INSTITUT FRANCOPHONE INTERNATIONAL

ĐHQGHN

VNU
ĐẠI HỌC QUỐC GIA HÀ NỘI
Vietnam National University, Hanoi

TRAVAIL DE GROUPE

MODULE : GENIE LOGICIEL AVANCE

MISE EN PLACE D'UNE APPLICATION DE GESTION DE STOCK : CAS D'UN RESTAURANT.

Période **: du 08 Août au 19 Novembre 2017**

Rédigé par :

- **MEDOU DANIEL MAGLOIRE,**
- **ABOUBACAR DJIBO Maman Sani** ;
- **ANDRE Perrault**

Etudiant en Master 1 des Systèmes Intelligents et Multimédia, IFI.
Promotion 21

Enseignant :
Dr. HO TUONG VINH

Année académique 2016-2017

LISTE DES FIGURES

LISTE DES TABLEAUX

TABLE DES MATIERES

DOSSIER DE L'EXISTANT

Résumé :

Aperçu :

INTRODUCTION

De nos jours, la croissance fulgurante de la population fait créer plusieurs sources de revenus (business) tels que celle de la restauration. La gestion d'un tel système est un vrai casse-tête pour les gestionnaires, car il présente deux parties qui sont la gestion de la clientèle et celle de la gestion du stock. Ainsi c'est dans le sens d'apporter un ouf de soulagement aux gestionnaires des restaurants que nous avons choisi développer une application dans le cadre du projet de développement du module « **Génie Logiciel Avancée** » afin d'aider les gestionnaires à la gestion de leurs restaurants plus précisément leur stock. Le présent rapport fait état des différentes étapes de la mise en place d'une telle application.

La suite du présent rapport consistera dans un premier temps à faire une étude détaillée du projet. Ensuite nous ferons une étude de l'existant, l'analyse, la conception et la réalisation (implémentation). Dans la partie implémentation, nous présenterons les différentes expérimentations que nous effectuerons. Par la fin nous effectuerons des tests pour évaluer la qualité du code de notre application.

I. ETUDE DE PROJET

Cette partie nous permettra de prendre en main le sujet soumis à notre étude pour effectuer un travail adapté et loin de toutes critiques. Pour ce faire, nous présenterons le contexte et la problématique de notre sujet.

1. Contexte

Le responsable du restaurant **LA CANTEEN** voudrait que la gestion de stock de son restaurant soit informatisée, il va donc de soi qu'une application de gestion doit être mise sur pied. Et durant notre échange, nous avons été mis au courant du fonctionnement attendu dudit restaurant.

2. Problématique

Le principal problème de la gestion de la Canteen est qu'elle se fait toujours de manières archaïque c'est-à-dire manuelle je vais dire sur du papier. Ainsi en quoi l'élaboration d'une application de gestion permettra au responsable d'augmenter ses ventes et mieux supervisé le fonctionnement de son restaurant ?

3. Objectifs

L'objectif du présent projet est de mettre en place une application de gestion du stock du restaurant de la Canteen de notre dortoirs (KTX My Dinh ca) afin de permettre au responsable d'avoir une bonne vision de sa gestion, aussi pour lui permettre augmenter ses ventes en étendant le champs de ses ventes sur le web. Au niveau des objectifs spécifiques qui sont les différents modules de notre application, nous allons les énumérer dans la partie III qui n'est rien d'autre que notre cahier des charges.

4. État de l'art

L'état de l'art est une phase importante dans toute étude d'un projet. Dans cette section, nous faisons un état des lieux de quelques applications sur la gestion de stock d'un restaurant existants. En effet plusieurs applications existent dans ce sens, mais deux (**GSM** et **SoftCaisse-Restaurant**) ont attiré notre attention de par leur simplicité et des fonctionnalités qui nous seront nécessaire pour la réalisation de notre projet. Nous avons :

- ✓ **Gestion de Stock Modulaire (GSM),** propose à l'utilisateur de gérer les stocks de marchandises en toute simplicité. Le logiciel gère les entrées/sorties directes des stocks mais peut également les planifier. Il s'occupe également de la planification des fabrications et de la valorisation CUMP. Enfin, on retrouve dans **Gestion de Stock Modulaire** une organisation classique avec référence/quantité/valeur et lorsqu'un seuil minimum de quantité s'en voit atteint, une alerte apparaît en colorisant la case en question. Disponible au : www.logiciel-gestionstock.fr.

- ✓ **SoftCaisse-Restaurant,** est une application qui a pour but de gérer les articles, fournisseurs, ventes, livraisons et client d'un restaurant. Il dispose de plusieurs fonctionnalités nous permettant de réaliser notre projet. Disponible au www.ami-web.com

5. La solution proposée

La solution que nous proposons pour la mise en place de notre application se base sur les deux applications présentées ci-haut dans l'état de l'art. Elle consiste à combinée les idées des deux applications énoncé précédemment. La première est celle de la gestion du stock (entrée, sorties des produits, gestion des alertes), la deuxième est celle de la gestion des articles, fournisseurs, ventes, et de la commande des clients en ligne. Nous présentons dans la suite de cette section les exigences fonctionnelles et non fonctionnelles de l'application que nous avons proposée, ainsi que le product backlog de notre implémentation.

II. CAHIER DES CHARGES

a. Gestion des exigences fonctionnelles

Les exigences fonctionnelles listent les opérations réalisables avec l'application, elles correspondent également à la manipulation et précisent l'environnement de l'application. Il s'agit des fonctionnalités du système.
Afin de fournir un travail de qualité à la hauteur des attentes des différents besoins qui nous ont été exposé lors de notre entretien avec le responsable, il est question pour nous de ressortir les différents modules de gestion qui constitueront au final l'application solliciter du restaurant et qui seront plus tard implémentés. Chacun de ces modules implémentant le fonctionnement d'un service du restaurant. Il s'agit de :

- ❖ Gestion du stock des articles
- ❖ Gestion des fournisseurs
- ❖ Gestion de l'Utilisateur (Administrateur, Gestionnaire et Caissier)
- ❖ Gestion caisse

- ❖ Gestion client
- ❖ Inventaires
- ❖ Journal (Logs)

b. Etude du fonctionnement du restaurant LA CANTEEN

La gestion d'un restaurant étant assez dense, comme présenté par les différents modules ci-dessus cités. Il est donc question pour nous de présenter de manière la plus simple et la plus abordable possible ces-derniers.

- ❖ Gestion fournisseur

Quand un fournisseur arrivera au sein du restaurant pour solliciter de fournir un produit quelconque dans la structure, le gestionnaire consultera son application pour voir si ce dernier est déjà enregistré dans la base de données. Sinon, il présentera le bon de commande envoyé par la structure ainsi que son bon de livraison. Une fois que c'est fait, le gestionnaire procède à son enregistrement en consignant les informations suivantes:

- ➢ Enregistrement du fournisseur
  - Nom
  - Prénom
  - Mail
  - Photo

En plus de l'enregistrement du fournisseur, le gestionnaire peut également :

- ➢ Ajouter un fournisseur
- ➢ Assigner un produit à un fournisseur
- ➢ Modifier un fournisseur
- ➢ Afficher la liste des fournisseurs
- ➢ Supprimer un fournisseur
- ➢ Imprimer la liste des fournisseurs

- ❖ Gestion du stock des articles

La gestion des entrées/sorties du restaurant en terme de articles (produit alimentaire) se fait par le Comptable matière (gestionnaire de stock) selon qui suit :

- ➢ Enregistrement de l'article: en consignant les informations suivantes
  - code Article
  - Référence
  - Désignation
  - Quantité
  - Prix Unitaire
  - Fournisseur
  - Catégorie

En plus de l'enregistrement du produit livré, le gestionnaire peut également :

- ➢ Ajouter un produit
- ➢ Assigner un produit à un fournisseur
- ➢ Modifier un produit
- ➢ Afficher la liste des produits par mot clé
- ➢ Afficher la liste des produits sortis
- ➢ Imprimer la liste des produits

- Sortie des produites
- Alerte avant rupture de stock (Stock d'alerte). Ceci de façon automatique par le logiciel
- Supprimer un produit

❖ Gestion des utilisateurs

La gestion du personnel du restaurant se fait par le l'administrateur :

- Enregistrement de l'employé: en consignant les informations suivantes
  - Nom
  - Prénom
  - Mail
  - Photo

En plus de l'enregistrement de l'employé, l'administrateur peut également :

- Ajouter un employé
- Modifier un employé
- Supprimer ou désactiver un employé
- Afficher la liste des employés
- Imprimer la liste des employés
- Attribuer des droits aux employés

❖ Gestion de la caisse

La gestion de la caisse du restaurant se fait par le Caissier selon qui suit :

- Ventes
- Afficher la liste des ventes
- Générer le total des ventes (Modification et suppression)

❖ Les inventaires

Les inventaires se font en comparant les états suivants:

- Etats entrées
- Etats sorties

❖ Le journal (Log)

Le journal est considéré ici comme notre mouchard qui devrait enregistrer toute activité qui est effectuée dans notre système. Cette fonctionnalité est réservée uniquement à l'administrateur et nous pouvons effectuer comme opération:

- Imprimer les états (événements)
- Modifier un événement
- Afficher les événements d'une période à une autre
- Exporter les événements (fichiers csv, xls…).

c. Exigences non fonctionnelles

Ils accompagnent les exigences fonctionnelles et sont généralement les spécificités avec lesquelles les besoins fonctionnels doivent être réalisés. Ce sont les besoins en matière de performance, de type de matériel ou de type de conception. Ces besoins peuvent concerner les

contraintes d'implémentation (langage de programmation, type de SGBD, de système d'exploitation…).

- Contrainte de temps

Le produit et le rapport devront être livrés dans le délai qui nous a été imparti.

- Contrainte de sécurité

L'accès à notre application est contraint à un login et un mot de passe. Donc le côté sécurité est assuré et même les modules de notre application ne sont pas ouverts à tout utilisateur.

NB : Notre application devra être extensible, c'est-à-dire qu'elle pourra y avoir une possibilité d'ajouter ou de modifier de nouvelles fonctionnalités.

d. Product Backlog

Tableau 1: Tableau des fonctionnalités à implémenter dans notre application par ordre de priorité.

Product Backlog					
ID	Nom	Importance	Estimation	Démo	Note
1	Sprint 1: Gestion du Produit Tâcher: • Créer • Ajouter • Modifier • Supprimer	1	8h	Renseigner le nom, la catégorie et la quantité pour la création et l'ajout. renseigner l'ID pour la modification et la suppression	Besoin d'un diagramme de séquence UML.
2	Sprint 2: Gestion du Produit Tâches: • Rechercher • Afficher • Imprimer	2	6h	Renseigner l'ID du produit	Besoin d'un diagramme de séquence UML.
3	Sprint 3: Gestion du Fournisseur Tâches: • Créer • Ajouter • Modifier • Supprimer	3	4h	Renseigner le nom, la catégorie et la quantité pour la création et l'ajout. renseigner l'ID pour la modification et la suppression	Besoin d'un diagramme de séquence UML.
4	Sprint 4: Gestion du Fournisseur Tâches: • Rechercher • Afficher • Imprimer	4	4h	Renseigner l'ID du produit	Besoin d'un diagramme de séquence UML.

5	Sprint 5: Mouvement Stock Tâches: • Entrée stock • Sortie stock	5	6h		Pas besoin du diagramme de séquence UML
6	Sprint 6: Vente: Tâches: • La Vente	6	4h	Indiquer le produit que vous voulez acheter	Besoin d'un diagramme de séquence UML.
7	Sprint 7: Gestion des Utilisateurs Tâches: • Créer • Ajouter • Modifier • Supprimer	7	6h	Renseigner le nom, la catégorie et la quantité pour la création et l'ajout. renseigner l'ID pour la modification et la suppression	Besoin d'un diagramme de séquence UML.

DOSSIER D'ANALYSE

Résumé :

Le dossier d'analyse est un document permettant de présenter une vue logique et dynamique du système de gestion de stock d'un restaurant « GSR », et les limites de ce système et en apporter une solution fiable, abordable et optimale.

Aperçu :

INTRODUCTION

Cette partie sera développée en deux sous-sections à savoir une méthodologie et la capture des besoins fonctionnels

I. METHODOLOGIE

La méthodologie consiste à trouver une méthode d'analyse convenable et adaptée au thème qui nous est soumis. Les méthodes d'analyse sont nombreuses et diverses les unes des autres. Toutes ces méthodes ont chacune des particularités qui les distinguent des autres. Pour que notre travail soit de qualité, nous avons opté pour *La Langage de Modélisation Orienté Objet* ***UML (Unified Modeling Language)*** qui n'est pas une méthode d'analyse mais plutôt un *langage de modélisation*. *UML* du sigle en français *Langage de Modélisation Unifié,* adopte de nombreux *Processus Unifié* pour en faire une méthode. Notre choix s'est donc fait sur le *processus de développement* ***2TUP*** associé à ***UML 2.0.*** Nous présenterons dans cette partie le processus choisie.

1. Présentation du processus de développement choisit : 2TUP

Un Processus définit une séquence d'étape, en partie ordonnée, qui concoure à l'obtention d'un système logiciel ou à l'évolution d'un système existant. L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles.

Un Processus unifié est un processus de développement logiciel construit sur UML ; Il se veut itératif et incrémental, centré sur l'architecture, conduit par le cas d'utilisation et piloté par le risque. La gestion d'un tel processus est organisée d'après les quatre (04) phases suivantes : une pré-étude, une élaboration, une construction et une transition. La modélisation métier est décrite par les six (06) disciplines fondamentales suivantes : la capture des besoins, l'analyse, la conception, l'implémentation, le test et déploiement qui définissent ainsi les activités de développement de tout Processus unifié. Le processus unifié doit donc être connu comme une trame commune des meilleures pratiques de développement, non comme l'ultime tentative d'élaborer de processus universel. Tout processus UP répond aux caractéristiques ci-après :

- ***Il est itératif et incrémental*** : la définition d'itérations de réalisation est en effet la meilleure pratique de gestion des risques d'ordre à la fois technique et fonctionnel. Le suivi des itérations constitue par ailleurs un excellent contrôle de coût et de délais.
- ***Il est piloté par les risques*** : Dans ce cadre, les causes majeures d'échec d'un projet logiciel doivent être écartées en priorité. Tels que l'incapacité de l'architecture technique à répondre aux contraintes opérationnelles et l'inadéquation du développement aux besoins des utilisateurs.
- ***Il orienté utilisateur*** : car la spécification et la conception sont construites à partir des besoins des utilisateurs émis par les acteurs du système.

A l'issue des évolutions du modèle fonctionnelle et de l'architecture technique, *la réalisation du système* en question consiste *à fusionner les résultats des deux (02) branches.* Cette fusion conduit à l'obtention ***d'un processus de développement en Y***, comme illustré par la figure suivante.

2. Représentation schématique du processus 2TUP

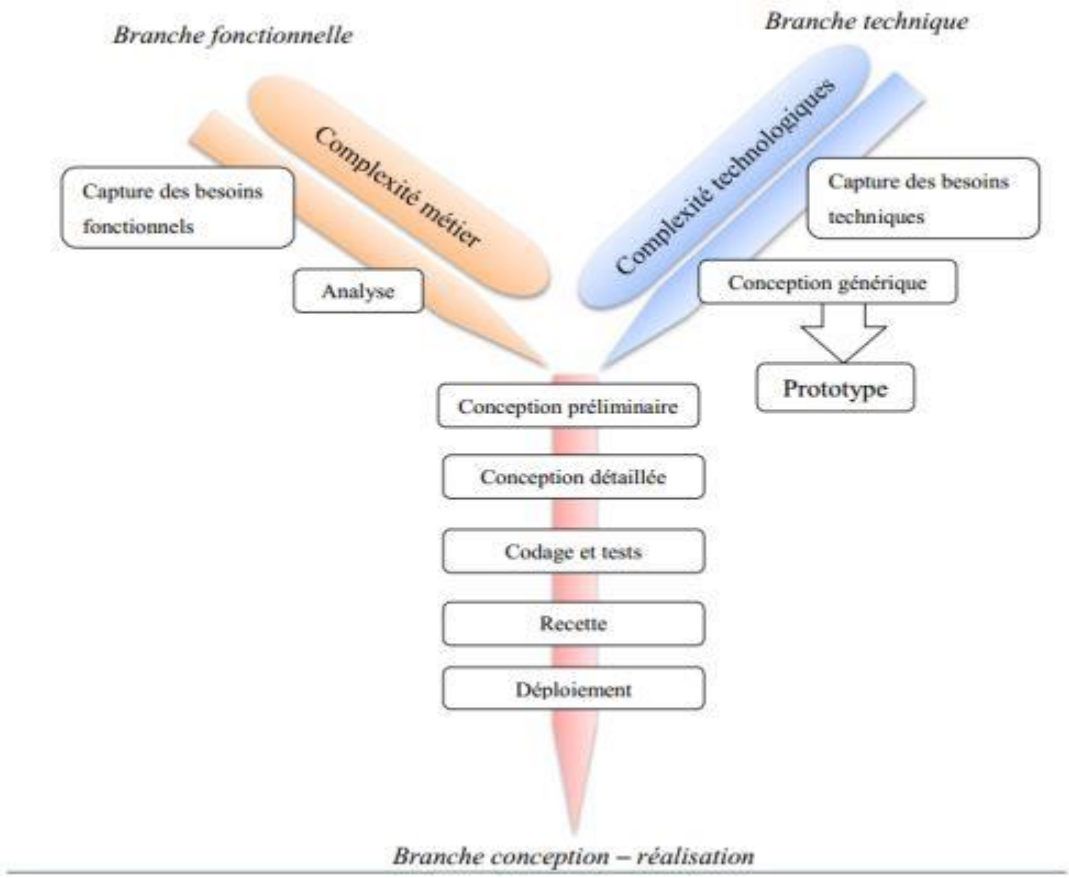

Figure 1: Processus de développement en Y

II. CAPTURE DES BESOINS FONCTIONNELS

Modéliser un système d'information permet de mieux comprendre le fonctionnement du système. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Dans cette partie, nous allons procéder à la capture des besoins fonctionnels et la capture des besoins techniques. La capture des besoins fonctionnels est la première étape de la branche gauche du cycle en Y. Dans cette section, nous allons successivement :

- Identifier les cas d'utilisation du système par acteur ;
- Présenter une organisation des cas d'utilisation (sous forme de diagramme) ;
- Décrire quelques scénarii nominaux et/ou alternatifs des cas d'utilisation.

