Diplôme **U**niversitaire **T**echnologique **S**pécialité **I**nformatique

Période du stage: 05 Avril 2004 – 11 Juin 2004

Responsable pédagogique: Catherine ESPINASSE

RAPPORT DE STAGE

Samuel ROLLET

GDM - Outils de Gestion des Magasins
Migration d'une Application VBA vers une solution Client Léger

CABINET ZULIAN

Ingénierie et Conseils Scientifiques

Le Président, Route de Lyon
01800 Pérouges
Tél.: (+33) (0)4 74 46 74 88
Fax: (+33) (0)4 74 46 08 69

Email : contact@cabinet-zulian.com
Web : www.cabinet-zulina.com

Maître de stage: Franck ZULIAN

Université Claude Bernard – Lyon1

Institut Universitaire de Technologie A

Département Informatique
71, rue Peter Fink - 01000 Bourg-en-Bresse
Tel: 04 74 45 50 50 - Fax: 04 74 45 50 51
Email: info@iutbourg.univ-lyon1.fr
Web : http://iuta.univ-lyon1.fr

SOMMAIRE

SOMMAIRE ..... - 2 -

REMERCIEMENTS ..... - 3 -

IUT : ..... - 3 -
CABINET ZULIAN : ..... - 3 -
AUTRES : ..... - 3 -

INTRODUCTION ..... - 4 -

PRESENTATION DE L'ENTREPRISE ..... - 5 -

CABINET ZULIAN ..... - 5 -
EASY COMPTAGE ..... - 5 -
ORGANIGRAMME ..... - 6 -
FONCTIONNEMENT DE L'ENTREPRISE ..... - 8 -

APPLICATION EXISTANTE ..... - 9 -

MODULE ETAT DES LIEUX ..... - 9 -
MODULE DE GESTION DES INTERVENTIONS ..... - 10 -
MODULE BUDGETAIRE ..... - 10 -
MODULE DE STATISTIQUES ..... - 10 -
MODULE DE PARAMETRAGES ..... - 10 -
AUTRES MODULES EN OPTION ..... - 10 -
LA BASE DE DONNEES ..... - 11 -

OBJECTIFS ..... - 12 -

MOYENS TECHNIQUES ..... - 13 -

LANGAGES ..... - 13 -
LOGICIEL DE DEVELOPPEMENT ..... - 14 -
SERVEURS ..... - 15 -
SGBD ..... - 15 -

TRAVAIL REALISE ..... - 16 -

SYSTEME DE CONTROLE D'ACCES ..... - 16 -
ELEMENTS DE FORMULAIRES ET D'AFFICHAGES DES DONNEES ..... - 18 -
GENERATION DE DOCUMENTS ..... - 25 -
CREATION D'UNE DOCUMENTATION TECHNIQUE ..... - 26 -

ORGANISATION DU TEMPS DE TRAVAIL ..... - 27 -

DECOUVERTE DE GDM & APPRENTISSAGE DES JSP ..... - 27 -
SYSTEME D'AUTHENTIFICATION & CONTROLE D'ACCES ..... - 27 -
CLASSES ELEMENTHTML ..... - 28 -
AFFICHAGE DES DONNEES ..... - 28 -
EDITION ET INSERTION DES DONNEES ..... - 28 -
GENERATION DE DOCUMENTS ..... - 28 -
AUTRES ELEMENTS REALISES ..... - 28 -

CONCLUSION ..... - 29 -

ANNEXES ..... - 30 -

SUJET DE STAGE TRANSMIS PAR L'ENTREPRISE ..... - 30 -
BIBLIOGRAPHIE ..... - 31 -
LEXIQUE ..... - 32 -
SYNOPSIS DES ECRANS DE GDM ..... - 34 -
PRINCIPE DE LA NAVIGATION DANS L'APPLICATION WEB DE GDM ..... - 36 -
HIERARCHIE DES CLASSES JAVA DE GDM ..... - 37 -
JOURNAL DE STAGE ..... - 39 -

Remerciements

IUT :

Catherine Espinasse
Professeur de mathématique au département informatique de l'IUT A, professeur encadrant de ce projet.

Christel Dartigue
Professeur de Java pour les 2nd années au département informatique de l'IUT A.

Cabinet Zulian :

Franck Zulian
Dirigeant de la société Cabinet Zulian, et Maître de Stage pour ce projet.

Sébastien Coste
Ingénieur conseil de la société Cabinet Zulian, et encadrant de ce projet.

Autres :

La communauté open source NetBeans,
Communauté qui a pour objectif de créer un environnement de développement open source équivalent aux environnements professionnels. Ce projet fut fondé en Juin 2000 par Sun Microsystems qui reste aujourd'hui son principal sponsor. Son principal produit est NetBeans IDE qui a été utilisé lors de ce projet.

The Jakarta Project. Communauté de développeur qui crée et maintient des solutions basées sur la plateforme Java. L'un de ses produits est le serveur d'application Tomcat utilisé pour visualiser les éléments réalisés pendant ce projet.

Sun Microsystems
Fondé en 1982 en Californie, Sun, dont la devise est "The Network Is The Computer"™, est aujourd'hui l'un des leaders du monde de l'internet. Sun est notamment le créateur de Java, qui est le langage au cœur de ce projet.

INTRODUCTION

Le Cabinet Zulian, où a été réalisé le stage présenté dans ce rapport, est un cabinet de conseil scientifique et dans le domaine informatique. Il effectue des audits pour ses clients, et développe des applications pour répondre à leurs besoins. Il distribue aussi quelques applications de gestion, la principale étant GDM (logiciel de gestion des points de vente destiné aux grandes chaînes de la distribution). Ce logiciel a été au centre de ce stage, en effet, l'objectif principal étant de faire migrer cette application vers un client Web, entièrement consultable par internet.

Le but du stage de seconde année est de nous mettre en contact avec le monde du travail, cela pour nous permettre d'avoir une première expérience professionnelle et de mettre en pratique toutes les connaissances acquises tout au long de notre formation à l'IUT.

Vous verrez tout au long de ce rapport le déroulement de ce stage, comment les compétences acquises à l'IUT ont été mises en pratique, et l'apport que ce stage a été, aussi bien au niveau des compétences qu'au niveau du vécu professionnel.

La première partie de ce rapport se charge de présenter l'entreprise, son fonctionnement et ses membres. Vous trouverez ensuite une présentation détaillée de l'application GDM existante, sur laquelle le stage demandait de travailler. Apres cette phase de présentation, vous trouverez des informations détaillées sur le travail réalisé pendant ce stage. Tout d'abord, les objectifs fixés par l'entreprise, puis la liste des moyens techniques utilisés aussi bien en terme de langages que d'applications. Ensuite, vous pourrez découvrir le travail réalisé de façon détaillée, et enfin comment a été organisé le temps de travail pendant ces dix semaines.

Vous trouverez de plus en annexe des éléments plus détaillés pour compléter ce rapport et permettre une compréhension plus précise.

PRESENTATION DE l'ENTREPRISE

Cabinet Zulian

Le Cabinet Zulian fut fondé en1994 par Franck Zulian à Grenoble, dans l'Isère. Il s'agit au départ d'une société de conseil dont l'activité est celle d'un bureau d'étude et de gestion de chantiers. Jusqu'en 1997, le Cabinet Zulian vend ses services exclusivement à une grande entreprise de conseil. Après la fin de sa collaboration avec cette entreprise, le Cabinet Zulian est amené à déménager sur Paris, ou il se développera dans le domaine informatique. C'est à cette époque que débute le développement de GDM (Gestion Des Magasins), pour un des clients, le Groupe André. D'autres applications de gestion plus spécifiques sont aussi développées pour répondre à la demande de certains clients.

Dans le même temps, le Cabinet Zulian réalise une étude pour le groupe André sur la mise en place de compteurs clients permettant d'évaluer la fréquentation et la proportion des ventes des magasins. A la suite de cette étude, il développe et met en place ses premiers systèmes de comptage. Cette activité en plein essor est à l'origine de la création de Easy Comptage, qui est présentée ci-après.

Aujourd'hui, les locaux du Cabinet Zulian sont situés à Pérouges, dans l'Ain, et son activité c'est recentré sur le consulting et le développement de logiciels de gestion, depuis la création d'Easy Comptage. Ses principaux clients se trouvent dans la distribution de textiles, de chaussures, les banques et les assurances.

L'activité de Consulting a pour but de réaliser une étude marketing et de performances. Aussi bien lors d'audites de solutions existantes, que pour répondre à la demande d'un client qui cherche le meilleur moyen de réaliser quelque chose.

En ce qui concerne l'activité de développement, elle est centrée sur la réalisation d'applications spécifiques aux produits de chaque enseigne clientes, le plus souvent suite à une étude. L'offre logicielle touchant plusieurs clients se limite à un petit nombre d'outils de gestion très techniques, et ne possédant pas de concurrents directs. On trouve parmi ces programmes un Estimatif Travaux pour la gestion des phases pré-ouverture d'un site, ou encor GDM, destiné aux grandes enseignes pour la gestion de leurs points de vente, et de leurs fournisseurs, pour centraliser les données. Cette application sur laquelle j'ai travaillé est présentée en détail plus loin.

Aussi bien dans son activité de consulting que de développement, l'objectif principale du Cabinet Zulian est toujours de trouver le meilleur moyen pour simplifier l'accès et la gestion des informations.

Easy Comptage

Fort de son expérience acquise dans l'étude des solutions de comptage clients et des remarques des magasins déjà équipés, le Cabinet Zulian décida de développer son propre système de comptage client. Hormis le souci de créer une solution rentable, il en assure une grande fiabilité en gérant tous les problèmes qu'il a pu rencontrer lors de ses audits ; c'est là la force du système créé.

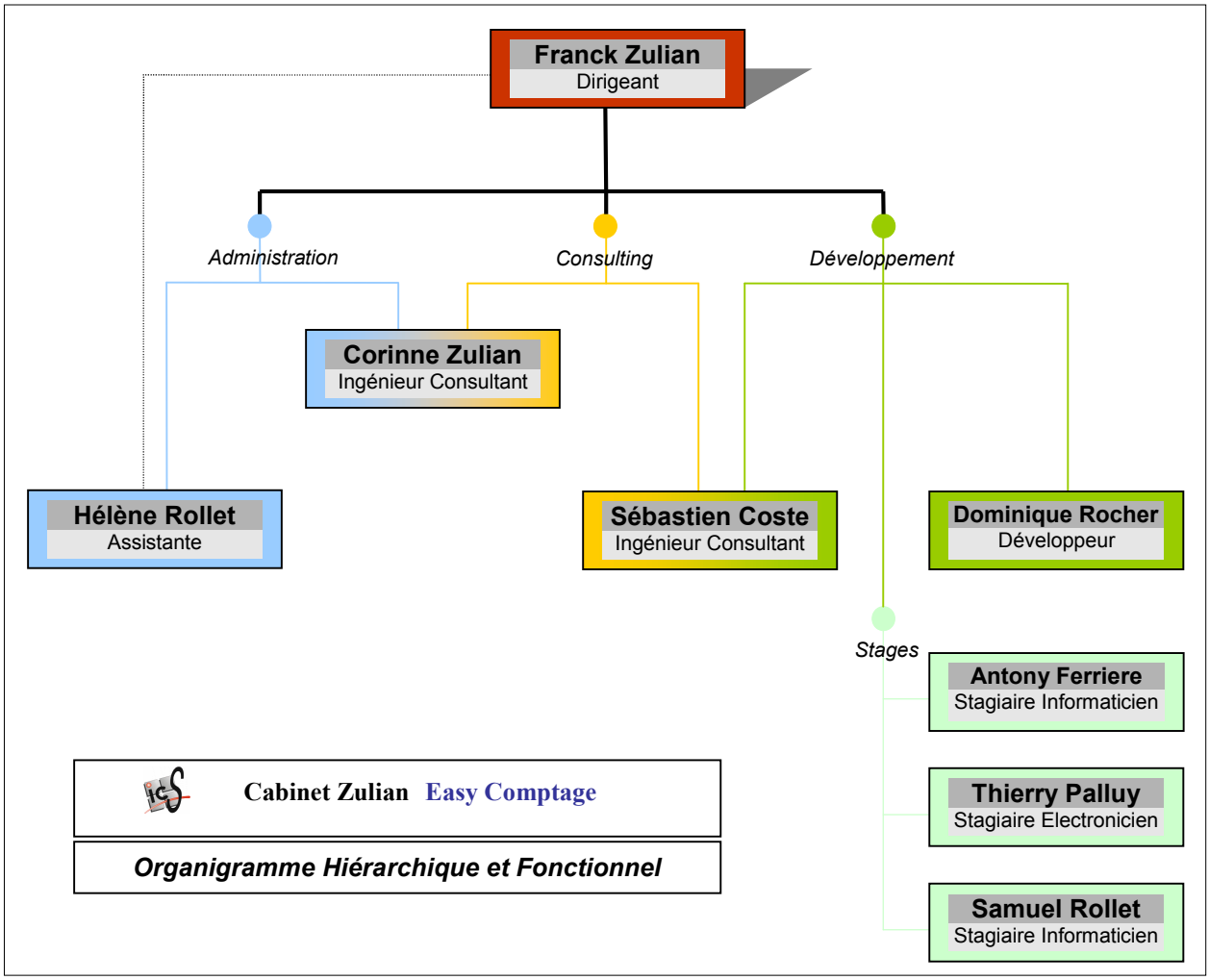
Alors que nombre de ses concurrents directs se contentent de compter les personnes sans contrôle viable, l'activité du Cabinet Zulian augmente, prenant des parts du marché de plus en plus importantes, jusqu'à devenir numéro un français dans le domaine.

L'activité et la charge de travail générées devenant de plus en plus importantes pour le seul Cabinet Zulian qui n'a à l'origine qu'une vocation de consulting ce qui juridiquement se limite à vendre des prestations sous forme d'honoraire, la société Easy Comptage est alors créée en Juillet 2003 pour satisfaire l'activité commerciale initiée par le cabinet. Cette société se dévoue exclusivement à la réalisation de solutions de comptage client. Elle est implantée dans les mêmes locaux que le Cabinet Zulian, à Pérouges.

Easy Comptage, employant une dizaine de personnes, installe entre 500 et 1000 solutions de comptage par an. Ces installations sont sous-traitées par 12 techniciens travaillant pour le compte d'Easy Comptage.

On estime aujourd'hui à 83000 le nombre de boutiques dédiées à l'équipement de la personne, dont 15000 faisant parties d'un réseau succursaliste susceptibles d'être intéressées par une solution de comptage. Actuellement environ 5000 sont équipées, dont près de la moitié par la société Easy Comptage.

Organigramme



Présentation des Dirigeants :

Nom : **Franck Zulian**
Fonction : Fondateur et dirigeant du Cabinet Zulian
Gérant de Easy Comptage
Ingénieur Conseil
Diplôme : Ingénieur Arts & Métiers.
Activité : Management, Consulting,
Marketing, relation avec les nouveaux clients,
Présentation des produits et services.

Nom : **Corinne Zulian**
Fonction : Administration
Ingénieur Consultant
Diplôme : Ingénieur Arts & Métiers
Date d'Arrivée : 2000
Activité : Consulting, Management, Administration,
Recrutement

Présentation des Employés

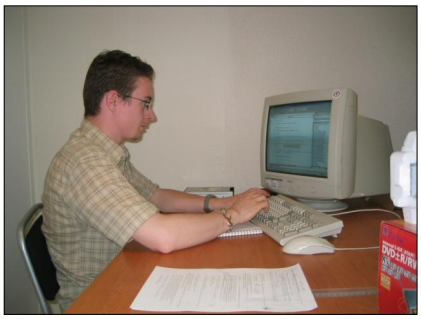
Nom : **Hélène Rollet**
Fonction : Assistante de Direction
Diplôme : Bac Secrétariat
Date d'Arrivée : 6 Novembre 2001
Activité : Administration, Standard.
Suivi des projets, organisation des déplacements.
Premier contact avec les clients.
Planification et diagnostic des installations de comptage.

Nom : **Sébastien Coste**
Fonction : Ingénieur Consultant
Diplôme : Ingénieur INSA dep. Informatique
Date d'Arrivée : Fin 2002, après projet de fin d'études.
Activité : Gestion de projets, Etude des besoins des clients, Consulting,
Développement de modules et d'application clients.

Nom : **Dominique Rocher**
Fonction : Développeur
Diplôme : DUT Mesures Physiques
DUT Informatique
Date d'Arrivée : 2000
Activité : Développement de modules et d'application clients.
Développement d'applications internes de gestion.

Présentation des Stagiaires

Nom : **Thierry Palluy**
Ecole : ENSERG – Grenoble
Etudes : Ingénieur Electronicien,
Systèmes et traitement du signal.
Date d'Arrivée : 1er Mars 2004
Activité : Refonte électronique du système de comptage clients.



Nom : **Antony Ferriere**
Ecole : CCI Rodez
Etudes : Ingénieur Informatique.
Date d'Arrivée : Depuis 2002 en alternance.
Activité : Travail sur les différents projets de développement en cours, depuis 2002 (GDM, Comptage clients…).

Nom : **Samuel Rollet**
Ecole : IUT A Lyon 1, Dep. Informatique
Etudes : DUT Informatique
Date d'Arrivée : 5 Avril 2004
Activité : Travail sur la nouvelle version de GDM en JSP.

Fonctionnement de l'entreprise

La position de mon bureau, dans la pièce principale de l'entreprise, face à l'assistante de Mr Zulian, m'a offert un emplacement privilégié pour pouvoir observer le fonctionnement des deux entreprises que sont le Cabinet Zulian et Easy Comptage.

Etant donné qu'il s'agit d'une petite structure, j'ai tout de suite été en contact avec tous les membres de ces deux entreprises. La bonne entente existant entre les différentes personnes m'a permis de m'intégrer rapidement. Sébastien Coste qui supervisait mon travail, m'indiquait quels étaient les besoins et dans quelle direction orienter mes recherches. Franck Zulian et Dominique Rocher ayant tous deux travaillé avant moi sur GDM, ils m'ont aussi fournit des informations sur son fonctionnement de cet outil. Hélène Rollet, ma 'voisine' de bureau, m'a permis de mieux comprendre le fonctionnement des entreprises et leurs activités. Enfin, la présence d'autres stagiaires nous a permis de partager nos expériences scolaires et nos connaissances réciproques.

L'activité de comptage client étant en développement constant, j'ai pu observer les difficultés que pouvait rencontrer une petite structure en pleine croissance. Notamment les problèmes posés par la sous-traitance des installations de matériels. Easy Comptage n'étant pas l'employeur des techniciens, elle ne peut pas planifier ses interventions directement, ce qui peut entraîner des délais, ou des problèmes de transmission de l'information.

Un autre élément intéressant dans une entreprise en plein développement est la recherche de nouveaux personnels. La quantité de travail gérée par l'activité de comptage client nécessita l'arrivée d'une nouvelle personne dans l'entreprise, une série d'entretiens a donc été mis en place, qui ont aboutit à l'embauche d'une personne. Malheureusement, cette personne ne s'étant pas adaptée au fonctionnement de l'entreprise, elle a décidé de partir. Cela a donc obligé les dirigeants à recommencer le processus de recrutement. Cet événement met bien en lumière les difficultés des entreprises à trouver des personnels leur correspondant.

