionnaire de paquets par défaut pour Node.js. Il offre un vaste écosystème de bibliothèques et de modules open source qui peuvent être facilement installés et utilisés dans les projets Node.js. [3]

Conclusion

Dans ce chapitre nous avons fait une synthèse des outils, techniques et méthodes utilisés dans la mise en place de ce module. Nous avons présenté une approche basée sur l'architecture en microservice qui met en avant des services web qui communiquent entre elles par le biais du protocole HTTP. Cette architecture est mis en place pour favoriser une simplicité, une robustesse et une facilité d'intégration accrue grâce à l'outil de conteneurisation et d'intégration continue qu'est **Docker** et aux divers Frameworks frontend et backend mis en oeuvre.

Chapitre 3

Résultat et discussion

Introduction

Ce chapitre présente les résultats de notre travail à travers des captures d'écran. Nous présenterons ainsi, l'interface et quelque séquences de fonctionnement du module de gestion des entités académiques.

3.1 Fonctionnement générale de l'application

Une interface de connexion va permettre à l'utilisateur de se connecter sur le système et de faire usage des fonctionnalités à partir du dashboard.

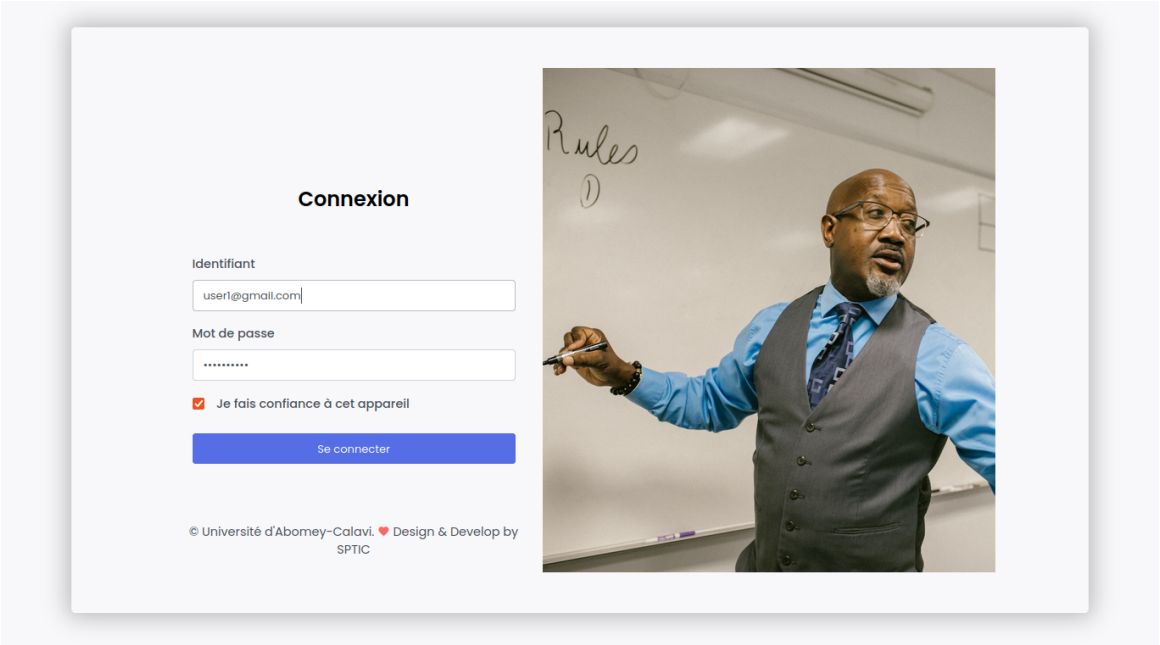


FIGURE 3.1 : Interface de connexion au module

3.1.1 Dashboard

Après l'authentification de l'utilisateur, le dashboard de travail est présenté. L'entité académique auquel il appartient est détecté, ainsi que son privilège sur le système.

