Ministère de l'Enseignement Supérieur de la Recherche et de l'Innovation

Université Assane Seck de Ziguinchor

UFR Sciences et Technologies

Département Informatique

Mémoire de fin d'études master

Mention : Informatique

Spécialité : Génie Logiciel

Sujet :

Conception et Implémentation d'une Application de Gestion des Tickets et des Accès au Restaurant Universitaire

Par : M. Mactar CAMARA

Soutenance le 11/05/2018

Membres du jury

- **Pr. Salamon SAMBOU (Président)**
- **Dr. Ibrahima DIOP (Encadreur)**
- **Dr. Youssou DIENG (Co-encadreur)**
- **Dr. Khadim DRAME (Rapporteur)**
- **Dr. Sérigne DIAGNE (Examinateur)**

Sous la direction de :

- **Dr. Ibrahima DIOP**
- **Dr. Youssou DIENG**

Sous la supervision de :

- **Pr. Salomon SAMBOU**

Année Universitaire : 2016 – 2017

Résumé

Le Centre Régional des Œuvres Universitaires et Sociales de Ziguinchor (CROUS-Z) occupe une place très importante au sein de l'Université. Il est chargé de la gestion des œuvres sociales destinées aux étudiants afin de leur offrir de meilleures conditions de vie. Son service de restauration joue un rôle capital au sein de cette direction et est fondamental dans la vie quotidienne de l'étudiant. Ainsi pour mener à bien sa mission, ce service de restauration peut avoir un système informatique pour la gestion de ses activités quotidiennes.

L'augmentation considérable du nombre d'étudiants à l'UASZ, a entrainé une complexité dans la gestion des activités du service de restauration particulièrement celle des tickets et de l'accès au restaurant. Ainsi, cela nous a conduits à proposer, dans le cadre de ce mémoire de fin d'étude de master, une application informatique pour la gestion des tickets et de l'accès au restaurant. Cette application permet l'automatisation de la gestion de tickets et d'accès au restaurant de l'Université. Elle permettra au service de restauration du CROUS-Z de :

- dématérialiser les tickets de restaurant ;
- gérer les comptes avec leurs crédits ;
- gérer les ventes de tickets (Créditer) ;
- gérer les accès au restaurant universitaire (Débiter) ;
- gérer les utilisateurs et leurs droits d'accès.

Mots clés: **Gestion des tickets, Gestion des accès, Restaurant Universitaire, CROUS-Z**

Abstract

The center of social university works is very important in the university. It is in charge of charity destined for student with a view to offer them better living condition. Its canteen service play a key role in this department and it is fundamental in daily life of student. So to complete successfully its work, the latter have to have at it command computer system for the management of its daily activities.

Nowadays the number of student at Assane Seck University is huge, this caused the complexity of service canteen, particularly luncheon voucher and entrance to the canteen. That why, as part of our master's dissertation, we have propose a reliable and optimal application for the management of luncheon voucher and the entrance to the canteen. This application allow for the automation of luncheon voucher and the entrance to the canteen management. With this application, the canteen service will be able to manage:

- dematerialize restaurant tickets;
- manage accounts with their credits;
- manage ticket sales;
- manage access to the university restaurant;
- manage users and their access rights.

Keywords: **Ticket management, management access, University restaurant, CROUS-Z**

Remerciements

Je tiens à exprimer mes très sincères remerciements à mes encadreurs Dr. Ibrahima DIOP et Dr. Youssou DIENG pour leur disponibilité, leurs conseils et leurs encouragements qui m'ont permis de réaliser ce travail dans les meilleures conditions.

Je tiens à remercier vivement les membres de jury qui ont accepté d'évaluer mon mémoire. Votre participation à mon jury de soutenance est un grand honneur pour moi.

J'adresse aussi mes reconnaissances à tous les professeurs du département d'Informatique de l'UASZ, pour la richesse et la qualité de leurs enseignements et les efforts fournis pour assurer à leurs étudiants une bonne formation.

Je voudrais aussi exprimer ma gratitude envers tous ceux qui m'ont accordé leur soutien, tant par leur gentillesse que par leur dévouement.

Je ne peux nommer ici toutes les personnes qui de près ou de loin m'ont aidé et encouragé mais je les en remercie vivement.

Enfin je tiens à dire combien le soutien quotidien de ma famille a été important tout au long de ces quelques années, je leur dois beaucoup.

Dédicaces

A ALLAH Le Tout Miséricordieux, ton amour, ta miséricorde et Tes grâces à mon endroit m'ont fortifié dans la persévérance et l'ardeur au travail.

A mon Père, Amadou Moussa CAMARA en vous, je vois un père dévoué à sa famille. Ta présence en toute circonstance m'a maintes fois rappelé le sens de la responsabilité.

A ma Mère, Adama DIEDHIOU en vous, je vois la maman parfaite, toujours prête à se sacrifier pour le bonheur de ses enfants. Merci pour tout.

A mes frères et sœurs pour qui, je le sais, ma réussite est très importante. Que Dieu vous Paye pour tous vos bienfaits.

À toute ma famille particulièrement mon tuteur et sa femme, pour leurs soutiens, leurs conseils partagés

A tous mes amis.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

2TUP: Two Track Unified Process

BD: Base de Données

COUD: Centre Œuvres Universitaires de Dakar

CROUS-Z: Centre Régional des Œuvres Universitaires et Sociales de Ziguinchor

DSE: Direction du Service aux Etudiants

EAR: Entreprise Archives

EJB: Entreprise Java Bean

Frk: Foreign Key

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

IHM : Interface Homme Machine

JEE : Java Entreprise Edition

JPA : Java Persistence API

JSF : Java Server Face

JSP : Java Server Pages

JTA : Java Transaction API

MLD : Modèle Logique de Données

MLDR : Modèle Logique de Données Relationnel

MPD : Modèle Physique de Données

MVC : Modèle Vue Contrôleur

PrK: Primary Key

PHP: HyperText Preprocessor

RU: Restaurant Universitaire

RUP: Rational Unified Process

SI : Système d'Information

SGBD : Système de Gestion de Base de Données

SQL : Structured Query Langage

UASZ : Université Assane Seck de Ziguinchor

UML: Unified Modeling Language

XHTML: eXtensible HyperText Markup Langage

XP: eXtreme Programming

Introduction générale

Actuellement, l'informatique connait une avancée technologique considérable dans tous les secteurs. Elle joue un rôle important dans le développement de l'entreprise et d'autres établissements comme la restauration en particulier celle universitaire.

Le restaurant universitaire fait partie des établissements que l'informatique pourra beaucoup aider. Il consiste à proposer un repas complet aux étudiants à prix subventionnés. Les informations traitées au sein de cette structure sont capitales. Elles sont indispensables et permettent à cette structure de pouvoir décider afin de prévoir un budget.

La gestion du restaurant est une activité qui occupe une place importante au niveau du service de restauration de l'université. La gestion manuelle du restaurant ralentit et rend l'exécution du travail très lente. Ceci entraine une mauvaise organisation du travail au sein des restaurants universitaires. La croissance du nombre des étudiants nécessite la mise en place d'une application de gestion rapide. Ces problèmes que rencontrent cette structure seront relatés en détail dans le chapitre qui suit tout en proposant des solutions et fixer des objectifs.

Ainsi, l'automatisation de la gestion des tickets et des accès au restaurant universitaire devient indispensable pour le service qui s'occupe de la restauration vu le nombre important de données à traiter au sein de ce dernier. C'est dans ce cadre que se situe ce présent mémoire. Il comprend quatre (04) chapitres.

Le premier chapitre décrit le sujet du mémoire. En effet, il permet d'exposer les problèmes que rencontre le service de restauration et de dégager la problématique du sujet. Enfin, ce premier chapitre permet de fixer les objectifs de ce présent mémoire.

Le deuxième chapitre intitulé spécification et analyse des besoins fonctionnels permet de faire une présentation des acteurs et des fonctionnalités du système de « gestion des tickets et des accès au restaurant universitaire ». Il intègre les diagrammes de cas d'utilisation ou (use case) montrant les relations entre les acteurs et les fonctionnalités du système. Il permet aussi de faire l'analyse des besoins fonctionnels du système en utilisant des diagrammes d'activité et de séquence.

Le troisième chapitre porte sur la conception du système. Il aborde l'ensemble des diagrammes de composants, de package et de déploiement, mais aussi la conception détaillée illustrant les diagrammes de classes et du dictionnaire de données nécessaire pour la réalisation du projet.

Le quatrième et dernier chapitre dénommé implémentation et présentation du système illustre le codage des différentes entités, les interfaces, mais aussi les outils de développement utilisés pour l'implémentation et fait une présentation générale de l'application.

Chapitre I : Description du sujet

Ce premier chapitre porte sur la description du sujet. Il aborde les présentations du Centre des Œuvres Universitaire et Sociales de Ziguinchor (CROUS-Z) et du **service de restauration** de l'Université Assane Seck de Ziguinchor. Il est aussi question de décrire les problèmes rencontrés dans sa gestion au sein de ce service, d'aborder la problématique du sujet et d'exposer les objectifs du projet.

I. De la Direction du Service aux Etudiants DSE au Centre Régionale des Œuvres Universitaires de Ziguinchor CROUS-Z

Placée sous la tutelle du Rectorat, la direction du service aux étudiants (DSE) était chargée de la gestion des œuvres sociales destinées aux étudiants. Cette direction était chargée de l'accompagnement afin d'offrir de meilleures conditions de vie aux étudiants.

Cette DSE est remplacée par le Centre des Œuvres Universitaires de Ziguinchor qui est installé le 13 novembre 2017. Nouvellement adoptée par l'Assemblée Nationale en sa séance du vendredi 19 février 2016, cette loi a été introduite dans un contexte d'élargissement de la carte universitaire pour offrir aux étudiants un meilleur cadre de vie, améliorer la gouvernance et la gestion des œuvres sociales universitaires. Le CROUS-Z a pour mission d'améliorer les conditions de vie des étudiants. Il s'occupe:

- de l'hébergement des étudiants ;
- de leur restauration ;
- de leur prise en charge médicale ;
- de l'animation sociale, culturelle et sportive du campus ;
- et des affaires estudiantines (ressources financières et service à la communauté, etc.).

Il est composé de plusieurs services:

- le service du logement ;
- le service de la restauration ;
- le service médical ;
- le service de l'animation sociale, culturelle et sportive.

Dans la partie suivante, nous faisons une présentation du restaurant universitaire.

II. Présentation du restaurant universitaire

Un restaurant universitaire (RU) est une cantine, un lieu de restauration collective destiné aux étudiants. Les restaurants universitaires ont pour objectif de servir un repas complet à prix modéré. Ils sont situés à proximité des sites universitaires. Grâce aux restaurants universitaires, les étudiants issus de tous les milieux ont la possibilité de prendre à l'extérieur de chez eux un repas par jour. Ce restaurant fonctionne tous les jours pendant l'année universitaire à des heures bien précises pour le petit déjeuner, le déjeuner et le diner. Il faut des tickets pour accéder au restaurant ; ces derniers sont vendus à 75 francs pour le petit déjeuner et 150 francs pour les déjeuner et diners.

Le restaurant universitaire est géré par l'intermédiaire d'un repreneur privé à qui le directeur du CROUS-Z confie une mission de servir des repas aux étudiants, sur la base d'un cahier de charges et d'un appel d'offres fait sur la base du code des marchés publics. Il fonctionne par le biais d'un formatage c'est-à-dire que le CROUS-Z met à la disposition du privé (gérant) tout le matériel nécessaire. Ce dernier (le repreneur) l'exploite pour le compte du CROUS-Z moyennant une contrepartie.

Ainsi, un contrat est signé pour que le CROUS-Z verse un montant tous les 10 jours pour le compte du gérant. Ce repreneur est accompagné d'un personnel déjà en place pour assurer le bon fonctionnement du restaurant. Ce personnel appartenant au CROUS-Z est composé de :

- un chef de cuisine ou chef cuisine qui élabore l'ensemble des plats proposés à la carte du restaurant tout en coordonnant l'activité de la cuisine avec celle de la salle pour assurer le service. Il veille aussi au respect des normes de sécurité et d'hygiène dans sa salle de cuisine.
- un maitre d'hôtel qui est chargé de la coordination de l'ensemble du personnel de service, dont le service de table.
- des contrôleurs, généralement deux, dont le premier représentant le privé se charge de la facturation des tickets et le second mandaté par le CROUS-Z fait un contrôle sur les tickets.

Sachant que la gestion revient toujours pour le compte du privé c'est-à-dire au repreneur.

Cependant, malgré les efforts qu'ils effectuent à l'endroit des étudiants, nous remarquons qu'ils rencontrent d'énormes problèmes dans sa gestion particulièrement dans l'achat et la vente de tickets, mais aussi dans l'accès au restaurant. Nous allons essayer de relater dans la partie qui suit, les insuffisances et défaillances dans son mode de fonctionnement.

III. Problèmes dans la gestion du restaurant

La restauration occupe une place importante et indispensable au sein du CROUS-Z. Malgré cela, ce restaurant rencontre d'énormes problèmes dans sa gestion de vente de tickets et d'accès à cause de son mode de fonctionnement manuel. Nous avons essayé dans cette partie d'énumérer les problèmes liés à son fonctionnement manuel :

- les longues files d'attente pour acheter des tickets ;
- les longues files d'attente pour accéder au restaurant ;
- les passages multiples des étudiants pour un repas ;
- difficulté d'établir des bilans journaliers ;
- difficulté du partage des recettes entre CROUS-Z et repreneurs ;
- difficulté de disposer de données et de statistiques sur l'accès des étudiants au RU.

IV. Les objectifs de notre projet

Après avoir détecté les insuffisances dans la procédure actuelle sur la gestion du restaurant universitaire particulièrement sur la gestion des tickets et des accès au restaurant, notre solution consiste à concevoir et implémenter un système de gestion du RU qui comblera les manquements et les défaillances notées.

Les objectifs spécifiques de ce travail sont:

- dématérialiser les tickets de restaurant ;
- gérer les comptes et leurs crédits ;
- gérer les ventes de tickets(Créditer) ;
- gérer les accès au restaurant universitaire(Débiter) ;
- implémenter une base de données complète pour la gestion du restaurant.

Pour parvenir à une bonne résolution des problèmes rencontrés par le service de restauration en matière de sa gestion des tickets et des accès, nous optons la méthode (ou processus) unifiée **2TUP**.

V. Cadre méthodologique

Pour modéliser d'une manière claire et précise la structure et le comportement de notre système indépendamment de tout langage de programmation, nous allons adopter le langage de modélisation UML (Unified Modeling Language) et la démarche 2TUP.

1. Présentation de UML

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. UML représente le standard de modélisation objet le plus répandu et le plus utilisé aujourd'hui **[1].**

2. Processus Unifié

Un processus de développement unifié est un processus construit sur UML (Unified Modeling Language).Plus exactement, ce sont les meilleures pratiques du développement objet suivies pour la réalisation d'un système. Un processus unifié se distingue par les caractéristiques suivantes **[1]** :

- Itératif : le logiciel nécessite une compréhension progressive du problème à travers des raffinements successifs et développer une solution effective de façon incrémentale par des itérations multiples.
- Piloté par les risques : les causes majeures d'échec d'un projet logiciel doivent être écartées en priorité.
- Centré sur l'architecture : le choix de l'architecture logicielle est effectué lors des premières phases de développement du logiciel. La conception des composants du système est basée sur ce choix.
- Conduit par les cas d'utilisation : le processus est orienté par les besoins utilisateurs présentés par des cas d'utilisation.

Dans la communauté objet, ils existent plusieurs processus unifiés en vogue comme eXtreme Programming (XP) et Rational Unified Process (RUP). Dans notre étude, on a choisi de travailler avec le processus 2TUP ; parce qu'il cible des projets de toute taille et il a pu faire une large place dans le domaine de la technologie et les risques des projets.

3. Le processus 2TUP

Le processus**2TUP** apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information des organisations.

Le processus 2TUP (Two Track Unified Process) est un processus unifié. Il gère la complexité technologique en donnant part à la technologie dans son processus de développement **[1]**.Le 2TUP propose un cycle de développement qui sépare les aspects

techniques des aspects fonctionnels et propose une étude parallèle des deux branches : fonctionnelle (étude de l'application) et technique (étude de l'implémentation). Le processus2TUP s'articule autour de trois branches : Branche technique, Branche fonctionnelle et Branche de conception et réalisation **[1].**

- **Branche fonctionnelle**: les principales étapes de la branche fonctionnelle se présentent comme suit :
  - la frontière fonctionnelle entre le système et son environnement : les activités attendues des différents utilisateurs par rapport au système.
  - l'étape d'analyse : elle consiste à étudier précisément les spécifications fonctionnelles de manière à obtenir une idée de ce que va réaliser le système en termes de métier **[1].**
- **Branche technique :** les principales étapes de la branche technique se présentent comme suit :
  - l'étape capture des besoins techniques: cette étape recense toutes les contraintes sur les choix de technologies pour la conception du système, les outils et le matériel sélectionnés ainsi que la prise en compte des contraintes d'intégration avec l'existant (pré requis d'architecture technique) **[1].**
  - l'étape conception générique: elle définit les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle permet de générer le modèle de conception technique qui définit les Frameworks **[1].**
- **Phase conception – réalisation :** les principales étapes de cette branche se présentent comme suit :
  - l'étape conception préliminaire: Cette étape permet de produire le modèle de conception système. Ce dernier organise le système en composants, délivrant les services techniques et fonctionnels ; ce qui induit le regroupement des informations des branches techniques et fonctionnelles **[1].**
  - l'étape conception détaillée: elle permet d'étudier comment réaliser chaque composant. Le résultat fournit l'image prête à fabriquer du système complet **[1].**
  - l'étape de codage: elle permet d'effectuer la production des composants et les tests des unités de code au fur et à mesure de leur réalisation.

- l'étape de recette: elle consiste à valider les fonctionnalités du système développé.

La **figure1** suivante détaille les étapes de développement des trois branches du processus 2TUP**.**

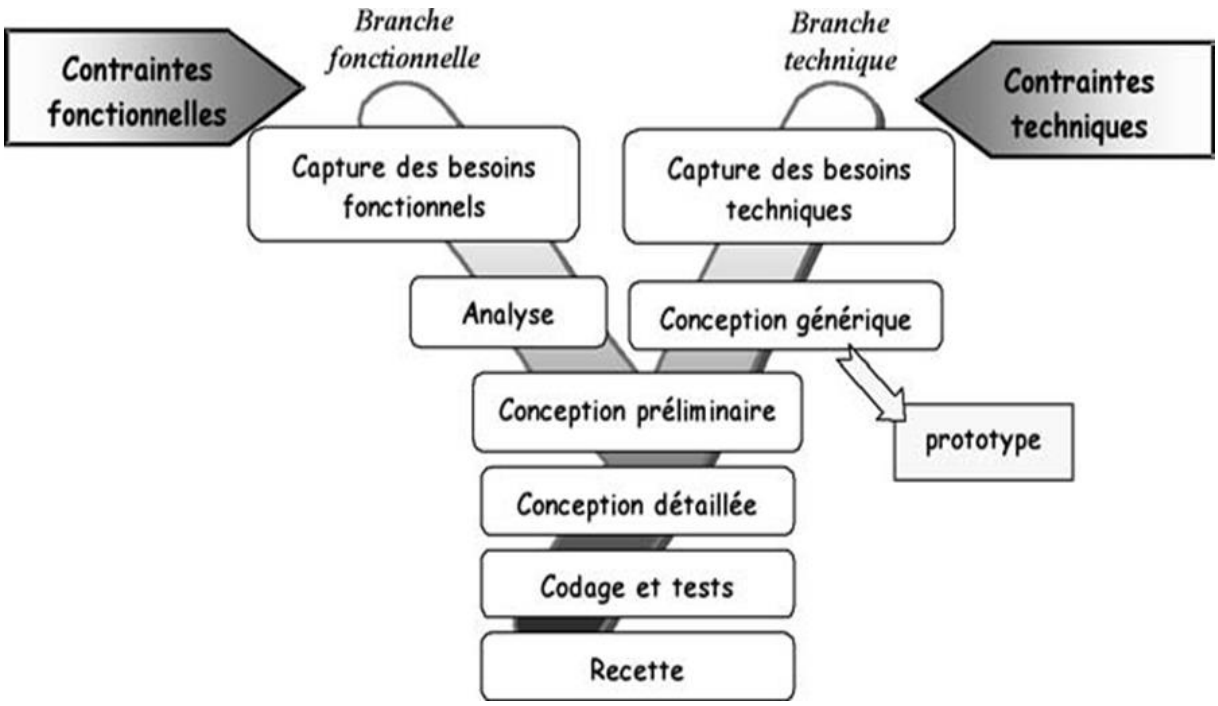

Figure 1: Présentation du processus 2TUP [2]

En résumé ce chapitre a fait l'objet d'une description détaillée du sujet. Il a permis de présenter la structure qui abritera l'application. Le sujet ainsi décrit dans sa globalité a conduit à la description des problèmes rencontrés par le service de restauration de l'Université. Ce chapitre a aussi permis de dégager la problématique du sujet et de fixer les objectifs.

