UCL Université catholique de Louvain

Earth and Life Institute (ELI)
Earth and Climate (ELIC)

Projet de premier brevet

Examen d'avancement au grade d'informaticien-expert

Pierre-Yves Barriat

Institut ELI – pôle ELIC

Dénomination de la fonction : informaticien de recherche

Fonction exercée depuis le : 5 novembre 2007

Grade et barème actuels : informaticien – 12/2

Grade et barème sollicités : informaticien-expert – 13/3

Auteur	Date	Version	Commentaire
PY Barriat	07/12/2016	00	Draft initial
PY Barriat	09/12/2016	01	Version corrigée et soumise à RHUM
PY Barriat	14/01/2019	02	Actualisation et fin conception cahier des charges

Place Louis Pasteur, 3 bte L04.03.08, B-1348 Louvain-la-Neuve, Belgique
Tél. +32 (0)10 47 33 02 – fax +32 (0)10 47 27 78 – pierre-yves.barriat@uclouvain.be

Introduction

Contexte

Les chercheurs sont de plus en plus confrontés à la gestion d'énormes quantités de données scientifiques. Ces dernières sont soit produites et analysées localement soit téléchargées depuis les bases de données d'autres centres de recherche ou de dépôts centralisés.
La gestion de cette masse de données constitue un réel problème pour de nombreux pôles de recherche de notre institution. En effet, chaque chercheur ou groupe de chercheurs est soumis à une phase de formation et d'adaptation afin de maîtriser les différentes méthodologies à appliquer pour pouvoir obtenir et travailler sur telles ou telles données.

Historique

Actuellement, du moins en ELIC et dans de nombreux autres groupes avec lesquels nous avons des contacts directs, un étudiant ou un chercheur désirant analyser un jeu de données scientifiques spécifique (domaine, type, sources, etc.) doit d'abord effectuer une recherche plus ou moins informelle : « mon labo possède-t-il déjà les données ? », « un collègue peut-il me les fournir ? », « puis-je les trouver ailleurs à l'UCL ? », « quel organisme, ou quelle institution pourrait me les fournir ? », etc.
Si les données ne sont pas disponibles localement, la seconde étape est de les récupérer, c'est-à-dire les rapatrier depuis un serveur distant vers un serveur de travail afin de pouvoir les traiter et les analyser. Cela nécessite de mettre en place une connexion sécurisée et un protocole spécifique permettant les transferts automatisés d'un grand nombre de fichiers volumineux.
Enfin en dernière étape, le produit de ces données doit être stocké et - éventuellement être - distribué.
Les étapes de ce long processus varient en fonction des centres de recherche, des bases de données, de la nature des données, etc.

Description du projet

Les objectifs

Dans le cadre de ce premier brevet, nous proposons de développer un nouveau service implémentant une gestion automatisée de toutes les étapes du processus décrit ci-dessus.
Les objectifs poursuivis par ce service sont d'offrir :

- une base de données recensant les jeux de données disponibles dans le produit
- une application web pour l'utilisateur
- une application web pour le gestionnaire
- un produit portable, performant, basé sur des technologies libres
- un produit modulaire et adaptable en fonction des spécificités des jeux de données et des opérations à effectuer sur ceux-ci

Afin de faciliter son développent, le service sera initialement orienté vers des données « familières », c'est-à-dire liées aux axes de recherche du pôle ELIC. Il sera toutefois suffisamment général pour être facilement étendu à toute forme de données en fonction des besoins.

Ce projet se positionne au niveau d'un institut de recherche (ELI). Les services de l'institution en relation avec le projet ou impacté sont le CISM, le SRI et le SGSI.

