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
DU 03/04/2018 AU 31/08/2018

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

Réalisé par:

Asbathou Biyalou-Sama
Étudiant en
Licence 3 Informatique parcours Info

Sous la direction de:

Tuteur d'entreprise:

M. Stéphane Ducasse
Chef de l'équipe projet RMOD

Tuteur d'université:

M. Samuel Hym
Enseignant cherheur 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 prende 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

INTRODUCTION

L'informatique nous amène de plus en plus à traiter des informations. Généralement en informatique, la répétition est considérée comme une chose à éviter. Dans la plupart des tacĥes quotidiennes allant de l'écriture de code à la rédaction de rapport, de livres, ou d'articles, on évite le plus possible de se repéter. 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 l'exporter dans divers formats. Ne pas se repéter 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». Pillar permet 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. Cet 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 appropié, 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 meilleurs 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 oeuvrant pour l'évolution de l'orienté objet. Ceci afin d'en apprendre un peu plus et complété nos connaissances théoriques.

1. CONTEXTE

Nous allons ici présenté 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. Nous allons ensuite présenté le centre de Lille au sein du quel on a effectué notre stage. Et enfin l'équipe projet RMoD dans laquelle nous avons travaillée.

1.1. Présentation générale[1][2]

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. Inria présente encore une administration décentralisée, tout en voulant garder une mutualisation des services pour tous les centres pour une meilleure gestion administrative.

L'institut a divers partenariats avec plusieurs universités françaises Paris Saclay, Paris Diderot, Lille ... et des écoles. Il a également des partenariats avec d'autres laboratoires nationaux.

1.2. Centre Inria Lille – Nord Europe[3][4][5]

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

La recherche dans le centre se fait au moyen d'entités qualifiées d'équipes-projets. Le centre Inria de Lille compte 16 équipes-projets 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 présente ainsi divers partenariats avec les institutions de la place dont notamment l'Université de Lille - Sciences et Technologies et le laboratoire Cristal.

1.3. Equipe-projet RMoD[6]

Notre stage s'est déroulé au sein de l'équipe RMoD. Elle est située 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, directeur de recherche à Inria. L' équipe est constituée de membres permanents, d'ingénieurs de recherche et de doctorants. L'équipe travaille donc principallement sur l'évolution des applications orientées objet et sur la réingenierie d'applications. Elle se sert pour cela d'un langage purement objet, dynamique et réfléxif «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. **Projet[7]**

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 axé sur Pillar. Pillar est une chaine de production de documents (web, présentations, livres). Elle permet d'écrire dans un format pivot et de générer plusieurs formats extérieurs. Notre mission a été 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 est basé sur Pier qui avait déjà le modèle pour générer des documents dans différents formats. La syntaxe pillar définie, permettait de définir les titres, de faire la mise en forme, d'inclure des fichiers, …

Pillar était alors d'une grande utilité pour la communauté Pharo. L'outil présentait malgré tout quelques points d'amélioration. L'équipe s'est donc mis dans une démarche de refactorisation de Pillar.

Ceci afin de le rendre plus stable et ouvert aux extensions. A notre arrivée, Pillar était donc en pleine refonte 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 principallement 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 commencer par des missions nous permettant de découvrir l'outil peu à peu afin de plutard mener des taĉhes nécessitant la compréhension complète du système.

2.1. Problématique

Pillar est utilisé pour générer des documents sous divers formats. Les formats les plus utilisés sont le PDF et le HTML. La génération en PDF se sert de Latex pour avoir le rendu final. Ainsi plusieurs aspects dont une table des matières était gérée par Latex. Les documents générés en PDF étaient donc de qualité satisfaisante. Pour la génération en HTML, le rendu n'était pas attrayant. Il n'y avait pas de menu de navigation permettant de parcourir le document. Ces documents générés en HTML sont destinés à être publiés sur le web. Mais il y avait pas un moyen assez souple de le faire.

Au niveau du processus de génération, il y a aussi quelques problèmes. La modification d'un seul fichier entraîne la régénration de tous les fichiers pillar du projet. Les éventuelles erreurs qui subviennent lors de la génération sont difficiles à suivre. En effet le système de report de Pillar n'affiche parfois pas les erreurs ou les affiche mal. Ceci rends difficile la traque des erreurs.

Ainsi nous avons été amenés à mettre en place des solutions pour pallier à ces problèmes.

