



Le génie pour l'industrie

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS GTI792 PROJET DE FIN D'ÉTUDES EN GÉNIE DES TI

**MISE EN PLACE D'UN ENVIRONNEMENT TECHNIQUE
PROJET ETS-PACIQ (APPLICATION WEB)**

RIAD CHEBLI
CHER11118502

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur

Alain April

MONTRÉAL, LE 7 AOUT 2014
ÉTÉ 2014



RIAD CHEBLI

REMERCIEMENTS

Mr Nicolas Brousseau et Mr Youssef Tariq, les précédents étudiants membres de ce projet qui ont répondu à mes nombreuses questions.

MIGRATION D'ENVIRONNEMENTS ET GESTION D'UNE WEB APPLICATION

RIAD CHEBLI
CHER11118502

RÉSUMÉ

Ce rapport décrit le travail fait dans le cadre d'une transition d'environnement d'une application de type web déployée sur un intranet privé et un domaine privé. L'application a été conçue pour répondre aux besoins de contrôle de processus du centre hospitalier universitaire Sainte-Justine de Montréal. Le premier cycle de développement de l'application étant terminée, le mandat est de prendre en main la transition de l'application depuis l'environnement de développement vers un environnement de test et/ou production. Ce document s'adresse et peut servir à n'importe quel développeur ou gestionnaire de la solution.

La solution dans son ensemble est développée et supportée par des technologies .NET, elle comprend l'application elle-même en page *asp.NET* qui utilisent une base de données Microsoft *SQL Server* pour la persistance des données, et un serveur web applicatif ; *Microsoft IIS*. Les étapes pour mener cette solution vers un environnement de test ont été les suivantes ;

1. Déployer le code sur le serveur web applicatif *IIS* du nouvel environnement sur le réseau interne de l'ÉTS
2. Configurer les accès à la base de données *SQL Server* sur le serveur du nouvel environnement
3. Créer les procédures de sauvegardes (*backups*) automatiques de la base de données *SQL Server*
4. Mettre en place un outil de contrôle de version du code
5. Gestion du domaine et des accès à ce dernier

Pour la plupart de ces étapes, la documentation utilisée aura été celle de *Microsoft*. De plus, plusieurs pratiques et techniques utilisées dans ce travail proviennent de ma mon expérience professionnelle et académique.

La solution est présentement déployée et fonctionnelle sur le réseau interne de l'ÉTS, elle est prête pour un deuxième cycle en plus d'avoir atteint un niveau de maturité supplémentaire lors de cette migration d'environnement avec l'ajout des sauvegardes automatisées et de l'outil de contrôle de version pour le code.

TABLE DES MATIÈRES

INTRODUCTION	1
CHAPITRE 1 MIGRATION ET DÉPLOIEMENT DU CODE DE L'APPLICATION	2
1.1 Description du travail effectué, de la problématique et des méthodes utilisées	2
1.2 Documentation utilisée, Justifications	3
1.3 Problèmes rencontrés / Leçons tirées / Recommandations	3
CHAPITRE 2 STRUCTURE DE LA BASE DE DONNÉES ET PROCÉDURE DE SAUVEGARDE AUTOMATISÉ	4
2.1 Description du travail effectué, de la problématique et des méthodes utilisées	4
2.1.1 Sauvegarde automatisée avec <i>SQL Agent</i>	5
2.2 Documentation utilisée, Justifications	6
2.3 Problèmes rencontrés / Leçons tirées / Recommandations	6
CHAPITRE 3 OUTIL DE CONTRÔLE DE VERSION DE CODE	8
3.1 Description du travail effectué, de la problématique et des méthodes utilisées	8
3.2 Problèmes rencontrés	9
CHAPITRE 4 GESTION DU DOMAINE ET DE L'ENVIRONNEMENT	10
4.1 Description du travail effectué, de la problématique et des méthodes utilisées	10
4.2 Leçons tirées / Recommandations	10
CONCLUSION	11
RECOMMANDATIONS	12
BIBLIOGRAPHIE	13

