



Université de Technologie de Compiègne  
rue du Dr Schweitzer  
Compiègne, 60200 France

MULTIPOSTING



MULTIPOSTING  
3 Rue Moncey  
Paris, 75009 France

## Rapport de Stage d'assistant ingénieur

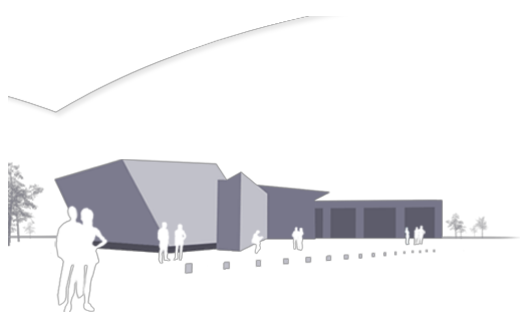
TN09

Filière Ingénierie des Connaissances et des Supports d'Informations (ICSI)

# Développement, maintenance et amélioration du système de multidiffusion d'offre d'emploi

Suiveur l'UTC : Monsieur Claude MOULIN

Responsable Multiposting : Monsieur Alexandre SADONES



KEWEI HU

02/06/2014



# Remerciement

Je tiens tout d'abord à remercier Monsieur Benjamin BONNY, le responsable d'équipe générateur, qui m'a beaucoup aidé durant ce stage. Ses largement aides sont très précieux.

Je tiens mon grand remercie à Alexandre SADONES, CTO de Multiposting, de m'avoir permis d'intégrer son équipe et pour toute l'aide apporter qu'il m'a apporté afin.

Je tiens aussi à remercier Monsieur Pierre-Yves SCOPOLINI ainsi que tous les membres de l'équipe qui m'ont beaucoup aidé durant ce stage.

Mes remerciements vont à Florian GARCIA, Anthony RAYMOND, Bruno Devaux, Joris Andrade, Damien Coqueux et Guillaume Gomez tous collègues et membres des équipes Générateur et Support, pour les bons moments, les rires, et l'excellent travail d'équipe qui ont fait de ce stage une expérience humaine et professionnelle très enrichissante.

Mes remerciements vont à Axel VITTECOQ, Administrateur Système, pour avoir su gérer le réseau d'une main de fer et avoir toujours répondu présent lors des quelques pannes.

Ma reconnaissance va également à Anne-Sophie VASSEUR et Geneviève NOUGAYREDE, dirigeant les Ressources Humaines, pour tout le suivi et l'attention pendant tout au long du stage.

Tila TRUJILLO, toute l'équipe Support Client qu'elle dirige, Pierre-Gaël PASQUIOU, Lucile OSTERTAG, et toute l'équipe Biz-dev ont ma reconnaissance pour leur excellente organisation des tâches, leur communication sans faille avec les développeurs.

Je tiens à remercier Monsieur Claude MOULIN, pour sa visite au Multiposting et ses aides très précieuse.

Plus généralement, je remercie l'ensemble du personnel de Multiposting pour leur esprit jeune et dynamique, leur travail d'équipe efficace, leur accueil chaleureux et l'excellente ambiance qu'ils ont su instaurer, pendant le travail mais aussi lors des soirées. Ils ont fait de ce semestre de stage une très bonne expérience professionnelle et humaine.

# Résumé

Ce travail a été réalisé dans le cadre de mon stage d'assistant ingénieur(TN09) de l'Université de Technologie de Compiègne chez la société Multiposting à Paris. J'ai intégré dans une équipe de développement sur un projet en cours. La plus grosse partie d'analyse ont déjà été fait, le système mis en place est fonctionnel, le but de mon stage sera de maintenir l'application en place en corrigeant les bugs, testant des fonctionnalités sur le système actuelle et de participer à l'amélioration du système en ajoutant de nouvelles modèles et méthodes, intégrant des nouveau site et complétant les outils interne pour faciliter la maintenance du projet.

Le projet répond aux besoins des recruteurs qui veulent gagner en temps et en efficacité dans le domaine de l'e-recrutement. On propose aux recruteurs un produit qui va leur permettre de diffuser leur offres d'emploi (de stage, de contrat d'apprentissage, etc.) sur plusieurs sites (jobboards<sup>1</sup> et écoles). Ce rapport a pour le but de décrire, positionner et analyser mes missions et mes activités au sein de Multiposting, faire la synthèse des travaux effectués pendant mon stage et faire un bilan des compétences acquises au regard des compétences nécessaires à l'exercice du métier d'ingénieur.

Mots-clés : Multidiffusion, Jobboard, Développement Web, Générateur, E-recrutement

---

<sup>1</sup> Un jobboard est un modèle de site emploi permettant aux demandeurs d'emploi d'accéder à une liste d'offres d'emploi et de déposer leur CV dans une base de données à destination des recruteurs

## Contenu

Présentation .....	1
Présentation de l'entreprise .....	1
Présentation du produit.....	2
Présentation des équipes.....	3
Stage .....	4
Langues, outils, technologies et méthodologies .....	4
Langues informatiques .....	4
Framework.....	5
Méthodologie.....	6
Outils .....	14
Concept principale de multiposting.....	16
Frontend.....	16
Backend .....	17
Base de donnée .....	18
Comment ça marche pour la multidiffusion .....	18
Concept.....	18
Processus de multidiffusion.....	21
Traitement d'erreur.....	23
Déroulement du travail .....	25
Mission .....	26
Projet Générateur.....	26
Projet Report.....	41
Conclusion.....	47
Bibliographique .....	48
Annexe.....	49

# Présentation

## Présentation de l'entreprise

### Le contexte

L'avènement d'Internet et son incroyable expansion ont instantanément changé la face du Monde, sur les plans personnel, économique, social, relationnel, professionnel... Durant les dernières décennies qui ont vu croître cette technologie révolutionnaire, les sociétés de tous les pays ont dû très rapidement s'adapter à cet outil, modifier leur façon de travailler, de communiquer, d'appréhender l'information et de concevoir les choses, afin de rester compétitives. On ne compte plus le nombre de nouvelles entreprises qui, ayant étudié le fonctionnement d'Internet, ont su en déterminer les forces, les faiblesses, les ressources et les besoins, et ont ainsi pu exploiter des opportunités que cet outil présentait. Certaines de ces entreprises ont particulièrement bien réussi, et sont devenues les géants que l'on connaît aujourd'hui : Google, Facebook...

Un exemple de secteur ayant subi d'importants changements et celui des ressources humaines. Etant évidemment basé sur la communication entre différents agents, l'arrivée d'Internet ne pouvait que modifier tous les outils et méthodes déjà en place. Les sites se spécialisant dans la recherche d'emploi ou dans la recherche de futurs employés ont donc fleuri par milliers, répondant aux besoins exprimés, de différentes manières : LinkedIn, Monster, Vivastreet, RegionsJob...

La plupart d'entre eux fonctionnent par un système d'annonces : l'entreprise, cherchant de nouveaux employés, poste une annonce sur le site, contenant à la fois des informations sur le poste proposé et des critères que doivent satisfaire les candidats. Ce dépôt peut être payant, gratuit avec certaines options payantes, ou même entièrement gratuit. Chaque site dispose de son propre modèle économique. De même, de très nombreux sites d'écoles et d'universités proposent un formulaire pour déposer des offres de stages, ou des offres d'emploi à destination des jeunes diplômés.

Statistiquement, donc, plus le nombre de sites diffusant l'annonce d'une entreprise est grand, plus cette entreprise a de chances de trouver son candidat idéal. Elle devient bien plus visible.

Cependant, on vient de voir que chacun de ces sites disposait de son propre modèle économique. De plus, le processus pour poster une annonce est souvent fastidieux, et varie de site en site : il faut parfois créer un compte, parfois non, renseigner des informations nécessaires sur un site mais pas sur l'autre... Pour la plupart, il faut remplir un très long formulaire, pour d'autres cela fonctionne par mail...

Le responsable Ressources Humaines d'une entreprise ne peut donc décemment pas passer ses journées à diffuser l'annonce sur tous ces sites : cela est une perte de temps (et donc d'argent) gigantesque...

Multiposting est une jeune entreprise qui propose une solution à ce problème.

### La naissance de Multiposting

Multiposting a été créée en 2008 par Gauthier MACHELON et Stéphane LE VIET. Leur objectif était de fournir aux entreprises un outil leur permettant d'automatiser le processus précédemment décrit. A partir d'informations fournies par un de ses clients, l'entreprise Multiposting devait être en mesure de multidiffuser l'annonce sur autant de sites que possibles, ainsi que d'assurer le suivi de ces annonces, et le cheminement des réponses des candidats. Très rapidement, l'entreprise a connu une croissance phénoménale.

le produit phare est la 1ère solution de multi-diffusion d'annonces d'emploi et de stage sur Internet. Avec une croissance annuelle supérieure à 100%, Multiposting travaille désormais avec plus de 800 clients (Total, PwC, Thales, Crédit Agricole, Lagardère, Société Générale, Capgemini, Dassault Systèmes), dont 32 sociétés du CAC 40.

Les fondateurs de Multiposting sont à l'origine d'une seconde start-up, Work4 Labs, basée à San Francisco, devenue en un an le leader mondial des solutions de recrutement sur Facebook et qui compte aujourd'hui plus de 17 000 clients.

### Chiffres-clés

- Plus de 100 employés.
- Une croissance de 100% par an.
- 7,75 millions d'euros de chiffres d'affaires en 2013.
- Multidiffusion sur plus de 500 sites d'emploi et plus de 1800 écoles.
- Plus de 7 millions d'annonces diffusées en 2012, dans plus de 50 pays, par plus de 800 clients, dont 32 sociétés du CAC 40.

### Le marché

De par la nature du produit principal vendu par Multiposting, les clients les plus ciblés sont les grandes entreprises disposant d'un important service de Ressources Humaines à part entière. En effet, ayant une forte activité dans ce domaine, ce sont celles qui tirent le maximum d'avantages de l'automatisation de ce processus. C'est pour cela qu'autant de grandes sociétés du CAC 40 sont devenues clients de Multiposting.

## Présentation du produit

Dans ce projet on fait office d'intermédiaire entre les recruteurs, les jobboards, les écoles et les candidats. Le but est de permettre la multidiffusion d'offre d'emploi pour faire gagner du temps dans la saisie des annonces et mesurer la performance des sites utilisées. Les recruteurs pourront poster leurs offres d'emplois (de stage, ou autre type de contrat) sur plusieurs jobboards ou écoles après avoir saisi son offre une seule fois. Multiposting propose également d'autres services, Voici les trois produits principales :

**MULTIPOSTING**



**Multiposting** : Multi-diffusez vos annonces en 1 clic sur plus de 500 sites, 1 800 écoles et sur les réseaux sociaux. Multiposting est le produit principal de l'entreprise. Il

permet la multidiffusion d'offres d'emploi. L'entreprise cliente se connecte sur son espace sur le site Multiposting et peut, à partir de celle-ci, poster de nouvelles annonces, les gérer, ou consulter les statistiques.



## MULTISEARCH

**MultiSearch** : Trouvez le candidat idéal en une seule recherche sur les plus grandes CVthèques (gratuites et payantes) ainsi que sur les réseaux sociaux.

MultiSearch est un outil de multi-sourcing. Cela signifie qu'à l'inverse de poster des annonces, il permet à un service RH de rechercher des candidats sur Internet, en utilisant de nombreuses sources, telles que des CVthèques ou des réseaux sociaux (LinkedIn).



**AD Recruter** : Boostez votre campagne de recrutement auprès d'une audience ciblée sur Google.

Ad Recruter est un outil qui utilise les logiciels AdSense et AdWords de Google. Il permet aux clients de faire la promotion de leurs offres d'emploi par le biais des annonces Google, ainsi que d'analyser la performance de cette campagne.



## Présentation des équipes

L'équipe Multiposting est divisée en plusieurs équipes :



- Les **Devs** : les développeurs expérimentés qui composent cette équipe se consacrent au développement innovant. Ce sont eux qui, par exemple, ont réalisé le passage à la V3 (refonte du logiciel Multiposting), s'occupent des logiciels Multisearch et AdRecruiter, et mettent au point de nombreuses applications internes à l'entreprise et gèrent les outils de travail. Ils établissent la grande base informatique de tous les produits de l'entreprise.
- Les **Dev-Prod** : c'est l'équipe dont les autres stagiaires et moi faisons partie. A l'inverse des Devs qui se concentrent sur le développement pur, notre équipe était chargée du « développement production ». Il ne s'agissait pas nécessairement d'innover, mais surtout de développer les programmes nécessaires à l'intégration des nouveaux jobboards (sites d'emploi) et écoles à l'outil Multiposting, ainsi qu'à la résolution d'imports et autres tâches support spécifiques à certains clients.
- Les **Graphistes** : ils se chargent de l'aspect visuel et de l'ergonomie des outils fournis aux clients.
- Le **Support Client** : ils guident les clients de Multiposting dans leur utilisation de l'outil, répondent à leurs besoins et sont entièrement disponibles pour les nombreuses demandes des clients. Ils travaillent en étroite collaboration avec les Dev-Prod (les deux open-spaces sont adjacents), assurant le lien entre les exigences des clients et les solutions proposées par les développeurs.
- Les **Commerciaux** : leur objectif est de démarcher de nouveaux clients pour aboutir sur des contrats.
- Les **Biz-Dev** : cette équipe s'occupe de tâches diverses et variées concernant à la fois les pôles développeur, commercial et client. Ils constituent une sorte de plaque tournante entre les Commerciaux et les Devs et Dev-Prod, d'où leur nom.

## Stage

### Langues, outils, technologies et méthodologies

#### Langues informatiques

##### PHP



PHP est un langage de scripts libre généralement utilisé pour mettre en place des pages Web dynamiques par un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage de façon locale, en exécutant les programmes en ligne de commande. PHP est un langage disposant depuis la version 5 des fonctionnalités de modèles objets complets. En raison de la richesse de sa bibliothèque, on désigne parfois PHP comme une plate-forme plus qu'un simple langage.



## Python



Le Python est un langage de script de script interprété comme le php. Aujourd'hui le python est très apprécié en raison de sa syntaxe plus aérée donc plus facile à lire et de ce fait il est de plus en plus utilisé.

De plus c'est un langage objet et multi-plateforme.

J'avais déjà entendu parler de ce langage et pendant mon stage, on utilise python pour les nouvelles intégration ou les créations des nouveaux outils car on veut migrer au fur et à mesure des codes vers python, du coup la compréhension et l'utilisation de ce langage a été primordiale pour ce stage.

## SQL

SQL (Structured Query Language) est un langage informatique d'interrogation de bases de données. La partie langage de manipulation de données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données.

## Framework

Un framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un squelette de programme. Il est souvent fourni sous la forme d'une bibliothèque logicielle, et accompagné du plan de l'architecture cible du framework.

## Django



Django est un framework open-source de développement web en Python. Il a pour but de rendre le développement web 2.0<sup>2</sup> simple et rapide. Pour cette raison, le projet a pour slogan « Le framework web pour les perfectionnistes sous pression ». Développé au départ pour les sites de la compagnie de Lawrence, Django a été publié sous licence BSD à partir de juillet 2005.

Django est un framework qui s'inspire du principe MVC<sup>3</sup> ou MTV (la vue est gérée par un template) composé de 3 parties distinctes :

Un langage de templates flexible qui permet de générer du HTML, XML ou tout autre format texte ;

Un contrôleur fourni sous la forme d'un "remapping" d'URL à base d'expressions rationnelles ;

---

<sup>2</sup> Le Web 2.0 est l'évolution du Web vers plus de simplicité (ne nécessitant pas de connaissances techniques ni informatiques pour les utilisateurs) et d'interactivité (permettant à chacun, de façon individuelle ou collective, de contribuer, d'échanger et de collaborer sous différentes formes).

<sup>3</sup> MVC est abréviation de mot Modèle-Vue-Contrôleur, il est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective.

Une API d'accès aux données est automatiquement générée par le framework compatible CRUD. Inutile d'écrire des requêtes SQL associées à des formulaires, les requêtes SQL sont générées automatiquement par l'ORM(mapping objet-relationnel).

Site Web : <https://www.djangoproject.com/>

## Symfony



Symfony est un framework MVC libre écrit en PHP 5. En tant que framework, il facilite et accélère le développement de sites et d'applications Internet et Intranet.

Site Web : <http://symfony.com/>

## Méthodologie

### Gestion de version (Github)

La gestion de versions consiste à maintenir l'ensemble des versions d'un ou plusieurs fichiers (généralement en texte). Essentiellement utilisée dans le domaine de la création de logiciels, elle concerne surtout la gestion des codes source.



