Résumé

Afin d'offrir à leurs clients un service de distribution de qualité et aussi maintenir un niveau concurrentiel élevé dans le marché, SNTL Damco a vue nécessaire d'optimiser les flux de transport.

C'est dans ce cadre que s'inscrit notre projet de fin d'études qui consiste en la réalisation d'une application de gestion du suivi des bons de livraison et aussi un outil de décision pour le choix du transporteur.

Pour la réalisation de ce projet nous avons adopté le processus de développement en Y (2TUP) qui favorise le travail en groupe et permet une meilleure gestion des risques. Pour la modélisation nous nous sommes basés sur le langage UML.

En ce qui concerne le développement, nous avons utilisé l'architecture J2EE avec les Framework JPA/Hibernate pour le mapping objet relationnel et Spring comme Framework transversal.

Abstract

In order to offer their customers a good service of distribution and also maintain a high competitive level in the market, SNTL Damco saw the optimization of the traffic flows as necessary.

It is in this context that our graduation project falls. It basically consists of developing an application that manages the tracking of delivery notes. It is also a decision tool for the choice of the transporter.

For this project, we have adopted the development process of Y (2TUP) which encourages teamwork and allows a better risk management. And for modeling this, we have based on the UML language.

As for the development, we have worked on an application following the J2EE architecture with JPA / Hibernate Framework for object-relational mapping. We have also used Spring as a transversal Framework.

Liste des tableaux :

Liste des figures :

Table des matières :

Introduction générale :

De nos jours les nouvelles technologies web ont provoquées un changement profond dans le mode de communication des organisations, elles facilitent la gestion et le partage des données et améliorent la productivité au sein de l'entreprise.

Une application Web est un logiciel applicatif manipulable grâce à un navigateur Web, placée sur un serveur et se manipule en actionnant des *widgets* à l'aide d'un navigateur Web, *via* un réseau informatique (Internet, intranet, réseau local, etc.).

La société nationale de transport et logistique (SNTL) numéro 1 de transport routier et de la logistique au Maroc s'allie à DAMCO l'opérateur logistique international et filière du groupe AP Moller-Maersk pour créer SNTL Damcologistics. Ce partenariat offrira au marché marocain une nouvelle offre logistique totalement intégrée ainsi que des solutions personnalisées.

Le principal problème du service de transport de la SNTL Damco est l'absence de tarification et de tableau comparatif du coût de transport par destination, la solution adoptée est un outil de décision sous Excel, mais vu l'absence de gestion et de fiabilité de cette solution, les dirigeants ont vu nécessaire d'informatiser cette solution.

Notre travail consiste en la création et la mise en place d'une application web permettant la gestion des voyages des camions et la gestion du cycle de vie d'un bon de livraison.

Ce travail est divisé sur 5 chapitres :

Le 1^{èr} chapitre décrira le contexte général du projet qui comporte une bref présentation de la société, la problématique et la solution proposée.

Le 2ème chapitredéfinira le processus de développement adopté et décriral'analyse fonctionnellequi définira les besoins et détaillerales aspects fonctionnels du futur système.

Le 3ème chapitre décrira l'analyse technique qui définira les besoins et les aspects techniqueset la mise en place de l'architecture de l'application.

Le 4éme chapitre comportera la partie conception et réalisation qui définira le modèle de conception du système.

Le 5éme chapitre présentera les outils de développement et les technologies choisies et aussi un aperçu de l'application.

Chapitre 1 : Contexte général du projet

1. Présentation de l'organisme d'accueil

1.1. DAMCO

DAMCO (ex-MaerskLogistics) est une filiale du groupe AP Moller-Maersk et fait partie du top 10 des prestataires de logistique à l'échelle internationale.Il emploie 10.500 personnes dans plus de 90 pays et exerce ses activités dans 120 pays.

Figure 1 : Damco dans le monde

DAMCO offre une large gamme de services :

- Gestion internationale du transport maritime, aérien et routier.
- Gestion de la chaîne d'approvisionnement.
- Entreposage et distribution.

DAMCO possède 30 ans d'expérience dont 14 ans de présence au Maroc. Son chiffre d'affaires tourne autour de 2 milliards de dollars. Il gère 1,1 million de m^2 d'entrepôts de par le monde. DAMCO est présent sur trois sites au Maroc et à partir de son bureau casablancais, il gère ses filiales en Algérie et en Tunisie.

La Société nationale du transport et de la logistique (SNTL) et le groupe Maersk créent une joint-venture SNTL DamcoLogistics qui opère au centre logistique SNTL à Mohammedia[Figure 2 : Centre logistique SNTL Mohammedia], ce centre est stratégiquement situéà proximitédu portde Casablanca etles routes principales et gère 100.000 m^2 d'entrepôts, ainsi il offre des espacesde rangementadaptésaux besoins du marché:

- Une aire pour le stockage des conteneurs.
- Des aires de stockage pour les produits en vrac.
- Des aires de stationnement des remorques.
- Un centre d´affaires, constitué de bureaux modulables et équipé de salles de réunion ainsi que des nouvelles technologies de l´information.

Figure 2 : Centre logistique SNTL Mohammedia

Nous avons effectué notre stage au centre logistique SNTL [Figure 2 : Centre logistique SNTL Mohammedia] au département informatique de Damco Maroc.

1.2. Chiffres clés

1.2.1. Chiffre d'affaires global

Durant l'exercice 2012, SNTL Damco a réalisé un chiffre d'affaires de 45 838 KDH à fin décembre 2012.

L'activité Warehouse a généré presque 2/3 du chiffre d'affaires (70%) suivi par Le Containers yard avec 16 % du chiffre d'affaires. La distribution quant à elle

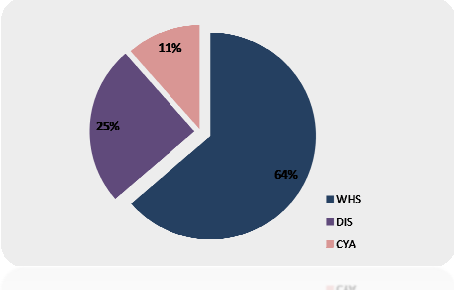


Figure 3 : Structure du chiffre d'affaires

contribue à hauteur de 14% au chiffre d'affaires 2012.

1.2.2. Chiffre d'affaires Distribution :

SNTL Damco enregistre sur le cours de l'exercice 2012 un chiffre d'affaires de 6,3 MDH pour l'activité distribution.

(FAGOR et LG Electronic ont contribué respectivement à 60,3% et 36,5% au CA distribution).

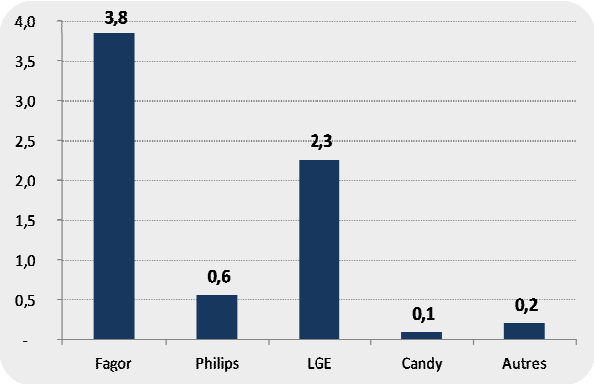


Figure 4 : Chiffre d'affaires par clients

2. Présentation du sujet

2.1. Problématique

L'activité principale de la société SNTL Damco est le stockage de marchandise et sa distribution vers les différentes villes du royaume à partir du centre logistique SNTL à Mohammedia, parmi les clients (Donneur d'ordre) bénéficiant de ce service on cite FAGOR, LG Electronics, PHILIPS Electronics, YAMAHA, NIKE …

Pour offrir à leurs clients un service de distribution de qualité et aussi maintenir un niveau concurrentiel élevé dans le marché, les dirigeants de la SNTL Damco ont vu nécessaire d'optimiser les flux de transport.

Après avoir effectué une étude du fonctionnement du service transport, les dirigeants ont constaté :

- L'absence de tarification du coût de transport par destination de SNTL Damco.
- L'absence d'un tableau comparatif du coût de transport par destination.
- La décision du transporteur se fait d'une manière manuelle.
- L'absence d'un suivi des états du bon de livraison.

La solution qui a été adopté pour résoudre ce problème est un outil de décision pour le choix du transporteur sous EXCEL [Tableau 1 : Solution sous Excel].

City	Price / m3	Volume	Income	CS	LORD	SNTLDAMCO	CM1			CM1 MAX	DECISION
							CS	LORD	SNTLDAMCO		
Agadir	250,61	30	7518,3	3700		3 152	3918,3	4018,3	4366,3	4366,3	SNTLDAMCO
Beni Mellal	99,26		0	1900	1 900	1 432	0	0	0	0	DECISION
Casablanca	39,46		0	600	800	649	0	0	0	0	DECISION
El Jadida	69,81		0	1000	1 200	1 000	0	0	0	0	DECISION
Essaouira	129,56		0		2 700	2 221	0	0	0	0	DECISION
Fès	119,5		0	2300	2 300		0	0	0	0	DECISION
Inzegane	250,61		0				0	0	0	0	DECISION
Kenitra	100,85		0	1200	1 300	1 172	0	0	0	0	DECISION
Khouribga	120,37		0	1250	1 300	1 267	0	0	0	0	DECISION

Tableau 1 : Solution sous Excel

La solution proposée [Tableau 1 : Solution sous Excel] est destinée à l'entrepôt de FAGOR, elle représente le prix/m^3pour chaque ville et pour les camions de type 14 Tonnes. La SNTL Damco fait appel à des prestataires (Transporteurs) en cas de non disponibilité des camions de sa flotte qui sont CS et LORD, il faut noter que chaque prestataire à un prix par ville sans tenir compte du prix/m^3.

La tarification du coût de transport par ville de SNTL Damco dépend :

- Des frais de gestion des camions :
  - Assurance
  - Vignette
  - Amortissement
  - Visite technique
  - Vidange
  - Pneus
- Des frais d'utilisation des véhicules :
  - Carburant
  - Réparation
  - Péage
- Des frais du conducteur :
  - Salaire net
  - Charges sociales(CNSS)
  - Prime de l'aïd
  - Cout de téléphone
  - Déplacement

L'entrepôt de stockage FAGOR est géré par différents employés. Parmi ces employés on distingue une personne qui gère les voyages des camions et une autre qui gère les commandes et les bons de livraisons.

Le chargé des bons de livraison reçoit une commande (Bon de préparation) de la part de FAGOR et calcule le volume global de la commande, ainsi le chargé des voyages saisi le volume global dans le fichier EXCEL [Tableau 1 : Solution sous Excel], la décision prise est effectuée selon le transporteur affichant un prix de revient maximal.

2.2. Solution

Vu le nombre important des voyages et des bons de livraison que gère les employées des entrepôts et les nombreux problèmes des fichiers EXCEL(manque de contrôle et de traçabilité, absence de fiabilité, les prix entre les donneurs d'ordre et les transporteurs sont confidentiels.). Les dirigeants ont ainsi prévus une application d'aide à la décision pour le choix du transporteur, et de la gestion des bons de livraison et de leur cycle de vie.

Pour répondre à un besoin réel et urgent en termes de contraintes fonctionnelles l'application doit permettre aux utilisateurs :

- La gestion des bons de livraison
- Le suivi du cycle de vie d'un bon de livraison
- La gestion des voyages
- Génération des décisions des voyages
- D'avoir une trace des voyages effectués

L'application à réaliser doit être portable, facile à maintenir, fiable et doit aussi offrir une vaste possibilité de déploiement, alors on a choisi la plateforme Java EE pour que l'application soit de qualité.

3. Conduite du projet

3.1. Processus de développement

Le processus de développement constitue un facteur déterminant dans la réussite d'un projet, du fait qu'il orchestre ses différentes phases et trace les principaux traits de sa conduite. Pour cela, le choix d'une méthode de développement doit être élaboré au préalable afin d'obtenir un produit de qualité tout en respectant les délais et les exigences fixés.

Pour ce faire, nous avons comparé 2 méthodes : RUP (Rational UnifedProcess) et 2TUP (2 TrackUnifiedProcess) qui sont des processus unifié itératif et incrémental. Cette comparaison nous a permis d'éliminer la méthode RUP puisqu'elle est lourde, coûteuse à personnaliser et destinée pour les projets mobilisant plus de 10 personnes.

Ensuite, nous avons opté pour la méthode 2TUP puisqu'elle couvre toutes les phases sans être très exhaustive et lourde, et fait une large place à la technologie et convient à des projets de taille quelconque.

Vu que nous ne disposions pas d'un cahier de charges précis et amplement détaillé, nous avons senti que la méthode 2TUP est plus appropriée puisqu'elle traite les projets selon deux axes différents : fonctionnel et technique, et donc nous permettra de répondre aux changements et aux spécifications de l'entreprise.

De plus, la méthode 2TUP va augmenter le taux de succès du projet car elle permet d'anticiper et de limiter les risques.

3.2. Planification

Conformément au processus de développementadopté, nous avons pu déterminer les différentes étapes du projet [Tableau 2 : Planification du projet].

Étape	Titre	Description de l'étape	Durée
Étape 1	Expression du besoin	Compréhension et l'évaluation du cadre duprojet	2 semaines
Étape 2	Analyse	Analyse fonctionnelle et technique	3 semaines
Étape 3	Conception	Définir finement le projet et organiser sa réalisation	4 semaines
Étape 4	Réalisation	Réalisation du projet tracé dans la partie conception	12 semaines

Tableau 2 : Planification du projet

La planification globale du projet est expliquée dans le diagrammede GANTT [Figure 5 : Planning du projet].

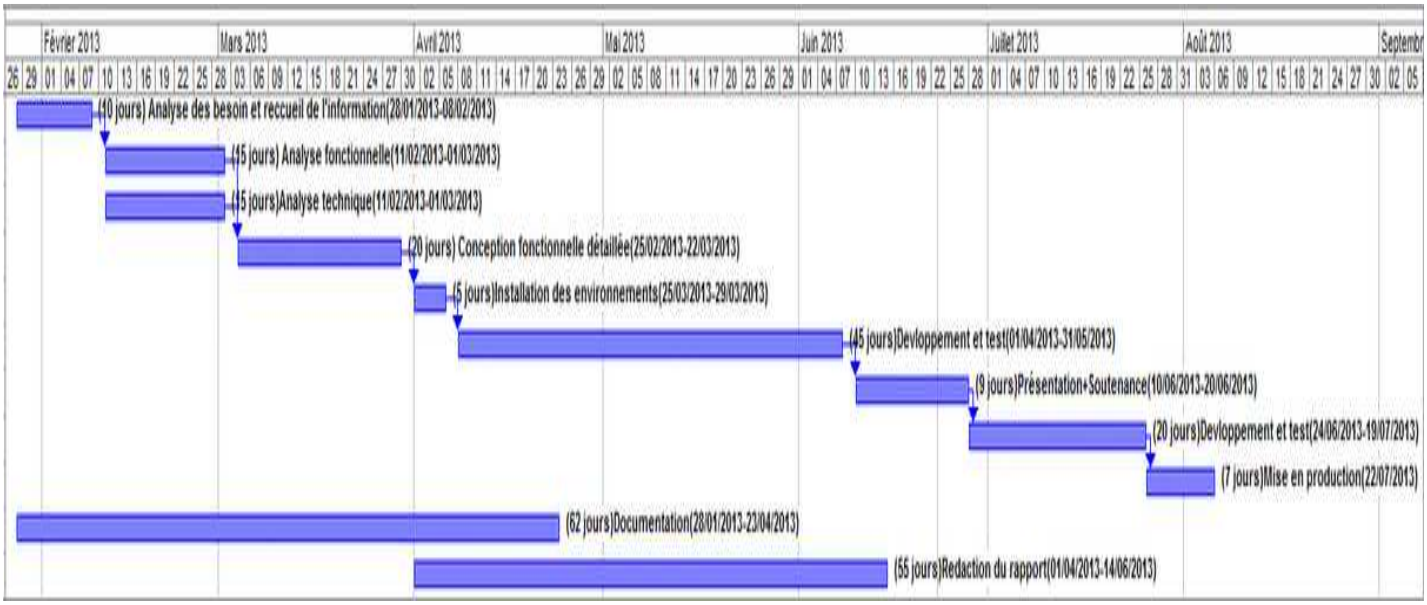


Figure 5 : Planning du projet

Chapitre 2 : Analyse Fonctionnelle

1. Méthode de développement (2TUP)

2TUP [Figure 6 : Approche 2TUP] est un processus de développement logiciel qui implémente le processus unifiébasé sur l'UML (UnifiedModelingLanguage). Il propose un cycle de développement qui sépareles aspects techniques des aspects fonctionnels en partant du constat que toute évolution peutse traiter parallèlement, suivant un axe fonctionnel et un axe technique. Ensuite, et enfusionnant les résultats de ces deux axes (branches), on arrive à réaliser le système désiré cequi nous donne un cycle de développement sous forme de Y [1].

1.1. Branche fonctionnel

Les étapes de la branche fonctionnelle sont :capture des besoins fonctionnels, spécification des besoins fonctionnels et analyse.

- Capture des besoins fonctionnels : cette phase a pour objectif de définir la frontière fonctionnelle entre le système et son environnement.
- Analyse : consiste à étudier précisément les spécifications fonctionnellesde manière à obtenir une idée de ce que va réaliser le systèmeen terme de logique métier [2].

1.2. Branche technique

Les étapes principales de la branche technique se présentent comme suit:

- La capture des besoins techniques: Cette étape recense toutes les contraintes sur les choix technologiques 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).
- La conception générique: 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 [2].

