École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique

Cahier de spécification système & plan de développement		
Projet :	**Mise en place d'un serveur d'Intégration Continue et Qualité du Code**	
Emetteur :	Florent Clarret	Coordonnées : florent.clarret@etu.univ-tours.fr
Encadrants :	Nicolas Ragot	Coordonnées : nicolas.ragot@univ-tours.fr
	Pascal Meichel	Coordonnées : pascal.meichel@univ-tours.fr
	Vincent T'Kindt	Coordonnées : tkindt@univ-tours.fr
Date d'émission :	8 janvier 2015	

Validation

Nom	Date	Valide (O/N)	Commentaires
Nicolas Ragot	**17/12/2014**	**N**	-
Nicolas Ragot	**04/01/2014**	**O**	-
Pascal Meichel	**06/01/2014**	**O**	-

Historique des modifications

Version	Date	Description de la modification
01	**09/12/2014**	Version initiale du document
02	**16/12/2014**	Version modifiée du document
03	**18/12/2014**	Version modifiée du document
04	**04/01/2015**	Version modifiée du document
05	**06/01/2015**	Version finale du document

Table des matières

Cahier de spécification système

1.1 Introduction

Le cahier de spécification système et plan de développement est le document qui va servir de base et de référence à l'ensemble du projet de fin d'étude intitulé "Mise en place d'un serveur d'intégration continue et de qualité de code". Ce projet a été choisi par l'élève Florent Clarret et est encadré par Mr Nicolas Ragot, Mr Pascal Meichel ainsi que Mr Vincent T'Kindt.

Ce projet de fin d'études fait suite à un autre projet réalisé par Anne Castier lors de l'année universitaire 2013-2014. Ce document reprend donc certaines parties du cahier de spécification système qui avait été rédigé lors de ce projet. En outre, nous effectuerons aussi certaines références directes à ce document. Dans ce cas, veuillez vous en référer directement.

Le cahier de spécification ci-présent définit l'ensemble des besoins et des raisons qui ont poussé à la réalisation de ce projet de fin d'étude. Il permet de définir les fonctionnalités, le contexte ainsi que l'ensemble des contraintes auxquelles la solution qui sera fournie sera soumise.

Enfin, dans une seconde partie, ce document permettra de mettre en évidence le plan de développement du projet ainsi qu'une explication détaillée de l'ensemble des tâches à réaliser au cours de ce projet. Ce plan devra être suivi au maximum afin de permettre sa finalisation.

1.2 Contexte de la réalisation

1.2.1 Contexte

Comme il l'a été rappelé au sein du cahier de spécification d'Anne Castier, le but de ce projet consiste à mettre à disposition, au sein de Polytech' Tours, un serveur d'intégration continue et de surveillance de la qualité du code. Les personnes susceptibles de pouvoir se servir de ce serveur seront :

- Les étudiants de Polytech Tours.
- Les doctorants de Polytech Tours.
- Les enseignants chercheurs de Polytech Tours.

L'accès à ce serveur se devra d'être simple et permettra aux utilisateurs de pouvoir mettre en oeuvre les bonnes pratiques d'intégration continue et de qualité de code. L'ensemble de ces outils et de ces mécanismes seront décrits dans la suite de ce document. (Section 1.6, p. 15).

1.2.2 Objectifs

Les objectifs de ce projet sont multiples. Le premier de ces objectifs est la mise en place d'une machine virtuelle déployée au sein d'un serveur qui permettra d'accueillir l'ensemble des outils que nous souhaiterons mettre à disposition des utilisateurs finaux. Ces outils permettront de mettre en oeuvre une intégration continue et un suivi de la qualité du code. Il faudra donc prendre en main ces logiciels afin de comprendre leur fonctionnement et de pouvoir les utiliser correctement.

Contrairement à ce qui avait été initialement prévu lors du projet de fin d'études d'Anne Castier, la machine virtuelle que nous utilisons sera directement implantée dans le data center de la DTIC. Cela permettra à l'ensemble de l'université François Rabelais de pouvoir accèder à ce dernier depuis n'importe quel ordinateur du réseau ou de l'extérieur. De plus, cela permet de centraliser l'ensemble des serveurs de l'université afin de pouvoir faciliter la maintenance ou les contrôles de l'usage de ceux-ci.

Les choix des outils qui seront mis à disposition sur ce serveur ont d'ores et déjà été fixés. L'objectif intermédiaire sera donc de prendre en main ces différents logiciels. Nous allons cependant rajouter un ensemble d'outils et de plugins afin d'augmenter les possibilités offertes par le serveur.

Une fois le serveur d'intégration continue déployé au sein du réseau, il sera alors nécessaire de permettre aux utilisateurs de pouvoir y accéder et d'utiliser l'ensemble des fonctions qu'il offrira. Pour cela, nous allons configurer une machine virtuelle pour qu'elle possède tous les outils nécessaires pour communiquer avec le serveur. Une machine virtuelle de développement a déjà été réalisée par le service informatique, il sera donc plus intéressant de rajouter nos outils sur cette machine virtuelle plutôt de que d'en créer une nouvelle entièrement dédiée à cette tâche.

Lorsque le serveur d'intégration continue sera déployé, il sera important d'effectuer une étape de test afin de vérifier que celui-ci sera capable de remplir pleinement son rôle, et ce, quelque soit le nombre de personnes faisant appel à ce service.

Enfin, le résultat de ce projet sera ammené à être utilisé par de nombreux utilisateurs, expérimentés ou non, dans le domaine du développement, de l'intégration continue et de la surveillance de la qualité du code. Il sera donc nécessaire de fournir une documentation la plus détaillée possible pour qu'ils puissent prendre en main rapidement et simplement les outils mis à leur disposition. De plus, une documentation portant sur l'administration du serveur devra être mise en place afin de pouvoir maintenir et faire évoluer notre solution.

1.3 Description générale

1.3.1 Environnement du projet

Nous souhaitons déployer au sein de l'université François Rabelais un serveur d'intégration continue et de qualité de code. Un tel serveur n'existe pas encore sur le réseau de l'université. De ce fait, si les utilisateurs souhaitent mettre en place une démarche d'intégration continue et de qualité de code à l'heure actuelle, cela n'est pas possible au sein de l'université.

Cependant, un serveur de gestion de version est déjà mis en place sur le réseau. C'est un serveur qui permet de gérer des projets géré par le logiciel SVN. Ce serveur est mis à disposition de l'ensemble des étudiants ainsi que des enseignants-chercheurs de Polytech' Tours. Il permet de pouvoir gérer des projets de manière collective dans le cadre de TP ou de projets par exemple. Ce serveur SVN est associé à un serveur Redmine permettant d'effectuer de la gestion de projet collaborative.

L'objectif principal de notre projet serait donc de venir s'interfacer avec ce serveur déjà existant pour pouvoir s'appuyer dessus. Ceci est tout à fait envisageable et à déjà été réalisé dans le cadre du projet de fin d'études d'Anne Castier.

Enfin, comme nous l'avons vu, notre serveur sera implanté directement au sein du data center de l'université. Nous n'aurons alors aucune difficulté pour y accéder depuis n'importe où dans l'université, ou de l'extérieur. Cela simplifie grandement la gestion de ce serveur.

1.3.2 Caractéristiques des utilisateurs

Nous avons eu l'occasion de parler des utilisateurs du serveur que nous allons mettre en place. D'une manière globale, nous pouvons distinguer deux grandes catégories : les utilisateurs et les administrateurs.

Nous conservons ici les mêmes catégories d'utilisateurs qui avaient été établies lors de la rédaction du premier cahier de spécification réalisé l'année précédente.

