|  |
| --- |
| Département Informatique  Promotion 2019/2020 |

**Rapport de stage présenté en juillet 2020  
par**

**­**

**PRAGA Yvain**

**en vue de l’obtention de la licence CIASIE**

Remerciements

Je remercie tout d’abord la Banque Internationale à Luxembourg pour m’avoir pris comme stagiaire au sein de leurs locaux.

Je remercie également mon maitre de stage William pour m’avoir guidé et aidé tout le long de ce stage.

# Sommaire

[1. Introduction 5](#_Toc11609506)

[2. Présentation de l’entreprise 6](#_Toc11609507)

[3.Memoire 7](#_Toc11609508)

[5. Conclusion 44](#_Toc11609513)

[6. Références 45](#_Toc11609514)

Introduction

L’entreprise dispose d’une application web interne qui offre plusieurs fonctionnalités aux agents pouvant accéder à celle-ci. Elle permet de répertorier les différents évènements du jour, consulter des évènements passés, constater l’état du matériel physique ou encore la diffusion des informations entre agents.

L’application tourne actuellement sur un serveur **Windows Server 2012 R2**. Cependant celui-ci arrive bientôt à expiration. Pour se préparer à cet évènement, l’entreprise a décidé de migrer l’application sur un nouveau serveur. Celui-là sera un **Windows Server 2016**.

Cependant, lors d’une première tentative de migration de l’application en question, de nombreux problèmes ont émergé au sein de celle-ci. Des erreurs de dates, de mise en forme, de technologie. En effet, elle a été développée pour être exclusivement visionnée par le navigateur Internet Explorer. Comme ce navigateur posait des problèmes de sécurité, de mise à jour ou encore de performances, il a été convenu de rendre l’application plus cross-browser en lui permettant d’être exécutée sous le navigateur **Google Chrome**, second navigateur disponible sur le matériel informatique des employés de l’entreprise, avec Internet Explorer.

Pour résoudre cette problématique, deux choix se présentaient à nous. Le premier était de réécrire tout l’application dans un autre langage tel que Java par exemple. Le second choix était de corriger les différents bugs, problèmes qui survenaient tout en **gardant l’application tel quel**.

Nous avons donc convenu avec mon chef, de garder la deuxième solution. En effet, le logiciel applicatif possède bon nombre de sous-applications, fonctionnalités et services. Suffisamment pour que je ne puisse tout réécrire dans le temps imparti du stage. J’ai dû donc

L’entreprise m’a donc embauché pour gérer avec mon supérieur hiérarchique, la migration de l’ancien vers le nouveau serveur, ainsi que les problèmes sous-jacents qui surviennent dans l’application au moment de cette migration.

Présentation de l’entreprise

La BIL, Banque Internationale à Luxembourg, est une banque luxembourgeoise située à Luxembourg-ville au Luxembourg. Il s ‘agit de la banque universelle la plus ancienne du Grand-duché ou Luxembourg.

Elle a été fondée en 1856 et propose des services de trésorerie, marché financiers, de banque privée, d’entreprise et de détail. Au Luxembourg, la banque dispose d’un réseau d’une quarantaine d’agences. Elle propose une série de produits et services à sa clientèle constituée de particuliers, de Petites et Moyennes Entreprises et de grandes entreprises. À l’international, la BIL offre des services de banque privée et de gestion de fortune en proposant des services de gestion et de structuration de patrimoine ainsi que des solutions d'investissement sur mesure.

Sa forme juridique est société anonyme. Une société anonyme est une société commerciale considérée fiscalement comme une société de capitaux, ayant un capital social composé d’action.

Elle compte à ce jour environ 2000 employés et collaborateurs dans le monde entier et figure parmi les plus grands employeurs du Luxembourg. Elle a joué un rôle actif dans le développement économique du Luxembourg depuis sa création.



Ci-dessus, les locaux de la BIL à Luxembourg mais également mon lieu de travail pour le stage.

A noter que la BIL en 2000 est devenue une filiale du groupe bancaire Dexia. Une fusion entre le Crédit Local de France et le Crédit Communal de Belgique.

La BIL est présente aujourd’hui avec des filiales dans plusieurs pays à travers le monde, notamment en Suisse et au Danemark. Mais également avec des succursales à Dubaï, Singapour, Bruxelles et Bahreïn.

Mon intégration

Le service dans lequel j’ai été intégré travaille dans le support matériel, la sécurité et l’administration de l’entreprise. Ce service comprend six employés, chacun a un travail différent et à part mon supérieur qui possède quelques notions, aucun d’entre eux ne fait de l’informatique.

Ils m’ont cependant bien accueilli et mis à l’aise dans leur bureau. Mon environnement de travail se constitue d’un bureau avec un ordinateur à distance. C’est-à-dire que le matériel physique n’est pas dans la même salle que moi mais dans une pièce avec d’autres machines. Ce que j’ai avec moi est un petit boîtier pour faire la connexion à cet ordinateur. J’ai également deux écrans, une souris et un clavier.

