Le génie pour l'industrie

Rapport technique
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS LOG792

L'assurance qualité de Fiduciam

Gislain ARMAND
ARMG14059104

Département de génie logiciel et des TI

Professeur superviseur
Alain April

MONTRÉAL, avril 2016

Remerciements

Moi, Gislain Armand, tient à remercier les membres de Fiduciam, soit Luc Valade, Charles Valade et Frédéric Hanna, pour leur soutien tout au long de ce projet. Malgré le fait qu'ils étaient tous emballés par le fait d'avoir un module administrateur pour leur application, aucun d'eux ne m'a laissé tomber lors de ce projet. J'ai toujours pu compter sur chacun d'eux lorsqu'un problème est survenu et encore mieux, leur participation m'a réellement permis de produire un projet de qualité.

Ce projet marque la fin de l'aventure Fiduciam pour moi. Par contre, grâce à ces trois personnes, ce fut la plus belle expérience entrepreneuriale. J'espère que le logiciel produit pourra vous être utile.

L’assurance qualité de Fiduciam

Gislain Armand
ARMG14059104

Résumé

Dans le cadre d’un baccalauréat en génie du logiciel, un des cours obligatoires est le cours de projet de fin d’études. L’objectif de ce cours est de démontrer la maîtrise des connaissances acquises tout au long des études et de les adapter à un cas réel. En parallèle, une entreprise en démarrage a besoin d’un module administratif à une application qu’elle a mise en production le 1er janvier 2016.

Est-ce qu’il y a un projet plus réel que de développer une application pour un client qui a un besoin criant? Le cours était donc une opportunité de jumeler un projet personnel à un cours d’école. La problématique de ce projet est donc de voir comment les connaissances acquises au cours d’un séjour à l’École de technologie supérieure s’appliquent à un cas réel. L’objectif est donc de partir d’un concept, soit une idée dans la tête d’un client et de passer à travers le cycle complet de production d’un logiciel et de comparer la pratique avec la théorie. L’ÉTS est une excellente école. Par contre, logiciel est un domaine qui change très rapidement et l’hypothèse pouvant être faite est qu’il y a des différences entre la théorie et la pratique.

Suite à l’approbation du projet, une planification du projet a été faite en considérant les phases théoriques de développement d’un logiciel ainsi que les contraintes imposées par le cours telles que les livrables ainsi que la durée limite du projet. La première étape était donc de rencontrer le client et d’identifier ces besoins afin de produire un document de vision et un document de spécifications des requis. Ensuite, un document d’architecture du logiciel devait être produit avant de s’élancer dans l’implémentation. En théorie, le logiciel devrait ensuite est construit de A-Z sur papier, mais considérant le contexte, seul un document d’analyse fonctionnelle a été produit expliquant les problèmes majeurs identifiés ainsi qu’une solution. L’implémentation de la solution pouvait maintenant être faite. Une fois l’application complétée, elle pouvait être présentée et remise au client. Finalement, une analyse du projet a été faite afin d’être en mesure d’améliorer le cycle de production pour le prochain contrat.

Table des matières

Table des figures

Liste des abréviations, sigles et acronymes

Abréviation	Nom complet
ETS	École de technologie supérieure
API	Application programming interface ou interface de programmation
SQL	Structured query language ou langage de requête structurée
FrogPHP	"Framework" développé par Fiduciam
JSON	Javascript object notation ou notation d'objet en Javascript
Url	Uniform resource locator ou localisateur uniforme de ressource

Introduction

Technologies Fiduciam Inc. est une jeune entreprise composée de 3 jeunes entrepreneurs et d'un mentor. Fiduciam, un logiciel développé par l'entreprise a été conçu avec une intention précise, soit de permettre aux huissiers de justice d'administrer un dossier d'administration de saisie de revenu adéquatement en fonction du nouveau Code de procédure civil qui entrait en vigueur le 1er janvier 2016. Les quatre membres sont très fiers de l'application développée et la vision de l'entreprise a continué à grandir suite à cette première étape importante. Suite à la mise en production de l'application, il n'y a pas vraiment eu de ressources attribuées à l'assurance qualité de l'application dans son ensemble, spécialement au niveau serveur. Un processus d'assurance a été mis en place, par contre il n'est pas utilisé adéquatement.

En date du 1er janvier 2016, l'application n'avait pas de module administrateur qui permet de connaître l'état du système. En fait, les administrateurs devaient effectuer des requêtes SQL dans la base de données en production afin d'extraire les données qui leurs sont pertinentes, et ce à tous les jours. C'est un très bon moyen de supprimer des données critiques suite à une simple erreur de manipulations, surtout considérant que les administrateurs ne sont pas tous habiles avec les bases de données. Pour ce qui est des tests système, le seul moyen de les effectuer est en utilisant les interfaces utilisateurs et elles ne testent pas vraiment le système adéquatement.

Un module destiné aux administrateurs leur serait un atout pour permettre à cette jeune entreprise immature de s'améliorer.

L'objectif de ce projet est d'utiliser cette opportunité afin de leur développer un produit qui répond à leur besoin en passant par le cycle complet du développement d'un logiciel. L'analyse de leurs besoins devra tout d'abord être faite, pour ensuite faire la conception de l'application, puis l'implémentation de la conception et finalement valider le logiciel qui sera remis à l'entreprise. C'est une excellente manière de s'assurer que l'ensemble des connaissances acquises durant le baccalauréat en génie logiciel à l'École de technologie supérieure a été maîtrisé.

Chapitre 1 - Projet de fin d’études

1.1 Le cours

Suite à un stage 3, un étudiant de l’École de technologie supérieure en ingénierie du logiciel doit effectuer le cours projet de fin d'études en génie logiciel (LOG792). Ce cours a pour but de permettre à un étudiant de démontrer qu’il maîtrise les connaissances acquises dans l’ensemble des cours effectué dans les dernières années par l’étudiant. Pour ce faire, il doit adapter ses connaissances à un cas réel de son choix en fonction de ses intérêts.

L’étudiant est libre de choisir un projet, par contre le projet doit être approuvé par un professeur superviseur qui le suivra tout au long de la session.

1.2 L’opportunité

En tant que membre fondateur de Technologies Fiduciam Inc. Gislain Armand avait l’opportunité de développer un module complémentaire à l’application développée qui permettrait aux administrateurs de l’entreprise à prendre de meilleures décisions et qui leur permettraient de tester les limites de l’application développée.

Suite à une discussion avec les administrateurs, ce module serait un module très utile et qui arriverait à un bon moment dans la suite de l’entreprise. Pour un projet de fin d’études, ce module s’applique très bien puisqu’il permettrait de refaire un tour complet de l’ensemble des apprentissages acquis durant l’ensemble du baccalauréat en génie logiciel. Alors, la proposition de ce projet a été acceptée par le professeur superviseur Alain April.

1.3 Proposition de projet

Avant de commencer ce projet, il est important de faire une préparation au projet auprès du professeur superviseur afin de démontrer que le projet répond bel et bien aux critères du cours. Alors, un document de proposition de projet a été remis contenant un résumer complet de ce qui allait se produire durant les 4 prochains mois. La problématique et le contexte, l’objectif du projet, la méthodologie, les livrables ainsi que les techniques et outils sont expliqués brièvement. En annexe, le plan de travail est présenté, ce qui assure que le temps minimum à mettre sur le projet est respecté.

Concevoir un document de ce genre permet de donner une direction et une vision globale au projet. Les étudiants ne sont pas des experts en gestion de projet et il est très facile de perdre l’objectif de vue.

Chapitre 2 - Technologies Fiduciam Inc.

