République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab, Blida, USDB.

Faculté Des Sciences
Département Informatique

Mémoire Pour L'obtention
Du Diplôme Master LMD en informatique
Option : *ingénierie de logiciel*

Sujet :

Conception et réalisation d'un système d'information de Gestion des billetteries

Présenté par :
- CHIBANI Noureddine
- MADI Azzedine

Proposer et diriger par :
- *Dr BOUSTIA Narhimene*
- *Mr MOHELLEBI Samir*

Devant le jury composé de :

Soutenue le: 03/06/2011

Nom présidente du jury : Hamouda
Nom examinateur 1 : Azzouz
Nom examinateur 2 : Mustapha

2010/2011

REMERCIEMENT

Avant toute chose, nous tenons à remercie le Dieu le tout puissant de nous avoir donné la volonté, la patience et le courage de poursuivre et d'achever ce travail dans les bonnes conditions.

Nos remerciements également notre promotrice ***Mlle BOUSTIA****, pour nous avoir encadré, diriger et conseiller tout au long de ces mois de travail ainsi pour son aide précieuse lors de la correction de ce mémoire ;*

Nous n'oublierons pas le soutien constant et les encouragements de notre grande famille.

En fin, on tient à remercier les membres de jury qui vont faire l'honneur d'apprécier notre travail.

Merci

Dédicace

Je dédie ce modeste travail à mes très chers parents qui m'ont toujours soutenu pendant toute ma vie, je leur souhaite le bonheur et la bonne santé.

A toute ma grande famille ; mes frères: ***Boualam, Ibrahim et Mohamed***

A toute ma famille oncles tantes cousins cousines ;

A mon compagnon dans ce parcours – Azzedine et sa grande famille;

Et a tous mes amis, et en particulier à : ***Amine, Abd el razzak, Maamer, Youssef, Mostafa.***

A toute personne porte des sentiments pour moi ;

Merci à tous ceux qui m'ont aidé de près ou de loin

Noureddine

TABLE DES MATIERES

Liste des figures :

LISTE DES TABLEAUX

Résumé :

Nous proposons dans le cadre de ce projet de fin d'études, d'appliquer la méthode **2TUP** au problème de la gestion des billetteries dans **L'entreprise Nationale du Transport Maritime des Voyageurs** , La gestion d'un cycle de vie de logiciel requit un grand sens de la rigueur et de l'adaptation aux changements continuels imposés au monde de l'entreprise. C'est pour ça que nous avons choisi d'axer notre travail sur une méthode de développement qui est née de travaux poussés vers la standardisation et la flexibilité, et ce pour répondre aux contraintes actuelles de gestion des développements.

Les mots clés :

Gestion des billetteries, la méthode 2TUP, architecteur 3 tiers, MVC.

Introduction générale

1. INTRODUCTION :

De nos jours, une des tendances les plus en vue et qui concerne tous les secteurs de développement, est l'*informatisation*.
Depuis l'apparition de l'informatique et son introduction dans le monde économique, les entreprises et entités publiques aspirent à optimiser et à rendre fiable la gestion de leur structure interne.

L'entreprise Nationale du Transport Maritime des Voyageurs (E.N.T.M.V) vend plusieurs billets chaque jour qu'ils sont difficiles de les gérer bien précise

Actuellement les divers composants du système étudié sont réalisés d'une manière anarchique car aucune analyse ni méthode de conception de systèmes d'informations n'a été utilisée afin de réaliser des applications selon un modèle bien défini. L'information est éparpillée dans plusieurs bases de données indépendantes ce qui pose des problèmes de gestion des données. Ces bases de données sont développées avec l'ACCESS et géré avec le QR (un logiciel de saisie monoposte sous MS-DOS.). Dans la version actuelle, le composant orienté vers le support du service des billetteries n'a pas encore été conçu et réalisé

Notre but est d'appliquer une méthode rigoureuse de conduite d'un projet réel. Le projet en question concerne la gestion des billetteries

Ce projet a pour objectif principal de proposer une solution à un problème concret, et ceci en partant d'une définition des besoins. Nous espérons à travers celui-ci, démontrer l'importance de l'application d'une méthodologie de développement, et aussi permettre par la suite que ce produit puisse être évolutif et facilement maintenable par des intervenants tiers.

2. STRUCTURE DU MEMOIRE :

. Le mémoire sera découpé en 5 grands chapitres :

1. **Introduction générale :** nous parlerons dans ce chapitre les problématiques et les objectifs que nous voudrions atteindre.

2. **Présentation de l'entreprise** : nous parlerons dans ce chapitre sur l'architecteur et l'organisation de l'entreprise et aussi les moyens de l'entreprise

3. **La méthode 2TUP** : ce chapitre contient les définitions des différents concepts utilisés dans ce document.

4. **Conception du logiciel** : c'est ici que nous allons appliquer la méthode 2TUP au problème de la *gestion des enseignements* en suivant les phases suivantes : ***l'étude préliminaire***, ***capture des besoins fonctionnels***, ***analyse***, **conception** .

5. **Réalisation :** Dans ce chapitre nous allons d'abord démontrer le fonctionnement de notre application
6. **Conclusion générale**

CHAPITRE 1 : Présentation de l'entreprise

1. introduction :

L'entreprise est une organisation dont l'activité est la combinaison d'un certain nombre de moyens humains et matériel par la production d'un bien ou d'un service

La description de ces moyens et les relations existantes entre eux définit la structure organisationnelle de l'entreprise. Cette structure dépend du milieu dont lequel se trouve l'entreprise (son environnement) de la nature de l'activité, de la taille et la longueur de son cycle d'exploitation.

C'est à partir de ces définitions qu'on procède dans ce chapitre à l'analyse de la structure organisationnelle de l'***E N T M V*** et cela à travers l'organigramme qui est une représentation schématique de la structure.

On, va ainsi faire un tracé sur l'***E N T M V*** en générale, son historique, ses principales missions, et son potentiel et réalisation.

2. présentation de l'organisme d'accueil :

2.1 Présentation de l'entreprise :

Est une entreprise de prestation de service qui assure essentiellement le transport maritime qui est un élément important de par sa contribution au développement économique et à l'aménagement du territoire.

2.2. Historique :

L'entreprise Nationale du Transport Maritime des Voyageurs « ENTMV Algérie- Ferries » a été créée par décret *n° 87-155 du 14 juillet 1987.*

Elle est issue de la restructuration de la *SNTM / CNAN* au sein de laquelle l'activité de transport maritime de voyageurs était décentralisée avant d'être érigée en Unité de Production Autonome puis en Entreprise Socialiste.

A compter du 07 avril 1990, *l'ENTMV* prend le statut d'E.P.E en application du décret n° 88-01 du janvier 1988 portant loi d'orientation sur les Entreprise Publique Economique.

Sa capitale sociale qui s'élevait à l'origine à 15 millions de Dinars est passée à 405 millions de Dinars en décembre 1997.

2.3 Mission :

Les textes portant création de l'entreprise fixent pour mission à l'ENTMV :

- Le transport maritime de passagers et des autos- passagers.
- Le transport maritime du fret.
- Les activités annexes au transport maritime et notamment la consignation des navires de transport de passagers, la représentation générale et la vente de billetterie.

2.4 Moyens de production :

L'ENTMV dispose pour l'accomplissement de sa mission d'une flotte de transport d'un réseau d'agences commerciales de vente de billetterie de voyages. De centres de consignation de navires ainsi que de structures organisationnelles, à savoir :

- ♣ Un siège social comprenant cinq Direction Opérationnelles.
- ♣ Trois Direction Régionales implantées en Algérie : Oran, Alger et Annaba.
- ♣ Une délégation Régionale à l'étranger implantée à Marseille.

L'ENTMV emploie actuellement un effectif global d'environ **1600** personnes dont **1000** navigants et **600** sédentaires.

Le personnel navigant comprend deux catégories : **360** en personnel de conduite et **640** en personnel d'hôtellerie.

Le personnel sédentaire est réparti à raison de **275** personnes au niveau du siège Social et **325** au niveau des Structures Régionales dont **23** personnes à la Délégation à l'étranger.

2.4.1 Flotte de transport :

La flotte de transport maritime de voyageurs de ***l'ENTMV*** composée à l'origine de navires propriété a été renforcée à la fin de la décennale quatre-vingt-dix par des navires affrétés en raison du développement du marché et en vue de pallier aux risques de défaillance d'ancienne flotte vieillissante.

Depuis 1995, ***l'ENTMV*** a procédé graduellement au renouvellement de l'ancienne flotte héritée de la CNAN par l'acquisition de trois nouveaux car-ferries désignés ci-après :

Natures	Années et lieux de construction	Années de mise en service
TARIQ IBN ZIYAD **TASSILI II** **EL-DJAZAIR II**	**1995-Espagne** **2004-Espagne** **2005-Espagne**	**1996-Janvier** **2004-Novembre** **2005-Mai**

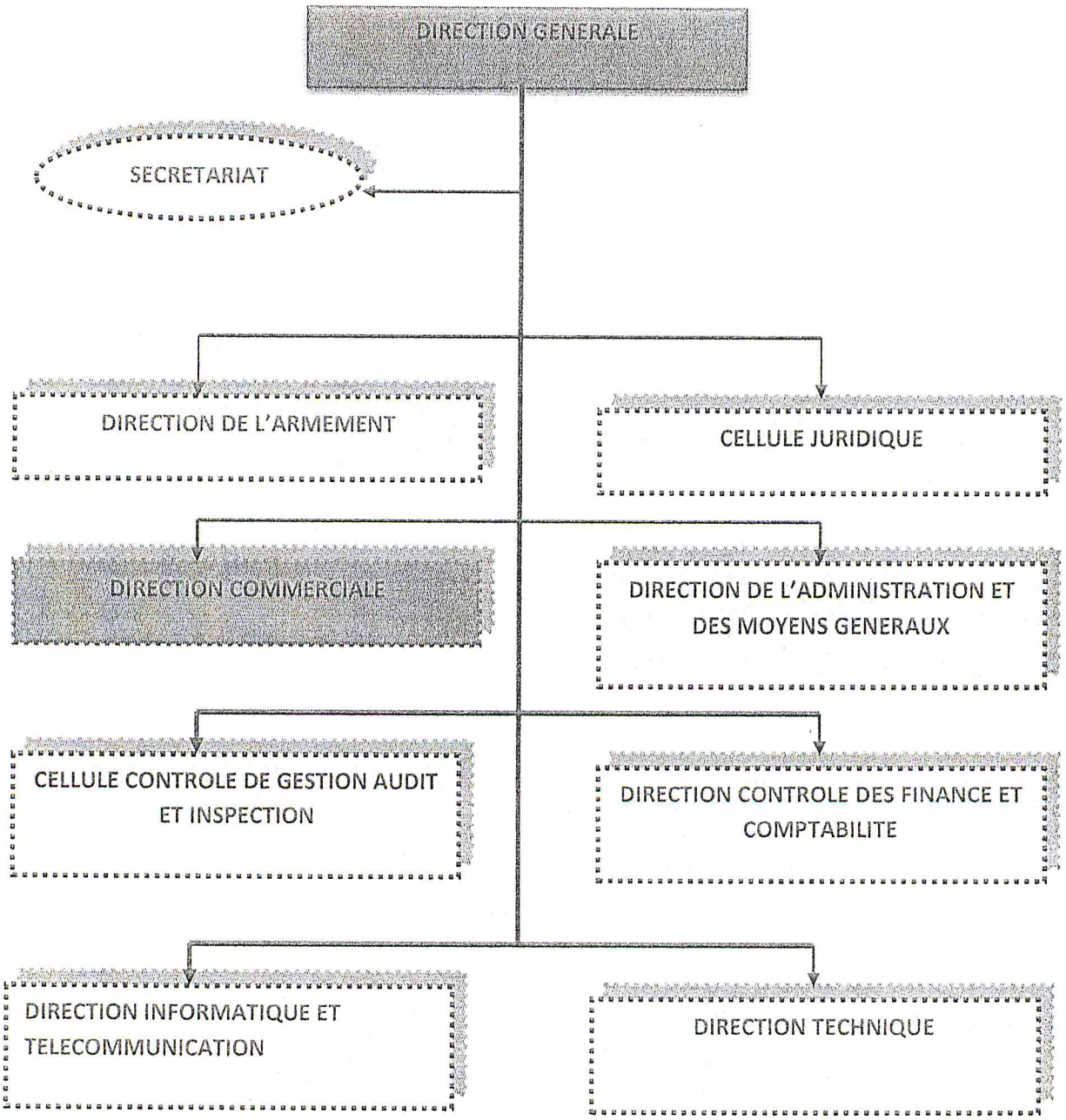
Tableaux 1 : les trois nouveaux car-ferries

2-5 Organigramme :

Description des organigrammes :

La direction générale se compose de plusieurs directions qui sont représentées dans l'organigramme général.

L'agence représente le cœur de structure d'accueil, qui bat pour donner vie aux autres organes principaux.



Organigramme générale de l'entreprise

2-6 les structures fonctionnelles de l'entreprise :

- *L'E.N.T.M.V* est composées de plusieurs directions qui sont :

Direction des finances et de la comptabilité.

Ses missions principales sont :

- La définition, la mise en œuvre et le contrôle des procédures et méthodes de gestion financière et comptable.
- La mise en œuvre de la comptabilité.
- L'établissement des bilans (et document annexes) de l'entreprise.
- La tenue à jour du manuel de comptabilité de l'entreprise.
- L'élaboration et la mise en place des procédures d'établissement et de suivi des budgets d'exploitation et d'investissement.
- L'élaboration du budget général d'exploitation et d'investissement et des rapports périodiques d'exécution du budget.
- Mise en place des procédures de gestion et de contrôle de la trésorerie.
- Gestion et suivi des contrats de financement des investissements.
- Suivi financier des contrats d'approvisionnement.
- Gestion et suivi des comptes bancaires et de chèque postaux.
- Vérification de la conformité des états de paie avec la réglementation en vigueur et les dispositions du règlement intérieur de l'entreprise.
- Alimentation des comptes des délégations régionales des agences en Algérie et à l'étranger.

Le directeur central des finances et de la comptabilité est assisté de :

1- Une sous-direction Finances et budget.
2- Un sous-directeur comptabilité.
3- Une sous-direction Audit Financier et comptable.

Direction technique :

Ses missions principales sont :

- Le maintien des car-ferries en conformité avec les règles de sécurité.
- Définir de mettre en œuvre et de contrôle les méthodes et procédures de gestion technique de la flotte.
- Le suivi permanent de l'état des navires (pont machines et espèces passagers)
- La réalisation des études techniques relative aux aménagements des cars ferries et de la construction neuve.
- L'information permanente sur le marché de la construction neuve et de la réparation.

- L'élaboration et la mise en œuvre de la politique de l'entreprise en matière d'approvisionnement technique notamment en pièces détachées et matière consommables.
- Le maintien des car-ferries en conformité avec les règles de sécurité et au respect de la réglementation à bord en matière de sécurité.
- Définir les normes d'équipement et d'exploitation matière de navigation et de sécurité.
- Définir, de mettre en œuvre et de control la politique de maintenance des car-ferries et la standardisation des équipements et de promouvoir les méthodes et procédures d'une gestion technique performante.
- Analyser et préconiser les mesures à prendre pour tous les problèmes techniques.
- Définir les procédures de commande et les normes stockage de gestion des stocks et de contrôle de consommation de ces produits.

Le directeur technique est assistée de :

1- Un sous-directeur technique.
2- Un sous-directeur approvisionnement.
3- Inspecteurs bord sécurité.
4- Services études.

Direction de l'armement :

Ses missions principales sont :

- Equipement le navire en hommes. Equipement et matière consommable.
- Elaboration et mise en place de la politique de l'entreprise en matière de gestion du personnel navigant.
- Définition du plan de carrière du personnel navigant.
- Suivi de l'évolution des effectifs navigant au sein des car-ferries et gestion des embarquements.
- Recueil et diffusion des textes concernant législation du travail et les règlements internationaux.
- Contrôle du respect de l'application de ces textes.
- Elaboration et diffusion des règles de travail à bord des CF.
- Participation aux études relatives à l'organisation du travail à bord tendant à favoriser la sécurité la rentabilité et l'efficacité.
- Participation à la définition et à l'organisation d'une meilleure vie à bord.

Le directeur de l'armement est assisté d'un sous-directeur.

Direction informatique et télécommunication :

Ses missions principales sont :

- Conception et mise en place d'une méthodologie de développement adaptée aux objectifs retenus.
- Conception et réalisation d'application informatiques.
- Prise en charge de la saisie et de la préparation.
- Arrêté les mesures conservatoires pour assurer la sécurité des donnée
- Assurer la fiabilité du système de réservation.
- Maintenance des applications opérationnelles.
- Réalisation et suivi du plan informatique de l'entreprise.
- Formulation suivie et contrôle des normes de fonctionnement du centre de calcul
- Définir et asseoir une procédure homogène de réservation par voit informatique.

La directeur informatique est assisté de :

1- Un chef de département exploitation.
2- Un chef de département système.
3- Un chef de département études.
4- Un chef de département réservation.

2-7 Présentation de la structure d'accueil :

2.7.1 Mission et responsabilité :

La direction commerciale est un organe chargé de la commercialisation du produit (transport), auprès de ses différentes agences et structures commerciale.

A ce titre, elle a pour missions principales :

- L'élaboration de la politique commerciale de l'entreprise en matière de transport de passagers et autos passagers, de fret et de proposer les axes de son développement.
- L'élaboration des horaires en collaboration avec les structures concernées et en coordination avec les partenaires étrangers.
- Le suivi des rotations et mouvement des navires.
- La préparation des données utiles à la détermination des objectifs des ventes.
- Le suivi des ventes.
- L'élaboration des manifestes.
- La gestion des contrats et autre documents relatifs aux commissions de représentation de consignation et aux tarifs portuaires.
- Centralisation des comptes d'escales.
- Le règlement des litiges pouvant naître des comptes d'escales concernant des écarts.
- Le suivi des transferts après transmission des manifestes et contrôle d'usage.

- Le suivi de la consignation des navires transporteurs de passagers.
- Mener les actions de développement du réseau vente.
- Arrêter le programme publicitaire conformément aux orientations de l'entreprise.
- Concevoir les documents publicitaires et se charger de leur diffusion.
- Fixe les dotations hôtelières pour chaque navire.

Ses structures fonctionnelles sont :

- Délégation régionale.
- Département finance et comptabilité.
- Service précontrôle.

Agence commerciale :
Il est composé de :

Le chef d'agence :

- Transmet une demande, en précisant le type de billetterie et la quantité.
- Reçoit la billetterie.
- Etablis l'état récapitulatif des émissions par quinzaines.
- Etablis la situation mensuelle.

Le chef de comptoir :

- Fourni la quantité de la billetterie aux agents de comptoir.
- Signe les oppositions faites par les clients.
- Fait le suivi du stock.

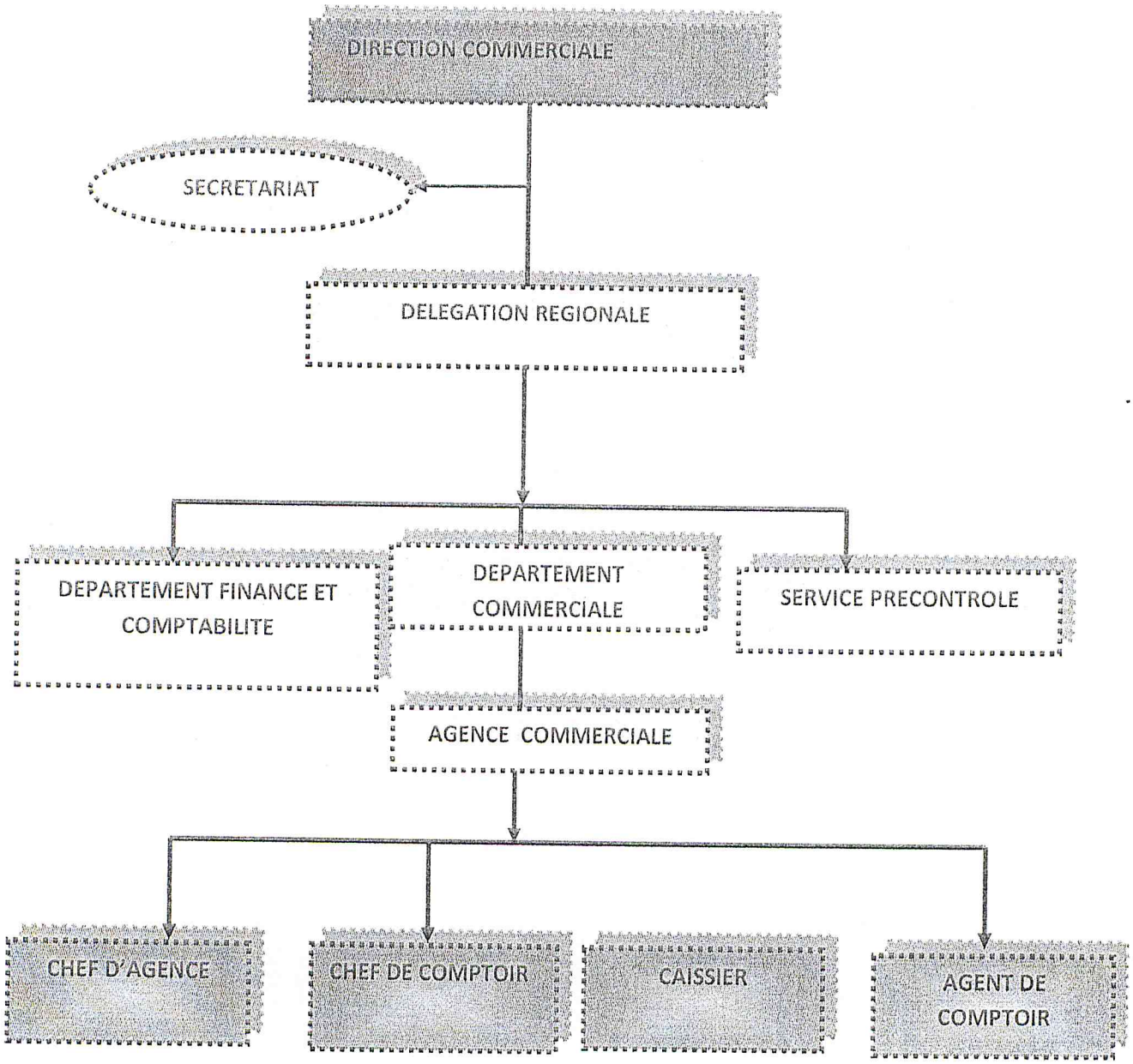
Agent de comptoir :

- La vérification de l'identité du client.
- La réservation en fournissant toutes les informations possibles.
- Les remboursements.
- Les modifications de dates.
- Les annulations.
- Les sur classements.
- Les modifications des trajets.
- Détache les souches non utilisables, et établit le billet.

Le caissier :

- Remplir le brouillard de caisse.
- Remet le billet au client.
- Encaisse l'argent, et fait la transmission à la banque.
- Reçoit les chèques.

2-8 Organigramme de la structure d'accueil :



CHAPITRE 2 : La méthode 2TUP

2. COMPARAISON ENTRES LES DIFFERENTES APPROCHES :

a. Modélisation :

Par le passé, le modèle *Entité-Relation* représentait une grande partie des approches les plus utilisées.

Actuellement, les nouvelles technologies s'appuient sur le modèle objet. En termes d'analyse et de modélisation objet, UML est devenu un standard incontournable, stabilisé, industriel.

b. Conduite de projet :

Au début, le cycle en cascade (ex : le cycle en V) était très utilisé. Mais on a vite constaté son incapacité à s'adapter aux différents changements.

Une méthode semi-itérative est apparut (RAD) pour pallier aux carences de ce dernier.

L'itératif s'est ensuite imposé, parce qu'il réduit la complexité de réalisation des phases, en travaillant par approches successives et incrémentales.

Une méthode fortement axée sur l'itératif et le modèle UML est alors apparut, il s'agit de **UP (Unified Process).** Cette méthode comme son nom l'indique, a été le fruit de travail de plusieurs personnes voulants « *unifier* » les différentes méthodes objets existantes à ce moment comme *Booch*, OMT et *OOSE*.

On constate aujourd'hui, l'émergence d'une nouvelle approche : les méthodes agiles (Agile Modeling). C'est des méthodes itératives à planification souple qui leur permettent de s'adapter à la fois aux changements du contexte et de spécifications du projet. (Chromatic, 2005)

3. DEFINITION D'UN PROCESSUS DE DEVELOPPEMENT LOGICIEL :

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent 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. [Rocques & Vallée, 2004]

4. LE PROCESSUS UNIFIÉ :

Le Processus Unifié (PU ou UP en anglais pour **Unified Process**) est une méthode de développement logiciel construite sur UML; elle est *itérative* et *incrémentale*, *centrée sur l'architecture*, *conduite par les cas d'utilisation* et *pilotée par les risques*.

- Itérative et incrémentale : la méthode est itérative dans le sens où elle propose de faire des itérations lors de ses différentes phases, ceci garantit que le modèle construit à chaque phase ou étape soit affiné et amélioré. Chaque itération peut servir aussi à ajouter de nouveaux incréments.

- Conduite par les cas d'utilisation : elle est orientée utilisateur pour répondre aux besoins de celui-ci.

- Centrée sur l'architecture : les modèles définit tout au long du processus de développement vont contribuer à établir une architecture cohérente et solide.

- Pilotée par les risques : en définissant des priorités pour chaque fonctionnalité, on peut minimiser les risques d'échec du projet.

La gestion d'un tel processus est organisée d'après les 4 phases suivantes :

1. **Préetude(Inception) :** c'est ici qu'on évalue la valeur ajoutée du développement et la capacité technique à le réaliser (étude de faisabilité).

2. **Elaboration :** sert à confirmer l'adéquation du système aux besoins des utilisateurs et à livrer l'architecture de base.

3. **Construction :** sert à livrer progressivement toutes les fonctions du système.

4. **Transition :** déployer le système sur des sites opérationnels.

Chaque phase est elle-même décomposée séquentiellement en itérations limitées dans le temps (entre 2 et 4 semaines). Le résultat de chacune d'elles est un système testé, intégré et exécutable. L'approche itérative est fondée sur la croissance et l'affinement successifs d'un système par le biais d'itérations multiples. Le système croît avec le temps de façon incrémentale, itération par itération, et c'est pourquoi cette méthode porte également le nom de développement itératif et incrémental. Il s'agit là du principe le plus important du Processus Unifié.

Ces activités de développement sont définies par 6 disciplines fondamentales qui décrivent *la capture des besoins, la modélisation métier, l'analyse et la conception, l'implémentation, le test et le déploiement.*

Notons que ces différentes étapes (ou disciplines) peuvent se dérouler à travers plusieurs phases.

Le processus unifié doit donc être compris comme une trame commune des meilleures pratiques de développement. [Rocques & Vallée, 2004].

5. LE PROCESSUS 2TUP :

On dit de la méthode **UP** qu'elle est *générique* c.à.d. qu'elle définit un certain nombre de critères de développement, que chaque société peut par la suite personnaliser afin de créer son propre processus plus adapté à ses besoins.

C'est dans ce cadre que la société *Valtech* a créé la méthode **2TUP.**

2TUP signifie « 2 Track Unified Process» .C'est un processus qui répond aux caractéristiques du **P**rocessus **U**nifié. Le processus 2TUP apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information de l'entreprise. En ce sens, il renforce le contrôle sur les capacités d'évolution et de correction de tels systèmes.

« 2 Track» signifie littéralement que le processus suit deux chemins. Il s'agit des « chemins fonctionnels » et « d'architecture technique », qui correspondent aux deux axes de changement imposés au système d'information [Rocques & Vallée, 2004].

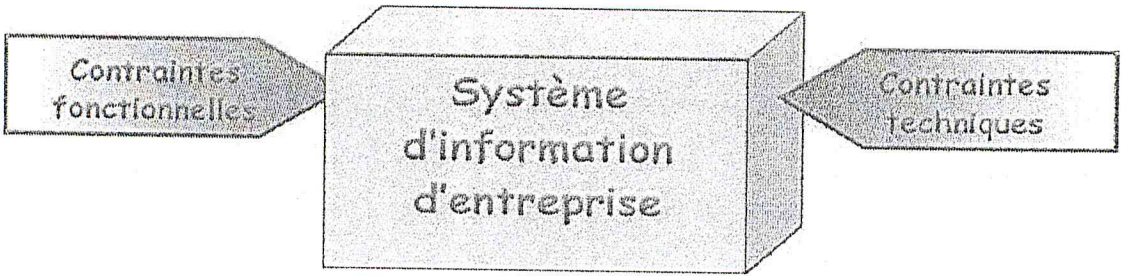


Figure 1 : Le système d'information soumis à deux types de contraintes

La branche gauche (fonctionnelle) : capitalise la connaissance du *métier* de l'entreprise. Elle constitue généralement un investissement pour le moyen et le long terme.
Les fonctions du système d'information sont en effet indépendantes des technologies utilisées.
Cette branche comporte les étapes suivantes :

- La capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs.
- L'analyse.

La branche droite (architecture technique) : capitalise un savoir-faire technique. Elle constitue un investissement pour le court et moyen terme. Les techniques développées pour le système peuvent l'être en effet indépendamment des fonctions à réaliser.
Cette branche comporte les étapes suivantes :

- La capture des besoins techniques.
- La conception générique.

La branche du milieu : à l'issue des évolutions du *modèle fonctionnel* et de *l'architecture technique*, la réalisation du système consiste à *fusionner* les résultats des 2 branches. Cette fusion conduit à l'obtention d'un processus en forme de Y.
Cette branche comporte les étapes suivantes :

- La conception préliminaire.
- La conception détaillée.
- Le codage.
- L'intégration.

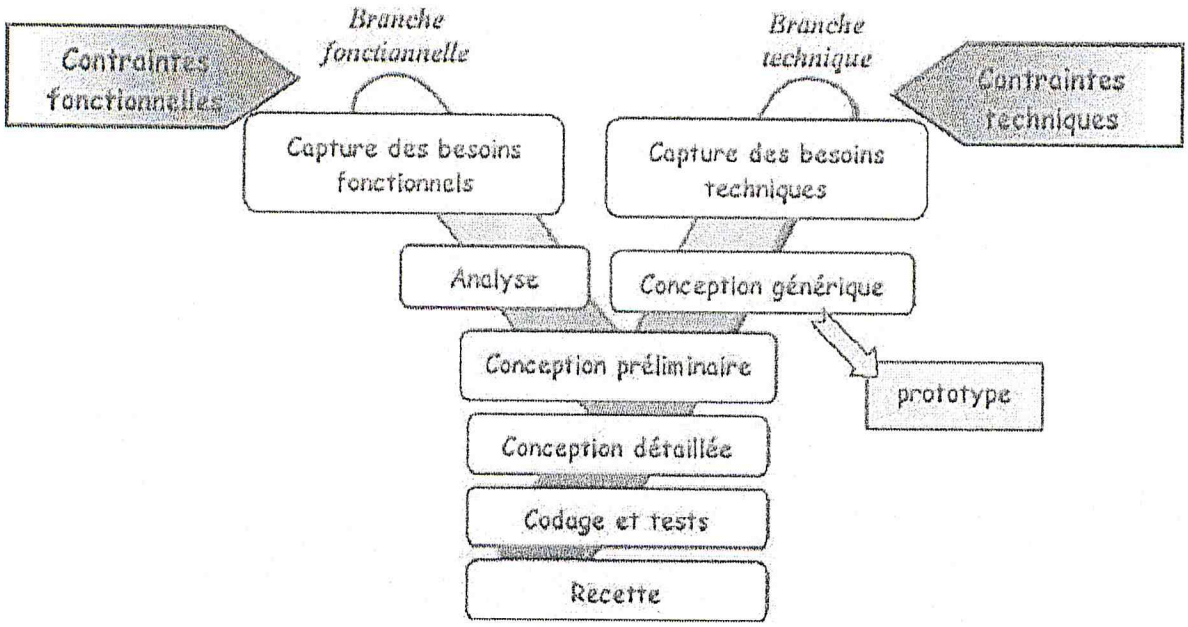


Figure 2 : Le processus de développement en Y

6. UN PROCESSUS DE MODELISATION AVEC UML :

Le processus 2TUP s'appuie sur UML tout au long du cycle de développement, car les différents diagrammes de ce dernier permettent par leur facilité et clarté, de bien modéliser le système à chaque étape.

« Unified Modeling Language » : UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier, concevoir des solutions et communiquer des points de vue. [Pitman, 2006]

UML unifie à la fois les notations et les concepts orientés objet.il ne s'agit pas d'une simple notation, mais les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage, c'est pour ça qu'UML est présenté parfois comme une méthode alors qu'il ne l'est absolument pas.

UML unifie également les notations nécessaires aux différentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis la définition des besoins jusqu'au codage. [Roques, 2006]

Voici une présentation rapide des différents diagrammes UML qui vont être utilisés tout au long du projet :

- **Le diagramme des cas d'utilisation :** représente la structure des fonctionnalités nécessaires aux utilisateurs du système. Il est normalement utilisé lors des étapes de capture des besoins fonctionnels et techniques.

- **Le diagramme d'activités :** représente les règles d'enchaînement des activités et
  Actions dans le système. Il peut être assimilé comme un algorithme mais schématisé.

- **Le diagramme de packages :** présent depuis UML 2.0, ce diagramme modélise des catégories cohérentes entre elles, pour un souci de partage des rôles.
  Correspond à l'étape de modélisation des différents scénarios d'un cas d'utilisation.

- **Le diagramme de classes :** sûrement l'un des diagrammes les plus importants dans un développement orienté objet. Sur la branche fonctionnelle, ce diagramme est prévu pour développer la structure des entités manipulées par les utilisateurs.
  En conception, le diagramme de classes représente la structure d'un code orienté objet.

- **Le diagramme de séquence :** représente les échanges de messages entre les objets, dans le cadre d'un fonctionnement particulier du système.

- **Le diagramme d'états :** représente le cycle de vie d'un objet. Il spécifie les états possibles d'une classe et leur enchainement.
  Ce diagramme est utilisé lors des étapes d'analyse et de conception.

CHAPITRE 3 : CONCEPTION DU LOGICIEL

1. ETUDE PRÉLIMINAIRE :

L'étude préliminaire (ou Préetude) est la toute première étape du processus 2TUP. Elle consiste à effectuer un premier repérage des besoins fonctionnels et opérationnels, en utilisant principalement le texte, ou diagrammes très simples. Elle prépare les activités plus formelles de capture des besoins fonctionnels et de capture techniques [Rocques & Vallée, 2004].

1.1. Grands choix techniques

On va utiliser une approche itérative et incrémentale, fondée sur le processus en Y (2TUP*) et on a choisis comme techniques clés pour ce projet :

- la modélisation objet avec **UML**.
- les architectures 3-tiers avec **SGBDR**.
- Utilisation des *Design Patterns* (**MVC**,...).
- la plate-forme Java (avec **JSP** et **JDBC**).
- Utilisation de l'environnement de développement intégré **Eclipse**

Remarque : la branche droite qui désigne l'étude de l'architecture technique va être ignorée, du fait du manque de temps à notre disposition et de la complexité de la chose.