GitHub est un service web d'hébergement et de gestion de développement de logiciels, utilisant le programme Git. Ce site est développé en Ruby on Rails et Erlang par Chris Wanstrath, PJ Hyett et Tom Preston-Werner. GitHub propose des comptes professionnels payants, ainsi que des comptes gratuits pour les projets de logiciels libres.

Multiposting a choisi l'outil Git comme **logiciel de gestion de versions**, couplé avec le service d'hébergement GitHub. Tous les développeurs disposent de leur compte sur GitHub, et sont autorisés à contribuer aux projets de Multiposting. Afin de communiquer avec GitHub, ils utilisent Git.

### Répertoires

Pour les raisons de sécurité, Multiposting dispose des répertoires privés sur ce service, et y stocke tout son code source, chaque répertoire contient les code d'un projet de l'entreprise, par exemple, les code source du projet «Générateurs» sont stocké dans le répertoire Multiposting/Multiposting-Generators. Dans la figure 1, elle affiche tous les répertoires privés du Multiposting.

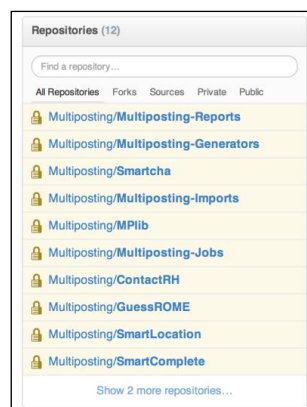


Figure 1 les répertoires sur GitHub

### Branche :

Chaque projet dispose de sa branche principale, « master » par exemple, et pour les projet important , il dispose une branche « production », regroupant tout le code qui est actuellement en production. Pour la modification du code, chaque développeur va créer ses propres branches, divergeant de la branche principale, afin de réaliser ses tâches sans risquer d'endommager le travail existant. Par exemple, le développeur crée une branche depuis la branche principale « master », il travaille sur cette branche, fait des **commit** pour sauvegarder des modifications en local, pour certains répertoires, on a un script qui appel « pré-commit », il se lance quand le développeur fait un commit, et il examine les syntaxe en utilisant des script comme pep8 pour les code python et php -l pour les code en PHP. et quand il finit sa travail sur cette branche, il fait un **git push**, afin que les modification sont indexé dans le serveur hébergement de Github.

### Pull Request :

Pour demander une merge de code de sa propre branche ver la branche principale, le développeur ouvert un **pull request(PR)** sur le site Github. Cela signifie que le développeur considère que sa modification, ou son intégration, est prête à être testée. Il signale donc qu'il souhaite fusionner sa branche (et toutes les modifications effectuées dessus) à la branche principale, afin de porter son travail en production.

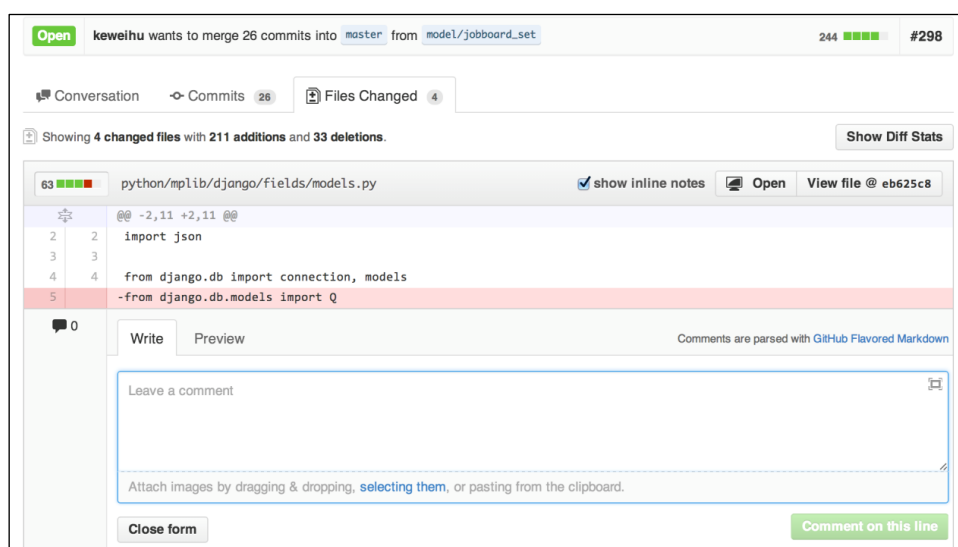


Figure 2 un pull request sur GitHub

## Review du code

Quand un **PR** a été créé, le développeur doit renvoyer ce PR au quelqu'un d'autre (plus souvent c'est le responsable), alors le PR va être examinée par les examinateur, qui les valident ou les refusent, en écrivant éventuellement des commentaires sur la page de visualisation des différences dans le code source(voir figure 2).

Site Web : <https://github.com/>

## Gestion de tâche (JIRA)



JIRA est un système de suivi de bugs, un système de gestion des incidents, et un système de gestion de projets développé par Atlassian Software Systems. A chaque fois qu'un problème est constaté une tâche est créée et elle sera ensuite traitée dans les plus brefs délais.

JIRA est un des outils principaux utilisés par l'ensemble de l'entreprise. Tous les travaux sont distribués par des tâches créées dans le système JIRA.

Chaque employé dispose de son propre compte sur l'espace JIRA, à partir duquel il peut créer de nouvelles tâches, consulter celles qui lui sont assignées, les modifier, les faire évoluer... Lorsqu'un utilisateur crée une tâche, une page entière consacrée à la tâche est créée, contenant bon nombre d'informations utiles(voir figure 3).

The screenshot shows a JIRA issue page. At the top, it says 'Multiposting Generators / MPGEN-8309' and 'Topinge.com / Changement des valeurs du champ fonction'. Below this are buttons for 'Edit', 'Comment', 'Assign', 'More', 'To Backlog', 'To Test', and 'Workflow'. The 'Details' section on the left lists: Type: Story, Priority: Major, Affects Version/s: None, Component/s: None, Labels: None, Salesforce.com Case Reference: 500200000023W8RAAV - Topinge.com / Changement des valeurs du champ fonction, and Jobboard ID: 3167. The 'Status' is 'READY FOR DEVELOPMENT' (highlighted in yellow) with a '(View Workflow)' link. Other fields include Resolution: Unresolved and Fix Version/s: None. The 'People' section on the right shows Assignee: Kewei Hu, Reporter: Antoine Labourier, and 3 watchers. The 'Dates' section shows Due: 10/Jan/14, Created: 11/Dec/13 12:52 PM, and Updated: a few seconds ago. The 'Description' section contains the text: 'Bonjour, • Jobboard : Topinge (3167) Nous utilisons la liste de fonction de Canaljob pour Topinge et cela n'est pas bon. Voici la liste des vraies valeurs fonction pour Topinge : Pouvez-vous les intégrer ?'. There is also an 'Agile' section with a 'View on Board' link.

Figure 3 la description d'une tâche JIRA

En effet, dès la création et jusqu'à son accomplissement, une tâche sera toujours assignée à un employé (sauf si elle est en « Unassigned », mais c'est un cas particulier rare), qui devra l'examiner et la traiter au plus vite. Une fois qu'il aura terminé son travail dessus, il devra réassigner cette tâche à la personne suivante dans le cycle de développement. Dans quasiment tous les cas, il devra à la fois changer l'état de la tâche.

L'état d'une tâche est ce qui définit sa progression, son avancée. Il permet de savoir à quelle étape se situe l'équipe par rapport à la résolution du problème.

Pour permettre d'évaluer le Workload d'une tâche, sur le système JIRA, on enregistre le temps utilisé de chaque étape, par exemple, dans la figure 4, on enregistre 30 minutes de travail sur une

tâche, et dans le colonne droit (Time tracking), il affiche les temps total estimé pour résoudre cette tâche, le temps restant et le temps déjà passé sur la tâche, Cela est utile surtout pour les développeurs, qui peuvent alors savoir ce qui a été fait et ce qu'il reste à faire et la quantité de travail effectué sur chaque tâche, mais aussi pour le Support Client, qui peut alors tenir d'éventuels clients informés.

Figure 4 enregistre de travail sur JIRA

## Workflow<sup>4</sup>

Une tâche peut être sous de nombreux états. Les principaux sont : Ready for development (prête à être traitée), In Progress (en cours), Need Information(demande d'information), Waiting Review(attend d'examen), Testing (en test), Resolved (résolue en attente de fermeture), Closed (fermée). Grâce à eux, une tâche dispose alors d'un véritable cycle de vie qui lui est propre, que l'on appelle le workflow.

Pour les tâche du projet générateur, quand la tâche est créé, elle est dans le statut de « backlog », et le manager du projet va décider quand et à qui attribuer la tâche.

Quand le manager décidé de attribuer la tâche, la tâche est en statut « selected for development » et la tâche est bien assigné à un développeur.

Maintenant, la tâche est chez le développeur, mais peut-être que la description de la tâche ne est pas précise, du coup, pour demander des information, le développeur peut changer le statut de la tâche ver « Wating information » et l'assigner la tâche à un personne qui peut répondre des questions. Et après la personne a complété les information sur la tâche, il va réassigner la tâche au développeur.

Quand tous les information sont prêts, le développeur peut commencer la tâche en passant la statut de la tâche ver « In progress ». Dans la figure 5, j'essaye de changer le statuts d'une tâche de « In progress » ver « Awaiting information ».

<sup>4</sup> Un workflow (anglicisme signifiant flux de travail) est la représentation d'une suite de tâches ou opérations effectuées par une personne, un groupe de personnes, un organisme, etc.

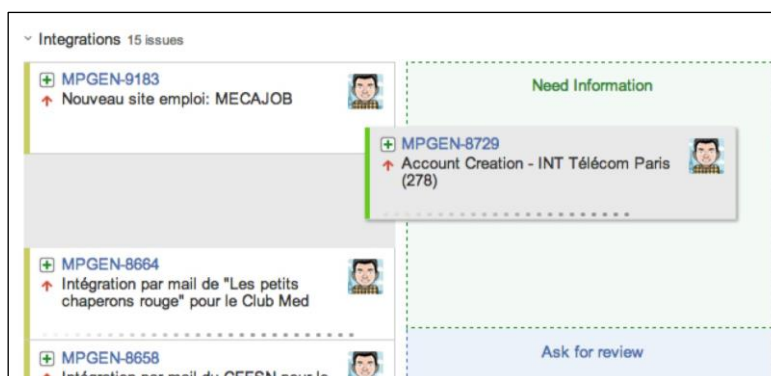


Figure 5 le changement de statut d'une tâche sur JIRA

Quand le développement est fini et si la tâche a besoin un changement sur le code, ou il faut exécuter des requête SQL, des scripts sur le serveur de production, le développeur propose toute la demande dans la tâche au manager en passant la tâche ver « Review », et assigner la tâche au manager, et le manager va décider de effectuer le solution proposé par le développeur ou refuser cette solution. Dans le premier cas, le manager passe la tâche ver le statut « Testing » et l'assigner au développeur, Ou bien pour la deuxième cas, le manager ajoute des commentaire sur la tâche en expliquant les problème sur la solution, et réassigne la tâche au développeur avec le statut « In progress ».

Pour les tâche en statut « Testing », le développeur va tester le fonctionnement de la modification sur le serveur de production, si tout va bien, il peut résoudre la tâche en respectant certains critère selon la type de la tâche, ou s'il a trouvé des problème, il repasse la tâche ver « In progress » pour corriger des problèmes. Voici, dans la figure 6, elle présente un workflow le plus souvent utilisé du projet générateur. Dans l'annexe, vous pouvez trouver la version complet de workflow du projet.

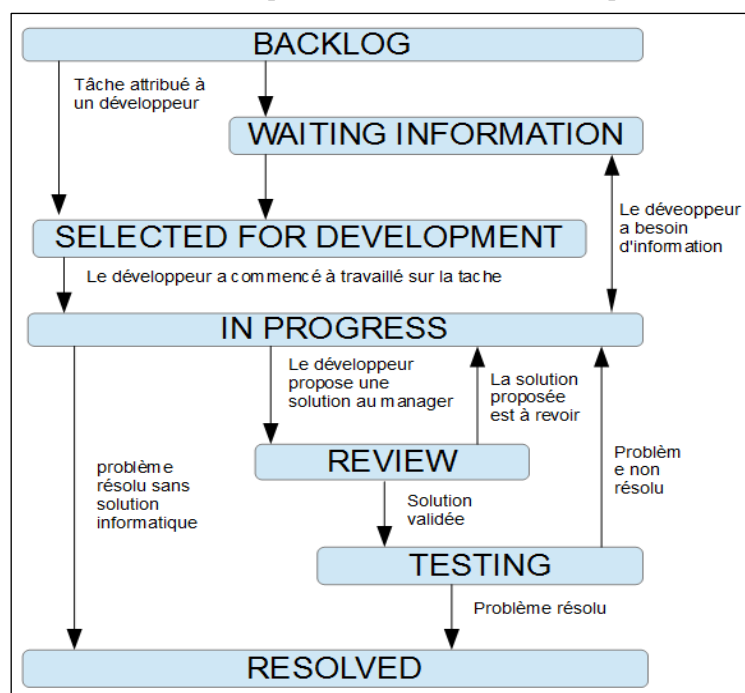


Figure 6 le workflow du projet générateur

## Dashboard

Une autre interface utile de JIRA est celle du Dashboard. Elle permet de visualiser un ensemble de tâches, sous différents critères et filtres choisis, sous la forme d'un tableau. L'utilisation la plus répandue de ce Dashboard est une vue permettant à un employé de voir toutes les tâches qui lui sont assignées, classées par catégorie (Intégration, maintenance, support...), état et priorité. Cette fonctionnalité très utile permet à chacun de rapidement faire le point sur son propre travail.

Un des caractères le plus utile est que on peut définir des filtres de tâche dans le Dashboard, les filtres nous permet de filtrer des tâche qui nous n'intéresse pas et afficher les tâches qui sont attribué chez nous même. Par exemple, dans la figure 7, on a ajouté 4 filtre : les tâches attribué à moi, les tâches qui sont en retard, les tâches qui n'ont pas de échéance et les tâche qui me concerne. Grace aux filtres du Dashboard, on peut travailler plus efficace sans risque de perdre des information importante.

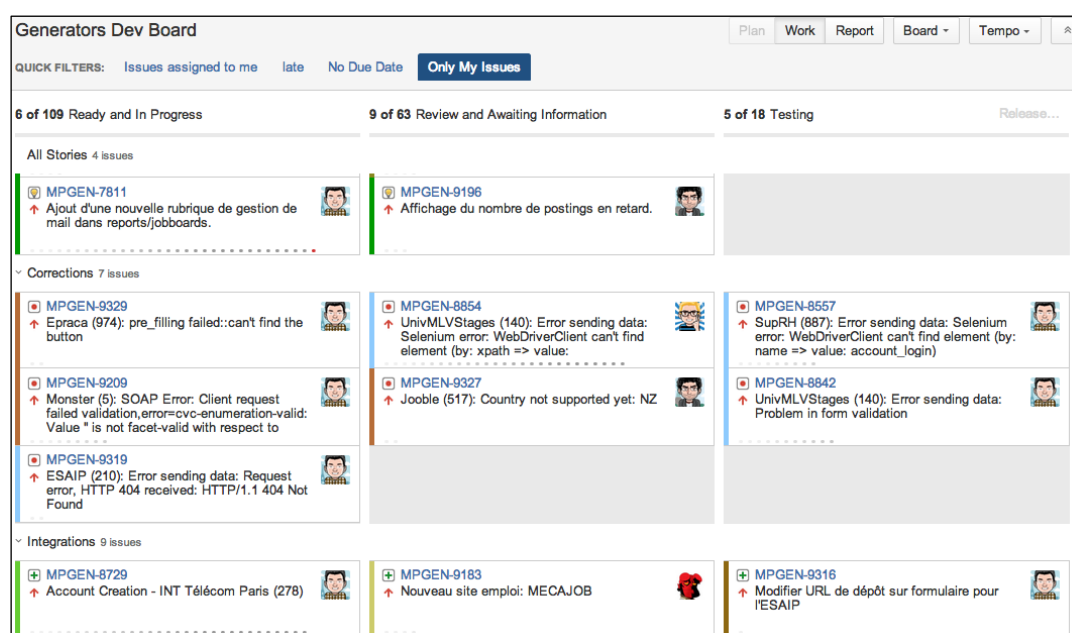


Figure 7 le DashBoard en appliquant le filtre pour afficher tous mes tâches

Enfin, JIRA dispose de nombreuses autres interfaces et options qui peuvent être très utiles : notification lors d'un commentaire, création de sous-tâches ou de liens entre les tâches (exemple : une tâche en bloque une autre), deadline (due date), statistiques...