L’entreprise dispose de deux parkings pour permettent aux employés de se garer sans problèmes. Une cantine et une cafétéria sont également intégrées pour manger sur le lieu de travail, petit-déjeuner comme déjeuner.

Habitant à Metz, j’effectuai le trajet entre mon lieu de travail à Luxembourg-ville et mon domicile chaque jour. Je prenais le vélo pour me rendre à la gare de Metz puis le train jusqu’à Luxembourg-ville et je finissais à pied pour rejoindre l’entreprise.

Premier pas

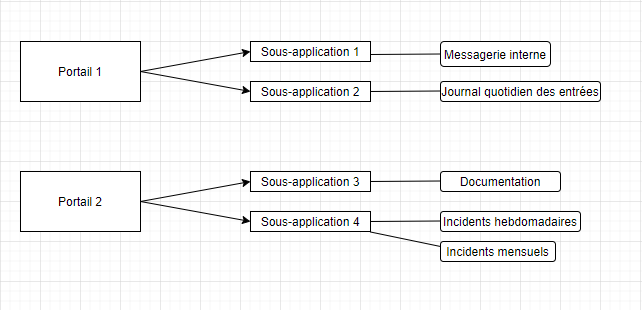
Introduction

Avant de me lancer tête baissé dans le code, il fallait effectuer un travail d’étude. Etudier l’application côté client, le langage utilisé et les logiciels nécessaires au développement de celle-ci.

La première étape fut d’appréhender l’application sur laquelle j’allai travailler.

Découverte de l’application

La structure de l’application est assez basique. On dispose de **deux portails** qui répertorient les sous-applications. Sur chaque sous-application, on retrouve un ou **plusieurs services et/ou fonctionnalités**. Ci-dessous, un schéma d’exemple.

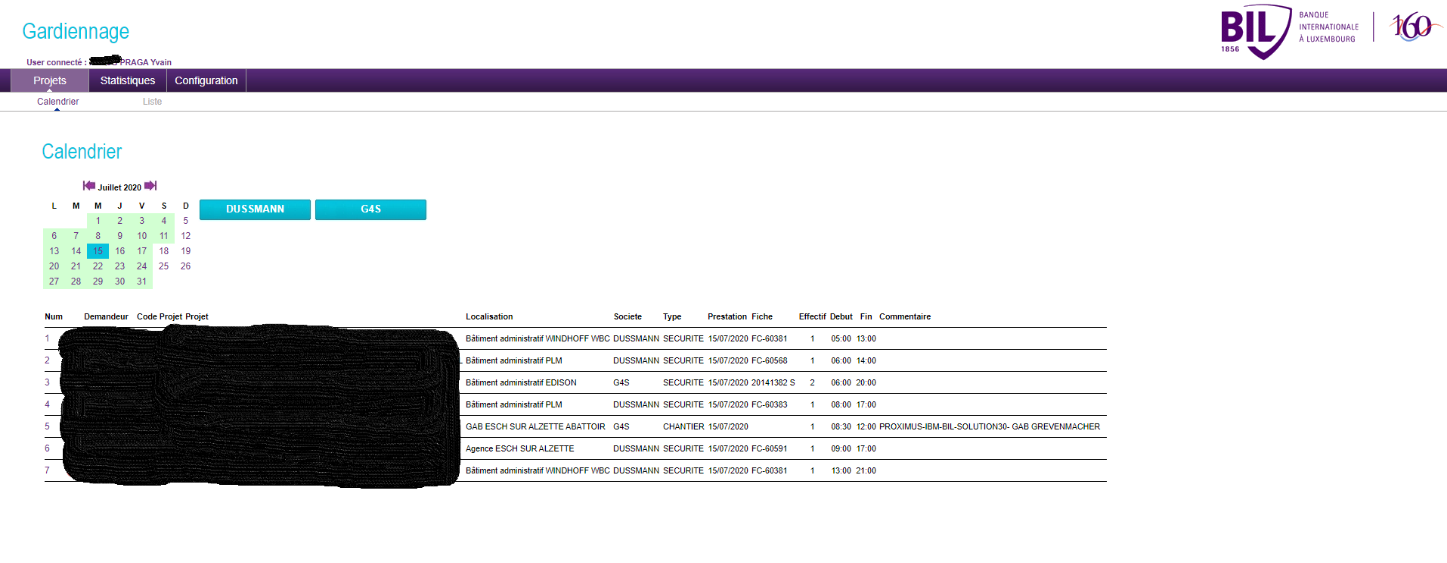


L’application possède plusieurs dizaines de sous-applications, dont la plupart a été codée pour être intégrée à celle-ci. Cependant, plusieurs font parties d’autres logiciels applicatifs extérieurs au nôtre. Et elles sont ajoutées pour répondre au besoin de l’agent.

