Le génie pour l'industrie

RAPPORT TECHNIQUE
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS LOG792 PROJET DE FIN D'ÉTUDES EN GÉNIE LOGICIEL

SÉCURITÉ DU SYSTÈME BIMWEB
CONCEPTION D'UN SYSTÈME D'AUTORISATION D'APPLICATION WEB

SAMUEL MILETTE-LACOMBE
MILS26059100

DÉPARTEMENT DE GÉNIE LOGICIEL ET DES TI

Professeur-superviseur
Alain April

MONTRÉAL, 23 AVRIL 2015
HIVER 2015

Remerciements

Il est important de remercier certaines personnes qui ont contribué à ce que le projet soit mené à bien. Mathieu Dupuis, étudiant au doctorat en génie logiciel, a agi en tant que client et promoteur du projet en effectuant un transfert de connaissance pour permettre de faciliter la prise en main du projet de BIMWeb. Agissant en tant que conseiller, Mathieu s'est montré disponible pour des questions pendant la durée du projet et a permis même de contribuer à valider certaines solutions proposées. Il est nécessaire de souligner également la contribution de Christos Karras qui a agi en tant qu'expert des technologies .NET en validant les principales hypothèses de la conception et de l'implémentation proposée.

SÉCURITÉ DU SYSTÈME BIMWEB
CONCEPTION D'UN SYSTÈME D'AUTORISATION D'APPLICATION WEB

SAMUEL MILETTE-LACOMBE
MILS26059100

Résumé

Le rapport technique vise à analyser et élaborer des stratégies de sécurité pour l'application web BIMWeb. BIMWeb est une plateforme d'interopérabilité dans le domaine de la construction permettant aux divers intervenants dans un projet de construction de partager données et programmes afin de faciliter la construction de bâtiment. Des modules de tierce partie peuvent s'intégrer au système pour étendre les fonctionnalités des utilisateurs. Les données partagées doivent être protégées pour éviter qu'elle soit compromise par des utilisateurs malintentionnés ou des systèmes externes. Le mandat consiste d'abord à analyser les besoins en matière de sécurité. Ensuite, des stratégies comprenant principalement l'authentification et l'autorisation des utilisateurs ainsi que l'intégration des modules de tierces doivent être analysées. Étant donné que le système en est encore à un stade embryonnaire, des décisions architecturales et de conception concernant le système global doivent être prises. Ce rapport s'adresse à toute personne désirant contribuer au projet BIMWeb.

Bien que plusieurs solutions aient été analysées, une seule est vraiment applicable au contexte. Elle consiste en un mécanisme d'autorisation basé sur le contrôle d'accès RBAC (Role Based Access Control) selon le modèle décrit par le NIST, en utilisant la librairie ASP.NET Identity pour gérer l'appartenance des utilisateurs au site web. Le mécanisme RBAC développé se démarque du mécanisme de base offert dans ASP.NET en permettant de ne pas considérer seulement le rôle d'un utilisateur pour effectuer une autorisation, mais l'opération désirant être effectuée et l'entité du système concerné, offrant ainsi un grain plus fin.

Table des matières

1 Introduction 1
2 Analyse des besoins de sécurité du système de BIMWeb 2
2.1 Analyse des besoins de sécurité 2
2.1.1 Authentification des utilisateurs 2
2.1.2 Autorisation des utilisateurs 2
2.1.3 Intégration des modules de tierce partie 3
2.1.4 Contraintes 4
3 Tactiques architecturales couvertes et scénarios d'attribut de qualité 4
3.1 Scénarios d'attributs de qualité de sécurité 5
4 Bonnes pratiques de sécurité dans une application web 7
4.1 Nom d'utilisateur 7
4.2 Politique de complexité du mot de passe 7
4.2.1 Longueur 7
4.2.2 Complexité 7
4.3 Prévention des attaques par force brute 7
4.4 Prévention des attaques de type Cross-Site Request Forgery (CSRF) 8
4.5 Cryptage des sessions avec HTTPS 9
5 Conception du système d'autorisation 10
5.1 Méthodologie et contexte de la conception 10
5.2 Comparaison des modèles théoriques d'autorisation 10
5.2.1 Modèle DAC 10
5.2.2 Modèle MAC 10
5.2.3 Modèle RBAC 11
5.2.4 Modèle ABAC et CBAC 11
5.2.5 Sélection d'un modèle d'autorisation 12
5.3 Comparaison des types d'authentification dans ASP.NET 12
5.3.1 Authentification Windows 12
5.3.2 Authentification par formulaire 13
5.3.3 Sélection d'un type d'authentification 13
5.4 Comparaison des librairies de gestion de l'appartenance d'ASP.NET 13
5.4.1 ASP.NET Membership 13
5.4.2 Asp.Net Simple Membership 14
5.4.3 Asp.Net Universal Providers 14

5.4.4 Asp.Net Identity 15
5.4.5 Sélection d'un système de gestion de l'appartenance 15
5.5 Implantation de la librairie ASP.NET Identity 16
5.6 Conception du modèle conceptuel, relationnel et gestion de la persistance transparente 18
5.6.1 Modèle relationnel 18
5.6.2 Utilisation d'un « Enum Flag » pour identifier une opération autorisée par une permission 21
5.6.3 Vérification hiérarchique des permissions 21
5.6.4 Problème de l'incompatibilité objet-relationnel (*object-relational impedence mismatch)* 24
5.6.5 Persistance transparente avec Entity Framework 24
5.6.6 Comparaison des approches : Database First, Model First et Code First 25
5.6.7 Utilisation du système de migrations de la base de données 27
5.6.8 Patron Data Access Layer 27
5.7 Mécanisme de vérification des permissions 29
6 Validation avec un expert .NET 34
6.1 Mécanisme d'autorisation 34
6.2 Intégration de modules 34
7 Survol de l'intégration de module sécuritaire sous ASP.NET 35
7.1 Confinement avec AppDomain 35
7.2 Managed Add-ins framework (MAF) 35
7.3 Active Directory et autorisation 36
8 Conception du système d'attribution des permissions des utilisateurs 36
8.1 Décisions de conception 37
8.1.1 Gestion des permissions dans le contrôleur des rôles 37
8.1.2 Utilisation de HMTL Helper 37
8.2 Prototypes d'interface 37
9 Conception des tests de sécurité 39
9.1 Portée des tests 39
9.2 Méthodologie 39
9.3 Difficultés d'isolement des tests 40
10 Conclusion et discussion 40
11 Recommandations 42
11.1 Éviter de précipiter la sécurisation des systèmes 42

11.2 Respecter le principe du moindre privilège dans les attributions de permissions 42
12 Références 43
Annexe I - Document de vision BIMWeb 45
Annexe II - SRS de BIMWeb 46
Annexe III - Proposition de projet 47
Annexe IV - Analyse des besoins de sécurité 48
Annexe V – Version finale des jalons et des efforts 49

Liste des tableaux

Tableau 1 Entités nécessitant une autorisation (pris tel quel dans l'analyse des besoins en annexe) 2
Tableau 2 Opérations possible sur les entités 3
Tableau 3 Matrice opérations et entités 3
Tableau 4 Scénario de qualité S1 5
Tableau 5 Scénario de qualité S2 6
Tableau 6 Scénario de qualité S3 6
Tableau 7 Opérations améliorées sur les entités 22
Tableau 8 Matrice améliorée des opérations et entités 22

Liste des figures

Figure 1 Tactiques de sécurité couvertes (Bass, Clements, & Kazman, 2012, p. 154) 5
Figure 2 Représentation relationnelle du modèle RBAC selon (Post, 2010) 18
Figure 3 Schéma relationnel de la solution 19
Figure 4 Comparaison de l'approche des patrons Repository et Unit Of Work rapport à l'approche sans patrons (Dykstra, 2013) 29
Figure 5 Illustration du fonctionnement de la communication des Add-ins (Microsoft MSDN, 2015) 35
Figure 6 Prototype de l'interface d'édition et de création de rôles 37
Figure 7 - Prototype de l'interface de création et d'édition de permission 38

Liste des extraits de code

Extrait de code 1 configuration ASP.NET Identity 16
Extrait de code 2 « Enum Flags » pour identifier les opérations 21
Extrait de code 3 Exemple de test d'une opération 21
Extrait de code 4 Interface d'entité avec permission..... 23
Extrait de code 5 Exemple de migration..... 27
Extrait de code 6 Première méthode d'extension de vérification des permissions 30
Extrait de code 7 Deuxième méthode d'extension de vérification des permissions 31
Extrait de code 8 Troisième d'extension de vérification des permissions 31
Extrait de code 9 Exemple d'attribut Autorize sur un rôle..... 32
Extrait de code 10 Exemple d'attribut Autorize dysfonctionnel..... 32
Extrait de code 11 Attribut de filtre pour contrôleur 33
Extrait de code 12 Exemple d'utilisation d'attribut de filtre..... 33
Extrait de code 13 Exception soulevée lors de l'utilisation de Effort 39

Glossaire

Terme	Définition
ABAC	Attribute Based Access Control
AC	Access Control
Acteur	Utilisateur du système.
BIM	Building Information Modeling
CSRF	Cross-Site Request Forgery
CBAC	Claims Based Access Control
DAC	Discretionary Access Control
DAL	Data Access Layer
DBA	Database Administrator
GUID	Global Unique Identifier
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
LINQ	Language-Integrated Query
MAC	Mandatory Access Control
MAF	Managed Add-ins framework
MVC	Modèle-vue-contrôleur
NIST	National Institute of Standards and Technology
NOSQL	Not only SQL
ORM	Object-relational mapping
OWASP	Open Web Application Security Project
OWIN	Open Web Interface for .NET
URL	Uniform Resource Locator
RBAC	Role Based Access Control
SQL	Structured Query Language
SRS	Software Requirement Specification
XML	Extensible Markup Language

1 Introduction

La sécurité est un aspect très important à considérer lors de la conception d'une application web. Il est nécessaire d'être à l'affut des techniques visant à protéger les données d'un système, car les utilisateurs malintentionnés et les pirates informatiques renouvellent constamment leurs méthodes pour arriver à mener des attaques efficaces. La sécurité du logiciel est un domaine très vaste et couvrir l'ensemble de ses perspectives dépasse du cadre de ce projet de recherche. Cependant, dès que l'on songe à sécuriser une application web, il est impossible de négliger certains aspects fondamentaux de la sécurité comme l'authentification et l'autorisation des utilisateurs.

Ce projet de recherche a comme objectif l'analyse, la conception et l'implémentation de stratégies de sécurité pour la future plateforme BIMWeb. Les besoins du client, en termes de sécurité, pour cette nouvelle plateforme seront analysés. Les objets du système qui doivent être sécurisés seront décrits et des règles d'accès pour chacun des objets seront définies. Ces règles, permettront de définir comment les utilisateurs, de la future plateforme web, pourront, à travers des autorisations spécifiques, accéder et modifier des données. Ensuite, la conception et l'implémentation d'un mécanisme d'autorisation seront présentées ainsi que de la mise en place d'une plateforme permettant de gérer l'authentification et l'appartenance des utilisateurs à l'application web. Il sera question également de faire un survol des stratégies de sécurisation des modules de tierces parties que BIMWeb doit supporter. Pour arriver à atteindre ces objets, de bonnes pratiques de la sécurité informatique sur le web seront identifiées. Finalement, un expert des technologies .NET, validera l'approche choisie et donnera ses commentaires.

2 Analyse des besoins de sécurité du système de BIMWeb

2.1 Analyse des besoins de sécurité

Les besoins de sécurité, décrits à l'annexe IV, seront synthétisés ici. En somme, il s'agit de gérer l'authentification des utilisateurs et de leur autoriser l'accès à certaines ressources.

2.1.1 Authentification des utilisateurs

Le système sera composé d'utilisateurs de deux types:

- d'employés des compagnies inscrites dans le système
- de collaborateurs externes (p. ex. sous-traitant, contractuels) pour un modèle de projet donné.

Les collaborateurs externes peuvent être invités à participer à l'élaboration de certains modèles pour certains projets. Ces collaborateurs constituent la majorité des utilisateurs du système. Les utilisateurs seront de toutes provenances, et ils devront posséder un compte utilisateur enregistré au système BIMWeb. Leur courriel corporatif sera typiquement utilisé, dans la plupart des cas, pour les identifier. Aucune authentification externe, par exemple avec le protocole OpenId, n'est planifiée. Les utilisateurs accèderont, typiquement, au système BIMWeb à travers Internet sans avoir besoin d'être connectés à un réseau particulier.

2.1.2 Autorisation des utilisateurs

L'autorisation permet aux utilisateurs authentifiés d'accéder aux différentes ressources devant être protégées.
Ces entités sont détaillées dans le tableau suivant :

Tableau 1 Entités nécessitant une autorisation (pris tel quel dans l'analyse des besoins en annexe)

Nom	**Description**
Compagnie	Compagnie inscrite dans le système. Il s'agit du conteneur principal dans lequel on peut y ajouter des projets.
Projet	Projets de construction assignés à une compagnie. Le projet contient un ou plusieurs modèles BIM.
Modèle	Maquette 3D d'un bâtiment composée d'objets paramétrables et intelligents. Lors d'un téléversement d'un modèle, les données du fichier sont extraites et sont placées dans la base de données du système. Le fichier doit aussi être stocké sur le système.
Module	Un module de tierce partie pour étendre la fonctionnalité de base de BIMWeb

Trois des quatre entités ont une relation hiérarchique entre elles. Un modèle fait partie d'un projet et un projet fait partie d'une compagnie. Un module n'a aucune relation hiérarchique. Le système doit pouvoir attribuer des rôles pour pouvoir accéder à ces différentes entités.

Les opérations que l'on désire autoriser pour les entités sont les suivantes :

Tableau 2 Opérations possible sur les entités

Droit	**Lettre**	**Description**
Lecture	R	Accès en lecture
Mise à jour	U	Accès en écriture
Création	C	Peut créer l'entité
Suppression	D	Peut supprimer l'entité
Invitation	I	Peut inviter un utilisateur externe à la compagnie (seulement pour le modèle)

La matrice suivante permet de présenter les rôles possibles pour chaque paire d'entités.

Tableau 3 Matrice opérations et entités

		Compagnie				**Projet**				**Modèle**				
Entité	**Rôle**	**R**	**U**	**C**	**D**	**R**	**U**	**C**	**D**	**R**	**U**	**C**	**D**	**I**
Compagnie	User	X												
	Admin	X	X			X	X	X	X	X	X	X	X	
	AdminBIM	X				X	X	X	X	X	X	X	X	
Projet	Gestionnaire de projet Bim	X				X	X			X	X	X	X	X
Modèle	Lecteur	X				X				X				
	Éditeur	X				X				X	X	X		

Les entités évoquées jusqu'à maintenant ne sont pas les seules à devoir être protégées. La base de données et le système de fichier sont implicitement inclus dans les éléments à protéger. En revanche, les utilisateurs ne doivent pas interagir avec ceux-ci, mais plutôt avec l'application qui, de manière transparente, communiquera avec le système de fichier ou la base de données.

2.1.3 Intégration des modules de tierce partie

Un aspect important du projet BIMWeb, est d'étendre les fonctionnalités par l'intégration de modules de tierce partie. Il est impossible de savoir à l'avance les fonctionnalités qui seront développées. Par conséquent, il est nécessaire de prévoir l'accès à une base de données, soit celle du système central ou une autre base de données dédiée ou partagée, pour que les modules puissent stocker des données d'une manière personnalisée. Ces modules devront opérer de manière sécuritaire avec le système central.

2.1.4 Contraintes

Les contraintes sont essentiellement l'utilisation de « *ASP.NET MVC* » version 4 ou 5. L'éditeur de développement est « *Visual Studio Ultimate* 2012 ». Une base de données « SQL Server 2012 » devra être utilisée. L'édition précise de SQL Server n'a toutefois pas été précisée. L'édition utilisée dans le cadre du projet est la version 2012 « *Developper* ».

3 Tactiques architecturales couvertes et scénarios d'attribut de qualité

L'ouvrage *Software Architecture in Practice* (Bass, Clements, & Kazman, 2012) accordent une grande importance aux attributs de qualités et aux tactiques architecturales. Les attributs de qualité permettent de quantifier la rencontre des exigences du client et constituent un bon outil pour guider l'architecture et la conception.

L'attribut de qualité principal du présent projet est la sécurité. Voici une traduction libre de la définition de cet attribut de qualité selon les auteurs: la mesure des capacités d'un système à protéger les données d'accès non autorisées tout en donnant l'accès aux personnes et aux systèmes qui sont autorisés.
Les auteurs définissent six aspects de la sécurité à considérer, les définitions de ces aspects sont des traductions libres:

1. La *confidentialité* est la propriété de données ou de services qui sont protégés des accès non autorisés.
2. *L'intégrité* est la propriété de données ou de services qui ne sont pas sujets à des modifications non autorisées.
3. La *disponibilité* est la propriété d'un système à être disponible de manière suffisante pour une utilisation optimale. Le déni de service est un exemple d'attaque pouvant mettre à mal cette propriété.
4. *L'authentification* est la vérification de l'identité d'un acteur (un utilisateur ou un système) participant à une transaction et la vérification que cette identité est bien celle que l'acteur prétend incarner.
5. La *non-répudiation* est la garantie qu'un l'expéditeur d'un message ne peut plus tard nier d'avoir envoyé le message. C'est aussi la garantie que le destinataire du message ne peut nier avoir reçu le message.
6. *L'autorisation* est la propriété d'un système de permettre à un utilisateur d'utiliser un certain privilège (permission) pour effectuer une tâche spécifique.

Pour chaque attribut de qualité figure des tactiques architecturales permettant de contribuer à renforcer l'attribut. Ce rapport se limite à trois catégories de tactiques de la sécurité: la détection des attaques, la résistance aux attaques et la réaction aux attaques. Également, seules les tactiques de détection d'intrusion, d'authentification, d'identification, d'authentification et d'autorisation des acteurs, du cryptage des données ainsi que la révocation des accès seront abordées. Voici l'arbre récapitulatif des tactiques couvertes.

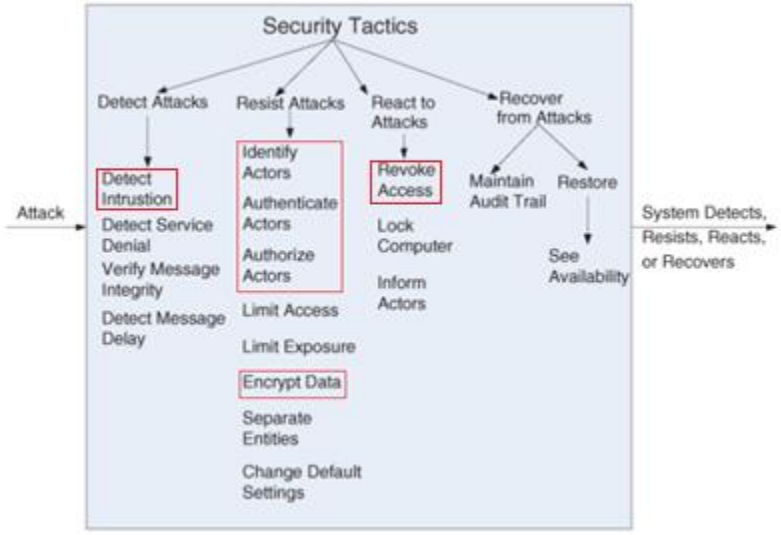

Figure 1 Tactiques de sécurité couvertes (Bass, Clements, & Kazman, 2012, p. 154)

3.1 Scénarios d'attributs de qualité de sécurité

Voici quelques scénarios d'attribut de qualité qui ont été élaborés pour guider la conception des stratégies de sécurité :

Tableau 4 Scénario de qualité S1

Source du stimulus	Utilisateur non authentifié
Stimulus	Désire s'authentifier en soumettant le formulaire de connexion avec des informations de compte valides
Environnement	Non connecté à un réseau spécifique Réseau ouvert
Artéfact	Système d'authentification
Réponse	Le système authentifie l'utilisateur
Mesure de la réponse	L'utilisateur est authentifié ou non

Tableau 5 Scénario de qualité S2

Source du stimulus	Utilisateur authentifié
Stimulus	Désire effectuer une action non autorisée
Environnement	Non connecté à un réseau spécifique Réseau ouvert
Artéfact	Données dans le système
Réponse	Le système bloque la requête de l'utilisateur et le redirige vers une page d'accès non autorisée
Mesure de la réponse	La requête est rejetée ou est acceptée

Tableau 6 Scénario de qualité S3

Source du stimulus	Utilisateur non authentifié
Stimulus	Cinquième tentative de connexion échouée
Environnement	Non connecté à un réseau spécifique Réseau ouvert
Artéfact	Système d'authentification
Réponse	Verrou du compte utilisateur
Mesure de la réponse	L'utilisateur ne peut plus se connecter avec son compte pendant 20 min même si celui-ci réussit à entrer le bon mot de passe.

4 Bonnes pratiques de sécurité dans une application web

Les sections qui suivent décrivent un ensemble non exhaustif de pratiques recommandées par l'OWASP (Woschek, 2015), pour élaborer dans un premier temps, la politique décrivant la composition des noms d'utilisateur et des mots de passe, et dans un second temps, les tactiques visant à se prémunir de certaines attaques.

4.1 Nom d'utilisateur

Les acteurs sont identifiés avec un nom d'utilisateur. Voici quelques caractéristiques qu'un nom d'utilisateur doit avoir :

- Être unique
- Être insensible à la casse (ex. : « Smith » est équivalent à « smith »)
- Peut être une adresse courriel. L'adresse courriel est par défaut unique et est facile à retenir.

4.2 Politique de complexité du mot de passe

Il est important de définir une politique claire pour la longueur minimum et maximum des mots de passe. Il est aussi primordial de s'assurer d'une complexité minimale du mot de passe. Le système doit bloquer tout changement ou définition de mot de passe ne satisfaisant pas les critères de cette politique. Une bonne politique permet de rendre quasi impossible qu'un pirate devine les mots de passe des usagers.

4.2.1 Longueur

Il est recommandé d'avoir un minimum de 10 caractères, en deçà duquel le degré de sécurité devient trop faible (Woschek, 2015, p. 13). La longueur maximum devrait être limitée à 128 caractères pour encourager l'utilisateur à choisir un mot de passe complexe composée d'une phrase, ce qui est plus facile à retenir.

4.2.2 Complexité

Un mot de passe devrait avoir un degré de complexité minimal pour éviter qu'il soit trop facile à deviner. Voici les caractéristiques de complexité minimale recommandée :

Au moins 3 de ces 4 règles

- Au moins 1 caractère en majuscule (A-Z)
- Au moins 1 caractère en minuscule (a-z)
- Au moins 1 chiffre (0-9)
- Au moins 1 caractère spécial (l'espace doit compter comme un caractère spécial)

Il ne faut également pas avoir plus de deux caractères identiques qui se suivent.
Lorsque l'utilisateur a à définir ou à changer son mot de passe, il est recommandé de lui donner une indication de la sécurité actuelle du mot de passe entré.

4.3 Prévention des attaques par force brute

Une attaque par force brute, dans le contexte d'authentification, se décrit par le fait d'essayer une à une toutes les combinaisons possibles de mots de passe afin de tenter de se connecter frauduleusement à un compte utilisateur inscrit dans le système. Un pirate peut aussi tenter, de manière manuelle, d'essayer plusieurs mots de passe, mais cette technique n'est pas très efficace en raison du temps nécessaire pour faire chaque

tentative. Les pirates utilisent donc habituellement des techniques visant à automatiser l'essai de mots de passe en utilisant par exemple un dictionnaire (c.-à-d. attaque par dictionnaire) et un script pour tenter de s'authentifier avec chaque possibilité de mot de passe puisé dans ce dictionnaire. Pour se prémunir de ces attaques, il est nécessaire de limiter le nombre d'authentifications échouées consécutives. Une première solution consiste à bloquer tout compte utilisateur si plus de « n » tentatives de connexions consécutives ont échouées. Les comptes pourraient alors être débloqués, par l'administrateur du système, sur demande. Se contenter de cette protection a par contre pour effet de permettre une autre vulnérabilité, permettant un autre type d'attaque, qui consiste à faire verrouiller un certain nombre de comptes en faisant échouer des tentatives d'authentification sur un grand nombre d'utilisateurs. Une solution améliorée, consiste à toujours bloquer les comptes dans les mêmes circonstances, mais de les débloquer automatiquement après un certain délai. Grâce à cette solution, l'efficacité des attaques, par force brute, est drastiquement réduite. L'Open Web Application Security Project (OWASP) ne recommande pas de nombre précis de tentatives consécutives échouées, ni de délai précis pour le déblocage. Le nombre de tentatives échouées avant blocage du compte pourrait être fixé à cinq. Le temps nécessaire pour un déblocage automatique pourrait être par exemple, de vingt minutes.