2.1 Contexte juridique

Le 1er janvier dernier, un nouveau Code de procédure civil entrait en vigueur. Ce nouveau code de procédure civil a pour objectif d'améliorer l'accès à la justice pour les Québécois et de favoriser une justice plus rapide et moins coûteuse. Pour ce faire, le Ministère de la Justice a décidé de privatiser certains services.

Un des services qui s'est vu privatiser est l'administration de la saisie de revenue par les huissiers de justice du Québec. Justicom Inc, est une entreprise qui se voit devoir offrir le service d'administration de la saisie de revenu à l'ensemble de ses clients. Donc, en préparation de cela, l'entreprise a décidé de développer leur propre logiciel maison.

2.2 Création du projet

Un des employés de Justicom Inc., Charles Valade, a fait appel à Gislain Armand, un étudiant en ingénierie du logiciel à l'École de technologie supérieure, et Frédéric Hanna, un diplômé en ingénierie du logiciel de l'École de technologie supérieure. Suite à plusieurs discussions, une entente a été conclue et le projet est né.

Initialement, l'entreprise voulait un logiciel pour eux-mêmes, par contre la décision finale a été de concevoir le logiciel en prévision de vendre un accès à d'autres Études d'huissiers de justice du Québec. L'entreprise Technologies Fiduciam Inc. a été créée.

2.3 Cycle de développement

Puisque l'entreprise se dirigeait dans la création d'un produit dont la qualité importe énormément et dont le logiciel continuerait d'évoluer dans le temps, un cycle de développement adéquat a dû être mis en place.

Tout d'abord, l'application web a été liée à la branche maître d'un serveur Git. Afin de développer cette application, deux autres branches ont aussi été créées, soit une branche d'assurance qualité et une branche de développement. Il y a aussi deux bases de données qui ont été créées, soit une pour l'application en production et une pour la version d'assurance qualité et de développement. De cette manière, l'application en production est totalement séparée de l'application en développement.

Toutes les nouvelles fonctionnalités sont développées sur la branche de développement, afin que les développeurs soient constamment sur la même version de l'application en développement. Lorsqu'une ou plusieurs fonctionnalités ont été implémentées, la branche de développement est migrée vers la branche d'assurance qualité. Sur cette branche, l'ensemble des fonctionnalités doivent être testé et lorsqu'elles sont approuvées, la branche d'assurance qualité est migrée vers la branche en production.

Chapitre 3 - Les besoins de l'entreprise

L'application de Fiduciam est maintenant en production. Évidemment, comme toutes petites entreprises, les besoins évoluent constamment et rapidement. Les nouveaux besoins devaient donc être identifiés.

Suite à l'acceptation du projet, il était crucial de rencontrer les parties prenantes afin d'identifier ces nouveaux besoins. Une longue rencontre de plusieurs heures a été organisée pour discuter de la vision de l'entreprise. Avant la rencontre, les prochaines actions à effectuer étaient très mal définies, puisque c'était la première fois que les administrateurs se retrouvaient dans une position où ils devaient administrer une application et non utiliser une application.

Dans la salle, il y avait l'ensemble des parties prenantes, soit les administrateurs et les développeurs. La discussion a d'abord été orientée vers les besoins des administrateurs, soient ceux qui doivent prendre les décisions face aux prochaines actions que l'entreprise doit prendre, puis elle a été orientée vers les besoins des développeurs, soient ceux qui doivent assurer la qualité de l'application et l'ajout des fonctionnalités.

De nombreuses idées en sont ressortisses. Plus la discussion avançait, plus les parties prenantes avaient une vision qui ne considérait aucunement les ressources disponibles, soit en temps disponible de développement dans ce contexte éducatif. La vision a été ajustée en conséquence.

Une fois la vision rendue réaliste, il était maintenant temps de définir les requis du logiciel qui serait à développer. Cette discussion fut un peu plus technique, par contre les parties prenantes ont été capables de suivre et d'ajouter les requis qu'ils jugeaient pertinents.

Tout au long de la rencontre, il y a eu de nombreuses notes qui ont été prises et beaucoup de schémas ont été dessinés sur un tableau blanc qui a grandement contribué à la communication des idées entre les différents membres. Deux documents ont été produits suite à cette rencontre, soit un document de vision et un document des spécifications des requis du logiciel à développer. Le premier document a été révisé par Charles Valade et le second par Frédéric Hanna.

Dans le document de vision, les sujets suivants sont abordés. Aborder ces sujets permet de mettre en contexte les personnes qui vont travailler sur le produit logiciel à concevoir.

- Positionnement de l'entreprise
- Parties prenantes et utilisateurs
- Vue d'ensemble du logiciel
- Caractéristiques

Le document de spécifications des requis logiciels, quant à lui, présente les sujets suivants. Ces sujets permettre de prendre la vision de l'entreprise et d'établir la base de ce que le logiciel doit faire.

- Les cas d'utilisation
- Les acteurs
- Les exigences
    - Fonctionnelles
    - Non fonctionnelles
- Contraintes de conception

Il est important de passer par cette étape puisqu'un logiciel de qualité devra être livré. Meilleurs sont ces documents, moins il y a de chances que les requis changent en cours de projet. Si un requis se voit changer ou mal compris, le coût de changement sera beaucoup élevé que s'il avait été défini à ce moment du projet.

Chapitre 4 - Architecture

Les deux documents produits précédemment ont permis de mettre sur papier la vision de l'entreprise ainsi que les requis du logiciel à concevoir. Avant de commencer à concevoir le logiciel, l'architecture globale devait être planifiée. Fiduciam impose plusieurs contraintes qui se doivent d'être respectées. De plus, beaucoup de systèmes devront interagir ensemble et donc un document d'architecture logicielle semble être très approprié pour la situation actuelle.

En prenant le temps d'analyser les contraintes de Fiduciam, il est facile de se rendre compte qu'une architecture de très haut niveau est imposée. L'entreprise n'est pas à sa première application et les applications qu'elle a conçues utilisent toutes le même "framework" du nom de FrogPHP. De cette manière, lorsqu'il y a une mise à jour à faire, elle est faite sur l'ensemble des applications. Elle tient donc à ce cette application s'adapte à cette structure.

Un document d'architecture a donc été produit. Le document présente une vue module du système et une décomposition de premier niveau de chacune de ces modules. Par la suite, une vue de composants et connecteurs est présentée pour démontrer l'interaction entre les modules. Puis, une vue d'allocation est analysée pour donner un aperçu de la manière à implémenter ce système. Suivant cela, il est possible de voir l'arbre d'utilité et les scénarios rattachés à ce dernier. Par ailleurs, une analyse détaillée des 9 scénarios différents, suivant la méthodologie ATAM, est présentée. Le rapport termine par une discussion des notions apprises et utilisées et procure une synthèse du travail réalisé.

Le document n'a pas été présenté ou révisé par les parties prenantes de Fiduciam. Par contre, la vue modules a été présentée afin de confirmer que la structure respecte la structure qui est présentement mise en place. La vue a été approuvée.

À ce moment, la manière dont la récupération des données en lien avec l'application à tester n'avait pas encore été établie, tout comme la structure des données. Par contre, Fiduciam s'engage à être disponible pour en parler lorsque le moment sera approprié. Frédéric Hanna a tout de même mentionné qu'un point d'entrée dans l'application sur la branche d'assurance qualité est en cours de développement afin d'être en mesure de répondre aux besoins de l'outil administratif.

