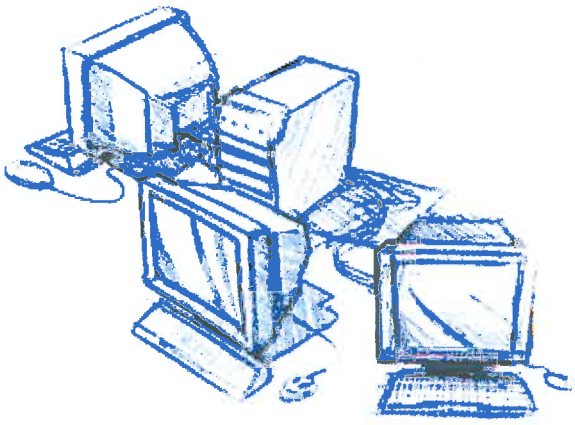
Jean LAMY

Promotion IUP3 2002/2003

IUP GMI D'AVIGNON

Développement d'une application de Gestion de Parc Informatique



Maître de stage : *Nathalie MOITRIER*

Période : *14 avril au 14 septembre 2003*

INRA

Unité CSE

INRA

Institut National de la Recherche Agronomique

Remerciements

Tout d'abord je tiens à remercier André CHANZY, le directeur de l'unité CSE qui m'a accueilli au sein de son unité.

Puis mon maître de stage, Mme Nathalie MOITRIER qui m'a permis de réaliser ce stage et dont le suivi et les conseils ont été très bénéfique pour moi.

Je tenais à remercier aussi Sylvain BARBACE, Philippe CLASTRE, Dominique RIPOCHE, Ghislain SEVENIER, Patrice LECHARPENTIER (membres du groupe SOSI) pour leur aide pendant ces 5 mois.

Enfin un dernier grand merci à l'ensemble de l'unité pour son accueil très sympathique.

Tables des matières

Tables des illustrations

Avant propos

Afin de valoriser ses connaissances et d'avoir une expérience du monde professionnel, la formation d'Ingénieur Maître à l'IUP d'Avignon prévoit un stage de 4 mois minimum en dernière année. C'est dans ce cadre là que l'INRA d'Avignon, et plus particulièrement l'unité **C**limat **S**ol **E**nvironnement (CSE) m'a proposé un stage de 5 mois dont l'objectif était le développement d'un outil de gestion de parc informatique.

Les principales raisons qui m'ont poussé à choisir ce stage sont tout d'abord les aspects techniques qu'il regroupait : application WEB connectée à une base de donnée, ce qui correspond à mes centres d'intérêts informatique ; de plus connaissant l'INRA pour y avoir réalisé mon projet de fin d'étude, j'ai été attiré par le cadre très agréable du centre d'Avignon.

Ce stage s'inscrivant dans la suite d'un projet de fin d'étude dans lequel une analyse détaillée a été conçue mais non validée, j'ai donc effectué l'ensemble des étapes de réalisation d'un logiciel, de l'analyse aux tests finaux, travail que je vous présente dans ce rapport.
Après vous avoir présenté le cadre de travail ainsi que le projet qui m'a été confié, j'aborderais les moyens, outils, méthodes utilisés et décrirais par la suite l'ensemble des étapes de la réalisation. Enfin, je terminerais par un bilan.

1.Cadre de Travail

a. L'INRA

L'Institut National de la Recherche Agronomique a été créé en 1946 en application de la loi Tanguy Prigent, le ministre de l'agriculture de l'époque. Il s'agit d'un établissement public à caractère scientifique et technologique, placé sous la double tutelle des ministères chargés de la recherche et de l'agriculture.
L'institut à plusieurs rôles :

- Il doit œuvrer au service de l'intérêt public tout en maintenant l'équilibre entre les exigences de la recherche et les demandes de la société.
- Il doit produire et diffuser des connaissances scientifiques et des innovations, principalement dans les domaines de l'agriculture, de l'alimentation et de l'environnement.
- Enfin, il doit contribuer à l'expertise, à la formation, à la promotion de la culture scientifique et technique, au débat science / société.

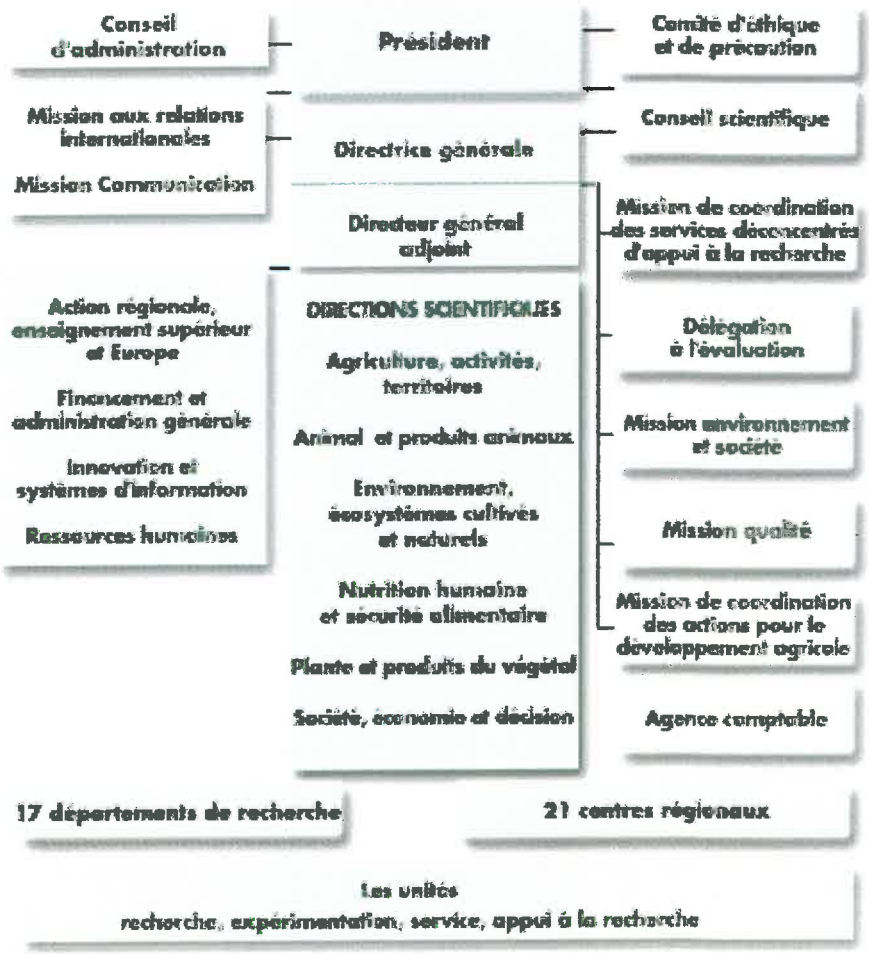


Figure 1 : organigramme INRA

Présent sur tout le territoire, il est organisé de la manière suivante :

- 17 départements de recherche touchant l'agriculture, l'alimentation et l'environnement regroupés dans 5 directions scientifiques (cf. figure1).
- 21 centres régionaux répartis en près de 200 sites dans toute la France.
- 257 unités de recherche (dont 128 associées à d'autres organismes comme le CNRS, le CIRAD, universités,…).
- 80 unités expérimentales.
- 131 unités d'appuis et de services.

Sur le plan des moyens financiers, le budget de l'INRA est de 573 millions d'euros (env. 3,7 milliards de francs…)

b. Le centre d'Avignon

Il a été fondé en 1953 et constitue un maillon important du dispositif de l'INRA tant par sa taille que par les thématiques de recherches qui lui sont confiées. Sa vocation est aussi bien régionale que nationale. Son implantation vauclusienne lui a conféré traditionnellement une responsabilité vis à vis de l'agriculture méditerranéenne pour la qualité de la production maraîchère et fruitière et pour l'industrie régionale de la transformation agroalimentaire.
Les programmes scientifiques du Centre Inra d'Avignon lui confèrent également une forte implication dans la politique scientifique nationale de l'institut.
Deux pôles majeurs caractérisent le centre :

- Gestion de l'environnement, pour les territoires cultivés et la forêt méditerranéenne.
- Maîtrise de la qualité des produits cultivés et transformés pour la santé du consommateur.

Le centre compte en moyenne 570 agents permanents et 400 temporaires par an. Il dispose d'un budget annuel de 38 millions d'euros, salaires compris et publie annuellement 450 documents.
Le centre compte 13 unités de recherche et 6 unités expérimentales réparties sur 3 régions, majoritairement en PACA (cf. figure 2)

Figure 2 : les implantations du centre d'Avignon

c. L'unité CSE

L'INRA est structuré en départements de recherche, construits sur la base des disciplines scientifiques nécessaires pour couvrir tous les besoins de l'agriculture.
Cette structure se retrouve sur le centre d'Avignon par l'existence des « unités », qui sont sous la tutelle d'un ou plusieurs de ces départements.
L'unité CSE, (**C**limat, **S**ol et **E**nvironnement) dirigée par André Chanzy, sous la tutelle du département Environnement et Agronomie, est une jeune unité (2ans) puisqu'elle est le « résultat » de la « fusion » des deux anciennes unités Science du sol et Bioclimatologie. Elle mène des travaux de recherche sur :

- La description des transferts de masse (eau, gaz, particules) et d'énergie dans le continuum nappe sol plante atmosphère.
- Le couplage des transferts de masse dans le sol avec les cycles biogéochimiques.
- Le développement de la végétation des écosystèmes cultivés en relation avec le climat, les propriétés du sol et les pratiques agricoles.

Ces travaux ont pour objectifs finalisés :

- La quantification de l'impact environnemental des pratiques agricoles et des épandages de déchets sur la qualité des eaux (eaux de nappe en particulier) et des sols.
- L'optimisation des ressources pour les cultures (irrigation, fertilisation, agriculture de précision) et la mise au point d'itinéraires techniques préservant l'environnement.
- La prévision des rendements et la délimitation des potentialités des zones de production agricole en fonction du complexe agropédoclimatique ou des changements climatiques.

Les recherches fondamentales portent sur la description des processus, sur leur spatialisation en allant des échelles locales aux échelles régionales et sur l'élaboration de modèles dynamiques de fonctionnement des écosystèmes. Ceux ci sont utilisés dans les études plus finalisées comme outil de diagnostic, de pronostic ou de simulation de données en complément des observations expérimentales. Pour accompagner ces recherches, l'unité mène des travaux méthodologiques sur la

caractérisation des propriétés physiques et chimiques des sols (flux hydrique, propriétés de transfert, ambiance géochimique…).
L'unité fonctionne avec 50 agents permanents dont 20 chercheurs, 8 ingénieurs, 14 assistants ingénieurs / techniciens, 3 agents techniques, 3 secrétaires, 3 documentalistes et 14 doctorants.
Elle est composée de 4 équipes : Couplage entre transferts et cycles biogéochimiques – Transfert de masse et d'énergie dans le sol – Télédétection – Fonctionnement de l'espace agricole.

d. Le groupe SOSI

Au niveau gestion de ressources informatique l'INRA a mis en place sur chaque centre des **E**quipes d'**I**nformatique **C**ollective (EIC) qui s'occupent principalement de la politique informatique générale, des serveurs du centre, de la messagerie, du WEB. Face à l'émergence des postes de travail individuels il a été nécessaire de mettre en place des « personnes ressources informatique » dans chaque unité. Ces « personnes ressources » sont désignées par les directeurs d'unités et travaillent en relation avec l'EIC de leur centre. Sur le centre d'Avignon ces personnes ressources forment un groupe nommé SOSI (pour **SOS I**nformatique).
L'unité CSE a une équipe SOSI conséquente de 6 personnes ; ces personnes, réparties en 2 équipes sur les 2 bâtiments de l'unité (bâtiment SOL et CLIMAT,) ont pour mission de mettre en œuvre et maintenir l'outil informatique au service des agents de l'unité. Pour cela, l'équipe s'organise autour des thèmes suivants :

- Renouvellement du parc matériel : achat et installation de matériel
- Opérations de maintenance multi plateformes (incidents, mise à jour software, mise à jour hardware, réseau)
- Animation auprès de l'utilisateur (mail, présentations)

L'animateur et responsable de ce groupe, Monsieur Philippe Clastre, a la charge du budget prévisionnel.

Il est important de préciser que ces personnes ne consacrent que quelques pourcents de leur temps de travail à cette activité informatique et que le recrutement début 2003 d'une personne à plein temps a entraîné des changement dans l'organisation des tâches : ces personnes vont devoir se recentrer sur leurs cœurs de métiers respectifs et diminuer le temps passé sur les activités SOSI.
Ceci permet de omprendre l'importance de pouvoir disposer d'un outil de gestion du parc informatique adapté aux besoins pour aider au bon fonctionnement des outils de l'unité.

2.La mission

a. Contexte

i. Origine des besoins

Suite au regroupement en janvier 2001 de deux anciennes unités dans une seule, l'unité CSE, il y a eu réunification des moyens informatiques et de leur gestion avec des fonctionnements différents qu'il a été nécessaire de revoir.
Le fait que l'unité soit répartie sur 2 bâtiments ne facilite pas l'organisation et la bonne gestion de l'outil informatique.
De plus, il n'existait aucun outil de gestion du parc à proprement parler au niveau du centre d'Avignon. Le groupe devait donc se débrouiller par ses propres moyens (oral, mail, affichage,...).

