REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'enseignement supérieur et de la recherche scientifique

Université **M**ouloud **M**ammeri **T**izi-**O**uzou

Faculté de Génie Electrique et d'Informatique

Département d'Informatique

MEMOIRE

En vue de l'obtention du diplôme
de Master LMD en Informatique
Ingénierie des Systèmes d'Informations

Thème

Conception et réalisation d'un système de facturation dédie aux ERP/CRM

Dirigé par :

Mr M.SI MOHAMED.

Réalisé par :

Mr MANI Yacine
Melle SIACI Lydia

Juillet 2015

Remerciements

Nous tenons à remercier vivement notre promoteur Mr M.SI MOHAMED et les Co-promoteurs Mr N.SI REMDANE et M^{lle} R.NESNAS de nous avoir proposé ce sujet, pour la qualité de leur encadrement, et leurs suivis durant toute la durée du projet.

Nous remercions chaleureusement les membres du jury pour l'honneur qu'ils nous font en acceptant de juger ce modeste travail.

Enfin, nous remercions tous l'effectif de MARINE SOFT ainsi que M^{lle} O.NENAS pour leurs contributions au bon accomplissement de notre travail en particulier

Dédicaces

A mes très chers parents

Que ce modeste travail, soit l'exaucement de vos veux tant formulés et de Vos prières quotidiennes,

A mes très chers frères et sœurs;

A mes nièces et neveu ALYCIA, ALICIA et MASSY
A tous mes amis
A NOUNOU, DJEF, HAKIM, SOFIANE, MARZOUK et HAKIM

Qui m'ont soutenu durant cette dure année, Je vous remercie.

**********YACINE

A mes très chers parents

Que ce modeste travail, soit l'exaucement de vos veux tant formulés et de Vos prières quotidiennes,

A mes très chers frères et sœurs;

A mon oncle et sa petite famille ;

A mes amis

Qui m'ont soutenue durant cette dure année, Je vous remercie.

**********LYDIA

Résumé

Dans le cadre de stage de fin d'étude en vue de l'obtention du diplôme de MASTER II en informatique option « Ingénierie des Systèmes d'Information », la Sarl MARINE SOFT, société de service et d'ingénierie en informatique, nous a proposé un projet visant à concevoir et réaliser un système d'information pour la gestion de facturation standard. Ce système est dédié à l'ERP et au CRM de l'entreprise et des clients potentiels.

Il s'agit d'une application 3 tiers de facturation des transactions commerciales, qui traite la facture pro forma, initiale, avoir/complémentaire, le règlement et le suivi règlement, ainsi que la gestion des avances avec plusieurs modes de versement pris en charge par une fonctionnalité nommée 'Compte client' afin de facilité les tâches de payements aux clients et fidéliser ces derniers par la création des comptes.

Le présent projet se résume, à concevoir et réaliser un système qui permettra de répondre aux besoins de la gestion de facturation. Celui-ci devra, entre autre, être conçu sous les normes des ERP/CRM concernant la standardisation des traitements, la centralisation des données et la gestion des ventes et prestation de service.

Mots clés :

Marine soft, ISI, 3tiers, Approche par processus, ERP, CRM, Procédé de facturation, Facture standard, STD_fct, UML, JEE, Client/serveur, Java.

SOMMAIRE

Chapitre 3 : Conception du futur système

SOMMAIRE

Chapitre 4 : Réalisation

LISTE DES FIGURES

Chapitre 1

Chapitre 2

LISTE DES FIGURES

Chapitre 3

Chapitre 4

LISTE DES TABLEAUX

Chapitre 2

Chapitre 3

Introduction générale

La dématérialisation de factures consiste à faire passer des documents du support physique (le plus souvent papier) au support numérique/électronique. Depuis quelques années, suite aux progrès technologiques et aux modifications législatives, les factures sont dématérialisables tout en gardant leur valeur fiscale.

Ainsi que la fidélisation des clients est devenu un concept très sollicité par les entreprises du 20eme cycle qui a donné naissance aux CRM comme une stratégie de travail, une base des relations fournisseurs-clients mais aussi une démarche qui englobe le processus de vente « le marketing- la vente- la comptabilité ».

Comme toute entreprise se voulant concurrente sur le marché national voir international, MARINE SOFT «société de service et d'ingénierie en informatique» entièrement spécialisée dans la conception, la réalisation, la mise en route et l'assistance technique de divers logiciels relatifs à des activités liées au transport maritime, doit saisir les opportunités pour s'adapter aux nouveaux marchés et aux progrès technologiques pour refaçonner ses produits, réduire ses coûts, et améliorer la qualité des services qu'elle offre à ses clients.

Pour l'obtention du diplôme « Master II en informatique (Ingénierie des systèmes d'information) », un projet de fin d'étude nous a était proposé par la direction technique de Marine soft, intitulé « **conception et réalisation d'un système de facturation dédie aux ERP/CRM** » qui traite un processus important du métier de la Facturation.

Ce processus doit prendre en charge les fonctions suivantes :

- Rubriques de facturations (famille, groupe) ;
- Factures initiale, avoir, complémentaire;
- Convention de payement ;
- Facturation à crédit ;
- Règlement des factures (par tranches, avec différent mode de payement, à crédit) ;
- Gestion des versements de clients;
- Un outil de recherche multicritères des factures (type de facture, entre deux dates, numéro de facture…etc.) ;
- Journal de caisse entre deux dates ;

.

Aujourd'hui, les données sont au cœur des entreprises. Qui estime qu'elles doivent être dissociées de la logique métier. Comme nous adhérons à ce paradigme et pour la bonne organisation de notre travail, nous avons adopté la structure chapitrale suivante :

Les technologies ERP / CRM => nous définissons dans cette phase quelques concepts tels que ERP, CRM qui sont considérés dans le cadre de notre projet comme des connaissances nécessaire pour la mise en place de ce dernier.

Etude de l'existant => les objectifs de cette phase sont de faire un récapitulatif de notre environnement de travail, de définir le besoin de notre organisme d'accueil et d'aboutir à travers une étude comparative de ces système existants au choix d'une solution informatique la plus adaptée.

Analyse et Conception => dans cette phase nous mettrons en œuvre l'outil imposé par l'entreprise "UML" qui nous permettra de couvrir, via un vocabulaire commun, tous les aspects de l'analyse et de la conception d'un logiciel, en favorisant une démarche souple fondée sur les interactions entre les différentes vues que l'on peut avoir d'une application.

Réalisation et Implémentation =>l'objectif de cette phase est de construire le système STD_FCT en utilisant les outils nécessaire à sa réalisation et de présenter les résultats dans en effectuant des captures d'écran.

Annexes => l'objectif cette phase est d'introduire ces concepts JAVA EE, UML, et l'architecture client/serveur que nous avons appliquée au fur et à mesure du mémoire.

CHAPITRE 1

[Technologie ERP/CRM]

Introduction

Actuellement, dans un marché de plus en plus concurrentiel, l'accès des entreprises aux nouvelles technologies tend à modifier la communication entre les différents acteurs du monde des affaires. Notamment entre l'entreprise et ses clients.

A l'heure de la mondialisation, les CRM/ ERP apparaissent comme un des moyens de renforcer la multinationalisation des entreprises avec une structure mondiale de partage de l'information.

Nous verrons dans ce chapitre les notions fondamentales concernant les ERPs et CRMs afin de nous familiariser à ces deux technologies auquel s'intègrera notre futur système.

1. Customer Relationship Management CRM

Dans cette section nous allons présenter le CRM dans son intégralité afin d'avoir une vue d'ensemble de cet outil et de son fonctionnement ce qui nous permettra par la suite d'adapter notre futur système aux normes auquel il obéit pour mieux l'intégrer.

1.1. Origine du CRM [MH]

- **Avant le CRM**

Avant le CRM, les entreprises ont fait très peu pour garder les clients après l'achat. De nombreuses entreprises, surtout les grandes ne se sentent pas dans le besoin de satisfaire les clients. Les cadres ont l'état d'esprit qu'ils pourraient facilement remplacer leurs clients. Bien que cela puisse être partiellement vrai dans le passé, l'entrée dans l'ère de l'information, change tout. Depuis 1980, les clients ont commencé à devenir plus conscients des produits qu'ils achètent, des choix alternatifs et des sociétés d'où ils ont acheté.

- **Au commencement**

L'entreprise en 1980 a commencé à utiliser la base de données pour suivre les clients existants et potentiels. C'est ce qui a finalement conduit à la création d'un logiciel de CRM. Décrit comme «le marketing de base de données", cette méthode a entraîné l'envoi de relevés aux clients et la tenue de groupes de discussion. Cependant, de nombreux inconvénients

existent avec cette méthode particulière. Par exemple, alors que les entreprises peuvent collecter des données, le traitement, l'analyse et l'interprétation ont été très difficiles et prennent du temps.

- **Les années 90**

Le terme «Gestion de la Relation Client» a été inventé dans les années 90 qui ont vraiment vu la technologie et des logiciels CRM évoluent au marketing de base de données simple. Contrairement au passé, les entreprises ont commencé à utiliser le CRM comme un moyen de donner encore à la clientèle. Par exemple, de nombreuses entreprises de cartes de crédit et compagnies aériennes offrent des incitations telles que des points de fidélité, ces programmes sont conçus non seulement pour obtenir de nouveaux clients, mais aussi pour améliorer la fidélité des clients.

- **Début des années 2000**

Durant les années 2000, le CRM est devenu plus abordable et personnalisable. Cette dernière version du logiciel de CRM appliquée dans de nombreux domaines, comme n'importe qui peut le modifier à leurs besoins spécifiques, a également contribué à maturité le CRM. Les entreprises peuvent collecter plus d'informations à différents niveaux et les traiter beaucoup plus rapidement et plus efficacement que jamais auparavant.

- **CRM Aujourd'hui**

Logiciel de CRM est aujourd'hui essentiellement utilisé dans le domaine des services à la clientèle et de la technologie. Les institutions financières, les entreprises de technologie et de l'industrie des télécommunications utilisent généralement des logiciels de CRM plus que d'autres industries et de domaines.

1.2. Définition d'un CRM [E.Raouia]

- **Vue fonctionnelle**

La Gestion de la Relation Client connue sous le nom anglo-saxon (Customer Relationship Management) est la dernière fonction marketing créé dans l'entreprise. Née d'un besoin accru de conquête des clients, le GRC aborde le marketing non pas par la logique de produit mais par la logique du client : comment connaitre le client, ou le trouver, comment le fidéliser et comment le retenir ?

- **Vue logiciel**

Outils indispensables, les solutions de Gestion de la Relation Client apportent l'infrastructure nécessaire à l'atteinte des objectifs des entreprises.

Ils doivent permettre à tous les interlocuteurs de l'entreprise de prendre la dimension du client dans la globalité de sa relation avec l'entreprise et pas seulement dans sa relation individuelle avec chaque métier ou service interne, la connaissance du client est ainsi impactée.

Chaque département de l'entreprise ne détient plus sa connaissance du client mais toute l'entreprise dispose à tout moment d'une vision à 360° du client contenant l'ensemble des interactions du client avec l'entreprise quel que soit le moment et l'interlocuteur.

Afin de fournir cette vision, les solutions CRM offrent à tous les interlocuteurs de l'entreprise en contact avec le client des fonctions adéquates et plus particulièrement :

- Les actions marketing (avant-vente) : le CRM permet d'analyser et de tirer profit des flux d'informations tirés des études de marché notamment à travers des outils tels que l'Entreprise Marketing Automation.
- La gestion des ventes : le CRM automatise également la gestion des ventes à travers de nouveaux outils (Sales Forces Automation) qui permettent d'améliorer l'efficacité des vendeurs.
- La gestion du service après-vente : à travers l'instauration de centres d'appels chargés de l'assistance à distance des clients, le CRM permet à l'entreprise d'être à l'écoute de ses clients et de répondre à leurs questionnements.

Le partage des informations client de qualité entre toutes ses fonctions au sein d'une base de données client unique constitue la pierre angulaire de la mise en place d'une infrastructure CRM.

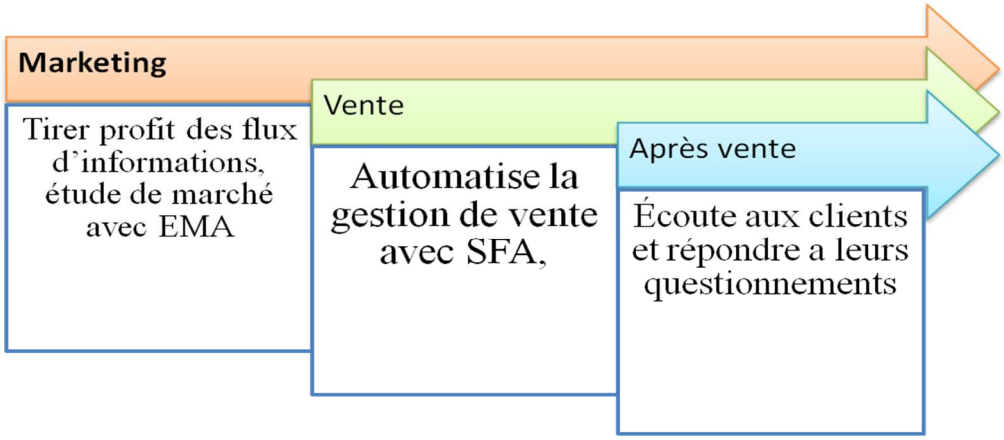


Figure 1: les fonctionnalités offertes par le CRM

1.3. Objectifs et fonctionnalités [FAB 07]

L'objectif sur lequel est fondé le CRM est : « Mieux servir le client » cette approche n'est pas gratuite, elle bénéficie bien sûr au client qui sera mieux compris donc mieux traité mais elle bénéficiera surtout à l'entreprise qui, ayant épuisé ses propres ressources de productivité, de pouvoir ainsi :

- Recruter de nouveaux clients
- Fidéliser ses clients
- Augmenter la consommation moyenne de chaque client.

Le CRM s'impose donc comme une source de profit mais également comme une source d'économie, avec une architecture modulaire dont chacun de ses objectifs précis et ses fonctionnalités fondamentales sont :

Figure 2 : vision de la gestion de la relation client

I. Le module 'marketing et analyse'

Le module "marketing et analyse" permet aux entreprises d'étudier les comportements des clients, d'envoyer leurs offres (publicitaires et promotionnelles en général) grâce à divers moyens de communication, de gérer tout ce qui englobe la relation commerciale. Le module "marketing et analyse" prend en compte les fonctionnalités suivantes :

- Le **Mailing**, soit l'envoi en nombre d'un document. Le mailing est dit " personnalisé " si on utilise des champs pour modifier le message en fonction du destinataire. En général, utilisé avec une liste d'adresses de diffusion.
- L'**e-mailing**, qui est l'équivalent électronique du marketing direct, consistant à prospecter et/ou fidéliser ses clients, via l'émission groupée et automatique de courriels (e-mails).
- Le **faxing** et les **SMS** pour effectuer du publishing par ces deux modes de transmissions.
- Un **requêteur** complet, c'est à dire un système qui permet de réaliser des requêtes de manière aisée (sans connaître le langage SQL ou QBE spécifique aux bases de données), à travers une interface ergonomique.
- La **gestion de documentation commerciale/marketing** qui permet la création et enregistrement de documentations commerciales/ marketing types.

- La **veille concurrentielle**, soit la surveillance des forces et des faiblesses de l'organisation, de l'entreprise, de la fabrication, des coûts, etc., en comparaison avec la concurrence.
- La **gestion des territoires commerciaux**, c'est à dire la gestion de la répartition des représentants ou commerciaux sur les territoires.
- La **gestion WEB**, c'est à dire la gestion du contenu, du nombre de visites, du chemin parcouru par le client sur le site Internet.
- Le **reporting/Etat**, c'est à dire la gestion du contenu d'un rapport/état avec la possibilité de modèles. Ces rapports/états sont imprimables.
- La **gestion call center**, soit la gestion d'appels téléphoniques, récupération d'informations, etc.
- La définition de **règles de workflow**, à savoir la transmission automatique d'informations (documents, e-mails, pop-up, etc.) au sein d'une entreprise, en fonction de ses processus métiers.
- Des **analyses**, **statistiques** et **graphiques** (histogrammes, camemberts, etc.) doivent pouvoir être générés.

II. Le module ‘vente ‘

Le module "vente" permet de gérer tout ce qui se rapporte à une vente et, plus particulièrement dans notre cas, la gestion de relation client. Ceci en vue de permettre aux entreprises de prévoir, d'analyser et donc ensuite de pouvoir fixer des plans marketing destinés à leurs clients. Ce module prend en compte les fonctionnalités suivantes :

- La **gestion des contacts** (clients et/ou prospects) à travers un système de fiche qui regroupe toutes les informations d'un client ou prospect (nom, prénom, adresse, âge, profession, adresse e-mail, etc.). Il doit également être possible de relier les contacts entre eux (parrainage, plusieurs fiches contacts, même famille, etc.).
- La **gestion des doublons**, c'est à dire la gestion de l'unicité des informations pour une meilleure qualité de celles-ci.
- La **gestion des opportunités**, qui permet aux équipes des ventes de collaborer et de conclure les affaires plus rapidement. Par exemple, en offrant la possibilité de mettre à jour les informations relatives aux contrats, d'assurer le suivi des événements jalons, des opportunités et d'enregistrer toutes les interactions relatives aux opportunités à partir d'un point unique.

- La **gestion des processus de vente** complets à travers des formulaires : devis, commande, livraison, retour, avoir, facture, etc.
- Un **catalogue** de produits et les tarifs multiples de façon centralisés afin d'augmenter la cohérence, d'offrir un accès aisé aux données produites et aux informations de tarification précises.
- La **planification des ventes** c'est à dire la programmation des actions et opérations de vente à mener, les objectifs, les moyens à mettre en œuvre, les durées, etc.
- La **gestion des comptes**, à savoir la gestion de toutes les données de compte client, notamment les informations concernant les contacts, les organigrammes des clients, le rôle joué par chaque contact dans la relation commerciale, les documents utiles, les partenaires impliqués dans le compte, etc.
- La **gestion des contrats**, c'est à dire la gestion de l'ensemble du cycle de vie client, de l'approbation d'un contrat à son renouvellement.

III. Le module ’gestion et organisation’

Le module "gestion et organisation" contient tout ce qui permet à l'entreprise de gérer, suivre et organiser tous ses documents. Ce module prend en compte les fonctionnalités suivantes :

- La **gestion de documents** (privés accessibles selon certains droits et publics accessibles par tous).
- Le **suivi/historique des tâches**, c'est à dire des informations de suivi (trace) des opérations effectuées sur les événements et les applications reliées.
- L'**import/export** (en une seule fois) de données contenues, ou à ajouter à une base de données.
- Un **tableau de bord,** c'est à dire d'un gestionnaire ou d'un décideur présentant des indicateurs permettant de suivre et d'anticiper le fonctionnement et l'activité de l'entreprise ou du service.
- Une **messagerie électronique**, à savoir un système permettant l'envoi et/ou la réception de courrier électronique.
- Un **agenda** (public, privé) est un outil qui permet d'associer des actions à des moments, et d'organiser ainsi son temps (alertes possibles).
- Des **alertes**, soit un type de messages visant à informer un/des utilisateurs (en général, suite à une modification d'informations de base de données).

- La **gestion de pièces jointes** est la possibilité d'associer une pièce jointe de tous types (image, photo, vidéo, audio) à un compte, un contact, un produit, etc.

1.4. Cycle de gestion de la relation client [SH]

Comme nous l'avons vu, le CRM couvre l'ensemble du processus d'échange entre le client et l'entreprise : la prospection, la vente et le service. Pour ce faire, le CRM doit :

- **Communiquer**: ne jamais perdre le contact avec le client lui-même, échanger en permanence des informations avec toute personne ou organisme disposant d'informations client pertinentes.

Techniquement, à ce stade, on parlera de CRM collaboratif qui regroupe l'ensemble des canaux d'échanges entre l'entreprise, les opérateurs du marché et le client (e-mail, call centers...etc.).

- **Analyser:** Chaque information ainsi récoltée devra être notée, classée et centralisée afin d'être ensuite passée au crible de l'analyse statistique.

Techniquement, cette étape du processus est qualifiée de CRM analytique qui regroupe des outils analytiques puissants comme le Datawarehouse (entrepôt de données) ou le Datamart, plus spécialisé (Marché de données). En fait, tous les outils permettant de rechercher une information pertinente dans une masse importantes de données clients. En d'autres termes, le Datamining.

- **Traiter:** A chaque donnée analysée doit correspondre une réponse adéquate. Une solution.

Ici, techniquement, nous parlerons de CRM opérationnel c'est-à-dire de l'automatisation de la réponse.

1.5. Les avantages offerts par le CRM

Pour faire face à la concurrence et rester compétitif, il est avantageux d'utiliser des outils tels que le CRM dont on peut en tirer les bénéfices suivants:

- ✓ Fidéliser ses clients,
- ✓ Faciliter l'accès à l'information sur les besoins des clients et avoir une base d'information commune à plusieurs employés,
- ✓ Maîtriser la relation avec les clients,
- ✓ Attirer de nouveaux clients,

- ✓ Augmenter les ventes, ainsi le chiffre d'affaires de l'entreprise,
- ✓ Améliorer l'image de marque de l'entreprise,
- ✓ Gagner du temps,
- ✓ Optimiser la collaboration entre les différents services de l'entreprise (commercial, marketing, service après-vente),
- ✓ Aide à renforcer les activités commerciales sur le long terme en faisant gagner en efficacité sur la gestion des processus de ventes.

2. Enterprise Ressource Planning (ERP)

A ce niveau du chapitre, nous allons présenter l'ERP dans son intégralité afin d'avoir une vue d'ensemble de cet outil et de son fonctionnement ce qui nous permettra de mieux positionner notre futur système dans un ERP.

2.1. Origine de l'ERP [WIKI 15]

L'origine des ERP se trouve dans les méthodes de planification des besoins en composants qui ont été développées dans le cadre d'un impératif d'intégration de plus en plus poussée des fonctions de gestion de l'entreprise. Dans les années 1960, Joseph Orlicky étudie le programme de production de Toyota et conçoit le Material Requirements Planning (MRP). Puis Oliver Wight et George Plossl mettent au point le MRP manufacturing resource planning (MRP2). D'où une évolution en trois phases :

1. MRP0 acronyme de Material Requirements Planning Zero (planification des besoins en matières 0) : méthode de calcul des besoins matière, mise au point en 1965 ;
2. MRP1 acronyme de Material Requirements Planning One : première application industrielle de la gestion intégrée des flux de production, mise au point en 1971 ;
3. MRP2 acronyme de Manufacturing Resources Planning Two (planification des ressources pour la fabrication 2) : en plus du calcul des besoins nets en matières premières et composants, effectue une planification des lancements en tenant compte des capacités des ressources par période ; mise au point en 1979.

À partir de 1990 environ, la logique introduite par le MRP s'étend progressivement à l'ensemble des fonctions de l'entreprise, pour donner l'**«ERP»** (E comme entreprise).

Enfin, les profondes modifications nécessitées par le passage informatique à l'an 2000 et le passage à l'euro (dans la zone euro), ont fait que les progiciels de gestion intégrés ont été considérés comme une opportunité technique à saisir pour remplacer des systèmes informatiques vieillissants et difficilement convertibles pour des raisons d'obsolescence technique. L'expansion considérable de ces logiciels dans les années 1990 s'explique en grande partie par les améliorations techniques apportées dans la conception et la mise à jour des systèmes d'information, mais aussi par les opportunités conjointes de refonte des organisations ouvertes à cette occasion.

2.2. Définition d'un ERP [FAB 06]

L'acronyme ERP signifie "Enterprise Ressource Planning" traduit en français par Progiciel de Gestion Intégré ou PGI. ERP est le terme le plus couramment utilisé.

Émanant d'un concepteur unique, un ERP est un progiciel qui permet de gérer l'ensemble des processus d'une entreprise intégrant l'ensemble de ses fonctions comme la gestion des ressources humaines, la gestion financière et comptable, l'aide à la décision, la vente, la distribution, l'approvisionnement, la production ou encore du e-commerce.

Le principe fondateur d'un ERP est de construire des applications informatiques correspondantes aux diverses fonctions citées précédemment de manière modulaire sachant que ces modules sont indépendants entre eux, tout en partageant une base de données unique et commune au sens logique.

L'autre principe qui caractérise un ERP est l'usage de ce qu'on appelle un moteur de workflow qui permet, lorsqu'une donnée est enregistrée dans le SI, de la propager dans les modules qui en ont l'utilité, selon une programmation prédéfinie.

Ainsi, on peut parler d'ERP lorsqu'on est en présence d'un SI composé de plusieurs applications partageant une seule et même base de données, par le biais d'un système automatisé prédéfini et éventuellement paramétrable, un moteur de workflow.

2.3. Caractéristique d'un ERP [PN 10]

- Il est issu d’un concepteur unique.
- Une modification sur un module provoque une mise à jour en temps réel des autres modules liés.
- Un ERP garantit l’unicité des informations, grâce à la centralisation des données dans une base unique, accessible à tous les modules applicatifs.

- Un ERP facilite l'audit en cas de dysfonctionnement, permettant d'identifier facilement le ou les modules concernés; il est facile de retrouver et d'analyser l'origine de chaque information.
- Un ERP peut suffire à couvrir la totalité des besoins de l'entreprise en termes de système d'information (la nature modulaire de l'ERP permet également de l'implémenter progressivement, module par module, selon les besoins).
- Moins de flux papiers
- Moteur de WORKFLOW .

2.4. Architecture modulaire des ERP's [FAB 06]

Un ERP est modulaire dans le sens où il est possible de n'avoir qu'une ou plusieurs applications en même temps, ou peu à peu. Les applications modulaires telles que les ERP permettent d'être sûr de la compatibilité des modules entre eux, ils s'imbriquent comme des blocs de Lego et fonctionnent ensemble (pas de vérification de compatibilité à effectuer).

L'architecture modulaire schématisée ci-dessus intègre plusieurs modules touchant aux grandes fonctions d'une entreprise que l'on peut détailler de la manière suivante: le module finance, logistique et e-commerce.

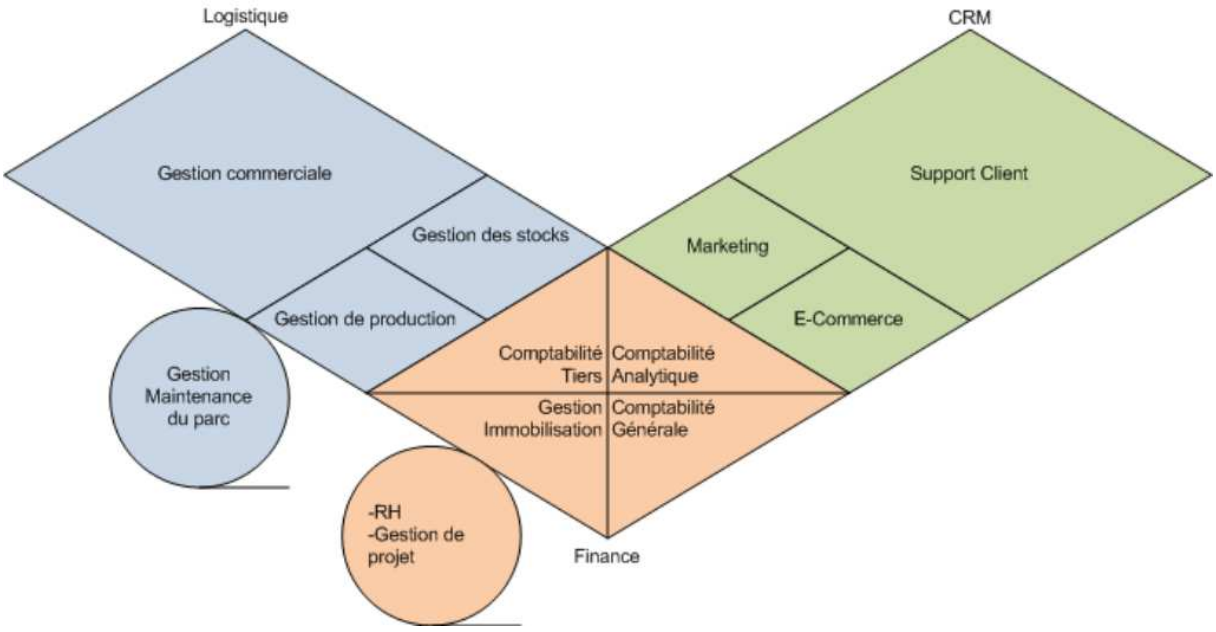


Figure 3: Architecture modulaire d'un ERP

I. Le module Finance

Le module finance se divise en 4 voire 5 sous-modules. Il rassemble les données pertinentes (en particulier les données comptables de l'entreprise) pour établir sur un plan international des comptes annuels. Ce module effectue du contrôle de gestion et des prévisions concernant les objectifs de l'entreprise.

De même, il permet de faire de la comptabilité tiers, analytique, générale, de gérer les immobilisations (gestion des investissements), les ressources humaines, soit l'administration du personnel, des frais de déplacement et du temps de travail ainsi que la gestion de paie et les besoins en postes.

Ce module est complexe car il faut avoir de bonnes connaissances en comptabilité.

II. Le module logistique

Le module logistique ou encore appelé Negoce, est le module le plus convoité par les entreprises car il permet de gérer tout ce qui se rapporte aux ventes/achats, en particulier la gestion des stocks qui coûtent chers aux entreprises.

Ce module gère les commandes clients et les livraisons. Il assure une liaison directe avec le compte de résultats et le système de production. Il permet l'optimisation des processus de workflow, la gestion précise des stocks, des contrôles qualités et factures. Suite au contrôle qualité, il y a coordination et déclenchement des mesures correctives. Enfin, il est possible de planifier/gérer/suivre la maintenance du matériel.

III. Le module e-commerce

Ce module permet de faire du e-commerce. C'est un logiciel de Gestion Relation Client (GRC) plus communément appelé Customer Relationship Management (CRM). Il permet d'effectuer les statistiques voulues (sous forme de requêtes) sur tous types de données (en particulier clients, vendeurs, production, fournisseurs,...). Un raquetteur est disponible et il n'est pas utile que l'utilisateur connaît le SQL. Mais sachant le nombre très important de tables, celui-ci doit donc impérativement savoir "lire" un Modèle Conceptuel de Données (MCD) que les éditeurs mettent à disposition. Ensuite ses résultats sont analysés en fonction des besoins de l'entreprise. En particulier pour ce module, un outil de reporting est associé.

Enfin, ce module permet, suite aux résultats, d'effectuer de l'e-mailing, des offres marketing.

2.5. Les avantages d'un ERP

Avant de mettre en place un ERP, chaque service avait son propre système d'information. Pour faire le lien entre ces différents systèmes, les situations suivantes se produisaient :

- Double voire triple saisie des mêmes informations dans des systèmes d'informations distincts.
- Au mieux, l'entreprise faisait développer des interfaces informatiques entre ses différents SI.

Pour mettre fin à cette situation, les entreprises ont décidé d'implémenter un ERP dont les bénéfices sont les suivants :

- Éviter la redondance d'informations entre différents SI de l'entreprise.
- Cohérence et homogénéité des informations.
- Disposer d'un outil multilingue et multidevises (très adapté aux multinationales comme aux PME/PMI qui veulent exporter).
- Éviter des restitutions d'informations divergentes entre différents services et donc apaiser les conflits qui en résultaient.
- Une meilleure coordination des services et un meilleur suivi du processus de commande qui inclut la prise de commande, l'enregistrement d'une sortie de stock, l'expédition de la commande et l'émission d'une facture.
- Une normalisation de la gestion des Ressources Humaines, en particulier pour les entreprises qui gèrent de nombreuses entités, parfois géographiquement dispersées.
- Création d'un environnement de travail standardisé, identique pour tous.
- Optimisation des processus de gestion.
- Intégrité et unicité du Système d'information.
- Communication interne et externe facilitée par le partage du même système d'informations.
- Minimisation des coûts (formation et maintenance).
- Mise à disposition d'indicateurs, de tableaux de bord plus fiables que lorsqu'ils étaient extraits de plusieurs systèmes différents.
- Meilleure coordination des services et clarification des processus (par exemple du processus de commande de la prise de commande à l'émission d'une facture en passant par l'enregistrement d'une sortie de stock et l'expédition de la commande) ;
- Harmonisation de la gestion des ressources humaines, notamment pour les entreprises "multi entités", parfois géographiquement dispersées.

3. La relation entre le CRM et l'ERP [KL 15]

Enterprise Resource Planning (ERP) et Customer Relationship Management (CRM) sont deux systèmes conçus pour automatiser et simplifier les processus dans une entreprise. Alors que le CRM est un système de gestion du Front-Office de l'entreprise, ERP est impliqué dans la gestion du back-office. La relation entre ERP et CRM est que ce sont deux systèmes qui ont besoin de travailler en collaboration pour que le front office et le dos de l'entreprise travaillent efficacement.

La gestion de la relation client est le processus de la façon dont une entreprise gère ses contacts clients. Comme son nom l'indique, le CRM se concentre sur comment, quand, où et pourquoi l'entreprise interagit avec les clients actuels et potentiels. Il comprend la gestion des activités de marketing pour attirer de nouveaux clients et de communiquer avec les clients existants. Les comptes clients, la gestion des activités de la clientèle et soutien à la clientèle sont les autres principaux domaines couverts par ce processus.

Le progiciel de gestion intégré est un système qui gère la façon dont l'entreprise est gérée et exploitée, il traite des fonctions d'affaires de la société. Certains des procédés qui relèvent de l'ERP sont: service à la clientèle, la gestion des stocks, la planification de la production des biens et services que l'entreprise vend, suivi des commandes, et l'achat de fournitures pour l'activité en général. L'ERP se concentre sur l'automatisation des processus de la gestion de projet, la distribution financiers, les ventes, et les zones de fabrication de petites et moyennes entreprises.

ERP et CRM se concentrent sur différentes parties de l'organisation. Bien que la gestion de la relation client (CRM) tende à se concentrer sur le visage de l'organisation et la planification des ressources de l'entreprise, (ERP) tend à se concentrer sur les opérations internes de l'organisation, les systèmes ERP et CRM ont un certain chevauchement. Certains systèmes ERP ont des fonctions de CRM intégrées dans le logiciel et certains systèmes du CRM possèdent également des fonctions de l'ERP. Bien que la majorité des fournisseurs de services et de logiciels entre les deux systèmes demeurent distincts, certains fournisseurs travaillent sur la création d'un système complet qui englobe à la fois l'ERP et le CRM d'une entreprise.

En automatisant les deux processus les clients et les processus internes de l'entreprise, ERP et CRM œuvrent afin de rationaliser toutes les fonctions de cette activité - que ce soit interne ou externe. L'essentiel est que ces deux domaines de l'automatisation aident à garder des facteurs commerciaux importants de tomber à travers les mailles du filet. Lorsque l'extrémité avant et arrière de l'entreprise communique les uns avec les autres, il couvre toutes les bases.

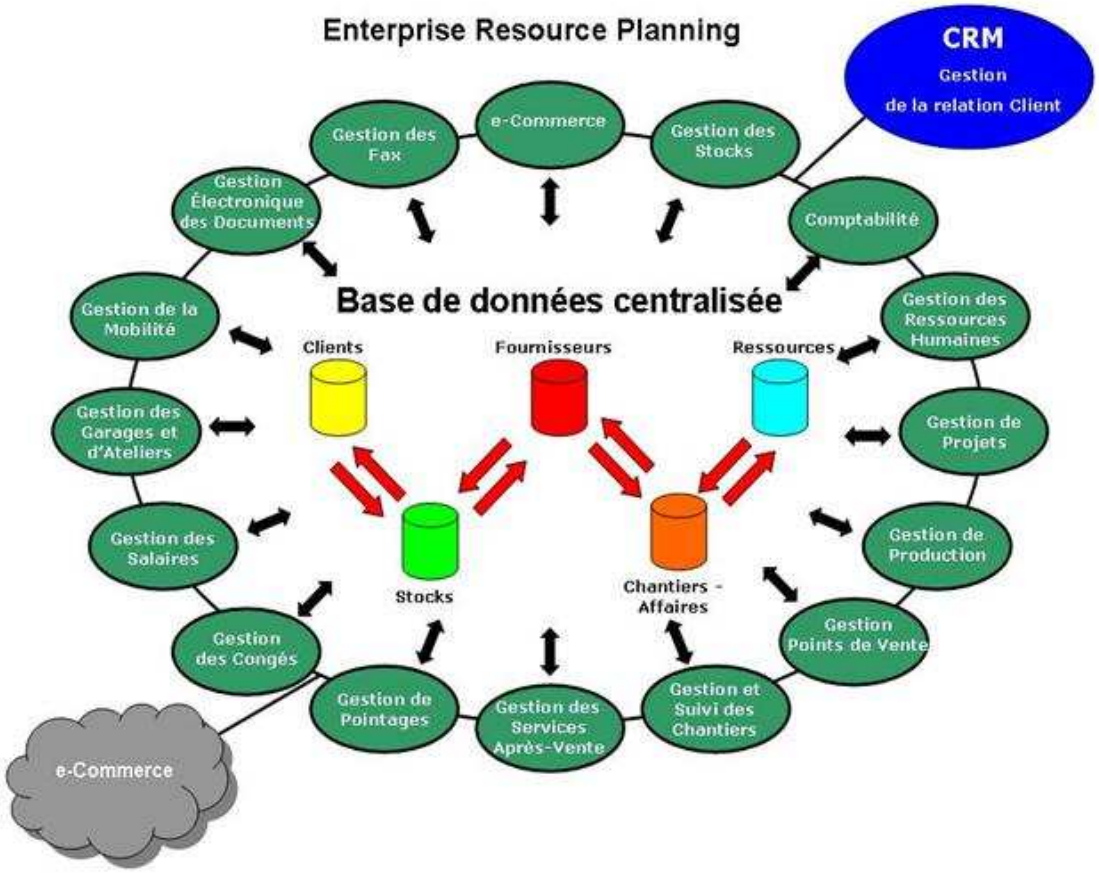


Figure 4 : La relation entre le CRM et l'ERP

Conclusion

Le CRM et l'ERP offrent des avantages remarquables aux entreprises. Alors qu'un projet CRM touche les processus de l'entreprise et modifie les habitudes de travail de nombreux utilisateurs, l'ERP est un vrai BIG BANG dans l'univers des systèmes d'informations, qui commence de plus en plus à être le cœur central des entreprises.

Bien qu'étant semblable, les systèmes ERP et CRM utilisent des approches différentes pour augmenter les profits. L'ERP se concentre sur la réduction des coûts en rendant les processus d'affaires plus efficaces et en réduisant le montant du capital consacré à ces processus. Le CRM travaille pour augmenter les profits en produisant un plus grand volume de ventes. En effet, avec une base standardisée des données du client, il est plus facile d'améliorer la relation et la fidélisation du client.

CHAPITRE 2

[ANALYSE DE L'EXISTANT]

Introduction

Nous avons inclus ce chapitre dans l'objectif d'introduire notre organisme d'accueil par une brève description, afin de prendre connaissance du domaine dans lequel ce dernier souhaite concevoir son nouveau système. Par la suite de posé la problématique pour cerner notre champs d'étude et de l'analyser et enfin de proposer une solution informatique.

1. Présentation de l'organisme d'accueil

Issue de la société **AMS (A**lgérienne **M**aritime **S**ervice) créée en 1995 en tant que représentant des armateurs Allemand Sloman Neptun et Cargo Levant qui devient en 1997, agent consignataire de navire et courtier maritime (date de libéralisation du domaine maritime en Algérie), Marine soft évolue en tant qu'entité autonome depuis septembre 1998.

1.1. Identité de Marine Soft

MARINE SOFT (MS), SARL est une société de service et d'ingénierie en informatique. Les métiers de Marine soft couvrent tous les angles d'un projet informatique, de la définition des besoins jusqu'à l'installation des machines, la mise en œuvre des solutions et l'assistance technique des divers logiciels relatifs à des activités liées au transport maritime. Cette compétence de MS permet non seulement d'avoir la maitrise du projet sans à gérer plusieurs prestataires, mais aussi à réduire les coûts et diminuer les risques.

1.2. Les objectifs de Marine Soft

Les principaux objectifs de Marine Soft sont :

- Anticiper les réels besoins des entreprises en matière de logiciels et proposer des produits pertinents, performants, faciles à mettre en œuvre, souples d'utilisation, extrêmement fiables en exploitation, surtout ouverts et évolutifs.
- Privilégier l'essentiel, à savoir l'utilisateur et l'opérationnel, avec une priorité absolue pour la qualité de service et la satisfaction totale des clients.

- Suivre dans le temps afin de prendre en compte les travaux et les recommandations des instances normatives, l'environnement institutionnel, le contexte économique national et international, les communautés sectorielles et les donneurs d'ordres.
- Accompagner les clients et les partenaires dans la mise en œuvre concrète pour réussir un projet.
- Travailler le plus possible en partenariat avec les principaux opérateurs économiques d'une part et les éditeurs de progiciels d'autre part.

1.3. A propos de Marine soft

1.3.1. Fiche signalétique de Marine soft

- **Dénomination** : MS/Marine Soft
- **Forme juridique** : Privé (Sarl)
- **Catégorie de la structure** : Société
- **Création** : 1998
- **Adresse** : 15, Rue des Frères Bouzid, Belle vue, 16200 El-Harrach, Alger, Algérie.
- **Tél**:+ 213 (0) 21 522 602 **Fax**: + 213 (0) 21 521 769

1.3.2. L'organigramme de Marine Soft

La représentation graphique de la structure fonctionnelle et de l'organisation hiérarchique des services de Marine Soft:

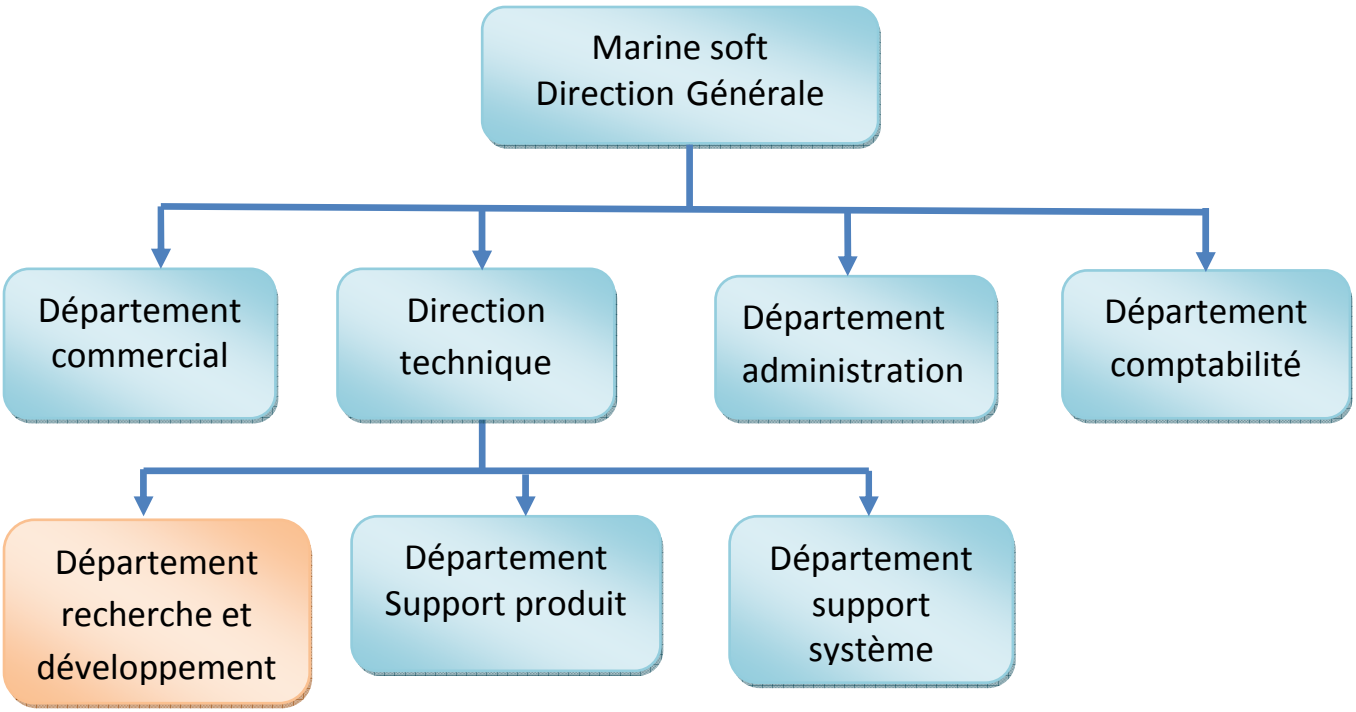


Figure 1: Organigramme de Marine Soft

1.3.3. Description de l'organigramme de Marine Soft

I. Département recherche et développement

- ✓ Développement de la gamme de progiciels de MARINE SOFT.
- ✓ Maintenance progiciels.
- ✓ Assistance technique

.

II. Département Support Produit

- ✓ Formation des utilisateurs.
- ✓ Installation et paramétrage des logiciels.
- ✓ Assistance téléphonique.
- ✓ Télémaintenance.
- ✓ Service help (outils d'aide au support).

III. Département support Système

- ✓ Gestion des parcs informatiques.
- ✓ Assistance téléphonique et télémaintenance.
- ✓ Maintenance (hardware, administration, serveur).
- ✓ Configuration de messagerie.
- ✓ Suivi des achats du matériel.

IV. Département administration

✓ Administration des ventes (Contrats &Conventions de Formation).

V. Département comptabilité

✓ Comptabilité/paie.

✓ Facturation/Recouvrement.

1.4. Les produits de marine soft

Marine Soft développe des logiciels pour chaque maillon de la chaîne de transport et ses produits sont en constante évolution :

I. Produits en architecture Client/serveur 2tiers:

Désignation	Année	Composants	Utilisateur
GAM Global Agency Management	1998	✓ Le suivi des Opérations Navire. ✓ La documentation. ✓ Déclaration douanière. ✓ Suivi des situations BL. ✓ Gestion des Conteneurs. ✓ Établissement des comptes escale et des comptes armateur. ✓ Facturation.	Agence consignataires
ZSD Zone Sous Douane	1999	✓ suivi de la documentation. ✓ la gestion de la zone. ✓ la facturation.	Entrepôt sous douane Port sec
MSGMAN Gestion de la manutention	2002	✓ Suivi des Opérations Navire. ✓ Gestion des moyens humains et matériels. ✓ Gestion de l'exploitation. ✓ Suivi de l'état de débarquement & embarquement. ✓ Statistiques. ✓ Facturations.	Les ports
		✓ Suivi des dossiers.	