Chapitre 5 - Configurations initiales

Il était maintenant temps de commencer à implémenter la structure telle que définie dans le document d'architecture. Avant tout, Fiduciam devait fournir l'ensemble du code source du thème acheté, un accès vers le projet FrogPHP ainsi que deux répertoires Git, soit un pour chaque système de l'outil administratif. De plus, Fiduciam a créé un point d'entrée dans l'application à tester qui permet d'avoir accès aux données de l'API. À ce moment, la seule requête possible à effectuer est une requête "GET" vers le lien suivant :

http://api-dev.fiduciam.quebec:81/v2/frogDefinition

Au moins, il retourne les informations concernant la structure des données à utiliser, soit un tableau pour les modèles, un tableau pour les points d'entrée ainsi que des informations en lien avec l'API (version et branche). De plus, une branche a été créer dans le projet FrogPHP afin que des modifications puissent être apportées sans avoir besoin de leur autorisation. Cette branche ne sera migrée qu'à la livraison du produit uniquement. De cette manière, les nouvelles modifications n'allaient pas affecter les projets en cours de l'entreprise.

L'étape de configurations initiales des systèmes est une étape cruciale pour la mise en route de l'application puisque tout le reste en découle. Meilleure est la structure de l'application, plus facilement le reste sera maniable. Alors, la première étape consiste en la création de l'application web. Un thème a été fourni, et une des recommandations faites fut de démonter le plus possible ce thème en réduisant au maximum les dépendances avec les librairies incluses, chose qui a été faite. Les librairies ont tout de même été gardées dans le projet puisque lorsqu'une d'elles devra être utilisée, seul un lien vers la librairie en question devra être ajouté. Le lien suivant pouvait être utilisé afin de se retrouver dans ce thème, soit le thème mis en ligne :

http://template.fiduciam.quebec/

Une fois faite, une première version sur le serveur Git de l'application a été enregistrée. Fiduciam pourra donc utiliser cette version pour tout prochain projet qu'elle envisage faire. La seconde étape est d'ajouter un singleton configurable qui permet de communiquer avec le point d'entrée qui donne accès aux données de l'API et par le fait même, préparer un autre singleton qui permet la communication avec le prochain système qui sera à mettre en place, soit l'API administrative. Par contre, ce singleton ne doit pas être configurable. Un second enregistrement sur le serveur Git a été effectué.

La troisième étape a été de créer la structure de l'API en intégrant FrogPHP au projet. Ce "framework" a été beaucoup plus facile à configurer que prévu. Lors de la remise du lien vers ce projet, des instructions très primitives avaient été fournies afin d'être en mesure de l'intégrer à ce nouveau projet et elles se sont avérées suffisantes. Le premier point d'entrée était maintenant accessible, donc un premier enregistrement de ce projet pouvait maintenant être fait sur le serveur Git de ce projet.

La dernière étape était de valider l'interopérabilité de tous les systèmes entre eux telle que décrite dans la vue d'allocation du document d'architecture. Initialement, il y avait quelques problèmes. Par contre, ce fut des problèmes mineurs qui n'ont pas été très difficiles à régler. Une fois tout enregistré sur les serveurs Git, la première étape importante de l'application était terminée et donc la première présentation à Fiduciam a été effectuée, ce qui permettait de valider que l'application se dirigeait dans la bonne direction en fonction de la vision de Fiduciam.

C'était le cas!

En voyant la bonne progression du projet, l'entreprise a décidé de mettre la branche principale de chaque répertoire Git en production. De cette manière, elle pourrait commencer à tester le tout au fur et à mesure que le projet avance, mais aussi elle pourrait commencer à utiliser l'application.

Chapitre 6 - Rapport d’étape

La moitié de la session approche, et il était temps de donner un suivi de l’avancement du projet au professeur superviseur du projet. Un rapport d’étape devait donc être remis, ce qui permettait de valider que le projet avançait bel et bien selon les prévisions faites en début de session.

Pour ce qui est de l’avancement de ce projet, il progressait à bon rythme et tous les livrables avaient été produits. Par contre l’ordre dans lequel ils ont été produits n’était pas le même que prévu. Initialement, une vision de comment le projet devrait être produit avait été envisagé en considérant le fait une des parties prenantes de Fiduciam allait travailler sur l’outil administratif. Par contre, la première rencontre a grandement modifié la vision de l’outil et donc tout le processus de développement a dû être ajusté. En fait, au lieu d’essayer de prendre de l’avance en mettant en priorité certain document ou même implémentation, la constatation que le projet devait être fait dans l’ordre adéquat a été faite.

Initialement, la suite du projet n’avait pas encore été planifiée puisque la première rencontre avec les membres n’avait pas encore été effectuée et les requis n’avaient pas été déterminés. Évidemment, les parties prenantes avaient une vision beaucoup plus large que le temps disponible afin de réaliser le tout, alors la portée a dû être réduite. Le rapport d’étape a d'ailleurs permis d’effectuer le plan de travail pour la seconde moitié de la session, et cette fois, la planification a été effectuée dans l’ordre adéquat sans essayer de sauver du temps.

En gros, la seconde moitié de la session était constituée de l’implémentation de la structure et des cas d’utilisation décidés dans la première moitié de la session. Avant tout, il y avait un document technique qui devait être créé afin de permettre un avancement plus rapide des algorithmes dont l’implémentation nécessitait une certaine analyse préliminaire.

Bref, le rapport d’étape a été mis à jour en considérant ce qui avait été planifié, produit et la direction que prend le projet suite à tout ce qui venant de se passer.

Chapitre 7 - Analyse fonctionnelle

Avant de s'élancer dans la programmation des différents algorithmes qui effectueront les tests auprès de l'API à tester, il est important de comprendre les différents problèmes qui risquent d'arriver lors de la programmation. De cette manière, lorsque viendra le temps d'implémenter les différents algorithmes, l'analyse aura déjà été effectuée et une solution recommandée aura aussi été élaborée.

Donc, un document d'analyse fonctionnelle a été rédigé afin de présenter les problèmes potentiels qui seront à résoudre, suivi d'une analyse et d'une solution recommandée pour chacun des problèmes.

Essentiellement, trois problèmes importants ont été identifiés. Le premier est la génération du premier arbre logique ainsi que son utilisation. Lorsque les tests seront effectués, tous les comportements permis doivent être effectués, et ce dans tous les ordres possibles. Le second problème est l'analyse des réponses reçues. Un arbre devra être généré à partir des réponses envoyées par l'API et cons, un arbre logique devra être construit à partir de ces informations. Le troisième problème est la visualisation de ce dernier arbre.

Chapitre 8 - Implémentation de l'application

À ce stade-ci, le projet est configuré, les systèmes sont en mesure de communiquer entre eux et tout l'aspect théorique du projet a été mis sur papier. La programmation des différents cas d'utilisation peut maintenant commencer.

CU01 - Visualiser les données pertinentes à un administrateur

Le premier cas d'utilisation à implémenter est la visualisation des données pertinentes à un administrateur. Avant tout, une communication avec les parties prenantes devait être établie. La liste des données à extraire n'avait pas encore été établie. Cette requête est arrivée à un mauvais moment pour l'entreprise. Chacun des membres était très occupé et ne pouvait donc pas y répondre. Alors, afin de ne pas prendre trop de retard et ne pas rendre ce cas d'utilisation comme un élément bloquant au projet, la structure a été mise en place.

Ce cas d'utilisation a été pensé de manière à prendre en considération le prochain cas d'utilisation qui est la visualisation des données pertinentes à un administrateur en détail. Donc, l'idée dernière était de mettre les données spécifiques dans des vues de la base de données puis faire le compte ("count") de toutes les lignes présentes dans cette vue lors de la demande. Donc, une donnée pertinente est associée à un titre et une vue.

Il ne restait plus qu'a implémenter ce concept et à acheminer les données vers l'application web en les fessant passer par l'API. Au niveau visuel, la figure 1 présente la vue qui permet de répondre au premier cas d'utilisation.