2.2. Mise en place de templates

L'une des premières tâches fut de concevoir de nouveaux templates pour améliorer le rendu HTML des documents. Les documents étaient pour la plupart du temps générés en PDF mais aussi en HTML. La génération au format PDF était correcte. Mais en HTML le rendu n'était pas satisfaisant. En effet la naviguation dans le document n'était pas possible. Le template HTML ne présentait pas de menu de naviguation du document. On a alors conçu un template mustache présentant un menu de naviguation à gauche afin d'accueillir la table des matières du document. Ce sera un menu avec des liens vers les titres dans le document.

```
<!DOCTYPE html>
<html lang="en">
 <head>
 …..
    <title>{{title}}</title>
    <link href="/style.css">
 <head>
 <body>
   …
   …
  {{content}}
 </body>
</html>
```

Figure 1 : Structure minimale d'un template mustache HTML

Le contenu proprement dit du document sera présenté à droite. Il s'agit d'un template qui par la suite sera ajouté comme thème par défaut pour le rendu HTML.
Le menu doit accueillir une table des matières. Il a fallu alors extraire cette table des matières du document souhaité.

2.3. Mise en place du générateur de table de 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.

2.3.1. Modèle de tables de matières

Extraire la table des matières revenait en fait à avoir une représentation de cette table en mémoire. Celle ci devait ensuite être envoyé au template mustache. Nous avons eu une approche objet pour représenter une table des matières dans le langage Pharo. Ce dernier étant purement objet. Une table des matières devait être composé d'éléments qui sont soit de simples titres ou des titres ayant des fils.
On était donc partit sur un modèle avec 3 classes : une super classe pour encapsuler le compartement général d'un élément de table de 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.

2.3.2. **Implémentation**

La construction de la table des matières consistait en une phase de parcours des fichiers pillar constituant le livre. Pendant le parcours, il fallait extraire les titres pour en faire des éléments de la table des matières. Pour ce faire, on a donc mis en place le design pattern visiteur. 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 a fallu alors visiter ces fichiers inclus pour récupérer des éventuels titres. Le visiteur mis en place a ainsi permis de visiter les titres et les inclusions de fichiers.

- La visite des titres s'est faite assez simplement, cela a consistét juste à créer un objet élément de table de matières avec la référence vers le fichier dont il est issu et son niveau dans la hiérarchie;
- Pour la visite des inclusions fichiers, cela revenait à visiter le nouveau fichier inclus. Cette visite a permis de créer des objets éléments de tables de matières. Ces éléments issus du fichier inclus étaient insérés dans la hiérarchie. Ils étaient insérés en ayant connaissance du dernier titre ajouté. 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 en visitant les éléments garder une trace des derniers éléments visités. Cette trace devait lui permettre de construire une bonne hiérarchie. Il a fallu une structure assez adaptée pour permettre au visiteur de remplir au mieux cette mission.

2.3.2.1. **Structure utilisée**

Nous avons essayé avec des structures classiques comme les listes, mais sans succès. Ceci a plutôt complexifié le visiteur. On s'est rendu de plus en plus compte qu'il faudrait des structures séquentielles avec des certaines spécificités. L'une de ces spécificités était le fait de n'avoir directement accès qu'au dernier élément ajouté. Ceci correspondait donc aux propriétés de la structure de pile. En effet à chaque fois qu'on visitait un élément, deux cas se présentaient. Lorsqu'il s'agissait du premier titre du docuemnt, il était ajouté à la pile. Ce titre était mis à la racine de la hiérarchie. Pour tout autre cas, c'est-à-dire pas le premier élément, il fallait faire une comparaison de niveaux.

En effet le dernier élément ajouté à la pile était récupéré. On faisait alors une comparaison entre le nouveau titre à ajouter et celui récupéré. Cette comparaison permettait de savoir si le nouveau titre devait être ajouté comme fils, frère ou parent du dernier titre dans la hiérarchie. Après cela, le titre était placé au sommet de la pile en tant que dernier élément visité.

Avec cette stucture, l'extraction de la table des matières s'est faite plus simplement. Avec de la récursivité, cela semblait presqu'automatique.

2.3.3. **Tests**

Le processus de réalisation de ce module s'est fait avec des tests. Ces tests ont permis de valider les fonctionnalités implémentées. Nosu avions mis en place notre modèle. On a essayé de valider notre modèle avec quelques essais d'implémentation. Une fois ceci effectué, on était sur de notre modèle. Nous avons ainsi procédé à la rédaction de 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.