Site Web : <https://www.atlassian.com/fr/software/jira/>

## Confluence



Au fur et à mesure de la réalisation des missions et de l'intégration de nouveaux jobboards (voir la figure 8) ou de nouvelles écoles, il arrivait très souvent que les développeurs rencontrent des cas particuliers. Certains générateurs, par exemple, étaient programmés ou utilisés d'une façon unique. D'autres fois, ils rencontraient un nouveau Webservice dont il fallait assimiler l'utilisation.

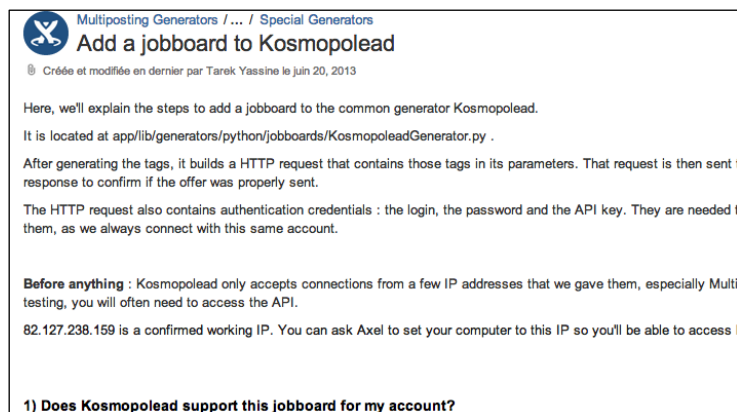


Figure 8 une page de confluence pour intégrer des nouveaux jobboards

De plus, il leur était fréquent de développer de nouveaux scripts et outils internes visant à accélérer et faciliter le travail de l'équipe à l'avenir.

Enfin, l'arrivée de stagiaires et de nouveaux employés étant très fréquente, il faut trouver un moyen de les former au maniement des outils de base, des scripts particuliers, mais surtout aux conventions de traitement des tâches (le process).

Pour ces raisons, les développeurs de Multiposting utilisent un outil nommé **Confluence**. il est un gestionnaire de contenu, un wiki qui contient des tutoriels sur les tâches récurrentes, les rapports d'erreurs, de la documentation, et toutes informations que l'on souhaite partagé au sein de l'entreprise. Et ce outils est également édité par la société Atlassian.

Cet outil est très utile, lorsque nous rencontrons un jobboard particulier ou nous développons un script visant à servir d'outil interne, nous étions très souvent encouragés à contribuer à la documentation en écrivant nous-même des articles sur Confluence.

La documentation de l'entreprise est vraiment important pour le intégration continue, grâce au Confluence, on peut facilement créer, modifier et référencier les documentation, ça nous permet de partager des connaissance, noter des memo, etc. les documentations sont fortement liés avec le système JIRA, on peut les trouver rapidement par différentes critères, par exemples, les tâches associé, les mots clés, etc.

SiteWeb : <https://www.atlassian.com/fr/software/confluence>

## Intégration continue (Jenkins)

L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée.

Pour appliquer cette technique, il faut d'abord que :

1. Le code source soit partagé (en utilisant des logiciels de gestion de versions tels que CVS, Subversion, git, etc).
2. Les développeurs intègrent (commit) quotidiennement (au moins) leurs modifications.
3. Des tests d'intégration soient développés pour valider l'application.

Ensuite, il faut un outil d'intégration continue tel que CruiseControl ou Jenkins (anciennement Hudson) par exemple.



En cas de Multiposting, il utilise Jenkins comme l'outil d'intégration continue. Et comme j'ai déjà présenté, il utilise Git comme le logiciel de gestion de version et les développeur fassent des commit quotidiennement. Du coup, Multiposting a bien appliqué l'intégration continue.



## Jenkins

Jenkins est écrit en Java, il fonctionne dans un conteneur de servlets tel qu'Apache Tomcat, ou en mode autonome avec son propre serveur Web embarqué.

L'avantage pour utilisation de « Jenkins » est divers, Jenkins peut compiler les codes en continue, lancer automatiquement les tests unit ou fonctionnel, vérifier les coding rule(règles de codage) et lancer des analyse de code et générer des documentations.

Dans le cas de Multiposting, il utilise souvent Jenkins pour faire des tests sur les items ci-dessous :

**Couverture de code** : c'est une mesure utilisée en génie logiciel pour décrire le taux de code source testé d'un programme. Ceci permet de mesurer la qualité des tests effectués. Pour évaluer le taux de la couverture de code, on a besoin de la test unitaire ou fonctionnel.

**Test unitaire** : c'est une procédure permettant de vérifier le bon fonctionnement d'une partie précise (par exemple, une fonction d'une classe) d'un logiciel ou d'une portion d'un programme. C'est une très bonne habitude de créer des test unitaire pour des fonctions importantes.

**PEP8** : PEP est la abréviation de « Les propositions d'amélioration de Python » (Python Enhancement Proposals), Pep - 8 est le huitième article dans le indexé, il propose un guide pour le style de code python (Style Guide for Python Code). Par exemple la règle pour le nomination des variables, l'indentation de chaque ligne, etc. Le but est de créer un convention de code pour que tout le monde peuvent lire le code facilement, sans ambiguïté.

**Pylint** : Pylint est un logiciel de vérification de code source et de la qualité du code pour le langage de programmation Python. Il utilise les recommandations officielles de style de la PEP 8. Et il peut aussi vérifier des items comme le nombre de variable dans une méthodes, les variable déclaré mais jamais référencé dans la fonction, les objets référencé mais pas encore importé, etc. le Pylint est pour trouver des problèmes potentielles et pour améliorer la qualité du code.

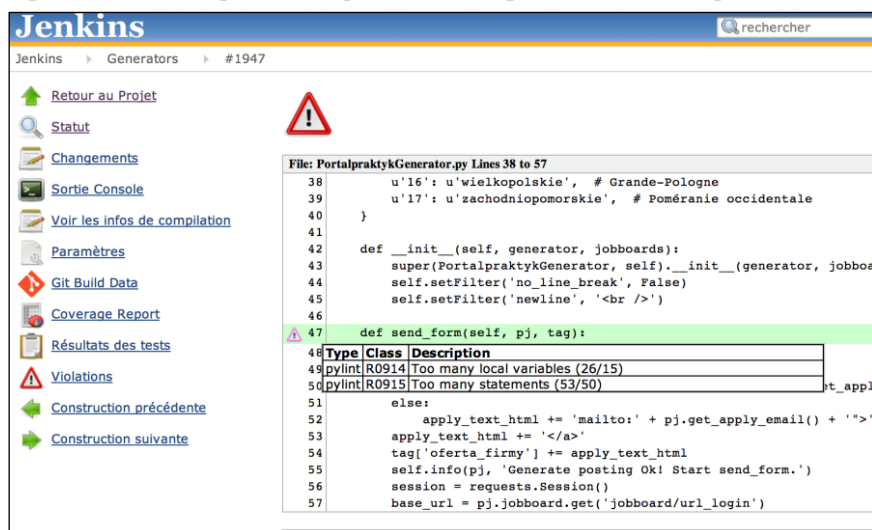


Figure 9 la page détaillé sur la partie Pylint du test Jenkins

Dans la domaine de génie logiciel, les tests sont très important pour les développement, du coup, Multiposting déploie le système Jenkins pour améliorer au fur et à mesure les codes des projets. Et les test Jenkins sont déclenché par un mécanisme nommée « build triggers » :

1. Quand une modification est poussé dans GitHub
2. Quand un « pull request » est ouvert.

Dans les deux cas, un « builder » sera lancé, il va créer un « virtualenv » et installer des dépendances, et après, il va lancer des tests programmés.

Après le lancement, Jenkins va archiver les résultat dans un répertoire spécifique et les afficher dans une page sur le site interne de Jenkins, Jenkins aussi supporte des API des autres site, par exemple, il va transmettre un state au GitHub pour dire que si le teste Jenkins est bien passée ou pas, ici, dans le page de « pull request »de GitHub, il affiche la résultat pour que l'examineur peut se trouver facilement la qualité du code.

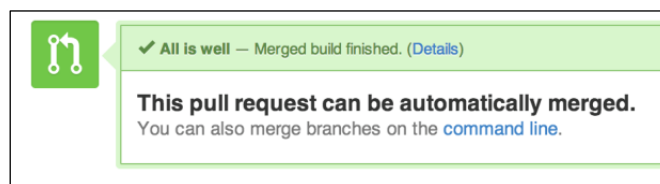


Figure 10 l'rèussi de la test Jenkins

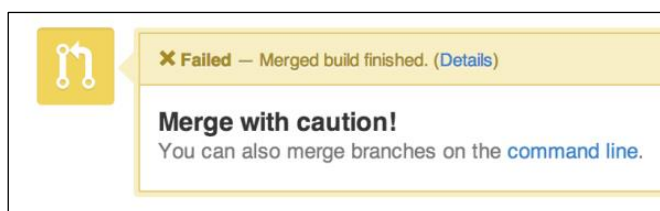


Figure 11 l'échec sur le test Jenkins

Sur le site de Dashboard Jenkins, le développeur peut voir les détails de résultat afin de améliorer les code ou trouver des problème.

Grace à la technologie de « virtuelenv », on peut lancer les test des Jenkins sur l'espace de développement, celui nous permet de faire des test unitaire en Dev.

## Outils

### MySQL



MySQL est un système de gestion de base de données (SGBD). Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle et Microsoft SQL Server. Il est utilisé pour gérer le contenu des pages web entre autre.

### MongoDB



MongoDB est un système de gestion de base de données orientée documents,

répartissable sur un nombre quelconque d'ordinateurs, efficace pour les requêtes simples, et ne nécessitant pas de schéma prédéfini des données.

L'avantage de l'utilisation de MongoDB est que le mongoDB est plus performant que les RDBMS<sup>5</sup>, et MongoDB est orienté document, du coup, MongoDB est une bonne solution pour stocker un volume important de données non structuré et manipuler un volume de données important malgré une forte charge.

Mais car MongoDB est une base de données orientée document au lieu de la base de données relationnelle, on perd la relation entre les données.

## Google Doc



Google Documents est une suite bureautique web gratuite développée par Google. C'est une suite des évolutions de Google Spreadsheets, version déjà mise en ligne auparavant et de Writely, logiciel de traitement de texte. Ces programmes fusionnés permettent un travail en ligne et collaboratif et seraient l'un des nouveaux fers de lance des applications du Web 2.0.

Pendant le stage, on utilise Google Docs pour partager des journaux de développement, les feuilles pour importer les jobboards, les résultats de test sur les différents produits, etc.

## Serveur Dev7 et virtualenv

Les fichiers qui contiennent les codes informatiques qui font fonctionner l'API sont sur un serveur (de production) distant que l'on ne modifie jamais directement afin de ne pas polluer le système en place. C'est pourquoi on travaille sur un autre serveur distant (dev7.multiposting.fr) qui est un "clone" du serveur de production. La machine dev7 est un serveur de test où l'on retrouve tous les éléments présents sur le serveur principal (la base de données, le Backend, le Frontend, les scripts, ...) et grâce à ce serveur de test on peut travailler sans endommager le système.

Afin de simuler l'environnement de serveur de production, on utilise l'outil « virtualenv » pour créer des environnements Python isolés. Il crée un environnement qui dispose de ses propres répertoires d'installation, qui ne partagent pas les bibliothèques avec d'autres environnements de « virtualenv » (et éventuellement ne pas accéder aux bibliothèques globales).

Dans mes projets, j'ai participé, on a trois virtualenv différents : app, report, jenkins.

Le virtualenv « app » est le plus souvent utilisé, il crée un environnement qui est identique à celui de la serveur de production de Multiposting sur le projet Générateur et le projet de MPJobs<sup>6</sup>. Dans ce virtuel environnement, on peut exécuter des scripts de générateur, créer des champs ou des mappings sur le Backend pour vérifier le bon fonctionnement des codes sur les projets avant de les mettre dans le serveur de production.

<sup>5</sup> RDBMS : Relational DataBase Management System : Base de données relationnelle

<sup>6</sup> Un Projet de MultiPosting, il contient la création, maintenance et amélioration du Frontend et Backend de la produit Multiposting.

## ContactPool

En plus de multi diffuser les offres d'emploi de ses clients, Multiposting redirige tous les mails en provenance des jobboards vers une seule et même adresse, propre au compte client, se terminant en `@{domain_client}.contactrh.com`. Elle permet alors de regrouper tous les mails de postulation des candidats, afin de les transmettre aux clients.

Cependant, ces adresses ne sont pas à l'abri des spams. De plus, elles reçoivent beaucoup de mails de confirmation d'inscription, de vérification de mail, etc...

Afin de trier les mails, les développeurs ont mis au point un outil interne, qu'ils ont baptisé Contactpool. Ce programme effectue un tri préalable de tous les mails en plusieurs catégories (CV, spam, confirmations...), en utilisant un système de machine-learning. Au fur et à mesure que les utilisateurs « corrigent le tir » en retriand manuellement les mails, Contactpool apprend à distinguer les demandes de candidature des spams et autres mails impertinents. Cet outil devient donc de plus en plus fiable avec le temps.

Afin d'entraîner le programme, et surtout de s'assurer que tout était bien trié, nous devions trier les mails manuellement chaque jour. Ainsi, chaque midi, l'équipe des Biz-Devs les triaient, et les Dev-Prod prenaient le relais le soir.



Figure 12 l'interface pour trier des mails

## Concept principale de multiposting

### Frontend

Le Frontend est l'interface qui se présente directement sur le client, il contient des rubriques pour déposer, modifier ou supprimer des offres, consulter les statistique du postings, ou bien gérer des information de compte client, actuellement, Multiposting propose deux version de front end : v2 qui est créé par framework Symfony en PHP et V3 qui est créé par Django en Python. La figure 13 présente la page d'accueil du frontend V3.

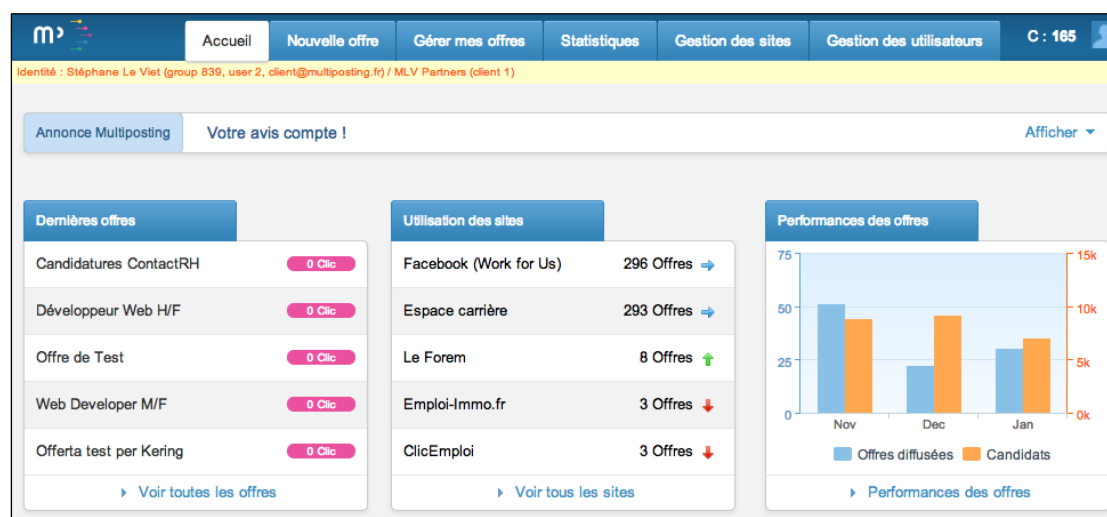


Figure 13 Espace client du Multiposting

Le Frontend de Multiposting permet les client de diffuser des annonce d'offre sur un grand nombre de jobboard et des écoles, gérer des offres qui sont déjà diffusé, analyser des performance de chaque offre sur chaque jobboard.

## Backend

Le Backend du Multiposting est un espace non seulement pour les support ou Biz-Dev mais aussi pour les développeur. Backend est l'outil visuel qui vient compléter l'imposante base de données. Il dispose de nombreuses interfaces qui permettent de consulter ou modifier des statuts du postings, ajouter ou changer des config sur un client, ajouter ou modifier des valeur du champs, gérer des jobboard ou générateur, etc. Dans la figure 14, elle affiche des information (statuts, erreur, config, etc.) du posting sur les jobboards, s'il y a des erreurs sur le posting, le champs *errors* va tourner en rouge.