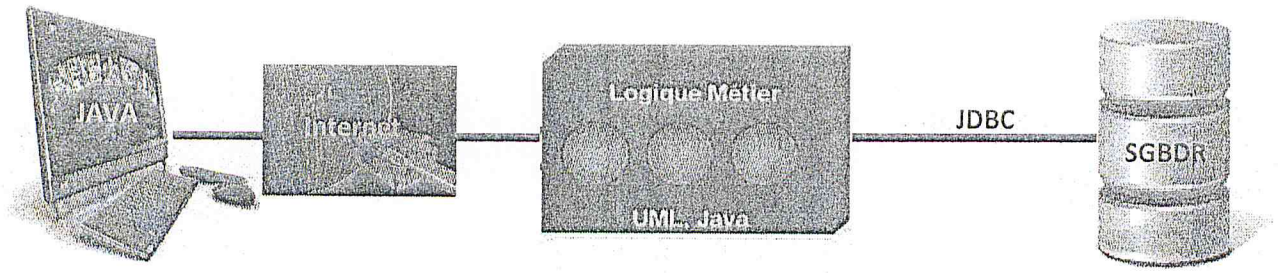


Figure 3 : Architecture logicielle préliminaire

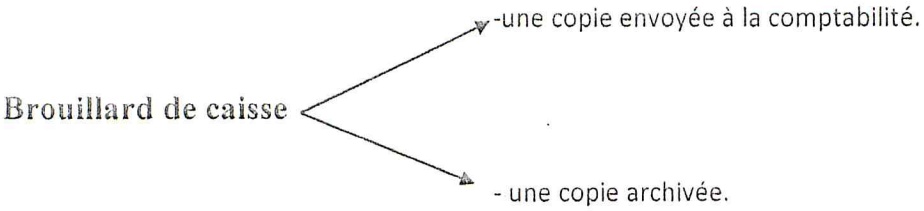
*2TUP: Two Track Unified Process

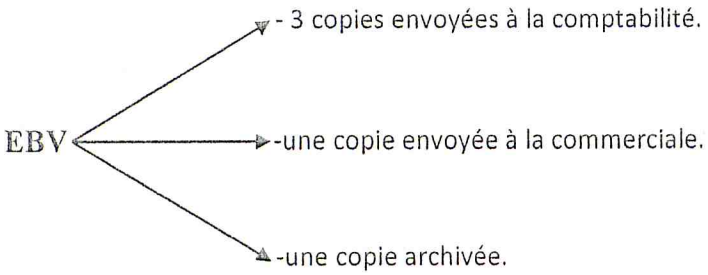
1.2. Recueil des besoins fonctionnels :

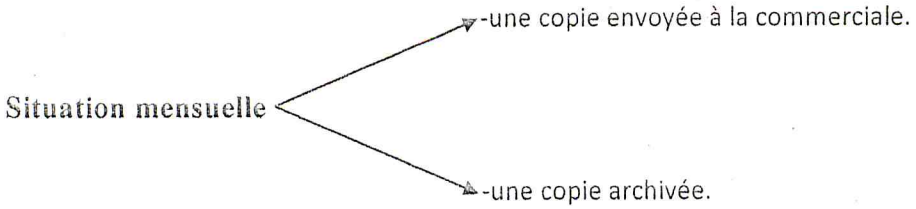
Un premier tour d'horizon des besoins exprimés par les employés de l'entreprise a permis d'établir le cahier des charges préliminaire suivant :

Traitement des billets vendus :

Les détails des billets vendus sont saisis dans un logiciel QR* au niveau de l'agence à partir du coupon comptabilité, fiche de modification ou fiche de perception par le caissier quotidiennement pour éditer un brouillard de caisse* (l'état physique)(2 copies) et par le chef de comptoir chaque quinze jours pour éditer l'EBV (5 copies) et par le chef d'agence mensuellement pour éditer la situation mensuelle*(2 copies).







* Brouillard de caisse : les détails des ventes journalières.
* Situation mensuelle : la synthèse mensuelle de toutes les ventes.
*QR : un logiciel de saisie monoposte sous MS-DOS.

Vérification et contrôle des EBVs :

Au niveau du Département Commerciale et Logistique, des contrôleurs de tarification contrôlent et vérifient les EBVs par rapport aux coupons comptabilités reçues des différentes agences de l'entreprise, si des erreurs sont trouvées le chef du Département Commerciale et Logistique envoie une lettre aux agences concernées en précisant ces erreurs, sinon ils seront archivés.

Au niveau du Département Comptabilité, des comptables contrôlent et vérifient les EBVs par rapport aux brouillards de caisse reçues des différentes agences de l'entreprise si des erreurs sont trouvées le chef du Département Comptabilité envoi une lettre aux agences concernées en précisant ces erreurs, sinon un traitement comptable sera effectué sur eux ensuite une copie de l'EBV sera envoyé à la direction informatique, enfin ils seront archivés.

Traitement des situations mensuelles :

Au niveau du Département Commerciale et Logistique, des agents commerciaux saisissent dans le logiciel QR toutes les situations mensuelles reçues des différentes agences de l'entreprise pour éditer une situation mensuelle globale qui contient l'écart entre les commissions et les charges (masse salariale....etc.)

Traitement de l'EBV :

Au niveau du Direction Informatique, des techniciens saisissent dans le logiciel QR l'EBV reçue de la Département Comptabilité et le manifeste (l'état des billets voyagés) pour calculer l'écart entre eux, s'il égale à 0 ils seront archivés, sinon ils seront envoyés à la Direction Régional pour résoudre le problème.

1.3. Recueil des besoins opérationnels :

Sécurité :

Lors de sa connexion, chaque employé doit être reconnu du système par un nom, un mot de passe et la fonction qu'il occupe.

Un administrateur système est chargé de définir les profils des utilisateurs.

1.4. Les acteurs du système SIAF* :

Nous allons maintenant énumérer les acteurs susceptibles d'interagir avec le système, mais d'abord nous donnons une définition de ce que c'est un acteur.

Définition : un *acteur* représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié. [Rocques & Vallée, 2004]

Les acteurs du système SIAF* identifiés dans un premier temps sont :

- ***Le caissier :***
  Le caissier a pour mission de saisir les détails des billets vendus quotidiennement et établir le brouillard de caisse.

- ***Le chef de comptoir :***
  Le chef de comptoir a pour mission établir l'état des billets vendus chaque 15 jours à partir des coupons comptabilité.

- ***Le chef d'agence :***
  Le chef d'agence a pour mission établir la récapitulation mensuelle de toutes les ventes de l'agence.

- ***Le contrôleur de tarification :***
  Le contrôleur de tarification a pour mission de contrôler et vérifier les copies des EBVs par rapport aux coupons comptabilités reçues de toutes les agences de l'entreprise.

- ***L'agent commercial :***
  L'agent commercial a pour mission de saisir toutes les situations mensuelles reçues des différentes agences de l'entreprise et établir une situation mensuelle globale qui contient l'écart entre les commissions et les charges de ces agences.

- ***Le comptable :***
  Le comptable a pour mission de contrôler et vérifier les copies des EBVs par rapport aux brouillards de caisse reçues de toutes les agences de l'entreprise et établir le relevé des anomalies.

- ***Le technicien :***
  Le technicien a pour mission de saisir l'EBV reçue de la Département Comptabilité et le manifeste (l'état des billets voyagés) pour calculer l'écart entre eux.

- ***Administrateur système*** :
  L'administrateur système gère les profils des utilisateurs et les mots de passe.

* SIAF : Système d'Information de l'Algérie Ferrié .

1.5. Les messages entre SAIF et ses acteurs :

On va détailler les différents messages échangés entre le système et l'extérieur.

Définition **:** un *message* représente la spécification d'une communication unidirectionnelle entre les objets qui transporte de l'information avec l'intention de déclencher une activité chez le récepteur. [Rocques & Vallée, 2004]

Le système SIAF émet (entre autres) :

- L'état des billets vendus pour le chef de comptoir.
- Le brouillard de caisse pour le caissier.
- La situation mensuelle pour le chef d'agence.
- La situation mensuelle globale pour l'agent commercial.
- Le relevé des anomalies pour le comptable.

Le système SIAF reçoit (entre autres) :

- Les détails des billets vendus du caissier.
- Les détails des coupons comptabilités du chef de comptoir.
- Les détails des situations mensuelles de l'agent commercial.
- Les détails des billets voyagés (manifeste) par le technicien.

1.6. Diagramme de contexte dynamique de SIAF :

Tous les messages (système ↔ acteurs) identifiés précédemment peuvent être représentés de façon synthétique sur un diagramme, que l'on peut qualifier de diagramme de contexte dynamique. [Rocques & Vallée, 2004]

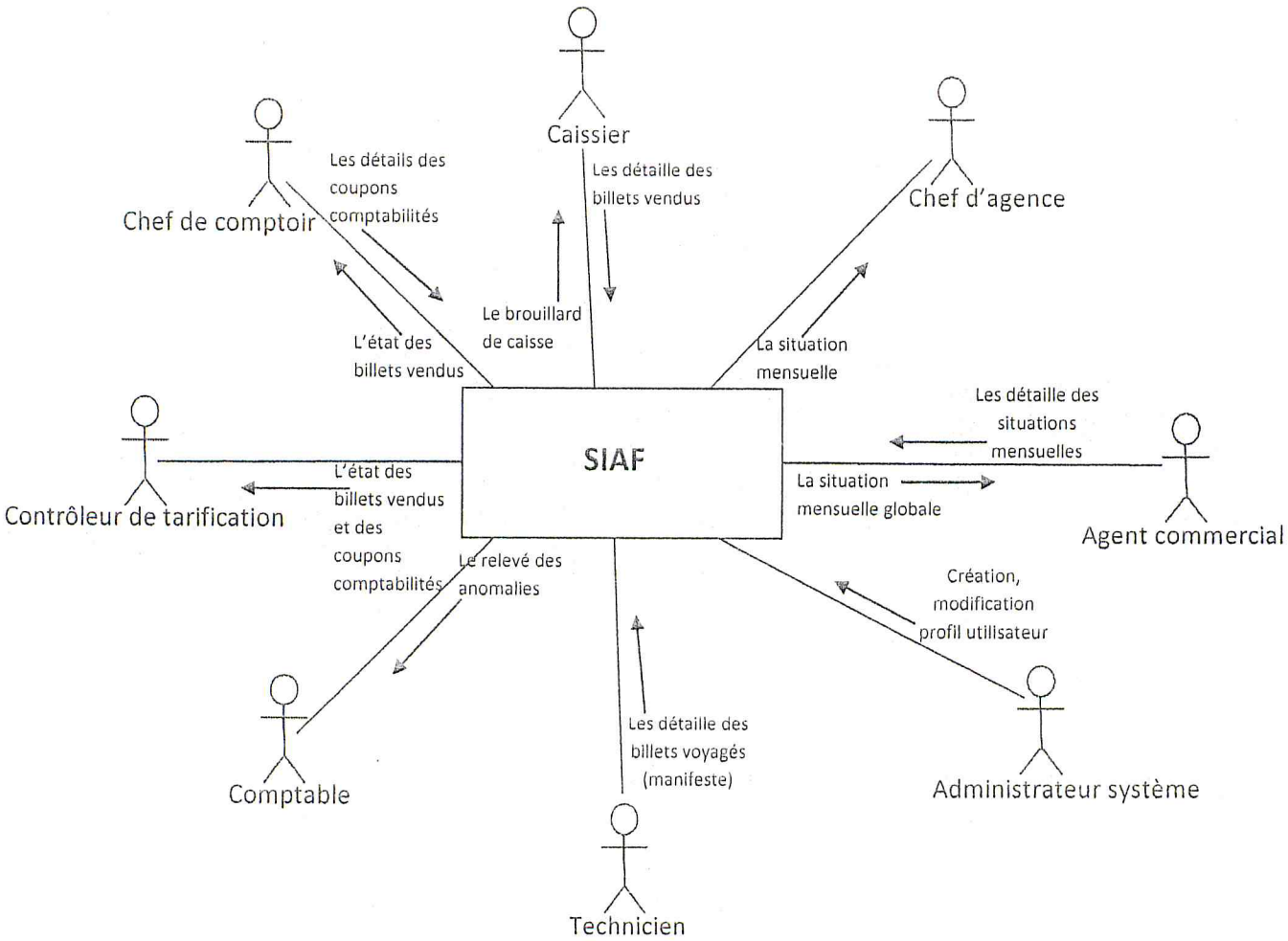


Figure 4 : Diagramme de contexte dynamique de SIAF

Pour ne pas surcharger ce premier diagramme, nous avons volontairement omis :

- Les actions de consultation pure.
- Les actions de connexion/déconnexion au système.

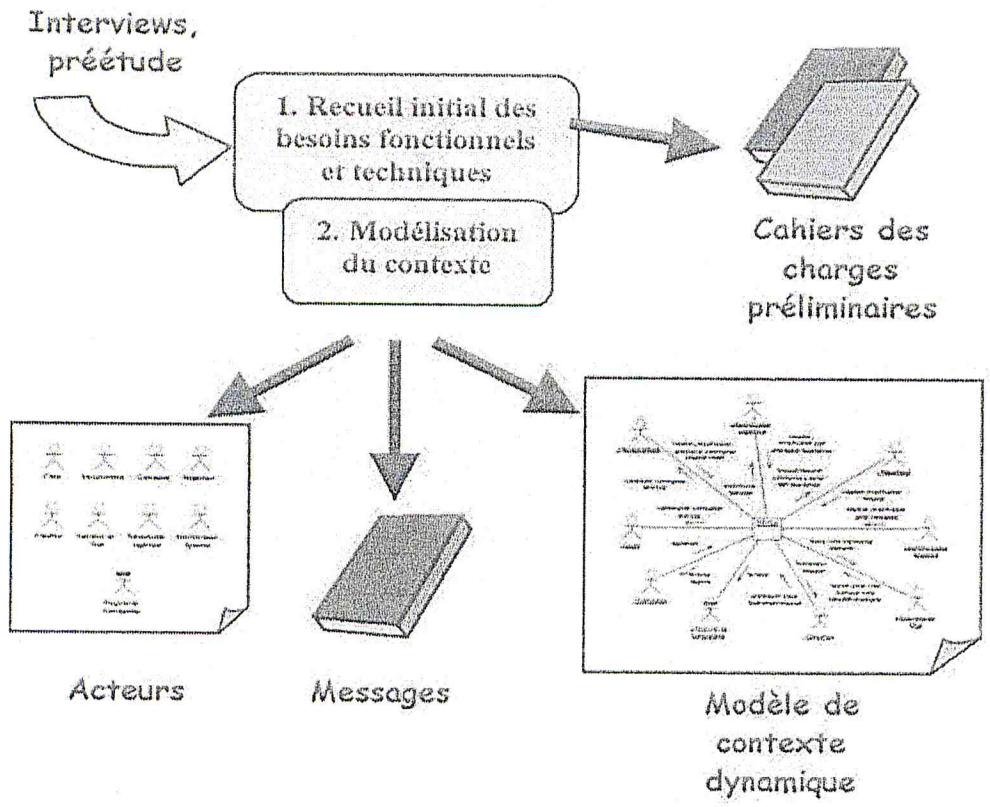


Figure 5 : Résumé des activités et des produits de l'étude préliminaire

2. CAPTURE DES BESIONS FONCTIONNELS :

2.1. Objectifs du chapitre :

Ce chapitre traite du rôle que tient UML pour compléter la capture des besoins fonctionnels ébauchée durant l'étude préliminaire. La technique des cas d'utilisation est la pierre angulaire de cette étape. Elle va nous permettre de préciser l'étude du contexte fonctionnel du système, en décrivant les différentes façons qu'auront les acteurs d'utiliser le futur système. [Rocques & Vallée, 2004]

Nous verrons successivement dans ce chapitre comment :

- Identifier les cas d'utilisation du système par ses acteurs,
- décrire les cas d'utilisation,
- organiser les cas d'utilisation,
- Identifier les classes candidates du modèle d'analyse.

2.2. Liste préliminaire des cas d'utilisation de SIAF :

Voici la technique recommandée pour identifier les cas d'utilisation à partir du modèle de contexte : considérez l'intention fonctionnelle de l'acteur par rapport au système dans le cadre de l'émission ou de la réception de chaque message, En regroupant les intentions fonctionnelles en unités cohérentes, vous obtiendrez les cas d'utilisation recherchés. Le tableau ci-dessous permet d'établir le résultat de ce travail, en montrant le lien entre les cas d'utilisation identifiés, les acteurs principaux et secondaires, et les messages provenant du contexte. [Rocques & Vallée, 2004]

Cas d'utilisation	**Acteur principal, acteurs secondaires**	**Message(s) émis/reçus par les acteurs**	
		émet	**reçoit**
Traiter les billets vendus	Caissier	Les détaille des billets vendus	Le brouillard de caisse
	Chef de comptoir	Les détails des coupons comptabilités	L'état des billets vendus
Synthétiser les ventes	Chef d'agence		La situation mensuelle
Consulter l'EBV et les coupons comptabilité	Contrôleur de tarification		L'état des billets vendus et les détails des coupons comptabilités
Gérer les situations mensuelles	Agent commercial	Les détails des situations mensuelles	La situation mensuelle globale
Réaliser le relevé des anomalies	Comptable	Le brouillard de caisse	Le relevé des anomalies
Calculer l'écart entre les billets vendus et voyagés	Technicien	Les détails des billets voyagés (manifeste)	

Tableau2 : Liste des acteurs et des messages par cas d'utilisation

2.3. Description préliminaire de cas d'utilisation :

Voici une description préliminaire des cas d'utilisations énumérés précédemment :

Traiter les billets vendus :

- **Intention :** traiter les billets vendus au sein de l'agence pour élaborer l'EBV, le brouillard de caisse et la situation mensuelle.
- **Action :** saisir les détails des billets vendus à partir des coupons comptabilités.

Synthétiser les ventes :

- **Intention :** avoir une récapitulation mensuelle des ventes de l'agence.
- **Action :** à partir des détails saisis des billets vendus élaborer une situation mensuelle.

Consulter l'EBV et les coupons comptabilité :

- **Intention :** contrôler et vérifier les copies des EBVs par rapport aux coupons comptabilités.
- **Action :** rechercher l'EBV par quinzaine et les coupons de comptabilités par le N° de billet.

Gérer les situations mensuelles :

- **Intention :** avoir l'écart entre les commissions et les charges de ces agences.
- **Action :** saisir toutes les situations mensuelles reçues des différentes agences de l'entreprise et établir une situation mensuelle globale.

Réaliser le relevé des anomalies :

- **Intention :** s'assurer qu'il n'y a aucun problème lors de l'élaboration de l'EBV.
- **Action :** contrôler et vérifier les copies des EBVs par rapport aux brouillards de caisse.

Calculer l'écart entre les billets vendus et voyagés :

- **Intention :** s'assurer que tous les billets vendus sont voyagés.
- **Action :** saisir l'EBV reçue de la Département Comptabilité et le manifeste (l'état des billets voyagés) pour calculer l'écart entre eux.

2.4. Diagramme de cas d'utilisation de SAIF

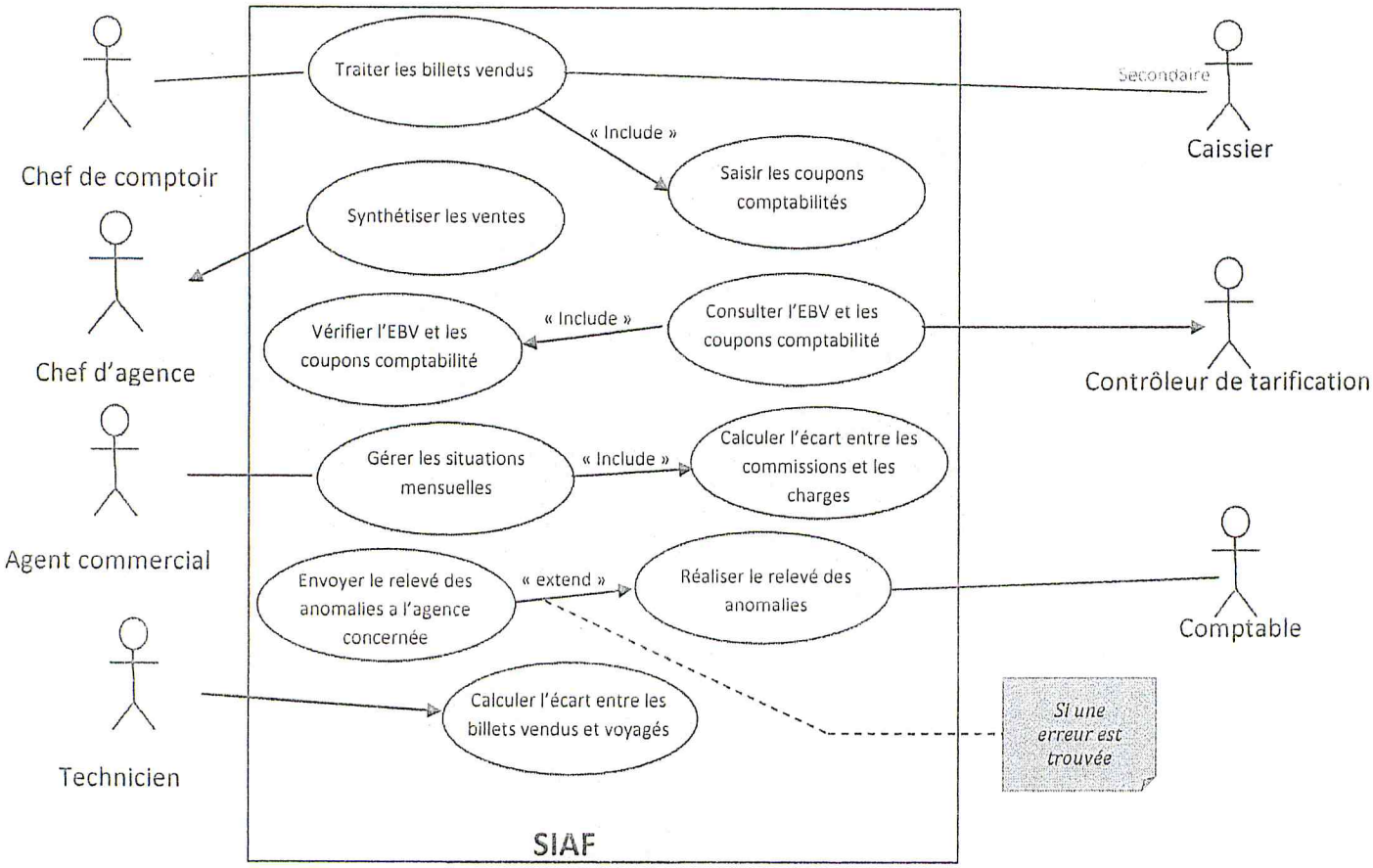


Figure 6 : diagramme de cas d'utilisation de SAIF

2.5. Description détaillée des cas d'utilisation

Nous allons maintenant détailler chaque cas d'utilisation qui doit faire l'objet d'une définition a priori qui décrit l'intention de l'acteur lorsqu'il utilise le système et les séquences d'actions principales qu'il est susceptible d'effectuer. Ces définitions servent à fixer les idées et n'ont pas pour but de spécifier un fonctionnement complet et irréversible. [Rocques & Vallée, 2004]

Les descriptions vont être organisées de la façon suivante :

- **Un sommaire d'identification :** va résumer les propriétés du cas d'utilisation.
- **Une description détaillée :** des Préconditions au déclenchement du cas d'utilisation doivent être spécifiées, un scénario nominal décrivant celui-ci additionné à des scénarios alternatifs et d'exceptions.
- **Les diagrammes (*optionnels*):** plusieurs diagrammes vont apparaitre (mais pas nécessairement) pour apporter une compréhension additive au cas d'utilisation.

2.5.1. Cas d'utilisation «Traiter les billets vendus» :

Sommaire d'identification :

Titre : Traiter les billets vendus.

But : traiter les billets vendus au sein de l'agence pour élaborer l'EBV, le brouillard de caisse et la situation mensuelle.

Résumé : saisir les détails des billets vendus à partir des coupons comptabilités, fiches de modification ou fiches de perception.

Acteur : Chef de comptoir (principal), Caissier (secondaire).

Description des enchainements :

Pré conditions :

- Le chef de comptoir est authentifié.
- Il existe au moins un billet vendu.

Scénario nominal :

Ce cas d'utilisation commence lorsque le chef de comptoir demande au système de créer un nouvel état des billets vendus ou un nouveau brouillard.

Enchaînement (a) Créer un brouillard de caisse

Le caissier fournit un nom d'identification

Enchaînement (b) Créer l'EBV

Le chef de comptoir choisit fixe la quinzaine dont l'EBV est réaliser.

Enchaînement (c) Affecter les billets vendus

Le chef de comptoir affecte les billets vendus à un EBV par quinzaine.

Le caissier affecte les billets vendus au brouillard de caisse quotidiennement.

Enchaînement (d) Valider l'EBV

Le chef de comptoir valide l'EBV, donc il doit préciser la date de début de fin de la quinzaine.

Enchaînement (e) Valider le brouillard de caisse

Le caissier valide le brouillard de caisse, il doit alors préciser la date de la validation.

Enchainements alternatifs :

Enchaînement (f) Modifier l'EBV ou le brouillard de caisse

Le chef de comptoir désaffecte ou modifie les billets vendus.

Enchaînement (g) Annuler l'EBV

Le chef de comptoir annule un EBV non encore validé ou un EBV validé avant qu'il envoie au poste suivant.

Enchaînement (h) Annuler le brouillard de caisse

Le caissier annule un brouillard de caisse non encore validé ou un EBV validé avant qu'il envoie au poste suivant.

2.5. 2. D'utilisation «Synthétiser les ventes» :

Sommaire d'identification :

Titre : Synthétiser les ventes.

But : avoir une récapitulation mensuelle des ventes de l'agence.

Résumé : à partir des détails saisis des billets vendus élaborer une situation mensuelle.

Acteur : Chef d'agence.

Description des enchainements :

Pré conditions :

- Le chef d'agence est authentifié.
- Il existe au moins un billet vendu.

Scénario nominal :

Ce cas d'utilisation commence lorsque le chef d'agence demande au système de créer une nouvelle situation mensuelle.

Enchaînement (a) Créer une situation mensuelle

Le chef d'agence un nom d'identification

Enchaînement (b) Affecter les billets vendus

Le chef d'agence affecte les billets vendus à une situation mensuelle.

Enchaînement (c) Valider la situation mensuelle

Le chef d'agence valide la situation mensuelle, donc il doit préciser la date.

Enchainements alternatifs :

Enchaînement (d) Modifier la situation mensuelle

Le chef d'agence désaffecte ou modifie les billets vendus.

Enchaînement (e) Annuler la situation mensuelle

Le chef d'agence annule une situation mensuelle non encore validé ou une situation mensuelle validé avant qu'il envoie au poste suivant.

2.5. 3. Cas d'utilisation «Consulter l'EBV et les coupons comptabilité» :

Sommaire d'identification :

Titre : Consulter l'EBV et les coupons comptabilité.

But : contrôler et vérifier les copies des EBVs par rapport aux coupons comptabilités.

Résumé : rechercher l'EBV par quinzaine et les coupons de comptabilités reçues de toutes les agences de l'entreprise par le N° de billet.

Acteur : Contrôleur de tarification.

Description des enchainements :

Pré conditions :

- Le contrôleur de tarification est authentifié.
- La réception de l'EBV.
- La réception des coupons comptabilité.

Scénario nominal :

Ce cas d'utilisation commence lorsque le contrôleur de tarification demande au système de consulter l'EBV et les coupons comptabilité.

Enchaînement (a) Consulter l'EBV

Le contrôleur de tarification cherche l'EBV par quinzaine.

Enchaînement (b) Consulter les coupons comptabilité

Le contrôleur de tarification cherche l'EBV par quinzaine.

Enchaînement (c) Vérifier les coupons comptabilité

Le contrôleur de tarification vérifie les coupons de comptabilités reçues des différentes agences de l'entreprise par rapport à la planche tarifaire.

Enchaînement (d) Vérifier l'EBV par rapport les coupons comptabilité

Le contrôleur de tarification vérifie l'EBV par rapport aux coupons de comptabilité reçues des différentes agence de l'entreprise.

[Exception1 : UneAnomalieTrouver]

Description des enchainements :

[Exception1 : UneAnomalieTrouver] : un message d'erreur s'affiche sur l'écran du contrôleur de tarification indiquant les anomalies trouvées.

2.5. 4. Cas d'utilisation «Gérer les situations mensuelles» :

Sommaire d'identification :

Titre : Gérer les situations mensuelles.

But : Avoir l'écart entre les commissions et les charges de ces agences.

Résumé : saisir toutes les situations mensuelles reçues des différentes agences de l'entreprise et établir une situation mensuelle globale.

Acteur : Agent commercial.

Description des enchainements :

Pré conditions :

- L'agent commercial est authentifié.
- Il a reçue au moins une situation mensuelle.

Scénario nominal :

Ce cas d'utilisation commence lorsque l'agent commercial demande au système de créer une nouvelle situation mensuelle globale.

Enchaînement (a) Créer une situation mensuelle globale

L'agent commercial fournit un nom d'identification

Enchaînement (b) Affecter les situations mensuelles

L'agent commercial affecte les situation mensuelles reçues de toutes les agences de l'entreprise a une situation mensuelle globale.

Enchaînement (c) Valider la situation mensuelle globale

L'agent commercial valide la situation mensuelle globale, donc il doit préciser la date de sa validation.

Enchainements alternatifs :

Enchaînement (f) Modifier la situation mensuelle globale

L'agent commercial désaffecte ou modifie les situations mensuelles reçues de toutes les agences de l'entreprise.

Enchaînement (g) Annuler la situation mensuelle globale

L'agent commercial annule une situation mensuelle globale non encore validé ou une situation mensuelle globale validé avant qu'il envoie au poste suivant.

2.5. 5. Cas d'utilisation «Réaliser le relevé des anomalies» :

Sommaire d'identification :

Titre : Réaliser le relevé des anomalies.

But : s'assurer qu'il n'y a aucun problème lors de l'élaboration de l'EBV.

Résumé : contrôler et vérifier les copies des EBVs par rapport aux brouillards de caisse.

Acteur : Comptable.

Description des enchainements :

Pré conditions :

- Le comptable est authentifié.
- Il a reçue au moins un EBV et ses brouillards de caisse.

Scénario nominal :

Ce cas d'utilisation commence lorsque le chef de comptoir demande au système de créer un nouvel état des billets vendus ou un nouveau brouillard.

Enchaînement (a) Créer un relevé d'anomalie

Le comptable fournit un nom d'identification

Enchaînement (b) Vérifie l'EBV par rapport aux brouillards de caisse

Le comptable vérifie l'EBV par rapport aux brouillards de caisses reçues de toutes les agences de l'entreprise.

Enchaînement (c) Valider le relevé d'anomalie

Le comptable valide le relevé d'anomalie, donc il doit préciser les erreurs trouvées lors de la vérification.

Enchainements alternatifs :

Enchaînement (d) Modifier le relevé des anomalies

Le comptable désaffecte l'EBV ou les brouillards de caisse ou bien modifie l'EBV ou les brouillards.

Enchaînement (g) Annuler le relevé des anomalies

Le comptable annule un relevé des anomalies non encore validé ou un relevé des anomalies validé avant qu'il envoie au poste suivant.

2.5.6. Cas d'utilisation «Calculer l'écart entre les billets vendus et voyagés» :

Sommaire d'identification :

Titre : Calculer l'écart entre les billets vendus et voyagés.

But : s'assurer que tous les billets vendus sont voyagés.

Résumé : saisir l'EBV reçu de la Département Comptabilité et le manifeste (l'état des billets voyagés) pour calculer l'écart entre eux.

Acteur : Technicien.

Description des enchainements :

Pré conditions :

- Le technicien est authentifié.
- Il a reçu au moins un EBV et un manifeste.

Scénario nominal :

Ce cas d'utilisation commence lorsque le technicien demande au système de consulter l'EBV et le manifeste.

Enchaînement (a) Consulter l'EBV

Le technicien recherche l'EBV par quinzaine.

Enchaînement (b) Consulter le manifeste

Le technicien recherche le manifeste par numéro de voyage.

Enchaînement (c) Calculer l'écart

Le technicien calcule l'écart entre les billets vendus et voyagés .

[Exception1 : EcartDeferentDe0]

Description des enchainements :

[Exception1 : EcartDeferentDe0] : un message d'erreur s'affiche sur l'écran du technicien indiquant la somme de l'écart.

2.6. Structuration des cas d'utilisation dans des packages :

Cette phase va permettre de structurer les cas d'utilisations en groupes fortement cohérents, ceci afin de préparer le terrain pour la prochaine phase qui est le « découpage en catégories ».

Définition : un *package* représente un espace de nommage qui peut contenir :

1. Des éléments d'un modèle.
2. Des diagrammes qui représentent les éléments du modèle.
3. D'autres packages.

La structuration des cas d'utilisations se fait par domaine d'expertise métier c.à.d. les éléments contenus dans un *package* doivent représenter un ensemble fortement cohérent et sont généralement de même nature et de même niveau sémantique [Rocques & Vallée, 2004].

Cas d'utilisation	Acteurs	Package
Traiter les billets vendus	Caissier	Gestion des ventes
	Chef de comptoir	
Synthétiser les ventes	Chef d'agence	
Consulter l'EBV et les coupons comptabilité	Contrôleur de tarification	Gestion commercial
Gérer les situations mensuelles	Agent commercial	
Réaliser le relevé des anomalies	Comptable	Comptabilité
Calculer l'écart entre les billets vendus et voyagés	Technicien	Gestion des billets
Gérer les profils	Administrateur	Services support

Tableau 3 : Liste des cas d'utilisation et de leurs acteurs par package

2.7. Identification des classes candidates :

Cette phase va préparer la *modélisation orientée objet* en aidant à trouver les classes principales du futur modèle statique d'analyse.

La technique utilisée pour identifier les classes candidates est la suivante :

- Chercher les noms communs importants dans les descriptions textuelles des cas d'utilisation
- Vérifier les propriétés « objet » de chaque concept (identité, propriétés comportement), puis définir ses responsabilités.