MS TRANSIT	2001	✓ Déclaration. ✓ La gestion du crédit. ✓ Gestion des Conteneurs. ✓ Les réimprimés. ✓ Suivi de la tarification. ✓ Établissement des comptes clients. ✓ Facturation.	Agences transitaires
MS TRANSPORT	2002	✓ La gestion des transferts. ✓ La gestion des transports.	Agences transitaire
GPP Gestion des Parcs Pleins	2005	✓ Gestion des Conteneurs. ✓ Suivi des différents déplacements du conteneur. ✓ Facturation.	Parc de conteneurs pleins
GPV Gestion des Parcs Vides	2006	✓ Gestion des Conteneurs. ✓ Suivi des différents déplacements du conteneur. ✓ Facturation.	Parc de conteneurs vides
GRC Gestion de la Réparation Conteneur	2006	✓ Le suivi des mouvements conteneurs (entrée /sortie). ✓ L'établissement des états conteneur ✓ La facturation	Parc qui s'occupe de la réparation des conteneurs endommagés des transitaires

II. Produits en architecture client/serveur 3tiers (réalisé)

La désignation	Date de	Composant	Langage de	Utilisation

	création		**programmation**	
GMAO Gestion Maintenance Assistée par Ordinateur	2013	✓ Gestion patrimoine. ✓ Gestion stock. ✓ Gestion d'intervention. ✓ Gestion d'achat. ✓ Gestion de ressources humaines.	JAVA	Port d'alger

III. produits en architecture 3tiers (en cours de réalisation)

La désignation	**Type**	**Composant**	**Langage de programmation**	**Utilisation**
GLS Global Logistique Software	ERP	✓ Le transit. ✓ La consignation des navires et des marchandises. ✓ La manutention portuaire des marchandises. ✓ Le transport routier. ✓ L'entreposage des marchandises sous douane. ✓ La facturation.	JAVA	✓ Agences ✓ consignataires ✓ Transitaire ✓ Ports ✓ Ports secs ✓ Entrepôts sous douane

2. Problématique et objectif

2.1. Problématique

Comme tout éditeur et intégrateur de solution logiciel, MARINE SOFT doit saisir les opportunités dues aux nouveaux marchés et aux progrès technologiques. La relation client doit être de plus en plus réactive et interactive.

Des critiques sont soulevées à partir desquelles et en se basant sur leurs expériences dans l'édition de plusieurs progiciels, il a été constaté que chacun des progiciels réalisés intègre un module de facturation dont les fonctionnalités sont étroitement liées aux autres modules métiers (d'exploitation), en plus d'être conçu sur la base d'une architecture 2 tiers limitée et dépassée à cause des couts de maintenance quelle procure .

2.2. Objectifs

Notre projet de fin d'études s'inscrit dans le cadre d'une solution optimisé de la gestion de facturation suivant les normes d'un CRM qui sera intégré comme module dans les progiciels de Marine Soft existants et même dans d'autres ERPs et systèmes d'informations sur le marché et cela en se basant sur une architecture 3-tiers sous la plateforme JEE.

Notre projet de fin d'année consiste donc à proposer, concevoir et réaliser un système d'information de gestion de la facturation qui satisfait ce besoins.

- **Organisation du travail en équipe**

Recherche et développement est une équipe de marine soft, constituée des ressources humaines suivantes :

- Le directeur général.
- Le directeur technique.
- Deux ingénieurs de développement.
- Un technicien support produit.
- Deux stagiaires.

Le couple développeurs/stagiaires se voit affecter la responsabilité de mener à bien la conception, la réalisation et la mise en œuvre de l'application qu'on lui a confiée. Les

directeurs font le suivi du projet et assurent le choix technologique des outils, les architectures et les Framework qui seront utilisés pour la réalisation de notre projet.

- **Planification du projet**

La planification consiste à prévoir le déroulement du projet tout au long des phases constituant le cycle de vie prévu pour ce dernier. C'est ainsi qu'on a pu diviser la réalisation du projet de mise en place d'un système d'information de gestion de facturation en quatre phases temporellement comme le montre le tableau ci-dessous :

		N, D, J	Février				Mars				Avril				Mai				Juin				Juillet	
			S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2
PHASE 1 Théorie (annexe) (14%)	C/S (2%)																							
	UML (5%)																							
	JEE (7 %)																							
PHASE 1 Théorie (Chapitre 1) (7%)	Les technologies ERP/CRM (7%)																							
PHASE 2 Analyse de l'existant (Chapitre 2) (12%)	Présentation de l'organisme d'accueil (2%)																							
	Etude de l'existant (10%)																							
PHASE 3 Analyse et Conception (Chapitre 3) (25%)	Modélisation Fonctionnelle (15%)																							
	Modélisation Statique (10%)																							
PHASE 4 Réalisation Et Implémentation (Chapitre 4) (38%)	Architecture Et outil De développement et d'implémentation (4%)																							
	L'application sous différent angle (34%)																							
Finalité (4%)	Vérification et livraison (4%)																							

Tableau 1: Général de la planification temporel de mémoire

3. Analyse de l'existant

Afin d'arriver à élaborer une solution informatique qui répond aux exigences de l'entreprise, nous allons dans une première étape diagnostiquer le système existant, pour y parvenir on a adopté l'approche par processus comme une stratégie d'analyse dans le cas des systèmes d'informations de l'entreprise, précisément de leur module de facturation, on plus d'étudier le procédé général de facturation. A la fin de l'analyse on est passé à l'étape d'élaboration d'un bilan du diagnostic présenter en une liste de critiques en se basant sur modèle d'Ichikawa.

3.1. Les pré-requis

3.1.1. Choix de l'approche [AQ]

N'importe quel organisme est composé de nombreux processus liés les uns aux autres et qui doivent parfaitement fonctionner ensemble si l'on veut être performant, et de ce raisonnement, l'approche par processus est née comme une méthode destinée à maîtriser et améliorer le fonctionnement d'un organisme.

Adoptée par l'entreprise depuis des années, l'approche par processus a prouvé son utilité et sa valeur ajoutée à la qualité et à la performance des logiciels de marine soft. L'approche par processus est une exigence de l'ISO 9001 et consiste à :

a) Déterminer les processus nécessaires au SMQ[1] et leur application dans tout l'organisme.
b) Déterminer pour chaque processus ses éléments entrants et ses produits ou services en sortie.
c) Déterminer la séquence et les interactions entre ces processus.
d) Déterminer les critères et les méthodes pour assurer la maîtrise des processus.
e) Maîtriser les ressources et les informations nécessaires au bon fonctionnement des processus.
f) Surveiller, mesurer et analyser les processus.

[1] SMQ: *Le Système de management de la qualité est l'organisation mise en place par une entreprise (ou un organisme) pour atteindre sa politique et ses objectifs qualité.*

g) Mettre en œuvre les actions nécessaires pour atteindre les objectifs et l'amélioration continue.

A. Définition de l'Approche par processus [*CAPITALRH*]

L'approche par processus désigne l'application d'un système de processus au sein d'un organisme ainsi que l'identification, les interactions et le management de ces processus.

La norme ISO 9001 encourage l'adoption d'une approche par processus lors du développement, de la mise en œuvre et de l'amélioration de l'efficacité d'un système de management de la qualité, afin d'accroître la satisfaction des clients par le respect de leurs exigences.

B. Le processus

I. Définition

Un processus est un ensemble de ressources et d'activités liées qui transforment des éléments entrant en éléments sortant. Autrement dit, c'est une boite noire qui a une finalité (les données de sortie) qui, pour atteindre cette finalité, utilise des éléments extérieurs (données d'entrée) et les transforme (en leur donnant une valeur ajoutée) par du travail et des outils (activités et ressources). ***[AQ]***

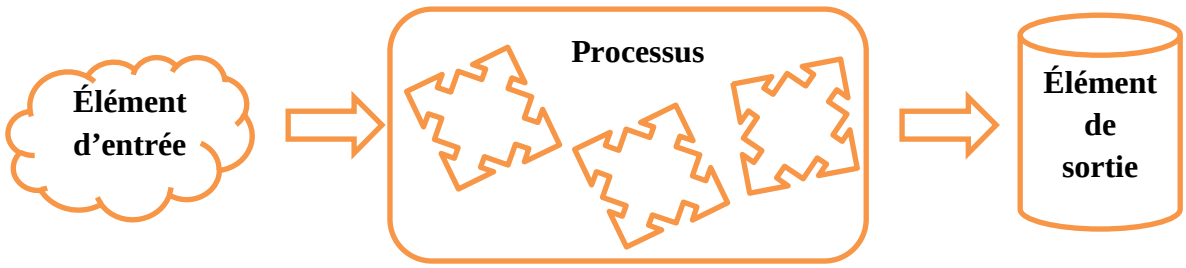


Figure 2 : Représentation d'un processus

Processus : «Toute activité utilisant des Ressources et gérée de manière à permettre la transformation d'éléments d'entrée en éléments de sortie » **source ISO 9000:2000**

II. Caractéristiques d'un processus

Une fois le processus identifié, il s'agit de le définir de manière synthétique, à travers les 5 éléments suivants : «

- un nom : court et évocateur pour tout le monde,

- une finalité : expression de la raison d'être du processus en utilisant des verbes d'action,
- les activités principales : liste des activités menées au sein du processus,
- les données d'entrée : éléments pris en charge par le processus (Matière),
- les données de sortie : éléments restitués par le processus (Produit / Service). » **[*AC 10*]**

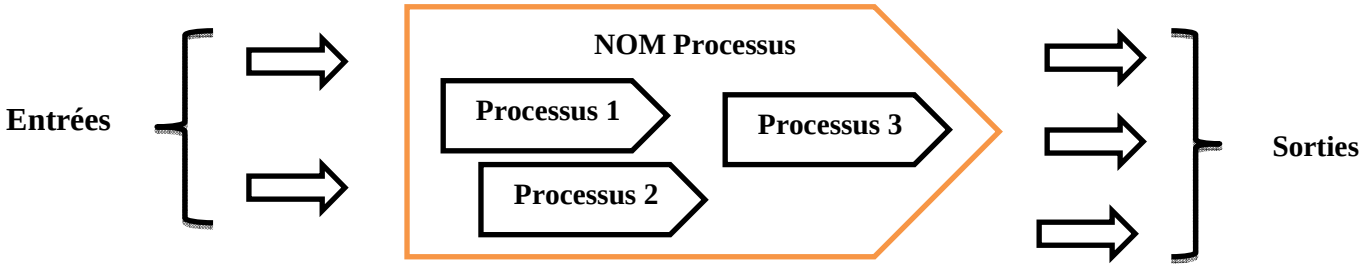
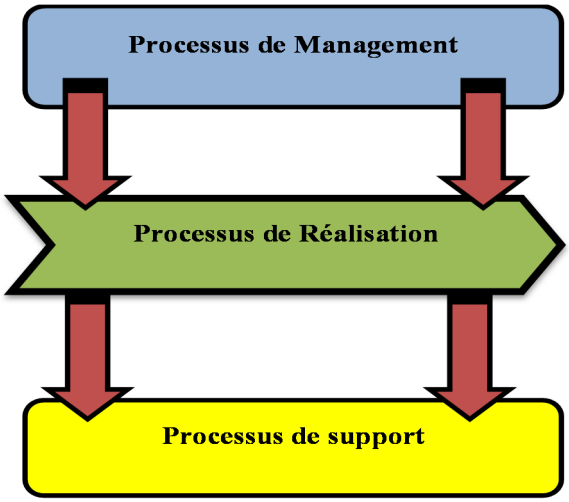


Figure 3 : Représentation des caractéristiques d'un processus

III. Type de processus

3 types de processus pour une approche processus complète :



Processus de réalisation	Processus contribuant directement à la réalisation du produit ou du service, depuis la détection du besoin du client à sa satisfaction. Ils correspondent au cœur de métier de l'organisme. Exemples : recherche et développement, conception, fabrication, livraison …
Processus de support	Processus qui contribuent au bon déroulement des autres processus en leur apportant les ressources nécessaires. Exemples : maintenance, ressources humaines, maîtrise de la documentation ; métrologie …
Processus de management	Processus qui contribuent à la détermination de la stratégie, de la politique qualité et au déploiement des objectifs à travers tous les processus de l'entreprise. Ils permettent leur pilotage et la mise en œuvre des actions d'amélioration.

Tableau 2: les types de processus de qualité [*AQ*]

Remarque :

- Dans la démarche optée pour réaliser ce chapitre, on se restreindra à l'application du processus de réalisation qui nous permettra de déterminer les séquences et interactions entres les processus des systèmes d'information existant par la construction de leur cartographie (une représentation des liens existants entre les différents processus de l'organisme), donc les processus de support et de management ne font pas objet dans ce qui suit.

C. Étapes de Construction de la cartographie [*HB JPW 03*]

L'approche par processus est une approche systémique. Cela veut dire, qu'il y aura plusieurs niveaux d'analyse. Nous allons utiliser ce principe d'analyse à plusieurs niveaux pour construire la cartographie concernant le processus de réalisation

Nous verrons ci-après au fur et à mesure de la construction de la cartographie l'apparition de nouvelle définition telle que macro-processus, processus élémentaire.

Étape 1 : Décrire l'entreprise tout entière comme un macro-processus

Avant de réaliser la cartographie détaillée d'une entreprise, il faut passer par cette première étape qui est la cartographie de niveau 1 en :

1. Représentent dans un premier temps cette entreprise tout entière comme un macro-processus qui est caractérisé par un nom, en l'occurrence le nom de l'entreprise, des entrées et des sorties et une phrase qui décrit les activités qui transforment les entrées en sorties, en rajoutant de la valeur.

2. Puis identifier :

   ✓ Les éléments d'entrée (EE) : ressources nécessaires pour démarrer un processus. Ils proviennent en ce qui concerne la matière première des fournisseurs externes ou internes (autres processus) et coté demande il provient des clients.

   ✓ Un ou plusieurs éléments de sortie (ES) : résultat de la transformation des éléments d'entrée par le processus. Ils sont destinés à un ou plusieurs clients externes ou internes (autres processus).

3. Enfin pour plus de clarté, groupé les entrées et sorties par provenance (P)/destination (D) en indiquant celles-ci dans la cartographie.

Le schéma suivant récapitule ce que doit être en principe la cartographie de niveau 1:

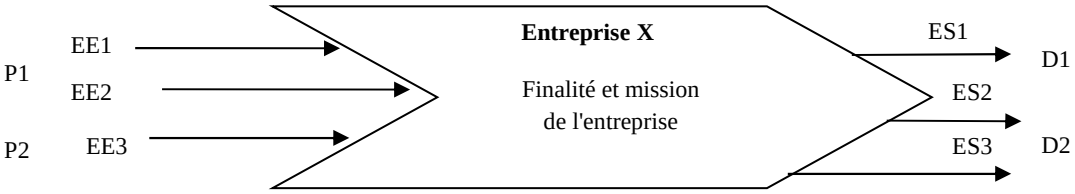


Figure 4 : Cartographie de niveau 1 (macro-processus)

Remarque :

S'assurer aussi que tous les acteurs de l'entreprise comprennent et partagent cette représentation graphique.

Étape 2 : Décrire les processus qui prennent en charge les entrées du macro-processus

À travers le premier schéma, nous avons décrit l'entreprise comme une « boîte noire », dont on ne connaît que les entrées et les sorties et pour laquelle nous n'avons décrit que sommairement ce qui est censé se passer à l'intérieur. Il s'agit maintenant d'ouvrir cette boîte noire et de décrire ce qui s'y passe. Nous descendons donc d'un niveau d'analyse et créons la cartographie de niveau 2.

Pour construire la cartographie de niveau 2, la méthode consiste à « tracer » d'abord toutes les entrées du schéma de niveau 1. Pour chaque entrée, il faut identifier quelle est la « boîte » qui la prend en charge. Nous appellerons cette « boîte » processus élémentaire (PE).

Afin de trouver ces processus élémentaire, c'est conseillé d'aller sur le terrain en suivant, très concrètement, auprès des acteurs concernés, qui prend en charge une entrée, quel traitement il effectue, quel est le résultat de ce traitement, où va le résultat de ce traitement.

Ce travail consiste donc à suivre tous les flux entrants.

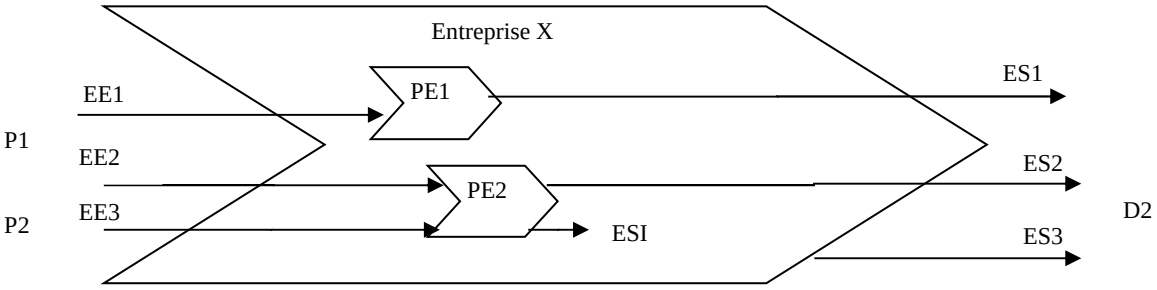


Figure 5 : Cartographie de niveau 2 de l'étape 2

ESI: Élément Sortent Interne au macro processus

Remarque :

Après avoir tracé toutes les entrées on peut rencontrer à ce niveau les cas suivants :

- que de nouvelles « sorties internes » sont apparues ;
- que toutes les sorties ne sont pas reliées à des processus élémentaires.

Le schéma est donc incomplet. Pour compléter la cartographie de niveau 2, nous allons « remonter la chaîne », en partant des sorties encore « orphelines » pour déterminer les processus élémentaires dont elles sont issues.

Étape 3 : Décrire les processus élémentaires qui génèrent les sorties « orphelines »

Dans le cas où on retrouve des sorties orphelines, il faut juste seulement identifier les processus élémentaires qui les génèrent.

Le schéma ci-contre montre le résultat de ce travail pour l'entreprise X:

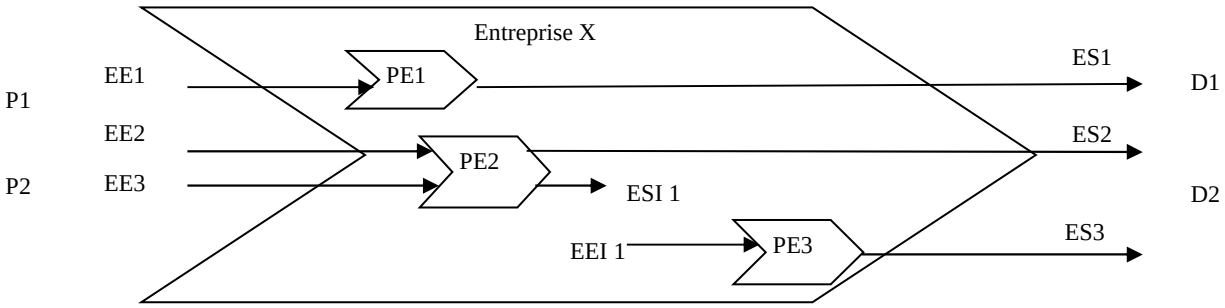


Figure 6 : Cartographie de niveau 2 de l'étape 3

EEI: Élément Entrent Interne au macro processus

Remarque :

Après avoir pris en compte toutes les entrées et sorties du schéma de niveau 1, on peut comme même retrouver encore des trous dans le schéma. Il s'agit des entrées et sorties internes des processus élémentaires identifiés qui sont en « électron libre ».

Il faut donc combler ces trous, c'est-à-dire identifier les processus élémentaires manquants.

Étape 4 : décrire les processus élémentaires qui manquent dans la chaîne

Il s'agit de prendre chaque « sortie interne » de décrire les processus qui les prennent en charge. Bien sûr, on peut aussi le faire dans l'autre sens, en partant des « entrées internes ». D'ailleurs nous constaterons que de nouveaux processus apparaitront également, que de

nouvelles entrées et sorties « internes » sont décrites, essentiellement pour expliquer les interactions entre les processus.

Le schéma ci-contre montre le résultat de ce travail pour l'entreprise X:

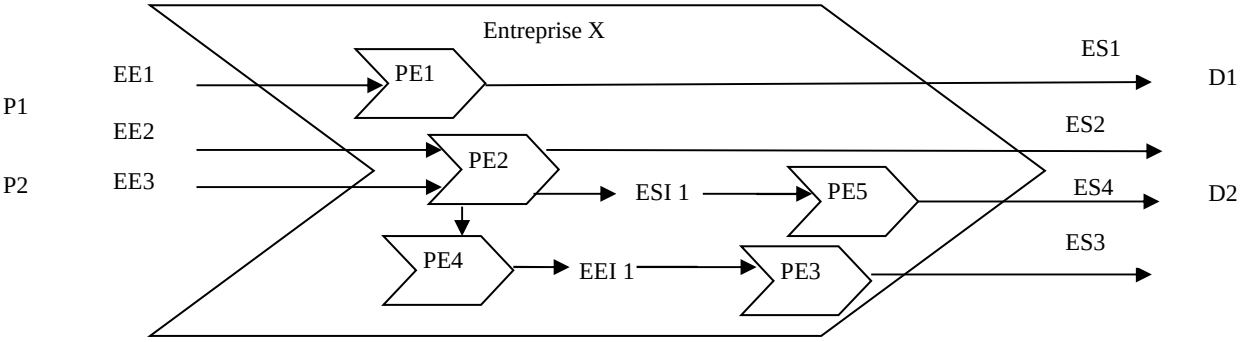


Figure 7 : Cartographie de niveau 2 de l'étape 4

Remarque:

La cartographie de niveau 2 doit être une chaîne ininterrompue de processus élémentaires qui relie toutes les entrées et sorties de la cartographie de niveau 1.

3.1.2. Présentation des concepts liés à la facturation

Nous avons intégré cette partie au chapitre afin de présenter les notions de base de la facturation et nous familiariser avec ces concepts, ce qui nous permettra par la suite de mieux saisir les différentes méthodes de facturation des systèmes existants.

La facturation en général est basée sur l'enchainement d'un ensemble de concepts de base à savoir :

1. **Facture pro forma**

La facture pro forma est un document non-comptable qui fait office d'évaluation du montant qui sera facturé au client, sous réserve que ce dernier accepte les conditions incluses dans cette "facture-devis".

2. **Facture initiale**

Une facture initiale (facture de doit) est la facture de vente établie par le vendeur-fournisseur au client-acheteur qui a une valeur comptable.

3. **Facture avoir/complémentaire**

Il est possible qu'une entreprise tienne à modifier sa facture initiale. Un événement (prévu ou imprévu) peut amener le vendeur à augmenter ou à diminuer (à la

demande du client) une facture qu'il a déjà envoyé, ce qui oblige le vendeur à établir un des deux documents suivants :

- **Facture avoir (note de crédit)**

La facture d'avoir est un document commercial établi par le fournisseur afin de rembourser au client une dette due à une remise en cause de certains éléments de la facture initiale.

- **Facture complémentaire (note de débit)**

La facture complémentaire est un document commercial établi par le fournisseur afin d'informer l'acheteur qu'il lui doit un montant complémentaire à celui mentionné sur la facture, en cas d'un constat de la part du vendeur d'une sous facturation sur les produits/services.

4. Règlement de la facture

La dernière étape de l'opération d'achat-vente est le règlement, le paiement que verse l'acheteur au fournisseur. La date et les conditions du règlement sont généralement mentionnées sur la facture. Deux modes de règlement sont utilisés : le règlement au comptant et le règlement à terme (ou à crédit).

4.1. Le règlement au comptant

Le règlement au comptant peut se faire de trois façons : à la commande, à la livraison, dans un délai suivant la livraison.

4.2. Le règlement à terme (ou à crédit)

Une facture à crédit est une facture non réglée lors de sa création, ce qui veut dire que le flux entrant n'est plus l'argent (il n'y a pas d'encaissement au moment de la réalisation de l'opération) mais l'engagement donné par le client de payer plus tard.

4.3. Le suivi du règlement

Le suivi du règlement des factures à crédit permet de régler les factures à crédit (initiale ou complémentaire) des clients conventionnés.

3.2. L'étude des systèmes existants

Toute étude conceptuelle devra reposer sur une bonne connaissance du système existant qui ne peut se réaliser qu'à travers une analyse exhaustive et complète de la situation actuelle.

Celle-ci permettra de cerner les carences et les causes des défaillances du système et d'avoir, ainsi, une meilleure connaissance de l'envergure du problème étudié.

Cette partie est consacrée à l'étude de l'existant qui se tiendra d'abord sur le procédé de facturation général puis, sur l'étude des systèmes d'informations existants dans l'entreprise plus particulièrement de leurs modules de facturation.

3.2.1. L'étude du procédé de facturation général existant

Nous avons décidé de portée notre étude sur la facturation générale existante à laquelle tous les métiers obéissent afin d'avoir une idée plus neutre sur le domaine de la facturation sans le lier à un métier.

A. Les mentions de la facture

La facture est le document établi par le vendeur au compte du client à la suite d'une vente, dans le but de déclencher le règlement des biens ou services achetés. La facture contient plusieurs types d'informations structurées de la manière suivante:

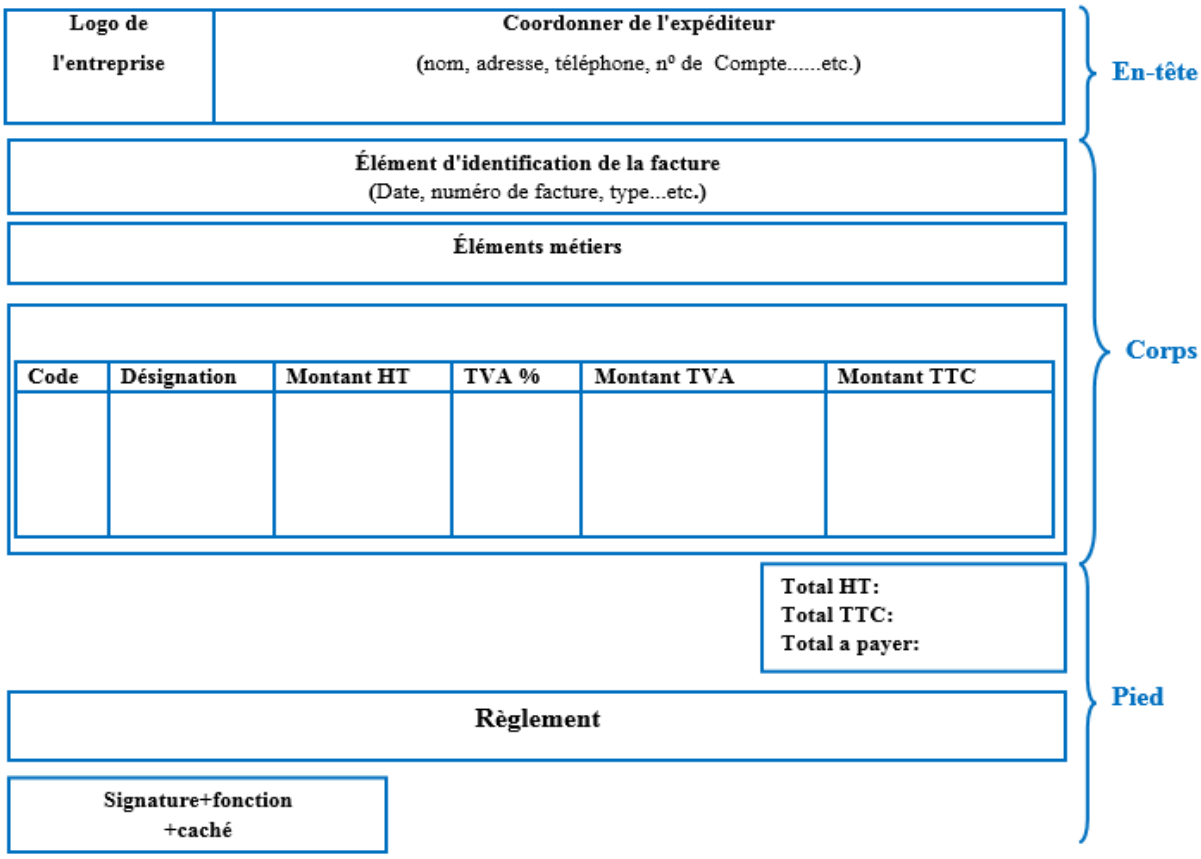


Figure 8 : Squelette général d'une facture

B. Enchainement des processus lier à la facturation

Le schéma suivant récapitule l'enchainement des processus lié à la facturation :

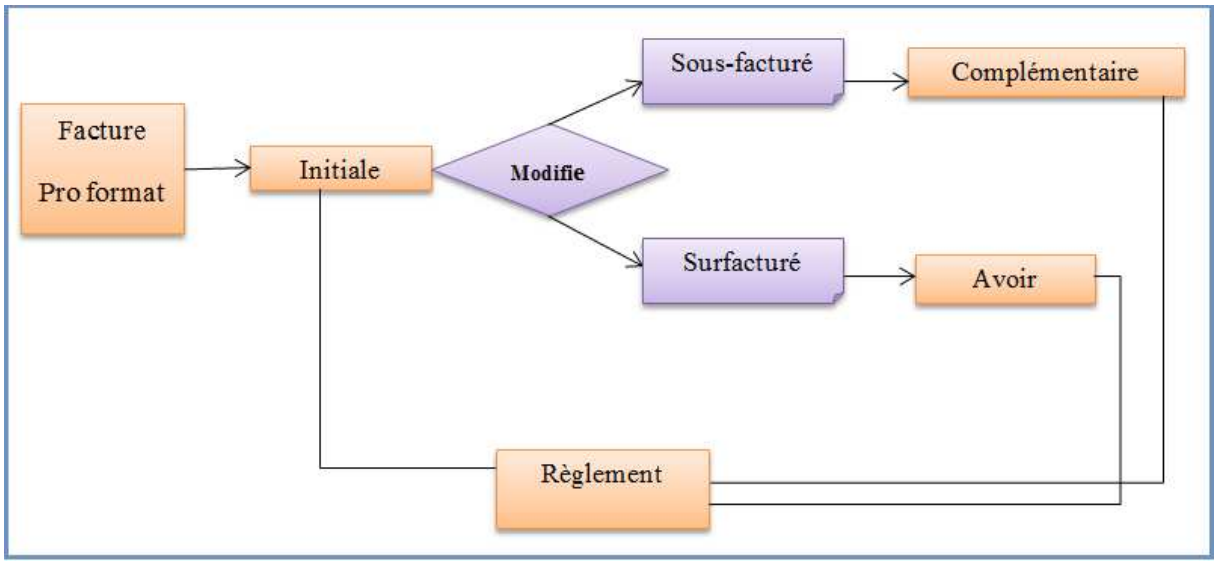


Figure 9 : Enchainement des processus lier à la facturation

3.2.2. Diagnostic du système informatique existant dans l'entreprise

A ce stade du projet , il nous a été demandé de porter le diagnostic des systèmes existants sur les trois (3) applications les plus importantes (GAM, ZSD et MSGMAN) qui englobent le procédé de facturation, dans le but de mieux comprendre et cerner son mode de fonctionnement, la politique d'implémentation et de développement de Marine Soft et pouvoir récolter l'ensemble des données manipulées dans le cadre de la facturation.

Dans le schéma suivant nous avons délimite le champ d'études comme suit :

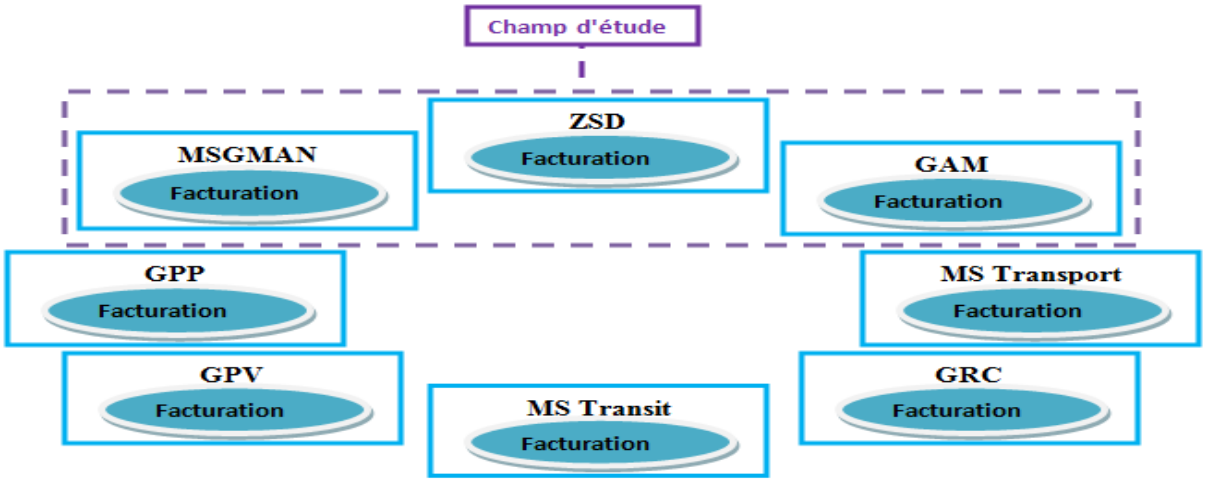


Figure 10 : Champ d'étude

La raison pour laquelle le choix de l'étude se portera précisément sur ces trois logiciels est due au fait que :

Produits	Motif
GAM	Structure complexe, englobe toutes les notions du métier (facturation)
ZSD	Valeur commerciale
MSGMAN	Le module facturation est conçu avec logique exclusive

Afin de comprendre et diagnostiquer le système existant qui va nous permettre de faire le bilan, nous avons mis en œuvre les techniques suivantes:

- Les enquêtes sont menées par : des interviews avec nos encadreurs puis les exécutants des logiciels ainsi nous avons réuni toute la documentation nécessaire.

- analyse globale des flux d'information qui circulent entre les processus du module de facturation, qui nous a amené à déterminer le rôle de chaque processus et comprendre leurs enchainements dans chaque progiciel.

Ces techniques nous ont permis de récolter et structurer les informations comme suit:

A. Présentation de GAM

I. Global Agency Management (GAM)

GAM « **G**lobal **A**gency **M**anagement » est un progiciel qui s'adresse aux agents consignataires de navires, il permet d'automatiser du début de la chaîne de traitements jusqu'à la fin les opérations de gestion quotidienne, assuré par les modules sur lesquelles ils sont répartis.

II. Les rôles des modules de GAM

Dans le but d'avoir une idée générale sur les fonctionnalités qu'intègre GAM, on a pensé à présenter sous forme de tableau la description des modules que compte le logiciel comme suit :

MODULES	DESCRIPTION
	Permet d'accéder aux fichiers et le paramétrage de tout le progiciel.
	Permet d'accéder aux fichiers de bases (type d'emballage, type navire, … etc.).
	Assure la gestion opérationnelle (dossiers, situations portuaires, … etc.).
	Permet la saisie des B/L, l'édition des avis d'arrivées, l'édition des fiches de recette pour les opérations d'importation, … etc.
	Permet la saisie des booking, la taxation des B/L (Bill of Lading), l'édition des pro format d'exportation, l'édition des fiches de recette pour les opérations d'exportation.

	Permet l'établissement du manifeste douanier, l'établissement du manifeste rectificatif, … etc.
	Permet l'établissement du bon à délivrer, le suivi des situations des B/L, … etc.
	Permet l'établissement des factures avoirs, B/L, complémentaires et surestaries, l'édition de l'état de caisse, …
	Assure le suivi des conteneurs, la gestion des mouvements des conteneurs, la gestion des pro-format de surestaries, suivi des cautions, … etc.
	Permet l'établissement des comptes des dossiers, l'édition des notes de débit douane, des comptes dossiers complémentaires, …etc.
	Permet l'établissement des comptes courants dossiers (compte armateur), la gestion des comptes dossiers transférés à l'armateur,… etc.
	Permet l'échange informatisé des données entre le progiciel et les autres logiciels armateurs, … etc.

Tableau 3 : Description des modules du progiciel GAM

III. Représentation de GAM sous forme de processus

Nous allons représenter dans cette partie GAM en utilisant l'approche par processus, on commencera par une cartographie du macro processus (N) de tout le logiciel pour passer par la suite à la schématisation d'un processus élémentaire (N-2) de GAM qui est la facturation (Module Caisse) comme suit :

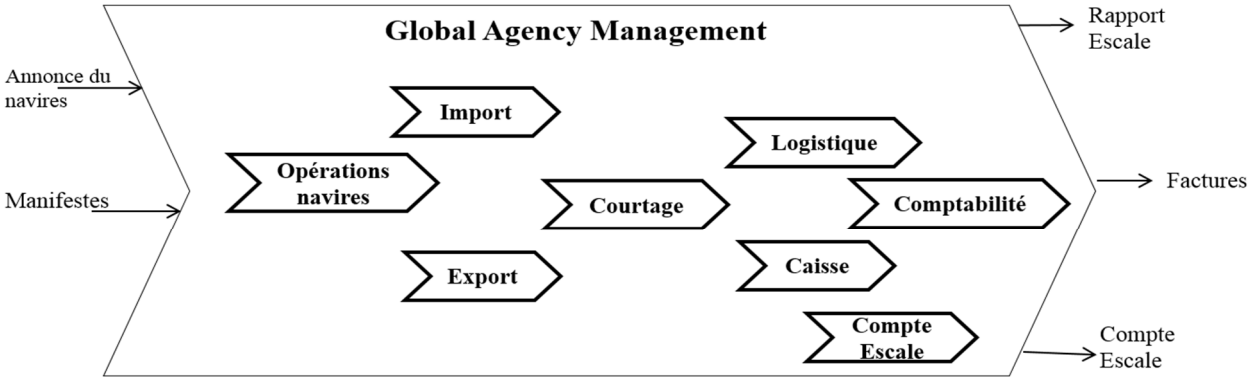


Figure 11: Cartographie Macro-processus « N» de Global Agency Management (GAM)

La shématisation de GAM on ce basent sur l'approche par processus,sous forme de macro-processus nous a permis d'identifier :

- Les entrées suivantes :
  - Annonce du navires
  - Manifestes
- Les sorties suivantes :
  - Rapport escale
  - Factures
  - Compte escale
- Les opérations de transformation des entrées en sorties répartie chacune dans les modules du progiciel en respectant un enchainement précis comme suit :

On commence par les opérations navire qui assure le suivi dés l'annonce des navires et l'envoi des manifestes, par la suite on fait appel au module import en même temps que celui d'export pour l'établissement de la documentation et le suivi des situations B/L, ainsi on passe au module de courtage pour la gestion des conteneurs, en parallèle on effectue l'établissement des factures assurées par le module caisse, pour ensuite créer les comptes escales et enfin envoyer tout ce qui est facturé à la comptabilité à travers la passerelle.

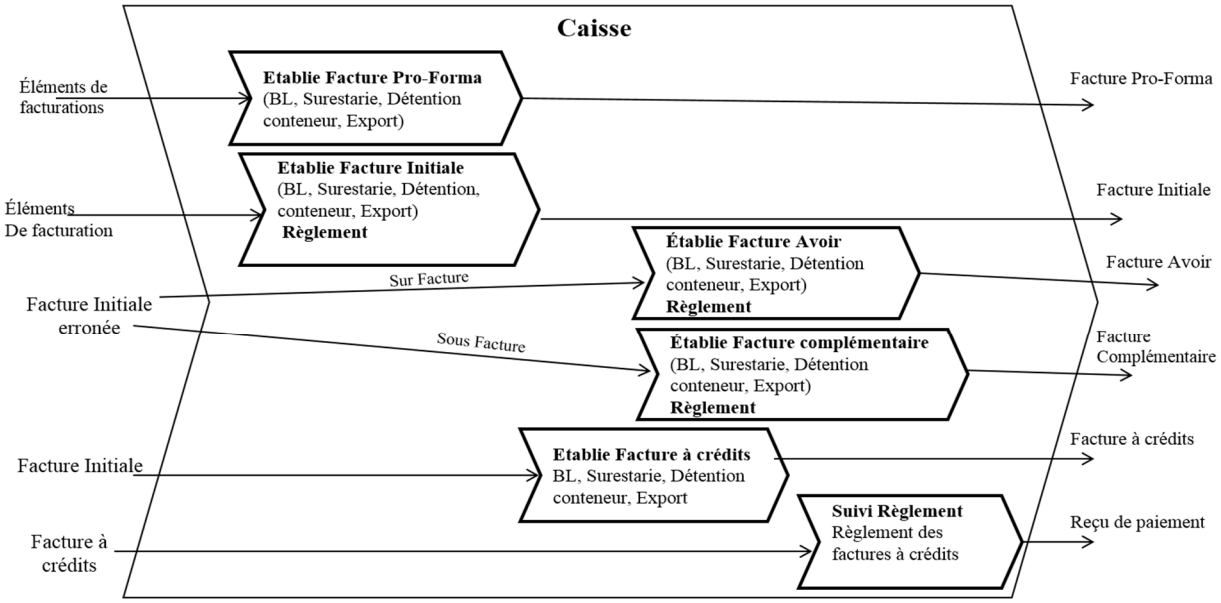


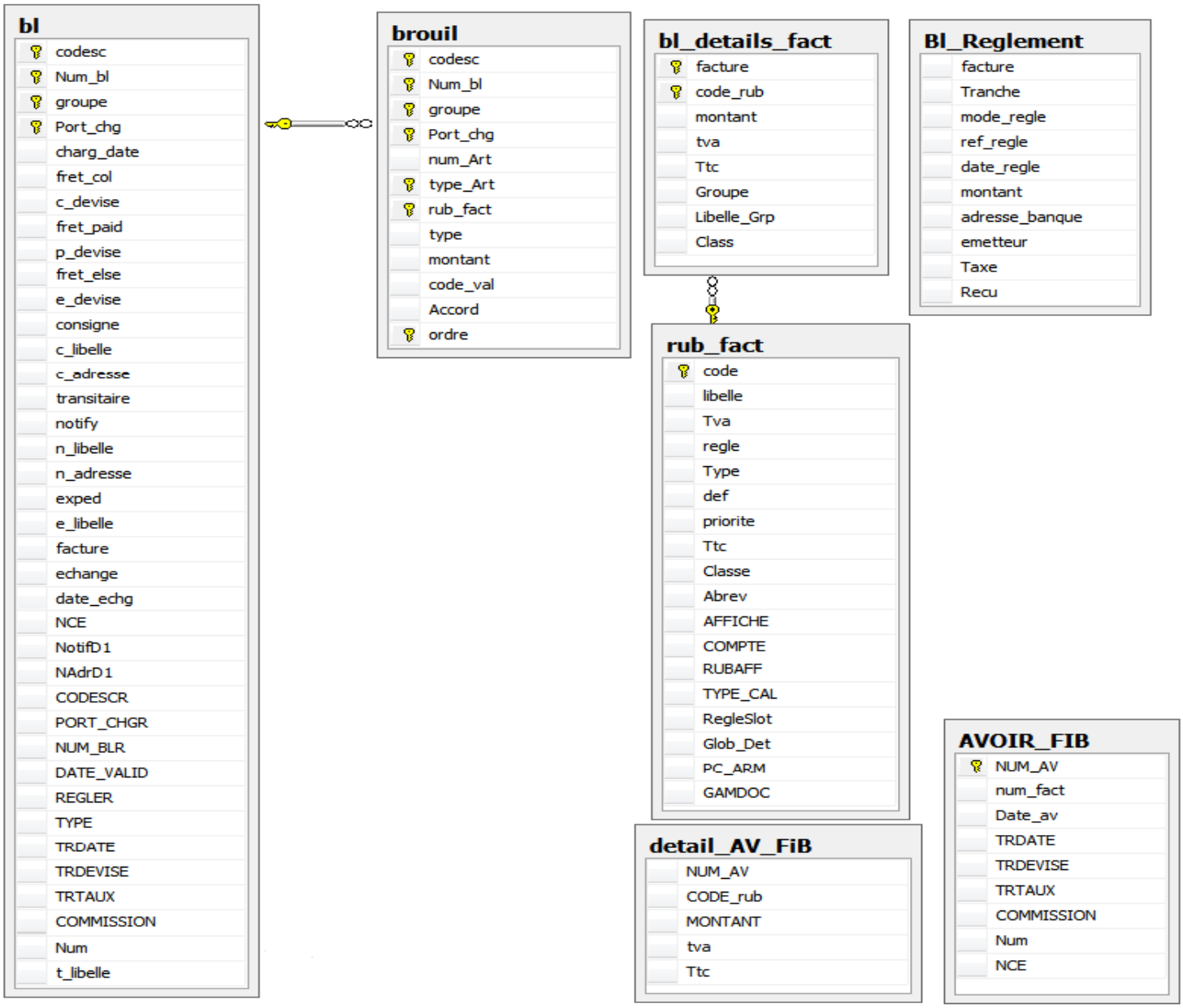
Figure 12 : Cartographie processus élémentaire « N-2 » du module caisse de GAM.