AFU										[state]	[errors]	[config]					
Atmosphères - tbin@multiposting.fr										[state]	[errors]	[config]					
3265271/	5086934,	174	9389	2013-11-04	STAG	Stage Assistant Chef de Projet Marketing	Stage	add	generator: fatal								
Coles SA - Quentin										[state]	[errors]	[config]					
3261350/	5036584,	174	11593	2013-10-31	04645	TPE/PFE Evaluation des gains et des risques de l'utilisation d'une porteuse GPS H/F	Stage	add	generator: fatal								
Agrojob										[state]	[errors]	[config]					
Danone - Arzum										[state]	[errors]	[config]					
3264676/	5077661,	1549	2535	2013-11-03	35354	Stage Exploitant transport H/F	Stage	active									
EmbaucheHandicap.fr										[state]	[errors]	[config]					
Kantar - Arzum										[state]	[errors]	[config]					
3257032/	4990667,	1707	11625	2013-10-29	STAGECO	Assistant chargé de communication interne (H/F)	Stage	active									

Figure 14 l'interface de gestion de posting dans le Backend

## Base de donnée

Comme on utilise deux Framework différents pour le Frontend et Backend, et au début Multiposting utilise MYSQL et après il ajoute MongoDB et commencer de migrer les donnée depuis MYSQL ver MongoDB, il y a un mécanisme de synchronisation entre les deux base de donnée.

Pour Multiposting, pour certains tableaux, les données ne sont pas très corrélé, mais le volume de donnée est énormément (des millions des offres et des millier de jobboards dans la base de donnée), du coup, le MongoDB est une bonne choix car c'est plus facile d'ajouter des espaces pour les instances actuelles et le cherche de la base de donnée est plus rapide.

Pour les produits de Multiposting, il contient plus de 220 tableau et une quantité impressionnante d'informations. Elle a de nombreux rôles : conserver tous les informations sur les offres renseigné par le client , conserver les informations sur les clients et les jobboards, enregistrer les valeur et label du champs, les mappings, les information sur les générateurs, les logs, etc.

Pour les requetés spéciaux qui ne propose pas sur le Backend, on utilise souvent des requêtes SQL pour insérer ou sélectionner des information dans la base de donnée, Comme les données contenues dans cette base sont très sensibles et peuvent causer un grave dysfonctionnement du produit à la moindre mauvaise manipulation, elle n'est accessible en écriture qu'aux responsables d'équipes. Elle l'est en lecture pour tous les autres développeurs, qui disposent également d'une copie individuelle de cette base, sur laquelle ils peuvent effectuer tous leurs tests. On l'appelle « la base de Dev», pour la distinguer de « la base de production ».

## Comment ça marche pour la multidiffusion

### Concept

Le concept pour multi-diffuser une offre est qu'il permet le client de choisir autant de jobboards et écoles il veut diffuser comme dentinaire et remplir une seule formulaire, et après une clique, l'offre sera diffuser automatiquement sur tous les jobboards et écoles il a choisi.

### Offre

Un objet **offer** est créé dans la base de données quand un client dépose un offre sur le Frontend. Il est lié au client et lui permettra de revenir modifier son annonce, ou consulter ses statistiques.

### Postings

Un posting, en un mot, présente une action de diffusion sur les jobboards, Un objet **posting** est créé et lié au offre dans les situations suivants :

- Quand l'objet offre est créé.

- Quand le client ajoute des nouveau jobboard dans la liste de diffusion d'offre.
- Quand le client reposte l'offre qui est expiré.

L'avantage de cette stratégie est qu'il nous permet de séparer les postings avec des différents date de expiration et la durée de publication.

Au niveau technique, La table *posting* contient des infos général au posting (identifiant, date de création ...).

La table *posting\_values*, elle contient les valeurs des champs communs (titre, description, référence, ...) du posting et peu importe sur quel jobboard on envoie l'offre ces informations sont généralement présentes.

## Jobboard

Un jobboard peut être un site d'emploi comme le pôle emploi, APEC, ou bien un site carrier d'entreprise ou un site d'école permettant l'entreprise à déposer les offres.

L'objet **jobboard** contient des information essentiel pour jobboard(nom, type, etc.)

## PostingJobboard

Pour un posting et pour chaque jobboard ou site d'école que le client a coché, un objet *posting\_jobboard* est créé, liant un posting à un jobboard. Par exemple, si le client choisit de diffuser sur 10 sites pour une offre, 10 nouvelles entrées apparaîtront dans la tableau *posting\_jobboard*, associant l'id du posting à l'id du jobboard.

Au niveau technique, La table *posting\_jobboard* contient les infomations nécessaires pour que le système puisse identifier les jobboards dont le posting est destiné.

La table *posting\_jobboard\_values*, elle contient les champs et ses valeurs du posting spécifique au jobboard choisi.

## Générateur

Le générateur est le bout de code qui va diffuser les annonces aux jobboards c'est un App de Framework Django ou Symfony, ils postent les offres aux jobboards et aux écoles, pour un jobboard, il contient au moins un générateur pour lui permet d'envoyer des offres, et pour les différent jobboard, ils peuvent utiliser le même générateur pour envoyer des offres. et généralement, un jobboard ne utilise qu'un seule générateur pour diffuser les offres. quand un générateur est se lancé, il cherche tous les postings dans le base de donnée qui est dans le statue qu'il supporte, on peut définir des statues supporté par le générateur en ajoutant un flag dans le colonne « *supported\_status* » de la tableau « *generator* ».

Il faut savoir que chaque jobboard réceptionne les offres de manières différentes. Il y a des jobboards qui proposent un formulaire à remplir, certains reçoivent (ou viennent récupéré) des flux d'offres c'est à dire des fichiers (XML ou CSV) qui contiennent tous les détails nécessaires qui décrivent les offres d'emplois à diffuser.



## Cron (planning de lancement de script)

Pendant mon stage, pour le projet générateur, on utilise successivement deux méthodes pour automatiser le lancement régulière des générateurs :

### 1. Crontab

Le Crontab est le nom du programme sous Unix (et Linux) qui permet d'éditer des tables de configuration du programme cron. Ces tables spécifient les tâches à exécuter et leur horaire d'exécution avec possibilité d'horaire périodique. Dans la figure 15, elle présente le planning de lancement de générateur sur le serveur Prod11.

```

1 #####
2 ## Remove generators locks older than 23 hours on prod servers.
3 */15 * * * * @prod11 find $MP_APP/temp/generators/ -mmin +1380 -name '*.lock' -delete
4 */15 * * * * @prod12 find $MP_APP/temp/generators/ -mmin +1380 -name '*.lock' -delete
5 */15 * * * * @prod13 find $MP_APP/temp/generators/ -mmin +1380 -name '*.lock' -delete
6
7 #####
8 00 09 * * mon-fri @prod11 source /home/multiposting/Envs/virtualenv_reports_app/bin/activate &
9 00 10-20 * * mon-fri @prod11 source /home/multiposting/Envs/virtualenv_reports_app/bin/activat
10
11 ## Générateurs - Diffusion immédiate
12
13 0-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4713 # FED Africa
14 2-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4714 # FED Human
15 4-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4715 # FED Finance
16 6-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4716 # FED Supply
17 8-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4717 # FED Office
18 10-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4718 # FED Legal
19 12-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 4719 # FED Business
20 14-59/15 * * * * @prod11 $MP_APP/bin/run-script generate -s -c -j 6476 # FED Construction

```

Figure 15 le une partie de Crontab sur le projet Générateur

Pour chaque générateur du jobboard, on ajoute le script d'exécution dans le Crontab et indique les horaires de lancement.

L'avantage de Crontab est qu'il facile et rapide, pour automatiser un générateur, on n'a besoin que d'ajouter une ligne dans le Crontab, il n'y a pas de modification sur la base de donnée.

### 2. Celery

Celery est un projet open source queue de la tâche, Elle se concentre sur opération en temps réel, mais aussi prend en charge la planification.

Dans le projet générateur, on a créé un tableau dans la base de donnée MongoDB pour la gestion de la lancement du générateur, dans le tableau, on sauvegarde comme le Crontab, les horaire de lancement d'un générateur pour un jobboard, et chaque ligne est comme une tâche, et dans le serveur de production, on regarde le tableau tous les 5 seconde, pour savoir s'il y a des tâche à exécuter, si oui, le Celery diffuser les tâches dans les queues des deux serveur de production, et chaque serveur contient au maximum 16 workers, et Celery choisit un worker par tâche en fonction de la charge du serveur.

Avantage de Celery est qu'on peut gérer les lancement du générateur plus raisonnable, car tous les horaire sont stocké dans la base de donne, et comme on utilise le queue pour diffuser les tâches, c'est plus stable que le Cron car s'il y a des accidents surviennent dans le serveur de production, pour le Cron, on ne peut pas rétablir les lancement en cours, il faut attendre le prochaine lancement du générateur défini dans le Cron, mais pour Celery, si une tâche est diffusé dans le queue, il reste toujours là si la tâche n'a pas été exécuté..



## Processus de multidiffusion

Il y a deux méthodes pour nous transmettre les offres qui seront multi-diffusées

La première est de passer par le Frontend, un formulaire qui prend en compte toutes les contraintes (nombre de caractères maximum, niveau d'expériences, diplômes) des jobboards sélectionnés est proposé au client.

La seconde méthode est généralement utilisée pour les clients qui ont des sites carrières. Le principe est que l'on va sur leur site, lire les informations concernant les offres d'emplois, extraire et formater les données pour en faire des postings qui seront ensuite traités comme les postings créés via le Frontend.

Ici, je vais présenter en détaillant la première méthode pour multi-diffuser les offres.

### 1. Choisissez les jobboard à multi diffuser.

C'est le premier étape pour le client, le client doit d'abord choisir les jobboards qu'il veut diffuser les offres, la liste de jobboard est affichée sur l'interface de client. Dans la figure 16, c'est la liste de jobboard(généralistes et spécialisés) disponible pour un client.

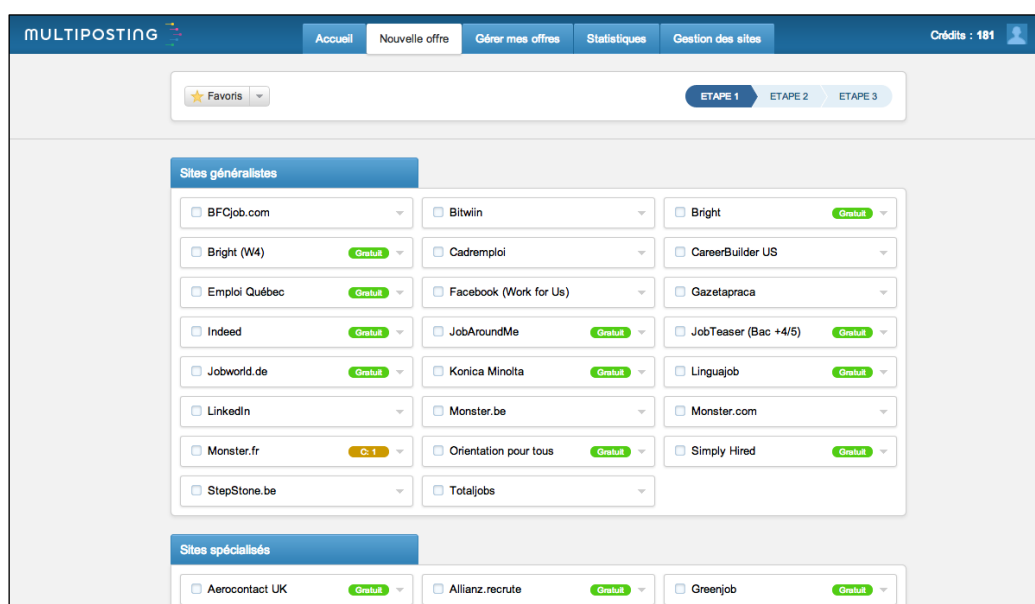


Figure 16 l'interface pour sélectionner les jobboards pour multi diffuser des offres

### 2. Création de l'annonce

Dans cette étape, le client doit transmettre toutes les informations nécessaires concernant ses annonces que nous stockerons ensuite dans la base de données avant de les traiter. Si le client a choisi plusieurs jobboards, et les jobboards contiennent les différents champs spécifiques, le client doit d'abord remplir les champs en commun de tous les jobboards une seule fois, et après les champs spécifiques de chaque jobboard. Dans la figure 17, le client a choisi de diffuser son annonce sur BeepJob et CareerBuilder, du coup, après il a rempli les champs communs, il doit remplir les champs spéciaux des deux jobboards(début de contrat, langage, etc. en fonction de jobboard il a choisi).

The screenshot shows a web form for configuring jobboard postings. It includes several dropdown menus for 'Fonction', 'Sous-fonction', 'Secteur', and 'Sous-secteur', each with a 'Sélectionner...' button. There is a text input for 'Mots clés' and radio buttons for 'Configuration à utiliser' (Groupe or Société). Below these are sections for 'Beepjob' and 'CareerBuilder', each with a 'Début de contrat' dropdown and a 'language' dropdown. The 'CareerBuilder' section also has a 'Rendre cette offre anonyme' checkbox and a 'Package' text input.

Figure 17 les champs communs et spécifiques de jobboard

Lorsqu'une annonce est créée, une nouvelle entrée est insérée dans les tables *posting*, *posting\_jobboard*, *posting\_values* et *posting\_jobboard\_values* de la base de données de Multiposting.

Une fois que les postings ont été créés, ils ne nous restent plus qu'à les diffuser vers les jobboards sélectionnés par les clients et vérifier que les offres ont bien été postées.

Pour le client, il peut désormais gérer les offres (modifier, supprimer une offre, ajouter d'autres jobboards, etc.) et consulter la performance d'offre par offres ou par jobboard sur son interface client de Multiposting.

### 3. Envoi des offres

Pour poster les offres aux jobboards on a lancé des générateurs qui seront lancés de manière automatisée et leur rôle est de récupérer les offres dans la base de données, poster les offres en respectant les contraintes du jobboard et vérifier que l'offre a bien été envoyée, si tel n'est pas le cas, un message d'erreur nous est envoyé afin que nous puissions régler le problème.

C'est le générateur qui va vérifier que les contraintes du jobboard auquel il est associé sont respectées et puis il se chargera de poster les offres et vérifier que tout a fonctionné correctement.

Quand une nouvelle offre vient d'être créée elle est dans l'état **new**, un script qui a été développé par les développeurs de Multiposting est exécuté toutes les 5 minutes pour faire passer les offres dans l'état **add**, c'est à dire prêt à être diffusé.

L'information sur l'état d'un posting est disponible sur le Backend. et généralement, le flux de la changement d'état est ci-dessous:

Pour les nouvelles offres : NEW => ADD => ACTIVE

Pour les offres modifiées par client : UPDATE => ACTIVE

Pour les offres supprimées par client : DELETE => DELETED

Pour les offres qui sont expirées : ACTIVE => EXPIRED

Pour le statut de NEW vers ADD, Multiposting utilise un script nommé *postUpdater* qui tourne non-stop pour manipuler des offres en NEW, il s'occupe par exemple d'envoyer des emails de confirmation pour certains clients, ajouter des paramètres supplémentaires pour certains jobboards et mettre les postings en ADD.

Pour les posting qui sont expiré, c'est-à-dire la date de fin de diffusion est inférieure à aujourd'hui. on utilise aussi un script non-stop qui s'occupe de exécuter les opération pour des postings qui sont expiré, par exemple, il met le statuts de posting ver DELETE afin que le générateur peut envoyer une requête de fermeture de posting au jobboard pour les posting qui sont expiré, mais plus généralement, il met la statuts de posting ver EXPIRED, afin qu'on n'a plus pris en compte des postings.

Pour les autres type de statuts, la changement de statuts sont pris en compte par le générateur, chaque fois on lance le générateur, il boucle sur tous les posting qui sont associé au générateur et avec des statuts supporté (il regarde le colonne generator, flags et error sur le table *posting\_jobboard* pour filtrer les postings), et ensuite après la exécution, on lance la méthode *postingUpdater* pour mettre à jour les statuts de posting correspondant à la opération effectuée.

Ces sont des flux de statuts le plus souvent pour Multiposting, mais on peut changer le flux dans le générateur si le workflow dans jobboard est spécifique, par exemple, on ajoute un statue ADD PENDING entre ADD et ACTIVE si les offres envoyé dans le jobboard a besoin d'être validé par jobboard et il faut recevoir un mail de validation et cliquer un lien de validation pour faire publier l'offre.

A chaque fois que le système effectue une action sur un posting, la table *posting\_jobboard* est mise à jour les attributs que l'on surveille sont « state » et « errors » car ce sont eux qui nous informe si une offre a bien été posté.

## Traitement d'erreur

Comme Multiposting supporte un grand nombre de jobboards (école, site carrière), c'est souvent que Multiposting rencontre des erreur de différente type tous les jours. Du coup, il faut établir un processus de traitement d'erreur qui soit efficace et rapide pour diffuser des offres à temps.