Définition :

Une *responsabilité* est une sorte de contrat, ou d'obligation, pour une classe. Elle se place à un niveau d'abstraction plus élevé que les attributs ou les opérations. On formalise ensuite ces concepts métier sous forme de classes et d'associations rassemblées dans un diagramme statique pour chaque cas d'utilisation. Ces diagrammes préliminaires sont appelés *diagramme des classes participantes* », n'ont pas pour objectif d'être complet. Ils servent uniquement à démarrer la découverte des classes du modèle d'analyse [Rocques & Vallée, 2004].

a) Diagramme des classes participantes du cas d'utilisation «Traiter les billets vendus» :

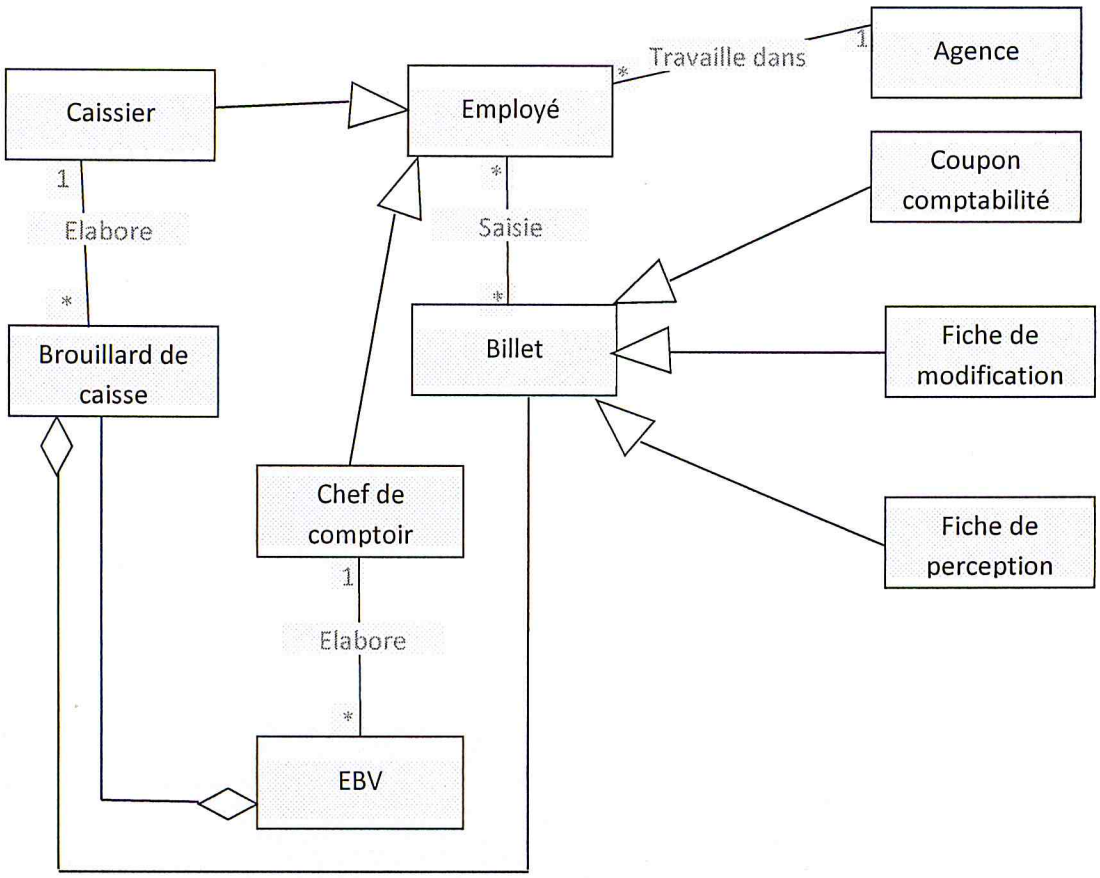


Figure 7 : Diagramme des classes participantes du cas d'utilisation «Traiter les billets vendus»

b) Diagramme des classes participantes du cas d'utilisation «Synthétiser les ventes» :

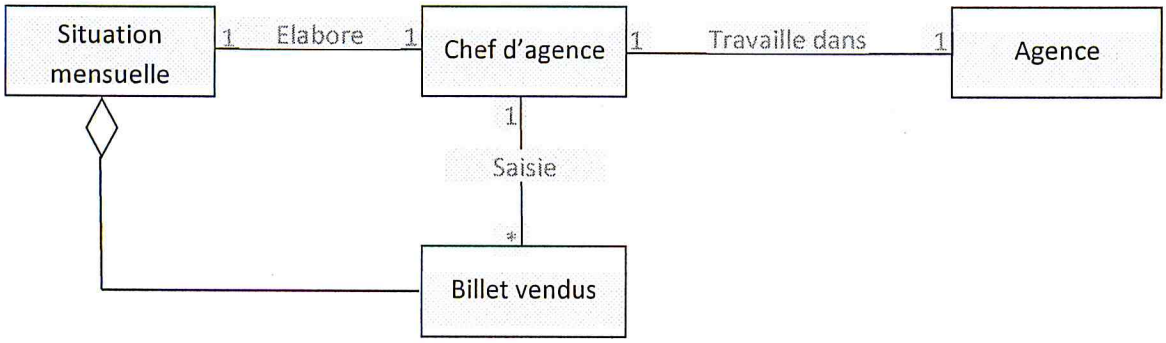


Figure 8 : Diagramme des classes participantes du cas d'utilisation «Synthétiser les ventes»

c) Diagramme des classes participantes du cas d'utilisation «Consulter l'EBV et les coupons comptabilité» :

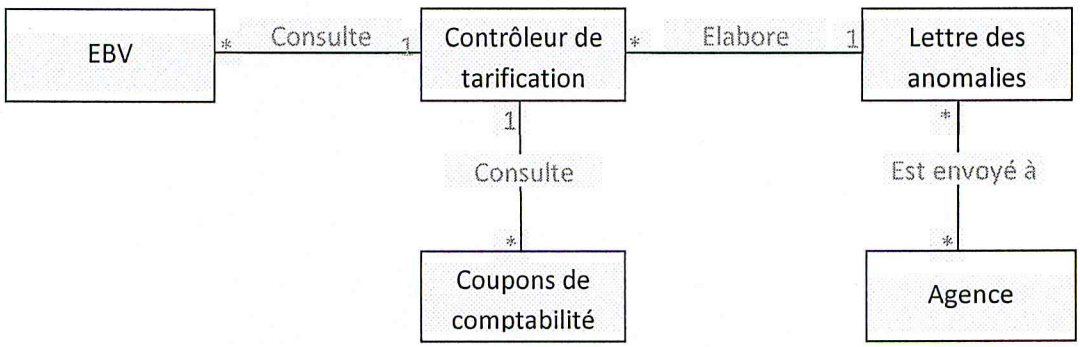


Figure 9 : Diagramme des classes participantes du cas d'utilisation «Consulter l'EBV et les coupons comptabilité»

d) Diagramme des classes participantes du cas d'utilisation «Gérer les situations mensuelles» :

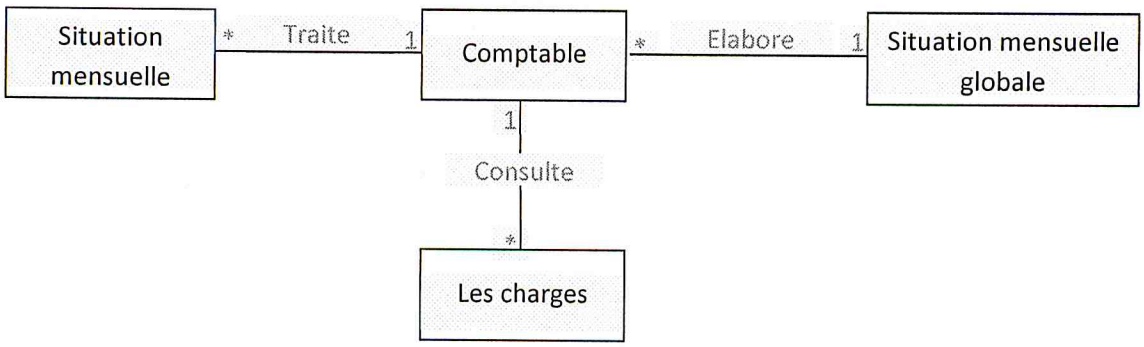


Figure 10 : Diagramme des classes participantes du cas d'utilisation «Gérer les situations mensuelles»

e) Diagramme des classes participantes du cas d'utilisation «Réaliser le relevé des anomalies» :

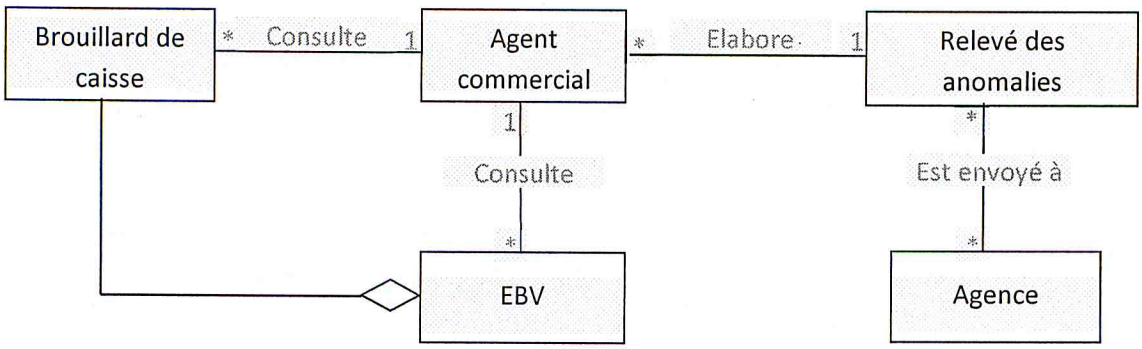


Figure 11 : Diagramme des classes participantes du cas d'utilisation «Réaliser le relevé des anomalies»

f) **Diagramme des classes participantes du cas d'utilisation «Calculer l'écart entre les billets vendus et voyagés» :**

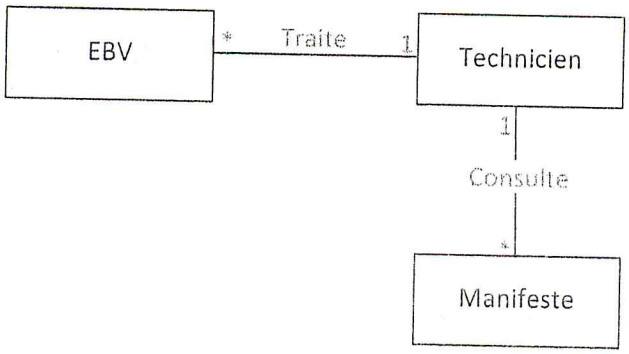


Figure 12 : Diagramme des classes participantes du cas d'utilisation «Calculer l'écart entre les billets vendus et voyagés»

3. L'ANALYSE :

3.1. Découpage en catégories :

Cette phase marque le démarrage de **l'analyse objet** du système à réaliser.

Elle utilise la notion de *package* pour définir des catégories de classes d'analyse et découper le modèle UML en blocs logiques les plus indépendants possibles.

Le découpage en catégories constitue la première activité de l'étape d'analyse et elle va s'affiner de manière itérative au cours du développement du projet. Elle se situe sur la branche gauche du cycle en Y et succède à la capture des besoins fonctionnels. [Rocques & Vallée, 2004]

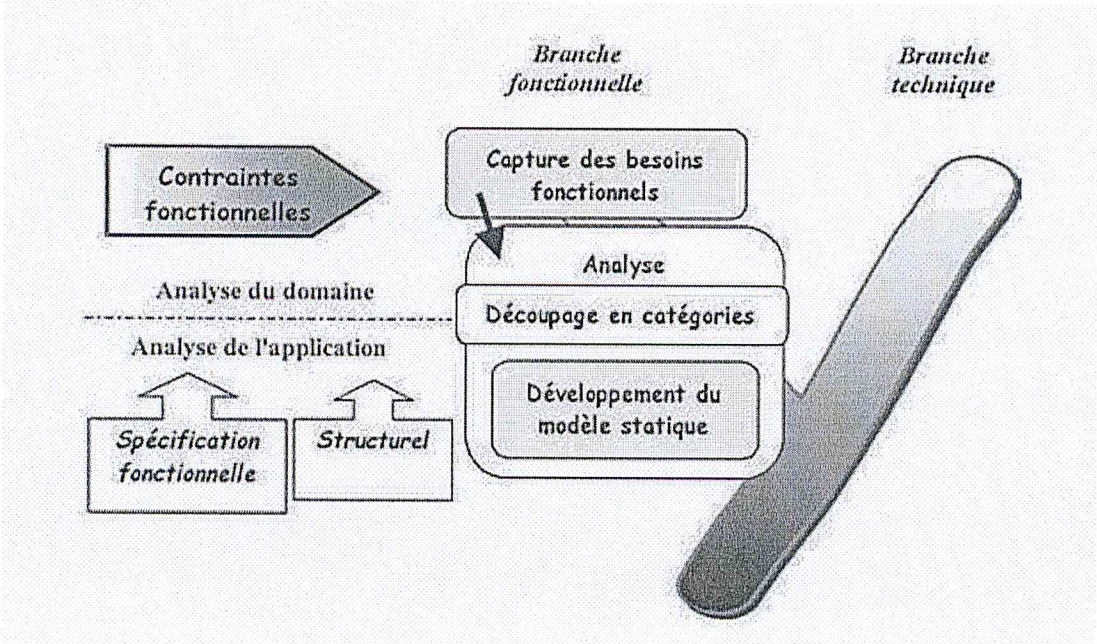


Figure 13 : Situation du découpage en catégories dans 2TUP

3.1.1. Notion de catégorie :

La classe représente une entité de structuration trop petite dès lors qu'on s'attaque à un projet réel. Au-delà d'une douzaine de classes, il est utile de regrouper les classes fortement couplées en unités plus grandes. Le couplage s'exprime à la fois structurellement par des associations, des agrégations, des compositions ou des généralisations, mais aussi dynamiquement par les interactions qui se produisent entre les instances des classes. Bien sûr, plus le nombre de classes devient important, et plus cette structuration s'avère indispensable. G. Booch [Booch 96] a introduit le concept de *catégorie* pour nommer ce regroupement de classes qui constitue la brique de construction du modèle structurel d'analyse. [Rocques & Vallée, 2004]

Le terme *catégorie* n'appartient pas en standard au langage UML qui comporte en revanche le concept plus général de package. Pour notre part, nous avons conservé ce terme, afin de différencier les catégories qui structurent un modèle construit sur des classes, du concept plus générique de package. Nous représentons graphiquement les *catégories* comme des stéréotypes de packages. [Rocques & Vallée, 2004]

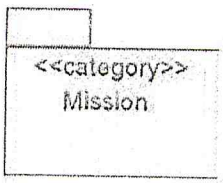


Figure 14 : Représentation graphique d'une catégorie

3.1.2. Premier découpage en catégories de SIAF :

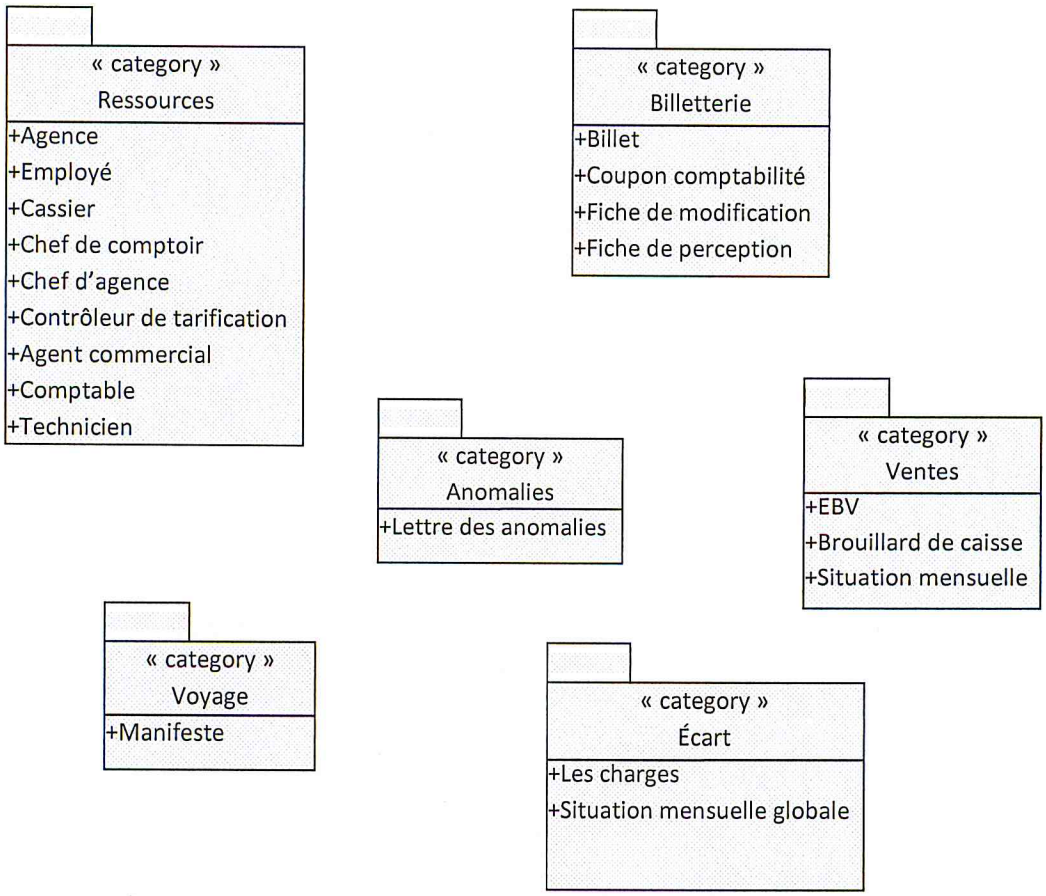


Figure 15 : Premier découpage en catégories de SIAF

3.1.2.1 Diagramme de classes préliminaire de la catégorie « Ressources » avec les navigabilités :

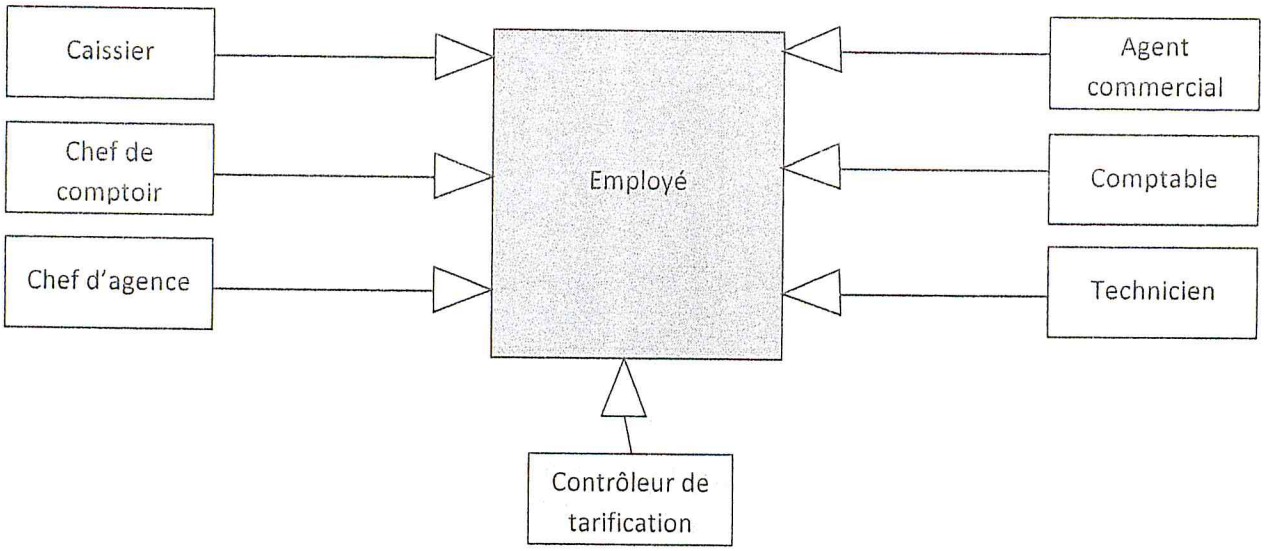


Figure 16 : Diagramme de classes préliminaire de la catégorie « Ressources » avec les navigabilités

3.1.2.2 Diagramme de classes préliminaire de la catégorie « Billetterie » avec les navigabilités :

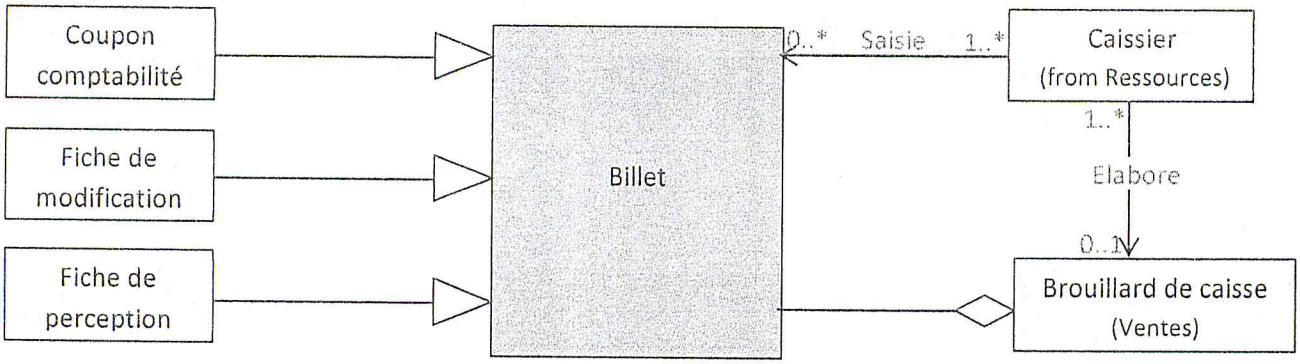


Figure 17 : Diagramme de classes préliminaire de la catégorie « Billetterie » avec les navigabilités

3.1.2.3. Diagramme de classes préliminaire de la catégorie « Ventes » avec les navigabilités :

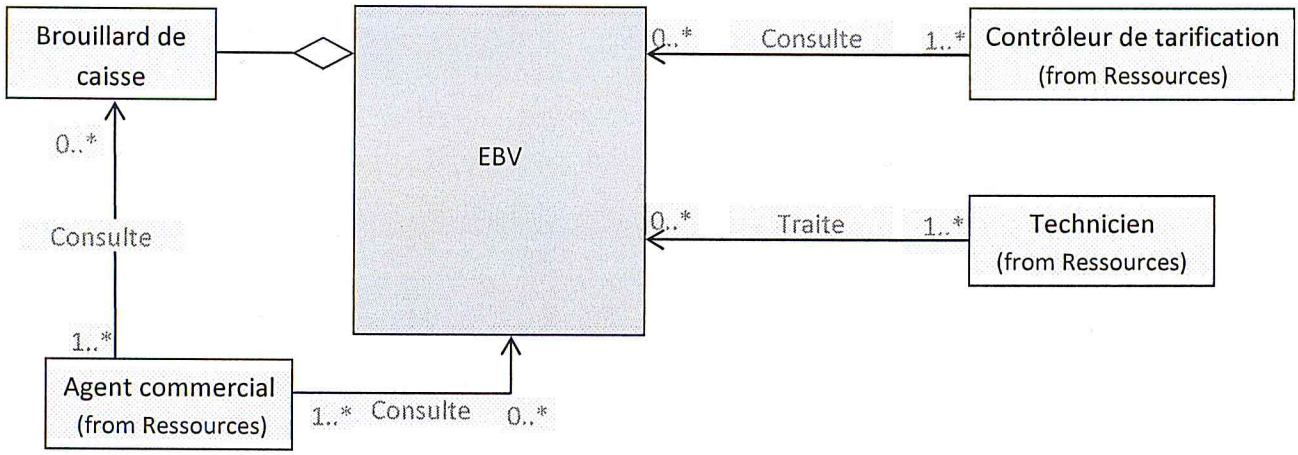


Figure 18: Diagramme de classes préliminaire de la catégorie « Ventes » avec les navigabilités

3.1.2.4. Diagramme de classes préliminaire de la catégorie « Écart » avec les navigabilités :

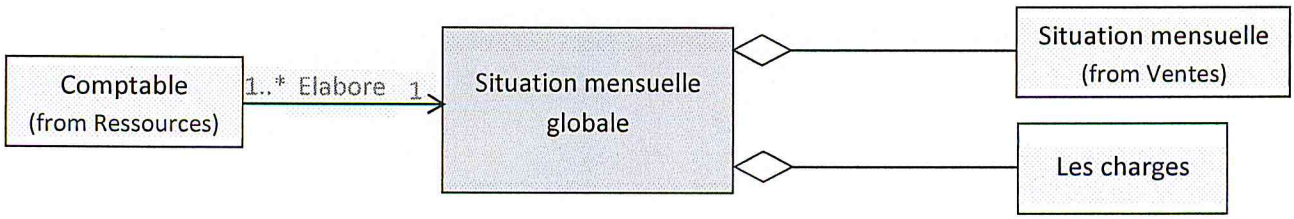


Figure 19: Diagramme de classes préliminaire de la catégorie « Écart » avec les navigabilités

3.1.2.5. Diagramme de classes préliminaire de la catégorie « Anomalies » avec les navigabilités :

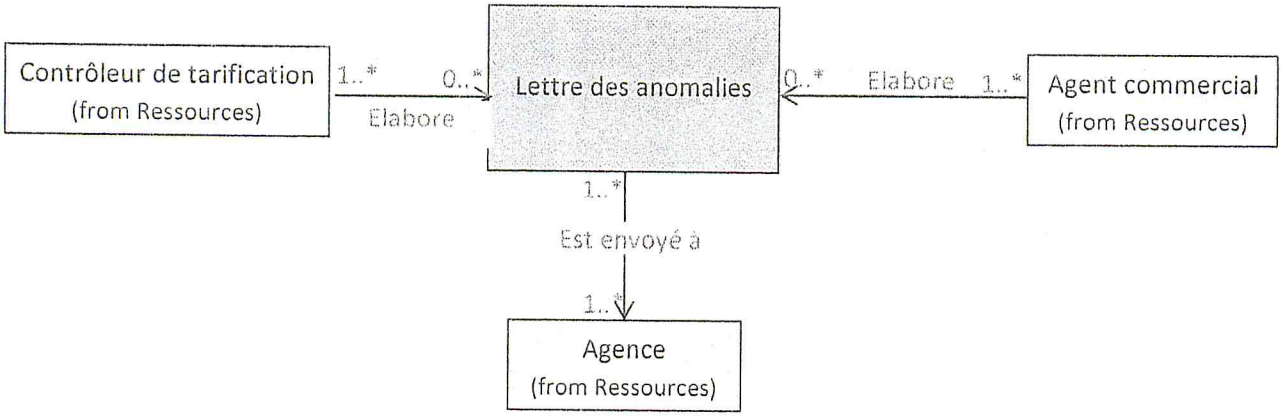


Figure 20 : Diagramme de classes préliminaire de la catégorie « Anomalies » avec les navigabilités

3.1.2.6. Diagramme de classes préliminaire de la catégorie « Voyage » avec les navigabilités :

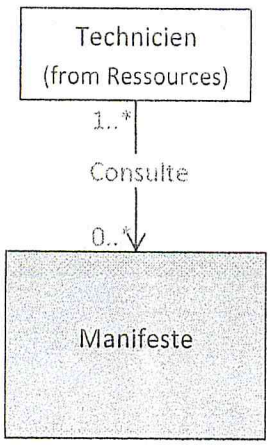


Figure 21 : Diagramme de classes préliminaire de la catégorie « Voyage » avec les navigabilités

3.1.2.7 Diagramme de packages d'analyse :

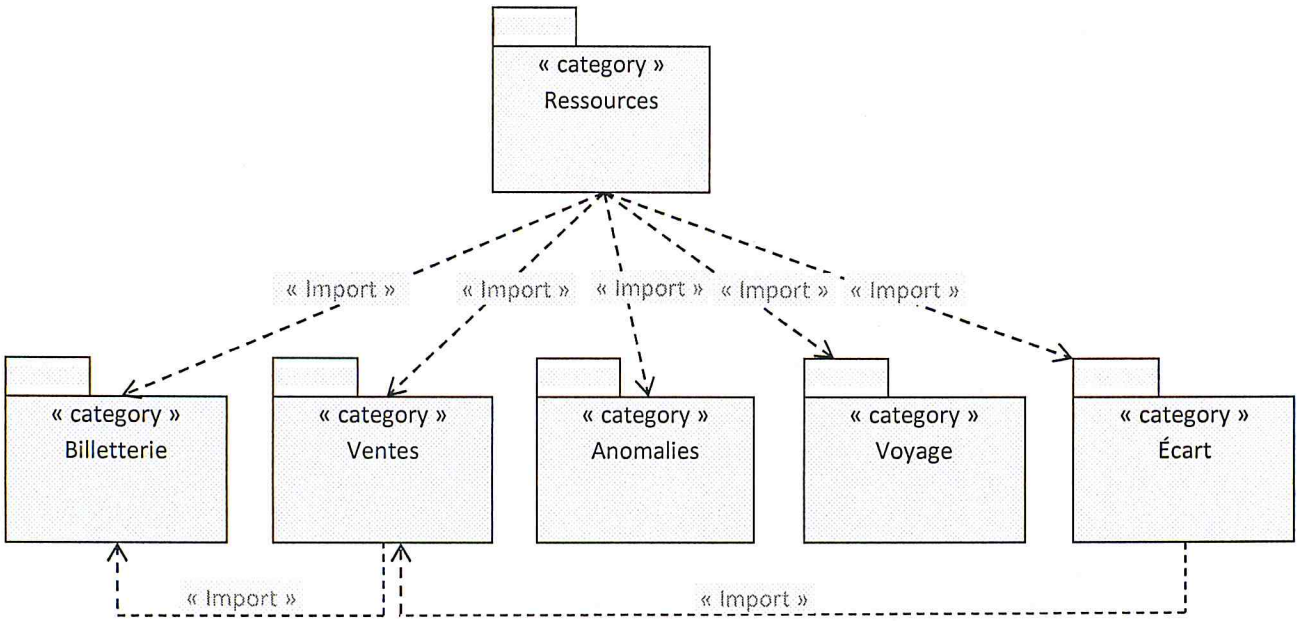


Figure 22 : Diagramme de packages d'analyse

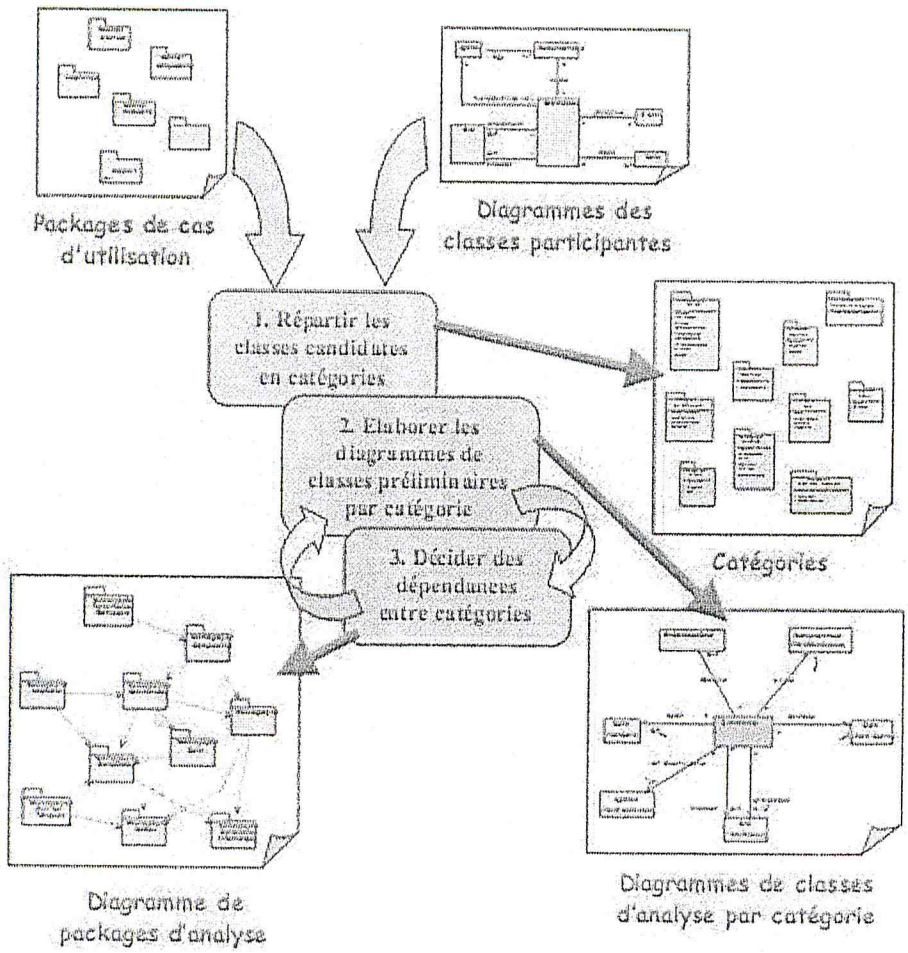


Figure 23 : Les démarches d'élaboration du modèle structurel

3.2. Développement du modèle statique :

Cette phase va nous permettre d'illustrer les principales constructions du diagramme de classes UML durant l'étape d'analyse, elle constitue la deuxième activité de l'étape d'analyse. Elle se situe sur la branche gauche du cycle en Y et succède au découpage en catégories. Les diagrammes de classes établis sommairement dans les diagrammes des classes participantes, puis réorganisés lors du découpage en catégories, vont être détaillés, complétés, et optimisés. [Rocques & Vallée, 2004]

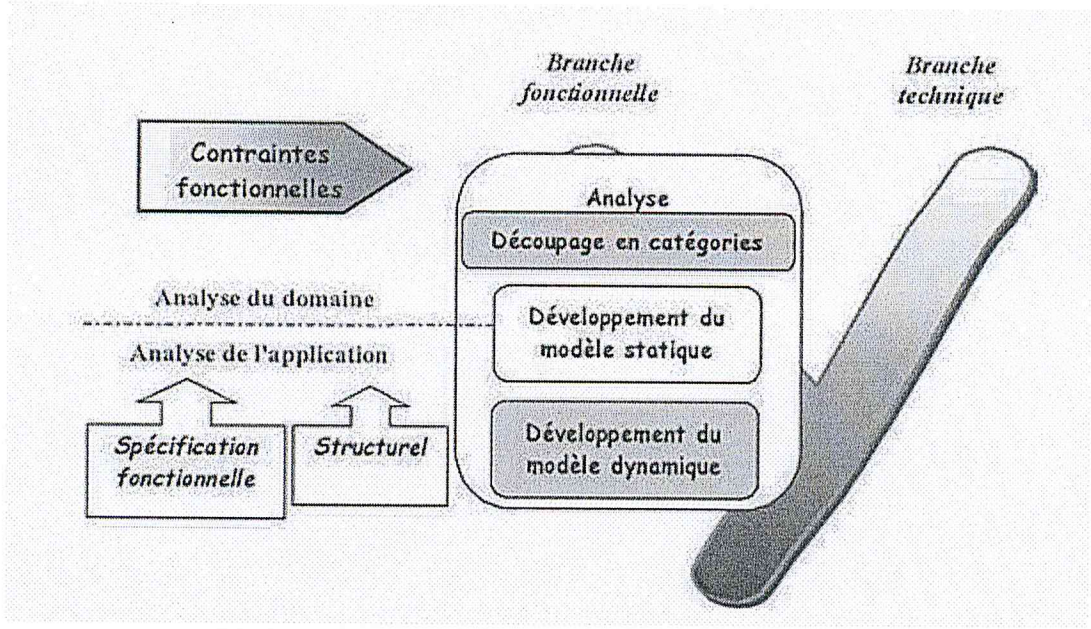
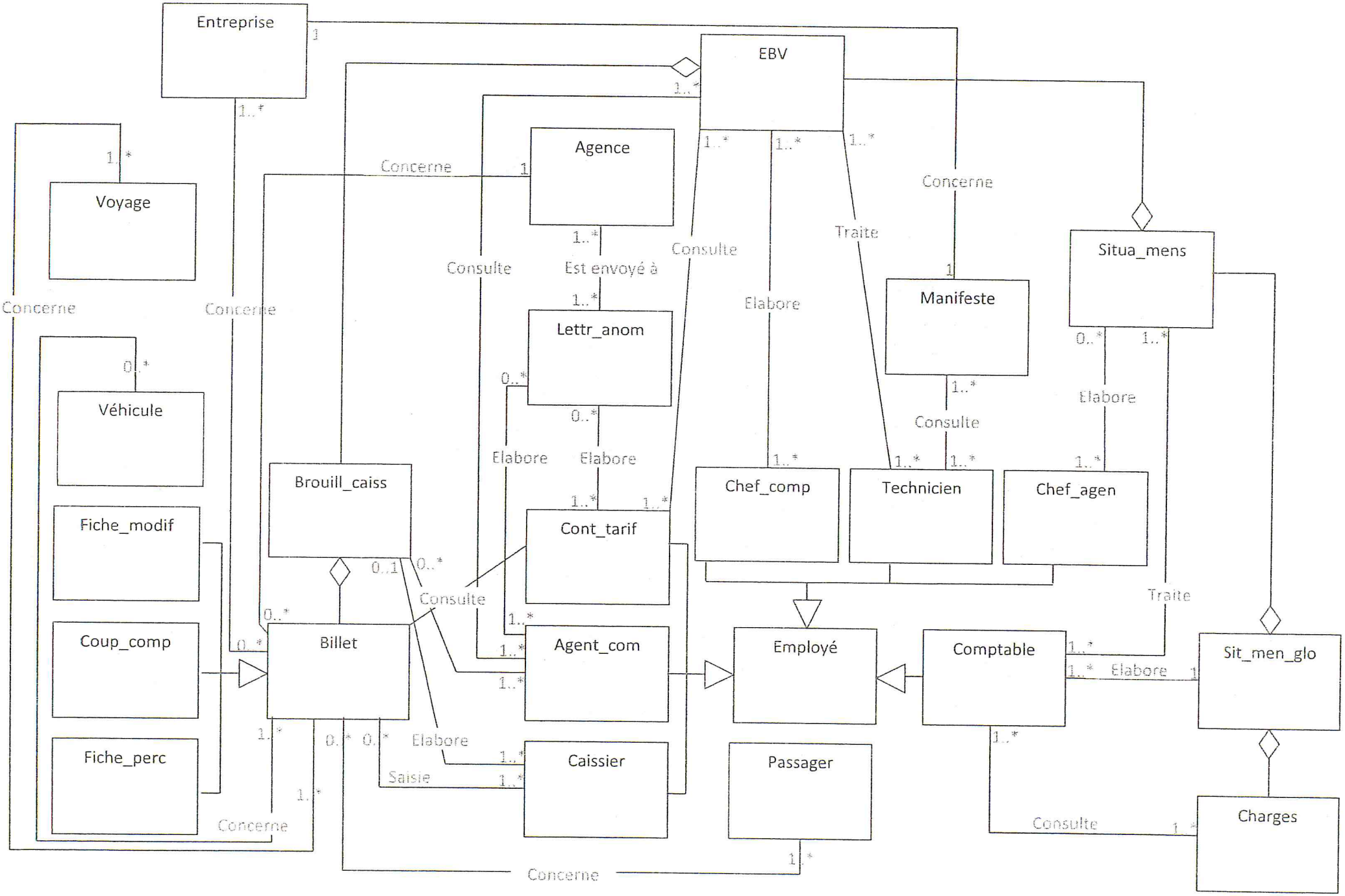


Figure 24 : Situation du développement du modèle statique dans le 2TUP

3.2.1. Diagramme de classe général :


Entreprise
EBV
Agence
Voyage
Concerne
Consulte
Est envoyé à
Traite
Elabore
Manifeste
Situa_mens
Lettr_anom
Véhicule
Brouill_caiss
Chef_comp
Technicien
Chef_agen
Fiche_modif
Cont_tarif
Coup_comp
Billet
Agent_com
Employé
Comptable
Sit_men_glo
Fiche_perc
Caissier
Passager
Saisie
Charges

3.2.2. Raffinage des classes objet :

Classe	Désignations de classe	Attribut
EBV	Etat billets vendus	- Code EBV - Période d'émission
Brouill_caiss	Brouillard de caisse	- Code brouillard de caisse - Date - Dépenses
Situa_mens	Situation mensuelle	- Code situation mensuelle - Date
Sit_men_glo	Situation mensuelle global	- Code - Date
Billet	Billet	- Numéro billet - Numéro réservation - Date émission - Info additionnel - Nom passager - Date de naissance - Véhicule - Ancien billet - Trajet - Date trajet - Heure trajet - Navire - Réduction - Cat - Installation - Prix HT - Supplément - Frai port - Fiscale - D.T - AJO - D'HONOR
Agence	Agence	- Code agence - Nom agence - Région - Adresse - Numéro de téléphone
Employé	Employé	- Code employé - Nom - Prénom - Salaire
Lettr_anom	Lettre d'anomalie	- Code - Période - Quinzaine

Charges	Charges	- Code - Mois - Agence - Charge personnel - Charge patronat
Manifeste	Manifeste	- Code - montant
Voyage	Voyage	- Code de voyage - Trajet - Date de trajet - Heure de trajet - Navire
Véhicule	Véhicule	- Code véhicule - Type véhicule
Passager	Passager	- Code de passager - Nom - Prénom - Date de naissance

Tableau 4 : Raffinage des classes objet

3.3. Développement du modèle dynamique :

3.3.1. Objectifs de chapitre :

Ce chapitre va nous permettre d'illustrer l'utilisation des concepts dynamiques d'UML et des diagrammes associés en phase d'analyse.

