RÉPUBLIQUE DÉMOCRATIQUE DU CONGO

Ministère de l'Enseignement Supérieur et Universitaire

UNIVERSITÉ NOUVEAUX HORIZONS
OSER INNOVER PERSÉVÉRER

Développement d'une application d'aide à la délibération des étudiants de l'Université Nouveaux Horizons

---

Travail de fin de cycle présenté en vue de l'obtention du diplôme de licence en Informatique de Gestion

Présenté par : Musanzi Mavula Wilfried

Directeur : Dr. Saint Jean Djungu
Co-directeur : Ass. Jonathan Kabemba

Année académique 2022 - 2023

RÉSUMÉ

ABSTRACT

RÉMERCIEMENTS

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

0 INTRODUCTION GÉNÉRALE

0.1 APERÇU GÉNÉRALE

Ce présent travail consiste à la conception et au développement d'une application web d'aide à la déliberation des étudiants à l'université Nouveaux Horizons, dans le but de faciliter la gestion des données des étudiants, la publication des résultats et de réduire le circuit de circulation des informations.

Nous parlons d'aide à la déliberation des étudiants parce que la déliberation des étudiants est un processus complexe et prend en compte plusieurs aspects tels que la conduite, les notes, la présence, etc. Et poutant dans ce travail nous n'avons pris en compte qu'un seul aspect qui est la performance académique des étudiants.

Pour ce faire, nous avons commencé par analyser la problématique et les besoins de l'université Nouveaux Horizons en matière de déliberation des étudiants, envue de proposer une solution adéquate.

Nous avons ensuite fait une étude comparative des applications existantes, des technologies utilisées pour leur développement afin de nous inspirer de leurs fonctionnalités et de leurs technologies. Nous nous sommes aussi intéressés à la manière dont les autres universités gèrent cette problématique.

Nous avons, après une étude minitieuse des applications existantes procédé à la conception de notre application en nous basant sur les besoins de l'université Nouveaux Horizons et sur l'extension des fonctionnalités des applications existantes pour ne pas réinventer la roue.

Nous avons ensuite procédé au développement de notre application en utilisant les technologies que nous avons jugées les plus adaptées.

0.2 CONTEXTE ET MOTIVATION

Ce projet est axé sur la déliberation des étudiants à l'université Nouveaux Horizons, comme mentionner ci-dessus la déliberation est un processus prend en compte plusieurs aspects et nous nous sommes focalisés sur un seul aspect qui est la performance académique de ces derniers.

Ce travail trouve tout son essort dans le fait que l'université Nouveaux Horizons est une institution qui se veut moderne, nous voulons donc apporter notre contribution à l'essor de cette institution en lui fournissant un outil qui répondra à ses besoins et qui pourra s'intégrer facilement dans son système d'information.

Nous sommes aussi motivés par le fait de pouvoir fournir une base qui pourra être étendue pour obtenir un système de gestion des données des étudiants plus complet et plus efficace.

0.3 PROBLÉMATIQUE

Une meilleure solution est celle qui répond à un réel besoin, l'université Nouveaux Horizons est actuellement confrontée à un problème de gestion des données des étudiants et de publication des résultats.

Voici un bref aperçu du circuit de circulation des informations :

— Après les examens le professeur ou le chargé du cours envoie soit les notes annuelles soit les notes de tous les travaux ansi que l'examen au décanat de l'Université en copie au doyen de la faculté.
— Le décanat après réception transmet les notes réçues soit au format pdf soit Excel(xlsl) soit manuscrit au président du jury.
— Le jury à son tour calcule les moyennes et après déliberation communique les résultats aux étudiants.

Vous conviendrez avec nous que le circuit de circlulation des informations et assez long, de plus, il est difficile de gérer les données des étudiants de manière efficace et la publication des résultats n'est pas aussi évidente qu'elle le devrait.

Par ailleurs, il est difficile de suivre l'évolution des étudiants d'une année à une autre.

Les dites données peuvent être :

— les cours en compléments,
— le(s) relevé(s) de chaque année parce qu'un étudiant peut en avoir plusieurs en une année,
— etc.

0.4 MÉTHODES ET TECHNIQUES

Pour la réalisation de ce travail nous avons utilisé plusieurs méthodes et techniques qui nous ont permis de mener à bien notre projet en nous permettant de réunir les informations nécessaires envue d'en tirer des conclusions et de proposer la solution la mieux adaptée.

0.4.1 *Méthodes*

1. La méthode analytico-déductive : Nous avons analyser la problématique évoquée ci-dessus en partant des faits concrets pour aboutir à une conclusion générale.
2. La méthode descriptive : Nous avons décrit notre problématique de manière précise et objective.
3. La méthode comparative : Nous avons eu à comparer la manière dont la problématique est gérée ailleurs pour en dégager les similitudes et les différences.

0.4.2 *Techniques*

Nous avons utilisée est la technique documentaire et plus précisément la technique de la recherche bibliographique.

Nous avons consulté des ouvrages, des articles, des documents et des sites web pour avoir des informations sur les applications existantes et les technologies utilisées pour leur développement. Nous avons aussi consulté des documents sur les méthodes de conception et de développement d'applications web.

Nous avons aussi utilisé la technique de l'entretien pour avoir des informations sur les besoins du corps acédemique de l'Université Nouveaux Horizons et sur les fonctionnalités qu'ils souhaiteraient avoir dans une application d'aide à la délibération également sur la manière dont les universités soeurs gérent cette problématique.

0.5 ETAT DE LA QUESTION

Etant dans une université la déliberation est un processus commun, nous avons pensé que d'autres universités ont forcément eu à faire face à la même problématique que nous.

Nous nous sommes donc intéressés à la manière dont les autres universités gèrent la délibération des étudiants et les outils qu'elles utilisent.

Nous avons trouvé des concepts qui nous ont aidé à mieux comprendre la problématique et à mieux cerner les besoins.

Nous nous manquerons pas de les mentionner dans la suite de ce travail.

0.6 OBJECTIFS

En vue d'apporter une solution efficace, flexible et solide à la problématique posée, nous avons défini l'objectif de fournir une application web qui :

- permettra de racourcir le circuit de circulation des informations,
- permettra de faciliter la publication des résultats,
- permettra de faciliter le suivi des données des étudiants,
- peut s'intégrer facilement dans le système d'information de l'université, et aux autres applications existantes telles que moodle, Goole Classroom, etc.

0.7 SUBDIVISION DU TRAVAIL

Ce travail est subdivisé en quatre chapitres, le premier chapitre est consacré aux généralités où nous avons défini quelques concepts clés pour une meilleure compréhension du travail, le deuxième chapitre est consacré à l'état de l'art dans cette partie du travail nous avons fait une étude de l'existant, le troisième chapitre est consacré à la conception où nous présentons les choix techniques et les différents modèles et le quatrième chapitre est consacré à la présentation des résultats.

1

GÉNÉRALITÉS

Dans cette partie du travail, nous allons définir quelques termes utiles qui seront utilisés dans ce document pour une meilleure compréhension de ce dernier.

1.1 LA DÉLIBÉRATION

La délibération est une confrontation de vue visant à trancher un problème ou un choix difficile par l'adoption d'un jugement ou d'une décision réfléchie. Elle peut être effectuée par un individu seul, mais aussi par un groupe d'individus ou une collectivité. Elle débouche en général sur une décision ou un choix.

Compte tenu de cette définition, nous remarquerons qu'à la base il faut donc qu'il y ait un sujet, une question ou un fait sur lequel les avis peuvent divergés et qui nécessite une prise de décision.

1.2 UN LOGICIEL

Un logiciel est un ensemble de programmes informatiques qui permettemt à un ordinateur ou un systeme informatique de réaliser une tâche spécifique, il donne à ce dernier la capacité, il est destiné à resoudre un probleme précis.

Un logiciel peut classé comme système, applicatif, standard, spécifique, ou libre,

1.2.1 *Un logiciel système*

Le logiciel système est un ensemble de programmes informatiques et de bibliothèques logicielles qui fournit un environnement permettant de créer et d'exécuter des logiciels applicatifs. Les fonctionnalités de base d'un ordinateur telles que la manipulation des fichiers et des périphériques sont apportées par le logiciel système. Le logiciel système est lancé avant le logiciel applicatif et joue le rôle d'intermédiaire entre le logiciel applicatif et le matériel de l'ordinateur.

Les logiciels systèmes ont été créés dans le but de mieux adapter les ordinateurs aux besoins des programmeurs de logiciels applicatifs : Ils leur permettent de se concentrer sur les problèmes propres à l'application et faire abstraction des particularités de la machine. Contrairement au logiciel applicatif, le logiciel système est fortement dépendant de la machine. Les logiciels système offrent des services aux logiciels applicatifs et ne sont pas exploités directement par l'usager.

Les systèmes d'exploitation, les pilotes, les langages de programmation, et les utilitaires sont des logiciels système. L'utilisation des langages de programmation est rendue possible par divers programmes tels que le compilateur, l'assembleur, l'éditeur de liens et le chargeur.

1.2.2 *Un logiciel applicatif*

Une application, un applicatif ou encore une appli, une app est, dans le domaine informatique, un programme (ou un ensemble logiciel) directement utilisé pour réaliser une tâche, ou un ensemble de tâches élémentaires d'un même domaine ou formant un tout

Typiquement, un éditeur de texte, un navigateur web, un lecteur multimédia, un jeu vidéo, sont des applications. Les applications s'exécutent en utilisant les services du système d'exploitation pour utiliser les ressources matérielles.

1.2.3 *Un logiciel standard*

Un produit informatique standard ou produit informatique COTS (sigle emprunté à l'expression d'origine anglaise « commercial off-the-shelf » qui signifie : vendu sur étagère) désigne tout produit informatique fabriqué en série et disponible dans le commerce, non réalisé pour un projet en particulier1.

Ces produits informatiques (logiciel ou matériel) sont de plus en plus utilisés dans des projets qui ont pour but de réduire les coûts de conception, de développement et de maintenance.

1.2.4 *Un logiciel spécifique*

Un logiciel spécifique est un logiciel développé sur commande à l'attention d'un client donné, par opposition à un logiciel standard, qui est un développé sur initiative d'un éditeur, et vendu à de nombreux clients.

Le terme anglais correspondant à logiciel spécifique est "custom software", ou " bespoke software". Les Britanniques parlent de bespoke development pour désigner le développement spécifique (développement d'un logiciel spécifique).

La construction d'un logiciel spécifique est une prestation de service, qui consiste à fournir l'expertise technique et la main d'œuvre nécessaire. Les fonctionnalités, le planning de livraison, et les conditions de paiement font l'objet d'un contrat entre le prestataire et le client. Le consommateur est fortement impliqué dans le processus de construction et signe la réussite du travail.

La quasi-totalité des logiciels spécifiques sont des logiciels applicatifs. Les acheteurs de logiciels spécifiques sont des moyennes et grandes entreprises.

La construction de logiciels spécifiques est pratiquée depuis les années 1960, et elle était initialement la seule manière d'obtenir des logiciels applicatifs. En 1998 dans l'Union européenne, 45 % de la production de logiciels concerne des logiciels spécifiques.

1.2.5 *Un logiciel libre*

Un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication par autrui en vue de sa diffusion sont permises, techniquement et juridiquement1, ceci afin de garantir certaines

libertés induites, dont le contrôle du programme par l'utilisateur et la possibilité de partage entre individus.

Ces droits peuvent être simplement disponibles cas du domaine public ou bien établis par une licence, dite « libre », basée sur le droit d'auteur. Les « licences copyleft » garantissent le maintien de ces droits aux utilisateurs même pour les travaux dérivés.