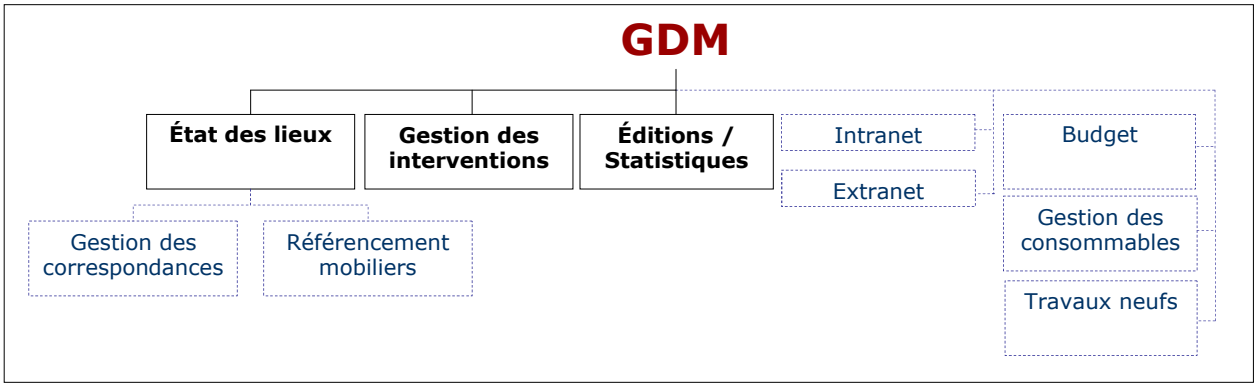
J'ai aussi pu constater les attentes des clients, qui demandent toujours un service irréprochable, pour un prix minimum et dans des délais les plus court possibles.

La force des dirigeants et des employés du Cabinet Zulian et Easy Comptage est de toujours trouver des solutions aux problèmes rencontrés, pour satisfaire les clients. Cela entraîne des plannings en constante évolution, et demande à chacun de s'investir pour répondre positivement à toutes les demandes.

Cette façon de réagir fonctionne particulièrement bien, en effet, aucune publicité n'est réalisée, et les nouveaux clients arrivent sur les conseils des anciens qui sont satisfaits par les services qui leur ont étés offerts.

APPLICATION EXISTANTE

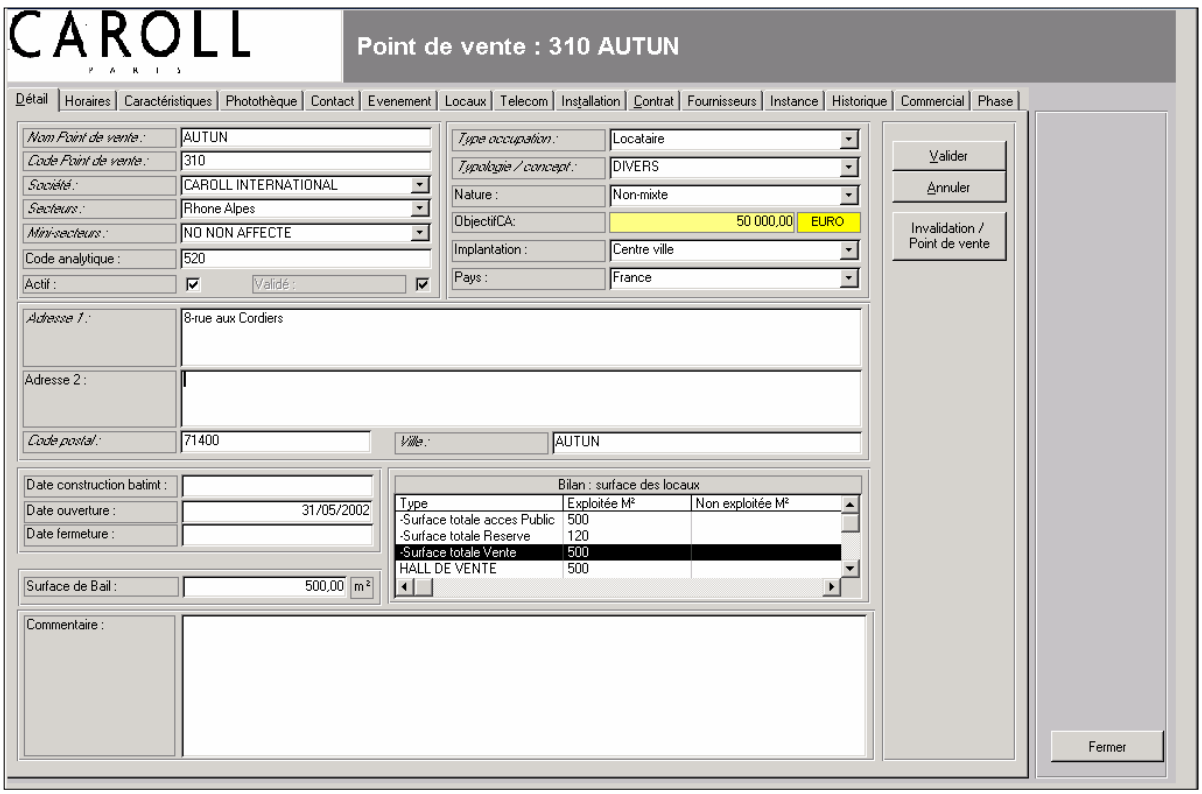
GDM est un logiciel conçu pour apporter aux grandes enseignes tous les outils nécessaires pour la gestion de leurs points de ventes et pour centraliser toutes les données sur ceux-ci. Il est le fruit d'une collaboration avec plus d'une vingtaine d'enseignes tel que le Crédit Agricole, Devred, La Halle… GDM se base sur une application développée en VBA sous Access 97, permettant un accès depuis des clients lourds à l'ensemble des données, mais il offre aussi un Intranet et un Extranet pour l'accès et l'édition des données depuis des poste légers. L'intérêt d'un tel outil est de capitaliser chaque saisie d'informations et de supprimer les fichiers indépendants contenant des informations redondantes et contradictoires, cela, en centralisant toutes ces données dans une base de donnée unique. Il est divisé en plusieurs modules, dont certains en option, que nous allons présenter ci-après.



Listes des Modules de GDM

Module Etat des Lieux

Ce module est destiné à la gestion des points de ventes, des fournisseurs et des contrats. Son objectif est de "stocker dans une seule base de données l'ensemble des informations nécessaires à une enseigne".



Détail d'un point de vente

GDM permet le référencement de chaque point de vente avec de nombreuses informations très détaillées, comme le montre l'écran présenté ci-dessus : coordonnées, matériel installé, caractéristiques, plans et photos, horaires d'ouverture, locaux, événements…

Ce module permet aussi la gestion des fournisseurs, il permet de connaître leurs activités, leurs points de ventes d'intervention, leurs disponibilités…

Enfin, il offre le suivi des contrats entre les points de vente et les fournisseurs. Cela permet de connaître les dates de visites, théoriques et réelles, le résultat de la visite…

Module de Gestion des Interventions

Cet élément répond aux besoins quotidiens des services Travaux et Maintenance. Il permet de "déclencher rapidement des interventions, sans doublon ni incompatibilité, et en assurant un suivi efficace".

L'utilisateur peut effectuer des recherches, consulter l'historique des interventions, connaître celles en cours et effectuer un suivi… Il peut en quelques cliques effectuer une nouvelle demande d'intervention qui est automatiquement transmise par mail ou par fax au magasin et à l'entreprise devant intervenir.

Enfin, il est possible de connaître les interventions en retard, et d'effectuer des relances rapidement.

Module Budgétaire

Ce composant est une option récente de GDM. Il permet de définir un budget général, des lignes budgétaires, et de leur définir des objectifs. Il est ensuite possible d'effectuer un suivi des dépenses réalisées, et de les comparer avec les objectifs. Il est aussi possible de définir les mêmes éléments de manière plus précise pour chaque point de ventes, et ainsi, de pouvoir effectuer des comparaisons.

Module de Statistiques

Ce module permet d'effectuer des analyses sur le fonctionnement des différents éléments des autres modules. Comme par exemple les statistiques correspondant au chiffre d'affaires, la répartition des investissements, ou l'analyse des magasins ou des contrats… L'utilisateur à la possibilité de paramétrer les informations qu'il désire et les conditions qu'il souhaite appliquer pour créer des statistiques personnalisées.

Il est aussi possible d'exporter les informations recueillies dans des fichiers Excel pour qu'elles puissent être diffusées facilement.

Module de Paramétrages

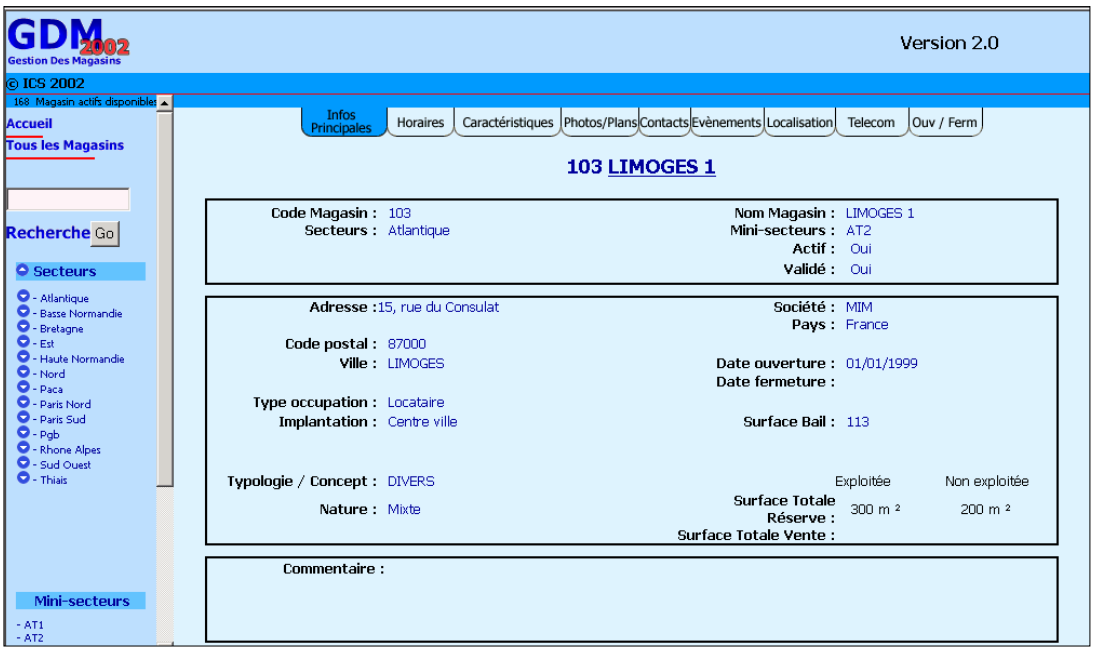
GDM offre une interface totalement paramétrable. En plus des fonctions classiques de gestion des utilisateurs, chaque zone de texte peut être paramétrée, la dénomination de 'Magasin' peut devenir 'Point de Vente'. Chaque libellé peut être adapté aux besoins de l'utilisateur, et le contenu des listes déroulantes des formulaires peut être complété à volonté.

Autres Modules en Option

Parmi les options, la plus importante et l'existence d'un Intranet et d'un Extranet consultables depuis un simple navigateur internet. Développés en ASP, ils donnent accès à une quantité importante d'informations, issues du module Etat des Lieux.

L'intranet, destiné aux services communication, marketing ou commerciaux, a pour objectif de donner un accès de consultation des données pour l'ensemble de l'enseigne.

L'extranet permet l'édition à distances des données, par un accès sécurisé. Il permet aux partenaires de consulter et de mettre à jours les informations pour lesquelles ils disposent des droits d'édition.



Intranet GDM

D'autres options sont encor disponibles, comme le Référencement du Mobilier, permettant de connaître la capacité des magasins en fonction des normes marchandising de l'enseigne ; la Gestion des Correspondances pour les standardiser et assurer une meilleure traçabilité ; la Gestion des Travaux Neufs pour réaliser des estimatifs et assurer le suivi budgétaire des travaux ; et enfin, la Gestion des Consommables, pour simplifier la gestion des commandes et rationaliser leur suivi.

La Base de Données

GDM se base sur une importante base de données. Au départ l'application Access ne fonctionnait qu'avec une base de donnée Access puis elle a évolué pour être compatible avec SQLServeur et Oracle.

La structure de la base est assez complexe, pour permettre de stocker toutes les informations et tous les cas particuliers. Le nombre de table est supérieur à cent, ce qui est due au grand nombre d'informations différentes gérées par GDM.

GDM est donc un outil de gestion très complet, offrant de nombreuses options pour répondre aux nombreux besoins des différents clients. Il est en constante évolution, et de nouveaux modules et/ou fonctionnalités sont ajoutés suites aux demandes des clients.

Aujourd'hui, GDM a besoin d'évoluer vers une technologie plus récente, pour faciliter encor plus l'accès et le traitement des données.

OBJECTIFS

L'objectif de ce stage était de travailler sur la migration de l'application GDM existante, vers une version sous forme de client Web plus performante et accessible plus facilement. (Vous pouvez retrouver le libellé exact du sujet en Annexe 1).

La technologie qui a été choisie est la technologie des JSP (JavaServerPages), langage puissant, basé su le Java, et offrant des possibilités de mise en place de véritables applications Web. En effet, l'exécution de différents composants sur le serveur donne la puissance d'une application à une interface présentée aux utilisateurs sous forme de pages Web.

Bien entendu, étant donné l'importance de l'application existante, une migration complète n'était pas possible en dix semaines. D'autant plus pour une personne seule n'ayant au départ aucunes connaissances de la technologie JSP ni de l'importante base de donnée (plus de cent tables) sous-jacente.

L'objectif pendant ces dix semaines était de mettre en place les éléments nécessaires au fonctionnement générale de l'application : consultation des données, édition, contrôle d'accès, édition de documents papiers. Cela a aboutit à la mise en place de quelques écrans offrant ces différentes fonctionnalités et permettant de tester leur fonctionnement.

Principalement encadré par Sébastien Coste, j'ai donc commencé par découvrir l'application existante, puis le langage JSP, et ensuite, j'ai consacré mon temps à la mise en place de ces différents composants. (Le détail des étapes du stage est présenté dans la partie Organisation du temps de travail).

Tout au long du développement, j'ai cherché à créer des composants les plus modulaires possibles, pour faciliter leur fonctionnement, leur évolutivité, et leur réutilisation. L'utilisation d'un langage objet comme le Java se prêtait particulièrement bien à cet objectif, et le découpage des fonctions de manipulation des données, de contrôle des accès, et d'affichages en différents composants (le modèle MVC présenté plus loin) a permis un découpage encor plus strict.

MOYENS TECHNIQUES

Langages

Le langage Java, développé par Sun Microsystème depuis 1995, s'est imposé comme un standard pour les applications client-serveur et la programmation système, avec une position équivalente au C/C++. Soutenus par une grande communauté de développeurs, il a la particularité d'être disponible gratuitement, tout en restant la propriété de Sun.

Le principal avantage de ce langage est de permettre l'exécution de ses programmes sur toutes les plateformes, grâce à la pré-compilation et à l'utilisation d'une machine virtuelle (JVM) pour exécuter les programmes.

Ce langage objet demande une grande rigueur dans le nommage des éléments et des packages, mais offre au développeur une grande gamme de packages déjà développés et facilement réutilisables.

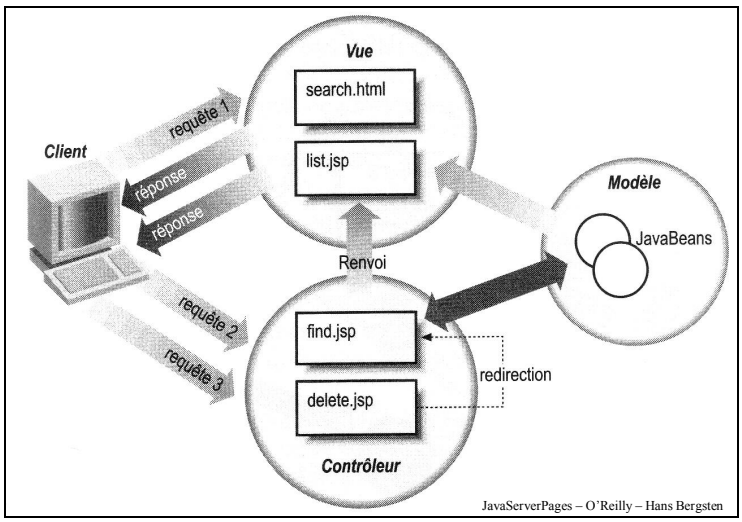
Un élément très utile de ce langage est la commande **Javadoc**, qui permet de construire une documentation au format HTML ou PDF avec une présentation standardisée, simplement en respectant quelques règles simples dans l'écriture des commentaires à l'intérieur du code java. Je l'ai utilisé pour réaliser la documentation de tous les éléments que j'ai développés, afin de permettre aux futurs développeurs qui travailleront sur mon projet de comprendre facilement son fonctionnement.

J'ai utilisé ce langage pour développer de nombreux éléments de mon projet, comme les JavaBeans et les Servlets (voir plus loin) et j'ai pu ainsi approfondir mes connaissances de ce langage que nous avions abordé en cours. Mon projet demandait l'utilisation des JSP, qui sont dérivées du langage Java, et qui lui sont fortement liées.

Les **JavaServerPages** constituent une technologie pour le développement de pages web avec un contenu dynamique.
Les JSP sont des pages contenant des balises statiques HTML classiques et aussi des éléments JSP spécifiques, qui vont être exécutées par le serveur lors de la demande d'une page ; les résultats sont ensuite assemblés à la partie statique, pour être envoyés à l'utilisateur.

Cette approche est similaire à celle de l'intégration PHP/HTML, cependant, en couplant les JSP avec d'autres technologies Java (JavaBean, Servlets), on obtient un fonctionnement plus performant.

Le **modèle de conception MVC** (décrit par Xerox pour le langage Smalltalk) vise à séparer les éléments d'une application en trois unités : le Model, la Vue, et le Contrôleur ; ce qui a pour objectif de séparer le traitement, le stockage et la présentation des données. Ce modèle de conception permet la mise en place d'applications bien structurées, et faciles à maintenir, en effet, les modifications sont indépendantes et plus aisées, aussi bien au niveau du traitement que de la présentation.



Modèle MVC regroupant JSP, JavaBeans et Servlets

En JSP, le rôle du contrôleur est pris en charge par des Servlets qui sont chargés de traiter toutes les requêtes de l'utilisateur. Les Sevlets étant de véritables classes Java, on peut ainsi utiliser toute la puissance de ce langage pour effectuer tous les traitements voulus. Le rôle du Modèle est pris en charge par des JavaBeans, ils prennent en charge la manipulation et le stockage des données dans l'application. Les JavaBeans sont des classes Java normales respectant un ensemble de directive, notamment dans le nommage des fonctions ; les spécifications des JSP permettent une utilisation facile des JavaBeans au travers d'un ensemble de balises standard. Le rôle de la vue est pris en charge par les pages JSP elles mêmes, qui ne se chargent donc que de la présentation des données. Cela permet de limiter la quantité de code au sein même des pages, ce qui les rend plus lisibles, et permet une meilleur évolutivité de la mise en page.

