RECONSTRUCTION DU SITE INTRANET DE GESTION DES SALLES DE REUNION

CNAM 2005-2006

NFE 210
Ingénierie des Systèmes d'Information

Frédéric Petitdemange

SOMMAIRE

SOMAIRE DES ILLUSRATIONS

SOMAIRE DES TABLES

1 Introduction

Le projet que je souhaite partager avec vous est celui qui m'a été offert de faire sur une petite application de type web dont la fonction principale est la gestion des salles de réunion. Cette application est journalièrement utilisée soit par le personnel de la réception pour effectuer les réservations nécessaire soit par l'ensemble des membres de l'entreprise pour visualiser les disponibilités ou l'équipement des salles de réunion. C'est donc une application à la fois simple dans sa définition et importante du point de vue fonctionnel pour une entreprise effectuant beaucoup de réunion avec ces clients, prestataires ou personnels internes.

Pour effectuer notre travail de reconstruction de notre application, nous allons dans un premier temps effectuer une rétro conception de notre site en effectuant un "Design Recovery" afin de remonter au niveau d'abstraction le plus élevé via une approche "Botton-up". Nous effectuerons par la même occasion une "Redocumentation" de tout ou partie de la documentation manquante à la bonne compréhension et à la poursuite de la reconstruction du site. Enfin, nous approcherons la phase de "Restructuring" qui nous permettra d'effectuer une modification de la représentation actuelle des données, de la structure de l'application ou du code en une représentation plus proches du réel ou des bonnes pratiques de développement. Mais aussi du "Refactoring" qui consiste à simplifier, optimiser et documenter un code source. Ceci afin de préparer la phase de "Forward Engineering", phase classique et normale de conception vers l'avant d'une application (opposé aux phases de "Reverse Engineering" dans le sens du développement).

Figure 1 - Concept de la reconstruction

Lors de la reconstruction de cette application, une question va très vite surgir à propos de l'approche choisie, l'approche "Botton-Up" versus "Top-Down". Dans un cadre dénué d'un intérêt CNAMIEN pour l'élaboration de ce projet, j'aurais de très vite changé d'approche pour une approche "Top-Down". J'aurais repris par les fonctionnalités attendu en écrivant le cahier des charges dans une phase normale de conception avancé de l'application et inclus une recherche sur l'étagère d'un logiciel existant. L'approche "Botton-Up" a été gardée pour mettre en avant au travers de ce document les problématiques de reconstruction d'un site web a travers une méthodologie ou manière de faire et en mettant en avant les problématiques de qualité de l'application.

Le concept de la qualité est basé sur le contrôle puis l'amélioration de cette qualité et enfin son management comme présenté par la figure suivante.

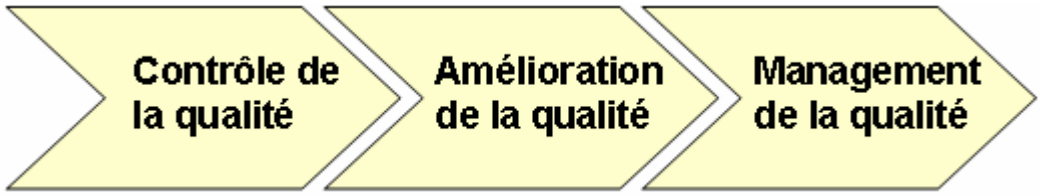

Figure 2 - Concept de la qualité

Ces deux concepts peuvent être fusionné et étoffé pour donner un tableau croisé des différents concepts de la reconstruction et de la qualité. Ce tableau pouvant servir dans ces deux dimensions pour faire d'un concept (Design recovery, refacturing ou d'autre) un élément permettant de faire de la reconstruction et de la qualité.

Reconstruction	Modification	Implémentation	Engénierie
Reconstruction SGBD & DB	Refonte du MCD	Implémentation du nouveau MCD	Ajout de fonctionnalité au MCD
Reconstruction du code source	Refactoring		Ajout de fonctionnalité
Contrôle de la Qualité	Amélioration de la Qualité		Management de la Qualité

Figure 3 - Tableau croisé des concepts de reconstruction et de qualité

C'est cette présentation des données qui sera suivit tout au long de la présentation de la reconstruction de notre application.

Nous allons dans un premier temps présenter l'application dans son contexte, puis présenter la reconstruction effectuée sur les données, son SGBD, la structure fichier de l'application, le code source et le design. En troisième phase, je présenterai la phase de "restructuring" ou phase reconstruction-évolution du modèle, de la structure ou du code. Suivra une notation de la qualité de l'application, une présentation de quelques bonnes pratiques du Web et je finirai par la conclusion.

2 Présentation – Contexte

L'application a été conçu 6 ans plus tôt dans un contexte très fort de webisation à outrance ou le client léger été le mode choisi par tous pour apporter les fonctionnalités demandé par les clients/utilisateurs. A cette époque, le DSI et/ou le responsable des moyens généraux en place ont pour des raisons non connu préféré de travailler avec un développeur étranger (Pays-Bas). Le responsable des moyens généraux n'avait aucune connaissance dans le développement WEB et souhaité fortement travailler avec ce développeur étranger, le DSI qui avait une forte connaissance de ces projets puisque lui-même chef de projet WEB en SSII n'avait par contre que peut de temps à y consacrer. Les besoins ont été succinctement formalisés via email, cet email n'étant plus disponible à ce jour.

Les recherches d'informations sur cette application se sont soldées par:

- plus de développeur – il a totalement disparu de la scène et n'est plus joignable.
- aucune documentation (définition des besoins, cahier des charges, description fonctionnel, etc.…), même obsolète… il n'y a rien, voir même il semble qu'il n'y a jamais rien eu.
- perte de mémoire complet de la gestion de ce projet.

Les seuls éléments tangible restant à notre disposition sont les utilisateurs, le responsable des moyens généraux qui ne se rappelle plus de tout ce qui c'est décidé sur ce projet mais qui souhaiterait le voir évoluer, et bien sure l'application elle-même.

Pourtant, quelques points positifs viennent s'ajouter à ce descriptif.

- Le premier est que l'application fonctionne sur un serveur WEB intégrant un gestionnaire de base de données dont l'application fait usage (gage me semble t'il d'un minimum de sérieux et d'intérêt pour ce projet).
- Le second point positif est que l'application n'a jamais subi de problème technique hormis des problèmes de performance lors de mise en charge du gestionnaire de base de données par des bases issue de nos propres clients pour y faire des d'analyses (gage me semble t'il du sérieux du développement et du modèle conceptuelle de données utilisé).
- Le troisième et non des moindre, est que les utilisateurs sont content de leur application.

2.1 La gestion des salles de réunion

L'application web de gestion des salles de réunion est accessible par deux types de population. Les premiers peuvent

- effectuer les réservations
- modifier ces réservations
- supprimer ces réservations
- gérer le réapprovisionnement des salles

Et les seconds ne font que visualiser l'ensemble des informations de réservation.

2.1.1 Le suivit des salles

Le suivit des salles de réunion consiste pour les membres de la réception de gérer les salles, le besoins de ceux qui en on besoins en fonction des équipements installé (système de vidéoconférence, projecteur, lecteur DVD, système de conférence téléphonique, etc.), et le réapprovisionnement de ces salles en boisson, café et autres services.

Les membres de la réception sont les seules à pouvoir effectuer la création, modification ou annulation de réservation des salles de réunion.

L'interview m'apprendra aussi qu'une réservation peut-être récurrente dans le temps et que l'interface graphique à été volontairement choisi pour une utilisation la plus simple possible. Un intérimaire devant pouvoir effectuer une réservation simplement, rapidement et sans connaissance préalable.

L'application a les interfaces graphiques suivant pour effectuer toute la gestion des salles de réunion.

L'interface principale présente la réservation des salles pour la journée en cours comme le présente la Figure 4 - Etat de la réservation des salles pour la journée en cours.

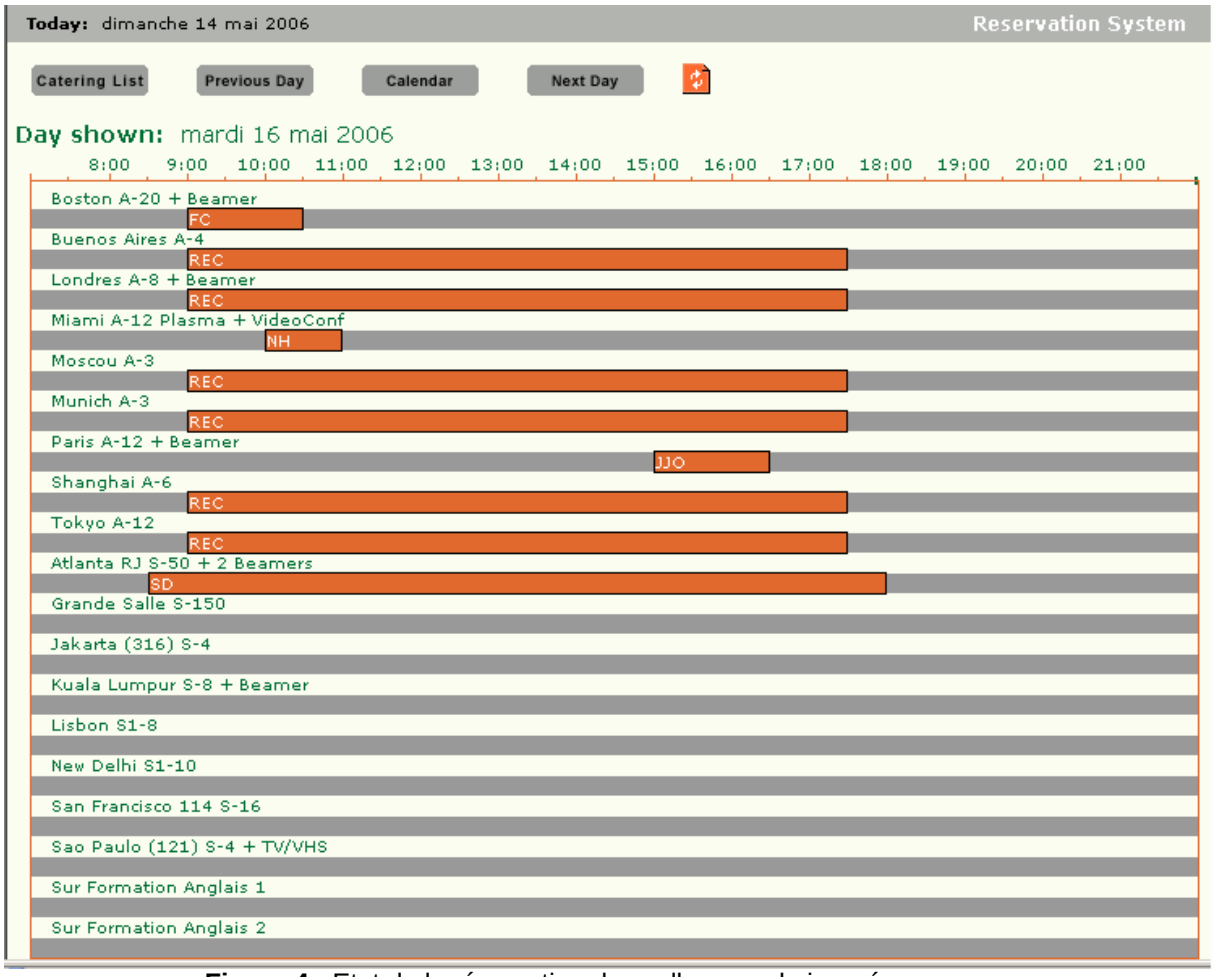

Figure 4 - Etat de la réservation des salles pour la journée en cours.

Le formulaire suivant présente l'interface de réservation d'une salle.

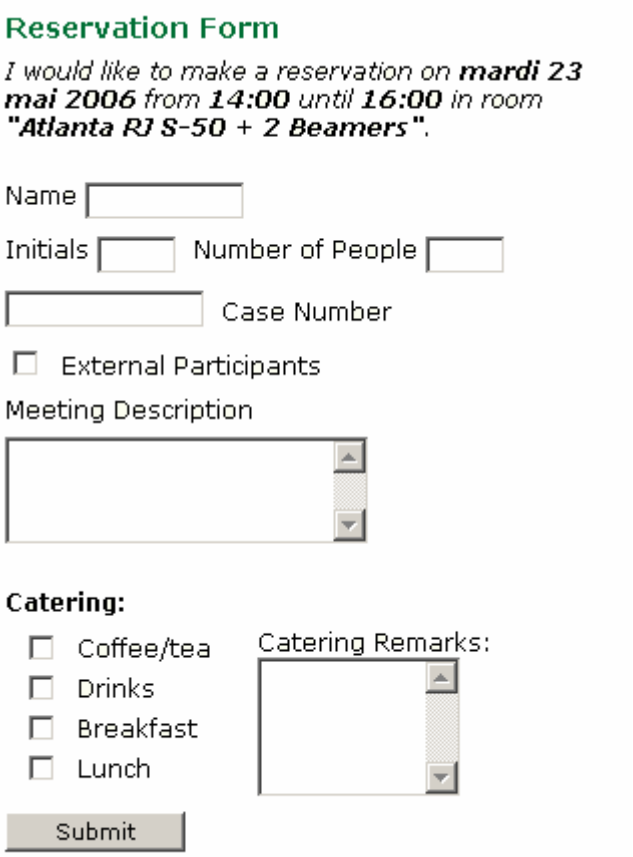

Figure 5 - Formulaire de réservation d'une salle

Les formulaires suivants présents la manière d'effectuer une réservation récurrente. Le premier permet de sélectionner la création ou la suppression d'une réservation récurrente.