Les logiciels libres constituent une alternative à ceux qui ne le sont pas, qualifiés de « propriétaires » ou de « privateurs ». Ces derniers sont alors considérés par une partie de la communauté du logiciel libre comme étant l'instrument d'un pouvoir injuste, en permettant au développeur de contrôler l'utilisateur.

Le logiciel libre est souvent confondu à tort avec :

— les gratuiciels (freewares) : un gratuiciel est un logiciel gratuit propriétaire, alors qu'un logiciel libre se définit par les libertés accordées à l'utilisateur. Si la nature du logiciel libre facilite et encourage son partage, ce qui tend à le rendre gratuit, elle ne s'oppose pas pour autant à sa rentabilité principalement via des services associés. Les rémunérations sont liées par exemple aux travaux de création, de développement, de mise à disposition et de soutien technique. D'un autre côté les logiciels gratuits ne sont pas nécessairement libres, car leur code source n' est pas systématiquement accessible et leur licence peut ne pas correspondre à la définition du logiciel libre.

— l'open source : le logiciel libre, selon son initiateur, est un mouvement social qui repose sur les principes de Liberté, Égalité, Fraternité ; l'open source quant à lui, décrit pour la première fois dans La Cathédrale et le Bazar, s'attache aux avantages d'une méthode de développement au travers de la réutilisation du code source.

1.2.6 *Un logiciel propriétaire*

Un logiciel propriétaire, logiciel non libre ou parfois logiciel privatif voire logiciel privateur, est un logiciel qui ne permet pas légalement ou techniquement, ou par quelque autre moyen que ce soit, d'exercer simultanément les quatre libertés logicielles que sont l'exécution du logiciel pour tout type d'utilisation, l'étude de son code source (et donc l'accès à ce code source), la distribution de copies, ainsi que la modification du code source.

1.3 LE WEB

Le World Wide Web, littéralement la « toile (d'araignée) mondiale », abrégé www ou le Web, la toile mondiale ou la toile, est un système hypertexte public fonctionnant sur Internet. Le Web permet de consulter, avec un navigateur, des pages accessibles sur des sites. L'image de la toile d'araignée vient des hyperliens qui lient les pages web entre elles.

Le Web est une des applications d'Internet2, distincte d'autres applications comme le courrier électronique, la visioconférence et le partage de fichiers en pair à pair. Inventé en 1989-1990 par Tim Berners-Lee suivi de Robert Cailliau, c'est le Web qui a rendu les médias grand public attentifs à Internet. Depuis, le Web est fréquemment confondu avec Internet ; en particulier, le mot toile est

souvent utilisé dans les textes non techniques sans qu'il soit clair si l'auteur désigne le Web ou Internet.

1.4 UNE API

Une interface de programmation d'application1 ou interface de programmation applicative, souvent désignée par le terme API pour « Application Programming Interface », est un ensemble normalisé de classes, de méthodes, de fonctions et de constantes qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes « consommateurs » peuvent se servir des fonctionnalités du programme « fournisseur ».

On parle d'API à partir du moment où une entité informatique cherche à agir avec ou sur un système tiers et que cette interaction se fait de manière normalisée en respectant les contraintes d'accès définies par le système tiers. On dit alors que le système tiers « expose une API ».

À ce titre, des interactions aussi diverses que la signature d'une fonction, une URL ou un RPC par exemple sont parfois considérés comme des API (ou micro-API) à part entière.

Dans l'industrie contemporaine du logiciel, les applications informatiques se servent de nombreuses interfaces de programmation, car la programmation réutilise des briques de fonctionnalités fournies par des logiciels tiers. Cette construction par assemblage nécessite pour le programmeur de connaître la manière d'interagir avec les autres logiciels qui dépend de leur interface de programmation. Le programmeur n'a pas besoin de connaître les détails de la logique interne du logiciel tiers, et celle-ci n'est pas nécessairement documentée par le fournisseur. Seule l'API est réellement nécessaire pour utiliser le système tiers en question.

Des logiciels tels que les systèmes d'exploitation, les systèmes de gestion de base de données, les langages de programmation ou les serveurs d'applications comportent une ou plusieurs interfaces de programmation.

1.5 LES MÉTHODES DE DÉVELOPPEMENT LOGICIEL

Dans le domaine du génie logiciel, un processus de développement logiciel est un processus de planification et de gestion du développement logiciel. Il consiste généralement à diviser le travail de développement de logiciels en étapes ou sous-processus plus petits, parallèles ou séquentiels, afin d'améliorer la conception et/ou la gestion du produit. Il est également connu sous le nom de cycle de vie du développement logiciel (SDLC). La méthodologie peut inclure la prédéfinition de livrables et d'artefacts spécifiques qui sont créés et complétés par une équipe de projet pour développer ou maintenir une application.

La plupart des processus de développement modernes peuvent être vaguement décrits comme agiles.

D'autres méthodologies incluent la méthode en cascade, le prototypage, le développement itératif et incrémental, le développement en spirale, le développement rapide d'applications et la programmation extrême.

Un "modèle" de cycle de vie est parfois considéré comme un terme plus général pour une catégorie de méthodologies et un "processus" de développement de logiciels comme un terme plus spécifique pour se référer à un processus spécifique choisi par une organisation spécifique. Par exemple, il existe de nombreux processus de développement de logiciels spécifiques qui correspondent au modèle de cycle de vie en spirale. Ce domaine est souvent considéré comme un sous-ensemble du cycle de vie du développement des systèmes.

1.5.1 *Les méthodes agile*

Dans le développement de logiciels, les pratiques agiles (parfois écrites "Agile") comprennent la découverte des besoins et l'amélioration des solutions grâce à l'effort de collaboration d'équipes auto-organisées et interfonctionnelles avec leur(s) client(s)/utilisateur(s) finaux, Popularisées dans le Manifeste pour le développement agile de logiciels de 2001, ces valeurs et principes ont été dérivés et sous-tendent un large éventail de cadres de développement de logiciels, y compris Scrum et Kanban.

Les méthodes agiles de développement de logiciels couvrent un large éventail du cycle de vie du développement de logiciels. Certaines méthodes se concentrent sur les pratiques (par exemple, XP, programmation pragmatique, modélisation agile), tandis que d'autres se concentrent sur la gestion du flux de travail (par exemple, Scrum, Kanban). Certains soutiennent les activités de spécification et de développement des exigences (par exemple, FDD), tandis que d'autres cherchent à couvrir l'ensemble du cycle de développement (par exemple, DSDM, RUP).

Les méthodes de développement de logiciels agiles les plus connus sont les suivantes :

1. Le développement adaptatif de logiciels (DAL) est un processus de développement de logiciels issu des travaux de Jim Highsmith et Sam Bayer sur le développement rapide d'applications (RAD). Il incarne le principe selon lequel l'adaptation continue du processus au travail en cours est l'état normal des choses. Le développement adaptatif de logiciels remplace le cycle traditionnel en cascade par une série répétée de cycles de spéculation, de collaboration et d'apprentissage. Ce cycle dynamique permet un apprentissage et une adaptation continus à l'état émergent du projet. Les caractéristiques d'un cycle de vie ASD sont qu'il est axé sur la mission, basé sur les fonctionnalités, itératif, limité dans le temps, axé sur le risque et tolérant au changement. Comme pour le RAD, le DSA est également un antécédent du développement agile de logiciels.
2. Le processus unifié agile (AUP) est une version simplifiée du processus unifié rationnel (RUP) développé par Scott Ambler. Il décrit une approche simple et facile à comprendre pour développer des logiciels d'application commerciale en utilisant des techniques et des concepts agiles tout en restant fidèle au RUP. L'AUP applique des techniques agiles, notamment le développement piloté par les tests (TDD), la modélisation agile (AM), la gestion agile du changement et le remaniement de la base de données, afin d'améliorer la productivité.
3. Le développement rapide d'applications (RAD), est à la fois un terme général pour les approches de développement de logiciels adaptatifs et le nom de la méthode de développement rapide de James Martin. En général, les approches RAD du développement de logiciels mettent moins l'accent sur la planification et davantage sur un processus adaptatif. Les prototypes sont souvent

utilisés en plus ou parfois même à la place des spécifications de conception. L'approche RAD est particulièrement bien adaptée (mais pas uniquement) au développement de logiciels basés sur les besoins de l'interface utilisateur. Les concepteurs d'interfaces utilisateur graphiques sont souvent appelés outils de développement rapide d'applications. D'autres approches du développement rapide comprennent les modèles adaptatif, agile, en spirale et unifié.

4. Scrum est un modèle léger qui aide les personnes, les équipes et les organisations à générer de la valeur par le biais de solutions adaptatives à des problèmes complexes. Il est couramment utilisé dans le développement de logiciels, mais aussi dans d'autres domaines tels que la recherche, les ventes, le marketing, l'éducation et les technologies de pointe. Il est conçu pour des équipes de dix membres ou moins qui divisent leur travail en objectifs à atteindre au cours d'itérations délimitées dans le temps, appelées sprints. Chaque sprint ne dure pas plus d'un mois et dure le plus souvent deux semaines. L'équipe scrum évalue les progrès réalisés lors de réunions quotidiennes limitées dans le temps, d'une durée maximale de 15 minutes, appelées scrums quotidiens (réunion debout). À la fin du sprint, l'équipe tient deux autres réunions : une revue de sprint destinée à présenter le travail effectué aux parties prenantes et à solliciter des commentaires, et une rétrospective de sprint destinée à permettre à l'équipe de réfléchir et de s'améliorer.
5. La programmation extrême (XP) est une méthodologie de développement de logiciels visant à améliorer la qualité des logiciels et la réactivité à l'évolution des besoins des clients. En tant que type de développement agile de logiciels, elle préconise des versions fréquentes dans des cycles de développement courts, afin d'améliorer la productivité et d'introduire des points de contrôle permettant d'adopter les nouvelles exigences des clients.

   Parmi les autres éléments de la programmation extrême, citons : la programmation en binôme ou l'examen approfondi du code, les tests unitaires de l'ensemble du code, la programmation de fonctionnalités uniquement lorsqu'elles sont réellement nécessaires, une structure de gestion horizontale, la simplicité et la clarté du code, l'attente de changements dans les exigences du client à mesure que le temps passe et que le problème est mieux compris, ainsi qu'une communication fréquente avec le client et entre les programmeurs. La méthodologie tire son nom de l'idée que les éléments bénéfiques des pratiques traditionnelles d'ingénierie logicielle sont portés à des niveaux "extrêmes". Par exemple, les révisions de code sont considérées comme une pratique bénéfique ; poussées à l'extrême, les révisions de code peuvent être continues (c'est-à-dire la pratique de la programmation en binôme).

1.5.2 *Le prototypage*

Le prototypage de logiciels est l'activité qui consiste à créer des prototypes d'applications logicielles, c'est-à-dire des versions incomplètes du programme logiciel en cours de développement. Il s'agit d'une activité qui peut se dérouler dans le cadre du développement de logiciels et qui est comparable au prototypage tel qu'il est connu dans d'autres domaines, tels que l'ingénierie mécanique ou la fabrication.

Un prototype ne simule généralement que quelques aspects du produit final et peut être complètement différent de celui-ci.

Le prototypage présente plusieurs avantages : le concepteur et le réalisateur du logiciel peuvent obtenir un retour d'information précieux de la part des utilisateurs dès le début du projet. Le client et l'entrepreneur peuvent comparer si le logiciel réalisé correspond à la spécification du logiciel, selon laquelle le programme logiciel est construit. Cela permet également à l'ingénieur logiciel de se faire une idée de l'exactitude des estimations initiales du projet et de savoir si les délais et les étapes proposés peuvent être respectés. Le degré d'exhaustivité et les techniques utilisées dans le prototypage ont fait l'objet de développements et de débats depuis leur proposition au début des années 1970.

