Université Assane Seck de Ziguinchor

UFR des sciences et technologies

Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de master

Mention : Informatique

Spécialité : Génie Logiciel

Dématérialisation de l'expression des besoins du personnel de l'Université Assane Seck de Ziguinchor (EBP-UASZ), cas du département Informatique

Présenté par :

M. Diambar SENE

Le 29/04/2023 à 08 H à la salle Cisco de l'UASZ

Sous la direction de :

Pr Ibrahima DIOP **&** Pr Ousmane DIALLO

Devant le jury composé de :

Pr Youssou	DIENG	Président de Jury	UASZ
Pr Youssou	FAYE	Examinateur	UASZ
Dr Mouhamadou	GAYE	Rapporteur	UASZ
Pr Ibrahima	DIOP	Encadreur	UASZ
Pr Ousmane	DIALLO	Co-encadreur	UASZ

2021-2022

Résumé

L'expression des besoins de l'Université Assane SECK de Ziguinchor (UASZ) repose sur un fichier Excel. C'est un fichier Excel que l'UASZ met à la disposition de son personnel pour leur permettre d'exprimer leurs besoins en matériel ou en fourniture afin d'accomplir leurs fonctions. Le fichier Excel est mis à disposition par le responsable des services généraux aux chefs de département qui transmettent ensuite aux collègues Enseignants-Chercheurs, lesquels doivent contribuer à la sélection du matériel à commander. Cependant, cette méthode présente plusieurs limites, notamment la gestion par envoi multiples de mails, la difficulté de suivre l'avancement des commandes et d'enregistrer les données de manière structurée et accessible, entre autres.

Pour remédier à ces limites, nous proposons dans ce mémoire une application web permettant d'automatiser le processus de gestion de l'expression des besoins. Ainsi, tous les acteurs impliqués dans ce processus peuvent accéder en ligne à un catalogue de produits afin d'effectuer et de gérer leurs commandes depuis n'importe quel ordinateur ou appareil mobile grâce à une interface intuitive avec des fonctionnalités adaptées.

Pour mettre en place cette application, nous avons utilisé l'architecture de microservices, ce qui la rend facile à développer, à déployer et à maintenir. En outre, l'approche CI/CD, qui représente une mise en œuvre de la culture DevOps, a été utilisée pour développer l'application, garantissant ainsi une qualité et une stabilité optimales. Une spécification claire des besoins a permis de formaliser les données et de les stocker dans des bases de données MySQL (My Structured Query Language). Le Framework Spring boot a été utilisé pour le développement de la partie backend, tandis que Thymeleaf a été utilisé pour la partie frontend.

Abstract

The expression of needs of the University Assane SECK of Ziguinchor (UASZ) is based on an Excel file. It is an Excel file that UASZ provides to its staff to enable them to express their needs in equipment or supplies to perform their duties. The Excel file is made available by the head of general services to the heads of department who then transmit it to their colleagues, who must contribute to the selection of the material to be ordered. However, this method has several limitations, including management by sending multiple emails, the difficulty of tracking the progress of orders and recording data in a structured and accessible manner, among others.

To remedy these limitations, we propose in this thesis a web application to automate the process of managing the expression of needs. Thus, all actors involved in this process can access online a product catalog in order to perform and manage their orders from any computer or mobile device thanks to an intuitive interface with adapted functionalities.

To implement this application, we used the microservices architecture, which makes it easy to develop, deploy and maintain. In addition, the CI/CD approach, which represents an implementation of the DevOps culture, was used to develop the application, thus ensuring optimal quality and stability. A clear specification of the requirements allowed the data to be formalized and stored in MySQL (My Structured Query Language) databases. The Spring boot framework was used for the development of the backend part, while Thymeleaf was used for the frontend part.

Dédicaces

Par la grâce d'Allah, le tout puissant je dédie ce travail :

A mon père

C'est grâce à toi que nous sommes là aujourd'hui. Tu n'as ménagé aucun effort pour nous mettre dans des conditions exemplaires. Tu es et tu resteras le meilleur père.

Trouve dans ce travail l'expression de mon amour filial.

A ma mère

Les mots ne sont jamais assez forts pour exprimer la reconnaissance d'un fils envers sa mère. Tu es pour nous une mère exemplaire qui a su nous entourer d'un grand amour sans nous faire oublier le respect de soi-même et l'aide à son prochain.

Je te dédie ce travail du fond du cœur en espérant que tu pourras être un jour fier de moi.

Tous mes vœux de santé, de bonheur et de longue vie.

A mes frères et sœurs

Puissions-nous rester unis solidaires dans la vie et fidèles à l'éducation que nos chers parents ont su nous inculquer.

Que Dieu continue à guider vos pas.

A ma famille d'accueil

Sans vous je ne saurais arriver là où je suis. Merci pour vos soutiens, et votre considération.

A tous mes amis

Vos soutiens, conseils et encouragements me tiennent à cœurs.

Remerciements

Je remercie ALLAH (SWT) le Tout-Puissant de m'avoir permis de mener à terme ce projet qui est pour moi le point de départ d'une merveilleuse aventure. Nous prions également sur son noble prophète Seydina MOUHAMED (PSL).

Je tiens à remercier chaleureusement et à témoigner toute ma reconnaissance à mes encadrants Pr. Ibrahima DIOP et Pr. Ousmane DIALLO pour avoir donné de leur temps et de leur intelligence à la réussite de ce projet. Ils représentent pour moi un modèle de réussite et une source de motivation permanente. Leur disponibilité, et leur sens aigu de l'humanisme m'ont marqué.

Nos vifs remerciements vont également aux membres du jury notamment au Pr. Youssou DIENG, au Dr. Mouhamadou GAYE et au Pr. Youssou FAYE qui ont accepté d'évaluer notre travail.

Je tiens également à remercier le corps professoral du département d'Informatique de l'Université Assane Seck de Ziguinchor, pour la richesse et la qualité de leurs enseignements et les efforts fournis pour assurer à leurs étudiants une bonne formation.

Je remercie également M. Abdou Aziz DIEDHIOU Ingénieur informatique (Développeur Sénior back end) depuis le Canada pour le soutien technique qu'il m'a apporté pour la réalisation de ce projet.

Mes remerciements s'adressent à mes parents, mon père Kora SENE et ma mère Khady NDIAYE, qui m'ont inculqué un esprit de combativité et de persévérance et qui m'ont toujours poussée et motivée dans mes études. Sans eux, certainement je ne serai pas à ce niveau.

Merci également à mes frères et sœurs. Les mots me manquent pour les qualifier. Ils n'ont jamais cessé de me soutenir, de m'encourager de me conseiller et de m'aider.je vous dois beaucoup.

Je remercie tous mes camarades de promotion pour leur collaboration tout au long de ma formation.

Je tiens 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.

Table des matières

Liste des figures

Liste des tableaux

Liste abréviations

UASZ : Université Assane SECK de Ziguinchor

UFR : Unité de Formation et de Recherche

API: Application Programming Interface

UML: Unified Modeling Language

DTOs: Data Transfer Objects

MYSQL: My Structured Query Language

SQL : Structured Query Language

SGBD : Système de Gestion de Base de Données

IDE: Integrated Development Environment

HTTP: Hypertext Transfer Protocol

URL: Uniform Resource Locator

JS: JavaScript

HTML : Extensible HyperText Markup Language

INTRODUCTION GENERALE

L'Université Assane Seck de Ziguinchor (UASZ), tout comme les autres institutions universitaires, joue un rôle primordial dans l'éducation et la recherche. La gestion de l'université est une tâche complexe qui nécessite une coordination efficace de toutes les activités qui y sont menées. Le responsable des services généraux et les chefs de département sont responsables de la coordination de différentes tâches telles que la gestion de l'expression des besoins du personnel, c'est-à-dire des demandes de matériels ou de fournitures nécessaires à l'exercice de leurs activités.

L'expression des besoins du personnel de l'UASZ est actuellement effectuée à partir d'un fichier Excel mis à disposition par le responsable des services généraux aux secrétaires et aux chefs de département qui le transmettent aux collègues Enseignants-Chercheurs qui doivent contribuer aux choix du matériel à commander. Cette méthode présente plusieurs limites, notamment la gestion par envois multiples de mails, la difficulté à suivre l'avancement des commandes et à enregistrer les données de manière structurée et accessible, l'absence de tableau de bord, la fusion manuelle des commandes, l'absences de statistiques et de pertes de donnée, etc.

Pour pallier ces limites, ce projet de mémoire propose de concevoir et d'implémenter un système d'information optimal et sûr pour l'automatisation de l'expression des besoins du personnel, en particulier du département informatique et en général de l'UASZ.

Cette automatisation consiste à dématérialiser l'expression des besoins du personnel en leur permettant de gérer leurs demandes de matériel depuis leur terminal et de pouvoir les suivre jusqu'à la livraison.

Pour la mise en œuvre de cette application, la gestion de l'expression des besoins du département d'Informatique est prise comme modèle de base. De l'étude de son fonctionnement, se sont découlées quelques idées qui ont permis de proposer les fonctionnalités du système.

L'application est basée sur l'architecture microservices, qui permet de scinder les différentes fonctionnalités en petits services distincts et facilement maintenables. Pour garantir une livraison rapide et fiable, nous avons adopté l'approche CI/CD, qui intègre les pratiques DevOps pour le développement et la livraison continue du logiciel.

Nous avons choisi une approche de développement guidée par le contrat de l'API, appelée Contract-first. Cette approche consiste à construire le contrat de l'API en premier lieu, générant ainsi la documentation et le code ensuite. Pour ce faire, nous avons utilisé YAML et SwaggerHub pour écrire nos contrats d'API.

Pour développer l'application, nous avons utilisé des technologies telles que Java, Spring Boot, Spring Cloud et Thymeleaf. Java, Spring Boot et Spring Cloud sont utilisés pour développer le backend, tandis Thymeleaf est utilisé pour développer la partie frontend de l'application.

Ainsi, pour bien structurer notre travail, ce rapport de mémoire est réparti en sept (7) chapitres qui abordent les différents aspects du sujet :

1. Description du sujet : Ce chapitre présente le sujet de notre mémoire en décrivant les défis de la gestion de l'expression des besoins du personnel. Il énonce également la problématique sous-jacente et les objectifs visés par notre travail.
2. Architecture générale et choix technologique : Ce chapitre se concentre sur l'architecture globale du système et sur les choix technologiques qui ont été effectués pour sa mise en œuvre.
3. Spécification et conception de l'API utilisateur : Ce chapitre comprend une description des acteurs et des fonctionnalités de l'API utilisateur ainsi qu'une analyse des besoins fonctionnels à l'aide des diagrammes de cas d'utilisation et de séquence.
4. De même, le chapitre dédié à la spécification et la conception de l'API catalogue suit les mêmes étapes que celui de l'API utilisateur.
5. Le chapitre concernant la spécification et la conception de l'API commande suit également les mêmes étapes que celui de l'API utilisateur.
6. Implémentation : Ce chapitre présente les outils de développement utilisés pour l'implémentation.
7. Présentation de l'application : Ce chapitre est composé d'une série d'images qui présentent l'application développée. Il donne une vue d'ensemble des fonctionnalités et de l'interface utilisateur.

CHAPITRE I : DESCRIPTION DU SUJET

Introduction

Ce premier chapitre revêt une grande importance, car il permet de poser les bases pour une compréhension approfondie de la dématérialisation de l'expression des besoins du personnel de l'UASZ. Il est donc crucial d'examiner l'existant afin d'évaluer les méthodes actuellement utilisées pour traiter ce sujet et de déterminer leurs avantages et leurs limites. Cette démarche permet de mettre en évidence les défis actuels auxquels le personnel de l'UASZ est confronté lorsqu'il exprime ses besoins.

En prenant en compte les limites de l'existant, le chapitre propose une solution pour surmonter ces défis et améliorer les processus actuels. Cette approche est importante car elle permet d'optimiser les processus et de les rendre plus efficaces, ce qui peut également améliorer l'expérience du personnel de l'UASZ, qui sera en mesure d'exprimer plus facilement et plus rapidement ses besoins.

Enfin, le chapitre énonce les objectifs du sujet ainsi que la méthodologie utilisée pour les atteindre. Cette étape est cruciale, car elle permet de s'assurer que les objectifs du projet sont clairement définis et que les méthodes utilisées pour les atteindre sont solides et justifiées.

I. La gestion de l'expression des besoins

L'expression des besoins pour le personnel dans le cadre de leur travail est un processus important qui permet d'identifier et de hiérarchiser les ressources et les outils dont le personnel a besoin pour accomplir leur mission de manière efficace. Les besoins peuvent varier selon les enseignants, les universités et les programmes d'enseignement, mais certains des éléments les plus couramment identifies incluent.

1. Matériel pédagogique : les enseignants peuvent avoir besoin d'outils et de ressources pour soutenir leur enseignement, tels que des livres, des supports de cours, des vidéos de projecteur, etc.
2. Technologie de l'information : les enseignants peuvent avoir besoin d'accès à des outils technologiques pour enseigner et communiquer avec les étudiants, tels que des ordinateurs, des tableaux interactifs, des logiciels, etc.
3. Espace de travail : les enseignants peuvent avoir besoin d'un environnement de travail confortable et fonctionnel, y compris un bureau spacieux et bien équipé, etc.

L'expression des besoins pour le personnel enseignant peut être effectuée par l'intermédiaire de sondages, d'entretiens, de groupes de travail, etc. Les résultats peuvent être utilisés pour élaborer un plan d'action pour satisfaire les besoins identifies.

A l'UASZ, le personnel s'effectue leur expression des besoins en suivant un processus clairement défini. Cette démarche permet de garantir que les besoins en matériel du personnel sont pris en compte de manière efficiente et en accord avec les procédures établies.

En effet, ce processus bien défini pour la commande de matériel permet à l'administration de l'université de planifier ses achats en conséquence et d'éviter les erreurs ou les duplications de commande. De plus, en suivant cette procédure, le personnel peut s'assurer que sa commande sera traitée.

En somme, ce processus clairement défini pour la commande de matériel à l'UASZ permet d'optimiser l'efficacité des opérations et de garantir que les besoins du personnel sont satisfaits de manière adaptée.

Ce processus de gestion des demandes de produits pour le personnel du département informatique repose sur un fichier Excel envoyé par le responsable des services généraux à chaque chef de département. Ce fichier Excel contient une liste de produits organisée en catalogue pour faciliter la commande par le personnel.

Le personnel peut ainsi indiquer la quantité souhaitée pour chaque produit directement dans le fichier Excel, ou ajouter un produit qui n'est pas répertorié dans une colonne spécifique. Ce système permet de centraliser les demandes de manière claire et organisée, ce qui facilite la gestion des commandes.

Une fois que tout le personnel a terminé leur demande, le chef de département regroupe les commandes de son département en un seul fichier Excel rempli. Ce fichier est ensuite envoyé au responsable des services généraux qui se charge de regrouper les commandes de tous les départements et d'acheter les produits nécessaires.

Le Tableau 1 ci-dessous illustre le processus de l'expression des besoins du personnel de l'UFR des sciences et technologies destiné aux départements.

Tableau 1: Processus de l'expression des besoins du personnel de l'UASZ

Etape	Acteur	Action
1	Responsable des services généraux	Construit un fichier Excel contenant l'ensemble des produits regroupés par catalogue
2	Responsable des services généraux	Envoi ce fichier Excel à chaque chef de département
3	Chef de département	Transmet le fichier Excel à tout le personnel de son département
4	Personnel	Indique la quantité souhaitée pour chaque produit dans le fichier Excel
5	Personnel	Si un produit n'est pas répertorié, ajoute ce produit dans une colonne dédiée et indique la quantité souhaitée
6	Chef de département	Regroupe les commandes de son département en un seul fichier Excel rempli
7	Chef de département	Envoie ce fichier au responsable des services généraux
8	Responsable des services généraux	Regroupe les commandes de tous les départements
9	Responsable des services généraux	Achète les produits nécessaires
10	Responsable des services généraux	Envoie les produits aux chefs de département pour distribution

Ce fichier Excel qui est mis à disposition des enseignants par le chef de département, est organisé par catalogue qui regroupent les différents produits nécessaires pour l'enseignement, tels que les fournitures de bureau, les matériels informatiques, les produits d'entretien, les équipements scientifiques, etc. Chaque produit est décrit avec sa désignation, la quantité souhaitée, sa nature et le prix unitaire.

Ce système permet aux enseignants d'exprimer facilement leurs besoins et de les catégoriser pour une meilleure organisation. Si un enseignant a besoin d'un produit qui ne figure pas dans la fiche, il peut l'ajouter dans une colonne spécifique, en indiquant la quantité souhaitée. Ce

processus permet une grande flexibilité pour les enseignants, leur permettant de personnaliser leurs besoins selon leurs besoins spécifiques.

L'avantage de ce système est qu'il facilite la communication entre les enseignants et le chef de département pour l'approvisionnement des matériels. Les enseignants n'ont plus besoin de faire des demandes individuelles qui peuvent être difficiles à suivre, ce qui permet un traitement plus rapide et plus efficace des demandes.

Les **Figure** 41, 2, 3 et 4 ci-dessous illustrent le fichier de l'expression des besoins du personnel de l'UFR des sciences et technologies destiné aux départements.

	A	B	C	D	E
1		**Fourniture de bureau**			
2	**N°**	**Designation**	**Quantite**	**Nature**	**P.U**
3	1	Agenda Ministre			2 000
4	2	Agenda President			2 500
5	3	Agenda pochette			1 500
6	4	Memo avec calendrier			3 000
7	5	Agrafeuse bb JACKY	4		1 500
8	6	Agrafeuse GM JACKY	4		3 000
9	7	Agrafeuse grand model avec manche			23 000
10	8	Blanco ruban scooter	3		500
11	9	Bloc éphéméride	1		1 200
12	10	Bloc note Gm avec spirale	20		650
13	11	Bloc note Pm avec spirale	20		350
14	12	Bloc note moyen avec spirale	40		250
15	13	Boite d'archives gm			450
16	14	Boite d'archives pm			350
17	15	Bracelet élastique		boite	700

Figure 1: Extrait 1 du fichier Excel (2021-2022) pour l'expression du besoin du personnel de l'UFR ST

		Matériel de cours et de tp			
151	N°	DESIGNATION	**Quantite**	Nature	P.U
152	147	Dictaphone numerique olympus dp -20 1GB mémoire 1H 4Dossiers de 100 fichiers			50 000
153	148	GPS garmi 70			140 000
154	149	pointeur laser wireless présentateur (logitech R400) pour vidéo projecteur			25 000
155	150	Ph mètre et conductimètre en 1			140 000
156	151	Adaptateur pour Mac/air			15 000
157	152	Adaptateur pour Mac/air (8 en 1)			15 000
158	153	Adaptateur USB/VGA			5 000
159	153	Adaptateur Mac : Hub Usb-basix T9 (9 in 1) type-C vers Usb3.0x3 avec sortie hdmi, lecteur de carte Sd, Micro Sd, Rj45, Vga pour ordinateur et téléphone de type C	16		25 000
160	154	Cable HDMI			2 500
161	155	masque nasal chrirugical		boite de 50	40 000
162	156	masque barriere lavable		unite	400
163	157	modem routeur wifi			25 000
164	158	routeur wifi			35 000
165	159	Switch 24 ports			35 000

Figure 2: Extrait 2 du fichier Excel (2021-2022) pour l'expression du besoin du personnel de l'UFR ST

	Total				
	Produit d'entretien				
N°	Désignation	Quantite	Nature	P.U	P.T
205	Déodorant véhicule (Stella Skyline)	20		2200	
206	ESSUIE PIED	5		2500	
207	INSECTICIDE GM		Bombe	2000	
208	LAVE VITRE	20		1500	
209	MOUCHOIR POUR VEHICULE			850	
210	PAPIER HIGIENIQUE		sachet de 4	850	
211	LAVE MAIN TOILETLE		bouteille	750	
212	gel sanitaire	5	flacon	1500	
213	DEODORANT BUREAU		bombe	1500	
214	Bloc désodorisant pour WC avec support accroché au bord de la chaise		Boite (tavolette) de 3	1500	
215	SAVON EN POUDRE AJAX GM 700 GRS		boite	1800	

Figure 3: Extrait 3 du fichier Excel (2021-2022) pour l'expression du besoin du personnel de l'UFR ST

	Total				
	Matériel informatique				
N°	**DESIGNATION**	**Quantite**	Nature	Prix U	M
225	Imprimante hp laserjet pro 404 dn cf réseau noir/blanc avec cable français et deux cartouches en plus de la cartouche d'essai			343 000	
226	Imprimante multifonction laser jet Pro MFP M 479 fdw/couleur avec cable français et 2 jeux de cartouches en plus des cartouches d'essai			678 000	
227	Imprimante couleur laserjet pro M254nw, réseau avec cable français et deux jeux de cartouches en des cartouches d'essai			470 165	
228	Imprimante 3D Ender-3 V2 avec pilotes pas à pas TMC2208 silencieux nouvelle interface utilisateur et 4.3 pouces couleur Lcd Carborundum lit en verre imprimante 3D			300 000	
229	Chargeur pour photocopieur ir 2016			345 000	
230	Chargeur pour photocopieur konica c360			450 000	
	Ordinateur fixe complet (Unité centrale, Ecran (21 à 24 pouce), clavier azerty iso,) processeur 2.9 à 4Ghz intel core i7 (7e ou 8e Génération), RAM 4 à 8Go, disque dur 500 à1000Go, lecteur de cartes mémoir, graveur dvd, Windows 10 Professional avec Licence, cable				

Figure 4 : Extrait 4 du fichier Excel (2021-2022) pour l'expression du besoin du personnel de l'UFR ST

Cependant, ce système présente également des limites, telles que le travail pénible de fusion des commandes des enseignants, le manque de suivi des commandes pour le personnel, le manque de statistiques, et le risque de perte de données en raison de l'utilisation d'un fichier Excel. Ces limites peuvent entraver l'efficacité globale du système et affecter la satisfaction des enseignants et du personnel.

Le processus actuel de gestion des commandes est complexe et pénible pour le responsable des services généraux, le chef de département, le directeur de laboratoire et le personnel, ce qui peut entrainer des erreurs et des retards.

II. Les limites de l'existant

Le système actuel basé sur l'utilisation du fichier Excel présente plusieurs limites qui rendent son utilisation peu pratique et peu fiable.

Au cours du processus cité précédemment, nous avons identifiés plusieurs limites tels que :

1. Les limites pour le personnel

- Pas conviviale, ne suscite pas l'envie de passer une commande.

Le fichier Excel présente des difficultés en termes de navigation, ce qui peut rendre la tâche fastidieuse et décourager les utilisateurs. De plus, l'absence d'une interface conviviale rend ardue la recherche rapide et efficace des information nécessaires. Par conséquent, il est essentiel de mettre en place un système plus convivial et intuitif, qui facilitera la navigation et permettra aux utilisateurs de trouver aisément les informations dont ils ont besoin.

- Absence de suivi disponible pour les Commandes passées.

Dans un fichier Excel, il n'y a pas de système de notification pour informer le personnel de l'état de leur commande. Par conséquent, les utilisateurs doivent contacter leur chef de département qui, à son tour, contacte le responsable des services généraux pour obtenir des informations sur l'état de leur commande. Cette absence de suivi automatique et de notification entraîne un processus de communication supplémentaire et peut causer des retards dans la communication des informations relatives aux commandes.

2. Pour le chef de département

Ces limites peuvent avoir un impact significatif sur l'efficacité de leurs opérations et leur capacité à répondre aux besoins

- Une gestion par envoie multiples de mails.

Une gestion par envoi multiple de mails peut présenter des inconvénients tels que des erreurs d'envoi ou des oublis. Dans le cas d'un chef de département qui envoie le fichier Excel a ces collègues enseignants-chercheurs, cela peut entraîner des problèmes tels que :

- Des erreurs d'envoi : il est possible que certaines adresses e-mail ne soient pas correctement saisies, ou que des problèmes techniques empêchent certains e-mails d'être envoyés. Cela peut entraîner des retards ou des incompréhensions quant à la réception du fichier, ce qui peut perturber la communication entre les différents intervenants.
- Des oublis : dans le cas où le chef de département envoie le fichier Excel à tous les collègues enseignants-chercheurs, il peut arriver qu'il oublie d'en envoyer à certains d'entre eux. Cela peut entraîner des frustrations ou des sentiments de négligence chez les personnes qui ne reçoivent pas le fichier.

- La difficulté à suivre l'avancement des commandes.

La difficulté à suivre l'avancement des commandes est une limite importante associée à l'utilisation d'un fichier Excel pour gérer les commandes. Bien que le fichier Excel puisse être utilisé pour enregistrer les détails des commandes, il ne fournit pas les outils nécessaires pour suivre facilement l'avancement des commandes.

En effet, le fichier Excel ne permet pas de suivre en temps réel l'état des commandes. Les informations relatives à l'avancement des commandes sont souvent dispersées dans différents onglets du fichier Excel, ce qui peut rendre difficile leur suivi et leur mise à jour. Cela peut entraîner des retards dans le traitement des commandes.

- La difficulté à enregistrer les données de manière structurée et accessible.

La difficulté à enregistrer les données de manière structurée et accessible est une limite importante associée à l'utilisation d'un fichier Excel pour gérer les commandes. En effet, bien que le fichier Excel puisse être utilisé pour enregistrer les données de commande, il n'offre pas les fonctionnalités avancées nécessaires pour faciliter la recherche, l'analyse et l'accès aux données.

Par exemple, l'utilisation d'un fichier Excel peut rendre difficile la recherche de données spécifiques ou l'analyse de données pour obtenir des informations importantes sur les commandes en cours. Il peut également être difficile de trier et de filtrer les données en fonction de différents critères, ce qui peut rendre la gestion des commandes plus compliquée et moins efficace.

De plus, il peut être difficile de garantir la qualité des données enregistrées dans le fichier Excel. Les erreurs de saisie et les données dupliquées peuvent compromettre la précision et la fiabilité des informations stockées dans le fichier Excel.

- La fusion des commandes est un travail fastidieux.

Le chef de département doit passer du temps à fusionner les différentes commandes des enseignants de son département, ce qui peut être pénible et chronophage. Cela peut également entrainer des erreurs ou des omissions dans la fusion des commandes.

- Absence de tableau de bord.

Le chef de département et le directeur de laboratoire n'ont pas de tableau de bord pour suivre l'état des commandes, ce qui peut rendre la gestion plus difficile et augmenter le risque

d'erreurs. Ils ne peuvent pas facilement savoir quelles commandes ont été passées, quelles sont en cours de traitement, ou encore quelles sont été livrées.

- Absence d'historique des commandes passées par le département.

L'absence d'un historique des commandes précédentes peut rendre difficile l'identification de tendances de commandes et la prévision des besoins futurs du département. Cela peut également rendre difficile l'analyse des erreurs de commandes et les problèmes de livraison.

- Des pertes de données.

L'utilisation d'un fichier Excel pour la gestion des commandes peut entrainer la perte de données en cas de dysfonctionnement du fichier ou de perte accidentelle. Les données stockées dans un fichier Excel ne sont pas toujours protégées contre les erreurs humaines. En raison de la nature manuelle du processus, il est possible de commettre des erreurs lors de la saisie des données, de les modifier ou de les supprimer par inadvertance. De plus, le fichier Excel peut être sujet à des problèmes techniques tels que des pannes de système, des erreurs de sauvegarde ou des erreurs de manipulation, ce qui peut entraîner une perte irréversible de données.

- Les statistiques sont absentes.

L'absence de statistique et de rapports sur les commandes passées, les fournisseurs utilisés, les couts et les délais de livraison peut rendre difficile l'évaluation des performances du département et la prise de décision éclairée. Les informations sur les statistiques peuvent aider à identifier les problèmes et à élaborer des plans d'action pour améliorer les performances.

- Difficulté à mettre à jour et à synchroniser les données.

Il est fastidieux de mettre à jour et de synchroniser les données d'un fichier Excel envoyé par plusieurs personnes, ce qui peut entrainer des erreurs et des retards.

3. Pour le responsable des services généraux

Le responsable des services généraux à une responsabilité clé dans la gestion des commandes, et il doit coordonner les besoins des différents départements pour placer des commandes groupées auprès des fournisseurs. Cependant, il peut y avoir des limites dans le système de commande qui compliquent cette tâche pour le responsable des services généraux, ce qui peut avoir un impact négatif sur la qualité des commandes.

- Une gestion par envoie multiples de mails.

Une gestion par envoi multiple de mails peut présenter des inconvénients tels que des erreurs d'envoi ou des oublis. Dans le cas du responsable des services généraux qui envoie le fichier Excel à tous les chefs de département pour qu'ils puissent le partager avec leurs collègues enseignants-chercheurs, cela peut entraîner des problèmes tels que :

- ❖ Des erreurs d'envoi : il est possible que certaines adresses e-mail ne soient pas correctement saisies, ou que des problèmes techniques empêchent certains e-mails d'être envoyés. Cela peut entraîner des retards ou des incompréhensions quant à la réception du fichier, ce qui peut perturber la communication entre les différents intervenants.
- ❖ Des oublis : dans le cas où le responsable des services généraux envoie le fichier Excel à tous les chefs de département, il peut arriver qu'il oublie d'en envoyer à certains d'entre eux. Cela peut entraîner des frustrations ou des sentiments de négligence chez les personnes qui ne reçoivent pas le fichier.

- ➢ La difficulté à suivre l'avancement des commandes.

La difficulté à suivre l'avancement des commandes est une limite importante associée à l'utilisation d'un fichier Excel pour gérer les commandes. Bien que le fichier Excel puisse être utilisé pour enregistrer les détails des commandes, il ne fournit pas les outils nécessaires pour suivre facilement l'avancement des commandes.

En effet, le fichier Excel ne permet pas de suivre en temps réel l'état des commandes. Les informations relatives à l'avancement des commandes sont souvent dispersées dans différents onglets du fichier Excel, ce qui peut rendre difficile leur suivi et leur mise à jour. Cela peut entraîner des retards dans le traitement des commandes.

- ➢ La difficulté à enregistrer les données de manière structurée et accessible.

La difficulté à enregistrer les données de manière structurée et accessible est une limite importante associée à l'utilisation d'un fichier Excel pour gérer les commandes. En effet, bien que le fichier Excel puisse être utilisé pour enregistrer les données de commande, il n'offre pas les fonctionnalités avancées nécessaires pour faciliter la recherche, l'analyse et l'accès aux données.

Par exemple, l'utilisation d'un fichier Excel peut rendre difficile la recherche de données spécifiques ou l'analyse de données pour obtenir des informations importantes sur les commandes en cours. Il peut également être difficile de trier et de filtrer les données en fonction

de différents critères, ce qui peut rendre la gestion des commandes plus compliquée et moins efficace.

De plus, il peut être difficile de garantir la qualité des données enregistrées dans le fichier Excel. Les erreurs de saisie et les données dupliquées peuvent compromettre la précision et la fiabilité des informations stockées dans le fichier Excel.

- Travail pénible de fusion des commandes des départements.

Le responsable des services généraux doit recueillir les besoins de chaque département, puis les fusionner en une seule commande pour passer la commande auprès du ou des fournisseurs. Si le système de commande ne permet pas une fusion automatique des commandes, Le responsable des services généraux doit le faire manuellement, ce qui peut être long et fastidieux. Cela peut entrainer des retards de commandes, des erreurs de commande, ou encore une perte de temps et d'énergie pour le responsable des services généraux.

- Absence de tableau de bord.

Le responsable des services généraux doit être en mesure de suivre facilement l'état des commandes, et les besoins en fournitures, afin de planifier les commandes futures et d'éviter les pénuries de fourniture. Si le système de commande ne dispose pas d'un tableau de bord clair et facile à utiliser, le responsable des services généraux peut avoir du mal à suivre les commandes essentielles pour la gestion des commandes.

- Absence d'historique des commandes des départements et des UFR.

L'historique des commandes des différents départements et UFR est une source d'information essentielle pour le responsable des services généraux. Cela permet de comprendre les tendances d'utilisation des fournitures, les préférences des utilisateurs, les besoins de fourniture, etc. Si le système de commande de fourniture ne dispose pas d'un historique complet des commandes, il peut être difficile pour le responsable des services généraux de comprendre les tendances et d'adapter les commandes en conséquence.

- Des pertes de données.

L'utilisation d'un fichier Excel pour la gestion des commandes peut entrainer la perte de données en cas de dysfonctionnement du fichier ou de perte accidentelle. Les données stockées dans un fichier Excel ne sont pas toujours protégées contre les erreurs humaines.

- Les statistiques sont absentes.

L'absence de statistique et de rapports sur les commandes passées, les fournisseurs utilisés, les couts et les délais de livraison peut rendre difficile l'évaluation des performances du département et la prise de décision éclairée. Les informations sur les statistiques peuvent aider à identifier les problèmes et à élaborer des plans d'action pour améliorer les performances.

Apres avoir identifié les lacunes du système actuel, il est important de proposer une solution efficace qui permettra de remédier à ces problèmes. En effet, en prenant en compte les limites existantes, il est possible de concevoir une solution plus solide et complète qui répondra aux besoins actuels. En conséquence, en proposant une solution qui prend en compte les insuffisances du système actuel, nous pourrons corriger ces lacunes et améliorer les résultats globaux.

III. Solution proposée

Après avoir identifié les difficultés dans la procédure actuelle de gestion de l'expression des besoins, notre solution consiste à concevoir et à implémenter un système de dématérialisation complet du système actuel qui corrigera les manquements et les défaillances notées.

Le nouveau système de dématérialisation permettra de remplacer le processus manuel basé sur le fichier Excel par un système automatisé et numérique. Voici les points clés de notre solution:

En backend, le système sera réalisé selon l'approche des microservices, qui est une approche architecturale et organisationnelle du développement logiciel. Dans cette approche, l'application est composée de petits services indépendants qui communiquent entre eux via des API bien définies. Les architectures basées sur les microservices simplifient la mise à l'échelle et accélèrent le développement des applications, ce qui permet d'optimiser l'innovation et les délais de mise sur le marché.