Ces services et fonctionnalités sont là pour faciliter le travail des agents. Comme liste non-exhaustive des services et fonctionnalités, nous avons un journal quotidien, mensuel et annuel d’évènements divers tel que des incidents au sein de la BIL ou sur des installations extérieures appartenant à la BIL (pour exemple un distributeur de billet). On retrouve des sous-applications de gestion du matériel physique de la BIL tel que des clés, des répertoires des prises de services des agents ou de leur fin de service. D’autres sont là pour lister plusieurs documents et informations utiles.

Plusieurs base de données stockées sur le serveur viennent remplir notre application. Mais aucune API ou service web n’est utilisé pour récupérer les données.

Ci-dessus un exemple de sous-application :



Par soucis de sécurité, je vais cacher certaines informations. C’est aussi pour cette raison que je ne peux montrer les deux portails. En revanche je peux montrer certaines sous-applications, à condition de cacher les données sensibles.

Cette sous-application nommée Gardiennage est un exemple simple, on dispose d’un calendrier pour varier entre le jour, le mois ou encore l’année, ceci dans le but de voir les données associées à la date choisie.

Les données sont représentées sous forme de tableau avec différentes colonnes pour le type d’information. De plus on peut interagir avec chaque ligne en cliquant dessus. Cela nous permet de modifier, supprimer ou ajouter des données.

Et enfin, certains onglets sur le menu violet, permettent de naviguer entre les différentes fonctionnalités que proposent la sous-application. Dans notre cas, nous disposons d’une fonctionnalité Statistiques et configuration. Elles nous permettent d’établir des statistiques sur les données et de changer l’affichage de celles-ci dans le tableau qui se trouve dans l’onglet d’accueil.

Premier pas

La seconde étape du mon travail fut la prise en main du langage.

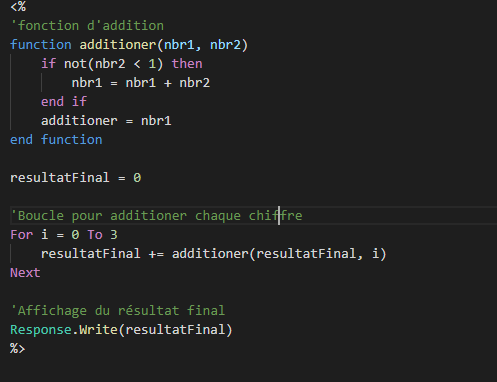
Découverte du langage

L’application sur laquelle je travaille est écrite en **ASP Classic** pour le côté serveur, HTML/CSS et Javascript pour le côté client. Maitrisant déjà l’HTML/CSS et le JavaScript, seul l’ASP Classic fut une découverte pour moi.

ASP classic ou Active Pages Server est un langage côté serveur développé en 1998 par Microsoft. Lil n’est à ne pas confondre avec le langage **ASP.net**, développé 3 ans plus tard par Microsoft encore. Malgré un nom similaire, ces langages ne sont pas du tout pareils. **ASP classic est interprété** et **.NET compilé**. Pour être clair sur ces deux termes, un programme dans un langage interprété (Classic) est exécuté au fur et à mesure de la lecture de celui-ci. Tandis qu’un programme compilé (.NET), sera lu et exécuté entièrement dès le début.

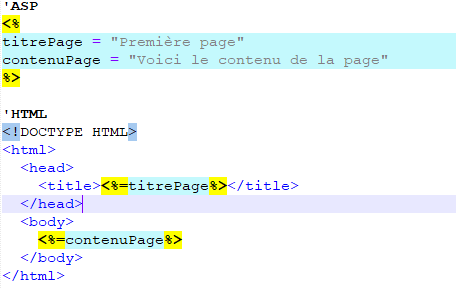
L’ASP classic bien que plutôt ancien, plus de 20 ans, propose un minimum de **fonctions implémentées au langage** pour faciliter le travail du codeur. Je parle de fonctions basiques, tel que la récupération d’informations de l’ordinateur de l’utilisateur (variables d’environnement, date, …) ou la manipulation de chaîne de caractères, de listes ou encore de nombres. Cela peut sembler plutôt évident mais il est agréable de ne pas avoir à programmer soi-même ce genre de fonctions. De plus, on retrouve encore quelques forums traitant de problèmes et questions rencontrés avec l’ASP Classic. On est loin d’avoir la communauté d’un langage populaire comme Python. Cependant, ces **divers forums** m’ont bien aidé pour m’aider dans mon travail.

Petit exemple d’ASP Classic dans lequel nous additionnons les chiffres de 0 à 10.



Le langage en lui-même est **clair et intuitif**. Cependant, il n’inclut pas certaines **règles de syntaxes de programmation**, comme les accolades, utilisées dans les langages que j’utilise ces derniers temps (PHP, JavaScript, Python). En revanche, l’ASP Classic a l’avantage de pouvoir **intégrer facilement de l’HTML**. C’est-à-dire que dans une page, on va pouvoir construire la page HTML pour le client mais également intégrer les requêtes, calculs, tâches faits côté serveur, dans cette même page. Mais cela ne veut pas dire que l’on peut exécuter du code serveur à n’importe quel moment.