4.4 Prévention des attaques de type Cross-Site Request Forgery (CSRF)

L'attaque CSRF permet à un utilisateur malintentionné, ne disposant pas des droits requis pour une action, de contourner le mécanisme d'autorisation en faisant exécuter à sa place, l'action désirée (p. ex. supprimer un élément) par un utilisateur disposant des droits nécessaires. Un cas typique est que l'attaquant connait le lien (c.-à-d. l'URL) pour effectuer une action en particulier, mais que celui-ci ne dispose pas des privilèges requis pour faire cette action. L'attaquant envoie alors un message à une victime disposant des privilèges requis, à l'aide d'un faux fichier image contenant un script permettant d'aller à l'URL de l'action. La victime ouvre ce message et, se faisant, son navigateur tente de récupérer le contenu de l'image et exécute l'URL. Le résultat est l'exécution de l'action malveillante. Le navigateur ne reconnait pas le fichier d'image et n'affiche donc rien de spécifique à l'écran. La victime est donc complice de l'attaquant sans le savoir.

La pratique recommandée par l'OWASP, pour éviter ces situations, est l'utilisation d'un jeton de validité inséré dans les formulaires. Cette approche se nomme communément « *Synchronizer Token Pattern* ». Ce jeton permet qu'un formulaire soumis (avec une action « POST ») ne soit autorisé que s'il a été produit quelques minutes auparavant, avec l'aval du serveur, en autorisant l'affichage du formulaire en question lors de la requête (avec une action « GET »).

L'implémentation de la validation de ce jeton est relativement facile en « ASP.NET MVC ». Cela consiste essentiellement à insérer, dans la section formulaire de chaque vue MVC, l'instruction « `@Html.AntiForgeryToken()` » et l'attribut « `[ValidateAntiForgeryToken]` » sur chaque méthode « POST » des contrôleurs.

4.5 Cryptage des sessions avec HTTPS

Le cryptage des sessions est recommandé par OWASP non seulement pour le processus d'authentification, mais aussi pour l'application web au complet. Cela permet de crypter l'ensemble du trafic généré par les sessions des utilisateurs avec l'objectif d'éviter que des attaquants puissent intercepter les données non chiffrées échangées et les utiliser à mauvais escient, par exemple en utilisant frauduleusement l'information confidentielle dans le but de nuire aux compagnies inscrites sur le système BIMWeb.

5 Conception du système d'autorisation

5.1 Méthodologie et contexte de la conception

La conception de la sécurité du système BIMWeb est réalisée dans un contexte initial où aucune décision architecturale et conceptuelle n'a réellement été prise et documentée. En effet, les analyses technologiques n'ont pas encore été réalisées ni documentées. Bien que les documents de vision et le SRS, présentés en annexe, aient pu être utilisés comme base pour obtenir une vue d'ensemble du projet, il reste bon nombre de questionnements à élucider. Par exemple, comment la gestion de la « scalabilité » de la base de données, nécessaire pour satisfaire les exigences de performances, qui n'ont pas encore été précisées, sera-t-elle menée ? De plus, quelle approche choisir par rapport à la gestion de la persistance des données de la base de données ? Ce sont toutes des questions qui auraient dû être précisées avant de se concentrer véritablement sur l'aspect sécurité d'une manière détaillée.

5.2 Comparaison des modèles théoriques d'autorisation

L'analyse des besoins, évoquée précédemment, a permis quand même de définir une politique d'accès en statuant que la notion de rôles serait utilisée pour décider quelles données un acteur a le droit d'accéder ou de modifier. Parmi les patrons de contrôle d'accès existants, il existe le «Role Based Access Control» (RBAC), c'est-à-dire le contrôle d'accès à base de rôles. Il serait tentant d'aller vers cette voie puisque la notion de rôle est au cœur de cette politique d'accès. Cependant, il existe d'autres patrons de contrôle d'accès possibles. Il est donc pertinent d'explorer l'utilité de ces autres patrons afin d'être certain d'avoir un portrait global des avenues possibles. Ceci permet à l'ingénieur de prendre une décision éclairée. Les sections suivantes seront consacrées à décrire les principaux patrons de contrôle d'accès.

5.2.1 Modèle DAC

Le modèle «Discretionary Access control» (DAC) base ses autorisations sur un utilisateur et/ou sur les groupes attitrés à un utilisateur. Le DAC est dit discrétionnaire puisqu'un utilisateur disposant de droits sur une donnée peut donner des droits à un autre utilisateur sur ces mêmes données (Jordan, 1987, p. 1). Puisque chacun des utilisateurs peut donner à sa guise à d'autres utilisateurs, des accès sur les objets dont il est autorisé, il est difficile, voire impossible, de centraliser la gestion des autorisations par un administrateur gérant une politique d'accès applicable à tous les utilisateurs.

5.2.2 Modèle MAC

Un autre modèle se nomme le Mandatory Access Control (MAC). Ce type de contrôle d'accès est en opposition au DAC puisque son principe fondamental est de baser les autorisations sur une politique de sécurité globale et totalement centralisée interdisant donc aux utilisateurs de décider eux-mêmes des droits d'accès sur les données dont il dispose d'autorisations. Dans la pratique, ce contrôle d'accès est typiquement utilisé pour les systèmes à exigences de sécurité très élevée comme les applications militaires et

autres applications contenant des données extrêmement sensibles. Un exemple d'utilisation dans ce contexte consiste à assigner un niveau de criticité à chaque type d'information, et un niveau de criticité à chaque utilisateur du système. Un accès à une information est alors permis pour un utilisateur seulement si celui-ci a un niveau de criticité suffisant pour l'information en question.

5.2.3 Modèle RBAC

Ce type de contrôle permet d'autoriser une action selon le rôle de l'utilisateur. Un rôle est différent d'un groupe décrit dans le DAC (NIST, 2014). Un groupe est utilisé dans un contexte où les groupes et les utilisateurs peuvent accumuler tous les deux des permissions. Dans un contexte de rôles comme dans le RBAC, il est impossible de donner des permissions à un utilisateur sans que cette attribution soit faite à travers l'attribution de rôles. Chaque rôle est relié à des permissions permettant des opérations (par ex. : lecture, écriture, modification, etc.) sur les objets du système. Un accès est autorisé pour un utilisateur seulement si cet utilisateur a un rôle relié aux permissions requises pour faire l'action demandée. Le RBAC offre la flexibilité d'implémenter des notions du DAC et du MAC. Par exemple, il peut y avoir une attribution de permissions de manière discrétionnaire en autorisant les rôles hiérarchiques où un gestionnaire de projet pourrait attribuer des droits sur des objets du système à d'autres utilisateurs. Également, il est possible de mettre en place des caractéristiques du MAC en ayant certains principes de la politique d'accès qui soient appliqués à tous les utilisateurs du système. RBAC permet d'implémenter des contraintes complexes comme la séparation des tâches ou mieux connues en anglais sous le nom de « *Separation of duties* ». La séparation des tâches permet, entre autres, de réduire l'exposition à la fraude ou au conflit d'intérêts dans une organisation (SANS Technology Institute) en vérifiant par exemple qu'un utilisateur ne peut détenir deux rôles conflictuels tels que le rôle d'acheteur et le rôle d'approbateur d'achat. Le RBAC comporte un mécanisme de session permettant de résoudre ce problème en gérant l'activation des rôles des utilisateurs selon une période de temps définis.

5.2.4 Modèle ABAC et CBAC

L'Attribute Based Access Control (ABAC) est un contrôle d'accès récent. L'ABAC considère les entités du système à titre d'utilisateur et les ressources, mais aussi tout autre attribut. Un attribut peut être n'importe quelle information pertinente comme la taille d'une ressource, un type d'opération à effectuer, la date de création, le propriétaire de la ressource, l'heure actuelle, etc. L'ABAC permet un contrôle plus granulaire des autorisations que toutes les autres stratégies présentées jusqu'à maintenant dans ce rapport.

Il est intéressant de noter que le terme ABAC est régi par le National Institute of Standards and Technology (NIST) et que l'entreprise Microsoft utilise aussi le terme CBAC (Claim Based Access Control) pour désigner la « version Microsoft » du contrôle d'accès par attribut. Au lieu d'attribut, Microsoft se contente de parler de « *claims* », qui au bout du compte, représentent le même concept. La différence entre les deux contrôles d'accès est trop faible pour qu'elle soit décrite dans le cadre de ce projet.

5.2.5 Sélection d'un modèle d'autorisation

Le DAC semble, à première vue, bien adapté aux besoins du système BIMWeb puisque les besoins spécifient le désir d'avoir une hiérarchie dans les responsabilités des utilisateurs où un contrôle discrétionnaire doit être autorisé en permettant notamment aux gestionnaires de projet BIM de donner le droit à des collaborateurs externes de modifier les données d'un modèle. Cependant, le DAC permet de supporter l'attribution de permissions directement à des utilisateurs sans passer par des rôles. Or les besoins précisent que chaque utilisateur devrait avoir un rôle bien spécifique. Utiliser seulement le DAC comme seul contrôle est donc inapproprié.

Choisir le MAC comme seul contrôle d'accès ne semble pas plus approprié à la situation puisqu'il est trop rigide et tel qu'expliqué précédemment, les notions du DAC (c.-à-d. son contrôle discrétionnaire) devront être utilisées.

L'ABAC, de son côté, permet un contrôle très fin des accès. Cependant, le degré de complexité de la politique d'accès définie dans l'analyse des besoins n'est pas très élevé. De plus, les autorisations dans BIMWeb ne dépendent pas d'attributs autres que l'utilisateur, l'entité, et l'opération à effectuer sur l'entité. Il s'avère que l'ABAC est justement conçu pour faire face à des autorisations dépendantes d'attributs multiples tels que l'heure de la journée à laquelle une demande d'accès est faite ou les relations entre les utilisateurs.

Le modèle RBAC est donc le type de contrôle d'accès restant à étudier pour notre cas d'étude. Il s'avère que c'est celui qui s'applique le mieux aux besoins présentés puisqu'il est plus flexible que le DAC et le MAC individuellement. Il permet d'intégrer des notions de contrôle d'accès discrétionnaire et obligatoire. Il permet également de représenter les rôles et leurs permissions pour chacun des différents objets du système.

5.3 Comparaison des types d'authentification dans ASP.NET

Sous « ASP.NET », il y a deux méthodes d'authentification supportées nativement par le cadriciel MVC (Chadwick, Snyder, & Panda, 2012, pp. 178-191). Les sections suivantes seront consacrées à les décrire.

5.3.1 Authentification Windows

L'authentification Windows est utilisée pour sécuriser une application utilisée dans un même domaine (Bass, Clements, & Kazman, 2012, p. 183). Cela s'explique par le fait que l'authentification est réalisée à partir du compte utilisateur, sur la machine locale, utilisée pour accéder à l'application web. Il faut alors que ce compte soit reconnu par le serveur de l'application web. Il est alors impératif que l'ordinateur de l'utilisateur soit connecté au même domaine que celui du serveur. En raison de l'utilisation d'un compte de ce type, le processus d'authentification est transparent puisqu'il ne s'agit que de valider une deuxième fois l'identité du compte et aucune reconnexion de la part de l'utilisateur n'est alors nécessaire. Cette technique utilise l'envoi d'un jeton de sécurité

Windows pour chaque requête HTTP envoyée. Le *jeton* est ensuite corroboré avec les informations du compte utilisateur approprié définies dans le domaine du serveur.

5.3.2 Authentification par formulaire

L'authentification par formulaire est utilisée particulièrement pour les sites publics. Cette méthode consiste en une authentification à travers un formulaire pour saisir compte utilisateur et mot de passe pour ensuite valider les données auprès d'un gestionnaire de l'appartenance qui effectue ou non la connexion de l'utilisateur. Un gestionnaire d'appartenance consiste en une liste d'éléments logiciels qui peuvent être intégrés à l'application web pour gérer, de manière simplifiée, l'appartenance d'utilisateur au site web. L'appartenance à un site web inclut des opérations telles que la création et la suppression de comptes utilisateur et la gestion des mots de passe. Il existe différents fournisseurs pouvant être utilisés ayant chacun leur particularité et leur domaine d'application. Les principaux fournisseurs sont listés dans la section suivante.

5.3.3 Sélection d'un type d'authentification

Étant donné que le système n'est pas destiné à être utilisé sur un intranet défini par un domaine restreint et commun à tous les utilisateurs, il faut éliminer la stratégie d'authentification Windows qui demande à ce que les utilisateurs fassent partie d'un même domaine. La stratégie retenue est donc l'authentification par formulaire.

5.4 Comparaison des librairies de gestion de l'appartenance d'ASP.NET

Une librairie de gestion de l'appartenance permet de contrôler l'adhésion des utilisateurs à l'application web. Elle gère, en premier lieu, les informations sur l'adhésion des utilisateurs. Ces informations sont typiquement les propriétés de l'utilisateur (p.ex. son nom d'utilisateur, son mot de passe, son courriel), des attributs visant à augmenter la sécurité de l'application (p.ex. nombre de connexions échouées depuis la dernière connexion, date du dernier changement de mot de passe, l'état du verrouillage du compte utilisateur, les rôles des utilisateurs). Ensuite, la librairie offre un moyen de faire persister l'information sur un support externe comme une base de données. La librairie vient également avec du code source exemplaire que l'on peut intégrer, tel quel, dans l'application afin de bénéficier des fonctionnalités fournies. Il s'agit typiquement de vues, de modèles et de contrôleurs et de certains de fichiers de configuration. Quand tout est mis en place, il est alors possible de gérer l'inscription des utilisateurs à l'application web, le blocage de comptes basé sur des règles de sécurité, et effectuer une authentification par un fournisseur externe.

Étant donné qu'il existe plusieurs librairies de gestion de l'appartenance, il est alors nécessaire, pour l'ingénieur, de comparer les différents systèmes afin de faire un choix éclairé.

5.4.1 ASP.NET Membership

Cette librairie, parue en 2005 (Rastogi, Anderson, Dykstra, & Galloway, 2013), répond aux besoins de sécurité initiaux d'une application web. Elle permet notamment de gérer

l'authentification par formulaire et l'utilisation de « SQL Server » pour la persistance. Des fonctions utiles de sécurité sont incluses dans cette librairie. Il s'agit, entre autres, du blocage automatique de compte lorsqu'une certaine limite de connexions échouées est atteinte ainsi que la vérification de la complexité des mots de passe. Cependant, cette librairie comporte certaines lacunes.

- SQL Server est la seule base de données pouvant être utilisée par cette librairie. Elle inclue d'ailleurs des procédures stockées et des vues. Également, des agents sont utilisés, du côté de la base donnée, rendant le couplage avec le type de base de données très important.
- Il est difficile de personnaliser le schéma de la base de données pour y ajouter des champs dans les mêmes tables que celles utilisées par la librairie.
- Il est impossible d'utiliser les fournisseurs de connexions externes, utilisant l'interface ouverte OWIN, utilisés communément par des services tels que : Facebook, Google et Twitter.

Il est à noter qu'il existe plusieurs « *Membership provider* » héritant de la classe de base permettant d'offrir une liaison avec différents services, tels que « Active Directory » et « Open ID ».

5.4.2 Asp.Net Simple Membership

Cette librairie a été créée afin de simplifier l'ajout de la gestion de l'appartenance (c.-à-d. le *Membership*) à une application web. Des personnalisations des informations des tables utilisées par la librairie peuvent être faites plus aisément. Par exemple, il n'est pas nécessaire de devoir ajouter une deuxième table « UserInfo » pour étendre les champs des utilisateurs comme c'est le cas dans *ASP.NET Membership*. Cette librairie a l'avantage de supporter plus de types de bases de données. Elle permet l'utilisation du service de base de données relationnelle basée sur le nuage, ce qui inclut le support de S*QL Azure*, *SQL Server CE*, *SQL Server Express* ainsi que *LocalDB*. Également, la librairie effectue seulement des appels SQL et donc n'est pas dépendantes de procédures stockées ou autres objets du côté de la base de données. Cette librairie comporte tout de même certaines lacunes.

- Seulement les bases de données de famille « SQL Server sont supportées et donc les bases de données non relationnelles (NO SQL) ne peuvent être utilisées.
- Encore une fois, l'utilisation de fournisseurs d'authentification externes avec OWIN n'est pas disponible.
- Les fonctionnalités de sécurité intégrées comme le blocage des comptes utilisateurs ne sont plus présentes dans ASP.NET Membership.

5.4.3 Asp.Net Universal Providers

Bien que le nom exclue le terme « Membership », il n'en est pas moins que cette librairie a été bâtie sur l'architecture de « ASP.NET Membership ». Par conséquent, il y a les mêmes limitations. Il semblerait cependant que tout type de base de données supporté par « Entity Framework » soit supporté étant donné que la librairie a été programmée en utilisant ce cadriciel et l'approche « Code First » (Rastogi, Anderson, Dykstra, & Galloway, 2013). La librairie permet notamment le support de « *SQL Server Compact* ».

5.4.4 Asp.Net Identity

Cette librairie relativement récente est une refonte complète de la gestion d'appartenance dans « ASP.NET ». Elle offre de nombreux avantages par rapport aux précédentes librairies :

- Intégration du service d'intergiciel OWIN
- Facilité de modification pour supporter des types de bases de données non relationnels.
- Un ajout de fonctions de sécurité telles que la vérification de la complexité du mot de passe, le blocage de compte et le déblocage de compte après un certain temps.
- La possibilité de valider l'identité d'un utilisateur par de multiples facteurs tels que l'envoi de SMS à un numéro de téléphone et l'envoi de courriel.
- Elle intègre la possibilité la possibilité d'effectuer un contrôle d'accès par attribut (ABAC) en intégrant la notion de « *Claim* ».
- Elle représente la librairie la plus à jour et la plus susceptible d'être maintenue le plus longtemps par Microsoft.

5.4.5 Sélection d'un système de gestion de l'appartenance

Il apparait être un meilleur choix d'utiliser « *ASP.NET Identity* ».

Premièrement, les bonnes pratiques de sécurité recommandent notamment le contrôle de la complexité des mots de passe et le blocage des comptes selon le nombre de tentatives de connexions échouées. Ces fonctions de sécurité sont présentes et fonctionnelles que sur deux des librairies évoquées : « ASP.NET Identity » et « ASP.NET Membership ». « *ASP.NET Membership* » n'offre cependant pas de fonctionnalité, par défaut, permettant de débloquer automatiquement un compte après un certain temps. Bien qu'il soit possible de développer soi-même les fonctionnalités manquantes, il semble que ce soit inadéquat, car cela ajoute plus de code à maintenir et considérant qu'il s'agit d'une fonctionnalité générique à toutes les applications, il semble plus approprié d'utiliser une librairie existante, maintenue par le fournisseur, pour ne pas avoir à le faire soi-même et la maintenir par la suite.

Deuxièmement, il s'agit de la librairie la plus extensible, notamment avec le fait qu'elle n'a pas de restriction sur le type de base de données. Moyennant l'ajout d'une couche d'accès aux données (*Data Access Layer*) appropriée au type de base de données désiré, gérant les rôles et les utilisateurs prenant la forme de « RoleStore » et de « UserStore » personnalisé, il est possible de gérer de nombreux autres types de bases de données provenant d'autres fournisseurs. Bien qu'il soit présentement planifié d'utiliser seulement « SQL Server », rien ne dit que plus tard, ce requis ne changera pas lors de constations du non-respect des objectifs qui pourraient être établis par rapport aux attributs de qualité tels que la performance.

En revanche, aucune des librairies évoquées n'offre une implémentation pour gérer l'autorisation d'accès par entité pour les utilisateurs. Donc même en sélectionnant « ASP.NET Identity » comme solution potentielle, il faudra tout de même réaliser un modèle de données et son code source pour faire la vérification des permissions.

5.5 Implantation de la librairie ASP.NET Identity

L'implantation « d'ASP.NET Identity » s'est faite en téléchargeant, à partir des paquets « Nuget », le projet d'exemple regroupant tous les éléments nécessaires. Le projet de base contient certaines particularités. Les identifiants (ID) des entités des objets de base sont des « Globally Unique Identifier » (GUID). Un GUID est un identificateur de type texte comportant 16 octets. Il contient des chiffres, des lettres et des tirets. Le nombre élevé de possibilités de GUID différents rend théoriquement impossible d'avoir deux identifiants identiques. Par contre, dans le présent projet, bien qu'il est planifié d'avoir un très grand nombre d'enregistrements par tables, rien ne porte à croire que l'attribut d'identifiant (ID) comportant une valeur numérique entière soit inapproprié. Le stockage de 16 octets, pour chaque enregistrement, est élevé par rapport à un stockage d'un type INT (4 octets) ou BIGINT (8 octets) en SQL Server. Un GUID étant plus complexe qu'un nombre entier, sa gestion est plus difficile. GUID est typiquement utilisé lorsque plusieurs systèmes partagent des ID où les conflits entre les mêmes identifiants doivent être évités. Il est donc normal que « ASP.NET Identity » offre cette implémentation de base parce que cette libraire est orientée authentification externe avec notamment le support d'intergiciel « Owin ». Or il se trouve qu'aucune authentification externe n'est requise pour le projet. Le code d'« ASP.NET Identity » a donc été modifié pour gérer des « ID » numériques classiques.

Il a été mentionné précédemment qu'un des avantages de « ASP.NET Identity » est qu'il fournit une implémentation fonctionnelle de plusieurs fonctions de sécurité. Un fichier de code, nommé « IdentityConfig.cs », qui se trouve dans le dossier « App_Start » de l'application, permet de configurer les différentes fonctions de sécurité. Voici un extrait du fichier mettant en évidence les aspects importants :

```
            // Configure validation logic for passwords
            manager.PasswordValidator = new PasswordValidator
            {
                // Taille minimum d'un mot de passe de 10 caractères
                RequiredLength = 10,
                // Requiert des caractères autres que des caractères alphanumé-
riques
                RequireNonLetterOrDigit = true,
                // Doit contenir des chiffres
                RequireDigit = true,
                // Requiert des lettre en minuscule
                RequireLowercase = true,
                // Requiert des lettre en majuscule
                RequireUppercase = true,
            };

            //Définit que tous les utilisateur sont sujet au verrouillage de
compte
            manager.UserLockoutEnabledByDefault = true;
            // Déblocage de compte après 20 minutes
            manager.DefaultAccountLockoutTimeSpan = TimeSpan.FromMinutes(20);
            // Tentatives maximales de tentatives de connexions échouées avant
blocage de compte
            manager.MaxFailedAccessAttemptsBeforeLockout = 5;
```

Extrait de code 1 configuration ASP.NET Identity

Il est à noter qu'il est possible dans le même fichier de configurer « ASP.NET Identity » pour qu'il dispose d'un service de messagerie pour qu'il puisse envoyer les courriels de confirmation lors d'inscription à l'application. Une validation SMS peut également être utilisée en utilisant un service de messagerie téléphonique.

5.6 Conception du modèle conceptuel, relationnel et gestion de la persistance transparente

5.6.1 Modèle relationnel

Le modèle relationnel est inspiré d'une proposition de schéma relationnel basé sur le modèle RBAC du NIST selon (Post, 2010).