1.3.2.1 Les administrateurs du serveur

Les personnes qui seront amenées à effectuer la maintenance et l'administration du serveur d'intégration continue seront les membres de l'équipe du service informatique de Polytech' Tours.

1.3.2.2 Les utilisateurs finaux

La seconde catégorie de personnes qui seront à même d'utiliser ce serveur est beaucoup plus large que la précédente. En effet, cette catégorie regroupe les étudiants, mais aussi les enseignants-chercheurs et les doctorants. Ils pourront utiliser ce serveur dans le cadre de travaux pratiques ou encore afin de mener à bien des projets tels que les Projets d'Ingénierie Collective ou encore des Projets de Fin d'Etude. Nous pouvons donc supposer que ces utilisateurs ont bien des connaissances en informatique. Cependant, il est fort probable que la majorité d'entre eux ne possèdent aucune connaissance dans le domaine de l'intégration continue ainsi que de la qualité du code. Il sera par conséquent primordial d'élaborer une documentation très précise et complète afin qu'ils soient tous en mesure de pouvoir utiliser correctement le serveur. Ainsi, leurs compétences dans ce domaine s'améliorera rapidement au cours du temps. De plus, ce seront des utilisateurs très régulier du service, les projets pouvant durer de nombreux mois. Ces utilisateurs n'auront que des droits restreints sur le serveur et ne pourront donc que créer, configurer et supprimer leurs projets.

Nous pouvons résumer tout cela à travers la matrice suivante :

	Administrateurs	**Utilisateurs**
Connaissances Informatiques	Oui	Oui
Experience de l'outil	Oui	Oui/Non
Degré d'utilisation	Occasionnel	Régulier
Droit d'accès	Administration	Utilisation

Les utilisateurs autorisés à accéder au service d'intégration continue seront explicitement précisés dans l'interface de gestion de notre serveur. Ainsi, en plus d'être dans le domaine de l'université, il faudra aussi que l'administrateur valide l'accès aux utilisateurs ou a des groupes d'utilisateurs. Cette gestion des accès des utilisateurs sera basée sur l'Active Directory et nous pourrons définir une liste d'utilisateurs en se basant sur les groupes de celui-ci.

1.3.3 Fonctionnalités et structure générale du système

Par rapport au précédent projet de fin d'étude, la structure générale ainsi que les fonctionnalités restent les mêmes. Cependant, il n'est plus possible d'installer le serveur de qualité de code SonarQube directement au sein d'un conteneur web. En effet, depuis la version 4.5, il est nécessaire d'installer SonarQube de manière complétement autonome. Cette contrainte nous force donc à ne plus utiliser de serveur de conteneur. En effet, l'un des principaux intérêts de l'utilisation d'un tel serveur était de pouvoir regrouper à la fois le serveur de qualité de code ainsi que le serveur d'intégration continue dans une seule et même entité. Désormais, nous avons donc décider d'installer nos deux services de manière complétement autonome, sans utiliser de serveur tiers. Malgré cela, notre système s'architecture toujours en deux parties indispensables :

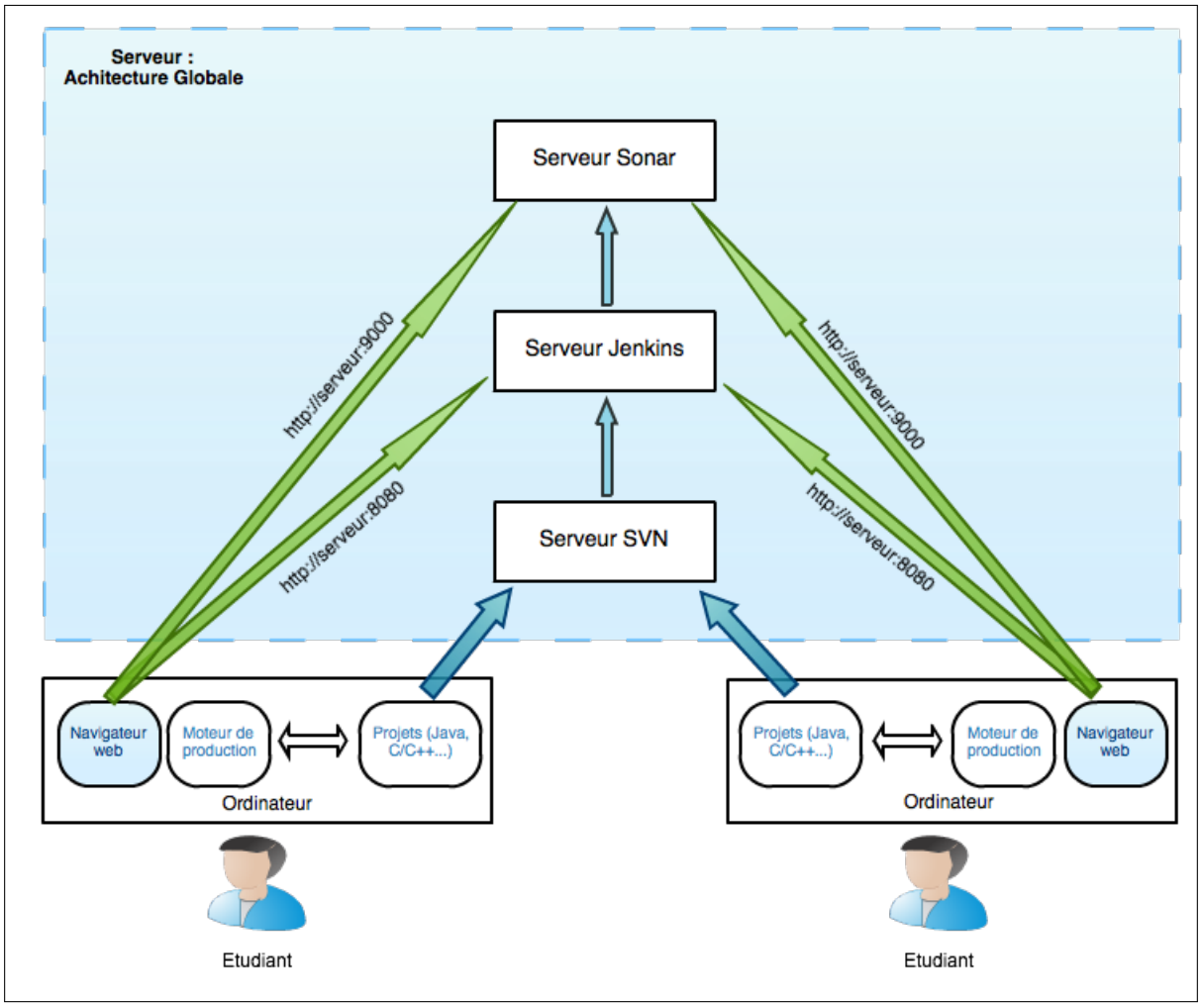

FIGURE 1.1 – Architecture générale du système d'intégration continue

Comme nous pouvons le voir, les deux parties principales sont inchangées, c'est bien l'organisation de la partie serveur qui est légérement modifiée. Le système est donc constitué de l'environnement du client et celui du serveur.

1.3.3.1 L'environnement du client

Afin de pouvoir exploiter au maximum le serveur d'intégration continue, l'environnement du client doit être doté au minimum des logiciels suivants :

- Un environnement de développement tel que Netbeans ou Eclipse par exemple, selon les plugins utilisés.
- Un navigateur web afin de pouvoir accéder aux interfaces d'utilisation de notre serveur.
- Un logiciel de gestion de version. Dans notre cas, il faudra utiliser SVN puisque nous nous basons sur le dépôt SVN interne de l'école.
- Un moteur de production qui va nous servir à pouvoir construire le projet à partir des sources afin de fournir un exécutable. Ce moteur sera fortement lié à l'environnement de développement.
- Un utilitaire permettant d'exécuter à distance le service de qualité de code. En effet, il est possible de lancer une analyse de code depuis Jenkins mais aussi directement depuis la machine cliente, c'est

le rôle de cet utilitaire.