Voici une brève description de ces quatre (4) API métiers, qui joueront des rôles spécifiques dans le processus de gestion de l'expression des besoins.

- **API UTILISATEUR :** cette API sera conçue pour gérer les utilisateurs. Elle fournira des fonctionnalités telles que la création, la mise à jour et la récupération des informations d'utilisateur. Elle pourra également être utilisée pour la gestion des identifications de connexion et de la sécurité. Les développeurs pourront utiliser cette API pour intégrer la gestion des utilisateurs dans leur propre application.
- **API CATALOGUE :** cette API sera conçue pour gérer les produits et catégories. Elle fournit des fonctionnalités telles que la création, la récupération de la liste des produits et des catégories, la mise à jour des informations produits et catégories et la gestion des images associées aux produits et catégories. Les développeurs pourront utiliser cette API pour intégrer la gestion des produits et catégories dans leur propre application.
- **API COMMANDE :** cette API sera conçue pour gérer les commandes. Elle fournira des fonctionnalités telles que la création, la mise à jour et la récupération des informations de commandes. Les développeurs pourront utiliser cette API pour intégrer la gestion des commandes dans leur propre application.
- **API STATISTIQUE :** cette API sera conçue pour fournir des statistiques sur les différents aspects de l'application. Elle utilisera Python pour fournir une solution puissante pour l'analyse de données. Elle pourra être utilisée pour générer des graphiques, des tableaux et des données des statistiques sur les différents aspects de l'application, tels que les commandes, les utilisateurs enregistres, etc. Elle pourra également fournit des données sur l'utilisation de l'application, ce qui pourra être utiliser pour optimiser les performances et améliorer l'expérience utilisateur. Les développeurs pourront utiliser cette API pour intégrer les statistiques dans leur propre application. Ces différentes API travailleront de manière indépendante mais interagiront entre elles pour assurer le bon fonctionnement du système global. Elles communiqueront via des interfaces bien définies, utilisant des protocoles pour faciliter l'échange de données.

En **frontend**, nous avons développé des pages web dédiées à la gestion des commandes pour le personnel de l'UASZ en utilisant les services fournis par les API mises en place en backend. Ces pages offrent une interface conviviale et facile à utiliser, permettant aux enseignants de passer leurs commandes de manière rapide et efficace. Elles automatisent également certaines tâches telles que le suivi de l'état des commandes ou leur traitement. Enfin, elles permettent de centraliser et de sécuriser les données de commande en les stockant dans une base de données.

Les interfaces du client web sont conçues de manière ergonomique, en mettant l'accent sur l'expérience utilisateur.

- Une page de connexion.
- Des pages pour la gestion des utilisateurs et des rôles.
- Des pages pour la gestion des UFR, départements et laboratoires.
- Une page gestion des catégories.
- Une page gestion des produits.
- Une page de gestion des statuts.
- Une page gestion de panier.
- Une page gestion des commandes.
- Une page de suivi des commandes.
- La gestion du budget destiné au matériel.
- La possibilité d'imprimer les commandes au format CSV.

En combinant un backend basé sur des microservices avec un frontend convivial et intuitif, notre solution vise à fournir une expérience utilisateur optimale pour la gestion des commandes du personnel de l'UASZ. Cela permettra d'améliorer l'efficacité du processus de commande, de réduire les erreurs et les retards, et de garantir la satisfaction des enseignants dans leurs besoins en matière de matériel pédagogique.

IV. Les Objectifs du mémoire

L'objectif général de ce mémoire est de réaliser la solution proposée ci-dessus, pour l'automatisation de la gestion de l'expression des besoins du personnel de l'UFR ST et de ses départements. Cette solution sera basée sur une architecture en microservices et permettra aux personnels de gérer efficacement leurs commandes via une interface intuitive et des fonctionnalités adaptées. Elles pourront également accéder à leurs commandes à partir de n'importe quel ordinateur ou appareil mobile et suivre l'avancement de leurs commandes, ce qui facilitera le travail en équipe et améliorera la transparence de la gestion des commandes. L'objectif final est de remplacer le processus actuel de gestion des commandes basé sur un fichier Excel par une solution numérique moderne, efficace et accessible de n'importe où pour le personnel du département informatique. Cela permettra de résoudre les problèmes et les limitations rencontrés avec l'utilisation du fichier Excel, tels que la complexité de la navigation, l'absence de notifications sur l'état des commandes et les risques de perte de données. La

solution proposée offrira une interface conviviale, une automatisation des tâches et une centralisation sécurisée des données de commande.

Les objectifs spécifiques de ce mémoire sont les suivants :

- créer des microservices techniques tels que le service de configuration, le service d'enregistrement (Eureka) et le service proxy (Gateway) afin de faciliter la communication entre les différents microservices métiers ;
- développer les microservices métiers tels que l'API commande, l'API catalogue, l'API utilisateur et l'API statistique en utilisant une architecture de microservices pour améliorer la flexibilité et la scalabilité du système ;
- assurer une communication efficace entre les différents microservices en utilisant des protocoles standardisés tels que RESTful API ;
- mettre en place des tests unitaires et des tests d'intégration pour garantir la qualité du code et la fiabilité du système ;
- déployer les microservices sur une plateforme cloud pour garantir une haute disponibilité et une meilleure gestion des ressources ;
- surveiller les microservices en temps réel en utilisant des outils de surveillance et de gestion de logs pour garantir une haute disponibilité et une performance optimale ;
- évaluer les performances du système en termes de temps de réponse, de disponibilité et de scalabilité pour identifier les domaines d'amélioration et de développement futurs.

Conclusion

L'expression des besoins des enseignants a été étudiée pour déterminer les limites actuelles et les opportunités d'amélioration. La solution proposée vise à résoudre les défis actuels et à offrir une meilleure expérience pour le personnel.

L'objectif général est de moderniser et de simplifier les processus de collecte et de gestion des besoins des enseignants, tout en garantissant l'exactitude et la pertinence des informations. Les objectifs spécifiques comprennent la mise en place d'une plateforme de dématérialisation efficace, la simplification des processus administratifs et la mise à disposition d'outils en ligne pour les enseignants.

Le chapitre suivant se concentrera sur l'architecture générale et le choix technique pour la mise en œuvre de la dématérialisation des besoins des enseignants. Ce chapitre couvrira les aspects clés de l'architecture, tels que les systèmes et technologie associées. Les décisions de choix

technique seront prises en fonction de critères clés tels que la performance, la scalabilité, la fiabilité et la sécurité. L'objectif est de garantir la réussite du projet de dématérialisation des besoins des enseignants en utilisant les meilleurs pratiques et les technologies les plus adaptées.

CHAPITRE II : ARCHITECTURE GENERALE ET CHOIX METHODOLOGIQUES

Introduction

L'architecture est un élément essentiel dans la conception et la création d'une application. Elle permet de définir les modèles et les techniques utilisés pour structurer et organiser l'application de manière optimale. En suivant une architecture bien définie, on peut s'appuyer sur une feuille de route claire et respecter les meilleures pratiques pour assurer la qualité et la performance de l'application.

Dans ce chapitre, nous aborderons l'architecture choisie pour le développement du backend de notre application, qui est basée sur une approche microservices. Dans la première partie du chapitre, nous présenterons en détail l'architecture microservices, en expliquant ses principes fondamentaux et ses avantages. Nous discuterons également de la manière dont cette architecture répond aux besoins spécifiques de notre solution pour la gestion des besoins des enseignants.

Dans la deuxième partie du chapitre, nous aborderons les choix techniques réalisés dans le cadre de notre architecture. Nous examinerons les différentes technologies et outils utilisés pour mettre en œuvre les microservices, en prenant en compte des critères tels que la performance, la scalabilité, la fiabilité et la sécurité. Nous expliquerons également comment ces choix techniques sont alignés sur les objectifs du projet et contribuent à la réussite de la dématérialisation des besoins des enseignants.

I. Architecture microservices

Avec les architectures monolithiques, tous les processus sont étroitement couplés et s'exécutent comme un système unique. Cela signifie que si l'un des processus de l'application enregistre un pic de demande, toute l'architecture doit être mise à l'échelle. L'ajout ou l'amélioration de fonctionnalités d'une application monolithique devient plus complexe au fur et à mesure de la croissance de la base de code. Cette complexité limite les expérimentations et rend l'implémentation de nouvelles idées difficiles. Les architectures monolithiques augmentent les risques en matière de disponibilité des applications, car de nombreux processus dépendants et étroitement couplés renforcent l'impact d'un seul échec de processus.

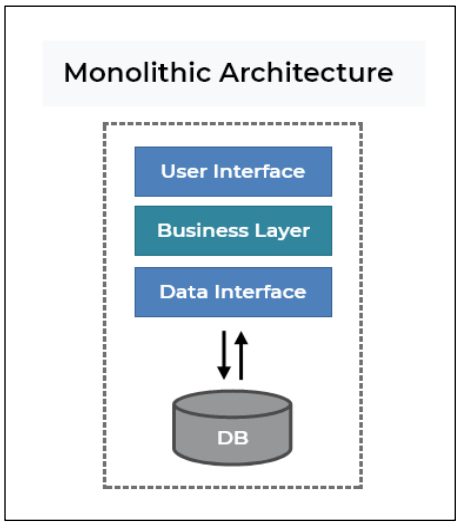


Figure 5: Architecture monolithique

Avec une architecture de microservices, une application est développée à l'aide de composants indépendants, qui exécutent chaque processus de l'application sous forme de service. Ces services communiquent via 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.

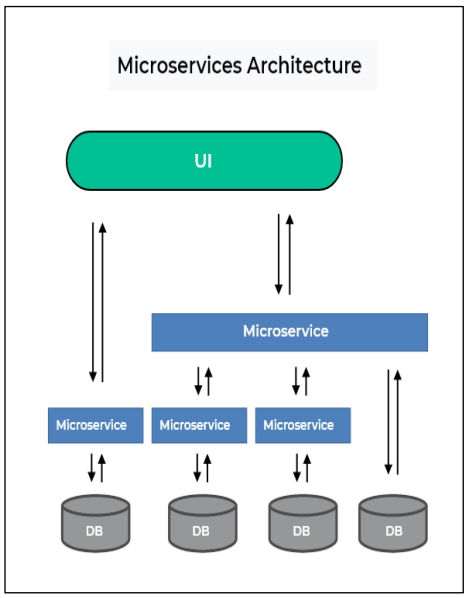


Figure 6: Architecture microservices

Les microservices constituent une approche architecturale et organisationnelle du développement logiciel, dans laquelle le logiciel se compose de petits services indépendants qui communiquent via des API bien définies. Cependant, ces services sont détenus par de petites équipes auto-contenues. Les architectures de microservices simplifient la mise à l'échelle et accélèrent le développement des applications, pour des innovations et des délais de mise sur le marché optimisés. L'architecture microservice présente plusieurs points forts.

- **Composants indépendants** : premièrement, tous les services peuvent être déployés et mis à jour indépendamment, ce qui donne plus de flexibilité. Deuxièmement, un bogue en microservices n'a d'impact que sur ce service particulier et n'influence pas l'ensemble de l'application. Enfin, il est beaucoup plus facile d'ajouter de nouvelles fonctionnalités à une application microservices qu'à une application monolithique.
- **Une compréhension plus aisée** : divisée en composants plus petits et plus simples, une application microservice est plus facile à comprendre et à gérer. Il suffit de se concentrer sur un service spécifique lié à un objectif commercial que vous avez.
- **Meilleure évolutivité** : un autre avantage de l'approche microservices est que chaque élément peut être mis à l'échelle indépendamment. L'ensemble du processus est donc plus rentable et plus rapide qu'avec les monolithes, où l'application entière doit être mise à l'échelle même si elle n'en a pas besoin. En outre, chaque monolithe a des limites en termes d'évolutivité, de sorte que plus le nombre d'utilisateurs augmente, plus le monolithe pose des problèmes. Par conséquent, de nombreuses entreprises finissent par reconstruire leurs architectures monolithiques.
- **Flexibilité dans le choix de la technologie** : les équipes d'ingénieurs ne sont pas limitées par la technologie choisie dès le départ. Elles sont libres d'appliquer diverses technologies et Framework pour chaque microservice.
- **Le niveau supérieur d'agilité** : toute défaillance dans une application microservices n'affecte qu'un service particulier et non l'ensemble de la solution. Ainsi, tous les changements et toutes les expériences sont mis en œuvre avec moins de risques et moins d'erreurs.[2]

a. Les éléments de notre architecture

L'architecture d'un logiciel décrit la manière dont seront agencés les différents éléments d'une application et comment ils interagissent entre eux.

L'architecture de micro-service désigne un style d'architecture utilise dans le développement d'application. Elle permet de décomposer une application volumineuse en composants indépendants, chaque élément ayant ses propres responsabilités. Pour diffuser la requête d'un utilisateur unique, une application basée sur des microservices peut appeler plusieurs microservices internes pour composer sa réponse.

Notre système sera composé de sept (7) modules ou API dont quatre (4) modules métiers et trois (3) modules techniques permettant de fournir une solution complète pour gérer les utilisateurs, le catalogue, les commandes, les statistiques, le service de configurations, la découverte de service (Eureka) et le service de passerelle (proxy). Les sept (7) API sont les suivantes :

- **API UTILISATEUR (SPRING SECURITY) :** cette API sera conçue pour gérer les utilisateurs. Elle utilisera Spring Sécurité pour fournir des fonctionnalités de sécurité telles que l'authentification et l'autorisation des utilisateurs. Elle pourra être utilisée pour gérer les identifiants de connexion, les mots de passe, les rôles d'utilisateurs et les autorisations associées ;
- **API CATALOGUE (SPRING BOOT) :** cette API sera conçue pour gérer les produits et catégories. Elle utilisera Spring Boot pour fournir une architecture de développement rapide et flexible. Elle sera utilisée pour gérer les informations produits et catégories, les images associées aux produits et catégories et les mises à jour associées ;
- **API COMMANDE (SPRING BOOT) :** cette API sera conçue pour gérer les commandes. Elle utilisera Spring Boot pour fournir une architecture de développement rapide et flexible. Elle sera utilisée pour gérer les informations de commande, les mises à jour et les récupérations associées ;
- **API STATISTIQUES (PYTHON) :** cette API sera conçue pour fournir des statistiques sur les différents aspects de l'application. Elle utilisera Python pour fournir une solution puissante pour l'analyse de données. Elle pourra être utilisée pour générer des graphiques, des tableaux et des données statistiques sur les commandes, les utilisateurs enregistres, etc ;
- **CONFIG SERVICE (SPRING CLOUD) :** cette API sera conçue pour gérer les configurations de l'application. Elle utilisera Spring Cloud pour fournir une solution de configuration distribuée pour les différents composants de l'application. Spring Cloud Config, permet d'éviter de bloquer l'application à chaque mise à jour de la configuration. Il permet de centraliser tous les fichiers de configuration dans un dépôt GIT et se positionner comme serveur de fichiers de configuration. Ainsi, lorsqu'un fichier de configuration est mis à jour dans le dépôt GIT, Spring Cloud Config se met à servir cette nouvelle version, obligeant le micro-service à le prendre en compte à la volée ;

- **DISCOVERY SERVICE (SPRING CLOUD)** : cette API sera conçue pour fournir une solution de découverte de service pour les différents composants de l'application. Elle utilise également Spring Cloud pour fournir une solution de découverte de service distribuée pour les différents composants de l'application. Spring boot nous offre la dépendance Spring Cloud Discovery (Eureka), Eureka va permettent l'enregistrement des instances de Microservices en vue d'être découvertes par d'autres Microservices. Nous devons aussi ajouter la dépendance Config Client qui permet aux clients de se connecter au serveur Spring Cloud Config pour récupérer la configuration de l'application. Chaque microservice qui se démarre va se connecter avec le service d'enregistrement pour publier sa référence (nom, adresser IP, port) ;
- **GATE WAY (SPRING CLOUD) :** cette API sera conçue pour fournir une passerelle pour les différents composants de l'application. Elle utilise également Spring Cloud pour fournir une solution passerelle flexible et évolutive. Elle centralise les points d'entrée de l'application, gérer les requêtes entrantes et fournir un accès sécurisé aux différents composants de l'application. Le service proxy (Gateway), va faire l'orchestration autrement-dit toutes les requêtes qui arrivent généralement du partie frontend vont être à destination du service proxy. Le service proxy se débrouille pour rediriger ou bien forwarder la requête vers le bon service. Le service proxy à chaque requête qui arrive il envoi la requête vers le service d'enregistrement pour récupérer l'adresse du service, après il peut le contacter pour exécuter l'action qui a été demande au niveau de la requête.

Nous avons utilisé httpClient pour la communication entre les microservices. Cela a permis une gestion efficace des requêtes et des réponses entre les différents composants du système. HttpClient offre une interface flexible et facile à utiliser pour les requêtes réseau, ce qui a contribué à simplifier la mise en place de la communication inter-microservices. Http Client est un module Java qui permet d'effectuer des requêtes http et d'interagir avec des services Web. Il peut être utilisé pour envoyer des demandes GET, POST, PUT, DELETE, etc. à un serveur et pour recevoir les réponses de ce dernier. Pour utiliser httpClient, il est d'abord nécessaire d'ajouter une dépendance dans le projet Java. Ensuite, il est possible de créer une instance d'httpClient et d'utiliser ses méthodes pour envoyer des requêtes HTTP à un serveur.

b. Architecture générale de la solution

L'architecture présentée ci-dessous décrit le système que nous avons conçu. Ce système est constitué de sept microservices différents, à savoir : Catalogue, Commande, Utilisateur, Statistique, ainsi que les microservices techniques Proxy, Registry et Config.

Lorsque les microservices Catalogue, Commande, Utilisateur et Statistique sont démarrés, ils récupèrent leur configuration à partir du service Config, ce qui permet une centralisation de la configuration via Git. Une fois les microservices démarrés, ils s'enregistrent auprès du service Registry en envoyant leur port, nom et adresse IP. Cette étape est cruciale car elle permet au service Proxy de récupérer ces informations et de rediriger les utilisateurs vers le bon microservice.

Lorsqu'un utilisateur envoie une requête, le service Proxy récupère les informations sur les microservices à partir du service Registry, puis redirige la demande vers le microservice approprié. Cette architecture de microservices permet une communication efficace entre les différents composants du système et garantit une expérience utilisateur de qualité.

En somme, notre architecture de microservices utilise des services tels que Config, Registry et Proxy pour faciliter la communication entre les différents composants du système. Les microservices récupèrent leur configuration à partir du service Config et s'enregistrent auprès du service Registry, tandis que le service Proxy se charge de rediriger les demandes des utilisateurs vers le microservice approprié en utilisant les informations disponibles dans le service Registry.

Architecture globale du système

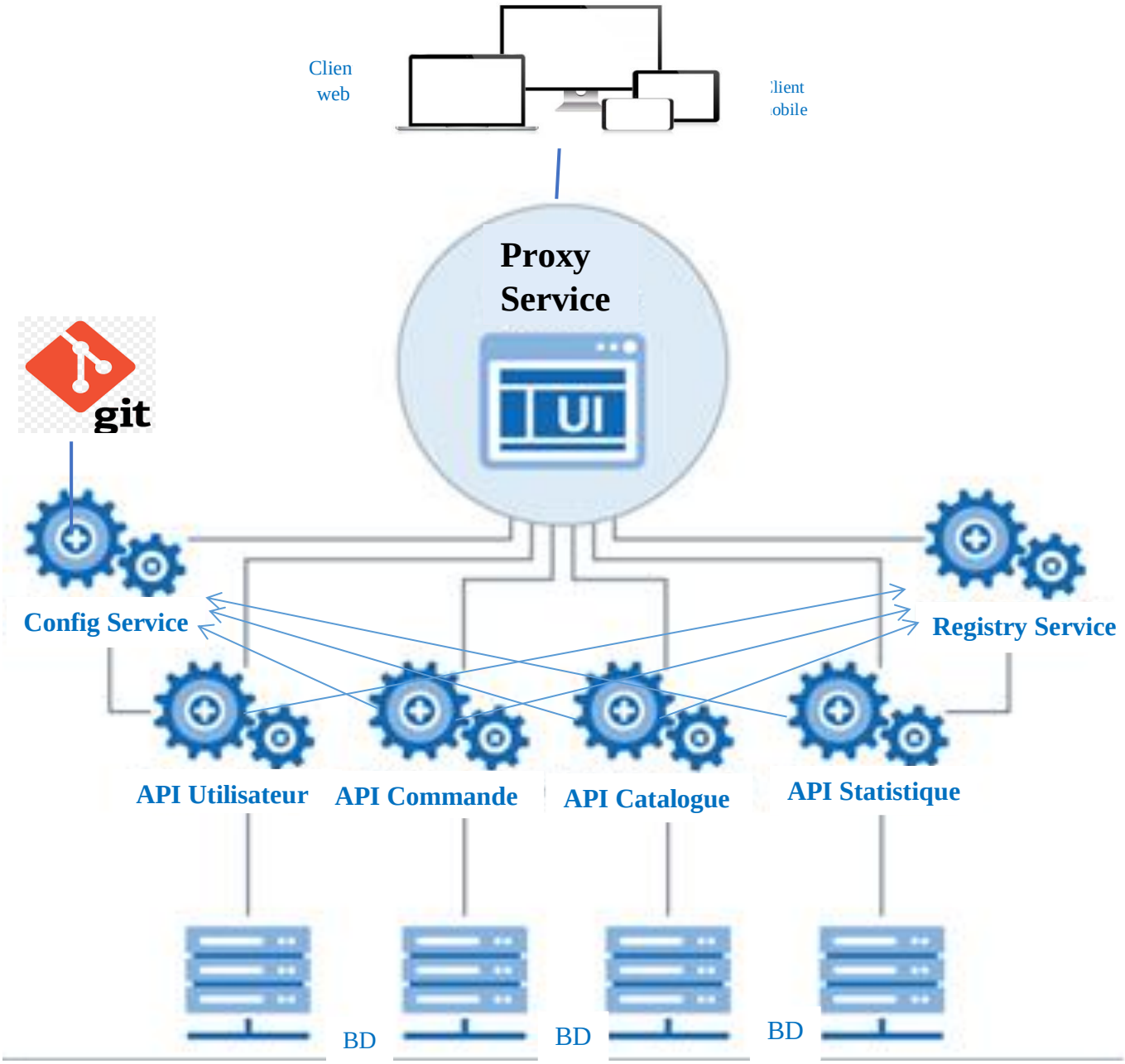


Figure 7: Architecture du système

Pour une bonne gestion de ce projet, nous optons pour la méthode (ou processus) agile DevOps et CI/CD.

II. Méthode agile, DevOps et CI/CD

En information, on parle toujours de projet, et dans chaque projet dont on entend parler, il est développé en suivant une méthode de développent permettant sa réalisation en suivant un

ensemble de processus. Le choix d'une méthode de développement est vital et dépend toujours du type de projet. Dans notre cas, nous adopterons la méthode « Agile ».

a. Méthode agile [3]

La méthodologie Agile s'oppose généralement à la méthodologie traditionnelle "Waterfall", ou dite "de cycle en V". Plus souple et plus flexible, elle place les besoins du client au centre des priorités du projet.

Lors de sa création, cette approche devait servir à la gestion des projets de développement web et informatique. Aujourd'hui, la méthode Agile est de plus en plus répandue. Ce succès s'explique notamment par son adaptabilité à de nombreux types de projets, tous secteurs confondus.

Suite à l'observation d'un taux d'échec élevé des projets dans les années 1990, 17 experts en développement logiciel se réunissent aux Etats-Unis en 2001. Ils veulent mettre en commun leurs méthodes respectives. Le « Manifeste Agile » (*Agile Manifesto* en anglais) naît de cette rencontre. Il détermine les valeurs et les principes fondamentaux de la méthode.

Au cœur de la méthode Agile résident une plus grande implication du client et une meilleure réactivité des équipes. Ce manifeste prône en outre 4 valeurs fondamentales de la démarche :

- L'équipe, soit des individus et des interactions, plutôt que des processus et des outils ;
- L'application, c'est-à-dire des fonctionnalités opérationnelles plutôt que de la documentation exhaustive ;
- La collaboration avec le client, plutôt que la contractualisation des relations ;
- L'acceptation du changement, plutôt que le suivi d'un plan.

De ces valeurs découlent les 12 principes généraux suivants :

1. Satisfaire la clientèle en priorité.
2. Accueillir favorablement les demandes de changement.
3. Livrer le plus souvent possible des versions opérationnelles de l'application.
4. Assurer une coopération permanente entre le client et l'équipe projet.
5. Construire autour de personnes motivées.
6. Privilégier la conversation en face-à-face.

7. Mesurer l'avancement du projet en matière de fonctionnalité de l'application.
8. Faire avancer le projet à un rythme soutenable et constant.
9. Porter une attention continue à l'excellence technique et à la conception.
10. Faire simple.
11. Responsabiliser les équipes.
12. Ajuster à intervalles réguliers son comportement et ses processus pour être plus efficace.

Fonctionnement de la méthode Agile

La méthodologie Agile se base sur une idée simple. Planifier la totalité de votre projet dans les moindres détails avant de le développer est contre-productif.

Vous perdez du temps si vous organisez tous les aspects de votre projet en amont. Il est effectivement rare que tout se passe exactement comme prévu. Souvent, des aléas surviennent et vous forcent à revoir votre planification.

La méthode Agile recommande de se fixer des objectifs à court terme. Le projet est donc divisé en plusieurs sous-projets. Une fois l'objectif atteint, on passe au suivant, et ce jusqu'à l'accomplissement de l'objectif final. Cette approche est plus flexible. Puisqu'il est impossible de tout prévoir et de tout anticiper, elle laisse la place aux imprévus et aux changements.

Autre point important : la méthode Agile repose sur une relation privilégiée entre le client et l'équipe projet. Sa satisfaction étant la priorité, l'implication totale de l'équipe et sa réactivité face aux changements s'imposent. Le dialogue est privilégié. C'est le client qui valide chaque étape du projet. Il convient donc de prendre en compte l'évolution de ses besoins. Des ajustements sont effectués en temps réel afin de répondre à ses attentes.

Avec l'approche Agile, rien n'est figé. L'équipe projet doit être capable de se remettre sans cesse en cause et de chercher continuellement à évoluer [3].

En tant que précurseur de DevOps, et donc de l'intégration, la livraison et le déploiement continus (CI/CD), la méthode agile est étroitement liée à ces approches. Comprendre la philosophie de la méthode agile peut vous aider à mieux exploiter le CI/CD tout en implémentant un pipeline CI/CD mettant en pratique la méthode agile.

b. DevOps et CI/CD

- **Culture DevOps**

DevOps se concentre sur les limites de la culture et des rôles lors du processus de développement agile. L'intention de DevOps est d'éviter l'impact négatif que la surspécialisation et les rôles cloisonnés dans une organisation ont sur l'empêchement d'une réponse rapide ou même efficace aux problèmes de production. Les organisations DevOps font tomber les barrières entre les opérations et l'ingénierie en formant chaque équipe aux compétences de l'autre. Cette approche améliore la capacité de chacun à apprécier et à participer aux tâches de chacun et conduit à une collaboration de meilleure qualité et à une communication plus fréquente.

- **Intégration continue/livraison continue (CI/CD)**

L'intégration continue (IC) est une pratique de génie logiciel où les membres d'une équipe intègrent leur travail avec une fréquence croissante. Conformément à la pratique de l'IC, les équipes s'efforcent d'intégrer au moins quotidiennement et même toutes les heures, approchant l'intégration qui se produit "en continu".

Historiquement, l'intégration a été une activité d'ingénierie coûteuse. Ainsi, pour éviter le thrash, CI met l'accent sur les outils d'automatisation qui pilotent la construction et les tests, en se concentrant finalement sur la réalisation d'un cycle de vie défini par logiciel. Lorsque CI réussit, l'effort de construction et d'intégration diminue, et les équipes peuvent détecter les erreurs d'intégration aussi rapidement que possible.

La livraison continue (CD) est à l'empaquetage et au déploiement ce que CI est à construire et à tester. Les équipes qui pratiquent le CD peuvent créer, configurer et empaqueter des logiciels et orchestrer son déploiement de manière à ce qu'il puisse être mis en production d'une manière définie par logiciel (faible coût, automatisation élevée) à tout moment.

Des pratiques CI/CD performantes facilitent directement le développement agile, car les modifications logicielles atteignent la production plus fréquemment. En conséquence, les clients ont plus d'occasions d'expérimenter et de donner leur avis sur le changement.

Qu'est-ce que le CI/CD dans DevOps ? Et comment sont-ils liés à Agile ?

Comment CI/CD, Agile et DevOps sont-ils liés dans le développement réel ? Les équipes d'ingénierie commencent souvent par CI parce que c'est dans leur timonerie. Une concentration DevOps peut aider les organisations à comprendre la configuration, le packaging et l'orchestration nécessaires pour définir encore plus le logiciel du cycle de vie, créant ainsi une

pratique CD plus précieuse. La pratique du CI/CD dans DevOps, à son tour, ajoute au développement agile.

Voici un moyen simple et rapide de différencier Agile, DevOps et CI/CD :

- **Agile** se concentre sur les processus mettant en évidence le changement tout en accélérant la livraison.
- **CI/CD** se concentre sur les cycles de vie définis par logiciel mettant en évidence des outils qui mettent l'accent sur l'automatisation.
- **DevOps** se concentre sur la culture mettant en évidence les rôles qui mettent l'accent sur la réactivité.[4]

Il existe deux types d'approche pour développera une api.il y a l'approche code first qui consiste à commencer la spécification de l'API par le code et l'approche contract-First qui consiste à commencer par le contrat d'API.

III. Contract-first avec OpenApi et SwaggerHub

L'approche contract-first signifie fondamentalement que tout effort d'API, qu'il s'agisse d'une ou de plusieurs dans un programme, commence par un processus de conception. Dans ce modèle, les API sont définies de manière itérative que les humains et les ordinateurs peuvent comprendre, avant même qu'aucun code ne soit écrit. L'objectif est que chaque équipe parle le même langage et que chaque outil utilisé exploite la même conception d'API. La différence cruciale ici par rapport à une approche API-first est que, bien que l'API soit extrêmement importante, le processus de conception est ce qui garantit que toutes les parties prenantes sont impliquées et que leurs besoins sont satisfaits lors de la création.

Design-First commence avec des individus techniques et non techniques de chacune des fonctions impliquées participant au processus de rédaction d'un contrat qui définit l'objectif et la fonction de l'API (ou de l'ensemble d'API). De toute évidence, cette approche nécessite un certain temps initial consacré à la planification. Cette phase vise à s'assurer que lorsque vient le temps de commencer à coder, les développeurs écrivent du code qui n'aura pas besoin d'être supprimé et réécrit plus tard. Cela permet de créer des API itératives et utiles qui, à leur tour, conduisent à un meilleur programme d'API plus évolutif - et à la valeur pour votre entreprise - dans son ensemble.[5]

L'approche contract-first a de nombreux avantages :

- Favorise l'expérience développeur et la collaboration avec les autres équipes (front, Ops, etc.) facilite l'expression du besoin avec les responsables services généraux et minimise le risque d'incompréhensions au cours de développement.
- Permet aux équipes de développement de travailler en parallèle en utilisant un serveur de mock.
- Permet l'utilisation d'outils de génération de code pour générer des serveurs et des clients divers langages de programmation.
- Permet l'utilisation d'outils de génération de documentation pour écrire une doc.
- Permet l'utilisation de nombreux outils. [6]
- Réutilisabilité.
- Séparation des préoccupations.
- Les équipes de développement ne se bloquent pas.
- Bonne et constante communication entre les équipes.
- Modèles d'API cohérents.

SwaggerHub est une plate-forme en ligne où vous pouvez concevoir vos API REST -qu'il s'agisse d'API publiques, d'API privées internes ou de microservices. Le principe de base de SwaggerHub est Design First, Code Later**,** Autrement dit, vous commencez par disposer votre API, ses ressources, ses opérations et ses modèles de données, et une fois la conception terminée, vous implémentez la logique métier.

Les définitions d'API sont écrites au format OpenAPI (anciennement connu sous le nom de Swagger). Ils sont enregistrés dans le cloud SwaggerHub et peuvent être synchronisés avec des systèmes externes tels que GitHub ou Amazon API Gateway. Vous pouvez également collaborer avec votre équipe sur SwaggerHub et maintenir plusieurs versions d'API au fur et à mesure de son évolution.

Avec SwaggerHub, vous pouvez :

- Définissez vos API au format OpenAPI.
- Hébergez toutes vos définitions d'API en un seul endroit.
- Stockez vos composants d'API communs (tels que les modèles de données et les réponses) dans des domaines et référencez-les à partir des définitions d'API.
- Collaborez sur les définitions d'API avec votre équipe.
- Générez du code serveur et client et transmettez-le à GitHub, GitLab, Bitbucket ou Azure DevOps Services.

- Partagez vos API publiquement et en privé.
- Itérer la conception de l'API et gérer plusieurs versions d'API.

OpenAPI Spécification (anciennement Swagger Spécification) est un format de description d'API pour les API REST. Un fichier OpenAPI vous permet de décrire l'intégralité de votre API, notamment.

- Points de terminaison disponibles (/users) et opérations sur chaque point de terminaison (GET /users, POST /users) ;
- Paramètres de fonctionnement Entrée et sortie pour chaque opération.
- Méthodes d'authentification.
- Coordonnées, licence, conditions d'utilisation et autres informations.

Les spécifications de l'API peuvent être écrites en YAML ou JSON. Le format est facile à apprendre et lisible à la fois pour les humains et les machines. La spécification OpenAPI complète est disponible sur GitHub : Spécification OpenAPI 3.0

Conclusion

Le choix de l'architecture microservice, associe à la méthodologie Agile et à la culture DevOps implémenté par CI/CD, ainsi que la définition des contrats avec OpenAPI SwaggerHub, représente une avancée significative vers une approche moderne et efficace du développement et de la livraison de logiciels. Les microservices offrent une plus grande flexibilité et une évolutivité accrue, tandis que la méthodologie Agile, DevOps et CI/CD garantie une livraison rapide et fiable des fonctionnalités. La définition des contrats avec OpenAPI SwaggerHub assure une interopérabilité et une collaboration efficaces entre les différents services.