Un autre élément qui fait la force des JSP est la possibilité de pouvoir développer des **Actions Personnalisées**. Ce sont des Bean spécifiques qui permettent d'étendre les balises utilisables dans les JSP, et permettent de localiser un traitement spécifique dans une classe donnée, et de le rendre facilement utilisable dans toutes les pages à l'aide d'une balise personnalisée. Cela permet encor de supprimer des éléments de traitement du corps même de la page JSP.

Tout au long du projet, j'ai donc cherché à réaliser des éléments les plus indépendants les uns des autres et qui pourraient être facilement réutilisables. De plus, j'ai cherché au maximum à éliminer la présence de code dans les pages JSP, cela pour rendre le code des pages le plus compréhensible possible en respectant le modèle MVC.

Les feuilles de style en cascade (CSS) constituent une norme publiée par le W3C, destinées à contrôler la présentation visuelle des pages web. Une feuille de style comprend des règles qui décrivent comment les éléments d'une page, correspondant à des marqueurs HTML spécifiques, doivent être affichés.

J'ai décidé d'utiliser cette technologie pour la mise en page de l'application GDM affin de limiter le code de mise en page dans le code même des pages ; cela limite les répétitions, et facilite l'évolutivité de la charte graphique. De plus cela permet de garantir une meilleure unicité de la présentation dans l'ensemble du site.

Autres Langages :

Au cours de ce stage, j'ai été amené à utiliser d'autres langages. J'ai dus étudier des portions de codes en VBA pour comprendre le fonctionnement de l'ancienne version de GDM qui était développé sous Access et en VBA. J'ai aussi eu quelques contactes avec de l'ASP, ce langage étant utilisé pour l'intranet et l'extranet GDM sur lesquelles j'ai fait quelques tâches de mise à jours, et que j'ai étudié pour m'inspirer de leur fonctionnement.

Logiciel de Développement

Site Officiel : http://www.scintilla.org/

SciTE est un éditeur de texte très utile pour l'édition de code. De taille réduite (<1Mo), et plus puissant que le Bloc-Note. En effet, il permet notamment une coloration syntaxique pour une grande gamme de langages différents (HTML, XML, Java, SQL…).
Il m'a été très utile aussi bien pour l'édition de scriptes SQL, de fichiers de configurations en XML, que de classes java.

Site Officiel : *http://www.oracle.com/tools/*

Oracle JDeveloper est un environnement intégré, développé par Oracle Corporation, permettant aux développeurs d'applications de développer, debugger, déployer, réutiliser et personnaliser des applications "multi-tier" basées sur des composants Java et XML.

Il renferme ADF (Application Development Framework), un environnement de développement complet censé, selon Oracle, simplifier le travail des développeurs Java "de tous niveaux".

C'est l'outil qui m'a été conseillé lors de mon arrivée, mais il s'est avéré très difficile à utiliser, en effet, il présente une interface complexe, et requiert une bonne maîtrise du mode de fonctionnement des JSP, des Servlet, des JavaBean et des EJB pour pouvoir être compris. Etant totalement novice, je n'ai pas pu me familiariser avec cet outil apparemment puissant, mais assez complexe. De plus, l'entreprise ne comportant pas de développeurs java, personne ne pouvait me guider dans cette tâche. J'ai donc opté pour un outil plus classique mais néanmoins performant, NetBeans.

Site Officiel : *http://www.netbeans.org/*

NetBeans IDE est un environnement de développement open source soutenus par Sun. Il fournit une interface complète permettant de développer des applications Java ou JSP, des outils destinés à la gestion de projets, au contrôle des versions et au débogage. Il intègre le serveur Tomcat (voir plus loin) et le JDK 1.4.

Il s'est avéré plus facile à appréhender que Jdeveloper, grâce à son interface intuitive, mais aussi par la grande quantité d'aide disponible sur le net. Il offre tous les outils nécessaires à la mise en place de pages JSP : coloration syntaxique, auto-complétion, mais aussi un débuggeur permettant l'exécution pas à pas des pages JSP.

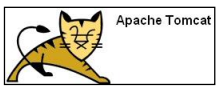
Site Officiel : *http://argouml.tigris.org/*

AorgoUML est un outil de modélisation UML respectant la norme UML 1.3 définie par l'OMG. Entièrement codé en Java, il peut fonctionner sur toutes les plateformes munies d'une machine virtuelle Java. Sa puissance n'est pas comparable à des produits commerciaux comme Rational Rose, mais il offre de nombreuses fonctionnalités très utiles, comme l'exportation vers une base de donnée, ou la génération de code Java pour les classes modélisées. De plus, il est capable de détecter les erreurs dans les modèles, et en informe l'utilisateur.

L'interface rappelle celle de Rational Rose que nous avons utilisée en cours, en plus sobre, l'adaptation a donc été assez rapide.

J'ai choisit d'utiliser cet outil car l'entreprise ne possédait pas de logiciel de modélisation, et qu'il s'agit d'un produit gratuit dont les fonctionnalités étaient suffisantes pour 'utilisation que je devrai en faire.

Serveurs



Site Officiel : *http://jakarta.apache.org/tomcat/passe*

Tomcat est un serveur web totalement écrit en Java. Dans sa version actuelle (5.0), il implémente les spécifications 2.4 des JavaSerlet, et 2.0 des JSP.

Tomcat est développé en open source au sein du projet Apache Jakarta, dont le but est de fournir des solutions serveur basées sur la plate-forme Java, de qualité identique aux applications commerciales.

Il existe de nombreux autres serveurs libres ou commerciaux supportant les JSP. J'ai opté pour Tomcat car il est le serveur de référence utilisé par Sun pour développer les spécifications des JavaSerlet et JSP.

SGBD

Site Officiel : *http://office.microsoft.com/home/*

Access 97 est un logiciel de base de donnée développé par Microsoft. L'application GDM était jusqu'à maintenant un programme développé en VBA sous Access 97, et étant capable d'utiliser une base de donnée Access, SqlServeur, ou Oracle. L'ensemble des écrans de cette version était développé en utilisant des formulaires intégrés à Access. L'un des problèmes posés par une telle architecture est le regroupement des données et de leur traitement, qui va à l'encontre d'un développement modulaire.

Site Officiel : *http://www.oracle.com/database/*

Oracle Database est un SGBDR développé par Oracle Corporation, leader mondial des bases de données. Il permet la définition, la manipulation, la cohérence et la confidentialité des données. De plus, il garantit leur intégrité, et permet leur sauvegarde et leur restauration. Il est fourni avec de nombreux outils d'administration.

L'application GDM permettait déjà l'utilisation d'une base de donnée Oracle. L'objectif était de faire évoluer cette base pour permettre la gestion des nouveaux éléments. J'ai donc été amené très fréquemment à utiliser les outils oracle pour mon projet. Aussi bien pour créer des tables, que pour insérer ou sauvegarder des données. Cela a été facilité par le fait que nous ayons utilisé des bases Oracle lors de nos cours de SGBD à l'IUT.

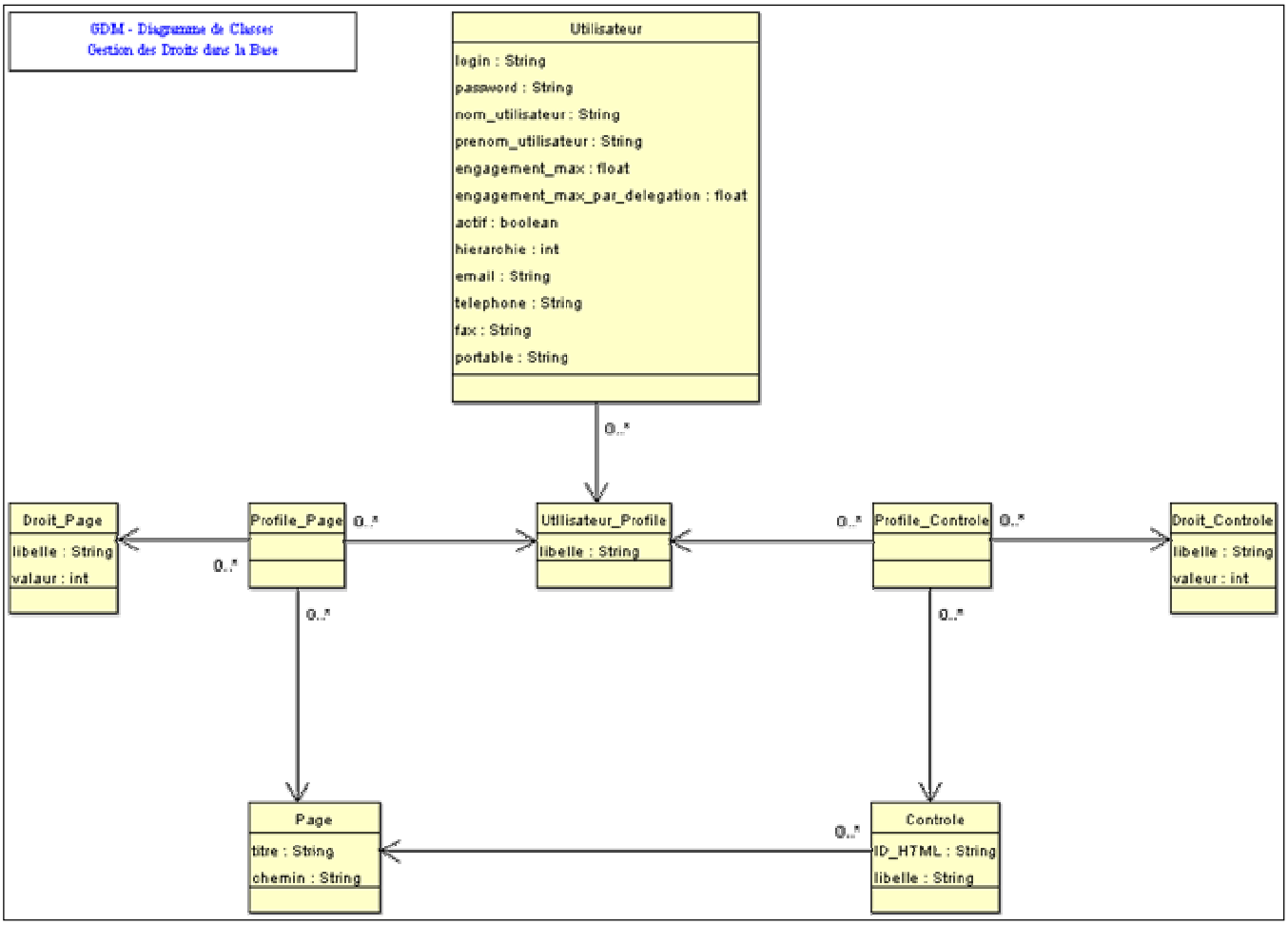
TRAVAIL REALISE

Système de Contrôle d'Accès

Les objectifs de ce système de contrôle d'accès étaient multiples. Il s'agissait de créer les tables nécessaires, dans la base de donnée, pour pouvoir connaître les droits d'accès d'un utilisateur à une page donnée, mais aussi à chaque élément de cette page. De plus, ce système devait être le plus indépendant possible de l'application GDM, pour pouvoir être facilement réutilisable pour une autre application.

Modélisation

Le diagramme ci-dessous, réalisé avec le logiciel ArgoUML, montre la structure des tables nécessaires à la mise en place d'un tel contrôle d'accès.



Modèle des tables nécessaires au contrôle d'accès. (UML 1.3)

La table Utilisateur était une table existante de l'application GDM, elle contenait déjà les éléments login et password, qui étaient utilisés dans l'application Access existante pour contrôler l'accès à l'application. Le fait du passage à une application Web et la demande d'un contrôle plus précis a demandé la création des autres tables pour permettre un contrôle de l'accès à chaque page, et à chaque élément de ces pages. L'objectif d'un tel contrôle étant de limiter l'accès de chaque utilisateur seulement aux données le concernant, et avec des possibilités de consultations ou d'éditions suivant ses autorisations.

Il est rapidement apparu que plusieurs utilisateurs pouvaient avoir les mêmes possibilités de consultations, ce qui a entraîné la création d'une table de profiles (Utilisateur_Profile). Chaque utilisateur appartenant à un profile, et les droits d'accès étant simplement définis pour chaque profil différent, cela limitait la répétition des informations pour des utilisateurs appartenant au même profil.

En ce qui concerne le contrôle d'accès à chaque page, il est possible grâce aux deux tables Page et Droit_Page.

La table Page permet d'identifier chaque page par son chemin, qui correspond à la partie de l'URL de la page, sans le nom de domaine, ni le nom du dossier contenant l'application JSP (cela permet au contrôle de s'effectuer quelque soit le nom de la machine, le nom de domaine, et le nom du dossier contenant l'application Web). De plus, cette table contient le titre de la page, qui est utilisé lors de la génération de la page, pour l'affichage dans le navigateur.

La table Droit_Page définit les différents types d'accès possibles aux pages, de façon littérale, et en leur attribuant une valeur. Dans l'état actuel, les deux valeurs sont Accès autorisé, et accès interdit, mais il est possible de rajouter facilement des types d'accès si un traitement spécifique est nécessaire pour une page ou un groupe de pages données.

La table Profile_Page se charge de faire le lien entre un profile, une page, et le droit d'accès à cette page pour ce profile. Sur ce diagramme, elle aurait pu être représentée par une relation ternaire, mais le logiciel de modélisation utilisé ne permettait pas ce genre de représentation.

On peut donc, en connaissant un utilisateur et la page qu'il demande, obtenir son profile, et ainsi, savoir quel est son droit d'accès.

Le système de contrôle d'accès aux différents éléments de chaque page (appelés contrôles), fonctionne suivant le même schéma, mais avec quelques spécificités.

La table Contrôle permet d'obtenir des informations sur un élément, qui peut être un élément de formulaire (bouton, zone de texte, checkbox…), un tableau, un lien… : ID_HTML est un identifient unique d'un élément, sous forme de chaîne, qui est utilisé dans le code des pages pour identifier chaque élément ; le libellé donne une courte description de l'élément.

La table Droit_Controle a la même structure que Droit_Page, et contient les différents droits d'accès, qui sont à ce jour : non consultable, lecture seule, édition, et édition obligatoire.

La table Profile_Controle permet de faire le lien entre Droit_Controle, Contrôle, et Utilisateur_Profile. Grâce à elle, on sait pour un profile donné, qu'elle est le droit d'accès à chaque éléments d'une page. L'affichage des éléments ne se faissant pas de la même manière suivant le droit d'accès.

Mise en Place

La mise en place s'est fait en deux étapes, tout d'abord dans une base Oracle indépendante, pour pouvoir tester le fonctionnement, puis sur une base GDM sous oracle, qui est une base de test, copie d'une base réel d'un client.

Cette mise en place est passée par la création d'un scripte SQL se chargeant de créer les différentes tables. Il a fallu pour cela utilisé le langage SQL, et aussi le langage PL/SQL spécifique à Oracle.

Utilisation

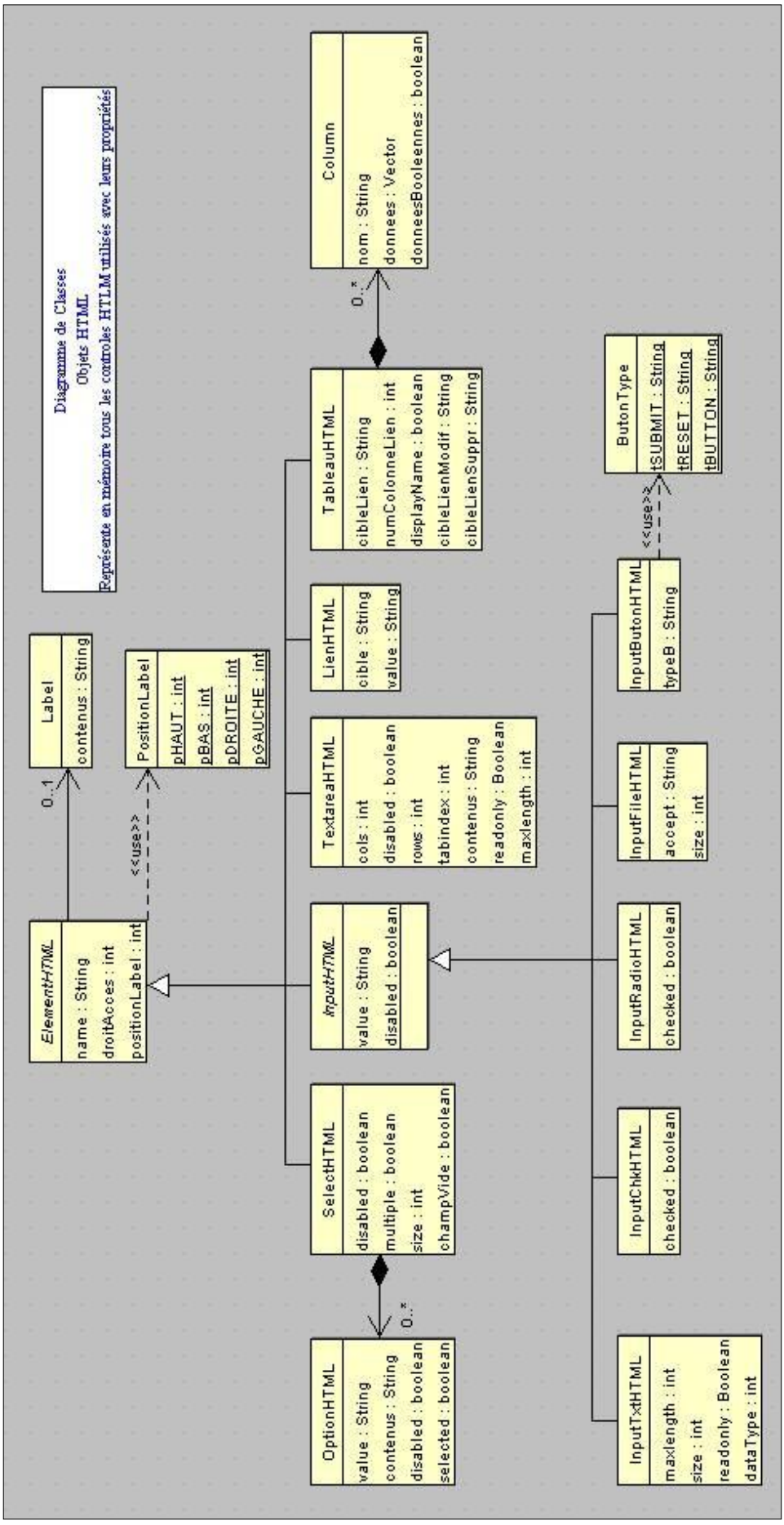
Les informations stockées permettent d'effectuer trois contrôles différents lors de l'accès aux données.

Tout d'abord, l'utilisateur doit s'identifier en utilisant son login et son mot de passe. Le serveur vérifie la validité de ses informations, si il ne retrouve pas les mêmes informations dans la base de donnée, il rejette la tentative d'accès, et renvoie le navigateur sur la même page d'identification. Par contre, si l'utilisateur s'identifie correctement, une session est crée, ou sont mémorisées les informations relatives à cet utilisateur, et utiles pour l'exécution du programme (Nom, prénom, login, identifiant du profile). A chaque requête d'une nouvelle page, la présence de ses informations est vérifiée, si elles existent, le déroulement classique se poursuit, sinon, l'utilisateur est renvoyé sur la page d'authentification, et un message d'erreur lui est affiché. Cela permet de bloquer les utilisateurs qui chercheraient à accéder à une page dont ils connaissent l'adresse, sans s'identifier. Cela permet aussi de prendre en charge les sessions qu sont restées inactives trop longtemps, et qui ont étés supprimées par le serveur.

