DEDICACES

Je dédie ce travail

A ma mère, une femme battante, courageuse, aimante, source de tendresse et d'amour pour son soutient infaillible. Une femme qui n'a jamais cesser de croire en moi et qui ne cesse de prier pour ma réussite.

A mon père, un homme formidable qui m'a toujours soutenu et qui a fait tout son possible pour m'aider à avancer dans la vie.

Vous avez, de par votre amour, votre soutient, vos précieux conseils, votre éducation, œuvré à ma réussite. Pour toute assistance aussi bien matérielle qu'humaine, recevez à travers ce travail l'expression de ma gratitude, de mon amour et de tout mon respect envers vous.

A mon frère, qui n'a cessé d'être un exemple pour moi et ainsi **qu'à mes sœurs.**

Au professeur Mr Lamine Diop pour tout son aide.

A tous mes camarades de l'Université Amadou Hampate Ba

A tous ceux qui ont collaboré de près ou de loin à l'élaboration de ce travail.

Que Dieu vous protège et vous accorde santé, prospérité et longue vie

REMERCIEMENTS

Avant tout, je rends grâce à Allah et prie au nom du prophète Mouhamed PSL pour m'avoir permis de mener à terme ce projet qui présente pour moi le point de départ de ma carrière.

Il m'est agréable d'exprimer ma reconnaissance auprès de toutes les personnes, dont l'intervention au cours de ce projet, a favorisé son aboutissement.

En ce sens, je tiens à adresser chaleureusement mes remerciements,

À mes parents et ma famille sans qui rien de tout cela ne serait possible,

À **Monsieur Lamine Diop** pour le temps, l'attention, les directives précieuses et les conseils pertinents qui ont été d'un appui considérable au cours de ce projet.

Je tiens également à exprimer ma gratitude envers le président et les membres du jury pour avoir accepté d'évaluer ce travail.

Je ne saurais terminer sans remercier mes ami(e)s ainsi que mes camarades de classe avec qui j'ai passé d'excellents moments. A tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Je saisis également l'occasion pour adresser mes remerciements à tout le corps professoral de l'Université Amadou Hampaté Bâ de Dakar, pour la qualité de leurs enseignements ainsi que leur disponibilité tout au long de ces années de formation.

RÉSUMÉ

Dans le monde contemporain, l'information est devenue le principal moteur de progrès économique et social. L'important rôle qu'elle a dans une organisation lui fait consacrer des investissements importants et une attention particulière.

Dans le cadre de l'obtention de notre diplôme de LICENCE en Science et Technologie de l'Information et de la Communication (STIC) à l'Université Amadou Hampate Ba de Dakar, nous avons été appelés à réaliser un projet de fin d'études afin de clôturer notre formation du premier cycle universitaire.

Un projet qui consistera en l'analyse, la conception et la réalisation d'une solution informatique (application web) de gestion d'une école privée (Groupe Scolaire Ecole Plus).

Cette application sera conçue dans le dessein de gérer les différents employés, élèves, notes, bulletins, inscriptions et paiements afin de faciliter l'administration de l'établissement scolaire en permettant de digitaliser toutes les tâches qui avant étaient manuelles et le tout à travers une interface web simple d'utilisation (intuitif). Ainsi, son principal objectif sera de faciliter tout le travail administratif.

Nous avons opté pour une démarche définie comme suit :

- Recenser les besoins fonctionnels et non fonctionnels du projet.
- L'étude technique et la conception détaillée de l'application.
- Réalisation

Pour bien mener le développement de ce projet, nous avons choisi la méthodologie agile qui part du principe que spécifier et planifier dans les détails l'intégralité d'un produit avant de le développer, semble plus adéquate à notre contexte, et plus précisément la méthode SCRUM qui privilégie la livraison rapide d'un prototype, opérationnel par définition, afin que les clients, donneurs d'ordre et membres de l'équipe puissent l'évaluer.

ABSTRACT

In the contemporary world, information has become the main driver of economic and social progress. The important role it has in an organization makes it devote significant investments and special attention.

As part of obtaining our LICENSE diploma in Science and Technology of Information and Communication (STIC) at the Amadou Hampate Ba University of Dakar, we were called upon to carry out an end-of-studies project to complete our undergraduate education.

A project that will consist of the analysis, design and implementation of an IT solution (web application) for the management of a private school (School Group Ecole Plus).

We have opted for an approach defined as follows:

- Identify the functional and non-functional needs of the project.
- Technical study and established design of the application
- Realization

This application will be designed with the intention of managing the various employees, students, grades, report cards, registrations and payments in order to facilitate the administration of the school establishment by making it possible to digitize all the tasks which before were manual and all through an easy-to-use (intuitive) web interface. Thus, its main objective will be to facilitate all the administrative work.

To successfully carry out the development of this project, we chose the agile methodology, which assumes that specifying and planning in detail the entirety of a product before developing it, seems more appropriate to our context, and more precisely the method SCRUM which favors the rapid delivery of a prototype, operational by definition, so that customers, principals and team members can evaluate it.

LISTE DES ABRÉVIATIONS

Abréviations	**Significations**
AJAX	Asynchronous JavaScript and XML
AJDT	AspectJ Development Tools
CGI	Code General des Impôts
CIN	Carte d'Identité National
CSS	Cascading Style Sheets
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DSS	Diagramme de séquence système
GSEP	Groupe Scolaire Ecole Plus
HTML	HyperText Markup Language
MVC	Model View Controller
ORM	Object–relational mapping
PHP	Préprocesseur Hypertexte
POO	Programmation Oriente Objet
RAD	Rapid application development
RMI	Rempote methode invocation
SADT	Structured Analysis and Design Technique
SGBDR	Système de gestion de base de données relationelle
SI	Système d'information
SII	Système d'information informatisé
TD	Travauw Dirigés
TIC	Technologie de l'Information et de la communication
UP	Unified Process
UML	Unified Modeling Langage

TABLE DES MATIERES

Introduction Générale

Actuellement, le monde connait une avancée technologique considérable dans tous les secteurs et cela à l'aide de l'informatique qui joue un rôle important dans le développement de nombreuses entreprises et organisations. Le domaine informatique représente la révolution marquante la plus nécessaire et la plus innovante. En effet, loin d'être un éphémère phénomène de mode, ou une tendance passagère, l'informatique vient apporter de multiples applications à notre mode de vie. De plus, aucun domaine n'est resté étranger à cette stratégie qui offre tant de services aussi bien pour l'entreprise ou l'administration que pour le personnel.

Avant l'avènement de l'informatique, l'enregistrement des informations était manuel et se faisait sur des supports en papier. Ce qui engendrait beaucoup de problèmes tel que la perte de temps considérable dans la recherche de ces informations ou la dégradation de ces derniers. Un problème qui surtout, se ressentait dans le cadre scolaire.

Au Sénégal, organiser, structurer et gérer les dossiers scolaires des élèves, de même que les employés s'avèrent être un réel problème dû au fait que toutes les tâches sont faites manuellement entrainant ainsi la lenteur de toute activité.

La nouvelle logique de l'organisation du travail demande aux établissements d'éducation et d'apprentissage d'utiliser essentiellement l'information comme matière première pour pouvoir être plus efficace. Ils doivent donc intégrer un développement du système d'information dans leurs investissements stratégiques, dans la mesure où il structure la saisie, le stockage, l'organisation et la communication de l'information. Au-delà de l'utilisation individuelle de l'informatique, c'est surtout la mise en communication des ordinateurs, qui, permet de révolutionner les méthodes de travail. Les technologies de l'information et de la communication (TIC) constituent désormais des ressources importantes, voir incontournables dans le domaine éducatif. Le défi de l'intégration des TIC consiste donc à exploiter efficacement ces techniques afin qu'elles servent les intérêts du système éducatif.

Aujourd'hui, les écoles privées auxquelles nous rattacherons d'ailleurs notre étude, font partie intégrante des établissements scolaires où l'informatique pourra aider. En effet, la croissance exponentielle du nombre d'élèves oblige la mise en place d'une gestion rationnelle et rapide, or et jusqu'à ce jour, la manière de gérer

manuellement est encore dominante d'où la nécessité d'introduire l'informatique dans ces centres de formations.

Le principal objectif de ce projet sera donc de mettre en place un nouveau système de gestion scolaire facilitant la saisie et le partage des informations. La mission consiste en l'analyse, la conception et la réalisation des modules de gestion des établissements à savoir les élèves, les employés, etc.

L'étude sera axée sur quatre grands points structurés en sept chapitres. La première partie, **Cadre théorique et méthodologique,** sera subdivisée en deux chapitres.

- Le premier chapitre (chapitre 1) intitulé le cadre dans lequel nous allons définir la problématique, les objectifs de la recherche ainsi que les hypothèses de cette dernière.
- Dans le second chapitre (chapitre 2) nommé cadre méthodologique, on parlera du cycle de vie d'un. En fin nous vous décrirons le choix de la méthode d'analyse et de conception.

Ensuite nous nous pencherons sur la deuxième partie, **Le cadre analytique**, qui sera composée d'un chapitre

- Ce chapitre (chapitre 3) nommé analyse de l'existant nous permettra de recenser les points forts et faibles du système en cours grâce à une analyste objective du système.

A celles-là s'ajoutera une troisième partie, **Le cadre conceptuel**. Cette partie sera axée sur les points de vue fonctionnels et statiques.

- Le premier chapitre (chapitre 4) de cette troisième partie intitule Point de vue fonctionnel présentera les diagrammes de cas d'utilisation et de séquence.
- Quant au second chapitre (chapitre 5) intitule Point de vue statique, il présentera les diagrammes de classes.

Et pour finir, la quatrième et dernière partie, intitule **Réalisation,** divisée en deux chapitres que sont :

- Plateforme de développement (chapitre 6)
- Présentation de l'application (chapitrer 7),

1ère partie : Le cadre théorique et méthodologique

1. Chapitre 1 : Le cadre théorique et méthodologique

1.1. La problématique :

Nous avons interrogé le directeur de l'école qui nous a fait part de quelques difficultés auxquels l'établissement fait face, mais pour localiser leur source, nous nous sommes mis en pratique avec elle. Et après une observation continuelle, nous avons pu observer certaines insuffisances. En effet, nous avons pu constater, pendant notre observation au sein du Groupe Scolaire Ecole Plus (GSEP) que la gestion scolaire s'avère être un travail fastidieux et multifacette tant les quantités d'informations et de détails à traiter au quotidien ne cessent d'augmenter de même que la majeure partie des traitements se fait manuellement, ce qui engendre un certain nombre de problèmes tels que :

- Volume important des informations traitées manuellement, ce qui provoque parfois des erreurs dans l'établissement des documents (égarement lors de l'écriture des données).
- Recherche difficile sur les registres qui engendre une perte de temps.
- Insécurité des informations.
- Possibilité d'erreur dans le remplissage des différents documents et registres.
- Possibilité d'erreur dans le calcul des statistiques.
- Nombre important des archives qui engendre une difficulté de stockage.
- Détérioration des archives à force de leur utilisation trop fréquente.
- Mauvaise codification sur quelques objets dans la gestion d'information.

En ce sens, l'informatisation du système nous semble être la solution la plus adéquate pour pallier à ces problèmes puisqu'elle répond mieux aux problèmes souvent fréquentés dans la gestion manuelle.
Numériser toutes les tâches permettra d'assurer l'accès instantané aux données et une sécurisation de ces dernières, ce qui simplifie le travail administratif.

C'est dans ce cadre que s'inscrit notre Projet de fin d'Etude : << **Gestion d'établissement scolaire** >> qui a pour objectif d'informatiser le système de gestion des étudiants et du personnel du **GSEP** ainsi que leurs utilisateurs. De ce fait, nous avons proposé aux responsables de l'école de leur concevoir une application Web qui

va gérer les activités de l'école et qui va permettre par la suite de minimiser le support papier et d'améliorer la rapidité de l'accès à l'information.

Pour ce faire nous avons assigné à notre étude quelques objectifs tels que :

1. Rapidité dans l'établissement des différents documents.
2. Facilité de la recherche et l'accès aux informations.
3. Stockage des informations sur des supports informatiques pour assurer leur sécurité.
4. Gain de temps dans le calcul des statistiques.
5. Automatiser les tâches qui se traitent manuellement.

1.2. Les objectifs de la recherche :

L'objectif de ce projet est de remédier aux défaillances et problèmes rencontre au sein de l'établissement. Pour cela, nous proposons la solution qui nous a paru la plus adéquate c'est-à-dire de concevoir une application qui assurera la gestion des élèves, des enseignants, des employés ainsi que du corps administratif de l'établissement

L'application permettra donc d'assurer :

- La gestion des utilisateurs (enseignants, caissiers, administrateurs, élèves)
- La gestion des élèves (Saisie de notes, des absences, des convocations, la gestion des emplois du temps, des salles, l'Edition des bulletins séquentiels, trimestrielle et annuels des élevés, des cartes d'identités scolaires et des certificats de scolarités).
- La gestion des professeurs (Gestion de l'assiduité des professeurs, saisie, validation et suivi du projet pédagogique, édition de la fiche pédagogique séquentielle, trimestrielle et annuelle).
- La gestion du calendrier scolaire (Horaire et emploi du temps)
- La gestion des évaluations (Contrôle, Composition et Examens Blanc)
- La gestion des absences et des retards (Elèves, Professeurs et Personnel)
- La gestion des notes et des bulletins

La solution informatique proposée doit notamment permettre :

- Un suivi par élève
- Le traitement des demandes d'accès aux dossiers des élèves.
- L'amélioration de la gestion des informations en les centralisant.
- L'accès à l'historique des élèves précédents, etc.
- L'établissement des documents qualités relatifs à la gestion du dossier d'un élève.
- Amélioration de la disponibilité du dossier d'un élève et des informations qu'il contient, y compris en cas d'urgence.
- L'automatisation de l'ensemble des procédures de gestion
- La consolidation des données dans une seule base de données pour notamment suivre le parcours des élèves dans les autres écoles privées
- L'archivage des données

2. Chapitre 2 : Le cadre méthodologique :

2.1. Cycle de vie d'un logiciel

On parle souvent du cycle de vie d'un logiciel. Par cela, on entend toutes les phases de développement du logiciel, de l'établissement des besoins du client jusqu'à l'achèvement du logiciel en tant que produit commercial.

Le "cycle de vie d'un logiciel" (en anglais **software lifecycle**) désigne donc toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. On parle du cycle de vie d'un logiciel pour définir des repères intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés et la vérification du processus de développement. La création d'un logiciel est très complexe et nécessite un ordre précis des étapes.

Découper le cycle de vie d'un logiciel en étapes "unitaire" va permettre de mettre en place des jalons et des processus de vérification et de validation qui leur seront associés. Le cycle de vie d'un logiciel va vous permettre de vous assurer de la qualité du produit et maintenir le coût du projet, de maîtriser les coûts de réalisation. En effet, plus vous réparez une erreur ou une anomalie tardivement, plus la correction de l'application va coûter chère.

Ce cycle de vie contient différentes activités dans différentes phases du développement du logiciel, en particulier :

- **La définition des objectifs** : Cette étape consiste à déterminer la finalité du projet et son inclusion dans une stratégie globale.
- **L'analyse des besoins, appelée aussi spécifications des besoins** : Cette phase doit contenir la mise du logiciel dans son contexte (type de produit, nouveau/altéré) et l'étude de l'existant. L'étude de l'existant désigne l'étude des produits similaires dans le marché (veille concurrentielle) et l'étude du processus ou des logiciels similaires à l'entreprise. Les besoins de l'entreprise peuvent contenir des besoins fonctionnels (des fonctionnalités que le produit doit automatiser) et des besoins non fonctionnels (disponibilité, rapidité de calcul.
- **La conception** : La conception utilise les spécifications pour décider des solutions proposées. Elle peut contenir la description des fonctionnalités de l'application.

Il s'agit des fonctionnalités précisées lors de la spécification des besoins. La conception peut contenir la conception des interfaces, la conception des données et la conception de l'architecture matérielle.

- **Le codage** : Il transforme des solutions proposées lors de la conception en un code opérationnel. Les techniques de codage dépendent du langage et doivent être bien conforme à la conception.
- **Le test** : Il s'agit de la phase de test et de validation. Les tests déterminent les bugs techniques, les bugs fonctionnels et la qualité du logiciel. Pour cela, on peut utiliser des logiciels de test, des techniques et des benchmarks.
- **Le déploiement appelé "phase de livraison" et "phase de mise en exploitation"** : La phase de déploiement regroupe toutes les activités qui mènent à l'installation et la mise en marche de l'application développée (installation des serveurs, setup et configuration des composants du logiciel développé et le test de déploiement).
- **La maintenance** : Cette étape consiste à ajuster l'application après la livraison du produit au client. Elle a pour but de corriger les erreurs et les anomalies du système et modifier le système pour y ajouter des fonctionnalités.

2.2. Méthode d'analyse et de conception

2.2.1. Classification des méthodes d'analyse

Les méthodes d'analyse et de conception peuvent être divisées en quatre grandes familles

2.2.1.1. Les méthodes cartésiennes ou fonctionnelles

Avec ces méthodes (SADT, SA-SD), le système étudié est abordé par les fonctions qu'il doit assurer plutôt que par les données qu'il doit gérer. Le processus de conception est vu comme un développement linéaire. Il y'a décomposition systématique du domaine étudié en sous domaines, eux-mêmes décomposés en sous domaines jusqu'à un niveau considéré comme élémentaire.

2.2.1.2. Les méthodes systémiques

Dans les méthodes systémiques (Merise, REMORA, etc.), le système est abordé à travers l'organisation des systèmes constituants l'entreprise. Elles aident donc à construire un système en donnant une représentation de tous les faits pertinents qui

surviennent dans l'organisation en s'appuyant sur plusieurs modèles à des niveaux d'abstraction différents (conceptuel, organisationnel, logique, physique, etc.)

2.2.1.3. Les méthodes objets

L'approche objet permet d'appréhender un système en centrant l'analyse sur les données et les traitements à la fois. Les stratégies orientées objet considèrent que le système étudié est un ensemble d'objet coopérant pour réaliser les objectifs des utilisateurs. Les avantages qu'offre une méthode de modélisation objet par rapport aux autres méthodes sont la réduction de la « distance » entre le langage de l'utilisateur et le langage conceptuel, le regroupement de l'analyse des données et des traitements, la réutilisation des composants mis en place.

2.2.1.4. Approche oriente aspect

Bien qu'en étant encore à ses débuts, la **Programmation Orientée Aspect** commence à se faire connaître et séduit. C'est un principe novateur qui permet de résoudre les problèmes de séparation des préoccupations d'une application. Le code résultant devient plus lisible, réutilisable et le remplacement de composants se fait rapidement et à moindre coût du fait de la séparation des préoccupations. Cette séparation se fait par la création d'aspects contenant le code à greffer à l'application. Un programme appelé « tisseur » greffe ensuite les aspects de façon statique après la compilation, ou de façon dynamique au moment de l'exécution.

Il existe déjà des tisseurs matures, comme **AspectJ** pour Java, qui est parfaitement intégré à Eclipse grâce au plug-in **AJDT**. La plateforme .NET possède aussi des tisseurs très prometteurs, tels que **AspectDNG** qui permet déjà une utilisation professionnelle.

2.2.2. Définition de quelques mots -clés

- **Code cible ou code de base** : Ensemble de classes qui constituent une application ou une bibliothèque. Ces classes n'ont pas connaissance des aspects (il n'y a donc aucune dépendance), elles ne se "doutent" pas qu'elles vont constituer la cible d'un tissage.
- **Aspect** : Il s'agit du code que l'on souhaite tisser. Selon les tisseurs, un aspect peut être une simple classe ou constituer un élément d'un langage spécifique.

- **Zone de greffe** : Ce sont les "endroits" dans le code cible dans lesquels aura lieu le tissage.
- **Tissage** : Processus qui consiste à ajouter (tisser, greffer, injecter) le code des aspects sur les zones de greffe du code cible.
- **Tisseur** : Programme exécutable ou Framework dynamique qui procède au tissage. Le tisseur peut être invoqué :
    - Statiquement avant la compilation du code cible (le tisseur travaille sur le code source),
    - Statiquement pendant la compilation du code cible (le tisseur intègre un compilateur du langage de programmation),
    - Statiquement après la compilation du code cible (le tisseur travaille sur le code précompilé : **bytecode**),
    - Dynamiquement avant exécution du code (au moment de la compilation Just-In-Time),
    - Dynamiquement pendant l'exécution du code (par la technique d'interception d'invocation de méthode, utilisée intensivement par les Framework d'inversion de contrôle).