1.5.3 *Le développement itératif et incrémental*

Le développement itératif et incrémental est une combinaison de la conception itérative ou de la méthode itérative et du modèle de construction incrémental pour le développement.

L'utilisation de ce terme a débuté dans le domaine du développement de logiciels, une combinaison de longue date des deux termes itératif et incrémental ayant été largement suggérée pour les efforts de développement de grande envergure. Par exemple, le document DOD-STD-2167 de 1985 mentionne : "Pendant le développement d'un logiciel, plus d'une itération du cycle de développement du logiciel peut être en cours en même temps" et "Ce processus peut être décrit comme une 'acquisition évolutive' ou une approche de 'construction incrémentale'". Dans le domaine des logiciels, la relation entre les itérations et les incréments est déterminée par le processus global de développement du logiciel.

1.5.4 *Le développement en spirale*

Le modèle en spirale est un modèle de processus de développement de logiciels axé sur le risque. Basé sur les risques propres à un projet donné, le modèle en spirale guide l'équipe vers l'adoption d'éléments d'un ou de plusieurs modèles de processus, tels que le prototypage incrémentiel, en cascade ou évolutif.

1.5.5 *La méthode en V*

Le cycle en V (« V model » ou « Vee model » en anglais) est un modèle d'organisation des activités de développement d'un produit qui se caractérise par un flux d'activité descendant qui détaille le produit jusqu'à sa réalisation, et un flux ascendant, qui assemble le produit en vérifiant sa qualité. Ce modèle est issu du modèle en cascade dont il reprend l'approche séquentielle et linéaire de phases.

Il l'enrichit cependant d'activités d'intégration de système à partir de composants plus élémentaires, et il met en regard chaque phase de production successive avec sa phase de validation correspondante, lui conférant ainsi la forme d'un V1.

Issu de l'ingénierie système, le cycle en V est souvent considéré comme un cycle de projet, alors qu'ingénierie système et gestion de projet sont complémentaires. L'ingénierie système va se focaliser

sur le développement du produit, alors que la gestion de projet va se concentrer sur l'atteinte des bénéfices attendus par le client ou l'utilisateur. Le cycle en V n'est donc pas un cycle de projet.

1.5.6 *Le développement en cascade*

Le modèle en cascade, ou « waterfall » en anglais, est une organisation des activités d'un projet sous forme de phases linéaires et séquentielles, où chaque phase correspond à une spécialisation des tâches et dépend des résultats de la phase précédente. Il comprend les phases d'exigences, de conception, de mise en œuvre et de mise en service.

Le modèle en cascade est un cycle de vie de projet issu des industries manufacturières et du secteur de la construction, où une conception préalable est nécessaire, compte tenu des fortes contraintes matérielles et des coûts élevés afférents aux changements de la conception en cours de réalisation. Il est utilisé notamment dans les domaines de l'ingénierie et du développement de logiciels.

1.6 GIT

Git est un logiciel de gestion de versions de code source décentralisé. C'est un logiciel libre et gratuit, créé en 2005 par Linus Torvalds, auteur du noyau Linux, et distribué selon les termes de la licence publique générale GNU version 2. Le principal contributeur actuel de Git, et ce depuis plus de 16 ans, est Junio C Hamano.

Depuis les années 2010, il s'agit du logiciel de gestion de versions le plus populaire dans le développement logiciel et web, qui est utilisé par des dizaines de millions de personnes, sur tous les environnements (Windows, Mac, Linux).

Git est aussi le système à la base du célèbre site web GitHub, le plus important hébergeur de code informatique.

1.7 GITHUB

GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion de versions Git. Ce site est développé en Ruby on Rails et Erlang par Chris Wanstrath, PJ Hyett et Tom Preston-Werner. GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres.

Le site assure également un contrôle d'accès et des fonctionnalités destinées à la collaboration comme le suivi des bugs, les demandes de fonctionnalités, la gestion de tâches et un wiki pour chaque projet. Le site est devenu le plus important dépôt de code au monde, utilisé comme dépôt public de projets libres ou dépôt privé d'entreprises.

En 2018, GitHub est acquis par Microsoft pour 7,5 milliards de dollars.

1.8 LINUX

Linux ou GNU/Linux est une famille de systèmes d'exploitation open source de type Unix fondés sur le noyau Linux créé en 1991 par Linus Torvalds. De nombreuses distributions Linux ont depuis vu le jour et constituent un important vecteur de popularisation du mouvement du logiciel libre.

Si, à l'origine, Linux a été développé pour les ordinateurs compatibles PC, il n'a jamais équipé qu'une très faible part des ordinateurs personnels. Mais le noyau Linux, accompagné ou non des logiciels GNU, est également utilisé par d'autres types de systèmes informatiques, notamment les serveurs, téléphones portables, systèmes embarqués ou encore superordinateurs. Le système d'exploitation pour téléphones portables Android qui utilise le noyau Linux mais pas GNU équipe aujourd'hui 85 % des tablettes tactiles et smartphones.

1.9 LA MODÉLISATION

La modélisation est le processus de représentation d'un système, d'un processus, d'un objet ou d'un concept sous forme de modèle.

Un modèle est une représentation abstraite et simplifiée d'un système réel, qui peut être utilisée pour comprendre, analyser, prédire et concevoir le système.

1.10 UML (UNIFIED MODELING LANGUAGE)

Le Langage de Modélisation Unifié, de l'anglais Unified Modeling Language (UML), est un langage de modélisation graphique à base de pictogrammes conçu comme une méthode normalisée de visualisation dans les domaines du développement logiciel et en conception orientée objet.

L'UML est une synthèse de langages de modélisation objet antérieurs : Booch, OMT, OOSE. Principalement issu des travaux d e Grady Booch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group (OMG). UML 1.0 a été normalisé en janvier 1997 ; UML 2.0 a été adopté par l'OMG en juillet 2005. La dernière version de la spécification validée par l'OMG est UML 2.5.1 (2017).

Voici les principales notations graphiques utilisées dans UML :

1.10.1 *le diagramme de classes*

Le diagramme de classes est un schéma utilisé en génie logiciel pour présenter les classes et les interfaces des systèmes ainsi que leurs relations. Ce diagramme fait partie de la partie statique d'UML, ne s'intéressant pas aux aspects temporels et dynamiques.

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent

de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être reliées grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, représentées par un arc spécifique dans le diagramme de classes.

Elles sont finalement instanciées pour créer des objets (une classe est un moule à objet : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).

1.10.2 *Le diagramme de cas d'utilisation*

Les diagrammes de cas d'utilisation (DCU) sont des diagrammes UML utilisés pour une représentation du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet. En effet, un cas d'utilisation (use cases) représente une unité discrète d'interaction entre un utilisateur (humain ou machine) et un système. Ainsi, dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), et ils apparaissent dans les cas d'utilisation.

1.10.3 *le diagramme d'objets*

Le diagramme d'objets, dans le langage de modélisation de donnée UML, permet de représenter les instances des classes, c'est-à-dire des objets. Comme le diagramme de classes, il exprime les relations qui existent entre les objets, mais aussi l'état des objets, ce qui permet d'exprimer des contextes d'exécution. En ce sens, ce diagramme est moins général que le diagramme de classes.

Les diagrammes d'objets s'utilisent pour montrer l'état des instances d'objet avant et après une interaction, autrement dit c'est une photographie à un instant précis des attributs et objet existant. Il est utilisé en phase exploratoire.

1.10.4 *le diagramme de séquence*

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language.

1.10.5 *le diagramme d'activités*

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables (multi-threads ou multi-processus). Le diagramme d'activité est également utilisé pour décrire un flux de travail (workflow).

1.10.6 *le diagramme de déploiement*

En UML, un diagramme de déploiement est une vue statique qui sert à représenter l'utilisation de l'infrastructure physique par le système et la manière dont les composants du système sont répartis ainsi que leurs relations entre eux. Les éléments utilisés par un diagramme de déploiement sont principalement les noeuds, les composants, les associations et les artefacts. Les caractéristiques des ressources matérielles physiques et des supports de communication peuvent être précisées par stéréotype.

1.11 L'HTML (HYPERTEXT MARKUP LANGUAGE)

Le HyperText Markup Language, généralement abrégé HTML ou, dans sa dernière version, HTML5, est le langage de balisage conçu pour représenter les pages web.

Ce langage permet d'écrire de l'hypertexte (d'où son nom), de structurer sémantiquement une page web, de mettre en forme du contenu, de créer des formulaires de saisie ou encore d'inclure des ressources multimédias dont des images, des vidéos, et des programmes informatiques. L'HTML offre également la possibilité de créer des documents interopérables avec des équipements très variés et conformément aux exigences de l'accessibilité du web.

Il est souvent utilisé conjointement avec le langage de programmation JavaScript et des feuilles de style en cascade (CSS). HTML est inspiré du Standard Generalized Markup Language (SGML). Il s'agit d'un format ouvert.

1.12 CSS (CASCADING STYLE SHEETS)

Les feuilles de style en cascade, généralement appelées CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

Le css sert à donner vie à une page web, il permet de mettre en forme le contenu HTML d'une page web. Il permet de définir des styles pour chaque élément HTML et de les appliquer à l'ensemble des pages d'un site.

1.13 UN LANGUAGE DE PROGRAMMATION

Un langage de programmation est un langage informatique destiné à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire, de significations, mais aussi d'un environnement de traduction censé rendre sa syntaxe compréhensible par la machine.

Les langages de programmation permettent de décrire d'une part les structures des données qui seront manipulées par l'appareil informatique, et d'autre part d'indiquer comment sont effectuées les manipulations, selon quels algorithmes. Ils servent de moyens de communication par lesquels le programmeur communique avec l'ordinateur, mais aussi avec d'autres programmeurs ; les programmes étant d'ordinaire écrits, lus, compris et modifiés par une équipe de programmeurs.

Les langages de programmation peuvent être classés en plusieurs catégories, telles que :

1. Les langages de programmation impératifs : qui décrivent les étapes à suivre pour accomplir une tâche. Les langages de programmation impératifs peuvent être classés en deux catégories : les langages de programmation procéduraux et les langages de programmation orientés objet.
2. Les langages de programmation fonctionnels : qui se concentrent sur les fonctions mathématiques et l'évaluation d'expressions, etc.
3. Les langages de programmation orientés objet : qui sont basés sur des objets et leurs interactions, etc. Les langages de programmation orientés objet peuvent être classés en deux catégories : les langages de programmation orientés objet basés sur les classes et les langages de programmation orientés objet basés sur les prototypes.
4. Les langages de script : qui sont utilisés pour automatiser des tâches et des processus, etc. Les langages de script sont généralement interprétés plutôt que compilés.
5. Les langages de programmation déclaratifs : qui décrivent le résultat souhaité plutôt que les étapes à suivre pour l'obtenir.

Il existe de nombreux langages de programmation différents, chacun ayant ses propres avantages et inconvénients. Voici quelques-uns des langages de programmation les plus courants :

— C++,
— JavaScript,
— Java,
— CSharp,
— Ruby,
— Python,
— Etc.

1.14 UN FRAMEWORK

En programmation informatique, un framework (appelé aussi infrastructure logicielle, infrastructure de développement, environnement de développement, socle d'applications, cadre d'applications ou cadriciel) est un ensemble cohérent de composants logiciels structurels qui sert à créer les fondations ainsi que les grandes lignes de tout ou partie d'un logiciel, c'est-à-dire une architecture.