LISTE DES TABLEAUX

	Page
Tableau 1 - Problèmes/Solutions du code.....	2
Tableau 2 - Recommandations	12

LISTE DES FIGURES

Figure 1 - Étapes de la tâche de sauvegarde	5
Figure 2 - Tâche de sauvegarde programmée dans SQL Server Agent.....	6
Figure 3 - Capture d'écran d'Assembla	9

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

ÉTS : École de technologie Supérieure

CHU-STJ ; Centre Hospitalier Universitaire Sainte-Justine

SQL : Structured Query Language (aussi Simple Query Language)

INTRODUCTION

Afin de combler les besoins du centre hospitalier universitaire Sainte-Justine, en termes d'accréditation et de contrôle de qualité dans les processus, un projet recherche et développement avec l'ÉTS a été démarré, il s'agit d'une solution logicielle pour permettre à l'hôpital de mieux gérer et de quantifier les résultats des différentes initiatives d'amélioration de la qualité mises en place par le groupe qualité de l'hôpital. Le projet ayant terminé sa première itération de développement d'un prototype, la problématique principale est que tout l'environnement de ce prototype était localisé sur l'ordinateur personnel du responsable de l'infrastructure sortant de ce projet, et qu'il fallait maintenant déployer cette infrastructure sur un serveur interne du Dr. Alain April à l'ÉTS (c.-à-d. à l'aide de la technologie de machine virtuelle) afin d'assurer la pérennité du logiciel. Le prototype n'étant pas encore à sa version finale de production (c.-à-d. qu'il puisse être utilisé par le personnel du CHU), mais étant plutôt dans sa version de test fonctionnels, impliquant qu'un peu de développement pourrait être nécessaire, il faut donc s'assurer de maintenir une version fonctionnelle et disponible en tout temps.

Les principales composantes logicielles de la solution utilisent les composantes typiques d'une solution .NET de *Microsoft* :

1. Un serveur Web Applicatif (Microsoft *Internet Information Services*);
2. Un serveur de bases de données transactionnelles (Microsoft *SQL Server*);
3. Une application web (pages *ASP .NET*) ;
4. Un domaine Windows.

Un logiciel de cette envergure, composé de plusieurs entités fonctionnelles nécessite une transition contrôlée. Ce rapport présente les étapes réalisées et qui ont permises d'en arriver à une transition efficace du logiciel et de permettre quelques améliorations :

- I. Déployer le code sur le serveur web applicatif IIS du nouvel environnement (sur le réseau interne de l'ÉTS);
- II. Configurer des accès à la base de données SQL Server (sur le serveur à l'ÉTS);
- III. Créer des procédures de sauvegardes automatiques;
- IV. Mettre en place un outil de contrôle de version du code source, et
- V. Configurer la gestion du domaine et les accès à ce dernier.

CHAPITRE 1

MIGRATION ET DÉPLOIEMENT DU CODE DE L'APPLICATION

1.1 Description du travail effectué, de la problématique et des méthodes utilisées

Le code source du prototype actuel est sous forme de pages *asp.NET* et la plupart de la logique applicative est exécutée par des procédures en bases de données qui sont appelées par les pages *asp.NET*. Le problème principal de cette implantation était que le code devait passer d'un environnement de développement local à un environnement corporatif mis en place par les techniciens de l'ÉTS et Nicolas Brousseau. Avant le début de ce projet, une version de l'application était déployée sur le serveur *IIS* de l'environnement de l'ÉTS mais n'était pas fonctionnelle.

Dans un premier temps, une analyse de l'architecture logicielle de l'application a été nécessaire. Dans un second temps, la configuration du déploiement de l'application sur le serveur web *IIS* a été analysée. Les deux analyses, combinées à une expérience personnelle des déploiements sur les serveurs web applicatifs ont permis de rapidement de déceler deux problèmes, résumés au tableau 1. Ces activités d'investigation ont été les premières et ont permis la familiarisation avec le produit logiciel et de son environnement technique.