### 1. La création des tâche de correction

Lorsqu'on a des offres en erreurs, elles sont répertoriées par jobboards et par type d'erreurs afin de constituer un rapport quotidien.

Tous les jours, les erreurs répertoriées sont analysé et traité par un application, il crée des tâches de correction qui seront attribuées aux développeurs. L'application est connecté à la base de données et à l'outils de gestion de tâches **JIRA**. Les actions possibles sont de vérifier le nombre fois où l'offre a été relancée, retirer les erreurs pour relancer le posting et créer une tâche ou éventuellement ajouter à une tâche existante le posting en erreur. Avant de créer une nouvelle tâche l'application s'assurer que la tâche a été relancé au moins trois fois pour les erreur inconnus. En effet, parfois les erreurs ne nécessitent aucune corrections de notre part car elle peuvent être dû au fait que la connexion a été interrompu, jobboard indisponible, ou d'autre problème temporaire, etc. Il faut être sûr que le problème n'est pas lié à la connexion internet afin de pas créé de tâche inutilement.

Si les erreur persiste après on le relance trois fois, l'application va essayer de regarder s'il y a déjà des tâche similaire en cours et essayer d'ajouter des posting en erreurs dans la tâche avant de créer une nouvelle tâche.

Par contre, pour des erreurs de raison connus(problème sur Mapping, Localisation, etc.)

l'application va créer ou mettre à jour une tâche tout de suite pour éviter des relances inutiles. Dans l'interface ci-dessous (figure 18), on peut facilement regarder tous les erreurs actuels sur les générateurs (le nombre de posting en erreur par type, les tâches JIRA associé, etc.), dans cette interface, on peut aussi retirer tous les erreurs d'un générateur pour lui permettre de relancer des posting en erreur. Dans la figure suivante, elle affiche des nombre d'erreur en fonction de type (erreur sur la remplissage du champs, les postings en retard, les erreurs s'occupé par d'autres équipes, etc.) et de jobboard. Celui nous permet de savoir concrètement les erreurs au niveau du projet générateur et sur chaque jobboard et les tâches associées à chaque erreur.

MP Generators


Home

Errors Stats

Schedule List

Go to Error interface

Back to portal



Kewei Hu

Errors stats

Total	Gen	Field (Gen)	Field (Import)	Late	Other Teams
592499	0.154 %	0.113 %	0.154 %	0.506 %	0.124 %

Name	All	Errors	Fields (Gen)	Fields (Import)	Late	Jira	Available issues
Total	1739	13	417	633	649	22	
Interaction (966) - Interaction (7427)	7	1	0	0	0	0	
USGPeople (351) - Start People (3227)	6	1	1	0	0	0	
Monster (5) - Monster.fr (5)	41	1	2	38	0	0	
ManageursForm (252) - Manageurs.com (35)	2	1	1	0	0	0	
LeboncoinForm (542) - Le Bon Coin (4521)	382	1	136	15	230	0	
CareerBuilder (63) - Recrulex (33)	1	1	0	0	0	0	

Name	All	Errors	Fields (Gen)	Fields (Import)	Late	Jira	Available issues
Total	3757	33	251	281	2340	833	
UNAM (829) - UNAM (ENSAM Alumni) (6745)	41	1	0	0	0	0	
INTStages (97) - INT Télécom Paris (278)	18	1	13	0	0	0	
ASSAS (361) - Université Panthéon-Assas (3253)	4	1	0	0	0	0	
ENISEFormEmplo (924) - ENISE (4844)	18	1	0	0	17	0	
ESA (474) - ESA (Ecole Supérieure d'Assurances) (3393)	1	1	0	0	0	0	
Anakrys (627) -	78	0	0	0	0	78	MPGEN-7646

Figure 18 Interface de gestion des erreurs de Générateur

## 2. Le traitement des tâches de correction

Nous nous occupons uniquement des erreurs sur les générateurs, en plus précise, les erreurs de type unknown (non connu), validation, mapping, localisation, etc. Les autres types d'erreurs sont traités par les autres équipes, par exemple, les erreurs de type « invalid config » sont traitées par l'équipe support qui va vérifier les configurations du client et les mettre à jour si besoin.

Les erreurs attrapées sont différentes, et on a créé des pages sur Confluence (voir figure 19) pour identifier des différents types d'erreurs et la solution pour chaque type d'erreur.

**How to fix error fatals**  
Crée par Benjamin Bonny, dernière modification par Tristan Roussel le août 27, 2013

**ERROR\_FATAL:**  
For each ERROR\_FATAL there is an associated message used to precise errors.  
Here are the most known messages which should be used by generators when calling skip me

Nomenclature	Diagnostic
<b>Localisation Problem</b>	The localisation on incriminated posting-jobboard can not be matched with jobboard values through the associated generator.
<b>Salary Problem</b>	The way we transform the salary in associated generator is not compatible with what is awaiting the jobboard. It is a special case of constraints problem.
<b>Constraints Problem</b>	A jobboard constraint is not respected by posting-jobboard values after generator transformation.
<b>Config Problem</b>	Associated configuration does not allow generator to work as usual.
<b>Account Problem</b>	Associated configuration do not allow to post this posting-jobboard on the jobboard.
<b>Validation Problem</b>	Diagnostic is very uncertain. Errors can belong to one of the previous category. It's often because form has been filled with incompatibles or empty values.
<b>Mapping</b>	Field(s) cannot be associated to other fields through a mapping.

Figure 19 le page de confluence pour fixer les erreurs fatals

Dans le chapitre suivant, je vais prendre des exemples sur des tâches de correction concernant des différent erreurs classiques sur le projet générateur.

### 3. Résoudre des tâche de correction.

Les tâches de corrections une fois créé sont attribuées aux développeurs et une fois traitées, les posting sont relancés les conditions ci-dessous sont respectés :

- Les postings ne sont plus en erreur de type « attente générateur ».
- Une erreur similaire ne se reproduira plus avec la correction.
- Les postings sont actifs ou dans certains cas une erreur adapté (error\_field, error\_invalid, étate inactive etc.).

Si les condition sont bien respecté, on peut « résoudre » la tâche, et l'assigner au reporter.

## Déroulement du travail

Chaque matin, avant l'arrivée du bureau, les tâches sont sélectionnées et attribuées au développeur par le responsable du projet et à l'arrivée au bureau, l'ensemble de l'équipe se réunit pour le point du matin. Chacun dit sur quelles tâches il travaille et ce qu'il va faire dans la journée en fonction des tâches dans le système JIRA.

Après le point du matin, chacun commence ses tâches librement, mais à priori, on trie des tâche en fonction de priorité et de type, concrètement, on commence sur des tâches qui sont bloqués, et après des tâche de priorité critique et à la fin, des tâches de priorité normal. Sur le type de tâche, on résout les tâches de correction et après les tâches d'intégration. Du coup, généralement, on travaille sur les tâches de correction dans la matinée, et on commence des intégrations dans après-midi. Pendant la journée, c'est très souvent que on fait voir avec d'autre équipes, par exemple, pour demander des information sur des tâches urgent, ou bien résoudre des problèmes

ensemble

A la fin de la journée, l'ensemble de l'équipe se réunit encore une fois pour le point du soir, chacun dit ses problèmes rencontrés dans la journée, et l'ensemble d'équipe discutent pour partager ses avis s'ils ont déjà rencontré des problèmes similaires, et plus souvent, le responsable va proposer des solutions ou une alternative pour la problème.

## Mission

Pendant mon stage chez Multiposting, je travaille principalement sur deux projets : le projet Générateur et le projet Report. J'ai également travaillé sur le projet MPlib qui s'occupe la création, la maintenance et l'amélioration sur les modèle est les méthodes de base pour tous les projet de Multiposting.

## Projet Générateur

Ce projet contient les tâches liées à tout ce qui touche les générateurs : leur création, modification, maintenance, association à d'autres jobboards, modification de la planning de lancement du générateurs, etc. pendant mon stage, j'ai consommé la plupart de temps sur ce projet.

## Présentation du Générateur

### Type de générateur

En fonction de comment le jobboard réceptionne les offres on fait hériter notre nouveau générateur du générateur parent le plus approprié, dans la figure 20, il présente la hiérarchie de la héritage de générateur.

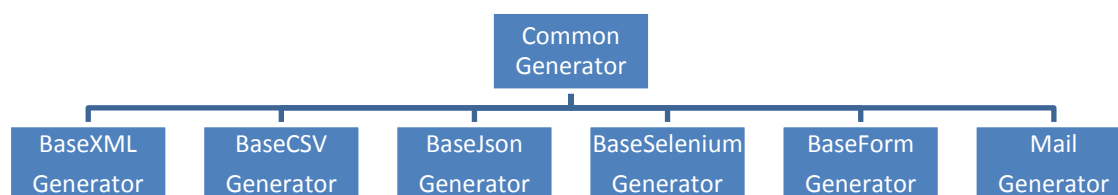


Figure 20 Hiérarchie des classes générateurs

Les générateurs héritant de ces classes peuvent aussi faire appel à des méthodes qui sont définies dans des classes "indépendantes" des générateurs grâce à l'environnement Symfony qui permet de charger automatiquement des classes sans avoir à les importer dans les fichiers. Et pour les générateurs en python, on importe des méthode ou classe depuis d'autre module pour les utiliser dans le générateur.

Grace au la méthode d'héritage, chaque générateur est les fils de la classe « CommonGenerator », du coup, on peut lancer un générateur, peu n'importe le type, en utilisant la même script, voici les

processus de la lancement du générateur.

```

GlobalStart()
  jobboardStart(jobboard)
    postingStart(posting_jobboard)
      generate(posting_jobboard)
        generateTags
        .....
      postingEnd(posting_jobboard)
    jobboardEnd(jobboard)
  globalEnd()

```

Dans le processus « *jobboardStart* », on prépare des paramètres pour chaque jobboard associé dans la générateur on est en train de lancer, par exemple, dans ce processus, on génère l'entête de flux pour chaque jobboard.

Dans le processus « *generate* », la méthode principale est la méthode « *generateTags* », elle aussi la méthode la plus utilisée pendant tous les méthodes de générateur.

La méthode « *generateTags* » est à redéfinir impérativement dans tous les générateurs. Elle renvoie un tableau associatif avec la clé du champs à compléter et sa valeur. C'est aussi dans cette méthode que l'on s'assure que toutes les contraintes du jobboard sont respectées.

Le processus « *postingEnd* » met à jour l'état de posting en fonction le flux de transaction défini dans chaque générateur. Et pour certains type de générateur, par exemple, pour le générateur de flux, il met à jour le flux en ajoutant les contenu de posting généré dans le processus « *generate* »

## Les flux (XML, JSON, CSV)

Pour faciliter notre partenaire d'importer les offres, Multiposting a proposé un service de feed (flux de données), et le partenaire peuvent lire le feed par l'url avec un token (identificateur) spécifique proposé par Multiposting, le format de l'adresse de feed est comme: [http://contactrh.com/get/\[token\]](http://contactrh.com/get/[token]), le token est une chaîne de caractère de longueur 16 générée aléatoirement pour chaque flux XML.

Quand le client fait un GET sur ce URL, on appelle la méthode « *getData* » dans le générateur, et dans cette méthode, on essaye de retourner le contenu d'un flux, qui est généré par le générateur et contient ensemble d'offre depuis le dépôt FTP de Multiposting pour le transmettre à notre partenaire.

On peut aussi transmettre le flux en fonction de client ou pays en modifiant le mode de génération de fichier de flux.

## XML

Ce type de générateur est désigné pour générer le flux XML et les envoyer par web service ou les déposer dans le FTP de Multiposting, ou bien le déposer directement sur le serveur FTP du jobboard.

Pour la génération du fichier XML, on peut créer un fichier par offre ou créer un seul fichier qui contient toutes les offres à diffuser pour le jobboard (voir figure 21), dans ce cas, on lance une seule fois le générateur de XML par jour, et on ajoute chaque offre comme un nœud fils dans la hiérarchie du fichier XML.

Un seule fichier pour tous les offres	Un offre par fichier			
<pre>&lt;offres&gt;   &lt;offre id=1001&gt;     ....   &lt;/offre&gt;   &lt; offre id=1002&gt;     ...   &lt;/offre&gt;   .... &lt;/offres&gt;</pre>	<table><tr><td><pre>&lt;offre id=1001&gt;   ..... &lt;/offre&gt;</pre></td></tr><tr><td><pre>&lt;offre id=1002&gt;   ..... &lt;/offre&gt;</pre></td></tr><tr><td><pre>.....</pre></td></tr></table>	<pre>&lt;offre id=1001&gt;   ..... &lt;/offre&gt;</pre>	<pre>&lt;offre id=1002&gt;   ..... &lt;/offre&gt;</pre>	<pre>.....</pre>
<pre>&lt;offre id=1001&gt;   ..... &lt;/offre&gt;</pre>				
<pre>&lt;offre id=1002&gt;   ..... &lt;/offre&gt;</pre>				
<pre>.....</pre>				

Figure 21 Full Feed et Incremental Feed du XML

Pour ce type de générateur, on utilise souvent la méthode « *generate\_xml\_recursive* » pour générer le flux XML rapidement, cette méthode elle prend un dictionnaire comme entrée, et elle génère le flux en respectant la hiérarchie du dictionnaire.

Je prends un exemple d'un de mes générateur de XML(voir figure 22), c'est une partie de code de la méthode *generate\_tags*, la clé de la dictionnaire va être utilisé comme le nom du balise dans le XML, et la clé « atr » indique que ses valeurs sont utilisé comme les valeurs de attribut dans la balise, et les dictionnaire dans la clé « val » va être présenté dans la valeur du balise. La clé « cdt » indique que les valeur dans la balise est dans le tags de CDATA qui présente que ces valeurs sont interprété comme des données pas comme des caractère spéciaux de XML.

```

95     tags['JobPost'] = {
96         'val': OrderedDict([
97             ('BaseInfo', OrderedDict([
98                 ('atr', OrderedDict([
99                     ('CustomerID', pj.get_config_value('config_client_id')),
100                    ('CompanyID', pj.get_config_value('config_company_id')),
101                    ('Status', pj_status),
102                ])),
103                ('val', OrderedDict([
104                    ('JobTitle', OrderedDict([
105                        ('val', pj.get_value('titre')),
106                        ('cdt', True),
107                    ])),
108                    ('Location', OrderedDict([
109                        ('val', pj.get_value('jobboards/51job/city') or pj.get_value(
110                            'jobboards/51job/region')),
111                    ])),
112                    ('Functions', OrderedDict([
113                        ('val', OrderedDict([
114                            ('Function1', OrderedDict([
115                                ('val', functions[0]),
116                            ])),
117                        ])),
118                    ])),
119                    ('Salary', OrderedDict([
120                        ('val', salary_code),
121                    ])),

```

Figure 22 le code qui définir la hiérarchie de XML

Et le flux XML correspondant à cette partie de code est dans la figure 23 :



```

<JobPost>
  <BaseInfo CustomerID="111" CompanyID="222" DivisionID="333" Status="A">
    <JobTitle>
      <![CDATA[ Finance Assistant ]]>
    </JobTitle>
    <Location>0401</Location>
    <Functions>
      <Function1>0502</Function1>
      <Function2>0502</Function2>
    </Functions>
    <Salary>2</Salary>
  </BaseInfo>
</JobPost>

```

Figure 23 le XML généré par la méthode *generate\_xml\_recursive*

## JSON<sup>7</sup>

Le générateur JSON est utilisé dans deux cas:

- Déposer les offres sur jobboard par webservice
- Générer un flux de format JSON pour le générateur Smartcha.

Dans cette partie, on discute sur le premier cas, pour la deuxième, on discutera dans le partie de générateur Smartcha.

Les étapes principale de générateur JSON sont suivants:

1. Générer le contenu de chaque fields
2. Transformer ver le format JSON
3. Transmettre le flux JSON au jobboard via Webservice

Pour le premier et la deuxième étapes, ils sont similaires que les méthodes de flux XML, par contre, pour la troisième étape, on utilise des Webservice proposé par le jobboard pour transmettre le flux JSON, les méthodes générales sont:

POST : souvent pour déposer des offres.

PUT : souvent pour modifier des offres.

DELETE : souvent pour supprimer des offres.

## Formulaire(Form, Selenium, smartcha)

### Form

Ce le type le plus générale et le plus nombreux dans le projet générateur de Multiposting, comme le générateur précédent, il a besoin que l'on redéfinissent, les méthodes abstraites définit dans ses classes mères. Les méthodes sont sensiblement les même : **generateTags**.

Ce type de générateurs se distingue par la méthode **sendForm**, c'est cette méthode qui va poster le formulaire. Son principe de fonctionnement est simple, il instancie un objet **WebFormClient** qui va se rendre sur la page internet du jobboard, charger le formulaire à remplir et ensuite il remplit le formulaire avec la méthode **fillForm** qui sera transmis via la méthode post qui va tout simple effectuer une requête POST.