Enfin, ayant dégagé des objectifs assez précis, mais aussi pour cause de budget, il était impossible pour l'unité d'acquérir un logiciel commercial, trop complexe et trop cher.
C'est donc pour ces raisons qu'ils ont sollicité un stagiaire afin de lui confier la mission de développer une application pour gérer le parc informatique de leur unité.

ii. Fonctionnement actuel

Pour bien situer le contexte, voici l'état actuel du parc informatique de l'unité CSE :

- 80 PC (Windows 95, 98, XP, 2000, Linux)
- 5 Terminaux (+ 10 clients Soft)
- 2 Serveurs TSE (pour la bureautique sur terminal)
- 4 Serveurs Linux (Sauvegarde, Calcul, serveur WEB)
- 4 Serveurs SUN (Traitement d'Image, Calcul
- 3 Imprimantes réseaux
- Réseau : TCP/IP RJ45 10/100MB
- Accès aux ressources collectives du centre (disque, CPU, impression couleur, WEB)

Comme indiqué précédemment, 5 personnes (parmi 6) du groupe SOSI consacrent « officiellement » de 5% à 20% de leur temps aux tâches informatiques. Les problèmes informatiques sont déclarés par les utilisateurs soit par mail, téléphone, ou par « visite » directe à un SOSI... Les SOSI travaillent la plupart du temps dans l'urgence et n'ont donc pas le temps nécessaire pour mettre en place des moyens adaptés pour les aider dans leurs tâches de gestion du parc, ce qui engendre un défaut de traçabilité des opérations et des difficultés pour obtenir une vision claire de l'état du parc.

b. Objectifs et intérêts

L'objectif est de réaliser une application WEB de gestion de parc informatique offrant les possibilités suivantes :

- Gestion de tout le matériel informatique du parc (Ajout, Modification, Suppression, Recherche, Tableaux de bords)
- Gestion des logins utilisateurs (pour gérer l'accès à l'application)
- Gestion des problèmes et interventions sur le matériel et/ou le logiciel
- Gestion des réservations de matériel (rétro projecteur, portable, poste de travail,...)

L'application sera accompagnée d'une documentation technique (afin d'assurer la maintenance future) et d'un manuel utilisateur.

Les intérêts d'un tel outil sont multiples. Tout d'abord, tous les utilisateurs utilisent une seule et même application via leur navigateur WEB. De plus celle-ci est accessible par n'importe qui, depuis n'importe quel endroit de l'unité. Enfin sur le plan organisationnel, elle permet de rendre l'utilisateur plus indépendant dans certaines tâches.

3.Méthodes et Organisation du travail

i. Méthodes

J'ai utilisé plusieurs méthodes enseignées lors de mon cursus à l'IUP. Pour la conception de la base de donnée, j'ai utilisé la méthode MERISE afin de définir le **M**odèle **C**onceptuel de **D**onnées (MCD) et le **M**odèle **P**hysique d**e** **D**onnées. De plus pour ce projet j'ai utilisé les approches usuelles de réalisation de logiciel en passant par une phase d'analyse, de conception puis de développement.

ii. Organisation

Afin de vérifier et de réajuster si nécessaire les objectifs fixés, une réunion hebdomadaire était organisée avec une partie des membres du groupe SOSI CSE.

Ce qui permettait une bonne communication entre les différentes personnes concernées par le projet et un suivi au plus prêt.

De plus, l'INRA comptant plusieurs stagiaires en informatique, nous nous sommes rencontrés lors de réunions pour faire le point sur l'avancement de chacun et mettre en commun nos idées, nos problèmes, nos choix,…

iii. Planification

Afin de bien se situer par rapport à l'avancement du projet, il était important de réaliser un planning prévisionnel. Celui-ci a été réalisé en accord avec mon maître de stage au début du stage et a été mis à jour par la suite en fonction des priorités établies.

Lors des réunions d'avancement, ce planning a servi a vérifier la bonne conduite du projet et des modifications ont été faites suites à des ajustements d'objectifs.

Je vous propose ici une version simplifiée du planning prévisionnel :

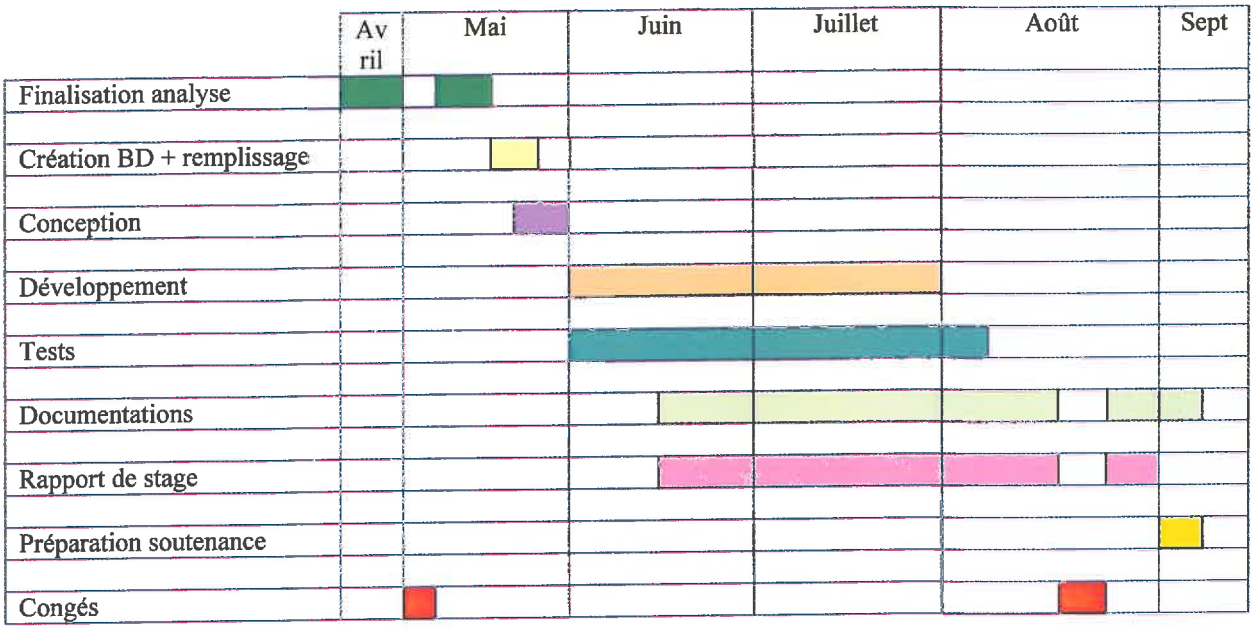


Tableau 1 : planning prévisionnel, période du 15 avril-15 septembre 2003

Pour une version détaillée de ce planning, avec entre autre la comparaison avec le planning réel, veuillez vous référer à l'*annexe 1*

4.Réalisation

a. Contexte

Dans le processus de réalisation de l'application, deux phases ont été réalisées lors d'un projet de fin d'étude : le cahier des charges ainsi que l'analyse, celle-ci n'étant toutefois pas complète.

Pour mener a bien mon projet, je devais donc dans un premier temps reprendre et finaliser l'analyse puis faire la conception jusqu'aux tests finaux et réaliser la documentation. Je vais détailler ces étapes ci-après.

b. Analyse

L'analyse a donc fait l'objet d'un sujet de projet de fin d'étude de l'IUP, auquel j'ai participé.

Le principal objectif était de fournir au groupe SOSI CSE un document d'analyse regroupant la totalité des besoins relatifs à une application de gestion de parc (*voir plan du document d'analyse en annexe 3*).

L'analyse ayant été en grande partie faite lors du projet de fin d'étude, cette partie n'a pas occupé une grande place dans mon stage. Néanmoins, celle-ci n'étant pas tout à fait complète, il a fallu la reprendre pour la compléter et enfin la valider avec le client. Un document d'analyse a donc été produit comportant l'analyse de tous les cas d'utilisation possible de l'application ainsi qu'une représentation conceptuelle de l'organisation des données.

Au début de mon stage j'ai complété cette analyse en fonction des remarques réalisées par les membres du groupe « SOSI CSE ».

L'application « gestion de parc » sera utilisée par les membres du groupe SOSI pour la gestion du matériel informatique (PC, imprimantes..) et autres (vidéo projecteur, ...) et des interventions réalisées sur l'ensemble du matériel. Mais des parties de l'application devront être accessibles par de « simple » utilisateur au niveau de la réservation du matériel et de la demandes d'interventions.

Il y a donc deux profils d'utilisateur avec des accès sur les données de la base différents (*voir annexe 4 : les diagrammes utilisateurs*) :

- **Administrateur** : accès total sur la base de données, mise à jour du matériel informatique et autres, recherches sur des critères précis d'un matériel, gestion des demandes d'intervention
- **Utilisateur** : accès restreints en modification sur certaines données c'est à dire uniquement sur la création de demandes d'interventions et de réservations.

Le groupe SOSI a sous sa responsabilité le matériel informatique mais aussi du matériel commun comme les vidéos projecteurs, les imprimantes, les portables,...

De plus le matériel informatique est en perpétuelle évolution, de nouveaux composants pourraient apparaître.

Pour ne pas « fermer » le type des données stockés dans la base,le **M**odèle **C**onceptuel de **D**onnée (MCD) a été réfléchi et conçu pour pouvoir permettre l'évolutivité de la base de donnée au niveau matériel. On peut ainsi rajouter un nouveau composant qui pas été prévu à ce jour.

Trois entités ont été définies :

- L'entité **Composant** définit tout ce qui est commun à l'ensemble du matériel (composant) (par exemple la date d'achat, le nom, l'utilisateur,…).
- L'entité **Type_Comp** : elle correspond à une catégorie spécifique (ou type). Chaque matériel appartient à un type.
- L'entité **Car_Type** : les particularités de chaque type de matériel se retrouvent dans cette entité.

Afin de mieux comprendre je vous propose l'extrait de mon MCD correspondant à ce principe d'évolutivité :

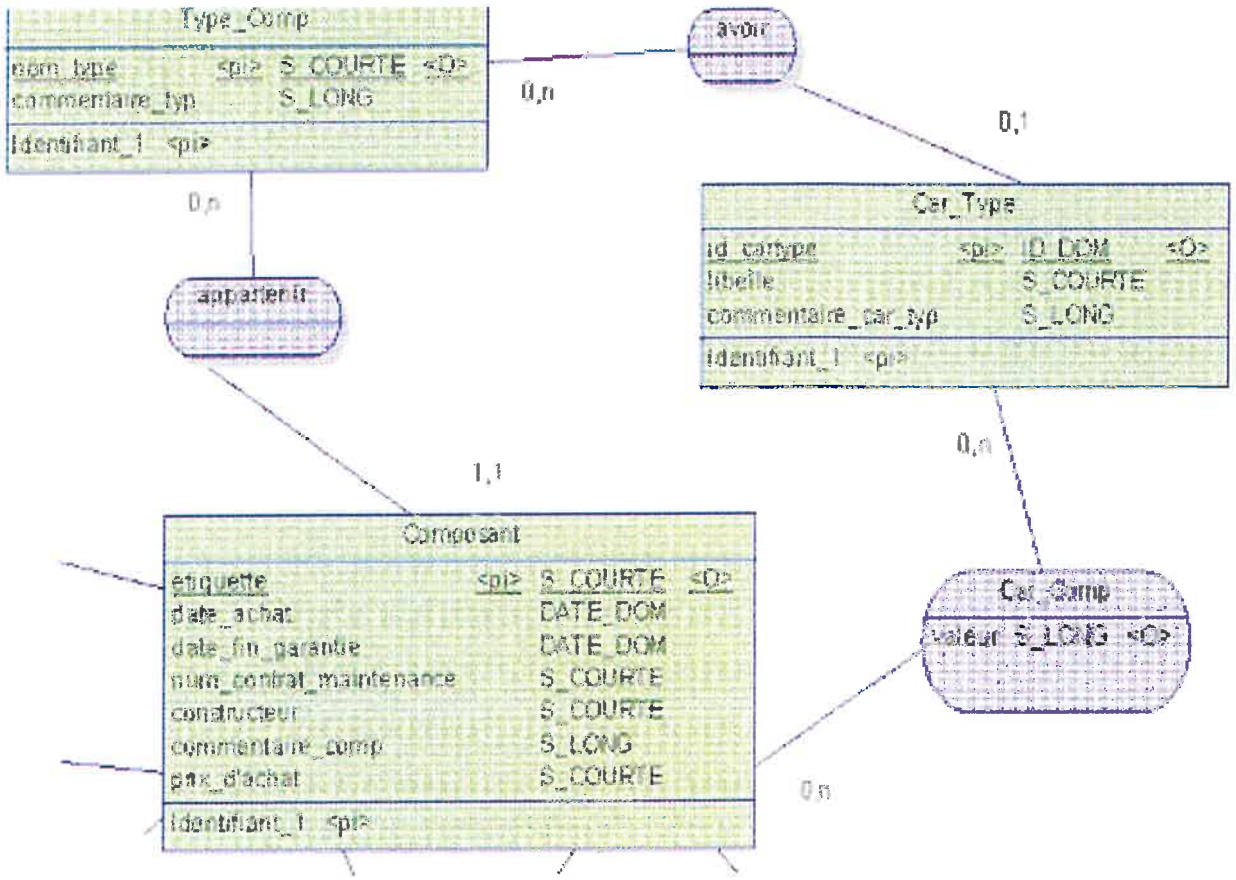


Figure 3 : MCD, gestion évolutive des types de matériel

Prenons l'exemple pour une machine dont l'*etiquette* est *pclim12*:

Cette machine appartient au type PC. Ce type a les caractéristiques suivantes : RAM, OS, IP, MAC,….

On pourra donc attribuer des valeurs aux caractéristiques d'un PC pour *Pclimxx* (entité **Car_COMP**).

De plus il sera aisé d'ajouter une caractéristique, par exemple « carte mère » au type PC sans avoir à toucher la structure de la base de donnée.

Et de même si on a besoin de rajouter le type « Palm ».

Pour la finalisation du Modèle Conceptuel de Données (MCD) j'ai utilisé la méthode **MERISE 2** de représentation des données (sous PowerAMC).

Dans le cadre de ce stage de 5 mois, il n'était pas possible de réaliser correctement l'ensemble des fonctionnalités décrites dans le document d'analyse. Des priorités ont donc été définies avec le groupe SOSI et révisées à un rythme hebdomadaire.

c. Conception

i. Outils utilisés

Tous les outils utilisés étaient des outils Open Source sous licence GPL, BSD (*voir annexe 5*) ou en version d'évaluation.