Problème	Cause	Solution
La page d'accueil du site n'est pas retrouvée	Le serveur de développement de Visual Studio reconnaît la page de démarrage du projet, IIS ne la reconnaît pas par défaut, et il essaie les pages les plus courantes, c.-à-d. <code>index.html</code>	Configurer manuellement la page de démarrage par défaut dans la configuration du site IIS
La connexion vers la base de données ne s'effectue pas	L'adresse du serveur de base de données dans le code pointait vers l'ancienne base de données sur l'ordinateur de développement du développeur de l'application.	Changer l'adresse de connexion pour faire référence au nouveau serveur de base de données de la machine serveur de l'ÉTS

Tableau 1 - Problèmes/Solutions du code

1.2 Documentation utilisée et justifications

Les bonnes pratiques en T.I. suggèrent de bien communiquer avec les gens responsables d'un logiciel, car ces gens peuvent transmettre leur connaissance du système ce qui est très utile quand une nouvelle personne essaie d'en assimiler la connaissance. Il faut noter que dans ce volet du travail, il y a eu très bonne communication avec l'ancien administrateur du serveur de développement et le développeur du code. Ce travail a été effectué à l'aide de bonnes pratiques de la maintenance en génie logiciel et une analyse d'impact.

1.3 Problèmes rencontrés, leçons apprises et recommandations

Le principal problème, dans cette situation, n'était pas de nature technique mais plutôt de nature organisationnelle. Le développeur, de son côté, ne connaît pas le serveur *IIS* et pour simuler l'exécution de son prototype, il n'a qu'à utiliser le bouton « *play* » de *Visual Studio*. De son côté, l'administrateur du serveur, ne fait que déployer ce code source sans le modifier. Le résultat de ce mode de fonctionnement isolé est que le code source n'est pas adapté pour un déploiement a été déployé sur le serveur *IIS*. Leçon apprise : Toujours avoir quelqu'un qui possède la vue d'ensemble de la solution technique. Pour gérer le déploiement du code, Microsoft offre un outil de déploiement qui s'intègre à son environnement de développement Visual Studio, cet outil, le *Web Deploy*, permet de déployer un logiciel à partir d'une machine de développement directement vers un serveur *IIS*. Cet outil permet aussi de préconfigurer le code source selon différents environnements, par exemple une « *Connection String* » qui change dépendant de l'environnement (c.-à-d. développement, test et production).

CHAPITRE 2

STRUCTURE DE LA BASE DE DONNÉES ET PROCÉDURE DE SAUVEGARDE AUTOMATISÉE

2.1 Description du travail effectué, de la problématique et des méthodes utilisées

L'application PACIQ utilise une base de données *SQL Server* pour enregistrer les données de l'application, et aussi pour permettre les fonctions d'intelligence d'affaire (*B.I – Business Intelligence*). Lors de l'examen de la configuration du serveur de base de données, deux problèmes majeurs ont été constatés : 1) la base de données applicative n'existe qu'en une seule version, ce qui veut dire que quand le développeur change son modèle de persistance au niveau du code applicatif, la personne responsable du volet *B.I* n'avait aucune idée de ces changements, ce qui occasionne forcément des problèmes aux procédures *ETL* (*Extract Transform & Load*). 2) il serait souhaitable que le développeur soit responsable de l'intégrité du code source avec la base de données qu'il utilise. Par exemple si le développeur ajoute un champ « Nom2 » à sa table « Rapports », les changements devraient être aussi effectués dans la base donnée, et cette manuellement.

Selon les bonnes pratiques du génie logiciel, pour palier à ces problèmes de développement logiciel, les données sont normalement déployées en trois copies : 1) développement; 2) test; et, 3) production. Cette duplication permet au développeur d'être libre dans la manipulation du modèle des données, sans se préoccuper de toujours maintenir une version fonctionnelle. Quand ces dernières sont prêtes pour les essais, les données et le modèle de données sont transférés vers la base de données de test, où on peut faire des tests sans impacter les autres, et finalement un dernier transfert vers la base de données de production est effectué où le modèle de données est fixé et les données enregistrées et utilisées sont des vraies données de production.

