REPUBLIQUE DU SENEGAL

UNIVERSITE ASSANE SECK DE ZIGUINCHOR

L'excellence, ma référence

UFR : SCIENCES ECONOMIQUES ET SOCIALES

Département : Economie – Gestion

MEMOIRE MASTER 2

Mention : Management des systèmes d'information

Spécialité : Méthodes Informatiques Appliquées à la Gestion des Entreprises (MIAGE)

Thème :

Conception et implémentation d'un système d'information pour la gestion de l'hébergement des étudiants de l'UASZ

Présenté et soutenu par :

M. Malick SEYE

Sous la direction de

M. Serigne DIAGNE,

Maitre-Assistant

Devant un jury composé de :

Président :	
M. Babacar NDIAYE	Maitre de conférences agrégés
Examinateurs :	
M. Ibrahima DIOP	Assistant
M. Bassirou DIENE	Enseignant-Chercheur
M. Serigne DIAGNE	Maitre-Assistant

28 Février 2017

Résumé

A l'instar des autres Universités du Sénégal, les crises qui gangrènent l'Université Assane Seck de Ziguinchor ces dernières années émanent, pour l'essentiel, du volet social alors que d'importantes ressources humaines, matérielles et financières ont été mobilisées par les pouvoirs publics afin de mettre les étudiants dans de meilleures conditions de vie et d'études.

Concernant le campus social, des lenteurs ainsi que des pertes d'informations sont notées dans sa gestion en raison de l'inadaptation du système actuel aux méthodes offertes par les Technologies de l'Information et de la Communication.

Ces travaux de mémoire de fin d'études ont justement pour objectif d'apporter une solution à ce problème en concevant et en mettant en place un système d'information pour la gestion du campus social du Centre Régional des Œuvres Universitaires et Sociales de Ziguinchor.

Pour la réalisation de ce travail, nous avons fait l'état de l'art et choisi UML pour la modélisation. Les modèles obtenus ont été intégrés via la plateforme JEE sous une base de données MySQL et Tomcat fait office de serveur d'application. L'application permettant d'automatiser la gestion de l'hébergement des étudiants bénéficiaires de lits est présentée grâce à des captures d'écran afin de montrer ses fonctionnalités.

Mot clés : CROUS/Z, Gestion campus social

Abstract:

Like the other universities in Senegal, the crises that have plagued the Assane Seck University in Ziguinchor in recent years are essentially social in nature, while significant human, material and financial resources have been injected Public authorities in order to give students better living and study conditions.

Concerning the social campus, delays and loss of information are noted in its management due to the inadequacy of the current system to the methods offered by the Information and Communication Technologies.

The aim of this dissertation study is to solve this problem by designing and implementing an information system for the management of the social campus of the Regional Center for University and Social Works in Ziguinchor.

For the realization of this work, we made the state of the art and chose UML for modeling. The models obtained were integrated via the JEE platform under a MySQL database and Tomcat acts as an application server. The application allowing to automate the management of the accommodation of students benefiting from beds is presented through screenshots in order to show its functionalities.

Key words: CROUS / Z, Social campus management

Dédicaces

Je dédie ce travail à :

- ✓ Ma femme Sokhna Anta SAMB pour sa compréhension, son engagement à me soutenir tout au long de ce travail et son esprit de dépassement ;
- ✓ Ma mère Sokhna DIOP qui m'a donné une bonne éducation de base et m'a inculqué la patiente
- ✓ Mon père Mohamed SEYE qui n'a ménagé aucun effort pour notre réussite, il est l'unique personne qui a pris en charge toutes mes études ;
- ✓ Mon frère Mamadou SEYE que ce travail soit pour toi un repère dans ton chemin ;
- ✓ Mes sœurs Thierno NDIAYE, Fama DIOP et Absa SEYE ;
- ✓ Jamatou Hizboulahi Li Khitmatil Khadim plus particulièrement Dahiratou Salam ;
- ✓ Mon Cheikh Cheikhouna Cheikh Oumar FALL qui sans lui ce travail ne verra pas le jour, son soutien moral est une importante capitale ;
- ✓ Tante Ndeye Yaka SANE, ma mère à Ziguinchor ;
- ✓ Dahira Matlaboul Fawzeyni de l'UASZ qui m'a accueilli dans cette institution ;
- ✓ Mes oncles et tantes ;
- ✓ Mes grands-mères et grands-pères ;
- ✓ Mes cousines et cousins ;
- ✓ Mes amis;
- ✓ Mes camarades de classes durant tout mon cursus scolaire;
- ✓ Mes promotionnaires Seydina Oumar DIAGNE, Vieux Moussa DIEDHIOU, Ousmane BALDE, Ndeye Anthia SECK, Ndiaté SENE, Mariéme COULIBALY, Les Haïtiens, etc.
- ✓ L'Amicale des étudiants Ressortissants de Mbour à Ziguinchor ;
- ✓ Toi qui lis ce mémoire.

Remerciements

- ✓ *Je remercie tout d'abord Allah (mon Dieu) de m'avoir donné la capacité de réfléchir et d'écrire, la force d'y croire, la patience d'aller jusqu'au bout du rêve et je prie sur son prophète (PSL)*

Un merci à

- ✓ *Celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, ma mère Sokhna DIOP*
- ✓ *Mon père Mohamed SEYE, école de mon enfance, qui a été mon ombre durant toutes mes années d'études, et qui a veillé tout au long de ma vie, à m'encourager, à me donner l'aide et à me protéger.*
- ✓ *Mon guide spirituel Cheikh Sidy Mokhtar MBACKE qui m'a enseigné la quintessence de la vie ;*
- ✓ *Mon cheikh, celui qui n'a ménagé aucun effort pour ma réussite dans tous les domaines, je veux nommer Cheikh Oumar FALL ;*
- ✓ *Sokhna Anta SAMB, ma femme, ma meilleure amie pour sa patience ;*
- ✓ *Que dieu les gardes et les protège.*
- ✓ *Mon encadrant Dr Serigne DIAGNE, enseignant-chercheur à l'UASZ qui a accepté de m'accompagner humblement dans la réalisation de ce travail, de consacrer tout son temps pour que ce mémoire compte parmi les meilleurs de l'institution. Que Dieu vous assiste tout au long vos projet.*
- ✓ *Membres du jury d'avoir acceptés de lire et d'évaluer ce mémoire ;*
- ✓ *Mes professeurs qui ont participé à ma formation ; M. A BADJI, M. A DIATTA, M. B DIENE, Dr B BASSE, Dr C O BALDE, Dr Kh GAYE, Dr M NDIAYE, Dr D NDOUR, Dr M MENDY, Dr Y FAYE, Dr Y DIENG, M. E KABOU, etc. Je réserve un remercie avec toute gratitude au Directeur de l'UFR SES, Prof Babacar NDIAYE pour ses conseil et son engagement sans faille.*
- ✓ *Mes familles d'accueil à Ziguinchor ; Ibra TOURE, El Hadji SOW et Malamine DIEDHIOU ;*
- ✓ *Tout le personnel du CROUS de Ziguinchor mention spéciale à M. GASSAMA ;*
- ✓ *Tout le personnel de l'UFR ST plus particulièrement à M. Alioune Badara DIENG, Chef de service pédagogique et à Mme MANE Hady MBACKE, Chef de service administration de m'avoir accueilli dans une ambiance chaleureuse de travail et leur soutien matériel ;*

- ✓ *Etudiants qui m'ont accueilli : Yaya DIALLO, Souleymane DIALLO, Oumou KEITA etc.*
- ✓ *Modou DIOP, étudiant M1 en Génie logiciel et Dame DIAW, ingénieur en informatique pour leur soutien technique ;*
- ✓ *Toutes les personnes qui de près ou loin ont contribué à l'école de ma vie.*

Table des matières

Résumé 1
Abstract: 2
Dédicaces 3
Remerciements 4
Table des matières 6
Liste des figures 8
Liste des tableaux : 9
Liste des abréviations 10
Introduction générale 11
1. Contexte : 11
2. Problématique : 11
3. Objectif : 12
4. Organisation du mémoire : 12
Chapitre 1 : Etat de l'art 14
1. Contexte et Problématique 14
1.1. Présentation de la structure : 14
1.2. CROUS de Ziguinchor : mode de gestion actuel 15
1.3. Problématique : 16
1.4. La solution : Critique et proposition de solution 17
2. Système d'information : 17
2.1. Les différentes méthodes d'analyse : 18
2.2. Choix de la méthode à utiliser : 23
Conclusion : 24
Chapitre 2 : **Spécification et analyse des besoins fonctionnels** 25
1. Acteurs : 25
2. Fonctionnalités : 25
3. Diagramme de cas d'utilisation : 30
3.1. L'attribution des lits : 31
3.2. Le payement et suivi : 32
3.3. L'expression des besoins et suivi : 33
4. Diagramme de séquence : 34
4.1. S'authentifier : 34
4.2. Insérer les listes par ordre mérite : 35

4.3. Consulter les listes des bénéficiaires : 36
4.4. Enregistrer payement: 37
4.5. Valider une codification : 38
4.6. Faire une réclamation : 39
4.7. Consulter les états de payement : 40
5. Diagramme de classe : 41
6. Modèle logique de données : 44
Conclusion 46
Chapitre 3 : Implémentation du modèle 47
1. Architecture 47
1.1. Couche présentation : 47
1.2. Couche métier : 48
1.3. Couche accès aux données : 48
2. Les langages utilisés : 48
2.1. Framework Spring MVC : 49
2.2. Langage SQL : 52
3. La sécurité : 52
4. Le Système de Gestion de Base de Données utilisé : 53
5. Le serveur : 54
6. Les outils logiciels : 54
Conclusion 55
Chapitre 4 : Présentation du système informatique obtenu 56
Prise en main du système 56
1. Menus de l'application 56
2. Interfaces de l'application 57
Conclusion 70
Conclusion générale 71
1- Bilan 71
2- Critique 71
3- Perspectives 72
Bibliographie: 73

Liste des figures

Figure 1: Attribution des lits 31
Figure 2: Payement et suivi 32
Figure 3 : Expression des besoins et suivi 33
Figure 4 : S'authentifier 34
Figure 5: Insérer les listes par ordre de mérite 35
Figure 6: Consulter les listes des bénéficiaires 36
Figure 7: Enregistrer paiement 37
Figure 8: Valider une codification 38
Figure 9: Faire une réclamation 39
Figure 10: Consulter les états de payement 40
Figure 11: Diagramme de classe 42
Figure 12: Architecture trois tiers **[B9]** 47
Figure 13: Les différents modules de Spring **[B13]** 50
Figure 14: Architecture Spring **[B14]** 53
Figure 15 : Menu Accueil 56
Figure 16 : Menu Admin 56
Figure 17 : Menu UFR 56
Figure 18 : Menu Chef de résidence 57
Figure 19 : Menu Chef de pavillon 57
Figure 20: Menu Agent comptable 57
Figure 21: Menu Bénéficiaire 57
Figure 22: Interface Accueil 58
Figure 23: Recherche d'un bénéficiaire dans une classe 58
Figure 24: Liste des bénéficiaires d'une classe 59
Figure 25: Interface Connexion du système 59
Figure 26: Interface Administrateur : 60
Figure 27: Liste des utilisateurs 61
Figure 28:Liste des comptes 61
Figure 29: Liste des rôles 62
Figure 30: Interface chef de résidence : 63
Figure 31:Ajout des résidences : 64
Figure 32:Liste des lits 64
Figure 33: Interface chef de service pédagogique 65
Figure 34:Liste des bénéficiaires des classes 65
Figure 35: Interface Agent comptable 66
Figure 36:Liste des payements 66
Figure 37: Interface pavillon 67
Figure 38: Bénéficiaire déjà payé 68
Figure 39: Réclamation des bénéficiaires 68
Figure 40: Bénéficiaires d'un pavillon 68
Figure 41: Interface bénéficiaire 69
Figure 42: Etat paiement bénéficiaire 69
Figure 43: réclamation d'un bénéficiaire 70

Liste des tableaux :

Tableau 1: Cas d'utilisation S'authentifier 27
Tableau 2: Cas d'utilisation Gérer les résidences 28
Tableau 3: Cas d'utilisation Consulter la liste des bénéficiaires 28
Tableau 4: Cas d'utilisation les états de payement 29
Tableau 5: Cas d'utilisation Saisir les listes cas sociaux 29

Liste des abréviations

UASZ: Université Assane SECK de Ziguinchor.

UFR: Unité de formation et de recherche.

COUD: Centre des Œuvres Universitaires de Dakar.

CROUS: Centre Régional des Œuvres Universitaires et Sociales.

SAL: Service d'Accueil et Logement.

S.A.D.T: **S**tructured **A**nalysis and **D**esign **T**echnic

MERISE: Méthodes d'Etude et de Réalisation Information pour les Systèmes d'Entreprise

MCC: Modèle Conceptuel de Communication.

MCD: Modèle Conceptuel de Données.

MOT: Modèle Organisation de Traitement.

UML: Unified Modeling Language

TIC: Technologies de l'Information et de Communication.

CSS: Cascading Style Sheets.

JS: Java Script

XML: Extensible Markup Language.

MVC: Model View Controller.

IHM: Interface Homme Machine.

JSP: Java Server Pages.

2 TUP: Two Track Unified Process.

SGBD: Système de Gestion de Base de Données.

SQL: Structured Query Language.

MySQL: My Structured Query Language.

LDD : Langage de Définition de données

LMD : Langage de Manipulation de données

LCD : Langage de contrôle de données

Introduction générale

1. Contexte :

L'université Assane Seck de Ziguinchor (UASZ) compte aujourd'hui plus de 4000 étudiants. Ce nombre augmente chaque année. Pour faire face à cette augmentation vertigineuse des effectifs estudiantins, l'Etat a doté les nouvelles universités de centres régionaux des œuvres sociales. Ces nouvelles structures ont pour vocation d'être au service des étudiants dans les domaines de l'hébergement, de la restauration et du volet médical, missions jadis prises en charge le Centre des Œuvres Universitaires de Dakar (COUD).

La volonté de l'Etat de mettre en place ses nouvelles structures résulte en partie des dysfonctionnements noté avec le portage par le COUD. A l'UASZ, on peut citer, parmi ces sources de dysfonctionnements, un recours insuffisant aux TIC au niveau des différents services de gestion des œuvres sociales qui impacte négativement sur l'efficacité et l'efficience des services rendus aux étudiants.

Ainsi pour accomplir ses missions avec plus d'efficacité, le Centres Régionales des Œuvres Universitaires et Sociales (CROUS) de Ziguinchor a intérêt à s'adapter d'ores et déjà à la réalité de l'ère numérique. En effet, étant donné que l'informatique est de nos jours incontournable dans le fonctionnement de toute institution qui se veut pérenne, il est impératif que la structure puisse s'approprier des TIC pour optimiser les résultats de ses différentes activités.

Aussi pour atteindre ses objectifs, il est nécessaire que le CROUS procède à l'aide de l'outil informatique à une planification des moyens et des actions à mener. Et c'est toute la signification qu'il faut donner à la conception d'un système d'information permettant l'automatisation des tâches liées à la gestion du campus social.

2. Problématique :

La gestion du campus social devient chaque année plus complexe avec l'augmentation de nouveaux pavillons et du nombre d'étudiants. Ainsi chacune de ces évolutions rend plus lourd les charges de travail du personnel car il va valoir prendre en considération à tout instant la corrélation entre ces entités.

Canaliser l'information dans une organisation demeure une question fondamentale. Le CROUS, dans sa gestion, manipule beaucoup d'information dont le campus social qui constitue

le centre de gravité de sa mission. Ainsi, structurer le stockage et la circulation de cette banque d'information revient à réduire les tâches de cette dite organisation étatique.

Mieux, toujours dans son fonctionnement, il interagit avec des acteurs externes (partenaires) tels que les étudiants, les UFR, etc. De ce fait, la manière dont l'information est utilisée, suscite davantage des réflexions.

En effet, répondre à ces questions suivantes permettrait de mieux cerner et atteindre l'objectif que nous nous sommes fixés.

- ✓ ***Quel type d'information le CROUS de Ziguinchor peut-il générer ?***
- ✓ ***Comment l'information devra- t-elle être collectée, traitée, stockée et diffusée ?***
- ✓ ***Comment l'information relative aux partenaires est-elle gérée ?***
- ✓ ***Comment les automatiser ?***

3. Objectif :

La mise en place de ce système d'information permettra non seulement une circulation plus facile des informations mais aussi va aider le directeur et ses collaborateurs à prendre des meilleures décisions au moment opportun.

En effet, l'objectif général de cette étude est de Concevoir un système d'information pour **La gestion du campus social** du Centre Régional des Œuvres Universitaires et Sociales de Ziguinchor. Pour atteindre cet objectif, nous le déclinons en objectifs spécifiques que sont :

- ✓ ***Attribuer les chambres / lits :***
  - ♦ ***Mérite,***
  - ♦ ***Cas sociaux.***
- ✓ ***Suivre le payement des droits de logement :***
- ✓ ***Exprimer et suivre les besoins des bénéficiaires*** :

4. Organisation du mémoire :

Ce mémoire est divisé en quatre(4) chapitres :

- Le premier chapitre s'intitule l'état de l'art, et fait d'abord une présentation de la structure en question, ensuite pose le problème et son mode de gestion actuel, puis faire une critique de ce mode de fonctionnement et en fin une étude des méthodes de conception des système d'information est faite afin de choisir celle que nous allons utiliser dans le chapitre suivant.

- Le deuxième chapitre concerne la Modélisation de notre système d'information qui montre les trois types diagrammes (Cas d'utilisation, Séquence et Classe) que nous avons utilisé.
- Implémentation de l'application à travers les modèles conçus est présentée dans le troisième chapitre. Ici nous avons présenté l'architecture utilisée, les différents outils de modélisation, les langages utilisés et quelques contraintes.
- Le quatrième et dernier chapitre porte sur la Présentation du système obtenu après l'implémentation des modèles.

Chapitre 1 : Etat de l'art

1. Contexte et Problématique

1.1. Présentation de la structure :

L'université Assane SECK de Ziguinchor a ouvert ses portes en février 2007 avec un effectif de 257 étudiants répartis dans trois Unités de Formation et de Recherche (UFR) que sont: l'UFR des Sciences et Technologies, l'UFR des Sciences Economiques et Sociales et l'UFR des Lettres, Arts et Sciences Humaines. En 2012, l'UFR des Sciences de la Santé a ouvert ses portes. Cette jeune université est implantée dans le quartier périphérique de Diabir, dans la commune de Ziguinchor. Lieu approprié pour la recherche fondamentale et la recherche-développement, pour la production et la diffusion de connaissance indispensable au Sénégal. L'Université Assane SECK de Ziguinchor contribue fortement à valoriser les atouts économiques et culturels de la Casamance, à créer les conditions d'une paix durable dans cette région, qui, non seulement recèle d'importantes ressources culturelles et humaines, mais aussi constitue, eu égard à sa position géographique, un axe stratégique d'intégration sous régionale.

En effet, comme toute université digne de ce nom, le volet social est géré par le Centre des Œuvres, cette jeune institution ne serait pas en reste. Dès sa création, l'université Assane SECK de Ziguinchor était portée par le COUD jusqu'à ce que la loi n° 2016-08 portant création des CROUS de Ziguinchor, de Bambey et de Thiès soit votée à l'assemblée nationale lui accordant son autonomisation.

Le Centre Régional des Œuvres Universitaires et Sociales de Ziguinchor (CROUS) est un établissement public à caractère administratif dont le but est d'assister les étudiants de l'université et de réglementer les prestations versées aux bénéficiaires des œuvres universitaires.

Le CROUS a pour objectif de favoriser l'amélioration des conditions de vie et d'étude des étudiants de l'UASZ sur les plans social, culturel et sportif. Il assure particulièrement la restauration, le logement et l'amélioration de la couverture médicale et sanitaire dans le campus. Il est composé de plusieurs services parmi lesquels le service Accueil et Logement dont l'automatisation de son fonctionnement constitue l'objet même de notre réflexion.

Le Service Accueil et Logement (SAL) est le service qui s'occupe de l'accueil de l'étudiant et de son hébergement dans le campus universitaire. C'est la fusion entre le Service du Bénéfice des Œuvres et le Service des Cités qui a donné naissance à ce service (SAL). Il peut être considéré comme la porte d'entrée des étudiants aux services du CROUS. Les cités

universitaires sont mixtes. Elles sont ouvertes du 1er Octobre au 31 Juillet et sont fermées du 1er Aout au 30 Septembre de chaque année. Le SAL se fixe comme objectif de :

- ✓ Octroyer des bénéfices des œuvres aux étudiants ;
- ✓ Donner des logements dans la limite des places disponible ;
- ✓ Accueillir et orienter les étudiants au sein de la cité universitaire ;

1.2. CROUS de Ziguinchor : mode de gestion actuel

1.2.1. Attribution des lits :

L'attribution des lits est une procédure permettant d'octroyer les lits aux étudiants communément appelés bénéficiaires des œuvres sociales. Cette attribution se fait selon l'ordre de mérite et une partie des lits est réservée aux cas sociaux. En début de chaque année universitaire, la commission codification se réunit pour procéder à l'octroi des lits.

La commission codification est composée du chef de service hébergement, des délégués d'étudiants et des Chefs de service pédagogique de chaque UFR. Elle fait l'attribution des lits selon le mérite et tient compte des cas sociaux.

- ➢ Mérite : Les lits sont octroyés par ordre de mérite selon les résultats annuel de l'année précédente. La répartition se fait comme suit :
  - Le nombre de lits par classe : ***(effectif de la classe * nombre de lits (cota mérite)) / effectif total des étudiants***.
  - Le nombre de lits par genre : ***nombre de fille ou garçon * Le nombre de lits de la classe / effectif de la classe.***

Après répartition, l'attribution se fera par ordre de mérite suivant la moyenne annuelle et le mode de validation donc selon le rang.

- ➢ Social : La partie restante des lits est réservée à l'amicale des délégués et aux étudiants remplissant les critères sociaux établis par la commission codification. A la sortie de cette commission, le chef de service hébergement introduit les listes dans le fichier.

