Sporacid Studios

SimplETS
Logiciel de gestion de club étudiant
Rapport final
Version <1.2>

Préparé par :
Jean Bernier Vibert
Patrick Lavallée
Simon Turcotte-Langevin
François Gagné

Historique des révisions

Date	Version	Description	Auteur
2015-07-21	1.0	Version initiale	Patrick Lavallée Jean Bernier Vibert
2015-08-01	1.1	Section 2 Section 4 Section 5	Simon Turcotte Langevin François Gagné
2015-08-04	1.2	Section 2 Section 5	Simon Turcotte Langevin Patrick Lavallée François Gagné Jean Bernier Vibert

Table des matières

1. Introduction 5
- 1.1 Objectif du document 5
- 1.2 Vue d'ensemble 5

2. La situation actuelle du projet 5
- 2.1 Administration 5
  - 2.1.1 Hébergement 5
  - 2.1.2 Gestion du système 5
  - 2.1.3 Gestion des clubs étudiants 5
- 2.2 Conception 5
  - 2.2.1 Cas d'utilisations 5
  - 2.2.2 Schématisation et implémentation de la base de données effectuées 6
- 2.3 Documentation 6
  - 2.3.1 Document de vision/architecture/SRS 6
  - 2.3.2 Wiki utilisateur et développeur 6
  - 2.3.3 Documentation dans le code 6
  - 2.3.4 Service de documentation automatique de l'API 6
- 2.4 Implémentation 6
  - 2.4.1 Back-end 6
  - 2.4.2 Front-end 6

3. Analyse du travail effectué 7
- 3.1 Présentation des tâches 7
- 3.2 Répartition des tâches – Administration 8
- 3.3 Répartition des tâches - Conception 8
- 3.4 Répartition des tâches – Documentation 9
- 3.5 Répartition des tâches – Implémentation 9

4. Défis rencontrés 10
- 4.1 Défis politiques 10
  - 4.1.1 Nombreux intervenants du projet 10
  - 4.1.2 Protection des données nominatives étudiantes 10
  - 4.1.3 Hébergement de la solution 10
- 4.2 Défis technologiques 10
  - 4.2.1 Migration du projet vers la pile Microsoft 10
  - 4.2.2 Courbe d'apprentissage dû à la modernité des technologies 10
- 4.3 Défis humain 10
  - 4.3.1 Assurer une pérennité au projet 10
  - 4.3.2 Coordination de l'équipe 10
  - 4.3.3 Analyse du domaine d'affaires 11

5. Améliorations possibles 11
- 5.1 Bonification de la solution 11
  - 5.1.1 Environnement de test 11
  - 5.1.2 Automatisation du déploiement 11
  - 5.1.3 Feedback des utilisateurs 11
  - 5.1.4 Support de sérialisation NoSQL 11
  - 5.1.5 Amélioration du cache 11
  - 5.1.6 Amélioration des interfaces 11

5.2 Pérennité 12
5.2.1 Transfert de connaissance 12
5.2.2 Améliorer le wiki du développeur 12
5.2.3 Améliorer le wiki de l'utilisateur 12
5.2.4 Trouver une équipe 12

Annexe A : Points en suspens et décisions 13

Annexe B : Source et documentation du projet 13

Annexe C : Documents finaux 13

1. Introduction

1.1 Objectif du document

L'objectif de ce document est de dresser un portrait de la situation actuelle du projet de fin d'études de l'équipe *SimplETS*. Il présentera aussi un post-mortem de ce dernier démontrant le processus de décision prises en cours de projet ainsi que les défis relevé.

1.2 Vue d'ensemble

Ce document contiendra les sections suivantes:

1. La situation actuelle du projet
2. Analyse du travail effectué
3. Les défis rencontrés
4. Les améliorations possibles
5. Les annexes

2. La situation actuelle du projet

Cette section détaille la situation actuelle des différentes facettes du projet de gestion des clubs étudiants SimplETS. Ce portrait détaille les responsabilités au point de l'administration de l'application, l'état de sa conception, de sa documentation et finalement de son implémentation.

2.1 Administration

2.1.1 Hébergement

Tout d'abord, l'application a besoin d'être hébergée à l'intérieur des infrastructures de l'université. Plusieurs raisons expliquent ce besoin. L'application nécessite d'avoir accès à l'annuaire des usagers de l'école (LDAP) afin d'authentifier ses utilisateurs. Par conséquent, l'application doit être dans le même domaine que cet annuaire. De plus, cette situation permet de bénéficier de la sécurité offerte à l'intérieur du réseau de l'ÉTS. Elle permettrait aussi, dans un futur, d'offrir une authentification unique (single sign-on).