Le second contrôle s'effectue au niveau de chaque page, pour vérifier si l'utilisateur dispose des droits suffisants pour y accéder. Connaissant l'identifiant du profile de l'utilisateur qui a été mémorisé lors de l'identification, et la page demandée, le serveur vérifie le droit d'accès associé. Si l'utilisateur ne dispose pas du droit d'accès à la page, il est renvoyé vers une page d'erreur l'en informant.

Enfin, le dernier contrôle se charge de vérifier, pour chaque élément de la page référencée dans la base de donnée, comment l'utilisateur peut y accéder (lecture, modification…). Ce contrôle très précis est effectué par les composants présentés après.

Eléments de Formulaires et d'affichages des Données

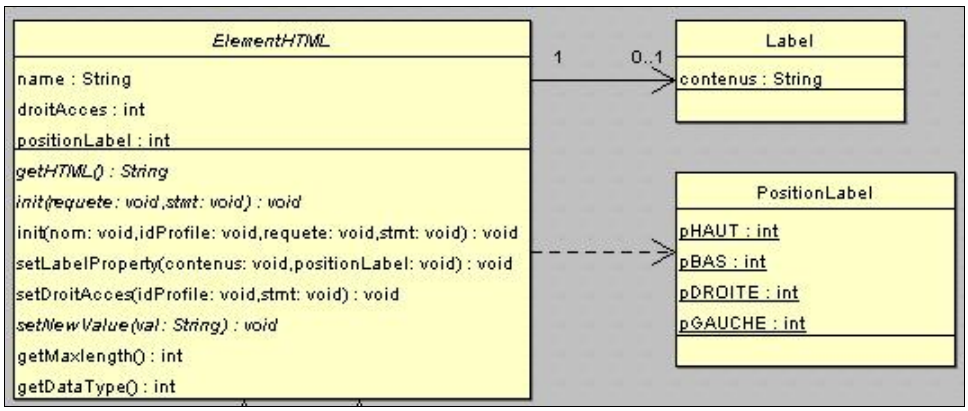


L'application GDM permet la consultation de nombreuses données, mais permet aussi l'édition de ces mêmes données elle comporte donc à l'origine de nombreux formulaire qu'il fallais reproduire dans un environnement d'application Web. L'affichage ne se faisant qu'au format HTML, il était nécessaire d'utiliser tous les éléments de formulaires existants fournis pas ce langage (listes déroulantes, Boites de texte mono ligne et multi lignes, cases à cocher…). La grande quantité de ces éléments nous a poussé à chercher comment généraliser au maximum les traitements qui leur sont associés par le serveur (récupération du contenu, affichage, gestion des nouvelles valeurs).

Le diagramme ci-dessus représente les classes Java qui ont été développées pour simplifier au maximum la mise en classe des nombreux formulaires.

Il montre la mise en place d'une classe par élément de formulaire : Select, Input (text, checkbox, radio, file, button, submit, reset), Textarea ; mais aussi d'autre éléments d'affichage sur lesquels on souhaitait effectuer un contrôle : Tableau, lien.

Les techniques classiques de la programmation objet ont été utilisées pour généraliser les attributs et les méthodes communes, ce qui explique les classes ElementHTML et InputHTML, dont dérivent toutes les autres classes.



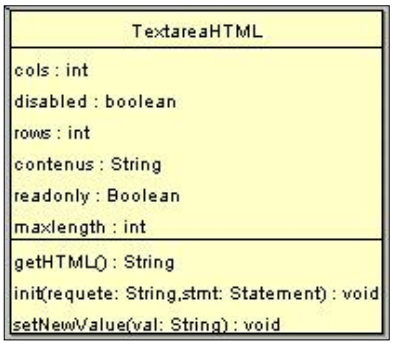
La classe mère ElementHTML est une classe abstraite qui généralise tous les autres éléments. Elle comporte un nom, qui permet d'identifier chaque élément et qui correspond à l'ID_HTML stocké la table Controle de la base de données. Un droit d'accès, qui permet de savoir si l'élément est interdit à la consultation, en lecture seule, en édition, ou encor en édition obligatoire (élément qui doit être obligatoirement être complété lors d'une édition du formulaire). Enfin, chaque élément est composé d'un Label (qui permet d'associer une étiquette à chaque élément de formulaire), et de sa position par rapport à l'élément.

La classe Label possède une structure très simple, la seule caractéristique d'un label étant le texte qu'il affiche. La classe PositionLabel définit des constantes indiquent le positionnement du label. (Une telle classe a été mise en place pour contourner une contrainte du langage Java, qui ne permet pas la définition d'énumération).

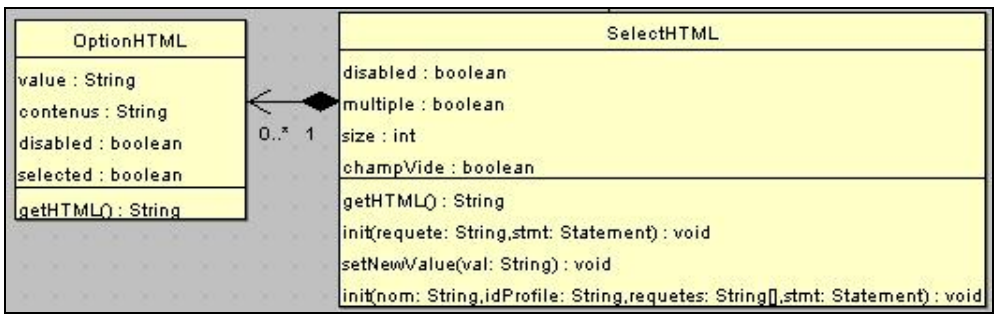
Les méthodes abstraites de ElementHTML définissent les traitements qui doivent être effectués par tous les éléments, mais qui possèdent une implémentation particulière pour chacun.
La méthode *setLabelProperty(contenus,position)* permet de définir le contenu et la position du label.
La méthode *setDroitAcces(idProfile,stmt)* permet à l'élément, lorsque son nom a été défini, de se connecter à la base de donnée (grâce à l'argument stmt, un Statement (élément permettant d'exécuter une requête SQL)), et de récupérer la valeur du droit d'accès dans la table Droit_Controle, qui est associée à son nom, et à l'identifiant du profile qui lui est passé en paramètre.
La méthode *init(nom,idProfile,requete,stmt)* permet d'initialiser la valeur du nom de l'élément, ensuite, elle fait appel à *setDroitAcces()*, en lui donnant l'identifiant du profile idProfile), puis elle fait appel à la méthode abstraite *init(requete,stmt)*, qui, pour chaque élément, effectue la requête SQL qui lui est passée en paramètre, sur la base de donnée avec laquelle le Statement stmt est en relation. Cette requête retourne le contenu de l'élément qui l'exécute. Le contenu et la manière de la traiter étant spécifique à chaque élément, cette méthode est implémentée différemment pour chacun.
La méthode *setNewValue(val)* est une méthode abstraite destinée à la mise à jours du contenus des éléments en passant simplement en paramètre la nouvelle valeur courante. L'objectif est d'éviter l'appel aux fonctions *init()* dont l'exécution complète n'est pas justifiée pour une mise à jours du contenu. Cette méthode est abstraite pour les mêmes raisons que *init(requete,stmt)*.

Enfin, la méthode abstraite *getHTML()* a pour but de retourner sous forme de chaîne de caractères, la code HTML qui correspond à l'élément, en prenant en compte la position du label et le

droit d'accès. Cette méthode est utilisée lors de la génération des pages que visualise l'utilisateur, pour lui afficher les données dans un formulaire HTML classique.

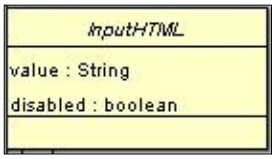


La classe TextareaHTML a pour bute de représenter l'élément de formulaire Textarea (champ de saisie pour l'entrée de texte sur plusieurs lignes). Il hérite de toutes les propriétés et méthodes de ElementHTML, et implémente ses méthodes abstraites. Il possède de plus des attributs qui lui sont propres, qui permettre de connaître : la largeur et la hauteur de la zone de saisie affiché (clos et row), le contenu (contenus), si l'élément est en lecteur seule activé ou désactivé (disabled et readonly). Toutes ses propriétés correspondent aux attributs de l'élément HTML textarea et sont utilisés lors de la génération du code HTML de l'élément; celui-ci ne permettant pas de définir la longueur maximale du texte saisi, l'attribut maxlength, qui correspond à la taille du champ dans la base d'où est issue l'information contenue par l'élément, n'est pas utilisé directement lors de la génération de l'HTML, mais permet d'effectuer un test avant l'insertion de la valeur saisie dans la base de donnée.
Les méthodes implémentées par TextareaHTML sont les méthodes abstraites héritées de ElementHTML. La méthode *init(requete,stmt)* permet de récupérer dans la base de données, en exécutant le requête passée en paramètre, la valeur de l'attribut contenu. La méthode *setNewValue(val)* est destinée à la mise à jour du contenu en passant simplement en paramètre la nouvelle valeur courante. Cette méthode n'effectue pas de traitement particulier dans le cas des TextareaHTML, et est identique à *setContenus()*, mais son existence permet de généraliser le traitement des mises à jours de tous les ElementsHTML, dont certains demandent un traitement spécifique. Enfin, la méthode *getHTML()* retourne sous forme de chaîne, le code HTML de l'élément pour permettre son affichage dans une page pour l'utilisateur. Le code généré prend en compte les valeurs des différents attributs (nom, label, taille…). De plus, lors de la génération HTML, le droit d'accès est pris en compte et produit des résultats différents. Si l'élément est non consultable, le code retourné est un simple commentaire HTML (< !-- -->) indiquant dans le corps de la page que cet élément est non consultable. Si l'élément est en lecture seul, le contenu est retourné sous forme de simple texte, et non sous forme d'élément de formulaire. Si il est éditable, il est affiché en tant qu'élément de formulaire Textarea. Enfin, si il est obligatoire, il est retourné sous forme de Textarea, encadré de balises spécifiques ( ) permettant de le mettre en page différemment les éléments obligatoires grâce à une feuille de style CSS.
Le fonctionnement général de ces trois fonctions étant relativement identique pour chaque élément, le détail de leur fonctionnement ne sera précisé pour les autres éléments que lors de comportement spécifique.

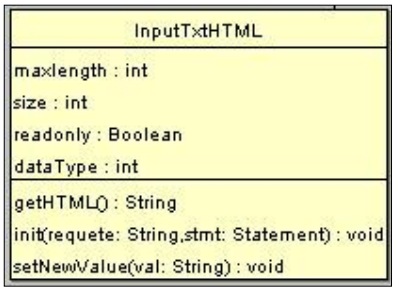


La classe SelectHTML a pour but de représenter l'élément de formulaire Select (menus à choix multiple et listes déroulantes). Elle hérite de ElementHTML. Elle possède des attributs qui lui sont propres, qui permettent de savoir : si ce Select est activé ou désactivé (disabled), si une sélection multiple est possible (multiple), sa taille size (égale à 1, le select sera affiché sous forme d'un menus déroulant, supérieur, sous forme de liste déroulante). Un attribut particulier est champVide, qui ne correspond pas directement à une propriété du code HTML de l'élément, mais qui indique si la liste d'élément du select doit contenir un élément vide, autrement dit, si la liste doit offrir à l'utilisateur de pouvoir sélectionner un élément neutre. Par défaut cet attribut est à faux, les listes ne se composant que des éléments définis par les OptionHTML.
Le SelectHTML peut se composer de plusieurs OptionsHTML. Cette classe correspond à l'élément de formulaire HTML Option, qui est un élément (une ligne) d'une liste déroulante ou d'un menu déroulant. Ses attributs sont : sa valeur, qui est utilisée lors de l'envoi des valeurs d'un formulaire, son contenus, qui est la valeur affichée à l'utilisateur, ainsi que deux éléments permettant de désactiver et de

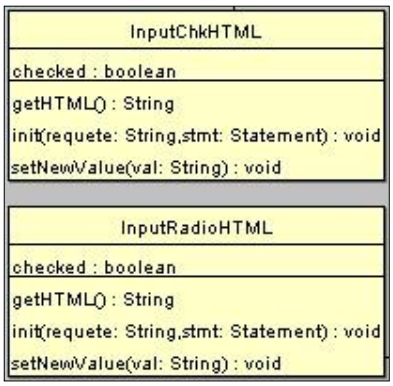
sélectionner l'élément. Sa seul méthode spécifique est *getHTML()* qui retourne sous forme de chaîne le code HTML de l'élément option, ce code prenant en compte les valeurs de chaque attribut. ('<option value ='1' selected>Un contenus</option>',' <option value ='2' disabled>Autre Contenus</option>',…)
Les méthodes implémentées par SelectHTML sont soit des méthodes spécifiques, soit des méthodes héritées de ElementHTML. Parmi les méthodes abstraites héritées, on trouve tout d'abord *init(requete,stmt)*, qui va récupérer dans la base de donnée son contenu (valeur et contenus de tous les OptionHTML). La fonction *steNewValue(val)* se charge de mettre à jours l'élément sélectionné dans la liste, en recherchant parmi les OptionHTML celui qui a la même valeur que l'argument val. La fonction *getHTML()* a un comportement classique, identique à Textarea.
SelectHTML possède de plus deux fonctions d'initialisation spécifiques, qui prennent deux requêtes SQL au lieu d'une pour les autres éléments. La première requête à pour but de récupérer l'élément sélectionné de la liste, et la seconde, le contenu totale de la liste. Cela permet de construire un élément Select avec la bonne valeur sélectionnée.



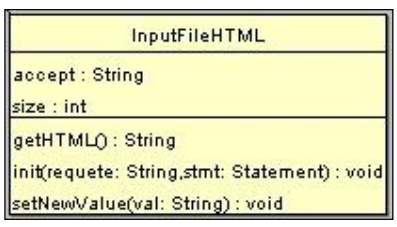
La classe InputHTML est une autre classe abstraite, qui hérite de ElementHTML, et qui généralise les attributs de tous les éléments de formulaire de type Input (boutons, zone de texte…). Elle comporte deus attributs spécifiques : value, attribut existant dans tous les éléments de Input, et disabled, qui précise si l'élément est actif ou grisé.



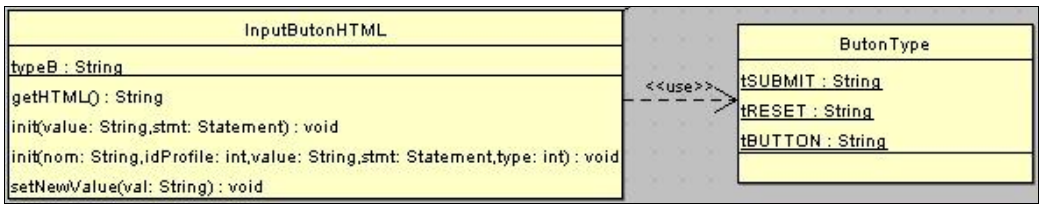
La Classe InputTxtHTML hérite de InputHTML, et représente un élément de formulaire HTML <input type='text'… qui est un élément de saisie de texte. Ses attributs permettent de connaître la longueur maximale de saisie, la taille affichée, et si l'élément est en lecteur seul. Un attribut supplémentaire permet de connaître le type des données sources dans la base de données, cet élément peur être utilisé pour formater les données d'un type particulier (Date…), lors de l'affichage ou de l'édition de la base de donnée.
Les trois méthodes de traitement abstraites héritées sont implémentées. *init(request,stmt)* n'as pas de fonctionnement particulier, à l'exception de la récupération du type de données de la colonne source dans la base de donnée, et du formatage au format dd-mm-yyyy des informations qui sont stockés sous forme de dates. La fonction *setNewValue(val)* n'as pas de comportement particulier, et *getHTML()* génère le code de l'élément sur le même principe que TextareaHTML.



Les deux classes InputChkHTML et InputRadioHTML ont une structure semblable, et représentent les éléments de formulaire checkbox et radiobouton (case à cocher et bouton radio). Leur seul attribut particulier indique si l'élément est sélectionné. La fonction d'initialisation se charge de regarder dans la base pour savoir si l'élément est sélectionné ou non. La fonction *setNewValue(val)* modifie l'état (cochée ou non). Enfin, la fonction *getHTML()* a un comportement classique. Une particularité de InputChkHTML est que sa valeur est toujours à -1 (vrai dans la base de donnée GDM), pour cet élément, l'important n'est pas la valeur, mais le fait qu'il soit coché ou non. Plus tard, une évolution pourra sans doute permettre de grouper les RadioButon, pour pouvoir afficher facilement des groupes de boutons radio.

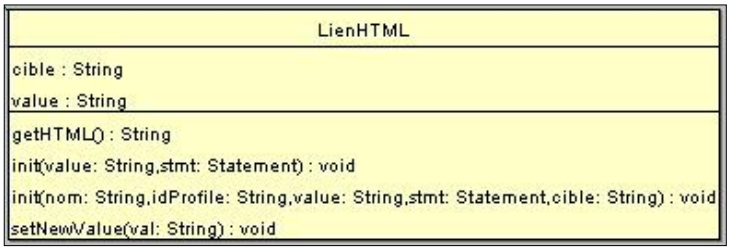


La classe InputFileHTML représente un élément de formulaire permettant de soumettre de fichiers. Le principal attribut indique la liste des types de fichiers acceptés (accept).
La fonction *getHTML()* a un comportement classique. *init(request,stmt)* ne se connecte pas à la base de donnée, et modifie simplement la valeur affichée dans le bouton. Cette même opération est effectuée par *setNewValue(val)*.

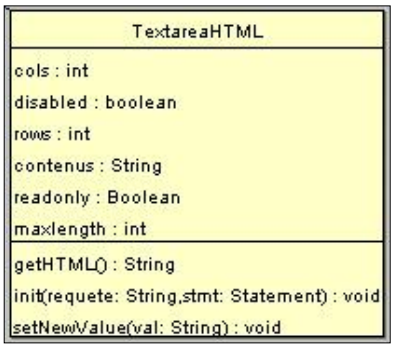


La dernière classe héritant de InputHTML est InputBoutonHTML. Elle correspond aux éléments de formulaire s'affichant comme des boutons, et permettant : de soumettre un formulaire, de le remettre à zéro, ou d'effectuer une action personnalisée (les types d'input correspondant sont : submit, reset, button). Le seul attribut particulier indique le type de bouton ; les différents types étant définis par la classe ButonType, qui liste les différents types suivant le même principe que PositionLabel.
La fonction *getHTML()* a un comportement classique. *init(request,stmt)* et *setNewValue(val)*ont le même comportement que dans InputFileHTML. Il existe en plus une fonction d'initialisation spécifique, qui permet de préciser le type du bouton en plus des éléments habituels.

Deux autres éléments, qui ne sont pas des éléments de formulaires font partie des classes développées. Cela principalement pour pouvoir effectuer un contrôle sur leur affichage.