1.2.2. Le paiement et suivi des droits de logement

- ➢ Paiement : Les bénéficiaires ont l'obligation de verser une somme de 12 000 f cfa au moment de la codification répartie comme suit : caution (5000 f), mensualité entrée et sortie (6000 f) et un guide d'étudiant (1000 f). Ils payent aussi chaque mois 3000 f cfa le lit. Apres paiement, le

bénéficiaire a doit à une clé, un matelas, deux draps, une couverture, une chaise et une table.

- Suivi : Le chef de résidence et ses collaborateurs ont la lourde tâche de faire payer le bénéficiaire ladite somme avant d'occuper son lit. Ils veillent aussi au suivi du paiement des mensualités.

1.2.3. Expression des besoins et suivi : entretien

- Expression des besoins : Le bénéficiaire doit signaler tous les manquements constatés dans la chambre au niveau de son chef de pavillon.
- Suivi : Les chefs de pavillon, après avoir noté les manquements, doivent saisir les ouvriers afin qu'ils passent au niveau des chambres concernées et effectuent les réfections.

1.3. Problématique :

La gestion du campus social se fait toujours de façon manuelle car pour la réalisation des différentes tâches, le personnel utilise des carnets de convention, des dossiers résidences, des quittances, des fiches de déclamations et travaux effectués, des bons d'entrées, des registres d'enregistrement etc.

Le service a en charge la gestion des pavillons, qui sont répartis en chambres, et chaque chambre dispose de deux lits. En début d'année, le comptable matières met à la disposition des chefs de pavillons des matelas, des draps et des couvertures, qui procèdent à leur distribution. Le chef de service hébergement reçoit des chefs de service pédagogique de chaque UFR les effectifs des étudiants éligibles (de la Licence 1 au Master 2) ; de même, les effectifs et le nombre de filles et garçons par classe.

Le service en place ne dispose d'aucune application, il utilise en plus des cahiers (registre, carnet, etc.), des feuilles Excel et l'archivage est toujours archaïque. De ce fait, il sera difficile de faire une quelconque recherche ou de détecter aisément un problème particulier.

Le retard des codifications est causé par la réception tardive des listes de passage des étudiants. Chaque année le rectorat par les biais des UFR veut attendre toutes les listes de passage avant de l'envoyer au niveau du CROUS.

Le manque d'infrastructure pour le personnel du CROUS : bureaux, matériels de bureau, magasins. Jusqu'à présent le CROUS ne dispose pas de locaux au sein de l'université.

Au regard de tout ce qui précède, mettre en place une plateforme taillée sur mesure est nécessaire voire obligatoire afin de pouvoir palier à ses manquements. Ce système d'information devra permettre de faciliter la circulation des flux d'informations liées à l'attribution des lits (mérite et cas social), le suivi des paiements des droits de logement et à l'expression des besoins des bénéficiaires (et suivi).

1.4. La solution : Critique et proposition de solution

- **Critique** :

Malgré le manque d'outils, le service se débrouille avec les moyens du bord pour atteindre ses objectifs qui sont rappelons les : l'attribution des lits et le suivi quotidien au sein du campus. Le système en place bien qu'il soit manuel, il réussit toujours la quasi-totalité des attentes des bénéficiaires mais avec une perte de temps conséquente.

Toutefois tout système a des limites encore plus un système purement manuel. Nous notons une certaine lenteur dans l'exécution du travail, beaucoup d'erreurs (des doublons, des oublis, des omissions, etc.), un manque de suivi des besoins exprimés par les bénéficiaires.

- **Proposition de solution** :

Après analyse et critique de l'existant, nous avons jugé nécessaire d'apporter une solution à cette problématique. La solution est de mettre en place un système d'information qui gère tous les processus de la gestion du campus social du Centre des Œuvres Universitaires et Sociales de l'Université Assane SECK de Ziguinchor.

Nous mettrons en place un système qui intègre en même temps la codification et toutes les tâches qui vont avec. Ainsi, les interactions entre les acteurs seront plus cohérentes.

2. Système d'information :

Le système d'information (SI) est un "ensemble d'éléments (personnel, matériel, logiciel...) permettant d'acquérir, traiter, mémoriser et communiquer des informations". **[B1]**

Le SI est un outil de communication et de coordination entre les différents services et domaines de gestion de l'entreprise. Il doit produire et diffuser des informations nécessaires aux opérations d'une part et aux choix stratégiques et tactiques d'autre part. Le S.I doit permettre de connaitre le présent, de prévoir, de comprendre le fonctionnement et d'informer rapidement.

Le système d'information a pour objectif de restituer au différent membre de l'entreprise les informations sous une forme directement utilisable afin de faciliter la prise de décision.

Pour optimiser les informations, le S.I. doit remplir quatre tâches spécifiques :

- ✓ La collecte : l'origine de l'information peut être externe ou interne via les diapositives techniques mises en place.
- ✓ Le stockage : L'information doit pouvoir être disponible et pérenne.
- ✓ Le traitement : le choix du support utilisé puisqu'il va falloir trouver un formalisme pour traiter l'information : Soit la centralisation, soit la décentralisation, soit la distribution.
- ✓ La diffusion : Elle doit répondre à quatre critères : origine et destination, forme, délai et dimension.

Pour comprendre la manière dont l'information circule au sein d'une organisation, nous allons étudier les différentes méthodes d'analyse et de conception des systèmes d'information.

2.1. Les différentes méthodes d'analyse :

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. Elle est aussi la conception d'un modèle selon son objectif et les outils utilisés. Le processus de modélisation vise à obtenir une solution acceptable du système informatique. La solution finalement retenue n'est pas obtenue en une seule itération. Plusieurs étapes sont nécessaires ; ces étapes successives permettent de raffiner le niveau de détails du système à réaliser. Les premières étapes donnent une vision globale et permettent d'avancer dans la compréhension du problème. [**B2**]

Sur ce, il existe beaucoup de méthodes, mais nous allons en citer quelques-unes afin de pouvoir choisir une qui sera utilisée dans ce mémoire.

2.1.1. Les méthodes cartésiennes :

L'analyse par les traitements trouve sa source dans la notion de méthode dans le sens où celle-ci se propose de résoudre un problème grâce à un cadre procédural. Ce cadre fixe les étapes à respecter, leur enchaînement, ainsi que les entrées et les sorties correspondant à chaque étape. Cette école de pensée est essentiellement **cartésienne**. Pour nous, l'analyse par structuration des traitements est issue de cette ligne de pensée.

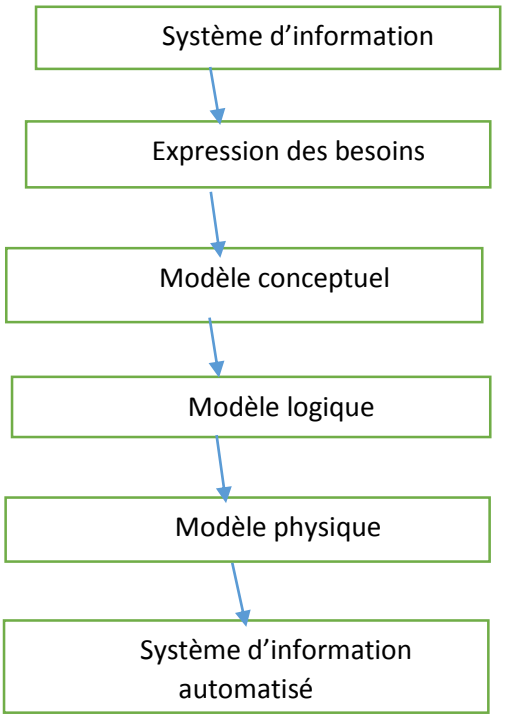
La méthode S.A.D.T (**S**tructured **A**nalysis and **D**esign **T**echnic) répond à ces critères. . Cette méthode a été mise au point par la société Softech aux Etats Unis. La méthode SADT est une méthode d'analyse par niveaux successifs d'approche descriptive d'un ensemble quel qu'il soit. On peut appliquer le SADT à la gestion d'une entreprise tout comme à un système automatisé**. [B3]**

2.1.2. Les méthodes classiques ou systémiques :

L'analyse en structures de données étudie le problème à modéliser en mettant l'accent sur les relations entre les éléments modélisés dans le logiciel. Cette approche est celle inaugurée par l'école **systémique.** Le logiciel est avant tout conceptualisé comme une composante du système d'organisation global de l'entreprise. Sa conception débute, comme toute analyse par la définition du système et de ses frontières.

MERISE est une méthode de conception, de développement et de réalisation de projets informatiques. Le but de cette méthode est d'arriver à concevoir un système d'information. La méthode MERISE est basée sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et physiques .La séparation des données et des traitements assurent une longévité au modèle. En effet, l'agencement des données n'a pas à être souvent remanié, tandis que les traitements le sont plus fréquemment.

La conception du système d'information se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. Il s'agit donc de valider une à une chacune des étapes en prenant en compte les résultats de la phase précédente. D'autre part, les données étant séparées des traitements, il faut vérifier la concordance entre données et traitements afin de vérifier que toutes les données nécessaires aux traitements sont présentes et qu'il n'y a pas de données superflues. [**B4**]



L'expression des besoins est une étape consistant à définir ce que l'on attend du système d'information automatisé, il faut pour cela :

- ✓ faire l'inventaire des éléments nécessaires au système d'information
- ✓ délimiter le système en s'informant auprès des futurs utilisateurs

Cela va permettre de créer le **MCC** (Modèle conceptuel de la communication) qui définit les flux d'informations à prendre en compte.

L'étape suivante consiste à mettre au point le **MCD** (Modèle conceptuel des données) et le **MCT** (Modèle conceptuel des traitements) décrivant les règles et les contraintes à prendre en compte.

Le modèle organisationnel consiste à définir le **MOT** (Modèle organisationnel des traitements) décrivant les contraintes dues à l'environnement (organisationnel, spatial et temporel).

Le modèle logique représente un choix logiciel pour le système d'information.

Le modèle physique reflète un choix matériel pour le système d'information.

2.1.3. Les méthodes objets :

Les paradigmes de représentation que nous venons d'exposer, permettent de conduire la démarche de développement de façon différente et de pallier dans une large mesure les inconvénients des méthodes classiques ; cette nouvelle méthode est la méthode objet.

L'analyse orientée objet s'articule autour de quatre principes d'intelligibilité du réel :

- ✓ La modélisation à partir d'entités qui sont des instances d'une *classe d'objets*
- ✓ *L'encapsulation* à l'intérieur de ces objets des données : les *attributs* et des procédures de traitement de ces données : les *services*
- ✓ La classification de ces objets les uns par rapport aux autres à partir de trois types de relations : *inclusion*, *connexion* et *héritage*
- ✓ La coopération des objets les uns avec les autres par de biais de *messages*.

La communication entre informaticiens et non-informaticiens est meilleure dans la mesure où le concept d'objet est plus intuitif. D'autre part, les langages objets sont plus proches des schémas de représentation et permettent de mettre facilement en œuvre les concepts de l'analyse objet. En d'autres termes, les modèles utilisés pour la description du problème sont très proches des systèmes de programmation.

L'encapsulation permet de progresser dans la résolution de l'incompatibilité entre structuration des traitements et structuration des données puisqu'un objet en constitue une synthèse. L'approche orientée objet permet de substituer à la dualité données et procédures une structure unique. Toutefois, les schémas de représentation de type données et ceux de type traitements sont toujours utilisés dans les phases d'analyse et de conception. Cependant, le premier souci de l'analyste n'est plus de trouver les fonctionnalités du système, mais d'identifier les objets. Il faut ensuite attribuer les procédures à ceux-ci. Les priorités sont radicalement différentes : On s'appuie sur la structure sous-jacente du domaine d'application plutôt que sur des besoins fonctionnels liés à un seul problème.

Les phases d'études préliminaires sont raccourcies et la conception se fait par un cycle en spirale au lieu d'un cycle en V. En effet, l'héritage permet de tester facilement un objet avant même de le définir complètement. Le temps global consommé par les activités de développement n'est par contre pas plus court en objet : les gains se font par la suite lors des phases de maintenance. D'ailleurs la réversibilité des choix est aisée par l'introduction de

nouveaux objets ou l'ajout d'attributs ou de services. Finalement la nature décentralisée de l'architecture objet permet de répartir les décisions tout le long du cycle de développement. Il s'agit plus d'un travail par raffinages successifs qu'un effort de conversion d'une représentation à une autre. On peut parler à ce sujet de processus 'sans rupture'.

Le langage UML (Unified Modeling Language) répond à ces critères. Le langage de modélisation unifié est un langage de modélisation graphique à base de pictogrammes conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet.

La notation UML est un **langage visuel** constitué d'un ensemble de schémas, appelés des **diagrammes**, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour **représenter** le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

Les diagrammes sont dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie. Il en existe ***13*** (depuis UML 2.3). [**B5**]

- Diagrammes de structure ou diagrammes statiques :
  - Diagramme de classes : représentation des classes intervenant dans le système.
  - Diagramme objet : représentation des instances de classes (objets) utilisées dans le système.
  - Diagramme de composants : représentation des composants du système d'un point de vue physique, tels qu'ils sont mis en œuvre (fichiers, bibliothèques, bases de données…)
  - Diagramme de déploiement : représentation des éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage …) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.
  - Diagramme des paquets : représentation des dépendances entre les paquets (un paquet étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML), c'est-à-dire entre les ensembles de définitions.
  - Diagramme de structure composite : représentation sous forme de boîte blanche les relations entre composants d'une classe.

- Diagramme de profils : spécialisation et personnalisation pour un domaine particulier d'un méta-modèle de référence d'UML.

➢ Diagrammes de comportement :

- Diagramme des cas d'utilisation : représentation des possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire de toutes les fonctionnalités que doit fournir le système.
- Diagramme états-transitions : représentation sous forme de machine à état le comportement du système et ses composants.
- Diagramme d'activité : représentation sous forme de flux ou d'enchainement d'activités le comportement du système ou de ses composants.

➢ Diagrammes d'interaction ou diagrammes dynamiques :

- Diagramme de séquence : représente de façon simplifiée une fonctionnalité et se concentrant sur les échanges de messages entre les objets.
- Diagramme global d'interaction : représentation des enchainements possibles entre les scénarios préalablement identifiés sous de diagramme de séquences.
- Diagramme de temps : représentation des variations d'une donnée au cours du temps

2.2. Choix de la méthode à utiliser :

Nous pouvons dire d'après tout ce qui précède que l'invention de langages de programmation adaptés à la résolution de problèmes de simulation a permis l'émergence d'un nouveau mode de représentation. Celui-ci repose sur les concepts tels que la modélisation d'un problème comme une interaction entre objets coopérants munis à la fois de services et d'attributs; ou encore la création d'objets à partir de classes abstraites pouvant hériter des propriétés d'autres classes.

De même que les écoles cartésiennes et systémiques ont profondément influencé les paradigmes de représentation classiques, le paradigme 'orienté objet' a un impact important sur le développement des applications informatiques. En effet, c'est à partir de ces paradigmes que se construisent les applications. Ces nouvelles représentations changent non seulement la forme des schémas utilisés, mais permettent également de faire évoluer le métier même de l'informaticien : relation avec les clients, division des tâches, architecture des logiciels.

En termes d'analyse et de conception de SI (Système d'information), le langage UML est aujourd'hui devenu un standard incontournable. Il permet de faire une modélisation graphique

à base de pictogramme. C'est un langage de modélisation orienté objet et est constitué aujourd'hui de 13 diagrammes. Ces diagrammes couvrent presque la totalité des besoins de modélisation potentiellement nécessaires à la conception des logiciels

Nous avons choisi UML parce que c'est un langage qui est assez complet avec ses divers diagrammes qu'il propose. Ces diagrammes peuvent accompagner le développeur de la phase de la conception jusqu'au déploiement de l'application. Le langage UML peut aussi nous aider à aller dans les détails séquentiels de chaque cas d'utilisation. Il permet de plus à migrer vers l'implémentation de l'application beaucoup plus facilement.

Conclusion :

Nous avons pu montrer dans ce chapitre les failles liées à la gestion du campus social du CROUS de Ziguinchor tout en faisant une étude comparative des méthodes de conception et de choisir la modélisation objet.

Chapitre 2 : Spécification et analyse des besoins fonctionnels

1. Acteurs :

- ✓ **Etudiant** : Il peut visiter la plateforme, il ne peut faire qu'une seule et unique action qui est de consulter la liste des bénéficiaires. Il n'a pas besoin de code accès, il accède directement à la plateforme. Sur ce, l'accès aux autres fonctionnalités de cette plateforme lui seront refusé.
- ✓ **Bénéficiaire** : Il est un étudiant avec un niveau d'accès plus avancé. Il accède à la plateforme via son code d'accès, il a la possibilité de consulter son état de payement, de faire une réclamation et suivi. De ce fait, l'accès aux autres fonctionnalités de la plateforme lui seront refusé.
- ✓ **Chef de service pédagogique** : Il représente le rectorat car il détient les listes de passage de chaque classe par ordre de mérite. Il se connecte avec son code d'accès, il n'a qu'une seule et unique possibilité ; d'insérer les listes par ordre de mérite. Bien entendu, l'accès aux autres fonctionnalités lui seront refusé.
- ✓ **Utilisateur** : C'est l'ensemble du personnel du CROUS qui interagisse avec notre système. Ils ont chacun un code d'accès qui leurs donnent des possibilités rétreintes.
    - **Agent comptable :** Il gère les payements de la caution et des mensualités.
    - **Chefs de pavillon** : Il valide les codifications, gère les réclamations.
    - **Chef de résidences** : En plus des possibilités des chefs de pavillon, il ajoute, modifie ou supprime les résidences, les pavillons, les chambres et les lits.
    - **Chef de service hébergement** : Il coiffe tout le service des cités. De ce fait, il détient toutes les possibilités du chef de résidence mais aussi il insère la liste des cas sociaux et les quotas.

2. Fonctionnalités :

- ✓ **S'authentifier** : Cette fonctionnalité permet aux acteurs d'accéder à la plateforme via leur profil. Excepté l'étudiant, tous les autres acteurs doivent s'authentifier.
- ✓ **Insérer les listes par ordre de mérite**: L'obtention des listes par ordre de mérite marque le début de la procédure de codification. Elles concernent seulement les étudiants éligibles (de la Licence 1 au Master 2). Chaque chef de service pédagogique insère la liste des classes dans la plateforme pour permettre au système d'attribuer le cota des méritants.

- ✓ **Saisir les listes des cas sociaux** : Les listes sont issues de la commission codification. Cette dernière définie des critères sociaux. Cette fonctionnalité complète la distribution des lits.
- ✓ **Gérer les résidences** : Ce cas permet de faire d'ajouter, de modifier ou de supprimer une résidence et composant (pavillon, chambre et lits) du campus social. Sans cette fonctionnalité la distribution des lits ne sera pas possible.
- ✓ **Gérer les quotas méritant et cas sociaux** : Ce cas permet d'ajouter, de modifier ou de supprimer le quota ; on octroie d'abord un nombre de lits pour les méritants et en suite pour les cas sociaux.
- ✓ **Consulter les listes des bénéficiaires** : Les listes de bénéficiaires seront disponibles après l'insertion du quota. Ce cas permet à tout utilisateur de pouvoir regarder la liste des bénéficiaires.
- ✓ **Enregistrer payement caution** : Chaque bénéficiaire doit verser une caution afin qu'il puisse accéder à son lit. Ce cas permet à l'agent comptable d'enregistrer ce payement.
- ✓ **Enregistrer payement mensualité** : Après le payement de la caution, ce cas permet à l'agent comptable d'enregistrer le payement des mensualités des bénéficiaires, ces derniers ont l'obligation de verser chaque mois les frais de loyer.
- ✓ **Valider une codification** : Cette fonctionnalité sera possible après l'enregistrement du payement de la caution, il permet de valider une codification afin que le bénéficiaire puisse avoir accéder à la plateforme. Car après la validation, le système envoie au bénéficiaire par mail son nom d'utilisateur et son mot de pass.
- ✓ **Consulter son état de payement** : Après la validation de leur codification par le chef de pavillon, cette fonctionnalité donne aux bénéficiaires la possibilité d'avoir une vue d'ensemble à tout moment sur sa situation actuelle.
- ✓ **Faire une réclamation** : cette fonctionnalité permet au bénéficiaire de signaler à temps un quelconque problème survenu dans sa chambre.
- ✓ **Visualiser les réclamations** : Ce cas permet de recenser les manquements au niveau des chambres exprimés par les bénéficiaires.
- ✓ **Signaler traiter**: Après chaque traitement d'une réclamation, la personne compétente (chef de pavillon) signale le problème résolu qui sera transmis au bénéficiaire.
- ✓ **Suivre sa réclamation** : Ce cas permet au bénéficiaire de bien suivre le problème qui a été déclaré.

- ✓ **Consulter les états de payement** : Ce cas permet d'avoir une vision globale sur le payement ; les bénéficiaires qui sont règle et ceux qui ne le sont pas.
- ✓ **Remplacer bénéficiaire** : Après l'attribution des lits, le bénéficiaire doit automatique venir payer ses droits de codification sinon, grâce à cette fonctionnalité, le système le remplace avec celui qui suit la liste d'entente.
- ✓ **Envoyer convocation** : Le bénéficiaire, après le payement de la caution, doit verser chaque une somme sinon, le système l'envoie une convocation.

Nous allons présenter sous forme de tableaux quelques fonctionnalités ou cas d'utilisateur afin de mieux cerner leur fonctionnement.