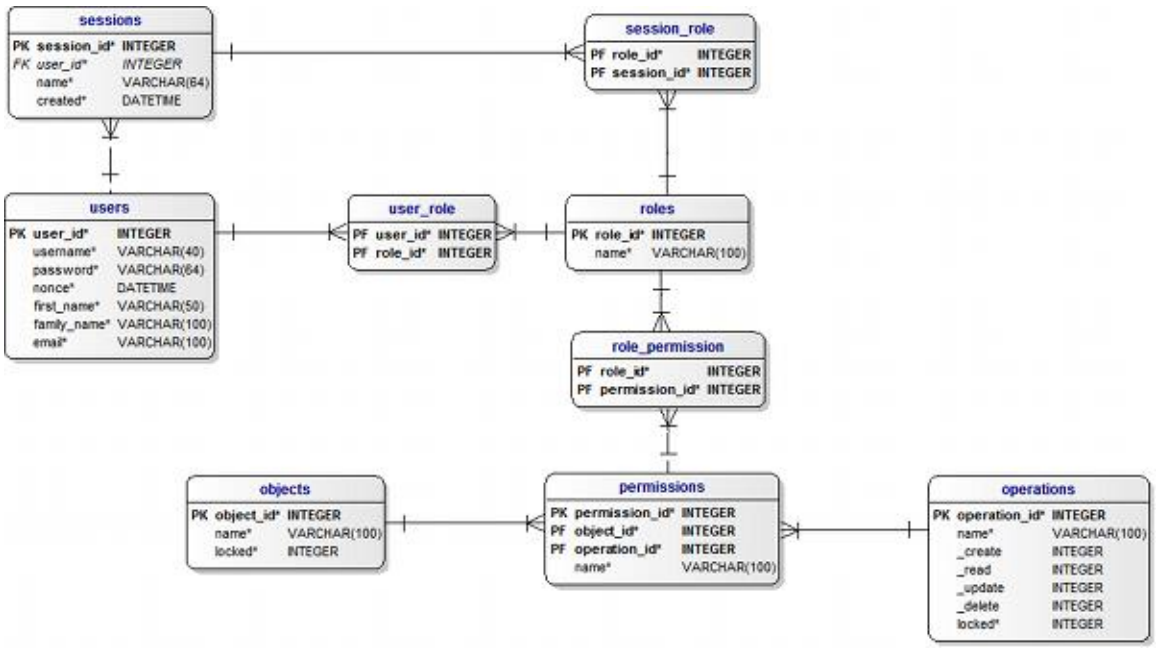

Figure 2 Représentation relationnelle du modèle RBAC selon (Post, 2010)

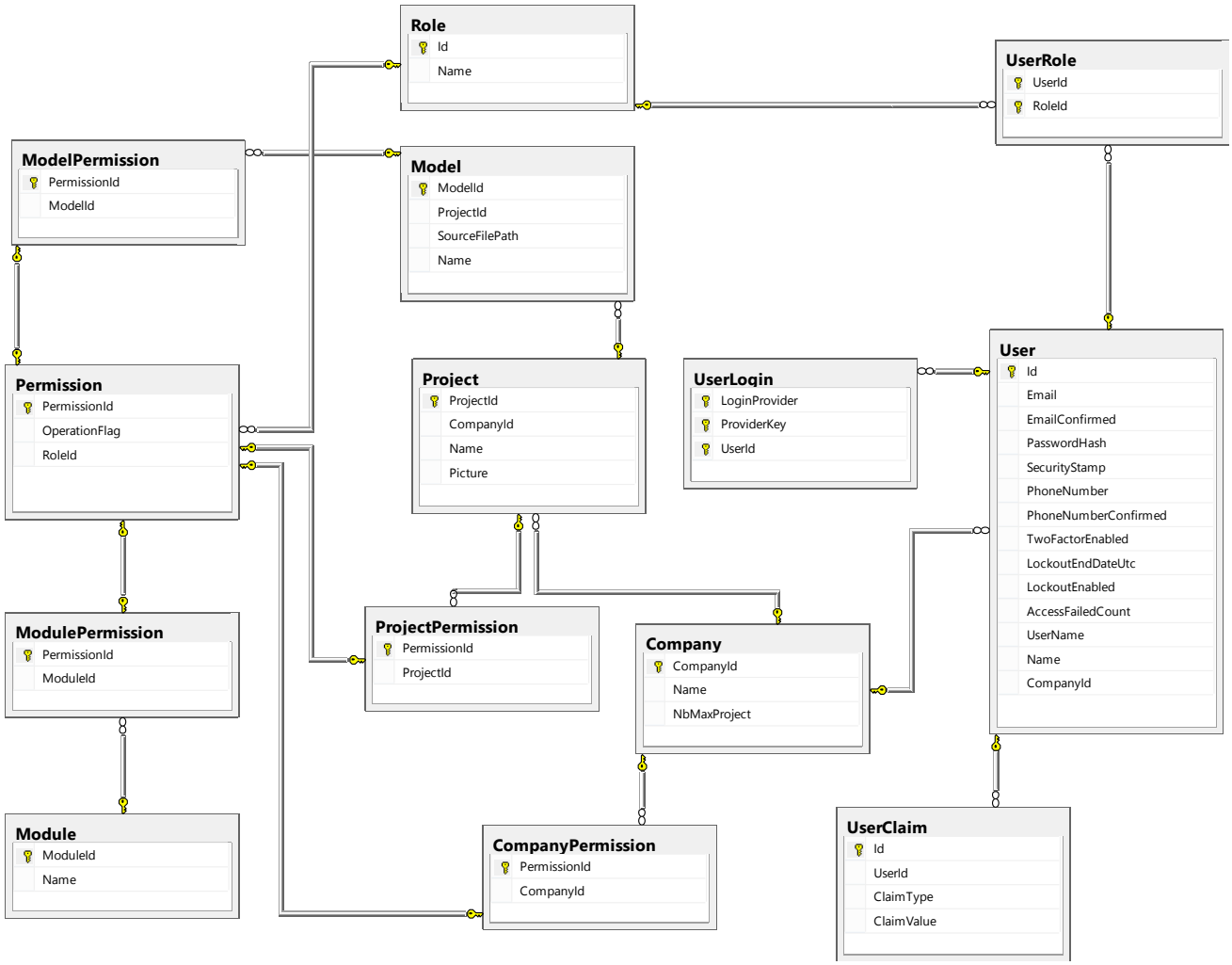

Figure 3 Schéma relationnel de la solution

Bien que le schéma du modèle RBAC (Figure 2) ait fortement inspiré le schéma final, il faut noter que le schéma de départ utilisé est celui fourni et imposé par la librairie « ASP.NET Identity ». Les tables « User, UserLogin, UserClaim, Role et UserRole » proviennent d'ailleurs de ce dernier schéma de base. Les tables « UserLogin et UserClaim » sont en revanche inutiles puisque le concept de « *claim* » n'est pas utilisé et l'authentification externe non plus. Concernant la table « *User* », toutes les colonnes autres que « CompanyId » et « name » sont utilisées par « ASP.NET Identity » pour gérer l'authentification et sont présentes également dans le schéma de base.

Ce schéma relationnel comporte certaines différences par rapport au schéma théorique du modèle RBAC (Figure 2).

Les tables « *sessions* » et « *session_roles* » n'ont pas été retenues puisqu'elles réfèrent au concept de séparation des tâches. Or ce concept n'est pas présent dans les besoins de départ identifiés.

Les tables « Permission » et « Operations » ont été fusionnées. Cette décision a été prise puisqu'il s'est avéré plus efficace de considérer une opération comme étant un seul

« *flag* » indiquant les opérations permises. Ainsi, au lieu d'avoir une colonne « _create » et « _read » pour indiquer si l'opération permet de lire et écrire, une colonne « *OperationFlag* » contient la représentation décimale d'un nombre composé d'une série de « *flags* » binaires correspondant aux actions permises. La signification des « *flags* » est gérée du côté applicatif seulement. Étant donné qu'il ne reste que cette seule colonne dans la table « Operation », et que la relation « Permission » et « Operation » est de plusieurs à un, il est pertinent de fusionner les deux tables. Ces modifications ont, par contre, quelques désavantages. Cela a pour effet de dénormaliser le schéma en permettant la redondance de valeur dans le champ « OperationFlag ». En outre, les valeurs de la colonne « OperationFlag » ne peuvent être validées du côté de la base de données puisque cela nécessiterait une connaissance des valeurs possibles qui est seulement accessible du côté applicatif.

La colonne « *name* » a été enlevée de la table « Permission » puisque celle-ci n'est pas pertinente. En effet, les opérations activées par la colonne « OperationFlag » permettent de décrire suffisamment la permission.

La table de liaison « Role_Permissions » a été enlevée et une clé étrangère vers le rôle a été mise dans la table Permissions. Cela permet d'empêcher qu'une permission soit attitrée à plusieurs rôles. Bien que cela a comme conséquence de faire de la répétition de données. Il se trouve que les données ne sont pas lourdes en mémoire, car il y a seulement trois champs de valeurs entières. Cette décision a été prise afin de faciliter la gestion des attributs de permissions. En effet, les données d'une permission sont trop peu nombreuses et trop simples pour que ce soit avantageux pour l'utilisateur de faire une recherche d'une permission existante au lieu de lui-même ajouter une nouvelle permission. De plus, la gestion de la suppression et du changement des permissions est facilitée par le fait que la suppression ou la modification n'affecte pas d'autres rôles que celui modifié.

La table « *Objects* » dans BIMWeb représente les différentes entités. Une table par entité a donc été ajoutée. Pour lier une permission à une entité spécifique, il est nécessaire de considérer la notion d'héritage. Ceci consiste à avoir une permission et une relation enfant consistant en une permission pour une entité. La stratégie de délégation permet cette situation et a donc été utilisée pour faire la modélisation. Des tables enfants de la table « *Permission* » ont été ajoutées (c.-à-d. « *CompanyPermission* », « *ModulePermission* », « *ModelPermission* », « *ProjetPermission* »). La contrainte de généralisation/spécialisation est disjointe et complète. La contrainte disjointe indique que la spécialisation est exclusive, c'est-à-dire qu'une permission d'entité (p. ex. « ModelPermission ») ne peut avoir la même « Permission » qu'une autre permission d'entité. La contrainte complète signifie ainsi qu'une permission doit obligatoirement se spécialiser en une permission par entité. Du point de vue de la base de données, les moyens pour s'assurer du respect de ces contraintes sont très difficiles à implémenter puisque cela nécessiterait des déclencheurs complexes qui seraient lourds à maintenir. Ce sera donc l'application qui se chargera d'assurer la cohérence des spécialisations.

5.6.2 Utilisation d'un « Enum Flag » pour identifier une opération autorisée par une permission

Il a été mentionné précédemment qu'une permission possède un attribut « OperationFlag » permettant d'indiquer les opérations permises. Les valeurs possibles sont gérées du côté applicatif. Il a été décidé d'utiliser un « Enum » de type « Flag ». Ce type de donnée fonctionne comme une série de « flags » binaires activés ou non. Chaque « flag » possède sa propre valeur décimale d'une puissance de 2.

```
[Flags]
public enum Operation
{
    CreateLowerEntities = 1,
    ReadLowerEntities = 2,
    UpdateLowerEntities = 4,
    DeleteLowerEntities = 8,
    ManageUsersAndPermissionsLowerEntities = 16,
    Read = 32,
    Update = 64,
    Delete = 128,
    ManageUsersAndPermissions = 256
}
```

Extrait de code 2 « Enum Flags » pour identifier les opérations

L'utilisation d'un « Enum Flag » permet d'avoir une opération permettant un ensemble d'opérations. Cela permet de faciliter la vérification des permissions contrairement à l'utilisation d'un champ booléen pour chaque opération possible. On pourrait affecter, par exemple, une opération de lecture et une opération de suppression à un champ « Operation ». On peut ensuite vérifier si l'opération permet la suppression. Le code C# pour effectuer de telles opérations ressemblerait à ceci :

```
Operation currentOperation = Operation.Delete | Operation.Read;

if (currentOperation.HasFlag(Operation.Delete)
        Console.Write("L'opération permet la suppression");
```

Extrait de code 3 Exemple de test d'une opération

5.6.3 Vérification hiérarchique des permissions

La matrice suggérée et les opérations possibles actuellement décrites dans la partie d'analyse des besoins ont un défaut majeur : il n'y a pas de droits qui représentent la possibilité de gérer les usagers et les permissions. Également, étant donné la nécessité d'attribuer des permissions de manière hiérarchique, des permissions doivent être attribuées constamment à chaque rôle qui doit avoir des droits sur les entités inférieures dans le cas de création de nouvelles entités inférieures. Prenons le cas du rôle « *Admin* » d'une compagnie. Ce rôle doit pouvoir avoir l'autorité complète sur toutes les entités inférieures à la compagnie sur lequel le rôle porte. Cela signifie que pour chaque nouveau projet, ou modèle de cette compagnie, le système devra ajouter des droits au rôle en

question. Cela rend difficile la gestion des accès. Une solution est d'avoir des droits qui permettent d'attribuer des permissions de manière descendante.

Les droits possibles avec l'ajout de la gestion de la hiérarchie sont les suivants.

Tableau 7 Opérations améliorées sur les entités

Droit	Lettre	Description
Création d'entités inférieure	CI	Peut créer des entités inférieures
Lecture	R	Accès en lecture
Lecture des entités inférieures	RI	Permets la lecture sur les entités inférieures
Mise à jour	U	Accès en écriture
Mise à jour des entités inférieures	UI	Permets les mises à jour sur les entités inférieures
Suppression	D	Peut supprimer l'entité
Suppression des entités inférieures	DI	Permets les suppressions des entités inférieures
Gestion des utilisateurs et les permissions	M	Peut inviter un utilisateur externe à la compagnie (seulement pour le modèle)
Gestion des utilisateurs et des permissions des entités inférieures.	MI	Permets la gestion des utilisateurs et des permissions des entités inférieures.

La matrice améliorée est la suivante

Tableau 8 Matrice améliorée des opérations et entités

		Compagnie									Projet									Modèle								
Entité	**Rôle**	**CI**	**R**	**RI**	**U**	**UI**	**D**	**DI**	**M**	**MI**	**CI**	**R**	**RI**	**U**	**UI**	**D**	**DI**	**M**	**MI**	**CI**	**R**	**RI**	**U**	**UI**	**D**	**DI**	**M**	**MI**
Compagnie	User		X																									
	Admin		X	X	X	X		X		X																		
	AdminBIM		X			X		X		X																		
Projet	Gestionnaire de projet Bim		X	X							X		X		X		X	X	X									
Modèle	Lecteur		X									X									X							
	Éditeur		X									X			X						X		X		X			

Dans l'implémentation proposée, chaque entité du système à protéger implémente l'interface « *IEntityWithPermission* ». Cette interface représente une entité qui dispose de permissions. Elle permet de masquer l'hétérogénéité entre les différents types d'entités

pour permettre d'exposer les attributs communs, c'est-à-dire la liste de permissions, l'entité supérieure et le type de l'entité. L'attribut « *UpperEntity* » permet de représenter une liste simplement chainée vers le haut dans la hiérarchie des entités puisqu'elle référence le parent de l'entité. Rappelons la hiérarchie des entités :
Compagnie -> Projet -> Model. L'entité de module est quant à elle non hiérarchisée donc l'attribut « *UpperEntity* » aura la valeur « *null* » pour indiquer qu'elle n'a pas de parent.

L'introduction d'une liste chainée permet ainsi une validation des permissions de manière ascendante à « n » niveaux hiérarchiques. Cette solution permet d'ajouter, au besoin, autant de niveaux hiérarchiques que nécessaire. À partir d'une entité, on peut naviguer dans ses entités parentes à la recherche d'une permission spécifique. Cela est particulièrement utile lorsqu'on veut vérifier si un utilisateur a un droit spécifique sur une entité dont la permission n'est pas explicitement accordée à l'utilisateur.

Il convient aussi de définir ici les deux types de permission possibles. Le premier est le type de permission explicite qui consiste en une permission qui permet l'opération directe sur l'entité. Par exemple, les permissions d'opération « *Read, Update, Delete et ManagerUsersAndPermissions* » sont dites explicites puisqu'elles s'appliquent explicitement sur une entité spécifique. On peut par exemple déduire directement qu'un utilisateur est autorisé à lire un projet si cet utilisateur possède un rôle en allouant une permission sur le projet dont l'opération allouée permet la lecture. Par contre, si la permission explicite nécessaire n'est pas présente, il faut tout de même valider les entités supérieures pour s'assurer que l'une d'elles permet une opération dite implicite vers le niveau inférieur qui viendrait alors faire en sorte d'autoriser l'opération. Le type de permission implicite caractérise les opérations « *CreateLowerEntities, ReadLowerEntities, UpdateLowerEntities, DeleteLowerEntities et ManageUsersAndPermissionsLowerEntities* ».

Il est intéressant de noter que l'opération de création d'entités n'est pas disponible en version explicite puisqu'il n'est pas possible d'autoriser une création sur l'entité elle-même puisque celle-ci doit être créée avant de pouvoir y assigner des permissions. Pour créer une entité, par exemple un projet, il est nécessaire d'aller voir si l'utilisateur possède une permission spécifiant l'opération *CreateLowerEntities* sur la compagnie dans laquelle le projet souhaite être créé.

```
public interface IEntityWithPermission
{
    IEnumerable<Permission> Permissions{get;}
    IEntityWithPermission UpperEntity { get; }
    System.Type Type { get; }
}
```

Extrait de code 4 Interface d'entité avec permission

5.6.4 Problème de l'incompatibilité objet-relationnel (*object-relational impedence mismatch)*

Le problème d'incompatibilité objet-relationnel consiste en une série de difficultés techniques et conceptuelles qui sont souvent rencontrées dans une base de données relationnelle utilisée par une application orientée objet. Une base de données relationnelle n'est pas compatible avec le paradigme orienté objet sous plusieurs aspects. Par exemple, elle ne permet pas d'héritage et de polymorphisme, elle n'a pas non plus de notions d'encapsulation.

L'incompatibilité, dans le présent projet, présente un problème important puisqu'il survient dans une partie cruciale, celle de la traduction des requêtes « LINQ » en « SQL ». En effet, « LINQ » permet d'effectuer des requêtes complexes sur des collections de données en mémoire, mais également sur des objets « IQueryable » qui à l'aide d'« Entity Framework » se transforment en « SQL » pour aller interroger la base de données. C'est une bonne pratique, afin de réduire la mémoire utilisée par l'application, d'effectuer la requête « LINQ » la plus restrictive sur la base de données avec un « IQueryable » et le mode « *Lazy Loading* » activé, de manière à faire le filtre du côté de la base de données plutôt que du côté applicatif. Cependant, il se trouve que les requêtes « LINQ » ne peuvent être converties en « SQL » lors de cas comme les suivants :

- Utilisation d'attribut calculé du côté applicatif
- Utilisation de relation d'héritage ou de polymorphisme

La vérification hiérarchique des permissions utilise justement des relations d'héritage et des attributs calculés (c.-à-d. « UpperEntity » est un attribut calculé). La vérification hiérarchique à « n » niveaux fonctionne donc seulement dans un cadre où tous les objets sont placés en mémoire où « LINQ » peut utiliser sans restriction leurs attributs. Il est alors nécessaire d'optimiser l'approche de vérification hiérarchique de manière à ce qu'elle se conforme aux exigences de performance (pas encore spécifiés clairement dans les besoins du projet) tout en gardant l'aspect générique de la vérification, mais ceci dépasse le cadre du présent projet de recherche.

5.6.5 Persistance transparente avec Entity Framework

Le cadriciel *« Entity Framework »,* appelé souvent simplement par son abréviation EF, sera utilisé dans le présent projet. Ce cadriciel permet de faire la liaison objet-relationnel (ORM), c'est-à-dire une correspondance entre les objets d'une application orientée objet et les données de la base de données. Il s'agit de persistance caractérisée de la transparente puisque que les lectures/écritures dans la base de données seront gérées en arrière-plan sans que le développeur ait explicitement à exécuter des commandes « SQL » pour la mise à jour des données. Cette technique permet d'abstraire l'accès aux données pour se concentrer, le plus possible, sur la logique d'affaires et de ne pas se soucier des aspects internes de la base de données. L'utilisation d'un « ORM » permet également de masquer l'hétérogénéité entre les différents types de Système de Gestion de Base de Données (SGBD) qui peuvent être utilisés. Par exemple, bien que le « SGBD » « SQL Server » soit supporté nativement par EF, d'autres SGBD, par exemple Oracle et MySQL peuvent être utilisés en installant certains intergiciels. EF permet de traiter les

requêtes en langage de requête « LINQ » utilisée pour interroger les collections d'objets, afin de faire les conversions en « SQL ».

EF est un choix de prédilection pour une application web « ASP.NET MVC ». Premièrement parce que le paradigme orienté objet est cœur d'une application de ce genre. Deuxièmement parce que cela permet de faire des requêtes performantes en supportant notamment des modes de chargements efficaces. Le mode « *Lazy Loading* » (c.-à-d. de chargement tardif) permet de différer le plus longtemps possible le chargement des données d'un objet. L'exemple typique pour illustrer ce concept et le chargement d'un client et de ses commandes. Il n'est pas souhaitable de charger l'ensemble des commandes du client si l'information désirée est seulement le nom du client. Le paramètre « *Lazy Loading* » permet donc de s'assurer de charger les commandes que lorsque l'on désire explicitement y accéder. Bien que les besoins du système ne précisent pas de scénarios de performances et d'estimations pour indiquer la taille des données qui seront, à court et moyen termes, gérés par le système, il est loisible de croire que la portée du projet fait en sorte d'avoir un nombre très élevé d'objets à gérer. L'optimisation de la performance, bien qu'elle dépasse du cadre de ce projet, est une considération importante et l'utilisation d'un « ORM », comme « Entity Framework » permettra d'optimiser les ressources utilisées dans le futur.

5.6.6 Comparaison des approches : Database First, Model First et Code First

« Entity Framework » permet d'être utilisé selon trois modes : « Database First », « Model First » et « Code First ». Cette section s'attarde à analyser les différentes approches. La comparaison est basée sur l'ouvrage de (Chadwick, Snyder, & Panda, 2012, p. 159).

5.6.6.1 Database First

Cette approche s'adresse aux développeurs qui préfèrent une conception orientée donnée. Elle est typiquement utilisée lorsqu'une base de données existe et surtout lorsqu'elle a été conçue par un concepteur de BD (voire même un administrateur de BD (DBAs)) et lorsqu'une grande importance est accordée à la qualité du schéma relationnel. À partir d'une base de données existante, « Entity Framework » génère les modèles d'affaires basés sur les tables et les colonnes. Un fichier de configuration « .edmx » est créé et maintenue et contient de l'information à propos du modèle relationnel, du modèle conceptuel de données et la liaison entre les deux. Un outil « designer » présent dans « Visual Studio » permet d'assister le développeur lors de la personnalisation du modèle conceptuel généré afin qu'il correspondre aux besoins de l'application.

5.6.6.2 Model First

« Model First » s'applique lorsqu'une base de données est inexistante et que l'on désire la générer à partir d'un modèle de données créé à travers un assistant. Avec l'option « designer » de « Entity Framework » sous « Visual Studio », le développeur peut créer le modèle de données à partir de zéro. La fonction « Entity Framework » génère ensuite le schéma relationnel de la base de données. Comme l'approche « Database First », un fichier de liaison doit être maintenu.

5.6.6.3 Code First

Cette approche est centralisée autour du code. Elle consiste à concevoir des classes en spécifiant tous les champs et les attributs (c.-à-d. le metadata) nécessaires pour la génération d'un schéma relationnel. Pour chaque champ, il est possible de spécifier des attributs de type « *Data Annotation* » afin d'indiquer à la fonction « Entity Framework », par exemple, les contraintes d'intégrité référentielle, d'unicité, de clé primaire, de taille maximale, etc. L'avantage de cette méthode est que la base de données peut-être générée seulement à partir des classes sans avoir à spécifier des fichiers de configuration contenant les règles de liaison entre les entités. Cette approche permet aussi de s'éloigner le plus possible de la base de données pour se concentrer que sur le code. L'approche « Code First » permet également l'utilisation de migrations de la base de données qui permet de versionner la base de données. Le thème de migrations est couvert dans la section suivante. La génération de la base de données est effectuée par convention de manière dynamique à l'exécution. Un désavantage de cette approche est qu'il n'est pas possible de faire des changements directs dans la base de données, car le sens des changements doit se faire du code vers la base de données et non le contraire.

5.6.6.4 Sélection d'une approche

Dans la sélection d'une approche, trois facteurs majeurs sont à considérer par l'ingénieur : les préférences des développeurs, c'est-à-dire s'ils sont plus concentrés autour du code ou bien autour de la base de données, la situation initiale qui inclue la possession d'une base de données existante ou non, et finalement, la maintenabilité et la facilité de gestion des versions des approches.

Pour le premier facteur, la préférence du développeur est principalement de se concentrer sur le code source et de ne pas trop se soucier de comment « Entity Framework » génère la base de données derrière. Il est intéressant que l'on oppose ici gestion par le code et gestion par une interface graphique. Or il est de l'avis du développeur que le code est plus efficace pour spécifier un comportement puisqu'il a un contrôle explicite dessus. Du code autogénéré en revanche, comme c'est le cas avec « Database First » et « Model First » à partir d'un assistant générant du code (c.-à-d. des balises XML), offre moins de contrôle. L'approche « Code First » est donc appropriée pour ce facteur.

Pour le deuxième facteur, la situation initiale comportait une base de données, cependant, celle-ci n'est pas complexe. De plus, des classes étaient déjà présentes, mais certaines ne correspondaient pas à la base de données. Le gain de partir d'une base de données existante n'est donc pas élevé. Aucune approche ne se démarque véritablement pour ce facteur.

Pour le troisième facteur, « Code First » offre un net avantage en permettant d'être versionné aisément par l'utilitaire de gestion des migrations. Chaque incrément des migrations peut être versionné sur un gestionnaire de code source.

L'approche « Code First » est donc l'approche qui se démarque le plus pour les différents facteurs évoqués, elle a donc été sélectionnée pour ce projet.

5.6.7 Utilisation du système de migrations de la base de données

L'approche « Code First » choisie pour la liaison entre « Entity Framework » et la base de données offre un avantage important de permettre de versionner les changements apportés à la base de données.

Une migration est décrite essentiellement par une classe contenant une méthode pour faire la migration (UP) et une méthode pour défaire la migration (DOWN). Le code généré dans ces méthodes n'est pas du SQL, mais bien un dialecte propre au système de migration qui est facile à comprendre et à modifier.

L'exemple suivant montre une instruction effectuant la création d'une table avec ses champs et ses propriétés.

```
CreateTable(
    "dbo.Company",
    c => new
        {
            CompanyId = c.Int(nullable: false, identity: true),
            Name = c.String(nullable: false, maxLength: 100),
            NbMaxProject = c.Int(nullable: false),
        })
    .PrimaryKey(t => t.CompanyId);
```

Extrait de code 5 Exemple de migration

Lorsqu'une migration est réalisée, les classes mappées sont inspectées pour trouver tout changement et une nouvelle classe de migration est créée dans le dossier « *Migrations* » du projet « ASP.NET ». Pour qu'une classe soit considérée dans le processus, elle doit être contenue dans le contexte de base de données utilisé par la migration courante. Ainsi, l'activation des migrations se fait par contexte de base de données. Dans le présent projet, il n'existe qu'un contexte. Il est à noter qu'une table de base de données nommée « *_MigrationHistory* » est maintenue par le système de migration de « Entity Framework ». Par contre, le contenu de la table n'a pas à être édité par le développeur puisque la gestion est faite de manière transparente.