Une autre lacune observée sur serveur de base de données actuel est l'absence de procédures de sauvegarde automatique des données. En fait, il est important de faire des sauvegardes, premièrement en cas de corruption des données sur l'une des bases de données,

et deuxièmement pour permettre de rafraichir des données de test ou de développement avec les données de production lorsqu'on commence de nouvelles itérations de développement.

Le travail planifié et effectué dans ce volet du projet a été :

1. Les bases de données, ont été dupliquées, afin de permettre au développeur de continuer à développer pendant qu'une version fonctionnelle est maintenue pour permettre au client ainsi qu'au propriétaire de l'application de l'utiliser, à tout moment, pour faire des tests ou des démonstrations;
2. Une procédure de sauvegarde automatisée des données au moyen de SQL Server Agent a été mise en place et testée.

2.1.1 Sauvegarde automatisée avec l'aide de *SQL Agent*

SQL agent est un service, du serveur de base de données, qui permet d'exécuter des *scripts*, selon un horaire régulier, sur une base de données. Une tâche de sauvegarde a été créée et effectue une sauvegarde complète (non-incrémentale) des bases de données suivantes : « DW_PACIQ_TEST », « DW_PACIQ_TEST », « QP3_TEST » et « QP3_DEV ». La procédure est exécutée tous les dimanches soir à 18h00. S'il y a une erreur, celle-ci est inscrite dans le journal des évènements de Windows. Le dossier de sauvegarde est déplacé sur la racine du serveur sous le chemin : C:\PACIQ_DATA_BACKUP\PACIQ_FullBackup.bak plutôt que dans le dossier d'installation de *SQL Server* afin de permettre de le retrouver plus facilement. La tâche peut être retrouvée sous l'icône du *SQL Server Agent* comme on peut le voir a la figure 2.

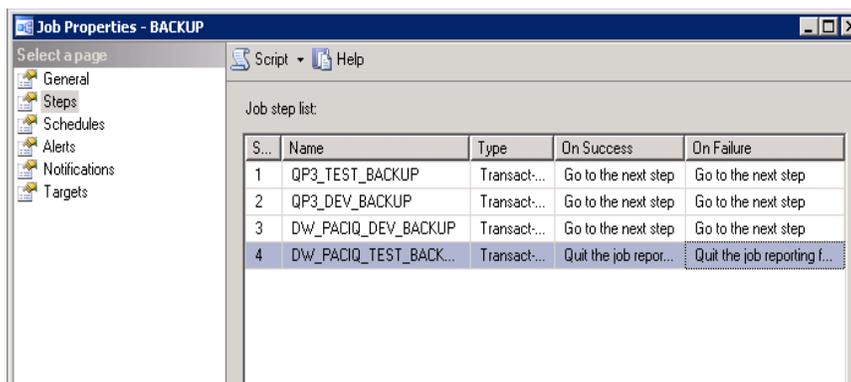


Figure 1 - Étapes de la tâche de sauvegarde

The screenshot shows the 'Job Activity Monitor' window for a SQL Server Agent. The main area displays a table of 'Agent Job Activity' with columns: Name, Enabled, Status, Last Run Outcome, Last Run, Next Run, Category, Runnable, Scheduled, and Category. The jobs listed are BACKUP, CDW_PACIQ_PACIQ_0, CDW_PACIQ_PACIQ_1, CDW_PACIQ_PACIQ..., ETL_PACIQ, and syspolicy_purge_history. The BACKUP job is 'Idle' and 'Succeeded'. The CDW_PACIQ_PACIQ_0, CDW_PACIQ_PACIQ_1, and CDW_PACIQ_PACIQ... jobs are 'Idle' and 'Failed'. The ETL_PACIQ job is 'Idle' and 'Unknown'. The syspolicy_purge_history job is 'Idle' and 'Succeeded'. The 'Last Run' column shows dates from 2014-08-10 to 2014-06-26. The 'Next Run' column shows dates from 2014-08-10 to 2014-08-11. The 'Scheduled' column shows 'yes' for all jobs. The 'Category' column shows '[Uncategorized...]' for all jobs. The 'Runnable' column shows 'yes' for all jobs. The 'Status' column shows 'Idle' for all jobs. The 'Last Run Outcome' column shows 'Succeeded' for BACKUP and syspolicy_purge_history, 'Failed' for CDW_PACIQ_PACIQ_0, CDW_PACIQ_PACIQ_1, and CDW_PACIQ_PACIQ..., and 'Unknown' for ETL_PACIQ. The 'Category' column shows '0' for all jobs.