1. Base de donnée

- **Power AMC Designer version 9.5.736** de chez Sybase (http://www.sybase.com/home) en version d'évaluation limité à 45 jours. Ce logiciel a été utilisé pour réaliser le MCD, le Modèle Physique des Données (MPD) ainsi que les scripts SQL de création de la base de donnée. Il s'agit d'un des logiciels les plus reconnus dans ce domaine et qui s'est révélé très utile dans la modélisation de la base de donnée.

- **PostgreSQL 7.3.2** (http://www.postgresql.org) a été choisi comme Système de Gestion de Base de Données Relationnel. Il est disponible sous licence BSD (gratuit). Après m'être documenté il m'a semblé être un très bon SGBD-R et s'est trouvé parfaitement adapté dans mon cas, contrairement à MySQL (autre SGBD gratuit) qui péchait par le non-support des transactions

- **PhpPgAdmin version 3.00 beta RC1** de Christopher Kings-Lynne (http://phppgadmin.sourceforge.net/) basé sur le projet PhpMyAdmin de Tobias Ratschiller. Il s'agit d'une application Php d'administration de base de donnée sous PostgreSQL. Elle permet de réaliser toutes sortes d'opérations d'administration (création de tables / séquences / triggers / vues, insertion / suppression / modification d'enregistrement,…) le tout grâce à une interface conviviale.

2. Développement

Dev-Php IDE 1.9.4 de Leonardo Garcia (http://devphp.sourceforge.net). C'est un éditeur Php, HTML, Javascript,… sous licence GPL qui supporte la coloration syntaxique et l'aide intuitive des fonctions Php. J'ai donc utilisé ce logiciel pour le développement de l'application, pour écrire les scripts PHP, les « templates » XHTML ainsi que le Javascript et la feuille de style CSS..

Après validation de l'analyse, un document de conception a été rédigé pour spécifier le fonctionnement général de l'application, expliquer la phase de mise en place de la base de donnée, décrire l'interface de l'application, expliquer le choix des outils, des technologies. Je vous présente succinctement ces parties.

ii. La base de donnée

Afin de créer la base de donnée, à partir du MCD j'ai généré le Modèle Physique de Données (représentation réelle des tables)sous PowerAMC. Puis dans un second temps j'ai utilisé

l'option de génération automatique des scripts de création des tables, séquences,... toujours sous PowerAMC. Cela m'a permis ensuite d'avoir une bonne base pour créer la base de donnée sous **PostgreSQL**. J'ai ensuite créé un script de création par table puis un script où étaient déclarées les contraintes d'intégrité (voir annexe 7 : les scripts SQL). Enfin, j'ai effectué des tests d'insertion, suppression sur les tables.

Toutes ces phases sont détaillées dans le document de conception.

Problème de choix de la base de donnée

Lors de cette phase de la conception, j'ai du remettre en cause le choix initial effectué pour la base de donnée, à savoir **MySQL 4.3.2**, un « SGBD » très répandu. Ce choix avait été fait après recherche et documentation (je voulais savoir si MySQL 4 gérait les contraintes d'intégrité, un critère de choix primordial dans mon cas). De plus étant déjà installé et utilisé au sein de l'INRA d'Avignon, cela a peut être précipité la préférence initiale à MySQL par rapport à PostgreSQL.

Au moment de la création de la base de donnée et de ses contraintes d'intégrité (les clés étrangères qui lient deux tables), en réalisants des tests, je n'arrivais pas à vérifier que ces contraintes étaient prises en comptes. Ces clés étrangères (*voir exemple de clé étrangère* ci dessous) qui font références à des clés primaires d'une autre table, étaient bien détectées mais les contraintes sur insertion ou modification de données n'étaient pas prises en compte (pour être rigoureux il peut être intéressant lorsque il y a une suppression / modification d'une clé primaire, de répercuter cela aux étrangères concernées). Après avoir lu quantité de documents (documents officiels, forums,...) j'ai trouvé un sujet sur un forum stipulant que, contrairement à ce qu'il était annoncé dans la documentation officielle, MySQL dernière version acceptait les clés étrangères mais en aucun cas ne les gérait correctement pour répondre aux questions d'intégrités. J'ai donc décidé de changer de SGBD, plutôt que de gérer ces contraintes d'intégrités dans le code (lourdeur et temps de programmation accru), je me suis donc orienté vers PostgreSQL 7.3.2, qui lui s'est avéré très performant, et gère parfaitement les clés étrangères.

Je ne pense pas que ce problème ait été pénalisant pour le projet, bien au contraire. La seule déception est d'avoir été confronté à un manque de clarté dans la documentation officielle de MySQL, ce qui m'aurait permis de choisir PostgreSQL directement.

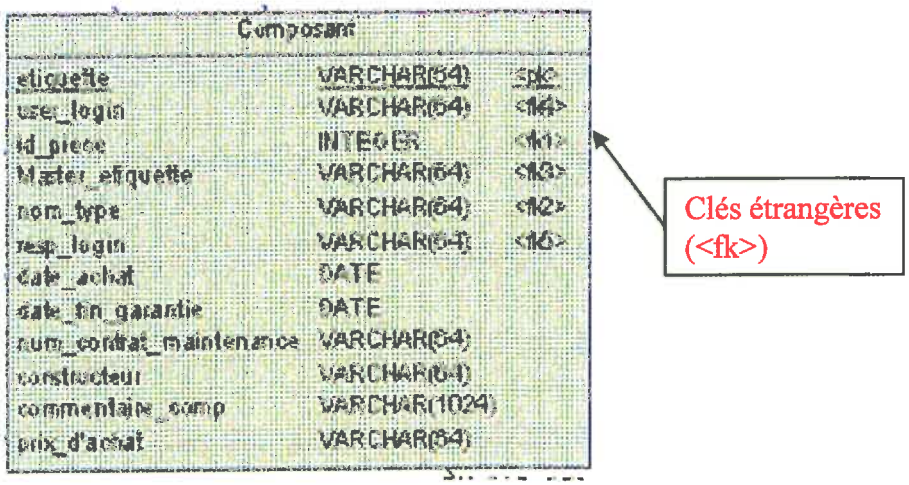


Figure 4 : exemple de clés étrangères (fk)

Explication de la figure 4

Cette figure montre les clés étrangères de la table Composant. Celles-ci sont marquées par le symbole *<fk>*. Prenons la clé étrangère **user_login** : celle-ci fait référence à la clé primaire **login** de la table **Personne** (*voir annexe 6 : le MPD*). Lors de la suppression d'un enregistrement de la table personne pour un login=toto par exemple, tous les enregistrement de la table Composant dont le

champ user_login était égal à toto verront ce champ mis à NULL **automatiquement**. Ceci n'est bien sûr qu'un cas particulier pour expliquer les relations entre tables.

iii. L'interface graphique

Il s'agissait de décrire le fonctionnement de l'application : enchaînement des écrans, mais aussi les contrôles fait sur les données saisies et l'aspect général comme la présentation (menu,…), la résolution d'écran la mieux adapté,…

De manière générale, l'application a été pensé comme beaucoup de site WEB, avec un menu en haut, un espace pour afficher chaque partie du menu et un pied de page contenant certaines informations.

De plus afin d'aider au développement, j'ai réalisé un plan du site en indiquant pour chaque page le script PHP concerné (*par exemple pour la page d'ajout d'un matériel le script concerné est addcomp.php*).

iv. Choix des technologies pour le développement

Je vous présente tout d'abord un schéma montrant l'utilisation, et le lien entre tous les outils utilisés. Ils seront décrits succinctement par la suite.

Le choix de ces outils est soit imposé (le cas de PHP) soit issus d'une réflexion menée grâce aux informations trouvées sur Internet et/ ou mes connaissances,…

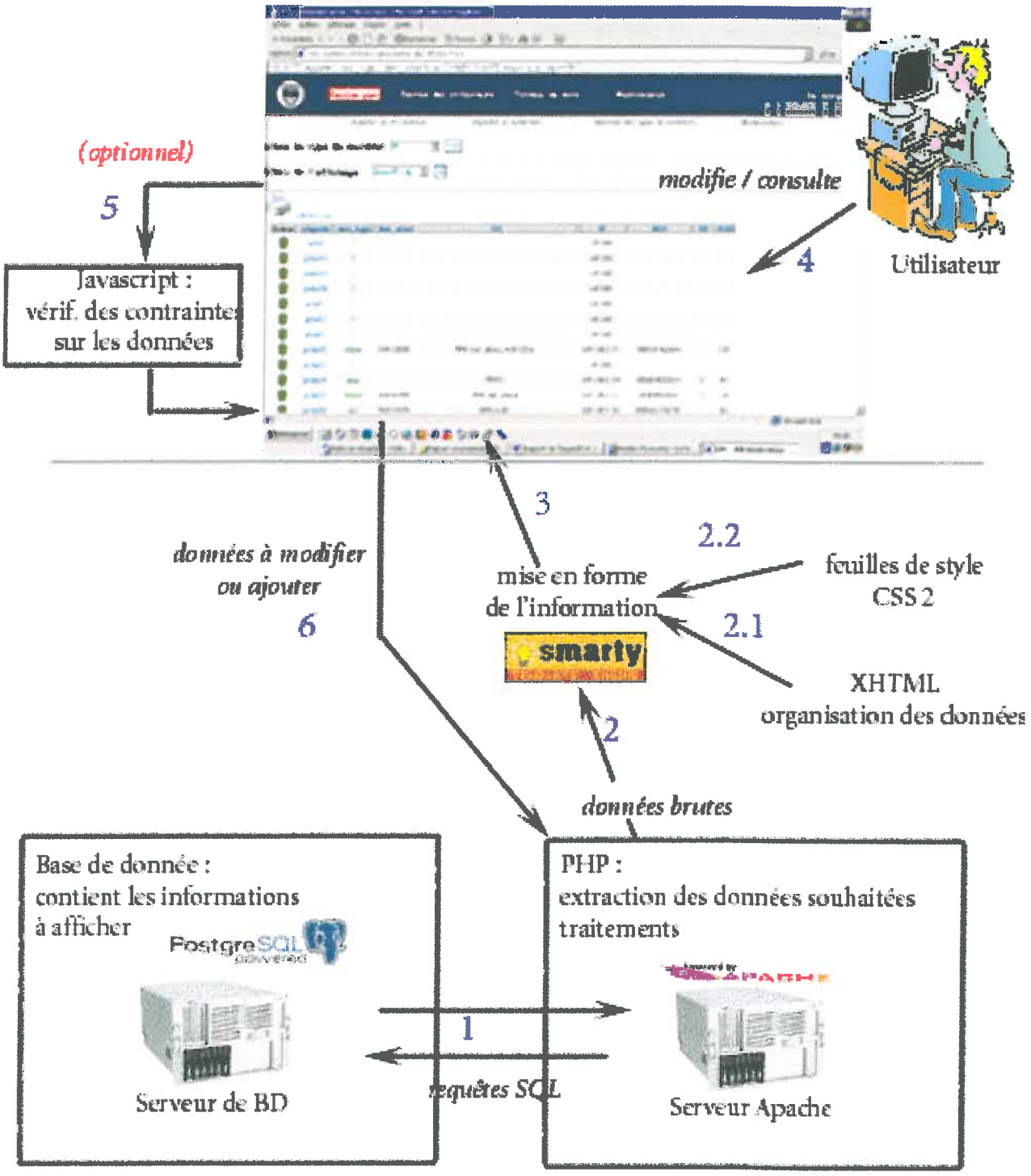


Figure 5 : utilisation et liens entres les technologies

1. PHP

PHP est un acronyme récursif qui signifie « Hypertext Preprocessor ». Il s'agit d'un langage de script HTML exécuté depuis le serveur WEB. Sa syntaxe est largement empruntée aux langages C, Java et Perl.

Il permet aux développeurs de créer des pages WEB dynamique de manière assez simple. Son grand avantage étant qu'il est « Open source » et que son utilisation est très répandu.

Techniquement, PHP tient la comparaison avec ses concurrents : ASP, ColdFusion, Perl ou Java Server Pages (JSP).

Dans un souci de facilité de maintenance, étant donné que des membres du groupe SOSI possédaient quelques connaissances du langage PHP, c'est celui-ci qui a été choisi.

2. XHTML 1.0 et 1.1

XHTML (e**X**tensible **H**yper **T**ext **M**arkup **L**anguage) 1.0 est une extension de HTML 4, qui est le langage à balise qui s'est imposé comme standard du langage de publication du World Wide Web. Le plus de XHTML, le « X », est qu'il est conforme à la norme XML et à la fois, si certaines règles simples sont suivies, qu'il fonctionne avec les agents utilisateurs conforme au HTML 4.

XHTML est conçu en gardant à l'esprit la notion d'interopérabilité entre les agents utilisateurs (dont font partie les navigateurs).

Dans notre cas, XHTML1.1 a été choisi car il s'agit du futur standard des pages Internet et afin de pouvoir faire fonctionner l'application sur tous les types de navigateurs présent dans l'unité, tout en prévoyant de la rendre compatible avec les futures versions des navigateurs.

3. CSS2

CSS est l'acronyme de **C**ascading **S**tyle **S**heet (Feuille de style en cascade). Ce sont donc des feuilles de style qui définissent la façon d'afficher les données des fichiers XHTML ou HTML. Il s'agit d'un langage pratique et simple

Dans le cadre de la présentation de l'application, CSS2 sera utilisé pour mettre en forme les documents XHTML généré grâce aux scripts PHP afin d'avoir une mise en forme correcte dans tous les navigateurs.

4. Javascript

Il s'agit d'un langage créé à l'origine par Netscape en 1996 permettant de faire réagir par exemple une page WEB aux évènements déclenchés par l'utilisateur (clique sur un bouton,...). Il s'agit, comme son nom l'indique, d'un langage proche de Java, mais aucunement aussi complet. Il permet seulement de modifier et contrôler le contenu d'un document affiché dans un navigateur.

Ce langage sera utilisé pour vérifier certaines contraintes lors de la saisie de donnée. Ces opérations auraient pu être réalisées par un script PHP mais il aurait fallu passer par le serveur, ce qui aurait engendré des performances moindres.