Titre	**S'authentifier**
Résumé	Ce cas d'utilisation permet à l'utilisateur d'avoir accès au système en fonction de son profil
Acteurs	Chef de service pédagogique, Bénéficiaire, Chef de résidence, Chef de pavillon, Agent comptable, Chef service hébergement
Prés Condition	L'utilisateur doit avoir un compte valable
Scénario nominal	• Clique sur se connecter • Saisir login et mot de pass • Le système vérifie les informations
Scénario alternatif	• Information incorrecte • Demande de renseigner à nouveau les informations
Post condition	Le système ouvre la page concernée

Tableau 1: Cas d'utilisation S'authentifier

Titre	Gérer les résidences
Résumé	Ce cas d'utilisation permet d'ajouter, de modifier ou de supprimer une résidence dans le système
Acteurs	Chef de résidence
Prés Condition	S'authentifier
Scénario nominal	• Demande formulaire d'ajout • Renseigner et valider
Scénario alternatif	• Information incorrecte • Demande de renseigner à nouveau les informations
Post condition	Indiquer résidence (ou pavillon ou chambre ou lit) ajoutée

Tableau 2: Cas d'utilisation Gérer les résidences

Titre	Consulter la liste des bénéficiaires
Résumé	Ce cas d'utilisation permet de vérifier les listes des bénéficiaires selon la classe
Acteurs	Tous les utilisateurs
Prés Condition	
Scénario nominal	• Saisir l'identifiant d'une classe et valider
Scénario alternatif	• Information incorrecte • Demande de renseigner à nouveau les informations
Post condition	Affiche la liste de la classe concernée

Tableau 3: Cas d'utilisation Consulter la liste des bénéficiaires

Titre	**Consulter les états de payement**
Résumé	Ce cas d'utilisation permet d'avoir un vue d'ensemble sur les payements et les non payements des bénéficiaires
Acteurs	Chef de résidence
Prés Condition	S'authentifier
Scénario nominal	• Clique sur payement effectif • Clique sur non payement
Scénario alternatif	
Post condition	Affiche les bénéficiaires qui ont déjà payé Affiche les bénéficiaires qui n'ont pas encore payé

Tableau 4: Cas d'utilisation les états de payement

Titre	**Saisir les listes cas sociaux**
Résumé	Cas d'utilisation permet d'introduire la liste des cas sociaux
Acteurs	Chef de service hébergement
Prés Condition	S'authentifier
Scénario nominal	• Demande formulaire • Renseigner les informations et valider
Scénario alternatif	• Information incorrecte • Demande de renseigner à nouveau les informations
Post condition	Indication liste ajoutée

Tableau 5: Cas d'utilisation Saisir les listes cas sociaux

3. Diagramme de cas d'utilisation :

Les diagrammes de cas d'utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs (actors), ils interagissent avec les cas d'utilisation (use cases). Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les cas d'utilisation (*use case*) sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation. Un acteur et un cas d'utilisation sont mis en relation par une association représentée par une ligne. L'activité du système a pour objectif de satisfaire les besoins de l'acteur. Il existe trois types de relation :

Les **inclusions (***include***),** ce type d'interaction, le premier cas d'utilisation inclut le second et son issue dépend souvent de la résolution du second.

Les **extensions** (*Extend*) représentent des prolongements logiques de certaines tâches sous certaines conditions. Autrement dit un cas d'utilisation A étend un cas d'utilisation B lorsque le cas d'utilisation A peut être appelé au cours de l'exécution du cas d'utilisation B.

La **généralisation** ou **spécialisation** : Le cas d'utilisation A est une généralisation de B, si B est un cas particulier de A c'est-à-dire lorsque A peut-être substitué par B pour un cas précis. Ces

Il est également possible d'appliquer à un acteur la relation de **généralisation**. Cela se fait notamment lorsqu'un acteur est un "sous-type" d'une autre catégorie d'acteurs. [**B6**]

Nous avons dans ce mémoire trois diagrammes de cas d'utilisation. Pour chacun, nous allons ressortir ses acteurs et ses fonctionnalités.

3.1.L'attribution des lits :

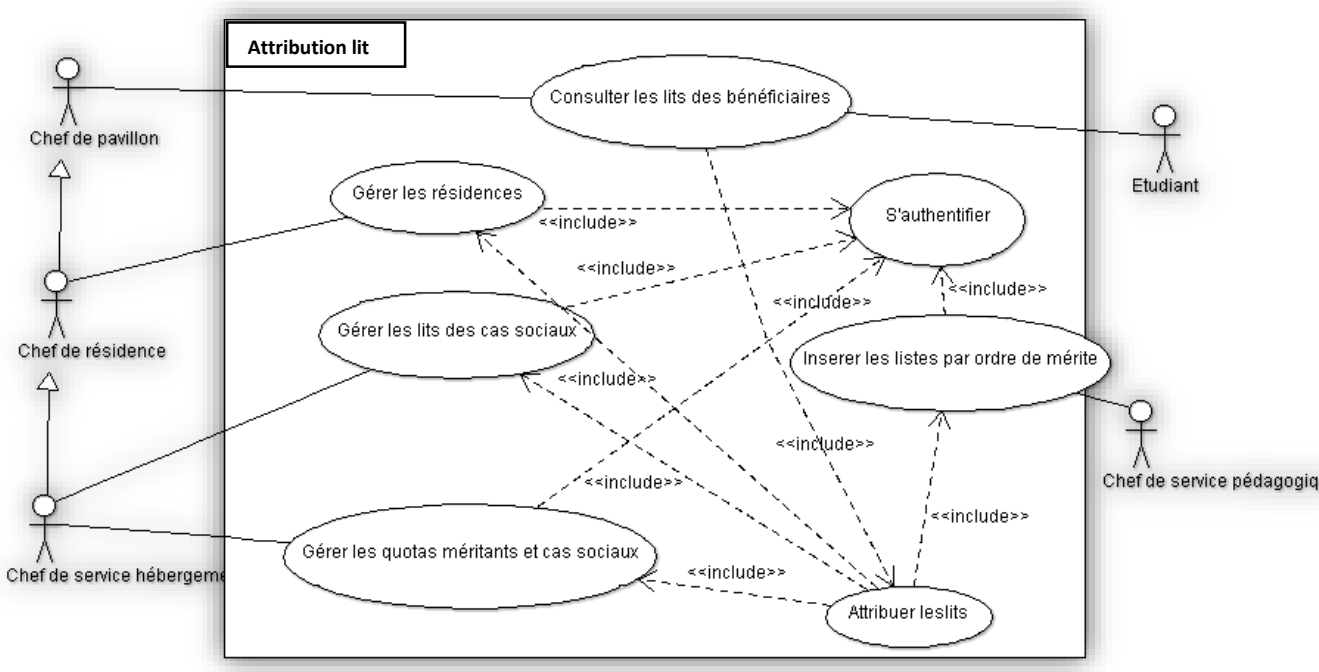


Figure 1: Attribution des lits

Comme nous l'allons présenter en dessus, dans ce diagrammes, les acteurs sont représentées par des bonhommes et sont à l'extrémité, les cas d'utilisation ou fonctionnalités du système par les losanges et les relations par les traits (pleins ou pointillés).

Ce diagramme de cas d'utilisation montre la manière dont l'attribution des lits est gérée ; l'interaction des acteurs et les fonctionnalités de la plateforme. Ici nous avons cinq acteurs et sept cas d'utilisation.

Les acteurs sont représentés à gauche, nous avons utilisé comme relation la **généralisation** entre quelques acteurs. Exemple : Le Chef de service hébergement hérite les possibilités du chef de résidence, ce dernier fait aussi de même pour le chef de pavillon.

Pour les cas d'utilisation, nous avons utilisé une seule relation « *inclusion* » entre eux. Par exemple : avant d'accéder à certaines fonctionnalités de la plateforme excepté attribution des lits et consulter la listes des bénéficiaires, il faut impérativement s'authentifier. Aussi avant que l'attribution des lits soit possible, il faut à la fois les lits, les quotas, les listes de passage et/ou les listes des cas sociaux. En plus pour que l'étudiant consulte sa liste des bénéficiaires, il faut que le système attribue les lits.

3.2.Le payement et suivi :

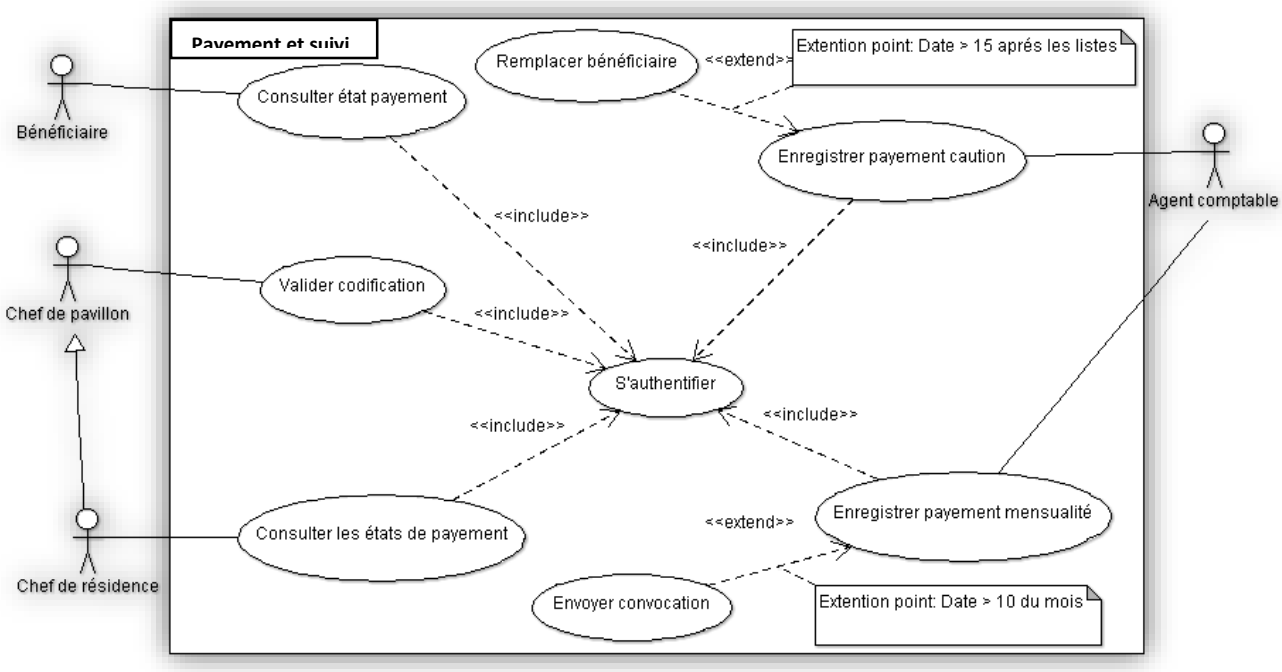


Figure 2: Payement et suivi

Ce diagramme de cas d'utilisation nous montre comment l'intérêt du payement est important dans le processus de la codification. Même formalisme que le diagramme précédent pour les acteurs et les cas d'utilisation. Dans ce cas de figure nous avons quatre acteurs et huit cas d'utilisation. Tous les acteurs doivent s'authentifier pour pouvoir accéder à une fonctionnalité. Aussi pour que le bénéficiaire consulter son état de payement, il va falloir que le chef de pavillon valide sa codification et cela sous-entendu que l'agent comptable a déjà enregistré son payement de caution. Dans un délai de 15 jours si le bénéficiaire ne paye pas cette caution, le système le remplace par le suivant sur la liste d'entente. Aussi l'agent comptable enregistre le payement des mensualités. Le système envoie automatiquement une convocation lorsque le bénéficiaire accuse un retard de payement de 10 jours. Le chef de résidence a la possibilité de consulter les états de payement des bénéficiaires. En plus, grâce à la **généralisation** que nous avons utilisée entre le chef de résidence et le chef de pavillon, la première cité à toutes les possibilités que ce dernier.

3.3.L'expression des besoins et suivi :

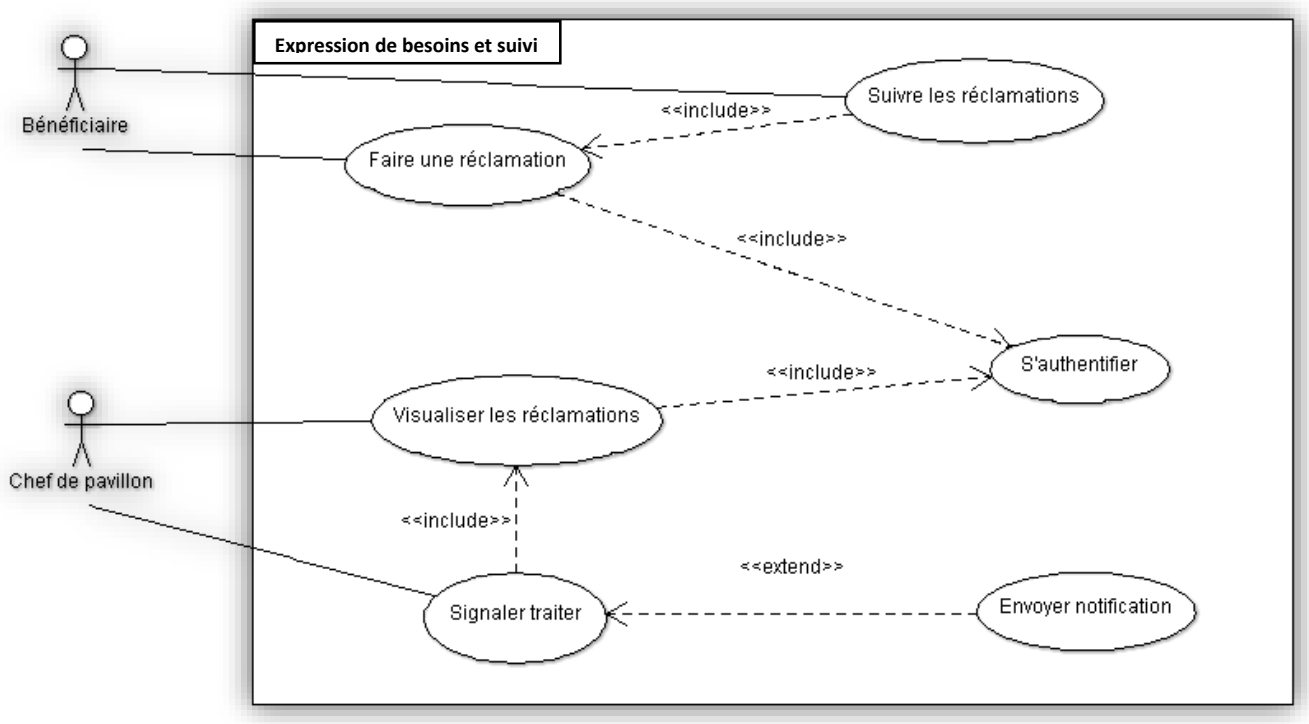


Figure 3 : Expression des besoins et suivi

Ce diagramme nous décrit l'expression des besoins des bénéficiaires et le suivi. Même règle que les diagrammes précédents pour la représentation. Ici nous avons deux acteurs et six cas d'utilisation. Tous les acteurs s'authentifient avant d'accéder à la fonctionnalité concernée. Le bénéficiaire se connecte et fait une réclamation. Le chef de pavillon peut se connecter temporairement pour visualiser les réclamations. Après visualisation, le chef de pavillon saisit l'ouvrier concerné puis il signale avoir traité le problème, automatiquement le système envoie une notification au bénéficiaire concerné. Aussi le bénéficiaire a la possibilité de suivre sa réclamation.

4. Diagramme de séquence :

Le diagramme de séquence permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation. Dans un souci de simplification, on représente l'acteur principal à gauche du diagramme, et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

La dimension verticale du diagramme représente le temps, permettant de visualiser l'enchaînement des actions dans le temps, et de spécifier la naissance et la mort d'objets. Les périodes d'activité des objets sont symbolisées par des rectangles, et ces objets dialoguent par le biais de messages. [**B6**]

Nous allons représenter quelques cas d'utilisation sous forme de diagramme de séquence.

4.1.S'authentifier :

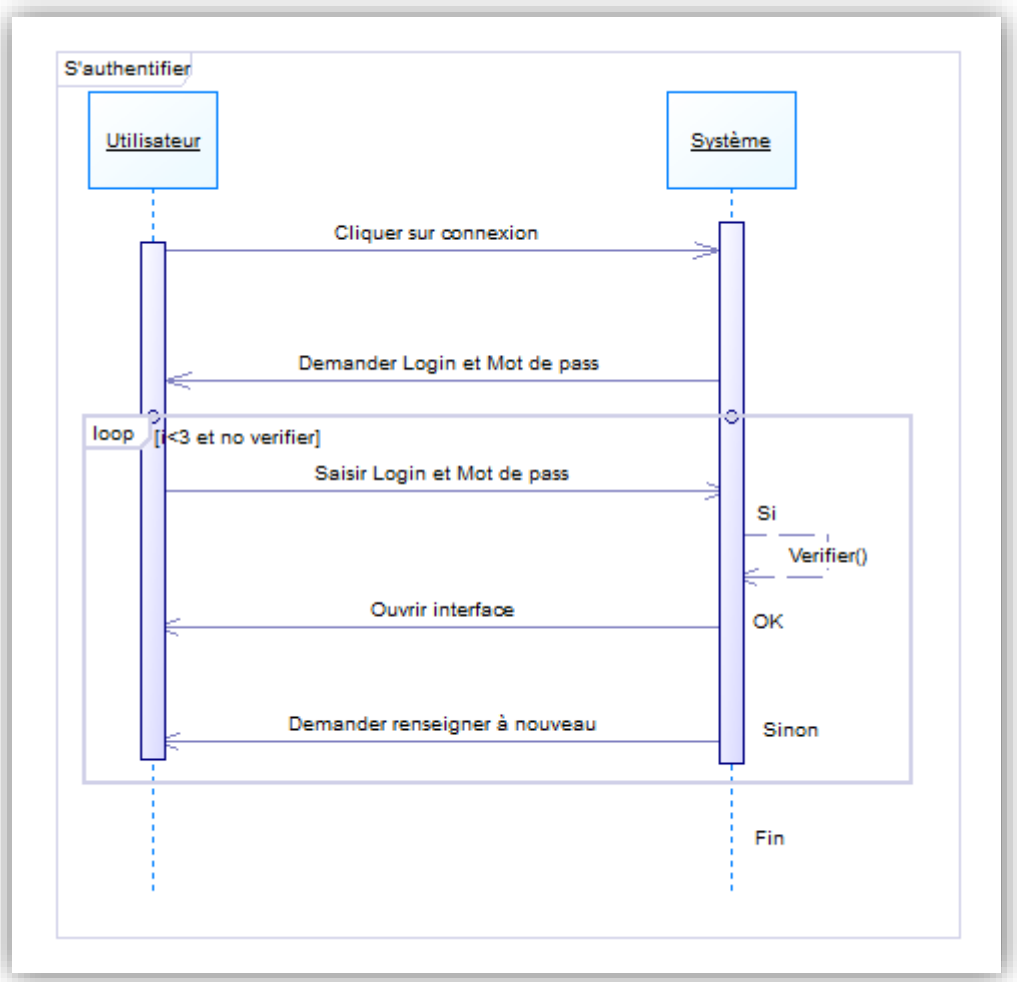


Figure 4 : S'authentifier

- ✓ L'utilisateur clique sur le bouton **Connexion**
- ✓ Le système lui répond en lui en demandant d'entrer son nom d'utilisateur et son mot de pass
- ✓ L'utilisation va renseigner tous les champs
- ✓ Le système vérifie les informations :
    - Si elles sont correctes, le système va ouvrir l'interface concerné
    - Sinon, le système va demander à l'utilisateur de saisir à nouveau les informations
    - Cette action va se répéter trois fois, après le système va obliger de fermer la session

4.2.Insérer les listes par ordre mérite :

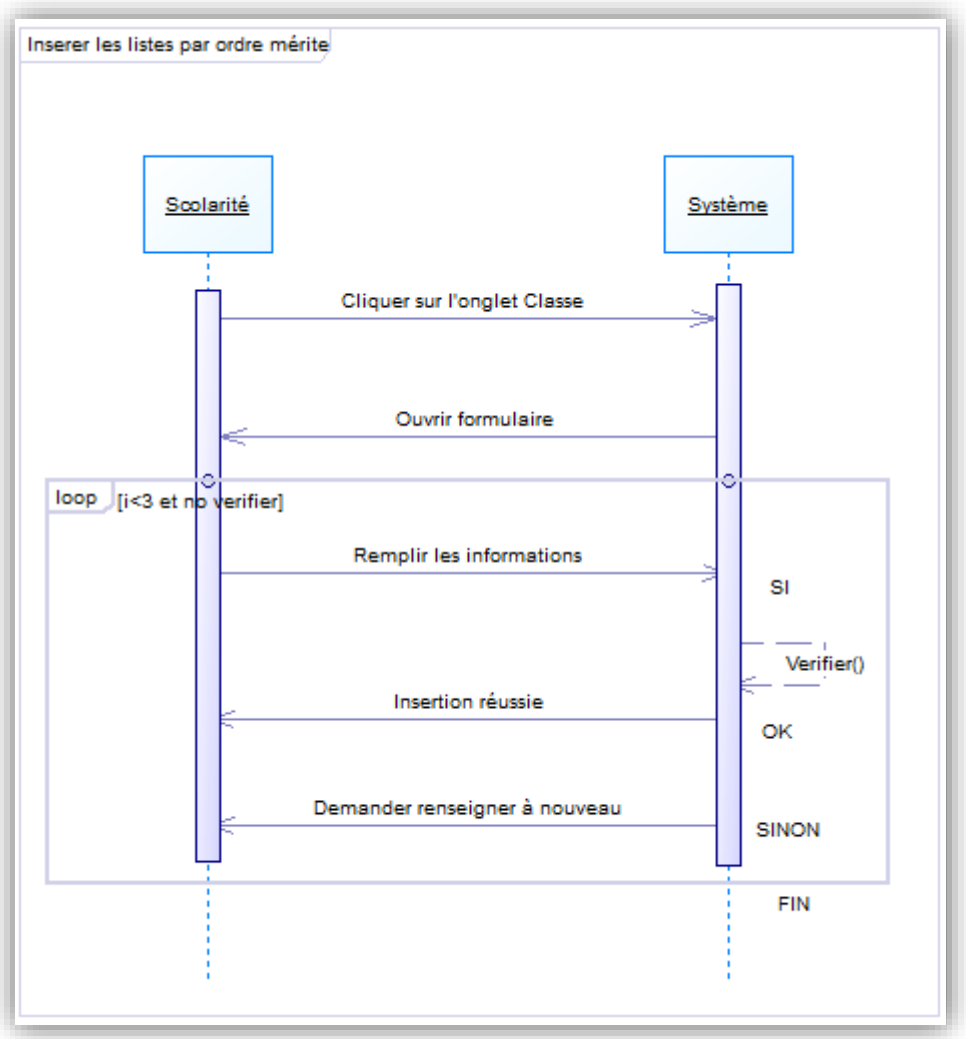


Figure 5: Insérer les listes par ordre de mérite

- ✓ La Chef de service pédagogique clique sur l'onglet Classe
- ✓ Le système lui répond en lui ouvrant un formulaire