IV. Les tables de la base de données

A ce niveau de l'étude la représentation de la base de données de GAM sera restreinte à celles des tables concernant la facture B/L et Surestarie:

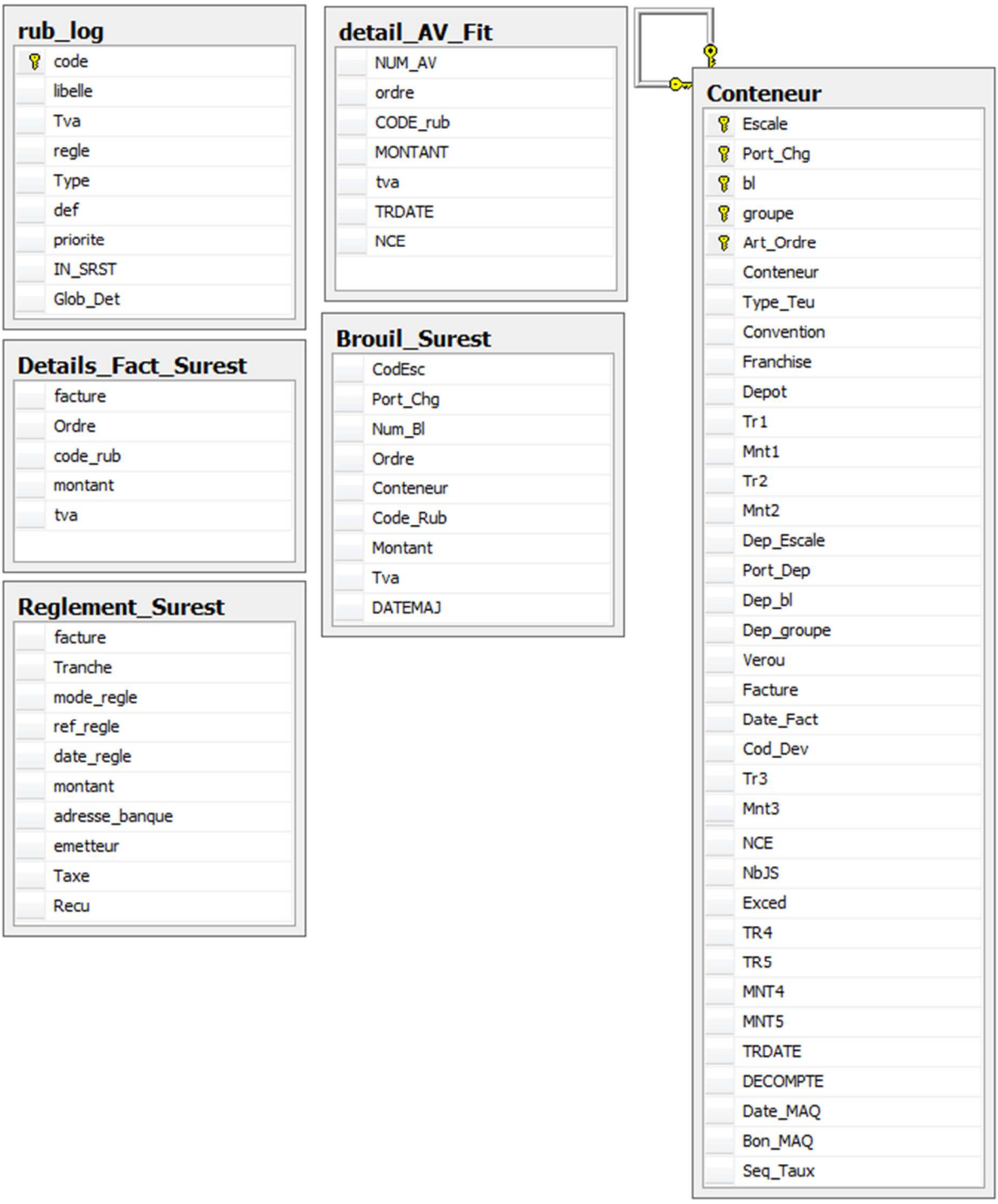
i. Les tables de la Facture B/L (Bill of Lading)

Le schéma suivant représente les tables concernant la facture B/L:



ii. Les tables de la Facture surestarie

Le schéma suivant représente les tables concernant la facture surestarie:



B. Présentation de ZSD

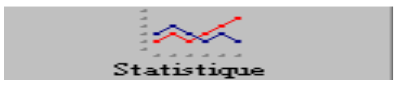
I. Zone Sous Douane (ZSD)

ZSD s'adresse en priorité aux entreprises de gestion des aires de stockage sous douane (Terminal à conteneurs, port sec, entrepôt public, zone sous douane…). L'une de ses vocations premières est de fournir le maximum d'automatismes et de contrôles dans la

gestion quotidienne, pour tout utilisateur appelé à se servir du progiciel n'ayant pas ou peu de connaissances informatiques.

II. Les rôles des modules de ZSD

Dans le but d'avoir une idée générale sur les fonctionnalités qu'intègre ZSD, on a pensé à présenter sous forme de tableau la description des modules que compte le logiciel comme suit :

MODULES	DESCRIPTION
	Permet d'accéder aux fichiers d'Administration (Conventions, Rubriques Facturation).
	Permet d'accéder aux fichiers de référence (Armateur, Agent, Navire, etc.).
	Permet la saisie des manifestes D1 des articles à mettre en Zone Sous Douane.
	Permet la gestion des entrées /sorties des conteneurs dans la Zone Sous Douane.
	Permet la facturation automatique des conteneurs livrés de la Zone Sous Douane.
	Offre des fonctions de maintenance et de sauvegarde des données du progiciel.
	Permet de visualiser la situation de la Zone avec une vue en trois dimension (réelle).
	Offre des fonctions de transfert et de consolidation des données du progiciel.
	Offre au gestionnaire de la Zone, des statistiques concernant le Parc.

	Offre à l'utilisateur de la Zone, l'aide contextuelle concernant le Progiciel.

Tableau 4 : Description des modules du progiciel ZSD

III. Représentation de ZSD sous forme de processus

Nous allons représenter dans cette partie ZSD en utilisant l'approche par processus, on commencera par une cartographie du macro processus (N) de tout le logiciel pour passer par la suite à la schématisation d'un processus élémentaire (N-2) de ZSD qui est la facturation (Module Caisse) comme suit :

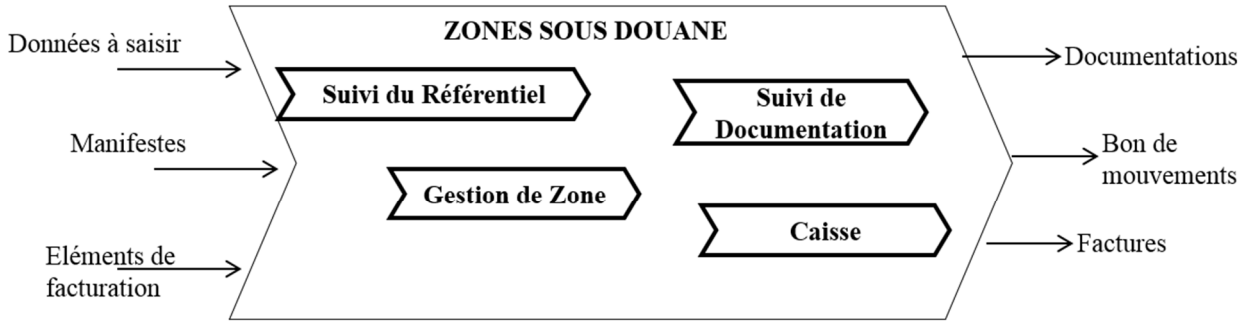


Figure 13 : Cartographie macro-processus « N » de Zones Sous Douane (ZSD)

La shématisation de ZSD on ce basent sur l'approche par processus,sous forme de macro-processus nous a permis d'identifier :

- Les entrées suivantes :
  - Données à saisir
  - Manifestes
  - Eléments de facturations
- Les sorties suivantes :
  - Documentations
  - Bon de mouvements
  - Factures

❖ Les opérations de transformation des entrées en sorties répartie chacune dans les modules du progiciel en respectant un enchainement précis comme suit :

Dés que on a comme entré les manifeste , les éléments de facturation et les données à saisir on fait appel au logiciel ZSD pour la gestion Armateur, Agent, Navire…etc. assuré par module de suivi référence, la gestion des entrées /sorties des conteneurs assumé par le module de gestion de zones, la saisie des manifestes D1 des articles a classé dans Zone Sous Douane et enfin l'établissement des facture effectué par le module de facturation.

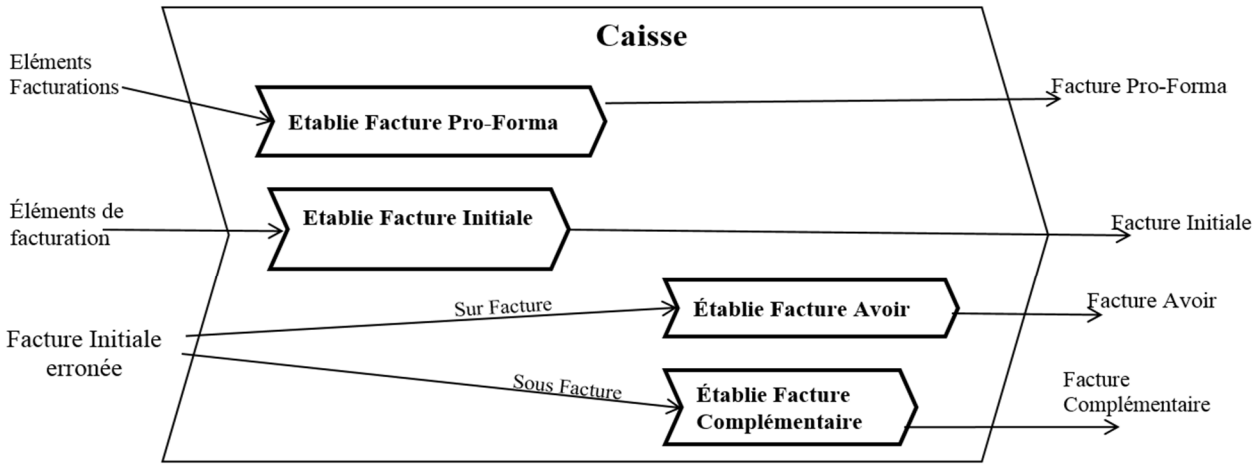


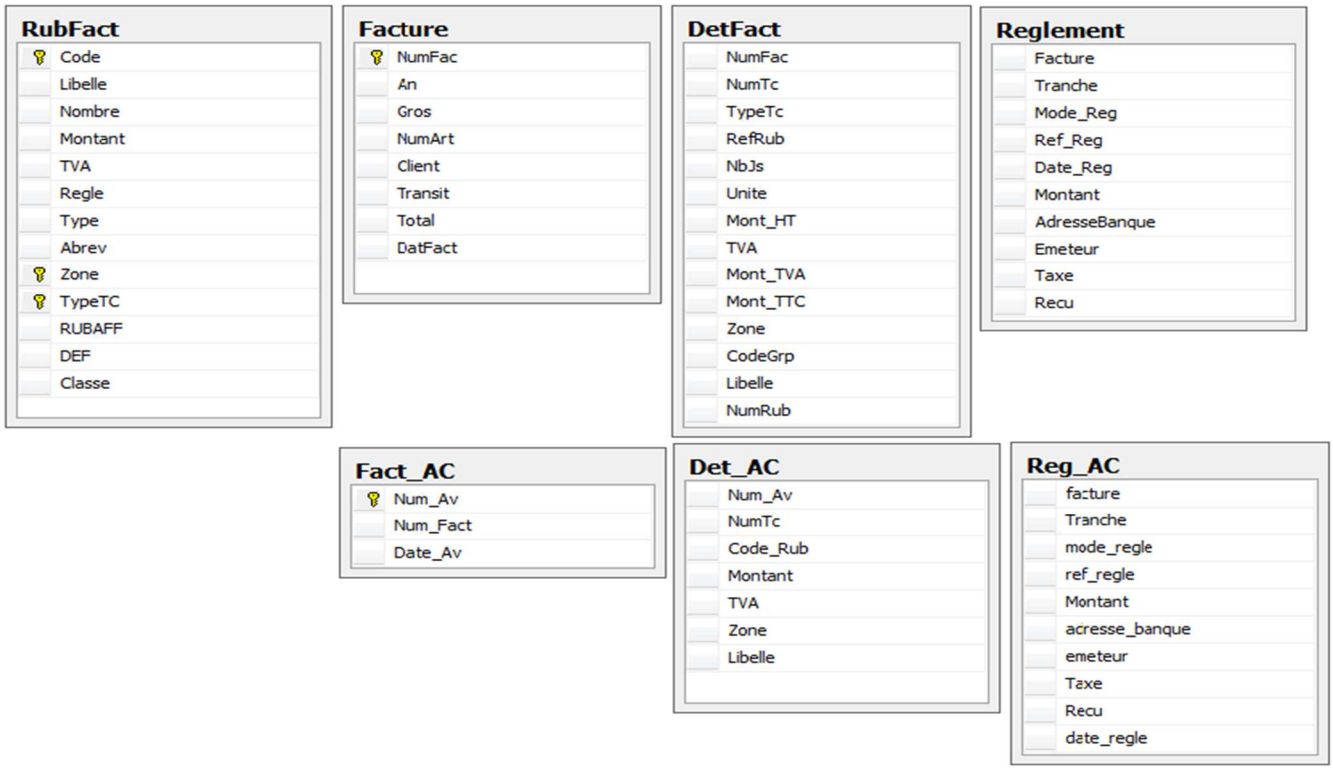
Figure 14 : Cartographie processus élémentaire « N-2 » du module caisse de ZSD.

IV. Les tables de la base de données

A ce niveau de l'étude la représentation de la base de données de ZSD sera restreinte à celles des tables concernant la facture Client et Armateur:

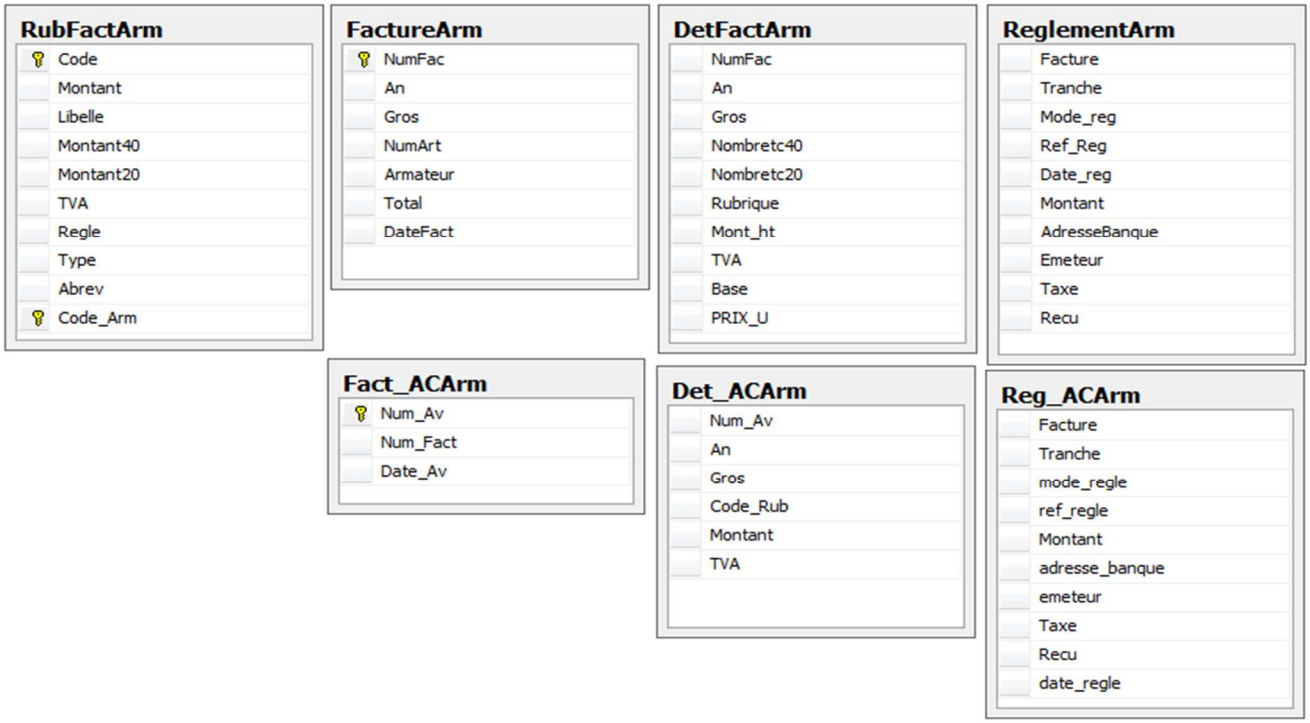
i. Les tables de la Facture Client

Le schéma suivant représente les tables concernant la facture Client:



ii. Les tables de la Facture Armateur

Le schéma suivant représente les tables concernant la facture Armateur:



C. Présentation de MSGMAN

I. Marine Soft de Gestion de la Manutention (MSGMAN)

MSGMAN «Marine Soft Gestion de la Manutention » est un progiciel qui s'adresse aux manutentionnaires, il permet d'automatiser du début de la chaîne de traitements jusqu'à la fin, les opérations de gestion quotidienne assuré par les module sur lesquelles ils sont répartie.

II. Les rôles des modules de MSGMAN

Dans le but d'avoir une idée générale sur les opérations de gestion assurée par les modules qu'intègre MSGMAN, on a pensé à présenter sous forme de tableau les modules que compte le logiciel comme suit :

MODULES	DESCRIPTION
	Permet d'accéder aux fichiers permettant le paramétrage de tout le progiciel.
	Permet d'accéder aux fichiers de bases (navires, agents, type navire, … etc.).
	Permet de pointer les ouvriers.
	Permet l'établissement des factures : Extra-frais, transfert, pointeurs, location-grues, débarquement et embarquement. Les factures avoirs complémentaires, l'édition de l'état de caisse, journal des factures… etc.
	Production et exploitation des états statiques permettant le bon suivi des opérations de traitement des navires.
	Permet l'échange informatisé des données entre le progiciel MSGMAN et les autres logiciels GESPort, … etc.

Tableau 5 : Description des modules du progiciel MSGMAN

III. Représentation de MSGMAN sous forme de processus

Nous allons représenter dans cette partie MSGMAN en utilisant l'approche par processus, on commencera par une cartographie du macro processus (N) de tout le logiciel pour passer par la suite à la schématisation d'un processus élémentaire (N-2) de MSGMAN qui est la facturation (Module Caisse) comme suit :

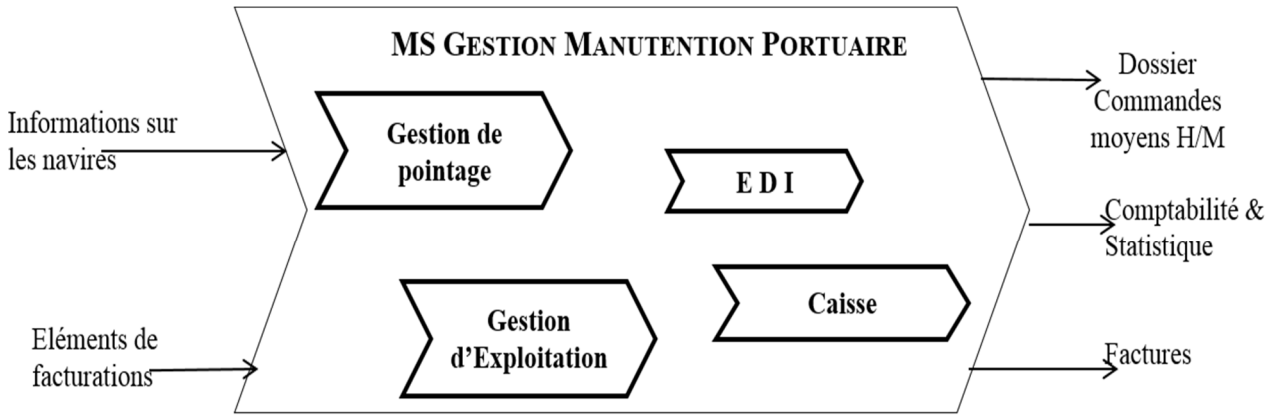


Figure 15 : Cartographie Macro-processus « N » de Gestion de Manutention Portuaire (MSGMAN)

La shématisation de MSGMAN on ce basent sur l'approche par processus,sous forme de macro-processus nous a permis d'identifier :

- Les entrées suivantes :
  - Informations sur les navires ,
  - Elements de facturation,
- Les sorties suivantes :
  - Dossier commandes moyens H/M
  - Comptabilité & statistique
  - Factures
- Les opérations de transformation des entrées en sorties réparties chacune dans les modules du progiciel en respectant un enchainement précis comme suit :

Dés que les informations sur les navires et elements de facturations arrive, on fait appel au logiciel MSGMAN pour le bon suivi des opérations de traitement des navires assuré par module gestion d'éxploitation, pointer les ouvriers assumé par le module de gestion des

poitage , l'établissement des facture effectué par le module de facturation enfin envoyé ces facturé à la comptabilité grâce à la passerelle de comptabilité.

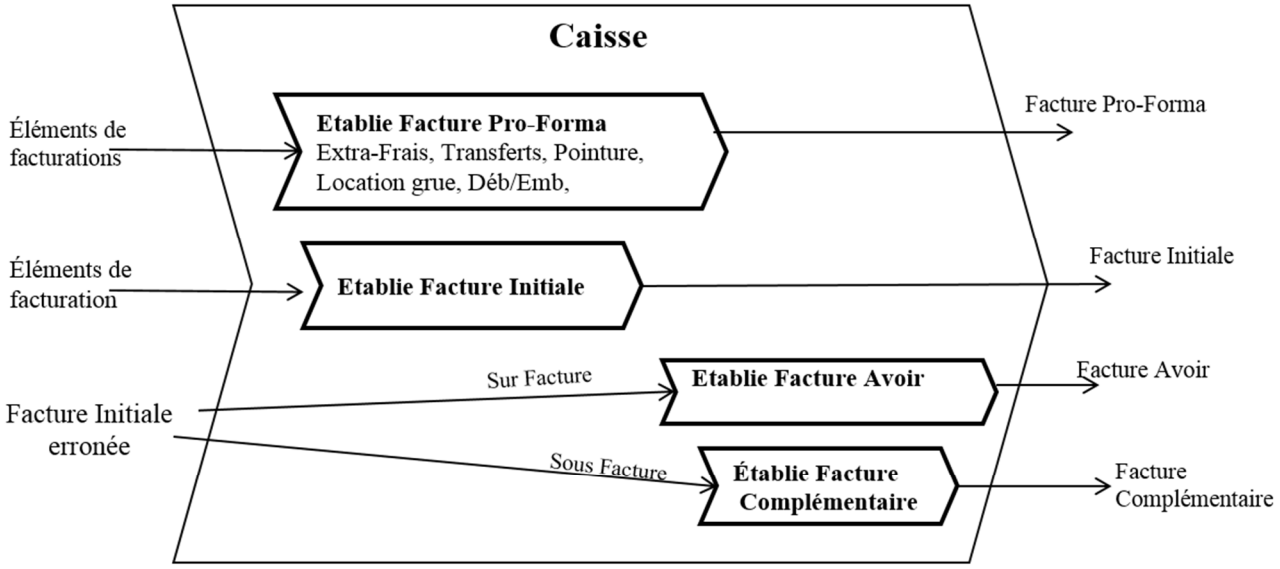


Figure 16 : Cartographie processus élémentaire « N-2 » du module caisse de MSGMAN.

IV. Les tables de la base de données

A ce niveau de l'étude la représentation de la base de données de MSGMAN sera restreint à celles des tables du module facturation schématisé ci-dessus :

Ligne_Facture
NFACTURE
PRESTATION
JOUR
OPERATION
NOMBRE
QUANTITE
TARIFU
MONTANT
UNITE
TVA
CODEPREST
CODEDET
CODE2
RED_MAJ
CUMULMT
CUMULTVA
ODRE
CODE_PREST

Facture
NFACTURE
DATEF
CODE_ARM
NUMDOS
TAXEFIXE
MHT
TVA
TIMBRE
MTTC
TYPEF
TYPER
ANNUL
DATEA

Dossier(Escale)
NUMDOS
CODE_ESC
CODE_NAV
GROS
VOYAGE
ETA
RADE
ACCOSTAGE
TERME_TRANSP
ETS
DEPART
NBRCALE
TIRAN_AR
CLOSOP
DEBUTOP
FINOP
CODE_CMD
FACTURE
MARCH_IMP
MARC_EXP
NUM_BROUIL
NFACTURE

Detail_Brouil
NUM_BROUIL
CODE
CODE_CONDIT
UNITE
NOMBRE
QUANTITE
JOUR
OPERATION
TARIF
TYPE_MARCH
RED_MAJ
TVA
MONTANT
FORF
TENG
TENG
CODEM
LIBELLE
ORDRE
NUM_BROUIL

Brouil
NUM_BROUIL
FTYPE
TYPE
NUMDOS
CODE_AG
CODE_ARM
NUM_CONV
SOURCE_CONV
TYPEB
TYPER
NUM_BROUIL
CODE
CODE_CONDIT
ORDRE

Avoir_complémentaire
NFACTURE
TAXEFIXE
MHT
TVA
TIMBRE
MTTC
DATEF
TYPE
NUM_AV
CUMUL

Reglement
NFACTURE
TRANCHE
CODE_MODEP
REF_REGLE
DATE_REGLE
MONTANT
ADR_BANQUE
EMETTEUR
TAXE

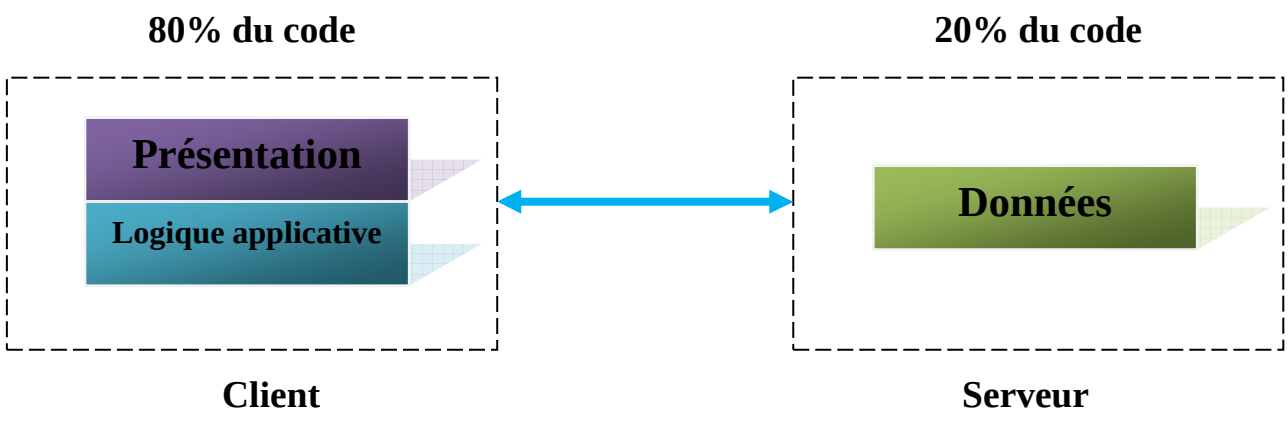
Det_avoir
NUM_AV
PRESTATION
MONTANT
TVA
CODEPREST
CODEDET
CODE2
MTTC
ORDRE
NUM_AV

Prestation
CODE_PREST
LIBELLE_PREST
TYPE
ABREV
NUM_BROUIL
CODE
CODE_CONDIT
ORDRE

Reglement_avoir
NUM_AV
TRANCHE
CODE_MODEP
REF_REGLE
DATE_REGLE
MONTANT
ADR_BANQUE
EMETTEUR
TYAXE

D. Architecture et répartition de code

Les produits analysés s'appuient sur l'architecture client/serveur 2tiers, leur code est réparti comme suit :



3.3. Bilan du diagnostic

La phase précédente a pour objectif, d'analyser la situation existante afin d'identifier dans le cadre d'un projet de conception et réalisation d'un nouveau système d'information, les différents anomalies et les axes d'amélioration. Dans cette perspective, il est important de bien élaborer un bilan critique qui fait comprendre le contexte et les enjeux sur lequel s'appuiera la nouvelle conception.

Dans le cadre de la structuration du bilan de l'existant, nous avons convenu de s'appuyer sur le diagramme d'ISHAKAWA, qui est un outil d'analyse et de cause à effet qui est une représentation structurée de toutes les causes qui conduisent à une situation.

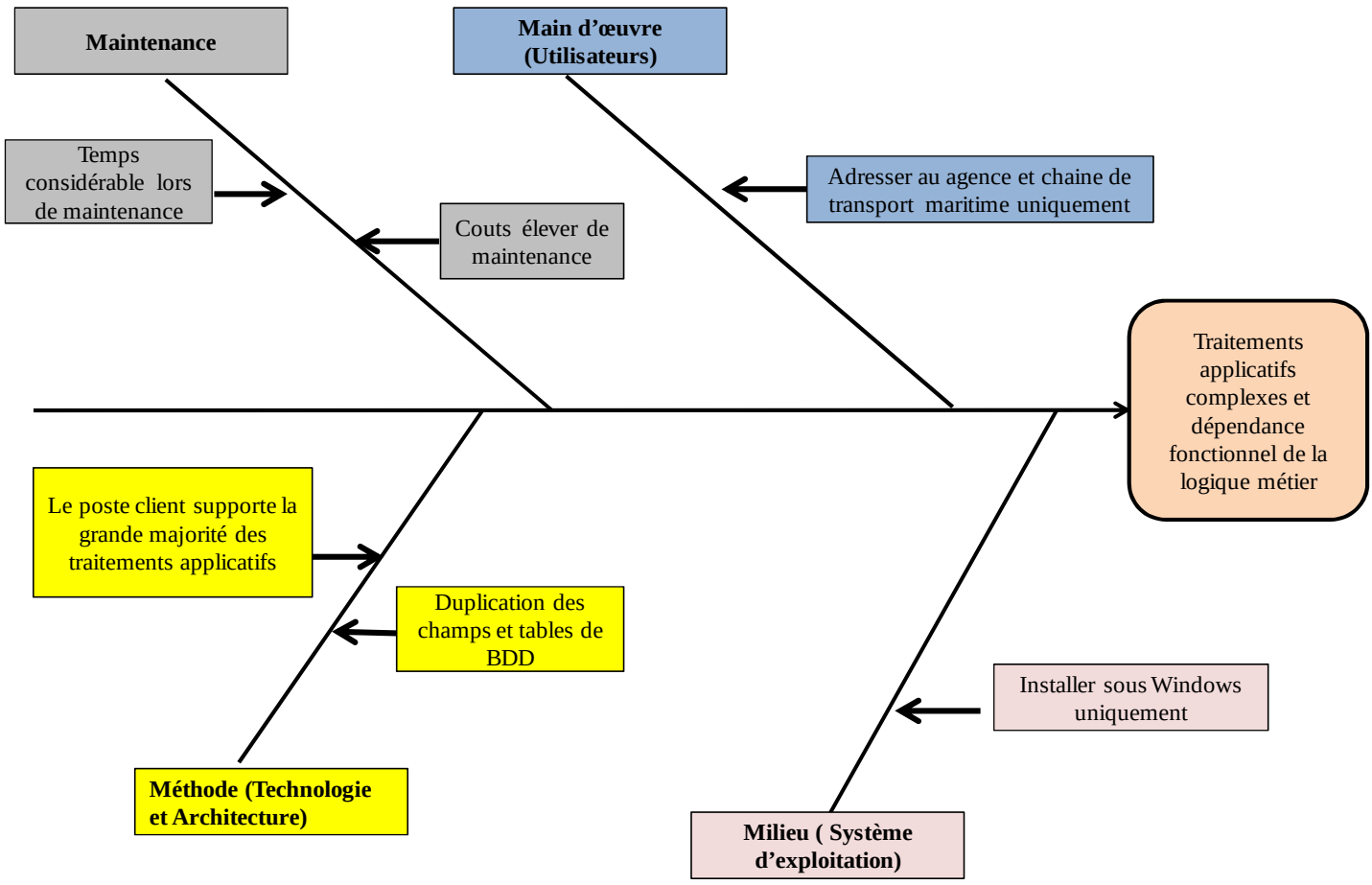


Figure 17: Diagramme d'Ichikawa du système existant

Notre diagnostique a atteint les systèmes d'informations de marine soft sur plusieurs volets à savoir :

a) L'architecture

L'architecture adoptée par marine soft en occurrence client-serveur 2tiers, a impacté ses systèmes d'informations par les limites de cette architecture à savoir :

- Son manque de modularité lorsque toute la logique d'application se trouve sur le poste client ne facilite pas la maintenance et la réutilisation : une modification de l'application ou de la structure de la base de données nécessite un redéploiement sur les postes clients.
- Rigidifie par les couts et la complexité de la maintenance

Ce diagramme montre les coûts de développement et de maintenance (l'unité en dollars) par rapport à la complexité et la durée de vie des applications. [Gartner group].

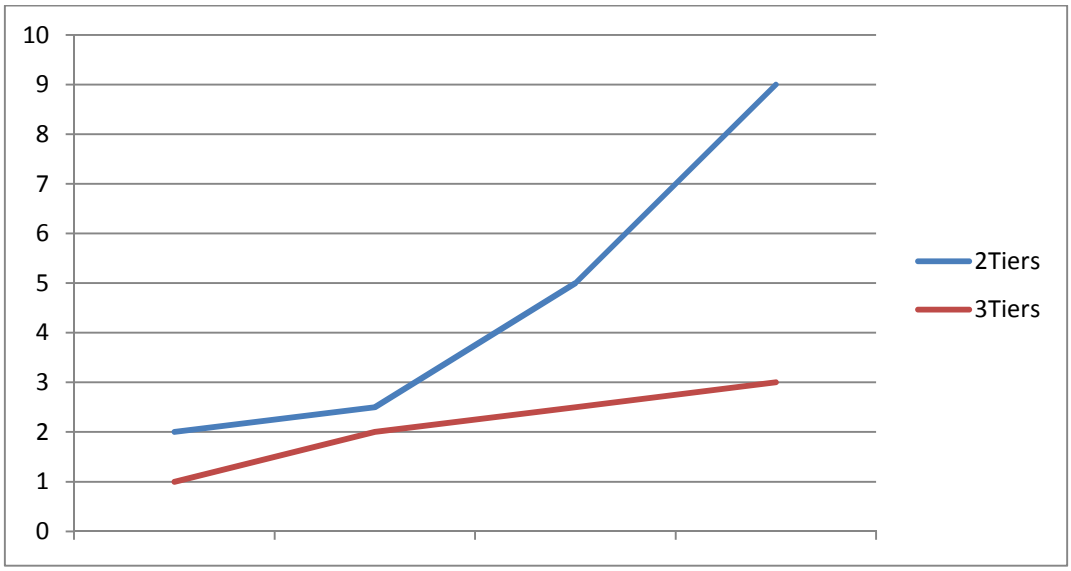


Figure 18: Diagramme comparative des deux architectures 2 et 3 tiers

En plus, les applications sont supportées par un seul système d'exploitation qui est le Windows, ce qui impose au client l'installation de ce système.

b) L'utilisation

Des applications destinées à des clients précis :

Les applications développés en sien de marine soft sont utilisable, par les intervenants de la logique et la chaîne de transport maritimes.

c) Le fonctionnement

➢ La dépendance du système de facturation de la logique métier :

Le module facturation qui existe dans toutes les applications dépond du système d'informations qui l'intègre et sa se constate dans les tables de la base de données.

➢ Redondance des champs et des tables :

Il y a plusieurs champs et tables qui signifient la même chose mais qui sont appelés différemment.

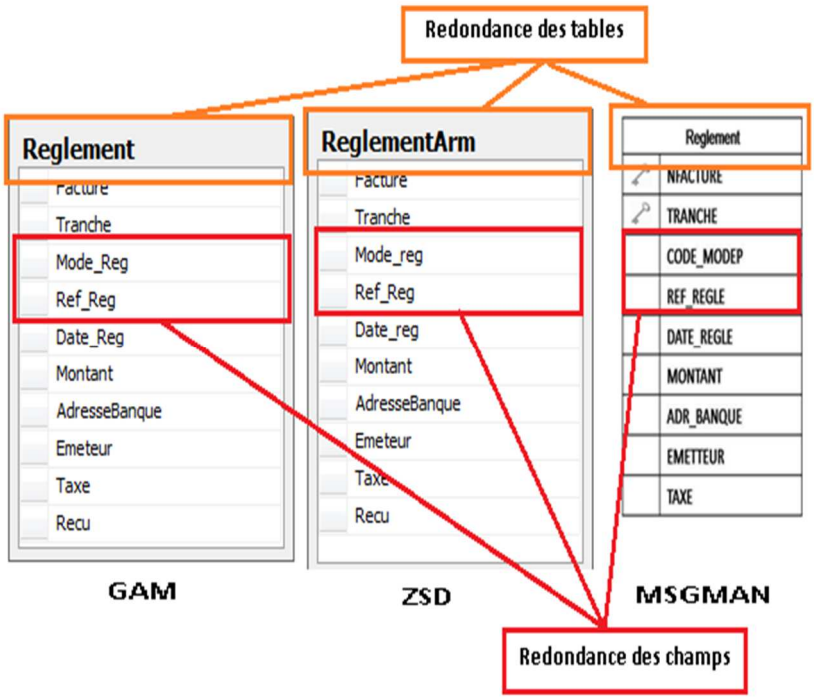


Figure 19 : Présentation de la redondance des champs et des tables.

3.4. Présentation de la solution informatique

Après avoir fait le constat des différents systèmes d'informations existants, et pour atteindre les objectifs tracés auparavant dans le sujet de notre travail, le STD_fct qui est un système d'informations traitant la facturation et qui se base sur deux principes cités dans notre thème à savoir :

1. La facturation découlant de l'exploitation du métier qui l'intégrait.
2. La facturation standard (les éléments de taxation ne dépendent d'aucun métier).

Le grand défi est de séparer les fonctionnalités de facturation de tous ce qui reste et de les fournir sous forme standard qui sera intégré dans tous les progiciels de marine soft existants, même dans un ERP /CRM et systèmes d'informations sur le marché.

3.4.1. Présentation d'ERP GLS[2]

ERP GLS, *Enterprise Resource Planning Global Logistic Software*, solution qui se traduit simplement en un progiciel qui est destiné à la logistique du transport maritime et intègre les principales composantes fonctionnelles du l'entreprise. Etant un système unifié, les utilisateurs de différents métiers travaillent dans un environnement applicatif identique. Ce modèle permet d'assurer l'intégrité des données, la non-redondance de l'information, ainsi que la réduction des temps de traitement.

Rappel

La conception et la réalisation du système de facturation qui, nous a été confié est **le noyau** de l'***ERP GLS,*** ce qui justifier son importance et sa complexité de réalisation.

3.4.2. Description de la solution

1. Vers une architecture 3tiers

Le nouveau système d'information doit respecter l'architecture client/serveur 3tiers, cette architecture donne une autre dimension pour les produits de l'entreprise :

La mise en œuvre d'une architecture " 3tiers " nécessite sur l'ensemble des plates-formes des logiciels génériques, appelés globalement " Middleware ", qui fournit les mécanismes techniques supportant les échanges d'information entre Clients et Services.

Le " Middleware " se définit comme l'ensemble des logiciels mis en œuvre " en dessous des applications, au-dessus des O.S. et entre les plates-formes " pour assurer les échanges entre les composants Présentation, Traitement et Accès aux données dans les architectures Client/serveur.

[2] ERP GLS est présenté d'une manière détaillée dans le premier chapitre « Présentation général »

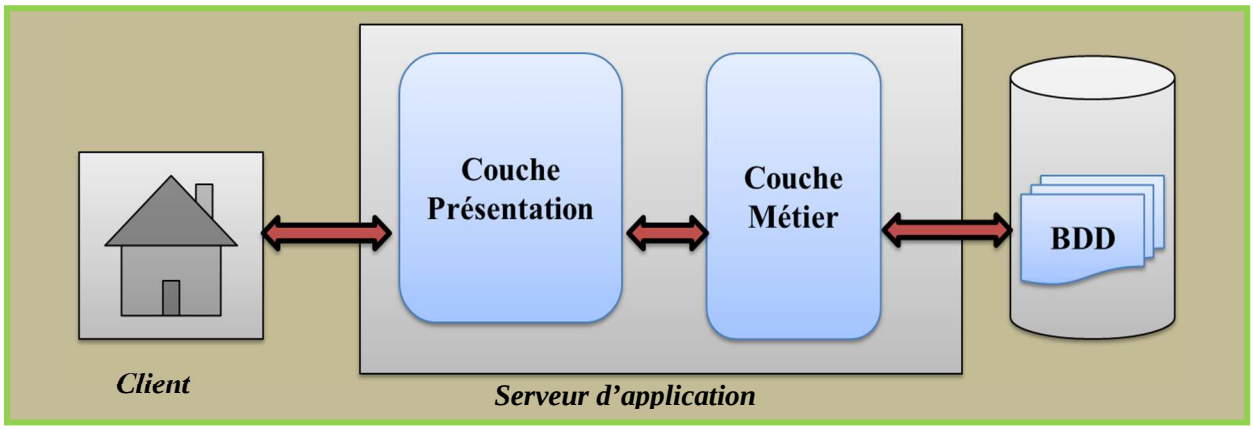


Figure 20: Architecture 3tiers

2. La Séparation de la logique métier

ERP GLS élimine la dépendance de la logique métier de la présentation (client léger) pour permettre la standardisation des modules de l'ERP. Ce qui donnera à STD_fct une indépendance des modules de l'*ERP GLS*.

3. La centralisation de base de données

L'argument convaincu de STD_fct coté base de données est la possibilité de modéliser et centralisée une BDD (élimine la duplication des tables de facturation et on plus élimine la redondance des champs).

4. La simplicité de la maintenance

L'analyse des systèmes existants a montré que les coûts et le temps de maintenance est très élevé, d'où l'obligation de concevoir un système de facturation standard qui permet en cas de modification de maintenir une fois seulement (sur le serveur d'application).

STD_fct est un projet d'une démarche claire : qui doit être conçu pour respecte la règle des 3S suivantes :

- Simple à installer
- Simple à utiliser
- Simple à maintenir

Il sera basé sur les fonctionnalités déjà existantes dans les produits Marine de Soft, et il doit répondre à des besoins bien définis :

- Gestion des factures : établir, éditer et le suivi d'une factures,
- Gestion des comptes client : ajouté, modifié et supprimer des comptes ainsi que la

gestion des versements sur le compte.

- Gestion des règlements : effectuer et lister les règlements des factures

Avec possibilité de l'exploitation vers d'autres systèmes, ainsi que la séparation des fonctions techniques. Il se caractérise de plus par sa simplicité d'utilisation et sa facilité de maintenance.

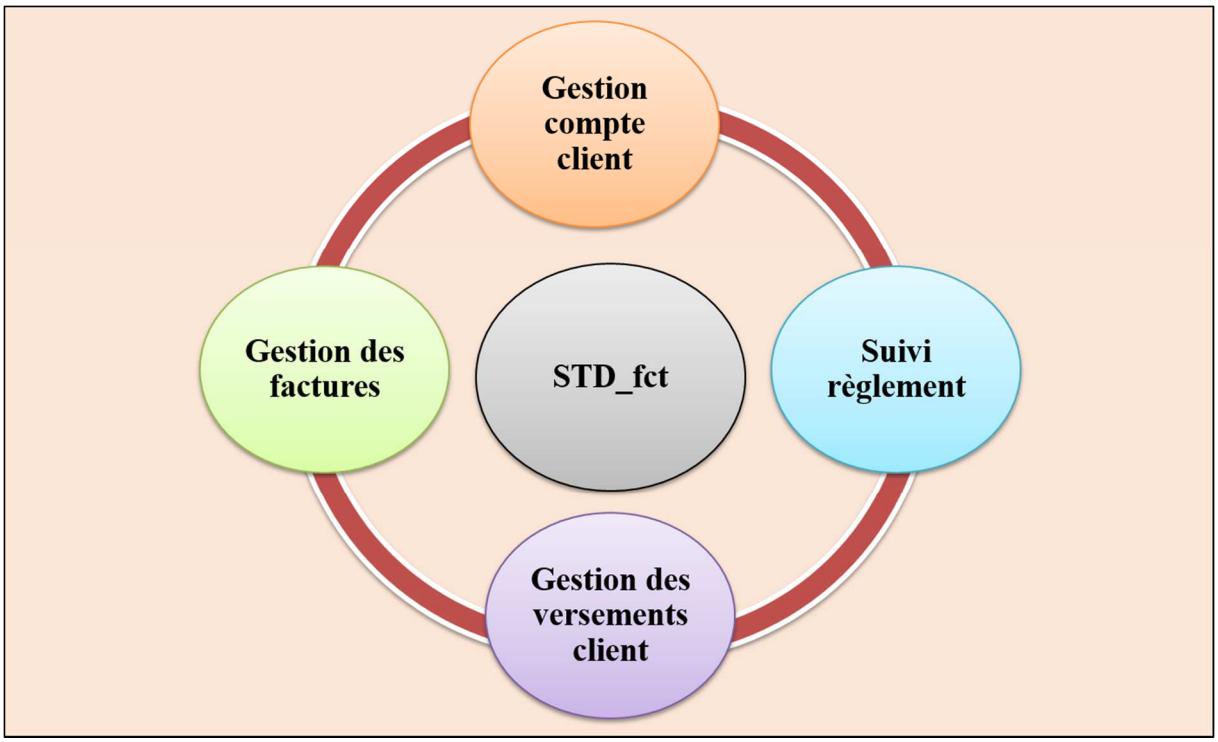


Figure 21 : Les fonctionnalités principales de STD_fct

Conclusion

Les lacunes et les anomalies constatés après avoir analysé les applications existantes de Marine Soft, que ce soit architecturelles ou fonctionnelles, à prouver la nécessité d'un système de facturation standard qui répond aux exigences de l'*ERP GLS.*

L'émergence de ce nouveau système dans un CRM, qui est un ensemble d'outils et techniques permettant de gérer la relation avec la clientèle en automatisant les différentes composantes de la relation client, permet de bénéficie des privilèges de ce dernier.

Cette solution ne pouvant pas se réaliser sans la mise en place d'un système d'information qui répondra aux besoins des futurs utilisateurs. Cet aspect fera l'objet du prochain chapitre.

CHAPITRE 3

[CONCEPTION DU FUTUR SYSTEME]

Introduction

Afin d'aboutir au développement de meilleures applications, il est nécessaire d'avoir une bonne maîtrise et organisation du travail et donc suivre une démarche méthodologique rigoureuse. Pour cela le choix d'un formalisme de conception est d'une très grande importance.

Pour le développement de notre application nous avons opté pour une démarche de conception orientée objet, en nous basant sur la modélisation UML.

La démarche s'articule autour de deux étapes : l'analyse des besoins et la conception.

- ✓ En phase d'analyse des besoins, nous mettons en évidence les exigences et les acteurs du futur système.
- ✓ En phase de conception, nous présentons les descriptions détaillées des résultats de l'analyse.

La figure suivante montre la représentation graphique de la démarche de modélisation que nous avons choisie pour concevoir notre application.

