

UNIVERSITE DE LILLE,
Campus Cité Scientifique,
59655, Villeneuve d'Ascq

INRIA – Lille Nord Europe,
Parc Scientifique de la Haute Borne,
40, avenue Halley,
59650, Villeneuve d'Ascq



RAPPORT DE STAGE

AMÉLIORATION DES PROCESSUS DE GÉNÉRATION
DE DOCUMENTS
AVEC L'OUTIL PILLAR

Réalisé par:

Asbathou Biyalou-Sama
Étudiant en
Licence 3 Informatique parcours Info

Tuteur d'entreprise:

M. Stéphane Ducasse
Chef de l'équipe projet RMOD

Sous la direction de:

Tuteur d'université:

M. Samuel Hym
Enseignant chercheur à
l'Université de Lille

REMERCIEMENTS

Mes remerciements vont à :

- M. Stéphane Ducasse, Directeur de recherche à INRIA, chef de l'équipe projet RMOD pour son encadrement, ses conseils et l'ambiance de travail qu'il mettait au sein de l'équipe afin d'assurer un travail efficace;
- M. Samuel Hym, Maître de conférences, enseignant chercheur à l'université de Lille pour son suivi, les retours et remarques apportées tout au long de ce travail;
- M. Guillermo Polito, Ingénieur de recherche, membre de l'équipe RMOD pour toutes ses connaissances qu'il m'a permis d'acquérir et son énergie dans le travail.

RESUME

Lors de mon stage au sein de l'équipe RMOD du centre de recherche INRIA, j'ai eu à travailler sur l'amélioration d'un outil de génération de documents "Pillar". Mon stage, sanctionné par un rapport de stage, a donc porté sur "l'amélioration des processus de génération de documents avec l'outil Pillar". Le plus souvent les articles sont rendus sous formats pdf et html pour consultation direct en ligne. Le format pdf grâce à Latex a un rendu contenant une table des matières, un menu permettant de naviguer, alors que le rendu html n'a pas de menu rendant ainsi la navigation moins facile. Il a fallu donc fournir un template proposant un menu construit à partir de la table des matières afin de pouvoir naviguer dans le document. En effet avec un visiteur on a effectué un parcours du document tout en notant les titres et leur niveau dans la hierarchie, ces derniers ont permis de constituer une table des matières qui a été inflaté dans un template préalablement élaboré à cet effet. La table des matières étant obtenue, nous avons réajuster le processus de génération pour prendre en compte cette dernière, offrant par là une meilleure stabilité de Pillar garantie par une batterie de tests. Nous avons ainsi pu mettre en pratique les designs patterns et le développement dirigé par les tests avec intégration continue.

SOMMAIRE

REMERCIEMENTS.....	1
RESUME.....	2
SOMMAIRE.....	3
INTRODUCTION.....	4
1. CONTEXTE.....	5
1.1 Présentation générale.....	5
1.2 Centre Inria Lille – Nord Europe.....	6
1.3 Equipe-projet RMOD.....	6
1.4 Stage.....	6
2. CONTRIBUTION.....	8
2.1. Mise en place de templates.....	8
2.2. Mise en place du générateur de table de matières.....	8
2.2.1. Modèle de tables de matières.....	8
2.2.3. Tests.....	10
2.2.4. Intégration du module.....	11
2.2.4.1. L’environnement d’intégration.....	11
2.2.4.2. Modifications apportées dans l’environnement.....	13
2.2.5. Tests fonctionnels.....	14
2.2.6. Solutions apportées.....	15
CONCLUSION.....	16