5. Smarty

Smarty est un moteur de template (http://smarty.php.net/). Il s'agit d'une classe PHP dont le but est de séparer la logique applicative de la présentation. Ce genre de système permet une

maintenance plus facile et plus claire. Grâce à cette classe, les traitements sont effectués dans les scripts PHP et les informations à afficher sont indiquées à l'instance de cette classe ainsi que le fichier contenant l'organisation de ces informations (le fichier « template » contenant le XHTML).

Ainsi lors de modification ne concernant que l'interface, il n'est pas nécessaire d'aller dans les scripts PHP, et inversement.

v. Description des fonctions PHP

Toujours dans un soucis d'aider le développement, j'ai décrit le comportement de certaines fonctions qu'il fallait développer : ajout de matériel, recherche, …

En voici un exemple :

(entre paranthèses sont spécifiés les arguments.

ajoutPersonne(tous les champs de la table personne sauf mail (généré auto))

Responsabilités : Créer une nouvelle personne dans la base avec les paramètres spécifiés

Exceptions :

- Si une personne qui a le même login existe → échoue.
- Si le login ou le pass ont plus de 8 caractères → échoue.
- Si statut <> TITULAIRE ou STAGIAIRE ou MO ou THESARD ou autre valable → échoue.
- Si date_deb >= date_fin → échoue (attention au changement d'année !).
- Si isadmin=TRUE et pass =NULL → échoue.
- Si resp_login est non vide ET ne correspond pas à un login dans la base --> échoue.
- Si resp_login et login sont égaux

Type d'accès : INSERT sur la table PERSONNE.

Description détaillé :

L'adresse mail est généré automatiquement à partir du login de la manière suivante :

login@avignon.inra.fr

Figure 6 : description d'une fonction

Difficulté de maintenance

Le fait de vouloir décrire l'algorithme des fonctions développées dans le document de conception a posé une difficulté : celle de maintenir ce document en cohérence avec le développement. Car à l'algorithme initial, des modifications, en dehors des problèmes d'implémentation du langage, ont pu être effectuées (par exemple : le rapprochement des applications de gestion de sauvegarde et de gestion de parc à entraîné que l'exception ***« si isadmin=true et pass=null →échoue »*** soit enlevé car tous les utilisateurs ont un mot de passe permettant d'accéder à l'application de gestion de sauvegarde et si ils ont le statut administrateur d'accéderà la partie administration de la gestion de parc).J'ai donc essayé de répercuter au mieux les modifications apportées aux fonctions dans le développement (arguments, exceptions,…), mais pour une question de temps, seule les fonctions complexes (accès à plusieurs tables…) apparaissent dans le document.

d. Rapprochement de deux applications

i. Motivations

Au travers de différentes discussions et réflexions avec le stagiaire développant l'application de gestion des sauvegardes (*voir le rapport de Marc Serra pour plus de détail*), il est apparu que les applications « gestion du parc informatique » et « système de sauvegarde des ressources informatiques de l'unité » étaient liées. En effet, dans le système de sauvegarde, la gestion des logins utilisateurs, ainsi que celle des machines est identique à celle de la gestion du parc informatique.

Il a donc été décidé – après accord avec les membres du groupe SOSI – de faire le rapprochement entre ces deux applications, au niveau de la base de donnée mais aussi au niveau de l'interface, développées avec la même technologie (PHP) et de faire la partie « administration » commune. Représentant une grosse partie de mon application, j'ai réalisé le développement de cette partie commune, en accord avec Marc Serra (pour coordoner la définition des noms de variables transitant entre les deux applications, se mettre d'accord sur des éléments graphiques,... par exemple).

ii. Impacts

Pour les utilisateurs, cela est transparent, quant aux administrateurs il n'ont qu'une seule base à maintenir à jour et donc moins d'opérations à effectuer. Pour les développeurs, cela évite le développement de deux parties administrateurs mais aussi la création de deux bases distinctes (*voir annexe 8 : fusion de la base de données, le MCD global*). Techniquement, la base de données du système de sauvegarde est intégrée à la base de donnée de la gestion de parc. Quant aux interfaces elles communiquent entre elles par le biais de liens entre les différentes parties.

e. Développement

i. Remplissage de la base de donnée

Les SOSIS tiennent actuellement à jour l'inventaire de leur parc dans des fichiers « Excel ». Pour éviter la saisie manuelle de l'ensemble de ces informations, j'ai réalisé un script permettant d'importer les données de ces fichiers, un par bâtiment. Ces scripts permettent l'insertion dans les différentes tables de la base de données grâce à la lecture des fichiers au format ASCII.

L'importation de données existantes a permis de vérifier la bonne cohérence de la base mais aussi de quantifier le volume des données contenues dans la base de données : **143** enregistrements dans la table COMPOSANT, **1192** dans la table CAR_COMP et **90** dans la table PERSONNE).

Difficultés d'importation

L'unité étant divisée en deux, une personne s'est occupée du recensement des machines du bâtiment SOL et une autre du bâtiment Climat pour fournir deux documents Excel. Malgré des indications sur le format des données que je souhaitais : valeurs possibles pour les OS, adresse MAC, valeur des vitesse CPU,... il y avait des différences dans les deux fichiers Excel, ce qui m'a obligé à vérifier tout d'abord manuellement ces fichiers afin qu'il soit importables (suppression des accents dans les login par exemple).

ii. Personnalisation de l'affichage

La liste des matériels comportant beaucoup de colonnes (c'est à dire que le nombre de caractéristiques associées au matériel de type « PC » est très important), il a été décidé de créer des profils « utilisateurs » pour personnaliser l'affichage de l'information. C'est à dire que chaque administrateur choisira les caractéristiques à afficher pour tel type de matériel. Ce besoin ayant été découvert lors du développement, je n'ai pas eu le temps de faire un retour sur l'analyse comme il aurait été nécessaire et avec l'accord des SOSIS j'ai donc choisi la solution de créer des fichiers de préférences que l'on peut créer, modifier ou supprimer.
Ainsi un administrateur peut facilement personnaliser son affichage et lorsqu'il se logue ses préférences sont prises en compte automatiquement (le fichier de préférence est utilisé pour construire la liste du matériel).

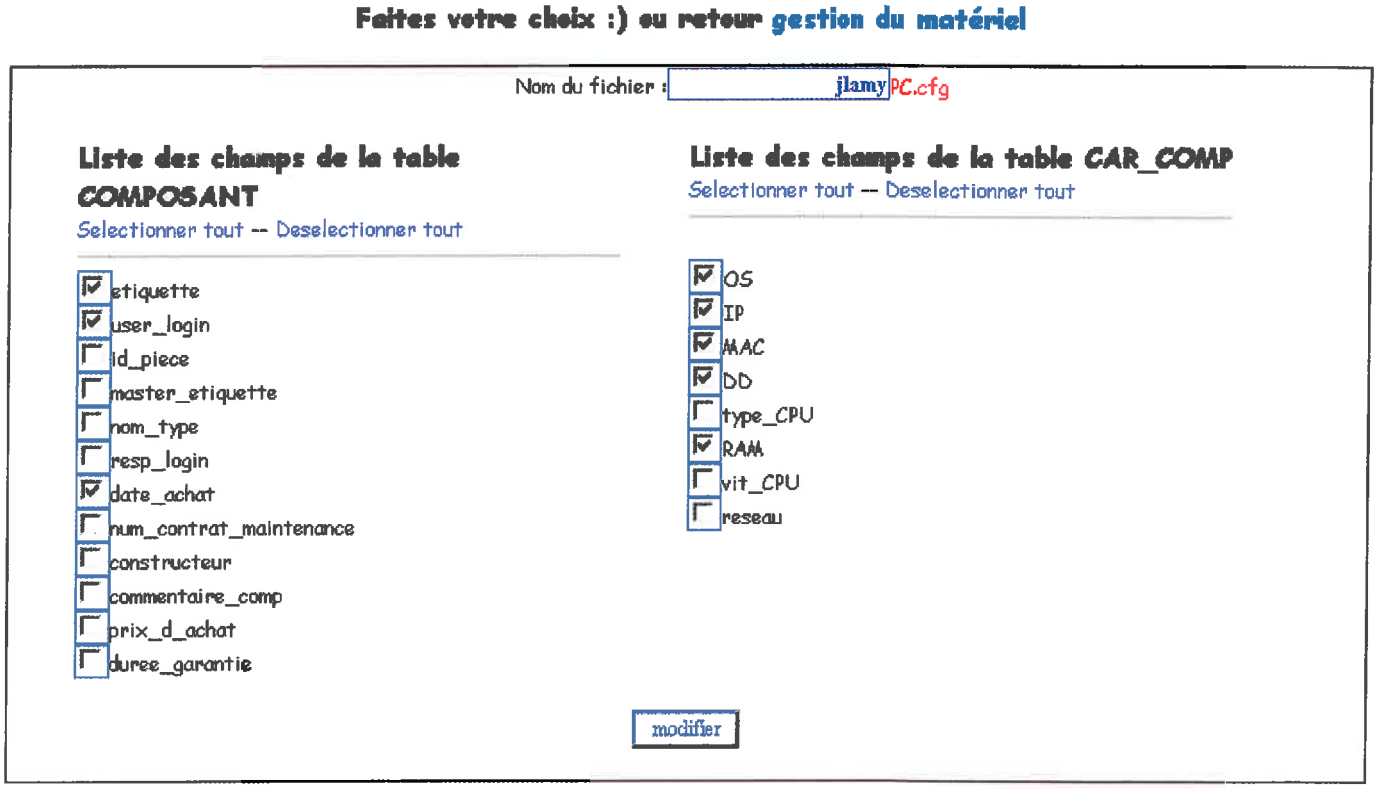


Figure 7 : fichier de personnalisation de l'affichage

Cette capture d'écran (**figure 7**) montre comment on créé un fichier texte de préférence pour un type de matériel (ici le type étant PC). On coche tous les champs que l'on souhaite voir affiché puis on clique sur modifier (ou créer si c'était pour une création, ici il s'agit en fait d'une modification).

Problème soulevé par ces fichiers de configurations

Cette fonctionnalité a posé deux type de problèmes, un simple a régler, un autre plus complexe.
Cas Simple : Lors de la suppression d'un type de matériel, l'ensemble des fichiers concernant ce type est supprimé.
Cas Complexe : En cas de suppression d'une caractéristiques (par exemple RAM sur la figure 7), il faudrait que cette caractéristique ne soit plus dans les fichiers dans lesquels elle se trouvait. Par manque de temps il m'a été impossible de développer une solution très propre : celle de parcourir chaque fichier et de supprimer la ligne correspondante. J'ai opté pour la solution qui consiste à

vérifier à la lecture du fichier de configuration que la caractéristique lu est bien présente dans la base de données.

iii. Développement de fonctions de modification d'informations

Les données à afficher sont extraites de la base de donnée, traitées puis mise en forme pour être affichées, modifiées et retraitées pour enregistrement dans la base de donnée.

Je vous propose ici un exemple d'une fonction développée permettant la modification d'informations, du PHP au XHTML en partant du résultat. Cette fonction permet la modification des caractéristiques d'une machine existante dans la base de données. La réalisation de cette fonction fait intervenir l'ensemble des outils techniques décrites dans la figure 5 (*utilisation et liens entres les technologies*).

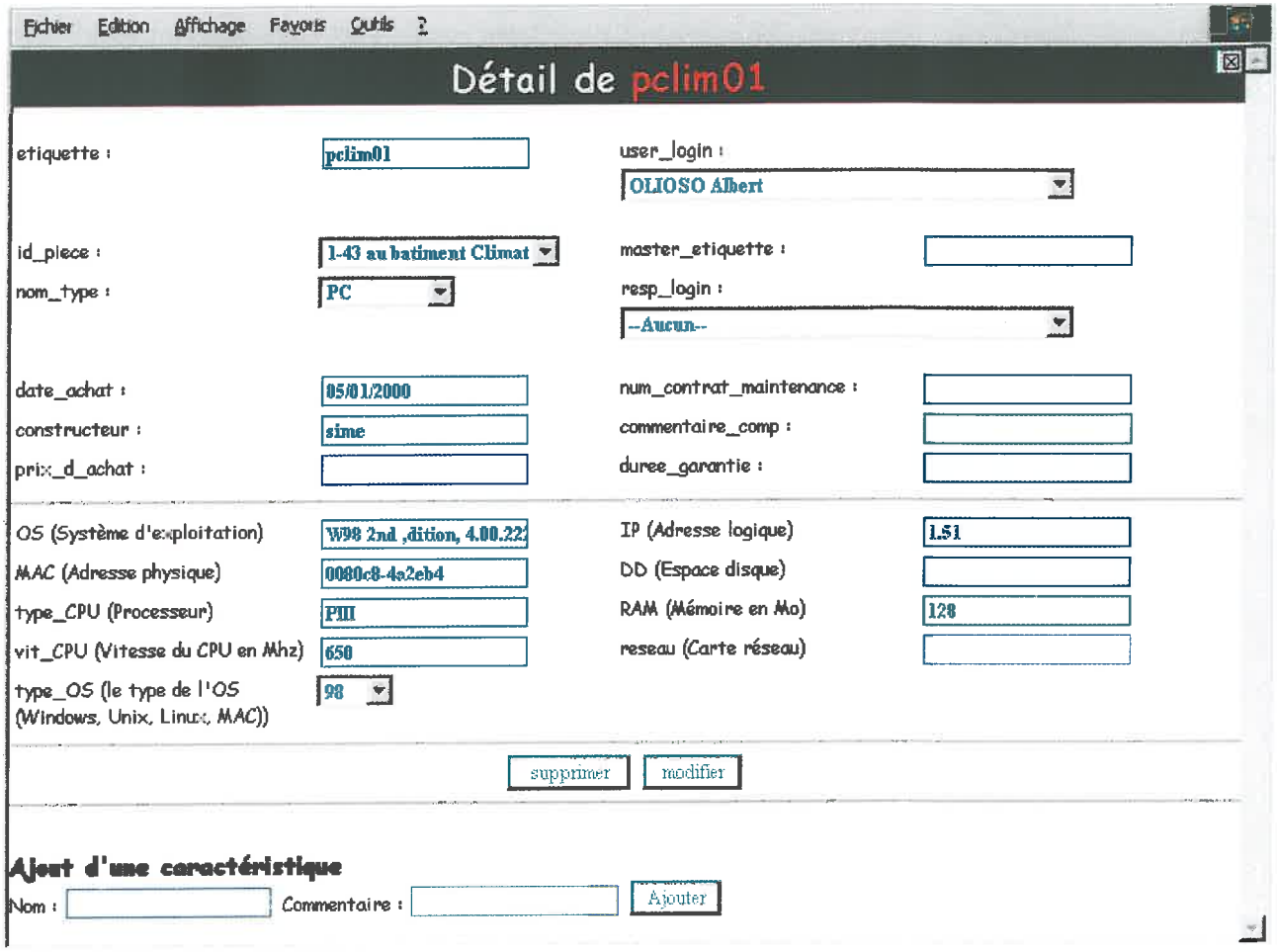


Figure 8 : aperçu de l'écran de modification d'une machine existante

Affichage

Lorsque un administrateur demande l'affichage d'une machine (d'un PC par exemple), une nouvelle fenêtre s'ouvre (une « popup ») permettant directement la modification des informations sur la machine. Sur la capture d'écran les boites de textes situées en dessous du premier traits (à partir de OS), correspondent aux caractéristiques qui dépendant du type de matériel. Ici ces caractéristiques sont celles d'un PC. En bas de l'écran il y a la possibilité d'accéder directement à la fonction d'ajout d'une nouvelle caractéristique au type de matériel PC. Puis les deux boutons Supprimer et Modifier font appel aux procédures de suppression et de modification.

