Royaume du Maroc
Université Mohammed V - Souissi
École Nationale Supérieure
d'Informatique et d'Analyse des
Systèmes - **ENSIAS**

Mémoire de Projet de Fin d'Études

Pour l'Obtention du Titre

d'Ingénieur d'État en Informatique

Option : Génie Logiciel

Sujet

Conception et réalisation d'un serveur de change de devises en libre service pour Wincor Nixdorf

Soutenu par :

Najib SAFIR

Sous la direction de :

M. Ahmed ETTALBI

M. Khalid SAADAOUI

Année universitaire 2004 – 2005

Dédicace

A mes très chers parents;
Les mots ne peuvent exprimer ma gratitude envers tout ce que vous avez fait pour moi.

A ma sœur et mes frères
Ilham, Nabil et Réda

A toute ma famille

A ma chère amie Amina

A mes chères sœurs et mes frères de l'ENSIAS

A tous mes amis de Meknès

Pour tous le soutient que vous m'avez offert, je vous dis ***Merci***

A tous ceux qui m'aiment

Je dédie ce travail…

Najib


بسم الله الرحمن الرحيم
رب اشرح لي صدري
ويسر لي أمري

Remerciements

Il m'est agréable d'exprimer ma reconnaissance auprès de toutes les personnes, dont l'intervention au cours de ce projet, a favorisé son aboutissement.

Je tiens à exprimer ma gratitude à M. Khalid SAADAOUI, mon encadrant à WINCOR-NIXDORF, pour ses directives précieuses et ses conseils pertinents qui m'ont appuyé considérablement dans ma démarche.

Mes remerciements les plus sincères à M. Ahmed ETTALBI, mon encadrant à l'ENSIAS, pour les conseils qu'il a prodigué, son judicieux encadrement, ainsi que pour son assistance dans la rédaction du rapport.

Je tiens aussi à remercier tous les membres du jury qui m'ont fait l'honneur d'accepter de juger mon travail.

Je saisis aussi l'occasion pour remercier M. AbdelIlah YASSIR Directeur Marketing Produits qui n'a ménagé ni son temps ni son énergie pour aider à élaborer ce travail. De plus, j'exprime ma profonde gratitude et je tiens à remercier, tout le personnel de WINCOR-NIXDORF pour leur soutien et pour leur générosité considérable quant à l'offre de l'information.

Je profite de l'occasion pour remercier le cadre professoral de l'ENSIAS, pour la formation prodigieuse qu'ils m'ont prodiguée.

Que tous ceux qui m'ont aidé, de près ou de loin, trouvent ici l'expression de mes sentiments les meilleurs.

Abstract

This document is produced during my end-study project within WINCOR-NIXDORF Corporation. As objective, I had to conceive and develop software that exchanges currencies in self-service.

The **ProChangeServer** must be able to carry out all transactions data from remote Automatic Teller Machines and integrate them into the bank daily accountant. As a consequence, the data will be organized and structured. So it can be used later easily. Furthermore, the software offers an administration console to manage all resources.

To that purpose, I take advantage of Y life cycle (2TUP) and I used the UML language for the system modelling.

The software is a Web technologies based solution. In fact, I used several Frameworks under the J2EE platform: Struts for Web Presentation, Hibernate for Object/Relational mapping, JasperReports for Report, Log4j for logging and JAAS API for security.

The software stands on JBoss Application Server and uses Oracle 9i relational Database.

In this document, I will explain more in detail each step I passed in order to accomplish that work.

Résumé

Ce rapport est le fruit du travail que j'ai réalisé dans le cadre de mon projet de fin d'étude au sein de la société WINCOR-NIXDORF. Ce projet avait pour objectif de concevoir et de développer une solution de change de devises en libre service.

Le projet **ProChangeServer** doit permettre la récupération des données des transactions effectuées quotidiennement par les Guichets Automatiques Bancaires pour les intégrer dans l'écriture comptable de la banque. La solution apportée permettra l'organisation de ces données pour une meilleure exploitation. De plus, elle permet la supervision de ces guichets. L'application offre aussi une console d'administration pour gérer ses ressources.

Pour le développement du projet, j'ai suivi le cycle de développement en Y (2TUP). Pour la modélisation du système, j'ai utilisé le langage UML.

La réalisation du projet est basée sur les nouvelles technologies du Web. En effet, j'ai eu recours à plusieurs Frameworks traitant plusieurs aspects du développement sous la plate-forme J2EE. J'ai donc utilisé le Framework Struts pour la présentation Web, le Framework Hibernate pour le mapping objet/relationnel, le Framework JasperReports pour l'éditions des rapports, le Framework Log4j pour la journalisation et l'API JAAS pour la sécurité.

Le système tourne sur le serveur d'application JBoss AS et utilise le système de gestion de base de données relationnelle ORACLE 9i.

Le présent rapport permet de présenter les différentes étapes par lesquelles je suis passé afin de réaliser le travail qui m'a été confié.

Liste des abréviations

Abréviation	Designation
2TUP	2 Track Unified Process
AO	Aspect Orientation
API	Application Programming Interface
ATM	Automatic Teller Machine
BMP	Bean Managed Persistence
CMP	Container Managed Persistence
CORBA	Common Object Request Broker Architecture
EJB	Entreprise Java Bean
GAB	Guichet Automatique Bancaire
GPL	General Public License
IDE	Integrated Development System
IHM	Interface Homme Machine
J2EE	Java Second Enterprise Edition
JAAS	Java Authentication and Authorization Services
JDBC	Java Database Connectivity
JNDI	Java Naming and Directory Interface
JSP	Java Server Pages
LGPL	Lesser General Public License
MAD	MAroc Dirham
MVC	Model View Controller
OMG	Object Management Group
PAQ	Plan Assurance Qualité
RUP	Rational Unified Process
SGBDR	Système de Gestion de Base de Données Relationnelles
SWT	Standard Widget Toolkit
UML	Unified Modeling Language
URL	Uniform Resource Locator
WSW	Websphere Studio Workbench
XML	eXtensible Markup Language

Liste des figures

Table des matières

Introduction

Les sociétés de service en informatique sont les premiers acteurs du développement dans le secteur des technologies de l'information. Ils se trouvent dans un contexte concurrentiel qui les pousse à améliorer la qualité du service offert à la commande des clients, pour pouvoir s'imposer dans le marché.

Dans cette perspective, le modèle métier libre service offre aux utilisateurs la possibilité d'interagir directement avec les systèmes centraux et les bases de données d'arrière-plan de l'entreprise. Parmi les solutions d'actualité, qui profitent de ce modèle, le change automatique des valeurs des devises étrangères en valeur équivalente en devise locale.

Habituellement, la procédure de change est effectuée manuellement. De ce fait, elle présente quelques défauts, par exemple le banquier peut se tromper quand il compte de l'argent, aussi l'échange n'est disponible que lors des heures de travail des fonctionnaires des banques. De plus, les hôtels et les sites touristiques, qui sont les endroits concernés par la solution, sont éloignés du centre de la ville, donc loin des services de base que peuvent en bénéficier entre autre le change des devises étrangères.

Pour combler le manque observé, la société WINCOR-NIXDORF a décidé de lancer un projet visant la réalisation d'une solution de change de devises en libre service. Ce nouveau produit peut être utilisé 24h sur 24, sans arrêt. De ce fait, les transactions de change peuvent être effectuées pendant la nuit et même pendant les jours fériés.

Du point de vue commercial, il faut prendre l'initiative et saisir cette opportunité pour devancer les concurrents en saisissant les nouvelles solutions exprimées sur le marché ; le potentiel est prouvé par l'intérêt économique exprimé par les grandes banques.

Les objectifs de cette solution sont appelés **ProChange**. La partie guichet automatique est appelée **ProChangeATM**, elle est développée à part. Quant à la partie Serveur qui fait l'objet de mon stage de fin d'étude, elle s'appelle **ProChangeServer**.

Dans le premier chapitre de ce rapport, je vais présenter le contexte général du projet. La présentation détaillée du projet fera l'objet du deuxième chapitre. Alors que le troisième chapitre est consacré à l'étude fonctionnelle du projet et le quatrième à l'étude technique. Dans le cinquième chapitre, je vais détailler la conception du projet. Quant au dernier chapitre, il sera réservé à la phase de réalisation. En conclusion, je vais reprendre le travail réalisé et je vais exposer les perspectives envisagées.

A la fin du rapport, je vais présenter en complément les différentes annexes traitant le cycle de développement Y et le plan assurance qualité.

Chapitre 1 : Contexte général du projet

Dans ce chapitre, je vais présenter une fiche descriptive de l'entreprise au sein de laquelle j'ai passé mon projet de fin d'étude. Je vais présenter, aussi, le contexte du projet confié et ses objectifs à atteindre.

1. Présentation de l'organisme d'accueil

1.1. Mission de WINCOR-NIXDORF

WINCOR-NIXDORF est une multinationale allemande qui est une société de services et d'ingénierie industrielle et informatique. Son domaine d'activité est focalisé essentiellement sur les secteurs de la Banque, de la distribution, de la Poste et des grands comptes [wwwWINCOR].

Son activité de veille technologique, renforcée par les retours d'expérience, a permis à WINCOR-NIXDORF de rester en permanence à l'avant-garde des technologies industrielles et informatiques. Elle cultive son savoir-faire pour proposer à ses clients les meilleures solutions adaptées à leurs besoins et leurs environnements de travail. De ce fait, la vocation première de WINCOR-NIXDORF est de servir les intérêts, immédiats et futurs, de tous les clients, en leurs fournissant les informations, les avis et les services dont ils peuvent en avoir besoin.

Ceci peut être résumé comme suit:

- anticiper les nécessités et les besoins du client en lui proposant des solutions adéquates à son métier.
- accompagner l'acquéreur dans son processus d'amélioration pour raffiner ses exigences fonctionnelles et dégager, de ce fait, le choix optimum.
- concevoir et développer les applications spécifiques.
- déployer la solution chez le client en formant les utilisateurs, en les assistant et en fournissant l'expertise requise à travers un support technique adéquat.
- assurer la maintenance des solutions déployées en étant attentifs aux remarques et aux réclamations du client.

Grâce à son expertise reconnue autour des leaders technologiques mondiaux (Oracle, Microsoft, etc.) et à sa renommée auprès de grands comptes nationaux (banques, postes, administrations publiques, etc.), WINCOR-NIXDORF est devenue

un partenaire de choix pour le développement de solutions spécifiques et les solutions à base du libre service.

1.2. Structures et organisation de WINCOR-NIXDORF Maroc

WINCOR-NIXDORF Maroc est organisée selon l'organigramme suivant :

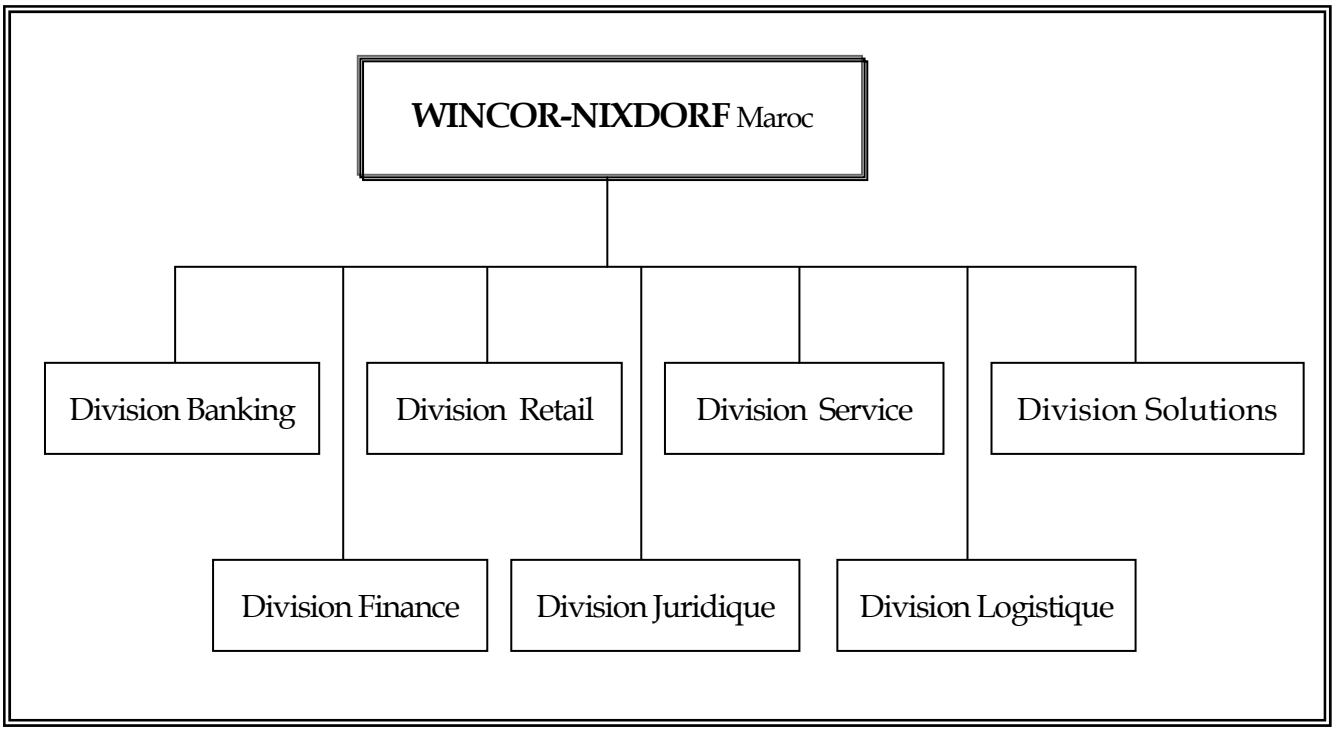


Figure 1 : organigramme de WINCOR-NIXDORF Maroc.

Cependant, les divisions centrales sont en nombre de trois. Elles sont décrites comme suit :

- pôle Banking et Retail.
- pôle Distribution.
- pôle Service.

Dans ce qui suit, je vais présenter ces trois divisions :

- pôle Banking et Retail.

Ce pôle a pour mission de fournir les outils et les solutions relatives aux besoins spécifiques de chaque banque ou poste. Grâce à une conception intégrée des services, elle garantit une valeur ajoutée sensible, une efficacité remarquable et un sérieux engagement.

Les produits qu'offre WINCOR-NIXDORF dans ce domaine sont présentés dans le schéma des canaux de distribution libre-service suivant :

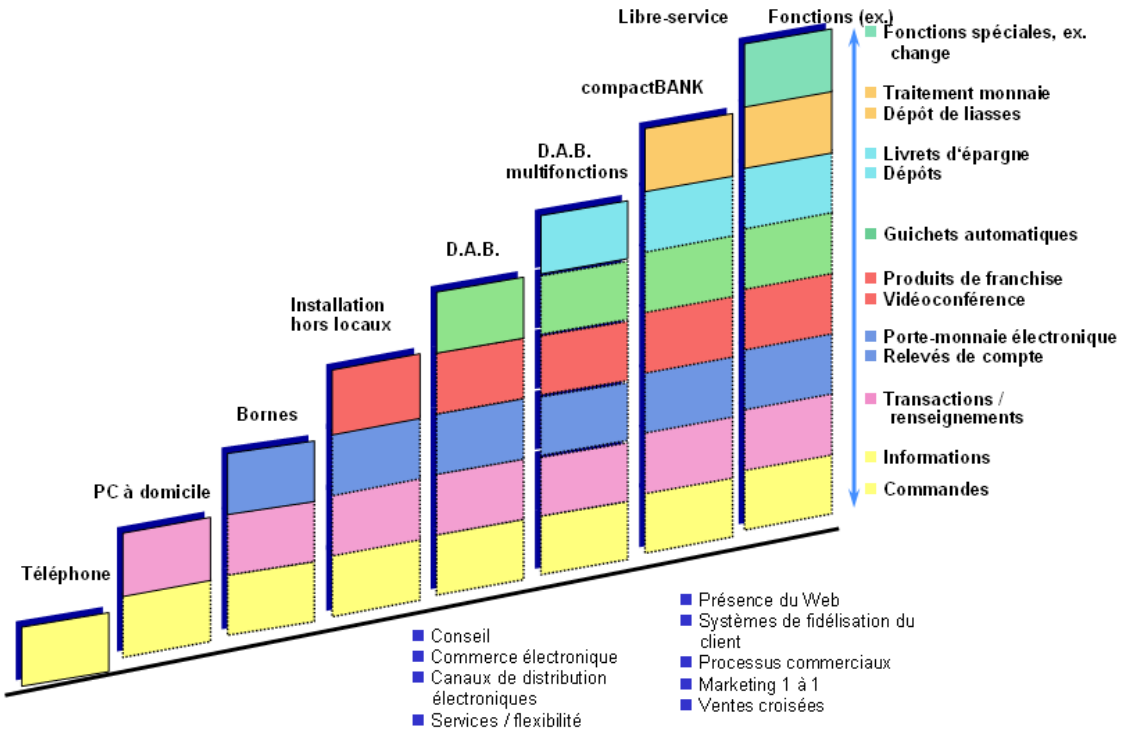


Figure 2 : canaux de distribution libre-service.

- **pôle Distribution**

Ce pôle a pour mission d'élargir le domaine d'activité de la société. Dans cet esprit, le marché de distribution est alimenté par les nouveaux usages du marketing que les entreprises peuvent en bénéficier. Parmi ces usages :

- développer de nouveaux canaux pour les distributeurs par la construction de produits permettant d'élargir leurs cibles actuelles des clients.
- diversifier la nature des clients en exploitant les opportunités offertes dans les domaines tels que la loterie, l'hôtellerie, etc.

Le schéma suivant présente l'offre de solutions complètes :

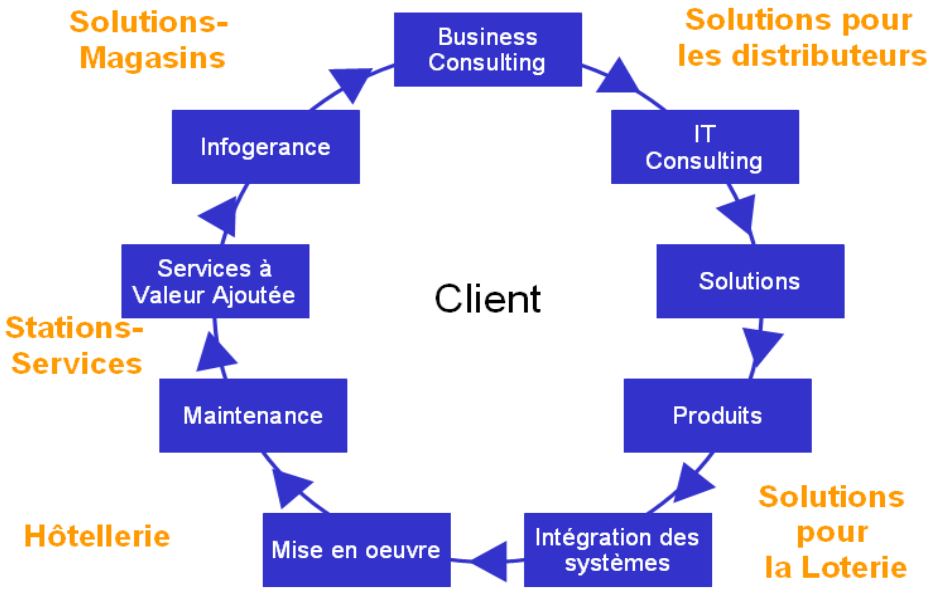


Figure 3 : offre de solutions complètes.

- **pôle Service**

Les professionnels de Wincor-Nixdorf offrent leurs compétences dans des domaines précis, de la formation à la maintenance, de l'intégration au développement d'applications spécifiques.

Le tableau suivant résume les fonctions remplies par ce pôle :

Call Center National	Réception de tous les appels provenant du Maroc.
Centre de Dispatching	filtrage des appels. qualification et diagnostic techniques des appels. affectation des techniciens. suivi de la résolution des pannes.
Help Desk National	assistance téléphonique aux utilisateurs. assistance téléphonique aux techniciens sur site.
Stock des pièces	gestion des stocks de pièces de rechange.
Maintenance préventive	gestion des plannings de maintenance préventive avec le client.
Centre de réparation	réparation des matériels ne pouvant être réparés sur site. analyse individualisée des pièces fréquemment en panne. proposition de solutions techniques.

Figure 4 : fonctions remplies par le pôle service.

2. Présentation du sujet

Le projet consiste en la conception et la réalisation d'un serveur de change de devises en libre service « **ProChangeServer** » pour la société de service WINCOR-NIXDORF. Dans ce qui suit, je vais situer le contexte de ce projet et en étaler les objectifs.

2.1. Contexte du projet

Le modèle métier libre service offre aux utilisateurs la possibilité d'interagir directement avec les systèmes centraux et les bases de données d'arrière-plan de l'entreprise. Parmi les solutions d'actualité, qui profitent de ce modèle, le change

automatique des valeurs des devises étrangères en valeur équivalente en devise locale.

Quelle est l'utilité du change ?

Notre pays se voit sensibilisé de l'importance du domaine touristique pour son développement socio-économique. Les touristes sont de nationalités variées, donc utilisent des devises autres que notre devise locale ; le Dirham. A cet égard, l'un des services fondamentaux qu'on doit fournir étant d'offrir la possibilité aux touristes d'utiliser la monnaie locale.

Pourquoi automatiser cette procédure ?

Habituellement, la procédure de change est effectuée manuellement. Ce qui présente quelques défauts. Les aspects du problème peuvent être résumés comme suit :

- le service de change n'est disponible que lors des heures de travail des fonctionnaires des banques qui ont la possibilité d'effectuer de telles transactions (9h-12h et 14h- 17h).
- le banquier peut se tromper quand il compte de l'argent.
- l'échange manuel est un travail répétitif sans valeur ajoutée pour le banquier. Une approche plus avisée serait de se servir des ressources humaines dans des travaux acquérants plus de réflexion.
- les hôtels et les sites touristiques, qui sont les endroits concernés par la solution, sont éloignés du centre de la ville, donc loin des services de base que peuvent en bénéficier entre autre le change des devises étrangères.

En tenant compte des inconvénients de la procédure manuelle ainsi dégagés, l'automatisation des transactions de change offre plusieurs avantages :

- la solution peut être utilisée 24h sur 24, sans arrêt. Il suffit d'alimenter le guichet par des billets suffisants.
- la flexibilité, en dehors du temps de travail habituel, est assurée, ainsi les transactions de change peuvent être effectuées pendant la nuit comme pendant les jours fériés.

- l'intégration au système existant est prise en considération par l'utilisation des GABs qui effectuent les transactions de retrait ordinaires ; cette amélioration est prévue par l'adoption d'une nouvelle génération de guichets. Ils devront permettre de rendre la monnaie en pièce jusqu'au 5 centimes près.

- la sûreté des billets contenus dans les guichets automatiques est garantie par le fait que les GABs sont blindés, donc les billets sont chargés dans un coffre fort.

- le champ, visé par l'application, peut être élargi pour atteindre d'autres places aussi importantes telles que les aéroports et les supermarchés.

Du point de vue commercial, il faut prendre l'initiative et saisir cette opportunité pour devancer les concurrents en saisissant les nouvelles solutions exprimées sur le marché, car ce nouveau marché suscite un intérêt économique exprimé par les grandes banques et les grandes structures commerciales.

Aussi, il y a élargissement du domaine d'activité de l'entreprise pour atteindre le marché international, ainsi l'exploitation de la solution peut être bénéfique pour d'autres pays qui expriment les mêmes besoins expliqués auparavant. Cette solution est déployée en intranet. Elle va donc permettre un gain énorme de temps de traitement et une facilité notable pour le change de devise.

Les objectifs de cette solution sont appelés **ProChange**. La partie guichet automatique, appelée **ProChangeATM**, est développée en Pologne. Quant à la partie Serveur, appelée **ProChangeServer**, est présentée dans le paragraphe qui suit. L'intégration de ces deux parties, en un produit, a eu besoin de plusieurs itérations pour l'aboutissement du **ProChange** dans sa version fonctionnelle et exploitable.

2.2. Objectifs du projet

Le projet **ProChangeServer** a pour objectif de récupérer les données des transactions effectuées quotidiennement pour les intégrer dans l'écriture comptable de la banque ou l'institution financière en question. Pour aboutir à ce résultat, la réalisation est appréhendée par l'accomplissement des points suivants :

- distribuer les paramètres des GABs pour assurer leurs fonctionnements normaux. Ces paramètres sont regroupés selon des types fonctionnels et d'environnement : timeout, messages, contenu du reçu, périodicité des balances, etc.

- distribuer les taux de changes aux GABs chaque fois qu'une mise à jour est effectuée. La variation du taux de change est contrôlée, puisque la validation de la nouvelle valeur doit être accordée à un utilisateur privilégié.

- collecter les données quotidiennes des transactions de change qui sont éparpillées sur plusieurs guichets automatiques. Ces données contiennent différents types d'informations ; comme le montant en Dirham (ou en devise étrangère) d'une transaction donnée ou bien les informations sur les coupures[1] utilisées.

- consolider ces données, reçues sous format texte, sous une interface unique dans une base de données permettant par la suite d'extraire les renseignements et des indicateurs pertinents au suivi et au monitoring.

- anticiper les anomalies qui peuvent survenir et prévoir les traitements adéquats relatifs à chacun des cas ; ces problèmes sont d'origine réseau en général.

- gérer toutes les ressources de l'application en permettant l'ajout, la mise à jours et la suppression ; gérer les GABs, les devises, les utilisateurs, les profiles, les messages, le multi langage et les paramètres généraux du serveur **ProChangeServer**.

- assurer un service d'édition afin de pouvoir gérer les rapports et les statistiques relatifs aux principales informations utilisées par la banque dans son fonctionnement. De plus, permettre l'exportation des données sous formats standards tels que les fichiers PDF, HTML, EXCEL et XML.

- protéger le système par un mécanisme de sécurité flexible permettant d'accéder aux fonctionnalités du système selon le profil de l'utilisateur avec différents niveaux de granularité.