Produit du projet

Les délivrables de ce projet prendront la forme :

D'un protoype de plateforme web dans laquelle l'utilisateur peut :

- sélectionner un ou plusieurs jeux de données (observations, modèles, intervalles, variables, fréquences, etc.) proposés dans l'application (données sur site ou distantes)
- signaler un jeu de données à ajouter ou à modifier
- télécharger un jeu de données vers une machine spécifique
- appliquer des traitements simples des données automatiquement (sélection de variables ou d'un domaine, statistiques de base, etc.)
- produire des figures, des graphiques, etc.
- référencer et/ou téléverser de nouvelles données

D'un prototype de plateforme web dans laquelle le gestionnaire peut :

- gérer les droits des utilisateurs,
- monitorer l'application et le(s) serveur(s)
- gérer la base de données

Catégories du projet: application scientifique, application web, application de gestion

Contraintes

Contraintes de délais

- septembre 2016 : soumission d'acte de candidature à l'examen
- 17 octobre 2016 : validation de l'objet du brevet par la Commission paritaire
- décembre 2016 : soumission du présent projet pour le premier brevet
- 15 décembre 2016 : analyse du niveau du projet par la Commission paritaire
- janvier 2019 : actualisation et initialisation du projet

Durée du projet : 12 mois
Livraison estimée d'un prototype minimum viable : fin 2019
Les échéances intermédiaires sont détaillées dans la section de planification du projet.

Contraintes de réalisation

Ressources

- Accessibilité à une infrastructure de développement et de validation (test) Concernant l'environnement de test, prise en compte des points suivants :
    - localisation et hébergement des serveurs : qualité des performances, possibilité d'accès, coût, sécurité, etc.
    - réseau (type, vitesse et performance des liaisons, disponibilité, support)
    - outils de sécurisation des transactions et des serveurs (certificats, authentification par clés, mots de passe, firewall, etc.)
    - procédures, outils et ressources pour assurer la gestion et la maintenance : du réseau, du matériel, des logiciels, des accès, de l'usage, des coûts, du support utilisateur, de la performance, etc.
- Accessibilité à des données extérieures (par exemple : NASA, CMIP, ECMWF, etc.) : identification des sources de données, démarches administratives, protocoles de connexion (« batch process » possible sur DB distante ou non ?), coûts éventuels, etc.

Techniques

- localisation des applications et des bases de données (répartition des processus applicatifs entre serveurs, etc.)
- outils utilisés issus de logiciels libres
- environnement de travail
- architecture et fonctionnalités de l'application :
    - pages web
    - middleware
    - bases de données
- langages de développement
- outils de sécurisation des transactions et des serveurs (certificats, authentification par clés, mots de passe, firewall, etc.)
- performances que le système doit supporter dans 90% des cas : temps de réponse utilisateur, outils de mesure des performances, disponibilité requise, etc.)
- reproductibilité et persistance de l'application
- évolutivité de la solution (possibilités)

Autres contraintes

- documentation requise
- méthode d'analyse (performance, risques)
- évolutivité de la solution (coût)
- plan de formation des utilisateurs et des gestionnaires
- support utilisateurs

Déroulement du projet

Planification

Les grandes phases du projet seront les suivantes :

Phase 1 : Initialisation du projet

Fin de rédaction du cahier des charges, choix techniques
Calendrier : fin janvier 2019

Phase 2 : Analyse et conception

Conception globale de l'application : analyse fonctionnelle, modélisation
Calendrier : fin mars 2019

Phase 3 : Développement

Evaluation à l'aide du cahier des charges en cours…
Calendrier : fin août 2019

Phase 4 : Tests d'intégration

Intégration de l'ensemble des développements dans l'environnement de test
Calendrier : fin septembre 2019

Phase 5 : Documentation et présentation

Documentation de l'application + présentation du prototype aux utilisateurs
Calendrier : fin novembre 2019

Organisation et suivi

La phase d'initialisation du projet est soumise à la validation de la Commission paritaire. L'ensemble des activités introduites dans la planification des tâches sera discuté et suivi par Hugues Goosse en tant que coach-évaluateur et par Yves Deville en tant que second évaluateur. RHUM sera régulièrement informé du suivi global du travail. Le développement du projet nécessitera des interactions avec de nombreux groupes au sein de l'UCL comme détaillé dans les spécifications techniques.