Un des avantages des migrations est que l'on peut spécifier des données à créer lors d'un déploiement de la base de données. La technique appelée « *Seed* » consiste à mettre des insertions de données dans la méthode « *Seed* » de la configuration de la migration (Fichier Configuration.cs) dans le dossier « Migrations ». Cela peut-être utile si l'on désire à chaque lancement de l'application, avoir certaines données pour faire des tests d'intégration ou de système.

5.6.8 Patron Data Access Layer

Il est en général une bonne pratique de découpler l'accès aux données de la logique d'affaire de l'application. Les principales raisons sont que l'on veut habituellement avoir une bonne encapsulation des opérations. On veut des modules cohésifs et restreindre le couplage des modules entre eux. On veut ainsi éviter d'avoir à modifier la logique d'affaires dès que le support de persistance de données change. Un exemple typique de

changement est une mise à jour du « SGBD » utilisé ou bien un changement pur et simple du type de « SGBD ». Afin de mettre à l'abri la logique d'affaire de ces changements, il convient d'utiliser le patron « *Data Access Layer» (DAL).*

Une implémentation très simple, voire naïve, d'un DAL serait typiquement de définir une interface définissant des méthodes regroupant des opérations typiques sur les données comme « ObtenirTousLesProjets », « SupprimerProjet », « ModifierUneCompagnie », etc. Ensuite une classe concrète implémenterait l'interface et contiendrait un contexte de base de données « *DbContext »*, cette classe serait utilisée par les modules de code de la logique de l'application pour lire/modifier/supprimer/créer les données. Cette méthode, bien qu'elle résolve bien le problème de couplage précédemment abordé, comporte tout de même un problème majeur. Elle n'est pas générique puisque l'on doit définir par exemple une série d'opérations à effectuer sur les données, et ce, pour chaque type d'entité de la base de données (p. ex. « ObtenirTousLesProjets », « ObtenirTousLesModeles »). Cela fait en sorte de dupliquer beaucoup de code.

Une autre solution consiste à utiliser des sous-patrons du « DAL ». Les patrons « *Repository »* et « *Unit of Work »* figurent dans les bonnes approches pour implémenter le « DAL ». L'implémentation réalisée est fortement basée sur un tutoriel paru sur le site d'« ASP.NET » (Dykstra, 2013).

Un « *repository »* ou dépôt de données contient un ensemble d'opérations sur les données d'un type d'entité en particulier. Ce dépôt peut être généralisé en incorporant la notion de type générique disponible sous C#. Ainsi, toutes les opérations n'ont qu'à être implémentées une seule fois.

L'unité de travail ou « *Unit of Work »* coordonne le travail des différents dépôts de données (un par type d'entité) en partageant un seul contexte de base de données pour tous les dépôts.

Outre la réutilisation du code que cela apporte, « Repository » et « Unit of Work » permet de faciliter les tests unitaires. En effet, il est habituellement recommandé de ne pas utiliser un véritable mécanisme de persistance sur disque lors de test unitaire, mais plutôt un module de persistance en mémoire. Une bonne approche consiste à utiliser une couche d'accès aux données sous forme de « Mock ». Le « Repository » et le « Unit of Work » peuvent tous deux être « moqués » pour être capables de gérer le système de stockage en mémoire utilisé pour les tests.

Étant donné que les utilisateurs et les rôles sont déjà gérés par « ASP.NET Identity » avec du côté de la logique d'affaires, un « RoleManager » et un « UserManager », et du côté de la couche d'accès aux données, un « RoleStore » et un « UserStore », il été décidé de ne pas court-circuiter cette manière de faire et donc de ne pas étendre le patron « Repository » et « Unit of Work » pour les utilisateurs et des rôles. Les utilisateurs et les rôles doivent donc être accédés via les « RoleManager » et « UserManager ».

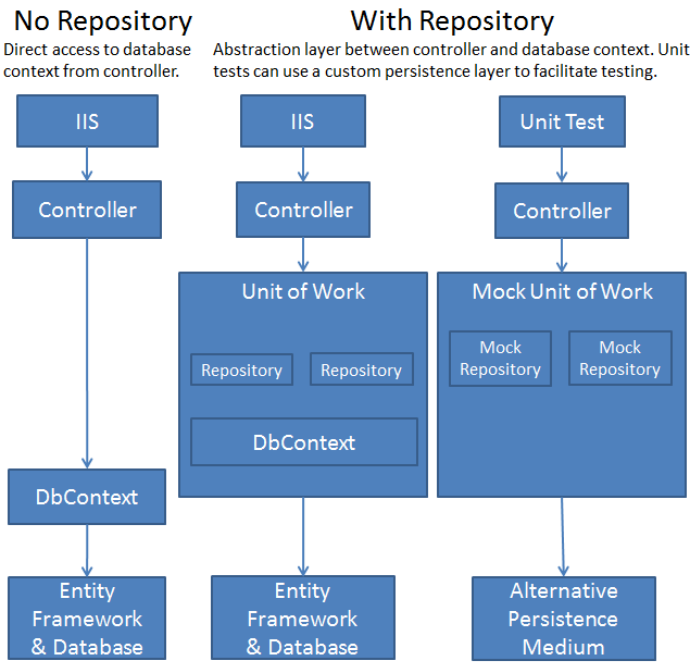

Figure 4 Comparaison de l'approche des patrons Repository et Unit Of Work rapport à l'approche sans patrons (Dykstra, 2013)

5.7 Mécanisme de vérification des permissions

La vérification d'une permission concerne un utilisateur, une opération, et une entité. La vérification consiste donc à vérifier si pour un utilisateur, il existe un rôle relié à une permission autorisant l'opération demandée et qui est reliée à l'entité concernée.

En « ASP.NET MVC », on veut typiquement faire une vérification de permission dans trois scénarios. Le premier scénario est une vérification à un endroit arbitraire dans le code d'une méthode de contrôleur. Le deuxième scénario est lorsque dans une vue, on veut vérifier si l'on doit afficher ou masquer certaines informations à l'utilisateur. Le troisième scénario est lorsque l'on désire restreindre l'accès à un contrôleur ou à une méthode de contrôleur en ayant recours à un attribut (p. ex. « *Authorize* »).

Avant de considérer comment satisfaire les scénarios, il est nécessaire d'avoir une méthode qui effectue la vérification de la permission. Cette méthode pourrait être ajoutée dans la classe « *User* ». Par contre, cette classe ne contient pas directement les rôles assignés à l'utilisateur puisque cette information est accessible seulement dans le « *UserManager* » et dans le « *RoleManager* » fourni *avec « ASP.NET Identity »*. Un utilisateur et la vérification de permission ne sauraient également être cohésifs. Par contre, on doit considérer que de disposer de l'accès à une méthode « *HasPermission* », à partir d'un « *User* », serait très utile. Or il se trouve qu'il est possible de rendre accessible la méthode « *HasPermission* » par la classe « *User* » tout en gardant cette méthode à l'extérieur de celle-ci, en utilisant une méthode d'extension. Une méthode d'extension consiste à mettre l'expression (par exemple, « *this Type nomVariable* ») comme premier paramètre d'une méthode pour rendre accessible cette méthode au type indiqué. Le code

ci-dessous représente l'implémentation d'une méthode d'extension permettant de rendre une instruction de ce genre possible : « *User.HasPermission(Operation.Read, entity)* »

```
public static bool HasPermission(this User user, Operation requestedSingleOperationFlag,
IEntityWithPermission entity, ApplicationRoleManager roleManager, ApplicationUserManager
userManager)
        {
            if ((requestedSingleOperationFlag & (requestedSingleOperationFlag - 1)) != 0)
            {
                throw new NotSupportedException("A permission check for multiple operation at a
time is not supported.");
            }

            bool hasPermission = false;
            IEntityWithPermission currentEntity;
            Operation currentOperation;
            bool searchingUpperEntitiesPermissions;

            IList<string> roleStrings = userManager.GetRoles(user.Id);

            foreach (String roleString in roleStrings)
            {
                Role role = roleManager.FindByName(roleString);

                foreach (Permission permission in role.Permissions)
                {
                    currentEntity = entity;
                    currentOperation = requestedSingleOperationFlag;
                    searchingUpperEntitiesPermissions = false;

                    while (currentEntity != null && currentOperation != default(Operation))
                    {
                        hasPermission = (currentEntity.Permissions.Any(
                            p => p.OperationFlag.HasFlag(currentOperation) &&
                            p.PermissionId == permission.PermissionId));

                        if (hasPermission)
                            return hasPermission;

                        if (!searchingUpperEntitiesPermissions)
                        {
                            currentOperation =
OperationUtil.ConvertOperationToLowerEntitiesOperation(currentOperation);
                            searchingUpperEntitiesPermissions = true;
                        }

                        // Check the upper entities permissions to find permission that
                        // allows lower entitie operations
                        currentEntity = currentEntity.UpperEntity;
                    }
                }
            }

            return hasPermission;
        }
```

Extrait de code 6 Première méthode d'extension de vérification des permissions

Il est à noter que la méthode nécessite un « *RoleManager* » et un « *UserManager*, » car ces deux objets sont nécessaires pour être en mesure de trouver les rôles et les permissions requises pour l'autorisation. Pour éviter la nécessité de fournir le « *RoleManager* » et le « *UserManager* » à chaque vérification de permissions, une seconde méthode d'extension peut être implémentée pour obtenir ces paramètres.

```
  public static bool HasPermission(this User user, Operation requestedSingleOperationFlag,
IEntityWithPermission entity)
        {
            ApplicationRoleManager roleManager =
HttpContext.Current.GetOwinContext().Get<ApplicationRoleManager>();
            ApplicationUserManager userManager =
HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();
            return HasPermission(user, requestedSingleOperationFlag, entity, roleManager,
userManager);
        }
```

Extrait de code 7 Deuxième méthode d'extension de vérification des permissions

Le fait d'avoir deux méthodes dont une qui prend le « RoleManager » et le « UserManager » en paramètre et une autre qui prend seulement les paramètres « Operation » et « IEntityWithPermission » s'explique par le fait que les tests en seront facilités. En effet, lors des tests unitaires, la meilleure pratique est de toujours de tester la plus petite quantité de code possible. Or aller chercher le « RoleManager » et le « UserManager » selon le « HttpContext » courant est une mauvaise pratique puisque cela exige que les tests créent un contexte et tous les autres objets contenus dans celui-ci, notamment le « OwinContext » qui est typiquement instancié au lancement de l'application web. Les tests seront abordés de manière plus détaillée dans la section « Conception des tests de sécurité ».

Maintenant que la méthode pour vérifier la permission est présente, on devrait pouvoir l'appeler. Cependant, il serait pertinent de ne pas avoir à aller chercher l'utilisateur courant (connecté) à chaque fois que l'on désire effectuer un test d'autorisation. En effet, sélectionner l'utilisateur courant nécessite quelques lignes de codes et afin de diminuer le plus possible la duplication de ce code, il serait intéressant qu'une méthode réutilisable le fasse à notre place. La solution consiste en une troisième méthode d'extension qui rend cette fois la méthode « *HasPermission* » disponible pour les contrôleurs.

```
        public static bool HasPermission(this ControllerBase controller, Operation
requestedOperation, IEntityWithPermission entity)
        {
            ApplicationUserManager userManager =
HttpContext.Current.GetOwinContext().GetUserManager<ApplicationUserManager>();
            User user =
userManager.FindById(controller.ControllerContext.HttpContext.User.Identity.GetUserId<int>());
            return (user != null && user.HasPermission(requestedOperation, entity));
        }
```

Extrait de code 8 Troisième d'extension de vérification des permissions

Les trois méthodes d'extensions montrées permettent de satisfaire le premier et le deuxième scénario. En effet, de tout contrôleur effectuant l'important du « *NameSpace* » contenant les méthodes d'extensions, il est possible d'exécuter l'instruction « this.HasPermission(operation, entity) » pour effectuer le test de permission. Pour le deuxième scénario qui se trouve à faire la vérification dans une vue, on peut faire la même chose en invoquant l'expression « ViewContext.Controller.HasPermission(…) ».

Pour le troisième scénario, où l'accès à un contrôleur ou une méthode doit être restreint, on peut typiquement sécuriser un contrôleur ou une méthode, en indiquant l'attribut « *Autorize* ». Par exemple, sur une méthode « *AdministratorsOnly* » d'un contrôleur, nous

pourrions vouloir faire un filtre et refuser toute personne n'ayant pas le rôle d'administrateur.
Cela pourrait se faire de la manière suivante :

```
[Authorize(Role = "Administrator"]
public ActionResult AdministratorsOnly()
{
    return View();
}
```

Extrait de code 9 Exemple d'attribut Autorize sur un rôle

Comme évoqué précédemment, le cas qui nous occupe ne considère pas seulement un rôle pour autoriser l'accès, mais également les permissions liées à ce rôle, les opérations et l'entité. L'attribut « Authorize » est donc de base inadaptée à un grain fin comme celui que l'on désire.

On pourrait alors ajouter un attribut « operation » et « entity » dans un attribut personnalisé de « AuthorizeAttribute ». Par contre, les paramètres de l'attribut doivent être constants. Or l'entité concernée par une opération ne peut être mise en constante dans l'annotation puisqu'il serait pris en paramètre dans la méthode du contrôleur. Par exemple, la méthode suivante ne pourrait pas fonctionner :

```
[Authorize(Operation=Operation.Delete, Entity = model]
public ActionResult DeleteModel (IEntityWithPermission model)
```

Extrait de code 10 Exemple d'attribut Autorize dysfonctionnel

La solution est alors de créer un attribut de filtre personnalisé, héritant de « *ActionFilterAttribute* », dont une de ses méthodes, « *OnActionExecuting* », prend en paramètre un « *ActionExecutingContext* » permettant d'accéder aux paramètres passés par la méthode sur lesquels est apposée l'annotation. Dans l'exemple précédent, il s'agit de l'entité à supprimer. Le paramètre de l'entité est alors recherché dans les paramètres de la méthode selon son type (IEntityWithPermission) et non selon un nom spécifique afin d'être plus générique.

```
public class RBACAttribute : ActionFilterAttribute
    {
        public readonly Operation RequestedOperation;

        public RBACAttribute(Operation requestedOperation)
        {
            this.RequestedOperation = requestedOperation;
        }

        public override void OnActionExecuting(ActionExecutingContext filterContext)
        {
            base.OnActionExecuting(filterContext);
            bool hasPermission = false;

            // Find controler parameters that contains a value of IEntityWithPermission type.
            IEnumerable<IEntityWithPermission> entityParameters = filterContext.ActionParameters
.Where(p => p.Value is IEntityWithPermission)
.Select(p => (IEntityWithPermission)p.Value);

            // Only one IEntityWithPermission should be provided
            if (entityParameters.Count() == 0)
                throw new System.ArgumentException("Missing parameter that implements: " +
typeof(IEntityWithPermission).FullName + " on the controller");
            else if (entityParameters.Count() > 1)
                throw new System.ArgumentException("Too much parameters that implements: " +
typeof(IEntityWithPermission).FullName + " on the controller");
            else
                hasPermission = filterContext.Controller.HasPermission(RequestedOperation,
entityParameters.First());

            if (!hasPermission)
                filterContext.Result = new HttpUnauthorizedResult();
        }
    }
```

Extrait de code 11 Attribut de filtre pour contrôleur

Si l'autorisation échoue, l'utilisateur est redirigé vers une page d'accès non autorisé.

Le scénario trois peut maintenant être satisfait. L'accès à un contrôleur dans ce cas-ci à une méthode de contrôleur, peut être implémenté comme ceci :

```
        [RBAC(Operation.Delete)]
        public ActionResult DeleteModel (IEntityWithPermission model)
        {
            ...
        }
```

Extrait de code 12 Exemple d'utilisation d'attribut de filtre

6 Validation avec un expert .NET

Un expert des technologies « .NET », M. Christos Karras, de l'entreprise « MATRICIS Informatique de Montréal », a été rencontré durant le projet pour valider le mécanisme d'autorisation et donner des pistes de solutions pour l'intégration de modules sécuritaire.

6.1 Mécanisme d'autorisation

Il est de l'avis de l'expert que le mécanisme d'autorisation RBAC devra considérer des éléments supplémentaires lors des vérifications des permissions lors des modifications des entités. Si une façade permet qu'on lui passe un objet à modifier comme une entité de projet par exemple, il faut soit s'assurer que toutes les modifications effectuées à l'objet et ses entités référencées soient autorisées, ou bien veiller à autoriser seulement la modification sur l'entité (le projet et non les modèles qu'il référence par exemple). En effet, un module externe pourrait envoyer un objet « Projet » contenant des entités référencées qui auraient été modifiées alors que l'utilisateur pourrait avoir seulement des permissions de modification pour l'objet « Projet ».

6.2 Intégration de modules

De manière générale, l'expert a suggéré de limiter ce que le code des modules peut faire en permettant par exemple d'appeler seulement certaines classes ou « DLL » spécifiques, et non l'ensemble des objets définis dans l'application centrale BIMWeb. Cela peut se faire avec du confinement (c.à-d. « *sandboxing* »). Le confinement peut se faire selon lui en utilisant des « AppDomain » (Application Domain) et le cadriciel « *MAF » (Managed Add-ins Framework).* Ces deux éléments seront traités à la section suivante.

L'expert a mis de l'avant que le mécanisme d'autorisation du système pour l'accès aux entités de base du système (projets, modèles, compagnies, etc.) devrait être utilisé et non contourné par les modules. De cette manière, la base de données et le système de fichier de l'application centrale ne devraient pas être accédés directement, mais plutôt indirectement à partir de façades sécurisées accessibles par les modules. L'expert explique aussi que les modules doivent être confinés de manière à limiter au minimum les opérations qu'ils peuvent faire.

M. Karras souligne également qu'il est souhaitable d'avoir une base de données par module ou bien par compagnie/module dans le but de séparer les données des compagnies tout en assurant une extensibilité à chaque module d'utiliser le schéma de base de données désiré.

Pour le stockage des fichiers des modules, l'expert suggère d'avoir le partitionnement de répertoire « module+compagnie » et de ne permettre qu'un accès indirect via une façade sécurisée. La façade pourrait retourner des « *Streams* » de lecture et d'écriture à des endroits spécifiques selon les droits du module.

7 Survol de l'intégration de module sécuritaire sous ASP.NET

Cette section sera un survol sur les stratégies sous « ASP.NET » pour faire une intégration de modules sécuritaire. Il y a quelques raisons qui expliquent pourquoi il n'y aura qu'un survol. Premièrement, les besoins concernant l'intégration de module sont imprécis. Le SRS et le document de vision ne couvrent pas avec un grain assez élevé le fonctionnement de l'intégration des modules. Ensuite, aucune décision architecturale ou conceptuelle n'a été prise lors du début de l'étude de la sécurité de l'intégration des modules. Cela rend difficile l'élaboration des stratégies de sécurité puisque l'architecture du système et ses composants restent à définir. De plus, une intégration de module sécuritaire pourrait faire l'objet d'un projet de fin d'études entier étant donné la complexité que cela peut amener. Les conseils de l'expert « .NET » rencontré ont néanmoins permis d'orienter les recherches.

7.1 Confinement avec AppDomain

Un « AppDomain » consiste en un environnement d'exécution de type carré de sable (*sandbox*) qui confine du code pour qu'il ne puisse pas être en mesure d'appeler et d'utiliser toutes les méthodes du système central (dans ce cas-ci BIMWeb). Dans le contexte où du code confiné (c.-à-d. un module d'extension) souhaite appeler du code de l'hôte, il est nécessaire d'établir une manière de communiquer avec lui de manière sécuritaire. C'est ce à quoi le « Managed Add-ins Framework » (MAF), décrit dans la section suivante, permet de faire.

7.2 Managed Add-ins framework (MAF)

Ce cadriciel permet de faciliter la communication entre un hôte (*host*) et des plugiciels (traduction libre de « *add-in* »). Cette communication peut être faite dans un environnement interdomaine où les plugiciels sont confinés de l'hôte et des autres plugiciels. Du point de vue de ce cadriciel, la seule différence entre un hôte et un plugiciel est que l'hôte est celui qui active les plugiciels.

L'hôte et le plugiciel communiquent via un pipeline. Le pipeline est une communication symétrique (dans les deux sens) qui permet l'échange de données. Le pipeline est composé d'une série d'interfaces logicielles permettant de faire le pont entre l'hôte et le plugiciel et établir un contrat d'appel entre les deux.

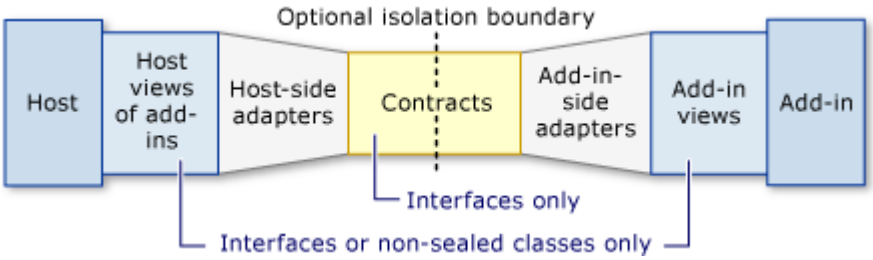

Figure 5 Illustration du fonctionnement de la communication des Add-ins (Microsoft MSDN, 2015)

Les éléments de vue (*views*) du pipeline sont des interfaces ou des classes (dont l'héritage est possible) qui permettent d'offrir une vue sur des méthodes qui seront appelées de bout en bout.

Les adaptateurs (*adapters*) du pipeline convertissent les interfaces ou les classes des vues en contrat (type *IContract*) ou bien l'inverse en convertissant les contrats en classes et en interfaces.

Le contrat (*contract*) est le point de contact entre le plugiciel et l'hôte.

« MAF » permet l'utilisation de plusieurs types de niveaux d'isolation entre les plugiciels et les hôtes. Microsoft MSDN indique que le modèle le plus utilisé est le modèle d'isolation où chaque plugiciel est confiné des autres plugiciels et de l'hôte (Microsoft MSDN, 2015). Cela se défend dans le contexte du présent projet puisque chaque plugiciel peut potentiellement vouloir protéger ses propres données et opérations.

7.3 Active Directory et autorisation

Une autre option était d'utiliser « Active Directory » pour non seulement l'authentification des utilisateurs, mais aussi pour l'autorisation des utilisateurs. Les raisons évoquées sont principalement que les permissions des utilisateurs pourraient être directement reliées à la base de données ou au système de fichier, et donc que l'accès et la modification des données pourraient être faits directement par les modules sans passer par le système central puisque c'est l'utilisateur connecté qui permettrait de valider les accès à la base de données. Le mécanisme d'autorisation, consistant seulement en la validation du côté de la base de données, des permissions des utilisateurs par rapport aux permissions sur le schéma, n'offre cependant pas le niveau de granularité et d'extensibilité de permission suffisant. Premièrement parce que le système requiert une autorisation à un niveau plus fin que le niveau table or « SQL Server » ne permet pas un accès au niveau ligne. Deuxièmement, les permissions modélisées dans le système ne concernent pas uniquement les opérations « CRUD » (c.-à-d. des actions create, read, update et delete), mais également des opérations plus complexes par exemple la lecture d'entités descendantes d'une entité (par exemple, un rôle qui donne l'autorisation de le lire tous les projets et modèles d'une compagnie) ou bien une permission de pouvoir gérer les droits des utilisateurs pour une certaine entité.

8 Conception du système d'attribution des permissions des utilisateurs

Le projet d'exemple « ASP.NET Identity » téléchargé du gestionnaire de paquet « *Nuget* » comprend déjà un outil minimal pour la gestion des usagers des rôles. Par contre, celui-ci n'intègre pas la gestion de permissions, d'opérations sur des entités. L'implémentation de ces fonctionnalités a été entamée, mais elle est inachevée dû à un manque de temps. Cependant, certaines décisions ont été prises et méritent d'être documentées.

8.1 Décisions de conception

8.1.1 Gestion des permissions dans le contrôleur des rôles

Il a été décidé de gérer les permissions dans le même contrôleur que les rôles puisque ceux-ci sont très fortement corrélés. Les permissions n'ont qu'une existence dans un rôle particulier et les interfaces ont été pensées pour que la tâche d'édition de permissions ne soit pas réalisée sans un contexte d'un rôle particulier puisque le scénario utile pour les utilisateurs gérant les permissions est potentiellement de se diriger vers les rôles pour en éditer les permissions et non vers une liste de permissions pour changer le rôle attribué à une permission.

8.1.2 Utilisation de HMTL Helper

Des « HTML Helper » (c.-à-d. assistants HTML) ont été utilisés pour l'ébauche des vues « MVC ». Ceci consiste à faciliter la création de vues « MVC » en utilisant un générateur de contrôle utilisateur pour faire une tâche particulière. L'assistant permet de faire la liaison du contrôleur utilisateur, au modèle (Model Binding) utilisé dans la vue. Par exemple, l'assistant HTML « MvcEnumFlags » a été utilisé pour faire la liaison entre l'« Enum » d'une opération qui peut avoir plusieurs valeurs activées, en utilisant l'instruction suivante dans la vue :

```
@Html.CheckBoxesForEnumFlagsFor(model => model.OperationFlag)
```

Il est possible d'afficher la correspondance en liste de cases à cocher. En gérant pour le développeur la liaison du modèle, l'assistant HTML économise beaucoup de code et augmente la rapidité de développement des vues.