C’est lors du chargement de la page que le code serveur est traité, viens ensuite le code client. Dans cet exemple, on déclare des variables et on les initialise en ASP, puis on affiche leur valeur dans la page HTML :



La partie HTML comme une page HTML classique peut intégrer des fichiers CSS, JavaScript. Mais aussi des librairies, framework et outils web. Tels que JQuery, FontAwesome ou encore W3.CSS.

Malgré cette facilité d’intégrer plusieurs langages dans un seul fichier, rien n’empêche de séparer l’HTML et l’ASP dans des fichiers différents.

Dans mon cas, la solution de **d’intégrer les langages dans un seul fichier** a été la plus utilisée par l’ancien développeur.

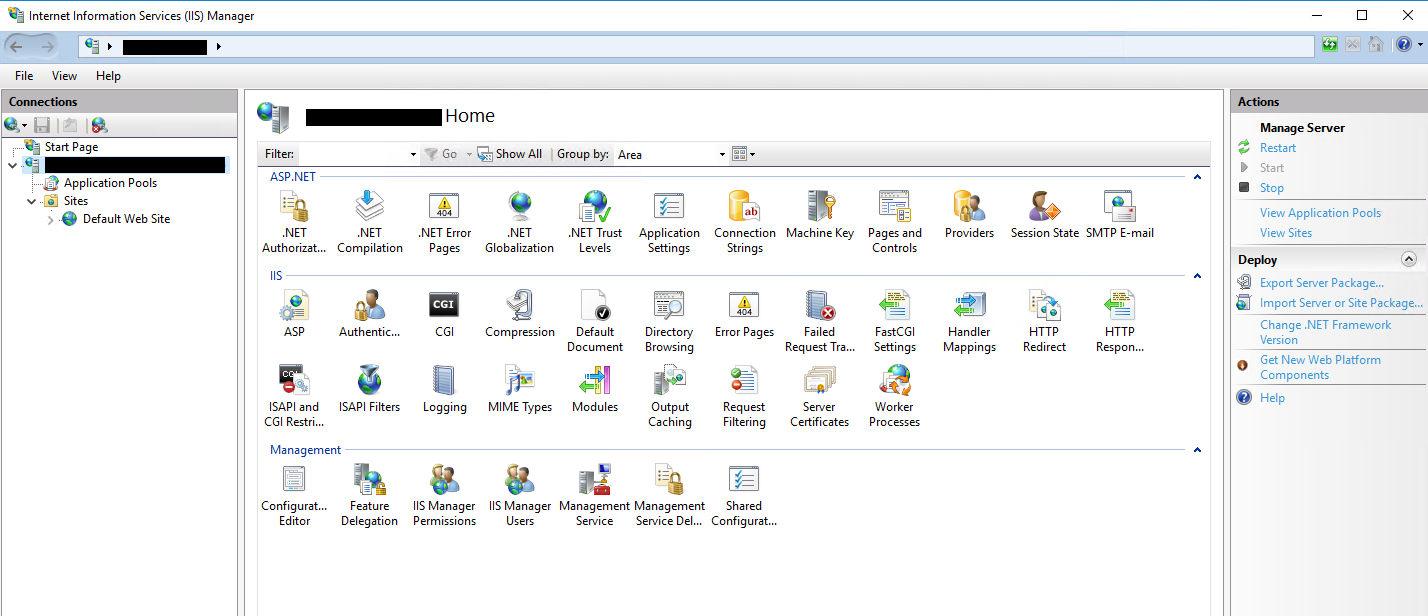
Premier pas

J’ai ensuite appris à utiliser différents logiciels utilisé pour le développement de l’application. Cet apprentissage s’est déroulé plutôt lors de mon travail de migration que pendant l’étude.

Découverte de l’IIS

Tout d’abord le logiciel IIS, **Internet Information Services** ou Gestionnaire des Services Internet en français.

Ce logiciel sert de serveur web pour le système d’exploitation Windows sur plusieurs versions. Grâce à IIS, on peut facilement **gérer le déploiement** et le **paramétrage** de son application web. En effet, de nombreux modules sont et peuvent être intégrés à notre application. Pour citer quelques fonctionnalités, nous pouvons gérer les redirections en cas d’erreurs (400, 500, …), récupérer les logs de connexion, faire de la réécriture d’URL, gérer les accès et les authentifications, filtrer les requêtes ou encore ajouter des méthodes d’authentifications. Ci-dessous l’interface du logiciel :