- ✓ L'utilisation va renseigner tous les champs
- ✓ Le système vérifie les informations :
    - Si elles sont correctes, le système va confirmer la réussite de l'insertion
    - Sinon, le système va demander à l'utilisateur de saisir à nouveau les informations
    - Cette action va se répéter trois fois, après le système va obliger de fermer la session

4.3.Consulter les listes des bénéficiaires :

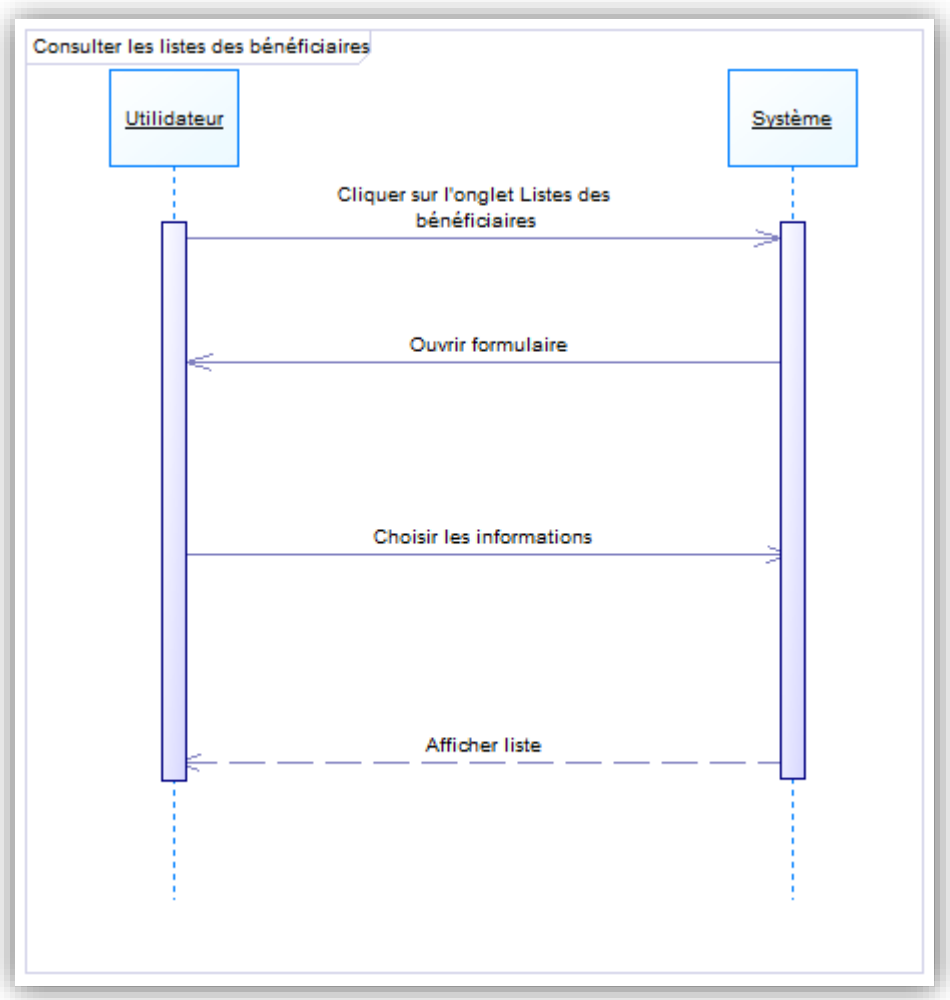


Figure 6: Consulter les listes des bénéficiaires

- ✓ L'utilisateur clique sur l'onglet Liste des Bénéficiaire
- ✓ Le système lui répond en lui ouvrant un formulaire
- ✓ L'utilisation va renseigner le champ de recherche
- ✓ Le système lui affiche la liste

4.4.Enregistrer payement:

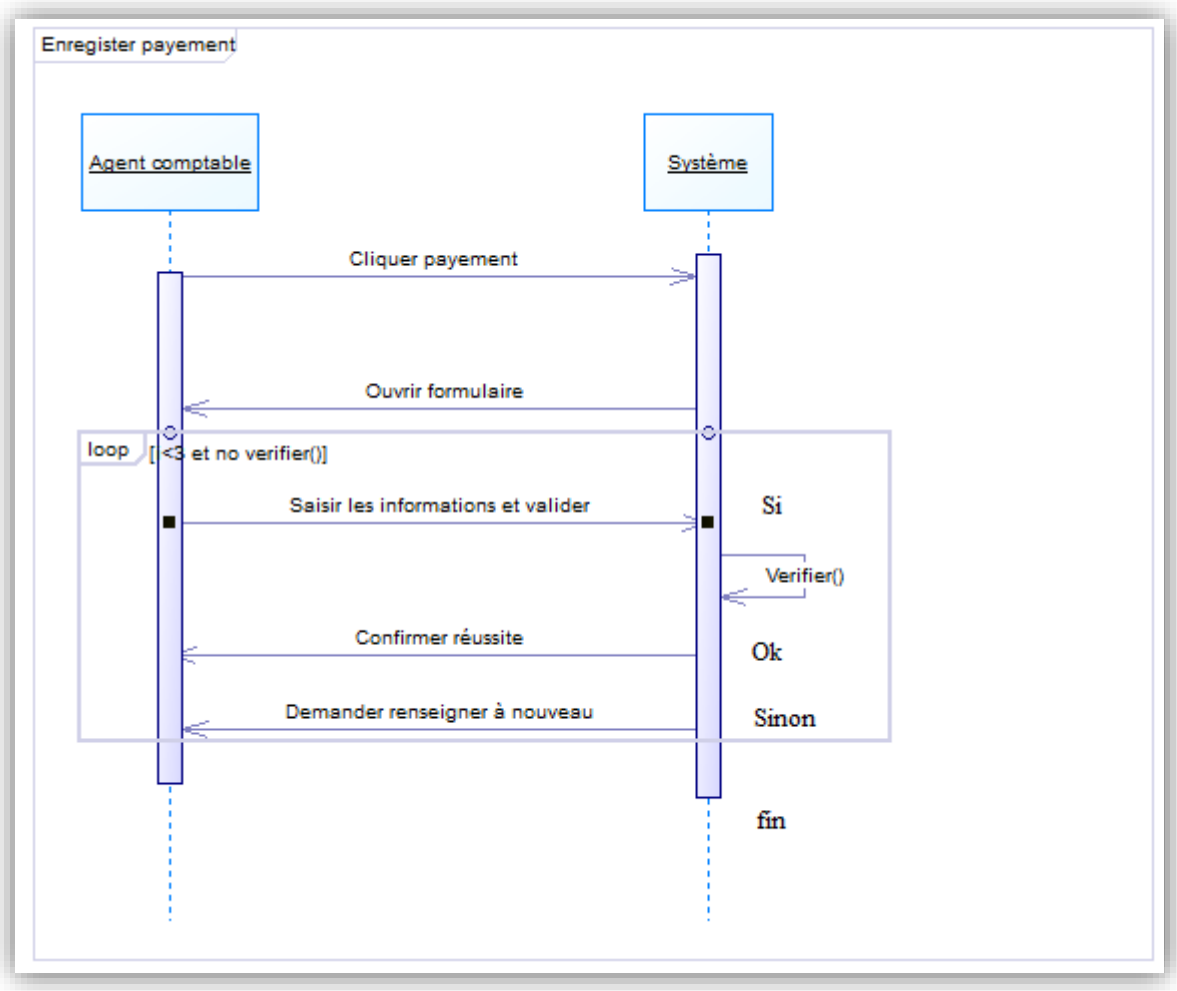


Figure 7: Enregistrer paiement

- ✓ L'Agent comptable clique sur l'onglet payement
- ✓ Le système lui répond en lui ouvrant un formulaire
- ✓ L'utilisation va renseigner tous les champs en spécifiant s'il s'agit d'une caution ou d'une mensualité
- ✓ Le système vérifie les informations :
    - Si elles sont correctes, le système va confirmer la réussite de l'insertion
    - Sinon, le système va demander à l'utilisateur de saisir à nouveau les informations
    - Cette action va se répéter trois fois, après le système va obliger de fermer la session

4.5.Valider une codification :

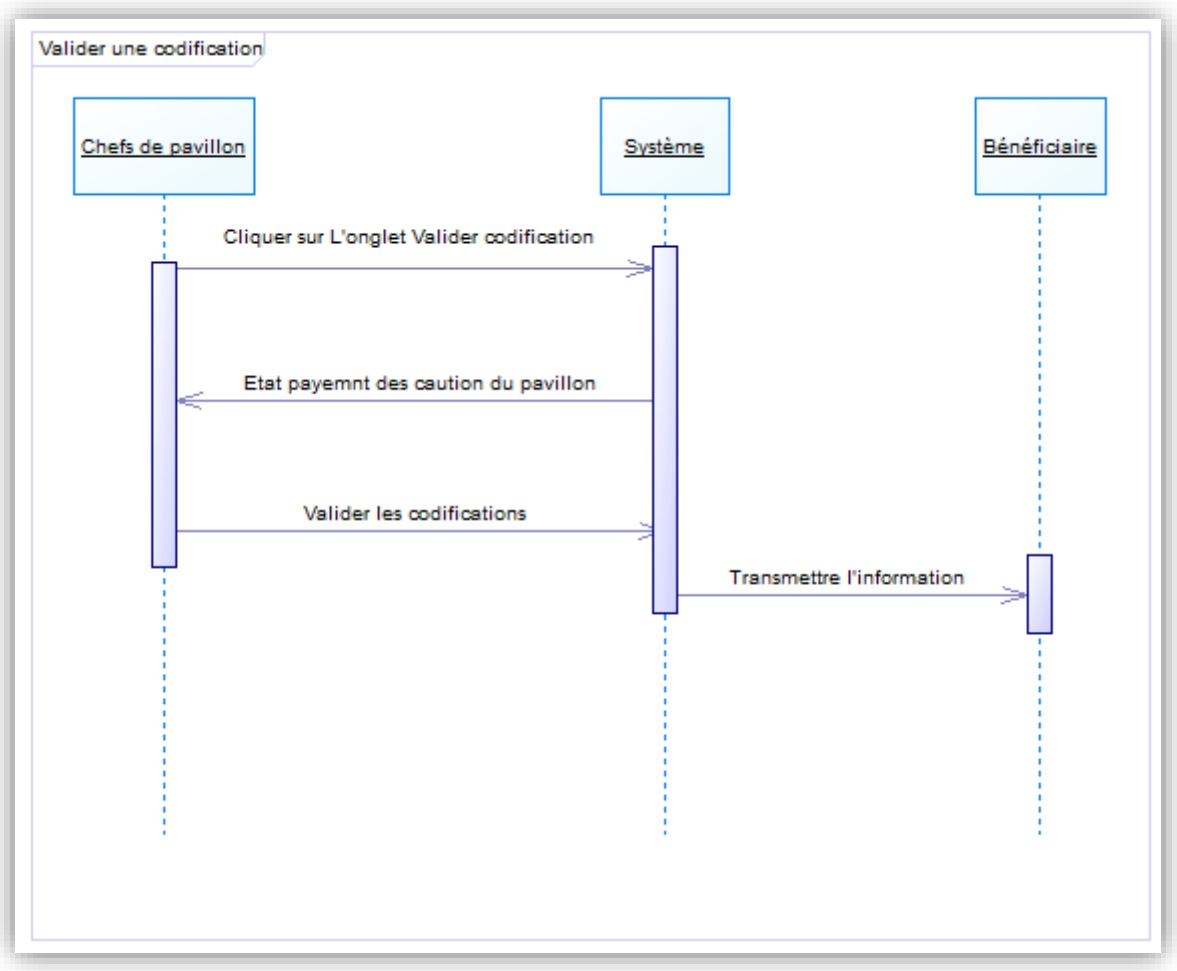


Figure 8: Valider une codification

- ✓ Le Chef de pavillon clique sur l'onglet Valider codification
- ✓ Le système lui répond en lui affichant l'état de payement du pavillon
- ✓ Le Chef de pavillon va valider les codifications des bénéficiaires qui ont déjà payé les cautions
- ✓ Le système transmet l'information aux bénéficiaires concernés : mot de pass par défaut

4.6.Faire une réclamation :

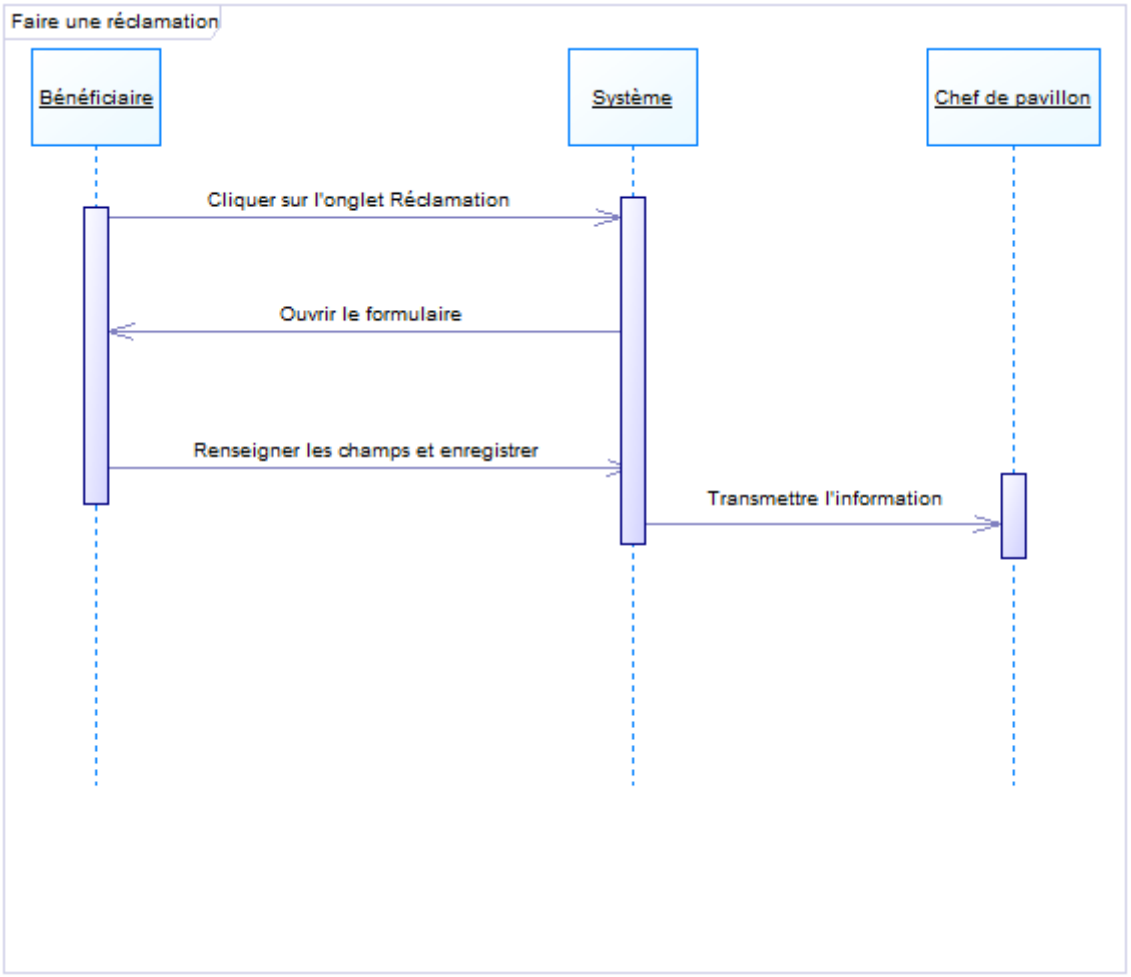


Figure 9: Faire une réclamation

- ✓ Le Bénéficiaire clique sur l'onglet Réclamation
- ✓ Le système lui répond en lui ouvrant un formulaire
- ✓ Le Bénéficiaire va renseigner les champs et enregistre
- ✓ Le système transmet l'information aux Chefs de pavillon concernés

4.7.Consulter les états de payement :

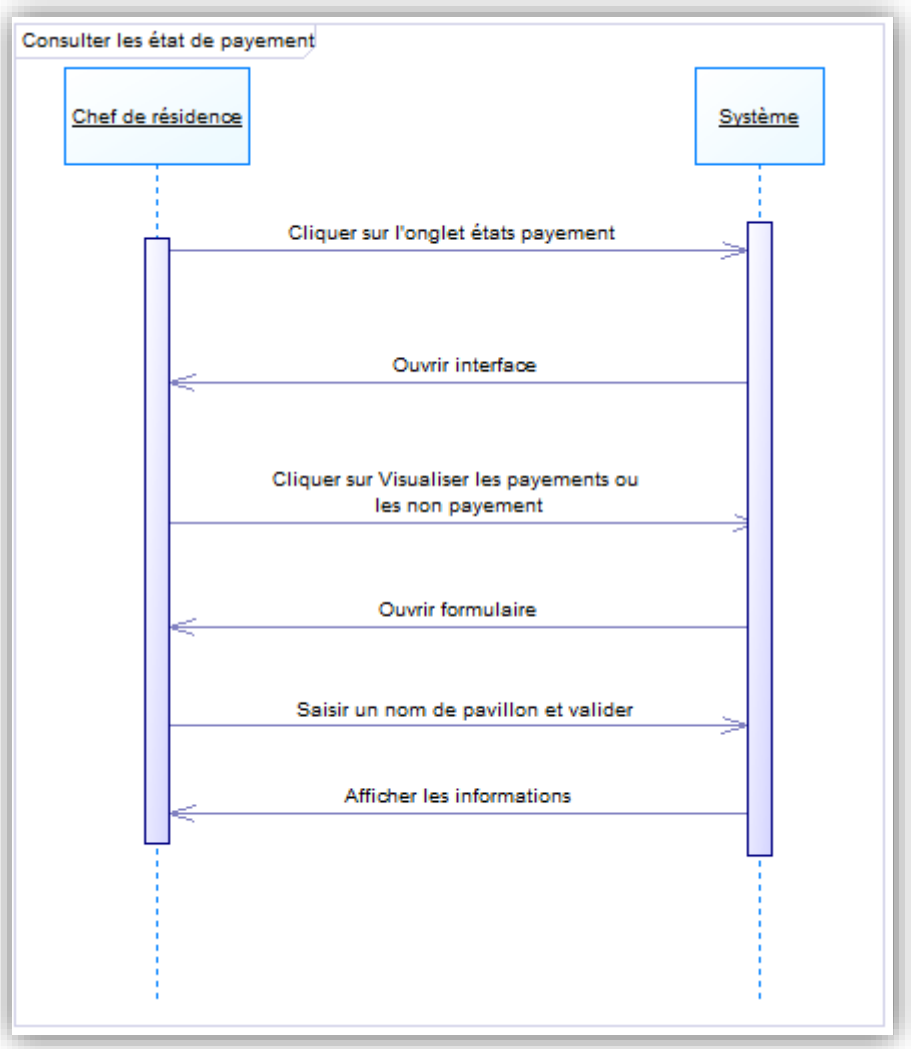


Figure 10: Consulter les états de payement

- ✓ Le chef de résidence clique sur l'onglet Etats payement
- ✓ Le système l'ouvre l'interface concerné
- ✓ Le chef résidence peut : visualiser les payements, les non payement
- ✓ Le système répond en lui ouvrant un formulaire
- ✓ Le chef de résidence saisir un nom d'un pavillon
- ✓ Le système affiche les informations concernant le pavillon en question.

5. Diagramme de classe :

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Le diagramme de classes montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

Les principaux éléments de cette vue statique sont les classes et leurs relations : association, composition, généralisation et plusieurs types de dépendances, telles que la réalisation et l'utilisation. Nous allons définir seulement les relations utilisées dans ce mémoire.

- L'**association** est une connexion physique ou conceptuelle entre classes donc entre objets. Un lien est une instance d'une association. Chaque association peut être identifiée par son nom. Une association entre classes représente les liens qui existent entre les instances de ces classes.
- L**'agrégation** est une association non symétrique, qui exprime une relation de subordination. Elle représente une relation de type "ensemble / élément". L'agrégation est une forme particulière d'association entre plusieurs classes.
- La **composition** est un cas particulier de l'agrégation dans laquelle la vie des composants est liée à celle des agrégats. Elle fait souvent référence à une contenance physique. La composition implique, en plus de l'agrégation, une coïncidence des durées de vie des composants: la destruction de l'agrégat (ou conteneur) implique automatiquement la destruction de tous les composants liés.
- La **généralisation** est la relation entre deux autres classes ou plus partageant un ensemble commun d'attributs et/ou d'opérations. Elle désigner la relation de classification entre un élément plus général et un élément plus spécifique.