1. Identifier les cas d'utilisation

Un cas d'utilisation (use case) peut être définit comme la représentation d'un ensemble de séquences et d'actions réalisées par le système et produisant un résultat observable faisant intervenir un acteur particulier. Un cas d'utilisation modélise un service rendu par le système à un acteur du système.

a. Liste des acteurs

Un acteur est toute entité externe, physique ou logique qui interagit avec le système étudié. Un acteur peut consulter et/ou modifier l'état du système. Dans notre projet, les acteurs sont les suivant :

- L'administrateur qui n'est rien d'autre que le Directeur du restaurant
- Le gestionnaire
- Le/la caissier(e)

b. Liste des cas d'utilisation par acteur

Un cas d'utilisation exprime les interactions *acteurs/système* et apporte une valeur ajoutée « notable » à l'acteur concerné. Il permet de spécifier ce *que fera le futur système sans spécifier comment cela sera fait*. Le tableau ci-dessous présente les cas d'utilisation en fonction des acteurs.

Cas d'utilisation	Acteur principal
Gérer article	Administrateur et Gestionnaire
Gérer fournisseur	Administrateur et Gestionnaire
Gérer client	Administrateur
Gérer utilisateur	Administrateur
Gérer rôles	Administrateur
Gérer catégorie	Administrateur
Gérer Caisse	Caissier(e) et Administrateur
Journal	Administrateur

2. Diagramme des cas d'utilisation

Maintenant que nous avons identifié les cas d'utilisation et les acteurs, nous allons pouvoir les représenter sous un diagramme tous ces cas d'utilisation.

a. Formalisme du diagramme

Le diagramme des cas d'utilisation comporte les éléments suivants :

- ***Un acteur,*** représenté par un *bonhomme* avec son *nom inscrit dessous* ;
- ***Le système,*** représenté par un *rectangle* avec son *nom inscrit au-dessus* ;
- ***Le cas d'utilisation,*** représenté par une *ellipse* contenant le nom du cas (généralement un verbe à l'infinitif). Les cas d'utilisation sont représentés à l'intérieur du système ;
- ***Les relations,*** elles sont multiples. On distingue :
    - *La relation d'association*, chemin de communication entre un acteur et un cas d'utilisation. Elle est représentée par un *trait continu.*
    - *La relation d'inclusion ou relation « include »,* elle est l'une des relations existantes entre deux cas d'utilisation. Elle montre que le cas d'utilisation de base utilise systématiquement le cas inclus. Elle est représentée par une flèche interrompue, stéréotypé d'un *« include », d'un « extend » ou d'un « use ».*
    - *La relation de généralisation ou relation d'héritage,* cette relation est existante dans les deux situations (entre cas et entre acteur). Cette relation montre que le

cas (l'acteur) de base est un exemple tout particulier du cas (l'acteur) dont il hérite. Elle est représentée par une flèche d'un trait fort ayant à sa tête un triangle.

La représentation graphique d'un diagramme de cas d'utilisation simplifié se présente comme suit :

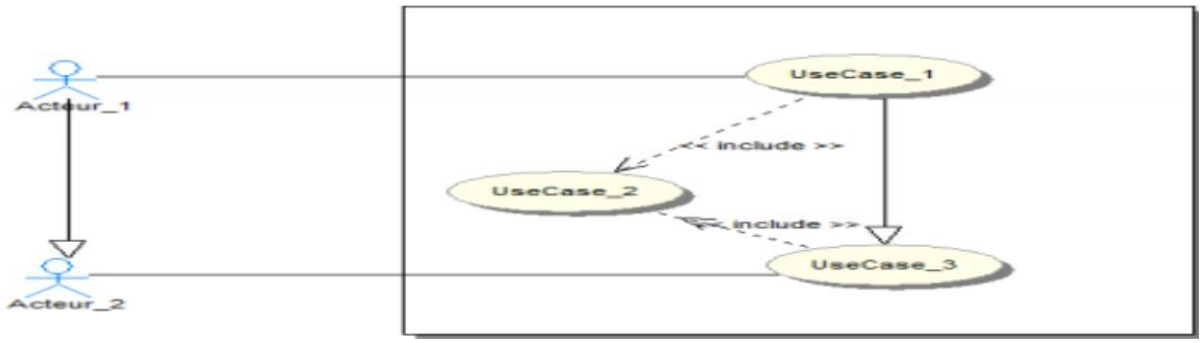

Figure 2: Formalisme du diagramme de cas d'utilisation

b. Représentation des diagrammes des cas d'utilisation :

Pour une meilleure lisibilité, nous avons fait un regroupement *par module de gestion* avant de présenter le diagramme des cas d'utilisation général.

i. Diagramme du cas « Gérer les articles » :

La gestion des articles intègre les fonctionnalités suivantes : *Ajouter un article, Modifier un article, Supprimer un article, Rechercher un article, Afficher la liste des articles, Afficher les informations d'un article.* Le diagramme ci-dessous le présente.

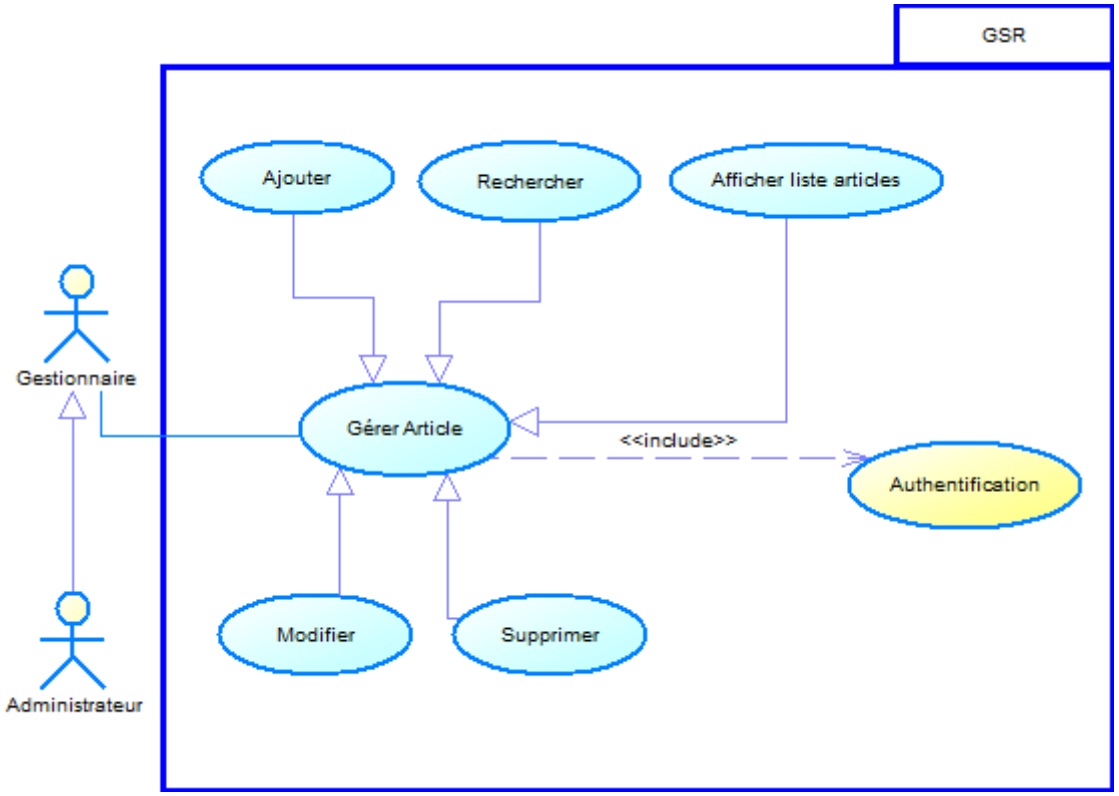

Figure 3: Diagramme de cas d'utilisation "Article"

ii. Diagramme du cas « Gérer les clients » :

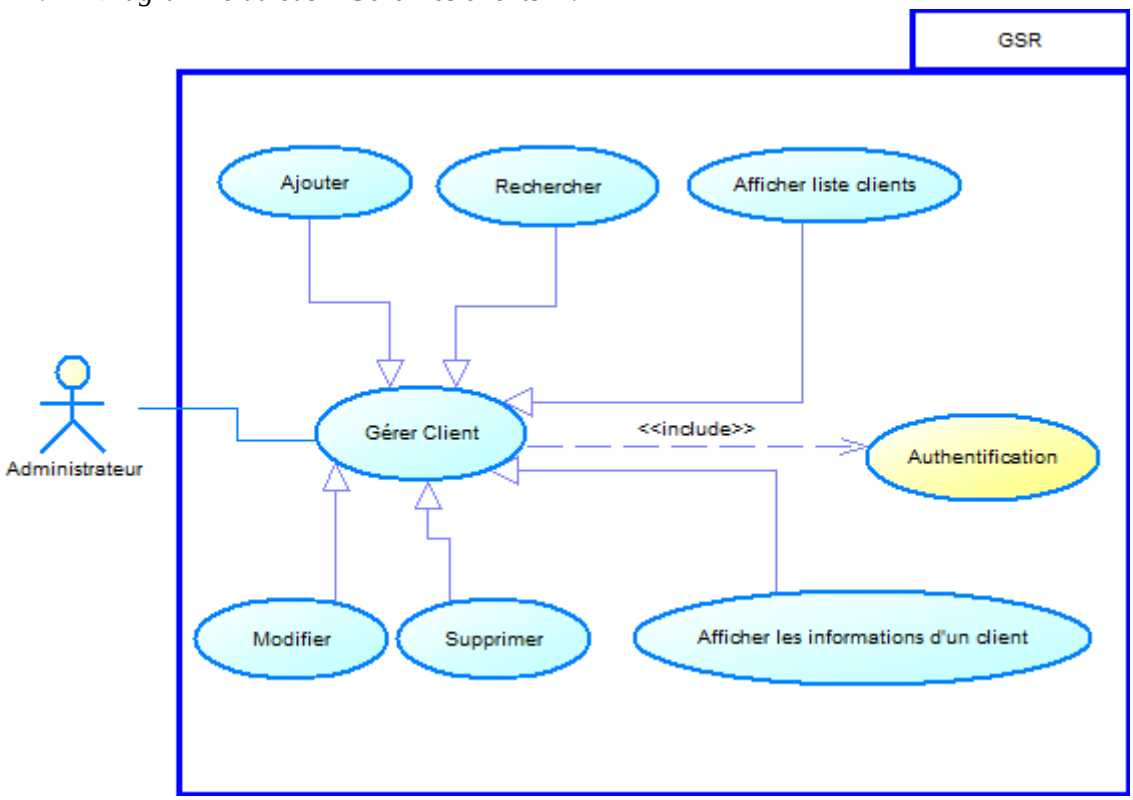

Figure 4: Diagramme de cas d'utilisation "Client"

iii. Diagramme du cas « Gérer les fournisseurs » :

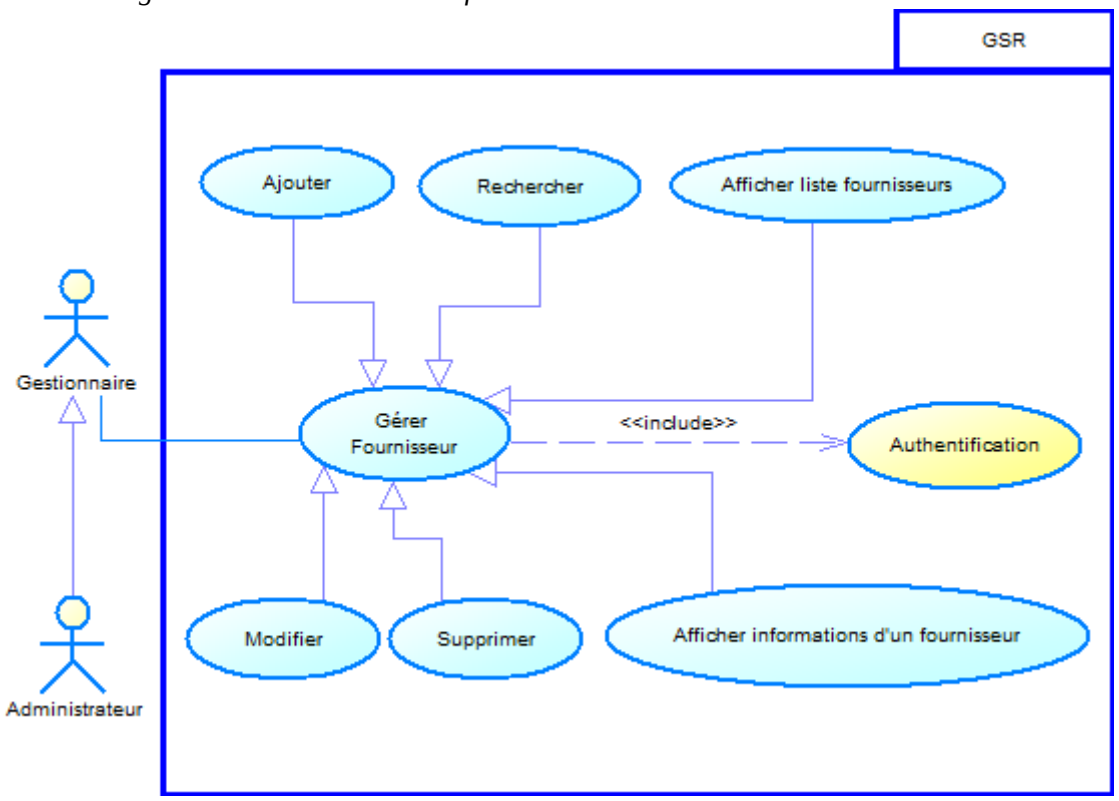

Figure 5: Diagramme de cas d'utilisation "Fournisseurs"

iv. Diagramme du cas « Gérer les utilisateurs » :