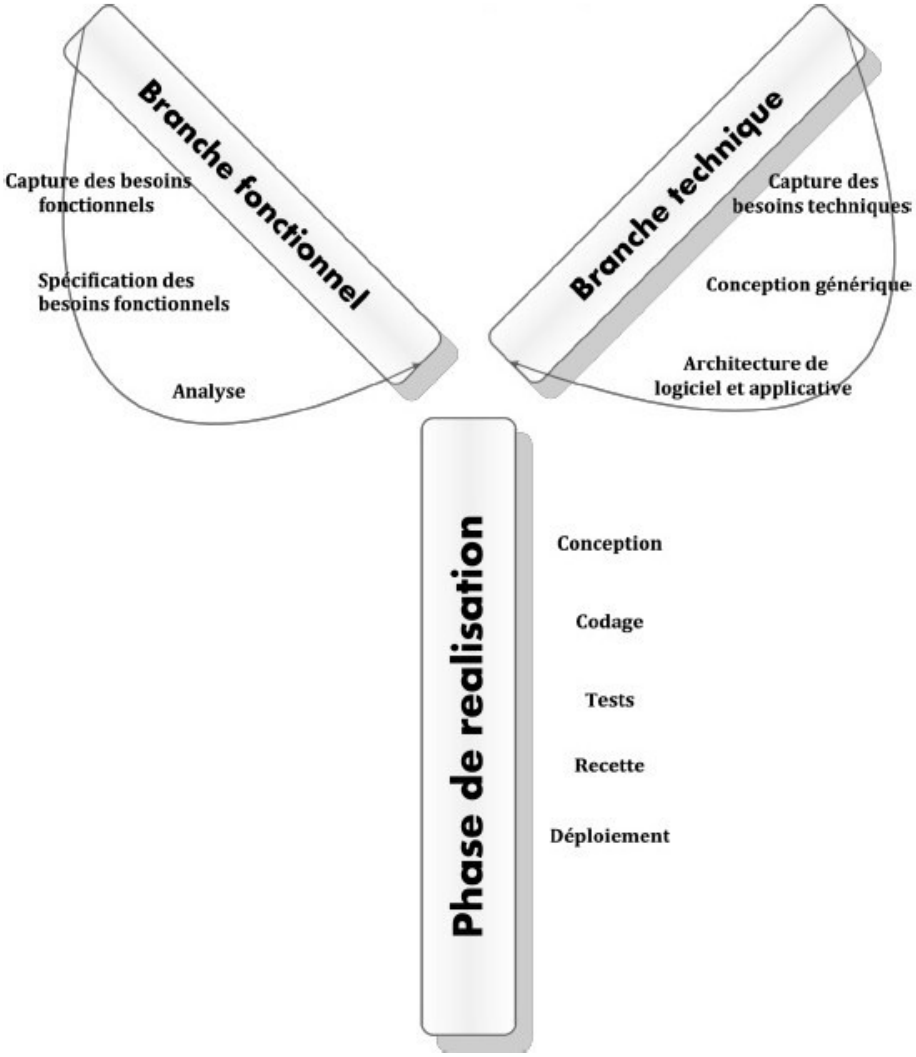


Figure 6 : Approche 2TUP

1.3. Branche conception et réalisation

Les étapes principales de cette branche se présentent comme suit :

- La conception préliminaire : Cette étape permet d'intégrer le modèle d'analyse fonctionnelle dans l'architecture technique de manière à tracer la cartographie des composants du système à développer.
- La conception détaillée : Permet d'étudier comment réaliser chaque composant. Le résultat fabriqué fournit l'image prête du système complet.

- Le Codage : Permet d'effectuer la production des composants et les tests des unités de code au fur et à mesure de leur réalisation.
- La recette : Consiste à valider les fonctionnalités du système développé [2].

2. Capture des besoins fonctionnels

2.1. Différents acteurs

Après nos 3 réunions avec les dirigeants et les futurs utilisateurs du système, nous avons dégagé 3 acteurs :Utilisateur, Administrateur, Super-Administrateur [Tableau 3 : Description des acteurs].

Acteurs	Description
Utilisateur	aura accès, après authentification, à la gestion des voyages et des bons de livraison.
Administrateur	peut, en plus des droits de l'utilisateur, gérer les camions, les destinations, les villes, les transporteurs, les donneurs d'ordres.
Super-Administrateur	peut, en plus des droits de l'administrateur, gérer les comptes et les permissions, et a le droit de supprimer chaque entité du système.

Tableau 3 : Description des acteurs

2.2. Exigences fonctionnelles

Pour faciliter le développement et la maintenance, nous avons divisé l'application en 3 modules, Administration, Décision et Bon de livraison.

2.2.1. Module Décision

Ce module doit permettre à l'utilisateur de :

- Créer un voyage
- Afficher la décision selon le calcule automatique du coût du voyage
- Ajouter les bons de préparation

- Ajouter les bons de livraison appartenant au voyage
- Consulter des voyages

Module Bon de livraison

Ce module doit permettre la gestion et le suivi des bons de livraison. L'utilisateur doit pouvoir :

- Créer des bons de livraison
- Ajouter les différentes dates d'un bon de livraison selon le transporteur
- Modifier l'état d'un bon de livraison
- Ajouter un commentaire selon l'état du bon de livraison

2.2.2. Module Administration

Ce module doit permettre à l'administrateur de gérer:

- Les types de camion
- Les villes et les destinations
- Les transporteurs
- Les donneurs d'ordre
- Les bons de livraison
- Les voyages

Le Super-Administrateur a le droit de gérer les comptes et aussi limiter l'accès aux ressources de l'application à travers le module de sécurité dynamique.

De plus l'application doit non seulement présenter une interface agréable et facile à utiliser mais aussi effectuer un nombre minimal de requêtes au près des serveurs sans oublier les facteurs de stabilité et de rapidité d'exécution.

3. Analyse des spécifications fonctionnelles

3.1. Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation sont utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel.

Le diagramme de cas d'utilisation [Figure 7: Diagramme des cas d'utilisation global du système] représente tous les acteurs et leurs interactions avec le système.

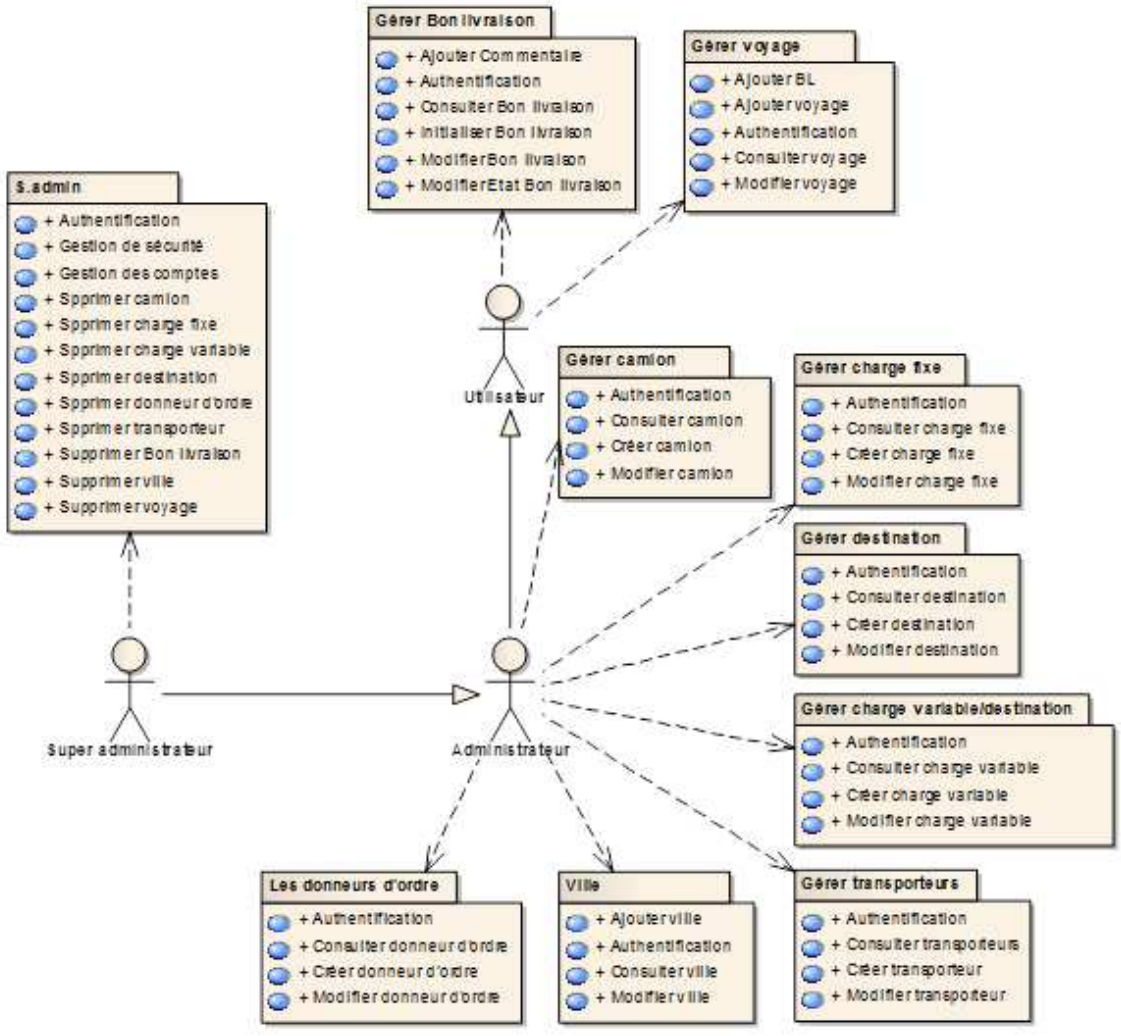


Figure 7: Diagramme des cas d'utilisation global du système

Le tableau [Tableau 4 : Authentification] représente plus en détail l'authentification des utilisateurs du système.

Titre	**Authentification**
Intention	Authentification des utilisateurs
Acteurs	• Utilisateur • Administrateur • Super-Administrateur
Pré-conditions	Serveur disponible
Commence quand	L'acteur veut se connecter
Définitions des enchainements	• Saisie du login et du mot de passe • Validation
Finis quand	Déconnection de l'acteur
Exceptions	• Utilisateur invalide • Mot de passe incorrect • Serveur non disponible
Post-conditions	Aucune

Tableau 4 : Authentification

3.1.1. Utilisateur

L'utilisateur a le droit de gérer les bons de livraison [Figure 8 : Diagramme des cas d'utilisation gérer Bon Livraison] et les voyages des camions [Figure 9 : Diagramme des cas d'utilisation gérer voyage].

3.1.1.1. Gestion des bons de livraison

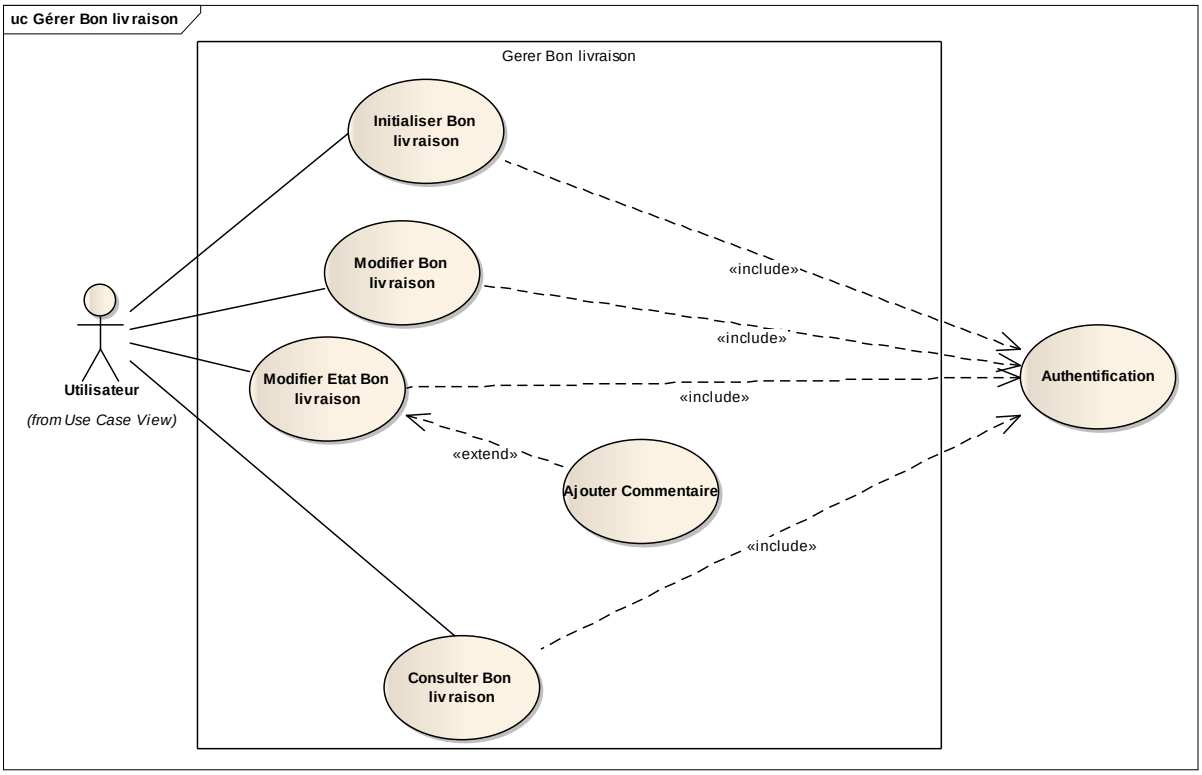


Figure 8 : Diagramme des cas d'utilisation gérer Bon Livraison

Titre	Créer un bon de livraison
Intention	Créer un bon de livraison
Pré-conditions	Utilisateur est connecté
Commence quand	L'utilisateur veut ajouter un bon de livraison
Définitions des enchainements	Remplir les champs de saisie
Finis quand	Validation de la saisie
Exceptions	Le code du bon de livraison existe déjà. La date saisi et antérieure à la date courante
Post-conditions	Le bon de livraison est enregistré selon le transporteur, son état sera modifier automatiquement a la saisi des dates de retour du BL.

Tableau 5 : Créer un bon de livraison

3.1.1.2. Gestion des voyages des camions

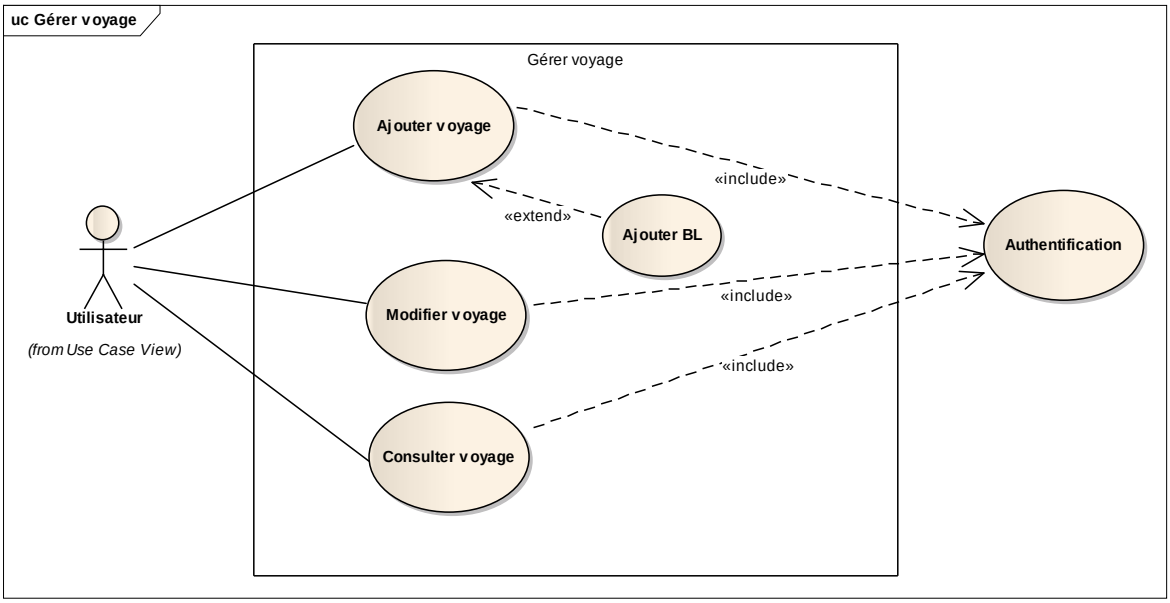


Figure 9 : Diagramme des cas d'utilisation gérer voyage

Titre	Créer un voyage
Intention	Ajouter un voyage
Pré-conditions	Utilisateur est connecté
Commence quand	L'utilisateur saisi les informations nécessaires pour un voyage
Définitions des enchainements	• Saisi le volume • Choisi le type de camion • Choisi la ville et les destinations du voyage • Choisi la décision à partir des 3 choix possible
Finis quand	Validation de la saisie
Exceptions	Le volume saisi est trop grand par rapport au volume des camions.
Post-conditions	Le voyage est enregistréainsi que les coûts du voyage calculés automatiquement.

Tableau 6 : Créer un voyage

3.1.2. Administrateur

L'administrateur a le droit de gérer les camions [Figure 10 : Diagramme des cas d'utilisation gérer camion], les destinations [Figure 11 : Diagramme des cas d'utilisation gérer destination], les transporteurs [Figure 12 : Diagramme des cas d'utilisation gérer transporteur], les donneurs d'ordres [Figure 13 : Diagramme des cas d'utilisation gérer donneur d'ordre] et les villes [Figure 14 : Diagramme des cas d'utilisation gérer ville].