8.2 Prototypes d'interface

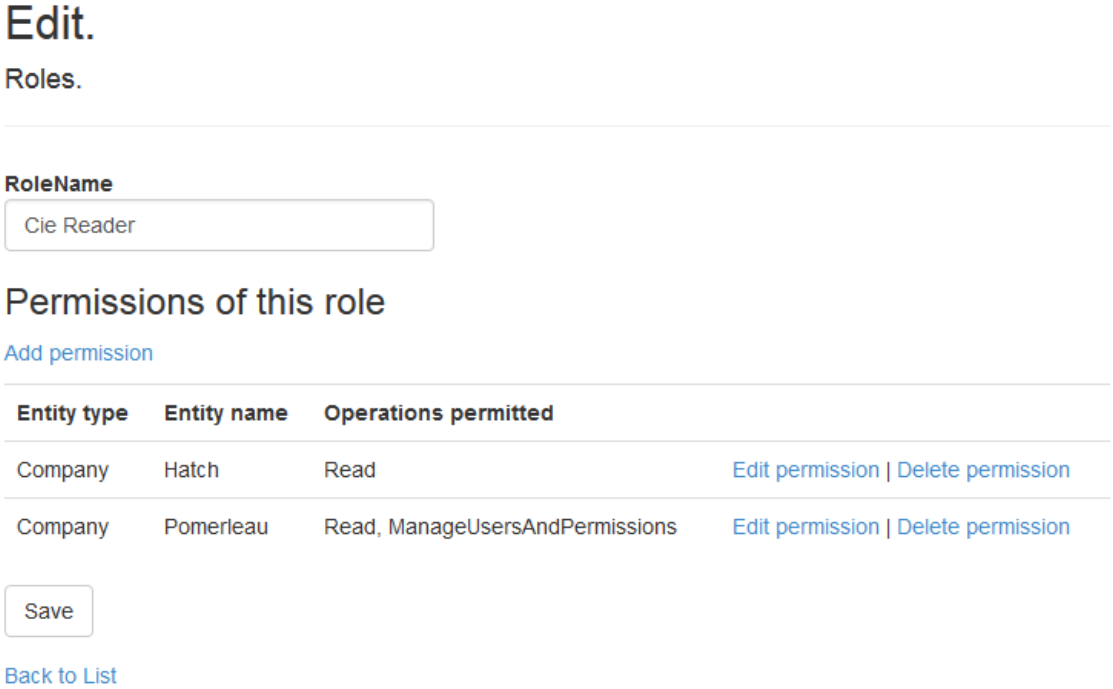

Figure 6 Prototype de l'interface d'édition et de création de rôles

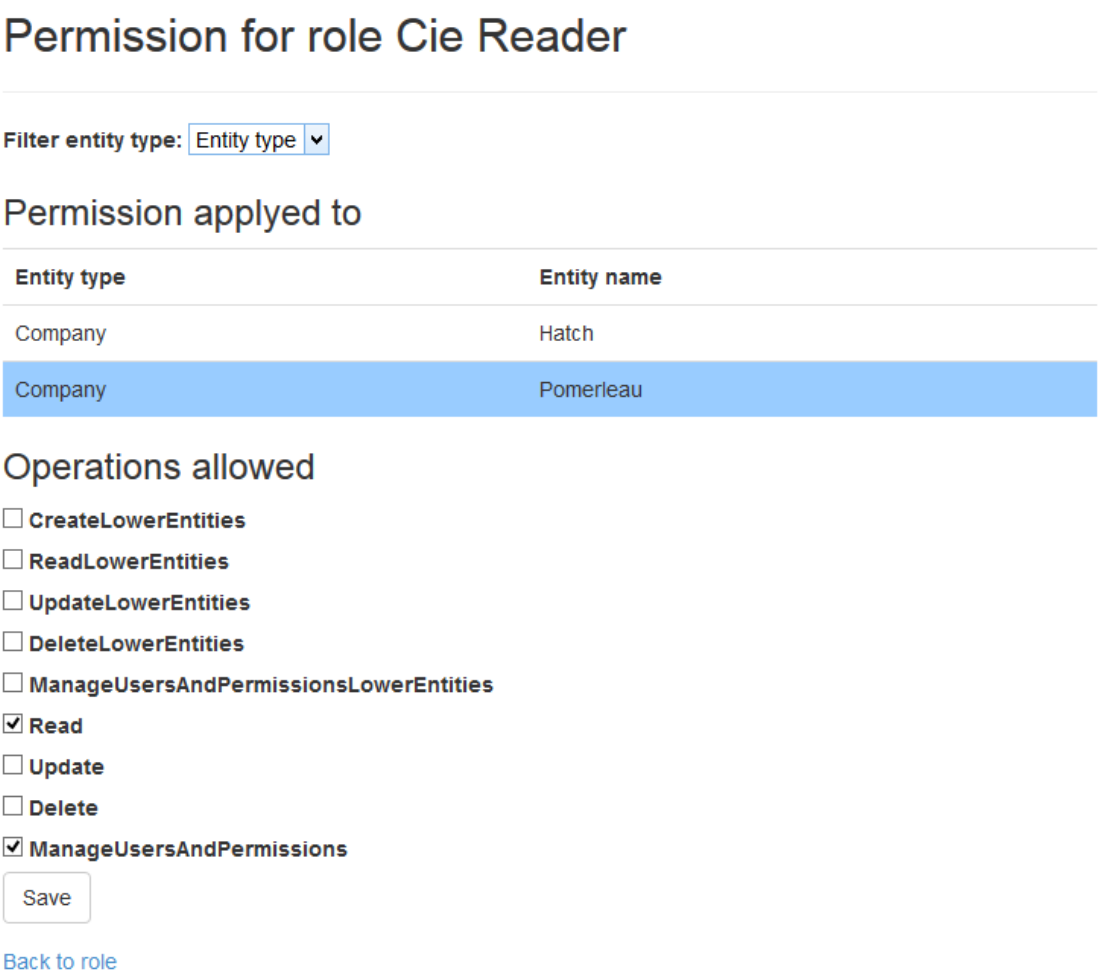

Figure 7 - Prototype de l'interface de création et d'édition de permission

9 Conception des tests de sécurité

9.1 Portée des tests

Les tests réalisés portent sur le mécanisme d'autorisation, c'est-à-dire le mécanisme de vérification des autorisations pour l'accès aux données des entités du système. Les tests visent essentiellement à valider le bon fonctionnement de la méthode « HasPermission ». Par manque de temps, aucun test n'a été réalisé pour tester le code de l'attribut de filtre « RBAC » visant à restreindre l'accès aux contrôleurs et aux méthodes de contrôleurs.

9.2 Méthodologie

Une approche de test unitaire a été utilisée, mais elle n'est pas purement unitaire puisque les unités de codes n'ont pu être totalement isolées des autres. Les causes de cette difficulté seront discutées dans la prochaine section.

Les tests ont été réalisés en prenant en compte trois paramètres, le type de permission nécessaire pour obtenir une autorisation (explicite ou implicite), le nombre de niveaux hiérarchiques à traverser pour trouver une permission dans le cas des permissions implicites et le nombre d'opérations demandées.

L'outil de test unitaire utilisé est celui intégré de base dans « Visual Studio ». Une bonne approche pour réaliser des tests est de ne pas utiliser la véritable base de données puisque le cout de traitement est élevé et on veut généralement des tests qui s'exécutent rapidement. L'approche qui a été expérimentée est l'utilisation d'un « *package Nuget* » nommé « Effort ». « Effort » consiste en un « Provider ADO.NET » qui exécute les opérations dans une base de données en mémoire à place d'une véritable base de données externe persistante. Les avantages sont principalement l'optimisation des performances des tests et la facilitation des tests par le fait que l'on n'a pas à se soucier d'avoir une véritable base de données pour effectuer les tests et devoir à chaque test vider les enregistrements dans la table pour éviter d'avoir des conflits entre chaque test.

L'approche consiste à utiliser une fausse connexion de base de données provenant d'une usine à connexion du paquet « Effort », et à la passer ensuite au contexte de la base de données pour que les opérations par-derrière ne soient opérées qu'en mémoire.

Malheureusement, l'approche s'est avérée infructueuse puisque « Effort » ne supporte pas le contexte « d'ASP.NET Identity ». En effet, une erreur survient lors de l'utilisation avec un contexte héritant du contexte de base d'« ASP.NET Identity ».

```
System.ArgumentException: System.ArgumentException: Keyword not supported: 'instanceid'.
```

Extrait de code 13 Exception soulevée lors de l'utilisation de Effort

En regardant la documentation de « Effort » sur le site officiel, le problème est rapporté et semble avoir été réglé, cependant, le correctif sera disponible seulement lors de la prochaine version (https://aspnetidentity.codeplex.com/workitem/1991).

Il ne suffira lors de la prochaine que de retenter de faire fonctionner « Effort ».
Il a été possible tout de même de réaliser les tests avec l'approche de l'utilisation de la véritable base de données. Par contre, cela est fonctionnel seulement dans le cas où la base de données est vidée avant l'exécution de chaque test. Une base de données locale a d'ailleurs été utilisée, ce qui permet d'éviter d'avoir à utiliser une véritable instance de SQL Server.

9.3 Difficultés d'isolement des tests

L'utilisation d'un cadriciel comme « ASP.NET » et de « Entity Framework » a un désavantage important : il est difficile d'isoler les différents composants des autres pour les tester séparément. Beaucoup de bouts de code du cadriciel utilisent des objets contextuels de l'application. Par exemple, le « HttpContext » regroupe de nombreuses informations détaillant le contexte de la requête en cours et l'utilisateur connecté. Les bouts de codes qui font ce genre d'appel sont difficiles à tester puisqu'il se réfère à des objets de l'application souvent créés au démarrage de l'application. Bien sûr, il est toujours possible de créer ces objets contextuels en tant qu'objet « *Mock* », mais ce processus est lourd et complexe dans le cas d'une application d'envergure utilisant plusieurs cadriciels simultanément. Le présent projet, où l'accent était sur la recherche sur la sécurité, n'a pas laissé suffisamment de temps pour implémenter de tel objet « *Mock* » pour avoir un isolement optimal des objets.

10 Conclusion et discussion

L'étude de cas BIMWeb était un projet ne disposant initialement pas d'architecture et de conception clairement documentée. En outre, aucune stratégie de sécurité n'était documentée. C'est-à-dire que les concepts principaux de sécurité que sont l'authentification et l'autorisation des utilisateurs n'avaient pas été analysés.

Cette recherche visait à analyser les besoins de sécurité de la future application/plateforme BIMWeb, d'élaborer les différentes approches pour satisfaire ses besoins et les comparer pour ne choisir que celles vraiment appropriées. Des décisions d'envergure concernant l'architecture du logiciel, de la conception et de l'implémentation ont dû être prises pour le système global et pas uniquement sous l'aspect sécurité.

Un mécanisme d'autorisation basé sur le contrôle d'accès RBAC a été conçu et implémenté. La librairie « ASP.NET Identity » a été expérimentée et mise en place pour gérer l'appartenance des utilisateurs au futur site web. Le mécanisme RBAC développé permet d'offrir un niveau de granularité intéressant pour les autorisations en considérant l'opération désirant être effectuée et l'entité du système concerné.

Même si ce projet de recherche est terminé, il reste tout de même du travail à faire pour avoir un système de sécurité opérationnel pour le futur système. En excluant les fonctionnalités (c.-à-d. les exigences fonctionnelles) du SRS, qui doivent être nécessairement implémentées, un travail important devra être fait pour réévaluer, avec la présence d'autres contraintes et attributs de qualité, que les artéfacts de conception développés sont adéquats.

De plus, du point de vue de la sécurité, tout n'a pas été couvert. La figure 1 « Tactiques de sécurité couvertes » à la section 3 de ce document illustre quelles tactiques architecturales sont couvertes et lesquelles ne le sont pas. L'identification, l'authentification et l'autorisation des utilisateurs été mené à bien comme décrit précédemment. La détection d'intrusion et la révocation des accès ont été couvertes, mais seulement de manière non exhaustive en bloquant les comptes utilisateurs quand ceux-ci sont utilisés par des tentatives de connexions répétées. Le cryptage des données, bien qu'évoqué dans la section des bonnes pratiques, n'a pas été appliqué à l'ensemble du système. Des procédures doivent donc être entamées pour se munir d'un certificat « SSL/TLS » auprès d'un fournisseur, pour que le protocole « HTTPS » soit utilisé pour chiffrer l'ensemble de trafic géré par BIMWeb. Même si quelques attaques comme l'attaque par force brute, et l'attaque de type « *Cross-Site Request Forgery* » ont été élaborées et que des stratégies ont été évoquées, il ne s'agit que d'un petit ensemble. Il est donc nécessaire de parcourir d'une manière exhaustive des documents de bonnes pratiques comme celui de OWASP (Woschek, 2015) pour identifier d'autres attaques susceptibles de rendre BIMWeb vulnérable et mettre en place des stratégies pour s'en prémunir.

11 Recommandations

11.1 Éviter de précipiter la sécurisation des systèmes

La sécurité du système aurait pu être entamée bien après certaines étapes, dont l'élaboration de l'architecture et de la conception du système global à haut niveau. La sécurisation d'un système requiert d'avoir une bonne connaissance de l'architecture. Bien que la sécurité, comme tout attribut de qualité, influence l'élaboration d'une architecture, commencer à concevoir un système en ayant en tête seulement l'attribut de qualité de sécurité revient à négliger les autres attributs de qualité ou du moins rend les choix architecturaux plus difficiles puisqu'il faut se baser sur des notions non documentées des autres attributs de qualité. Le contexte initial du développement a supposé l'utilisation de « SQL Server », cependant, aucune analyse de performance documentée n'avait été faite à savoir si cette plateforme convenait par rapport au débit et la quantité de données qu'elle devra gérer. Il s'agit ici de l'attribut de qualité de performance qui est un des attributs les plus importants à analyser avant de démarrer un projet. La conséquence fut que l'élaboration des stratégies de sécurité a abouti sur des décisions architecturales globales du projet, ce qui dépasse le mandat initial du projet. Or pour avoir une concentration optimale sur l'aspect sécurité, il aurait été mieux que ces décisions d'envergure comme le choix du SGBD soient documentées au préalable. La recommandation est d'élaborer le plus tôt possible une analyse architecturale du système global avec des scénarios d'attribut de qualité. L'analyse ATAM peut être utilisée pour évaluer l'architecture et renforcer la confiance dans l'architecture développée envers les différents intervenants du projet. L'avantage principal de cette analyse est qu'elle est facile et rapide à faire.

11.2 Respecter le principe du moindre privilège dans les attributions de permissions

Lorsque le système sera opérationnel, il est recommandé de donner des directives aux gestionnaires d'utilisateur et de permissions de respecter le principe du moindre privilège (*least privilege*). Ce principe dicte qu'il faut attribuer le rôle ayant les permissions les moins élevées nécessaires à l'utilisateur pour faire ses tâches (Chadwick, Snyder, & Panda, 2012). Si un utilisateur a besoin temporairement d'un rôle lui donnant plus de droits, il faut lui accorder l'élévation de droits seulement pour le temps où il en besoin et lui réattribuer ensuite le rôle inférieur qu'il avait.

12 Références

Bailey, D. (2011, 05 24). *Don't Do Role-Based Authorization Checks; Do Activity-Based Checks.* Consulté le 02 27, 2015, sur Lost Technies: http://lostechies.com/derickbailey/2011/05/24/dont-do-role-based-authorization-checks-do-activity-based-checks/

Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice, 3e éd.

C. Hu, V., Ferraiolo, D., Kuhn, R., Schnitzer, A., Sandlin, K., Miller, R., et al. (2014). *Guide to Attribute Based Access Control (ABAC) Definition and Considerations.* National Institute of Standards and Technology .

Chadwick, J., Snyder, T., & Panda, H. (2012). Programming ASP.NET MVC 4. O'Reilly.

Coyne, E., & R. Weil, T. (2013). ABAC and RBAC: Scalable, Flexible, and Auditable Access Management. *IT Pro, 1520 -9202*(13), pp. 14-16.

Dykstra, T. (2013, 07 30). *Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application*. Consulté le 04 15, 2015, sur ASP.NET: http://www.asp.net/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application

Galloway, J. (2012, 08 30). *SimpleMembership, Membership Providers, Universal Providers and the new ASP.NET 4.5 Web Forms and ASP.NET MVC 4 templates.* Consulté le 02 27, 2015, sur Web Blogs Asp.NET: http://weblogs.asp.net/jongalloway/simplemembership-membership-providers-universal-providers-and-the-new-asp-net-4-5-web-forms-and-asp-net-mvc-4-templates

Jin, X. (2014). *ATTRIBUTE-BASED ACCESS CONTROL MODELS AND IMPLEMENTATION IN CLOUD INFRASTRUCTURE AS A SERVICE.* San Antonio: Graduate Faculty of The University of Texas.

Jordan, C. S. (1987). *Guide to Understanding Discretionary Access Control in Trusted Systems.* Fort George G. Meade: National Computer Security Center.

Microsoft . (2015). *Migrations Code First*. Consulté le 04 16, 2015, sur MSDN: https://msdn.microsoft.com/fr-ca/data/jj591621.aspx

Microsoft. (s.d.). *ActiveDirectoryMembershipProvider Class.* Consulté le 02 27, 2015, sur MSDN: https://msdn.microsoft.com/en-us/library/system.web.security.activedirectorymembershipprovider%28v=vs.110%29.aspx

Microsoft MSDN. (2015). *Add-ins and Extensibility*. Consulté le 04 17, 2015, sur MSDN: https://msdn.microsoft.com/en-us/library/bb384200(v=vs.110).aspx

Milette-Lacombe, S. (2015). *Sécurité du système BIMWeb, Analyse des besoins.* Montréal: Département de génie logiciel et des T.I. de l'École de technologie Supérieure.

NIST. (2014). *ROLE BASED ACCESS CONTROL - FREQUENTLY ASKED QUESTIONS.* Consulté le 02 27, 2015, sur NIST: http://csrc.nist.gov/groups/SNS/rbac/faq.html

Post, M. (2010, 01 09). *NIST RBAC Data Model.* Consulté le 02 27, 2015, sur Mint IT: https://www.mind-it.info/nist-rbac-data-model/

Rastogi, P., Anderson, R., Dykstra, T., & Galloway, J. (2013, 10 17). *Introduction to ASP.NET Identity*. Consulté le 03 27, 2015, sur ASP.NET:

http://www.asp.net/identity/overview/getting-started/introduction-to-aspnet-identity

Ravi, S. S., Edward, J. C., Hal, L. F., & Charles, E. Y. (1996, 02). Role-Based Access Control Models. *IEEE Computer, 29*(2), pp. 38-47.

SANS Technology Institute. (s.d.). *Role Based Access Control to Achieve Defense in Depth.* Consulté le 02 27, 2015, sur SANS Technology Institute: http://www.sans.edu/research/security-laboratory/article/311

Singh, R. R. (2013, 11 29). *Understanding and Using Simple Membership Provider in ASP.NET MVC 4.0.* Consulté le 02 27, 2015, sur CodeProject: http://www.codeproject.com/Articles/689801/Understanding-and-Using-Simple-Membership-Provider

Wasson, M. (2014, 06 16). *Use Code First Migrations to Seed the Database*. Consulté le 04 2015, 16, sur ASP.NET.

Wloch, S. (2015, 02 13). *Custom Roles Based Access Control (RBAC) in ASP.NET MVC Applications.* Consulté le 02 26, 2015, sur CodeProject: http://www.codeproject.com/Articles/875547/Custom-Roles-Based-Access-Control-RBAC-in-ASP-NET

Woschek, M. (2015, 02 11). *OWASP Cheat Sheets.* Consulté le 02 27, 2015, sur Owasp: https://www.owasp.org/images/9/9a/OWASP_Cheatsheets_Book.pdf

Visualisateur 3D Web

Un portail BIM

Version : 1.0
Date d'émission : 6 mai 2014
Date de révision :

DOCUMENT DE VISION

Auteurs et propriétaires
de la propriété intellectuelle :
Mathieu DUPUIS - ÉTS

Superviseur :
Prof. Alain APRIL - ÉTS

Collaborateur (client)
Présenté à : Ivanka Iordonova

Historique des révisions

Date	Version	Description	Auteur
2014-04-28	0.1	Première Version	Mathieu Dupuis
2014-05-03	0.5	Revue	Alain April
2014-05-03	0.6	Correction selon commentaires reçus	Mathieu Dupuis
2014-06-17	1.0	Correction selon commentaires du client	Mathieu Dupuis

Table des matières

1. Introduction 7
1.1 Objectif 7
1.2 Portée 7
1.3 Définitions, acronymes et abréviations 7
1.4 Description sommaire du produit 7
1.5 Références 8
2. Positionnement 8
2.1 Opportunités d'affaires 8
2.2 Énoncé du problème 8
2.3 Positionnement du produit 8
3. Descriptions des intervenants et des utilisateurs 10
3.1 Résumé des intervenants 10
3.2 Vue d'ensemble des utilisateurs prévus 12
3.3 Environnement physique 13
3.4 Principaux besoins fonctionnels des intervenants et utilisateurs 14
4. Vue d'ensemble du produit 17
4.1 Perspective du produit 17
4.2 Principaux avantages prévus 18
4.3 Hypothèses et dépendances 19
4.4 Prévisions de coûts et de prix 19
4.5 Licences et installation 19
5. Caractéristiques du produit 19
5.1 CAR01 – Avoir un rendu 3D 19
5.2 CAR02 – Naviguer dans le modèle 3D 19
5.3 CAR03 – Sélectionner un ou plusieurs objets objet d'un modèle 3D 19
5.4 CAR04 – Voir les propriétés d'un objets d'un modèle 3D 20
5.5 CAR05 – Sélectionner plusieurs objets selon un filtre 20
5.6 CAR06 – Créer un RFI associé à un ou plusieurs objets 20
5.7 CAR07 – Avoir un portail de projet 20
5.8 CAR08 – Avoir un Add-in dans Revit pour envoyer au portail 20
5.9 CAR09 – Voir le détail d'une demande 20
5.10 CAR010 – Afficher le modèle selon la discipline 20

5.11 CAR011 – Afficher le modèle selon le niveau 20
5.12 CAR012 – Afficher le modèle selon la section 20
5.13 CAR013 – Afficher le modèle selon l'horaire de travail 20
5.14 CAR014 – Joindre un fichier à une demande 20
6. Contraintes techniques et contraintes du projet 21
7. Critères de qualité (exigences non fonctionnelles) 21
8. Attributs des caractéristiques 22
9. Autres exigences du produit 24
9.1 Standards applicables 24
9.2 Exigences du système 24
9.3 Exigences de performance 24
9.4 Exigences environnementales 24
10. Exigences de documentation 24
10.1 Manuel de l'utilisateur 24
10.2 Aide en ligne 24
10.3 Guides d'installation, de configuration, et fichier à lire 24
Annexe 25
Annexe A - Attributs des caractéristiques 25

Liste des tableaux

Tableau 1: Liste des définitions, acronymes et abréviations 7
Tableau 2: Énoncé du problème 8
Tableau 3: Positionnement du produit 9
Tableau 4: Intervenants du projet 11
Tableau 5: Liste des utilisateurs du système 12
Tableau 6: Liste des besoins des intervenants du projet 16
Tableau 7: Matrice des besoins répondus par caractéristique 18
Tableau 8: Attributs des caractéristiques du système 23

Liste des figures

Figure 1: Perspective du produit..... 17

1. Introduction

1.1 Objectif

L'objectif de ce document est d'identifier la portée fonctionnelle du projet de Visualisateur 3D web afin d'offrir une vue d'ensemble des fonctionnalités et des besoins exprimés par les différents intervenants du projet.

1.2 Portée

Ce document couvre la réalisation d'un portail web permettant d'afficher un modèle 3D et d'un module pour y lier la gestion documentaire.

1.3 Définitions, acronymes et abréviations

Terme	Définition
Add-In	Plugiciel pour Revit
BIM	Building Information Modeling
Clash	Chevauchement entre deux disciplines dans le modèle BIM
CMIC	Logiciel de gestion des ressources de l'entreprise qui contient les demandes
Contrainte	Un problème bloquant, lors de la construction ou la modélisation, qui doit être résolu à l'aide d'une demande.
Discipline	Corps de métier dans la construction, tel que : Architecture, Structure, Plomberie.
Demande	Submittal / RFI / RFQ
MEP	Mécanique / Électrique / Plomberie
Modèle BIM	Maquette numérique 3D composée d'objet intelligent et paramétrable.
Module	Librairie de code permettant d'étendre les fonctionnalités du portail web.
Objet	Représentation paramétrable d'un objet physique dans un modèle BIM, telle que : une porte, un mur, une fenêtre, etc.
RFI	Demande d'information (Request For Information)
RFQ	Demande de soumission (Request For Quote)
Section	Une partie d'un plan
Submittal	Communication diverse sur le projet

Tableau 1: Liste des définitions, acronymes et abréviations

1.4 Description sommaire du produit

Le produit logiciel visé consiste en un portail web qui permettra à une entreprise du domaine de la construction de publier leur modèle BIM fait avec l'aide du logiciel Revit dans un portail web. Ce portail permettra à l'utilisateur de consulter seulement l'information nécessaire à son

travail. Lors d'une première itération, ce portail web sera restreint à quelques utilisateurs, soit les coordonnateurs et surintendants de projets. Ce logiciel permettra aux utilisateurs de localiser les documents de projet dans un modèle 3D. Ceci permettra d'avoir une meilleure vue d'ensemble de l'état du projet de construction et de mieux gérer la liste des contraintes du chantier. Typiquement, une contrainte identifiée sera résolue suite à une demande à un ou plusieurs fournisseurs de services impliqués. La résolution d'une contrainte permet de continuer la construction du bâtiment. Le visualisateur 3D mettra en évidence visuellement les sections qui font l'objet de demandes en cours (selon leur état) pour que le responsable du chantier puisse prendre les décisions qui s'imposent pour accélérer le travail.