Le client ne doit pas nécessairement disposer d'une connexion internet, l'unique obligation est de se situer dans le réseau de l'université. Concrètement, un utilisateur peut installer ses outils ou utiliser les machines virtuelles prévues à cet effet.

1.3.3.2 L'environnement du serveur

En ce qui concerne l'environnement du serveur, voici les éléments qu'il doit posséder :

- Un outil d'évaluation de la qualité de code. Dans notre cas il s'agira de l'outil nommé SonarQube.
- Un serveur d'intégration continue. Nous utilisons ici le logiciel Jenkins qui est un outil très complet.
- Le serveur de gestion de version. Le gestionnaire peut être installé soit directement sur la machine soit sur un serveur externe. Dans notre cas, nous allons nous appuyer sur le serveur déjà présent sur le réseau de l'université.
- Un ensemble d'outils nécessaire à l'installation de notre serveur d'intégration continue ainsi que notre outil d'évaluation de la qualité du code. Nous penserons par exemple à une base de données.

Il faudra veiller à ce que les interfaces d'utilisation de notre serveur soient bien accessibles depuis l'ensemble du réseau de l'université.

1.3.4 Contraintes d'exploitation et de maintenance

1.3.4.1 Contraintes de développement

Les contraintes de développement de ce projet sont sensiblement identiques à celle déjà présentes dans le projet de fin d'étude d'Anne Castier. Voici la liste complète de ces dernières :

1. Notre système devra fournir l'ensemble des outils afin de pouvoir travailler avec des projets dans des langages différents tels que le Java, le C/C++ ainsi que le C#. Ces trois langages sont les plus utilisés lors des projets réalisés au sein de l'école.
2. Le moteur de production choisi avait été Maven.
3. En ce qui concerne l'outil de gestion de versions, il a été décidé de nous reposer entièrement sur celui présent au sein du réseau de l'école. Il sera donc primordial de s'interfacer avec celui-ci.
4. Les outils utilisés au sein de ce serveur seront ceux qui ont été décidé lors du projet d'Anne Castier, à savoir Jenkins ainsi que SonarQube pour les éléments principaux.
5. Le projet devra suivre le plan de développement que nous avons précisé dans la seconde partie de ce rapport. (Section 2.2, p. 26)

En plus de celles-ci, d'autres contraintes ont été rajoutées :

1. En plus des trois langages prévus, il serait intéressant de pouvoir travailler avec d'autres langages comme le Php par exemple. L'ajout du support d'autres langages seraient un atout. Cette partie n'avait pas été traitée lors du projet précédent.
2. Nous pourrions étendre le choix des moteurs avec l'intégration de Ant afin de rendre notre serveur plus souple.
3. Une attention toute particulière sera apportée à vérifier le fonctionnement correct du serveur afin qu'il remplisse son rôle.
4. Une documentation riche devra être rédigée sur tous les aspects du projets.

1.3.4.2 Contraintes d'exploitation

La principale contrainte d'exploitation que nous avons réside dans le fait que la capacité de stockage de notre serveur doit être suffisante. En effet, nous avons l'obligation d'être en mesure de stocker l'ensemble des projets de tous les utilisateurs au sein de notre serveur tant pour la partie intégration continue que pour la partie contrôle de la qualité du code. Le serveur étant potentiellement utilisable par des centaines de personnes, nous pourrions être ammenés à devoir gérer plusieurs centaines de projets simultanément. Afin de pouvoir répondre à ce problème, nous avons décidé de fixer en premier lieu une taille de base pour assurer le stockage de nos données. Cet espace de stockage pourra être par la suite revu à la hausse ou à la baisse en cas de besoin. Des tests seront effectués au cours du projet afin d'obtenir des estimations plus précises. Initialement, nous estimons qu'une capacité de 1 TO de données sera suffisant. En plus de la capacité de la base de données, il faut fixer une taille limite qu'un projet Jenkins ne doit pas dépasser. Nous avons décidé de fixer une limite théorique de 10GO de données par projet car il n'est pas rare qu'un projet soit très volumineux avec les sources, les exécutables ainsi que les éventuelles ressources.

En dehors de la capacité de stockage de notre serveur, il faut aussi qu'il soit capable d'assurer de forte montées en charge d'un point de vue puissance de calcul. En effet, le processus d'intégration continue requiert de compiler les projets et il faut donc que le serveur soit capable d'assurer le traitement de tous les projets. Des tests de montées en charge seront nécessaires afin de s'assurer que le serveur sera bien capable de traiter toutes les demandes des utilisateurs. En effet, en fin d'année lors de la période des projets, il est possible que de nombreux groupes composés de 2 à 10 personnes utilisent notre service. Pour cela, nous avons défini trois critères à prendre en compte :

Temps de réponse acceptable : Notre interface se doit d'être la plus réactive possible afin de limiter au maximum le temps d'attente de l'utilisateur. Une base de temps de latence de l'ordre d'une seconde en moyenne serait acceptable.

Fréquence d'utilisation : Comme l'avait décrit Anne, la fréquence moyenne d'utilisation de notre service serait de 8h par semaine maximum pour un projet en moyenne.

Temps d'indisponibilité acceptable : Si le serveur est indisponible, il est nécessaire que cela soit durant la nuit afin d'impacter au minimum les utilisateurs. Une période de 2 à 8h pourrait être acceptée lorsque les étudiants en auront le moins besoin, c'est à dire hors période de projets.

Au-delà du bon dimensionnement des capacités de notre serveur, il faudra aussi construire un système de surveillance automatique. Ce système permettra de mettre en place des alertes lorsque des seuils fixés à l'avance seront dépassés. Un seuil concernant l'espace disponible restant devra être mis en place ainsi qu'un seuil permettant de limiter le nombre de tâches à réaliser par le serveur. L'objectif ici est de prévenir au maximum les problèmes que pourrait rencontrer le serveur et ainsi de simplifier l'administration de celui-ci par l'équipe du service informatique.

Nous l'avons vu, les utilisateurs seront séparés en deux grande catégories : les administrateurs et les simples utilisateurs. Les administrateurs auront l'ensemble des droits possibles sur le serveur ainsi que sur les projets des utilisateurs. Quant à eux, les utilisateurs auront le droit de créer et de configurer des projets sur notre serveur. Les droits associés aux utilisateurs est un point crucial dans le déploiement d'un tel serveur puisque tous les projets y sont stockés. Si ces droits étaient mal configurés, il serait alors possible à une personne d'avoir accès à des projets auxquels il n'aurait jamais dû avoir accès.

1.3.4.3 Maintenance et évolution du système

Une fois le serveur d'intégration continue déployé et complétement opérationnel, il sera nécessaire d'en effectuer la maintenance. Une solide documentation sur le fonctionnement du serveur ainsi que les différentes procédures à réaliser pour maintenir le système devra être rédigée. Cette documentation aura pour rôle de simplifier au maximum le travail à réaliser par le service informatique. Par conséquent, il sera important de prévoir l'ensemble des cas que pourra rencontrer l'équipe du service informatique afin de pouvoir y répondre.

De plus, la documentation permettra aussi de faire évoluer le système en cas de besoin. Cependant, l'analyse des besoins réalisée a permis de cerner au mieux les besoins que nous avions et les évolutions doivent rester rares.