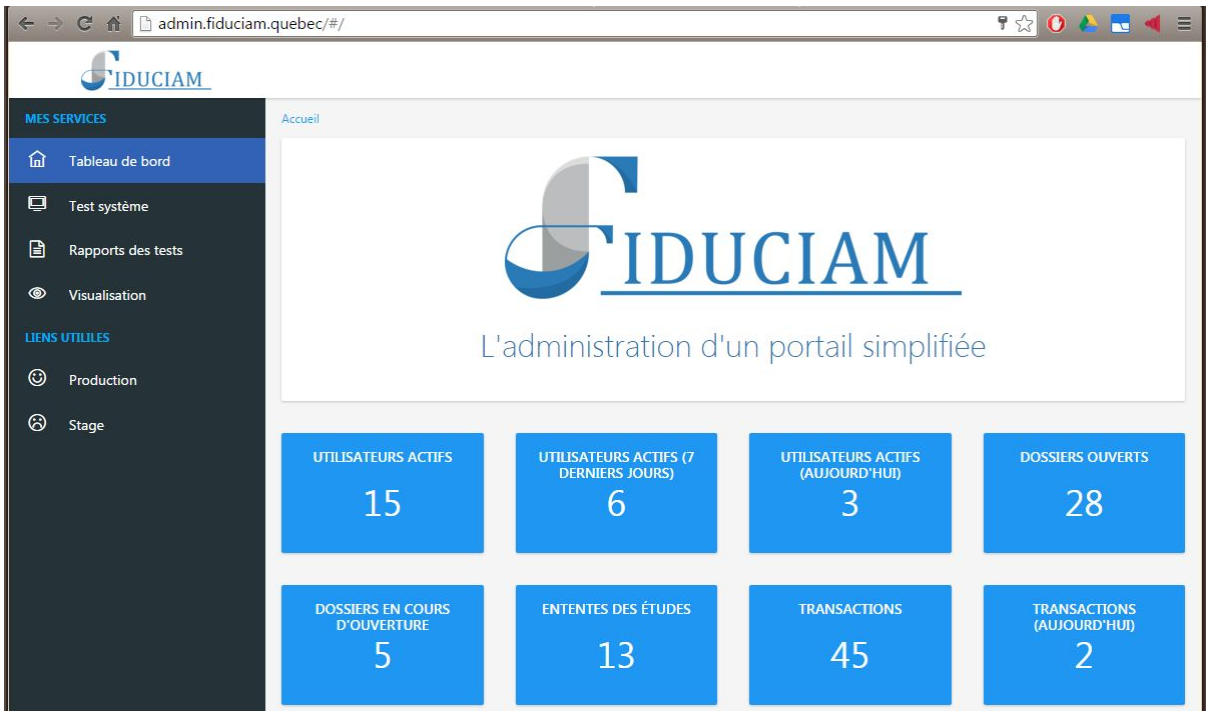

Figure 1 : Visualisation des données pertinentes à un administrateur

C’est lorsque l’utilisateur clique sur une des données qu’il a accès le mène à la visualisation de la donnée en détail.

Lors de la création de ce cas d’utilisation, la page d’accueil ne contenait que le logo. Alors, lorsque Fiduciam est arrivé avec la liste des données a affiché, l’entreprise a demandé de mettre la vue de visualisation des données pertinentes dans la page d’accueil, ce qui a été effectuer sans problèmes.

Alors, voici la liste des données pertinentes que Fiduciam voulait avoir dans le tableau de bord, nom choisi pour cette page :

- Nombre d’utilisateurs actifs
- Nombre d’utilisateurs actifs dans les 7 derniers jours
- Nombre d’utilisateurs actifs aujourd’hui
- Nombre de dossiers ouverts
- Nombre de dossiers en cours d’ouverture
- Nombre d’ententes avec des études d’huissiers de justice

Les vues ont été créées localement puis envoyées afin qu’elles soient intégrées à la base de données en production. Tout ce que Fiduciam eut besoin de faire fut de rouler un script. Il s’est exécuté sans aucune erreur, au plaisir de l’administrateur système.

Il ne restait qu’à tester l’application web avec ces données.

CU02 - Visualiser les données pertinentes à un administrateur en détails

À ce moment, les vues étaient présentes dans la base de données. Donc l’acheminement des données vers l’application web a été relativement simple à implémenter. Par contre, la difficulté rencontrée dans ce cas d’utilisation fut l’utilisation d’une librairie du thème afin de faire l’affichage des données détaillées.

Il a avait un tableau qui était approprié pour ce cas d’utilisation, surtout que dans l’exemple, il y avait une boîte de texte qui permet de faire une recherche dans la table en question. Donc, pour ce faire, l’exemple a été utilisé pour le projet à titre de référence afin d’intégrer la librairie au cas d’utilisation. La figure 2 présente le résultat final de la vue.

Il est important de noter qu’une des contraintes de la librairie est l’utilisation d’espace et de caractères spéciaux dans les titres des colonnes. Donc, puisqu’aucun de ces caractères n’était permis, ils ont du être remplacés. Les espaces ont été remplacés par un souligné tandis que les autres caractères ont simplement été supprimés du titre.

Figure 2 : Visualisation des données pertinentes à un administrateur en détail

Le format des données a été ajusté au niveau de l'API administrative (excluant les titres). Comme la figure 2 le démontre, on retrouve le titre de la donnée pertinente, suivi d'une boîte de texte puis du tableau. Au fur et à mesure que du texte est inséré dans la boîte de texte, les données dans le tableau sont ajustées en fonction de ce qui est recherché par l'utilisateur. De plus, si le tableau est trop volumineux, les données supplémentaires sont mises dans une page séparée au tableau. L'utilisateur peut donc naviguer à travers les données facilement.

Une fonctionnalité qui a été ajoutée est la possibilité de marquer le nom d'une table comme paramètre dans le localisateur de ressources uniformes (Url) de la vue. Suite à cela, si la table existe dans la base de données, les données seront affichées.

La finalisation de ce second cas d'utilisation marquait la fin d'une autre étape importante et donc l'avancement a encore été présenté aux parties prenantes. Elles étaient satisfaites du résultat. Par contre, lorsque la fonctionnalité supplémentaire a été mentionnée, une réaction d'inquiétude a suivi. Par contre après réflexion et argumentation, Fiduciam a tenu à la garder présente dans l'application.

CU03 - Effectuer un test système

Lors de la dernière présentation, Fiduciam a mentionné que le point d'entrée venait d'être terminé et que les données retournées pouvaient être utilisées afin de tester l'application et voici les données disponibles pour les modèles et les points d'entrée :

Modèle :

- Le nom de la classe
- Un tableau des propriétés
- Un exemple du modèle

Point d'entrée :

- Le nom de la classe
- Un tableau des méthodes disponibles ("get", "post", "put", "delete")

- Les paramètres pour chacune des méthodes
- Le modèle à utiliser

Donc l'idée est de tester toutes les méthodes de tous les points d'entrée dans tous les ordres possible. Juste le fait de penser à cela donne des frisons. Il y a 17 points d'entrée à tester, ce qui donne un total de 3,55 X 10^{14} possibilités (17!), et ce excluant les différentes méthodes, les modèles et les paramètres.

Dans le document d'analyse fonctionnelle, ce problème a été abordé et une solution a été proposée. Initialement, la solution proposée n'a pas été utilisée. Au lieu, l'arbre a été construit au complet en mémoire… ce ne fut pas une très bonne idée, mais c'était assez drôle de devoir aller dans le gestionnaire des tâches de Google Chrome pour fermer cette fenêtre. Un enregistrement sur une branche différence a tout de même été fait pour garder une trace cette belle implémentation.