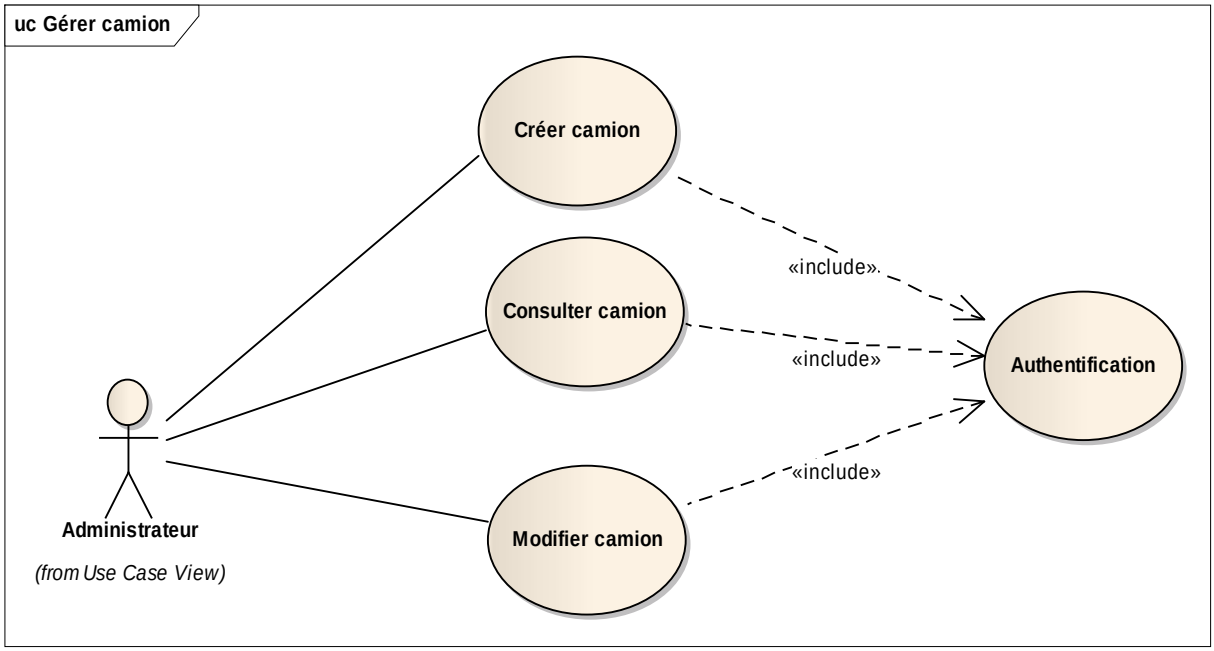


Figure 10 : Diagramme des cas d'utilisation gérer camion

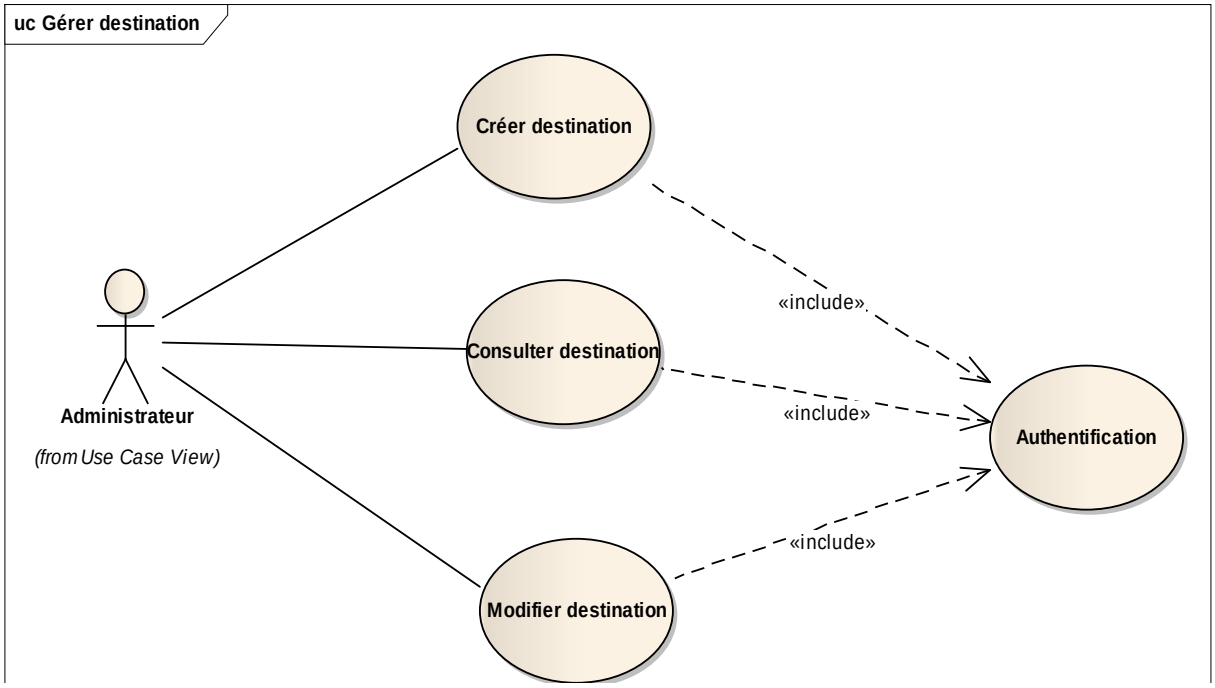


Figure 11 : Diagramme des cas d'utilisation gérer destination

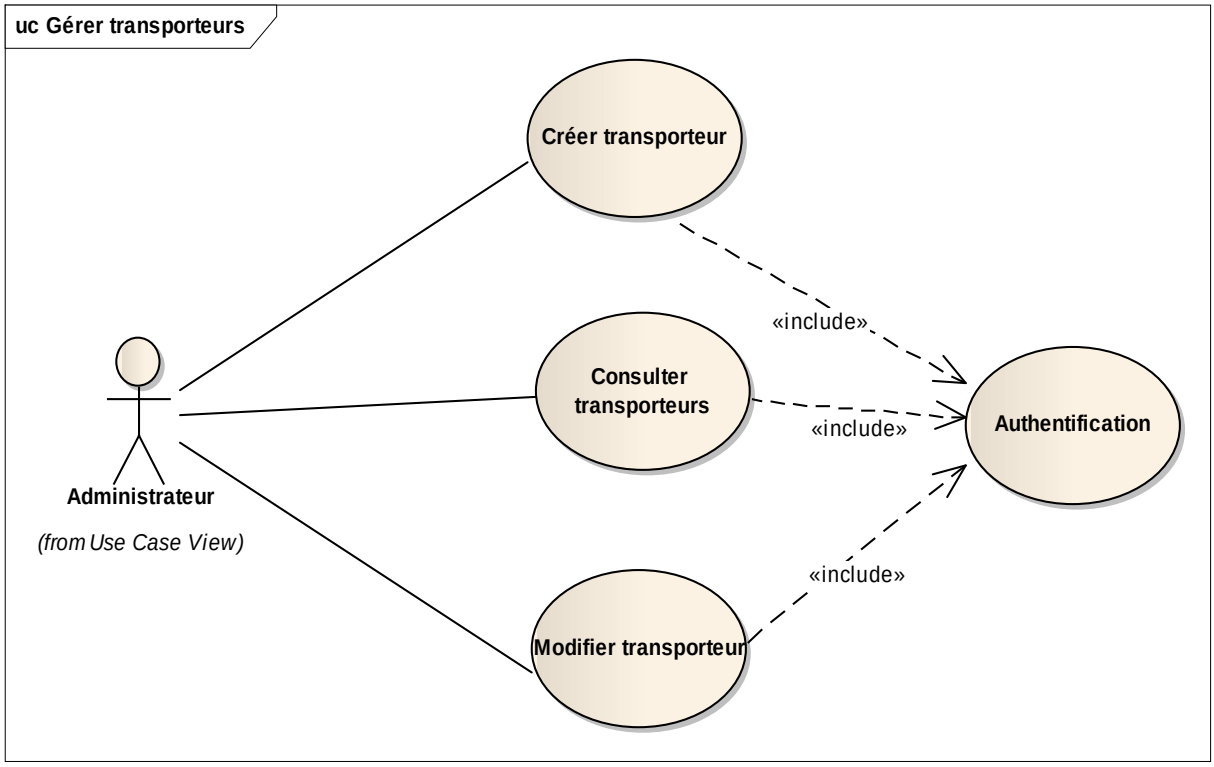


Figure 12 : Diagramme des cas d'utilisation gérer transporteur

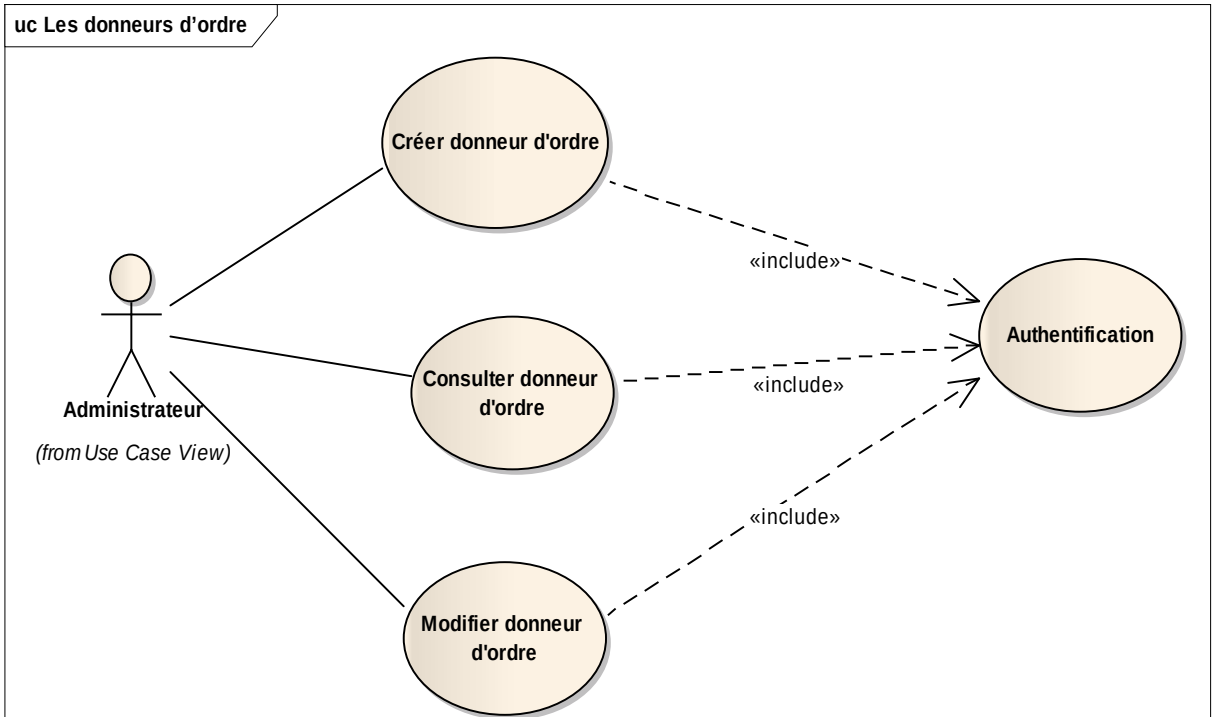


Figure 13 : Diagramme des cas d'utilisation gérer donneur d'ordre

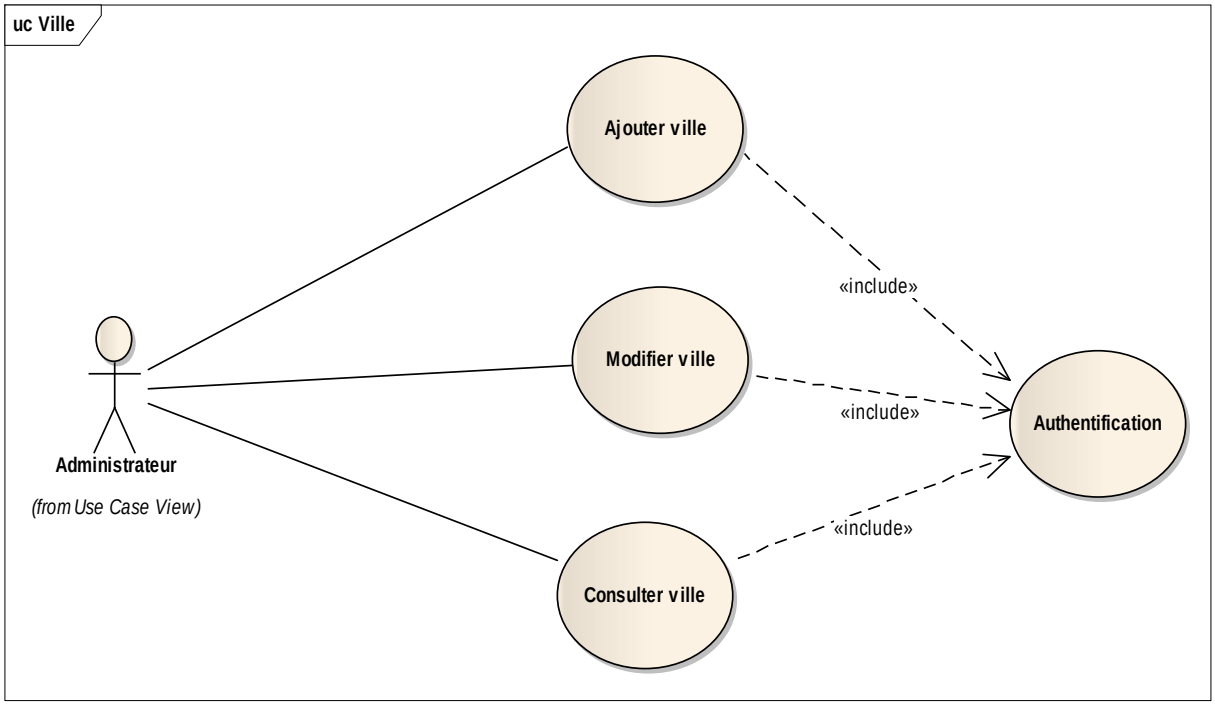


Figure 14 : Diagramme des cas d'utilisation gérer ville

3.1.3. Super Administrateur

Le Super Administrateur a le droit de gérer les comptes, la sécurité, les permissions et la suppression de chaque action du système [Figure 15 : Diagramme des cas d'utilisation des différentes actions d'un super Administrateur].

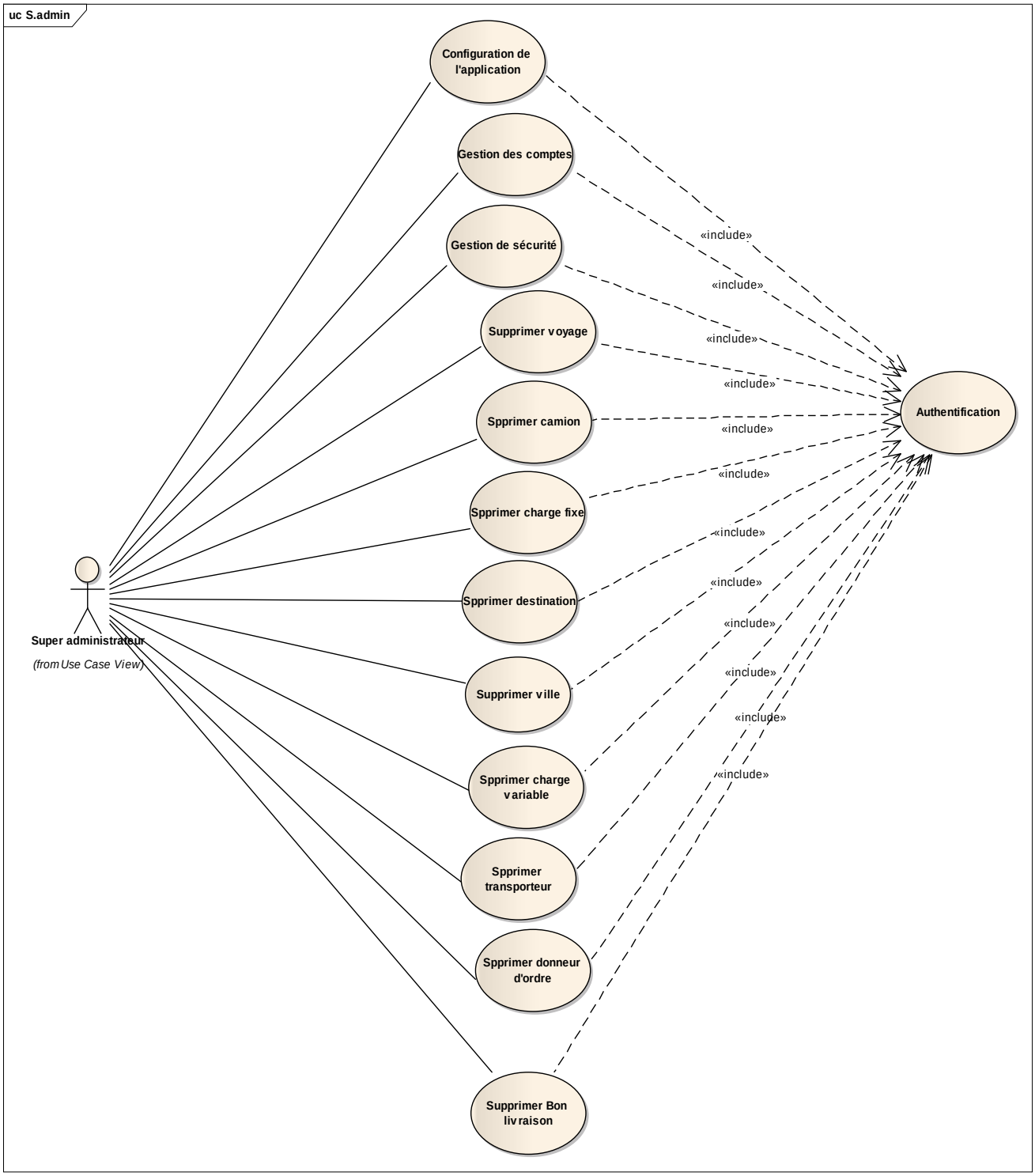


Figure 15 : Diagramme des cas d'utilisation des différentes actions d'un super Administrateur

Titre	Créer un compte
Intention	Créer un compte pour pouvoir se connecter en mode Administrateur
Pré-conditions	Connexion en mode Super Administrateur
Commence quand	Le Super Administrateur veut créer un compte Administrateur
Définitions des enchainements	Remplir les champs de saisie
Finis quand	Validation de la saisie
Exceptions	• Login existe déjà • Email existe déjà
Post-conditions	L'inscription est réussie

Tableau 7 : Créer un compte administrateur

Chapitre 3 : Analyse technique

1. Recensement des besoins techniques

La phase de capture des besoins techniques consiste à extraire et à faire un diagnostic des spécifications physiques et logicielles de l'application en sebasant sur l'abstraction élaborée dans la phase de l'étude fonctionnelle quise déroule en parallèle.