INTRODUCTION

L'informatique nous amène de plus en plus à traiter des informations, documents. Généralement en informatique, la répétition est considérée comme une chose à éviter. Dans la plupart des tâches quotidiennes allant de l'écriture de code à la rédaction de rapport, de livres, ou d'articles, on évite le plus possible de se répéter. En effet il arrive souvent qu'on ait besoin d'avoir un article ou rapport sous différents formats mais ayant toujours le même contenu. Dans ces cas, il est toujours souhaitable d'avoir un outil permettant de rédiger un document unique et de pouvoir décider du type de sortie tout en s'adaptant niveau interaction à chacun des formats. Ne pas se répéter tout en gardant un aperçu visuel agréable, adapté est donc important. C'est dans ce cadre que l'outil « Pillar » a été proposé par la communauté « Pharo » pour permettre de rédiger des livres, des articles, de la documentation de code dans une syntaxe assez souple et de pouvoir les avoir sous les formats pdf, html, epub. Ce outil a vraiment permis à divers acteurs d'avoir un article au format pdf mais aussi directement consultable sur un site sous forme de pages html. Au niveau des formats pdf, le passage par Latex permettait d'avoir un rendu approprié, standard. Par contre les documents html générés jusqu'à lors n'offraient pas un aperçu très adapté pour le web. Ainsi, il fallait améliorer les pages html proposés nativement dans Pillar. Il s'agissait entre autres d'avoir un menu représentant la table des matières du document afin de pouvoir naviguer aisément dans le document. Le travail s'est donc articulé autour de comment améliorer le processus pour qu'il prenne en compte les différents changements dont la table des matières. D'autres types de dysfonctionnement existants déjà ont dû être également corrigés afin d'offrir un outil plus stable.

La communauté Pharo est une communauté libre qui bien qu'étant assez jeune, est très active. Elle est toujours à la recherche de meilleures solutions pour les développeurs en général. L'équipe projet RMOD d'INRIA, qui supporte le langage Pharo, se sert de celui-ci pour travailler sur l'évolution des applications orientées objet, sur le devenir de ces applications. Passionné de l'environnement orienté objet et de tout ce qui est open source, nous avons donc voulu être au contact d'une équipe dynamique pour l'évolution de l'orienté objet. Ceci afin d'en apprendre un peu plus et compléter nos connaissances théoriques.

1. CONTEXTE

Nous allons ici présenter le cadre dans lequel notre stage s'est effectué. En effet notre stage s'est effectué dans un des centre de recherche de l'institut INRIA. Il s'agira de présenter celle ci sur le plan national, puis le centre de Lille au sein du quel on a effectué notre stage. Et enfin l'équipe projet dans laquelle nous avons travaillé.

1.1 Présentation générale

L'Institut National en Informatique et en Automatique (INRIA) est un institut de recherche français crée le 3 janvier 1967 sous le nom d'IRIA Institut de Recherche en Informatique et en Automatique. Les domaines de recherche autour desquels portent les projets des équipes d'Inria :

- Mathématiques appliquées, calcul et simulation ;
- Algorithmique, programmation, logiciels et architectures ;
- Réseaux, systèmes et services, calcul distribué ;
- Perception, Cognition, Interaction ;
- Santé, biologie et planète numériques.

Inria est composé de plusieurs centres de recherche autonomes sur l'étendue du territoire dont : Bordeaux - Sud-Ouest, Grenoble – Rhône-Alpes, Lille - Nord Europe, Nancy - Grand Est, Paris – Rocquencourt siège de l'institut, Rennes - Bretagne Atlantique, Saclay – Île-de-France, Sophia Antipolis – Méditerranée. L'Inria présente ainsi une administration assez décentralisée avec des directions dans les différents centres avec les services adéquats pour chaque centre.

L'institut oeuvre également à la formation à travers des partenariats avec plusieurs universités françaises Paris Saclay, Paris Diderot, Lille ... et des écoles, ainsi qu'avec d'autres laboratoires nationaux Il participe à des projets sur le plan européen à travers le consortium ERCIM dont il est membre fondateur.

1.2 Centre Inria Lille – Nord Europe

Le centre de recherche Inria de Lille a été inauguré en 2008. Il est situé à Villeneuve d'Ascq au parc scientifique de la Haute Borne. Il s'est donc installé dans un environnement hautement technologique avec le campus scientifique de l'université de Lille qui est juste à côté.

La directrice du centre depuis le 1 Mars 2017 est Isabelle Herlin. Le centre est composé de services administratifs et d'entités de recherche. On peut distinguer donc certains services administratifs dont :

- SRH: Service des Ressources Humaines
- SMGRH: Service de Management et Gestion des Ressources Humaines
- SAER: Service des Assitantes d'Equipes de Recherche
- SIC

La recherche dans le centre se fait au moyen d'entités qualifiées d'équipes-projets. Le centre Inria de Lille en compte 16 dont certaines sont délocalisées. C'est le cas de l'équipe Bonsai situé dans les locaux du laboratoire Cristal (Centre de recherche en Informatique et Signal Automatique) de l'Université de Lille. Le centre a donc divers partenariats avec les institutions de la place dont notamment l'Université de Lille, faculté des Sciences et Technologies et le laboratoire Cristal.