LienHTML, qui représente un simple lien dans une page, hérite de ElementHTML. Ces deux attributs spécifiques indiquent la cible du lien, et la valeur affichée pour ce lien. La méthode *getHTML()* a un comportement classique prenant en compte les droits d'accès. La fonction *init(request,stmt)* se charge d'initialiser la valeur affichée du lien, et la fonction *setNewValue(val)* de la modifier. Ces deux méthodes n'utilisent pas le statment qui leur est donné. De plus, LienHTML possède une méthode d'initialisation spécifique, qui permet de préciser la cible du lien.

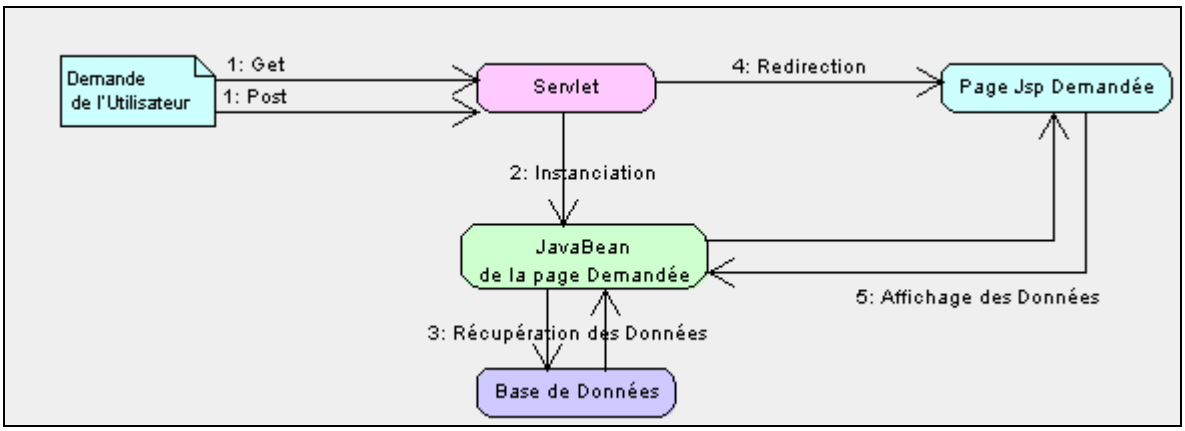


Le dernier élément est TableauHTML. Il permet de connaître toutes les informations sur un tableau HTML, et est utilisé pour l'affichage de données sous forme de grandes listes. Un tableau est composé de Column, renfermant les informations sur chaque colonne du tableau : le nom, la liste des données, et si les données contenues sont sous forme de booléens ; si c'est le cas, lors de l'affichage, les valeurs sont remplacées par oui ou non, pour une meilleure lisibilité.
L'élément TableauHTML offre de nombreuses possibilités. Les attributs cibleLien et numColoneLien permet de définir un lien vers lequel chaque ligne pointera, en utilisant comme attribut de l'URL la valeur de la colonne désignée par numColoneLien. Cela peur être par exemple utilisé lors de l'affichage de la liste des magasins, pour permettre d'accéder à la page de détail des magasins en donnant l'ID du magasin, pour l'affichage correct des informations. De plus, pour chaque ligne du tableau il est possible d'ajouter un lien vers une page de modification des données de chaque ligne (grâce à cibleLienModif), ou d'ajouter un lien permettant de supprimer les informations d'une ligne (grâce à cibleLienSuppr). Un dernier attribut indique si le nom du tableau doit être affiché ou non.
La méthode *setNewValue(val)* n'effectue aucune opération ici, mais on peut envisager de l'utiliser dans le future pour un traitement spécifique. La méthode *init(request,stmt)* se charge de récupérer les informations affichées par le tableau et de les stocker dans les colonnes, de vérifier si il s'agit de données booléennes pour les formater. La méthode getHTML() retourne le code correspondant au tableau, en prenant en compte le droit d'accès : non consultable, le tableau n'est pas affiché, consultable en lecture, le tableau est affiché sans les liens de modification et de suppression, et édition, ou le tableau est affiché avec pour chaque lignes les liens de modification et de suppression si ils ont étés précisés ; de plus, sont affichés : les noms des colonnes, si ils sont précisés, et le nom du tableau si il doit l'être.
TableauHTML possède en plus des méthodes spécifiques, pour indiquer des noms aux colonnes, pour attribuer un nom au tableau qui devra être affiché, et pour spécifier les informations relatives aux liens (affichage du détail, modification suppression).

Structuration du Site et des Packages

Pour rendre l'application Web réalisée la plus modulaire possible, je me suis attaché à mettre en place une structure respectant le paradigme MVC (déjà présenté).

L'affichage des données est entièrement pris en charge par les pages JSP, qui ne comporte pas de code Java, mais juste des balises pour faire appel aux données stockées dans les JavaBean, et aux Actions Personnalisées (Tag), développées spécialement par l'application.



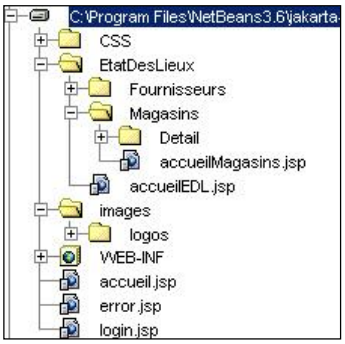
Principe de fonctionnement

La gestion des requêtes de l'utilisateur est gérée par des Servlts. Ceux-ci se chargent de la sécurité (authentification) de la création des JavaBean, et de la redirection de l'utilisateur vers la page demandée. La création d'un seul servlet pour l'ensemble de GDM l'aurai rendu trop important, il a donc été décidé de découper les tâches de contrôle entre plusieurs servlts, un pour chaque module de l'application, cela pour limiter la quantité de code dans chacun et permettre de mettre en place des comportements particuliers à chaque module plus facilement. Les JavaBean se chargent eux de la manipulation et du stockage des données dans l'application. Il existe un JavaBean pour chaque page JSP. Chaque JavaBean contient les informations affichées dans la page, principalement des éléments dérivant de ElementHTML, mais aussi des informations plus simples comme des chaînes de caractères. Quand le JavaBean est crée, il initialise ses informations, en fonction des information sur l'utilisateur courant. Lorsque la page JSP s'affiche, elle recherche le Bean, et récupère les informations qu'il contient.

L'utilisation des actions personnalisées a favorisé la mise en place de certaines opérations à effectuer sur chaque page, comme le contrôle de l'authentification. Si un utilisateur demande une page sans s'être identifié, il est alors redirigé vers la page d'authentification. Cette action est effectuée par une action personnalisée, les pages JSP ne comportent donc que l'instruction *<gdm:controle_authentification />* qui fait appel au Tag, plutôt que le code complet de la vérification, cela pour éviter des répétitions de code.

L'enchaînement des opérations réalisées par le serveur lors du passage d'une page à une autre est le suivant :
Le servlet vérifie si la page demandée existe, si elle n'existe pas, il renvois l'utilisateur sur la page d'accueil avec un message d'erreur, sinon, il instancit le Bean associé à la page demandée, le mémorise de manière à ce qu'il soit accessible par, et redirige l'utilisateur sur la page demandée.
Le Bean se charge de récupérer ses données, en se connectant à la base de donnée, en fonction des informations sur l'utilisateur qui a fait la demande.
Avant de permettre l'affichage de la page, des vérifications sont effectuées. Tout d'abord, une action personnalisée contrôle que l'utilisateur est bien identifié. Si il ne l'est pas, il est redirigé vers la page d'authentification. Sinon, une autre action personnalisée vérifie le droit d'accès de l'utilisateur à la page qu'il a demandé. Si il n'a pas les autorisations suffisantes, il est renvoyé vers une page d'erreur l'informant des restrictions d'accès, et il peut ensuite revenir à la dernière page qu'il consultait.
Si ces deux vérifications sont réussies, la page JSP s'affiche, et présente les informations contenues dans le bean qui lui est associé.
Un schéma présentant cet enchaînement est disponible en Annexe 5 de ce rapport.

Etant donné la grande quantité de pages JSP qui doivent être créées, il était nécessaire de définir une arborescence précise pour les stocker. Il en va de même pour toutes les classes Java développées (JavaBean, Servlets, Action Tags, classes classiques), pour lesquelles l'arborescence de leur package est importante.



La structure du site est assez classique, avec un dossier pour la feuille de style et un pour les images. On trouve à la racine la page d'authentification (login.jsp), la page d'accueil, et la page d'erreur. De plus, comme le montre le dossier EtatDesLieux, un dossier par Modules de l'application devra être mis en place. Pour chaque module, les pages sont séparées dans des sous dossiers, suivant la nature des informations (ici fournisseurs et magasins). Chaque niveau de détail des informations affichées pas les pages est délimité par un nouveau niveau de sous-dossiers.

On visualise en plus un dossier particulier, nommé WEB-INF. Il s'agit d'un dossier utilisé par le serveur. Un de ses intérêts est que bien qu'il se trouve dans l'arborescence de l'application, il n'est pas accessible depuis l'extérieur, mais seulement par le serveur. Il contient trois sous dossiers très important : classes, lib et tlds, ainsi qu'un fichier de configuration, web.xml.

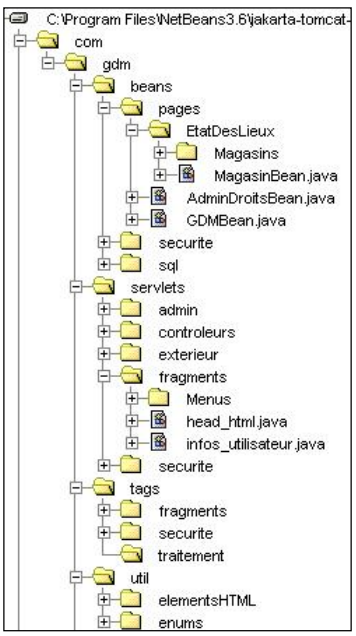
Le dossier classes a pour fonction de contenir les arborescences de dossiers, contenant des classes java qui peuvent êtres utilisées dans l'application Web (servlets, javabean…)

Le dossier lib contient des packages compressés au format .jar. On peut par exemple y déposer le package des drivers de la base de donnée utilisé, ou toutes autres librairies dont les classes sont utilisées dans l'application.

Lib et classes ont donc tous deux pour objectif de contenir des classes, soit dans une arborescence correspondant à la structure du package, soit empaquetés dans des fichiers jar. Le nom de ces deux dossiers est défini par la spécification 2.2 des Servlets.

Enfin, le dossier tlds contient les fichiers de configuration des actions personnalisées (Action Tag) au format XML. Ces fichiers définissent : le nom de la librairie de Tag (gdm ici), la liste des Tags avec leur position, et leurs arguments (obligatoires ou non). Il s'agit du nom de dossier standard, mais il n'est pas défini par les spécifications actuelles.

Le fichier web.xml est un 'descripteur de déploiement'. C'est un fichier XML qui contient toutes les informations pour le serveur lors du lancement de l'application. Il contient notamment des informations sur chaque servlets et les librairies de Tags, pour qu'elles puissent être chargées par le serveur au démarrage. Il contient aussi des paramètres sur les sessions, les pages par défaut, des constantes…



L'architecture du package mis en place pour GDM devait lui aussi être précis. Pour permettre de retrouver facilement les éléments suivant leurs types et leurs utilités. Sous l'arborescence com.gdm, les différentes classes développées sont divisées tout d'abord suivant leur type : beans pour les JavaBeans, servlets pour les JavaServlets, tag pour les actions personnalisées, et util pour les classes java classiques utilisées dans l'application.

Les beans sont ensuite divisés en trois catégories : sql, pour les beans chargés de contenir des connexions aux bases de données, securite, chargé de stocker les informations relatives à l'authentification des utilisateurs, et enfin pages, ou se situent tous les Beans associés à chaque page. Sous page, on trouve une arborescence correspondant aux différents modules de l'application, puis aux différents types d'informations. Les Beans des pages ayant des comportements semblables, certaines de leurs méthodes ont été généralisées. Le bean de base, GDMBean, fournit une méthode de gestion des nouvelles valeurs des ElementsHTML, qui est utilisée dans tous les beans composés de ces éléments.

Dans le dossier EtatDesLieux, on trouve le bean MagasinBean, qui hérite de GDMBean, et généralise le comportement de tous les beans qui contiennent des informations relatives aux magasins. Il implémente une méthode capable d'obtenir dans la base de données le nom et le numéro du magasin courant. Le dossier Magasins contient tous les beans associés aux différentes pages affichant des informations sur les magasins (détail, horaires…) ; ils héritent tous de MagasinBean, et possèdent des fonctions spécifiques chargées d'initialiser le contenu de leurs différents éléments, de gérer les éditions.

Les Servlets sont eux aussi divisés suivant leur utilisées. Les principaux se trouvent sans contrôleurs, il s'agit des éléments chargés de gérer les demandes de pages par les utilisateurs (comme expliqué précédemment, il existe un servlet par module de l'application). Dans le dossier exterieur, se trouvent les servlets qui traitent des demandes particulières, nécessitant des données en provenance d'autres sites, ou renvoyant vers un autre site (affichage de plans de localisation des magasins avec Mappy). Ensuite, le dossier fragment contient des servlets qui sont chargés de créer des portions de pages qui sont communes à plusieurs pages, comme l'entête, l'affichage des infos sur l'utilisateur ou les menus. Ces servlets sont simplement appelés par les pages, et écrivent dans la sortie standard le code HTML qu'ils sont chargés de générer. Cette technique a été préférée à la création de JSP qui auraient contenu le même code et que les pages auraient pus intégrer, car les JSP auraient du se trouver dans l'arborescence du site, et donc accessibles depuis l'extérieur, alors que les Servlets se trouvent dans le dossier WEB-INF, et ne sont donc pas consultables directement. La dernière catégorie de Servlets est la catégorie securite, qui contient les servlets gérant les sessions des utilisateurs, lors de leurs tentatives de connexions, et de leurs déconnexions.

Les Action Tags divisés suivant les mêmes critères. Dans fragments, on retrouve des Tags capables de produire des portions de code. Ces éléments sont souvent liés à l'existence d'autres composants, comme par exemple le Tag BaliseForm, qui génère la balise *<form>* qui indique le début d'un formulaire, il consulte de droit d'accès au bouton de soumission du formulaire, et ne produit le lien de soumission du formulaire qui si l'utilisateur a accès au bouton pour l'envoyer ; ou le Tag LienPlan, qui crée un lien vers le servlet gérant la redirection vers Mappy, uniquement si les informations sur l'adresse sont suffisantes. Les Tags de sécurité, quant à eux, sont chargés de contrôler l'accès aux pages. Ils vérifient si l'utilisateur est bien identifié, et si il dispose des droits d'accès suffisant pour la page demandée ; si l'un de ces deux critères n'est pas vérifié, l'utilisateur est redirigé vers une page d'erreur.

Enfin, les classes Java traditionnelles, situées dans le dossier util, se divisent pour l'instant en deux catégories : les ElementsHTML, qui sont les classes correspondant aux composants présentés dans le chapitre précédent ; et dans enum, les classes définissant des énumérations, utilisées pour faciliter la lisibilité du code (*acces=ACCES_AUTORISE ;* plutôt que *acces=1 ;*).

Il existe donc une grande quantité d'éléments, appartenants à des catégories différentes, chacune avec des possibilités de traitement différents. Toute la difficulté a donc été de comprendre le fonctionnement de ces éléments qui m'étaient inconnus (à l'exception des classes java de base), et de structurer le package de manière la plus logique possible.

Génération de Documents

L'application GDM permet aux utilisateurs de générer des documents, aussi bien des tableaux consultables avec Excel, que des documents de synthèse pour être imprimés ou transmis par fax.

Il fallait donc mettre en place des composants de génération de documents pour une application développée en Java. Le format choisi est le format pdf, qui est un format standard pour la diffusion d'information. Son intérêt est d'être très répandu, d'offrir des documents non éditables pour lesquel on peut limiter les accès, et d'avoir un logiciel de consultation gratuit (Adobe Reader).

La première idée a été d'étudier la structure de tels documents, pour comprendre la syntaxe régissant les éléments contenus dans ces fichiers (textes, images, schémas). L'objectif était de pouvoir ensuite générer des documents à volonté en écrivant directement le code qui les compose.

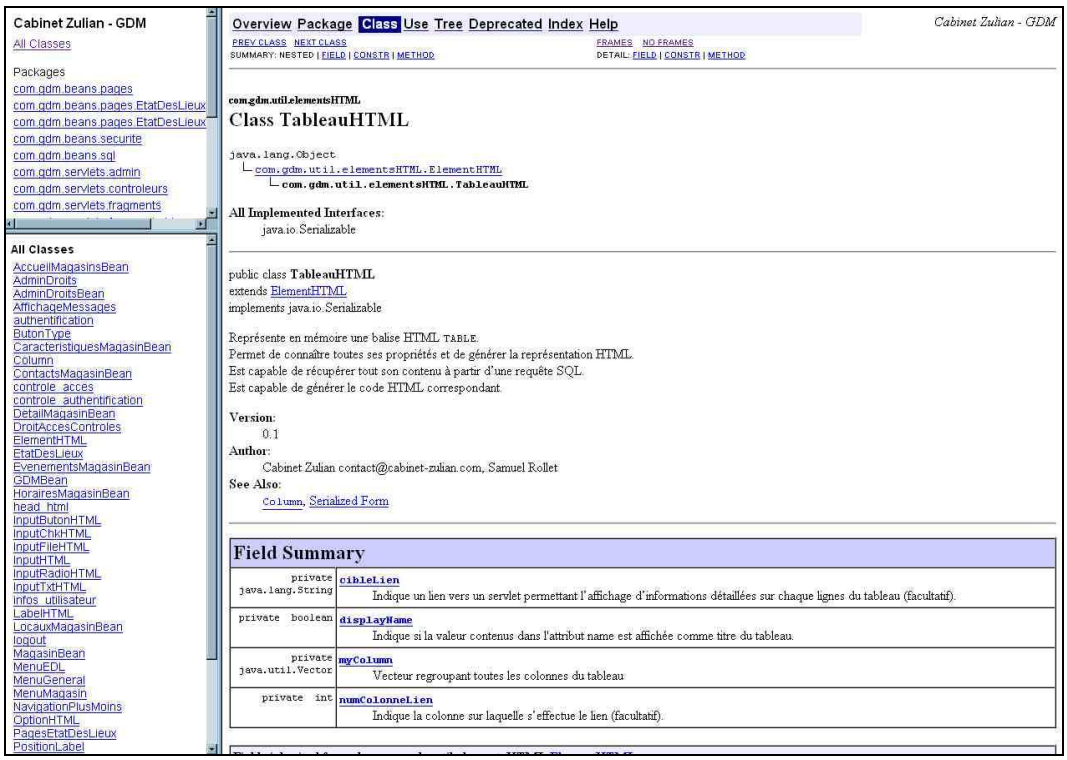
Cette approche s'est rapidement avérée infructueuse, bien qu'il existe une documentation de référence sur ce format, éditée par Adobe, et une librairie d'édition des PDF (Pjx distribuée par Etymon). En effet, la syntaxe particulière utilisée dans las PDF est assez complexe, et les possibilités d'édition assez limitées.