Le chapitre suivant se concentre sur la spécification et la conception de l'API Utilisateur. Ce processus impliquera une analiyse approfondie des exigences des utilisateurs pour déterminer les fonctionnalités nécessaires et les spécifications techniques pour implémenter l'API Utilisateur. L'implémentation de l'API Utilisateur sera réalisée en utilisant les meilleures pratiques et les technologies les plus récentes, pour garantir une qualité élevée et une performance optimale. Enfin, le déploiement de l'API Utilisateur sera effectué avec succès en utilisant des processus éprouvés pour une livraison sans problème aux utilisateurs finaux.

CHAPITRE III : SPECIFICATION ET CONCEPTION DE L'API UTILISATEUR

Introduction

La spécification et l'analyse des besoins fonctionnels servent à identifier les acteurs du système (ensemble d'éléments interagissant entre eux selon un certain nombre de règles) et à associer chacun l'ensemble des fonctionnalités avec lesquelles il intervient dans l'objectif de donner un résultat optimal et satisfaisant au client.

Nous débutons ce chapitre en élaborant une spécification, à l'aide des outils Swagger et OpenAPI, des exigences fonctionnelles que l'API utilisateur doit satisfaire. Nous poursuivrons ensuite avec la conception de l'API, en utilisant la spécification comme point de départ.

I. SPECIFICATION

La spécification est un élément crucial du processus de développement de l'API, car elle définit les fonctionnalités de base qui doivent être incluses dans l'API pour répondre aux besoins de l'utilisateur. La spécification nous permet également de travailler de manière collaborative avec les parties prenantes pour nous assurer que nous avons une compréhension claire des exigences fonctionnelles avant de passer à la phase de conception.

Les **figures 8,9,10 et 11** ci-dessous représentent une partie de la spécification de l'API Utilisateur.

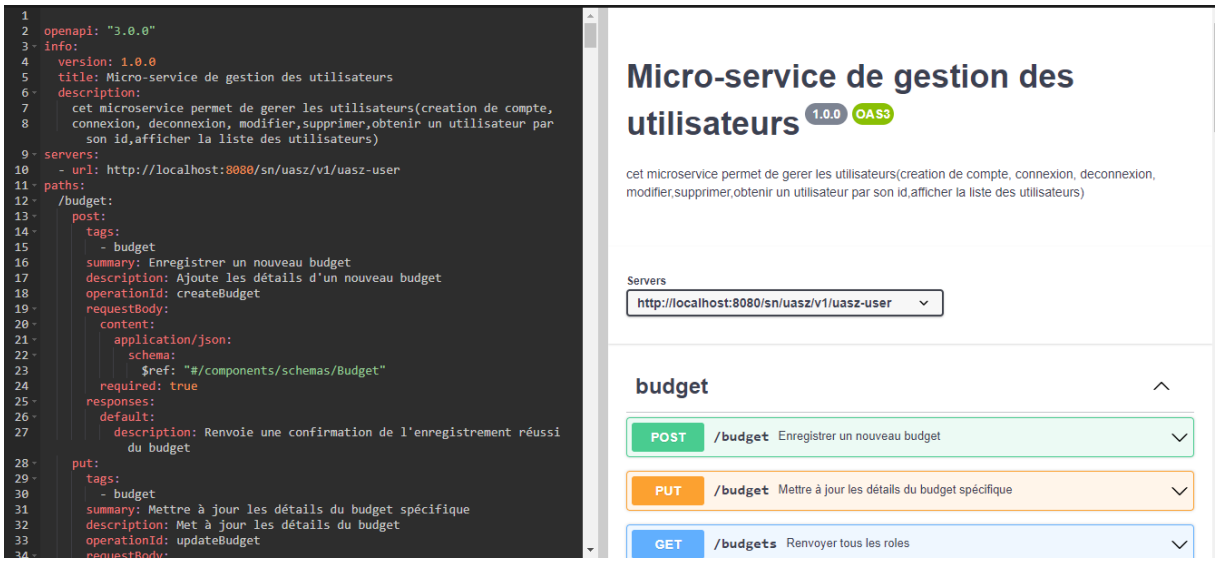


Figure 8: Figure : Extrait 1 de la spécification de l'API UTILISATEUR

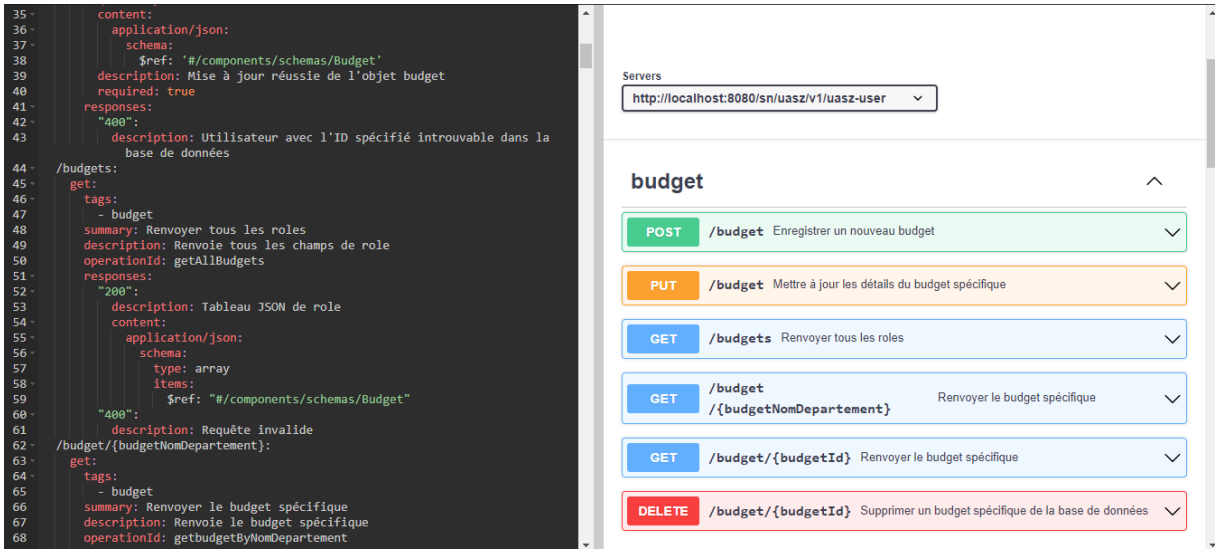


Figure 9: Extrait 2 de la spécification de l'API UTILISATEUR

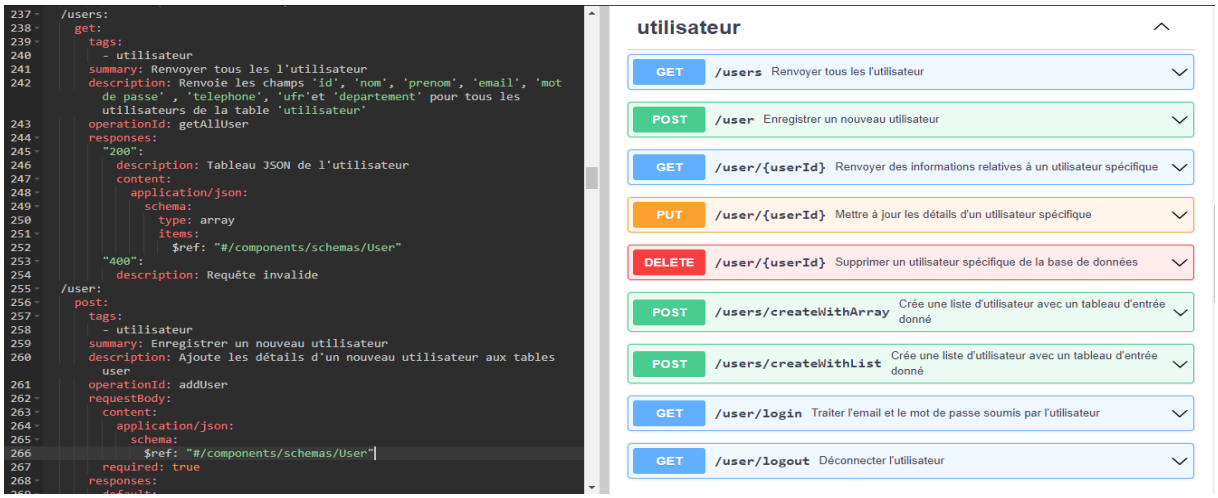


Figure 10: Extrait 3 de la spécification de l'API UTILISATEUR

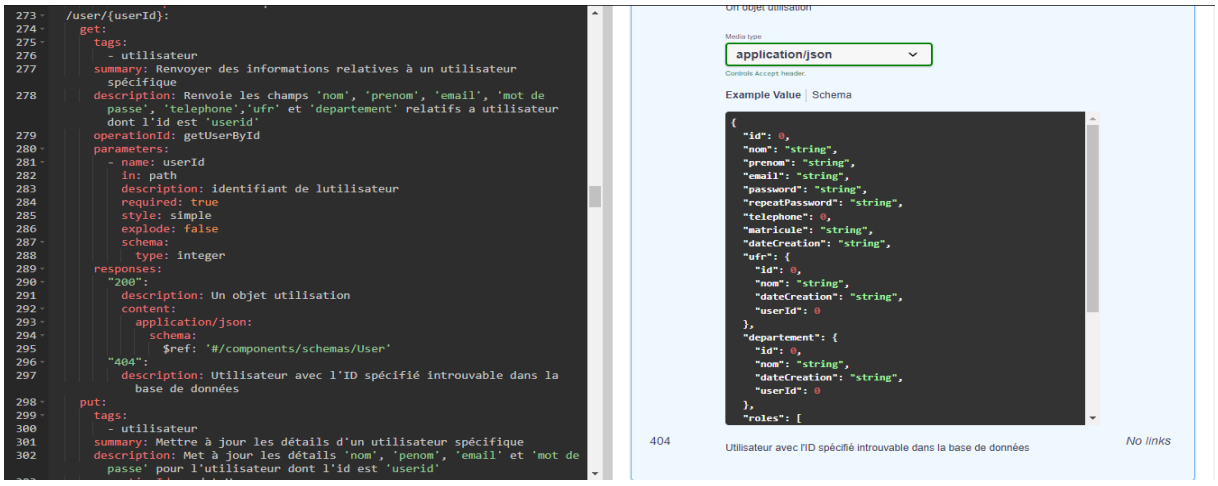


Figure 11: Extrait 4 de la spécification de l'API UTILISATEUR

Tout d'abord, cette spécification est au format YAML pour décrire une API RESTful. Elle commence par spécifier la version d'OpenAPI utilisée (3.0.0), suivie d'informations générales sur l'API, notamment son titre, sa description et sa version.

Ensuite, elle spécifie un serveur sur lequel l'API est déployée (http://localhost:8080/sn/uasz/v1/uasz-user). Le bloc suivant définit les différentes routes de l'API, chacune avec des opérations HTTP telles que GET, POST, PUT et DELETE. Chaque opération est décrite avec un résumé, une description, un ID d'opération et une liste de paramètres de requête.

Par exemple, la première route /budget définit une opération POST pour créer un nouveau budget avec les détails fournis dans le corps de la requête. Elle spécifie également une opération PUT pour mettre à jour les détails d'un budget existant et une opération GET pour récupérer tous les budgets dans un tableau JSON.

Les autres routes sont similaires, avec des opérations GET, POST, PUT et DELETE pour gérer les utilisateurs, les rôles, les UFR, les départements et les budgets. Chaque opération spécifie les paramètres requis et les réponses possibles, y compris les codes HTTP et les descriptions.

Enfin, la spécification contient une section components qui définit les schémas de données utilisés dans l'API, tels que Budget, Utilisateur, Rôle, UFR et Département. Ces schémas décrivent les champs de données nécessaires pour créer ou mettre à jour des utilisateurs, des rôles, des budgets des UFR ou des départements.

Cette spécification OpenAPI Swagger décrit une API RESTful pour le microservice de gestion des utilisateurs. Elle fournit une documentation claire et précise des différentes routes, des paramètres et des réponses possibles pour les opérations HTTP.

L'avantage d'une telle spécification est qu'elle permet de documenter clairement l'Api pour les développeurs qui souhaitent l'utiliser. Elle permet également de générer automatiquement une documentation interactive, ainsi que du code client et serveur à partir de la spécification. Cela facilite grandement le processus de développement, car les développeurs peuvent utiliser la documentation générée pour comprendre rapidement comment interagir avec l'API et commencer à écrire du code sans avoir à passer trop de temps à comprendre la structure de l'API.

De plus, la spécification Swagger/OpenAPI facilite la communication entre les équipes de développement et les parties prenantes du projet, telles que les chefs de produit, car elle fournit

une documentation claire et précise de l'API. Elle permet également de maintenir l'API à jour plus facilement, car les modifications apportées à la spécification peuvent être utilisées pour régénérer la documentation interactive et le code client et serveur.

Afin de répondre aux points précis du cahier des charges, et bien comprendre les fonctionnalités de notre api, nous identifions les besoins des acteurs, nous produisons les diagrammes de cas d'utilisation.

1. Identification des acteurs

Un acteur peut être considérer comme une entité externe interagissant avec le système à travers des actions sur ce dernier. Dans notre cas on considère deux types acteurs, les acteurs principaux et les acteurs secondaires. Les premières sont ceux qui le système rend service et les seconds sont solliciter pour des informations supplémentaires. Le **tableau 2** ci-dessous montre l'ensemble des acteurs intervenant dans l'API UTILISATEUR

Tableau 2 : les acteurs de l'API UTILISATEUR

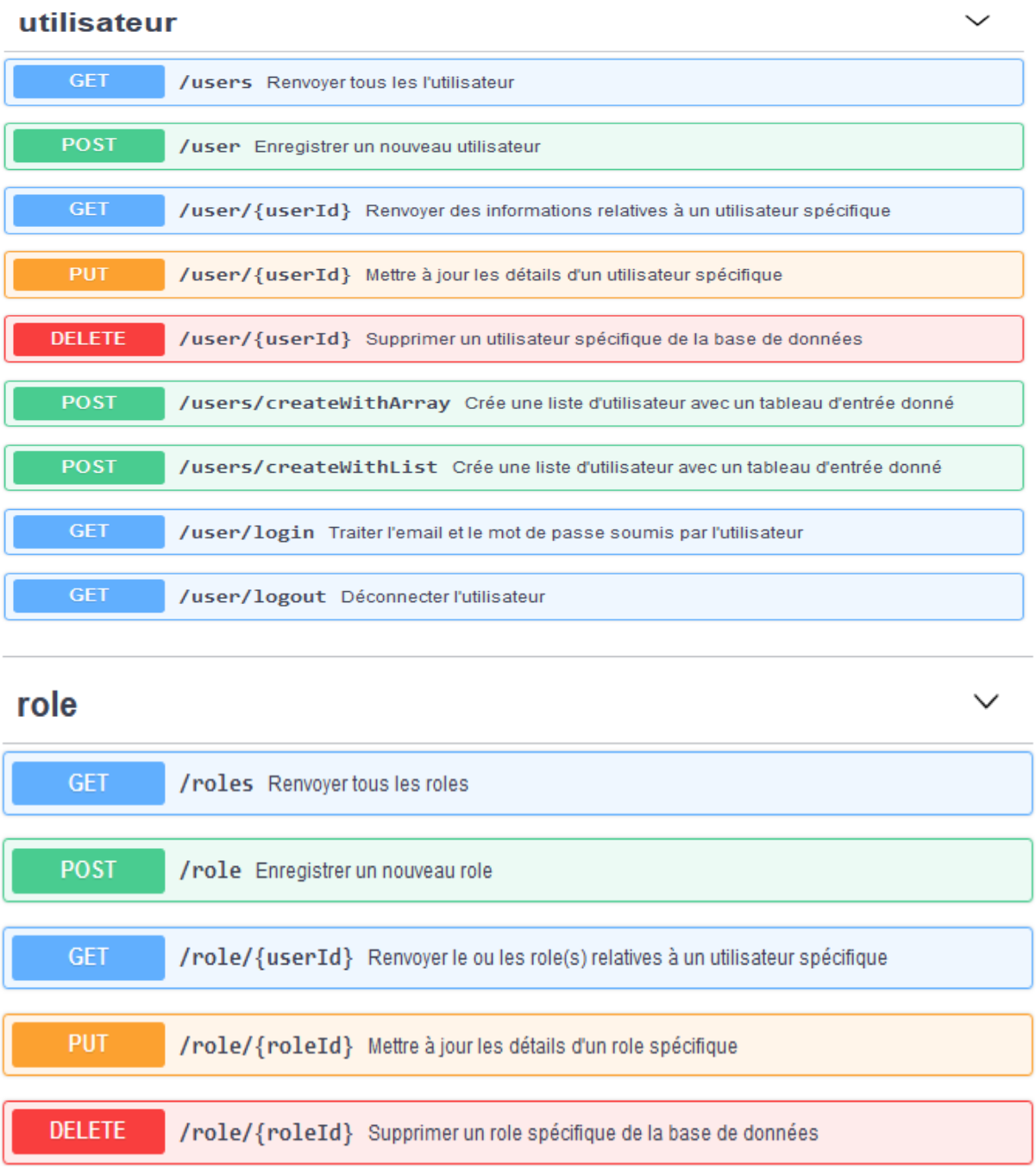
Acteurs	Rôles
Administrateur	Gérer les utilisateurs, les rôles, les UFR, les départements et les statuts des commandes
Enseignant	Exprimer son besoin par des commandes en ligne
Secrétaire	Exprimer son besoin par des commandes en ligne
Chef de département	Gérer les commandes de son département
Directeur laboratoire	Gérer les commandes de son laboratoire
Responsable des services généraux	Gérer les commandes de son UFR

2. Identification des fonctionnalités

Les fonctionnalités sont les actions ou service 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.

Lorsque l'on crée une API, il est important de s'assurer que les données qui y sont échangées sont correctement structurées et organisées. Pour cela, les développeurs peuvent utiliser la

documentation Swagger, qui est un outil de spécification d'API permettant de décrire les opérations qui sont disponibles via cette API. La **figure 12** ci-dessous montre l'ensemble des méthodes que notre api pourra rendre à l'utilisateur.



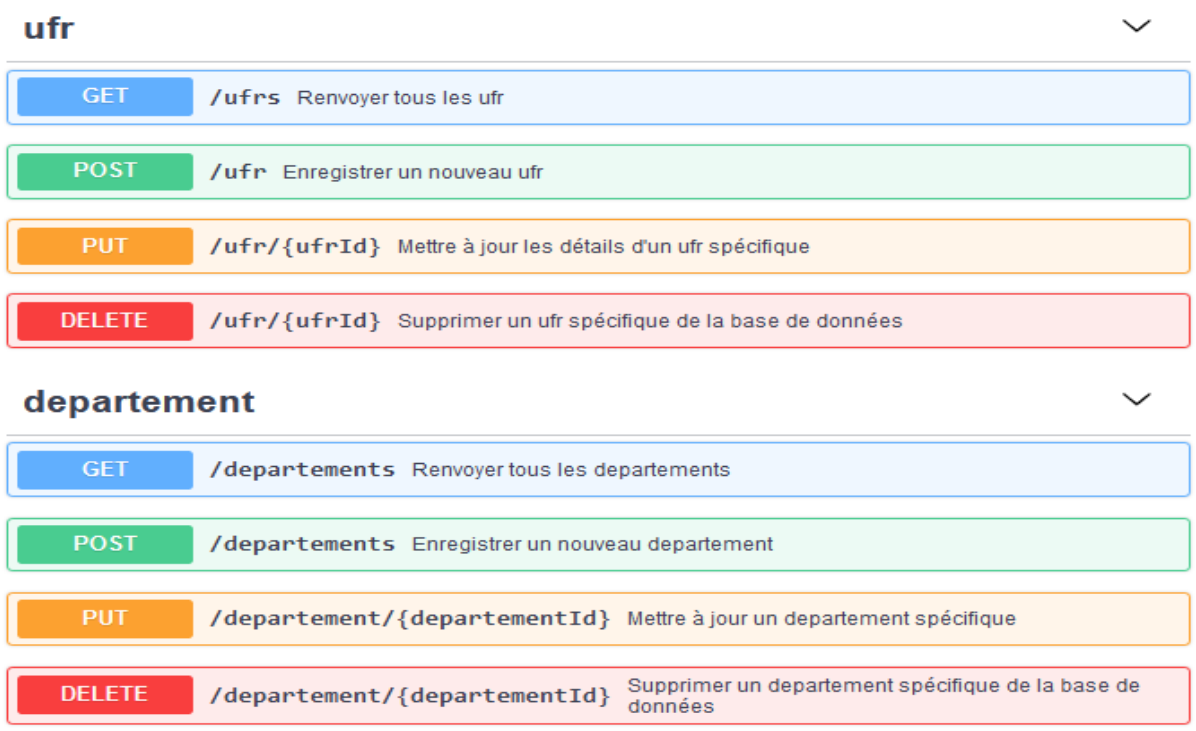


Figure 12: les Endpoint de l'API UTILISATEUR

La figure ci-dessus illustre les différentes méthodes disponibles pour interagir avec l'API utilisateur. Cette visualisation est rendue accessible grâce à la documentation de swagger.

En utilisant la documentation de Swagger, les développeurs peuvent facilement comprendre et naviguer dans les différentes options offertes par l'API utilisateur. La visualisation graphique facile la compréhension et permet de mieux planifier l'intégration de l'API dans un projet donne.

Les diagrammes de cas d'utilisation sont utilisés pour modéliser les besoins. Il permet de représenter les interactions fonctionnelles entre les acteurs et le système, d'organiser et d'identifier les grandes fonctionnalités du système, à formaliser les besoins sous une forme graphique simple pour être compréhensible par toutes les personnes impliquées dans le projet.

a. Diagramme de cas d'utilisateur de l'administrateur

Le diagramme (**figure 13**) ci-dessous montre les fonctionnalités auxquelles l'administrateur a accès à notre système. Les différentes fonctionnalités sont représentées sous forme de cas d'utilisation, qui décrivent une action ou ne tache que l'utilisateur peut effectuer dans le système. Les cas d'utilisation sont reliés à l'acteur, qui est généralement l'utilisateur du système. Par des flèches qui montre la relation entre les différentes fonctionnalités et l'acteur.

Ce diagramme permet de comprendre rapidement et clairement les différentes fonctionnalités disponibles pour l'administrateur et comment elles sont liées les unes aux autres,

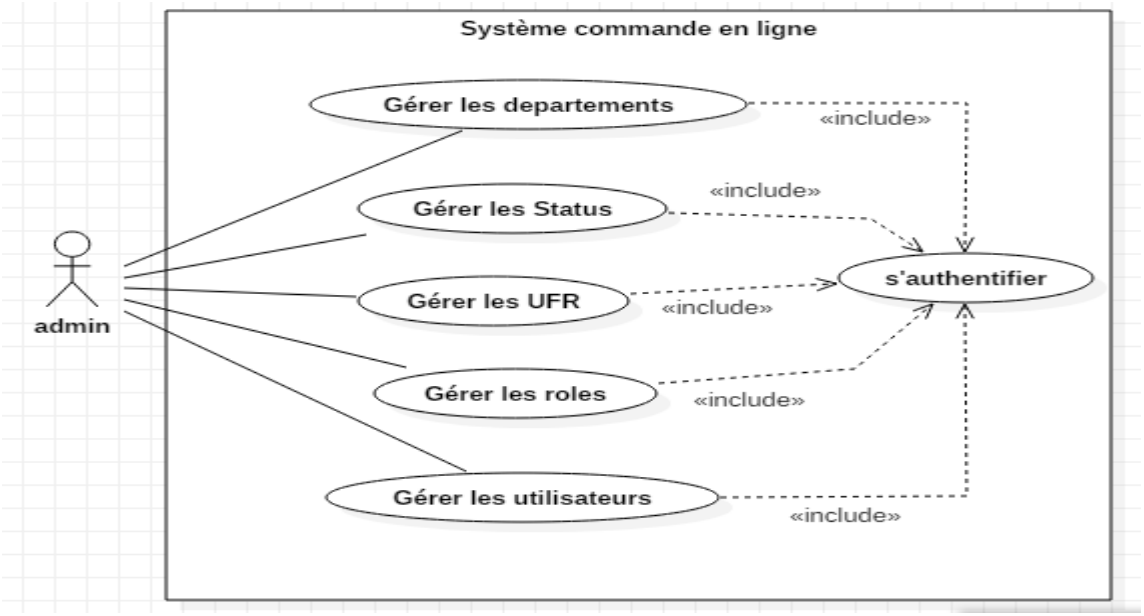


Figure 13: Diagramme de cas d'utilisation de l'administrateur

Dans le diagramme ci-dessus, nous pouvons voir que l'administrateur doit s'authentifier pour pouvoir utiliser les fonctionnalités attribues par le système.

b. Description du cas d'utilisation « s'authentifier »

La description d'un cas d'utilisation décrit en détail une tache ou une interaction spécifique entre l'utilisateur et le système. Dans le cas du cas d'utilisation « s'authentifier », il s'agit de la fonctionnalité qui permet à l'utilisateur de se connecter au système en fournissant des informations d'identification telles que son adresse email et son mot de passe. La description se fera sous forme de tableau. Le **tableau 3** ci-dessous illustre la description de l'authentification.

Tableau 3 : Description du cas d'utilisation "s'authentifier »

Nom	S'authentifier
Acteur(s)	• Chef de département • Responsable des services généraux • Admin
Description	S'authentifier avant une quelconque manipulation du système
Préconditions	L'existence d'un compte utilisateur valide

Démarrage	L'utilisateur lance l'application.
Scenario nominal	• Au lancement de l'application, un formulaire d'authentification est lancé. • Dans ce formulaire, il demande à l'utilisateur de donner son login et son mot de passe. • Si les informations saisies par l'utilisateur sont correctes, il pourra maintenant continuer.
Scenario alternatifs	• Il a la possibilité de réessayer la saisie si par inadvertance qu'il entre un mot de passe ou un login incorrect. • L'accès sera refusé à toute autre personne ne faisant pas partie de la base (qui n'est ni un chef de département, ni un responsable des services généraux, ni un administrateur)
Poste condition	Aucun

3. Analyse des besoins fonctionnels du système

L'analyse des besoins fonctionnels consiste une partie importante dans le processus de développement. Elle permet de voir les différents comportements du système suite à une action demander. Dans cette partie, nous ferons l'analyse de quelque cas d'utilisation en présentant leurs diagrammes de séquence pour savoir comment les éléments du système interagissent entre eux et avec les acteurs.

L'authentification est une partie très importante dans notre système. Le diagramme de séquence nous permet de voir les scenarios possibles du cas « s'authentifier »

a. Diagramme de séquence du cas « s'authentifier »

L'authentification concerne pratiquement tous les utilisateurs du système. Dans cette partie nous allons expliquer comment l'administrateur va procéder pour se connecter.

Il doit s'authentifier en saisissant son email et son mot de passe (fourni par le développeur). Si les données saisies sont correctes, il est redirigé vers sa page d'accueil. Par contre si par inadvertance les données saisies sont incorrectes il aurait la possibilité de ressayer. La **figure 14** ci-dessous illustre le diagramme de séquence du cas d'utilisation « s'authentifier »

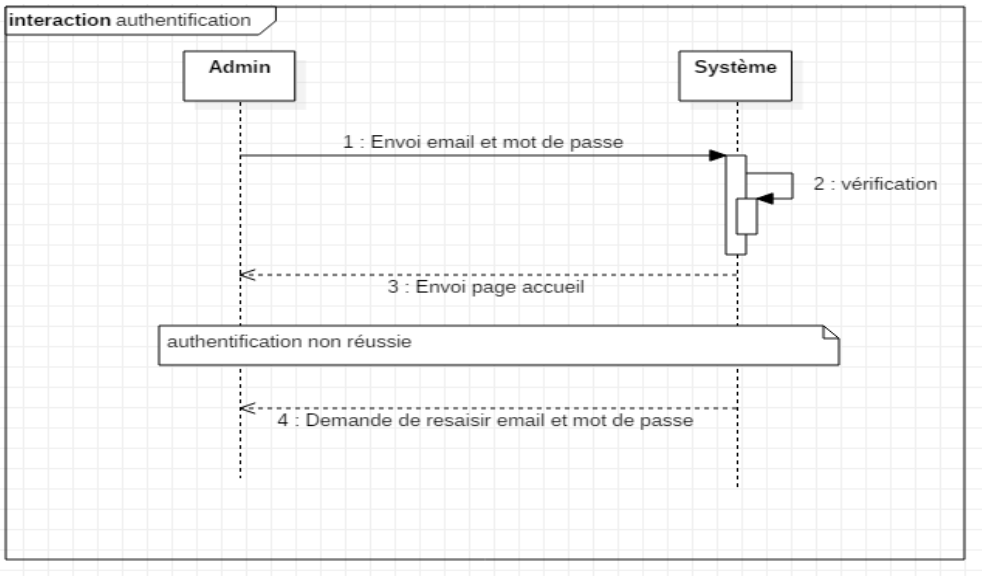


Figure 14: Diagramme de séquence « s'authentifier »

L'inscription comme les autres cas d'utilisation nécessitent un ensemble de séquence pour sa réalisation.

b. Diagramme de séquence du cas « s'inscrire »

L'administrateur demande le formulaire d'inscription, le système affiche le formulaire, en suite il remplit le formulaire, le système vérifie les données du formulaire et l'insert dans la base de données. Si la vérification n'a pas bien passée le système lui envoie un message indiquant l'erreur. La **figure 15** ci-dessous illustre le diagramme de séquence du cas d'utilisation « s'inscrire »

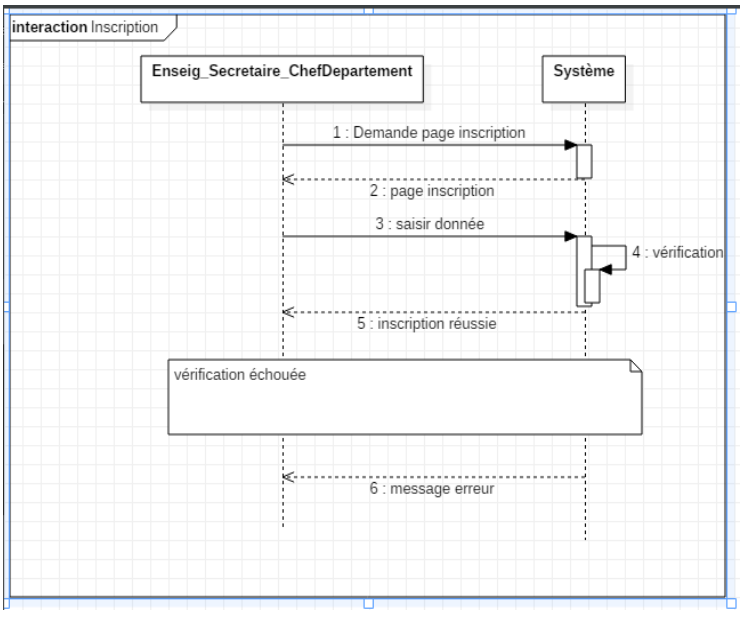


Figure 15: Diagramme de séquence « s'inscrire »

II. CONCEPTION

La conception de l'API est une étape cruciale dans le processus de développement d'une API. Pour faciliter cette étape, il est recommandé d'utiliser des outils tels qu'OpenAPI et Swagger.

OpenAPI est une spécification de format de description d'API qui permet de décrire les fonctionnalités d'une API de manière claire et structurée. Il fournit une manière standardisée de décrire les opérations, les paramètres et les réponses de l'API, ce qui facilite la collaboration entre les différentes parties prenantes impliquées dans le développement de l'API.

Swagger, quant à lui, est un ensemble d'outils pour la conception, la documentation et le test d'API. Il fournit une interface utilisateur conviviale pour la documentation de l'API, ainsi que des outils pour générer du code client et des tests automatisés.

En utilisant OpenAPI et Swagger pour la conception de l'API, les développeurs peuvent bénéficier de plusieurs avantages. Tout d'abord, cela permet de décrire les fonctionnalités de l'API de manière claire et précise, ce qui facilite la compréhension des exigences fonctionnelles par toutes les parties prenantes.

Ensuite, cela permet de faciliter la collaboration entre les différents membres de l'équipe de développement. La documentation de l'API générée avec Swagger est conviviale et facilement accessible, ce qui permet à tous les membres de l'équipe de comprendre rapidement les fonctionnalités et l'utilisation de l'API.

Enfin, cela permet d'améliorer la qualité de l'API. En utilisant une spécification de format de description d'API standardisée, les développeurs peuvent s'assurer que l'API est bien structurée et conforme aux normes établies. De plus, en utilisant les outils de test automatisé fournis par Swagger, les développeurs peuvent garantir que l'API fonctionne correctement et qu'elle est fiable.

En somme, la conception de l'API avec OpenAPI et Swagger est une méthode recommandée pour faciliter le processus de développement de l'API. Cela permet de décrire les fonctionnalités de manière claire et précise, de faciliter la collaboration entre les membres de l'équipe et d'améliorer la qualité de l'API [7].

1. Diagramme de classe

Les diagrammes de classe sont l'un des types de diagramme d'UML les plus utiles, car ils permettent de décrire la structure d'un système en modélisant ses classes, ses attributs ses opérations et les relations entre ses objets.

a. Diagramme de classe « service utilisateur »

Le diagramme (**figure 16**) ci-dessous montre les différentes classes participantes à la réalisation complète du service utilisateur. Les quatre classes sont :

- Utilisateur : Représente un utilisateur du système. Cette classe contient des informations telles que le nom d'utilisateur, le mot de passe, le courriel, etc.
- Rôle : Représente les différents rôles attribués à un utilisateur. Par exemple, un utilisateur peut avoir des rôles tels que "administrateur", "enseignant", etc.
- UFR : Représente les différentes Unités de Formation et de Recherche.
- Département : Représente les différents départements d'une UFR

Les relations entre ces classes peuvent être décrites de la manière suivante :

- Un utilisateur peut avoir plusieurs rôles.
- Un utilisateur peut appartenir à une seule UFR.
- Une UFR peut comprendre plusieurs départements.
- Un utilisateur peut appartenir à un seul département.