Nous verrons tout d'abord comment décrire des scénarios mettant en jeu un ensemble d'objets échangeant des messages. Ces interactions peuvent être décrites au moyen de deux types de diagrammes : le diagramme de séquence, qui met l'accent sur la chronologie des messages et le diagramme de communication (appelé collaboration en UML 1.x), qui souligne les relations structurelles des objets en interaction.

Nous détaillerons ensuite la façon de décrire le cycle de vie d'un objet d'une classe particulière, au fil de ses interactions et de son évolution propre. Le diagramme d'états permet en effet une description précise et exhaustive des états d'un objet et des transitions causées par l'arrivée d'événements, y compris les réponses de l'objet. Nous verrons donc comment utiliser efficacement les nombreux concepts des diagrammes d'états [Rocques & Vallée, 2004].

3.3.2. Identifier les scénarios:

Un cas d'utilisation décrit un ensemble de scénarios. Un scénario décrit une exécution particulière d'un cas d'utilisation du début à la fin.il correspond à une sélection d'enchainements du cas d'utilisation.

On peut distinguer plusieurs types de scénarios :

- **Nominaux** : ils réalisent les *post conditions* du cas d'utilisation, d'une façon naturelle et fréquente ;
- **Alternatifs** : ils remplissent les *post conditions* du cas d'utilisation, mais en empruntant des voies détournées ou rares ;
- **Aux limites** : ils réalisent les *post conditions* du cas d'utilisation, mais modifient le système de telle sorte que la prochaine exécution du cas d'utilisation provoquera une erreur ;
- **D'erreur** : ne réalisent pas les *post conditions* du cas d'utilisation. Il faut signaler que tous les scénarios possibles ne peuvent être énumérés et décrits du fait qu'ils en existent beaucoup. C'est pour cette raison que nous allons faire une description des scénarios les plus pertinents [Rocques & Vallée, 2004] .

3.3.2.1 Le cas d'utilisation : « *Traiter les billets vendus* » : TBV*

- **Les scénarios nominaux :**
  - TBV_N1 : créer un nouveau brouillard de caisse
  - TBV_N2 : créer un nouveau EBV

- **Les scénarios alternatifs :**
  - TBV_A1 : modifie un brouillard de caisse par ajout d'un billet
  - TBV_A2 : modifie un brouillard de caisse par la modification du contenu d'un billet
  - TBV_A3 : modifie un brouillard de caisse par suppression d'un billet
  - TBV_A4 : modifie un EBV par ajout d'un brouillard de caisse
  - TBV_A6 : modifie un EBV par la modification du contenu d'un brouillard de caisse
  - TBV_A5 : modifie un EBV par suppression d'un brouillard de caisse

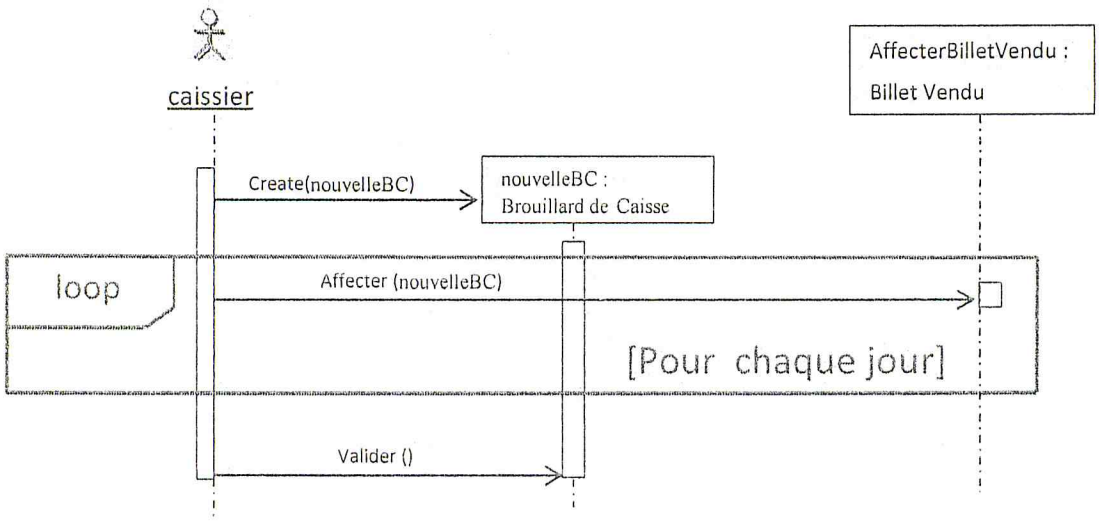


Figure 25 : Diagramme de séquence du scénario TBV_N1

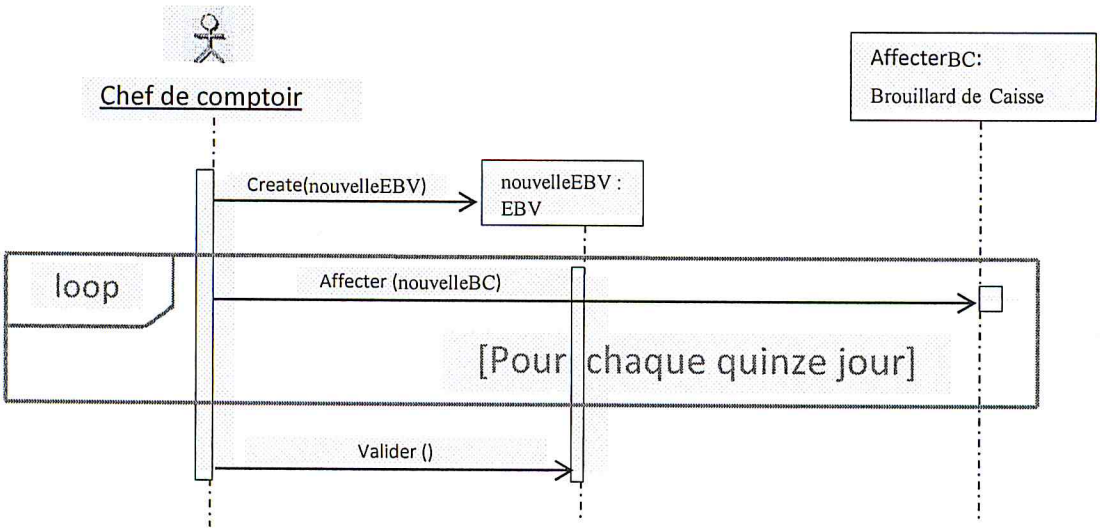


Figure 26 : Diagramme de séquence du scénario TBV_N2

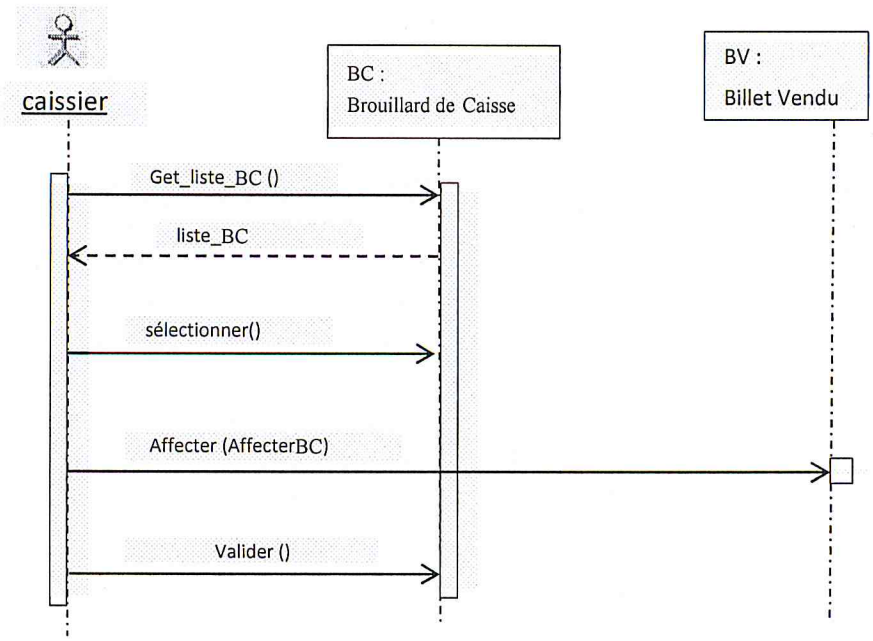


Figure 27 : Diagramme de séquence du scénario **TBV_A1**

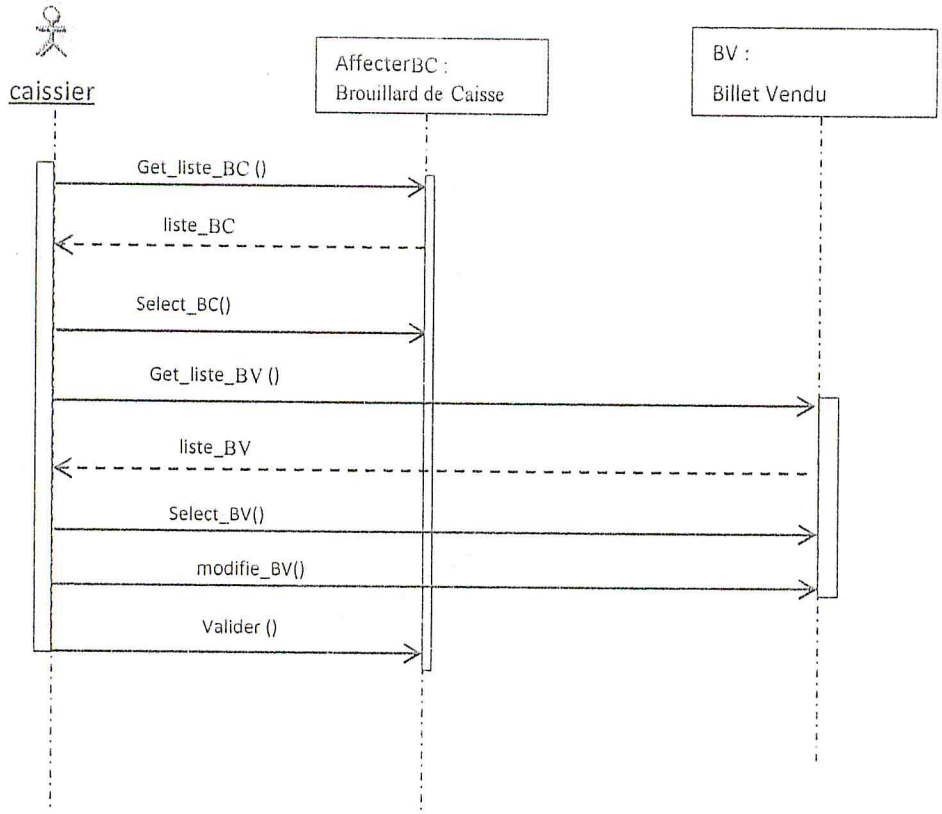


Figure 28 : Diagramme de séquence du scénario TBV_A2

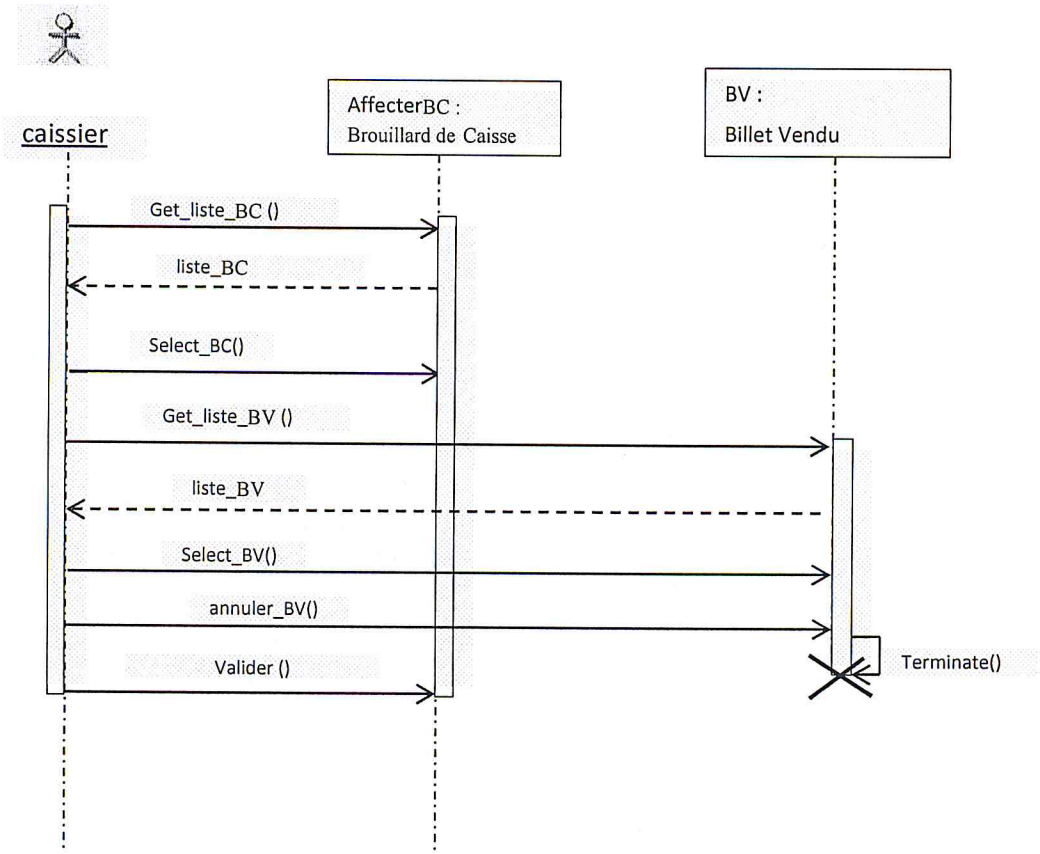


Figure 29 : Diagramme de séquence du scénario **TBV_A3**

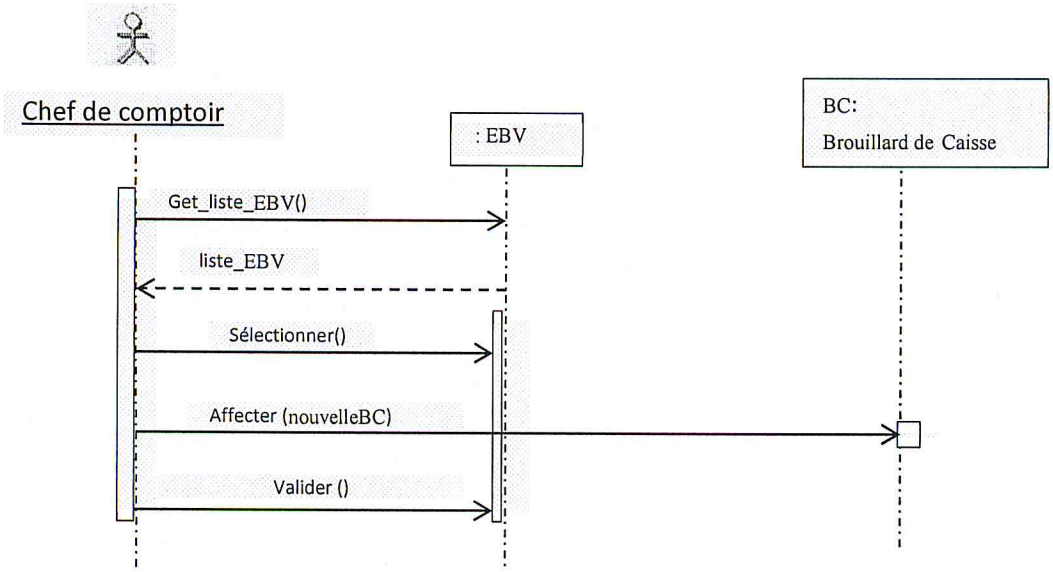


Figure 30 : Diagramme de séquence du scénario **TBV_A4**

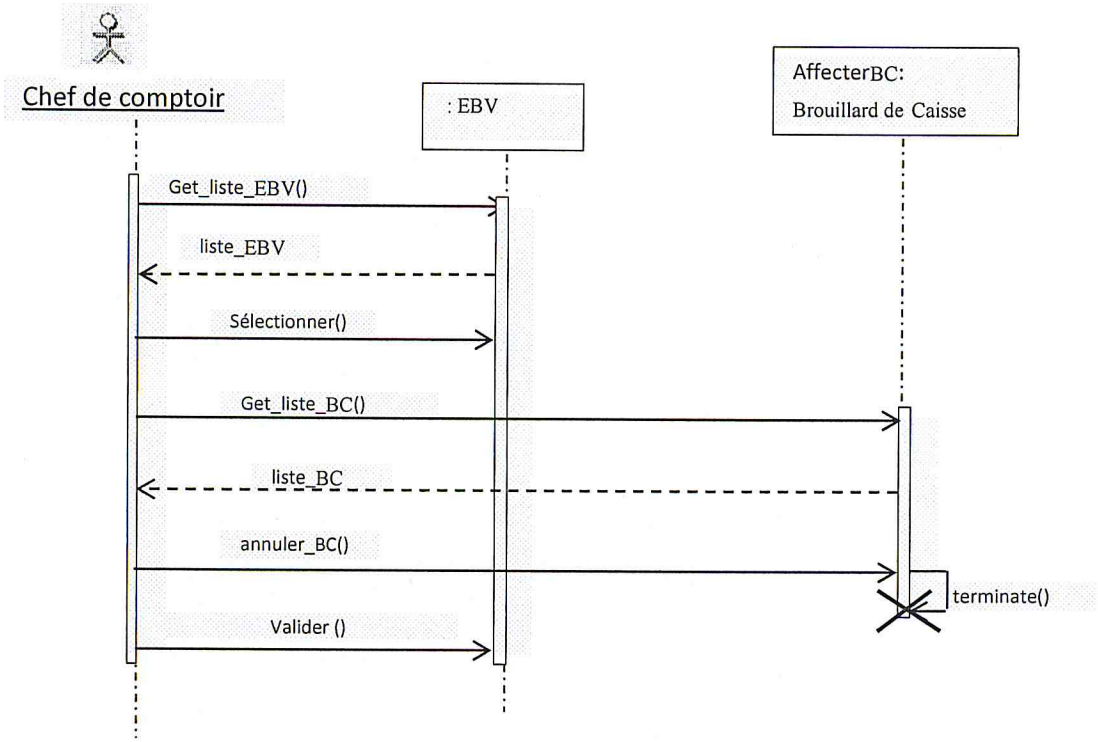


Figure 32 : Diagramme de séquence du scénario **TBV_A6**

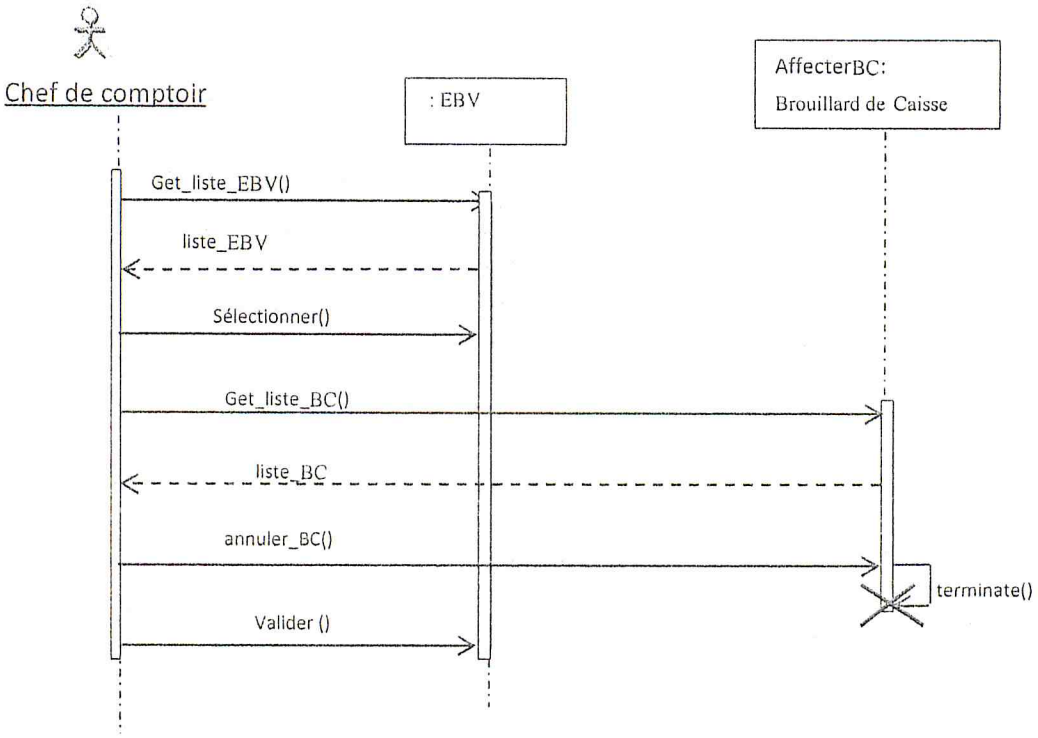


Figure 32 : Diagramme de séquence du scénario TBV_A6

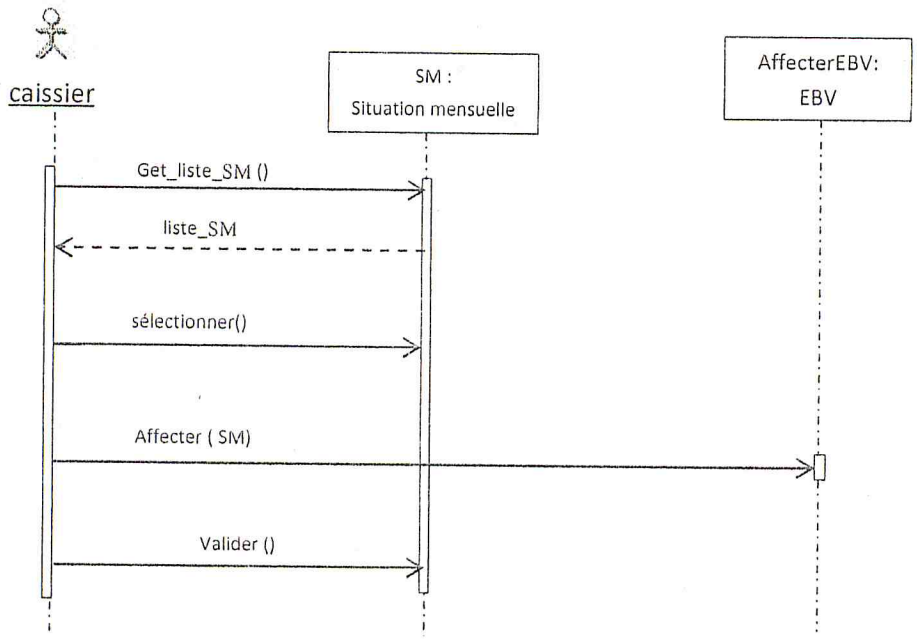


Figure 34 : Diagramme de séquence du scénario SV _A1

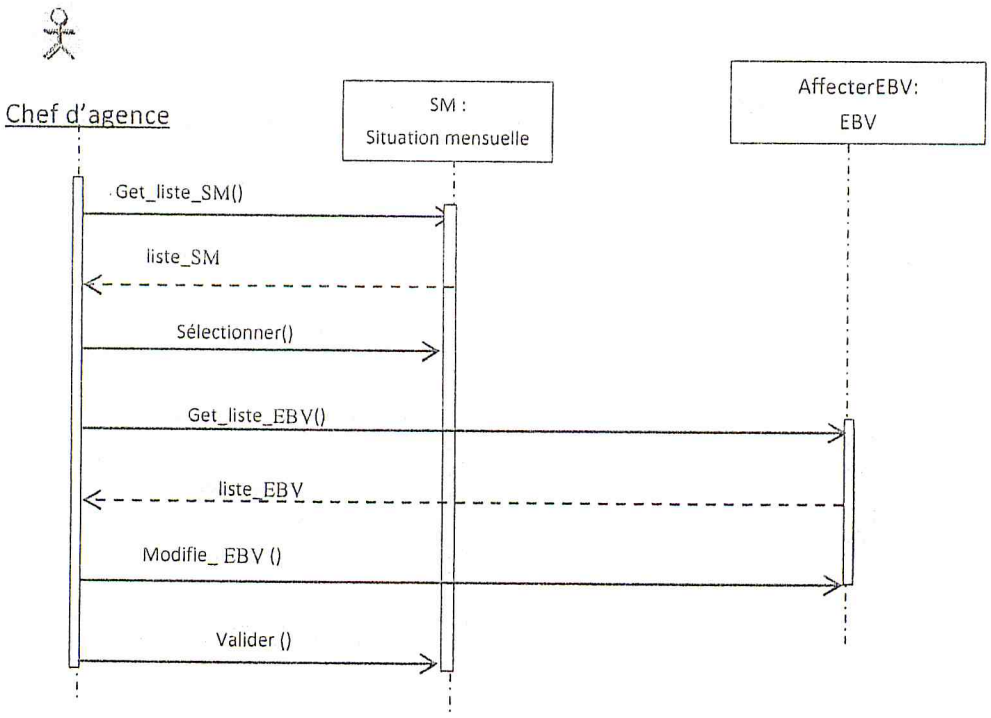


Figure 35 : Diagramme de séquence du scénario SV _A2

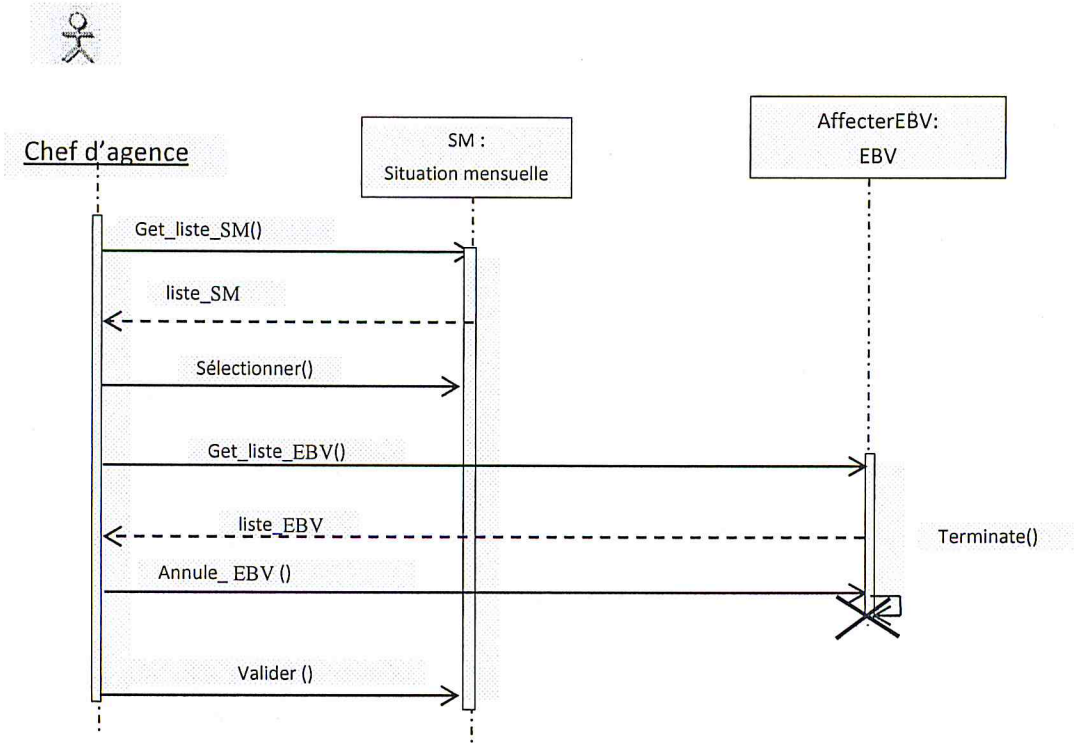


Figure 36 : Diagramme de séquence du scénario SV _A3

3.3.2.3 Le cas d'utilisation : « *Gérer les situations mensuelles* » :GSM*

- **Les scénarios nominaux :**
  - GSM _N : créer une nouvelle situation mensuelle
- **Les scénarios alternatifs :**
  - GSM _A1 : modifie une situation mensuelle globale par ajout d'une situation mensuelle
  - GSM _A2 : modifie une situation mensuelle globale par la modification du contenu d'une situation mensuelle
  - GSM _A3 : modifie une situation mensuelle globale par suppression d'une situation mensuelle