[1] Coupure : une portion d'argent représentée par une seule unité. Le Dirham marocain a les coupures en billets de 200, 100, 50 et 20. Aussi, il a les coupures en pièces de 10, 5, 2, 1, 0.5, 0.2, 0.1 et 0.05.

- doter **ProChangeServer** d'un dispositif de journalisation afin de pouvoir retrouver la trace du déroulement de l'exécution du système.

En plus, **ProChangeServer** doit être facilement exploitable, fiable et extensible, ainsi il doit contenir aussi :

- un guide utilisateur pour expliquer aux utilisateurs comment utiliser et configurer le système.
- un guide développeur qui explique l'architecture du système pour permettre aux développeurs d'étendre le système si le besoin est exprimé.

Conclusion

Dans ce chapitre, j'ai présenté la société WINCOR-NIXDORF chez laquelle j'ai passé le stage de fin d'étude. J'ai aussi présenté le contexte dans lequel s'inscrit le projet **ProChangeServer**.

Dans le chapitre suivant, je vais aborder en détail l'étude du projet où je vais dégager les fonctionnalités du système et présenter le dossier de pilotage.

Chapitre 2 : Étude du projet

Dans ce chapitre, je vais présenter le projet en détail afin d'en étaler les fonctionnalités et d'en estimer l'envergure. Ensuite, je vais dévoiler le dossier pilotage du projet en question pour atteindre les objectifs fixés.

1. Caractéristiques générales de ProChangeServer

L'organisation, d'une banque, a un grand impact sur la manière dont les données, concernant ses activités, sont traitées. Et comme l'organisation diffère, la solution **ProChangeServer** est destinée pour être exploité par n'importe quelle banque. Pour cela, elle doit supporter tout type d'organisation.

Compte tenu de la différence entre les systèmes d'information des banques, **ProChangeServer** doit offrir la possibilité d'être déployé sur différents systèmes d'exploitation (Windows, Linux) et de supporter différents SGBDR (Oracle, SQL Server et autres). De ce fait, il serait plus flexible pour l'utilisateur final.

Les systèmes d'informations bancaires deviennent de plus en plus complexes et exigeants, donc pour pouvoir intégrer **ProChangeServer** dans le système d'information de la banque, il doit être paramétrable et extensible.

De nos jours, les architectures logiciels et les outils (de modélisation et de développement) sont de plus en plus nombreux. Ce qui signifie que la mise en œuvre de **ProChangeServer** doit tenir compte de ces variantes afin d'assurer un meilleur fonctionnement. Et bien sur cela n'est possible qu'en prenant compte des considérations relatives à la fiabilité des outils et leurs compatibilités dans l'élaboration de l'environnement technique ciblé par les banques.

2. Fonctions principales de ProChangeServer

2.1 Gestion des balances des GABs

La balance de change est l'ensemble d'informations relatives à chaque GAB effectuant des opérations de change. Ces données sont remontées au serveur quotidiennement et sont subdivisées en 5 volets qui sont :

- récapitulatif de la journée.
- liste des transactions.

- détail des transactions par devise.
- détail des transactions par les coupures de la devise étrangère.
- détail des transactions par les coupures MAD (billets et pièces).

La gestion des balances des GABs englobe la circonspection des cinq sections suivantes :

Section 1 : Collecte des balances

Automatiquement et à une heure paramétrée, les balances disponibles sont assemblées. A ce moment, pour chaque GAB, deux cas se présentent : soit il a transmis sa balance soit il ne l'a pas encore envoyé (à cause d'un problème réseau par exemple).

Pour les GABs ayant transmis leurs transactions quotidiennes, **ProChangeServer** récupère ces données et passe à leurs traitements. Pour les GABs n'ayant pas transmis leurs transactions quotidiennes, **ProChangeServer** attend une période déterminée puis recommence la procédure de récupération pour les GABs restants.

Section 2 : Traitement des balances

A cette étape, le serveur vérifie la conformité des données au format de transmission permit et examine leurs valeurs sémantiques. Ceci étant examiné pour chaque GAB. Là encore, deux cas se présentent ; une balance est soit déclarée valide soit elle est dite invalide.

Si une balance est jugée valide, elle est insérée dans la base de données. En même temps, il y a préparation de l'environnement pour les balances prochaines. Si une balance est non certifiée, elle est déclarée rejetée et il y a appel de la section suivante.

Section 3 : Gestion des balances rejetées et non envoyées

Pour chaque GAB n'ayant pas transmis sa balance, soit elle était mal reçue soit elle était témoignée non valide, l'administrateur système ou le superviseur des GABs

doit intervenir pour récupérer cette balance. Dans ce cas, la balance peut être récupérable dans le même processus normal ou elle peut ne pas l'être.

Pour la balance récupérable, le processus de collecte est relancé manuellement. Cependant, si la configuration de la collecte est automatique, le processus se lance à une heure paramétrée après résolution des problèmes survenus[2]. L'intégration dans la base de données est aussitôt effectuée. Pour les balances irrécupérables[3], le système permet la saisie manuelle de toutes les transactions de change.

Section 4 : Saisie des balances irrécupérables

S'il n'est pas possible de récupérer la balance d'un GAB donné, **ProChangeServer** permet la saisie manuelle de toutes les transactions de change effectuées par ce GAB. Afin de prévoir ce cas, chaque GAB est doté d'un système de trace représenté par un journal des transactions. Celui-ci est un dispositif similaire à la quittance reçue par le client.

Pour réaliser cette fonctionnalité, **ProChangeServer** offre une interface de saisie pour le gestionnaire du GAB. Ce dernier doit d'abord ouvrir une période de saisie, puis saisir à partir du journal les transactions effectuées. Une fois la saisie terminée, le gestionnaire doit clôturer cette période. Après ladite clôture, l'administrateur système doit valider cette saisie pour relancer le processus d'intégration de ces données.

2.2. Préparation de la balance des GABs

La balance de change est le récapitulatif d'informations relatives à tous les GABs effectuant les opérations de change pendant une période bien explicitée. Elle est calculée pour faire le point sur les activités effectuées. Une telle synthèse est envoyée au serveur central de la banque soit automatiquement soit manuellement selon les objectifs de l'administration.

[2] Problème réseau en général.
[3] Crash du disque dur en général.

La périodicité de la balance globale peut coïncider avec celle de la balance des GABs, mais peut dépendre de la clôture des écritures comptables de la banque. Dans ce dernier cas, la synthèse tient compte des jours fériés et des week-ends donc il s'agit de gérer un calendrier dynamique.

La balance globale est aussi paramétrable dans la mesure où elle peut être générée par institution ou produite par agence, etc. La structure de la balance globale est constituée par le chiffre d'affaire quotidien en dirham réalisé ainsi que le total des commissions en profit. Elle contient, aussi, le montant total acquis par devise étrangère.

Pour les balances GABs non envoyées à temps, le système **ProChangeServer** permet soit de les intégrer dans la balance en cours soit de les envoyer comme des balances complémentaires à celles envoyées incomplètes.

2.3. Gestion des taux de change

La gestion des taux de change inclut la prise en charge de leurs valeurs et de leurs durées de validité. Si une mise à jour est nécessaire, les nouvelles valeurs peuvent être modifiées de deux façons différentes :

- saisie manuelle par un utilisateur habilité à le faire.
- chargement à partir d'un serveur dédié. Auquel cas, il faut utiliser un mécanisme sécurisé.

Si la banque opte pour une mise à jour manuelle des taux de change, le responsable chargé de cette mission devra créer une nouvelle configuration contenant les dernières valeurs des taux de change utilisées. Si cette personne décide d'apporter des modifications aux taux de change, cette nouvelle configuration sera affectée à tous les GABs.

La distribution des taux de change est contrôlée par un mécanisme d'autorisation. Si une nouvelle mise à jour est effectuée par un utilisateur, il introduit une variation relative aux taux de change. Ainsi si cette variation dépasse un seuil qui lui est accordé, le système refuse de tenir compte de ces nouvelles modifications.

Si les nouveaux taux de change sont récupérés à partir d'un serveur tiers, la création de la configuration ainsi que sa distribution seront semblables à celles relatives à la méthode manuelle. Le mécanisme de validation est aussi appliqué en affectant un profil au serveur des taux de change.

Le système doit sauvegarder les valeurs des taux de change dans la base de données permettant, ainsi, leurs consultations pour une analyse a posteriori.

2.4. Gestion des paramètres des GABs

Lors de son fonctionnement, le GAB doit prendre en considération les différents types de paramètres relatifs à l'environnement d'exécution (timeout, messages, contenu du reçu, périodicité des balances, etc.).

Ces paramètres sont regroupés en configurations et c'est le système qui se charge de les fournir aux GABs déclarés actifs. Ainsi, toute création d'une nouvelle configuration doit suivre une procédure bien définie. Celle-ci est semblable à la procédure appliquée aux taux de change expliquée dans la partie « gestion des taux de change ».

Compte tenu de la multitude des paramètres à gérer, le mécanisme de chargement d'une nouvelle configuration doit être organisé. Ces paramètres sont donc classés par catégories fonctionnelles (paramètres système, paramètres d'initialisation, paramètres de fonctionnement, timeout, etc.) pour faciliter leur gestion.

Dans cette vision, à chaque paramètre est associé une catégorie. L'administration de cette catégorie est affectée à un groupe d'utilisateurs. Par conséquent, un utilisateur peut agir sur une catégorie particulière de paramètres sans avoir accès à une deuxième catégorie.

2.5. Paramètres généraux

- **Hiérarchie des GABs**

L'hiérarchie d'un GAB représente son niveau d'appartenance dans l'échelle de la banque. Cependant, le degré hiérarchique d'un GAB peut différer d'une banque à

une autre. Pour englober différents types de hiérarchie, le système propose l'hiérarchie suivante :

Mandataire

Institution

Agence

GAB

Le mandataire englobe des institutions ; chacune de celles-ci organise ses agences et chaque agence gère un ou plusieurs GABs. Par exemple, le mandataire ATTIJARI-WAFA est constitué de deux institutions : BCM et WAFA Banque où chacune d'elles ayant des agences réparties dans différentes villes. D'autre part, ce modèle permet de répondre à la tendance des banques pour la fusion.

Un mandataire est responsable de plusieurs banques. Par exemple, le groupe Banque Populaire supervise des banques localisées au niveau de chaque région. Celles–ci, contrôlent leurs agences de façon autonome. Généralement, le système permet de gérer l'hiérarchie d'une banque en s'adaptant au modèle de son arborescence.

- **Gestion des GABs**

ProChangeServer permet de gérer tous les GABs offrants le service de change d'une banque donnée, ainsi il reconnaît chaque GAB par son numéro, l'agence à qui il appartient et l'adresse où il est localisé. De plus, on indique, le nombre de cassettes qu'il contient, l'existence ou non du « coins dispenser[4] », son statut et la version du GAB installé : « Standard » ou « Advanced ».

Une machine « Standard » ne contient pas de « coins dispenser ». Une machine « Advanced » contient le « coins dispenser ». Deux cas se présentent, soit la version du GAB prend en compte l'utilisation du « coins dispenser », soit elle ne le permet pas.

[4] Le « coins dispenser » est un supplément qui contient les pièces de monnaies.

Quant au statut du GAB, on indique trois états : activé ou désactivé ou supprimé. Initialement, il est déclaré comme désactivé et il sera activé lorsqu'il est opérationnel. Ceci étant pour faciliter la gestion.

- **Gestion des devises**

ProChangeServer gère toutes les devises permises à circuler dans le territoire marocain. Chaque devise est caractérisée par son code ISO, son nom, son unité et son statut qui peut prendre l'état activé ou non activé.

L'activation d'une devise est conditionnée par le nombre maximal de devises supportées par le GAB et par la liste des devises pouvant être échangée au Maroc selon les notifications dictées par BANK AL MAGHRIB à cet égard. Finalement, il faut noter que l'activation de la devise sera manuelle et sera contrôlée par une personne responsable.

- **Gestion des profils et des utilisateurs**

Puisque différents intervenants utilisent **ProChangeServer**, leur administration est assurée. En effet, chaque utilisateur a son login, son mot de passe, son nom, la date début validité, la date fin validité, activé ou pas et le profil associé. Ainsi, la gestion des profils consiste à gérer par profil son nom et ses droits.

ProChangeServer permet d'ajouter un nouvel utilisateur. Une fois intégré, il lui serait donc possible de modifier ses caractères, son profil et d'autres paramètres. De plus, le cas de suppression est accepté. Toutefois, il faut l'utiliser selon les règles de gestion appliquées en vigueur.

Il est possible, entre autre, de permettre à une personne physique plusieurs rôles dans le système. La gestion des droits, dans ce cas, doit prendre en compte la confrontation des accès. Par exemple : accès refusé pour un objet + accès accepté pour cet objet = accès refusé pour cet objet.

- **Gestion des messages et le multi langage**

Le caractère d'internationalisation est pris en compte pour chaque utilisateur. Ceci étant géré au niveau de chaque menu et au niveau de chaque fenêtre grâce à l'interface de l'application qui s'adapte selon la langue sélectionnée par l'utilisateur.

Afin de permettre un modèle générique d'interface, une fenêtre donnée peut être présentée pour tous les utilisateurs d'un profil. Ce qui va les distinguer c'est l'activation ou la désactivation d'un menu ou d'un bouton ou d'un autre composant de l'interface.

D'autre part, le système intègre l'affichage des messages d'accompagnement de l'utilisateur d'une manière proche à sa compréhension ; il s'agit du retour d'informations des événements déclanchés soit par l'utilisateur soit par le système. De plus, le système reporte les changements d'états relatifs à l'objet manipulé ; assurant de ce fait, la validation de l'opération effectuée ou du renvoi d'une notification expliquant la cause et la nature de l'erreur.

- **Gestion des paramètres ProChangeServer**

Le fonctionnement du Serveur est régit par la consultation d'une multitude de variables de configuration. Celles-ci sont très utiles pour l'administration du système (chemin des fichiers de la balance, chemin du fichier taux, chemin du fichier paramètre, périodicité du lancement du processus de collecte des balances, durée du processus de collecte, etc.).

Comme les paramètres à administrer sont nombreux, ces paramètres sont classés par catégories fonctionnelles : des paramètres système, des paramètres de disponibilité, des paramètres de sécurité et des paramètres de performance.

- **Rapports et consultations**

Pour permettre la consultation des données des transactions effectuées, l'application permettra de générer des rapports relatifs aux principales informations utilisées par la banque. Pour la génération et l'exploitation de ces éditions, l'utilisateur ne sera

pas perdu par la multitude de leurs menus. Ainsi, un modèle générique sera exploité par toutes les requêtes de consultation.

Pour enrichir l'utilisation des consultations, l'application permettra de les sauvegarder dans des fichiers sous une multitude de format : XML, HTML, PDF et EXCEL. L'édition permet de publier différents états et rapports concernant l'activité de change sur l'ensemble des GABs :

- informations sur la transaction.
- informations sur les transactions par rapport à une devise ou par coupure.
- informations sur la balance quotidienne ou la balance globale.

3. Fonctions optionnelles de ProChangeServer

3.1. Purge de la base de données

La purge de la base de données permet de supprimer les anciennes transactions pour augmenter les performances du SGBD. Les données supprimées seront archivées et il est possible de les restaurer pour des fins de consultation ou de diagnostique. Cette fonction est très importante vu le volume des transactions de change qui seront effectuées.

3.2. Purge des fichiers

Compte tenu du volume des fichiers balance des GABs chaque année qui est d'environ 9 Go (365 jours, 100 GABs, 500 transactions/jour et 500 octets /transaction/GAB), il serait judicieux d'archiver les balances sur des supports autre que le disque dur pour augmenter ainsi la performance.

3.3. Traitement des réclamations

Afin d'intégrer les remarques et les critiques des utilisateurs des GABs, le système propose un volet réclamation. Celui-ci doit être pris en considération comme un attribut de l'entité transaction pour optimiser le temps de calcul lors du traitement

des réclamations. Sans ce traitement, la jointure pour inclure les réclamations émises dégradera énormément la performance de la base de données.

3.4. Univers éditions

Pour permettre aux décideurs une vue globale et détaillée du système d'information géré par **ProChangeServer**, il serait judicieux d'offrir un univers d'édition plus professionnel. Pour cette fin, il y a des logiciels avancés qui peuvent répondre aux attentes désirées. L'outil « Business Object », entre autres, permet à l'utilisateur de générer des modèles d'états personnalisés et avancés. Le rôle de la solution offerte est de créer l'environnement de base afin de faciliter l'exploitation des données par ces outils dédiés.

4. Objectifs en terme de qualité

Dans les paragraphes précédents, j'ai détaillé les exigences fonctionnelles du futur système **ProChangeServer**. Dans ce qui suit, je vais présenter les objectifs du projet en terme de qualité.

Le cahier de charge constitue un conducteur majeur qui guide l'accomplissement du projet. Cependant, tout développement logiciel doit répondre aux besoins implicites du projet. Parmi ces contraintes, il y a les facteurs de qualité suivants :

- disponibilité : le système doit être en permanence à la disposition de ses utilisateurs.
- fiabilité : le système doit exécuter les fonctions attendues avec la précision requise (taux de défaillance minimal).
- évolutivité : il doit être possible d'étendre le système.
- réutilisation : il doit avoir la possibilité de réutiliser certains modules du système.
- indépendance de l'outil : surtout pour les services techniques, il doit être possible de changer l'outil utilisé sans beaucoup trop de changements à effectuer.

5. Dossier de pilotage

Le dossier de pilotage est un guide qui rassemble un ensemble de volets permettant d'atteindre les objectifs fixés pour le projet sous la contrainte du délai fournit, la démarche suivie et la conduite de projet. Dans cette vision, je présente le plan d'assurance qualité, le planning du projet et le cycle de développement suivi. Ils seront détaillés dans les trois parties suivantes.

5.1. Plan assurance qualité

Dans le but d'obtenir un logiciel satisfaisant les besoins fonctionnels, techniques et qualité, un plan d'assurance qualité est réalisé. Ce plan contient toutes les orientations suivies et appliquées. La description totale du PAQ est présentée dans l'annexe 2.

5.2. Cycle de développement

Le processus de développement adopté est le processus en Y ou Two Track Unified Process (2TUP) [wwwCLUBJAVA]. Ce processus itératif et incrémental, dérive du « Unified Process », mais il consacre plus d'importance pour les aspects techniques auxquels il réserve toute une branche de son cycle. Cet intérêt a pour but de réduire le risque technologique.

Y ne se contente pas des risques technologiques, mais il assure aussi une gestion des risques de toute nature. Pour plus de détail concernant le cycle en Y se reporter à l'annexe 1.

Le choix du cycle en Y est justifié par ses propriétés confirmées et par le besoin exprimé par **ProChangeServer** en terme de fonctionnalités et des technologies à utiliser. Néanmoins, une étude comparative entre le cycle en Y et d'autres cycles de développement s'impose afin d'en choisir celui qui est le plus adapté. L'étude a prouvé la puissance du cycle en Y et elle a démontré qu'il est approprié au projet.

Cette comparaison porte sur trois processus de développements les plus populaires et les plus utilisés :

- le processus RUP (Rational Unified Process).
- le cycle en V.
- le cycle en Y.

La figure5 suivante est un récapitulatif de l'étude comparative réalisée [wwwIMPROVE] [wwwPSTMARTIN].

Les aspects critiques, du projet étudié, méritent d'être résolus avec plus de souplesse. Parmi ces points, la multitude des fonctionnalités exprimées, l'utilisation des plus nouvelles technologies, le respect du temps réservé à l'élaboration du projet, etc. Pour éviter cette immensité des risques, je me suis concentré sur l'élaboration d'une première version du projet n'ayant que les fonctionnalités principales. Ensuite, j'ajoutais les fonctions les plus indispensables au fur et à mesure. Ceci étant établi par une méthode itérative et incrémentale.

Pour esquiver les risques des nouveautés technologiques, j'explorais les solutions offertes pour s'y adapter et pour surpasser les surprises qui peuvent en survenir. Le modèle d'implémentation donc tire de grands profits des solutions offertes par les technologies étudiées.

D'après le la figure 5, seul Y propose un intérêt vif pour la gestion de la complexité technique. Les deux autres n'y attachent pas grande attention. En effet, RUP propose plutôt un cadre complet pour la conduite de projet, mais n'attache pas trop d'importance pour le développement lui-même. Le cycle en V quant à lui, permet de maîtriser le développement à travers les vérifications à la fin des phases, mais il n'offre toujours pas de gestion de la complexité technique.

Il s'avère donc incontestablement que c'est effectivement 2TUP qui est le plus approprié au projet étudié.

Processus	Description	Avantages	Inconvénients
RUP	RUP est à la fois une méthodologie de conduite et de développement de projets et un outil prêt à l'emploi	Propose des modèles de documents, et des canevas pour des projets types	-Coûteux à personnaliser -Très axé processus, au détriment du développement : peu de place pour le code et la technologie
V	Chaque phase en amont de la production du logiciel prépare la phase correspondante de vérification en aval de la production du logiciel	Préparation des phases de vérification au moment de l'Analyse et de la Conception	-Obligation de définir la totalité des besoins au départ -Validation fonctionnelle tardive
2TUP	S'articule autour de l'architecture	Fait une large place à la technologie et à la gestion du risque	Ne propose pas de documents types

Figure 5 : étude comparative des cycles de développement.

5.3. Planning

Dés la première réunion avec l'encadrant à WINCOR-NIXDORF, nous avions élaboré un planning prévisionnel (la figure 6 suivante) que nous avons ajusté par la suite au fur et à mesure de l'avancement du projet. Les tâches inscrites dans ce planning correspondent aux phases caractérisant le cycle du développement Y.

Id	Tâche	Date de début	Date de fin	Durée
1	**Etude préliminaire**	**21/02/2005**	**25/02/2005**	**5 j**
2	**Etude Fonctionnelle**	**28/02/2005**	**18/03/2005**	**15 j**
2.1	Capture des besoins fonctionnels	28/02/2005	04/03/2005	5 j
2.2	Analyse	07/03/2005	18/03/2005	10 j
3	**Etude technique**	**21/03/2005**	**08/04/2005**	**15 j**
3.1	Architecture logicielle et outils	21/03/2005	25/03/2005	5 j
3.2	Développement des Frameworks techniques	28/03/2005	08/04/2005	10 j
4	**Conception**	**11/04/2005**	**29/04/2005**	**15 j**
5	**Codage et test**	**02/05/2005**	**01/07/2005**	**45 j**
Total				**95j**

Figure 6 : planning prévisionnel.

Conclusion

Dans ce chapitre, j'ai détaillé les exigences fonctionnelles et techniques de la solution **ProChangeServer**. Puis, j'ai étalé les objectifs en terme de qualité de cette solution. Ensuite, j'ai présenté le dossier de pilotage.

Dans le chapitre suivant, je vais aborder en détail l'étude fonctionnelle du projet où je vais présenter la phase de capture des besoins fonctionnels et celle d'analyse.

Chapitre 3 : Étude fonctionnelle du projet

Dans ce chapitre, je vais présenter la phase de capture des besoins fonctionnels où j'ai modélisé le système du point de vue usager/système. Elles sont raffinées par les diagrammes de cas d'utilisation et par la maquette IHM. Je vais exhiber aussi la phase d'analyse où j'ai détaillé le modèle d'analyse construit.

1. Capture des besoins fonctionnels

Lors de cette phase, je vais déterminer l'ensemble des services offerts par le système **ProChangeServer**, à travers l'énumération et la description textuelle de l'ensemble des cas d'utilisation inspectés du système. Par ailleurs, je vais dévoiler la maquette interface Homme/Machine adoptée par le système.

1.1. Services attendus de ProChangeServer

Dans ce paragraphe, je vais dégager l'ensemble des cas d'utilisation du système. Ce seront toutes les manipulations que les différents utilisateurs effectuent en interagissant avec le système. Afin d'y parachever, je vais suivre la démarche suivante :

- définir les acteurs du système.
- lister les cas d'utilisation.
- donner une description textuelle des scénarios.
- construire les diagrammes de séquence associés aux scénarios.

Les acteurs principaux du système sont les personnes qui utilisent les fonctions principales du système [MULLERGAERTNER00]. Dans le cas de **ProChangeServer**, ce sont les acteurs : décideur, banquier et chef banquier. Les acteurs secondaires du système sont les personnes qui effectuent des tâches administratives ou de maintenance sur le système [MULLERGAERTNER00]. Ce seront les acteurs : administrateur et guichetier. Afin d'effectuer l'inventaire des divers cas d'utilisation du système, il suffit de recenser les attentes et les opportunités offertes à chacun des utilisateurs dégagés auparavant.

L'acteur décideur va utiliser le système pour consulter l'état d'avancement des balances quotidiennes et de la balance globale générée. De plus, il peut consulter les statistiques sur toutes les activités exécutées. Il doit donc s'authentifier pour accomplir les tâches qui lui sont permises. Le diagramme de cas d'utilisation du système associés à l'acteur décideur est énoncé dans la figure suivante :

Figure 7 : diagramme de cas d'utilisation associé à l'acteur décideur.

L'acteur banquier va utiliser le système plus fréquemment. Il va s'occuper de la collecte des balances récupérées à partir des différents GABs. De plus, il va intervenir si l'un des GABs actifs a envoyé une balance non valide (à cause des problèmes réseaux par exemple).

Quant les balances GABs sont acceptées, il archive les fichiers reçus. Par contre, si elles sont erronées, il va signaler le problème et va contacter le guichetier pour recommencer le transfert des données. Par ailleurs, le banquier va mettre à jour les taux de change et va saisir les paramètres relatifs aux balances des GABs et de la balance globale. Il doit donc, lui aussi, s'authentifier pour accomplir les tâches qui lui sont permises.