L'objectif fondamental de cette phase est dedégager les contraintes d'environnement et d'implémentation mises en jeu.

- La gestion de la performance.
- La dépendance des plateformes techniques.
- Lacapacité de la maintenance.
- L'extensibilité et la fiabilité.

En effet, notre application doit être conforme, en plus des exigences fonctionnellesextraites, à un nombre de contraintes techniques tel :

- Un serveur Tomcat 7 est envisagé pour déployer l'application.
- La base de données est de type MySQL.
- Le serveur apache sera configuré pour fonctionner en mode sécurisé et servira comme serveur frontal.
- Le développement des composants doit être réutilisable permettant l'extensibilité de l'application dans le futur.
- L'architecture applicative doit être organisée en couches distinctes.

2. Architecture physique

Une architecture 3-tiers [Figure 16 : Architecture physique] est la solution adoptée dans notre projet, elle est subdivisée en trois composants majeurs :

- Le composant serveur central qui se charge de la réalisation des traitements logiques et métiers.
- Le serveur client qui héberge la couche présentation et les JavaBeans.
- Les différents clients qui accéderont à ce serveur.

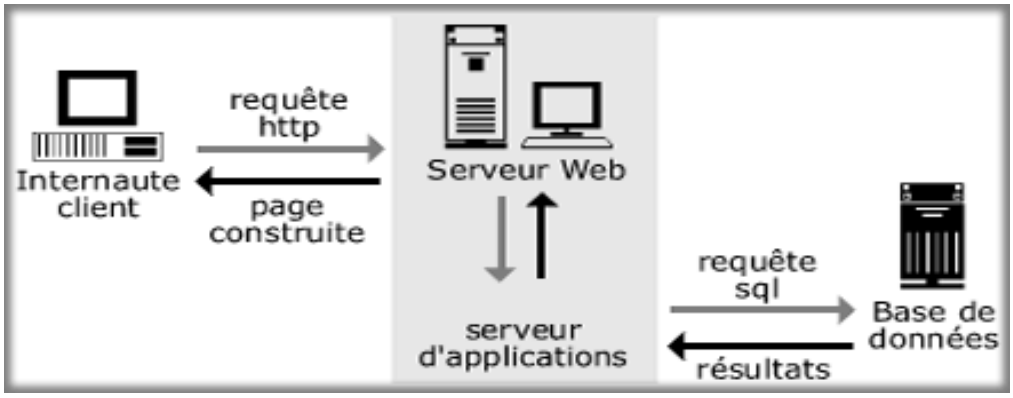


Figure 16 : Architecture physique

Notre choix est justifié par plusieurs raisons : le fait que les utilisateurs accèdent à l'application à distance à travers des requêtes « http ». De plus, le besoin d'assurer une grande souplesse, une sécurité flexible, une centralisation des traitements, et une meilleure performance, implique un partage des tâches entre les différents serveurs, c'est-à-dire que les applications sont délocalisées au niveau du serveur, et chaque serveur est spécialisé dans une tâche bien précise.

Composants	Description
Serveur WEB	En charge de répondre aux requêtes WEB.
Serveur d'applicationJ2EE central	Contient l'application proprement dite, ses tâches sont : • Récupérer les ressources dynamiques et effectuer les traitements nécessaires. • Implémenter la logique applicative et la logique métier. • Effectuer l'interface avec la base de données MySQL. • Effectuer l'interface avec le moteur de processus métier.
Serveur de base de données	Serveur qui contient la base de données, seul l'administrateur du système en a l'accès à l'aide d'un frontal.

Tableau 8 : Les composants serveurs du projet

3. Spécifications logicielles

Dans ce paragraphe, après une description de l'architecture physiqueconçue, nous allons dévoiler la plate-forme applicative, en l'occurrence, nousallons dresser la suite logicielle et l'ensemble des briques choisies pour le déploiement de l'application.

Dans notre projet, nous avons adopté un modèle de conception DesignPattern qui décrit une meilleure pratique ou une solution prouvée qui permetde résoudre à toutes contraintes les difficultés qui lui sont attachées, enmettant l'accent sur le contexte et sur les conséquences et les impacts de la solution.

Durant le cycle de développement de notre projet, nous avons choisi d'implémenter certains modèles de conception, notamment, le MVC 2 asynchronedu (Model, View, Controller2)[1], l'inversion de contrôle IOC[2], l'objet de transfert de données DTO[3], et l'objetd'accès aux données DAO[4]. Cette partie détaille le principe de chacun de cesDesign Patterns en rappelant ses concepts clés.

3.1. Modèle Vue Contrôleur 2

Le MVC est un modèle de conception qui repose sur la volonté de séparerles données, les traitements et la présentation. Ainsi l'application, sujet denotre projet, se retrouve segmentée en trois composants essentiels. Chacunde ces trois composants joue un rôle bien défini :

Le modèle représente les données et les règles métiers. C'est dans ce composant qu'on effectue les traitements liés au cœur du métier. Les donnéespeuvent être liées à une base de données, des EJBs ou des services Web. Ilest important de noter que les données sont indépendantes de la présentation.En d'autres termes, le modèle ne réalise aucune mise en forme, par ailleurs, ces données peuvent être affichées par plusieurs vues.

[1] MVC2 ou model viewcontroller est un design pattern de la couche présentation
[2] IOC ou Invertion of control est un nouveau concept de programmation permettant la séparation des couches
[3] DTO ou Data Transfer Obejects est un design pattern qui permet de le transfert de données entres couches de l'application
[4] DAO ou Data AccesObjects est un design pattern qui permet de séparer les données et les traitements sur ces données

La vue correspond à l'IHM, elle présente les données et interagit avec l'utilisateur.Dans le cadre de notre application, nous avons conçu des interfacesJSP, mais n'importe quel composant graphique peut jouer ce rôle.

Le contrôleur, quant à lui, se charge d'intercepter les requêtes de l'utilisateur,d'appeler le modèle puis de rediriger vers la vue adéquate. Il nedoit faire aucun traitement. Il ne fait que de l'interception et de la redirection.

Le MVC très pratique, peut se révéler lourd à mettre en place. Cecià cause de la multitude de contrôleurs à implémenter. Afin de simplifier laréalisation d'un tel modèle, une nouvelle version a été introduite : le MVC 2.C'est exactement le même modèle de conception à la différence qu'il n'y aplus qu'un seul contrôleur qui se charge de rediriger la requête vers le bontraitement [3].

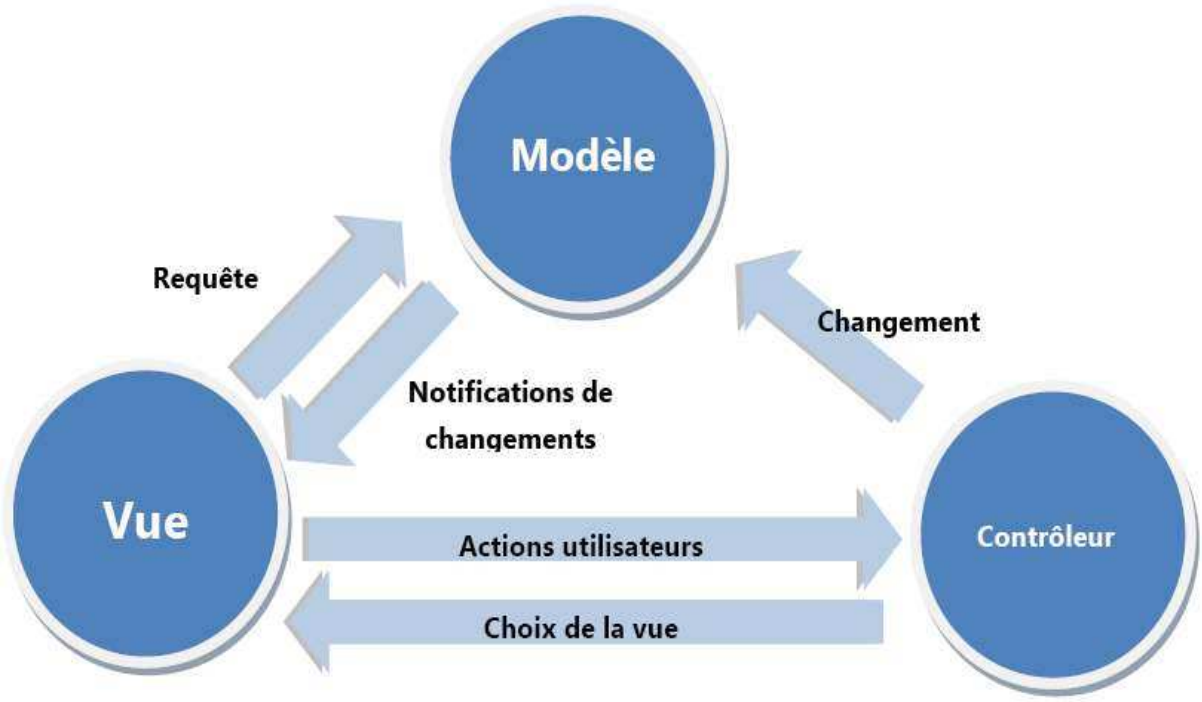


Figure 17 : Modèle Vue Contrôleur

3.2. Conteneur léger

3.2.1. Inversion de contrôle

Le principe de l'inversion de contrôle est au moins aussi ancien que celuide la programmation événementielle. Il s'agit d'un principe générique utilisépar de nombreux frameworks apparus bien avant la notion de conteneur léger.L'inversion de contrôle est aussi appelée principe Hollywooden référence à laphrase mythique « ne nous appelez pas, nous vous appellerons ».

Les conteneurs légers proposent une version spécialisée de l'inversion de contrôle. Ils se concentrent en fait sur le problème de la gestion des dépendancesentre objets et leur instanciation, notamment dans le cadre d'une dissociationdes interfaces et des implémentations [4].

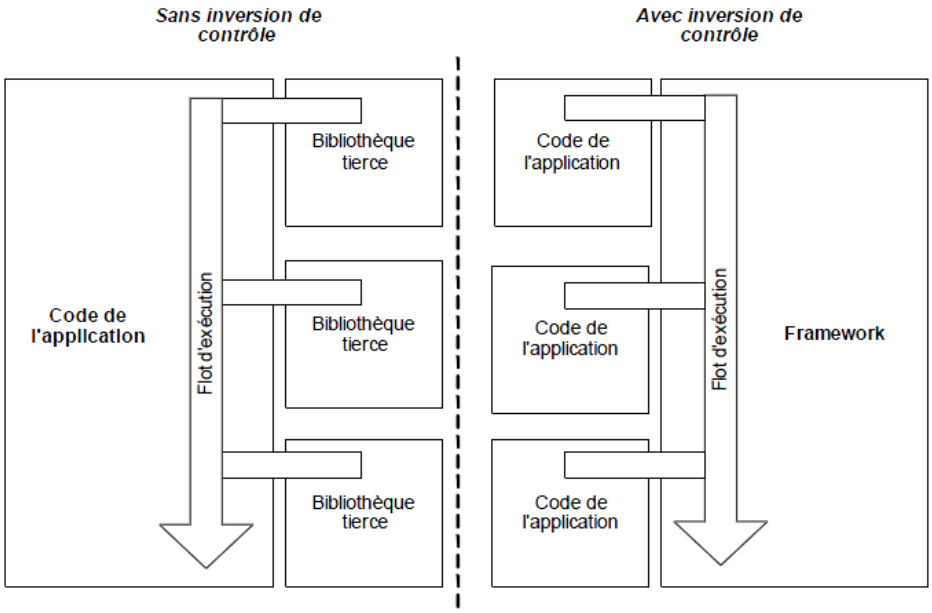


Figure 18 : Inversion de contrôle

3.2.2. Injection de dépendances

L'injection vise à rendre l'inversion de contrôle de la gestion des dépendancesla plus transparente possible vis-à-vis des classes concernées. Commenous l'avons vu à la section précédente, la recherche de dépendances créeun lien explicite entre les classes et le conteneur léger en charge de la gestionde leurs dépendances.

Grâce à l'injection de dépendances, nous pourront transformer ce lien explicite en lien implicite. Pour procéder à l'injectiondes dépendances, le conteneur léger initialise directement les objets(à partir d'un référentiel), libérant ainsi l'application de cette charge.

Aulieu d'utiliser l'opérateur new, le conteneur léger injecte dans l'applicationles instances dont elle a besoin, comme l'illustre [Figure 19 : Injection de dépendances]. Pour effectuercette initialisation, le conteneur peut implémenter deux méthodes : l'injectionde dépendances via le constructeur ou l'injection de dépendances via les modificateurs [4].

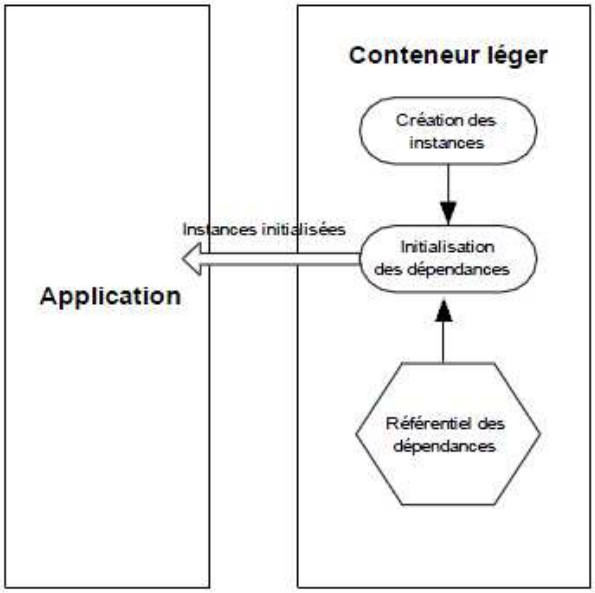


Figure 19 : Injection de dépendances

3.3. Design pattern DAO

Les DAO (Data Access Object), ou objets d'accès aux données, ont la tâche de créer, récupérer, mettre à jour et effacer des objets Java, d'où l'expression associée pattern CRUD (Create, Retrieve, Update, Delete). Cesont des classes utilitaires qui gèrent la persistance des objets métier. Nous séparons ainsi les données stockées dans des JavaBeans, et le traitement deces données.

Un DAO est un objet threadsafe, c'est-à-dire qu'il est accessible en simultané par plusieurs threads. À l'inverse d'un JavaBean, qui contient des données spécifiques de l'action en cours, un DAO ne fait que du traitement. Ilpeut, par conséquent, être commun à plusieurs

actions parallèles. Pour cetteraison, les DAO peuvent être implémentés comme des singletons, avec uneseule instance partagée par l'ensemble des objets de l'application [5].

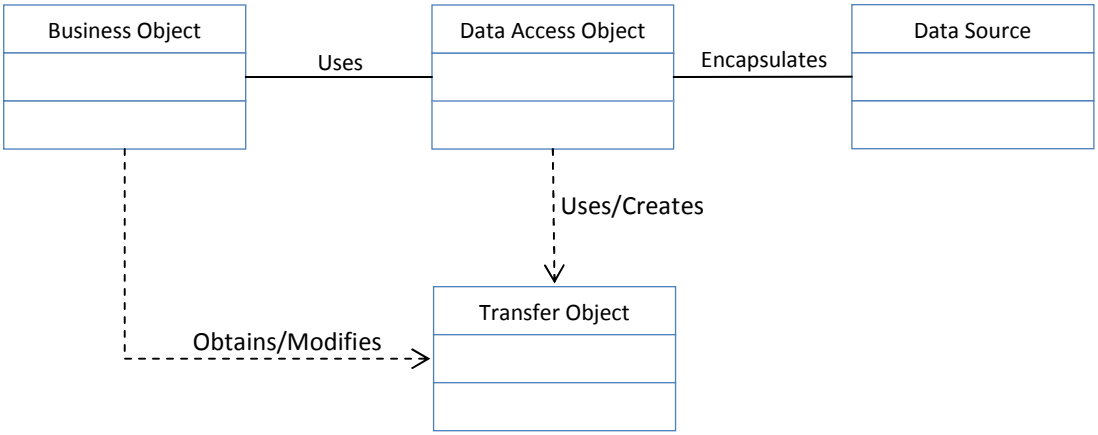


Figure 20 : Design pattern DAO

3.4. Design pattern DTO

Dans la conception d'applications distribuées, et pour satisfaire une seulerequête client, on se trouve souvent obligé d'émettre un nombre importantd'appels vers une interface distante, ce qui accroît le temps de réponse au-delà de l'acceptable. L'objet de transfert de données, Data Transfer Object(DTO), dit aussi Value Object, est un modèle de conception qui tente derésoudre la problématique suivante : préserver la simplicité de la sémantiqued'une interface d'appel de procédure sans être soumis aux problèmes de latenceinhérents à la communication distante [6].