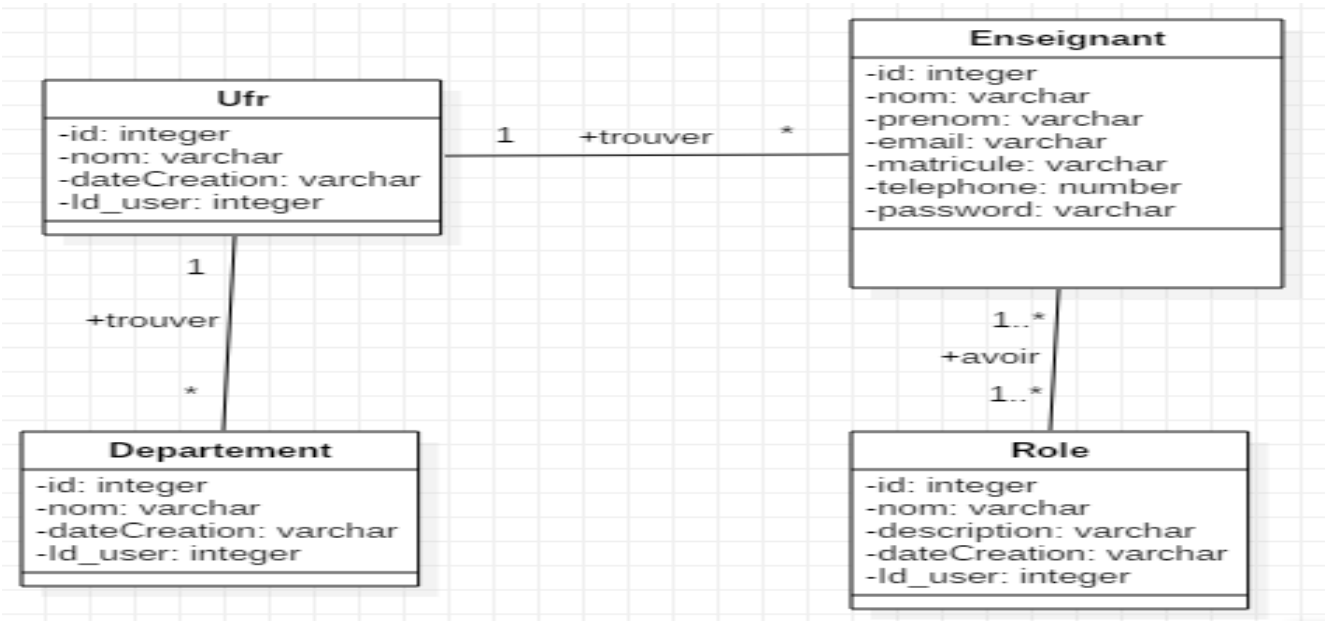


Figure 16: Diagramme de classe « service utilisateur »

2. Schéma de donnée

En se référant à la documentation Swagger, les développeurs peuvent ainsi visualiser le schéma de données de leur API utilisateur. Ce schéma décrit l'ensemble des données qui sont disponibles via l'API, ainsi que leur structure et leur format. Grâce à cette représentation, les développeurs peuvent mieux comprendre les relations entre les différentes données et identifier les erreurs éventuelles dans la structure des données.

L'utilisation de la documentation Swagger permet de rendre l'API plus facile à comprendre et à utiliser pour les développeurs qui souhaitent l'intégrer à leurs applications. En fournissant une documentation claire et structurée, les développeurs peuvent gagner du temps et éviter les erreurs lors de l'intégration de l'API à leur application. La **figure 17** ci-dessous représente l'ensemble des données utilisées pour élaborer l'API UTILISATEUR.

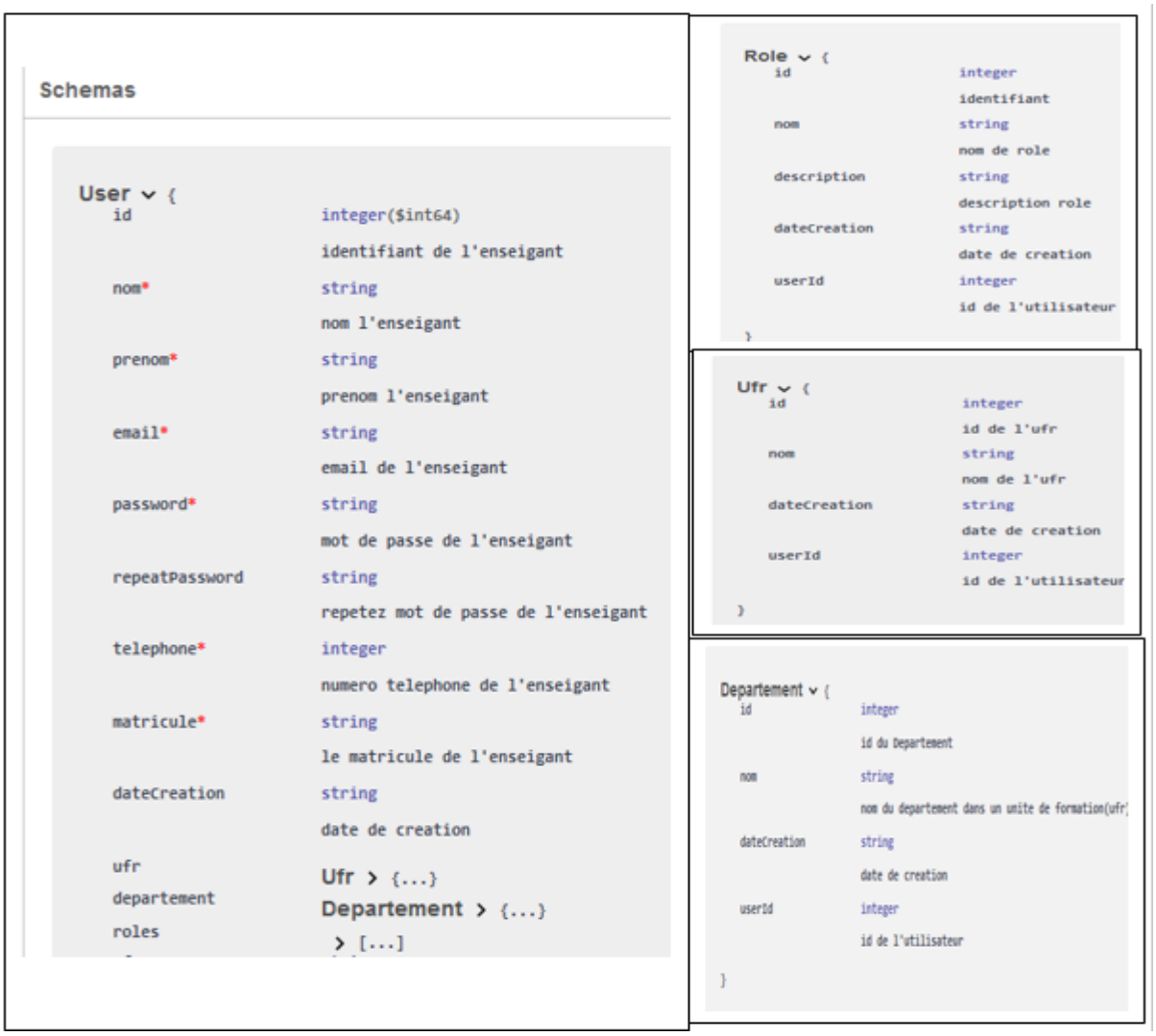


Figure 17: Schéma de donnée défini par Swagger de l'API UTILISATEUR

Conclusion

Le choix d'utiliser la méthode OpenAPI et Swagger pour réaliser la spécification et la conception est judicieux, car ces méthodes ont permis de générer facilement la documentation et le code technique de l'API, ce qui a facilité la communication entre les différentes équipes de développement impliquées dans le projet. De plus, l'application de l'approche contract-first a permis de garantir la cohérence et la compatibilité de l'API. Cette approche assure également que tous les acteurs impliqués dans le projet ont une compréhension claire des fonctionnalités attendues de l'API.

En utilisant la méthode OpenAPI et Swagger pour appliquer l'approche contract-first, l'équipe de développement a pu gagner du temps et améliorer la qualité du projet en réduisant les erreurs et les incohérences. En somme, l'utilisation de ces méthodes et outils a permis d'assurer une qualité optimale de l'interface utilisateur pour l'ensemble du projet.

La spécification et la conception de l'API Utilisateur sont des étapes essentielles pour garantir la qualité de l'interface utilisateur. Cette démarche rigoureuse permet de s'assurer que l'API réponde aux besoins et aux attentes des utilisateurs, offrant une expérience fiable et conforme aux spécifications. Les chapitres suivants, tels que l'API Catalogue, l'API Commande et l'API Statistique, appliquent les mêmes principes que l'API Utilisateur pour assurer une qualité optimale de l'interface utilisateur.

CHAPITRE IV : SPECIFICATION ET CONCEPTION DE L'API CATALOGUE

I. SPECIFICATION

La spécification joue un rôle clé dans le développement de l'API en définissant les fonctionnalités de base qui doivent être intégrées pour répondre aux besoins des utilisateurs. En travaillant de manière collaborative avec les parties prenantes, la spécification permet de clarifier les exigences fonctionnelles avant de passer à la conception de l'API.

Les **figures 18,19,20 et 21** ci-dessous représentent une partie de la spécification de l'API Catalogue.

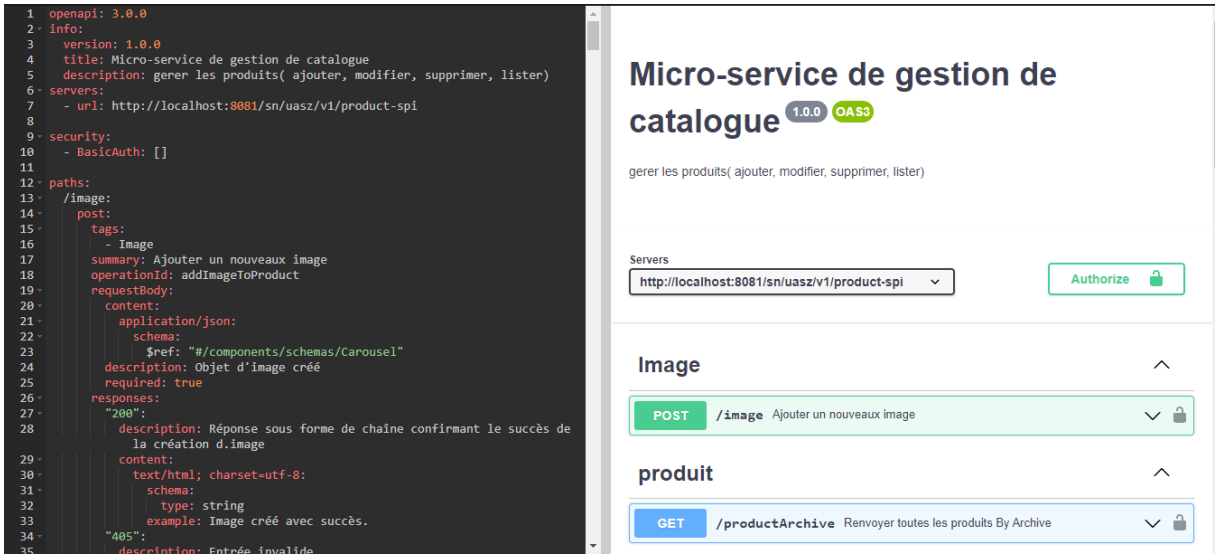


Figure 18: Extrait 1 de la spécification de l'API CATALOGUE

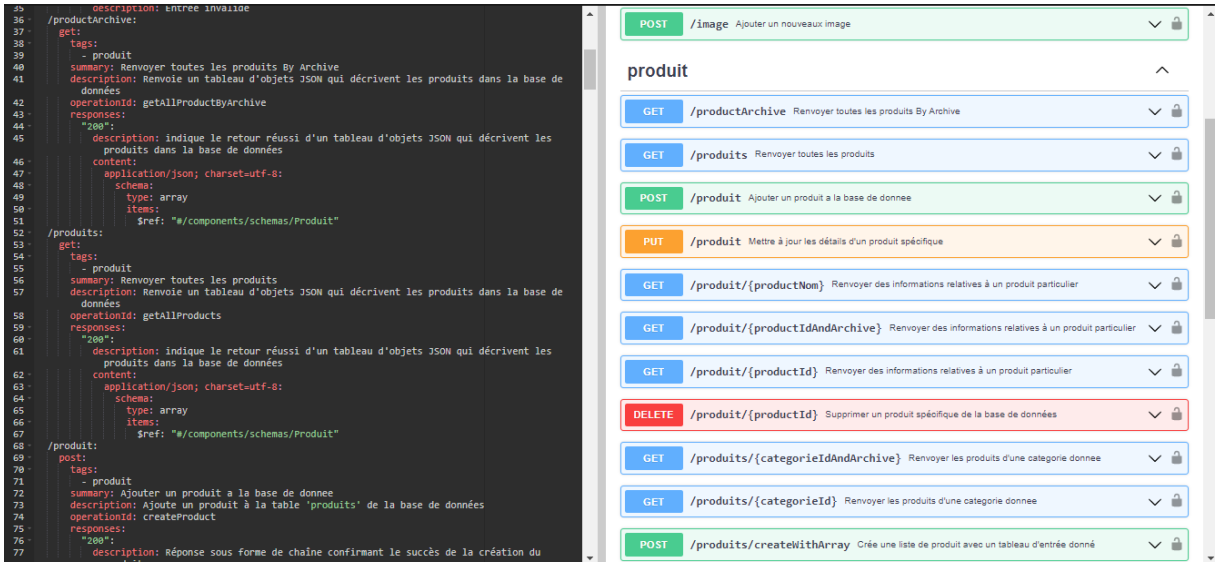


Figure 19: Extrait 2 de la spécification de l'API CATALOGUE

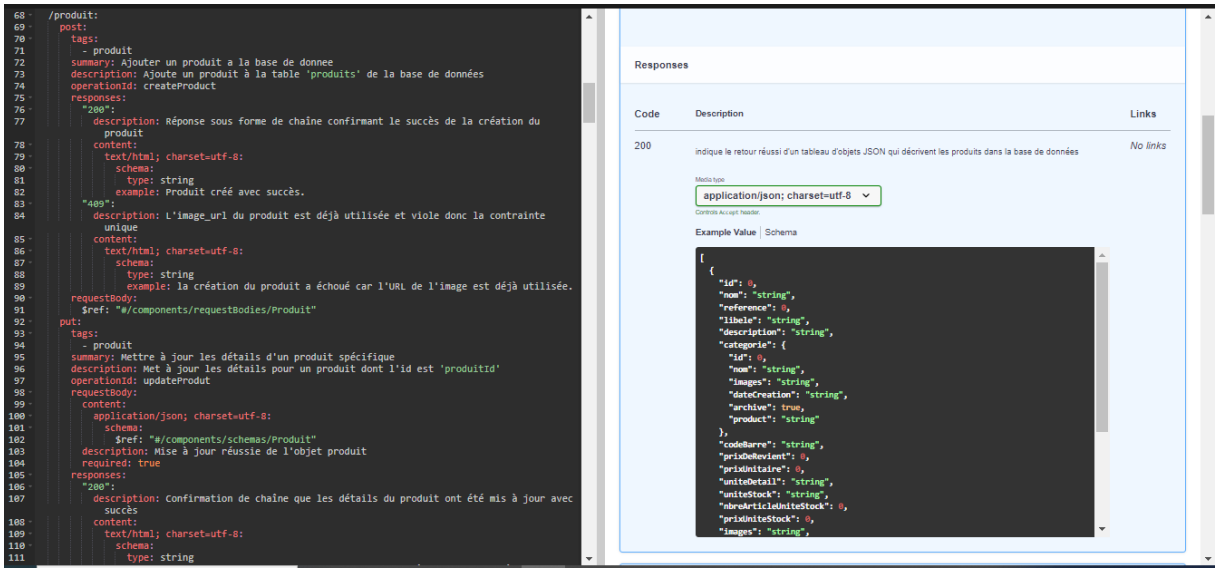


Figure 20: Extrait 3 de la spécification de l'API CATALOGUE

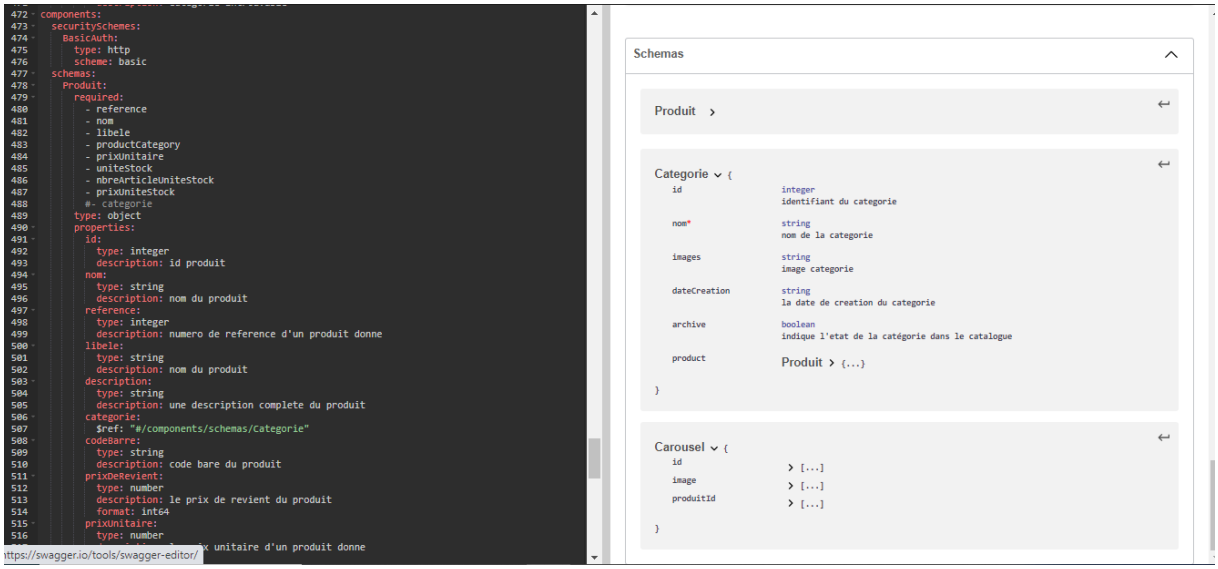


Figure 21: Extrait 4 de la spécification de l'API CATALOGUE

Nous avons défini dans la spécification de l'API Catalogue toutes les fonctionnalités nécessaires pour répondre aux besoins de l'utilisateur. Les avantages d'une telle spécification comprennent notamment une documentation claire et précise de l'API pour les développeurs, la possibilité de générer automatiquement une documentation interactive ainsi que du code client et serveur, et une communication facilitée entre les équipes de développement et les parties prenantes du projet. Afin de répondre aux points précis du cahier des charges et de bien comprendre les fonctionnalités de l'API, nous identifions les besoins des acteurs et produisons les diagrammes de cas d'utilisation.

1. Identification des acteurs

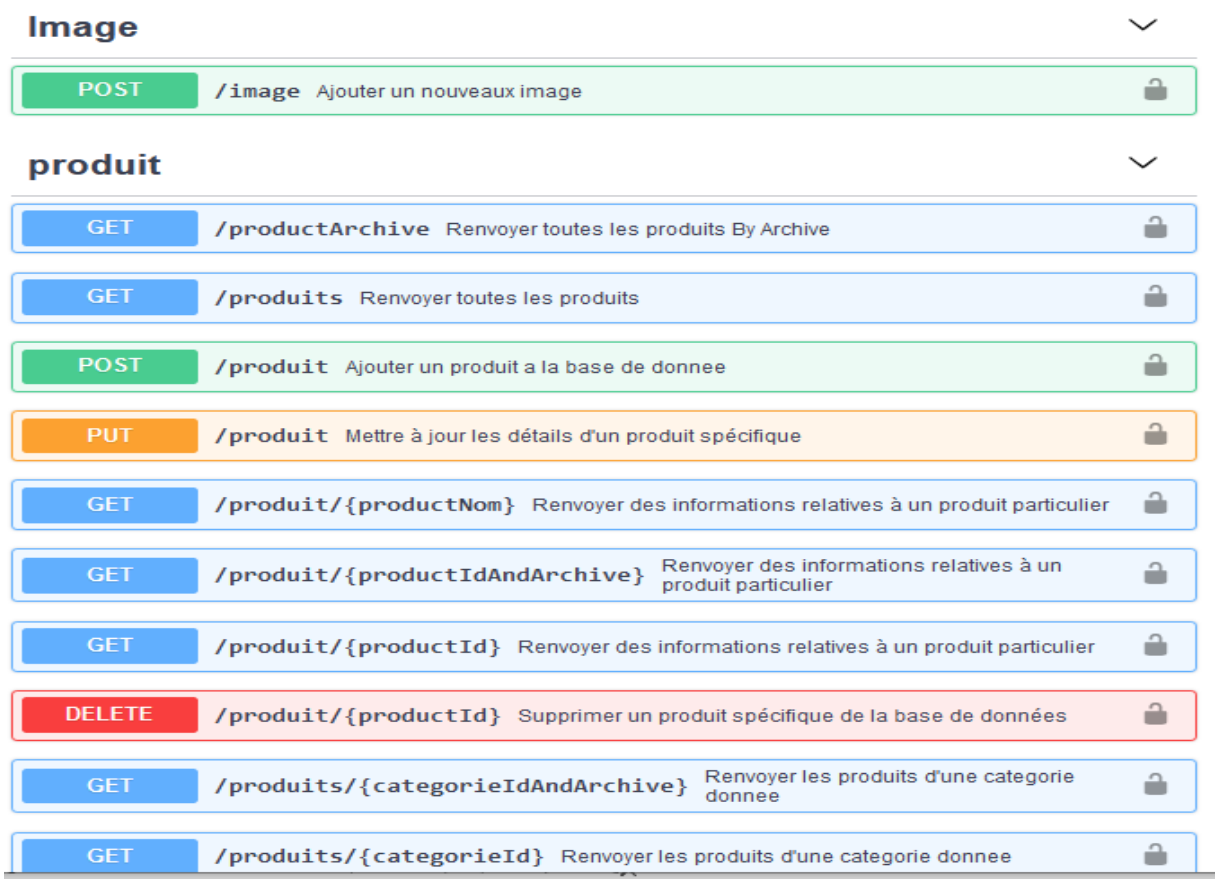
Un acteur peut être considère comme une entité externe qui interagit avec le système en effectuant des actions sur celui-ci. Le **tableau** 4 ci-dessous présente tous les acteurs impliqués dans notre API.

Tableau 4: les acteurs de l'api catalogue

Acteurs	Rôles
Responsable des services généraux	Il gère les produits et les catégories
Chef de département	Consulter
Personnel	Consulter

2. Identification des fonctionnalités

La **figure** 22 ci-dessous présente toutes les méthodes que notre API est capable de fournir à l'utilisateur.



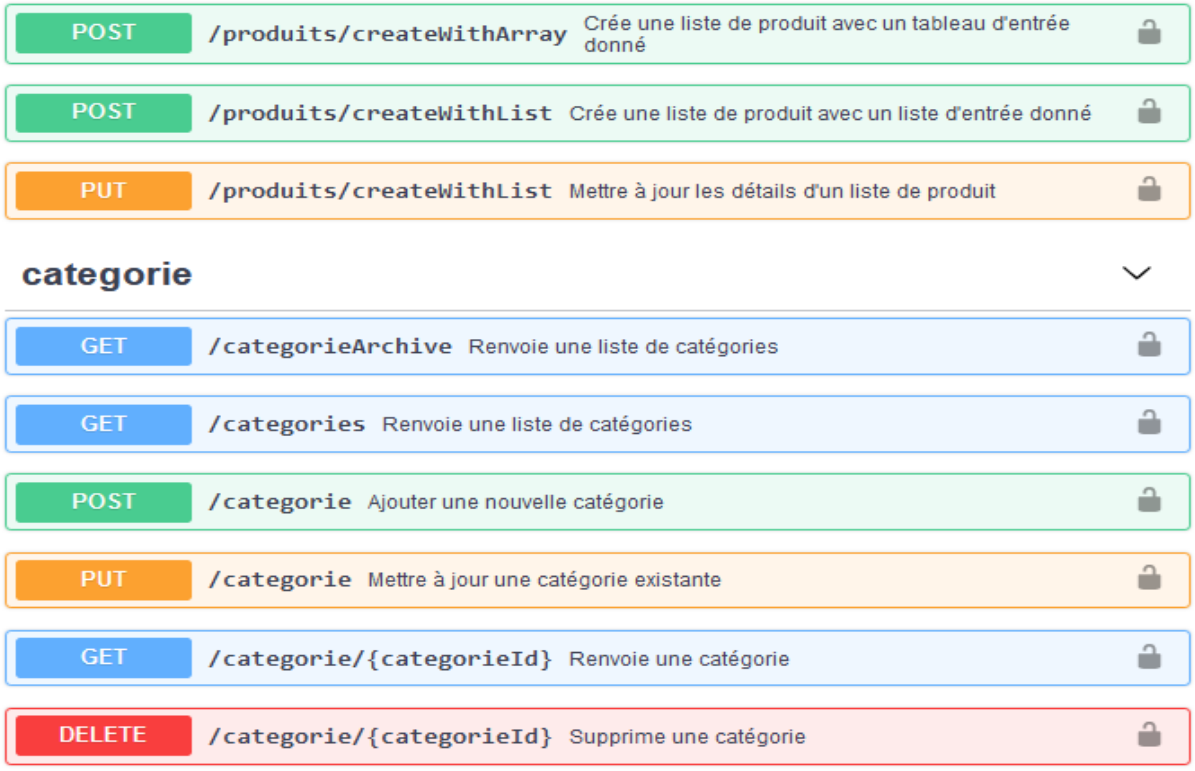


Figure 22: les fonctionnalités de l'API Catalogue

La figure ci-dessus présente les méthodes disponibles pour interagir avec l'API catalogue. Cette visualisation est possible grâce a la documentation Swagger qui facilite la compréhension et la planification de l'intégration de l'API dans un projet donne.

3. Diagramme de cas d'utilisateur

Les diagrammes de cas d'utilisation sont des outils utilises pour modéliser les besoins d'un système. Ils permettent de représenter de manière graphique les interactions fonctionnelles entre les acteurs et le système, d'organiser et d'identifier les grandes fonctionnalités du système, et de formaliser les besoins de manières claire pour toutes les personnes impliquées dans le projet.

3.1Diagramme de cas d'utilisateur du responsable des services généraux

Le diagramme de cas d'utilisation ci-dessous représente les interactions entre le responsable des services généraux et le système pour les fonctionnalités administratives. Le responsable des services généraux à accès à ces fonctionnalités pour gérer les produits, les catégories, les commandes confirmer par le chef de département et gérer les paniers non finalisés. Le diagramme permet de visualiser les responsabilités du responsable des services généraux dans

le système et les actions qu'il peut effectuer pour garantir un bon fonctionnement. La **figure 23** illustre le diagramme de cas d'utilisation du responsable des services généraux.

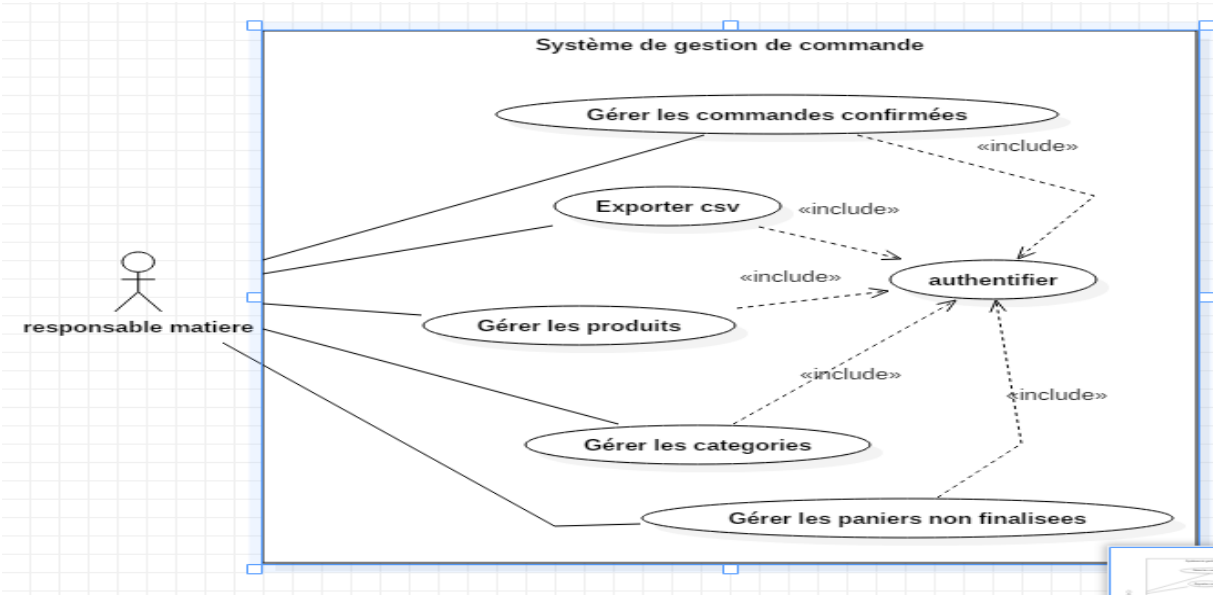


Figure 23: Diagramme de cas d'utilisation du responsable des services généraux

Le Diagramme ci-dessus indique que, une fois authentifié, le responsable des services généraux aura la responsabilité de gérer divers aspects du système. Il pourra gère les commandes confirmées par le chef de départements, les paniers non finalisés ainsi que les catégories et les produits.

3.2Description du cas d'utilisation « gérer produit »

Dans le tableau suivant, nous allons décrire le cas d'utilisation « Gérer produit » présenté dans le diagramme. Le **tableau 5** ci-dessous illustre la description.

Tableau 5: Description du cas d'utilisation « gérer produit »

Etape	Action	Rôle
Précondition	S'authentifier	Responsable des services généraux
Post condition	Les mises à jour effectuées sur un produit doivent être vu dans la base de données, le site et l'interface responsable des services généraux.	

Démarrage	Le menu responsable des services généraux est lancé avec les différentes options	
Scenario nominal		
Cas d'utilisation « Supprimer un produit »	1) Le système affiche la page liste des produits. 2) Le responsable des services généraux peut supprimer un ou plusieurs produits. 3) Le système supprime le ou les produits désirés.	
Cas d'utilisation « Modifier un produit »	1) Le système affiche la page liste des produits. 2) L'administrateur peut modifier une ou plusieurs données d'un ou plusieurs produits 3) Le système modifie le ou les produits désirés.	
Cas d'utilisation « Chercher un produit »	1) Le système affiche la page liste des produits. 2) L'administrateur écrit le nom du produit qu'il cherche 3) Le système affiche le produit cherché. **Scenario relatif** : 1) Si le produit cherché n'existe pas, le système affiche la page liste des produits vide.	
Cas d'utilisation « Ajouter produit »	1) Le système affiche la page ajouter un produit 2) L'administrateur ajoute le produit en introduisant les données du produit	

	3) L'administrateur confirme l'ajout 4) Le système ajoute le produit sur la base de données. **Scenario relatif** : 1) Si le produit ajoute existe déjà, le système affiche un message « produit déjà existant »	

Ce tableau décrit les différentes étapes et actions que le responsable des services généraux peut effectuer pour gérer les produits dans le système. Il peut accéder à la fonctionnalité « gérer produit », visualiser la liste des produits, ajouter de nouveaux produits, les modifier ou les supprimer selon les besoins.

4. Analyse des besoins fonctionnels du système

L'analyse des besoins fonctionnels est une étape cruciale dans le processus de développement. Elle permet d'examiner les différents comportements du système en réponse à une action demandée. Au cours de cette phase, nous effectuons l'analyse de plusieurs cas d'utilisation en présentant leurs diagrammes de séquence afin de comprendre comment les différents éléments du système interagissent entre eux et avec les acteurs.

4.1. Etude des activités « ajouter produit »

L'activité "Ajouter produit" fait partie du cas d'utilisation de la gestion des produits. Cette activité est conçue pour le responsable des services généraux et a pour objectif d'ajouter de nouveaux produits au système.

Lors de l'ajout d'un nouveau produit, le responsable des services généraux peut saisir les informations nécessaires telles que le nom, la description, le prix, la catégorie, etc. Il peut également télécharger des images pour les associer au produit.