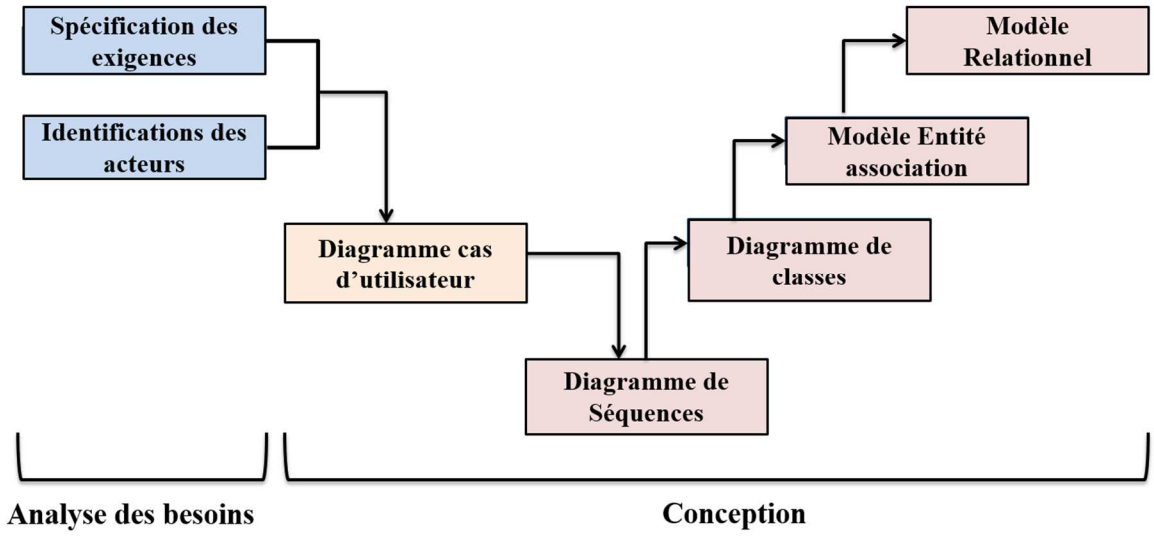


Figure 1 : La représentation de la démarche de modélisation.

1. Méthodologie : UML

UML est née de la collaboration de trois (3) experts (Rumbaugh, Booch et Jacobson) qui ont focalisés leurs attentions sur les deux aspects : modélisation et formalisation afin de concevoir un langage de modélisation standard et universel utilisé notamment pour le développement informatique en langage objet.

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.

UML unifie à la fois les notions et les concepts orientés objet. Il ne s'agit pas d'une simple notation, mais les concepts transmis par un diagramme avec une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage. UML a une dimension symbolique et ouvre une nouvelle voie d'échange de visions systématiques précises. Ce langage est certes issu de développement logiciel mais pourrait être appliqué à toute science fondée sur la description d'un système. Dans l'immédiat, UML intéresse fortement les spécialistes de l'ingénierie système. *[Pascal Roques, UML 2 modéliser une application web]*

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 spécification jusqu'au codage.

2. Analyse des besoins

Dans cette section nous détaillerons dans un premier temps les exigences fonctionnelles du futur système "STD_fct", à savoir les fonctionnalités requises par l'utilisateur : établissement d'une facture, règlement d'une facture, rechercher une facture...etc. Nous ajouterons ensuite des exigences non fonctionnelles (performances…etc.), enfin nous identifierons les acteurs utilisant l'application pour nous placer dans l'optique du démarrage d'un projet.

2.1. Spécification des exigences

2.1.1. Exigences fonctionnelles

Il s'agit de besoins spécifiques qui décrivent les caractéristiques du système ou des processus que le système doit exécuter. On trouve dans cette catégorie les règles métier.

Partant de ce principe nous sommes arrivés dans le cas de notre application aux exigences suivantes :

➢ L'application doit pouvoir :

- Communiquer avec le logiciel de comptabilité.
- Etre intégrer comme module dans d'autres logiciels tels qu'un ERP.
- Importer le dossier de chaque client c'est-à-dire tout ce qui est entête, élément métier, le détail du dossier (Quantité, Montant HT, TVA, libelle (désignation)).

➢ L'application doit permettre à l'utilisateur de faire :

- Gestion des factures
  - Établir une facture (pro forma, initial, complémentaire/avoir).
  - Éditer une facture (pro forma, initial, complémentaire/avoir).
  - Rechercher des factures selon différents critères (type facture, entre deux date, numéro de facture, état de facture) et pouvoir accéder à un résultat sélectionner.
  - Suivre les factures impayé (initiale/complémentaire).
  - Effectuer le règlement d'une facture (initiale, avoir/complémentaire).
  - Consulter le total de l'argent cumulé dans une période spécifique (entre deux dates).
  - Possibilité de classé les rubriques par famille et/ou par groupe dans une facture.
  - Classé les facture avec un état (facture à crédit/ facture comptent) selon que le client soit conventionné ou non.

- Gestion des comptes client :
  - Ajouter un compte client.
  - Supprimer un compte client.
  - Modifier un compte client.
  - Rechercher des comptes clients selon différents critères (code client, code versement, entre deux dates) et pouvoir accéder à un résultat sélectionné.
  - Gérer les versements d'un compte client
    - Ajouter un versement.
    - Modifier un versement.
    - Supprimer un versement.
    - Rembourser les avances.

- Gestion des règlements :
  - Effectuer le règlement en un ou plusieurs tranche(s) chacun avec un mode de payement spécifique (espèce/versement du compte client /chèque) des deux types :

- Au comptant.
- A crédit (suivi des règlements).

- Rechercher par critères (type facture, entre deux dates, numéro de facture, état de facture, code client, code versement):
  - De facture(s) trier par ET/OU
    - Date de création.
    - Numéro de facture.
    - Type de facture (initiale, avoir, complémentaire, pro forma).
    - Etat (en comptent / a crédit).
  - De compte client (s) trier par ET/OU
    - Date de création.
    - Code client.
    - Nom du client.
  - De règlement (s) trier par ET/OU
    - Mode de règlement.
    - Type de règlement.
    - Client.
    - Date de règlement.

➢ L'application doit permettre à l'administrateur de faire :

- Gérer les comptes utilisateurs:
  - Ajouter un compte utilisateur.
  - Supprimer un compte utilisateur.
  - Modifier les accès d'un utilisateur.
  - Rechercher les utilisateurs.
- Paramétrer le système :
  - Définir les rubriques, la famille et groupe aux quelle elles appartiennent.
  - Ajouter des rubrique(s)/famille(s)/groupe(s).
  - Modifier les rubriques, les familles, les groupes.

2.1.2. Exigence non-fonctionnelles

Il s'agit d'exigences qui ne concernent pas spécifiquement le comportement du système. Elles identifient des contraintes internes et externes du système. Elles doivent avoir des valeurs quantitatives.

Dans le cadre de ce travail, l'application devra respecter les exigences suivantes :

- **Portabilité:** fonctionne sur plusieurs plateformes (Utilisable avec plusieurs systèmes d'exploitation).

- **Modificabilité:** ajout de nouvelles fonctionnalités.
- **Réutilisabilité:** de composantes, de code, de designs, et même d'exigences dans d'autres systèmes.
- **Maintenabilité:** modifications de fonctionnalités, corrections.
- **Compréhensibilité:** conception, architecture et code facile à comprendre/apprendre.
- **Intégrabilité:** facilité à intégrer des composantes.
- **Performance:** Temps d'attente.
- **Utilisabilité :** Capacité pour un utilisateur d'exécuter une tâche dans un temps donné après une formation d'une durée déterminée.
- **Efficacité:** usage minimal de ressources: mémoire, processeur, etc.
- **Fiabilité:** des calculs, de la précision.
- **Sécurité:** Accès personnalisés, connexions sécurisées.
- **Adaptabilité:** à d'autres environnements ou problèmes.
- **Passage à l'échelle:** pour grandes quantités ou données.

2.2. Spécifications des acteurs

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit avec un système. [*LA 15*]

Les acteurs humains identifiés pour notre application sont les suivants :

- L'utilisateur : il s'agit de toute personne s'étant identifié/authentifié sur l'application. Il aura donc en sa possession un login et un mot de passe. Il s'agit de l'acteur principal qui dispose des fonctionnalités auquel on lui a donné l'accès dans le cadre de son travail.
- L'administrateur : il s'agit de la personne chargé d'administrer l'application dont les fonctionnalités sont restreintes à la gestion des comptes utilisateur (login, mot de passe) et au paramétrage du système selon le besoin métier.

Remarque

L'administrateur peut jouer le rôle d'un utilisateur.

3. Conception

3.1. Modélisation fonctionnelle

Après avoir identifié les acteurs qui interagissent avec le système, nous y développerons un premier modèle UML de haut niveau, pour pouvoir établir précisément les frontières du système. Dans cette optique, nous allons identifier les cas d'utilisation et construire un diagramme reliant les acteurs et les cas d'utilisation.

Ensuite, nous précisons le point de vue fonctionnel en détaillant les différentes façons dont les acteurs peuvent utiliser le système. À cet effet, nous allons rédiger des descriptions textuelles de cas d'utilisation, ainsi que dessiner des diagrammes UML complémentaires comme les diagrammes de séquence.

3.1.1. Diagramme de cas d'utilisation

Un cas d'utilisation (use case) représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier. **[***LA 15***]**

Dans le cadre de notre projet nous sommes arrivés aux cas d'utilisations suivants:

a. Pour un utilisateur :
- Etablir une facture (pro forma/initiale/avoir/complémentaire),
- Effectuer le règlement,
- Effectuer le suivi règlement,
- Etablir reçu de paiement,
- Gérer les comptes client,
- Editer,
- Effectuer une recherche multicritère d'une facture/ règlement/ comptes client,

b. Pour un administrateur :
- Paramétrer le système,
- Gérer les utilisateurs,

c. Pour utilisateur/administrateur:
- S'authentifier,

Les cas d'utilisation et acteurs recensé ci-dessus nous permettrons de construire les diagrammes de cas d'utilisations qui montre les interactions fonctionnelles entre les acteurs et le système à l'étude comme suit :

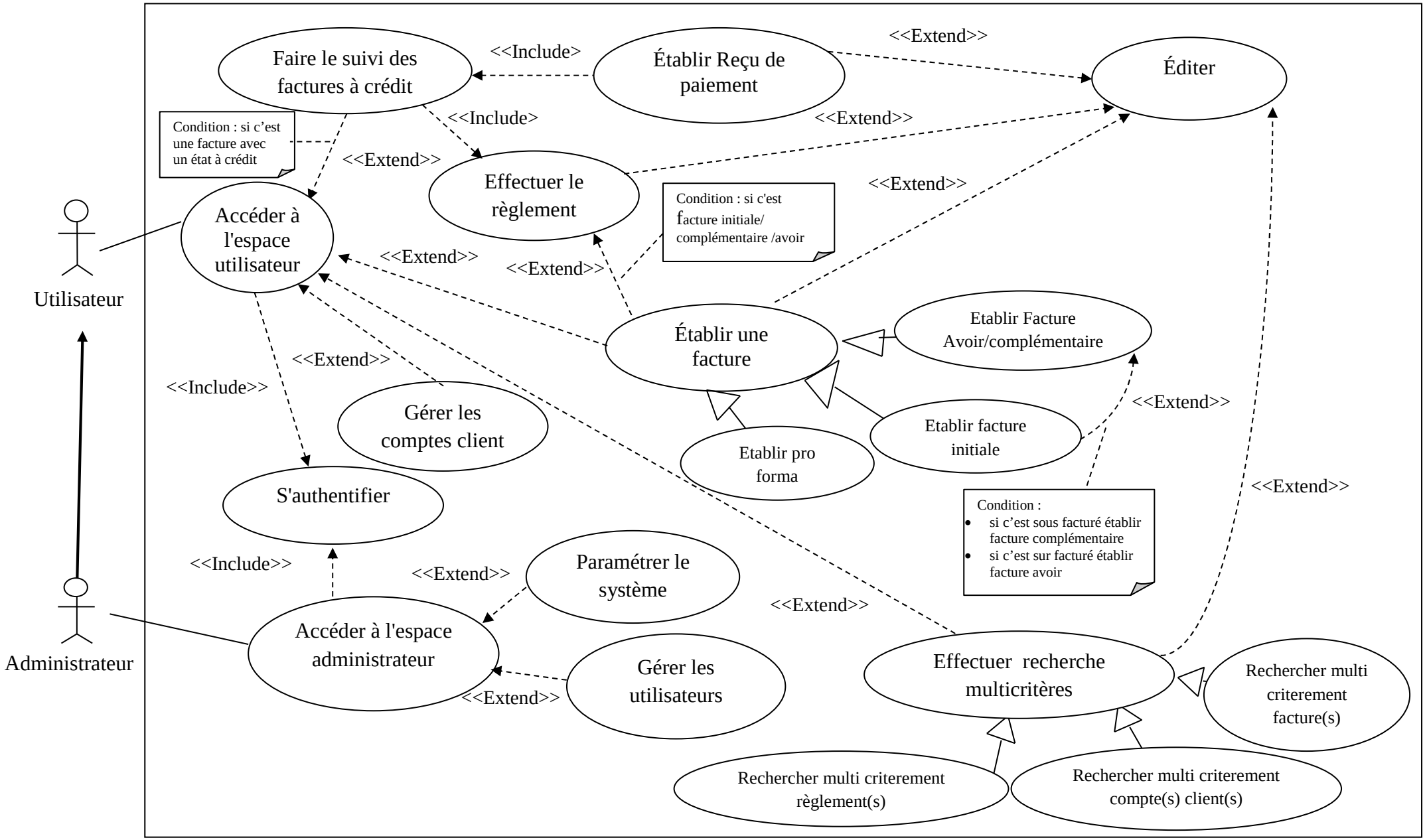


Figure 2 : Diagramme de cas d'utilisation globale

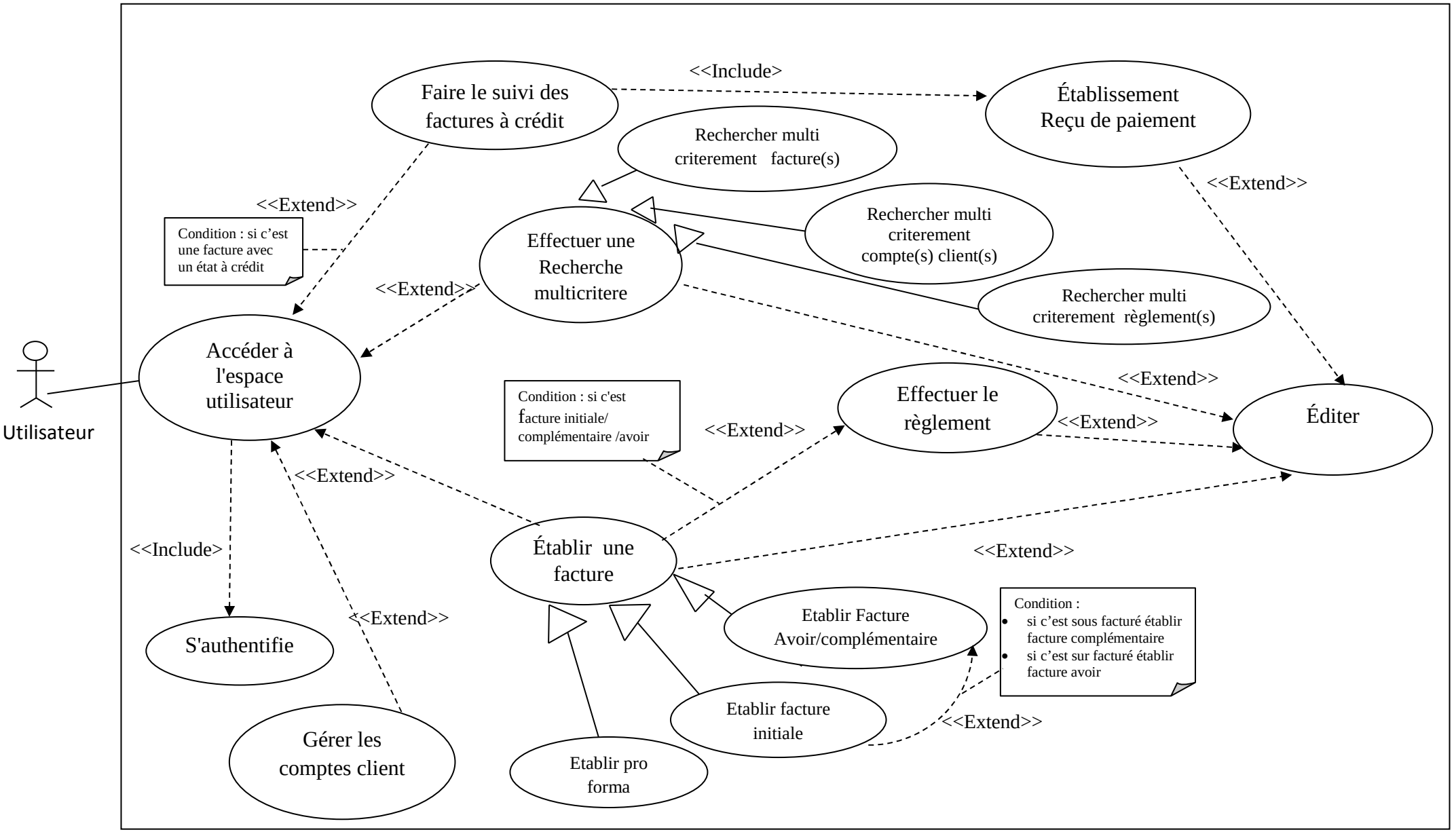


Figure 3: Diagramme de cas d'utilisation globale de l'utilisateur

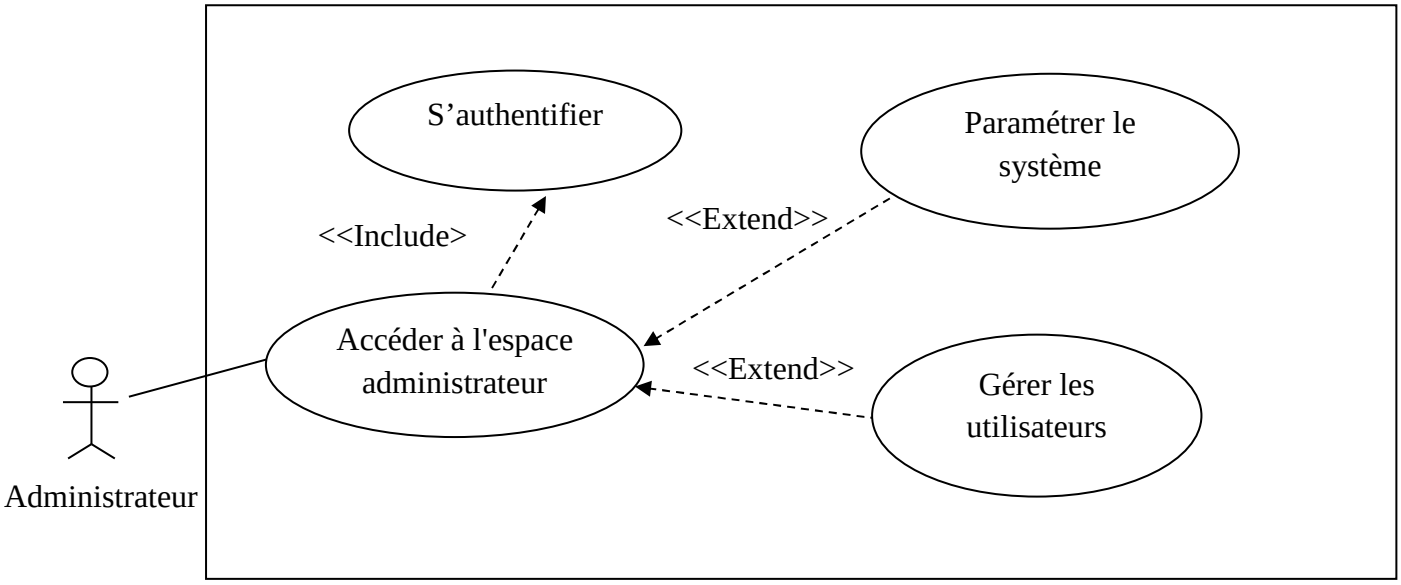


Figure 4 : Diagramme de cas d'utilisation globale de l'administrateur

3.1.2. Spécifications détaillés d'un cas d'utilisation

I. Spécification des taches

Chacun des acteurs que nous avons définis précédemment, effectue un certain nombre de taches qu'on résume dans le tableau suivant :

Acteur	Taches
L'utilisateur	T1: S'authentifier. T2: Accéder à l'espace utilisateur. T3: Gérer les comptes clients. T4: Établir une facture (pro forma /initiale /avoir /complémentaire,). T5: Effectuer le règlement d'une facture (initiale/avoir/complémentaire). T6: Effectuer le suivi règlement d'une facture à crédit (initiale/complémentaire) T7: Effectuer une recherche multicritère de facture(s)/ règlement(s)/ compte client(s). T8: Éditer une facture (pro forma/initiale/avoir/complémentaire), un reçu de paiement, la liste des résultats d'une recherche (facture/ règlement/compte client). T9: Se déconnecter
L'administrateur	T10: T1 T11: Accéder à l'espace administrateur T12: Gérer les utilisateurs T13 : Paramétrer le système. T14: T9.

Tableau 1 : La liste de tâches effectuées par les acteurs

II. Spécification dynamique (Description textuelle d'un cas d'utilisation, Diagramme de séquence)

Pour détailler la dynamique du cas d'utilisation, nous allons recenser de façon textuelle toutes les interactions entre les acteurs et le système. Le cas d'utilisation aura un début et une fin clairement identifiés. Nous préciserons également les variantes possibles, telles que le cas nominal, les différents cas alternatifs et d'erreur, tout en essayant d'ordonner séquentiellement les descriptions, afin d'améliorer leurs lisibilités.

Remarque :

Pour documenter les cas d'utilisation, la description textuelle est indispensable, mais le texte ne montrant pas comment les enchaînements se succèdent. En outre, la maintenance des évolutions s'avèrera souvent fastidieuse. Il est donc recommandé de compléter la description textuelle par un ou plusieurs diagrammes dynamiques UML telle que le diagramme de séquence détaillé que nous allons voir succédé chaque description textuelle choisi. [*LA 15*]

a. Description du cas d'utilisation **« S'authentifier »**

Acteurs : L'administrateur, l'utilisateur.

Objectifs : Permet aux acteurs de s'identifier par un login et un mot de passe pour accéder à leurs espaces,

Pré-conditions : Les acteurs doivent être créé dans la base de données, connaitre leur identifiants et ne pas être identifié,

Scénario nominal :

1. L'acteur demande système,
2. Le système affiche le formulaire d'identification,
3. L'acteur saisie son mot de passe et son login et valide,
4. Le système vérifie les données saisies avec celles existantes dans la base de données,
5. Le système réoriente vers l'espace de l'acteur et l'affiche.

Alternatives :

5. a. L'acteur n'existe pas dans la BDD,
   1. Le système affiche un message informant l'acteur qu'il n'est pas inclus dans la BDD,
5. b. Mot de passe et /ou login erronés,
   1. Le système affiche un message d'erreur et propose de corriger les données saisies,

Le cas d'utilisation reprend à l'étape 3 du scénario nominal,

Post-conditions : L'espace acteur est atteint.

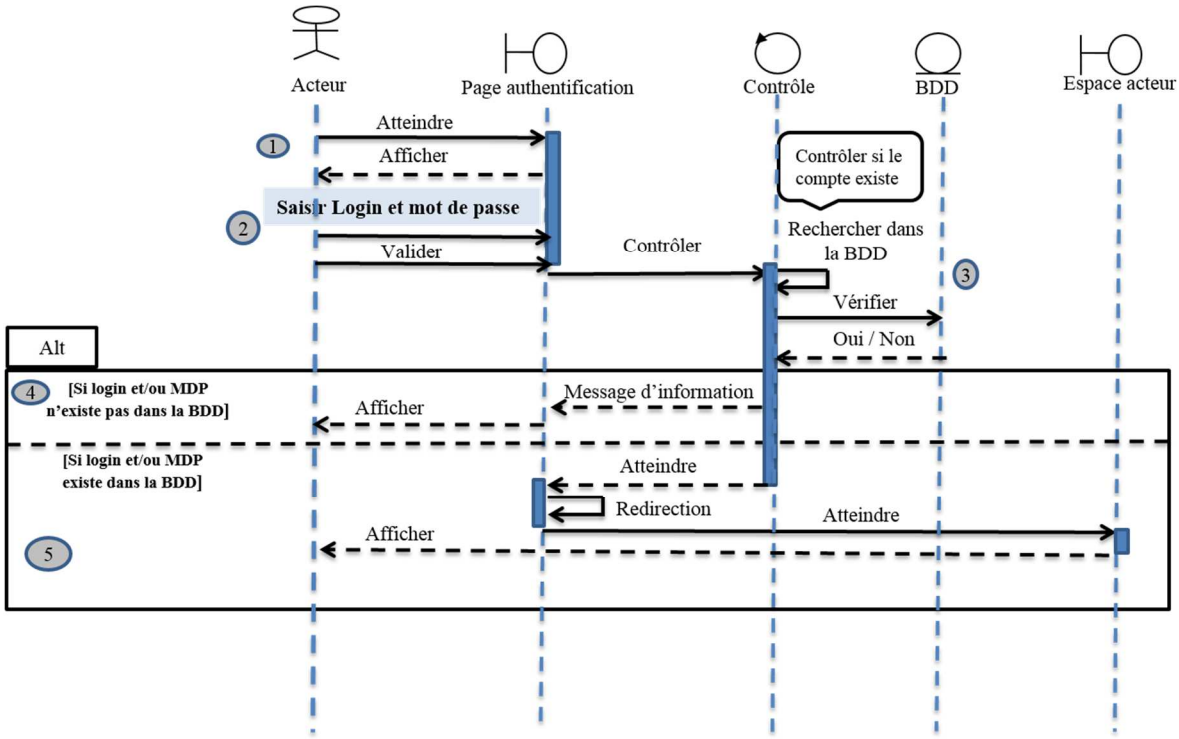


Figure 5-a : Diagramme de séquence détaillé du cas d'utilisation « s'authentifier »

Action	Authentification
1	L'acteur (administrateur ou utilisateur) demande le système, le système atteint et affiche le formulaire d'authentification.
2	L'acteur saisi son login, mot de passe et valide.
3	Le système contrôle l'existence du login et mot de passe dans la BDD.
4	Si le login ET mot de passe n'existe pas donc le système affiche un message d'information et redonne la main sur le formulaire.
5	Si le login ET mot de passe existe donc le système réoriente vers l'espace de l'acteur

Tableau 2-a : Description du diagramme de séquence du cas d'utilisation « s'authentifier »

b. Description du cas d'utilisation **« Établir une facture initiale»**

Acteurs : Utilisateur

Objectifs : Permet à l'utilisateur d'établir une facture initiale,

Pré-conditions : S'authentifier à l'espace utilisateur, Existence de dossier à facturé,

Scénario nominal :

1. L'utilisateur sélectionne "Facture initiale",
2. Le système affiche le formulaire de facture initiale composé des éléments suivants:
   - ✓ Elément à remplir obligatoire :
     - N° dossier,
   - ✓ Elément importé :
     - Coordonner de l'expéditeur (nom, adresse, téléphone, email, n° de compte...etc.),
     - Code client,
     - Éléments d'identification de la facture (date, Nom de la facture...etc.),
     - Élément métier,
     - Le détail des rubriques (Code, Désignation, Montant HT, TVA.),
   - ✓ Elément calculé :
     - Total HT, Total TTC, Total à payer,
     - Le détail des rubriques (Montant TVA, Montant TTC),
3. L'utilisateur rempli le champ N° dossier et sélectionne le bouton "valider",
4. Le système vérifier les erreurs de syntaxe,

« L'utilisateur répète les actions '3'tant que y'a des erreurs syntaxiques »

5. Le système vérifie l'existence du n° de dossier saisie,
6. Le système importe les données du dossier, son détails et rempli « éléments à importer »,
7. L'utilisateur sélectionne le bouton "valider",
8. Le système fait appel au règlement (le client est non conventionner/ conventionner décide de faire un règlement),
9. Le système enregistre la facture et le règlement avec un état (régler/à crédit).

Alternatives :

8. a. Le client est conventionner et désire ne pas effectuer le règlement,

1. Le système enregistre la facture et classe son règlement avec un état à crédit.

8. b. Le dossier est déjà facturé ou n'existe pas dans la BDD,

1. Le système affiche un message d'information chacun des cas et redonne la main sur le formulaire.

10. L'utilisateur désire éditer la facture

1. L'utilisateur clique sur éditer,

2. Le système fait appel à l'édition,

Post-conditions : la facture est établie.

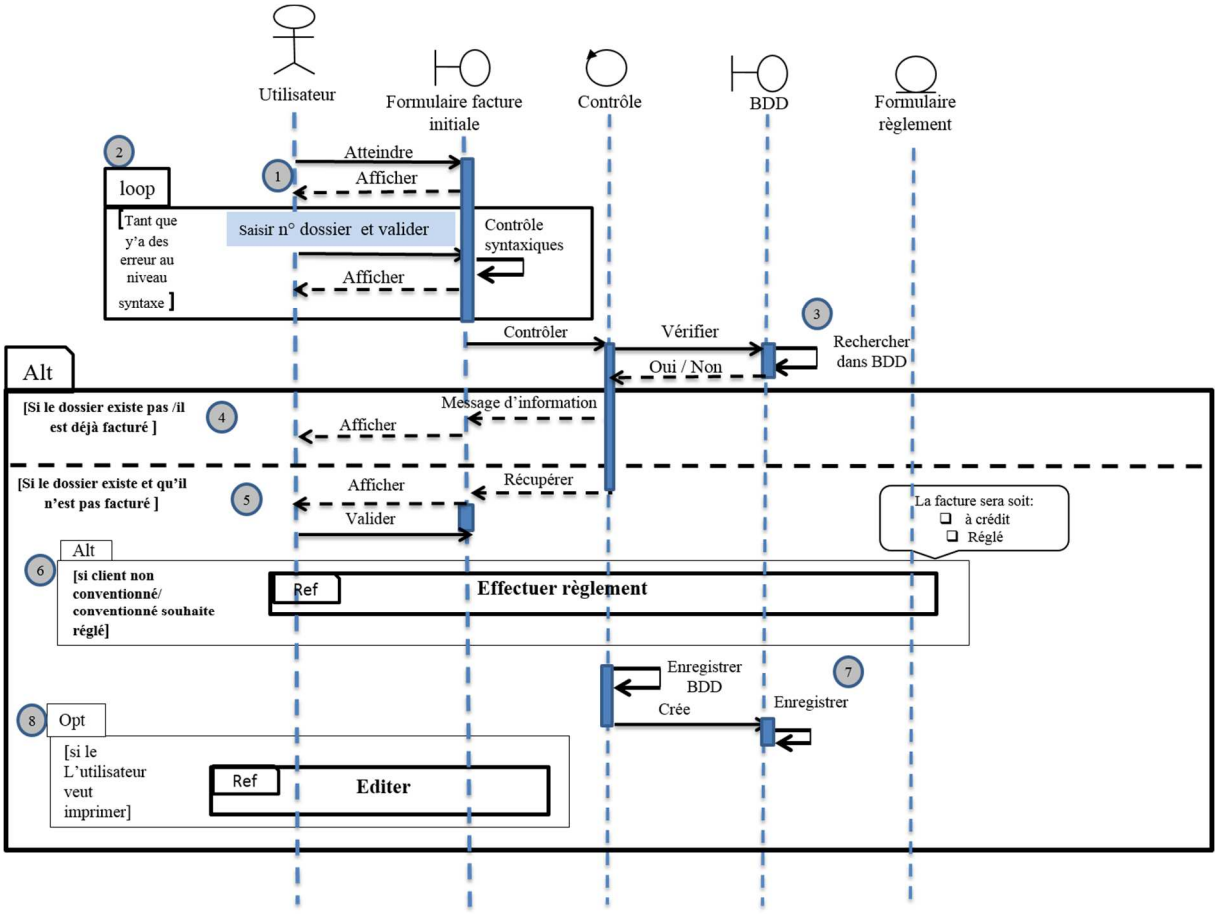


Figure 6-b : Diagramme de séquence détaillé du cas d'utilisation « Etablir une facture initiale »

Action	**Etablir facture initiale**
1	L'utilisateur sélectionne « facture initiale », le système atteint et affiche le formulaire de la facture initiale.
2	L'utilisateur rempli le champ « n° dossier » et valide, le système vérifie les erreurs syntaxique tant que il y'en a.
3	Le système vérifie l'existence du dossier et de son détail qui correspond au « n° dossier » saisie.
4	Si le dossier n'existe pas/dossier déjà facturé, le système affiche un message d'information selon le cas détecté.
5	Si le dossier existe le système récupère les données du dossier, son détail et les a affichés, l'utilisateur valide.

6	Le système fait appel au règlement dans le cas où le client est non conventionné/ conventionné et désire faire le règlement.
7	Le système enregistre la facture et le règlement avec un état (à crédit/réglé).
8	Si l'utilisateur désire éditer il fait appel à l'Edition.

Tableau 3-b : Description du diagramme de séquence du cas d'utilisation « Etablir une facture initiale »

c. Description du cas d'utilisation **« Établir facture avoir/complémentaire »**

Acteurs : Utilisateur

Objectifs : Permet à l'utilisateur d'établir une facture avoir/complémentaire dans le cas où il y'aurai une surfacturation/ sous facturation de la facture initiale.

Pré-conditions : S'authentifier à l'espace utilisateur, la facture initiale est surfacturé/sous facturé,

Scénario nominal :

1. L'utilisateur sélectionne "Facture complémentaire/ facture avoir",
2. Le système affiche le formulaire de la facture à établir composé des éléments suivant:
   - ✓ Elément à remplir obligatoire
     - n° de facture initiale (facture à modifier),
   - ✓ Eléments importé
     - Le détail des rubriques (Montant TVA, Montant TTC),
   - ✓ Elément calculé des éléments importés,
     - Coordonner de l'expéditeur (nom, adresse, téléphone, n° de compte...etc.),
     - Code client,
     - Élément métier,
     - Le détail des rubriques (code, désignation, montant HT, TVA, Montant TVA, Montant TTC),
   - ✓ Eléments déduit de(s) modification(s) de la facture,
     - Total HT, total TTC, total à payer,
     - Montant restants, montant régler, montant à régler.
3. L'utilisateur saisie « n° de facture initiale » et sélectionne le bouton « valider »,

4. Le système vérifie les erreurs de syntaxes,

« L'utilisateur répète l'action '3' tant que y'a des erreurs syntaxiques »,

5. Le système importe les éléments de la facture initiale et rempli « élément importé »,
6. L'utilisateur effectue la modification du montant d'une rubrique déjà facturé selon la facture établie (avoir/complémentaire),
7. Le système contrôle le montant modifié de manière à être conforme au type de facture établie,

« L'utilisateur répète l'action '6' tant que y'a des modifications a effectué ET/OU des erreurs de modification »,

8. L'utilisateur sélectionne « Terminer »,
9. Le système recalcule « Eléments déduit de(s) modification(s) de la facture»,
10. L'utilisateur sélectionne le bouton «Valider»,
11. Le système fait appel au règlement (le client est non conventionné/ conventionné et désire faire un règlement),
12. Le système enregistre la facture et le règlement avec un état (à crédit/régler).

Alternatives :

8. a. Si la facture est complémentaire, le client peut décider d'ajouter de(s) nouvelle(s) rubrique(s) dans la facture,

1. L'utilisateur choisi la rubrique à ajouter en la sélectionnent et rempli les champs la concernant (désignation, montant HT, TVA),
2. L'utilisateur sélectionne « Ajouter »,
3. Le système ajoute la nouvelle rubrique à la facture,

« L'utilisateur répète les actions 1,2 jusqu'à la fin des ajouts à effectuer»

4. L'utilisateur sélectionne « Terminer »,

Le cas d'utilisation continue à l'étape 9 du scénario nominal

11. a. Le client est conventionné et désire ne pas effectuer le règlement

1. Le système enregistre la facture et classe son règlement avec un état à crédit,

13. L'utilisateur désire éditer la facture

1. L'utilisateur clique sur éditer,

2. Le système fait appel à l'édition,

Post-conditions : la facture est établie.

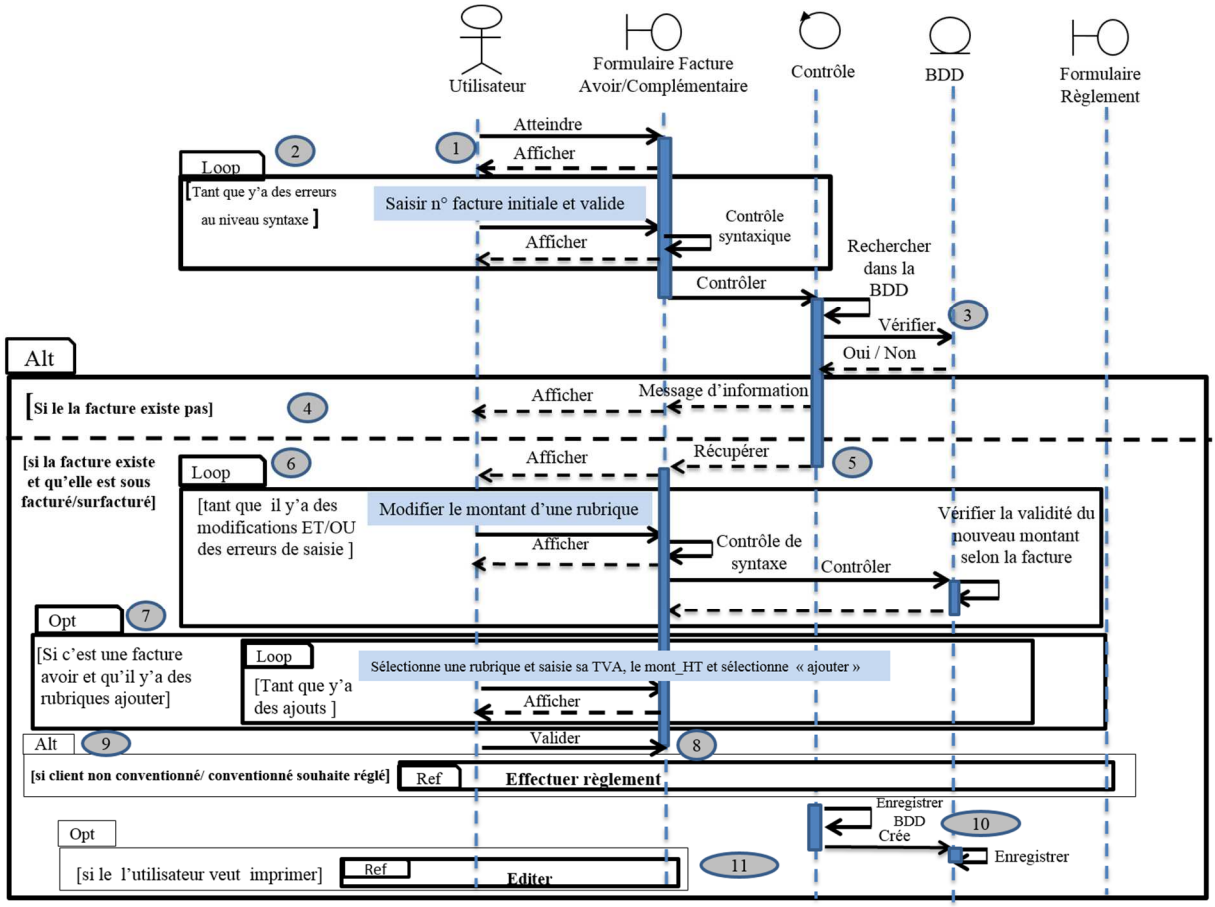


Figure 7-c : Diagramme de séquence détaillé du cas d'utilisation « Etablir une facture avoir/complémentaire »

Action	Créer « Facture avoir/complémentaire »
1	L'utilisateur sélectionne « facture avoir/complémentaire », le système atteint et affiche le formulaire de la facture avoir/complémentaire.
2	L'utilisateur rempli le champ « n° facture initiale » et valide, le système vérifie les erreurs syntaxique tant que il y'en a.
3	Le système vérifie l'existence d'une facture initiale qui correspond au « n° de facture initiale» saisie.
4	Si la facture n'existe pas, le système affiche un message d'information qui confirme l'inexistence de la facture.
5	Si la facture existe, le système récupère les données de la facture initiale et les affiches.
6	L'utilisateur modifie autant qu'il veut les montants des rubriques existantes, le système vérifie entre temps les erreurs syntaxique et la validité des nouveaux montants.
7	L'utilisateur peut dans le cas d'une facture avoir effectué l'ajout(s) de rubrique(s).
8	L'utilisateur clique sur « valider ».
9	Le système fait appel au règlement si le client est non conventionné/ conventionné souhaitent réglé.
10	Le système enregistre la facture et le règlement avec un état (à crédit/réglé).
11	Si l'utilisateur désire éditer il fait appel à l'Edition.

Tableau 4-c : Description du diagramme de séquence du cas d'utilisation « Etablir une facture avoir/complémentaire »

d. Description du cas d'utilisation **« Crée un compte client »**

Acteurs : L'utilisateur

Objectifs : Ce cas d'utilisation permet au client d'avoir un compte pour la gestion de ses versements.

Pré-conditions : S'authentifier à l'espace utilisateur, Ne pas exister dans la base de données.

Scénario nominal :

1. L'utilisateur sélectionne le lien "Crée un compte",
2. Le système affiche une page avec le formulaire de création de compte composé des éléments suivant :
   - ✓ Code client
   - ✓ Elément concernent client:
     1°.Nom, 2°.Adresse, 3°.Email, 4°.Téléphone, 5°.Code client attribué automatiquement.
   - ✓ Elément concernent versement :
     1°.Montant, 2°.Type de versement, 3°.Date du versement4° l'émetteur,
3. L'utilisateur saisie le « code client » et valide,
4. Le système confirme l'inexistence d'un compte lié au « code client »,
5. L'utilisateur saisi les champs des « éléments concernent client »,
6. L'utilisateur remplie les champs des « éléments concernent le versement »,
7. L'utilisateur clique sur le bouton « Ajouter »,
8. Le système enregistre l'enregistrement du versement dans la grille,
9. Le système redonne la main sur les « éléments concernent versement »,

« L'utilisateur répète les actions 6,7 jusqu'à la fin des versements a ajouté»

10. L'utilisateur clique sur le bouton « valider »,
11. Le système enregistre les données.

Les alternatives :

6.a. Le client n'a pas son versement sur lui au moment de la création de son compte

1. L'utilisateur va directement à l'étape 10 du scénario nominal.
2. Le système enregistre les données.

4.a. Le système confirme l'existence d'un compte client lié au code client

1. Le système importe les donnée du client et rempli « élément concernent client»,

Le cas d'utilisation reprend à l'étape 6 du scénario nominal,

12. L'utilisateur désire éditer la facture

1. L'utilisateur clique sur éditer,

2. Le système fait appel à l'édition,

Post-conditions : Nouveau compte client crée.

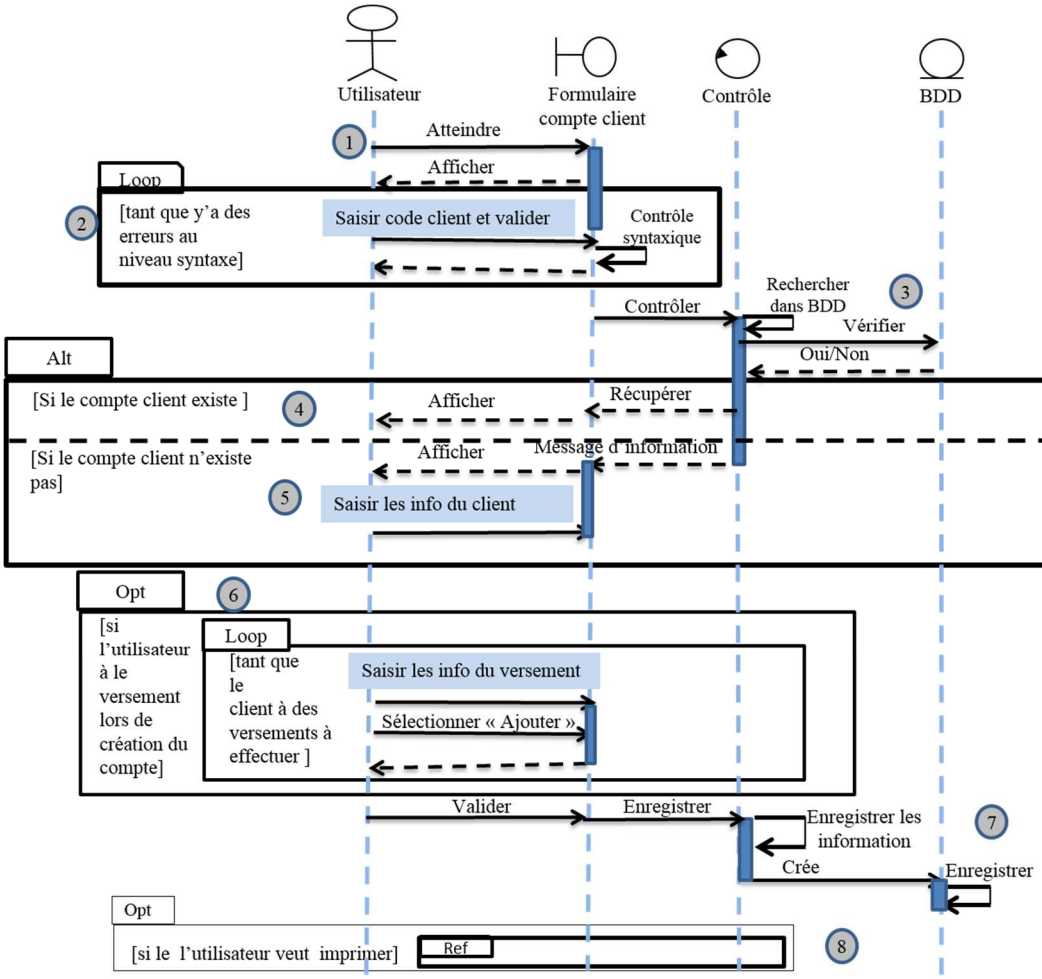


Figure 8-d : Diagramme de séquence détaillé du cas d'utilisation « Crée un compte client »

Action	Créer compte client
1	L'utilisateur sélectionne « compte client », le système atteint et affiche le formulaire du compte client.
2	L'utilisateur rempli le champ « code client » et valide, le système vérifie les erreurs syntaxique tant que il y'en a.
3	Le système vérifie l'existence du compte client dans la base de données.
4	Si le compte client existe, le système récupère les données concernant le client.
5	Si le compte client n'existe pas, l'utilisateur saisie les informations concernant le client.
6	L'utilisateur, s'il a des versements au moment de la création, peut ajouter autant de versement qu'il souhaite en remplissant à chaque fois les informations sur le versement effectuer et clique sur ajouter, le système enregistre un enregistrement du versement à chaque versement dans la grille et l'affiche.
7	L'utilisateur valide, le système enregistre les données
8	Si l'utilisateur désire éditer il fait appel à l'Edition.