Un framework se distingue d'une simple bibliothèque logicielle principalement, d'une part par son caractère générique, faiblement spécialisé, contrairement à certaines bibliothèques ; un framework peut à ce titre être constitué de plusieurs bibliothèques, chacune spécialisée dans un domaine.

Un framework peut néanmoins être spécialisé dans un langage particulier, une plateforme spécifique, un domaine particulier : communication de données, data mapping, etc.. D'autre part, il impose

un cadre de travail, dû à sa construction même, guidant l'architecture logicielle voire conduisant le développeur à respecter certains patrons de conception ; les bibliothèques le constituant sont alors organisées selon le même paradigme.

Les frameworks sont donc conçus et utilisés pour modeler l'architecture des logiciels applicatifs, des applications web, des middlewares et des composants logiciels. Les frameworks sont acquis par les informaticiens, puis incorporés dans des logiciels applicatifs mis sur le marché, ils sont par conséquent rarement achetés et installés séparément par un utilisateur final.

1.15 NODEJS

Node.js est une plateforme logicielle libre en JavaScript, orientée vers les applications réseau évènementielles hautement concurrentes qui doivent pouvoir monter en charge.

Elle utilise la machine virtuelle V8, la bibliothèque libuv pour sa boucle d'évènements, et implémente sous licence MIT les spécifications CommonJS.

Parmi les modules natifs de Node.js, on retrouve http qui permet le développement de serveur HTTP. Ce qui autorise, lors du déploiement de sites internet et d'applications web développés avec Node.js, de ne pas installer et utiliser des serveurs webs tels que Nginx ou Apache.

Concrètement, Node.js est un environnement bas niveau permettant l'exécution de JavaScript côté serveur.

Node.js est utilisé notamment comme plateforme de serveur Web, elle est utilisée par GoDaddy, IBM, Netflix, Amazon Web Services, Groupon, Vivaldi, SAP, LinkedIn, Microsoft, Yahoo !, Walmart, Rakuten, Sage et PayPal.

1.16 UNE BASE DES DONNÉES

Une base de données permet de stocker et de retrouver des données structurées, semi-structurées ou des données brutes ou de l'information, souvent en rapport avec un thème ou une activité ; celles-ci peuvent être de natures différentes et plus ou moins reliées entre elles.

Leurs données peuvent être stockées sous une forme très structurée (base de données relationnelles par exemple), ou bien sous la forme de données brutes peu structurées (avec les bases de données NoSQL par exemple). Une base de données peut être localisée dans un même lieu et sur un même support informatisé, ou répartie sur plusieurs machines à plusieurs endroits.

La base de données est au centre des dispositifs informatiques de collecte, mise en forme, stockage et utilisation d'informations. Le dispositif comporte un système de gestion de base de données (abréviation : SGBD) : un logiciel moteur qui manipule la base de données et dirige l'accès à son contenu. De tels dispositifs comportent également des logiciels applicatifs, et un ensemble de règles relatives à l'accès et l'utilisation des informations.

Lorsque plusieurs objets nommés « bases de données » sont constitués sous forme de collection, on parle alors d'une banque de données.

Il existe plusieurs types de bases de données comme les bases des données en mémoire, les plus courantes sont les bases de données relationnelles et les bases de données NoSQL.

1.16.1 *Les bases de données relationnelles*

Une base de données relationnelle est une base de données où l'information est organisée dans des tableaux à deux dimensions appelés des relations ou tables, selon le modèle introduit par Edgar F. Codd en 1960. Selon ce modèle relationnel, une base de données consiste en une ou plusieurs relations. Les lignes de ces relations sont appelées des nuplets ou enregistrements. Les colonnes sont appelées des attributs.

Les logiciels qui permettent de créer, utiliser et maintenir des bases de données relationnelles sont des systèmes de gestion de bases de données relationnelles (SGBDR).

Pratiquement tous les systèmes relationnels utilisent le langage SQL pour interroger les bases de données. Ce langage permet de demander des opérations d'algèbre relationnelle telles que l'intersection, la sélection et la jointure.

1.16.2 *Les bases de données NoSQL*

NoSQL désigne une famille de systèmes de gestion de base de données (SGBD) qui s'écarte du paradigme classique des bases relationnelles. L'explicitation la plus populaire de l'acronyme est Not only SQL (« pas seulement SQL » en anglais) même si cette interprétation peut être discutée.

La définition exacte de la famille des SGBD NoSQL reste sujette à débat. Le terme se rattache autant à des caractéristiques techniques qu'à une génération historique de SGBD qui a émergé autour des années 20102. D'après Pramod J. Sadalage et Martin Fowler, la raison principale de l'émergence et de l'adoption des SGBD NoSQL serait le développement des centres de données et la nécessité de posséder un paradigme de bases de données adapté à ce modèle d'infrastructure matérielle.

L'architecture machine en clusters induit une structure logicielle distribuée fonctionnant avec des agrégats répartis sur différents serveurs permettant des accès et modifications concurrentes mais imposant également de remettre en cause de nombreux fondements de l'architecture SGBD relationnelle traditionnelle, notamment les propriétés ACID.

2

ÉTAT DE L'ART

Avant de commencer à développer notre application, nous avons d'abord comparer les différentes applications existantes pour en dégager les fonctionnalités clés, nous avons ensuite mis sur un balance les avantages et les inconvénients de chacune d'elles pour enfin proposer une solution qui répondrait aux besoins de l'Université Nouveaux Horizons.

2.1 PRÉSENTATION DES PRINCIPALES APPLICATIONS EXISTANTES

Nous sommes conscients qu'il en existe certainement lusieurs autres, mais nous avons choisi de nous limiter à celles mentioner ci dessous car elles sont les plus utilisées, les plus populaires et couvrent la plupart des besoins.

2.1.1 *Moodle*

1. **Présentation de l'application**
   Moodle est un système de gestion de l'apprentissage (LMS) open source, c'est-à-dire un logiciel permettant la création, la gestion, la distribution et la surveillance de cours en ligne. Il permet aux enseignants et aux formateurs de créer des cours en ligne interactifs, d'organiser des activités d'apprentissage, de communiquer avec les étudiants et de suivre leur progression.
2. **Fonctionalités clés**
   - Gestion des cours : Moodle permet de créer des cours en ligne et d'organiser des activités d'apprentissage en utilisant des outils tels que des forums de discussion, des leçons, des wikis, des glossaires, des quiz, des devoirs, des sondages, etc.
   - Gestion des utilisateurs : Moodle permet de gérer les utilisateurs, de les inscrire aux cours, de les suivre et de leur attribuer des rôles spécifiques (enseignant, étudiant, administrateur, etc.).
   - Communication : Moodle offre plusieurs outils de communication, y compris les forums de discussion, les messages privés, les chats en temps réel et les webinaires.
   - Suivi des progrès : Moodle permet aux enseignants de suivre les progrès des étudiants en visualisant leurs notes, leurs activités et leurs contributions sur les forums de discussion et autres outils.
   - Accessibilité : Moodle est conforme aux normes d'accessibilité pour les personnes handicapées, ce qui permet à tous les utilisateurs de profiter pleinement des activités d'apprentissage en ligne.
3. **Inconvénients et limites**
   - Expertise technique requise : Moodle est un logiciel complexe dont l'installation et la configuration requièrent une certaine expertise technique.

— Sécurité : Moodle est une cible populaire pour les pirates informatiques, il est donc important de prendre des mesures pour sécuriser votre site Moodle. Cela inclut l'utilisation de mots de passe forts, l'installation de mises à jour de sécurité, et la surveillance du site pour toute activité suspecte.
— Evolutivité : Moodle peut être adapté à un grand nombre d'utilisateurs, mais il peut être difficile de gérer un grand site Moodle.
— Moodle est conçu pour répondre aux besoins généraux de l'enseignement en ligne. Il n'est pas conçu pour répondre aux besoins spécifiques d'une institution.

2.1.2 *PowerSchool*

1. **Présentation de l'application**
   PowerSchool est un système d'information pour les établissements scolaires qui permet de gérer les informations relatives aux étudiants, aux enseignants et aux programmes académiques. Il est utilisé par plus de 45 millions d'utilisateurs dans le monde entier, notamment dans les écoles primaires, secondaires et les établissements d'enseignement supérieur.
2. **Fonctionalités clés**
   — Gestion des inscriptions et des admissions : PowerSchool permet de gérer les inscriptions et les admissions des étudiants, y compris la collecte des informations de base, la gestion des documents requis, la communication avec les parents et les étudiants, et la planification des cours.
   — Gestion des notes et des absences : PowerSchool permet aux enseignants d'enregistrer les notes et les absences des étudiants, et de partager les informations avec les parents et les étudiants.
   — Gestion des emplois du temps et des calendriers académiques : PowerSchool permet de gérer les emplois du temps des enseignants et des étudiants, ainsi que les calendriers académiques et les événements scolaires.
   — Gestion des frais de scolarité et des paiements : PowerSchool permet de gérer les frais de scolarité et les paiements, y compris la collecte et le suivi des paiements, la génération de factures, et la communication avec les parents et les étudiants.
   — Communication avec les parents et les étudiants : PowerSchool permet de communiquer avec les parents et les étudiants via des messages personnalisés, des bulletins électroniques, des alertes automatiques, et d'autres outils de communication.
   — Gestion des bibliothèques : PowerSchool permet de gérer les bibliothèques de l'établissement, y compris la gestion des prêts de livres, la gestion des retours, et la gestion des réservations de livres.
   — Rapports et analyses : PowerSchool permet de générer des rapports et des analyses sur les performances académiques des étudiants, les tendances de fréquentation, les statistiques de paiement, et bien plus encore.

3. **Inconvénients et limites**
   — Coût : Le coût initial de mise en place de PowerSchool peut être élevé, en particulier pour les établissements scolaires plus petits. Les coûts peuvent également augmenter avec l'ajout de fonctionnalités supplémentaires.
   — Complexité : PowerSchool est un système complexe qui peut nécessiter une formation pour les administrateurs, les enseignants et les utilisateurs finaux. Les établissements scolaires doivent également consacrer du temps et des ressources à la configuration et à la gestion du système.
   — Personnalisation : Bien que PowerSchool offre de nombreuses fonctionnalités, il peut être difficile de personnaliser le système pour répondre aux besoins spécifiques de chaque établissement. Les établissements peuvent être limités dans leur capacité à personnaliser le système en fonction de leurs besoins spécifiques.
   — Dépendance : Les établissements qui utilisent PowerSchool dépendent du système pour gérer les données des étudiants et de l'établissement. En cas de panne du système ou de perte de données, cela peut avoir des conséquences importantes pour l'établissement.

Apres la présentation de ces deux applications, nous nous intéréssé à la manière dont les universités soeurs de l'Université Nouveaux Horizons gèrent leurs données et leurs processus académiques, l'une d'elle s'est démarquée par son approche innovante.

2.1.3 *Esis Salama*

1. **Présentation**
   Esis salama est une institution d'enseignement supérieure, ce qui lui vaut une mention dans ce présent travail c'est la manière dont elle gère la délibération.
2. **Outils utilisés**
   — Excel : Ce dernier est un tableur édité par Microsoft pour les systèmes d'exploitation Windows et Mac OS X. Il est principalement utilisé pour le calcul, l'analyse de données et les graphiques. Excel est l'un des programmes les plus populaires pour la gestion des données et des calculs.
   — Plateforme développée en interne : Cette dernière facilite la publication des résultats des étudiants et permet aux étudiants de consulter leurs résultats et de faire des recours en ligne.
3. **Inconvénients et limites**
   — Cette façon de faire est très chronophage,
   — Nécessite beaucoup de ressources humaines

2.2 CONCLUSION

En conclusion nous retiendront que l'idéal serait de produire une application qui comblerait les lacunes des applications existantes et les étendrait mais l'idéal n'est pas toujours réalisable. Nous proposons donc une solution qui resoud les gros inconvenients et limitations des applications existantes et qui est facilement extensible et modifiable.