Une classe est la description formelle d'un ensemble d'objets ayant une sémantique et des caractéristiques communes. 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. [**B6**]

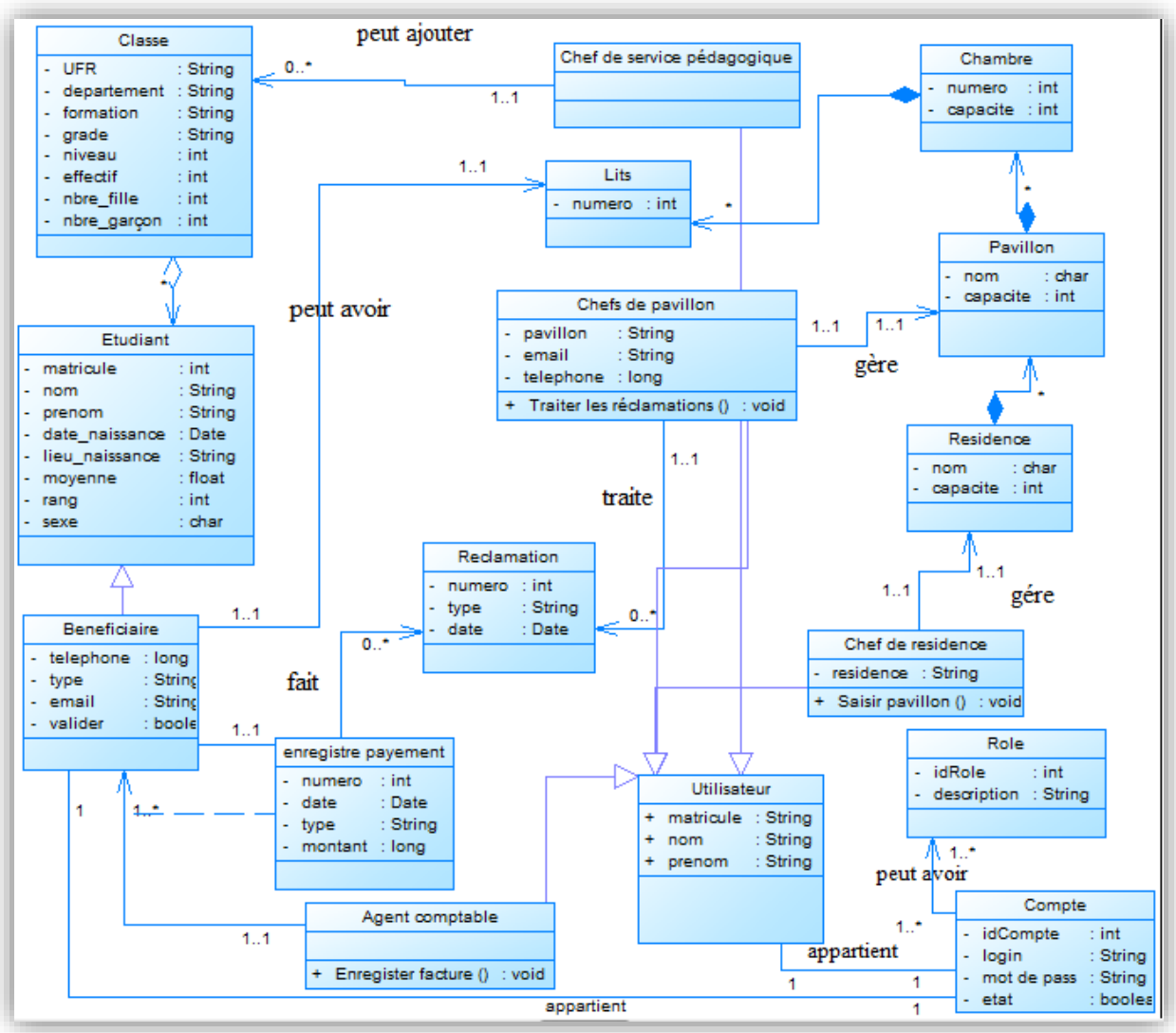


Figure 11: Diagramme de classe

Nous allons amener quelque explication de ce diagramme de classe :

✓ Relation entre *Chef de service pédagogique* et *Classe* est une **association** :

1.1 du côté de la classe *Chef de service pédagogique* traduit qu'une Classe ne peut être ajouté que par un agent de la Chef de service pédagogique

0..* du côté de la classe *Classe* signifie qu'un agent de la Chef de service pédagogique peut ajouter zéro à plusieurs Classe

✓ Relation entre *Classe* et *Etudiant* est une **agrégation** :

Cette relation signifie qu'une classe est composée d'une ou plusieurs étudiants.

✓ Relation entre *Etudiant* et *Bénéficiaire* est une **généralisation :**

Cette relation veut dire que la classe bénéficiaire en plus de ses propres attributs hérite ceux de la classe étudiant

- ✓ Relation entre *Bénéficiaire* et *Réclamation* est une **association** :

1..1 du côté de la classe *Bénéficiaire* traduit qu'une réclamation est fait par un et un seul bénéficiaire.

0..* du côté de la classe *Réclamation* traduit qu'un bénéficiaire peut ne pas concerner une réclamation comme, il peut faire plusieurs réclamations.

- ✓ Relation entre Utilisateur et (Chef de service pédagogique, Agent comptable, Chef de résidence et Chef de pavillon) est une **généralisation** :

Cette relation signifie que les classes Agent comptable, Chef de résidence et Chef de pavillon héritent tous les attributs de Utilisateur.

- ✓ Relation entre *Résidence* et *Pavillon* est une **composition :**
- ✓ Relation entre *Pavillon* et *Chambre* est une **composition :**
- ✓ Relation entre *Chambre* et *Lit* est une **composition :**

Ces relations traduit que si on supprime l'agrégat, les autres classes disparaissent automatiquement. Par exemple : si résidence n'existe plus, il n'y aura plus de Pavillon ni chambre ni lit.

- ✓ Relation entre *Bénéficiaire* et *Lit* est une **association** :

1..1 du côté de la classe *Lit* traduit qu'un bénéficiaire peut avoir un et un seul lit

1..1 du côté de la classe *Bénéficiaire* signifie qu'un lit concerne un et un seul bénéficiaire

- ✓ Relation entre *Bénéficiaire* et *Agent comptable* est une **association :**

1..1 du côté de la classe Agent comptable traduit qu'un payement concerne un et un seul bénéficiaire

1..* du côté de la classe Bénéficiaire traduit qu'un agent comptable peut enregistrer un à plusieurs payement

- ✓ Relation entre *Chef de résidence* et *Résidence* est une **association :**

1..1 du côté de la classe Résidence traduit qu'un chef de résidence ne peut gérer qu'un et un seul résidence.

1..* du côté de la classe Chef de résidence traduit qu'une résidence est géré par un et un seul chef de résidence

Relation entre *Chef de pavillon* et *Pavillon* est une **association :**

1..1 du côté de la classe Pavillon signifie qu'un chef de pavillon gère un et un seul pavillon

1..1 du côté de la classe Chef de pavillon signifie qu'un pavillon concerne un et un seul chef de pavillon

Relation entre *Compte* et *Rôle* est une **association :**

1..* du côté de la classe Compte signifie qu'un rôle peut être possédé par un ou plusieurs comptes

1..* du côté de la classe Rôle veut dire qu'un compte peut posséder un à plusieurs rôles

Relation entre *Compte* et *(Bénéficiaire et Utilisateur)* est une **association :**

1 du côté de la classe Compte traduit qu'un utilisateur peut avoir un et un seul compte pour se connecte.

1 du côté des classes Bénéficiaire et Utilisateur signifie qu'un compte appartient à un seul utilisateur.

6. Modèle logique de données :

Le modèle logique des données consiste à décrire la structure de données utilisée sans faire référence à un langage de programmation. Il s'agit donc de préciser le type de données utilisées lors des traitements. Ainsi le modèle logique dépend du type de base de données utilisé.

Après une présentation du diagramme de classe nous allons passer au modèle relationnel, tout en précisant les règles de passage.

- **Transformation des Classes d'objets**
  - la classe se transforme en une table (relation)
  - les attributs de la classe deviennent des attributs de la table
  - choisir un attribut (ou groupe d'attribut) de la classe pouvant jouer le rôle de clé primaire; si aucun attribut ne convient, il faut en ajouter un à la table.
- **Transformation des associations**
  - Cas Association binaire de type plusieurs [(1..*) ou (0..*)] à un [-(1..1) ou (0..1)]

- Engendre la migration de la clé primaire de la table mère à la table fille en clé étrangère.
- Cas d'une liaison n-aire plusieurs (0..*) à plusieurs (0..* ou 1..*)
    - L'association ou la classe-association devient une relation.
    - La clé primaire est composée des clés primaires des relations obtenues.
    - Les éventuels attributs de la classe-association deviennent des attributs de la nouvelle relation
- Cas de l'héritage
    - Transformer chaque sous classe en une relation. La clé primaire de la superclasse devient clé primaire de chaque sous classe. [**B7**]

En appliquant ces règles de transformation d'un diagramme de classe vers un modèle relationnel, nous allons avoir le modèle logique de données.

Utilisateur (matricule, nom, prénom, #idCompte)

Chef de résidence (matricule, #résidence)

Chef de pavillon (matricule, #pavillon)

Agent comptable (matricule)

Résidence (nom, capacité)

Pavillon (nom, capacité, #nom_résidence)

Chambre (numéro, capacité, # pavillon)

Lit (numéro, #chambre)

Réclamation (numéro, type, montant, date, #chambre)

Chef de service pédagogique (matricule, nom, prénom, UFR)

Classe (idClasse, UFR, département, formation, grade, niveau, effectif, nombre_fille, nombre_garçon, matricule_Sco)

Etudiant (matricule, nom, prénom, date_naissance, lieu_naissance, moyenne, range, sexe, #idClasse)

Bénéficiaire (matricule, téléphone, type, valider, #lit)

Enregistre payement (numéro, #matricule_B, date, type, montant)

Compte (idCompte, login, mot de pass, état, #utilisateur)

Rôle (idRole, description, #compte)

Conclusion

Ce chapitre est le cœur du travail car il nous a permis de ressortir toutes les fonctionnalités qui devront être développer dans la plateforme. C'est aussi le moment de montre tous les acteurs qui interviennent dans cette plateforme avec leur droit d'accès.

Chapitre 3 : Implémentation du modèle

1. Architecture

L'architecture peut être définie comme étant l'art de concevoir, de combiner et de disposer par des techniques appropriées. **[B8]** Il existe plusieurs types d'architectures dont l'architecture trois tiers appelé aussi architecture à trois niveaux ou architecture à trois couches. L'architecture logique du système est divisée en trois niveaux ou couches : couche présentation, couche métier et couche accès aux données.

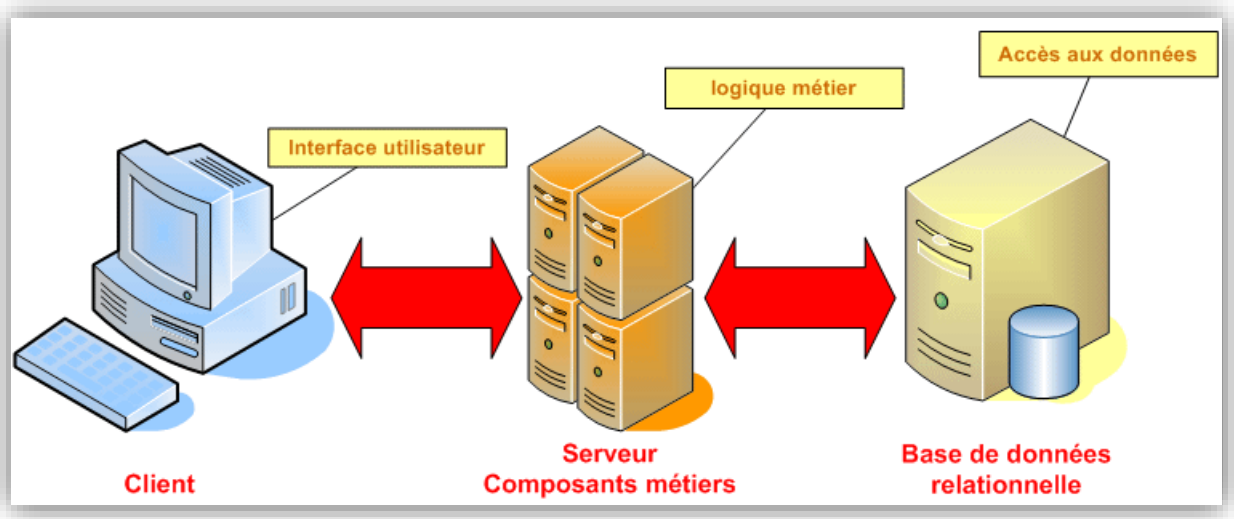


Figure 12: Architecture trois tiers ***[B9]***

1.1. Couche présentation :

Elle correspond à la partie de l'application visible et interactive avec les utilisateurs. On parle d'interface homme machine. En informatique, elle peut être réalisée par une interface graphique.

Dans cette partie nous avons utilisé le pattern MVC (Model View Controller) qui est une méthode de conception qui divise l'Interface Homme Machine (IHM) en un modèle (modèle de données), des vues (présentation, interface utilisateur) et un contrôleur (logique de contrôle).

Le contrôleur prend en charge toutes les demandes des clients afin de mettre à jour leurs vues ou leurs modèle. Le modèle décrit les données manipulées par l'application. La vue représente l'Interface Homme-Machine. C'est avec cette partie de la couche présentation que l'utilisateur interagit.

La couche présentation relaie les requêtes de l'utilisateur à destination de la couche métier, et en retour lui présente les informations renvoyées par les traitements de cette couche. Il s'agit donc ici d'un assemblage de services métiers et applicatifs offerts par la couche inférieure.

Nous avons utilisé certaines technologies pour améliorer l'ergonomie de la partie vue de l'application telle que Bootstrap et JQuery.

- ✓ Bootstrap nous a été d'une grande utilité parce qu'il nous a permis d'avoir de jolies pages en écrivant moins de code CSS et JS. Tous les composants sont déjà mis en forme, il suffit tout simplement d'inclure le nom de la classe qu'on veut au niveau des pages HTML. C'est un outil gratuit et open source, de plus il est compatible avec la plupart des navigateurs.
- ✓ JQuery est une bibliothèque JavaScript rapide, petite et riche en fonctionnalités. Il facilite la manipulation, la gestion des évènements et l'animation dans la partie vue.

1.2. Couche métier :

Elle correspond à la partie fonctionnelle de l'application. Elle implémente la logique et décrit les opérations que l'application exécute sur les données en fonction des requêtes des utilisateurs.

Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche. Elle offre des services applicatifs et métier à la couche présentation. Pour fournir ces services, elle s'appuie, le cas échéant, sur les données du système, accessibles au travers des services de la couche inférieure. En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés.

1.3. Couche accès aux données :

Elle consiste en la partie gérant l'accès aux données du système. Ces données peuvent être propres au système, ou gérées par un autre système. La couche métier ne gère pas cet aspect, qui est transparents pour elle. Elle accède aux données de manière uniforme.

2. Les langages utilisés :

Un langage de programme est défini comme étant un langage permettant de formuler des algorithmes et de produire des programmes informatiques. Il existe plusieurs langages de programme, mais ce mémoire, nous avons utilisé la plate-forme JEE (Java Edition Entreprise) qui est un ensemble de technologies créées par SUN MicroSystems. C'est une extension de la plate-forme standard JSE (Java Standard Edition). Autrement dit, la plate-forme Java EE est

construite sur le langage Java et la plate-forme Java SE. Cependant, elle y ajoute un grand nombre de bibliothèques remplissant un certain nombre de fonctionnalités que la plate-forme standard ne remplit pas d'origine. L'objectif majeur de Java EE est de faciliter le développement d'applications web robustes et distribuées, déployées et exécutées sur un serveur d'applications.

Après avoir fait une recherche avancée sur cette plate-forme, nous avons découvert quelques Framework tels que Struts et Spring MVC que nous avons utilisés pour l'implémentation de nos modèles

- Struts :

Struts est un Framework pour applications web développé par le projet Jakarta de la fondation Apache. Il met en œuvre le modèle MVC 2 basé sur une seule servlet faisant office de contrôleur et des JSP pour l'IHM. L'application de ce modèle permet une séparation en trois parties distinctes de l'interface, des traitements et des données de l'application.

Struts se concentre sur la vue et le contrôleur. L'implémentation du modèle est laissée libre aux développeurs : ils ont le choix d'utiliser des JavaBeans, un outil de mapping objet/relationnel, des EJB ou toute autre solution.

Pour le contrôleur, Struts propose une unique servlet par application qui lit la configuration de l'application dans un fichier au format XML. Cette servlet de type ActionServlet reçoit toutes les requêtes de l'utilisateur concernant l'application. En fonction du paramétrage, elle instancie un objet de type Action qui contient les traitements et renvoie une valeur particulière à la servlet. Celle-ci permet de déterminer la JSP qui affichera le résultat des traitements à l'utilisateur.

- Spring :

Spring est un framework libre pour construire et définir l'infrastructure d'une application java dont il facile le développement et le teste. Il existe Spring MVC et Spring Boot qui est une continuité de Spring MVC**. [B10]**

2.1. Framework Spring MVC :

Spring est un Framework open source qui offre la possibilité de créer des applications Java EE plus robustes dans un intervalle de temps réduit. Il est actuellement l'un des frameworks Java EE le plus populaire. **[B10]** L'injection de dépendance (Dependency Injection : DI) ou l'inversion de contrôle (Inversion of Control : IOC) et la programmation orientée (aspect, Aspect Oriented Programming : AOP) sont le cœur de ce framework.

L'injection de dépendance est définie comme étant « *un mécanisme qui permet d'implémenter le principe de l'inversion de contrôle. Il consiste à créer dynamiquement*

(injecter) les dépendances entre les différentes classes en s'appuyant sur une description (fichier de configuration ou métadonnées). Ainsi les dépendances entre composants logiciels ne sont plus exprimées dans le code de manière statique mais déterminées dynamiquement à l'exécution» **[B11]**

La programmation orientée aspect est définie comme étant *« un paradigme de programmation qui permet de traiter séparément les préoccupations transversales, qui relèvent souvent de la technique, des préoccupations métier, qui constituent le cœur d'une application* » **[B12]**

Le Framework Spring contient un ensemble de modules qui traversent l'ensemble des trois couches de l'architecture trois tiers. Ces modules sont répertoriés sur la figure 13 suivant:

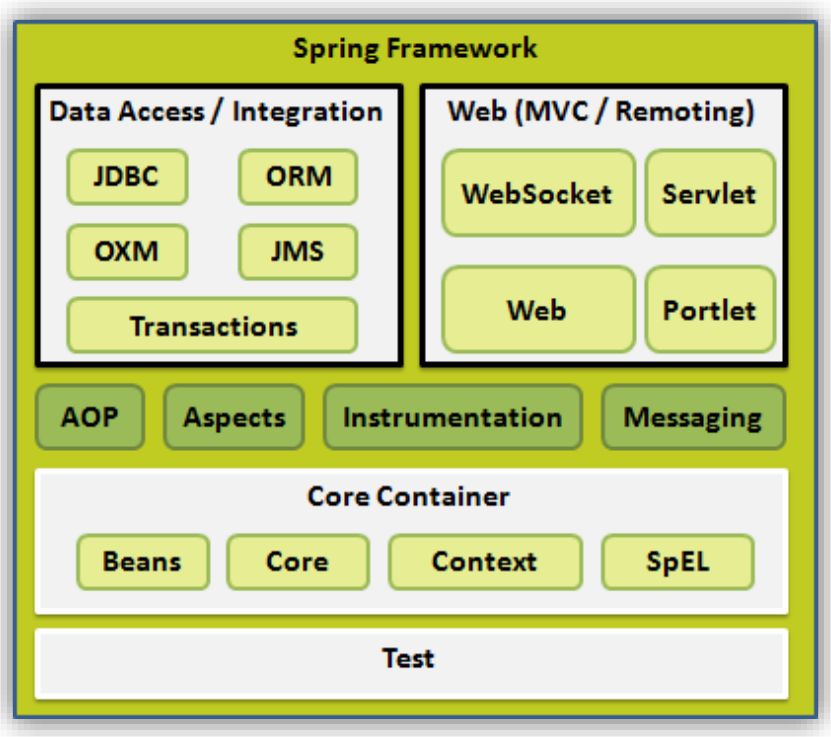


Figure 13: Les différents modules de Spring ***[B13]***

- Core :

Il définit le noyau de Spring et permet de faire l'inversion de contrôle ou l'injection de dépendance.

- Beans :

Ce module est une implémentation du design pattern Fabric, il permet de faire l'instanciation ou la création des objets. Il permet d'avoir un couplage faible entre les différents objets.

- Context :

Ce module définie un contexte qui permet de retrouver les objets de l'application.

- Expression Language

C'est un langage de requête qui peut être utilisé pour rechercher des objets.

- Test :

Ce module offre la possibilité de faire des tests unitaires et des tests d'intégration. C'est un module très intéressant pour faire de l'intégration continue. Il existe différents niveaux de test: Test unitaire, Test d'intégration, Test de charge, Test fonctionnel, Test sécurité etc. Le test qui nous intéresse le plus ici est le Test unitaire. C'est une procédure permettant de vérifier le bon fonctionnement d'une partie précise d'un programme. Les tests unitaires sont utilisés pour garantir la non régression du code, détecter plus facilement les bugs, voir l'avancement d'un projet, etc.

- AOP (Aspect Oriented Programming) :

Spring intègre le module AOP qui permet ici de faire de la programmation orientée aspect. Le but de ce module est de séparer le code métier du code technique.

- Aspects :

Le module Aspects s'intègre avec AspectJ qui est encore une fois un framework de programmation orientée aspect puissant et mature.

- JDBC :

Ce module permet d'établir une connexion avec une base de données.

- ORM (Object Relational Mapping) :

Ce module permet de faire une intégration de spring avec les framework de mappage objet relationnel tel que Hibernate.

- OXM (Object XML Mapping)

Ce module permet de faire la correspondance de tout ce qui est XML et objet au même titre que le module ORM.

- JMS (Java Message Service)

Ce module fournit un support pour exploiter l'API JMS qui permet ici d'envoyer ou de recevoir des messages asynchrone entre applications.

- Transactions

Ce module s'occupe de la gestion des transactions. Avec Spring lorsqu'une méthode est transactionnelle nous ajoutons l'annotation **@Transactional** avant la méthode.