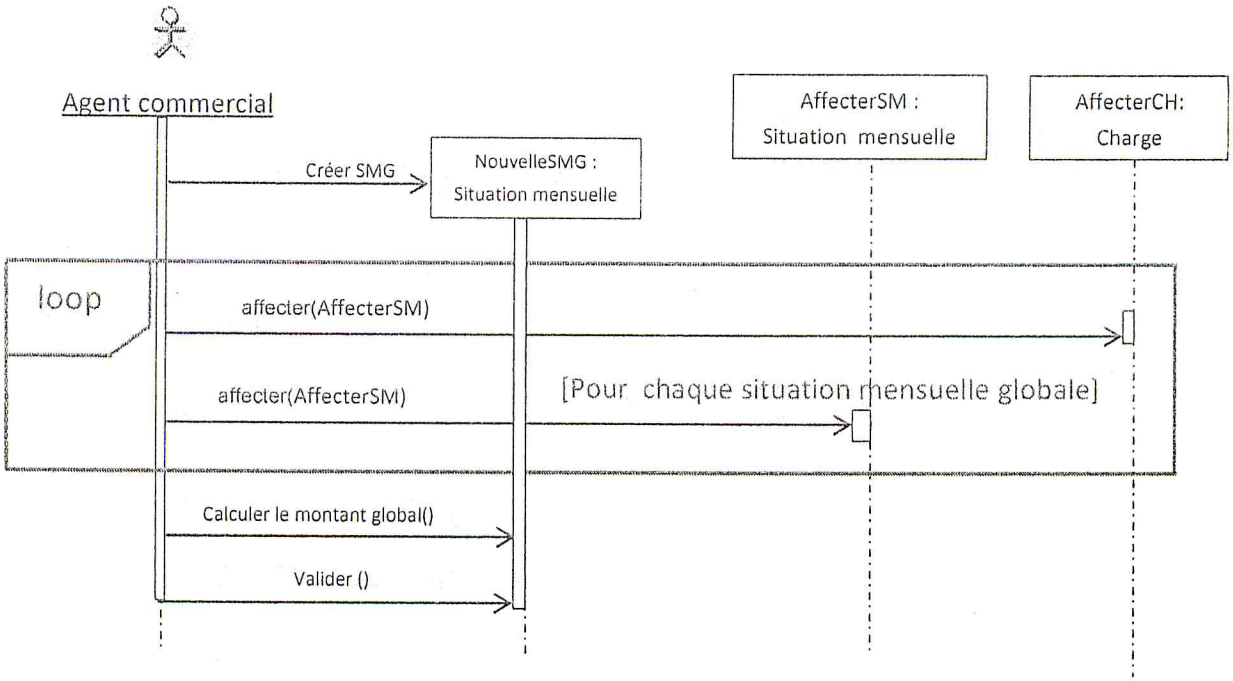


Figure 37 : Diagramme de séquence du scénario GSM_N1

GSM* : Gérer les situations mensuelles

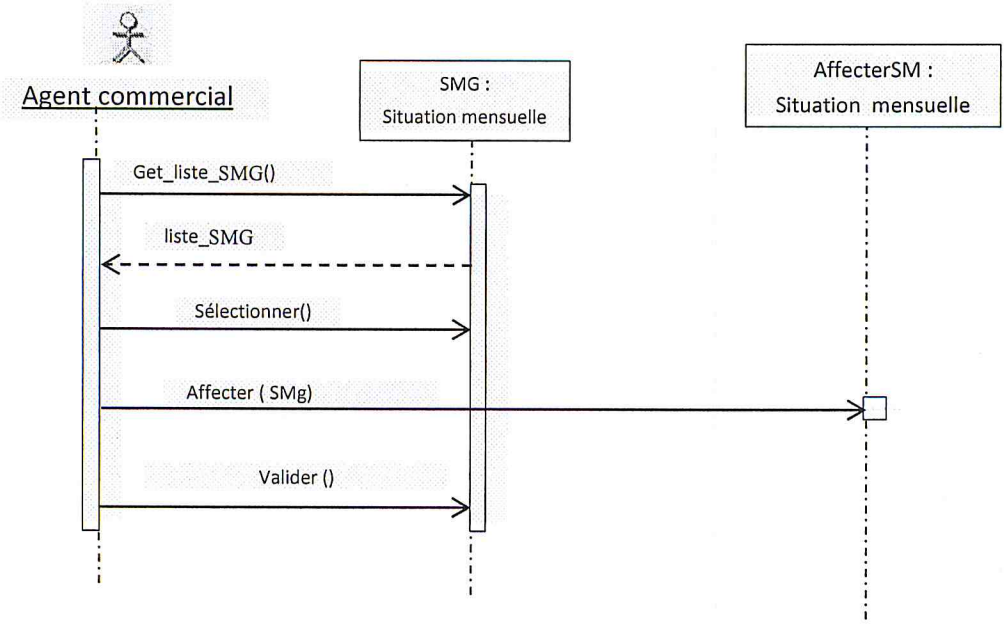


Figure 38 : Diagramme de séquence du scénario GSM _A1

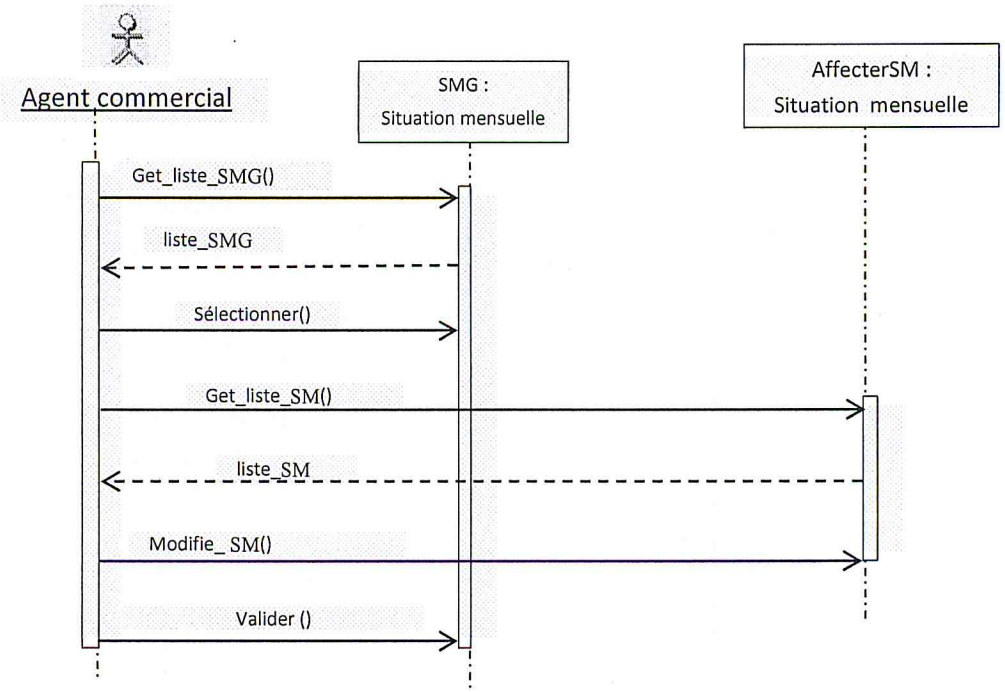


Figure 39 : Diagramme de séquence du scénario GSM _A2

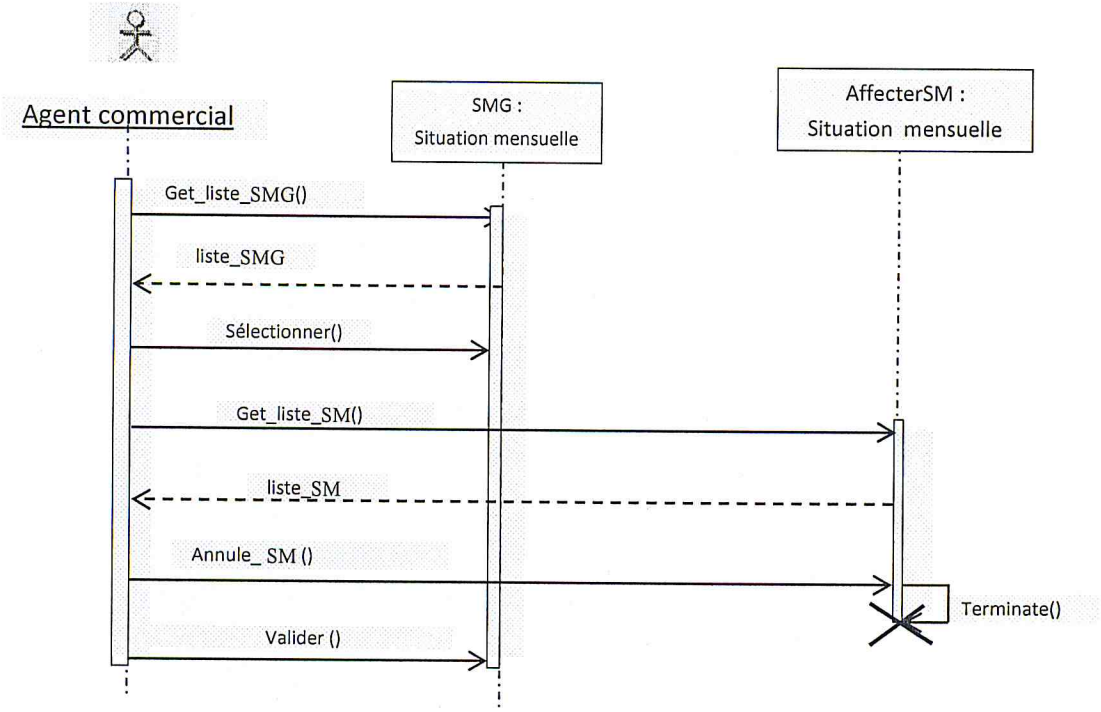


Figure 40 : Diagramme de séquence du scénario GSM _A3

3.3.2.4 Le cas d'utilisation : « *Consulter l'EBV et les coupons comptabilité* » : CECC*

❖ **Les scénarios nominaux :**

➢ CECC _N : vérifier les copies des EBVs par rapport aux coupons compatibilité

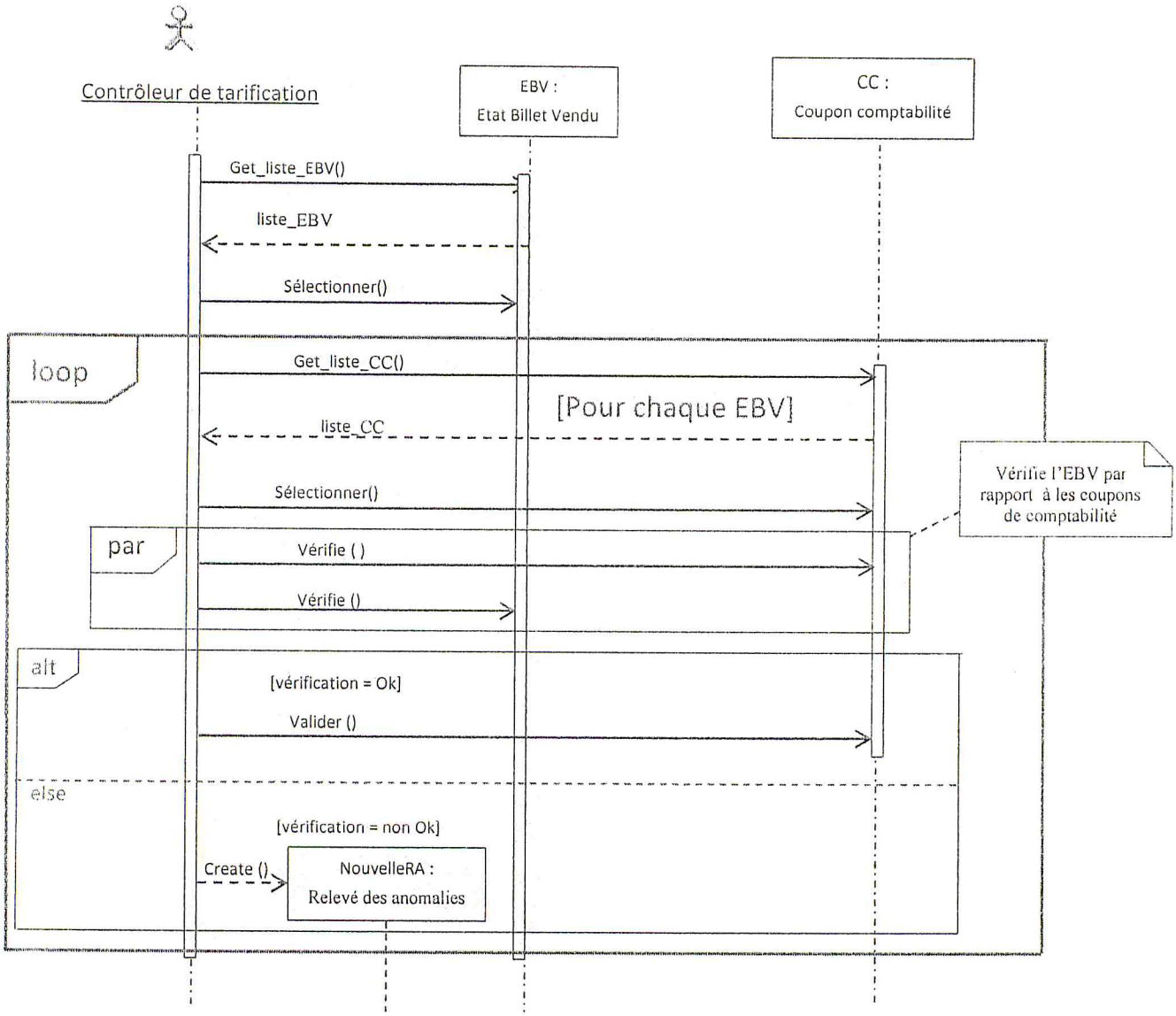


Figure 41 : Diagramme de séquence du scénario CECC _N

CECC *: Consulter l'EBV et les coupons comptabilité

3.3.2.5 Le cas d'utilisation : « *Réaliser le relevé des anomalies* » : RRA*

❖ **Les scénarios nominaux :**

RRA _N : vérifier les copies des EBVs par rapport aux brouillards de caisse

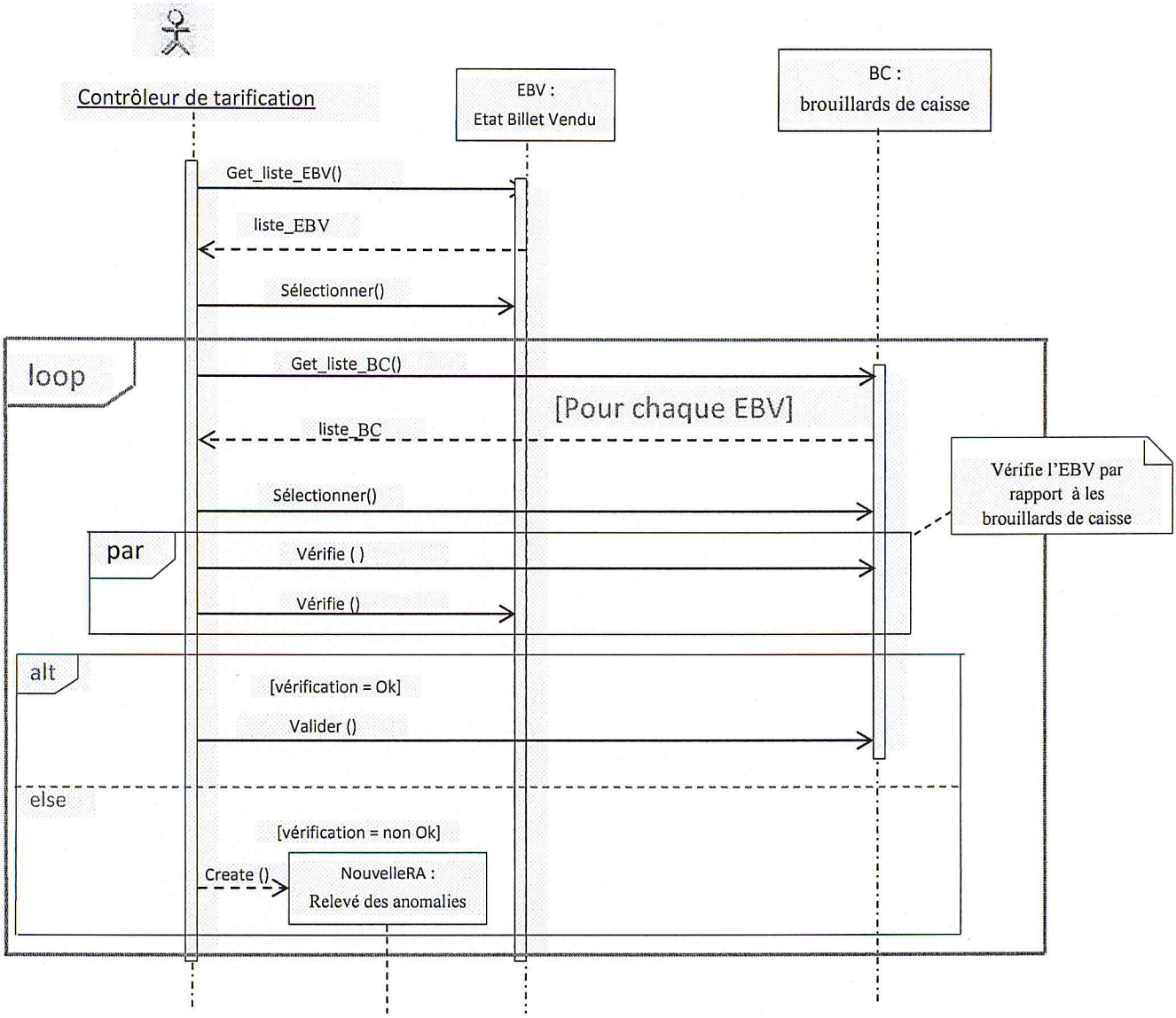


Figure 42 : Diagramme de séquence du scénario RRA _N

RRA *: Réaliser le relevé des anomalies

3.3.2.6 Le cas d'utilisation : « *Calculer l'écart entre les billets vendus et voyagés*» : CEBVV*

❖ **Les scénarios nominaux :**

CEBVV _N : vérifier les copies des EBVs par rapport au manifeste

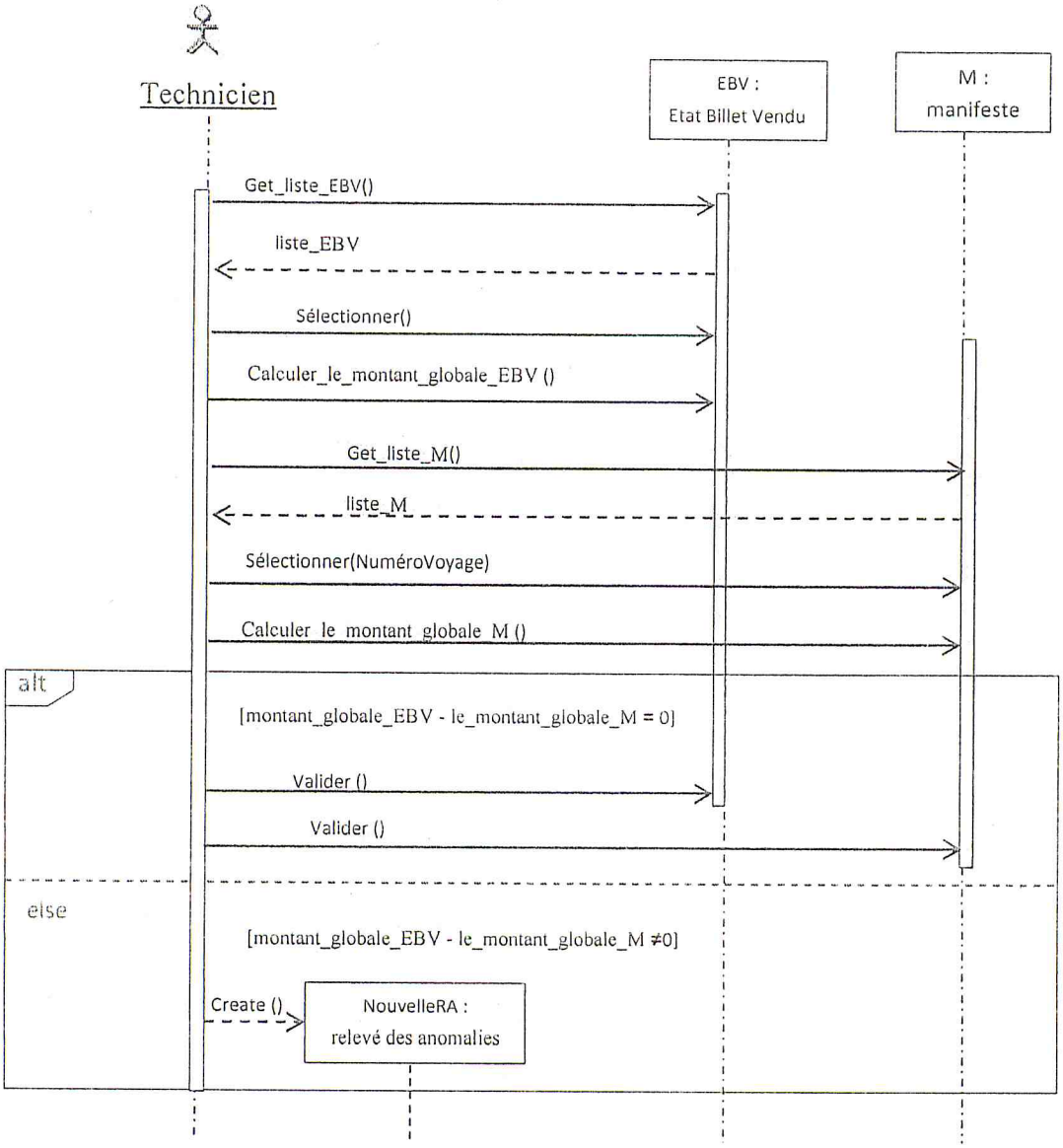


Figure 43 : Diagramme de séquence du scénario CEBVV _N

CEBVV *: Calculer l'écart entre les billets vendus et voyages

3.3.3. Construire les diagrammes d'états :

Maintenant que les scénarios ont été formalisés, la connaissance de l'ensemble des interactions entre objets permet de se représenter les règles de gestion dynamique du système. Il faut cependant se focaliser sur les classes aux comportements les plus riches de 0manière à développer précisément certaines de ces règles dynamiques. On recourt à cet effet au concept de *machine à états finis*, qui consiste à s'intéresser au cycle de vie d'un objet générique d'une classe particulière au fil de ses interactions avec le reste du monde, dans tous les cas possibles. Cette vue locale d'un objet, décrivant comment il réagit à des événements en fonction de son état courant et passe dans un nouvel état, est représentée graphiquement sous forme d'un *diagramme d'états*[Rocques & Vallée, 2004]

Définition :

Un *état* représente une situation durant la vie d'un objet pendant laquelle :

- il satisfait une certaine condition ;
- il exécute une certaine activité ;
- ou bien il attend un certain événement.

Un objet passe par une succession d'états durant son existence. Un état a une durée finie, variable selon la vie de l'objet, en particulier en fonction des événements qui lui arrivent. [Rocques & Vallée, 2004]

3.3.3.1 Diagramme d'état de classe brouillard de caisse :

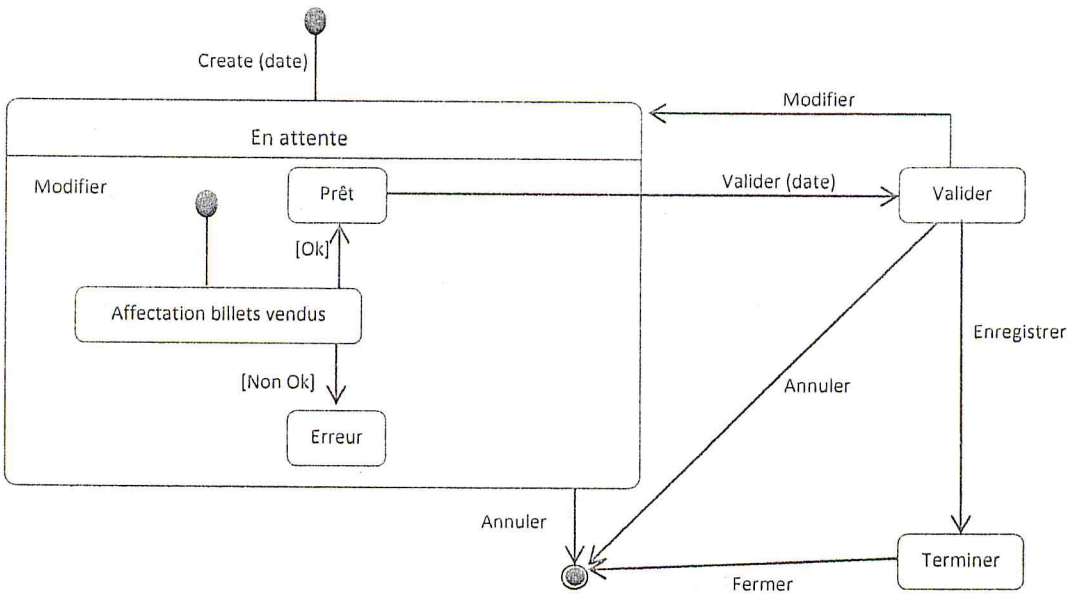


Figure 44 : Diagramme d'état de classe brouillard de caisse

3.3.3.2 Diagramme d'état de classe situation mensuelle global:

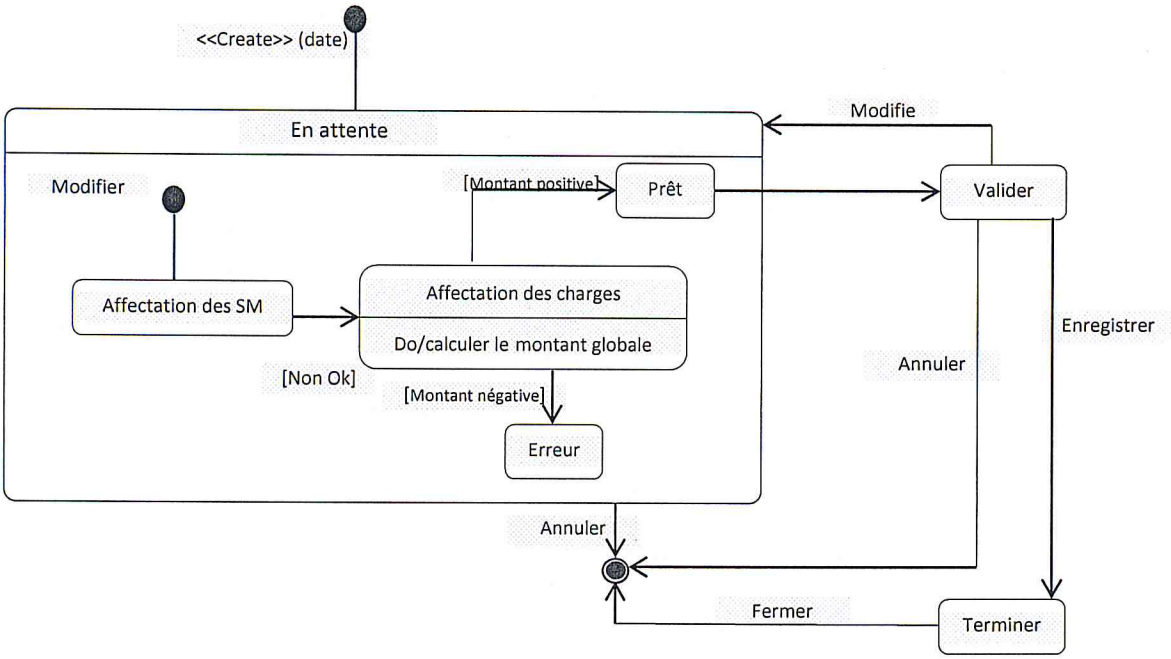


Figure 45 : Diagramme d'état de classe situation mensuelle global

3.3.4.1 Raffinage des méthodes des classes d'objets :

classe	méthode	Description
Billet	Créer_Nouv_billet()	Créer un nouveau billet
	Modifie_billets()	Modifier le continue de billet
	Annuler_billets()	Annuler un billet
	Valider_billet()	Valider un billet
EVB	Créer_Nouv_EBV()	Créer un nouveau EBV
	Modifie_EBV()	Modifier le continue d'EBV
	Annuler_EBV()	Annuler un EBV
	Valider_EBV()	Valider un EBV
Brouill_caiss	Créer_Nouv_BC()	Créer un nouveau brouillard de caisse
	Modifie_BC()	Modifier le continue de brouillard de caisse
	Annuler_BC()	Annuler un brouillard de caisse
	Valider_BC()	Valider un brouillard de caisse
Situa_mens	Créer_Nouv_SM()	Créer une nouvelle situation mensuelle
	Modifie_SM()	Modifier le continue de la situation mensuelle
	Annuler_SM()	Annuler une situation mensuelle
	Valider_SM()	Valider une situation mensuelle
Sit_men_glo	Créer_Nouv_SMG()	Créer une nouvelle situation mensuelle globale
	Modifie_SMG()	Modifier le continue de situation mensuelle global
	Annuler_SMG()	Annuler une situation mensuelle global
	Valider_SMG()	Valider une situation mensuelle globale
Utilisateur	Créer_Nouv_Utl()	Créer un utilisateur
	Modifie_ Utl()	Modifier le continue d'utilisateur
	Annuler_ Utl ()	Annuler un utilisateur
	Valider_ Utl()	Valider un utilisateur

Tableau 5 : Raffinage des méthodes des classes d'objets

4. CONCEPTION :

La phase de conception succède à la phase d'analyse dans le processus de développent.

Elle se base sur les différents modèles de l'analyse : modèle statique et modèle dynamique.

4.1 Architecteur de l'application :

La technologie Objet requiert une architecture. C'est cette architecture qui organise les interactions entre objets. On a l'habitude de regrouper ces objets en classes, ces classes en domaines, et ces domaines en couches.

Les couches permettent de présenter l'architecture de l'application. Les équipes de réalisation s'attribuent alors des responsabilités sur le développement de chaque couche. Aussi, si modéliser est indispensable, construire une architecture à couche est un critère de qualité dans le cadre d'un développement Objet. Reste à choisir le nombre de couches et à définir leur contenu [Rocques & Vallée, 2004].

4.1.1 Architecteur 3-Tiers :

Pour avoir une architecture robuste, modulable et évolutive, il nous faut utiliser le principe de « couche ». Nous allons donc séparer au maximum les différents types de traitement de l'application (Dao, Métier, Présentation).

Ceci correspond à une architecture 3-Tiers :

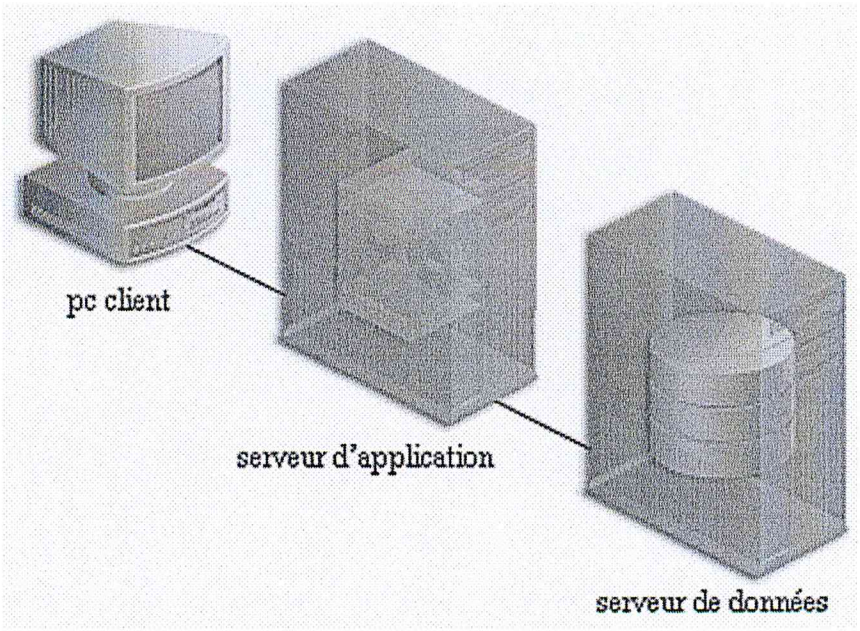


Figure 46 : Architecteur 3-Tiers

4.2 Concevoir les classes :

Les classes qui proviennent de l'analyse ne sont pas toujours conformes aux possibilités du langage d'implémentation. Dans certains cas, une analyse orientée objet est réalisée dans un langage non objet. La transformation des classes en codage est alors particulièrement importante pour conserver la trace du passage de l'analyse au code. Java offre bien entendu une transformation beaucoup plus directe. Certaines formes telles que les métaclasses, les états ou les héritages multiples sont cependant tolérées par les analystes mais inconnues de Java. Concevoir les classes consiste tout d'abord à expliciter comment ces concepts devront être traduits dans le code

Concevoir les classes, c'est aussi en introduire de nouvelles soit pour prendre en charge des responsabilités purement techniques, soit pour décharger une classe d'analyse de certains de ces aspects techniques. Les liens qui rattachent ces classes entre elles doivent le plus souvent correspondre à des *design patterns*.

Concevoir les classes, c'est enfin redistribuer les messages et les événements du modèle dynamique sur les différentes couches de conception. Pour les systèmes 3-tiers, nous pensons plus particulièrement à la redistribution des échanges entre les couches métier et application. Ces échanges s'appuyant sur les capacités d'un réseau physique doivent faire l'objet d'optimisations. Dans cette optique, il convient que les diagrammes d'états soient retravaillés au niveau de la conception [Rocques & Vallée, 2004].

Le Design Pattern Etat :

Le *design pattern* État [Gamma 95] est une façon de concevoir le diagramme d'états d'une classe d'analyse. La gestion des états est déléguée de sorte qu'à chaque état corresponde une classe du patron. Une classe gère ainsi les activités et les transitions attachées à l'état qu'elle représente [Rocques & Vallée, 2004].