Le diagramme de cas d'utilisation du système associé à l'acteur banquier est exposé dans la figure suivante :

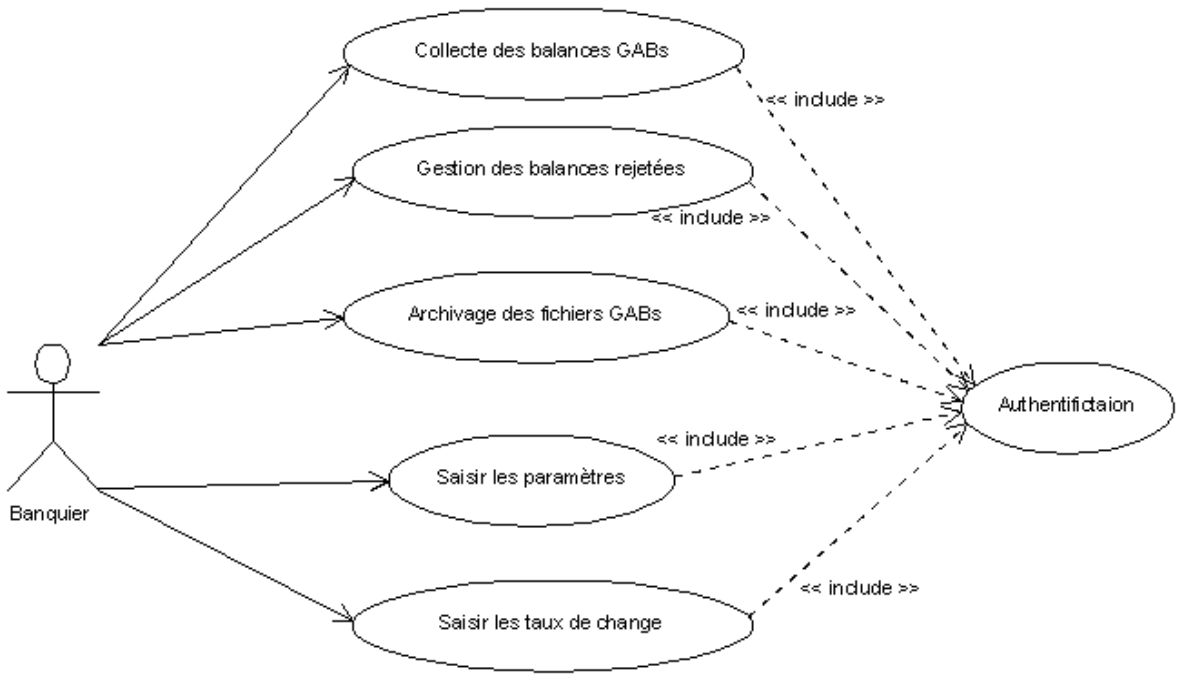


Figure 8 : diagramme de cas d'utilisation associé à l'acteur banquier.

Le rôle de **l'acteur chef banquier** inclut les rôles de l'acteur banquier et l'acteur décideur. Après la collection des balances des GABs par le banquier, il va clôturer la journée. Par la suite, il génère la balance globale pour le serveur central de la banque afin d'effectuer les écritures comptables relatives au module « change de devises ». En plus, il a la possibilité de valider le taux de change saisit par le banquier. Par conséquent, ces nouveaux taux de change sont distribués aux GABs comme une mise à jour.

Aussi, il valide les paramètres de la balance des GABs et la balance globale, saisit par le banquier. Ainsi, il peut adapter le système aux écritures comptables de la banque. Le diagramme de cas d'utilisation du système associé à l'acteur chef banquier est présenté dans la figure suivante :

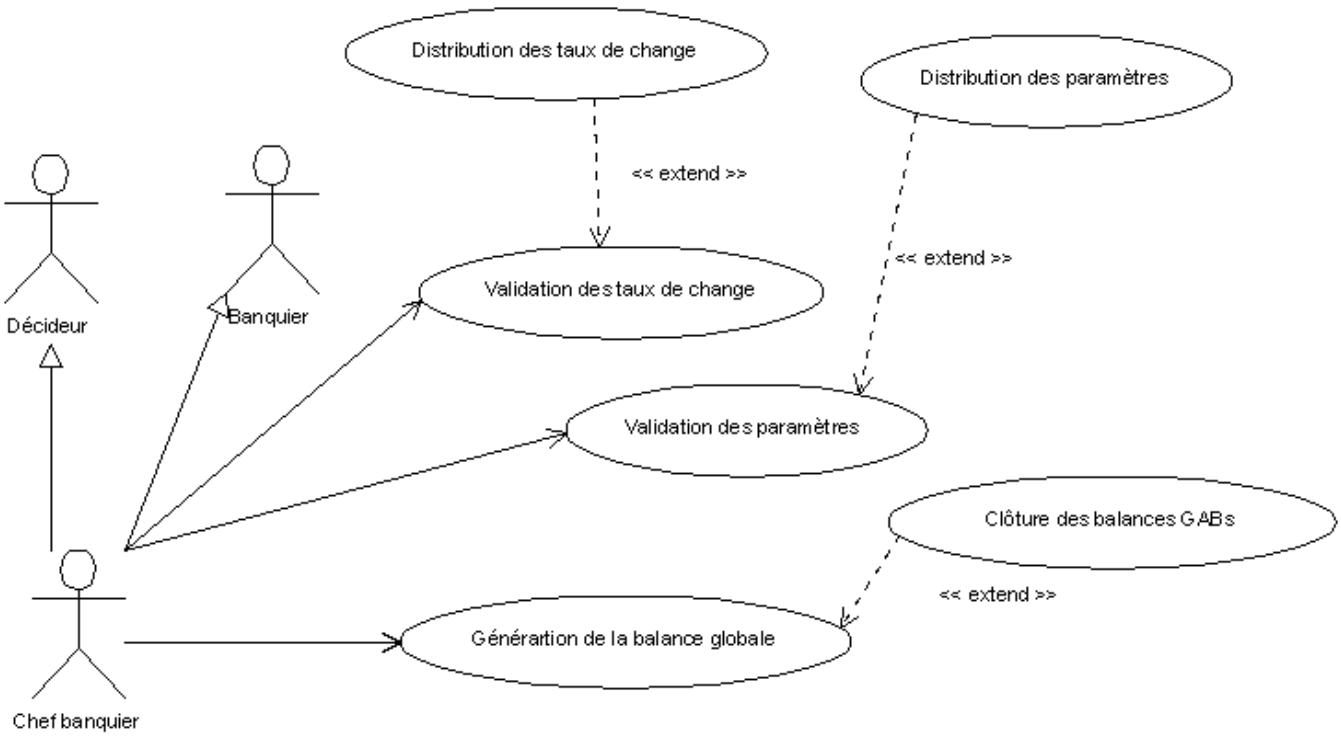


Figure 9 : diagramme de cas d'utilisation associé à l'acteur chef banquier.

L'acteur administrateur gère les ressources de l'application telles que les institutions d'une banque, ses agences, ses GABs, les devises, etc. De plus, il régit les paramètres de fonctionnement relatifs aux GABs et au fonctionnement du serveur. Encore, il alimente la base de données par les données essentielles que les autres utilisateurs n'ont pas le droit de les manipuler. Cependant, il a une vision totale de l'exécution de l'application par la consultation des événements enregistrés.

Par ailleurs, il définit la sécurité du système en attribuant des rôles et des privilèges aux autres utilisateurs. Il doit donc s'authentifier pour accomplir les tâches qui lui sont permises. Le diagramme de cas d'utilisation suivant résume les interactions avec le système associées à l'acteur administrateur :

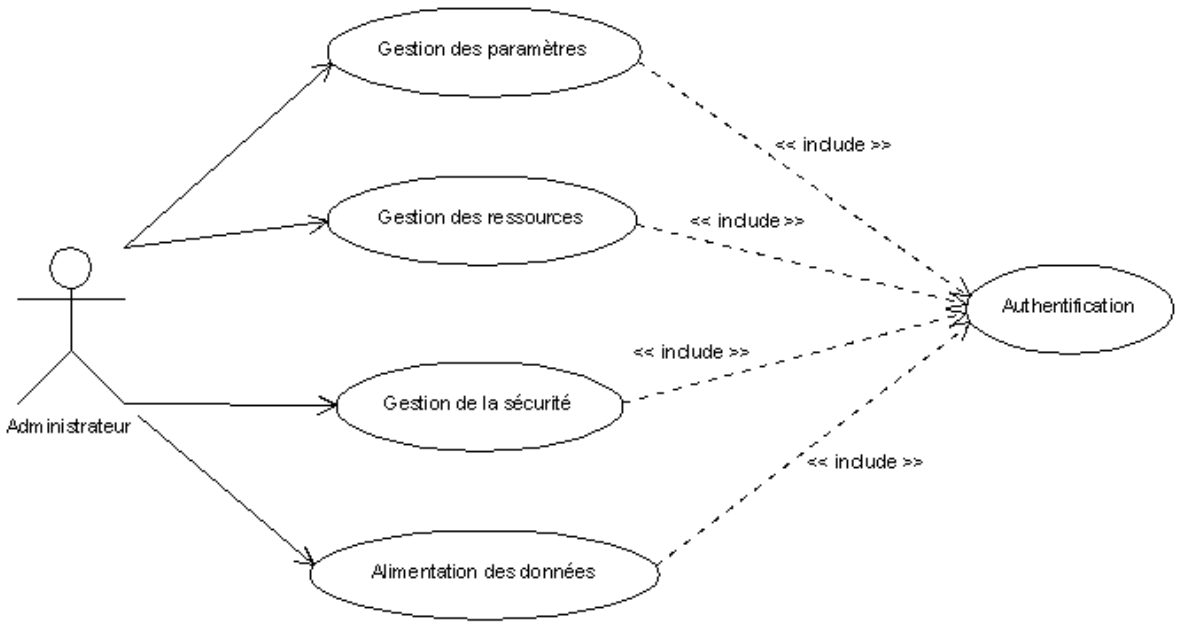


Figure 10 : diagramme de cas d'utilisation associé à l'acteur administrateur.

L'acteur guichetier qui sera chargé d'utiliser le système pour saisir la balance du GAB qui n'a pas été récupéré de façon automatique (à cause d'un crash disque par exemple).

Il doit, comme pour les autres acteurs, lui aussi s'authentifier pour accomplir les tâches qui lui sont agréées. Le diagramme de cas d'utilisation du système associé à l'acteur guichetier est présenté dans la figure suivante :

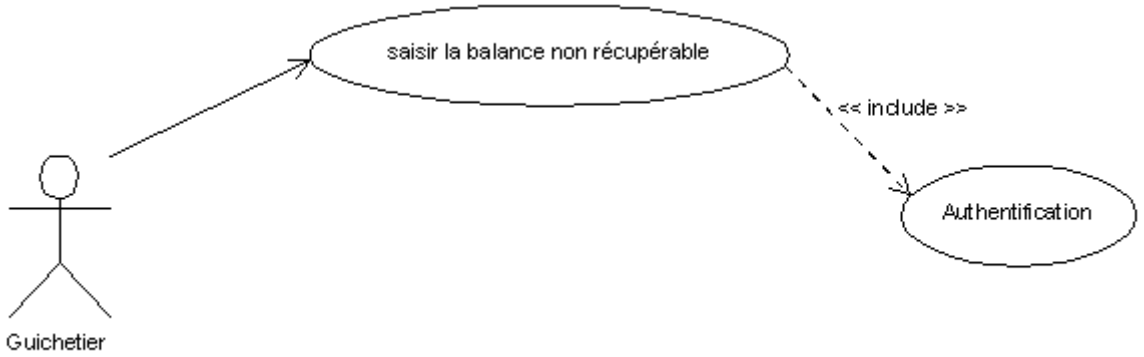


Figure 11 : diagramme de cas d'utilisation associé à l'acteur guichetier.

Les différents cas d'utilisation sont ainsi présentés. Je vais commencer, maintenant, à les raffiner et en dégager plus de détail afin d'avoir plus de visibilité utilisateur/système. L'étape analyse sera par la suite plus claire et facile à manœuvrer.

Au cours de l'interaction avec le système, un cas d'utilisation nécessite des conditions et doit aboutir à une finalité bien déterminée. De ce fait, il suit un scénario particulier et déclenche une exception s'il est hors contexte défini par les règles de gestion. L'exemple du tableau suivant montre cette description textuelle pour le cas d'utilisation mise à jour des taux de change [ROQUESVALLEE01] :

Acteur Primaire	Banquier.
But	- refléter les taux de change réels approuvés par **BANK AL MAGHRIB.**
Pré conditions	- authentification du banquier auprès du **ProChangeServer.**
Déclencheur	- après une période déterminée[5].
Scénario principal	- le banquier demande l'ajout de nouveaux taux. - **ProChangeServer** crée une nouvelle configuration contenant les dernières valeurs des taux de change. - le banquier modifie les informations relatives au nouveau jalon. - **ProChangeServer** vérifie la variation relative à chaque devise. Deux cas se présentent. Si accepté [A1], sinon [E1].
Scénarios alternatifs	A1 : - **ProChangeServer** crée une nouvelle configuration et effectue la mise à jour. - **ProChangeServer** vérifie le droit de distribution. Deux cas se présentent. Dans l'affirmatif [A2]. Dans la négation [E2].
	A2 : - la configuration est distribuée. - une confirmation est retournée au banquier.
Scénarios d'exception	E1 : **ProChangeServer** signale au banquier que la mise à jour n'est pas effectuée.
	E2 : la distribution doit être effectuée par le chef banquier.
Règles de gestion	- chaque mise à jour doit passer par la création d'une nouvelle configuration.

Figure 12 : description textuelle du cas d'utilisation mise à jour du taux de change.

La distribution textuelle, ainsi exprimée, représente toutes les informations du cas d'utilisation détaillé. Cependant, l'enchaînement n'est pas mis en relief. C'est pourquoi, il serait judicieux de compléter cette description textuelle par le diagramme de séquence relatif au cas d'utilisation étudié.

[5] Actuellement, les taux de change sont mis à jour chaque semaine.

Le diagramme de séquence, adjoint au cas d'utilisation de mise à jour des taux de change, est le suivant :

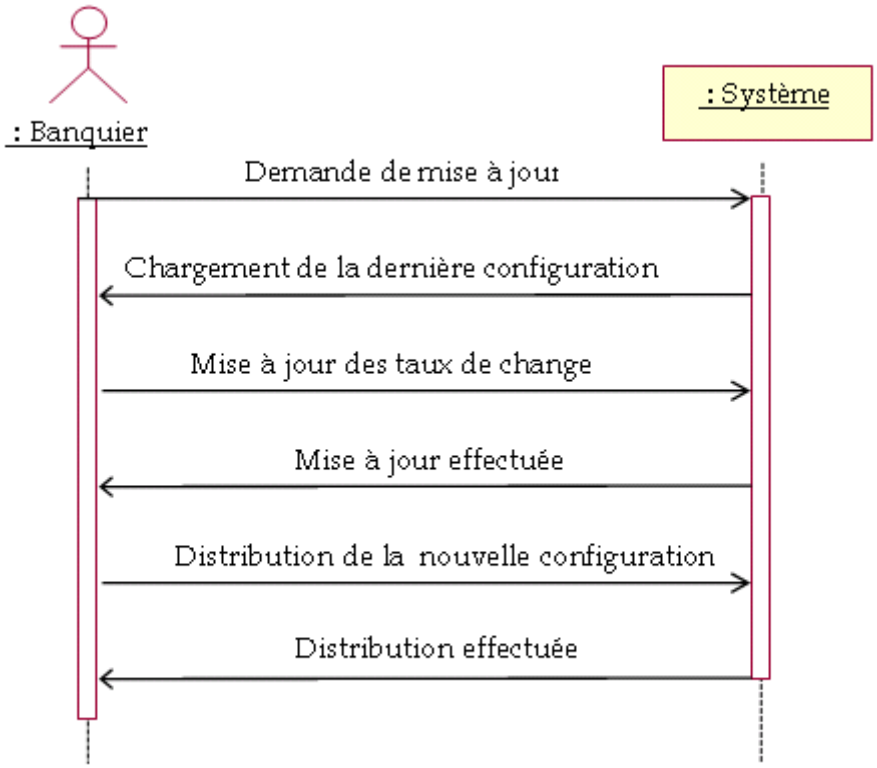


Figure 13 : diagramme de séquence du cas d'utilisation mise à jour des taux de change.

1.2. Maquette du système

Dans le but de valider les besoins fonctionnels tels qu'ils sont recensés et d'avoir une idée sur l'IHM du projet **ProChangeServer**, j'ai proposé une maquette html pour le système. En effet, il est toujours plus simple de vérifier les besoins sur des écrans que sur du texte. La figure suivante représente l'écran relatif à la gestion des taux de change :

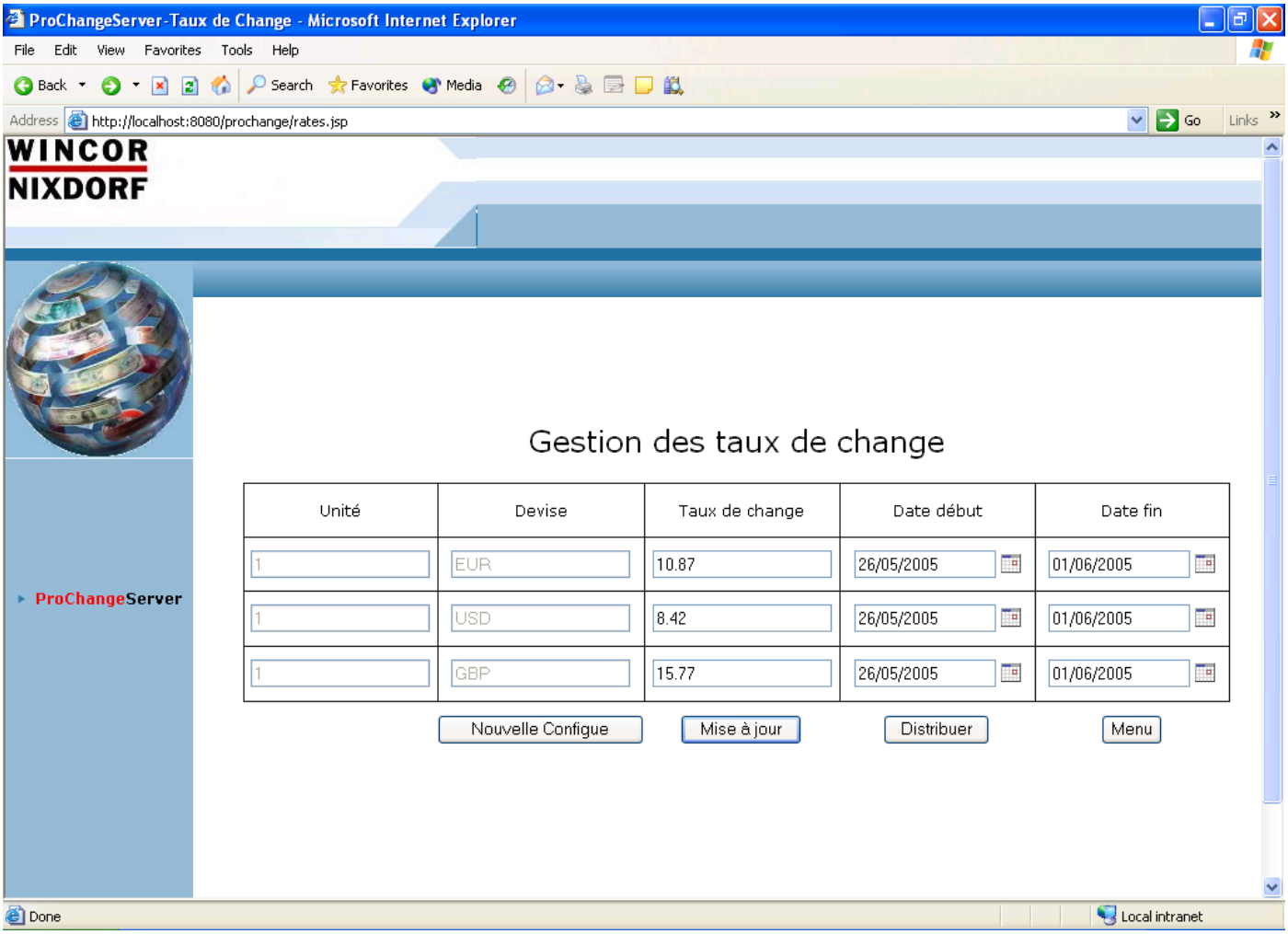


Figure 14 : maquette du système **ProChangeServer**.

A la fin de cette phase de capture des besoins fonctionnels, j'ai rédigé un ensemble de livrables :

- les diagrammes de cas d'utilisation.
- un document regroupant la description textuelle de l'ensemble des cas d'utilisation.
- la maquette HTML pour le système.

2. Analyse

La phase d'analyse comprend les activités qui permettent d'aboutir au modèle d'analyse du système en partant des cas d'utilisation et des besoins fonctionnels [JIMCONALLEN00]. Ce modèle d'analyse est constitué par les classes et les collaborations de classes qui traduisent les comportements dynamiques, détaillés

dans les cas d'utilisation et les besoins. Ces classes représentent l'aspect métier de l'application. En effet, l'analyse se concentre sur le fonctionnel en ignorant les contraintes techniques qui seront étudiées dans la partie technique. Aussi, l'essentiel dans cette phase est de s'assurer que tous les besoins fonctionnels sont réalisés dans la modélisation du système.

Pour présenter le travail effectué dans cette phase, j'ai tout d'abord détaillé les diagrammes de séquence élaborés qui ont permis de dégager la composition interne du système en terme d'objet. Par la suite, je vais présenter le modèle d'analyse construit et sa décomposition en paquetages.

Finalement, je vais étudier le comportement des entités émanées à travers les diagrammes d'état/transition.

2.1. Composition interne du système

Tout logiciel est considéré comme un système où interagissent des d'objets entre eux afin d'accomplir une finalité. Dans ce sens, le but de cette activité est d'explorer cette composition interne en terme d'objet.

Le mécanisme de UML qui permet d'exprimer cette dynamique et cette collaboration est le diagramme d'interaction. Le diagramme de séquence est un diagramme d'interaction qui traduit le flot du scénario d'un cas d'utilisation et cela en terme de classes qui doivent être implémentée.

Dans cet esprit, pour dégager les objets composants l'architecture étudiée, j'ai élaboré les diagrammes de séquence associés aux cas d'utilisation dégagés dans la phase précédente. Pour présenter le travail réalisé dans ce sens, j'ai choisi de décrire deux diagrammes de séquence qui concernent :

- la collecte des transactions d'un GAB.
- la remontée de la balance globale au serveur central de la banque.

La collecte des transactions du GAB n'est intégrée dans le système que si tous les détails, qui lui sont relatifs, sont validés. Ces détails constituent la balance du GAB

effectuant des opérations de change. Elle est remontée au serveur quotidiennement et automatiquement. Par ailleurs, elle est structurée et découpée en 5 parties complémentaires:

- récapitulatif de la journée.
- liste des transactions.
- détail des transactions par devise.
- détail des transactions par les coupures de la devise étrangère.
- détail des transactions par les coupures MAD (billets et pièces).

Au fur et à mesure de son récupération, une balance peut avoir plusieurs états : validée ou rejetée ou récupérée.

Quand la collecte de tous les GABs est effectuée, le système renvoie la balance globale à la banque centrale. Celle-ci permet de faire le point sur les activités effectuées pendant une période définie. Ainsi, elle sera envoyée vers le serveur central de la banque soit automatiquement, soit manuellement.

La périodicité de la balance globale peut coïncider avec celle des GABs, mais peut dépendre de la clôture comptable de la banque, donc il faut tenir compte des jours fériés et des week-ends. Si la banque opte pour une clôture comptable, le système gère un calendrier des journées.

L'analyse prend en compte tout ceci, afin de construire une architecture qui permet de réaliser les fonctionnalités requises. Dans ce qui suit, je vais présenter les diagrammes de séquence correspondants aux cas d'utilisation « la collecte des transactions d'un GAB » et « la remontée de la balance globale au serveur central de la banque ». Ensuite, je vais dégager la partie du modèle d'analyse qui correspond à ces diagrammes.

Le premier diagramme est associé au cas d'utilisation « collecte des transactions d'un GAB ». Je rappelle que ce scénario se déclenche automatiquement. Il est

présenté dans la figure 15. Sur ce diagramme, j'ai relevé une suite d'échanges de messages qui sont comme suit :

- à une heure paramétrée, une tâche planifiée se déclenche pour créer une nouvelle transaction.
- la transaction fait appel aux paramètres de récupération en utilisant l'objet utility. Quand ces paramètres sont disponibles, ils sont renvoyés à l'objet transaction en question.
- la transaction valide les données extraites du fichier envoyé par le GAB. Si le format des données dans ce fichier est conforme, le message de validité est renvoyé à l'objet transaction. La transaction retourne à l'objet tâche sa validité.
- L'objet tâche ajoute la transaction courante aux données récupérées à partir du GAB.