1.3 Equipe-projet RMOD

Notre stage s'est déroulé au sein de l'équipe RMOD ayant ses bureaux au troisième (3ème) étage du bâtiment B d'Inria. Il s'agit d'une équipe-projet d'Inria ayant pour activité « Analyses et construction de langage pour l'évolution d'applications orientées objet ». Elle est dirigée par Stéphane Ducasse, titulaire d'un HDR. L'équipe est constituée de membres permanents, d'ingénieurs de recherche et de doctorants. L'équipe travaille donc principalement sur l'évolution des applications orientées objet et sur la réingénierie d'applications. Elle se sert pour cela d'un langage purement objet, dynamique et réflexif « Pharo » dont elle a activement participé à la mise en place.

Ainsi une partie de l'équipe travaille sur l'évolution du langage, des concepts orientés objet, de l'environnement et outils autour du langage. L'autre partie fait de l'analyse de code pour sortir des métriques sur les systèmes informatiques de diverses sociétés. L'équipe participe

aussi activement à la mise en œuvre de ces recherches à travers une société qu'elle a mis en place Synectique, qui fait de l'analyse de code. Comme réalisations de l'équipe, nous pouvons citer Pharo, Moose, Roassal, Seaside, Pillar ...

1.4 Stage

Nous avons effectivement effectué notre stage au sein de l'équipe Rmod du 02 avril 2018 au 30 août 2018. Il s'agit d'un stage sur la conception objet axée sur Pillar. Pillar est une chaîne de production de documents (web, présentations, livres). Elle permet en fait d'écrire dans un format pivot et de générer plusieurs formats extérieurs. Notre mission était donc de :

- stabiliser et modulariser Pillar afin d'en extraire un noyau qui sera utilisé pour commenter les classes en Pharo
- définir un framework de configuration
- améliorer les aspects de publication de sites web
- améliorer les templates de génération d'HTML

Pillar est un projet démarré en 2006 menée par Damien Cassou, ancien membre de l'équipe Rmod. Le projet avait permis donc d'avoir un outil capable de générer des livres en pdf, en html, epub, des présentations, des slides. Il fut basé sur Pier qui avait également tout le modèle pour générer des documents dans différents formats. Toute la syntaxe telle qu'elle est connue actuellement avait été alors définie à savoir les titres, la mise en forme, inclusions de fichiers, ... Ainsi elle était utilisée par la communauté Pharo pour publier leurs articles. Elle ne présentait pourtant pas de ligne directrice permettant de suivre les différentes étapes du processus de génération, rendant alors difficile toute

tentative d'extension. Ainsi depuis quelques temps l'équipe s'est donné pour mission de refactorer le code de Pillar afin d'en dégager les principales étapes.

A notre arrivée, Pillar était donc en pleine refonte surtout sur le plan conception avec de nouveaux packages créés pour regrouper les étapes et les points d'extension possibles.

2. CONTRIBUTION

Mon travail au sein de l'équipe Rmod s'est articulé principalement autour de l'outil Pillar. Les missions effectuées devraient donc permettre d'assurer une meilleure stabilité de l'outil et d'apporter des fonctionnalités nouvelles. Ainsi nous avons commencé par des missions nous permettant de découvrir l'outil petit à petit afin de finalement mener des tâches nécessitant la compréhension complète du système.

2.1. Mise en place de templates

L'une des premières tâches fut de concevoir de nouveaux templates pour améliorer le rendu de certains types de rendu. En effet avec Pillar, certains membres de la communauté Pharo génèrent très souvent des livres sous format pdf généralement et en html quelques fois. La génération au format pdf était correcte, mais en html le rendu n'était pas satisfaisant puisqu'il n'y avait pas de table des matières (menu) permettant de parcourir aisément le livre. On a donc conçu un fichier template présentant un menu à gauche afin d'accueillir la table des matières d'un document.