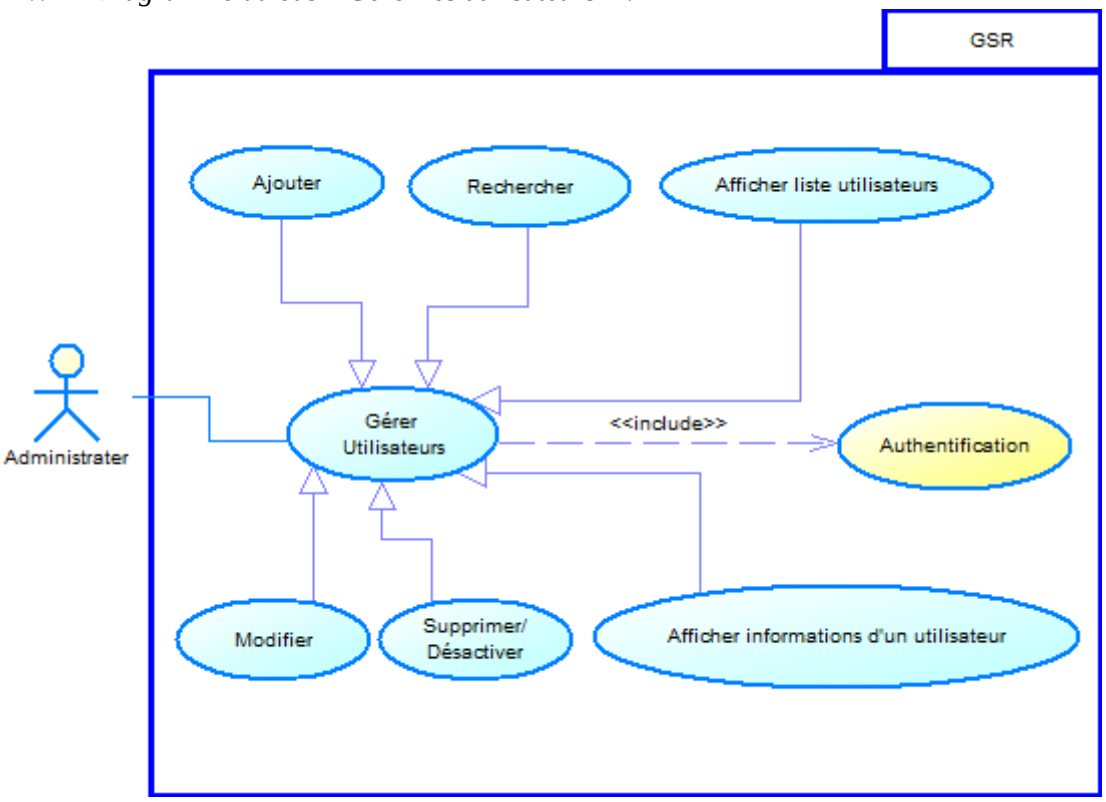

Figure 6: Diagramme de cas d'utilisation "Utilisateurs"

v. *Diagramme du cas « Gérer les rôles » :*

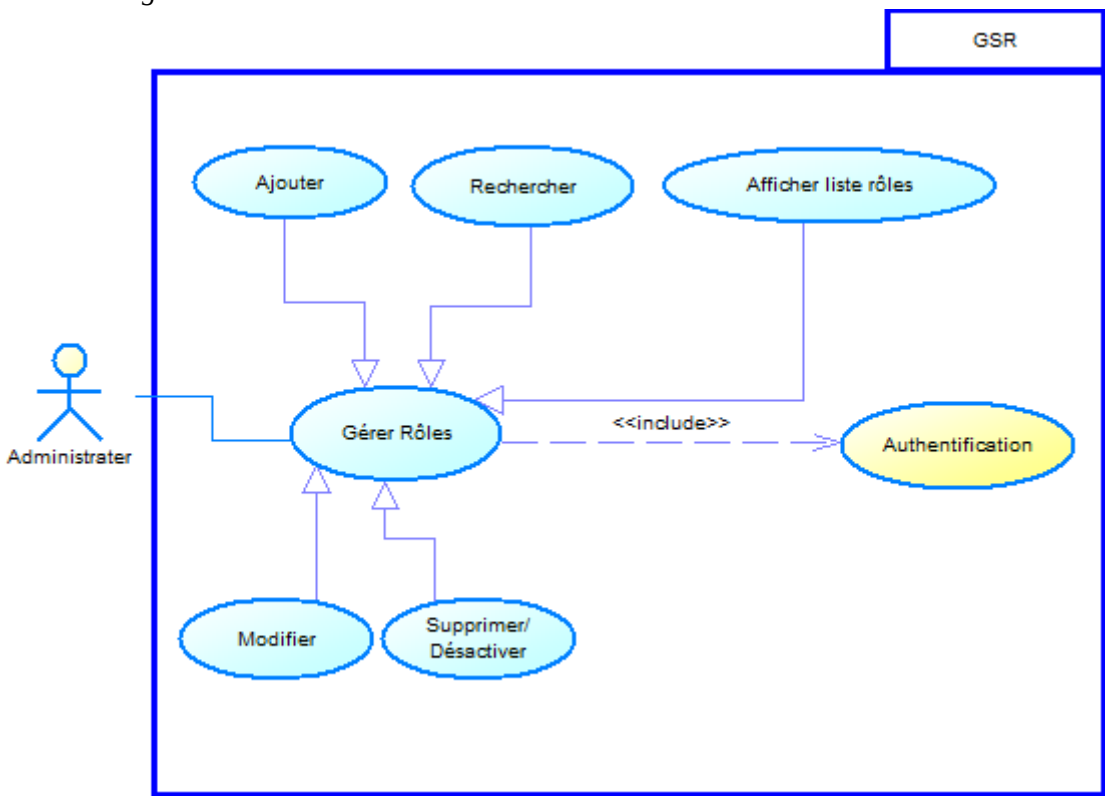

Figure 7: Diagramme de cas d'utilisation "Rôles"

vi. *Diagramme du cas « Gérer les catégories » :*

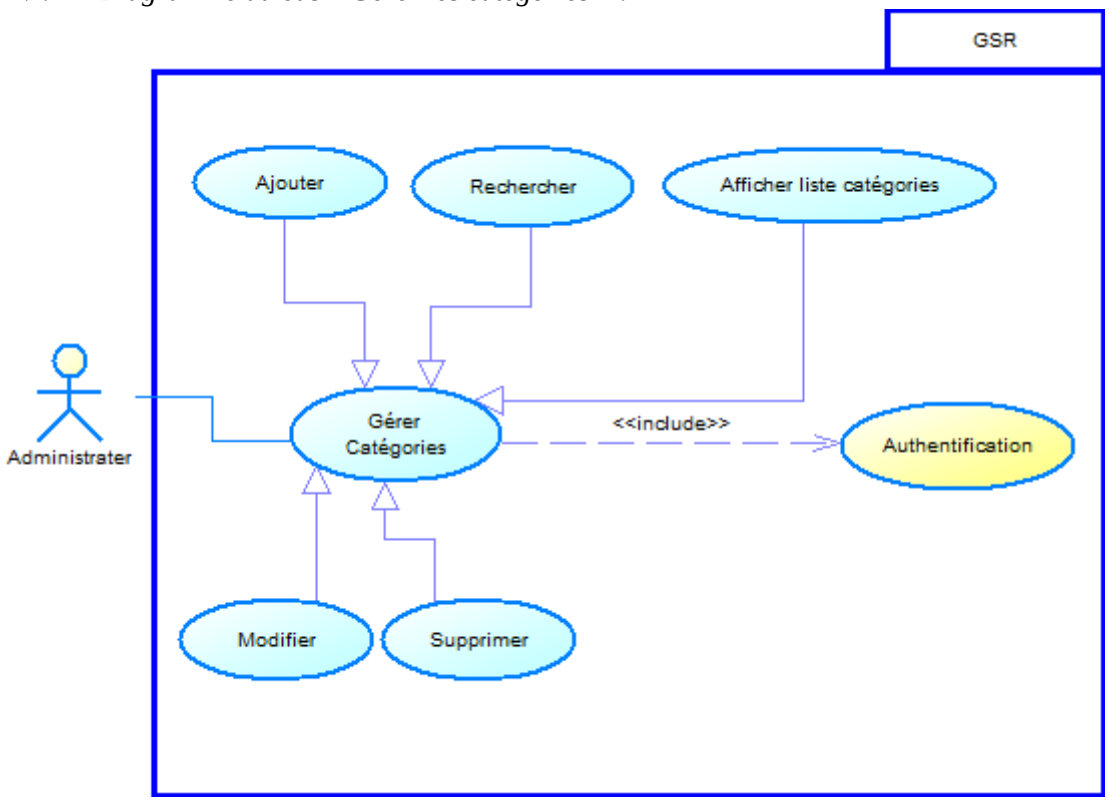

Figure 8: Diagramme de cas d'utilisation "Catégories"

vii. Diagramme du cas « Gérer la caisse » :

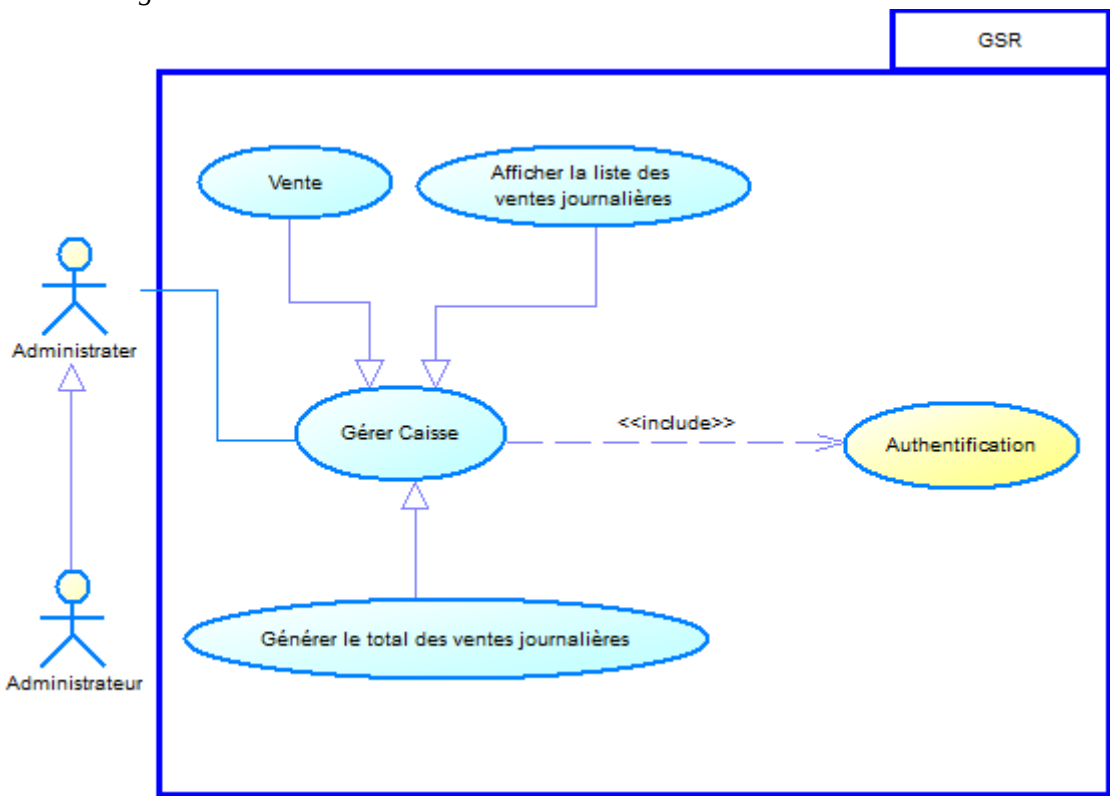

Figure 9: Diagramme de cas d'utilisation "Caisse"

viii. Diagramme du cas « Journal » :

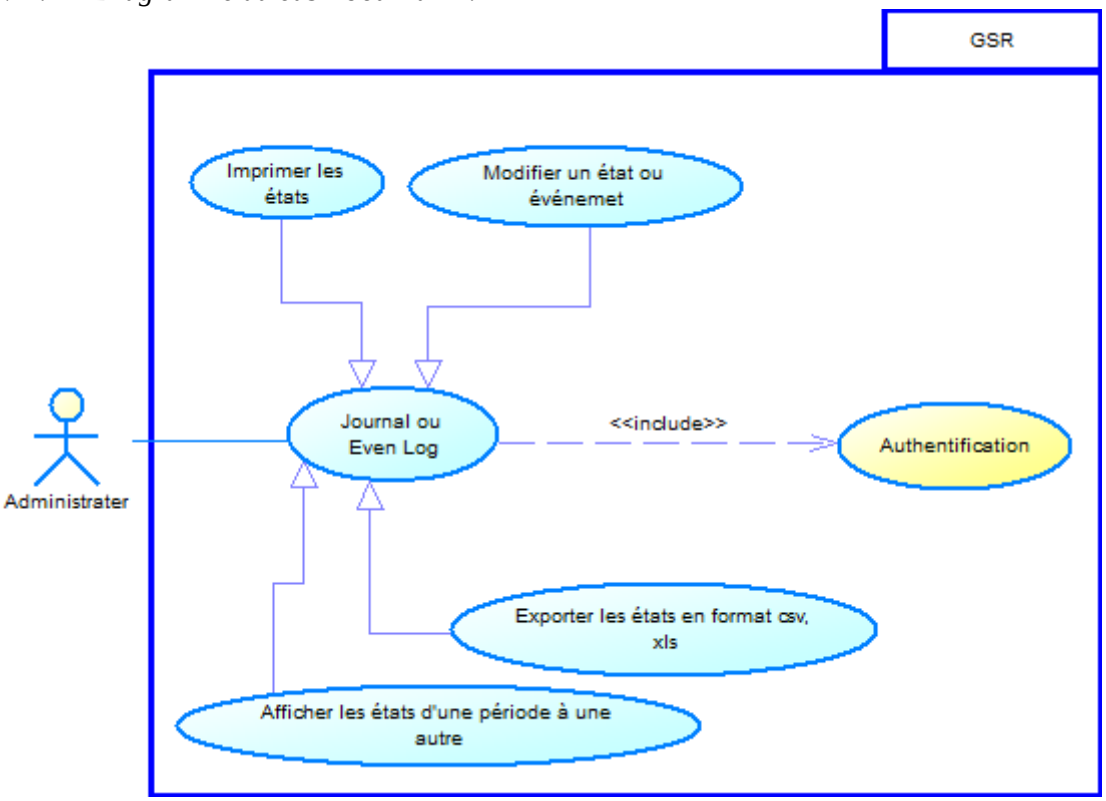

Figure 10: Diagramme de cas d'utilisation "Journal"

ix. Diagramme du cas général du système

Voici une vue général des différent module décrit précédemment sous forme de diagramme.

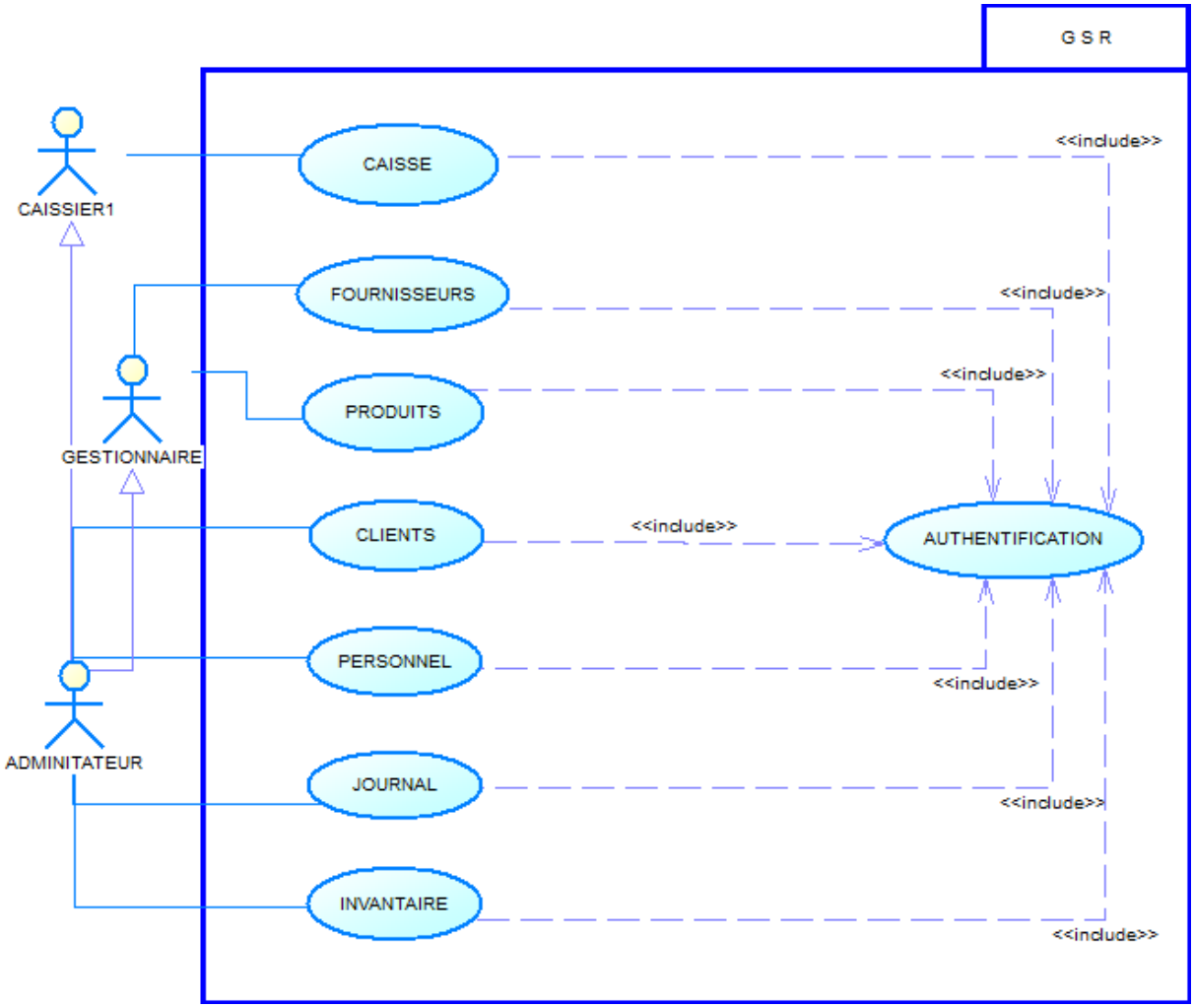

Figure 11: Diagramme de cas d'utilisation général

3. Description textuelle de quelques scénarii

Après avoir présenté les cas d'utilisation qui sont associé à chaque module, nous allons dans cette partie décrire chaque *scénario* qui leur est associé. ***Un scénario*** peut être décrit comme un ensemble d'échanges d'évènements entre l'acteur et le système.

i. S'authentifier

Tableau 2: Scénario du cas "s'authentifier".