Il a donc fallu rechercher un moyen plus simple pour pouvoir afficher facilement des données en provenance de la base de donnée dans des documents pdf. L'idée a donc été d'utiliser un outil de reporting, qui permet facilement de générer des rapports avec des sources de données différentes, notamment des bases de données. Le premier outil abordé a été Crystla Reports, édité par Business Objects, et qui était déjà utilisé dans l'entreprise avec la plateforme .NET de Microsoft. Le prix de la licence de développement, pour une plateforme java étant assez élevé, j'ai effectué des recherches pour trouver d'autres outils offrant des fonctionnalités similaires. Un élément intéressant semble être JasperReports, qui est un outil de reporting open source permettant la création de documents dans de nombreux formats (pdf, html, csv, xls…). Combiné à iReport, logiciel de mise en page des rapports, qui peuvent être ensuite utilisés à partir de n'importe quel programme java, cette solution apparaît comme répondant aux besoins exprimés. Au jour ou est édité ce rapport, ces éléments sont en cours de test pour comprendre leur fonctionnement et évaluer leurs performances, cependant, ils semblent être des outils assez simples et offrant les fonctionnalités requises pour être utilisés avec GDM.

Création d'une Documentation Technique

N'étant que la première personne à travailler sur ce projet, je devais fournir une documentation détaillée sur tous les éléments que j'ai développés, pour permettre aux développeurs futures qui vont reprendre mon travail de comprendre son fonctionnement.

En plus des diagrammes, qui ne donnent qu'un aperçu général, et des commentaires classiques dans le code, qui ne respectent pas de convention précise, il fallait produire quelque chose de plus conséquent. Pour cela, j'ai utilisé la commande javadoc intégrée à Java. Cette commande, présentée plus haut, permet de produire une documentation complète en respectant quelques règles dans l'écriture des commentaires.



Aperçu de la présentation de la documentation réalisée pour GDM

Il suffit de commencer ses commentaires par /** au lieu de /*, et de les terminer par */. Pour mettre en page certains éléments (italique, paragraphes…) il est de plus possible d'intégrer de balises HTML dans les commentaires. Pour rendre la documentation la plus complète possible, il est nécessaire de commenter chacun des élément du code : classes, attributs, méthodes, arguments. Ensuite, en exécutant la commande javadoc, une documentation au format HTML est produite, sur le même modèle que la documentation officielle produite par Sun pour le java. Il est aussi possible de produire cette documentation au format PDF. Un autre intérêt est que cette documentation est reconnue par la plupart des environnements de développement, qui l'afficha lors de l'utilisation des classes déjà commentées pour faciliter le développement.

Commenter de manière aussi précise toutes les classes développées demande beaucoup de temps, et d'ajouter les commentaires à mesure pour ne pas oublier des éléments. Grâce à cette technique, le fonctionnement de tous les éléments qui composent le package com.gdm (présenté ci dessus) est consultable facilement et de manière détaillée, ce qui devrait faciliter la compréhension du projet pour les futurs développeurs.

ORGANISATION du TEMPS de TRAVAIL

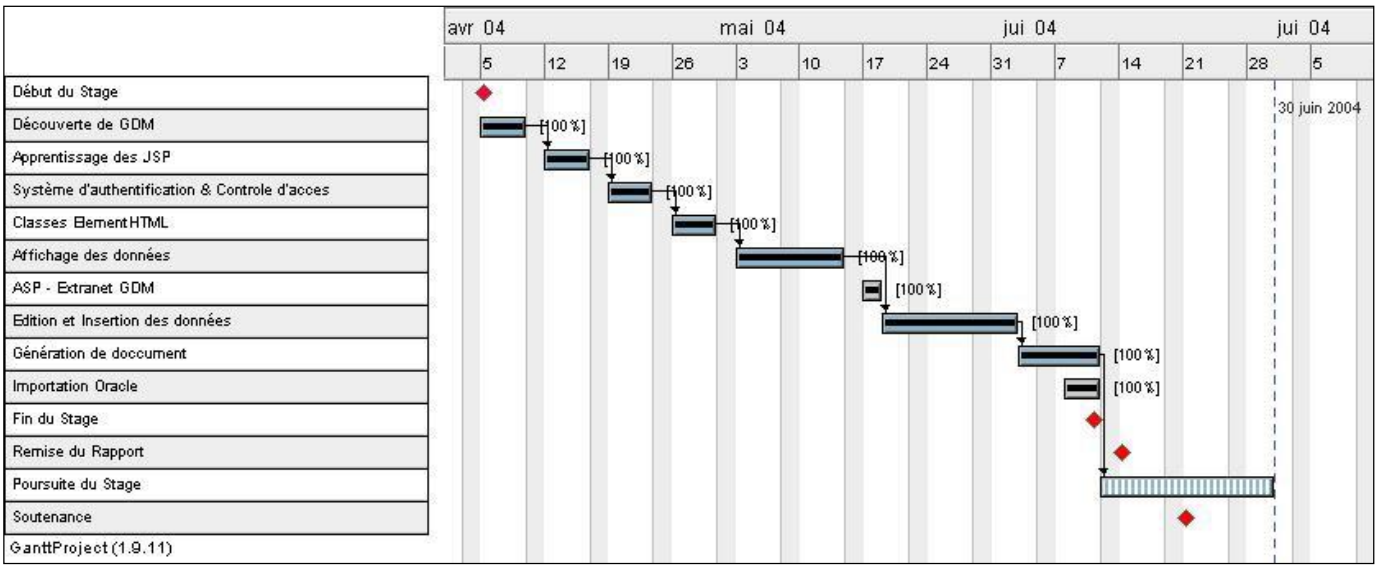


Diagramme de Gantt présentant le déroulement du Stage

Comme le montre de diagramme, mon travail s'est déroulé en plusieurs phases qui ce sont enchaînées, depuis le premier contact avec GDM, jusqu'au développement des principaux composants.

Découverte de GDM & Apprentissage des JSP

Les deux premières semaines ont été dédiées à l'apprentissage des notions nécessaires pour mon stage.

Pendant la première semaine, j'ai consacré mon temps à la découverte de GDM, et à la recherche d'informations sur les outils de développement possibles (JDeveloper et NetBeans). La phase de prise de contact avec GDM a été assez longue, du fait du grand nombre de possibilités offertes par ce logiciel, et des nombreux concepts métiers qui ont nécessités des explications de la part de Mr Zulian et de Mr Coste. Pour connaître en détail les différents écrans, et sur la demande de mon maître de stage, j'ai rédigé un document listant, pour chaque écran, les données affichées et les actions possibles.

La seconde semaine a été consacrée à l'apprentissage des JSP et des spécificités du Java pour les JSP (JavaBean, JavaServlets). Bien que connaissant déjà les bases du Java, j'ai dus apprendre beaucoup de concepts particuliers, que je n'avais jamais rencontrés. Cette phase fut assez difficile, d'abord car je ne connaissais pas les JSP, mais aussi car personne ne connaissait ce langage dans l'entreprise. J'ai donc dû apprendre seul, avec l'aide d'un livre (*JavaServerPages* - O'Reilly) et de cours en ligne disponible sur internet. Pour débuter, j'ai installér un serveur Tomcat, que j'ai du configurer, puis j'ai commencé par visualiser les exemples disponibles avec mon livre. J'ai ensuite créé une page de base, en intégrant tout le code de traitement dans le fichier JSP, puis une page utilisant un JavaBean, pour me familiariser avec les différents composants (JSP, JavaBean, JavaServlets).

Système d'Authentification & Contrôle d'Accès

Après avoir acquis les connaissances de base sur les JSP, je me suis consacré à la mise en place du premier élément utile à GDM : le contrôle des accès aux pages et aux données qui a été présenté en détail plus haut. Cela m'as permis de travailler avec une base de données Oracle, et d'apprendre quelques notions de la gestion de ces bases (Utilisateurs, Schémas…).

Dans le même temps, j'ai installé l'environnement de développement NetBeans, et mis en place les premiers éléments de contrôle d'accès, pour tester le bon fonctionnement. Le principal avantage de NetBeans est d'intégrer le serveur Apache Tomcat, ce qui facilite son utilisation, et d'écrire automatiquement les fichiers de configurations (web.xml et librairie d'action personnalisées).

Classes ElementHTML

L'étape suivante, liée au contrôle d'accès, a été le développement des classes ElementHTML qui ont déjà été présentées plus en détail. Cela est passé par une modélisation avec ArgoUML, pour rechercher la meilleure façon d'organiser les différents éléments, puis par la programmation de ces éléments.

Pour tester le fonctionnement de ces éléments, une première page les utilisant a été mise en place, qui a permis de détecter quelques problèmes qui méritaient d'être corrigés.

Affichage des Données

Après avoir testé le fonctionnement des trois types de contrôles d'accès (authentification, accès aux pages, accès aux données), il a fallu mettre en place le principe d'affichage des données, cela en créant plusieurs pages de l'application, pour pouvoir tester un maximum de cas possibles. Cela est passé par l'affichage de la liste des magasins, et des informations plus précises (détail, horaires, contacts, évènements…).

Dans le même temps, ont étés développés des composants de l'application pour la rendre la plus évolutive possible : menus, éléments de navigation entre les magasins, affichage des messages d'erreur, des titres et sous-titres.

Edition et Insertion des Données

Une fois le fonctionnement de l'affichage des données devenues correctes et les principaux problèmes résolus, l'étape suivante a été le travail sur l'édition des données. Tout d'abord la modification, puis l'insertion. Cette étape a elle aussi demandé du temps, bien que l'affichage des données fonctionnait correctement, il était nécessaire d'ajouter de nombreuses fonctionnalités aux Beans et aux autres classes, heureusement, la structure très modulaire a évité au maximum de devoir modifier les éléments déjà existants se chargeant de la gestion de l'affichage.

Un problème a été rencontré au niveau du codage des caractères entre les éléments java, codé en Unicode, et les informations des formulaires, envoyées codées en UTF. Cela entraînait la perte des caractères accentués lors de l'édition des formulaires, il a donc fallut modifier le jeux de caractères utilisés lors de l'affichage des données dans le navigateur de l'utilisateur.

Une des principales tâches effectuées lors de l'insertion de données est la vérification du format de ces données (types, longueur), et le reformatages quand nécessaire (dates).

Génération de Documents

Apres avoir mis en place le principe de l'édition des données, et supprimer les principaux bugs, j'ai commencé à m'intéresser à la partie de génération de documents. Cette étape est aujourd'hui toujours en cours, mais devrait être facilitée par l'utilisation d'un outil de reporting JasperReports, dédié à la génération de rapports à partir d'informations stockées dans une base de données.

Autres Eléments Réalisés

Durant ces dix semaines, j'ai aussi eu l'occasion de travailler avec d'autres technologies. Comme le montre le diagramme, j'ai travaillé sur de l'Asp, pour effectuer la migration du site extranet existant de GDM vers une version de base de données Oracle avec laquelle il n'était pas compatible. J'ai aussi étudié les possibilités d'exporter et importer des bases de données Oracles. L'objectif étant de pouvoir importer sur le serveur de l'entreprise un base de données GDM d'un client, pour effectuer des tests. Le principal problème à résoudre est un problème de versions, la version locale d'Oracle étant inférieure à celle du client.

Comme on peut le voir sur ce diagramme, mon stage a été prolongé jusqu'à la fin du mois de juin, et je continuerai mon projet en juillet. Cela s'explique par l'importance de ce projet, mais traduit aussi l'intérêt porté par l'entreprise à mon travail.

CONCLUSION

Comme vous avez pu le constater tout au long de ce rapport, de nombreuses difficultés ont été rencontrées pendant ce stage, aussi bien au niveau de l'apprentissage des langages de programmation, que pour effectuer des recherches pour trouver les meilleurs choix dans la conception de tous les éléments. Ce sont ces difficultés qui ont fait l'intérêt de ce stage, en effet, j'ai pu apprendre de nombreuses notions, et appréhender des technologies, comme les JSP, que je ne connaissais pas, et que j'ai trouvé passionnantes.

Cela n'aurai bien entendu pas été possible sans les connaissances acquises tout au long de ma formation suivie à l'IUT, au cours de laquelle j'ai appris de nombreux concepts de base de l'informatique, que j'ai pu mettre en œuvre et approfondir.

Le projet de migration de l'application GDM vers un client Web n'est pas encor achevé, l'importance d'une telle opération n'étant pas possible en dix semaines pour une seule personne ; mais des éléments essentiels, pour le contrôle des accès aux données, l'affichage et la mise à jour de celles-ci sont déjà réalisés. De plus, le principe de fonctionnement de l'application est exprimé avec la mise en place des premières pages de l'application. Je vais poursuivre mon travail sur ce projet jusqu'à la fin Juillet ; pendant cette période, je chercherai à réaliser de nouveaux éléments, et je devrai sans doute revenir sur les éléments déjà effectués pour les améliorer et corriger d'éventuelles erreurs.

Cette expérience a été le premier projet informatique de longue durée sur lequel j'ai pu travailler, et il m'a permis d'approfondir mes connaissances et d'améliorer mes compétences professionnelles ; elle a aussi été un premier contact bénéfique avec le monde du travail, j'ai dû m'intégrer dans cette entreprise où tous les employés se connaissent, et je pense avoir réussit, en étant à l'écoute des demandes de chacun, et en n'hésitant pas à demander des conseils lorsque nécessaire.

ROLLET Samuel

ANNEXE 1
Sujet de Stage Transmis par l'Entreprise

Vous pouvez trouver ici le libellé exact du sujet qui a été transmis par l'entreprise le 6 février 2004 avant le début du stage pour la rédaction de la convention de stage.

Objet : Migration d'une application VBA client/serveur vers une solution client léger

Détail : Dans le cadre de l'évolution de notre produit GDM, nous étudions différentes solutions technologiques, en consultation avec nos clients actuels et potentiels. Parmi ces évolutions, nous souhaitons mettre en oeuvre une version complète de l'applicatif existant sous forme de client web.
Ceci nécessitera une étude des fonctionnalités existantes et de la structure de la base de données pour une refonte accompagnée de nouvelles fonctions. Cette étude sera accompagnée par le chef de projet afin de proposer l'architecture définitive.
Une passerelle devra être mise en place afin d'effectuer la migration de celles des clients déjà installés.
Cette application fonctionnera à la fois sur les bases de type Access, SQL Server et Oracle.
Les derniers choix technologiques (langages et serveurs utilisés...) seront déterminés dans les semaines à venir.
La conception des nouvelles fonctionnalités se fera par prototypage et maquettage, ces fonctionnalités étant directement affinées puis validées par nos clients.

ANNEXE 2
Bibliographie

Livres

Java 2 (le guide du développeur)	P.-Y. Saumont & A. Mirecourt	Eyrolles
JavaServerPages	H. Bergsten	O'Reilly
HTML précis & Concis	J. Niederst	O'Reilly
CSS précis & Concis	E. A. Meyer	O'Reilly
Modélisation Objet avec UML	P.-A. Muller & N. Gaertner	Eyrolles
PDF Reference, fourth edition		Adobe Systems Incorporated
JDBC™ 3.0 Specification		Sun Microsystems, Inc.
Java™ 2 SDK, Standard Edition Documentation		Sun Microsystems, Inc.

Sites Internet

http://java.sun.com/j2se/1.4.2/docs/api/	Documentation officielle Java
http://java.sun.com/products/jsp/	Documentation officielle JSP
http://www.w3schools.com/	Documentation HTML & CSS
http://www.w3.org/	Standardisation HTML & CSS Service de validation HTML & CSS
http://www.laltruiste.com/	Cours de Java et SQL
http://www.oracle.com/	Site officiel de Oracle® Corporation
http://www.netbeans.org/kb/	Documentation officielle de NetBeans IDE

ANNEXE 3
Lexique

ANSI
American National Standards Institute. Organisation de volontaires fondée en 1918 ; regroupant plus de 1300 membres, dont les plus grandes compagnies informatiques, elle a pour bute de créer des standards pour l'industrie informatique. (C ANSI, FDDI…)

API
Application Program Interface, ensemble de routines, de protocoles et d'outils pour développer des applications.

ASP
Active Server Page. Cette technologie propriétaire Microsoft permet de générer dynamiquement des documents HTML. Il s'agit d'une technologie concurrente de JSP.

CSV
Comma Separated Values. Valeurs séparées par des virgules (et par extension, par un caractère particulier). Il s'agit d'une forme primitive, mais très robuste, de base de données, où chaque ligne est un enregistrement où les champs sont séparés par un caractère prédéfini.

IDE
Integrated Development Environmentn, Environnement de programmation intégré à une application.

JDBC
Java DataBase Connectivity, API Java permettant aux programmes Java d'exécuter des requêtes SQL et d'interagir avec une base de données. Similaire à ODBC, mais développé spécialement pour les programmes en Java.

JDK
Java Development Kit, SDK permettant de produire de programmes en Java. Développé par la division JavaSoft de Sun. La version la plus récente intègre l'architecture des composants JavaBeans et le support JDBC.

Métalangage
Langage permettant de définir un autre langage.

ODBC
Open DataBase Connectivity, méthode d'accès aux bases de données standard, développé par Microsoft Corporation.

OMG
Object Management Group, consortium de plus de 700 compagnies. L'objectif de cette organisation est la promotion orientée objet, et de fournir un framework commun pour le développement d'application utilisant des techniques orientées objet. OMG est à l'origine des spécifications Cobra et UML.

Paradigme
Représentation du monde, manière de voir les choses. "Le paradigme objet met en évidence l'organisation fondée sur les comportements et les responsabilités"

PDF
Portable Document Format, format de fichier développé par Adobe Systems pour représenter les documents de manière indépendante de l'application originale, du système d'exploitation, et du matériel. Les documents PDF peuvent contenir du texte des images, des graphismes… Un lecteur gratuit est disponible pour de nombreuses plateformes.

SDK
Software Development Kit, package de programmation permettant de développer des applications pour une plateforme spécifique. Habituellement, un SDK inclut une ou plusieurs API, des outils de programmation et une documentation.

SmallTalk
L'un des premiers langages à objets, conçu selon certains en 1972 par Alan Kay, et selon d'autres en 1980 au Xerox PARC.

SQL
Structured Query Language, langage de requêtes standardisé pour les bases de données. Descendant de SEQUEL crée en 1974 par IBM, SQL apparut en 1979 grâce à Oracle Corporation. SQL fut standardisé par l'ANSI à partir de 1986.

SGBDR
Système de Gestion de base de données Relationnel. RDB ou RDBMS en anglais. Les données sont stockées dans différentes tables mises en relation, ce qui favorise l'accès et le maintient des données.

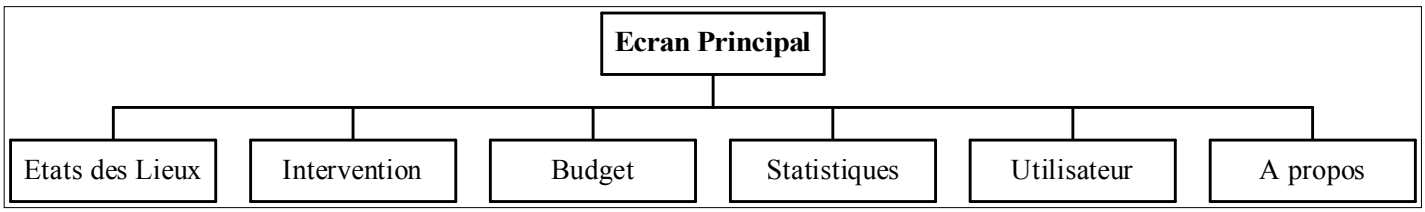
SGML
Standard Generalized Markup Language. Métalangage utilisé pour définir de façon générale des langages définissant des documents hypertextes de toutes sortes, normalisé sous le nom d'ISO 8879. HTML et XML en sont des dérivés.