2.2. Mise en place du générateur de table des matières

Ayant une idée du type de rendu espéré, on s'est donc mis à la conception du module qui allait nous permettre d'extraire la table des matières du document à générer.

2.2.1. Modèle de tables des matières

Extraire la table des matières revenait en fait à avoir une représentation de cette table en mémoire qu'on allait après envoyer au template. Pharo étant un langage purement objet, on a ainsi créé une classe représentant ce que c'est qu'une table des matières d'un point de vue programme. Une table des matières devait être composée d'éléments qui sont soit de simples titres ou des titres ayant des fils. On était donc parti sur un modèle avec 3 classes : une super classe pour encapsuler le compartiment général d'un élément de table des matières puis 2 classes filles représentant respectivement un simple titre et un titre composé d'autres d'autres titres. On essayait donc de mettre en œuvre le design pattern composite puisque celui qui allait utiliser la table des matières ne devrait pas se soucier du type d'éléments que c'est.

On a donc commencé à mettre en place cette architecture et à créer des objets correspondant. Mais pendant l'évolution du travail, nous avons constaté certaines contraintes que posaient ce modèle à 3 classes. On était obligé lors de la gestion de demander le type d'objets avant de faire certaines opérations, ce qui est contraire à la logique objet qui est de donner des ordres sans se soucier des traitements derrière. La différence fondamentale étant la présence ou non de fils, on a donc décidé de revenir à un modèle à une classe avec un attribut représentant une collection de fils. Un titre simple ayant donc une collection vide.

2.2.2.

Le mécanisme de construction de la table des matières consistait à parcourir les fichiers pillar constituant le livre et à extraire les titres pour en faire des éléments de la table des matières. On a donc visité les titres pour en faire des éléments de la table des matières. Il y avait aussi un cas particulier, celui des fichiers incluant d'autres fichiers. Il fallait donc visiter ces fichiers inclus pour en tirer les tables de matières. Pour ce faire, on a donc mis en place le design pattern visiteur. Celui-ci était chargé de visiter les titres et les inclusions de fichiers.

- La visite des titres se faisait assez simplement, ça consistait juste à créer un objet élément de table de matières avec sa référence de fichier dont il est issu et son niveau de titre.
- Pour la visite des inclusions fichiers, ça revenait à visiter le nouveau fichier inclus et de créer des objets éléments de tables de matières. Ces éléments issus du fichier inclus devraient être après insérés dans la table des matières du fichier parent, ceci tout en tenant compte des niveaux de titres afin d'avoir une bonne hiérarchie.

Avec cette architecture d'éléments à visiter, le visiteur devait donc en visitant les éléments garder une trace des derniers éléments visités pour pouvoir construire une bonne hiérarchie. Ce qui implique donc qu'il a fallu une structure assez adaptée pour permettre au visiteur de remplir au mieux cette mission.

2.2.2.1 Structure utilisée

Après certaines tentatives, il nous est apparu que c'était une structure de pile qui permettrait

de suivre tous les éléments déjà traités et d'avoir accès seulement au dernier élément visité. En effet à chaque fois qu'on visitait un élément, après l'avoir mis au bon niveau de hiérarchie dans la table des matières, on l'ajoutait à la pile. Il était donc au sommet de la pile. Par la suite, quand l'élément suivant était visité (titre ou inclusion de fichiers), pour savoir le niveau de hiérarchie où il fallait l'insérer on récupérait l'élément en haut de pile c'est à dire le dernier élément ajouté à la table des matières. Avec cet élément en haut de pile, on pouvait alors comparer pour savoir si l'élément suivant était un frère ou un fils dans la hiérarchie. Avec cette architecture mise en place, l'extraction de la table des matières s'est faite de façon assez intuitive.

2.2.3. Tests

Le processus de réalisation de ce module s'est fait avec les tests afin de valider les fonctionnalités implémentées. En effet après une première tentative, nous avons écrits des tests. Ces tests nous ont permis de nous rendre compte des failles qu'il y avait dans le code proposé. On a donc procédé aux corrections permettant de rendre ces tests positifs. Ainsi après ce premier essai et ces tests écrits, à chaque fois qu'il y avait un nouvel aspect à ajouter, on récrivait les tests afin qu'ils prennent en compte cet aspect. On procédait donc aux modifications nécessaires en ayant une manière de s'assurer de ce qu'on voulait. Les tests écrits nous donnaient donc une assurance par rapport à toute évolution qu'on voulait faire. Les tests ont été réalisés à deux niveaux.