2.1.2 Gestion du système

Chaque club utilisant l'application SimplETS aura un niveau d'autonomie assez élevé. Toutefois, il va de soi que certaines actions devront être effectuées à un niveau privilégié. Parmi ces actions se trouveront la création de nouveaux clubs et l'attribution du rôle de capitaine à un membre d'un club. Afin d'effectuer ces actions privilégiées, un usager devra posséder les droits d'administrateur système au niveau de l'application.

2.1.3 Gestion des clubs étudiants

Un certain niveau d'administration est aussi nécessaire au sein de chaque club. C'est pourquoi un ou des responsables pourront être désignés afin d'effectuer ces tâches. Parmi celles-ci se retrouveront l'ajout de membres dans leur club, la personnalisation de certains éléments du site et la gestion des privilèges. Cette gestion de privilèges permet de restreindre l'accès aux différentes sections du club. Dans une évolution de la solution, les responsables de club pourraient définir de nouveaux rôles afin de gérer de façon personnalisée les accès à leur club.

2.2 Conception

2.2.1 Cas d'utilisations

Pour la portée du projet, huit cas d'utilisation ont été identifiées. Ces cas d'utilisation couvrent les principaux besoins des clubs, c'est-à-dire la gestion des membres, des fournisseurs, des commandites et de l'authentification. Pour ces différents cas, nous avons pris en compte les différents scénarios nominaux ainsi que ceux d'exception afin de bien caractériser les processus d'affaires liés au cycle de vie des différents clubs. De nouveaux cas pourraient être ajoutés afin de détailler des besoins comme la gestion de réunions, d'inventaire et d'événements.

2.2.2 Schématisation et implémentation de la base de données effectuées

En plus de couvrir l'ensemble du schéma nécessaire à l'implémentation des principaux cas d'utilisation, l'équipe a décidé de schématiser aussi certains besoins futurs. Par conséquent, la base de données contient plus d'objets qu'initialement nécessaire au projet. Cette initiative permettra donc de faciliter l'évolution future de l'application vers ces nouveaux besoins.

La base de données est actuellement générée l'aide de différents scripts. Ces scripts sont difficiles à maintenir; il serait donc judicieux, à l'avenir, d'automatiser ce processus via la librairie de mappage objets-relationnels.

2.3 Documentation

2.3.1 Document de vision/architecture/SRS

2.3.2 Wiki utilisateur et développeur

À l'aide des outils offerts par *GitHub*, l'équipe a pu rendre disponible aux acteurs concernés une documentation s'adressant à leurs besoins potentiels. Cette documentation sous forme de Wiki explique aux utilisateurs les différentes interfaces leur étant disponibles et de quelle façon ils doivent les utiliser afin de répondre à leurs besoins.

La section pour les développeurs, quant à elle, définit les différentes étapes nécessaires à l'utilisation de la solution afin d'implémenter de futurs évolutions de l'application. Cette portion donne aussi les étapes nécessaires afin de consulter l'API des différentes méthodes offertes par les services web.

2.3.3 Documentation dans le code

L'ensemble des différentes méthodes et stratégies utilisées au sein de l'application sont documentées à même le code et ce, autant au niveau des services web qu'au niveau de l'application utilisant ces services. Ceci permettra à un développeur néophyte de se familiariser plus facilement avec la plateforme offerte par l'application. Cette documentation se scinde en deux sections : la documentation des entêtes de méthodes, via le schéma XML de .NET ainsi que les commentaires d'ordre général qui permettent de faciliter la compréhension des algorithmes.

2.3.4 Service de documentation automatique de l'API

Un ensemble de méthodes sont offertes par les services web. Afin de faciliter la découverte et la compréhension de celles-ci, une méthode particulière a été ajoutée au sein des services web. Cette méthode permet d'exposer les différents détails d'utilisation sur l'ensemble des services web. Un développeur peut donc consulter cette méthode afin de facilement connaître les prés et post conditions d'un service offert.

2.4 Implémentation

2.4.1 Back-end

Le *back-end* a été traduit complètement de Java à C# et est complété a 95%. Il reste cependant les messages d'exception, de contrat et de validation à écrire dans un fichier de ressource. À noter que toutes les clés uniques identifiant les messages sont déjà créées. La quasi-totalité des cas d'utilisation sont implémentés. Les fonctionnalités qui restent à implémenter ne sont pas critiques au bon fonctionnement du système. Les préoccupations transversales (cross-cutting concerns), telles que l'autorisation, la validation et l'authentification, sont complétées à 100%.