L'organisation de cette fenêtre est décrite grâce au fichier template en XHTML, le style étant défini dans une feuille de style CSS2 séparée que j'ai créé.

Voici un petit extrait du fichier template XHTML de la modification d'une machine.

Par exemple pour l'affichage de l'étiquette, la syntaxe « XHTML » sera la suivante :

```
<input type="text" name="machine[{$key}]"  id="machine[{$key}]"
value="{$item}" onblur="verif_etiquette(this);" />
```

L'attribut ***type*** définit le type de donnée, ***name*** indique le nom de la variable que va récupérer le script PHP après validation, l'***id*** sert aux vérifications Javascript, ***onblur*** indique la gestion de l'événement « sur perte du focus » en Javascript.

Prenons maintenant le bouton modifier :

```
<input type="submit" class="bouton" value="modifier" name="validation"
onclick="if(modif_matos())return true ; return false" />
```

Ce bouton de type ***submit*** en fonction du résultat de la fonction Javascript modif_matos() valide le formulaire.
L'attribut ***class*** désigne l'élément dans la feuille de style qui sera utilisé pour l'affichage du bouton
Ces éléments se trouvant à l'intérieur de la balise ***form*** :

```
<form action="matdetail.php" method="post" name="form" id="form">
...
</form>
```

l'attribut ***action*** correspond au nom du script appelé lors du clic sur un bouton de type ***submit*** ou plus exactement lorsque l'évènement ***submit*** survient.

Le fichier css2 explicitera les points suivant : style, taille, mise en page (centré, gauche…) de la balise texte…
Par exemple voici la définition de l'affichage d'un bouton

```
input.bouton
{
  color : #006699;
  text-align : center;
  background : #EEEEEE ;
  border : 2px #006699;
  border-style : outset;
  font-weight:normal;
  cursor : pointer;
}
```

Ici, ce qui est intéressant de voir est que l'on peut définir le type de curseur (***cursor***) utilisé lorsque la souris se trouve au dessus d'un élément de cette classe là.

Vérification des données

Lorsque l'utilisateur clique sur le bouton modifier, une vérification sur le format de certaines données (dates, caractères spéciaux,…) est effectuée grâce à du Javascript.
Voici un exemple d'une fonction Javascript simple qui vérifie si une date est valide :

```
function verif_achat(obj) //obj designe l'objet champ texte de la date_achat
{
//teste qu'un probleme dans la saisie de la date, donc message d'alerte
if(obj.value!="" && isDate(obj.value)==false)
  {
      alert("GPI a dit : la date d'achat n'est pas valide");
      obj.focus(); //replace le focus sur le champ date_achat
      obj.select();//selectionne le texte saisie precedemment
      return false;
```

```
  }
  return true;
//-----------------------------------------------------------------------
```

Traitements et modifications dans la base de données

Enfin les données sont traitées dans un script PHP : recherche dans la base de données des informations à modifier puis si les informations sont trouvées, il y a modification des données. Les recherches, modifications, ajout, suppressions…des informations dans la base de données sont réalisées par des requêtes SQL.
Voici comment la requête de modification d'un matériel est construite :

```
  $query = "UPDATE COMPOSANT SET ";
//$machine correspond au tableau contenant le doublon nom_champ / valeur
//a ajouter dans la table COMPOSANT
//Pour chaque element de $machine
  while((($elem = each($machine))!=FALSE)
  {
    $value = $elem[value];

    $query = $query." ,".$elem[key]."=".(empty($elem[value])==TRUE ? "NULL"
:"'$value'");
  }
  $query = $query." where etiquette='$val_etiquette'";
//Ici on declare une connection à la base de donnée
//en utilisant une classe php nommé DB
  $dblink = new DB;
//Execution de la requete
  $dblink->query($query);
```

Cette exemple montre l'utilisation d'une classe nommé DB. L'avantage de cette classe est de permettre la connexion et l'utilisation de différents SGBD en utilisant toujours les mêmes fonctions, ce qui n'est pas le cas en PHP (*pg_connect* pour postgreSQL et *mysql_connect* pour MySQL)

iv. Développement de fonctions d'affichage : la liste des personnes

Les données à afficher sont extraites de la base de donnée, traitées, mise en forme puis affichées. Aucune modification des données n'est nécessaire.
Dans l'exemple de l'affichage de la liste des personnes, je vous montre toutes les étapes de cette fonction et j'insiste sur l'utilisation de la classe PHP Smarty, le moteur de templates qui permet de séparer la présentation (XHTML) et le traitement (PHP).

Etape 1 : la requête SQL simple

Grâce à la classe DB, la requête permettant de récupérer la liste des personnes est exécuté.
Etape 2 : Affichage de la liste (utilisation de la classe Smarty)

PHP (dans le script persadmin.php)

Cette exemple de code PHP montre l'utilisation de la classe Smarty.

```
//Appel a la fonction
$tabPers = recupTousPers($tri,$_SESSION["tri_style"]);
//Indique a l'instance de smarty une nouvelle variable que l'on utilisera
//dans le template persadmin.tpl
```

```
  $smarty->assign("tabPers",$tabPers);
//Indique a l'instance de smarty qu'il faut afficher la page grace au template
passé en argument
  $smarty->display('./admin/persadmin.tpl');
```

XHTML (dans le fichier persadmin.tpl)

Cet exemple montre lui les particularités de Smarty au niveau XHTML

```
//Declaration du tableau, de ces attributs, espacement des cellules,…

  <table class="liste" id="table" width="100%" style="border-collapse :
collapse" cellspacing="0px" cellpadding="4px">

//En tete du tableau, contient le nom des colonnes

  <thead>
    <tr><th class="liste">Login</th><th class="liste">Nom</th><th
class="liste">Prénom</th><th class="liste">Responsable</th><th
class="liste">Mail</th><th class="liste">Statut</th><th class="liste">Date
d'arrivée</th><th class="liste">Date de départ</th>
    </tr>
  </thead>

//Fin entete
//Corps du tableau XHTML, contient les données extraites de la base de donnée

  <tbody>

//Boucle foreach qui parcours le tableau $tabpers qui contient les données
récupérées
//Syntaxe particuliere à Smarty {}

  {foreach from=$tabPers item=current_pers name="pers"}
    <tr>
      <td><span style="color : #006699">{$current_pers.login}</span></td>
      <td>{$current_pers.nom}</td>
      <td>{if $current_pers.prenom!=""} {$current_pers.prenom} {else}  
{/if} </td>
      <td>{if $current_pers.resp_login!=""}{$current_pers.resp_login}{else}<span
style="font-style : italic ; color :red ">Aucun</span>{/if}</td>
      <td align="center"><a
href="mailto:{$current_pers.mail}">{$current_pers.mail}</a></td>
      <td>{$current_pers.statut}</td><td>{$current_pers.date_deb}</td>
      <td>{$current_pers.date_fin}</td>
    </tr>
  {/foreach}
</tbody>

//Pied de page du tableau

 <tfoot>
  </tfoot>
  </table>
```

Sur la figure suivante, vous pouvez voir le résultat de l'affichage du tableau de la liste des personnes :

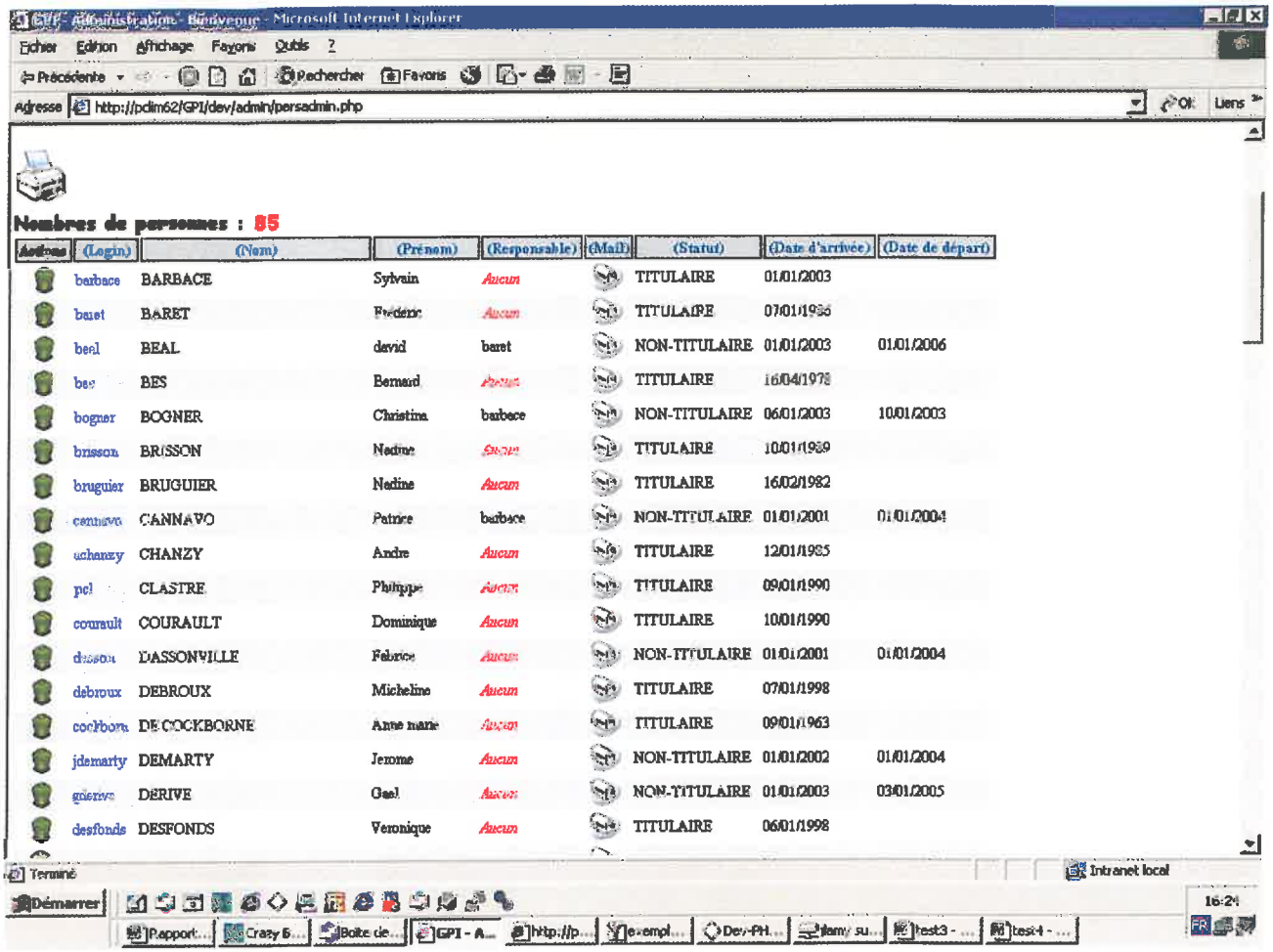


Figure 9 : aperçu de liste de personnes

v. Impressions

Les utilisateurs de l'application voulaient pouvoir imprimer les différentes listes (matériels ou personnes). J'ai donc pris en compte ce besoin dans le développement (*voir figure9 : l'icône représentant une imprimante*). En cliquant sur l'icône de l'imprimante, une nouvelle page s'ouvre n'affichant que les données à imprimer, permettant ensuite de faire un aperçu avant l'impression pour être sûr de l'orientation de page à choisir. Voici cette page :

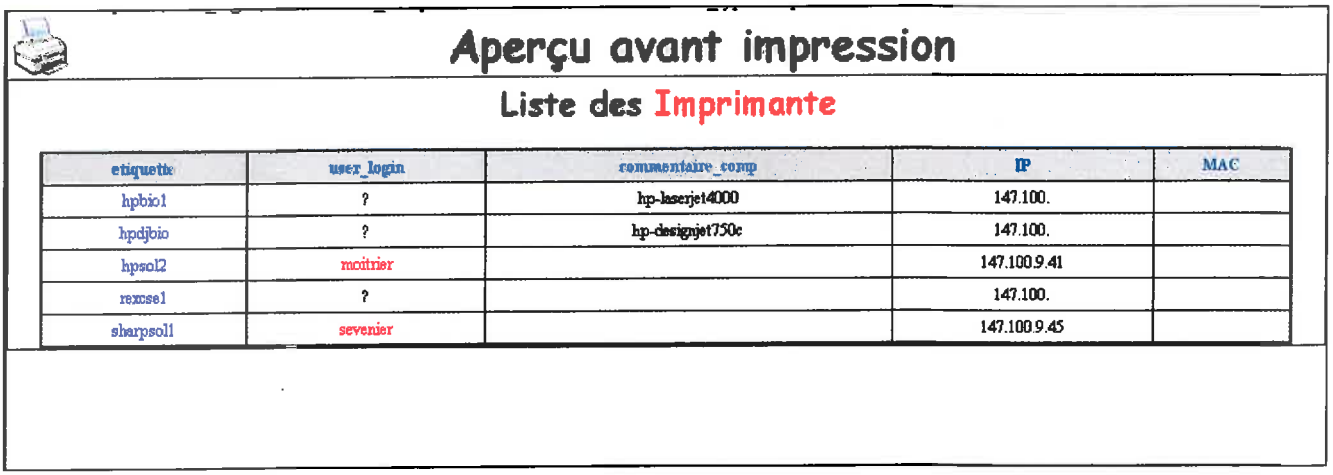


Figure 10 : exemple d'impression

En cliquant de nouveau sur l'imprimante dans la page représenté figure 10, la boite de dialogue d'impression propre au Navigateur s'ouvre.

vi. Tests

1. Tests unitaires

Durant le processus de développement, j'ai testé chaque fonction puis chaque script que je développai. Ces tests consistent à vérifier que les fonctions réalisent bien ce qu'elles sont censées faire, que les contraintes spécifiées dans le document de conception sur les paramètres sont bien vérifiées (au niveau Javascript comme PHP).
Voici un exemple de test réalisé :

Ajout d'une personne, il faut vérifier an niveau PHP que :

- Lorsque l'on entre un login déjà présent dans la base de donnée il nous retourne une erreur
- Lorsque l'on entre un login de responsable non présent dans la base de données, il nous retourne une erreur.