1.5 Références

Modèle vision : Modèle fournit par Alain April
Revit : Autodesk. 2014. « Building Design Software | Revit Family | Autodesk ». < www.autodesk.com/products/autodesk-revit-family/overview >. Consulté le 2014-05-06.
Unity3D : Technologies, Unity. 2014. « Unity - Game Engine ». < https://unity3d.com >. Consulté le 2014-05-06.

2. Positionnement

2.1 Opportunités d'affaires

Développer un composant utilisable par tous les utilisateurs de Revit.

2.2 Énoncé du problème

Le problème	L'utilisation d'une liste papier (ou un chiffrier Excel sur tablette) des demandes est inefficace, car la fouille des données, les tris et la présentation visuelle ne permettent pas de bien cerner et visualiser les contraintes et leur état envers leurs résolutions.
affecte	L'avancement de la construction d'un projet
dont l'impact est	La gestion inefficace des demandes qui retarde le début, ou la poursuite, de la construction d'une section
Une bonne solution serait	De localiser les demandes (contraintes) à l'aide d'un modèle 3D afin d'aider à les localiser et à voir l'état de la résolution dans un modèle 3D

Tableau 2: Énoncé du problème

2.3 Positionnement du produit

Pour	Les entreprises de construction qui utilisent Revit
Qui	Cherchent à améliorer leur gestion de contraintes

Le Visualisateur 3D Web	Est un portail web de visualisation 3D
Qui	Permet d'afficher un modèle directement à partir d'une connexion internet. Permet d'afficher les demandes
Contrairement à	Chercher dans une liste de demandes une à une en ayant juste un code de localisation.
Notre produit logiciel	Doit avoir une interface simple pour être utilisé qui facilite la visualisation rapide des contraintes. Doit être extensible par l'intermédiaire de module externe.

Tableau 3: Positionnement du produit

3. Descriptions des intervenants et des utilisateurs

3.1 Résumé des intervenants

Nom	Poste	Responsabilités
Daniel Forgues	Directeur de recherche	• Gérer la chaire de recherche Pomerleau.
Alain April	Co-Directeur de recherche	• Superviser les étudiants informatiques de la chaire de recherche Pomerleau; • Valider le bon déroulement de la réalisation du projet.
Ivanka Iordanova	Directrice BIM	• Gérer l'intégration du BIM et la R&D; • Planifier et suivre la progression des travaux BIM sur les projets; • Vérifier la cohérence de la coordination BIM.
Éric Lessard	Directeur des technologies de l'information	• Gérer la programmation et le support informatique; • Gérer le réseau informatique; • Gérer les télécommunications.
Fernando Valdivieso	Spécialiste sénior intégration BIM & LEAN	• Fusionner les différents modèles pour un projet; • Identifier et gérer les clashs; • Coordonner les réunions; • Faire l'intermédiaire entre tous les intervenants du projet; • Faire la coordination intégrée de toutes les disciplines; • Produire de la documentation pour la construction et le suivi du chantier / de la construction; • Aider le développement de la vision future du BIM et du LEAN Construction; • Promouvoir la notion de LEAN Construction;

Nom	Poste	Responsabilités
Simon Brunet	Coordonnateur BIM	• Fusionner les différents modèles pour un projet; • Identifier et gérer les clashs; • Coordonner plusieurs réunions; • Faire l'intermédiaire entre tous les intervenants du projet; • Faire la coordination intégrée de toutes les disciplines; • Produire de la documentation pour la construction et le suivi du chantier / de la construction.
Éric Lacelle	Coordonnateur BIM	• Fusionner les différents modèles pour un projet; • Identifier et gérer les clashs; • Coordonner plusieurs réunions; • Faire l'intermédiaire entre surintendants et les gestionnaires de projets; • Mélange de coordinateurs MEP et BIM.

Tableau 4: Intervenants du projet

3.2 Vue d'ensemble des utilisateurs prévus

Titre	Responsabilités	Intervenant
Surintendant	• Responsable du chantier; • Faire la discipline sur le chantier; • Gérer les ressources humaines et matérielles; • Gérer la sécurité sur le chantier; • Gérer le personnelle en force propre (interne); • Gérer les problèmes du chantier; • Consulter les demandes pour savoir s'il peut débuter les travaux.	Suggestion : Patrick Mercier, Francis Martin
Coordonnateur	• Créer et gérer les demandes; • Faire la planification des échéanciers; • Faire le suivi de chantier (les délais, matériaux); • Organiser les réunions chantiers; • Surveiller la conformité du programme LEED; • Entre le surintendant et les gestionnaires de projets.	Éric Lacelle Simon Brunet
Coordonnateur MEP	• Même responsabilité que le coordonnateur, mais spécialisé MEP; • Charger de la mise en service.	Éric Lacelle Simon Brunet
Coordonnateur BIM	• Fusionner les maquettes; • Faire le modèle 4D; • Faire les réunions coordination (pour le 4D et 3D); • Gérer l'intégration et l'interaction avec la nouvelle technologie; • Assurer les suivis avec les intervenants du projet; • Gérer les clashs avec les outils BIM.	Éric Lacelle Simon Brunet
Professionnel	• Créer des plans conformes aux attentes • Construire selon les plans soumis	Éric Lacelle Simon Brunet
Gérant de projet	• Coordonne et gère les sous-traitants	Aucun
Sous-traitant	• Construire selon les plans soumis	Aucun

Tableau 5: Liste des utilisateurs du système

3.3 Environnement physique

Bureau

Ce milieu est calme avec un bureau et chaise désignés pour l'utilisateur. L'environnement est sans intempérie, poussière et bris.

Roulotte

Ce milieu est relativement calme avec un espace restreint pour le nombre d'utilisateurs. L'environnement est sans intempérie et à risque réduit de bris, mais est propice à la poussière.

Chantier

Ce milieu est bruyant sans place pour s'assoir ou pour utiliser un ordinateur. L'utilisation des tablettes électroniques est plus appropriée. L'environnement est propice aux intempéries, à la poussière et au bris.

3.4 Principaux besoins fonctionnels des intervenants et utilisateurs

Besoin	Priorité	Préoccupations	Solution actuelle	Solution proposée
B01 – Partager le modèle 3D	7	Tous les intervenants dans un projet doivent avoir accès au même modèle.	Partage de fichier sous divers système (FTP, Box, Dropbox)	Aucune
B02 – Voir rapidement les contraintes associées à un secteur	4	La construction d'une section doit être commencée le plus rapidement possible et pour ce faire ils doivent résoudre les contraintes de cette section le plus rapidement possible.	L'utilisateur utilise la case localisation dans la liste des demandes de CMIC pour ordonner la liste des demandes à traiter.	L'utilisateur voit un indicateur sur tous les objets affectés par au moins une contrainte.
B03 – Lier les demandes au plan 2D	5	Plusieurs intervenants n'utilisent que les plans 2D pour travailler.	Les intervenants interagissent directement avec un plan 2D papier ou à l'aide d'outils smart-use. Dans CMIC, le numéro de plan est inscrit dans un champ désigné.	Les intervenants interagissent directement avec un modèle 3D qui reflète les changements dans les modèles 2D.

Besoin	Priorité	Préoccupations	Solution actuelle	Solution proposée
B04 – Fusionner plusieurs modèles de différent modèle	2	Pour un même bâtiment, il y a plusieurs modèles et certains utilisateurs ont besoin de voir plusieurs plans simultanément.	Les modèles sont fusionnés sous Revit et/ou Naviswork	Les modèles sont envoyés individuellement au serveur et le portail permet d'en afficher plusieurs à la fois.
B05 – Naviguer facilement dans un plan ou modèle	3	L'utilisateur doit facilement retrouver la section du bâtiment qu'il recherche	L'utilisateur parcourt les plans papier de la construction visuellement. Dans d'autres cas, il parcourt le dossier de plan (format PDF) ou navigue dans le modèle Revit/Naviswork via un clavier et une souris.	L'utilisateur peut naviguer dans le modèle via le portail soit par les RFI, soit en utilisant le clavier/souris, écran tactile ou un contrôleur externe tel qu'un contrôleur jeu XBOX.
B06 – Connaitre l'avancement des travaux selon l'échéancier.	8	L'utilisateur doit facilement visualiser l'évolution des travaux	L'utilisateur peut utiliser Naviswork pour savoir l'avancement des travaux à une date précise.	Aucune solution proposée.
B07 – Voir les contraintes associées à une tâche.	9	L'utilisateur doit gérer son horaire et doit savoir la cause des retards	Aucune	Aucune solution proposée.

Besoin	Priorité	Préoccupations	Solution actuelle	Solution proposée
B08 – Voir les caractéristiques des différents objets d'un plan.	6	L'utilisateur doit connaitre les caractéristiques des différents objets d'un plan pour choisir les bons matériaux.	Les caractéristiques importantes sont localisées dans le plan 2D. Dans le cas du modèle Revit, toutes les caractéristiques peuvent être ressorties.	Les caractéristiques importantes seront affichées dans l'écran principal. La liste des caractéristiques sera fournie sur demande.
B09 – Visualisation du modèle 3D	1	Tous les intervenants dans un projet doivent être capables de consulter le modèle 3D.	Les intervenants partagent un modèle et le visualisent à l'aide de leur logiciel (si leur logiciel est compatible) ou l'entreprise de construction développe une version compatible à leur logiciel.	La visualisation du modèle se fera directement dans le portail Web évitant les erreurs de compatibilité.

Tableau 6: Liste des besoins des intervenants du projet

4. Vue d'ensemble du produit

4.1 Perspective du produit

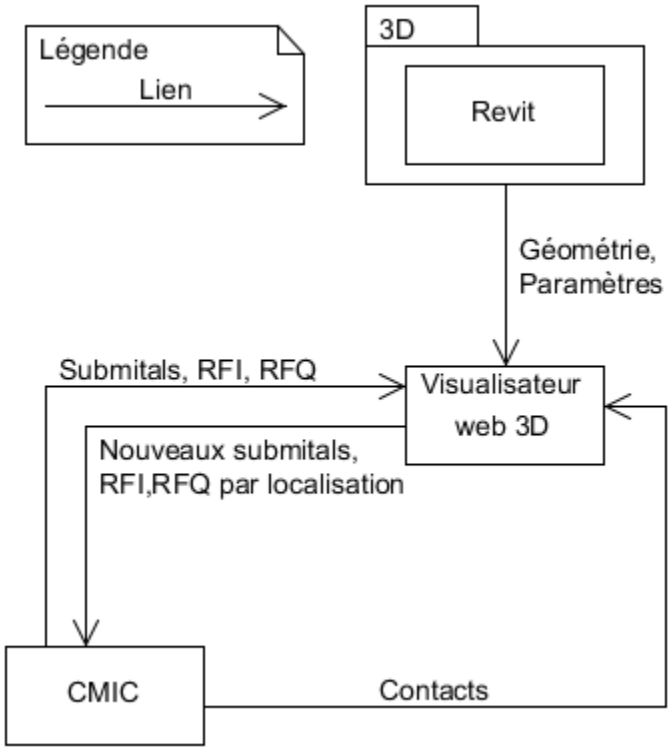

Figure 1: Perspective du produit

4.2 Principaux avantages prévus

Bénéfices / Caractéristiques	CAR 01	CAR 02	CAR 03	CAR 04	CAR 05	CAR 06	CAR 07	CAR 08	CAR 09	CAR 10	CAR 11	CAR 12	CAR 13	CAR14
B01 – Partager le modèle 3D							X	X						
B02 – Voir rapidement les contraintes associées à un secteur	X	X	X		X	X			X					X
B03 – Lier les demandes au plan 2D			X											
B04 – Fusionner plusieurs modèles de différent modèle										X	X	X	X	
B05 – Naviguer facilement dans un plan ou modèle		X												
B06 – Connaitre l'avancement des travaux selon l'échéancier.														
B07 – Voir les contraintes associées à une tâche.									X				X	X
B08 – Voir les caractéristiques des différents objets d'un plan.			X	X										
B09 – Visualisation du modèle 3D	X	X					X	X						

Tableau 7: Matrice des besoins répondus par caractéristique

4.3 Hypothèses et dépendances

HYP01 – L'ÉTS fournis le matériel informatique nécessaire à la réalisation du projet
HYP02 – Pomerleau permet l'utilisation de leur programme de modélisation 3D pour la réalisation du projet.
HYP03 – Pomerleau permet l'utilisateur de leurs fichiers de projets BIM afin de tester le projet avec des données réels.
HYP04 – Pomerleau fournis les services web permettant d'obtenir une liste des différentes demandes pour un projet.
HYP05 – Pomerleau fournis les services web permettant d'obtenir les informations sur les différentes demandes.
HYP06 – Pomerleau fournis les services web permettant de créer et modifier les différentes demandes.
HYP07 – On suppose que les utilisateurs ont accès à internet.
HYP08 – On suppose que les technologies d'affichage 3D sur le web sont suffisamment efficaces et évoluées pour l'affichage d'un modèle 3D BIM.
HYP09 – On suppose que un serveur SQL standard peut gérer les données générer par les modèle 3D BIM.

4.4 Prévisions de coûts et de prix

Le logiciel est développé par les auteurs sans frais pour Pomerleau.

4.5 Licences et installation

L'ÉTS concède à Pomerleau une licence d'utilisation du Logiciel à des fins internes uniquement. Cette Licence sera gratuite, non exclusive, incessible et pour une durée de trois (3) ans à compter de la date de la signature de cette entente (ci-après « Période d'utilisation »).

5. Caractéristiques du produit

Cette section décrit les principales caractéristiques du logiciel proposé.

5.1 CAR01 – Avoir un rendu 3D

Permettre à l'utilisateur d'avoir un rendu 3D d'un modèle directement dans le portail web.

5.2 CAR02 – Naviguer dans le modèle 3D

Permettre à l'utilisateur de naviguer dans le modèle 3D de la même manière que proposée actuellement par Autodesk ou via un contrôleur externe.

5.3 CAR03 – Sélectionner un ou plusieurs objets objet d'un modèle 3D

Permettre à l'utilisateur de sélectionner un objet du modèle et le mettre en surbrillance. À l'aide d'une combinaison de touches, l'utilisateur peut étendre sa sélection à d'autres objets.

5.4 CAR04 – Voir les propriétés d'un objet d'un modèle 3D

Permettre à l'utilisateur de voir la liste exhaustive de tous les paramètres contenus dans un objet du modèle.

5.5 CAR05 – Sélectionner plusieurs objets selon un filtre

Permettre à l'utilisateur de sélectionner plusieurs objets selon un filtre basé sur un paramètre des objets.

5.6 CAR06 – Créer un RFI associé à un ou plusieurs objets

Permettre à l'utilisateur de créer un RFI associé à un ou plusieurs objets. Lors de la création, le logiciel le transmettra à CMIC et enverra un courriel aux destinataires.

5.7 CAR07 – Avoir un portail de projet

Permettre à l'utilisateur d'avoir plusieurs projets accessibles simultanément dans le portail.

5.8 CAR08 – Avoir un Add-in dans Revit pour envoyer au portail

Permettre à l'utilisateur d'avoir un 'add-in' qui transfère la géométrie et les paramètres du modèle vers le portail. Cet 'add-in' transforme les données de Revit dans un format normalisé compatible avec le portail.

5.9 CAR09 – Voir le détail d'une demande

Permettre à l'utilisateur de voir le détail d'une demande en tout temps. Tous les détails contenus dans CMIC sont aussi disponibles dans le portail. Les modifications faites dans cette demande seront reportées vers CMIC.

5.10 CAR010 – Afficher le modèle selon la discipline

Permettre à l'utilisateur d'afficher un ou plusieurs modèles 3D dynamiquement. Ces modèles correspondent aux différents modèles fournis par les différentes disciplines de la construction.

5.11 CAR011 – Afficher le modèle selon le niveau

Permettre à l'utilisateur d'afficher le modèle selon un niveau sélectionné par l'utilisateur.

5.12 CAR012 – Afficher le modèle selon la section

Permettre à l'utilisateur d'afficher le modèle selon un secteur défini dans le modèle.

5.13 CAR013 – Afficher le modèle selon l'horaire de travail

Permettre à l'utilisateur de visualiser les objets qui seront construits ou en construction selon un délai précis.

5.14 CAR014 – Joindre un fichier à une demande

Permettre à l'utilisateur de joindre un fichier à une demande afin de lui permettre de joindre des photographies des problèmes détectés.

6. Contraintes techniques et contraintes du projet

Toutes les communications entre le visualisateur et CMIC sont dépendantes du département des Technologies de l'information. Leur coopération sera nécessaire pour :

- La création d'un service web pour obtenir les informations d'une demande.
- La création d'un service web pour créer une demande.
- La création d'un service web pour modifier une demande.
- La création d'un service web pour obtenir une liste de demande pour un projet.

7. Critères de qualité (exigences non fonctionnelles)

CN01 – Le portail devra être accessible à partir d'un ordinateur de bureau ou d'une tablette mobile (Android, Ipad et Surface).

CN02 – Les données devront être sécurisées.

8. Attributs des caractéristiques

Cette section présente une synthèse des caractéristiques du logiciel selon les bénéfices qu'ils apportent au client, l'effort requis pour les implémenter, le risque relié à leur implémentation et leur stabilité (probabilité de changement). La signification de la valeur de ces attributs est présentée à l'annexe A.

Caractéristiques	État	Bénéfice	Effort	Risque	Stabilité
CAR01 – Avoir un rendu 3D	Proposé	Critique	Moyen	Moyen	Haute
CAR02 – Naviguer dans le modèle 3D	Proposé	Critique	Bas	Élevé	Basse
CAR03 – Sélectionner un ou plusieurs objets objet d'un modèle 3D	Proposé	Critique	Moyen	Élevé	Haute
CAR04 – Voir les propriétés d'un objet d'un modèle 3D	Proposé	Important	Bas	Bas	Haute
CAR05 – Sélectionner plusieurs objets selon un filtre	Proposé	Important	Haut	Moyen	Moyen
CAR06 – Créer un RFI associé à un ou plusieurs objets	Proposé	Critique	Moyen	Élevé	Haute
CAR07 – Avoir un portail de projet	Proposé	Important	Moyen	Faible	Haute
CAR08 – Avoir un Add-in dans Revit pour envoyer au portail	Proposé	Critique	Haut	Moyen	Moyen
CAR09 – Voir le détail d'une demande	Proposé	Important	Bas	Élevé	Haute
CAR10 – Afficher le modèle selon la discipline	Proposé	Critique	Moyen	Élevé	Basse
CAR11 – Afficher le modèle selon le niveau	Proposé	Utile	Bas	Élevé	Haute
CAR12 – Afficher le modèle selon la section	Proposé	Utile	?	Élevé	Basse
CAR13 – Afficher le modèle selon l'horaire de travail	Proposé	Utile	Haut	Élevé	Basse

Caractéristiques	État	Bénéfice	Effort	Risque	Stabilité
CAR14 – Joindre un fichier à une demande	Proposé	Utile	Bas	Élevé	Basse

Tableau 8: Attributs des caractéristiques du système

9. Autres exigences du produit

9.1 Standards applicables

Aucun

9.2 Exigences du système

9.2.1 Portabilité

Le système doit fonctionner sur différentes plateformes : PC, IPad, Android Tablet, Windows Surface.

9.3 Exigences de performance

9.3.1 Affichage du modèle 3D

L'affichage du modèle 3D doit être, au minimum, aussi vite que dans Autodesk Glue.

9.3.2 Rendu du modèle 3D

L'affichage du modèle 3D doit se faire au moins à un taux de rafraichissement de 4 images par seconde.

9.4 Exigences environnementales

9.4.1 Serveur web

Le projet nécessite un serveur web qui est protégé des intempéries, des variations de température et des coupures de courant.

10. Exigences de documentation

10.1 Manuel de l'utilisateur

Aucun manuel de l'utilisateur ne sera fait lors de ce projet.

10.2 Aide en ligne

Aucune aide en ligne ne sera faite lors de ce projet.

10.3 Guides d'installation, de configuration, et fichier à lire

À la fin du projet, le logiciel sera installé et fonctionnel sur un serveur web à l'ÉTS. Les utilisateurs devront installer le composant Unity3D et ils auront accès au visualisateur 3D web via leur fureteur web. Les fureteurs web supporté par le projet seront ceux supportés par Unity3D, soit principalement : Firefox, Chrome et Internet Explorer.

Annexe

Annexe A - Attributs des caractéristiques

Cette section présente la définition des attributs que nous avons associés aux différentes caractéristiques du logiciel au tableau 8.

État

Proposé	Cet état indique que la caractéristique est proposée et doit faire l'objet de discussion au sein de l'équipe de projet en vue de son acceptation ou de son rejet.
Approuvé	Cet état indique que la caractéristique a été retenue pour un prochain développement.
Incorporé	Cet état indique que cette caractéristique a été incorporée au système au cours d'un développement.

Bénéfice

Critique	Ce niveau indique que la caractéristique est primordiale pour le logiciel. Ce qui signifie que le client ne voudra pas avoir un système sans cette caractéristique.
Important	Ce niveau indique que cette caractéristique est importante. Cependant, si elle n'est pas implémentée, le logiciel peut être utilisé. Il revient au client de décider s'il souhaite avoir le logiciel sans cette fonctionnalité ou si le projet s'arrête.
Utile	Le système va intégrer toutes les fonctionnalités importantes pour atteindre le bénéfice prévu. Toutefois si le temps le permet et si l'équipe de projet est disponible, certaines caractéristiques peuvent être ajoutées pour accroître le bénéfice du client.

Effort

Haut	Ce niveau indique que la quantité d'effort requis est d'au moins deux semaines de travail.
Moyen	Ce niveau indique que la quantité d'effort requis est entre une et deux semaines de travail
Bas	Ce niveau indique que la quantité d'effort requis est en dessous d'une semaine de travail.

Risque

Haut	Ce niveau indique qu'il y a un haut taux d'incertitude sur la durée ou coût de l'implémentation, voir même un risque d'annulation.
Moyen	Ce niveau indique qu'il y a une certaine incertitude sur la durée ou le coût de l'implémentation.
Bas	Ce niveau indique que la durée et les coûts sont bien définis et ont peu de chance de changer.

Stabilité

Haut	Ce niveau indique que cette caractéristique ne subira pas de changement. Cela montre aussi qu'elle a été bien comprise.
Moyen	Ce niveau indique que la caractéristique peut subir des changements.
Bas	Ce niveau de stabilité indique que la probabilité que cette caractéristique subisse des changements est élevée.