- Web

Ce module permet de faire la gestion des vues c'est-à-dire de faire des applications web J2EE basées sur le modèle MVC.

- Struts

Ce module permet de faire l'intégration avec le framework Struts qui est un framework pour créer des applications java web.

2.2. Langage SQL :

Le SQL est un langage permettant de communiquer avec une base de données. Ce langage informatique est notamment très utilisé par les développeurs web pour attaquer les données d'une application web. Il est le langage standard des Systèmes de Gestion de Bases de Données (SGBD) relationnelles (SGBDR). C'est à la fois:

- ✓ Un langage de manipulation des données (LMD; ordres SELECT, UPDATE, INSERT, DELETE),
- ✓ Un langage de définition des données (LDD ; ordres CREATE, ALTER, DROP),
- ✓ Un langage de contrôle de l'accès aux données (LCD ; ordres GRANT, REVOKE).

Le langage SQL est utilisé par les principaux SGBDR: SQL Serveur, Oracle, PostgreSQL, MySQL. Dans ce mémoire nous l'avons utilisé avec MySQL.

3. La sécurité :

Spring framework offre aussi un module de sécurité. Spring Security est un module incontournable d'une application développée en Spring. Il apporte tout le nécessaire pour sécuriser une application et a l'avantage d'être personnalisable. La notion de sécurité informatique n'est pas une mince affaire et sa mise en place est parfois longue et demande d'être constamment adaptée au niveau réseau, serveurs, application. Spring Security n'intervient que sur le domaine applicatif.

Il est utilisé pendant l'implémentation du module gestion des utilisateurs et des rôles de l'application. La connexion, la déconnexion, l'affichage des vues et l'exécution de certaines méthodes du contrôleur sont gérés par ce module. Pour ce faire, il suffit de préciser le rôle de l'utilisateur qui a l'habilité de voir une vue ou d'exécuter une méthode. Il intervient au niveau de la couche présentation de l'architecture trois tiers de notre application.

Le framework spring implémente l'architecture suivant :

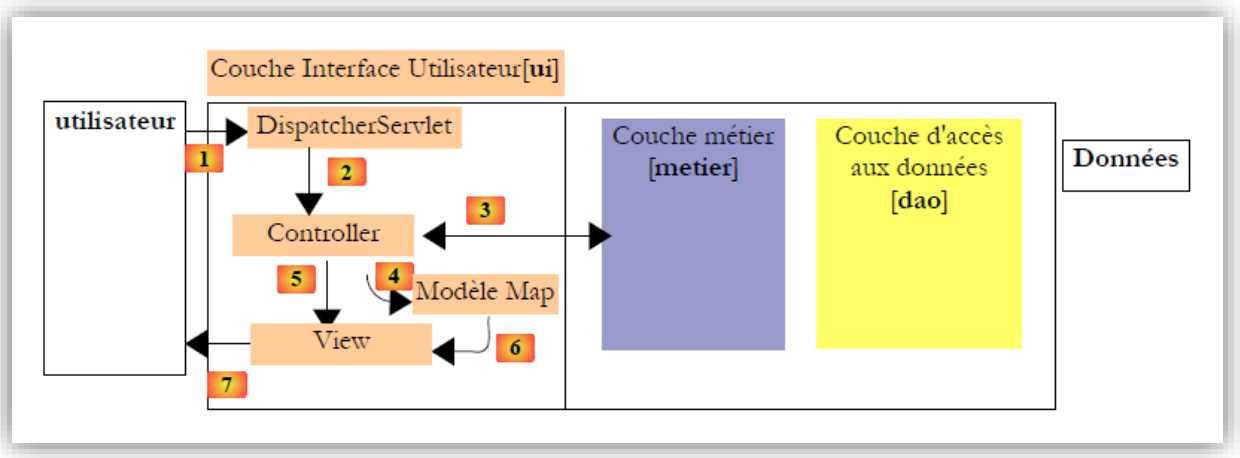


Figure 14: Architecture Spring ***[B14]***

Le framework Spring traverse toutes les couches de l'architecture trois tiers. Au niveau de la couche présentation, il implémente le patron architectural MVC. Avec Spring framework nous avons deux contrôleurs : un contrôleur principal (la classe **org.springframework.web.servlet.DispatcherServlet**) et un contrôleur secondaire.

Une fois que la requête du client arrivée au niveau du serveur, le contrôleur principal fait exécuter l'action demandée par le contrôleur secondaire. Cette dernière fait le traitement nécessaire et génère une réponse de type **java.util.Map.** Pour ce faire il peut faire appel à la couche métier pour la récupération de certaines informations. Le contrôleur choisit une vue et construit un modèle nécessaire à celle-ci. Le contrôleur principal demande alors à la vue choisie de s'afficher.

4. Le Système de Gestion de Base de Données utilisé :

Un système de gestion de Base de Données est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données.

Parmi les SGBD les plus connus il est possible de citer : MySQL, PostgreSQL, MS Access, Oracle, SQL Server.

- ✓ PostgreSQL : solution libre et gratuite, moins connue du grand public mais proposant des fonctionnalités inexistantes dans MySQL.
- ✓ Oracle : solution propriétaire et payante, massivement utilisée par les grandes entreprises. C'est un SGBD très puissant, robuste et est l'un des plus complets, mais l'un des plus chers également;

- ✓ SQL Server : solution propriétaire et payante, c'est un SGBD très puissant, robuste avec une interface ergonomique et facile à utiliser, sa capacité de stockage des données est aussi importante mais un grand problème avec SQL Server est dédié aux entreprises utilisant le Système d'exploitation Windows seulement ;
- ✓ MySQL : c'est le plus populaire au monde avec sa vitesse, sa fiabilité et sa facilité d'utilisation, son point fort est d'être gratuit et peux être utilisé même par les débutants au domaine**. [B15]**

Après étude et analyse, nous avons choisi d'utiliser le SGBS MySQL pour notre application. En effet, comme pour tout choix de technologie, la décision peut être influencée par plusieurs contraintes : coût, performances, support, etc. MySQL nous est familier et peut répondre parfaitement à nos attentes.

5. Le serveur :

Pour faire fonctionner une application web Java EE, nous avons besoin de mettre en place un serveur d'applications. Il en existe plusieurs dans le marché mais nous avons choisi d'utiliser Tomcat, car c'est un serveur léger, gratuit, libre, multiplateforme et assez complet. **[B16]**

Le cœur d'un serveur d'applications Java est le conteneur de servlets, puisque les servlets sont les éléments essentiels d'une application web écrite en Java (elles reçoivent les requêtes et renvoient les réponses). Le conteneur de servlets gère des servlets (sait où se trouvent physiquement les classes Java, pour quelles URL les appeler...), et les exécute lorsqu'elles sont demandées.

6. Les outils logiciels :

Pour la réalisation de ce mémoire, nous avons utilisé des logiciels et des matériels :

- ➢ *Eclipse* : Eclipse est une plateforme de développement ouverte composée de frameworks extensibles, outils et runtimes pour construire, déployer, et gérer le cycle de vie logiciel.
- ➢ ***PowerAMC*** : **PowerAMC** est un logiciel de modélisation. Il permet de modéliser les traitements informatiques et leurs bases de données associées.
- ➢ ***ArgoUML :*** ArgoUML est un logiciel de conception UML. Cet outil graphique utilise la plateforme Java et dispose des standards XMI, SVG et PGML. Étant sous licences BSD et Open Source, ArgoUML dispose d'extensions permettant à beaucoup de langages d'être reconnus (Java C++, C#, PHP, SQL), ainsi que nombreuses de langues, dont le Français. Ce logiciel est entièrement gratuit. ArgoUML propose aux développeurs un outil de représentation UML, leader de la scène open source.

Conclusion

Ce chapitre nous a permis d'aborder l'architecture trois tiers utilisé pour la plateforme, de définir le langage de programme choisi et les mesures de sécurité qui vont avec. Aussi nous avons étudié quelques Systèmes de Gestion de Base de Données qui existent afin d'en choisir MySQL pour notre travail et en fin ce chapitre était pour nous l'occasion de décliner le serveur Tomcat et les outils de modélisations.

Chapitre 4 : Présentation du système informatique obtenu

Prise en main du système

Notre travail est arrivé à son terme, nous avons passé de la modélisation à l'implémentation. Ainsi, dans ce chapitre, nous présentons l'application développée à travers ses principales fenêtres. En effet, la démarche méthodologie (2TUP) utilisée prend en considération l'avis des utilisateurs tout au long du travail. Cette phase parait primordiale car il donne une compréhension de la plateforme en expliquant en détail certaines fonctionnalités et les mesures de sécurités. Rappelons les trois composants de notre système d'information :

- Attribution des lits :
- Payement des droits de logement et leur suivi :
- Entretien des chambres et leur suivi :

1. Menus de l'application

Nous allons citer les menus principaux de l'application que nous détaillerons un par un.

- Accueil

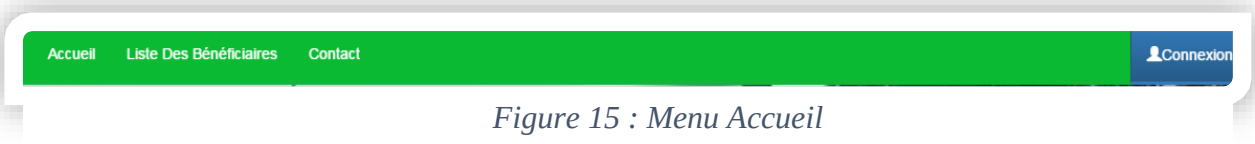


Figure 15 : Menu Accueil

- Administrateur

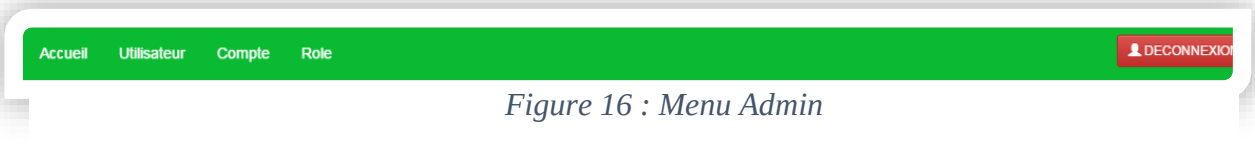


Figure 16 : Menu Admin

- Chef de service pédagogique

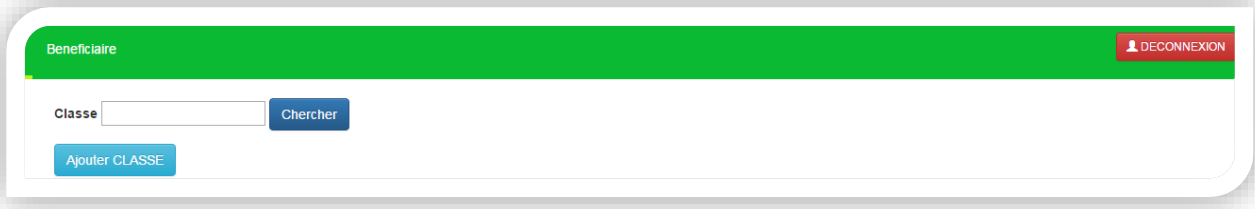


Figure 17 : Menu UFR

- Espace Chef de résidence

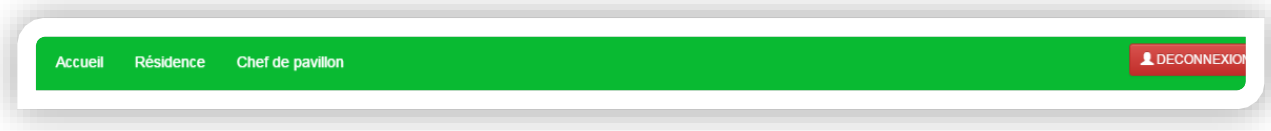


Figure 18 : Menu Chef de résidence

- Espace chef de pavillon

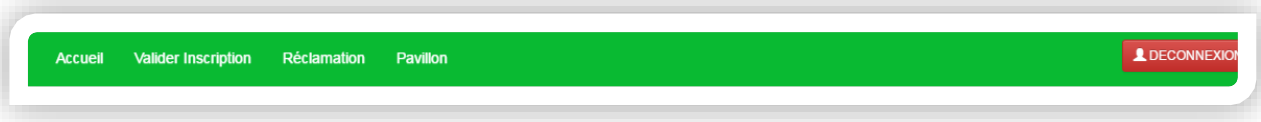


Figure 19 : Menu Chef de pavillon

- Espace agent comptable

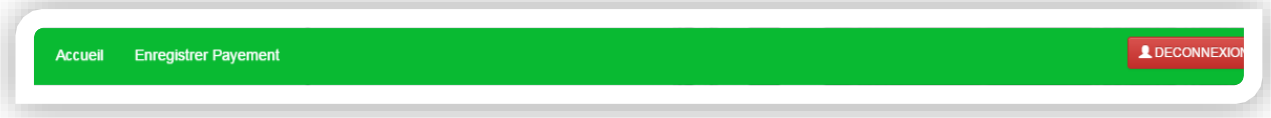


Figure 20: Menu Agent comptable

- Espace bénéficiaire

Figure 21: Menu Bénéficiaire

2. Interfaces de l'application

Nous allons présenter quelques fonctionnalités de la plateforme. Toutefois.

- ✓ Interface Accueil :

Cette interface sur la figure 22 constitue la page d'accueil de la plateforme, l'accès est autorité à toute personne (étudiants, PER, PAT et autres) voulant obtenir des informations sur les listes des bénéficiaires et les contacts du personnel du CROUS de l'Université Assane SECK de Ziguinchor. A partir de cette fenêtre, l'utilisateur peut uniquement consulter la liste des bénéficiaires de chaque classe en cliquant sur l'onglet ***Liste bénéficiaire***.

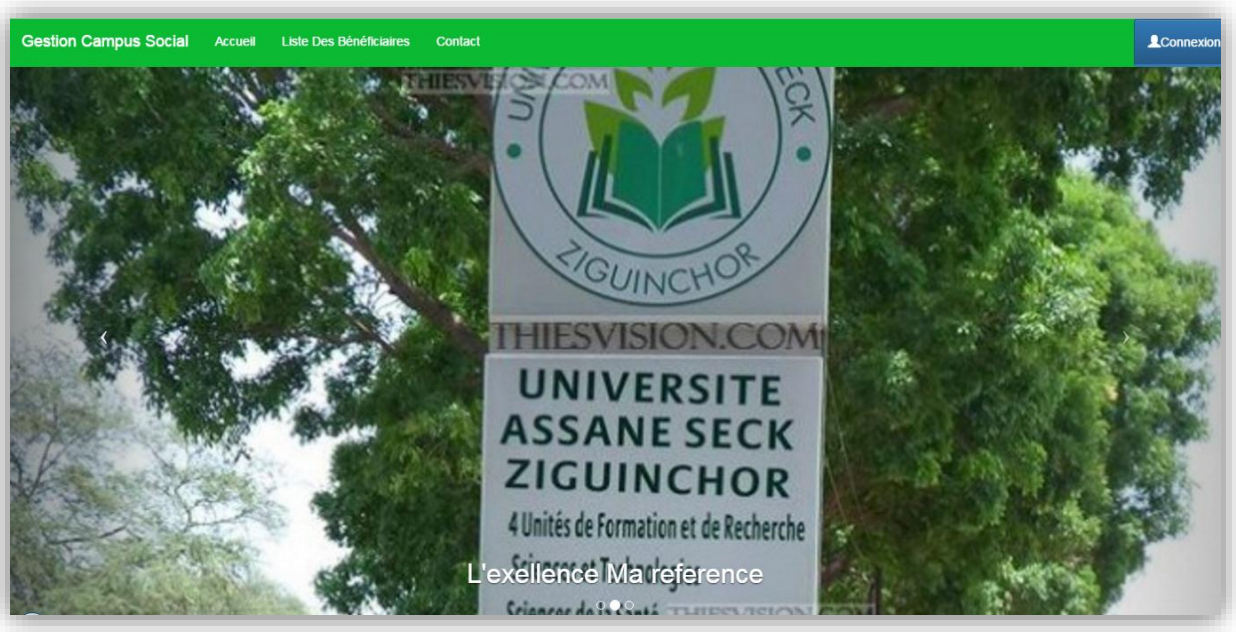


Figure 22: Interface Accueil

Cette page présentée sur la figure 23, est la fenêtre qui s'ouvre après avoir cliqué sur l'onglet ***Liste bénéficiaire***. Elle permet de chercher un étudiant dans une liste de bénéficiaire en donnant l'identifiant de sa classe. A partir de cette liste l'étudiant saura s'il est bénéficiaire ou pas.

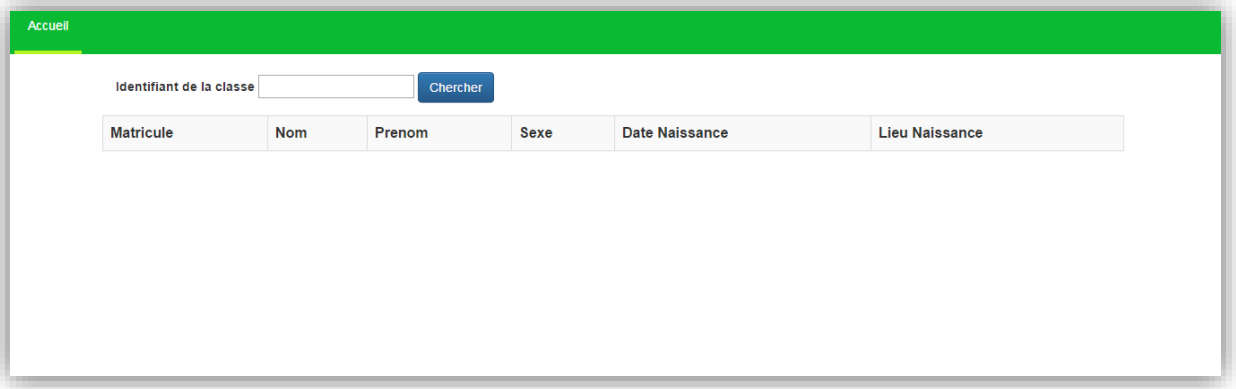


Figure 23: Recherche d'un bénéficiaire dans une classe

Cette liste est un exemple de résultat de recherche, l'utilisateur a choisi l'identifiant M2 MIAGE et la liste des bénéficiaires de cette classe s'affiche comme le montre la figure 24.

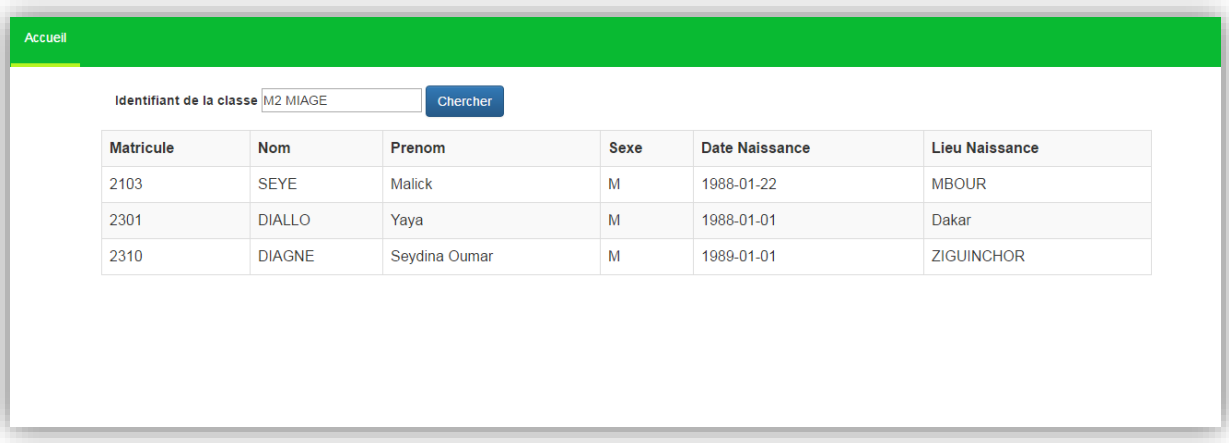


Figure 24: Liste des bénéficiaires d'une classe

Toujours dans cette même interface Accueil, les utilisateurs qui ont un code d'accès peuvent se connecter avec leur nom d'utilisateur et mot de pass en cliquant sur le bouton Connexion. Si l'authentification est réussi, le système ouvre l'interface correspond au profil de l'utilisateur en question.