La solution consiste à créer un DTO qui contient toutes les données nécessaires à l'appel, et modifier la signature de la méthode distante pour qu'elleaccepte le DTO en tant que paramètre unique et pour qu'elle renvoie unparamètre DTO unique au client. La meilleure solution est d'utiliser le modèle Assembler, dit aussi Data Transformer, qui crée des DTO à partird'objets métier et vice versa.

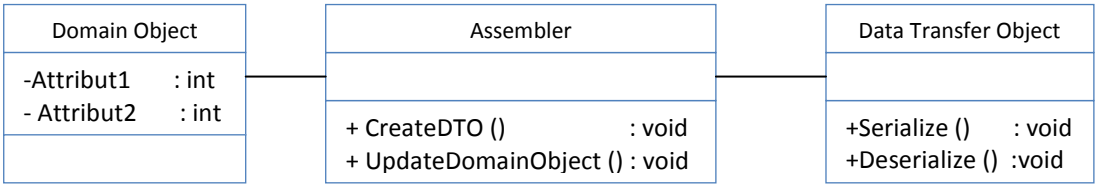


Figure 21 : Design pattern DTO

3.5. Architecture logique

L'architecture logique adoptée est une architecture en couches suivant le modèle MVC2 décrit dans le paragraphe précédent. Dans la suite nous dresserons, en partant du général au particulier, les composants des couches de cette architecture.

3.5.1. Architecture logique générale

Les applications modernes sont des applications qui adoptent une tendance typique dans le choix et l'organisation des couches structurant l'application. Principalement, nous distinguons les couches Web (présentation), couche de sécurité, couche service, couche métier et couche d'accès à la base de données.

- **La présentation des données**correspond à l'affichage,et l'interaction del'utilisateur avec l'application.
- **La sécurité**est la couche en charge de filtrer les accès et de vérifier les habilitations dédiées aux différents utilisateurs.
- **La couche service** est la couche qui rassemble les fonctionnalités du système. Chaque service est divisé enopérations qui constituent autant d'actions spécifiques que le servicepeut réaliser. Un service consiste essentiellement en une collection deservices qui interagissent et communiquent entre eux.
- **Le traitement métier** des données correspond à la mise en œuvre de l'ensemble des traitements et aussi la logique applicative.
- **L'accès aux données** correspond aux données qui sont destinées à être conservées sur la base de données, voire de manière définitive.

Figure 22 : Architecture logique

3.5.2. Architecture applicative

Dans notre approche, les couches communiquent entre elles à travers un modèle d'échange, et chacune d'entre elle propose un ensemble de services rendus. Les services d'une couche sont mis à disposition de la couche supérieure. On s'interdit par conséquent qu'une couche invoque les services d'une couche plus basse que la couche immédiatement inférieure ou plus haute que la couche immédiatement supérieure (chaque niveau ne communique qu'avec ses voisins immédiats).

L'architecture logicielle [Figure 23 : Architecture applicative] préconisée pour le développement de notre application multi-tiers est basée sur le concept de conteneur léger.

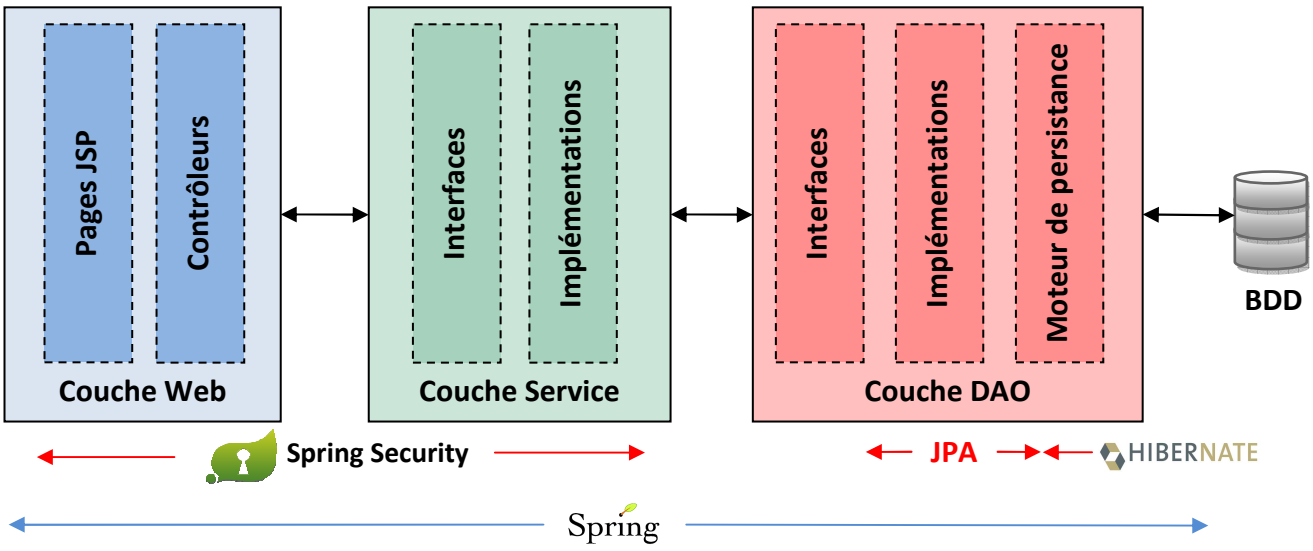


Figure 23 : Architecture applicative

Chapitre 4 : Conception

1. Diagramme de classes

Le diagramme de classes montre la structure interne du système. Il permetde fournir une représentation abstraite des objets du système, qui vontinteragir ensemble pour réaliser les cas d'utilisation.

1.1. Module Administration

Le diagramme [Figure 24 : Digramme de classe de la catégorie Administration] permet au Super-Administrateur de gérer les comptes et la sécurité de l'application.

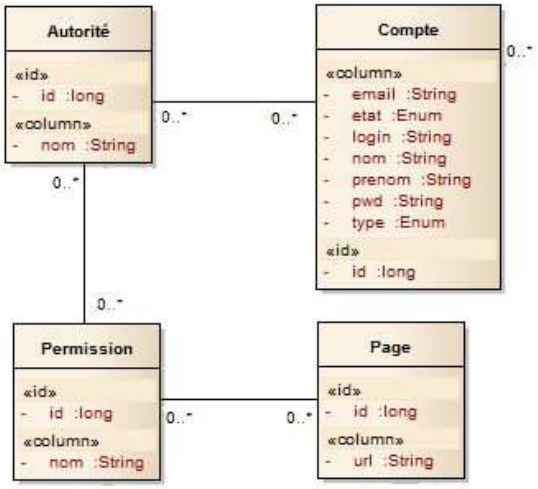


Figure 24 : Digramme de classe de la catégorie Administration

Classe	Description
Compte	Représente les utilisateurs du système
Autorité	Décrit les rôles de chaque type d'utilisateurs
Permission	Représente les droits d'accès pour chaque rôle
Page	Représente les URL des pages au quels l'utilisateur a le droit d'y accéder

Tableau 9 : Description de la catégorie Administration

1.2. Module décision

Le diagramme [Figure 25 : Digramme de classe de la catégorie Décision] permet à l'administrateur de gérer toutes les entités du module décision et aide l'utilisateur à prendre une décision sur chaque voyage.

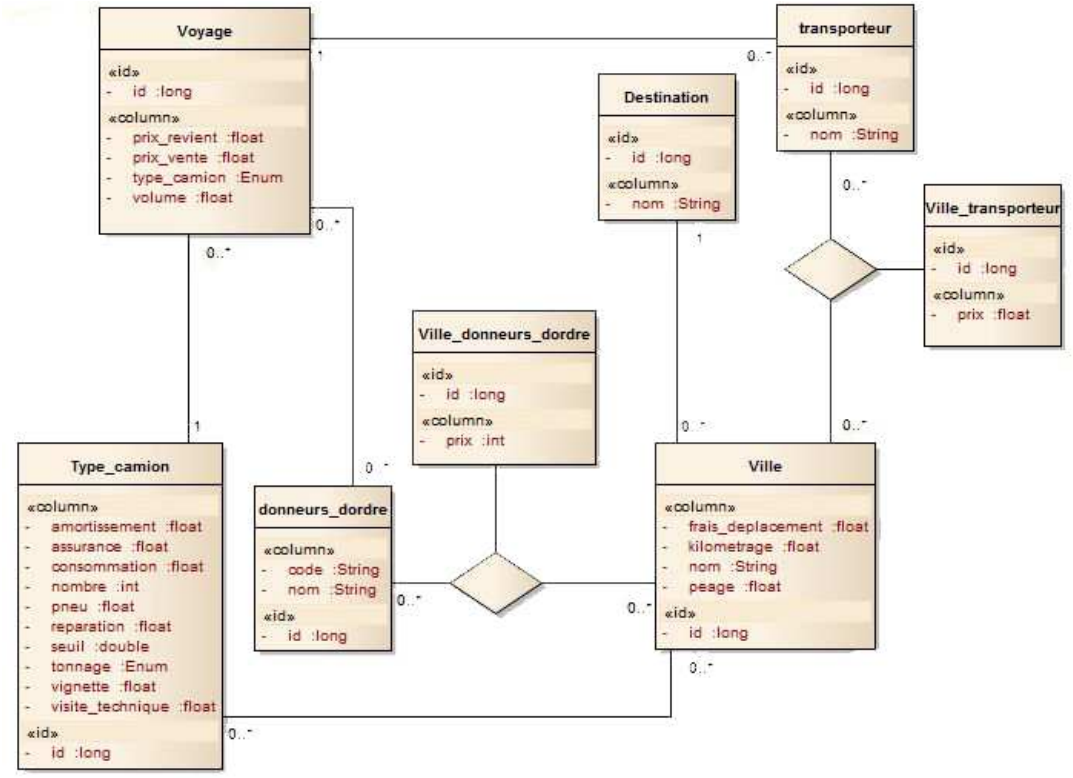


Figure 25 : Digramme de classe de la catégorie Décision

Classe	Description
Ville	Décrit les villes ou la livraison va être effectué (La ville de départ est Mohammedia)
Destination	Représente les destinations de chaque ville
Donneurs_dordre	Représente les clients aient leurs produits stocké dans les entrepôts
Transporteur	Représente les prestataires de transport et leur tarification pour chaque ville(En cas de non disponibilité des camions de la flotte interne)
Type_camion	Décrit les types de camions de la flotte de la SNTL-DAMCO
Voyage	Représente les voyages effectués par la flotte interne ou les prestataires

Tableau 10 : Description de la catégorie Décision

1.3. Module Bon de livraison

Le diagramme [Figure 26 : Digramme de classe de la catégorie Bon de livraison] permet à l'utilisateur de gérer les bons de livraison correspondant à chaque voyage.

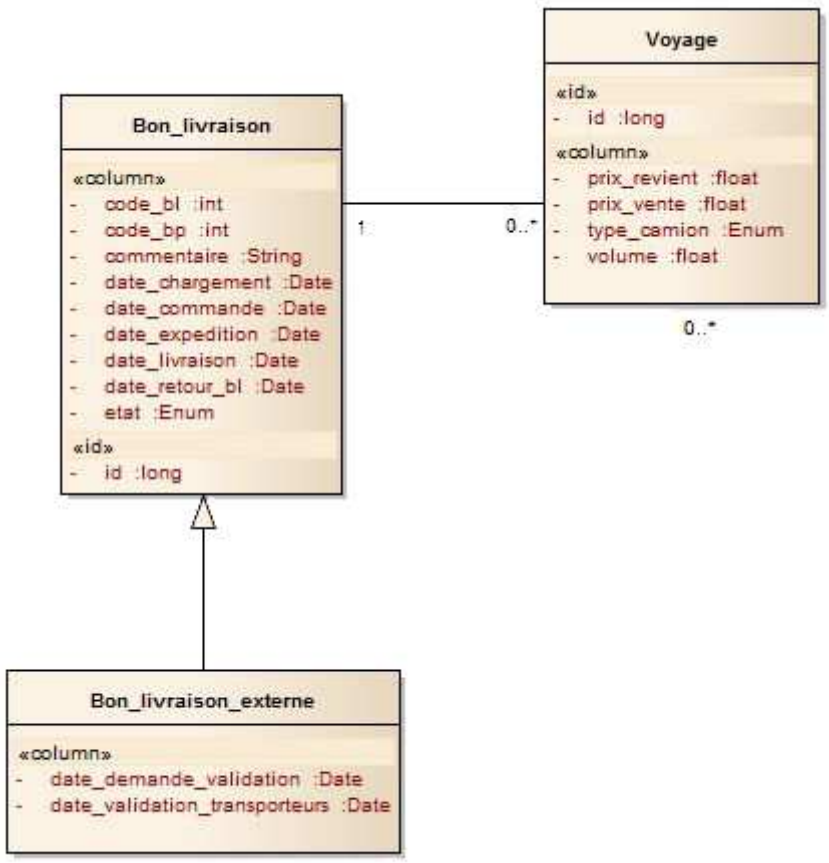


Figure 26 : Digramme de classe de la catégorie Bon de livraison

Classe	Description
Bon_livraison	Décrit les bons de livraisons traités par la flotte interne
Bon_livraison_externe	Cette classe hérite de classe Bon_livraison sauf qu'ils sont traités par les transporteurs externes.

Tableau 11 : Description de la catégorie Bon de livraison

1.4. Digramme de classe globale

Le diagramme [Figure 27 : Diagramme de classe globale] représente toutes les entités du système. On a mis en place un fichier de configuration pour toutes les informations à valeur fixe et ceux ayant une fréquence à modification faible.

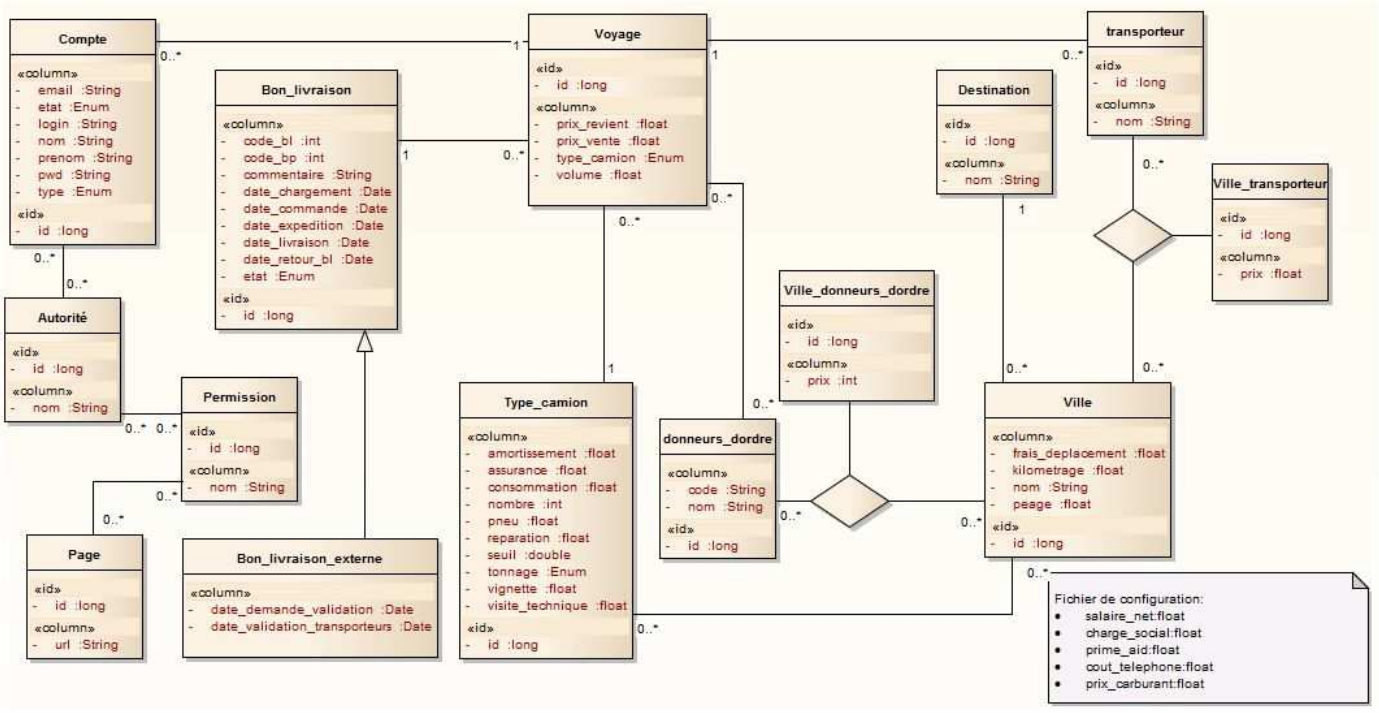


Figure 27 : Diagramme de classe globale

2. Diagramme d'activité

2.1. Processus d'un bon de livraison

Le processus [Figure 28 : Processus d'ajout d'un bon de livraison] commence par la création d'un nouveau bon de livraison ou l'utilisateur devra fournir les informations suivantes :

- Le numéro du bon de livraison.
- Les dates de commande, d'expédition et de chargement.
- La ville de destination.
- L'adresse de livraison.
- Le demandeur d'ordre.

Ensuite l'utilisateur devra choisir le transporteur du bon de livraison, à cette étape deux chemins sont possible :

- S'il choisit SNTL Damco (c.à.d. De traité le transport du bon de livraison en interne), le bon de livraison sera de type interne, et l'utilisateur devra saisir la date de livraison et la date du retour du bon de livraison après son retour physique.
- S'il choisit un transporteur externe, le type du bon de livraison sera de type externe, et l'utilisateur devra saisir de plus des dates d'un bon de livraison interne les dates de demande et de confirmation de la livraison qu'il obtiendrait auprès du transporteur.