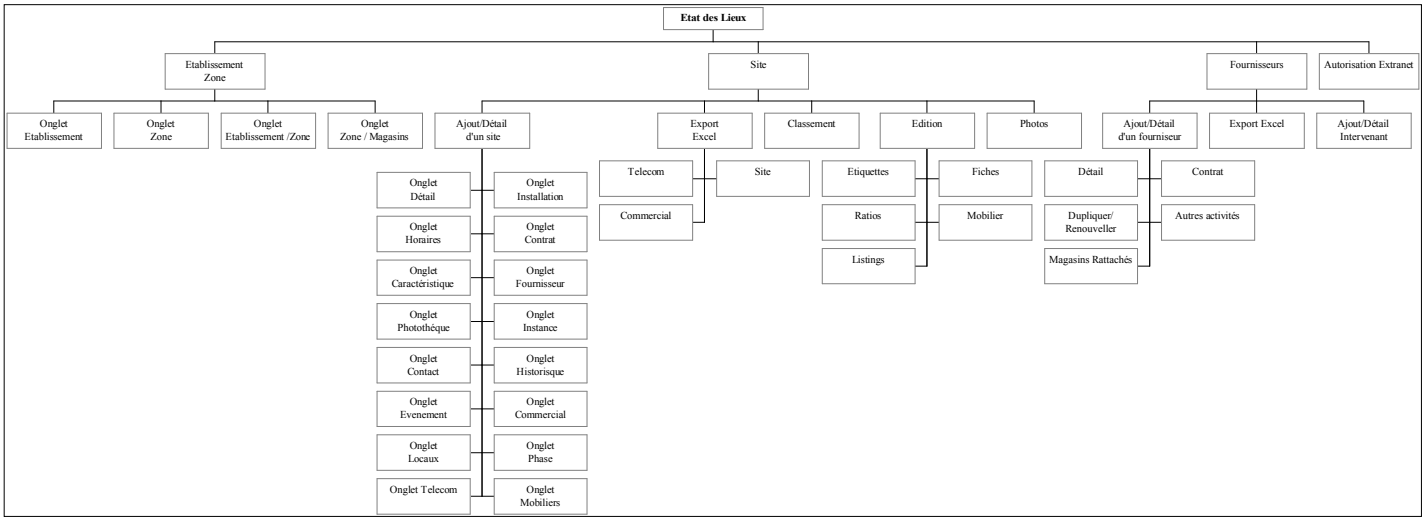
UML
Unified Modeling Language, langage de notation pour la spécification et la visualisation des applications complexes, particulièrement les projets orientés objet. UML se base sur des méthodes de conception plus anciennes.

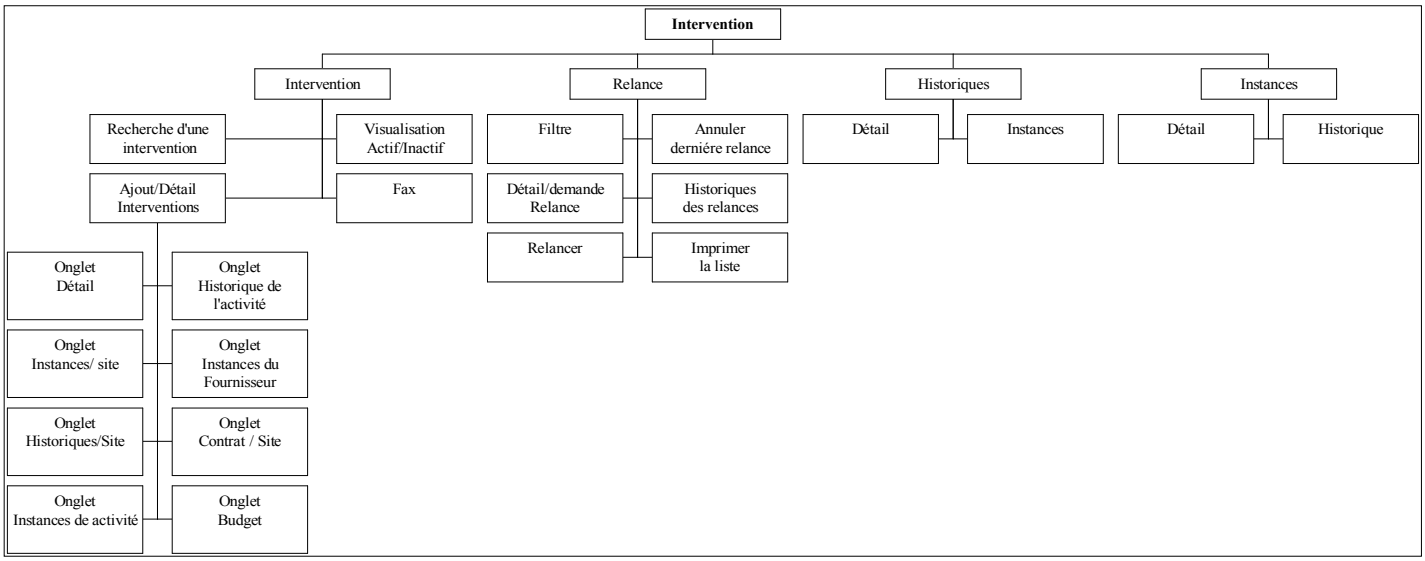
VBA
Visual Basic for Applications. Version de Visual Basic conçue pour automatiser les tâches dans des applications de bureautique.

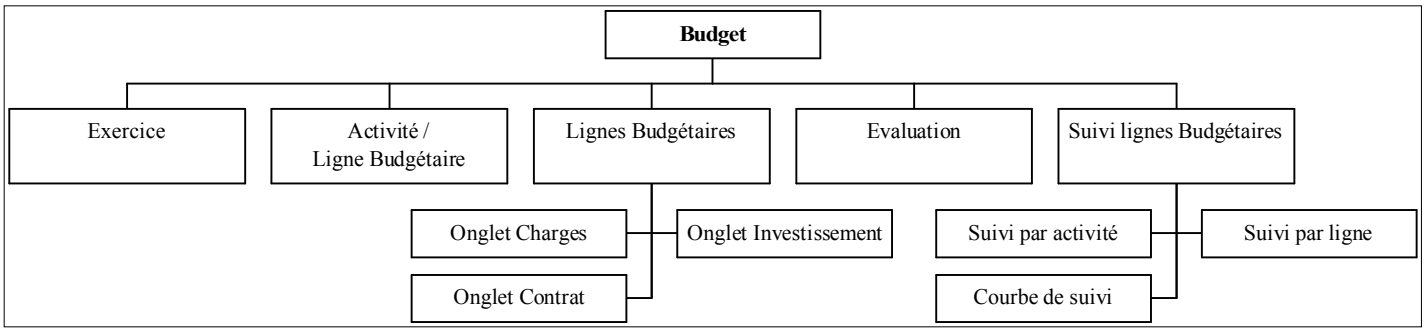
XML
eXtensible Markup Language. Norme d'échange de documents informatisés issue de SGML, grâce au travail du W3C. C'est une version simplifiée de XML qui permet de définir des balises personnalisées en fonction de ses besoins.

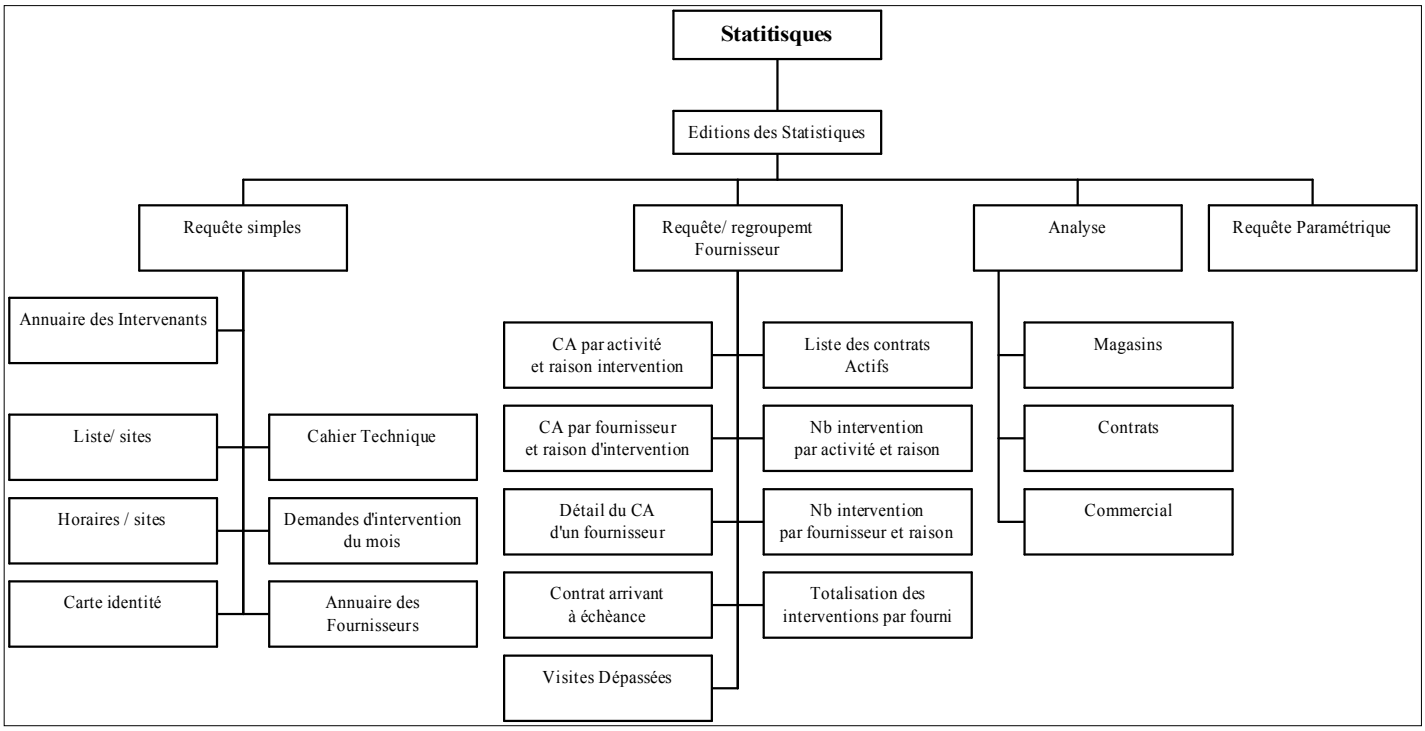
ANNEXE 4
Synopsis des Ecrans de GDM

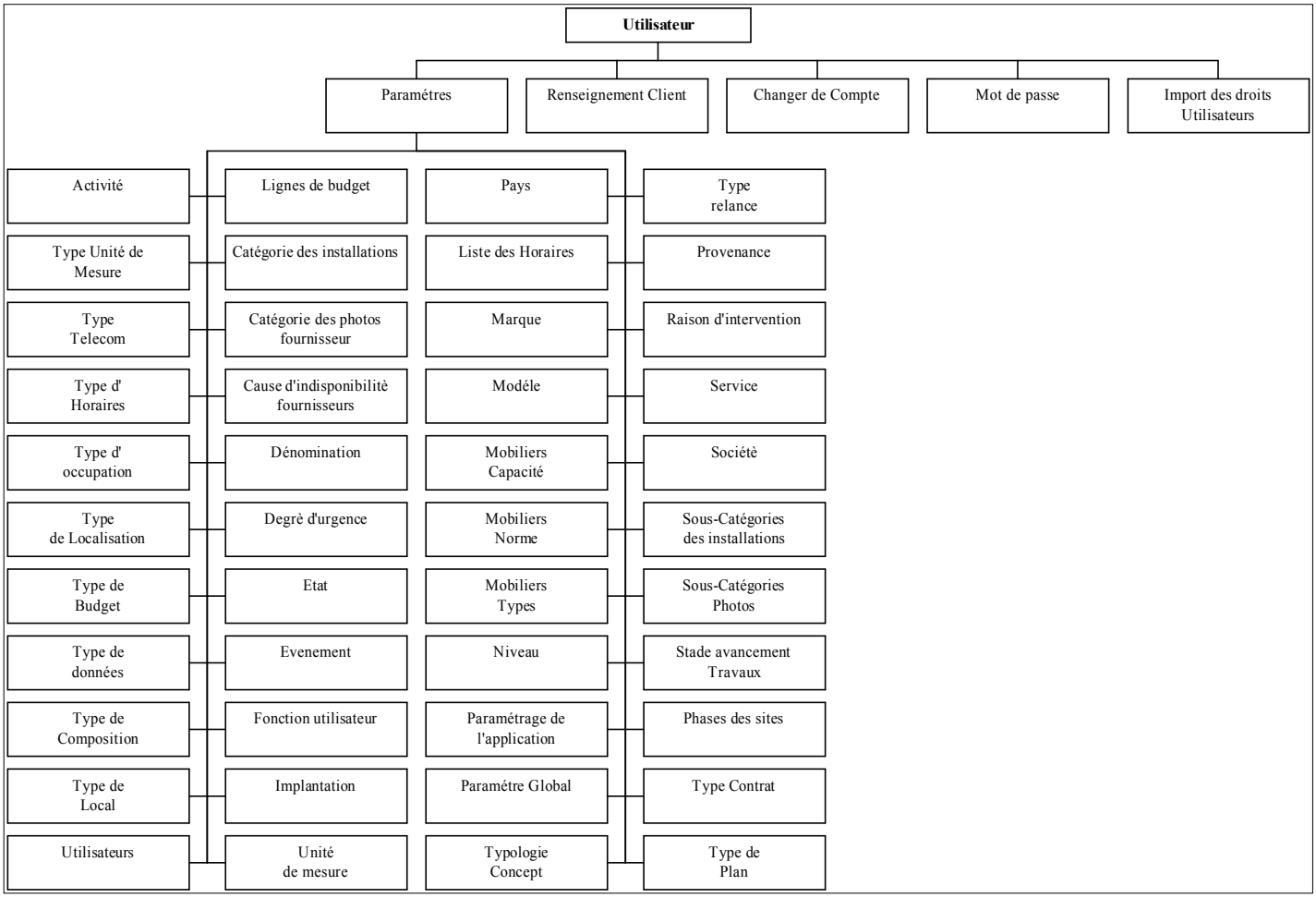




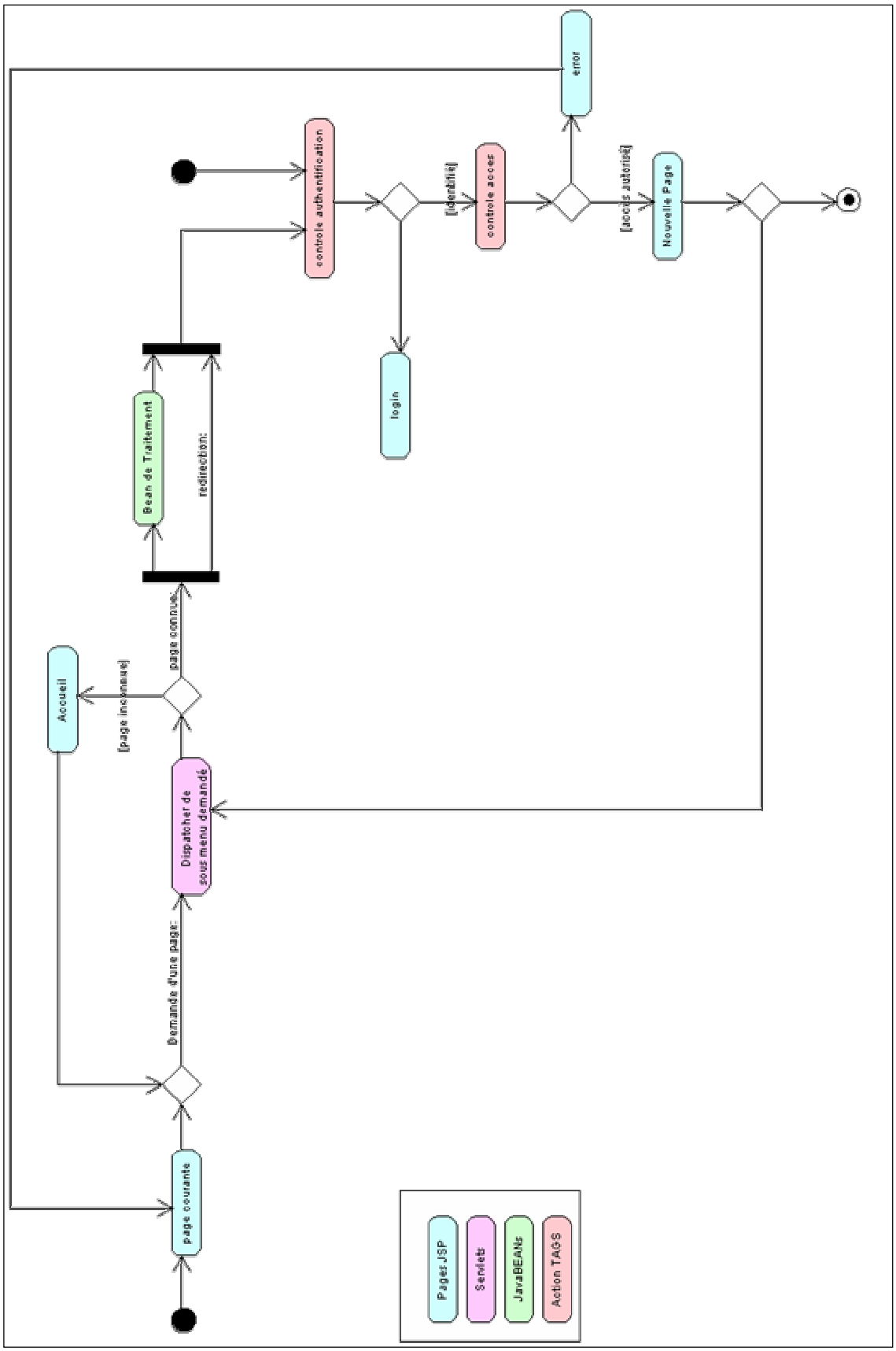





Statitisques
Editions des Statistiques
Requête simples
Requête/ regroupemt Fournisseur
Analyse
Requête Paramétrique
Annuaire des Intervenants
Liste/ sites
Horaires / sites
Carte identité
Cahier Technique
Demandes d'intervention du mois
Annuaire des Fournisseurs
CA par activité et raison intervention
CA par fournisseur et raison d'intervention
Détail du CA d'un fournisseur
Contrat arrivant à échèance
Visites Dépassées
Liste des contrats Actifs
Nb intervention par activité et raison
Nb intervention par fournisseur et raison
Totalisation des interventions par fourni
Magasins
Contrats
Commercial


Utilisateur
Paramétres
Renseignement Client
Changer de Compte
Mot de passe
Import des droits Utilisateurs
Activité
Type Unité de Mesure
Type Telecom
Type d' Horaires
Type d' occupation
Type de Localisation
Type de Budget
Type de données
Type de Composition
Type de Local
Utilisateurs
Lignes de budget
Catégorie des installations
Catégorie des photos fournisseur
Cause d'indisponibilitè fournisseurs
Dénomination
Degrè d'urgence
Etat
Evenement
Fonction utilisateur
Implantation
Unité de mesure
Pays
Liste des Horaires
Marque
Modéle
Mobiliers Capacité
Mobiliers Norme
Mobiliers Types
Niveau
Paramétrage de l'application
Paramétre Global
Typologie Concept
Type relance
Provenance
Raison d'intervention
Service
Sociétè
Sous-Catégories des installations
Sous-Catégories Photos
Stade avancement Travaux
Phases des sites
Type Contrat
Type de Plan

ANNEXE 5
Principe de la Navigation dans l'Application Web de GDM



Note : Ce schéma réalisé avec ArgoUML utilise les représentations d'un diagramme d'activité UML, mais il ne s'agit pas d'un diagramme d'activité, il a juste pour objectif de présenter les opérations effectuées lors de la demande d'une page.