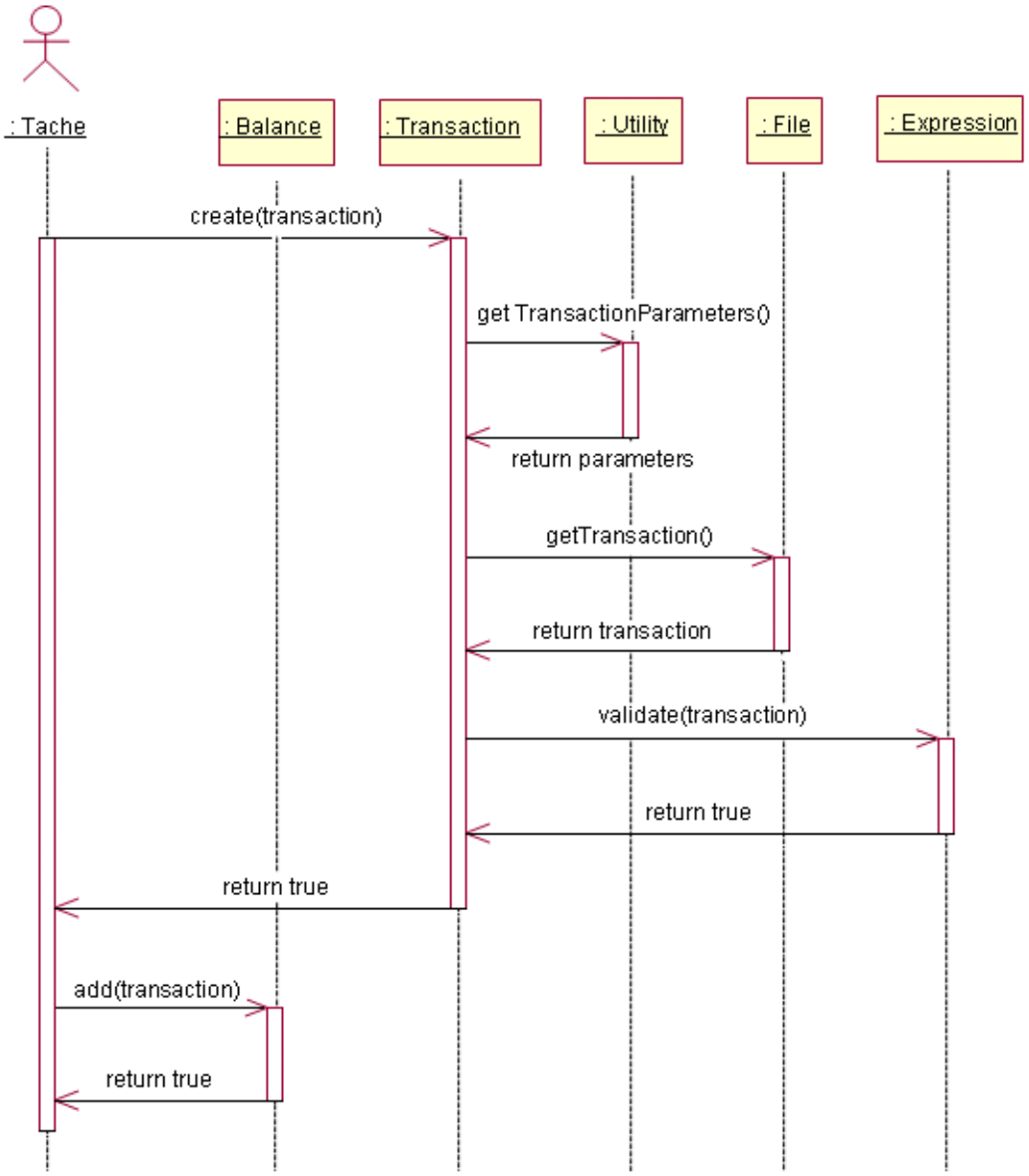


Figure 15 : diagramme de séquence de collecte des transactions d'un GAB.

Le deuxième diagramme de séquence proposé, correspond au cas d'utilisation remontée de la balance globale au serveur central de la banque.

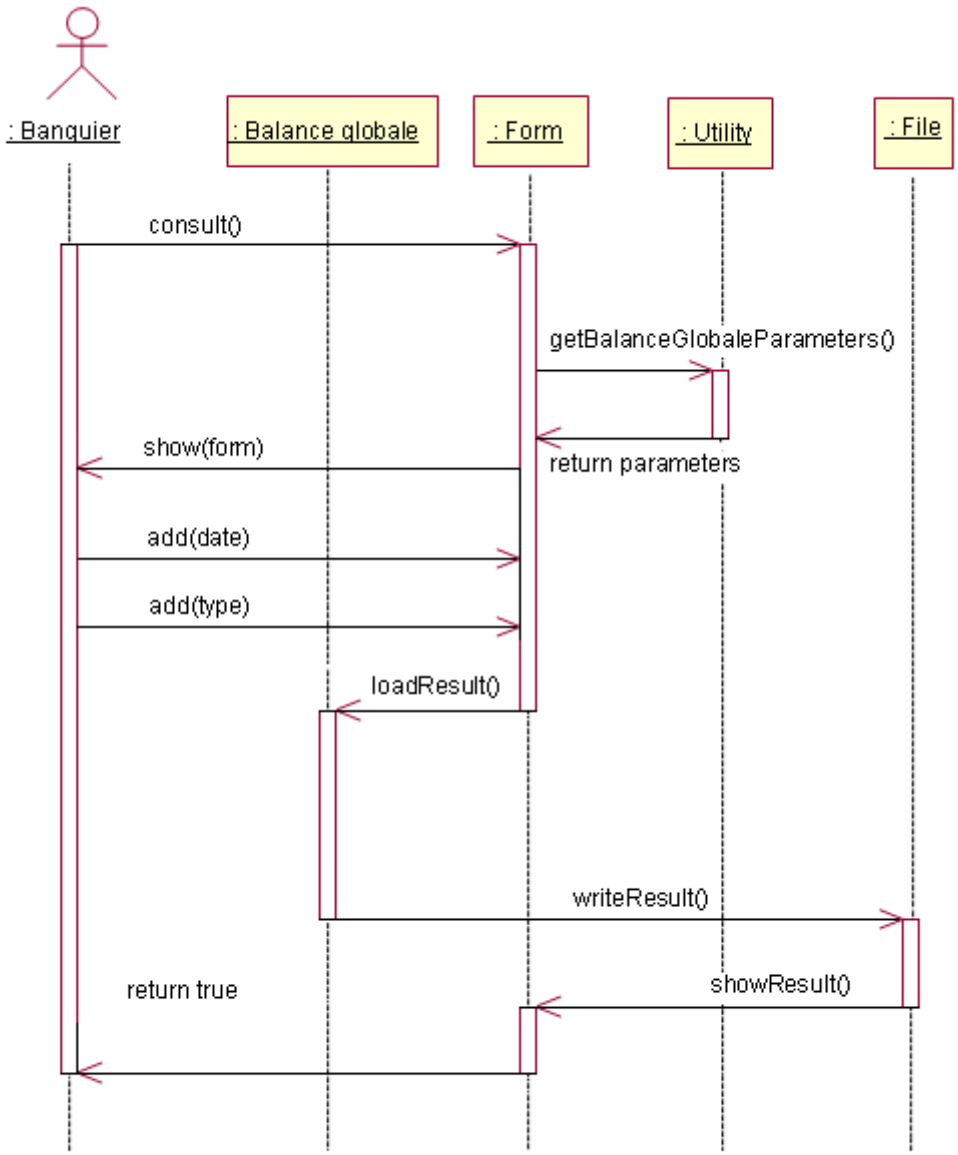


Figure 16 : diagramme de séquence de remontée de la balance globale au serveur central de la banque.

Ce diagramme illustre la remontée de la balance globale au serveur central de la banque. Il contient la séquence suivante de messages :

- le banquier consulte une forme réservée à la balance globale. Celle-ci charge les paramètres relatifs à la balance globale.

- la forme incite le banquier à sélectionner la balance désirée.

- le banquier sélectionne la date de la balance et le type de balance générale désirée. Puis, il valide son choix.

- La forme récupère ces paramètres et les remet à l'objet balance globale. Celui-ci retrouve les données voulues. Ensuite, il remet le résultat à une forme définie qui l'affiche au banquier. De plus, les données récupérées sont sauvegardées dans un fichier.

- le banquier envoie la balance globale, ainsi générée, à la banque centrale.

A partir de ces deux diagrammes de séquence et à partir des besoins exprimés dans le cahier des charges, la partie du diagramme de classe associée aux transactions est présentée dans la figure 17 :

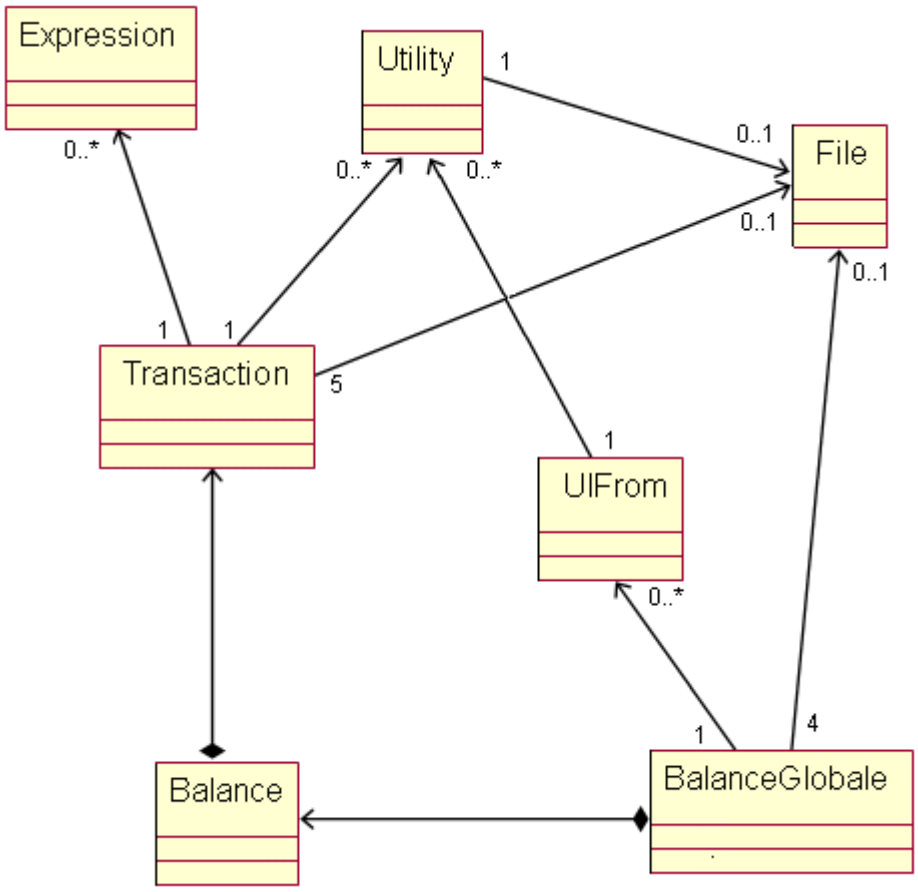


Figure 17 : partie du diagramme de classe associée aux transactions.

2.2. Construction du modèle d'analyse

A partir des diagrammes de séquence et des besoins fonctionnels, le modèle d'analyse est ainsi dégagé. En fait, ce modèle est bâti à partir des besoins et raffiné par les collaborations tirées des diagrammes de séquence.

Afin de simplifier l'analyse du système, une hiérarchie en paquetages UML s'avère la plus appropriée et la plus parlante. Ce découpage permet une meilleure représentation des parties de l'application puisqu'il est plus réduit et plus clair en explication.

La figure 18 suivante présente les principaux packages du système étudié :

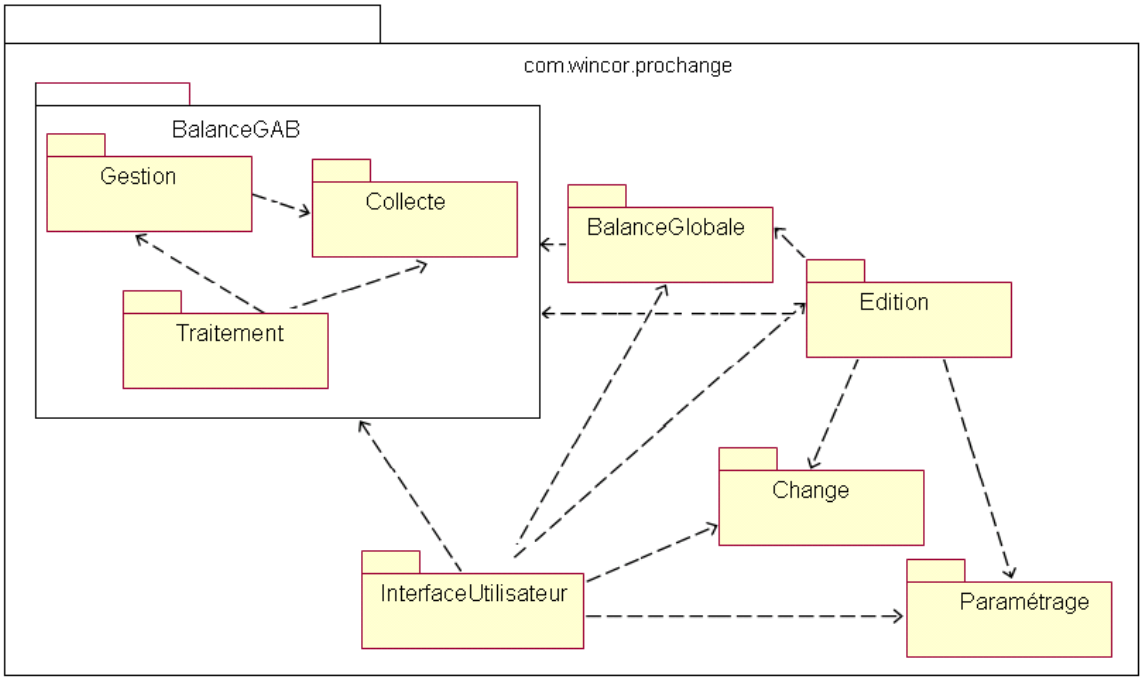


Figure 18 : diagramme de package du modèle d'analyse.

Comme les packages sont nombreux et pour faciliter le suivi du document, je vais décrire brièvement les packages analysés. Le package « **BalanceGAB »** inclut les trois packages : package « Collecte », package « Traitement » et package « Gestion ». Ils sont décrits comme suit :

Package « Collecte »

Ce package contient les classes qui permettent de réaliser les collaborations de collecte. J'ai présenté la collecte de transaction dans le diagramme de séquence (figure 15). Cependant, il y a quatre autres schémas similaires à la collecte de transaction qui sont :

- collecte du détail des transactions d'un GAB.
- collecte du détail des transactions d'un GAB par coupure étrangère.
- collecte du détail des transactions d'un GAB par la coupure locale.
- collecte du résumé de la journée de tous les GABs.

Package « traitement »

Les classes de ce package rassemblent les données des GABs ayant envoyé leurs balances quotidiennes, d'autres sont développées pour vérifier la conformité des données au format suivi et pour examiner leurs valeurs sémantiques.

Si c'est validé, des classes de sauvegarde sont ainsi appelées ; par l'insertion dans la base de données et la sauvegarde des données initialement récupérées à partir des GABs. De plus, il y a préparation de l'environnement pour la prochaine réception. Si les données reçues ne sont pas conformes, des classes pour traitement des anomalies sont ainsi appelées. Ces classes ont pour tâches :

- traitement des balances rejetées.
- traitement des balances non envoyées.
- traitement des balances récupérées.
- traitement des balances non récupérées.

Package « Gestion »

Pour chaque GAB n'ayant pas transmis sa balance ou elle est mal reçue, l'administrateur système ou le superviseur des GABs doit essayer de récupérer cette

balance. Dans le cas ou ce n'est pas possible de récupérer la balance d'un GAB donné à cause d'un crash disque, des classes de gestion sont appelées pour permettre de saisir manuellement toutes les transactions de change effectuées par le GAB en défaut.

Package « BalanceGlobale »

Ce package permet de remonter une balance globale pour une banque donnée comme il permet de la scinder par institution ou par agence. De plus, il informe le serveur central de la banque du chiffre d'affaire en dirham réalisé ainsi que le total des commissions. Il remonte aussi par devise le montant total encaissé. Ce package se base essentiellement sur le package « BalanceGAB ».

Package « Edition »

Ce package permet la consultation des données des transactions effectuées. En effet, il va générer des rapports relatifs aux principales informations utilisées par la banque. Un prototype générique sera exploité par toutes les requêtes d'édition. Pour enrichir l'utilisation de l'édition, ce package offre la possibilité de sauvegarde des données dans des fichiers sous une multitude de format : XML, HTML, PDF et EXCEL.

Package « Parametrage »

Ce package permet l'adaptation de l'application selon l'environnement d'exécution définit par un utilisateur privilégié. De plus cette fonctionnalité offre une grande souplesse dans la phase de déploiement et permet la sélection des services optionnels de l'application et la gestion de ses ressources.

Package « Change »

Ce package englobe les fonctionnalités nécessaires pour la gestion des taux de change, leur récupération et leur distribution.

Package « InterfaceUtilisateur »

Ce package englobe les fonctionnalités nécessaires pour la gestion de l'interface des utilisateurs.

2.3. Analyse du comportement des entités dégagées

Dans le but d'analyser le comportement des classes du modèle étudié, susceptibles d'avoir un comportement dynamique riche, ou d'avoir plusieurs états, des diagrammes d'état/transition sont ainsi construit.

Un diagramme d'état/transition spécifie la séquence d'états qu'un objet peut avoir durant sa vie en réponse aux événements qui lui adviennent, ainsi que les réactions correspondantes [ROQUESVALLEE01].

Toutes les classes du modèle d'analyse ne requièrent pas nécessairement un diagramme d'état/transition. Il s'agit donc de trouver celles qui ont un comportement dynamique complexe nécessitant une description poussée. Dans cette analyse, plusieurs classes requièrent un diagramme d'état/transition. Toutefois, je ne vais présenter qu'un seul concernant la « BalanceGAB ». D'après l'étude du cahier des charges, j'ai pu élaborer le diagramme d'état/transition suivant :

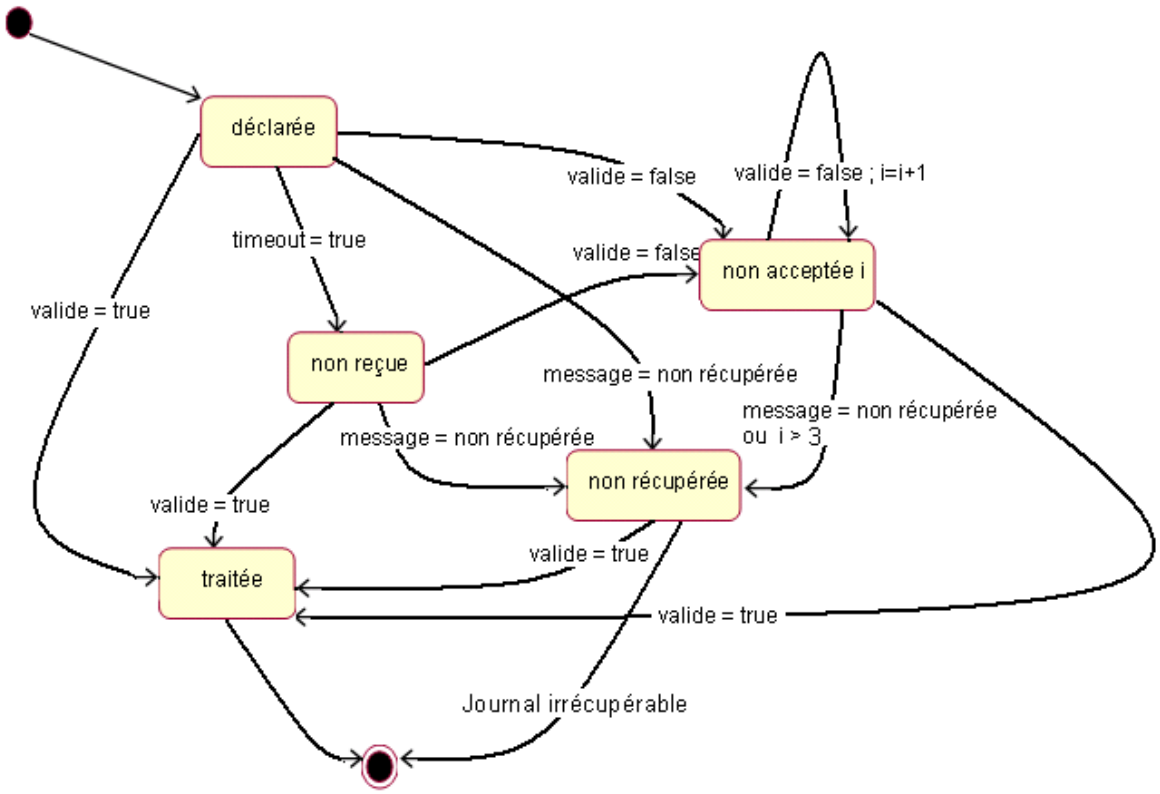


Figure 19 : diagramme état/transition de l'objet « BalanceGAB ».

Dans cette figure, une « BalanceGAB » étant déclarée pour chaque GAB actif dans une journée. A la fin de la journée, le processus de collecte se déclenche et agit sur l'objet « BalanceGAB ». Si la balance est valide, elle est positionnée pour être « traitée ». Sinon, elle est mise « non acceptée ». De plus, si elle est non reçue pendant un temps défini, elle est positionnée comme telle. Cependant, elle est connue « non récupérée » si un message venant du guichetier le confirmant ainsi.

Dans l'état non reçu, si elle est valide, elle est déclarée « traitée ». Sinon, elle est non acceptée ou non récupérée. Dans l'état non accepté, un compteur est incrémenté chaque fois que la balance reçue est erronée. Or, si elle dépasse un nombre paramétré ou un message le confirmant, elle sera définit comme étant non récupérée. Finalement, si elle est valide, elle est déclarée « traitée ».

Conclusion

Dans ce chapitre, j'ai détaillé l'étude fonctionnelle du projet. Elle est composée de deux phases : la capture des besoins fonctionnels et l'analyse. La phase de capture des besoins, consistait en la modélisation des utilisations du système et la

description textuelle de ces utilisations. La phase d'analyse avait pour objectif la modélisation du système par une collaboration d'objets qui permet de réaliser les fonctionnalités attendues du système.

Dans le chapitre suivant, je vais présenter l'étude technique qui met en relief l'architecture logicielle mise en place et les différentes solutions technologiques adoptées pour construire le modèle final de conception.

Chapitre 4 : Étude technique du projet

*Dans ce chapitre, je vais présenter la technologie J2EE adoptée pour **ProChangeServer.** De plus, je vais expliquer l'architecture logicielle caractérisant l'application. Ensuite, je vais exposer les Frameworks Open Source choisi pour le développement.*

1. Technologie J2EE

La technologie Java 2 Entreprise Edition (J2EE) définit la norme de développement des applications d'entreprise distribuées et multi tiers. Elle permet de simplifier les applications d'entreprise en se basant sur des composants modulaires normalisés. Ceci étant, en fournissant un ensemble complet de services et en gérant automatiquement un grand nombre de détails sur le comportement des applications. J2EE est composée de plusieurs technologies qui sont regroupées en trois catégories : composant, service et communication [wwwJAVASUN].

Technologies de type composant

Les technologies de type composant sont utilisées par les développeurs afin de créer des composants logiciels d'une application d'entreprise notamment l'interface utilisateur et la logique métier. Elles permettent de développer des modules réutilisables qui peuvent être intégrés dans d'autres applications. De plus, elles sont supportées par les services d'infrastructures de la plate-forme J2EE qui simplifie le développement des applications. Elles permettent, aussi, d'adapter les composants aux besoins des ressources de l'environnement dans lequel ils sont déployés [wwwJAVASUN].

Technologies de type service

Etant donné que la plupart des applications d'entreprise ont besoin d'accéder aux systèmes d'information d'entreprise, la plate-forme J2EE offre un ensemble de services tels que l'accès aux bases de données (JDBC), le service de désignation (JNDI), les transactions et les services de Messagerie.

Technologies de type communication

La plate-forme J2EE fournit un ensemble de technologies, permettant la communication entre clients et serveurs et entre les objets qui collaborent dans les différents serveurs. Comme l'exemple des connecteurs aux systèmes d'information existants et l'invocation des méthodes distantes à travers le standard OMG-CORBA.

1.1. Architecture J2EE

C'est une architecture soutenue par SUN® MICROSYSTEMS qui définit et fournit le modèle de programmation multi tiers basé sur l'architecture des composants. Avec ce modèle, des applications légères peuvent interagir facilement avec le système d'information et les composants implantant la logique d'entreprise sur des serveurs d'applications. L'architecture J2EE consiste en trois parties :

- components : les composants qui prennent en charge la présentation et la logique métier.
- containers : fournit un contexte aux composants.
- Connectors: fournit l'accès aux sources de données de l'entreprise.

Cet environnement supporte trois types de composants d'applications, dont les définitions seront présentées par la suite :

- applets.
- servlets (incluant JSP).
- Enterprise JavaBeans (EJB).

La figure 20 illustre les trois parties constituant l'architecture J2EE. Le noyau de serveur « J2EE Server Core » est un environnement de serveur d'application qui fournit les services de gestion de ressources et de transactions [wwwTHOMAS].

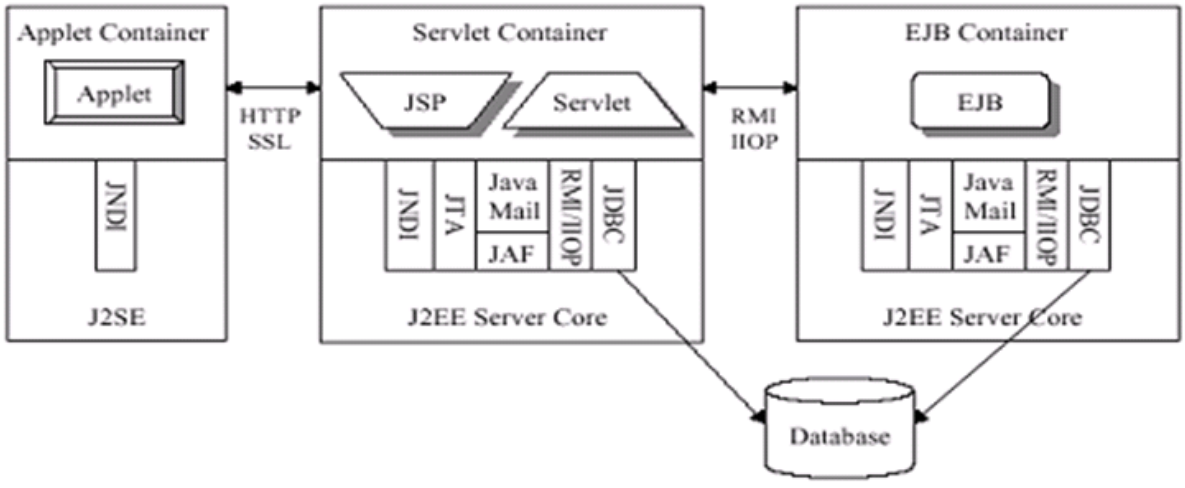


Figure 20 : architecture J2EE.

1.2. Composants de base de J2EE

Comme c'est mentionné, l'architecture J2EE est composée de plusieurs technologies qui sont regroupées en trois catégories : composant, service et communication. Détaillons ces composants de base de J2EE.