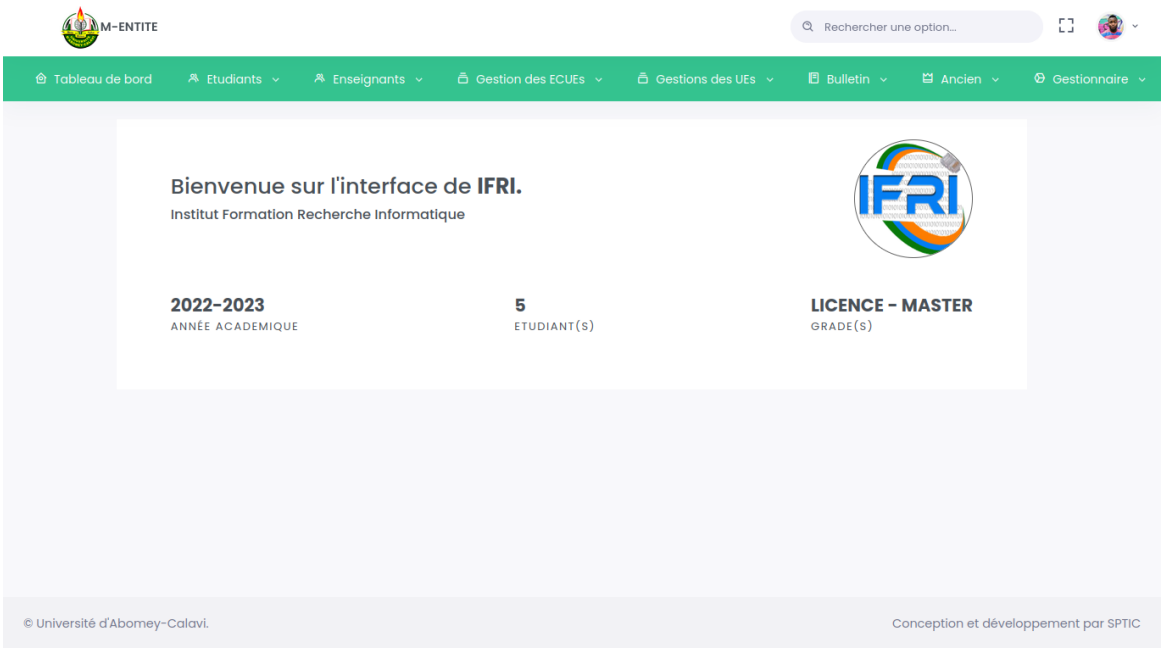


FIGURE 3.2 : Dashboard principal

Le menu est réparti en sept (7) rubriques principales, lesquelles ont des sous rubriques listant les actions à effectuer. Les menus principales sont entre autres :

- Étudiants ;
- Enseignants ;
- Gestions des ECUEs ;
- Gestion des UEs ;
- Bulletin ;
- Ancien ;
- Gestionnaire.

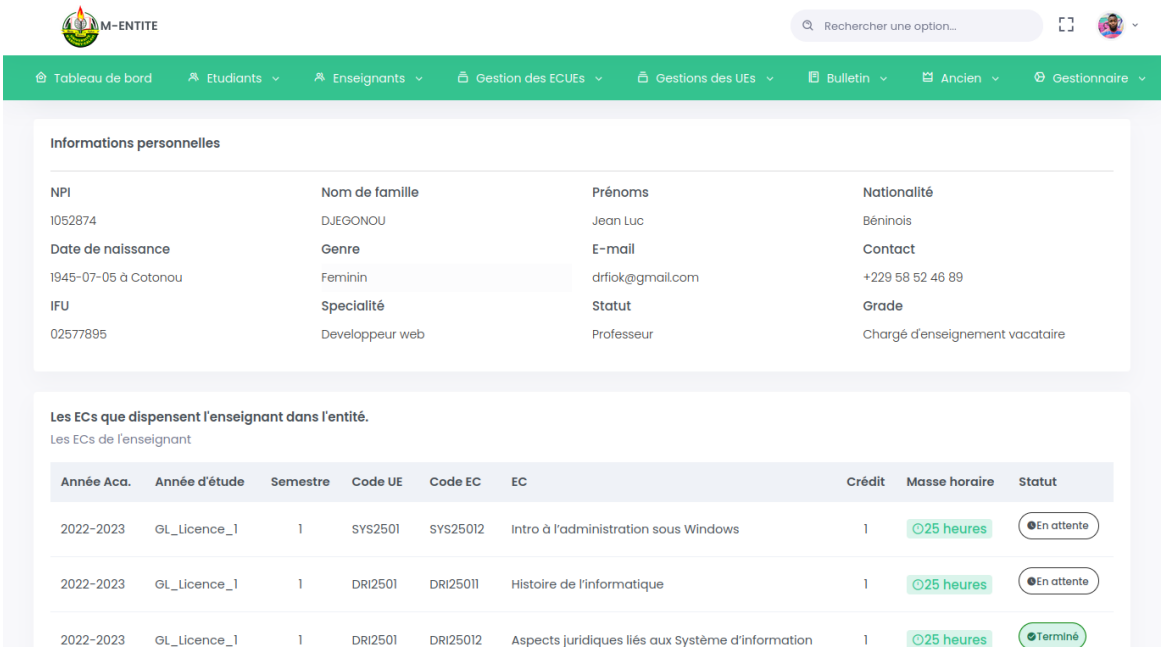


FIGURE 3.3 : Page récapitulant les informations relatives à un enseignant.