Annexe II - SRS de BIMWeb

Visualisateur 3D Web

Un portail BIM

Version: 1.0

Date d'émission: 2014-06-17

Date de révision:

System Requirements Specifications

Auteurs :

Mathieu Dupuis

Superviseur :

Alain April

Historique des révisions

Date	Version	Description des révisions	Auteur
2014-04-29	0.0	Début du document	Mathieu Dupuis
2014-05-15	0.1	Commentaires	Alain April
2014-05-22	0.2	Corrections après révision	Mathieu Dupuis
2014-06-17	1.0	Correction suite au commentaire du client	Mathieu Dupuis

Table des matières

1 Introduction 8
1.1 Avant-propos 8
1.2 Objectif 8
1.3 Portée 8
1.4 Références 8
1.5 Hypothèses et dépendances 8
2 Survol du modèle des cas d'utilisation 9
2.1 Diagramme des cas d'utilisation 9
2.2 Cas d'utilisation 10
UC01 - Consulter les paramètres d'un objet 10
UC02 - Consulter une demande 10
UC03 - Gérer une demande 11
UC04 - Créer une demande 12
UC05 - Transférer un modèle 13
3 Acteurs 14
AC01 - Surintendant 14
AC02 - Coordonnateur 14
AC03 – Professionnel 14
AC04 – Sous- traitant 14
AC05 – Gérant de projet 14
4 Exigences 14
4.1 Exigences fonctionnelles 14
REQ1XX – Exigences générales 14
REQ4XX – Exigences concernant la sécurité du système 15
REQ5XX – Exigence pour le transfert de modèle 15
REQ6XX – Exigences de visualisation 15
REQ7XX – Module de gestion documentaire 16

REQ8XX – Services Web pour la gestion documentaire 16
4.2 Exigences non fonctionnelles. 17
4.2.1 Fiabilité 18
4.2.2 Facilité d'utilisation 18
4.2.3 Maintenabilité 18
4.2.4 Portabilité 18
4.2.5 Nombre d'utilisateurs 18
4.2.6 Traçabilité 19
4.2.7 Performance 19
5 Documentation pour l'utilisateur et exigence du système d'aide 19
6 Contraintes de conception. 19
7 Interfaces 20
7.1 Interfaces utilisateurs 20
7.1.1 Page d'authentification 20
7.1.2 Page de projets 21
7.1.3 Page de visualisation du modèle 3D 22
7.2 Interfaces systèmes 23
7.2.1 Add-in transfert de modèle 23
7.2.2 Visualisateur 23
7.2.3 Module de gestion documentaire 24
7.2.4 CMIC 24
8 Standards applicables 24
8.1 JSON 24
8.2 Format des dates 24

Table des figures

Figure 1 : Diagramme des cas d'utilisation 9
Figure 2 : Page d'authentification 20
Figure 3 : Page de projets 21
Figure 4 : Visualisation du modèle 3D 22
Figure 5 : Sélection d'un objet dans le modèle 3D 22
Figure 6 : Diagramme des interfaces systèmes 23

Table des tableaux

Tableau 1 : Glossaire 7

Glossaire

Terme	Définition
Add-In	Plugiciel pour Revit
BIM	Building Information Modeling
Clash	Chevauchement entre deux disciplines dans le modèle BIM
CMIC	Logiciel de gestion des ressources de l'entreprise qui contient les demandes
Contrainte	Un problème bloquant, lors de la construction ou la modélisation, qui doit être résolu à l'aide d'une demande.
Discipline	Corps de métier dans la construction, tel que : Architecture, Structure, Plomberie.
Demande	Submittal / RFI / RFQ
MEP	Mécanique / Électrique / Plomberie
Modèle BIM	Maquette numérique 3D composée d'objet intelligent et paramétrable.
Module	Librairie de code permettant d'étendre les fonctionnalités du portail web.
Objet	Représentation paramétrable d'un objet physique dans un modèle BIM, telle que : une porte, un mur, une fenêtre, etc.
RFI	Demande d'information (Request For Information)
RFQ	Demande de soumission (Request For Quote)
SDK	Trousse de développement logiciel (Software Development Kit)
Section	Une partie d'un plan
Submittal	Communication diverse sur le projet

Tableau 1 : Glossaire

1 Introduction

1.1 Avant-propos

Ce document sera raffiné lors d'itérations successives. Tous les éléments en rouge de ce rapport ne seront pas réalisés dans la prochaine itération de développement de la plateforme d'intégration de logiciels du domaine de la construction..

1.2 Objectif

L'objectif de ce document est d'identifier le comportement du système de Visualisation 3D web afin d'offrir une vue d'ensemble de la plateforme d'intégration à développer.

1.3 Portée

Ce document couvre la réalisation d'un portail web permettant d'afficher un modèle 3D et d'un module pour y lier la gestion documentaire.

1.4 Références

Modèle SRS : Modèle fourni par Alain April

Revit : Autodesk. 2014. « Building Design Software | Revit Family | Autodesk ». <www.autodesk.com/products/autodesk-revit-family/overview>. Consulté le 2014-05-06.

Unity3D : Technologies, Unity. 2014. « Unity - Game Engine ». <https://unity3d.com>. Consulté le 2014-05-06.

1.5 Hypothèses et dépendances

HYP01 – L'ÉTS fournit le matériel informatique nécessaire à la réalisation du projet
HYP02 – Pomerleau, en tant que pilote des essais, permet l'utilisation de leur programme de modélisation 3D pour la réalisation du projet.
HYP03 – Pomerleau permet l'utilisateur de leurs fichiers de projets BIM afin de tester le projet avec des données réelles.
HYP04 – Pomerleau fournit les services web permettant d'obtenir une liste des différentes demandes pour un projet.
HYP05 – Pomerleau fournit les services web permettant d'obtenir les informations sur les différentes demandes.
HYP06 – Pomerleau fournit les services web permettant de créer et modifier les différentes demandes.
HYP07 – On suppose que les utilisateurs ont accès à internet.
HYP08 – On suppose que les technologies d'affichage 3D sur le web sont suffisamment efficaces et évoluées pour l'affichage d'un modèle 3D BIM.
HYP09 – On suppose qu'un serveur SQL standard peut gérer les données générées par les modèles 3D BIM.

2 Survol du modèle des cas d'utilisation

2.1 Diagramme des cas d'utilisation

Ce diagramme présente l'interaction entre les différents acteurs et le visualisateur web 3D.

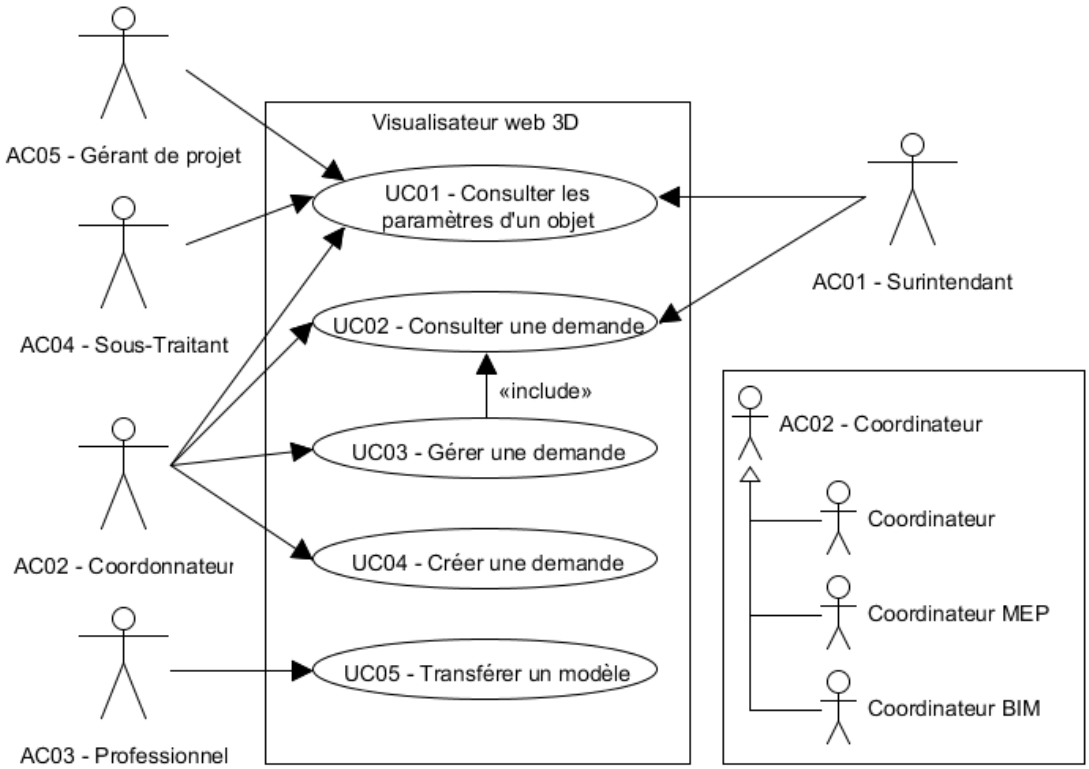

Figure 1 : Diagramme des cas d'utilisation

2.2 Cas d'utilisation

UC01 - Consulter les paramètres d'un objet

Le système permet de consulter tous les paramètres d'un objet dans le modèle 3D.

Acteur principal:

- Surintendant ou coordinateur.

Précondition

- L'utilisateur est authentifié au système;
- Le portail contient au moins un projet qui contient au moins un modèle;
- L'utilisateur est dans un modèle.

Postcondition

- La liste de paramètres pour l'objet est affichée à l'écran.

Flux principal

1. L'utilisateur sélectionner un objet du modèle.
2. Le système met en surbrillance l'objet.
3. L'utilisateur demande la liste des paramètres
4. Fin du cas d'utilisation

UC02 - Consulter une demande

Le système permet d'afficher une demande et toutes les informations que celle-ci contient.

Acteur principal:

- Surintendant ou coordinateur.

Précondition

- L'utilisateur est authentifié au système;
- Le portail contient au moins un projet qui contient au moins un modèle;
- L'utilisateur consulte un modèle.

Postcondition

- La demande et tous ses détails sont affichés.

Flux principal

1. L'utilisateur sélectionner un objet du modèle avec un indicateur de demande.
2. Le système met en surbrillance l'objet et affiche les demandes associées à l'objet.
3. L'utilisateur demande le détail d'une demande.
4. Le système affiche le détail de la demande
5. Fin du cas d'utilisation

UC03 - Gérer une demande

Les utilisateurs doivent faire le suivi des demandes associées aux objets du modèle 3D

Acteur principal:

- Coordinateur.

Précondition

- L'utilisateur est authentifié au système.
- Le portail contient au moins un projet qui contient au moins un modèle.
- L'utilisateur consulte un modèle.
- Le modèle contient au moins un objet avec une demande

Postcondition

- Les données de la demande sont mises à jour.

Flux Principal

1. Voir « CU02 - Consulter une demande ».
2. L'utilisateur modifie les champs de la demande.
3. Le système enregistre les changements.
4. Le système retourne à la consultation du modèle.
5. Fin du cas d'utilisation

Flux alternatif

3A. La demande a été complétée et elle est la dernière pour un objet.

1. Si c'est la première demande pour un des objets choisis.
2. Le système enlève l'indicateur de demandes sur l'objet.
3. Retour au flux principal à l'étape 4.

UC04 - Créer une demande

L'utilisateur crée une demande pour communiquer avec une personne externe d'une situation.

Acteur principal:

- Coordonnateur.

Précondition

- L'utilisateur est authentifié au système.
- Le portail contient au moins un projet qui contient au moins un modèle.
- L'utilisateur consulte un modèle.

Postcondition

- Une demande est créée pour un ou plusieurs objets du modèle.

Flux Principal

1. L'utilisateur choisit un ou plusieurs objets dans le modèle.
2. Le système met en surbrillance les objets.
3. L'utilisateur créer une demande.
4. Le système lui affiche le formulaire de création de demandes.
5. L'utilisateur choisit le type de demande qu'il veuille créer soit : RFI, RFQ ou Submittal.
6. Le système affiche les champs à remplir pour ce type de demande.
7. L'utilisateur remplit et enregistre sa nouvelle demande.
8. Le système créer la nouvelle demande selon les informations entrées.
9. Le système envoie un courriel au contact choisis.
10. Le système retourne à la consultation du modèle.
11. Fin du cas d'utilisation.

Flux alternatif

8A. Première demande pour un objet.

1. Si c'est la première demande pour un des objets choisis.
2. Le système affiche un indicateur de demandes sur l'objet.
3. Retour au flux principal à l'étape 10.

UC05 - Transférer un modèle

À la complétion de son modèle, le professionnel publie celui-ci sur le portail.

Acteur principal:

- Professionnel.

Précondition

- L'utilisateur à installer le Add-in de transfert dans Revit.
- Un modèle est chargé dans Revit.

Postcondition

- Le modèle est publié dans le portail.

Flux Principal

1. L'utilisateur exécute le Add-in de transfert dans Revit.
2. Le Add-in demande à l'utilisateur de s'authentifier.
3. L'utilisateur s'authentifie.
4. Le Add-in demande de choisir un modèle dans un projet.
5. L'utilisateur choisit le modèle à remplacer.
6. Le système reçoit le modèle du Add-in et remplace le modèle par celui envoyé.
7. Fin du cas d'utilisation.

Flux Alternatif

5A. Transfert d'un nouveau modèle.

1. L'utilisateur choisit de créer un nouveau modèle dans le projet.
2. Le système créer un modèle vide dans le projet.
3. Retour au flux principal à l'étape 6.

5B. Transfert d'un modèle d'un nouveau projet.

1. L'utilisateur choisit de créer un nouveau projet.
2. Le système créer un projet.
3. Retour au flux alternatif 5A.

3 Acteurs

AC01 - Surintendant

Les surintendants sont les personnes qui coordonnent les sites de construction. Leur objectif est de mettre en construction les sections qui n'ont plus de contraintes. Donc leur utilisation du système va se limiter à la consultation du modèle 3D et possiblement la consultation des demandes.

AC02 - Coordonnateur

Les coordonnateurs incluent les coordonnateurs, les coordonnateurs MEP et les coordonnateurs BIM, car leurs tâches effectuées avec le portail web sont le même. Les coordonnateurs sont les utilisateurs principaux du système. Les coordonnateurs se servent du système afin de voir les contraintes à régler pour permettre au surintendant de débuter les travaux.

AC03 – Professionnel

Les professionnels sont des personnes pour la plupart du temps externes à l'entreprise. Sur chaque projet de construction, ils peuvent être représentés par une firme externe différente du précédent projet. Leur rôle est de produire un modèle selon les requis fournis. Les professionnels vont utiliser le système majoritairement pour transférer le modèle.

AC04 – Sous- traitant

Les sous-traitants sont des personnes pour la plupart du temps externes à l'entreprise. Sur chaque projet de construction, ils peuvent être représentés par une firme externe différente du précédent projet. Leur rôle est de construire le bâtiment selon les modèles et plan fournis. Les sous-traitants vont utiliser le système majoritairement pour consulter le modèle 3D afin d'accéder aux submitals, car certains contiennent des plans plus détaillés.

AC05 – Gérant de projet

Les gérants de projets sont des personnes internes à l'entreprise qui gère les sous-traitants du projet. Pour les gérer il aura besoin de tous les accès que les sous-traitants ont accès, afin de vérifier qu'ils suivent bien les directives.

4 Exigences

4.1 Exigences fonctionnelles

REQ1XX – Exigences générales

REQ101 – Créer un projet.
L'utilisateur doit être capable de créer un nouveau projet comportant un nom et les informations nécessaires pour l'identifier. Le système doit créer un dossier de stockage pour contenir les différents modèles.

REQ102 – Lister les projets disponibles pour l'utilisateur.
Le système doit afficher les projets disponibles à l'utilisateur afin qu'il y accède.

REQ103 – Enregistrer le modèle reçu
Lors de la réception d'un modèle pour un projet. Le système le déplace à son emplacement final dans sa hiérarchie de dossier.

REQ4XX – Exigences concernant la sécurité du système

REQ401 – Gérer l'accès aux projets et aux modèles selon l'utilisateur.
REQ402 – Gérer l'utilisation des modules selon l'utilisateur.
REQ403 – Gérer l'accès au portail selon l'utilisateur.
REQ404 – Gérer les droits des utilisateurs pour le système.

REQ5XX – Exigence pour le transfert de modèle

REQ501 – Transférer la géométrie de Revit vers le portail dans un projet.
Le Add-in doit extraire la géométrie du modèle conçue par Revit et l'envoyer au portail. Le Add-in doit envoyer les données en utilisant le format de fichier compatible au logiciel Unity3D.

REQ502 – Transférer les paramètres de la géométrie de Revit vers le portail dans un projet.
Le Add-in doit extraire les paramètres de la géométrie du modèle conçue par Revit et l'envoyer au portail. Le Add-in doit envoyer les données en utilisant la syntaxe JSON.

REQ6XX – Exigences de visualisation

REQ601 – Afficher plusieurs modèles 3D comme étant un modèle 3D unique.
Le système doit afficher être capable d'afficher plusieurs modèles à partir des fichiers correspondants. Les modèles doivent se superposer selon leur géolocalisation. De plus les modèles doit avoir un éclairage omniprésent ce qui permet de voir n'importe quel salle ou parti du plan.

REQ602 – Naviguer dans le modèle 3D en mode « vol ».
Le système doit permettre à l'utilisateur de naviguer dans le modèle selon les contrôles d'Autodesk Revit, soit :
Déplacement avant / arrière :
Déplacement latéral haut / bas / gauche / droite : souris bouton 3 + bouger la sourie
Zoom avant / arrière : scroll up / scroll down
Rotation haut / bas / droite / gauche : souris bouton 3 + shift + bouger la sourie

REQ603 – Naviguer dans le modèle 3D en mode « 3^{e} personne ».
Le système doit permettre à l'utilisateur de naviguer dans le modèle selon les contrôles, soit :
Déplacement avant / arrière / gauche / droite : Touche w / s / a / d
Rotation droite / gauche / haut / bas : déplacement de la souris
Saut : Touche espace

REQ604 – Afficher les modèles en différents modes.
Le système doit permettre à l'utilisateur d'afficher la géométrie selon diférent rendu. Soit les rendu : « Wireframe », et transparant. Le mode « wireframe » permet de voir que les arrêtes des objets. Le mode transparent affiche les objets de façons semi-transparentes, ce qui permet de voir facilement les pièces de différentes couleurs.

REQ605 – Fusionner avec REQ604
REQ606 – Afficher les modèle en mode rendu final

Le système doit permettre à l'utilisateur d'afficher la géométrie en mode rendu final. Le mode rendu final permet d'afficher les objets avec leur texture réelle.

REQ607 – Sélectionner un ou plusieurs objets manuellement.
Le système doit permettre à l'utilisateur d'étendre sa sélection manuellement, par une combinaison de touches et de clics de souris.

REQ608 – Sélectionner plusieurs objets selon leur type ou leur famille.
Le système doit permettre à l'utilisateur de sélectionner tous les objets d'un même type ou d'une même famille à partir d'un simple clic.

REQ609 – Fusionner avec REQ608

REQ610 – Sélectionner plusieurs objets selon un paramètre des objets.
Le système doit permettre à l'utilisateur de sélectionner tous les objet qui ont la même valeur dans un paramètre données.

REQ611 – Colorer les objets sélectionnés.
Le système doit permettre de colorer les objets sélectionné.

REQ7XX – Module de gestion documentaire

*Il n'y a pas de consultation car elle est remplacée par la modification de la demande.

REQ701 – Ajouter un RFI associé à un objet.
REQ702 – Modifier un RFI.
REQ703 – Lister les RFI pour un projet.
REQ704 – Ajouter un submittal associé à un objet.
REQ705 – Modifier un submittal.
REQ706 – Lister les submittal pour un projet.
REQ707 – Ajouter un RFQ associé à un objet.
REQ708 – Modifier un RFQ.
REQ709 – Lister les RFQ pour un projet.

REQ8XX – Services Web pour la gestion documentaire

REQ801 – Envoyer la liste complète des RFI pour un projet contenu dans CMIC
Lors d'une demande de la liste complète des RFI pour un projet, le service web doit retourner au minimum :

- L'identificateur du RFI
- Date requise
- Date créée
- Date dernière réponse
- Sujet
- Statut

REQ802 – Créer un RFI pour un projet dans CMIC
Lors d'une demande de création d'un RFI, le service web doit créer un RFI dans CMIC et d'envoyer le email au personne lier au RFI.
La demande contiendra minimalement :

- L'expéditeur
- Le co-Auteur
- La personne à envoyer la demande
- Les personnes à envoyer une copie conforme.
- Le sujet
- Section
- La question

- La suggestion
- La date requise

La réponse contiendra le numéro de RFI ou -1 en cas d'échec.

REQ803 – Modifier un RFI pour un projet dans CMIC
Lors d'une demande de modification d'un RFI, le service web doit modifier le RFI dans CMIC.
Les données qui peuvent être changé sont au minimum:

- Le statut

REQ804 – Obtenir la liste des statuts disponible pour les RFI de CMIC.
Lors d'une demande de la liste de statuts au service web, le service doit retourner la liste des statuts existant pour les RFI.
L'information minimale nécessaire est :

- L'identificateur du statut
- La description du statut

REQ805 – Envoyer les détails d'un RFI
Lors d'une demande d'obtention des détail d'un RFI, le service doit retourner au minimum les champs suivants d'un RFI :

- L'expéditeur
- Le co-Auteur
- La personne à envoyer la demande
- Les personnes à envoyer une copie conforme.
- Le sujet
- Section
- La question
- La suggestion
- La date requise

REQ806 – Envoyer la liste complète des RFQ pour un projet contenu dans CMIC
REQ807 – Créer un RFQ pour un projet dans CMIC
REQ808 – Modifier un RFQ pour un projet dans CMIC
REQ809 – Obtenir la liste des statuts disponible pour les RFQ de CMIC.
REQ810 – Envoyer les détails d'un RFQ

REQ811 – Envoyer la liste complète des Submital pour un projet contenu dans CMIC
REQ812 – Créer un Submital pour un projet dans CMIC
REQ813 – Modifier un Submital pour un projet dans CMIC
REQ814 – Obtenir la liste des statuts disponible pour les Submital de CMIC.

REQ890 – Obtenir la liste des contacts de CMIC pour un projet
Lors d'une demande des contacts d'un projet au service web, le service web doit retourner la liste des contacts pour ce projet. La liste doit contenir minimalement ces informations :

- L'identificateur (Peut-être c'est le code du contact)
- Le code
- Le nom
- Le numéro du partenaire
- Le nom du partenaire

4.2 Exigences non fonctionnelles.

4.2.1 Fiabilité

- Dans le cas qu'une erreur survienne lors de l'affichage d'un modèle, le système doit être capable de fonctionner normalement pour l'affichage des autres modèles;
- Dans le cas qu'une erreur survient due à un module et le système ne peut continuer à fonctionner, l'utilisateur doit être capable de désactiver ce module dans une nouvelle session;
- Dans le cas qu'une erreur survient et le système ne peut continuer à fonctionner, l'utilisateur doit redémarrer l'application pour continuer à travailler;
- Dans tous les cas où une erreur survient, l'erreur est enregistrer dans le système de trace du serveur.

4.2.2 Facilité d'utilisation

- L'utilisateur doit être capable d'afficher la section voulue en moins de 30 secondes;
- L'utilisateur doit être capable de sélectionner les objets du modèle de façons intuitives;
- L'utilisateur doit être capable d'associer les demandes aux objets de façons intuitives;
- Dans le cas où l'utilisateur doit attendre plus de 5 secondes, le système doit y afficher une barre de progression ou équivalent.

4.2.3 Maintenabilité

- Lorsque d'un développeur veut étendre les fonctionnalités du système, il ne doit pas avoir à modifier le code du portail web, mais seulement créer un nouveau module.

4.2.4 Portabilité

- Le portail doit être utilisable sur PC, tablette Windows, tablette Android et tablette Ipad;
- Le Add-In pour le transfert des modèles doit s'intégrer à Revit 2013 et 2014.

4.2.5 Nombre d'utilisateurs

- Nombre d'utilisateurs prévus sur la BD : 3 (soit un par compagnie et 2 pour le fonctionnement interne);
- Nombre concurrent d'utilisateurs : 1500 (soit 30 par projet) ;
- Le système de visualisation 3D à plusieurs type d'utilisateurs, tel que :
    - Administrateur : A tous les droits dans toutes les compagnies et dans tous les projets;
    - Gestionnaire de projet : A tous les droits pour une compagnie, dont la création ou suppression d'un projet;
    - BIM manager : Ne peut créer un nouveau projet mais il peut créer ou écraser un modèle;
    - Utilisateur : Peut consulter seulement le modèle et utiliser les différents modules.
- Chaque module est responsable de définir des types d'utilisateur selon leurs fonctionnalités.

4.2.6 Traçabilité

- Les traces d'accès au page/service web sont conservées pendant 48 heures;
- Les traces d'erreur survenues dans la plateforme sont conservées pendant 48 heures;
- Les traces de requête SQL sont conservées pendant 48 heures.

4.2.7 Performance

- Le système doit être capable de soutenir 500 transactions par seconde;
- Le système doit être capable de conserver 50 projets simultanément contenant chacun 13 modèles, ce qui nécessite environs 570 gig de disque dur;
- Le système doit être capable de soutenir le transfert de 10 modèles simultanément à une vitesse de 30 mbps;
- L'affichage du modèle 3D doit se faire au moins à un taux de rafraichissement de 4 images par seconde;
- L'affichage du modèle 3D doit être, au minimum, aussi vite que dans Autodesk Glue.

5 Documentation pour l'utilisateur et exigence du système d'aide

Aucune documentation ne sera produite pour l'utilisateur. Par contre une formation sera effectuée avec les utilisateurs.

6 Contraintes de conception.

CR01 – Le système nécessite une connexion constante à Internet et si le service web du système est en panne le système est inaccessible.
CR02 – Le système doit pouvoir interagir avec les systèmes de construction les plus populaires. Pour la première itération ce seront ceux utilisés par Pomerleau.
CR03 – Les données devront être sécurisées.

7 Interfaces

7.1 Interfaces utilisateurs

Les interfaces utilisateurs présentées dans les sous-sections sont des ébauches de ce que le portail final peut ressembler.