Entreprise Java Beans

C'est un modèle de composant logiciel réutilisable spécifié par SUN® MICROSYSTEMS en collaboration avec des industriels. Il est destiné au développement et au déploiement d'applications établies en fonction du modèle multi tiers. Il permet aux concepteurs d'applications de bénéficier des avantages de Java et notamment de l'indépendance vis à vis de la plate forme.

L'idée est de programmer une seule fois de petits composants logiciels, de les utiliser et les réutiliser n'importe où [wwwREDBOOKIBM]. La spécification des EJB définit deux types de composants : les Session Beans, qui ne vivent que le temps de leurs activités et qui sont en permanence en interaction avec l'utilisateur. Par contre, les Entity Beans, sont persistants et représentent des données issues d'une base de données. Les Session Beans sont de deux types :

- Stateless Session Beans: sont des objets qui se comportent comme des composants serveurs et exécutent des actions à distance. Ces objets ne gardent aucune notion d'état entre deux invocations [PATZER00]. Ils reçoivent des requêtes provenant de différents clients et l'ordre des requêtes n'influe pas sur leur exécution.

- Statefull Session Beans: reprennent les fonctionnalités d'invocation à distance des Stateless Session Beans, mais ajoutent la capacité de conserver un état propre à chaque client, d'une invocation à une autre.

Les Entity Beans sont de deux types :

- Container Managed Persistance (CMP) : la persistance de l'objet est gérée automatiquement par le serveur.

- Bean Managed Persistance (BMP) : la persistance de l'objet est gérée manuellement par le bean lui-même. Ce deuxième type des Entity Beans demande un grand effort du développement de la part du programmeur vu qu'il doit contrôler la persistance de l'objet [wwwREDBOOKIBM].

Servlets

Une Servlet Java est un programme Java qui se branche sur un serveur Web. Il est possible d'étendre un serveur Web pour qu'il héberge des Servlets par le biais d'un moteur de Servlets. La technologie des Servlets s'avère en fait plus adaptée à répondre aux requêtes de l'utilisateur en invoquant des méthodes sur des EJB, puis en redirigeant la requête HTTP vers des pages HTML ou JSP qui afficheront le résultat de la requête. Les Servlets sont donc utilisées principalement comme un «centre d'aiguillage» qui interprète et dirige le flot de requêtes de l'utilisateur vers les composants EJB et renvoie à l'utilisateur les pages Web adéquates [IBMBOOK].

Java Server Pages

Les JSP permettent de séparer efficacement le codage HTML de la logique applicative des pages Web. Ils sont utilisés pour accéder à des composants réutilisables, tels que des Servlets, des JavaBeans et des applications compatibles avec le langage Java [IBMBOOK].

Après cette présentation de la plate-forme J2EE, je vais présenter l'architecture logicielle du système étudié, ainsi que les contraintes qu'elle subit, en passant par une définition de la notion d'architecture logicielle.

2. Architecture logicielle du système

Une architecture logicielle exprime un schéma d'organisation structurel fondamental pour les systèmes logiciels. Elle fournit un jeu de sous-systèmes prédéfinis, spécifie leurs responsabilités, et elle inclut les rôles et les instructions générales pour organiser les relations entre eux [wwwASTONS]. Ainsi, sans une bonne architecture, un système d'information est difficile à comprendre, prédire, gérer et optimiser. Les buts d'une architecture sont :

- la compréhension du système étudié en définissant ses limites.
- la gestion de la croissance du système.

2.1. Exigences de ProChangeServer

L'architecture logicielle de **ProChangeServer** répond à un certain nombre de buts et contraintes en terme de sécurité, de disponibilité et de maintenance. Le choix d'une architecture logicielle constitue une étape technologique primordiale. En effet, il faut définir les grands objectifs techniques de la future architecture, c'est-à-dire de bien prendre en compte l'ensemble des forces qui vont s'exercer sur le futur système ainsi que la puissance de chacune d'entre elles [wwwASTONA].

La figure suivante présente les objectifs techniques de l'architecture logicielle du **ProChangeServer** :

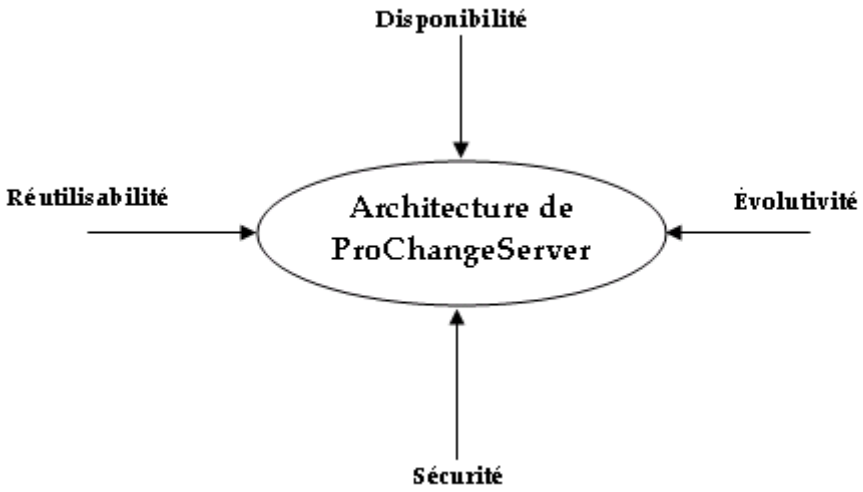


Figure 21 : objectifs techniques de l'architecture logicielle du **ProChangeServer**.

Le tableau suivant fournit plus de détail concernant les objectifs techniques du **ProChangeServer**:

Objectif	Description
Disponibilité	Il doit être en permanence à la disposition de ses utilisateurs.
Sécurité	**ProChangeServer** doit assurer un système de sécurité flexible permettant d'accéder aux fonctionnalités du système selon les droits de l'utilisateur, en plus de la gestion des autorisations et des authentifications.
Evolutivité	Il doit permettre d'étendre le système.
Réutilisation	**ProChangeServer** inclure la possibilité de réutiliser les modules du système.

Figure 22 : objectifs techniques de l'architecture logicielle du **ProChangeServer**.

2.2. Architecture logicielle de ProChangeServer

Parmi les différentes façons de structurer une architecture, la mieux comprise et maîtrisée en informatique est l'approche par couches [wwwASTONA]. Une couche (*Layer* en anglais) est une division logique, horizontale d'un système qui fournit une abstraction particulière du système aux couches supérieures [wwwASTONS]. Chaque couche possède des responsabilités spécifiques. Dans une structuration par couches, les couches inférieures prennent en charge des rôles qui offrent un fonctionnement de base pour les couches supérieures, permettant par la suite d'abstraire l'implémentation de ces services basiques.

Ainsi, j'ai adopté un découpage en couches. Une telle architecture permet également d'obtenir un niveau poussé de réutilisation en adoptant les solutions trouvées aux problèmes ayant un fonctionnement similaire. Cette exploitation est prise en compte pour chaque couche de l'architecture. La figure 23 suivante résume le découpage adopté :

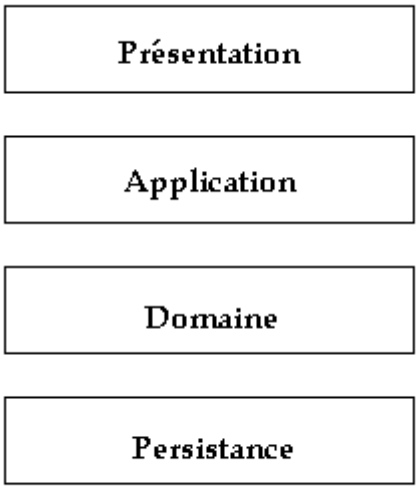


Figure 23 : architecture de **ProChangeServer** en couches.

Couche Présentation

Cette couche est adoptée principalement pour gérer l'aspect visuel des applications et pour gérer les interactions avec les utilisateurs. Chargée de dessiner les fenêtres et les autres composants graphiques, elle intercepte les événements et appelle la couche Application. De plus, elle vérifie les autorisations des utilisateurs auprès du service de sécurité. Les rôles de la couche Présentation sont comme suit :

- gestion du domaine de l'écran.
- affichage des pages Web.
- gestion de l'interaction avec l'utilisateur.
- validation syntaxique.

La couche Présentation établit deux relations avec la couche Application :

- demandes de contenu à la couche Application pour affichage et édition.
- ordres de traitement à la couche Application pour création, mise à jour, suppression, etc.

Couche Application

Son but principal est de fournir des services spécifiques à la couche Présentation. Ces services correspondent aux règles du métier définies lors de la phase d'analyse. Elle prend en charge les aspects de contrôle des cas d'utilisation. C'est concrètement dans cette couche que l'on voit les cas d'utilisation issus de la partie analyse. Les rôles de la couche Application sont les suivants :

- services spécifiques au domaine.
- services externes.
- contrôle des cas d'utilisation.

Couche Domaine

Cette couche est concentrée sur le métier de l'application, c'est-à-dire les règles métiers, sémantiques et logiques. C'est l'espace où réside le modèle du domaine. Sa charge principale consiste à garantir la validation sémantique de l'information métier. Les rôles de la couche Domaine sont résumés ainsi :

- comportement métier.
- validation sémantique.

Elle échange l'état des composants entités métiers avec la couche Persistance.

Couche Persistance

Cette couche est certainement l'une des plus importantes. C'est ici que l'on trouve les fonctionnalités de base qui permettent de créer, rechercher et supprimer des entités métier dans le respect des propriétés transactionnelles. C'est également dans cette couche que les mécanismes de conversion objet/relationnel peuvent prendre place. Le rôle de la couche Persistance est défini comme suit :

- assurer les services basiques de création, lecture, mise à jour et suppression.
- permettre la conversion Objet/Relationnel.

Ainsi, l'architecture choisie est basée sur la logique des couches. Chaque couche fournit aux autres couches des services et utilise les services des autres. Dans ce qui suit, je vais présenter les Frameworks utilisés au cours du développement.

3. Notion de Framework

Un Framework capitalise l'expertise nécessaire en matière de programmation pour résoudre une certaine classe de problème. Les programmeurs réutilisent les Frameworks pour profiter de cette expertise sans avoir à résoudre le problème. Un Framework aide les développeurs à fournir des solutions pour un problème et à mieux les maintenir. Il fournit une infrastructure bien conçue et bien pensée. Ainsi, lorsque des nouvelles parties sont créées, elles peuvent être ajoutées avec un impact minimal sur les autres parties du Framework.

Un Framework sert, non seulement, à réaliser une application plus rapidement, mais permet d'obtenir des applications de structures semblables. Ceci permet de simplifier considérablement leur maintenance [GHJV95]. Un Framework possède les caractéristiques suivantes :

- un Framework est composé de multiple classes ou composants chacun d'eux propose une abstraction d'un concept particulier.
- le Framework définit comment ces abstractions collaborent ensemble pour résoudre un problème.

Un bon Framework doit fournir un comportement générique qui peut être utilisé par plusieurs types d'applications. La notion de Framework est généralement liée à l'Open Source. En effet, c'est un logiciel livré avec son code source et offre un droit d'utilisation, de copie et de modification. Les produits Open Source sont régis par des licences. La plus célèbre entre elles est la GPL (General Public License) de la fondation GNU [wwwFSF]. Le choix d'utiliser les Frameworks Open Source est justifié par plusieurs raisons :

La première est sans doute financière, les Frameworks Open Source sont gratuits. La deuxième est technique ; le code source des Frameworks et des logiciels Open

Source étant disponible : un développeur peut comprendre le fonctionnement des Frameworks pour corriger ses bugs ou encore procéder à des évolutions.

Le fait qu'un Framework soit Open Source ne signifie pas qu'il soit de mauvaise qualité, au contraire beaucoup de meneurs des projets Open Source sont des personnes très compétentes qui s'investissent par défi technique et pour la reconnaissance de leurs pairs.

D'autres arguments plus marketing laissent entendre que les Frameworks Open Source sont plus stables et que le respect des standards et la qualité du support sont meilleurs. Ces avantages ont cependant un prix, la documentation associée aux projets Open Source est souvent très succincte, donc lors du choix d'une solution Open Source il faut tenir compte des critères suivants:

- niveau d'intégration des standards.
- maturité.
- facilité d'utilisation.
- documentation.

4. Frameworks utilisés

L'utilisation des Frameworks est indispensable pour le développement réussi d'un logiciel. Par la suite, je vais présenter les différents Frameworks Open Source utilisés [JIMCONALLEN00].

4.1. Framework Struts

Les applications Web sont devenues de plus en plus courantes dans les entreprises. Néanmoins, leur développement reste une pratique difficile dès que le projet devient un peu ambitieux. Le modèle MVC permet de séparer au maximum les modules (IHM et métier) afin de faciliter la maintenance des applications.

Le paradigme MVC est un schéma de programmation qui propose l'organisation de l'application en 3 parties :

- le modèle qui contient la logique et l'état de l'application.
- la vue qui représente l'interface utilisateur.
- le contrôleur qui gère la synchronisation entre la vue et le modèle. Le contrôleur réagit aux actions de l'utilisateur en effectuant les actions nécessaires sur le modèle et surveille les modifications du modèle et informe la vue des mises à jour nécessaires.

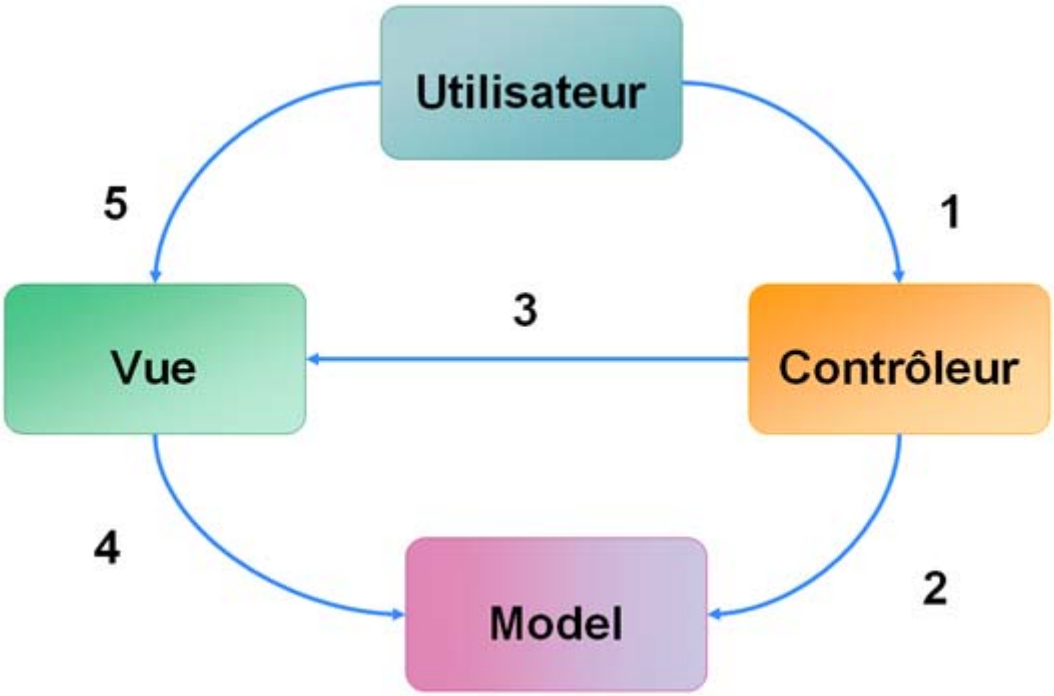


Figure 24 : modèle MVC.

La figure 24 décrit le traitement d'une requête par le modèle MVC :

1. l'utilisateur manipule l'interface homme/machine. Un événement est envoyé. Cet événement est récupéré par le contrôleur.
2. le contrôleur effectue l'action demandée par l'utilisateur en appelant les méthodes nécessaires sur le modèle.
3. le contrôleur informe la vue d'un changement d'état du modèle.
4. la vue interroge le modèle afin de connaître son état.
5. l'utilisateur voit le résultat de son action.

Une telle séparation favorise le développement et la maintenance des applications :

- le modèle étant séparé des autres composants, il est développé indépendamment.
- le modèle n'est pas lié à une interface, il peut donc être réutilisé.
- diminution de la duplication du code.
- centralisation du contrôle.
- augmentation de la maintenance du code.

Struts est un projet Open Source développé par la communauté Apache. Il a débuté en mai 2000 sous la direction de Craig Mc Clanahan, qui participe également au développement du serveur Tomcat.

Struts est un projet très actif qui fournit le Framework MVC comprenant les composants suivants :

- un contrôleur facilement configurable permettant d'associer des actions (méthode d'un objet Java) à des requêtes http (Servlet).
- des librairies de tags spécifiques pour créer facilement une vue (JSP).
- des utilitaires permettant de remplir automatiquement des champs et d'internationaliser les applications.
- la validation syntaxique des données et la gestion des callbacks des pages Web.

4.2. Framework Hibernate

Si le modèle objet offre des capacités de modélisation et d'abstraction avancées, il ne couvre pas la persistance des objets. Ainsi, la persistance n'est pas gérée automatiquement dans le modèle objet.

Par contraste, dans le modèle relationnel, les données sont persistantes : une fois que la structure du schéma relationnel est définie, les données qui y sont ajoutées restent accessibles durablement, tant qu'elles ne sont pas explicitement supprimées. La

durée de vie des données est totalement indépendante du cycle de vie de l'application.

Pour apporter ce caractère de durabilité des données au modèle objet, il convient donc de trouver un moyen de prolonger l'existence des objets au-delà de l'existence d'une session applicative. Le souci de la persistance objet consiste donc à élaborer et à mettre en œuvre les mécanismes permettant de sauvegarder, de façon fiable et durable, l'état des objets manipulés par une application.

J'ai adopté la solution du mapping objet/relationnel. Puisque la base de données relationnelle est la technologie la plus utilisée pour le stockage des données. De plus, les outils de mapping objet/relationnel permettent de réduire le temps de développement du code d'accès à la base de données.

Hibernate est un projet open source. Son rôle est de permettre d'accéder d'une manière objet aux données stockées dans une base de données relationnelle. Les principales fonctionnalités de Hibernate sont :

- l'API permettant de créer, modifier, récupérer des données dans une base de données relationnelles possédant un driver JDBC associé.
- la persistance transparente : pas d'intrusion dans le code Java pour rendre une classe persistante.
- l'assignation automatique des valeurs de clés étrangères.
- le mapping O/R défini dans un fichier XML.

4.3. Framework Log4j

La gestion des messages et des traces, permet le paramétrage de la manière dont l'application s'exécute. C'est pour faciliter la maintenance et de permettre une plus grande personnalisation (par exemple : les libellés définis par langue). Pour la gestion des traces, j'ai utilisé le Framework Open Source Log4j dont es fonctionnalités principales de Log4j sont :

- la configuration des paramètres de trace des applications se fait à travers un fichier texte.

- les paramètres de trace peuvent être changés au moment de l'exécution.
- les messages et les traces peuvent être redirigés vers plusieurs sorties : fichier, écran, etc.

4.4. Framework JasperReports

J'ai choisi le Framework JasperReports pour le développement d'éditions. JasperReports est un Framework Java Open source pour l'édition. Il a la possibilité de produire un contenu riche à travers l'écran, l'imprimante ou à travers les fichiers PDF, HTML, XLS, CSV et XML. Cet outil permet :

- le formatage dynamique des éléments du document.
- la programmation en java.
- l'accès dynamique à la base de données.
- l'utilisation des fichiers de style pour la présentation.
- la possibilité d'utiliser les couleurs.
- l'insertion des images dynamiquement.
- la séparation du modèle de développement (la vue et les données).

4.5. API JAAS

Il s'agit d'une API permettant l'implémentation de l'authentification et l'accès sécurisé dans les applets et les applications. Le modèle de permissions prend en compte l'origine physique d'une classe (pour un dossier ou une URL par exemple) mais aussi l'origine logique (grâce à la signature numérique de l'entreprise qui a développé la classe).

JAAS permet aussi de compléter un modèle de permissions différent en proposant des fonctionnalités additionnelles.

Conclusion

Ce chapitre a été consacré à l'étude technique du système. En effet, j'ai présenté la technologie utilisée ainsi que l'architecture logicielle de l'application. Ensuite, j'ai détaillé les Frameworks techniques utilisés dans le développement du serveur **ProChangeServer.**

Le chapitre qui suit, sera consacré à la conception de **ProChangeServer**.

Chapitre 5 : Conception du projet

Dans ce chapitre, je vais présenter la phase de conception. Je vais reprendre le modèle d'analyse selon les Frameworks techniques et l'architecture adoptés. Puis je vais exposer un exemple de factorisation de traitement. Ensuite, je vais présenter quelques exemples des solutions adoptées.

1. Reprise des scénarios

A cette étape, je vais reprendre le modèle d'analyse et le raffiner selon les spécificités de l'architecture adoptée et les solutions technologiques choisis. Cette reprise des schémas est réalisée en réadaptant les scénarios qu'offre l'application selon l'architecture dégagée et les Frameworks conservés.

Ces scénarios incluent, dans leurs logiques d'implémentation, les solutions techniques qu'offre chaque Framework utilisé et les dénouements que propose l'abstraction en couche. Dans ce sens, de nouvelles collaborations surgissent afin de répondre à cette logique.

La figure25 suivante présente le diagramme de collaboration relatif au cas d'utilisation collecte d'une transaction d'un GAB:

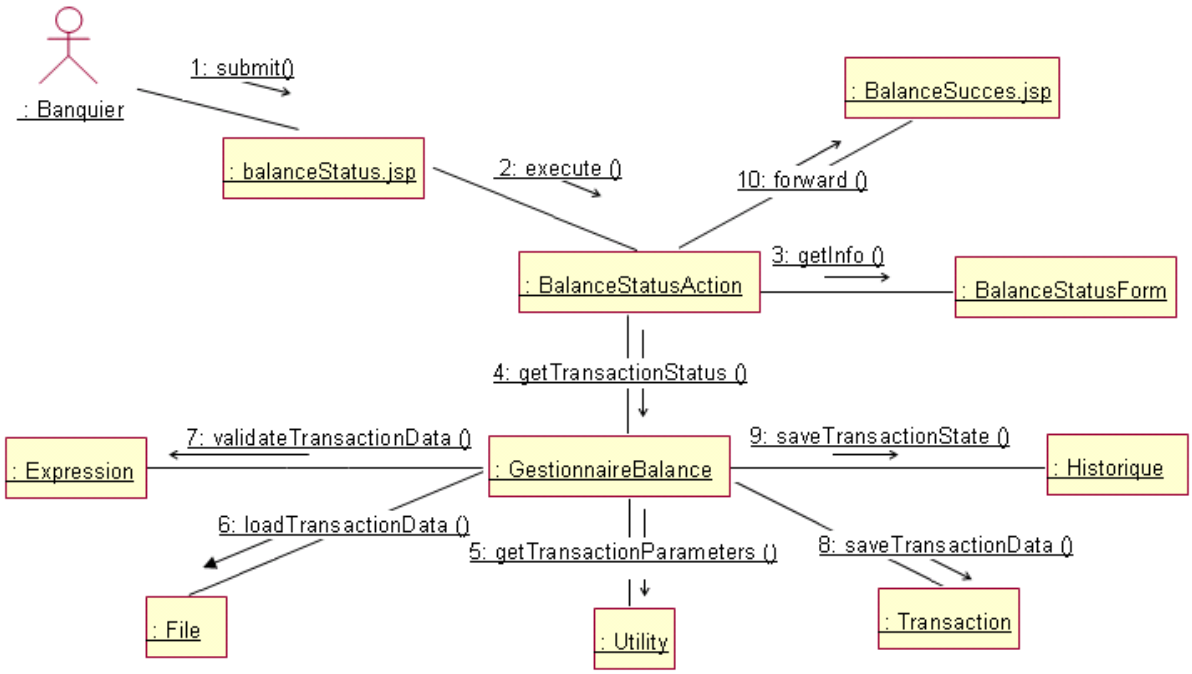


Figure 25 : diagramme de collaboration relatif à la collecte de transaction d'un GAB.

Sur ce diagramme de collaboration de nouvelles entités apparaissent. Ces entités sont dues à l'effet du Framework Struts qui propose, comme décrit précédemment, un cadre pour la gestion de l'interaction du système avec les utilisateurs.

Nécessairement, il y a la classe « BalanceStatusForm » qui hérite de la classe « Form » de Struts et qui comporte les informations manipulées par l'utilisateur. Ces informations sont récupérées au fur et à mesure du traitement de l'utilisateur pour construire des entités.

Quant à la classe « BalanceStatusAction » qui hérite de la classe « Action » de Struts, elle se charge d'appeler ses entités pour en prendre leurs valeurs. Quand une mise à jour est effectuée, elle redirige l'affichage de la page Web pour prendre en compte les nouvelles modifications.

Pour accomplir son rôle, la classe « BalanceStatusAction » fait appel à l'entité « GestionaireBalance » responsable du traitement des balances. Celle-ci se situe dans la couche Application de l'architecture adoptée. En effet, sa responsabilité est de gérer les objets du métier et leur appliquer les traitements nécessaires afin de réaliser la fonction désirée par l'utilisateur.

Le détail du diagramme de collaboration de la figure25 permet de bien expliquer le rôle des nouvelles entités. Ainsi, la séquence des échanges de messages du diagramme est la suivante :

- l'utilisateur consulte l'état de la balance quotidienne et demande l'intégration des nouvelles balances.
- le système redirige l'utilisateur vers une page Web, pour visualiser ledit état et lui permet de lancer automatiquement le processus de collecte des données pour les balances des GABs non encore intégrées.
- l'entité « BalanceStatusAction » est ensuite appelée pour scruter les nouvelles données venues et présenter le sommaire des balances déjà intégrées. Ainsi, elle formate ces données selon les paramètres retournés par l'appel de l'objet « BalanceStatusForm ». Cet acheminement d'appel automatique d'entités est détaillé dans le fichier de configuration Struts.
- à cette phase l'objet «BalanceStatusAction » fait appel à l'objet responsable du traitement « GestionnaireBalance ». Celui-ci se charge de la réalisation des traitements nécessaires à travers l'appel de la méthode « getTransactionStatus() ».