ANNEXE 6
Hiérarchie des Classes Java de GDM

- class java.lang.Object
  - class com.gdm.util.enums.**ActionsEdition**
  - class com.gdm.beans.pages.**AdminDroitsBean** (implements java.io.Serializable)
  - class com.gdm.util.elementsHTML.**ButonType**
  - class com.gdm.util.elementsHTML.**Column** (implements java.io.Serializable)
  - class com.gdm.util.enums.**DroitAccesControles**
  - class com.gdm.util.elementsHTML.**ElementHTML** (implements java.io.Serializable)
    - class com.gdm.util.elementsHTML.**InputHTML** (implements java.io.Serializable)
      - class com.gdm.util.elementsHTML.**InputButonHTML** (implements java.io.Serializable)
      - class com.gdm.util.elementsHTML.**InputChkHTML** (implements java.io.Serializable)
      - class com.gdm.util.elementsHTML.**InputFileHTML**(implements java.io.Serializable)
      - class com.gdm.util.elementsHTML.**InputRadioHTML** (implements java.io.Serializable)
      - class com.gdm.util.elementsHTML.**InputTxtHTML**(implements java.io.Serializable)
      - class com.gdm.util.elementsHTML.**LienHTML** (implements java.io.Serializable)
    - class com.gdm.util.elementsHTML.**SelectHTML** (implements java.io.Serializable)
    - class com.gdm.util.elementsHTML.**TableauHTML**(implements java.io.Serializable)
    - class com.gdm.util.elementsHTML.**TextareaHTML** (implements java.io.Serializable)
  - class com.gdm.beans.pages.**GDMBean** (implements java.io.Serializable)
    - class com.gdm.beans.pages.EtatDesLieux.**MagasinBean** (implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**AccueilMagasinsBean** (implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**CaracteristiquesMagasinBean** (implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**ContactsMagasinBean** implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**DetailMagasinBean** (implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**EvenementsMagasinBean** (implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**HorairesMagasinBean** (implements java.io.Serializable)
      - class com.gdm.beans.pages.EtatDesLieux.Magasins.**LocauxMagasinBean** (implements java.io.Serializable)
  - class javax.servlet.GenericServlet (implements java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig)
    - class javax.servlet.http.HttpServlet (implements java.io.Serializable)
      - class com.gdm.servlets.admin.**AdminDroits**

        - class com.gdm.servlets.securite.**authentification**
        - class com.gdm.servlets.controleurs.**EtatDesLieux**
        - class com.gdm.servlets.fragments.**head_html**
        - class com.gdm.servlets.fragments.**infos_utilisateur**
        - class com.gdm.servlets.securite.**logout**
        - class com.gdm.servlets.exterieur.**MapDisplay**
        - class com.gdm.servlets.fragments.Menus.**MenuEDL**
        - class com.gdm.servlets.fragments.Menus.**MenuGeneral**
        - class com.gdm.servlets.fragments.Menus.**MenuMagasin**
        - class com.gdm.**testJReport**
        - class com.gdm.**testPJX**
    - class com.gdm.util.elementsHTML.**LabelHTML** (implements java.io.Serializable)
    - class com.gdm.util.elementsHTML.**OptionHTML** (implements java.io.Serializable)
    - class com.gdm.util.enums.**PagesEtatDesLieux**
    - class com.gdm.util.enums.**PositionLabel**
    - class com.gdm.util.elementsHTML.infos.**RechercheDroitConsultation**
    - class com.gdm.util.enums.**RequeteAdministration**
    - class com.gdm.beans.sql.**sqlConnexion** (implements java.io.Serializable)
    - class javax.servlet.jsp.tagext.TagSupport (implements javax.servlet.jsp.tagext.IterationTag, java.io.Serializable)
        - class com.gdm.tags.fragments.**AffichageMessages**
        - class com.gdm.tags.fragments.**BaliseForm**
        - class com.gdm.tags.securite.**controle_acces**
        - class com.gdm.tags.securite.**controle_authentification**
        - class com.gdm.tags.fragments.**LienPlan**
        - class com.gdm.tags.fragments.**NavigationPlusMoins**
    - class com.gdm.beans.securite.**utilisateur_valide**(implements java.io.Serializable)

ANNEXE 7
Journal de Stage

Vous trouverez ici le journal réalisé pendant le stage pour archiver le travail effectué chaque jour. Son principal objectif n'est pas de donner des détails sur l'ensemble des tâches, mais de constituer un historique pour faciliter la rédaction de ce rapport.

04-05-2004 Lundi
- Présentation à l'équipe, de l'architecture du réseau
- Prise de contact avec le GDM existant
- Recherche des log nécessaires au développement : Oracle 9/10 iAS , JDeveloper, JDK

04-06-2004 Mardi
- Création de mon compte utilisateur
- Recherche sur les différentes versions des serveurs Oracle10gIAS, les licences, les prix
- Installation du frameworks JDeveloper, début du tutoriale fournit par Oracle (trop générale)
- Prise de connaissance de la structure de la base de donnée existante, modélisation avec Jdev, nombreuses tables, la base doit être refaite sous oracle. Refaire la conception ?

04-07-2004 Mercredi
- Analyse écran par écran de GDM pour mieux comprendre les fonctionnements et cerner les différentes fonctionnalisées
- Rédaction d'un document Word listant pour chaque écrans les champs affichés et les actions possibles, objectif : comprendre toutes les fonctionnalités
- Interrogation de Sébastien sur les concepts incompris (Poste, Phases…)
- Prévision du programme de jeudi : finir rédaction, recherche sur serveur Apache Tomate et installation
- Commencer à utiliser JDeveloper

04-08-2004 Jeudi
- Rédaction du document commencé mercredi terminée
- Test de JDeveloper, difficultés à comprendre car manque de connaissances en JSP
- Avec Jdev : établissement d'une connexion à une base Oracle et affichage de données dans une page JSP

04-09-2004 Vendredi
- Point de l'avancement avec le chef de projet
- Installation et configuration d'un serveur Tomcat pour pouvoir visualiser des pages JSP
- Récupération de la série d'exemple fournie par Oreilly
- Apprentissage des notions de base des JSP : JSP, JavaBean...
- Lecture du livre JavaServerPage d'Oreilly

04-13-2004 Mardi
- Décision avec le maître de stage de reporter le travail sur la base de donnée à la semaine prochaine : par de recréation de base, mais correction des erreurs de l'existante et suppression des tables inutiles
- Début de programmation d'un écran d'exemple reproduisant les fonctionnalités d'un écran existant de GDM : affichage des magasins (filtres, recherche, tris, détail, ajout)…

04-14-2004 Mercredi
- Poursuite de la programmation de l'écran test : connections à la base de donnée, affichage des résultats, possibilité de tris, possibilité de filtre, de recherche… Utilisation de CSS pour une présentation standard
- Installation de NetBeans, test rapide : apparaît plus simple que JDevloper

04-15-2004 Jeudi
- Mise en place du paradigme MVC pour apprendre à séparer présentation, stockage, traitement des données : utilisation de JSP couplé à JavaBean et Servlet
- Déploiement difficile, respect des règles de package, écriture du fichier web.xml dans le dossier WEB-INF

- Problème avec Tomcat, perte des données exemples à cause d'une fonction de suppression sans confirmation

04-16-2004 Vendredi
- Poursuite du travail sur les pages mettant en place le model MVC, résolution des problèmes de déploiement, affichage de données sur plusieurs pages

04-19-2004 Lundi
- Début du travail concret pour l'application
- Objectif : mettre en place un système d'authentification et de contrôle d'accès le plus générique possible pour être réutilisable. Contrôle d'accès à chaque élément des pages.
- Modélisation UML de la base de donnée nécessaire à un tel contrôle d'accès

04-20-2004 Mardi
- Mise en place de la base de donnée correspondant au modèle, sous Oracle.
- Mise en place du premier niveau de contrôle : authentification. (Respect MVC)
- Mise en place d'une redirection sur la page de login en cas de demande de page sans identification préalable

04-21-2004 Mercredi
- Amélioration du niveau 1 : fonctionnement dans les cas de sous répertoires, … (Chemins absolus généré automatiquement)
- Développement de Tag (actions personnalisées) pour le contrôle de Niveau 1 et 2)
- Mise en place du niveau 2 de sécurité : contrôle d'accès : vérification dans la base des autorisations suffisantes pour le profile de l'utilisateur, redirection sur une page d'erreur
- Réflexion sur les possibilités de mise en place du Niveau 3 : contrôle d'accès au données des pages. (Difficulté à trouver la technique la plus simple)

04-22-2004 Jeudi
- Amélioration des éléments développés précédemment pour les rendre plus simple à utiliser (valeurs par défaut…)

Première modélisation des techniques de mise en place du niveau 3.

04-23-2004 Vendredi
- Travail sur les possibilités d'amélioration des connections à la BD : rendu paramétrable avec des variables dans le web.xml, diminution des temps d'affichages en limitant les ouvertures de connexion.
- Recherche sur les Pool de connexion, tentative de mise en place de partage de connexion pour tous les utilisateurs, infructueux et dangereux.

04-26-2004 Lundi
- Poursuite des recherches de mise en place de Pool de connexion.
- Mise en place d'un système de connexion avec une connexion par session. Diminution considérable du temps d'affichage des pages, après l'identification.

04-27-2004 Mardi
- Modélisation des classes représentant les objets disponibles de formulaire dans une page.
- Début de développement des classes représentant en mémoire les éléments des pages.

04-28-2004 Mercredi
- Poursuite du développement des classes entamé le jours précédent.
- Commentaires de ces classes et génération de la JavaDoc correspondant.
- Modélisation des étapes de traitement pour l'accès à une page. (Niveau 3 de sécurité)
- Mise en place de Tag pour le Menu générale, et les sous menus (à poursuivre).

04-29-2004 Jeudi
- Mise en place de l'arborescence de dossiers pour la section Etat des Lieux
- Mise en place d'un servlet de contrôle pour la section Etat des Lieux

04-30-2004 Vendredi
- Début du travail sur la page d'affichage de la liste des magasins

- Mise en place d'une page d'administration pour la gestion des droits en utilisant les ElementsHTML (problèmes de réalisation du MVC)
- Attribution d'un label à chaque élément HTML (décidé avec Sébastien.)

05-03-2004 Lundi
- résolution des problèmes de mise en place Servlet, Bean, JSP pour la page d'administration
- Intégration des tables de gestion des droits à la base de données générale de GDM
- Amélioration des classes d'elementHTML.

05-04-2004 Mardi
- Fin de la page d'administration : gestion des erreurs, obligation de saisie.
- Poursuite de la page d'accueil des magasins : affichage de la liste des magasins.
- TableauHTML : intégration d'une fonction de construction à partir d'une requête SQL (à généraliser)
- Positionnement des Label générique.

05-05-2004 Mercredi
- Amélioration de la page d'accueil des magasins
- Création automatique de liens dans les éléments tableaux si les éléments nécessaires spécifiés.

05-06-2004 Jeudi
-Début d'affichage du détail d'un magasin : élément SelectHTML : intégration d'une fonction de remplissage et amélioration de la fonction d'affichage : gestion des droits d'accès et de la position du label si il existe.
- Intégrations des nouveaux droits d'accès aux pages et aux éléments dans la base.

05-07-2004 Vendredi
- Modification des éléments Checkbox et Textarea, … pour la gestion des droits d'accès et du label, pour le remplissage et la construction
- Affichage de toutes les infos sur le Détail d'un magasin (pas de mise en age)
- Création d'un TAG permettant une navigation +/- dans les magasins (générique pour être adapté aux fournisseurs)

05-10-2004 Lundi
Travail Alternatif :
Travail sur l'Extranet GDM existant en ASP : modification pour permettre le fonctionnement avec Oracle 8 : correction des requêtes SQL : JOINT, types de données incompatibles (CInt)…

05-11-2004 Mardi
- Poursuite et fin du travail sur l'Extranet
- Ecritures des commentaires des classes non commentés, mise à jours de la JavaDoc

05-12-2004 Mercredi
- Mise en place de nouvelles pages : Horaires
- Généralisation de la gestion des droits d'accès, chaque élément HTML peut récupérer sont droit d'accès dans la base.

05-13-2004 Jeudi
- Création d'un TAG permettant de généraliser l'affichage du titre, du sous titre et des messages d'erreur. Titre et sous Titre à porté de session. Création des titres par les servlets de contrôles, des sous titres par les servlet de traitement.
- Généralisation des propriétés des bean de traitement : GDMBean, EtatDesLieuxBean…

05-14-2004 Vendredi
- Mise en place de pages : Evènements, Contacts, Caractéristiques
- Modification de la BD : ajout des champs manquants, insertions de données d'exemple absentes.

05-17-2004 Lundi
- Mise en place de la page sur les locaux et les compositions associées.
- Création d'un second profile pour tester les différents droits d'accès : consultation, édition
- Début de la mise en place des possibilités d'édition : modifier supprimer sur lignes des tableau ; boutons pour soumettre les modifications de la page Détail magasin

- Rechercher sur les Doclet de génération de javadoc en PDF

05-18-2004 Mardi
- Javadoc avec doclet en PDF
- Tableau : intègre liens Supr :Modif si droit d'édition sur le tableau
- Servlet de traitement pour l'édition.

05-19-2004 Mercredi
- étude mise en place de l'édition
- Tag de génération de la balise forme (droits)
- identification des éléments obligatoires dans formulaires : CSS

05-24-2004 Lundi
- VISITE DE Mme ESPINASSE
- reprise du fonctionnement de l'édition
- Bean responsable du traitement des nouvelles données
- modification ElementsHTML pour mise à jours donnés
+ travail sur carte Easycomptage

05-25-2004 Mardi
- Poursuite de la modification des ElementsHTML : capable de se mettre à jour sans nouvelle connexion à la base
- intégration d'une fonction aux beans pour le stockage des données éditées et la màj du contenu des ElementsHTML
- mise en place de l'édition pour la page Détail Magasin : fonctionnement correct, à optimiser.

05-26-2004 Mercredi
- optimisation de l'édition
- début de mise en place de l'insertion. Evolution des ElementsHTML
- Création d'une carte des magasins pour pré visite comptage.
- Problème rencontré sur transmission de données : jeux de caractères (HTML/java)

05-27-2004 Jeudi
Résolution de problèmes :
- Jeux de caractère : ISO-8859-1 / UTF-8 pour transmission de données dans formulaire ->paramètre général dans web.xml
- fermeture des stmt : erreur dans tag de construction de form
- affichage des errorMsg (écrasement supprimé).
- suppression de colonne en excédent dans tableaux
- Ajout message : infoMsg
- Modification de BaliseForm pour intégration de paramètres

05-28-2004 Vendredi
- Modélisation et programmation de LienHTML
- Formulaire d'insertion : affichage formulaire vide.
- Commentaires
+ impressions plans

06-01-2004 Mardi
Insertion
- raz des champs
- ajout requêtes select élément par défaut
- contrôle données obligatoires : édition, insert : message d'info
- limite taille InputTxtHTML
- vérification type de données NUMERIC InputTxtHTML
- vérification dates : problème de formatage
- message dans sortie serveur (login logout)

06-02-2004 Mercredi
- formatage des éléments TIMESTAMP
- vérification du format des dates pour l'édition
- vérification de la taille (textarea et inputtxt)

06-03-2004 Jeudi
- Recherche sur la structure des PDF : langage Portable Document Format
- Premier contact avec PJX
- Mise en place d'un exemple modifiant le titre

06-04-2004 Vendredi
- Etude du fonctionnement du site Mappy
- Mise en relation de GDM avec Mappy
- Utilisation des adresses gdm
- Affichage direct du plan correspondant

06-07-2004 Lundi
-finition de la construction de l'affichage de plans
- correction <form>
- correction de problème après insertion (débuggage)
- Recherche sur les outils de reporting (Chrystal Report)

06-08-2004 Mardi
- Mise à jours de la Javadoc
- Premiers test de JasperReporting

Travail alternatif :
- Oracle import d'une BD gdm Mim de oracle 8i à oracle 8 problèmes de compatibilité descendante)

06-09-2004 Mercredi
- rapport
- mise à jours des diagrammes
- poursuite des recherches pour migrations de la base de donnée

06-09-2004 Jeudi
- Exécution des tutoriaux de iReport
- Apprentissage des découpages des rapports
- Etude de fichiers xml de JasperReport

06-11-2004 Vendredi
- exportation oracle : incompatibilité avec versions antérieurs
- récupération des requêtes : impossible car format de fichier propriétaire
- page de test d'intégration des rapports dans les JSP : problèmes d'exécution

Résumés

Anglais :

The Cabinet Zulian is a science and computer consulting group in expansion. One of is main product is GDM, a store management software aimed at the big companies This application currently exists as an Access and VBA system. The main topic of this internship was to work on the new version, as a Web client.

The work will continue during six weeks. The achieved work has consisted in the achievement of the database structure with the willingness to control the access to the data. Those data are used in elements of the application developed during this internship for authentication and access controls to pages and data. Elements to simplify as far as possible data edition and consultation in forms have also been developed. The web application structure has also been defined. The actual step is the reports production to be broadcasted and printed and containing data from the GDM database.

This internship has been a great experience, not only about enterprise running, but also about new technologies I have discover.

Français :

Le Cabinet Zulian est un cabinet de conseil scientifique et informatique en pleine croissance. L'une de ses principales applications et un logiciel de gestion des points de vente destiné aux maisons mères de la grande distribution (la Halle, Groupe André…). Cette application existe sous Access en VBA ; l'objectif de ce stage était de travailler sur la migration de cette application vers un client Web.

Le travail réalisé va être poursuivi pendant encore six semaines, il a consisté en la mise en place des éléments de base de donnée pour le contrôle d'accès aux données, le développement des éléments capables d'utiliser ces informations pour l'authentification et le contrôle d'accès aux pages et aux données. Ont aussi été développés des composants pour simplifier les plus possible l'affichage et l'édition des données dans des formulaires. La structure générale de l'application a elle aussi été mise en place. L'étape encor en cours consiste à la génération de documents diffusables ou imprimables comportant des informations issues de la base de données associée à GDM.

Ce stage a été une grande expérience sur le fonctionnement de l'entreprise, mais aussi une découverte de nouvelles technologies passionnantes.

Mots Clef :

Migration - JSP - Paradigme MVC - Contrôle d'Accès - Formulaires - Client Web - Tomcat - NetBeans
Apprentissage - Autonomie - Communication - Recherches

CABINET ZULIAN
Ingénierie et Conseils Scientifiques

Le Président, Route de Lyon
01800 Pérouges
Tél.: (+33) (0)4 74 46 74 88
Fax: (+33) (0)4 74 46 08 69
Email : contact@cabinet-zulian.com
Web : www.cabinet-zulina.com