7.1.1 Page d'authentification

Figure 2 : Page d'authentification

7.1.2 Page de projets

Figure 3 : Page de projets

7.1.3 Page de visualisation du modèle 3D

Lorsqu'un utilisateur entre dans un projet :

Figure 4 : Visualisation du modèle 3D

Lorsqu'un utilisateur sélectionne un objet :

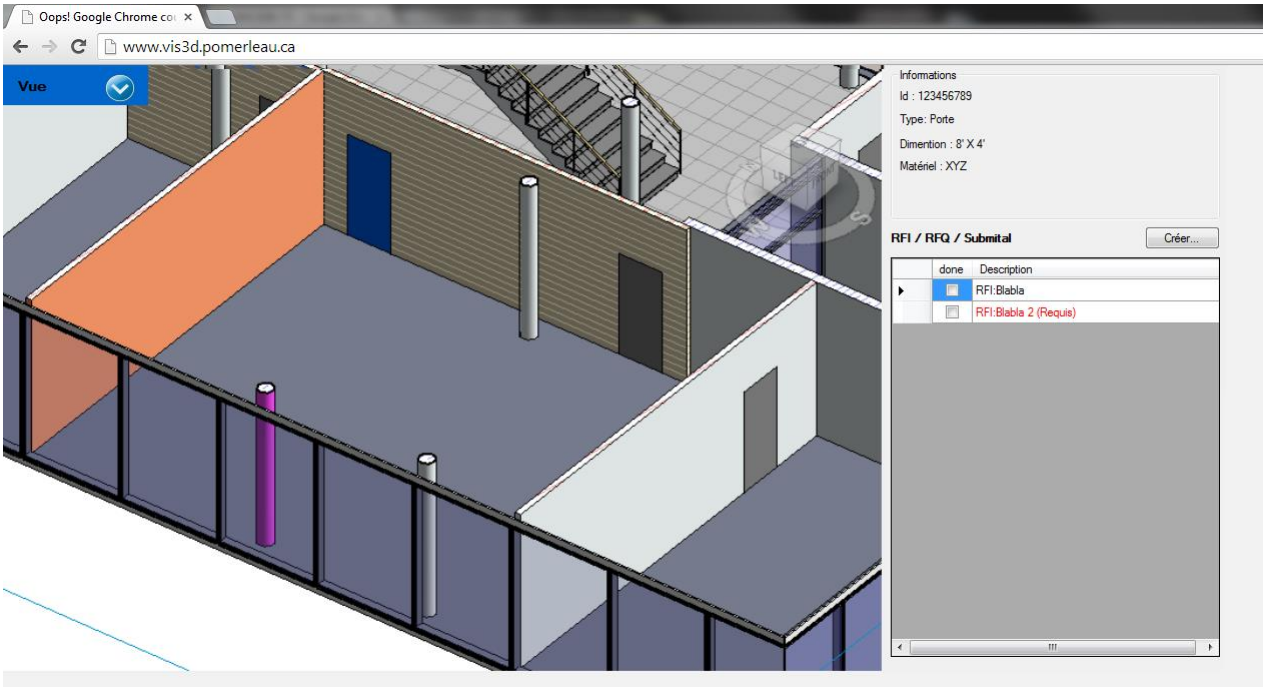

Figure 5 : Sélection d'un objet dans le modèle 3D

7.2 Interfaces systèmes

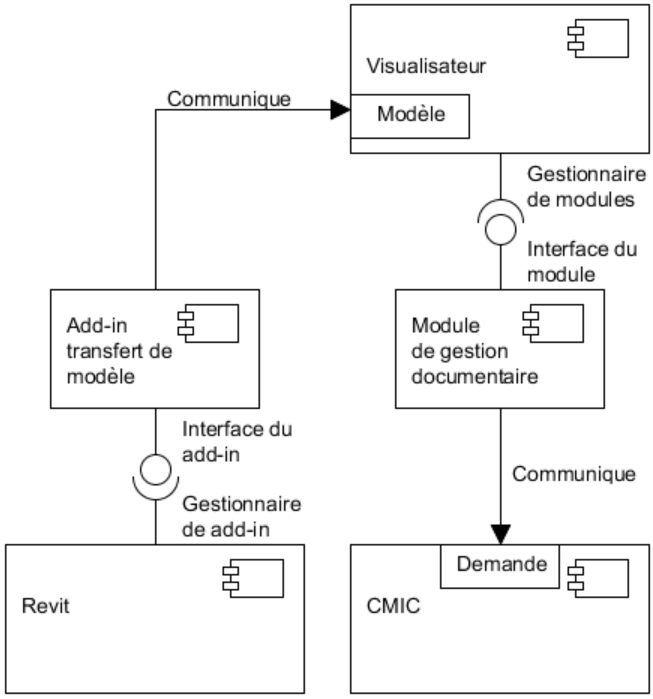

Figure 6 : Diagramme des interfaces systèmes

7.2.1 Add-in transfert de modèle

Le Add-in de transfert de modèle doit offrir un interface qui suit les spécifications définies par l'interface « IExternalCommand » fournie par le SDK de Revit.

7.2.2 Visualisateur

7.2.2.1 Réception des modèles

Le visualisateur doit offrir un service web qui permet de recevoir un modèle 3D contenue dans un fichier. Le service sera responsable d'enregistrer les informations nécessaires dans le système afin que l'utilisateur puisse accéder au modèle.

7.2.2.2 Gestionnaire du module d'extension

Le gestionnaire du module d'extension permet de charger des modules afin d'étendre les fonctionnalités du portail web. Celui-ci devra permettre au module externe d'accéder aux données de façon simple et sécuritaire.

7.2.3 Module de gestion documentaire

7.2.3.1 Interface du module

Une interface pour incorporer les modules doit être fournir au développeur. Cette interface permet au module d'accéder aux données contenues dans le visualisateur.

7.2.4 CMIC

7.2.4.1 Service web pour la gestion documentaire

CMIC doit offrir un ou plusieurs services web pour obtenir les informations nécessaires sur les demandes associées à un projet. Par ces services le système pourra modifier ou consulter les informations d'une demande.

8 Standards applicables

8.1 JSON

Le transfert des paramètres entre le add-in et le portail web sera dans un format JSON selon les spécifications données par le site officiel de celui-ci, soit : http://www.json.org/.

8.2 Format des dates

Toutes dates en chaine de caractères seront dans le format « AAAAMMJJ » lors du traitement des données et sous le format « AAAA-MM-JJ » lors de l'affichage des données. Ceci vaut pour les services JSON et pour le stockage dans la base de donnée, si il y a lieu.

*Ce standart ne s'applique pas au date qui sont représenté de façon numérique tel que par le type « datetime » de C#.

Annexe III - Proposition de projet

PROPOSITION
Projet de fin d'études
Département de génie logiciel et des TI

Sécurité de la plateforme BIMWeb

Auteurs
Samuel Milette-Lacombe
MILS26059100

Professeur superviseur
Alain April

Date
22 janvier 2015

Suivi des changements

*__A__ – Ajouté __M__ – Modifié __S__ – Supprimé

NUMÉRO DE VERSION	DATE aaaa/mm/jj	NUMÉRO DE FIGURE, TABLE OU SECTION	A* M S	BRÈVE DESCRIPTION DU CHANGEMENT	NUMÉRO DE DEMANDE CHANGEMENT
0.5	2015/01/20		A	Remplis toutes les sections, mais la planification n'a pas encore d'effort ni de date attribuée à chacun des jalons.	
0.7	2015/01/22		M	Attribue une date et une estimation de l'effort pour chaque jalon.	
0.8	2015/01/26		M	Correction des éléments soulevés lors de la revue par Mathieu Dupuis.	
0.9	2015/02/04		M	Ajustement des jalons pour refléter l'avancement	

TABLE DES MATIÈRES

1. Problématique et contexte 4
2. Objectifs du projet 4
3. Méthodologie 4
4. Livrables et planification 6
   4.1 Description des artéfacts 6
   4.2 Planification 6
5. Risques 7
6. Techniques et outils 7
7. Références 8
ANNEXE A : Plan de travail 9

1. PROBLÉMATIQUE ET CONTEXTE

BIMweb est une plateforme d'interopérabilité dans le domaine de la construction. Cette plateforme permet aux divers intervenants dans un projet de construction de partager données et programmes afin de faciliter la construction de bâtiment. Pour un même projet, la plateforme peut accueillir plus de 500 intervenants. Ceux-ci travaillent pour différente compagnie et dans différentes professions.

Le problème est le manque actuel de stratégies de sécurité pour la plateforme. Les règles de sécurité logicielles sont inexistantes concernant les objets du système qui doivent être sécurisés comme les projets et toutes les informations dérivées.

Un tel manque de stratégie de sécurité peut mener à ce que les données du système soient compromises en étant altérées, accédées ou supprimées par des utilisateurs malintentionnés ou des systèmes externes.

2. OBJECTIFS DU PROJET

Le projet consiste à élaborer des stratégies de sécurité qui cadrent avec les besoins des utilisateurs de la plateforme et de les implémenter dans celle-ci.

Le but premier consiste à définir les besoins du client en terme de sécurité pour la plateforme. Les objets du système qui doivent être sécurisés seront décrits et des règles d'accès pour chacun des objets seront définies. Ces règles permettront de définir comment les utilisateurs du système pourront, à travers des responsabilités et autorisations spécifiques, accéder et modifier les informations du projet.

Le but second est de définir des règles de sécurité logicielles permettant de s'assurer que les modules de tierce partie s'intégrant à la plateforme, accèdent et modifient les données du système de manière sécuritaire selon leurs autorisations respectives.

Les retombées du projet sont la protection des données. Cette protection permettra de contrer d'éventuelles attaques visant à effectuer des opérations non autorisées sur le système.

3. MÉTHODOLOGIE

Une approche itérative sera utilisée parce que les besoins en terme de sécurité ne sont pas encore bien définis et compris. À chaque fin d'itération, une rencontre avec le client sera planifiée afin de faire office de revue pour apporter des suggestions d'amélioration au travail fait

dans l'itération. Au début de l'itération suivante, un raffinement de l'itération précédente sera fait en tenant compte des suggestions apportées précédemment.

Une analyse des besoins en terme de sécurité sera réalisée. Les différentes stratégies de sécurité possibles seront ensuite analysées, et une sélection de stratégies sera faite et elle sera implantée dans le système.

Un système de gestion de versions GIT sera utilisé pour gérer les versions du code source. Une revue de code sera effectuée à chaque ajout de code pertinent dans GIT. Des tests seront élaborés pour trouver des défauts dans les stratégies de sécurité. Un nombre non limité de réunions et de consultations avec le client sera possible pendant les itérations pour répondre aux différentes questions qu'il pourrait y avoir sur les besoins de sécurité.

4. LIVRABLES ET PLANIFICATION

4.1 Description des artéfacts

Nom de l'artefact	Description
Analyse des besoins VX	Analyse des besoins en terme de sécurité. Définition des objets à protéger. Définition des rôles utilisateur et de leurs autorisations et restrictions par rapport aux objets du système. Définition du niveau de granularité des autorisations des utilisateurs pour chaque type d'objet du système.
Analyse des stratégies de sécurité VX	Définit les différentes stratégies de sécurité possibles pour satisfaire les besoins de sécurité. Une sélection de stratégies sera faite et le fondement de chacun des choix sera expliqué
Compte rendu de la revue VX	Améliorations et corrections notées lors de la revue avec le client concernant les analyses effectuées précédemment.
Base de données VX	C'est le schéma relationnel de la base de données spécifiquement pour la partie sécurité du système.
Code source VX	Il s'agit de l'implémentation des stratégies de sécurité définies précédemment.
Tests VX	Il s'agit d'un plan de test et des tests logiciels automatisés si possible pour tester les stratégies de sécurité implémentées.
Rapport d'étape	Document décrivant le déroulement du projet et rectifiant certains éléments de la proposition de projet qui auraient pu changer.
Rapport final	Document présentant les résultats du projet

X représente le numéro de la version

4.2 Planification

Voir Annexe A..

5. RISQUES

Risque	Impact	Probabilité	Mitigation / atténuation
Disponibilité trop faible du client pour consultation	Élevé	Moyenne	● Rassembler les questions à poser aux clients pour minimiser le nombre de rencontres ● Communiquer par courriel
Fonctionnalité à sécuriser non terminée ou non commencée.	Moyen	Élevée	● Développer des objets mock pour éviter de devoir attendre que la fonctionnalité soit implémentée.
Technologies mal maîtrisées ou mal choisies	Moyen	Moyenne	● Lecture de documents de référence à jour ● Réunir les bonnes pratiques et les appliquer
Besoins de sécurité non documentés	Élevé	Haute	● Documenter les besoins à l'aide du client le plus tôt possible dans les itérations.
Portée trop ambitieuse	Moyen	Élevée	● Faire un suivi pointu du déroulement du projet par rapport à la planification prévue et réduire la portée du projet si nécessaire.
Mauvaise compréhension des besoins	Élevée	Moyen	● Faire une revue avec le client, concernant les besoins de sécurité, avant de commencer toute implémentation ou toute élaboration de stratégies de sécurité.

6. TECHNIQUES ET OUTILS

Outils pour l'implémentation
- Visual Studio 2012 Ultimate
- Sql Server developper 2012
- Framework pour la création de mock object.

Outils pour la gestion de configuration
- Plugin git dans Visual Studio

Outils pour les documents de livrable
- Google Drive
- *Logiciel UML*
- *Microsoft Word*
- *Microsoft Windows*

7. RÉFÉRENCES

- SRS 1.0 - Visualisateur 3D Web, Mathieu Dupuis
- Document de vision - Visualisateur 3D Web, Mathieu Dupuis
- *Plan de gestion de projet - Visualisateur web 3D, Mathieu Dupuis*

ANNEXE A : PLAN DE TRAVAIL

#	Commence	Termine	Efforts estimés *	Tâches/Jalon	Livrable(s)/Artéfacts
1					
	2014-12-20	2015-01-07	2	Remise de la fiche de renseignements	Fiche de renseignements
	2015-01-07	2015-01-19	5	Analyse préliminaire pour spécifier les itérations du projet et les besoins sommaires de sécurité de BIMWeb.	
		2015-01-12	1	Rencontre – professeur superviseur	
	2015-01-07	2015-01-19	2,5	Installation et réglage de l'environnement de développement	
	2015-01-20	2015-01-23	6	Remise de la proposition de projet	Proposition de projet
	2015-01-07	2015-04-01	10	Consultation de documentation sur la sécurité	
	2015-01-07	2015-02-10	6	Familiarisation avec BIMWeb	
	2015-01-24	2015-02-06	10	Analyse des besoins	Analyse des besoins V1
	2015-02-06	2015-02-10	8	Analyse des stratégies de sécurité	Analyse des stratégies de sécurité V1
		2015-02-11	2,5	Rencontre de fin d'itération avec le client	Compte rendu de la revue de fin d'itération 1
2	2015-02-11	2015-02-15	5	Raffinement des analyses en intégrant les suggestions notées à la revue de fin d'itération.	• Analyse des besoins V2 • Analyse des stratégies de sécurité V2
	2015-02-11	2015-02-15	5	Modélisation de la base de données	Base de données V1
	2015-02-16	2015-02-22	13	Implémentation des stratégies de sécurité	• Code source V1
	2015-02-23	2015-03-01	5	Implémentation des tests des stratégies de sécurité	Tests V1
	2015-02-23	2015-03-01	6	Rédaction du rapport d'étape	Rapport d'étape
		2015-03-02	2,5	Rencontre de fin d'itération avec le client	Compte rendu de la revue de fin d'itération 2
3	2015-03-03	2015-03-08	5	Raffinement du code en intégrant les suggestions notées à la revue de fin d'itération.	• Code source V2 • Base de données V2 • Tests V2
	2015-03-03	2015-03-08	5	Rencontre(s) avec le client pour spécifier les besoins de sécurité et le fonctionnement de l'intégration des modules	

	2015-03-09	2015-03-15	5	Intégration de la notion de module à l'analyse des besoins	Analyse des besoins V3
	2015-03-09	2015-03-15	5	Intégration de la notion de module à l'analyse des stratégies de sécurité	Analyse des stratégies de sécurité V3
		2015-03-16	2,5	Rencontre de fin d'itération – professeur superviseur et client	• Compte rendu de la revue de fin d'itération 3
4	2015-03-16	2015-03-22	5	Raffinement des analyses en intégrant les suggestions notées à la revue de fin d'itération.	• Analyse des rôles et des droits des utilisateurs V4. • Analyse des stratégies de sécurité V4.
	2015-03-16	2015-03-22	3	Intégration de la notion de module à la base de données	Base de données V3
	2015-03-23	2015-03-29	8	Intégration de la notion de module à l'implémentation des stratégies de sécurité	• Code source V3 • Base de données V3
	2015-03-23	2015-03-29	5	Intégration de la notion de module aux tests pour les stratégies de sécurité implémentées	Tests V3
		2015-03-30	2,5	Rencontre de fin d'itération avec le client	Compte rendu de la revue de fin d'itération 4
	2015-03-31	2015-04-05	5	Raffinement du code et de la base de données en intégrant les suggestions notées à la revue de fin d'itération.	• Code source V4 • Base de données V4
	2015-04-01	2015-04-12	5	Préparation à la présentation	Support visuel de la présentation
		2015-04-13	0,5	Présentation	Présentation
	2015-04-01	2015-04-15	10	Rédaction du rapport final	Rapport
		Total	156 h		

Annexe IV - Analyse des besoins de sécurité

Projet de fin d'études
Département de génie logiciel et des TI

Sécurité du système BIMWeb
Analyse des besoins

Auteur
Samuel Milette-Lacombe
MILS26059100

Professeur superviseur
Alain April

Suivi des changements

*__A__ – Ajouté **M** – Modifié **S** – Supprimé

NUMÉRO DE VERSION	DATE aaaa/mm/jj	NUMÉRO DE FIGURE, TABLE OU SECTION	A* M S	BRÈVE DESCRIPTION DU CHANGEMENT	NUMÉRO DE DEMANDE CHANGEMENT
1.0	2015/02/07		A	Version initiale	
2.0	2015/02/27		AM	Révision après la revue par Mathieu Dupuis.	

Contenu

1 Introduction 4

1.1 Objectif 4

1.2 Portée 4

2 Contraintes de technologies et de plateforme 4

3 Connexions des utilisateurs du système 4

4 Entités à sécuriser 5

4.1.1 Entités principales 5

4.1.2 Autres entités 5

5 Niveau de granularité et fonctionnement des autorisations 6

5.1 Rôles des utilisateurs 6

5.2 Matrice de contrôle d'accès 7

5.2.1 Droits possibles 7

5.2.2 Matrice 7

6 Cryptage des données 7

1 Introduction

1.1 Objectif

L'objectif de ce document est d'identifier les besoins de sécurité informatique pour les fonctionnalités logicielles de l'application web BIMWeb. Plus précisément, le document vise à identifier les entités du système qui ont besoin d'être protégées des opérations non autorisées. Le document cible également les différents rôles qu'il est possible d'assigner aux utilisateurs.

1.2 Portée

Le document se concentre particulièrement à décrire les besoins en termes de sécurité pour l'authentification et l'autorisation des utilisateurs. Cependant, d'autres éléments importants à savoir pour pouvoir élaborer des stratégies de sécurité, comme les circonstances d'utilisation du système touchant à différents aspects de la sécurité seront également détaillés. Les besoins de sécurité générique à toute application web telle que le besoin de se protéger des injections SQL ne seront pas décrits dans ce document.

2 Contraintes de technologies et de plateforme

Les stratégies de sécurités devront donc être élaborées en prenant en compte les contraintes technologiques qui suivent. Il est entendu que BIMWeb est une application web utilisant les technologies ASP.NET MVC 4 avec une base de données SQL Server. Cependant, il n'y a pas de contrainte particulière à considéré en plus, des systèmes externes ou des librairies jugées utiles pour la sécurité du système.

3 Connexions des utilisateurs du système

Il y aura environ 3 employés par compagnie qui utiliseront le système. Tous les autres utilisateurs seront des collaborateurs externes auxquels les employés leur auront autorisé l'accès à certains modèles de projet via des invitations.

Il n'est pas jugé requis de gérer le type d'authentification dit externe comme le permet le protocole OpenId. Ce protocole permet à un utilisateur de s'authentifier à plusieurs applications web en utilisant un même compte stocké par un fournisseur d'authentification OpenId. L'application web est donc déresponsabilisé de valider l'authentification des utilisateurs. Ce type d'authentification n'est pas approprié à BIMWeb puisque tous les usagers utiliseront leur adresse courriel corporative en tant que nom d'utilisateur pour leur compte et non un compte qu'ils utiliseraient pour d'autres applications.

4 Entités à sécuriser

4.1.1 Entités principales

Nom	**Description**
Compagnie	Compagnie inscrite dans le système. Il s'agit du conteneur principal dans lequel on peut y ajouter des projets.
Projet	Projets de construction assignés à une compagnie. Le projet contient un ou plusieurs modèles BIM.
Modèle	Maquette 3D d'un bâtiment composée d'objets paramétrables et intelligents. Lors d'un téléversement d'un modèle, les données du fichier sont extraites et sont placées dans la base de données du système. Le fichier doit aussi être stocké sur le système.
Module	Un module de tierce partie pour étendre la fonctionnalité de base de BimWeb

4.1.2 Autres entités

La base de données et le système de fichier sont implicitement inclus dans les entités à protéger. Cependant, les utilisateurs ne devraient pas interagir directement avec celles-ci, mais plutôt avec l'application qui, de manière transparente pour l'utilisateur, communiquera avec le système de fichier et la base de données.

5 Niveau de granularité et fonctionnement des autorisations

Il y a trois niveaux d'accès qui correspond aux entités à sécuriser: la compagnie, le projet et le modèle. Chaque rôle donne un certain nombre d'autorisations pour le niveau de rôle attribué et autorise toutes les actions possibles pour les niveaux inférieurs. Un utilisateur ne doit pas se faire attribuer un rôle qui entre en conflit avec un autre rôle qui lui a été assigné. Ainsi, un utilisateur ne doit pas se faire assigner plus d'un rôle par entité. Également, un utilisateur ne devrait pas pouvoir se faire attribuer un rôle inférieur qui viendrait contredire un rôle supérieur. Par exemple, l'attribution d'un rôle d'Admin sur une compagnie sur un utilisateur disposant d'un rôle de Lecteur sur un projet de cette compagnie devrait résulter en la suppression du rôle de Lecteur pour éviter d'avoir des attributions contradictoires de rôles.

5.1 Rôles des utilisateurs

Entité	Rôle	Description et droits
Compagnie	User	Rôle de base. Ne peut rien faire. Un utilisateur ne devrait être assigné qu'à une compagnie. Il peut cependant être invité pour participer à un modèle d'un projet d'une autre compagnie.
	Admin	A les mêmes droits que l'AdminBim en plus de pouvoir changer le maximum de projet pour une compagnie (ce qui est considéré comme une modification de la compagnie).
	Admin BIM	Peut créer et supprimer des projets et des modèles dans les projets.
Projet	Gestionnaire de projet BIM	• Suppression/Ajout de modèle • Invite des intervenants externes pour des modèles • Définit le droit en écriture des invités sur les modèles
Modèle	Lecteur	Lire les données du modèle. Ce rôle permet à un utilisateur de voir le projet dont est issu le modèle. Il s'agit du rôle par défaut des utilisateurs invités pour un modèle. Un invité (comme tout utilisateur) peut aussi être muté comme éditeur si le gestionnaire de projet Bim lui en donne le droit.
	Éditeur	Combine le rôle de Lecteur en plus de pouvoir modifier le modèle. La modification de modèle inclut l'écrasement de modèle.

5.2 Matrice de contrôle d'accès

5.2.1 Droits possibles

Droit	Lettre	Description
Lecture	R	Accès en lecture
Mise à jour	U	Accès en écriture
Création	C	Peut créer l'entité
Suppression	D	Peut supprimer l'entité
Invitation	I	Peut inviter un utilisateur externe à la compagnie (seulement pour le modèle)

5.2.2 Matrice

		Compagnie				Projet				Modèle				
Entité	**Rôle**	**R**	**U**	**C**	**D**	**R**	**U**	**C**	**D**	**R**	**U**	**C**	**D**	**I**
Compagnie	User	X												
	Admin	X	X			X	X	X	X	X	X	X	X	
	AdminBIM	X				X	X	X	X	X	X	X	X	
Projet	Gestionnaire de projet Bim	X				X	X			X	X	X	X	X
Modèle	Lecteur	X				X				X				
	Éditeur	X				X				X	X	X		

6 Cryptage des données

Les données contenues dans la base de données concernant les entités n'ont pas à être cryptées puisqu'ils n'ont pas un degré de criticité suffisant.

Annexe V – Version finale des jalons et des efforts

Tâches/Jalon	Efforts estimés*	Efforts actuels*
Remise de la fiche de renseignements	2	2
Analyse préliminaire pour spécifier les itérations du projet et les besoins sommaires de sécurité de BIMWeb.	5	5
Rencontre – professeur superviseur	1	1
Installation et réglage de l'environnement de développement	2,5	4
Remise de la proposition de projet	6	6
Consultation de documentation sur la sécurité	20	10,5
Familiarisation avec BIMWeb	6	6
Analyse des besoins	10	9
Analyse des stratégies de sécurité	8	10,25
Rencontre de fin d'itération 1 avec le client	2,5	1
Raffinement des analyses en intégrant les suggestions notées à la revue de fin d'itération.	5	5,25
Modélisation de la base de données	5	4,5
Rédaction du rapport d'étape	6	2
Implémentation des stratégies de sécurité et réalisation d'une ébauche pour les interfaces de gestion des permissions	25	59
Implémentation des tests des stratégies de sécurité	5	15
Rencontre de fin d'itération avec le client	1	0
Rencontre(s) avec le client et un expert en .NET pour spécifier les besoins de sécurité et le fonctionnement de l'intégration des modules	3	1,5
Intégration de la notion de module à l'analyse des besoins	10	1,5
Préparation à la présentation	5	10
Présentation	0,5	0,5
Rédaction du rapport final	16	37,75
Total	150,5 h	191,75 h