Tableau 5-d : description du diagramme de séquence du cas d'utilisation « Crée un compte client »

e. Description du cas d'utilisation **« Effectuer le règlement »**

Acteurs : L'utilisateur

Objectifs : Permet à l'utilisateur d'effectuer le règlement des facture initiales, complémentaires et avoirs et de leur suivi si elles sont à crédit.

Pré-conditions : Le client est conventionner et désire effectuer le règlement / le client est non conventionner, le montant total de la facture (initiale, complémentaire ou avoir) concerne le règlement à effectuer.

Scénario nominal :

1. L'utilisateur demande le règlement,
2. Le système affiche le règlement composé des éléments suivant :

   - ✓ Eléments à remplir obligatoire,
     - – Montant de la tranche,
     - – Type de règlement,
     - – Emetteur
   - ✓ Eléments à remplir obligatoire du règlement par chèque,
     - – Référence du chèque,
   - ✓ Eléments déduit après traitement,
     - – Montant (restants, réglé, a réglé),
     - – Une grille ou s'affiche le détail du règlement effectuer,
3. L'utilisateur remplie les champs « à remplir obligatoire »,
4. L'utilisateur clique sur le bouton « Ajouter»,
5. Le système enregistre l'enregistrement de la tranche dans la grille,
6. Le système redonne la main sur les champs du formulaire de règlement,

« L'utilisateur répète les actions 3,4 jusqu'à ce que le montant réglé = montant total de la facture »

7. L'utilisateur clique sur le bouton « valider »,
8. Le système enregistre les données,

Les alternatives :

3. a. Le client désire régler sa facture par "Versement":

1. L'utilisateur sélectionne le mode de règlement par "Versement"
2. Le système lui affiche un lien vers le compte du client s'il a un compte,
3. L'utilisateur sélectionne la référence du versement sur lequel il veut débiter,
4. Le système ajoute toutes les informations du versement + le montant de la tranche selon le versement.

Le cas d'utilisation continue à l'étape 4 du scénario nominal

3. b. Le client désire régler sa facture par "Espèce":

1. L'utilisateur sélectionne le mode de règlement par "Espèce" et saisi le montant de la tranche, après le cas d'utilisation reprend à l'étape 4 du scénario nominal,

3. c. Le client désire régler sa facture par « Chèque »:

1. L'utilisateur sélectionne le mode de règlement par "Chèque",
2. L'utilisateur saisi les champs « Champ obligatoire du règlement par chèque», Le cas d'utilisation reprend à l'étape 4 du scénario nominal,

7. a. Montant total réglé < montant total de la facture d'un client non conventionné :

1. Le système annule le règlement et avec, la facture,

7. b. Montant total réglé < montant total de la facture d'un client conventionné :

1. Le système enregistre la facture avec l'état à crédit du montant restant a réglé,
2. Le système enregistre les données,

Post-conditions : Facture enregistrée.

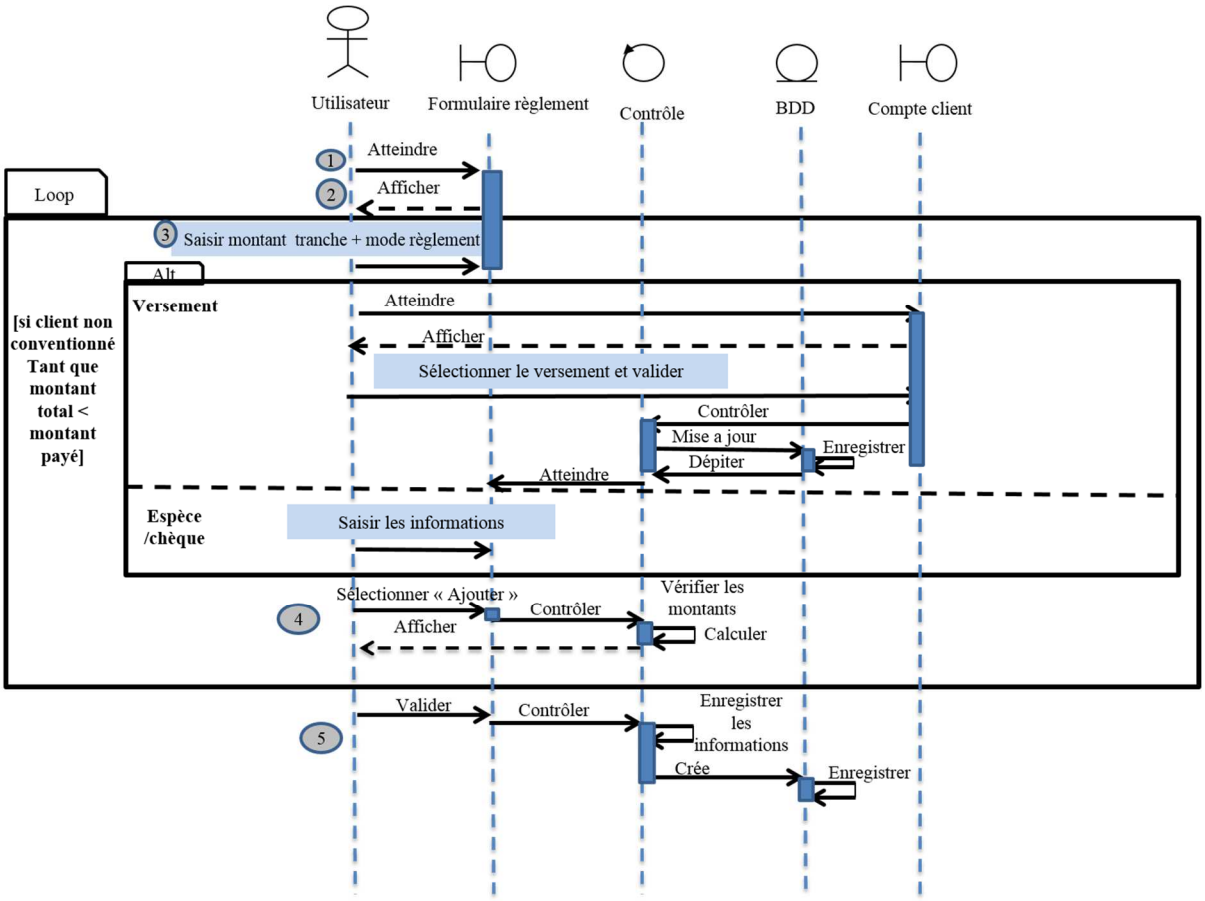


Figure 9-e: Diagramme de séquence détaillé du cas d'utilisation « Effectuer le règlement »

Action	Créer « Règlement »
1	Utilisateur attient le formulaire de règlement.
2	Le système affiche le formulaire de règlement.
3	Utilisateur saisi le montant de la tranche et le mode de règlement qui peut être par versement, chèque ou espèce.
4	L'utilisateur valide, le système enregistre l'enregistrement de la tranche dans la grille
5	L'utilisateur clique sur « valider », le système enregistre les données.

Tableau 6-e : Description du diagramme de séquence du cas d'utilisation « Effectuer le règlement »

3.2. Modélisation statique (diagramme de classe)

Le diagramme de classes est le point central dans un développement orienté objet. En analyse, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs.

En conception, le diagramme de classes représente la structure d'un code orienté objet ou, à un niveau de détail plus important, les modules du langage de développement. [*LA 15*]

La réalisation du diagramme de classe se base sur les règles de gestion qui permet de définir liens qui relient ces classes. Ce basent sur cella on est arrivé dans le diagramme de classe du notre futur système aux règles de gestion suivantes :

1. Une facture initiale peut faire aboutir plusieurs factures avoir et/ou complémentaire,
2. Une facture avoir/ complémentaire est aboutie par une seul facture initiale,
3. Une facture d'un client non conventionné contient un seul règlement,
4. Une facture d'un client conventionné contient plusieurs règlements à différente dates,
5. Une facture est possédée par un seul client,
6. Une facture contient un ou plusieurs détails de facturation,
7. Une facture initiale est composée d'un seul dossier et une seul fois,
8. Un client peut posséder plusieurs factures,
9. Un client peut posséder qu'un seul compte client,
10. Un compte client peut être composé de plusieurs versements,
11. Un compte client est possédé par un seul client,
12. Un versement compose un seul compte client,
13. Un versement possédé un seul mode de règlement,
14. Un mode règlement peut être possédé par plusieurs versements,
15. Un versement peut régler plusieurs tranches d'un ou plusieurs règlements d'une ou plusieurs factures,
16. Une tranche possède un seul mode règlement,
17. Une tranche réalise un seul règlement,
18. Un mode de règlement peut être possédé par plusieurs tranches d'un seul règlement,
19. Un règlement est réalisé par un ou plusieurs tranches de mêmes/différents mode de règlement,
20. Un règlement est contenu par une seule facture,
21. Un client conventionné peut ne pas effectuer de règlement,
22. Un détail de facturation est contenu dans une seule facture,
23. Un détail de facturation concerne une seul rubrique de facturation,
24. Une rubrique de facturation est concernée par un seul détail de facturation de différentes factures,
25. Une rubrique de facturation appartient à une seule rubrique famille,
26. Une rubrique famille est concernée par une à plusieurs rubriques de facturation.

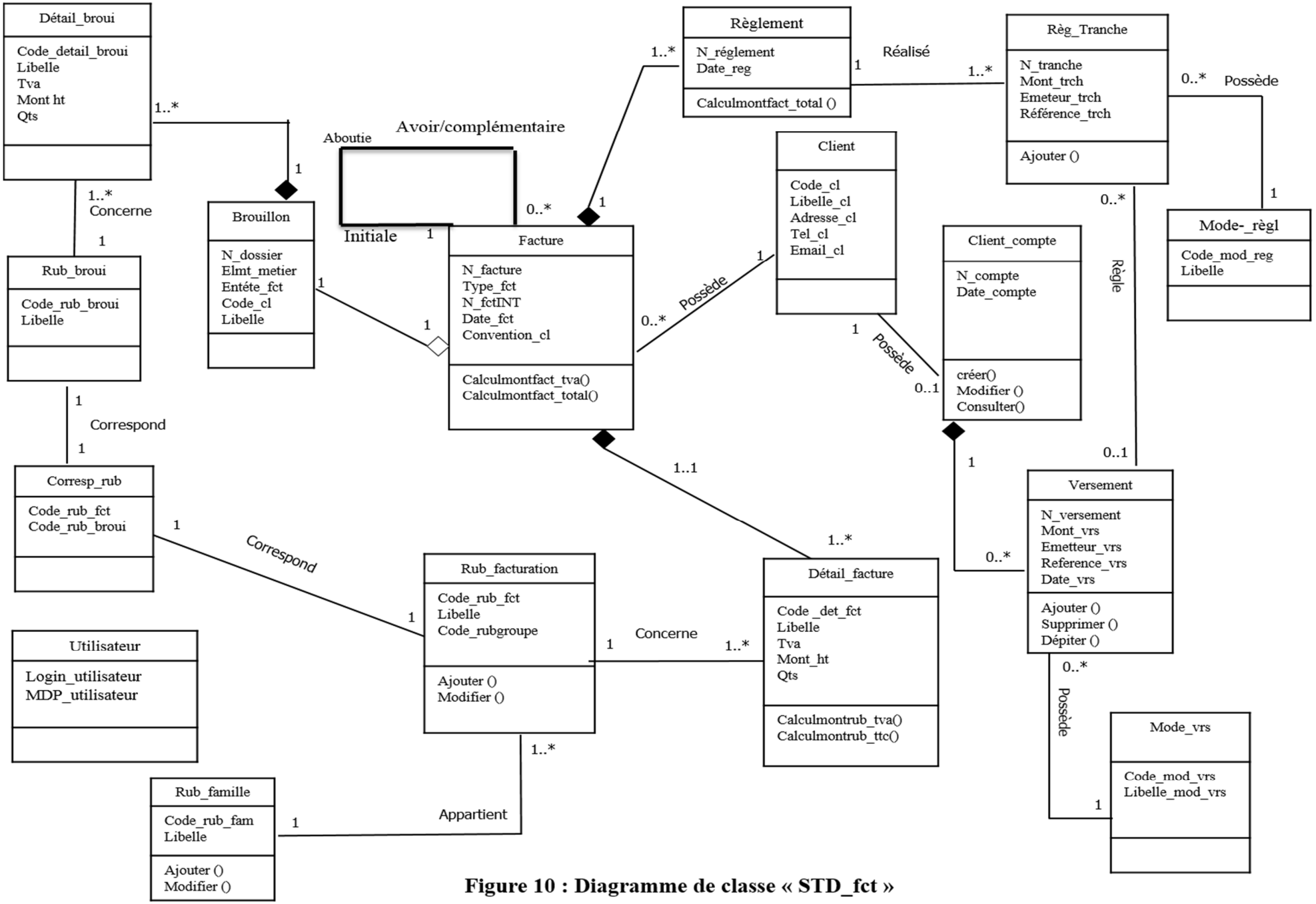


Figure 10 : Diagramme de classe « STD_fct »

4. Modélisation de BDD

Après avoir construit le diagramme de classe nous allons établir à ce niveau la base de donné lui correspondent en effectuent le passage du diagramme de classe au modèle relationnel

4.1. Modèle relationnel

4.2.1.Structure logique des tables de la base de données

En respectant les règles de passage qui existe du diagramme de classe au relationnel nous somme arriver aux résultats suivants :

- **Brouillon** (N_dossier, Elmt_metier, Entéte_fct,code_client,Libelle)
- **Détail_Broui** (Cod_rub_broui*,N_dossier*, Libelle, Tva, Mont_HT, Qts)
- **Rub_broui** (Code_rub_broui, Libelle)
- **Corresp_rub** (Code_rub_fct, Cod_rub_broui)
- **Rub_facturation** (Code_rub_fct, Libelle, Code_rubgroup, Code_rub_fam*)
- **Détail_facture** (Code_rub_fct*,N_facture*, Libelle, Tva, Mont_HT, Qts)
- **Facture** (N_facture, Type_fct, N_fctInt, Date_fct, Convention_cl, N_dossier*,Code_cl*)
- **Rub_famille** (code_rub_fam, libelle)
- **Règlement** (N_règlement, Date_rgl, N_facture*)
- **Rég_tranche** (N_tranche, Mont_trch, Emetteur_trch, Référence_trch, N_réglement*, N_versement*, Code_mod_reg*)
- **Mode_régl** (Code_mod_reg, Libelle_mode_reg)
- **Client** (Code_cl, Libelle_cl, Adresse_cl, Tel_cl, Email_cl)
- **Compte_client** (N_compte, Date_compte, Code_cl*)
- **Versement** (N_versement, Mont_vrs, Emeteur_vrs, Reference_vrs, Date_vrs, N_compte*, Code_mod_vrs*)
- **Mode_vrs** (code_mod_vrs, Libelle_mod_vrs)
- **Utilisateur** (Login_utilisateur, MDP_utilisateur)

4.2.2.Structure physique des tables de la base de données:

Le modèle physique des données est l'implantation des données (tables) issues du modèle logique des données en machine afin d'aboutir à la description des fichiers de base de données.

La structure des tables de la base de données est spécifiée ci-dessus:

Table : Brouillon

Champ	Type_donnée	Description	contrainte
N_dossier	Int ()	Numéro du dossier	**Pk**
Entete_fact	Varchar (300)	Entêté de la facture	
Elmt_métier	Varchar (300)	Elément du métier	
Code_cl	Char (5)	Code du client	
Libelle	Varchar (80)	Désignation du client	

Table : Détail_broui

Champ	Type_donnée	Description	Contrainte
N_dossier	Int	Code du détail du brouillon	Pk, Fk
Code_rub_broui	Char (6)	Code rubrique brouillon	Pk, Fk
Libelle	Varchar (250)	Désignation de la rubrique	
Tva	Numeric (6,2)	Taxe sur la valeur ajoutée	
Mont_HT	Numeric (14,6)	Montant hors taxe	
Qts	Int ()	Quantités	

Table : Rub_broui

Champ	Type_donnée	Description	Contrainte
Code_rub_broui	char (6)	Code de la rubrique brouillon	Pk
Libelle	Varchar (250)	Désignation de la rubrique	

Table : Corresp_rub

Champ	Type_donnée	Description	Contrainte
Code_rub_fct	Char (6)	Code de la rubrique facture	Pk,Fk
Code_rub_broui	Char (6)	Code de la rubrique brouillon	Pk,Fk

Table : Rub_facturation

Champ	Type_donnée	Description	Contrainte
Code_rub_fct	char (6)	Code de la rubrique facture	Pk
Libelle	Varchar (250)	Désignation rubrique de la facture	
Code _rubgroupe	char (6)	Code du la rubrique groupe	
Code_rub_famille	char (6)	Code de la rubrique famille	Fk

Table : Rub_famille

Champ	Type_donnée	Description	Contrainte
Code_rub_famille	Char (6)	Code rubrique famille	Pk
libelle	Varchar (250)	Désignation de la rubrique famille	

Table : Détail_facture

Champ	Type_donnée	Description	Contrainte
N_facture	Int ()	Numéro de facture	Pk, Fk
Code_rub_fct	Char (6)	Code de la rubrique facture	Pk, Fk
Libelle	Varchar (250)	Désignation de la rubrique facturée	
Tva	Numeric (6,2)	Taxe sur la valeur ajoutée	
Mont_HT	Numeric (14,6)	Montant hors taxe	
Qts	Int ()	Quantités	

Table : Facture

Champ	Type_donnée	Description	Contrainte
N_facture	Int ()	Numéro de facture	Pk
Type_fct	Char (1)	Type de facture	
N_fctInt	Int ()	Numéro facture initiale	Fk
Date_fct	Datetime ()	Date de l'établissement de la facture	
Convention_cl	Bit	Convention de client	
Code_cl	Char (5)	Code du client	Fk
N_dossier	Int ()	Numéro du dossier	Fk

Table : Règlement

Champ	Type_donnée	Description	Contrainte
N_règlement	Int ()	Numéro de règlement	Pk
Date_reg	Datetime ()	Date du règlement	
N_facture	Int ()	Numéro la facture	Fk

Table : Règ_tranche

Champ	Type_donnée	Description	Contrainte
N_tranche	Int ()	Numéro de la tranche	Pk
Mont_trch	Numeric (14,6)	Montant de la tranche	
Emeteur_trch	Varchar (150)	Emetteur de la tranche	
Référence_trch	Varchar (150)	Référence de la tranche	
N_reglement	Int ()	Numéro de règlement	Fk
N_versement	Int ()	Numéro de versement	Fk
Code_mod_reg	Varchar (5)	Code du mode de règlement	Fk

Table : Mode_règl

Champ	Type_donnée	Description	Contrainte
Code _mod_reg	Varchar (5)	Code du mode de règlement	Pk
Libelle_mod_reg	Varchar (250)	Désignation du mode de règlement	

Table : Client

Champ	Type_donnée	Description	Contrainte
Code_cl	Char (5)	Code du client	Pk
Libelle_cl	Varchar (80)	Désignation du client	
Adresse_cl	Varchar (200)	Adresse du client	
Tel_cl	Varchar (80)	Téléphone du client	
Email_cl	Varchar (80)	Email du client	

Table : Compte client

Champ	Type_donnée	Description	Contrainte
N_compte	Int ()	Numéro du compte	Pk
Date_compte	Datetime	Date de création du compte	
Code_cl	Char (5)	Code du client	Fk

Table : Versement

Champ	Type_donnée	Description	Contrainte
N_versement	Int ()	Numéro du versement	Pk
Mont_vrs	Numeric (14,6)	Montant du versement	
Emetteur_vrs	varchar (150)	Emetteur du versement	
Référence_vrs	Varchar (150)	Référence du versement	
Date_vrs	Datetime ()	Date du versement	
N_compte	Int ()	Numéro de compte client	Fk
Code_mod_vrs	Varchar (5)	Code du mode versement	Fk

Table : Mode_vers

Champ	Type_donnée	Description	Contrainte
Code_mod_vrs	Varchar (5)	Code du mode de versement	Pk
Libelle_mod_vrs	Varchar (25)	Libelle de mode de versement	

Table : Utilisateur

Champ	Type_donnée	Description	Contrainte
Login_utilisateur	Varchar (20)	Login de l'utilisateur	Pk
MDP_utilisateur	Varchar (20)	Mot de passe de l'utilisateur	

Conclusion

Dans ce chapitre, nous avons pu concevoir un système d'information pour la gestion de la facturation standard nommé « STD_fct » en se basant sur les diagrammes du langage UML.

La prochaine étape consistera donc en la concrétisation du modèle de la solution que nous avons proposé, en d'autres termes, la réalisation des différents objets et des différentes fonctionnalités qui la constituent. Cet aspect fera l'objet du prochain chapitre.

CHAPITRE 4

[REALISATION]

Introduction

Après avoir présenté précédemment la conception et le fonctionnement global de notre système**,** nous arrivons dans ce chapitre à la mise en œuvre de notre application.

Ce chapitre sera divisé en deux parties : la première partie sera consacrée à la description des outils de développement de notre application. La deuxième partie contiendra les détails techniques d'implémentation des différentes fonctionnalités offertes par notre système.

1. Architecture et outils de développement

Dans cette première partie, nous allons présenter le support matériel et logiciel qui nous a servi d'appui pour le développement de notre application, afin de la mettre en œuvre, nous les citons dans ce qui suit :

1.1. Architecture

1.1.1. JEE (java entreprise édition) [JC O5]

L'architecture trois-tiers a été entièrement conçue selon la norme Java EE (Java Enterprise Edition). Celle-ci est une norme proposée par la société Sun et s'appuie entièrement sur le langage Java, ce qui assure une bonne portabilité et maintenabilité du code.

Cette norme définit les couches suivantes :

I. Couche présentation

Elle correspond à la partie de l'application visible et interactive avec les utilisateurs, elle relaie les requêtes de l'utilisateur à destination de la couche métier, et en retour lui présente les informations renvoyées par les traitements de cette couche.

II. Couche métier

Elle se décompose en 3 parties :

a. Logique métier :

Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique », et qui décrit les opérations que l'application opère sur les données en fonction des

requêtes des utilisateurs, effectuées à travers de la couche présentation.

Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

b. Accès aux données :

Elle consiste en la partie gérant l'accès aux données du système. Cette couche permet à la couche métier à accéder aux données d'une manière uniforme, celle-ci n'a pas à s'adapter aux changements possibles dans les couches supérieures puisque toutes les opérations sont maintenues transparentes par la couche d'accès aux données.

c. Objets de données :

Elle assure la persistance des données, à chaque table de la base de données correspond une classe entité, des annotations sont aussi utilisées pour mettre en œuvre les concepts de POO (Programmation Orienté Objet) tels que l'héritage ou le polymorphisme.

III. Couche BDD: Inclut la base de données.

L'utilisation de Java EE dans le développement et l'exécution d'une application propose plusieurs avantages:

- ✓ Une architecture d'application basée sur des composants permet un découpage de l'application et donc une séparation des rôles lors du développement.
- ✓ La possibilité de s'interfacer avec le système d'information existant grâce à de nombreuses API : JDBC, JNDI, JMS, JCA ...
- ✓ la possibilité de choisir les outils de développement et le ou les serveurs d'applications utilisés qu'ils soient commerciaux ou libres.

1.1.2. Design pattern [JP 05]

Design patterns, formes, modèles ou bien patrons de conception. Décrivent, formalisent des solutions éprouvées à des problèmes spécifiques et récurrents. « Un patron décrit un

problème devant être résolu, une solution, et le contexte dans lequel cette solution est considérée. Il nomme une technique et décrit ses coûts et ses avantages » [Johnson 1997].

Exemple : le modèle MVC

Le modèle MVC cherche à séparer les couches présentation, traitement et accès aux données. Une application web respectant ce modèle pourrait être architecturée de la façon suivante :

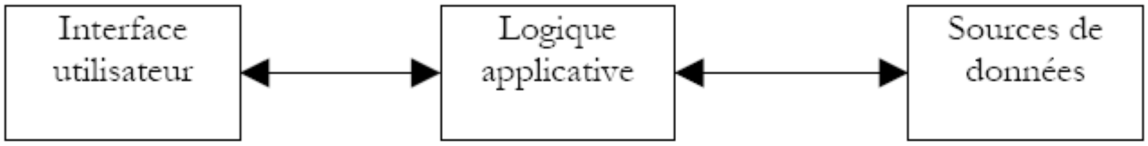


Figure 1 : Architecture du modèle MVC

a. **La vue :** Ce qu'on nomme la vue est en fait une IHM (Interface Homme – Machine). C'est ce que l'utilisateur peut voir de l'application : les pages Web, les états de sortie, … etc.
b. **Le modèle :** Il s'agit en général d'un ou plusieurs objets JAVA (beans), ces objets s'apparentent à ce qu'on appelle la « couche métier » de l'application. Ils exécutent des traitements spécifiques au métier de l'application en arrière-plan.
c. **Le contrôleur :** C'est un ensemble d'objets JAVA ou autres, qui fait office de coordinateur entre la vue et le modèle, et gère les flux d'informations entre eux.

1.2. Développement

1.2.1. Java [EP 05]

C'est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut être utilisé sur internet pour des petites applications intégrées à la page web ou encore comme langage serveur. Son succès est dû à un ensemble de caractéristiques dont voici un aperçu :

✓ Langage de programmation objet et fortement typé : difficiles pendant le développement, l'approche objet et le typage fort du langage Java rendent plus robuste un programme Java dès sa conception.

✓ Gestion des exceptions : Java intègre la gestion des exceptions autant pour faciliter la mise au point des programmes (détection et localisation des bogues) que pour rendre un programme plus robuste.

✓ Multitâche : grâce aux threads, Java permet de programmer l'exécution simultanée de plusieurs traitements et la synchronisation des traitements qui partagent des informations.

✓ Bibliothèque très riche : la bibliothèque fournie en standard avec Java couvre de nombreux domaines (gestion de collections, accès aux bases de données, interface utilisateur graphique, accès aux fichiers et au réseau, utilisation d'objets distribués, XML…, sans compter toutes les extensions qui s'intègrent sans difficulté à Java !)

✓ Exécutable portable : comme l'exprime l'accroche *Write Once Run Anywhere*, un programme Java, une fois écrit et compilé, peut être exécuté sans modification sur tout système qui prend en charge Java.

✓ Système de sécurité : Java protège les informations sensibles de l'utilisateur et le système d'exploitation de sa machine en empêchant l'exécution des programmes conçus de façon malintentionnée (contre un virus par exemple).

Critères de choix

✓ Richesse : un des aspects important de l'environnement de JAVA est sa richesse de ses librairies des classes JAVA.

✓ Interprète, portable et indépendant des architectures matérielles. Cette caractéristique est un avantage primordial pour Java face à des applications transmises par un réseau et exécutées sur des machines hétérogènes.

1.2.2. HTML [G.Pierre]

HTML est l'abréviation de HyperText Markup Language, soit en français « langage hypertexte de balisage ». Ce langage a été créé en 1991 et a pour fonction de structurer et de donner du sens à du contenu.

HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des éléments programmables tels que des applets[1]. Il est souvent utilisé simultanément avec

[1] *La forme de bytecode Java. Une applet Java peut fonctionner dans un navigateur web, grâce à une machine virtuelle Java*.

des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade).

Critères de choix

- ✓ Simplicité (il est simple à utiliser).
- ✓ Indépendance (sa conception lui permet de rester indépendant vis à vis des plateformes et de pouvoir être échangé sur les réseaux).
- ✓ Aucun logiciel spécialisé n'est nécessaire pour composer des pages HTML, il peut être composé sur n'importe quel système.

1.2.3. JavaScript [D-MS]

JavaScript est langage de script, qui permet notamment de manipuler les éléments d'une page Web, d'interagir avec le navigateur internet et de réagir aux actions de l'utilisateur. Contrairement à certaines idées reçues, JavaScript est un langage moderne qui offre de nombreuses fonctionnalités puissantes comme :

- ✓ Le développent orienté objet ou encore la gestion des exceptions.
- ✓ Les fonctions du JavaScript ne doivent pas attendre pour des réponses de leurs serveurs pour agir, ce qui accélère l'ouverture des sites web.

Le JavaScript est relativement simple et facile à apprendre. Il ne nécessite pas un programme spécial pour l'interpréter, ni pour l'écrire. De plus, JavaScript n'occupe pas un grand espace sur les sites web.

Critères de choix

- ✓ Vitesse(Les fonctions du JavaScript ne doivent pas attendre pour des réponses de leurs serveurs pour agir).
- ✓ Simplicité (le JavaScript est relativement simple et facile à apprendre).
- ✓ Versatilité - Le JavaScript ne nécessite pas un programme spécial pour l'interpréter, ni pour l'écrire.

1.2.4. CSS (Les feuilles de styles) [G.Pierre]

Créées pour prendre en charge tous les aspects graphiques et les rendus sur les différents médias (écran, mais aussi imprimante, synthèse vocale, assistant personnel, etc.), les feuilles de styles ajoutent la couche graphique au document web et à sa structure.

CSS *(Cascading Style Sheets)* est un langage spécifique au Web, fréquemment employé comme complément du langage HTML, et dont la fonction est de former des feuilles de styles chargées de la mise en forme des documents web. Il gère l'esthétique (couleurs, typographie) et diverses fonctionnalités.

Son champ d'action ne se limite pas au média screen (écran), mais s'étend également aux médias print (imprimante), projection (présentations projetées), braille (tablettes à l'usage des aveugles), embossed (impression en braille), aural/speech (propriétés auditives), handheld (assistants numériques) et tv (Web-TV).

La version actuelle de CSS (CSS 2) complète la version précédente, CSS 1, avec laquelle elle est entièrement compatible.

Critères de choix

- ✓ Permettre des choses irréalisables en html (formater un contenu structuré.).
- ✓ Allègement du code-source des pages Web donc de plus petits fichiers et un chargement plus rapide.
- ✓ plus facile de faire des changements d'ensemble (un seul fichier CSS à modifier plutôt que toutes les pages une à une)

1.2.5. JSF (JAVA SERVER FACES) [G.Pierre]

L'Interface web est entièrement gérée par le framework JSF. Ce framework est destiné à la plate-forme Java, permettant de développer des interfaces graphiques améliorées pour des applications web exécutées côté serveur. JSF se compose d'un ensemble d'API servant notamment à représenter les composants graphiques, à gérer les états et à supporter les événements. Il dispose également d'un ensemble de balises conçues pour exprimer les interfaces JSF à l'intérieur d'une page JSP (page web dynamique). D'autres outils du Web 2.0 sont utilisés : XHTML, CSS, JavaScript, XML, …etc.

Critères de choix

✓ JSF proposer un framework qui puisse être mis en oeuvre par des outils pour permettre un développement de type RAD (Rapid Application Development) pour les applications web et ainsi faciliter le développement des applications de ce type.

✓ La maintenabilité (une maintenance simplifiée grâce à sa structure).

✓ Organisation du code (Il sépare la présentation des traitements).

✓ JSF s'adapte parfaitement au développement d'applications web complexes en facilitant leur écriture.

1.2.6. RichFaces [GA 07]

Le projet RichFaces est une plateforme de composant pour les interfaces utilisateurs avancées permettant l'intégration aisée d'Ajax dans les applications commerciales utilisant JavaServer Faces, un framework Java pour le développement d'applications Web.

Critères de choix

✓ Elle est implémentée par le Framework JSF.

✓ Permettre des choses irréalisables en html (les calendriers, les grid…).

1.2.7. JPQL [GA 07]

JPQL, ou Java Persistence Query Language, est un langage de requêtes déclaratif s'inspirant de la syntaxe de SQL. Sa particularité est de manipuler des objets dans sa syntaxe de requête et de retourner des objets en résultat. On manipule donc des objets dans une requête JPQL, puis le mécanisme de mapping transforme cette requête JPQL en langage compréhensible par une base de données relationnelle (en SQL).

L'avantage de JPQL est que le développeur n'a pas à connaître un nouveau langage, JPQL est plus intuitif pour un développeur Java, car il utilise une approche objet. En effet, le développeur manipule son modèle objet, et non une structure de données, en utilisant la notation pointée (c'est-à-dire maClasse.monAttribut).

JPQL crée donc une abstraction par rapport à la base de données. Il est portable quel que soit le moteur utilisé via son interface Query.

Critères de choix

✓ Un langage indépendant du SGBD (Système de gestion de base de données) car il se base sur l'approche orienté objet (Il permet de manipuler des objets, mais pas des tables).

✓ Il se base sur le concept de Java EE.

1.2.8. BIRT [D-MS]

BIRT est une bibliothèque écrite en Java qui permet de générer des états à partir d'un modèle et d'une base de données ou (de listes) d'objets remplis. BIRT est accompagné d'un plugin Eclipse qui permet de dessiner les rapports à générer et les tester.

Critères de choix

✓ Générez vos rapports dans le format de votre choix : Html, PDF, Word, Excel, PowerPoint, …

✓ Utilisation une large palette de composants d'affichage pour composer des rapports : tableaux, grilles, images, graphiques …

✓ Exploitation simultanée de différentes sources de données : bases de données relationnelles, fichiers XML, Web Services, objets Java, ...

1.3. Serveur de données (SQL SERVER 2005) [OSG 14]

SQL Server 2005 (Nom de code: **Yukon**) est le futur SGBDR (Système de Gestion de Bases de Données Relationnelles) de la plateforme Microsoft.

Basé sur les points forts de son prédécesseur (SQL Server 2000), Yukon inclura beaucoup de nouvelles fonctionnalités qui vous permettront, vous et votre Entreprise, de devenir plus productif. Il vous permettra, entre autres choses :

✓ De créer et déployer des applications plus sûres, plus puissantes et plus fiables ;

✓ De proposer aux développeurs un environnement de développement riche, souple et moderne permettant de créer des applications de bases de données plus sûres ;

✓ De partager des données entre diverses plates-formes, applications et systèmes pour faciliter les connexions, tant internes qu'externes.

1.4. Serveur d'applications (JBOSS 5.0.1) [OSG 14]

JBoss Application Server est un serveur d'applications J2EE Libre entièrement écrit en Java, publié sous licence GNU LGPL. Puisque le logiciel est écrit en Java, JBoss Application Server peut être utilisé sur tout système d'exploitation fournissant une machine virtuelle Java.

1.5. Environnement de développement (Eclipse 3.4 (Ganymede)) [OSG 14]

Eclipse est un IDE (environnement de développement intégré) écrit en Java, extensible par des greffons, multi-langages et multi-plates-formes, qui s'intègre particulièrement bien à GNOME.

Il est d'abord conçu pour le langage Java mais ses nombreux greffons en font un environnement de développement pour de nombreux autres langages de programmation (C/C++, Python, PHP, Ruby, …).

Toutes les fonctions qu'on peut attendre de ce genre de logiciel sont présentes ou existent sous forme de greffons (coloration syntaxique, complétion, debuggé, gestion de projets, intégration aux gestionnaires de versions, …).

2. Présentation de l'application

Nous nous somme efforcé dans cette partie du chapitre à la présentation de notre application réalisé sous différente ongle comme suit :

2.1. Schéma fonctionnel de l'application

Réaliser le schéma fonctionnel ou l'arborescence d'une application c'est représenté les différentes rubriques et les relations existantes entre elles, à l'aide de rectangles pour les rubriques et de flèches pour les relations, donc l'arborescence décrit l'organisation et le contenu d'une application.

Le plus important à considérer lorsqu'on prépare le design d'une application, c'est la satisfaction des utilisateurs en fournissent à l'usager l'information qu'il souhaite en un minimum d'étapes et donc un minimum de temps. On parle alors de la "règle des 3 clics" selon laquelle toute information de votre application doit être disponible en maximum 3 clics de souris. En conséquence on a opté pour une arborescence de type hiérarchique, qui est une structuration organisé en arbre dans laquelle les différents thèmes dépendent d'une seul et unique page, soit l'index ou page d'accueil appelée « la racine », puis les autres pages apparaissent ensuite dans un ordre logique dans lequel on aura un nombre de rubriques accessibles depuis la page d'accueil et un nombre de sous-rubriques accessibles depuis une rubrique.

L'arborescence d'une application doit être au préalable réalisée sur papier avant de commencer la création. Elle est la base de toute application web à sa création.

Dans le cas de notre application on est arrivé après une analyse de nos informations à l'arborescence suivante :

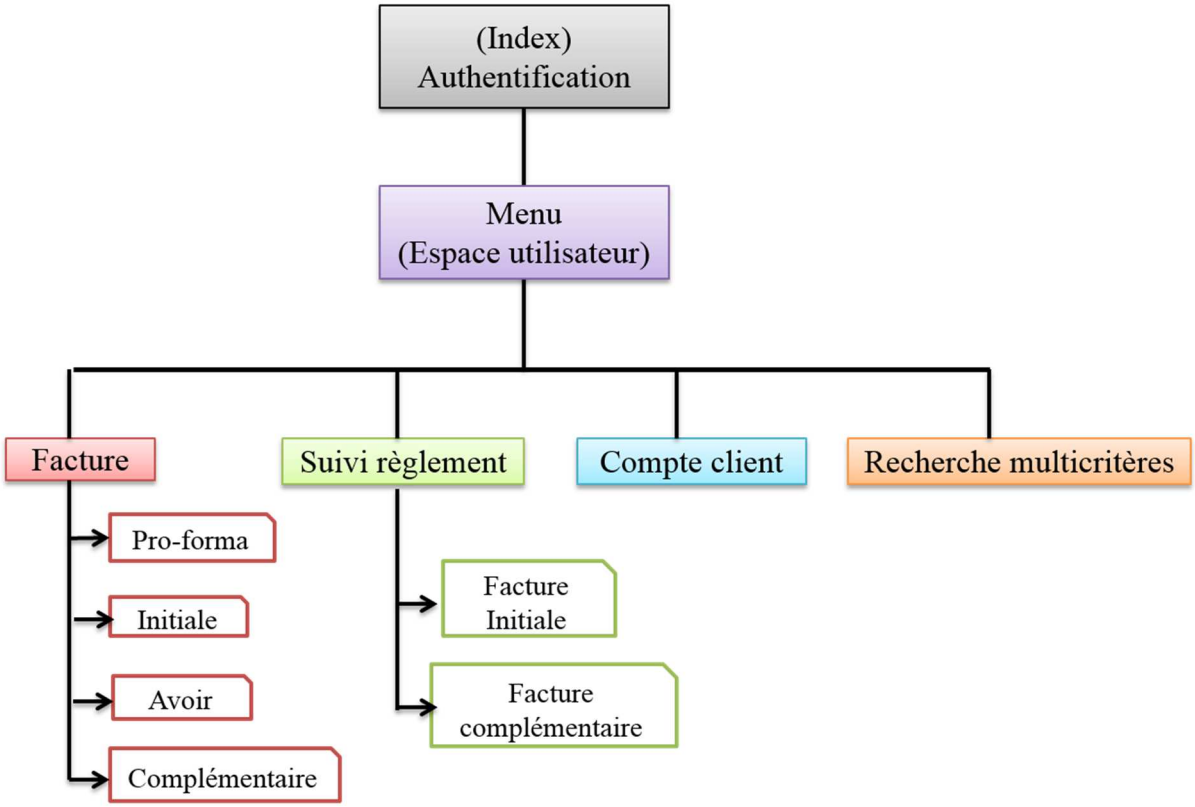


Figure 2:Schéma fonctionnel de STD_fct

2.2. Schéma de la base de donné de l'application

Avant de passé à la partie traitement et présentation de notre application nous avons d'abord créé avec le MYSQL server 2005 la base de données [STD_fct] contenant les différentes tables du chapitre conception.

Nous verrons ci-dessus le schéma de la base de données de notre application après son entière création :

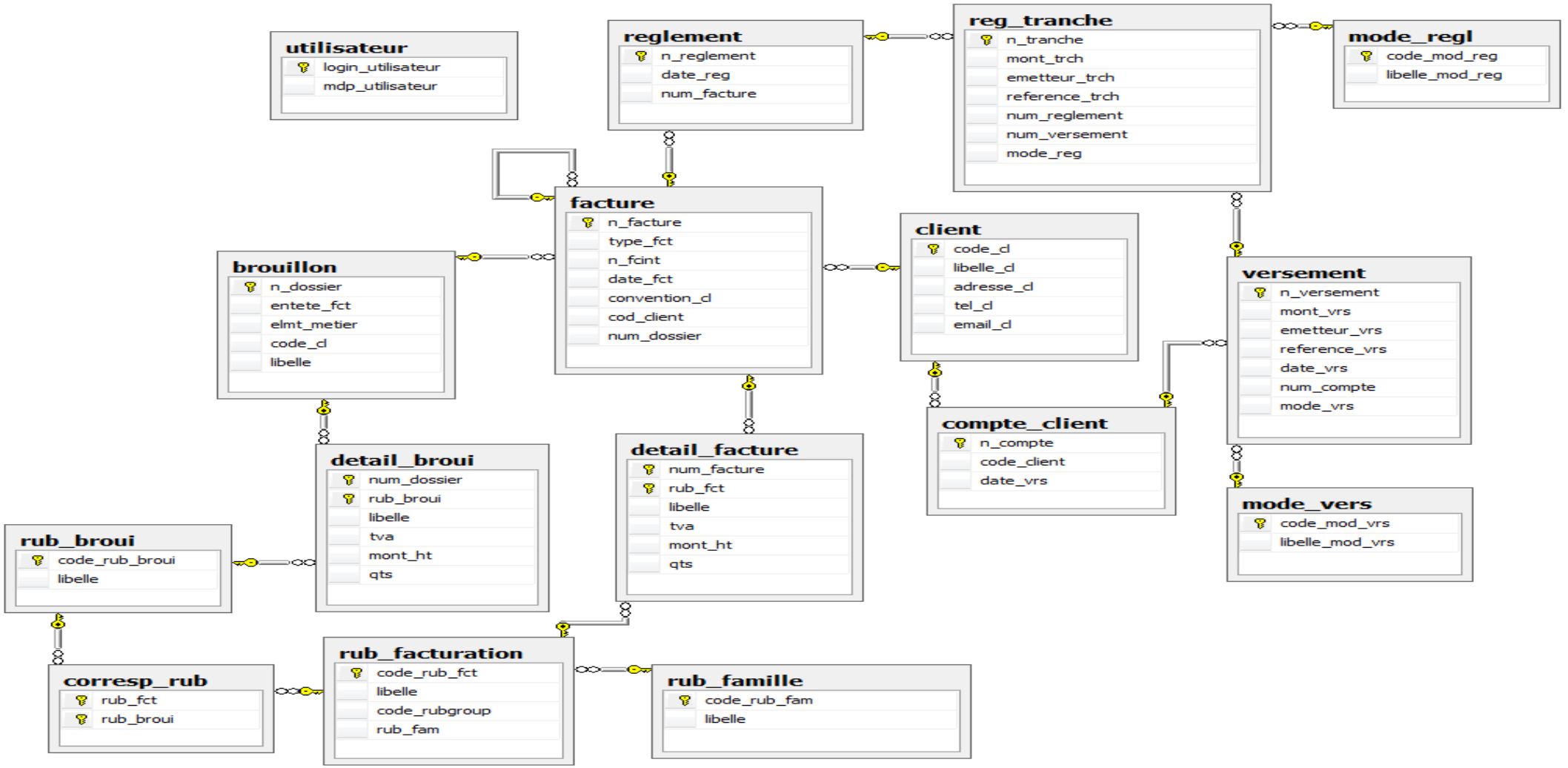


Figure 3: Schéma de la base de donné STD_fct

Remarque :

Après avoir créé cette base de données, on a établi une connexion ECLIPCE/MYSQL serveur à l'aide du pilote JDBC.

2.3. Schéma applicatif de l'application

2.3.1. Schéma applicatif général

Le développement d'une application Java EE implique la création de différent livrables comme le montre le schéma ci-dessus suivant :

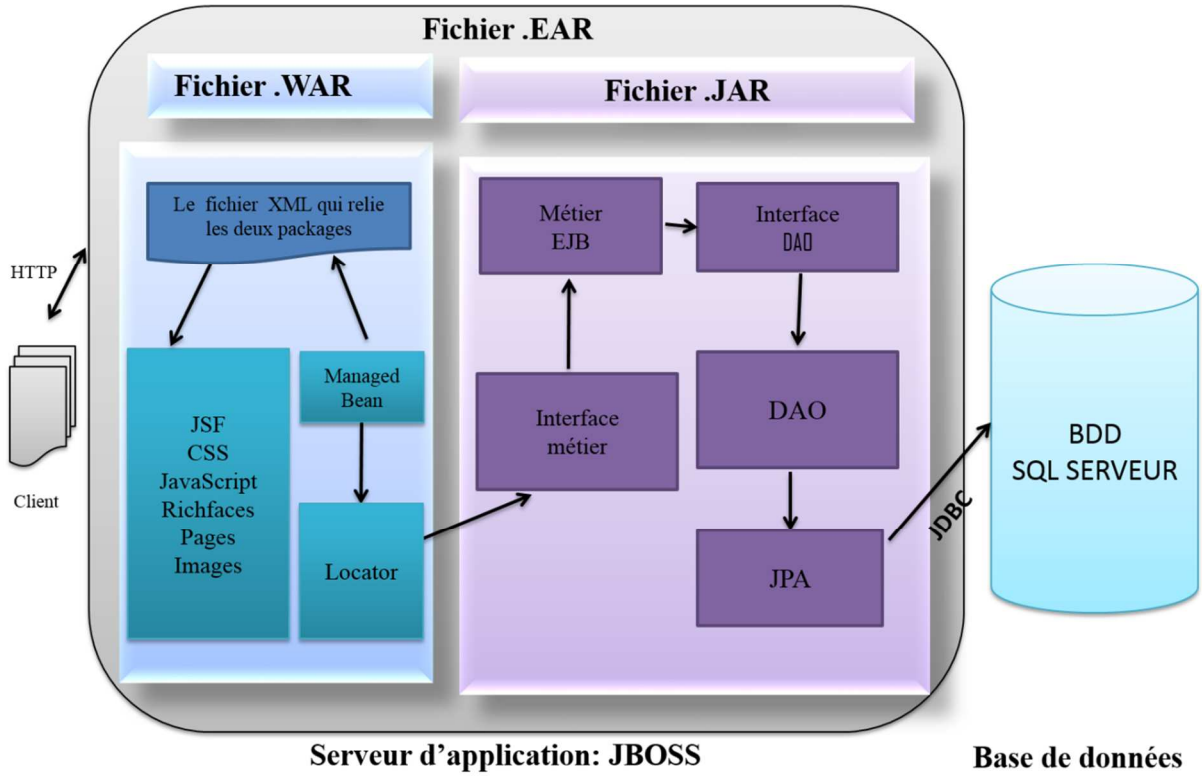


Figure 4 : Schéma applicatif général

I. Livrable EAR (Entreprise ARchive)

Ce format de fichier est utilisé pour emballer tous les composants d'une application Web. Le but de ce format d'archive est de rendre le déploiement plus facile. Ce fichier doit être déployé sur le serveur d'application JEE (JBoss).