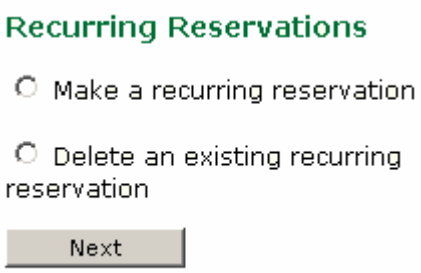

Figure 6 - Formulaire de réservation récurrente - 1

L'étape suivante consiste dans le choix de la salle de réunion et dans celui de la récurrence de cette réunion, soit :

- journalière
- hebdomadaire
- mensuelle

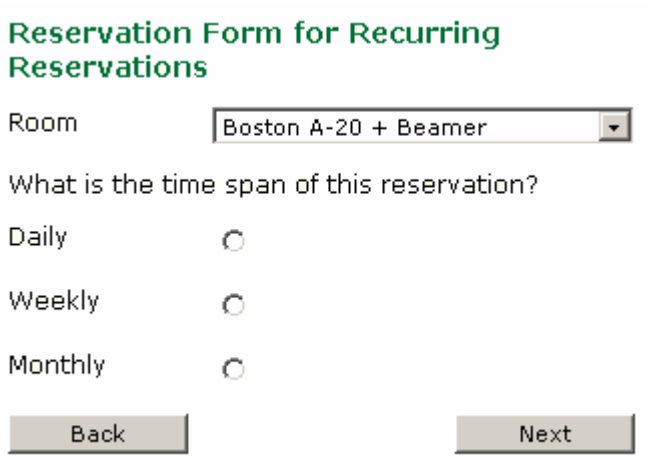

Figure 7 - Formulaire de réservation récurrente - 2

Enfin, la prise de réservation récurrente se termine par la sélection d'information horaire et de date.

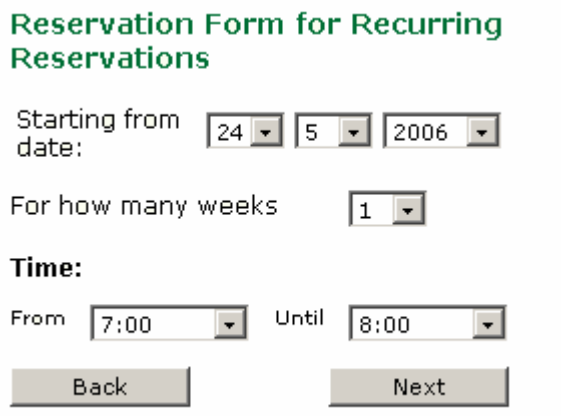

Figure 8 - Formulaire de réservation récurrente - 3

2.1.2 Le suivit des approvisionnements

Le suivit des réapprovisionnements consiste pour une personne de la réception à vérifier que les besoins exprimés par les utilisateurs lors de la réservation sont bien pris en compte lors de la préparation des salles et de délivrer ce service en portant la bouteille d'eau, le café ou les plateaux repas commandés. Pour cela, elle on besoins de vérifier le jour d'avant les besoins pour effectuer les commandes nécessaire, le matin même pour organiser l'installation dans les

salles des commandes et au cours de la journée pour s'assurer d'aucun oublie.

A noter que notre application effectue un suivit des approvisionnements mais en aucun cas la commande de ceux-ci.

Le formulaire suivant présente le formulaire de vérification de l'approvisionnement des salles.

Catering List

mardi 23 mai 2006

Time	Room name	Participants	Coffee	Drinks	Breakfast	Lunch	Comments
7:00 - 10:00	Atlanta RJ S-50 + 2 Beamers	3					
8:00 - 19:30	Moscou A-3	2					
8:00 - 10:00	Shanghai A-6	3					
8:30 - 11:30	Boston A-20 + Beamer	15					
8:30 - 10:00	Buenos Aires A-4	1					
9:00 - 10:30	New Delhi S1-10	3					prendre salle Aguesseau si possible
9:00 - 17:00	Munich A-3	2					
9:00 - 11:00	Paris A-12 + Beamer	5					
9:00 - 11:00	Miami A-12 Plasma + VideoConf	10					
9:00 - 12:00	San Francisco 114 S-16	3					
9:00 - 10:00	Londres A-8 + Beamer	8					9 petits déjs OK

Figure 9 - Formulaire de vérification des approvisionnements

2.2 L'historique de l'application

L'historique de l'application est assez simple à décrire…

Il n'y en a pas ou très peu. La date de mise en service est au alentour de l'année 2000, sans certitude. Personne ne se rappel non plus du nom du développeur. Il semble qu'il n'y a jamais eu de documentation que ce soit sous la forme d'un cahier des charges ou d'une documentation de fonctionnement.

Il n'y a rien si ce n'est l'application elle-même, les utilisateurs de ce système et le commanditaire ou "Maître d'Ouvrage" de l'application.

2.3 Le périmètre de travail

Le périmètre de travail n'incluse pas les serveurs proprement dit (Web ou SQL) ni tout autre élément de l'infrastructure physique de l'entreprise. On ne cherchera pas à modifier le nom de l'application (impact DNS,..) ni la localisation de l'application (impact server web,…). Nous ne travaillons que sur l'application et uniquement sur elle.

2.4 Les nouvelles fonctionnalités demandées

Les nouvelles fonctionnalités demandées sont :

- une gestion des actions effectuées, pour savoir qui à fait quoi
- l'administration (création/suppression) des salles de réunion

Il n'y a pas d'autre nouvelle fonctionnalité puisque l'application donne toute satisfaction et que les utilisateurs de ce système ne s'en plaignent pas.

2.5 Les besoins non fonctionnels

Les besoins non fonctionnels cité par le MOA sont :

- simplicité d'utilisation
- fiable
- rapide

Afin de ne pas rajouter aux stresses quotidien des réceptionnistes utilisateurs principaux de ce système.

2.6 Les standard de l'entreprise

L'entreprise fonctionne dans un grand groupe mondial présent dans une soixantaine de bureau dans le monde soit cinquante pays différents. L'informatique est structurée autour de trois grands pôles infrastructure, application et service. L'ensemble est régi par des standards de fonctionnement qui pour notre exemple se résume au tout Microsoft…

Serveur Web, serveur de base de données, operating système Microsoft, le tout suivit par des systèmes de monitoring afin de mesuré les taux de disponibilités et par une équipe chargé de suivre au quotidien la maintenance hardware et software (patch de sécurité) de ces serveurs.

3 Reconstruction

La reconstruction du système d'information se fera soit de manière séquentiel soit de manière plus itérative lorsque la recherche des informations seront plus complexe.

Pour effectuer notre travail de reconstruction de notre application, nous allons dans un premier temps effectuer une rétro conception de notre site en effectuant un "Design Recovery". Il s'agit de remonter au niveau d'abstraction le plus élevé par une approche "Botton-up". Nous partons des éléments de base qui sont mis en notre possession (base de données, schéma de la base, code source,..) pour reconstruire ou recréer les documents de base permettant une compréhension sans équivoque du fonctionnement de l'application comme le modèle conceptuel de données, la structure de l'application ou le cahier des charges fonctionnels de l'application. Nous effectuerons par la même occasion une "Redocumentation" de tout ou partie de la documentation manquante à la bonne compréhension et à la poursuite de la reconstruction du site. Ce sont entre autre les documents décrivant les éléments précédents à la bonne compréhension de l'application. Enfin, nous approcherons la phase de "Restructuring" qui nous permettra d'effectuer les modifications de la représentation actuelle des données, de la structure de l'application ou du code source de l'application en une représentation plus proches du réel ou des bonnes pratiques de développement. Ceci afin de préparer la phase de "Forward Engineering", phase classique et normale de conception vers l'avant d'une application (opposé aux phases de "Reverse Engineering" dans le sens du développement) afin de prendre en compte les modifications demandées.

Je commencerais par le plus simple et le plus visible, l'architecture du système à travers les éléments clé (serveur Web, serveur de Base de Données,..). Puis la reconstruction de la structure des pages web de l'application. S'en suivra la reconstruction de la base de données, l'analyse du code, la recherche des règles de fonctionnement, des règles utilisées pour le design et de la sécurité. Enfin, je finirais par les besoins fonctionnels et non fonctionnels, les besoins non couvert et par une conclusion sur cette reconstruction. C'est une approche de type "Botton-up" ou l'on part de l'existant pour remonter vers le cahier des charges originel, vers le besoin exprimé ou pas des utilisateurs.

3.1 L'architecture système

3.1.1 Le serveur web

C'est peut-être la partie la plus simple du puzzle. Très vite j'ai identifié le serveur web hébergeant l'application… Je savais que l'application est une application locale, d'où exit les serveurs web des data-centers. Ce point a été confirmé par une simple vérification DNS. Localement, il n'y a qu'un seul serveur, donc cela ne pouvais être que lui. Un rapide coup d'œil sur les applications supportées m'a confirmé sur ce point.

3.1.2 Le Système de Gestion de Base de Données

Ayant trouvé le serveur web hébergeant l'application, un rapide coup d'œil ma permis de voir que ce serveur supportait aussi un moteur de SGBD de type SQL. J'ai alors vérifié dans les pages web source que c'était bien un serveur MS SQL qui était requêté. A ma grande surprise, c'était un moteur Microsoft de type MS Access 7.0. Le fichier "acces_base.inc" chargé au début de plusieurs pages de l'application, pointait vers le fichier "RoomsParis.mdb", ne laissant aucun doute.

Les pages écrites en ASP démontraient le besoins d'avoir des pages web dynamique.

L'architecture de cette application est donc articulée autour d'un serveur web IIS et d'un moteur de BD MS Access.

3.2 L'architecture des pages web

La reconstruction de l'architecture du site passe pas deux étapes. La première consiste à retrouver le chemin complet, au travers de toutes les pages, empruntées par l'application lors de son exécution. Cela permet ainsi de retrouver les pages de tests ou non utilisées. Le résultat est un graphe connexe pouvant avoir plusieurs niveaux d'arborescence dont la particularité est d'avoir de nombreuses pages se connectant à la page principale de l'application. Ce qui présente une certaine circularité de la structure de l'application.

La seconde consiste à retrouver les fonctionnalités principales du site et de reconstruire la carte "map" du site au travers de ces grandes fonctionnalités. Ce sont vers ces pages que l'utilisateur du site sera redirigé en fonction des besoins demandés.

3.2.1 La structure des fichiers du site intranet

Le principe utilisé pour reconstruire la structure de l'application est la mise en place d'une arborescence ou une page pointe vers une autre page.

Le but est d'établir un graphe connexe (ce qui n'est pas toujours le cas) ou chaque pages est un nœud du graphe, chaque liens vers une autre page est un arc du graphe. Chaque arc du graphe sera donc vu comme une nouvelle fonction possible ou une suite logique dans la mise en œuvre de cette fonction. Si l'arbre n'est pas connexe, cela veut dire que des pages ne sont jamais adressées et sont donc inutiles à l'application. Une particularité des structures web et de présenter des arbres connexe circulaire (mais pas toujours) dont les

dernières feuilles de l'arbre pointent vers la "home page" pour revenir au point de départ.

La première page choisi est notre nœud de départ. A partir de ce nœud, on défini toutes les pages adressées par celle-ci créant un arc reliant notre nœud principale au nœud(s) secondaire(s). Chaque page pointé (nœud de l'arborescence) devient une branche qu'il faudra parcourir. On itère ainsi chaque page adressé afin de parcourir toutes les branches de l'arborescence. Une fois cela fait, il faut effectuer cette recherche sur toutes les autres pages non parcouru. Il se peut en effet que l'on n'ait pas commencé par le début ou qu'il y ait plusieurs commencements possibles.

La première page choisit fut celle proposé par le serveur web et indiqué par le serveur DNS. J'ai donc commencé la construction de mon graphe à partir de la page "Home.asp".