L'ajout de produits permet de maintenir à jour la base de données du système, ce qui est important pour une bonne gestion des produits et une expérience utilisateur satisfaisante

4.2. Diagramme de séquence du cas « ajouter produit »

Apres l'authentification du responsable des services généraux, il peut accéder à plusieurs fonctionnalités. Lorsqu'il sélectionne l'option « Ajouter produit », le système affiche la page

correspondante, le responsable des services généraux remplit les informations sur le produit et le système vérifie ces données. Si elles sont complètes et que le produit n'existe pas déjà dans la base de données, il est inséré, sinon le système envoie un message d'erreur explicatif. La **figure 24** montre le diagramme de séquence du cas utilisateur « ajouter produit »

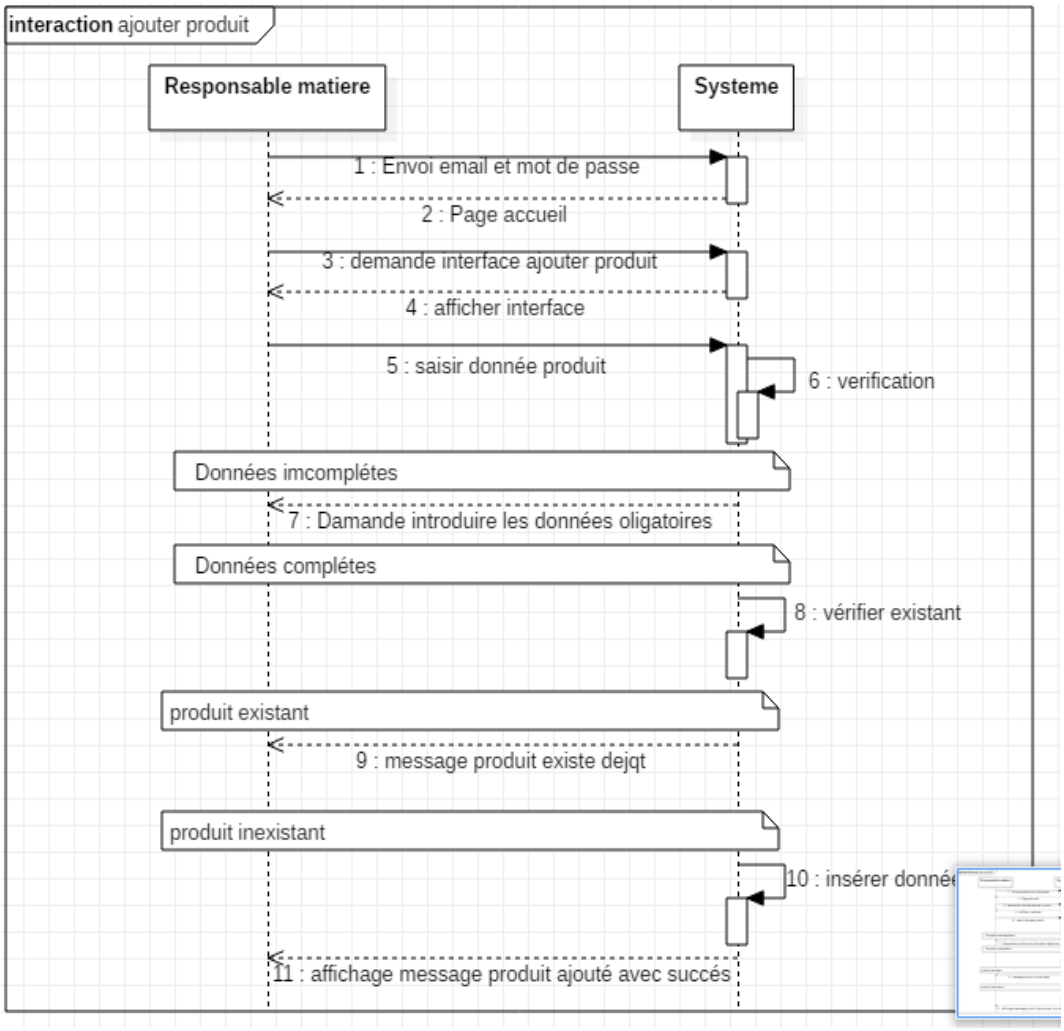


Figure 24 : Diagramme de séquence « ajouter produit »

II. CONCEPTION

Nous avons utilisé OpenAPI et Swagger pour concevoir l'API catalogue.

OpenAPI est une spécification qui permet de décrire les API REST de manière standardisée, tandis que Swagger est un outil open source qui facilite la création, la documentation et la maintenance des API conformes à OpenAPI.

En utilisant OpenAPI et Swagger, nous avons pu élaborer une API catalogue qui est cohérente, bien documentée et facilement compréhensible. Pour plus de détail sur la conception, voir le chapitre précédent

1. Diagramme de classe

Les diagrammes de classes constituent l'un des types les plus utiles de diagramme UML. Ils permettent de décrire la structure d'un système en modélisant les classes, les attributs, les opérations et les relations entre les objets.

Les diagrammes de classes sont largement utilisés dans la phase de conception d'un projet logiciel. Ils fournissent une représentation visuelle claire de la structure du système et aident les développeurs à comprendre les différentes composantes du système.

1.1 Diagramme de classe « service produit »

Le diagramme de classe est un type de diagramme UML (Unified Modeling Language) qui représente les classes d'un système et leurs relations. Le diagramme de classe ci-dessous montre les différentes classes qui participent à la réalisation complète du service catalogue.

Les trois classes présentées sont.

- Produit : cette classe représente un produit dans le catalogue de service. Elle peut contenir des informations sur les détails du produit, telles que la description, le prix, les caractéristiques, etc.
- Catégorie : cette classe représente une catégorie dans le catalogue de service. Elle peut contenir des informations sur les différentes catégories de produits, telles que les électroniques, les cartouches, etc.
- Carrousel : cette classe représente un carrousel qui peut être utilisé pour ajouter plusieurs images à un produit. Elle peut contenir des informations sur les différentes images associées à un produit, telles que les images de produit.

Les relations entre ces classes peuvent être décrites de la manière suivante.

- Un produit peut être dans une seule catégorie.
- Une catégorie peut avoir plusieurs produits.
- Un carrousel peut avoir plusieurs images.

La **figure 25** ci-dessous montre les différentes classes qui composent le service produit.

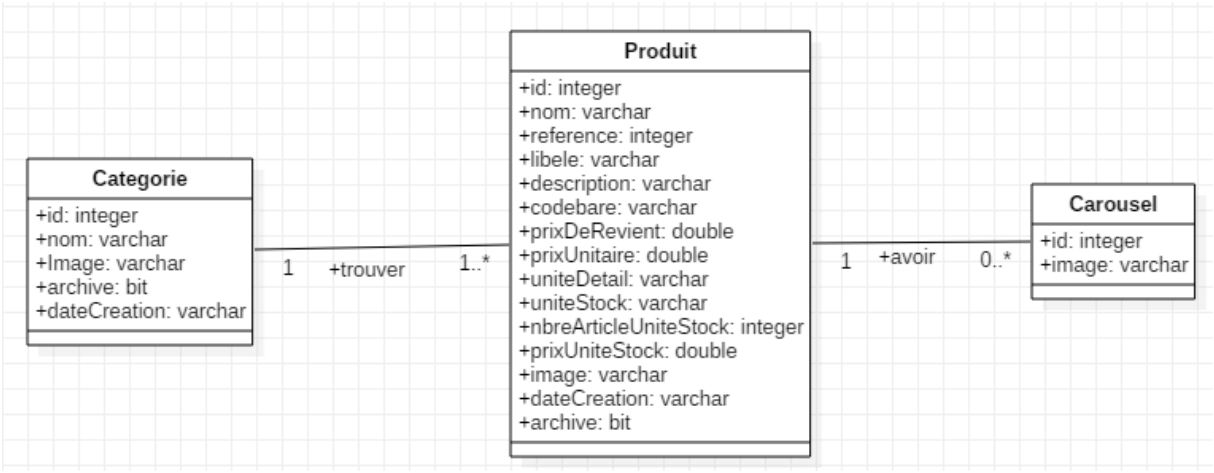


Figure 25 : Schéma de donnée api catalogue

2. Schéma de donnée

Nous avons utilisé OpenAPI de Swagger pour définir le schéma de données de l'API catalogue. En utilisant Open API et Swagger, nous avons pu décrire clairement les données disponibles via montre API catalogue et leur structure, ainsi que les opérations possibles sur ces données. La documentation Swagger offre une représentation visuelle pratique du schéma de données, ce qui facilite la compréhension et la maintenance de l'API.

La **figure 26** ci-dessous représente la partie « Schéma de données » de la documentation de l'API catalogue.

Schemas

Produit {

id	integer	
	id produit	
nom*	string	
	nom du produit	
reference*	integer	
	numero de reference d'un produit donne	
libele*	string	
	nom du produit	
description	string	
	une description complete du produit	
categorie	**Categorie** > {...}	
codeBarre	string	
	code bare du produit	
prixDeRevient	number($int64)	
	le prix de revient du produit	
prixUnitaire*	number($int64)	
	le prix unitaire d'un produit donne	
uniteDetail	string($int64)	
	unite de detail	
uniteStock*	string($int64)	
	unite stock	
nbreArticleUniteStock*	number($int64)	
	nombre d'article dans l'unite de stock	
	nombre d'article dans l'unite de stock	
prixUniteStock*	number($int64)	
	prix d'unite de chaque produit dans le stock	
images	string	
	image produit	
dateCreation	string	
	la date de creation du produit	
archive	boolean	
	indique l'etat du produit dans le catalogue	
carousel	> [...]	
categories	string	

}

Categorie {

id	integer	
	identifiant du categorie	
nom*	string	
	nom de la categorie	
images	string	
	image categorie	
dateCreation	string	
	la date de creation du categorie	
archive	boolean	
	indique l'etat de la catégorie dans le catalogue	
product	> {...}	

}

```
Carousel ⌄ {
    id                  integer
                        identifiant du carousel
    image               string
                        nom de l'image
    produitId           integer
                        identifiant du produit
}
```

Figure 26: Schéma de donnée api catalogue

Conclusion

L'utilisation de la méthode OpenAPI et Swagger pour la spécification et la conception de l'API a permis de générer facilement la documentation et le code technique, tout en garantissant la cohérence et la compatibilité de l'API grâce a l'approche contract-first. Cette méthode a non seulement facilité la communication entre les différentes équipes de développement, mais a également réduit les erreurs et les incohérences pour améliorer la qualité de l'interface utilisateur. Les étapes rigoureuses de spécification et de conception de l'API catalogue ont été cruciales pour assurer que l'API réponde aux besoins et aux attentes des utilisateurs, offrant une expérience fiable et conforme aux spécifications.

Les mêmes principales seront appliques pour les chapitres suivants tels que l'API commande et l'API statistique afin de garantir une qualité optimale de l'interface utilisateur pour l'ensemble du projet.

CHAPITRE V : SPECIFICATION ET CONCEPTION DE L'API COMMANDE

Introduction

L'analyse des besoins fonctionnels permet d'identifier les différents acteurs du système ainsi que les fonctionnalités associées à chacun pour atteindre un résultat optimal et satisfaisant pour l'utilisateur. Dans ce chapitre, nous commençons par élaborer une spécification des exigences fonctionnelles de l'API utilisateur en utilisant les outils Swagger et OpenAPI. Ensuite, nous concevons l'API en utilisant cette spécification comme point de départ.

I. SPECIFICATION

La spécification est essentielle pour le développement de l'API car elle établit les fonctionnalités fondamentales qui doivent être incluses pour satisfaire les besoins des utilisateurs. En travaillant avec les parties prenantes, la spécification facilite la clarification des exigences fonctionnelles avant de passer à la phase de conception de l'API.

Les **figures 27,28,29 et 30** ci-dessous représentent une partie de la spécification de l'API Commande.

Figure 27 : Extrait 1 de la spécification de l'API COMMANDE

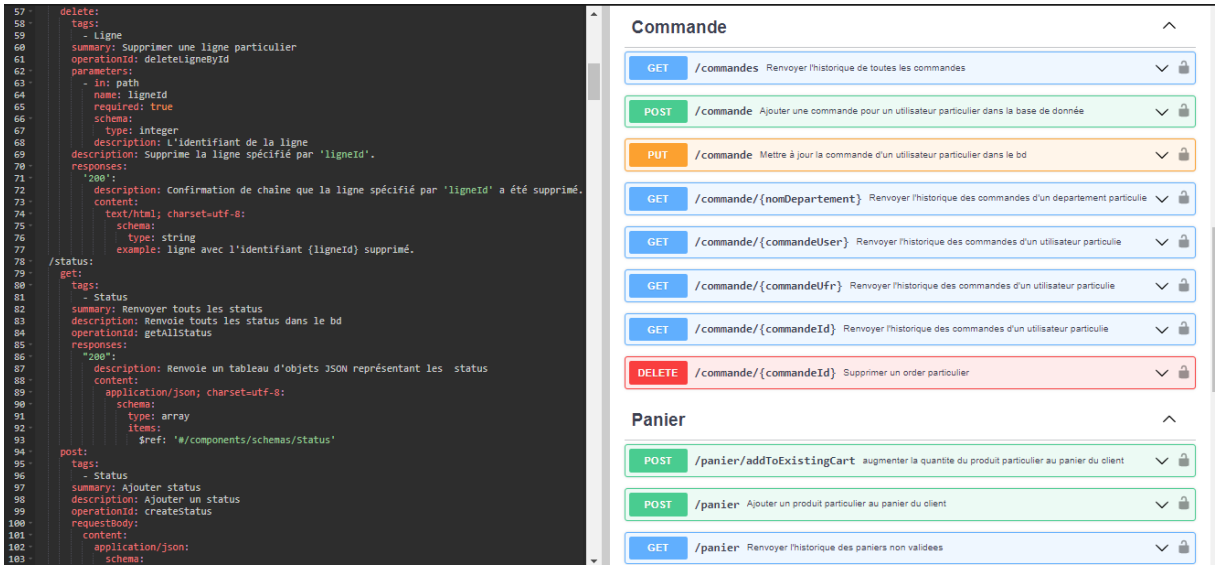


Figure 28: Extrait 2 de la spécification de l'API COMMANDE

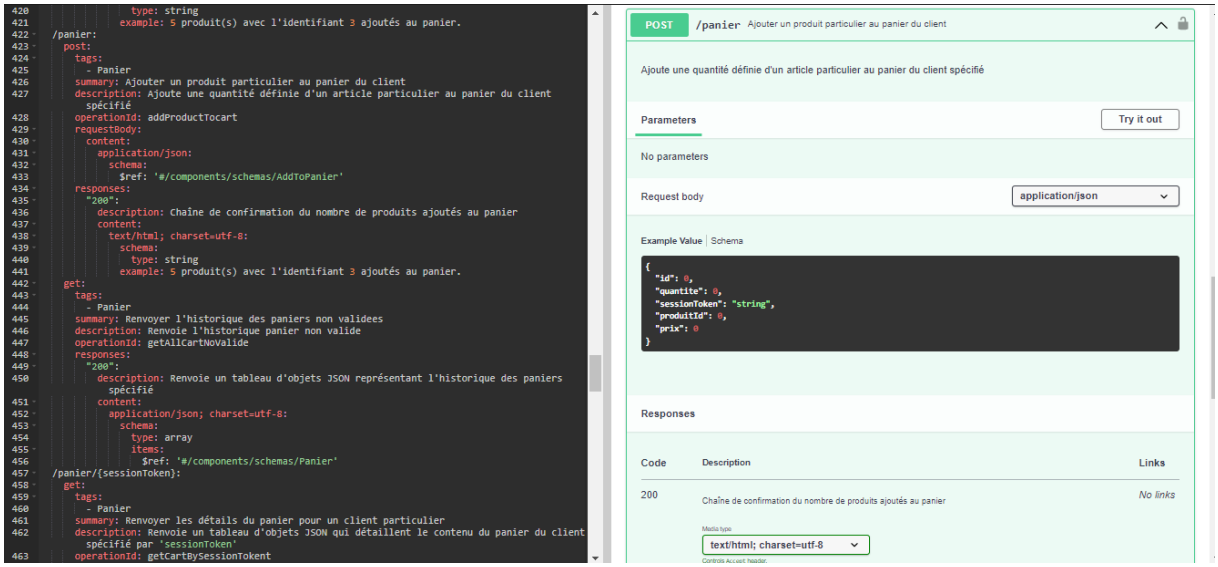


Figure 29: Extrait 3 de la spécification de l'API COMMANDE

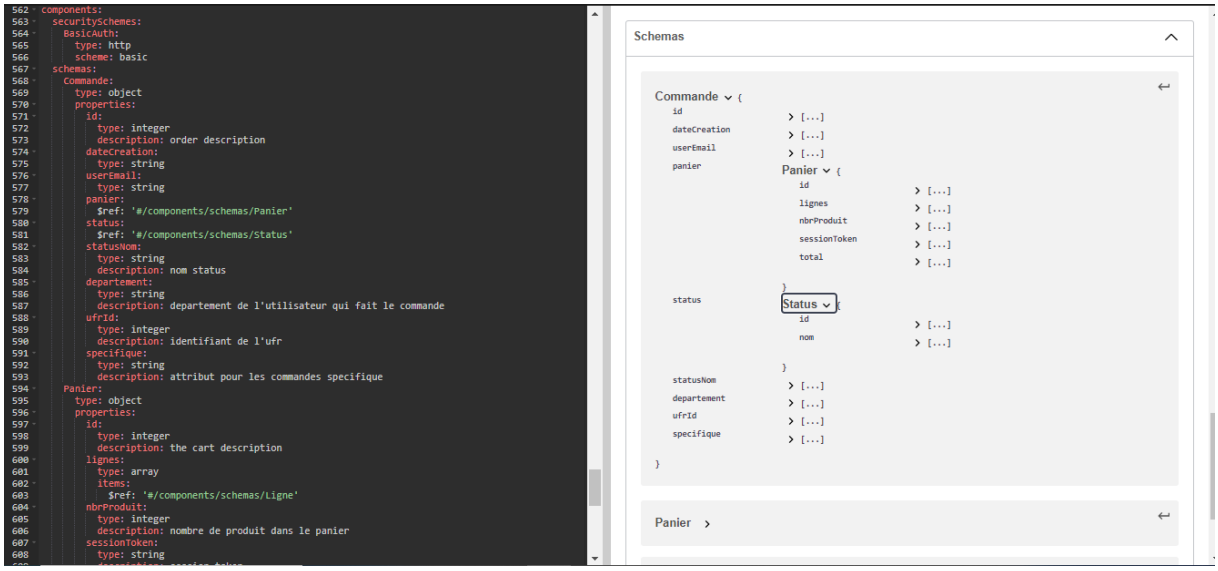


Figure 30: Extrait 4 de la spécification de l'API COMMANDE

La spécification de l'API commande est écrite en format YAML pour décrire une API RESTful. Elle spécifie les routes de l'API, les opérations HTTP et les paramètres de requête pour chacune des routes. Les schémas de données utilisés dans l'API, tels que Commande, Panier et Ligne, sont également définis. Cette spéciation fournit une documentation claire et précise, permettant une communication plus facile entre les équipes de développement et les parties prenantes du projet. Elle facilite également la maintenance de l'API, permettant aux modifications apportées à la spécification d'être utilisées pour régénérer la documentation interactive et le code client et serveur. Les besoins des acteurs ont été identifies et des diagrammes de cas d'utilisation seront produits pour comprendre les fonctionnalités de l'API.

Veuillez consulter le chapitre III de l'API Utilisateur pour plus de détails sur la spécification.

1. Identification des acteurs

Les acteurs sont des entités externes qui interagissent avec le système à travers des actions. Deux types d'acteurs sont considères : les acteurs principaux, qui sont ceux qui bénéficient du service rendu par le système, et les acteurs secondaires, qui sont sollicités pour des informations supplémentaires. **Le tableau 6** ci-dessous présente tous les acteurs impliqués dans l'API Commande.

Tableau 6: les acteurs de « l'API Commande »

Acteurs	Rôles

Responsable des services généraux	Il gère les commandes confirmées et les paniers non validées
Chef de département	Il gère l'ensemble des commandes de son département. Il a aussi la possibilité de passer des commandes
Enseignant, Secrétaire	Ils peuvent passer des commandes et les suivre jusqu'à la livraison.

2. Identification des fonctionnalités

Les fonctionnalités du système correspondent aux actions ou services qu'il peut fournir en réponse aux sollicitations des acteurs concernés. Ces services peuvent être soit externes, nécessitant l'intervention d'un acteur, soit internes, c'est-à-dire invisibles mais fonctionnant en interne lors du déclenchement d'un processus donné.

Lors de la création d'une API, il est essentiel de veiller à ce que les données échangées soient correctement structurées et organisées. Les développeurs peuvent utiliser la documentation Swagger, un outil de spécification d'API permettant de décrire les opérations disponibles via cette API. **La figure 31** ci-dessous illustre l'ensemble des méthodes que notre API peut proposer aux utilisateurs.

Commande

GET	/commandes	Renvoyer l'historique de toutes les commandes
POST	/commande	Ajouter une commande pour un utilisateur particulier dans la base de donnée
PUT	/commande	Mettre à jour la commande d'un utilisateur particulier dans le bd
GET	/commande/{nomDepartement}	Renvoyer l'historique des commandes d'un departement particulie
GET	/commande/{commandeUser}	Renvoyer l'historique des commandes d'un utilisateur particulie
GET	/commande/{commandeUfr}	Renvoyer l'historique des commandes d'un utilisateur particulie
GET	/commande/{commandeId}	Renvoyer l'historique des commandes d'un utilisateur particulie
DELETE	/commande/{commandeId}	Supprimer un order particulier

Panier

POST	/panier/addToExistingCart	augmenter la quantite du produit particulier au panier du client
POST	/panier	Ajouter un produit particulier au panier du client
GET	/panier	Renvoyer l'historique des paniers non validees
GET	/panier/{sessionToken}	Renvoyer les détails du panier pour un client particulier
DELETE	/panier/{sessionToken}	Supprimer un panier particulier du panier d'identification spécifié
PUT	/panier/{sessionToken}	Mettre à jour panier particulier dans le bd
DELETE	/panier/{produitId}/{sessionToken}	Supprimer un produit particulier du panier du client spécifié

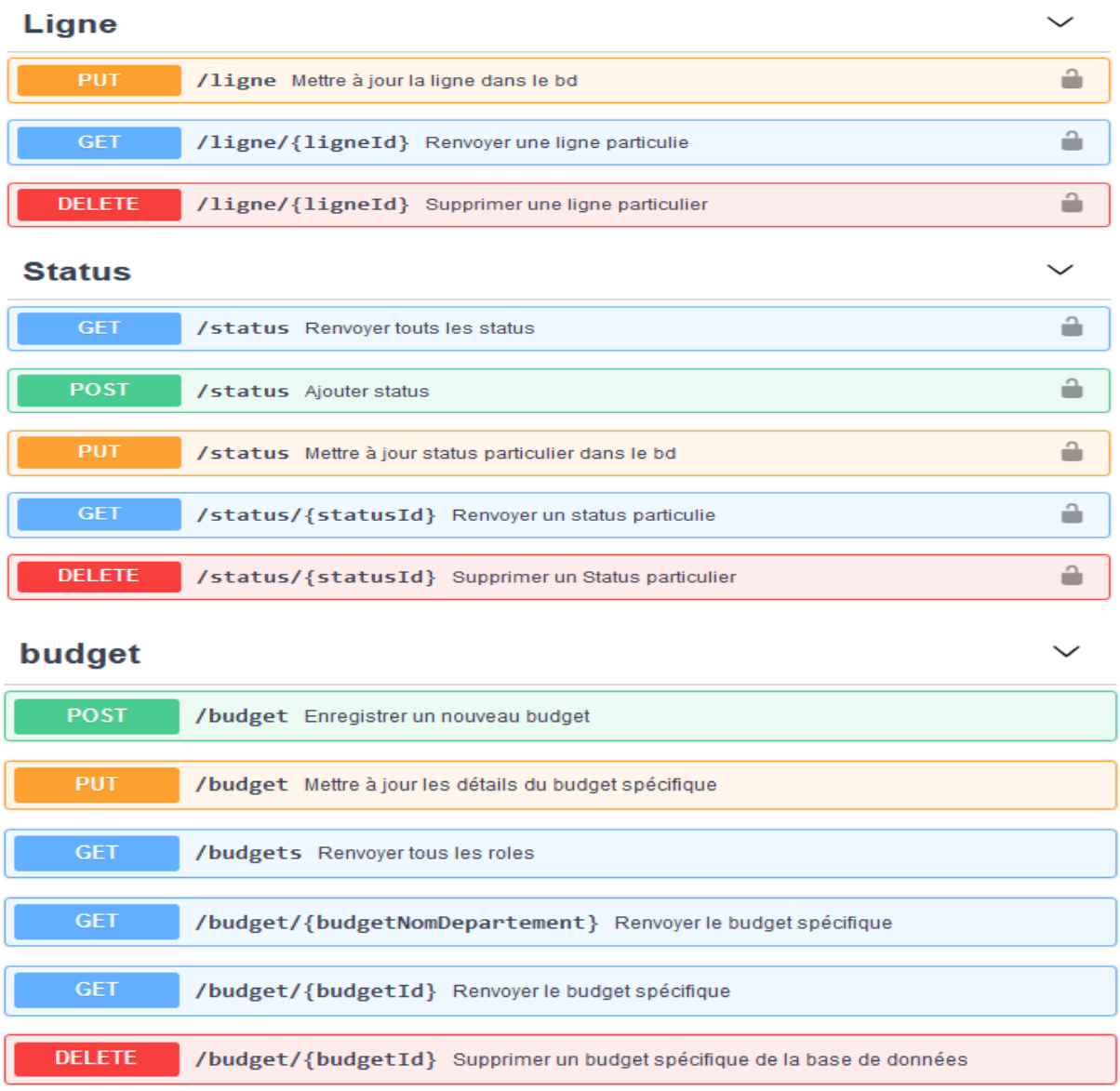


Figure 31: les fonctionnalités de L'API Commande

3. Diagramme de cas d'utilisateur

Les diagrammes de cas d'utilisation sont des outils utilisés pour modéliser les besoins d'un système. Ils permettent de représenter les interactions fonctionnelles entre les acteurs et le système, d'organiser et d'identifier les principales fonctionnalités du système, et de formaliser les besoins de manière graphique et accessible à toutes les parties prenantes du projet.

Ils offrent une vue d'ensemble claire et simplifiée des interactions entre les acteurs et le système.

3.1 Diagramme de cas d'utilisateur du chef de département

Le diagramme de cas d'utilisation ci-dessous (**figure** 32) représente les interactions entre le chef de département et le système pour les fonctionnalités administratives. Le chef de département a accès à ces fonctionnalités pour gérer les commandes, et le budget. Le diagramme permet de visualiser les responsabilités du chef de département dans le système et les actions qu'il peut effectuer pour garantir un bon fonctionnement.

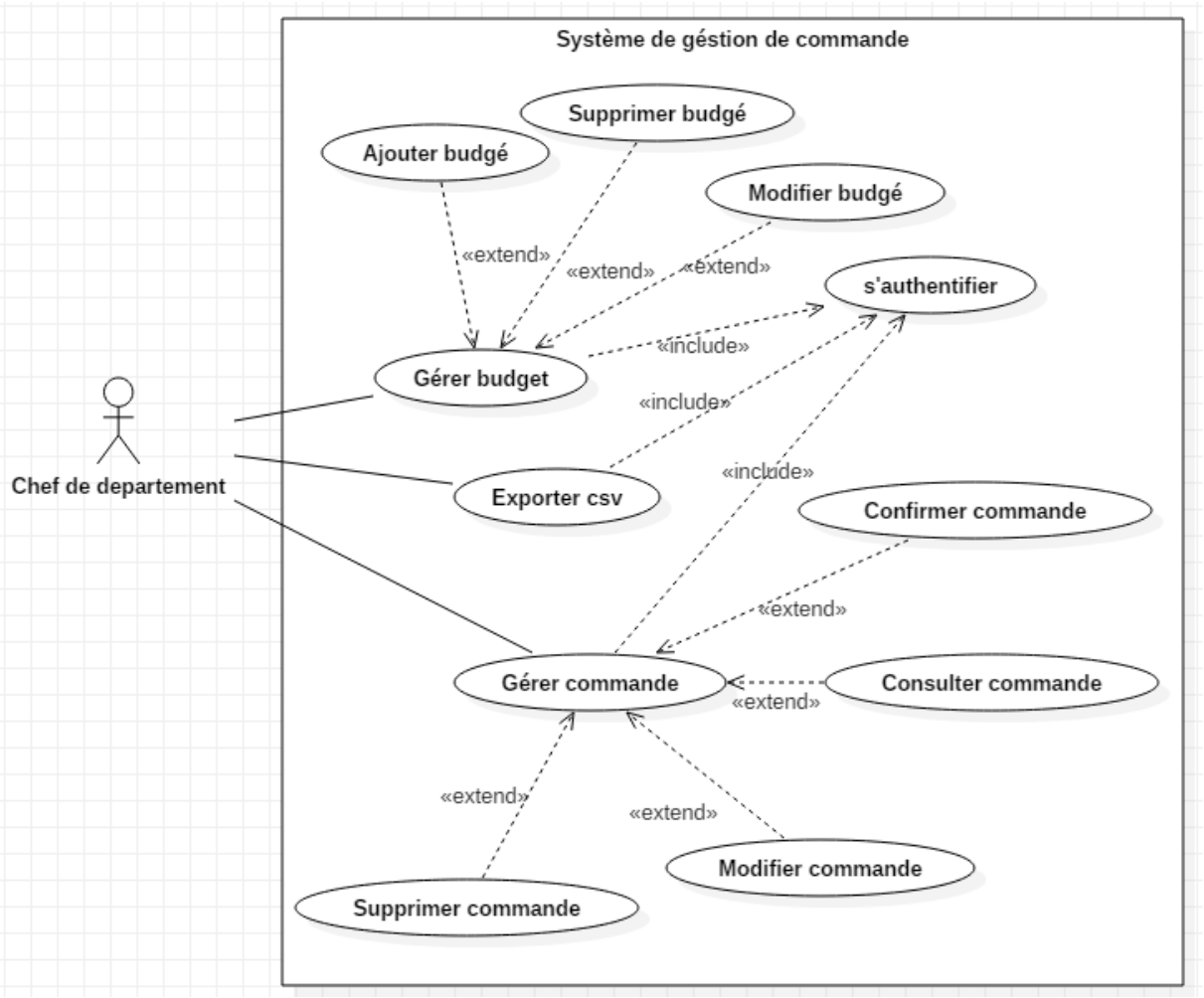


Figure 32: Diagramme de cas d'utilisation du chef de département.

Le Diagramme ci-dessus indique que, une fois authentifié, le chef de département aura la responsabilité de gérer divers aspects du système. Il pourra gère les commandes, le budget ainsi importer les commandes sous le format csv.

3.2 Description des cas d'utilisation

Cette partie a pour but de décrire certains cas d'utilisation de notre diagramme en utilisant un tableau. Les différents cas d'utilisation seront décrits de manière claire et concise pour en

faciliter la compréhension. Les informations présentées dans le tableau incluront les actions à effectuer.

a. Description du cas d'utilisation « gérer commande »

Le cas d'utilisation « Gérer commande par le chef de département » se concentre sur la gestion des commandes pour un département spécifique. Le chef de département est responsable de la gestion des commandes pour son département et peut effectuer les actions suivantes.

- Consulter les commandes : le chef de département peut consulter la liste des commandes du département.
- Confirmer les commandes : le chef de département peut confirmer ou refuser une commande en fonction des besoins.
- Mettre à jour les informations de la commande : le chef de département peut mettre à jour les informations associées à une commande.

Le résultat attendu de ce cas d'utilisation est de permettre au chef de département de gérer les commandes pour son département de manière efficace, en offrant un moyen simple et rapide de consulter, mettre à jour et confirmer les commandes. Cela garantit que les commandes sont gérées en temps voulu pour répondre aux besoins.

a. Description du cas d'utilisation « passer commande »

Le **tableau 7** ci-dessous illustre la description

Tableau 7 : Description du cas d'utilisation « « passer commande »

Nom	Passer commande
Résume	Ce cas d'utilisation permet à tout acteur concerné de passer une ou des commandes en ligne
Acteur	• Enseignant • Secrétaire • Chef de département
Précondition	• Il faudra s'authentifier • Le panier doit avoir au moins un produit.
Postcondition	• Les commandes vont être enregistrées dans la base de données et par la suite vues par le chef de département.

Scenario normal	• Le système affiche la page liste des produits contenant les images, les prix, les noms et le choix de la quantité. • Il choisit un produit en cliquant sur le bouton ajouter au panier. • Le système affiche la page liste des produits du panier après avoir cliqué sur le panier. • Il pourra modifier la quantité. • Confirmer la commande

Le cas d'utilisation "Passer commande" se concentre sur la procédure pour passer une commande pour un produit ou un ensemble de produits. Les utilisateurs peuvent effectuer les actions suivantes.

- Sélectionner les produits : les utilisateurs peuvent sélectionner les produits qu'ils souhaitent commander en naviguant dans la liste de produits disponibles.
- Ajouter des produits au panier : les utilisateurs peuvent ajouter des produits à leur panier en spécifiant la quantité souhaitée pour chaque produit.
- Vérifier le contenu du panier : les utilisateurs peuvent vérifier le contenu de leur panier et apporter des modifications si nécessaire.
- Passé la commande : les utilisateurs peuvent passer la commande.

Le résultat attendu de ce cas d'utilisation est de permettre aux utilisateurs de commander des produits de manière simple et rapide. Cela garantit que les commandes sont passées de manière efficace et en temps voulu, ce qui permet de répondre aux besoins des utilisateurs de manière rapide et efficace.

b. Description du cas d'utilisation « voire liste des paniers non finalisés »

Le **tableau 8** ci-dessous illustre la description.

Tableau 8: Description du cas d'utilisation « voire liste des paniers non finalisés »

Non	Panier non validée
Acteur	Responsable des services généraux
Description	Lister l'ensemble des paniers non validées par l'utilisateur
Préconditions	Le responsable des services généraux doit s'authentifier

Scenario nominal	• Le menu responsable des services généraux est lance avec les différentes options • Il peut ainsi aller vers l'option panier non validée.
Post condition	Aucun

Le cas d'utilisation « Voire liste des paniers non finalisés par les utilisateurs » concerne la gestion des paniers inachevés par les utilisateurs. Le responsable des services généraux est en charge de les supprimer. Les actions suivantes sont implémentées dans ce cas d'utilisation.

- Accès à la liste des paniers : le responsable des services généraux accède à la liste des paniers non finalisés créés par les utilisateurs.
- Affichage des paniers : la liste des paniers non finalisés est affichée pour le responsable des services généraux, permettant une vérification rapide et facile de ces paniers.
- Suppression des paniers : le responsable des services généraux peut choisir de supprimer un ou plusieurs paniers non finalisés en sélectionnant les paniers appropriés à partir de la liste affichée.