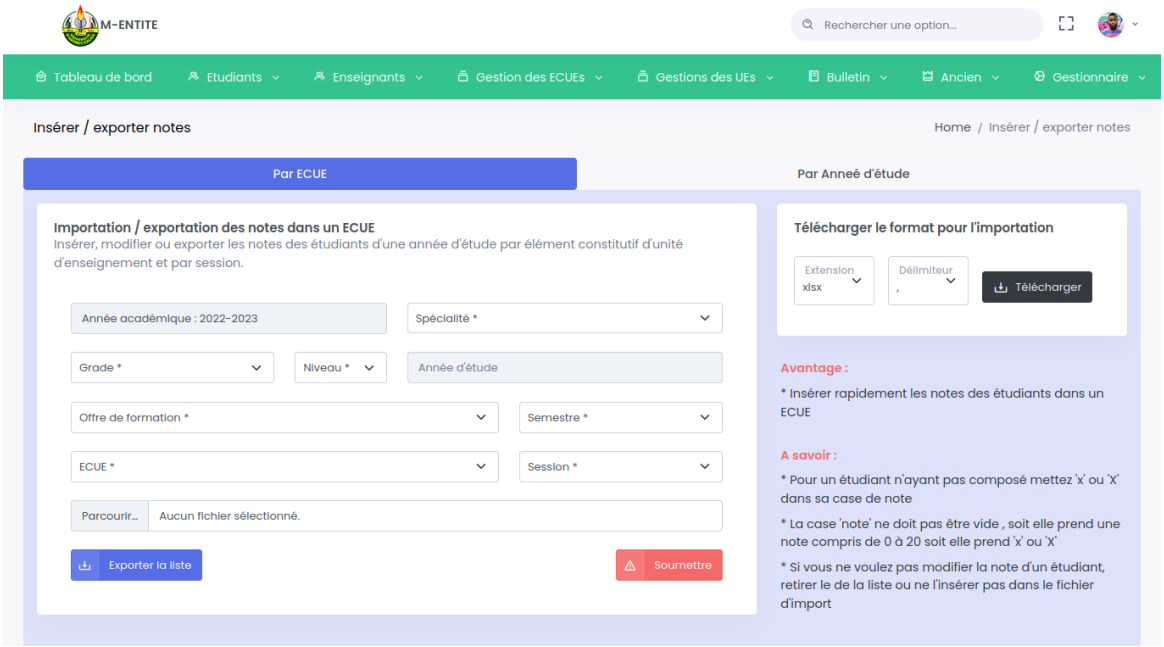


FIGURE 3.4 : Importation des notes

La capture ci-dessus montre l'interface d'importation des notes. Elle peut être effectué par ECUE ou par année d'étude. Un canevas d'importation a été définit. C'est donc par un fichier excel (au format xlsx ou csv) que les notes sont insérées et chargées dans le système. A partir de l'interface, on peut récupérer le fichier type d'importation de notes.

3.1.2 API et documentation

Ce module doit interagir avec d'autres services à commencer par le service fournissant l'interface utilisateur du module entité. Ainsi, une documentation répondant aux spécifications OpenAPI a été mise en place avec les outils appropriés. La documentation donne les informations relatives à chaque