Evaluation

La méthode d'analyse et les critères d'évaluation du projet sont soumis au règlement des examens d'avancement au grade d'informaticien-expert (document du 30 janvier 2006).

Spécifications techniques

Quelques pistes pour les points encore à définir :

- connexion réseau GB, agrégation de liens, etc. (contact : SRI)
- localisation serveurs (contacts possibles : CISM pour DCIII ou SGSI pour Pythagore)
- serveur(s) de stockage (contacts possibles : ELI, ELIC, CISM, SGSI)
- serveur(s) d'application (contacts possibles : ELI, ELIC, CISM, SGSI)
- Environnement de travail sous distribution linux virtualisée (Vagrant, Docker ?)
- Reproductibilité aisée de la configuration (Ansible ?)
- Persistance via système de gestion de versions (Git ou Mercurial)
- Application en architecture 3-tiers à client léger

- Plusieurs possibilités de langages de développement pour les couches présentation & middleware: PHP, R, Python, Java, Nodejs
- Possibilités de frameworks associés : Zend ou Symfony, Shiny, Django, Spring, Express, etc.
- Possibilité d'ajout d'objectifs supplémentaires pour la couche middleware: programmation fonctionnelle (scala) et/ou « language agnostic » pour les différentes tâches
- Couche accès aux données locale puis via DB distribuée : multi-sites UCL et multi-sites externes (sans doute hors MVP)
- Base de données SQL (MySQL, Postgresql ?)
- Echange de données inter-sites via protocole GridFTP (sans doute hors MVP)

Spécifications de réalisation

Quelques points encore à détailler :

- maquette ou démonstration fonctionnelle : objectifs, représentativité par rapport au projet complet, configuration, plan de travail, ressources, critères d'acceptation avant de poursuivre les travaux ;
- détails du calendrier des prestations : début, fin, phases, check-points ;
- planning de disponibilité des ressources mises à disposition (quantité, qualification, dates, lieux)
- méthodologie, plan et outils requis pour effectuer les tests :
    - fonctionnels, de performance et de qualité,
    - de montée en charge du réseau et des applications, d'ergonomie,
    - des fonctions de sauvegarde et de reprise ;

Catégories du projet: application scientifique, application web, application de gestion

Analyse et conception du projet

Choix techniques

L'objectif est la création d'une interface de gestion de données sous forme d'une application web service.

Framework

Un framework est un cadre qui permet de structurer le travail de développement grâce à un ensemble d'outils, une structure et des modèles prêts-à-l'emploi.
Étant donné l'étendue des développements à effectuer pour concevoir une application web moderne, un framework est indispensable.

Django est un framework Backend Open Source développé en Python. Il a été spécialement créé pour réaliser des sites web puissants et de haut niveau. Il embarque tous les composants utiles, que ce soit la gestion de vues, l'authentification, le mapping objet-relationnel, une documentation détaillée, etc.

Python est un avantage car c'est le langage le plus utilisé par les chercheurs en ELIC. En outre, les services IT de l'UCL utilisent également ce framework pour les nouveaux développements web.
Une alternative solide serait Ruby on Rails (RoR). Il est le framework libre le plus populaire ces 5 dernières années, et a été conçu pour développer des applications web plus rapidement. Il permet aux développeurs de créer des fonctionnalités avec moins de code. Mais si RoR nécessite peu de configuration, il exige aussi plus de conventions. En outre, le niveau d'expertise pour se lancer est une barrière à l'entrée pour les débutants. Enfin Ruby nécessite des ressources serveur plus importantes que Django et sa technologie comme son utilisation sont en déclin.