- Vérification du modèle : ce sont des tests écrits pour vérifier certaines propriétés de la classe utilisée pour représenter les éléments de tables de matières. En effet dans l'écriture du module, il a fallu faire des comparaisons entre les différents éléments de la table des matières. On s'est rendu compte lors des comparaisons d'égalité que la redéfinition de l'égalité pour les classes `Pillar` était un peu naïve car tenant juste que deux objets issus de la même classe pouvaient être égaux. Il fallait que nos comparaisons aillent au-delà juste de ce aspect, alors nous avons redéfinis l'égalité sur la classe `PRToc` en tenant compte des valeurs des attributs de l'objet. En définissant ceci, il a fallu également définir la méthode `hash` qui va de paire avec l'égalité, utilisée

par d'autres méthodes de recherche. Ainsi nous avons écrits des tests pour vérifier cette parité de la méthode hash avec l'égalité et aussi que l'égalité tous les aspects escomptés.

- Vérification du visiteur : ici, ce sont des tests nous ayant permis de valider certaines fonctionnalités souhaitées. Nous avons effectivement faits divers scénarios permettant de vérifier le comportement du programme. Des cas les plus simples de fichiers ne contenant que des titres aux fichiers avec inclusions d'autres fichiers, ceci nous a permis de détecter à plusieurs reprises les anomalies de notre code. Il s'agissait donc pour ces tests de vérifier que pour un fichier donné, la table des matières extraite par le visiteur correspondait bien à ce qu'on espérait tant au niveau de la hiérarchie qu'aux valeurs d'attributs. Ces tests ci ont été mis en place après un premier essai, et après les avoir fait passé vert, à chaque tour d'ajout de code, on les relançaient pour s'assurer d'avoir rien cassé dans le système. On avait donc une assurance dans notre développement.

2.2.4. Intégration du module

L'intégration de ce module d'extraction de tables de matières au reste de l'outil Pharo s'est fait en plusieurs phases. Il y a d'abord eu la revue de code avec les encadrants afin de voir d'un point de vue conception et style de code, les améliorations qu'on pouvait apporter. En effet avant d'intégrer ce module dans le processus global de génération de documents, il fallait s'assurer le design utilisé pouvait être compatible avec ce qu'il y avait déjà. Ainsi nous avons eu avec l'encadrant à réécrire certains bouts de code, à rendre certaines utilisations de variables plus explicites, etc.

2.2.4.1. L'environnement d'intégration

Il s'agit ici de présenter le processus dans lequel va s'installer le module développé et à quel endroit devrait il être placé pour assurer une bonne continuité des processus. Le module développé à pour but d'avoir un menu de navigation dans les documents html générés. Nous allons donc analyser plus en détail le processus de génération de documents en html.

➤ Génération d'un document en html

Pillar base ses rendus sur le principe d'archetypes. Un archetype représente un modèle de rendu composé d'un template mustache et des fichiers externes nécessaires à celui ci, comme dans le cas du html des fichiers css, js ou images référencés dans la page html. L'archetype contient aussi des exemples de fichiers de configuration indiquant comment il s'utilise et quelques métadonnées nécessaires à son fonctionnement. La notion d'archetype est à rapprocher à celle de thème mais avec des d'autres choses en plus, fichiers de configuration, exemples, etc. Les archetypes contiennent donc des fichiers mustache qui seront utilisés plutard pour présenter le contenu brut des fichiers pillar. Etant des templates mustache, cela veut dire qu'elles contiennent des variables non encore définies, celles ci le seront après qu'elles aient été extraites des données de configuration comme le nom du site, l'url de base ou des fichiers pilla eux mêmes. Ainsi avant de générer un document, on décide de l'archetype à utiliser. Celui ci est installé dans un dossier nous donnant donc accès au template mustache. L'archetype est installé avec un fichier pillar.conf contenant une configuration par défaut. Ce fichier nous permet généralement dans le cas de fichiers html de définir le point de base dans la résolution de fichiers, le dossier de sortie des fichiers générés, la location du template, le titre du document, les auteurs, et d'autres métadonnées relatives au document. Certains éléments de cette configuration peuvent être écrasés par ceux qui seront définis dans les fichiers pillar. Les fichiers pillar sont ensuite parsés pour en extraire les différents éléments syntaxiques sous forme d'objets pillar. Ces objets sont en suite transformés en éléments html correspondant. Ils sont tous alors conservés dans une variable mustache par défaut appelée content. Celle ci sera par la suite insérée dans un dictionnaire contenant toutes les variables qu'il faut au template musatche. Ce dictionnaire sera donc passé au template mustache qui remplacera donc les variables par leur valeur. On obtient ainsi un fichier html prêt à l'emploi. Par défaut le répertoire de sortie des fichiers html générés est « _result/html ». Un serveur pourra être défini dans ce répertoire afin de visualiser ces fichiers sous forme de site web.