Le workflow générale pour ce type de générateur est suivant :

Générer des tags => Se connecter au site => Accéder à la page de dépôt => Remplir le formulaire d'offre => Valider le formulaire => Chercher le phrase de validation => Changer le statuts d'offre  
Je vais prendre l'exemple du générateur qui diffuse des offres sur le site pracujw, un jobboard

<sup>7</sup> JSON (JavaScript Object Notation) est un format de données textuelles, générique, dérivé de la notation des objets du langage ECMAScript.

polonais.

La formulaire de jobboard est ci-dessous, c'est une formulaire très classique qui contient trois étapes : Ajouter une offre, Valider la offre et Confirmer le dépôt.

Figure 24 un formulaire classique pour déposer des offres

D'abord, on récupère le nom des champs depuis la formulaire pour la méthode **generateTags**, et on aussi crée des Mappings ou des champs spécifique si nécessaire. Ici, on va créer un mapping de région polonais, ce mapping fait un association de les région polonais dans le système de Multiposting avec les ids de région défini dans la formulaire de jobboard (voir figure 25) :

```

15     mapping_region = {
16         1: u"podlaskie",
17         2: u"pomorskie",
18         3: u"śląskie",
19         4: u"świętokrzyskie",
20         5: u"małopolskie",
21         6: u"lubuskie",
22         7: u"łódzkie",
23         8: u"wielkopolskie",
24         9: u"podkarpackie",
25         10: u"zachodniopomorskie",
26         11: u"kujawsko-pomorskie",
27         12: u"mazowieckie",
28         13: u"dolnośląskie",
29     }

```

Figure 25 le Mapping de la région polonaise

Comme dans le système de Multiposting, on a déjà des champs commun pour remplir des champs de jobboards, on a plus besoin de créer des champs spécifique pour le jobboard, du coup, on peut créer la méthode *generateTags* (*generate\_tags* en python) pour remplir des champs dans la formulaire de jobboard, voici, dans la figure 26, c'est la méthode *generate\_tags* du générateur pour le jobboard.

```

53 def generate_tags(self, pj, tag):
54     category_id = pj.jobboard.get('jobboard/PracujwPL/category_id')
55     if not category_id:
56         self.skip_mapping(pj, 'config du champs manquant!')
57     tag['type_id'] = pj.get_field_mapping('type_contrat//jobboard/PracujwPL/type_contrat')
58     tag['category_id'] = pj.get_value(category_id)
59     tag['title'] = pj.get_value('titre')
60     city_id = self.utils.get_region(pj, self.mapping_region, 'pl')
61     if not city_id:
62         city_id = -1
63     tag['city_id'] = city_id
64     text_apply = self.utils.get_apply_text(pj)
65     if pj.is_url and pj.get_apply_url():
66         text_apply = text_apply.replace(pj.get_apply_url(),
67                                         '''+pj.get_apply_url()+'':'+pj.get_apply_url()''')
68
69     tag['description'] = u'%(desc)s \n\n %(apply)s' % {
70         'desc': self.filter(pj, self.utils.get_format_description(pj, None, True, True)),
71         'apply': text_apply,
72     }
73     tag['company'] = pj.get_value('company/name')
74     tag['url'] = pj.get_value('company/url').replace('http://', '')
75     tag['poster_email'] = pj.get_apply_email()

```

Figure 26 la méthode *generate\_tags*

Dans la méthode, on a bien rempli les champs `type_id`, `category_id`, `title`, etc. en utilisant directement les valeurs de posting ou les méthodes pour transmettre des valeurs.

Ensuite, on va créer la méthode *sendForm* (*send\_form* en python), cette méthode va instancier un objet de **WebFormClient** et diriger le client à manipuler sur la formulaire.

```

37 def send_form(self, pj, tag):
38     client = WebFormClient()
39     base_url = pj.jobboard.get('jobboard_url_1')
40     client.get(base_url)
41     client.fill(base_url, tag)
42     client.post(base_url)
43     self.save_log(client.response.content, pj, '_step2.html')
44     posting_url = client.response.url.replace('verify', 'publish')
45     self.commit(pj)
46     client.post(posting_url)
47     self.save_log(client.response.content, pj, '_response.html')
48     validation = re.compile('confirm')
49     result = re.search(validation, client.response.url)
50     if result is None:
51         self.skip_validation(pj, 'Validation sentence not found')

```

Figure 27 le code de la méthode *send\_form*

Dans la méthode, on « ouvert » le premier page de la formulaire en utilisant la méthode *client.get(base\_url)*, ensuite, on remplit la formulaire en utilisant les donnée préparé par la méthode *generate\_tags*, et on « envoi » la formulaire en utilisant la méthode *post*, et on « valide » une deuxième fois pour accéder à la page de confirmation (ligne. 46 de la figure 27). Maintenant, on arrive à la dernière page, on doit chercher le phrase de validation, si on ne trouve pas, c'est-à-dire il y a des erreur inconnu pendant la processus précédent, il faut lever une erreur de validation (ligne. 51 sur la figure 27), si on a bien trouvé la phrase de validation, c'est-à-dire l'offre est bien envoyé au jobboard, on peut continuer les processus suivant sans enlever des erreurs.

## Selenium

Pour beaucoup de jobboards et des sites d'écoles, la procédure de dépôt d'offre d'emploi ou de stage se fait via un formulaire. Dans la plupart des cas, une simple requête POST émise vers le site suffit à transmettre les informations. Néanmoins, il arrive souvent que le formulaire soit composé

d'éléments complexes, soumis à certaines règles, criblé d'éléments Javascript peu maniables, ou constitué de trop d'étapes fastidieuses. Par exemple, dans la figure 28, le formulaire contient des Javascript pour contrôler le remplissage de formulaire, il faut cliquer sur des boutons dans le formulaire pour continuer le dépôt d'offre.

Figure 28 un formulaire qui demande l'exécution de Javascript

Pour réussir à diffuser les annonces dans de tels cas, Multiposting utilise l'outil Selenium. Il s'agit d'un automate de navigation. Les générateurs Selenium sont réalisés comme les générateurs BaseForm, la différence se trouve dans la méthode *sendForm*. Au lieu d'instancier un objet **WebFormClient** cette fois-ci on va instancier un objet **WebDriverClient** qui va ouvrir un navigateur et chaque méthode appelée sur le nouvel objet instancié va simuler une interaction utilisateur (ouvrir une page par l'URL, cliquer sur tel bouton, remplir tel champ). Il suffit d'identifier l'élément de la page, dans le code HTML, par son attribut « name » ou « id », et si l'élément ne contient pas les deux attributs, on utilise le « XPath<sup>8</sup> » qui indique le chemin pour trouver les éléments dans la DOM<sup>9</sup>.

L'intérêt de ce pilotage est qu'il est possible d'écrire une suite d'instructions directement dans le code PHP du générateur, qui seront alors exécutées automatiquement lors de la diffusion d'annonces. En particulier, vu que les commandes sont écrites directement dans le code, elles sont facilement remaniables, et peuvent utiliser n'importe quelle donnée accessible depuis le programme. Ainsi, lorsque l'on lance le générateur avec l'outil Selenium, on peut le voir ouvrir un navigateur, et remplir automatiquement, devant nous, tous les champs d'un formulaire avec les données entrées par le client.

## Smartcha

Lorsqu'on a un formulaire qui nécessite la validation d'un Captcha (voir la figure 29), on parle de générateur Smartcha, on désigne un générateur qui nécessite la validation d'un Captcha.

Le générateur Smartcha contient deux parties de code : générateur qui génère le flux JSON en utilisant les données dans le posting et un générateur qui pilote le navigateur pour remplir automatiquement les champs de jobboard en utilisant le flux JSON.

Le deuxième générateur va lire le flux JSON depuis le serveur de feed (comme le flux XML) en utilisant le url avec le token, en suite, il récupère les valeurs dans le flux JSON et utilise des valeurs pour remplir tous les champs sauf le champs de Capcha de la formulaire, en ce moment, le

<sup>8</sup> XPath est un langage (non XML) pour localiser une portion d'un document XML

<sup>9</sup> DOM : Le Document Object Model

générateur va localiser ce champs et demander un humaine pour remplir ce champs, quand le champs Capcha est bien rempli, le générateur va chercher le phrase de validation et changer le statut de posting.

Figure 29 des formulaires avec Capcha

## Mail

Le générateur mail est souvent utilisé pour diffuser des offres sur l'école, car pour certains écoles, ils ne proposent pas de formulaire pour déposer des offres, mais ils proposent une adresse mail pour contacter le responsable de stage d'école.

Dans ce cas, on associe le jobboard avec un générateur Mail, le jobboard est maintenant de type **jobboard mail**, et les offres déposées par client vont être diffusées par mail. Car il n'y a pas de contrainte dans le contenu de mail, du coup, pour les jobboard mail, on l'associe avec la même générateur – **SmartMail**, et le générateur va utiliser des templates en fonction de la langue du posting pour générer le contenu de mail et la pièce jointe.

Plus techniquement, Pour la génération de contenu de mail, d'abord on génère une page HTML en utilisant le template de Django, si le destinataire ne souhaite pas recevoir des mails qui contiennent des éléments HTML, on utilise un module de python nommé « html2text » pour convertir la page html vers un texte qui est propre et facile à lire.

Pour la pièce jointe, Le générateur **SmartMail** peut générer deux types de pièces jointes : Docx et PDF, d'abord, on initialise le document en définissant les styles de chaque paragraphe (titre, logo, contenu, etc.), après on utilise un module python nommé « docx » pour générer le fichier Docx ou « SimpleDocTemplate » pour générer le fichier PDF en fonction des styles définis dans le processus précédent.

## Correction (maintenance)

Les tâches de corrections sont créées lorsqu'on a des postings en erreurs comme je l'ai expliqué dans le chapitre « traitement d'erreur ».

Le but de ses tâches est de maintenir le système de multidiffusion à jour et de corriger les imperfections qui n'ont pas été détectées pendant les intégrations.

Généralement les tâches de corrections sont dues à des mises à jours du côté du jobboard ou d'une intégration qui n'a pas pris en compte toutes les exigences du jobboard. Pour résoudre ces tâches on consulte les logs. Les logs sont les fichiers dans lesquels on a enregistré toutes les étapes effectuées par le générateur lorsqu'il va poster une offre. Si il y a un message d'erreur c'est dans les logs que l'on trouve. Dans le meilleur des cas une tâche de corrections se résout en 20 minutes au

maximum. Parfois il arrive que le problème ne soit pas évident, bien que la solution peut être très simple.

Voici des tâches de correction les plus courantes pendant mon stage :

- **Département, région et pays non supporté**

C'est le plus fréquent erreurs dans la tâche de correction, mais c'est aussi une erreur relativement plus facile à corriger, le problème vient de un client pose une offres à un jobboard, mais les information sur la localisation d'offre est pas encore supporté par le jobboard ou par le générateur, du coup le générateur ne trouve pas de valeur correspondant pour remplir le champs pour le jobboard.

Pour résoudre ce type d'erreur, il suffit d'ajouter un mapping de valeur dans la système de Multiposting ver la valeur dans la jobboard, du coup, il faut rendre dans le site de jobboard pour trouver le bon valeur pour présenter la localisation dans le jobboard.

- **Fill\_form failed**

Ce type d'erreur généralement vient du générateur de type **Form** ou **Selenium** car c'est une erreur sur la méthode *fill\_form*.

Cette type d'erreur nous dit que le générateur n'arrive pas de trouver le formulaire par l'URL indiqué dans le système de Multiposting.

Il y a trois causes possible :

Le premier c'est le générateur est pété dans l'étape précédent (étape de connexion au jobboard, par exemple), du coup, il n'arrive pas de accéder le page qui contient le formulaire.

La deuxième cause est que le jobboard a changé le nom du formulaire ou l'url de formulaire.

La troisième cause c'est le pire cas, le jobboard a tous changé les éléments de la formulaire.

Dans le premier cas, il faut vérifier que le générateur avait bien arrivé au page de la formulaire, si non, il faut regarder est-ce qu'il y a des étapes précédents, est-ce que c'est le problème de login (pour des formulaire qui est réservé au adhérent).

Pour le deuxième cas, il faut trouver le bon nom de la formulaire, et faire des changement correspondant dans le générateur.

Pour le troisième cas, il faut regarder si la nouveau formulaire est totalement différent qu'avant, si oui, il faut demander l'équipe support pour créer une tâche de réintégration.

- **Erreur sur le décodage de caractères**

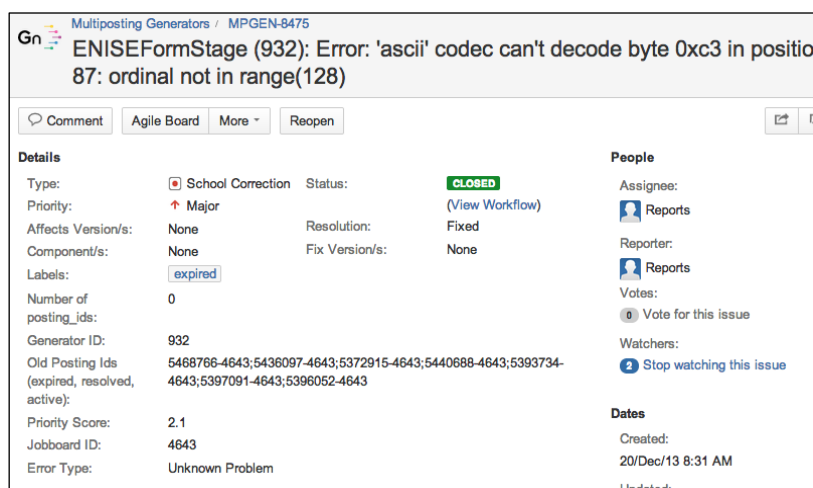


Figure 30 la page de la tâche pour résoudre l'erreur décodage

Ce type d'erreur est très souvent sur les générateur en python, car par default, python utilise ASCII (python 2.7) pour stocker des données, mais dans le interface de Multiposting pour déposer des offres, les clients peuvent valider des offres avec tous type d'encodage (UTF-8, ISO 8859-1, etc.), Donc dans le générateur, quand il essaye de manipuler des valeur avec des encodage inconnu, le python va enlever une exception de ce type d'erreur, et éventuellement attrapé par le système de rapport de Multiposting.

Pour ce type d'erreur, il faut bien connaître l'encodage de texte sur le posting, les décodé et enfin les encodé avec l'encodage supporté par le jobboard, mais de certain cas, des méthodes ne supporte pas certain encodage, du coup, il faut filtrer les caractères inconnu ou les bien transmettre des caractères par un caractère avec l'encodage supporté, voici, dans la figure 31, c'est un dictionnaire pour faire des remplacement de caractères :

```

293     _special_chars_replace = {
294         '-': '-',
295         'é': 'e', 'è': 'e', 'ê': 'e', 'É': 'E', 'Ê': 'E',
296         'à': 'a', 'â': 'a', 'À': 'A', 'ä': 'a',
297         'ù': 'u', 'û': 'u',
298         'ç': 'c', 'Ç': 'C', 'ć': 'c',
299         'ô': 'o', 'ó': 'o', 'Ó': 'O',
300         'î': 'i', 'ï': 'i', 'İ': 'i',
301         'ł': 'l', 'Ł': 'l',
302         'ñ': 'n', 'ń': 'n', 'Ś': 's', 'ś': 's',
303         'ż': 'z', 'ź': 'z', 'Ź': 'Z', 'ż': 'z',
304     }

```

Figure 31 le dictionnaire pour remplacer des caractères spéciaux

#### ● Phrase de validation non trouvé

Ce type d'erreur est très souvent dans les tâches de correction, les causes pour ce type d'erreur est variant, mais le plus générale cas, c'est à cause de la mal respections des contraintes des offres, par exemple, un ou plusieurs champs obligatoire ne sont pas remplis, la format de code postal ou l'adresse email ne sont pas correctes, etc.

Du coup, pour résoudre ce type d'erreur, il faut trouver les contraintes qui ne sont pas respectés, et les ajouter dans le interface de Multiposting, ou faire des modification au niveau du générateur.

## Intégration

L'intégration d'un jobboards ou d'une école émane généralement d'une demande client et si besoin on (l'équipe Support Client) contacte le jobboard pour avoir de la documentation et toutes les informations nécessaires. Dans la figure ci-dessous(figure 32), elle présente le flux de travail pour des tâches d'intégration.