Cas d'utilisation : **S'authentifier**
Résumé : Entrer un login et mot de passe valide avant d'y accéder à l'application
Acteur : Employé (Administrateur, Gestionnaire, caissier, etc)
Déclencheur : Démarrer l'application
Scénario nominal : 1. Affichage ou ouverture du formulaire de connexion ;

2. Saisie des informations ; 3. Validation des données saisies ; 4. Ouverture de l'espace de travail.
Scénario alternatif : 1. Saisie erronée et retour au formulaire ; 2. Retour à l'étape 2 du scénario nominal ; 3. Modification des informations ; 4. Validation.
Post-condition : Ouverture de l'espace de travail.

ii. Gérer articles

Tableau 3: Scénario du cas "Ajouter un article"

Cas d'utilisation : **Ajouter article**
Résumé : La commande une fois répondue, les articles sont ajoutés dans le stock.
Acteur : Administrateur et Gestionnaire.
Déclencheur : 1. L'acteur s'authentifie 2. Une commande a été répondue
Scénario nominal : 1. Cliquez sur « Ajouter » ; 2. Renseigner et valider le formulaire ; 3. Enregistrement et affichage direct de l'article ajouté.
Scénario alternatif : 1. Saisie erronée et retour au formulaire ; 2. Retour à l'étape 2 du scénario nominal ; 3. Modification des informations ; 4. Validation.
Post-condition : Article enregistré dans la BDD (Base De Données)

iii. Gérer client

Tableau 4: Scénario du cas "Modifier client"

Cas d'utilisation : **Modifier client**
Résumé : Une fois un client enregistré, ses informations personnelles peuvent être modifiées.
Acteur : Administrateur.
Déclencheur : 1. L'acteur s'authentifie 2. Une modification est demandée
Scénario nominal : 1. Cliquez sur le bouton modifié du client concerné ; 2. Renseigner le ou les champs à modifier et valider le formulaire ; 3. Enregistrement et affichage direct des informations modifiées du client.
Scénario alternatif : 1. Saisie erronée et retour au formulaire ; 2. Retour à l'étape 2 du scénario nominal ; 3. Modification des informations ;

4. Validation.
Post-condition : Mises à jour effectuées et enregistré dans la BDD (Base De Données)

4. Description graphique de quelques scénarii

Décrire un scénario ne se fait pas uniquement de manière textuelle, une description graphique peut tout aussi bien se faire. Pour ce faire, il est plus apte d'utiliser des diagrammes dynamiques (Activité ou état/transition). Dans notre cas, nous utiliserons ***le diagramme d'activité.***

Les diagrammes d'activité permettent de mettre l'accent sur les traitements. Ils sont donc tout à fait adaptés à la modélisation du cheminement de flot de contrôle et de flot de données. Ils peuvent être utilisés pour décrire une fonctionnalité induisant un flot de contrôle traversant le système, décrire avec précision le contenu d'opération d'une classe.

a. Eléments du diagramme

A titre de rappel, Les éléments du diagramme d'activité sont :

- *Le couloir d'unité d'organisation,* qui identifie **l'acteur ou la classe.**
- *La transition ou flux*, représentée par **une flèche** ;
- *Le branchement conditionnel (Décision),* représenté par **un losange** ;
- *L'activité,* représentée par un rectangle **arrondi** ;
- *Le Nœud objet ;*
- *L'état initial,* représenté par un **cercle plein** ;
- *L'état final.*

b. Formalisme du diagramme d'activité

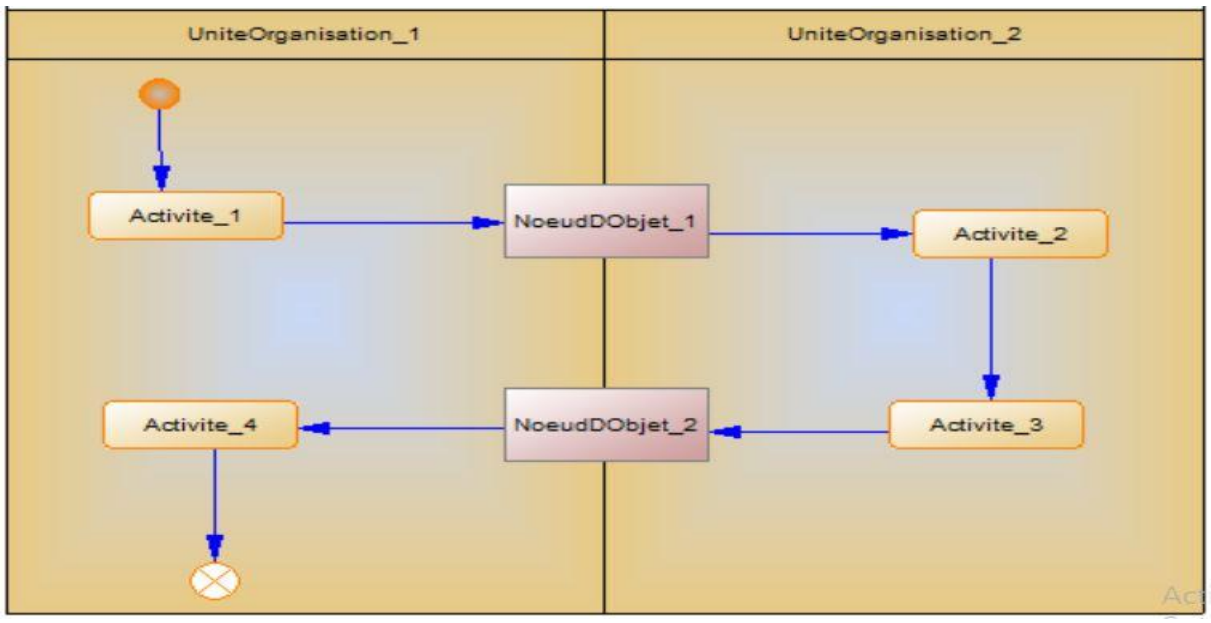

Figure 12: Formalisme du diagramme d'activité

c. Représentation de quelques diagrammes d'activités

i. Diagramme d'activité « Authentification »

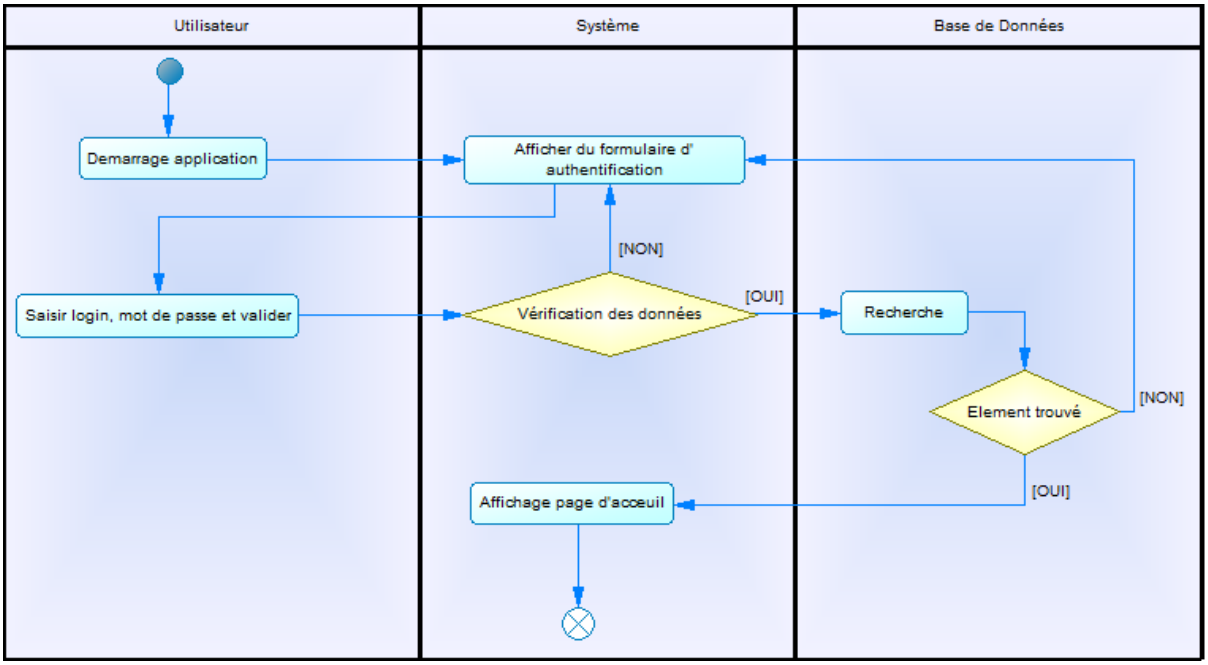

Figure 13: Diagramme d'activité "Authentification"

ii. Diagramme d'activité « Ajouter Article »

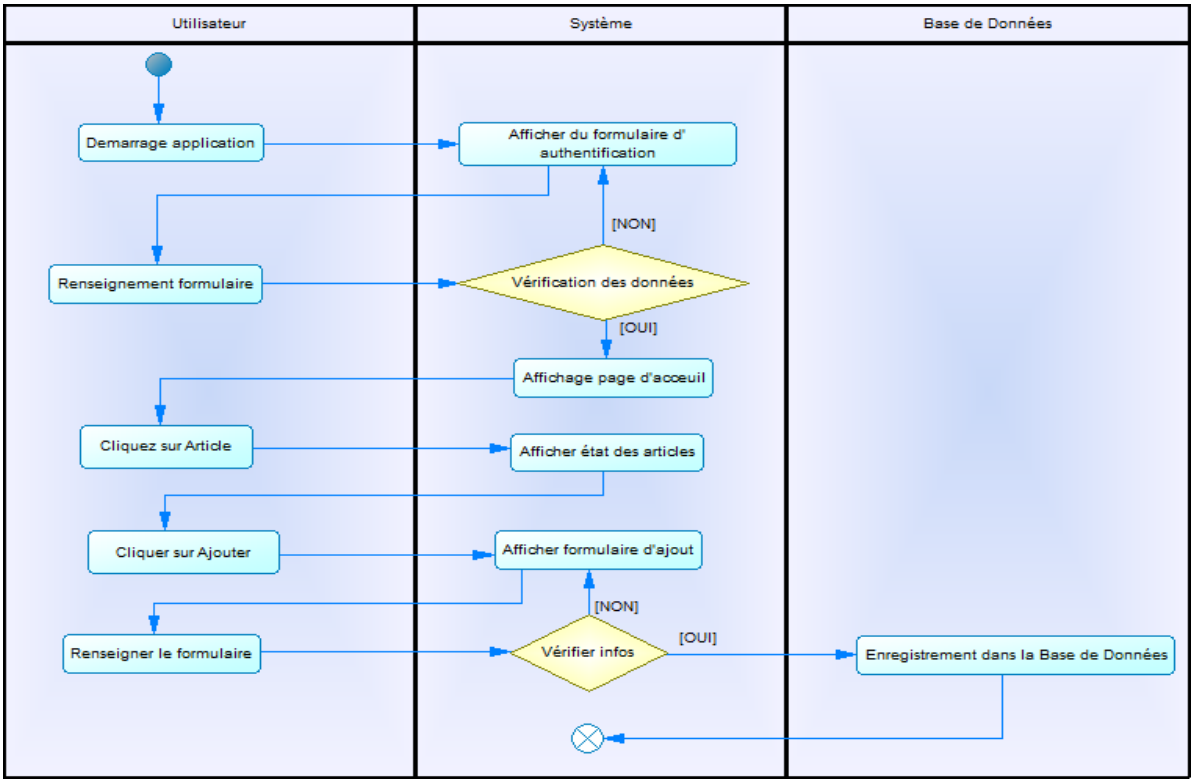

Figure 14: Diagramme d'activité "Ajouter Article"

iii. Diagramme d'activité « Modifier Catégorie »

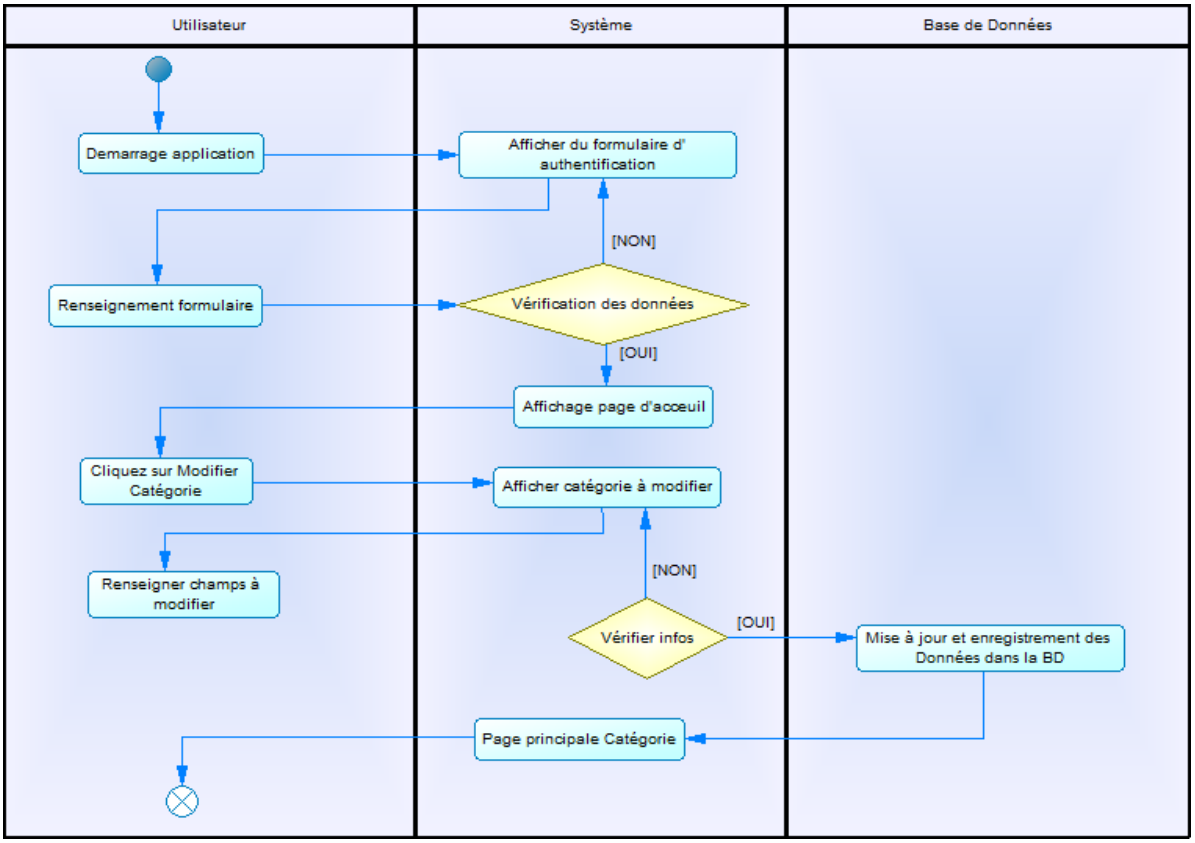

Figure 15: Diagramme d'activité "Modifier Catégorie"

CONLUSION

Après la rédaction du cahier des charges qui nous a permis présenté dans la première partie qui nous a ouvert les portes pour une analyse adapté au système de fonctionnement et répondant aux besoins. Nous avons dans cette partie commencée par déterminer la méthode d'analyse adaptée pour à la fois répondre aux besoins et regroupé toutes les fonctionnalités du système. Pour ce faire, nous avons choisi ***langage de modélisation UML,*** et avons opté pour ***le processus Unifié 2TUP*** de ***UML 2.0.*** *2TUP* nous offre une approche orienté objet en *trois (03) branches* suivant une architecture en ***Y***. Nous pouvons dire de tout cela que, cette seconde partie nous a permis de présenter *le rôle de chaque acteur dans le système au travers des diagrammes des cas d'utilisation*, et encore bien mieux, de décrire *les enchaînements de chaque rôle avec le système du début à la fin à travers les diagrammes d'activités.* A la suite de cette partie qui portait sur la modélisation de la *branche de gauche du processus* ***2TUP,*** présenterons la partie *conception*.

DOSSIER DE CONCEPTION

Résumé :

Le dossier de conception est un document permettant de présenter l'aspect technique de l'application qui sera réaliser, de ressortir son architecture et aussi l'ensemble des composants nécessaire à son bon fonctionnement.

Aperçu :

INTRODUCTION

Une fois l'analyse de notre système d'information terminé, les fonctionnalités et attentes dégagées et la description textuelle et graphique des scénarii présentée, nous pouvons sans plus tarder prendre en main la conception de notre application. Concevoir une application de gestion se résume principalement comme suit : *concevoir une base de données* ***stable et fiable,*** *concevoir une interface d'administration de la base de données* 'qui représente notre application) et enfin *déployer l'application.* Ces étapes se font de manière hiérarchique et ordonnée ayant chacune un ensemble d'étape à suivre afin d'être réalisée. Le dossier de conception permet de décrire les dépendances des divers composants (logiciels ou matériels) utilisées pour implémenter notre système. Ce dossier présente et documente l'architecture du système. Dans cette partie, le travail sera architecturé comme suit :

- *Spécifications techniques ;*
- *Architecture physique du système ;*
- *Architecture technique de l'application*
- *Conception détaillée de l'application.*

I. SPECIFICATIONS TECHNIQUES

En informatique, ***l'architecture*** désigne la structure inhérente d'un système informatique, l'organisation des *différents éléments du système (logiciels, matériels, humains et/ou informations)* et *les relations existantes entre ces éléments*. L'architecture des systèmes informatiques est représentée sous forme graphiques tels que les diagrammes entités associations. Il peut y avoir plusieurs diagrammes d'architectures pour un même système, tels que : l'architecture des informations, l'architecture métier, l'architecture technique et l'architecture applicative. La technologie objet requiert une architecture adaptée et fiable. C'est cette architecture qui organise les interactions entre objet. Il est courant que l'on regroupe ces objets en classes, ces classes en domaines, et ces domaines en couches. Ces couches permettent de présenter l'architecture de l'application. Ainsi, si modéliser est indispensable, construire une architecture en couche est un critère de qualité dans le cadre d'un développement Objet. Reste donc à choisir le nombre de couches de l'application ainsi que le contenu de chacune d'elle.