Figure

2.2.4.2. Modifications apportées dans l'environnement

L'intégration du module dans cet environnement a commencé d'abord par la mise au format approprié pouvant être accepté par les templates mustache. Notre module d'extraction de table des matières permet de récupérer un objet représentant la navigation du document. Pour après avoir un menu dans le rendu html, il faut que les valeurs de ce objet se retrouvent d'une quelconque façon dans le template mustache qui sera écrit en html. Ainsi, il a fallu implémenter sur ce objet une méthode permettant de le convertir en dictionnaire correspondant au format adéquat pour l'insertion dans un template mustache.

Ceci étant fait, on avait donc à modifier une étape de la génération html. Après que les fichiers aient été parsés, presque toutes les variables étaient alors regroupées pour une bonne écriture du fichier html à partir du template. En effet la variable récupérée « content » représente le contenu du fichier html. Sur la classe correspondant au rendu html, nous avons alors redéfini la méthode permettant de récupérer tous le contenu. Cette méthode redéfinie devait en plus de cela, ajouté au dictionnaire des variables, la variable « tableOfContents » avec pour valeur le dictionnaire généré à partir de l'objet table des matières récupéré du module d'extraction. Avec cette redéfinition, seule la classe output correspondant au html voyait un changement dans son flot d'exécution avec la recherche de la table des matières. Les classes intervenant dans le cadre de rendus pdf ou epub ne savaient donc pas qu'il y avait eu un quelconque changement. Des tests ont été également mis en place pour vérifier cet aspect là. Avec ces modifications apportées, la génération d'un document en html permettait d'avoir un rendu avec menu de navigation en ayant utilisé l'archetype proposant ce menu.

Intégration de la config Chrysal

Fig

2.2.5. Tests fonctionnels

Suite aux modifications apportées et l'intégration complètement effectuée du module de table des matières, nous avons procédé à des tests pratiques du module avec des livres existants. Nous avons écrits des tests unitaires nous permettant de valider les briques posées pour réaliser ce module. Malgré ces tests unitaires, il a fallu reproduire des cas d'utilisations pratiques qui nous ont permis de valider le module et de considérer des éventuelles corrections et évolutions.

La communauté Pharo a une association SquareBracketAssociates faisant office de maison d'édition de livres et documents traitant de Pharo. Ainsi avec l'encadrant nous avons récupéré certains livres pertinents du compte github de l'association. Nous avons procédé à la génération au format html de ces documents puisque à la base ces documents avaient été générés en pdf. On avait alors les fichiers pillar ayant servi à générer ces livres.

Le travail a donc consisté premièrement à installer le nouvel archetype de livres présentant un menu de navigation afin qu'on puisse avoir accès au template mustache qui sera utilisé pour le rendu. Ceci ayant été installé, le fichier de configuration pillar.conf devait être modifié. On a alors défini de nouvelles variables comme la base url nécessaire dans le cas de document html, le type de rendu espéré, le writer qui devait être utilisé pour ce cas (donc le `htmlWriter`), le template mustache à utiliser et plus important pour tester effectivement notre module de table des matières, le fichier de base qui devait indiquer le point d'entrée du document. Ceci représente le point à partir duquel on doit commencer à élaborer la table des matières. Il a fallu donc porter une attention particulière à la définition de la clé « `tocFile` » pour pouvoir effectivement générer la table des matières. Après tout ceci, nous avons effectivement pu générer les livres sous le format html.