Après ces premières tentatives, nous avions effectivemment des tests réfletant ce qu'on espérait. Ainsi à chaque nouvel aspect, des tests étaient écrits pour pouvoir ce aspect.

On procédait donc aux modifications nécessaires. Ceci tout en ayant un moyen de vérifier nos modifications. Les tests écrits nous donnaient donc une assurance par rapoort 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 ds 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. Par défaut l'égalité ne tenait compte que de la classe d'origine. Si deux objets étéient issus de la même classe, alors ils étaient égaux. Nos comparaisons ne devaient pas s'arrêter à ce unnique aspect. Alors nous avons redéfinis l'égalité sur la classe représentant les éléments d'une table de matières.

Cette nouvelle définition tenait 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 recheche.

Ainsi nous avons écrits des tests pour vérifier cette parité de la méthode hash avec l'égalité et aussi que l'égaité 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 de notre implémentation. Des cas simples de fichiers ne contenant que des titres ont été testés. Les fichiers avec inclusions d'autres fichiers ont également été testés. 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 esperait tant au niveau de la hiérarchie qu'aux valeurs d'attributs. Ainsi pour les inclusions cycliques des erreurs étaient déclenchées. Nous nous sommes vraiment servis de ces tests pour avancer sereinement dans de le développement.

  A plusieurs reprises ils nous ont permis de détecter nos erreurs en essayant de mettre place une nouvelle fonctionnalité.

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 faite en plusieurs phases. Il y a d'abord eu la revue de code avec les encadrants. Ceci afin de revoir la conception, le style de code et d'autres points. En effet avant d'intégrer ce module dans le processus global de génération de documents, il a fallu s'assurer que le design utilisé pouvait être compatible avec ce qu'il y avait déjà. Ainsi nous avons eu avec l'encadrant à réecrire certains bouts de code, à rendre certaines utilisations de variables plus explicites, à améliorer quelques aspects.

2.2.4.1. **L’environnement d’intégration**

Il s’agit ici de présenter l’environnement dans lequel va s’installer le module développé. On pourra ainsi savoir où le placer pour assurer une bonne continuité du 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. Dans le cadre de rendu HTML, il s’agit de fichiers css, js ou images référencés depuis le template mustache. L’archetype contient aussi des exemples de fichiers de configuration indiquant comment il s’utilise et quelques métadonnées nécessaires à son fonctionnement. Ainsi nous pouvons voir un fichier «pillar.conf» et un template HTML à la figure 2. La notion d’archetype est à rapprocher de celle de thème mais avec des éléments en plus. Il s’agit de fichiers de configuration, d’exemples, etc.

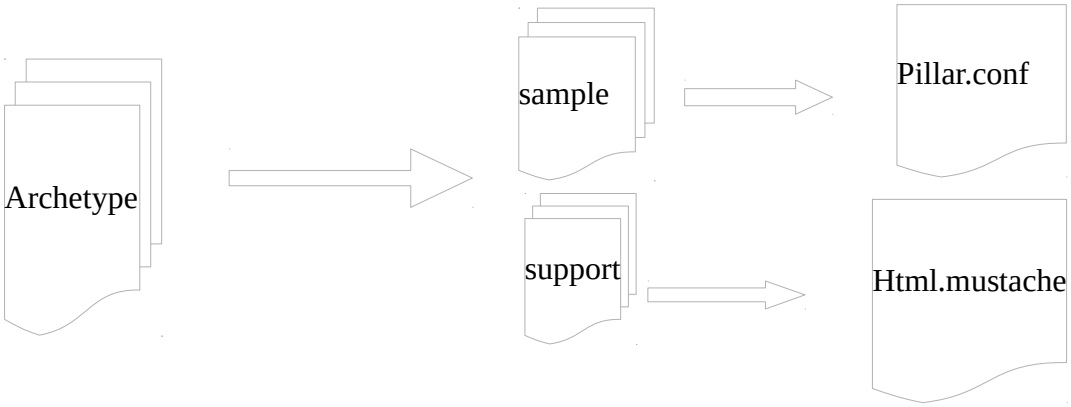

Figure 2 : Structure d'un archetype Pillar

Les archetypes contiennent donc des fichiers mustache qui seront utilisés plutard pour présenter le contenu brut des fichiers pillar. Les templates mustache contiennent des variables non encore définies comme c'est le cas de «title» et «content» à la figure 1. Celles ci le seront après qu'elles aient été extraites des données de configuration. Les données de configration peuvent être le titre du livre, les auteurs, le chemin d'accès principal, le chemin vers le template à utiliser. 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 templates HTML de définir le point de base dans la résolution de fichiers, le dossier de sortie des fichiers générés, la localisaion 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.