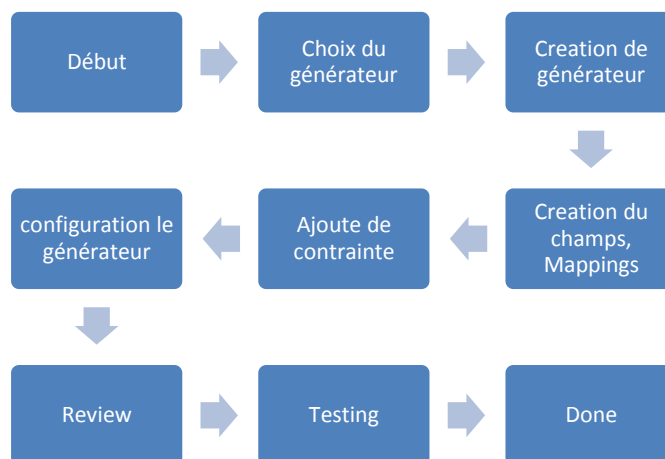


Figure 32 le flux de la tâche d'intégrations

### Choix du type de générateurs

Le choix du type générateur est généralement évident, et elle est parfois imposé lorsque le jobboard donne des consignes strict. Lorsqu'on doit intégrer un générateur de type flux(XML ou CSV) ou un formulaire avec Captcha, il n'y a aucune question à se poser et on passe directement à l'étape de création du générateur. En revanche lorsqu'on a un formulaire, on doit commencer par analyser si ce n'est pas trop compliqué d'exécuter des requêtes POST. Théoriquement c'est toujours possible d'envoyer une requête POST lorsqu'on a un formulaire à intégrer mais parfois le JavaScript en place nous empêche d'envoyer le formulaire correctement et c'est là qu'on choisit de faire un générateur de type *BaseSelenium*, on évite autant que possible de faire ce genre de type de générateur car il consomme beaucoup de ressource.

### Création du générateur

Il faut tout d'abord créer le générateur sur le Backend pour que le système sache qu'il y a un nouveau générateur d'intégrer au système et afin de savoir quel générateur est utilisé pour diffuser sur un jobboard, on associe le jobboard à un générateur.

Il est possible d'associer plusieurs jobboards au même générateur à condition que les jobboards ont le même comportement (on fait cela souvent avec les jobboards mails). Ceci est aussi un moyen de limiter à duplication de code.

Lorsqu'on associe un générateur à jobboard on crée une entrée dans la table *jobboard\_generator* et lorsqu'on désire désassocier jobboard d'un générateur on peut exécuter simplement une requête



SQL pour mettre à jour la base de données en supprimant l'items concernant dans la tableau *jobboard\_generator*.

## Création de champs et mappings

Parfois, il arrive qu'il manque des champs sur le Frontend sur le formulaire que l'on propose au client et en fonction des jobboards, on a des champs spécifique à rajouter. Ou bien pour diffuser un offre, il faut utiliser un compte ou un configuration spécifique par client.

On crée le champs sur le Frontend en choisissant ses propriétés (texte, sélect, etc.).Maintenant, ce champs est existé dans le Backend de Multipositng, par exemple, dans la figure 33, on vient de créer un champs domaine pour le jobboard Pole Emploi, mais pour qu'il soit visible dans la formulaire du jobboard de l'interface de Frontend, il faut associer le champs avec le jobboard en ajoutant un item dans le tableau *jobboard\_posting\_field*. Le champs sont ensuite associés au jobboard auquel il est destiné afin qu'il soit visible sur le Frontend.

**Modifier le champ: jobboard/ANPE/Domaine**

Nom: jobboard/ANPE  
 Type: Select  
 Catégorie: Secteur  
 Display: 512  
 Groupe:  
 Position:  
 Parent: -  
 Retour à la liste [VALIDER]

**Label et items, par culture**

Ajouter un itemAjouter une culture  
 Culture: C  
 Label: Domaine mêt

ID	Parent	Clé	Label	Ordre	Actif
49795	A		Agriculture et pêche, Es	1	-
49796	B		Arts et Façonnage d'ouv	2	-
49797	C		Banque, Assurance, Imi	3	-
49798	D		Commerce, Vente et Gr	4	-
49799	E		Communication, Média	5	-
49800	F		Construction, Bâtiment	6	-
49801	G		Hôtellerie-Restauration	7	-
49802	H		Industries	8	-
49803	I		Installation et Maintena	9	-
49804	J		Santé	10	-
49805	K		Services à la personne	11	-
49806	L		Spectacle	12	-
49807	M		Support à l'entreprise	13	-
49808	N		Transport et Logistique	14	-

[VALIDER]

Figure 33 La création de champs domaine pour le jobboard Pole Emploi

Et pour le champs de configuration, on le crée comme le champs sur le Frontend, mais on l'associe au jobboard en utilisant le tableau *jobboard\_config\_field* pour que le champs soit visible sur le config du client de jobboard. Dans la figure 34, les champs « mot de passe », « Facebook page ID », « initial job state », etc. sont des champs spécifique pour le config du jobboard.

**Service Recrutement 1 (G2535) @ Facebook (Work for Us) (J1977)**

Pour vous rendre sur le site de Facebook (Work for Us) cliquez : [ici](#)

Envoyer le nom de l'entreprise :  Facebook (Work for Us)

Champs supplémentaires

Mot de passe : \*  LMqtDRfbhaE3HrMKATCNo67KgJGFCiyd ✓

Options URL candidature :  2280 ✓

Crédits job board :

Facebook Page ID : \*  197292926807 ✓

Crédits initiaux :

Ne pas transmettre le descriptif société : ☐ Facebook (Work for Us)

Initial job state :  Select... Facebook (Work for Us)

Posting Culture par Défaut :  Select... Facebook (Work for Us)

Figure 34 la page de la gestion des config de client sur un posting

Pour certains champs de type sélect, on récupère les champs du jobboard qui ensuite sera mappé avec les champs proposés par Multiposting afin de ne pas demander aux clients de saisir deux fois la même informations. Généralement les champs concernés les plus souvent sont type de contrat (voir la figure 35), niveaux d'études, années d'expériences et bien d'autres.

Par défaut	CDI	CDD
0 Par défaut	0 CDI	0 Invalide[+]
1 CDI	1 CDI	1 CDD
2 CDD	2 CDD	2 CDI
3 Stage	3 Stage	3 Freelance
4 Intérim	4 Intérim	4 Intérim
5 Freelance/Indépendant	5 Freelance	5 VIE
6 Contractuel	6 Freelance	6 Alternance
7 Apprentissage/Alternance	7 Alternance	7 Stage de fin d'études
8 VIE	8 VIE	8 Stage
9 Franchise	9 Freelance	9 Job étudiant

Figure 35 Association du mapping de type de contrat

Sur l'illustration ci-dessus le champs de gauche contient les valeurs visible sur le Frontend et le champs de droite les valeurs disponible sur le jobboard. Au milieu on voit à quel valeur sont associés les champs que l'on propose car en effet, il est impératif que toutes les valeurs des champs que l'on propose soit associé à une valeur d'un champs disponible sur le jobboard afin d'éviter tout problème de fonctionnement.

SmartMapping :

C'est un projet récent destiné à faciliter la création et la modification du Mapping, il compare les valeurs de champs gauche et champs droit en utilisant un algorithme complexe et les tri par « similarité » décroissant, c'est-à-dire le plus corrélé champs est placé dans le premier, et le deuxième corrélé champs est place dans le deuxième place et ainsi de suite.

Figure 36 l'interface de smart mapping

Ici, dans la figure 36, pour le champs de gauche « administration – service généraux » le smartMapping conseille que le champs de droit « administration / Secrétariat » est le plus possible mapping

Une fois tous les champs ont été mis en place et que toutes les contraintes à prendre en compte ont été répertoriées, on peut commencer à coder le générateurs dans le langage choisit (PHP ou python).

## Ajoute des contraintes sur les champs

Il probable que le jobboard a ajouté des contraintes sur le champs pour déposer des offres, et pour conformer le jobboard, on doit ajouter les contraintes correspondant dans notre interface, du coup, on a souvent besoin de rendre un champs obligatoire, ou bien pour un champs select-multiple la nombre d'élément maximum ou pour un champs de texte, le nombre de caractères maximum, etc. Dans la figure 37, elle affiche un non-respect de la contrainte de 4 éléments maximum à choisir.

Figure 37 la contrainte sur le champs fonction

Pour ajouter les contraintes, on utilise la tableau *field\_data* pour mettre un champs obligatoire on utilise la tableau *field\_data* plus *field\_data\_i18n* si on veut ajouter des contraintes complexe par exemple, on ajoute des Regular expression.

Pour mettre un champs obligatoire dans le Frontend, il suffit de ajouter une ligne dans le *field\_data* en précisant le *field\_id*, *jobboard\_id*, type de contrainte et flag de contrainte, ici, pour mettre le champs obligatoire on met *type=2* et *flags=3*.

Pour les d'autre type de contrainte, on utilise les différent type de « *type* » et « *flags* », et on ajoute la contrainte dans le type de JSON dans la valeur de la tableau *field\_data\_i18n*.

## Configuration sur le générateur

Pour automatiser le générateur, on utilise Celery pour gérer les horaires de lancement de générateur, et pour l'intégration, il faut ajouter une tâche dans la base de donnée en précisant le générateur, le jobboard(si on ne précise pas, dans la tâche, on va lancer le générateur pour tous les jobboard associés dans le planning), le nom de la tâche et les horaires qui est le plus important.

La format de horaire est similaire que celle dans le Crontab (voir annexe).

Ici, c'est un exemple pour ajouter une tâche dans le base de donnée.

```
EntryCollectionDocument=EntryDocument.get_document_class('celery_schedule_generators')
ecd = EntryCollectionDocument(
    name='Sueddeutsche',
    task = 'generators.tasks.generate.generate_tasks',
    schedule='45', '23', '*', '*', '*',
    kwargs={"generator_name": "SmartXml", "jobboard_id": "6960"},
    options={'queue': 'prod11'})
ecd.save()
```

Ici, on utilise l'objet *EntryDocument* de Framework Django pour exécuter des requête sur le base de donnée, dans le tableau *celery\_schedule\_generators*, les plus importants paramètres sont « *schedule* », « *kwargs* » et « *options* ».

Le champs « *schedule* » présente le planning de la tâche, ici, '45', '23', '\*', '\*', '\*', présente la tâche sera lancé à 23 heure 45 minute tous les jours .

« *kwargs* » est une dictionnaire de paramètres, le clé « *generator\_name* » présente le nom du générateur on va utiliser dans la tâche, et le « *jobboard\_id* » présente l'id du jobboard dont on diffuse des offres, si on laisse ce paramétré vide, le générateur va diffuser les offres de tous les jobboard associés dans la tâche.

« *Options* » présente l'endroit pour exécuter la tâche, ici, prod11 indique que la tâche va se lancer dans le serveur prod11.

## Amélioration

Les amélioration sur le projet sont souvent distribué par des tâche de type « *story* », pendent mon stage j'ai amélioré des méthode très souvent utilisé dans le projet générateur et créé des tests unitaires correspondent, par exemple :

*Get\_start\_contract\_date* : cette méthode retourne la date début de contrat, car dans certain posting, le recruteur ne précise pas la date de contrat début, du coup, cette méthode doit utiliser d'autre champs dans le posting pour déduire la date début du contrat.

*Get\_end\_contrat\_date* : comme la méthode de *get\_start\_contract\_date*, cette méthode renvoie un date de fin de contrat pour des contrat avec durée limité.

*Get\_apply\_email* : cette méthode renvoie l'adresse mail pour envoyer la candidature de chaque offre.

Car les méthodes ci-dessus sont presque utilisées par tous les générateurs de Multiposting, du coup, pour être sûr que la méthode améliorée fonctionne sur les anciens générateurs, on a créé des tests unitaires pour les méthodes précédentes.

J'ai amélioré aussi des scripts pour automatiser la création de jobboard école mail et le associer directement aux clients. Cette script nous permet d'intégrer un jobboard d'école dans un instant.

Mais le inconvénient de cette script est qu'il faut toujours préparer un fichier CSV qui contient les informations nécessaires pour la création, mais s'il y a des erreurs sur les données, il faut recréer chaque fois le fichier et ré-exécuter le script, et l'exécution de la script demande un accès sur le serveur de production. Du coup, j'ai aussi réalisé un Mini projet sur la création de jobboard mail en utilisant une interface visuelle.

## Projet Report

Le projet Report concerne principalement la création des interfaces pour accélérer et faciliter le travail, les modifications sur les données qui sont très souvent utilisées sur le jobboard et le générateur, des outils pour augmenter l'efficacité sur le travail fréquent, etc.

Dans mon stage, je travaille principalement sur quelques tâches de type story et un mini projet du projet Report, je vais présenter en détail mon mini projet du projet Report.

### Mini projet

- Présentation de la modèle MVC

Avant de commencer la présentation de mon mini projet, je souhaite rappeler le modèle MVC, le framework Django et Symfony utilisent le modèle MVC, ici c'est le modèle MVC de Django (voir la figure 38):

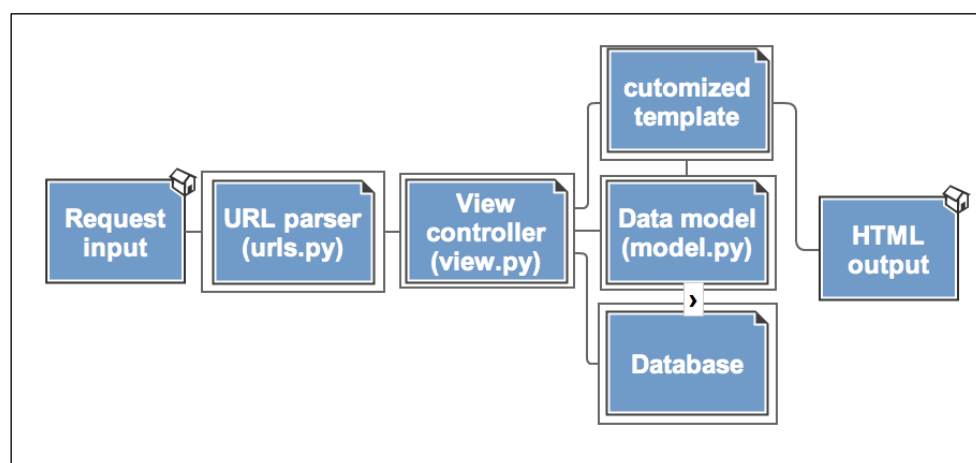


Figure 38 Modèle MVC de Framework Django

Sur le schéma, une demande (request) est envoyée du client au Django et est acheminée à travers l'analyseur de l'URL (urls.py). Il est alors dirigé vers une classe spécifique qui est configurée dans le fichier views.py qui à son tour accède au modèle de données et base de données documentée

dans `models.py`. Enfin, `views.py` renvoie la sortie au client en fusionnant les données de la base de données et d'un Template personnalisé.

## Présentation du projet

Nom du projet : Ajoute d'une nouvelle rubrique de gestion de mail dans reports.

Chaque semaine, Multiposting reçoit des demande de client pour intégrer des nouveaux écoles par envoyer des mails qui contiennent des offres, et avant, l'équipe Biz-Dev crée un fichier CSV qui contient tous les information nécessaire pour la génération de jobboard mail pour école, et lui donne au équipe Dev-Prod pour les scanner dans le systèmes de Multiposting.

Mais ça arrive souvent que les information sur le fichier CSV ne sont pas complet, ou il contient des erreurs, du coup, chaque fois, le membre d'équipe Dev-Prod doit vérifier le fichier CSV ligne par ligne à la main avant de le scanner dans le système, ça prend beaucoup de temps pour vérifier le formulaire CSV et le plus important, pour scanner le fichier CSV, il faut lancer un script dans le serveur de production, qui aura besoin de responsable de projet pour exécuter le script.

Du coup, le projet est créé pour simplifier le scan du fichier CSV : on propose une interface(page web interne) au Biz-Dev, cette interface permet de télécharger le fichier, vérifier les erreur sur la formulaire, donner une interface pour corriger les erreurs et les scanner dans le systèmes de Multiposting si le formulaire est bon après la correction.

Le projet aussi contient une partie pour créer une interface qui permet de créer des jobboard mail directement sans avoir besoin un fichier CSV. Et bien sûr, l'interface va vérifier tous les erreurs avant de le passer dans le système.

Grace à ce projet, l'intégration d'école par mail n'aura plus besoin l'équipe de Dev-Prod. Et la génération de jobboard mail sera plus stable et bien fondé.