2.2.3. Avantages et inconvénients

Le couplage entre les modules gérant des aspects techniques peut être réduit de façon très importante, en utilisant ce principe, ce qui présente de nombreux avantages :

- **Maintenance aisée** : les modules techniques, sous forme d'aspect, peuvent être maintenus plus facilement du fait de son détachement de son utilisation,
- **Meilleure réutilisation** : tout module peut-être réutilisé sans se préoccuper de son environnement et indépendamment du métier ou du domaine d'application. Chaque module implémentant une fonctionnalité technique précise, on n'a pas besoin de se préoccuper des évolutions futures : de nouvelles fonctionnalités pourront être implémentées dans de nouveaux modules qui interagiront avec le système au travers des aspects.
- **Gain de productivité** : le programmeur ne se préoccupe que de l'aspect de l'application qui le concerne, ce qui simplifie son travail.

- **Amélioration de la qualité du code** : la simplification du code qu'entraine la programmation par aspect permet de le rendre plus lisible et donc de meilleure qualité.

Par contre le tissage d'aspect qui n'est finalement que de la génération automatique de code inséré à certains points d'exécution du système développé, produit un code qui peut être difficile à analyser (parce que généré automatiquement) lors des phases de mise au point des logiciels (débogages, test, etc.).
Mais en fait cette difficulté est du même ordre que celle apportée par toute décomposition non linéaire (fonctionnelle ou objet par exemple).

2.2.4.UP et RAD

Dans cette partie nous allons présenter deux démarches d'analyse et de conception. Il s'agit de **UP** et de la méthode **RAD**.

2.2.4.1. United Process (processus unifié))

2.2.4.1.1. Définition

Le processus unifié est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. C'est un patron de processus pouvant être adapté à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprise, à différents niveaux de compétence et à différentes tailles de l'entreprise.

- UP est Itératif

L'itération est une répétition d'une séquence d'instructions ou d'une partie de programme un nombre de fois fixé à l'avance ou tant qu'une condition définie n'est pas remplie, dans le but de reprendre un traitement sur des données différentes.

Elle qualifie un traitement ou une procédure qui exécute un groupe d'opérations de façon répétitive jusqu'à ce qu'une condition bien définie soit remplie.

Une itération prend en compte un certain nombre de cas d'utilisation et traite en priorité les risques majeurs.

Figure 3.1 : Illustration du caractère itératif de UP

- UP est centré sur l'architecture

Une architecture adaptée est la clé de voûte du succès d'un développement. Elle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité...).

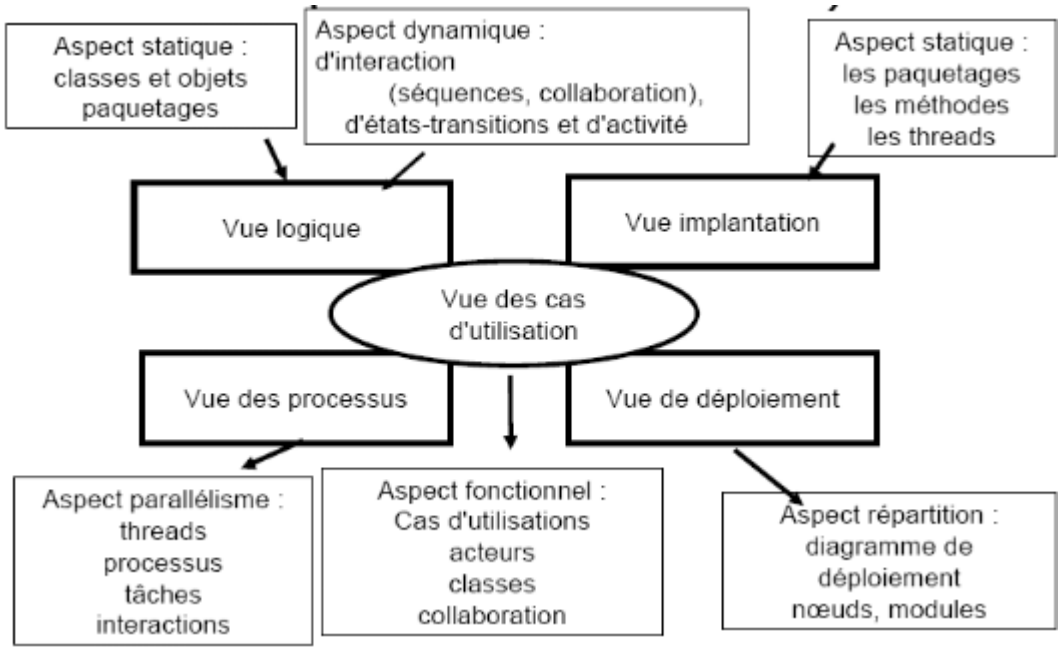


Figure 3.2 : Modèle représentant les 4+1 vues de Kruchten

- UP est piloté par les cas d'utilisation d'UML

Le but principal d'un système informatique est de satisfaire les besoins du client. Le processus de développement sera donc accès sur l'utilisateur.

Les cas d'utilisation permettent d'illustrer ces besoins. Ils détectent puis décrivent les besoins fonctionnels (du point de vue de l'utilisateur), et leur ensemble constitue le modèle de cas d'utilisation qui dicte les fonctionnalités complètes du système.

2.2.4.1.2. Vie du processus unifiée

L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques. UP est un ensemble de principes génériques adapté en fonctions des spécificités des projets.

UP répond aux préoccupations suivantes :

- **QUI** participe au projet ?
- **QUOI**, qu'est-ce qui est produit durant le projet ?
- **COMMENT** doit-il être réalisé ?
- **QUAND** est réalisé chaque livrable ?

UP gère le processus de développement par deux axes.

- L'axe vertical représente les principaux enchaînements d'activités, qui regroupent les activités selon leur nature. Cette dimension rend compte l'aspect statique du processus qui s'exprime en termes de composants, de processus, d'activités, d'enchaînements, d'artefacts et de travailleurs.
- L'axe horizontal représente le temps et montre le déroulement du cycle de vie du processus ; cette dimension rend compte de l'aspect dynamique du processus qui s'exprime en termes de cycles, de phases, d'itérations et de jalons.

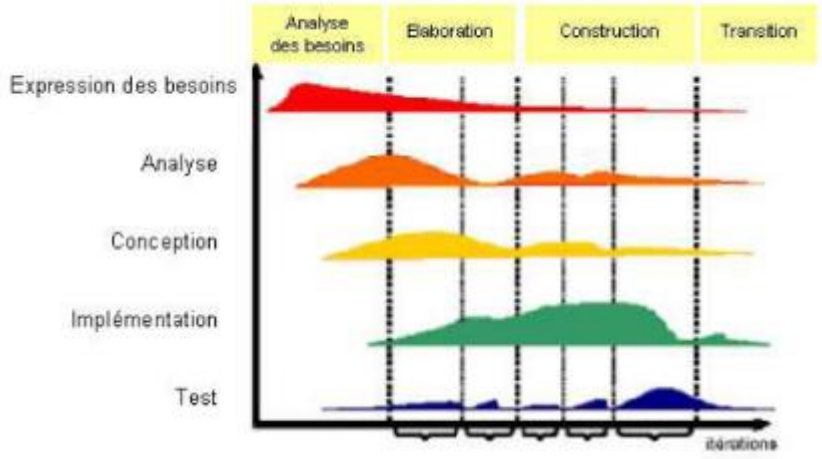


Figure 3.3 : Architecture bidirectionnelle du PU

UP répète un certain nombre de fois une série de cycle qui s'articule autour de 4 phases qui sont : analyse des besoins, élaboration, construction, transition.

Pour mener efficacement un tel cycle, les développeurs ont besoins de toutes les représentations du produit logiciel qui sont : un modèle de cas d'utilisation, un modèle d'analyse détaillant les cas d'utilisation et procéder à une première répartition du comportement, un modèle de conception, finissant la structure statique du système sous forme de sous-systèmes de classes et interfaces, un modèle d'implémentation, intégrant les composants, un modèle de déploiement, définissant les nœuds physiques des ordinateurs, un modèle de test, décrivant les cas de test vérifiant les cas d'utilisation, et une représentation de l'architecture.

2.2.4.1.3. Les phases

Pour bien mener un projet du début à la fin, UP préconise un certains nombre de phases.

2.2.4.1.3.1. Expression des besoins

L'expression des besoins comme son nom l'indique, permet de définir les différents besoins : inventorier les **besoins** principaux et fournir une liste de leurs fonctions, recenser les **besoins fonctionnels** (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation, appréhender les **besoins non fonctionnels** (technique) et livrer une liste des exigences. Le modèle de cas d'utilisation présente le système du point de vue de l'utilisateur et représente sous forme de cas d'utilisation et d'acteur, les besoins du client.

2.2.4.1.3.2. Analyse

L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution. Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation et les structure sous une forme qui facilite la compréhension (scénarios), la préparation (définition de l'architecture), la modification et la maintenance du futur système.

Il s'écrit dans le langage des développeurs et peut être considéré comme une première ébauche du modèle de conception.

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine les principales interfaces et les transcrit à l'aide d'une notation commune. Elle constitue un point de départ à l'implémentation :

- Elle décompose le travail d'implémentation en sous-système
- Elle crée une abstraction transparente de l'implémentation

2.2.4.1.3.3. Implémentation

L'implémentation est le résultat de la conception pour implémenter le système sous formes de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments du même type. Les objectifs principaux de l'implémentation sont de planifier les intégrations des composants pour chaque itération, et de produire les classes et les sous-systèmes sous formes de codes sources.

2.2.4.1.3.4. Tests

Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque itération, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun.

2.2.4.2. RAD

Aujourd'hui, qualité et réactivité font partie des objectifs généraux de beaucoup d'entreprises. Cela entraîne un certain nombre de projets, qui tout en apportant satisfaction aux utilisateurs, doivent être menés dans un délai court. C'est à cela que répond la méthode RAD.

RAD est une méthode basée sur le partenariat. L'utilisateur s'affirme le vrai maître de son application et, par sa participation active, il s'en approprie la réalisation. Le RAD et le prototypage permettent de réaliser en concevant, tout en testant ce que l'on réalise. La méthode RAD propose de remplacer le cycle de vie classique par un autre découpage temporel. Le déroulement est d'abord linéaire, puis il suit le modèle de la spirale. Les étapes sont au nombre de cinq.

2.2.4.2.1. La phase d'incubation

Cette phase permet de définir le périmètre général du projet, de structurer le travail par thèmes, de sélectionner les acteurs pertinents et d'amorcer une dynamique de projet. Elle représente environ 6% du projet en charge.

2.2.4.2.2. La phase Expression des besoins

Cette phase permet de spécifier les exigences du système lors des entretiens avec les utilisateurs et de définir la solution globale sur les plans stratégique, fonctionnel, technologique et organisationnel. Cette phase représente environ 9% du projet.

2.2.4.2.3. La phase de Conception

Cette phase permet de concevoir et de modéliser le futur système avec le concours des utilisateurs pour l'affinage et la validation des modèles. C'est aussi dans cette phase que nous validons le premier niveau de prototype présentant l'ergonomie et la cinématique générale de l'application. Cette phase représente environ 23% du projet.

2.2.4.2.4. La phase de Construction

Durant cette phase, notre équipe développe l'application module par module. L'utilisateur participe toujours activement aux spécifications détaillées et à la validation des prototypes. **Plusieurs sessions itératives sont nécessaires**. Cette phase représente environ 50% du projet.

2.2.4.2.5. La phase de Mise en œuvre

Des recettes partielles ayant été obtenues à l'étape précédente, il s'agit dans cette phase d'officialiser une livraison globale et de transférer le système en exploitation et maintenance. Cette phase représente environ 12% du projet.

2.3. Choix de la méthode d'analyse et de conception

2.3.1. Définition des concepts

L'analyse permet de lister les résultats attendus, en termes de fonctionnalités, de performance, de robustesse, de maintenance, de sécurité, d'extensibilité, etc. L'analyse

répond donc à la question « que faut-il faire ? » et a pour but de se doter d'une vision claire et rigoureuse du problème posé et du système à réaliser en déterminant ses éléments et leurs interactions. Elle met l'accent sur une investigation du problème et des exigences, plutôt que sur une solution. A ce niveau d'abstraction, on doit capturer les besoins principaux des utilisateurs en identifiant les éléments du domaine, ainsi que les relations et interactions entre ces éléments : les éléments du domaine sont liés au(x) métier(s) de l'entreprise, ils sont indispensables à la mission du système, ils gagnent à être réutilisés (ils représentent un savoir-faire).

La conception permet de décrire de manière non ambiguë, le plus souvent en utilisant un langage de modélisation, le fonctionnement futur du système, afin d'en faciliter la réalisation. La conception menée à la suite de la phase d'analyse répond donc à la question « comment faut-il faire ce qu'il faut faire ? ». Elle met l'accent sur une solution conceptuelle qui satisfait aux exigences plutôt que sur une **implémentation**[*]. Le processus de conception a ainsi pour but de figer les choix techniques (outils, langages, Framework). Il permet ainsi de modéliser tous les rouages d'implémentation et de détailler tous les éléments de modélisation issus des niveaux supérieurs. Les modèles sont optimisés, car destinés à être implémentés.

Ainsi, l'**analyse** sert à la découverte et à la compréhension et a pour but d'élaborer les spécifications tandis que la **conception** sert à structurer et à développer une solution.

2.3.2. Pourquoi une méthode ?

Les systèmes d'information ont beau gagné en complexité, les temps de développement n'en sont pas pour autant extensibles. Il faut, dès lors, privilégier l'approche métier, associer utilisateurs et informaticiens, optimiser les ressources et la technologie pour garantir délais et budget. Les méthodes répondent à ces exigences et permettent la construction d'applications fonctionnellement et techniquement conformes aux attentes des divers intervenants du projet.

L'impératif est clair : plus vite, moins cher et de meilleure qualité. Le succès d'un projet dépend désormais de deux facteurs essentiels : l'implication des utilisateurs et une méthode garantissant la réussite du projet tout autant que la qualité de l'application.

Nous devons noter qu'avec les progrès du génie logiciel plusieurs méthodes ont vues le jour. Nous allons, dans le paragraphe suivant, les décrire et les classifier.

2.3.3. Choix méthodologique

La conception d'un système d'information n'est pas évidente car il faut réfléchir à l'ensemble de l'organisation que l'on doit mettre en place. La phase de conception nécessite des méthodes permettant de mettre en place un modèle sur lequel on va s'appuyer. La modélisation consiste à créer une représentation virtuelle d'une réalité de telle façon à faire ressortir les points auxquels on s'intéresse.

Ce type de méthode est appelé m2thode d'analyse. Il existe plusieurs méthodes d'analyse, mais les plus connues sont MERISE ET UML respectivement issue des méthodes systémiques et objets.

2.3.3.1. Etude comparative entre les approches Merise et UML

2.3.3.1.1. Merise

Issue de l'**analyse systémique**, la méthode **MERISE** (Méthode d'Etude de Réalisation Informatique par Sous-Ensemble) est née dans les années 1970, à la demande du ministère de l'industrie, et a surtout été utilisée en France, par les SSII de ses membres fondateurs (Sema Métra, ainsi que par la CGI Informatique) et principalement pour les entreprises, notamment des grandes administrations publiques ou privées.
Pour aider le concepteur dans ces tâches, la méthode Merise - méthode d'analyse, de conception et de réalisation de systèmes d'informations informatisés propose un ensemble de formalismes et de règles destinées à modéliser de manière indépendante les données et les traitements du système d'information. Ces modèles ne sont qu'une base de réflexion pour le concepteur et un moyen de communication entre les divers acteurs du système d'information. Seule la validation de l'ensemble se fera en commun.

Parmi les informations qui appartiennent au système d'information, certaines doivent ou peuvent faire l'objet d'un traitement automatisé grâce aux outils informatiques. Pour assurer la cohérence du système d'information, la méthode Merise propose une démarche d'informatisation comportant les étapes suivantes :

- Le schéma directeur : dont le rôle est de définir, de manière globale, la politique d'organisation et d'automatisation du système d'information. Pour ce faire, il est nécessaire de répertorier l'ensemble des applications informatiques existantes à modifier et à développer. Pour rendre contrôlable et modulable ce développement, il est nécessaire de découper le système d'information en sous-ensembles homogènes et relativement indépendant. Ces sous-ensembles sont appelés domaines. Par exemple, on peut trouver le domaine « Approvisionnement », le domaine « Personnel ». Les résultats attendus à la fin de cette étape sont une définition précise des domaines, une planification du développement de chaque domaine et un plan détaillé, année par année, des applications qui doivent être réalisées.
- L'étude préalable par domaine : qui doit aboutir à une présentation générale du futur système de gestion (modèles des données et des traitements) en indiquant les principales novations par rapport au système actuel, les moyens matériels à mettre en œuvre, les bilans coût – avantage. Cette étude est réalisée en 4 phases :
    - Une phase de recueil qui a pour objectif d'analyser l'existant afin de cerner les dysfonctionnements et les obsolescences les plus frappantes du système actuel.
    - Une phase de conception qui a pour objectif de formaliser et hiérarchiser les orientations nouvelles en fonction des critiques formulées sur le système actuel et d'autre part des politiques et des objectifs de la direction générale. Cela revient à modéliser le futur système avec une vue pertinente de l'ensemble.
    - Une phase d'organisation dont l'objectif est de définir le système futur au niveau organisationnel : qui fait quoi ?
    - Une phase d'appréciation dont le rôle est d'établir les coûts et les délais des solutions définies ainsi que d'organiser la mise en œuvre de la réalisation. A cet effet un découpage en projets est effectué
- L'étude détaillée par projet qui consiste d'une part à affiner les solutions conçues lors de l'étude préalable et d'autre part à rédiger, pour chaque procédure à mettre en œuvre, un dossier de spécifications détaillé décrivant les supports (maquettes d'états ou d'écran) ainsi que les algorithmes associés aux règles de gestion… A l'issue de cette étude, il est possible de définir le cahier des charges utilisateurs qui constitue la base de l'engagement que prend le concepteur vis à vis des utilisateurs.

Le fonctionnement détaillé du futur système, du point de vue de l'utilisateur, y est entièrement spécifié.

- La réalisation dont l'objectif est l'obtention des programmes fonctionnant sur un jeu d'essais approuvés par les utilisateurs.
- La mise en œuvre qui se traduit par un changement de responsabilité : l'équipe de réalisation va en effet transférer la responsabilité du produit à l'utilisateur. Cette étape intègre en particulier la formation des utilisateurs. Après une période d'exploitation de quelques mois, la recette définitive de l'application est prononcée.
- La maintenance qui consiste à faire évoluer les applications en fonction des besoins des utilisateurs, de l'environnement et des progrès technologique

Cette démarche lourde et parfois complexe est adaptée à l'automatisation de « gros 5 systèmes d'information ». Pour des informatisations plus modestes, elle peut être perçue comme un carcan, et il convient donc de l'adapter afin de retenir uniquement les concepts et/ou les étapes appropriées aux besoins.

2.3.3.1.2. UML

UML (en anglais Unified Modeling Language ou « langage de modélisation unifié ») est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

UML est l'accomplissement de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jackobson, UML est à présent un standard défini par l'Objet Management Group (OMG). La dernière version diffusée par l'OMG est UML 2.3 depuis mai 2010.

UML est avant tout un support de communication performant, qui facilite la représentation et la compréhension de solutions objet :

- Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions.
- L'aspect formel de sa notation, limite les ambiguïtés et les incompréhensions.

- Son indépendance par rapport aux langages de programmation, aux domaines d'application et aux processus, en font un langage universel.

Petit aparté :
La notation graphique d'UML n'est que le support du langage. La véritable force d'UML, c'est qu'il repose sur un métamodèle. En d'autres termes : la puissance et l'intérêt d'UML, c'est qu'il normalise sémantique des concepts qu'il véhicule !
Qu'une association d'héritage entre deux classes soit représentée par une flèche terminée par un triangle ou un cercle, n'a que peu d'importance par rapport au sens que cela donne à votre modèle. La notation graphique est essentiellement guidée par des considérations esthétiques, même si elle a été pensée dans ses moindres détails.

Par contre, utiliser une relation d'héritage, reflète l'intention de donner à votre modèle un sens particulier. Un "bon" langage de modélisation doit permettre à n'importe qui de déchiffrer cette intention de manière non équivoque ! Il est donc primordial de s'accorder sur la sémantique des éléments de modélisation, bien avant de s'intéresser à la manière de les représenter. Le métamodèle UML apporte une solution à ce problème fondamental.

UML est donc bien plus qu'un simple outil qui permet de "dessiner" des représentations mentales... Il permet de parler un langage commun, normalisé mais accessible, car visuel. Il représente un juste milieu entre langage mathématique et naturel, pas trop complexe mais suffisamment rigoureux, car basé sur un métamodèle.
UML comme cadre d'une analyse objet :

Une autre caractéristique importante d'UML, est qu'il cadre l'analyse. UML permet de représenter un système selon différentes vues complémentaires : les diagrammes. Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle ; c'est une perspective du modèle.

Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis) et véhicule une sémantique précise (il offre toujours la même vue d'un système).
Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système. Les diagrammes permettent donc d'inspecter un modèle selon différentes perspectives et guident l'utilisation des éléments de modélisation (les concepts objet), car ils possèdent une structure.

Une caractéristique importante des diagrammes UML, est qu'ils supportent l'abstraction. Cela permet de mieux contrôler la complexité dans l'expression et l'élaboration des solutions objet.

UML opte en effet pour l'élaboration des modèles, plutôt que pour une approche qui impose une barrière stricte entre analyse et conception. Les modèles d'analyse et de conception ne diffèrent que par leur niveau de détail, il n'y a pas de différence dans les concepts utilisés. UML n'introduit pas d'éléments de modélisation propres à une activité (analyse, conception...) ; le langage reste le même à tous les niveaux d'abstraction.

Cette approche simplificatrice facilite le passage entre les niveaux d'abstraction. L'élaboration encourage une approche non linéaire, les "retours en arrière" entre niveaux d'abstraction différents sont facilités et la traçabilité entre modèles de niveaux différents est assurée par l'unicité du langage.

UML favorise donc le prototypage, et c'est là une de ses forces. En effet, modéliser une application n'est pas une activité linéaire. Il s'agit d'une tâche très complexe, qui nécessite une approche itérative, car il est plus efficace de construire et valider par étapes, ce qui est difficile à cerner et maîtriser UML permet donc non seulement de représenter et de manipuler les concepts objet, il sous-entend une démarche d'analyse qui permet de concevoir une solution objet de manière itérative, grâce aux diagrammes, qui supportent l'abstraction

Comme UML n'impose pas de méthode de travail particulière, il peut être intégré à n'importe quel processus de développement logiciel de manière transparente. UML est une sorte de boîte à outils, qui permet d'améliorer progressivement vos méthodes de travail, tout en préservant vos modes de fonctionnement.

Intégrer UML par étapes dans un processus, de manière pragmatique, est tout à fait possible. La faculté d'UML de se fondre dans le processus courant, tout en véhiculant une démarche méthodologique, facilite son intégration et limite de nombreux risques (rejet des utilisateurs, coûts...)

2.3.3.1.3. Comparaison

2.3.3.1.3.1. Niveau d’abstraction

L’approche Merise : Le cycle d'abstraction permet de sérier les niveaux de préoccupations lors de la description ou de l'analyse du système. Les trois niveaux

retenus correspondent à des degrés de stabilité et d'invariance de moins en moins élevés.

- Le niveau conceptuel,
- Le niveau logique
- Le niveau physique.

L'approche UML propose différentes notions (cas d'utilisation, paquetage, classe, composant, nœud) et différents diagrammes pour modéliser le système aux différents niveaux d'abstraction.

2.3.3.1.3.2. Approche fonctionnelle

L'approche Merise propose une approche descendante où le système réel est décomposé en activités, elles-mêmes déclinées en fonctions. Les fonctions sont composées de règles de gestion, elles-mêmes regroupées en opérations. Ces règles de gestion au niveau conceptuel génèrent des modules décomposés en modules plus simples et ainsi de suite jusqu'à obtenir des modules élémentaires... Les limites d'une telle approche résident dans le fait que les modules sont difficilement extensibles et exploitables pour de nouveaux systèmes.

L'approche UML : Les fonctions cèdent la place aux cas d'utilisation qui permettent de situer les besoins de l'utilisateur dans le contexte réel. A chaque scénario correspond des diagrammes d'interaction entre les objets du système et non pas un diagramme de fonction.

2.3.3.1.3.3. Dualité des données de traitements

L'approche Merise propose de considérer le système réel selon deux points de vue : un point de vue statique (les données), un point de vue dynamique (les traitements). Il s'agit d'avoir une vision duale du système réel pour bénéficier de l'impression de relief qui en résulte, et donc consolider et valider le système final. L'approche UML : L'approche objet associe les informations et les traitements. De cette façon, elle assure un certain niveau de cohérence.

2.3.3.1.3.4. Tableaux comparatifs

Merise	UML
Méthode d'analyse et de conception de système d'information.	Langage de représentation d'un système d'information.
Méthode de modélisation de données et traitements orienté bases de données relationnelles.	Système de notation orienté objet.
Relationnel	Objet
Franco-français	International
Schéma directeur, étude préalable, étude détaillée et la réalisation.	Langage de modélisation des systèmes standard, qui utilise des diagrammes pour représenter chaque aspect d'un système : statique, dynamique, etc.
Plus adapté à une approche théorique	Plus orientée vers la conception
Du "bottom up" de la base de données vers le code	Du "top down" du modèle vers la base de données.

En conclusion, Merise et UML sont techniquement complémentaires. Si l'on considère que le SI est modélisable comme deux sous-systèmes inclus l'un dans l'autre : le SIO (système d'information organisationnel) représentant le métier, englobant le SII (système d'information informatisé) représentant l'application informatique associée. Merise est adaptée au SIO, UML au SII. Maintenant, selon le positionnement de chacun (consultant, analyste métier, concepteur de logiciel, développeur), on s'appuiera plus ou moins sur chaque partie de méthode.

Selon l'étude effectuée, la méthode la plus convenable pour la modélisation des bases de données biométriques, serait la **méthode UML**.

2.3.3.2. Choix de l'approche UML

La modélisation est incontournable pour permettre aux différents acteurs de coopérer et de dialoguer efficacement. Ainsi Il est donc important de connaître le langage et les techniques de modélisation et de savoir quels modèles sont les plus appropriés dans chaque situation. En effet, de la même manière qu'il n'est pas judicieux d'utiliser tous les éléments de la langue française pour écrire un article, il n'est pas non plus obligatoire d'utiliser tous les concepts du langage UML dans un projet.

Nous allons, dans les paragraphes suivants, présenter l'intérêt d'utiliser UML dans la conception d'un SIG ainsi que les différents diagrammes utilisés par ce langage.

2.3.3.2.1. Pourquoi UML ?

Nombreuses sont les tentatives d'application des méthodes de conception des systèmes d'information pour la mise en œuvre des systèmes d'information géographique. Les raisons les plus significatives de leur inefficacité dans la conception des systèmes d'information géographique sont entre autres les suivantes.

- Ces méthodes reposent sur deux approches distinctes, privilégiant soit les données, soit les traitements. Or le fonctionnement d'un système d'information géographique repose justement sur une interdépendance étroite entre les données et les traitements.
- L'information géographique s'organise hiérarchiquement et les traitements qui sont appliqués sont souvent complexes et reposent aussi sur des processus d'agrégation de l'information. Or les méthodes proposées par les systèmes d'information d'entreprise ne prennent en compte que des traitements simples de type flux ou échange de données.
- L'utilisation des méthodes de conception des SI présuppose que soient, au préalable, clairement identifiés les besoins des applications et que l'on ait la maîtrise de leur évolution. Or en matière d'analyse spatiale, de gestion et de planification territoriale, les besoins s'identifient le plus souvent en fonction des données dont on dispose, des résultats issus des traitements réalisés, et de l'évolution des requêtes adressées par les utilisateurs. Il est certain que la

modélisation UML ayant été élaborée pour répondre aux besoins des systèmes d'information classiques n'est donc pas conçue, a priori, pour répondre aux spécificités des systèmes intégrant de l'information géographique Ces derniers gérant à la fois des données graphiques et des données non graphiques sont considérés comme un cas particulier des SI.

Toutefois, il semble qu'a priori les principaux concepts proposés par UML soient pertinents pour l'analyse et la conception des systèmes de gestion et d'analyse des territoires. Cette approche peut constituer un support intéressant en termes d'acquisition des connaissances, de structuration de l'information géographique à intégrer dans l'outil à concevoir, et de spécification des fonctionnalités de l'outil.

2.3.3.2.2. Diagrammes structurels (les vues statiques)

Ces diagrammes permettent de visualiser, spécifier, construire et documenter l'aspect statique ou structurel du système d'information. Il s'agit outre les diagrammes de cas d'utilisation, des diagrammes de classes, d'objets, mais aussi de déploiement et de composants

2.3.3.2.2.1. Diagramme de cas d'utilisation

Les cas d'utilisation sont une technique de description du système étudié privilégiant le point de vue de l'utilisateur. Ils décrivent sous la forme d'actions et de réactions, le comportement d'un système et servent, par conséquent, à structurer les besoins des utilisateurs et les objectifs correspondants du système.

La détermination et la compréhension des besoins sont souvent difficiles car les intervenants sont noyés sous de trop grandes quantités d'informations. Or, comment mener à bien un projet si l'on ne sait pas où l'on va ?

Les uses cases ne doivent pas chercher l'exhaustivité, mais clarifier, filtrer et organiser les besoins. Ils ne doivent donc en aucun cas décrire des solutions d'implémentation. Leur but est justement d'éviter de tomber dans la dérive d'une approche fonctionnelle, où l'on liste une litanie de fonctions que le système doit réaliser.

2.3.3.2.2.2. Diagramme d'objets

Ce type de diagramme UML montre des objets (instances de classes dans un état particulier) et des liens (relations sémantiques) entre ces objets. Les diagrammes d'objets s'utilisent pour montrer un contexte (avant ou après une interaction entre objets par exemple). Il sert essentiellement en phase exploratoire, car il possède un très haut niveau d'abstraction.

2.3.3.2.2.3. Les diagrammes de classes

Les diagrammes de classes expriment de manière générale la structure statique d'un système, en termes de classes et de relations entre ces classes.

Une classe permet de décrire un ensemble d'objets (attributs et comportement), tandis qu'une relation ou association permet de faire apparaître des liens entre ces objets.

2.3.3.2.2.4. Les diagrammes de paquetage

Les paquetages sont des éléments d'organisation des modèles. Ils regroupent des éléments de modélisation, selon des critères purement logiques. Ils permettent d'encapsuler des éléments de modélisation et de structurer un système en catégories (vue logique) et sous-systèmes (vue des composants). Ils servent de « briques » de base dans la construction d'une architecture.

2.3.3.2.2.5. Diagramme de composant et de déploiement

Les diagrammes de composants permettent de décrire l'architecture physique et statique d'une application en termes de modules : fichiers sources, librairies, exécutables, etc. Ils montrent la mise en œuvre physique des modèles de la vue logique avec l'environnement de développement.

Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels. Les ressources matérielles sont représentées sous forme de nœuds, connectés entre eux à l'aide d'un support de communication

2.3.3.2.3. Diagrammes comportementaux

Ils modélisent les aspects dynamiques du système, c'est à dire les différents éléments qui sont susceptibles de subir des modifications. Parmi eux on distingue, les diagrammes de séquence, de collaboration, d'états - transitions et d'activités.

Ils représentent la dynamique du système, à savoir, non seulement les interactions entre le système lui-même et les différents acteurs du système, mais aussi, la façon dont les différents objets contenus dans le système communiquent entre eux.

2.3.3.2.3.1. Diagramme de séquence

Les diagrammes de séquences permettent de des collaborations entre objets selon un point de vue temporel, on y met l'accent sur la chronologie des envois de messages. Les diagrammes de séquences peuvent servir à illustrer un cas d'utilisation

2.3.3.2.3.2. Diagramme de collaboration

Les diagrammes de collaboration montrent des interactions entre objets (instances de classes et acteurs). Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent.

2.3.3.2.3.3. Diagramme d'états-transitions

Ces diagrammes servent à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions. Ils permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.

Un état se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des valeurs des attributs d'un objet.

Une transition représente le passage instantané d'un état vers un autre. Une transition est déclenchée par un événement. En d'autres termes : c'est l'arrivée d'un événement qui conditionne la transition. Les transitions peuvent aussi être automatiques, lorsqu'on ne spécifie pas l'événement qui la déclenche. En plus de spécifier un événement précis, il est aussi possible de conditionner une transition, à l'aide de "gardes" : il s'agit d'expressions booléennes, exprimées en langage naturel (et encadrées de crochets).

2.3.3.2.3.4. Diagramme d'activités

UML permet de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation, à l'aide de diagrammes d'activités (une variante des diagrammes d'états-transitions). Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles. Le passage d'une activité vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre (elles sont automatiques).

En théorie, tous les mécanismes dynamiques pourraient être décrits par un diagramme d'activités, mais seuls les mécanismes complexes ou intéressants méritent d'être représentés.

2.3.3.2.4. Points fort et point faible d'UML

2.3.3.2.4.1.1. *Les* points forts d'UML

UML est un langage formel et normalisé :

- Gain de précision
- Gage de stabilité
- Encourage l'utilisation d'outils
- UML est un support de communication performant :
- Il cadre l'analyse.
- Il facilite la compréhension de représentations abstraites complexes. Son caractère polyvalent et sa souplesse en font un langage universel.

2.3.3.2.4.2. Les points faibles d'UML

- La mise en pratique d'UML nécessite un apprentissage et passe par une période d'adaptation ;
- Le processus

Le langage UML, offre ainsi de nombreux avantages pour l'analyse et la conception d'un système d'information géographique. Par conséquent, nous combinerons UML au Processus Unifié pour mener à terme notre système d'information.

2ème partie : Le cadre analytique

3. Chapitre 3 : Analyse de l'existant

3.1. Analyse des besoins

Dans cette section, nous allons étudiés les besoins fonctionnels, et non fonctionnels (techniques)

La spécification de besoins constitue la phase de départ de toute application à développer dans laquelle nous allons identifier les différents besoins. Elle doit décrire sans ambiguïté le logiciel à développer. Elle est constituée d'un ensemble de documents et de modèles.

Toutes les personnes impliquées dans le projet doivent avoir accès à la spécification des besoins.

Nous distinguons deux types de besoins :

1. Les besoins fonctionnels qui conduisent à l'élaboration des cas d'utilisation, ils présentent les fonctionnalités attendues de notre application.
2. Les besoins techniques (non fonctionnels) qui aboutissent à la rédaction d'une des exigences de système pour sa réalisation et son bon fonctionnement, ils permettent d'éviter le développement d'une application non satisfaisante.

3.1.1. Spécification des besoins fonctionnels

Les besoins fonctionnels ou besoin métiers représentent les actions que le système doit exécuter, il ne devient opérationnel que s'il les satisfait. Ils expriment une action que doit effectuer le système en réponse à une demande (sorties qui sont produites pour un ensemble donné d'entrées).

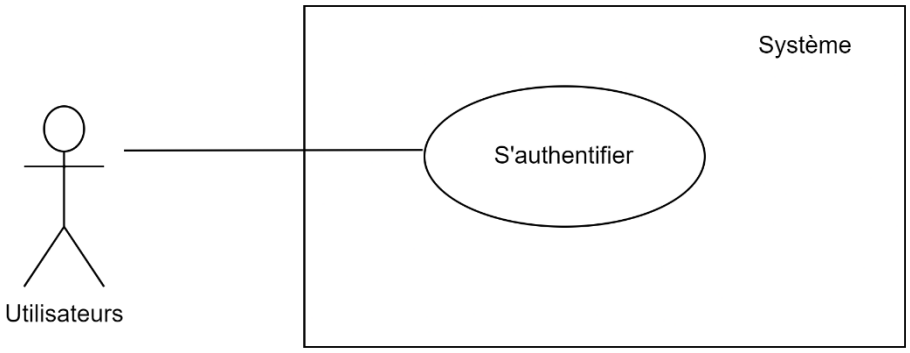
L'application que nous allons développer devra regrouper les fonctionnalités qui permettront à l'utilisateur de :

- S'authentifier
- Gérer les élèves
- Gérer les classes
- Gérer les matières
- Gérer les enseignants
- Gérer les notes et bulletins
- Gérer les inscriptions et les paiements
- Avoir une base de données pour le stockage des étudiants et des enseignants.
- Manipuler et mettre à jour la base de données

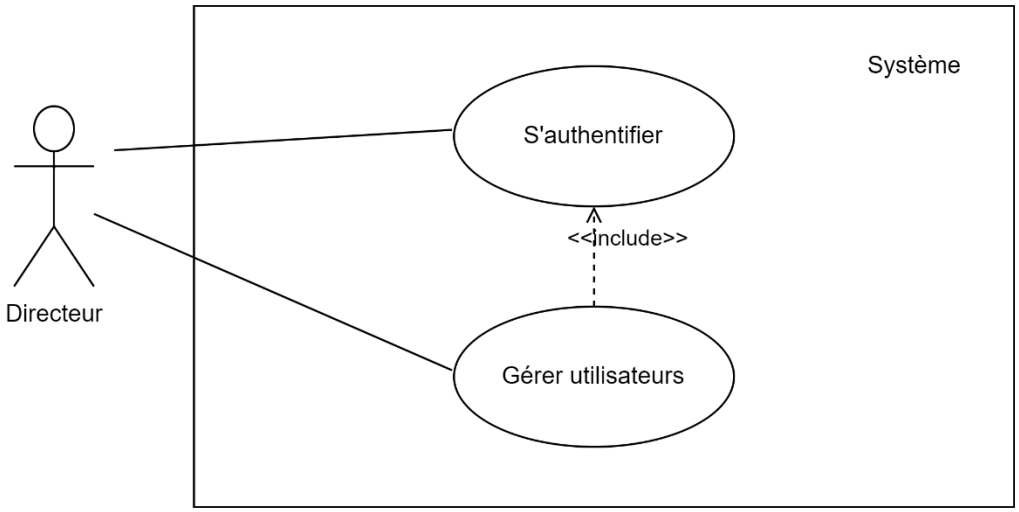
Nous allons essayer d'illustrer ces besoins fonctionnels à travers quelque diagrammes UML (analyse).