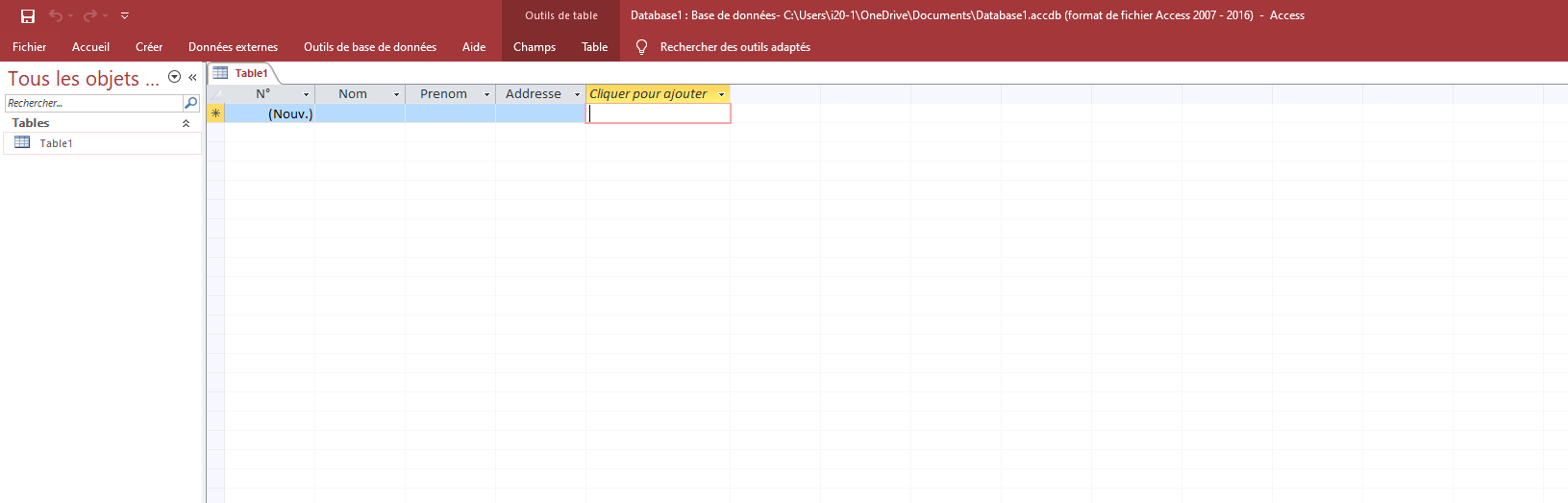
On retrouve avec IIS, les fonctionnalités qu’offre Apache2 pour Linux. J’ai pu utiliser ces deux outils durant ma licence professionnelle. Ce sont de bonnes solutions en termes de gestion et déploiement de serveur web. Un des avantages d’IIS est l’interface, elle est plutôt intuitive, il n’y a pas besoin de faire de la ligne de commande avec IIS. En revanche cet outil est disponible seulement sous Windows.

Cet outil m’a dont permit de paramétrer l’application sur le nouveau serveur.

Découverte de Microsoft Access

Le deuxième logiciel que j’ai appris à utiliser fut **Microsoft Access**. Il s‘agit d’un outil faisant partie de la suite Microsoft Office et permettant d’**éditer des base de données**.

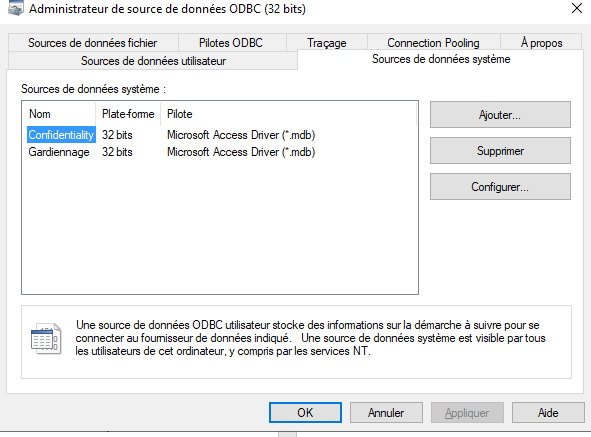
Ci-dessous l’interface de ce logiciel :



Access est plutôt simple à utiliser lorsqu’on est familier avec les logiciels de la suite Office, tel que Word. Avec cet outil j’ai donc pu manipuler les bases de données et les étudier. J’ai également pu tester la fonctionnalité permettant de créer des requêtes et de les exécuter. Cependant je ne maitrise clairement pas cette fonctionnalité mais je n’en ai pas eu grand besoin.

La plupart des bases de données sont stockées sur le serveur avec l’application. Cependant l’application se connecte à quelques serveurs SQL à distance. Pour mener à bien la communication entre les différentes bases de données et l’application, il fallut reparamétrer l’ODBC du nouveau serveur.

ODBC (Open Database Connectivity) est un intergiciel qui permet à une application informatique de manipuler plusieurs bases de données dans un environnement Windows.



Il suffit de déclarer chaque base de données dans cette interface et de lui donner un nom. Chaque application voulant en utiliser une répertoriée dans cette liste, devra pointer vers son nom.

La migration

Introduction

Une fois ce travail de prise en main et d’observation effectué, j’ai pu commencer mon travail de migration. La problématique principale de de ce projet est la suivante :

« Lors de la migration de l’application, de nombreux problèmes ont émergés rendant impossible l’utilisation de celle-ci »

Mon chef avait préalablement tenté de migrer cette application seul. Il a ensuite remarqué ces différents problèmes et a voulu les résoudre. Cependant, il y avait tellement d’erreurs, qu’il a finalement décidé de m’embaucher pour les régler.