Chapitre II : Spécification et analyse des besoins fonctionnels

Après avoir défini le processus de développement de l'application, nous allons passer à la phase de spécification et d'analyse des besoins fonctionnels. Dans un premier temps, nous entamerons par la spécification et ensuite l'analyse des besoins fonctionnels. L'analyse nous permet de comprendre les besoins de la structure. Cette compréhension induit évidemment à l'identification des acteurs, à la définition des fonctionnalités du système et à l'élaboration des diagrammes de cas d'utilisation.

Dans ce chapitre, nous nous intéressons à la spécification et à l'analyse des besoins fonctionnels du système de gestion des tickets et des accès au RU de l'Université Assane Seck de Ziguinchor.

I. Spécification des besoins fonctionnels

La spécification des besoins fonctionnels permet de préciser l'étude du contexte fonctionnel du système en décrivant les différentes manières qu'auront les acteurs à l'utiliser. Cette spécification passe par l'identification des différents acteurs et la définition des fonctionnalités du système de gestion de vente des tickets de restaurant de l'Université Assane Seck de Ziguinchor. C'est aussi dans cette partie que nous allons élaborer les diagrammes de cas d'utilisation.

1. Présentation des acteurs du système

Un acteur désigne un rôle externe joué par une personne ou une chose qui interagit avec le système. Les acteurs représentent les rôles joués dans le système. Le **tableau 1** ci-dessous donne l'ensemble des acteurs qui interviendront dans le futur système avec leurs rôles.

Tableau 1: Présentation des acteurs du système

Acteurs	**Rôles**
Vendeur de tickets	**Gérer la vente des tickets :** Activer les comptes, Créditer les comptes des étudiants

	et Inventorier les ventes de ticket effectuées (par repas, par jours, etc.).
Portier	**Gérer accès étudiant** : Vérifier l'identité des étudiants et Débiter les comptes des étudiants lors des repas.
Étudiant	**Gérer son compte** : Activer, Consulter et Créditer son compte et Voir les historiques d'entrées au restaurant et crédit.
Administrateur	**Administrer l'application :** Afficher, Rechercher, Ajouter, Modifier et Supprimer des utilisateurs.

2. Présentation des fonctionnalités du système

Les besoins fonctionnels ou métiers représentent les actions que le système doit exécuter, il ne devient opérationnel que s'il les satisfait. L'étude préliminaire liste l'ensemble des besoins du système. Pour répondre aux problèmes soulevés, nous avons répertorié l'ensemble de fonctionnalités nécessaires pour ce système dans le tableau 2 qui suit :

Tableau 2: Présentation des fonctionnalités du système

Fonctionnalités du système	**Acteurs**
S'authentifier	Tout utilisateur
Activer compte	Vendeur de tickets, Etudiant
Consulter compte	Vendeur de tickets, Etudiant
Créditer compte	Vendeur de tickets, Etudiant
Inventorier les ventes de tickets	Vendeur de tickets
Voir l'historique de ses entrées au restaurant	Étudiant
Voir l'historique de ses achats de tickets	Étudiant
Vérifier identité d'un étudiant	Portier

Débiter compte	Portier
Ajouter utilisateur	Administrateur
Modifier utilisateur	Administrateur
Rechercher utilisateur	Administrateur
Supprimer utilisateur	Administrateur

3. Diagrammes de cas d'utilisation

Le diagramme de cas d'utilisation définit un ensemble d'opérations d'un système tel que l'utilisateur le voit de l'extérieur. Il capture le comportement du système. Cette vision orientée utilisateur nous permet donc d'exprimer les besoins des utilisateurs qui constituent les acteurs du système. Pour ce travail, chaque acteur possède un diagramme de cas d'utilisation. Ces **figures 2, 3, 4 et 5** suivantes représentent les diagrammes de cas d'utilisations des différents acteurs du système :

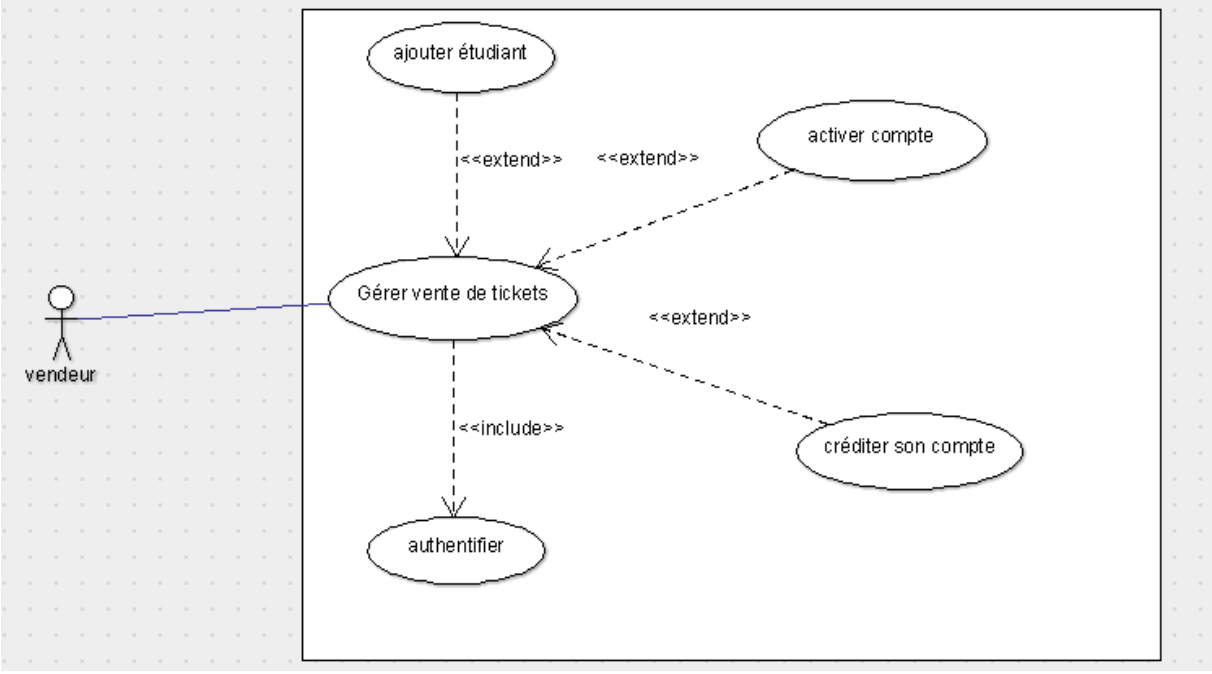

Figure 2: Diagramme de cas d'utilisation Vendeur

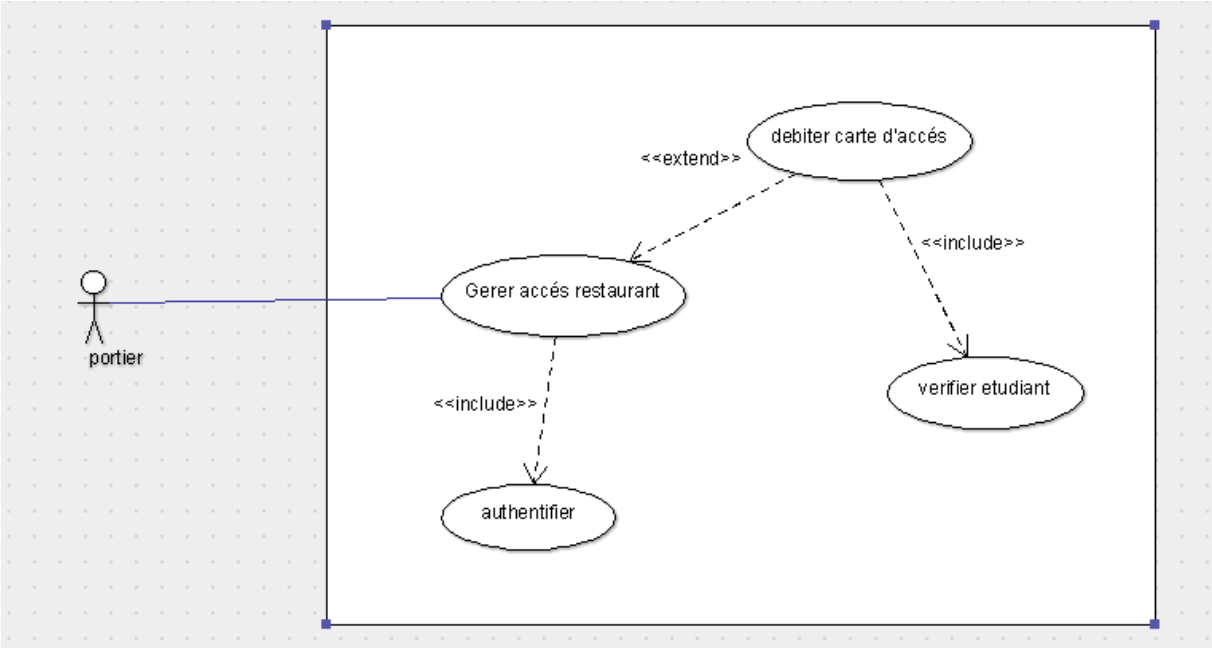

Figure 3: Diagramme de cas d'utilisation Portier

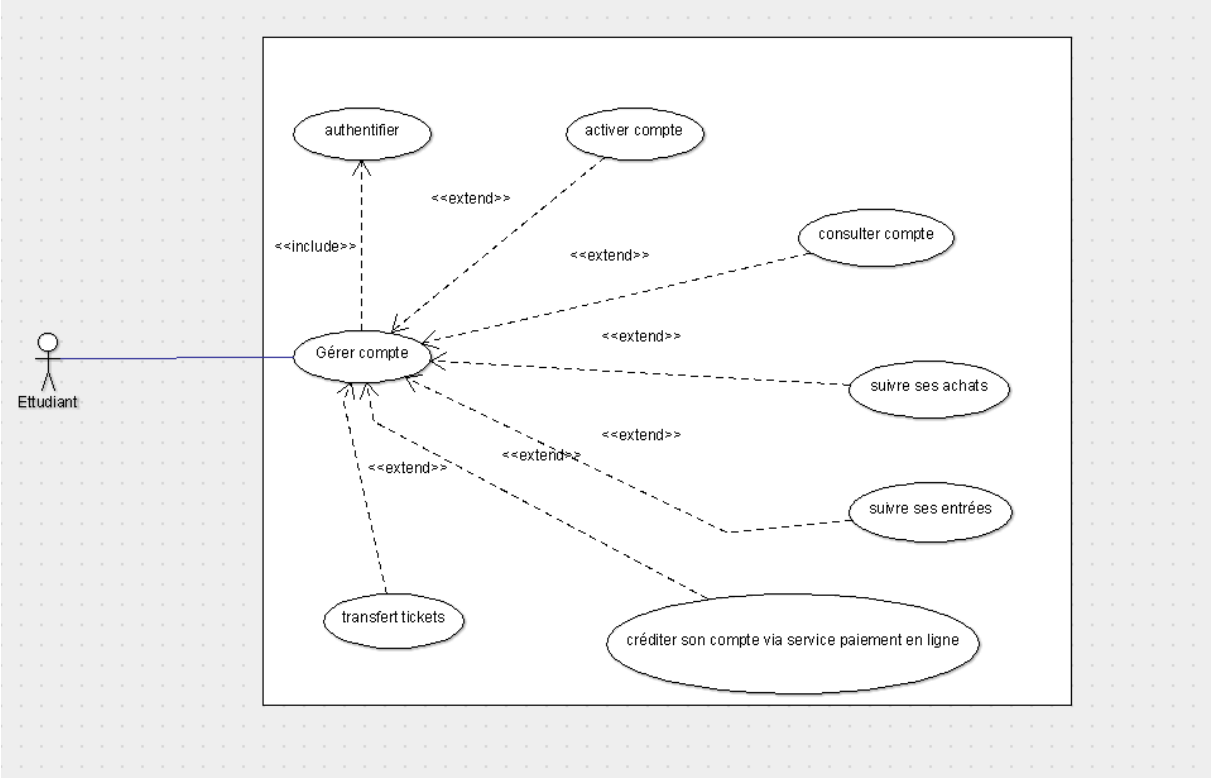

Figure 4: Diagramme de cas d'utilisation Etudiant

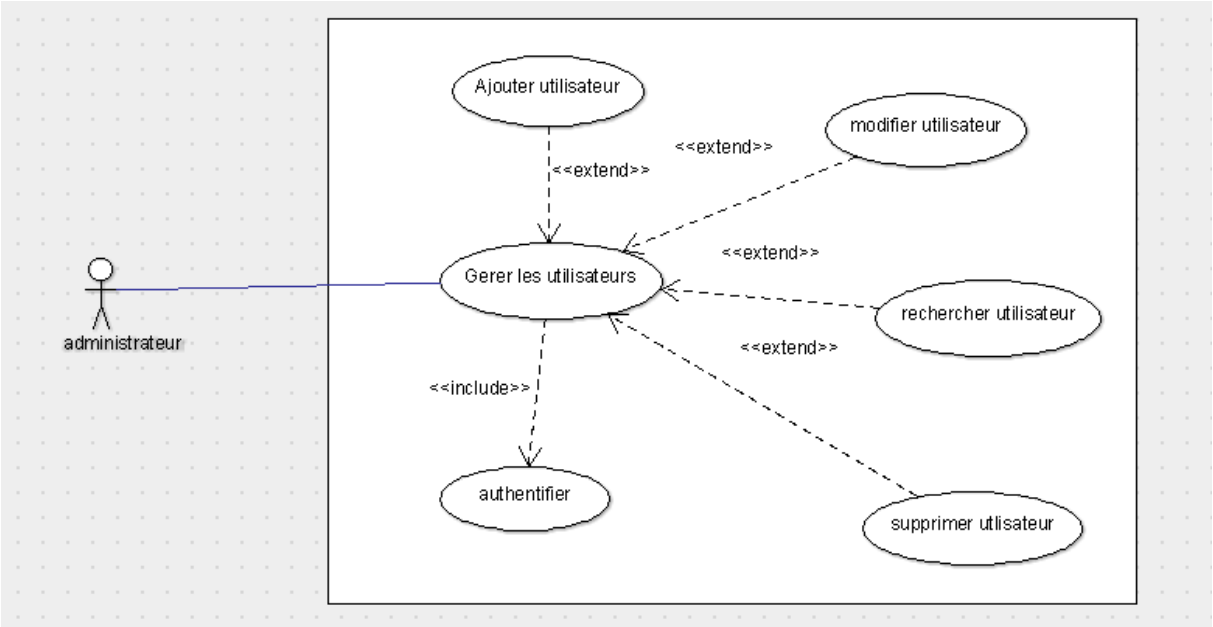

Figure 5:Diagramme de cas d'utilisation Administrateur

II. Analyse des besoins fonctionnels

Dans le processus de développement d'un logiciel, l'analyse constitue une étape très importante. Elle ouvre le système pour établir la structure des objets d'analyse avec des diagrammes d'activité et de séquence.

Dans cette partie, nous allons faire l'analyse des besoins fonctionnels en décrivant les activités des fonctionnalités du système.

❖ **L'activité authentification des utilisateurs**

Le cas d'utilisation « s'authentifier » est composé de plusieurs activités. La première consiste à saisir un login et mot de passe de l'utilisateur. Une fois que cette étape est exécutée, le système procède à une vérification. Si les données saisies sont incorrectes, un message d'erreur s'affiche et l'activité doit être reprise. Elle ne sera effective que si les données sont correctes. Par ailleurs, si les champs sont bien renseignés, elle donne accès à la page d'accueil. L'ensemble de ces enchaînements est décrit dans le diagramme d'activité de la **figure 6** suivante :

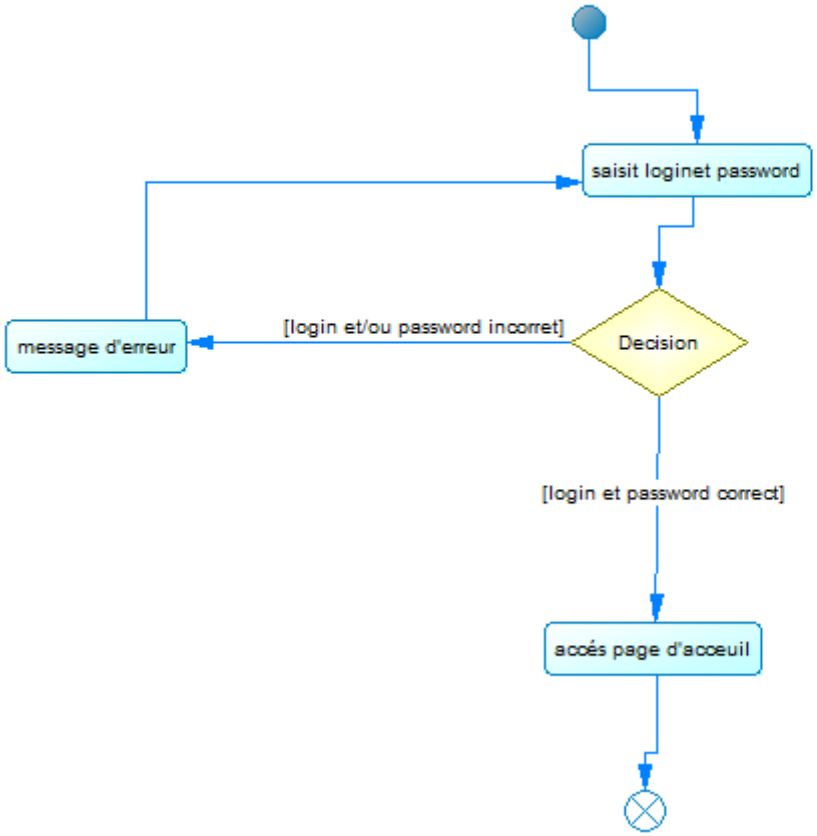

Figure 6: Diagramme d'activité du cas « authentifier »

La description du **tableau 3** et le diagramme de séquence de la **figure 7** ci-dessous montrent le scénario entre l'utilisateur et le système :

- le système demande à l'utilisateur de s'authentifier ;
- l'utilisateur saisit son login et mot de passe ;
- le système envoie les données saisies pour vérification au niveau de la BD ;
- si les informations saisies sont OK ;
- l'utilisateur accède à sa page d'accueil.