Name	Enabled	Status	Last Run Outcome	Last Run	Next Run	Category	Runnable	Scheduled	Category...
BACKUP	yes	Idle	Succeeded	2014-08-10 12:...	2014-08-10 18:...	[Uncategorized...]	yes	yes	0
CDW_PACIQ_PACIQ_0	yes	Idle	Failed	2014-06-26 20:...	not scheduled	[Uncategorized...]	yes	no	0
CDW_PACIQ_PACIQ_1	yes	Idle	Failed	2014-06-26 20:...	not scheduled	[Uncategorized...]	yes	no	0
CDW_PACIQ_PACIQ...	yes	Idle	Failed	2014-06-26 20:...	not scheduled	[Uncategorized...]	yes	no	0
ETL_PACIQ	yes	Idle	Unknown	never	not scheduled	[Uncategorized...]	yes	yes	0
syspolicy_purge_history	yes	Idle	Succeeded	2014-08-10 02:...	2014-08-11 02:...	[Uncategorized...]	yes	yes	0

Figure 2 - Tâche de sauvegarde programmée dans SQL Server Agent

2.2 Documentation utilisée et justifications

La documentation *MSDN* sur le *SQL Server Agent* ainsi que l'expérience industrielle acquise, lors de stages en milieu industriel pour la configuration des bases de données et leurs duplications, ont été utilisés pour ce projet.

2.3 Problèmes rencontrés, leçons apprises et Recommandations

Compte tenu des besoins, non spécifiés, en sauvegarde de données en développement, la procédure de sauvegarde est maintenant adéquate. Toutes les bases de données sont maintenant sauvegardées, et ce de manière complète. Les sauvegardes de la base de données de développement sont optionnelles en ce sens qu'une sauvegarde la base de données de test pourrait être restaurée sur la base de données de développement en cas de besoin. Par exemple si un développeur ajoute une procédure qui accidentellement efface les données. Dans ce cas les données de test seront accessibles et représenteront les dernières données à jour. Dans un second temps, la fréquence et le moment des sauvegardes devront être éventuellement ajustées dépendamment de la fréquence de développement (c.-à-d. le nombre d'itérations de développement par année). Par exemple, s'il y a plus d'un développeur et que la fréquence du changement des données et/ou du modèle de donnée est plus importante, on pourra augmenter la fréquence des sauvegardes (c.-à-d. une fois par jour plutôt qu'une fois par semaine).

Une dernière recommandation, concernant les résultats des opérations de sauvegardes qui sont inscrites dans le journal des évènements Windows. L'éventuel administrateur du serveur pourrait trouver intéressant de recevoir un courriel si l'une de ces procédures échoue. Cette option est aussi offerte par le *SQL Server Agent* mais n'a pas été implantée.

Finalement, il faudrait sauvegarder, dans un autre emplacement que la machine serveur, puisque si celle-ci a une défaillance, les données seront toutes perdues. Présentement, la sauvegarde de l'image de la machine serveur est effectuée par les administrateurs du réseau de l'ÉTS, et permet quand même une certaine sauvegarde des données.