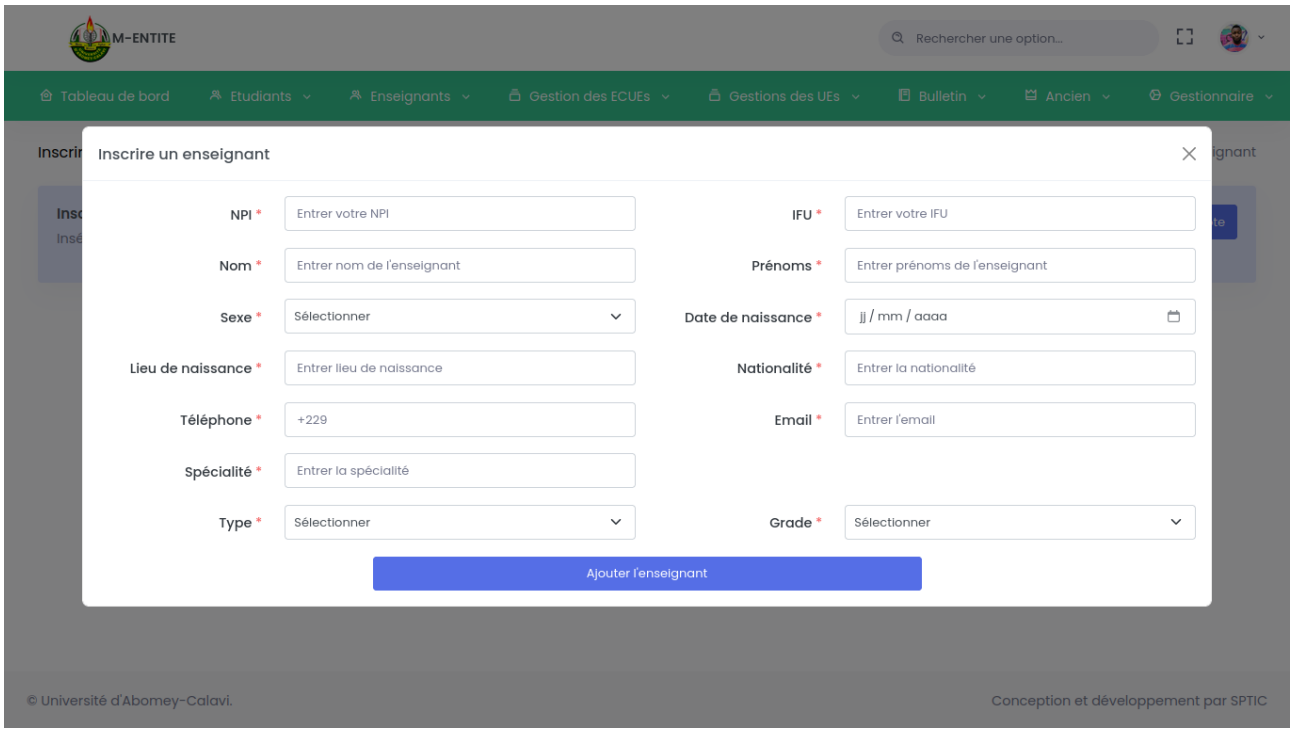


FIGURE 3.5 : Formulaire d'inscription d'un nouvel enseignant dans le système.

route (paramètres, résultats, format de données, etc) et fournit aussi un environnement de test de chacune d'elle.



FIGURE 3.6 : Documentation swagger de l'API

3.2 Discussion

L'application web fournit va constituer une aide à la gestion des entités académiques de l'UAC. A cet effet l'architecture adoptée va permettre de fournir des informations à d'autres modules ou microservices de l'architecture globale. A titre d'exemple, le module de l'espace étudiant, à travers nos endpoints (route HTTP) d'API et permettre de connaître le parcours de l'étudiant dans une entité. Aussi, les données traitées au sein de ce module peuvent être collectées et traitées de façon organisée pour des statistiques par la scolarité centrale ou l'équipe rectorale.

Néanmoins, il présente quelques insuffisances, qui sont entrain d'être minutieusement prises en charge et corrigées sous peu. Il s'agit d'une authentification centralisée. A cet effet, l'UAC dispose d'un annuaire LDAP, qui est une base de données hiérarchiques des différents acteurs de notre institution en fonction de leur rôle et accréditation. Un microservice devra prendre cela en charge, et à qui nous devrons déléguer l'authentification des différents acteurs sur la plate-forme. Ainsi, pour les besoins de ce projet, une authentification en interne est mise en oeuvre.

Aussi, certaines tâches liées en backend comme l'importation des notes restent lourdes sur le système, compte tenu de la quantité des données à importer et la concurrence liée à une utilisation régulière. Ces traitements massifs en arrière plan risquent de ralentir le système quand ils seront lancés. De ce fait il est prévu d'utiliser un broker, RabbitMQ pour le traitement asynchrone de tâches, et la gestion de file d'attente.

Conclusion

Dans ce chapitre nous avons présenté les résultats de notre travail sur le module des entités académiques de l'UAC. Les résultats obtenus au terme de la phase de développement de l'application ont été présentés sous forme de capture d'écran montrant les fonctionnalités développées disponibles dans l'interface utilisateur de l'application web.

Aussi, nous avons aussi présentés quelques limites concernant ce module notamment la gestion des tâches lourdes et l'authentification centralisée des utilisateurs.