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

Description de cas d'utilisation « s'authentifier »	
Titre	S'authentifier
Résumé	Permet de contrôler l'accès au système
Acteur (s)	Vendeur, Portier, Etudiant et Administrateur
Pré condition	Avoir un compte d'utilisateur

Scénario nominal	✓ L'utilisateur saisit son identifiant et son mot de passe ; ✓ Le système vérifie les informations saisies au niveau de la Base de données ; ✓ Le système récupère le profil de l'utilisateur.
Post condition	Accéder à la page d'accueil avec son profil de l'utilisateur
Exception	Saisie d'un identifiant ou d'un mot de passe incorrecte

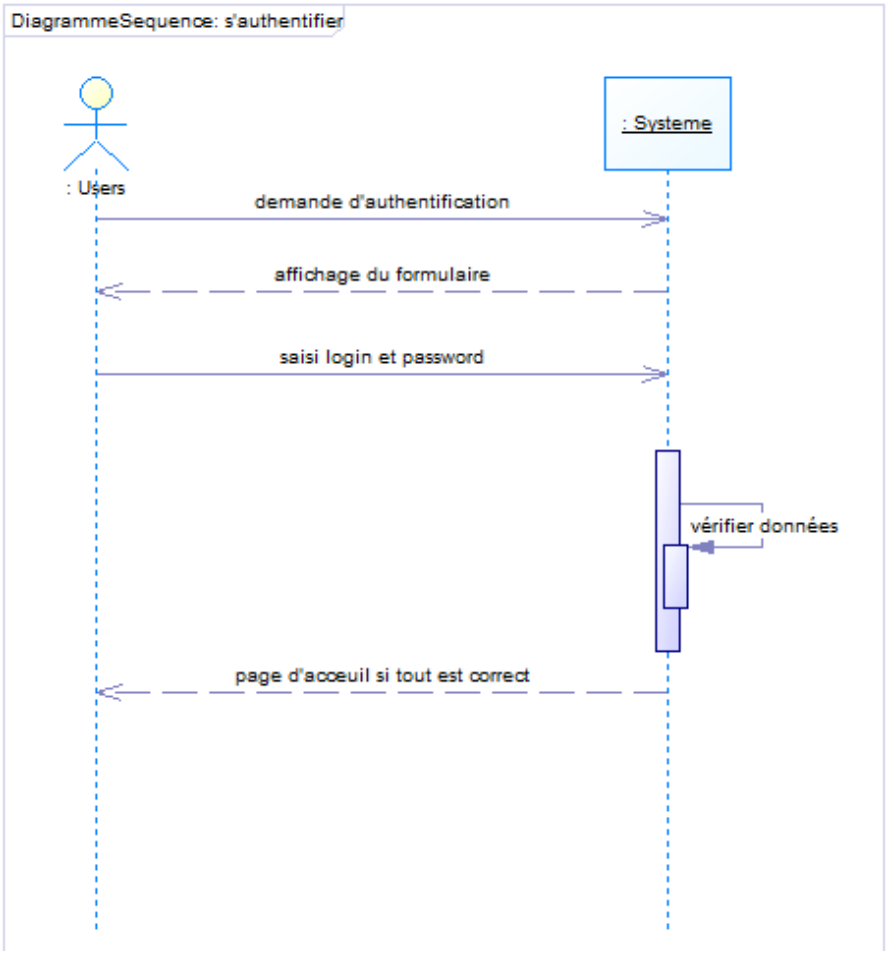

Figure 7: Diagramme de séquence du cas s'authentifier

❖ L'activité vérifier et débiter

L'utilisateur (portier) passe à la vérification sur les informations de l'étudiant en entrant le numéro de carte de ce dernier, le système procède à la vérification des données. Ainsi si les informations saisies sont conformes (le numéro de carte), le système lui renvoie les informations concernant l'étudiant. Il valide en débitant le compte de l'étudiant et lui donne accès dans le restaurant et dans le cas contraire l'étudiant n'aura pas accès.

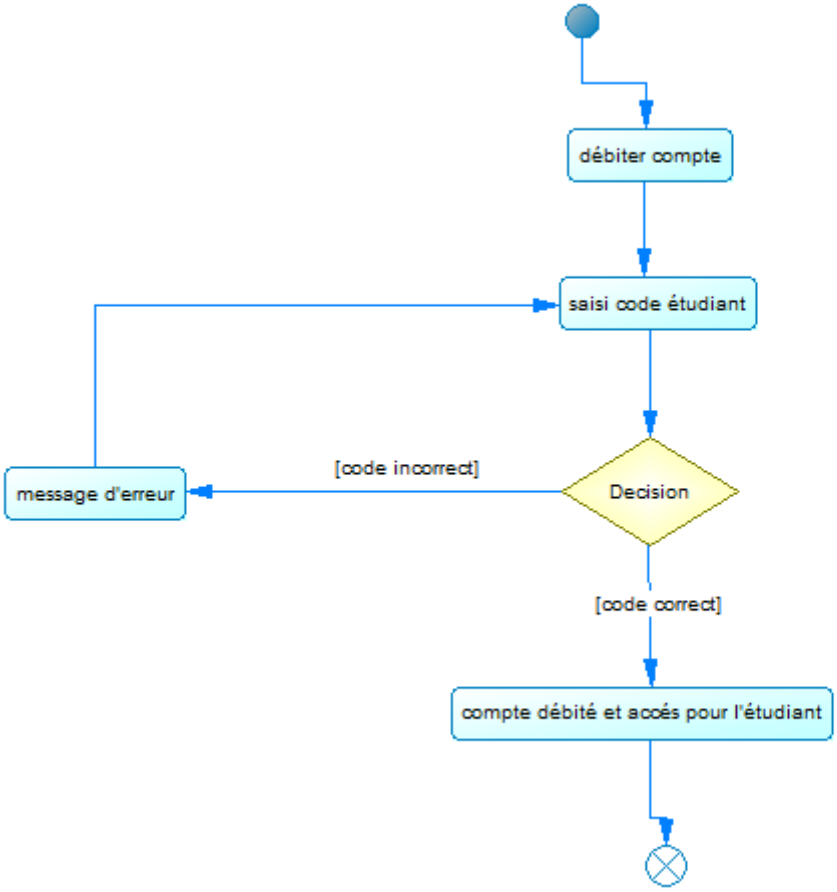

Figure 8: Diagramme d'activité du cas « vérifier et débiter »

La description du **tableau 4** et le diagramme de séquence de la **figure 9** qui suivent montre le scénario entre le portier et le système :

- le système affiche le formulaire ;
- le portier saisit les données pour vérification ;
- le système envoi les données saisies pour vérification au niveau de la BD ;
- si le numéro d'étudiant ou numéro de compte est OK ;
- le portier débite le compte et donne accès l'étudiant.

Tableau 4: Description du cas d'utilisation Vérifier et débiter

Description de cas d'utilisation « vérifier et débiter »	
Titre	Vérifier et débiter
Résumé	Permet de vérifier et de débiter le compte d'un étudiant
Acteur (s)	Portier

Pré condition	Avoir un compte d'utilisateur
Scénario nominal	✓ Le système affiche le formulaire ; ✓ Le portier saisit les données pour vérification ; ✓ Le système envoi les données saisies pour vérification au niveau de la BD.
Post condition	Le portier débite le compte et donne accès à l'étudiant
Exception	Opération échouée

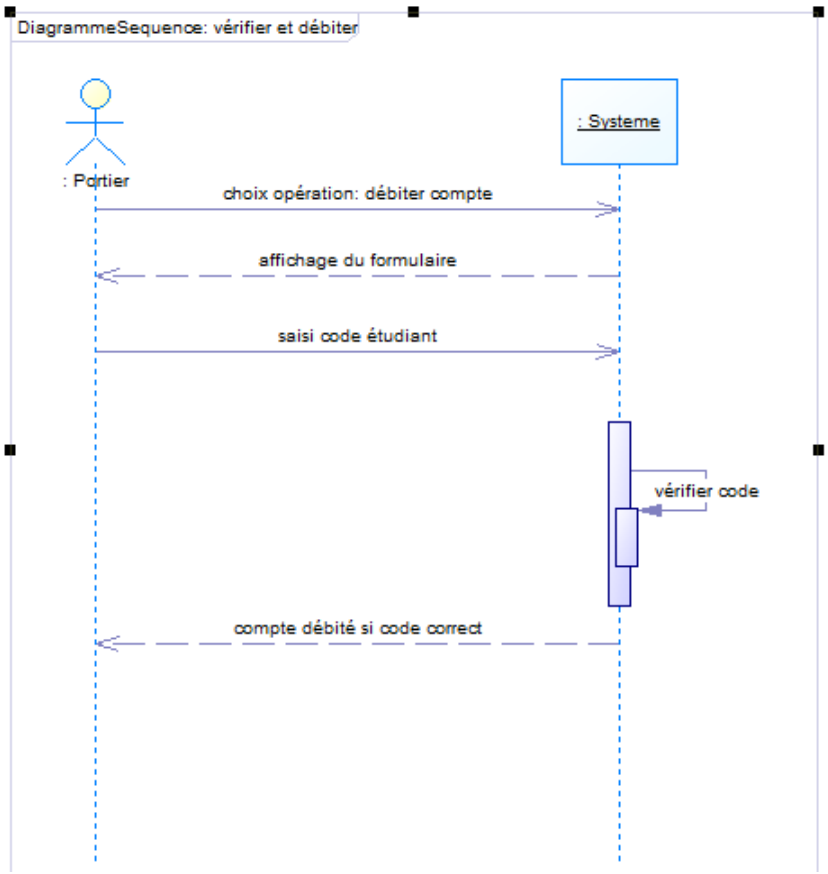

Figure 9: Diagramme de séquence du cas « vérifier et débiter »

❖ L'activité créditer compte

Dans cette activité, l'utilisateur (vendeur) choisit l'opération« **créditer compte »**. Le système lui fournit un formulaire. Il le remplit et valide. Le système procède à la vérification des données (numéro de compte). Si elles sont conformes aux règles établies alors la vente est effectuée et le compte de l'étudiant est crédité en fonction du nombre de tickets vendus, sinon l'opération échoue.

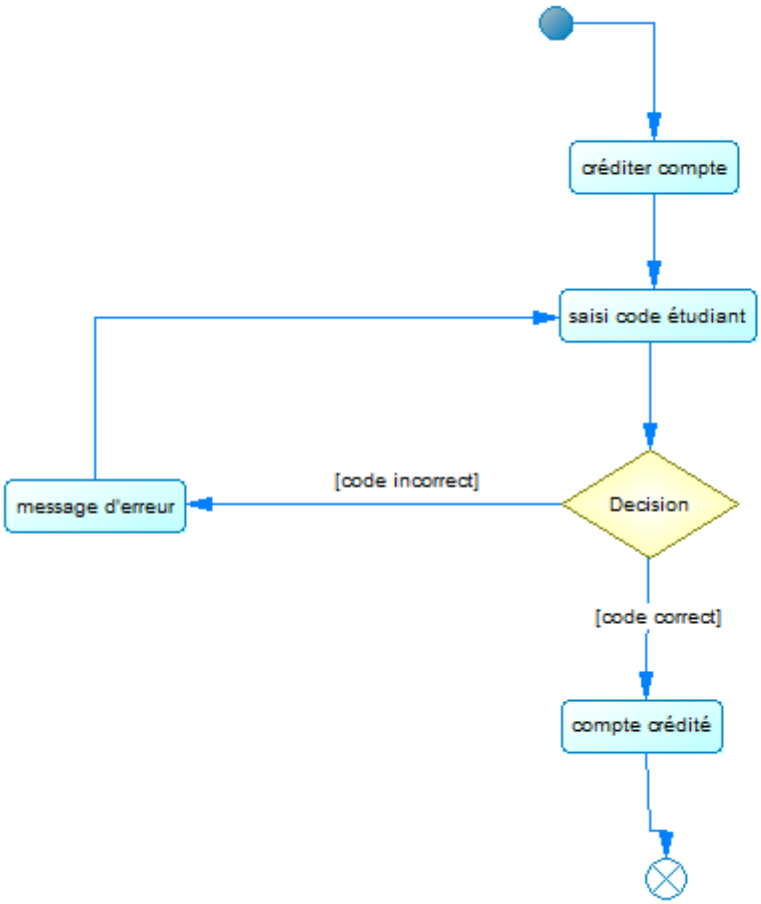

Figure 10: Diagramme d'activité du cas « créditer compte »

La description du **tableau 5** et le diagramme de séquence de la **figure 11** suivants montrent le scenario entre le vendeur et le système :

- le système affiche le formulaire ;
- le vendeur saisit les données ;
- le système envoi les données saisies pour traitement au niveau de la BD ;
- si les informations saisies sont OK ;
- la vente est effectuée et le compte de l'étudiant est crédité en tickets.

Tableau 5: Description du cas d'utilisation créditer compte

Description de cas d'utilisation « créditer compte »	
Titre	Créditer compte
Résumé	Permet de créditer le compte d'un étudiant

Acteur (s)	Vendeur, Etudiant
Pré condition	Avoir un compte d'utilisateur
Scénario nominal	✓ Le système affiche le formulaire ; ✓ Le vendeur saisit les données ; ✓ Le système envoie les données saisies pour vérification au niveau de la BD.
Post condition	Le vendeur crédite le compte de l'étudiant
Exception	Opération échouée

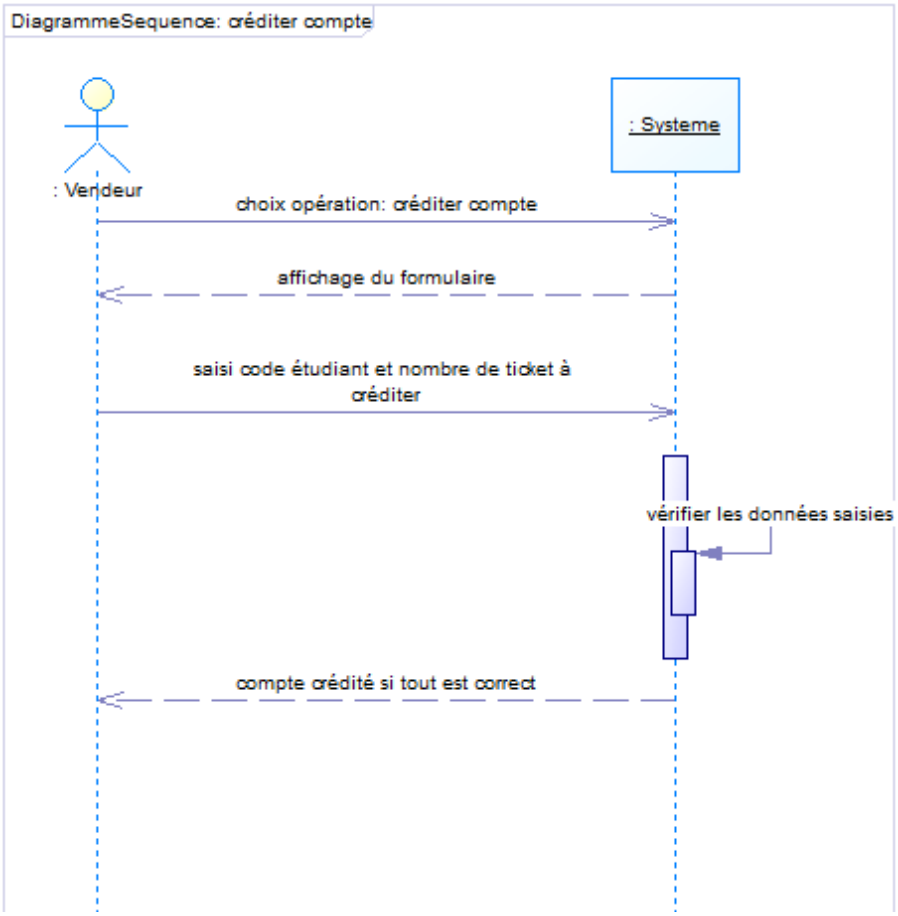

Figure 11: Diagramme de séquence du cas « créditer compte »

❖ **L'activité consulter compte**

L'utilisateur (étudiant) choisit l'opération « **consulter compte** ». Le système lui fournit un formulaire. Il entre son numéro de compte et valide. Le système procède à la vérification des données. Si elles sont conformes aux règles établies, alors le système lui affiche l'état de son compte.

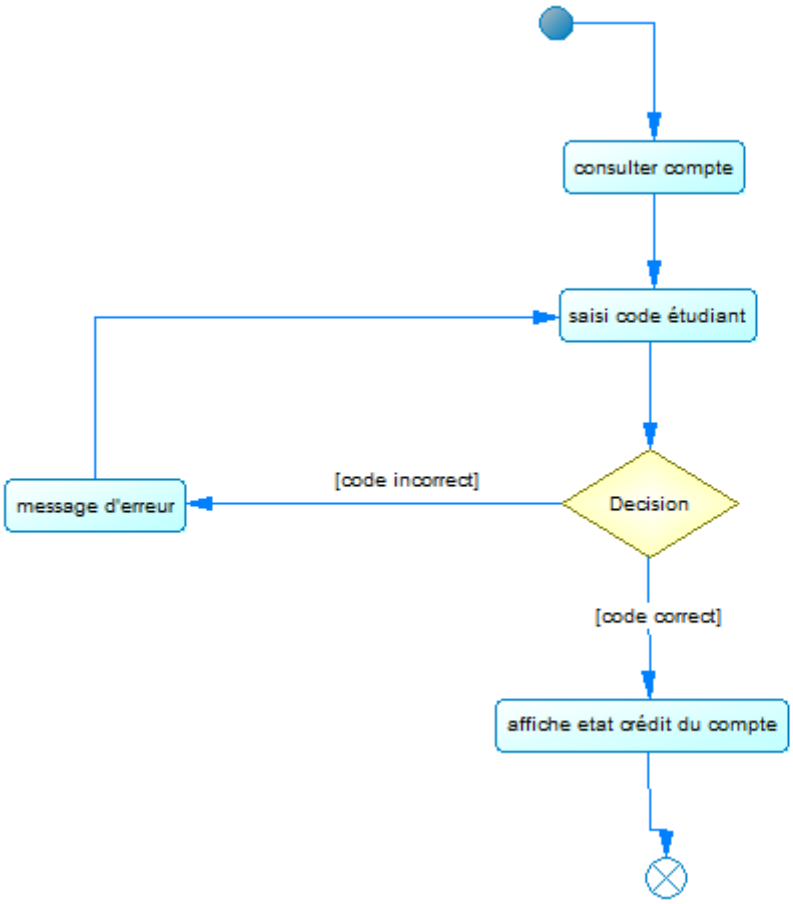

Figure 12: Diagramme d'activité du cas « consulter compte »

La description du **tableau 6** et le diagramme de séquence de la **figure 13** suivant montrent le scénario entre l'étudiant et le système :

- l'étudiant choisit l'opération « consulter compte » ;
- le système affiche le formulaire ;
- l'étudiant saisit les données (numéro de compte);
- le système envoie les données saisies pour traitement au niveau de la BD ;
- si les informations saisies sont OK ;
- le système affiche l'état des crédits de l'étudiant sur son compte.

Tableau 6: Description du cas d'utilisation Consulter compte

Description de cas d'utilisation « consulter compte »	
Titre	Consulter compte
Résumé	Permet à l'étudiant de savoir l'état de son compte
Acteur (s)	Étudiant

Pré condition	Avoir un compte d'utilisateur
Scénario nominal	✓ L'étudiant choisit l'opération « consulter compte » ; ✓ Le système affiche le formulaire ; ✓ L'étudiant saisit son numéro de compte ; ✓ Le système envoie les données saisies pour traitement au niveau de la BD.
Post condition	Le système affiche l'état des crédits de l'étudiant sur son compte
Exception	Opération échouée

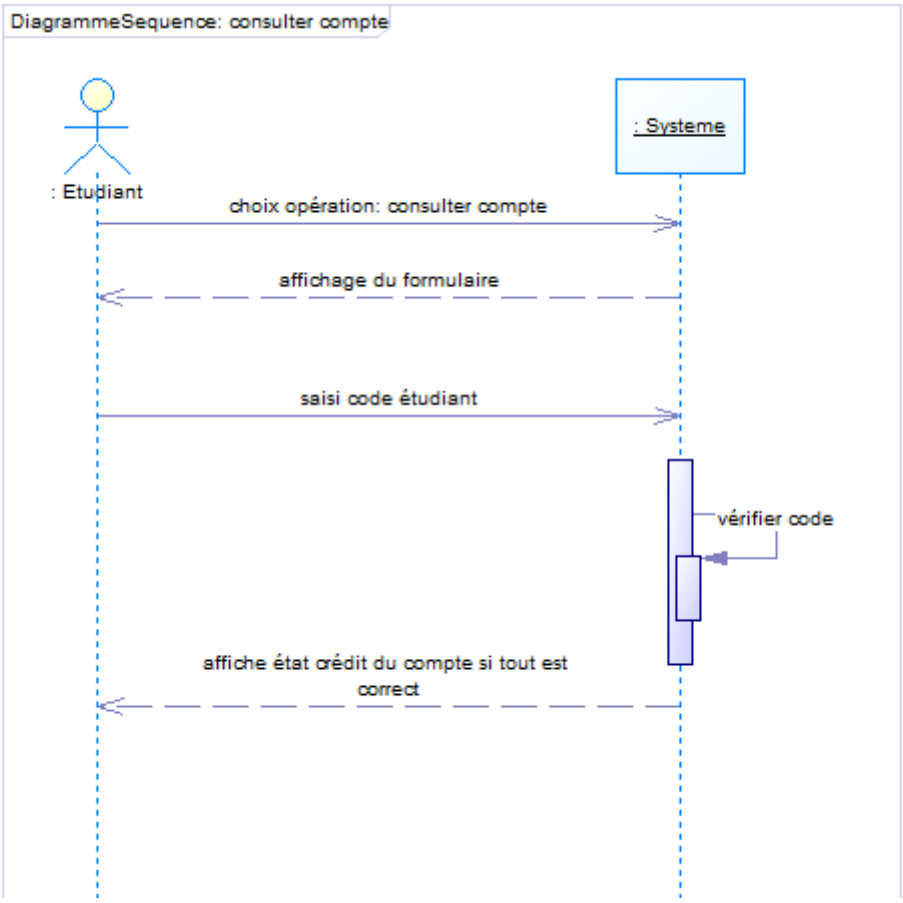

Figure 13: Diagramme de séquence du cas « consulter compte »

❖ L'activité créditer compte

L'utilisateur (étudiant) choisit une opération « **créditer compte** ». Il paye via les services en ligne (Orange Money, Joni Joni etc.). Le système lui fournit un formulaire ; il remplit et valide le formulaire. Le système procède à la vérification des données. Si elles sont conformes aux règles établies, alors l'achat est effectué sinon l'opération échoue.