Je vais présenter dans cette partie, les erreurs et problèmes majeurs rencontrés lors de la migration et comment je les ai corrigés.

Le navigateur

Un des problèmes majeurs fut le navigateur. En effet, l’application entière a été développée pour être compatible qu’avec Internet Explorer. C’est-à-dire que l’application ne s’exécutera pas correctement avec Google Chrome, Opera, Mozilla Firefox, ou d’autres navigateurs. Cela pose problème puisque Internet Explorer est le navigateur le plus à la traine et limitant technologiquement de nos jours. C’est pourtant le logiciel de navigation web par défaut sur plusieurs versions de Windows notamment sur celle utilisée par la BIL. Des failles de sécurité et des non-comptabilités HTML et styles CSS sont à répertorier. Les ordinateurs utilisés par les employés se servant de l’application disposent d’Internet Explorer ainsi que Chrome. Nous avons donc convenu de corriger l’application pour pouvoir l’utiliser sous Chrome. Pour alléger la lecture et l’écriture, je vais raccourcir le nom Internet Explorer par IE.

1. Le CSS et HTML

Puisque IE est le navigateur le plus limitant pour les styles et propriétés CSS, corriger le CSS n’a pas été une tâche ardue. Au contraire, la mise en forme de l’application et le codage de celle-ci étant basique, j’ai pu m’en occuper assez rapidement. On retrouve un seul fichier CSS pour une sous-application entière, pas de langages et technologies que j’ai pu apprendre lors de ma licence CIASIE (frameworks CSS, SASS, technique de dispositions, …). Mise à part quelques infimes changements, je n’ai pas touché au style de l’application. Les agents l’utilisant depuis quelques années, ils ont développé des habitudes d’utilisation, changer l’interface leur demanderai un certain temps d’adaptation pour réinstaller une routine. De plus, développer un design qui conviendrait à tous les utilisateurs n’est pas dans mon champ de compétences pour le moment et cela m’aurai demandé beaucoup de temps. Il aurait fallu récolter les différentes demandes des utilisateurs en termes de design, les réaliser puis qu’elles soient validées pour finalement les appliquer sur toute l’application.

Pour la partie HTML, j’ai pu remarquer une tendance à complexifier et rallonger inutilement la structure HTML des pages clients. Effectivement, plusieurs balises sont redondantes et inutiles. D’autres sont là pour servir de technique de disposition des éléments. A défaut d’utiliser des techniques de disposition CSS tel que flexbox, l’ancien développeur a donc préféré ou a dû utiliser des balises HTML pour placer les éléments HTML.

1. La sécurité

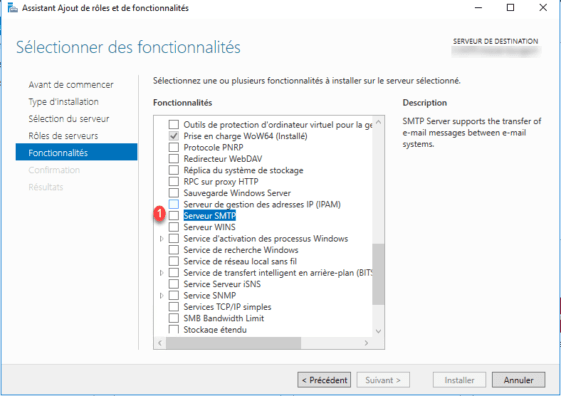
Certaines sous-applications propose comme fonctionnalité d’envoyer des mails pour partager des données à certains services de l’entreprise. Pour effectuer cela, l’application utilise ActiveXObject qui est un objet inclus dans le langage JavaScript. Cependant, il n’est disponible que sous IE. Il permet d’exécuter des commandes dans l’interface système Shell. Dans notre cas, le programme ouvrait le logiciel Outlook pour préremplir les différents champs et laisser l’utilisateur valider ou modifier avant d’envoyer l’e-mail.

Malheureusement, ActiveXObject permet de faire plus que cela. J’ai pu par exemple éteindre mon ordinateur professionnel avec cet objet. Le problème de sécurité est plutôt évident mais comme exemple, un développeur mal intentionné pourrait programmer une fork bomb qui s’exécuterai sur chaque ordinateur accédant à l’application.