3.1.1.1. Diagramme de cas d'utilisation (DCU)

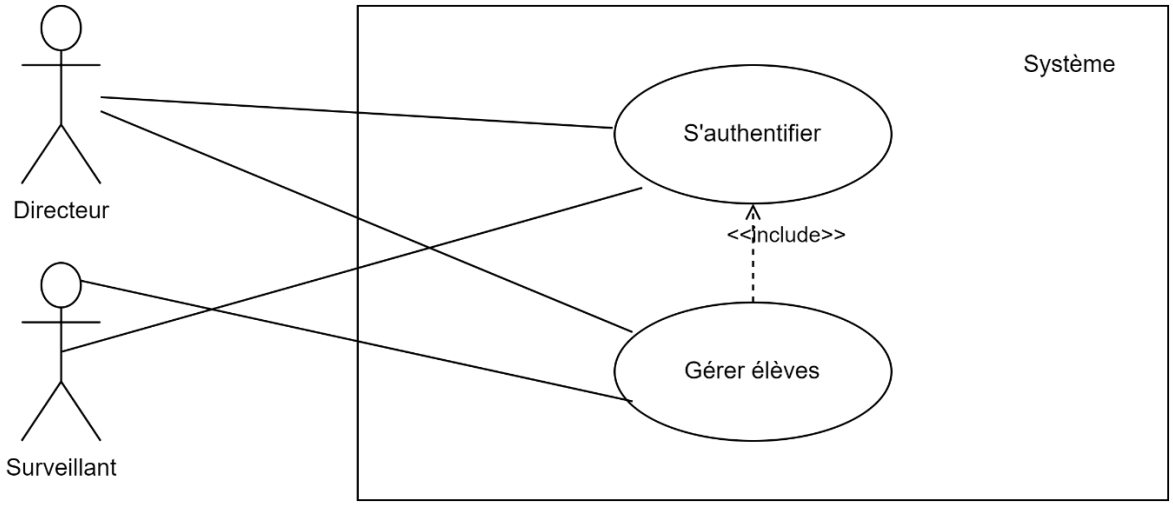
3.1.1.1.1. Gérer authentification



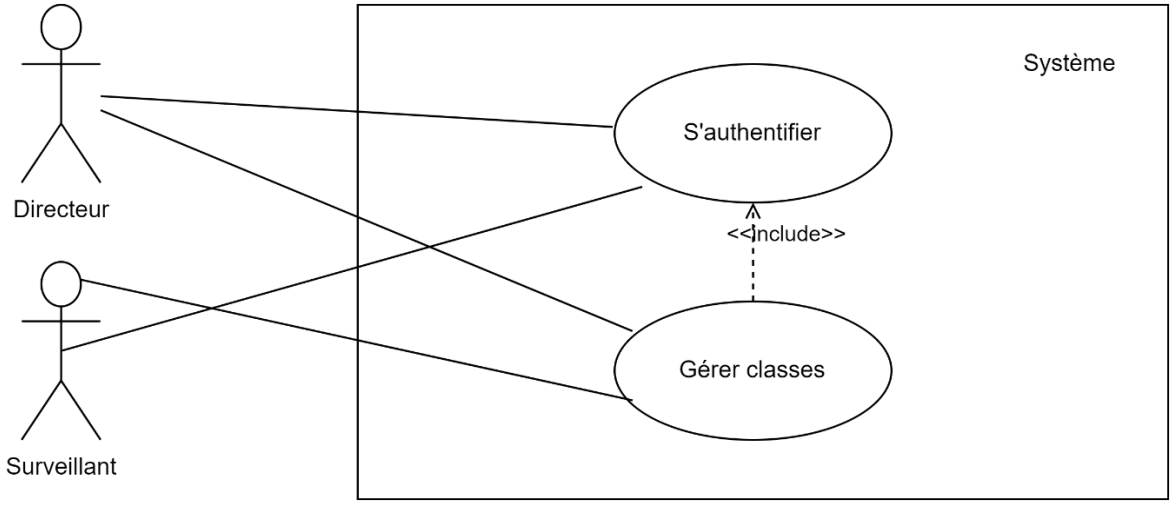
3.1.1.1.2. Gérer utilisateurs



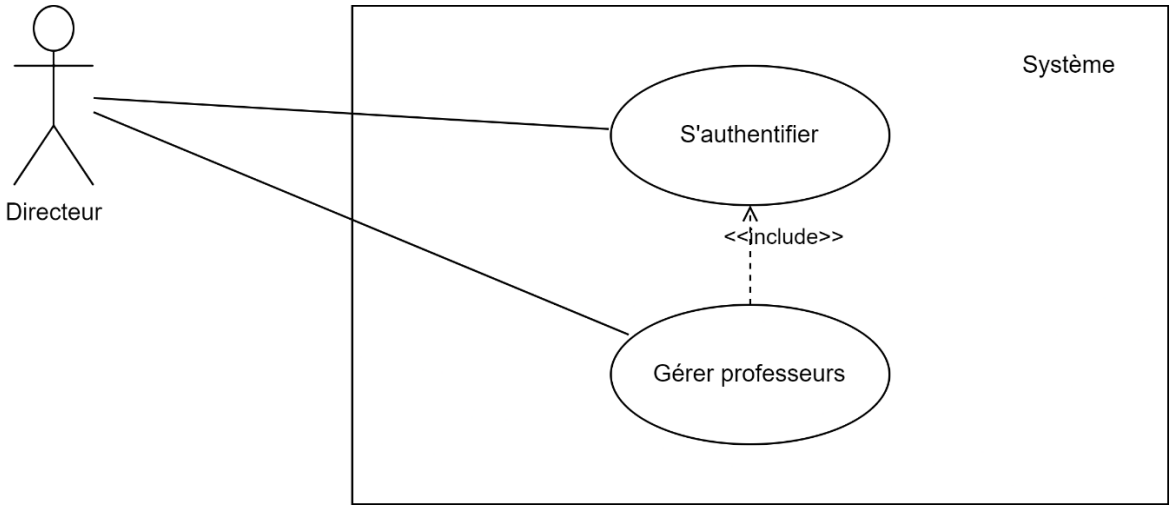
3.1.1.1.3. Gérer élèves



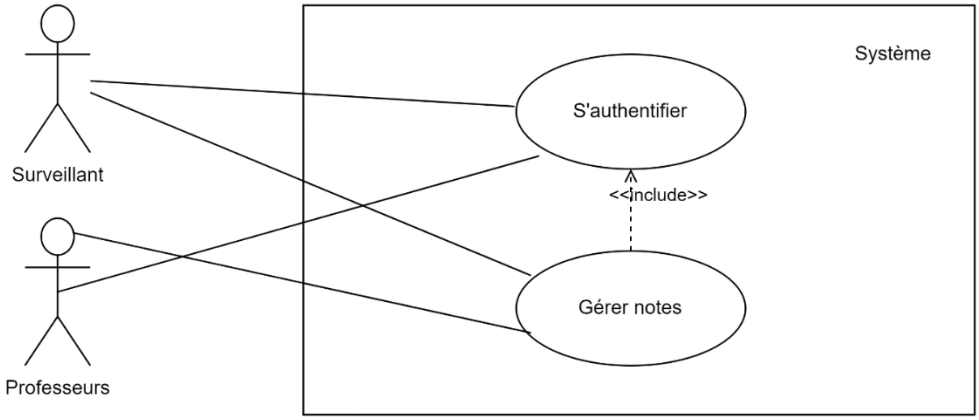
3.1.1.1.4. Gérer classe



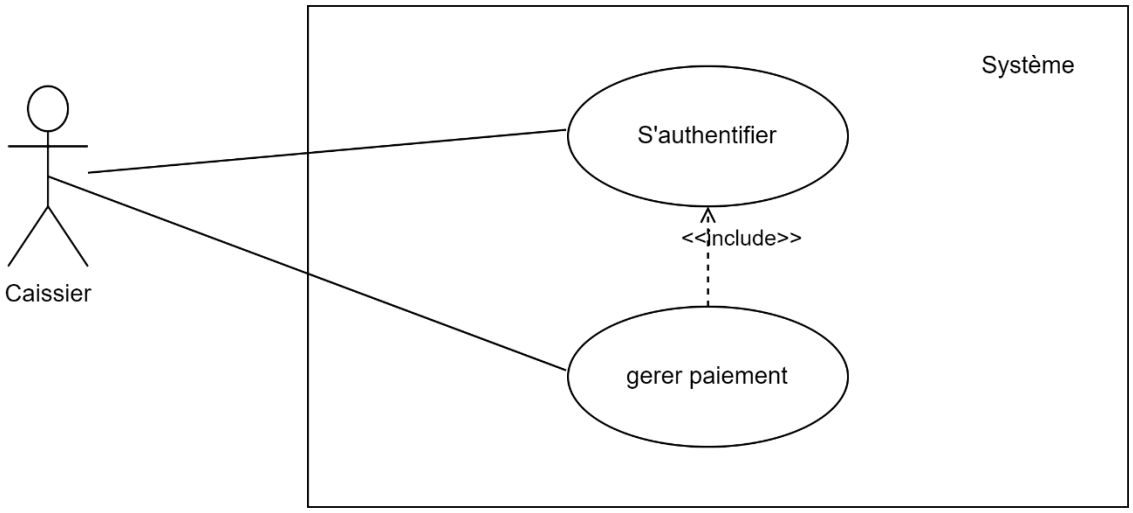
3.1.1.1.5. Gérer professeurs



3.1.1.1.6. Gérer notes



3.1.1.1.7. Gérer paiements



3.1.1.2. Diagramme de séquences (DSS)

Un diagramme de séquence système (DSS) permet de décrire le comportement du système vu de l'extérieur (par les acteurs) sans préjuger de comment il sera réalisé. Il permet donc de spécifier la nature exacte des échanges entre le système et les acteurs au cours de la réalisation des cas d'utilisation. Il oppose le système (une ligne de vie) aux acteurs (une ligne de vie par acteur qui interagit)

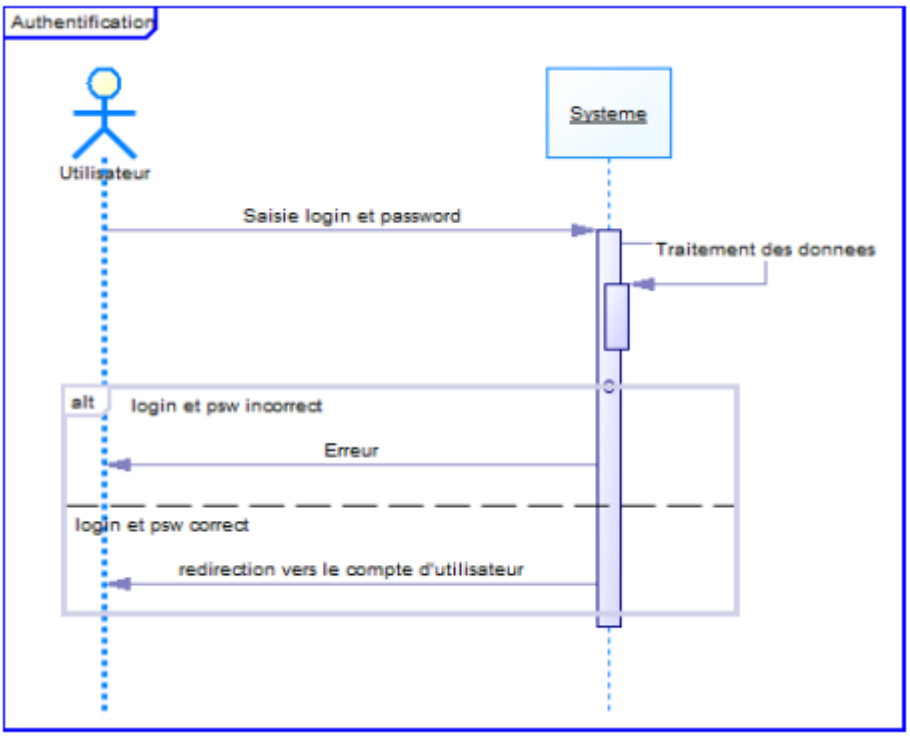
Le système est vu comme une « boîte noire » qui sera ouverte (décrite) seulement en conception.

3.1.1.2.1. Cas de l'authentification

Scenario :

- Saisie le login et mot de passe.
- Envoyer login et mot de passe.
- Traitement des informations envoyées.
- En cas d'erreur, l'authentification est rejetée.

Si non, redirection d'utilisateur vers ses rôles

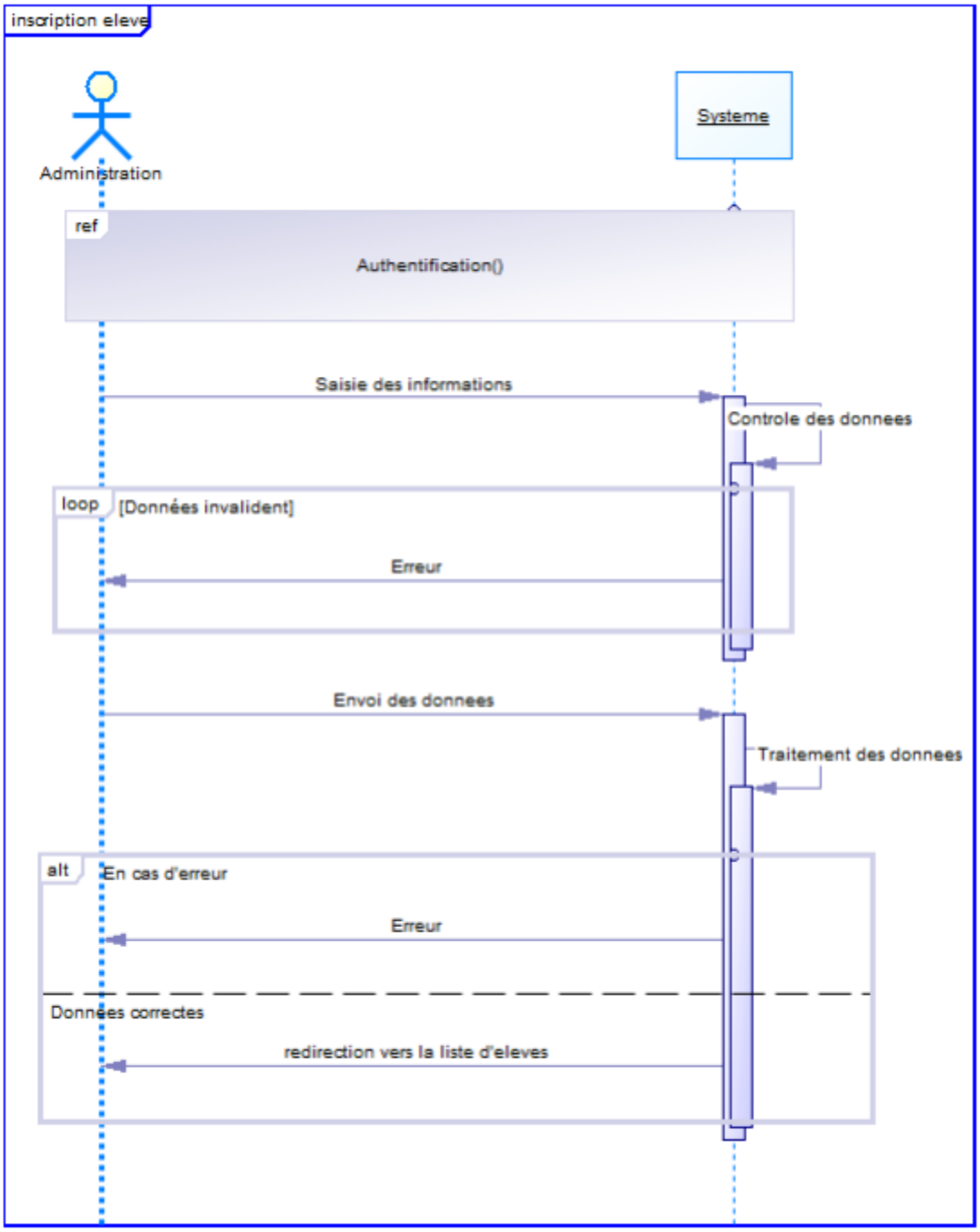


3.1.1.2.2. Cas de l'Inscription d'un élève

Scenario :

- Saisie les informations de l'élève.
- Contrôle des données en temps réel (matricule – cne – cin) en cas de duplication.
- Validation de la saisie.
- Traitement des informations envoyé.
- En cas d'une anomalie, l'inscription est rejetée en précisant l'erreur effectuée.

Si non, l'inscription est effectuée avec succès avec redirection d'utilisateur vers la liste d'élèves

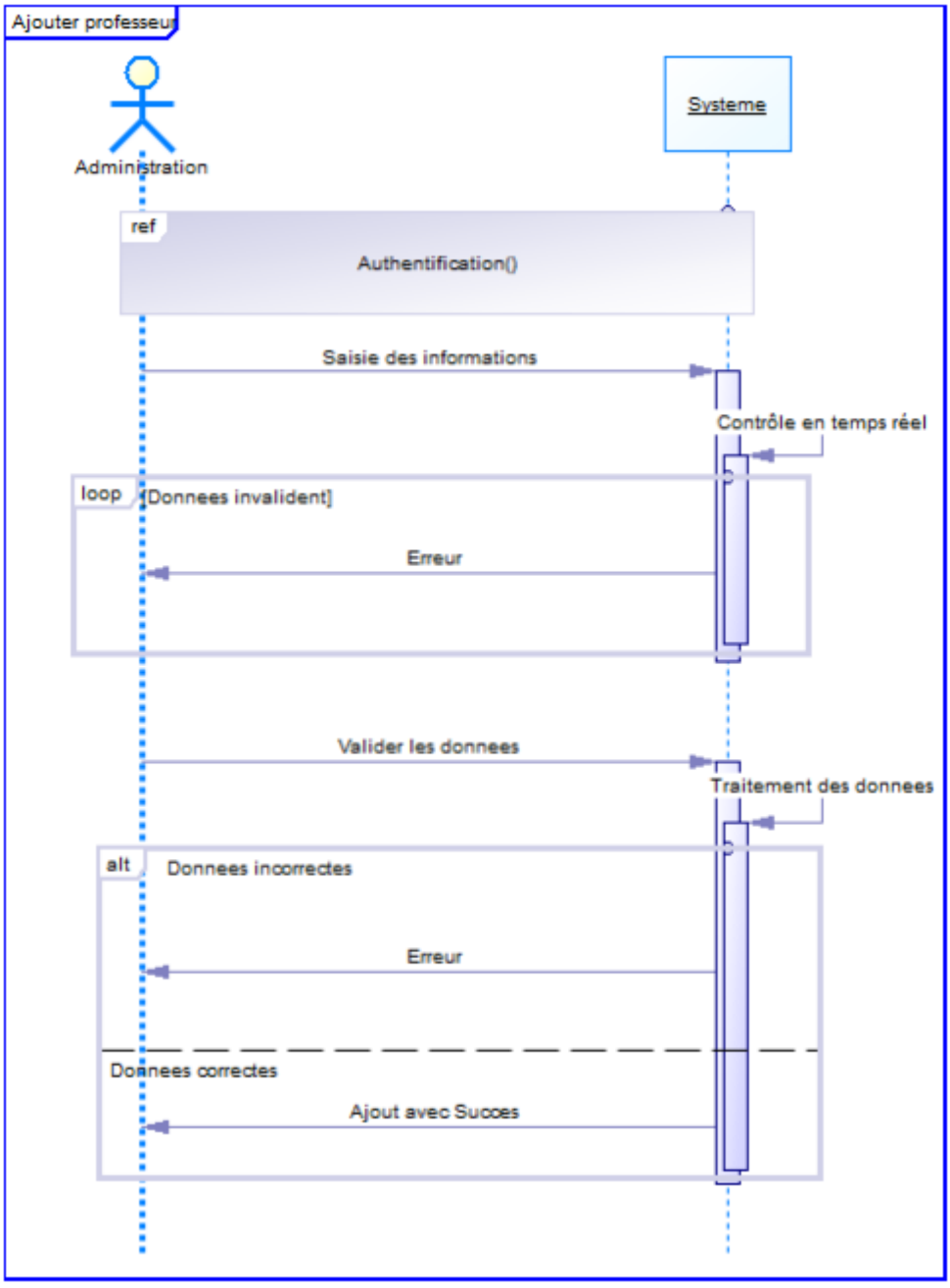


3.1.1.2.3. Cas de l'Ajout d'un professeur

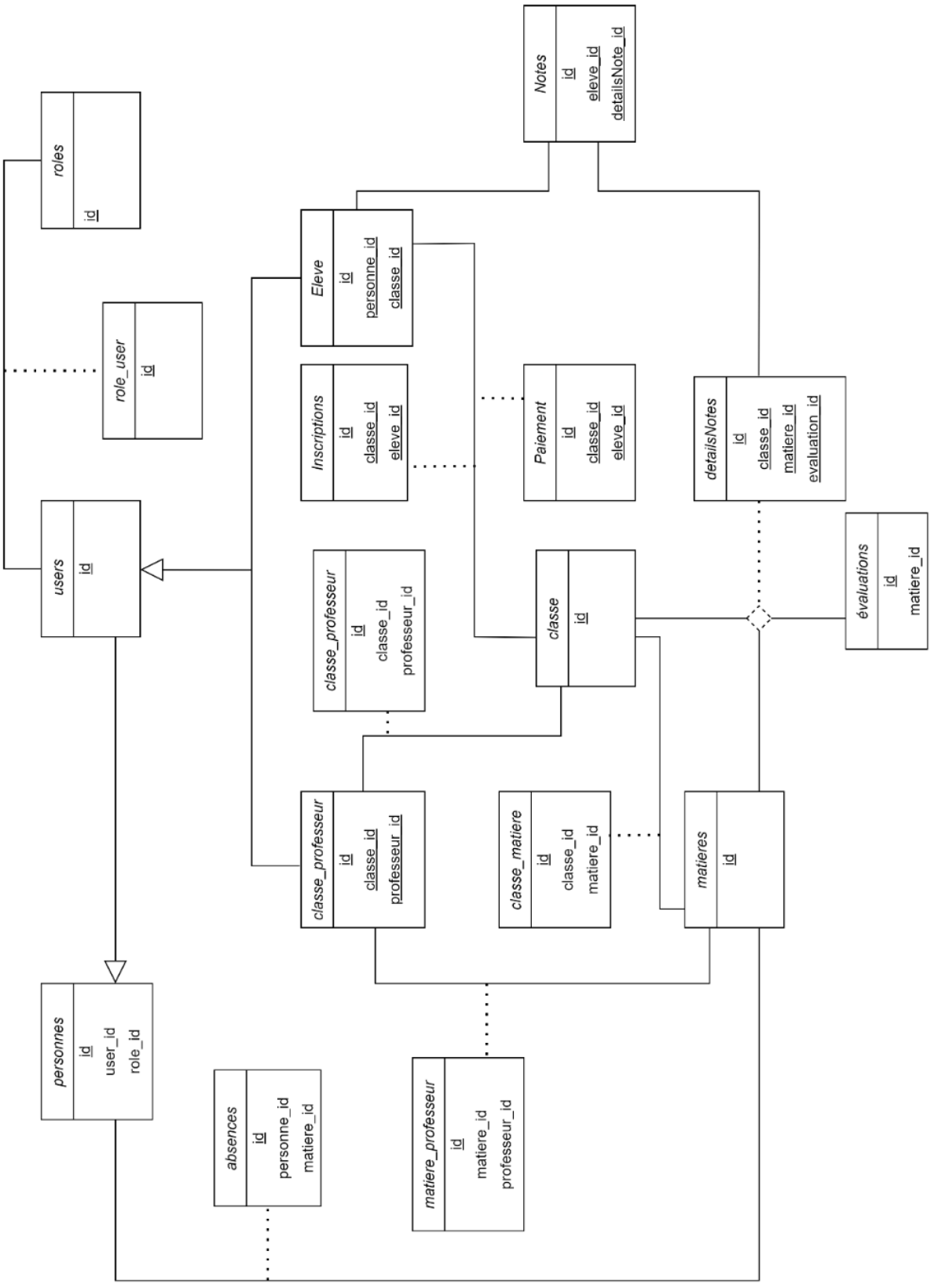
Scenario :

- Saisie des informations concernant le professeur.
- Contrôle des données en temps réel, en cas de duplication.
- Validation de la saisie.
- Traitement des informations envoyées.