Le résultat attendu de ce cas d'utilisation est de garantir que les paniers inachevés ne restent pas dans le système indéfiniment, ce qui peut entraîner une utilisation inefficace des ressources et de l'espace de stockage. La suppression de ces paniers permet également de maintenir un système propre et ordonné, facilitant ainsi la gestion des commandes pour le responsable des services généraux.

c. Description du cas d'utilisation « gérer panier »

Tableau 9: Description du cas d'utilisation « gérer panier »

Nom	Gérer panier
Acteur(s)	Enseignant, Secrétaire, Chef département
Précondition	Le panier doit avoir des produits
Scenario nominal	1. Le système affiche la page liste produits dans le panier. 2. S'il choisit de supprimer un ou plusieurs produits dans le panier. 3. Le système supprime le produit désiré.

	4. S'il choisit de modifier la quantité du produit dans le panier. 5. Le système modifie la quantité du produit désiré

Le cas d'utilisation « Gérer panier » permet à l'utilisateur de gérer son panier.

Les étapes suivantes sont impliquées dans ce cas d'utilisation.

- Connexion de l'utilisateur : l'utilisateur se connecte à son compte sur la plateforme de commande en ligne.
- Affichage du panier : une fois connecté, l'utilisateur peut accéder à son panier de commande en cliquant sur le bouton "Panier" dans le menu.
- Ajout/Suppression de produits : l'utilisateur peut ajouter ou supprimer des produits de son panier en cliquant sur les boutons "+" ou "-" associés aux produits.
- Modification de la quantité : l'utilisateur peut modifier la quantité des produits dans son panier en utilisant les boutons "+" et "-".
- Finalisation de la commande : l'utilisateur peut finaliser sa commande en cliquant sur le bouton "Passer la commande" une fois qu'il est satisfait du contenu de son panier.
- Suivi de la commande : l'utilisateur peut suivre l'état de sa commande en accédant à la section "mes commandes" de son compte.

Ce cas d'utilisation permet à l'utilisateur de gérer son panier de manière intuitive et efficace, ce qui améliore l'expérience de commande pour l'utilisateur.

4. Analyse des besoins fonctionnels du système

L'analyse des besoins fonctionnels joue un rôle essentiel dans le processus de développement. Elle vise à comprendre les différents comportements du système en réponse à une action spécifique demandée. Dans cette partie, nous analyserons plusieurs cas d'utilisation en présentant leurs diagrammes de séquence. Ces diagrammes nous permettront de visualiser les interactions entre les éléments du système ainsi qu'avec les acteurs impliqués. Ils nous aideront à comprendre comment ces éléments interagissent entre eux pour satisfaire les besoins fonctionnels identifiés.

III. Etude des activités « Confirmer commande »

L'étude de l'activité « Confirmer commande » est destinée uniquement au chef de département. Ce cas d'utilisation est lié à la gestion des commandes pour le département en question et seul le chef de département est responsable de la confirmation des commandes. Le chef de département doit être en mesure de vérifier les détails de la commande et de confirmer si tout est correct pour finaliser la commande.

a. Diagramme de séquence « Confirmer commande »

Le chef de département est connecté et est redirigé vers son menu. Il peut alors choisir l'option « liste des commandes », ce qui affichera la liste des commandes associées à son département. Pour visualiser les détails d'une commande, il peut cliquer sur le bouton « voire détail commande ». Ensuite, il peut choisir de confirmer ou nom la commande en fonction de ce qu'il constate dans les détails. La **Figure 33** ci-dessous illustre le diagramme de séquence « confirmer commande »

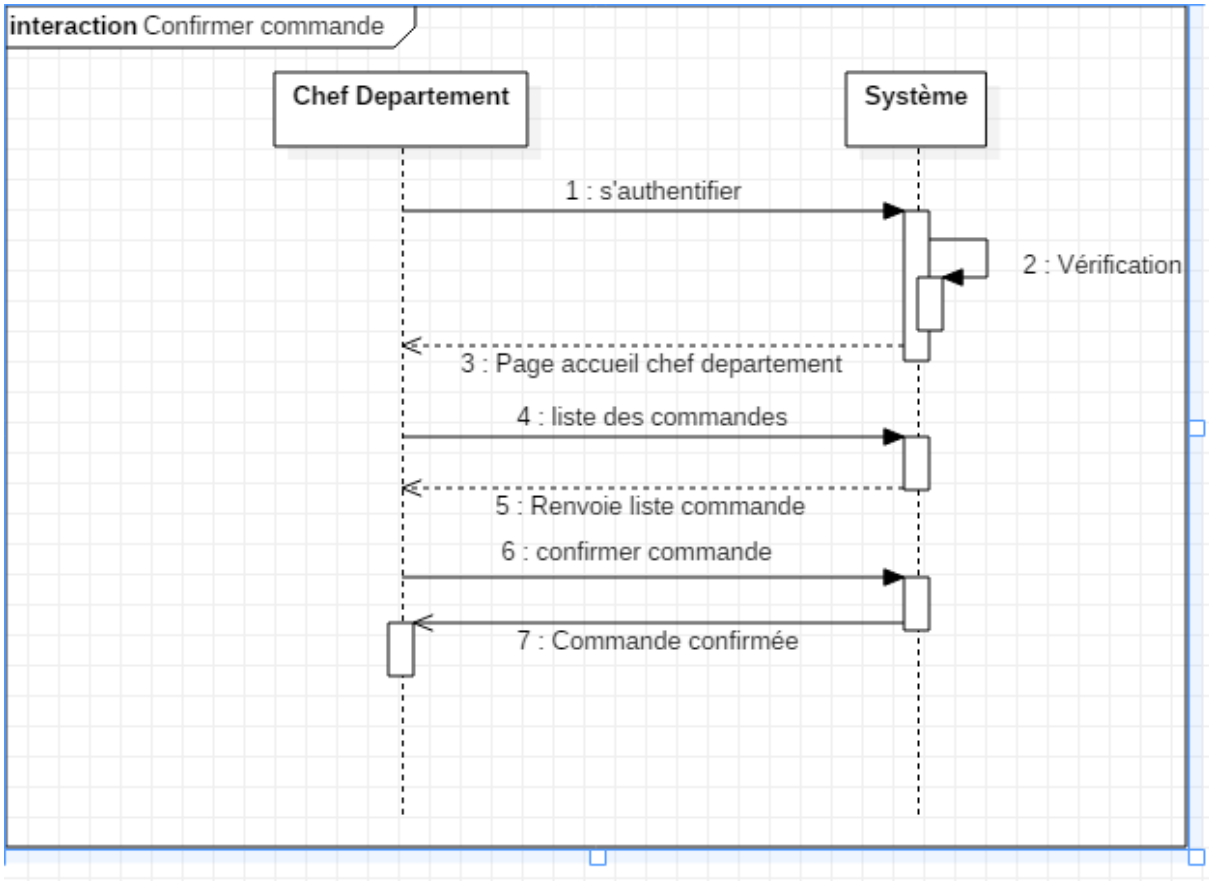


Figure 33 : Diagramme de séquence « confirmer commande »

II. CONCEPTION

La conception de l'API est cruciale dans le développement et l'utilisation d'outils comme OpenAPI et Swagger facilite cette étape. OpenAPI permet de décrire les fonctionnalités de l'API de manière structurée, tandis que Swagger offre une interface conviviale pour la documentation et les tests d'API. Utiliser OpenAPI et Swagger présente plusieurs avantages, notamment une description claire des fonctionnalités, une collaboration facilitée entre les membres de l'équipe de développement et une amélioration de la qualité de l'API grâce à des tests automatisés.

Veuillez consulter le chapitre III de l'API Utilisateur pour plus de détails sur la conception.

1. Diagramme de classe

Les diagrammes de classe sont des outils essentiels en UML car ils permettent de représenter la structure d'un système en modélisant ses classes, leurs attributs, leurs opérations et les relations entre les objets. Ils offrent une vision claire et organisée de la composition et des interactions des éléments clés d'un système, ce qui facilite la compréhension de la conception et de son fonctionnement.

1.1 Diagramme de classe « service commande »

Le diagramme de classe représente les différentes classes impliquées dans la réalisation complète du service de commande. Les classes comprennent.

- Commande : représente une commande effectuée par un utilisateur.
- Produit : représente un produit disponible.
- Panier : représente un panier de commande où les produits peuvent être ajoutés.
- Ligne : représente une ligne de produit dans le panier.
- Utilisateur : représente un utilisateur qui effectue une commande.
- Statut : représente le statut d'une commande, par exemple confirmé.

Une commande peut être associée à un seul statut à la fois, comme par exemple confirmé. Dans une ligne, il peut y avoir un seul produit. Un panier peut contenir plusieurs lignes de produits. Chaque commande appartient à un seul utilisateur. La **figure 34** ci-dessous montre les différentes classes qui compose le service commande.

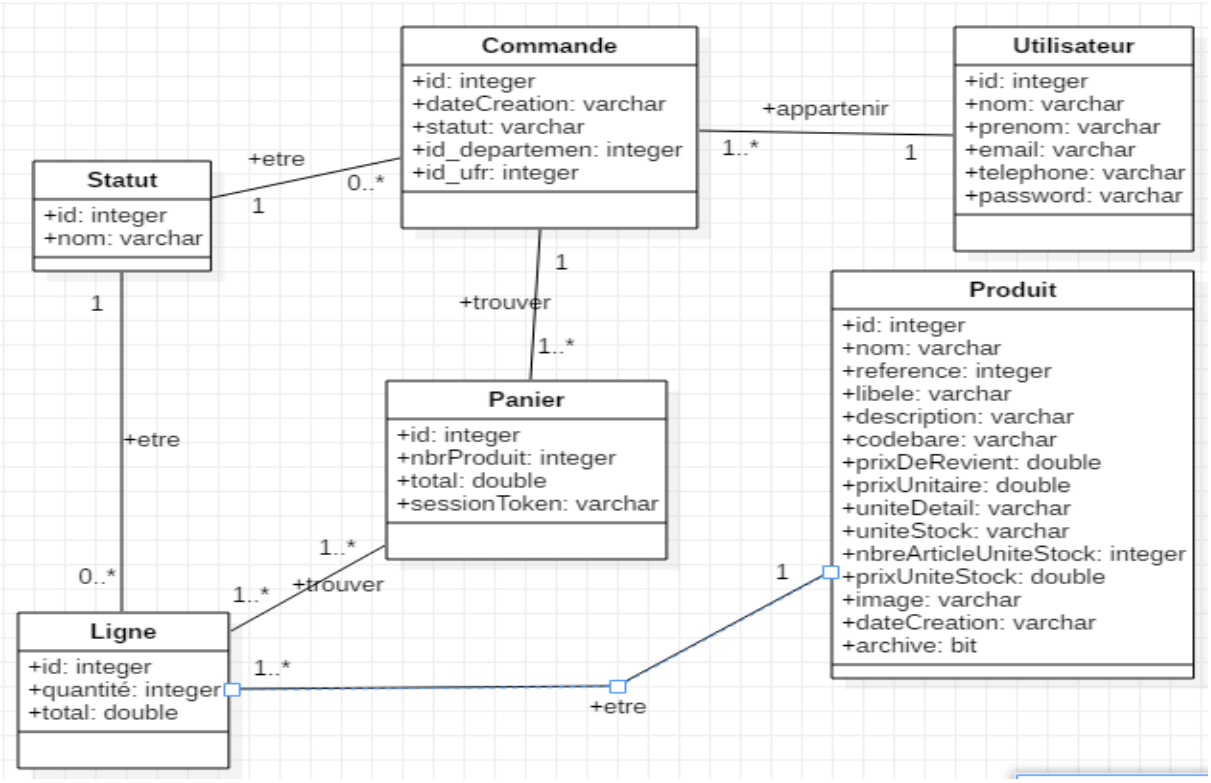


Figure 34: Diagramme de classe « service commande »

2. Schéma de donnée

La documentation Swagger permet de visualiser le schéma de données de l'API catalogue. Le schéma ci-dessous décrit les données disponibles, leur structure et leur format. Cela facilite la compréhension des relations entre les données et aide à détecter les erreurs éventuelles dans leur structure. La documentation rend l'API plus accessible et facile à intégrer dans les applications en fournissant une documentation claire et structurée, permettant d'économiser du

temps et d'éviter les erreurs lors de l'intégration de l'API. **La figure 35** ci-dessous illustre les données utilisées pour développer l'API commande.

Panier ⌄ {

id	integer	
	the cart description	
lignes	› [...]	
nbrProduit	integer	
	nombre de produit dans le panier	
sessionToken	string	
	session token	
total	number($64bit)	
	la somme total de l'ensemble des produits sur le panier	

}

Ligne ⌄ {

id	integer	
	the cart description	
prod	integer	
	id du produit	
quantite	integer	
	quantite du produit	
total	number($64bit)	
	la somme total de l'ensemble des produits sur le panier	
status	**Status** › {...}	
statusNom	string	
	nom status	

}

Status ⌄ {

id	integer	
	identifiant status	
nom	string	
	status du commande	

}

AddToPanier ⌄ {

id	integer	
	identifiant AddToPanier	
quantite	integer	
	quantite	
sessionToken	string	
	sessionToken	
produitId	integer	
	id produit	
prix	number($64bit)	
	prix du produit	

}

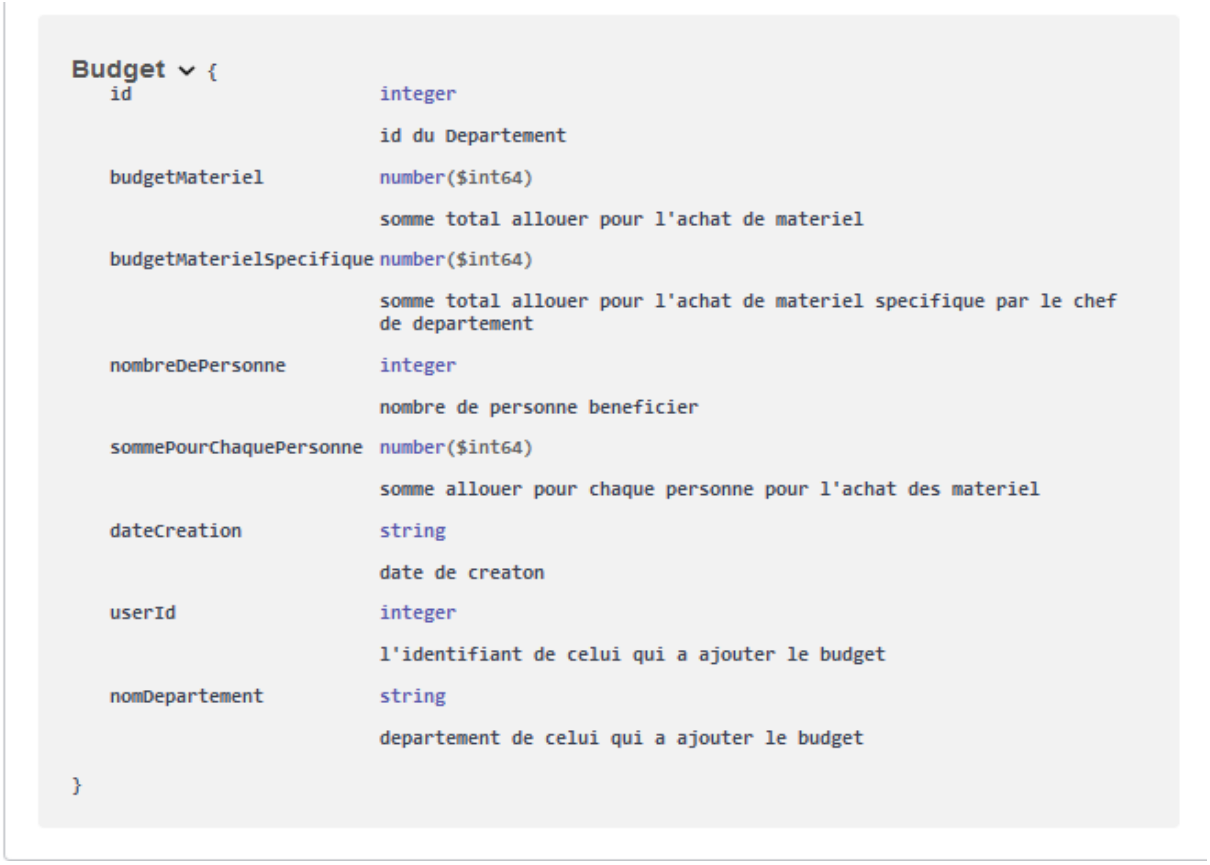


Figure 35: Schéma de donnée api commande

Conclusion

Grâce à l'utilisation de la méthode OpenAPI et Swagger, la spécification et la conception de l'API ont été facilitées, avec une génération rapide de la documentation et du code technique, ainsi qu'une garantie de cohérence et de compatibilité grâce à l'approche contract-first. En conséquence, la communication entre les différentes équipes de développement a été améliorée, tandis que la qualité de l'interface utilisateur a été améliorée grâce à une réduction des erreurs et des incohérences. Les étapes rigoureuses de spécification et de conception de l'API commande ont été cruciales pour s'assurer que les besoins et les attentes des utilisateurs étaient pris en compte, offrant une expérience fiable et conforme aux spécifications. Le prochain chapitre présentera l'implémentation des différentes APIs du projet.

CHAPITRES VI : IMPLEMENTATION DES APIs

Introduction

Le chapitre d'implémentation des APIs se concentre sur la réalisation concrète des services et des fonctionnalités nécessaires pour la gestion des besoins des enseignants. Ce chapitre constitue une étape cruciale du projet, car il s'agit de concrétiser les concepts et les décisions architecturales discutés précédemment.

Dans cette partie, nous présenterons les détails techniques de l'implémentation des APIs, en mettant l'accent sur les différentes fonctionnalités et services offerts par chaque API. Nous expliquerons comment chaque API a été conçue pour répondre aux besoins spécifiques des enseignants et faciliter la gestion de leurs commandes.

Nous aborderons également les technologies et les outils utilisés dans le processus d'implémentation, en décrivant les choix techniques effectués pour garantir des performances optimales, une scalabilité adéquate et une sécurité renforcée. Nous mettrons en évidence les bonnes pratiques de développement utilisées, telles que la modularité, la réutilisabilité du code et la gestion des erreurs.

De plus, nous présenterons des exemples concrets d'utilisation des APIs, en expliquant comment les enseignants peuvent interagir avec les différentes fonctionnalités pour passer leurs commandes, suivre l'état de leurs demandes et communiquer avec les responsables des services généraux

Enfin, nous évaluerons la qualité de l'implémentation en nous appuyant sur des critères tels que la conformité aux spécifications, la robustesse du code, les performances et la sécurité. Nous discuterons également des éventuelles difficultés rencontrées et des solutions mises en place pour les surmonter.

I. IMPLEMENTATION

1. Les serveurs utilisés pour notre application

1.1 SGBD(MySQL)

MySQL est une base de données relationnelle libre qui a vu le jour en 1995 et très employée sur le Web, souvent en association avec PHP (langage) et Apache (serveur web). MySQL

fonctionne indifféremment sur tous les systèmes d'exploitation (Windows, Linux, Mac OS notamment).

Le principe d'une base de données relationnelle est d'enregistrer les informations dans des tables, qui représentent des regroupements de données par sujets (table des clients, table des fournisseurs, table des produits, par exemple). Les tables sont reliées entre elles par des relations.

Le langage SQL (acronyme de Structured Query Language) est un langage universellement reconnu par MySQL et les autres bases de données et permettant d'interroger et de modifier le contenu d'une base de données[6].

1.2 Apache Tomcat

Apache Tomcat est un logiciel de serveur d'applications web open source conçu pour la programmation en Java et développé et maintenu par Jakarta, le groupe de projets open source Java de la fondation Apache.

L'objectif initial du logiciel Apache Tomcat est d'héberger et de déployer les servlets Java. Les servlets sont les programmes Java exécutés du côté serveur et qui reçoivent les requêtes des clients, les interprètent et génèrent les réponses demandées.

L'envoi des résultats peut se faire de façon directe ou via un protocole comme http.

Apache Tomcat fournit les fonctionnalités de base du traitement du serveur web pour les servlets Java. Il couvre tout le cycle de vie des servlets**,** qui se compose de 3 méthodes :

- Init () : qui se charge d'initier la servlet. Cette méthode est exécutée une seule fois, soit lors du démarrage du serveur, soit lorsqu'elle est déclenchée par le client.
- Service () : qui se charge de traiter les requêtes clients et générer les réponses adéquates.
- Destroy () : la méthode destroy () est invoquée par Tomcat à la fin de l'exécution de la servlet pour nettoyer les traces des activités et libérer les ressources

Tomcat est le logiciel de serveur web préféré des développeurs pour les implémentations Java. La dernière version stable d'Apache Tomcat 9.0.21 est sortie le 7 juin 2019.

D'un point de vue global, Apache Tomcat est chargé de fournir un environnement d'exécution pour les servlets. Il permet donc aux développeurs d'exécuter leurs applications web Java[8].

1.3 Serveur web (XAMPP)

XAMPP est un ensemble de logiciels permettant de mettre en place facilement un serveur Web, un serveur FTP et un serveur de messagerie.

Il s'agit d'une distribution de logiciels libres (X Apache MySQL Perl PHP) offrant une bonne souplesse d'utilisation, réputée pour son installation simple et rapide[8].

Il contient également le logiciel de gestion de bases de données MySQL et le serveur web apache Tomcat.

2. Les technologies utilisées

2.1 Java

Java est un langage de programmation largement utilisé pour coder des applications web. Il a été fréquemment choisi parmi les développeurs depuis plus de deux décennies, des millions d'applications Java étant utilisées aujourd'hui.

Java est un langage multiplateforme, orienté objet et centré sur le réseau, qui peut être utilisé comme une plateforme à part entière. Il s'agit d'un langage de programmation rapide, sécurisé et fiable qui permet de tout coder, des applications mobiles aux logiciels d'entreprise en passant par les applications de big data et les technologies côté serveur [9].

2.2 Spring boot et Spring cloud

Java Spring Boot (Spring Boot) est un outil qui accélère et facilite le développement d'applications Web et de microservices avec Spring Framework grâce à trois fonctionnalités principales : Configuration automatique, Une approche avisée de la configuration, La possibilité de créer des applications autonomes[10].

Spring Cloud fournit des outils pour les développeurs pour construire rapidement et facilement des patrons communs de systèmes répartis (tel que des services de configuration, de découverte ou de routage intelligent) [10].

2.3 Thymeleaf

Thymeleaf est un moteur de Template Java moderne côté serveur pour les environnements Web et autonomes. L'objectif principal de Thymeleaf est d'apporter *des modèles naturels* élégants à votre flux de travail de développement - HTML qui peut être correctement affiché dans les navigateurs et également fonctionner comme des prototypes statiques, permettant une collaboration plus étroite au sein des équipes de développement.

Avec des modules pour Spring Framework, une foule d'intégrations avec vos outils préférés et la possibilité de brancher vos propres fonctionnalités, Thymeleaf est idéal pour le développement Web HTML5 JVM moderne - bien qu'il puisse faire beaucoup plus[11].

2.4 Bootstrap

Bootstrap est un Framework frontal gratuit pour un développement Web plus rapide et plus facile. Bootstrap comprend des modèles de conception basés sur HTML et CSS pour la typographie, les formulaires, les boutons, les tableaux, la navigation, les modaux, les carrousels d'images et bien d'autres, ainsi que des plugins JavaScript facultatifs. Bootstrap vous donne également la possibilité de créer facilement des conceptions réactives [12].

2.5 JQuery

JQuery est ce qu'on appelle une « librairie » ou une « bibliothèque » JavaScript. Le rôle d'une librairie, en informatique, est de simplifier l'utilisation d'un certain langage de programmation en fournissant un ensemble de codes déjà prêts à l'emploi.

En l'occurrence, la librairie jQuery consiste en un ensemble de blocs de codes JavaScripts préconçus et qui vont être généralement enfermés dans des méthodes. Il va donc nous suffire d'appeler ces méthodes pour exécuter le code qu'elles contiennent [13].

3. Les outils utilisés pour l'implémentation

3.1 SwaggerHub

SwaggerHub est une plate-forme en ligne sur laquelle vous pouvez concevoir vos API, qu'il s'agisse d'API publiques, d'API privées internes ou de microservices. Le principe de base de SwaggerHub est Design First, Code Later. Autrement dit, vous commencez par disposer votre

API, ses ressources, ses opérations et ses modèles de données, et une fois la conception terminée, vous implémentez la logique métier.

Les définitions d'API sont écrites au format OpenAPI (anciennement connu sous le nom de Swagger) ou AsyncAPI. Ils sont enregistrés dans le cloud SwaggerHub et peuvent être synchronisés avec des systèmes externes tels que GitHub ou Amazon API Gateway. Vous pouvez également collaborer avec votre équipe sur SwaggerHub et maintenir plusieurs versions d'API au fur et à mesure de son évolution [14].

3.2 Intellij IDEA

IntelliJ IDEA est un IDE intelligent et tenant compte du contexte qui permet de travailler sur toutes sortes d'applications en Java et dans d'autres langages de la JVM tels que Kotlin, Scala et Groovy. De plus, IntelliJ IDEA Ultimate vous aide à développer des applications web full-stack grâce à ses puissants outils intégrés, à la prise en charge de JavaScript et de ses technologies connexes et à la prise en charge avancée de Framework populaires tels que Spring, Spring Boot, Jakarta EE, Micronaut, Quarkus et Helidon. IntelliJ IDEA peut être complété par des plugins gratuits développés par JetBrains afin de pouvoir travailler avec d'autres langages de programmation, parmi lesquels Go, Python, SQL, Ruby et PHP [15].

3.3 StarUML

StarUML est un outil de génie logiciel dédié à la modélisation UML et édité par la société coréenne MKLabs. Il est multiplateforme et fonctionne sous Windows, Linux et MacOS. La dernière version gère l'ensemble des diagrammes définis par UML 2, ainsi que plusieurs diagrammes SysM, le organigrammes, les diagrammes de flux de données, et les diagrammes entité-association [16].

3.4 Git

Git est un système de contrôle de version qui a été inventé et développé par Linus Torvalds, également connu pour l'invention du noyau Linux, en 2005.

Il s'agit d'un outil de développement qui aide une équipe de développeurs à gérer les changements apportés au code source au fil du temps.

Les logiciels de contrôle de version gardent une trace de chaque changement apporté au code dans un type spécial de base de données.

Git est le plus connu des VCS (versionning control system), c'est un projet open source très puissant qui est utilisé par l'ensemble de la communauté des développeurs [17].

3.5 Postman

Postman est un logiciel permettant de créer et de tester des requêtes HTTP. Il vous permet de les personnaliser dans les plus fins détails grâce à une interface ergonomique et intuitive. Vous pouvez choisir la méthode de la requête, entrer l'URL du serveur que vous voulez interroger, et rajouter tous les paramètres possibles pour une requête HTTP. Le logiciel tient un historique de vos requêtes. Il est très utile pour tester un API. Nous l'utiliserons pour s'initier au protocole HTTP en créant différentes requêtes [18].

4. Réalisation de l'API UTILISATEUR avec Spring Boot et Spring Cloud

a. Création d'un projet Spring Boot avec Spring Initializr

Une façon très pratique est d'utiliser Spring Initializr, qui se trouve sur le site start.spring.io. C'est un générateur d'applications qui permet de démarrer rapidement un projet Spring Boot. Sur le site start.spring.io, le formulaire de Spring Initializr va nous aider à créer notre projet. Ici nous pouvons choisir le type de projet : Maven ou Gradle, le langage de programmation : Java, Kotlin ou Groovy, la version de Spring Boot qu'on souhaite utiliser. On peut saisir aussi les metadata de notre projet, choisir le packaging : Jar ou War, choisir la version de Java qu'on souhaite utiliser. Et ici, on peut ajouter des dépendances vers d'autres librairies à notre projet. On peut par exemple ajouter Web, JPA, Security. On clique sur le bouton GENERATE, le projet est téléchargé en local sur notre machine [19].

Pour la création du projet nous devons ajouter des dépendances suivantes.

- Spring web : permet de créer des applications Web, à l'aide de spring MVC. Il utilise Apache Tomcat comme conteneur intègre par défaut.
- Spring Date JPA : Qui fait partie de la grande famille Spring Data, facilite la mise en œuvre de référentiels basés sur JPA. Ce module traite de la prise en charge améliorée des couches d'accès aux données basées sur JPA. Il facilite la création d'applications alimentées par Spring qui utilisent des technologies d'accès aux données.
- La mise en œuvre d'une couche d'accès aux données d'une application a été fastidieuse pendant un certain temps. Trop de code passe-partout doit être écrit pour exécuter des requêtes simples ainsi que pour effectuer la pagination et l'audit. Spring Data JPA vise

à améliorer considérablement la mise en œuvre des couches d'accès aux données en réduisant l'effort au montant réellement nécessaire. En tant que développeur, vous écrivez vos interfaces de référentiel, y compris les méthodes de recherche personnalisées, et Spring fournira automatiquement l'implémentation [20].

- Lombok : c'est une bibliothèque java utilisé pour réduire le code passe-partout. Maintenant on utilise seulement les annotations de Lombok pour générer les constructeurs, les getters et les setters.

  Exemple :

  @NoArgsConstructor : génère des constructeurs sans paramètre.

  @AllArgsConstructor : génère des constructeurs avec paramètre.

  @Data : pour la génération des getters et des setters.
- MySQL Driver : Dépendance utilisée pour se connecter à la base de données.
- Eureka Discovery Client : Dépendance utilisées pour mettre à chaque microservice qui démarrent de publier sa référence sur le service d'enregistrement.
- Config client : Dépendance utilisées pour centraliser la configuration de notre microservice
- Spring Security : Dépendance utilisées pour gérer l'authentification des utilisateurs dans l'application.

La **figure 36** ci-dessous illustre l'interface de création de projet Spring boot avec Spring Initializr.

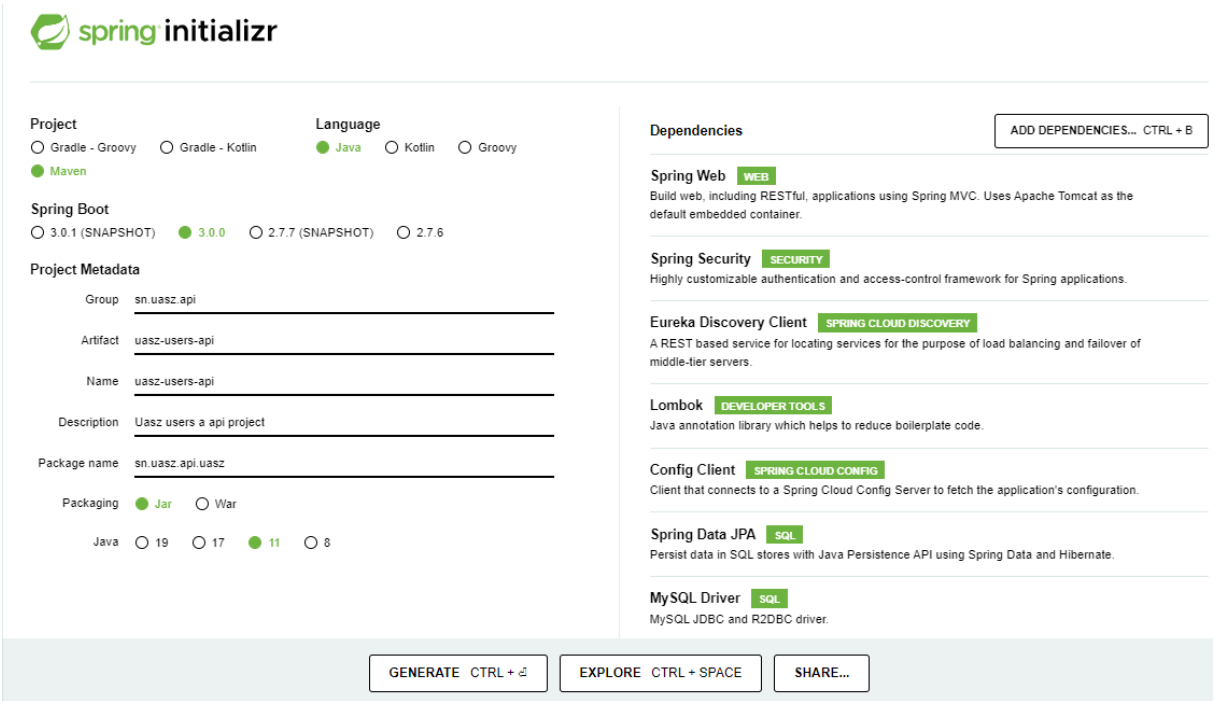


Figure 36: Interface de création de projet Spring boot avec Spring Initializr

b. Implémentation du logique métier

L'implémentation du logique métier est effectué en deux phases ;

Exemple API Utilisateur.

Première phase :

- Création d'un nouveau projet nomme « **uasz-user-spi** ».
- Ajouter le fichier du contrat sur le dossier '**resources'**
- Ajouter les dépendances au niveau du fichier '**POM'**
- Compiler avec « mvn spring-boot : run » pour générer le fichier .jar

Deuxième phase :

- Maintenant, on ouvre le projet « **uasz-users-api** » et on insère cette dépendance au niveau du fichier « **pom** » qui implémente l'ensemble des méthodes qui se trouve dans le fichier .jar.

c. Arborescence des fichiers de l'api

Dans cette partie nous présentons les dossiers qu'on a utilisés pour réaliser l'api.

Le dossier src : c'est dans ce dossier que tous les fichiers qui compose l'api sera défini. Ce dossier contient également des répertoires comme :

- **src/main/java** : Ici on trouve l'ensemble des fichiers java qui constitue la partie backend.
- **src/main/resources** : ce répertoire est composé de deux dossiers et un fichier qui permettent de gérer la partie frontend de l'application.

Le dossier static : contient la partie static de l'api.

Le dossier templates est compose de l'ensemble des fichiers html.

Le fichier application.properties permet de gérer la configuration de l'api.

Le dossier targette : Il contient les fichiers exécutables après compilation

Le fichier POM (Projet Object Model) : il contient l'ensemble des configurations, des dépendances, des plugins dont l'api a besoin. On trouve également dans ce fichier des informations comme la version de java utilisée, la version de spring etc. La figure ci-dessous montre l'arborescence.