CHAPITRE 3

OUTIL DE CONTRÔLE DE VERSION DE CODE

3.1 Description du travail effectué, de la problématique et des méthodes utilisées

Durant ce projet, à plusieurs reprises, il est arrivé que le développeur change le code source de l'application afin de satisfaire aux exigences du client. Cette situation occasionnait les problèmes suivants :

1. Étant le seul à avoir la dernière version du code, il devient difficile pour l'administrateur du serveur de garantir que la dernière version du code est celle qui est déployée à tous;
2. Si plusieurs personnes développent sur la même application, il devient laborieux et couteux en temps de synchroniser les versions de code source de chacun, et difficile de retrouver la dernière version fonctionnelle du code source.

Lors de la première itération de développement, vu que la réalisation était faite à l'aide d'un seul développeur, ce n'était pas très important d'avoir la gestion de la configuration. La venue d'un deuxième développeur (c.-à-d. le développeur BI) et le fait que le travail effectué a nécessité de corriger des défauts en parallèle avec le développeur principal ont entraînés une perte de temps considérable causé par la recherche de qui avait la dernière version du code et laquelle était la plus fonctionnelle. Initialement, une solution temporaire proposée fût l'utilisation de *Google Drive* comme outil de dépôt du code. Cependant, de manière plus permanente il fallait instaurer, au minimum, un système de contrôle de version. De plus, afin d'installer un environnement de génie logiciel, il fallait aussi se préparer à l'installation d'un système de prise en charge des changements (c.-à-d. tickets). Fondé sur mon expérience la décision a été prise d'utiliser le logiciel en ligne *Assembla* (www.assembla.com) à cette fin.

Ce logiciel offre les fonctionnalités suivantes :

- Contrôle de version;
- Gestion des tickets;
- Disponibilité d'un web Service API;
- Un wiki;
- Gestion SCRUM et autres.

La disponibilité d'un web Service, dans le produit assembla, permet par exemple, de développer une page dans l'application web à partir de laquelle le client de l'application pourrait directement créer des tickets pour le signalement de nouvelles fonctions et de défauts. Pour l'instant, la version d'essai se limite à 2 utilisateurs et 50 mégaoctets de dépôt, cependant cela peut être mis à niveau (version payante) et permet jusqu'à 12 giga-octets de dépôt, ce qui pourrait s'avérer aussi une solution pour la sauvegarde des données de la base de données de production.

NAME	DATE	COMMIT MESSAGE	REVISION
branches	2014-05-14	[www-data] Automatically created readme.textile and /trunk, /branches, /tags directories. We recommend you to put all your code there.	1
tags	2014-05-14	[www-data] Automatically created readme.textile and /trunk, /branches, /tags directories. We recommend you to put all your code there.	1
trunk	2014-07-02	[Riad Chebli] Initial Import	2
readme.textile	2014-05-14	[www-data] Automatically created readme.textile and /trunk, /branches, /tags directories. We recommend you to put all your code there.	1

Assembla.readme
In assembla you can use `textile`, `html` or `markdown` to add some sugar to your readme files. Just add the extension to your readme file, so we can recognize the markup you want to use!

Figure 3 - Capture d'écran d'Assembla

3.2 Problèmes rencontrés

Toutes les fonctions du logiciel Assembla ne sont pas encore activement utilisées. Il faudrait planifier son utilisation et voir si il convient vraiment aux besoins de ce projet.