Les apports sont :

— Le fait que cette application soit développée en interne permettra de l'adapter aux besoins de l'Université Nouveaux Horizons et de rester indépendant des éditeurs de logiciels.
— L'application sera facilement extensible et modifiable.
— L'application peut facilement s'intégrer au système existant.
— L'application permettra de réduire les coûts en temps et ressources humaines car elle fera la grande partie du travail certes sous la supervision humaine.

3

CONCEPTION

Dans ce chapitre nous allons faire le tour des différentes décisions techniques prises, et outils utilisés.

Nous allons commencer par présenter les technologies utilisées, puis nous allons parler du choix architectural de l'application, et enfin nous allons parler des différents modèles utilisés.

3.1 CHOIX TECHNIQUES ET MOTIVATIONS

3.1.1 *MySQL*

Le choix de la base de données est une étape importante dans le développement d'une application.

Il existe plusieurs types de bases de données, chacune avec ses propres avantages et inconvénients.

Dans ce projet, nous avons choisi d'utiliser une base de données relationnelle, car elle est la plus adaptée à notre cas d'utilisation.

Elle nous permet de stocker les données de manière structurée, et de les manipuler facilement à l'aide de requêtes SQL. Nous avons choisi d'utiliser MysQL, car c'est une base de données relationnelle populaire, avec une grande communauté de développeurs et une documentation complète.

3.1.2 *Architecture de l'application*

Nous avons choisi une architecture client-serveur pour notre application. Le serveur est responsable de la logique métier et de la gestion des données, et le client est responsable de l'interface utilisateur. Le client communique avec le serveur via une API REST (Representational State Transfer), qui est un ensemble de conventions et de bonnes pratiques pour la conception d'API web. L'API REST permet au client d'effectuer des opérations CRUD (Create, Read, Update, Delete) sur les données stockées sur le serveur. Nous avons choisi d'utiliser une API REST, car elle est simple à mettre en œuvre et facile à utiliser. Elle permet également de séparer la logique métier de l'interface utilisateur, ce qui facilite la maintenance et l'évolutivité de l'application. Nous avons choisi d'utiliser JSON (JavaScript Object Notation) comme format de données pour l'API REST, car il est léger, facile à lire et à écrire.

3.1.3 *Le processus unifié*

Nous avons choisi d'utiliser le processus unifié pour le développement de notre application. Le processus unifié est un processus de développement de logiciels itératif et incrémental de la famille des méthodes agiles.

le processus unifié utilise des modèles UML (Unified Modeling Language) pour la conception et la modélisation.

3.1.4 *Typescript*

Nous avons choisi d'utiliser Typescript pour le développement de notre application. Typescript est un langage de programmation open source développé par Microsoft. Il est conçu pour le développement d'applications JavaScript à grande échelle. Il ajoute des fonctionnalités au JavaScript, comme le typage statique, les classes, les interfaces, les modules, etc. Il est compilé en JavaScript, et peut donc être utilisé sur n'importe quel navigateur web ou serveur web.

Le plus gros avantages de Typescript est qu'il est facile à apprendre et à utiliser, et il nous permet de détecter les erreurs de programmation avant l'exécution du code, de plus il est compilé en JavaScript, et peut donc être utilisé sur n'importe quel navigateur web ou serveur web ce qui nous permet d'avoir un code client et serveur en Typescript on ne change donc pas de language entre le client et le serveur.

3.1.5 *NestJS*

Nous avons choisi d'utiliser NestJS pour le développement l'API(partie serveur). NestJS est un framework open source pour le développement d'applications Node.js. Il est basé sur Express, et utilise Typescript. Il est conçu pour créer des applications évolutives et efficaces. Il utilise une architecture modulaire, et est composé de plusieurs modules, chacun avec sa propre fonctionnalité.

3.1.6 *NextJs*

Nous avons choisi d'utiliser NextJS pour le développement de notre interface utilisateur (partie client). web. NextJS est un framework open source pour le développement d'applications React. Il est basé sur React, et utilise Typescript.

3.1.7 *Git*

Nous avons choisi d'utiliser Git pour la gestion de version de notre code. Git est un système de contrôle de version distribué open source. Il est conçu pour gérer les petits et grands projets avec rapidité et efficacité. Il est utilisé pour suivre les modifications apportées au code source, et pour faciliter la collaboration entre les développeurs. Nous avons choisi d'utiliser Git, car il est facile à apprendre et à utiliser, et il nous permet de suivre les modifications apportées au code source, et de revenir à une version antérieure du code source si nécessaire.

3.1.8 *GitHub*

Nous avons choisi d'utiliser GitHub pour l'hébergement de notre code. GitHub est un service d'hébergement de code source basé sur Git. Il est conçu pour faciliter la collaboration entre les développeurs. Il est utilisé pour héberger des projets open source et privés.

Nous avons choisi d'utiliser car ce projet est open source et nous voulons que ceux qui souhaitent contribuer au projet puissent le faire facilement.

3.1.9 *Tailwindcss*

Nous avons choisi d'utiliser Tailwindcss pour le développement de notre interface utilisateur (partie client). web. Tailwindcss est un framework css pronant l'approche utility-first.

Conclusion

Nous retiendrons par ailleurs que toutes les décisions prises ne sont pas forcément les meilleures, mais elles ont été prises en fonction des contraintes et des objectifs du projet.

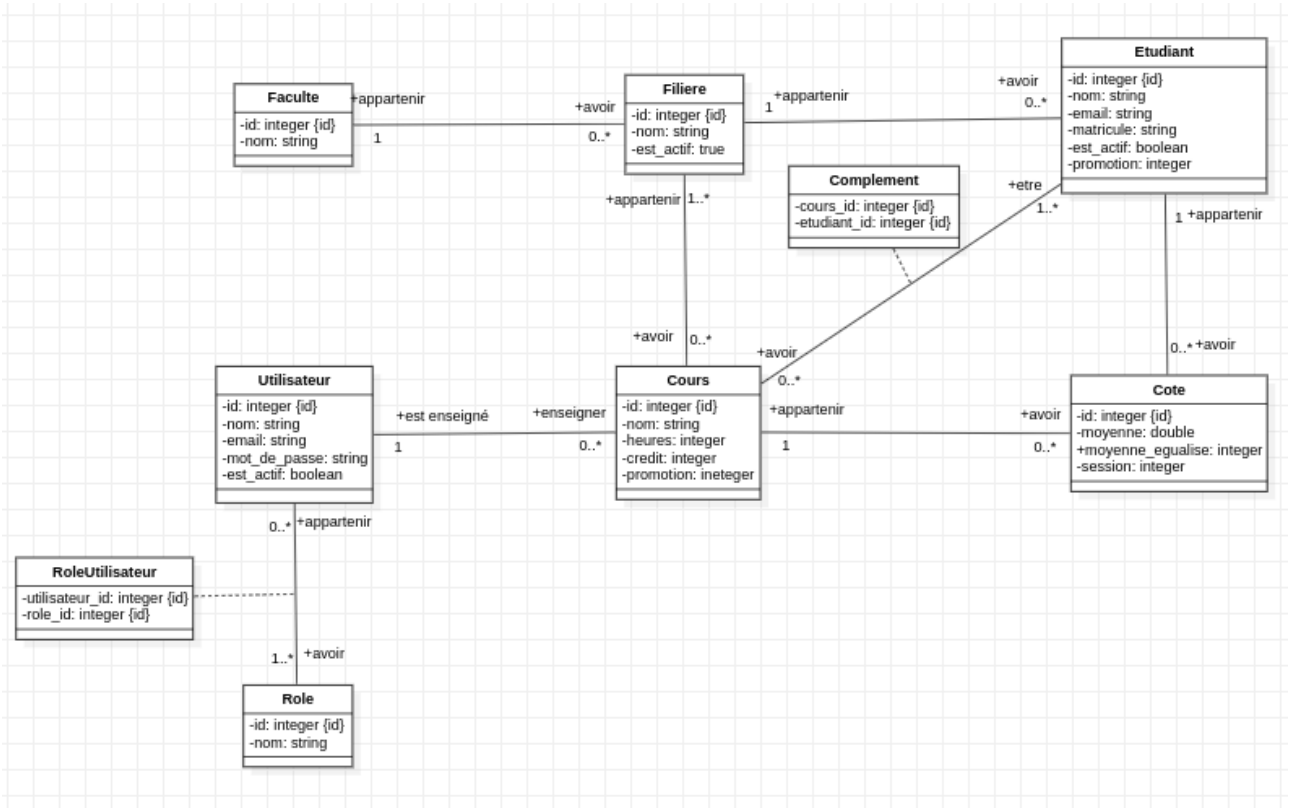
Nous voulons non seulement résoudre le problème de la délibération, mais aussi fournir une solution qui soit facile à maintenir et à faire évoluer à l'avenir et facilement adaptable à d'autres problèmes similaires surtout extensible.

Le choix d'une API offre une grande flexibilité et une grande extensibilité car la solution peut être utilisée par d'autres applications et peut être facilement intégrée à d'autres systèmes et peut être utilisée pour concevoir un autre type d'application (mobile, web, desktop, etc.).