Les APIs sont organisées sous forme de package. En java, un package est un groupe de classe, de sous-package et d'interface. Les packages permettent d'organiser les classes dans une structure de dossier, ce qui les rend plus faciles à trouver et à utiliser mieux encore les packages facilite la réutilisation du code.

d. Les packages da la partie Backend

i. Model

Dans ce packages on trouve l'ensemble des classes représentant les entités de notre api. Une entité peut être définie comme une instance de classe persistante c'est à dire on pourra sauvegarder ou charger dans une base de données relationnelle. Chaque entité a des propriétés (ou attributs) qui lui son propre. Chaque attribut est défini par un type de valeur qui peut être une chaine de caractère (string), un nombre entier (Integer) etc. Les entités crée sont ensuit générer en tables grâce aux annotations de JPA. Les annotations de JPA qui facilitent la création des tables sont les suivantes :

@**Entity** : cette annotation indique a JPA qu'il s'agit d'une entité persistante. si la classe n'est pas persistante dans ce cas on ignore l'annotation.

@**Table** : cette annotation est facultative car si elle est absente la table portera le même nom que celui de la classe. Elle est utilisée pour déterminer le nom de la table.

@**Id** : cette annotation indique qu'il s'agit la clé primaire de la table. Elle est obligatoire.

@**GeneratedValue** : permet de générer la valeur de la clé primaire.

@**Column** : Elle n'est pas obligatoire. Elle permet de donner un nom à la colonne grâce à **name,** de gérer certaines contraintes comme **nullable=false** qui indique que l'attribut ne doit pas être nul.

@JoinColumn () : Elle permet de marque une colonne comme une colonne de jointure pour une association d'entités.

@**OneToMany**(1-N), @**ManyToOne**(N-1), @**OneToOne** (1-1), @**ManyToMany**(N-N) : nous permet de mettre des relations entre nos différentes tables.

Etc…

ii. Repository

Le package « repository » qu'on nomme souvent DAO (Date Access Object) permet de définir l'ensemble des interfaces de notre api. Il est en relation avec la base de données. Pour utiliser les méthodes natives de Spring Data, il faudra que notre interface étende l'interface **CrudRepository**<UserDto, Long> avec le nom de la classe et le type de la clé primaire.

iii. Services

Le package « services » permet de définir l'ensemble des méthodes nécessaires pour notre api. Le package service peut être organise en deux sous package. Une partie pour gérer toutes les interfaces métiers et une autre partie pour gérer les classes implémentant les interfaces. La classe implémentant l'interface doit avoir l'annotation @**Service**.

iv. Controller

Le package « Controller » permet de gérer l'ensemble des contrôleurs de l'api.

@**RestController** : c'est la version spécialisée du contrôleur. Il inclut les annotations @Controller et @ResponseBody.

@**Autowired** : assure l'injection de dependance.il permet accéder à l'ensemble des méthodes trouvant dans l'interface.

@**GetMapping**, @**PostMapping**, @**DeleteMapping**, @**PutMapping** : gèrent les requêtes http correspondant à l'Uri donnée.

v. Mapper

Le package « mapper » permet de gérer l'ensemble des mappeurs de l'api. L'objectif des mappeurs c'est de faire la correspondance entre deux choses. Pour faire le Mappage entre les entités JPA et les DTO (Object de transfert de données) on a utilisé la dépendance MapStruct qui est un outil de générateur de code qui permet d'éviter le code passe-partout en mappant automatiquement les entités JPA en DTO et vice versa.

vi. Exception

Le package « Exception » permet de gérer l'ensemble des exceptions de l'api. En java les exceptions représentent le mécanisme de gestion des erreurs.

@**ControllerAdvice** est une annotation qui permet de gérer les exceptions sur l'ensemble de l'application globalement c'est-à-dire dans un seul composant.

@**ExceptionHandler (**value=**ImpossibleAjouterUser. Class)** est une annotation qui permet de gérer les exceptions spécifiques. L'attribut valu prend le nom de la classe qu'on gère ses exceptions.

vii. Security

Le package « **Security** » permet de gérer la partie sécurité de l'application. Quand on parle de sécurité on fait allusion à l'authentification, les contrôles d'accès et les redirections.

@**Configuration** : indique à spring que la classe doit être instanciée comme un Bean au démarrage.

@**EnableWebSecurity** : active les fonctionnalités nécessaires à la configuration.

e. Les package de la partie Frontend

i. Static

C'est dans ce package qu'on met toute la partie statique de notre application. Les éléments de cette partie interagissent avec les pages web qui se trouvent dans la partie Template de l'application.

Dans ce package on trouve les dossiers suivants :

- **Le dossier css** regroupe tous les fichiers CSS (Bootstrap-select-min, style)
- **Le dossier img** regroupe toutes les images.
- **Le dossier JS** regroupe les fichiers JS (Bootstrap-select-min, osahan)
- **Le dossier sass** renferme le fichier SCSS (style.scss)
- **Le dossier vendor** renferme un ensemble de dossier et fichier

ii. Templates

Le package « templates » permet de définir l'ensemble de nos pages web de l'application c'est-à-dire les fichiers HTML.

5. Implémentation des services techniques

Les services techniques sont les Edge Microservices utilisés dans l'orchestration. Les principaux services techniques utilisés sont **: Service Discovery, Service Config et Service Proxy**.

- **Implémentation du service Discovery**

Spring boot facilite la création des microservices. Avec seulement quelque annotation de Spring data on met en place un microservice. Le service Discovery permet à chaque microservice qui est démarré de se connecter avec lui pour publier sa référence c'est-à-dire son nom, son adresse IP et son port.

La **figure 37** ci-dessous illustre l'interface de création du service Discovery avec Spring Initializr.

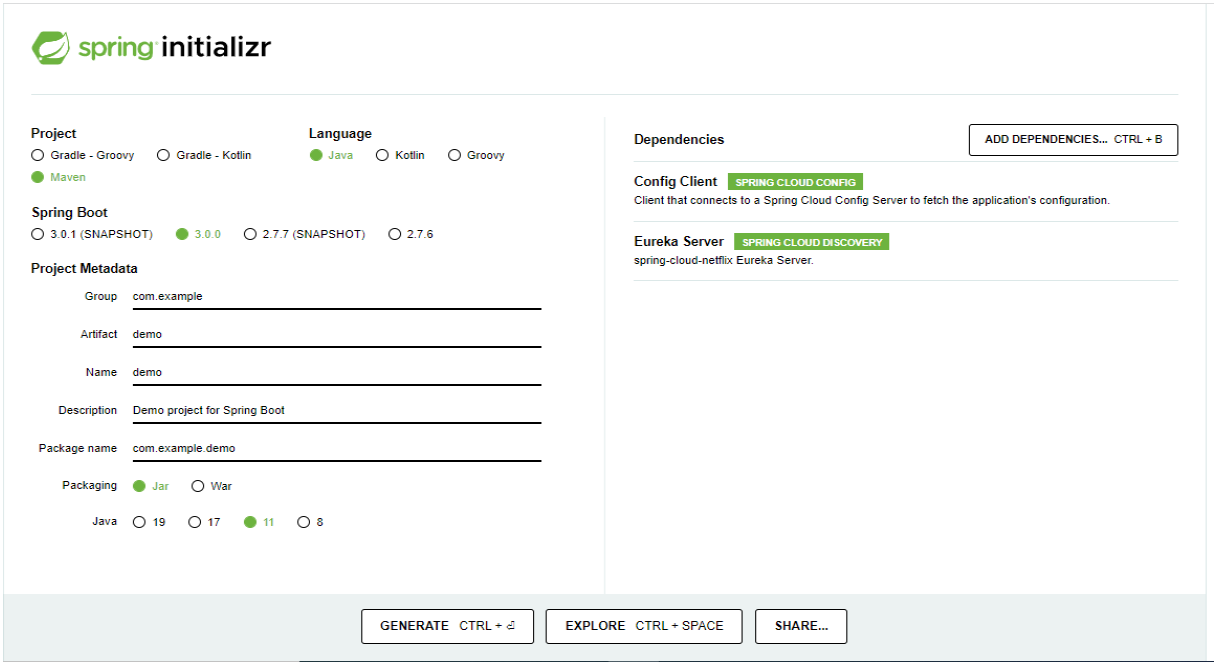


Figure 37:création du microservice « Discovery server »

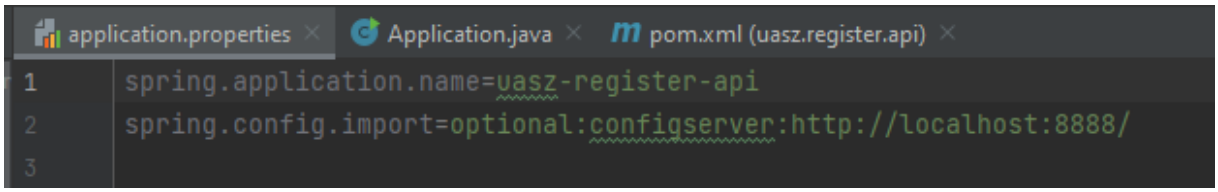
Apres la génération du projet, il faudra ajouter l'annotation @**EnableEurekaServer** dans la classe principale. Voir figure ci-dessous.

```
@EnableEurekaServer
@SpringBootApplication
public class Application {

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

}
```

Ensuite au niveau du fichier application.properties nous mettons Uri du microservice « **config server »** pour récupérer sa configuration. Voir figure ci-dessous



- **Implémentation du service Config**

Ce service permet de centraliser la configuration de l'ensemble de nos microservice c'est-à-dire au lieu de créer un fichier de configuration pour chaque microservice on va créer un repository GIT dans lequel on va mettre en place les fichiers de configuration et par la suite chaque microservice qui démarré demande le service « **Config server** » de lui fournir sa configuration. La **figure 38** ci-dessous illustre l'interface de création du service Config avec Spring Initializr.

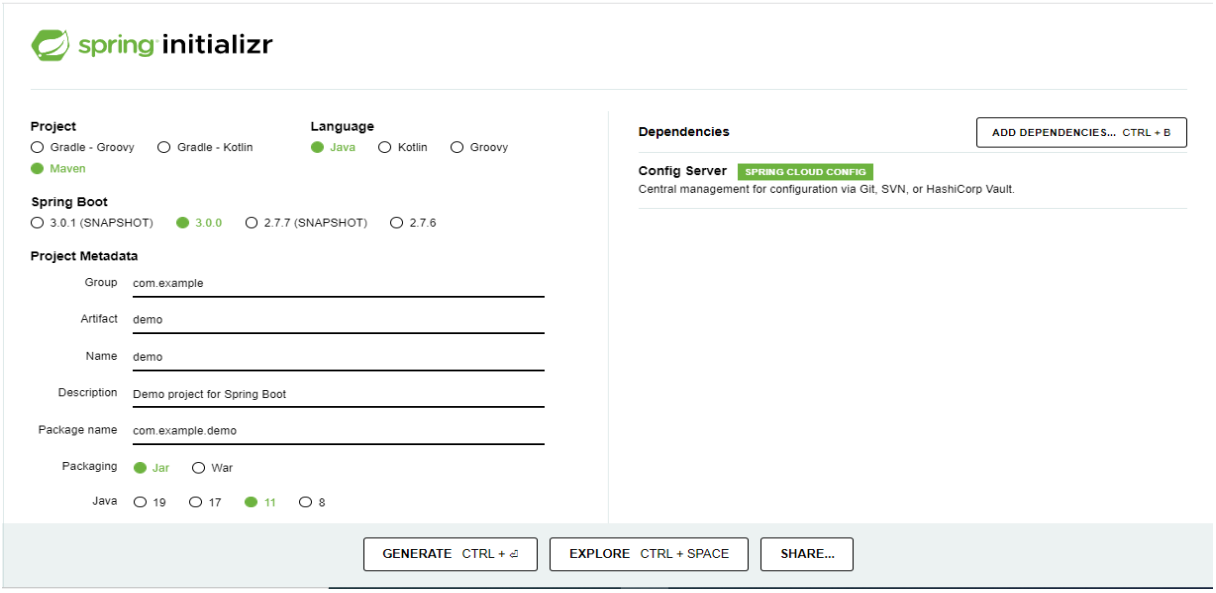


Figure 38: création microservice « Config server »

Apres avoir générer le projet, il faudra ajouter l'annotation @**EnableConfigServer** dans la classe principale. Voir figure ci-dessous.

```
1 usage
@EnableConfigServer
@SpringBootApplication
public class Application {

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

}
```

Ensuite au niveau du fichier application.properties nous mettons Uri du repository git distant contenant les fichiers de configuration. Voir figure ci-dessous

```
application.properties   Dockerfile
1  server.port=8888
2  spring.application.name=uasz.config.api
3  spring.cloud.config.server.git.uri=git@github.com:DiambarSENE/cloud-config.git
4  spring.cloud.config.server.git.default-label=master
5  spring.cloud.config.server.git.searchPaths=*-uasz
6  spring.cloud.config.server.git.username=DiambarSENE
7  spring.cloud.config.server.git.password=
```

- **Implémentation du service Proxy**

Ce service permet de rediriger les requêtes des clients vers le service concerné.

La **figure 39** ci-dessous illustre l'interface de création du service Proxy avec Spring Initializr.

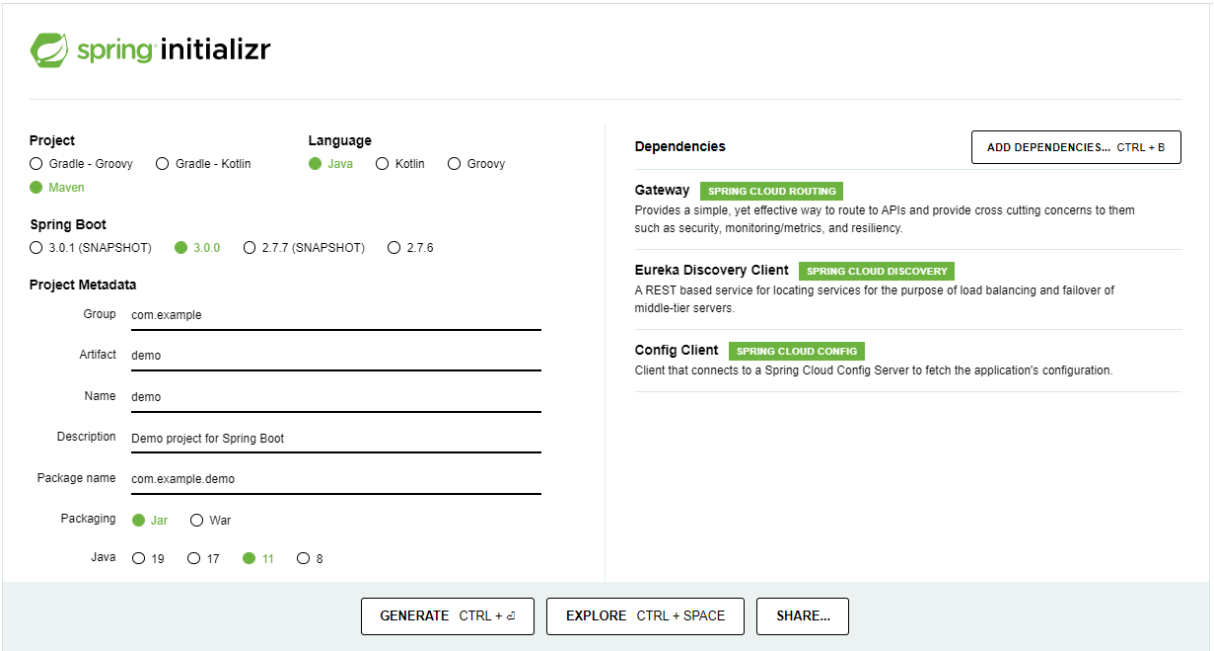


Figure 39: création du microservice « Proxy server »

Ensuite au niveau du fichier application.properties nous mettons Uri du microservice « **config server »** pour récupérer sa configuration. Voir figure ci-dessous

```
Application.java × templates\fragments.html × admin\fragments.html × application.properties ×
1  spring.application.name=uasz-proxy-api
2  spring.config.import=optional:configserver:http://localhost:8888/
3
```

II. PRESENTATION DU SERVICES DE L'API COMMANDE

Nous allons maintenant procéder à la vérification du bon fonctionnement de notre backend. Pour nous assurer que notre API Commande fonctionne correctement, nous pouvons nous appuyer sur les méthodes de requête HTTP dans Postman. Voici quelques exemples d'appels de méthode que nous avons employé :

GET : nous utilisons la méthode GET pour récupérer des informations sur une commande spécifique ou sur tous les commandes de l'API Commande. Par exemple, pour récupérer des informations sur une commande spécifique, nous avons envoyé une requête GET à l'API en spécifiant l'adresse email de la propriétaire de la commande dans l'URL. (Voir la **figure 40** ci-dessous).

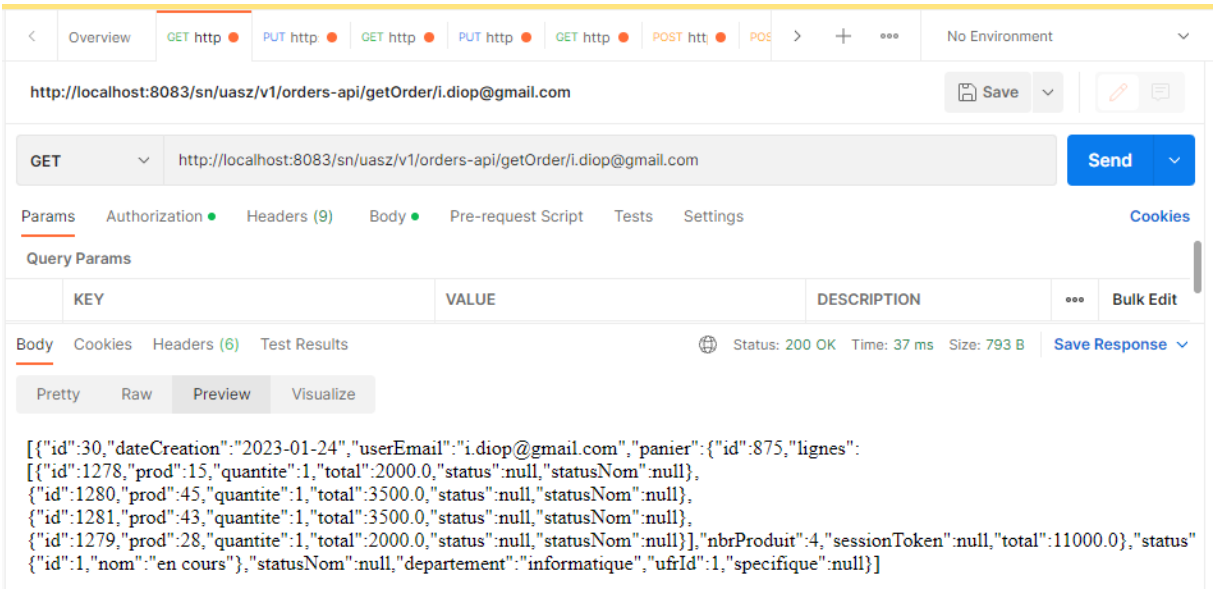


Figure 40: Réponse de la requête GET pour une commande spécifique avec Postman

Nous pouvons également récupérer toutes les informations sur toutes les commandes en envoyant une requête GET à l'URL de l'API. (Voir la **figure 41** ci-dessous).

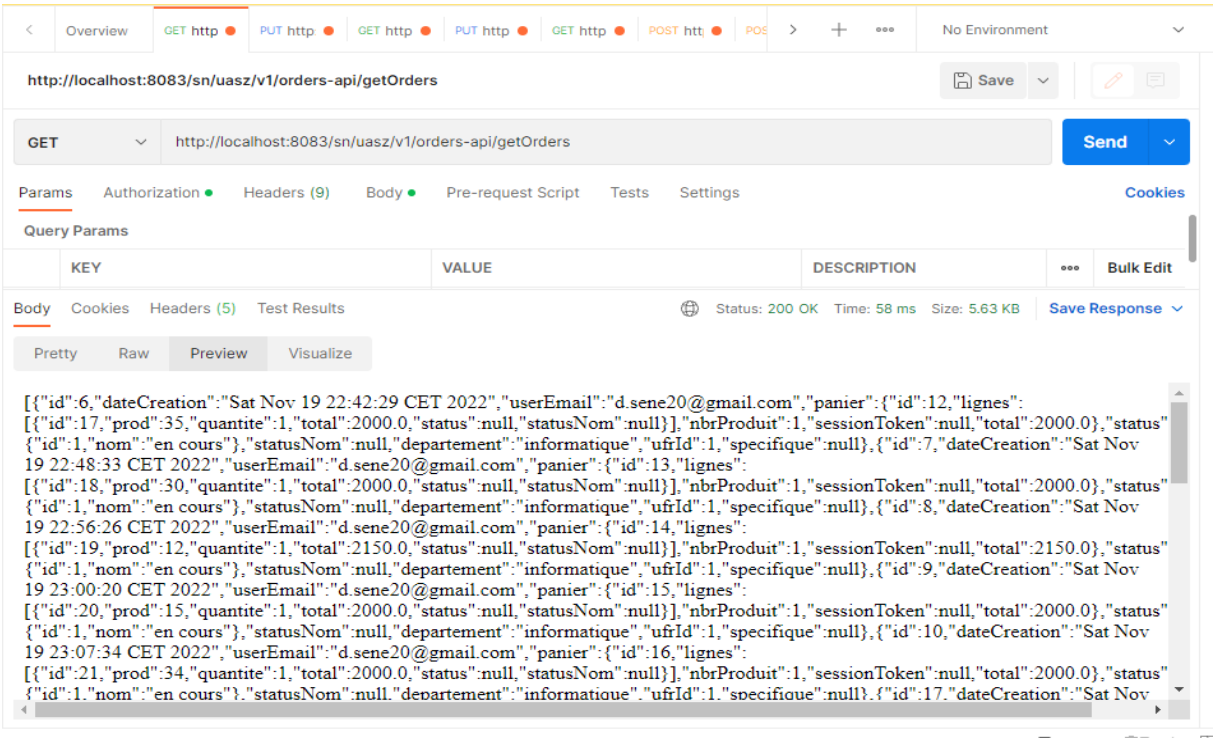


Figure 41:Réponse de la requête GET sur toutes les commandes avec Postman

Nous pouvons également récupérer toutes les commandes d'une UFR ou d'un département en spécifiant le nom de ce dernier dans l'URL de l'API. (Voir la **figure 42** ci-dessous).

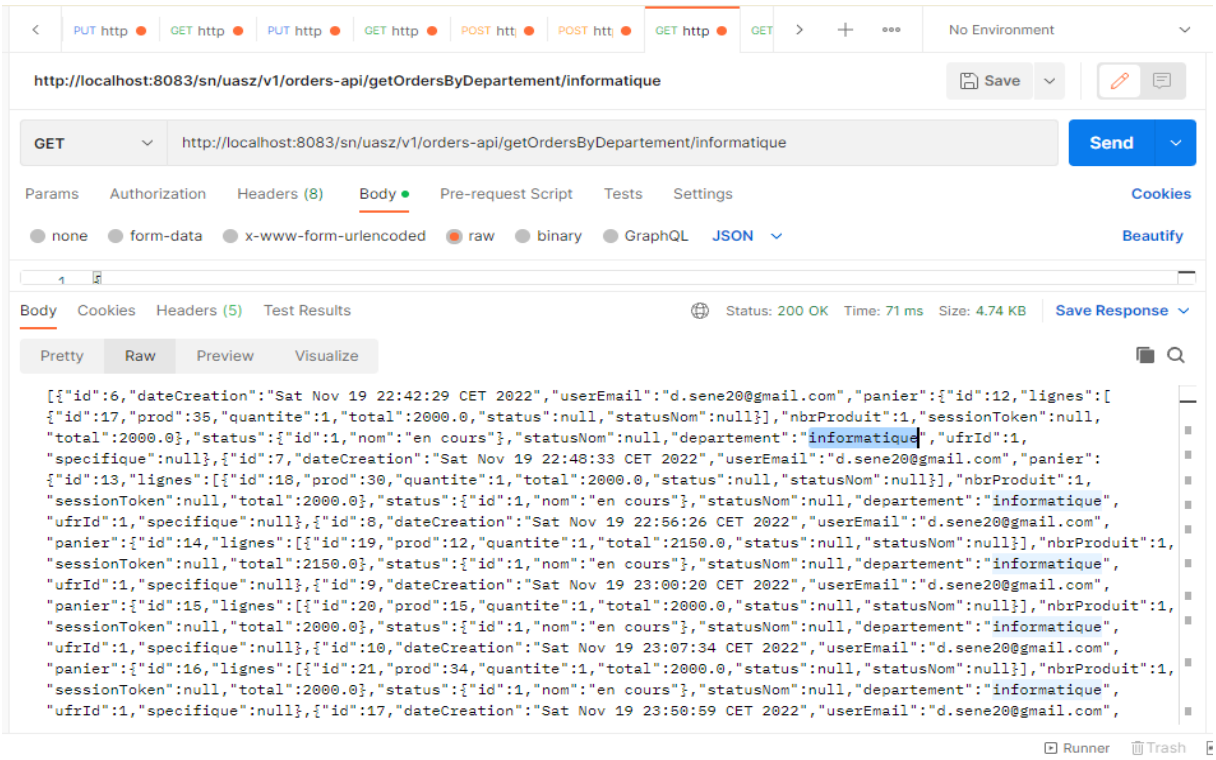


Figure 42:Réponse de la requête GET sur toutes les commandes du département Informatique avec Postman

POST : nous utilisons la méthode POST pour ajouter une nouvelle commande à notre API. Pour cela, nous devons envoyer les informations de la commande telles que l'identifiant du panier, le(s) ligne(s) du panier c'est-à-dire le nom du produit, la quantité et le prix unitaire, l'adresse mail, l'UFR, et le département de l'utilisateur à l'API en utilisant la méthode POST. Nous pouvons ensuite vérifier si la commande a été ajouté en envoyant une requête GET pour récupérer les informations de la commande que nous avez ajouté. (Voir **la figure 43** ci-dessous).

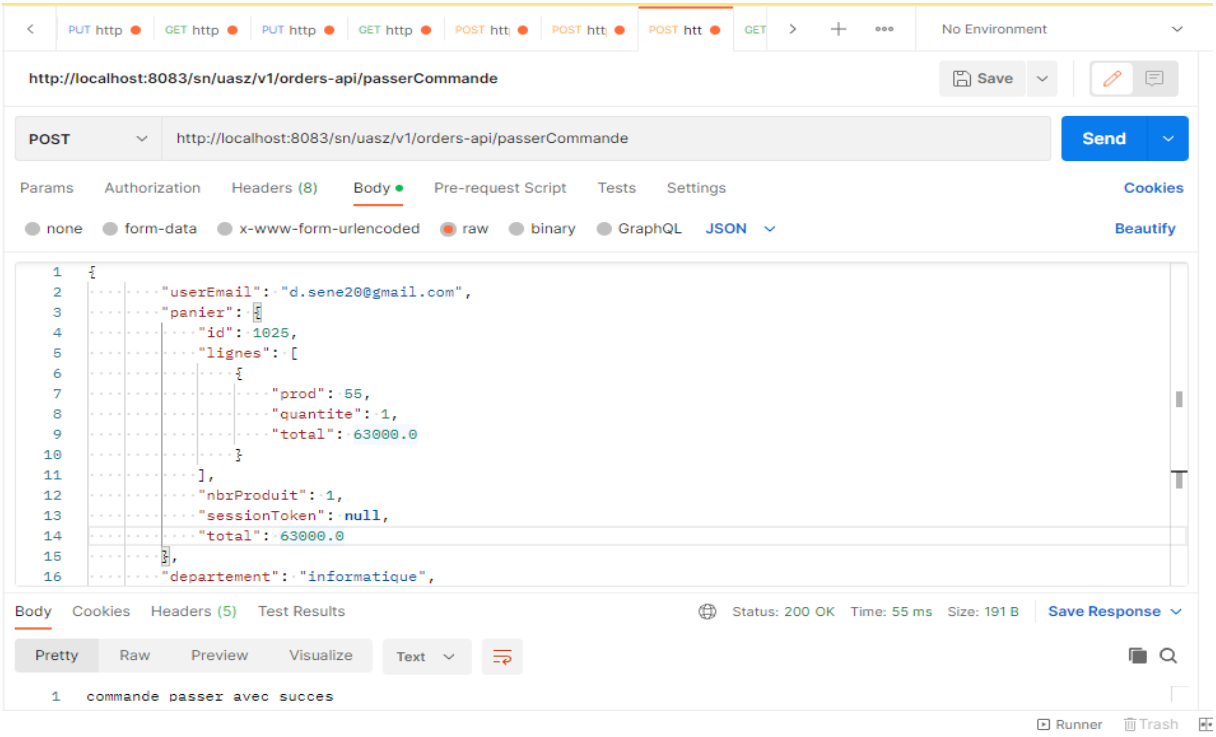


Figure 43: Ajout d'une nouvelle commande avec Postman

PUT : nous utilisons la méthode PUT pour mettre à jour les informations d'une commande existant dans notre API. Pour cela, nous devons envoyer les nouvelles informations de la commande, à l'API en utilisant la méthode PUT. Nous pouvons ensuite vérifier si les informations de la commande ont été mises à jour en envoyant une requête GET pour récupérer les informations de la commande mis à jour. (Voir **la figure 44** ci-dessous).

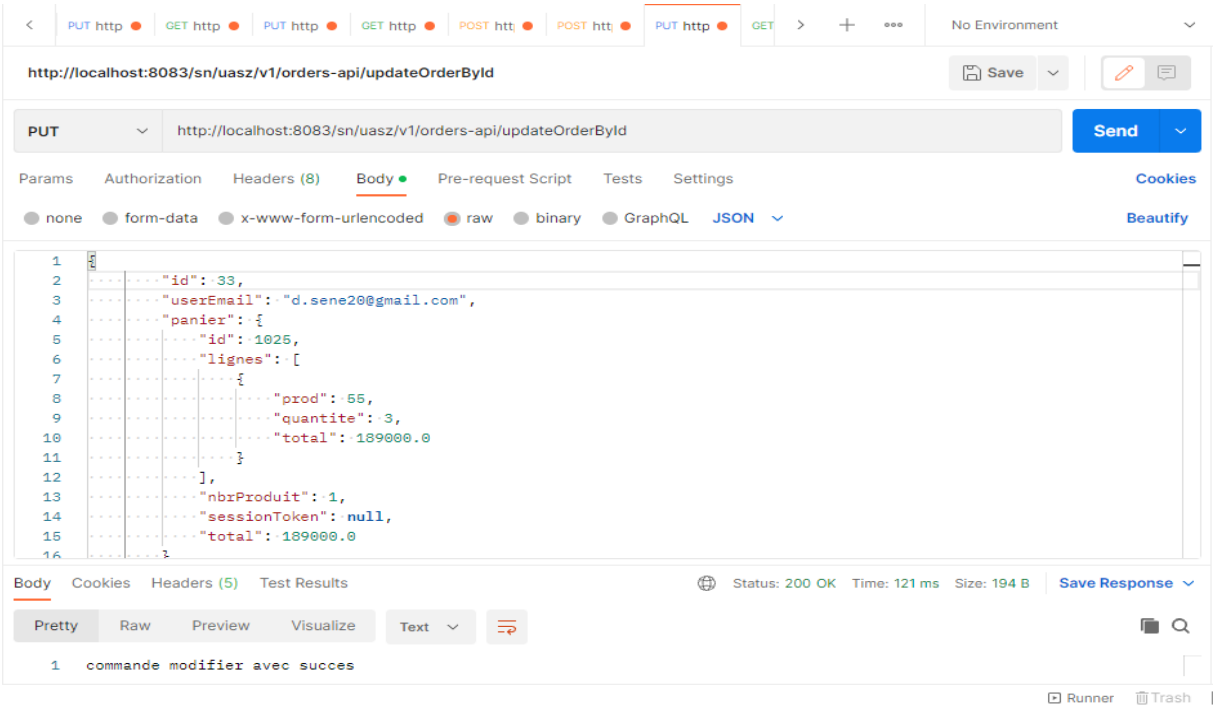


Figure 44:Modification d'une commande avec Postman

DELETE : nous utilisons la méthode DELETE pour supprimer une commande spécifique de notre API. Pour cela, nous devons envoyer une requête DELETE à l'API en spécifiant l'ID de la commande que nous souhaitons supprimer. Nous pouvons ensuite vérifier si la commande a été supprimé en envoyant une requête GET pour récupérer toutes les informations sur les commandes restants dans l'API. (Voir **la figure 45** ci-dessous).

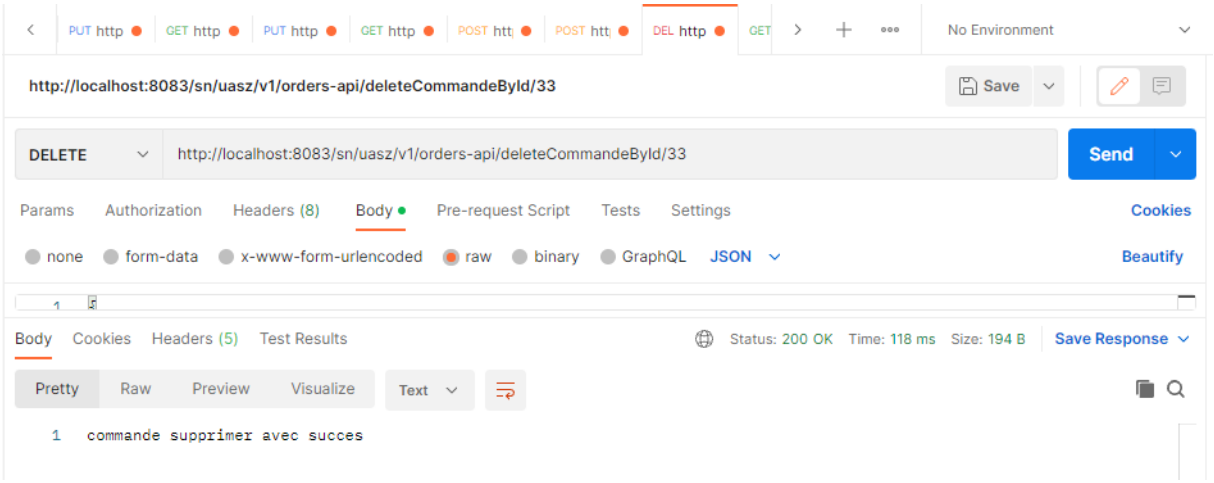


Figure 45: Suppression d'un utilisateur avec Postman

Conclusion

En utilisant ces méthodes de requête HTTP dans Postman, nous pouvons vérifier si notre API Commande fonctionne correctement en ajoutant, mettant à jour, récupérant et supprimant des

informations de la commande. Il est important de vérifier notre API avec différentes méthodes de requête HTTP pour garantir qu'elle fonctionne correctement et pour éviter les erreurs et les problèmes potentiels. Il est important de noter que nous avons effectué les mêmes opérations pour les autre API et tout fonctionne parfaitement.