C’est pour cette raison que cet objet n’est pas disponible sur d’autres navigateurs tel que Chrome. Pour garder cette fonctionnalité il fallait donc faire autrement que d’ouvrir Outlook. Ma première solution était d’utiliser la fonction mailto :

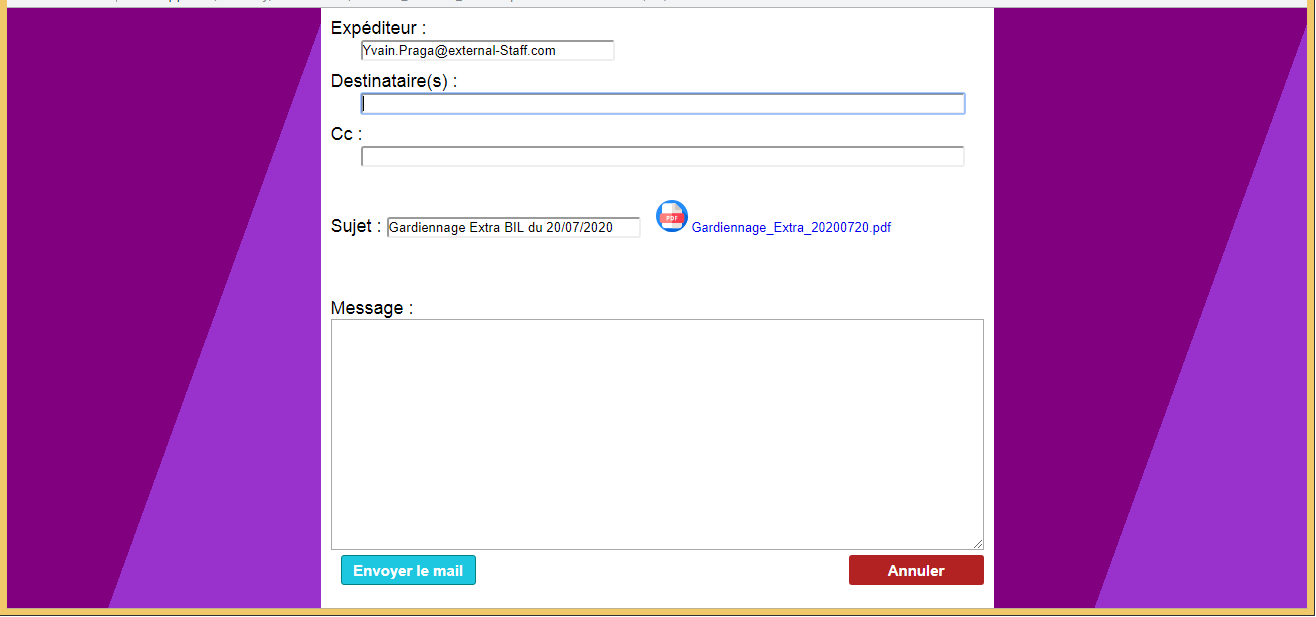


Cependant, on ne peut inclure de pièce jointe. Ma deuxième solution fut d’envoyer un mail avec la technologie CDO (Collaboration Data Objects). Cette technologie est disponible avec l’ASP Classic et permet d’envoyer des mails côté serveur. Cependant, il a été nécessaire d’activer une fonctionnalité sur le serveur hébergeant l’application.



Cette fonctionnalité se nomme serveur SMTP. SMTP (Simple Mail Transfer Protocol) est un protocole d’envoi d’e-mail d’un serveur à un autre. Grâce à celui-ci j’ai pu donc créer une messagerie dans l’application.

Ci-dessous l’interface utilisateur que j’ai programmé, basique mais fonctionnelle, je peux joindre des fichiers et préremplir les différents champs.



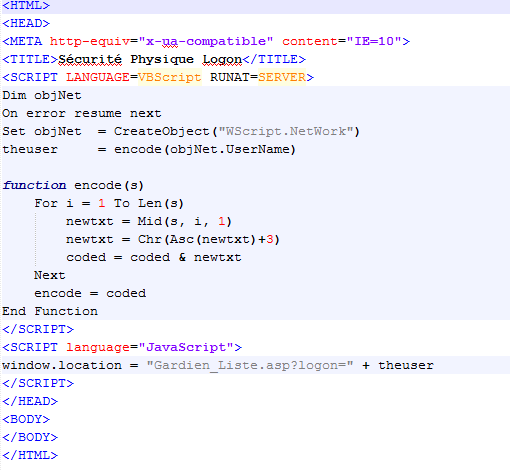
Seul désavantage de cette solution, l’e-mail n’est pas répertorié dans la boite « Envoyé » de la messagerie l’expéditeur. En revanche le destinataire reçoit tout de même l’e-mail. De plus, l’ancienne solution qui consistait à ouvrir l’application Outlook pouvait prendre jusqu’à 35 secondes rien que pour charger le logiciel de messagerie. Tandis que la messagerie intégré ne prendra pas plus de deux secondes pour l’ouverture et l’envoi.

La migration

Authentification

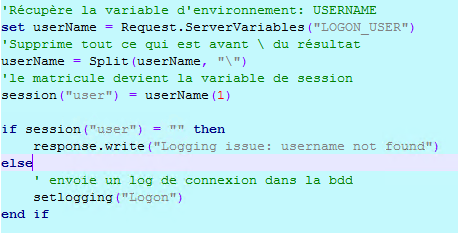
L’application dispose d’un **système d’authentification** basé sur le **matricule de l’utilisateur**. Au sein de l’entreprise, tout employé en possède un. Grâce à celui-ci, l’utilisateur peut accéder à plus ou moins de contenu selon ses droits. Chaque page de l’application inclus deux fichiers avant de s’exécuter. Un qui répertorie les fonctions utilisées plusieurs fois dans chaque page et un second qui va charger les variables de session utiles mais également gérer l’authentification. On peut apparenter cette étape à un middleware.