## Analyse de la modèle

Car c'est une sous projet de MpJobs, la plupart de modèle sont fait, mais car le projet est pour créer des école de Mail, et le modèle de la partie école n'existe que dans le Framework Symfony, du coup il faut migrer des modèles de Symfony ver Django :

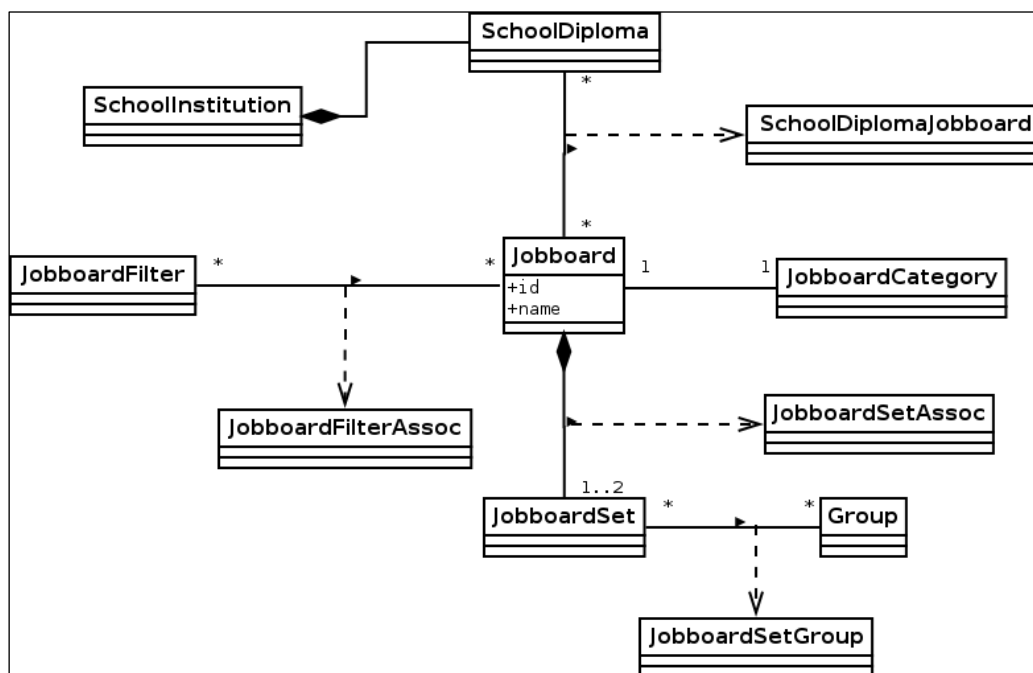


Figure 39 Relation UML sur la partie jobboard école de Multiposting

Sur le schéma, le centre est la classe *jobboard*, pour les *jobboard* de type école, il est associé à un catégorie, des filtres, des diplôme et un ou deux *jobboard set* en fonction de la drapeau du *jobboard*.

Pour l'association de *jobboard* et *filter*, car un *Jobboardfilter* peut être utilisé par plusieurs *jobboard*, et un *jobboard* peut avoir plusieurs filtre, c'est une relation « many-to-many », du coup, on utilise une classe d'association : *JobboardFilterAssoc* pour associer les deux classe.

Pour la classe *SchoolDiploma*, il présente un diplôme délivrer par une école, du coup, la classe *SchoolInstitution* compose plusieurs *SchoolDiploma*.

Pour l'association de *jobboard* et *SchoolDiploma*, car une école peut avoir plusieurs *jobboards* (un *jobboard* par un département par exemple), du coup, c'est aussi une association « many-to-many », ici , on utilise la classe d'association *SchoolDiplomaJobboard* pour associer un *jobboard* avec des diplôme et éventuellement avec un école.

Pour le *jobboardset*, c'est pour séparer les interface de stage et d'emploi pour un *jobboard*. Il est créé pour avoir deux interfaces différent pour des utilisateur(présenté par la classe *Group*), afin que pour un *jobboard*, les différents utilisateurs peut être autorisé de diffuser des offres de stage ou d'emploi ou les deux.

Du coup, le *jobboard* de type école sont associé avec la classe *Group* par l'association de *jobboardSetGroup* avec un intermédiaire de la classe *jobboardset* au lieu de *JobboardGroup* pour les *jobboard* non école qui sont associé directement.

## Création de vues

Pour faciliter le design, je utilise trois vues principales :

1. Vue pour choix du mode de création : télécharger un fichier ou création à la main.

Figure 40 l'accueil de la création de jobboard mail

Dans ce vue, on peut télécharger un fichier csv en cliquant le bouton « choisissez un fichier » et après on clique Submit juste au-dessous pour valider le téléchargement.

On peut aussi cliquer le bouton « submit » à droite pour entrer à la formulaire de création d'école manuellement.

2. Vue pour ajouter des école ou modifier des donnée posté par le fichier.

Figure 41 l'interface pour éditer les données

Dans ce vue, on peut ajouter ou modifier des valeur dans la formulaire, et une message d'erreur va afficher sur la page si la valeur de un champs est incorrecte.

On peut aussi ajouter un nouveau ligne en cliquant le bouton « + » juste au-dessus de la bouton « submit », et si on veut supprimer une ligne, on peut facilement cliquer le bouton « corbeille » pour supprimer toute la ligne correspondant.

3. Vue pour créer les jobboard et afficher les résultat de la création .

Le nom du vue est « SchoolImportView », il reçoit les donnée « post » par le vue précédent, et il appelle la méthode pour la création jobboard mail et l'association de jobboard générateur , le processus est ci-dessous :

Création de *jobboard*

Association de *jobboard* avec des *jobboard\_set* selon l'interface

Création de *Institution*

Création de *School\_diploma*

Association de *school\_diploma\_institution*

Association de multi language

Création de *jobboard\_posting\_field* et *jobboard\_config\_filed* selon les flags

Association de *group* avec *jobboard*



Création de générateur mail

Association de *jobboard\_generator*

Après le procédure finit, il affiche les résultats sur la création de jobboard mail (Voir la figure 42) en présentant les ids de jobboards créés, le statuts de la création, etc.

Jobboard ID	Jobboard name	Category	Flag	Groups	Interface	Institutions	Emails	Status
7472	UTC Compiègne	2	7	645	stage/emploi	837	blabla@utc.fr	🟢
7473	UTBM	2	2	1235	stage	156	bere@r.com	🟢

Return to Menu

Figure 42 Résultats de la création de jobboard école mail

4. Vue pour API de validation de donnée pour la création d'école.

Car c'est un API, du coup, il utilise pas le template pour « render » la réponse.

Il prend une liste de liste qui contiennent les valeur des formulaire comme entrée, et il renvoie une liste de dictionnaire correspondant au statuts de chaque valeur et les message d'erreur si la formulaire contient des erreurs de la entrée. un exemple de entrée sortie de cette vue est disponible dans l'annexe.

Évidemment, le plus compliqué vue est la vue 3 pour ajouter est modifier de la création de école Mail car il accepte deux type de requête : la requête POST vient de télécharger de fichier et la requête GET vient de la création à la main. Et il aussi traite de la communication entre la formulaire et l'API.

## Création de Template

Le template est une partie essentielle pour le présentation, le syntaxe de template dans le framework Django est très similaire que le syntaxe de python, voici un exemple :

La balise `{{ variable.key }}` présente le valeur de variable transmit par le vue, et le balise `{% tags %}` pressente les tags dans le template, Certains créent texte dans la sortie, certains contrôlent le flux en effectuant des boucles ou logique, etc.

Dans le mini projet, chaque vue utilise un Template différent pour séparer les différents étapes.

## Création de contrôleur

Comme présenté avant, dans le Django on utilise l'analyseur syntaxique de l'URL comme le contrôleur, et le dans le fichier `url.py` il contient un mapping de url parser ver le vue associé.

Voici les mapping concernant pour le projet :

school/upload/	•SchoolUploadView
school/add_edit/	•SchoolAddOrEditView
school/import/	•SchoolImportView
school/validation/	•SchoolValidationView

Figure 43 le correspondance entre l'url et vue

Le contrôleur appelle la vue correspondante quand le format de l'URL est matché. Vous pouvez également regarder le code du contrôleur dans l'annexe.

# Conclusion

Ce stage a été sous plusieurs aspects riche d'enseignements. Le stage consistait à participer au développement et à la maintenance d'un système de multidiffusion d'offre d'emplois sur le Web.

D'abord, c'est une expérience professionnelle très précieuse pour moi, pendant la stage, j'ai connu beaucoup des collègues très sympas, et j'ai aussi appris des connaissances sur l'entreprise.

J'ai pu améliorer mes connaissances sur le système Linux, en programmation web, et aussi tiré profit sur le fait de travailler en équipe pour améliorer la rentabilité. Pour des méthodes informatiques, j'ai acquis des expériences sur le Git, Jenkins, JIRA et des méthodes agiles comme Extreme Programming (XP) qui sont très utiles pour tous les développements du projet informatique. J'ai également pu m'améliorer dans des technologies comme le Python, le PHP, le SQL, et la Programmation Orientée Objet et bien pour des technologies sur le web interface design comme le JavaScript, le CSS, les templates, etc..

Le contact avec le monde du travail m'a permis de progresser dans de nombreux domaines, notamment technique et social. J'ai aussi découvert de nouveaux outils et de nouvelles technologies qui sont venus compléter mon apprentissage.

Le stage est mon première expérience de longue durée comme un métier d'ingénieur dans une entreprise informatique. J'ai donc pu découvrir de l'intérieur les rouages d'une entreprise efficace : une excellente coopération et une communication sans failles, aussi bien entre les services qu'au sein d'une même équipe, le tout dans une ambiance jeune, dynamique et enjouée. Cependant, il s'agit bien d'une des qualités de l'ingénieur : l'adaptation, et je suis convaincu que ce stage m'a permis de progresser sur ce point.

En conclusion, mon stage s'est révélé comme étant une période d'apprentissage qui m'a permis de mettre en œuvre des compétences scolaires et d'acquérir des compétences professionnelles et humaines.

# Bibliographique

Confluence Multiposting : <https://multiposting.atlassian.net/wiki/>

Wikipedia.fr : <http://fr.wikipedia.org/wiki/>

Using Django to create complex sites simply, Erik Mitchell,

<http://www.erikmitchell.info/2010/09/28/using-django-to-create-complex-sites-simply/>

Une Référence Visuelle de Git, Mark Lodato,

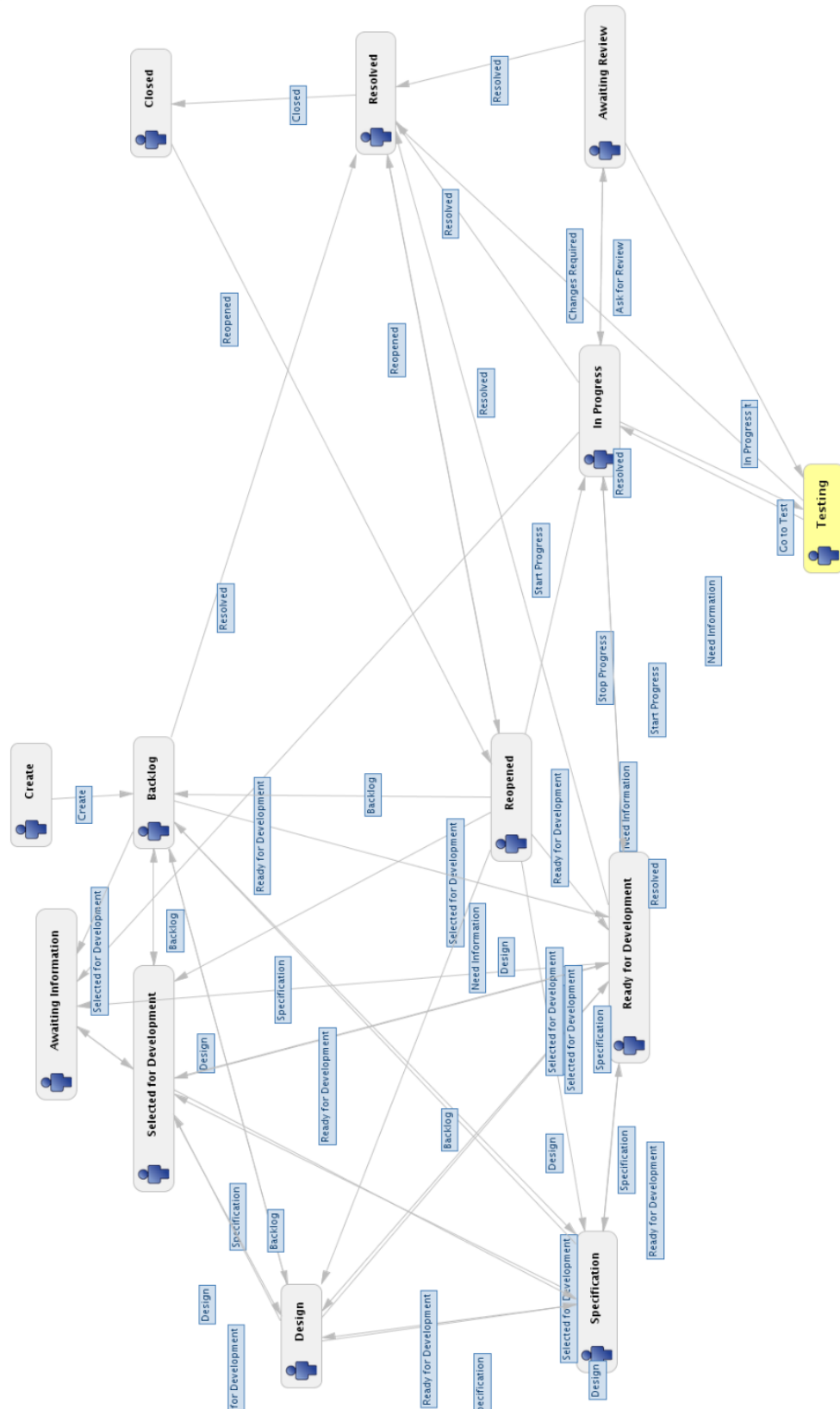
<http://marklodato.github.io/visual-git-guide/index-fr.html>

MongoDb : la base de données orienté documents, Thomas,

<http://code4fun.fr/mongodb-base-de-donnees-orientee-document/>

# Annexe

- Le workflow complet du Projet Générateur :



### • Notation de Crontab :

Le format de Crontab est ci-dessous :

```
mm hh jj MMM JJJ
```

- mm représente les minutes (de 0 à 59)
- hh représente l'heure (de 0 à 23)
- jj représente le numéro du jour du mois (de 1 à 31)
- MMM représente l'abréviation du nom du mois (jan, feb, ...) ou bien le numéro du mois (de 1 à 12)
- JJJ représente l'abréviation du nom du jour ou bien le numéro du jour dans la semaine :
  - 0 = Dimanche
  - 1 = Lundi
  - 2 = Mardi
  - ...
  - 6 = Samedi
  - 7 = Dimanche (représenté deux fois pour les deux types de semaine)

Pour chaque valeur numérique (mm, hh, jj, MMM, JJJ) les notations possibles sont :

- \* : à chaque unité (0, 1, 2, 3, 4...)
- 5,8 : les unités 5 et 8
- 2-5 : les unités de 2 à 5 (2, 3, 4, 5)
- \*/3 : toutes les 3 unités (0, 3, 6, 9...)
- 10-20/3 : toutes les 3 unités, entre la dixième et la vingtième (10, 13, 16, 19)

Si, sur la même ligne, le « *numéro du jour du mois* » et le « *jour de la semaine* » sont renseignés, alors **celery** exécutera la tâche quand **l'un** des champs correspond. Par exemple, la ligne suivante indique que la tâche doit être exécutée les vendredis **ainsi que** le 13 de chaque mois, à 00:00 du matin :

```
0 0 13 * 5
```

### • Le code de url.py sur le mini projet Reports

Nom du fichier : Url.py

```
url(r'^school/import/$',
    SchoolImportView.as_view(),
    name='school_import_base'),
url(r'^school/upload/$',
    SchoolUploadView.as_view(),
    name='school_upload_base'),
url(r'^school/validation/$',
    SchoolValidationView.as_view(),
    name='school_validation'),
url(r'^school/add_edit/$',
    SchoolAddOrEditView.as_view(),
    name='school_add_edit_base'),
```

- **Un exemple de entrée/sortie de vue schoolValidation :**

Entrée :

```
length:1
table[0][]:UTC Compiègne
table[0][]:2 - Écoles d'ingénieurs
table[0][]:3
table[0][]:blabla@utc.fr
table[0][]:456
table[0][]:emploi
table[0][]:837
table[0][]:
table[0][]:
table[0][]:True
```

Sortie :

```
[{"category": "success", "group": "success", "name": "success", "language": "warning",
"generator": "success", "diploma": "danger", "institution": "success", "specific": "success",
"email": "success", "flag": "success", "interface": "success", "msg_error": {"diploma": "Please
input all or id(s) of diploma.", "language": "Please input a language."} }]
```