II. PRESENTATION DE L'ARCHITECTURE DE L'APPLICATION

En ce qui concerne les couches, nous allons les présenter pour une bonne compréhension dans un tableau avec des exemples pour chacune d'elle.

Tableau 5: Tableau descriptif de l'architecture de notre application

Description des couches	Exemple de GSR
La couche DAO, est en charge de la gestion des relations avec les sources de données, quelles qu'elles soient. Elle offre des services d'accès, de recherche, de création, de mise à jour, de suppression de données stockées dans une base de	utilisation d'une base de données relationnelle

données. Cette couche s'appuie sur une interface standard qui est **JPA** (Java Persistence API) qui permet de masquer l'implémentation réelle du moteur de persistance qui est Hibernate.	
La couche service, permet l'implémentation de l'ensemble de la logique métier de l'application, indifféremment des sources de données utilisées et de la présentation. Elle s'appuie sur les couches DAO et model pour effectuer des opérations CRUD (Create, Research, Update, Delete) sur des objets persisté et leur appliquer ensuite des traitements métier.	**Objet** : Produit, Fournisseur, Administrateur, etc. **Service** : commande fournisseur, mouvement produit, vente, etc.
La couche Web, c'est la couche à laquelle a accès le client et les personnels du restaurant. C'est dans cette couche que sont créés les différents contrôleurs et les vues qui seront affichées à l'utilisateur dans un navigateur web par l'intermédiaire d'un serveur web comme TomCat.	Service interne de l'application : authentification du personnel; Service en ligne de l'application : accès aux offres et menu du restaurant; requête client : seront gérons gérer par le protocole HTTP et l'affichage par HTML
La couche sécurité : c'est au niveau de cette couche que nous allons gérer la sécurité de notre application, c'est à dire l'authentification des utilisateurs. elle est intégrée au sein de la couche web.	Authentification des utilisateurs (personnel)
La couche utilisateur, c'est la couche qui est chargée de gérée l'interface utilisateur. C'est à partir de cette interface que l'utilisateur peut effectuer ses opérations.	Service en ligne : interface commande utilisateur (http, Servlet)

III. FIGURE DE L'ARCHITECTURE TECHNIQUE DE GSR

Dans cette section il sera question de présenter le principe architectural et l'aspect conceptuel de la présente application. En effet comme l'implémentation de la présente application sera effectuer sur une plateforme web, le choix de l'architecture de l'application se porter sur l'architecture en couche, dont le principe repose sur le fait que chaque couche ne traite de manière autonome qu'une partie bien précise du système, un ensemble de fonctionnalités bien définies dans un cadre de responsabilités restreint. Ainsi avec une telle architecture, chacune des couches publie ensuite ses services spécialisés à destination d'une autre couche du système de niveau supérieur. Chaque couche se concentre donc sur ses préoccupations propres (accès aux données, logique métier,

Présentation, etc.) et fait appel à une ou plusieurs couches de niveau inférieur lorsqu'elle sort de son secteur de responsabilités. La structuration en couche de ce modèle est présentée ci-dessous :

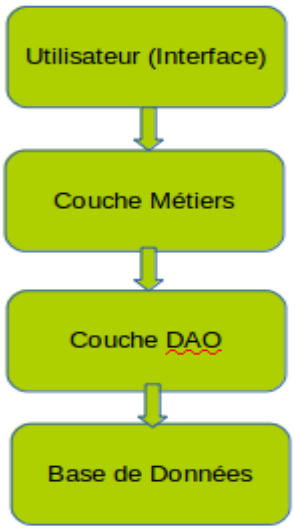

Figure 16: Architecture en couche

Ainsi pour mettre cette architecture, nous avons utilisés le Framework Spring. Le choix de ce dernier s'explique par le fait qu'il nous permet de garantir le respect strict de la séparation des couches applicatives comme précisé ci-haut. De plus la gestion de la correspondance des Interfaces et Implémentations est géré par Spring grâce à une configuration XML. En outre Spring dispose d'un module nous permettant la gestion de la sécurité de l'application sans problème.

L'image ci-dessous présente l'architecture de notre système à l'aide du Framework Spring.

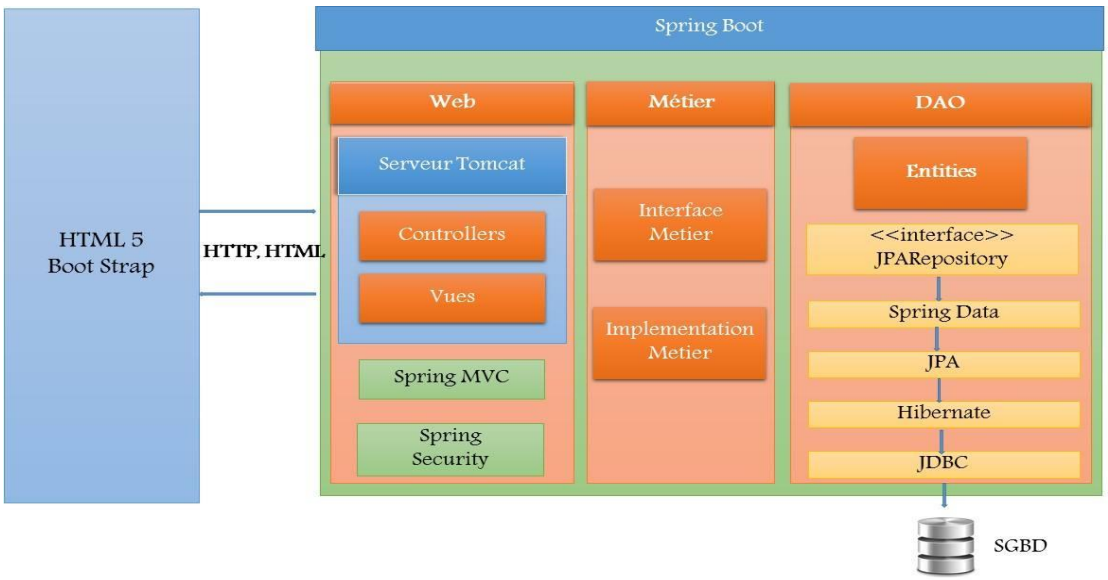

Figure 17: Architecture technique du système GSR

Explication complémentaire sur l'architecture choisie.

Au démarrage de l'application, c'est Spring qui démarre en premier, il va gérer les différentes couches de l'application en faisant l'injection des dépendances. Il y a un serveur de base de données qui est MySQL.

- Les données seront stockées dans un SGBD relationnel MySQL.
- L'application sera composée de quatre (04) couches
  - La couche Utilisateur ;
  - La couche DAO, basée sur Spring Data, JPA, Hibernate et JDBC ;
  - la couche Métier ;
  - La couche Web, basée sur le modèle MVC.
- La sécurité, sera assurée par Spring Security.

IV. CONCEPTION DETAILLEE DE L'APPLICATION

La conception détaillée est la partie ou l'on représente les éléments intervenant et les actions que ces derniers émettent entre eux avec/dans le système. Dans cette partie, nous présenterons le diagramme de classe de l'application et les séquences de quelque processus.

1. Diagramme de classe

Le diagramme de classe est considéré comme le plus important de la Modélisation Orienté Objet. Il présente la structure interne du système modélisé. Le diagramme de classe permet de fournir une représentation abstraite des objets du système qui vont intervenir ensemble pour réaliser les cas d'utilisations. Les principaux éléments du diagramme de classe sont : les classes et les relations (Généralisation, dépendance, associations etc.).

a. Formalisme du diagramme de classe

Le diagramme de classe le plus simple présente deux classes et une association comme le montre la figure :

Figure 18: Formalisme simplifiée du diagramme de classe

Il est tout de même possible qu'un diagramme de classe présente une ou plusieurs classe-association. Classe-association possède les mêmes propriétés qu'une classe et une association. Elle se connecte à deux ou plusieurs classes. Et est caractérisée par un trait discontinu entre les classes et l'association.

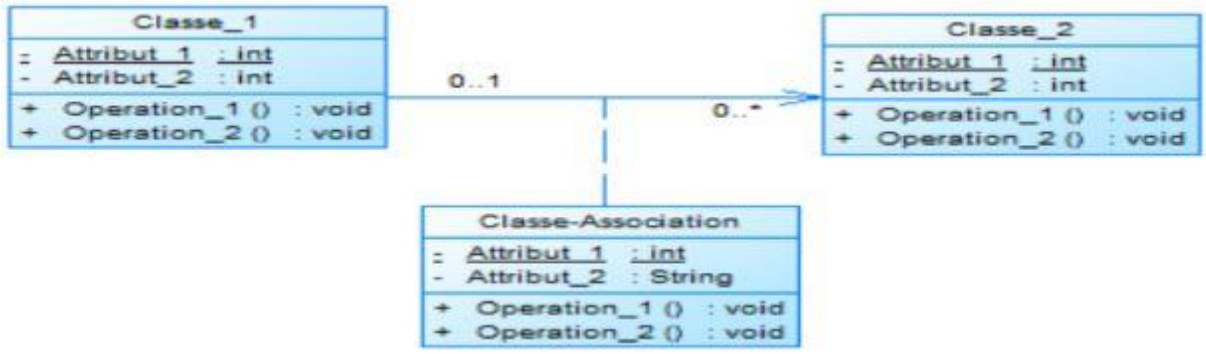

Figure 19: Formalisme évolué du diagramme de classe

b. Diagramme de classe de notre futur système

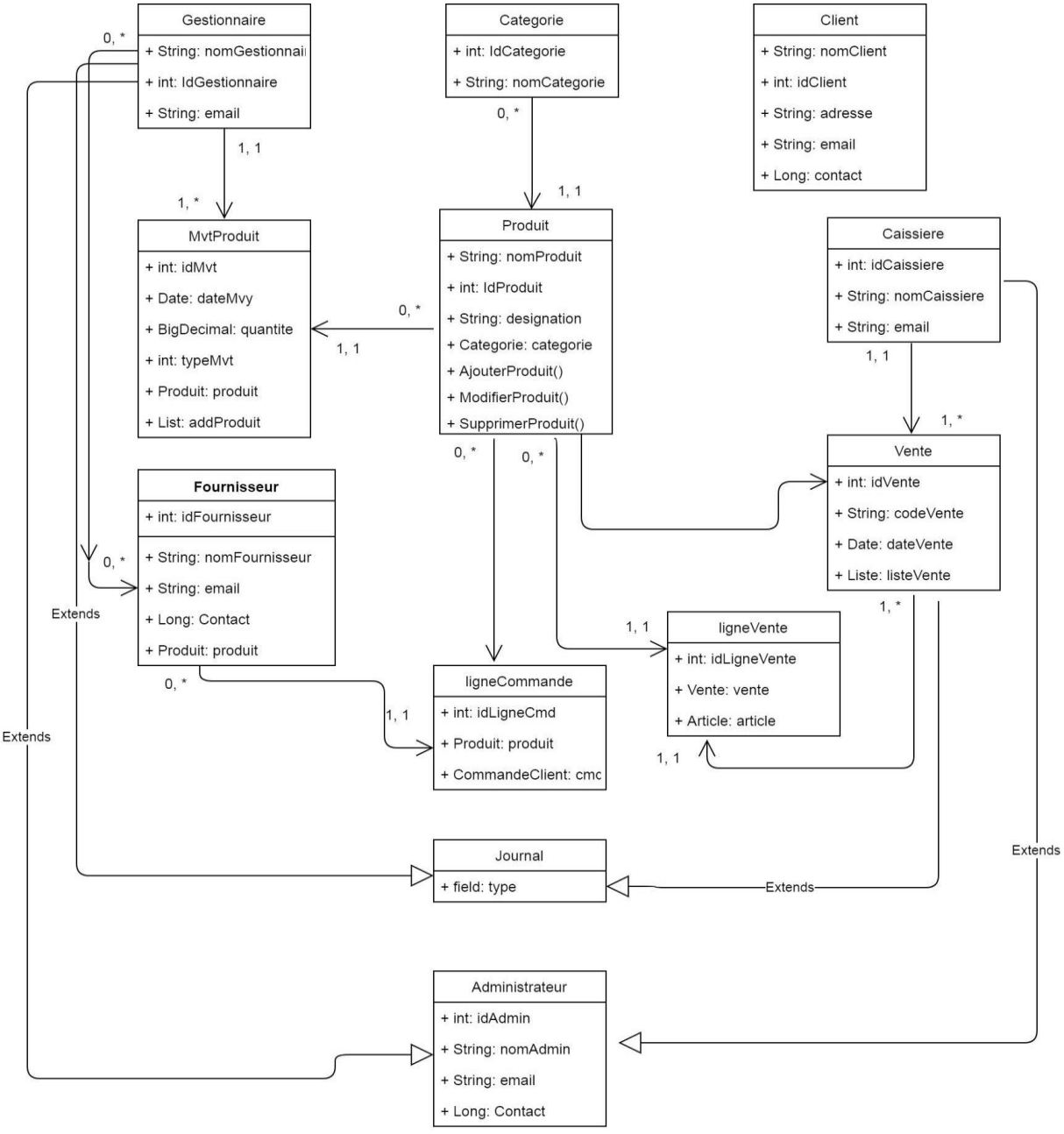

Figure 20: Diagramme de classe du système

Présenté ainsi, nous pouvons sans soucis de se tromper connaitre le rôle de chaque classe et surtout les relations entre elles.

2. Diagramme de séquence

Le diagramme de séquence permet de représenter les interactions entre les objets en précisant la chronologie des échanges de message. Son but principal est de donné une description chronologique sur le déroulement des d'utilisation entre les acteurs ou les objets.

a. Elément du diagramme

Les éléments principaux de ce diagramme sont :

- **L'objet et/ou la classe** qui peut être représenté comme un acteur s'il d'un acteur (un bonhomme), ou par un rectangle portant le nom de l'objet si n'est pas un acteur.
- **La ligne de vie**, illustrée par une ligne discontinue connectée à un élément Elle représente l'existence de cet élément dans le temps.
- **La communication**, celle-ci est illustrée par une flèche horizontale continue, allant de la ligne de vie de l'émetteur vers la ligne de vie du récepteur. Elle est étiquetée du nom de l'opération à exécuter. Une communication indique l'émetteur envoie un message au récepteur.

b. Formalisme du diagramme de séquence

De manière générale, le diagramme de séquence se présente comme suit :

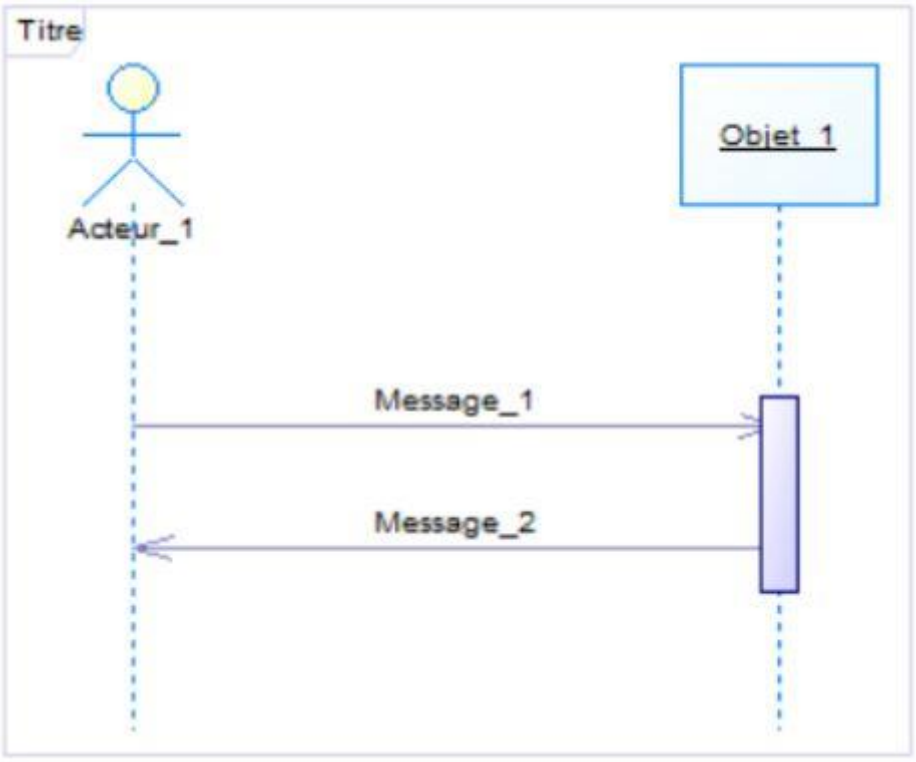

Figure 21: Formalisme du diagramme de séquence

c. Représentation de quelque diagramme de séquence

i. Diagramme de séquence « Authentification »

Le Scénario : Tout utilisateur saisi son Login et le mot de passe. Le système exécute la requête en vérifiant les éléments saisi (types de données). Si les informations saisies sont correctes alors, le système affiche la page d'accueil à défaut un message le système retourne le formulaire d'authentification.