3.2 MODÈLES

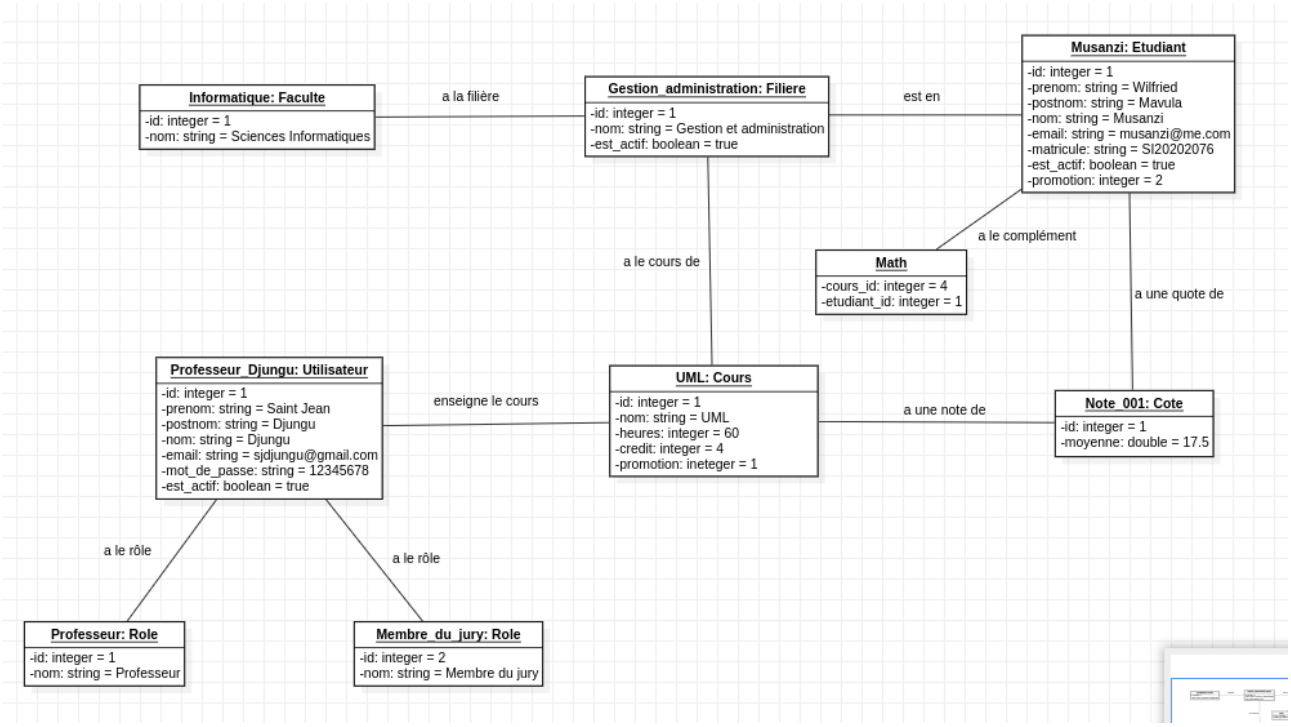
3.2.1 *Diagramme de classes*

FIGURE 1 – Diagramme de classes



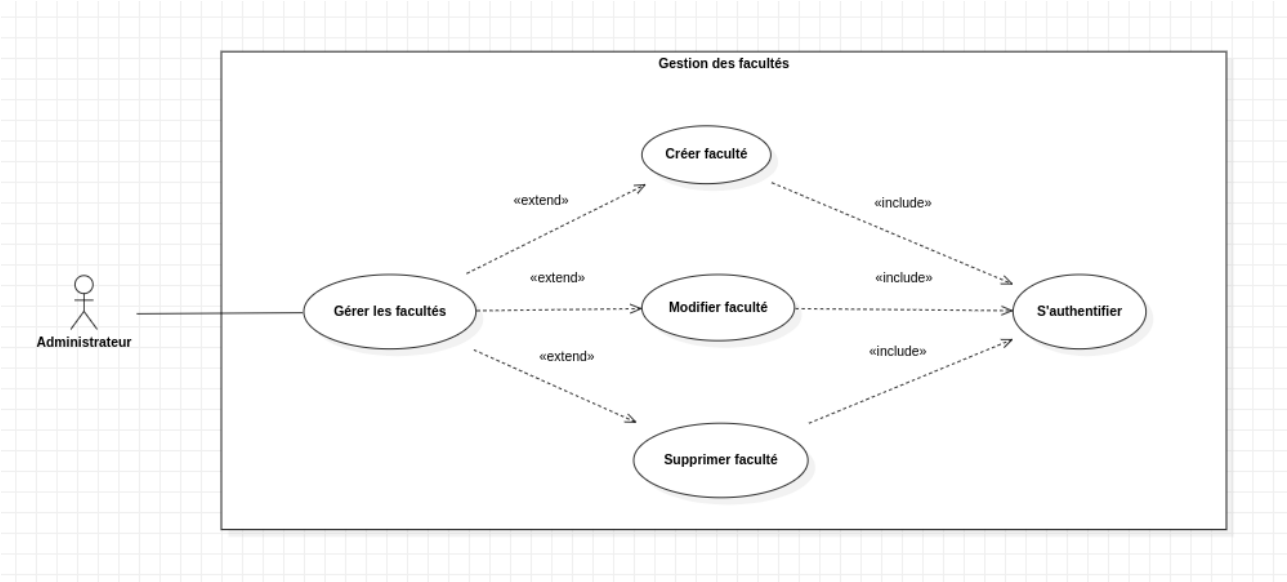
3.2.2 *Diagramme d'objets*

FIGURE 2 – Diagramme d'objets



3.2.3 *Diagramme de cas d'utilisation*

FIGURE 3 – Cas d'utilisation : Gestion des facultés



3.2.4 *Diagramme de séquence*

FIGURE 4 – Cas d'utilisation : Gestion des filières

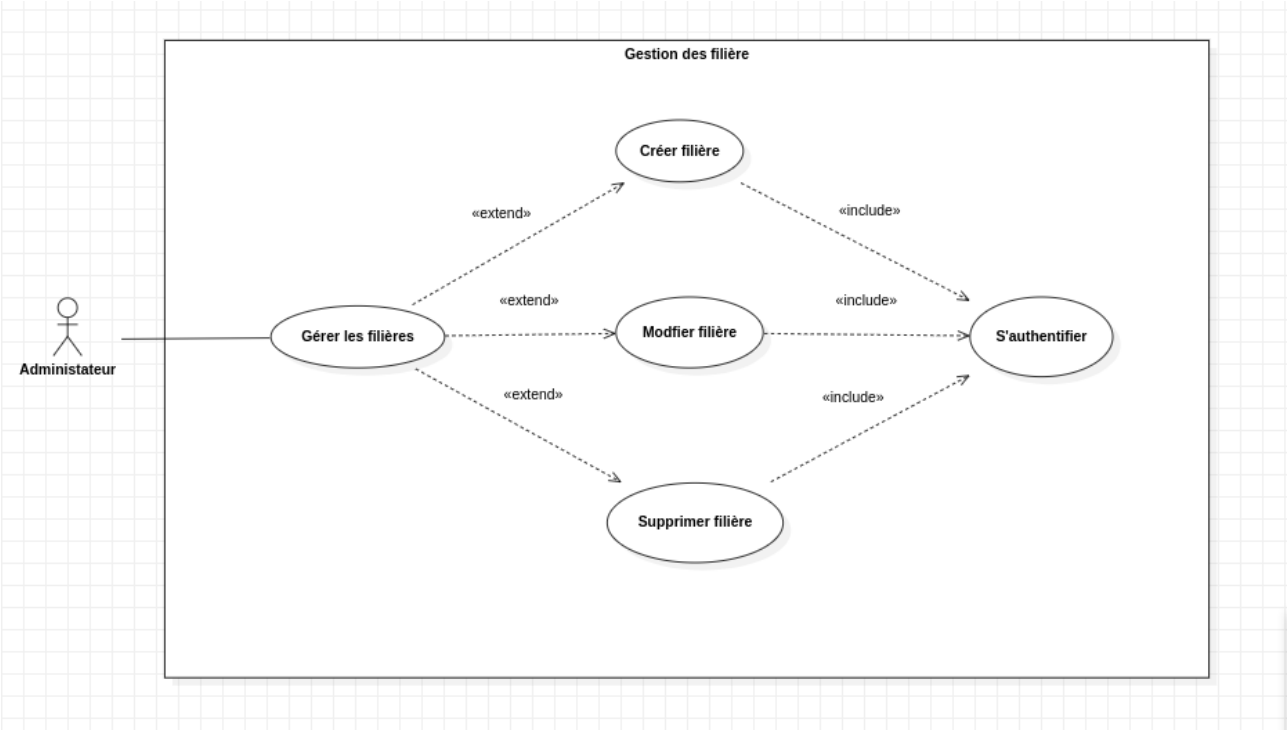


FIGURE 5 – Cas d'utilisation : Gestion des cours