Afin de faciliter l'utilisateur de l'application pour les utilisateurs, nous avons développé la partie frontend. Cela leur permet d'accéder aux APIs en utilisant des interfaces web conviviales. Le chapitre suivant présente l'application cote frontend.

CHAPITRES VII : PRESENTATION DE L'APPLICATION (FRONT END)

Introduction

Le chapitre de présentation de l'application (front-end) se concentre sur l'interface utilisateur et l'expérience utilisateur de la solution de gestion des besoins des enseignants. Ce chapitre vise à décrire la conception et la mise en œuvre de l'interface utilisateur, ainsi que les fonctionnalités offertes aux utilisateurs finaux.

Nous commencerons par présenter les objectifs et les principes de conception qui ont guidé le développement de l'interface utilisateur. Nous expliquerons comment nous avons cherché à offrir une interface intuitive, conviviale et adaptée aux besoins des enseignants. Nous aborderons les considérations d'ergonomie et d'accessibilité, ainsi que les meilleures pratiques de conception d'interfaces.

Ensuite, nous décrirons en détail les différentes pages et fonctionnalités de l'application. Nous expliquerons comment les enseignants peuvent accéder à leurs commandes, passer de nouvelles commandes, suivre l'état de leurs demandes, interagir avec les responsables des services généraux, etc. Nous mettrons en évidence les interactions, les flux de travail et les validations qui ont été implémentés pour garantir une expérience utilisateur fluide et efficace.

Nous aborderons également les aspects visuels de l'interface, en décrivant le choix des couleurs, des polices, des icônes et des images pour créer une identité visuelle cohérente et attrayante. Nous expliquerons comment l'interface a été adaptée pour s'adapter à différents appareils et résolutions d'écran, en utilisant des techniques de conception responsive.

Enfin, nous évaluerons l'expérience utilisateur de l'application en nous basant sur des critères tels que la facilité d'utilisation, l'efficacité des tâches, la satisfaction des utilisateurs et la conformité aux besoins des enseignants. Nous discuterons des retours d'utilisateurs et des éventuelles améliorations qui pourraient être apportées à l'interface pour optimiser l'expérience utilisateur.

Dans cette partie, nous allons faire la présentation de l'application à travers des captures d'écran des interfaces.

1. Pages des profils « enseignant » et « secrétaire »

Les enseignants et secrétaires ont les possibilités de parcourir le catalogue pour remplir leur panier des produits qu'ils désirent commander. Ils peuvent par la suite valider leur commande conformément au budget prévu par son département et faire le suivi de leur commande dans le processus d'expression de besoin.

1.1. Page d'authentification

Cette page permet aux utilisateurs qui ont déjà un compte de se connecter pour accéder à l'ensemble des fonctionnalités réservé à son profil. La **figure 46** ci-dessous donne un aperçu de cette page.

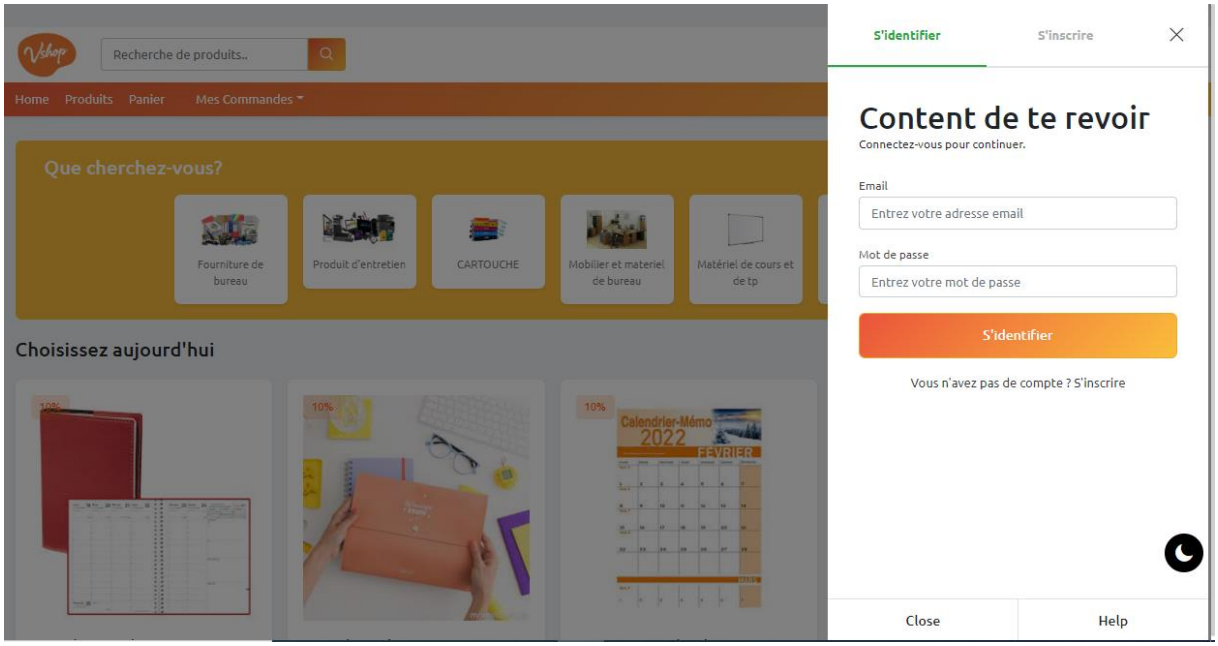


Figure 46: Page d'authentification

1.2. Page d'accueil et le catalogue des produits

L'objectif de la page d'accueil de l'application est de présenter à l'utilisateur l'ensemble des produits disponibles, de manière à ce qu'il puisse les visualiser facilement. La **figure 47** ci-dessous donne un aperçu de cette page.

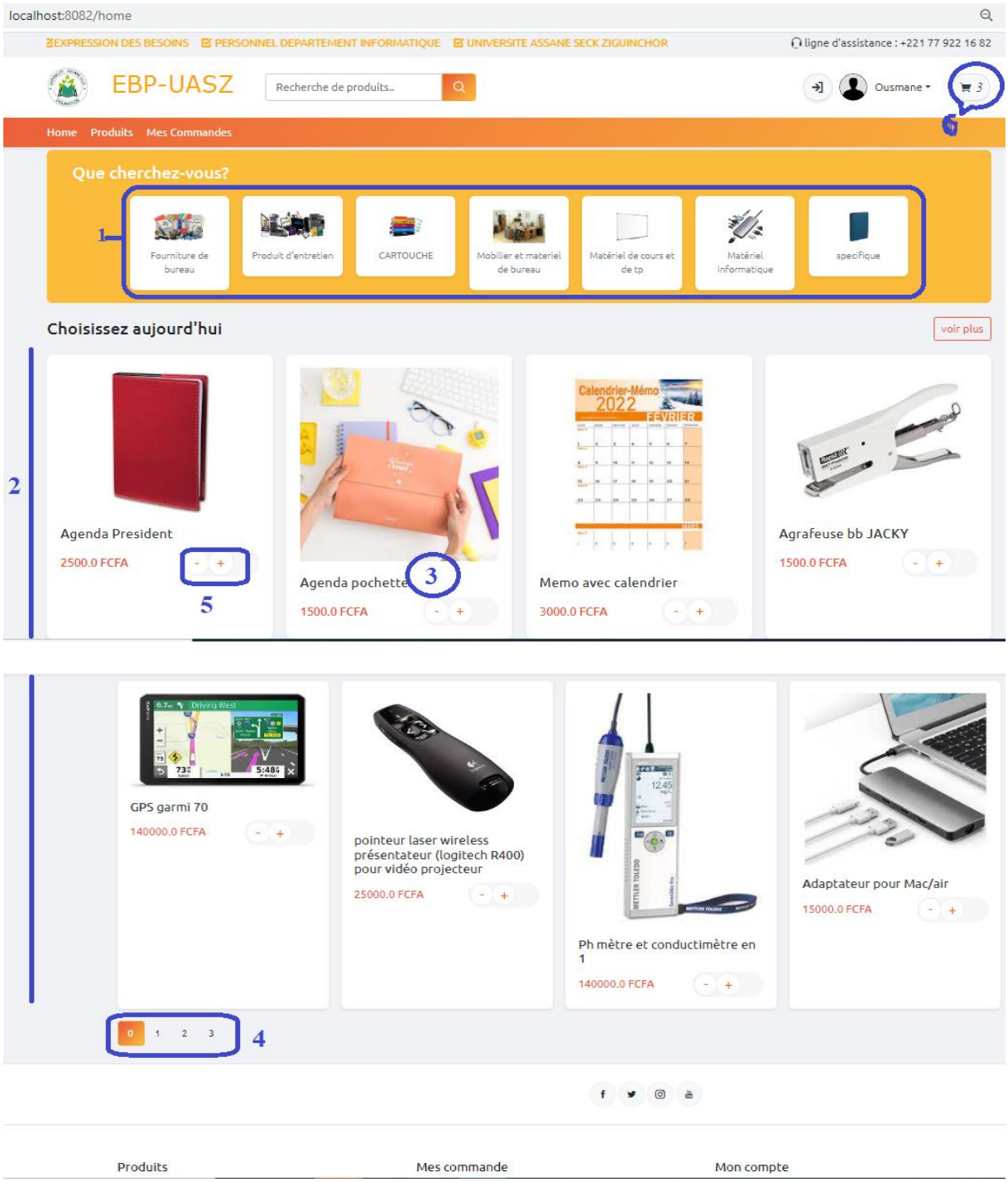


Figure 47: Page d'accueil des enseignants et des secrétaires de département

Cette page est divisée en six sections :

1. Liste des catégories permettant de trier les produits par catégories.
2. Liste des produits pour une page donnée.
3. Informations détaillées pour chaque produit.
4. Pagination pour naviguer entre les pages de produits.
5. Boutons d'ajout et de suppression de produit dans le panier.
6. Panier.

1.3.Page panier

Dans cette section, l'utilisateur peut finaliser sa commande. Cependant s'il décide d'ajouter d'autres articles, supprimer un produit ou des produits du panier, il peut cliquer sur le bouton approprie. La **figure 48** ci-dessous donne un aperçu de cette page.

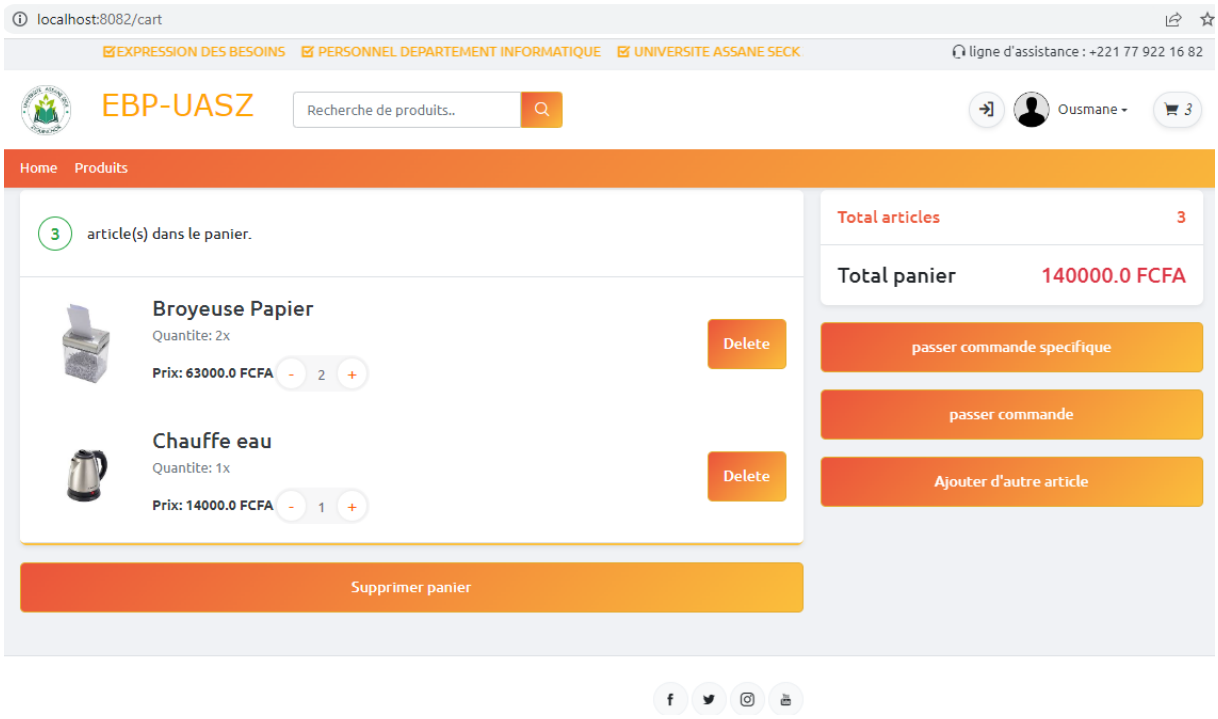


Figure 48: Page panier

1.4.Page de suivi des commandes

Cette page (**Figure 49**) donne à l'utilisateur la possibilité de suivre la commande. Il peut ici obtenir des informations sur l'état de sa commande, telles que « en cours », « confirmer », « annuler ».

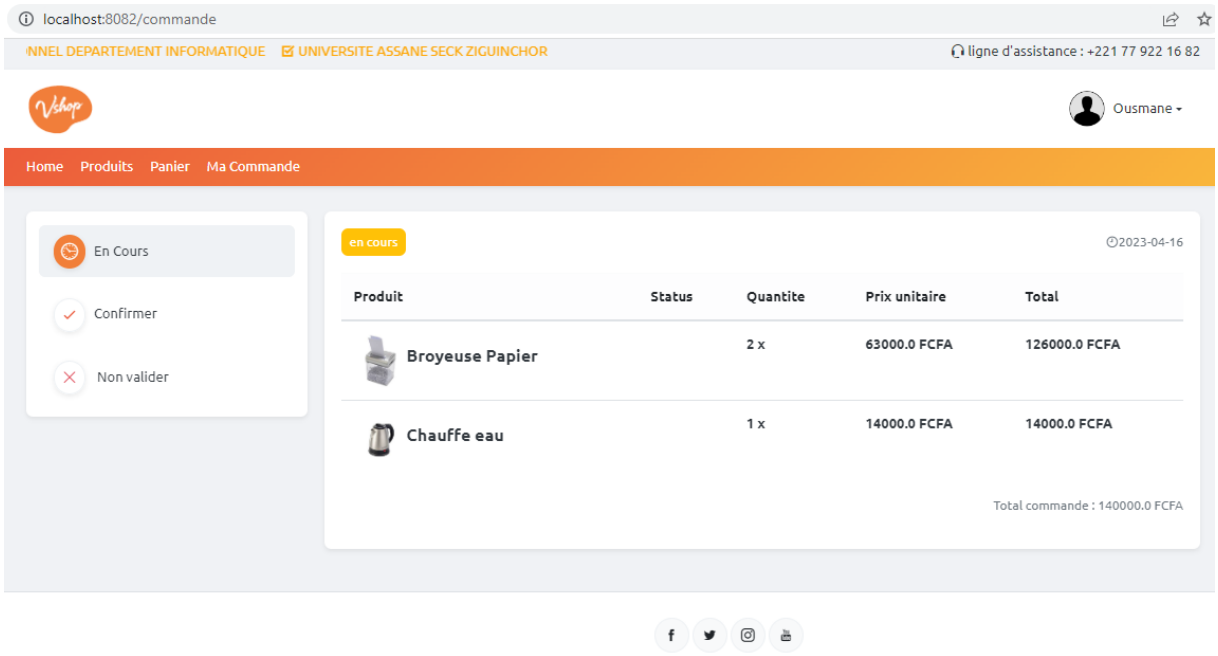


Figure 49: Page suivi des commandes

Le chef de département est charge de gérer les commandes des enseignant de son département ainsi que les commandes spécifiques et le budget matériel. Il a la possibilité d'importer un fichier CSV. Dans cette partie, nous allons présenter uniquement sa page d'accueil et la liste des commandes.

2. Pages du profil « chef de département »

Sur cette page, le chef de département peut consulter le nombre de commandes confirmées par les utilisateurs ainsi que les commandes spécifiques. La **figure 50** ci-dessous donne un aperçu de cette page.

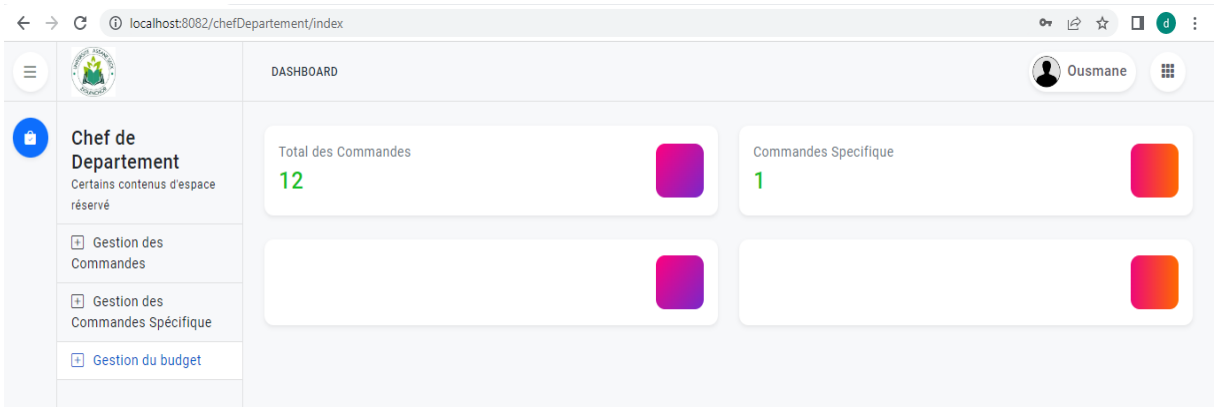


Figure 50: Tableau de bord d'un chef de département

2.1.Page liste des commandes d'un département

Cette interface permet au chef de département de consulter les informations sur les commandes confirmées par les enseignants et les secrétaires du département, et de visualiser les détails, de les confirmer, de les modifier ou de les supprimer. La **figure 51** ci-dessous donne un aperçu de cette interface.

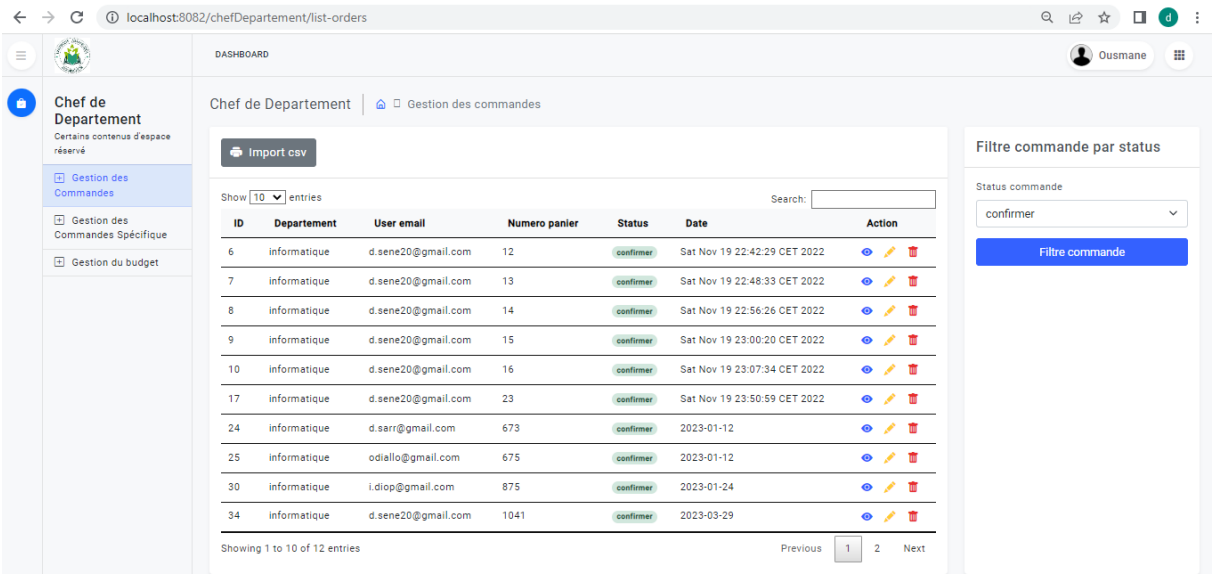


Figure 51: page liste des commandes d'un département

2.2.Page de détail commande à confirmer par le chef de département

Le chef de département peut consulter les informations détaillées de la commande d'un enseignant en cliquant sur le bouton « voir détail ». La **figure 52** ci-dessous donne un aperçu de cette page.

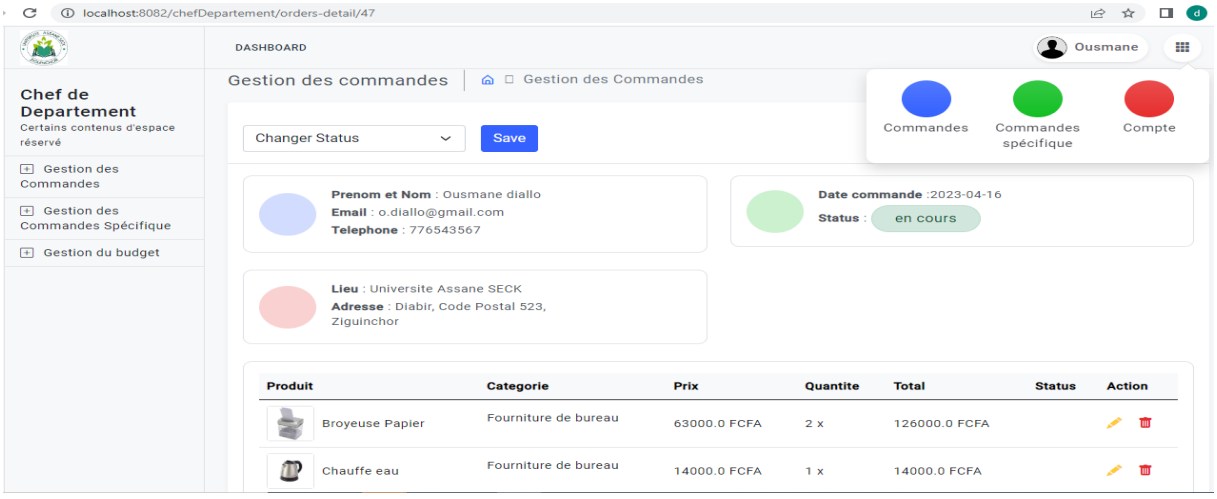


Figure 52:page de détail d'une commande donnée

3. Responsable des services généraux

Le responsable des services généraux est charge de gérer les commandes confirmées par les chefs de département ainsi que le catalogue. Il a la possibilité d'importer un fichier CSV. Dans cette partie, nous allons présenter uniquement la page liste des commandes confirmées et la page de détail d'une commande donnée.

3.1. Page liste des commandes confirmées par les chefs de département

Cette interface permet au responsable des services généraux de visualiser les informations concernant les commandes approuvées par le chef de département, et de changer le statut de la ligne de produit. La **figure 53** ci-dessous donne un aperçu de cette interface.

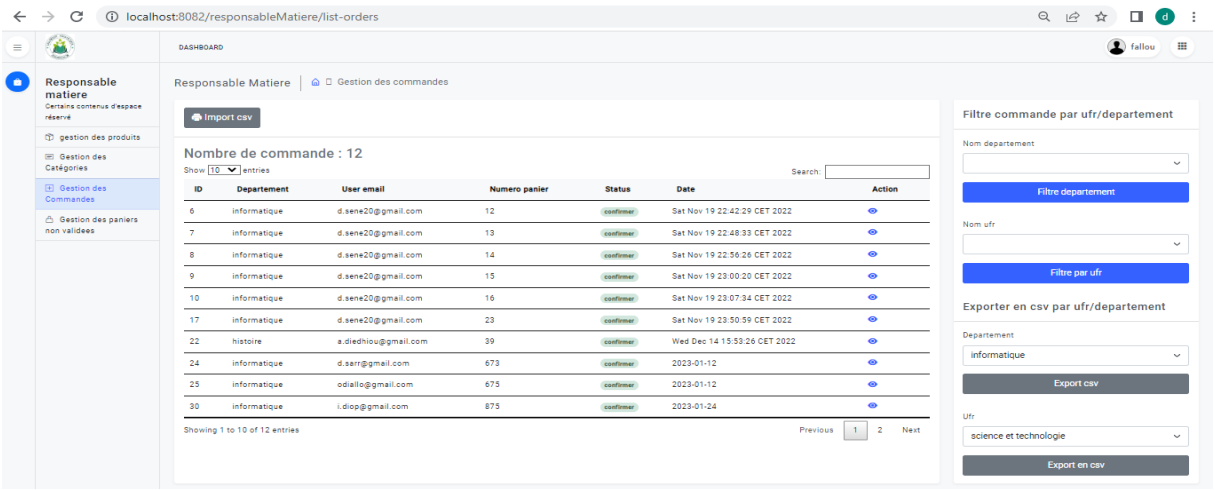


Figure 53:page liste des commandes confirmées

3.2. Page de détail commande confirmer pour le responsable des services généraux

Le responsable des services généraux peut consulter les informations détaillées sur les commandes approuvées par le chef de département en cliquant sur le bouton « voir détail ». La **figure 54** ci-dessous donne un aperçu de cette page.

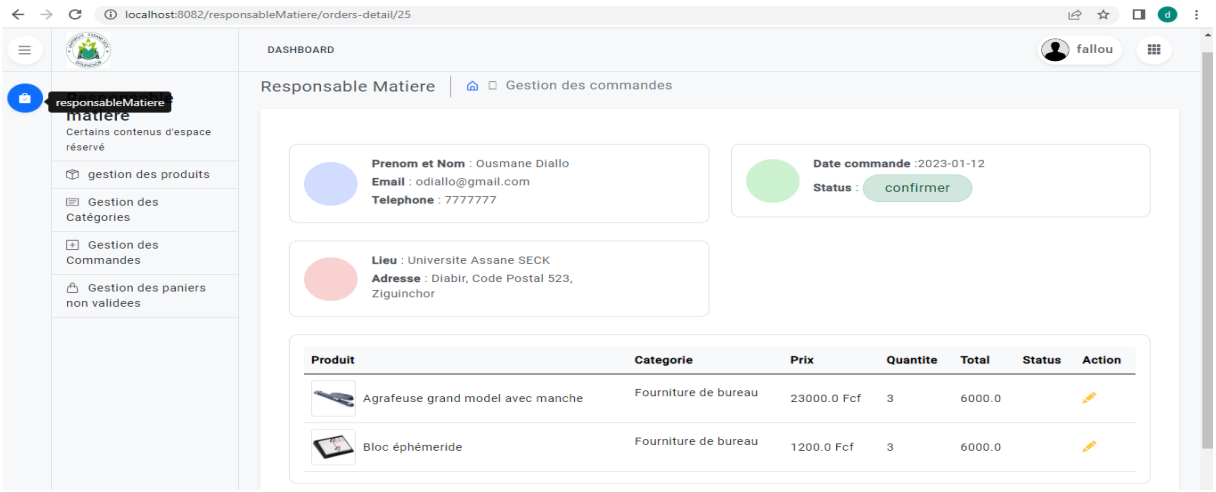


Figure 54:page de détail de commande du responsable des services généraux

La gestion des utilisateurs, des rôles, des UFRs et des départements est sous la responsabilité de l'administrateur, mais cette partie ne sera pas présentée dans ce mémoire.

En conclusion, ce chapitre a mis avant les interfaces de l'application, en précisant les utilisateurs ayant accès à celles-ci en soulignant que l'application est responsive. La conclusion générale de ce mémoire, qui suit, ouvre également des perspectives d'amélioration pour notre système.

En somme, cette présentation de l'application (front-end) a mis en évidence l'importance de concevoir une interface conviviale et adaptée aux besoins des enseignants pour faciliter la gestion des besoins. L'interface utilisateur offre une expérience intuitive, avec des fonctionnalités permettant aux enseignants d'accéder facilement à leurs commandes, de les suivre et de communiquer avec les responsables des services généraux. Cette interface contribue à améliorer l'efficacité et la satisfaction du personnel enseignant dans la gestion de leurs besoins

CONCLUSION

Notre projet de mémoire consistait à dématérialiser (ou automatiser) la gestion de l'expression des besoins du personnel de l'UASZ. Pour atteindre cet objectif, nous avons développé une application web divisée en deux parties : le back-end et le front-end.

En back-end, notre architecture est basée sur l'approche microservices en utilisant les technologies Java, Spring Boot et Spring Cloud. L'utilisation de ces technologies nous a permis de développer une application performante, fiable, facile à maintenir et sécurisée.

En font-end, l'utilisation de Thymeleaf nous a également permis de créer une interface utilisateur conviviale afin de faciliter l'expression des besoins du personnel au sein de l'UASZ.

Grâce à cette application, les enseignants pourront désormais passer leurs commandes de manière autonome, ce qui devrait leur permettre de gagner du temps et d'améliorer leur efficacité. De plus, en automatisant la gestion des commandes, nous espérons que les UFR pourront améliorer la qualité des achats de matériels, dès lors que l'achat est motivé par le personnel futur bénéficier.

La principale limite de l'application se trouve dans la manière actuelle d'ajouter les paniers, qui sont stockés dans la base de données dès leur création et qui peuvent parfois ne pas être liés à un utilisateur. Cela est dû au fait que nous avons donné à l'utilisateur la possibilité de remplir son panier et de se connecter par la suite. Ainsi, les paniers sans utilisateur restent dans la base de données. La solution consiste à utiliser des cookies pour stocker ces paniers lorsque l'utilisateur n'est pas connecté et une fois connecté, de les enregistrer dans la base de données. La résolution de ce problème constitue notre première perspective d'amélioration.

La force de l'application réside dans son architecture, qui est basé sur l'approche microservices. Ainsi, grâce à cette architecture, nous avons de nombreuses possibilités pour améliorer l'application et la rendre encore plus utile pour l'UASZ et son personnel. Par exemple, nous pouvons :

- améliorer les API déjà réalisées en fonction des retours des utilisateurs ;
- ajouter de nouvelles fonctionnalités ou API, telles que l'API Statistiques qui est dans l'architecture mais qui n'a pas encore été implémentée, ou l'API pour la comptabilité des matières
- intégrer l'application à d'autres outils de gestion utilisés par les UFR.

BIBLIOGRAPHIE

[1] Manuel de procédures de l'UASZ

WEBOGRAPHIE

[2] https://aws.amazon.com/fr/microservices/ (consulté le 12/12/2022)

[3] https://www.univ-constantine2.dz/CoursOnLine/Benelhadj-Mohamed/co/grain3_2.html

(Consulté le 12/10/2022)

[4] http://www.mosaique-info.fr/glossaire-web-referencement-infographie-multimedia-informatique/m-glossaire-informatique-et-multimedia/448-mysql-definition.html (consulté le 15/10/2022)

[5] https://www.syloe.com/glossaire/apache-tomcat/ (consulté le 15/10/2022)

[6] https://desgeeksetdeslettres.com/xampp-plateforme-pour-heberger-son-propre-site-web/ (consulté le 15/10/2022)

[7] https://www.ibm.com/topics/java-spring-boot (consulté le 20/10/2022)

[8] https://www.thymeleaf.org/ (consulté le 01/11/2022)

[9] https://aws.amazon.com/fr/what-is/java/ (consulté le 09/11/2022)

[10] https://www.w3schools.com/bootstrap/bootstrap_get_started.asp (consulté le 17/11/2022)

[11] https://www.pierre-giraud.com/jquery-apprendre-cours/introduction/ (consulté le 23/11/2022)

[12] https://support.smartbear.com/swaggerhub/docs/tutorials/getting-started.html(consulté le 01/12/2022)

[13] https://www.jetbrains.com/fr-fr/idea/features/#:~:text=Qu'est%2Dce%20qu',que%20Kotlin%2C%20Scala%20et%20Groovy. (Consulté le 03/12/2022)

[14] https://fr.wikipedia.org/wiki/StarUML (consulté le 05/12/2022)

[15] https://www.next-decision.fr/wiki/qu-est-ce-que-git (consulté le 07/12/2022)

[16] https://explorweb.github.io/cours2018/cours/postman.html (consulté le 07/12/2022)

[17] https://www.linkedin.com/learning/l-essentiel-de-spring-boot (consulté le 08/12/2022)

[18] https://spring.io/projects/spring-data-jpa (consulté le 09/12/2022)

[19] https://swagger.io/docs/specification/about/ (consulté le 10/12/2022)

[20]https://www.planzone.fr/blog/quest-ce-que-la-methodologie-agile (consulté le 02/02/2023)

[21]https://mobiskill.fr/blog/conseils-emploi-tech/microservices-ou-monolithes-quelle-architecture-choisir/ (consulté le 02/02/2023)

[22] https://www.synopsys.com/blogs/software-security/agile-cicd-devops-difference/(consulté le 03/02/2023)