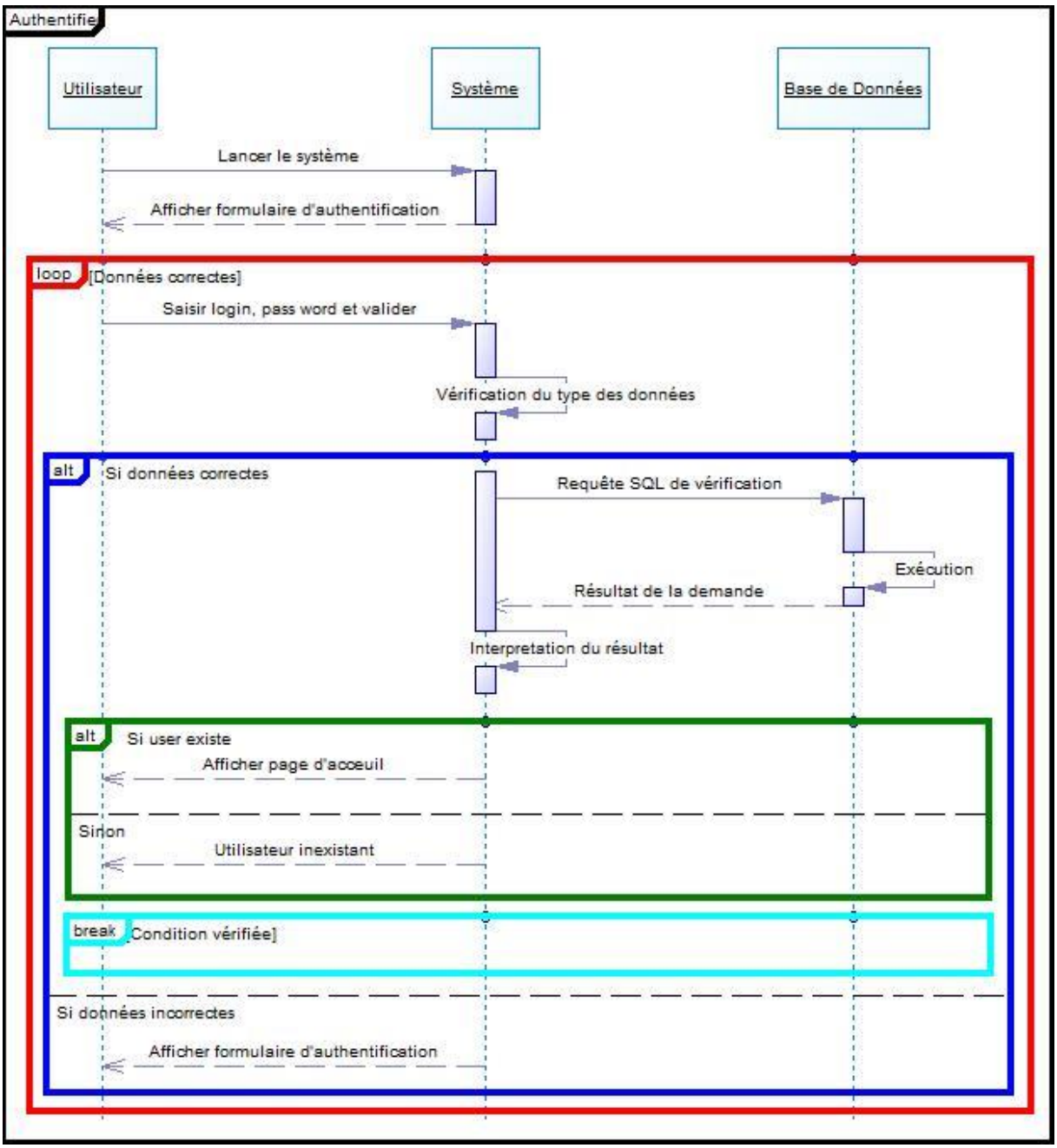

Figure 22: Diagramme de séquence "Authentification"

ii. Digramme de séquence « Modifier un Utilisateur »

Le Scénario :

- L'utilisateur est connecté ;
- L'utilisateur sélectionne l'option modifier d'un utilisateur enregistré ;
- Le système affiche le formulaire lié à la modification de cet utilisateur ;
- L'utilisateur modifie les champs du formulaire concernés et valide.

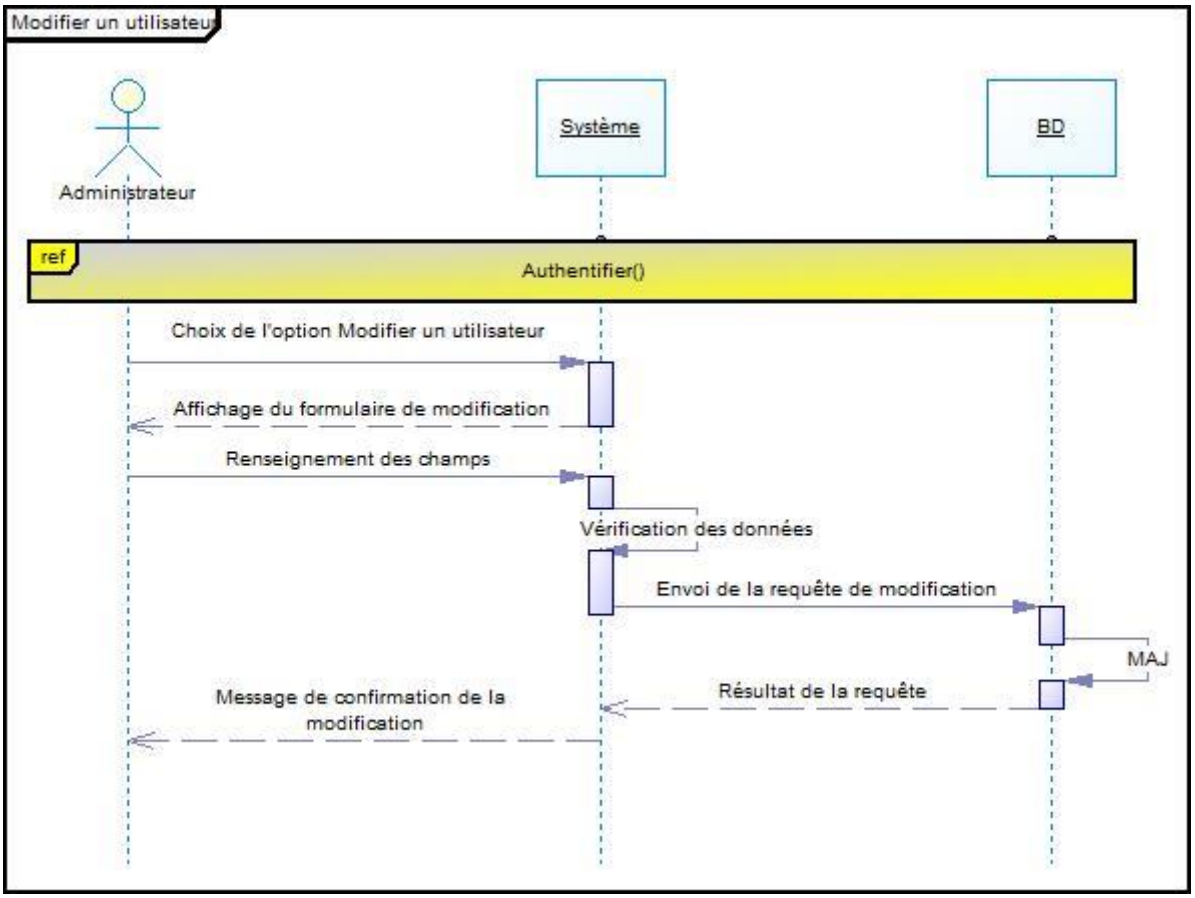

Figure 23: Diagramme de séquence "Modifier utilisateur"

iii. Diagramme de séquence « Ajouter un Fournisseur »

Le Scénario :

- L'administrateur ou le gestionnaire est connecté.
- L'administrateur clique sur le menu Fournisseur.
- Le système affiche la liste des Fournisseur existant dans le système ainsi que les opérations possibles.
- L'Administrateur fait le choix de l'opération c'est-à-dire clique sur « Ajouter »
- Le système affiche le formulaire d'Ajout. Si le formulaire est bien renseigné, le système vérifie ;
- Si l'espace d'hébergement et ce dernier existe alors le système charge les informations du Fournisseur et fait directement la mise à jour dans la BDD.

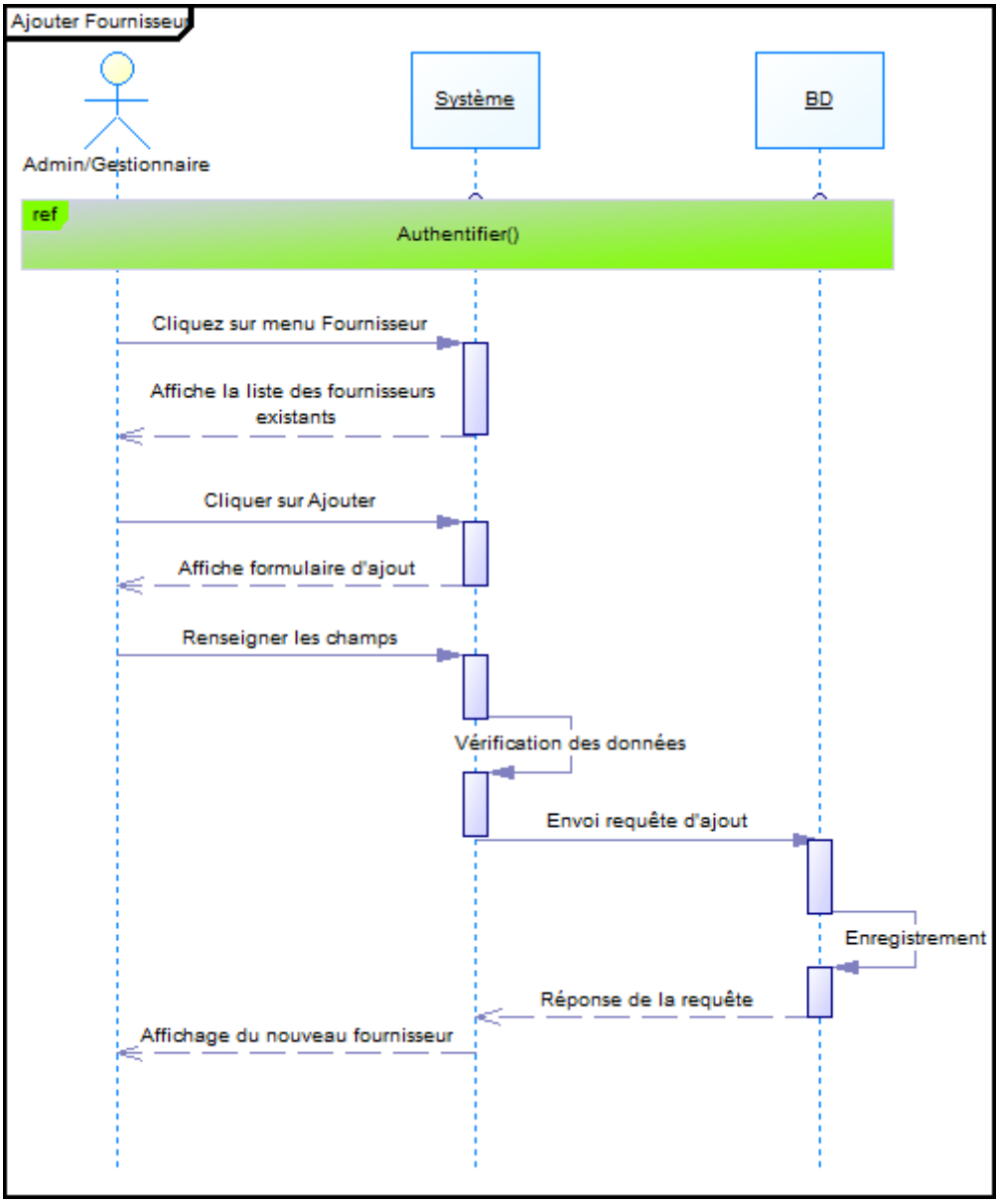

Figure 24: Diagramme de séquence "Ajouter Fournisseur"

iv. Diagramme de séquence « Supprimer »

Le Scénario : L'Utilisateur est connecté
L'Utilisateur clique l'option supprimer.
Le système affiche la boîte de dialogue demandant la confirmation de la suppression.
L'Utilisateur valide si l'action est intentionnelle au cas contraire, il annule l'action.

Cette opération est valable pour toutes les fonctionnalités du système.

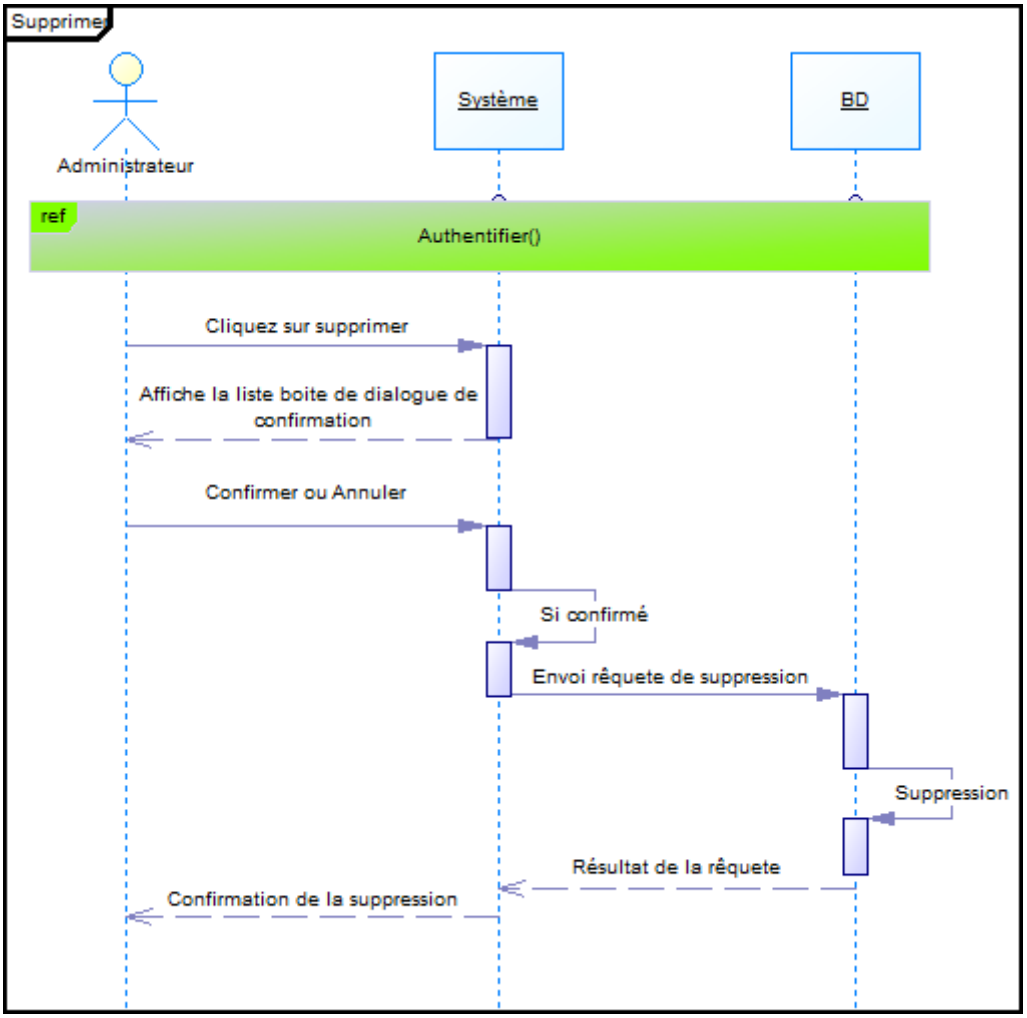

Figure 25: Diagramme de séquence "Supprimer"

CONCLUSION

Dans cette partie du travail, il était question pour nous de présenter respectivement les spécifications techniques de l'application, de présenter l'architecture technique appropriée pour le système à mettre sur pied et de faire une conception détaillée de ce-dernier. Nous pouvons retenir de tout ce qui précède que : le 3-tiers représente sans doute l'architecture la plus satisfaisante pour un déploiement les plus correct de l'application, grâce à sa structuration en trois couches (couche web, couche métier, DAO (couche accès aux données)) successive et adjacentes sur le plan de la communication. L'architecture technique nous présente donc de manière claire voyante les éléments matériels de chacune de ces couches, et leur dépendance réciproque. La présentation des classes et du diagramme de classe nous fais voire de manière encore superficielle la constitution de notre base de données. Pour donc éclaircir cette vue, nous allons passer à la réalisation.