- l'entité « GestionnaireBalance » récupère la requête de l'objet « BalanceStatusAction » et fait appel à l'objet « Utility» à l'aide de la méthode « getTransactionParameters() » pour charger les paramètres de la transaction.

- ensuite, l'entité « GestionnaireBalance » lance une instance de l'objet Transaction qui extrait chaque ligne du fichier de la transaction à l'aide de la méthode « loadTransactionData()». Comme c'est expliqué dans les besoins fonctionnels, il faut valider cette ligne de transaction.

- l'entité « GestionnaireBalance » crée une instance de l'objet « Expression » qui doit valider la transaction à l'aide de la méthode « validateTransactionData()». Cette méthode valide la transaction syntaxiquement à l'aide des expressions régulières.

- une fois ladite vérification est accomplie, l'objet « Transaction » est sauvegardé dans la base de données à l'aide de la méthode « saveTransactionData() ».

- pour concrétiser l'aspect de la journalisation, l'entité « GestionnaireBalance » crée une instance de l'objet « Historique » pour décrire l'exécution du système à cette étape à l'aide de la méthode « saveTransactionState() ».

- après, l'entité « BalanceStatusAction » récupère la bonne exécution de toutes les étapes appelées de l'entité « GestionnaireBalance ». Puis, elle affiche à l'utilisateur l'affirmation du succès du traitement par la redirection vers la page Web « balanceSucces.jsp ».

2. Factorisation des traitements

En observant les diagrammes de collaboration construits, j'ai constaté qu'il y a des traitements qui se répètent dans plusieurs cas d'utilisation. Une approche qui est la plus intuitive pour palier à cette similitude de traitement est de les factoriser. Cette démarche satisfait en même temps les objectifs fonctionnels tracés auparavant. La figure26 suivante représente le diagramme de collaboration où la partie entourée par un cadre doit être factorisée :

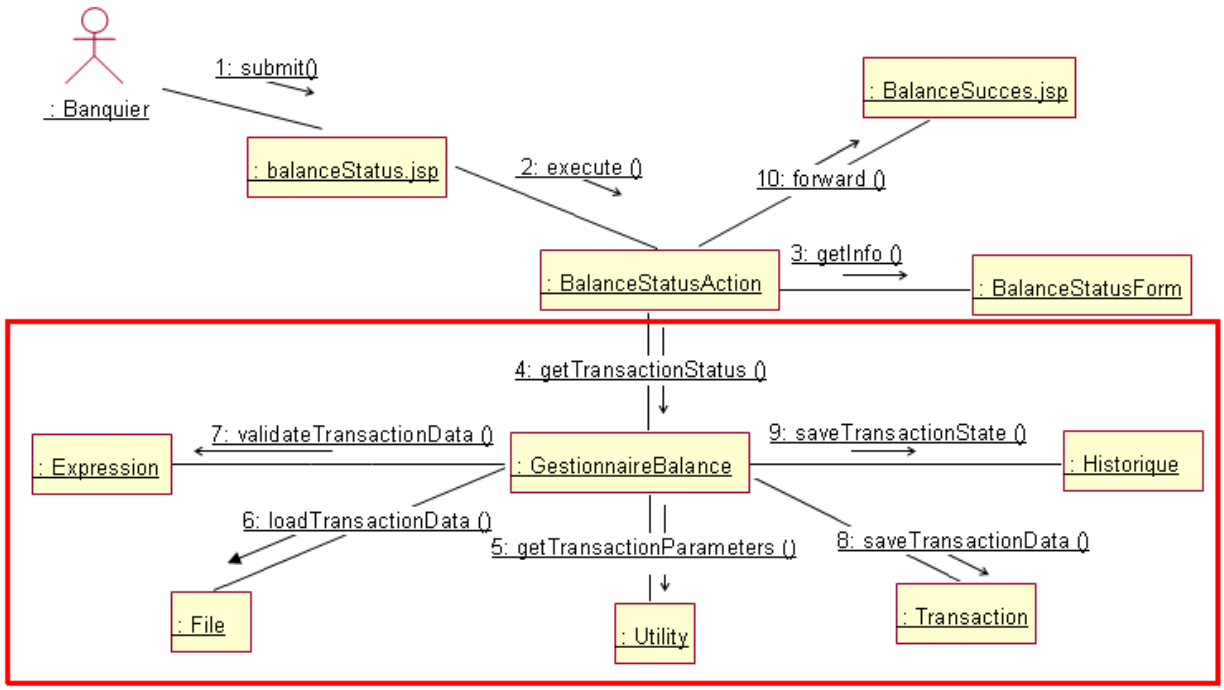


Figure 26 : partie du diagramme de collaboration devant être factorisée.

La similitude de traitement est visible sur deux niveaux :

Premièrement, ce diagramme de collaboration ne reflète que le traitement relatif à une seule transaction. Cependant, un GAB effectue plusieurs transactions quotidiennement. Ce traitement se répète autant de fois que le nombre des transactions. Deuxièmement et selon l'étude fonctionnelle, les données de la balance sont récupérées en cinq étapes :

- récapitulatif de la journée : « Balance ».
- liste des transactions : « Transaction ».
- détail des transactions par devise : « Denomination ».
- détail des transactions par les coupures de la devise étrangère : « Foreign ».
- détail des transactions par les coupures en Dirham : « Local ».

Le changement nécessaire dans le diagramme de collaboration de la figure25 est de remplacer l'appel de fonction « getTransactionStatus() » par « getBalanceStatus() », « getDenominationStatus() », « getForeignStatus() » et « getLocalStatus() ».

Le changement se poursuit même sur la logique de l'appel. Par exemple, la validation syntaxique de l'entité « Transaction » est différente de celle de l'entité « Denomination ».

Une classe générique est implémentée pour ce traitement, la différence réside dans les caractéristiques relatives à chacune des parties décrites. La figure27 suivante représente la classe « BalanceStatus » finalement adoptée pour factoriser celles de « TransactionStatus », « DenominationStatus », « ForeignStatus » et « LocalStatus ».

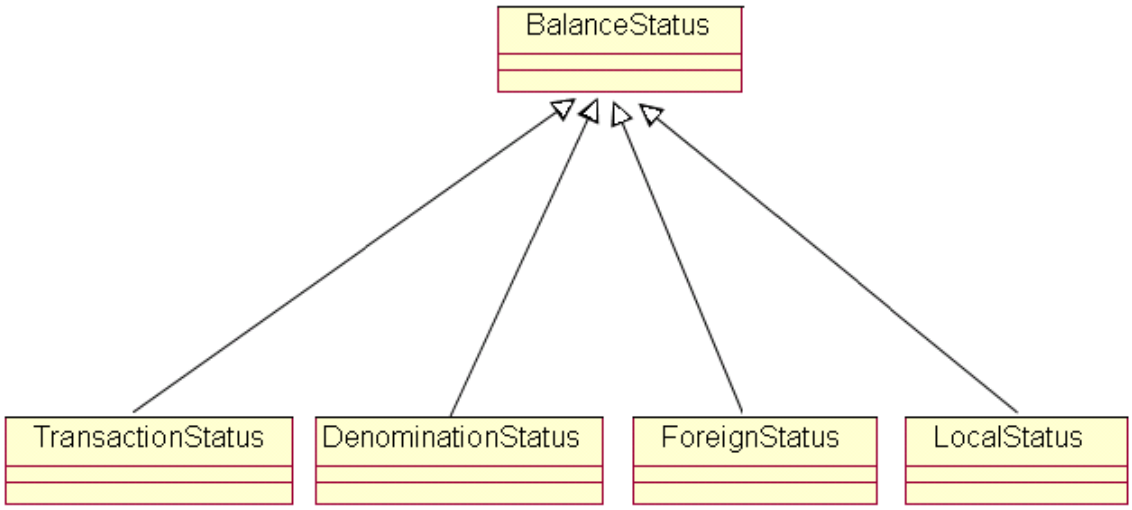


Figure 27 : factorisation du traitement en « BalanceStatus ».

3. Solutions adoptées

3.1. Multi langage et messages

Compte tenu du caractère multi langage adopté par l'application, la solution se base sur la manipulation des fichiers d'internalisations offerte par l'environnement J2EE. De ce fait, chaque clé dans le fichier correspond à un mot d'une langue donnée. Par ailleurs, à chaque langue correspond un fichier groupant un ensemble de clés.

Le fichier français est nommé « ApplicationRessources_fr.properties » et celui anglais américain est « ApplicationRessources_us.properties ». Chacun des fichiers a un ensemble de clés similaires, ce qui va les distinguer, c'est la valeur de cette clé. Par exemple, la clé « rate.update.confirm » correspond au message renvoyé à l'utilisateur lors de la confirmation de mise à jour du taux de change.

De ce fait, la clé a la valeur « La mise à jour du taux de change est effectuée » dans le fichier des messages en français et la valeur « exchange rate is updated successfully » dans le fichier des messages en anglais. Cette solution d'internationalisation s'adapte pour chaque utilisateur en détectant ses paramètres locaux. Elle est utilisée lors de l'affichage des libellés et les messages sur la page Web. D'autre part, ces valeurs sont récupérées au nivaux du code pour le contrôle dynamique des événements déclenchés par les utilisateurs.

La figure28 suivante représente l'adaptation de l'interface utilisateur selon la configuration relative à l'utilisateur en lui chargeant les messages et les libellés adéquats.

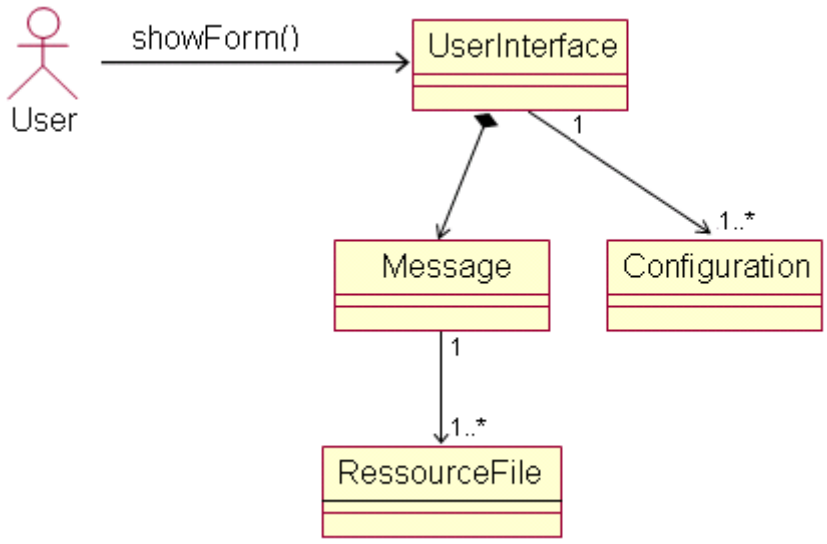


Figure 28 : Pattern pour l'adaptation de l'application selon les paramètres de l'utilisateur.

3.2. Saisie de la date

L'utilisateur doit saisir la date de temps à autre, par exemple pour afficher le récapitulatif des balances d'une période donnée, l'utilisateur est invité à saisir la date début et la date fin. Le problème se pose sur le format de la date saisie. En effet, les requêtes utilisant un paramètre date erroné ne trouvent pas leurs acheminements vers l'utilisateur.

Ainsi la conception d'une interface graphique représentant le calendrier s'avérait nécessaire. La figure suivante montre comment l'utilisateur peut manipuler une date sans avoir à la saisir manuellement.

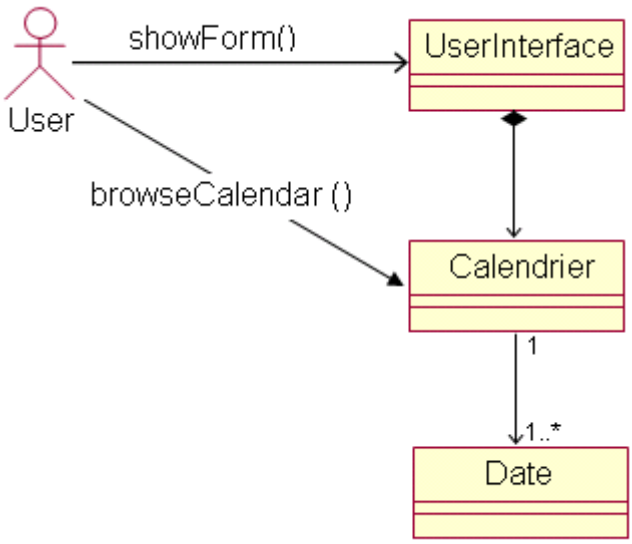


Figure 29 : Pattern de manipulation de la date à travers un calendrier.

3.3. Affichage dynamique

Au cours de leur manipulation, les données sont récupérées à partir d'une base de données. De ce fait, leur nombre et leur gestion doivent tenir compte de cet aspect dynamique. Le problème se pose sur l'indexation des données récupérées de la base de données. Toutefois, le Framework Struts standard gère ces données de façon statique, donc ne permet pas cet aspect d'indexation. La solution résidait dans l'intégration d'une librairie supplémentaire appelée « Struts-el ».

La figure30 montre l'adaptation effectuée pour palier à ce problème. L'exemple présenté est celui de la mise à jour du taux de change.

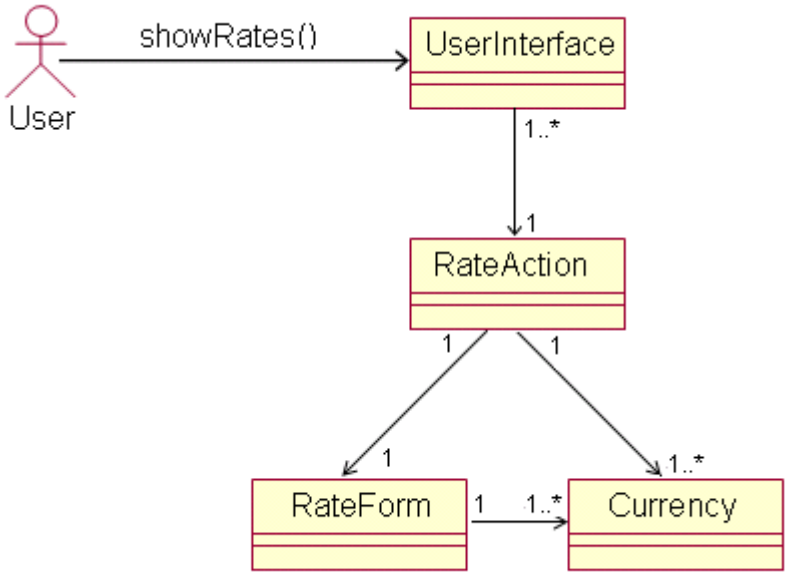


Figure 30 : Pattern de gestion dynamique des données dans la page Web.

La valeur ajoutée de l'aspect dynamique, offerte par la librairie « Struts-el », est l'ajout d'une classe « Currency ». Cette entité regroupe toutes les données devant être gérées de façon dynamique.

Ainsi, la nouvelle fonctionnalité agit sur les objets de la classe « Currency » en les regroupant dans un ensemble indexé mappé aux entités de la base de données. Par ailleurs, la classe « RateForm » se charge d'envoyer les données à l'utilisateur.

3.4. Edition dynamique

L'édition permet aux décideurs de visualiser les données effectivement utilisées dans l'application. Puisque les données sont récupérées à partir d'une base de données, elles doivent être gérées dynamiquement. Le problème se pose en l'adaptation du modèle d'édition quelque soit la taille des données récupérées de la base de données.

Le Framework JasperReport gère cet aspect dynamique. De ce fait, une compilation est effectuée chaque fois qu'il y a appel à la fonctionnalité d'édition. L'avantage de la séparation des données de la vue est présent dans ce Framework. La figure31 montre les composants nécessaires pour réaliser l'édition. L'exemple présenté est celui de l'édition du récapitulatif de la balance quotidienne :

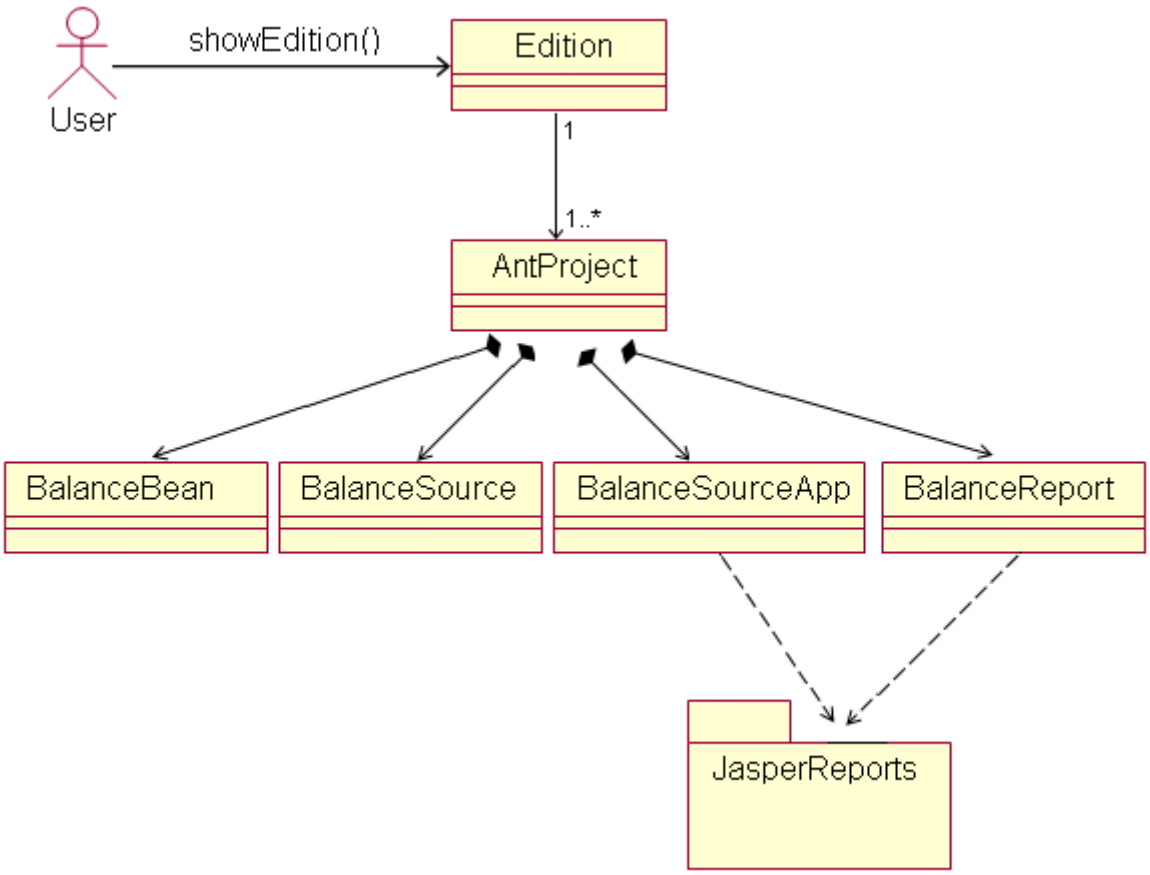


Figure 31 : Pattern l'édition de la balance.

L'utilisateur peut consulter la fonctionnalité d'édition de la balance en appelant l'objet « Edition ». Celui-ci fait appel au projet associé en le compilant. Le projet « AntProject » recompile les fichiers nécessaires.

- « BalanceBean » définit l'objet devant être imprimé.
- « BalanceSource » représente l'instanciation des objets « BalanceBean » dont les données sont obtenues à partir de la base de données.
- « BalanceReport » décrit le modèle graphique où les données doivent être insérées.
- « BalanceSourceApp » fait l'association entre les données et la vue pour construire la page désirée.

Par ailleurs, les deux entités « BalanceReport » et « BalanceSourceApp » utilisent l'API de JasperReports pour accomplir leurs finalités.

3.5. Tache planifiée

Une tache planifiée permet le lancement automatique des traitements répétitifs au début de chaque période donnée. Le problème était de trouver un mécanisme paramétrable et générique. Comme exemple, les GABs remontent leurs balances quotidiennement, l'utilisateur peut laisser ce traitement au système.

Une telle fonctionnalité est réalisée par une compilation chaque fois qu'il y a paramétrage de cette automatisation. La figure32 suivante résume les composants collaborant pour automatiser la collecte de la balance :

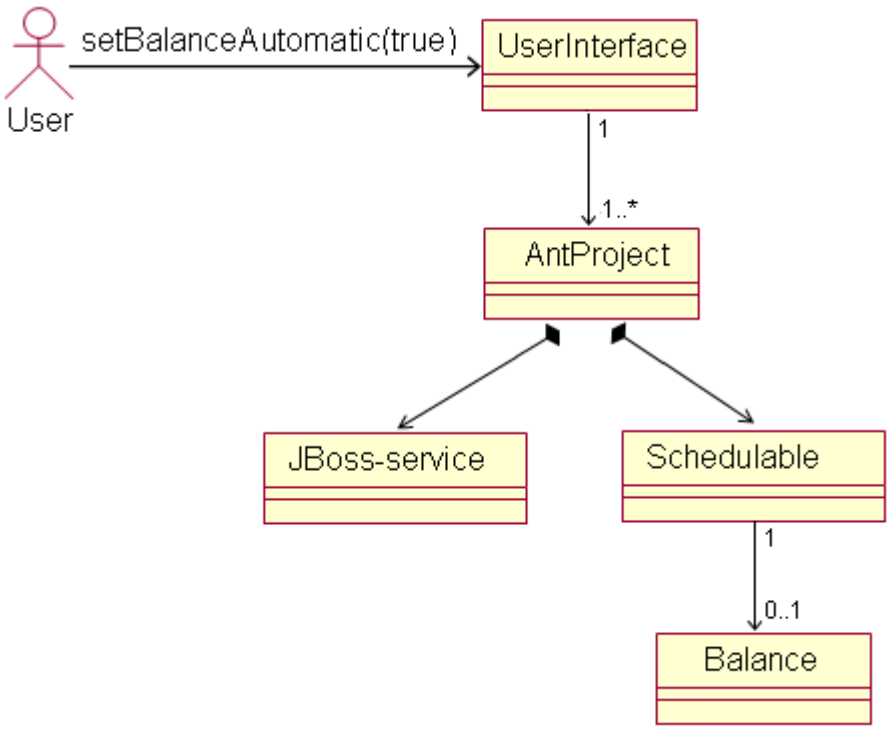


Figure 32 : Pattern d'automatisation de la collecte des balances GABs.

4. Modèle final de conception

Jusqu'ici, la construction du modèle conceptuel est accomplie et les solutions pour son implémentation sont vouées ; j'ai défini, l'architecture et les couches qu'il propose. Je vais, maintenant, présenter le diagramme de package final de conception dans la figure33 suivante :

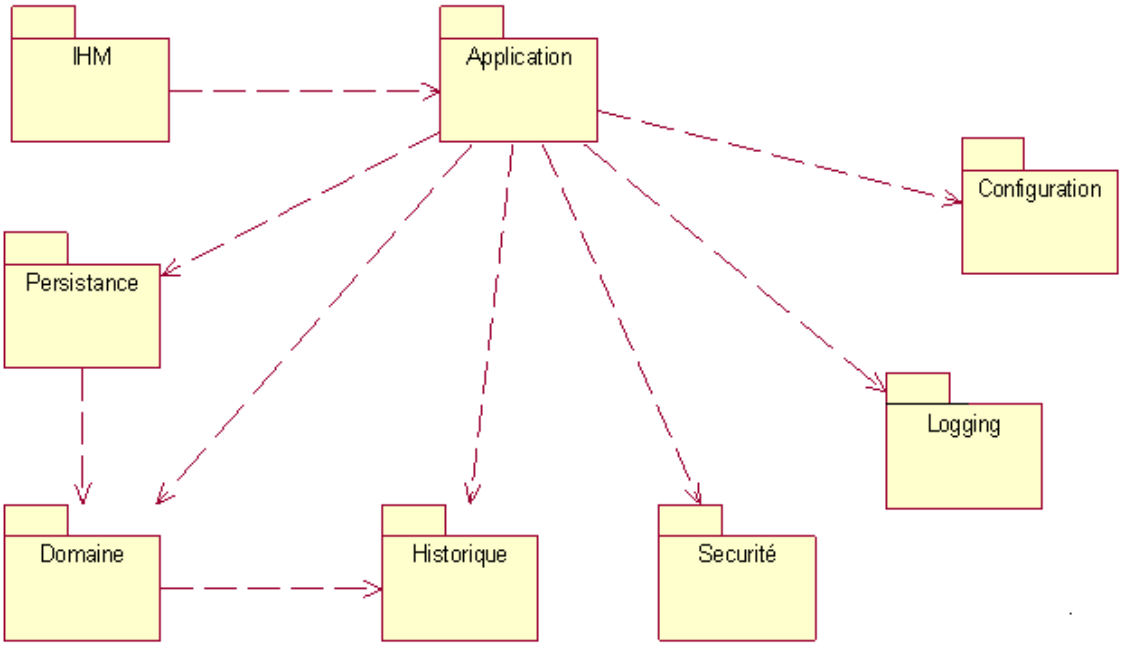


Figure 33 : diagramme de package final de conception.

Ce diagramme de package est structuré comme suit :

- IHM : ce package regroupe toutes les classes relevant de la couche présentation. C'est dans ce package, que j'ai mis les classes relatives au Framework Struts.
- Application : ce package regroupe les classes qui appartiennent à la couche Application de l'architecture réalisée. Il comporte les gestionnaires, qui effectuent les traitements nécessaires aux requêtes des utilisateurs et qui appellent les services techniques nécessaires.
- Configuration : ce package contient les classes qui encapsulent la configuration nécessaire à l'exécution du système.
- Logging : ce package représente le service de suivi et de traçage et de journalisation.
- Sécurité : ce package contient le Framework de sécurité et qui assure le contrôle de l'accès aux fonctionnalités du système.
- Domaine : ce package contient les classes métiers du système détaillées tout au long de l'étude fonctionnelle.
- Historique : ce package contient les classes de l'historisation.