L'itération des pages ASP fut rapide puisqu'il n'y avait que vingt-quatre pages.

La figure suivante présente la structure des fichiers se trouvant dans le répertoire racine de l'application.

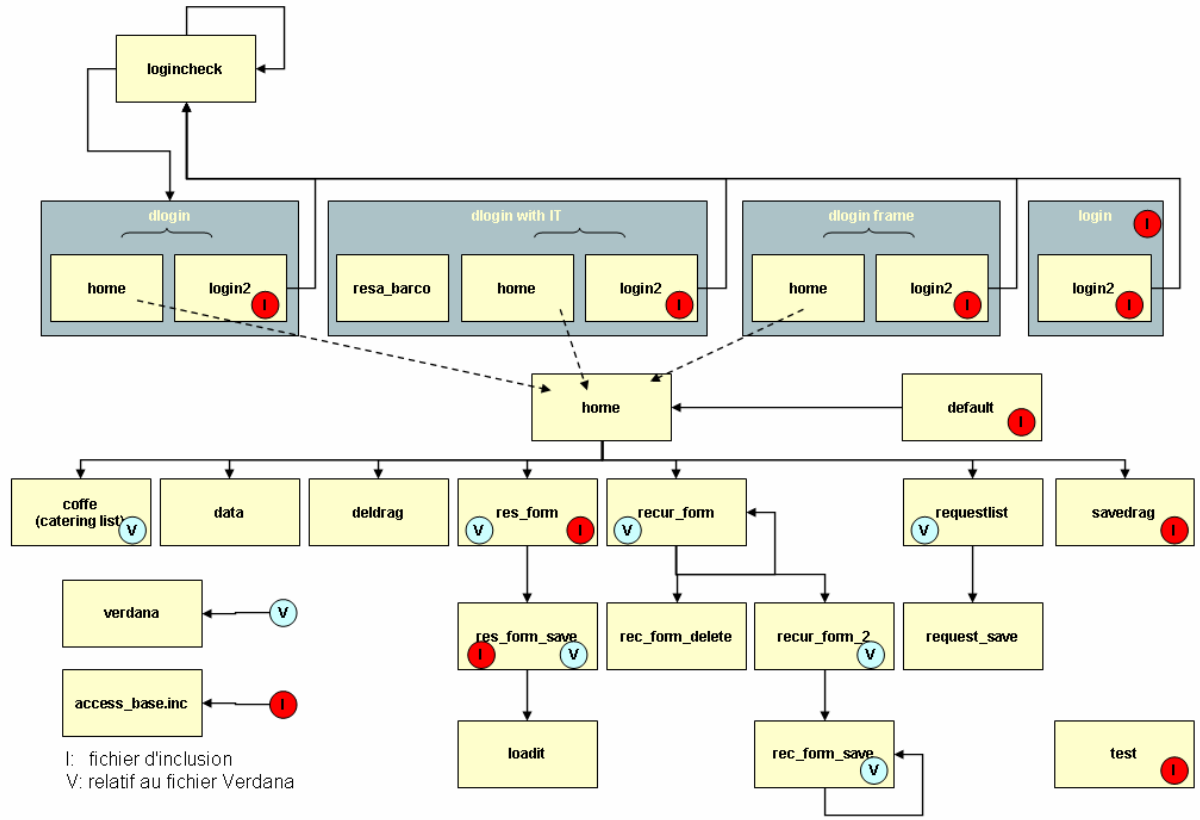

Figure 10 - Structure des fichiers de l'application

Cette première étape présente la structure des fichiers utilisés par l'application.

Premier constat, la présence d'un fichier non connecté au graphe. A lors que le graphe doit être connexe, Je découvre un fichier non connecté. Cela m'indique qu'il ne sert pas à l'application.

Deuxième constat, la présence de quatre possibilités pour ce connecter à l'application. Cela dénote soit une tentative d'ajout de fonctionnalité, soit une tentative de maintenance. Les noms des quatre points de départ me feront penser dans un premier temps à une tentative d'ajout, mais à l'utilisation je découvre un disfonctionnement majeur. S'agissait-il en faite d'une tentative d'ajout et de maintenance non résolu jusqu'à ces jours ?

La tentative de modification est corrélé par le faite que l'une des frames se nomme "Tool" est que justement cette application semble cruellement faire défaut de page d'administration permettant par exemple de gérer une salle de réunion. Nous verrons plus loin la manière dont cela été fait lors de l'analyse de la base de données.

La tentative de maintenance vient du faite que lorsque l'on se connecte à la base, la première page web charge la page ou s'effectue le login proprement dit et que celle-ci transmet l'ensemble des paramètres à une autre page pour vérification. En fonction de quoi cette dite page retour le résultat à la toute première page pour lui signifié une validation ou pas. L'un des arguments transmit outre la validation ou pas du login et le non de la frame… Là est l'erreur de programmation. Un mauvais nom de frame est transmit, l'application ne trouve pas et demande le chargement d'un second browser internet, même s'il y a validation du login… On se retrouve donc dans un fonctionnement normale de l'application avec deux pages web ouverte, l'une resté sur l'interface de login l'autre sur la page principale du site ouvert après login.

A première vue les pages terminales du graphe semblent retourner vers la page principale "home" du site. Nous verront plus loin qu'il n'en était rien en faite, obligeant les utilisateurs à faire un "back" sur le browser internet pour revenir à la "home page" du site. Il n'y a pas circularité de notre graphe connexe ou les pages d'extrémités de notre graphe ne pointent pas vers notre page de départ.

Mon graphe est donc ni connexe ni circulaire…

3.2.2 Le plan du site

A partir de l'analyse précédente et de l'application nous en déduisons facilement le plan du site. Pour effectuer ce plan, il est important de se focaliser sur les grandes fonctions du site intranet. Le ou les services que l'application nous offre pour pouvoir effectuer la gestion

de nos salles de réunion. C'est une analyse des fonctionnalités de notre application.

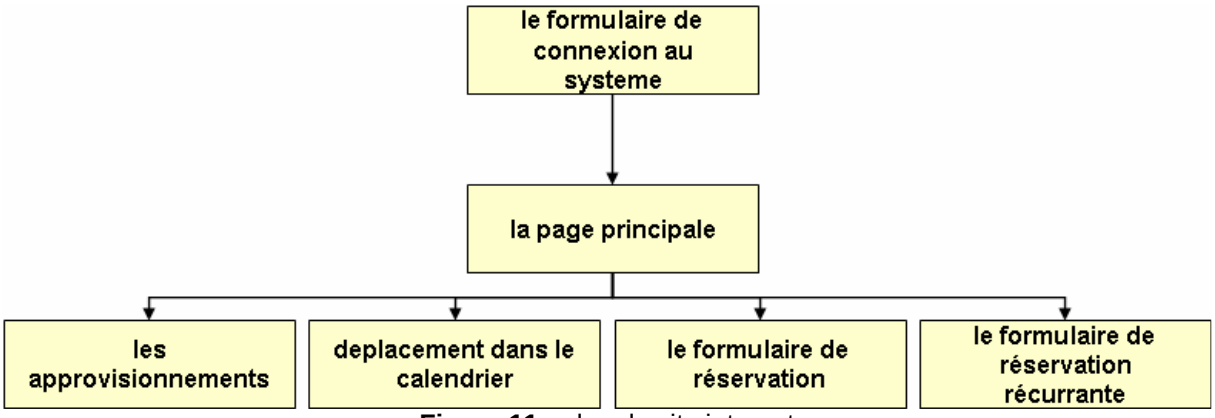

Figure 11 - plan du site intranet

Cela correspond au formulaire de connexion à l'application, la page principale, le rapport des approvisionnements, le déplacement dans le temps et les pages de réservation simple ou récurrente d'une salle.

3.3 La base de données

Un des fondements d'une application dynamique est le couplage fort entre l'application web et la base de données. Je présente ici l'analyse effectuée sur la base de données et son système de gestion afin de pouvoir reconstruire les éléments clé de l'application comme le Modèle Conceptuel de Donnée. L'analyse nous présentera le fichier physique de notre base, le schéma de la base et les tables proposées. Nous aborderons ensuite les aspects de maintenance et de sécurité (backup) de notre base.

3.3.1 Le gestionnaire de la base de données

Nous l'avons découvert précédemment que le moteur de notre base de donnée est un "simple" Access 7.0 de Microsoft. Cela aura au moins l'avantage de me facilité la lecture puisque l'ensemble est graphique.

La lecture du schéma de la base nous propose quatre tables et une requête prédéfinie. Bien sure il n'y a ni trigger, ni fonction, rien d'autres que ces quatre tables et sa requête.

Le fichier MDB fait presque dix-huit méga-octets (18Mo). Il est sauvegardé tout les soirs ainsi que code source de l'application par un serveur de sauvegarde dédié utilisant des bandes magnétiques. Hormis ce backup, il n'y a aucun autre plan de

restauration de prévu, ce qui n'est pas forcément mauvais puisque l'application n'a absolument rien de critique.

3.3.2 La base de données

Détaillons plus en avant les tables afin de découvrir et de reconstruire les modèles logique et conceptuel de données.

Nous avons quatre tables qui sont:

- room
- reservation
- recurring
- administrator

Le tableau suivant récapitule les informations issues soit du système d'exploitation pour définir la taille physique de la base, soit du schéma de la base de données.

Nom de la table	Nombre d'attribut	Volume du schéma de la table (en Ko)	Volume des données (en Ko)	Nombre d'occurrence
administrator	3	116	4	2
room	6	116	4	19
reservation	18	136	4114	30446
recurring	20	120	8	60

Table 1 - Description des informations physiques des tables

Le volume total des données utiles avoisine les 4350 Ko soit 4,35 Mo.
Ce qui dénote franchement avec la taille de la base de données qui fait prés de 18 Mo. Il semble qu'il n'y a jamais eu de maintenance complète sur cette base de données et qu'un nombre assez important de données a été supprimé et qu'aucune défragmentation de la base n'est jamais été faite. On voit ici une des caractéristiques de ce type de base de données que de croitre sans s'optimiser naturellement.

On peut donc aisément supposer que ni une maintenance des données ni une optimisation des tables n'ai jamais été effectuées.

Avançons un peu plus dans la découverte de nos tables et décrivons-les.

La table "Administrator"

administratorid	Autonumber - Long Interger	no duplicate
Administratoruser	Text 50	
Administratorpassword	Text 50	

Table 2 - Description de la table "Administrator"

La table "Recurring"

RecurringId	Autonumber – long Integer	
RecurringName	Text 50	
RecurringType	Number – Long Integer	1 = weekly 2 = monthly
RecurringFreq	Number – Long Integer	1 = every week 2 = every other week 3 ..
RecurringStartTime	Number – Long Integer	
RecurringEndTime	Number – Long Integer	
Recurring_RoomID	Number – Long Integer	
RecurringDuration	Number – Long Integer	
RecurringDayOfWeek	Number – Long Integer	
RecurringWeekOfMonth	Number – Long Integer	
RecurringCoffee	Text 50	
RecurringFris	Text 50	
RecurringBreakfast	Text 50	
RecurringLunch	Text 50	
RecurringText	Text 50	
RecurringAmount	Number – Long Integer	
RecurringMeetingEx	Text 50	
RecurringmeetingRemarks	Memo	
RecurringCaseNum	Text 100	
RecurringEmail	Text 100	

Table 3 - Description de la table "Recurring"

La table "Reservation"

ReservationId	Autonumber – Long Number	
ReservationName	Text 50	
ReservationDate	Date - Time	
ReservationStartTime	Number - single	
ReservationEndTime	Number - single	
Reservation_RoomId	Number - Long integer	
ReservationCoffee	Text 50	
ReservationFris	Text 50	
ReservationLunch	Text 50	
ReservationBreakfast	Text 50	
ReservationText	Text 50	
ReservationAmount	Number - Long integer	
ReservationRecurringId	Number - Long integer	
ReservationEmail	Text 50	
ReservationCheck	Text 50	
ReservationMeetingEx	Text 50	
ReservationmeetingRemarks	memo	
ReservationCaseNum	Text 100	

Table 4 - Description de la table "Reservation"

La table "Rooms"

Roomid	Autonumber – long integer	
Roomnaam	Text 50	Nom de la salle
Roommax	Number - Long integer	
Roomtext	Text 50	
Roomlocation	Text 50	1=Aguesseau 2=Surene
Roomtype	Number - long integer	1=normal room gebouw1 2=VP room gebouw1 3=normal room gebouw2 4=VP room gebouw2

Table 5 - Description de la table "Rooms"

Une lecture approfondie nous permet de découvrir quelques spécificités de ces tables que je vais passer en revue.

La table Administrator:

- Elle n'est pas liée aux autres tables.

C'est un cas classique des applications Web faisant appel à une table pour gérer les comptes de login à une application lorsque l'application n'est pas connecter à un système de gestion des comptes d'entreprise comme un LDAP.