➢ **Conception des états de la classe Brouillard de caisse:**

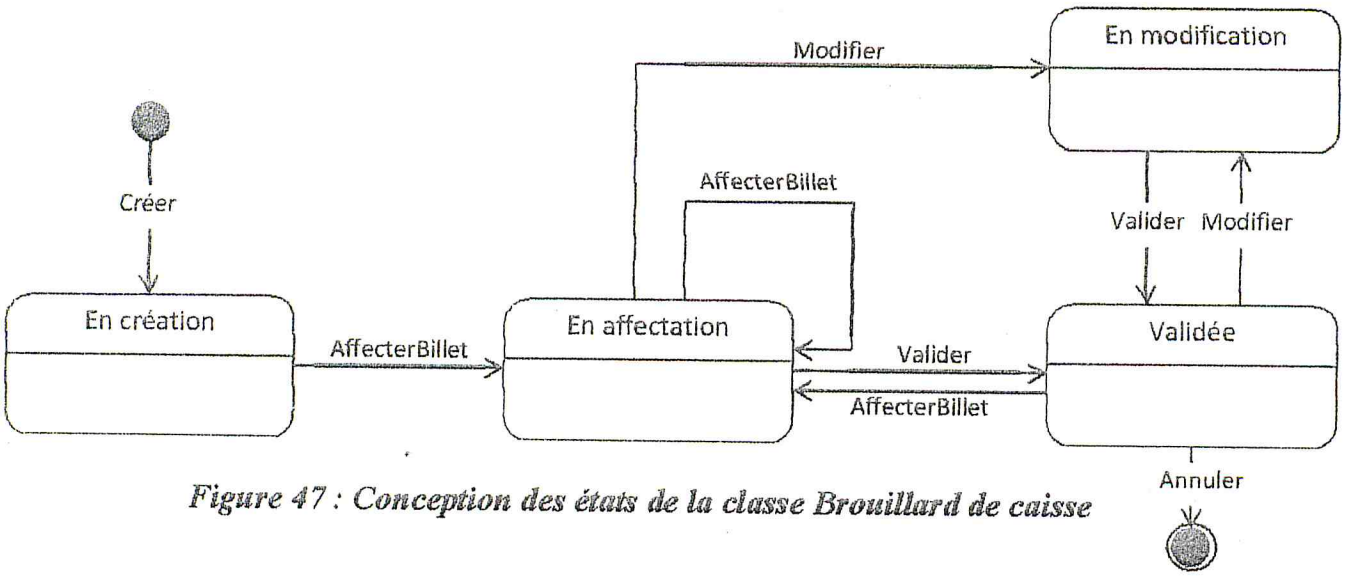


Figure 47 : Conception des états de la classe Brouillard de caisse

➢ **Environnement résultant des états de la classe Brouillard de caisse :**

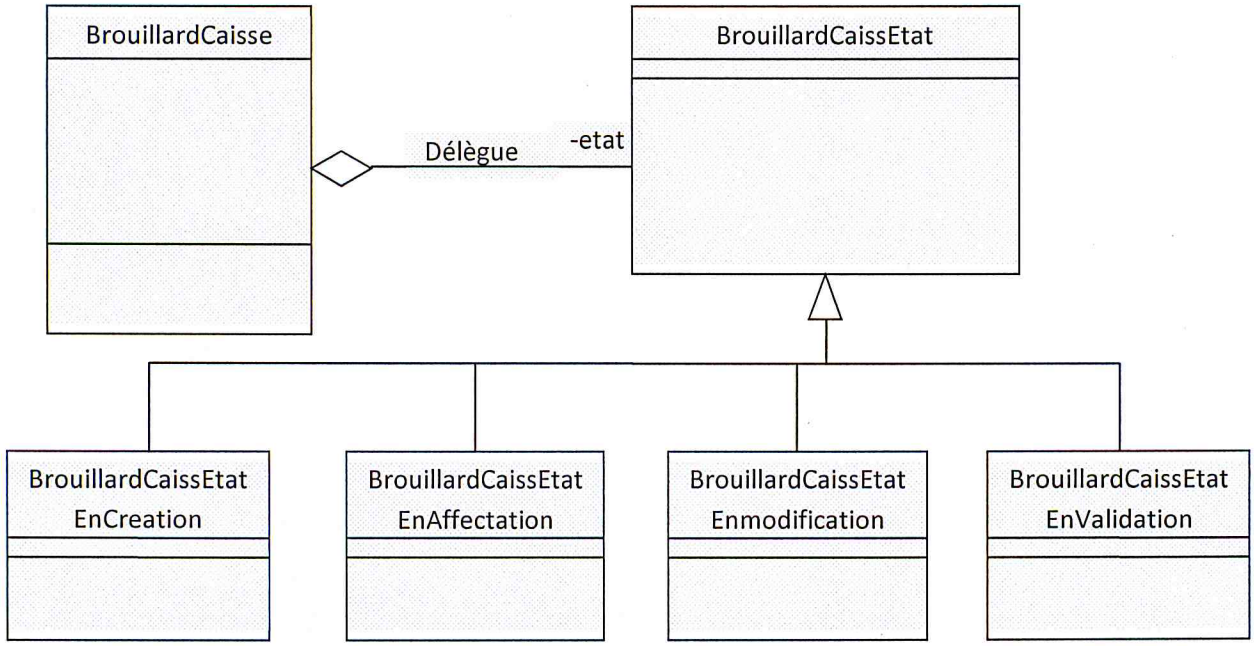


Figure 48 : Environnement résultant des états de la classe Brouillard de caisse

➢ **Conception des états de la classe Situation mensuelle globale:**

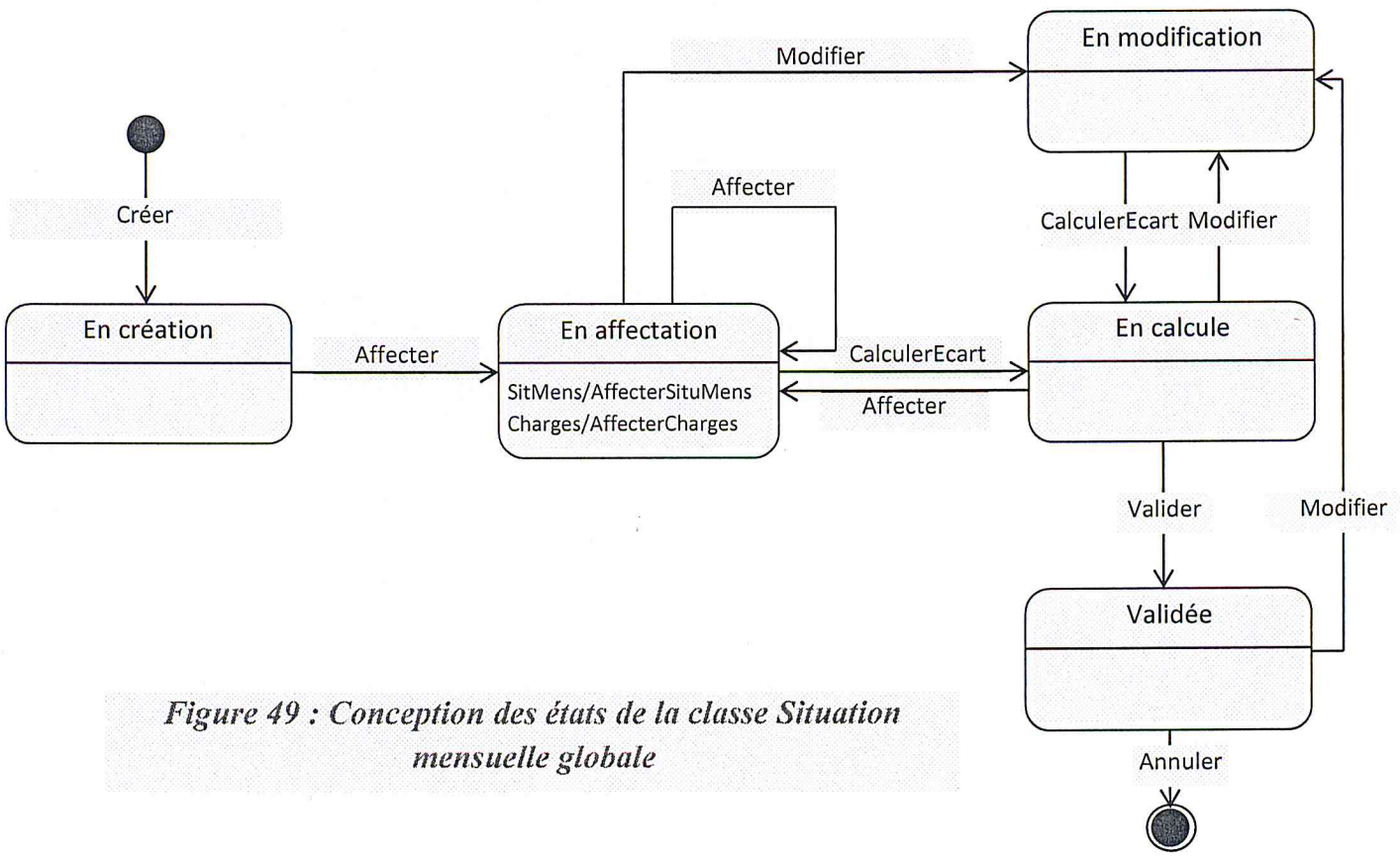


Figure 49 : Conception des états de la classe Situation mensuelle globale

➢ **Environnement résultant des états de la classe Brouillard de caisse :**

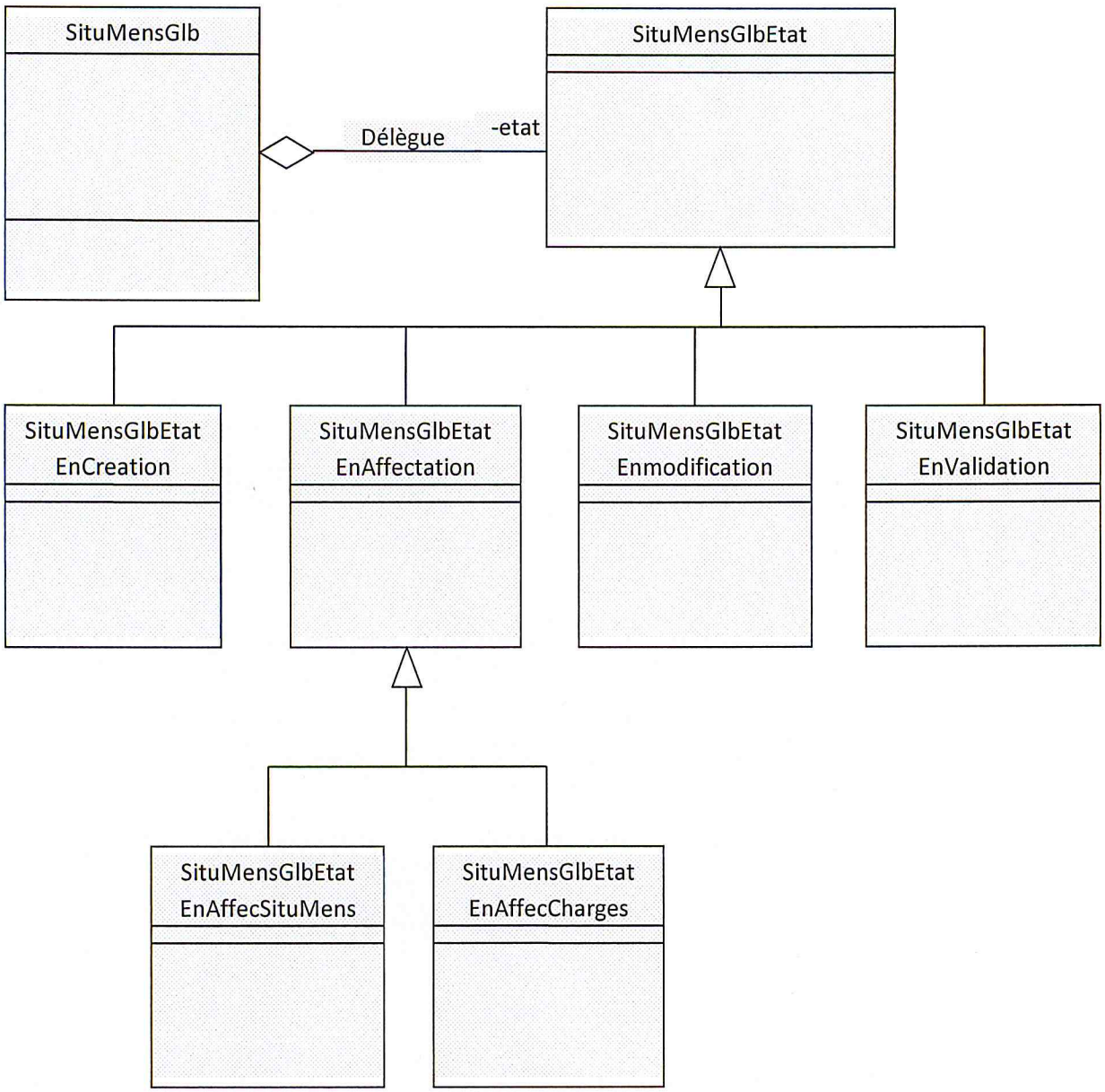


Figure 50 : Environnement résultant des états de la classe Brouillard de caisse

➢ **Conception des états de la classe Lettre d'anomalie:**

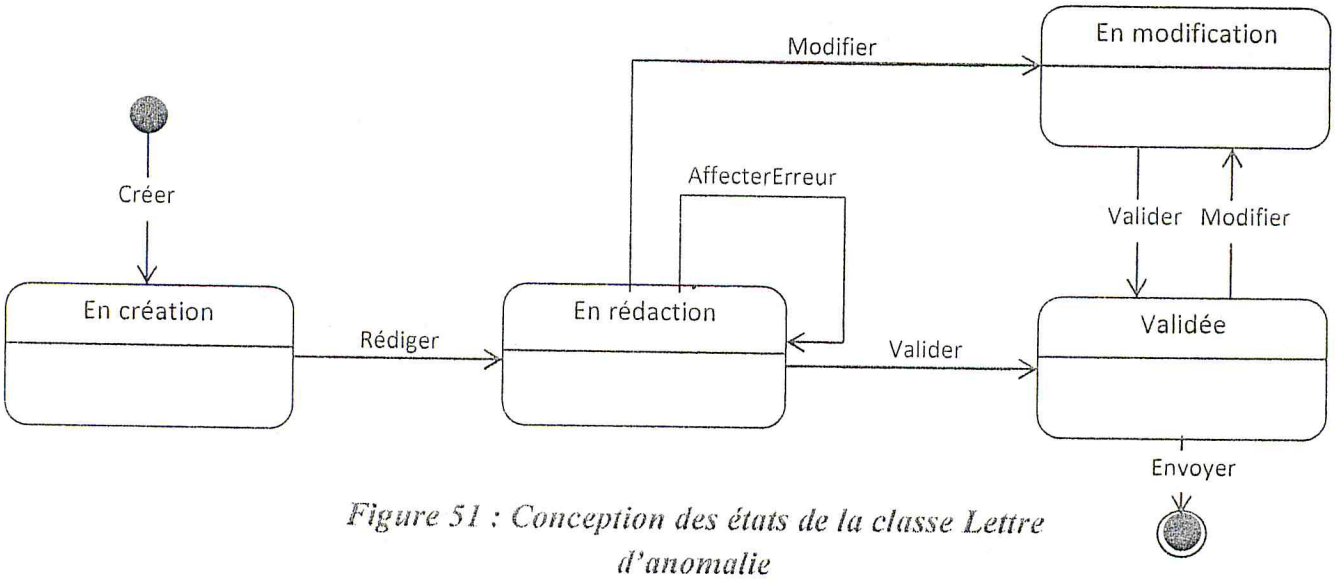


Figure 51 : Conception des états de la classe Lettre d'anomalie

➢ **Environnement résultant des états de la classe Brouillard de caisse :**

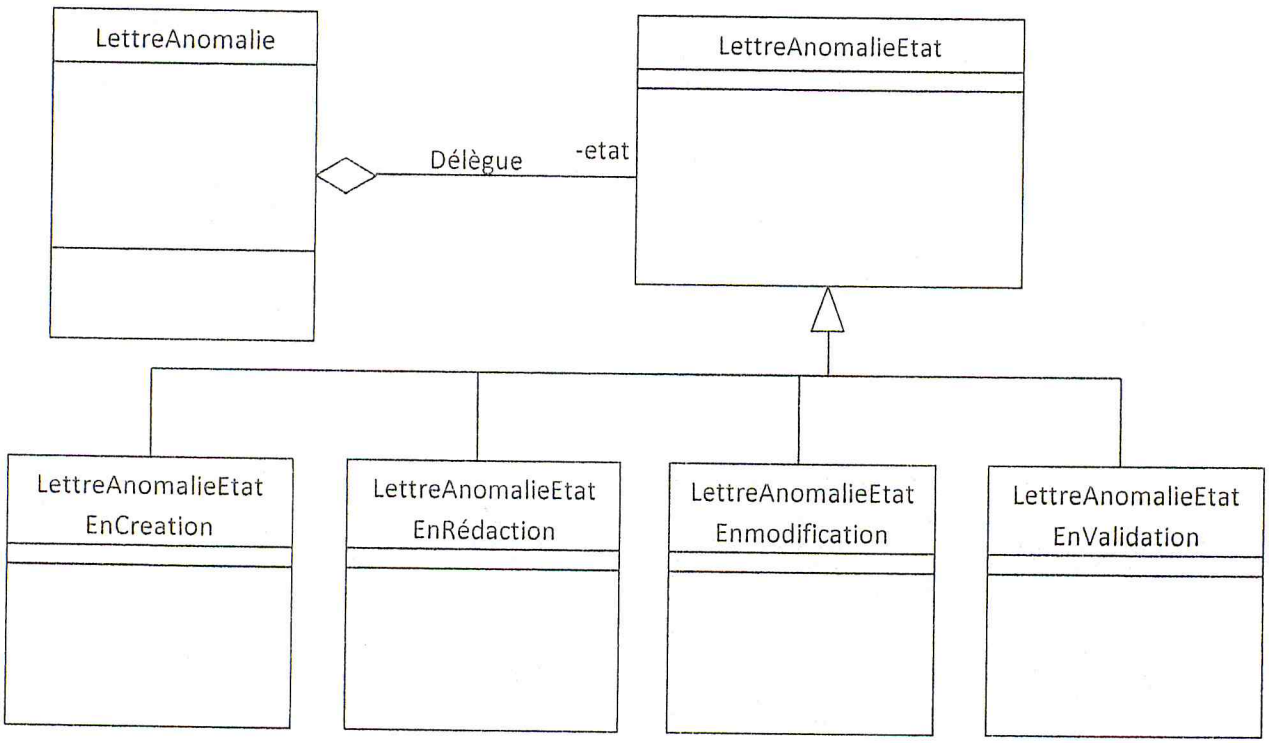


Figure 52 : Environnement résultant des états de la classe Brouillard de caisse

4.3 Concevoir les associations :

L'association est un concept inconnu de la plupart des langages orientés objet. Elle se transforme en attribut ou en tableau d'attributs [Rocques & Vallée, 2004].

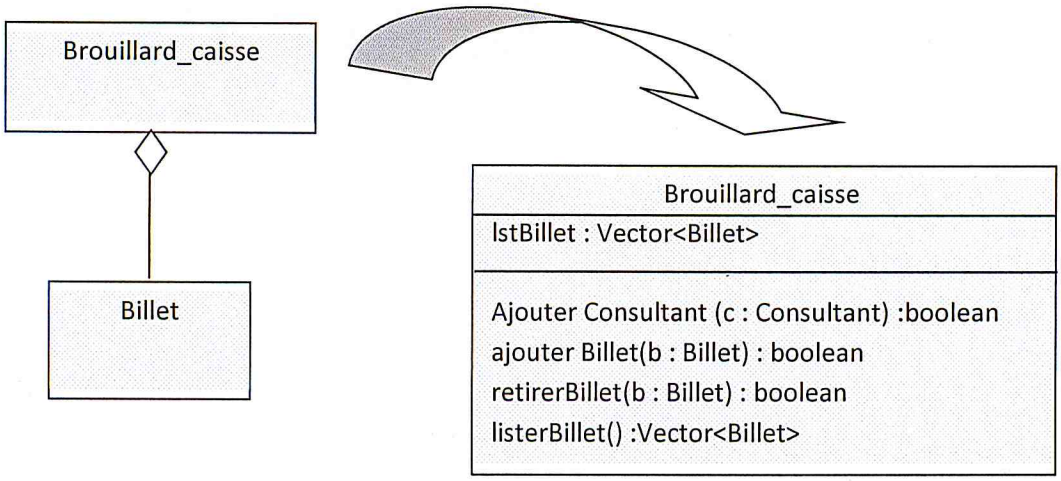


Figure 53 : Exemple d'une conception d'associations par des attributs et les opérations nécessaires à sa gestion

Le Design Pattern Itérateur :

Le design pattern Itérateur est une façon de concevoir l'accès séquentiel à un ensemble d'objets, sans avoir à exposer la structure interne de l'ensemble. Par ailleurs, l'itérateur offre une interface de parcours standard qui uniformise et facilite la manipulation des différents types de liste d'un même projet. Il offre enfin la possibilité de parcourir simultanément et indépendamment le même ensemble d'objets par des tâches parallèles.

L'objectif de l'itérateur est de déléguer les opérations de parcours d'une association. Par ailleurs, il a pour rôle de gérer l'état d'un élément courant, auquel on peut accéder séquentiellement au précédent ou au suivant [Rocques & Vallée, 2004].

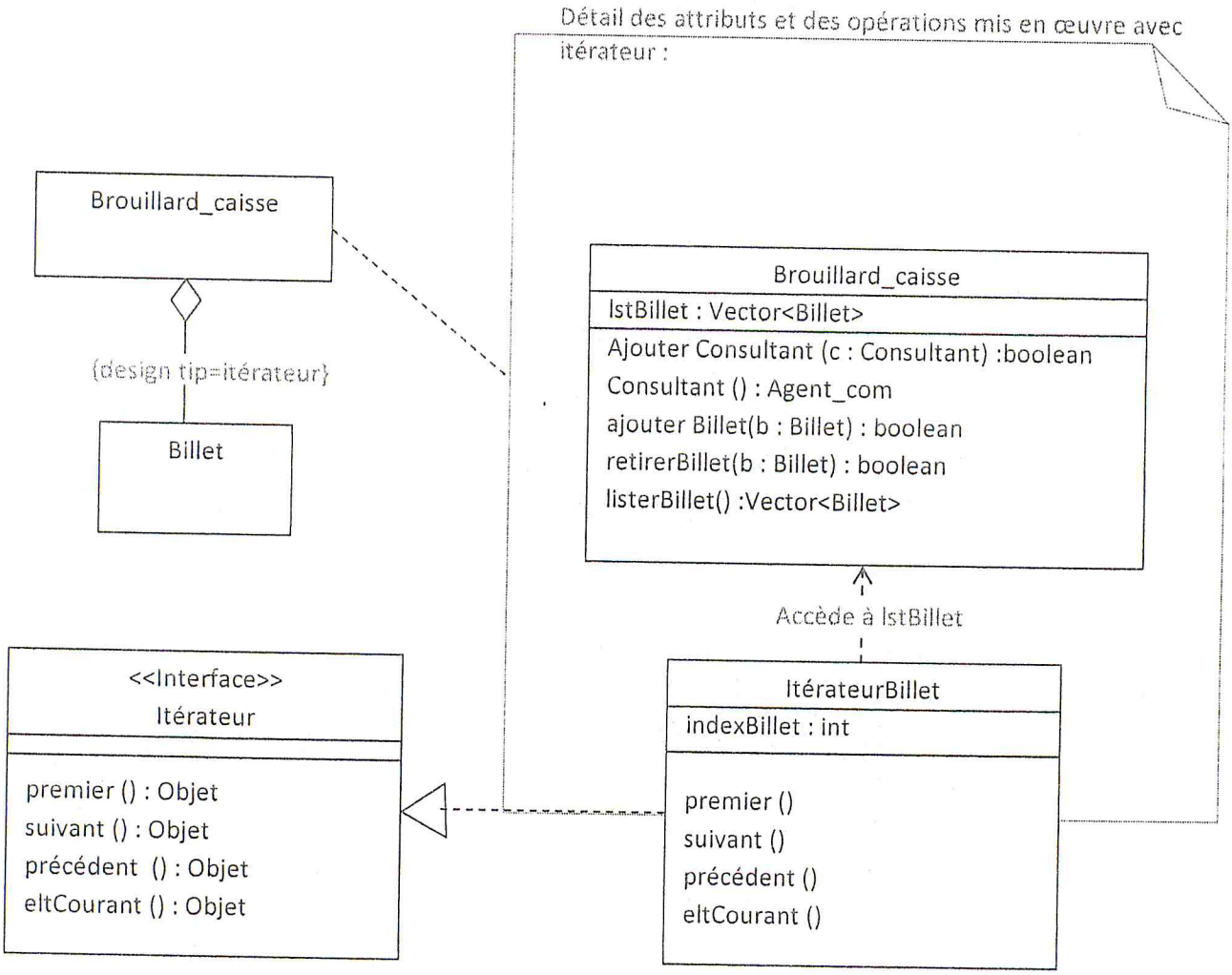


Figure 54 : Documentation du mécanisme {designTip=itérateur}

4.4 Concevoir les attributs :

La conception des attributs consiste principalement à définir le type des attributs identifiés en analyse.

Concevoir les attributs, c'est également spécifier leur visibilité et leur mode d'accès. Par défaut, un attribut est privé et ce principe reste invariable dans notre conception. La prise en compte des propriétés UML {changeable}, {readOnly} ou {frozen} est donc implicitement gérée par des opérations d'accès [Rocques & Vallée, 2004].

Billet
Numéro billet : int {frozen}
Numéro réservation : int {frozen}
Date émission : Date {readOnly}
Info additionnel : String {changeable}
Code de voyage: int {frozen}
Code de Véhicule : int {frozen}
Code de passager : int {frozen}
Ancien billet : int {readOnly}
Réduction : Monnaie {changeable}
Installation : Installation {changeable}
Prix HT : Monnaie {changeable}
Supplément : Monnaie {changeable}
Frai port : Monnaie {frozen}
Fiscale : Monnaie {frozen}
D.T : Monnaie {frozen}

Agence
Code agence : int {frozen}
Nom agence : String {frozen}
Région : String {readOnly}
Adresse : Adresse {readOnly}
Numéro de téléphone : Téléphone {changeable}

Employé
Code employé : int {frozen}
Nom : String {frozen}
Prénom : String {frozen}
Salaire : Monnaie {changeable}

Charges
Code : int {frozen}
Mois : Mois {changeable}
Agence : String {readOnly}
Charge personnel : Monnaie {changeable}
Charge patronat : Monnaie {changeable}

+

Voyager
Code de voyage : int {frozen}
Trajet : Trajet {changeable}
Date trajet : Date {changeable}
Heure trajet : Heure {changeable}
Navire : Navire {changeable}

Passager
Code de passager : int {frozen}
Nom : String {frozen}
Prénom : String {frozen}
Date de naissance : Date {frozen}

4.5 L'architecture MVC :

MVC est l'abréviation de Model-View-Controller, ce qui signifie en français : "Modèle-Vue-Contrôleur".

Il s'agit d'un design pattern, une technique de programmation. C'est une "façon de programmer et d'organiser son code" bien pensée. [Rocques & Vallée, 2004].

L'architecture MVC vous propose de séparer les éléments de votre programme en 3 parties :

- **Le modèle** : c'est la partie qui contient les données. Le modèle peut par exemple contenir la liste des voyageurs d'un billet, avec leurs noms, prénoms, âges...
- **La vue** : c'est la partie qui s'occupe de l'affichage. Elle affiche ce que contient le modèle. Par exemple, la vue pourrait être un tableau. Ce tableau affichera la liste des voyageurs si c'est ce que contient le modèle.
- **Le contrôleur** : c'est la partie "réflexion" du programme. Lorsque l'utilisateur sélectionne 3 voyageurs dans le tableau et appuie sur la touche "Supprimer", le contrôleur est appelé et se charge de supprimer les 3 voyageurs du modèle.

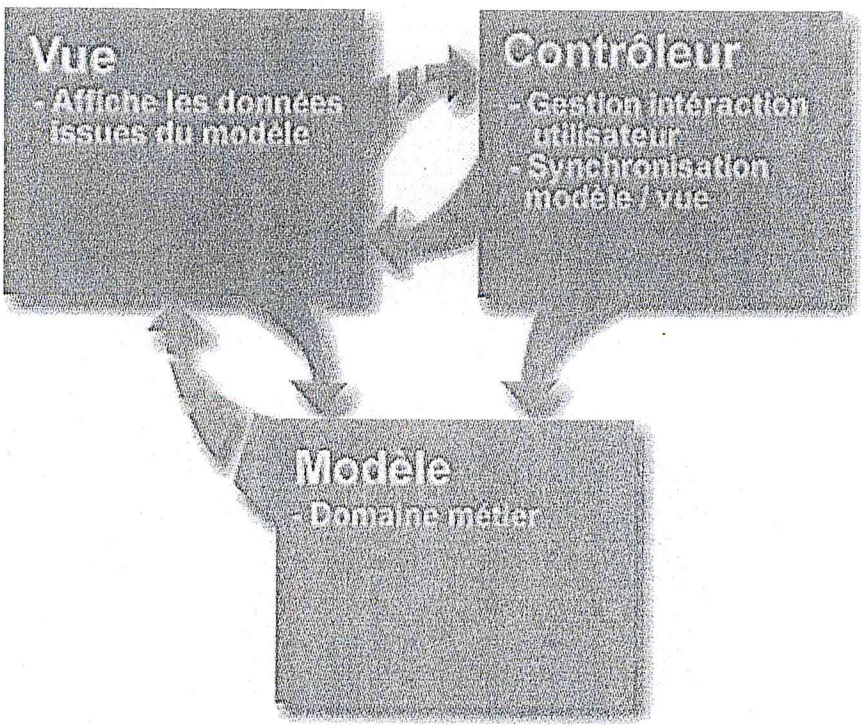


Figure 55 : L'architecture MVC

4.6 Conception de la couche de présentation :

La couche de présentation ou IHM se limite à la partie visible d'une application. Les environnements avec fenêtrage (type Windows) ou pages HTML équipées de mécanismes réflexes en JavaScript, correspondent aux choix technologiques les plus courants. Dans ces environnements, un utilisateur est face à trois grandes familles de concepts.

- Les fenêtres (ou pages) et leur contenu qu'il voit, redimensionne, bouge et réduit, font partie des concepts visuels ;
- Les actions qu'il peut déclencher et les changements d'aspects, font partie du comportement de la présentation ;
- Les flux de données qu'il transmet *via* des listes de choix, des champs d'édition font partie de l'échange d'informations avec l'application.

Concevoir ou documenter une couche de présentation revient à passer en revue ces trois aspects : le visuel, le comportemental et le fonctionnel [Rocques & Vallée, 2004].

❖ Conception de la vue d'édition d'un billet:

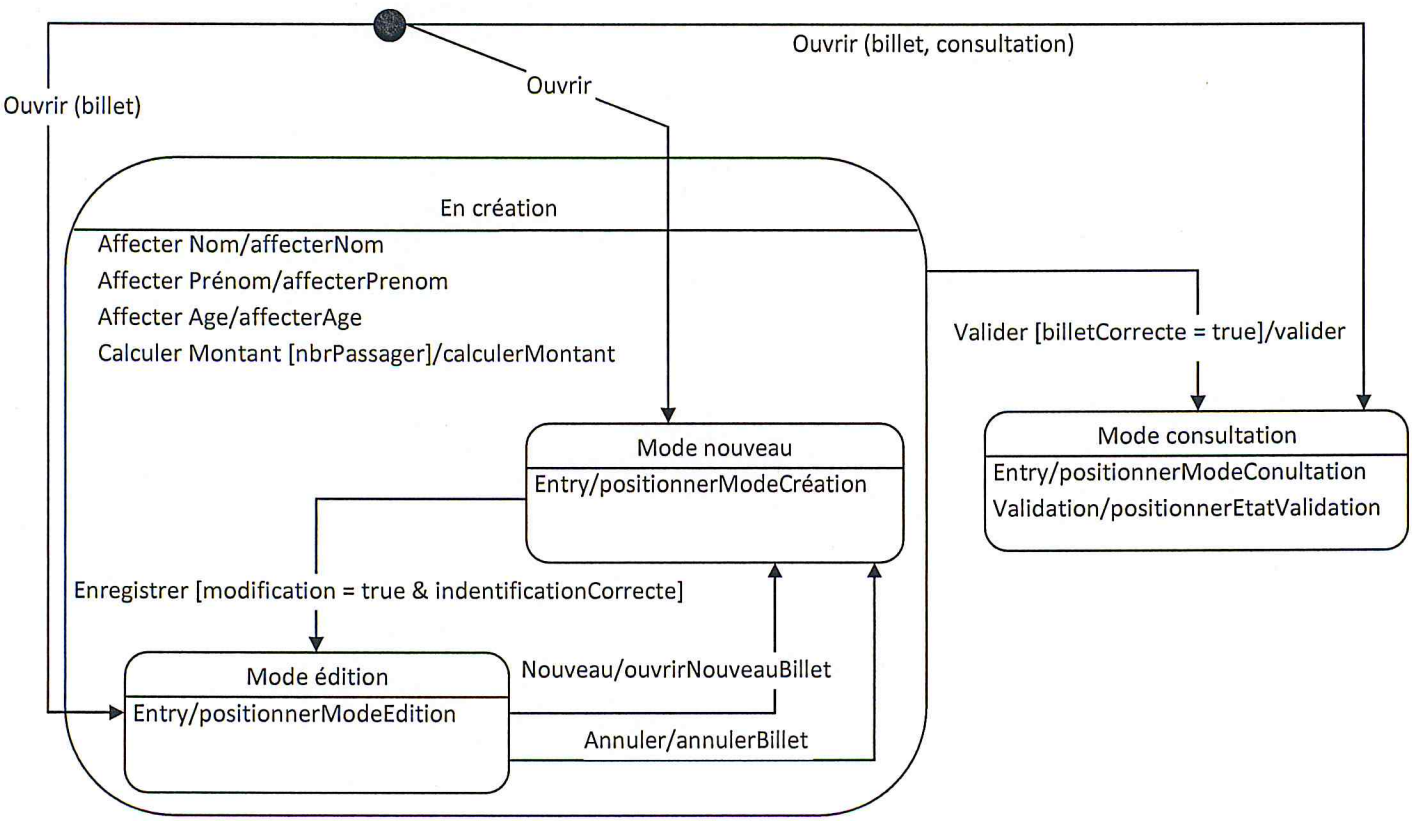


Figure 56 : États de la fenêtre d'édition d'un billet

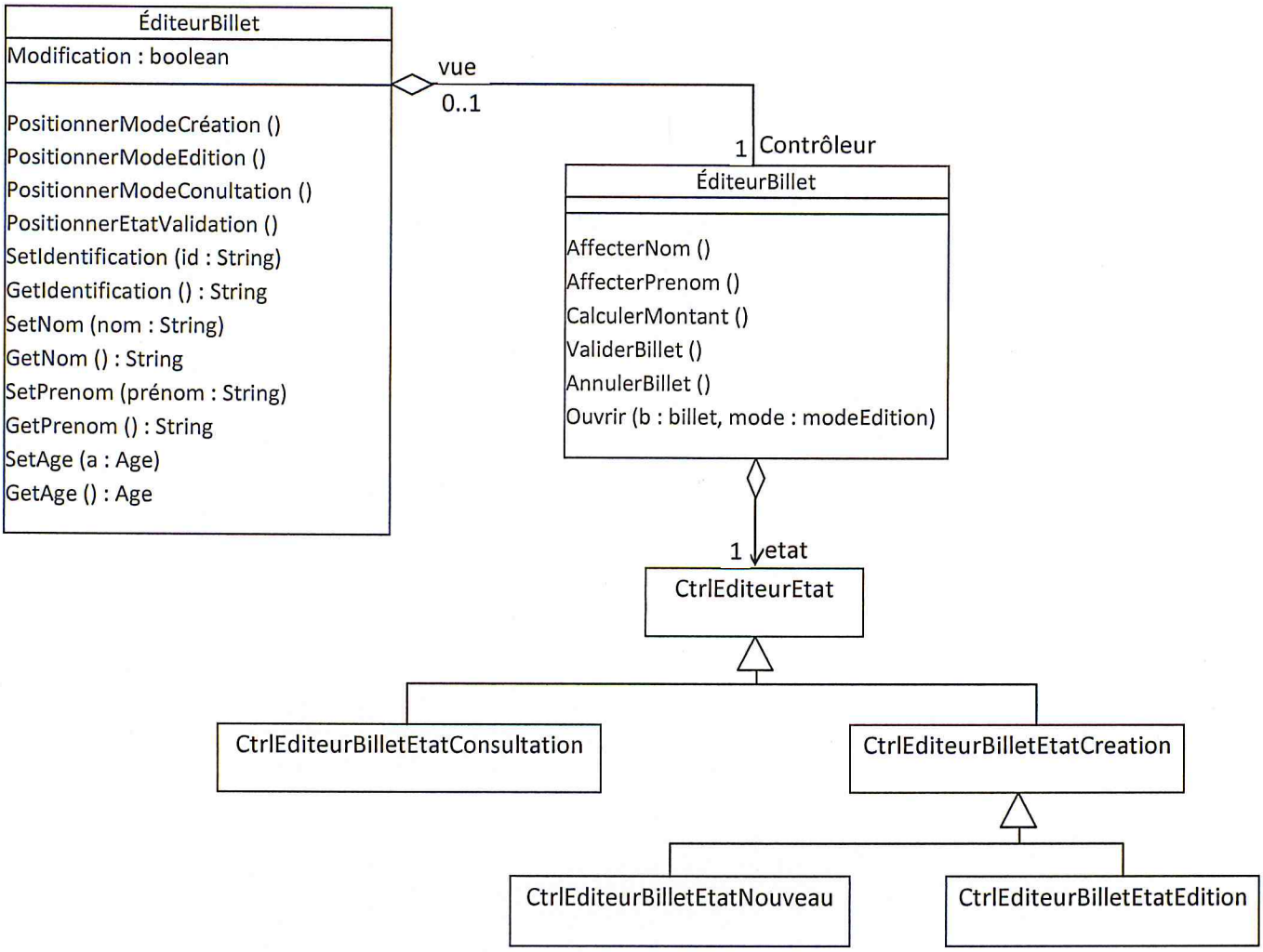


Figure 57 : Structure Vue-Contrôleur-États de la fenêtre Éditeur de billet

- ***Le design pattern observateur :***

Le *design pattern* Observateur [Gamma 95] consiste à synchroniser des objets en minimisant les dépendances qui devraient s'établir entre eux. Chaque objet n'a cependant pas un rôle symétrique : nous y distinguons le sujet et les observateurs.

Le sujet centralise les données et il est unique. Il comprend des opérations permettant aux observateurs d'accéder à ses données.

L'observateur restitue les données du sujet auquel il est abonné, et plusieurs peuvent se synchroniser sur le même sujet.

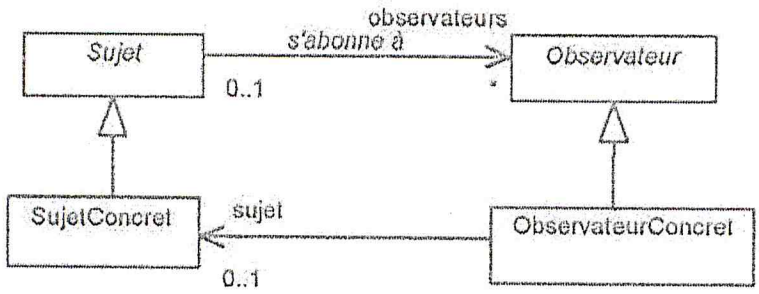


Figure 58 : Structure du design pattern Observateur

4.7 Conception de la couche application :

Le rôle de la couche de l'application consiste à piloter les processus d'interactions entre l'utilisateur et le système. Cette notion peut paraître abstraite, mais il s'agit généralement de mettre en œuvre toutes les règles nécessaires au maintien d'une application cohérente et à l'optimisation des échanges client/ serveur.

Pour toutes ces capacités, la conception d'une application s'appuie sur la définition des documents représentant en fait l'image mémoire des fenêtres ou des pages de présentation. Par définition, il existe un document par vue de l'application. Chaque document définit ensuite les opérations permettant aux pages ou fenêtres d'accéder aux informations nécessaires à l'affichage. [Rocques & Vallée, 2004].

➢ ***Le design pattern commande :***

Le *design pattern* Commande [Gamma 95] consiste à réifier un groupe d'opérations afin de pouvoir les traiter comme les ressources d'une application. On peut mémoriser de la sorte les dernières commandes effectuées par un utilisateur ou bien leur associer dynamiquement une séquence de touches clavier.

La structure du *design pattern* Commande inclut les éléments suivants :

- ✓ Une interface d'exécution générique (appelée ici « commande applicative », *CommandeApp* pour la différencier de l'objet métier *Commande*) ;
- ✓ Un récepteur correspondant à l'objet porteur de l'opération effectivement exécutée par la commande ;
- ✓ Un invocateur chargé de gérer et d'enchaîner l'exécution des commandes.

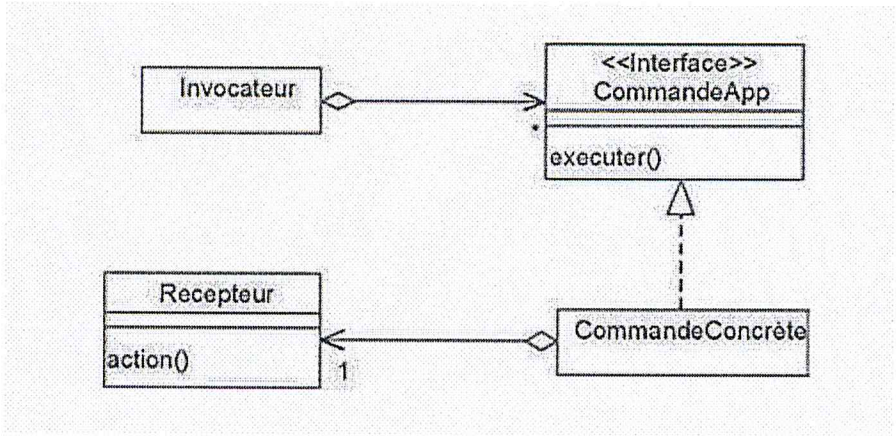


Figure 59 : Structure du design pattern Commande

➢ Conception du document d'édition de billet :

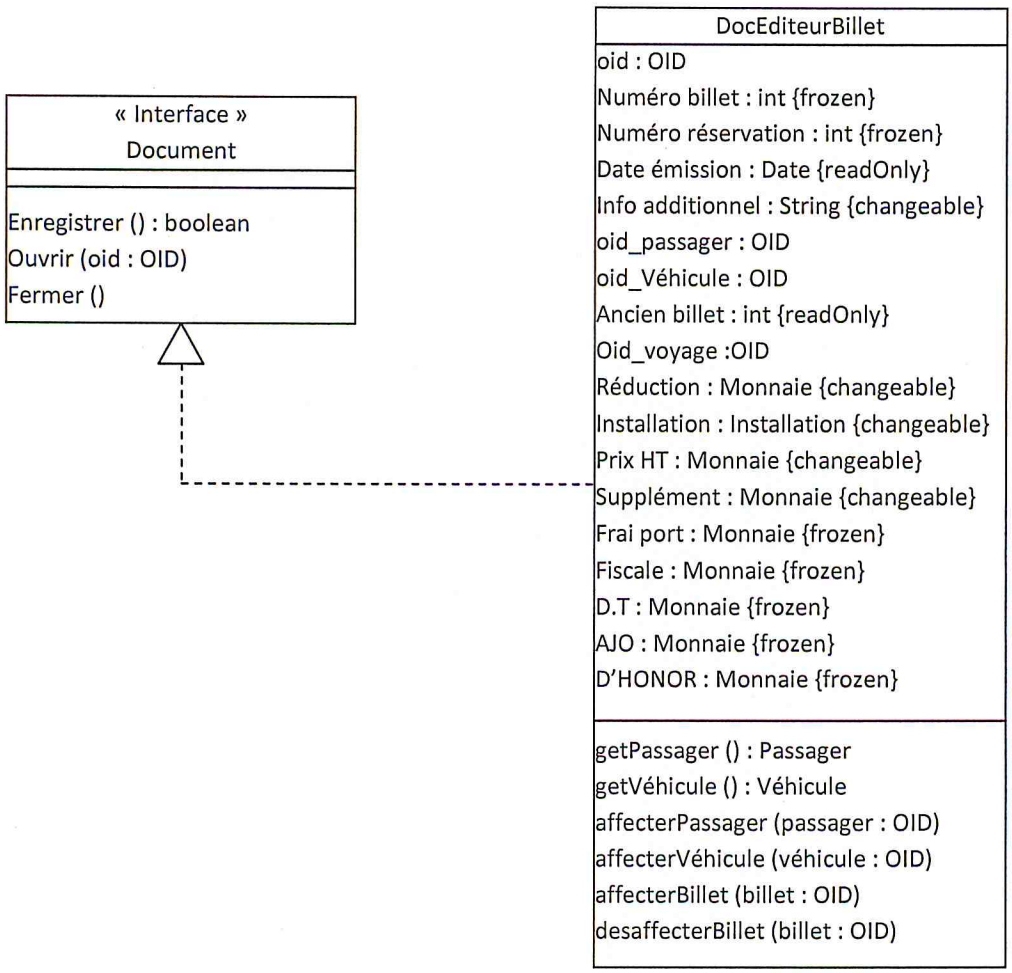


Figure 60 : Définition du document d'édition de billet

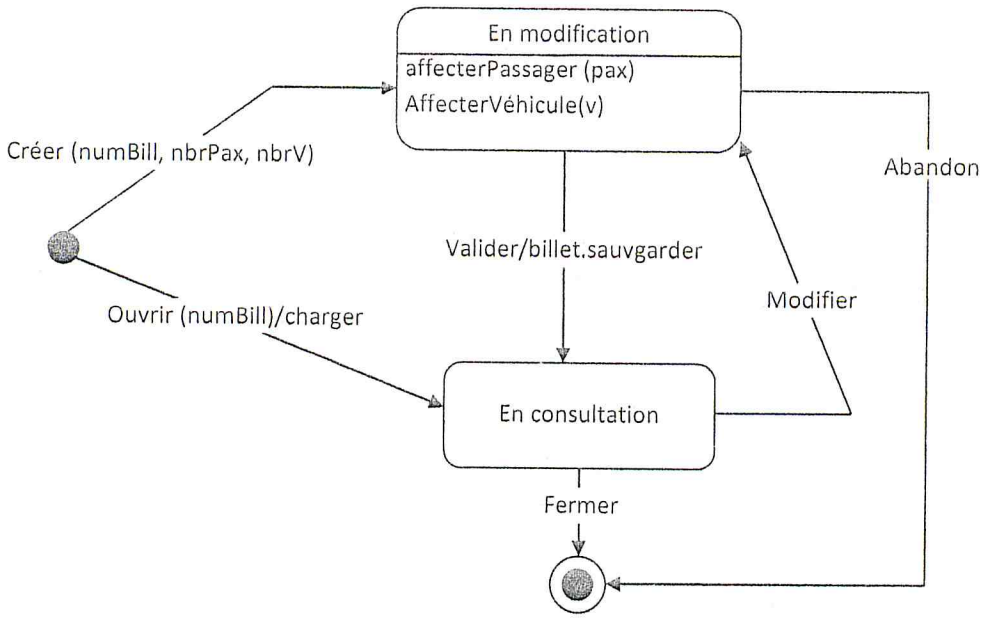


Figure 61 : Conception des états du document d'édition de billet

4.8 Conception du stockage des données :

La réalisation d'un stockage des instances varie suivant le mode de stockage retenu. Dans tous les cas, la réalisation d'un modèle objet facilite la maintenance des données stockées. Il existe aujourd'hui plusieurs modes de stockage possibles.

