Université Claude Bernard Lyon 1
Master Informatique

2015 - 2016

MIF-TI5

Projet PERF

Conception d'une plateforme pour l'alignement d'entités par flux

Alexandre Millot
Corentin Lonjarret
Antoine Hintzy
Murat Halat
Felician Raphael Matinya

Tableau des révisions du document

Version	Date	Modification
1	04/10/2015	Première version complète du dossier d'initialisation
2	10/10/2015	Seconde version modifiée du dossier d'initialisation (modification du contenu)
3	12/10/2015	Troisième version modifiée du dossier d'initialisation (modification du contenu et de la forme)
4	15/10/2015	Quatrième version modifiée du dossier d'initialisation (modification du contenu)

Table des matières

1 Introduction

1.1 Contexte

Dans le cadre des enseignements de base de données, et plus précisément des enseignements liés à l'UE MIF37, le projet PERF a pour objectif de munir l'équipe pédagogique BD d'un générateur de flux d'entités sémantiques facilement déployable, partageable pour l'ensemble d'une promotion d'étudiants et capable d'agréger les résultats de performance et de qualité générés par les algorithmes appliqués au flux d'entités (par exemple, des algorithmes d'alignements d'entités). De plus, la plateforme pourra également être exploitée dans un contexte orienté recherche.

1.2 Équipe

Ce projet est réalisé sous la direction de Fabien Duchateau et Nicolas Lumineau, porteurs du projet, maîtres de conférences à l'Université Lyon 1 et enseignants-chercheurs au sein du LIRIS, pôle Data Science, équipe Base de Données.

Notre équipe de développement est la suivante :

- Alexandre Millot : Chef de projet, correspondant avec les encadrants, concepteur base de données, développeur
- Corentin Lonjarret : Scrum master, concepteur base de données, développeur
- Antoine Hintzy : Expert technique partie web, développeur, designer
- Murat Halat : Développeur, concepteur
- Felician Raphael Matinya : Développeur, concepteur

1.3 Description du projet

1.3.1 Définition des termes et rôles

Pour appréhender le projet convenablement, il est nécéssaire de comprendre les notions fondamentales du projet.

1. On retrouve tout d'abord la notion **d'entité**. Une entité peut être définie comme un élément possédant plusieurs attributs le décrivant et le rendant unique. Par exemple, une personne peut être considérée comme un élément composé de plusieurs attributs, notamment son nom, son prénom, sa date de naissance et son adresse. Les valeurs données à ces attributs ainsi que la présence d'un attribut servant d'identifiant font d'une personne une entité unique.

2. Il convient également de définir la notion de ***dataset*** dans le cadre du projet. Un *dataset* est un ensemble d'entités provenant de différentes sources. Les *datasets* sur lesquels nous baserons notre travail sont consultables ici[1].

3. La notion de **fichier expert** est également abordée. Un fichier expert contient l'ensemble des alignements d'entités à retrouver dans un *dataset*. Celui-ci permet d'évaluer les performances (temps,précision) de l'alignement produit par un client.

4. Le ***master*** est la machine principale du système. Elle permet notamment la création et la configuration de *workers*. C'est l'entité qui s'occupe de la répartition de charge et qui reçoit les résultats provenant des clients.

5. Un ***worker*** est une machine sous la responsabilité d'un *master.* Celle-ci effectue les traitements que le master souhaite la voir accomplir.

Il existe deux rôles au sein de la plateforme que nous nous proposons de mettre en place. Premièrement, le rôle d'administrateur. Celui-ci est réservé aux professeurs qui seront en charge de gérer la plateforme et leur permettra de configurer le système selon leurs besoins. Deuxièmement, le rôle de client. Il sera, de façon générale, réservé aux étudiants qui seront les principaux utilisateurs de la plateforme.

1.3.2 Objectifs

Le projet est composé de deux objectifs principaux. Le premier objectif est d'élaborer un outils permettant d'orchestrer/administrer un ensemble de flux répartis sur différentes machines. Chaque machine aura la responsabilité de garantir pour son ou ses flux les contraintes/ propriétés que l'administrateur aura fixé (par ex. le débit du flux).