- Persistance : ce package regroupe le Framework de persistance générique adopté.

Au terme de cette phase de conception, j'ai rédigé les documents suivants :

- les diagrammes de collaborations correspondants à l'adaptation des diagrammes de séquence à l'architecture choisie.
- les diagrammes de classes des différents patterns et le Framework de traitement développés.
- le diagramme de package final.

Conclusion

Dans ce chapitre, j'ai présenté la phase de conception du projet dans lequel, j'ai repris les scénarios établis dans la phase d'analyse en incluant la branche technique étudiée. Puis j'ai exposé les solutions adoptées afin d'aboutir au modèle final de conception.

Dans le chapitre qui suit, je vais détailler la phase de réalisation du projet.

Chapitre 6 : Réalisation

Dans ce chapitre, je vais exposer la partie réalisation. Je vais présenter les outils utilisés et je vais décrire en détail l'implémentation du modèle final de conception. Après, je vais dévoiler quelques écrans de l'application.

1. Outils utilisés

1.1. SGBD Oracle

Oracle est un système de gestion de bases de données édité par la société du même nom Oracle Corporation, leader mondial des bases de données. Oracle se décline en plusieurs versions :

- Oracle Server **Standard**, une version comprenant les outils les plus courants de la solution Oracle.
- Oracle Server **Enterprise Edition**.

Oracle est un SGBD permettant d'assurer :

- la définition et la manipulation des données.
- la cohérence des données.
- la confidentialité des données.
- l'intégrité des données.
- la sauvegarde et la restauration des données.
- la gestion des accès concurrents.

Outre la base de données, la solution Oracle est un véritable environnement de travail constitué de nombreux logiciels permettant notamment une administration graphique d'Oracle, de s'interfacer avec des produits divers et d'assistants de création de bases de données et de configuration de celles-ci.

Les outils d'Oracle peuvent se classer selon diverses catégories :

- Les outils d'administration.
- Les outils de développement.
- Les outils de communication.

- Les outils de génie logiciel.
- Les outils d'aide à la décision.

Durant la phase de développement du projet, il fallait faire appel aux outils d'administration d'oracle pour l'implémentation de la base de données et pour la manipulation des données sources. Parmi les outils utilisés au cours de développement :

- Enterprise Manager Console.
- SQLPlus Worksheet.
- SQL Plus.
- Database Configuration Assistant.
- Import/Export : outil permettant d'échanger des données entre deux bases Oracle.

1.2. Serveur d'application JBoss

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

Gestion de la session utilisateur

Comme les clients utilisant une même instance d'application sur le serveur, il est nécessaire que le serveur d'application puisse conserver les contextes propres à chaque utilisateur. Or la plupart des serveurs d'application génèrent un identifiant unique pour chaque nouveau client et transmettent cet identifiant lors de chaque échange HTTP par URL longs, variables cachées ou cookies.

Gestion des montées en charge et reprise sur incident

Afin de gérer toujours plus d'utilisateurs, le serveur d'application doit pouvoir se déployer sur plusieurs machines et éventuellement offrir des possibilités de reprise

sur incident (même si dans la grande majorité des cas, on se contente d'une gestion des montées en charge au niveau réseau).

Ouverture sur de multiples sources de données

C'est le serveur d'application qui rend accessible les données des applications du système d'information. Il doit donc pouvoir accéder à de nombreuses sources de données. On s'attend également à ce qu'il fournisse des mécanismes performants tels que le pooling de connexion base de données.

Pour le choix d'un serveur d'application, j'en ai étudié quatre qui se basent sur la plate forme J2EE. Ce sont les serveurs d'applications suivants:

- BEA WebLogic Server.
- IBM WebSphere.
- JBoss Application Server.
- Oracle Application Server.

Je me suis inspiré d'une étude[6] basée sur la participation des 1,148 professionnelles des middleware. La source est le site web : Middleware.com, juin 2004. Cette étude, est aussi, commandée par SUN MICROSYSTEMS. La figure suivante montre la part de chacun de ces serveurs d'application :

[6] Réalisée par TheServerSide.com: online community of J2EE practitioners.

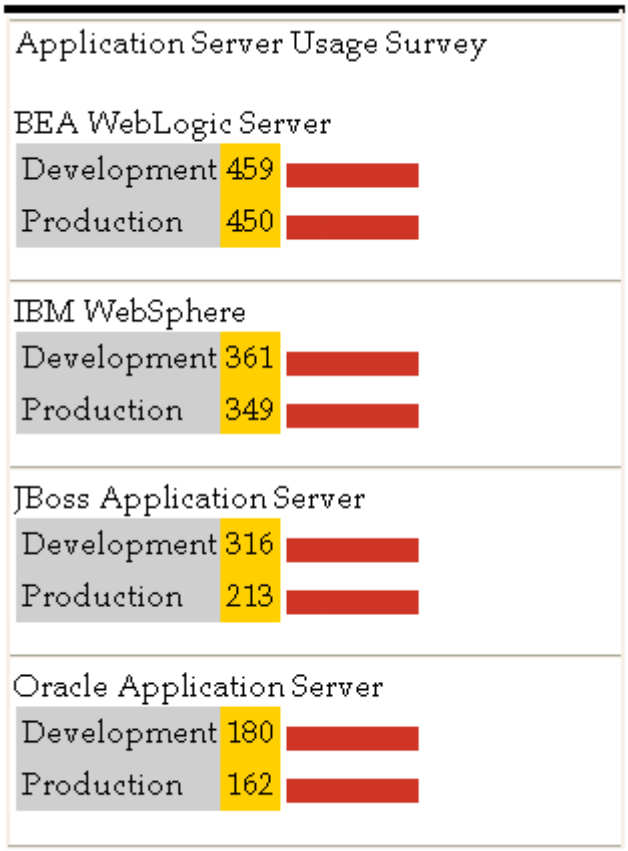


Figure 34 : les principaux serveurs d'application utilisant la plate forme J2EE.

JBoss Application Server 4.0 est le premier serveur d'application Open Source à être certifié compatible à la plate forme J2EE (Java 2 Platform, Enterprise Edition) en passant 23,000 tests de compatibilité. De plus, c'est le premier à implémenter l'AO (Aspect Orientation) pour Java. Apportant, ainsi, plus de productivité, de performance et la capacité de gestion pour les utilisateurs.

Par ailleurs, JBoss AS 4.0 est disponible pour les entreprises sous la licence Open Source LGPL (Lesser General Public License). Enfin, JBoss est compatible avec les principaux systèmes d'exploitation du marché. La figure suivante montre une étude[7] de la tendance des utilisateurs de JBoss relativement au système d'exploitation :

[7] Multiples réponses sont permises.

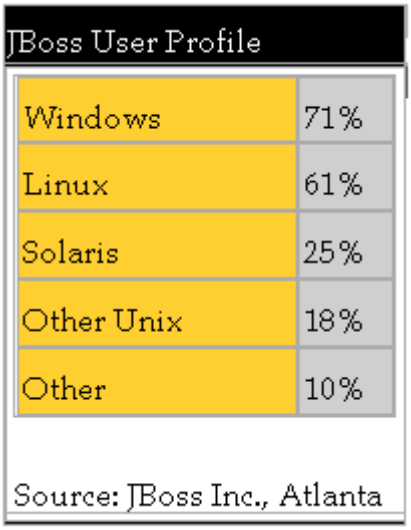


Figure 35 : tendance des utilisateurs du JBoss AS.

1.3. IDE Eclipse

Eclipse est un environnement de développement intégré (Integrated Development Environment), dont le but est de fournir une plate-forme modulaire, pour permettre de réaliser des développements informatiques [wwwDOUDOUX]. IBM est à l'origine du développement d'Eclipse, qui est d'ailleurs toujours le cœur de son outil Websphere Studio Workbench (WSW), lui même à la base de la famille des derniers outils de développement en Java d'IBM. Tout le code d'Eclipse a été donné à la communauté par IBM, afin de poursuivre son développement.

Eclipse utilise énormément le concept de modules nommés « Plug-ins » dans son architecture. D'ailleurs, hormis le noyau de la plate-forme nommé « Runtime », tout le reste de la plate-forme est développé sous la forme de « Plug-ins ». Ce concept, permet de fournir un mécanisme pour l'extension de la plate-forme afin d'offrir la possibilité à des tiers de développer des fonctionnalités qui ne sont pas fournies en standard par Eclipse.

Les principaux modules fournis en standard avec Eclipse concernent Java. Des modules sont en cours de développement pour d'autres langages notamment C++, Cobol, mais aussi pour d'autres aspects du développement (base de données, conception avec UML, …). Ils sont tous développés en Java, soit par le projet Eclipse soit par des tiers commerciaux ou en Open Source.

Bien que développé en Java, les performances à l'exécution d'Eclipse sont très bonnes, car il n'utilise pas Swing pour l'interface Homme/Machine mais un toolkit particulier nommé SWT associé à la bibliothèque JFace. SWT (Standard Widget Toolkit) est développé en Java par IBM, en utilisant au maximum les composants natifs fournis par le système d'exploitation sous jacent. JFace utilise SWT et propose une API pour faciliter le développement d'interfaces graphiques.

SWT et JFace sont utilisés par Eclipse pour développer le Workbench, qui organise la structure de la plate-forme et les interactions entre les outils et l'utilisateur. Cette structure repose sur trois concepts : la perspective, la vue et l'éditeur. La perspective regroupe des vues et des éditeurs, pour offrir une vision particulière des développements. En standard, Eclipse propose huit perspectives. Les vues permettent de visualiser et de sélectionner des éléments. Les éditeurs permettent de visualiser et de modifier le contenu d'un élément du Workspace. En plus, il offre un espace de travail qui facilite les tâches courantes du développeur Java.

En effet, il permet la coloration de la syntaxe, la complétion de code, le formatage et l'indentation automatiques du code, la navigation et l'accessibilité au sein des différentes ressources. L'espace de travail de l'IDE Eclipse apporte un confort visuel et une facilité inégalés dans la tâche de codage du développeur Java. Eclipse supporte en plus le factoring du code. Les capacités de factoring de code consistent à réarranger les objets et la logique d'une portion de code tout en préservant le comportement de l'ensemble. Eclipse fournit pour cela, un ensemble de facilités qui améliorent tout à la fois la lisibilité et la structure du code, l'extraction des méthodes et leur renommage dans le projet, par le biais d'assistants de transformation et de propagation des modifications avant aperçu.

Pour le choix d'un environnement de développement intégré, j'en ai étudié[8] cinq qui se basant sur la plate forme J2EE. Les environnements suivants sont les plus complets du marché. Cependant, Eclipse est gratuit:

[8] Comparaison http://mojavelinux.com/wiki/doku.php

IDE	Prix	Licence	Url
Eclipse 3.0	Gratuit	Open Source, CPL	http://eclipse.org/
IDEA (IntelliJ)	$500	Propriétaire	http://www.jetbrains.com/idea
JBuilder (Borland)	$3,500	Propriétaire	http://borland.com/jbuilder
JDeveloper (Oracle)	$995, gratuit pour utilisation non commerciale	Propriétaire	http://www.oracle.com/technology/products/jdev
NetBeans (Sun Microsystems)	Gratuit	Open Source (SPL)	http://netbeans.org/

Figure 36 : comparaison entre les IDEs de développement.

2. Reprise des scénarios

A cette étape, je vais traduire le modèle de conception pour construire l'implémentation. Cette reprise du modèle est effectuée en s'appuyant sur les solutions techniques adoptées.

Pour éclaircir cette démarche, je vais reprendre le diagramme de collaboration relatif au cas d'utilisation collecte de transaction d'un GAB (établit dans la partie de conception de la figure25) en expliquant l'implémentation réalisée.

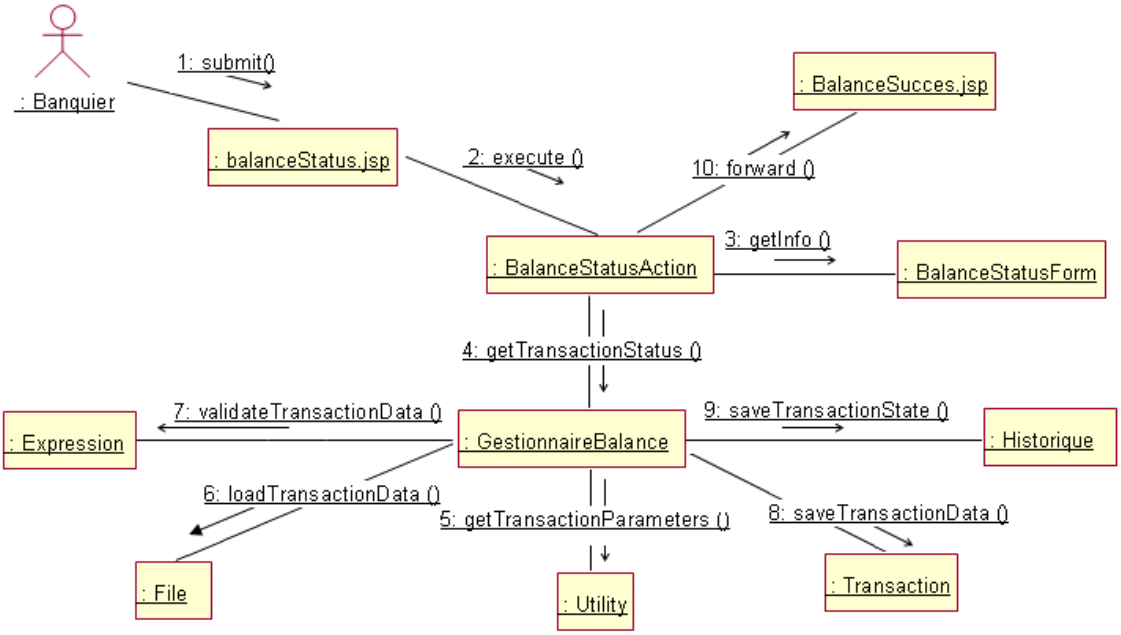


Figure 25 : diagramme de collaboration de collecte d'une transaction d'un GAB.

Après l'identification de l'utilisateur, il invoque la page « b_status.jsp » à partir du menu qui lui est associé. Cette figure37 est relative à la deuxième étape du diagramme de collaboration précédent :

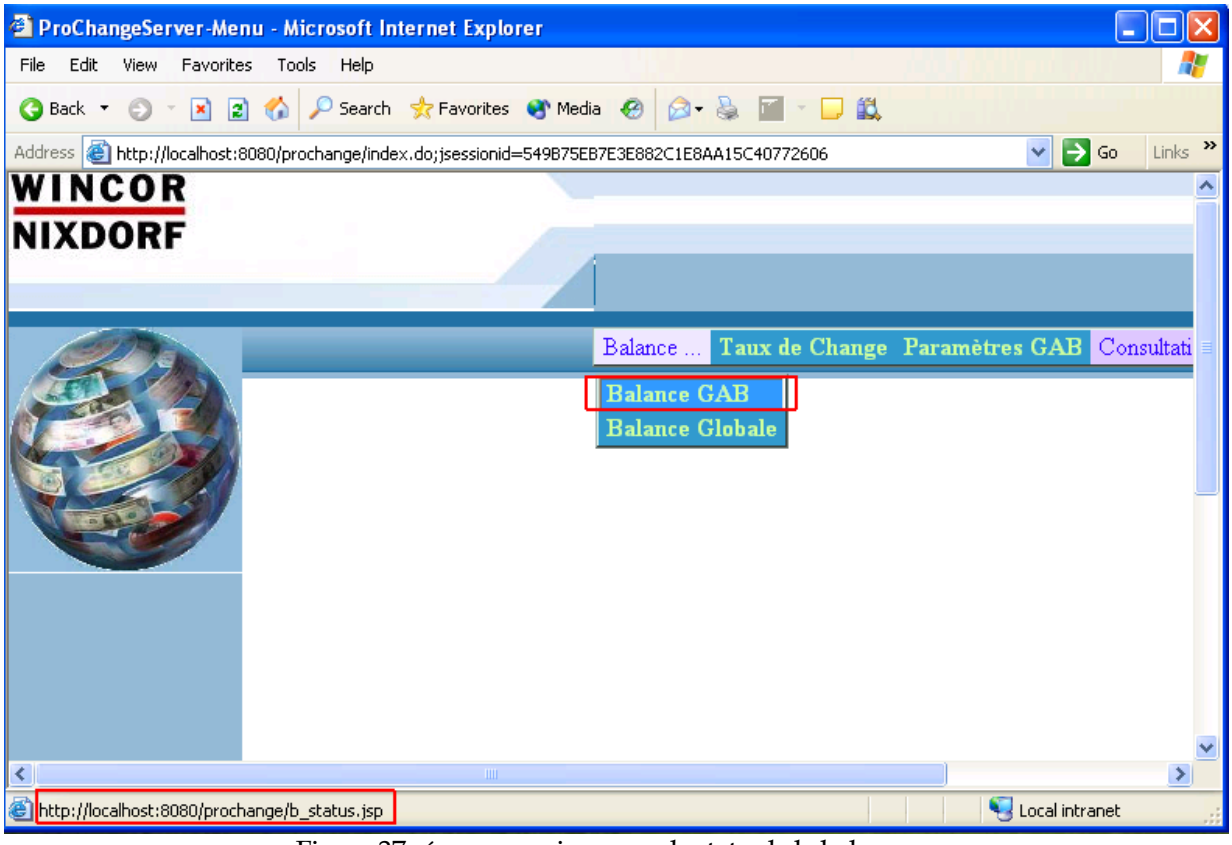


Figure 37 : écran pour invoquer le statu de la balance.

La requête de l'utilisateur est interceptée par la classe « B_statusAction ». La figure38 suivante définit les caractéristiques des objets « B_statusAction » et « B_statusForm ». Le fichier « struts-config.xml » résume cette association.

```
<form-beans >
   <form-bean name="b_statusForm" type="com.wincor.form.B_statusForm" />
   ...
</form-beans>

   <action
      attribute="b_statusForm"
      input="/b_status.jsp"
      name="b_statusForm"
      path="/b_status"
      scope="request"
      type="com.wincor.action.B_statusAction">
      <forward name="integrer" path="/b_status.jsp" />
      <forward name="menu" path="/menu.jsp" />
      <forward name="detail" path="/detail.jsp" />

   </action>
```

Figure 38 : mappage des objets dans « struts-config.xml ».

L'appel du message n°3 est réalisé à cette étape. L'objet « B_statusAction » récupère les données de l'objet « B_statusForm ». Ceci est réalisé après le casting de l'objet « form ». La figure 39 suivante présente la récupération des informations à partir de l'objet « B_ statusForm ».

```
public class B_statusAction extends Action {

    public ActionForward execute(
        ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response) {


        B_statusForm b_statusForm = (B_statusForm) form;

        String valider = b_statusForm.getValider();
```

Figure 39 : récupération des informations à partir de «B_statusForm ».

Après la récupération des paramètres de la transaction et le chargement des données dans des variables temporaires, la valeur syntaxique des données est contrôlée dans la partie n°7 du diagramme de collaboration en question. La vérification est accomplie suivant une expression régulière définissant le format des données utilisées.

La figure40 suivante représente cette validation syntaxique :

```
String time   = "([0-3](1)[0-9](1))/([0-1](1)[0-9](1))/([2](1)[0-9](3))" + " " +
                "([0-2](1)[0-9](1)):([0-5](1)[0-9](1)):([0-5](1)[0-9](1))";
String number = ",([0-9]*)";
String status = ",(OK|NOK)";
String total  = ",([0-9]*).([0-9]*)";

String textPattern1 = time + number + status + total + total;

    compiledRegex = Pattern.compile(textPattern1, Pattern.CASE_INSENSITIVE);

        try {
            Pattern pattern1 = Pattern.compile(textPattern1, Pattern.CASE_INSENSITIVE);
            Matcher matcher1 = pattern1.matcher("");}
        catch (Exception e) {}
```

Figure 40 : validation syntaxique des données.

Après ladite vérification, l'insertion dans la base de données est effectuée. Ensuite, la sauvegarde de l'état du système est accomplie. La figure41 suivante représente comment est réalisée la journalisation de l'état du système après la bonne réception des données.

```
Logger logger = Logger.getLogger(Transaction.class);

SimpleLayout layout = new SimpleLayout();

FileAppender appender = null;
try {
   appender = new FileAppender(layout,"ProChangeServer.log",false);
} catch(Exception e) {}

logger.addAppender(appender);
logger.setLevel((Level) Level.DEBUG);

logger.info( SysDate + " : La Transaction est effectuée avec succès");
```

Figure 41 : journalisation de l'état du système.

La bonne exécution de ces étapes constitue l'accomplissement du scénario « collecte d'une transaction d'un GAB ».

3. Interfaces de l'application

L'interface de l'application permet une meilleure communication entre le système et l'utilisateur. De ce fait, elle facilite l'utilisation et aide à la bonne exploitation des fonctionnalités.

Puisque la solution **ProChangeServer** se base sur les technologies Web, l'aspect ergonomique adopté tire profit des potentialités offertes par les navigateurs Web.

3.1. Authentification

L'authentification est la première fenêtre rencontrée par l'utilisateur. Elle lui permet de s'identifier en saisissant son nom et son mot de passe. Selon son groupe d'utilisateurs, le système lui donne ses droits d'accès. La figure42 suivante montre la forme d'authentification:

Figure 42 : authentification de l'utilisateur.

3.2. Menu

Après l'authentification, le menu est activé selon le groupe de l'utilisateur. La figure43 suivante illustre le menu relatif à l'administrateur :

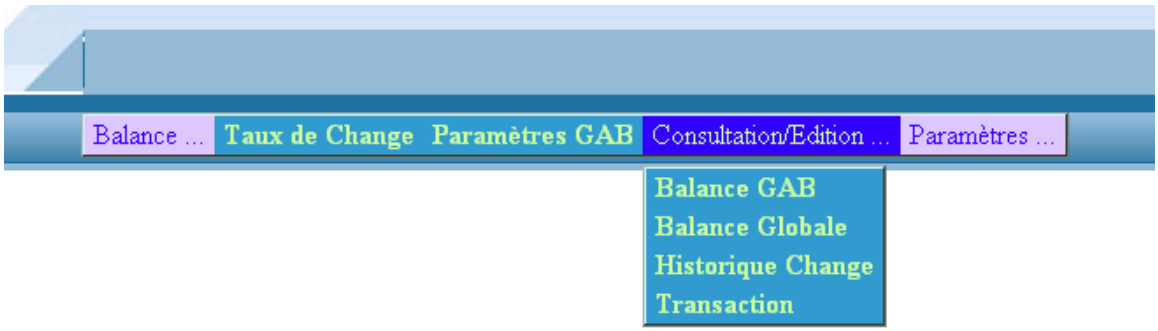


Figure 43 : menu relatif à l'administrateur.

3.3. Menu « Taux de change »

Un utilisateur privilégié peut mettre à jour les taux de change. La figure44 suivante reflète cette fonctionnalité :

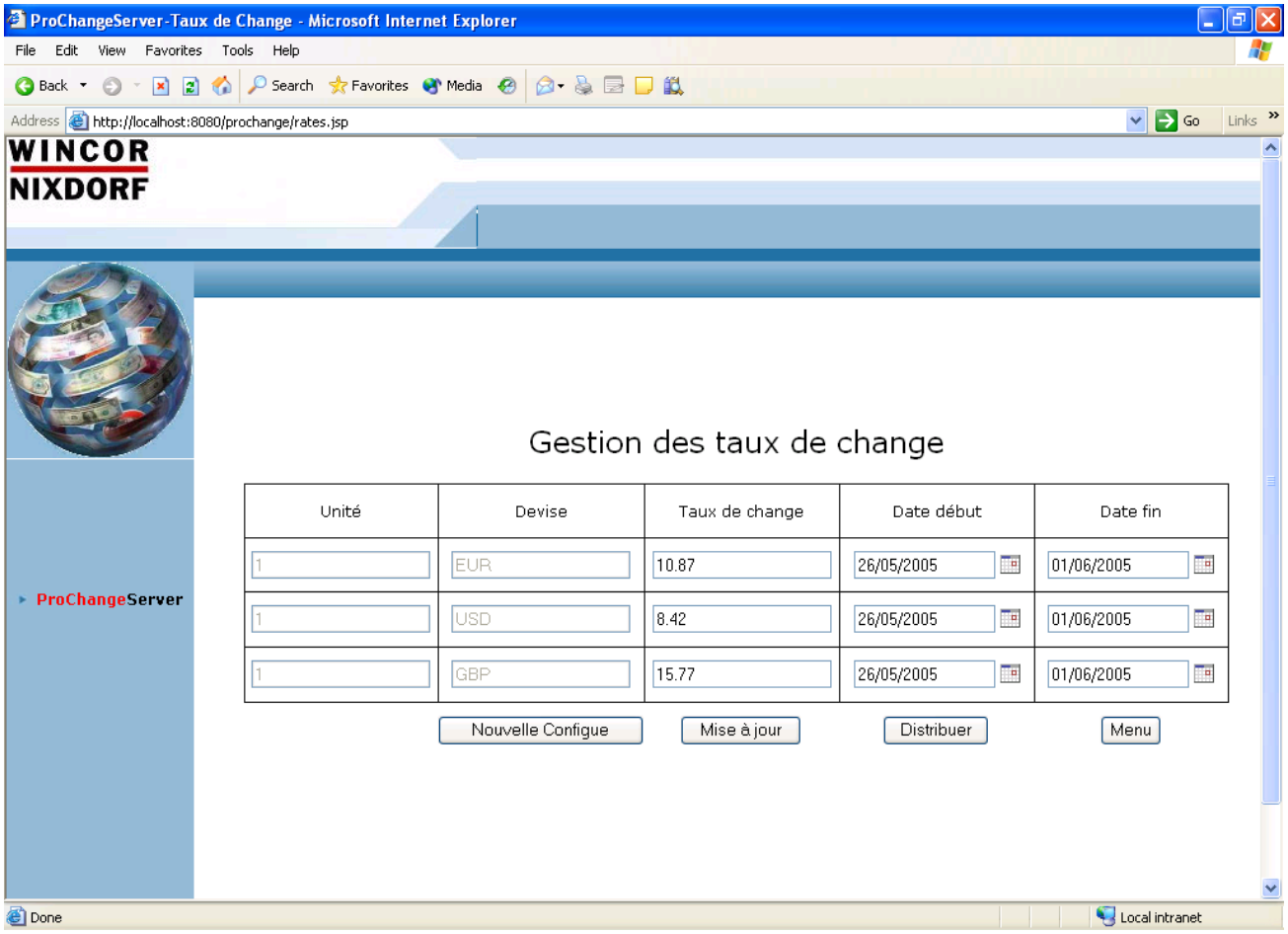


Figure 44 : écran de gestion des taux de change.