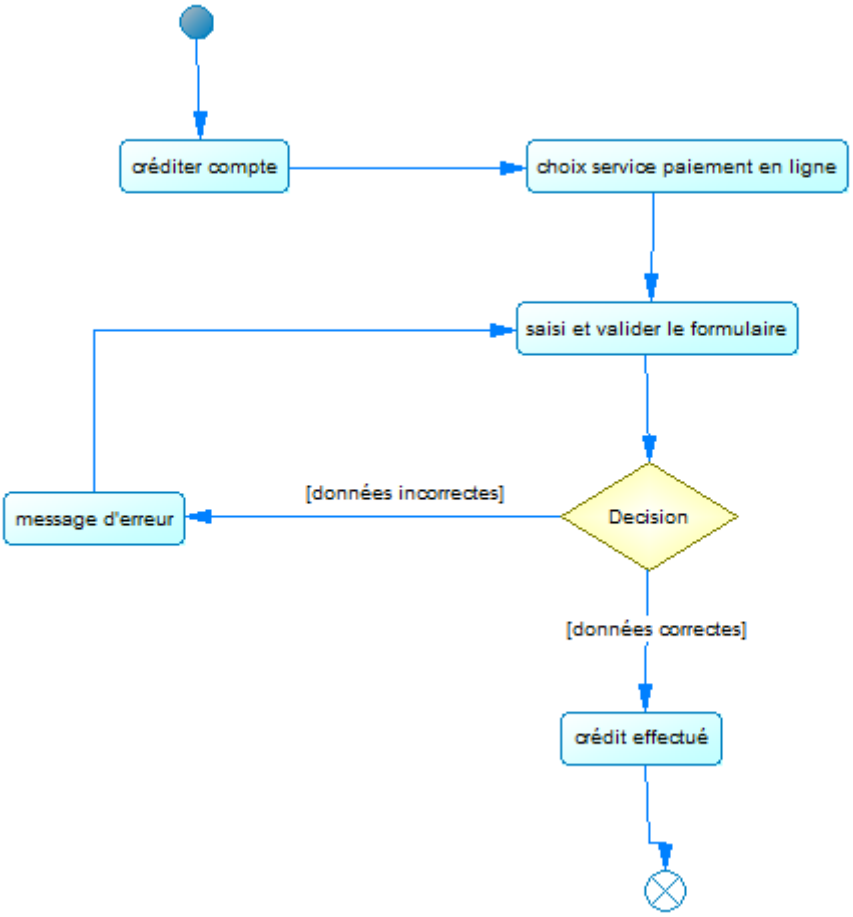

Figure 14: Diagramme d'activité du cas « créditer compte »

La description du **tableau 7** et le diagramme de séquence de la **figure 15** suivant montrent le scénario entre l'étudiant et le système :

- l'étudiant choisit l'opération « acheter tickets » ;
- le système affiche le formulaire ;
- l'étudiant saisit les données ;
- le système envoie les données saisies pour traitement au niveau de la BD ;
- si les informations saisies sont OK ;
- le système renvoie un message pour notifier que l'opération d'achat des tickets de l'étudiant est effectuée.

Tableau 7: Description du cas d'utilisation créditer compte

Description de cas d'utilisation « créditer compte »	
Titre	Créditer compte
Résumé	Permet à l'étudiant de créditer son compte via les services de

	paiement en ligne
Acteur (s)	Étudiant
Pré condition	Avoir un compte d'utilisateur
Scénario nominal	✓ L'étudiant choisit l'opération « créditer compte » ; ✓ Le système affiche le formulaire ; ✓ L'étudiant saisit son numéro de compte ; ✓ Le système envoie les données saisies pour traitement au niveau de la BD.
Post condition	Le système renvoie un message pour notifier que l'opération d'achat des tickets de l'étudiant est effectuée.
Exception	Opération échouée

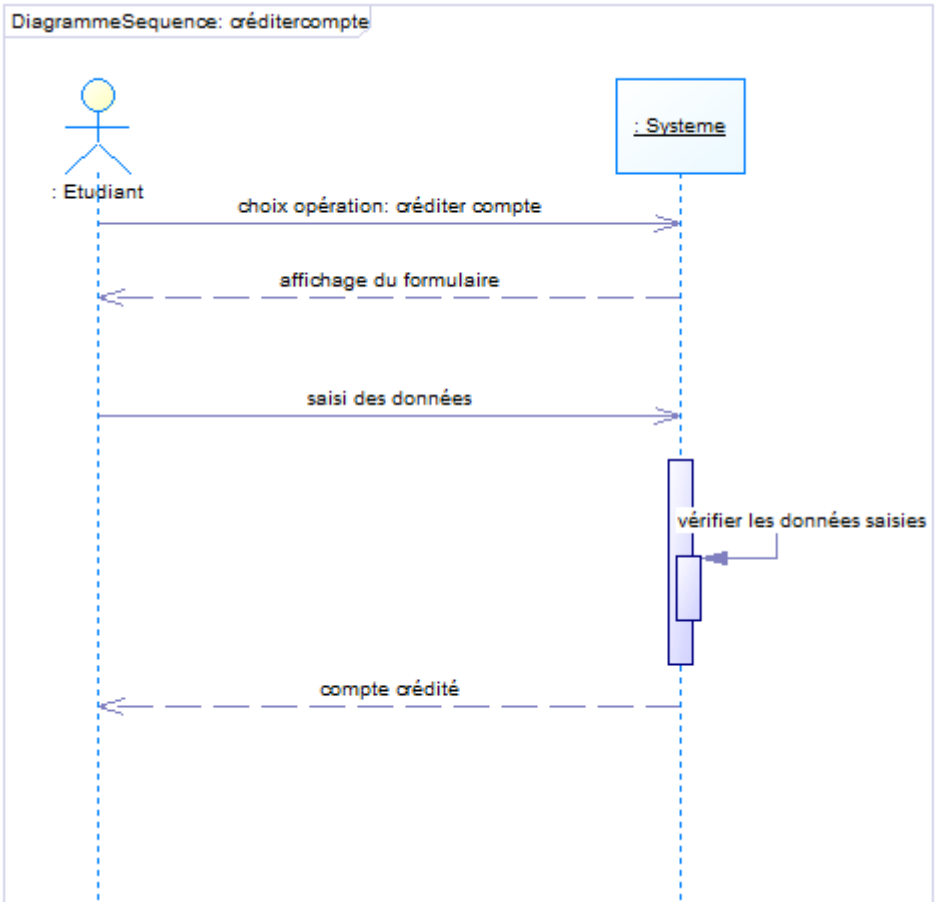

Figure 15: Diagramme de séquence du cas « créditer compte »

Ce chapitre nous a permis de faire la spécification et l'analyse des besoins fonctionnels nécessaire à la conception du projet. Ainsi nous passons au chapitre suivant qui consiste à la conception du système en suivant toujours le processus 2TUP.

Chapitre III : Conception du système

Dans ce chapitre, nous allons aborder d'abord la conception générale dans laquelle nous parlerons de l'architecture logicielle du système et des différents diagrammes, dont celui de composant, de package et de déploiement et enfin parler de la conception détaillée qui résulte de la mise en place du diagramme de classe et de l'élaboration du dictionnaire des données.

I. Conception générale

1. Architecture logicielle du système

L'architecture d'un logiciel est la structure des structures (modules) d'un système. Elle inclut:

- les composants logiciels ;
- les propriétés externes visibles de ces composants ;
- les relations entre ces composants.

C'est l'art de concevoir des espaces et de bâtir des édifices, en respectant des règles de construction empiriques ou scientifiques. Elle consiste à bien structurer un projet. Pour se faire, nous avons choisi d'utiliser le Modèle Vue Contrôleur 2 (MVC_2) qui hérite des propriétés du modèle MVC dont l'objectif global est de séparer les aspects traitements, données et présentation, mais aussi de définir les interactions entre ces trois aspects. Les données sont gérées par le modèle, la présentation par la vue, les traitements des actions et l'ensemble est coordonné par le contrôleur. Ce modèle garantit l'unicité du point d'entrée de l'application. Les composants de l'architecture sont **[3]**:

- **La vue (JSF, XHTML)**

La vue est responsable de l'interface Homme machine (IHM). Elle représente les pages web (JSP, JSF, XHTML**).** La vue est souvent implantée par un moteur de template (que l'on peut traduire par gabarit), dont les caractéristiques, avantages et inconvénients donnent lieu à de nombreux débats. D'autres types de vues existent, comme **WML** pour les dispositifs mobiles. La version JSF**2.0** utilise les **Facelets**. Les **Facelets** sont formées d'une arborescence de composants **UI** (également appelés widgets ou contrôles) **[3].**

- **Le modèle (ManagedBeans)**

Le modèle est responsable de la préservation de l'état d'une application entre deux requêtes HTTP, ainsi que des fonctionnalités qui s'appliquent à cet état. Toute donnée persistante doit être gérée par la couche modèle. Cela concerne les données de session (le panier dans un site de commerce électronique par exemple) ou les informations contenues dans la base de données (le catalogue des produits en vente pour rester dans le même exemple). Les classes Java spécialisées qui synchronisent les valeurs avec les composants UI, accèdent aux logiques métiers et gèrent la navigation entre les pages **[3]**.

- **Contrôleur (Faces servlet)**

Servlet principal de l'application qui sert de contrôleur et est déjà implémenté dans le Framework. Toutes les requêtes de l'utilisateur passent systématiquement par ce Servlet qui les examine et appelle les différentes actions correspondantes. Ce contrôleur sera déclaré dans le Web.xml et configuré dans le fichier faces-config.xml. **Faces-config.xml**

C'est un fichier de configuration de l'application définissant les règles de navigation et les différents Managed Bean utilisés **[3]**.

- **Web.xml**

Web.xml est un descripteur de déploiement qui permet de contrôler et de filtrer les pages web (JSP, JSF, XHTML) **[3]**.

La **Figure 16**ci-dessous donne un aperçu de l'architecture obtenue en nous plaçant d'emblée dans le cadre spécifique d'une application web.

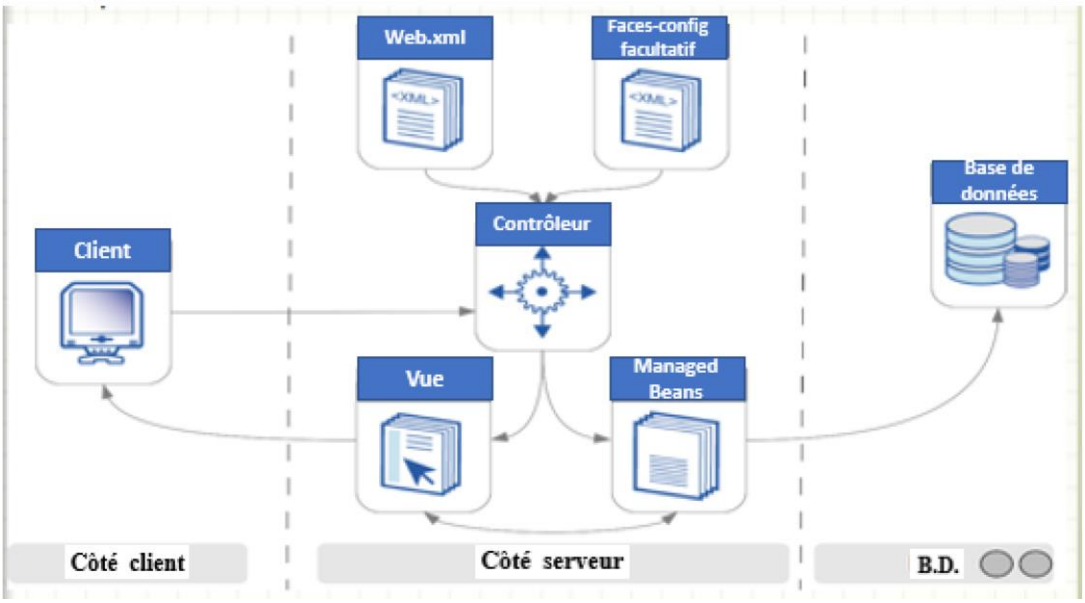

Figure 16: Architecture de l'application

2. Diagramme de composants

Ce diagramme représente l'ensemble des composants qui sont des unités autonomes représentées par un classeur structuré, stéréotypé « *component* », comportant une ou plusieurs interfaces requises ou offertes. La seule contrainte pour pouvoir substituer un composant par un autre est de respecter les interfaces requises et offertes **[4]**.

Un étudiant n'ayant pas encore activé son compte peut passer chez le vendeur, pour que ce dernier le crédite son compte.

La **figure 17** suivante représente celle proposée pour notre système:

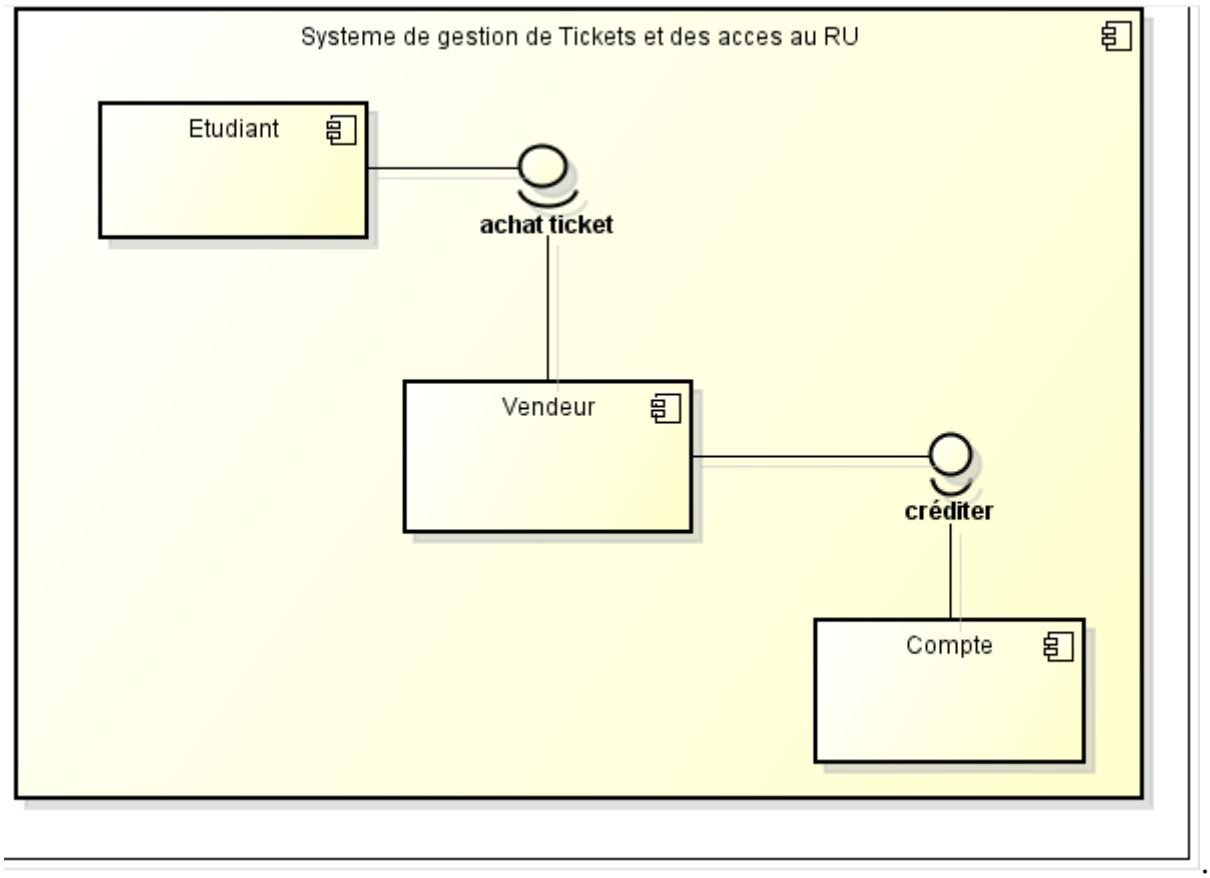

Figure 17: Diagramme de composant

3. Diagramme de paquetage

Les diagrammes de paquetages sont utilisés pour refléter l'organisation de paquetages et de leurs éléments. L'utilisation la plus courante pour les diagrammes de paquetages est d'organiser des diagrammes de cas d'utilisation et des diagrammes de classes. Bien que l'utilisation des diagrammes de paquetages ne se limite pas à ces éléments UML **[5]**. Ainsi, nous en avons illustré un pour notre système dans la **figure 18** qui suit:

Authentification : elle utilise la gestion des profils pour distinguer et vérifier les utilisateurs qui se connectent.

Gestion des users : les fonctionnalités qui se trouvent dans ce package sont « ajouter utilisateur », « rechercher utilisateur », « modifier utilisateur » et « supprimer utilisateur ».

Gestion des accès : les fonctionnalités qui se trouvent dans ce package sont « débiter compte ».

Gestion des tickets : les fonctionnalités qui se trouvent dans ce package sont « créditer compte », « activer compte » et « ajouter étudiant ».

Gestion compte : les fonctionnalités qui se trouvent dans ce package sont « activer compte », « suivi des crédits », « suivi entrées » et « acheter tickets »

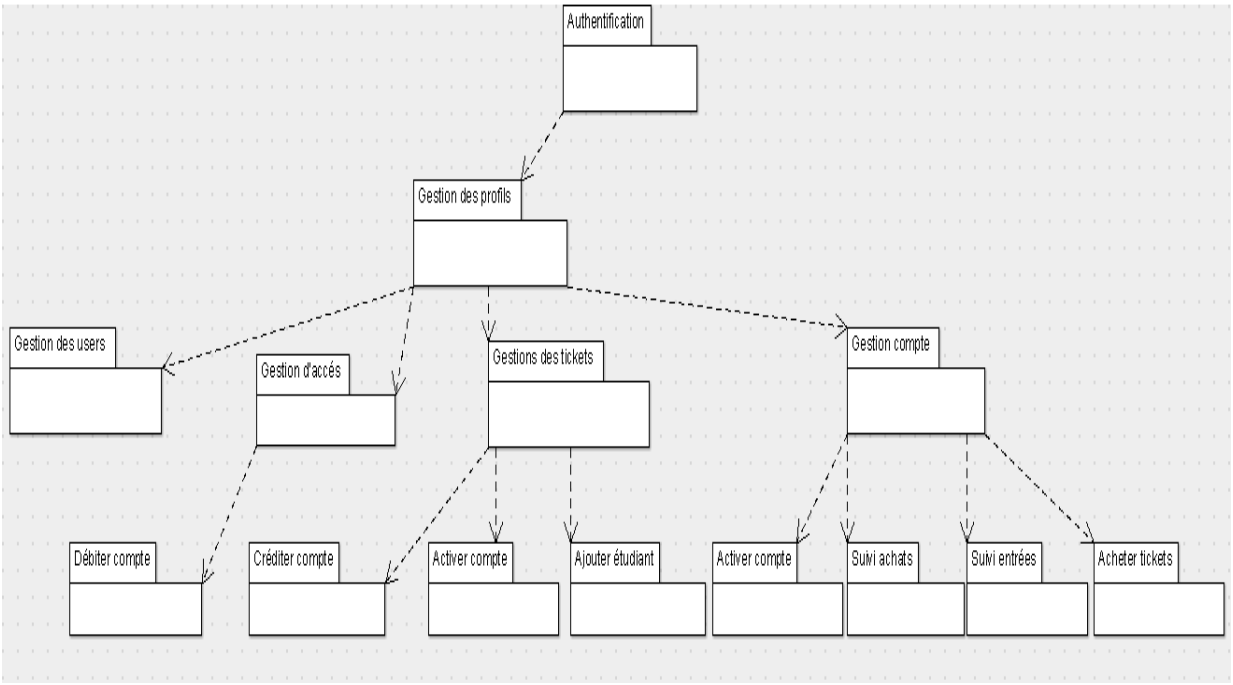

Figure 18: Diagramme de paquetage du système

4. Diagramme de déploiement

Le diagramme de déploiement fait partie des diagrammes structuraux (statique), il représente :

- la disposition physique des ressources matérielles qui constitue le système et montre la répartition des composants (élément logiciels) sur ces matériels ;
- la nature des connexions de communication entre les différentes ressources matérielles **[6].**

Le diagramme de déploiement propose une vision statique de la topologie du matériel sur lequel s'exécute le système. L'application est déployée sur des machines de 4Go de RAM et 2.00GHZde vitesse processeur pour la plupart. Le fichier «GesRestau.ear » déployé va être déposé dans le répertoire de configuration de JBOOS (C:\jbossas7.1.1.Final\standalone\deployments).Le serveur est contenu dans une machine de 8Go de RAM. La machine du Vendeur avec 4Go de RAM et 2.00GHZ de vitesse processeur utilise l'application, la machine du Potier et celle de l'administrateur avec les mêmes propriétés, l'étudiant utilise un smartphone. En plus de ça nous avons un serveur de base de données qui permet de stocker les données.