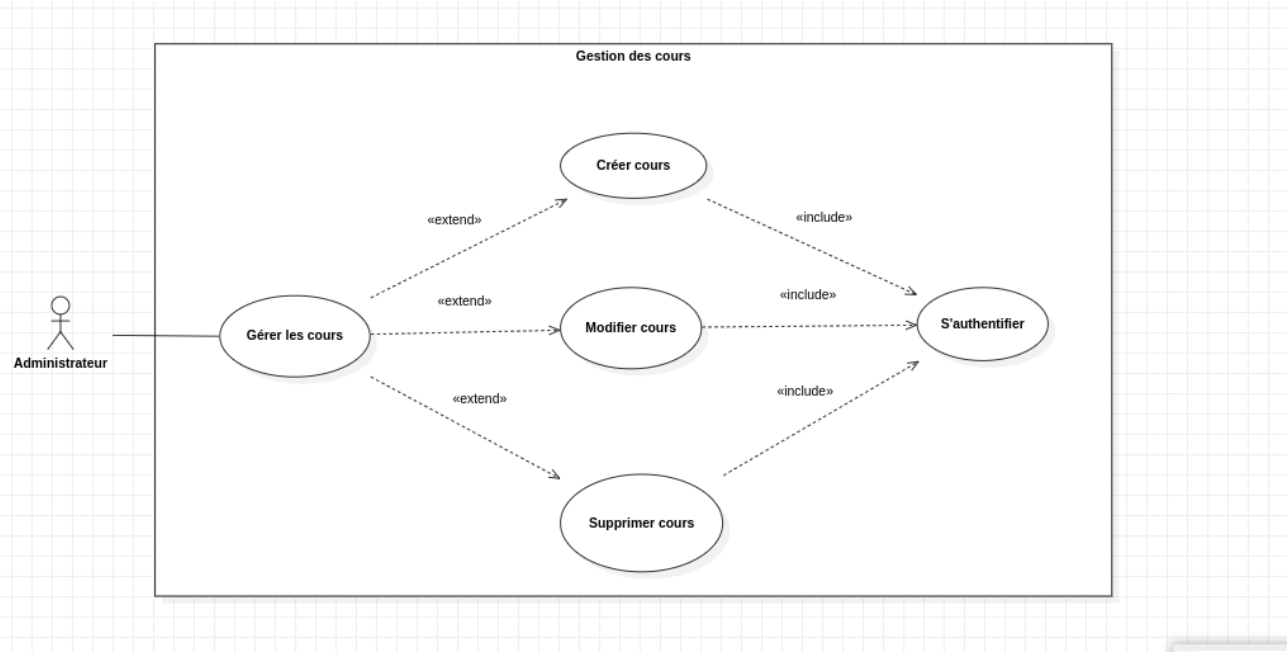


FIGURE 6 – Cas d'utilisation : Gestion des cotes

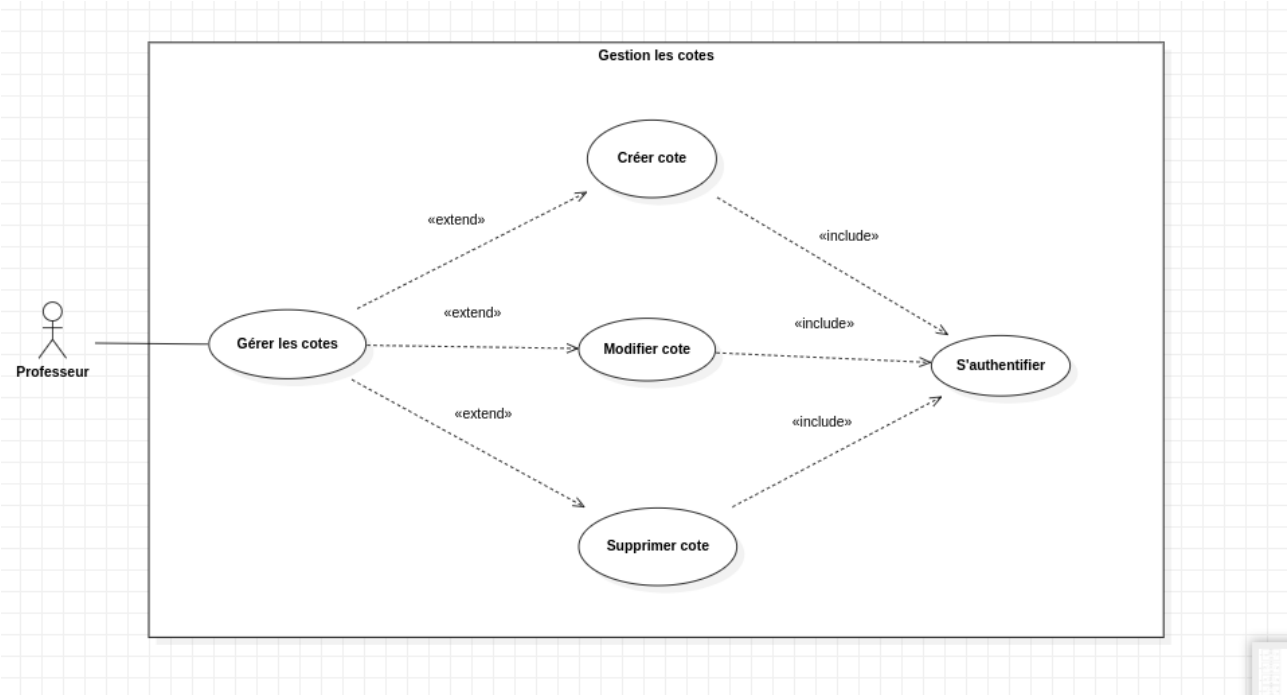


FIGURE 7 – Cas d'utilisation : Gestion des étudiants

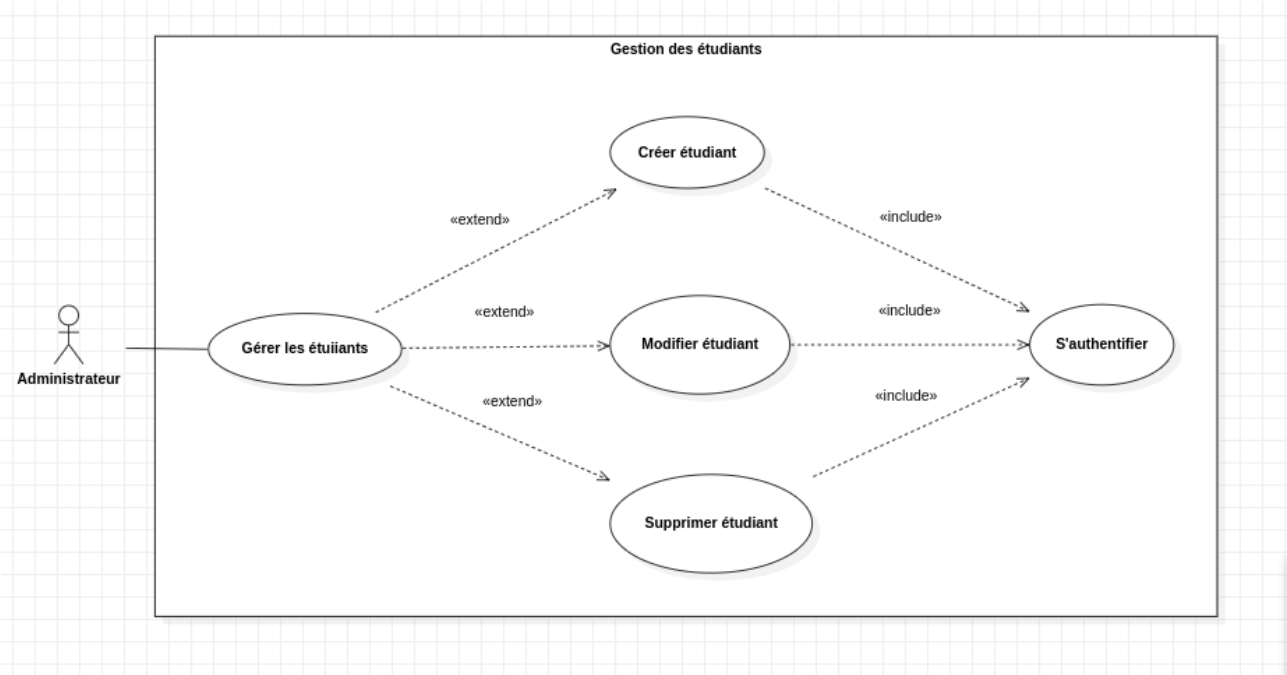


FIGURE 8 – Cas d'utilisation : Gestion des utilisateurs

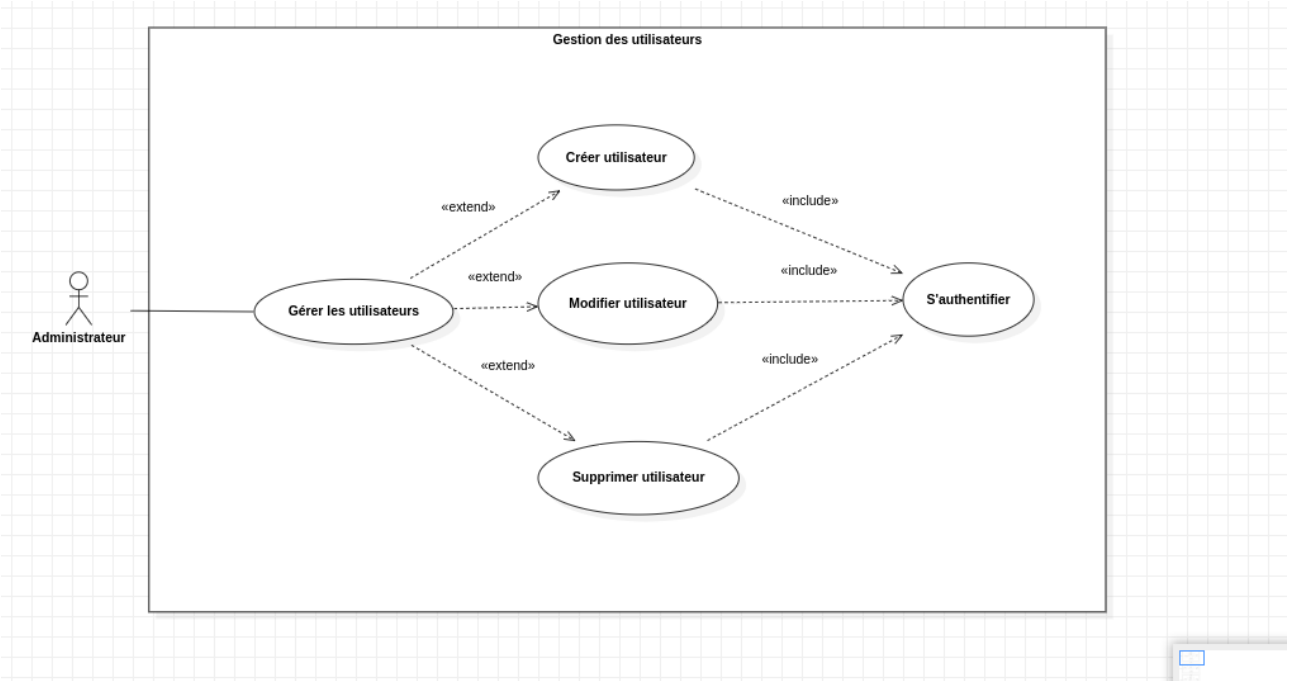


FIGURE 9 – Cas d'utilisation : Gestion des rôles

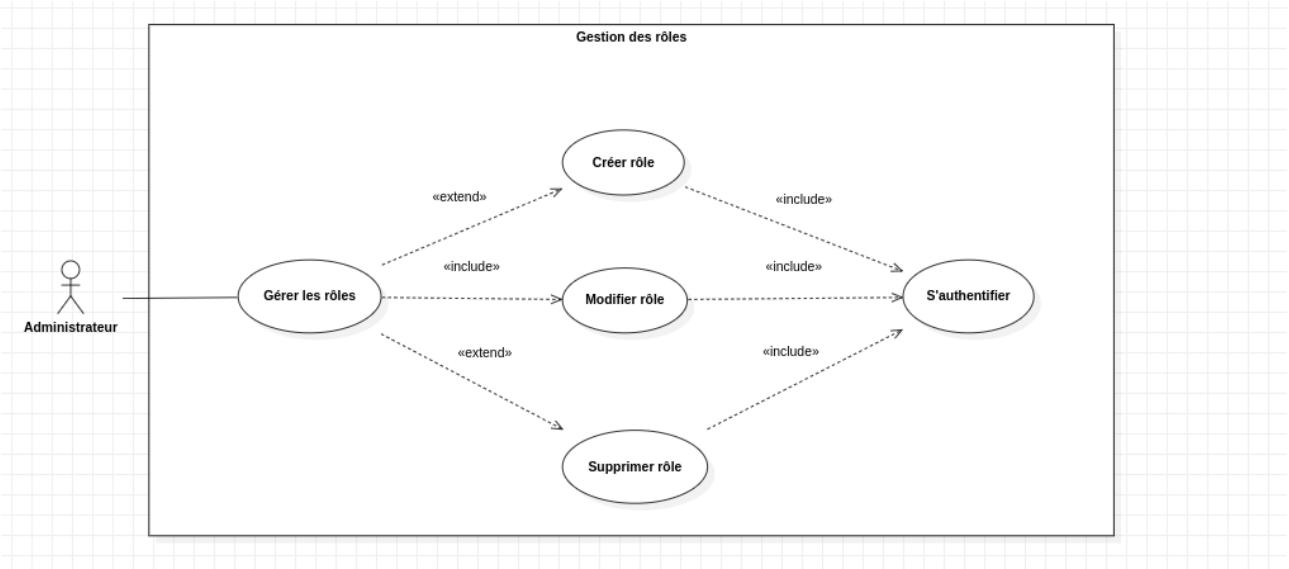


FIGURE 10 – Cas d'utilisation : Création des PDFs(Relevés)