La table Recurring:

- Cette table possède onze attributs que l'on retrouve dans la table réservation.

La table Réservation:

- Cette table possède douze attributs présents dans la table Recurring. (les onze précédents plus l'ID de recurring).

La table Rooms:

- L'attribut "Roomnaam" n'est pas atomique. En effet, l'entretient avec les utilisateurs me donne la clé de cet attribut. Prenons la première assertion de cette table "Boston A-20 + Beamer". Il est constitué du nom de la salle "Boston", puis de la lettre "A" pour la première lettre de l'immeuble où est localisé la salle, en l'occurrence "Aguesseau". Puis du nombre de personne maximum que celle-ci peut contenir "20" et enfin si celle-ci est équipé d'un projecteur vidéo "Beamer". Ce qui nous donne pour cette attribut: "Boston A-20 + Beamer".
- L'attribut "Roomtext" est non utilisé, il ne sert donc à rien.
- L'attribut "Roomlocation" est du type texte de taille 50. Mais en faite ne contient que des "1" ou "2". A la vérification dans le code source, ces "1" et "2" sont traduis par le nom du bâtiment qui les abrite. Ce n'est pas judicieux car cela oblige l'application à traduire ce champs et d'autre part l'entreprise étant très active un autre bâtiment peux être ajouté contenant des salles de réunion. Cela se traduirait par une modification obligatoire du code. En utilisant pleinement ce champ texte en y introduisant le nom du bâtiment directement, on pourrait éviter au code d'effectuer une opération inutile et des problèmes de maintenance du code.

Dans l'ensemble, les types de champs utilisés ne semblent pas toujours opportuns. L'utilisation de champs de type "text" est sur-utilisé comme par exemple pour l'attribut "ReservationBreakfast" qui est du type "text" et qui en faite devrait être booléen. L'utilisation du type Booléen peut concerner jusqu'à dix attributs de la base. Cela est d'autant plus vrai qu'à la vérification dans le code source de l'application, celui-ci traduit systématiquement le texte "yes" par une image "sticker" vert signifiant un oui. De plus, la taille de ces champs ne fait aucun doute sur le faite qu'elle n'a pas été l'objet d'une attention et qu'elle a été choisi large pour palier à d'hypothétique problème ou modification.

Le problème suivant qui retient mon attention est celui concernant ces attributs présent à la fois dans les tables

"Réservation" et "Recurring". Cela ressemble à un héritage entre l'une et l'autre des tables, et c'est le cas. Nous avons ici un héritage car la réservation récurrente est une sorte de réservation simple. A la lecture du contenu des tables, nous retrouvons plusieurs occurrences dont de nombreux attribut sont égaux entre ces deux tables "Recurring" et "Reservation" avec un Id commun qui est "Recurringid" égal à "ReservationRecurringid". La lecture des données nous apprend aussi que des informations issues de la table "Reservation" diffère de la table "Recurring". Nous trouvons l'explication dans le code source et dans l'interview effectué auprès des hôtesses chargés d'utiliser l'application. Il est possible une fois la réservation récurrente effectuée de modifier une occurrence pour ajuster la demande de l'utilisateur en ajoutant ou supprimant la présence d'un service (café par exemple). Une lecture plus approfondie encore du code source vérifié ensuite par des tests sur l'application me révèle qu'une fois faite, la réservation récurrente ne peux plus être modifié. Mieux encore, suite à un défaut de programmation il n'est pas possible non plus de la supprimer. Alors la question se pose, est-il vraiment nécessaire d'avoir tout ces attributs en doublon dans nos deux tables ? C'est une nouvelle fois le code source qui nous donnera la réponse. Les informations redondante entre les deux tables sont temporaire dans la table "Recurring" afin de créer toutes les occurrences nécessaire dans la table "Réservation". L'occurrence de la table "Recurring" ne devenant utilisable plus que pour la supprimer en supprimant toutes les occurrences créer dans la table "Reservation". Ce qui comme je viens de le dire ne sert à rien puisqu'un défaut de programmation empêche de visualiser la réservation récurrente que l'on vient de créer. Les attributs en doublon sont donc inutile et doivent être gérer par l'application lors de la création d'une réservation récurrente. Par contre, les autres attributs de "Recurring" doivent servir à pouvoir sélectionner correctement la réservation récurrente que l'on souhaite supprimer. Ces attributs sont donc utile, et le formulaire de suppression doit être réécris en conséquence.

L'indexation des identifiants des tables existe mais MS Access ne permet pas aisément de retrouver leur lieu de stockage, leur taille ou toutes autres informations.

Enfin, un autre problème survient qui me gênera tout au long de la reconstruction… Je ne parle pas le néerlandais. C'est donc un problème de taille car l'utilisation de plusieurs langues dans la base et dans le code source crée une difficulté de compréhension.

On le voit donc clairement, la base de données ne semble pas avoir fait l'objet d'un travail sérieux de mise au point. Les tables

ne sont pas normalisées, le type des attributs n'a pas été choisi consciencieusement et les tailles des types d'attribut non plus. Enfin, nous avons des données qui peuvent être en doublon et qui seront de toute manière inutile immédiatement après la création des occurrences nécessaire dans la table "Reservation".

Pourtant il y a ici une optimisation de la base qui mérite quand même d'être signalée. L'aplatissement des attributs de service, ceux en doublon entre les tables "Reservation" et "Recurring". Cette aplatissement permet un gain et optimise ainsi la base et le traitement par l'application. Cela a-t-il été volontaire ? J'en doute vu le peut d'attention apporté à cette base mais le souligne ici.

Les tableaux suivant présentent pour l'exemple quelques occurrences des tables sauf pour la table "Administrator"…

Exemple Reservation:

Reservation Id	Reservation Name	Reservation Date	Reservation StartTime	Reservation EndTime	Reservation _RoomId	Reservation Coffee	Reservation Fris	Reservation Lunch	Reservation Breakfast	Reservation Text	Reservation Amount	Reservation Recurring Id	Reservation Email	Reservation Check	Reservation Meeting Ex	Reservationmeeting Remarks	Reservation CaseNum
2497	cf	29/05/2001	7,75	15	10	yes		yes	yes	20 pts dej Pause à 10h45 cafe the biscuits	20	0	Candelon francois				21123/08
2570	AP	12/06/2001	8	20	10	yes		yes	yes	20 petits dej Lina's pour 20 pers	20	0	Angela Paul			Reunion avec des personnes de BCG Melbourne	15204/51
2571	AP	13/06/2001	8	20	10	yes		yes	yes	20 petits dej à 10h30 Lina's pour 20 pers	20	0	Angela Paul			Reunion aves des personnes de BCG Melbourne	15204/51
24239	NB	07/09/2004	8	17,5	22	yes		yes	yes	14 petit dej+17 plateaux mixte+2 thermos à 13h OK	20	0	BONNEFOI		Yes		21038/33

Exemple Room:

RoomID	RoomNaam	RoomMax	RoomText	RoomLocation	RoomTYpe
1	Boston A-20 + Beamer	25		1	1
2	Buenos Aires A-4	4		1	1
4	Londres A-6 + Beamer	8		1	1
5	Moscou A-3	3		1	1
6	Munich A-3	3		1	1
7	Shanghai A-4	5		1	1
8	Miami A-12 Plasma + VideoConf	12		1	1
9	Kuala Lumpur S-8 + Beamer	8		2	1
10	Grande Salle S-150	150		2	1
11	Paris A-12 + Beamer	12		1	1
13	Jakarta (316) S-4	4		2	1
15	Tokyo A-12	12		1	1
22	Atlanta RJ S-50 + 2 Beamers	50		2	1
23	Lisbon RJ S-14 + Beamer	14		2	1
24	Oslo RJ S-18 + Beamer	18		2	1
25	Sao Paulo (121) S-4 + TV/VHS	4		2	1
27	San Francisco 114 S-16	16		2	1
28	Sur Formation Anglais 1	1		2	1
29	Sur Formation Anglais 2	1		2	1

Exemple Recurring:

Recurring Name	Recurring Id	Recurring Type	Recurring Freq	Recurring StartTime	Recurring EndTime	Recurring _RoomID	Recurring Duration	Recurring DayOf Week	Recurring WeekOf Month	Recurring Coffee	Recurring Fris	Recurring Breakfast	Recurring Lunch	Recurring Text	Recurring Amount	Recurring MeetingEx	Recurring Meeting Remarks	Recurring CaseNum	RecurringEmail
AG	128	1	1	12	14	9	4	6	0				yes		10			21625/08	Gourevitch
rec	129	1	1	12	19	2	52	2	0	yes					2	Yes			RECRUITING
REC	130	1	1	12	19	2	52	4	0						2	Yes			REC
REC	131	1	1	12	19	4	52	4	0						2	Yes			REC
REC	132	1	1	12	19	5	52	4	0						2	Yes			REC
REC	133	1	1	12	19	6	52	4	0						2	Yes			REC
REC	134	1	1	12	19	7	52	4	0						2	Yes			REC
REC	136	1	1	12	19	4	52	2	0						2	Yes			REC
REC	137	1	1	12	19	5	52	2	0						2	Yes			REC
REC	138	1	1	12	19	6	52	2	0						2	Yes			REC
REC	139	1	1	12	19	7	52	2	0						2	Yes			REC
REC	140	1	1	12	19	2	52	6	0						2	Yes			REC
REC	142	1	1	12	19	4	52	6	0						2	Yes			REC
REC	143	1	1	12	19	5	1	6	0						2	Yes			REC
REC	144	1	1	12	19	7	52	6	0						2	Yes			REC
REC	145	1	1	12	19	6	52	6	0						2	Yes			REC
REC	146	1	1	12	19	5	52	6	0						2	Yes			REC
EP	148	0	0	12	14	4	2	0	0				yes		6		formation roll out windows 2000		PENVEN Edwige
EP	150	0	0	12	14	4	1	0	0				yes	Lina's (voir e Riva)	6		Formation roll out windows 2000	ADM/PAR-00	PENVEN Edwige
CC	152	0	0	8	18	1	1	0	0						11		FINANCIAL	22118/00	CALONNE CLEMENTINE
DP	153	0	0	7	22	18	1	0	0						4		GILEAD		Picard david
EG	154	0	0	8	22	12	1	0	0						6		interne		Guillet edouard
CH	155	0	0	10	12	6	2	0	0						2		cours d'anglais		HALLEZ charlotte
PL	156	0	0	14	15	12	1	0	0						2				POMMIER Laurent
JGP	158	0	0	14	15	8	1	0	0						2				PELADAN
CG	159	0	0	10	11	7	1	0	0						3				GARNIER
DP	160	0	0	7	22	19	4	0	0						5		PO Gervais Constantia petrou W Zein C Landié		PICARD david
CL	161	1	1	11	14	10	14	2	0						30		MMM		MMM

L'analyse des tables de la base de données "Gestion des salles de réunion" permet d'en déduire le modèle logique de données suivant. Il est le reflet des tables en production dans notre base.