Ces tests pratiques nous ont permis de faire quelques constats. Tous les livres présentaient effectivement un menu à gauche avec le contenu du livre dans un autre panel à droite. Mais les liens de navigation, les références vers les titres dans le document n'avaient pas d'effets. Les liens ne pointaient pas vers les titres correspondants dans le document. La navigation n'était donc pas effective. Aussi tous les chapitres d'un livre étaient générés dans un seul gros

fichier html rendant la page web extrêmement longue. Il a fallu de nouveau refaire une passe sur l'environnement pour corriger les problèmes.

2.2.6. Solutions apportées

Les corrections ont porté sur les liens de navigation. En fait concernant la grosse page html, il s'agit d'un soucis de choix sur lequel les concepteurs de pillar n'avaient pas décidé. C'était donc un problème connu qu'on avait décidé de gérer lorsqu'on aurait une stabilité de Pharo incluant le nouveau module.

Avec les liens de navigation, il a fallu s'assurer de la concordance entre le lien dans le menu et le titre dans le document. Donc parallèlement aux notions d'html, on devait créer une ancre sur un titre dans le document et ensuite avoir un lien pointant vers cet ancre. L'ancre est constituée d'une chaîne de caractères représentant la chaîne d'accès au fichier depuis la racine du site concaténée avec le titre lui même. Pour réaliser cette solution, nous avons agi à deux niveaux, en Pharo et dans le javascript de l'archetype. En Pharo, on a géré le lien vers l'ancre dans le menu, puisqu'on avait accès direct avec le module d'extraction. Les titres dans le contenu du document ont été gérés en javascript. Il aurait été plus compliqué de le faire en Pharo parce que du côté Pharo tout le contenu du document est encapsulé dans la variable content du template mustache. Faire une quelconque action sur les titres qui y sont contenus aurait amené à parcourir le dictionnaire correspondant au content afin d'opérer des actions sur ces derniers.

Alors en Pharo, on a ajouté une clé « href » au dictionnaire construit à partir de l'objet table des matières. Cette clé href était constituée de la référence vers le fichier dont est issu le titre suivi du titre converti en minuscules. Javascript avec sa capacité à manipuler le DOM html, nous a permis de cibler les titres du document. Nous avons donc procédé de la même manière que tout à l'heure pour construire une chaîne représentant de manière unique le titre. Avec ces deux manipulations effectuées du côté Pharo et du côté Javascript de l'archetype, les liens de navigation permettaient maintenant de manière effective d'atteindre les titres souhaités dans le document.

Sur la façon dont le fichier est généré, nous avons émis des réflexions sur un système nous permettant de générer d'une part tout en un dans le cadre de rendu pdf et d'autre part un fichier html pour chaque fichier pillar ou chapitre.

2.3. Déploiement automatique de documents html sous forme de sites web

La mise en place du module d'extraction de tables des matières a permis d'avoir des livres beaucoup plus faciles à consulter. La navigation se faisait donc comme pour un site avec des liens permettant d'atteindre d'autres éléments sur la page. Ces livres sont destinés à être disponibles via des sites web. Ainsi après qu'une personne ait rédigé et généré son livre au format html, elle allait maintenant déployer son livre sur un site web afin qu'il puisse être consulté par tous. Sachant dans la plupart des cas tout le processus qui allait suivre la rédaction d'un document, nous avons décidé d'automatiser les étapes de génération et de déploiement du livre afin de permettre une plus grande souplesse et facilité d'utilisation.

Cela a consisté donc à proposer des scripts permettant d'abord de générer le livre en html puis de le déployer en tant que site web. L'utilisateur final n'aurait juste qu'à rédiger son document et il le verrait automatiquement sur le web accessible à tous.

Ici, pour réaliser ces deux étapes du processus, nous avons opté pour les plateformes Github et Travis. Celles-ci permettant de faire gratuitement le déploiement automatique.