2.4.2 Front-end

Premièrement, la page principale de l'application englobant les menus, la navigation ainsi qu'une zone d'injection de contenu est complétée à 85%. Certaines règles d'affaires et certains éléments de menus n'ont pu être achevés. Deuxièmement, l'intégration des librairies de développement Web *Knockout.js* et *BootStrap* est complétée. L'équipe a développé un ensemble de méthodes utilitaires accélérant l'intégration des pages partielles dans le cycle de vie de l'application. Troisièmement, la portion de développement restante comprend la finalisation des pages partielles pour les modules de gestion de l'application. Finalement, l'équipe évalue la complétion de cette portion à 50%.

3. Analyse du travail effectué

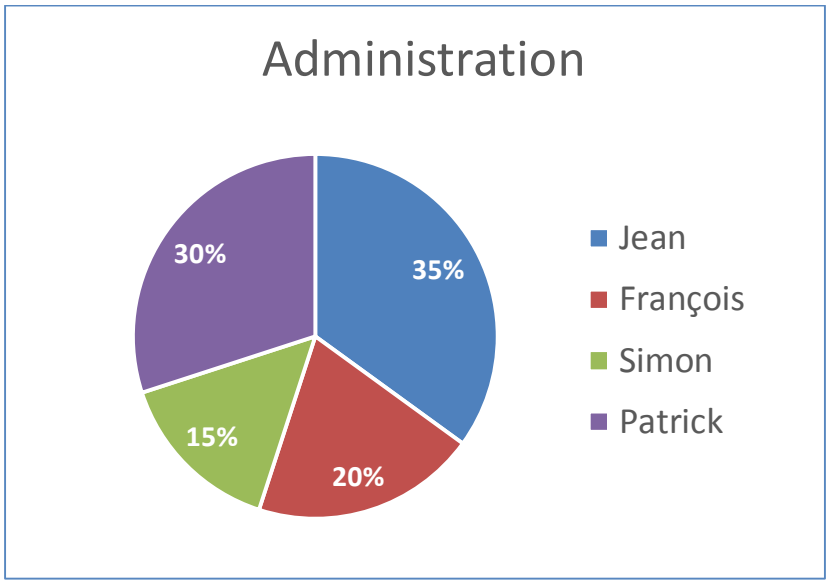
Cette section détail le travail qui a été effectué pendant la durée du projet. D'abord, une présentation globale des tâches effectuées et leur catégorie, suivie d'une analyse graphique de l'apport de chacun.