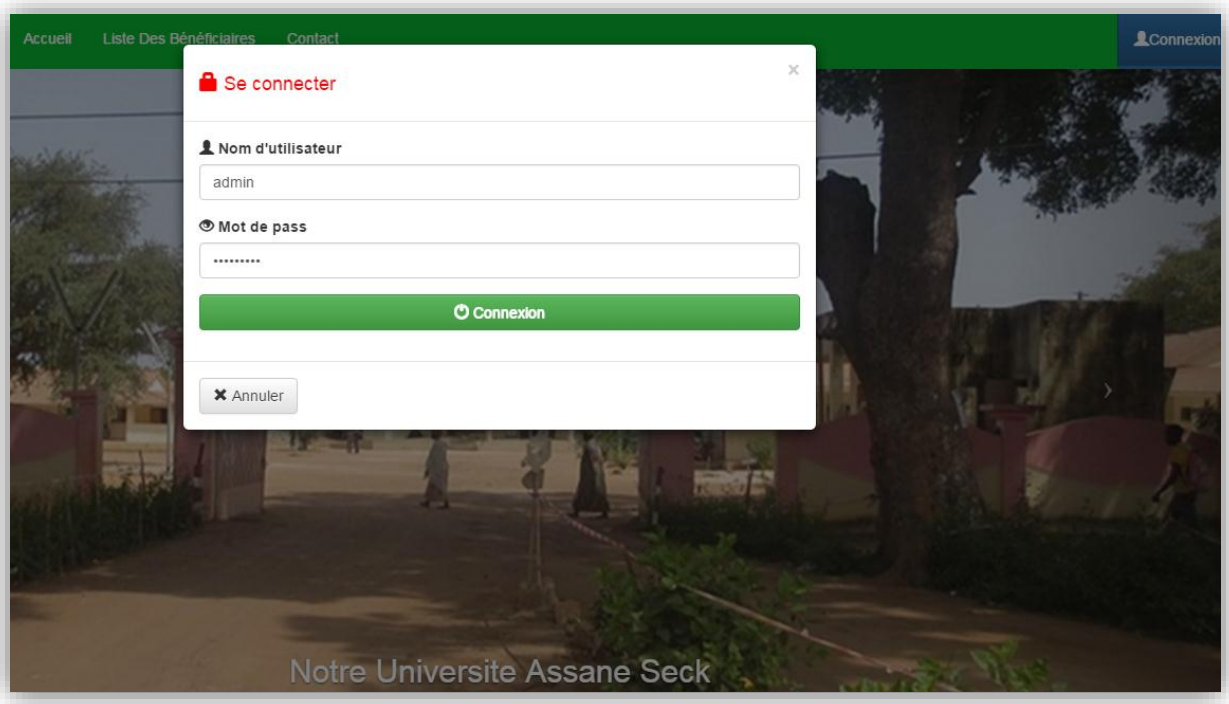


Figure 25: Interface Connexion du système

✓ Interface Administrateur :

Cette interface représentée sur la figure 26 est réservée à l'administrateur. Elle permet à l'administrateur d'ajouter, de modifier et de supprimer des utilisateurs, des comptes et des rôles (profil). Après l'ajout des utilisateurs, l'administrateur octroie chacun eux un compte rattaché au moins à un rôle. De ce fait chaque rôle détermine un profil et donne accès à une interface. Nous avons six différents rôles que :

- ROLE_ADMIN : c'est le profil permettant d'accéder à l'interface Admin ;
- ROLE_CR : c'est le profil qui donne accès à l'interface résidence;
- ROLE_CP : c'est le profil autorisant l'accès à l'interface pavillon ;
- ROLE_AC : c'est le profil donnant accès à l'interface Agent comptable;
- ROLE_UFR : c'est le profil qui permet à l'accès à l'interface chefs de service pédagogique ;
- ROLE_BEN : c'est le profil qui autorise l'accès à l'interface Bénéficiaires ;

Figure 26: Interface Administrateur :

La fenêtre sur figure 27 s'ouvre lorsque l'administrateur clique sur l'onglet **Utilisateur**. Elle donne la liste des utilisateurs existants et la possibilité d'en ajouter d'autres car un formulaire va s'ouvre en cliquant sur ***Ajouter utilisateur***. A partir de cette page, nous pouvons accéder à l'onglet **Compte**. La fenêtre représentée sur la figure 28, donne la liste des comptes, offre la possibilité d'ajouter de nouveaux comptes et permet d'accéder à l'onglet ***Rôle.*** Ce dernier, représenté sur la figure 29, donne la liste des rôles existants et permet d'en ajouter d'autres.

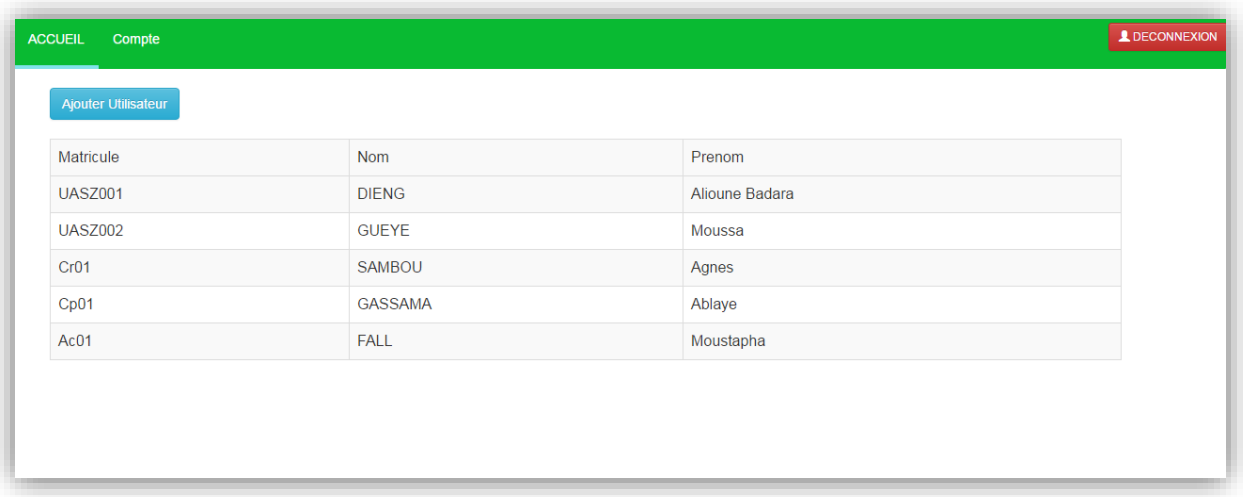


Figure 27: Liste des utilisateurs

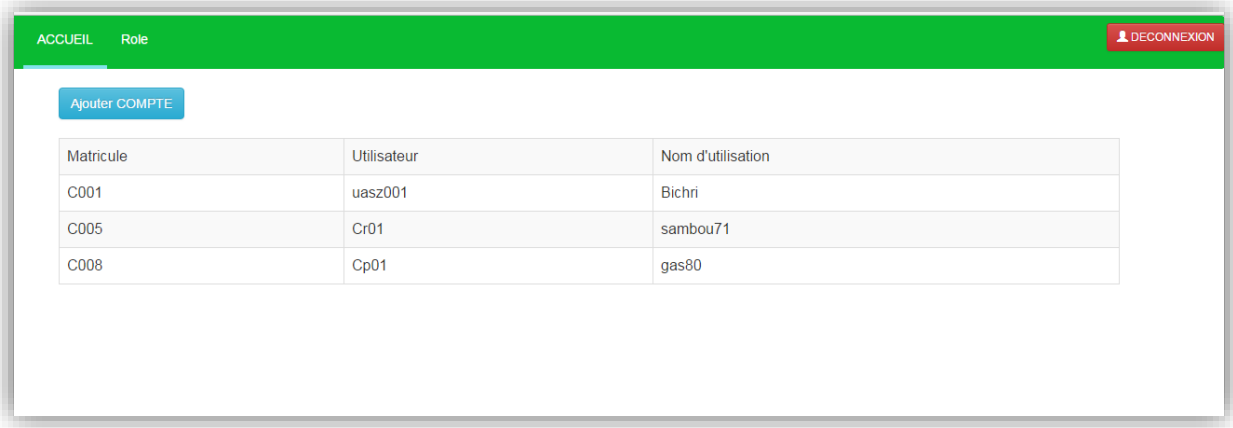


Figure 28:Liste des comptes

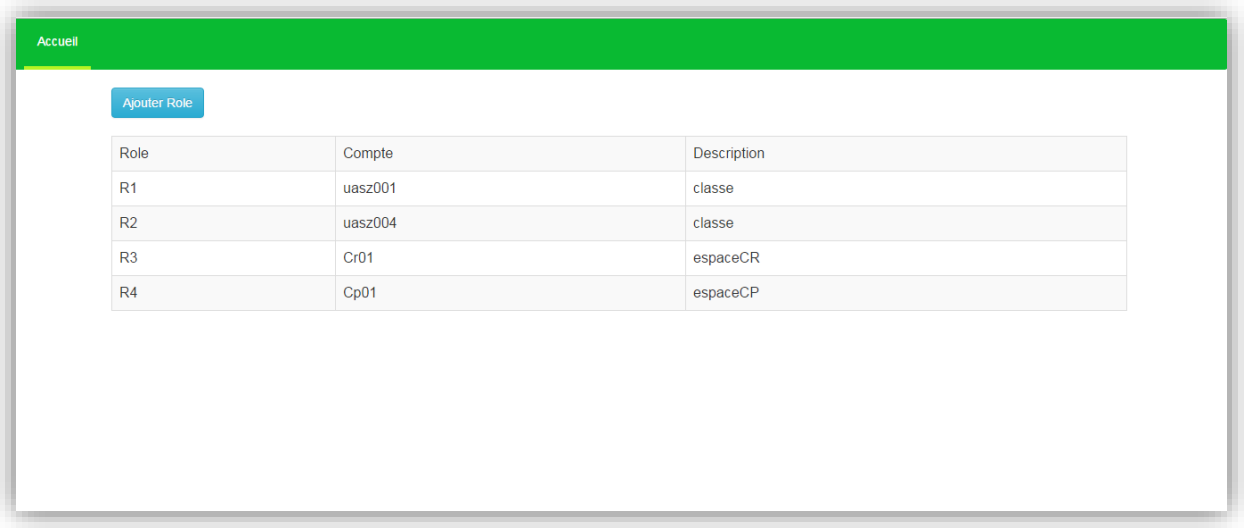


Figure 29: Liste des rôles

- ✓ Interface Chef de résidence :

Une fois que le chef de résidence se connecte, l'interface représentée sur la figure 30 apparait pour lui permettre d'ajouter, de modifier et de supprimer des résidences du campus social, des pavillons dans chacune d'elles, des chambres de chaque pavillon et des lits dans chaque chambre. En plus de ces fonctionnalités, rappelons que le chef de résidence a la possibilité d'effectuer les tâches du chef de pavillon, raison pour laquelle nous avons l'onglet ***Chef de pavillon*** dans cette interface.

Figure 30: Interface chef de résidence :

SI l'utilisateur clique sur l'onglet ***Résidence*** puis ***ajouté résidence,*** le formulaire représenté sur la figure 31 s'ouvre, après avoir renseigné tous les champs et clique sur le bouton « **ajouter** » la résidence est enregistrée dans la base. Il peut aussi cliquer sur le bouton « annuler » ou « fermer » au besoin. Ce processus est valable pour les autres onglets ***Pavillon***, ***Chambre*** et ***Lit*** (voir figure 32).

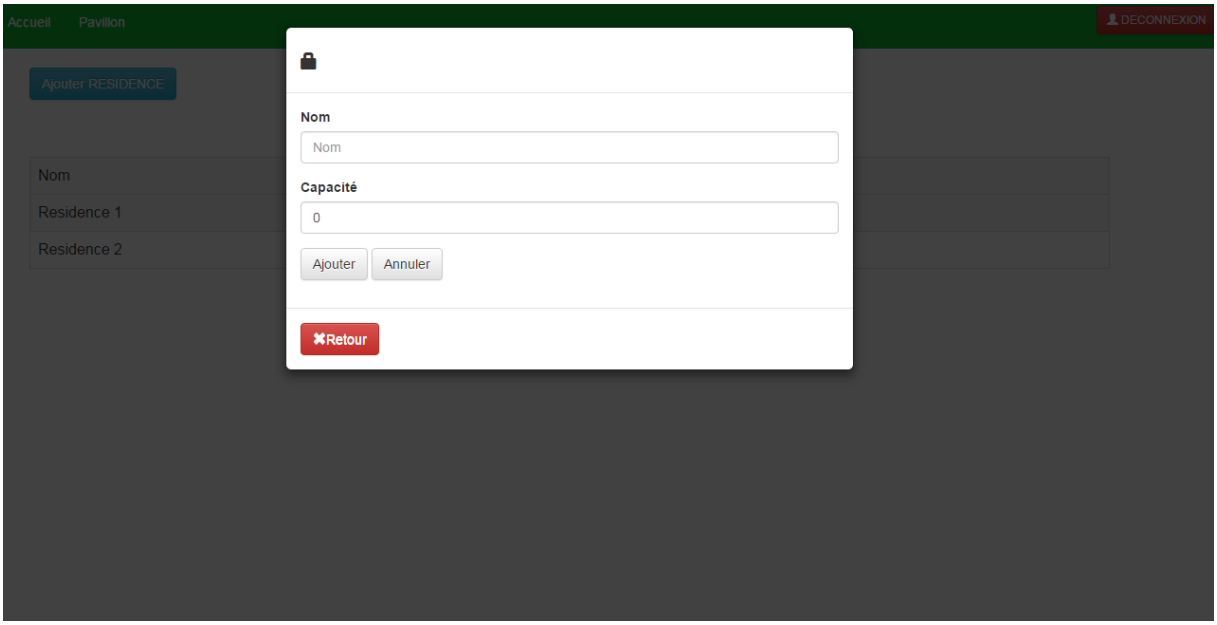


Figure 31:Ajout des résidences :

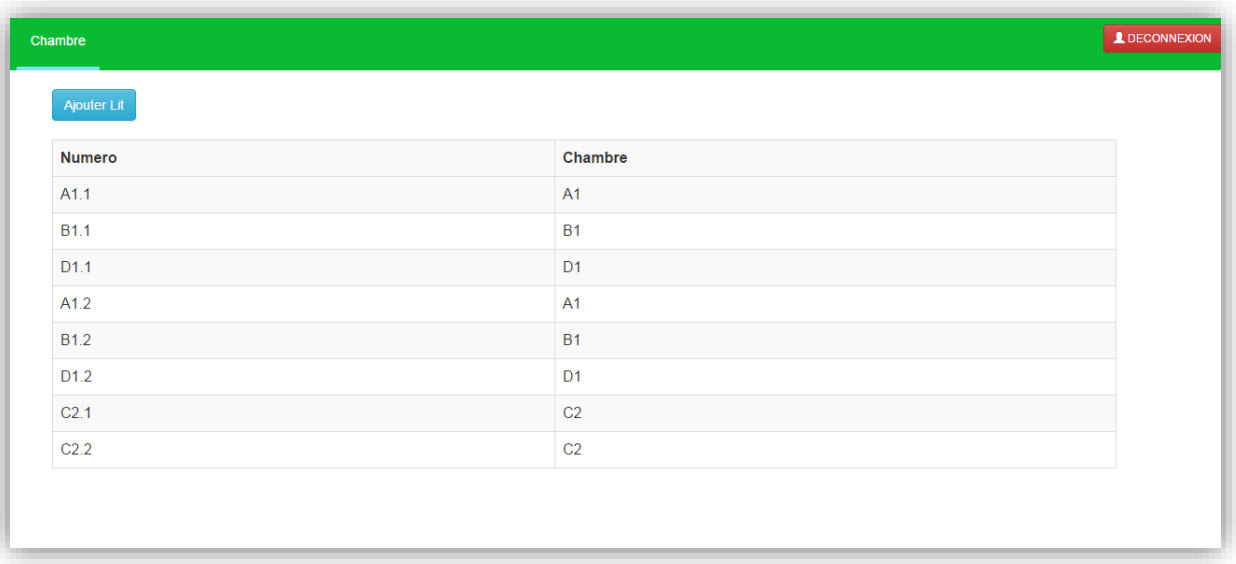
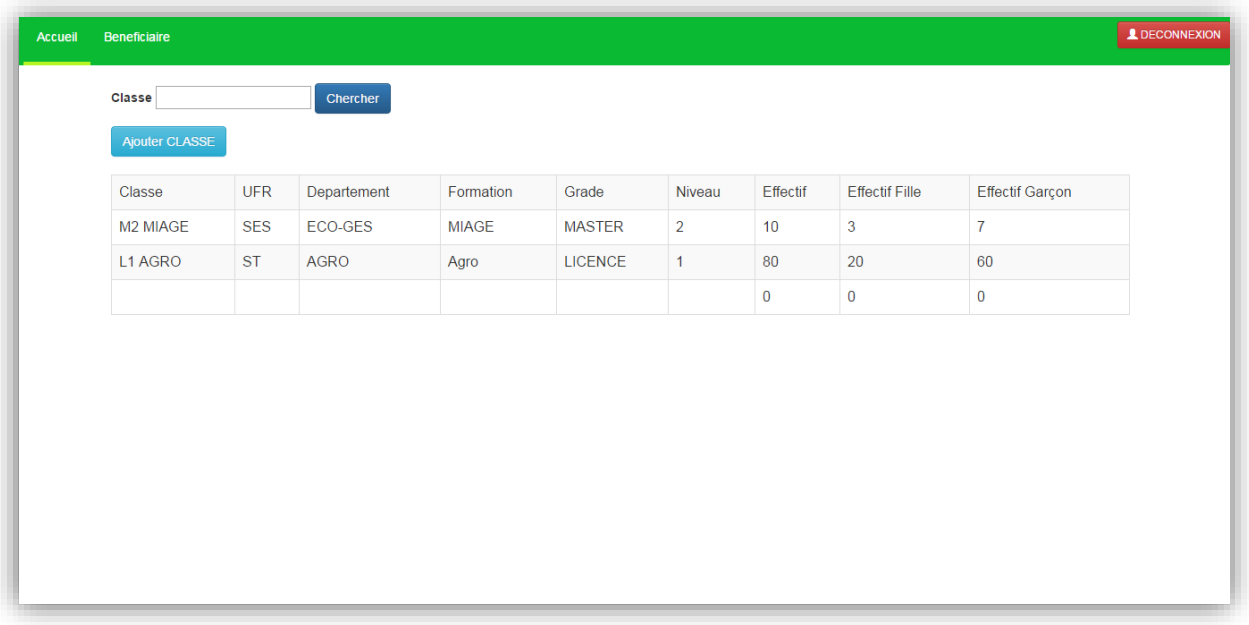


Figure 32:Liste des lits

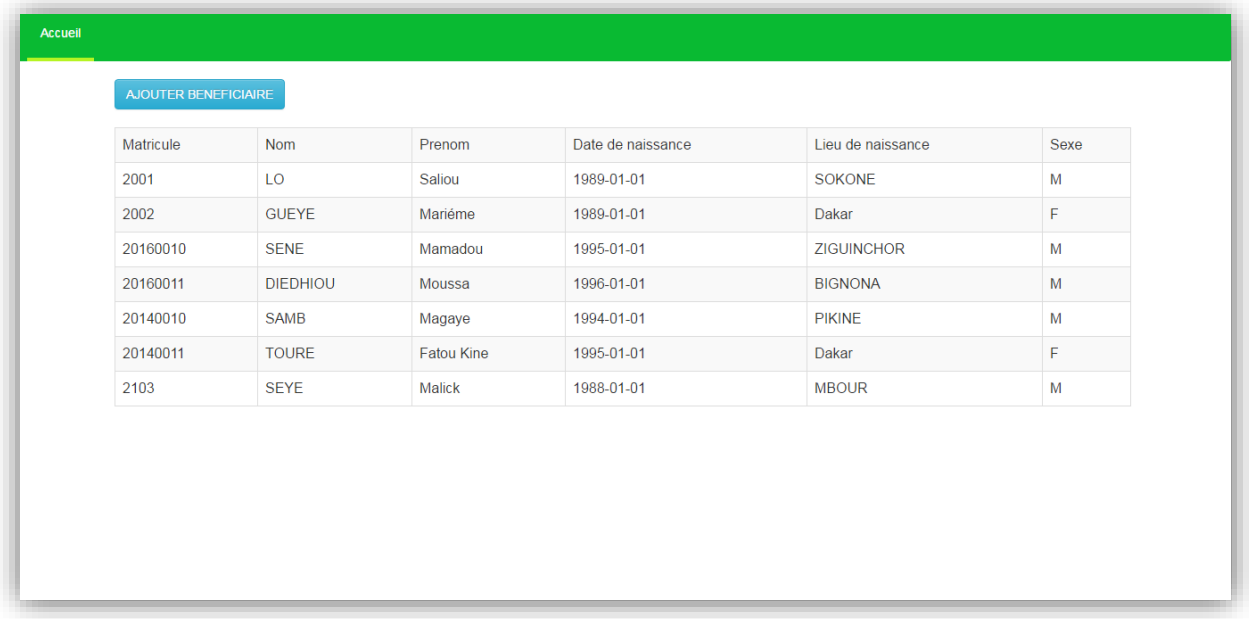
✓ Interface Chef de service de pédagogique :

La fenêtre comme nous montre la figure 33 permet aux chefs de service pédagogique de chaque d'insérer toutes les classes des formations de l'université. En cliquant sur l'onglet bénéficiaire, l'utilisateur a la possibilité de voir la liste des bénéficiaires de chacune de ces classes et d'en ajouter (Figure 34).



Classe	UFR	Departement	Formation	Grade	Niveau	Effectif	Effectif Fille	Effectif Garçon
M2 MIAGE	SES	ECO-GES	MIAGE	MASTER	2	10	3	7
L1 AGRO	ST	AGRO	Agro	LICENCE	1	80	20	60
						0	0	0

Figure 33: Interface chef de service pédagogique



Matricule	Nom	Prenom	Date de naissance	Lieu de naissance	Sexe
2001	LO	Saliou	1989-01-01	SOKONE	M
2002	GUEYE	Mariéme	1989-01-01	Dakar	F
20160010	SENE	Mamadou	1995-01-01	ZIGUINCHOR	M
20160011	DIEDHIOU	Moussa	1996-01-01	BIGNONA	M
20140010	SAMB	Magaye	1994-01-01	PIKINE	M
20140011	TOURE	Fatou Kine	1995-01-01	Dakar	F
2103	SEYE	Malick	1988-01-01	MBOUR	M

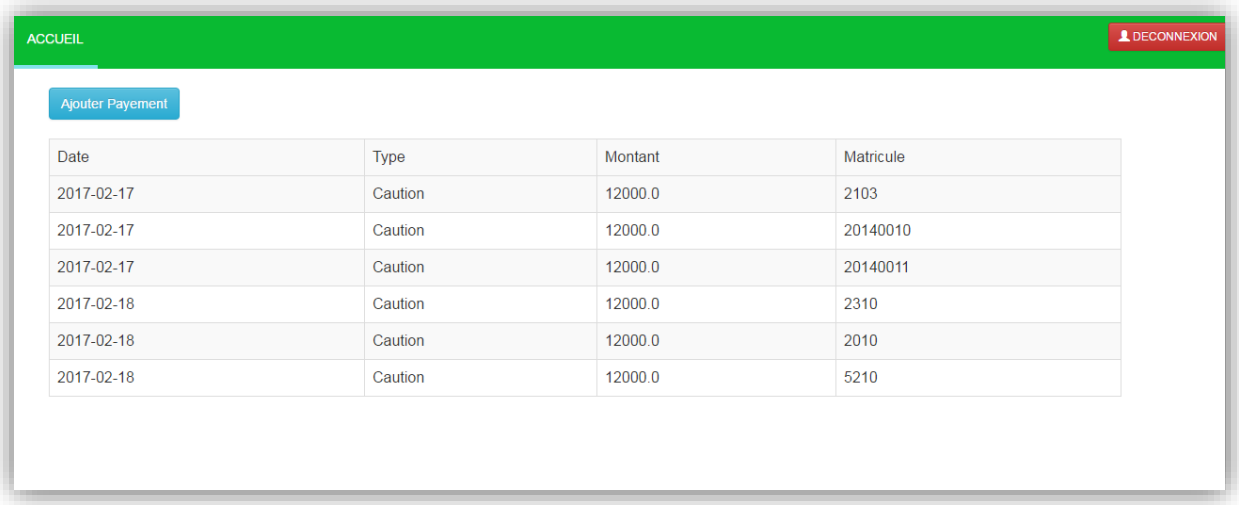
Figure 34:Liste des bénéficiaires des classes

✓ Interface Agent comptable :

Une fois que l'agent comptable se connecte sur la plateforme, le système ouvert l'interface représentée sur la figure 35. Il a la possibilité d'enregistrer le paiement d'une caution et/ou des mensualités pour chaque bénéficiaire en cliquant sur l'onglet ***Enregistré paiement***. Une

nouvelle fenêtre s'ouvre et affiche la liste des paiements et la possibilité d'en enregistrer d'autres (Voir figure 36).