Enfin, il sera important d'apporter des solutions quant aux problèmes de la sauvegarde des données présentes sur le serveur. En effet, si un problème survient, il faudra être en mesure de rétablir la configuration du serveur ainsi que l'ensemble des données qu'il contenait, et ce, le plus rapidement possible.

1.3.4.4 Modes de fonctionnement

Comme il l'avait été décidé dans le projet d'Anne Castier, nous ne considérerons pas dans notre cas de fonctionnement dégradé. Ainsi, seul le fonctionnement dit normal sera considéré, mettant à disposition l'ensemble des outils.

1.3.4.5 Sécurité

Comme nous l'avons vu dans une partie précédente (Section 1.3.2, p. 7), notre serveur d'intégration continue va reposer sur deux utilisateurs :

1. Les administrateurs.
2. Les utilisateurs.

Afin d'assurer la sécurité du serveur, nous allons travailler entièrement sur la définition de ces deux rôles. Afin de simplifier cette tâche, nous nous reposerons complétement sur l'Active Directory de l'université afin d'attribuer les droits d'administration aux seules personnes habilitées et les droits d'utilisation aux autres personnes.

1.3.4.6 Intégrité du serveur et des données

L'intégrité de notre serveur est un point fondamental. En cas de problème tel que la perte d'une partie des données ou encore la non-disponibilité du serveur, il est important de pouvoir relancer le système le plus rapidement possible. Il sera donc nécessaire d'effectuer des sauvegardes des données contenues sur le serveur mais aussi de permettre la remise en service des outils d'intégration continue. Des sauvegardes régulières devront ainsi être mise en place.

1.4 Architecture générale du système

L'architecte générale de notre système va suivre une architecture classique composée de plusieurs clients et d'un serveur mettant le service à disposition :

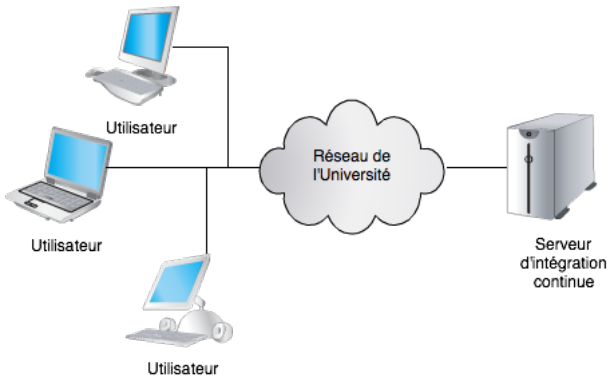

FIGURE 1.2 – Architecture de notre système

Au-delà du simple modèle client serveur, il est important de pouvoir montrer comment s'agencent les différents éléments de notre serveur d'intégration continue. Pour cela, voici un schéma montrant ces éléments :

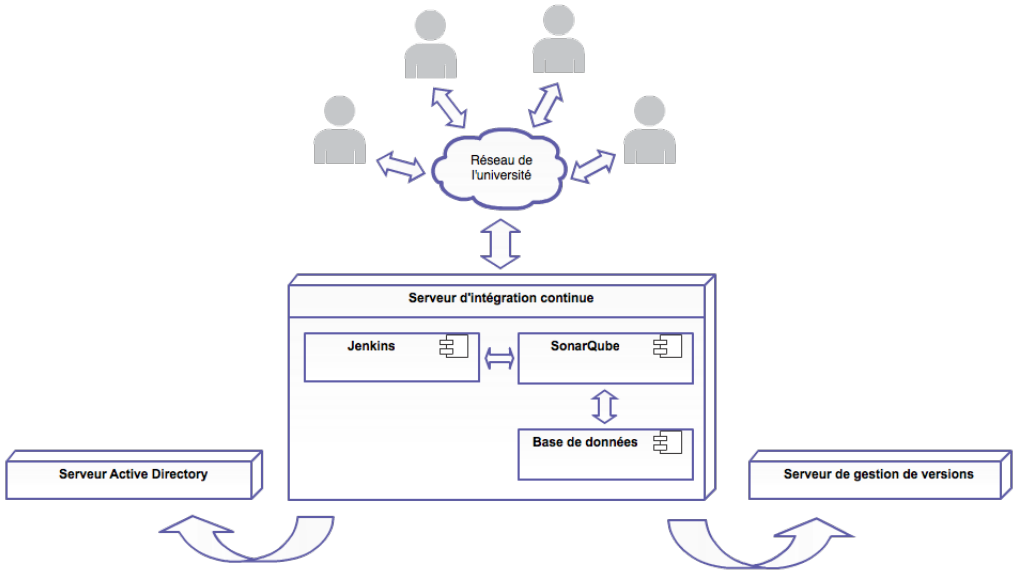

FIGURE 1.3 – Architecture complète du système

Les utilisateurs utilisent notre serveur par le biais de leur navigateur web. Ils peuvent configurer des jobs associés à leur projet. Les sources de ce projet sont obligatoirement stockées sur le serveur SVN de l'école qui est associé au Redmine. Une fois le projet Jenkins créé, le code source de l'application est demandé au serveur SVN puis compile et exécute le projet. Jenkins peut alors communiquer avec SonarQube pour lui demander d'effectuer une analyse de code sur ce même projet et l'utilisateur pourra y accéder via l'interface web. Afin de pouvoir authentifier les utilisateurs, Jenkins ainsi que SonarQube se base entièrement sur l'annuaire contenu dans l'Active Directory de l'Université.

1.5 Description des interfaces externes du logiciel

1.5.1 Interfaces matériel/logiciel

Dans le cadre de ce projet, il a été décidé pour des raisons de simplicité de mettre à notre disposition une machine virtuelle sur laquelle nous pourrons implanter notre serveur d'intégration continue. Cette machine nous servira aussi à effectuer différents tests, notamment de fonctionnement et de montée en charge du serveur. La raison de ce choix est qu'une machine virtuelle est facilement configurable en cas de modification de notre serveur et peut être facilement déplacée sur un autre serveur physique si besoin. Le dimensionnement des capacités est aussi simplifié via l'utilisation d'outil de virtualisation comme VMware vSphere.

1.5.2 Interfaces homme/machine

Comme nous venons de le voir, l'utilisateur de notre serveur d'intégration continue va venir intéragir avec notre machine virtuelle. Cette machine sera entièrement configurée, l'utilisateur n'aura alors qu'à se soucier de son propre projet. Tous les outils présentés dans ce rapport y seront présents. L'ensemble des intéractions avec ceux-ci se feront uniquement via un portail web afin de simplifier leur utilisation. L'ensemble des interfaces homme/machine utilisées seront celles des outils, elles sont donc d'ores et déjà existantes. De ce fait, nous n'aurons pas besoin de les développer nous-même. Nous pouvons distinguer deux types d'outils :

1. Les outils installés directement sur le poste de l'utilisateur tel que l'IDE et le gestionnaire de version SVN.
2. Les outils installés entièrement sur le serveur qui sont accessibles depuis un navigateur web.

Pour éviter à l'utilisateur de devoir à installer un environnement complet de développement, il aura la possibilité d'utiliser la machine virtuelle cliente que nous aurons créée.