Initialement, cette tâche se faisait par le biais d’un **fichier html externe**. Ci-dessous, le fichier en question qui vérifie l’authenticité de l’utilisateur tentant d’accéder à l’application :



Mais en passant sous Chrome, nous pouvons le faire avec seulement quelques lignes en ASP. Elle permet de récupérer la **variable d’environnement** LOGON\_USER qui est présente sur les ordinateurs des employés et a comme valeur leur matricule. Il suffit ensuite de vérifier l’existence de cet agent dans la base de données ainsi que ses droits pour ensuite lui donner les accès qui lui ont été associés.

Ci-dessous ma solution d’authentification qui remplace le fichier précédent.



Cette modification bien que petite permet de **gagner plusieurs secondes** à chaque ouverture de sous-applications. Ce qui n’est pas négligeable puisque certaines sous-applications chargent tellement de données qu’on peut atteindre **quelques dizaines de secondes** de chargement.

La migration

La date

Le plus gros problème de la migration concernait la date. En effet, la plupart des sous-applications utilise la date pour faire marcher les fonctionnalités qu’elles proposent. L’impact de ce problème était suffisamment important pour **bloquer des fonctionnalités** de bien fonctionner.

Soit la date récupérée est celle du jour, donc obtenue par le biais de l’ordinateur du client ou du serveur. Soit elle est sélectionnée par le calendrier disponible sur la plupart des sous-applications.

Le calendrier en question :



Nous pouvons changer le jour en cliquant sur les nombres qui correspondent au jour voulu. Les flèches violettes permettent de changer le mois. Lors d’un changement de date par le biais du calendrier, la nouvelle date est simplement ajoutée dans l’url comme paramètre et la page est rechargée avec cette nouvelle information. Grâce à cette donnée, chaque page va effectuer différentes tâches notamment des requêtes SQL. Une **mauvaise date** peut soit avoir un **mauvais format**, pour exemple mois/jour/année. Soit avoir une valeur qui ne correspond pas à l’action faite pour l’obtenir. Si l’utilisateur clique sur le 1er juillet et la date récupérée correspond au 10 août, c’est problématique. Une mauvaise date peut entraîner un **comportement inattendu** du programme et afficher, modifier de **mauvaises données**.

Il y a eu **deux facteurs** responsables de ce problème de date.

* Le premier était un **problème de codage**. L’ancien développeur a programmé plusieurs fonctions qui formattaient la date pour obtenir un affichage bien spécifique ou un format différent. Par exemple pour l’afficher comme ceci : Heure : minutes jour/mois/année.

Cependant à travers ces différentes fonctions, la date s’est retrouvée **transformée par des fonctions intégrées** au langage, tel que month(Date) qui récupère le mois d’une date, et des **manipulations** par mon prédécesseur. Cette manière de procéder semblait retourner les résultats attendu sous Internet Explorer mais a entrainé bon nombre d’erreurs une fois l’application migrée sous Google Chrome. De plus, les manipulations étaient souvent inutiles et répétitives.

J’ai donc modifié ces fonctions en **supprimant les manipulations** pour ne laisser place qu’aux fonctions du langage prévues à cet effet.

* Une fois le premier facteur d’erreur éliminé, plusieurs **problèmes d’affichage** et de récupération de la date ont été réglés. Cependant plusieurs jours après, j’ai remarqué d’autres problèmes de date mais qui était relatés à la **récupération des données**. Je m’explique, l’application récupère des données de base de données pour la plupart. Cependant, les données récupérées ne correspondent pas à celles demandées. Pour exemple : une sous-application présente les incidents journaliers, en revanche les incidents pour le 10 juillet apparaissent comme ceux du 7 octobre.

La raison de ce problème est simple mais cela m’a pourtant pris plusieurs jours avant de mettre le doigt dessus.

Lors d’une requête SQL qui récupère des données, dans notre cas la commande **SELECT**, le logiciel qui gère les bases de données **va convertir la date** avant de procéder à l’exécution de la requête. Le logiciel en question est Microsoft Access, à chaque requête SQL si une date est incluse dans la commande WHERE, elle sera **convertie au format anglais** c’est-à-dire que le mois et le jour seront inversés. Il est donc nécessaire de prendre en compte cette information lors de l’écriture d’une requête SQL.

Pour récupérer les incidents du 10 juillet 2020, une requête ressemblera donc à ça :

“SELECT incidents FROM db\_incidents WHERE 07/10/2020”

En revanche, pour tout autre requête d’insertion ou de mise à jour des informations d’une base de données, il faut **garder le format voulu**. Access ne convertira pas les données à insérer.

Par la suite, l’application ne répertoriait plus aucuns nouveaux problèmes liés à la date. Il me fallut cependant appliquer cette solution partout.

Bilan et perspective

Synthèse