L'installation de l'archetype étant faite, la phase de build en HTML pouvait être lancée. Cette phase commencait par le parsage des fichiers pillar. Ceci permettait de reconnaître la grammaire pillar et de construire les objets pillar correspondants. Ces objets sont en suite transformés construisant un arbre HTML avec les éléments correspondants.

Ils sont 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. Nous pouvons constater à la figure 4 que le dictionnaire

effectivement passé contient le «content» de la page. 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 ainsi mis en place dans ce répertoire afin de visualiser ces fichiers sous forme de site web.

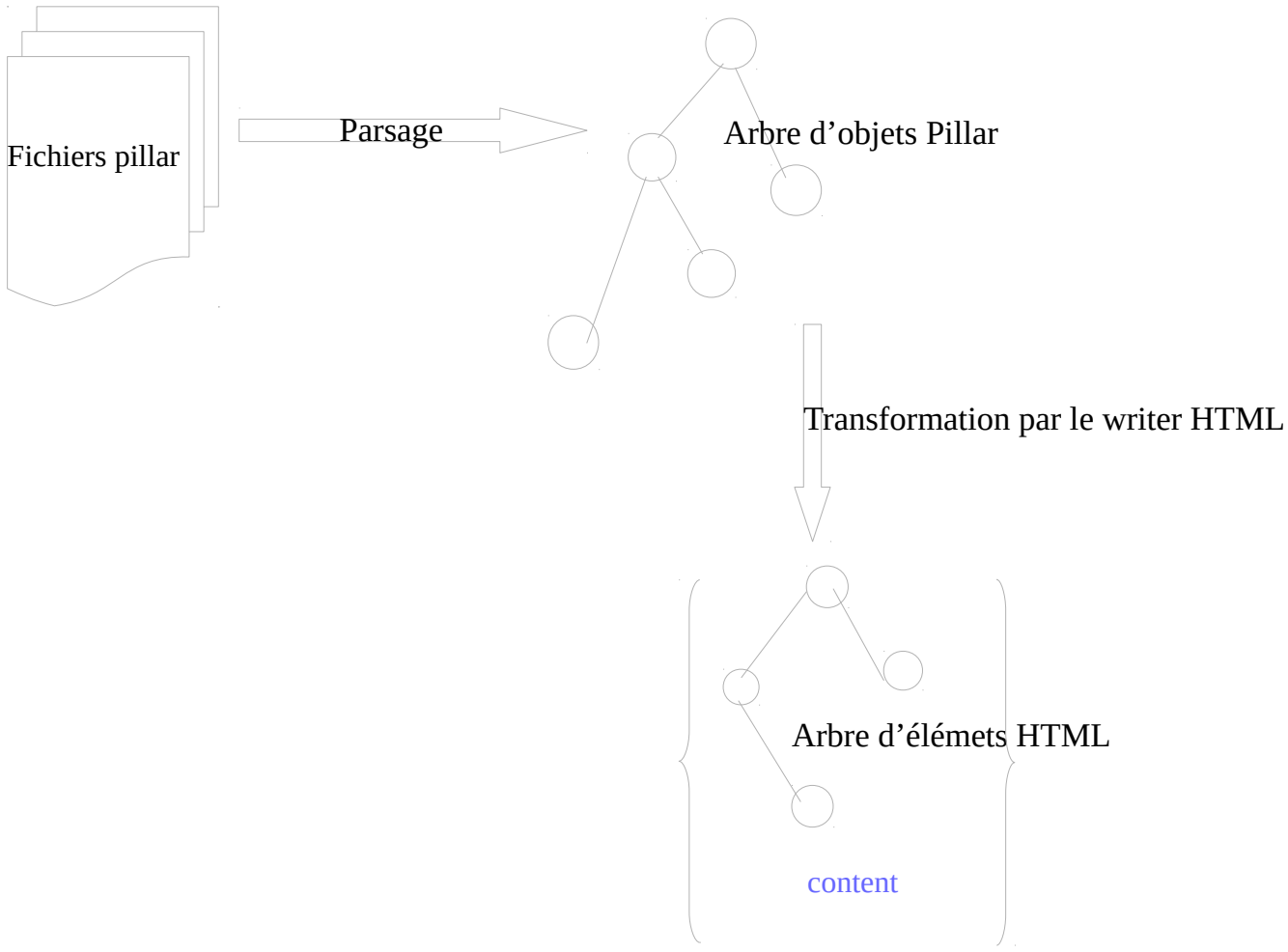

Figure 3: Processus de génération des éléments HTML

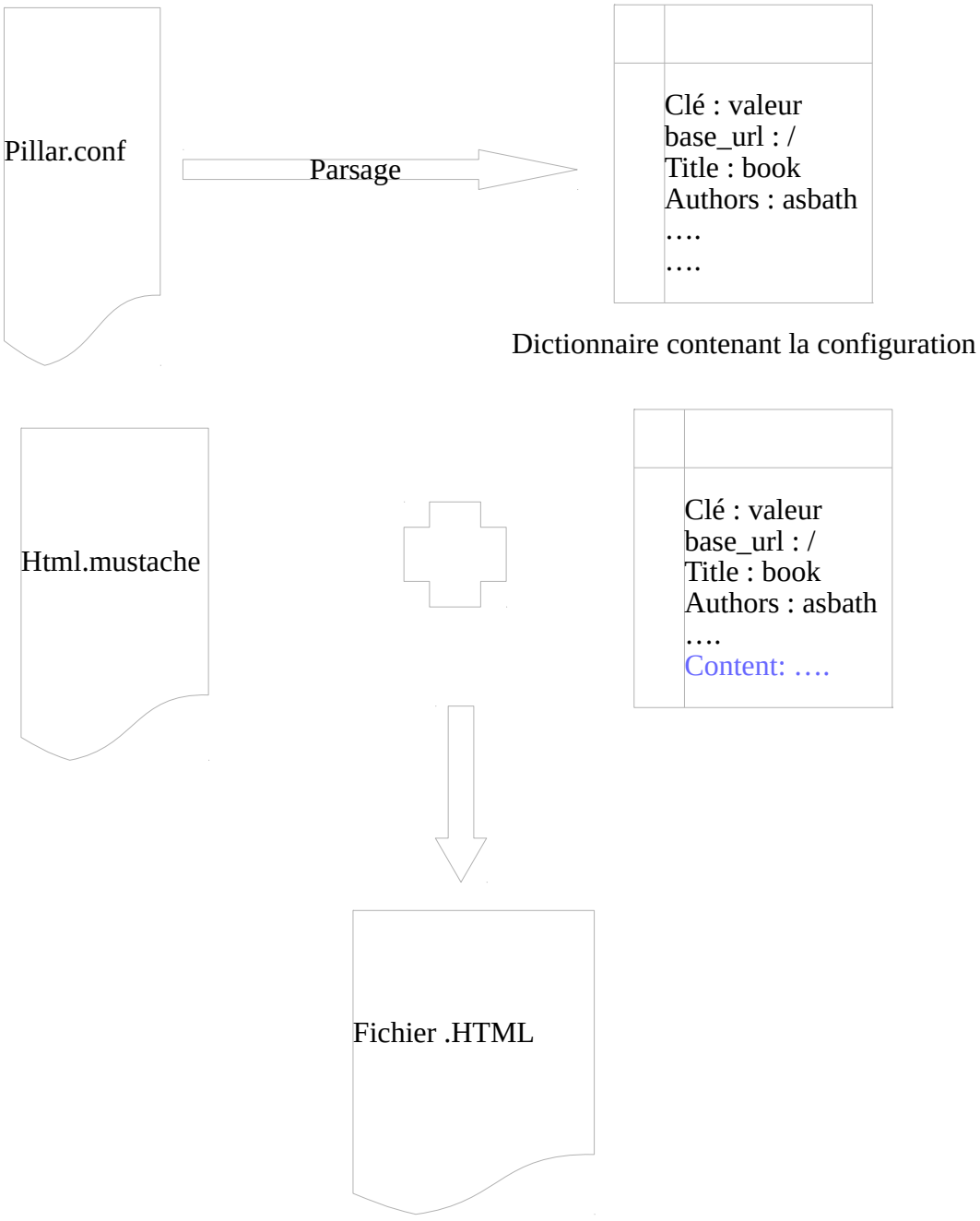

Figure 4: Processus de construction du fichier HTML à partir du dictionnaire de données

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 naviguation 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 récuperer 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.