Le diagramme de déploiement de la **figure 19** suivant décrit celui qui sera utilisé pour déployer notre système :

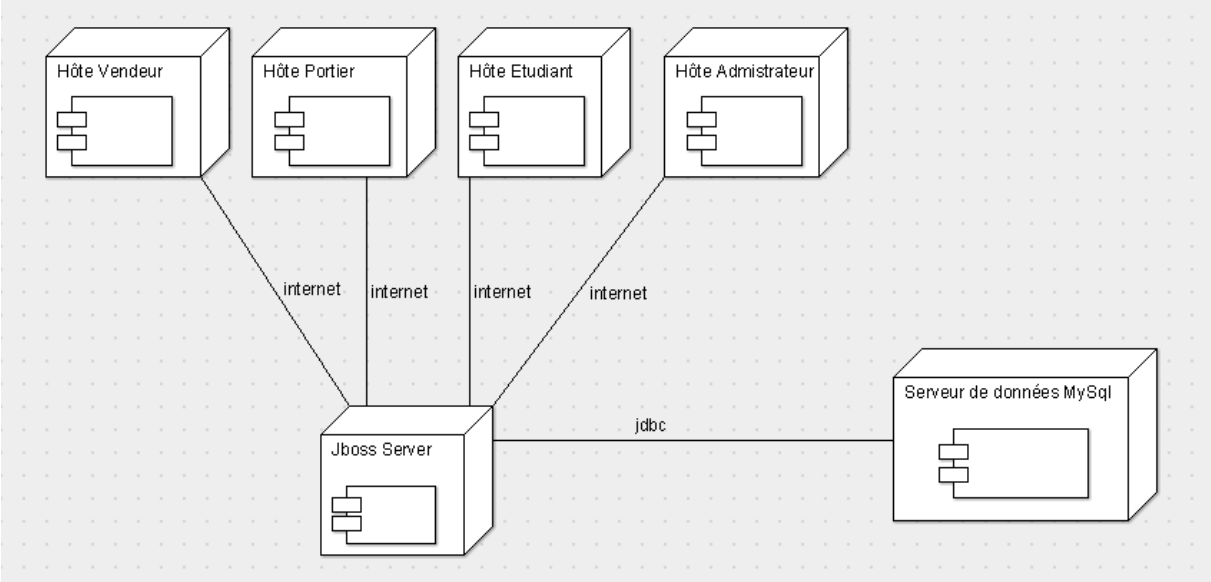

Figure 19: Diagramme de déploiement du futur système

II. Conception détaillée

1. Diagramme de classes

Une classe est une représentation abstraite d'un ensemble d'objets, elle contient les informations nécessaires à la construction de l'objet. La classe peut donc être considérée comme le modèle, le moule ou la notice qui va permettre la construction d'un objet. Le diagramme de classe est un diagramme structurel (statique) qui permet de représenter :

- les classes (attributs + méthodes) ;
- les associations (relations) entre les classes.

Le diagramme de classes est le plus important des diagrammes UML, c'est le seul qui soit obligatoire lors de la modélisation objet d'un système. Il modélise les concepts du domaine d'application, ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application.

Ainsi, les diagrammes de classes suivants représentent respectivement notre modèle opté pour ce projet.

a) Diagramme de classes participantes aux fonctionnalités du système de gestion du restaurant

Le diagramme de classes de la **figure 20** suivante décrit l'ensemble des classes participantes aux fonctionnalités du système de gestion du restaurant :

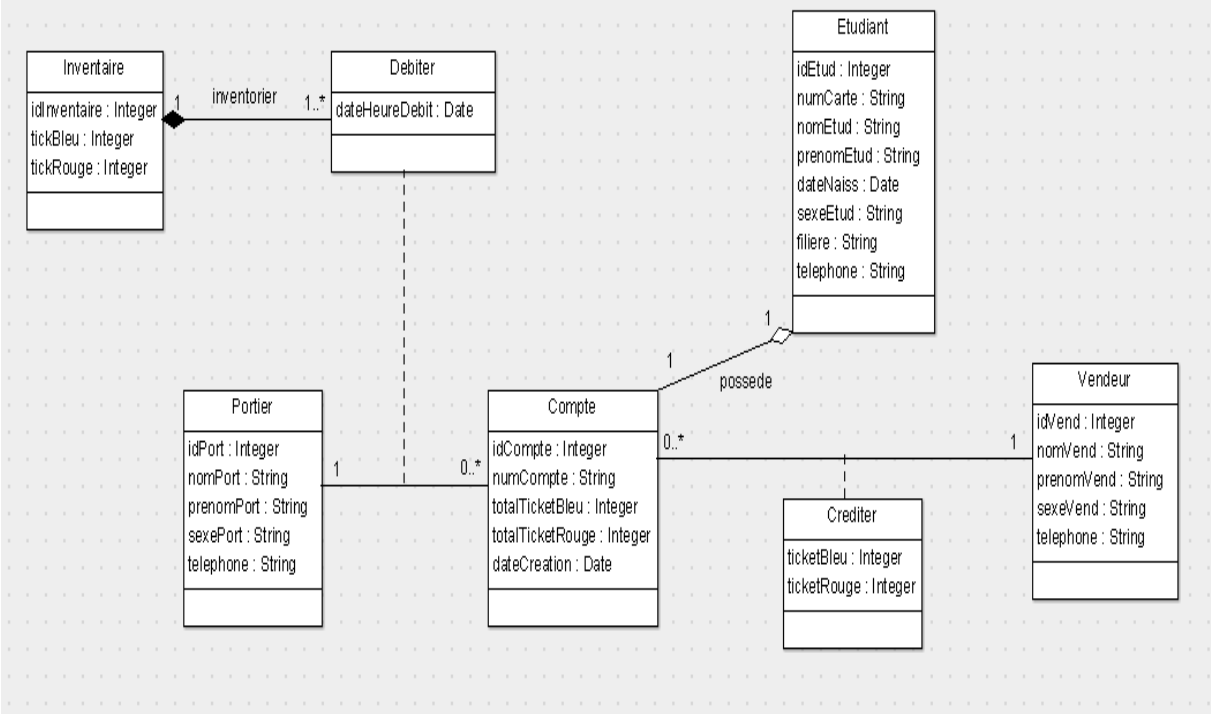

Figure 20: Diagramme de classes participantes aux fonctionnalités du système de gestion du restaurant

b) Diagramme de classes participantes à la fonctionnalité de l'authentification du système

Le diagramme de classes de la **figure 21** suivante décrit l'ensemble des classes participantes à la fonctionnalité d'authentification du système de gestion du restaurant :

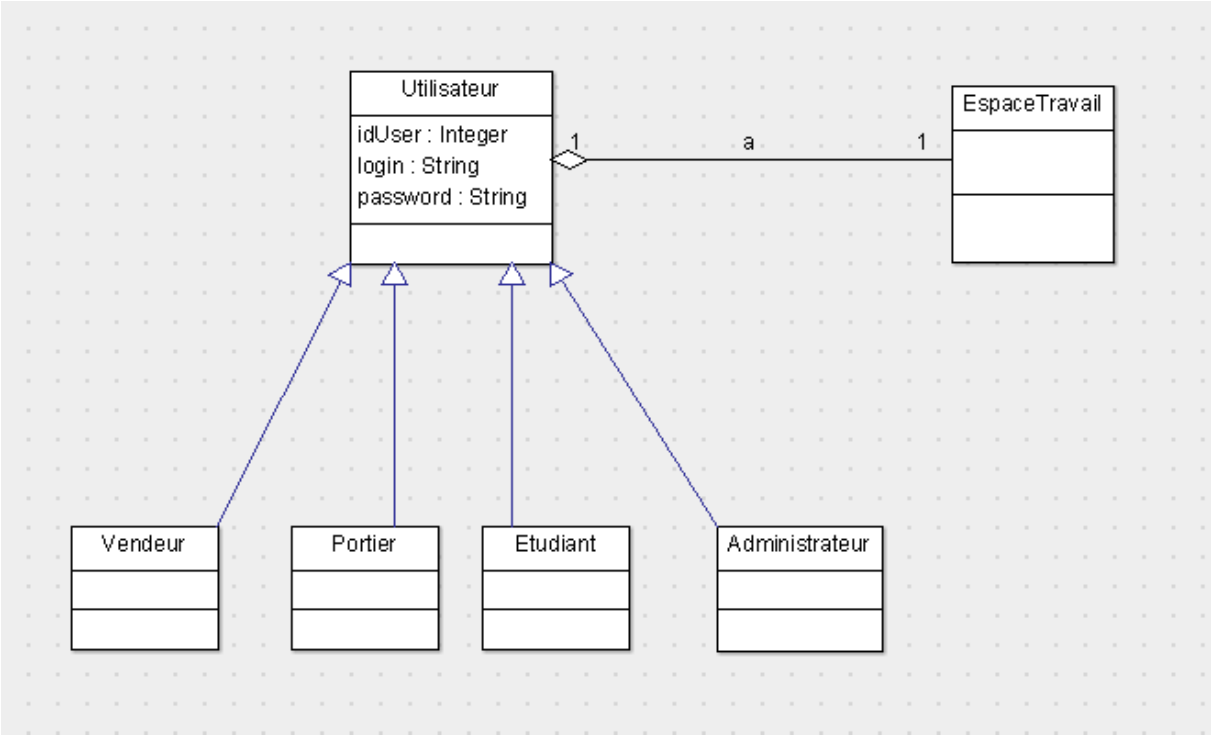

Figure 21: Diagramme de classes participantes à la fonctionnalité de l'authentification du système

2. Dictionnaire de données

Le dictionnaire de données est en fait le résultat de la phase de collecte des données. C'est comme vu précédemment la première phase à l'informatisation d'un SI (ou d'un domaine d'un SI). Cette phase est également appelée recueil d'information. En effet, pendant la phase de conception, les données recueillies et spécifiées sont inscrites dans un dictionnaire. Ce dictionnaire est un outil important, car il constitue la référence de toutes les études effectuées.

Le **tableau 8** ci-dessous représente l'ensemble des données utilisées pour élaborer ce système :

Tableau 8: Dictionnaire de données

Nom des Classes	**Nom symbolique**	**Signification ou description**	**Type de retour**
Compte	idCompte numCompte totalTickBleu	Identifiant du compte Numéro du compte Total des tickets bleu	Long String Integer

	totalTickRouge dateCreation	Total des tickets rouge Date de création du compte	Integer Date
Vendeur	idVendeur nomVendeur prenomVendeur sexeVendeur téléphone	L'identifiant du vendeur Nom du vendeur Prénom du vendeur Sexe du vendeur Téléphone du vendeur	Long String String String String
Portier	idPortier nomPortier prenomPortier sexePortier téléphone	Identifiant du portier Nom du portier Prénom du portier Sexe du portier Téléphone du portier	Long String String String String
Étudiant	idEtudiant numCarte nomEtud prenomEtud sexe dateNaiss filière téléphone	Identifiant de l'étudiant Numéro de carte de l'étudiant Nom de l'étudiant Prénom de l'étudiant Sexe de l'étudiant Date de naissance de l'étudiant Filière de l'étudiant	Long String String String String Date String String
Créditer	tickBleu tickRouge	Ticket bleu crédité Ticket rouge crédité	Integer Integer
Débiter	dateDebit heurDebit	Date où l'on a débité le compte Heure où l'on a débité le compte	Date Heure

Inventaire	tickBleu tickRouge	Inventaire des tickets bleus Inventaire des tickets rouges	Integer Integer

La conception générale nous a permis d'évoquer l'architecture du système, les différents diagrammes (composant, package et déploiement).Enfin nous avons terminé par la conception détaillée de l'application dans laquelle nous avons représenté les diagrammes de classe, afin de bien préparer la prochaine étape qui est l'implémentation et la présentation de l'application.

Chapitre IV : Implémentation et présentation du système

Ce chapitre couvre la création et la mise en œuvre des différents programmes, interfaces qui servent à la constitution de notre application et de ses fonctionnalités. Nous décrivons l'environnement de création du système, ensuite nous présenterons quelques interfaces résultantes.

I. Implémentation du système

Dans cette partie, nous nous intéresserons au modèle logique de données et à la présentation des outils utilisés pour assurer la réalisation de l'application.

1. La plateforme JEE

La technologie JEE permet de réaliser des applications d'entreprises:

- distribuées (composants EJBs stockés sur plusieurs serveurs) ;
- multi-utilisateurs ;
- pouvant se connecter à plusieurs bases de données (BDs réparties).

Les applications peuvent tourner sur un Intranet et/ou sur l'Internet. JEE permet de faire de la programmation de haut niveau (logique applicative ou métier). On ne se soucie pas donc des spécifications de bas niveau comme les BDs, la gestion des transactions, la sécurité…

Cette technologie JEE propose deux types de composants principalement:

- les composants WEB: Servlets, JSP (Java Server Pages), JSF (Java Server Faces) ;
- les composants EJBs (Enterprise Java Beans).

Les EJBs sont un type de composants implémenté coté serveur pour gérer la logique métier ou logique applicative **[7]**.

L'architecture JEE est composée principalement de deux composants:

- logique métier (Business Logic): implémentée par les EJB rassemblés sous forme de fichiers.jar et déployés dans un serveur d'application. Le déploiement se fait sous forme de fichiers EAR (Enterprise ARchives) ;

- logique de présentation (Présentation Logic): implémentée grâce aux JSP, Servlets, HTML, fichiers images, etc., pour interagir avec le client. Cette partie peut être rassemblée en archive web (.WAR) **[7]**.

La **figure 22** ci-dessous représente l'architecture logicielle de cette plateforme :

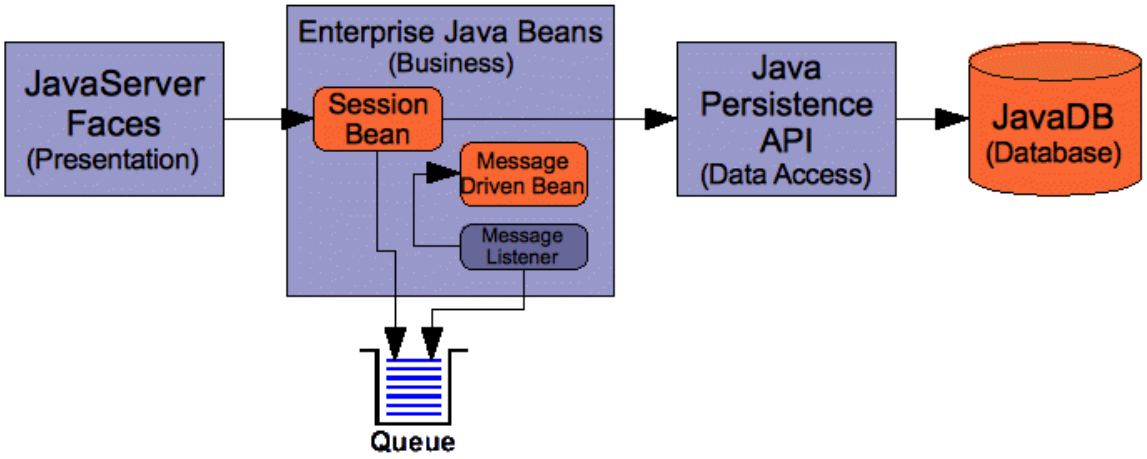

Figure 22: Architecture de la plateforme JEE [8]

2. Le modèle logique de données (MLD)

Le MLD ou Modèle Logique des Données est simplement la représentation textuelle du Modèle Physique des Données(MPD). Il s'agit juste de la représentation en ligne du schéma représentant la structure de la base de données.

À noter que le MLD prend parfois un R et devient MLDR ; le R signifie simplement Relationnel.

La **figure 23** suivante représente notre MLD pour ce projet ainsi :

- PrK sont les clés primaires ;
- FrK sont les clés étrangères.

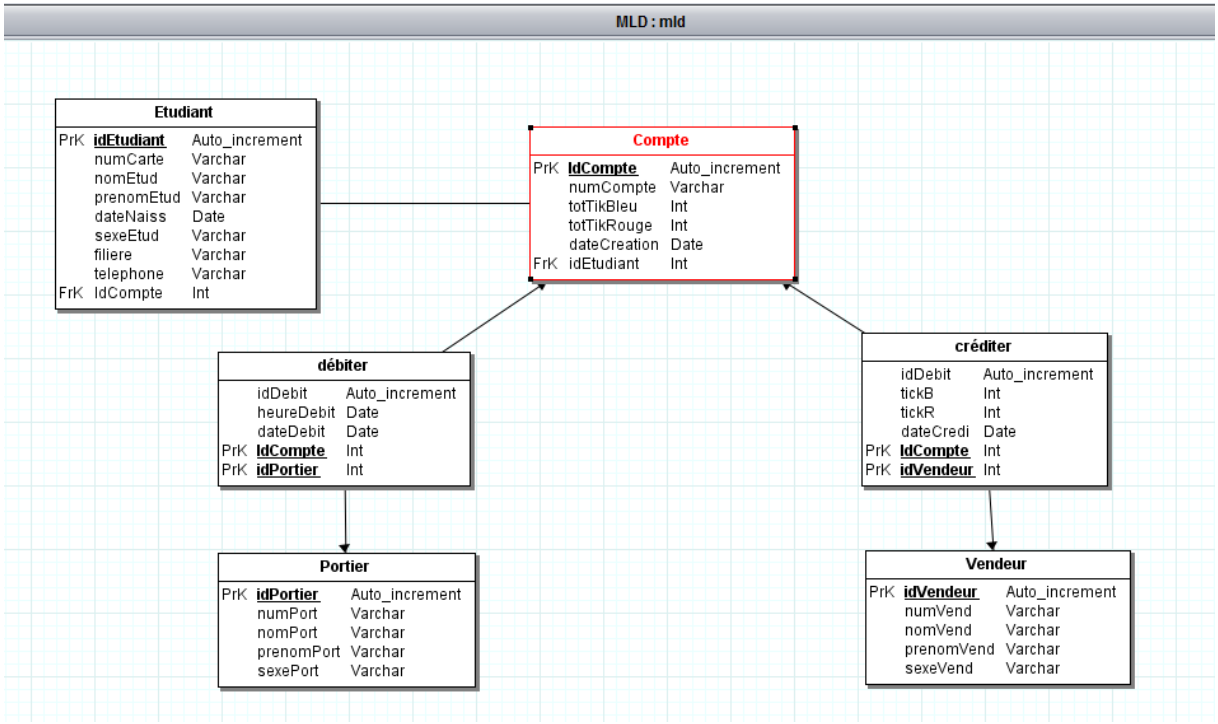

Figure 23: Modèle logique des données

3. Création de la base de données

La base de données est créée au niveau de Wampserver. Dans cette partie nous allons créer la base de données dénommée « db_restau ». Ce nom de la base de données est mentionné lors de la configuration du fichier « standalone » de JBoss. Le fichier jar exécutable de mysql_connector doit être collé dans le répertoire (C:\jboss\jboss-as-7.1.1.FinalV1\jboss-as-7.1.1.Final\modules\com\mysql\main) de configuration de JBoss.