- En cas d'une anomalie, l'ajout est rejeté en précisant l'erreur effectuée.
- Si non, l'ajout est effectué avec succès avec redirection d'utilisateur vers la liste des professeurs



3.1.1.3. Diagramme de classe (conception)



3.1.2. Spécification des besoins non fonctionnels (ou technique)

Les besoins techniques ou non fonctionnels sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt ils identifient les contraintes internes et externes du système. Les principaux besoins non fonctionnels de notre application sont des besoins de :

3.1.2.1. Sécurité

Ils définissent les niveaux d'accès possibles au système pour les utilisateurs. Dans notre cas, cette application doit avoir un niveau de sécurité assez élevé, les comptes des utilisateurs devront être sécurisés par des mots de passe. Ces mots de passe seront individuels et devront respecter certaines conditions (la longueur du code ; le code système, l'expiration de session, etc.). Les besoins de sécurités sont satisfaits par Laravel qui utilise un algorithme de cryptage évolué.

3.1.2.2. Disponibilité

Ils concernent le niveau de disponibilité qui doit être explicitement défini pour les applications critiques (Exemple : exigence de disponibilité 24h/24, 7j/7 sauf période de maintenance, à spécifier). Cette application devra fonctionner de manière efficace et ceux, sans défaillance. Les utilisateurs pourront compter sur sa fiabilité.
Ce besoin de disponibilité est satisfait grâce à l'hébergeur.

3.1.2.3. Performance

Ils décrivent les performances d'exécution du système, généralement en matière de temps de réponse (l'exemple d'une application Web), temps de chargement d'une page : le chargement d'une page Web dans le navigateur ne devrait pas prendre plus de 15 secondes en condition normale. Cette application devra répondre aux exigences des utilisateurs de façon optimale c'est-à-dire effectuer des opérations dans un laps de temps très court.
Ce besoin est garanti par le Framework.

3.1.2.4. Portabilité

Cette application sera multiplateforme. Elle fonctionnera sur tous les systèmes d'exploitation et tout type de terminal puisqu'il s'agit d'une application web, elle sera disponible sur tout support où il existe un navigateur. Aussi, cette application sera responsive c'est-à-dire qu'elle s'adaptera à la taille de l'écran. Cette responsivité est

assurée par Bootstrap. On peut affirmer que ce besoin est déjà satisfait par la définition d'une application web porté sur un module de cet ensemble, on peut affirmer que notre application répond au critère d'une solution ouverte et évoluée.

3.1.2.5. Ergonomie

L'ergonomie de l'application doit favoriser la lisibilité et l'accessibilité des informations, ainsi que leur réutilisation.
L'interface doit être conviviale, facile à manipuler par des utilisateurs simples (non informaticiens).

3.1.2.6. Administration

Le système doit permettre aux administrateurs de gérer les utilisateurs et groupes, leurs rôles et fonctions par module, et également de surveiller et gérer la liste des utilisateurs en ligne.

3.1.2.7. Intégrité

La solution doit avoir une interface d'authentification unique.
Tous les modules font partie de la même solution avec le principe de saisie unique.
L'accès aux fonctions du système et aux données sera permis seulement à base d'authentifications par nom d'utilisateur et mot de passe.
Les utilisateurs, les groupes d'utilisateurs et les fonctions doivent être gérés par les administrateurs à l'aide des interfaces appropriées.

3.2. Critique de l'existant

Lors de l'étude que nous avons faite au sein de l'établissement, nous avons relevé les problèmes suivants :

- Les données sont stockées dans des fichiers Excel, ce qui augmente le risque de perte d'informations (virus, absence de mécanismes de sauvegarde/restauration etc.),
- L'information est décentralisée et dispersée sur plusieurs fichiers et qui cause le problème de réplication et de redondance,

- Perte de temps liés à la re-saisie des données chaque fois. Une fois un de ces fichiers est mis à jour, impérativement les autres fichiers devront être modifiés pour garder l'intégrité des données,
- La complexité de la tâche du responsable qui doit vérifier tout au long de son travail si l'enseignant a atteint son du ou non (même chose pour les éléments d'enseignement) et de calculer les dus en heures de TD convertis,
- Le responsable devra obligatoirement maîtriser l'outil Excel, sinon il aura de grands problèmes, et il risque de ne pas être efficace dans son travail.
- Les soucis lies à la mise à jour des absences et des retards.

3.3. Solution proposée

Suite aux inconvénients cités i dessus, nous proposons la mise en place d'une solution qui automatise les différentes activités de l'école à l'instar de la gestion des ressources humaines (élèves, professeurs, etc.) et matérielles (salles, équipements, etc.) de l'école ainsi que la gestion inscriptions/paiements, des notes, etc. Cette application aura plusieurs apports pour l'école aussi bien sur le plan technique que sur le plan fonctionnel.

3.3.1. Apport sur le plan technique

Sur le plan technique, ce projet permet de centraliser les données dans un seul endroit (base de données unique) qui sera partagée par tous les modules de l'application. Donc la donnée sera saisie une seule fois et accessible pour tous les services de l'école.

Cette application permet aussi d'assurer la sécurité des données et leur fiabilité.

3.3.2. Apport sur le plan fonctionnel

Sur le plan fonctionnel, ce projet apporte deux avantages principaux. Le premier c'est le gain de temps relatif au traitement des données, et le second c'est la simplification de la tâche du responsable.

3ème partie : Le cadre conceptuel

4. Chapitre 4 : Point de vue fonctionnel

4.1. Notions de diagramme des cas d'utilisation (niveau conception)

Un cas d'utilisation représente un ensemble de séquences d'actions à réaliser par le système et produisant un résultat observable intéressant pour un acteur particulier représenté par des ellipses et limité par un rectangle pour représenter le système.

Le diagramme des cas d'utilisations identifie les fonctionnalités fournies par le système, les utilisateurs qui interagissent avec le système (acteurs), et les interactions entre ces derniers.

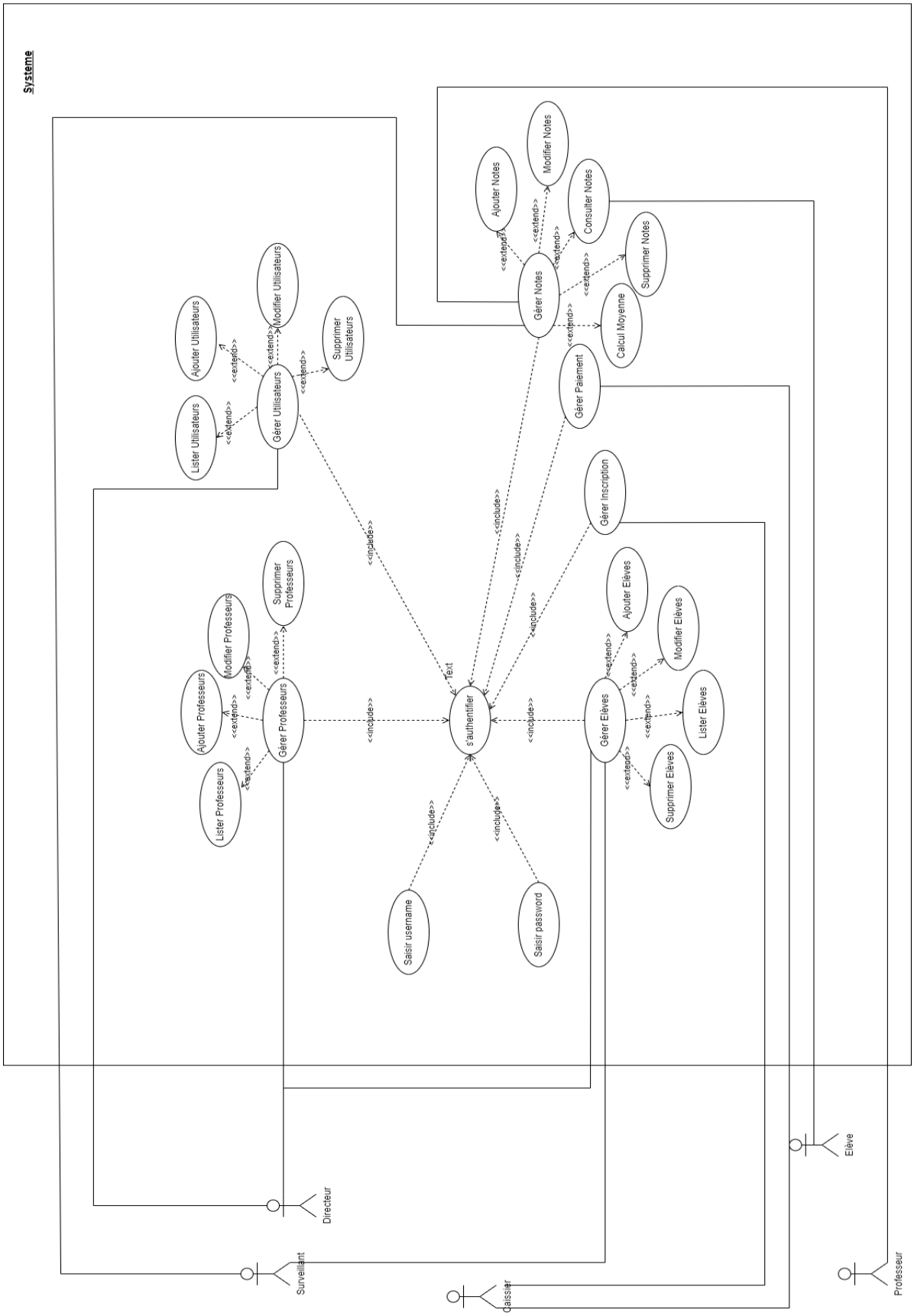
Il a pour but de donner une vision globale sur les interfaces de notre future application. Le diagramme de cas d'utilisation est constitué d'un ensemble d'acteurs qui agissent sur des cas d'utilisations et qui décrivent sous la forme d'actions et de réactions, le comportement d'un système du point de vue utilisateur. Ainsi il permet de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Il s'agit donc de la première étape UML d'analyse d'un système.

Un acteur est un utilisateur qui communique et interagit avec les cas d'utilisation du système. C'est une entité ayant un comportement comme une personne, système ou une entreprise.

Le système fixe les limites en relation avec les acteurs qui l'utilisent (en dehors du système) et les fonctions qu'il doit fournir (à l'intérieur du système).

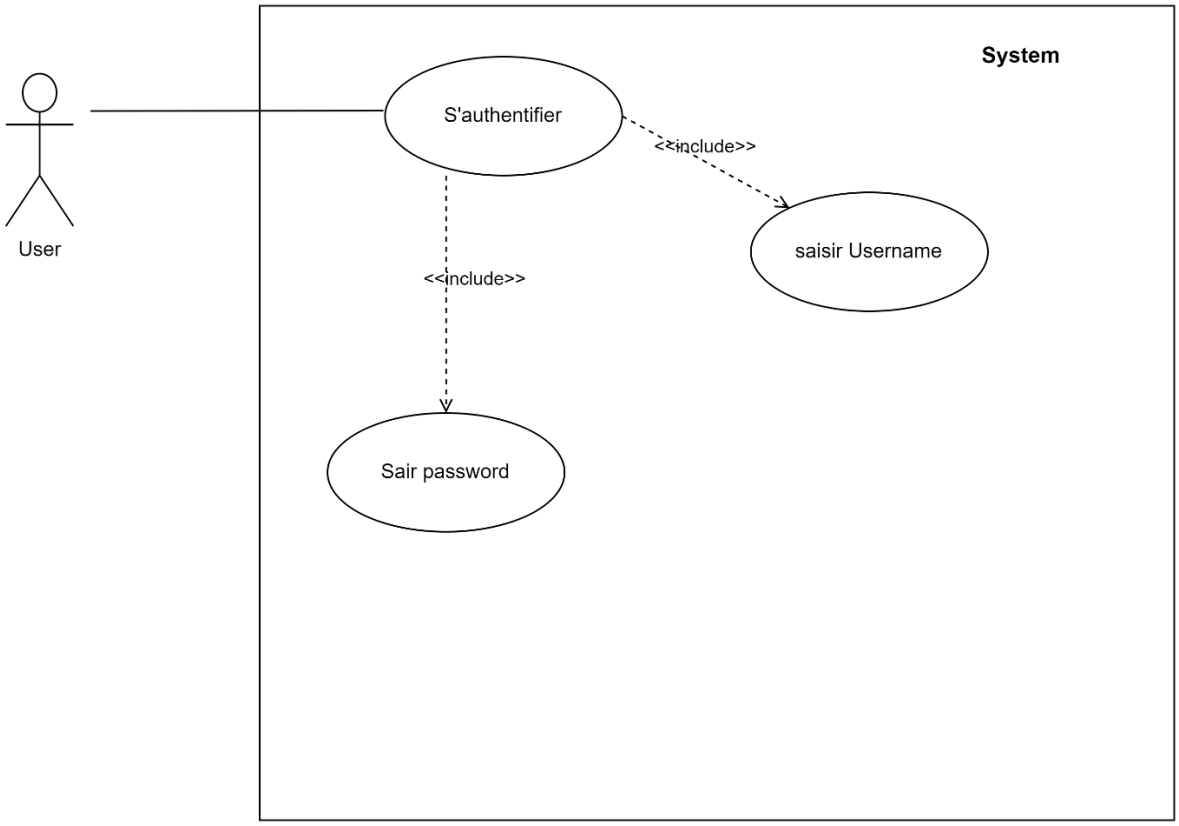
La figure sise ci-dessous présente le digramme global des cas d'utilisation de notre application**.**

4.2. Diagramme de cas d'utilisation générale



4.3. Diagramme de cas d'utilisation particulier

4.3.1. S'authentifier



Objectif : Permet d'identifier l'utilisateur pour s'assurer qu'il s'agit bien de celui qu'il prêtant être afin de lui donner accès aux fonctionnalités propices.

Acteurs principaux : Le directeur, le surveillant, le caissier, les professeurs et les élèves.

Précondition : L'utilisateur dispose d'un compte d'accès au système.

Scenario nominal :

- L'utilisateur saisit son nom d'utilisateur et son mot de passe.
- Le système vérifie le nom d'utilisateur et le mot de passe.
- Le système affiche l'´espace approprié pour chaque utilisateur.

Scenario alternatif :

- Login et /ou mot de passe sont incorrects, un retour vers la page d'authentification sera effectué avec un message d'erreur.

Description

- Description : L'utilisateur saisi son identifiant et son mot de passe. Il est reconnu par le système. Il peut accéder en fonction de son profil utilisateur

4.3.2. Gérer les utilisateurs

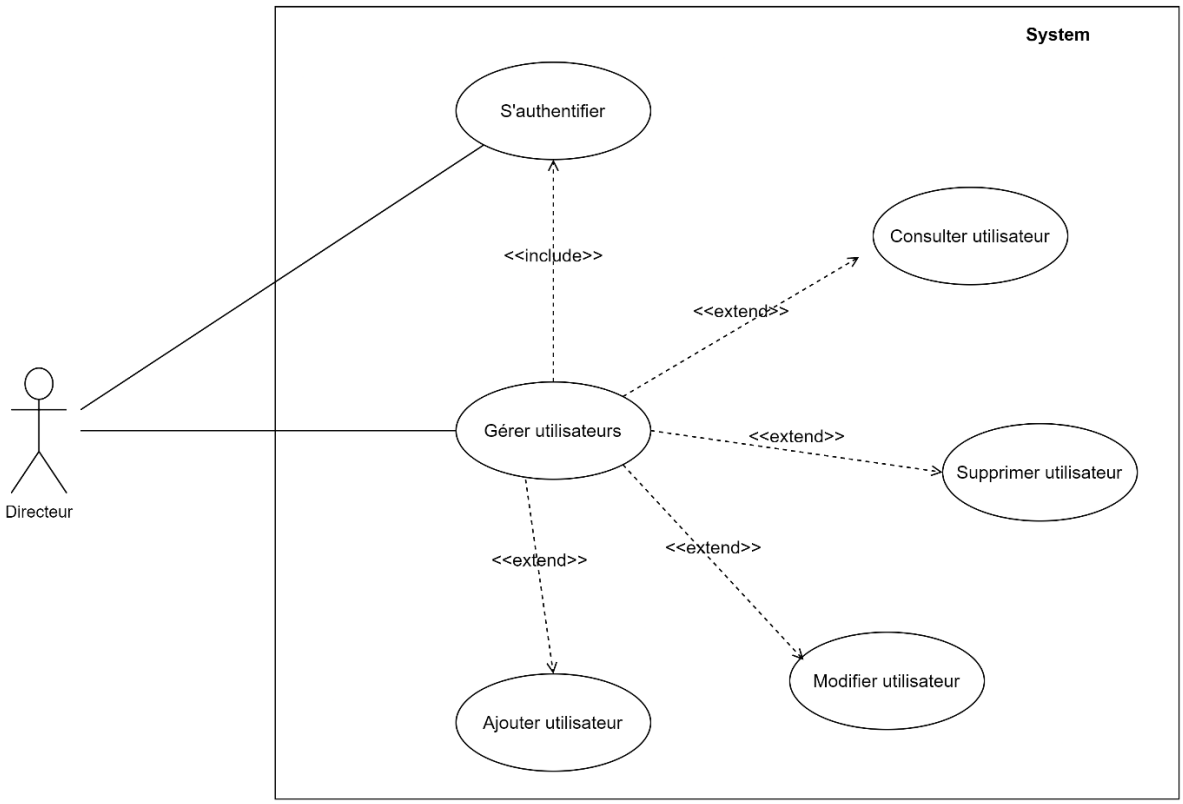


Figure 3.3 Diagramme de cas d'utilisation de : Gérer utilisateur.

Objectif : Permet à l'administrateur principal de vérifier et de faire la mise à jour des utilisateurs de l'application.

Acteurs principaux : Le directeur (administrateur principal)

Précondition : L'administrateur doit préalablement disposer d'un compte d'accès être connecté.

Scenario nominal :

Cas 1 : Créer un utilisateur

- L'administrateur choisit d'ajouter un dossier utilisateur.
- Le système affiche le formulaire à remplir.
- L'administrateur remplit et valide le formulaire.
- Le système ajoute les informations dans la base.
- Le système actualise la liste des utilisateurs et l'affiche.

Cas 2 : Modifier un dossier utilisateur

- L'administrateur choisit l'utilisateur à modifier.
- Le système affiche le formulaire de modification.
- Il modifie les champs voulus.
- Le système met à jour les informations dans la base.
- Le système actualise la liste des utilisateurs et l'affiche.

Cas 3 : Supprimer un dossier utilisateur

- L'administrateur choisit l'utilisateur à supprimer.
- Le système demande une confirmation.
- L'administrateur confirme ou annule la suppression.
- Le système supprime l'utilisateur de la base.
- Le système actualise la liste des utilisateurs et l'affiche.

Scenario alternatif :

Cas 1 :

Utilisateur existe déjà ou champs non conformes aux types, formulaire vide : un message d'erreur sera affiché.

Cas 2 :

Modification avec des champs vides, champs non conformes aux types : un message d'erreur sera affiché.

Cas 3 :

Utilisateur inexistant : un message d'erreur sera affiché

4.3.3. Gérer les élèves

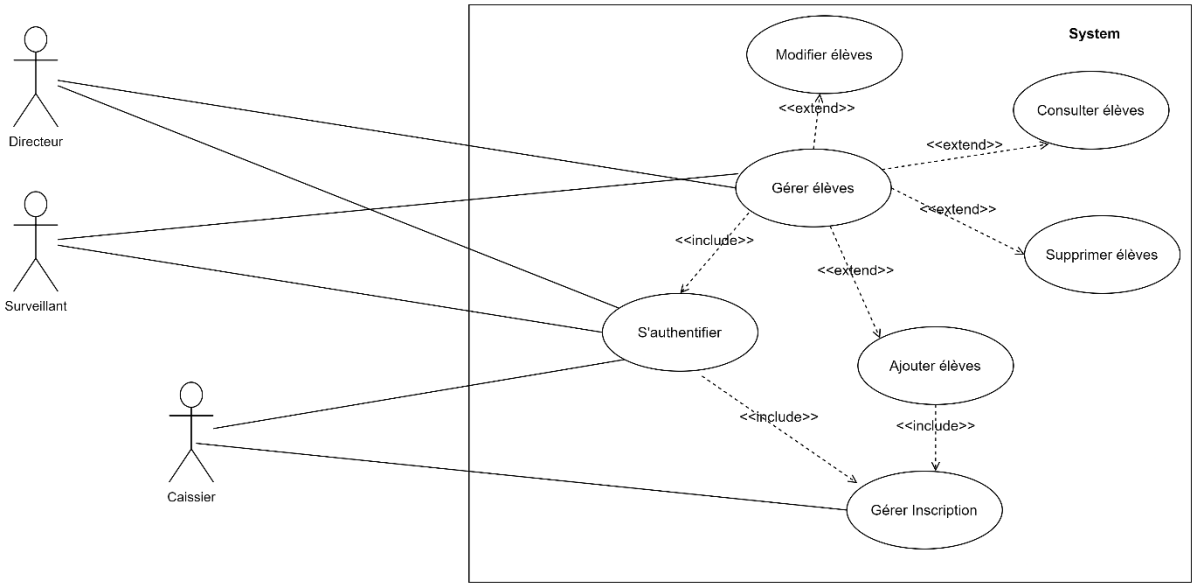


Figure 3.3 Diagramme de cas d'utilisation de : Gérer élèves.