Le second objectif consiste à élaborer un outil permettant de collecter, d'agréger et de visualiser les performances des algorithmes appliqués aux flux d'entités.

De manière transversale aux deux objectifs, les outils élaborés devront pouvoir supporter une montée en charge induite par les contraintes imposées sur les débits des flux et le nombre de clients/d'étudiants utilisant la plateforme.

1.3.3 Fonctionnalités

Concernant les fonctionnalités principales, un administrateur devra être capable :

- De déployer et configurer rapidement et facilement la plateforme ;
- De se connecter à la plateforme en tant qu'administrateur ;
- D'ajouter des *workers* pour gérer la montée en charge du système (augmentation du nombre de clients simultanés) ;
- D'ajouter de nouveaux *datasets* aux formats CSV et XML à la base MongoDB ;
- De définir les *datasets* à envoyer aux clients, ainsi que le débit d'envoi des entités ;
- De consulter les performances des résultats retournés par les clients.

---

[1] Dataset for entity resolution (Univ. Leipzig)
http://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution

Un client devra avoir la possibilité de:

- Dialoguer avec le serveur via une API Java en ligne de commande ;
- Récupérer l'ensemble des entités d'un *dataset* ;
- Retourner au serveur les résultats du traitement effectué sur les entités.

Parmi les améliorations envisagées, on compte :

- L'ajout de *datasets* au format SQL ;
- La possibilité pour les étudiants de visualiser le classement des performances ainsi que leur rang face aux autres étudiants via une interface côté client.

1.3.4 Architecture du système

Ci-dessous (Figure 1), une première version de l'architecture proposée pour la plateforme que nous allons développer.