Au niveau Javascript, les contraintes suivantes sont à vérifier :

- Le login ne doit pas dépasser les 8 caractères.
- La date de départ ne doit pas être inférieure à la date d'arrivée.
- Le login du responsable ne peut pas être le login de la personne que l'on ajoute

De plus, plus généralement, il faut vérifier qu'à l'exécution des fonctions Javascript et PHP de l'ajout d'une personne, cette personne est réellement ajoutée dans la base de données (succès de la requête SQL).

Tous ces tests ont permis de détecter et corriger un certains nombres d'erreurs.

vii. Tests d'intégration

Après les tests unitaires, chaque fonctionnalité est testée dans le cadre d'une utilisation normale de l'application. Ces tests permettent de vérifier le comportement entres chaque parties de l'application et de connaître les conflits dus à l'interaction de ces parties.

viii. Tests finaux

Lorsqu'une partie de l'application était fonctionnelle, elle était soumise aux tests des membres du groupe SOSI. Ce genre de test permet de se rapprocher d'une utilisation normale de l'application. On peut ainsi vérifier son bon fonctionnement, sa facilité d'utilisation, son adéquation avec les besoins exprimés,…

Ces tests ont été très efficaces et ont permis de corriger un certains nombres de bugs, d'erreurs ou d'oublis mais aussi d'améliorer l'interface, de faire émerger de nouveau besoins (par exemple la possibilité de personnaliser son affichage résulte besoin exprimé après un test qui a suivi l'intégration des données complètes dans la base et qui nécéssitent un « scroll » systématique pour voir tous les champs intéressants).

ix. Tests multi plateformes

Etant donné que l'application devait fonctionner aussi bien sous Linux, Windows avec les navigateurs Netscape, Internet Explorer 4 minimum et Mozilla 1.3 minimum, durant tout le développement j'ai effectué des tests en parallèle sur les différents OS et Navigateurs surtout pour

vérifier le bon fonctionnement des fonctions Javascript (les navigateurs ont malheureusement tendance à ne pas interpréter de la même manière certaines fonctionnalités Javascript).

f. Documentation

Un soin particulier a été apporté à la documentation. Tout d'abord j'ai effectué une mise à jour du document d'analyse. Puis j'ai réalisé les documents suivants : Document de conception, manuel de l'utilisateur et documentation technique. De plus, un CD a été réalisé contenant les outils utilisés, les sources de l'application, et les différentes documentations.

i. Document de conception

Il s'agit d'un document de conception générale et détaillée. Il explique les choix fait pour le développement et détaille certaines phases : création de la base de donnée, algorithme de certaines fonctions, définition de l'interface graphique,... (*Voir annexe 9 : le plan du document de conception*)

ii. Manuel de l'utilisateur

Il s'agit de la documentation sur l'application réalisée : son fonctionnement, sa présentation, ses objectifs,... A noter que pour l'instant les utilisateurs visés ne sont que ceux qui ont le statut d'administrateur (notamment le groupe SOSI). La partie pour les utilisateurs sans privilège sera faite lors du développement de la partie concernant la gestion des problèmes (faisant intervenir les utilisateurs « simples »).

Ce document est disponible sous plusieurs formats : Acrobat Reader (PDF), Word, ou consultable sur Internet (XML).

iii. Documentation technique

Afin de faciliter la tâche des personnes amenées à s'occuper de la maintenance de l'application mais aussi du développement d'autres parties de l'application, j'ai réalisé une documentation technique contenant les éléments suivant :

- Les références de tous les outils utilisés : URL, livres, versions des outils...
- Explications de l'installation du serveur Apache, de PostgreSQL, de PHP et de Smarty
- Description de l'arborescence des sources de l'application (contraintes à respecter)
- Liste des sources (expliquant le rôle de chaque fichier)
- Explications de la démarche de développement avec Smarty
- Explications sur l'ajout d'éléments

De même que le manuel de l'utilisateur, ce document est disponible sous plusieurs formats.

5.Bilan du stage

a. Finalité du projet

Le groupe SOSI dispose d'une application WEB testée dont l'interface a été conçue en accord avec leurs besoins et qui leur permet en tant qu'administrateur :

- D'avoir une vue personnalisée de l'ensemble du parc informatique classé par type de matériel (PC, Imprimantes, Serveurs,…)
- De pouvoir modifier ce parc, c'est-à-dire rajouter des machines, les modifier, les supprimer mais aussi ajouter un nouveau type de matériel (exemple : vidéo projecteur) ou encore ajouter une nouvelle caractéristique pour un type de matériel (exemple : carte mère pour les PC).
- D'avoir une gestion efficace des logins des utilisateurs (ajout, suppression, modification, recherche) de l'application de sauvegarde et plus tard de l'application elle-même (partie maintenance accessible aux utilisateurs).
- D'avoir des statistiques sur le parc (tableaux de bord) et de pouvoir en rajouter facilement grâce à la documentation.

b. Pourcentage d'accomplissement de la mission

Les objectifs suivants ont été atteints :

- Gestions des logins (ajout, suppression, modification, filtres, affichage, recherche)
- Gestion du matériel (ajout, suppression, modification, affichages, recherche)
- Gestion des types de matériel (ajout, suppression, modification,…)
- Préférences d'affichages

Les objectifs fixés pour le stage et non atteints sont :

- Gestion de la maintenance (problèmes et interventions)

Pour donner un pourcentage d'accomplissement je dirais 90% des objectifs ont été atteints. Pour la gestion de la maintenance, la documentation technique permettra d'aider une personne à la concevoir et j'estime le temps de développement à environ 1 mois (définitions des aspects graphiques avec le groupe SOSI, développement, tests des utilisateurs simples, documentation adapté)

c. Apports pour l'INRA

Du fait de la répartition en deux bâtiments, cette application permet d'avoir une centralisation de la gestion du parc informatique. Plusieurs points positifs sont à noter :

- Homogénéisation du fonctionnement des SOSI (ils travaillent avec les informations à jour et peuvent intervenir plus facilement sur une machine en connaissance de cause)
- Accès à l'état du parc en temps réel (permet par exemple une meilleur gestion des achats car ils peuvent avoir rapidement des informations sur les machines anciennes, plus sous garanties,…)

- Accessibilité depuis n'importe quel poste : ce qui simplifie les procédures de gestion, et est un moyen de garantir une pérennité de l'information et de l'application (« au plus c'est simple, au plus ça vie ! »).

d. Apports personnels

Grâce à ce projet j'ai pu avoir une expérience concrète de la réalisation d'une application complète c'est à dire la réalisation de toutes les étapes couvrant un projet informatique. En effet, en tenant compte du fait que j'ai débuté ce projet lors de mon projet de fin d'étude, j'ai donc effectué toutes les étapes d'analyse, de conception, de développement, de tests et de documentation, ce sur quasiment une année.

J'ai du faire preuve d'organisation, de rigueur, mais aussi de communication afin de proposer une solution adaptée aux besoins du groupe SOSI.

Sur le plan des connaissances techniques, j'ai pu approfondir mes connaissances en PHP, XHTML, Javascript, CSS, Base de données (SQL) mais aussi apprendre à installer et configurer des serveurs Apaches, des SGBD (PostgreSQL).

Cette expérience a été très bénéfique pour moi car je souhaite m'orienter dans le domaine des bases de donnée et du WEB. De plus j'ai beaucoup apprécié le fait de travailler avec des personnes non issues du milieu informatique. Je serais très motivé à l'idée de renouveler une telle expérience

Annexes

ANNEXE 1

Planning prévisionnel et réel

(P = Prévisionnel ; R = Réel)

Tâches		Avril		Mai				Juin				Juillet				Août				Septembre	
Finalisation Analyse	P																				
	R																				
Plate-forme de développement (recherches des outils et installation	P																				
	R																				
Création base de donnée + remplissage automatique	P																				
	R																				
Conception	P																				
	R																				
Maquettes de démo	P																				
	R																				
Développement	P																				
	R																				
Tests unitaires	P																				
	R																				
Tests d'intégration	P																				
	R																				
Tests multi plateformes	P																				
	R																				
Tests finaux	P																				
	R																				
Documentation et rapport de stage	P																				
	R																				
Congés	P																				
	R																				
Préparation soutenance	P																				
	R																				

ANNEXE 2

Exemples de comptes rendu de réunions
Recadrage des objectifs

Compte rendu : Réunion du vendredi 27 juin 2003-06-30

Objet

Réunion d'avancement du projet. Présentation de la maquette de l'application et discussion à son sujet.

Personnes présentes

- Nathalie MOITRIER : mon maître de stage, membre du groupe SOSI et initiatrice du projet
- Philippe CLASTRE : membre du groupe SOSI et initiateur du projet
- Sylvain BARBACE : administrateur réseau de l'unité et principal utilisateur de la future application

Notes

Bugs détectés

Au niveau de la modification d'une personne : problème avec les dates, vérifier que l'on travaille sur les bonnes dates.

Ajout de matériel

Il serait intéressant de regrouper les écrans pour l'ajout de matériel. Essayer d'éviter au maximum que l'utilisateur est à changer plusieurs fois de page.

2 solutions : soit choix du type dans une liste puis clique sur Ajouter un matériel et Ouverture du formulaire d'ajout de tel matériel. Soit clique sur un lien Ajouter puis choix du type sur une ligne et MAJ du formulaire d'ajout (par défaut : Ajout d'un PC).

Affichage du matériel

Faire de l'affichage dynamique.

Il faut que l'utilisateur ait la possibilité de choisir les différents champs qu'il désire afficher et qu'il puisse le changer.

Solution : Fichiers de configuration (quelques-uns, dont un par défaut) qui contiennent la liste des variables à afficher.

MAJ du doc Analyse / MAJ du doc de Conception

Pour l'instant, le temps est insuffisant pour mettre à jour le document d'analyse. Par contre, le document de Conception doit être mis à jour afin de contenir une explication de la présentation dynamique de l'information ainsi que des choix possibles.

Statistiques sur le parc

Il est important que l'application permette de faire des statistiques sur le parc. En voici une liste non exhaustive :

- Nombre de machines
- Moyennes des CPU
- Moyenne de HD
- Prix total des machines

Filtres sur la liste des personnes

Certains filtres doivent être disponibles pour la visualisation des personnes : Administrateur, Statut, Responsable, Nom.

Documentation technique

La documentation technique devra expliquer le développement en tenant surtout compte du fait qu'une personne désirant par exemple rajouter un filtre sur l'affichage des personnes puisse être capable de le faire en le lisant.