Objectif : Permet l'ajout, la modification et la suppression d'un élève

Acteurs principaux : Le directeur

Précondition : L'administrateur doit préalablement disposer d'un compte d'accès être connecté.

Scenario nominal :

Cas 1 : Créer un dossier élève

- L'utilisateur choisit d'ajouter un dossier élève.
- Le système affiche le formulaire à remplir.
- L'utilisateur remplit et valide le formulaire.
- Le système ajoute les informations dans la base.
- Le système actualise la liste des élève et l'affiche

Cas 2 : Modifier un dossier élève

- Modifier un dossier ´étudiant
- L'utilisateur choisit l'élève à modifier.
- Le système affiche le formulaire de modification.
- Il modifie les champs voulus. 4
- Le système met à jour les informations dans la base.
- Le système actualise la liste des élève et l'affiche.

Cas 3 : Supprimer un dossier élève

- L'utilisateur choisit l'élève à supprimer.
- Le système demande une confirmation.
- L'utilisateur confirme ou annule la suppression.
- Le système supprime l'élève de la base.
- Le système actualise la liste des élèves et l'affiche.

Scenario alternatif :

Cas 1 :

- Elève existe déjà ou champs non conformes aux types, formulaire vide : un message d'erreur sera affiche.

Cas 2 :

- Modification avec des champs vides, champs non conformes aux types : un message d'erreur sera affiché.

Cas 3 :

- Elève inexistant : un message d'erreur sera affiché.

4.3.4. Gérer les professeurs

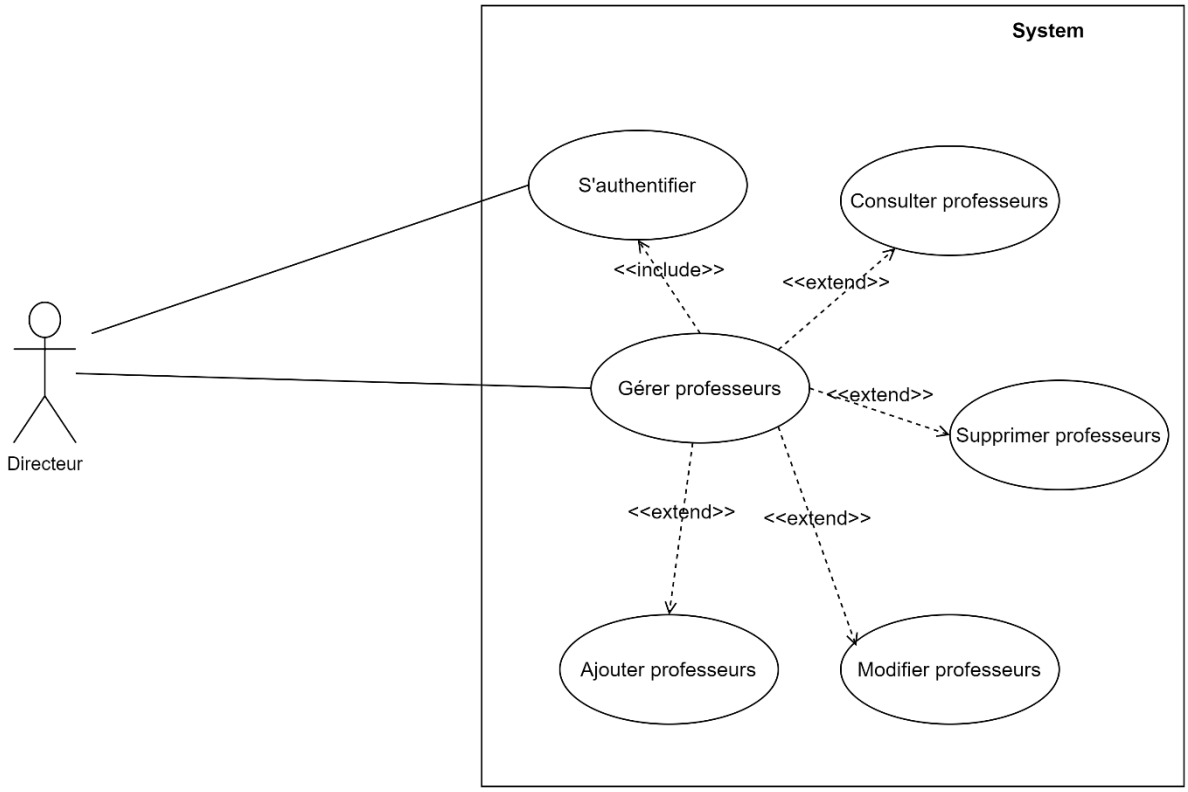


Figure 3.3 Diagramme de cas d'utilisation de : Gérer professeurs

Objectif : Permet l'ajout, la modification et la suppression d'un professeur

Acteurs principaux : Le directeur (administrateur)

Précondition : L'utilisateur doit préalablement être connecté.

Scenario nominal :

Cas 1 : Créer un dossier professeur

- L'utilisateur choisit d'ajouter un dossier de professeur.
- Le système affiche le formulaire à remplir.
- L'utilisateur remplit et valide le formulaire.

- Le système ajoute les informations dans la base.
- Le système actualise la liste des professeurs et l'affiche.

Cas 2 : Modifier un dossier de professeur

- L'utilisateur choisit le professeur à modifier.
- Le système affiche le formulaire de modification du professeur choisis.
- Il modifie les champs voulus.
- Le système met à jour les informations dans la base.
- Le système actualise la liste des professeurs et l'affiche.

Cas 3 : Supprimer un dossier professeur

- L'utilisateur choisit le professeur à supprimer.
- Le système demande une confirmation.
- L'utilisateur confirme ou annule la suppression.
- Le système supprime le professeur de la base.
- Le système actualise la liste des professeur et l'affiche.

Scenario alternatif :

Cas 1 :

- Professeur existe déjà ou champs non conformes aux types, formulaire vide : un message d'erreur sera affiché

Cas 2 :

- Modification avec des champs vides, champs non conformes aux types : un message d'erreur sera affiché

Cas 3 :

- Professeur inexistant : un message d'erreur sera affiché

Description :

L'utilisateur dispose d'un formulaire pour créer de nouveaux professeurs. Sur ce formulaire, il devra remplir les informations nécessaires des professeurs. L'utilisateur pourra modifier et supprimer ces informations. En plus d'une liste de tous les professeurs autorisés pendant une année académique, il a également la possibilité de modifier cette liste.

4.3.5. Gérer les notes

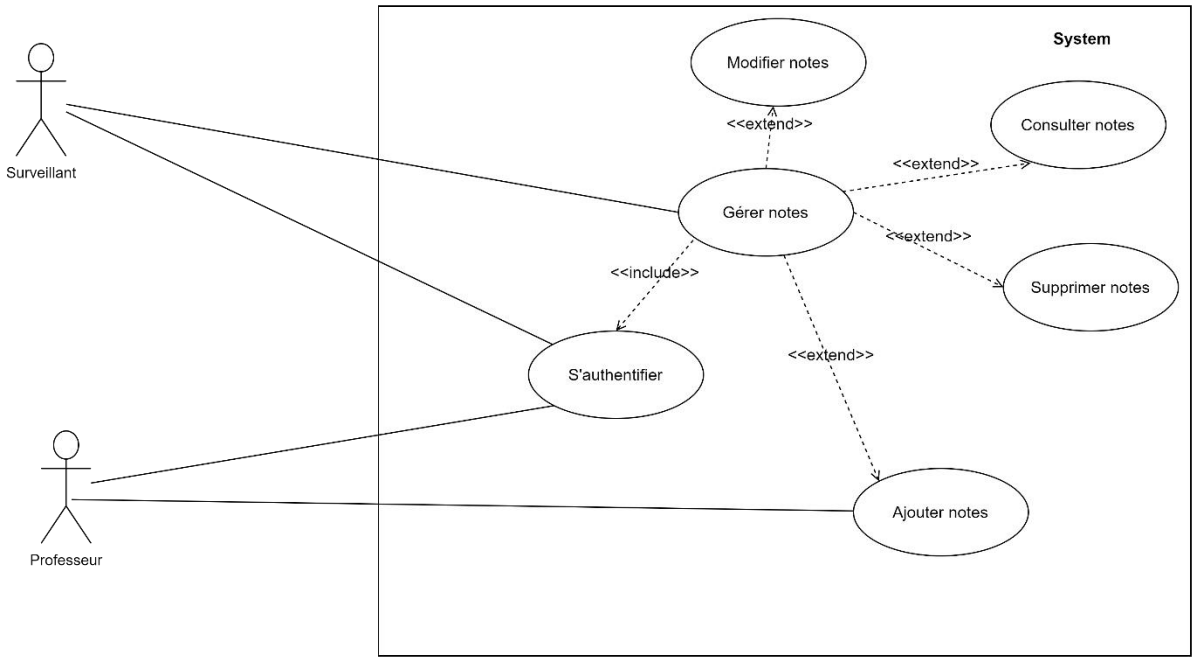


Figure 3.3 Diagramme de cas d'utilisation de : Gérer notes.

Objectif : Attribuer des notes aux élèves, calculer leurs moyennes et générer leur relevé de note. Avoir la possibilité de modifier les notes après réclamation, et ceux avant l'établissement du bulletin définitif. L'utilisateur peut consulter les notes également

Acteurs principaux : Le professeur

Précondition : le professeur doit préalablement être connecté.

Description

L'utilisateur clique sur gestion des notes, il dispose de la liste des inscrits pour une année académique rangée par classe et il renseigne les notes respectives de chacun. La moyenne de chaque matière est calculée de façon automatique par le système. Après le calcul des moyennes, ils peuvent être consulter par matière ou par classe.

Toutefois si le professeur détient un fichier numérique contant les notes de ses élèves (par exemple un fichier Excel), il lui est possible d'importer ces donnes directement, ce qui lui allège les taches.

5. Chapitre 5 : Point de vue statique

5.1. Notion de diagramme de classe.

Le diagramme de classes est considéré comme le diagramme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation. Il est important de noter qu'un même objet peut très bien intervenir dans la réalisation de plusieurs cas d'utilisation. Les cas d'utilisation ne réalisent donc pas une partition des classes du diagramme de classes.

Il s'agit d'une vue statique, car il fait abstraction des aspects temporels et dynamiques dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Chaque langage de Programmation orienté objet donne un moyen spécifique d'implémenter le paradigme objet (pointeurs ou pas, héritage multiple ou pas, etc.), mais le diagramme de classes permet de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

Le diagramme de classe est une description du système focalisé sur le concept de classe et d'association.

Elle représente un ensemble d'objets qui possèdent des propriétés similaires et des comportements communs décrivant en terme d'attributs et d'opérations. Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation

Une association consiste à présenter les liens entre les instances de classe.

5.1.1. Qu'est-ce qu'une classe ?

Une instance est une concrétisation d'un concept abstrait. Par exemple :

- La Ferrari Enzo qui se trouve dans votre garage est une instance du concept abstrait Automobile ;
- L'amitié qui lie Jean et Marie est une instance du concept abstrait Amitié ;

Une classe est un concept abstrait représentant des éléments variés comme :

- Des éléments concrets (ex. : des avions),
- Des éléments abstraits (ex. : des commandes de marchandises ou services),
- Des composants d'une application (ex. : les boutons des boîtes de dialogue),
- Des structures informatiques (ex. : des tables de hachage),
- Des éléments comportementaux (ex. : des tâches), etc.

Tout système orienté objet est organisé autour des classes.

Elle est la description formelle d'un ensemble d'objets ayant une sémantique et des caractéristiques communes et décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

C'est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique et qui sont utilisées dans la programmation orientée objet permettant de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Un objet est une instance d'une classe. C'est une entité discrète dotée d'une identité, d'un état et d'un comportement que l'on peut invoquer. Les objets sont des éléments individuels d'un système en cours d'exécution.

Par exemple, si l'on considère qu'Homme (au sens être humain) est un concept abstrait, on peut dire que la personne Marie-Cécile est une instance d'Homme. Si **Homme** était une classe, Marie-Cécile en serait une instance : un objet.

5.1.2. Caractéristique d'une classe

Une classe définit un jeu d'objets dotés de caractéristiques communes. Les caractéristiques d'un objet permettent de spécifier son état et son comportement. Les caractéristiques d'un objet étaient considérées comme soit des attributs, soit des opérations. Ce n'est pas exact dans un diagramme de classe, car les terminaisons d'associations sont des propriétés qui peuvent faire partie des caractéristiques d'un objet au même titre que les attributs et les opérations.

5.1.2.1. État d'un objet :

Ce sont les attributs et généralement les terminaisons d'associations, tous deux réunis sous le terme de propriétés structurelles, ou tout simplement propriétés, qui décrivent l'état d'un objet. Les attributs sont utilisés pour des valeurs de données pures, dépourvues d'identité, telles que les nombres et les chaînes de caractères. Les

associations sont utilisées pour connecter les classes du diagramme de classe ; dans ce cas, la terminaison de l'association (du côté de la classe cible) est généralement une propriété de la classe de base.
Les propriétés décrites par les attributs prennent des valeurs lorsque la classe est instanciée. L'instance d'une association est appelée un lien.

5.1.2.2. Comportement d'un objet :

Les opérations décrivent les éléments individuels d'un comportement que l'on peut invoquer. Ce sont des fonctions qui peuvent prendre des valeurs en entrée et modifier les attributs ou produire des résultats.

Une opération est la spécification d'une méthode. L'implémentation d'une méthode est également appelée méthode. Il y a donc une ambiguïté sur le terme méthode.

Les attributs, les terminaisons d'association et les méthodes constituent donc les caractéristiques d'une classe (et de ses instances).

5.2. Diagramme de classe global (conception)

Avant de vous présenter les diagrammes de classes de notre application, nous allons vous faire part, sous forme de tableau, des différentes classes que nous avons eu à ressortir lors de notre analyse puis nous essayerons de vous faire une petite description de quelques unes de nos classes métiers de l'application.

5.2.1. Tableaux de classe

Nom classe (tables)	Attributs
Absences	id, #personne_id, #matiere_id, nombreHeure, dateAbsence
AnneeAcademique	Id, dateDebutAnneeAcademique, dateFinAnneeAcademique
Classes	id, libelle Classe
Classe_Evaluation	id, #classe_id, #evaluation_id
Classe_Matiere	id, #classe_id, #matiere_id, coefficient
Classe_Profeseeur	id, #classe id, #professeur id
Elève	id, #user_id, #classe_id, nomTuteurLegal, numeroTuteurLegal
Evaluations	id, #matiere_id, libelleEvaluation, typeEvaluation, dateEvaluation
DetailsNote	id, #matiere_id, #classe_id, #evaluation_id
Inscription	id, #classe_id, #eleve_id, montantImscription, dateInscription
Matière	id, libelleMatiere
Matiere_Professeur	id, #matiere_id, #professeur id
Notes	id, #eleve_id, #detailsNotes_id, noteEvaluation
Paiements	id, #eleve_id, #classe_id, montantPaye, moisPaye, dateDePaiement
Personnes	Id, nom, prenom, dateDeNaissance, lieuDeNaissance, genre, cni, numeroTelephone, adresse
Professeurs	id, #user_id, grade, diplomeAcademique, diplomePedagogique, nombreAnneeExperience,
Rôle	id, name
Role_Users	id, #user_id, #role_id
User	id, #personne_id, email, password, email_verified_at, dateCreationCompte

5.2.2. Description de quelques classes métiers

AnneeAcademique : Nous pouvons observer la relation de cette entité avec presque tous les autres entités du diagramme de classe. Cela se justifie par le fait que le fonctionnement de l'application est fondamentalement lié à une année académique.

Inscription : L'inscription des étudiants dans les classes est réalisée par cette classe. Elle est fondamentale pour le système car elle permet de déterminer les élèves appartenant à une classe.

Elève : Cette entité concerne les élèves inscrits dans une année académique, elle représente l'effectif que l'application va gérer. Elle récolte tous les attributs (informations personnelles) concernant l'élève.

Matière : Cette classe réalise la gestion des matières de l'établissement. Elle insère, modifie et supprime des matières.

Classe : Cette entité permet de gérer les différentes classes (ou niveaux). C'est à travers elle qu'on arrive à insérer, et modifier et ou supprimer les classes en fonction de l'établissement.

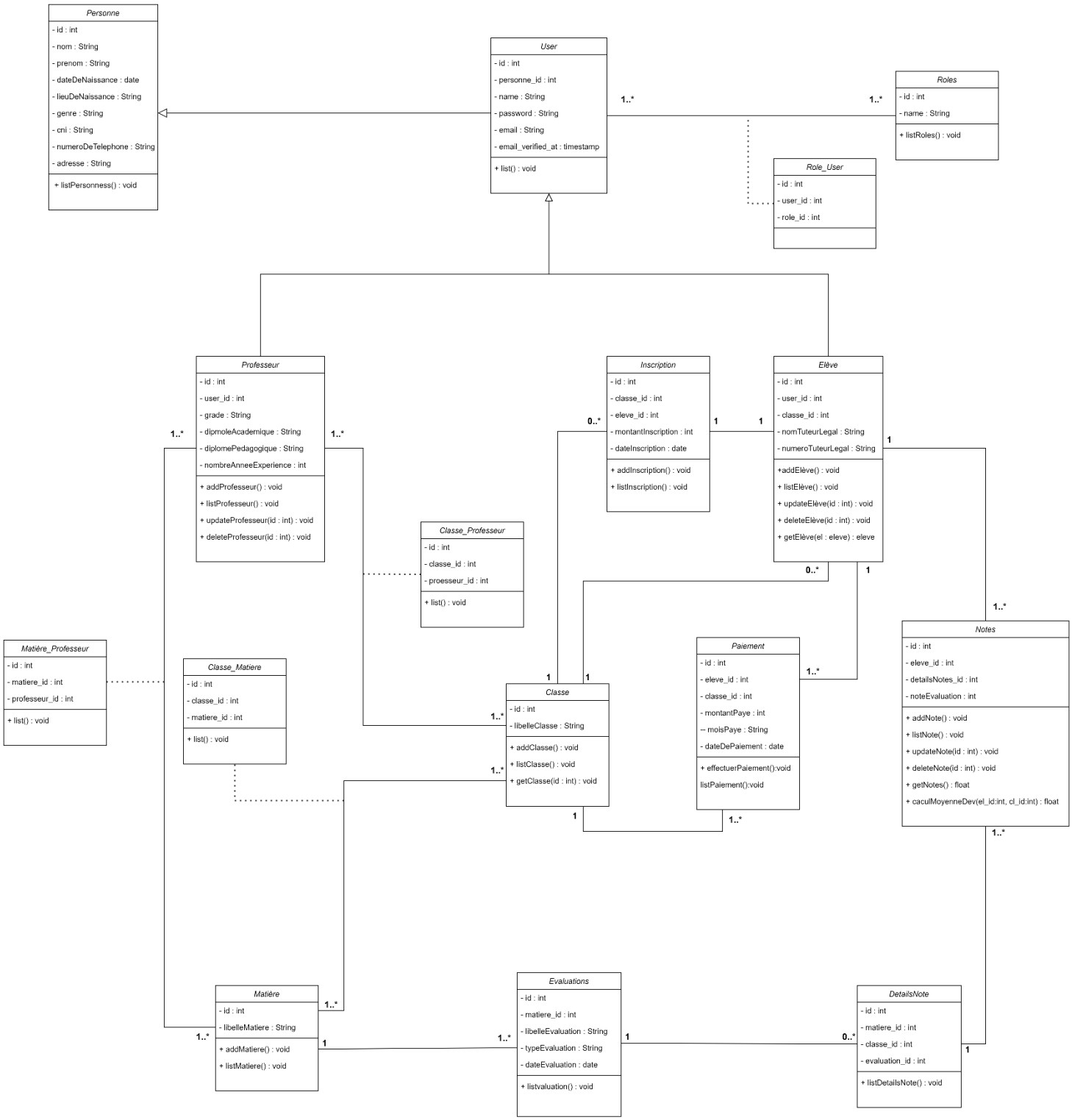
User : Cette entité est chargée de la gestion des utilisateurs du système. Elle est chargée de la création d'un utilisateur et de l'attribution des droits à celui-ci.