Après le retour du bon de livraison l'utilisateur doit fournir son état :

- Validé : la livraison a été effectuée sans aucun problème.
- Refusé : la livraison a été effectuée mais le destinataire n'a plus besoin de la marchandise.
- En instance : dans ce cas l'utilisateur devra choisir la nature du problème :
    - Problème du cachet.
    - Problème manque du produit.
    - Problème de paiement.
    - Problème de perte du Bon de livraison.
    - Retour partielle sur le BL.
    - Problème de qualité (le BL sera re-livré avec le même code)
    - Re-livraison complète du BL <**/relivrer**> (Dans ce cas l'utilisateur devra recommencer tout le processus)

Tous les bons de livraison en instance seront validés après traitement du problème sauf dans le cas de lare-livraison complète.

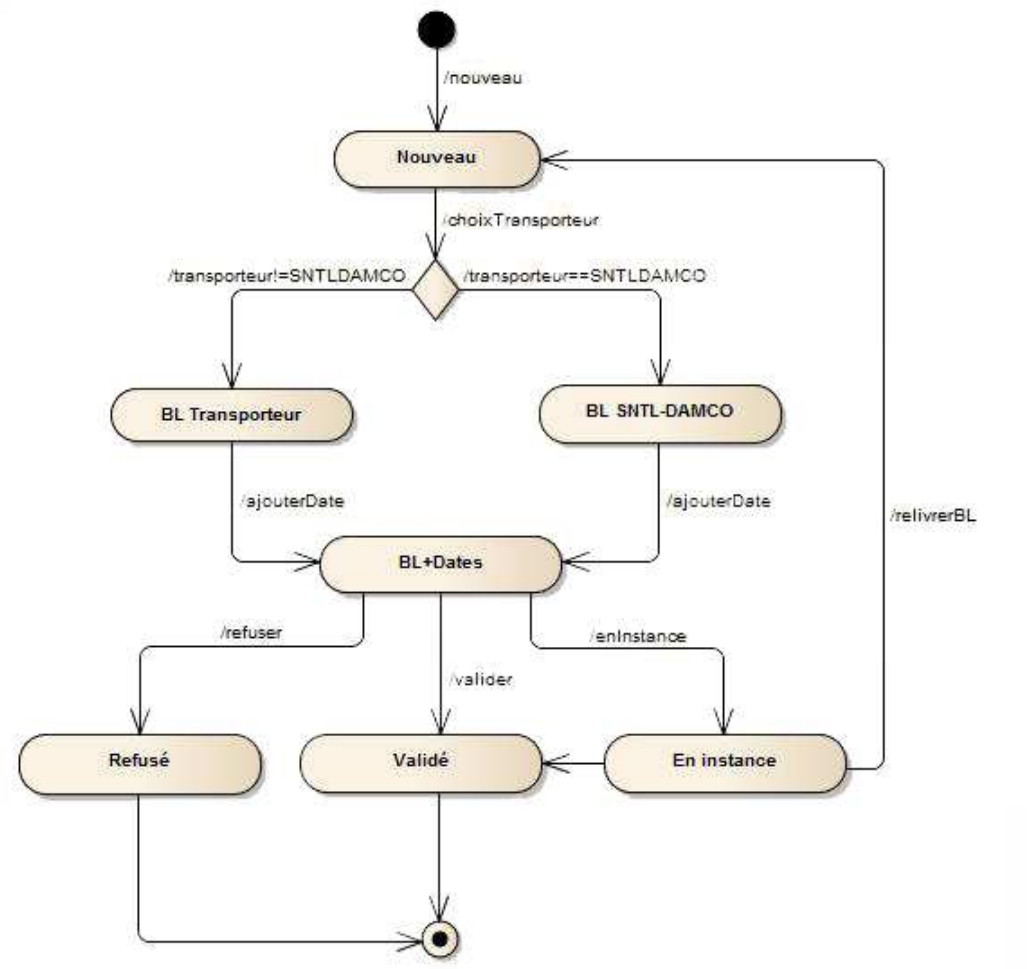


Figure 28 : Processus d'ajout d'un bon de livraison

2.2. Processus d'ajout d'un voyage

Le processus [Figure 29 : Processus d'ajout d'un voyage] commence par la création d'un nouveau voyage où l'utilisateur devra fournir les informations suivantes :

- Volume.
- Type de camion.
- Ville.
- Destinations.

La liste des camions est actualisée de sorte à ce qu'elle ne contienne que les camions pouvant contenir le volume saisi.

La liste des destinations est actualisée de sorte à ce qu'elle ne contienne que les destinations de la ville choisi.

Au moment où l'utilisateur doit choisir le type de camion adéquat pour le voyage, deux chemins sont possible :

- Si le type de camion existe (c.à.d. le type de camion sélectionné fait partie de la flotte des camions de SNTL Damco), alors l'application devra calculer le coût du voyage par rapport à la ville choisi et affiché la décision ayant le prix de revient maximal.
- Si le type de camion n'existe pas (c.à.d. la SNTL Damco ne possède pas de camion de ce type), alors l'application devra afficher le transporteur le moins chère par rapport à la ville choisi.

Après l'affichage de la décision l'utilisateur devra saisi les codes des bons de préparation et enregistrer le voyage. Les bons de livraisons seront ajoutés au voyage après leur impression sous format papier.

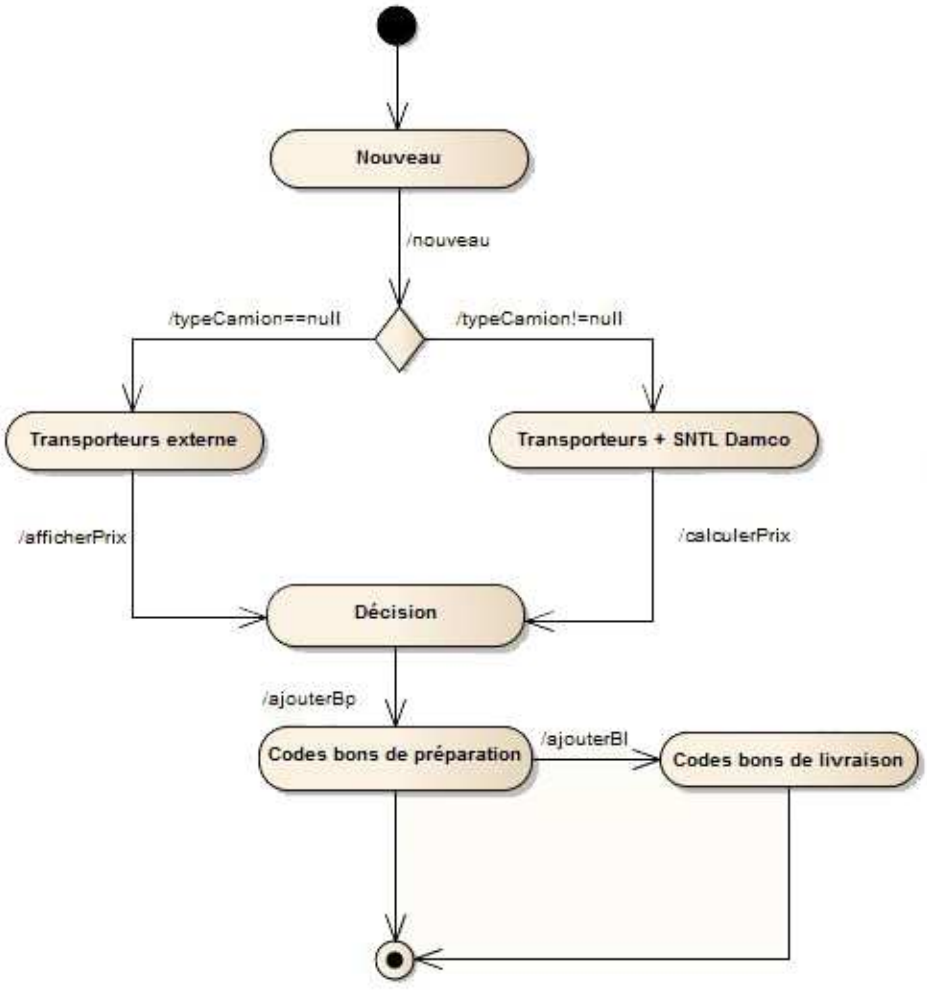


Figure 29 : Processus d'ajout d'un voyage

Chapitre 5 : Mise en œuvre

1. Outils de développement

1.1. Serveur d'application

1.1.1. Serveur Apache Tomcat

Apache Tomcat est un conteneur libre de servlets Java 2 Enterprise Edition. Issu du projet Jakarta, Tomcat est un projet principal de la fondation Apache Tomcat implémente les spécifications des servlets et des JSP (JavaServer Pages)du Java CommunityProcess.

Il est paramétrable par des fichiers Xmlet de propriétés, et inclut des outils pour la configuration et la gestion. Il comporte également un serveur Http [7].

1.1.2. Serveur MySQL

MySQL est un serveur de bases de données relationnelles. Un serveur de bases de données stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête.

Le SQL dans MySQL signifie StructuredQueryLanguage : le langage standardpour les traitements de bases de données [8].

1.2. Framework de développement

1.2.1. La plateforme JEE

JEE (Java Enterprise Edition) [16] est une norme proposée par la firme Sun, visant à définirun standard de développement d'applications d'entreprises multi niveaux, basées sur descomposants. Dans la mesure où JEE s'appuie entièrement sur java, elle bénéficie de sesavantages, à savoir : une bonne portabilité et une maintenabilité du code.

L'architecture JEE repose sur des composants distincts, interchangeables et distribués, ce qui permet :

- Une simplicité d'extension de l'architecture.
- Une haute disponibilité, ce qui garantit une bonne qualité de service.
- Une maintenabilité des applications

1.2.2. Spring Framework

Spring [9], est un conteneur léger (c.à.d. une infrastructure similaire à un serveur d'application J2EE). Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendance entre ces objets. Spring offre plusieurs services qui facilitent le développement et les tests des applications J2EE.

Pour notre travail, Spring nous a permis de diminuer la quantité de code et d'assurer le découplage des composants de l'application lors de l'utilisation de l'injection de dépendance (IOC).

Ce Framework est organisé en modules, reposant tous sur le modèle SpringCore et s'intégrant avec d'autres Framework.

1.2.3. Spring Data

Spring-Data-JPA [10]. est l'un des projets de Spring reposant sur Spring-Data, il vise à améliorer la mise en œuvre de la couche d'accès aux données en réduisant considérablement l'effort d'écriture du code d'implémentation en particulier pour les méthodes CRUD (Create, Read, Update, Delete) et de recherche.

La notion centrale dans Spring-Data-JPA est la notion "Repository". Le repository est une interface à écrire par le développeur oùil déclareles méthodes utiles d'accès aux données et Spring-Data-JPA fournit les implémentations nécessaires.

1.2.4. Spring Security

Spring Security [5]propose un modèle de sécurité éprouvé, stable, performant, et répond à l'ensemble des attentes : sécurisation des URL, des méthodes et des instances d'objets, fourniture de nombreux filtres (par exemple, permet l'authentification par formulaire, l'authentification automatique par cookie, etc…).

Pour des besoins de sécurité basiques, Spring Security propose des fonctionnalités faciles à mettre en œuvre, grâce notamment à son schéma XML dédié. Pour des besoins plus avancés, il est nécessaire d'appréhender des mécanismes plus avancés du framework.

1.2.5. Spring MVC

Le frameworkSpring fournit des intégrations avec les principaux frameworks MVC ainsi que sa propre implémentation. Forts de leur expérience dans le développement d'applications JavaEE, ses concepteurs considèrent que l'injection de dépendances offre un apport de taille pour concevoir et structurer des applications fondées sur le patron MVC.

Les principaux composants de Spring MVC peuvent être répartis en trois groupes, selon leur fonction :

- Gestion du contrôleur façade et des contextes applicatifs. Permet de spécifier les fichiers desdifférents contextes ainsi que leurs chargements. Le contrôleur façade doit être configuré defaçon à spécifier l'accès à l'application.
- Gestion des contrôleurs, consiste à configurer la stratégie d'accès aux contrôleurs, ainsique leurs différentes classes d'implémentation et leurs propriétés.
- Gestion des vues, consiste à configurer la ou les stratégies de résolution des vues ainsi queles frameworks ou technologies de vue mis en œuvre [5].

1.2.6. Framework de persistance : Hibernate/JPA

Le couple Hibernate/JPA permet de créer, de requêter et de manipuler des objets Java persistants, c'est-à-dire des objets Java correspondant à des enregistrements BDD. Ainsi chaque opération effectuée sur ces objets sera répercutée en base.

Hibernate est l'implémentation concrète du moteur de persistance. Outre le moteur lui-même, il offre un certain nombre d'APIs de requêtage. JPA offre un niveau d'abstraction supplémentaire en proposant un ensemble d'interfaces standard auxquelles les implémentations d'Hibernate (et d'autres Framework de persistances) se conforment [11].

1.2.7. JSR303 (Bean Validation)

La JSR 303 (*Java SpecificationRequest*) a pour objectif d'éviter la duplication de la validation des données dans les diverses couches de l'application en la localisant dans la définition des Beans Java. Ceci, dans le but de gagner en productivité et d'éviter les bugs liés à la redondance de la validation.

La JSR 303 définit un modèle de meta-données et une API pour valider les Beans Java. Cette validation s'effectue en utilisant les annotations mais il est possible d'utiliser des fichiers XML[12].

1.3. Environnement de développement

1.3.1. IntelliJ IDEA

Intellij IDEA, 10ème du nom, est un IDE considéré comme l'un des plus "intelligent" en matière de programmation Java. Sans être intrusif, la plateforme de développement se comporte comme une seconde main pendant l'écriture des lignes de code avec un débogage instantané, une correction douce, un apprentissage de votre code en vue de tests discrets.

Evidemment, l'intervention de l'IDE s'effectue en fonction de vos demandes et niveaux d'intervention requis par vos soins. L'interface dispose aussi d'un outil de design pour la réalisation d'interface utilisateur, d'un système de gestion des langues, de moteur de recherche, etc. Intellij IDEA supporte les langages Java bien entendu, mais aussi Javascript/Flex, HTML/xHTML/CSS, XMS/XSL, Ruby/jRuby, Groovy, SQL et FreeMarker/Velocity. Enfin, l'IDE s'adapte parfaitement aux technologies et frameworks tels que Spring, JSP, Ajax, JBossSeam, Web Services, Rails, Grails, etc. [13].

1.3.2. Maven

Maven est un outil de gestion de projet qui comprend un modèle objet pour définir un projet, un ensemble de standards, un cycle de vie, et un système de gestion des dépendances. Il embarque aussi la logique nécessaire à l'exécution d'actions pour des phases bien définies de ce cycle de vie, par le biais de plugins. Les projets Maven, les dépendances, les builds, les artefacts : tous sont des objets qu'il va falloir modéliser et décrire. Ces objets sont décrits dans un fichier XML appelé Modèle Objet de Projet « POM ».

Le POM indique à Maven quel type de projet il va devoir traiter et comment il va devoir s'adapter pour transformer les sources et produire le résultat attendu. Ainsi, comme le fichier web.xml décrit, configure et personnalise une application web Java, c'est la présence d'un fichier pom.xml qui définit un projet Maven. Il s'agit d'une déclaration décrivant un projet Maven, c'est le « plan » abstrait que Maven doit comprendre et suivre pour construire un projet [14].

1.3.3. Git/Bitbucket

Git est un logiciel de gestion de versions décentralisé. Il est conçu pour être efficace tant avec les petits projets, que les plus importants. Git fonctionne de façon décentralisée, c'est-à-dire que le développement ne se fait pas sur un serveur centralisé, mais chaque personne peut développer sur son propre dépôt. Git facilite ensuite la fusion (merge) des différents dépôts.

Le serveur utilisé dans le cadre du projet est Bitbucket. Ce dernier offre un service web d'hébergement et de gestion de développement de logiciels, utilisant Git.

2. Réalisation de la plate-forme

2.1. Structure du projet

Afin d'améliorer la maintenabilité, l'évolutivité et la modularité du projet, nous avons adopté une structure (Maven) organisée dans l'arborescence [Figure 30 : Diagramme de packages].