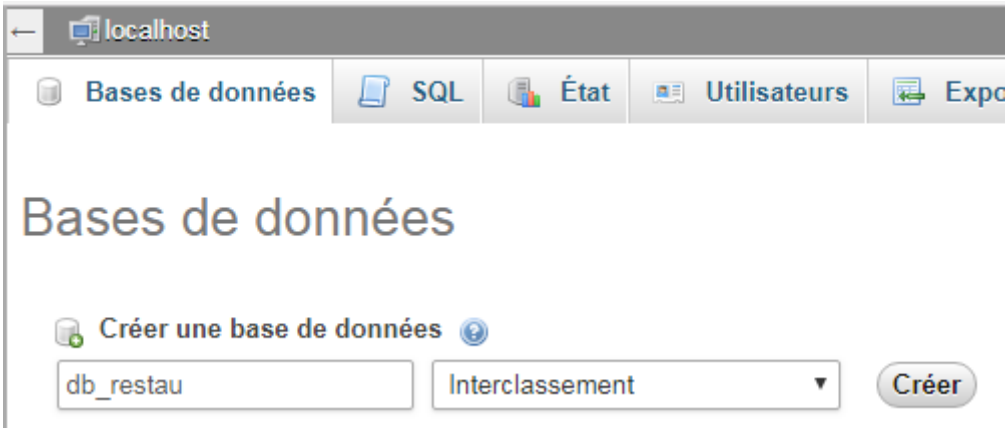

Figure 24: Création de la base de données

Par la suite toutes les tables seront créées automatiquement après démarrage du serveur d'application JBoss et implémentation de ces dernières sous éclipse. La **figure 25** suivante illustre cela :

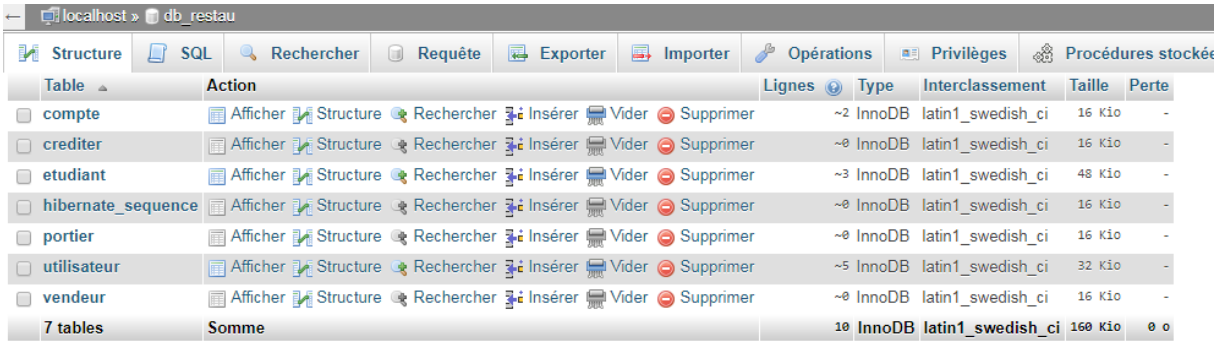

Figure 25: Les tables de la base de données

4. Les outils de développements

Pour développer notre application, nous avons utilisé plusieurs outils et nous allons faire leurs présentations dans cette partie.

a) WampServer

WampServer est une plateforme de développement Web. Il permet de développer des applications Web dynamiques à l'aide du serveur Apache2, du langage de scripts PHP et d'une base de données MySQL. Il possède également PhpMyAdmin pour gérer plus facilement les bases de données. WampServer s'installe facilement et son utilisation très intuitive permet de le configurer très rapidement (sans toucher aux fichiers de configuration). Contrairement aux autres solutions, WampServer permet de reproduire fidèlement notre serveur de production.

b) PhpMyAdmin

PHPmyadmin est une application web écrite en PHP, destinée à l'administration de MySQL. Elle nous permet entre autres de gérer nos bases de données, de créer des tables, champs, relation etc. Nous verrons plus bas la liste entière des tâches qui peuvent être accomplies avec cet outil. Réputé pour être sûr, encadré par une équipe extrêmement active et doté d'une interface conviviale, **PhpMyAdmin** s'est imposé comme référence dans la sphère des outils d'administration de serveur MySQL sur le web. Ce qui lui a valu le mérite d'être intégré dans nombreuses solutions de contrôles hébergement et de serveur WEB (PHP/MySQL) comme **WampServer**, XAMP, EasyPHP, CPanel…etc. Chaque version de Apache, MySQL et PHP dispose de sa propre configuration et de ses propres fichiers.

Les fonctionnalités de WampServer sont très complètes et très intuitives, nous ne détaillerons donc pas ici leur utilisation.

- Via un click gauche sur l'icône de WampServer, vous pouvez notamment:
  - gérer les services d'Apache et MySQL ;
  - passer en mode online/offline (accessible à tous ou limité à localhost) ;
  - installer et changer de version d'Apache, MySQL et PHP ;
  - gérer les paramètres de configuration de vos serveurs ;
  - accéder à vos logs ;
  - accéder aux fichiers de configuration ;
  - créer des alias.
- Via un clic droit :
  - changer la langue du menu de WampServer ;
  - accéder directement à cette page.

c) Eclipse JEE

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plateforme modulaire pour permettre de réaliser des développements informatiques **[9]**.

I.B.M. est à l'origine du développement d'Eclipse qui est d'ailleurs toujours le cœur de son outil Websphere Studio Workbench (WSW), lui-même à la base de la famille des derniers outils de développement en Java d'I.B.M. Tout le code d'Eclipse a été donné à la communauté par I.B.M afin de poursuivre son développement **[9]**.

Eclipse utilise énormément le concept de modules nommés "plugins" dans son architecture. D'ailleurs, hormis le noyau de la plateforme nommé "Runtime", tout le reste de la plate−forme est développé sous la forme de plugins. Ce concept permet de fournir un mécanisme pour l'extension de la plateforme, et, ainsi fournir la possibilité à des tiers de développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse **[9]**.

Les principaux modules fournis en standard avec Eclipse concernent Java, mais des modules sont en cours de développement pour d'autres langages notamment C++, Cobol, mais aussi pour d'autres aspects du développement (base de données, conception avec UML, ...). Ils sont tous développés en Java, soit par le projet Eclipse, soit par des tiers commerciaux ou en open source. Les modules agissent sur des fichiers qui sont inclus dans l'espace de travail

(Workspace). L'espace de travail regroupe les projets qui contiennent une arborescence de fichiers **[9].**

d) Serveur de bases de données : MySQL

Une base de données est un ensemble organisé d'informations mémorisées sur un support informatique. Elle permet aux utilisateurs d'accéder aux données et de faire des mises à jour. Celles-ci dépendent des privilèges qui leur sont octroyés. Cette base de données est locale. Ainsi, pour pouvoir gérer une base de données, il est nécessaire d'avoir un SGBD (Système de gestion des bases de données). Un SGBD est un logiciel qui permet de créer et de gérer une base de données. Il facilite le traitement et la manipulation des données grâce à des requêtes. Ces dernières sont faites via des langages reconnus par les SGBD. Le plus populaire de ces langages est SQL. Databasic, dataflex, dBase ou xBaseScript sont aussi des langages de manipulation de SGBD. Les principaux SGBD sont : Microsoft Access, Microsoft SQL server, Oracle, MySQL, PostgreSQL,Teradata, Berkeley BD, Interbase, etc. Il existe plusieurs types de SGBD. Pour notre projet, nous avons choisi **MySQL** comme serveur de base de données.

e) Serveur d'application : Jboss

Plusieurs serveurs d'application sont aujourd'hui utilisés. Chaque serveur présente des caractéristiques qui lui sont propres. Le choix d'un tel ou tel autre serveur réside dans le type de projet que l'on souhaite réaliser, aux moyens dont on dispose et des avantages qu'il fournit. Après avoir fait des recherches sur plusieurs serveurs d'application, le choix est porté sur Jboss. Jboss est un serveur d'application Java EE. Ce serveur est écrit en Java et distribué sous licence GPL. Il existe sous plusieurs versions. Celui utilisé dans cette application est **Jboss as 7**.

f) Client web

Le langage utilisé est le Java. Pour être plus précis, il s'agit du Java EE. Ce terme signifie Java Entreprise Edition anciennement appelé J2EE. Il fait référence à une extension de la plate-forme standard. Autrement dit, la plate-forme Java EE est construite sur le langage Java et la plate-forme Java SE. Elle y ajoute un grand nombre de bibliothèques remplissant tout un tas 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'application. Java EE permet au programmeur de faire du développement normalisé de composants modulaires réutilisables.

g) Framework JSF

La technologie JSF représente un Framework basé sur les composants utilisateurs utilisés pour développer des applications web. Il permet de représenter les composants, de gérer leur état et leur comportement. Elle facilite l'écriture d'interfaces utilisateurs en fournissant une bibliothèque de contrôle **[10]:**

- simples : zones de saisie classiques, boutons, liens hypertextes ;
- complexes : tableaux de données.

Cette technologie permet principalement de :

- faciliter le développement des pages web grâce à un Framework basé sur les composants ;
- représenter les composants UI et gérer leur état ;
- faire la gestion des évènements ;
- faire la validation côté serveur ;
- faire la conversion de données, etc.

Il existe plusieurs implémentations pour JSF :

- Myfaces ;
- RichFaces,etc.

Pour intégrer JSF dans un projet JEE, la première étape consiste à faire (lorsqu'on veut utiliser les ServerFaces) le mapping avec l'instance de FacesServlet dans le fichier de déploiement web.xml **[10]**.

```
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
<context-param>
```

Figure 26: Le fichier Web.xml

Ainsi, toutes les pages contenants le mapping/faces/ seront interprétées par le serveur comme contenant des composants JSF **[10]**.

Apres la déclaration de la Servlet JSF, on définit le chemin du fichier de configuration JSF « faces-config.xml ». Le nom et l'emplacement du fichier de configuration de JSF faces-config.xml sont spécifiés par un paramètre de contexte dans web.xml. Le chemin vers ce fichier est un chemin relatif à partir de la racine de l'application **[10]**.

La création d'une page JSF se résume à :

- ajouter les librairies contenant les Taglibs JSF ;

```
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
```

Figure 27: Librairie des Taglibs JSF

- l'intégration des composants et définir à quel Bean chaque composant est relié.

h) ArgoUML et PowerAMC

Pour concevoir une application, nous avons besoin d'outils de modélisation UML. Pour notre cas, nous avons choisi ArgoUML et PowerAMC pour faire notre conception.

- ArgoUML : ce logiciel est un outil de modélisation UML qui est Open Source. Il a été créé en 1999 par Toby Baier pour Tigris en partenariat avec l'université de Californie. La dernière version date du 13 octobre 2000, et c'est la version v0.8.1a. C'est un logiciel entièrement écrit en Java et qui est basé sur UML 1.3. Il nécessite l'installation de JDK1.2 ou plus. Son installation se fait facilement en mode graphique.

- PowerAMC : est l'un des premiers outils qui permet d'élaborer des modèles de données que cela soit MERISE, UML ou autre, de manière graphique et de les implémenter quel que soit le SGBD et ce de manière automatique. De même, l'outil permet de modéliser les processus métiers. Le lien entre la modélisation des données et la modélisation des processus peut être effectué, offrant ainsi aux entreprises qui possèdent PowerAMC / AMC Designor l'opportunité de mettre en œuvre un référentiel unique des développements et des processus que ceux-ci soient informatisés ou non. Aussi PowerAMC est une force dans tout nouveau projet d'entreprise car il permet d'identifier avec précision quels processus, quelles personnes et/ou quelles données seront impactés. L'estimation et maitrise des coûts en est grandie.

5. Le codage de la couche métier EJB

Dans cette partie, nous allons créer un projet EJB. Comme tout projet EJB, nous aurons à créer les interfaces (**Locale** et **Remote**), **implémenter** ces deux **interfaces** et créer les **entités** (***classes persistantes***).

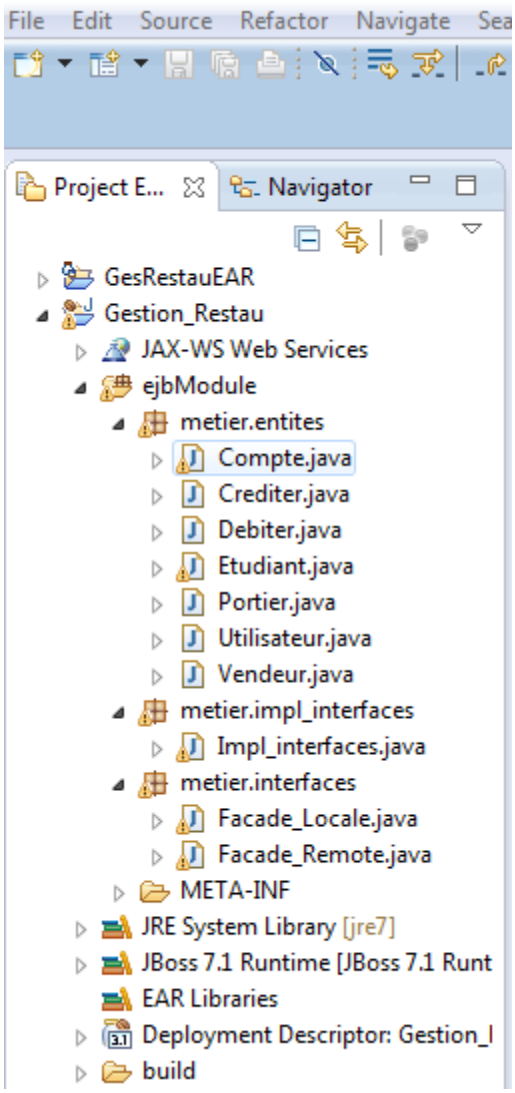

Figure 28: Structure de la couche métier EJB

6. Le codage des entités

Avec **JPA** (**Java Persistence API**), nous n'avons plus besoin d'être des spécialistes en base de données ou d'un SGBD quelconque. **JPA** nous permet de programmer les classes (entités) qui seront transformées en tables dans la base de données correspondant. JPA permet aux programmeurs d'utiliser des notions avancées en base de données tout en restant programmeurs.

@Entity signifie que cette classe est une table. JPA créera une table, de même nom que la classe, dans notre base de données.

@Id signifie que l'attribut qui est en dessous (idClasse) est la clé primaire de la table classe.

@GeneratedValue signifie que la clé primaire est automatiquement générée par le **SGBD**.

@OneToMany signifie que cette classe (table) est liée à plusieurs Utilisateurs (1 à plusieurs) **[11].**

Voici une présentation de quelques classes ou entités dans les **figures 29 et 30** suivantes :

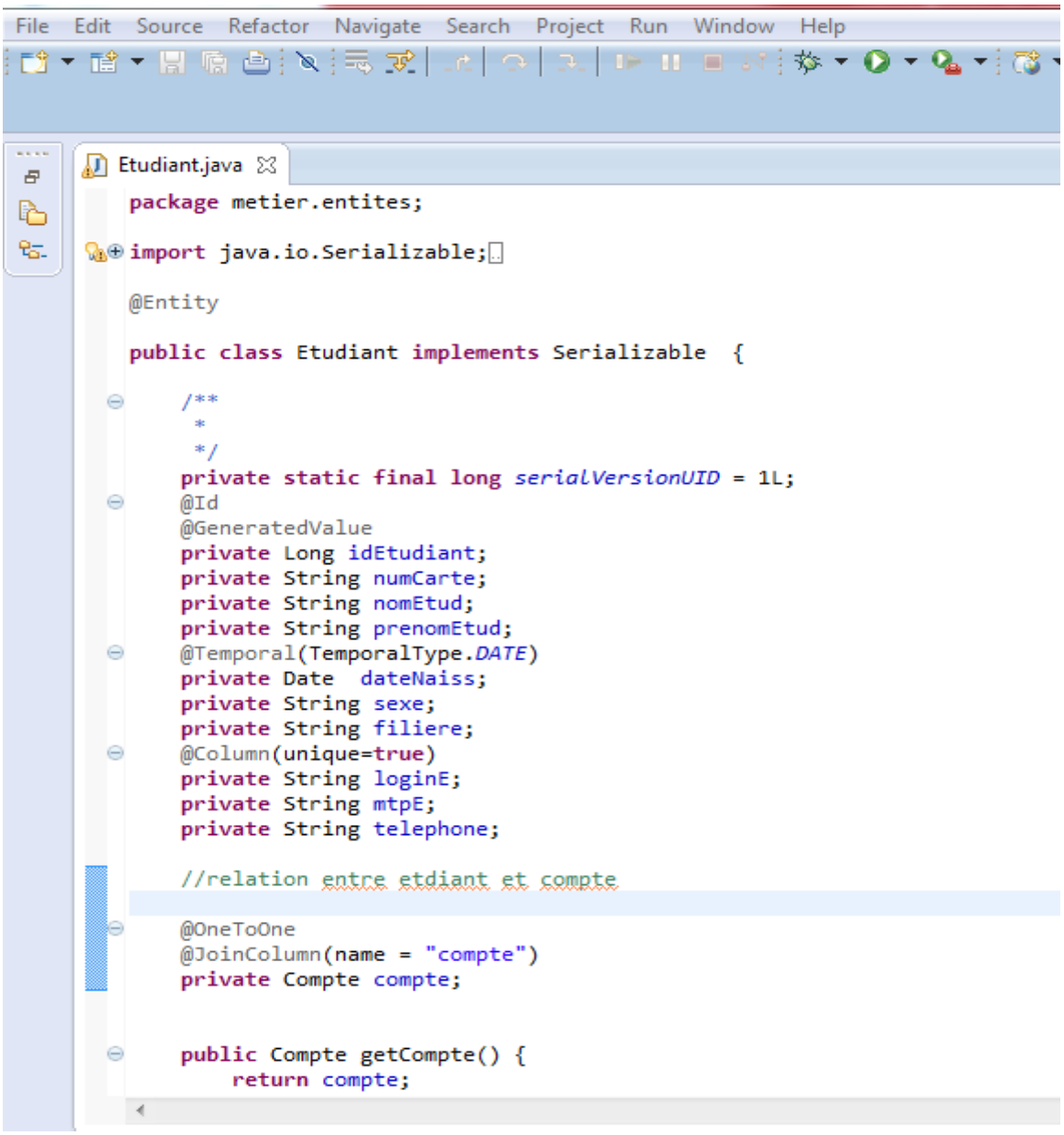

Figure 29: Entité étudiant

Etudiant.java *Compte.java

```java
package metier.entites;

import java.io.Serializable;

@Entity
public class Compte implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue
    private int idCompte;
    private String numCompte;
    private int totalTickBleu;
    private int totalTickRouge;
    @Temporal(TemporalType.DATE)
    private Date dateCreation;

    @Transient
    private String numCarte;

    // ############################## relation entre compte et etudiant ################################

    @OneToOne(cascade=CascadeType.ALL, fetch=FetchType.LAZY)
    @JoinColumn(name="etudiant" ,nullable=false ,referencedColumnName="idEtudiant")
    private Etudiant etudiant;

    public Etudiant getEtudiant() {
        return etudiant;
    }


    public void setEtudiant(Etudiant etudiant) {
        this.etudiant = etudiant;
```

Figure 30: Entité Compte

7. Le codage des interfaces

Dans cette partie, nous présenterons les différentes interfaces utilisées :