1.5.3 Interfaces logiciel/logiciel

Afin de fonctionner correctement, les logiciels que nous avons installés sur notre serveur doivent être correctement interfacés pour qu'ils puissent fonctionner ensemble. Les interactions entre ces outils sont décrites dans le schéma suivant (schéma issu du cahier de spécification d'Anne Castier) :

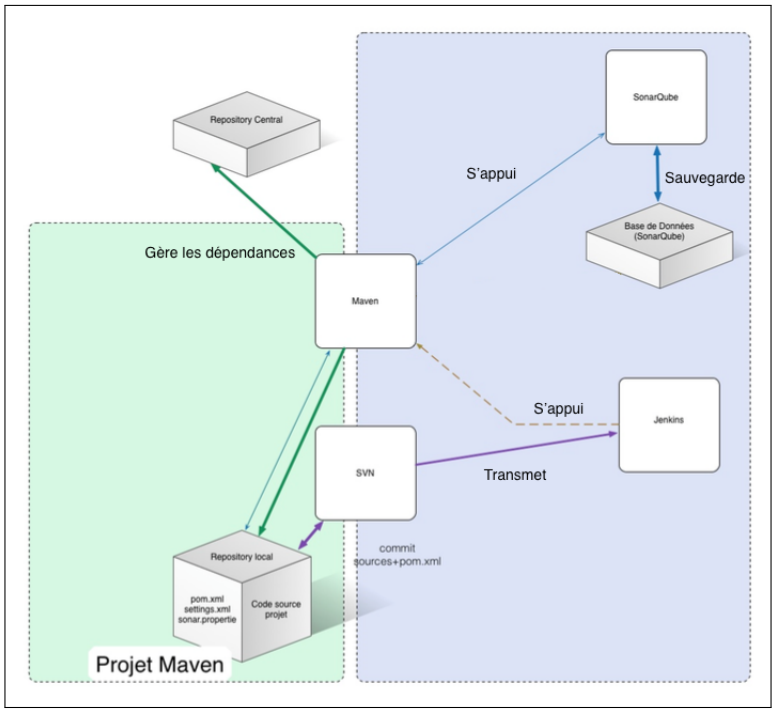

FIGURE 1.4 – Intéractions entre les logiciels installés

Afin d'interfacer l'ensemble des composants de notre projet, nous utilisons le moteur de production Maven. Il permet de faire le lien entre notre projet, un repository central qui permet de gérer les dépendances et le serveur d'intégration continue Jenkins. C'est un framework de scripting très complet et performant qui permet à la fois de générer les exécutables et d'interfacer des logiciels tiers. Concrétement, pour créer une projet Maven il suffit de créer à la racine d'un projet classique un fichier nommé pom.xml et d'avoir maven installé sur la machine. Ce fichier possède un format spécifique.

Ensuite nous avons une liaison directe entre le serveur de gestion de version SVN ainsi que Jenkins. Ce lien s'effectue lors de la création et la configuration d'un job. En effet, l'url vers le dépôt SVN du projet est alors requis. Il est aussi nécessaire de dire à Jenkins quel moteur de production nous utilisons dans notre projet (ici Maven) afin qu'il soit capable de construire le projet et d'exécuter les différents tests unitaires de manière complétement autonome.

Enfin, il faudra interfacer SonarQube avec le serveur Jenkins afin de pouvoir effectuer les mesures sur la qualité du code. Il existe plusieurs manière d'exécuter SonarQube au sein de notre projet via Jenkins ou directement grâce à Maven. Ici, nous allons uniquement travailler sur le lien entre Jenkins et Sonar. En effet, il est possible de demander à Jenkins qu'après chaque commit par exemple il effectue un build complet du projet, lance les tests et demande à Sonar d'analyser le code. Cela simplifie donc l'utilisation de tous ces éléments par l'utilisateur puisque tout est réalisé automatiquement.

1.6 Description des fonctionnalités

1.6.1 Intégration continue via Jenkins

1.6.1.1 Rappels sur Jenkins

Jenkins est l'un des serveurs d'intégration continue les plus populaire et l'un des plus utilisé dans le monde de l'entreprise. Jenkins est en réalité un fork d'un autre logiciel nommé Hudson. Jenkins a été

créé après plusieurs problèmes rencontrés entre la communauté de développement et Oracle qui était alors détentrice du logiciel. Ainsi, depuis cette scission, Hudson et Jenkins coéxistent dans le domaine de l'intégration continue mais leur développement suit des lignes différentes.

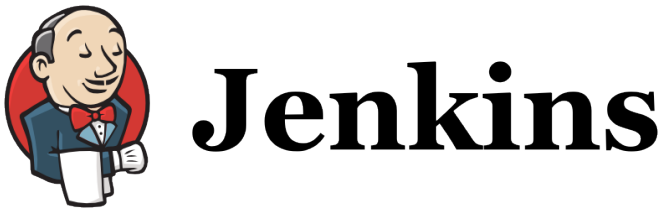

FIGURE 1.5 – Logo de Jenkins

Les deux rôles principaux de Jenkins sont :

1. Permettre la compilation et le lancement des tests de manière automatique lors de commit par exemple ou encore de manière périodique.
2. Permet d'actionner des tâches externes tel que dans notre cas SonarQube. Cependant, ce n'est pas la seule tâche externe qu'il est en mesure d'exécuter. De nombreux plugins permettent de gérer un grand nombre de logiciels.

1.6.1.2 Choix des plugins à installer

Afin d'augmenter les capacités de notre serveur d'intégration continue, un très grand nombre de plugins est disponible. Parmi ceux-ci, voici la liste de ceux que nous allons installer :

Active Directory Plugin : Permet l'accès à l'active directory pour la gestion des accès utilisateurs.

Apache Ant Plugin : Permet le support de Ant au sein de Jenkins.

Create Job Extended : Permet d'ajouter certaines fonctionnalités lors de la création de jobs.

Disk-Usage Plugin : Permet de surveiller l'espace disque occupé par Jenkins.

CMake : Permet à Jenkins de pouvoir construire des projets utilisant CMake.

Extended Read Permission Plugin : Permet d'ajouter diverses options de gestion des droits sur les jobs.

MSTest Plugin : Permet d'intégrer les rapports MSTest.

LDAP Plugin : Permet l'authentification via LDAP.

Maven Plugin : Permet le support de Maven dans Jenkins.

Monitoring : Permet d'avoir un grand nombre de statistiques et de graphiques sur le serveur Jenkins.

MSBuild Plugin : Permet de construire des projets Visual Studio.

NAnt Plugin : Permet de construire des projets .NET.

Sonar : Permet le lien avec le serveur Sonar.

xUnit Plugin : Permet d'enregistrer des rapports de tests xUnit.

Dans une seconde partie, afin de augmenter les possibilités offertes par Jenkins, il sera nécessaire d'installer d'autres plugins. Ces plugins permettront de gérer par exemple l'utilisation de langage tel que le PHP ou encore le moteur de production Ant. Ces plugins seront à déterminer dans la suite du projet à la suite d'une étude approfondie.

1.6.2 Qualité de code à l'aide de SonarQube

1.6.2.1 Description de SonarQube

FIGURE 1.6 – Logo de SonarQube

Le rôle de SonarQube est de permettre d'effectuer une analyse sur l'ensemble du code source que nous avons produit. Il permet d'en extraire un très grand nombre de statistiques, parmi celles-ci nous pouvons citer par exemple :

- Le nombre de ligne de code.
- Le pourcentage de couverture des tests.
- Le nombre de ligne de code dupliquée.

Afin de pouvoir mesurer la qualité du code, il est important de définir les régles qui nous permettront de déterminer si un code est correct ou non. Au sein de Sonar, de nombreuses régles sont déjà définies mais il est aussi possible d'en ajouter. Nous nous contenterons cependant dans un premier temps des régles qui sont directement implémentées avec l'outil nommé Findbugs.