Donc, l'approche recommandée dans l'analyse fonctionnelle a donc été implémentée et elle fonctionne très bien en plus d'être très raisonnable quant à l'utilisation des ressources de l'ordinateur. Il ne faut pas oublier que l'application doit rouler dans un navigateur web et que la puissance de l'ordinateur est inconnue.

Maintenant que l'application est en mesure d'effectuer un test comme voulu, le système doit être ajusté afin que l'analyse des données se fasse au niveau du serveur, soit au niveau de l'API administrative. Avant de commencer à tester l'API, un test est créé au niveau du serveur. Cela implique qu'un fichier texte est créé dans lequel la structure de l'objet d'un test est écrite en JSON. Utiliser cette approche permettait de modifier les données à tous les niveaux beaucoup plus facilement, puisque la réflexion est au coeur de la manipulation de toutes les données d'un test. Donc, lorsqu'un test est créé, les données en lien avec ce test spécifique sont enregistrées du même coup et un identifiant unique est renvoyé à l'application web. Elle devra l'utiliser afin d'envoyer des données en lien avec le test. Ensuite, au fur et à mesure que les réponses de l'API sont reçues par l'application web, elles sont aussitôt transmisses au serveur afin d'être écrite dans le fichier texte. Il est important de noter que seul le statut de la réponse est enregistré.

Un ajustement au cas d'utilisation a dû être effectué. Faire un test complet prenait un temps incroyable et lorsque le problème est venu aux oreilles des parties prenantes, une discussion suivit. La conclusion a été que l'utilisateur pourrait choisir les points d'entrée à tester et la manière dont ils doivent être utilisés. De cette manière, le temps est considérablement réduit et l'API ne se fait plus bombarder de requêtes inutiles. De plus, l'utilisateur est libre de tester l'API de la manière dont il le souhaite. La figure 3 présente l'interface qui permet de configurer le temps à effectuer.

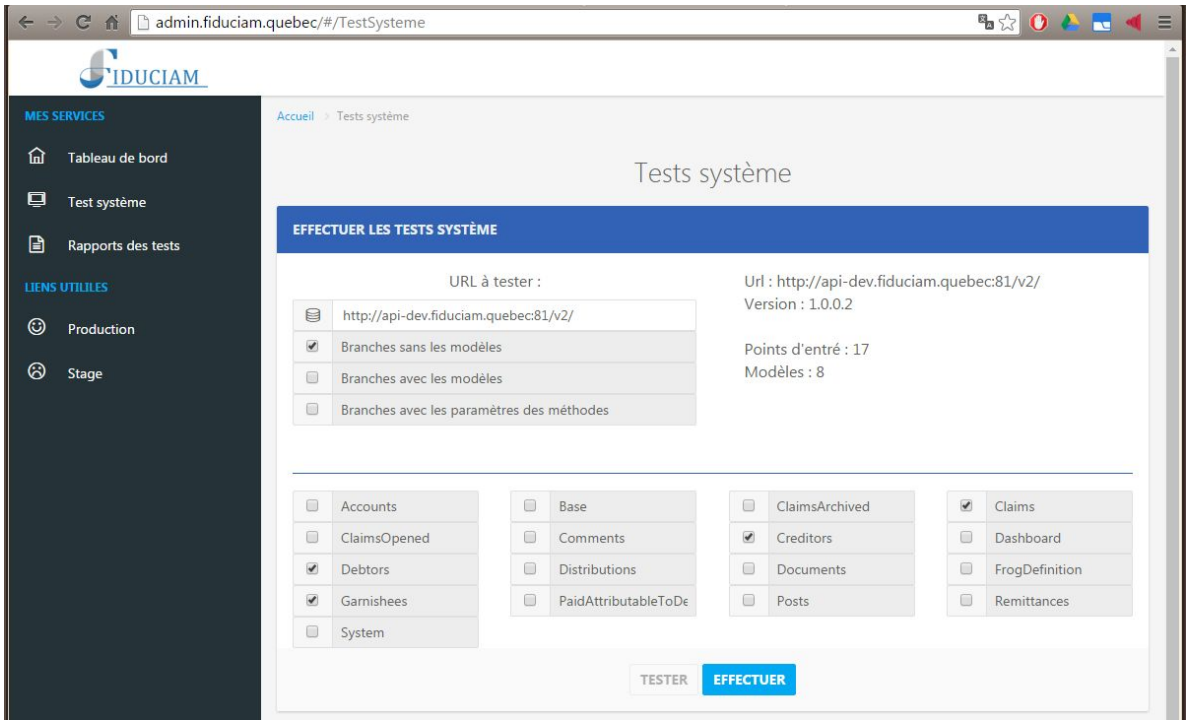

Figure 3 : Configuration d'un test système

Une fonctionnalité intéressante est que lorsqu'un test est effectué, l'utilisateur est libre de se promener dans l'interface comme il le veut. Par exemple, il peut aller visualiser un rapport d'un test système antérieur et revenir à celui en cours sans l'avoir interrompu. Aussitôt qu'il clique sur l'onglet "Test système" du menu à gauche, il retournera à la vue du test en cours, soit la vue présentée par la figure 4. De plus, l'utilisateur peut visualiser les données des points d'entrée et des modèles qui ont été envoyées par FrogDefinitionEndpoint sans interrompre le test en cours.

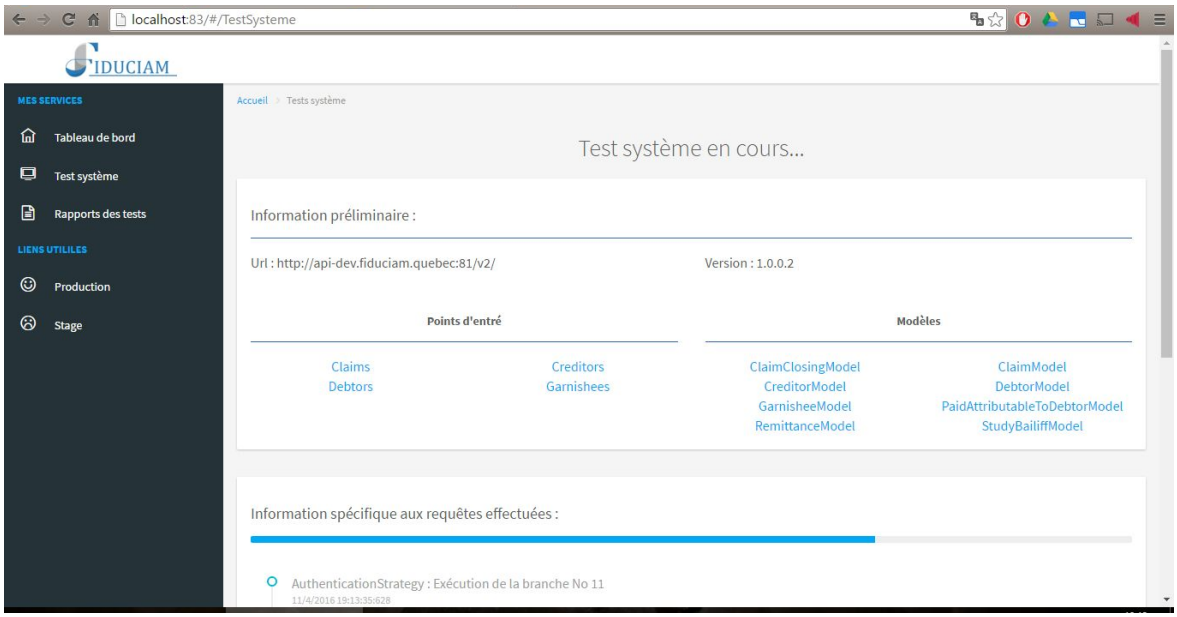

Figure 4 : Test système en cours

Une fois le test terminé, l'utilisateur est invité à visualiser le rapport du test ou simplement en recommencer un autre.