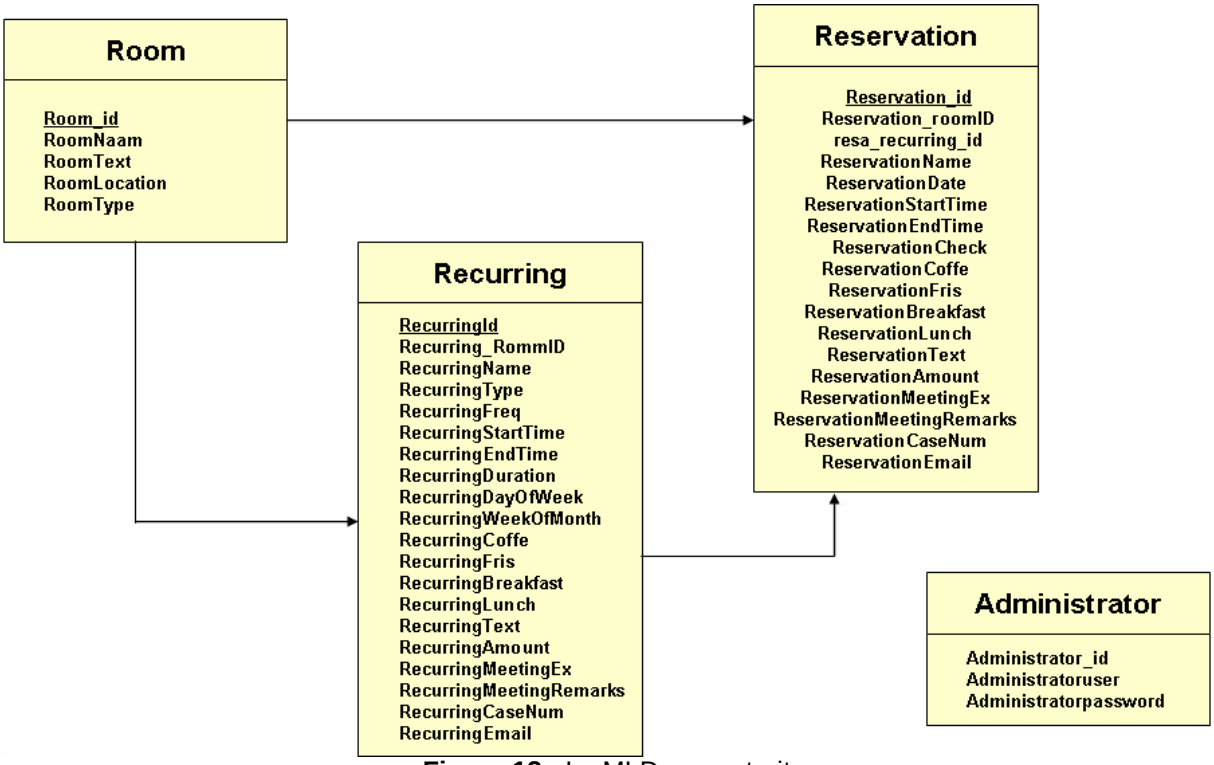

Figure 12 - Le MLD reconstruit

Issue du modèle logique de données nous effectuons une évolution de ce schéma vers le modèle conceptuel de données. Ce travail nous permettra par la suite de comprendre le fonctionnement de l'application et de pouvoir y porter les modifications attendues et demandées. Mais aussi les modifications nécessaires afin de rendre la base de données et le table normalisées avec un fonctionnement optimal. Ce qui contrairement aux interviews n'est pas le cas…

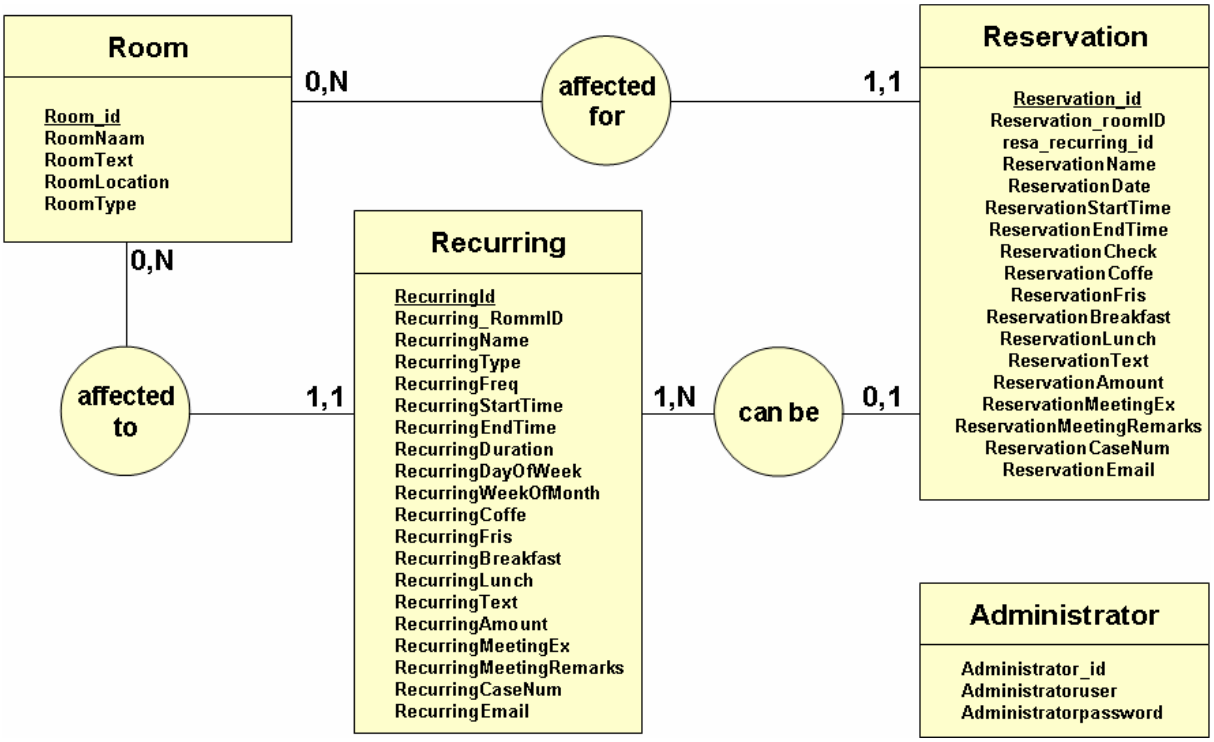

Figure 13 - Le MCD reconstruit

Les informations recueillis ne permettent pas d'entrevoir une séparation entre les données fixe et dynamique. On le voit, la base gère les données dynamiques, mais rien ne gère les données fixes. Et la lecture du code source me confirme que rien n'a été prévu.

3.3.3 Analyse des requêtes SQL

L'analyse des requêtes faites sur les tables de la base montre que la table "administrator" n'est requêtée que pour vérifier l'accès à l'application via le login afin de donner des accès de lecture-écriture à l'utilisateur. Aucune autre requête n'est effectuée sur cette table. Les autres requêtes sont pour la plus part du temps des jointures entre les tables.

L'application gère l'intégrité entre les tables lorsqu'une suppression est effectuée et qu'une réplication de cette suppression est nécessaire. L'application effectue donc une partie du travail qui devrait revenir à la base de données.

Plusieurs requêtes ramène beaucoup plus d'information que nécessaire (exemple avec les pages "Coffee.asp" et "rec_form_delete.asp"). Le résultat se solde soit par une impossibilité d'effectuer l'opération dans le cas de la suppression d'une réservation récurrente ou l'affichage d'information inutile dans le cas de la gestion des approvisionnements.

Il n'y a pas de requête d'insert ou d'update proprement dite mais le développeur utilise les fonctions internes liés à l'ASP pour effectuer des updates, cela alourdit considérablement le code source.

Voici pour exemple la manière utilisé par le développeur pour effectuer un "Update" d'une table.

```
sql2 ="select * From Reservations where ReservationId=0"
rstSave.open sql2,Session("LinkToDatabase"),1,3
rstSave.AddNew
rstSave.fields("ReservationName") = request("RecurringName")
rstSave.fields("ReservationDate") = res(x)
rstSave.fields("ReservationStartTime") = request("RecurringStartTime")
rstSave.fields("ReservationEndTime") = request("RecurringEndTime")
rstSave.fields("ReservationCoffee") = request("RecurringCoffee")
rstSave.fields("ReservationFris") = request("RecurringFris")
rstSave.fields("ReservationBreakfast") = request("RecurringBreakfast")
rstSave.fields("ReservationLunch") = request("RecurringLunch")
rstSave.fields("ReservationmeetingRemarks") = request("RecurringmeetingRemarks")
rstSave.fields("ReservationMeetingEx") = request("RecurringMeetingEx")
rstSave.fields("ReservationCaseNum") = request("RecurringCaseNum")
rstSave.fields("ReservationEmail") = request("RecurringEmail")
rstSave.fields("Reservation_roomid") = request("Recurring_roomid")
rstSave.fields("ReservationAmount") = request("RecurringAmount")
rstSave.fields("ReservationText") = request("RecurringText")
rstSave.fields("ReservationRecurringId") = RecurringId
rstSave.Update
rstSave.close
msg = "The reservation is made.<br>"
```

L'utilisation de ce code pour effectuer des modifications dans la base de données laisse à penser que le développeur à peu ou pas du tout de connaissance en SQL.

Ce qui pourrait expliquer le MCD de la base, des requêtes toujours du même niveau de complexité et pas toujours juste. Il n'est plus possible par exemple de supprimer une réservation récurrente, soit par ce que l'interface ne le permet pas, soit par ce que la requête prend en compte toutes les réservations récurrente même passé au lieu des futurs ou en cours…

D'autre part, des requêtes sont dupliquées ou triplées dans certaines pages du code source. Cela traduit un manque de fonction dédié ou le manque de programmation orienté objet et une grande pratique du "copier-coller".

Il n'y a aucune requête pour gérer les salles de réunion. Celles-ci ont donc été crée manuellement directement sur la base de données. De la même manière pour la suppression avec l'inconvénient déjà analysé que l'intégrité des tables n'est pas respecté puisque l'on retrouve dans la table réservation des occurrences lié à des salles de réunion qui n'existe plus.

Enfin, la charge de travail provoquée par ces requêtes n'a pas été mise en évidence, le serveur ayant des ressources suffisantes pour effectuer ce travail dans un temps tout à fait acceptable. Mais, même si la charge du serveur n'a pas été mise en évidence, la conception des requêtes nous permet une optimisation facile afin d'obtenir de substantiel gain de performance et un meilleur fonctionnement de certains modules développés.

3.3.4 Plan de maintenance

Nous l'avons vu plus haut, dans la section 3.3 Base de données que le plan de maintenance ne concerne que le backup par un agent dédier à cet effet sur les fichiers de la base et du code source. MS Access ne permettant, en automatique, rien d'autre. Il n'y a, ni ne semble avoir eu, d'optimisation ni de défragmentation des tables.

3.4 Analyse du code Source

L'analyse du code source nous montre l'utilisation de quatre langages différents pour gérer cette application :

- HTML
- ASP
- JAVASCRIPT
- VBSCRIPT

Il arrive même à presque toutes les pages que ces quatre langages soient utilisés en même temps. Cela alourdit considérablement la lecture et la compréhension du code source et permet à une erreur de ne pas être découvert rapidement. L'utilisation de tantôt l'un tantôt l'autre des langages coté client me laisse à penser que le développeur ne maitrise en faite réellement ni l'un ni l'autre.

La volonté d'utiliser plusieurs langages pour interpréter les commentaires, tantôt en Anglais tantôt en Néerlandais me laisse aussi la sensation que le développeur souhaite rester propriétaire de son code en y gardant une certaine opacité de lecture.

Nous voyons aussi que certaines lignes de code son purement et simplement recopiées dans l'état alors que l'on aurait pu effectuer une fonction ou une routine.

Le développeur a des lacunes dans la connaissance des langages qu'il utilise. En voici un exemple dans la page "loginCheck.asp". Cette page a pour but d'effectuer une requête SQL en étant sure que l'utilisateur ne pourra pas la reloader. On y arrive avec des arguments

et on est automatiquement renvoyé après notre requête SQL vers la home page. Seulement pour effectuer ce renvoi, il utilise les services d'un formulaire dont tous les champs sont cachés.

De la même manière, le programmeur utilise la fonction "javascript.history.go(-1) lorsque l'on click sur un bouton "Back" dans le formulaire de saisie d'une réservation récurrente ("rec_form.asp") afin de revenir en arrière. Comment fonctionne ce formulaire ? En faite, il est composé de plusieurs sous-formulaire comme présenté dans les Figure 6 - **Formulaire de réservation récurrente - 1**, Figure 7 - **Formulaire de réservation récurrente - 2** et Figure 8 - **Formulaire de réservation récurrente - 3**. Si on va jusqu'au bout du formulaire puis que l'on revient en arrière nous obtenons des effets de bord intéressant puisque cliquer deux fois sur le bouton "Back" avec cette fonction reviens à revenir au formulaire que l'on vient de quitter.

Il n'y a pas non plus de fichier de paramétrage permettant par exemple le paramétrage de l'accès à la base de données, ni la mise en variable de paramétrage de tout ce qui peut être modifié lors d'une migration. Il n'a nullement été pris en compte une gestion des données susceptibles d'être un jour modifié. On ne peut facilement ajouter un service sans devoir réécrire certaine page de l'application.

Les variables utilisées le sont généralement en Néerlandais, ne facilitant nullement la lecture et la compréhension, tout comme les très rares commentaires…

Certaines contraintes de fonctionnement listé dans les règles de construction pèsent lourdement sur le code. C'est le cas de la gestion des minutes qui si elle n'avait pas été traduite aurait permis un gain évident tant dans le code que dans la compréhension générale.

Enfin, nulle volonté de segmenté les langages de l'application afin par exemple de créer des routines d'écriture HTML à partir de routine ou fonction en ASP. Cela aurait gagné en lisibilité et simplicité.

Il n'y a pas non plus de véritable fichier d'inclusion avec une arborescence d'inclusion pour l'accès à la base, charger les fonctions nécessaire au fonctionnement ou les fonctions nécessaire à l'écriture HTML des pages.

Cela conduit à un allongement excessif de certaines pages. La page "Home.asp" faisant prés de trente et une pages papier. Ce qui alourdit la compréhension général.

Il n'y a pas de vraie méthodologie de développement et hormis l'enchaînement des pages web, rien ne semble vraiment structuré. Aucune notion de programmation objet, même symbolique, ni de structuration dans l'utilisation des langages utilisés.