CHAPITRE 4

GESTION DU DOMAINE ET DE L'ENVIRONNEMENT

4.1 Description du travail effectué, de la problématique et des méthodes utilisées

Dans ce projet de recherche, le volet le plus ardu fût la gestion du domaine et de l'environnement. La situation était que d'un côté, il y avait le client et le propriétaire de l'application qui demandait l'accès aux fonctionnalités de l'application pour pouvoir y effectuer des tests et des démonstrations. De l'autre côté, nous avons un développeur applicatif et un développeur B.I qui faisait constamment des changements dans le logiciel. Dans cette situation, il était impossible pour l'administrateur de ce système d'en garantir l'accès. Ce problème provenait du fait qu'il n'y avait pas la disponibilité du nombre d'environnement de travail adéquat et distinct pour chaque intervenant. À un certain moment du projet, j'ai dû retirer les accès « *Remote Desktop Connection* » au développeur BI et au développeur logiciel puisque la version déployée n'était plus fonctionnelle. Ayant agi comme un administrateur responsable de mon système, cela a causé de la frustration du côté BI qui n'avait plus d'accès. La source de ces irritants et qu'en fait, selon les bonnes pratiques du génie logiciel, il devrait y avoir un environnement de développement où la partie BI pourrait aussi gérer son propre serveur. Finalement, un domaine intranet avait déjà été mis en place, les accès y ont été configurés par Nicolas Brousseau, j'ai restauré ces accès après le litige avec le développeur BI.

4.2 Leçons apprises et recommandations

Une bonne communication et gestion de projet est la principale leçon à tirer de ce volet moins technique du projet. D'un côté, il est à retenir qu'il vaut mieux sensibiliser les utilisateurs à l'importance de séparer les accès et de ne pas simplement fonctionner en mode « peer to peer » (c.-à-d. où chacun a des droits égaux sur le même serveur). D'un autre côté il faut bien connaître les besoins d'accès de chacune des parties prenantes avant de démarrer.

CONCLUSION

Ce mandat, sur ce projet de recherche et développement industriel, visait la prise en charge de la transition d'une application web d'un mode isolé à un mode partagé. Ce mandat a été réalisé et l'environnement de travail a pu atteindre une meilleure maturité et sera plus facile à gérer dans le cas où le projet prendrait de l'expansion (c.-à-d. plusieurs développeurs, une plus grosse équipe BI ou plus d'administrateurs). Les principales réalisations de ce travail sont les suivantes :

- Application déployée et fonctionnelle (paciq.etsmtl.ca);
- Outil de contrôle de version de code et de gestion de tickets au (<https://www.assembla.com/code/paciq/>);
- Mise en place des procédures de sauvegarde automatisées des Base de données;
- Sensibilisation de l'équipe au besoin d'un nouvel environnement.

Ce projet est très jeune, et ce travail, d'une durée de six mois à temps partiel, visait plus à faire fonctionner le logiciel sur le serveur de l'ÉTS et d'établir les bases sur lesquelles les futures itérations de développement, et de gestion de ce projet pourront s'appuyer. Un travail plus approfondi aurait pu être réalisé sur chacun des volets du projet cependant il a été décidé d'effectuer de petites itérations d'améliorations. Je considère y avoir donné le meilleur de ma connaissance des systèmes T.I. tels que vécus en entreprise et avoir apporté mon expertise technique en .NET.

RECOMMANDATIONS

Voici le résumé des recommandations qui suivent ce travail.

Volet	Recommandations	Suggestions
Code applicatif	<ul style="list-style-type: none">• Commencer l'utilisation d'assembla• Utiliser Web Deploy	<ul style="list-style-type: none">• Migrer vers une architecture MVC
Environnement	<ul style="list-style-type: none">• Déployer un deuxième environnement	<ul style="list-style-type: none">• Des machines virtuelles feront l'affaire (VM)
Accès au serveur	<ul style="list-style-type: none">• Limiter le plus possible les accès en « RDP »	
Base de données	<ul style="list-style-type: none">• Tester régulièrement les procédures de sauvegardes	<ul style="list-style-type: none">• Peut contenir 3 environnements sur un même serveur

Tableau 2 - Recommandations

BIBLIOGRAPHIE

Microsoft Developer Network: SQL Server agent

<http://msdn.microsoft.com/en-ca/library/ms189237.aspx>

Microsoft Developer Network: Web Deployment Overview for Visual Studio and ASP.NET

[http://msdn.microsoft.com/en-us/library/dd394698\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/dd394698(v=vs.110).aspx)

Microsoft ASP .NET; MVC 5 Framework

<http://www.asp.net/mvc/mvc5>

Assembla

<https://www.assembla.com/spaces/paciq/start>