Contraintes

Lors de l'implémentation de ce cas d'utilisation, une constatation a été faite par rapport à l'API à tester. Il n'y a aucun standard en ce qui a trait aux réponses de l'API. Par exemple, lors de la création d'un dossier, la réponse retourne le numéro de dossier interne attribué au dossier en fonction des dossiers précèdent du client. Par contre, pour créer un débiteur, l'identifiant du dossier est requis, qui n'est pas le même que celui qui a été retourné lors de la création. Il ne peut donc pas être utilisé pour les prochaines requêtes.

Cette partie de la tâche a donc été abolie, puisque Fiduciam doit standardiser les requêtes afin d'être en mesure de tester adéquatement l'API.

CU04 - Visualiser le rapport d'un test système

Les données ont été enregistrées au niveau du serveur. Lorsqu'un utilisateur du système demande de visualiser un rapport, une analyse doit être effectuée afin d'en ressortir les informations puisque les données n'ont pas été triées dès leurs acceptations. Le faire de cette manière permet d'être beaucoup plus flexible quant à la manipulation des données et face à l'information pouvant être ressortir de chaque test système. Par contre, cela a un impact sur la performance du système.

Les informations pertinentes à afficher par rapport a un test système sont les suivantes :

- L'Url testée
- Le nombre de points d'entrée testés
- Le nombre de modèles
- Le nombre de branches par stratégies
- Le nombre de noeuds par stratégies
- Le nombre de requêtes effectuées

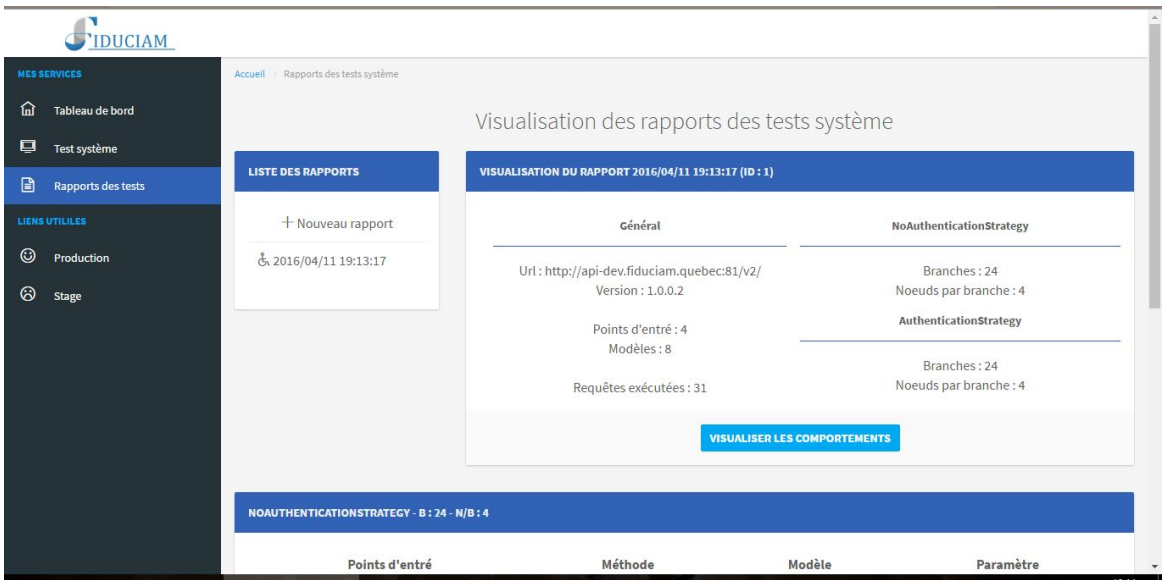

Figure 5 : Visualisation du rapport d'un test système

La figure 5 permet de voir comment ces informations sont présentées à l'utilisateur. De plus, un bouton est présent qui permet de visualiser l'arbre des comportements de ce cas d'utilisation.

CU05 - Visualiser l'arbre des comportements

Une modification a été effectuée au cas d'utilisation qui effectue un test système afin de donner plus de liberté à un utilisateur et de réduire considérablement le temps d'exécution d'un test système. Par le fait même, le cas d'utilisation de la visualisation de l'arbre des comportements doit être modifié aussi. Lors de la discussion avec Fiduciam lors de la modification du troisième cas d'utilisation, la modification de ce cas d'utilisation a aussi été abordée. La conclusion était qu'au lieu de visualiser la différence entre deux tests système, la visualisation d'un arbre des comportements devait être liée à un rapport. Donc, lors de la visualisation d'un rapport, la visualisation de l'arbre des comportements doit aussi être possible.

Avant de permettre l'affichage de l'arbre des comportements, une autre analyse doit être effectuée, soit l'analyse des réponses reçues. Pour ce faire, la solution recommandée dans le document d'analyse fonctionnelle a été utilisée. Donc une fois l'arbre reconstruit, il pouvait être acheminé afin d'être affiché.

Alors, dans le document d'analyse fonctionnelle, une librairie externe au projet a été recommandée afin d'effectuer la visualisation de l'arbre, par contre elle n'a pas été utilisée. Une des raisons était par le fait qu'elle était un peu trop restrictive. Il fallait que l'utilisateur puisse visuellement voir les cas problèmes. La façon implémentée au final n'est pas optimale elle non plus, par contre elle permet de voir facilement les cas problèmes à l'aide d'un code de couleur.

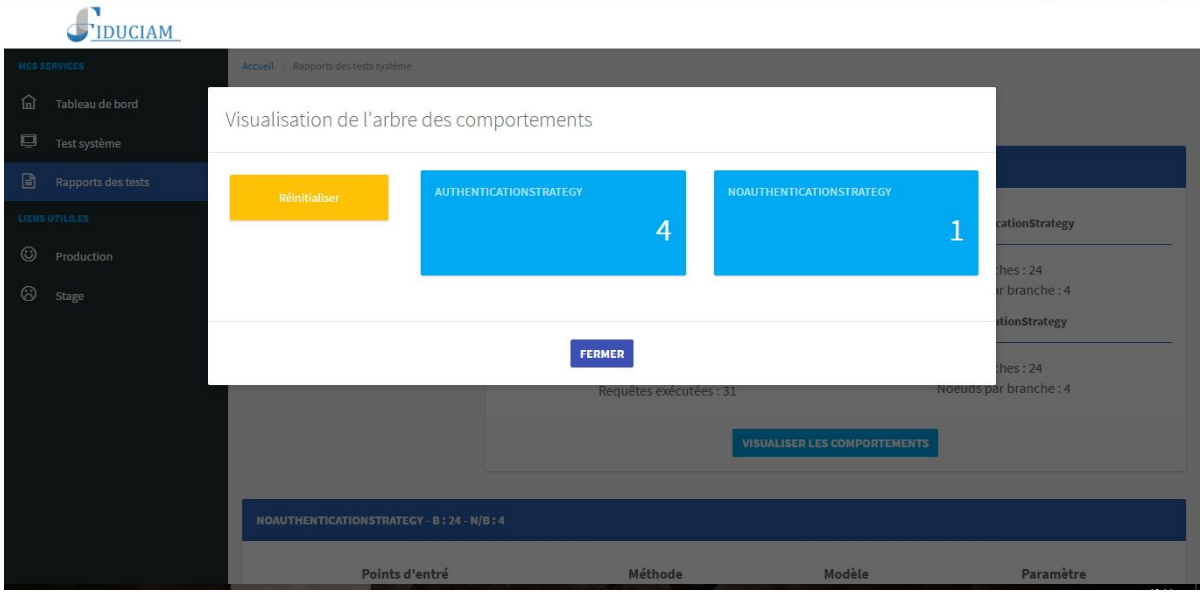

Figure 6 : Choix d'une stratégie

Comme le démontre la figure 6, la première étape à la visualisation de l'arbre des comportements est de choisir une stratégie. Le chiffre inscrit dans la case de la stratégie est la profondeur de l'arbre. Dans ce cas, il devient donc intéressant d'aller analyse la stratégie d'authentification. La seconde stratégie indique qu'il n'y a pas une branche qui s'est rendue plus basse que le premier niveau.