3.5 Reconstruction des règles de fonctionnement

A partir du code source on peut reconstituer certaines règles de fonctionnement utilisé pour le développement de l'application.

1. Une réservation ne peut être prise pour une durée inférieure à 15 minutes.
2. La durée d'une réservation s'effectue par incrément de 15 minutes.
3. Les minutes d'une réservation sont systématiquement traduis depuis et vers la base, en décimale. Un quart d'heure valant "25", une demi heure valant "50", trois quart d'heure valant "75" et une heure plein valant "00".
4. Une réservation peut être effectuée pour toute heure du jour et de la nuit sans aucune contrainte.
5. Une réservation récurrente est systématiquement crée dans la table "Recurring" avant que l'on crée les occurrences correspondantes dans la table "Reservation".
6. Possibilité de modifier un service pour n'importe quelle réservation même récurrente.
7. Impossibilité de modifier de manière récurrente un service demandé pour une réservation récurrente.
8. Proposer des requêtes SQL les plus simple possible pour une remonté d'information la plus importante possible.

Cette liste n'est pas exhaustive mais difficile de faire plus dans la jungle du code source. On imagine pourtant assez bien que toutes ces règles ne sont pas édictées par des besoins utilisateurs.

3.6 Analyse du design utilisé

A l'évidence, le programmeur a utilisé plusieurs moyens pour effectuer le design du site.

1. Il a tenté l'utilisation d'un CSS via un fichier ASP chargé régulièrement dans certaines pages ASP, pas dans toutes.
2. Il a régulièrement introduit directement dans le code HTML des balises qui aurait du être mis dans un fichier CSS.
3. Enfin, pour certaines présentations, il fait appel à un atelier de type "Frontpage" pour obtenir les designs recherché.

On le voit, la gestion graphique de l'application c'est effectué au plus facile. Cela aurait mérité d'être optimisé par un CSS.

Le fichier "Verdana.asp" semble être d'une autre utilisation et pourrait provenir d'une autre application. En effet, on y retrouve une description des utilisations possible de la police Verdana dans différentes couleur dans toutes les tailles de cette police.

3.7 Analyse de la sécurité

L'analyse du code nous révèle aussi que l'accès en écriture est validé de la manière suivante:

<frame name="main" src="home.asp?LogIn=ok" scrolling="YES">

Malgré la nécessité d'un login, on ne peut pas dire que la sécurité soit optimale. Ce qui est corroboré par le faite que l'accès au fichier MDB ne soit pas protégé et que les requêtes soient effectuées via le compte "SA" administrateur sans mot de passe.

La base n'est pas sécurisée, mais pour autant, cette application ne semble ne rien avoir de critique. Si ce n'est que la venue d'un client et donc la réservation d'une salle et considéré comme très importante voir prioritaire sur d'autre réunion. Ce point intéressant me rappelle les raisons donnée pour l'amélioration de l'application, celle de pouvoir traquer ce qui se passe, qui crée ou supprime une réservation. C'est surement là la raison de cette demande.

3.8 Les besoins fonctionnels initiaux

Pas de grande surprise par rapport à notre carte de l'application. On peut résumer les besoins fonctionnelles par :

- un login à l'application
- la réservation simple d'une salle de réunion
- la réservation récurrente d'une salle de réunion
- la commande au travers d'une réservation de service annexe (café, boisson, ...)
- la gestion des approvisionnements des salles en fonction des dites commandes
- la possibilité de supprimer une réservation récurrente
- le déplacement dans le temps pour visualiser l'agenda des salles
- la visualisation de l'agenda d'une salle à un jour donnée

3.9 Les besoins non fonctionnels initiaux

Malgré mes recherches, il ne m'a semblé qu'un seul besoin avait été assouvi. Mais ce besoin semble plus un souhait du développeur qu'une demande du MOA ou des utilisateurs. Ce souhait réussi est de rendre la compréhension du code le plus complexe possible et donc indispensable la présence et la relation avec le développeur de cette l'application…

J'ose espérer que je me trompe…

3.10 Les besoins initiaux non couvert

A prime abord, Il semble qu'un certain nombre de pages liées à l'administration du site ont été oubliées dans la livraison de l'application. Il n'y a pas de page d'administration du site pour gérer les salles de réunion, ni gérer l'historisation (annuelle) des réservations. On peut même supposer qu'une analyse réelle des besoins des utilisateurs du site (la réception) aurait conduit à des modifications ou la création d'interface spécifique pour la gestion des réapprovisionnements par exemple.

Du point de vue des besoins non fonctionnels, on ne peut pas dire que l'application permette la gestion des salles de réunion avec sérénité. De nombreux bug vient perturber le bon fonctionnement de l'application, on n'est pas toujours sure de ce que l'on fait et on ne sait pas si une erreur humaine puisse être prévisible et détecté suffisamment tôt. Ce qui je pense à conduit à un nouveau besoin initial, celui de sauvegarder dans un log système ou archivage toutes les actions qui sont effectués. Il n'est pas certain non plus que le moteur MS Access permette des traitements SQL rapide.

Au vue de la gestion de ce projet tant du coté MOA que MOE il est clair qu'aucun concept de besoin non fonctionnelle n'est été pris en compte.

3.11 Analyse de la reconstruction

La reconstruction que nous venons d'effectuer nous a permis de reconstruire le puzzle de notre application au travers des fichiers ASP de l'application, du SGDB et de la base. Il n'est pas toujours simple de comprendre après coup et sans autres informations les motivations des décisions qui ont conduit à ce résultat. Pourquoi une base Access ? Est-ce par méconnaissance du moteur MS SQL ? Est-ce par ce que le développeur n'avait pas de licence MS SQL ? Est-ce pour des raisons de maintenance de la base… Par contre, et on vient de le voir, notre esprit critique nous permet d'envisager l'architecture complète de l'application sous un jour différent.

Les manquements important dans la conception, la réalisation et même dans la livraison (puisqu'il manque au moins un module d'administration) nous démontre que non seulement le MOA ne connaissait rien au développement web, qu'il n'entendait rien non plus à la gestion d'un projet de cette nature et qu'il n'était encore moins l'utilisateur final car sans quoi, il se serait rendu compte de quelque chose… Tout compte fait, ce n'est peut-être pas un MOA…

4 Qualité de l'application

Il est intéressant dans un projet de reconstruction de confronté le résultat obtenu à des critères de qualité. Cela même que nous nous serions appliqué lors de la conception comme but à obtenir. A la différence que là, ce n'est plus l'objectif que l'on se fixe mais le résultat obtenu que l'on évalue.

Pour la notation, j'utiliserai le modèle suivant.

- 1 correspond à la meilleure note possible
- et 5 à la plus mauvaise.

Il s'agit de note subjective basé sur le ressenti plutôt que qualitative par rapport à un modèle mathématique par exemple car l'application est petite et le modèle conceptuelle est simple. Il n'est donc pas nécessaire de se reporté à une modélisation plus complexe de la qualité que celle du ressenti.

Je noterai dans un premier temps la qualité des données puis celle du Système d'Information à travers sa base, le code source et la structure fichier de l'application.

Les critères de qualité des données sont présentés ici:

- Fiabilité confiance accordée aux données
- Fraîcheur comparaison date saisie et date du jour
- Complétude taux de valeurs non manquantes
- Exactitude taux de valeurs correctes
- Validité par rapport à un format, type, etc…
- Crédibilité vraisemblance
- Actualité taux de valeurs non obsolètes
- Disponibilité facilité d'accès
- Cohérence par rapport à un ensemble de contraintes

Fiabilité	2
Fraîcheur	3
Complétude	4
Exactitude	2
Validité	4
Crédibilité	4
Actualité	3
Disponibilité	3
Cohérence	4

Table 6 - Qualité des données

Puis, c'est sur l'application que je porterai mon jugement en notant la qualité de la base de données puis le code source de l'application. Pour cela, j'utiliserai les critères réservé à un Système d'Information, la base pouvant être considéré comme un sous système d'information.

Les dimensions de la qualité d'un SI sont les suivantes:

- Facteurs de fonctionnement
  - l'exactitude, la fiabilité, l'efficience, l'intégrité et l'utilisabilité
- Facteurs d'évolution
  - la maintenabilité, la flexibilité et la testabilité
- Facteurs d'adaptation
  - la portabilité, la réutilisabilité et l'interopérabilité
- Deutsch & Willis
  - la sécurité, la manageabilité, et la survivabilité

Les facteurs de fonctionnement de la base de données.

l'exactitude	3
la fiabilité	5
l'efficience	4
l'intégrité	5
l'utilisabilité	4

Table 7 - Les facteurs de fonctionnement de la base de données

Les facteurs d'évolution de la base de données.

la maintenabilité	4
la flexibilité	4
la testabilité	4

Table 8 - Les facteurs d'évolution de la base de données

Les facteurs d'adaptation de la base de données.

la portabilité	5
la réutilisabilité	5
l'interopérabilité	5

Table 9 - Les facteurs d'adaptation de la base de données

Les facteurs de Deutsch & Willis appliqué à la base de données.

la sécurité	5
la manageabilité	4
la survivabilité	5

Table 10 - Les facteurs de Deutsch & Willis appliqué à la base de données

J'ai souhaité aussi noter les éléments suivant de la base de données

Pertinence des requêtes	3
simplicité du schéma	2
exactitude des types de données	4
complétude des besoins utilisateurs	3
documentation de la base	5
maintenabilité de la base	4

Table 11 - Qualité de la base

Ces dimensions viennent bien corréler l'appréciations faite jusqu'ici.
Puis je note la qualité du code source en appliquant les mêmes critères, ceux du Système d'Information. On supposera comme pour la base de données que le code source représente un sous système d'information du SI applicatif.

Les facteurs de Deutsch & Willis appliqué à la base de données.

l'exactitude	4
la fiabilité	5
l'efficience	4
l'intégrité	5
l'utilisabilité	4

Table 12 - Les facteurs de Deutsch & Willis appliqué à la base de données

Les facteurs d'évolution du code source.

la maintenabilité	4
la flexibilité	4
la testabilité	4

Table 13 - Les facteurs d'évolution du code source

Les facteurs d'adaptation du code source.

la portabilité	5
la réutilisabilité	5
l'interopérabilité	5

Table 14 - Les facteurs d'adaptation du code source

Les facteurs de Deutsch & Willis appliqué au code source.

la sécurité	5
la manageabilité	4
la survivabilité	5

Table 15 - Qualité du code source

Pour la structure de l'application, il est plus complexe de noter la qualité de celle-ci car je ne peux pas certifier que cette application n'ai pas été modifiée par une tierce personne depuis sa livraison. Je me bornerai à qualifier les éléments suivant.

utilisation des concepts de gestion graphique standardisée	4
conformité à une arborescence connexe	2
conformité à une arborescence circulaire	5

Table 16 - Qualité de la structure fichier

Enfin, j'effectuerai une qualification globale de l'application telle qu'on peut la ressentir à la lecture des éléments précédents.

La base	4
Le source	4
La structure	4
avis utilisateur	1

Table 17 - Qualité de l'application

J'ai choisi de prendre en compte l'avis utilisateur qui va à l'encontre de mon ressenti de la qualité de l'application car c'est surtout eux qui utilise au quotidien ce site web.

L'ensemble des notes portées sont corrélées avec les éléments décrit lors de la reconstruction de l'application au chapitre 3 page 15.

5 Les évolutions nécessaire issue de la reconstruction

La reconstruction que nous venons d'effectuer ainsi que de son analyse nous amène à penser des évolutions nécessaire au bon fonctionnement de l'application afin de pouvoir intégrer au mieux les nouvelles fonctionnalités demandées.

5.1 L'architecture système

La modification qui sera effectué sera de migrer le SGBD vers un serveur MS SQL. L'application ne nécessite pas un tel niveau de disponibilité, de sécurité ou de reprise en cas de panne mais cela permettra une homogénéisation avec les autres applications de l'entreprise permettant ainsi aux équipes en place de prendre le "contrôle" sur l'application. Il n'y aura pas d'autre modification come par exemple pour le serveur WEB, les entrées DNS,…

5.2 L'architecture des pages web

La structure des pages web sera modifiée pour faire disparaître les tentatives de maintenance du site ainsi que les erreurs de programmation référencées. Nous ferons apparaître à ce stade une page d'administration du site qui est grandement nécessaire pour gérer les salles de réunion. On remarquera que le graphe est connexe et que les retours se font correctement vers la "home page".