Figure 35: Interface Agent comptable



Date	Type	Montant	Matricule
2017-02-17	Caution	12000.0	2103
2017-02-17	Caution	12000.0	20140010
2017-02-17	Caution	12000.0	20140011
2017-02-18	Caution	12000.0	2310
2017-02-18	Caution	12000.0	2010
2017-02-18	Caution	12000.0	5210

Figure 36:Liste des payements

✓ Interface chef de pavillon :

L'interface sur la figure 37 concerne le chef de pavillon. Elle lui permet d'abord de valider les inscriptions en cliquant sur l'onglet ***valider les inscriptions***. Une fois cliqué sur cet onglet l'utilisateur verra la liste des bénéficiaires qui ont déjà payé leur caution et dont l'inscription n'est pas encore valider comme le montre la figure 38. Il peut en ce moment valider les inscriptions de ces bénéficiaires. Le système envoie à chacun d'eux une notification de validation avec un nom d'utilisateur et un mot de pass de connexion. Ensuite, il peut ***visualiser les réclamations*** faites par les bénéficiaires et faire de leur suivi de résolution représenté sur la figure 39. Enfin, il peut consulter les bénéficiaires de son ***pavillon*** avec leur état de paiement. A partir de cette fenêtre le chef de pavillon pourra rechercher un bénéficiaire dans son pavillon (voir figure 40).

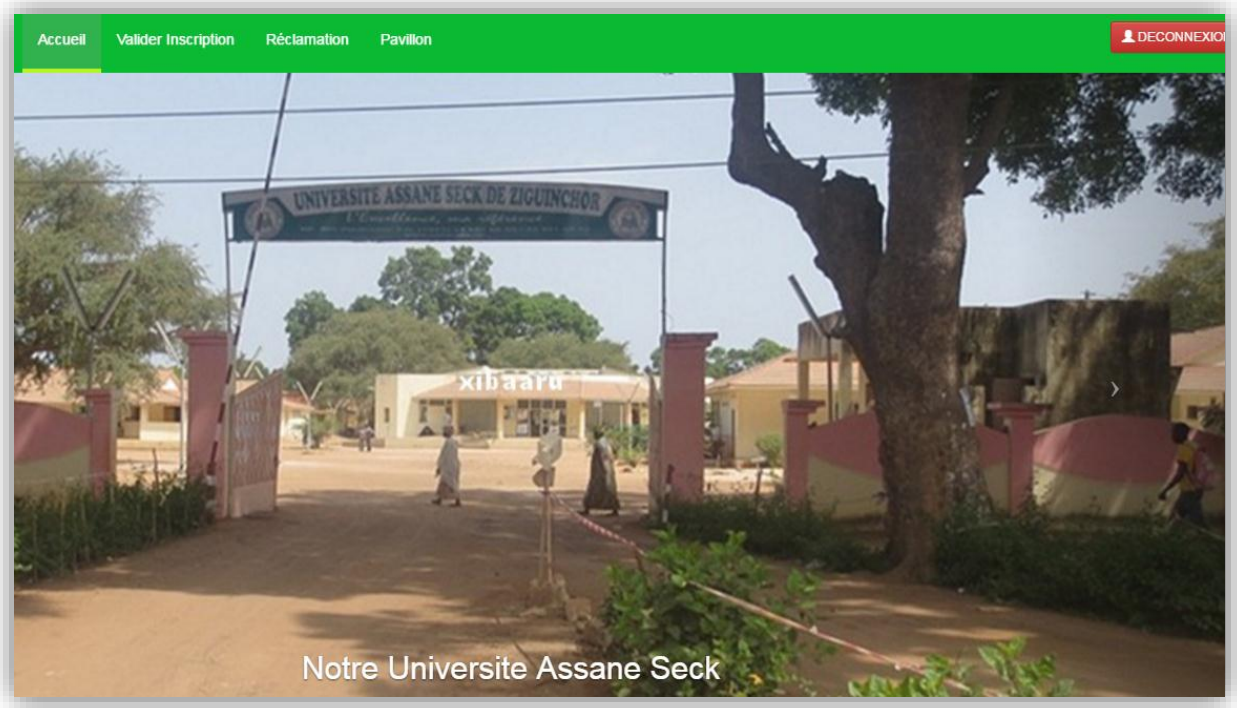


Figure 37: Interface pavillon

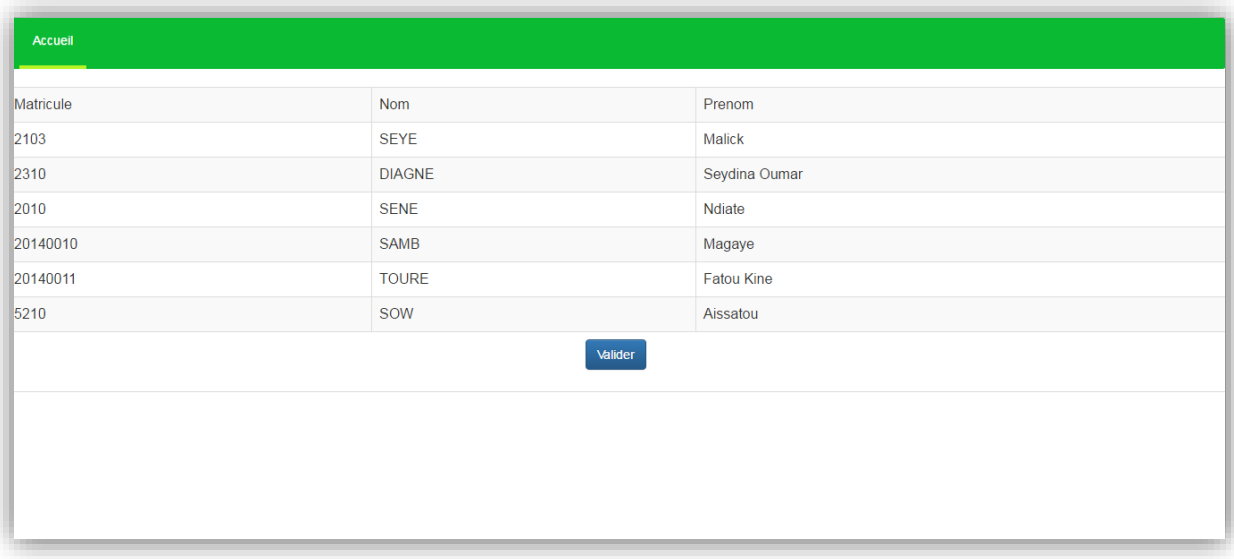


Figure 38: Bénéficiaire déjà payé

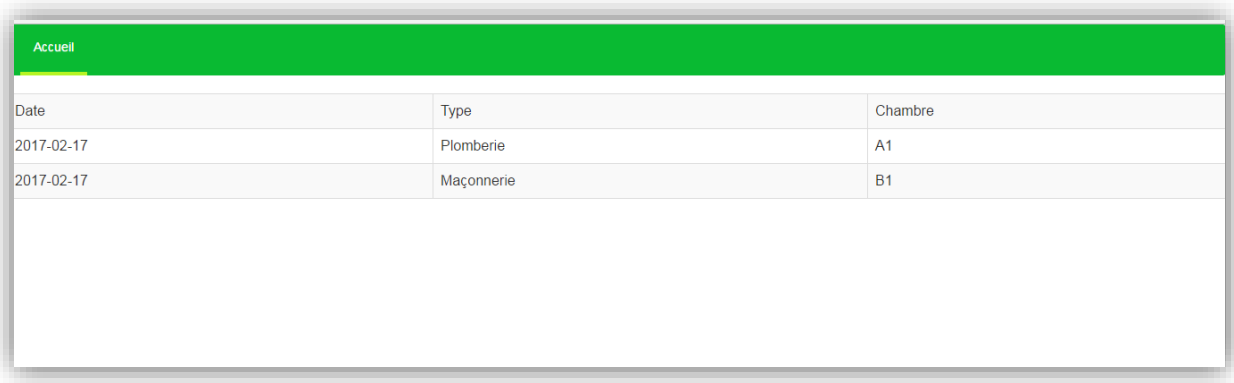


Figure 39: Réclamation des bénéficiaires

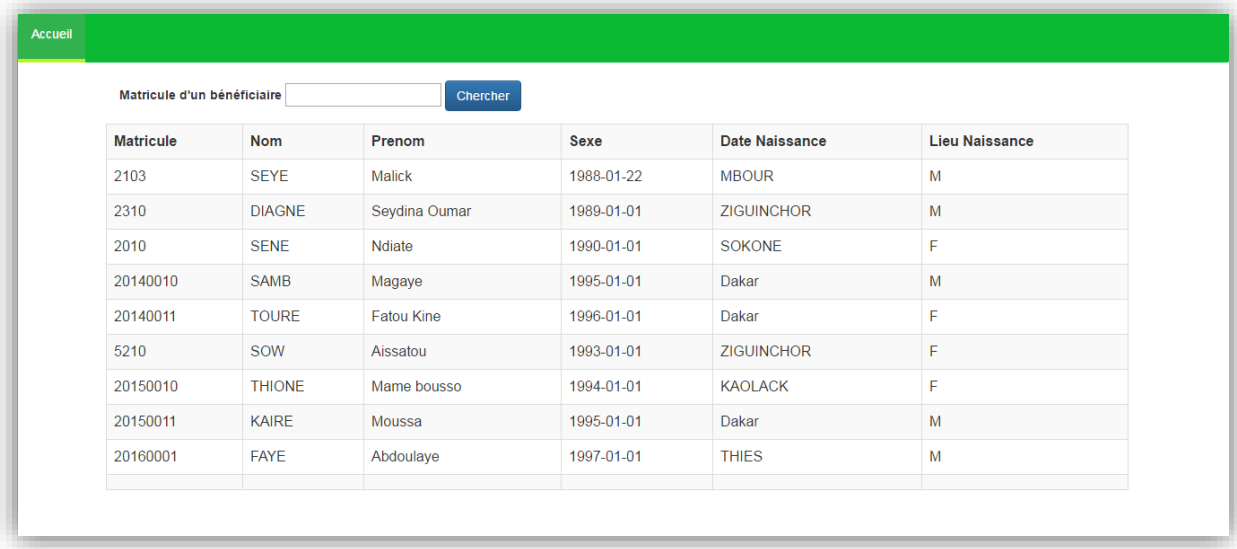


Figure 40: Bénéficiaires d'un pavillon

✓ Interface bénéficiaire

Après validation de leur inscription par le chef de pavillon qui affirme qu'ils ont déjà payé leur caution, les bénéficiaires reçoivent une notification de la part du système qui contient leur nom d'utilisateur et leur mot de pass. Grace à ces codes d'accès, ils peuvent accéder à la plateforme via l'interface représentée sur la figure 41. Ainsi, chaque bénéficiaire a la possibilité de consulter ses états de paiement (caution, mensualité) comme le montre la figure 42. Il peut aussi faire des réclamations représentées par la figure 43, qui seront transmis au chef pavillon concerné

Figure 41: Interface bénéficiaire

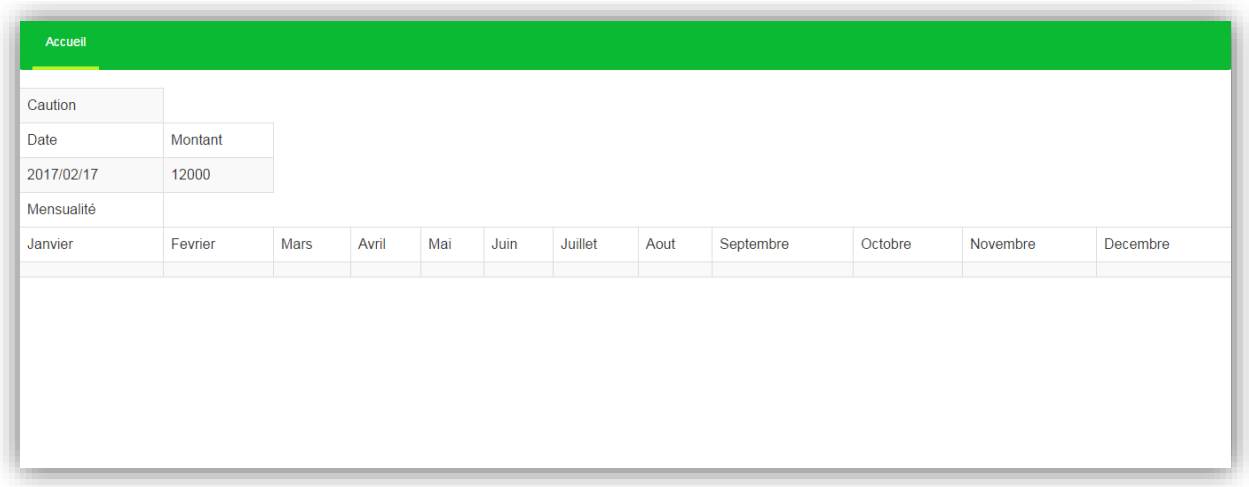


Figure 42: Etat paiement bénéficiaire

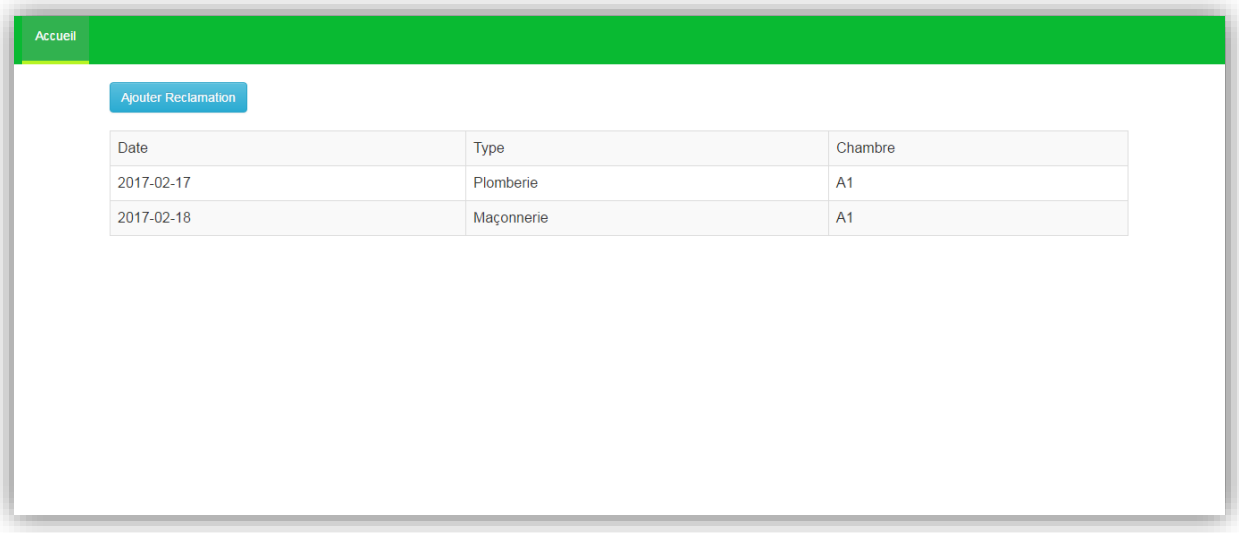


Figure 43: réclamation d'un bénéficiaire

Conclusion

Ce chapitre nous permis de présenter de façon détaillée l'utilisation de la plateforme avec ses différentes fonctionnalités ou interface. Il nous a permis de démontrer l'intérêt de l'application.

Conclusion générale

1- Bilan

Tout système d'information est construit pour satisfaire les besoins des futurs utilisateurs ; cette expression des besoins constitue la base de définition initiale du système à réaliser. L'expérience que nous avons acquise dans la commission sociale de l'amicale des déléguées des étudiants et les enquêtes effectuées au niveau du personnel du CROUS nous ont permis de déceler des défaillances dans la gestion du campus social qui est manuelle. C'est la raison pour laquelle nous avons porté notre réflexion sur cette problématique en proposant une solution informatique qui consiste à concevoir un système d'information pour la gestion du campus social du Centre Régional des Œuvres Universitaires et Sociales de Ziguinchor

Ainsi, dans ce mémoire nous avons donné dans le premier chapitre l'état de l'art qui consiste à faire une brève présentation de l'université Assane Seck de Ziguinchor et son centre des œuvres. Nous avons montré que les moyens utilisés par le CROUS dans la gestion du campus social sont manuels et ne permettent pas d'avoir les performances voulues. C'était aussi l'occasion d'exposer les différentes méthodes de conception de système d'information existante et d'en choisir UML pour la modélisation en se basant sur des critères biens fondés. La modélisation est représentée dans le deuxième chapitre, il présente les diagrammes de cas d'utilisation, de séquences et le diagramme de classe. Le modèle logique de donnée qui constitue la base de données est issue à la transformation du diagramme classe. Les modèles obtenus sont implémentés selon une architecture trois tiers. En plus de cela, les langages de programmation, le SGBD, le serveur d'application et les outils de modélisation utilisés après étude comparative sont détaillés dans le troisième chapitre. La présentation de la plateforme développée pour faciliter sa prise en main et montrer ces fonctionnalités constitue le quatrième et dernier chapitre.

2- Critique

Un système d'information n'est jamais parfaite car les personnes qui l'apprécie ou l'utilise, détectent perpétuellement ses limites. D'aucuns peuvent apporter d'autres solutions plus efficaces, plus cohérentes et plus pertinentes sur ce problème. Notre solution a ses limites qui sont dues soit à la maitrise insuffisante du framework Spring MVC que nous n'avons pas pu exploiter à fond, soit au manque d'information purement stratégique de la part du personnel

compétent. Ces raisons font que certaines fonctionnalités de la plateforme ne sont pas encore intégrées dans le système.

3- Perspectives

Un travail scientifique n'est jamais achevé à part entière d'où l'importante de décliner à chaque fois des perspectives. Dans cette présente plateforme, nous comptons utiliser les fichiers « csv » dans les listes de passage qui vont alléger la tâche aux Chefs de services pédagogiques des UFR. Dans l'avenir nous pouvons intégrer les autres domaines de la gestion du campus social dans le système d'information comme la gestion du patrimoine (Matelas, draps, couvertures etc.). Aussi les ouvriers peuvent avoir l'accès à la plateforme pour signaler les réparations effectuées.

Bibliographie:

[B1] **http://profs.vinci-melun.org/profs/adehors/CoursWeb2/Cours/Ch1/Ch1.php**

[B2] Michel Lissandre « Maitriser SADT

[B3] https://fr.wikipedia.org/wiki/M%C3%A9thodes_d'analyse_et_de_conception

[B4] MELESE .J, *Approche systémique des organisations,* Paris : Hommes et Techniques, 1979.

[B5] Pascal Roques, *UML 2 - Modéliser une application Web*, Eyrolles, 2007.

[B6] Pascal Roques, *UML 2 par la pratique - Études de cas et exercices corrigés*, Eyrolles, 2006.

[B7] M. Bigan, P.Bourey, D. Corbeel, H. Camus., *Conception des Systèmes d'informations*, Modélisation des données, TECHNIP, Paris, 2006.

[B8] https://www.olats.org/schoffer/archives/defarchi.htm

[B9]https://www.google.sn/search?q=architecture+trois+tiers&biw=1366&bih=700&source=lnms&tbm=isch&sa=X&sqi=2&ved=0ahUKEwj4wML7mqbSAhWKLMAKHWk0C4oQ_AUIBigB#tbm=isch&q=architecture+trois+tiers+en+JEE&imgrc=uDCQ1SZ4DP8DYM:

[**B10]**http://www.objis.com/formation-java/tutoriel-spring-introduction-spring-mvc.html

[B11] Wikipédia, «Injection de dépendances,» 10 07 2016. [En ligne]. Available: https://fr.wikipedia.org/wiki/Injection_de_d%C3%A9pendances.

[B12] Wikipédia, «Programmation orientée aspect,» 10 07 2016. [En ligne]. Available: https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_aspect.

[B13] Wikipédia, «Injection de dépendances,» ,10 07 2016. [En ligne]. Available: https://fr.wikipedia.org/wiki/Injection_de_d%C3%A9pendances

[B14] Wikipédia, «Programmation orientée aspect,» 10 07 2016. [En ligne]. Available: https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_aspect.

[B15] Tutorialspoint, «Spring Framework - Architecture,» 10 07 2016. [En ligne]. Available:

http://www.tutorialspoint.com/spring/spring_architecture.htm.

[B16]
http://www.ecllipseformation.meximas.com/site/Java/Tutoriels/J2EE/Introduction.html