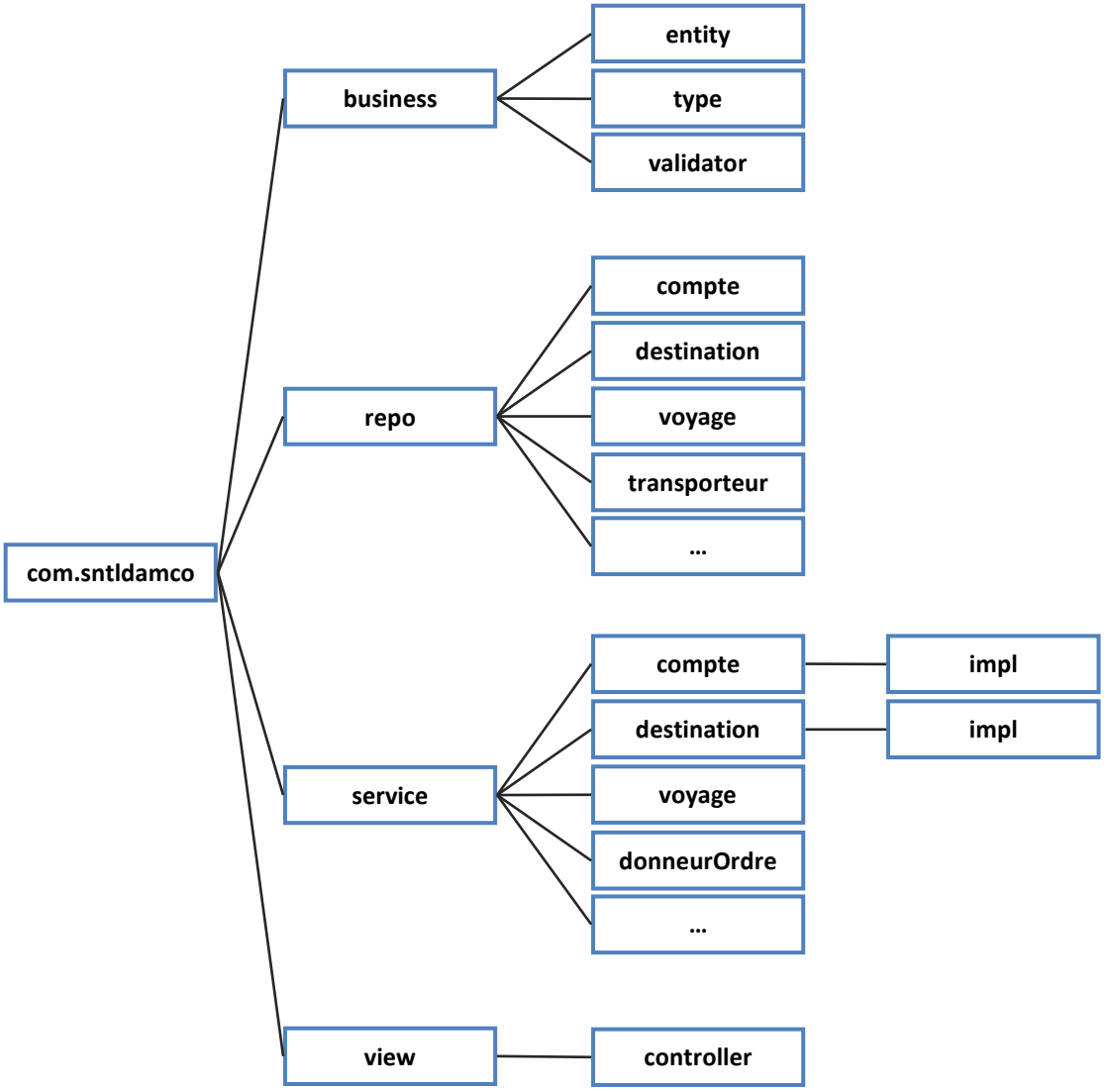


Figure 30 : Diagramme de packages

Le tableau [Tableau 12 : Description des packages du projet] décrit une liste non exhaustive des packages qui constituent notre application :

Package	Description
com.sntldamco.business	Contient l'ensemble des classes métier
com.sntldamco.type	Contient l'ensemble des constantes (Énumérations)
com.sntldamco.validator	Contient l'ensemble des implémentations de validateurs d'entités (JSR-303)
com.sntldamco.repo.compte	Contient les interfaces d'administration pour l'accès aux données (Spring Data)
com.sntldamco.repo.destination	Contient les interfaces de ville et destination pour l'accès aux données (Spring Data)
com.sntldamco.service.compte	Contient l'ensemble des interfaces des services d'administration
com.sntldamco.service.compte.impl	Contient les implémentations des classes du service d'administration
com.sntldamco.view.controller	Contient les Contrôleurs (Srping MVC) que les vues de l'application utilisent

Tableau 12 : Description des packages du projet

2.2. Optimisation des performances

Les performances sont une problématique souvent évoquée au sujet desbases de données. On parle du Tunning (ajustement des paramètres pour affiner les performances). Nous avons optimisé les performances sur les deuxniveaux :

- **Conceptuel :** A ce niveau nous avons créé les indexes adéquats sur les champs des tables. Nous avons choisi les types de données de façon à satisfaire les besoins du système et optimiser l'espace de stockage.
- **Applicatif :**A ce niveau nous avons utilisé le mécanisme de **la lecture différée** qui retarde la lecture des objets jusqu'au moment où l'application voudrais y accéder par navigation de référence [15].

3. Interfaces graphiques

3.1. Formulaire d'authentification

Par mesure de sécurité, L'accès à l'application est protégé par un formulaire d'authentification.

Comme nous l'avons mentionné dans les cas d'utilisation, l'application est accessible via trois modes (Super-Administrateur, Administrateur, Utilisateur)

Figure 31 : Page d'authentification

Pour se connecter, l'utilisateur doit donner son identifiant et son mot de passe. Si lesinformations saisies sont correctes, le système affiche la page d'accueil permettant l'accès auxfonctionnalités relatives au profil de l'utilisateur connecté. Dans le cas contraire un messaged'erreur est affiché à l'utilisateur lui indiquant la cause de l'échec de l'authentification.

Figure 32 : Interface échec d'authentification

3.2. Le gestion des comptes

Le menu que contient la page [Figure 33 : Interface créer compte], est le menu duSuper-Administrateur car lui seul peut gérer les comptes.

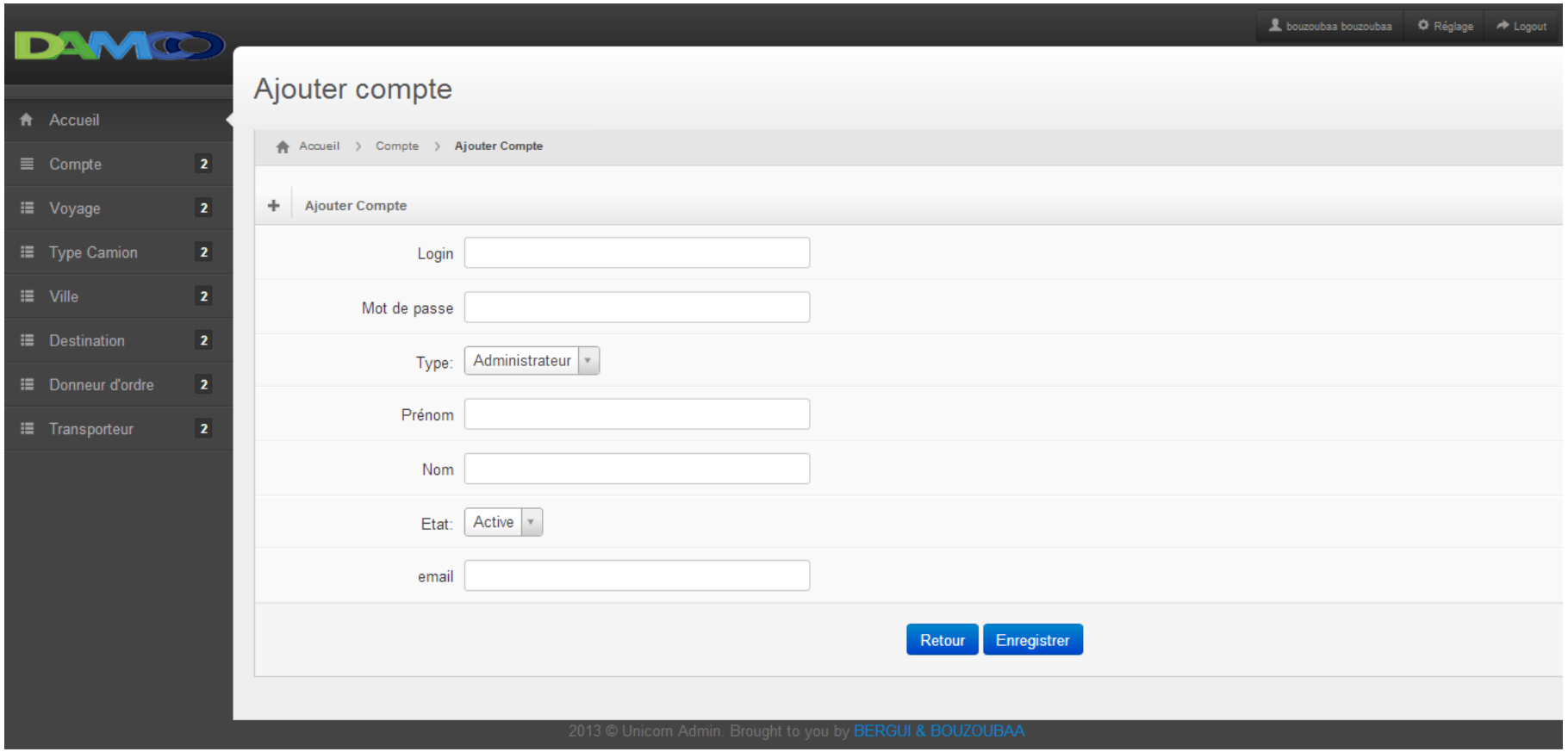


Figure 33 : Interface créer compte

L’interface [Figure 34 : Interface erreur créer compte] représente quelques erreurs que le Super-Administrateur peut commettrelors de l’ajout d’un compte.

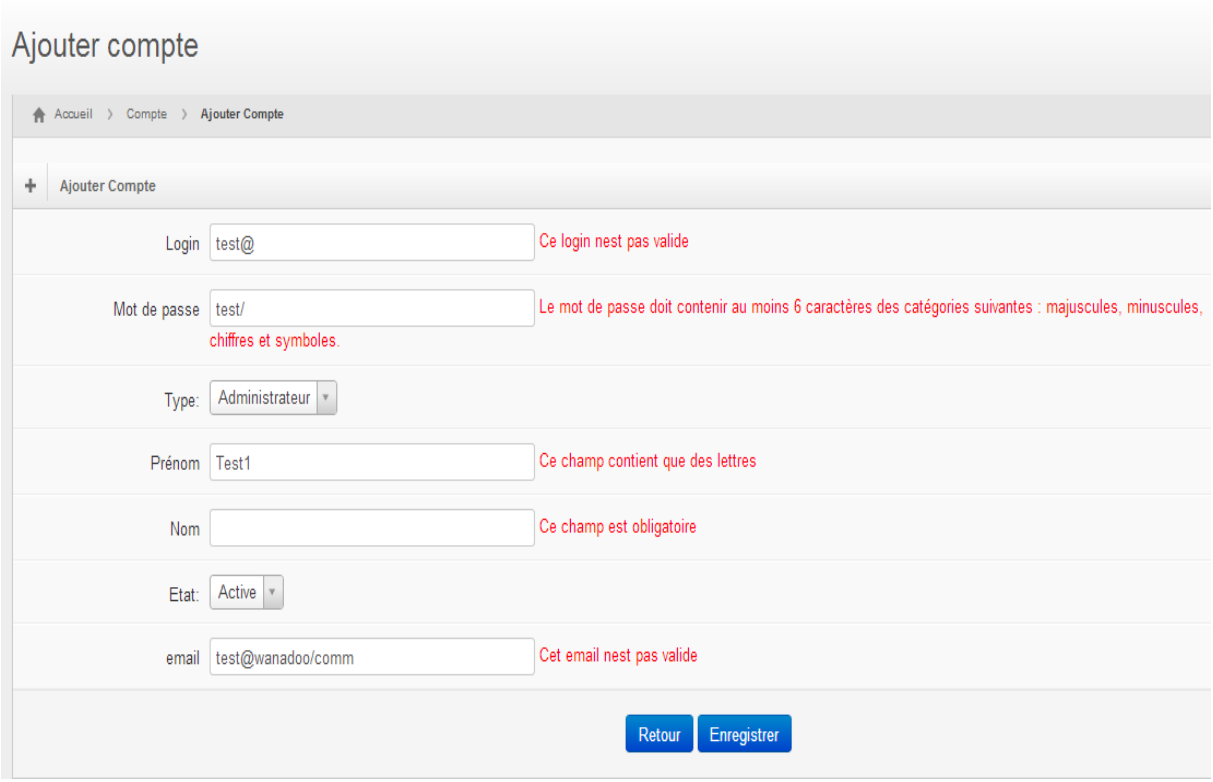


Figure 34 : Interface erreur créer compte

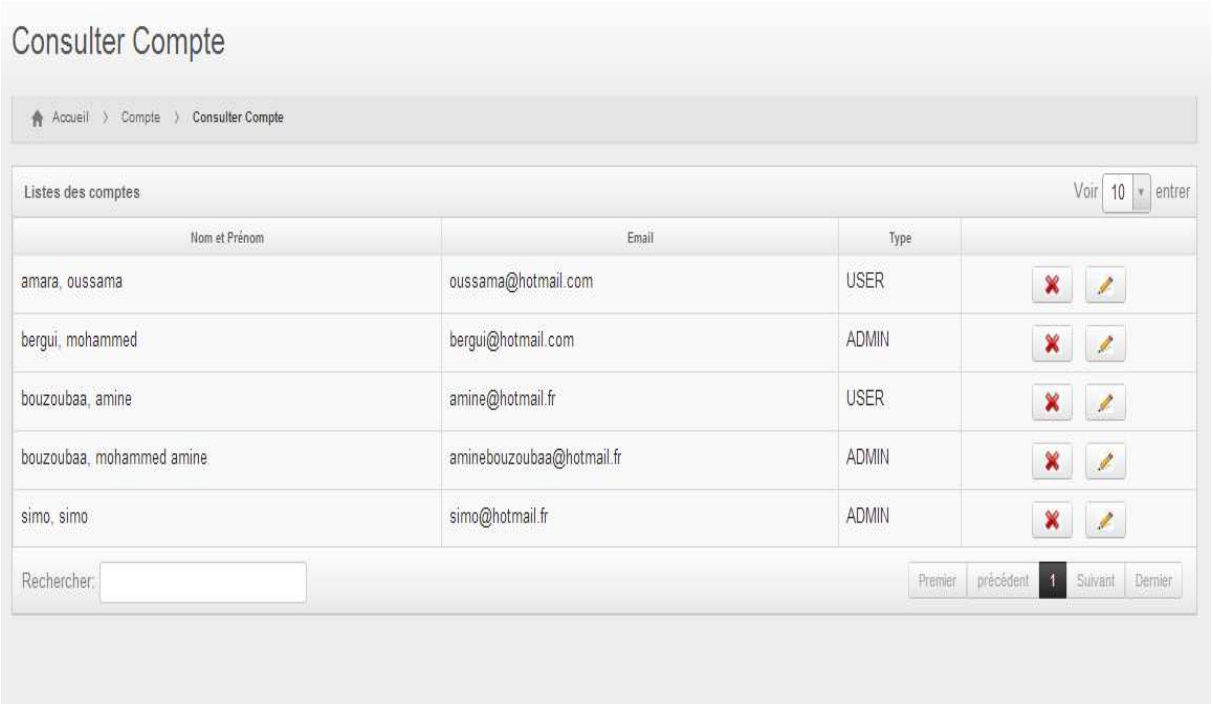


Figure 35 : Interface consulter compte

3.3. Les types de camion :

L'interface [Figure 36 : Interface ajouter camion] représente l'ajout d'un type de camion.

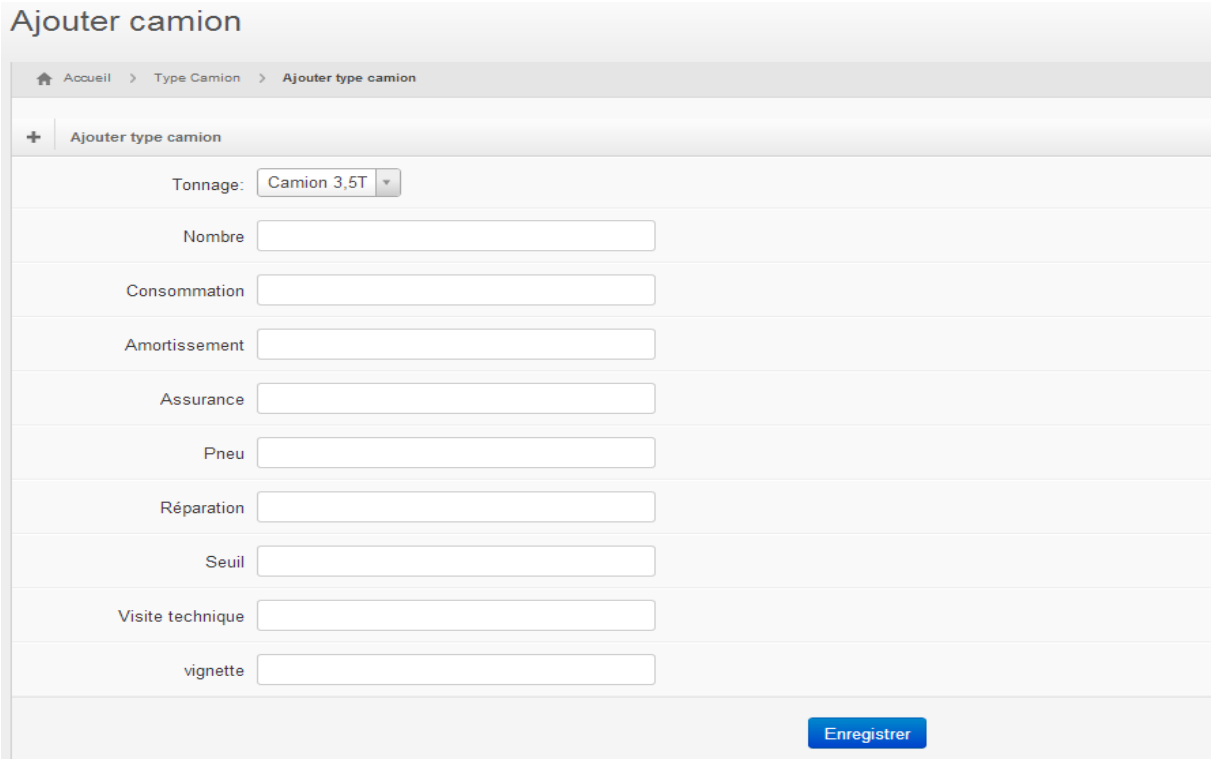


Figure 36 : Interface ajouter camion

Lors de la consultation des camions, l'Administrateur n'a pas le droit de supprimer un type de camion.

Figure 37 : interface consulter camion

3.4. Les villes

L'interface [Figure 38 : Interface ajouter ville] contient les champs à remplir pour créer une nouvelle ville.

Lors de l'ajout d'une ville son état est toujours activé sauf si l'utilisateur veut la désactiver au niveau de la consultation.