Compte rendu réunion du 25 juillet 2003

Personnes présentes :

Philippe CLASTRE
Sylvain BARBACE
Jean LAMY

Points abordés :

Avancement du projet

Objectifs fixé pour la suite :

- Finir les fichiers de préférences : si il existe un profil portant notre nom ce profil est affiché par défaut.
- Formulaire pour éditer ou créer un fichier de préférence.
- Suppression d'un type de matériel
- Impression de liste de matériel
- Recherche de matériel
- Tableaux de bord (penser à bien expliquer la marche a suivre pour en créer de nouveaux.
- Facultatifs : Mail a une liste de personne.
- Documentation : Utilisateur, technique

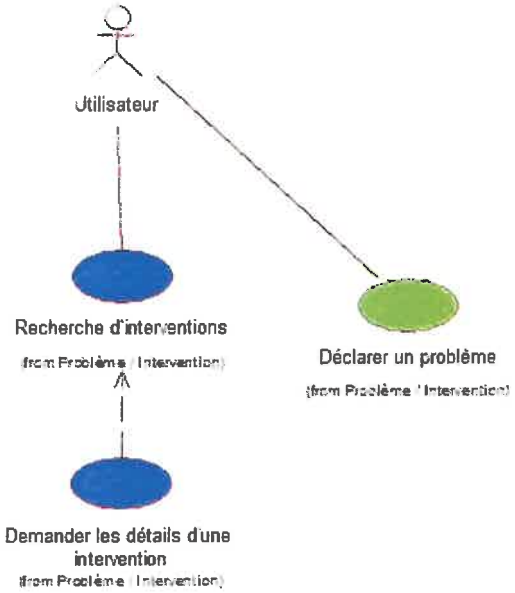
ANNEXE 3

Plan du document d'analyse

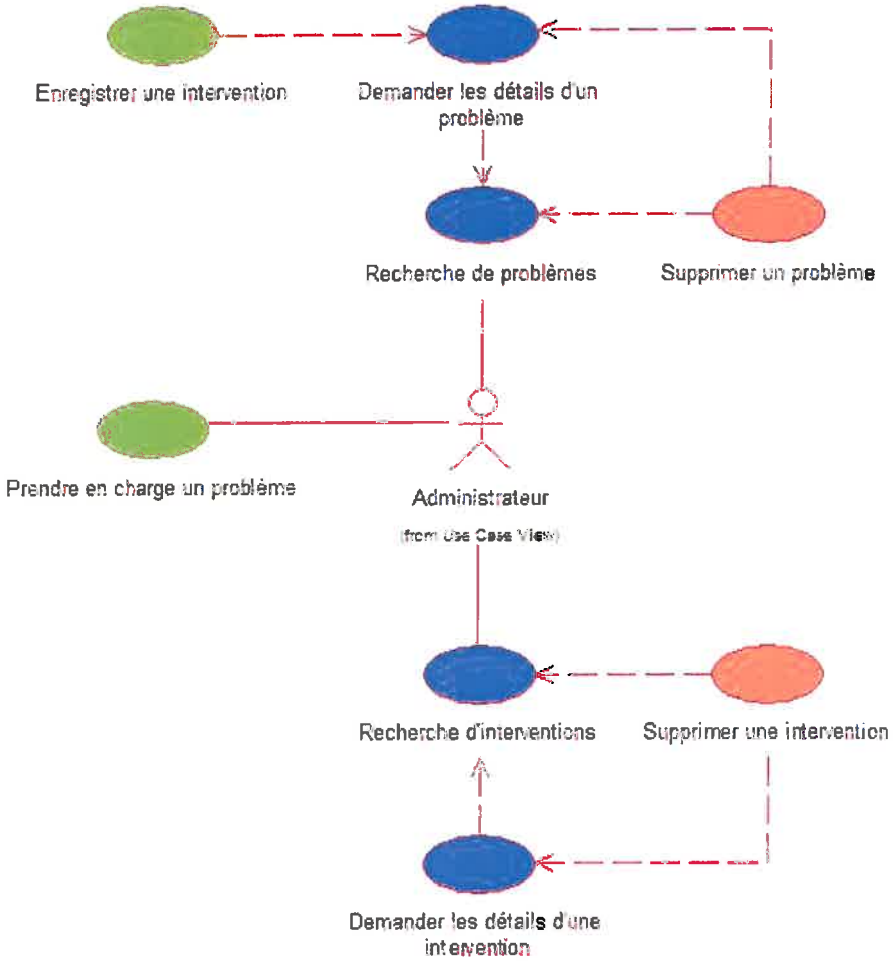
ANNEXE 4

diagramme utilisateurs

Utilisateur :



Administrateur (uniquement sur les problèmes) :



ANNEXE 5

Les licences libres

Licences GPL

Licence rédigée par Richard STALLMAN. Cette licence entraîne la libre diffusion gratuite d'un progiciel. Aucune entité ne peut donc « s'approprier » le code puisque ce dernier devient public. C'est un mode de licence où chacun est libre d'accéder, de diffuser, de modifier et de commercialiser un logiciel ouvert dès lors qu'il garantit lui aussi l'accès au code source de ce qu'il produit et respecte les copyright.

Licences BSD

Berkeley System (ou Software) Distribution (ou Design). L'université de Berkeley est connue dans le monde Unix pour les nombreux logiciels qu'elle a développé puis mis dans le domaine public, principalement en réaction contre les tarifs exorbitants pratiqués par AT&T. BSD désigne en particulier une famille de versions d'Unix issue de l'université de Berkeley en Californie, conçue pour le VAX de DEC et le PDP-11, qui a été ensuite portée sur les PC sous le nom 386BSD, puis de FreeBSD. D'autres implémentations existent : NetBSD, OpenBSD. Pendant longtemps, Berkeley et sa version de Unix furent à la pointe du milieu, avec l'introduction de nombreuses nouvelles fonctionnalités (e.g. vi, le C-shell...) et beaucoup de grands constructeurs ont basé leur propre version sur celui-ci. C'est a BSD que l'on doit les sockets (TCP/IP).

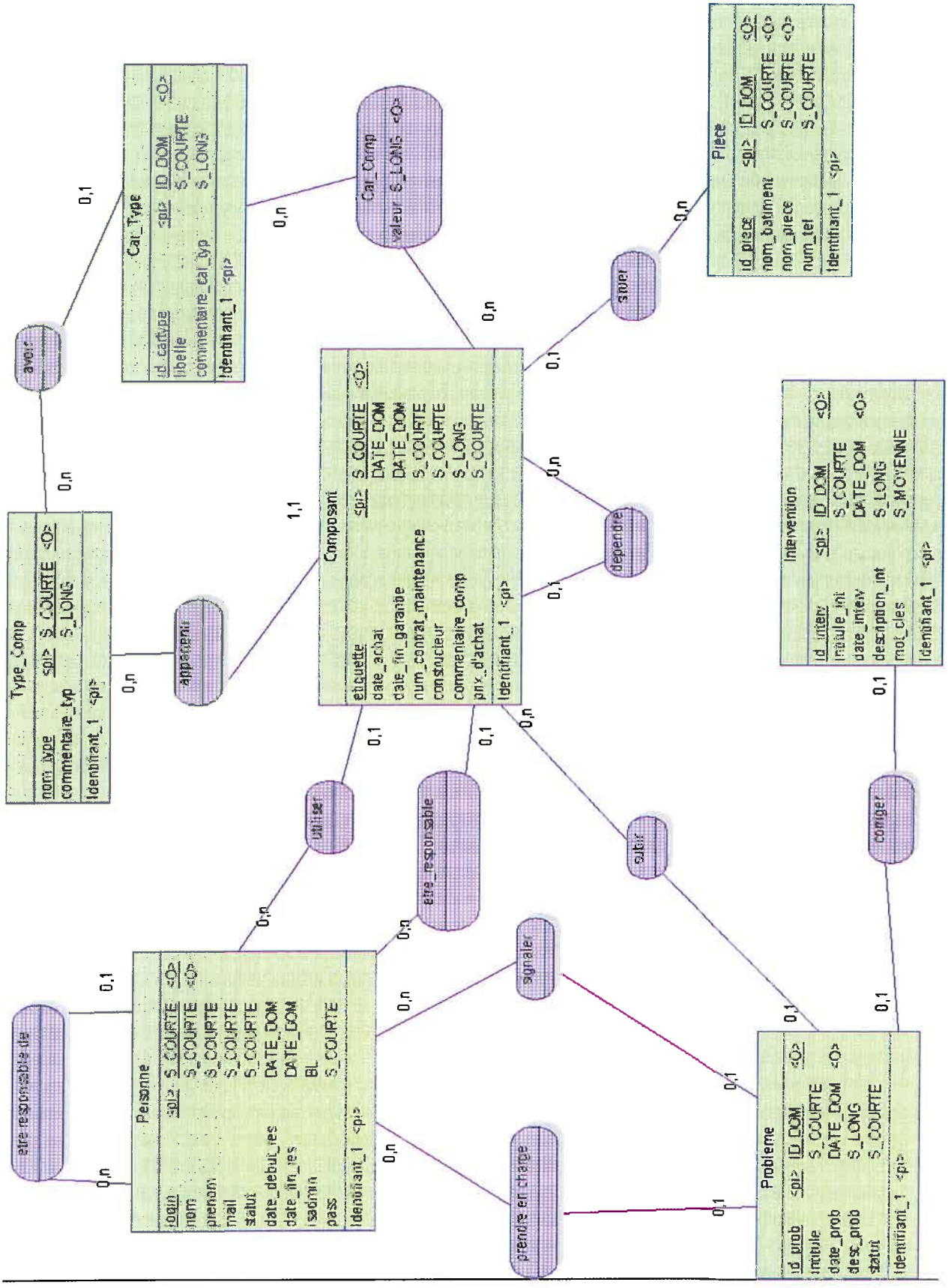
ANNEXE 6

Le Modèle Conceptuel de Données (MCD)

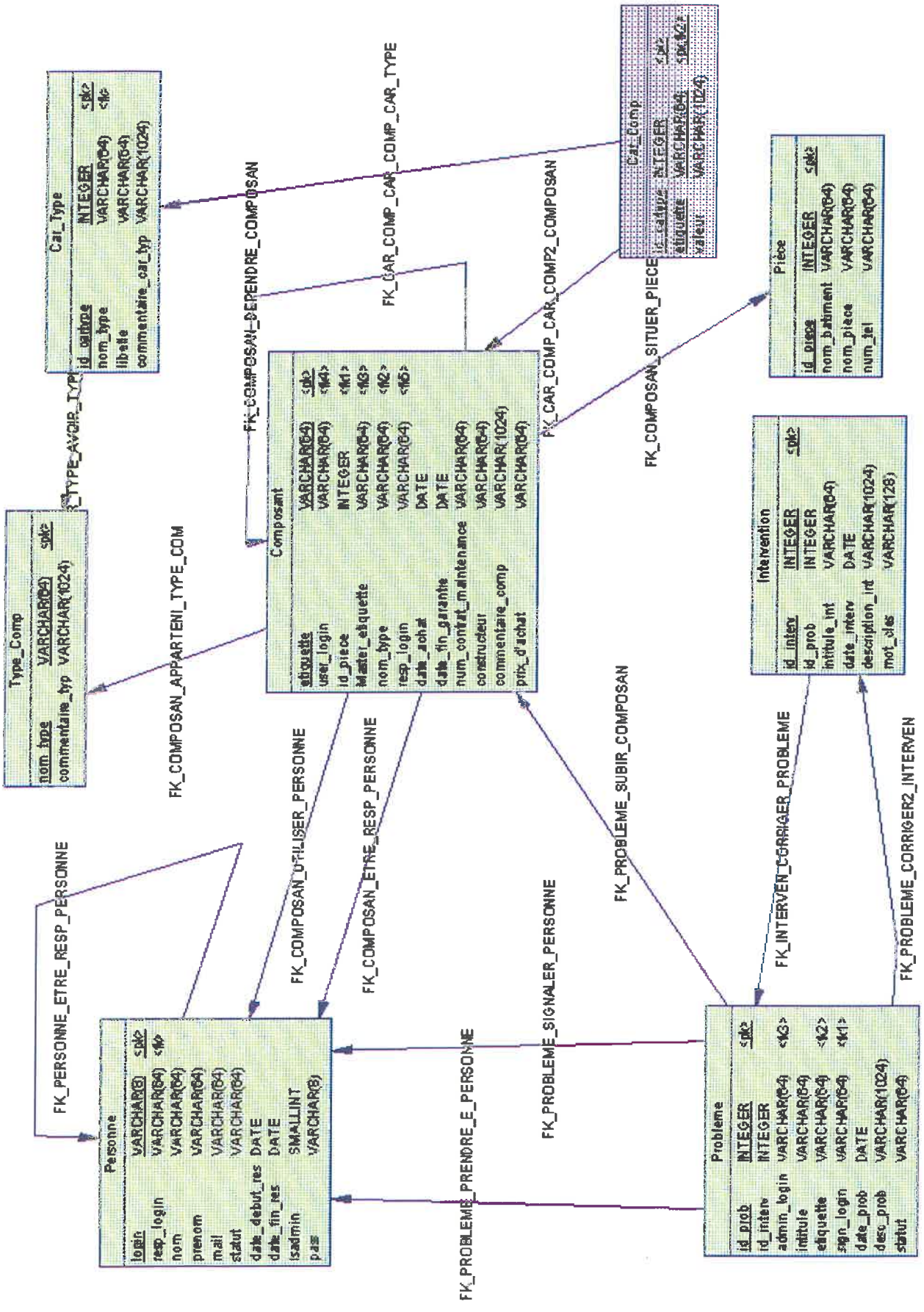
et

Le Modèle Physique de Données (MPD)

MCD


Type_Comp
nom_type
commentaire_typ
Identifiant_1
Car_Type
id_cartype
libelle
commentaire_car_typ
Identifiant_1
Car_Comp
valeur
Piece
id_piece
nom_batiment
nom_piece
num_tel
Identifiant_1
Composant
etiquette
date_achat
date_fin_garantie
num_contrat_maintenance
constructeur
commentaire_comp
prix_d'achat
Identifiant_1
Intervention
id_interv
intitule_int
date_interv
description_int
mot_cles
Identifiant_1
Personne
login
nom
prenom
mail
statut
date_debut_res
date_fin_res
isadmin
pass
Identifiant_1
Probleme
id_prob
intitule
date_prob
desc_prob
statut
Identifiant_1
avoir
appartenir
utiliser
etre_responsable
etre_responsable_de
signaler
prendre en charge
suivre
dependre
subir
corriger

MPD


Cat_Type
id_cattype
nom_type
libelle
commentaire_cat_typ
Car_Comp
id_caract
etiquette
valeur
Type_Comp
nom_type
commentaire_typ
Composant
etiquette
user_login
id_piece
master_etiquette
nom_type
resp_login
date_achat
date_fin_garantie
num_contrat_maintenance
constructeur
commentaire_comp
prix_d'achat
Piece
id_piece
nom_batiment
nom_piece
num_tel
Intervention
id_interv
id_prob
intitule_int
date_interv
description_int
mot_cles
Personne
login
resp_login
nom
prenom
mail
statut
date_debut_res
date_fin_res
isadmin
pass
Probleme
id_prob
id_interv
admin_login
intitule
etiquette
sign_login
date_prob
desc_prob
statut
FK_CAR_COMP_CAR_COMP_CAR_TYPE
FK_COMPOSAN_DEPENDRE_COMPOSAN
FK_CAR_COMP_CAR_COMP2_COMPOSAN
FK_COMPOSAN_SITUER_PIECE
FK_COMPOSAN_APPARTENI_TYPE_COM
FK_PERSONNE_ETRE_RESP_PERSONNE
FK_COMPOSAN_UTILISER_PERSONNE
FK_COMPOSAN_ETRE_RESP_PERSONNE
FK_PROBLEME_SUBIR_COMPOSAN
FK_INTERVEN_CORRIGER_PROBLEME
FK_PROBLEME_CORRIGER2_INTERVEN
FK_PROBLEME_SIGNALER_PERSONNE
FK_PROBLEME_PRENDRE_E_PERSONNE

ANNEXE 7

Scripts SQL

Ces scripts sont divisés en plusieurs fichiers, permettant d'agir sur une seule table, assouplissant ainsi l'utilisation.
Le nom du fichier est indiqué pour chaque table.