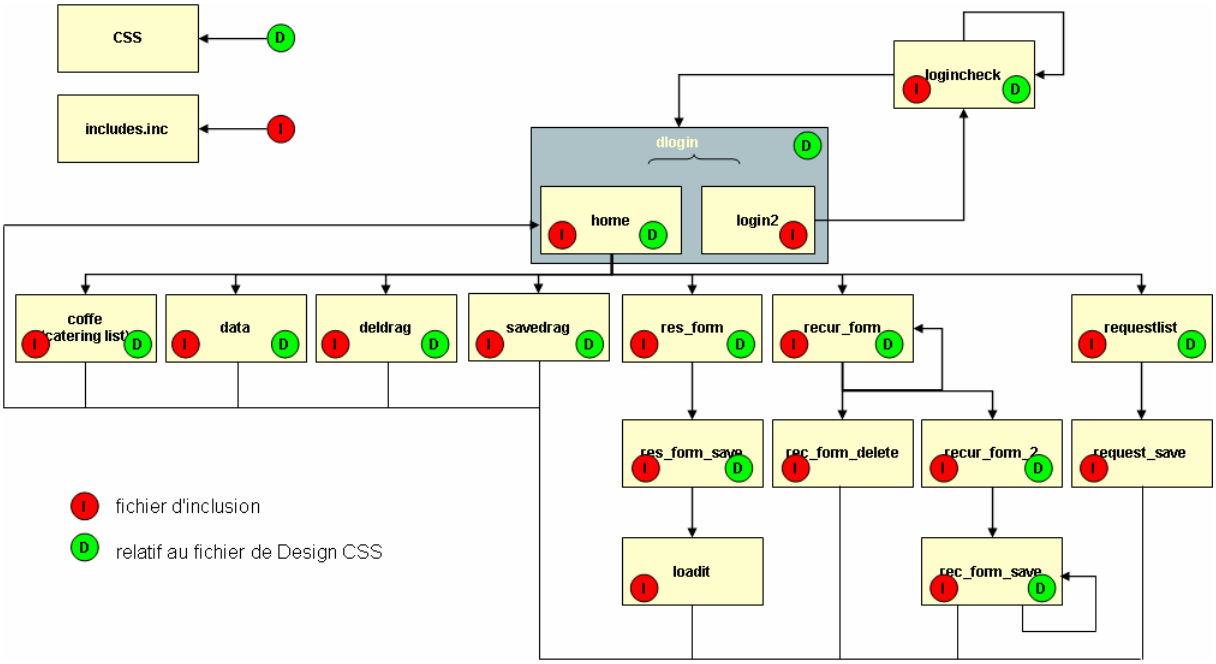

Figure 14 - Nouvelle structure des fichiers source

5.3 Base de données et optimisation

Suite à la reconstruction effectué sur le SGBD et la base de données et aux différentes remarque effectué, nous allons modifier cette ensemble afin de le rendre conforme aux bonnes pratiques et normalisé.

5.3.1 Le SGBD

La première pensé est de migrer notre base de données sur le moteur MS SQL plutôt que de le laisser sur Access 7.0. L'avantage sera la mise en œuvre de trigger et de fonction qui permettront de décharger l'application d'une partie de la gestion de la base comme par exemple lors de la suppression d'une salle de réunion, de la suppression d'une réservation récurrente ou encore lors de l'archivage dans les logs (fonction demandé par le MOA) d'une modification effectuée sur une réservation. L'application de ces quelques règles permettra d'obtenir une meilleure intégrité de la base et des données.

5.3.2 La base de données

C'est maintenant, fort de nos interrogations et propositions faite plus haut que nous allons modifier notre MCD pour le rendre conforme au réel que nous souhaitons avoir en y ajoutant les modifications nécessaires et demandées. C'est le MCD suivant.

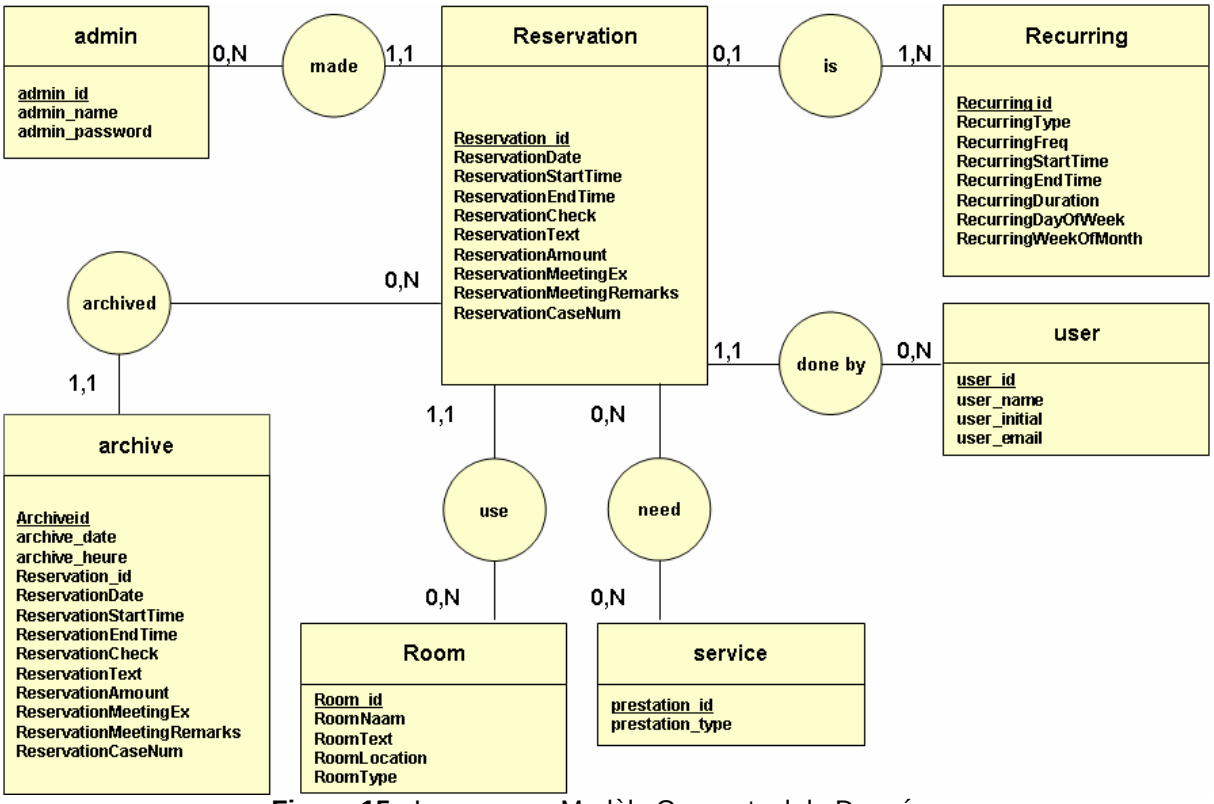

Figure 15 - Le nouveau Modèle Conceptuel de Données

La première remarque que l'on peut faire est que tous les noms d'attributs on changé afin de refléter le caractère international de l'entreprise. Cela doit permettre à un développeur d'une culture international (même Néerlandais) de pouvoir effectuer de nouvelles modifications si cela devenait nécessaire.

J'ai suivit mon raisonnement sur les attributs en doublon et ils ont disparu de la table "Recurring".

5.4 Le code source

Trois opérations vont être effectuées en priorité.

La première opération sera le "refactoring" qui va consister en la restructuration du code afin de le simplifier, de l'optimiser et de le documenter. On utilisera des fonctions ASP afin d'écrire le code HTML en prenant en compte l'existence d'un fichier de définition de

la charte graphique à travers le CSS. On créera aussi des fichiers d'inclusions dédié aux différents langages utilisé dans le code.

La seconde opération sera de débugger et de redévelopper les pages pour que les retours sur la page "Home" se fasse correctement et que le développement soit conforme aux bonnes pratiques de la programmation WEB. Mais aussi de mettre en œuvre plus de contrôle des champs par l'utilisation du "JavaScript".

Enfin, la troisième grande opération sera la mise en œuvre des fonctionnalités demandées ou manquantes (comme les pages d'administrations).

Quoi que pas tout à fais adapter à ce contexte de développement, je conseille quand même d'effectuer une programmation de type objet afin de mieux organiser la réutilisation des éléments, d'avoir un code mieux structurer, plus claire et aussi plus facile par la suite à modifier. Cela nécessite de revoir plus amont les besoins des utilisateurs afin de décrire cela dans divers diagrammes UML.

6 Les bonnes pratiques du développement WEB

On ne peut pas reconstruire une application WEB sans penser à la phase suivante de Forward Enginering et à l'utilisation des bonnes pratiques de développement spécifiques au mode WEB. L'utilisation des bonnes pratiques du WEB peuvent avoir un impact positif et significatif sur l'ensemble d'un projet. Ces bonnes pratiques peuvent être trouvées sur internet et ont pour focus toutes parties de l'application et du projet. J'en cite quelque uns principaux et commun mais chaque type de projet WEB peut avoir les siennes avec des spécificités financière, sécurité...

6.1 Mise en forme - CSS

Nous en avons déjà longuement parlé, mais ce point est si important que je n'hésite pas. L'utilisation de CSS fait gagner en simplicité, clarté et efficacité. Elle a aussi un impact positif lors de la reconstruction, le projet peu alors se voir ajouter un module dédier au graphisme du site. Il faut donc l'utiliser en préférence à toute autre possibilité.

6.2 Base de données et optimisation

Une base de donnée bien défini avec une séparation claire entre les données et les traitements y compris les triggers ou fonctions pour assurer l'intégrité d'un coté et l'application de l'autre est primordial. Il faut éviter le plus possible de coder en dur des informations variable ou des éléments devant gérer l'intégrité des bases… Il faut rendre à César ce qui lui appartient... de faire.

6.3 Conformité du site

Faire valider son code HTML par une DTD.
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional // EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
C'est une excellent façons de s'assurer que son code est valide, et nous avons vu que le notre ne l'était pas.

6.4 Interfaces

Il arrive trop souvent dans un application web de devoir jongler entre clavier et souris. Une bonne interface doit pouvoir être traiter au clavier seul, le faite de devoir utiliser la souris pour ce positionner sur le champ à remplir n'est pas une bonne pratique. D'autre part, une interface ne dois comporter d'information que pour ce qu'elle est censé faire. Il n'est donc pas normal de voir un formulaire "créer une réunion récurente" comporter un bouton supprimer une réservation récurrente.

6.5 Code source HTML

Fermé systématiquement les balises vides. Par conséquent, ne pas écrire ***
*** mais ***
*** ou ne pas écrire ***<hr>*** mais ***<hr />***.
Ecrire tous les attributs en minuscule et dans un formulaire, ne pas utilisé l'attribut ***checked*** mais ***checked="checked"***. Ces points faisant partie du standard de définition du W3C pour le WEB.

6.6 La gestion des différents langages utilisés

Il va aussi de soit qu'il faut prendre en compte que d'autre viendront relire et modifier notre travail. Dans un contexte international, une fusion peut intervenir et vous vous retrouver avec un code dans une langue que vous ne maitrisez pas. Utiliser l'Anglais comme langue international commune est primordial dans le cadre d'une entreprise présent sur les cinq continents et dans 60 pays dans le monde.

6.7 D'autres bonnes pratiques du web

On pourrait citer l'utilisation d'UML pour la conception, l'utilisation de classe ou code objet (même partiellement objet pour de l'ASP ou du PHP) afin d'avoir un code source propre, simple à maintenir et lisible sans effort.

On pourrait aussi citer l'utilisation du langage ASP pour l'écriture du code HTML afin de clarifier et simplifier le code source par appel de fonction. Pour exemple, une fonction "Debut_Formulaire()" pour générer en HTML le code nécessaire à l'écriture du début du code d'un formulaire. Plus lisible, on pourrait moins facilement oublier de fermé celui-ci.

D'autre part, l'utilisation du JavaScript coté client pour effectuer les vérification d'usage plutôt que de les faire coté serveur ferait gagner l'application en performance, la bonne séparation entre ce qui doit être fais coté client du coté serveur est importante et merite d'être respecté.

7 Conclusion

Le déclencheur d'une demande de reconstruction d'une application WEB est souvent une demande d'ajout de fonctionnalité. Malgré la simplicité de l'application, par sa base de donnée ou le nombre de ligne du code source, cette reconstruction aurait presque pu être scindée en deux projets distincts. Le premier pour reconstruire l'application et le second pour y effectuer les ajouts nécessaire tant l'application nécessite de modification ou de changement.

Nous avons passé en revue l'ensemble des éléments (infrastructure, base de données, schéma de la base, structure fichier, code source, design graphique de l'application,…) a travers différent filtre, tantôt pour contrôler la qualité des éléments, tantôt technique pour valider les choix effectuer, tantôt sémantique pour évaluer la manière dont cela avait été écris, tantôt conceptuel pour valider la conformité du model à l'exactitude du réel ou à la complétude. L'intérêt de ce travail été de présenter ou proposer des manières de faire suivant l'élément mis en lumière pour effectuer cette reconstruction.