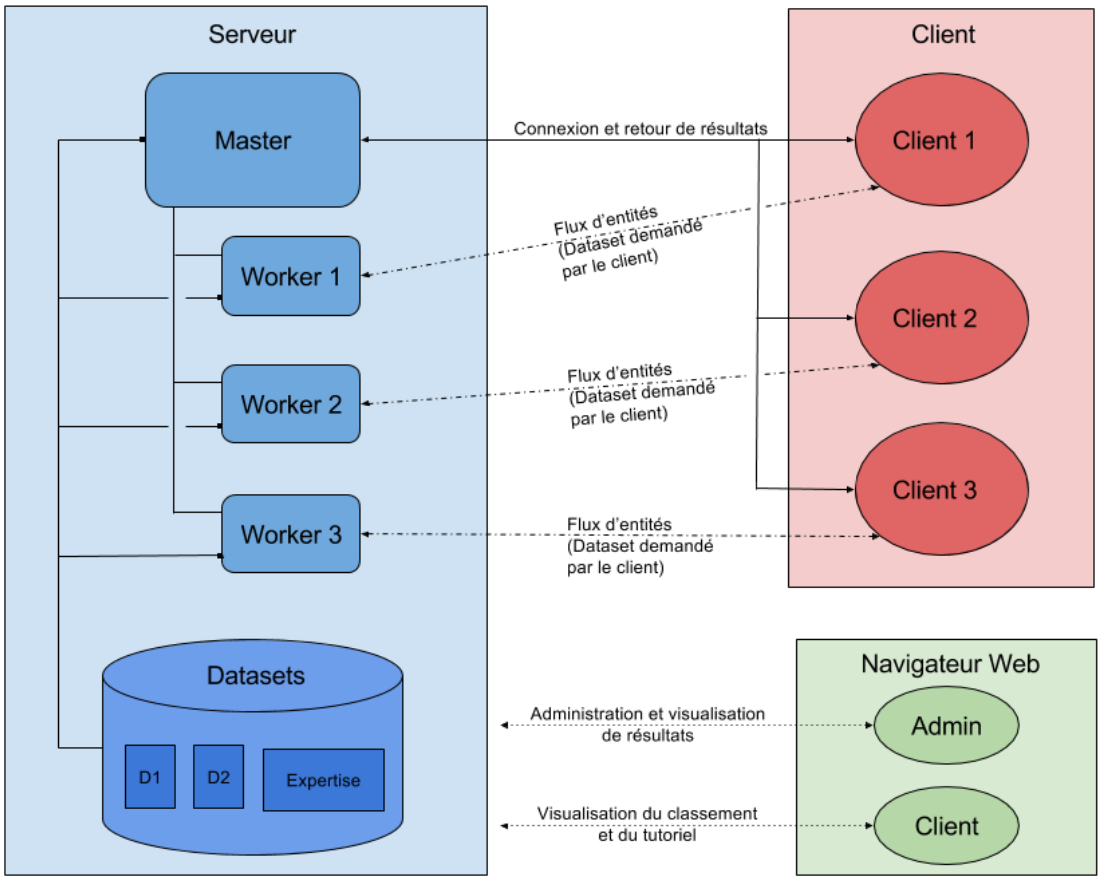

Figure 1 : Architecture Du Système

1.3.5 Scénarios

1.3.5.1 Cas d'utilisation : Administrateur/Enseignant

- Connexion à l'application via la page d'accueil (identification en tant qu'administrateur).
- Créer un projet (nom + descriptif).
- Ajout des *workers* nécessaires à la gestion des futurs clients.
- Ajout/sélection dans la base d'un ou plusieurs *datasets.*
- Lancer le projet (fin de la configuration).
- Connexion au tableau de bord pour visualiser les *workers* et leur activité (nombre de *worker*, débits demandés, débits assurés, défaillances).
- Générer un bilan "en l'état" des performances des algorithmes en cours d'évaluation.
- Clôture le projet et éditer les résultats.

1.3.5.2 Cas d'utilisation : Client/Etudiant

- Connexion à la page d'accueil de façon à se connecter à l'application.
- Accès au guide d'utilisation.
- Un client ouvre une connexion via l'api et demande au serveur (master) de recevoir des entités via un flux.
- Demande de connexion à un *worker* (récupération des ports d'émission des flux).
- Le client doit effectuer un traitement sur le flux d'entités (par exemple un algorithme d'alignement), puis renvoyer les résultats obtenus vers le serveur (master) en respectant un format imposé.
- Le serveur ayant accès aux fichiers experts utilise alors un algorithme de notation des résultats fournis par le client (évaluation).
- Affichage des résultats de performances de l'algorithme testé.
- Clôturer la connexion au *worker*.
- Connexion à la page d'agrégation des performances de la promotion (classement).

2 Résultats attendus

2.1 Livrables GDP

- Dossier d'initialisation D0 (12/10/2015)
- Dossier post-sprint D1 (26/10/2015)
- Dossier post-sprint D2 (07/12/2015)
- Dossier post-sprint D3 (08/02/2015)

2.2 Livrables techniques

2.2.1 Post-sprint 1

- Produit logiciel : Version minimale fonctionnelle du système mettant en œuvre l'ensemble des technologies choisies – Il devra notamment être possible de réaliser des échanges simples d'entités entre serveur et client. Les entités auront été ajoutées à la base et récupérées via le système.
- Première version du document architecture logiciel : décrit, notamment de façon schématique, les différents éléments du système.

2.2.2 Post-sprint 2

- Produit logiciel : Version entièrement fonctionnelle en local – Le système devra permettre l'échange d'entités entre plusieurs *workers* et clients à des débits différents et de noter les résultats fournis par les clients.
- Prototypes des interfaces.

2.2.3 Post-sprint 3

- Produit logiciel : Version finale/beta fonctionnelle.
- Documentation de déploiement et d'administration du logiciel : Document destiné aux administrateurs qui expliquera de quelle façon facilement déployer et administrer le système.
- Tutoriel d'utilisation du logiciel.
- Documentation de développement du logiciel : Permet d'expliquer les structures de données ainsi que les algorithmes dont est constitué le logiciel.
- Version finale du document architecture logiciel.

2.3 Autres

- Dossier de synthèse final
- Soutenance

3 Méthodes et outils

3.1 Contraintes

Contraintes de temps : Le but affiché est de pouvoir utiliser ce projet dans le cadre de l'UE M1IF37 du printemps 2016. Le système devra donc être fonctionnel et prêt à utiliser à la fin de l'UE TI5.

Contraintes de coûts : Une charge de travail total de 465h est prévue pour le projet.