- L'interface locale

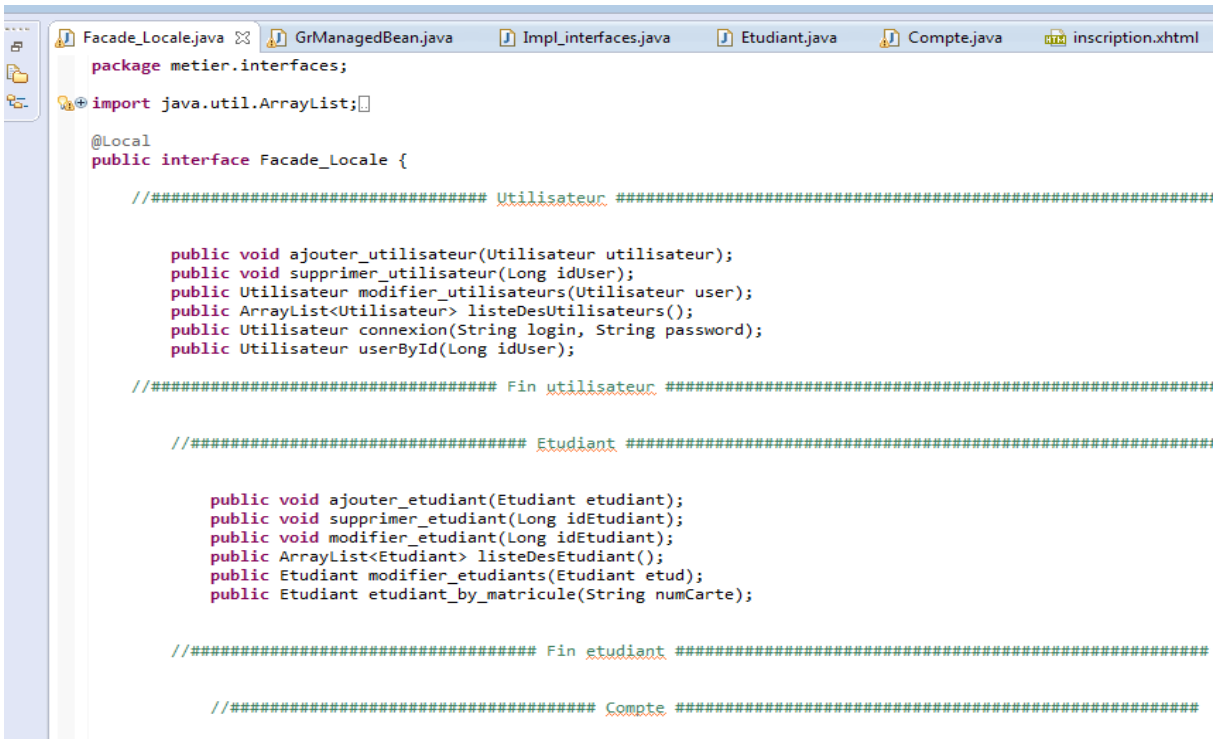

Figure 31: L'interface locale

- L'interface distante remote

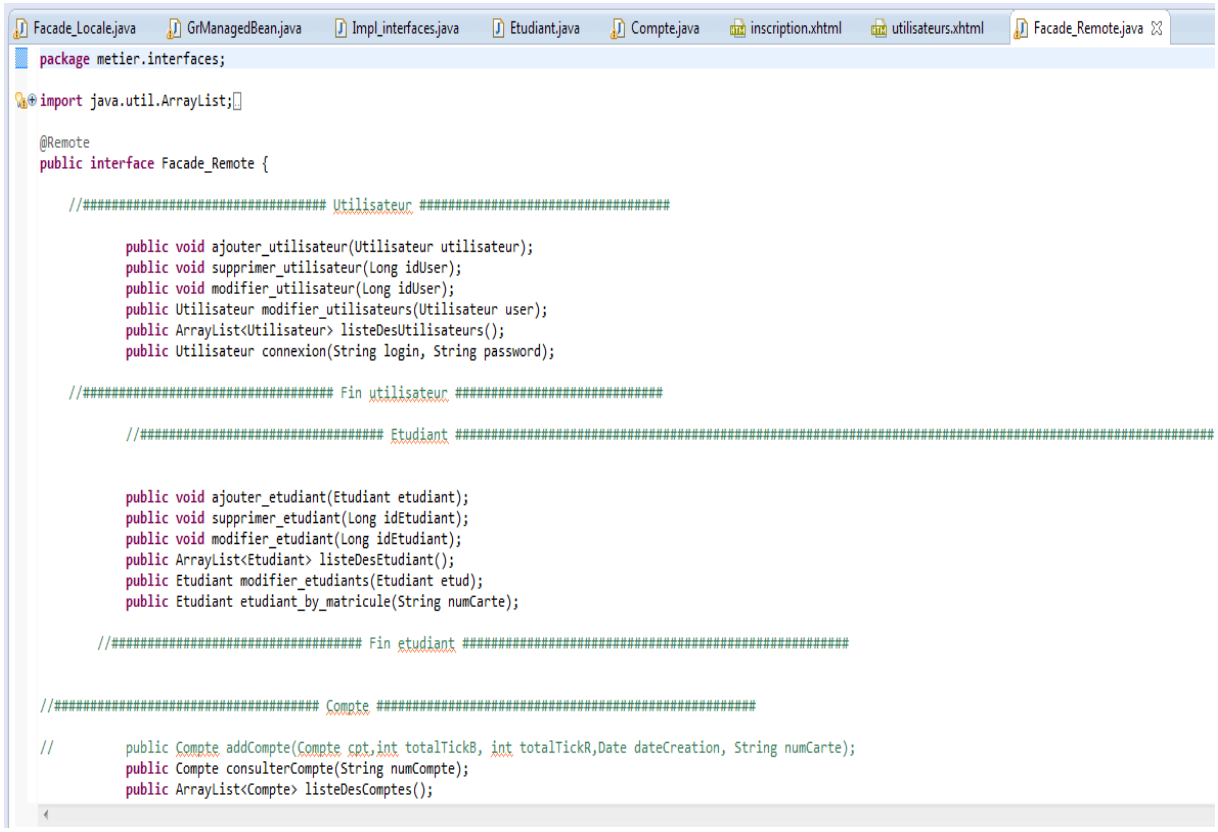

Figure 32: L'interface remote

8. Implémentation des interfaces

Cette **figure 33** illustre l'implémentation des interfaces locale et remote de notre application :

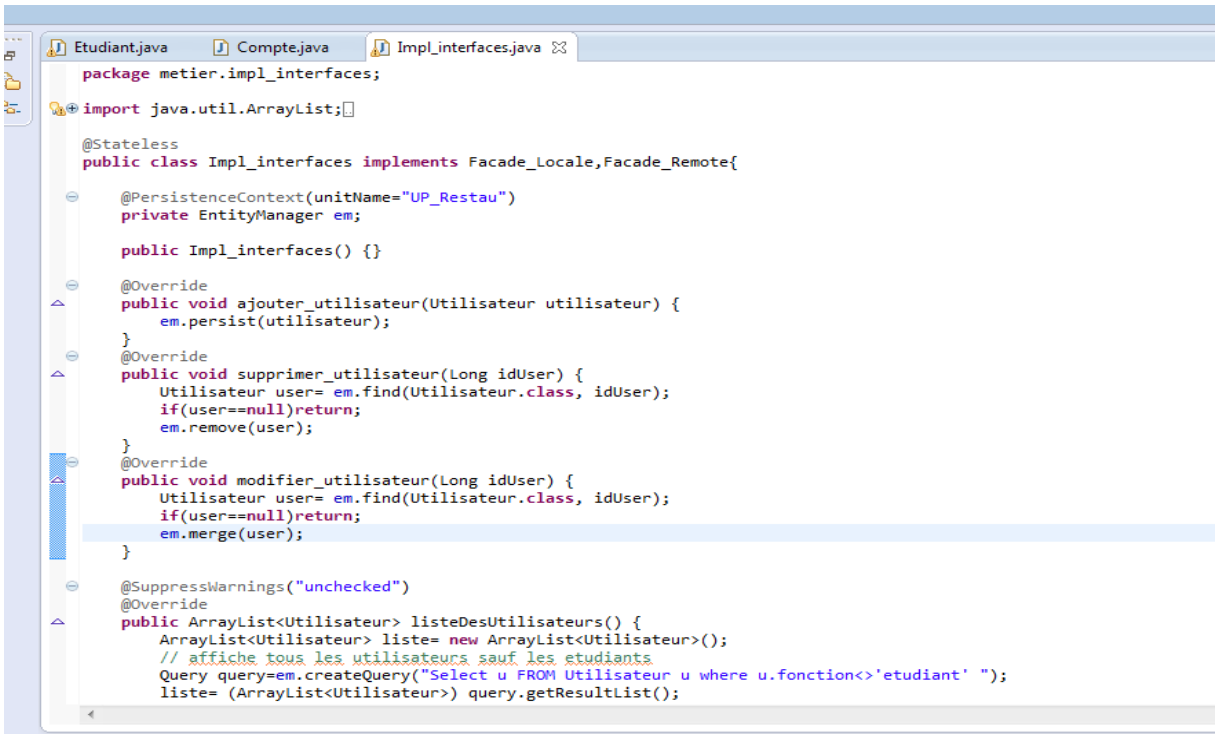

Figure 33: Interface implémentation

II. Présentation du système

Dans cette partie, nous faisons une présentation des fonctionnalités de cette application. Cependant certaines de ces interfaces ne sont pas présentées du fait que le développement est toujours en cours.

1. L'interface de connexion

Cette interface permet aux différents utilisateurs de se connecter. Si l'utilisateur donne un login et un mot de passe correct, il accède à son page d'accueil, sinon un message d'erreur lui est envoyé.

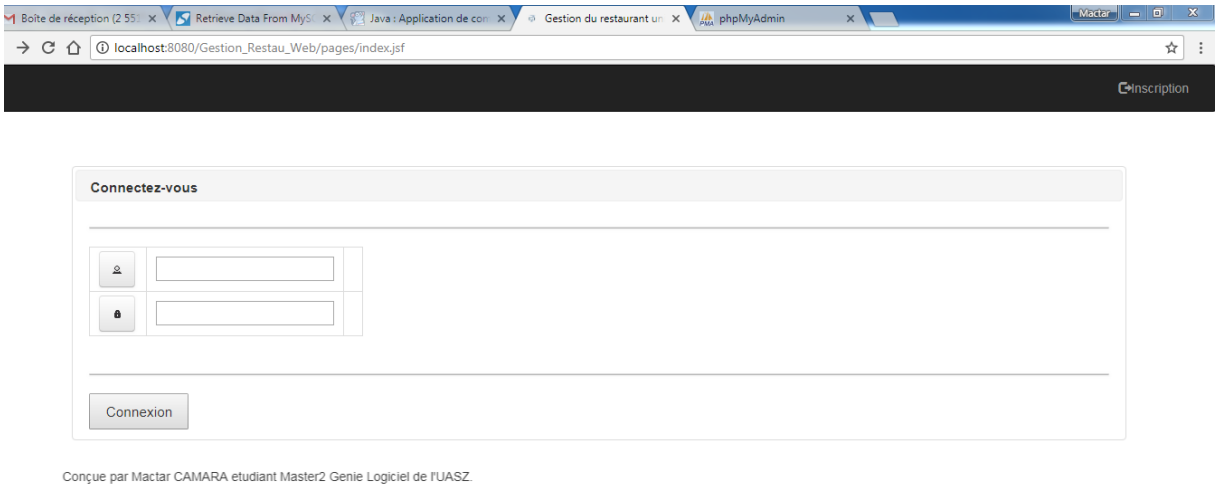

Figure 34: L'interface de connexion

2. L'interface du vendeur

Cette interface permet au vendeur de créditer les comptes, d'activer les comptes, mais aussi d'ajouter des étudiants qui n'existent pas encore dans la base de données. Les **figures 35, 36, 37 et 38**suivantes illustrent les différentes activités que peut faire le vendeur :

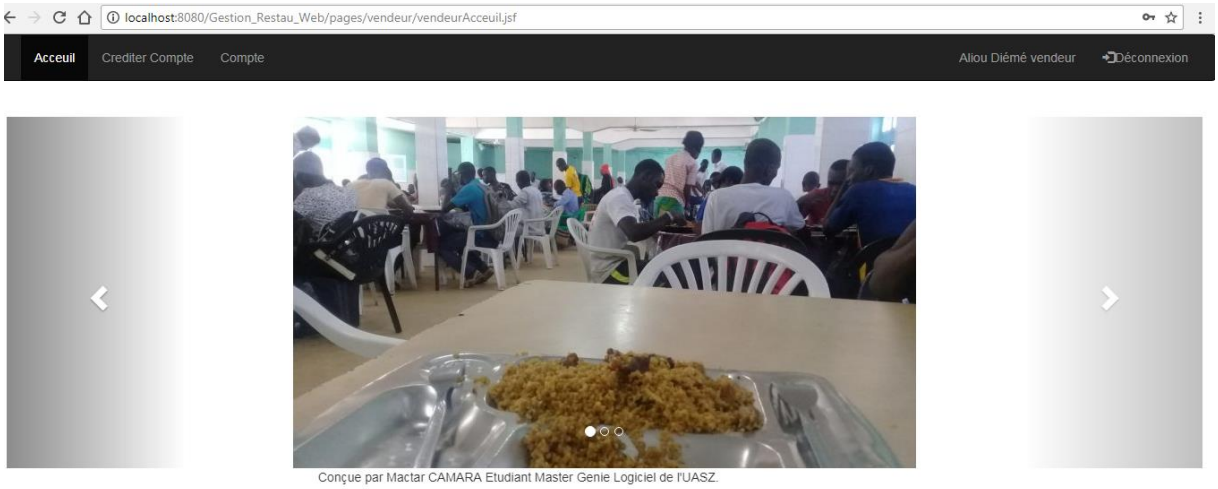

Figure 35: Interface accueil du vendeur

Pourcréditer un compte, il suffit juste de faire un double clic sur la ligne du compte à créditer, ensuite une boite de dialogue s'affiche pour saisir les données, afin de valider l'opération.

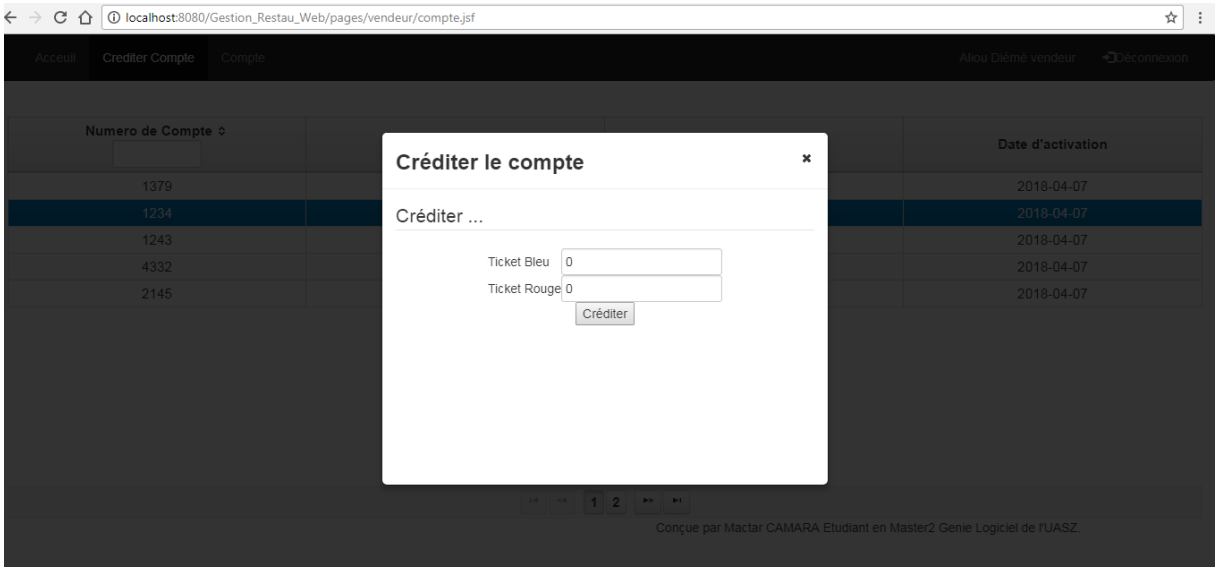

Figure 36: Interface créditer Compte

L'onglet « activer compte » permet au vendeur d'activer le compte des étudiants nouvellement ajoutés.

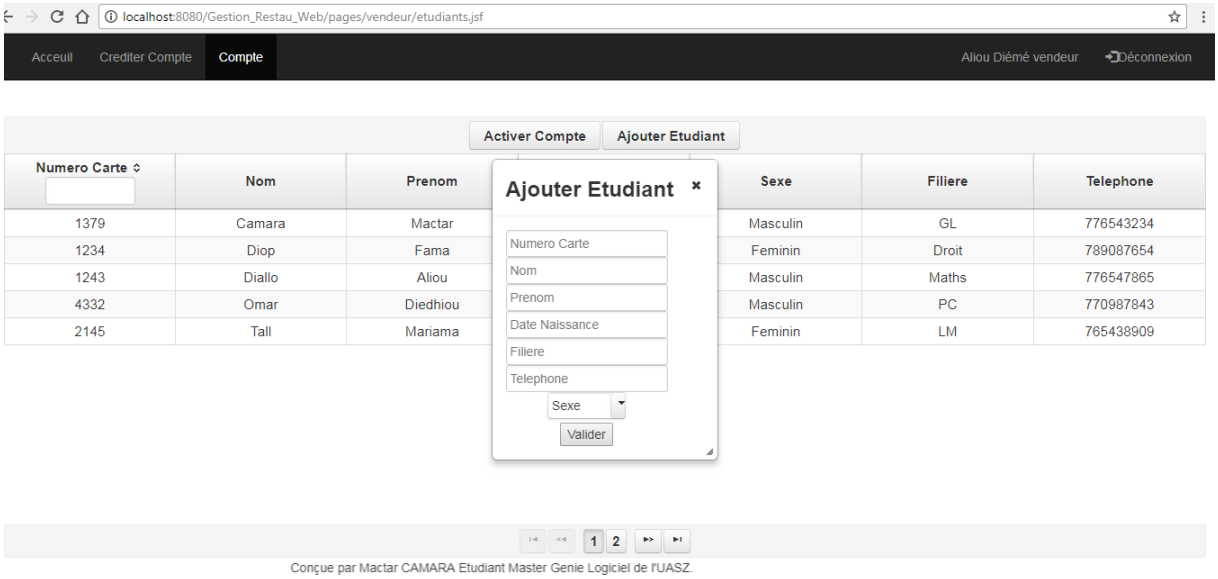

Figure 37:Fonctionnalité ajout étudiant

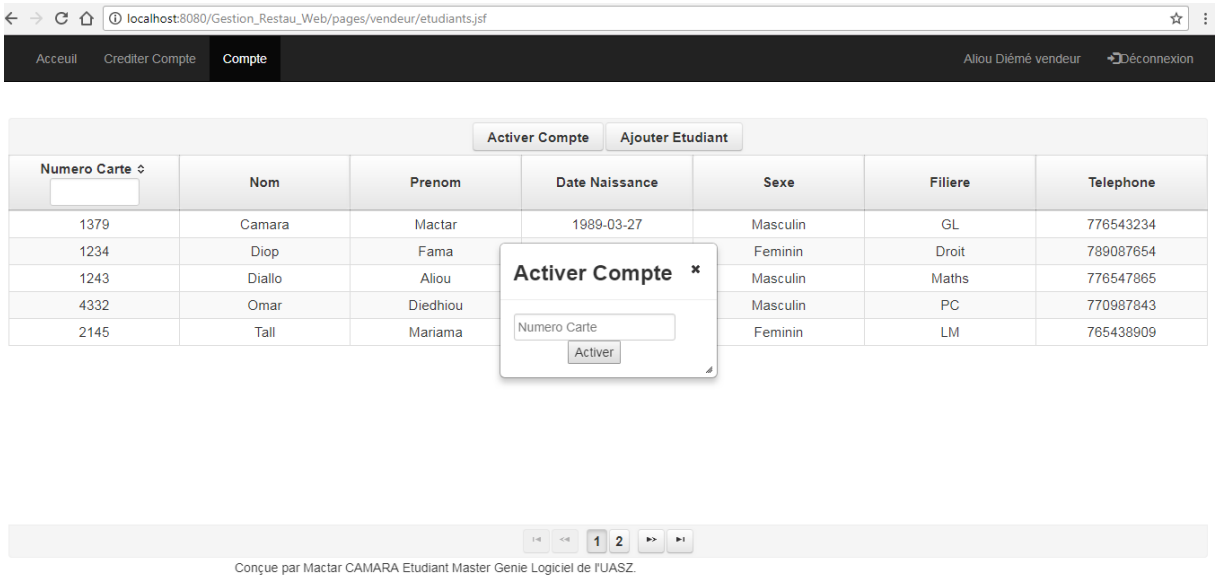

Figure 38: Fonctionnalité activer compte

3. L'interface du portier

Cette interface permet à son utilisateur de faire seulement des opérations de débits sur le compte des étudiants qui veulent accéder au restaurant. Les **figures 30 et 40** qui suivent montrent respectivement la page d'accueil et celle de la fonctionnalité « débiter » :