Conclusion Générale

Dans ce travail, nous avons proposé une solution capable de soutenir les entités académiques de l'UAC dans les tâches qui leurs incombent. De la gestion des étudiants aux enseignants, sans oublier les notes et délibérations, la solution proposée est aussi dans l'optique de répondre à un besoin de conception d'un espace numérique de travail. Ainsi, nous avons produit une application web se basant sur une architecture en microservice pour pouvoir interagir avec d'autres modules du projet. Ce module fait partie d'un système plus globale rentrant dans le cadre de l'espace numérique de travail de l'UAC. Nos prochains travaux sur ce module se pencheront sur l'ajout de nouvelles fonctionnalités répondant aux attentes et aussi l'aspect sécurité et performance, ce qui va permettre de garantir une bonne expérience utilisateur. Comme fonctionnalité, nous pouvons citer : la prise en charge de la présence et de l'absence des étudiants et sans oublier les relations parents-administration-partenaires.

Bibliographie

[1] Jeff NICKOLOFF et Stephen KUENZLI. *Docker in action*. Manning, 2019.

[2] Chris RICHARDSON. *Microservices Patterns*. Manning, 2019.

[3] Alex YOUNG, Bradley MECK et Mike CANTELON. *Node.js in action, Second Edition*. Manning, 2017, p. 10-15.

Webographie

[4] Consulté le 10 Oct 2023. URL : https://uac.bj/formation/nos-entites/.

[5] Consulté le 10 Octobre 2023. URL : https://uac.bj/formation/offre-de-formation/.

[6] https ://www.appvizer.fr/education/enseignement-sup/. Consulté le 09 Octobre 2023. URL : https://www.appvizer.fr/education/enseignement-sup/.

[7] Docker DOCUMENTATION. *Docker compose overview*. https ://docs.docker.com/compose/. Consulté le 05 Octobre 2023.

[8] Diogo FERNANDES. *Containerising Flask, Celery & Postgres*. https ://diogo-fernandes.medium.com/containe flask-celery-postgres-ec73f62f4eab. Consulté le 24 Juillet 2023. Jan. 2021.

[9] IBM. *What are microsercice ?* https ://www.ibm.com/topics/microservices. consulté le 04 Août 2023.

[10] *Introducing React.dev*. https ://react.dev/blog/2023/03/16/introducing-react-dev. Moderne react introduction. Consulté le 08 Octobre 2023.

[11] WIKIPEDIA. *React*. https ://fr.wikipedia.org/wiki/React. consulté le 08 Octobre 2023.

Table des matières

Dédicace ii

Remerciements iii

Résumé iv

. iv

Abstract v

Liste des figures vi

Acronymes vii

Glossaire viii

Introduction 1

1 Revue de littérature. 3

1.1 Présentation de l'architecture globale de l'Espace Numérique de Travail. 3

1.2 Les entités académiques 4

1.2.1 Définition et clarification. 4

1.2.2 Les activités de gestion dans les entités académiques 5

1.3 Application web 5

1.3.1 Composition 5

1.3.1.1 Frontend 5

1.3.1.2 Backend 6

1.4 Microservice 6

1.5 Les solutions existantes 6

1.5.1 Aimaira ERP 6

1.5.2 GestStudent 8

1.5.3 Egaster 9

1.6 Hypothèses de recherche 11

2 Méthode de conception et choix techniques 12

Introduction 12

2.1 Architecture 12

2.2 Méthodes 13

2.2.1 Modélisation 13

2.2.2 UML 14

2.2.2.1 Diagramme UML de classe . . . 14
2.2.2.2 Diagramme UML de cas d'utilisation . . . 15
2.2.2.3 Diagramme de séquence . . . 16
2.3 Outils . . . 18
2.3.1 Intégration continue et versionning . . . 18
2.3.2 Développement . . . 21
2.3.2.1 Python . . . 21
2.3.2.2 Bibliothèques tiers en backend . . . 21
2.3.2.3 Procédure d'installation des bibliothèques . . . 22
2.3.2.4 Visual Studio Code . . . 22
2.3.3 Base de données . . . 22
2.3.3.1 Postgresql . . . 22
2.3.3.2 DBeaver . . . 22
2.3.4 Test d'API . . . 22
2.3.5 Frontend . . . 23
Conclusion . . . 24

3 Résultat et discussion **25**
Introduction . . . 25
3.1 Fonctionnement générale de l'application . . . 25
3.1.1 Dashboard . . . 26
3.1.2 API et documentation . . . 27
3.2 Discussion . . . 29
Conclusion . . . 30

Conclusion **31**

Références bibliographiques **32**

Références webographiques **33**

Table des matières **34**