DOSSIER DE REALISATION (IMPLEMENTATION) ET GUIDE UTILISATEUR

Résumé :

Le dossier de réalisation présente une finalisation du projet sur le plan financier, technique, applicatif et même structurel. Mais aussi l'évolution conceptuelle du système sollicité

Aperçu :

INTRODUCTION

Le système de gestion du restaurant ainsi conçu vient faciliter le *suivi, le management et la traçabilité* du stock, de la clientèle, des fournisseurs, etc du restaurant. Il est clair que tous ses futurs utilisateurs n'auront pas une aisance dans l'utilisation de ce nouvel outil. Pour ainsi pallier à ce problème, nous présentons ici de prime à bord le résultat de l'implémentation, comment utiliser cette application sans nul besoin d'une documentation volumineuse quelconque (guide utilisateur). En annexe, nous allons présenter les tests effectués pendant de la réalisation de l'application.

I. IMPLEMENTATION

Pour la mise en place de notre système de Gestion de stock restaurant, qui est une application Java/Jee nous avons utilisé les outils et technologies suivantes :

- ✓ Langage Java (8) ;
- ✓ Spring (MVC, SECURITY, ...) ;
- ✓ Hibernate ;
- ✓ Serveur web: Apache Tomcat 8 (8.0.46) ;
- ✓ JSP/JSTL ;
- ✓ HTML, CSS, Bootstrap, JQuery ;
- ✓ IDE: Eclipse.

Le Modèle-Vue-Contrôleur (Model-View-Controller) (MVC) est une architecture et une méthode de conception qui organise l'interface homme-machine (IHM) d'une application logicielle. Cette architecture divise l'IHM en :

- ✓ Un modèle de données
- ✓ Un modèle de présentation
- ✓ Un contrôleur (logique métier, gestion des évènements, synchronisation).

Pour notre application nous disposons principalement de deux (02) rôles :

- ✓ Administrateur/ Gère tous les entités du système.
- ✓ Utilisateur/ Peut enregistrer les ventes, stocks et peut lister les clients, articles, fournisseurs, les commandes fournisseurs et commandes clients.

Environnement Matériel

Le développement de notre système a été réalisé sur des machines DELL Inspiron
De processeur Intel 3, 5
De mémoire vive de 6, 8Go.
De disque dur 512 Go.
Ecran 15 pouces.

Environnement Logiciel

Le développement de l'application se fait exclusivement sur des plateformes libres.
L'implémentation se fait sous des machines Windows 8.1 et aussi tester sur des machines Linux (Ubuntu), version 16.04, version Luna.

La sauvegarde des données

Nous avons enregistrées nos données dans une base de données MySQL d'où ses données ont été implémentées dans notre système et générée automatiquement lors du lancement de notre programme.