Donc, comme le présente la figure 7, nous pouvons voir le premier niveau de la stratégie d'authentification. Un code de couleur a été mis en place de la manière suivante :

- Vert : Comportement permis par l'API
- Rouge : Comportement non permis par l'API
- Jaune : Problème identifié dans la réponse de l'API

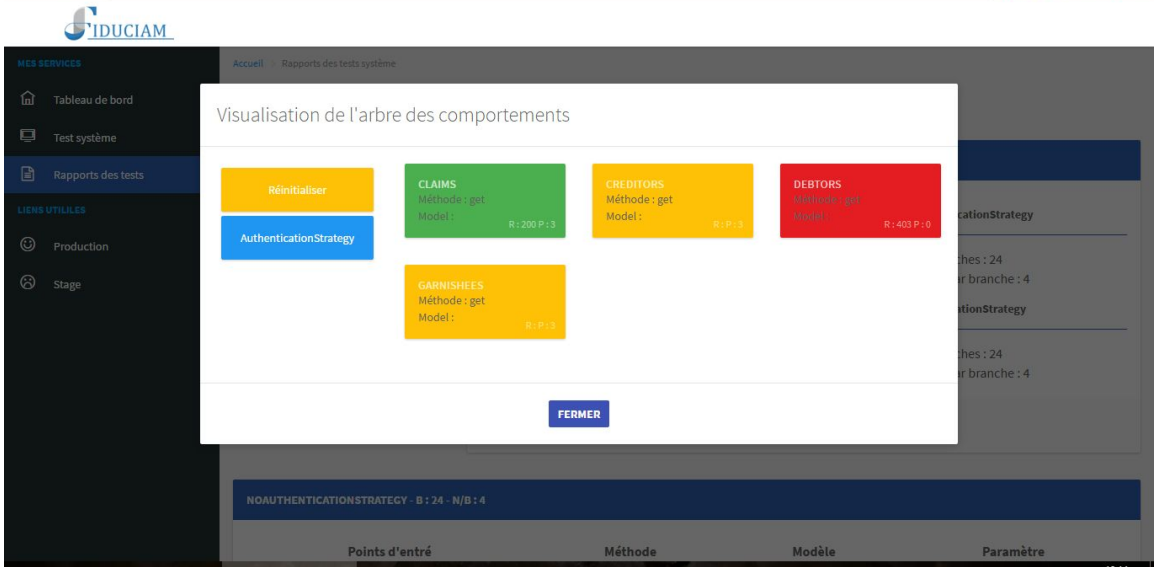

Figure 7 : Premier niveau de l'arbre de la stratégie d'authentification

De plus, pour chacun des noeuds, les informations suivantes sont présentées :

- Le nom du point d'entrée
- Le nom de la méthode
- Le modèle utilisé, s'il y a lieu
- La réponse reçue
- La profondeur du noeud

Figure 8 : Fin d'une branche

La figure 8 présente la fin d'une branche. L'utilisateur peut en tout temps se retrouver en regardant à gauche de l'écran. La liste des noeuds parcourus est présentée. L'utilisateur peut revenir en arrière en cliquant sur un noeud spécifique ou simplement sur le bouton de réinitialisation de l'arbre.

Lorsque cette fonctionnalité fut terminée, une petite présentation à Fiduciam a été faite afin de voir ce que l'entreprise en pensait. Elle était plutôt déçue puisqu'elle espérait être en mesure de visualiser l'arbre dans son ensemble sans avoir besoin de naviguer dans ce dernier.

Chapitre 9 - Présentation au client

L'application était maintenant fonctionnelle et complète; l'ensemble des cas d'utilisation était couvert. Par contre, avant de remettre l'application au client, il est important de valider que la vision de l'application ait été respectée et que les requis aient été comblés. Dans le cas de cette application, plusieurs requis ont changé. Toutes les modifications ont été notées afin de pouvoir les aborder lors de la présentation finale au client.

Alors, le temps venu de présenter le tout au client et d'effectuer une validation de tous les cas d'utilisation avec lui. Puisque les parties prenantes n'étaient pas disponibles lorsqu'il est venu temps de présenter l'application, la présentation a été fait par téléphone. Après tout, l'application est déjà disponible sur leur serveur. Lors de cette conversation téléphonique, quelques modifications mineures et une modification majeure ont été demandées.

Toutes les modifications mineures ont été effectuées. Par contre, pour ce qui est de la modification majeure, soit de pouvoir visualiser l'arbre dans son ensemble, la requête a été rejetée. Cela implique trop de temps, et la fin de session avance à très grand pas. Fiduciam, entreprise très collaborative, a compris.

Chapitre 10 - Discussion

Donc, après la livraison de l'application au client ainsi que son acceptation, il est important de faire une rétrospective sur tout ce qu'il s'est passé dans le projet, soit du moment où le projet a été accepté jusqu'à l'acceptation.

Plusieurs faits ont été constatés :

- Un client voit toujours plus grand que ce dont il a réellement besoin
- L'importance du document de vision et de spécifications des requis logicielles
- L'importance de la participation du client
- Les requis vont changer
- Le processus de développement logiciels doit être mis en place et stable
- Les documents n'ont pas à être maintenus

D'abord, lors de la première rencontre, soit celle qui a permis de mettre tout le monde sur le même niveau quant à la vision de l'entreprise, une idée en entraînait une autre. Lorsque l'on pense que, sensiblement, tout est possible en logiciel, il est important de prioriser certaines idées et d'estimer le coût de chacune d'elle. Une personne qui n'a jamais développé un logiciel n'a vraiment aucune idée de l'implication d'une idée. "Ah, juste ce requis aussi…", phrase qui revient très souvent. Toutes les fonctionnalités sont petites et simples pour un client. Donc, il est très important pour le fournisseur de l'application d'être ferme, et plus important encore, de connaître l'implication d'une fonctionnalité dans le système, même si les estimations ne sont pas exactes. C'est lui qui sera responsable de produire le logiciel et de respecter l'ensemble des contraintes (budgétaires, temps, requis, etc.).

Ensuite, une fois les requis choisi et défini, il est important de produire deux documents; un document de vision et un document des spécifications logicielles. Le premier document met les développeurs en contexte par rapport à l'entreprise. Il est donc plus facile de se mettre dans la peau du client et de comprendre pourquoi il a besoin d'un logiciel. Le second document permet de décrire les comportements de l'application aux yeux du client. C'est aussi un document très important puisque c'est ce document que les développeurs vont utiliser pour développer l'application. Ces deux documents permettent aussi de tracer une ligne sur ce qui a été conclu. Il y a de fortes changes que le client s'attendra tout me même à plus que ce qui a été convenu entre les deux partis.