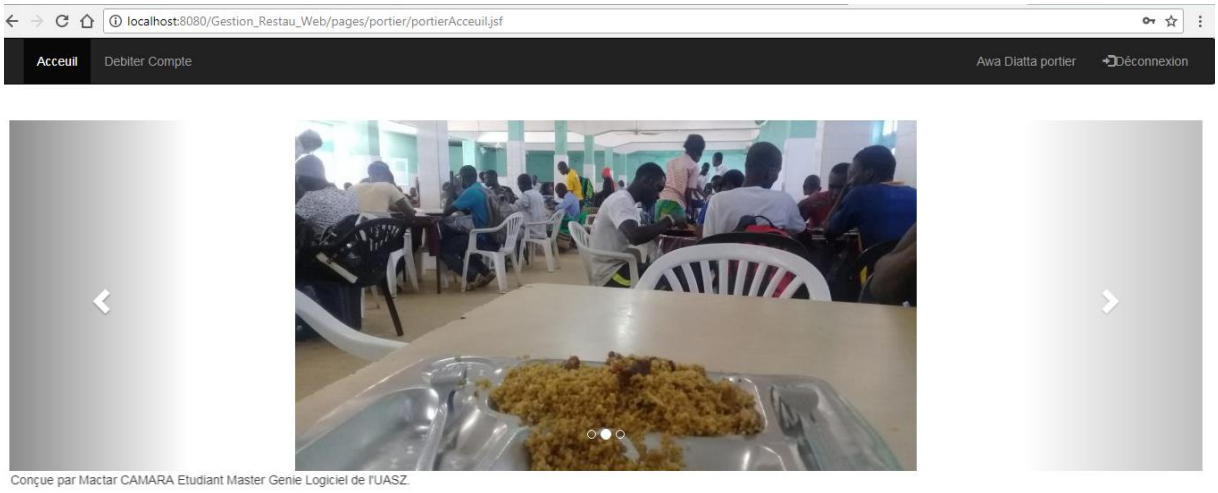

Figure 39: Page d'accueil du portier

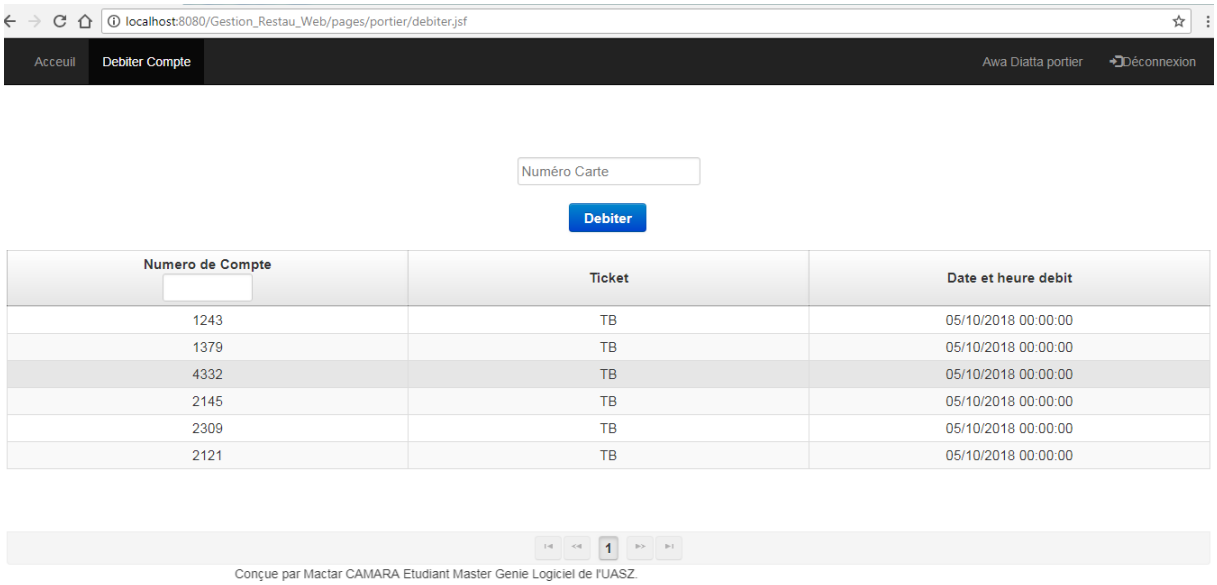

Figure 40: L'interface débiter compte

4. L'interface de l'étudiant

Cette interface permet à son utilisateur de faire un suivi de ses achats et de ses entrées, de faire ses achats et de transfert de tickets entre compte. Les **figures 41, 42 et 43** suivantes montrent les différentes actions :

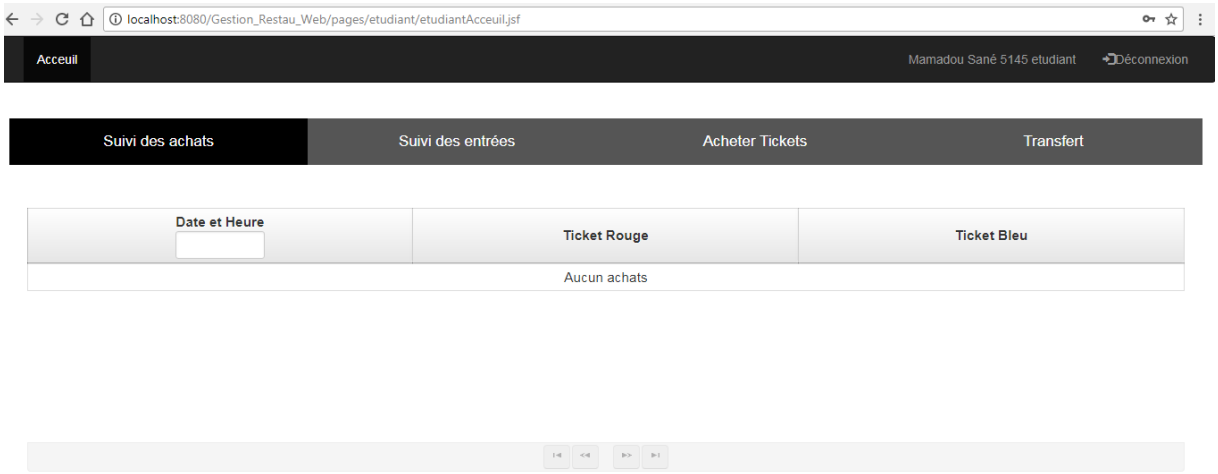

Figure 41: Fonctionnalité suivi des achats

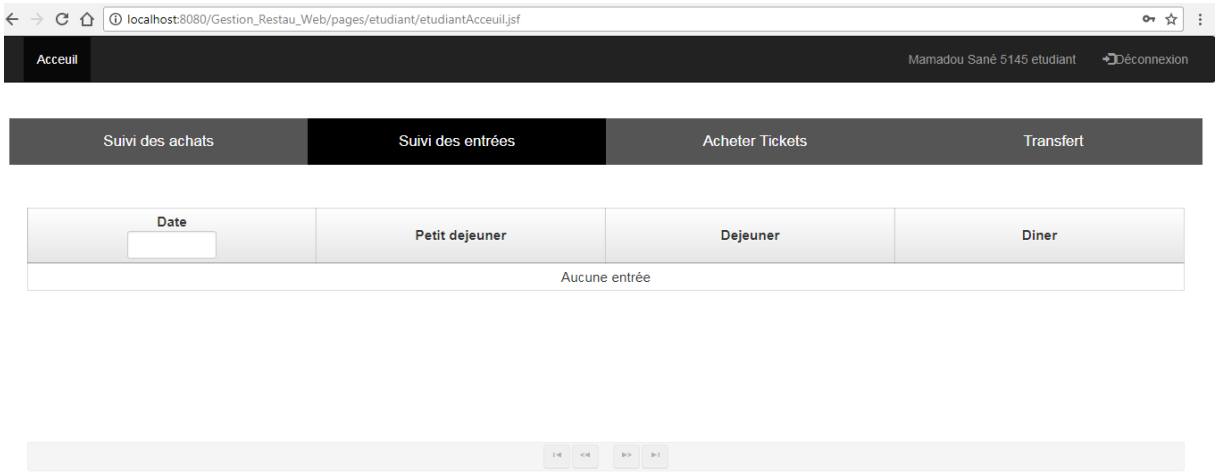

Figure 42: Fonctionnalité suivi des entrées

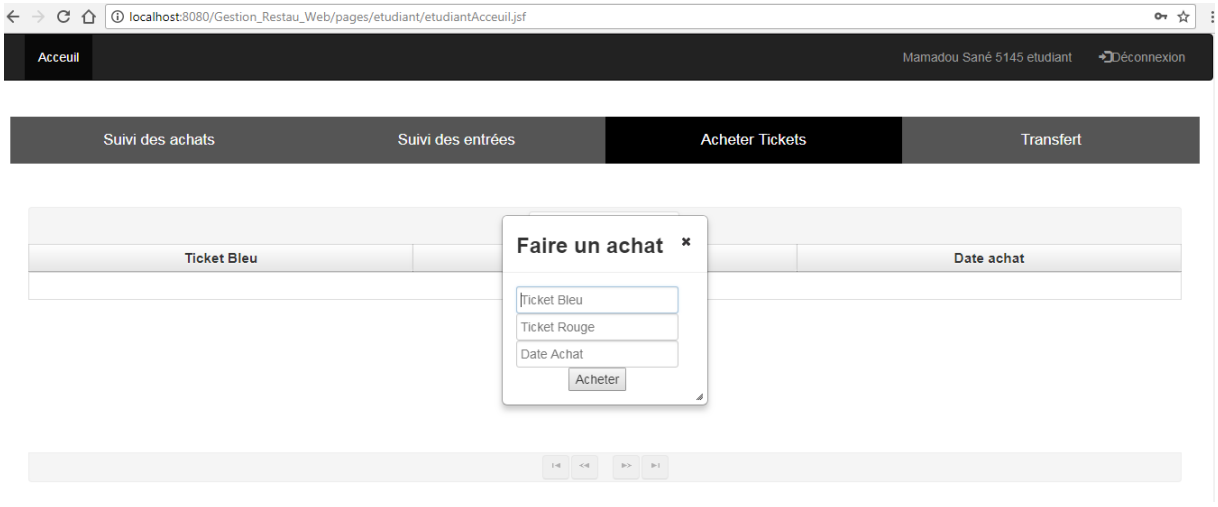

Figure 43: Fonctionnalité achat de tickets

5. L'interface de l'administrateur

Cette partie offre à son utilisateur des fonctionnalités d'ajout, de modification, de recherche sur les utilisateurs du système. Les **figures 44, 45 et 46** qui suivent illustrent ses différentes activités :

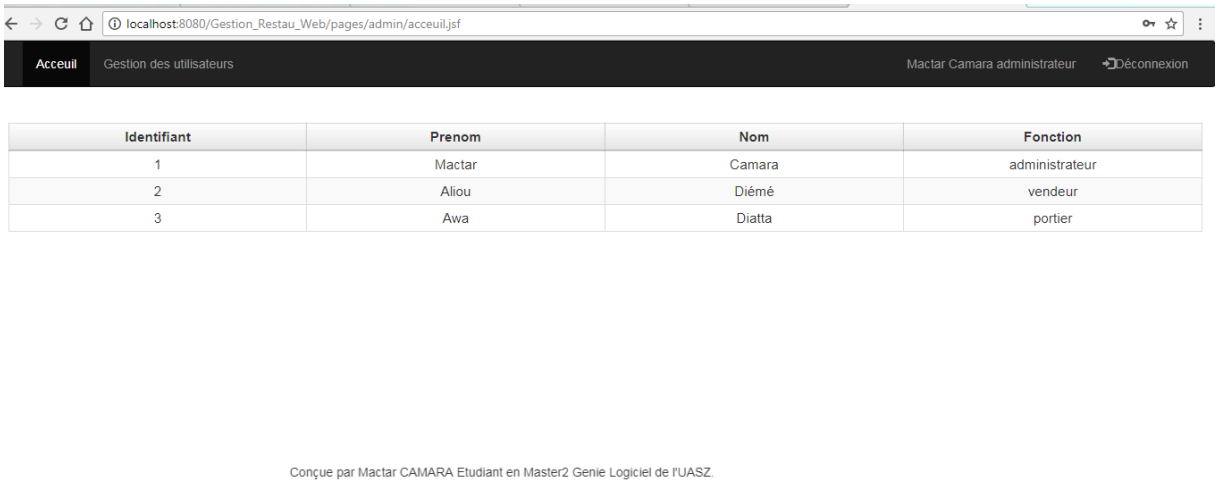

Figure 44: Page d'accueil de l'administrateur

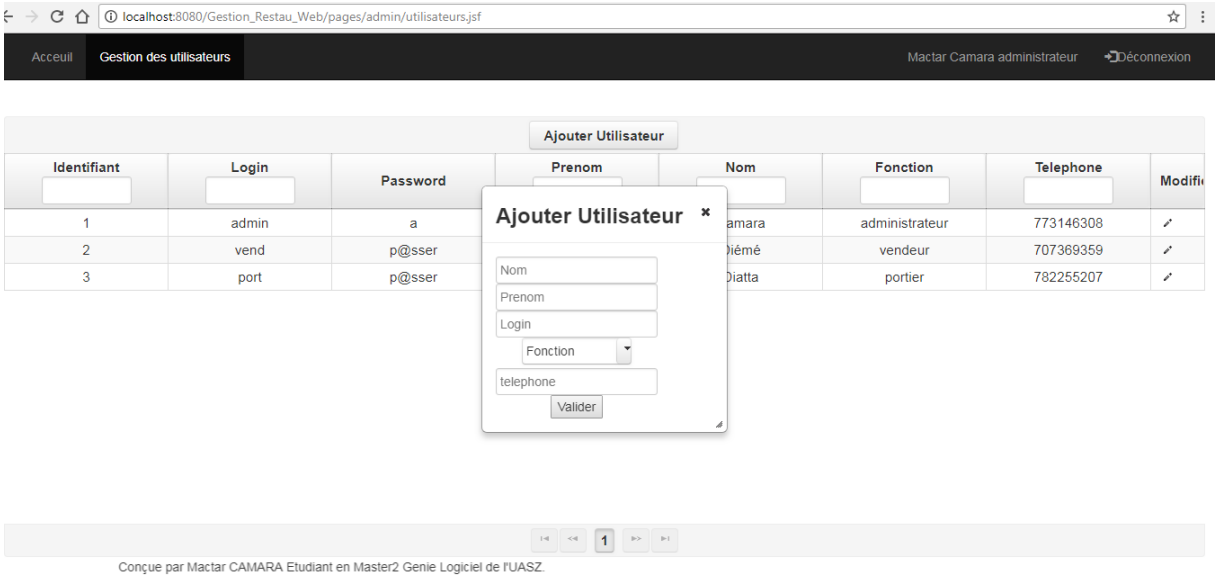

Figure 45: Fonctionnalité d'ajout utilisateur

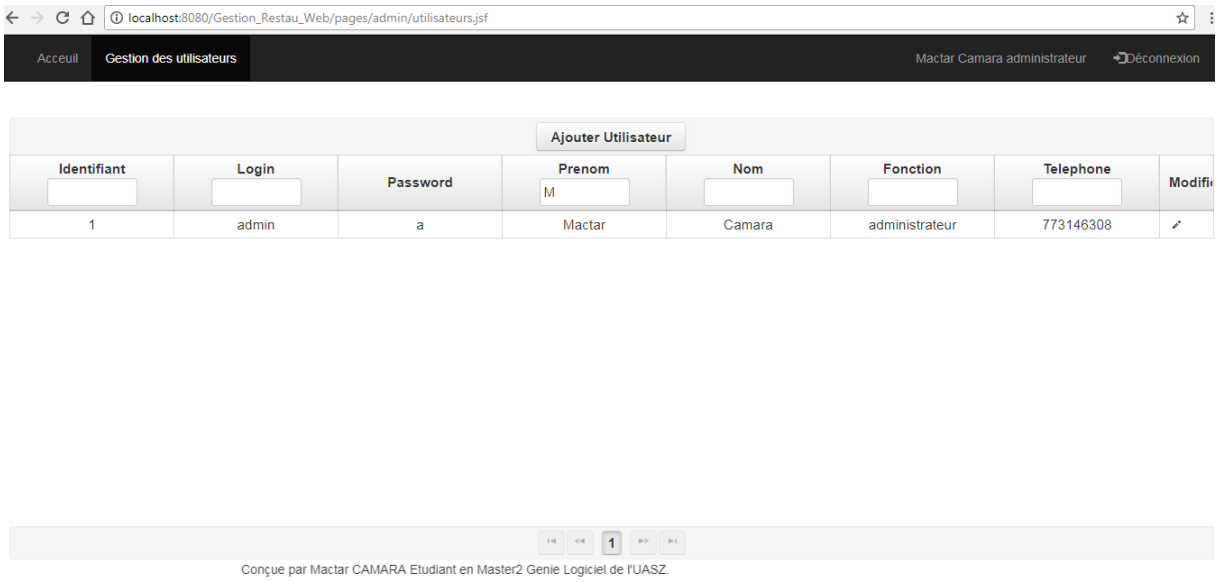

Figure 46: Fonctionnalité de recherche d'utilisateur

Ce chapitre nous a permis dans un premier temps d'aborder l'implémentation dans laquelle nous avons parlé des outils de développement, du modèle logique des données et du codage des différentes interfaces du système et dans un second temps nous avons fait une présentation des différentes interfaces réalisées pour notre système.

La partie qui suit fait une conclusion générale de ce présent mémoire, ouvrant aussi des perspectives pour une bonne amélioration de notre système.

Conclusion et perspectives

En définitive l'objectif de ce mémoire était d'automatiser l'activité de gestion du système de la vente des tickets et de l'accès au restaurant universitaire. Pour cela, j'ai réalisé une application interactive permettant de gérer les différents traitements de cette activité et de satisfaire les besoins des différents utilisateurs impliqués dans ce processus de gestion.

J'ai commencé mon travail par la compréhension du contexte de mon projet. Ensuite, j'ai réalisé une étude dans ce domaine, afin de pouvoir fixer les anomalies à éviter et les objectifs à réaliser pour avoir un système satisfaisant. En plus, j'ai passé à l'étude conceptuelle de mon application selon une approche orientée objet tout en me basant sur le langage UML et le processus unifié 2TUP. Par la suite, j'ai effectué le codage et l'implémentation de l'application.

Ce projet a été très bénéfique pour moi car il m'a permis non seulement de renforcer et enrichir mes connaissances théoriques dans le domaine de la conception, mais aussi de mettre en application mes connaissances acquises le long de mes études. Il m'a encore donné l'occasion de maîtriser le langage de programmation Java, la base de données et de me familiariser avec la conduite de projets informatiques.

En plus, ce projet était une bonne occasion pour réaliser un travail très concret, avec des objectifs clairs et bien définis et de se familiariser avec l'environnement de travail et de la vie professionnelle.

En perspective, nous envisageons d'améliorer cette application en ajoutant d'autres fonctionnalités comme :

- l'achat en ligne des tickets pour crédit son compte d'étudiant avec sa pochette électronique;
- le transfert de tickets entre étudiants ;
- une version mobile destinée aux étudiants afin de leur faciliter l'utilisation de cette application ;
- installer des capteurs de présence pour permettre aux étudiants de voir en temps réel les files d'attente au niveau des RU.

Bibliographie et Webographie

[1] : https://fr.scribd.com, consulté le 10/03/2018, https://fr.scribd.com/doc/49697489/Processus-de-Developpement-Y-Processus-2TUP

[2] : Zakaria BOUAZZA, Sofiane BENZERHOUN « 2TUP », https://fr.slideshare.net, consulté le 10/03/2018,https://fr.slideshare.net/preemptif/mthodologie-2-track-unified-process

[3] : M. Cheikh Souleymane BADIANE, « Conception et Développement d'une application informatique pour l'automatisation de la gestion des congés au sein de la DRH de l'UASZ et la dématérialisation des documents administratifs qui s'y rapportent », Mémoire de Master 2 Génie Logiciel soutenu en 2018 à l'UASZ.

[4] : www.devellopez.com, consulté le 29/03/2018, https://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-composants-deploiement

[5] :www.sparxsystems.fr, consulté le 29/03/2018,http://www.sparxsyste.fr/resources/uml2_tutorial/uml2_packagediagram.html

[6] :http://remy-manu.no-ip.biz, consulté le 29/03/2018, http://remy-manu.no-ip.biz/UML/Cours/coursUML9.pdf

[7] : Dr. Joseph NDONG, «JEE : EJB (Enterprise JavaBeans) » Janvier 2010, FST/UCAD

[8] : http://miageprojet2.unice.fr/, consulté, 17/03/2018, http://miageprojet2.unice.fr/Intranet_de_Michel_Buffa/Cours_composants_distribu%C3%A9s_pour_l'entreprise_%2F%2F_EJB_2009/TP1_2011_EJB_3.1%2F%2FJPA%2F%2FJSF2ms.

[9] : http://webpages.lss.supelec.fr, consulté le 29/03/2018, http://webpages.lss.supelec.fr/perso/hugues.mounier/Teaching/Java_files/JDocs/eclipseDoudoux.pdf

[10] : https://fr.slideshare.net, consulté 20/03/2018, https://fr.slideshare.net/dghaiesjihed/3-jsf

[11] : M. Ismaïla DIALLO « Développement d'une application de gestion de la scolarité et d'un site Web pour Ziguinchor Institut Polytechnique (ZIP) », Mémoire de Master 2 Génie Logiciel soutenu en 2017 à l'UASZ