Professeur : cette classe recense tous les enseignants de l'établissement, elle insère, modifie et supprime les enseignants.

Evaluation : Cette entité va permettre d'établir le calendrier des évaluations de l'école durant une année académique. Elle sert tout aussi recense les différentes évaluations au cours d'une année académique

Note : Une classe pour prendre en compte les notes des élèves sur les différentes évaluations ainsi que la gestion de celles-ci. Les moyennes des différentes matières et l'établissement des bulletins est réalisé grâce aux information recueillis à partir de cette classe.

5.2.3. Diagrammes



4ème partie : Réalisation

6. Chapitre 6 : Plateforme de développement

La réalisation du projet consiste dans un premier temps à traduire en langage informatique les concepts qui ont été élaborés pendant la phase de conception. Les langages informatiques « PHP », « Javascript », « HTML » et SQL ont été utilisés pour le développement du projet.
Les modules développés et testés seront livrés aux utilisateurs afin que ces derniers puissent les valider.

6.1. Environnement

6.1.1.MVC

L'architecture Modèle/Vue/Contrôleur (MVC) est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface. L'organisation globale d'une interface graphique est souvent délicate. Bien que la façon MVC d'organiser une interface ne soit pas la solution miracle, elle fournit souvent une première approche qui peut ensuite être adaptée. Elle offre aussi un cadre pour structurer une application.
Dans l'architecture MVC, les rôles des trois entités sont les suivants.

- Modèle : données (accès et mise à jour)
- Vue : interface utilisateur (entrées et sorties)
- Contrôleur : gestion des événements et synchronisation

6.1.1.1. Rôle du modèle

Le modèle contient les données manipulées par le programme. Il assure la gestion de ces données et garantit leur intégrité. Dans le cas typique d'une base de données, c'est le modèle qui la contient.

Le modèle offre des méthodes pour mettre à jour ces données (insertion suppression, changement de valeur). Il offre aussi des méthodes pour récupérer ses données. Dans le cas de données importantes, le modèle peut autoriser plusieurs vues partielles des données. Si par exemple le programme manipule une base de données pour la gestion

des effectifs, le modèle peut avoir des méthodes pour avoir l'effectif total, toutes les notes des élèves ainsi que leur différent bulletin.

6.1.1.2. Rôle de la vue

La vue fait l'interface avec l'utilisateur. Sa première tâche est d'afficher les données qu'elle a récupérées auprès du modèle. Sa seconde tâche est de recevoir toutes les actions de l'utilisateur (clic de souris, sélection d'une entrée, boutons, ...). Ses différents événements sont envoyés au contrôleur.
La vue peut aussi donner plusieurs vues, partielles ou non, des mêmes données. Par exemple, l'application de conversion de bases à un entier comme unique donnée. Ce même entier est affiché de multiples façons (en texte dans différentes bases, bit par bit avec des boutons à cocher, avec des curseurs). La vue peut aussi offrir la possibilité à l'utilisateur de changer de vue.

6.1.1.3. Rôle du Controller

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.

Dans le cas d'une base de données des emplois du temps. Une action de l'utilisateur peut être l'entrée (saisie) d'un nouveau cours. Le contrôleur ajoute ce cours au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle personne pour visualiser tous ses cours. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des cours de cette personne.

Le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants. En effet si un même objet reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement. Ce tri des événements peut s'avérer fastidieux et peut conduire à

un code pas très élégant (un énorme switch). C'est pour éviter ce problème que le contrôleur est réparti en plusieurs objets.

6.1.1.4. Les interactions

Les différentes interactions entre le modèle, la vue et le contrôleur sont résumées par le schéma de la figure suivante.

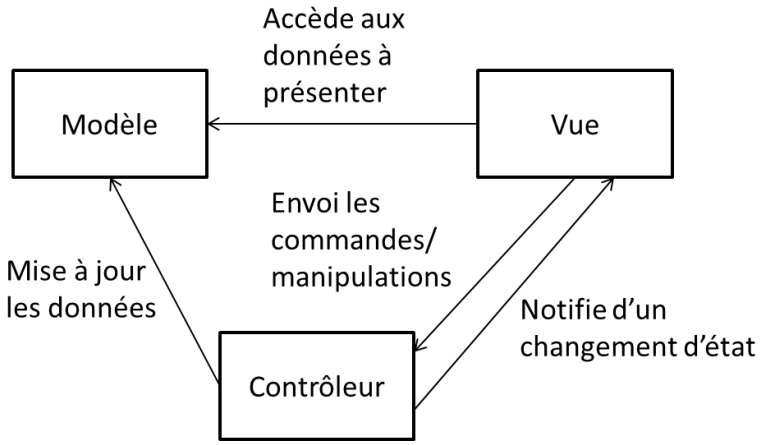


Figure 2.3 : Interactions entre le modèle, la vue et le contrôleur.

6.1.2. La technologie orientée objet

La programmation orientée objet (POO), ou programmation par objet, est un paradigme de programmation informatique, il consiste en la définition et l'interaction de briques logicielles appelées objets. Un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre. Il possède une structure interne et un comportement, et il sait interagir avec ses pairs. Il s'agit donc de représenter ces objets et leurs relations.

L'interaction entre les objets via leurs relations permet de concevoir et réaliser les fonctionnalités attendues, de mieux résoudre le ou les problèmes. Dès lors, l'étape de modélisation revêt une importance majeure et nécessaire pour la POO. C'est elle qui permet de transcrire les éléments du réel sous forme virtuelle.

Orthogonalement à la programmation par objet, afin de faciliter le processus d'élaboration d'un programme, existent des méthodologies de développement logiciel objet dont la plus connue est le Unified Software Development Process (UP) qui utilisent des langages de modélisation tels que le Unified Modeling Language (UML).

Même s'il est possible de concevoir par objets une application informatique sans utiliser des outils logiciels dédiés, ces derniers facilitent la conception, la maintenance et la productivité.

Parmi eux, on peut citer :

- Les langages de programmation (Java, YB.NET, Vala, Objective C, Eiffel, Python, Ruby, Ada, PHP, Smalltalk, LOGO, AS3, Haxe ...)

- Les outils de modélisation qui permettent de concevoir sous forme de schémas semi-formels la structure d'un programme (Objecteering, UMLDraw, Rhapsody, DBDesigner. ...)
- Les bus distribués (DCOM, CORBA, RMI, Pyro ...)
- Les ateliers de génie logiciel ou AGL (Visual Studio pour des Dotnet, NetBeans ou Eclipse pour le langage Java)
- La technologie objet est la voie la plus prometteuse pour déployer des systèmes basés sur des composants qui peuvent être adaptés et changés sans avoir à examiner la totalité du code.

6.1.3. Framework Laravel

6.1.3.1. Définition de Laravel

En programmation informatique, un Framework (appelé aussi cadre applicatif, cadre d'applications ou encore infrastructure de développement) désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture).
Les Framework sont 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.

6.1.3.2. Constitution de Laravel

Laravel a été créé par Taylor Otwell en juin 2011. Le référentiel Laravel/laravel présent sur le site GitHub contient le code source des premières

versions de Laravel. A partir de la cinquième version, le Framework est développé au sein du référentiel Laravel/Framework.

En peu de temps, une communauté d'utilisateurs du Framework s'est constituée. Laravel, initie une nouvelle façon de concevoir un Framework en utilisant ce qui existe de mieux pour chaque fonctionnalité. Par exemple toute application web a besoin d'un système qui gère les requêtes HTTP. Plutôt que de réinventer quelque chose, le concepteur de Laravel a tout simplement utilisé celui de Symfony en l'étendant pour créer un système de routage efficace.

En quelque sorte Otwel a fait son marché parmi toutes les bibliothèques disponibles. Laravel ce n'est pas seulement le regroupement de bibliothèques existantes, c'est aussi de nombreux composants originaux et surtout une orchestration de tout ça. Laravel est constitué de :

- Un système de routage perfectionné (RESTFul et ressources)
- Un créateur de requêtes SQL (langage de requête structurée) et ORM (object-relational mapping) performants
- Un moteur de Template efficace
- Un système d'authentification pour les connexions
- Un système de validation, de pagination, de migration pour bases de données, d'envoi d'emails
- Un système d'autorisations
- Une gestion des sessions ...

Durant cette section, nous présentons un scénario d'utilisation de l'application par quelques interfaces

6.1.4.XAMPP

XAMPP est un logiciel libre open source développée par des amis Apache. Le progiciel XAMPP contient des distributions Apache pour le serveur Apache, MariaDB, PHP et Perl. Et c'est essentiellement un hôte local ou un serveur local. Ce serveur local fonctionne sur votre propre ordinateur de bureau ou portable. Vous pouvez simplement installer ce logiciel sur votre ordinateur portable ou de bureau et tester les clients ou votre site Web avant de le télécharger sur le serveur Web ou l'ordinateur distant. Ce logiciel serveur XAMPP vous offre un environnement approprié pour tester les projets MYSQL, PHP, Apache et Perl sur l'ordinateur local.

La forme complète de XAMPP est X signifie multiplateforme, (A) serveur Apache, (M) MariaDB, (P) PHP et (P) Perl. Le multi-plateforme signifie généralement qu'il peut fonctionner sur n'importe quel ordinateur avec n'importe quel système d'exploitation. MariaDB est le serveur de base de données le plus connu et développé par l'équipe MYSQL. PHP fournit généralement un espace pour le développement web. PHP est un langage de script côté serveur. Et le dernier Perl est un langage de programmation et est utilisé pour développer une application web

6.1.4.1. Outils principaux de XAMPP

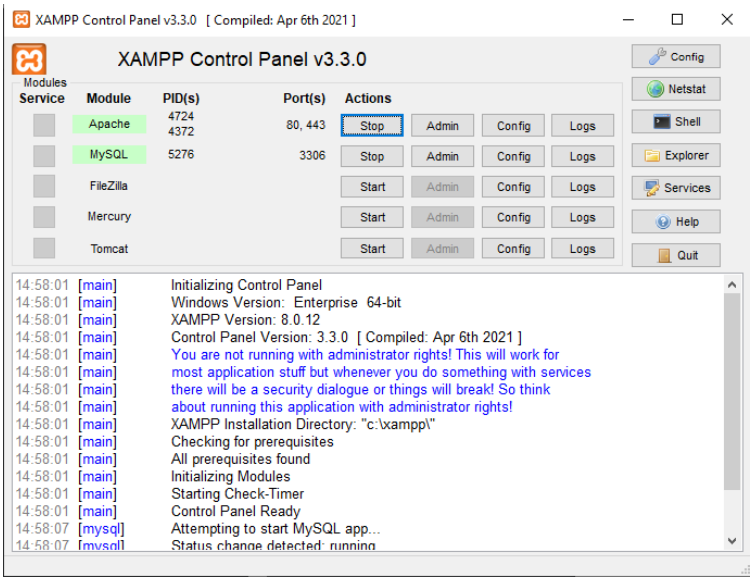
XAMPP contient des outils tels qu'Apache, MYSQL, PHP et Perl. Nous verrons à propos de ces outils. (Ganesan, 2017)

6.1.4.1.1. APACHE

Le serveur Apache est un logiciel libre open source qui est initialement développé par un groupe de développeurs de logiciels et maintenant il est maintenu par Apache Software Fondation. Apache HTTP est un serveur distant (ordinateur) si quelqu'un demande des fichiers, des images ou des documents en utilisant son navigateur, il servira ces fichiers aux clients utilisant des serveurs HTTP. Principalement les sociétés d'hébergement utilisent cette application pour créer un serveur VPS et un hébergement partagé pour leurs clients. (Ganesan, 2017)

6.1.4.1.2. MYSQL

MYSQL est un logiciel open source. C'est en fait un système de gestion de base de données relationnelle (SGBDR). Ce SQL est synonyme de langage de requête structuré. C'est le SGBDR le plus populaire et le mieux utilisé pour développer une variété d'applications logicielles basées sur le web. Avec l'aide de MYSQL, il est possible d'organiser les informations, gérer, récupérer et mettre à jour les données quand vous le souhaitez. (Ganesan, 2017)



6.2. Langage de programmation

6.2.1. HTML

Le HTML et sa variante plus stricte XHTML sont des langages de balisage des pages Web. Il n'y a pas si longtemps, le HTML servait à définir aussi bien la structure des pages que leur présentation visuelle. Aujourd'hui, ces deux aspects doivent être bien distincts et le X/HTML est destiné uniquement à représenter la structure d'une page : titres, sous-titres, paragraphes, images, formulaires de saisie, liens hypertextes, etc. C'est la base d'une page Web, parfois la seule considérée et utilisée par le logiciel qui visite cette page, comme les moteurs de recherche ou les navigateurs textuels. On qualifie de « statiques » les pages dont le code X/HTML n'est modifié ni par JavaScript, ni par PHP avant ou après l'affichage dans le navigateur.

6.2.2.CSS

Le code CSS (Cascading Style Sheets, ou feuilles de styles en cascade) permet de modifier la présentation des éléments X/HTML : couleur, taille, police de caractères, mais aussi position sur la page, largeur, hauteur, empilement, bref tout ce qui touche à la mise en page d'un document X/HTML. Ainsi, un même document X/HTML pourra changer d'apparence sans changer de structure, grâce uniquement à la modification des règles CSS qui lui sont appliquées. La séparation de la structure et de la présentation facilite ainsi la construction, mais aussi la maintenance et l'évolution des pages Web.

6.2.3.Bootstrap

Bootstrap a été développé en 2011 par l'équipe du réseau social Twitter. Bootstrap est un Framework frontend (HTML5, CSS et JavaScript) spécialement conçu pour le développement d'application web "responsive", c'est-à-dire qui s'adaptent automatiquement à différents dispositifs et tailles d'écran (tablettes, smartphones, desktop…etc.). Il fournit des outils avec des styles déjà en place pour des typographies, des boutons, des interfaces de navigation et bien d'autres encore. Il peut être utilisé pour créer par exemple, des pages de site de présentation, pour une interface graphique d'une application web ou être intégrer à un thème d'un CMS, Bootstrap est de plus en plus utilisé, il est devenu « le plus populaire des Framework FrontOffice pour développer des projets responsive et mobile-first sur le web. » (montuy, 2017)

6.2.4.PHP

La forme complète de PHP est le préprocesseur hypertexte. C'est un langage de script côté serveur qui vous aide à créer des sites Web dynamiques. Ce langage est principalement utilisé pour construire des applications logicielles basées sur le Web. C'est un logiciel open source et fonctionne bien avec MYSQL. Ce qui se passe réellement, c'est que le code PHP sera exécuté sur le serveur et que le code HTML sera affiché du côté du navigateur. (.xp-internet, n.d.)

Figure 4-5 : PHP logo

6.2.5.JQUERY

JQuery est une bibliothèque JavaScript concise et rapide qui peut être utilisée pour simplifier la gestion des événements, le déplacement de documents HTML, les interactions Ajax et l'animation pour le développement rapide de sites Web. JQuery simplifie les scripts côté client du HTML, simplifiant ainsi le développement des applications Web 2.0. JQuery est une bibliothèque libre, opensource et à double licence sous licence publique générale GNU. Il est considéré comme l'une des

bibliothèques JavaScript (JS) préférées disponibles aujourd'hui. À partir de 2012, il est utilisé par plus de la moitié des meilleurs sites Web. (définition, 2017).

6.2.6.AJAX

Ajax est un acronyme pour Asynchronous JavaScript and XML (« XML et Javascript asynchrones ») et désigne une solution informatique libre pour le développement de pages dynamiques et d'applications Web.

Le principal attrait d'Ajax réside dans le rendu dynamique des pages et une interactivité qui n'est pas sans rappeler le mode d'interaction du client lourd. L'avènement des interfaces riches (Rich Internet Application) permet au Web de fournir des sites associant une interactivité plus riche, jusque-là réservée au client lourd, tout en gardant une facilité de consultation qui leur sont propres.

Pour expliquer le concept d'AJAX en quelques mots, il s'agit de permettre l'échange d'informations entre le navigateur web (sur le poste de l'utilisateur) et le serveur web (chez l'hébergeur) sans recharger l'ensemble de la page web utilisée. Ces échanges s'effectuent de façon asynchrone (le mode synchrone est aussi possible) à l'aide de fonctions développées en langage Javascript.

7. Chapitre 7 : Présentation de l'application

7.1. Interface graphique

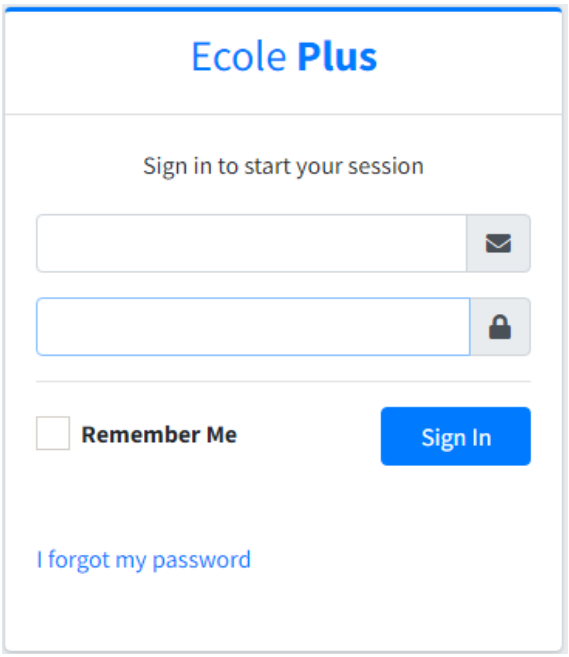
7.1.1. Page de connexion

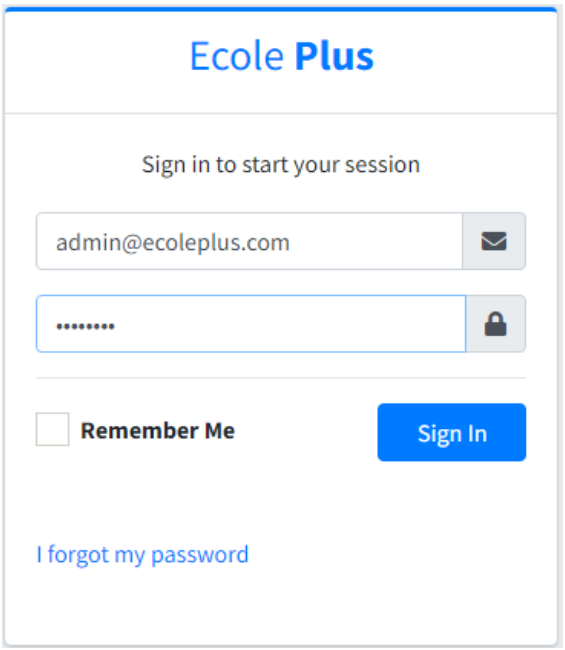
Cette page permet à l'utilisateur d'accéder à son propre compte pour profiter des services fournis par la plateforme

L'application a été conçu de sorte à ce que l'accès soit restreint, seuls les utilisateurs autorisées (utilisateurs ayant un compte) peuvent y accéder.

Pour pouvoir accéder au menu principal de l'application, il est impératif de s'authentifier, et cela en renseignant un identifiant et un mot de passe

Il n'y a pas de lien pour un page s'inscrire car cela relève de l'autorité de l'administrateur.