Contraintes techniques :

- Le système doit être en mesure de supporter la montée en charge. La plupart des tests seront réalisés avec 1-2 clients, cependant le système devra être capable de supporter jusqu'à 35-40 clients simultanément.
- La facilité de déploiement et d'administration du système est également une des contraintes clés du projet.
- Le déploiement possible sur une machine unique (*master* et *worker*).
- Le déploiement sera fait sous Linux.

3.2 Méthode

Nous avons décidé d'adopter une méthode agile comme méthode de gestion de projet, et plus précisément la méthode Scrum que nous avons déjà eu l'occasion de mettre en œuvre. Cela nous permettra notamment de prendre en compte l'évolution des besoins au fur et à mesure du projet.

Le projet sera composé de trois sprints et, avant chaque sprint, une liste des tâches a effectuer précise à la demi-journée sera mise en place. A la fin de chacun des sprints, une réunion aura lieu avec les porteurs du projet de façon à leur présenter la recette du sprint courant.

3.3 Outils

3.3.1 Environnement technique

Concernant l'environnement technique dans lequel nous allons travailler, les différentes contraintes ainsi que nos propres choix nous ont amenés à considérer les technologies suivantes:

- Concernant la partie stockage des données, le choix d'adopter **MongoDB** vient du fait que les données proviennent de sources hétérogènes et possèdent des structures différentes les unes des autres.
- **Express**, framework proposant plusieurs fonctionnalités permettant de créer un serveur web fonctionnel plus simplement qu'avec Node.js seul. Simplifie la création de services web (API REST, basée sur CRUD) et gère automatiquement les statuts HTTP.
- **Node.js**, pour le côté serveur.
- **AngularJS**, pour la partie client.
- **Sass** permettant la génération dynamique de feuilles de style.
- **HTML5** pour la structuration et la mise en forme des pages web.
- **Grunt**, pour la création de tâches automatisée en JavaScript.
- **npm** comme gestionnaire de paquets pour Node.js.
- Un gestionnaire de dépendances pour le web appelé **Bower**.
- **Docker**, logiciel permettant d'automatiser le déploiement d'applications dans des conteneurs logiciels. Concrètement, docker permettra d'empaqueter notre application dans un conteneur virtuel qui pourra être exécuté sur n'importe quel serveur Linux.

- **Cron**, programme permettant l'automatisation de scripts, notamment selon un cycle défini à l'avance. Celui-ci servira à automatiser l'envoi des flux d'entités selon un intervalle de temps.

Remarque : Le choix d'adopter les technologies précédentes nous a poussé à choisir **MEAN.JS**, un framework javascript permettant de simplifier et d'accélérer le développement d'applications web utilisant **M**ongoDB, **E**xpress, **A**ngularJS et **N**ode.js.

3.3.2 Gestion de projet

Pour mener à bien ce projet, nous avons décidé d'utiliser différents outils qui permettront d'améliorer la communication ainsi que le versionnage du projet et des documents qui lui sont associés :

- La **forge** qui permettra de suivre l'évolution des versions du projet ainsi que de gérer les tickets.
- **Trello**, pour la gestion des tâches, notamment la répartition de celles-ci entre les différents membres pour chaque sprint ainsi que pour la gestion de la rédaction des documents techniques.
- **Slack**, pour améliorer la communication entre les membres du groupe. Cet outil nous permettra notamment de discuter de façon directe des nouvelles idées, de l'architecture, des tâches à réaliser ainsi que des changements à envisager pour le projet.
- **Google Drive**, pour le partage des différents documents et schémas au fur et à mesure du projet.
- **Doodle**, pour simplifier la planification des réunions du groupe.

4 Macro-planning

4.1 Lots de travail

Cette partie concerne le découpage en lots de travail du projet à réaliser. Une version Gantt simplifiée des lots de travail est disponible (Figure 2).

1. Conception, implémentation de la base de données MongoDB (19/10/2015 au 23/10/2015).
    - Ajout et récupération de *datasets* au format CSV.
    - Calcul et stockage des notes au format JSON.

2. Conception et implémentation des interfaces (19/10/2015 au 05/02/2016)
    - Page de configuration pour les administrateurs.
    - Tableau de bord pour les administrateurs.
    - Page d'affichage des résultats côté client pour les étudiants.
    - Page d'accueil servant de tutoriel d'utilisation du système.