1.6.2.2 Choix des plugins à installer

Tout comme pour Jenkins, nous allons installer un ensemble de plugins pour étendre les possibilités de Sonar :

- Redmine plugin
- Ldap plugin
- Pdf report plugin
- C/C++/Objective C
- Scm stats plugin
- Build stability plugin
- Abacus plugin
- C# plugin
- C/C++ community
- Useless code tracker plugin

Ici aussi, nous serons amenés à enrichir le nombre de plugins installés.

1.6.3 Les langages supportés

Dans un premier temps, il est impératif que notre serveur d'intégration continue soit capable de travailler avec différents langages. Cette liste de langage a été élaborée lors des spécifications du projet d'Anne Castier. Elle comprend les langages les plus utilisés au sein du laboratoire de recherche ainsi que par les étudiants :

— Le langage C avec Visual Studio

— Le langage C++ avec Visual Studio
— Le langage C# avec Visual Studio
— Le langage Java avec Maven

Une fois cette liste de langages gérée par notre serveur, il serait intéressant de pouvoir l'élargir à d'autres si le temps le permet. Nous pensons par exemple à pouvoir intégrer le langage PHP.

1.6.4 Moteurs de production

FIGURE 1.7 – Logo de Maven

Afin que notre serveur Jenkins soit en mesure de pouvoir travailler avec notre projet, il est impératif d'utiliser des moteurs de production. Pour cela, nous utilisons de manière exclusive Maven.

Pour rendre notre serveur plus souple, nous pourrions essayer d'intégrer d'autres moteurs de production pour gérer nos projets. Par exemple, il serait possible de permettre l'utilisation du moteur nommé Ant.

FIGURE 1.8 – Logo de Ant

1.6.5 Gestion des projets

Afin de pouvoir administrer simplement notre serveur d'intégration continue, il faudra mettre en place des outils permettant de pouvoir effectuer des tâches simples telles que la suppression de projets. Tout comme pour le serveur de gestion de versions SVN, il est important de pouvoir donner la possibilité aux administrateurs de supprimer un grand nombre de projets rapidement. Il est aussi important de leur permettre de pouvoir modifier les projets en cas de mauvaise configuration de ceux-ci par les utilisateurs.

Un système d'alerte sera mis en place afin d'informer les administrateurs si un problème relatif à l'espace de stockage ou de la capacité de traitement du serveur risque d'arriver. Ils pourront ainsi prendre des mesures préventives et pouvoir ainsi éviter le problème.

1.7 Les livrables du projet

1.7.1 Machine virtuelle du serveur

Comme nous l'avons vu, l'objectif principal de ce projet de fin d'études est la mise à disposition d'un serveur permettant la mise en place des bonnes pratiques de l'intégration continue. Ce serveur aura la forme d'une machine virtuelle entièrement configurée et fonctionnelle. Cette machine virtuelle contiendra l'ensemble des outils dont nous avons fait la description au sein de la partie "Descriptions des fonctionnalités" (Section 1.6, p. 15). Comme nous l'avons vu, suite à une réunion avec Mr Meichel, il a été décidé d'intégrer cette machine virtuelle directement au sein du data center de l'université, à la DTIC.

1.7.2 Machine virtuelle des clients

Le second livrable que nous devrons réaliser consiste à configurer une machine virtuelle pour qu'elle puisse s'interfacer directement avec le serveur d'intégration continue. Tous les outils devront y être installés et configurés correctement. Suite à une réunion avec le service informatique, nous avons décidé de rajouter cet ensemble d'outils sur la machine virtuelle déjà existante et destinée au développement d'applications. En effet, il est beaucoup plus simple d'intégrer ces éléments dans cette machine virtuelle plutôt que d'en dédier une pour l'intégration continue. Cela permettra aussi de simplifier son utilisation pour les utilisateurs.

1.7.3 Documentation

Associé à ces deux machines virtuelles, il sera nécessaire de fournir une solide documentation tant pour les utilisateurs que pour les administrateurs. Cette documentation permettra aux utilisateurs de prendre en main facilement les différents outils. La documentation pour les administrateurs permettra quant à elle de pouvoir maintenir et faire évoluer facilement notre serveur. L'ensemble de ces documents est précisé dans le plan de développement et seront réalisés tout au long du projet.

Plan de développement

2.1 Découpage du projet en tâches

Afin de pouvoir planifier de manière plus simple le projet, nous avons découpé celui-ci en différentes phases qui devront se succéder les unes aux autres. Cette partie va permettre de détailler chacune des ses étapes.

2.1.1 Découverte et prise en main des outils d'intégration continue et de l'existant

2.1.1.1 Description de la tâche

Ce projet faisant suite au projet réalisé par Anne Castier dans le cadre de son projet de fin d'études, la première tâche à réaliser consistera donc à reprendre entièrement le travail qu'elle a pu accomplir. La documentation fournie par Anne étant très conséquente, il est primordial de bien la comprendre et la prendre en main. De plus, n'étant pas familier avec les systèmes d'intégration continue, il sera d'autant plus important de bien comprendre ce fonctionnement.

2.1.1.2 Déroulement

Nous pouvons découper cette première tâche en différentes sous-tâches :

1. Lecture de la documentation réalisée.
2. Déploiement des machines virtuelles mises en places lors du projet précédent.
3. Tests d'utilisation des outils d'intégration continue.

2.1.1.3 Estimation de la charge

Afin de mener à bien cette tâche, nous avons estimé la charge à 3 jours/homme.

2.1.1.4 Contraintes temporelles

Cette tâche a été réalisée au cours des première semaine de ce projet de fin d'études, soit de mi-septembre à mi-octobre.

2.1.2 Rédaction du cahier de spécification système et plan de développement

2.1.2.1 Description de la tâche

Au cours de cette tâche, il sera nécessaire de réaliser le cahier de spécification système ainsi que le plan de développement du projet. En plus de ce cahier, il faudra aussi réaliser une affiche de présentation du projet.

2.1.2.2 Livrables fournis

Les deux livrables qui seront fournis à la fin de cette tâche seront :

1. Le cahier des spécifications système muni du plan de développement.
2. L'affiche explicative du projet.

2.1.2.3 Estimation de la charge

Afin de mener à bien cette tâche, nous avons estimé la charge à 5 jours/homme.

2.1.2.4 Contraintes temporelles

Ce cahier de spécifications se devra d'être finalisé avant le 9 janvier 2015 avec l'affiche. Cependant, des pré-versions de ce document devront être soumis aux encadrants du projet au mois de décembre.

2.1.3 Installation et configuration de l'environnement du serveur

2.1.3.1 Description de la tâche

Lors du précédent projet, une machine virtuelle avait été réalisée. Cependant, suite à une réunion faite avec l'équipe du service informatique, nous avons décidé de construire une nouvelle machine virtuelle au sein de la DTIC. En effet, la VM que nous avions jusqu'à présent n'était pas conçue pour habriter un serveur pouvant fonctionner en production dans des conditions acceptables. De plus, des nouvelles versions des logiciels que nous utilisons sont apparues depuis la création de la machine virtuelle, rendant plusieurs choix qui avait été fait impossible à mettre en place désormais.

2.1.3.2 Cycle de vie

Cette tâche s'est découpée en une liste de plusieurs sous-tâches :

1. Choix du type de machine virtuelle.
2. Installation des composants du serveur.
3. Configuration des différents outils.
4. Rédaction du manuel d'installation de la machine virtuelle serveur.
5. Rédaction du manuel d'administration du serveur.

Cette tache sera réalisée de manière itérative. A chaque itération nous effectuerons un ensemble de tests afin de vérifier le bon fonctionnement de notre machine virtuelle. Si les tests ne sont pas concluants, alors nous réaliserons une nouvelle itération afin de corriger les erreurs et les problèmes que nous aurons détécté lors de l'itération précédente.

