Rapport - Projet informatique

Félix Bélanger-Robillard

April 5, 2017

Contents

1 Abstract

Le présent projet avait pour but de créer un webservice servant les fonctionnalités de BiBler et de l'intégrer à ReLiS (Revue Littéraire Systématique) afin d'offrir une plateforme web permettant d'allier les fonctionnalités propres à ces deux logiciels, le tout sous la supervision de monsieur Eugène Syriani et avec la précieuse collaboration de Brice-Michel Bigendako.

2 Respect de l'échéancier

2.1 9 février

Objectif initial: Webservice fonctionnel

Cet object a été respecté, en date du 9 février, le Webservice pouvait créer localement un serveur HTTP à l'aide du framework web.py, par contre il n'était pas encore sur un serveur Apache, tel qu'il était prévu éventuellement. Cet objectif a été atteint autour du 1er mars.

2.2 9 mars

Ojectif initial: Intégration avec ReLiS

ReLiS n'était toujours pas fonctionnel de façon locale à ce moment précis. La documentation et les tests avaient donc été entamés dans cette période afin de compenser le délai au niveau de l'intégration. D'un autre côté, une application simple en php était capable d'utiliser sans problème le webservice.

Le 28 mars, après de nombreuses heures de débogages avec Brice, nous sommes arrivés à corriger l'ensemble des bugs qui empêchaient le bon fonctionnement de ReLiS sur mon ordinateur personnel. Deux jours plus tard, Brice me fournissait de nouveaux controlleurs et de nouvelles pages afin de faire l'intégration avec le webservice. La même semaine, nous avons eu l'annonce que nous aurions un serveur uniquement pour ReLiS et le webservice.

Le 4 avril, une mise à jour de ma distribution linux à causer une erreur fatal au niveau de mon ordinateur de travail. Ayant plusieurs backups, je n'ai pas perdu de donner, mais j'ai complètement perdu mon environnement de test et mes configurations de serveur Apache. L'erreur étant systématique, j'ai dû m'adapter rapidement. Après avoir lu sur divers options plus sécuritaires, j'ai installé Fedora 25 sur mon ordinateur et réinstallé Apache ainsi que les modules nécessaires. C'est un incident malheureux qui m'aura appris que les mises à jours peuvent attendre, surtout en temps critiques comme la fin d'une session et à l'approche d'un déploiement.

2.3 9 avril

Ojectif initial: Tests et documentation

3 Technologies employées

3.1 BiBler

BiBler est le fondement du projet. Le webservice s'ancre sur l'API de BiBler et ne fait que délivrer ses fonctionnalités par le biais d'un serveur HTTP.

3.2 ReLiS

ReLiS utilise ensuite les fonctionnalités du webservice dans son importation d'articles, principalement pour générer les clés et pour modifier les entrées de sa base de données. BiBler permet aussi à ReLiS de corriger des fichiers BibTeX mal formatés.

3.3 PHP

Les serveurs de déploiement et le framework employé par ReLiS étant limité à php5.6, j'ai dû télécharger ces versions de php par des voies non officielles des distributions employées lors du projet. C'était également un langage que je connaissais peu, mais considérant que l'intégration se bornait à envoyer le bon paramètre à la bonne méthode du webservice, ça ne m'a pas été d'un grand préjudice, hormis dans le débogage initial de ReLiS.

3.4 Python

La version de python utilisée a été Python 3.6. Cela n'a pas causé de problème de compatibilité avec BiBler qui utilise Python 3.5, mais a obligé l'utilisation d'un autre serveur auprès du DIRO puisqu'ils ne pouvaient déployer que Python 3.4 sur le serveur disponible initialement.

Python est également un langage que je ne connaissais pas, mais la documentation nette, sa syntaxe simple et son orienté objet très similaire à Java a permis une transition assez facile vers ce langage.

3.5 Web.py

Web.py est un framework minimaliste en Python, ce qui permet d'avoir un webservice écrit exclusivement en Python et qui peut être déployer sur une serveur Apache2 tel que requis à la base du projet. Cette option permet de minimiser les dépendances technologiques au minimum.

3.6 Apache2

Le serveur Apache2 est probablement ce qui m'a pris le de temps et d'énergie. Premièrement, il y a eu la nécessité d'installer mysql (mariadb), php5.6, de mettre Python 3 par défaut, en plus des modules spécifiques à Apache tels que mod_wsgi (4.4.1) pour Python.

Le tout m'a grandement familiarisé avec les types de permissions et le fichier de configuration d'apache, ayant eu jusqu'à 5 serveurs HTTP en simultané au cours de la session sur mon ordinateur personnel.

4 Webservice

4.1 Conception

4.2 Fonctionnement

4.3 Tests

4.4 Performance

5 Conclusion