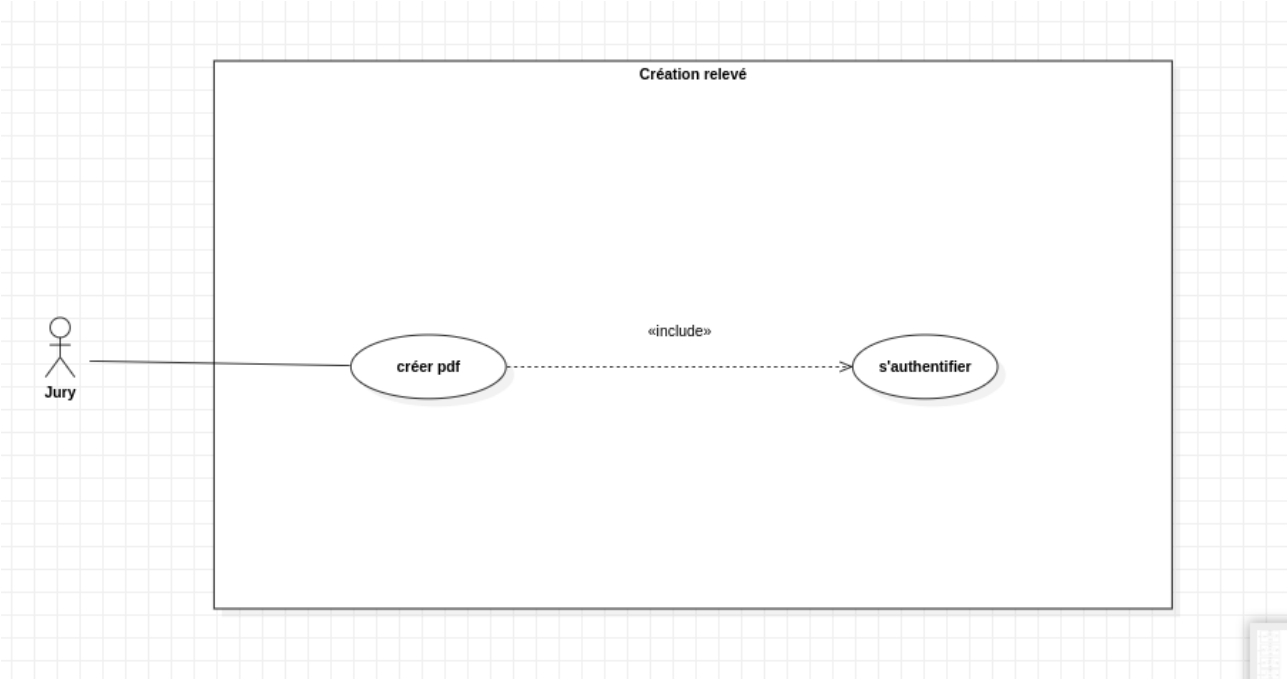


FIGURE 11 – Cas d'utilisation : Envoyer le pdf par mail

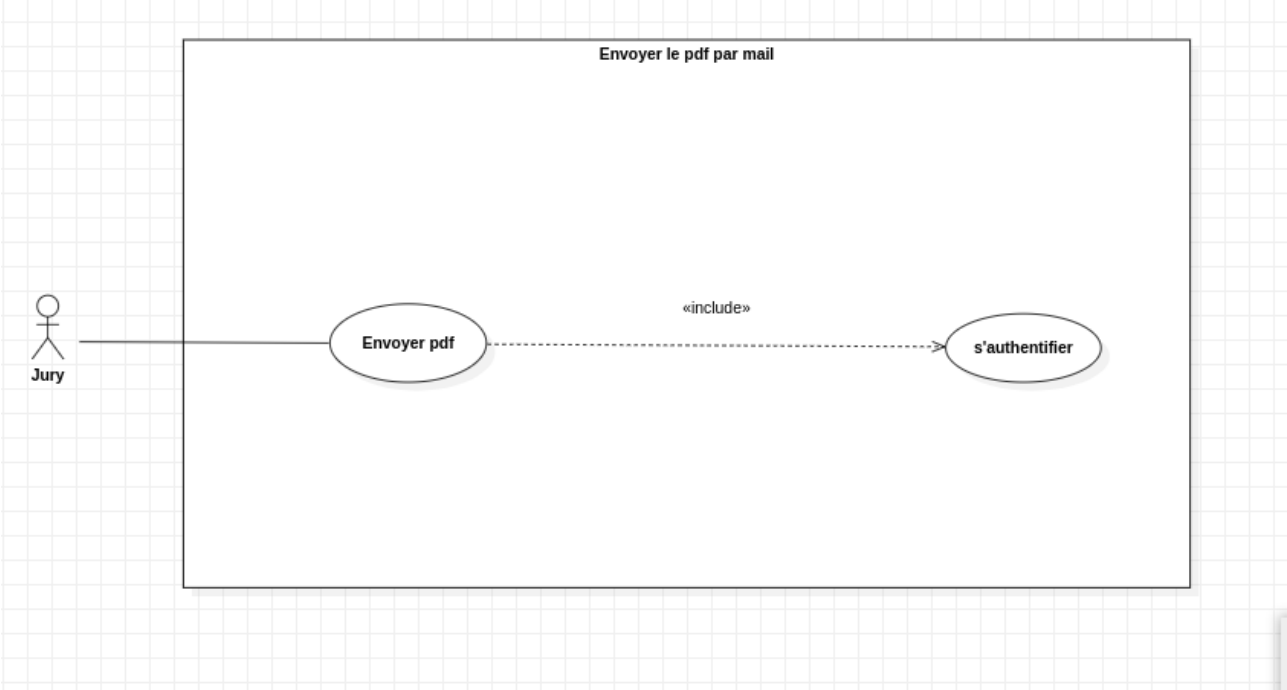
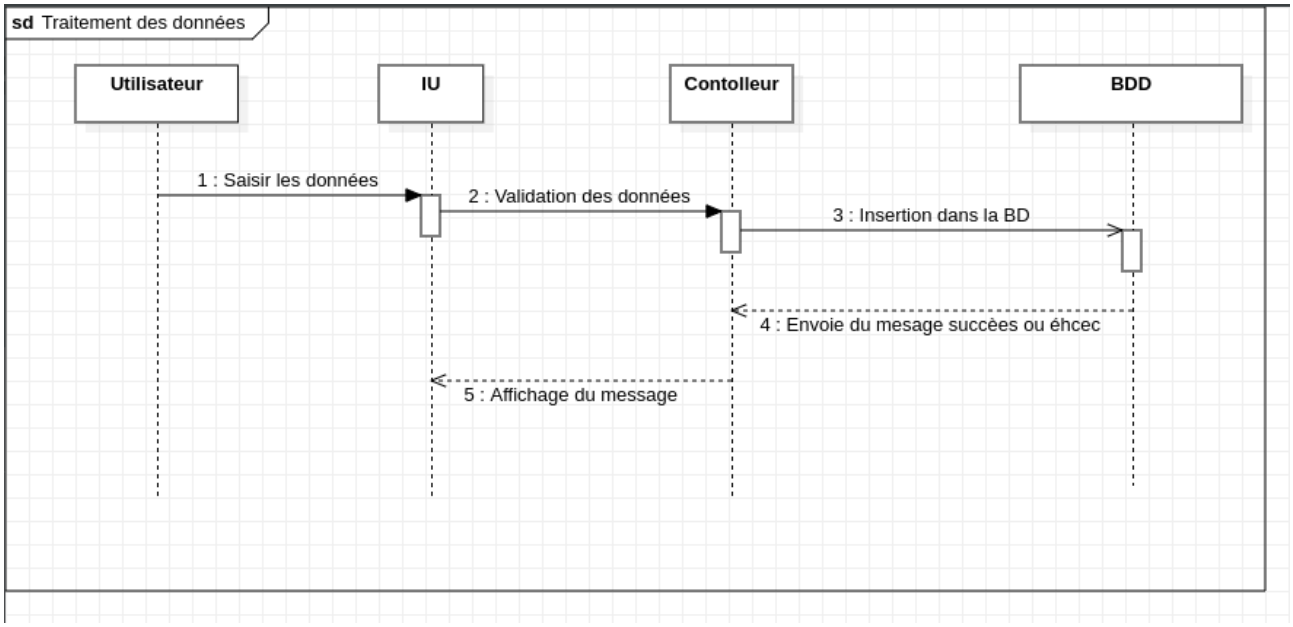


FIGURE 12 – Diagramme de séquence



4

PRÉSENTATION DES RÉSULTATS

4.1 PRÉSENTATION DE L'API REST

L'API comporte 44 routes au total que nous avons catégorisées en 9 groupes selon la ressource qu'elles manipulent ou leurs tâches.

4.1.1 *Les routes de l'authentification*

TABLE 1 – Tableau des routes de l'authentification

Méthode	Params	Corps	Réponses
POST /login	-	{ email, mot de passe }	{ jeton }
GET /profile	-	-	utilisateur : { id, nom, email, mot de passe, est_actif }
PATCH /profile	-	{ nom, email }	{ status, message }
PATCH /update-password	-	{ Ancien mot de passe, nouveau mot de passe }	{ status, message }

4.1.2 *Les routes de la gestion des facultés*

TABLE 2 – Tableau des routes de la gestion des facultés

Méthode	Action	Params	Corps	Réponse
POST /	Créer	-	{ nom }	{ status, message }
GET /	Lire	-	-	facultés[] : { id, nom }
GET / :id	Lire	{ id }	-	facultés : { id, nom }
PATCH / :id	Modifier	{ id }	{ nom }	{ status, message }
DELETE / :id	Supprimer	{ id }	-	{ status, message }

4.1.3 *Les routes de la gestion des filières*

TABLE 3 – Tableau des routes de la gestion des filières

Méthode	Action	Params	Corps	Réponse
POST /	Créer	-	{ nom, faculté }	{ status, message }
GET /	Lire	-	-	filières[] : { id, nom, faculté }
GET / :id	Lire	{ id }	-	filière : { id, nom, faculté }
PATCH / :id	Modifier	{ id }	{ nom, faculté }	{ status, message }
DELETE / :id	Supprimer	{ id }	-	{ status, message }

4.1.4 *Les routes de la gestion des cours*

TABLE 4 – Tableau des routes de la gestion des cours

Méthode	Action	Params	Corps	Réponse
POST /	Créer	-	{ nom, heures, crédits, promotion, enseignant, filière }	{ status, message }
GET /	Lire	-	-	cours[] : { id, nom, heures, crédits, promotion, enseignant, filière }
GET / :id	Lire	{ id }	-	cours : { id, nom, heures, crédits, promotion, enseignant, filière }
PATCH / :id	Modifier	{ id }	{ nom, heures, crédits, promotion, enseignant, filière }	{ status, message }
DELETE / :id	Supprimer	{ id }	-	{ status, message }

4.1.5 *Les routes de la gestion des étudiants*

TABLE 5 – Tableau des routes de la gestion des étudiants

Méthode	Route	Paramètre	Corps	Réponses
GET	/	-	-	{ étudiants[] }
GET	/ :id	{ id }	-	étudiant : { id, nom, matricule, email, filière, cours[] }
POST	/	-	{ nom, matricule, email, filière }	{ status, message }
PATCH	/ :id	{ id }	{ nom, matricule, email, filière }	{ status, message }
DELETE	/ :id	{ id }	-	{ status, message }
GET fields/ :fieldId/promotions/-promotionId	Lire	{ fieldId, promotionId }	-	étudiants[] : { id, nom, email, matricule, filière, promotion }
GET courses/ :coursID	Lire	{ coursId }	-	étudiants[] : { id, nom, email, matricule, promotion, compléments }

4.1.6 *Les routes de la gestion des cotes*

TABLE 6 – Tableau des routes de la gestion des cotes

Méthode	Route	Params	Corps	Réponses
POST /	Créer	-	{ moyenne, moyenne_eg, session, promotion_etudiant, cours, étudiant }	{ status, message }
GET /	Lire	-	-	cotes[] : { id, moyenne, moyenne_eg, session, promotion_etudiant, cours, étudiant }
GET / :id	Lire	{ id }	-	cote : {id, moyenne, moyenne_eg, session, promotion_etudiant, cours, étudiant }
PATCH / :id	Modifier	{ id }	{ id, moyenne, moyenne_eg, session, promotion_etudiant, cours, étudiant }	{ status, message }
DELETE / :id	Supprimer	{ id }	-	{ status, message }

4.1.7 *Les routes de la gestion des délibérations*

TABLE 7 – Tableau des routes de la gestion des délibérations

Méthode	Route	Params	Corps	Réponses
GET / :id	Lire	{ id }	-	{ nom, matricule, email, filière, cours[] }
GET generate-reports/ :id	Lire	{ id }	-	{ status, message }
GET send-reports/ :id	Lire	{ id }	-	{ status, message }

4.1.8 *Les routes de la gestion des utilisateurs*

TABLE 8 – Tableau des routes de la gestion des utilisateurs

Méthode	Route	Params	Corps	Réponses
GET /	Lire	-	-	utilisateurs[] : { id, nom, email, mot de passe, est_actif, rôles[] }
GET / :id	Lire	{ id }	-	utilisateur : { id, nom, email, mot de passe, est_actif, rôles[] }
POST /	Créer	-	{ nom, email, rôles[] }	{ status, message }
PATCH / :id	Modifier	{ id }	{ nom, email, rôles[] }	{ status, message }
DELETE / :id	Supprimer	{ id }	-	{ status, message }

4.1.9 *Les routes de la gestion des rôles*

TABLE 9 – Tableau des routes de la gestion des rôles

Méthode	Route	Params	Corps	Réponses
GET /	Lire	-	-	rôles[] : { id, nom }
GET / :id	Lire	{ id }	-	rôle : { id, nom }
POST /	Créer	-	{ nom }	{ status, message }
PATCH / :id	Modifier	{ id }	{ nom }	{ status, message }
DELETE / :id	Supprimer	{ id }	-	{ status, message }