Lors de cette phase, il faudra aussi rédiger un manuel d'installation. Ce manuel d'installation globale de la machine virtuelle permettra à quiconque d'être en mesure d'installer un serveur d'intégration continue à partir d'une machine complètement vierge. Il est important que ce document soit le plus complet possible afin que même en ayant peu de connaissances dans le domaine, la personne soit capable de réaliser l'installation simplement en étant guidé par le document. La machine virtuelle sera d'ores et déjà opérationnelle et n'aura donc pas besoin d'être installé. Cependant, il est important de garder un tel document à jour en cas de migration du serveur ou d'une ré-installation suite à un incident tel que la perte d'un disque dur par exemple.

En plus du manuel d'installation, il faudra aussi rédiger un document permettant à l'équipe d'administrateurs de connaître les procédures à suivre afin de réaliser différentes actions d'administration. On peut par exemple penser à différentes situations :

- Gestion des nouveaux utilisateurs ainsi que leurs droits sur la plateforme.
- Suppression des anciens projets qui ne sont plus utilisés afin de faire de la place sur le serveur.
- Effectuer les opérations de mise à jour du système.

Certains de ces éléments seront partiellement repris du manuel d'installation. Cette liste n'est cependant pas exhaustive et il sera nécessaire de prévoir toutes les actions que les administrateurs seront amenés à devoir réaliser.

2.1.3.3 Livrables fournis

A la fin de cette tâche, nous aurons été en mesure de fournir une machine virtuelle permettant aux utilisateurs de pouvoir mettre en place les bonnes pratiques d'intégration continue ainsi que de surveillance de la qualité du code. Cette machine virtuelle sera munie d'un manuel d'installation ainsi que d'un manuel d'administration.

2.1.3.4 Estimation de la charge

Cette tâche a été estimée à 12 jours/homme.

2.1.3.5 Contraintes temporelles

Le serveur d'intégration continue doit pouvoir être fonctionnel avant la fin du projet, c'est à dire en mai de l'année 2015.

2.1.4 Installation et configuration de l'environnement du client

2.1.4.1 Description de la tâche

En plus de la machine virtuelle serveur, il faudra mettre en place dans une machine les différents outils afin de pouvoir s'interfacer simplement avec le serveur d'intégration continue. Nous avons choisi pour cela de rajouter ces outils sur la machine servant au développement afin de faciliter son utilisation. Un manuel d'installation de cette machine devra être fourni.

Le service d'intégration continue sera utilisé dans le cadre de projets afin d'utiliser les bonnes méthodes de gestion de projet. Cependant, les utilisateurs peuvent n'avoir jamais utilisé de tels outils. Pour cela, il faudra rédiger un document qui permettra de prendre en main rapidement l'utilisation du serveur en y expliquant les points essentiels au fonctionnement de leur projet.

Au delà de la rédaction d'un manuel de démarrage rapide, un manuel utilisateur complet devra être réalisé. L'objectif de ce document serait, en plus de la prise en main du serveur ainsi que de ses fonctionnalités, de permettre à l'utilisateur de pouvoir utiliser les fonctionnalités les plus poussées du serveur. Toutes les options disponibles devront y être expliquées en détail pour qu'il puisse profiter de toute les possibilités qui sont offertes par le serveur et ainsi l'utiliser à 100%.

2.1.4.2 Cycle de vie

Cette tâche s'est découpée en une liste de plusieurs sous-tâches :

1. Identification des composants à ajouter à la machine virtuelle.
2. Mise en places des outils.
3. Tests de bon fonctionnement.
4. Rédaction du manuel d'installation.
5. Rédaction d'un manuel de démarrage rapide.
6. Rédaction d'un manuel utilisateur complet.

Tout comme pour l'installation et la configuration de la machine virtuelle serveur, nous allons réaliser cette tâche de manière itérative.

2.1.4.3 Livrables fournis

Le livrable fourni à la suite de cette tâche sera la machine virtuelle de l'université à laquelle les outils utiles à l'interfaçage avec le serveur d'intégration continue seront installés et configurés.

2.1.4.4 Estimation de la charge

Cette tâche a été estimée à 11 jours/homme.

2.1.4.5 Contraintes temporelles

Tout comme pour la machine virtuelle serveur, cette machine doit être livrée pour le mois de mai.

2.1.5 Tests de fonctionnement en conditions rééelles

2.1.5.1 Description de la tâche

Avant la fin du projet, il sera obligatoire d'effectuer une phase de test afin de vérifier le comportement de notre serveur en conditions rééelles. Cette phase nous permettra de dimensionner correctement les caractéristiques du serveur afin qu'il puisse répondre aux fortes charges qu'il sera amené à subir. Cette phase permettra aussi de vérifier si la configuration du serveur est correcte ou si des modifications doivent être réalisées.

2.1.5.2 Estimation de la charge

Cette tâche a été estimée à 15 jours/homme.

2.1.5.3 Contraintes temporelles

Cette phase sera réalisée avant le mois de mai 2015 et plus précisément durant la période des Projet d'Ingénierie Collective et des Projets de Fin d'Etudes afin de pouvoir demander aux étudiants de travailler avec ce serveur.

2.1.6 Mise en place d'outils de surveillance du système

2.1.6.1 Description de la tâche

Lorsque notre serveur d'intégration continue sera fonctionnel, il sera nécessaire de mettre en place un ensemble d'outils qui vont nous permettre de surveiller notre serveur afin d'éviter des surcharges. Cette surveillance portera sur différents aspects :

- L'espace de stockage disponible sur le serveur.
- Le nombre de job s'exécutant dessus.
- Le nombre de projets créés.

Cette liste n'est pas exhaustive et sera enrichie lors de la mise en place ces outils.

2.1.6.2 Livrables fournis

Le livrable fourni par cette tâche sera entièrement intégré au serveur d'intégration. Il consistera, en fonction des solutions choisies, en un ensemble de scripts ou de plugins associés à notre serveur d'intégration continue.

2.1.6.3 Estimation de la charge

Afin de mener à bien cette tâche, nous avons estimé la charge à 6 jours/homme.

2.1.6.4 Contraintes temporelles

Cette tâche se devra d'être réalisée avant la soutenance de fin de projet et après la phase de test de fonctionnement de notre serveur.

2.1.7 Ajout de fonctionnalités au serveur

2.1.7.1 Description de la tâche

Une fois la machine virtuelle serveur déployée et configurée, une nouvelle tâche serait d'étendre les possibilités offertes par le serveur et ainsi lui ajouter de nouvelles fonctionnalités.

La première fonctionnalité que nous pourrions rajouter serait d'être en mesure de travailler avec d'autres langages que le C/C++, le C# ou le Java. Intégrer des modules permettant de travailler par exemple en PHP ou en Python augmenterait le potentiel de notre serveur.

Pour le moment, notre serveur est limité à l'usage de Maven ou à Visual Studio. Il serait intéressant de permettre aux utilisateurs de travailler avec d'autres moteurs de production tel que Ant par exemple.

2.1.7.2 Estimation de la charge

Cette tâche a été estimée à 6 jours/homme.

2.1.7.3 Contraintes temporelles

La priorité de cette tâche est faible. En effet, l'ajout de fonctionnalités n'est pas une tâche impérative de notre projet. Nous réaliserons cette tâche si le temps le permet à la fin du projet et devra alors être terminée pour la soutenance de fin de projet et mai 2015.