II. Livrable JAR (Java ARchive)

Ce livrable contient :

- Les interfaces métier : Elles rendent les méthodes des classes métier visibles par les autres classes qui les utilisent (ManagedBean). Elles représentent la frontière de communication entre les classes métier et la classe Locator.
- Les classes métier : ce sont les EJBs session, elles modélisent les processus métier de notre application (traitements).
- L'interface DAO : Elle permet de publier les méthodes de la classe DAO pour qu'elles soient utilisées par les classes métier.
- La classe DAO : c'est une classe qui implémente les quatre opérations de base pour la persistance des données (JAP), soient : **C**reate, **R**ead, **U**pdate, **D**elete.
- Les classes données (JPA) : ce sont les EJBs entity, elles représentent les données stockées dans la base de données.
- Un fichier de configuration Persistance.xml, permettant la connexion à la base de données.

III. Livrable WAR (Web ARchive)

Ce fichier contient:

- Ecrans de l'application (pages, HTML, JSF).
- Images de l'application.
- Eléments de graphisme (Feuilles de style CSS, JavaScript, Richfaces).
- Classes Java métier (ManagedBean) : qui permet de sauvegarder des informations saisies dans le formulaire, ou récupérer depuis la base de données.
- Locator qui est une classe qui permet la communication entre la couche présentation et la couche métier, plus précisément entre le ManagedBean et couche métier.
- Fichiers de configuration permettant de configurer un service pour lequel la spécification JEE n'est pas précise. On a utilisé deux fichiers de configuration : le fichier web.xml qui sert à configurer l'application web et le faces-config.xml qui contient tous les ManagedBean des formulaires.

2.3.2.Schéma applicatif détaillé de l'application

Dans cette section, nous allons détailler le schéma applicatif de notre application en présentant le contenu des différents packages propre à chaque livrable comme suit :

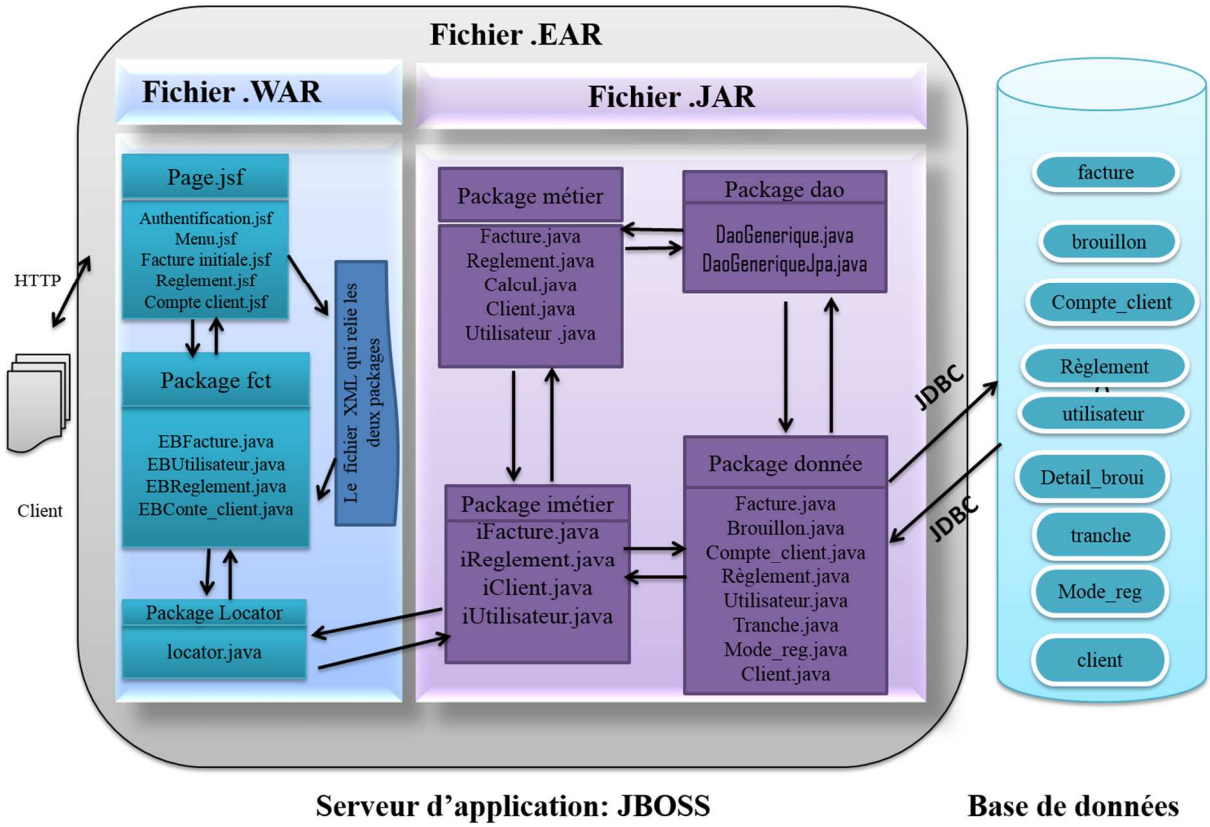


Figure 5 : Schéma applicatif détaillé

Package DAO

Afin de pouvoir exploiter nos objets (donc manipuler nos données), on a créé un package [com.marinesoft.donnee] qui contient : la classe DaoGeneriqueJpa.java qui implémente les quatre opérations de base CRUD et la classe DaoGenerique.java qui est une interface de DaoGeneriqueJpa.java.

Package donnée

Pour donner une image objet à notre base de données, on a créé un package [com.marinesoft.donnee] dans le module EJB, ce package représente les classes de données JPA de livrable .JAR.

Chaque entité de ce package encapsule les lignes d'une table de la base de données, donc on a créé autant d'entités dans ce package que de tables dans la base de données vue précédemment.

Package métier,

On a créé un package [com.marinesoft.metier] qui contient les classes (facture.java, utilisateur.java, reglement.java, compte_client.java…etc), qui servent à renfermer toutes les méthodes de traitement.

Package imetier,

On a créé un package [com.marinesoft.imetier] qui contient les interfaces (ifacture.java, iutilisateur.java, ireglement.java, icompte_client.java …etc.) afin d'assurer la publication de toute les méthodes des classes du package métier.

Package fct,

Afin de pouvoir sauvegarder et récupérer nos données on a créé un package [com.marinesoft.fct] qui contient les classes EBfacture.java, EButilisateur.java, EBreglement.java, EBcompte_client.java.

Package locator

afin d'assurer la communication entre les EB's du package fct et les interfaces imetier du package imetier on a créé un package [com.marinesoft.fct] qui contient la classe locator.java.

Un dossier de nos pages JSF

C'est un dossier qui contient les écrans réalisé de notre application : facture_initiale.jsf, reglement.jsf, compte_client.jsf, Menu.jsf.

Remarque :

La classe calcul est factorisé. Elle est appelée par les classes métier pour calculer :

- ✓ Le montant THT (Total Hors Taxe), le montant TTC (Toutes Taxes Comprises) et le montant total à payer des factures.
- ✓ Le montant TVA, le montant TTC des rubriques de la facture.

2.4. Présentation de quelque interface

L'application que nous avons réalisée se présente sous une fenêtre principale qui permet à l'utilisateur de s'authentifier pour qu'il puisse accéder au menu principal puis aux fonctionnalités que comporte notre application.

À travers les interfaces présentées ci-dessous, nous visons à donner une vue générale de ce notre application conçue.

2.4.1. Interface d'authentification

La figure 6 : illustre l'interface d'authentification qui est la première fenêtre télécharger et visualisé par l'utilisateur. Ce dernier devra taper son nom utilisateur, son mot de passe pour pouvoir accéder à la suite de l'application si les cordonnées (Nom d'utilisateur, Mot de passe, Professionnel) qu'il a saisies sont justes, sinon un message d'erreur s'affichera sur le formulaire.

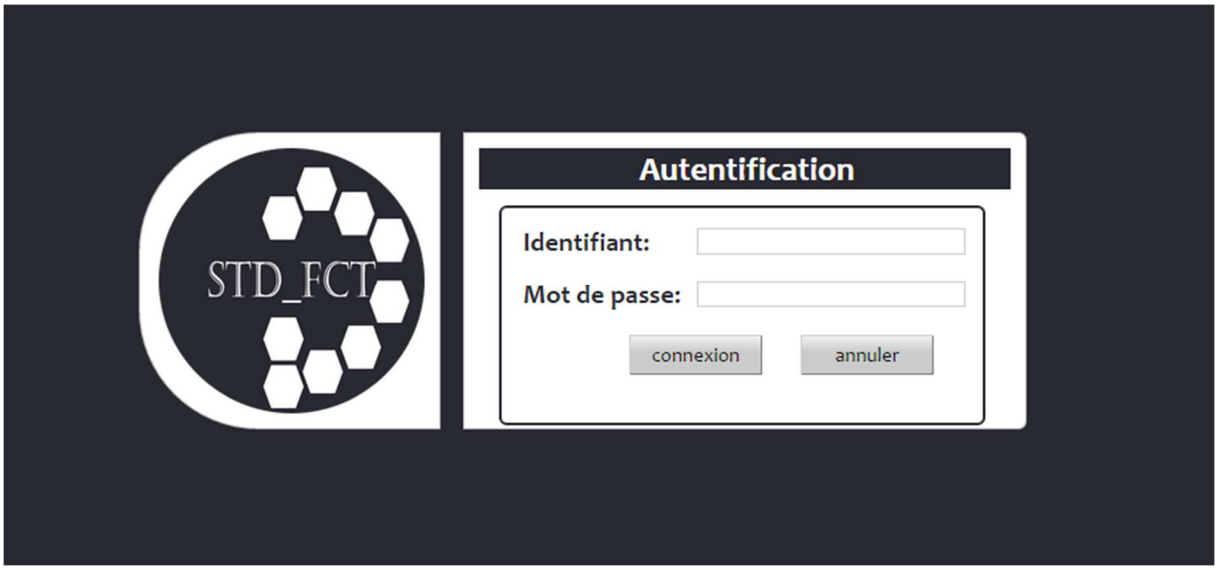


Figure 6 : Interface d'authentification

2.4.2. Interface du menu principale

La figure 7 illustre la page d'accueil de l'espace utilisateur auquel l'utilisateur accède après s'être authentifié, cette page dispose d'une navigation qui emmène dans différentes pages répartie dans 4 rubriques comme suit :

– Facture qui permet l'accès au formulaire des factures pro forma/Initiale/avoir/ complémentaire.

- Suivi des factures à crédit : qui permet une gestion, pour les clients (client conventionné) qui n'ont pas encore réglé leurs factures initiales ou complémentaire.
- Recherche multi critères : outil qui permet de recherche des factures /des règlements/ des comptes client selon différents critères.
- Compte client qui permet à l'utilisateur de créer des comptes pour les clients le souhaitant afin de gérer leurs versements.

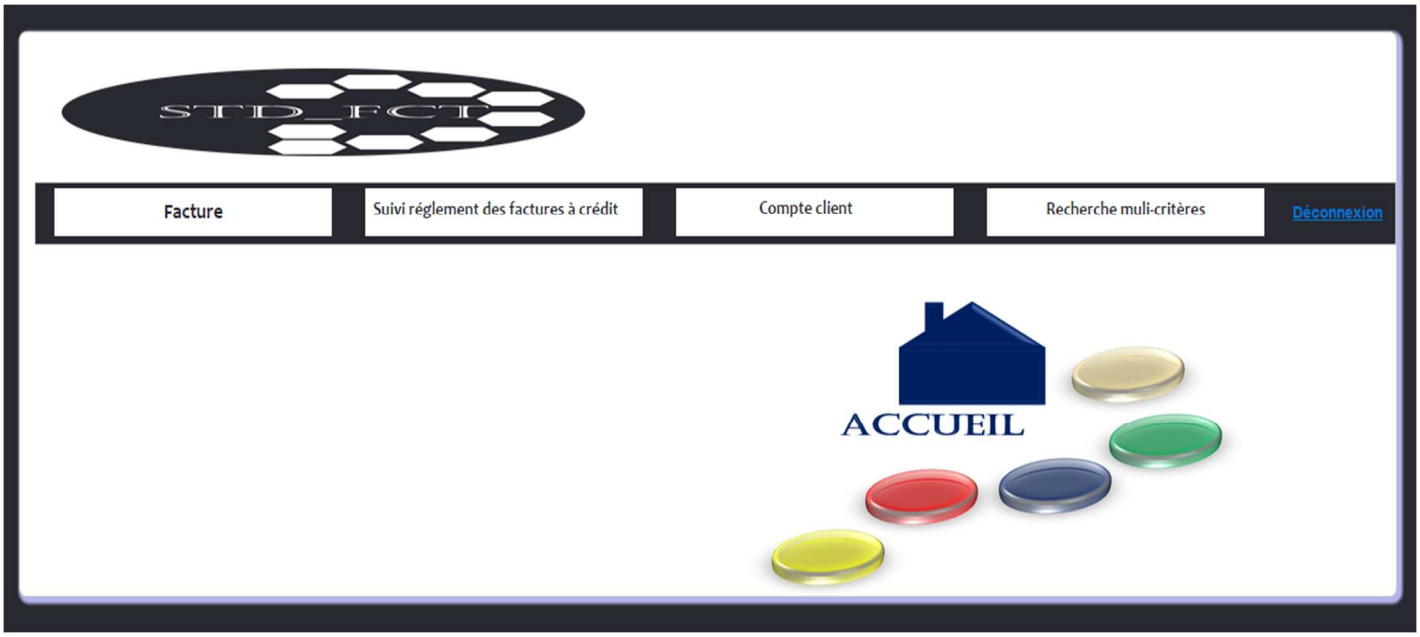


Figure 7 : Interface Menu de l'espace utilisateur

2.4.3. Interface de la facture initiale

La figure 8 illustre le formulaire de la facture initiale qui permettra à l'utilisateur d'établir une facture initiale en remplissant le champ N° de dossier afin de récupérer les informations concernant ce dernier. Si le dossier existe et qu'il n'est pas facturé le bouton régler s'active pour qu'il puisse accéder au règlement

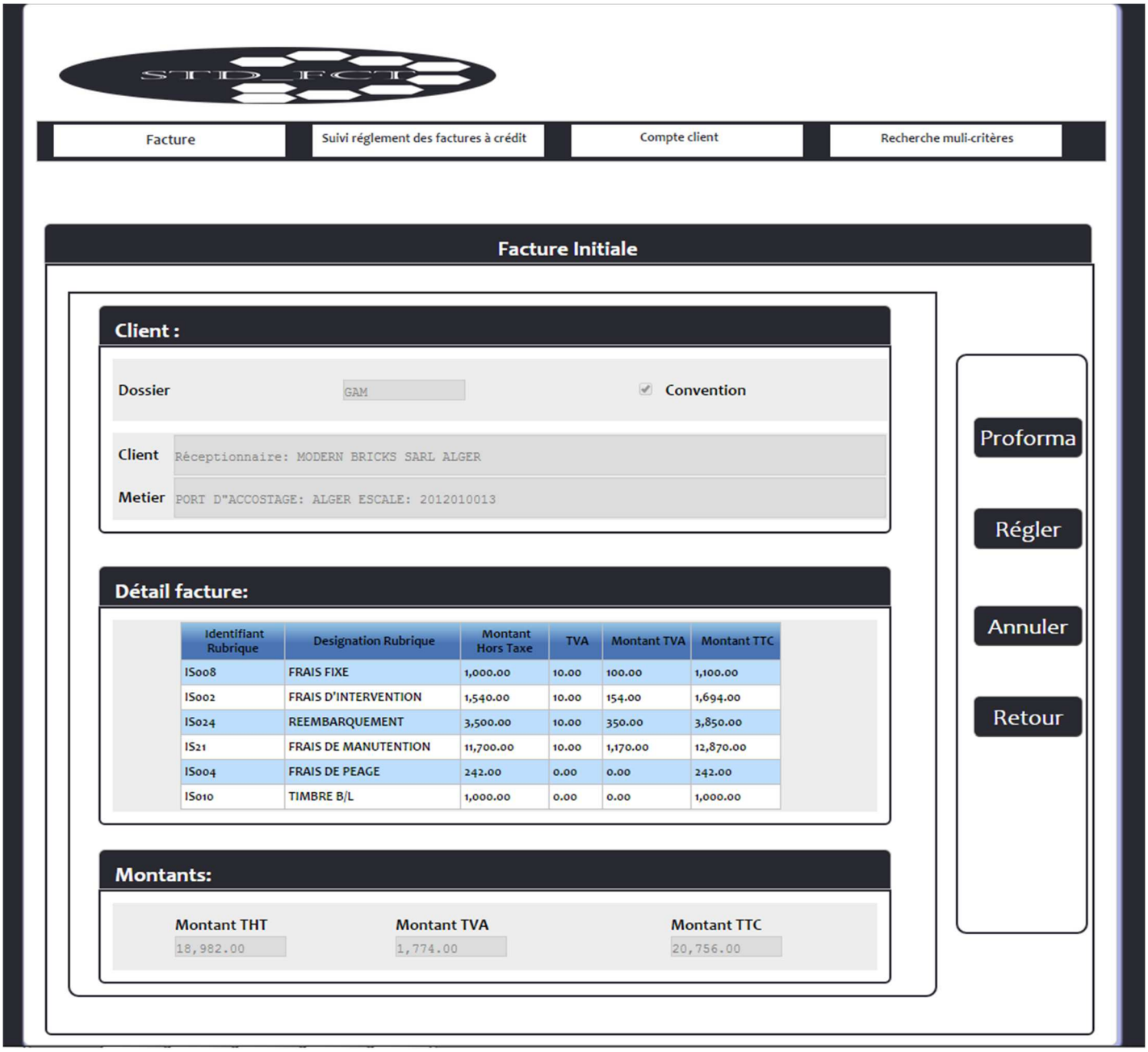


Figure 8 : Interface du formulaire de la facture initiale

2.4.4. Interface du règlement

La figure 9 illustre le formulaire du règlement qui permettra à l'utilisateur de régler sa facture initiale en remplissant toutes les informations nécessaires au règlement. Sachant que le client conventionné a le choix de ne pas régler sa facture avant l'enregistrement. Contrairement au client non conventionné qui doit régler la totalité de sa facture pour qu'il puisse l'enregistrer dans la base de données.

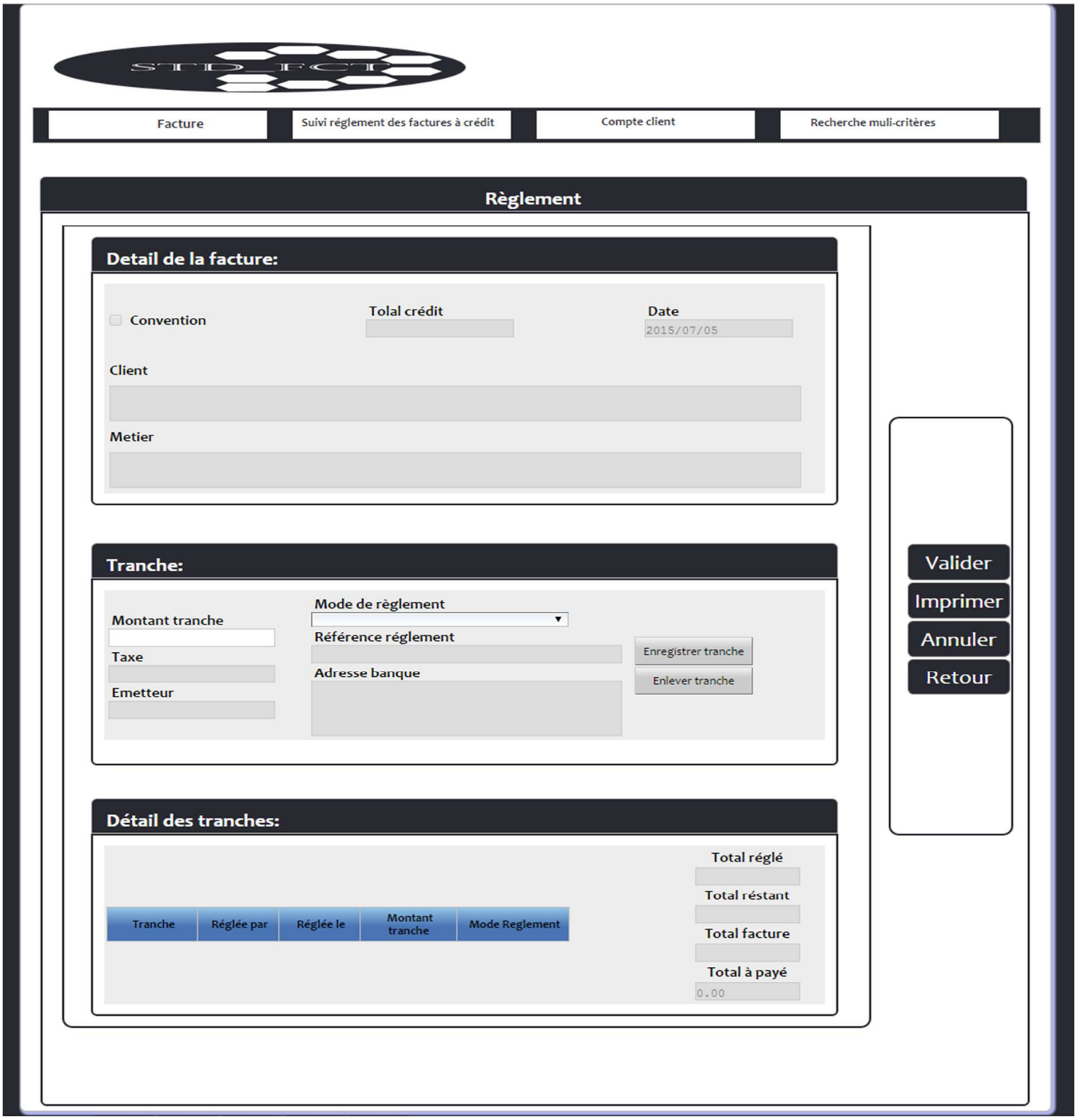


Figure 9: Interface du formulaire de règlement

2.4.5. Interface du compte client

La figure 10 illustre le formulaire du compte client qui permettra à l'utilisateur de créer des comptes pour les clients ayant des versements à effectuer en remplissant le code client afin de récupérer les informations le concernant puis, d'ajouter les versements du client on remplissant l'ensemble des champs nécessaires.

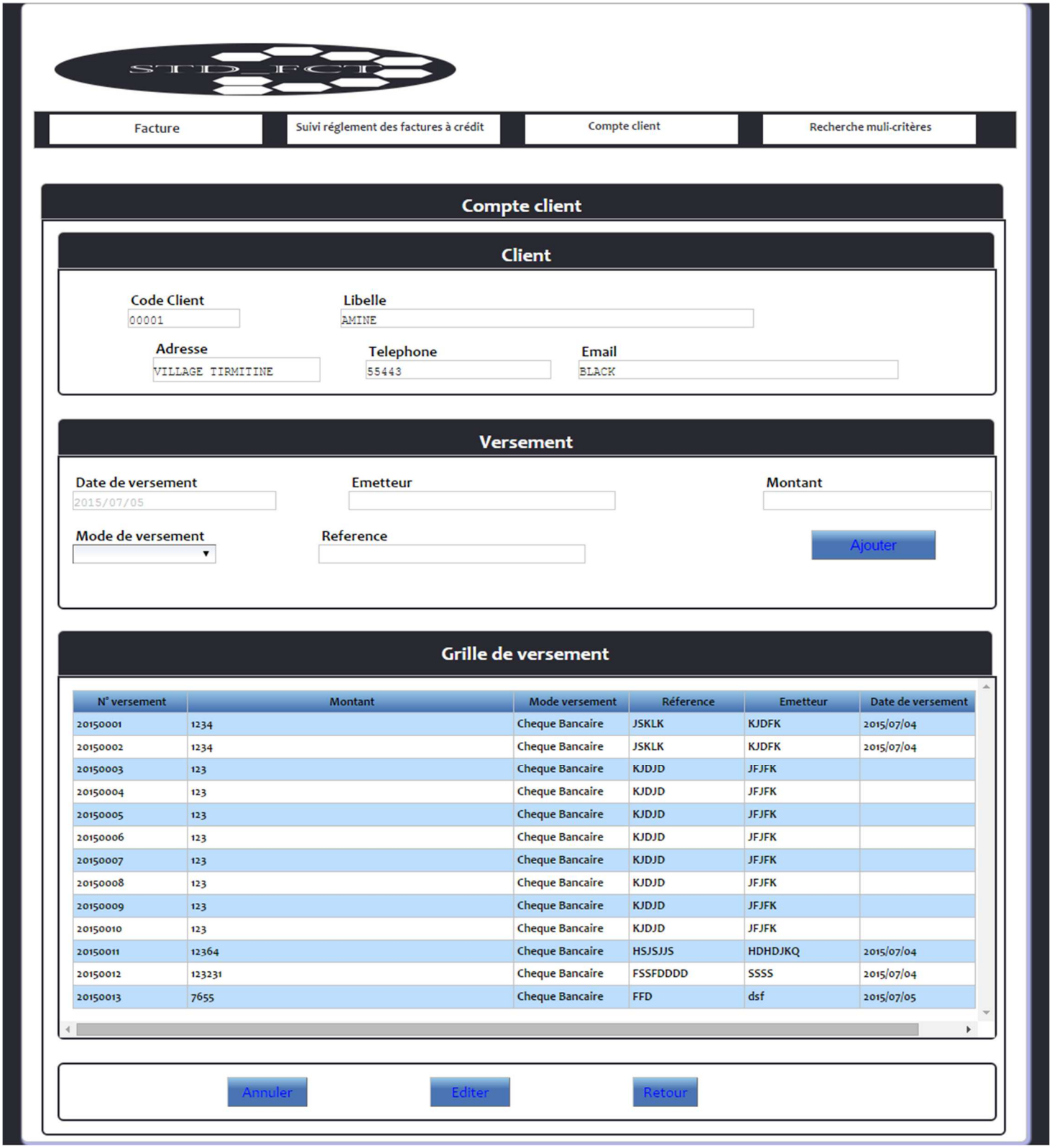


Figure 10: Interface du formulaire du compte client

Conclusion

Dans ce chapitre, nous avons cité en premier lieu les outils et l'environnement de développement de notre application, puis nous avons présenté l'architecture de notre application, en se basant sur une architecture client-serveur. Enfin, nous avons montré le côté client avec une brève explication à chaque interface.

Conclusion générale

Toute entreprise, quelle que soit sa vocation et son caractère, doit se mettre à jour de la progression technologique et faire face par l'automatisation de ses structures et la formation de son personnel afin d'améliorer son rendement et son service et d'assurer sa place sur le marché.

Notre travail s'inscrit dans cette allure fondant sur une démarche de conception et réalisation d'une application web 3tiers pour la facturation standard. Un noyau dédie au CRM/ERP.

Le stage que nous avons effectué dans l'enceinte de la structure de la direction recherche et développement de Marine soft, nous a permis de découvrir et d'acquérir des connaissances sur sa gestion. Ce projet à travers les étapes de mise en œuvre notamment l'analyse, la conception et la réalisation, nous a conduit à l'application du formalisme de conception « UML » et des concepts de réalisation « JEE, Java, SQL Serveur » avec tous leurs outils et fonctions et de se familiariser avec eux.

Notre application offre plusieurs services et rend aisée la tâche de facturation, à titre d'exemples :

- La gestion de processus de facturation, partant de la facture pro-forma, la facture initiale et le suivi des factures a crédit ainsi que la présence de concept « Compte client » comme un outil de gestion des relations clients.
- Séparation de la logique métier du domaine de la facturation ce qui ouvre la porte à une grande masse d'utilisateurs.
- La centralisation de la BDD ainsi que le développement en 3tiers facilite la maintenance de l'application.

En fin, nous espérons que ce travail sera d'un grand intérêt pour Marine soft et un guide pour les nouvelles promotions. Alors que ce travail n'est pas une fin en soi. Il peut toutefois être complété. En effet, il ne couvre qu'une partie des fonctionnalités conçues. De ce fait, des perspectives d'améliorations suivantes sont ouvertes :

- La gestion de menu : à travers d'une base de données dédié au « menu ».
- La segmentation des données : cette fonctionnalité nous permettra d'avoir un contrôle sur le travail des utilisateurs dans la base des données.
- L'implémentation d'une couche de communication avec les autres modules de l'ERP.

- L’implémentation d’une couche de communication avec un logiciel de comptabilité pour le CRM.

Liens

[MH]-Many How-l'histoire du logiciel CRM-consulter en Mars 2015-disponible sur : http://fr.many-how.com/article/logiciels/autres-logiciels/lhistoire-du-logiciel-crm.php

[FAB 07]-FleurèAnne Blain-qu'est-ce qu'un CRM-editer en avril 2007 consulter en Mars 2015-disponible sur : http://fablain.developpez.com/tutoriel/crm/presentcrm/

[E.Raouia]-Raouia Elhakimi-y a-t-il une différence entre la théorie du CRM et sa pratique ?-consulter en Mars 2015-disponible sur : http://www.marketing-etudiant.fr/docs/64df28a1ebfc9ec85578bb198e53a451-cmp.pdf

[SH]-Selim Halioui-CRM & Architecture centrée client-consulter en Mars 2015-disponible sur : http://www.madwatch.net/dossier/dossier_crmacc.pdf

[WIKI 15]-Wikipédia L'encyclopédie libre -progiciel de gestion intégré - mise a jour 4 Février 2015 -consulter Mars 2015 - disponible sur http://fr.wikipedia.org/wiki/Progiciel_de_gestion_int%C3%A9gr%C3%A9

[FAB 06]-Fleur-Anne Blain-Présentation général des ERP et leur architecture modulaire - publier le 7 Novembre 2006-consulter en Mars 2015 - disponible sur : http://fablain.developpez.com/tutoriel/presenterp/

[PN 10]-Philippe Norgeon - cours PGI, ERP (version PDF) - publier en 2010-consulter Mars 2015 -disponible sur http://www.guillaumeriviere.name/estia/si/pub/cours_ERP_PGI_2010.pdf

[KL 15]- Kristie Lorette -what is the relationship between ERP and CRM- mise a jour en 2015- consulter Mars 2015-disponible sur : http://www.wisegeek.com/what-is-the-relationship-between-erp-and-crm.htm

[AQ]-Axess Qualité le partenaire de votre performance-consulter Mars 2015-disponible sur : http://www.axess-qualite.fr/approche-processus.html

[CAPITALRH]-Capitalrh-l'approche processus-consulter Mars 2015-disponible sur : http://www.capitalrh.fr/L-approche-processus_a117.html

[AC 10]- L'amélioration continue-L'approche par processus comme outil de management-publier en 21 juillet 2010 consulter Mars 2015-disponible sur : http://www.ameliorationcontinue.fr/approche-processus/

BIBLIOGRAPGIE

[HB JPW 03]-Hans BRANDENBURG, Jean-Pierre WOJTYNA- L'APPROCHE PROCESSUS, mode d'emploi-publié en 2003 -consulté Mars 2015-disponible sur: http://www.eyrolles.com/Chapitres/9782708128880/chap1_Brandenburg.pdf

[LA 15]-Laurent Audibert - UML 2 : de l'apprentissage à la pratique –Mise a jour en 2015- Consulter janvier 2015- Disponible sur : http://laurent-audibert.developpez.com/Cours-UML/?page=introduction-modelisation-objet#L1-4

Ouvrage

[JC O5]-Joui Cyril-Introduction à J2EE : connaitre l'environnement J2EE-editer en Sep 2005, Consulté Mars 2015.

[JP 05]-Jean philippe-Pefactoring des applications Java/J2EE-editer en 2005-consulté Mars 2015.

[EP 05]-Emmanuel Puybaret -Java 1.4, les cahiers de programmeur-éditer en2005-consulté Mars en Juin 2015.

[G. Pierre]-Pierre Giraud-Appernez à coder en HTML et en CSS-consulter en Mars 2015

[GA 07]-Antonio Goncalves-Cahier des programmeurs JEE 5-editer en Mai 2007-consulté en Juin 2015

[OSG 14]-Open source guide-Sortie de RichFaces 4.5.1-editer en Décembre 2014-consulté en Juin 2015.

Document

[D-MS]-Documents de Marine Soft

ANNEXE

Annexe 1 : Architecture client /serveur

Annexe 2 : UML

Annexe 3 : JEE

Introduction

Dans l'informatique moderne, de nombreuses applications fonctionnent selon un environnement client-serveur; cette dénomination signifie que des machines clientes (faisant partie du réseau) contactent un serveur, une machine généralement très puissante en termes de capacités d'entrées sorties, qui leur fournit des services.

Aujourd'hui, la popularité du client-serveur ne cesse de croître et on assiste à un mouvement dit l'informatique client-serveur qui est extrêmement actif et qui remodèle l'utilisation de l'ordinateur.

Ce chapitre est un rappel de principe de fonctionnement du modèle client-serveur, mettre en évidence de ses avantages et ses limites. Nous verrons aussi qu'aujourd'hui les fonctions sont distribuées entre le client et le serveur, et que ce partage de fonctionnalités conduit à une nouvelle orientation du modèle client-serveur qui sera dit orienté client ou orienté serveur (plus répandu).

Partant de ce principe de séparation de fonctionnalités, on est passé des applications monolithiques aux applications client-serveur deux-tiers pour arriver aujourd'hui aux applications basées sur des architectures client/serveur multi-tiers. Enfin, nous verrons que l'architecture multi-tiers (dite aussi à base de composants) est la plus adéquate pour le développement des applications complexes qui ne peuvent plus être mises en place avec le modèle classique.

1. Présentation du concept client /serveur

Dans un monde ou la course à la productivité conduit les technologies à évoluer de plus en plus vite, le client-serveur s'est taillé une part de choix depuis le début des années 1990. En effet, il faut pouvoir disposer de systèmes d'information évolutifs permettant une coopération fructueuse entre les différentes entités de l'entreprise. Les systèmes des années 70 et 80 ne répondaient pas à ces exigences.

1.1. Origines et histoires

Avant 1980

- Les architectures étaient centralisées autour des calculateurs centraux (une unité autour sont relié des terminaux) mainframe, par exemple de type IBM.

- Les terminaux étaient passifs ayant des interfaces, caractères ou pages.
- La productivité des développeurs restait faible.
- La maintenance des applications était des plus difficiles.
- Les utilisateurs restaient prisonniers des systèmes propriétaires.

De 1980 jusqu'à 1990

- Les développements des transactionnel et des bases de données.
- Les systèmes ont commencé à migrer depuis des systèmes propriétaires vers des systèmes plus ouverts type Unix.
- Les bases de données relationnelles ont vu le jour accompagnées de langage de développement construits autour des données.
- SQL s'est imposé comme la norme d'accès aux données.
- Les réseaux notamment les réseaux locaux se sont développés.
- Les micro-ordinateurs se sont imposés dans les entreprises et ils ont apportés des interfaces graphiques conviviales.

De 1990 jusqu'à 2000

- Évolution des réseaux d'entreprise autour des serveurs ouverts offrant des interfaces standards pour permettre une connectivité avancée.
- Les vitesses de calcul des micros deviennent impressionnantes, le graphique est partout aux niveaux des interfaces, le besoin de partage des donnés devient essentiel aussi bien que pour l'accès transactionnel en temps réel, que pour l'accès décisionnel.

1.2. Définition

L'architecture client-serveur (C/S) est un modèle de fonctionnement logiciel qui peut se réaliser sur tout type d'architecture matérielle (petite ou grosse machine), à partir du moment où ces architectures peuvent être interconnectés (s'articule en général autour d'un réseau).

On parle de fonctionnement logiciel dans la mesure ou cette architecture est basée sur l'utilisation de deux types de logiciels, à savoir un logiciel serveur et un logiciel client s'exécutent normalement sur deux machines différentes. L'élément important dans cette architecture est l'utilisation de mécanismes de communication entre ces deux applications.

Une autre définition plus large peut être proposée :" Modèle d'architecture applicative ou les programmes sont répartis entre les deux processus clients et serveurs communiquant par des requêtes et des réponses".

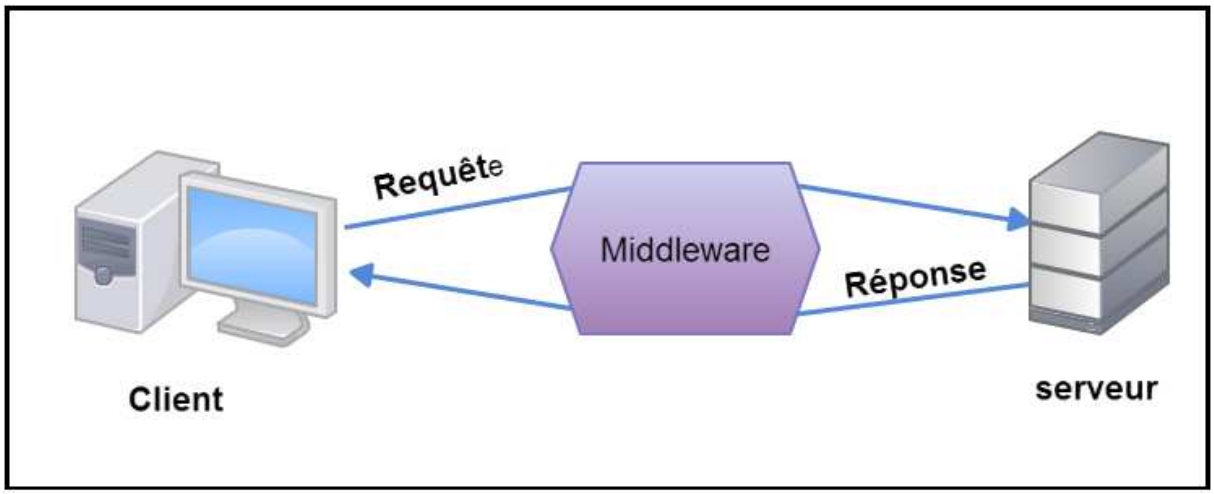


Figure 1 : Modèle client/serveur

1.3. Les trois (3) niveaux d'abstraction du modèle C/S

Dans l'architecture client-serveur, une application est constituée de trois couches:

a. Couche de présentation

Nommée aussi IHM (Interaction Homme Machine), cette couche permet l'interaction de l'application avec l'utilisateur, elle gère les saisies sur clavier, les actions avec la sourie et la présentation des informations à l'écran.

b. Couche applicative

Elle regroupe deux types de traitement :

- Les traitements locaux qui sont les contrôles effectués aux niveaux du dialogue avec l'IHM, ils assurent la cohérence des informations entre la couche présentation et les traitements globaux.

- Les traitements globaux représentent la couche métier de l'application, on parle parfois de logique métier ou business logique qui regroupe les traitements internes qui régissent une entreprise donnée.

c. Couche des données

Plus exactement celle de l'accès aux données, elle regroupe l'ensemble des mécanismes qui permettent la gestion des informations stockées par l'application.

Remarque:

- Ces trois (3) niveaux peuvent être imbriqués ou repartis de différentes manières entre plusieurs machines physiques.

- Le découpage et la répartition d'une application permettent de distinguer les architectures applicatives suivantes: l'architecture 1-tiers, l'architecture 2-tiers, l'architecture 3-tiers et les architectures n-tiers.

1.4. Acteurs composent le modèle C/S

Dans un modèle client-serveur on peut distinguer deux acteurs**:**

a. Client :

i. Définition

Programme demandant un service à un autre programme par l'envoi d'un message (requête) contenant le descriptif de l'opération à exécuter et attendant une réponse à cette opération par un message en retour.

ii. Type de client

- **Client léger**

Le poste client accède à une application située sur un ordinateur dit "serveur" via une interface et un navigateur Web. L'application fonctionne entièrement sur le serveur, le poste client reçoit la réponse "toute faite" à sa demande qu'il a formulé. (Appelée : "requête").

- **Client lourd**

Le poste client doit comporter un système d'exploitation capable d'exécuter en local une partie des traitements. Car le traitement de la réponse à la requête du client utilisateur va mettre en œuvre un travail combiné entre l'ordinateur serveur et le poste client.

- **Client riche**

Une interface graphique plus évoluée permet de mettre en œuvre des fonctionnalités comparables à celles d'un client "lourd". Les traitements sont effectués majoritairement sur le serveur, la réponse "semi-finie" étant envoyée au poste client, où le client "riche" est capable de la finaliser et de la présenter.

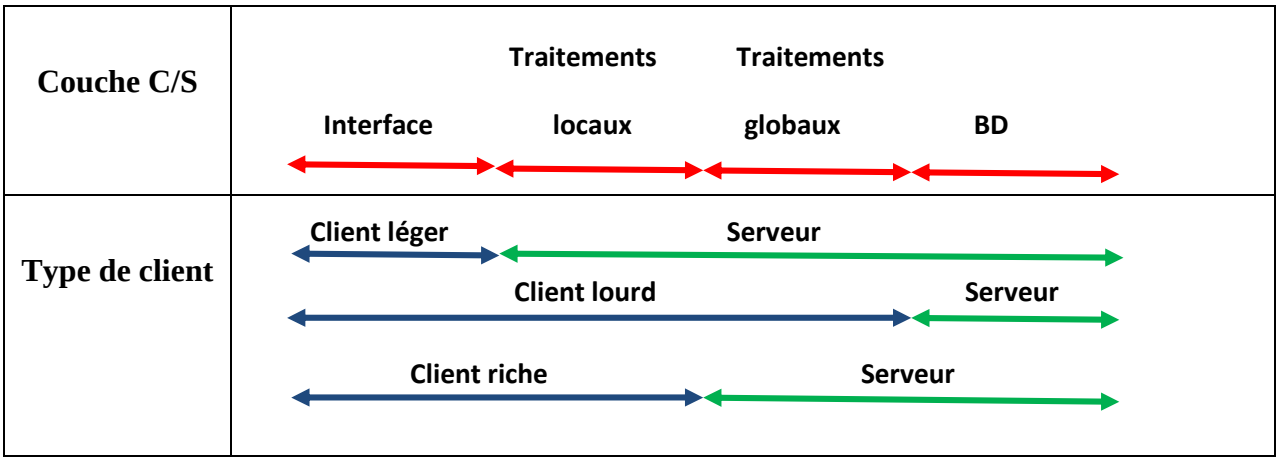


Figure 2: Représentation graphique des types de client

b. Serveur :

i. Définition:

Programme fournissant des services à d'autre programme ou bien, processus accomplissant une opération sur demande d'un client et transmettant la réponse à ce client.

ii. Type de serveur

- **Serveurs de fichiers :**

Un serveur de fichier est utilisé pour le stockage et la gestion des fichiers utilisateurs, ce type de serveur a souvent une grande capacité de stockage sur son espace disque. Ainsi, n'importe quel utilisateur connecté au réseau qui émet des requêtes en direction du serveur peut récupérer un ou plusieurs fichiers grâce à un Protocole.

Il doit être fiable, performant, autorise un fonctionnement permanent, dispose de possibilités d'extensions, permet les changements des disques sans interruption de fonctionnement du serveur.

- **Serveurs de base de données :**

Un serveur de base de données sert à stocker, à extraire et à gérer les données dans une base de données. Il permet également de gérer la mise à jour des données. Il donne un accès simultané à cette base a plusieurs serveurs et utilisateurs. Enfin, il assure la sécurité et l'intégrité des données. Et quand on parle de données, on entend peut-être des millions d'éléments simultanément accessibles par des milliers d'utilisateurs.

- **Les serveurs d'applications**

Les serveurs d'applications sont des logiciels occupants la couche centrale dans une architecture multicouche, qu'elle soit 3-tiers (postes clients, serveur d'applications, serveur de données) ou étendue (n-tiers).

Dans un sens plus large, un serveur d'application peut être une machine servant à héberger des applications, soit pour permettre leur exécution depuis un poste client (mode client serveur de données, généralement partage de fichiers et politiques de gestion des accès), soit pour déporter l'affichage sur le poste client (mode client serveur d'affichage).

- **Serveur groupware**

Un groupware serveur est un serveur informatique utilisé comme une connexion pour divers clients qui l'utilisent pour héberger et partager des fichiers dans le cadre d'un environnement de travail collaboratif. Le nombre de clients connectés à ce serveur dépend généralement de la portée et de la nature du projet. Depuis cela fait partie d'un projet de groupware, il y a généralement des logiciels installés sur les différents ordinateurs clients pour permettre une meilleure communication entre les clients et l'accès au serveur. Un serveur de groupware peut être utilisé pour réduire les communications inutiles ou répétitives entre les membres de l'équipe et d'accroître la productivité.

Remarque:

- Les clients et le serveur doivent utiliser le même protocole.
- Un serveur peut répondre à plusieurs clients simultanément.

1.5. Caractéristique des modèles C/S

Les propriétés qui caractérisent l'architecture client-serveur des autres systèmes informatiques réparties sont :

- **Le service :** comportement d'un programme qui peut rendre service à d'autre programmes. Un service est appelé par une requête suivant un certain protocole tel que le serveur est un fournisseur de service alors que le client est un consommateur de service.
- **Messages :** Les messages sont les moyens d'échanges entre le client et le serveur.
- **Partage de ressources :** Un serveur traite plusieurs clients et contrôle leurs accès aux ressources.
- **Protocole asymétrique :** Conséquence du partage de ressources, le protocole de communication est asymétrique, le client déclenche le dialogue ; le serveur attend les requêtes des clients.

- **Transparence de la localisation :** L'architecture client -serveur doit masquer au client la localisation du serveur (que le service soit sur la même machine ou accessible par le réseau). Transparence par rapport aux systèmes d'exploitation et aux plates-formes matérielles. Idéalement, l'application client serveur doit être indépendante de ces deux éléments.
- **Encapsulation des services :** Un client demande un service. Le serveur décide de la façon de lui rendre une mise à niveau du logiciel serveur doit être sans conséquence pour le client tant que l'interface message est identique.
- **Évolution :** Une architecture client-serveur doit pouvoir évoluer horizontalement (évolution du nombre de clients) et verticalement (évolution du nombre et des caractéristiques des serveurs).
- **L'adaptabilité** : des machines peuvent être ajoutées ou retirées du réseau; les performances des machines peuvent aussi évoluer.
- **Redimensionnement** : il est possible d'ajouter et de retirer des stations clientes. Il est possible de faire évoluer les serveurs.
- **Intégrité :** les données du serveur sont gérées sur le serveur d'une façon centralisée. Les clients restent individuels et indépendants.