Au delà de la manière d'effectuer cette reconstruction c'est en faite le fait que la reconstruction est constitué de plusieurs sous module pourtant chacun leur propre nom et fonction comme le "Design Recovery", la "Redocumentation", le "Restructuting" ou encore le 'Refactoring". Dans l'état actuel de notre application, difficile de passer à coter de tous ces modules de la reconstruction.
Mais la reconstruction à un coût non négligeable qui ne faut oublier. Cette élément financier est primordial car il doit permettre de choisir s'il faut effectuer cette reconstruction ou s'il vaut mieux repartir dans une approche plus classique du "Forward Engineering". Ce choix peut-être guidé par différents facteurs dont la qualité d'une application ou celle de la base, faut-il encore se pencher dessus pour s'en faire une idée. Facteur qualité qu'il faudra associer au temps passé à décortiquer l'application.

8 Annexe

Les annexes suivantes sont soit des fichiers complets ou des morceaux choisi afin d'illustrer la réalité du code source.

8.1 Listing du fichier "Dlogin.asp".

```
<%
response.expires=0
%>
<html>
<head>
     <title>BCG Room Reservation System</title>
     <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<frameset cols="0,*" frameborder="yes" border="0" framespacing="0" rows="*">
     <frame name="tool" src="">
     <%if request("LogIn")="ok" then %>
     <frame name="main" src="home.asp?LogIn=ok" scrolling="YES">
     <%else%>
     <frame name="main" src="login2.asp" scrolling="no">
     <%end if %>
</frameset>

<noframes>
<body bgcolor="#FFFFFF">
</body></noframes>
</html>
```

8.2 Listing du fichier "Login2.asp".

```
<!--#include file="acces_base.inc" -->
<%response.expires=0

%>

<html>
<head>
     <title>Buk-a-Rum</title>
</head>
<script>
     function PositionDiv()
           {
           if (document.body.offsetWidth > 800 && document.body.offsetWidth < 1024 )
                 {
                 document.all.menu.style.width= document.body.offsetWidth-24
                 }
           else if (document.body.offsetWidth >= 1024)
                 {
                 document.all.menu.style.width= 1000
                 }
           else
                 {
                 document.all.menu.style.width = 776
                 }
           }

</SCRIPT>
<link rel="stylesheet" href="verdana.asp">
<body  onload="PositionDiv()" onresize="PositionDiv()"  bgcolor="FCFDF1" background="img/bg.jpg" leftmargin="0"
topmargin="0" marginwidth="0" marginheight="0" link="#333333" vlink="#333333" alink="#333333" >
<DIV ID="menu" style="position:absolute; top:0px; left:0px; width:100%;">
     <table width=100% background=img/top.jpg height=78>
           <tr>
                 <td> </td></tr>
     </table>
     <table width=100% border=0 cellpadding=0 cellspacing=0>
```

```
          <tr>
                <td background=img/nav.gif align=left>
                <table width=97% border=0 cellpadding=0 cellspacing=0>
                      <tr>
                            <td class=v10 >   <b>Today:</b>
 <%=FormatDateTime(now(),vblongdate)%></td>
                            <td class=v12 align=right>
                                  <b><font color=ffffff>BCG Reservation System</font></b><br></td></tr>
                </table></td>
                <td background=img/nav.gif>
                      <img src=img/nix.gif width=10 height=23></td></tr>
    </table>
</div>

<div id=tijdschaal Style="width=720;visibility:visible;left:14;top:130;position:absolute;">
<form action="loginCheck.asp" method="post">
    <table width=300 cellpadding=3 border=0 cellspacing=0 background=img/nav.gif>
          <tr>
                <td class=v12 colspan=2 bgcolor="E1692E">
                      <font color="ffffff">Please login with your user name and password</font></td></tr>
          <tr>
                <td class=v12>User name</td>
                <td class=v11>
                      <input type="text" name="AdministratorUser" class="v11" style="Width:100"></td></tr>
          <tr>
                <td class=v12>Password</td>
                <td class=v11><input type="password" name="AdministratorPassword" class="v11"
style="Width:100"></td></tr>
          <tr>
                <td class=v12> </td>
                <td class=v12><input type="submit" name="Submit" value="Log In" class="v11" style="Width:100" size="4"
></td></tr>
    </table>
</form>
</div>

</body>
</html>
```

8.3 Listing du fichier "logincheck.asp".

```
<%response.expires=0
login=""
set rst=CreateObject("Adodb.recordset")
sql="select * from Administrator"
rst.open sql,Session("LinkToDatabase"),0,1
while not rst.eof
    if  rst.fields("AdministratorUser")=request("AdministratorUser")  and
rst.fields("AdministratorPassword")=request("AdministratorPassword") then
    login="ok"
    end if
    rst.movenext
wend
rst.close
set rst=nothing

if login="ok" then
%>
<html>
<body onLoad="javascript:document.form1.submit();">
    <form method="post" name="form1" action="DLogin.asp">
    <input type="hidden" name="LogIn" value="ok">
    </form>
    <%
    else
    %>
    <html>

<body>
    <script>
    alert('Your username and password combination are incorrect, please try again.')
    </script>
    <%
end if
%>

</body>
</html>
```

8.4 Listing du fichier "Coffee.asp".

```
<%
response.expires=0
dag=request("viewday")
maand=request("viewmonth")
jaar=request("viewyear")
datum=jaar & "," & maand & "," & dag
%>
<html>
<head>
     <title>Catering List</title>
</head>
<link rel="stylesheet" href="verdana.asp">
<body>
<table cellpadding=1 cellspacing=0 border=0>
<tr><td class=vb20 colspan=8><font color=CC0000>Catering List</font></td></tr><tr>
<td class=vb14 colspan=8><i><%=FormatDateTime(datum,vblongdate)%></i></td></tr>
<tr><td colspan=8> </td></tr>
</table>
<table cellpadding=2 cellspacing=0 border=1 bordercolorlight="#CC6600" bordercolor="#CC6600" bordercolordark="#CC6600">
<tr>
<td class=v12 width=100 align="center">Time</td>
<td class=v12 width=100 align="center">Room name</td>
<td class=v12 width=50 align="center">Participants</td>
<td class=v12 width=60 align="center">Coffee</td>
<td class=v12 width=60 align="center">Drinks</td>
<td class=v12 width=60 align="center">Breakfast</td>
<td class=v12 width=60 align="center">Lunch</td>
<td class=v12 width=150>Comments</td>
</tr>
<tr>
<td colspan=8></td>
</tr>
<%
set rst=CreateObject("Adodb.Recordset")
sql="Select * from Reservations INNER JOIN Rooms ON Reservations.Reservation_RoomId = Rooms.RoomID where
ReservationDate=dateserial(" & jaar & "," & maand & "," & dag & ") order by ReservationStartTime"
rst.open sql,Session("LinkToDatabase"),0,1
```

```
do while not rst.eof
reservationName=rst.fields("Reservationname")
reservationStartTime=rst.fields("reservationstarttime")
reservationEndTime=rst.fields("reservationendtime")
reservationtext=rst.fields("Reservationtext")
reservationamount=rst.fields("Reservationamount")
reservationCoffee=rst.fields("ReservationCoffee")
reservationFris=rst.fields("ReservationFris")
reservationBreakfast=rst.fields("ReservationBreakfast")
reservationLunch=rst.fields("ReservationLunch")
reservation_roomid=rst.fields("Reservation_roomid")
roomnaam=rst.fields("roomNaam")
tmp=int(reservationStartTime)
Select case (reservationStartTime-tmp)
      Case 0
            tmp=tmp & ":00"
      Case .25
            tmp=tmp & ":15"
      Case .5
            tmp=tmp & ":30"
      Case .75
            tmp=tmp & ":45"
      Case Else
            tmp=tmp & ":00"
End Select
starttijd=tmp
tmp=int(reservationEndTime)
Select case (reservationEndTime-tmp)
      Case 0
            tmp=tmp & ":00"
      Case .25
            tmp=tmp & ":15"
      Case .5
            tmp=tmp & ":30"
      Case .75
            tmp=tmp & ":45"
      Case Else
            tmp=tmp & ":00"
End Select
eindtijd=tmp
```

```
%>
<tr>
<td class=v10 valign=top align="center"><%=starttijd%> - <%=eindtijd%> </td>
<td class=v10 valign=top align="center"><%=RoomNaam%> </td>
<td class=v10 valign=top align="center"><%=reservationamount %> </td>
<td class=v10 valign=top align="center"> <% if ReservationCoffee="yes" then%><img src="img/check.gif" width="15"
height="15"><%end if%> </td>
<td class=v10 valign=top align="center"> <% if ReservationFris="yes" then%><img src="img/check.gif" width="15"
height="15"><%end if%> </td>
<td class=v10 valign=top align="center"> <% if ReservationBreakfast="yes" then%><img src="img/check.gif" width="15"
height="15"><%end if%> </td>
<td class=v10 valign=top align="center"> <% if ReservationLunch="yes" then%><img src="img/check.gif" width="15"
height="15"><%end if%> </td>
<td class=v10 valign=top ><%response.write replace(reservationtext,vbcrlf,"<BR>")%> </td>
 </tr>
<%
          rst.movenext
          loop
          rst.close
          set rst=nothing

%>
</body>
</html>
```

8.5 Listing du fichier: "res_form_save.asp".

```
<!--#include file="acces_base.inc" -->
<%
response.expires=0
msg=""
DateOld=""
dag=request("Dag")
maand=request("Maand")
jaar=request("Jaar")
datum=Dateserial(request("Jaar"),request("maand"),request("dag"))
if request("reservationId")="" then
resid=0
else
resid=request("reservationId")
end if
if not request("actie")="verwijder" then
                '' Form Check
                If request("ReservationName")="" then
                msg = msg & "Initials have to be filled in.<br>"
                end if
                 If request("ReservationStarttime")*100 > request("ReservationEndTime")*100 then
                msg = msg & "The reservation must begin earlier than it ends.<br>"
                end if
                If request("ReservationEmail")="" then
                msg = msg & "Name must be be filled in.<br>"
                end if
                If not request("ReservationAmount")="" then
                                set rst=CreateObject("Adodb.Recordset")
                                sql="Select * from Rooms where roomid=" & request("Reservation_roomid")
                                rst.open sql,Session("LinkToDatabase"),0,1
                                if not rst.eof then
                                      err.clear
                                      on error resume next
                                       If Cint(request("ReservationAmount"))>Cint(rst.fields("RoomMax")) then
                                       msg = msg & "The maximum of this room is " & rst.fields("RoomMax") & ".<br>"
```

```
                                    end if
                                end if
                                rst.close
                                set rst=nothing
                            else
                                msg = msg & "Number of people has not been filled in.<br>"
                end if
                if not err.Number=0 then
                msg=msg & "A mistake has ocurred. Please check the form.<br>"
                end if
                on error goto 0
                '' check if start time en/of end time tussen de afspraken vallen
                    set Check2=CreateObject("Adodb.Recordset")
                    sql2="Select * from Reservations where reservation_roomid=" & request("reservation_roomId") & _
                     " AND ReservationDate=dateserial(" & jaar & "," & maand & "," & dag & ")" & _
                     " AND ReservationId <>" & resid & _
                     " AND ( ((ReservationStartTime*100 between " & Request("ReservationStartTime")*100 & " AND " &
Request("ReservationEndTime")*100 & _
                    " and ReservationStartTime*100 <> " & Request("ReservationEndTime")*100  & _
                    ") OR (ReservationEndTime*100 between " & Request("ReservationStartTime")*100 & " AND " &
Request("ReservationENdTime")*100 & _
                    " And ReservationEndTime*100 <> " & Request("ReservationStartTime")*100 & _
                    " )) " & _
                    " OR (( " & Request("ReservationStartTime")*100 & " between ReservationStartTime*100 and
ReservationEndTime*100 " &  _
                    " and  " & Request("ReservationStartTime")*100 & " <> ReservationEndTime*100" &  _

                    " ) OR (" & Request("ReservationEndTime")*100 & " between ReservationStartTime*100 and
ReservationEndTime*100 " &  _
                    " and " & Request("ReservationEndTime")*100 & " <> ReservationStartTime*100  " &  _
                    " )) )"

                    Check2.open sql2,Session("LinkToDatabase"),0,1

                    if not Check2.eof then
                    msg=msg & "A reservation on this time and date has already been made.<br> Please remove this
before making this reservation."
                    end if
                    Check2.close
                    set Check2=nothing
```

9 Biblio

Web
http://www.w3c.org
http://fr.selfhtml.org/index.htm

Livresque

- Design Web: utiliser les standard CSS et XHTML – de Jeffrey Zeldman – EYROLLES – ISBN: 2-212-11548-2