L'importance de la participation du client. Malgré tout le temps consacré à la rédaction du document de vision et du document des spécifications logicielles, les attentes du client ne seront tout de même pas les mêmes que celles du fournisseur. Par le fait qu'à chacune des étapes, l'acceptation du client ait été demander a fait en sorte qu'aussitôt qu'un problème a été détecté, il a pu être corrigé. De cette manière, ce problème n'avait pas de répercussion sur le reste du projet. Par exemple, lorsqu'il est venu le temps d'effectuer l'implémentation d'un test système, des modifications aux requis ont été faites, et non pas suite à une demande du client, mais bien puisque le fournisseur a détecté un problème potentiel quant à

l'utilisation du logiciel. Ensemble, le client et le fournisseur ont pu trouver une solution adéquate qui respectait les besoins du client. De plus, le fait de faire des rencontres fréquentes avec le client et de lui présenter le logiciel permet au client de ne pas être trop surpris lors de la livraison finale et de réduire considérablement les modifications qui seront demandées et qui risquent de coûter très chers à modifier (temps, argent, ressources, etc.).

Les requis vont changer. Il est très difficile de prévoir tous les problèmes qui seront engendrés tout au long du projet, autant du côté du client que tu côté du fournisseur, spécialement si l'expertise d'un bord ou l'autre est faible. C'est justement lorsque les requis logiciels changent que l'importance de la communication avec le client entre en jeu. C'est cette communication qui permettra de passer à travers ces problèmes rapidement et qui permettra de rester sur les objectifs fixés en début de projet.

Lors de la conception de ce logiciel, le client avait un processus de développement mis en place. Visiblement, ce n'était pas leur premier projet de développement logiciel. Alors, une des recommandations était d'intégrer ce projet à leur cycle de développement. Ce fut une très bonne décision. Cela a permis d'intégrer beaucoup le client dans le projet, puisqu'il était au courant des mises à jour et pouvait offrir une rétroaction très rapide et pertinente sur les avancements. De plus, lorsqu'il y avait des retours en arrière à faire par rapport à l'implémentation des unités de code, le cycle le permettait très facilement, et donc il y avait beaucoup moins de perte de temps.

Finalement, les documents n'ont pas à être maintenus. Les requis vont changer et il y a beaucoup de décisions qui sont prises lorsqu'un problème survient. Constamment mettre à jour les documents entraîne une perte de temps incroyable et de toute façon, les parties prenantes sont trop occupées pour les lire. Par contre, il est important de noter les communications ainsi que les détails des communications. De cette manière, il y a des traces des changements. Après tout, c'est un logiciel pour une entreprise en démarrage et non pour le domaine militaire…

Conclusion et Recommandations

Dans le cadre d'un baccalauréat en génie du logiciel, un des cours obligatoires est le cours de projet de fin d'études. L'objectif de ce cours est de démontrer la maîtrise des connaissances acquises tout au long des études et de les adapter à un cas réel. En parallèle, une entreprise en démarrage a besoin d'un module administratif à une application qu'elle a mise en production le 1er janvier 2016.

Est-ce qu'il y a un projet plus réel que de développer une application pour un client qui a un besoin criant? Le cours était donc une belle opportunité de jumeler un projet personnel à un cours d'école. La problématique de ce projet est donc de voir comment les connaissances acquises au cours d'un séjour à l'École de technologie supérieure s'appliquent réellement à un cas réel. L'objectif est donc de partir d'un concept, soit une idée dans la tête d'un client et de passer à travers le cycle complet de production d'un logiciel et de comparer la pratique avec la théorie. L'ÉTS est une excellente école. Par contre, logiciel est un domaine qui change très rapidement et l'hypothèse pouvant être faite est qu'il y a des différences entre la théorie et la pratique.

Suite à l'approbation du projet, une planification du projet a été faite en considérant les phases théoriques de développement d'un logiciel ainsi que les contraintes imposées par le cours telles que les livrables ainsi que la durée limite du projet. La première étape était donc de rencontrer le client et d'identifier ces besoins afin de produire un document de vision et un document de spécifications des requis. Ensuite, un document d'architecture du logiciel devait être produit avant de s'élancer dans l'implémentation. En théorie, le logiciel devrait ensuite est construit de A-Z sur papier, mais considérant le contexte, seul un document d'analyse fonctionnelle a été produit expliquant les problèmes majeurs identifiés ainsi qu'une solution. L'implémentation de la solution pouvait maintenant être faite. Une fois l'application complétée, elle pouvait être présentée et remise au client. Finalement, une analyse du projet a été faite afin d'être en mesure d'améliorer le cycle de production pour le prochain contrat.

Tout au long de la production du logiciel, plusieurs constatations ont été faites dont les chargés de cours ont abordé. Par contre, c'est vraiment en le fessant qu'on s'en rend compte :

- Un client voit toujours plus grand que ce dont il a réellement besoin
- L'importance du document de vision et de spécifications des requis logicielles
- L'importance de la participation du client
- Les requis vont changer
- Le processus de développement logiciels doit être mis en place et stable
- Les documents n'ont pas a être maintenus

Ce projet a vraiment permis de faire un retour sur l'ensemble des connaissances acquises tout au long des études. Malheureusement, la durée du projet n'était que de quatre mois, donc la portée du logiciel devait être réduite, bien que c'était un projet destiné à une entreprise en démarrage. Alors, une des avenues pouvant être recommandée est de faire un logiciel, de la même envergure, mais pour le militaire et de comparer les processus exigés par ces deux types d'organisation, qui sont totalement une a l'opposée de l'autre...

Références

Justice Québec. 2016. « Nouveau Code de procédure civile » In Publications. En ligne <http://www.justice.gouv.qc.ca/francais/themes/ncpc/index.htm> Consulté le 20 janvier 2016.

S.I. Unik. 2016. « S.I. Unik : Logiciel de gestion des études de huissiers de justice » In Logiciels. En ligne <http://www.siunik.com/logiciels/jurisoft> Consulté le 20 janvier 2016.

Fiduciam. 2016. « PORTAIL FIDUCIAM - L'administration de la saisie de revenus simplifiée ». En ligne <http://Fiduciam.Quebec> Consulté le 22 janvier 2016.

Fiduciam. 2016. « Fiduciam : Application » In Se connecter. En ligne <http://App.Fiduciam.Quebec> Consulté le 22 janvier 2016.

Atlanssian : SourceTree. 2016. « Free Mercurial and Git Client for Windows and Mac | Atlassian SourceTree ». En ligne <https://www.sourcetreeapp.com/> Consulté le 21 janvier 2016.

JetBrains PhpStorm. 2016. « PhpStorm IDE :: JetBrains PhpStorm » In IDEs. en ligne <https://www.jetbrains.com/phpstorm/> Consulté le 21 janvier 2016.

Carnegie Mellon University en ligne <https://wiki.sei.cmu.edu/sad/index.php/Top_Level_Module_Uses_View> Consulté le 4 février 2016.

Carnegie Mellon University en ligne <http://www.sei.cmu.edu/architecture/tools/evaluate/atam.cfm> Consulté le 9 février 2016.

Gislain Armand. 2016. *Fiche de renseignements*, Montréal, 2 pages.

Gislain Armand. 2016. *Proposition de projet*, Montréal, 8 pages.

Gislain Armand. 2016. *Rapport d'étape*, Montréal, 14 pages.

Gislain Armand. 2016. *Document de vision pour Fiduciam*, Montréal, 11 pages.

Gislain Armand. 2016. *Document de spécification des exigences logicielles pour Fiduciam*, Montréal, 17 pages.

Gislain Armand. 2016. *Document d'architecture logicielle Fiduciam*, Montréal, 22 pages.

Gislain Armand. 2016. *Document d'analyse fonctionnelle pour Fiduciam*, Montréal, 11 pages.