2.1.8 Mise en place d'une plateforme pédagogique

2.1.8.1 Description de la tâche

Lors de ce projet, nous fournirons un grand nombre de documentation pour que les utilisateurs finaux ainsi que les administrateurs soient en mesure d'utiliser correctement le serveur d'intégration continue. Cependant, il serait utile pour les utilisateurs de mettre en place une plateforme pédagogique. Cette plateforme donnerait l'accès aux utilisateurs à un dépôt de source SVN contenant différents projets qu'ils pourraient utiliser afin d'effectuer des tests et prendre en main le serveur. Avec ces projets, un document sous la forme d'un ennoncé de travaux pratiques serait réalisé. Ce document permettrait aux utilisateurs de pouvoir prendre en main le serveur sur des projets types dans le cadre d'un cours de Génie Logiciel par exemple.

2.1.8.2 Livrables fournis

A la fin de cette tâche, nous fournirons :

1. Un dépôt SVN contenant des projets dans différents langages.
2. Un document sous la forme d'un TP.

2.1.8.3 Estimation de la charge

Cette tâche a été estimée à 5 jours/homme.

2.1.8.4 Contraintes temporelles

Tout comme pour l'ajout de fonctionnalités, la priorité de cette tâche est faible. Elle sera réalisée uniquement si le temps le permet avant la fin du projet.

2.1.9 Documentation annexes

En plus de l'ensemble des documents que nous venons de présenter, il sera fourni un dernier document qui sera le rapport final de ce projet. Ce document viendra alors compléter la liste de documents qui aura été réalisée tout au long du projet.

2.2 Planning

Afin de prévoir l'ordonnancement des tâches, vous trouverez ci-dessous un tableau récapitulatif de l'ensemble des tâches à réaliser. Ce tableau présente :

1. Le numéro de la tâche
2. Le nom de la tâche.
3. Le nom de la sous-tâche le cas échéant.
4. L'estimation de la charge associée à cette tâche.
5. L'ordre dans lequel les tâches seront réalisées.
6. La priorité de la tâche.
7. La date de début de tâche estimée.

#	Titre	Travail Planifié Donné	Drapeau	# Prédécesseurs	Date Début Attendue
0	**Projet de Fin d'Etude – Serveur d'intégration continue**				18/09/2014
1	Découverte et prise en main des outils	3 jours			18/09/2014
2	Rédaction du cahier de spécification système et poster	7 jours		6	13/11/2014
3	**Mise en place de la machine virtuelle serveur**				09/10/2014
4	Installation des composants du serveur	1 jour		1	09/10/2014
5	Configuration des outils du serveur	1 jour		4	16/10/2014
6	Rédaction du manuel d'installation serveur	2 jours		5	23/10/2014
7	Rédaction du manuel d'administration	3 jours		2	13/01/2015
8	**Mise en place de la machine virtuelle client**				20/01/2015
9	Installation des composants du client	1 jour		7	20/01/2015
10	Rédaction du manuel d'installation client	3 jours		9	21/01/2015
11	Rédaction du manuel utilisateur complet	4 jours		10	28/01/2015
12	Rédaction du manuel de démarrage rapide	3 jours		11	05/02/2015
13	Tests de fonctionnement en condition réelles	15 jours		12	12/02/2015
14	Mise en place des outils de surveillance	6 jours		13	26/03/2015
15	Rédaction du rapport final du projet	6 jours		14	09/04/2015
16	**Ajout de fonctionnalités**				01/04/2015
17	Ajout du support de nouveaux langages	3 jours		13	01/04/2015
18	Ajout du support de nouveaux moteurs de production	3 jours		17	08/04/2015
19	**Mise en place de la plateforme pédagogique**				15/04/2015
20	Création des projets types	2 jours		18	15/04/2015
21	Rédaction du sujet de TP	3 jours		20	21/04/2015

FIGURE 2.9 – Liste des tâches du projet

On peut résumer le tableau ci-dessus sous la forme d'un diagramme de Gantt suivant :

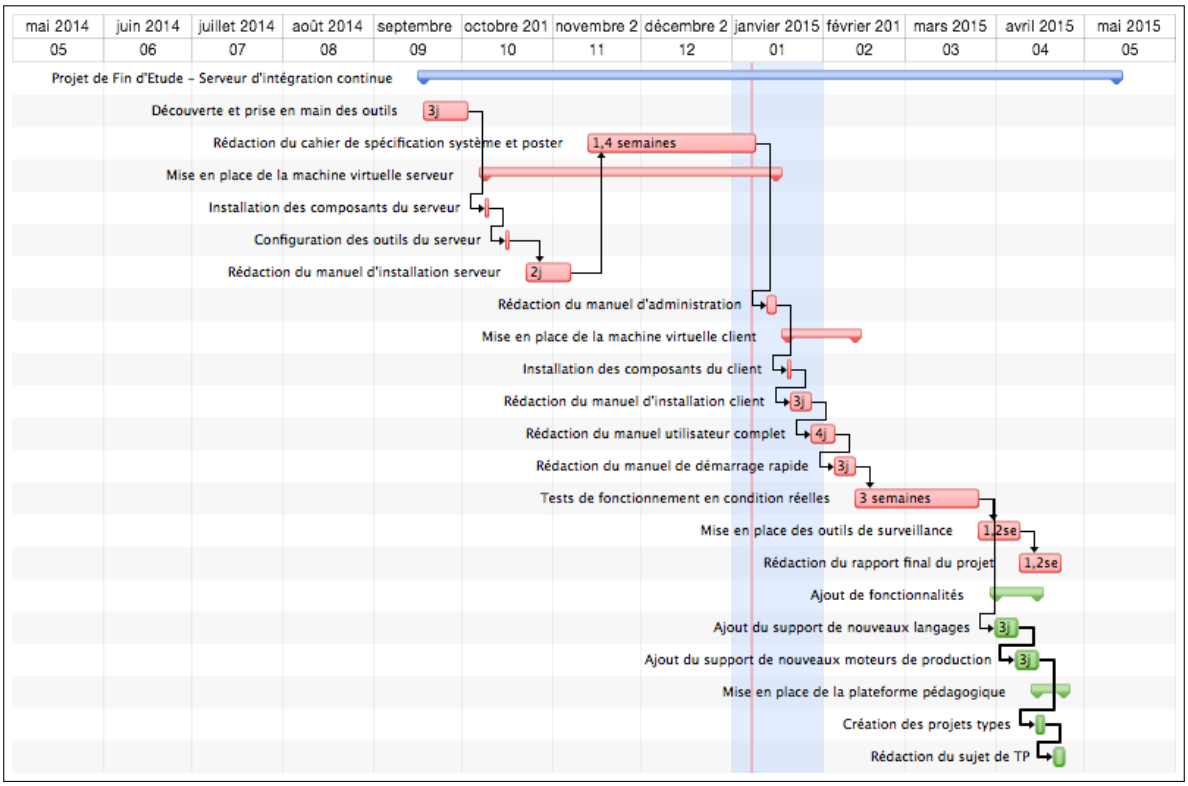

FIGURE 2.10 – Diagramme de Gantt

Glossaire

DTIC : Direction des Technologies de l'Information et de la Communication.

EDI : Environnement de développement intégré. IDE en anglais.

PHP : PHP : Hypertext Preprocessor. Langage de programmation.

SVN : Subversion. Logiciel de gestion de versions.

Références

Afin de concevoir ce document, je me suis appuyé sur les documents suivants rédigés par Anne Castier lors de son projet :

- Le cahier de spécification système et le plan de développement.
- Le rapport de Projet de Fin d'Etude.
- Le guide utilisateur.
- Le manuel d'installation.
- Le cahier de veille technologique.