3.1 Présentation des tâches

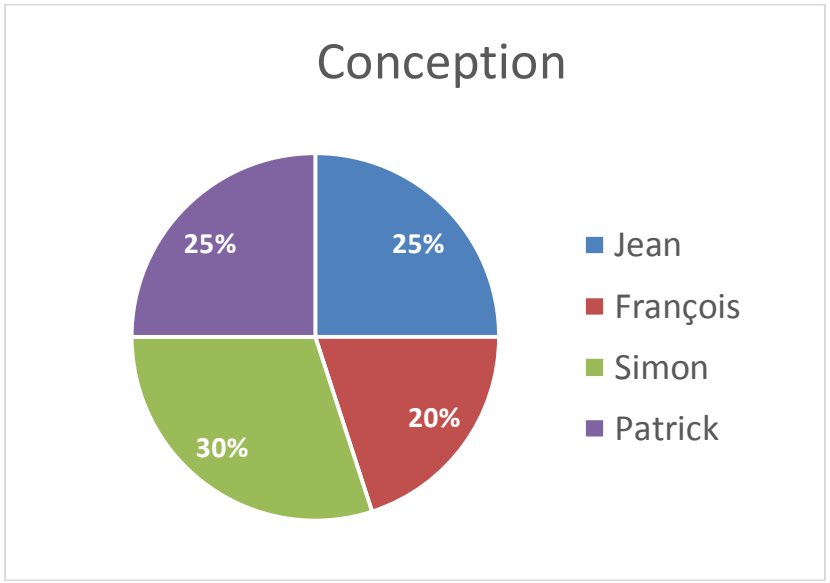
ID	Description de la tâche	Catégorie
1	Préparation de la présentation PREZI	Administration
2	Rencontres administratives avec différents acteurs de l'ÉTS	Administration
3	Recherche d'un groupe pour prendre le relais	Administration
4	Rencontre d'équipe hebdomadaire	Administration
5	Mise en place d'outils de gestion de projet	Administration
6	Recherche d'hébergement dans l'ÉTS	Administration
7	Conception de l'interface d'authentification	Conception
8	Conception de l'interface du profil	Conception
9	Conception de l'interface des membres	Conception
10	Conception de l'interface des commandites	Conception
11	Conception du modèle de donnée	Conception
12	Analyse du modèle d'affaires et des spécifications	Conception
13	Création d'un Wiki pour les développeurs	Documentation
14	Création d'un Wiki pour les utilisateurs	Documentation
15	Documentation automatisée de l'API	Documentation
16	S'assurer de l'exactitude des commentaires dans le code	Documentation
17	Rédaction du SRS	Documentation
18	Rédaction du plan de projet	Documentation
19	Rédaction du document de vision	Documentation
20	Rédaction du document d'architecture	Documentation
21	Rédaction du plan de test	Documentation
22	Perfectionnement technique dans les technologies incluent dans le projet.	Documentation
23	Conception de contrôles UI	Implémentation
24	Automatisation du déploiement de la base de données	Implémentation
25	Création du module des membres	Implémentation
26	Création du module d'authentification	Implémentation
27	Création du module des commandites	Implémentation
28	Implémentation du concept de Single Page Application (SPA)	Implémentation
29	Configuration de l'ORM Entity Framework	Implémentation
30	Création de l'interface du profil	Implémentation
31	Création de l'interface d'authentification	Implémentation
32	Création de l'interface des membres	Implémentation
33	Création de l'interface des commandites	Implémentation
34	Implémentation de la sécurité de l'application	Implémentation
35	Création du modèle de données	Implémentation
36	Implémentation du multilinguisme	Implémentation

Tableau 1 - Listes des tâches

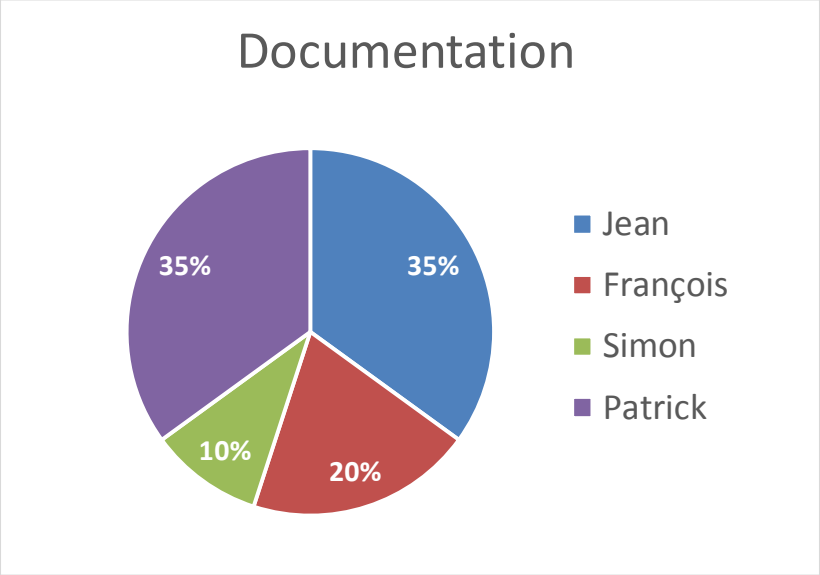
3.2 Répartition des tâches – Administration