3.4. Gestion de la date

Afin de faciliter l'exploration du calendrier et de faciliter la saisie de la date selon un format particulier, j'ai représenté le calendrier sous une page Web. La figure45 suivante indique la simplicité de la manipulation de la date:

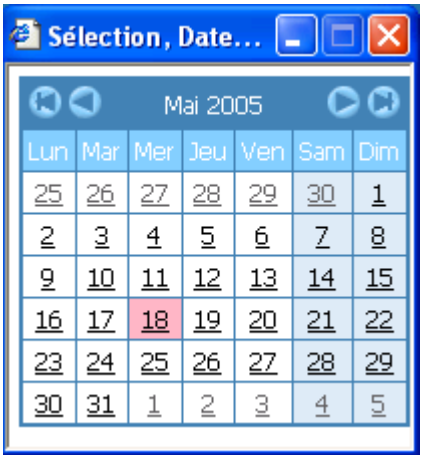
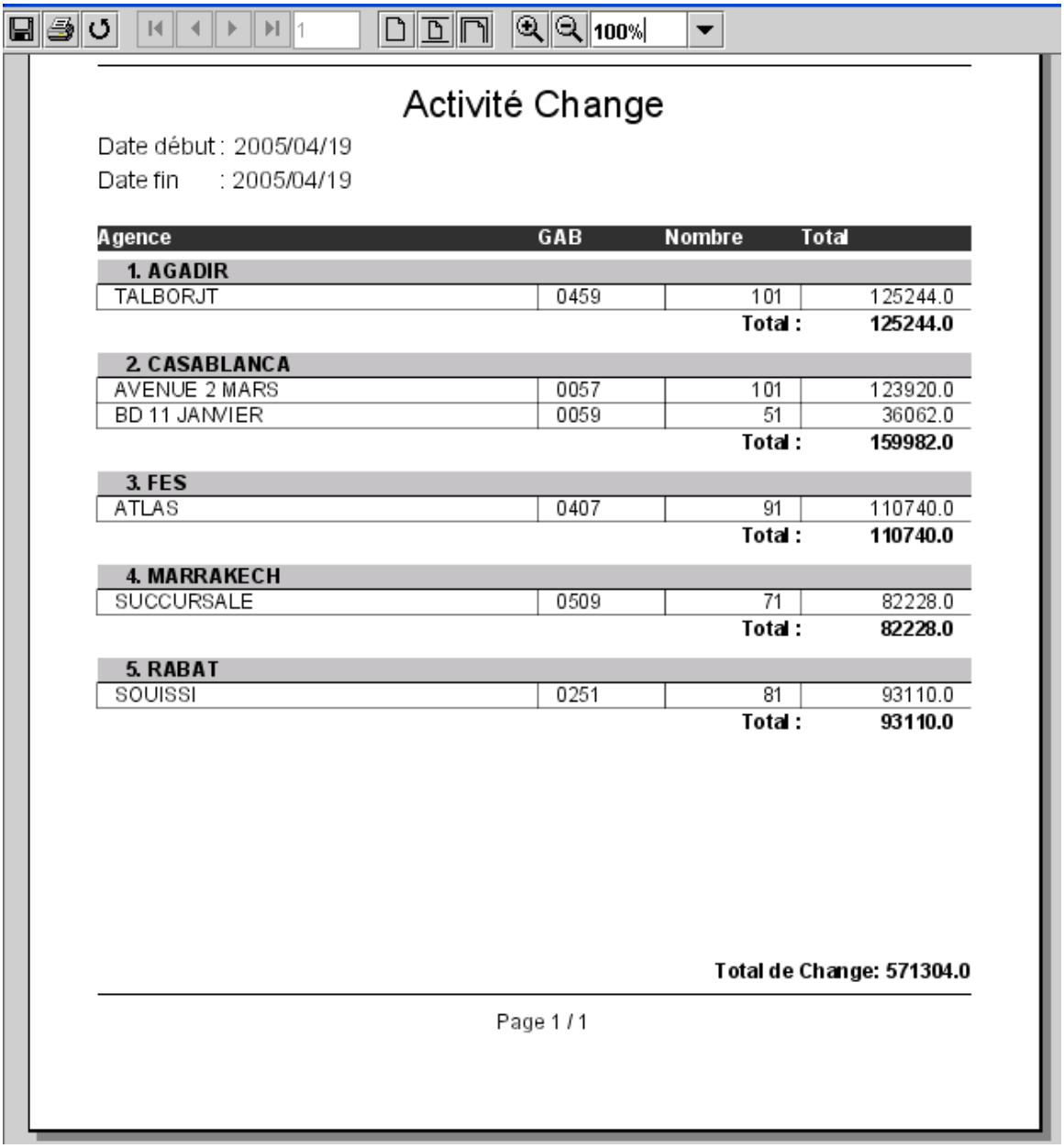


Figure 45 : calendrier pour sélectionner la date.

3.5. Edition des rapports

Pour consulter les données existantes dans la base de données, le système peut générer des éditions. La figure46 suivante illustre le sommaire de l'activité change d'une journée.



Activité Change

Date début : 2005/04/19
Date fin : 2005/04/19

Agence	GAB	Nombre	Total
1. AGADIR			
TALBORJT	0459	101	125244.0
		Total :	**125244.0**
2. CASABLANCA			
AVENUE 2 MARS	0057	101	123920.0
BD 11 JANVIER	0059	51	36062.0
		Total :	**159982.0**
3. FES			
ATLAS	0407	91	110740.0
		Total :	**110740.0**
4. MARRAKECH			
SUCCURSALE	0509	71	82228.0
		Total :	**82228.0**
5. RABAT			
SOUISSI	0251	81	93110.0
		Total :	**93110.0**

Total de Change: 571304.0

Page 1 / 1

Figure 46 : édition de l'activité de change.

Conclusion

Ce chapitre, a été consacré à la phase de réalisation. J'en ai présenté les outils utilisés et j'en ai exposé les étapes d'implémentation selon le modèle de conception construit. Enfin, j'ai présenté quelques écrans de l'interface homme/machine de l'application.

La conclusion de ce document fera l'objet de la partie suivante.

Conclusion

Mon projet consistait à concevoir et développer une solution de change de devises en libre service pour la société WINCOR-NIXDORF. Cette solution avait pour objectif de permettre la consolidation des données récupérées des Guichets Automatiques Bancaires et de les remonter au serveur de la banque centrale.

Pour réaliser ce projet, j'ai commencé par étudier les besoins que spécifie le cahier des charges. Ensuite, j'ai construit une modélisation du système en utilisant le langage UML, au terme de l'étude fonctionnelle du projet. Après, j'ai défini l'architecture logicielle de la nouvelle solution et j'ai choisi les Frameworks techniques nécessaires à la réalisation du projet. Toujours dans l'étude technique du projet, j'ai adopté un Framework de sécurité, un autre de mapping objet/relationnel et un service de journalisation.

Après l'étude technique, la conception de la solution est ainsi élaborée. Dans cette phase, j'ai implémenté le modèle d'analyse obtenu en le rendant extensible et réutilisable, pour répondre aux besoins futurs de l'application. En plus, j'ai construit des solutions qui permettent la factorisation des traitements du système et qui permettent la facilité de son utilisation.

Au niveau de la réalisation, j'ai implémenté les modules de base de l'application et l'administration de ses ressources. Cependant, il reste d'autres modules à étendre pour compléter les fonctionnalités exprimées pour le nouveau système. Durant tout le projet, j'ai veillé à respecter les objectifs requis en terme de qualité.

Ce travail a fait appel aux nouvelles technologies de développement qui sont riches et qui facilitent la traduction des besoins en produits logiciels. Cependant, elles nécessitent un effort considérable pendant la phase de documentation.

Au cours de la période du projet de fin d'études, j'ai eu l'opportunité de mettre en application différentes connaissances acquises durant mes études à l'ENSIAS au profit des besoins exprimés dans le monde professionnel relatif au service bancaire. Par ailleurs, j'ai tiré grand bénéfice de ce stage, aussi bien au niveau informatique

qu'au niveau professionnel. En effet, j'ai pu raffiner mes capacités d'abstraction et de conception. En plus, ce stage m'a permis de raffiner ma méthodologie de travail et de développer mon esprit d'équipe.

En perspective pour ce projet, je vais compléter les modules restants de l'application. En fait, le système doit subir des améliorations au niveau de la présentation en offrant plus de possibilités aux utilisateurs et plus de fonctionnalités d'administration.

Bibliographie

Ouvrages:

[ACHERGUIELMOUTAOUKIL04] A. Achergui & J. Elmoutaoukil, Rapport de PFE « réalisation d'une solution de gestion et de suivi de projet en intranet pour OMNIDATA» ENSIAS 2004.

[GHJV95] E. Gamma, R. Helm, R.E. Johnson, J. Vlissides, Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley 1995.

[HUSTED03] Ted Husted et al: Struts in Action, 2003.

[IBMBOOK] Cours AD65F, WebSphere Application Server – Advanced Edition and EJB, formation IBM.

[JIMCONALLEN00] Jim Conallen, Concevoir des applications web avec UML, Eyrolles 2000.

[MULLERGAERTNER00] Pierre-Alain Muller, Nathalie Gaertner, Modélisation objet avec UML, Eyrolles 2000.

[PATZER00] A. PATZER, Programmation Java côté serveur (Servlet, JSP, EJB), Eyrolles 2000.

[PATZER00] A. PATZER, Programmation Java côté serveur (Servlet, JSP, EJB), Eyrolles 2000.

[ROQUESVALLEE01] Pascal Roques, Franck vallée, UML en action, Eyrolles 2001.

Adresse Internet :

[wwwASTONA]	http://elisa.aston.fr
	Jérôme LOUVEL, Guide d'architecture.
[wwwASTONS]	http://elisa.aston.fr
	Jérôme LOUVEL, Guide des standards.
[wwwCLUBJAVA]	http://www.clubjava.com
	Didier Girard, Processus de développement et nouvelles technologies.
[wwwCROS]	http://www.thierrycros.net
	Concepts et formalismes UML.
[wwwDOUDOUX]	http://www.perso.wanadoo.fr/jm.doudoux/java/
	Développons en Java avec Eclipse (2003).
[wwwFSF]	http://www.fsf.org/
	Définition des différentes licences relatives aux produits Open Source.
[wwwIMPROVE]	http://www.improve-institue.com
	Processus de développement.
[wwwJAVASUN]	http://Java.sun.com/J2EE
	Simplified Guide to the J2EE, Sun Microsystems.
[wwwORACLE]	http://www.oracle.com
	site officiel de Oracle Corporation
[wwwPSTMARTIN]	http://www.pstmartin.freesurf.fr
	Philipe Saint Martin, Cycles de développement
[wwwREDBOOKIBM]	http://www.redbooks.ibm.com
	Servlet/JSP/EJB, Design and Implementation Guide for IBM WebSphere Application Servers.
[wwwTHOMAS]	http://www.psgroup.com
	A. THOMAS, Java 2 Platforme Entreprise Edition.
[wwwWINCOR]	http://www.wincor-nixdorf.com/internet/ma/
	présentation de la société wincor-nixdorf Maroc

Annexe 1 : Cycle de développement en Y

Ce chapitre est consacré à la branche fonctionnelle du cycle de développement. Je vais donc y présente.

Processus Y

Le processus en Y ou Two Track Unified Process (2TUP) est une variante du Unified Process qui s'articule plus sur les aspects techniques. En effet ce processus permet de gérer la complexité technologique, en lui réservant toute une branche de son cycle, dissociée de l'aspect fonctionnel. Il permet donc en plus, de réduire le risque technologique.

Y est de nature itérative et incrémentale et permet, de ce fait, de faire des itérations dans ses différentes phases. Aussi, 2TUP offre un cadre de gestion des risques à travers la déclaration et le suivi d'une liste des risques identifiés.

Le schéma représentant ce cycle de développement est le suivant :

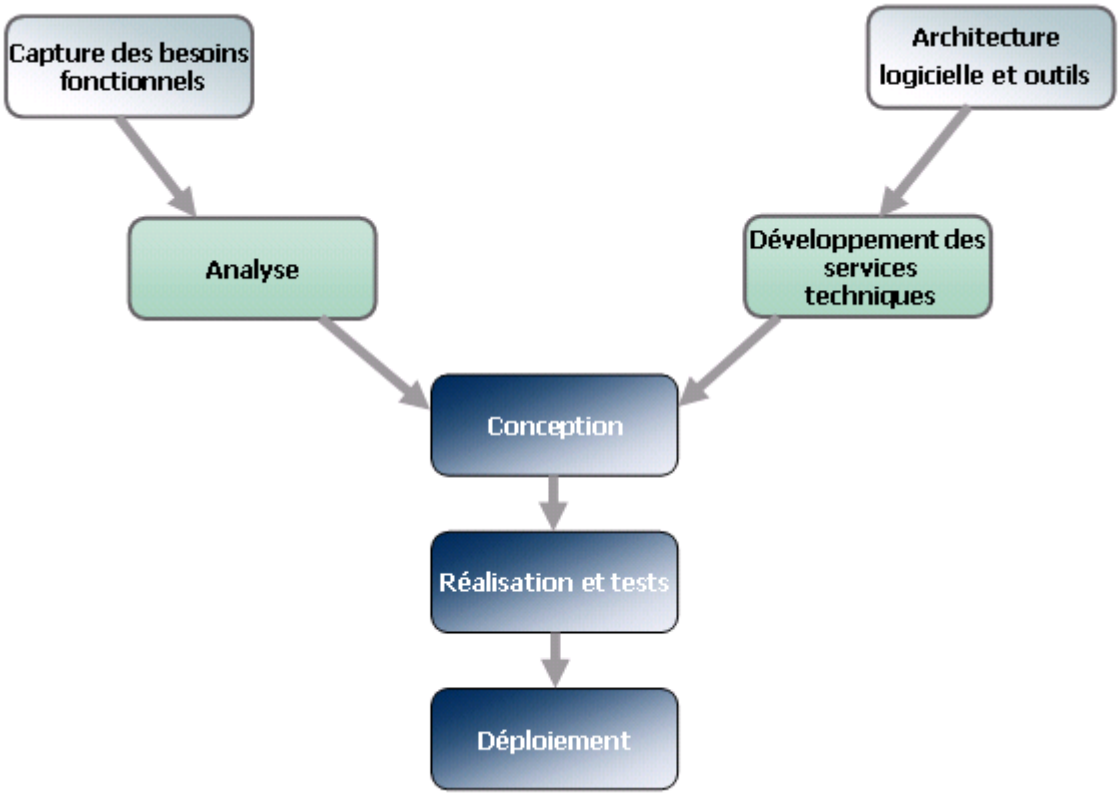


Figure 1 : cycle de développement en Y.

Ce processus se compose de deux branches, une fonctionnelle et l'autre technique. Ces deux branches se rencontrent dans la partie de la réalisation d'où son appellation de cycle en Y.

La branche fonctionnelle est composée de deux phases : capture des besoins fonctionnels et analyse ; l'objectif de la phase de capture des besoins, est de dégager et de modéliser les besoins fonctionnels du projet. Cette activité repose sur le modèle des cas d'utilisation du système et utilise les éléments de modélisation acteurs et cas d'utilisation [wwwCROS]. Il s'agit donc de construire les diagrammes des cas d'utilisation, les diagrammes de séquence et la description textuelle des cas d'utilisation. La construction d'une maquette du système sera de forte utilité par la suite pour représenter les besoins fonctionnels.

La phase d'analyse précise et structure les besoins pour mieux les comprendre et concourir à un modèle plus stable, au moyen de paquetages et des classes d'analyse [wwwCROS]. En effet cette phase, fournit les classes d'analyse, les réalisations des cas d'utilisation (par des collaborations entre objets d'analyse), les paquetages d'analyse et le modèle du système de point de vue analyse. Afin de dégager les classes du modèle, les analystes étudient les livrables de la phase précédente et déduisent une partie du modèle. Ensuite ils raffinent le modèle ainsi construit par l'élaboration des diagrammes de séquence ou de collaboration des objets qui réalisent les cas d'utilisation.

En ce qui concerne la branche technique, les objectifs sont [wwwIMPROVE] :

- rassembler les besoins techniques : sécurité, montée en charge, intégration à l'existant et autres.
- élaborer une architecture logicielle et applicative qui répond aux contraintes dégagées.
- identifier les besoins en Frameworks techniques afin de palier aux manques de la technologie. Exemple : gestion de la touche Back des navigateurs, formulaires de saisies interactives, personnalisation de l'interface graphique, moteur de persistance Objet/Relationnel avec expressions SQL/Objet.

- proposer des règles de développement afin d'industrialiser l'implémentation (gestion des exceptions, règles de nommage, règles de codage, etc.).

Après la branche fonctionnelle et la branche technique, le processus propose la phase de conception.

La phase de conception consiste à reprendre le modèle d'analyse et le refaire selon les décisions prises dans la branche techniques. Il s'agit donc d'adapter son modèle d'analyse à l'architecture adoptée et aux Frameworks techniques choisis et développés. C'est aussi dans cette phase que se fait la conception des modèles d'implémentation à travers le choix des designs pattern appropriés ou la construction de nouveaux modèles.

Après la phase de conception, les développeurs abordent la phase de codage et tests. Il s'agit là de coder les classes obtenues dans la phase précédente. Ensuite, ces classes sont testées unitairement en vue d'être intégrées dans le système tout entier. Une fois construit, le système est prêt pour l'activité de test. Cette activité a pour but de tester le système fabriqué en implémentation. Trois volets sont produits et utilisés :

- cas de test : ce qu'il faut tester dans le système. Ils s'apparentent aux cas d'utilisation.
- procédures de test : la démarche qui permet de dérouler le test.
- composants de test : l'environnement nécessaire pour pourvoir effectivement exécuter les cas de test.

Enfin, la phase de déploiement permet la mise en exploitation du logiciel construit, la formation des utilisateurs qui vont l'utiliser par la suite.

Annexe 2 : Plan Assurance Qualité

Ce chapitre est consacré à la branche fonctionnelle du cycle de développement. Je vais donc y présente.

1. Objectifs et caractéristiques du Plan Assurance Qualité

Le plan d'assurance et contrôle qualité est un document qui précise les éléments permettant de s'assurer de la mise en œuvre et de l'efficacité des activités prévues pour obtenir la qualité requise.

1.1. Objectifs du plan assurance qualité

Le présent plan a pour objectif d'exposer les dispositions prises pour répondre aux exigences spécifiées dans le cahier des charges du projet **ProChangeServer** développé pour WINCOR-NIXDORF. De plus, il permet d'obtenir la qualité du produit requise en contrôlant le planning du projet.

1.2. Domaine d'application

Les dispositions décrites dans ce plan d'assurance et de contrôle de la qualité couvrent le processus de développement, depuis la capture des besoins fonctionnels jusqu'à la mise en production du logiciel.

1.3. Responsabilités de réalisation et de suivi du plan

L'établissement, les mises à jour du plan et le suivi de son application ont été de ma responsabilité. La coordination des actions à entreprendre pour la bonne exécution du plan relève de la responsabilité de l'encadrant de WINCOR-NIXDORF.

Le tableau 1 résume ces responsabilités :

Intervenants	Edition	Validation	Suivi du plan	Application
Najib SAFIR	X			X
Khalid SAADAOUI		X	X	

Tableau 1 : responsabilités de réalisation et de suivi du PAQ.

2. Objectifs et engagements du projet

2.1. Objectifs génériques du projet

Le projet **ProChangeServer** vise la réalisation d'une solution de change des devises étrangères en devise locale. Cette solution doit permettre la récupération des informations envoyées par les GABs effectuant les transactions de change.

2.2. Objectifs qualité du projet

En terme de qualité le projet **ProChangeServer** doit satisfaire les critères suivants :

- disponibilité : le système doit être en permanence à la disposition de ses utilisateurs.
- fiabilité : le système le programme exécute les fonctions attendues avec la précision requise (taux de défaillance minimal).
- évolutivité : il doit être possible d'étendre le système.
- réutilisation : il doit être possible de réutiliser certains modules du système.
- indépendance de l'outil : surtout pour les services techniques, il doit être possible de changer l'outil utilisé.

2.3. Activités d'assurance et de contrôle de la qualité

Je suis tenu de respecter les dispositions décrites dans le PAQ et de vérifier l'adéquation du produit avec les normes en vigueur sur le projet. D'une autre part, le responsable du projet (l'encadrant) veille à ce que ces dispositions soient respectées.

Les activités qualités sont dans le tableau suivant :

Activités qualité	
Type d'activité	**Descriptif de l'activité**
Assurance qualité	élaboration du PAQ
Contrôle qualité	validation du travail
	contrôle de la bonne application des procédures applicables

Tableau 2 : activités d'assurance et de contrôle qualité.

3. Conduite du projet

3.1. Organisation du projet

L'organisation du projet est résumée dans les tableaux :

- **côté WINCOR-NIXDORF :**

Personne	Rôle
Khalid SAADAOUI	chef de projets

Tableau 3 : participants au projet côté WINCOR-NIXDORF.

- **côté ENSIAS :**

Personne	Rôle
Ahmed ETTALBI	professeur encadrant
Najib SAFIR	Stagiaire élève ingénieur

Tableau 4 : participants au projet côté ENSIAS.

D'autre part, un comité de pilotage est constitué et a pour mission de suivre la progression du projet par rapport au planning déjà établi, de modifier le PAQ, de proposer des décisions correctives ou de mettre en place des ajustements.

Ce comité est composé des personnes suivantes :

Personne
Ahmed ETTALBI, ENSIAS
Khalid SAADAOUI, WINCOR-NIXDORF

Tableau 5 : comité de pilotage

Les personnes chargées de la conception et le développement du projet sont les suivants :

Personne
Najib SAFIR, ENSIAS

Tableau 6 : développeur du projet

3.2. Planification et suivi du projet

Un planning prévisionnel a été établi lors de l'élaboration du cahier des charges et réajusté au fur et à mesure de l'avancement du projet (Tableau 7).

Id	Tâche	Date de début	Date de fin	Durée
1	**Etude préliminaire**	**21/02/2005**	**25/02/2005**	**5 j**
2	**Etude Fonctionnelle**	**28/02/2005**	**18/03/2005**	**15 j**
2.1	Capture des besoins fonctionnels	28/02/2005	04/03/2005	5 j
2.2	Analyse	07/03/2005	18/03/2005	10 j
3	**Etude technique**	**21/03/2005**	**08/04/2005**	**15 j**
3.1	Architecture logicielle et outils	21/03/2005	25/03/2005	5 j
3.2	Développement des Frameworks techniques	28/03/2005	08/04/2005	10 j
4	**Conception**	**11/04/2005**	**29/04/2005**	**15 j**
5	**Codage et test**	**02/05/2005**	**01/07/2005**	**45 j**
Total				**95j**

Tableau 7 : planning prévisionnel

Pour le suivi des travaux et de développement, il y a vérification des étapes suivantes :

- l'évaluation de l'avancement du projet.
- le traitement des problèmes et des risques.
- le contrôle de la cohérence des objectifs du projet (plan, coûts, etc.).

4. Démarche de développement

4.1. Cycle de développement

Le cycle de développement adopté pour le projet est le cycle en Y ou 2TUP. Il suit les principes exhibés dans l'annexe 1.

4.2. Description des étapes

4.2.1. Démarrage du projet

Cette étape consiste en la mise en place de l'environnement de développement nécessaire pour le projet.

Activité : appréhender le cahier de charge, réaliser le planning et le PAQ.

4.2.2. Capture des besoins fonctionnels

Cette étape consiste en la détermination de l'ensemble des services que le système est censé fournir à ses utilisateurs.

Activités :

- déterminer les acteurs et les cas d'utilisation du système.
- description textuelle des cas d'utilisation.
- construction de la maquette du système.

4.2.3. Analyse

Cette étape consiste en la construction d'un modèle d'analyse pour le métier du système.

Activités :

- élaborer les digrammes de séquence des cas d'utilisation.

- construction du diagramme de classe d'analyse.
- construction du diagramme de paquetage.
- construction de diagramme d'état/transition pour les classes d'analyse.

4.2.4. Définition de l'architecture logicielle et choix des outils

Le but de cette étape est de définir l'architecture logicielle à adopter et de déterminer les outils à utiliser pour répondre aux besoins de nature technique.

Activités :

- définition de l'architecture logicielle.
- choix des outils.

4.2.5. Développement des Frameworks Techniques

Le but de cette étape est de développer les Frameworks techniques nécessaires à la réalisation des besoins techniques et qui sont indépendants de l'aspect fonctionnel du projet.

Activités :

- développement du Framework de sécurité.
- développement du Framework de mapping objet/relationnel.
- développement du pattern de logging.

4.2.6. Conception

Le but de cette étape est de construire l'implémentation du modèle d'analyse et des couches logicielles de l'architecture choisie. Le Tableau résume les activités et les résultats de cette phase.

Activités :

- construction des diagrammes de collaboration pour raffiner le modèle d'analyse selon le Framework de présentation Struts et l'architecture choisie.
- implémentation du modèle d'analyse.
- construction d'un Framework de factorisation des traitements répétitifs.
- construction du diagramme des paquetages finaux.

4.2.7. Réalisation

Le but de cette étape est de construire les classes résultats de la conception et de les tester unitairement.

Activités :

- codage des modules.
- tests unitaires de ces modules.

Il faut signaler que certaines étapes ont subi un certain nombre d'itérations avant d'aboutir aux résultats réalisés.