Table PERSONNE → createPersonne.sql

```
drop table PERSONNE;

/*==============================================================*/
/* Table : PERSONNE                                             */
/*==============================================================*/
create table PERSONNE (
LOGIN                VARCHAR(8)          not null,
RESP_LOGIN           VARCHAR(8)          null,
NOM                  VARCHAR(64)         not null,
PRENOM               VARCHAR(64)         null,
MAIL                 VARCHAR(64)         null,
STATUT               VARCHAR(64)         null,
DATE_DEBUT_RES       DATE                null,
DATE_FIN_RES         DATE                null,
ISADMIN              INT2                null,
PASS                 VARCHAR(8)          null,
constraint PK_PERSONNE primary key (LOGIN)
);
```

Table COMPOSANT → createComposant.sql

```
drop table COMPOSANT;

/*==============================================================*/
/* Table : COMPOSANT                                            */
/*==============================================================*/
create table COMPOSANT (
ETIQUETTE            VARCHAR(64)         not null,
USER_LOGIN           VARCHAR(8)          null,
ID_PIECE             INT4                null,
MASTER_ETIQUETTE     VARCHAR(64)         null,
NOM_TYPE             VARCHAR(64)         not null,
RESP_LOGIN           VARCHAR(8)          null,
DATE_ACHAT           DATE                null,
DATE_FIN_GARANTIE    DATE                null,
NUM_CONTRAT_MAINTENANCE VARCHAR(64)         null,
CONSTRUCTEUR         VARCHAR(64)         null,
COMMENTAIRE_COMP     VARCHAR(1024)       null,
PRIX_D_ACHAT         VARCHAR(64)         null,
constraint PK_COMPOSANT primary key (ETIQUETTE)
);
```

Table TYPE_COMP → createTypeComp.sql

```
drop table TYPE_COMP;

/*==============================================================*/
/* Table : TYPE_COMP                                            */
/*==============================================================*/
create table TYPE_COMP (
NOM_TYPE             VARCHAR(64)         not null,
COMMENTAIRE_TYP      VARCHAR(1024)       null,
constraint PK_TYPE_COMP primary key (NOM_TYPE)
);
```

Table CAR_TYPE → createCarType.sql

```
drop table CAR_TYPE;

/*==============================================================*/
/* Table : CAR_TYPE                                             */
/*==============================================================*/
create table CAR_TYPE (
ID_CARTYPE              INT4              not null DEFAULT
NEXTVAL('SEQ_IDCARTYPE'),
NOM_TYPE          VARCHAR(64)       null,
LIBELLE           VARCHAR(64)       null,
COMMENTAIRE_CAR_TYP  VARCHAR(1024)   null,
constraint PK_CAR_TYPE primary key (ID_CARTYPE)
);
```

Table CAR_COMP → createCarComp.sql

```
drop table CAR_COMP;

/*==============================================================*/
/* Table : CAR_COMP                                             */
/*==============================================================*/
create table CAR_COMP (
ID_CARTYPE  INT4              not null,
ETIQUETTE   VARCHAR(64)     not null,
VALEUR      VARCHAR(1024)   not null,
constraint PK_CAR_COMP primary key (ID_CARTYPE, ETIQUETTE)
);
```

Table PIECE → createPiece.sql

```
drop table PIECE;

/*==============================================================*/
/* Table : PIECE                                                */
/*==============================================================*/
create table PIECE (
ID_PIECE             INT4                 not null   DEFAULT
NEXTVAL('SEQ_PIECE'),
NOM_BATIMENT         VARCHAR(64)          not null,
NOM_PIECE            VARCHAR(64)          not null,
NUM_TEL              VARCHAR(64)          null,
constraint PK_PIECE primary key (ID_PIECE)
);
```

Table INTERVENTION → createIntervention.sql

```
drop table INTERVENTION;

/*==============================================================*/
/* Table : INTERVENTION                                         */
/*==============================================================*/
create table INTERVENTION (
ID_INTERV            INT4                 not null   DEFAULT
NEXTVAL('SEQ_INTERV'),
ID_PROB              INT4                 null,
INTITULE_INT         VARCHAR(64)          null,
DATE_INTERV          DATE                 not null,
```

```
DESCRIPTION_INT      VARCHAR(1024)        null,
MOT_CLES             VARCHAR(128)         null,
constraint PK_INTERVENTION primary key (ID_INTERV)
);
```

Table PROBLEME → createProbleme.sql

```
drop table PROBLEME;

/*==============================================================*/
/* Table : PROBLEME                                             */
/*==============================================================*/
create table PROBLEME (
ID_PROB              INT4                 not null DEFAULT NEXTVAL('SEQ_PROB'),
ID_INTERV            INT4                 null,
ADMIN_LOGIN          VARCHAR(8)           null,
INTITULE             VARCHAR(64)          null,
ETIQUETTE            VARCHAR(64)          null,
SIGN_LOGIN           VARCHAR(8)           null,
DATE_PROB            DATE                 not null,
DESC_PROB            VARCHAR(1024)        null,
STATUT               VARCHAR(64)          null,
constraint PK_PROBLEME primary key (ID_PROB)
);
```

SEQUENCES (pour les clés primaires automatiques) → seqGPI.sql

```
create sequence SEQ_IDCARTYPE
increment 1
start 1;

create sequence SEQ_INTERV
increment 1
start 1;

create sequence SEQ_PIECE
increment 1
start 1;

create sequence SEQ_PROB
increment 1
start 1;
```

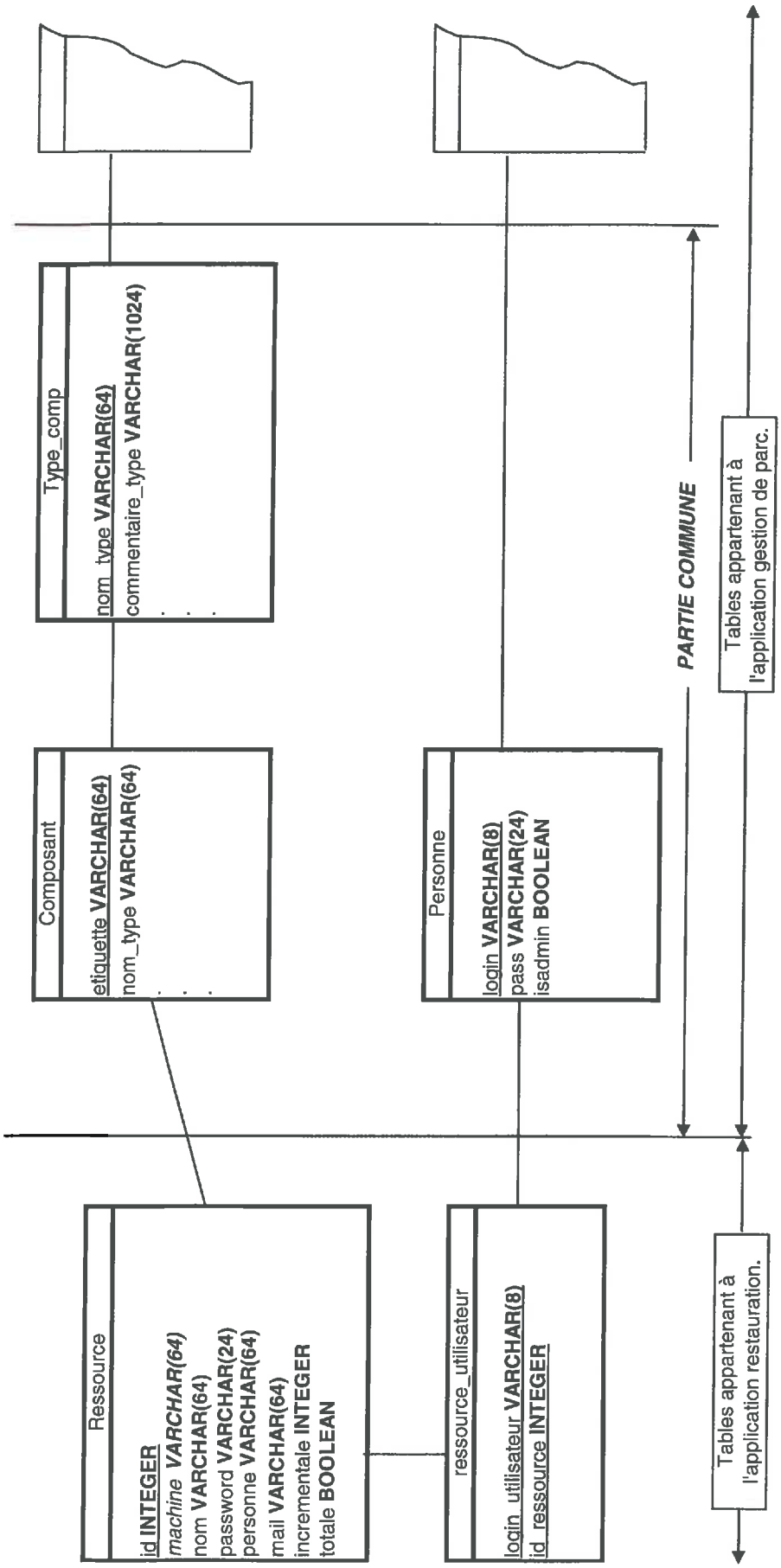
Clés étrangères → foreignGPI.sql

```
/*==============================================================*/
/* Nom de la base :  MPD_GPI                                    */
/* Nom de SGBD :  PostgreSQL 7                                  */
/* Date de cr,ation :  27/05/2003 11:43:18                      */
/*==============================================================*/


alter table CAR_COMP
   add constraint FK_CAR_COMP_CAR_COMP_CAR_TYPE foreign key (ID_CARTYPE)
      references CAR_TYPE (ID_CARTYPE)
      on delete cascade on update cascade;

alter table CAR_COMP
   add constraint FK_CAR_COMP_CAR_COMP2_COMPOSAN foreign key (ETIQUETTE)
      references COMPOSANT (ETIQUETTE)
      on delete cascade on update cascade;
```

```
alter table CAR_TYPE
   add constraint FK_CAR_TYPE_AVOIR_TYPE_COM foreign key (NOM_TYPE)
      references TYPE_COMP (NOM_TYPE)
      on delete cascade on update cascade;

alter table COMPOSANT
   add constraint FK_COMPOSAN_APPARTENI_TYPE_COM foreign key (NOM_TYPE)
      references TYPE_COMP (NOM_TYPE)
      on delete cascade on update cascade;

alter table COMPOSANT
   add constraint FK_COMPOSAN_DEPENDRE_COMPOSAN foreign key (MASTER_ETIQUETTE)
      references COMPOSANT (ETIQUETTE)
      on delete set null on update cascade;

alter table COMPOSANT
   add constraint FK_COMPOSAN_ETRE_RESP_PERSONNE foreign key (RESP_LOGIN)
      references PERSONNE (LOGIN)
      on delete set null on update cascade;

alter table COMPOSANT
   add constraint FK_COMPOSAN_SITUER_PIECE foreign key (ID_PIECE)
      references PIECE (ID_PIECE)
      on delete set null on update cascade;

alter table COMPOSANT
   add constraint FK_COMPOSAN_UTILISER_PERSONNE foreign key (USER_LOGIN)
      references PERSONNE (LOGIN)
      on delete set null on update cascade;

alter table INTERVENTION
   add constraint FK_INTERVEN_CORRIGER_PROBLEME foreign key (ID_PROB)
      references PROBLEME (ID_PROB)
      on delete set null on update cascade;

alter table PERSONNE
   add constraint FK_PERSONNE_ETRE_RESP_PERSONNE foreign key (RESP_LOGIN)
      references PERSONNE (LOGIN)
      on delete set null on update cascade;

alter table PROBLEME
   add constraint FK_PROBLEME_CORRIGER2_INTERVEN foreign key (ID_INTERV)
      references INTERVENTION (ID_INTERV)
      on delete set null on update cascade;

alter table PROBLEME
   add constraint FK_PROBLEME_PRENDRE_E_PERSONNE foreign key (ADMIN_LOGIN)
      references PERSONNE (LOGIN)
      on delete set null on update cascade;

alter table PROBLEME
   add constraint FK_PROBLEME_SIGNALER_PERSONNE foreign key (SIGN_LOGIN)
      references PERSONNE (LOGIN)
      on delete set null on update cascade;

alter table PROBLEME
   add constraint FK_PROBLEME_SUBIR_COMPOSAN foreign key (ETIQUETTE)
      references COMPOSANT (ETIQUETTE)
      on delete set null on update cascade;
```

ANNEXE 8

MPD Commun aux deux applications


Type_comp
nom_type VARCHAR(64)
commentaire_type VARCHAR(1024)
Composant
etiquette VARCHAR(64)
nom_type VARCHAR(64)
Personne
login VARCHAR(8)
pass VARCHAR(24)
isadmin BOOLEAN
PARTIE COMMUNE
Tables appartenant à l'application gestion de parc.
Ressource
id INTEGER
machine VARCHAR(64)
nom VARCHAR(64)
password VARCHAR(24)
personne VARCHAR(64)
mail VARCHAR(64)
incrementale INTEGER
totale BOOLEAN
ressource_utilisateur
login_utilisateur VARCHAR(8)
id_ressource INTEGER
Tables appartenant à l'application restauration.

ANNEXE 9

Plan du document de conception