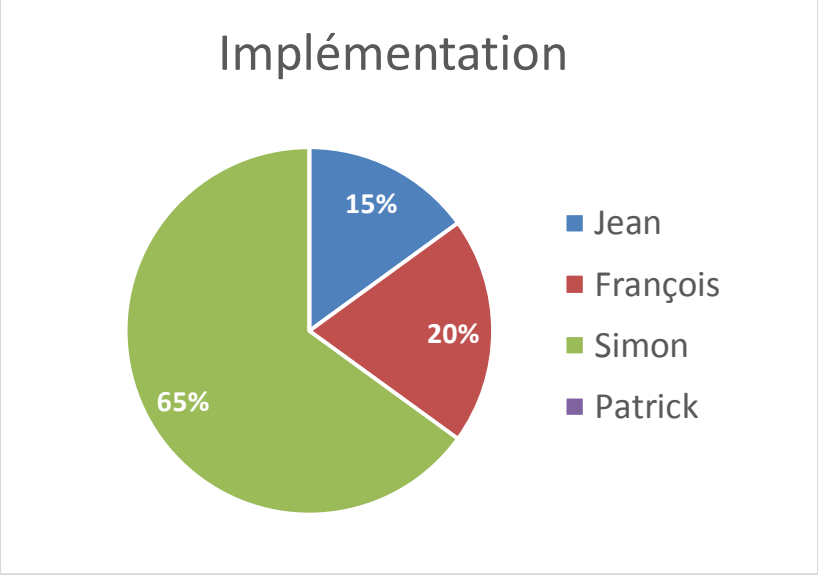
3.3 Répartition des tâches - Conception



3.4 Répartition des tâches – Documentation



3.5 Répartition des tâches – Implémentation



4. Défis rencontrés

Cette section présente les défis que l'équipe à due faire face pendant la réalisation du projet et sont divisés en trois catégories.

4.1 Défis politiques

4.1.1 Nombreux intervenants du projet

Plusieurs paliers administratifs ont dû être franchis. L'approbation de chacun d'entre eux était nécessaire pour le bon déroulement du projet; l'ÉTS et les clubs étudiants étant les clients du produit. L'équipe s'est dotée des outils de gestion administrative de la suite *Google (Calendar, Drive)*. L'orchestration des rencontres afin de bien vendre l'idée aux services de l'école faisait aussi partie de cette expérience formatrice.

4.1.2 Protection des données nominatives étudiantes

Depuis la dernière version, une notion de profil privé sécurisé protégeant les informations sensibles de tous les utilisateurs. Il en revient à l'utilisateur de compléter son profil et d'aiguiller les informations qu'il ne veut pas divulguer. Cette approche permet d'harmoniser les visions de *SimplETS*, des clubs étudiants et des avocats de l'école.

4.1.3 Hébergement de la solution

L'hébergement de la solution sur les infrastructures de l'ÉTS était conditionnel à la l'approbation par l'administration des clubs étudiants. D'ailleurs, la continuité et le support du développement devait être assuré après le déploiement de la solution. À défaut d'avoir pu trouver un successeur, l'équipe s'est résolue à ne pas déployer *SimplETS* sur les environnements de l'école.

4.2 Défis technologiques

4.2.1 Migration du projet vers la pile Microsoft

Lors de la première itération de *SimplETS* dans le cadre du cours GTI791, la solution a été implémentée en Java intégrant des technologies hétéroclites. La gestion de la configuration était longue, pénible et couteuse en temps. Un passage à la pile Microsoft était nécessaire. Par conséquent, l'équipe a dû traduire l'application vers les technologies Microsoft, ce qui demandait la connaissance des cadriciels .NET et Java.

4.2.2 Courbe d'apprentissage dû à la modernité des technologies

Pour le développement *front-end*, des librairies telles que *Knockout.js* et *Bootstrap 3* ont été utilisées. Ces technologies étaient nouvelles pour l'ensemble de l'équipe; par conséquent, la vélocité d'implémentation s'est révélée plus lente en début de projet. Cette courbe d'apprentissage à impactée la livraison d'une partie des interfaces graphiques.

4.3 Défis humain

4.3.1 Assurer une pérennité au projet

Les clubs étudiants qui poursuivent des activités connexes au développement Web n'ont pas manifesté l'intérêt de prendre le relais. D'une part, l'écosystème technologique nécessaire par *SimplETS* et ceux offerts par ces clubs ne sont pas compatibles. D'autre part, il était difficile de synchroniser la vision de l'équipe et celles des successeurs potentiels.

4.3.2 Coordination de l'équipe

Il a été difficile de bien coordonner les réunions d'équipes vu les différentes contraintes personnelles de chacun des membres.

4.3.3 Analyse du domaine d'affaires

Il était important de bien caractériser les besoins des clubs en matière de gestion administrative des clubs étudiants. Les cas d'utilisations se devait d'être simples et concis et les objets du domaine, bien détaillés. De plus, plusieurs types d'usager, peuvent interagir avec la plateforme; par conséquent, l'un des défis était de bien définir les rôles afin de bien conceptualiser les multiples paliers de sécurité du domaine d'affaires.

5. Améliorations possibles

5.1 Bonification de la solution

5.1.1 Environnement de test

Un environnement de test permettrait de grandement accélérer le processus de déploiement. En effet, une machine de compilation (build machine) s'assurerait, lorsque chaque développeur soumet du code, que son code compile, et que tous les tests réussissent. Par conséquent, le dépôt de code serait toujours dans un état stable et la solution progresserait davantage rapidement.