- Avec Github, on a exploité l'offre Github Pages qui permet d'héberger des sites web gratuitement à partir de votre compte github. Les sites web statiques sont ceux pris en charge. Et effectivement les livres étaient sous forme de pages html simples, sans dynamique. Github a ainsi servi de solution d'hébergement gratuit.
- Travis nous a permis l'automatisation des scripts. Il a été donc chargé de générer le livre au format html puis de le publier sur Github pages. En effet les scripts travis ont permis d'installer Pillar puis de générer le document en html. Après cela le livre généré était récupéré et déployé sur Github.

La communication entre Github et Travis devait se faire à travers un token généré par Github. Ce token était encrypté puis ajouté en tant que variable d'environnement dans Travis.

CONCLUSION

La mise en place de nouveaux modules dans Pillar est un travail en perpétuelle évolution puisque les besoins se font ressentir de jour en jour. Ainsi nous avons pu répondre à certains de ses besoins à travers les modules que nous avons mis en place. En effet la communauté a ressenti le besoin d'avoir des livres sous formats html avec présence de menu de navigation. Durant notre stage, nous avons eu à apporter des solutions à ce besoin à travers la mise en place d'un module de génération de table de matières d'un document. Nous avons effectivement conçu un archetype pillar proposant un menu de navigation. Le module d'extraction de table des matières permet d'avoir un objet qui est inflaté dans le template mustache permettant ainsi d'avoir une page html avec un menu de navigation du document. Après l'intégration de ce module dans le processus de génération de pillar, il a pu être testé et apprécié à travers la migration de certains livres de la communauté.

Cette période de travail au sein de l'équipe Rmod a permis de faire une passe sur pillar permettant de nettoyer l'environnement, de réorganiser le modèle objet de pillar, de repenser certaines méthodes de génération existantes. Nous avons ainsi pu faire une release de pillar incluant la génération de livres sous formats html avec navigation, une meilleure stabilité assurée par une meilleure couverture de tests, une documentation mise à jour. Ceci permettant donc une meilleure utilisation et exploitation de pillar pour gérer les documents, articles et livres. Il y a encore quelques étapes de la génération qu'on pourrait faire évoluer, leur conception. Egalement Pillar permettant de générer des pages web, on essaie d'exploiter cette fonctionnalité pour mettre en place un outil de génération complet de sites web statiques.

Cette immersion dans le monde de la programmation à travers ces tâches effectuées sur des projets réels a vraiment été une expérience nouvelle. Cette expérience nous a permis d'accéder à un autre niveau de connaissances, d'avoir de nouvelles manières d'appréhender la programmation. Ainsi sur le plan conception orientée objet, nous avons vraiment appris de nouvelles choses permettant de consolider certaines connaissances acquises à l'université. Nous avons par exemple eu une meilleure connaissance des designs patterns, une application réelle de ces derniers, les aspects à prendre en compte dans leur mise en œuvre. Nous avons pu mieux maîtriser des outils entre autres github pour gérer nos codes, travis pour l'automatisation de nos travaux.

BILAN

Ce premier contact avec le monde professionnel nous a permis de voir sous d'autres aspects le monde du travail. Notre stage s'est déroulé dans un centre de recherche, mais les missions et tâches en elles-mêmes n'étaient pas de la recherche. On a donc pu acquérir la culture de projets d'entreprise, le fait de trouver des solutions mais de les évaluer par rapport au projet. Il faut donc que les solutions trouvées aillent dans l'esprit du projet et soient durables. Nous avons acquis un bagage intellectuel sur la culture d'entreprise, les terminologies et outils de travail. Le travail au sein de l'équipe, le contact avec des membres de divers horizons, de cultures différentes nous a permis d'améliorer notre côté relationnel, communication. Ceci nous a permis ainsi de faire plus connaissance avec la culture française.

Une Licence en Informatique m'ouvre les portes d'un Master en Informatique. Après cela la question était de continuer en Doctorat ou d'intervenir en tant qu'ingénieur dans une entreprise. Mon stage n'a pas été un stage de recherche, mais le contact avec cette équipe de recherche, les discussions avec les doctorants, chercheurs nous amène à envisager un peu plus la poursuite vers une thèse de doctorat en Informatique. Ainsi nous envisageons faire un Master en conception avancée et infrastructures des applications afin d'étudier plus tard l'évolution des langages.

BIBLIOGRAPHIE