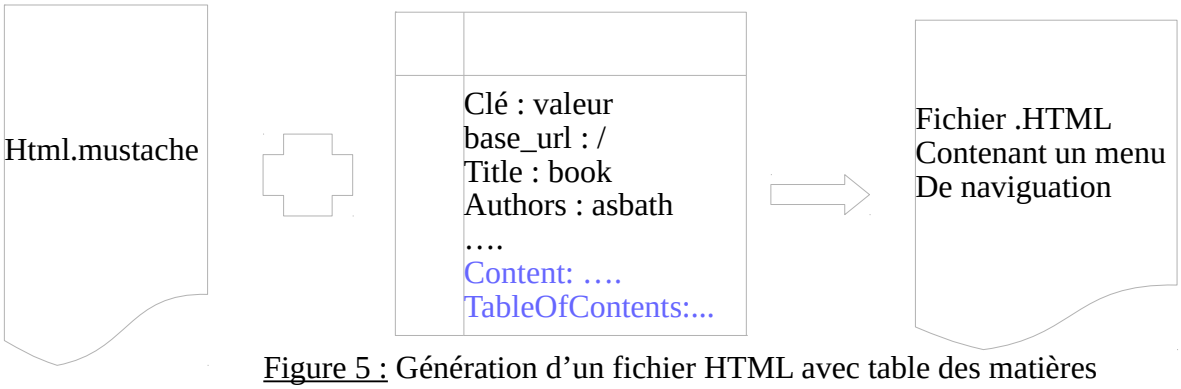

Figure 5 : Génération d'un fichier HTML avec table des matières

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 naviguation en ayant utilisé l'archetype proposant ce menu.

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 unitiaires, il a fallu reproduire des cas d'uitlisations 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 récupérer certains livres pertinents du compte github de l'association. Nous avons procéder 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 naviguation 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 indiqué le point d'entrée du document. Ceci représente le point à partir duquel on doit commencé à élaborer la table des matières. Il a fallu donc porter une attention particulière à la définition de la clé «tocFile» pour pouvoir effectivement généré 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 naviguation, 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 naviguation 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 passse sur l'environnement pour corriger les problèmes.

2.2.6. **Solutions apportées**

Les corrections ont porté sur les liens de naviguation. 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 naviguation, 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 concatenée avec le titre lui même. Pour réaliser cette solution, nous avons agis 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 amenér 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 as 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 naviguation 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 réfexions 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[8][9]**

La mise en place du module d'extraction de tables des matières a permis d'avoir des livres beaucoup plus faciles à consulter. La naviguation se faisait donc comme pour un site avec des liens permattant 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 consulter par tous. Sachant dans la plupart des cas tout le processus qui allait suivre la rédaction d'un document, nous avons décidé d'automiser les étapes de génération et de déploiement du livre afin de permettre une plus grande souplesse et facilité d'utilisation.
Cela a consité 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 dynamicité. 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. Nous avons pu répondre à certains de ces 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 naviguation. 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 naviguation. 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 naviguation 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 naviguation, 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'apréhender 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 as 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 plustard l'évolution des langages.

BIBLIOGRAPHIE

1:https://fr.wikipedia.org/wiki/
Institut_national_de_recherche_en_informatique_et_en_automatique
2: https://www.inria.fr/recherches/domaines-de-recherche
3: https://www.inria.fr/centre/lille/presentation/d-infos
4: https://www.inria.fr/institut/organisation/equipe-de-direction/isabelle-herlin
5: https://www.inria.fr/centre/lille/actualites/creation-du-centre-inria-lille-nord-europe
6: https://rmod.inria.fr/web
7:https://ci.inria.fr/pharo-contribution/job/EnterprisePharoBook/
lastSuccessfulBuild/artifact/book-result/PillarChap/Pillar.html
8: https://docs.travis-ci.com/user/deployment/pages/
9: https://help.github.com/articles/configuring-a-publishing-source-for-github-pages/

TABLE DES FIGURES