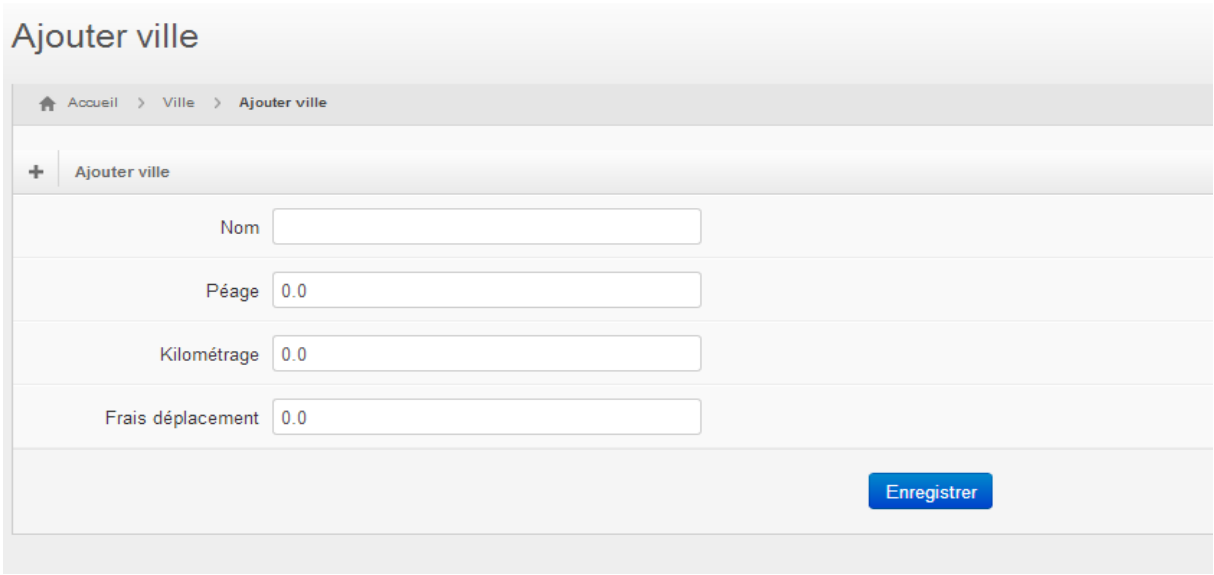


Figure 38 : Interface ajouter ville

Au niveau de la consultation, l'utilisateur a droit à 3 types d'actions : modifier, activer, désactiver.

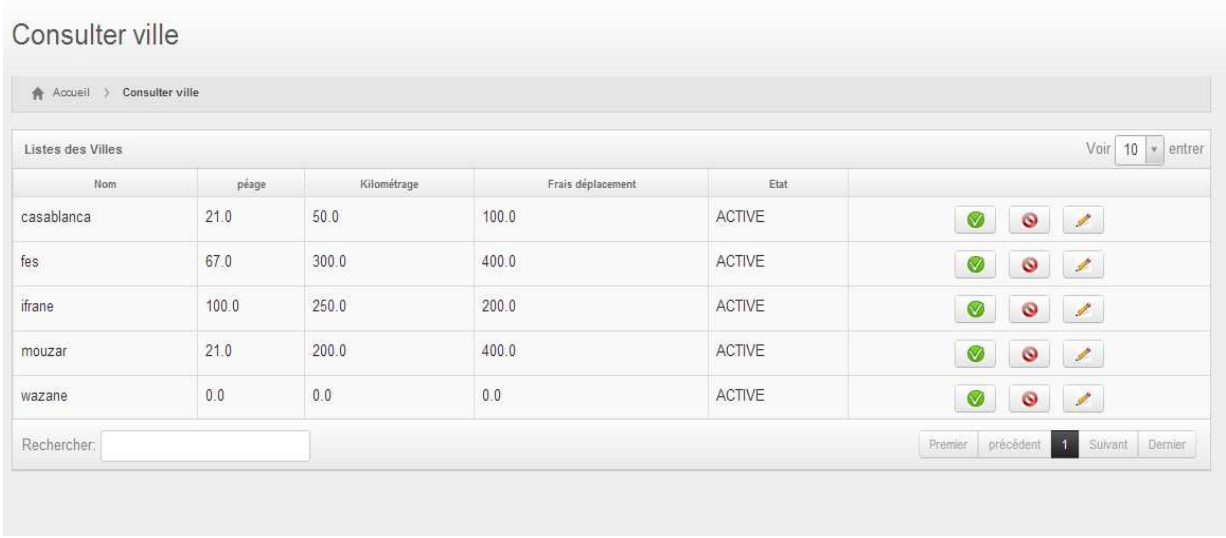


Figure 39 : Interface consulter ville

Conclusion générale :

Le service de transport de la SNTL Damcofaisait face à des problèmes de gestion des flux de transport et avait comme solution un outil de décision EXCEL. Notre travail consiste en la réalisation d'une application WEB qui permettra la gestion des voyages des camions et des bons de livraison.

Après avoir défini les fonctionnalités du système, élaboré différents diagrammes d'UML et réalisé une maquette du projet, nous nous sommes lancés au développement de l'application, tout en procédant par des tests unitaires.

Nous nous sommes basés sur la plateforme Java EE et les Framework JPA/Hibernate pour le mapping objet relationnel et Spring comme Framework transversal.

Notre travail s'étend sur une période de 6 mois, nous avons pu développer les modules Administration et Décision et on a comme perspectives :

- Développement du module bon de livraison.
- Amélioration du module administration.
- Amélioration de la gestion des erreurs.
- Mettre en place d'un système de journalisation.

Bibliographie

|1| **Pascal Roques, Franck Vallée.***UML2 en action: de l'analyse des besoins à la conception, 4ème édition.* Paris : Eyrolles, 2007.

|2| **Zeggar, Badrou.** Processus de développement en y, processus 2tup, février 2011. *www.zeggar.net.* [En ligne] février 2011. www.scribd.com/doc/49697489/Processus-de-Developpement-YProcessus.

|3| **RETAILLE, Jean Philippe.***Refactoring des applications Java/J2EE.* Paris : Eyrolles, 2005.

|4| **Dubois Julien, Retaillé Jean-Philippe, and Templier Thierry.***Spring par la pratique.* Paris : Eyrolles, 2007.

|5| **Cogoluègnes Arnaud, Templier Thierry, Dubois Julien, and Retaillé Jean-Philippe.***Spring par la pratique, 2ème edition edition.* Paris : Eyrolles, 2009.

|6| **Solomon, Fisher Tepper and Duskis.***Spring Persistence, a running start.* s.l. : Apress, 2009.

|7| **Wikipedia.** Apache Tomcat. *http://fr.wikipedia.org.* [En ligne] http://fr.wikipedia.org/wiki/Apache_Tomcat.

|8| MySQL 5.0 Reference Manual. *dev.mysql.com.* [En ligne] http://dev.mysql.com/doc/refman/5.0/fr/what-is.html.

|9| **SNYDER Bruce, VAN DE VELDE Thomas, DUPUIS Christian, LI Sing, HORTON Anne, and BALANI Naveen.***Beginning Spring Framework 2.* s.l. : Manning, 2005.

|10| **CHINE, Abderrazek.** Présentation, étape par étape, de Spring-DATA-JPA. *http://blog.netapsys.fr.* [En ligne] 24 Octobre 2012. http://blog.netapsys.fr/index.php/post/2012/10/23/Pr%C3%A9sentation-%C3%A9tape-par-%C3%A9tape-de-Spring-DATA-JPA.

|11| **Loïc Frering, Baptiste Meurant.** Tutoriel Hibernate/JPA - Spring2.5 - Tapestry5. *developpez.com.* [En ligne] 16 décembre 2008. http://loic-frering.developpez.com/tutoriels/java/hibernate-jpa-spring-tapestry/.

|12| **Mseddi, Ahmed.** JSR 303 (Bean Validation) : état des lieux. *http://blog.octo.com.* [En ligne] 7 septembre 2011 . http://blog.octo.com/jsr-303-bean-validation-etat-des-lieux/.

|13| Intellij IDEA. *www.01net.com.* [En ligne] http://www.01net.com/telecharger/linux/Programmation/fiches/100488.html.

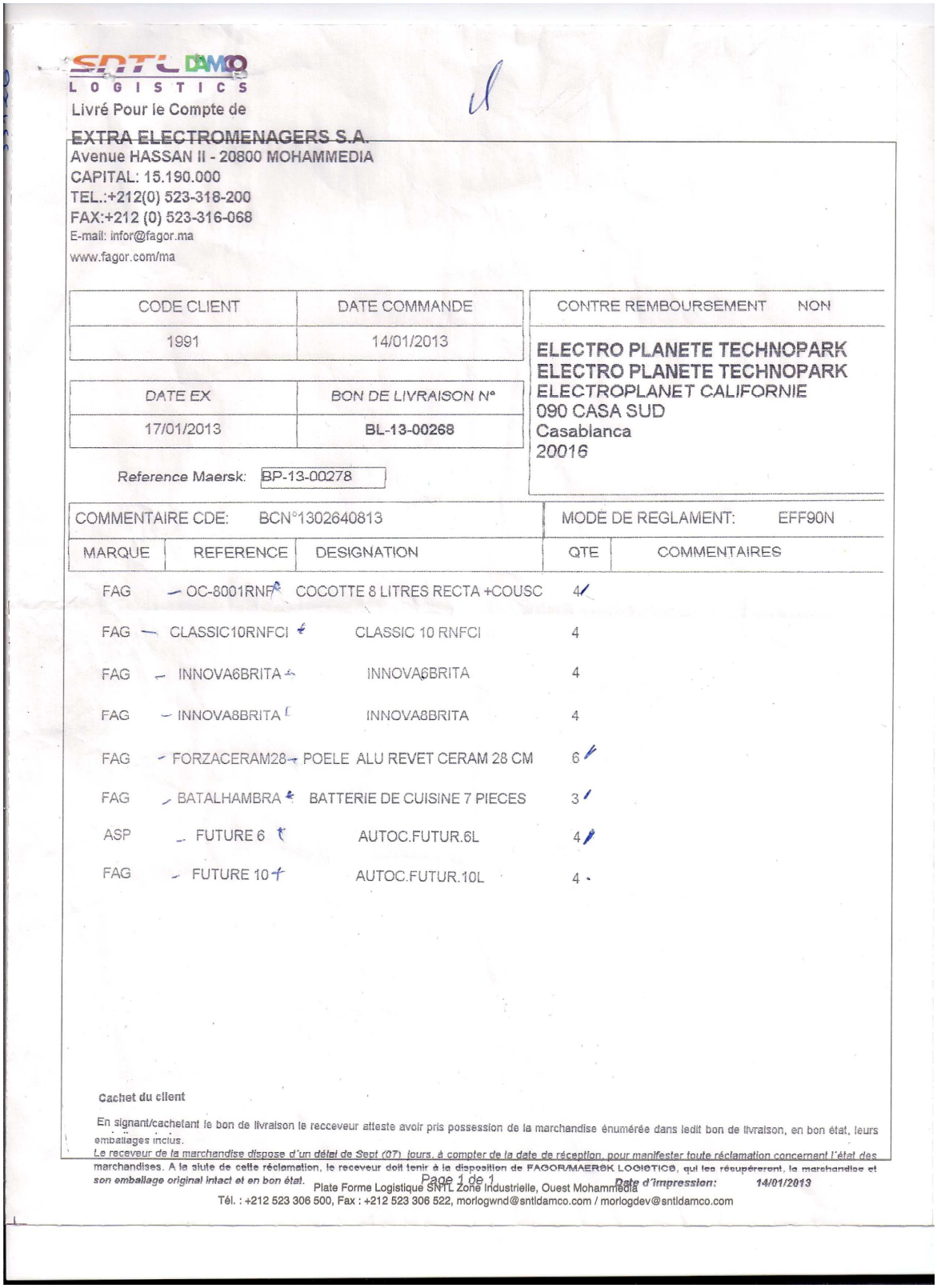
|14| **Sonatype Company.***Maven: The Definitive Guide.* Sebastopol, Californie : O'Reilly, 2008.

|15| **R.J, LORIMER.** Hibernate : Understanding lazy fetching. *www.javalobby.org.* [En ligne] août 2005. www.javalobby.org/java/forums/t20533.html.

|16| J2EE - Java 2 Enterprise Edition. *www.commentcamarche.net.* [En ligne] mai 2013. http://www.commentcamarche.net/contents/548-j2ee-java-2-enterprise-edition.

Annexe

Exemple d'un bon de livraison :



SNTL DAMCO LOGISTICS

Livré Pour le Compte de

EXTRA ELECTROMENAGERS S.A.
Avenue HASSAN II - 20800 MOHAMMEDIA
CAPITAL: 15.190.000
TEL.:+212(0) 523-318-200
FAX:+212 (0) 523-316-068
E-mail: infor@fagor.ma
www.fagor.com/ma

CODE CLIENT	DATE COMMANDE
1991	14/01/2013

CONTRE REMBOURSEMENT NON

ELECTRO PLANETE TECHNOPARK
ELECTRO PLANETE TECHNOPARK
ELECTROPLANET CALIFORNIE
090 CASA SUD
Casablanca
20016

DATE EX	BON DE LIVRAISON N°
17/01/2013	BL-13-00268

Reference Maersk: BP-13-00278

COMMENTAIRE CDE: BCN°1302640813 | MODE DE REGLEMENT: EFF90N

MARQUE	REFERENCE	DESIGNATION	QTE	COMMENTAIRES
FAG	OC-8001RNF	COCOTTE 8 LITRES RECTA +COUSC	4	
FAG	CLASSIC10RNFCI	CLASSIC 10 RNFCI	4	
FAG	INNOVA6BRITA	INNOVA6BRITA	4	
FAG	INNOVA8BRITA	INNOVA8BRITA	4	
FAG	FORZACERAM28	POELE ALU REVET CERAM 28 CM	6	
FAG	BATALHAMBRA	BATTERIE DE CUISINE 7 PIECES	3	
ASP	FUTURE 6	AUTOC.FUTUR.6L	4	
FAG	FUTURE 10	AUTOC.FUTUR.10L	4	

Cachet du client

En signant/cachetant le bon de livraison le recceveur atteste avoir pris possession de la marchandise énumérée dans ledit bon de livraison, en bon état, leurs emballages inclus.

Le receveur de la marchandise dispose d'un délai de Sept (07) jours, à compter de la date de réception, pour manifester toute réclamation concernant l'état des marchandises. A la aiute de cette réclamation, le receveur doit tenir à la disposition de FAGOR/MAERSK LOGISTICS, qui les récupérerent, la marchandise et son emballage original intact et en bon état.

Page 1 de 1 *Date d'impression:* *14/01/2013*

Plate Forme Logistique SNTL Zone Industrielle, Ouest Mohammedia
Tél. : +212 523 306 500, Fax : +212 523 306 522, morlogwnd@sntldamco.com / morlogdev@sntldamco.com

Détail de charges fixes et variables camions 3.5T :

CAMION 3,5T	
CHARGES	MONTANT(DH)
SALAIRE NET	5600
CHARGES SOCIALES CNSS + AT	1585
ASSURANCE VEHICULE	14000
VIGNETTE	1175
AMORTISSEMENT	7000
VISITE TECHNIQUE	620
VIDANGE	31000,28
PRIME DE L'AID	1900
COUT DE TELEPHONE	250
PNEU	23495,46667
TOTAL CHARGES FIXES	
• CARBURANT	
• PEAGE	
• DEPLACEMENT DE CHAUFFEUR	
• REPARATION	33383,46667
TOTAL CHARGES VARIABLES	
TOTAL	

Formule de calcul de la charge journalière d'un camion :

La charge journalière d'un camion = (Salaire net/26)+ (Charges sociales CNSS+AT/26)+(Assurances véhicule/365)+(Vignette/365)+(Amortissement/365)+ (Visite technique/365)+(Prime de aïd/365)+(Coût de téléphone/30) + (Vidange/365/Nombre de camions) +(Pneu/365/Nombre de camions)+(Réparation/365/Nombre de camions)

Tableau des prix par destinations pour les camions de type 3,5T :

Destination	Consonmation	Kilométrage	Litre de carburant	Prix de carburant	Charge fixe	Frais de déplacement	Péage	Prix total destination(DH)
AGADIR	15%	976	146,4	1200,48	370,06	200	476	2246,54
AZILAL	15%	610	91,5	750,3	370,06	100		1220,36
BEN SLIMANE	15%	94	14,1	115,62	370,06	70		555,68
BENI MELLAL	15%	486	72,9	597,78	370,06	100		1067,84
BERRECHID	15%	118	17,7	145,14	370,06	70	70	655,2
BOULMANE	15%	642	96,3	789,66	370,06	100		1259,72
CASA	15%	100	15	123	370,06	30		523,06
CHEFCHAOUEN	15%	630	94,5	774,9	370,06	120	104	1368,96
EL JADIDA	15%	248	37,2	305,04	370,06	70	76	821,1
ERRACHIDIA	15%	1074	161,1	1321,02	370,06	240		1931,08
ESSAOUIRA	15%	856	128,4	1052,88	370,06	100		1522,94
FES	15%	544	81,6	669,12	370,06	100	202	1341,18
HOCEIMA	15%	1098	164,7	1350,54	370,06	240	262	2222,6
IFRANE	15%	542	81,3	666,66	370,06	100	162	1298,72
KALAA DE SRAGHNA	15%	548	82,2	674,04	370,06	100	90	1234,1
KENIFRA	15%	598	89,7	735,54	370,06	100		1205,6
KENITRA	15%	232	34,8	285,36	370,06	80	102	837,42
KHEMISSAT	15%	310	46,5	381,3	370,06	80	122	953,36
KHOURIBGA	15%	282	42,3	346,86	370,06	100	70	886,92