- Le système de fichiers est actuellement le moyen le plus rudimentaire de stockage. Avec le mécanisme de sérialisation, Java a fortement simplifié la technique de stockage d'objets dans des fichiers. Le stockage en fichiers ne coûte donc pratiquement rien. Cependant, il ne permet que de lire ou d'écrire une instance par des moyens externes à l'application et il n'a aucune capacité à administrer ou à établir des requêtes complexes sur les données.
- La base de données relationnelle ou SGBDR est un moyen déjà plus sophistiqué. Il existe aujourd'hui une large gamme de SGBDR répondant à des besoins de volume, de distribution et d'exploitation différents. Le SGBDR permet d'administrer les données et d'y accéder par des requêtes complexes. C'est la technique la plus répandue, que nous avons retenue pour SIAF.
- La base de données objet ou SGBDO constitue la méthode la plus élaborée de toutes. Cette technique élude la conception d'un stockage des données puisqu'elle permet de stocker et d'administrer directement des instances de classe. Cette technique n'a pourtant pas connu un grand succès sur le marché des bases de données.
- La base de données XML ou SGBDX est un concept émergeant qui répond au besoin croissant de stocker des documents XML sans risque d'altération de ces derniers. Dans le cadre de développement orienté objet qui nous occupe, cela signifierait une translation intermédiaire entre nos objets Java et un format XML.
  [Rocques & Vallée, 2004].

- **Passage du modèle objet au modèle relationnel :**

Modèle objet	Modèle relationnel
Classe	Table
Attribut de type simple	Colonne
Attribut de type composé	Colonnes ou clé étrangère
Instance	T-uplet
OID	Clé primaire
Association	Clé étrangère ou Table de liens
Héritage	Clé primaire identique sur plusieurs tables

Tableau 6 : Équivalences entre les concepts objets et relationnels

➢ Conception du stockage des classes Billet :

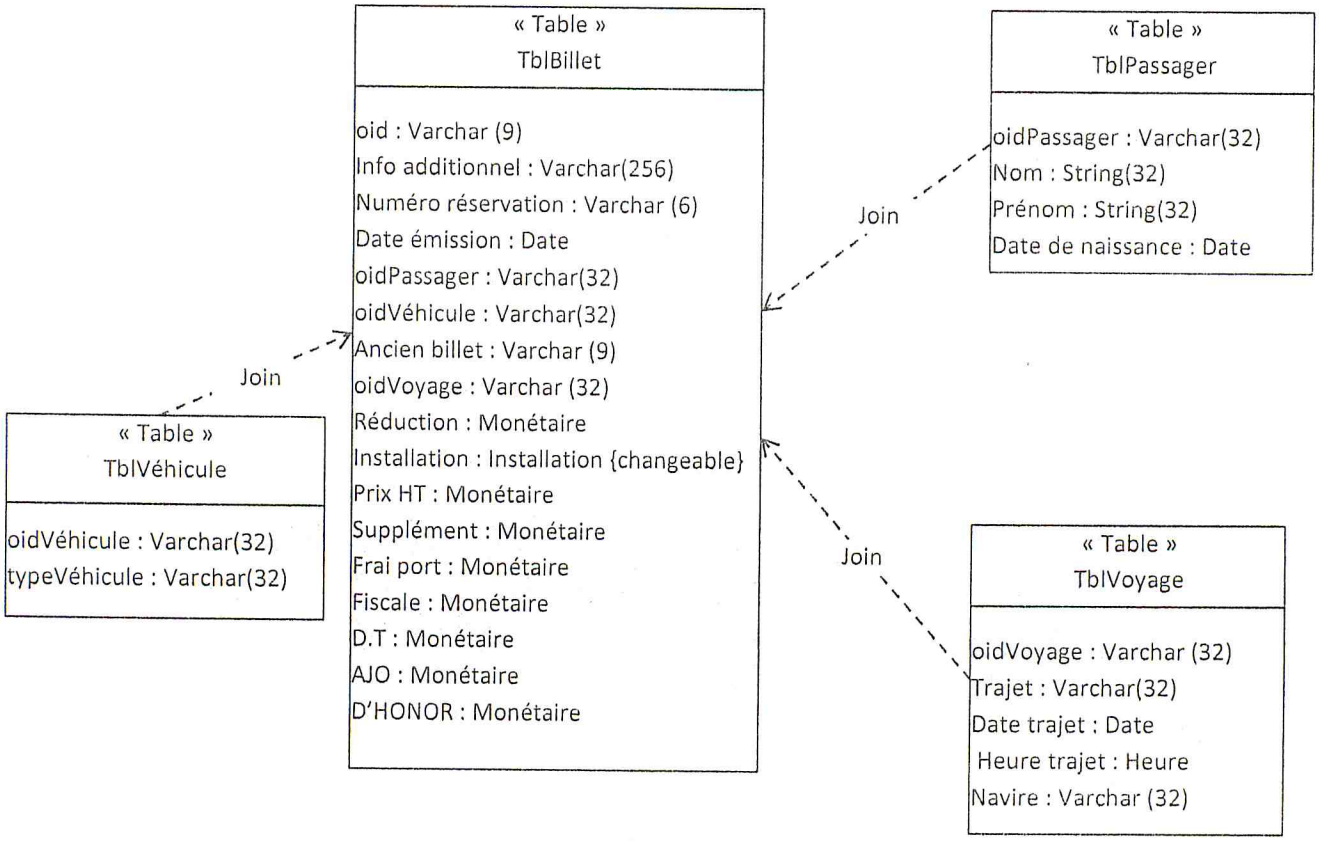
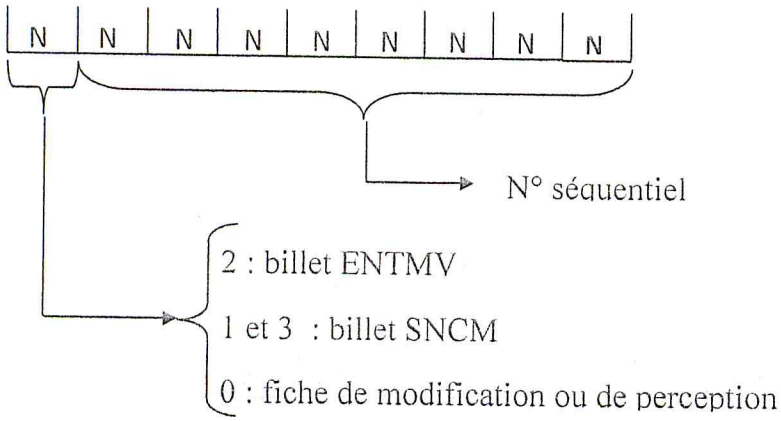


Figure 62 : Structure des tables relationnelles pour stocker les billets

4.9 Codification :

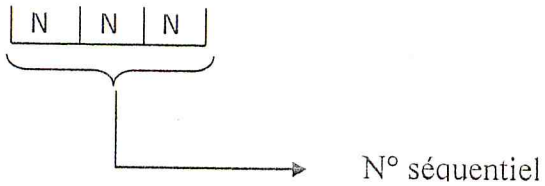
a. *Codification du billet :*

Chaque code de billet est composé de 09 positions numériques.



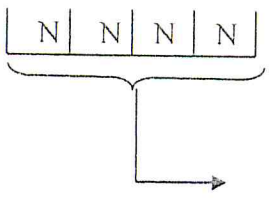
b. *Codification de l'EBV :*

Chaque code d'EBV est composé de 03positions numériques



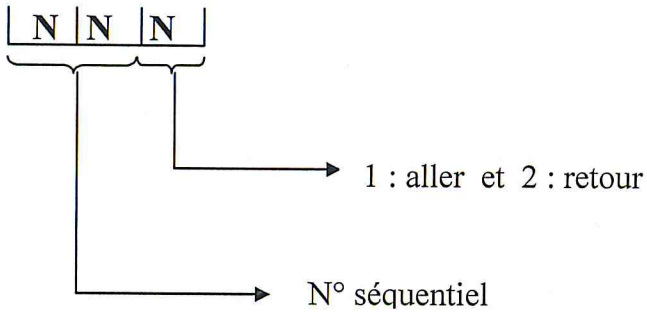
c. *Codification de l'agence :*

Chaque code d'agence est composé de 04 positions numériques



d. *Codification de voyage :*

Chaque code de voyage est composé de 03 positions numériques



CHAPITRE 4 : Réalisation

1. INTRODUCTION :

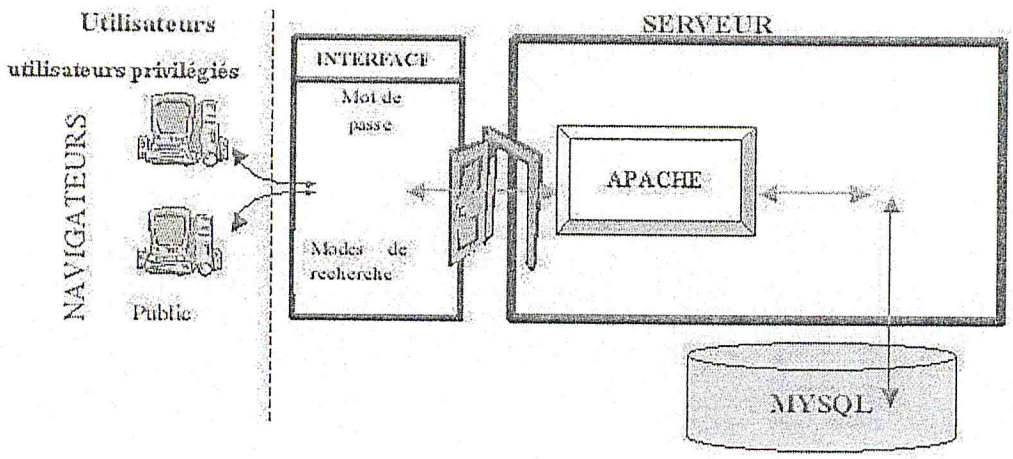
Après l'expression des besoins et la modélisation du système à développer, nous allons, dans cette partie, présenter son implémentation et sa mise en œuvre. Nous présenterons d'abord l'environnement de développement utilisé dans la réalisation du système, puis son architecture technique, les différentes interfaces utilisateur et les fonctionnalités de chacune.

2. REALISATION :

2.1 Architecture matérielle et technique du système :

Le choix s'est porté sur une architecture 3-tiers comportant d'un côté le serveur Web traitant les requêtes et où est stocké notre application, d'un serveur de base de données, et de postes de travail munis d'un browser permettant l'accès à notre application et donc la formulation des requêtes.

Architecture technique du système



Comme nous pouvons le constater à travers ce schéma, nous avons trois grands composants du système :

- Des navigateurs pour émettre des requêtes ;
- Des serveurs pour traiter les requêtes ;
- Un SGBD qui gère le répertoire et les métadonnées des documents.

2.2 Environnement de Développement :

2.2.1. Java/JEE

Java EE signifie **Java Enterprise Edition. J2EE (Java 2 Enterprise Edition)** était le terme précédent. J2EE désigne les technologies Java utilisées pour créer des applications d'entreprise avec le JDK 1.4 ou antérieur. En même temps que le JDK 1.5 amenait de nombreuses nouveautés dans le langage Java, Sun introduisait de nouvelles technologies s'appuyant sur ce langage amélioré afin de remédier à des lacunes de ces mêmes technologies dans J2EE. Le terme Java EE 5 ou Java EE a alors été utilisé pour désigner l'ensemble des technologies qui concourent à créer une application d'entreprise avec la plate-forme Java [développez, 2011]

2.2.2 JSP :

Les JSP (Java Server Pages) sont un standard permettant de développer des applications Web interactives, c'est-à-dire dont le contenu est dynamique. Il s'agit en réalité d'un langage de script puissant (un langage interprété) exécuté du côté du serveur (au même titre que les scripts PHP,ASP,...) et non du côté client (les scripts écrits en JavaScript ou les applets Java s'exécutent dans le navigateur de la personne connectée à un site).

Les JSP sont intégrables au sein d'une page Web en HTML à l'aide de balises spéciales permettant au serveur Web de savoir que le code compris à l'intérieur de ces balises doit être interprété afin de renvoyer du code HTML au navigateur du client.

Ainsi, les Java Server Pages s'inscrivent dans une architecture 3-tier, ce terme compliqué signifie qu'un serveur supportant les Java Server Pages peut servir d'intermédiaire (on parle généralement de serveur applicatif) entre le navigateur du client et une base de données (on parle généralement de serveur de données) en permettant un accès transparent à celle-ci. JSP fournit ainsi les éléments nécessaires à la connexion au système de gestion de bases de données, à la manipulation des données grâce au langage SQL. [besoindaide ,2011]

2.2.4. SGBD MYSQL:

MySQL est un système de gestion de base de données (SGBD). Selon le type d'application, sa licence est libre ou propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle et Microsoft SQL Server [Wikipédia ,2011].

Le serveur de gestion de base de données MySQL permet d'assurer :

- La définition et la manipulation des données,
- La cohérence des données,
- La sauvegarde et la restauration des données,
- La gestion des accès concurrents.

2.2.5. Serveur web Apache « Tomcat » :

Tomcat est un serveur d'applications Java. Il permet de générer une réponse HTML à une requête après avoir effectué un certain nombre d'opérations pour le client (un navigateur web en général), il reçoit toujours du HTML, seul langage qu'il comprend. Seule la manière dont la réponse est formée côté serveur change.

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 du Java . Il est paramétrable par des fichiers XML et de propriétés, et inclut des outils pour la configuration et la gestion. Il comporte également un serveur HTTP [Tomcat ,2011].

2.2.6. Java script:

JavaScript est un langage de script très populaire, utilisé notamment afin d'implémenter des traitements dans les pages Web. Tout au long de cet ouvrage, nous nous efforçons de détailler les différents concepts de ce langage ainsi que les mécanismes à mettre en œuvre afin de l'utiliser dans ce type de pages.

Le langage JavaScript permet de réaliser des traitements évolués dans les pages Web en offrant la possibilité d'utiliser les différents standards Web supportés par les navigateurs et d'interagir avec eux. Certaines spécificités de ces derniers restent néanmoins à prendre en compte afin de garantir leur portabilité

Longtemps freinée par le support hétérogène de ces standards, l'utilisation du langage JavaScript a été facilitée par l'apparition de bibliothèques intégrant la prise en compte de ces spécificités quant à la résolution de problématiques particulières. [Thierry& Arnaud,2005]

- **Le principal avantage** de Java Script réside dans la sécurité. En effet, les concepteurs ont interdits toutes les opérations pouvaient porter atteinte à la sécurité du visiteur. Le vol d'information, la destruction de fichier sont impossibles. Ce qui peut paraitre une faiblesse: il est impossible de créer un fichier, d'accéder à une base de données en Java Script est en fait un gros atout: PRATIQUEMENT TOUS LES VISITEURS ACCEPTENT LE JAVASCRIPT.

- **Le véritable inconvénient** de Java Script est sa compatibilité très limitée entre navigateur Internet Explorer et Netscape ont parfois des visions très différentes du Java Script. Cela oblige souvent à coder 2 Scripts pour la même action.

3. L'IMPLEMENTATION DE L'APPLICATION SIAF :

Apres avoir présenté les différents outils utilisés pour la réalisation de l'application SIAF, nous allons expliquer les différents modules de cette dernière.

3.1 Page d'authentification :

Afin d'éviter que n'importe qui accèdent aux informations , le système doit comporter des barrières qui assurent la confidentialité des information , seuls les utilisateurs identifiés par un nom d'utilisateur et un mot de passe auront accès aux informations

La page d'authentification du système SIAF est illustrée ci-dessous :

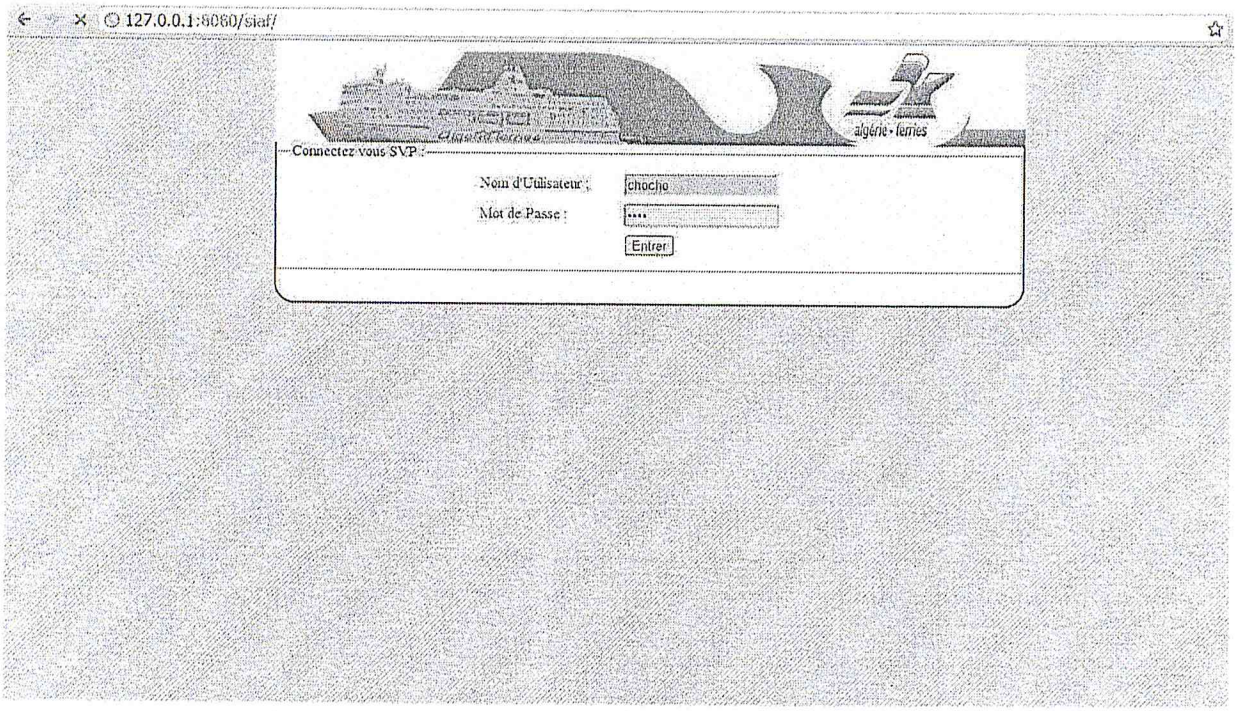


Figure 63 : interface d'authentification.

2.2. Page d'accueil :

La page principale du système SIAF est illustrée ci-dessous :

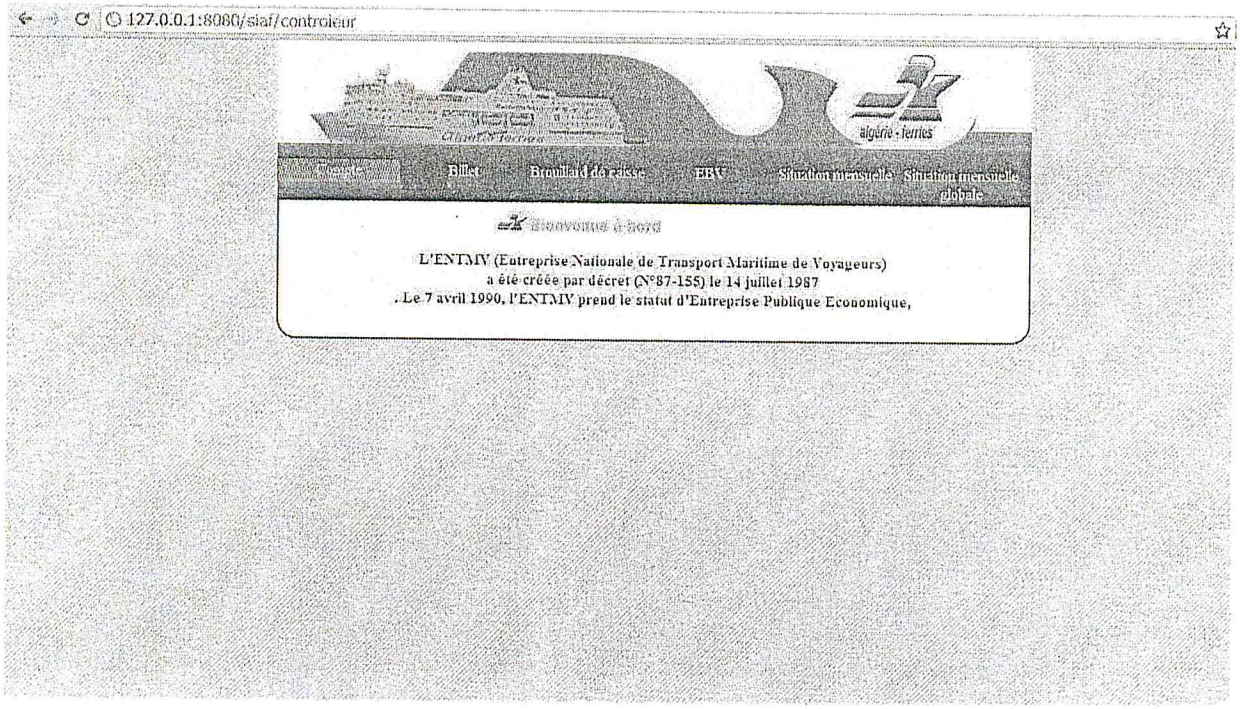


Figure 64 : page d'accueil.

A partir de cette page, l'utilisateur peut gérer toute l'application.

2.3. Page de l'ajout d'un nouveau billet :

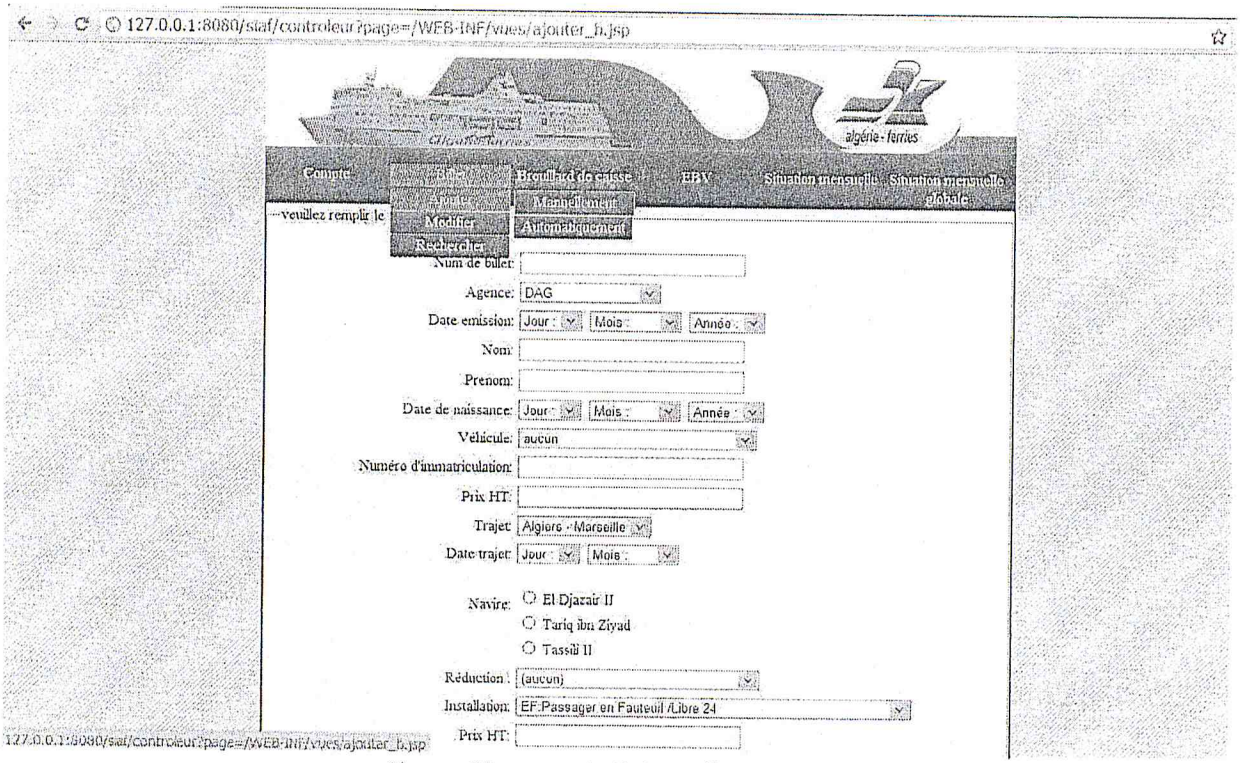


Figure 65 : page de l'ajout d'un nouveau billet.

Pour ajouter un billet il y a deux méthodes :

a. **Manuellement :** comme la figure 65, en remplissant tous les champs nécessaires puis en appuyant sur le bouton « entrer » pour envoyer les informations du billet vers la base de donnée
b. **Automatiquement :** pour l'ajout d'un billet automatiquement en sélectionnant un billet électronique sous forma txt par le bouton « Choose File » ensuite en valide par le bouton « Entrer » pour terminer l'opération de l'ajout.

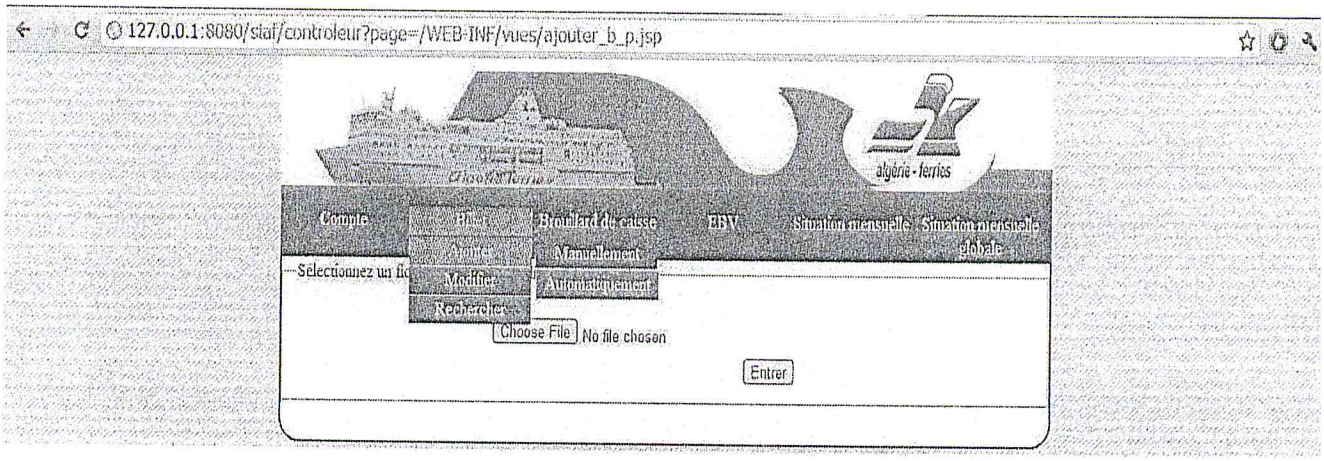


Figure 66 : page de l'ajout d'un nouveau billet automatiquement.

2.4 Page de la modification du billet :

Pour modifier ou supprimer un billet il suffit de donner le numéro du billet et puis valider par le bouton « Modifier »ou « supprimer ».comme montre la figure suivante :

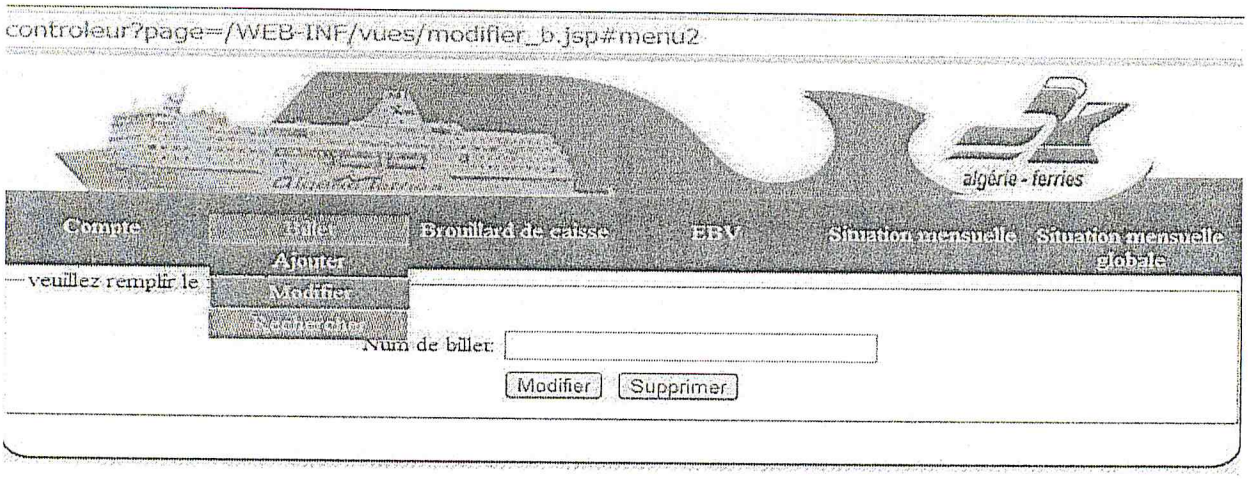


Figure 67 : page de modification d'un billet par le numéro.

Après avoir donné un numéro de billet et valider par le bouton « modifier », une page de modification avec les détails de l'ancien billet apparait comme montre la figure suivante :

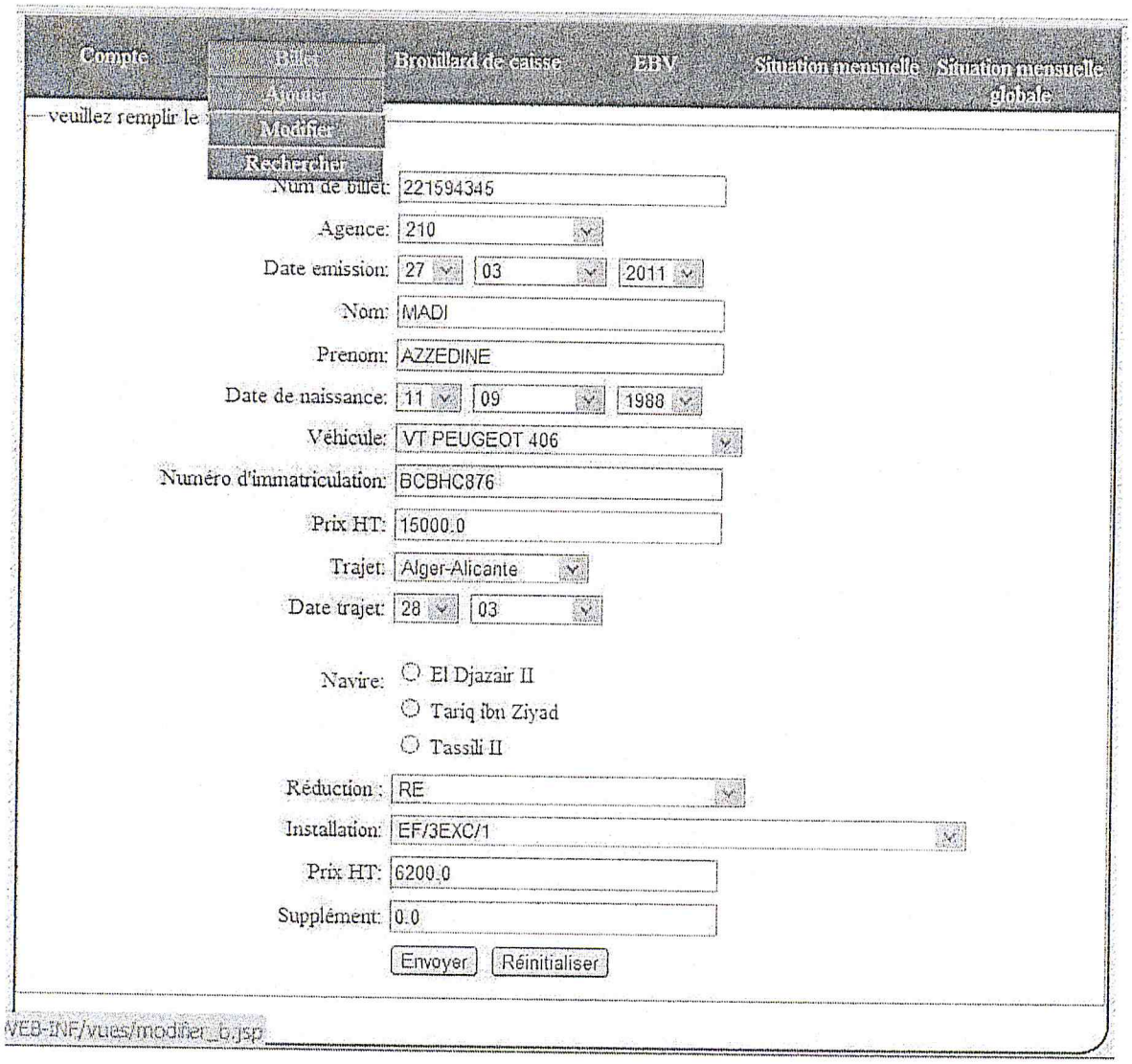


Figure 68 : page de modification d'un billet.

Pour la recherche des billets nous avons trois méthodes :

- Soit par le numéro de billet comme montre la figure 69 .