1.6. Les différents modèles de C/S

Le type du modèle client/serveur dépend essentiellement des services assurés par le serveur ainsi on peut distinguer :

a. C/S de présentation:

Le principe de ce modèle de C/S est que la représentation de l'interface graphique est prise en charge intégralement par le serveur.

Ce type d'application présentait jusqu'à présent l'inconvénient de générer un fort trafic réseau et de ne permettre aucune répartition de la charge entre client et serveur.

b. C/S de traitement:

Ce type est fondé sur la base de la répartition des traitements entre le serveur et le client.

Dans ce cas, le module client envoie des requêtes mais aussi des appels de procédures au module serveur qui les exécute et envoie le résultat.

Sur le serveur, les traitements (procédure) n'est pas obligatoirement associé la partie donnée.

c. C/S de données:

Repose sur le principe que le serveur abrite la gestion des données (c'est à dire que le serveur gère les données, leur intégrité, la sécurité...etc) et il envoit seulement les données correspondantes à la requête émise par le client qui abrite de son côté la partie traitement et présentation qui traite ces données.

C'est sans doute le plus connu et le plus répondu des modèles client-serveur, qui est utilisé par tous les grands SGBD.

2. Les architectures client serveur

2.1. Architecture 2-tiers

Pour résoudre les limitations survenues dans les architectures 1 tiers centralisé et repartie tout en conservant leurs avantages, on a scindé les applications en deux parties distinctes et coopérantes qui donnent naissances aux architectures 2 -tiers appelées aussi client-serveur de premier génération ou C/S de données.

Dans l'architecture 2-tiers, la gestion des données est prise en charge par un SGBD centralisé, s'exécutant le plus souvent sur un serveur dédié alors que la logique applicative est soit entièrement coté client intégrée avec la présentation , soit entièrement coté serveur (la partie client ne gère que l'interface utilisateur) , soit découpée entre la partie serveur et la partie client.

Le dialogue entre le client et le serveur se résume à l'envoi des requêtes et au retour des réponses.

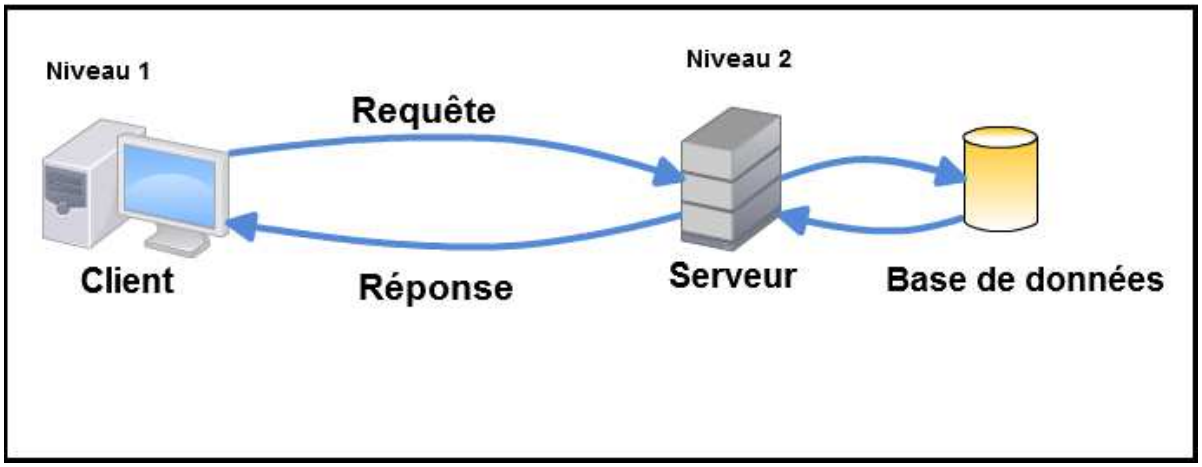


Figure 3 : Architecture client/serveur à 2 niveaux

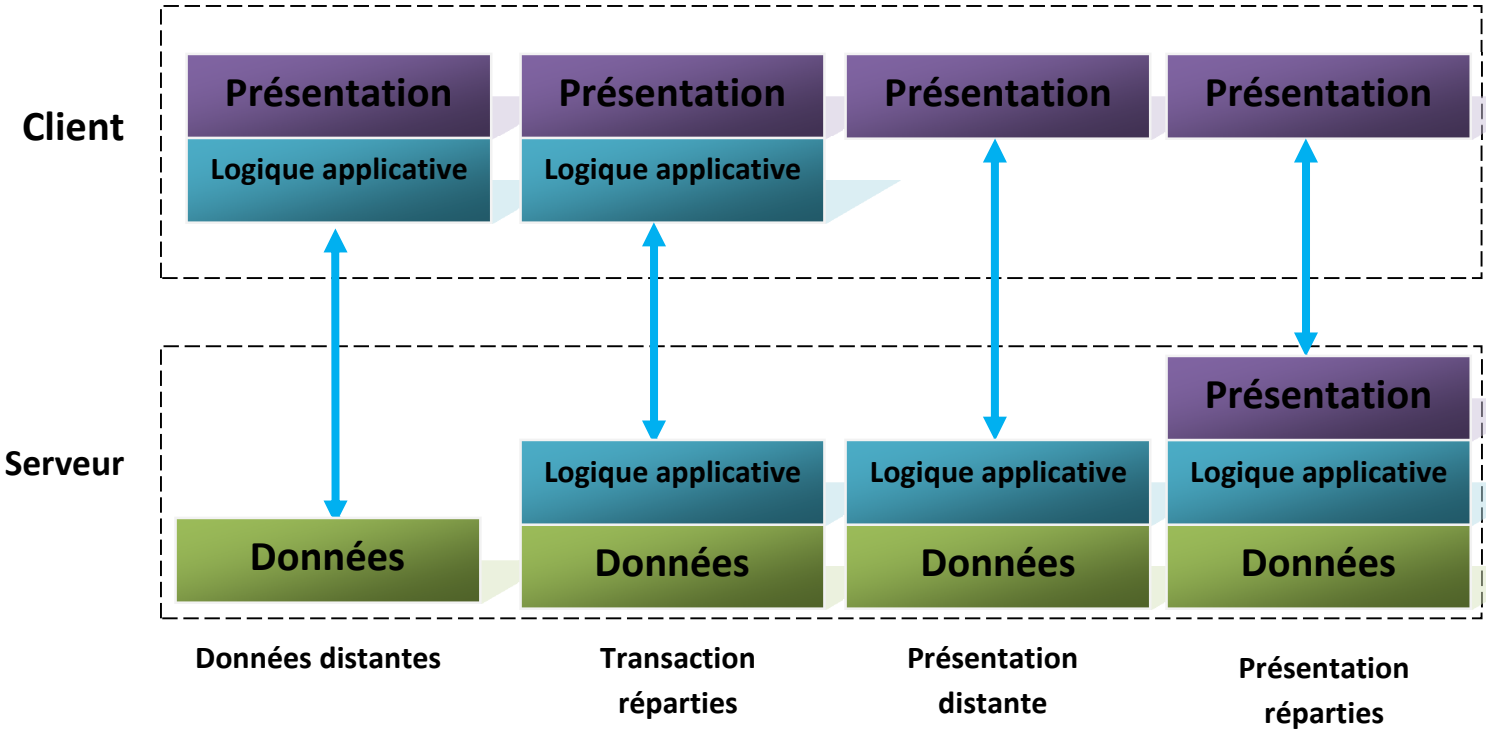


Figure 4 : Modèle de Gartner pour les systèmes a 2 niveaux (2-tiers)

a. Avantages

Ce type d'architecture offre

- ✓ Une interface utilisateur riche
- ✓ Des données centralisées
- ✓ L'efficacité pour un nombre réduit de client

b. Limites

On peut déceler un grand nombre d'inconvénients dans cette architecture, qui proviennent pour la plus part du type de client utilisé:

- ✓ Le problème du client dit obèse (lourd). En effet, on ne peut pas soulager la charge du poste client, qui supporte la grande majorité des traitements applicatifs.
- ✓ Une autre limitation est que chaque intervention engendre un coût important, en particulier pour la mise à jour des applications clientes.
- ✓ De même, la relation étroite, qui existe entre le programme client et l'organisation de la partie serveur, complique les évolutions de ce dernier.
- ✓ Enfin, la conversation entre le client et le serveur est assez bruyante et s'adapte mal a des bandes passantes étroites, la croissance du nombre du client est limité, car la performance se dégrade et la base de données est en surcharge.

Dans le but de passé au-dessus des faiblesses engendré par le client-serveur 2-tiers tout en conservant ces avantages, on a cherché une architecture plus évoluée qui facilite les forts déploiements à moindre coût, la réponse est apportée par les architectures 3-tiers.

2.2. Architecture 3-tiers

Dans l'architecture trois tiers, appelée aussi C/S de deuxième génération ou architecture à 3 niveaux contrairement au C/S 2-tiers, un niveau supplémentaire est ajouté entre les deux niveaux précédents, permettant de séparer les traitements de l'interface graphique et du serveur de base de données.

Il s'agit d'un modèle logique d'architecture applicative qui vise à modéliser une application comme un empilement de trois couches logicielles (étages, niveaux, tiers) dont le rôle est clairement défini :

- **la présentation :** des données correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur.
- **le traitement métier des données** : correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative.
- **L'accès aux données persistantes** : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

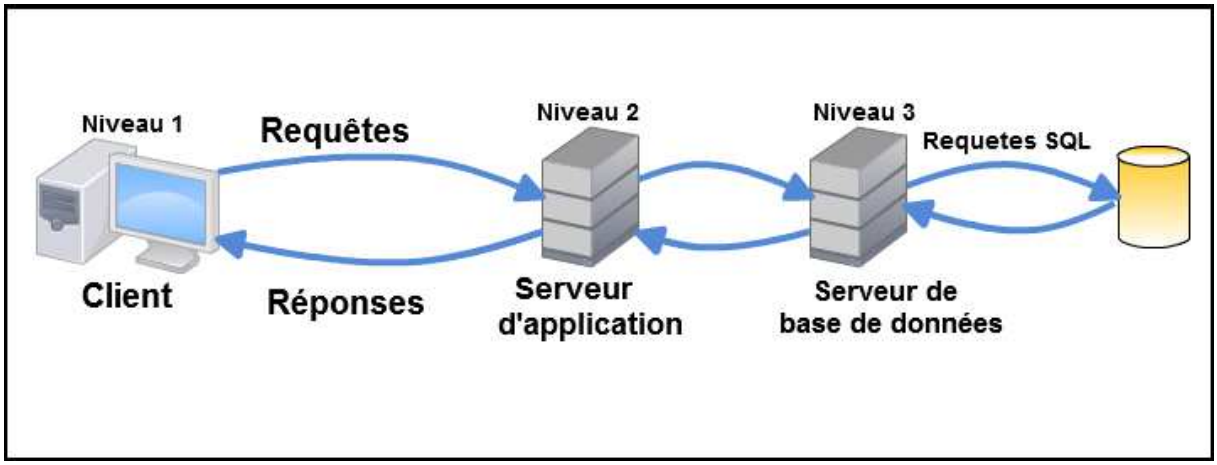


Figure 5 : Architecture client/serveur à 3 niveaux

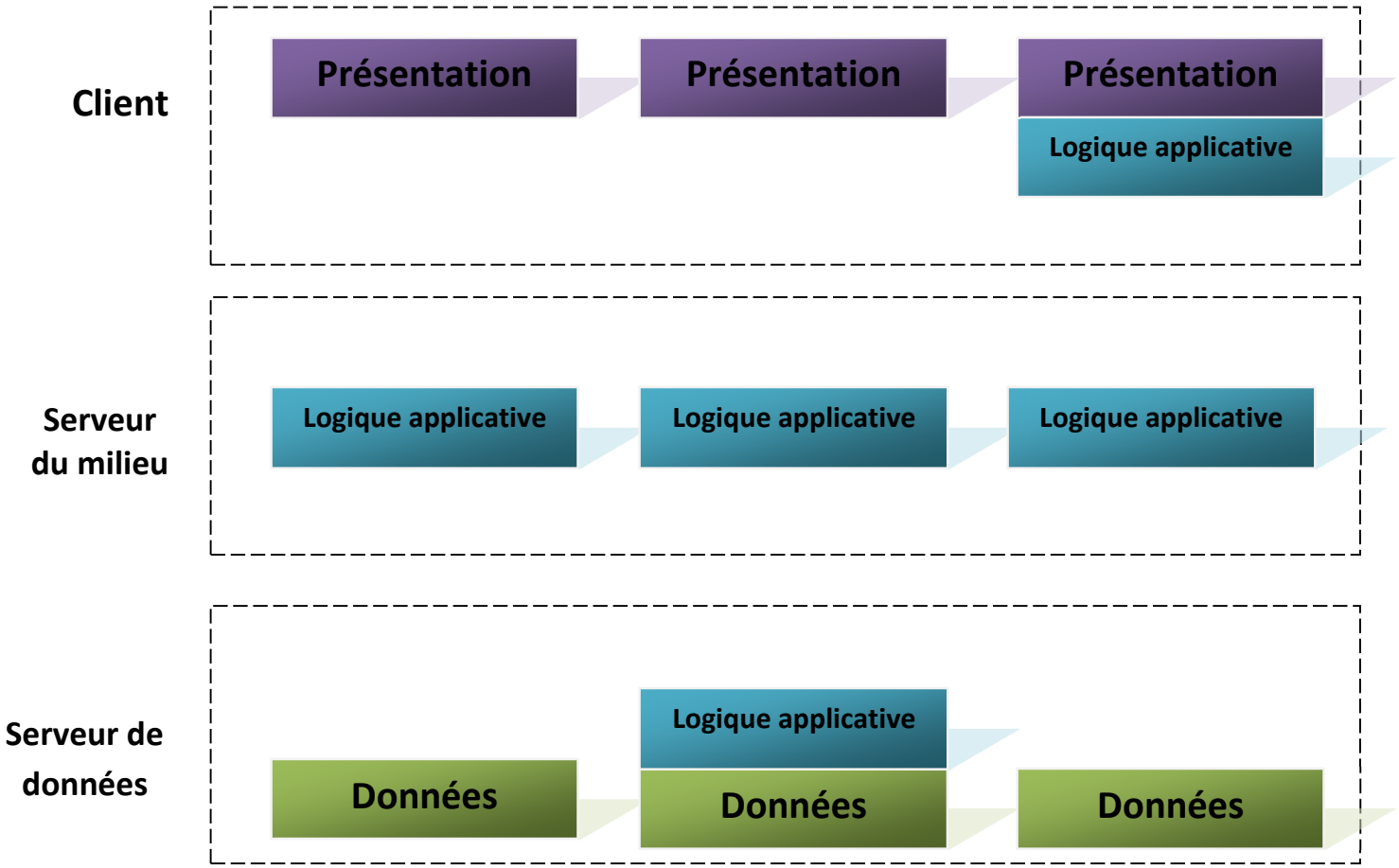


Figure 6 : Modèle de Gartner pour les systèmes a 3 niveaux (3-tiers)

a. Avantages

Étant donné que l'architecture 3-tiers se base sur la dépendance de ces niveaux applicatifs implantés sur des machines différentes, de ce fait:

- ✓ Le poste client ne supporte plus l'ensemble des traitements, il est moins sollicité et peut être moins évolué, donc moins coûteux.
- ✓ Les ressources présentes sur le réseau sont mieux exploitées, puisque les traitements applicatifs peuvent être partagés ou regroupés(le serveur d'application peut s'exécuter sur la même machine que le SGBD).
- ✓ La fiabilité et les performances de certains traitements se trouvent améliorés par leurs centralisations.
- ✓ Il est relativement simple de faire face à une forte montée en charge, en renforçant le service applicatif.

b. Limites

Le serveur d'application constitue la pierre angulaire de l'architecture et se trouve souvent fortement sollicité et il est difficile de répartir la charge entre le client et le serveur.

On se retrouve confronté aux épineux problèmes de dimensionnement serveur et de gestion de la montée en charge rappelant l'époque des mainframes.

De plus, les solutions mises en œuvre sont relativement complexes à maintenir et la gestion des sessions est compliquée.

Les contraintes semblent inversées par rapport à celles rencontrées avec les architectures 2-tiers : le client est soulagé, mais le serveur est fortement sollicité.

Remarque :

On a pu selon différents critères, comparées les deux architectures 2-tiers et 3-tiers, qui se résument dans le tableau suivant:

critères	2 tiers	3 tiers
Administration du système	Complexe (la couche application est physiquement répartie sur plusieurs postes clients)	Moins complexe (les applications peuvent être gérées centralement sur le serveur)
Sécurité	Faible (sécurité au niveau des données)	Élevée (raffinée au niveau des services ou des méthodes)
Encapsulation des données	Faible (les tables de données sont directement accessibles)	Élevée (le client fait appel à des services ou méthodes)
Performance	Faible (plusieurs requêtes SQL sont transmises sur le réseau, les données sélectionnées doivent être acheminées vers le client pour analyse)	Bonne (seulement les appels de services et les réponses sont mis sur le réseau)
Extensibilité	Faible (gestion limitée des liens réseaux avec le client)	Excellente (possibilité de répartir dynamiquement la charge sur plusieurs serveurs)
Réutilisation	Faible (application monolithique sur le client)	Excellente (réutilisation des services et des objets)
Facilité de développement	Élevée	En progression (des outils intégrés pour développer la partie du client et du serveur)
Lien serveur-serveur	Non	Oui (via le middleware serveur/serveur
Intégration des systèmes déjà en place	Non	Oui (via des passerelles encapsulées par les services ou objets)
Soutien internet	Faible (les limitations de la bande passante pénalisent le téléchargement d'application de type (fat-client)	Excellente (les applications de type "thin-client" sont facilement téléchargeable et les appels aux services repartissent la charge sur un ou plusieurs serveurs)
Sources de données hétérogènes	Non	Oui (les applications 3-tiers peuvent utiliser plusieurs bases de données dans la même transaction)
Choix de communications de type "riche"	Non (synchrone et RPC[1])	Oui (gestion asynchrone des messages, files de livraison, publication et abonnement "broadcast"
Flexibilité d'architecture matérielle	limitée	Excellente (possibilité de faire résider les couches 2 et 3 sur une ou plusieurs machines

Tableau 1: Comparaison du C/S 2-tiers et C/S 3-tiers

2.3. Architecture N-tiers

L'architecture n-tiers a été pensée pour pallier aux limitations des architectures 3-tiers et concevoir des applications puissantes et simples à maintenir.

[1] Remote Procedure Call, protocole, automatique appel de procédure distante

L'appellation n-tiers pourrait faire penser que cette architecture met en œuvre un nombre indéterminé de niveaux d'application, alors que ces derniers sont au maximum trois (les 3 niveaux d'une application informatique). En fait, l'architecture n-tiers qualifie la distribution de la couche applicative entre de multiple service et non la multiplication des couches.

Les composants "métiers" de la couche applicative sont spécialisés et indépendants.

Ils sont capables de communiquer entre eux et peuvent donc coopérer en étant implantés sur des machines distinctes.

Ce type d'architecture permet de distribuer librement la logique applicative, ce qui facilite la répartition de la charge entre tous les niveaux.

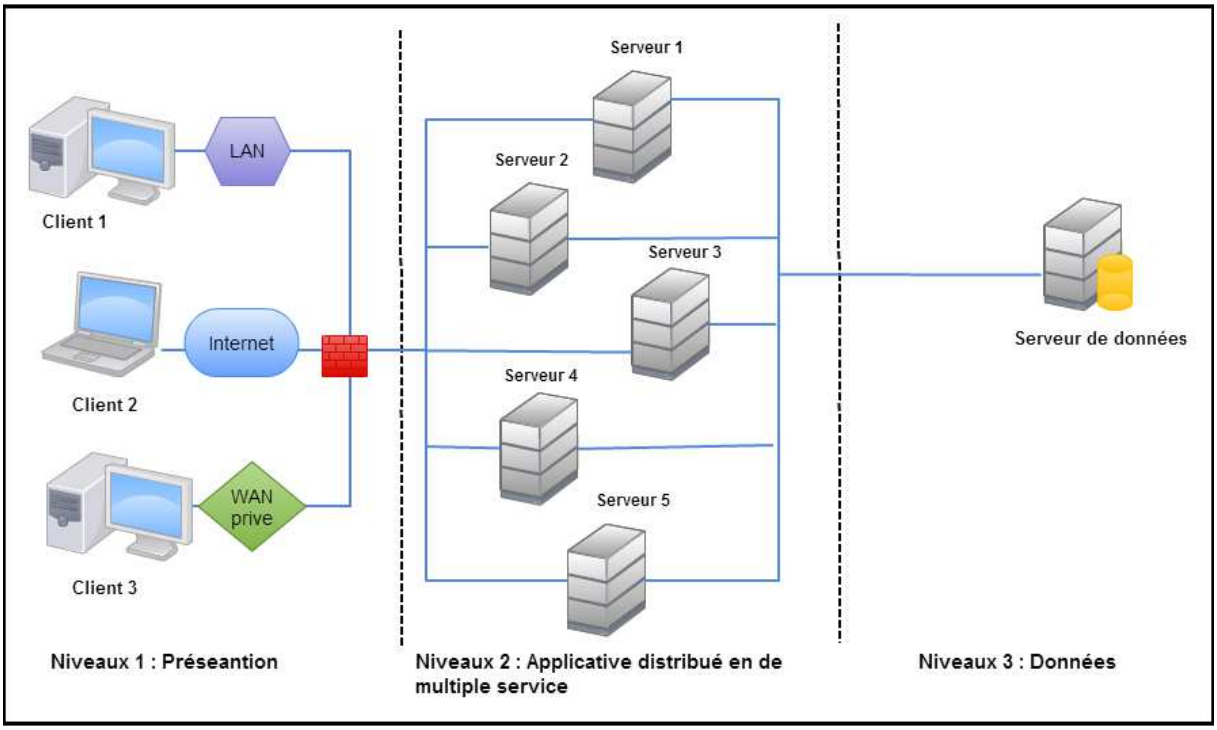


Figure 7 : Architecture client/serveur N-tiers

3. Les middlewares

Basé sur les techniques de communication client-serveur vues ci-dessus, le middleware assure les connexions entre les serveurs de données et les outils de développement sur les postes clients. Cette couche de logiciels très spécifique du client-serveur cache les réseaux et les mécanismes de communication associés. Elle assure une collaboration fructueuse entre clients et serveurs. Middleware n'ayant pas de traduction directe en français, nous introduiront le concept de **médiateur** qui correspond à un middleware particulier.

3.1. Définition

Ensemble des services logiciels construits au-dessus d'un protocole de transport afin de permettre l'échange de requêtes et des réponses associées entre client et serveur de manière transparente.

3.2. Les objectifs principaux d'un middleware :

L'objectif d'un médiateur est donc d'assurer une liaison transparente, c'est-à-dire de cacher l'hétérogénéité des composants mis en jeu. Il s'agit en particulier d'assurer la transparence aux réseaux, aux SGBD, et dans une certaine mesure aux langages d'accès.

- **Transparence aux réseaux :** le middleware assure le dialogue requête-réponse qu'ils soient locaux ou nationaux, voire internationaux et offrant une interface standard de dialogue à l'application.
- **Transparence aux serveurs** : la diversité des SGBD offrant des moyens de connexion et des langages de requêtes souvent différents. Le médiateur doit cacher la diversité et uniformiser ces langages en s'appuyant le plus possible sur les standards.
- **Transparence aux langages :** assuré l'envoi des requêtes et la réception des réponses pour tous les langages de développement utilisés coté client.

3.3. Les services assurés par le middleware :

Parmi les services qu'assure le middleware on retrouve:

- **Communication :** Permet la transmission de messages entre client et serveur sans altération. Ce service doit gérer la connexion au serveur, la préparation de l'exécution des requêtes, la récupération des résultats et la déconnexion de l'utilisateur.
- **Adressage :** Permet d'identifier la machine serveur sur laquelle est localisé le service demandé afin d'en déduire le chemin d'accès.
- **Conversion :** Service utilisé pour la communication entre les machines mettant en œuvre des formats de données différents (hétérogénéités des plates-formes).
- **Sécurité:** Permet de garantir la confidentialité et la sécurité des données grâce à des mécanismes d'authentification et de cryptage des informations.
- **Administration :** Permet la configuration et l'exploitation de l'ensemble des composants (système et application) des différentes plates-formes, généralement à partir d'un point central.

Conclusion

Dans ce chapitre, nous avons défini le client-serveur comme un mode de dialogue entre processus, le processus client demandant au processus serveur l'exécution de service par l'intermédiaire de requêtes transportées par des messages.

Outre l'avantage de pouvoir fédérer une panoplie de vendeurs d'outils, le client-serveur répond aux vrais besoins actuels des entreprise :

- ✓ **Les contraintes de l'entreprise** : les entreprise sont soumises à des contraintes de plus en plus fortes, aussi bien du monde extérieur (sont imposées par les clients de plus en plus exigeants, la régulation de plus en plus complexe,) que de l'intérieur de l'entreprise (se traduisent par des pressions sur les budgets, la difficulté à absorber les technologies nouvelles, et un manque général de temps et de moyens).
- ✓ **Mieux maitriser le système d'information** : l'intégration et l'évolution passent par l'architecture client-serveur, qui permet l'intégration des données mais aussi des traitements communs de l'entreprise au sein de serveur relationnel, et l'évolutivité des applications développées sur les clients de type PC ou station de travail.
- ✓ **Prendre en compte les évolutions des technologies** : le client-serveur apporte une modularité des composants matériels et logiciels. Ceci permet d'intégrer plus facilement les évolutions technologiques.

L'architecture client-serveur elle-même évolue vers l'architecture distribuée, ou un client peut s'adresser à différents serveurs et un serveur peut à son tour devenir client d'un autre serveur. Aujourd'hui, les applications sont conçues pour fonctionner dans toutes sortes d'environnements. Quel que soit le type d'application développée on a besoin du middleware pour connecter les applications entre elles. Pour y arriver, il est nécessaire de rajouter du code dans tous les systèmes qui ont besoin de dialoguer entre eux.

Introduction

Dans le cadre de développement d'une application, il ne convient pas de se lancer tête baissée dans l'écriture du code : il faut d'abord arranger ses idées, documenter, puis organiser la réalisation en définissant les modules et les étapes à suivre. Donc, afin de faire un bon développement, Il faut tout d'abord passer par la conception en suivant une démarche.

Dans notre cas nous avons choisis de modéliser avec le formalisme UML (Unified Modeling Language) qui offre une flexibilité marquante qui s'exprime par l'utilisation des diagrammes.

1. Quelques notions fondamentales

1.1. Notion modèle et modélisation

Un modèle est une représentation abstraite et simplifiée (i.e. qui exclut certains détails), d'une entité (phénomène, processus, système, etc.) du monde réel en vue de le décrire, de l'expliquer ou de le prévoir. Modèle est synonyme de théorie, mais avec une connotation pratique : un modèle, c'est une théorie orientée vers l'action qu'elle doit servir.

Modéliser un système avant sa réalisation permet de mieux comprendre le fonctionnement du système. C'est également un bon moyen de maîtriser sa complexité et d'assurer sa cohérence. Un modèle est un langage commun, précis, qui est connu par tous les membres de l'équipe et il est donc, à ce titre, un vecteur privilégié pour communiquer. Cette communication est essentielle pour aboutir à une compréhension commune aux différentes parties prenantes (notamment entre la maîtrise d'ouvrage et maîtrise d'œuvre informatique) et précise d'un problème donné.

Remarque :

Dans le domaine de l'ingénierie du logiciel, le modèle permet de mieux répartir les tâches et d'automatiser certaines d'entre elles. C'est également un facteur de réduction des coûts et des délais.

1.2.L'approche orienté objet :

L'approche orientée objet considère le logiciel comme une collection d'objets dissociés, et identifiés, définis par des propriétés. Une propriété est soit un attribut (i.e. une donnée caractérisant l'état de l'objet), entité élémentaire comportementale de l'objet). La fonctionnalité du logiciel émerge alors de l'interaction entre les différents objets qui le constituent. L'une des particularités de cette approche est qu'elle rapproche les données et leurs traitements associés au sein d'un unique objet.

2. Histoire de l'UML

Les méthodes utilisées dans les années 1980 pour organiser la programmation impérative (notamment Merise) étaient fondées sur la modélisation séparée des données et des traitements. Lorsque la programmation par objets prend de l'importance au début des années 1990, la nécessité d'une méthode qui lui soit adaptée devient évidente. Plus de cinquante méthodes apparaissent entre 1990 et 1995 (Booch, Classe-Relation, Fusion, HOOD, OMT, OOA, OOD, OOM, OOSE, etc.), mais aucune ne parvient à s'imposer. En 1994, le consensus se fait autour de trois méthodes :

- OMT de James Rumbaugh (*General Electric*) fournit une représentation graphique des aspects statique, dynamique et fonctionnel d'un système ;
- OOD de Grady Booch, définie pour le *Department of Defense*, introduit le concept de paquetage (*package*) ;
- OOSE d'Ivar Jacobson (Ericsson) fonde l'analyse sur la description des besoins des utilisateurs (cas d'utilisation, ou *use cases*).

Chaque méthode avait ses avantages et ses partisans. Le nombre de méthodes en compétition s'était réduit, mais le risque d'un éclatement subsistait : la profession pouvait se diviser entre ces trois méthodes, créant autant de continents intellectuels qui auraient du mal à communiquer.

Événement considérable et presque miraculeux, les trois gourous qui régnaient chacun sur l'une des trois méthodes se mirent d'accord pour définir une méthode commune qui fédérerait leurs apports respectifs (on les surnomme depuis « the Amigos »). UML (*Unified Modeling*

Language) est né de cet effort de convergence. L'adjectif *unified* est là pour marquer qu'UML unifie, et donc remplace.

En fait, et comme son nom l'indique, UML n'a pas l'ambition d'être exactement une méthode : c'est un langage.

L'unification a progressé par étapes. En 1995, Booch et Rumbaugh (et quelques autres) se sont mis d'accord pour construire une méthode unifiée, *Unified Method 0.8 ;* en 1996, Jacobson les a rejoints pour produire UML 0.9 (notez le remplacement du mot*méthode* par le mot *langage*, plus modeste). Les acteurs les plus importants dans le monde du logiciel s'associent alors à l'effort (IBM, Microsoft, Oracle, DEC, HP, Rational, Unisys, etc.) et UML 1.0 est soumis à l'OMG(5). L'OMG adopte en novembre 1997 UML 1.1 comme langage de modélisation des systèmes d'information à objets. La version d'UML en cours en 2008 est UML 2.1.1 et les travaux d'amélioration se poursuivent.

UML est donc non seulement un outil intéressant, mais une norme qui s'impose en technologie à objets et à laquelle se sont rangés tous les grands acteurs du domaine, acteurs qui ont d'ailleurs contribué à son élaboration.

3. Définition

UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue.

UML unifie à la fois les notions 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. UML a une dimension symbolique et ouvre une nouvelle voie d'échange de visions systématiques précises. Ce langage est certes issu de développement logiciel mais pourrait être appliqué à toute science fondée sur la description d'un système. Dans l'immédiat, UML intéresse fortement les spécialistes de l'ingénierie système. *[Pascal Roques, UML 2 modéliser une application web]*

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 spécification jusqu'au codage.

4. L'utilité de l'UML

Il permet grâce à un ensemble de diagrammes explicites, de représenter l'architecture et le fonctionnement des systèmes informatiques complexes en tenant compte des relations entre les concepts utilisés et l'implémentation qui en découle.

UML est avant tout un support de communication performant, qui facilite la représentation et la compréhension de solutions objet :

- ✓ Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation de solutions.
- ✓ L'aspect formel de sa notation limite les ambiguïtés et les incompréhensions.
- ✓ Son indépendance par rapport aux langages de programmation, aux domaines d'application et aux processus, en fait un langage universel.

UML est donc bien plus qu'un simple outil qui permet de "dessiner" des représentations mentales, Il permet de parler un langage commun, normalisé mais accessible, car visuel.

Il représente un juste milieu entre langage mathématique et naturel, pas trop complexe mais suffisamment rigoureux. Une autre caractéristique importante d'UML, est qu'il cadre l'analyse. UML permet de représenter un système selon différentes vues complémentaires : "les diagrammes".

5. Les diagrammes d'UML

UML 2.0 comporte 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 (*Interaction diagram*)
- 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, d'ailleurs dans le cadre de notre projet nous présentèrent que les diagrammes auquel nous feront appel et qui sont :

5.1. Diagramme de cas d'utilisation (*Use case diagram*)

Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique.

Il ne faut pas négliger cette première étape pour produire un logiciel conforme aux attentes des utilisateurs. Pour élaborer les cas d'utilisation, il faut se fonder sur des entretiens avec les utilisateurs.

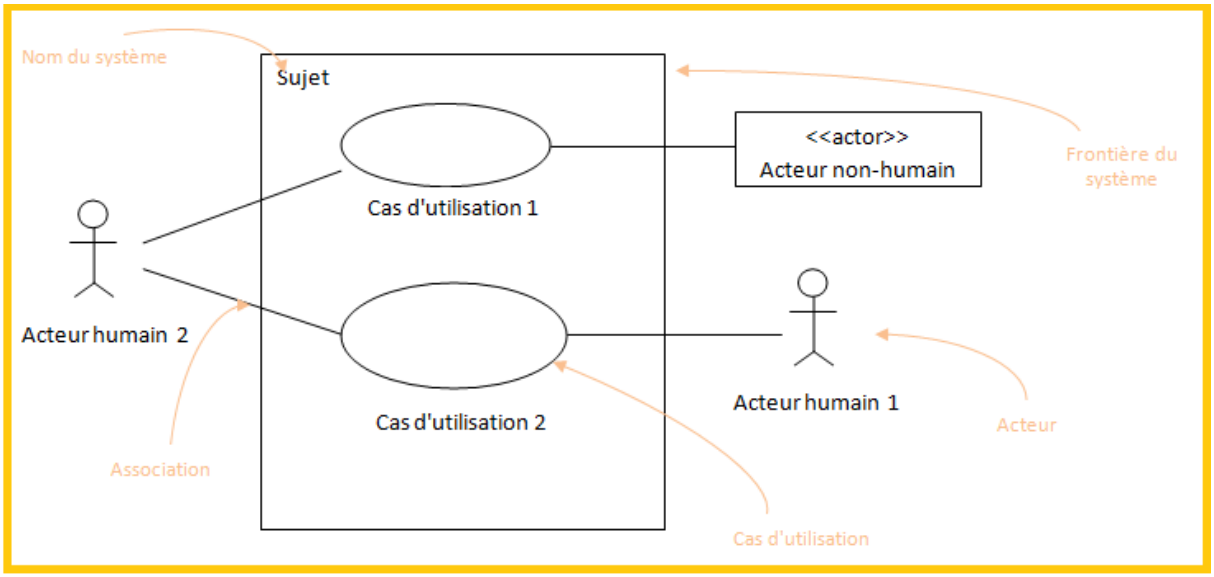


Figure 1 : Diagramme de cas d'utilisation

1. Acteur

1.1. Définition

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données.

1.2. Comment les identifier ?

Les acteurs candidats sont systématiquement :

- Les utilisateurs humains directs : faire donc en sorte d'identifier tous les profils possibles, sans oublier l'administrateur, l'opérateur de maintenance, etc. ;
- Les autres systèmes connexes qui interagissent aussi directement avec le système étudié, souvent par le biais de protocoles bidirectionnels.

1.3. Comment les représenter ?

La représentation graphique standard de l'acteur en UML est l'icône appelée stick man, avec le nom de l'acteur sous le dessin. On peut également figurer un acteur sous la forme rectangulaire d'une classe, avec le mot-clé <<actor>>. Une troisième représentation (Intermédiaire entre les deux premières) est également possible avec certains outils, comme cela est indiqué ci-après.

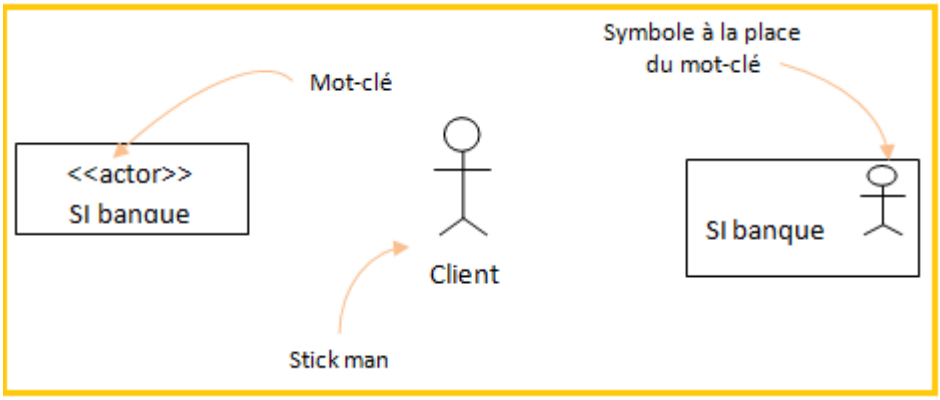


Figure 2 : Représentation graphique possible d'un acteur

Remarque

Faire prévaloir l'utilisation de la forme graphique du stick man pour les acteurs humains et une représentation rectangulaire pour les systèmes connectés.

1.4. Type d'acteur (Acteur principal ou secondaire)

Contrairement à ce que l'on pourrait croire, tous les acteurs n'utilisent pas forcément le système

Nous appelons **acteur principal** celui pour qui le cas d'utilisation produit un résultat observable.

Nous qualifions d'**acteurs secondaires** les autres participants du cas d'utilisation. Les acteurs secondaires sont souvent sollicités pour des informations complémentaires ; ils peuvent uniquement consulter ou informer le système lors de l'exécution du cas d'utilisation.

2. Cas d'utilisation

2.1. Définition

Un cas d'utilisation (« use case ») représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Chaque cas d'utilisation spécifie un comportement attendu du système considéré comme un tout, sans imposer le mode de réalisation de ce comportement. Il permet de décrire ce que le futur système devra faire, sans spécifier comment il le fera.

2.2. Identification un cas d'utilisation

L'objectif est le suivant : l'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système. Chaque cas d'utilisation correspond donc à une fonction métier du système, selon le point de vue d'un de ses acteurs.

Pour chaque acteur, il convient de :

- Rechercher les différentes intentions métier avec lesquelles il utilise le système,
- Déterminer dans le cahier des charges les services fonctionnels attendus du système.

Remarque :

Nommez les cas d'utilisation par un verbe à l'infinitif suivi d'un complément, du point de vue de l'acteur (et non pas du point de vue du système).

2.3. Représentation d'un cas d'utilisation

Les cas d'utilisation sont se représenter par une ellipse (un ovale) contenant le nom du cas reliés par des associations (lignes) à leurs acteurs (icône du « stick man », ou représentation graphique équivalente). Chaque association signifie simplement « participe à ». Un cas d'utilisation doit être relié à au moins un acteur.

Figure 3: Exemple de représentation d'un cas d'utilisation

2.4. Analyse d'un cas d'utilisation

Pour détailler la dynamique du cas d'utilisation, la procédure la plus évidente consiste à recenser de façon textuelle toutes les interactions entre les acteurs et le système.

Le cas d'utilisation doit avoir un début et une fin clairement identifiés. Il faut également préciser les variantes possibles, telles que le cas nominal, les différents cas alternatifs et d'erreur, tout en essayant d'ordonner séquentiellement les descriptions, afin d'améliorer leur lisibilité**.**

Donc une fois les cas d'utilisation identifiés, il faut encore les décrire, La fiche de description textuelle d'un cas d'utilisation n'est pas normalisée par UML. Tout fois la structuration suivante est recommandée :

Sommaire d'identification (obligatoire)	Inclut titre, résumé, dates de création et de modification, version, responsable, acteurs...
Description des scénarios (obligatoire)	Décrit le scénario nominal, les scénarios (ou enchaînements) alternatifs, les scénarios (ou enchaînements) d'erreur, mais aussi les prés conditions et les post conditions.
Exigences non fonctionnelles (optionnel)	Ajoute, si c'est pertinent, les informations suivantes : fréquence, volumétrie, disponibilité, fiabilité, intégrité, confidentialité, performances, concurrence, etc. Précise également les contraintes d'interface homme-machine comme des règles d'ergonomie, une charte graphique, etc.

2.5. Relation entre cas d'utilisation

Avec UML, il est en effet possible de détailler et d'organiser les cas d'utilisation en ajoutant des relations d'inclusion, d'extension et de généralisation entre cas d'utilisation ;

a. **Une relation d'inclusion :** formalisée par le mot clé « include » : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire.

b. **Une relation d'extension :** formalisée par le mot-clé « extend » : le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle.

c. **Une relation de généralisation/spécialisation :** les cas d'utilisation descendants héritent de la description de leur parent commun .Chaque d'entre eux peut néanmoins comprendre des interactions spécifiques supplémentaires.

5.2. Diagramme de séquence (*Sequence diagram*)

1. Définition

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language.

2. Type de diagramme

2.1. Diagramme de séquence système

Les cas d'utilisations décrivent les interactions des acteurs avec l'application. Lors de ces interactions, les acteurs produisent des messages qui affectent le système informatique et appellent généralement une réponse de celui-ci.

Ces messages sont isolés et c'est cela qui seront représentés graphiquement sur des diagrammes de séquence UML.

Les DSS (diagrammes de séquence système) montrent non seulement les acteurs externes qui interagissent directement avec le système, mais également ce système (en tant que boîte noire) et les événements système déclenchés par les acteurs. L'ordre chronologique se déroule vers le bas et l'ordre des messages doit suivre la séquence décrite dans le cas d'utilisation.

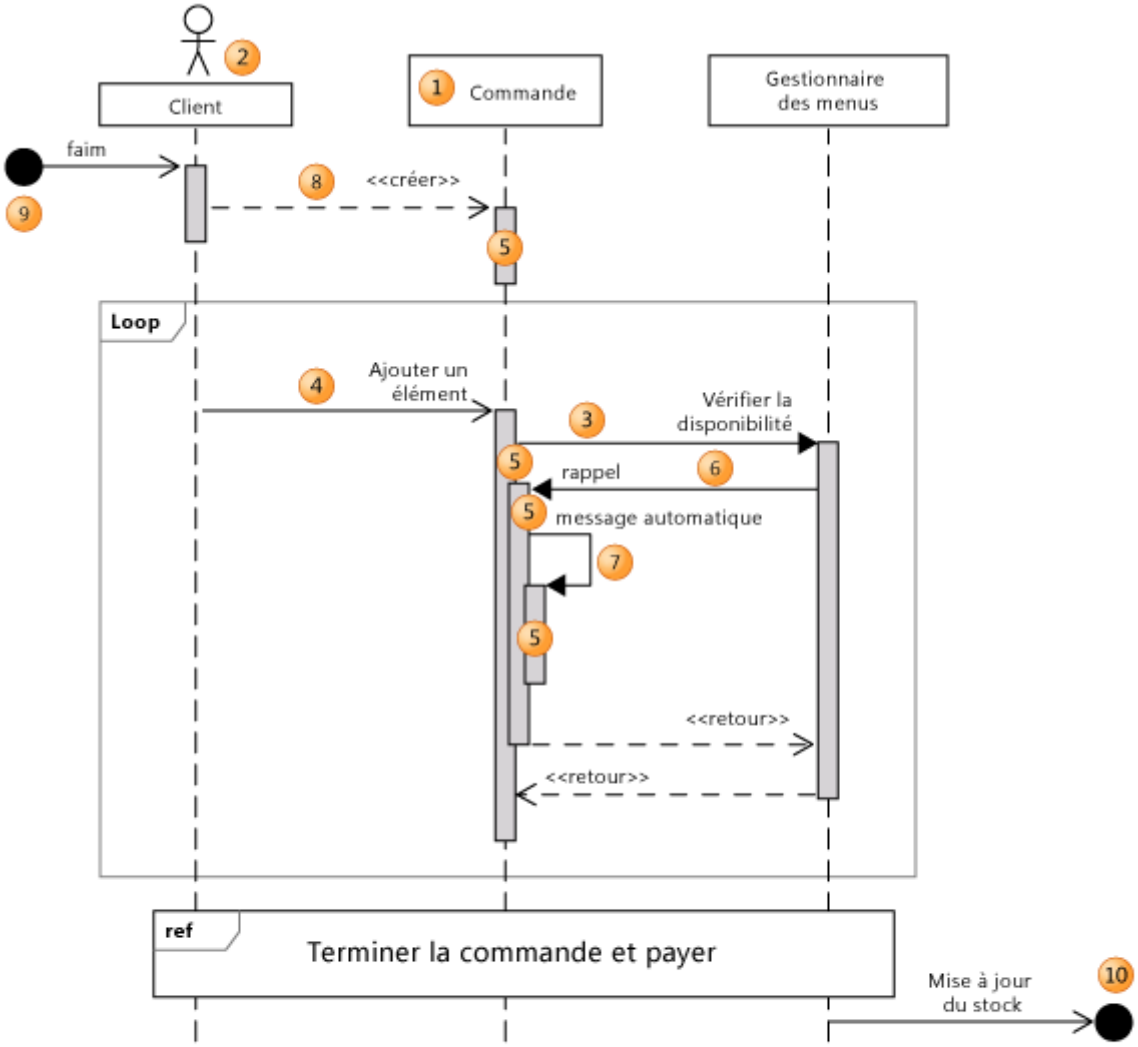
2.2. Diagramme de séquence enrichi (détaillé)

Par rapport aux diagrammes de séquence système vue précédemment, le système vu comme boite noire va être remplacé par un ensemble d'objets en interaction, pour cela nous allons voir trois types de classes d'analyse, à savoir les dialogues, les contrôles et les entités :

- Les entités: possèdent seulement des attributs. De plus les entités ne peuvent être reliées qu'aux contrôles.
- Les contrôles : vont seulement posséder des opérations, montrent le comportement du système informatique, ont accès à tous types de classe.
- Les dialogues vont posséder des attributs et des opérations, les attributs représentent des champs de saisie ou de résultats, les opérations représentent des actions de l'utilisateur sur IHM, ils ne peuvent être reliés qu'aux contrôles ou à d'autres dialogues.