5.1.2 Automatisation du déploiement

Présentement, il existe divers outils, créés par l'équipe, afin de déployer la solution. Par exemple, le dépôt de code contient un script PowerShell permettant de créer la base de données dans SQL Server. Ce script créé une procédure stockée permettant de donner les droits d'administrateur à un usager. Cependant, un usager ayant ces droits est nécessaire à l'intégration de la solution dans son environnement. Il existe donc plusieurs étapes au déploiement qui pourraient être automatisée. Une automatisation de ce processus faciliterait l'intégration des développeurs et des intégrateurs dans l'écosystème *SimplETS*.

5.1.3 Feedback des utilisateurs

Actuellement, aucun utilisateur n'a pu utiliser la solution, puisque les interfaces graphiques n'étaient pas assez avancées. Ainsi, le feedback des membres de clubs étudiants n'a pas pu être pris en compte. Par conséquent, il est fort probable que les interfaces ne soient assez intuitives pour ceux-ci. Inclure ces étudiants dans le processus de création serait bénéfique à la plateforme.

5.1.4 Support de sérialisation NoSQL

Les bases de données relationnelles sont majoritairement utilisées comme support de sérialisation dans les systèmes de gestion. Cependant, ces bases de données offrent beaucoup trop de fonctionnalités pour une application qui doit les utiliser seulement comme support de sérialisation. De plus, afin de faciliter l'intégration d'une base de données relationnelle dans une application, une couche applicative supplémentaire est nécessaire, soit la couche de mappage objet-relationnel (**O**bject-**R**elational **M**apping). Cette couche nuit grandement aux performances, puisqu'elle doit faire le pont entre deux types de représentation des données.

Les bases de données NoSQL permettent de sauvegarder des objets au format JSON ou BSON; elles sont donc davantage optimisées pour la sérialisation du domaine d'affaire de l'application. De plus, ces bases de données n'ont pas de schéma; la validation des propriétés n'est donc pas dédoublée et est assurée par l'application.

5.1.5 Amélioration du cache

Présentement, les caches de l'application ont été implémentées par l'équipe et ne peuvent pas être découplées de l'application web. L'authentification par jeton nécessite ce cache; par conséquent, lorsque l'application web est recyclée par le serveur web (*AppPool Recycling*), les caches sont perdues et les jetons sont par conséquent invalidés. Afin de palier à cette lacune, il serait possible d'opter pour un serveur de cache tel que Redis. Ces serveurs de cache permettraient de ne pas se soucier des *AppPool Recycling* du serveur web.

5.1.6 Amélioration des interfaces

Aucun designer graphique n'était présent dans l'équipe actuelle; par conséquent, les interfaces manquent de contenu. Il serait judicieux de trouver un tel designer afin qu'il puisse faire évoluer l'intuitivité et l'utilisabilité de la plateforme. De plus, un professionnel de l'ergonomie cognitive pourrait grandement aider l'équipe à organiser le contenu de l'application. Celui-ci pourrait s'assurer que l'agencement des modules est conforme au mode de fonctionnement du cerveau; l'intuitivité de l'application serait grandement améliorée.

5.2 Pérennité

5.2.1 Transfert de connaissance

Une documentation en ligne est disponible sur le dépôt Git de la solution. Elle comprend un guide pour les développeurs et les utilisateurs. Un système de ticket est disponible afin de pouvoir garder une trace des bogues répertoriés.

5.2.2 Améliorer le wiki du développeur

Il serait intéressant de développer davantage la section portant sur l'utilisation des services REST utilisant l'extension de Google Chrome *Advanced REST Client*. Bien qu'une documentation soit générée automatiquement utilisant les APIs de l'application, certaines commandes importantes peuvent être envoyées aux services Web accélérant le temps de développement et de débogage.

5.2.3 Améliorer le wiki de l'utilisateur

Toutes les pages et les menus doivent être détaillés dans le Wiki utilisateur.

5.2.4 Trouver une équipe

Le club Cédille se montrait intéressé à prendre le relais, mais il est possible qu'il y ait des difficultés d'intégration entre notre architecture système et leurs infrastructures infonuagiques. Tout club étudiant faisant du développement Web pourrait s'approprier la solution et continuer le développement de l'application.

Annexe A : Points en suspens et décisions.xlsx

Annexe B : Liens utiles.pdf

Annexe C : Plan de Test Services web.xlsx

Annexe D : Wireframes (Balsamiq)

Annexe E : ERD