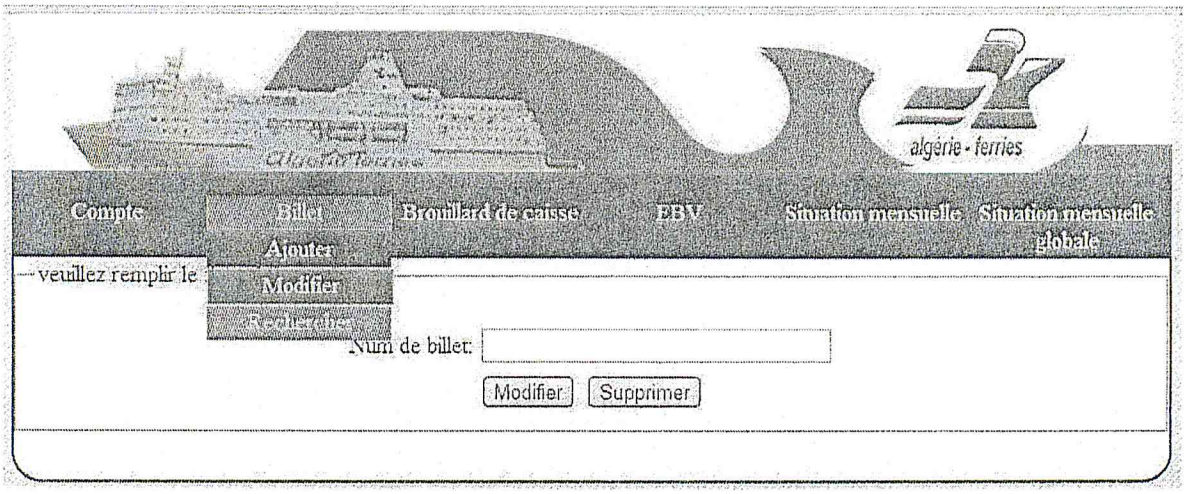


Figure 69 : page de recherche d'un billet par num.

- Soit par la date comme montre la figure 70.

Figure 70 : page de recherche d'un billet par la date.

- Soit par le code d'agence comme montre la figure 71.

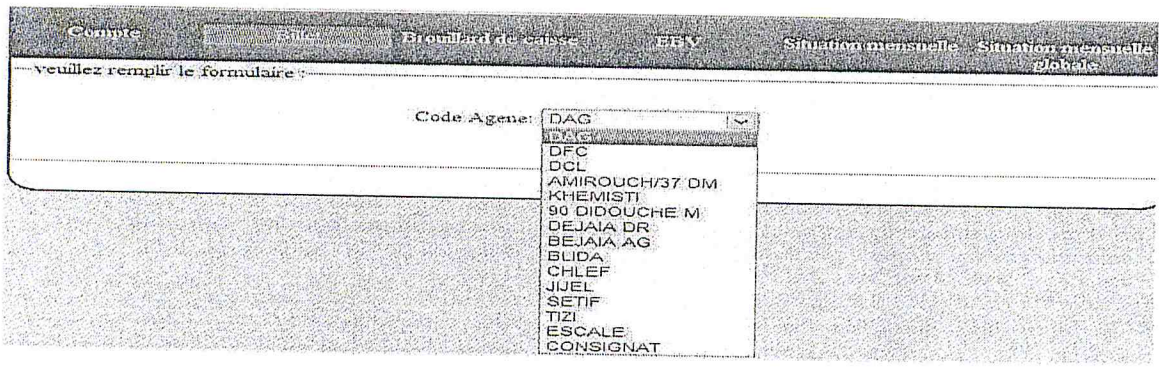


Figure 71 : page de recherche d'un billet par le code d'agence.

2.5 Page de l'affichage de brouillard de caisse :

Pour l'affichage d'un brouillard de caisse on choisit une date et puis on valider.

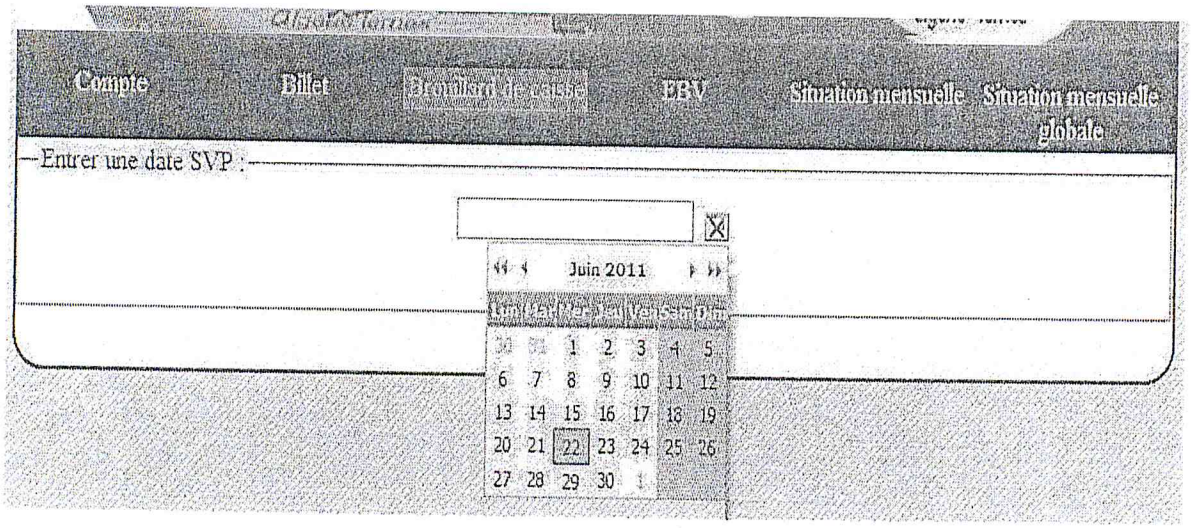


Figure 72 : afficher un bouillard de caisse.

Le résultat sera comme suit :

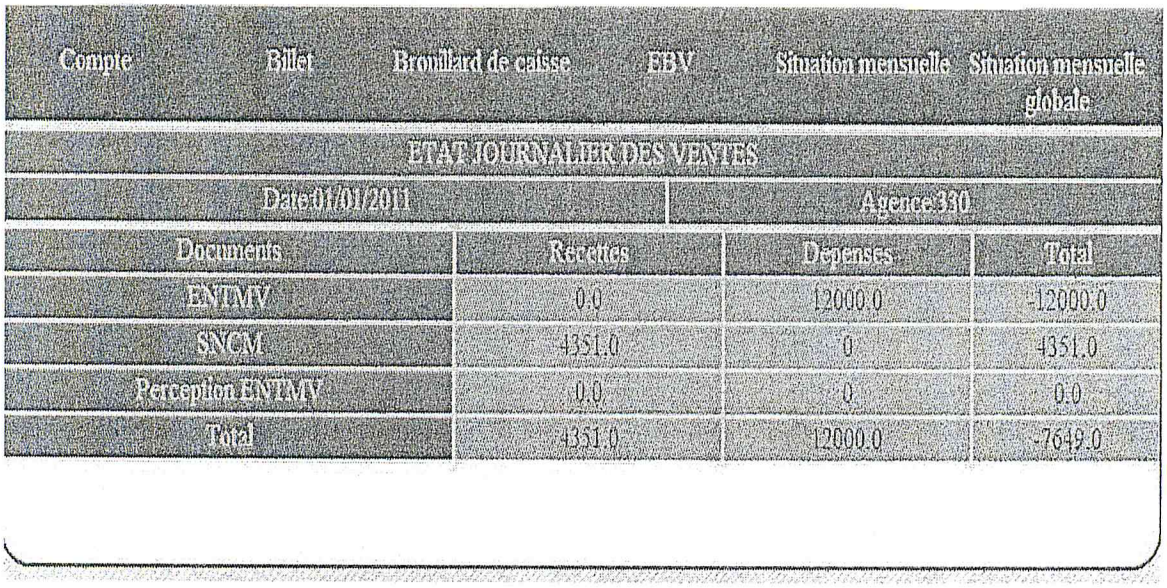


Figure 73 : l'affichage du bouillard de caisse.

2.6 Page de l'affichage d'EBV:

Pour l'affichage d'EBV en sélectionnant la quinzaine puis en appuyant sur le bouton envoyer Comme montre la figure74.

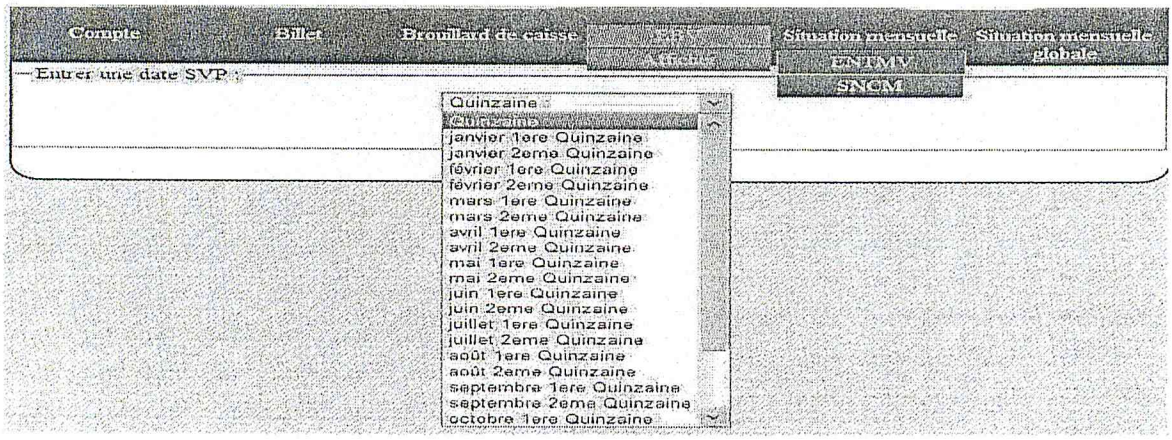
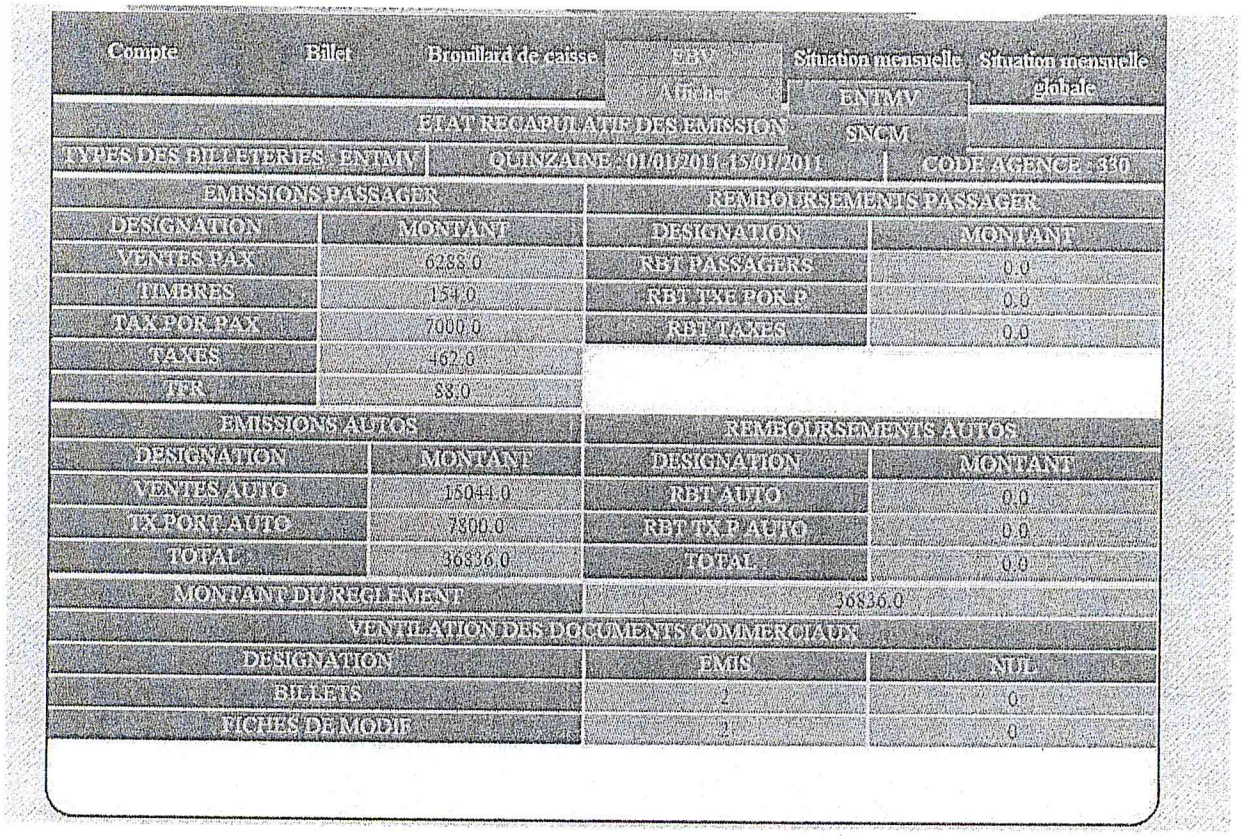


Figure 74 : afficher un EBV.

L'affichage d'EBV est comme montre la figure suivante :

Pour les billets ENTMV :



Compte | Billet | Brouillard de caisse | EBV | Situation mensuelle | Situation mensuelle globale

Afficher | ENTMV | SNCM

ETAT RECAPULATIF DES EMISSION

TYPES DES BILLETERIES : ENTMV | QUINZAINE : 01/01/2011-15/01/2011 | CODE AGENCE : 330

EMISSIONS PASSAGER		REMBOURSEMENTS PASSAGER	
DESIGNATION	MONTANT	DESIGNATION	MONTANT
VENTES PAX	6288.0	RBT PASSAGERS	0.0
TIMBRES	154.0	RBT TXE POR.P	0.0
TAX POR.PAX	7000.0	RBT TAXES	0.0
TAXES	462.0		
TFR	88.0		

EMISSIONS AUTOS		REMBOURSEMENTS AUTOS	
DESIGNATION	MONTANT	DESIGNATION	MONTANT
VENTES AUTO	16044.0	RBT AUTO	0.0
TX PORT AUTO	7800.0	RBT TX P AUTO	0.0
TOTAL	36836.0	TOTAL	0.0

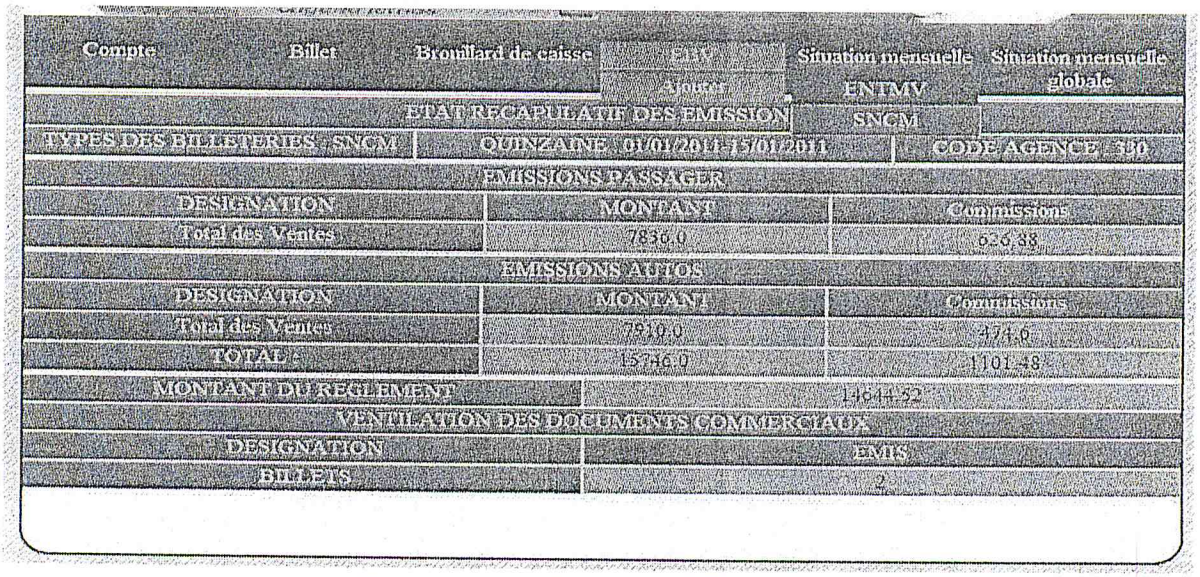
MONTANT DU REGLEMENT : 36836.0

VENTILATION DES DOCUMENTS COMMERCIAUX

DESIGNATION	EMIS	NUL
BILLETS	2	0
FICHES DE MODIF	2	0

Figure 75: l'affichage d'EBV ENTMV

Pour les billets SNCM :



Compte | Billet | Brouillard de caisse | EBV | Situation mensuelle | Situation mensuelle globale

Afficher | ENTMV | SNCM

ETAT RECAPULATIF DES EMISSION

TYPES DES BILLETERIES : SNCM | QUINZAINE : 01/01/2011-15/01/2011 | CODE AGENCE : 330

EMISSIONS PASSAGER

DESIGNATION	MONTANT	Commissions
Total des Ventes	7836.0	626.88

EMISSIONS AUTOS

DESIGNATION	MONTANT	Commissions
Total des Ventes	7910.0	474.6
TOTAL	15746.0	1101.48

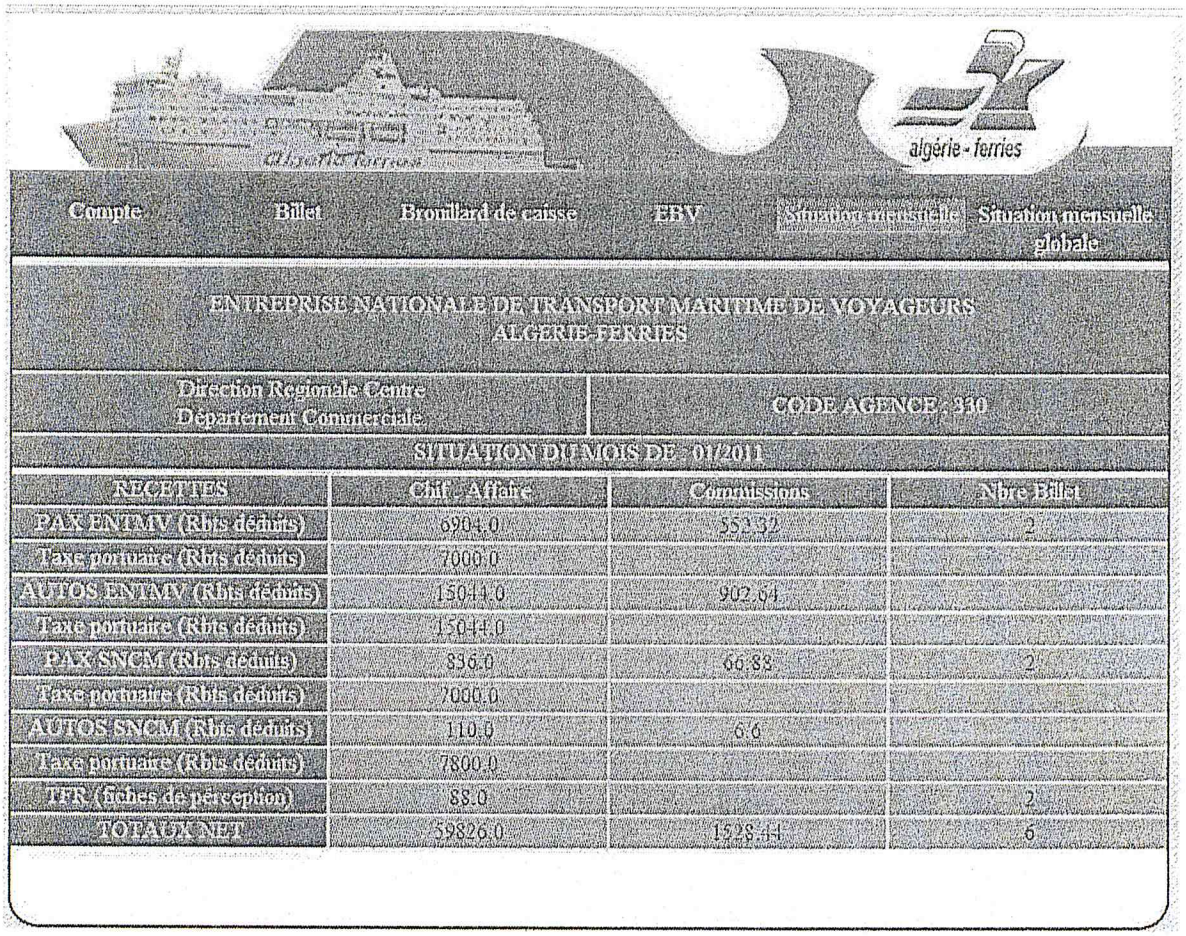
MONTANT DU REGLEMENT : 14644.52

VENTILATION DES DOCUMENTS COMMERCIAUX

DESIGNATION	EMIS
BILLETS	2

Figure 76: l'affichage d'EBV SNCM.

1.7 Page de l'affichage de la Situation mensuelle :



Compte | Billet | Bromillard de caisse | EBV | Situation mensuelle | Situation mensuelle globale

ENTREPRISE NATIONALE DE TRANSPORT MARITIME DE VOYAGEURS
ALGERIE FERRIES

Direction Régionale Centre
Département Commerciale

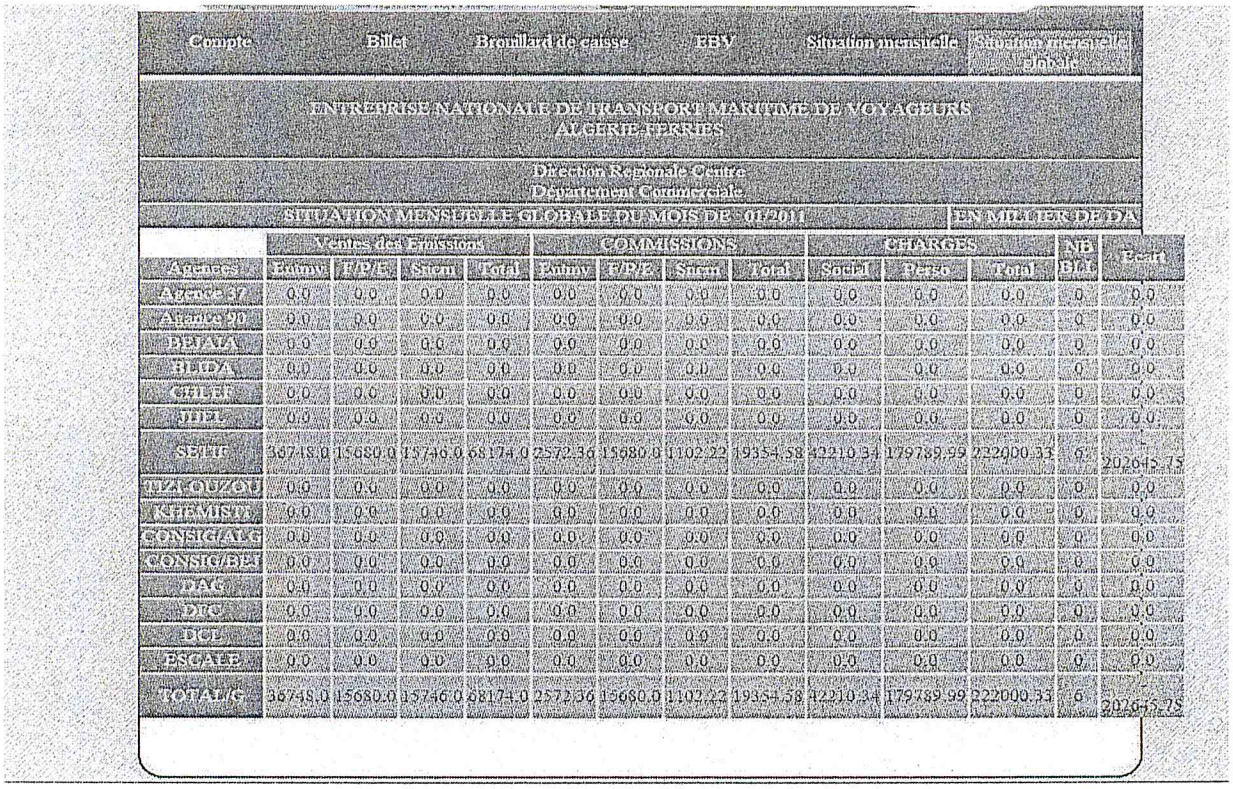
CODE AGENCE : 330

SITUATION DU MOIS DE : 01/2011

RECETTES	Chif . Affaire	Commissions	Nbre Billet
PAX ENTMV (Rbts déduits)	6904.0	552.32	2
Taxe portuaire (Rbts déduits)	7000.0		
AUTOS ENTMV (Rbts déduits)	15044.0	902.64	
Taxe portuaire (Rbts déduits)	15044.0		
PAX SNCM (Rbts déduits)	836.0	66.88	2
Taxe portuaire (Rbts déduits)	7000.0		
AUTOS SNCM (Rbts déduits)	110.0	6.6	
Taxe portuaire (Rbts déduits)	7800.0		
TFR (fiches de perception)	88.0		2
TOTAUX NET	59826.0	1528.44	6

Figure 77 : l'affichage de la situation mensuelle.

2.7 Page de l'affichage de la Situation mensuelle globale :



Compte | Billet | Brouillard de caisse | EBV | Situation mensuelle | Situation mensuelle globale

ENTREPRISE NATIONALE DE TRANSPORT MARITIME DE VOYAGEURS
ALGERIE-FERRIES

Direction Régionale Centre
Département Commerciale

SITUATION MENSUELLE GLOBALE DU MOIS DE 01-2011 — EN MILLIER DE DA

	Ventes des Emissions				COMMISSIONS				CHARGES			NB BLE	Ecart
Agences	Espèce	T/P/E	Suros	Total	Espèce	T/P/E	Suros	Total	Sociétés	Perso	Total		
Agence 17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
Agence 20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
BEJAIA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
BLIDA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
CHLEF	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
DJEL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
SETIF	36748.0	15680.0	15746.0	68174.0	2572.36	15680.0	1102.22	19354.58	42210.34	179789.99	222000.33	6	2026415.75
TIZI OUZOU	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
KHEMISTI	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
CONSIG/ALG	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
CONSIG/BEJ	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
DAC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
DTC	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
DCL	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
ESCALE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0.0
TOTAL/G	36748.0	15680.0	15746.0	68174.0	2572.36	15680.0	1102.22	19354.58	42210.34	179789.99	222000.33	6	2026415.75

Figure 78 : l'affichage de la situation mensuelle globale.

5. CONCLUSION :

Enfin, nous avons présenté la vue réelle de notre application, et des principaux modules qui la font fonctionner en toute simplicité, rapidité, grâce à une interface intuitive, simple respectant les concepts fondamentaux de l'IHM (interface homme machine).

CONCLUSION GENERALE :

Nous avons tenté à travers ce projet de démontrer l'importance de l'application d'une méthode de développement. Nous pensons aussi que 2TUP pourra être utilisée dans des projets de moyenne à grande envergure. A titre personnel, le bénéfice qu'on en a tiré est l'apprentissage de concepts à la pointe de la technologie et des tendances actuelles dans le monde professionnel. Une recherche profonde a été indispensable pour essayer de comprendre ces concepts-là. Nous pouvons citer à ce propos, un excellent livre traitant ce sujet qui s'appelle : ***UML 2 En Action.***

Ce projet nous a permis d'enrichir nos connaissances dans des domaines très variés comme : *L'Orienté Objet*, *UML*, *2TUP*, *le langage JAVA*, *les Design Patterns*...

En termes d'évolution, l'application pourra par la suite être adaptée à une utilisation à l'université. Par exemple, une base de données pourra être utilisée soit par le biais d'un pont **JDBC**, ou par le biais d'une solution de *Mapping Objet/ Relationnel* comme **Hibernate**. Aussi, un déploiement sur un réseau pourra être fait grâce au framework **J2EE**.

BIBLIOGRAPHIE :

[Rocques & Vallée, 2004] ⟶ Rocques, P., & Vallée, F. (2004). *UML 2 En Action (De l'analyse des besoins à la conception J2EE).* Eyrolles.

[Chromatic, 2005] ⟶ Chromatic. (2005). *Extreme programming.* O'Reilly.

[Meyer, 2000] ⟶ Meyer, B. (2000). *Conception et Programmation orientées objet.* Eyrolles.

[Pitman, 2006] ⟶ Pitman, N. (2006). *UML 2 en concentré.* O'Reilly.

[Roques, 2006] ⟶ Roques, P. (2006). *UML 2 en pratique.* Eyrolles.

[Développez, 2011] ⟶ juin 2011/ http://tahe.developpez.com/java/javaee/.

[Wikipédia ,2011] ⟶ juin-2011/ http://fr.wikipedia.org/wiki/MySQL .

[Gamma 95] ⟶ Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides, les quatre auteurs de [Gamma 95], souvent référencés en tant que GoF (Gang of Four).

[besoindaide ,2011] ⟶ juin-2011/ http://www.besoindaide.com/ccm/jsp/jspintro.htm .

[Tomcat ,2011] ⟶ http://www-igm.univ-mlv.fr/~dr/XPOSE2003/tomcat/tomcat.php?rub=25.

[Thierry& Arnaud,2005] ⟶ JavaScript pour le Web 2.0 Programmation objet, DOM, Ajax, Prototype, Dojo, Script.aculo.us, Rialto, réalisé par Thierry Templier et Arnaud Gougeon.

Annexes

1. L'APPROCHE ORIENTEE OBJET :

Notion de classe :

Tout d'abord, introduisons la notion de classe. Une classe est un type de données abstrait, caractérisé.

Par des propriétés (attributs et méthodes) communes à toute une famille d'objets et permettant de créer (instancier) des objets possédant ces propriétés. Les autres concepts importants qu'il nous faut maintenant introduire sont l'encapsulation, l'héritage et l'agrégation.

Encapsulation

L'encapsulation consiste à masquer les détails d'implémentation d'un objet, en définissant une interface. L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet.

L'encapsulation facilite l'évolution d'une application car elle stabilise l'utilisation des objets: on peut modifier l'implémentation des attributs d'un objet sans modifier son interface, et donc la façon dont L'objet est utilisé.

L'encapsulation garantit l'intégrité des données, car elle permet d'interdire, ou de restreindre, l'accès direct aux attributs des objets.

Héritage, Spécialisation, Généralisation et polymorphisme :

L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe. Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines. Plusieurs classes peuvent être généralisées en une classe qui les factorise, afin de regrouper les caractéristiques communes d'un ensemble de classes.

Ainsi, la spécialisation et la généralisation permettent de construire des hiérarchies de classes. L'héritage peut être simple ou multiple. L'héritage évite la duplication et encourage la réutilisation.

Le polymorphisme représente la faculté d'une méthode à pouvoir s'appliquer à des objets de classes différentes. Le polymorphisme augmente la généricité, et donc la qualité, du code.

L'Agrégation :

Il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe. Une relation d'agrégation permet donc de définir des objets composés d'autres objets. L'agrégation permet donc d'assembler des objets de base, afin de construire des objets plus complexes. (Meyer, 2000)

2. UML :

Introduction :

La description de la programmation par objets a fait ressortir l'étendue du travail conceptuel nécessaire: définition des classes, de leurs relations, des attributs et méthodes, des interfaces etc.

Pour programmer une application, il ne convient pas de se lancer tête baissée dans l'écriture du code : Il faut d'abord organiser ses idées, les documenter, puis organiser la réalisation en définissant les modules et étapes de la réalisation. C'est cette démarche antérieure à l'écriture que l'on appelle modélisation; son produit est un modèle.

Les spécifications fournies par la maîtrise d'ouvrage en programmation impérative étaient souvent floues: les articulations conceptuelles (structures de données, algorithmes de traitement) s'exprimant dans le vocabulaire de l'informatique, le modèle devait souvent être élaboré par celle-ci.

L'approche objet permet en principe à la maîtrise d'ouvrage des exprimer de façon précise selon un vocabulaire qui, tout en transcrivant les besoins du métier, pourra être immédiatement compris par les informaticiens. En principe seulement, car la modélisation demande aux maîtrises d'ouvrage une compétence, un professionnalisme qui ne sont pas aujourd'hui répandus.

UML en œuvre :

UML n'est pas une méthode (i.e. une description normative des étapes de la modélisation): ses auteurs ont en effet estimé qu'il n'était pas opportun de définir une méthode en raison de la diversité des cas particuliers. Ils ont préféré se borner à définir un langage graphique qui permet de représenter, de communiquer les divers aspects d'un système d'information (aux graphiques sont bien sûr associés des textes qui expliquent leur contenu). UML est donc un méta-langage car il fournit les éléments permettant de construire le modèle qui, lui, sera le langage du projet.

Il est impossible de donner une représentation graphique complète d'un logiciel, ou de tout autre système complexe, de même qu'il est impossible de représenter entièrement une statue (à trois dimensions) par des photographies (à deux dimensions). Mais il est possible de donner sur un tel système des vues partielles, analogues chacune à une photographie d'une statue, et dont la juxtaposition donnera une idée utilisable en pratique sans risque d'erreur grave.

Les diagrammes d'UML :

UML 2.0 comporte ainsi treize types de diagrammes représentant autant de vues distinctes pour représenter des concepts particuliers du système d'information. Ils se répartissent en deux grands groupes:

Diagrammes structurels ou diagrammes statiques (UML Structure) :

- diagramme de classes (Class diagram)
- diagramme d'objets (Object diagram)
- diagramme de composants (Component diagram)
- diagramme de déploiement (Deployment diagram)
- diagramme de paquetages (Package diagram)
- diagramme de structures composites (Composite structure diagram)

Diagrammes comportementaux ou diagrammes dynamiques (UML Behavior)

- diagramme de cas d'utilisation (Use case diagram)
- diagramme d'activités (Activity diagram)
- diagramme d'états-transitions (State machine diagram)
- diagrammes d'interaction(Interactiondiagram)
- diagramme de séquence (Sequence diagram)
- diagramme de communication (Communication diagram)
- diagramme global d'interaction (Interaction overview diagram)
- diagramme de temps (Timing diagram)

Ces diagrammes, d'une utilité variable selon les cas, ne sont pas nécessairement tous produits à l'occasion d'une modélisation. Les plus utiles pour la maîtrise d'ouvrage sont les diagrammes d'activités, de cas d'utilisation, de classes, d'objets, de séquence et d'états-transitions. Les diagrammes de composants, de déploiement et de communication sont surtout utiles pour la maîtrise d'œuvre à qui ils permettent de formaliser les contraintes de la réalisation et la solution technique.

3. LE LANGAGE JAVA :

Java est à la fois un langage de programmation et un environnement d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation. C'est la plateforme qui garantit la portabilité des applications développées en Java. (Cornell)

Lors de la création du langage Java, il avait été décidé que ce langage devait répondre à 5 objectifs :

1. utiliser une méthode orientée objet,
2. permettre à un même programme d'être exécuté sur plusieurs systèmes d'exploitation différents,
3. pouvoir utiliser de manière native les réseaux informatiques,
4. pouvoir exécuter du code distant de manière sûre,
5. être facile à utiliser et posséder les points forts des langages de programmation orientés objet comme le C++.

Le système Java est basé sur le *langage Java*, *la machine virtuelle java*, et *l'API JAVA* (ces deux derniers composants forment l'environnement d'exécution, ou JRE, pour *Java Runtime Environment*).

4. DI FFERENCE ENTRE L'ARCHITECTURE 3-TIERS ET LE MODELE MVC :

Les noms d'architectures MVC et 3-Tiers sont très couramment utilisés dans les cours de génie logiciel. Il est facile de s'emmêler les pinceaux car ces deux pratiques sont à la fois différentes et similaires.

L'architecture MVC :

Model View Controller (Modèle Vue Contrôleur) est souvent décrit comme un simple design pattern (motif de conception) mais c'est plus un architectural pattern (motif d'architecture) qui donne le ton à la forme générale d'une solution logiciel plutôt qu'à une partie restreinte.

Les trois parties du pattern MVC sont les suivantes :

1. ***Model*** : Le modèle défini les données de l'application et les méthodes d'accès. Tous les traitements sont effectués dans cette couche.
2. ***View*** : La vue prend les informations en provenance du modèle et les présente à l'utilisateur.
3. ***Controller*** : Le contrôleur répond aux événements de l'utilisateur et commande les actions sur le modèle. Cela peut entrainer une mise à jour de la vue.

L'architecture 3-Tiers :

L'architecture 3-Tiers sépare, tout comme MVC, l'application en trois parties bien distinctes :

1. ***User interface*** : La partie présentation de l'application.
2. ***Buisness logic*** : La couche métier qui s'occupe du traitement de l'information.
3. ***Data access*** : La partie accès et stockage des données

Architecture MVC ou 3-Tiers ?

La différence fondamentale se trouve dans le fait que l'architecture 3-Tiers sépare la couche *Buisness logic* (couche métier) de la couche *Data access* (accès aux données).

Pour qu'une application MVC soit une vraie application 3-Tiers il faut lui ajouter une couche d'abstraction d'accès aux données de type DAO (*Data Access Object*).

Inversement pour qu'une application 3-Tiers respecte MVC il faut lui ajouter une couche de contrôle entre *User interface* et *Buisness logic*.

Loin d'être antagonistes, ces deux pratiques se combinent et sont la fondation de la plupart des frameworks de création d'applications Web.