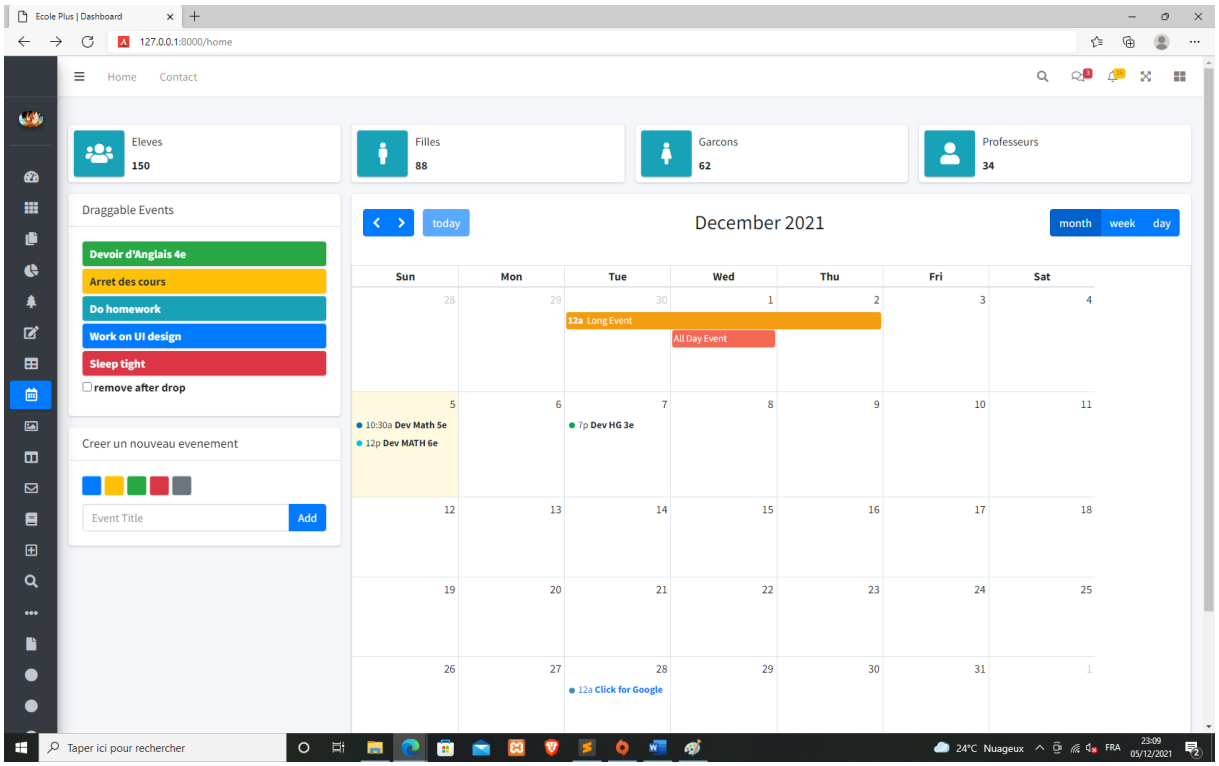


7.1.2.Page d'accueil

La page d'accueil est conçue pour avoir une vision générale des données auquel on a accès.

Lorsque l'utilisateur se connecte, celui-ci voit s'afficher un menu, selon le profil qu'il possède. L'utilisateur connecté est redirigé vers l'écran contenant le menu suivant :

On voit bien dans ce menu les différents modules de gestion que nous avons développés. L'utilisateur pour accéder à un module n'a qu'à cliquer sur l'icône correspondante.

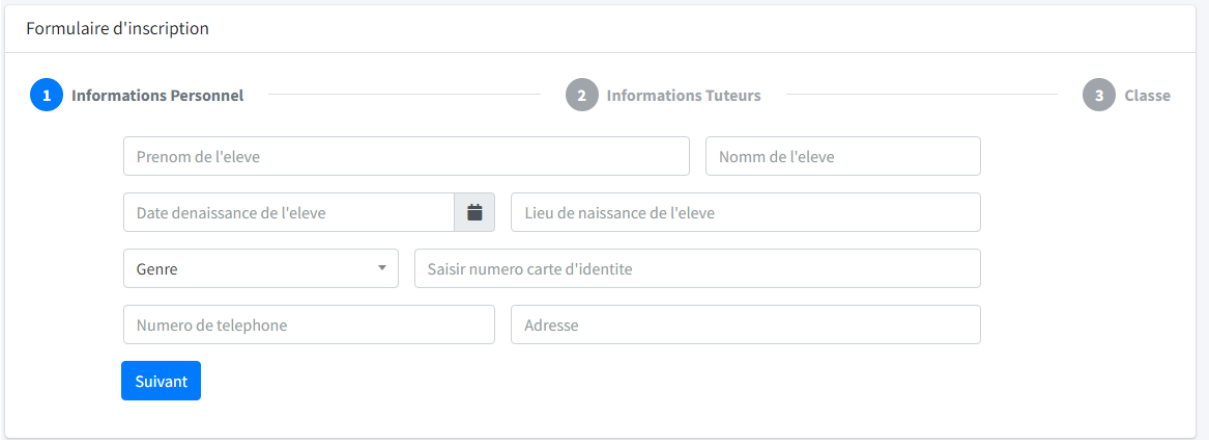


7.1.3.Page d'ajout Elève (Inscription)

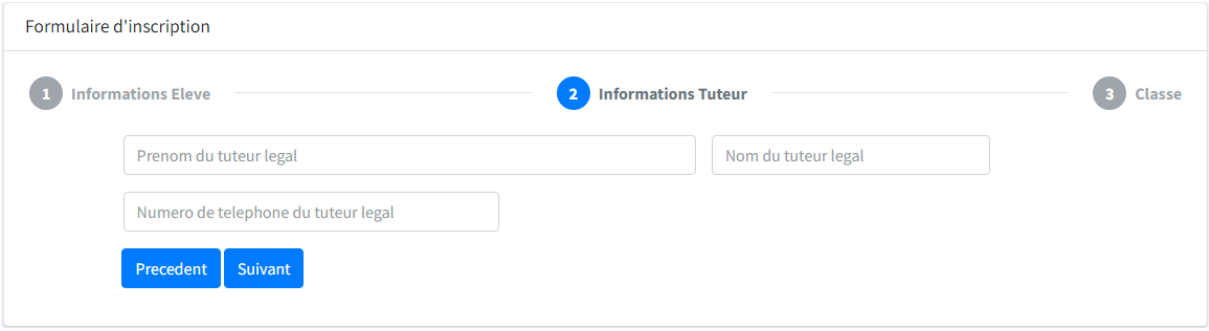
En cliquant sur le bouton « Ajouter un nouvel élève », une page contenant un formulaire constitué de trois onglets s'affiche. Chaque onglet contient un ensemble d'informations nécessaires pour enregistrer un élève. Le premier onglet « Informations personnelles » contient le nom, prénom, genre, code national élève, CIN ainsi que la date et le lieu de naissance. Le deuxième onglet « Information du tuteur » contient le nom, le prénom et le numéro de téléphone du tuteur. Le troisième onglet

quant à lui contient les informations sur la classe et un bouton pour confirmer les informations saisies dans les onglets précédents.

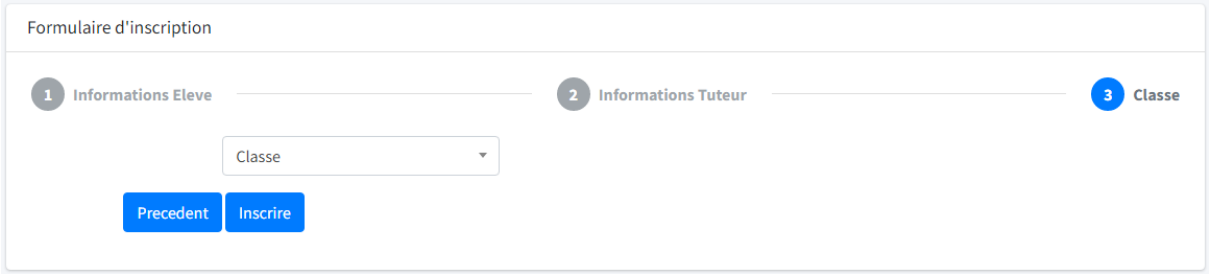
1er onglet



2ème onglet

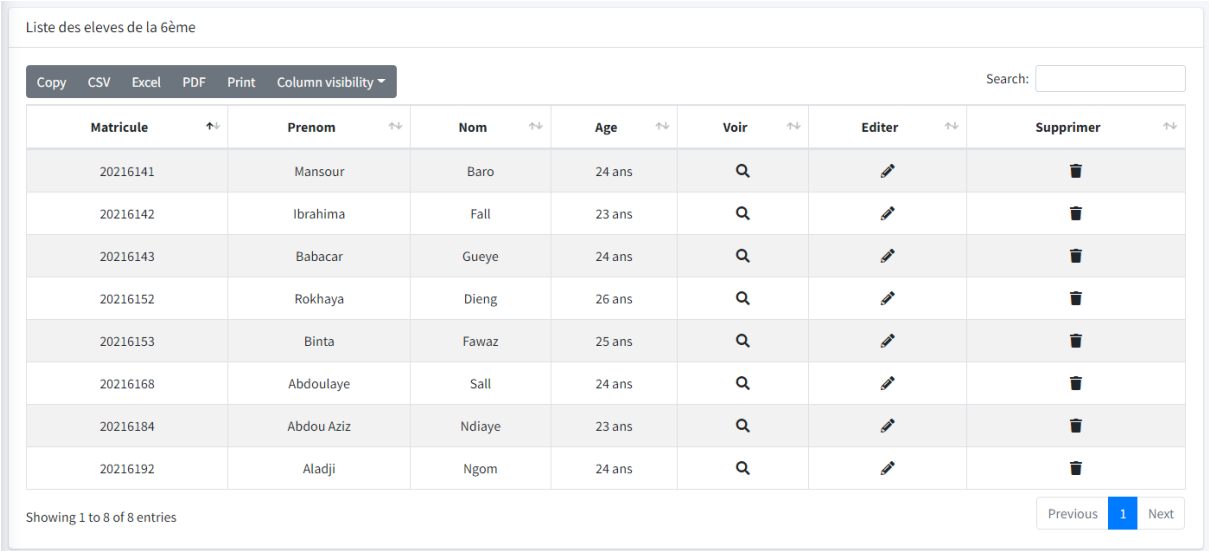


3ème onglet



7.1.4.Page de consultation des élèves

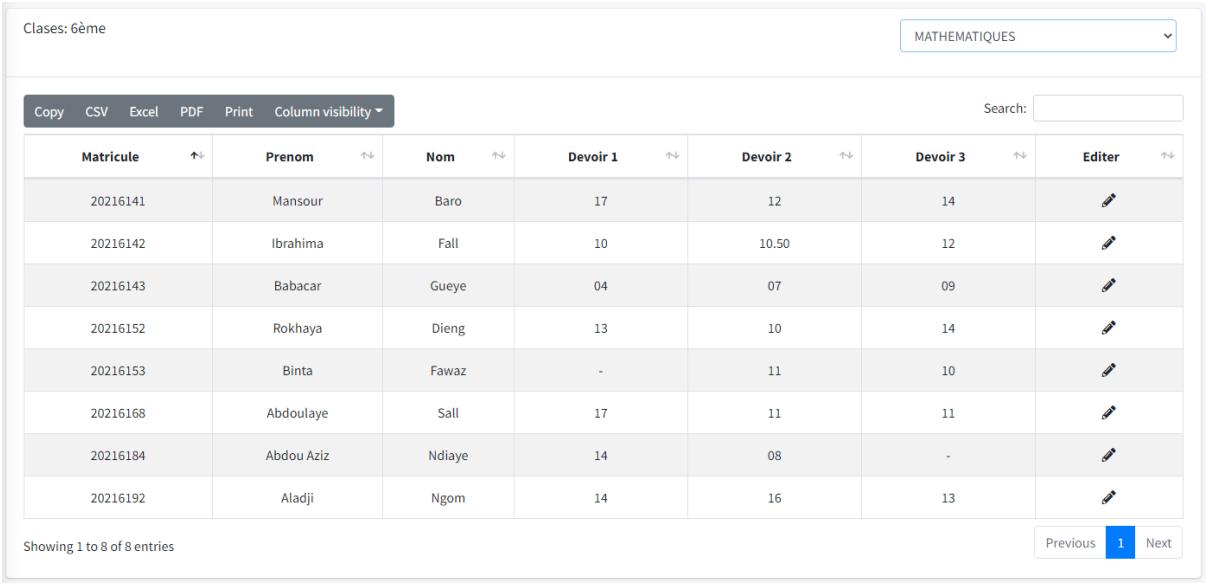
En accédant à la page Liste élèves, l'utilisateur est redirigé vers une page listant les différents élèves d'une classe donnée enregistrés sous forme d'un tableau, avec la possibilité de consulter, modifier ou supprimer les informations d'un élevé selon les droits d'accès dont dispose l'utilisateur, et cela en cliquant sur l'icône correspondante à chacune de ses actions.



7.1.5.Page de consultation des notes :

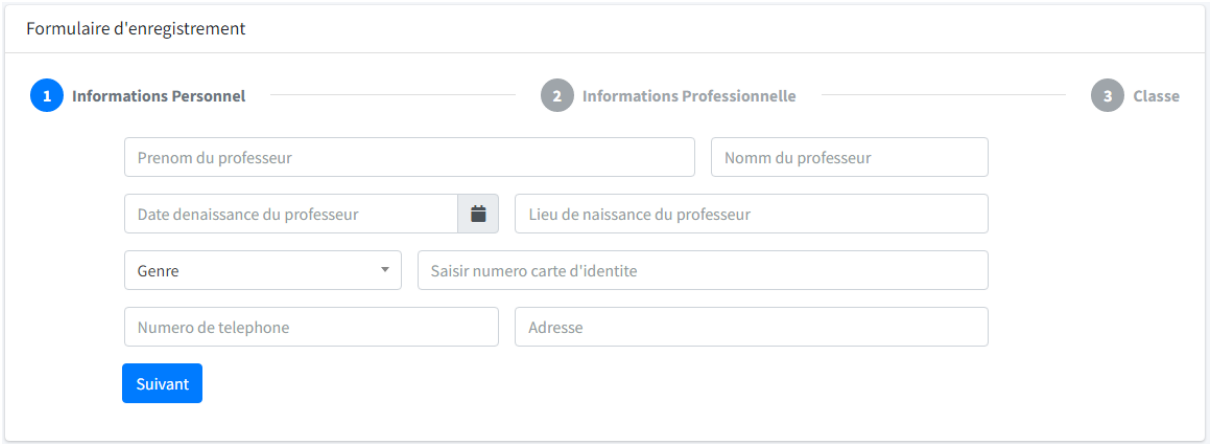
En accédant à cette page, l'utilisateur est en mesure de consulter les notes des élèves d'une classe. Il lui suffit de choisir une matière pour charger les différentes notes des élèves de la classes dans cette matière. Le chargement des notes après changement de matière se fait de façon automatique.

Selon les droits attribués à l'utilisateur, il peut consulter/modifier toutes les notes des tous les élèves ou éditer les notes d'un nombre restreint d'élèves.

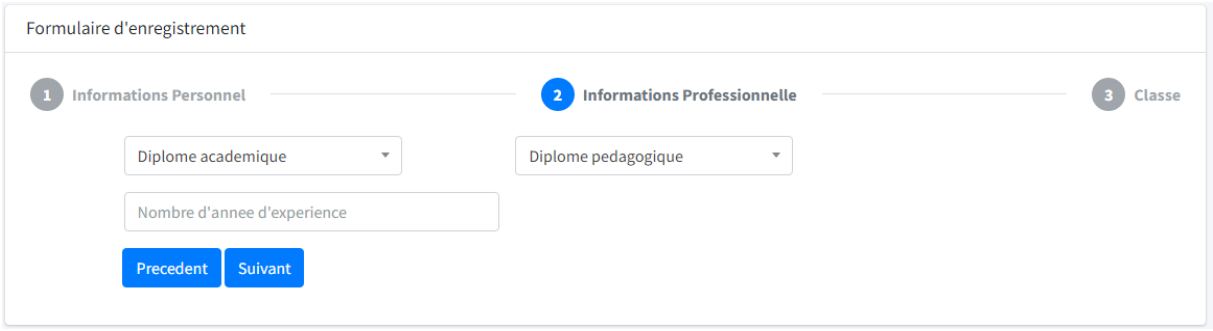


7.1.6. Page d'ajout d'un Professeur

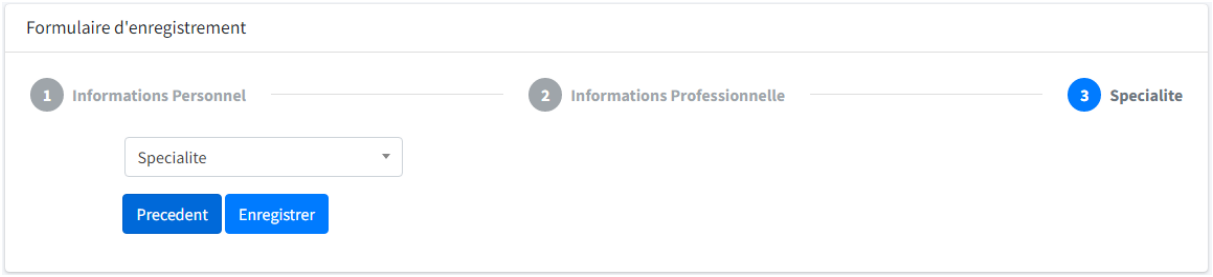
1er onglet



2ème onglet


Formulaire d'enregistrement
1 Informations Personnel
2 Informations Professionnelle
3 Classe
Diplome academique
Diplome pedagogique
Nombre d'annee d'experience
Precedent
Suivant

3ème onglet


Formulaire d'enregistrement
1 Informations Personnel
2 Informations Professionnelle
3 Specialite
Specialite
Precedent
Enregistrer

Conclusion Générale

L'objectif de notre projet de fin d'études était de concevoir et implémenter une application web de gestion d'école qui permet d'automatiser et de simplifier le processus de gestion d'une école privée en lui fournissant différents services liés au milieu scolaire. Cette phase pratique de notre formation consacrée à l'analyse, à la conception et à la programmation nous a conduits à partir des problèmes constatés, à satisfaire les besoins réels en adéquation avec les objectifs prescrits et les contraintes rencontrées.

Nous sommes arrivés à développer toutes les fonctionnalités du système visées au départ. L'intégration a été réalisée avec succès, la solution proposée à l'issue de ce travail permet d'assurer de façon efficace les objectifs visés.

La gestion des projets nécessite un fort investissement, car pour chaque projet les problématiques soulevées par les utilisateurs sont différentes. Pour répondre aux attentes de ces derniers, nous devons nous adapter du point de vue méthodologique, technique et organisationnel. Par conséquence, nous devons améliorer continuellement sur tous les aspects du processus de développement afin de maîtriser plusieurs outils tant du point de vue de la conception que du développement.

La réalisation d'un tel projet nous a permis d'approfondir nos connaissances acquises tous le long de notre formation, et de pratiquer de nouvelles technologies, elle nous a ainsi permis de maitriser le langage UML, et les outils de développement web. Bien évidemment, nous avons rencontré des difficultés pendant ce projet. Ces difficultés vont de la modélisation à la programmation. Ce travail nous a donné l'opportunité de toucher une partie de divers aspects du métier de développeur et du concepteur.

Nous n'avons pas la prétention d'avoir réalisé un travail parfait. C'est pourquoi, nous voudrions au-delà des imperfections et insuffisances que vous constaterez et rencontrerez, tenir compte de vos remarques, critiques, et suggestions afin de rendre possible la perfection de ce projet.

Webographie

- https://www.memoireonline.com/05/12/5812/m_La-realisation-dune-application-de-contrle-total-des-processus-dun-ordinateur-distant24.html
- https://www.filedoc.eu/fr/avantages-d-une-solution-de-ged
- https://gesecole.com/
- https://myscol.com/
- https://deptinfo.labri.fr/ENSEIGNEMENT/INITINFO/initinfo/supports/book/node50.html
- https://sokeo.fr/cycle-de-vie-logiciel-2/
- https://www.memoireonline.com/07/08/1287/m_mise-en-place-d-une-plate-forme-de-cartographie-dynamique5.html
- https://www.memoireonline.com/01/12/5149/m_Integration-d-un-observatoire-urbain-sur-Google-Maps19.html
- https://hal.archives-ouvertes.fr/hal-01714731/document
- https://fr.wikipedia.org/wiki/Programmation_orientée_aspect
- https://www.memoireonline.com/04/17/9870/m_Conception-et-realisation-dun-site-web-dynamique-pour-la-prise-de-rendez-vous-medicale-en-ligne7.html
- https://www.rapport-gratuit.com/conception-avec-uml/
- https://prive.iutenligne.net/wfj0UXGINyjUiM5b/informatique/langages/kettaf/UML/07encadrementdeprojet/04vues.html
- https://www.cours-gratuit.com/cours-uml/cours-le-processus-unifie
- http://dspace.univ-tlemcen.dz/bitstream/112/5500/5/chapitre1.pdf
- https://www.africmemoire.com/part.5-chap-iii-modelisation-902.html
- https://www.memoireonline.com/11/19/11304/m_Conception-d-un-systeme-de-gestion-de-worlflow-graphique19.html
- https://www.uv.es/nemiche/cursos/polycopies/5%20Merise.pdf
- https://www.memoireonline.com/04/17/9779/m_Mise-en-place-dun-systeme-informatise-de-transfert-dargent-cas-de-go-sarl10.html
- https://www.developpez.net/forums/d1708589/general-developpement/alm/modelisation/uml/autres-diagrammes/diagramme-sequence-probleme/
- https://www.memoireonline.com/04/15/9024/m_Conception-et-developpement-dune-application-web-sur-la-gestion-du-cursus-scolaire-des-eleves14.html