3. Conception et implémentation de l'API (19/10/2015 au 05/02/2016)
    - Partie serveur.
    - Partie client.
    - Communication client / serveur.

4. Conception et implémentation de la partie «container» (01/02/2016 au 05/02/2016)
    - Déploiement du *master* et des *workers.*
    - Mise en place de la communication *master-workers.*
    - Répartition de charge.

5. Gestion de projet, documentation et autres (28/09/2015 au 18/02/2016)
    - Documentation technique.
    - Documentation de gestion de projet.
    - Démonstration et vidéo de présentation.

6. Test des fonctionnalités et ajout de tests unitaires (19/10/2015 au 05/02/2016)
    - Tests de montée en charge.

7. Améliorations (01/02/2016 au 05/02/2016)
    - Ajout d'un document en base contenant le classement des performances des étudiants.
    - Import de *datasets* aux formats XML et SQL.

4.2 Phasage

Cette partie concerne la répartition du travail entre les différents sprints ainsi que les attentes concernant l'état du projet à la fin de chacun des sprints planifiés.

4.2.1 Tâches et attentes sprint 1

- Ajout et récupération de données dans la base MongoDB.
- Implémentation basique Serveur et Client avec connexion entre ces derniers.
- Échange simple d'entités au format JSON via *sockets*.
- État d'avancement du travail fourni dans les tâches précédentes et tests de fonctionnement du système unifié.
- Démonstration prévue: Une première version simple et fonctionnelle du système en local.

4.2.2 Tâches et attentes sprint 2

- Répartition des tâches avec plusieurs serveurs (un master et deux *workers* dans un premier temps).
- Création de l'interface administrateur.
- Création de la page d'accueil qui se présentera comme un tutoriel d'utilisation pour les clients.
- Création de l'interface client.
- Tableau de bord pour les administrateurs.

- Gestion des débits des flux d'entités entre *worker* et client.
- Script de notation des élèves en fonction des résultats fournis.
- Premiers tests de la partie « container ».

Démonstration prévue: Version entièrement fonctionnelle du système en local, sans amélioration.

4.2.3 Tâches et attentes sprint 3

- Mise en place complète de la partie « container » avec Docker.
- Gestion de classements au niveau base de données, client et serveur.

Phase de tests final et tests de montée en charge.

Démonstration prévue: Version finale fonctionnelle du système.

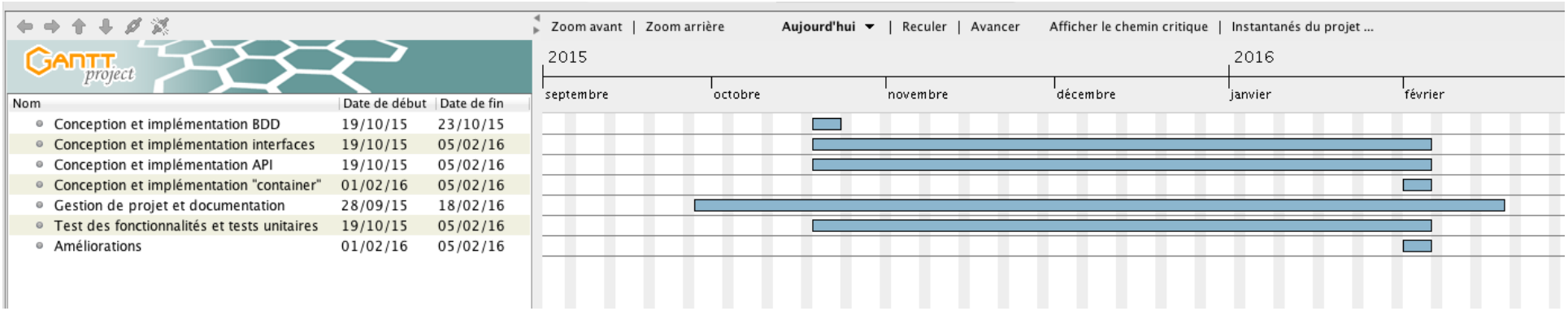

Figure 2 : Planification Des Lots De Travail