Python/Django sera préféré pour la conception du projet.
C'est un framework Full-Stack - il est très facile de combiner Django et Angular par exemple - et tout clé en main : modèles, côté serveur, panneau d'administration pour configurer un site sans coder, etc. Il utilise, comme souhaité, le patron de conception modèle-vue-contrôleur (MVC), c'est à dire que la structure du framework sépare les données (models) qui sont séparées des traitements (controller) qui sont eux-mêmes séparés de la vue (view/template). C'est également un outil idéal pour un projet collaboratif.
Django étant très populaire auprès des développeurs web, de nombreux projets sont apparus autour du framework. Par exemple dans notre cas, Ncdjango est un ensemble d'outils de gestion de données et de géotraitement écrits en Python qui fonctionnent sur des données NetCDF.

Environnement

Cette application sera conteneurisée. La conteneurisation logicielle permet une gestion simplifiée des dépendances: une application et toutes ses dépendances sont placées dans une seule unité. Le système hôte ne doit pas se soucier de ces dépendances. L'application conteneurisée est donc indépendante de l'architecture ou des ressources de l'hôte. Elle est donc plus flexible et plus facilement distribuable.
Si cette conteneurisation apporte son lot d'avantages en développement et pour les tests de validation, son utilisation reste plus discutable dans le contexte d'une mise en production. Nous en rediscuterons plus en avant dans ce projet.

Docker est la solution de conteneurisation la plus utilisée aujourd'hui. C'est un logiciel libre qui utilise une interface de programmation « Libcontainer » pour démarrer, gérer et arrêter les conteneurs. Il est basée sur le fonctionnement de LXC et y ajoute des capacités de niveau supérieur. Les conteneurs Docker peuvent servir d'images à d'autres conteneurs et le partage de conteneurs en public est possible via un service en ligne appelé Docker Hub. Il contient des images de conteneurs, ce qui permet aux utilisateurs de faire des échanges. Cela rend l'installation d'un conteneur extrêmement facile.

Outil de développement

PyCharm est un environnement de développement intégré utilisé pour développer en Python ainsi qu'avec Django. Il propose la possibilité de débugger en direct dans un conteneur Docker.
Vagrant est un logiciel libre et open-source pour la création et la configuration des environnements de développement virtuel. Il peut être considéré comme un wrapper autour de logiciels de virtualisation comme VirtualBox.

Méthodologie

L'application sera donc standardisée MVC, c'est-à-dire selon une architecture classique à trois couches.
La couche vue sera développée très simplement sur base de templates existants à l'UCL.
Les couches traitement et modèle présenteront les cas de figure suivants:

- données locales: traitement "on the fly" sur DB(s) locales 130.104
- données distantes
    - à posteriori (DB & protocoles connus)
    - à priori (infos de structures à soumettre)
- données distantes
    - indexées: traitement "on the fly" (batch process possible sur DB distante)
    - non-indexées: traitement différé (DB distante accessible en interactif uniquement)

Scénarios pour les données à posteriori et non-indexées :

- téléchargement tiers + demande d'intégration aux DB
- téléchargement à travers l'appli + intégration automatique aux DB locales

Comme nous souhaitons mettre à disposition des données pour quelles soient utilisées sur d'autres plateformes et qu'elles puissent intéragir avec d'autres données, une architecture REST ("REpresentational State Transfer") semble appropriée ici.
L'architecture REST est plus axée sur un modèle orienté ressources (les données, dans notre cas) que sur un modèle orienté fonctions. Elle imite la façon dont le web lui-même fonctionne dans les échanges entre un client et un serveur.
REST constitue donc une méthode d'intégration efficace puisque le service à développer ici concerne surtout la récupération de données. Aussi, plutôt que de définir toute une API (interfaces de programmation d'application) personnalisée mieux vaut utiliser un standard de manipulation des données CRUD (Create, Read, Update, Delete : créer, lire, mettre à jour, supprimer), qui "correspond" aux opérations HTTP (HyperText Transfert Protocol) GET, PUT, POST et DELETE. Ce fonctionnement ne repose pas sur la seule utilisation de ces opérateurs, mais sur une combinaison avec des URI.

Le "Django REST framework" va nous permettre de créer plus facilement une API REST sur notre application Django.

Un interfacage avec Amazon S3 serait un atout supplémentaire.