3. Lecture du diagramme de séquence

Le diagramme accompagné d'un tableau décrit les éléments que vous pouvez voir sur un diagramme de séquence:



Forme	Élément	Decription
1	**Ligne de vie**	-Une ligne verticale qui représente la séquence des événements qui se produisent dans un participant pendant une interaction, alors que le temps progresse en bas de ligne. Ce participant peut être une instance d'une classe, un composant ou un acteur.
2	**Acteur**	-Un participant qui est externe au système que vous développez. **Remarque :** Vous pouvez faire apparaître un symbole d'acteur en haut d'une ligne de vie en définissant sa propriété **Actor**.
3	**Message synchrone**	-L'émetteur attend une réponse à un message synchrone avant de continuer. Le diagramme montre à la fois l'appel et le retour. Les messages synchrones sont utilisés pour représenter des appels de fonction ordinaires dans un programme, ainsi que d'autres types de messages qui se comportent de la même façon.
4	**Message asynchrone**	-Un message qui ne requiert pas de réponse avant que l'expéditeur continue. Un message asynchrone affiche uniquement un appel venant de l'expéditeur. Utilisez cette option pour représenter la communication entre des threads distincts ou la création d'un nouveau thread.
5	**Occurence**	-Un rectangle grisé vertical qui s'affiche sur la ligne de vie d'un participant et représente la période

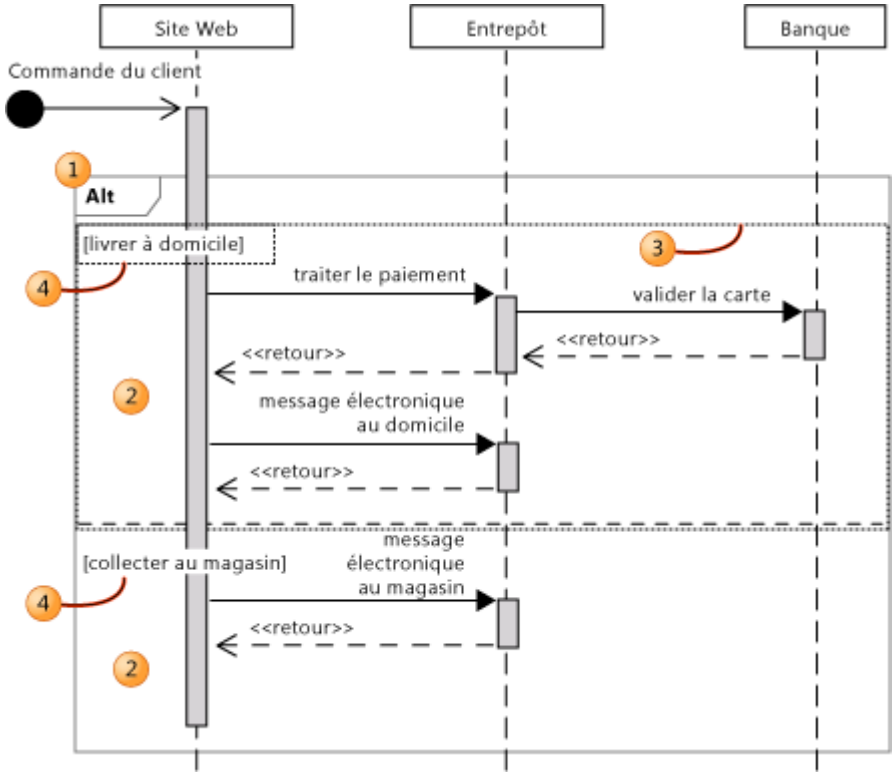
	D'éxécution	lorsque le participant exécute une opération. **Remarque:** L'exécution commence là où le participant reçoit un message. Si le message initialisant était un message synchrone, l'exécution se termine avec une flèche arrière renvoyée à l'expéditeur.
6	**Message de rappel**	-Un message retourné à un participant qui attend le retour d'un appel antérieur. L'occurrence d'exécution résultante s'affiche sur l'occurrence d'exécution existante.
7	**Auto-message**	-Un message d'un participant à lui-même. L'occurrence d'exécution résultante s'affiche sur l'exécution d'envoi.
8	**Crée un message**	-Un message qui crée un participant. Si un participant reçoit un message de création, ce doit être le premier qu'il reçoit.
9	**Message trouvé**	-Message asynchrone venant d'un participant inconnu ou non spécifié.
10	**Message perdu**	-Message asynchrone à un participant inconnu ou non spécifié.
X	**Événement de destruction**	-Représente le point vers lequel l'objet est supprimé ou plus accessible. Apparaît en bas de chaque ligne de vie.

Tableau 1 : Explication du diagramme de séquence.

4. Fragment d'interaction combiné

4.1. Définition

Un fragment combiné se compose d'un ou de plusieurs opérandes d'interaction et chacun d'entre eux contient un ou plusieurs messages, ou encore d'autre fragments combinés. Nous pouvant distinguer tous ces éléments dans l'illustration suivante suivie d'une brève explication



Les éléments présentés dans l'illustration sont les suivants :

1. Fragment combiné. Il existe plusieurs genres de fragments combinés. Cet exemple est un fragment combiné Alt que vous pouvez utiliser pour montrer que les autres séquences de messages peuvent se produire.
2. Opérandes d'interaction. Chaque fragment combiné contient au moins un opérande d'interaction qui peut contenir des messages, des utilisations d'interaction et de plus petits fragments combinés. Dans cet exemple, le fragment combiné Alt dispose de deux opérations d'interaction, affichant ainsi deux autres séquences de messages.
3. Nous pouvant sélectionner chaque opérande d'interaction individuellement en cliquant dessus. Dans cet exemple, l'opérande d'interaction supérieur est sélectionné afin que sa limite puisse être visualisée. En général, seule est visible la ligne de séparation entre les opérandes d'interaction.
4. Protections. Nous pouvant attribuer une protection à chaque opérande d'interaction. Il décrit la condition sous laquelle seront exécutés les messages de l'opérande d'interaction.

4.2. Genres de fragments combinés

Les types de fragments combinés suivants peuvent être utilisés pour décrire des variations qui peuvent se produire en différentes occasions:

Type de fragment	Description
Alternative ("alt")	-Il désigne un choix, une alternative. Il représente deux comportements possibles : c'est en quelque sorte l'équivalent du **SI...ALORS...SINON** : donc, une seule des deux opérandes d'interaction sera réalisée dans un scénario donné. La condition d'exécution d'une des deux opérandes d'interaction peut être explicite ou implicite. -L'utilisation de **else** permet d'indiquer que l'opérande d'interaction est exécutée si la condition du alt est fausse
Option ("opt")	-Il désigne un fragment combiné optionnel comme son nom l'indique : c'est à dire qu'il représente un comportement qui peut se produire... ou pas. Un fragment optionnel est équivalent à un fragment "alt" qui ne posséderait pas d'opérande else (qui n'aurait qu'un seul opérande d'interaction). Un fragment optionnel est donc une sorte de **SI...ALORS**.
Break ("break")	-Il est utilisé pour représentent des scenarios d'**exception** en quelque sorte. Les intéractions de ce fragment seront exécutées à la place des interactions décrites en dessous. Il y a donc une notion d'interruption du flot "normal" des interactions.
Parallel ("par")	Il utilisé pour représenter des interactions ayant lieu en **parallèle**. Les interactions des différents opérandes peuvent donc se mélanger, s'intercaler, dans la mesure où l'ordre imposé dans chaque opérande est respecté.
Loop ("loop")	Il est utilisé pour décrire un ensemble d'interaction qui s'exécutent **en boucle** donc Le fragment est répété un certain nombre de fois. Dans la protection, nous pouvant indiquer la condition sous laquelle il doit être répété. Les fragments combinés " Loop "ont les propriétés **Min** et **Max** qui indiquent les nombres minimaux et maximaux de fois que le fragment peut être répété. L'absence de restriction correspond à la valeur par défaut.

Tableau 2 : Type de fragments combinés.

5.3. Diagramme de classe

1. Définition

Le diagramme de classes est le point central dans un développement orienté objet. En analyse, il a pour objectif de décrire la structure des entités manipulées par les utilisateurs. En conception, le diagramme de classes représente la structure d'un code orienté objet ou,à un niveau de détail plus important, les modules du langage de développement.

Représentation d'une classe

Le diagramme de classes met en œuvre des classes, contenant des attributs et des opérations, et reliées par des associations ou des généralisations

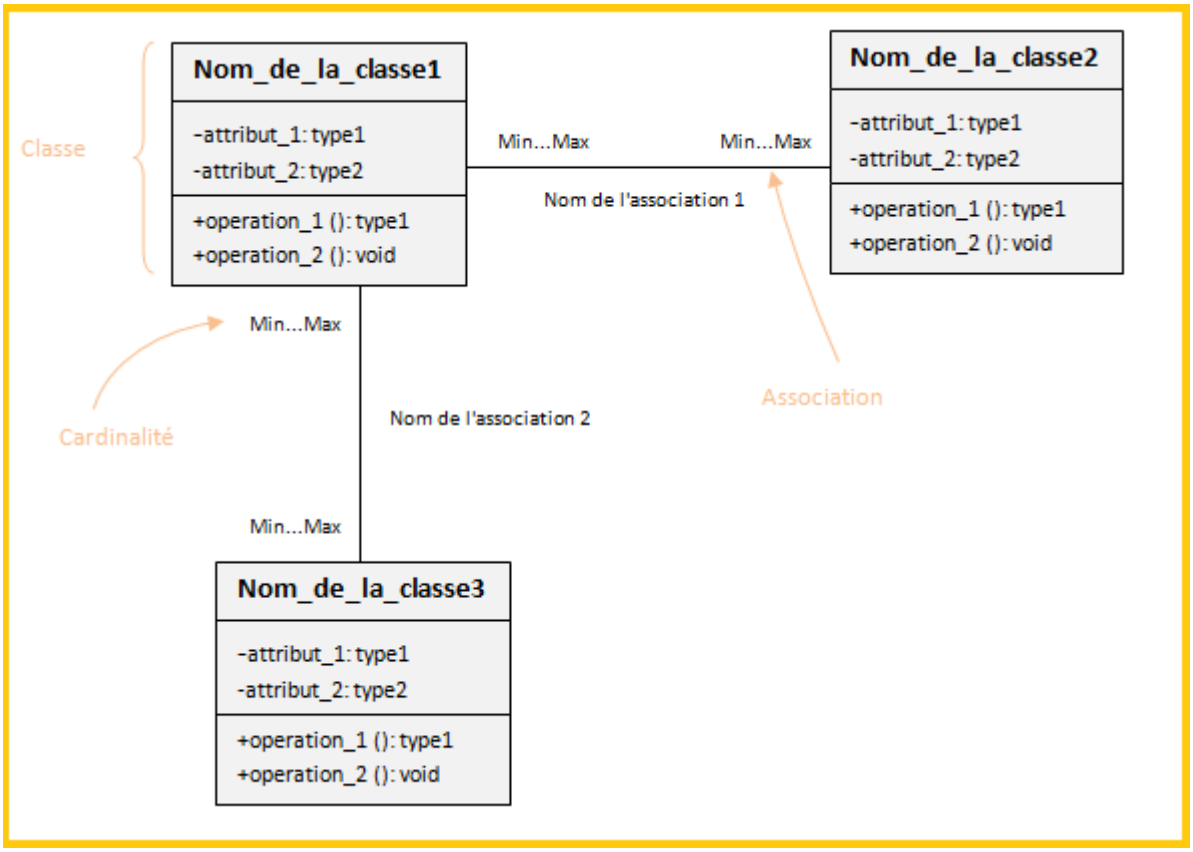


Figure 4: Diagramme de classe de conception

2. Principe et Définition de base

2.1. Classe et Objet

- **Une classe:** représente la description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques. On peut parler également de type.

- **Un objet** : est une entité aux frontières bien définies, possédant une identité et encapsulant un état et un comportement. Un objet est une instance (ou occurrence) d'une classe. Exemple : Pascal Roques est un objet instance de la classe Personne.

2.2. Attribut et Opération

- **Un attribut :** représente un type d'information contenu dans une classe.
- **Une opération:** représente un élément de comportement (un service) contenu dans une classe.

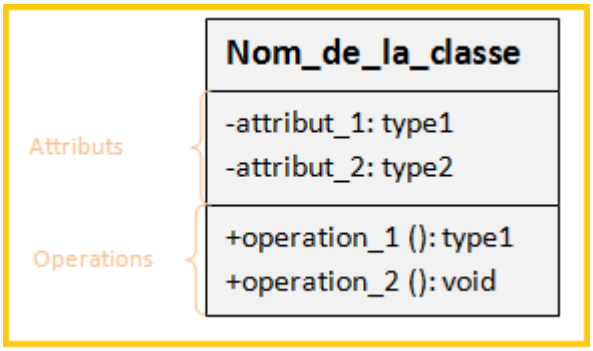


Figure 5 : Représentation des attributs et opération

2.3. Association, Agrégation et Composition

- **Une association** représente une relation sémantique durable entre deux classes.

Remarque: même si le verbe qui nomme une association semble privilégier un sens de lecture, une association entre concepts dans un modèle du domaine est par défaut bidirectionnelle. Donc implicitement.

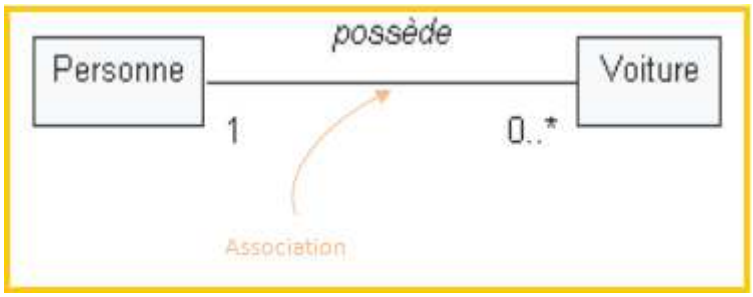


Figure 6: Exemple de multiplicité d'association

- **Une agrégation:** est un cas particulier d'association non symétrique exprimant une relation de contenance. Les agrégations n'ont pas besoin d'être nommées : implicitement elles signifient « contient », « est composé de ».

- **Une composition** : est une agrégation plus forte impliquant que :
  - un élément ne peut appartenir qu'à un seul agrégat composite (agrégation non partagée) ;
  - la destruction de l'agrégat composite entraîne la destruction de tous ses éléments (le composite est responsable du cycle de vie des parties).

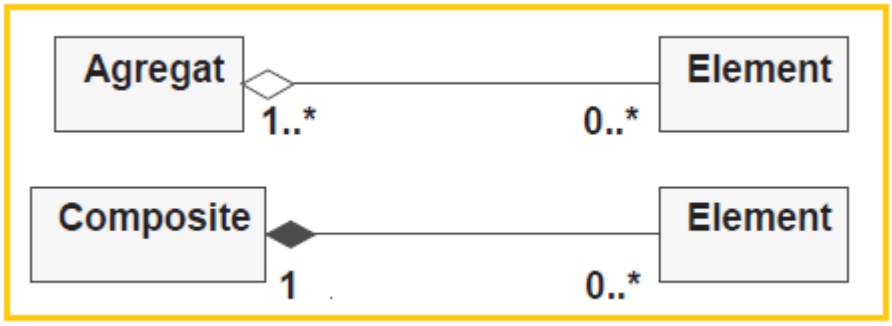


Figure 7: Exemple de d'agrégation et de composition

2.4. Généralisation, Super-Classe, Sous-Classe

- **Une super-classe:** est une classe plus générale reliée à une ou plusieurs autres classes plus spécialisées (sous-classes) par une relation de généralisation.
- **Une sous-classe:** « héritent » des propriétés de leur super-classe et peuvent comporter des propriétés spécifiques supplémentaires.

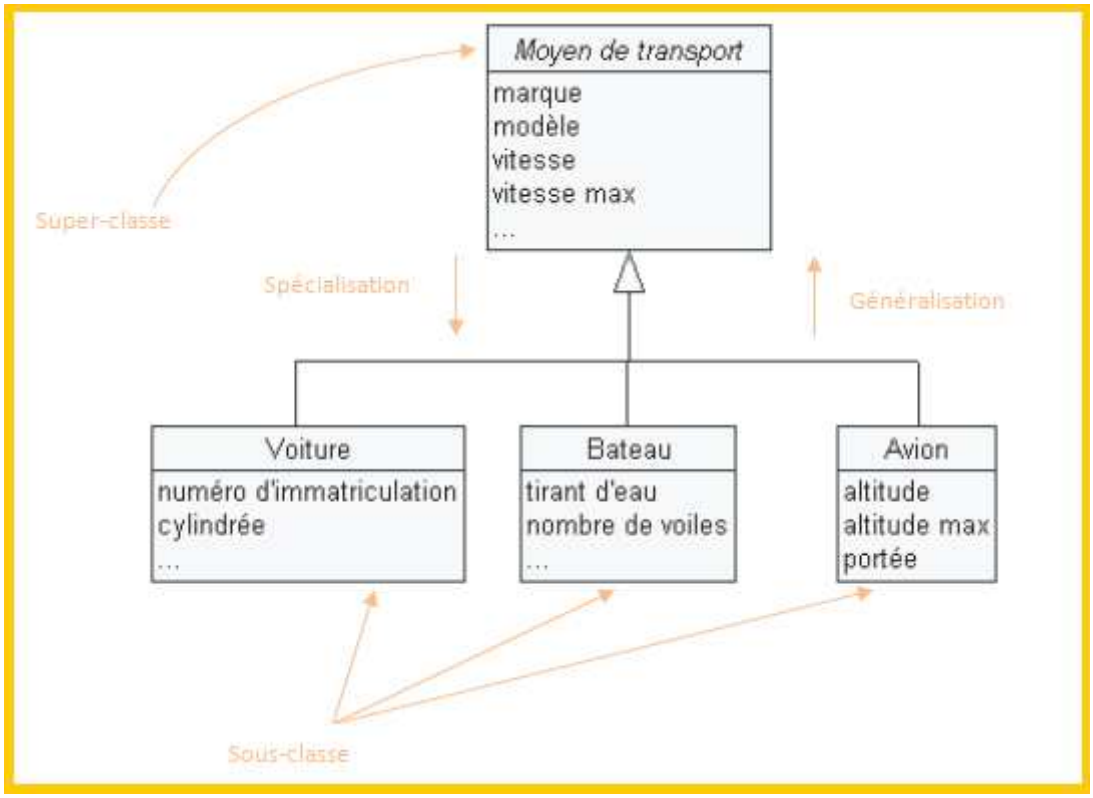


Figure 8: Exemple de super-classe et sous-classe

Remarque (conventions de nommage UML)

- Les noms des attributs commencent toujours par une minuscule (contrairement aux noms des classes qui commencent systématiquement par une majuscule) et peuvent contenir ensuite plusieurs mots concaténés commençant par une majuscule.
- Il est préférable de ne pas utiliser d'accents ni de caractères spéciaux.
- Les mêmes conventions s'appliquent au nommage des rôles des associations, ainsi qu'aux opérations.

6. Comment présenter un modèle UML

La présentation d'un modèle UML se compose de plusieurs documents écrits en langage courant et d'un document formalisé : elle ne doit pas se limiter au seul document formalisé, car celui-ci est pratiquement incompréhensible si on le présente seul. Un expert en UML sera capable dans certains cas de reconstituer les intentions initiales en lisant le modèle, mais pas toujours ; et les experts en UML sont rares. Voici la liste des documents qui paraissent nécessaires :

6.1. Document en langage courant

- Présentation stratégique

Elle décrit pourquoi l'entreprise a voulu se doter de l'outil considéré, les buts qu'elle cherche à atteindre, le calendrier de réalisation prévu, etc.

- Présentation des processus de travail par lesquels la stratégie entend se réaliser

Dans le but de permettre au lecteur de voir comment l'application va fonctionner en pratique, elle doit être illustrée par une esquisse des écrans qui seront affichés devant les utilisateurs de terrain.

- Explication des choix qui ont guidé la modélisation formelle

Il s'agit de synthétiser, sous les yeux du lecteur, les discussions qui ont présidé à ces choix.

6.2. Document formalisé (Modèle formel)

C'est le document le plus épais et le plus difficile à lire. Il est préférable de le présenter sur l'Intranet de l'entreprise. En effet, les diagrammes peuvent être alors équipés de liens hypertextes permettant l'ouverture de diagrammes plus détaillés ou de commentaires.

On doit présenter en premier le diagramme de cas d'utilisation qui montre l'enchaînement des cas d'utilisation au sein du processus, enchaînement immédiatement compréhensible ; puis le diagramme d'activités, qui montre le contenu de chaque cas d'utilisation ; puis le diagramme de séquence, qui montre l'enchaînement chronologique des opérations à l'intérieur de chaque cas d'utilisation. Enfin, le diagramme de classes, qui est le plus précis conceptuellement, mais aussi le plus difficile à lire, car il présente chacune des classes et leurs relations (agrégation, héritage, association, etc.).

Conclusion

A l'issue de cette étape nous avons pu exprimer et détaillé clairement les bases fondamentales du langage UML, nécessaire à tout informaticien désirant développer une application orienté objet.

Tout au long de ce chapitre nous avons pu apprendre à comment identifier les acteurs et leurs besoins afin de construire le diagramme de cas d'utilisation, puis nous l'avons détaillé en précisant comment les objets et les acteurs doivent collaborer ensemble selon, une dimension temporelle par l'utilisation des diagrammes de séquence. A la fin, nous avons décrit l'aspect statique avec les diagrammes des classes.

UML reste un langage et tout langage ne s'apprend vraiment qu'avec la pratique.

Introduction

De nombreuses possibilités existent pour réaliser des applications Internet depuis plusieurs années. Des langages ont été créés, des architectures et des environnements de travail ont été conçus pour répondre aux besoins et faciliter la tâche des développeurs. Sun (le concepteur de Java) a donc mis en place un ensemble de technologies pour réaliser des applications Web. Ces technologies sont regroupées sous le nom J2EE (Java 2 Entreprise Edition), désormais Java EE.

1. Les plates-formes

Une plate-forme est en informatique, une base de travail à partir de laquelle on peut écrire, lire, développer et utiliser un ensemble de logiciels, applications, sites internet ou autres projets de programmation.

Donc, une plateforme est une base générique qui fournit un ensemble de fonctionnalités utiles pour une majorité d'applications. Il peut exister plusieurs types de plates-formes, de la plus générique à la plus spécifique (optimisée pour un type de métier précis par exemple).

L'avantage principal d'utiliser une plate-forme est que l'équipe de développement n'a pas à s'acquitter de développer certaines tâches (connexion à la base de données par exemple, gestion d'objets …). Ce sont des tâches que l'on retrouve très souvent dans un grand nombre de projets et qui n'ont pas d'intérêts à être recoder à chaque fois (perte de temps et d'argent). De plus, il vaut mieux travailler sur une plate-forme qui présente une forte stabilité (ça évite de debugger inutilement !).

2. Définition de java EE

J2EE (Java 2 Enterprise Edition) est une norme proposée par la société Sun, visant à définir un standard de développement d'applications d'entreprises multi-niveaux, basées sur des composants.

On parle généralement de «plate-forme J2EE» pour désigner l'ensemble constitué des services (API) offerts et de l'infrastructure d'exécution. J2EE comprend notamment :

- Les spécifications du serveur d'application, c'est-à-dire de l'environnement d'exécution : J2EE définit finement les rôles et les interfaces pour les applications ainsi que

l'environnement dans lequel elles seront exécutées. Ces recommandations permettent ainsi à des entreprises tierces de développer des serveurs d'application conformes aux spécifications ainsi définies, sans avoir à redévelopper les principaux services.

- Des services, au travers d'API, c'est-à-dire des extensions Java indépendantes permettant d'offrir en standard un certain nombre de fonctionnalités. Sun fournit une implémentation minimale de ces API appelée J2EE SDK (J2EE Software Development Kit).

Remarque:

Depuis la version 5 de J2EE, le chiffre 2 a disparu pour faciliter la compréhension de la version et ne pas mélanger le chiffre 2 avec le numéro de version.

3. Les points forts de Java EE

Java EE bénéficie de nombreux avantages, en particulier une bonne portabilité et une maintenabilité du code.

De plus, l'architecture Java EE repose sur des composants distincts, interchangeables et distribuées, ce qui signifie notamment :

- Qu'il est simple d'étendre l'architecture.
- Qu'un système reposant sur Java EE peut posséder des mécanismes de haute disponibilité, afin de garantir une bonne qualité de service.
- Que la maintenabilité des applications est facilitée.

Java EE est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées. Elle est composée de deux parties essentielles :

- Un ensemble de spécifications pour une infrastructure dans laquelle s'exécutent les composants écrits en Java : un tel environnement se nomme **serveur d'application**.
- Un ensemble d'APIs qui peuvent être obtenues et utilisées séparément. Pour être utilisées, certaines nécessitent une implémentation de la part d'un fournisseur tiers.

L'utilisation de Java EE pour développer et exécuter une application propose plusieurs avantages:

- une architecture d'application basée sur les composants qui permet un découpage de l'application et donc une séparation des rôles lors du développement.
- la possibilité de s'interfacer avec le système d'information existant grâce à de nombreuses API : JDBC, JNDI, JMS, JCA ...
- la possibilité de choisir les outils de développement et le ou les serveurs d'applications

utilisés qu'ils soient commerciaux ou libres.

Java EE permet une grande flexibilité dans le choix de l'architecture de l'application en combinant les différents composants. Ce choix dépend des besoins auxquels doit répondre l'application mais aussi des compétences dans les différentes API de Java EE. L'architecture d'une application se découpe idéalement en au moins trois tiers.

4. L'architecture Java EE en couche

Afin de mieux comprendre le fonctionnement des applications JEE, quelques notions d'architecture logicielle sont nécessaires.

L'architecture en couches consiste à diviser une application en différents modules, qui constituent autant de couches. L'objectif est de proposer une meilleure répartition des rôles (chaque module a un rôle clairement défini), la séparation des traitements, ainsi qu'une réduction des dépendances entre les services. Chaque module se doit d'être indépendant des autres pour permettre une meilleure maintenabilité.

Une application peut aisément se diviser en trois niveaux distincts : les données, le traitement de ces données, et leur affichage.

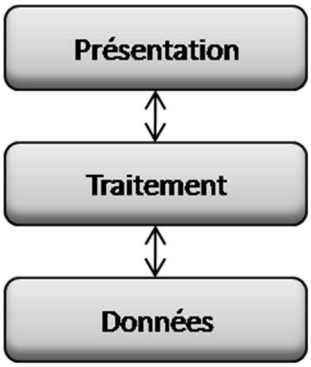


Figure 1 : Schéma des trois niveaux d'une application JEE

- **La couche de données :** regroupe le stockage et les mécanismes d'accès des données de façon à ce qu'elles soient utilisables par l'application au niveau traitement.
- **La couche de traitement :** concerne à la fois les tâches à réaliser par l'application sur les données et les traitements nécessaires suite à une action venant de l'utilisateur : vérification d'authentification, traitements divers, etc.
- **La couche présentation :** gère l'affichage des données et les interactions de l'application avec l'utilisateur. La séparation de cette couche permet notamment de proposer plusieurs

présentations pour une même application : la même couche de traitement peut alors servir pour une application lourde et pour une application légère.

4.1. Utilisation des couches : les points forts

Les principaux avantages que procure l'élaboration d'une architecture « multi-couches » sont :

- Structure de l'application adaptée à l'architecture de déploiement ciblée,
- Modularité de l'application,
- Evolution facilitée de l'application,
- Factorisation du code avec possibilité d'exploiter un framework ou des composants génériques (gain de temps et de performances).

5. L'architecture Java EE en détail

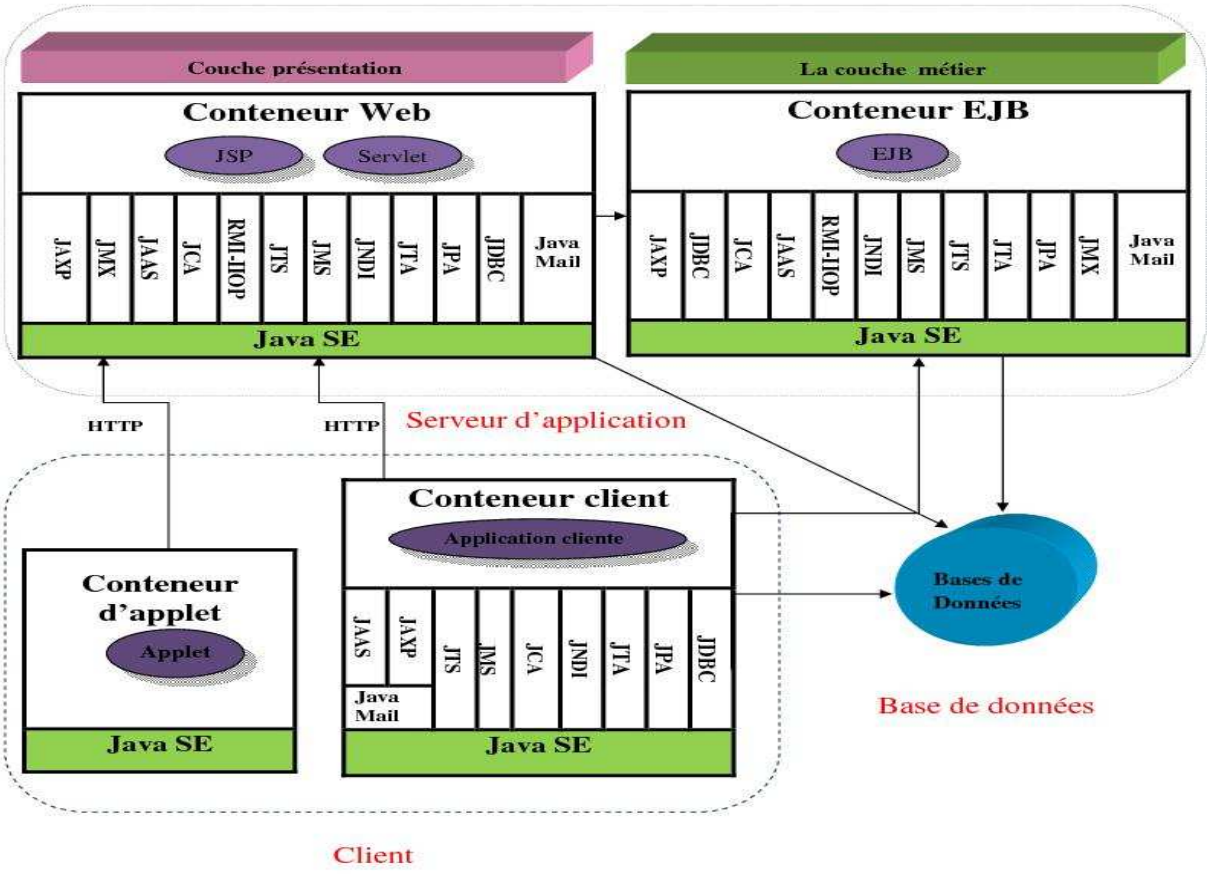


Figure 2. Schéma détaillé de l'architecture Java EE

Les différents rectangles définissent les conteneurs qui fournissent les services pour les différents composants (représenter par les rectangles dans les rectangles) l'ensemble des composants sont expliqué comme suit :

5.1. Les composants client

La plate-forme Java EE propose deux types de clients : les Applets et les applications clientes riches type Java SE.

5.1.1. Applets

Une applet est une application Java côté client, communique avec le serveur par http. Cette application utilise une interface graphique évaluée de type SWING est exécutée dans une machine virtuelle Java installée dans le navigateur. L'applet apparaît dans le document HTML au sein des balises <Applet>. Elle renvoie à un objet qui hérite de la classe Applet.

5.1.2. Application cliente

Une application de type client est un logiciel riche, qui s'exécute sur la machine de client et fournit un ensemble de services aux utilisateurs par l'intermédiaires d'une interface graphique évaluée encore appelée Graphical User Interface (GUI).

Le conteneur client est un jeu de librairies et d'API qui supporte le code client, et met à sa disposition un moyen d'accéder aux composants métiers de l'application.

Les applications clientes ont des interfaces utilisateurs qui peuvent directement interagir avec le tier EJB en utilisant RMI-IIOP.

Les clients ont un accès complet aux services de la plate-forme Java EE. Comme les services de nommage JNDI, l'envoi de messages et JDBC.

Le conteneur client gére l'accès à ces services et les communications RMI-IIOP.

5.2. La couche présentation

La couche ou tiers Web permet de réaliser la logique applicative des projets web. Ainsi, la couche Web concerne les parties M, V et C (Modèle, Vue, Contrôleur), donc elle assure la logique de navigation mais aussi la gestion des droits de l'utilisateur (authentification, droits).

Avec la technologie Java EE plusieurs langages sont proposés pour ce tiers avec les Servlets, JavaServer Page et JavaServer Faces.

Le client invoque le serveur avec une requête http et déclenche une servlet. La servlet réalise une tache de type contrôleur et envoie la réponse à une page JSP chargée de l'affichage (Vue) avec son moteur de balises nommée JS Tag Library.

Les servlets restent la solution la plus performance pour la création de services orientés application ne nécessite pas de réponse au contenu textuel évolué. Cependant, les traitements, l'accès aux données et la présentation sont mélangés. Les pages JSP sont plus appropriées pour générer les contenus textuels comme XHTML et les présentations au design évolué.

5.2.1. Les servlets

I. Présentation

Les servlets sont la base de la programmation Java EE. Une servlet est un composant Web conçu à partir d'une classe java déployée au sein d'une application, interagit avec un client Web par l'intermédiaire du protocole http via le mécanisme de requête/réponse. Bien que, la totalité du traitement puisse être effectuée dans la servlet, celle-ci fait souvent appel à des classes utilitaires pour le logique métier. Elles apportent un contenu dynamique en réponse à des requêtes client.

Une servlet qui s'exécute n'est chargée qu'une fois par le serveur (au démarrage du serveur ou lors de la première requête du client).

II. Avantage

Les servlets ont de nombreux avantages :

- Elles sont portables et évolutives.
- Elles sont performantes et rapides, car chargées en mémoires dès le premier appel.
- Elles sont disponibles, car elles s'exécutent dans un contexte multitâche (une seule instance créée).

5.2.2. Java Server Page

I. Présentation

Les JavaServer Pages sont utilisées pour la génération de contenu dynamique. Elles ont été introduites avec J2EE 1.4 en 1999 afin de développer des pages Web dynamiques de façon plus rapide que les servlets.

Une page JSP est une page HTML pouvant contenir de code Java (Le langage HTML décrit la manière dont s'affiche la page, le code Java servant à effectuer un traitement, par exemple récupérer les informations nécessaires pour effectuer une requête dans une base de données). D'un point de vue technique, il est possible de faire la même chose avec une page JSP qu'avec une servlet. La différence est essentiellement au niveau de la structure. Une page JSP est un squelette de page HTML contenant des morceaux de code Java, permettant de réaliser des traitements dynamiques ou d'intégrer des données.

La technologie JSP permet l'utilisation et la création de balises JSP (taglib) qui ajoute des fonctionnalités pour étendre les possibilités de développement. Le mécanisme JSP permet de séparer la logique contrôlant la présentation des données, du logique métier contrôlant les valeurs des données.

Les JSP ont plusieurs fonctionnalités :

- La création des sites dynamiques (page Web dynamique).
- Le travail avec les bases de données.
- L'amélioration de la sécurité (le code exécuté par le serveur et donc non accessible par les clients).
- L'utilisation des JavaBeeans.
- L'utilisation de balises personnalisées (taglib).

5.2.3. Java Server Faces

I. Présentation

Java Server Faces (JSF) est une technologie dont le but est de proposer un framework qui facilite et standardise le développement d'applications web avec Java. Son développement a tenu compte des différentes expériences acquises lors de l'utilisation des technologies standard pour le développement d'applications web (servlet, JSP) et de différents frameworks (Struts, ...).

Les pages JSF contiennent des balises qui décrivent les composants qui représenteront la page sur le serveur.

II. Service rendu par la JSF

- Permet de bien séparer l'interface utilisateur, la couche de persistance et les processus métier(MVC).
- Conversion des données (tout est texte dans l'interface utilisateur).
- Validation des données saisies par l'utilisateur.
- Automatisation de l'affichage des messages d'erreur en cas de problèmes de conversion ou de validation.
- Support d'Ajax sans programmation javascript (communication en arrière-plan et mise à jour partielle de l'interface utilisateur).
- Fournit des composants standards pour l'interface utilisateur, puissants et faciles à utilise.
- Les pages JSF sont utilisées pour l'interface avec l'utilisateur ; elles ne contiennent pas de traitements (pas de code Java ou autre code comme dans les pages JSP).
- Les backing beans font l'interface entre les pages JSF et le reste de l'application.
- Ces backing beans peuvent effectuer les traitements liés directement à l'interface utilisateur ; ils font appels à des EJB ou des classes Java ordinaires pour effectuer les autres traitements.

5.3. La couche métier

La couche métier offre des services applicatifs et métiers à la couche présentation. Pour fournir ces services, elle s'appuie sur les données du système, accessibles au travers des services de la couche d'accès aux données. En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés.

La couche métier compose donc de trois (3) parties :

a. Logique métier

Elle correspond à la partie fonctionnelle de l'application, celle qui implémente la « logique spécifique de l'application», et qui décrit les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées à travers de la couche présentation.

Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche.

b. Accès aux données

Elle consiste en la partie gérant l'accès aux données du système. Ces données peuvent être propres au système, ou gérées par un autre système. Cette couche permet à la couche métier d'accéder aux données d'une manière uniforme, celle-ci n'a pas à s'adapter aux changements possibles dans les couches supérieures puisque toutes les opérations sont maintenues transparentes par la couche d'accès aux données.

c. Objets de données

Elle assure la persistance des données, à chaque table de la base de données correspond une classe entité, des annotations sont aussi utilisées pour mettre en œuvre les concepts de POO (Programmation Orienté Objet) tels que l'héritage ou le polymorphisme.

5.3.1. EJB

Dans le « Business tiers » (couche métier) la plate-forme java EE se base sur la technologie EJB (Entreprise JavaBeans)

I. Définition

Enterprise JavaBeans (EJB) est une architecture de composants logiciels côté serveur pour la plateforme de développement Java EE.

Cette architecture propose un cadre pour créer des composants distribués (c'est-à-dire déployés sur des serveurs distants) écrit en langage de programmation Java hébergés au sein d'un serveur applicatif permettant de représenter des données (EJB dit *entité*), de proposer des

services avec ou sans conservation d'état entre les appels (EJB dit *session*), ou encore d'accomplir des tâches de manière asynchrone (EJB dit *message*). Tous les EJB peuvent évoluer dans un contexte transactionnel.

II. Type d'EJB

Il existe deux types d'EJB : les beans de session (session beans) et les beans entité (les entity beans). Depuis la version 2.0 des EJB, il existe un troisième type de bean : les beans orientés message (message driven beans). Ces trois types de bean possèdent des points communs notamment celui de devoir être déployés dans un conteneur d'EJB.

a. Beans de session

Les EJB session sont des EJB de service dont la durée de vie correspond à un échange avec un client. Ils contiennent les règles métiers de l'application. Il existe deux types d'EJB session : sans état (stateless) et avec état (stateful).

- **Les EJB session stateful**

Ils sont capables de conserver l'état du bean dans des variables d'instance durant toute la conversation avec un client. Mais ces données ne sont pas persistantes : à la fin de l'échange avec le client, l'instance de l'EJB est détruite et les données sont perdues. De manière concise, voici les propriétés d'un EJB Stateful :

- Des données sont retenues dans l'objet après un appel, c'est-à-dire qu'il conserve un état. On dit alors que l'objet est à état, ou *Stateful* ;
- L'accès à une instance de l'EJB est réservé à un seul client à la fois ;
- Les accès concurrents sont impossibles, le conteneur gère une liste d'attente en cas de tentatives simultanées.

- **Les EJB session stateless**

Ils ne peuvent pas conserver de telles données entre chaque appel du client. De manière concise, voici les propriétés d'un EJB Stateless :

- Aucune donnée n'est retenue ni enregistrée, c'est-à-dire qu'aucun état n'est retenu. On dit alors que l'objet est sans état, ou *Stateless* ;
- Aucun mécanisme ne garantit que deux appels consécutifs à une méthode d'un tel EJB visent une seule et même instance ;
- Les accès concurrents sont impossibles, mais le système est *threadsafe* tout de même puisque le conteneur envoie les requêtes simultanées vers des instances différentes du même EJB.

b. Beans entité

Ces EJB permettent de représenter et de gérer des données enregistrées dans une base de données. Ils implémentent l'interface EntityBean.

L'avantage d'utiliser un tel type d'EJB plutôt que d'utiliser JDBC ou de développer sa propre solution pour mapper les données est que certains services sont pris en charge par le conteneur.

Les beans entités assurent la persistance des données en représentant tout ou une partie d'une table ou d'une vue. Il existe deux types de bean entité :

- **Un bean entité CMP (container-managed persistence) :** c'est le conteneur d'EJB qui assure la persistance des données grâce aux paramètres fournis dans le descripteur de déploiement du bean. Il se charge de toute la logique des traitements de synchronisation entre les données du bean et les données dans la base de données.
- **Un bean entité BMP (bean-managed persistence) :** assure lui-même la persistance des données grâce à du code inclus dans les méthodes du bean.

Plusieurs clients peuvent accéder simultanément à un même EJB entity. La gestion des transactions et des accès concurrents est assurée par le conteneur.

c. Beans orientés message

Un Message Driven Bean est un composant métier recevant des messages de manière asynchrone (Le client n'a pas besoin de figer son exécution durant le traitement du MDB)

Les clients n'appellent pas directement des méthodes mais utilisent JMS pour produire un message et le publier dans une file d'attente

À l'autre bout, le MDB est à l'écoute de cette file d'attente et se « réveille » à l'arrivée d'un message. Il extrait ce dernier de la file d'attente, en récupère le contenu puis exécute un traitement.

5.4. La couche BDD

Elle consiste en la partie gérant l'accès aux de données du système. Ces données peuvent être propres au système, ou gérées par un autre système. La couche métier n'a pas à s'adapter à ces deux cas, ils sont transparents pour elle, et elle accède aux données de manière uniforme et l'accès se fait grâce à l'API JDBC.

API JDBC (Java Database Connectivity)

Une interface standardisée permettant à une application cliente sous Java d'accéder à une base de données, indépendamment du type de ce dernier. Pour pouvoir accéder à une base à travers l'JDBC, il est nécessaire de disposer d'un pilote compatible avec le système de base de données utilisée. Le rôle du pilote est de traduire les requêtes de l'application cliente, faite à travers l'API standardisée du JDBC, au format spécifique de la base serveur (et vice-versa).

JDBC n'est pas spécifique à une plate-forme spécifique.

5.5. Le serveur d'application Java EE

5.5.1. Définition

Le serveur d'application en général c'est l'environnement d'exécution des applications côté serveur. Il prend en charge l'ensemble des fonctionnalités qui permettent à N clients d'utiliser une même application.

Spécialement, les serveurs d'applications de type JEE sont des outils qui permettent l'exécution de composants Java côté serveur (servlets, JSP, EJB, ...) selon les spécifications de la plate-forme J2EE/Java EE qui proposent une architecture et un mode de fonctionnement standardisés.

5.5.2. Les conteneurs du serveur d'application

I. Le conteneur Web

Le conteneur web est un logiciel qui exécute les servlets et les JSP et assure leur cycle de vie. Il fournit des services qui peuvent être utilisés par les applications lors de leur exécution.

II. Le conteneur EJB

Un conteneur EJB est un environnement d'exécution pour un composant EJB. Il s'exécute par un serveur d'applications et prend la responsabilité des problèmes au niveau système.

Le conteneur EJB gère le cycle de vie de l'EJB et fournit aussi un nombre de services additionnels telle que : Gestion de la persistance, la sécurité, cadre de travail pour Business Logic, dimensionnement, portabilité et la gestion des erreurs.

5.5.3. Les APIs de Java EE

Les APIs sont des extensions Java, forment une collection d'outils logiciels permettant d'offrir en standard un certain nombre de fonctionnalités.

JEE regroupe un ensemble d'API pour le développement d'applications d'entreprise qui peuvent être classé en trois grandes catégories que nous verrons dans le tableau ci-dessous :

Catégorie	**Les API**	**Rôle**
Composant	Servlet	Un servlet est un composant coté serveur, écrit en Java, dont le rôle est de fournir une trame générale pour l'implémentation de paradigmes " requête-réponse ". Ils remplacent les scripts CGI tout en apportant des performances bien supérieures
	Java Server Page (JSP)	Elle se sert de balises semblables au XML ainsi que de scriplets Java afin d'incorporer la logique de fabrication directement dans le code HTML. JSP est un concurent direct de l'ASP et du PHP
	Entreprise Java Bean (EJB)	Composants serveurs contenant la logique métier
	Java Server Faces (JSF)	API dont le but est de proposer un framework qui facilite et standardise le développement d'applications web avec Java
Service	JDBC	Une API qui permet aux programmes Java d'interagir avec les bases de données SQL.
	Java Transaction API (JTA) Java Transaction Service (JTS)	La spécification JTA (Java Transaction API) définit des interfaces standards entre un gestionnaire de transactions et les éléments impliqués dans celles-ci : L'application, le gestionnaire de ressources et le serveur.
	Java Naming and Directory Interface (JNDI)	Accès aux services de nommage et aux annuaires d'entreprises
	Java Authentication and Authorisation Service (JAAS)	API permettant de gérer l'authentification et les droits d'accès d'un utilisateur ou d'un groupe d'utilisateur pour une application J2EE
Communication	Remote Method Invocation (RMI)et RMI-IIOP	API permettant la communication synchrone entre des objets Java s'exécutant sur des JVM distinctes
	Java Messaging Service (JMS)	API permettant à une application J2EE de créer, d'envoyer, de recevoir et de lire des données sous forme de messages de manière asynchrone
	Java Mail	API standard de gestion de courriers électroniques. Il permet d'envoyer et de recevoir du courrier électronique et de manipuler les messages (en-tête, sujet, corps, pièces jointes...).

Tableaux 1 : La liste des API de Java EE

5.5.4. Les points forts d'un serveur d'application

Les serveurs d'application possèdent un grands nombre d'avantages :

- Fournit un client beaucoup plus léger. Même si vous avez des codes source de Java que vous distribuez aux utilisateurs, il est très petit,

- Maintien l'intégrité de données et du code en centralisant la logique métier sur un ou sur quelque machines serveur, des mises à jour et des mises à niveau des applications pour tous les utilisateurs peuvent être garanties,
- Configuration centralisée.

Conclusion

Au cours de ce chapitre nous nous sommes familiarisés au fur et à mesure à la technologie Java EE. Cette dernière est adéquate à notre travail car elle rassemble à la fois les avantages d'une plate-forme, de Java et ceux de l'architecture trois (3) tiers.

Notre travail s'est consacré ensuite à l'étude de l'architecture Java EE en couche. Celle-ci se décompose en trois parties essentielles :

- Couche présentation,
- Couche métier,
- Couche Base de Données,

A la fin, nous nous sommes intéressés aux serveurs d'application Java EE et les différentes APIs qu'ils intègrent.