II. TESTS D'ACCEPTABILITES

1. Ecran de connexion

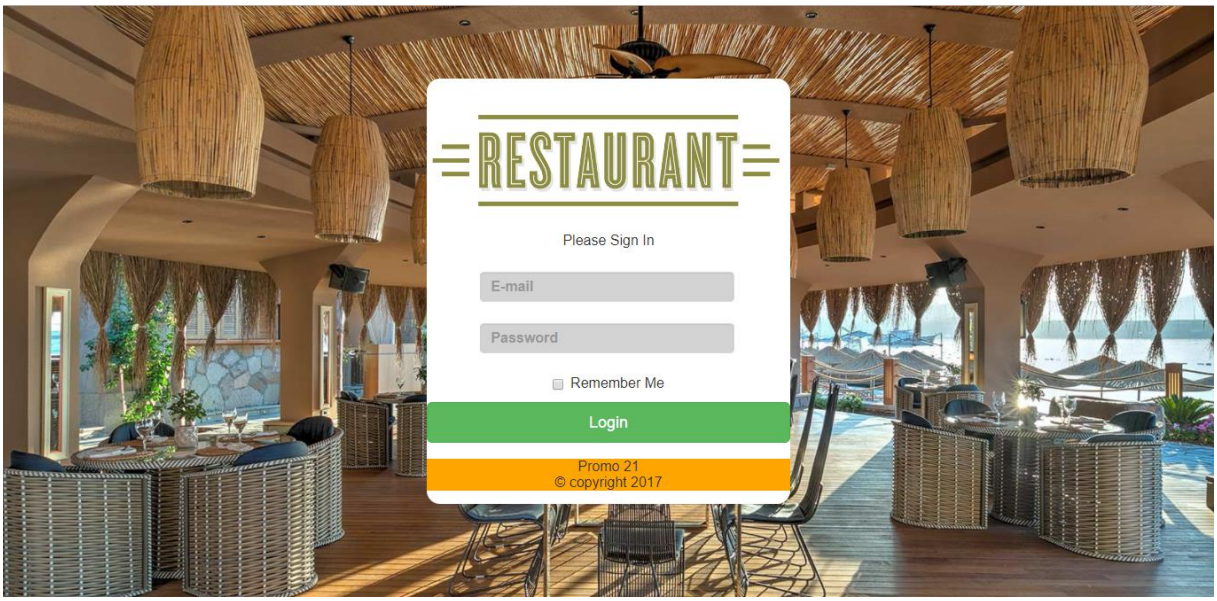

Figure 26: Page de connexion

2. Ecran d'accueil

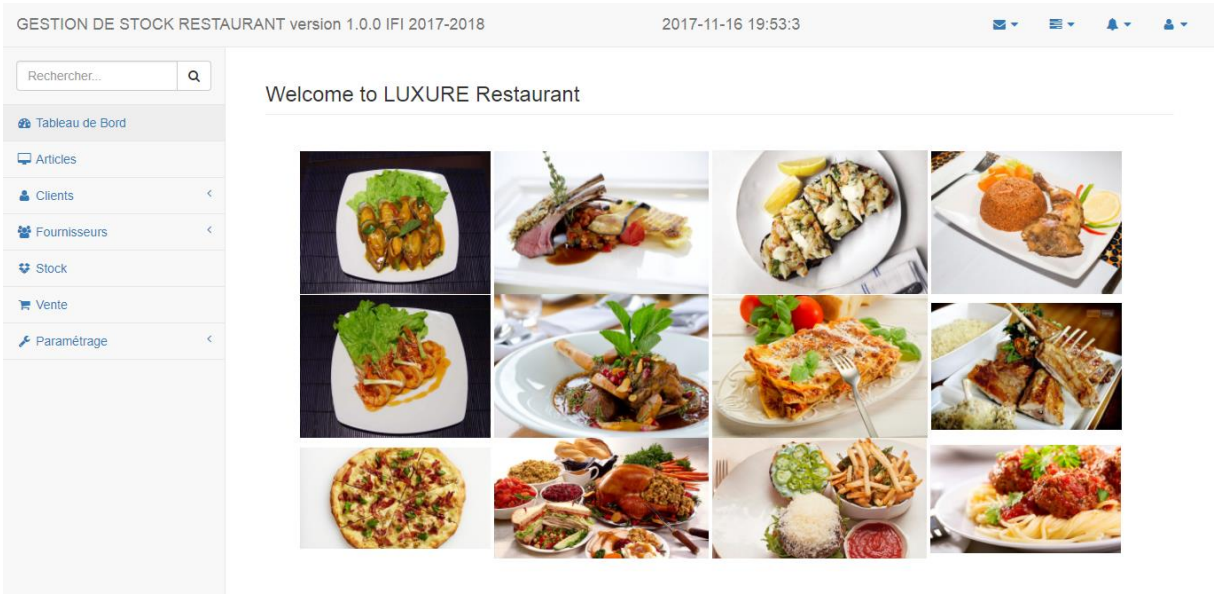

Figure 27: Page d'accueil

3. Module « Articles »

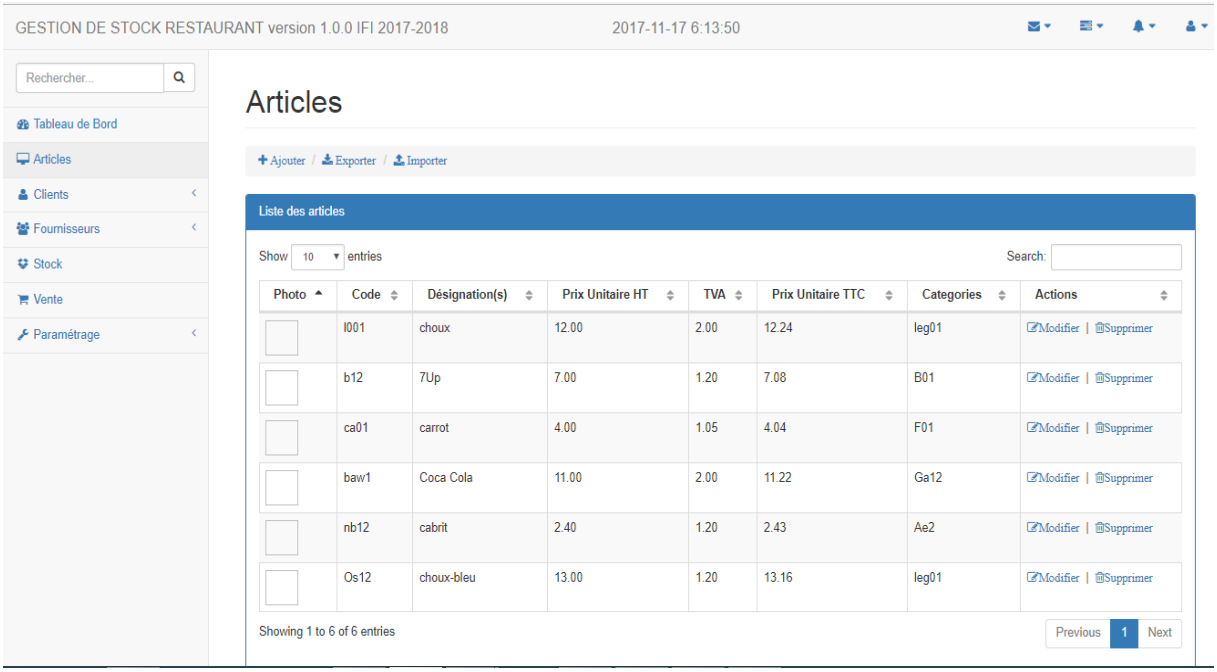

Figure 28: Page gestion des articles

4. Module « Clients »

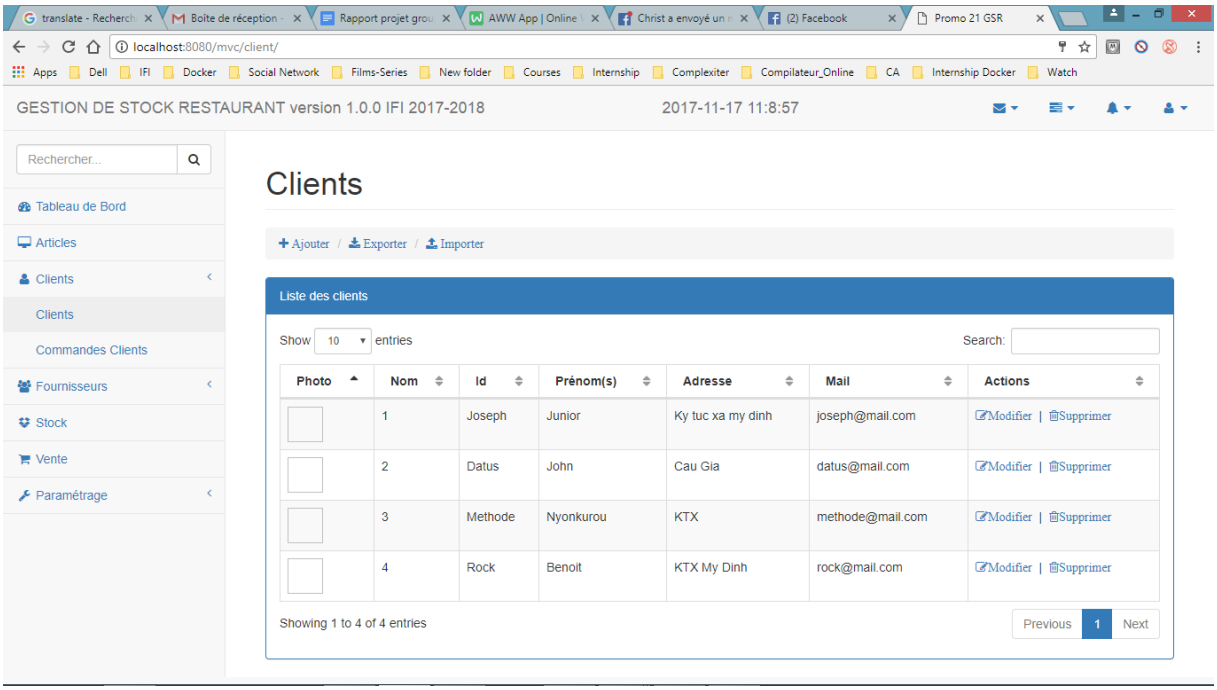

Figure 29: Page gestion client

5. Module « Fournisseur »

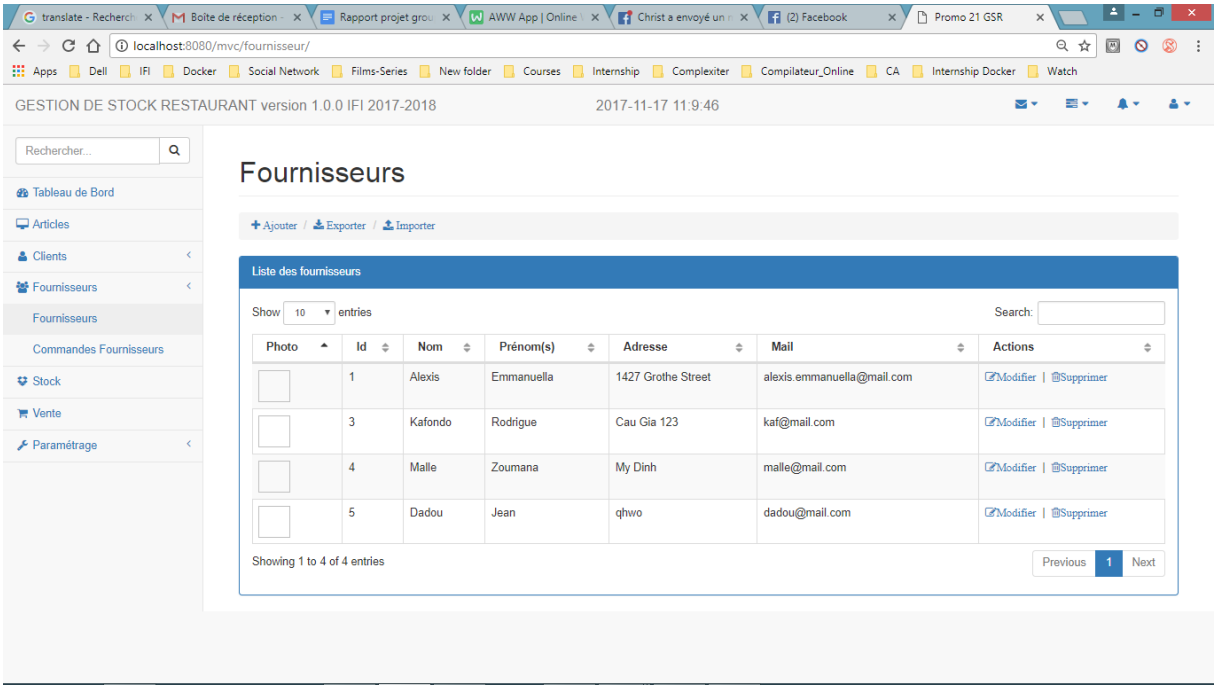

Figure 30: Page gestion fournisseur

6. Module « Stock »

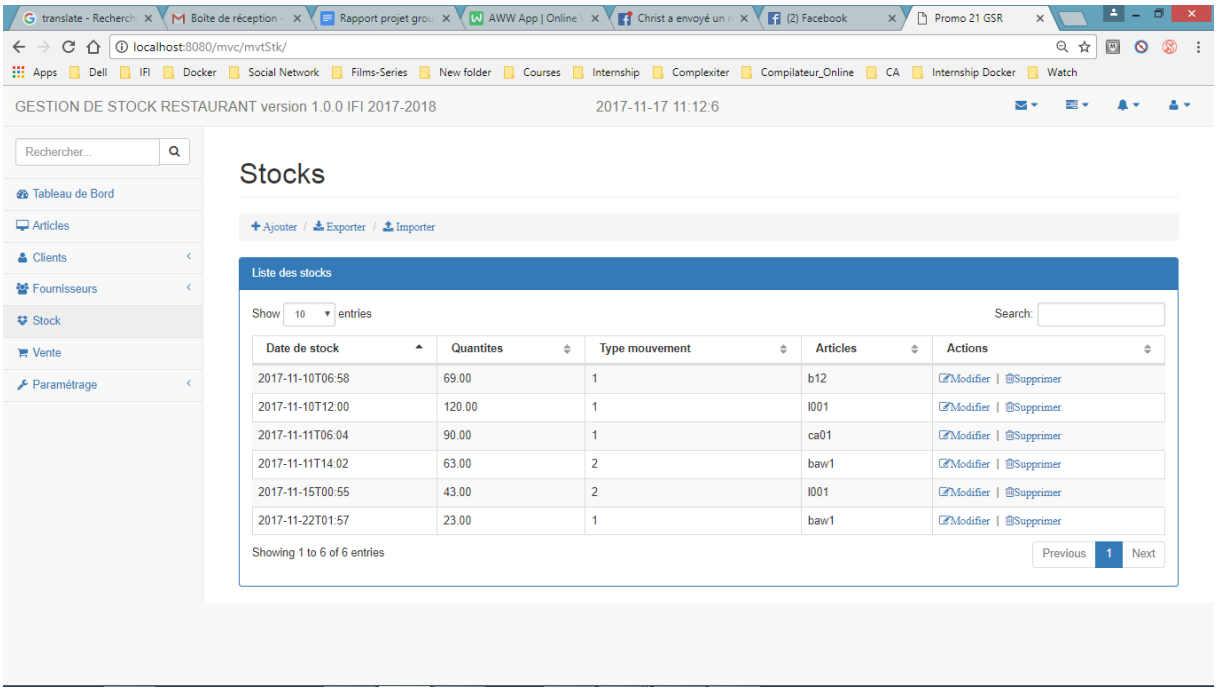

Figure 31: Page gestion stock

7. Module « Vente »

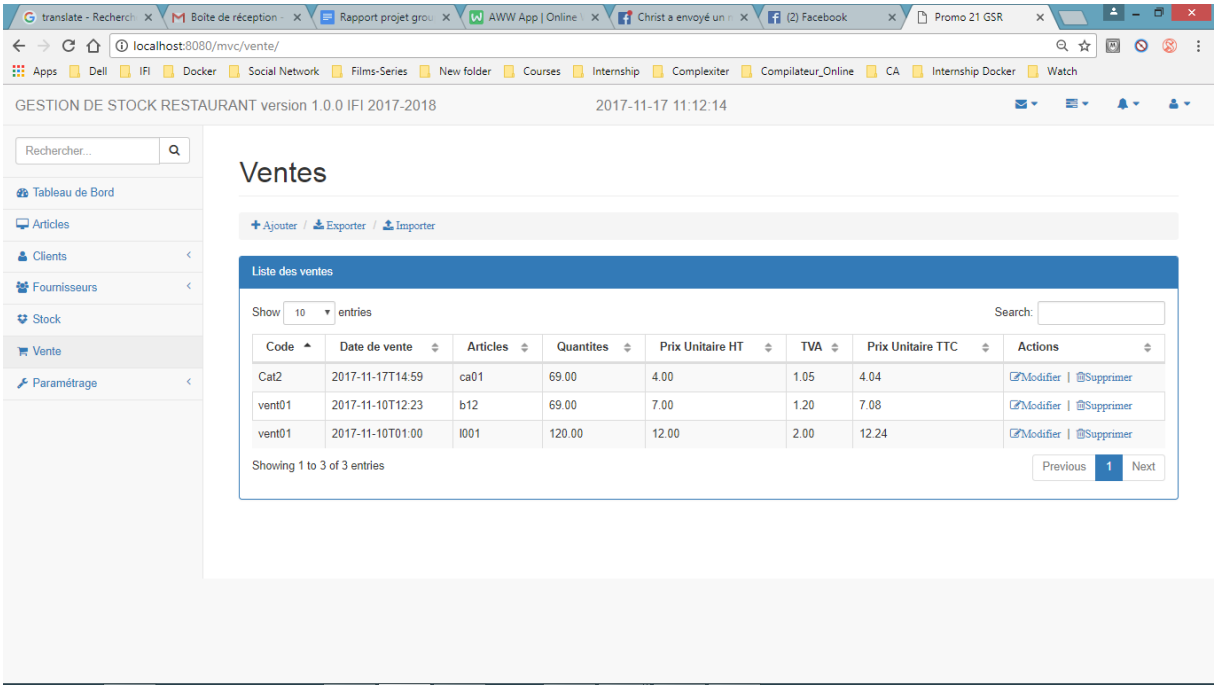

Figure 32: Page gestion des ventes

8. Module « Catégorie »

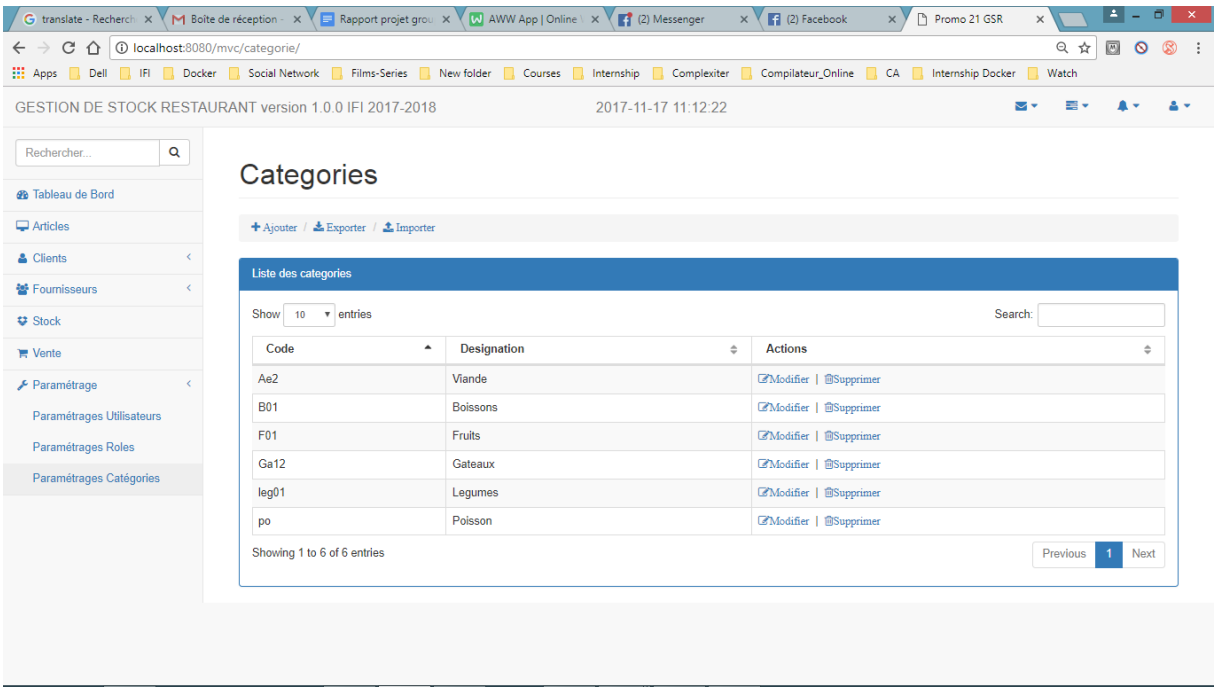

Figure 33: Page gestion catégorie

III. TESTS DYNAMIQUES ET STATIQUES

1. Tests unitaires avec JUnit

JUnit est un Framework ou un outil nous donnant la possibilité d'écrire et d'exécuter des tests unitaires sur des programmes implémentés en Java. Cependant, nous avons comme objectif pour notre projet d'écrire et d'exécuter des tests avec JUnit pour un ensemble de classes Java qui implémentent notre application GSR. Ceci à partir des spécifications données. Dans le document [3], les tests en **JUnit** sont regroupés au sein d'une classe Java qui doit hériter de la classe **TestCase**. Le nom des méthodes de test doit commencer par **test**. Le corps d'une méthode de test doit comporter trois parties :

- ✓ **le préambule**, qui permet l'initialisation des objets sur lesquels le test va être exécuté ;
- ✓ **le corps de test,** dans lequel la méthode à tester est appelée sur les objets créés ;
- ✓ **le post ambule,** qui permet de délivrer le verdict du test (succès ou échec) en vérifiant un ensemble de propriétés (assertions) sur l'état des objets après le test.

Dans le cas de notre application, nous allons écrire un cas de test qui nous permettra de confirmer que notre base de données a été belle et bien créée avec succès.

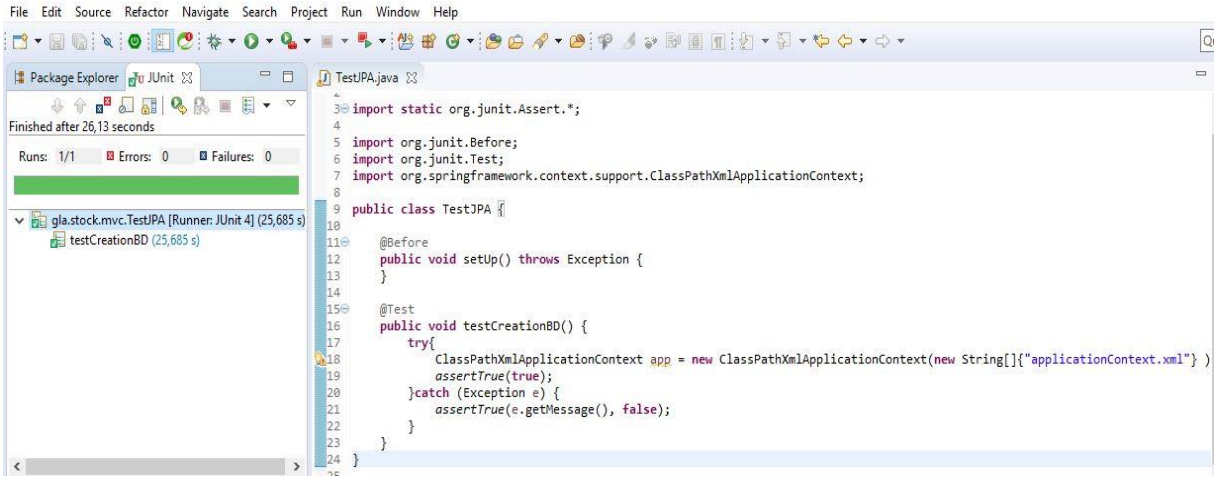

Figure 34: Ecran test de création de la base de données

Il est à noter qu'au démarrage de l'application, vous devez tout juste vous rassurez que votre SGBD est démarré et l'application ajoutée dans le serveur d'application utilisé. Si tel est le cas, lancer l'application et la base de donnée est automatiquement créée grâce cette instruction de la ligne 13 du fichier « data-source-config.xml » se trouvant dans « src/main/resources » dans le package « config ».

2. Test Statique

a. Test statique avec éclipse métrique

Nous présentons dans cette section le test statique que nous avions effectuée a l'aide de l'outil éclipse métrique. Ce test a pour but de vérifier si notre programme respecte tous les principes et règles de la programmation à savoir : le nombre de variable utilise, le nombre de ligne de code, le nombre de fonction, la complexité cyclomatique, le principe d'unicité de la classe, etc. l'image ci-dessous présente le résultat de notre test effectuée avec cet outil.

Metric	Total	Mean	Std. Dev.	Maxim...	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		0,98	0,772	3	/ProjectGSR/src/main/java/gla/stock/mvc/controller...	enregistrerMvtStk
Number of Static Attributes (avg/max per type	4	0,08	0,337	2	/ProjectGSR/src/main/java/gla/stock/mvc/entities/M...	
Efferent Coupling (avg/max per packageFragm		7,625	5,544	14	/ProjectGSR/src/main/java/gla/stock/mvc/dao/Imp	
Specialization Index (avg/max per type)		0	0	0	/ProjectGSR/src/main/java/gla/stock/mvc/LoginCon...	
Number of Classes (avg/max per packageFragr	50	6,25	5,911	14	/ProjectGSR/src/main/java/gla/stock/mvc/dao/Imp	
Number of Attributes (avg/max per type)	87	1,74	2,18	8	/ProjectGSR/src/main/java/gla/stock/mvc/entities/Ar...	
Abstractness (avg/max per packageFragment)		0,25	0,433	1	/ProjectGSR/src/main/java/gla/stock/mvc/dao	
Normalized Distance (avg/max per packageFra		0,213	0,333	1	/ProjectGSR/src/main/java/gla/stock/mvc/entities	
Number of Static Methods (avg/max per type)	0	0	0	0	/ProjectGSR/src/main/java/gla/stock/mvc/LoginCon...	
Number of Interfaces (avg/max per packageFra	26	3,25	5,651	14	/ProjectGSR/src/main/java/gla/stock/mvc/dao	
Total Lines of Code	2247					
Weighted methods per Class (avg/max per typ	377	7,54	5,797	18	/ProjectGSR/src/main/java/gla/stock/mvc/controller...	
Number of Methods (avg/max per type)	305	6,1	4,88	17	/ProjectGSR/src/main/java/gla/stock/mvc/entities/Ar...	
Depth of Inheritance Tree (avg/max per type)		1,24	0,427	2	/ProjectGSR/src/main/java/gla/stock/mvc/dao/Imp/...	
Number of Packages	8					
Instability (avg/max per packageFragment)		0,713	0,385	1	/ProjectGSR/src/main/java/gla/stock/mvc	
McCabe Cyclomatic Complexity (avg/max per		1,236	0,87	7	/ProjectGSR/src/main/java/gla/stock/mvc/controller...	enregistrerClient
Nested Block Depth (avg/max per method)		1,187	0,639	5	/ProjectGSR/src/main/java/gla/stock/mvc/controller...	enregistrerClient
Lack of Cohesion of Methods (avg/max per typ		0,252	0,383	0,933	/ProjectGSR/src/main/java/gla/stock/mvc/entities/Ar...	
Method Lines of Code (avg/max per method)	494	1,62	2,923	24	/ProjectGSR/src/main/java/gla/stock/mvc/controller...	enregistrerClient
Number of Overridden Methods (avg/max per	0	0	0	0	/ProjectGSR/src/main/java/gla/stock/mvc/LoginCon...	
Afferent Coupling (avg/max per packageFragn		12,5	18,748	55	/ProjectGSR/src/main/java/gla/stock/mvc/entities	
Number of Children (avg/max per type)	12	0,24	1,68	12	/ProjectGSR/src/main/java/gla/stock/mvc/dao/Imp/...	

Figure 35*: Résultat du test avec éclipse métrique*

Nous pouvons remarquer à travers l'état (tout en bleu) du résultat que notre programme respect toutes les contraintes des différentes métriques de cet outil, ce qui veux dire que notre programme respect les principes et règles de la programmation en Java. Nous pouvons aussi remarquer que les deux importantes métriques à savoir la complexité cyclomatique (qui permet de calculer la complexité du programme en comptant le nombre de condition) et la cohésion (qui est le principe ou une classe ne devrait avoir qu'une seule raison de changer) d'une classe ont tous une note moyenne ce qui est très intéressent parce qu'elles rendent la maintenance et la compréhension du code plus facilement.

b. Test avec PMD (Programming Mistake Detector)

Dans cette section nous présentons un autre outil très réputé dans l'analyse d'un code source. Il permet outre la détection des bugs, la détection des copier-coller (redondance) dans un programme.

Difficultés rencontrées :

Les principales difficultés rencontrées lors de la réalisation de ce projet sont plutôt d'ordre technique. L'une des difficultés qui nous a beaucoup affectée celle liée à l'environnement de travail plus précisément à l'outil (Framework Spring) utilisé pour l'implémentation de notre application. En effet c'est un nouveau outil pour nous c'était la première fois que nous l'avions utilisés ce qui a fait que nous avons eu du mal à le maitriser et a bien l'exploite pour réaliser notre application dans le délai qui nous a été impartis.

CONCLUSION

Dans le cadre du projet du cours de Génie Logiciel, nous avons travaillé sur la mise en place d'une application web pour la gestion du stock du restaurant de notre dortoir « KTX My Dinh ». Le présent projet a pour objectif principal de mettre en pratique nos connaissances en programmation web. Ainsi après avoir effectué une étude détaillée (étude du sujet, état de l'art et analyse) de notre projet, nous avons pu proposer une solution avec une architecture pour application.

Pour l'implémentation nous avons choisi le langage JAVA. Pour ce faire nous avons utilisé l'IDE Eclipse. Par la suite nous avons effectué différentes expérimentations. Les résultats sont assez satisfaisantes, car notre application a pu résoudre le problème majeur pour lequel elle est conçu c'est-à-dire la gestion du stock.

En somme nous pouvons dire que ce projet a été très instructif pour nous. En effet il nous a permis de mettre en œuvre nos connaissances en programmation JAVA pour mettre en place une application afin de résoudre un problème réel.

Perspectives

Les perspectives de notre application sont de permettre non seulement aux clients de pouvoir voir le menu en ligne mais aussi d'effectuer sa commande. De pouvoir gérer également la vente pour les commandes en lignes. De plus nous prévoyons également une amélioration de l'application en permettant à l'administrateur de voir toutes les opérations effectuées par un client, mais aussi de pouvoir faire l'inventaire des toutes opérations sur une période bien définie.

REFERENCES

[1]: Ho Tuong Vinh, « ***Présentation d 'UML*** » IFI – 2012. Pages 26, 32-42, 5-60.

[2]: Joseph Gabay**, « *UML 2 ANALYSE ET CONCEPTION* »** Mise en œuvre guidée avec études de cas**.** Dunod, Paris, 2008. Pages 32-58, 81, 105

[3] : Prof. Burkhart Wolff, D. Longuet, Lina Ye « ***Génie Logiciel Avancé*** : ***TP - Test unitaire avec JUnit***». Année 2011-2012. Page 1-3.

[4] : Pascal ROQUES, Franck VALLÉE ; UML en action « ***De l'analyse des besoins à la conception en java*** ». Groupe Eyrolles, 2003

https://www.modeliosoft.com/fr/ressources/exemples-de-diagrammes/diagrammes-de-classes-et-de-packages-uml.html